Undergraduate Project Report
2012/13

Low resource autonomous object-seeking robot using computer vision

Name:	Liu Zhijiao
Programme:	QCAH3
Class:	2009215109
QM Student No.	090466298
BUPT Student No.	09212987
Supervisor:	Christopher Harte

Date 2013.5.13

Project ID QCAH3 Low resource autonomous object-seeking robot using computer vision

Table of Contents

Project ID QCAH3 Low resource autonomous object-seeking robot using computer vision

Abstract

Building a robot with low resource is the main mission of this project. Based on the required functions, the robot needs to see the world, distinguish the colours, avoid obstacles, move to the target object, and bring it back to the start point. To fulfil the tasks, the implementation mainly contains three parts, robot's body, vision and movement.

Several low cost electrical components are used to build the robot body. STM32 microcontroller serves as the brain of the robot to do the control and calculation work, and the OV7670 camera module works like the eyes of human beings. The body of the robot are also made up of ultrasonic module, motor module and the connection.

Using OV7670 to accomplish the vision part and see the world requires the board not only gets the data from the camera, but also do the edge detection and colour classification. In this part, it contains the learning and implementation of some basic machine learning theories to help the robot to distinguish the environment.

For the movement part, the robot can move with required distance and certain turning angle, and it also can avoid the obstacles on its way. The previous function is achieved by the programming of the motor module. The collision avoidance is realised by the ultrasonic module.

Most of the programming work is done with C, Python and MATLAB.

This robot fulfils the majority requirements of the project, while the bringing back mission hasn't been finished because of the practical problems.

Project ID QCAH3 Low resource autonomous object-seeking robot using computer vision

摘要

本项目的主要任务是用低成本元件制作机器人。基于需求，此机器人能够看到周围的环境，分辨视觉中的颜色，躲避障碍物，寻找目标物并取回至出发地。为了实现这些任务，项目实施主要涉及三部分，机器人的躯体、视觉和行动。
机器人的躯体有一些低成本电子元件构成。STM32微控制器作为机器人的大脑主要负责控制和计算的工作。OV7670摄像头模块则犹如人眼一般给予机器人视觉。机器人躯体也由超声波测距模块、电机模块和接线构成。
运用OV7670来完成视觉部分不仅要求核心板能够从摄像头获取图像数据，还需要其完成边缘检测和颜色分类的功能。这部分包括了对基础机器学习理论的学习和应用，以帮助机器人分辨周边环境。
对于行动这部分，机器人可以根据给定的距离和转角行动，同时它也能避开旅途中的障碍物。前一个功能是由对电机模块的编程而实现的，后一个防撞功能则是由超声波测距模块实现的。
本项目大多数编程工作由C、Python和MATLAB完成。
此机器人实现了项目要求的绝大多数功能，而由于实际环境问题，取回目标物的任务尚未完成。

Project ID QCAH3 Low resource autonomous object-seeking robot using computer vision

Chapter 1: Introduction

1.1 Motivation

Artificial Intelligence (AI) and Machine Learning become more and more popular and widely used in different areas in these years, especially as the electronic components' price steadily dropping down and the prosperity of robotic research. AI algorithms used in the industry deal with enormous amounts of data to serve to the research and commercial demands. However, implementing AI into limited resource robot project requires finding the balance between processing rate and accuracy, which is a meaningful project to understand the AI algorithms and implement the machine learning theories in practical.

1.2 Purpose

The main purpose of this project is to learn and apply the operating principles of different components, computer vision and machine learning theories to implement them to build an autonomous target-seeking robot. This robot should not only find the target object with the OV7670 camera, but also can move towards to the object and bring it back to the start position. It is a combination of hardware and software. Programming, simulation and testing are the most significant parts of this project. What's more, implementing machine learning on this robot can help people get a better understanding of the edge detection and color classification, and get familiar with computer vision.

1.3 Realised functionalities

The robot achieves to see the world and move according to the requirements. In the vision part, two processes are finished. One is the STM32 core board sends signal to the OV7670 camera to trigger it to collect image data. The other is the color detection process, which requires the core board accomplishing the edge detection with Laplace operator and color classification with Naive Bayes classifier. In the movement part, three steps are needed. Firstly, the cooperation of ultrasonic module and motor module ensures the robot can realize the basic movement, like moving forward, backward, turning left, right, and avoiding obstacles. Secondly, based on the results of the image processing, the robot will adjust its moving direction to move towards to the target object. Thirdly, after the robot find the target and collect it, it will calculate the return path and try to bring the collected object back to the starting point.

The program running on the robot is written in C with KEIL 4.14. The captured image and color classification are processed in MATLAB to show the edges of the colors and train the data set to get the relevant probabilities. Demonstration and simulation mostly use Python to show the captured image composed by RGB pixels and fulfil the showing of the algorithms.

1.4 Structure

Chapter 1 introduces the motivation, the purpose, and the realized functions of this project and the structure of this report.

Chapter 2 gives the background knowledge and fundamental theories mentioned in this project. It contains three parts, hardware, software, and computer vision. Firstly, it presents brief introductions of STM32 core board, OV7670 camera, ultrasonic module and motor module used on the robot from architecture, working principle etc. aspects. Secondly, the mainly used softwares are introduced, KEIL 4.14 for STM32 programming, Python 2.7 for display and MATLAB for data processing. At the end of this chapter, the core methods used for computer vision in this project are presented, which are edge detection and colour classification.

Chapter 3 shows 6 steps of the whole design and implementation process. The basic communication between STM32 core board and PC is built at first through the serial. Then come to the image capture step. The signal sent from the PC triggers the board to collect one frame image data from the OV7670 camera and send those data to the PC. A Python program will use those data to output the one frame image. After that is the image processing step. This step focuses on the implementation of edge detection and colour classification to help the robot to distinguish the world by different colors. The next two steps are about using ultrasonic module to measure the distance and controlling the motor to ensure the robot can run and turn. The last one is the algorithm. Here introduces two main algorithms for object-seeking and return path calculation.

Chapter 4 indicates and discusses the specific results in this project, especially those testing results.

Chapter 5 concludes all the work and mentions the future work for further improvement.

Chapter 2: Background

This chapter includes three aspects of the background used in this project, which are hardware, software and computer vision. All the needed information of boards and modules is in the hardware aspect. Then the three mainly used softwares, KEIL, Python, and MATLAB, are briefly introduced here. The last aspect, computer vision, provides more significant background information of edge detection and colour classification.

2.1 Hardware

2.1.1 STM32 Core Board

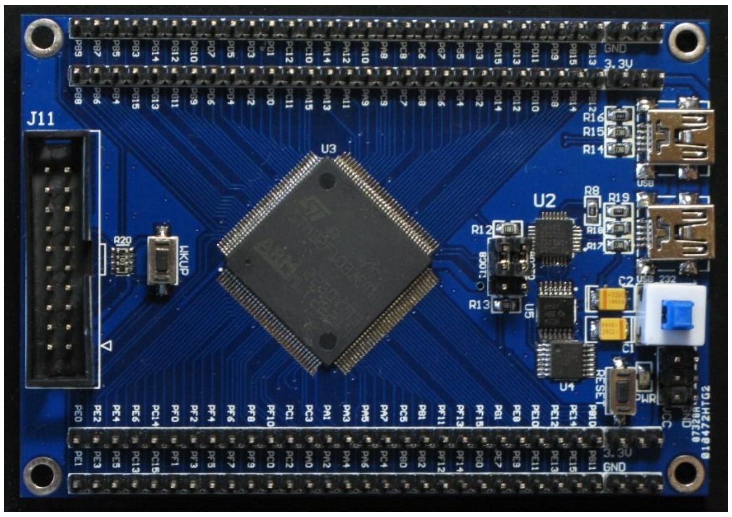


Figure 1: STM32F103ZET6

STM32F103ZET6 is used in this project(shown in Figure 1), which incorporates the Cortex-M3 32-bit core operating at 72 MHz, 512 Kbytes Flash, 64 Kbytes SRAM and 144 Pins for I/Os and peripherals. This board offers four 16-bit timers, two I2Cs, five USARTs, an USB, and other standard and advanced communication interfaces. [2]

Architecture

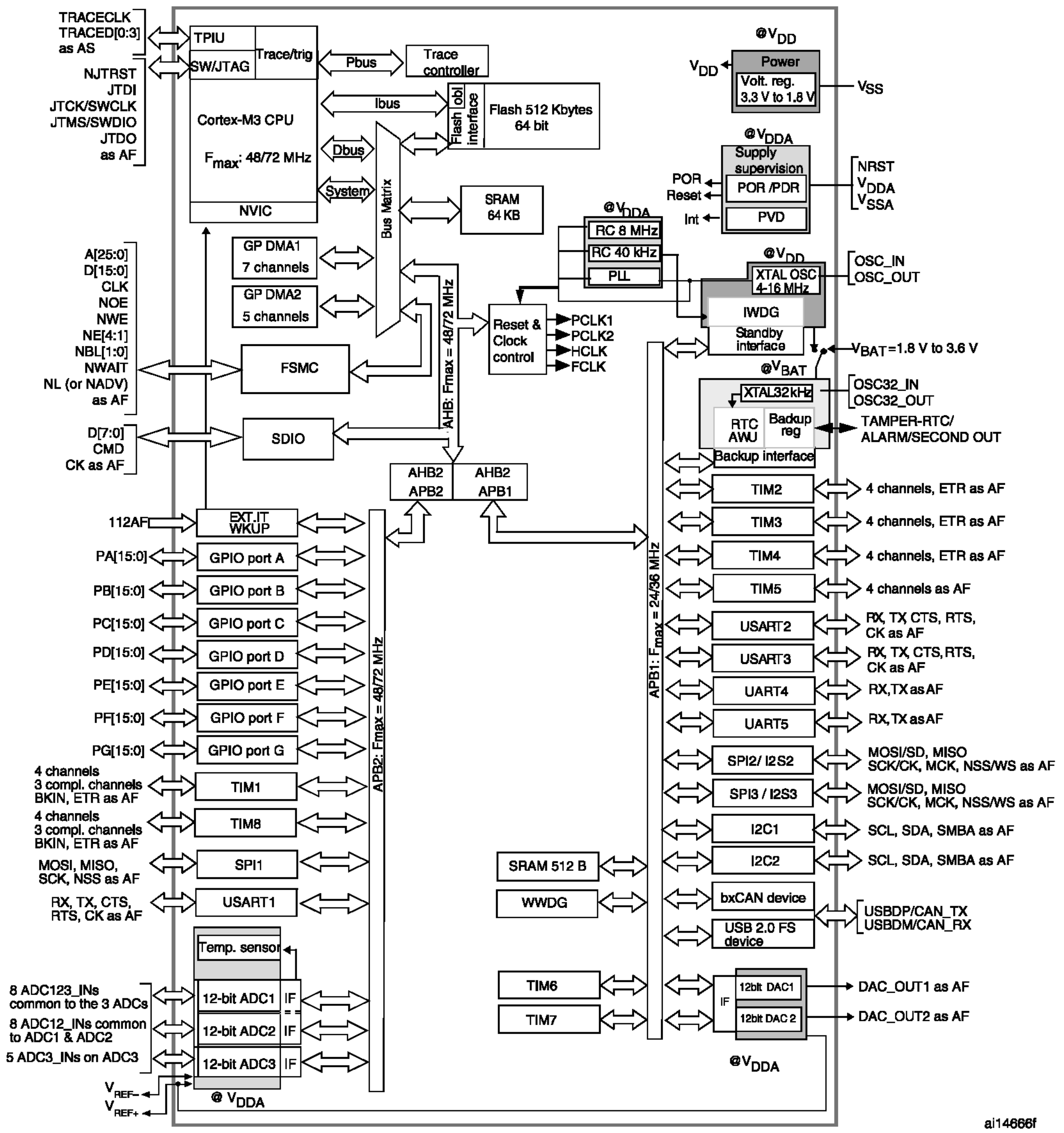


Figure 2: Circuit diagram of STM32F103xE

The circuit diagram of STM32F103xE, shown in Figure 2, shows the whole architecture of this core board. Besides the 72MHz Cortex-M3 CPU and 512 Kbytes Flash, there are many useful parts for this project, such as GPIO, FSMC, NVIC, JTAG, UART, etc. [3]

GPIO (General Purpose Input Output) are generic pins on the board. Users can control these input and output pins at run time. They are not defined with special purpose, which is useful to not only define the input output ports but also may work as other pins with specific functions.

FSMC (Flexible Static Memory Controller) is a specific storing controlling mechanism of STM32 series suffix with xC、xD and xE, whose Flash is more than 256 Kbytes. Flexible means that FSMC can response corresponding controlling signal to meet the requirements of signal speed by setting the register with special functions.

JTAG (Joint Test Action Group) is mainly used for debug, which offers the operations like break pointing and single stepping debug.

NVIC (Nested Vectored Interrupt Controller) handles low-latency exception and interrupt. It supports 240 dynamically reprioritizable interrupts. Each of them has 256 levels of priority. The NVIC and the Cortex-M3 CPU are closely coupled. That helps to improve the processing efficiency of low-latency interrupt and late arriving interrupt. It supports stacked (nested) interrupts as well.

UART (Universal Asynchronous Receiver/Transmitter) is commonly used in microcontrollers as an integrated circuit used for serial communications between a computer and peripheral devices.

STM32F10x Standard Peripherals Library

The STM32F10x Standard Peripherals Library contains the drivers for all of the standard peripherals. This library is a complete firmware package with a collection of routines, data structures and macros covering the features of STM32 peripherals. Description and examples are included in this library, which helps users to quick start without in-depth learning of each peripheral's specifications. It can save coding time and reduce the costs of application development and integration. [4]

2.1.2 OV7670 Camera Module

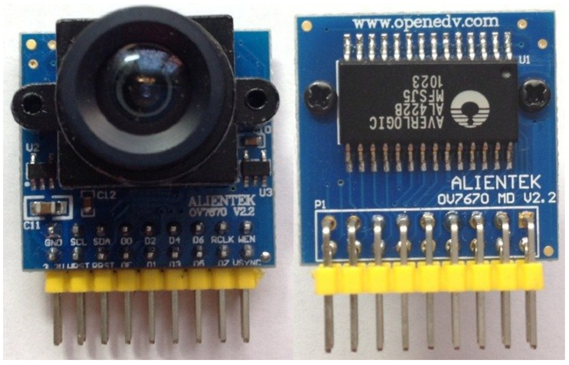


Figure 3: 2.1.2 OV7670 Camera Module

OV7670 Camera, produced by with AL422B FIFO does the job of collecting data in this project. It can operate at 30 frames per second (fps) in VGA. Users can control the image quality, data format, etc. With low noise, low cost, low power consumption, high integration, wide dynamic range and good low-light performance, OV7670 is not only cheap, but also powerful, which is fit the title – low resource.

Functional block diagram

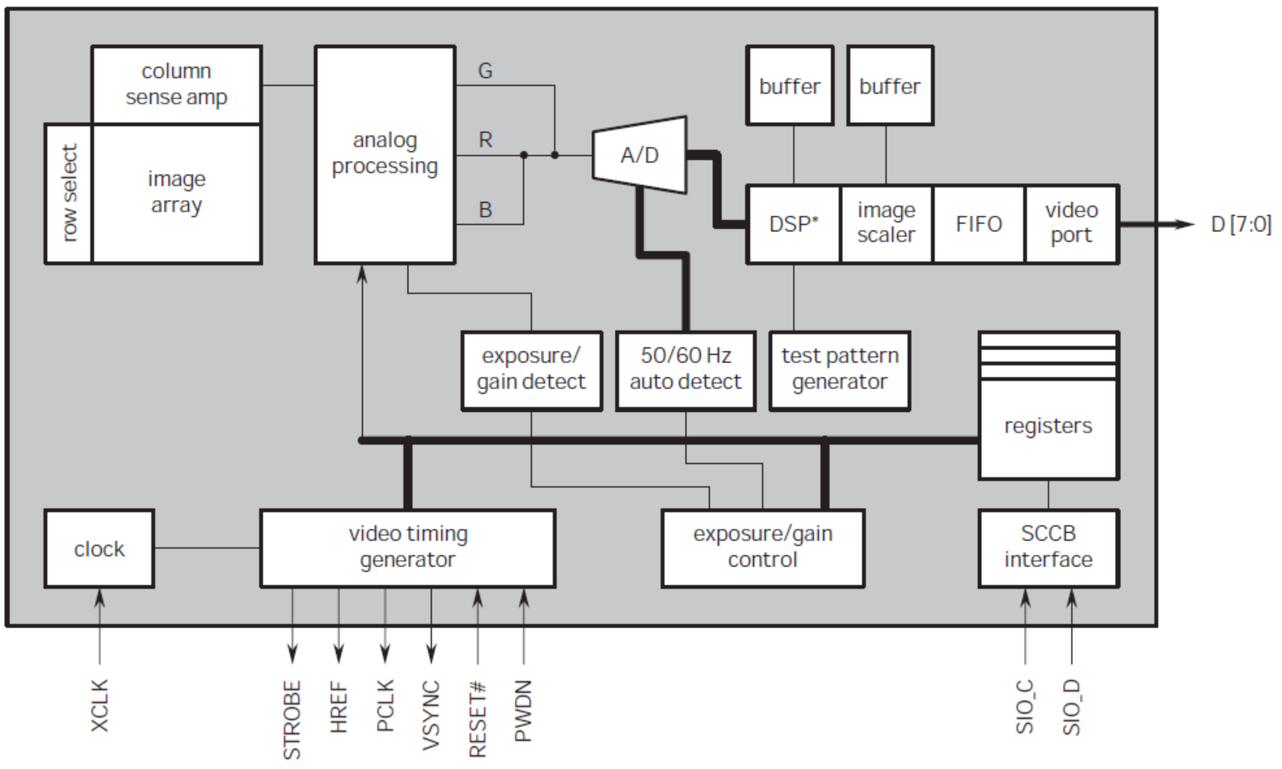


Figure 4: OV7670 functional block diagram

SCCB (Serial Camera Control Bus) is defined by OmniVision Technologies, Inc. for controlling most of the functions in OmniVisions family of CAMERACHIPTM sensors. In OV7670, the SCCB operates in a modified 2-wire serial mode. SCCB also can be set up to configure the registers.

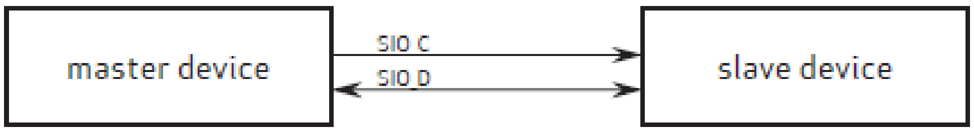


Figure 5: SCCB - Master and Slave

The 2-wire serial mode allows a master device to interact only one slave device shown in Figure 5.

