The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

Applied EEE Construction Project: Lab Report 3: Line following using OpenCV and level of autonomy

Lab report from sessions 5 & 6 of the applied EEE construction Project

Michael Bagge-Hansen
eeymb8@Nottingham.ac.uk

Abstract:
A summary of the tasks undertaken in Sessions 5 & 6. Adding a Raspberry Pi to the vehicle with a camera to complete computer vision tasks, allowing the vehicle to become autonomous. The line following system was not completed due to a lack of fine motor adjustment therefore overcorrecting for the changes in line. Other systems, Symbol detection, Stop light detection and Shape counting was developed with few problems.

April 2020
The University of Nottingham

Contents

Chapter 1

Introduction

This report summarises the objectives achieved in sessions 5 & 6 of the construction project. The main goal of which was to use a raspberry pi with open CV to create computer vision systems such as line following and symbol detection. The aim for the sessions is to complete a series of challanges around a track shown in figure 1.1.

Figure 1.1: consept drawing for track

1.1 Raspberry pi

The raspberry pi is a small single-board computer designed for projects such as this. It has a quad core 1.2 GHz ARM processor [1], a good choice for our project as it is powerful enough to run most programs with ease and are cheap enough to keep the price of the product to £32 [2]. The Raspberry pi has easy access to GPIO pins allowing for communication to the arduino's. The Pi's included ports; HDMI, micro USB for power, Ethernet and 4 USB 2 ports, allow for easy setup, shown in section 2.4.1.
The pi runs Raspbian, based on Debian, a linux os. Raspbian is basic, build for beginners. It launches to a desktop environment that is lightweight, perfect for running on the ARM possessor. It comes with the apt repository allowing for easy install of many applications.
The pi is an excellent choice for this project due to its from factor, price, easy setup and its large community, due to its popularity there is a wide range of support and documentation, meaning the problems you might have will already have been solved by someone else.

1.2 Open CV

Open CV is a c++ and python library for Computer vision. It has many advanced features such as object detection and 3D transformation, however for this project we will only be using the

more basic functions as follows: masking - creating a black and white image with white being only a specified color range and black being every other color. Create contour & poly contour - create an line at all the edges between the black and white, simplify the contour to a few straight lines.
To get used to the library a task was assigned who's objective was to find the main color in a image, completed in appendix 7.2.

Chapter 2

Physical Systems

2.1 Audio Amplifier

In order to increase the outputs available from the car an Set of speaker was to be added. This is to be made with an LM3886 op amp and will use the Raspberry pi's audio jack and be powered from the 5v and ground GPIO pins on the pi. Using the GPIO pins on the raspberry pi as sound output was another option, 2 variations on this where explored. to have make an i2c sound card or to connect the amplifier to the PWM GPIO pins.

PWM pins As shown in figure 2.1, the headphone jack is connected to PWM pins: PWM1 to left channel, PWM0 to the right. To access these channels with the GPIO pins 12 and 13 on ALT 0 [3].
this would allow for the circuit to improve the sound quality and would make the design of the vehicle more streamlined. This would be a good option if there was more of a space concern and would be ideal if custom PCBs could be made as there would be no bulky connector between the pi and the amplifier, however as all the components are made by hand it is easier to make everything on strip-board meaning the circuit would be much larger than necessary.

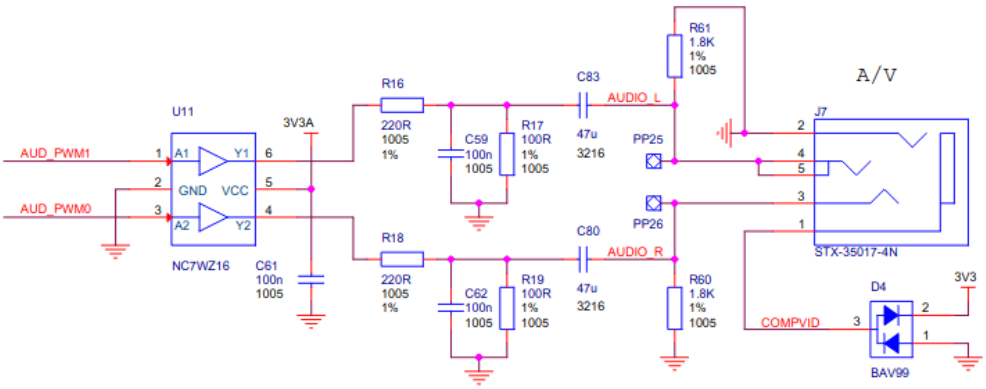


Figure 2.1: Raspberry pi headphone jack circuit [4]

i2c soundcard An i2c sound card would have similar benefits to the PWM pin option. It would also have the benefits of allowing audio input, which the raspberry pi does not have out of the box and would allow other devises on the bus to communicate. However it would also have similar downsides to the PWM pins as it would require a complex circuit with specialised chipsets as well, thereby not making it a viable option.

Headphone jack This system allows the circuit to be more plug and play, allowing for easy installation and debugging as the points of failure are easy to test individually. the circuit would be fairly simple as it would not require any other processing other than the amplifier its self. This is the most viable option considering the of the vehicle, so it was chosen as the continued design.

2.1.1 Circuit

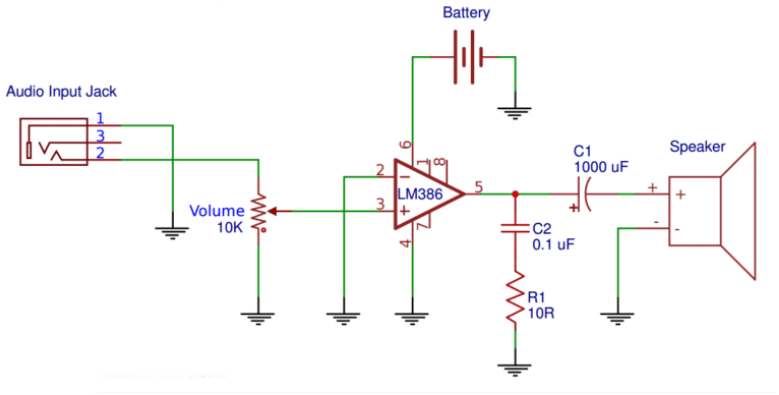
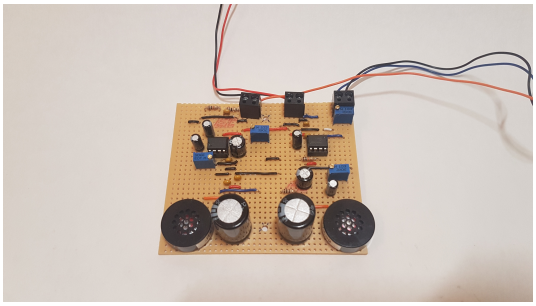
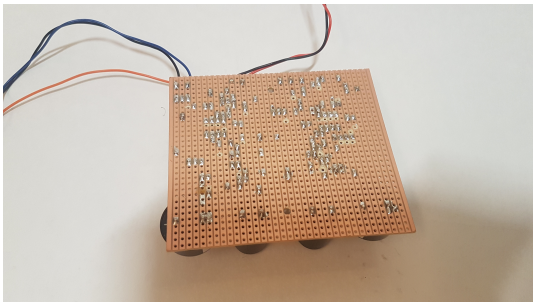


Figure 2.2: Audio amplifier circuit [5]

This amplifier circuit shown in figure 2.2 is from a website shown in [5], it is matched with 8Ω, $0.08W$ speakers from RS Components [6] they can be mounted to a pcb and are low resistance enough to be powered of the 5v supply from the GPIO pins on the raspberry pi. 2 of these circuits are constructed on a single pcb to keep the vehicle compact. The circuit's power is connected to a 2 pin header on the +5v and ground on the Raspberry pi Breakout Board though a screw terminal block, this allows the wires to be shortened or lengthened to suit the layout of the vehicle. The input signal is also connected by screw terminal blocks, and connected to the audio jack on the raspberry pi. Assembly of the audio amplifier shown in figure 2.3.



(a) Amplifier PCB front



(b) Amplifier PCB back

Figure 2.3: Amplifier PCB

2.2 I2C level shifter

The I2C bus on the arduino runs off 5v, whereas the I2C bus on the raspberry pi runs off 3.3v. Supplying the raspberry pi's I2C pins with 5v could damage the components of the pi, potentially making it unusable. Similarly with the arduino being fed 3.3v, although it would be able to handle the voltage it will behave unpredictably making it unusable as a communication signal.