This requires one of the following two master control methods in order to enable the SCCB communication. Number one, the master device must be able to maintain a tri-state data line. Number two, if the master cannot maintain a tri-state data line, it can drive the data line either high or low and note the transition to assert communications with the slave CAMERACHIP sensor.

DSP (Digital Signal Processor) works as a colour space converter that controls the interpolation from raw data to RGB or YUV. It also supports white/black correction, which relates to white balance. White balance technology is helpful to display or recover the real color of white area no matter the light source. This function of DSP solves the problem of low-light or high-light environment to ensure that the following color detection step could be implemented correctly.

2.1.3 Ultrasonic Module

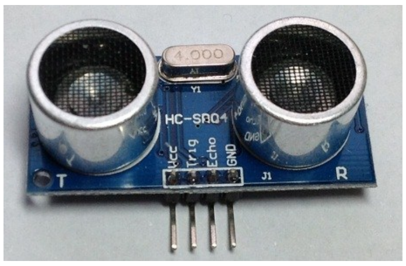


Figure 6: HC-SR04 Ultrasonic Module

HC-SR04 (Figure 6) is a stable and accurate ultrasonic module. It is widely used in robotic, obstacle avoidance, distance measurement, parking detection, etc. It is powered by DC 5V. Its detecting range can reach 450cm and its accuracy can be 0.2cm. There are four pins on it, as VCC, Trig, Echo, and GND. Use Trig pin with at least 10us high level signal to trigger the detection. Then the module automatically sends eight 40Hz square waves and checks whether there is a response signal. If there is a return signal back to the module, that will cause a high level on the Echo pin. The high level lasting time is the time between the sending and receiving. The sequence chart of the whole process is shown in Figure 7.

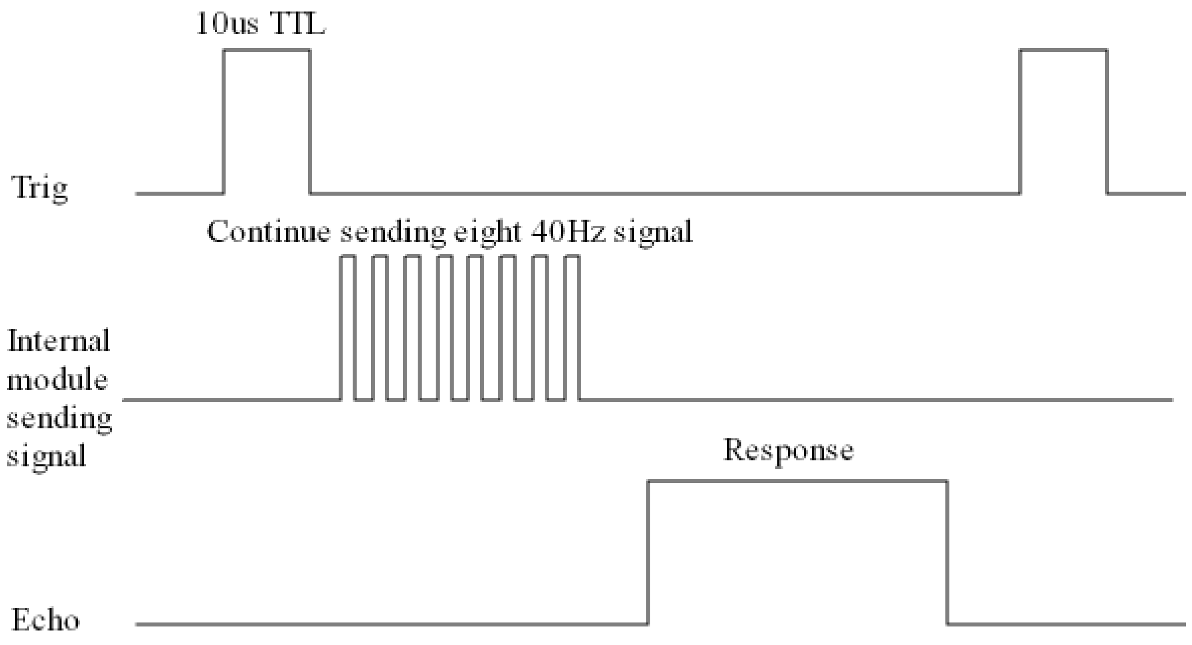

10us TTL
Trig
Continue sending eight 40Hz signal
Internal module sending signal
Response
Echo

Figure 7: HC-SR04 sequence chart

2.1.4 Motor Module

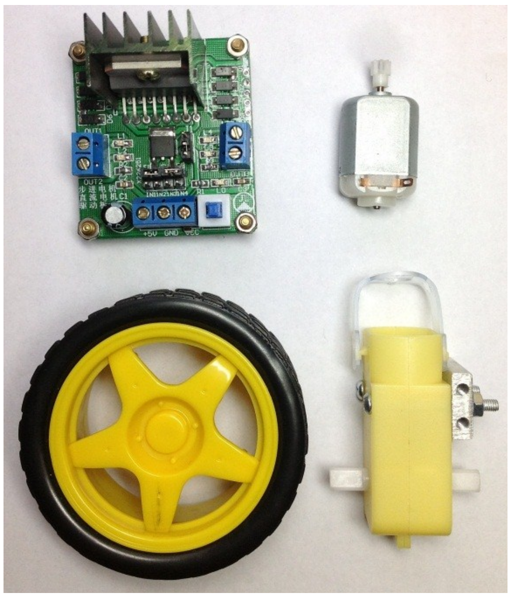


Figure 8: L298N driver board, motor, and wheel

L298N driver board is manufactured by STMicroelectronics. Users can use standard TTL logic levels to control this dual full-bridge driver board to drive DC and stepping motors. Two inputs are provided to enable or disable the relevant outputs independently. The outputs usually connect with motors' two pins to control them reversing or not. Its works with +5V to +35V and it also can provide +5V to +7V to support the logic part and external device, such as the core board and other modules.

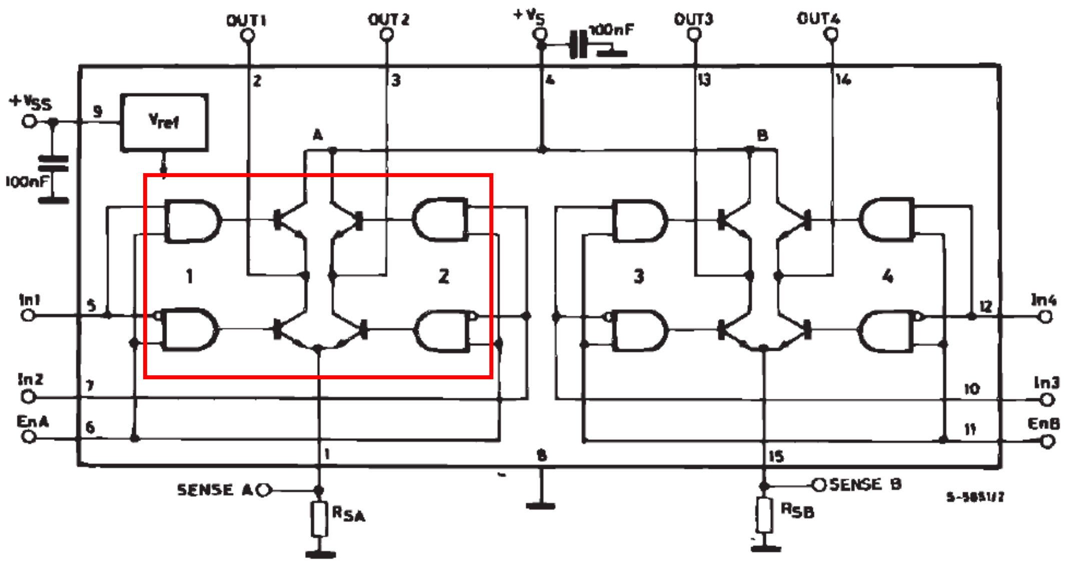


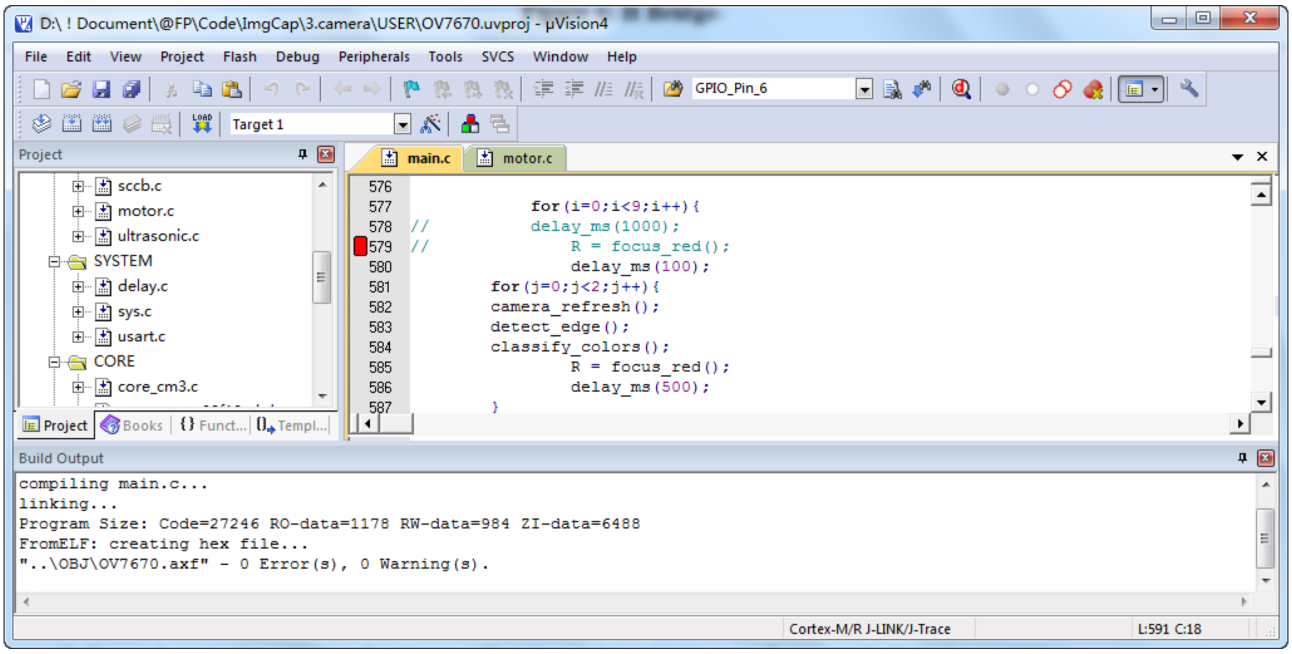
Figure 9: L298 Block Diagram

As it is shown in Figure 9, L298N uses Two H Bridges to control the output voltage.

An H bridge (in the red block) enables a voltage that can be applied across a load in either direction. This circuit is commonly used in robotics and other applications to ensure DC motors to run forwards and backwards. IN1 and IN2 control one wheel, and the other two INs control the other wheel. The high/low levels of different inputs will cause different performance of the motors. For example, IN1 and IN2 control wheel A. If IN1 is high level and IN2 is low level, the wheel will run forwards, and if IN2 is high and IN1 is low, the wheel will run backwards. When IN1 and IN3 are high level, the motor will run forwards. When IN2 and IN4 are high level, the motor will run backwards. As there are four inputs, this driver board can control 2 wheels simultaneously.

2.2 Software

2.2.1 Keil uVision v4.14



```
                for(i=0;i<9;i++){
//              delay_ms(1000);
//                      R = focus_red();
                        delay_ms(100);
        for(j=0;j<2;j++){
        camera_refresh();
        detect_edge();
        classify_colors();
                        R = focus_red();
                        delay_ms(500);
        }
```

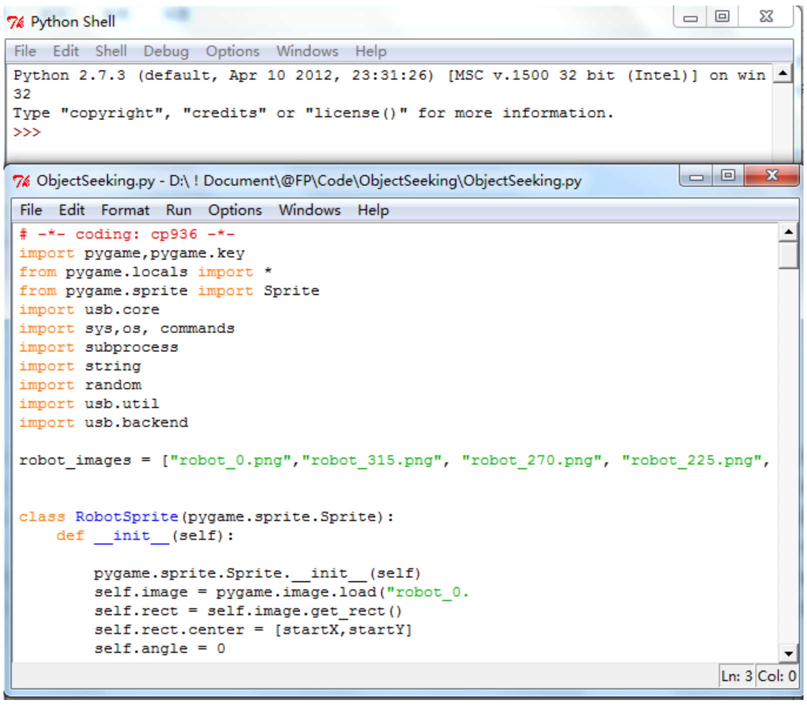
```
compiling main.c...
linking...
Program Size: Code=27246 RO-data=1178 RW-data=984 ZI-data=6488
FromELF: creating hex file...
"..\OBJ\OV7670.axf" - 0 Error(s), 0 Warning(s).
```

Figure 10: Keil uVision v4.14

Keil was founded in 1982 and implemented the first C compiler for the 8051 microcontroller. It provides a extensive range of development tools like ANSI C compiler, macro assemblers, debuggers and simulators, linkers, IDE, library managers, real-time operating systems and evaluation boards for 8051, ARM, etc.

The µVision IDE from Keil provides a powerful development environment that combines coding, program debugging, simulating, compiling, and project management in one program. This platform is easy to use and can help users quick start to create embedded programs. The µVision editor and debugger are also integrated in one single program that offers all-in-one embedded project development environment.

2.2.2 Python 2.7


Python Shell
File Edit Shell Debug Options Windows Help
ObjectSeeking.py - D:\ ! Document\@FP\Code\ObjectSeeking\ObjectSeeking.py
File Edit Format Run Options Windows Help
Ln: 3 Col: 0

```
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>>
```

```
# -*- coding: cp936 -*-
import pygame,pygame.key
from pygame.locals import *
from pygame.sprite import Sprite
import usb.core
import sys,os, commands
import subprocess
import string
import random
import usb.util
import usb.backend

robot_images = ["robot_0.png","robot_315.png", "robot_270.png", "robot_225.png",


class RobotSprite(pygame.sprite.Sprite):
    def __init__(self):

        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("robot_0.
        self.rect = self.image.get_rect()
        self.rect.center = [startX,startY]
        self.angle = 0
```

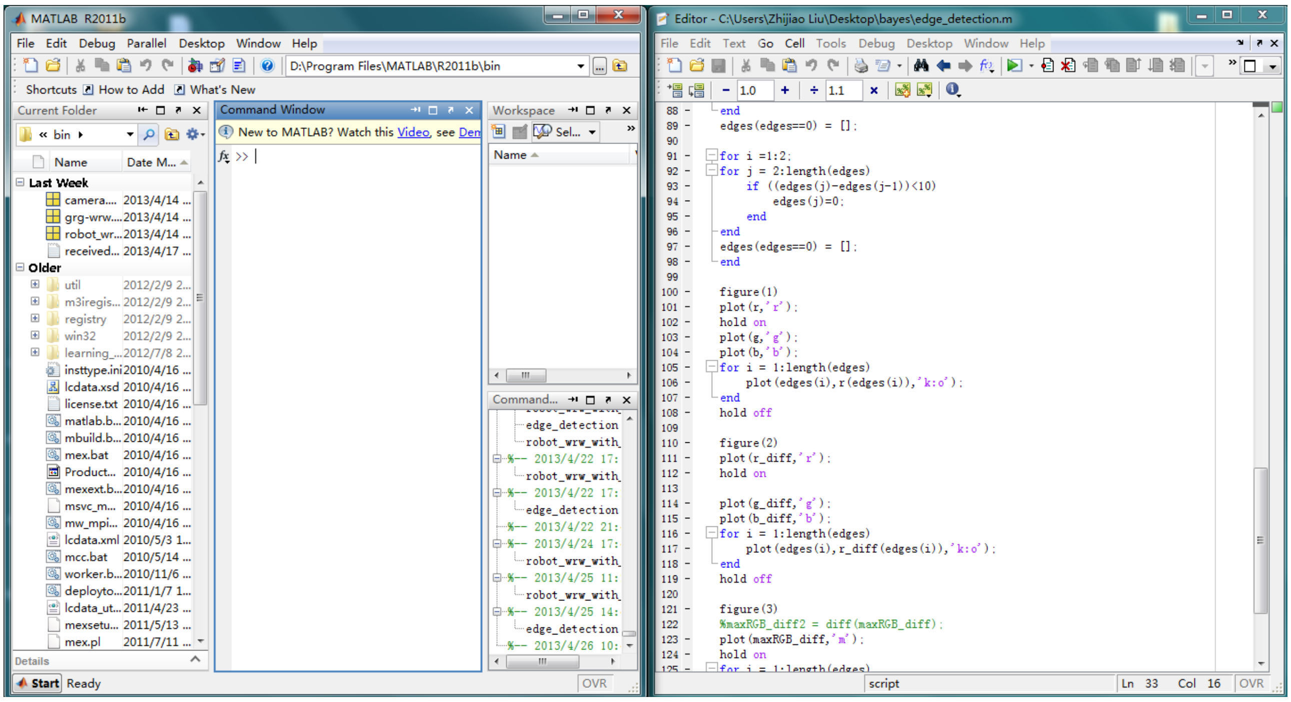
Figure 11: Python 2.7.3 in Windows 7

Python is a widely used multi-platform high-level programming language. Its syntax helps the programmers to accomplish similar functions with fewer lines of code than would that in C or other languages. Python, as a dynamic type system, manages the memory automatically, and has a great and wide-ranging standard library. This dynamic language, Python, often works as a scripting language and non-scripting contexts. With third-party tools, the code can be wrapped into separate executable programs. Python interpreters can work on different operating systems, Windows, Linux, MAC, etc. [5]

PyGame

PyGame is a cross-platform module of Python, which is mainly used for writing video games. It contains graphics libraries and sound libraries to improve the game programming. [6] PyGame is built on the Simple Direct Media Layer (SDL) library, which allows real-time development rather than those mechanisms in C. Most complicated functions in games are mainly related to the graphic parts. While in Python, it assumes that those can be distracted from the game logic, and use high-level languages to build the game. PyGame is portable and can works on almost every kind of operating systems. What’s more, PyGame is free and open source.