To fix this a level shifter is to be designed, this will allow the 3.3v signal from the Pi to be pulled up to 5v to allow the arduino to read it, and drop the 5v signal to 3.3v to prevent damage to the raspberry pi. Circuit diagram in figure 2.4

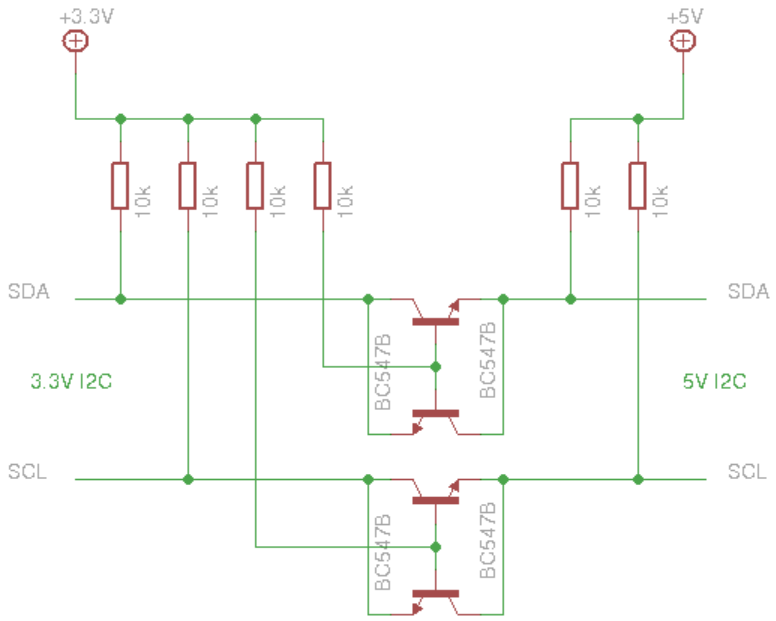
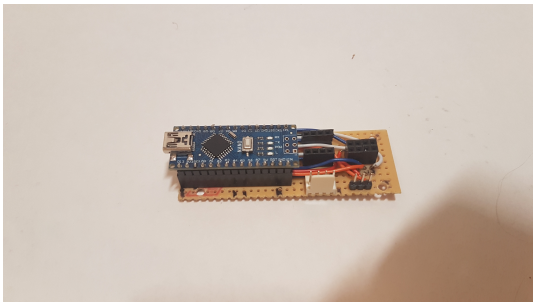


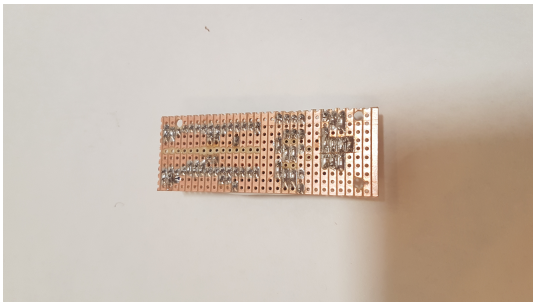
Figure 2.4: i2c level shifter circuit diagram [7]

2.3 Sensors with Arduino

As part of the specification it was required that the vehicle had some ultrasonic sensors, as well as this the camera was to have pitch control with a servo. It was decided that these sensors and the servos would be controlled with a arduino nano and communicate with the raspberry pi over i2c. This was decided to do this instead of connecting directly to the raspberry pi because the raspberry pi does not have libraries to control servos and ultrasonic sensors meaning custom code would have to be written whereas not much would be needed for the arduino. This approach would also leave room for expansion, for example on the circuit board there are pin headers for a RF module, allowing for the car to be remove controlled. Circuit assembly shown in figure 2.5. Code for system shown in



(a) Sensor circuit front



(b) Sensor circuit back

Figure 2.5: Sensor circuit

2.4 Raspberry Pi & Camera

2.4.1 setup

The setup for the raspberry pi is fairly simple, everything needed is on the os install, only thing needed to adjust is the password to prevent unauthorised use with the command `passwd pi`. Also to allow open cv programs with windows to run from an ssh terminal X11 forwarding should be enabled in the ssh config file: */ect/ssh/ssh_config*.
the only issue is connecting to Eduroam as the network is not supported by the built in systems on Raspbian. to access the network the instructions in appendix 7.1

The camera is a camera sold and manufactured by the raspberry pi foundation, it uses a Sony IMX219 8-megapixel sensor[8], found in figure 2.6.
To test the camera the `raspistill` is used, buy typing `raspistill -o imgage.jpg` an image file of the camera feed will be generated. Systems build into Linux will allow openCV to interface with the camera seamlessly

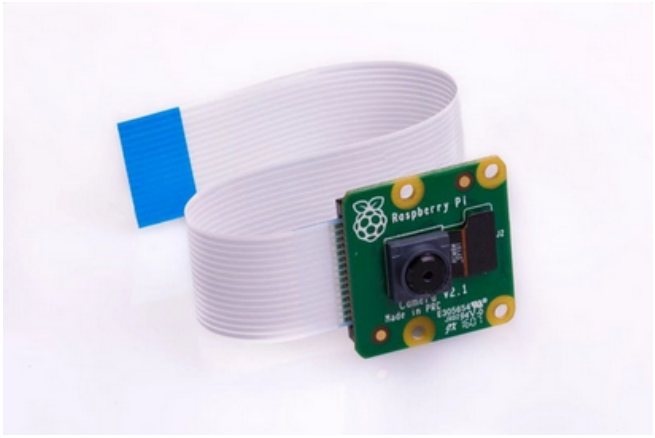


Figure 2.6: pi camera[8]

2.4.2 i2c communication

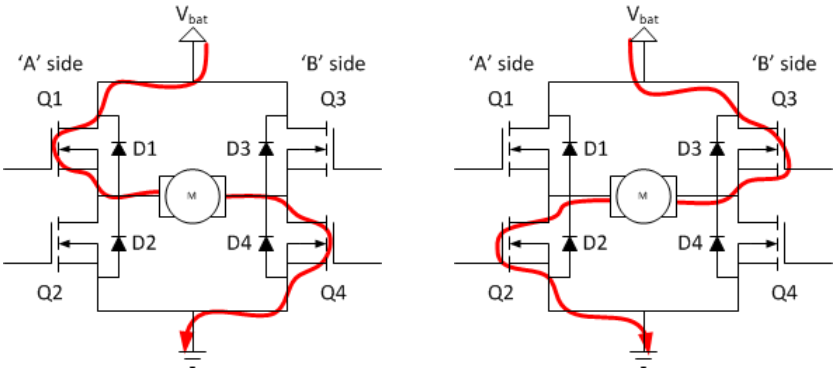
To communicate with the arduinos on the vehicle the library Pi2c, from Johnny Sheppard on GitHub [9]. The functions from the library are shown in listing 2.1

```
#include "pi2c.h"
int main(){
    Pi2c arduino(7); //Create a new object "arduino" using address "0x07"
    char receive[16]; //Create a buffer of char (single bytes) for the data

    //Receive from the Arduino and put the contents into the "receive" char array
    arduino.i2cRead(receive,16);
    //Print out what the Arduino is sending...
    std::cout << "Arduino Says: " << receive << std::endl;

    //Send an 16 bit integer
    arduino.i2cWriteArduinoInt(4356);

    return 0;
}
```

Listing 2.1: i2c commands example file [9]

2.5 Motor Control

The motors on the vehicle are powered by H-bridge circuits, they connect the 15v battery voltage to the motor through 2 of 4 MOSFETS, depending on the pair that is activated the current will flow in either direction across the motor making bi-directional control available, shown in figure 2.7.



(a) motor direction 1 (b) motor direction 2

Figure 2.7: H-bridge operation [10]

2 of the MOSFETS are PWM, allowing for speed control of the motors. PWM or pulse width modulation is a way of varying the power of a input signal, it will rapidly switch on and off the signal in pulses, and adjusting the width of there pluses to change the power of the signal.
The cars motors are controlled with an arduino, so to control the motors from the raspberry pi communication code must be written.

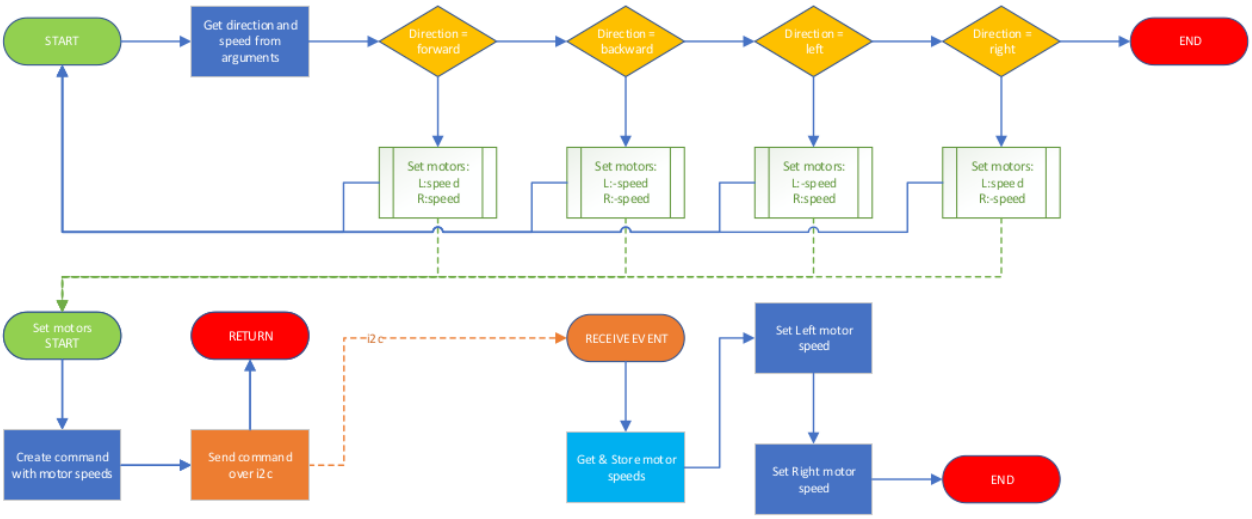


Figure 2.8: motor control code V1 flowchart

Chapter 3

Line Following System

3.1 Version 1

The design for the first version of the line following system was to split the camera feed into 3 sections, left, right and centre. Then depending which section had the most number of pixels, move the car in that direction. this was based on the infrared line following, there where 2 sensors on either side of the line would trigger a movement to either side.