2.2.3 MATLAB R2011b


MATLAB R2011b
File Edit Debug Parallel Desktop Window Help
D:\Program Files\MATLAB\R2011b\bin
Shortcuts How to Add What's New
Current Folder
Command Window
New to MATLAB? Watch this Video, see Demos
Workspace
Command...
Start Ready
Editor - C:\Users\Zhijiao Liu\Desktop\bayes\edge_detection.m
File Edit Text Go Cell Tools Debug Desktop Window Help
script
Ln 33 Col 16 OVR

```
    end
    edges(edges==0) = [];

    for i =1:2;
    for j = 2:length(edges)
        if ((edges(j)-edges(j-1))<10)
            edges(j)=0;
        end
    end
    edges(edges==0) = [];
    end

    figure(1)
    plot(r,'r');
    hold on
    plot(g,'g');
    plot(b,'b');
    for i = 1:length(edges)
        plot(edges(i),r(edges(i)),'k:o');
    end
    hold off

    figure(2)
    plot(r_diff,'r');
    hold on

    plot(g_diff,'g');
    plot(b_diff,'b');
    for i = 1:length(edges)
        plot(edges(i),r_diff(edges(i)),'k:o');
    end
    hold off

    figure(3)
    %maxRGB_diff2 = diff(maxRGB_diff);
    plot(maxRGB_diff,'m');
    hold on
    for i = 1:length(edges)
```

Figure 12: MATLAB R2011b

MATLAB (matrix laboratory) is a powerful numerical computing environment, developed by MathWorks. As a 4th generation programming language, MATLAB allows users to manipulate matrixes, plot functions and data, create user interfaces, implement a wide range of algorithms, and interface with the programs written in other languages. While MATLAB works basically for numerical computing, optional toolboxes, like MuPAD symbolic engine, Simulink, etc., improve symbolic computing capabilities and graphical multi-domain simulations.

2.3 Computer Vision

Computer vision is a front area that tries to replicate the capabilities of human vision to the electronics by distinguishing and understanding the images. This field contains the research and development of obtaining, processing, analyzing, and understanding images. That will output numerical information or data to help the electronics to learn the environment and make decisions. With the help of statistics, learning theory and the knowledge in other areas, this image understanding process can handle numerical information from image data to construct learning and training models. [7]

It is widely used and implemented in artificial intelligence and robotic. There is a huge overlap between computer vision and machine vision. Computer vision covers the essential technologies that automatically analyze image data, while Machine vision, combining automatic image analysis with other methods of machine learning, concerns the process of guiding the robot, which is more commonly applied in industry.

2.3.1 Edge Detection

Edge detection refers to those mathematical approaches to identify the discontinue points in a digital images. These points map the sharply changed brightness in the image to sudden change of color components. Edge detection is vital in image processing, machine vision and computer vision, especially used for feature detection and feature extraction.

The methods of edge detection can be roughly categorized into search-based and zero-crossing based. The search-based methods use the measurement of edge strength to detect edges. Firstly, get the first-order derivative, and then look for local directional maxima of the first-order derivative to estimate the edge's local orientation. The zero-crossing based methods use zero-crossing points in a second-order derivative to search for edges, regularly with the Laplacian.

The Laplacian Δ of f is defined as,

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f \qquad (1)$$

It also can be expressed as the sum of all the unmixed second partial derivatives in the Cartesian coordinate,

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \qquad (2)$$

In one-dimension, it will be simplified as,

$$\Delta f = \frac{d^2 f}{dx^2} \qquad (3)$$

Generally, after compute the first-order derivative, apply a threshold to filter the image points with low values. The higher the threshold is, the less the edges will be found, and that will improve the ability of being less vulnerable to noise and other unrelated terms in the image. While a high threshold may fail to detect subtle edges that only with a little change in brightness. [8]

2.3.2 Colour classification

In machine learning, classification is the process of deciding an input value belongs to which kind of categories based on training data containing known category. The individual inputs are evaluated by certain quantifiable properties, such as "Red", "Green", "White" "Black" or others, for colour type. Classification works as supervised learning process, i.e. learning from the correctly-identified instances. In actual implementation, it is named as "classifier" that maps input data to a given class based on the mathematical function.

Bayesian approach is common-used in classification, which calculates the probability of one belongs to each kind of the given categories to accomplish the classification. [9] To implement this approach, a bulky training data set is needed. The core function of this approach is Bayes' theorem:

For events A and B, where P(B) $\neq$ 0,

$$\mathrm{P}(A \mid B) = \frac{\mathrm{P}(B \mid A) \cdot \mathrm{P}(A)}{\mathrm{P}(B)} \tag{4}$$

Which also infers that posterior is proportional to prior times likelihood:

$$\mathrm{P}(A \mid B) \propto \mathrm{P}(B \mid A) \cdot \mathrm{P}(A) \tag{5}$$

Chapter 3: Design and Implementation

3.1 The Whole System and Environment

3.1.1 Design the whole system

Before design and implement each module of the robot, the structure of the whole system should be designed at first.

In the real world, the surrounding consists of 3D objects, while, as the specification mentions, the low resource robot cannot reach that accuracy like human vision. What's more, the real world contains infinite information. That is impossible to process such huge data only with a small single cheap microcontroller. To meet the requirements and make the project implementable, using 2D instead is a better solution, and now the robot can handle the data and algorithms with its limited memory.

According to the 4 tasks in the specification, there are two parts of this project, computer-based and robot-based. The structure of the whole system is shown in Figure 13.

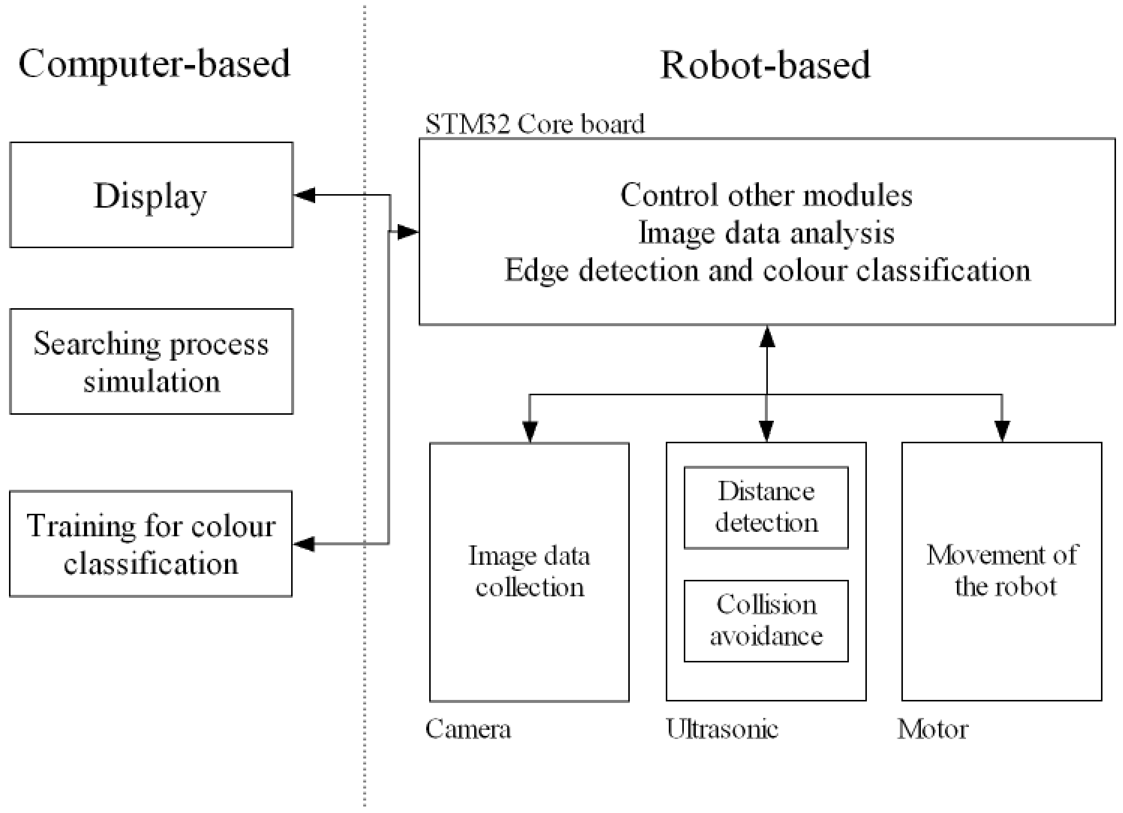


Figure 13: The structure of the whole system

3.1.2 Environment Design and Construction

The task is that finding the target and collecting it back to the starting position. What is the target for the robot? How can it distinguish the target from the background? To answer these two questions, different colours representing different objects in the environment will simply the task. Four colours are used to represent different things. Red is target's colour, and black is obstacle. The background is full of green and white colour block. The imitator of the environment is shown in Figure 14, and the environment in the real world is shown in Figure 15.

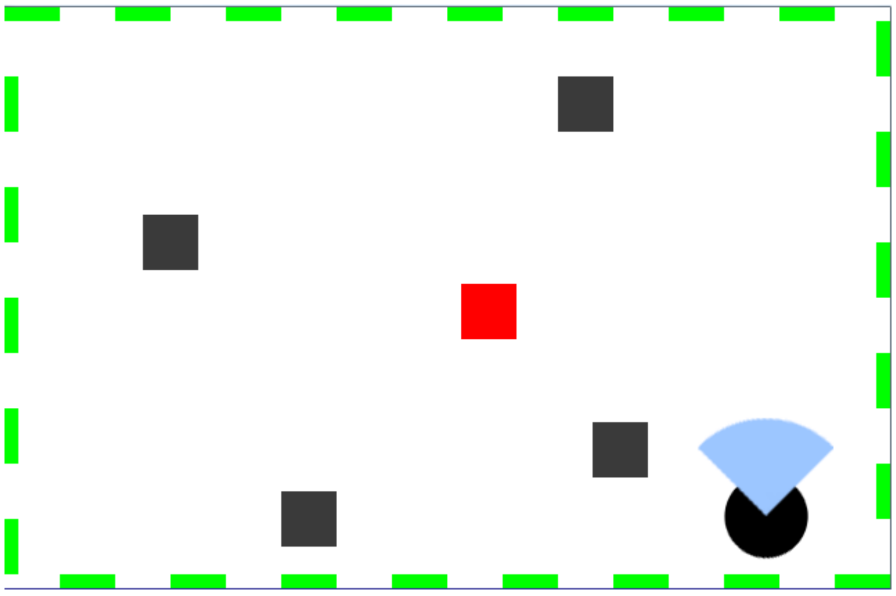


Figure 14: The imitator of the environment

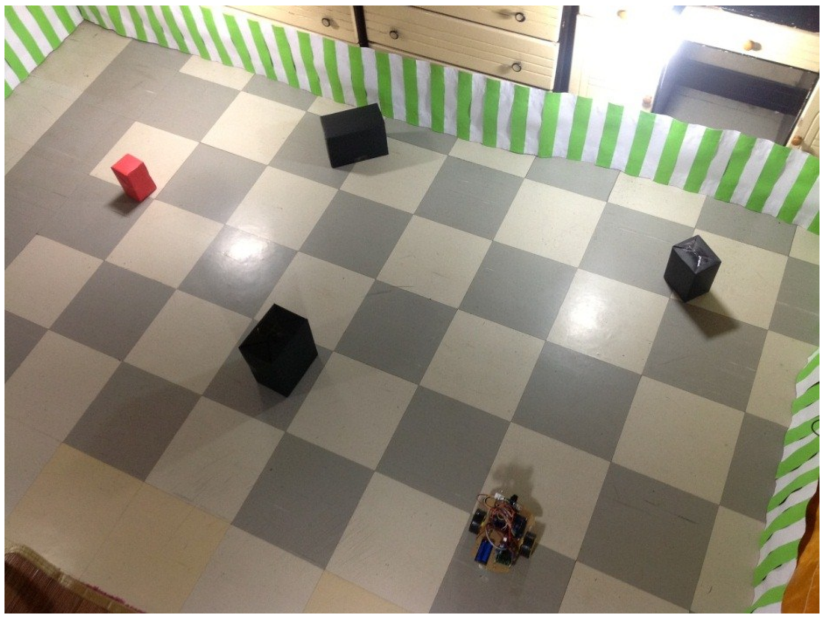
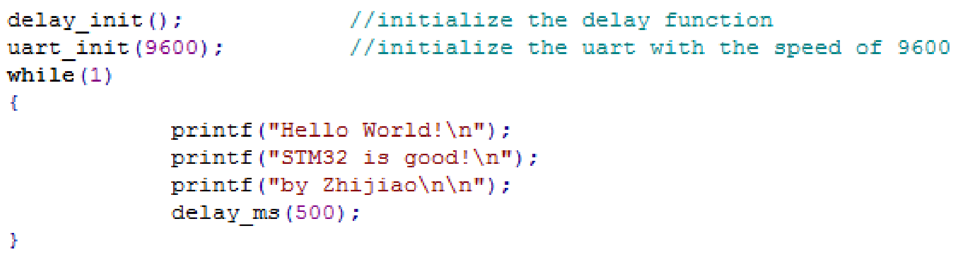


Figure 15: The real environment

3.2 Basic Communication between Board and PC

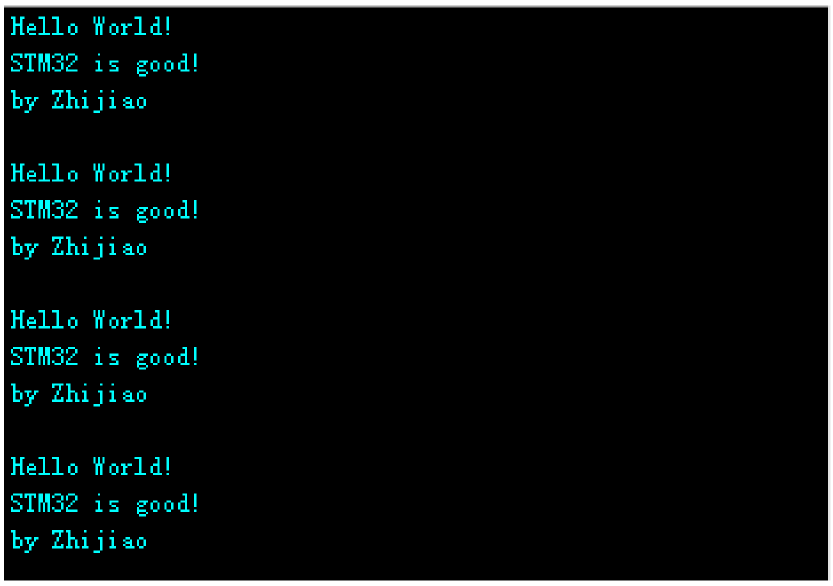
3.2.1 UART

Because the core board needs to send the image data to PC, the basic communication is important. I use UART to accomplish this task. UART sends data through the serial of the computer. As the simple code shows (in Figure 16), it just needs to initialize the delay function and uart function to accomplish the task that the computer always listens to the specific com port and receives the data sent from the core board. Figure 17 shows the result of this basic communication between computer and core board.



```
delay_init();              //initialize the delay function
uart_init(9600);           //initialize the uart with the speed of 9600
while(1)
{
            printf("Hello World!\n");
            printf("STM32 is good!\n");
            printf("by Zhijiao\n\n");
            delay_ms(500);
}
```

Figure 16: The code of sending data with UART

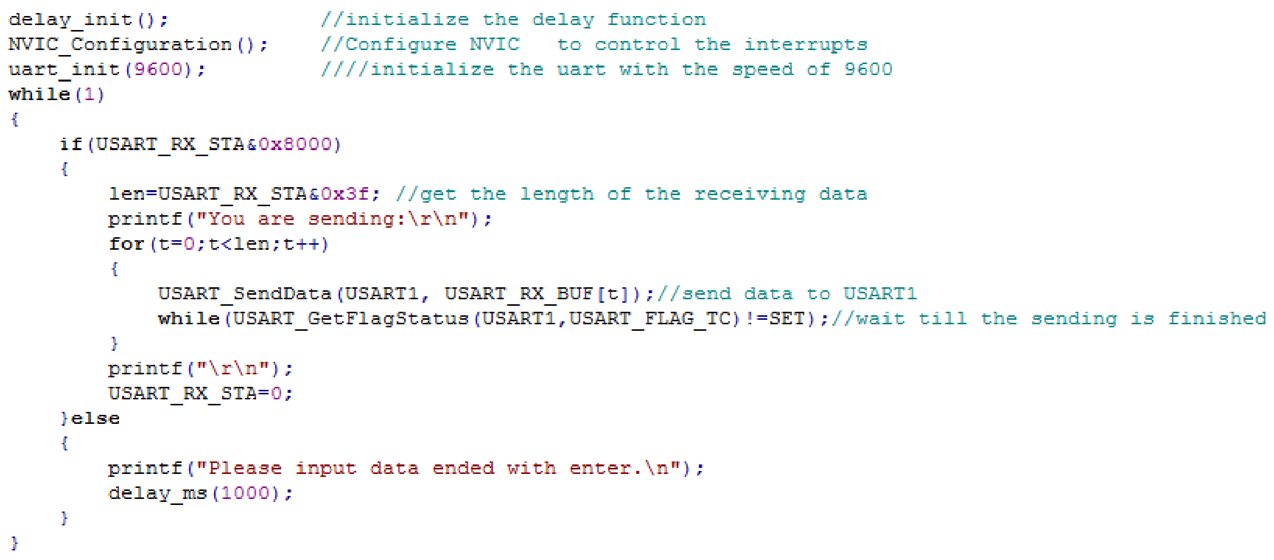


```
Hello World!
STM32 is good!
by Zhijiao

Hello World!
STM32 is good!
by Zhijiao

Hello World!
STM32 is good!
by Zhijiao

Hello World!
STM32 is good!
by Zhijiao
```