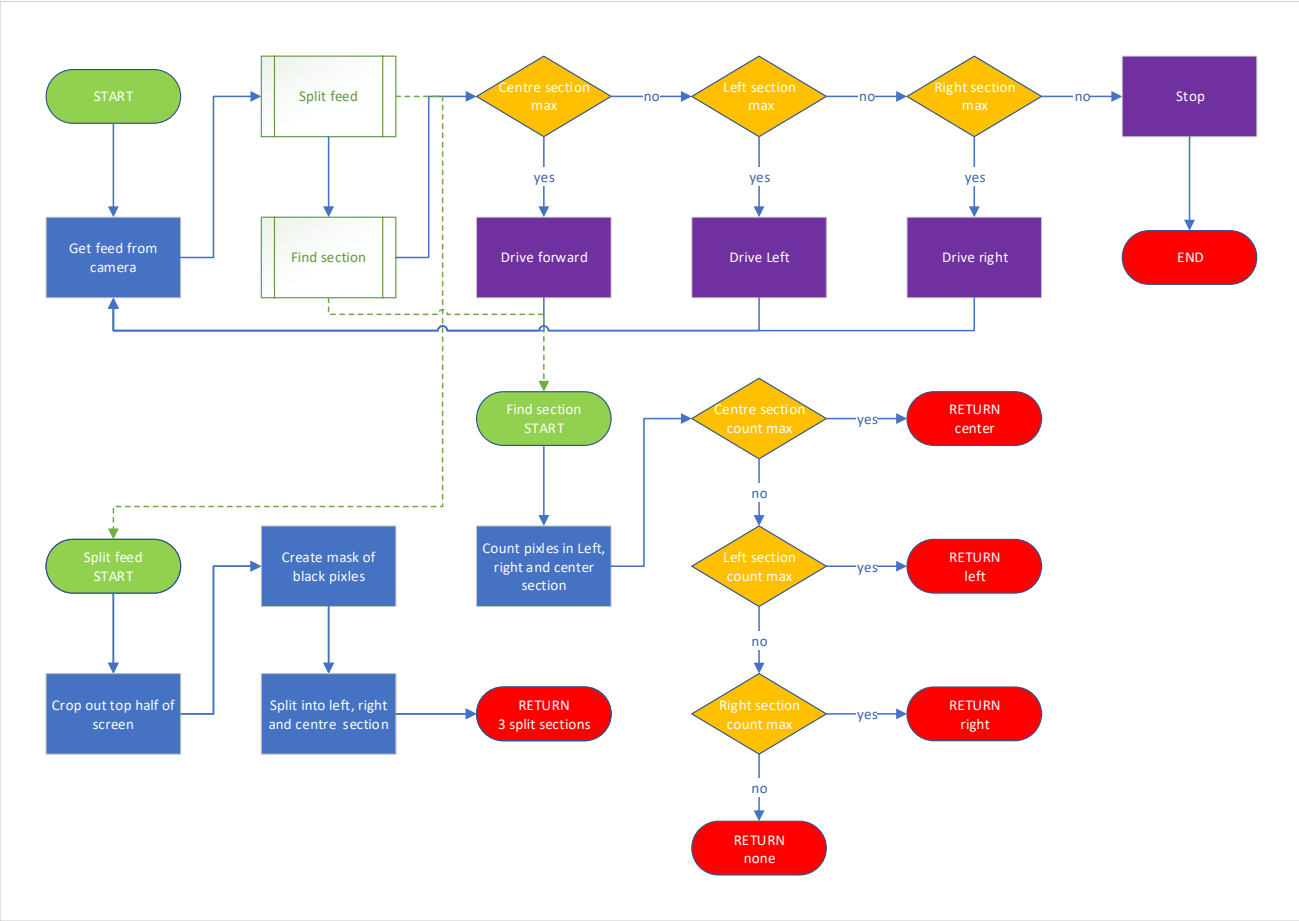


Figure 3.1: line following V1 flowchart

this version did not work as there was not enough fine adjustment to be made, this causes the vehicle to over correct often.

3.2 Version 2

Version 2 of the Line following system uses the open cv library more, it finds the x value of the line. With this value it will turn left for negative values, right for positive values, for a time based on the x values, the do nothing range and the scale, both of which is passed as runtime arguments to allow to for quick adjustment.

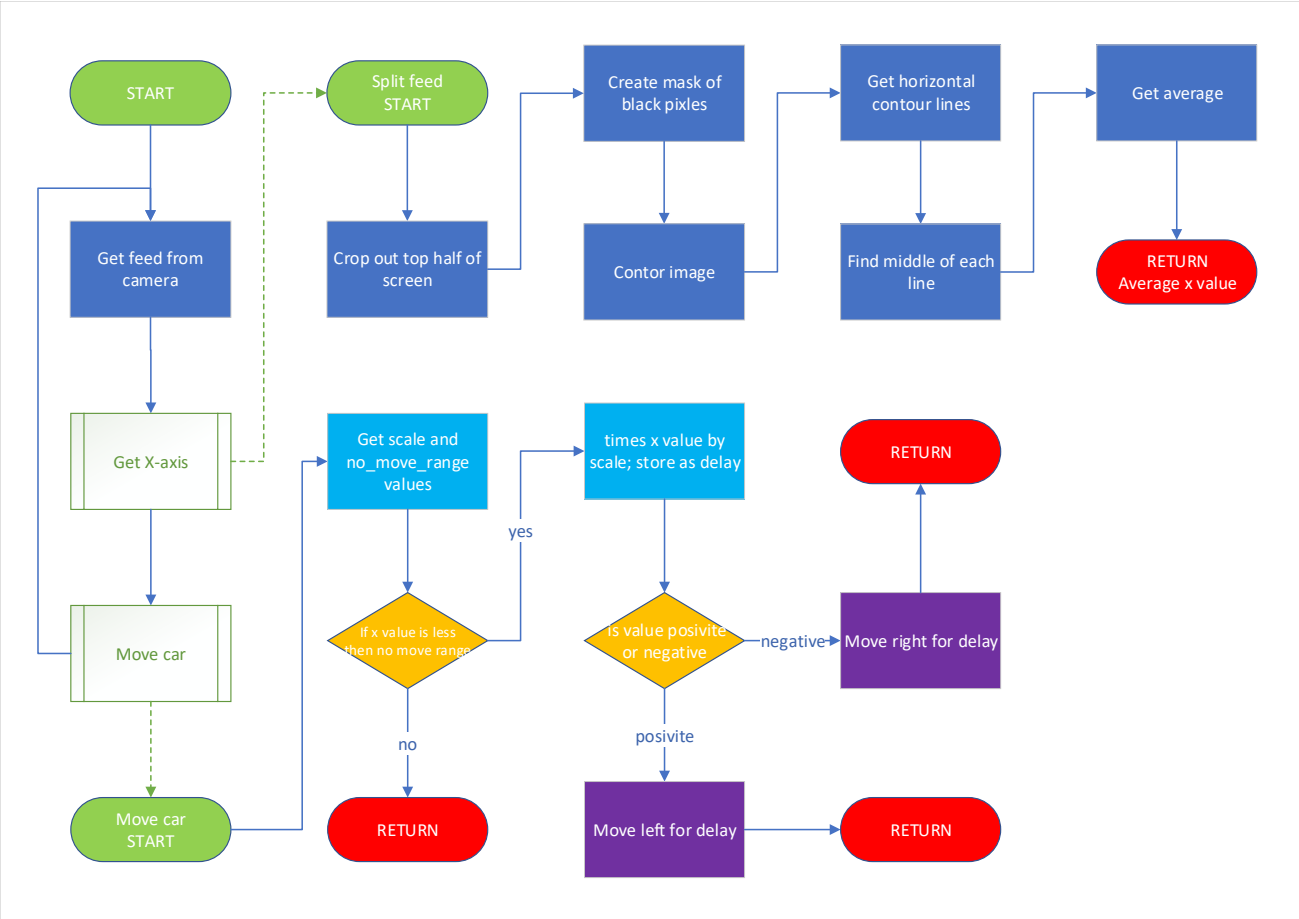


Figure 3.2: line following V2 flowchart

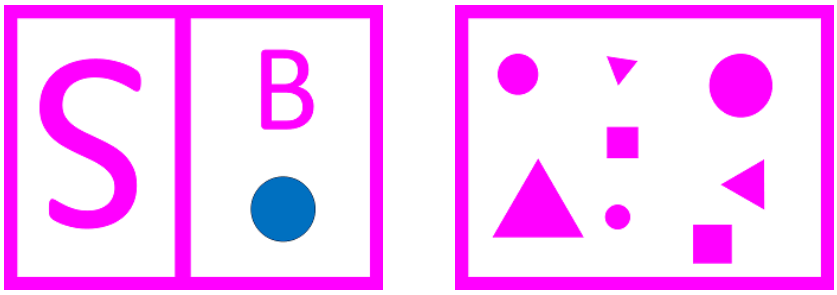
This version also over corrected, as there is the fine control is would suggest there may be a problem where the delay between the camera feed and the motor commands or the motors are not precise enough to make fine enough adjustments.