Figure 17: Computer receives the info from UART

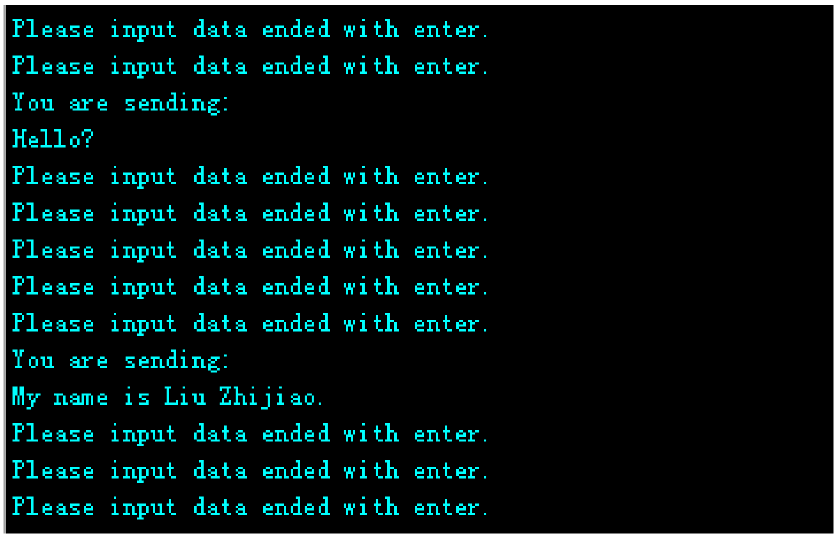
3.2.2 USART

USART is another way to send data through the serial. As different from UART, it can not only send but also receive data from the computer. Besides delay and uart, it needs to configure NVIC to control the interrupts that maintain the output of those info send from the computer. The code in main.c is shown in Figure 18, and the output is in Figure 19.



```
delay_init();             //initialize the delay function
NVIC_Configuration();     //Configure NVIC   to control the interrupts
uart_init(9600);          ////initialize the uart with the speed of 9600
while(1)
{
    if(USART_RX_STA&0x8000)
    {
        len=USART_RX_STA&0x3f; //get the length of the receiving data
        printf("You are sending:\r\n");
        for(t=0;t<len;t++)
        {
            USART_SendData(USART1, USART_RX_BUF[t]);//send data to USART1
            while(USART_GetFlagStatus(USART1,USART_FLAG_TC)!=SET);//wait till the sending is finished
        }
        printf("\r\n");
        USART_RX_STA=0;
    }else
    {
        printf("Please input data ended with enter.\n");
        delay_ms(1000);
    }
}
```

Figure 18: The code of sending and receiving data with USART



```
Please input data ended with enter.
Please input data ended with enter.
You are sending:
Hello?
Please input data ended with enter.
Please input data ended with enter.
Please input data ended with enter.
Please input data ended with enter.
Please input data ended with enter.
You are sending:
My name is Liu Zhijiao.
Please input data ended with enter.
Please input data ended with enter.
Please input data ended with enter.
```

Figure 19: The output of sending and receiving with USART

3.3 Image Capture

3.3.1 Get one frame image

After define relevant pins and set up timers and external interrupts, the process that OV7670 get one frame image is as follows:

OV7670 stores image data:

a) OV7670 waits for the sync signal

b) Reset the writing pointer of FIFO

c) Enable FIFO

d) Wait for the second sync signal

e) Write data

f) Disenable the FIFO

Read image data form OV7670:

a) Reset the reading pointer of FIFO

b) Give FIFO_RCLK(FIFO reading clock)

c) Read the high bit of the first pixel

d) Give FIFO_RCLK

e) Read the low bit of the first pixel

f) Give FIFO_RCLK

g) Repeat previous steps for getting the bits of all pixels

h) Finish the whole image reading process

The following Figure 20 shows the sequence chart of OV7670 to get image data. HREF, abbreviate for horizontal reference, indicates the process of sending a specific horizontal line of pixels. It is also named as the horizontal synchronization signal. VSYNC is another synchronization signal, which tells the state that whether the camera is sending image data. These two help the receiver, here is the core board, to keep in phase with the camera.

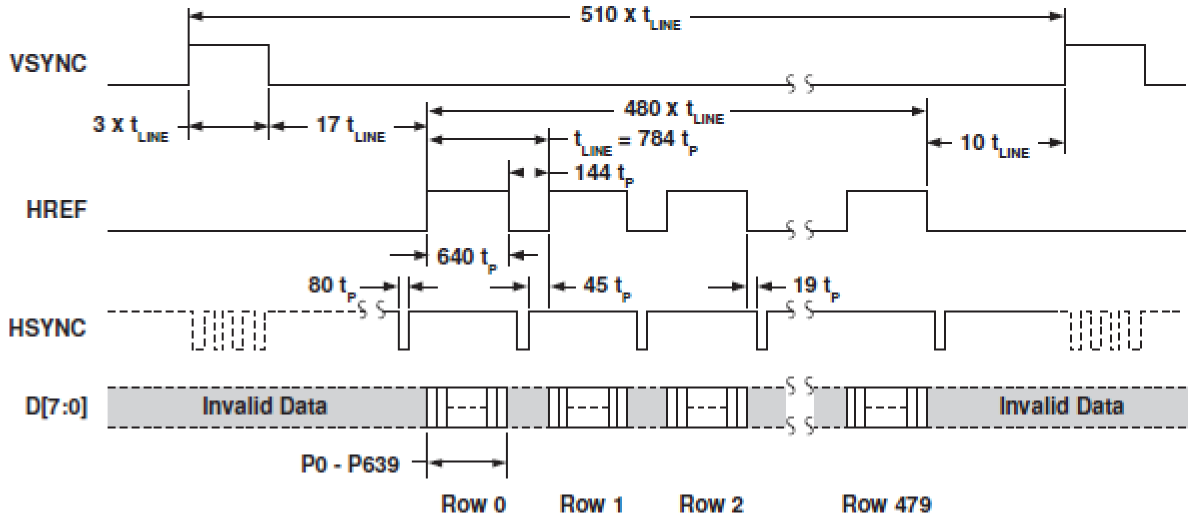


Figure 20: Sequence chart of OV7670

Understanding the working principles of OV7670 is not enough. The camera can only be chosen as sending image data in RGB565 mode, while the computer needs RGB888 mode to show the image that people can see.

RGB565 mode means that 5 bits store R parameter, 6 bits store G parameter, and another 5 bits store B parameter. This mode only uses 16 bits to store one pixel, which will save more pixels than the RGB888 mode when they have same memory. 16 bits can compose 65,536 kinds of colors. Green component gets an extra bit, which will have 32 more levels of intensity than other two components.

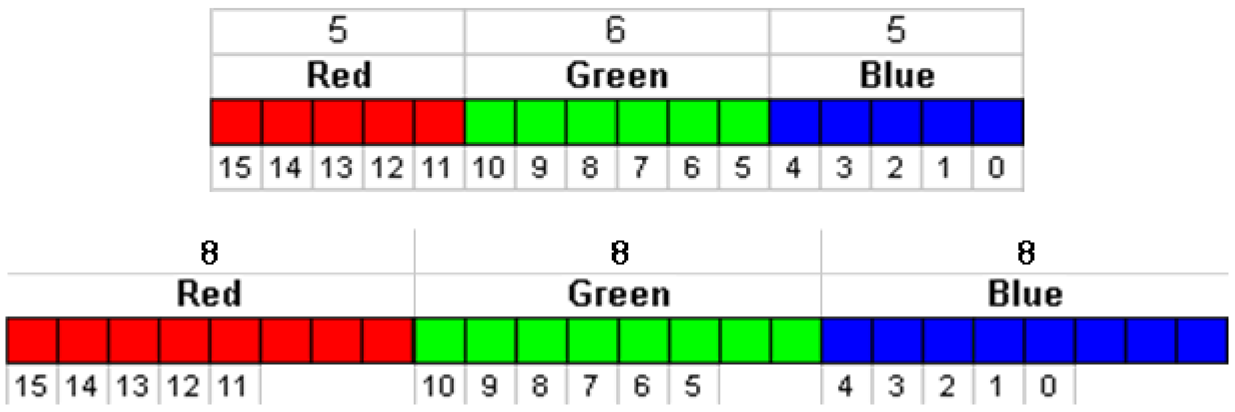


Figure 21: RGB565 and RGB888

To change RGB565 to RGB888 (Figure 21) requires adding extra bits to the three colour components. So, retain the most significant bits and then add 0s to complement the least significant bits, although that will decrease the precision.

One problem happens in the implementation of the conversion from RGB565 to RGB888. The converted image has a totally wrong colour, but when it is in the mode of gray scale, the image is correct. After analysis and several tests, that is caused by the RGB parameters not saving the right pixel's information. The variable type, u16, 16 bit, replaces the u8, 8bit, to store the colour parameters of the image and finally it works well.

After the core board receives the image data and turn those into RGB888 mode, it will wait for the external interrupt that the computer gives to trigger the updating process and send one frame of the image through the serial port.

3.3.2 Show the captured image

After the PC get the image data and save it as "received.txt", the python program will read all the RGB info and draw the whole picture as printing pixels one by one. However, it is not as simple as it seems. Two problems impact a great deal to receive one frame image data successfully.

Number one, the sync problem causes the pixels of the received image disordered (shown in Figure 22). This sync problem of the captured image leads to a gibberish pixels. The camera is straight running, but the command of capturing is not synchronized with the one frame image. This problem is solved by setting up a startup command. Only after the board gets the command sent from the serial, usually as sending "1" or other ASCII code, it starts to send the one frame data. That ensures that every time of the capture, the board sends an intact ordered image data to the computer and finally solves the sync problem.

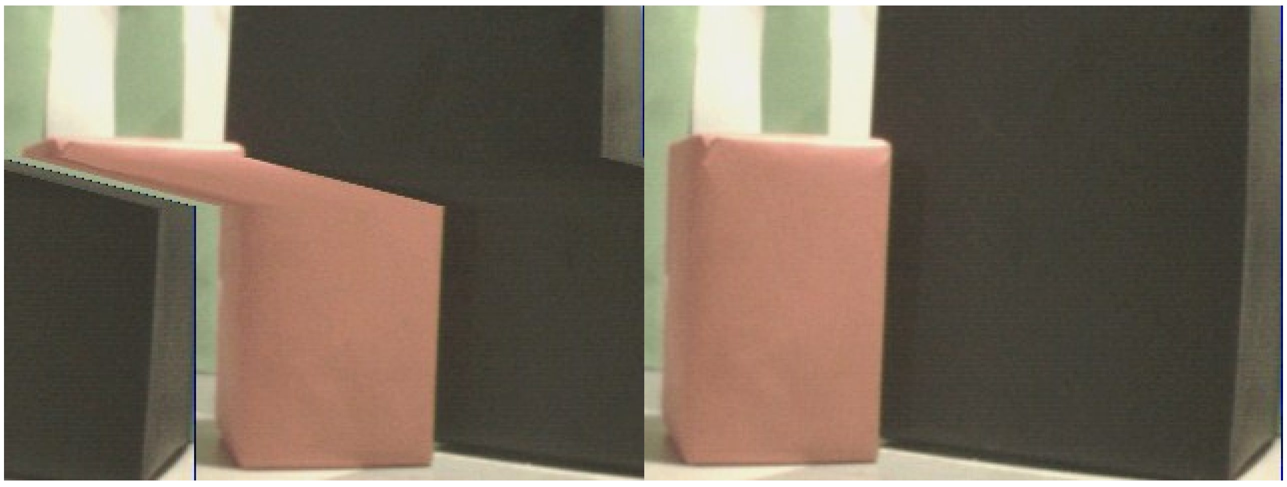


Figure 22: Unsynchronized image (left) and synchronized image (right)

Number two, because it should send at least 320*240*2 = 153600 bytes, the computer needs more than 10 minutes for sending the image data at the default baud rate of 9600. The low speed of sending problem causes that it takes almost fifteen minutes to send and receive one frame image data, which is too slow to capture and update one frame on the computer. What's more, this also causes the fake crash of the computer. To solve this problem, the code of sending extra info to distinguish the colour parameters is simplified at first, and then the baud rate is increased to 256000 by initialize the serial port at this speed. Now getting one frame image can be done in one minute.

After deal with these two problems and others, the showing part is much easier. Write a Python program that simulate the UI of this showing process, and then read all RGB data as the three colour components to be printed by PyGame pixel by pixel. Figure 23 and show the captured image in the dorm and in the experiment environment.

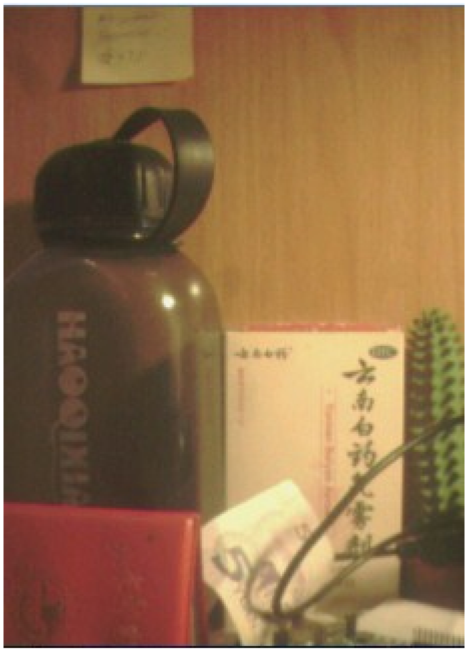


Figure 23: Captured image (in dorm)

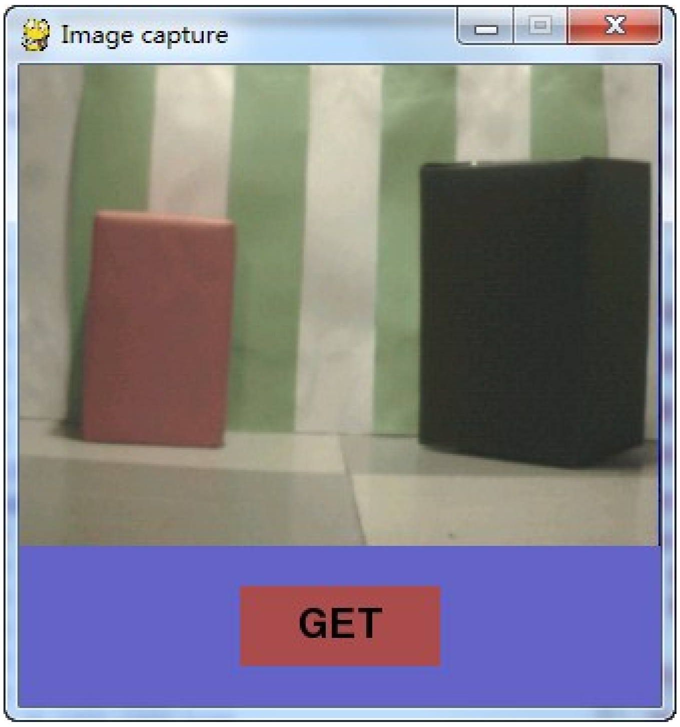


Figure 24: Captured image (in experiment environment)

3.4 Image Processing

There is no need to teach the robot to distinguish what exactly the object is, but the robot should make use of the limited image data to understand its environment and know what to do next. Therefore, image processing part, especially distinguishing the colours, is the core mission of this project.

The following two sections discuss how to accomplish edge detection and colour classification.

3.4.1 Edge detection

Edge detection is a fundamental tool in image processing, machine vision and computer vision.

From the information given in the background section, Laplace operator can help to detect the edges. A continuous Laplace operator could be modified to fit discrete data, which will become more implementable for the robot.

First-order differential:

$$f'[n]=f[n]-f[n-1] \tag{6}$$

Second-order differential:

$$\Delta f[n] = f[n] - 2f[n-1] + f[n-2] \tag{7}$$

The R, G, and B parameters of every pixel in one frame works like a series of discrete points, which can be plotted in Cartesian coordinates, where number of the pixels as X and the relevant R or G or B parameter as Y. It can also be proved that use the equations (6) and (7) to do the Laplace operator on discrete points (shown in Figure 25 and Figure 26).