Chapter 4

Symbol Detector Systems

Among the systems required for the vehicle is a symbol detection system, this is necessary to include this system as it will control the triggering of the other systems on the car. For example, if the shape counting system was running constantly it would count shapes from random parts of the camera feed, thereby causing an inaccurate reading. A trigger for theses systems also decreases the load on the raspberry pi, this will allow for more complicated systems to be running constantly as there would be more free processing power.

For the track shown in [intro figure] there are a few symbols that are required to be detected, examples of which are shown in figure 5.2.



(a) Blue shortcut (b) Shape counter

Figure 4.1: Examples of symbols

The design of the Symbol detector is as follows: first a mask is created for pink pixel, contours are created of this map and then simplified. The largest rectangular contour is assumed to be the outside frame and is passed through a transform perspective function witch will convert the feed into a flat image the same size as the symbol files. then the system loops through all the symbols, and compares black and white versions of the symbols to the image, the most like symbol will be outputted.

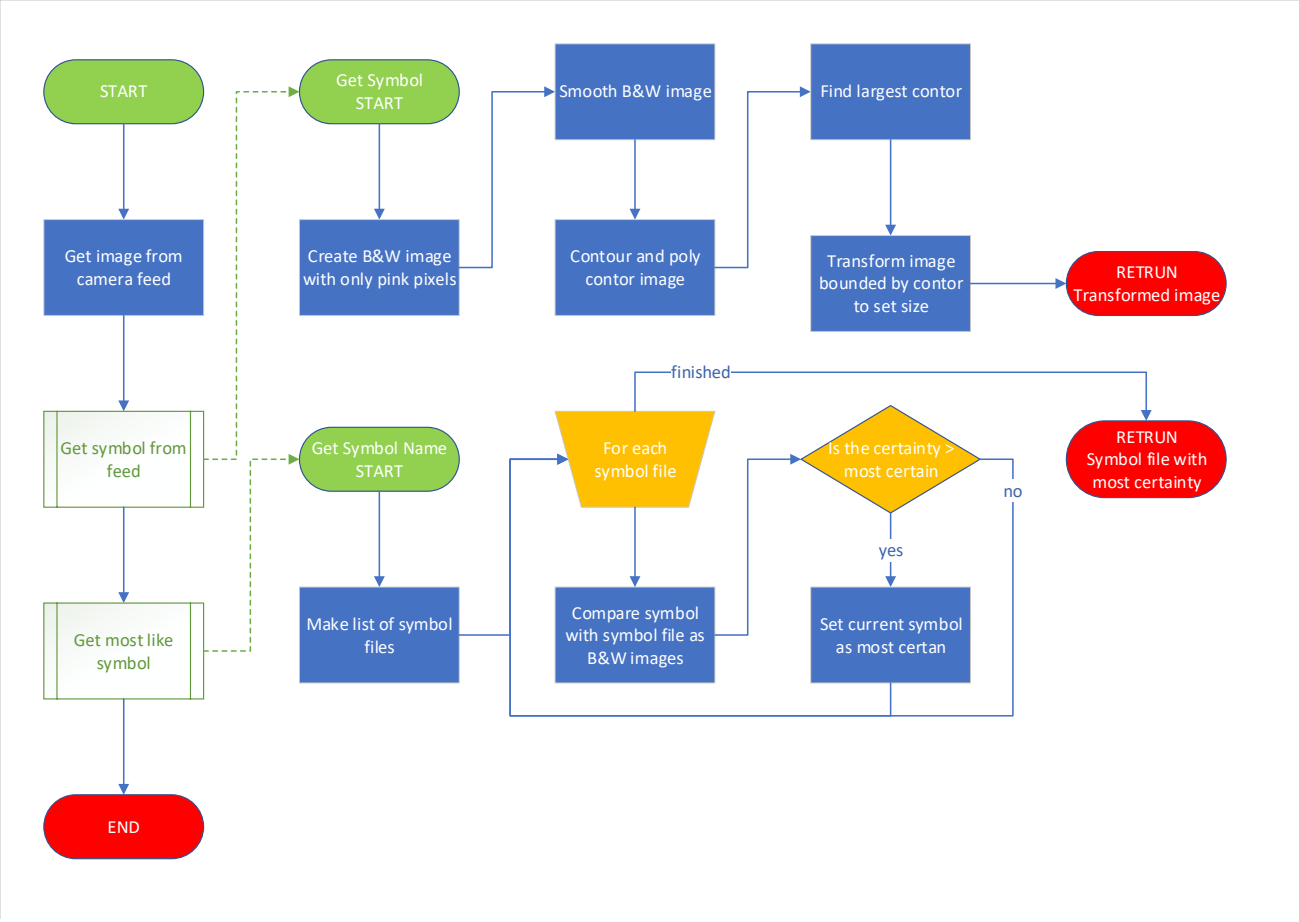
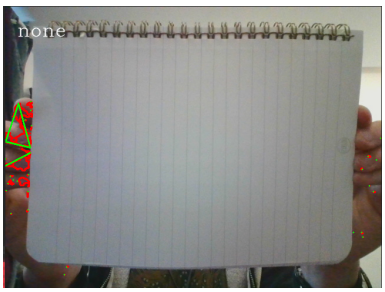
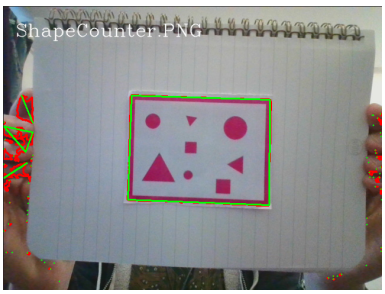


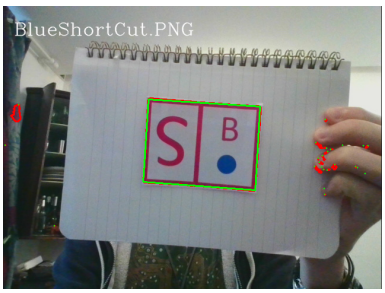
Figure 4.2: flowchart



(a) No symbol detected



(b) Shape counter detected



(c) Blue shortcut detected

(d) Stop light detected

Figure 4.3: Symbol detection tests

Chapter 5

Other Systems using Open CV

5.1 Stop Light Detection

To create the stop light detection, first the image was cropped around the green light and the red light separately, this is done to remove background of white pixels. Then the 2 images are compared to find the most white pixels, which ever has most is the light that is on

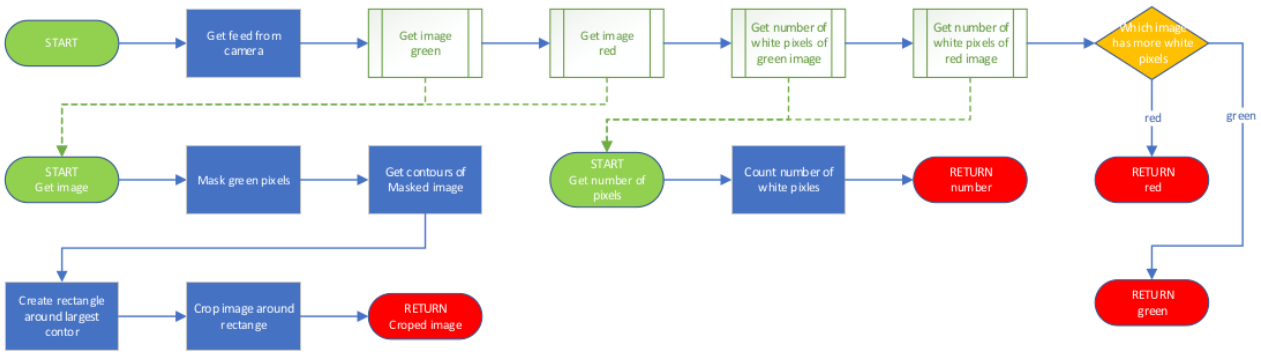
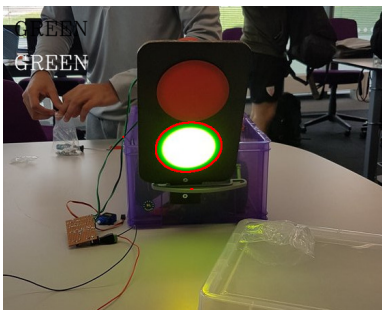
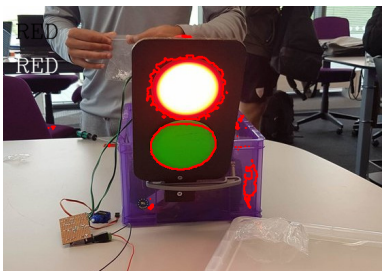


Figure 5.1: flowchart



(a) Green light detected



(b) Red light detected

Figure 5.2: Stop light detection

5.2 Shape Counting

To complete the shape counting system first the image was cropped around the blue border. then a mask of the shapes was made, the shapes where counted by creating contours of the mask and counting those contours.

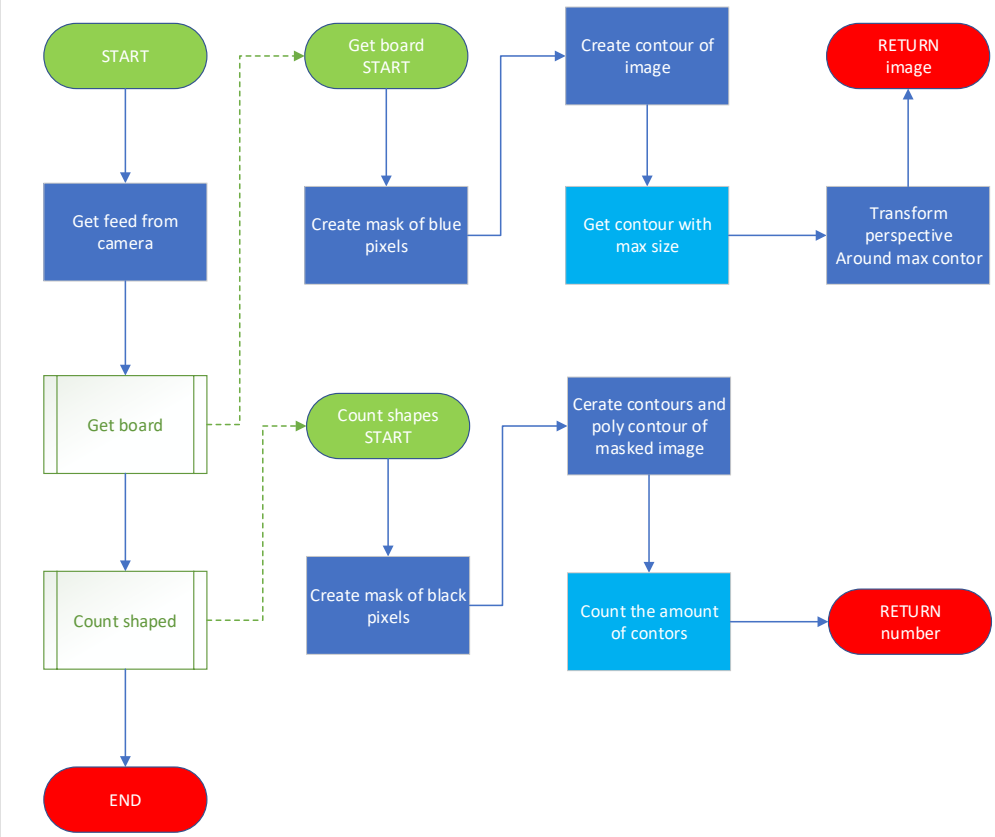
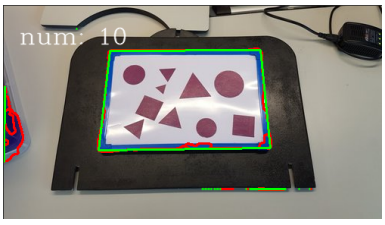
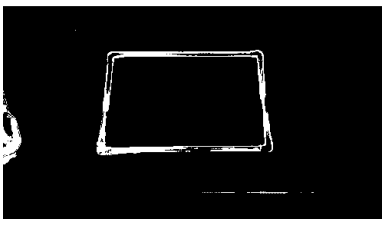


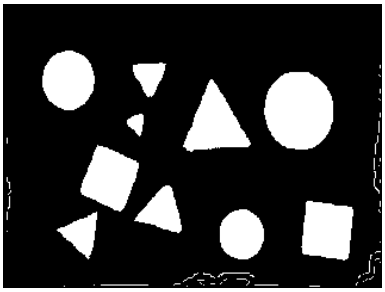
Figure 5.3: flowchart



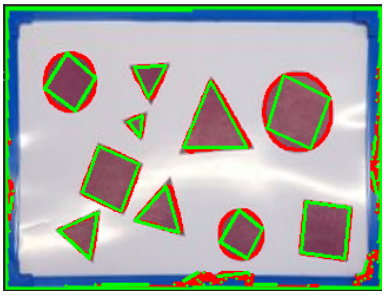
(a) Main screen



(b) Frame detected



(c) Shapes detected



(d) Shapes contor

Figure 5.4: Symbol detection tests

Chapter 6

Conclusion

The vehicles systems have been mostly completed to a high level, only system that has not been completed is line following, which could be solved with further motor control precision. The systems have yet to be consolidated into a single program to run on the track but as the line following is not complete it would not serve a useful propose.

6.1 Potential Improvements

- Remote control
  With the RF remote subsystem, a remote control could be adapted to control the main vehicle, a LCD screen could ba added to provide debugging information.

- Control motors with heading
  Using a gyroscope it would be possible to find the angle change of the car. When controlling the motors it could be easer to specify an angle change, making the line following system more accurate

- Closed loop motor control
  It was found that the only possible way to turn was by moving one set of wheels forward and one backward, when trying to turn by slowing/speeding up on set of wheels the the vehicle will remain going forward, this is assessed to be due to the extra resistance on on set of wheels. As the wheels are controlled by power input it would mean those with a higher resistance would turn slower, matching the speed of the opposite side's wheels. To solve this a closed loop system where by the motor RPM would be measured by encoders on each wheel and ran through a feedback loop to increase/ decrease the power if the motor RPM had deviated from the set value. see `https://www.electronics-tutorials.ws/systems/closed-loop-system.html`.

- web server to output diagnostics
  As the vehicle is connected to WIFI it is possible to host a web server, this could provide further controls not on the remote control.

- over the air arduino upload
  The raspberry pi can act as a full cabable PC meaning the arduinos could be updated syimply be running a cable from the USB ports on the pi to the arduino. however due to the tight layout this would not be an option. however a custom FTDI chip could be used to communicate with the arduino.

6.2 SAE levels of Autonomy

The SAE levels of Autonomy is an international standard for autonomous vehicles, the standard mostly applies to vehicles on the road making it easier for this vehicle to achieve a high level due to the lack of safety system required.

Level	Name	Narrative definition	DDT		DDT fallback	ODD
			Sustained lateral and longitudinal vehicle motion control	OEDR		
Driver performs part or all of the *DDT*						
0	No Driving Automation	The performance by the *driver* of the entire *DDT*, even when enhanced by *active safety systems*.	*Driver*	*Driver*	*Driver*	n/a
1	Driver Assistance	The *sustained* and *ODD*-specific execution by a *driving automation system* of either the *lateral* or the *longitudinal vehicle motion control* subtask of the DDT (but not both simultaneously) with the expectation that the *driver* performs the remainder of the *DDT*.	*Driver* and *System*	*Driver*	*Driver*	Limited
2	Partial Driving Automation	The *sustained* and *ODD*-specific execution by a *driving automation system* of both the *lateral* and *longitudinal vehicle motion control* subtasks of the *DDT* with the expectation that the *driver* completes the *OEDR* subtask and *supervises* the *driving automation system*.	***System***	*Driver*	*Driver*	Limited
ADS (“*System*”) performs the entire *DDT* (while engaged)			*System*	***System***	*Fallback-ready user (becomes the driver during fallback)*	Limited
3	Conditional Driving Automation	The *sustained* and *ODD*-specific performance by an *ADS* of the entire DDT with the expectation that the *DDT fallback-ready user* is *receptive* to *ADS*-issued *requests to intervene*, as well as to *DDT performance-relevant system failures* in other *vehicle* systems, and will respond appropriately.				
4	High Driving Automation	The *sustained* and *ODD*-specific performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	***System***	Limited
5	Full Driving Automation	The *sustained* and unconditional (i.e., not *ODD*-specific) performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	*System*	**Unlimited**

Figure 6.1: SAE levels of autonomy chart

The vehicle has achieved a level 4 - high Driving Automation, this is because the cars line following will allow it to sustain autonomous driving and the symbol detector will allow the vehicle to provide feedback.