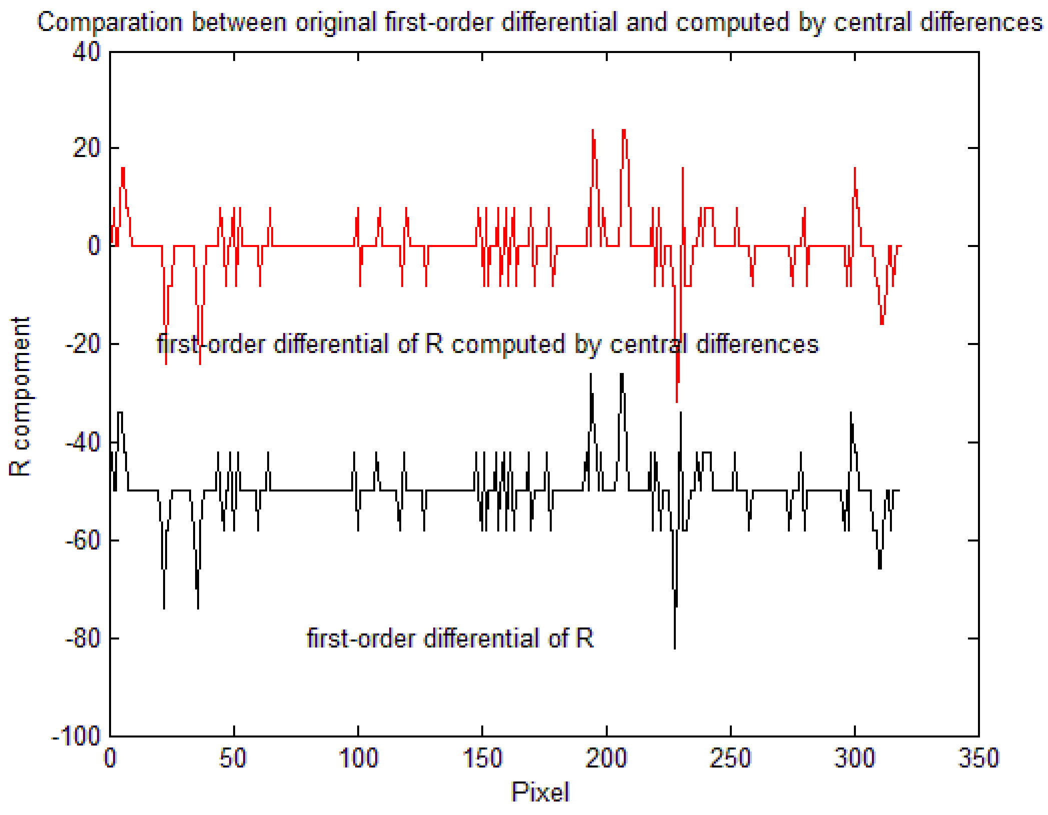


Figure 25: Proof of first-order differential on discrete points

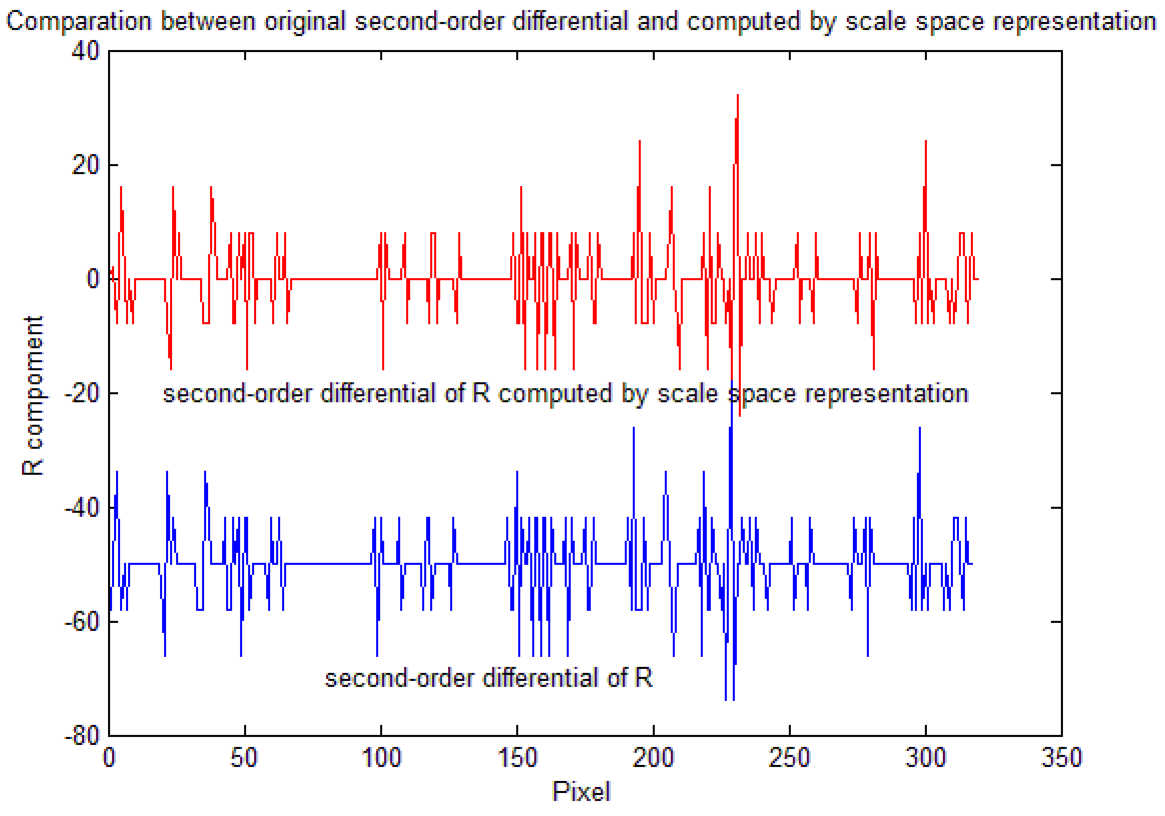


Figure 26: Proof of second-order differential on discrete points

Besides that, with observation and testing, two thresholds are set, +/-16 for first-order differential and -15 for second-order differential to get a better detection. Each line of RGB components, e.g. R component, does the first-order differential at first, and then filters those points between -16 and +16 by setting them as 0. Do the same things to all three colour components. Compare R, G, B components and choose the largest one (if the value is positive) or the smallest one (if the value is negative) of every point in these components to compose a new line, maxRGB. This new line represents the largest difference in every place where the colour changes the most, also known as the edges. In addition, in order to find the edge points easily, here implements another threshold -15. All the values above -15 will be filtered. Figure 27 shows the two thresholds' effect.

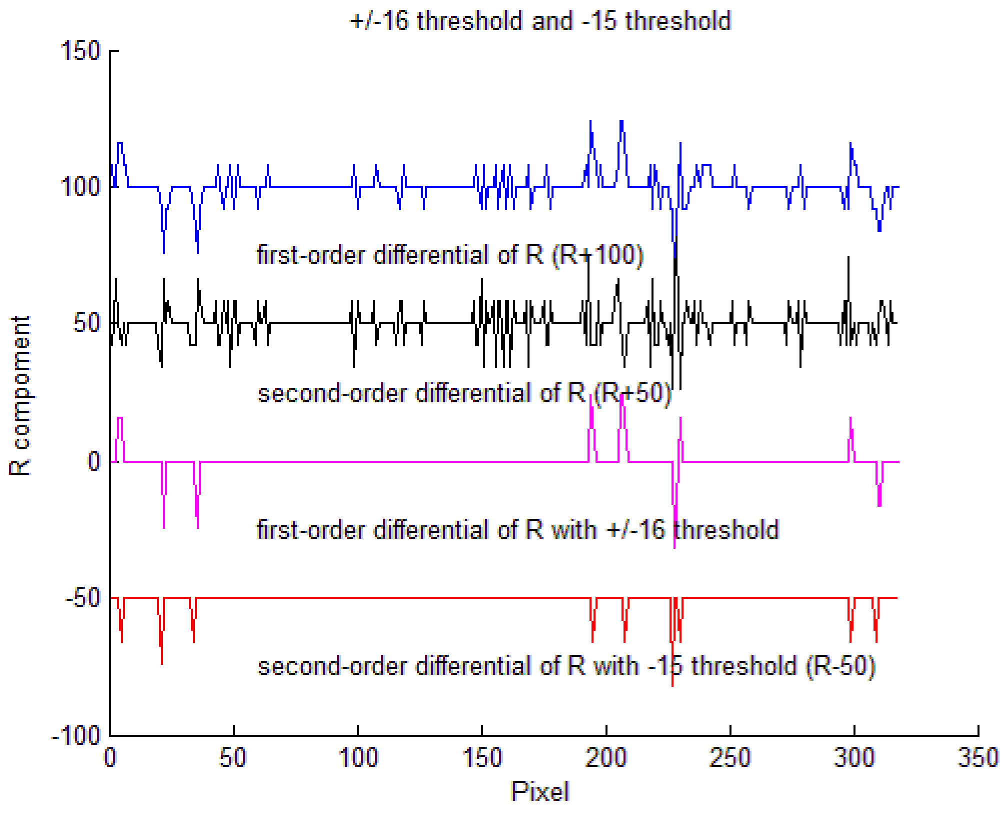


Figure 27: +/-16 threshold and -15 threshold

With the collaboration of Laplace operator and threshold, according to the original image, the edges are well defined. The whole process of edge detection is as follows.

Step 1: Output the center line's RGB parameters (Figure 28)

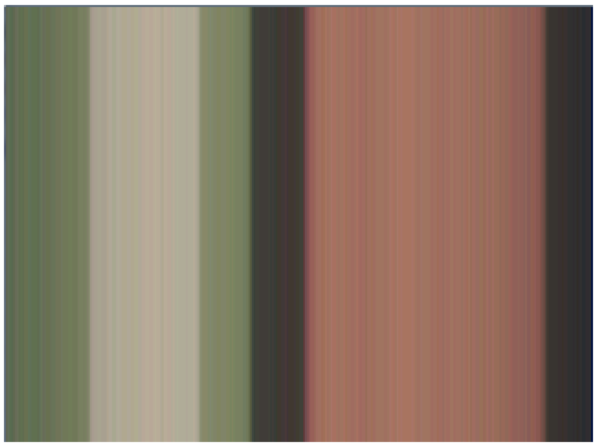


Figure 28: Centre line's RGB parameters

Step 2: Show the lines of R, G, B values (Figure 29)

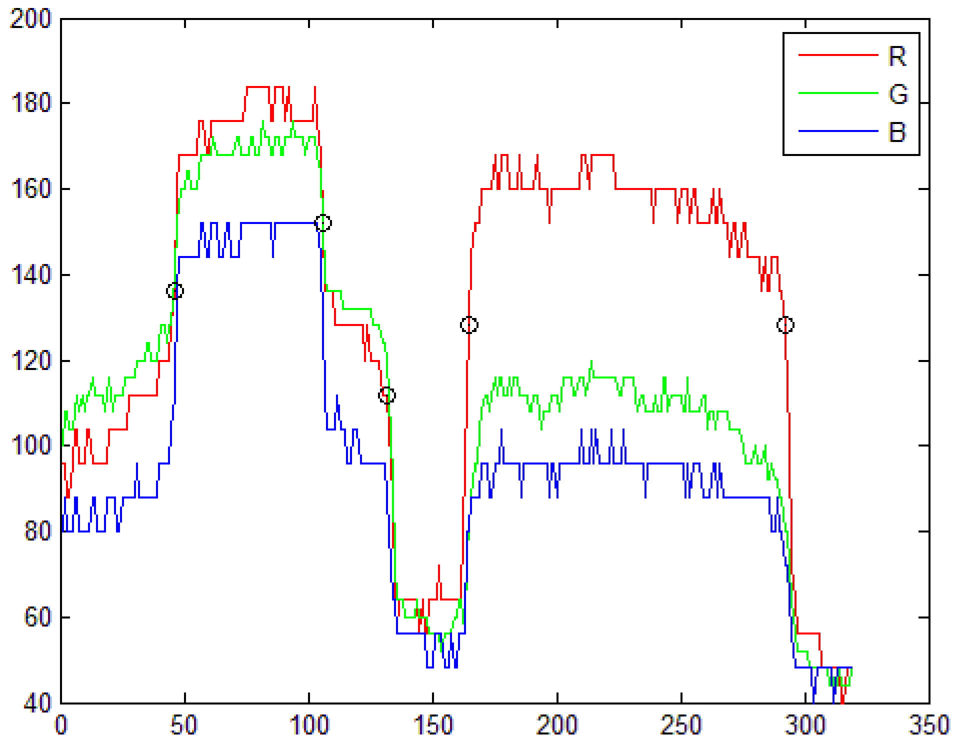


Figure 29: The lines of R, G, B values

Step 3: Do the first order differential (Figure 30)

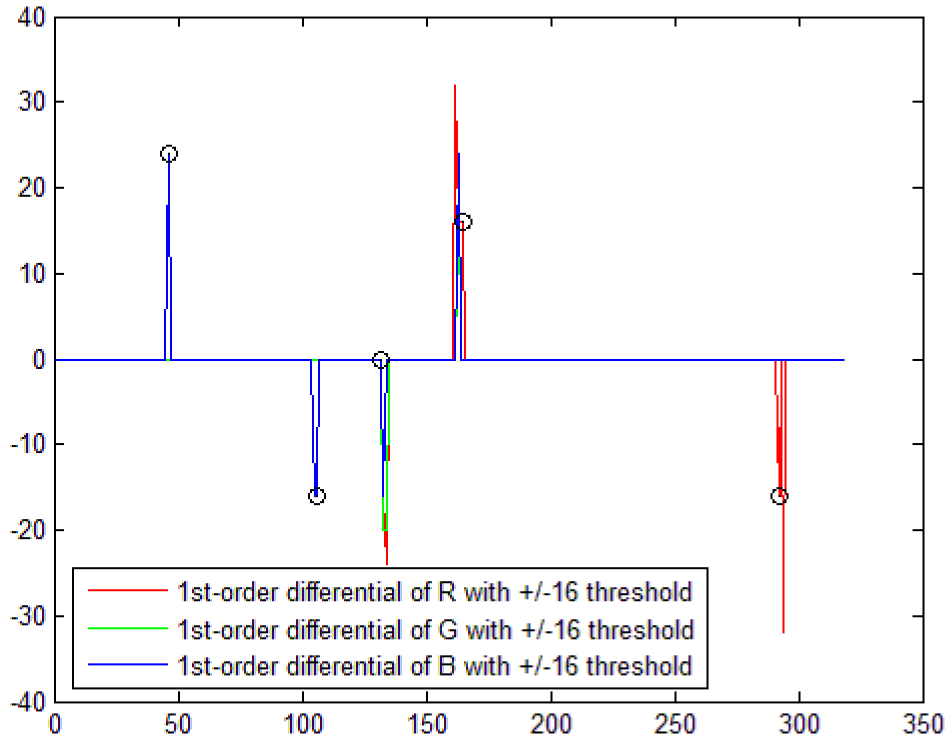


Figure 30: First order differential with +/- 16 thresholds

Step 4: Do the second order differential (Figure 31)

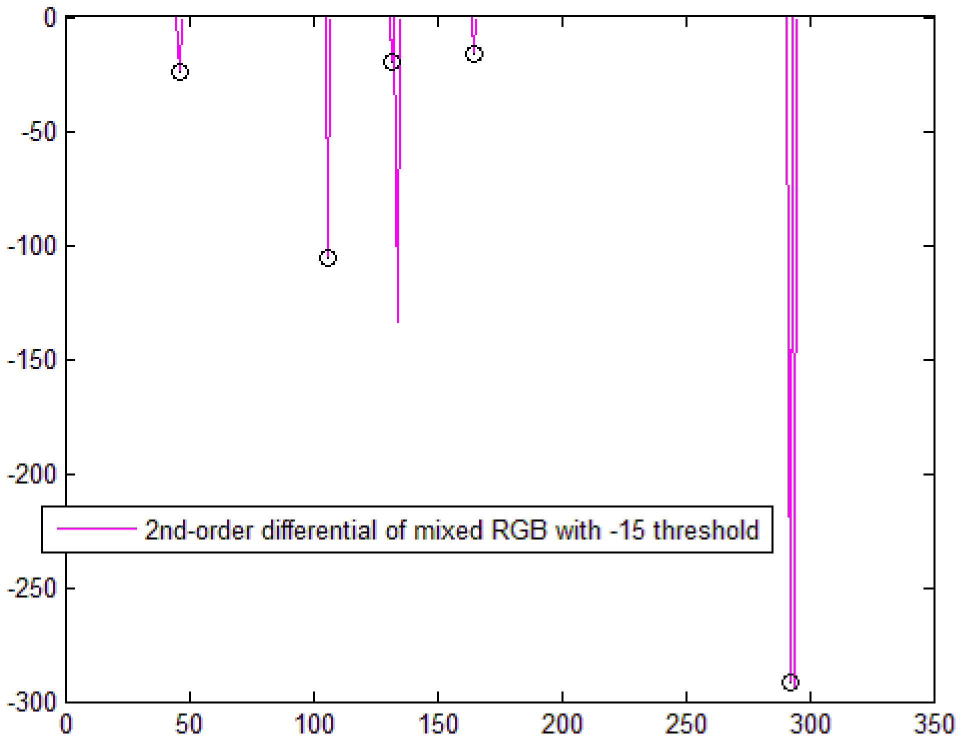


Figure 31: Second order differential with -15 threshold

The process simulation in MATLAB shows this algorithm can work for edge detection and that also can serve for the following process, colour classification. While, based on the requirements, the robot needs to do the edge detection when it runs for object-seeking. So, this algorithm needs to be implemented in C for downloading to the robot. That is similar with the code in MATLAB and it is relevantly easy to implement on the core board.

3.4.2 Colour classification

As the method mentioned before, here uses Bayesian approach to do the colour classification.

There are five steps of this approach, which are:

1. Get image RGB data
2. Define the main color of each pixel
3. Train and get the prior probability and posterior probability
    a) Count how many red pixels there and divide that by the total pixels to get the prior

probability, P(Red)

b) Count how many certain R(e.g. R=170) appear in the red block to get the posterior probability, P(R|Red) (e.g. P(170|Red))

c) Get P(G|Red), P(B|Red) and the probalility of R, G, B given other colors in the same way

4. Based on given x(r, g, b) to calculate P(Red|x), P(Green|x), P(White|x), P(Black|x)

5. Choose the largest probability to define the x(r, g, b)'s color

The flow chart in Figure 32 gives a more intuitional process of this approach.

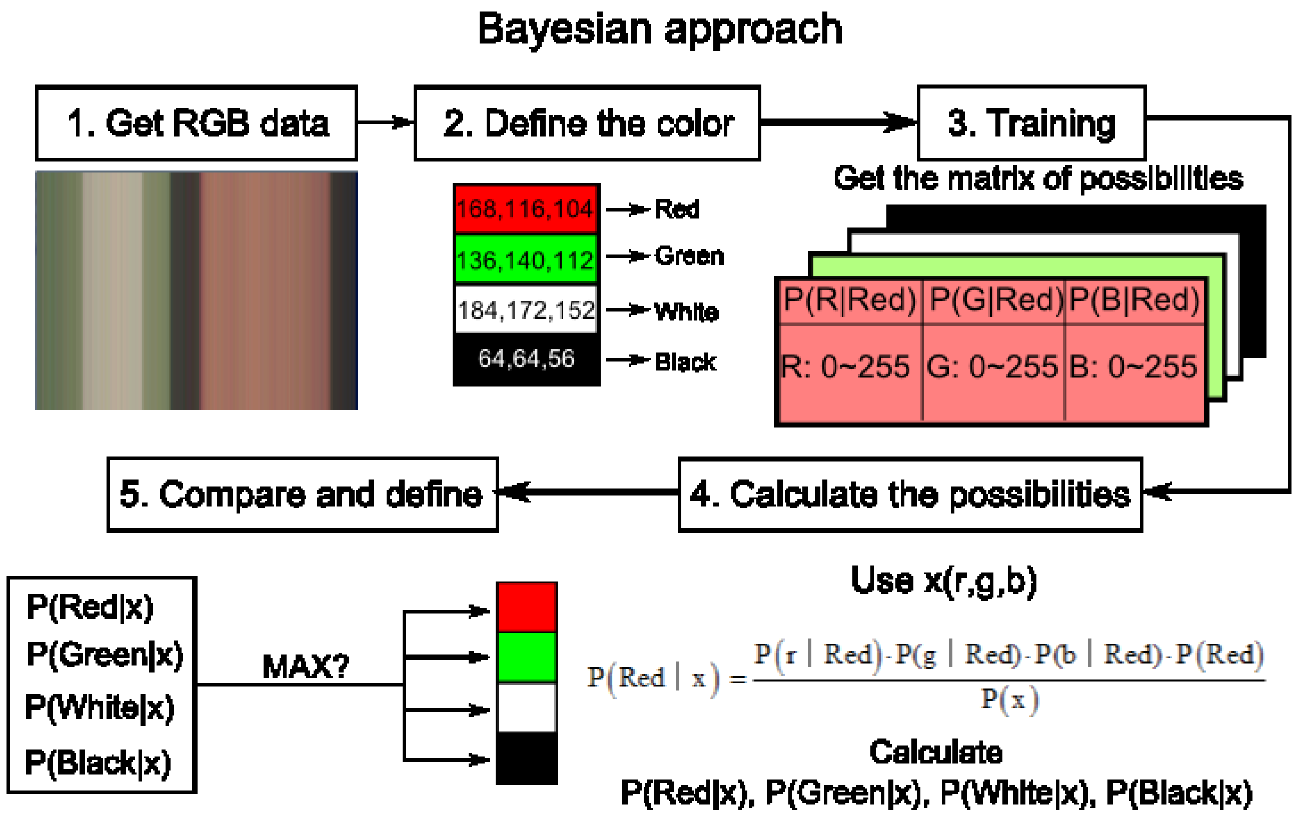


Figure 32: The flow chart of Bayesian approach

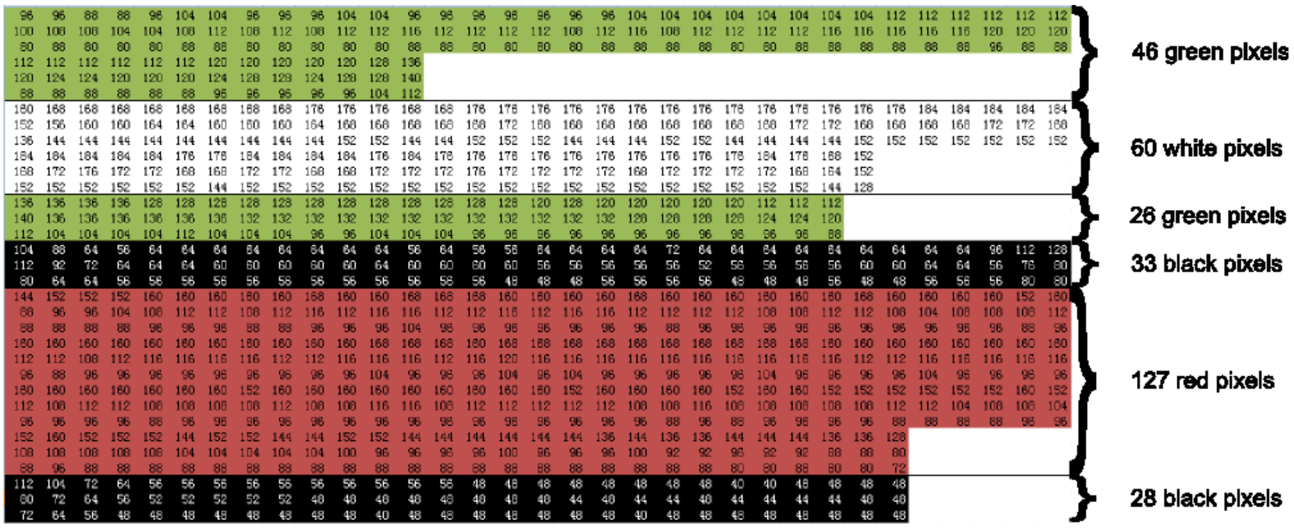
Here uses the process of getting the possibility of Red given x as an example.

$$
\begin{aligned}
P(\text{Red} \mid x) &= \frac{P(\text{Red}, x)}{P(x)} = \frac{P(x \mid \text{Red}) \cdot P(\text{Red})}{P(x)} \\
&= \frac{P(r \mid \text{Red}) \cdot P(g \mid \text{Red}) \cdot P(b \mid \text{Red}) \cdot P(\text{Red})}{P(x)}
\end{aligned}
\qquad (8)
$$

To define the main color of one pixel, it needs to compare the possibilities of that pixel is red, is green, is white, and is black. Find the largest possibility, and that colour will be the main colour of this pixel. As the equation (8) presents, the possibility of that the pixel, x, is red is determined by P(r|Red), P(g|Red), P(b|Red), and P(Red). P(r|Red) is the possibility of specific r given Red, which can be understood as how much does the specific r component contribute to define a pixel as red. P(g|Red) and P(b|Red) have similar meanings. P(Red) is the possibility that the red pixels' volume divided by the total volume in the whole training data set, also known as how many Reds are there in the whole set.

Here uses the RGB parameters of one center line, 320*3 integer values, to show the training process.

Suppose that the training data set has only 320 pixels, and all RGB parameters are presented in Figure 33.



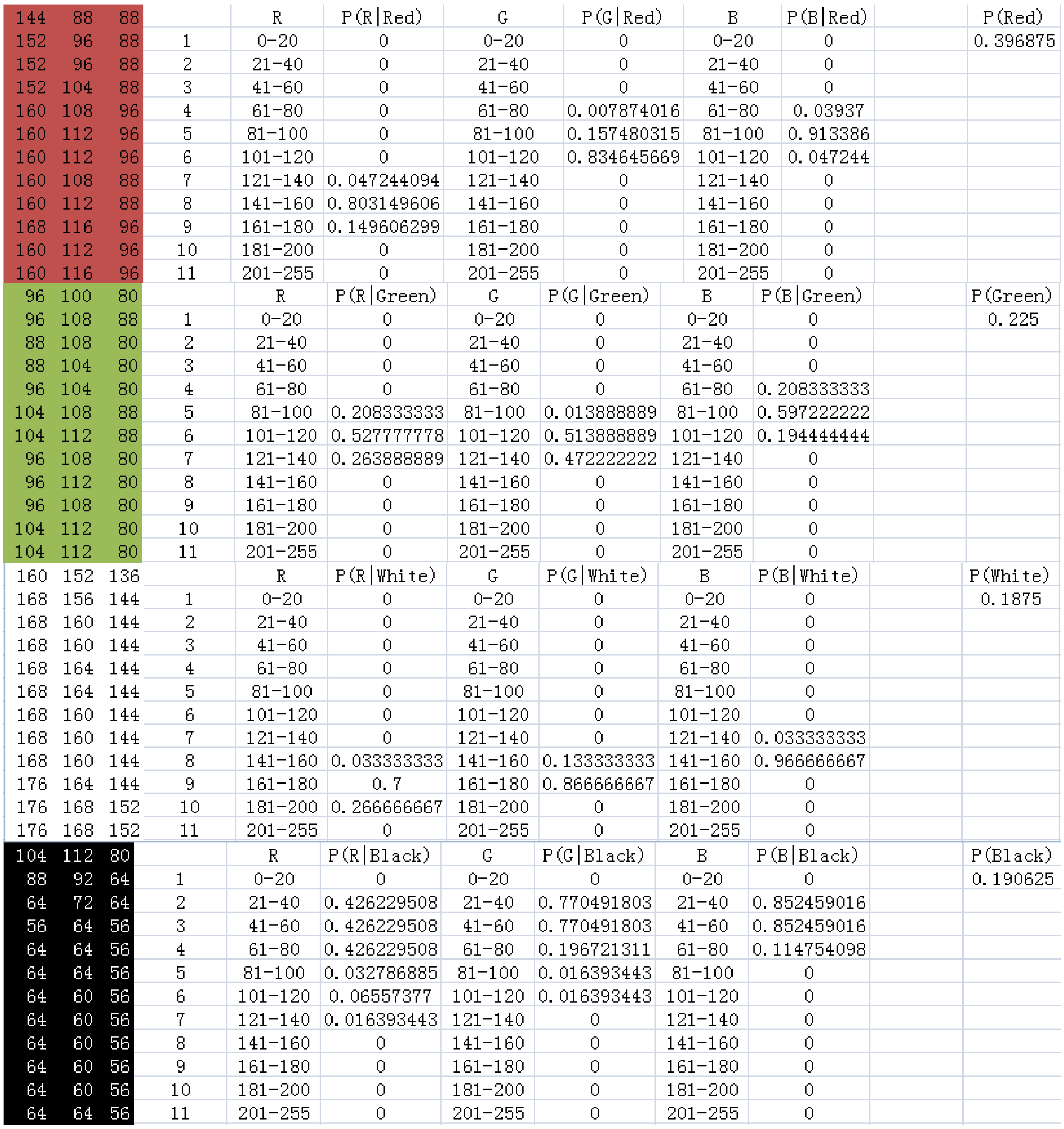
96	96	88	88	96	104	104	96	96	96	104	104	96	96	96	96	96	96	96	104	104	104	104	104	104	104	104	112	112	112	112	112	112	46 green pixels
100	108	108	104	104	108	112	108	112	108	112	112	116	112	112	112	112	108	112	116	108	112	112	112	112	116	116	116	116	116	120	120	120	
80	88	80	80	80	88	88	80	80	80	80	80	88	88	80	80	80	80	88	88	88	88	80	80	88	88	88	88	88	88	96	88	88	
112	112	112	112	112	112	120	120	120	120	120	128	136																					
120	124	124	120	120	120	124	128	128	124	128	128	140																					
88	88	88	88	88	88	96	96	96	96	96	104	112																					
160	168	168	168	168	168	168	168	168	176	176	176	168	168	176	176	176	176	176	176	176	176	176	176	176	176	176	176	184	184	184	184	184	60 white pixels
152	156	160	160	164	164	160	160	160	164	168	168	168	168	168	172	168	168	168	168	168	168	168	168	172	172	168	168	168	168	172	172	168	
136	144	144	144	144	144	144	144	144	144	152	152	144	144	152	152	152	144	144	144	152	152	144	144	144	144	152	152	152	152	152	152	152	
184	184	184	184	184	176	176	184	184	184	184	176	184	176	176	176	176	176	176	176	176	176	176	184	176	168	152							
168	172	176	172	172	168	168	172	172	168	168	172	172	172	176	172	172	172	172	168	172	172	172	172	168	164	152							
152	152	152	152	152	152	144	152	152	152	152	152	152	152	152	152	152	152	152	152	152	152	152	152	152	144	128							
136	136	136	136	128	128	128	128	128	128	128	128	128	128	128	128	120	128	120	120	120	120	120	112	112	112								26 green pixels
140	136	136	136	136	136	136	132	132	132	132	132	132	132	132	132	132	132	132	128	128	128	128	124	124	120								
112	104	104	104	104	112	104	104	104	96	96	104	104	104	96	96	96	96	96	96	96	96	96	96	96	88								
104	88	64	56	64	64	64	64	64	64	64	64	56	64	56	56	64	64	64	64	72	64	64	64	64	64	64	64	64	64	96	112	128	33 black pixels
112	92	72	64	64	64	60	60	60	60	60	64	60	60	60	60	56	56	56	56	56	52	56	56	56	56	60	60	64	64	56	76	80	
80	64	64	56	56	56	56	56	56	56	56	56	56	56	56	48	48	48	56	56	56	56	48	48	48	56	48	48	56	56	56	80	80	
144	152	152	152	160	160	160	160	160	168	160	160	168	168	168	160	160	160	160	168	160	160	160	160	160	160	168	160	160	160	160	152	160	127 red pixels
88	96	96	104	108	112	112	108	112	116	112	116	116	112	112	116	112	116	116	112	112	112	112	108	108	112	112	108	104	108	108	108	112	
88	88	88	88	96	96	96	88	88	96	96	96	104	96	96	96	96	96	96	96	88	96	96	96	96	96	96	96	96	96	96	88	96	
160	160	160	160	160	160	160	160	160	160	160	168	168	168	160	168	168	168	168	168	168	168	168	168	168	160	160	160	160	160	160	160	160	
112	112	108	112	116	116	116	116	112	112	116	116	116	112	116	120	116	116	116	116	116	116	116	116	116	116	112	112	116	116	116	116	116	
96	88	96	96	96	96	96	96	96	96	96	104	96	96	96	104	96	104	96	96	96	96	96	104	96	96	96	96	104	96	96	96	96	
160	160	160	160	160	160	160	152	160	160	160	160	160	160	160	160	160	152	160	160	160	160	152	160	160	152	152	152	152	152	152	160	152	
112	108	112	112	108	108	108	108	112	108	108	116	116	108	112	112	112	112	112	108	108	116	108	108	108	108	108	112	112	104	108	108	104	
96	96	96	96	88	96	96	96	96	96	96	96	96	96	96	96	96	96	96	96	88	96	88	96	96	96	96	88	88	88	88	96	96	
152	160	152	152	152	144	152	152	144	144	152	152	144	144	144	144	144	144	136	144	136	136	144	144	144	136	136	128						
108	108	108	108	108	104	104	104	104	104	100	96	96	96	96	100	96	96	96	100	92	92	96	92	92	88	88	80						
88	96	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	88	80	80	88	80	80	72						
112	104	72	64	56	56	56	56	56	56	56	56	56	56	48	48	48	48	48	48	48	48	40	40	48	48	48	48						28 black pixels
80	72	64	56	52	52	52	52	52	48	48	48	48	48	48	48	48	44	48	44	44	48	44	44	44	44	48	48						
72	64	56	48	48	48	48	48	48	48	48	40	48	48	48	48	48	48	48	40	48	48	48	48	48	48	48	48						

Figure 33: One frame pixels

Firstly, P(Red) equals the number of red pixels in the training data set divided by the total number. Here is that 127/320 = 0.396875.

Then do the counting process. If there are n pixels in the section 121-140, n is divided by the total number of red pixels (127) to get P(R|Red).

Repeat the previous steps to get the other probabilities. All of them in different sections are shown in Figure 34.



144	88	88		R	P(R|Red)	G	P(G|Red)	B	P(B|Red)		P(Red)
152	96	88	1	0-20	0	0-20	0	0-20	0		0.396875
152	96	88	2	21-40	0	21-40	0	21-40	0		
152	104	88	3	41-60	0	41-60	0	41-60	0		
160	108	96	4	61-80	0	61-80	0.007874016	61-80	0.03937		
160	112	96	5	81-100	0	81-100	0.157480315	81-100	0.913386		
160	112	96	6	101-120	0	101-120	0.834645669	101-120	0.047244		
160	108	88	7	121-140	0.047244094	121-140	0	121-140	0		
160	112	88	8	141-160	0.803149606	141-160	0	141-160	0		
168	116	96	9	161-180	0.149606299	161-180	0	161-180	0		
160	112	96	10	181-200	0	181-200	0	181-200	0		
160	116	96	11	201-255	0	201-255	0	201-255	0		
96	100	80		R	P(R|Green)	G	P(G|Green)	B	P(B|Green)		P(Green)
96	108	88	1	0-20	0	0-20	0	0-20	0		0.225
88	108	80	2	21-40	0	21-40	0	21-40	0		
88	104	80	3	41-60	0	41-60	0	41-60	0		
96	104	80	4	61-80	0	61-80	0	61-80	0.208333333		
104	108	88	5	81-100	0.208333333	81-100	0.013888889	81-100	0.597222222		
104	112	88	6	101-120	0.527777778	101-120	0.513888889	101-120	0.194444444		
96	108	80	7	121-140	0.263888889	121-140	0.472222222	121-140	0		
96	112	80	8	141-160	0	141-160	0	141-160	0		
96	108	80	9	161-180	0	161-180	0	161-180	0		
104	112	80	10	181-200	0	181-200	0	181-200	0		
104	112	80	11	201-255	0	201-255	0	201-255	0		
160	152	136		R	P(R|White)	G	P(G|White)	B	P(B|White)		P(White)
168	156	144	1	0-20	0	0-20	0	0-20	0		0.1875
168	160	144	2	21-40	0	21-40	0	21-40	0		
168	160	144	3	41-60	0	41-60	0	41-60	0		
168	164	144	4	61-80	0	61-80	0	61-80	0		
168	164	144	5	81-100	0	81-100	0	81-100	0		
168	160	144	6	101-120	0	101-120	0	101-120	0		
168	160	144	7	121-140	0	121-140	0	121-140	0.033333333		
168	160	144	8	141-160	0.033333333	141-160	0.133333333	141-160	0.966666667		
176	164	144	9	161-180	0.7	161-180	0.866666667	161-180	0		
176	168	152	10	181-200	0.266666667	181-200	0	181-200	0		
176	168	152	11	201-255	0	201-255	0	201-255	0		
104	112	80		R	P(R|Black)	G	P(G|Black)	B	P(B|Black)		P(Black)
88	92	64	1	0-20	0	0-20	0	0-20	0		0.190625
64	72	64	2	21-40	0.426229508	21-40	0.770491803	21-40	0.852459016		
56	64	56	3	41-60	0.426229508	41-60	0.770491803	41-60	0.852459016		
64	64	56	4	61-80	0.426229508	61-80	0.196721311	61-80	0.114754098		
64	64	56	5	81-100	0.032786885	81-100	0.016393443	81-100	0		
64	60	56	6	101-120	0.06557377	101-120	0.016393443	101-120	0		
64	60	56	7	121-140	0.016393443	121-140	0	121-140	0		
64	60	56	8	141-160	0	141-160	0	141-160	0		
64	60	56	9	161-180	0	161-180	0	161-180	0		
64	60	56	10	181-200	0	181-200	0	181-200	0		
64	64	56	11	201-255	0	201-255	0	201-255	0		

Figure 34: Matrix of probabilities

As the figure shows, the result of one center line training is inaccurate. The probabilities are zero in some sections, because of the lacking of the training data set. In the real training process, the data set contains 9933 pixels and the result of training is better, the distributions of probabilities are shown below. The detailed matrixes of probabilities are in appendix.

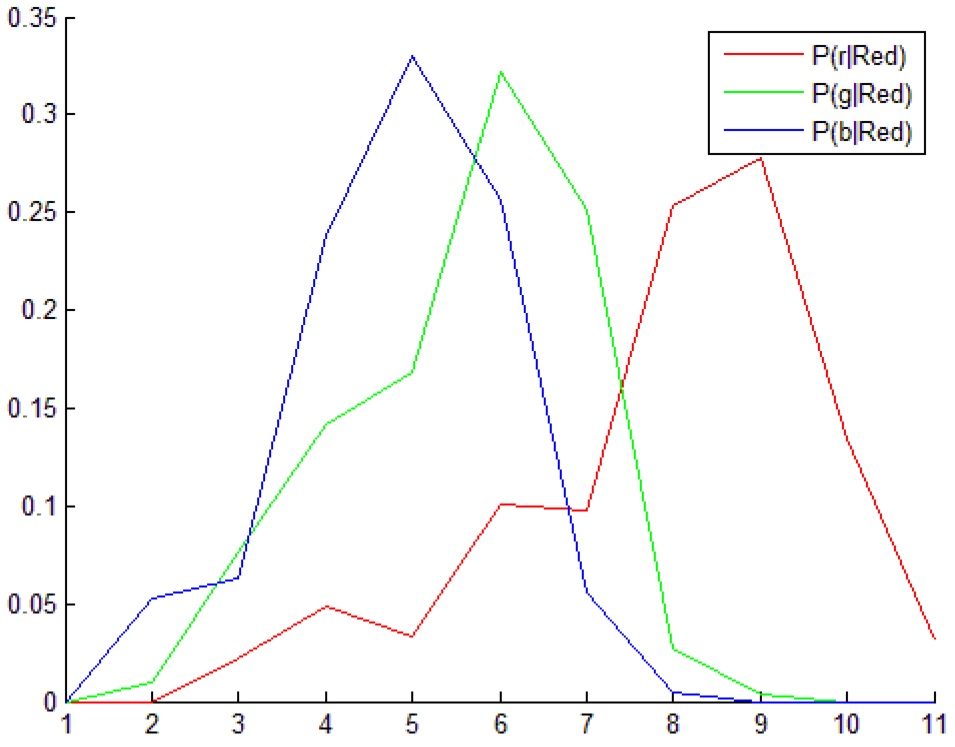


Figure 35: P(r|Red), P(g|Red), P(b|Red)

From Figure 35, it can conclude that compared with R component, G and B components contribute less to compose red colour. R component reaches the highest point in the section 9, also known as 160-180 in training.