Appendix

7.1 wifi access

```
Here are some files to help you use your Pi.

WiFi Password Security
======================

You first need to edit the file /etc/wpa_supplicant/wpa_supplicant.conf
changing 'USERNAME' to be your University username.

To do this use the command below

  sudo pico /etc/wpa_supplicant/wpa_supplicant.conf

Next you need to set the password however this needs to be in the file to allow
your Pi to connect to the eduroam WiFi network is plain text - in other
words anybody who has access to the file can see your password. If you have
left the default username and password as they are then anybody could log in
to your Pi and find your password.

You can use a hashed version of your password instead. A hash is a one way
function (i.e. it should not be possible to recover the password from the
hash).

If you have already configured the wpa_supplicant.conf file then run the file
set_wpa_password in the terminal/console as:

sudo ./set_wpa_password

Finding your Pi IP address
==========================

If you want to connect to your Pi with ssh or vnc you need to know its IP
address. If you do not want to hook up a display to find out the IP address
then we have put together a simple service to help you.

Once you have configured things as below, you will be able to go to

http://pifinder.eee.nottingham.ac.uk/

then enter your username and find out your IP address. This should work
everywhere on campus or in univeristy accommodation where the eduroam
network is. If you are in your own accommodation you will have to figure
something else out!

To configure the Pi finder run the following commands in the
terminal/console (if using putty you can use cut/paste)

# Now configure pi finder for this Pi:
```

```
48 sudo /usr/local/bin/pifinder_register
49
50 # Reboot
51 sudo reboot
52
53 # Now when the Pi has rebooted, go to http://pifinder.eee.nottingham.ac.uk
54 # and enter your username. Hopefully you will see your IP address in green!
```

Listing 7.1: wifi access instructions

7.2 Open CV task

Open CV color detector

Michael Bagge-Hansen

March 9, 2020

The image is first extracted from the command line as it is passed in from the user, first the program checks to see if the user had added an agrument to the calling of the program, if not it shows a message and returns -1, shown in listing 1.

```
if(argc < 2){
      cout << "Could not open or find the image!\n" << endl;
      cout << "Usage: " << argv[0] << " <Input image>" << endl;
      return -1;
   }
```

Listing 1: argument check

The inputed image is then stored as a matrix using open cv, this image is then croped to only display the center of the image. This is done because in all of the sample images the opject required to detect is in the center. The crop is done using ROI (Region of intrest) calculations. first the box which we want to crop is defined, the top coner of the ROI is defined to be $\frac{1}{4}$ the size of the image from the top corner of the image, the height and width is devined as half the size of the image. A new matrix is created to store the croped image, shown in listing 4.

```
    /* Set Region of Interest */

    cv::Rect roi;
    roi.x = src.size().width /4;
    roi.y =  src.size().height /4 + 20;
    roi.width = src.size().width /2 ;
    roi.height = src.size().height / 2 ;

    /* Crop the original image to the defined ROI */

    Mat crop = src(roi);
```

Listing 2: pixel count example

A function was created to calculate the most common color whos definition is shown below:

```
string mostCommonColor(Mat picture,int sat_tol, int val_tol);
```

Listing 3: most common color definition

This function goes cycles between colors, and calles a function, shown in listing **??** which will return the number of pixles within the range of the speciyed color,as shown in listing 4 . Once all colors have been counted, an algroithim is used to determin which color has the greatest number of pixels, shown in listing 5. the function returns the name of the color with the greates number of pixles which is then displayed to the console.

```
    //blue       100 - 130
    int blue_pix = numPixels(src,101,130,sat_tol,val_tol);

    printf("Blue: %d\n",blue_pix);
    pix[5] = blue_pix;
```

Listing 4: pixel count example

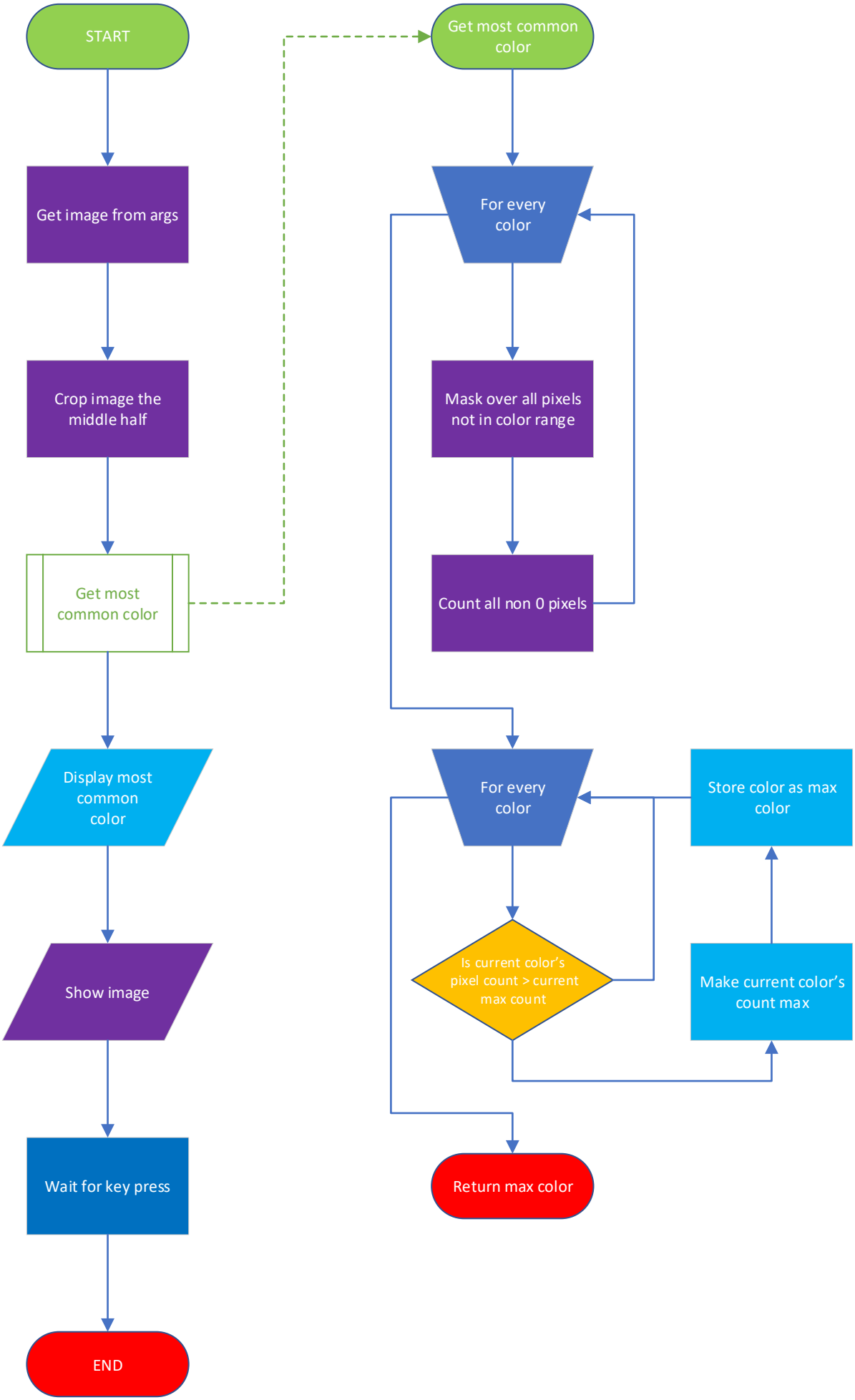
```
    int max_pix = 0, max_index = 0;

    for (int i = 0; i < 7; i++)
    {
        if(pix[i] > max_pix){
            printf("value of %d is bigger than value of %d:\t",i,max_index);
            printf("%d is bigger than %d\n",pix[i],max_pix);
            max_pix = pix[i];
            max_index = i;
        }
    }
```

Listing 5: get max algorithm

```
    int numPixels(Mat img,int low_hue,int high_hue,int sat_tol,int val_tol){

    Mat frameHSV;        // Convert the frame to HSV and apply the limits
    cvtColor(frame, frameHSV, COLOR_BGR2HSV);
    inRange(frameHSV, Scalar(low_hue, sat_tol, val_tol), Scalar(high_hue, (255-sat_tol), (255-val_tol) ),
     frameHSV);

    int pixels = 0;
    pixels = countNonZero(frameHSV);
    return pixels;
}
```

Listing 6: get number of pixles in range function

listing 6 shows a function to find the number of pixles within a range of hue values. This is done by first converting the image to HSV format, then masking the image not within the range of the hue valuses specifyed using the `inRange` function, a non zero value for saturation and value must be use when speciying the range of colors to mask so black and white pixles are not counted buy the function. Once the image is masked the function `countNonZero` can be used to find the number of pixels.