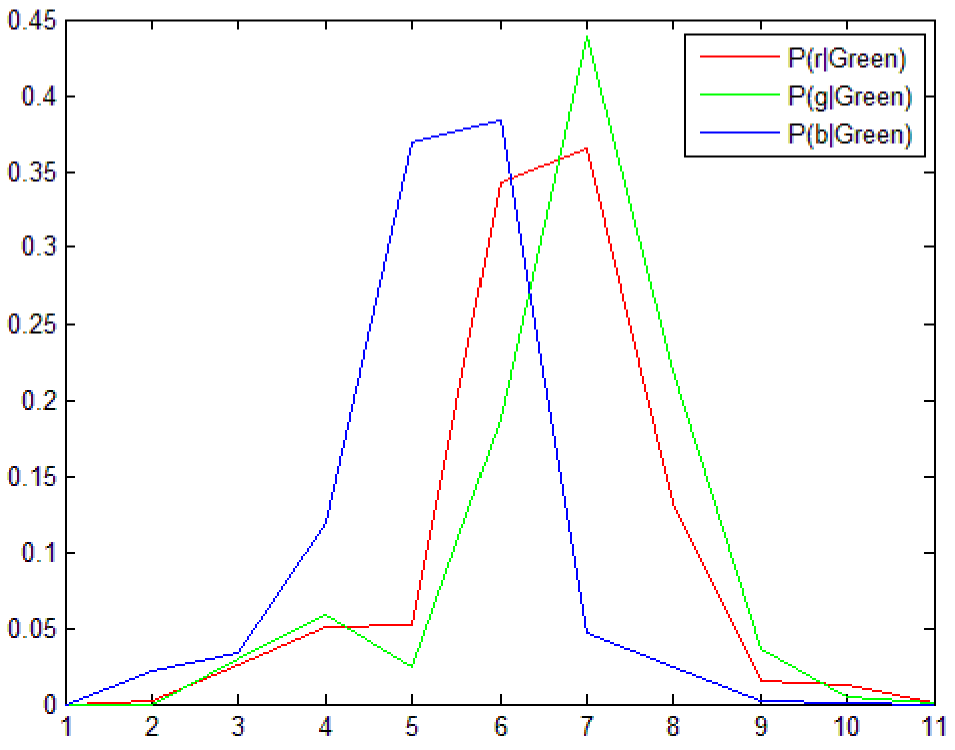


Figure 36: P(r|Green), P(g|Green), P(b|Green)

From Figure 36, it can conclude that the largest probability from B component still in lower setion. G component contributes more to compose green colour and it reaches the highest point in the section 7, also known as 121-140 in training.

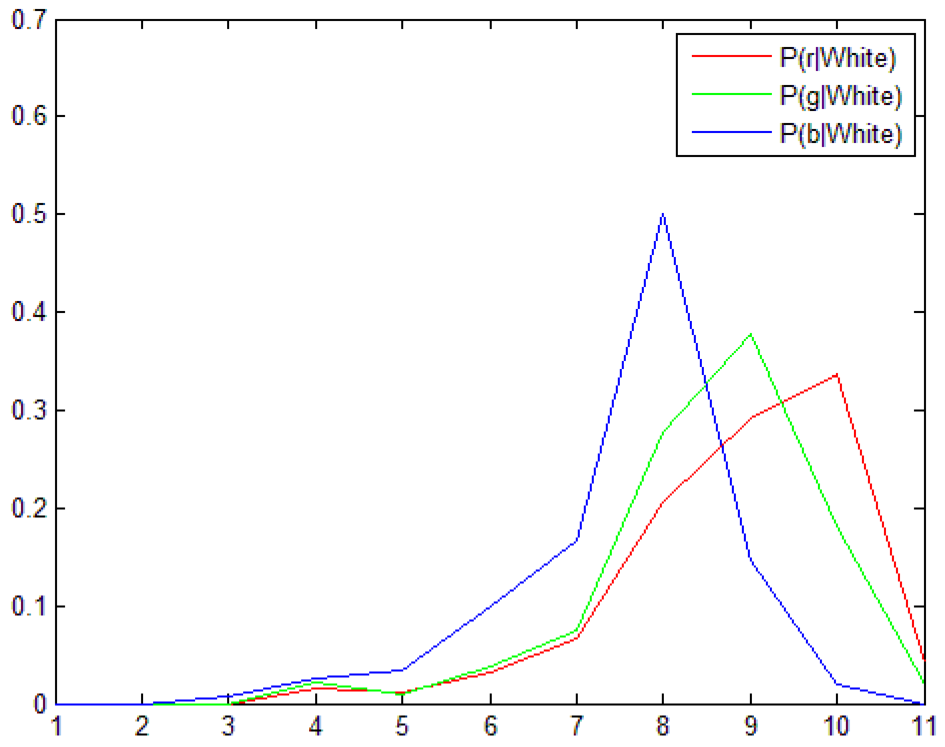


Figure 37: P(r|White), P(g|White), P(b|White)

From Figure 37, it can conclude that to compose white colour, it requires all the three components should be high enough. In the ideal condition, the figure should shows that all the three components are constantly increasing. However, this figure shows that in the real world, the defined white colour is not that "white", because in the computer definition, white should be 255,255,255. This phenomenon also indicates that from the captured image, we cannot get the real white colour.

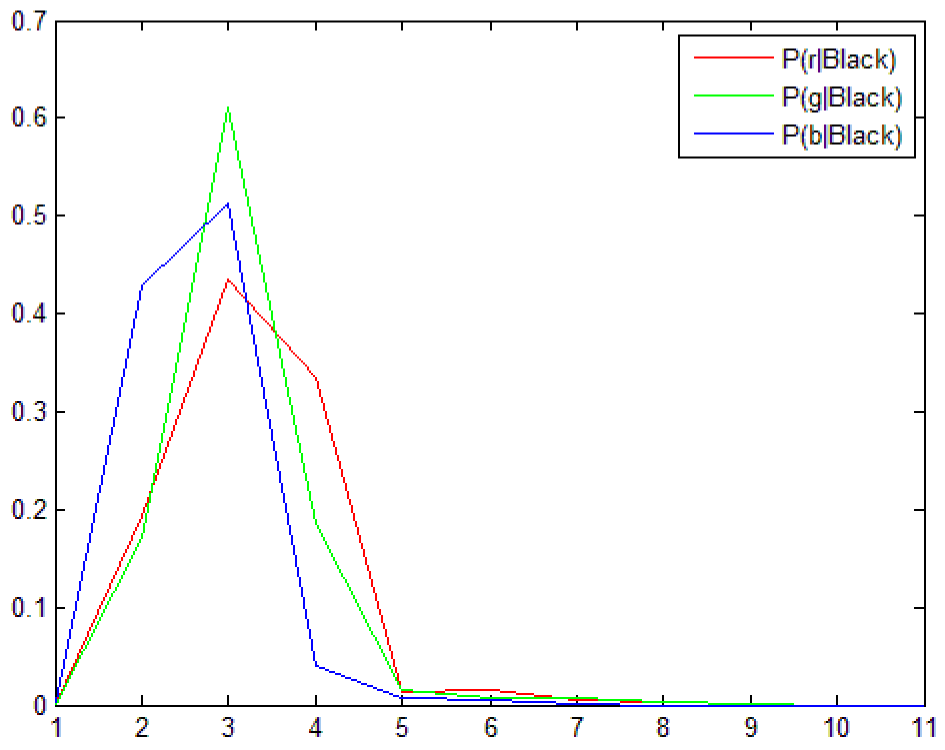


Figure 38: P(r|Black), P(g|Black), P(b|Black)

For the black colour, it has similar problem with the white one. From Figure 38 it can conclude that to compose black colour, it requires all the three components should be low enough. Here the largest probability happens in section 3, which is 41-60. In the ideal condition, black should be 0,0,0. This similar phenomenon tells that from the captured image, we also cannot get the real black colour defined in computer.

After plotting the distribution of these probabilities, those figures tell that different volume of RGB components compose different colours, which is close to the realistic. For example, High volume of R,G,B will contribute to white colour, while low volume of R,G,B will lead to black colour. There still remains problem between the training and the realistic, as when B component above 200, the possibility of b given White is zero, which should be very small but not zero. The answer to that question should be explained as we cannot get absolute white colour in real world when gather the image data, and the training data set may still be not enough.

In a word, the possibilities of r given Red, g given Red, b given Red and the possibility of Red all come from training, and that depends a great deal on the experiment environment.

By implementing the edge detection and colour classification in the robot, it can find the target with required colour.

3.5 Ultrasonic Distance Measurement

The core function of this ultrasonic module is to detect the distance between it and the obstacles. That can not only return the distance, but also serve for the collision avoidance. The following two sections explain the function of distance calculation and how to implement that function on the robot. The working principle of distance detection is shown in Figure 39.

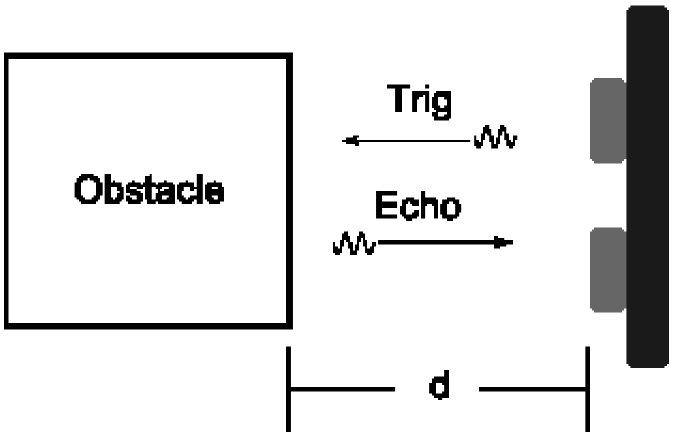


Figure 39: Working principle of distance detection

3.5.1 Distance calculation

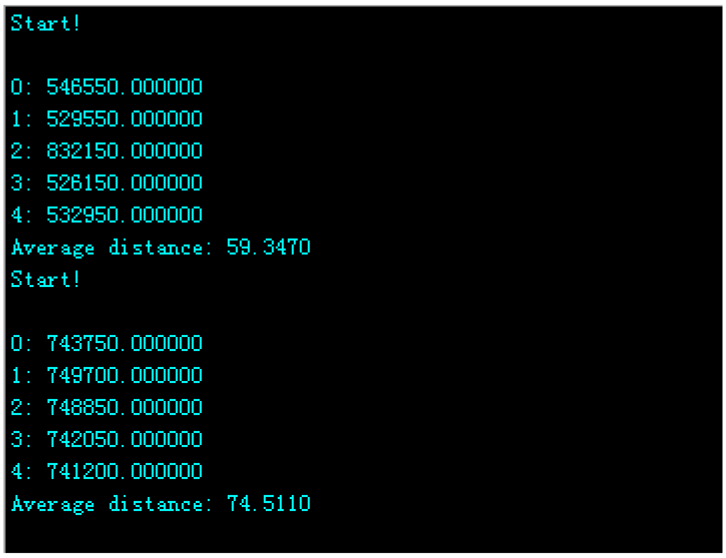
The function of calculating the distance between the ultrasonic module and the obstacle is

$$\begin{aligned} d &= vt = v_{wave} \cdot \frac{T_{duration}}{2} \\ &= 340m/s \times \frac{T}{2} = 170T \end{aligned} \qquad (9)$$

As the working principle mentioned in the background part, the ultrasonic module returns the time duration between sending signal and retrieving signal, as Tduration. Because the signal transmits with the speed of sound, the total distance d can be calculated with the equation (9).

3.5.2 Implement the measurement

To implement the measurement on the robot, detection and calculation only one time is far from enough. Here detects five times and calculates the average distance as the detected distance in the robot, more detailed results shown in Figure 40.



```
Start!

0: 546550.000000
1: 529550.000000
2: 832150.000000
3: 526150.000000
4: 532950.000000
Average distance: 59.3470
Start!

0: 743750.000000
1: 749700.000000
2: 748850.000000
3: 742050.000000
4: 741200.000000
Average distance: 74.5110
```

Figure 40: Distance detection

3.6 Motion

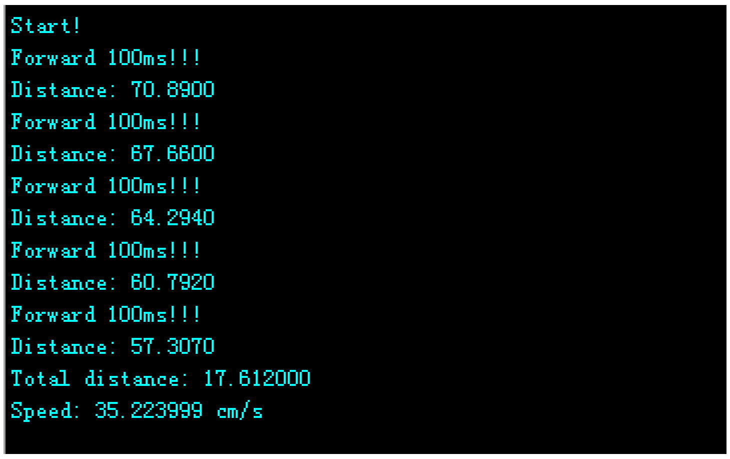
To realize that the robot can move independently, the motion system needs to provide power to the core board as well. The following sections indicate the process of making the robot move and turn with specific distance and angle and the process of speed detection.

3.6.1 Detect the speed

Cooperated with ultrasonic module, we can test out the real speed of the robot and use that to define more complicated algorithms for object-searching and return path calculating. The principle is quite easy, as use the difference between two distance detections and the running time of the robot to get the speed (Equation 10).

$$v = \frac{d}{t} = \frac{d_1 - d_2}{t} \tag{10}$$

To get the relevant accurate speed of the robot in the full power condition, the code is modified to meet the testing requirements. According to the modified code, when the robot gets the trig signal, it will run 500ms each time with distance detection. After five times testing, it can return the average speed of previous running. The process of testing is shown in Figure 41. The average speed is nearly 35cm/s.

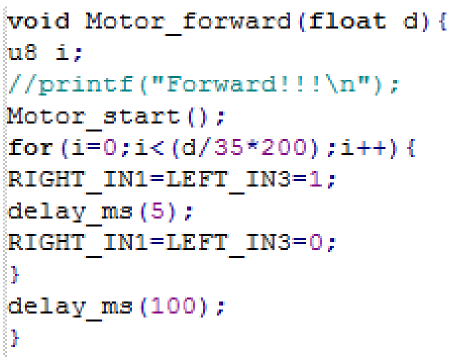


```
Start!
Forward 100ms!!!
Distance: 70.8900
Forward 100ms!!!
Distance: 67.6600
Forward 100ms!!!
Distance: 64.2940
Forward 100ms!!!
Distance: 60.7920
Forward 100ms!!!
Distance: 57.3070
Total distance: 17.612000
Speed: 35.223999 cm/s
```

Figure 41: Calculate the speed

3.6.2 Move and turn

There are four INs on L298N driver board. Each two of them control one wheel. For example, IN1 and IN2 control the right wheel. When IN1 is high level and IN2 is low level, the motor of the right wheel will run forward. Otherwise, the right wheel will run backward. The code of controlling movement is shown in Figure 42. Here also uses a for loop to switch very fast between high level and low level to provide approximate constant speed rather than giving high level with a long duration.

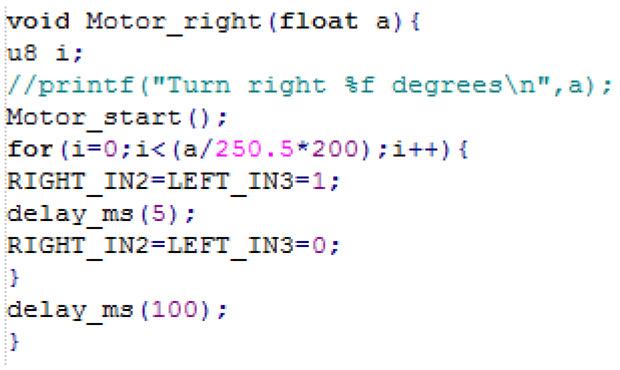


```
void Motor_forward(float d){
u8 i;
//printf("Forward!!!\n");
Motor_start();
for(i=0;i<(d/35*200);i++){
RIGHT_IN1=LEFT_IN3=1;
delay_ms(5);
RIGHT_IN1=LEFT_IN3=0;
}
delay_ms(100);
}
```

Figure 42: Function of moving forward

After get the speed of the robot, to make the robot turn with a specific angle is the next mission. According to the speed 35cm/s, use the following derivation to finish the code that can control the robot turning. The distance between two wheels is 16cm, so the perimeter is 16*3.1415 = 50.3cm.

So the robot will turn 35/50.3*360 = 250.5 degree per second. Implement that on the robot and the code is shown in Figure 43.



```
void Motor_right(float a){
u8 i;
//printf("Turn right %f degrees\n",a);
Motor_start();
for(i=0;i<(a/250.5*200);i++){
RIGHT_IN2=LEFT_IN3=1;
delay_ms(5);
RIGHT_IN2=LEFT_IN3=0;
}
delay_ms(100);
}
```

Figure 43: Function of turning right

3.7 Algorithm

The whole process includes two main algorithms, object-seeking and return path calculation, which covers most of the functions that the robot needs to achieve. The flow chart of the whole process is shown in Figure 44, and the following sections explain the implementation of these algorithms in detail.