START
Get image from args
Crop image the middle half
Get most common color
Display most common color
Show image
Wait for key press
END
Get most common color
For every color
Mask over all pixels not in color range
Count all non 0 pixels
For every color
Is current color's pixel count > current max count
Store color as max color
Make current color's count max
Return max color

7.3 Motor control arduino

```
1
2 #include "H61AEE_S01.h"
3 #include <Wire.h>
4
5 #define GO_RIGHT LEFT
6 #define GO_LEFT RIGHT
7
8 //Global Variables
9 Vehicle car;
10 int ForwardSpeed = 15;
11 int TurnSpeed = 50;
12
13 char state;
14 bool forward = true;
15
16 //Main Code
17 void setup()
18 {
19   Wire.begin(0x18);
20   Serial.begin(9600); //Configure the serial port to communicate with the PC
21   Wire.onReceive(receiveEvent);
22
23   car.setupVehicle();        //Configure the car library
24   car.enableMotors(true);  //Disable the motors so they don't move
25 }
26
27 void loop() {
28 }
29
30 void receiveEvent()
31 {
32   state = Wire.read();
33
34   Serial.println(state);
35
36   car.setDirection(LEFT, forwards);
37   car.setDirection(RIGHT, forwards);
38
39   switch (state) {
40     case 'f':
41       Serial.println("going forward");
42     car.setSpeed(LEFT, ForwardSpeed );
43     car.setSpeed(RIGHT, ForwardSpeed );
44       break;
45     case 'b':
46     Serial.println("going backwards");
47     car.setDirection(LEFT, backwards);
48     car.setDirection(RIGHT, backwards);
49     car.setSpeed(LEFT, ForwardSpeed );
50     car.setSpeed(RIGHT, ForwardSpeed );
51       break;
52     case 'l':
53     Serial.println("going left");
54     car.setDirection(LEFT, backwards);
55     car.setSpeed(LEFT, TurnSpeed);
56     car.setSpeed(RIGHT, TurnSpeed);
57       break;
58     case 'r':
59     Serial.println("going right");
```

```
60      car.setDirection(RIGHT, backwards);
61      car.setSpeed(LEFT, TurnSpeed);
62      car.setSpeed(RIGHT, TurnSpeed);
63        break;
64      case 's':
65      Serial.println("stopping");
66      car.setSpeed(LEFT, 0);
67      car.setSpeed(RIGHT, 0);
68        break;
69      default:
70      Serial.println("invalid command");
71        break;
72    }
73    emptyBuffer();
74  }
75
76  void toggleDirection() {
77    if (forward) {
78      car.setDirection(BOTH, backwards);
79      forward = false;
80    } else {
81      car.setDirection(BOTH, forwards);
82      forward = true;
83    }
84  }
85
86  void emptyBuffer(void)
87  {
88    while (Wire.available())
89    {
90      Wire.read();
91    }
92  }
```

Listing 7.2: Arduino motor code v1

```
1  #include <millisDelay.h>
2  #include <Wire.h>
3
4  #include "H61AEE_S01.h"
5
6
7  #define GO_RIGHT LEFT
8  #define GO_LEFT RIGHT
9
10 //Global Variables
11 Vehicle car;
12 int ForwardSpeed = 15;
13 int TurnSpeed = 50;
14
15 int STORE_dir;
16
17 int PREV_dir;
18
19
20 millisDelay motor_delay;
21
22 //Main Code
23 void setup()
24 {
25   Wire.begin(0x18);
26   Serial.begin(9600); //Configure the serial port to communicate with the PC
```

```
27   Wire.onReceive(receiveEvent);
28
29   car.setupVehicle();        //Configure the car library
30   car.enableMotors(true);  //Disable the motors so they don't move
31 }
32
33 void loop() {
34   if (motor_delay.justFinished()) {
35     Serial.println("timer finished");
36     move_time(STORE_dir,0,true);
37   }
38 }
39
40 void receiveEvent()
41 {
42   char state = Wire.read();
43
44   Serial.print("state:");
45   Serial.println(state);
46
47   STORE_dir = state;
48
49   int time_move = Wire.read();
50
51   Serial.print("delay time:");
52     Serial.println(time_move);
53
54   if (time_move != 0) {
55     time_move = (time_move * 10);
56     Serial.print("delay time corrected:");
57     Serial.println(time_move);
58
59     move_time(state, time_move, false);
60   } else {
61     PREV_dir = state;
62     move_car(state);
63   }
64   emptyBuffer();
65 }
66
67 void move_time(char dir, int time_spent, bool timer_done) {
68   Serial.print("dir:"); Serial.println(dir);
69   Serial.print("time:"); Serial.println(time_spent);
70   Serial.print("timer done:"); Serial.println(timer_done);
71   if (timer_done) {
72     move_car(PREV_dir);
73   } else {
74     move_car(dir);
75     motor_delay.start(time_spent);
76   }
77 }
78
79
80 void move_car(char dir) {
81   car.setDirection(LEFT, forwards);
82   car.setDirection(RIGHT, forwards);
83
84   switch (dir) {
85     case 'f':
86       Serial.println("going forward");
87       car.setSpeed(LEFT, ForwardSpeed );
```

```
      car.setSpeed(RIGHT, ForwardSpeed );
      break;
    case 'b':
      Serial.println("going backwards");
      car.setDirection(LEFT, backwards);
      car.setDirection(RIGHT, backwards);
      car.setSpeed(LEFT, ForwardSpeed );
      car.setSpeed(RIGHT, ForwardSpeed );
      break;
    case 'l':
      Serial.println("going left");
      car.setDirection(LEFT, backwards);
      car.setSpeed(LEFT, TurnSpeed);
      car.setSpeed(RIGHT, TurnSpeed);
      break;
    case 'r':
      Serial.println("going right");
      car.setDirection(RIGHT, backwards);
      car.setSpeed(LEFT, TurnSpeed);
      car.setSpeed(RIGHT, TurnSpeed);
      break;
    case 's':
      Serial.println("stopping");
      car.setSpeed(LEFT, 0);
      car.setSpeed(RIGHT, 0);
      break;
    default:
      Serial.println("invalid command");
      break;
  }
}

void emptyBuffer(void)
{
  while (Wire.available())
  {
    Wire.read();
  }
}
```

Listing 7.3: Arduino motor code v1.1

```
#include <millisDelay.h>
#include <Wire.h>

#include "H61AEE_S01.h"


#define GO_RIGHT LEFT
#define GO_LEFT RIGHT

//Global Variables
Vehicle car;
int ForwardSpeed = 15;
int TurnSpeed = 50;

int CONT_dir = 's';

struct command
{
  char dir;

```

```
  bool timed = false;
  int move_time = 0;
  millisDelay move_delay;
  bool timer_start = false;
  bool timer_done = false;

  bool done = false;
};
typedef struct command Command;

Command que[8];

int que_lenght = 0;
int que_pos = 0;



//Main Code
void setup()
{
  Wire.begin(0x18);
  Serial.begin(9600); //Configure the serial port to communicate with the PC
  Wire.onReceive(receiveEvent);

  car.setupVehicle();        //Configure the car library
  car.enableMotors(true);  //Disable the motors so they don't move
}

void loop() {

  if (que_lenght == 0) {
    //que = NULL;
    return;
  }

  Serial.print("que pos:"); Serial.println(que_pos);
  Serial.print("que lenght:"); Serial.println(que_lenght);

  if (que[que_pos].timed) {
    Serial.println("is timed");
    if (que[que_pos].timer_start) {
      Serial.println("timer stated");
      if (que[que_pos].move_delay.justFinished()) {
        Serial.println("timer finished");
        que[que_pos].timer_done = true;
        execute(que_pos);
      }
    } else {
      Serial.println("executing timed command");
      execute(que_pos);
    }
  } else {
    Serial.println("executing untimed command");
    execute(que_pos);
  }

  if (que[que_pos].done) {
    que_pos++;
  }


```

```
  if (que_pos >= que_lenght) {
    que_pos = 0;
  }

}

void receiveEvent()
{

  Command new_sent;
  char state = Wire.read();

  Serial.print("\tstate:");
  Serial.println(state);

  new_sent.dir = state;

  int time_move = Wire.read();

  Serial.print("delay time:");
  Serial.println(time_move);

  if (time_move != 0) {
    time_move = time_move * 10;
    Serial.print("delay time corrected:");
    Serial.println(time_move);
    new_sent.timed = true;
    new_sent.move_time = time_move;
  }
  //add to que


  que[que_lenght] = new_sent;
  que_lenght++;

  emptyBuffer();
}

void execute(int q_pos) {
  Serial.print("dir:"); Serial.println(que[q_pos].dir);
  Serial.print("timed:"); Serial.println(que[q_pos].timed);

  if (que[q_pos].timed) {
    Serial.print("time:"); Serial.println(que[q_pos].move_time);
    Serial.print("timer started:"); Serial.println(que[q_pos].timer_start);
    if (que[q_pos].timer_done) {
      Serial.println("timer stoped");
      set_motor(CONT_dir);
      Serial.println("executed");
      que[q_pos].done = true;
      que_lenght--;
    } else {
      Serial.println("timer started");
      set_motor(que[q_pos].dir);
      que[q_pos].move_delay.start(que[q_pos].move_time);
      que[q_pos].timer_start = true;
    }
  } else {
    Serial.println("not timed");
    set_motor(que[q_pos].dir);
    CONT_dir = que[q_pos].dir;
```

```
143     Serial.println("executed");
144     que[q_pos].done = true;
145     que_lenght--;
146   }
147
148 }
149
150
151 void set_motor(char dir) {
152   car.setDirection(LEFT, forwards);
153   car.setDirection(RIGHT, forwards);
154
155   switch (dir) {
156     case 'f':
157       Serial.println("going forward");
158       car.setSpeed(LEFT, ForwardSpeed );
159       car.setSpeed(RIGHT, ForwardSpeed );
160       break;
161     case 'b':
162       Serial.println("going backwards");
163       car.setDirection(LEFT, backwards);
164       car.setDirection(RIGHT, backwards);
165       car.setSpeed(LEFT, ForwardSpeed );
166       car.setSpeed(RIGHT, ForwardSpeed );
167       break;
168     case 'l':
169       Serial.println("going left");
170       car.setDirection(LEFT, backwards);
171       car.setSpeed(LEFT, TurnSpeed);
172       car.setSpeed(RIGHT, TurnSpeed);
173       break;
174     case 'r':
175       Serial.println("going right");
176       car.setDirection(RIGHT, backwards);
177       car.setSpeed(LEFT, TurnSpeed);
178       car.setSpeed(RIGHT, TurnSpeed);
179       break;
180     case 's':
181       Serial.println("stopping");
182       car.setSpeed(LEFT, 0);
183       car.setSpeed(RIGHT, 0);
184       break;
185     default:
186       Serial.println("invalid command");
187       break;
188   }
189 }
190
191 void emptyBuffer(void)
192 {
193   while (Wire.available())
194   {
195     Wire.read();
196   }
197 }
```