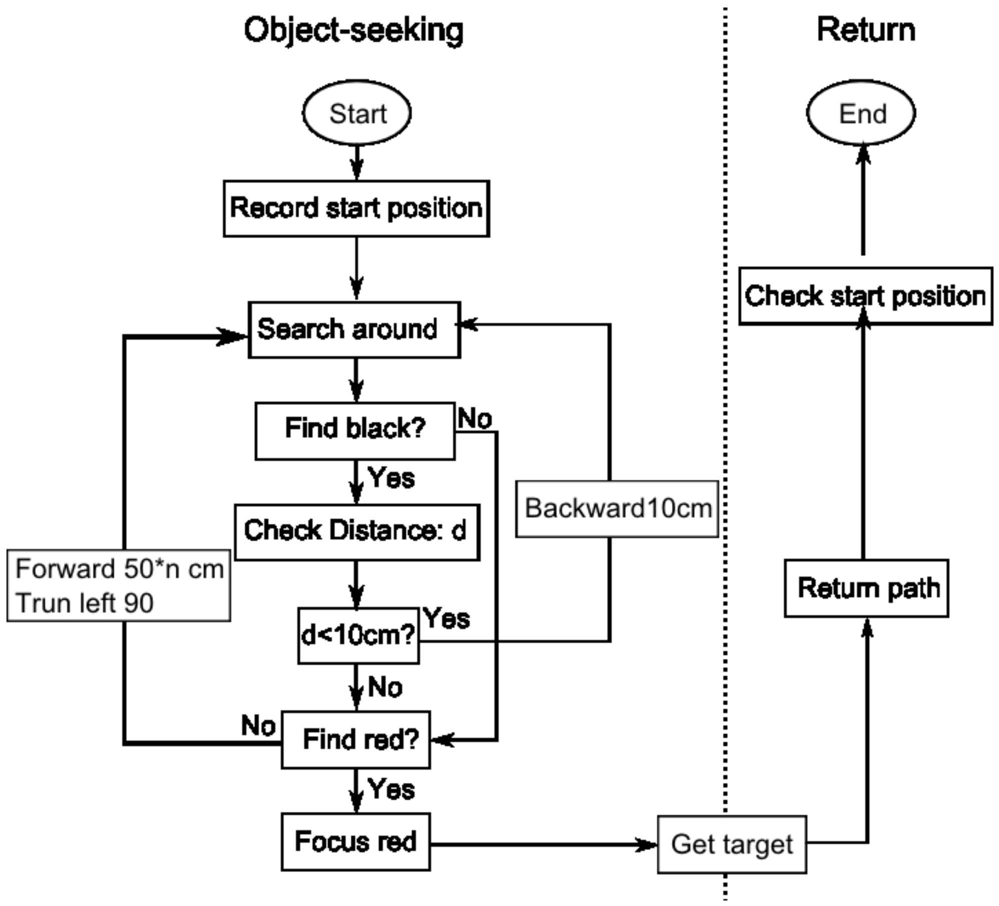


Figure 44: Object-seeking and return path calculation

3.7.1 Object-seeking algorithm

Object-seeking algorithm is the core of the whole process. It contains a number of loops and judgements. As the robot has been started, it will search around at first. The robot is sensitive to the black colour and red colour, because black means obstacle and red means target. Firstly, it will check whether there is a black obstacle in front of it. If yes, it will run backward to avoid collision. If no, the robot will check whether the red target appears in the view. If it cannot find the red area in the view, it will move forward a certain distance (based on the times, n, that the robot tried by not found the red target) and turn 90 degrees to search around the new place. While, if the red target has already existed in the view, it will call the focus red function to reach and collect the target. At the mean time of object-seeking, the robot will remember each step that it moves to find and collect the target, which will help it return to the starting position.

3.7.2 Return path calculation

Compared with the object-seeking algorithm, the return path calculation is much easier. A 2D array is used to record the steps that the robot moved when it searched the target. As a result, the robot can reverse those steps to bring the target back to the start point. This algorithm seems a little bit ideal, so a double check of the position to ensure that the robot can return to the starting position accurately is necessary.

Another function, named check_start_position, is implemented to do that job. Figure 45 shows the relationship between the distance and the pixels of background.

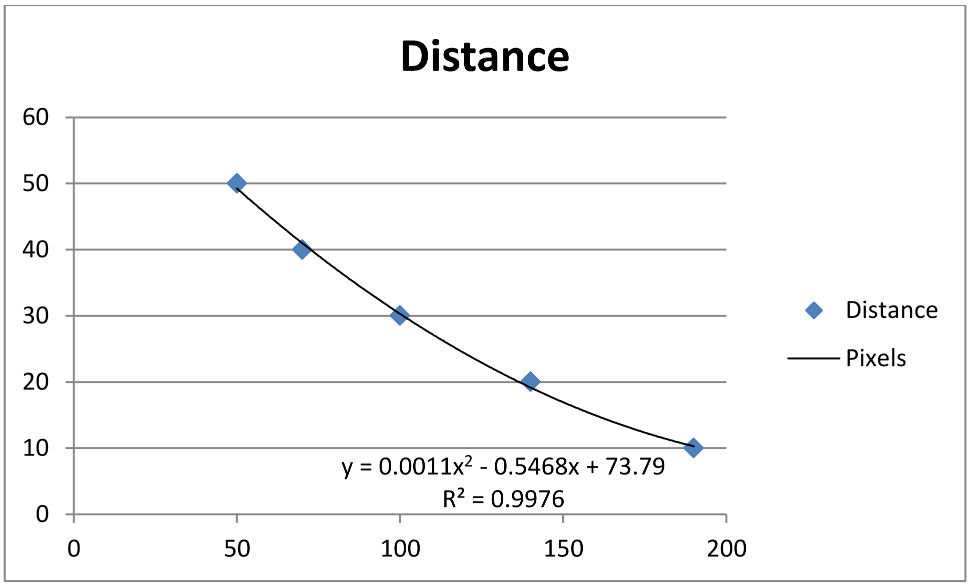


Figure 45: Relationship between the distance and the pixels

After collecting the target and reversing the movement, based on the 360-degree record of the start position, the robot will analyze the number of different kinds of pixels to locate its position and double check whether it is at the start point. If not, the robot will do the small scale adjustment to increase the position matching. However, that can only work in a small scale. If the difference between the start record and the check record is huge, the robot cannot do the adjustment.

Chapter 4: Results and Discussion

According to the design and implementation, all the modules can work well when the robot runs, and the robot can meet the basic requirements of object-seeking mentioned in the specification. All the work can be roughly concluded in parts.

Firstly, the robot can get one frame image from the camera and the available detect degree is nearly 40 degrees.

Secondly, more than 30 times of training is implemented in the edge detection and colour classification phase, which will ensure that the robot can define the colour accurately in the experiment environment.

Thirdly, after enable the ultrasonic module and motor module, the robot can run with a speed of 30cm/s and detect the distance between it and the target, which is also an important function used in the collision avoidance.

Lastly, the robot can find the target, move towards it, collect it, and bring it back to the starting position.

Although I tried to ensure the accuracy of the movement and the turning, the robot still suffers from that, which makes it quite hard to return to the exact starting position, even I've used the double check. Apart from that, the training will help the robot to define the colour easily, while it also makes the robot depend so much on the training environment. If in different environment, the brightness, the friction and other parameters may cause a huge difference of the result that this report shows.

Chapter 5: Conclusion and Further Work

After the study and implement in the past 7 month, the robot now can meet the fundamental requirements specified in the specification. The basic task of this project is that the robot needs to find the target with computer vision and bring that target back to the starting position.

To fulfil the task, the implementation is divided into five parts.

Number one is the design of the whole system and the construction of the environment. The interactions between different ends are defined and software environment and the real world experiment environment are constructed.

Number two is image collection and image processing, which is the core of this project. The STM32 core board sends sync signals to the OV7670 camera to trigger it for image data collection, and one frame image can be shown in the computer with a program written in Python. Besides the collection part, combining the learning of the computer vision, the core board also accomplishes the edge detection with Laplace operator and use the probabilities from colour classification with Bayesian approach to achieve finding the target colour red. Here indicates that the robot can see the world and find the target.

Number three is the implementation of the ultrasonic module. This module is not only used for distance detection, but also used for speed detection and collision avoidance.

Number four is the motion system. Using L298N driver board to drive the two DC motors and wheels is accomplished at first. Then more tests of the speed and turning angle are implemented to help the robot move with specific distance and turning angle. That makes it more convenient to control the robot.

Number five is the algorithms that enable the robot to seek object, avoid obstacles, and bring the target back to the start position. Two main algorithms, object-seeking and return path calculation are presented in this report. Based on the results of the image processing, the robot will adjust its moving direction to move towards to the red target. After the robot find the target object and collect it, the robot will calculate the return path and try to bring the collected object back to the starting point. What’s more, the cooperation of ultrasonic module and motor module ensures the robot can avoid black obstacles.

Though most of the project has accomplished, there are some aspects needing improvement. The robot cannot turn a very accurate degree, thus, that cannot guarantee a perfect 360-degree round search. To solve that problem in the future, steering engine and stepping motors may be used to get a more accurate result. Besides that, now the robot depends too much on the experiment environment to accomplish the colour classification, because there is no enough memory of the core board to store the training data and the robot cannot train by itself. In the future work, the approach of training needs modification to achieve that it can be done without store all training data, and the robot can train individually without the help of the computer.

References

[1] PANG Yunong, Undergraduate Project Report: Autonomous space tidying robot, 2012.5

[2] http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN1565/PF164495, 2013.4

[3] The Definitive Guide to the ARM Cortex-M3, Joseph Yiu, ISBN: 978-0-7506-8534-4

[4] http://brawikov.narod.ru/StdPerLibSTM32F10x/, 2013.4

[5] A Brain-Friendly Guide Head First Python, 1st Ed., 2010.11

[6] http://www.pygame.org/tags/pygame, 2012.11

[7] Machine Vision: Theory Algorithms Practicalities, E.R. Davies, 3rd Ed.

[8] http://en.wikipedia.org/wiki/Edge_detection, 2013.3

[9] http://en.wikipedia.org/wiki/Bayes%27_theorem, 2013.3

Project ID QCAH3 Low resource autonomous object-seeking robot using computer vision

Acknowledgement

I would like to express my special thanks of gratitude to my supervisor Dr. Christopher Harte who gave me the golden opportunity to do this wonderful project and provided me a great deal helpful advises to accomplish this project.

I would also like to thank PANG Yunong, who did a similar projest last year and shared his experience with me to help me improve the performance of my project

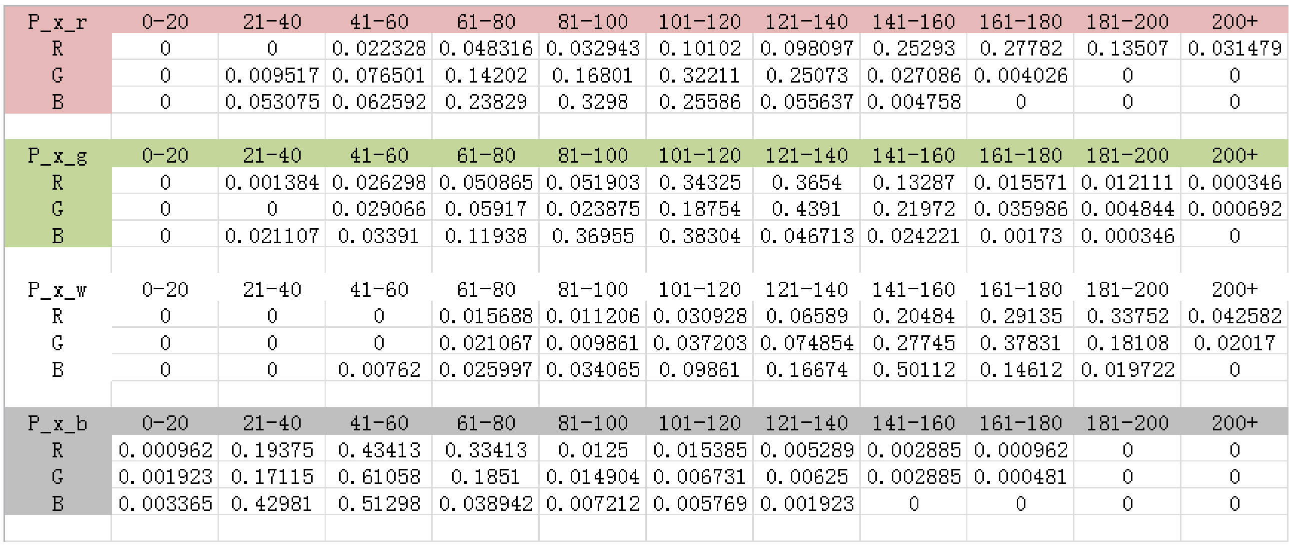
Thanks all my group mates, ZHANG Xuefeng, WANG Wenxin, HUO Yujia, ZHANG Mincong, LI Xuan, SONG Xiaorui, YANG Chunyu, ZHOU Yanchao and NI Jinsi. We helped and supported each other for the past year, especially in the final project aspect.

Appendix

1. The matrixes of the probabilities trained in Bayesian approach

2. Specification

The matrixes of the probabilities trained in Bayesian approach



P_x_r	0-20	21-40	41-60	61-80	81-100	101-120	121-140	141-160	161-180	181-200	200+
R	0	0	0.022328	0.048316	0.032943	0.10102	0.098097	0.25293	0.27782	0.13507	0.031479
G	0	0.009517	0.076501	0.14202	0.16801	0.32211	0.25073	0.027086	0.004026	0	0
B	0	0.053075	0.062592	0.23829	0.3298	0.25586	0.055637	0.004758	0	0	0
P_x_g	0-20	21-40	41-60	61-80	81-100	101-120	121-140	141-160	161-180	181-200	200+
R	0	0.001384	0.026298	0.050865	0.051903	0.34325	0.3654	0.13287	0.015571	0.012111	0.000346
G	0	0	0.029066	0.05917	0.023875	0.18754	0.4391	0.21972	0.035986	0.004844	0.000692
B	0	0.021107	0.03391	0.11938	0.36955	0.38304	0.046713	0.024221	0.00173	0.000346	0
P_x_w	0-20	21-40	41-60	61-80	81-100	101-120	121-140	141-160	161-180	181-200	200+
R	0	0	0	0.015688	0.011206	0.030928	0.06589	0.20484	0.29135	0.33752	0.042582
G	0	0	0	0.021067	0.009861	0.037203	0.074854	0.27745	0.37831	0.18108	0.02017
B	0	0	0.00762	0.025997	0.034065	0.09861	0.16674	0.50112	0.14612	0.019722	0
P_x_b	0-20	21-40	41-60	61-80	81-100	101-120	121-140	141-160	161-180	181-200	200+
R	0.000962	0.19375	0.43413	0.33413	0.0125	0.015385	0.005289	0.002885	0.000962	0	0
G	0.001923	0.17115	0.61058	0.1851	0.014904	0.006731	0.00625	0.002885	0.000481	0	0
B	0.003365	0.42981	0.51298	0.038942	0.007212	0.005769	0.001923	0	0	0	0

北京邮电大学
本科毕业设计（论文）任务书
Project Specification Form

学院　School	International Schoo	专业 Programme	Telecommunications	班级Class	2009215109
学生姓名 Name	LIU Zhijiao	学号 BUPT student no	09212987	学号 QM student no	090466298
指导教师姓名 Supervisor	HARTE Christopher	职称 Academic Title	Teaching Fellow		
设计（论文）题目 Project Title	Low resource autonomous object-seeking robot using computer vision				
题目分类 Scope	Implementation	Hardware and Software	Hardware		

主要任务及目标Main tasks and target：	By
Task 1: Become familiar with the microcontroller and camera modules	2012.12.31
Task 2: Design AI algorithm for searching the unknown space	2013.2.15
Task 3: Develop and test simple video processing software to enable robot to see its world	2013.3.15
Task 4: Demonstrate the robot seeking an object	2013.4.15

Measurable outcomes
1) Working microcontroller/camera system that can drive a robot chassis
2) Demonstrable image processing code
3) Demonstration of robot seeking the object in its world

主要内容Project description：

The task in this project is to develop a robot that can see its world using a small camera module. The robot should be able to investigate its world using its primitive vision system and attempt to find a known specific object (for example the only red object in the test space). The robot should then bring that object back to its starting position. A project from last year developed the camera module interface for a microcontroller so this information and library code will be provided to accelerate development of the robot and the AI component of the work.

Project outline

This project requires that the robot can search target object with the help of a camera, collect it and bring it back to starting point. The main functions to fulfill the tasks are specified below in hardware, software, and driver aspects.The hardware system contains the main functions of control, sensing, motion and UI. Control section is the core of this project. Due to the high efficiency of ARM processor, the brain of the robot will be a STM32 microcontroller. Sensing section meets the requirements of detection. There will be three modules in this section, which are camera module using OV7670 for pattern recognition, infrared module ultrasonic module for calculating distance and avoiding obstacles. Motion section includes wheels, couplers, motors, etc. to enable that the robot can move and veer. UI section mainly helps the users to interact with the robot. Because this robot will work automatically, the interaction only includes the launching.The software system mainly does the algorithm and logical work. Firstly, it can map a specific area and complete video processing with pattern recognition algorithms. Secondly, it can calculate the path to the target object without hitting obstacles. Thirdly, it can record the moving distance and turning angle to make sure that it will move back to the starting point. Lastly, it can change mode between push and pull to meet different conditions. To implement the software, C and Python programming are needed.A driver is the connection between the hardware and software to ensure all the logical algorithms can be real executed on the hardware. Three kinds of drivers play a momentous role to this project. Camera driver contributes to reading the video data from the camera. Sensor driver generates signal to the microcontroller when it finds unusual situation. Motion driver is responsible for all the movement functions, such as moving forward, backward, turning left, right, etc.

[1] PANG Yunong, Undergraduate Project Report: Autonomous space tidying robot, 2012.5
[2] David Cook, Robot Building for Beginners (2nd Ed.), TECHNOLOGY IN ACTION™, ISBN-13 (electronic): 978-1-4302-2749-6
[3] J. Palacin, A. Sanuy, X. Clua, Autonomous mobile mini-robot with embedded CMOS vision system, 5-8 Nov. 2002

Fill in the sub-tasks and select the cells to show the extent of each task

	Nov		Dec		Jan		Feb		Mar		Apr		May	
Task 1: Become familiar with the microcontroller and camera modules														
Background reading: determine using which microcontroller, camera, and other modules.	■													
Learn the programming of microcontroller and get image data from the camera.		■	■											
Learn video processing and accomplish pattern recognition.		■	■											
Understand how the microcontroller cooperates with other modules and design a prototype.				■										
Task 2: Design AI algorithm for searching the unknown space														
Map the valid searching area.				■										
Final exam preparation					■									
Realize multi-step searching.						■	■							
Calculate the path and implement the moving forth and back algorithm.						■	■							
Task 3: Develop and test simple video processing software to enable robot to see its world														
Design and implement drivers.			■	■		■								
Simulate the robot's vision by computer.						■	■	■						
Revise the software.							■	■	■					
Task 4: Demonstrate the robot seeking an object														
Control the wheels to reach its destination and collect target object.								■	■					
Change the mode between push and pull.									■	■				
Come back to the starting position.									■	■	■			
Write reports and prepare for viva.		■	■	■		■	■	■	■	■	■			

Risk Assessment

Table 1: Risk Assessment Table

Description of Risk	Ultrasonic is burned	Motors cannot work	Robot crashed
Description of Impact	Can't detect distance	Can't move	Some parts are broken
Likelihood Rating	High	Medium	Low
Impact Rating	Low	Medium	High
Preventative Actions	Prevent the ultrasonic directly connecting to the VCC	Make sure every process is in correct order	Changing step by step, not to update too many things one time

Environmental Impact Assessment

1. Cost of manufacture

   I bought almost every component from Taobao, and it cost about 300 RMB for the whole robot.

2. Waste disposal and recycling

   The batteries of this robot are rechargeable, which can be recycled and used later.

3. Energy use in service

   My computer and the robot are the main two energy consuming devices in this project.

4. Savings in energy

   When programming, I shut down the robot power to save energy.