Listing 7.4: Arduino motor code v2

7.4 Motor control Pi

```
1 #include <stdio.h>
```

```
2 #include <stdlib.h>
3 #include <time.h>
4
5 #include <iostream>
6 #include <string>
7 #include <sstream>
8
9 #include "pi2c.h"
10
11 int goFroward(int argc, char** argv);
12 int goBackward(int argc, char** argv);
13 int turnLeft(int argc, char** argv);
14 int turnRight(int argc, char** argv);
15
16 int toInt(char* string);
17 int setSpeeds(int left, int right);
18 int delay(double lenght);
19
20 Pi2c GBL_arduino(0x18); //Create a new object "arduino" using address "0x08"
21
22 int main(int argc, char** argv)
23 {
24     if (argc <= 1){
25         return 0;
26     }
27     char* command = argv[1];
28
29     char com = command[0];
30     printf("command:[%c]\n",com);
31
32     switch(com){
33     case 'f':
34         printf("going forwards\n");
35         goFroward(argc, argv);
36         break;
37     case 'b':
38         printf("going backwards\n");
39         goBackward(argc, argv);
40         break;
41     case 'l':
42         printf("turn left\n");
43         turnLeft(argc, argv);
44         break;
45     case 'r':
46         printf("turn right\n");
47         turnRight(argc, argv);
48         break;
49     defualt:
50         printf("nothing\n");
51         break;
52 }
53     return 0;
54 }
55
56 int goFroward(int argc, char** argv){
57     int speed = 0;
58     int time = 0;
59
60     for (int i = 0; i < argc; i++){
61         if(argv[i][0] == '-'){
62             printf("\tflag %c\n",argv[i][1]);
```

```
63              switch (argv[i][1])
64              {
65              case 't':
66                  time = toInt(argv[i+1]);
67                  printf("\t time:%d\n",time);
68                  break;
69              case 's':
70                  speed = toInt(argv[i+1]);
71                  printf("\t speed:%d\n",speed);
72                  break;
73              default:
74                  break;
75              }
76          }
77          printf("#%d: %s\n",i ,argv[i]);
78      }
79
80      //set motor speeds
81      setSpeeds(speed,speed);
82      delay(time);
83      printf("stoped\n");
84      setSpeeds(0,0);
85
86      return 0;
87
88  }
89  int goBackward(int argc, char** argv){
90      int speed = 0;
91      int time = 0;
92
93      for (int i = 0; i < argc; i++){
94          if(argv[i][0] == '-'){
95              printf("\tflag %c\n",argv[i][1]);
96              switch (argv[i][1])
97              {
98              case 't':
99                  time = toInt(argv[i+1]);
100                 printf("\t time:%d\n",time);
101                 break;
102             case 's':
103                 speed = toInt(argv[i+1]);
104                 printf("\t speed:%d\n",speed);
105                 break;
106             default:
107                 break;
108             }
109         }
110         printf("#%d: %s\n",i ,argv[i]);
111     }
112     //setmotorspeeds
113     setSpeeds(-speed,-speed);
114     delay(time);
115     printf("stoped\n");
116     setSpeeds(0,0);
117
118     return 0;
119 }
120 int turnLeft(int argc, char** argv){
121     int speed = 0;
122     int time = 0;
123
```

```
124     for (int i = 0; i < argc; i++){
125         if(argv[i][0] == '-'){
126             printf("\tflag %c\n",argv[i][1]);
127             switch (argv[i][1])
128             {
129             case 't':
130                 time = toInt(argv[i+1]);
131                 printf("\t time:%d\n",time);
132                 break;
133             case 's':
134                 speed = toInt(argv[i+1]);
135                 printf("\t speed:%d\n",speed);
136                 break;
137             default:
138                 break;
139             }
140         }
141         printf("#%d: %s\n",i ,argv[i]);
142     }
143     //setmotorspeeds
144     setSpeeds(-speed,speed);
145     delay(time);
146     printf("stoped\n");
147     setSpeeds(0,0);
148 
149     return 0;
150 }
151 int turnRight(int argc, char** argv){
152     int speed = 0;
153     int time = 0;
154 
155     for (int i = 0; i < argc; i++){
156         if(argv[i][0] == '-'){
157             printf("\tflag %c\n",argv[i][1]);
158             switch (argv[i][1])
159             {
160             case 't':
161                 time = toInt(argv[i+1]);
162                 printf("\t time:%d\n",time);
163                 break;
164             case 's':
165                 speed = toInt(argv[i+1]);
166                 printf("\t speed:%d\n",speed);
167                 break;
168             default:
169                 break;
170             }
171         }
172         printf("#%d: %s\n",i ,argv[i]);
173     }
174     setSpeeds(speed,-speed);
175     delay(time);
176     printf("stoped\n");
177     setSpeeds(0,0);
178     return 0;
179 }
180 
181 int toInt(char* string) {
182 
183   std::string s = string;
184 
```

```
185     int i =0;
186 
187   try
188   {
189     i = std::stoi(s);
190   }
191   catch (std::invalid_argument const &e)
192   {
193         return 0;
194   }
195   catch (std::out_of_range const &e)
196   {
197         return 0;
198   }
199 
200   return i;
201 }
202 
203 int setSpeeds(int left, int right)
204 {
205     char data[5] = {};
206     data[0] = 's';
207 
208     if (left < 0)
209     {
210         data[1] = '-';
211     }else{
212         data[1] = ' ';
213     }
214     data[2] = (char)abs(left);
215 
216     if (right < 0)
217     {
218         data[3] = '-';
219     }else{
220         data[3] = ' ';
221     }
222     data[4] = (char)abs(right);
223 
224     GBL_arduino.i2cWrite(data, 5);
225     return 0;
226 }
227 
228 int delay(double lenght){
229     clock_t start = clock();
230     while(1){
231         clock_t current = clock();
232         double time = (double)(current - start) / (CLOCKS_PER_SEC);
233         if(time >= lenght){
234             return 1;
235             }
236      }
237      return  0;
238     }
```

Listing 7.5: Pi motor code v1

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 
5 #include <iostream>
```

```
#include <string>
#include <sstream>

#include "pi2c.h"

int contine(char dir);
int stopped(char dir, int time);

int delay(double lenght); // in seconds
int toInt(char* string);

Pi2c GBL_arduino(0x18); //Create a new object "arduino" using address "0x08"

int main(int argc, char** argv)
{
    if (argc <= 1){
        return -1;
    }
    char* command_string = argv[1];

    char command_char = command_string[0];
    printf("command:[%c]\n",command_char);


    if(command_char == 'f' || command_char == 'b'){
        printf("continuous\n");
        contine(command_char);
    }else if (command_char == 'l' || command_char == 'r')
    {
        printf("not continuous\n");
        if (argc <= 2){
            printf("invalid command\n");
        return -1;
        }
        char* time_char = argv[2];
        int time = toInt(time_char);

        printf("delay:%d\n",time);

        stopped(command_char, time);
    }else if (command_char == 's')
    {
        printf("stoping\n");
        contine(command_char);
    }else{
        printf("invalid comand\n");
    }

}

int contine(char dir){
    printf("sending dir\n");
    GBL_arduino.i2cWriteArduinoInt(dir);

    return 0;
}

int stopped(char dir, int time){
    printf("sending dir\n");

    char data[2] = {dir,time};
```

```
67
68      GBL_arduino.i2cWrite(data, 2);
69
70      return 0;
71  }
72
73  int toInt(char* string) {
74
75    std::string s = string;
76
77      int i =0;
78
79    try
80    {
81      i = std::stoi(s);
82    }
83    catch (std::invalid_argument const &e)
84    {
85          return 0;
86    }
87    catch (std::out_of_range const &e)
88    {
89          return 0;
90    }
91
92    return i;
93  }
94
95  int delay(double lenght){
96      clock_t start = clock();
97      while(1){
98          clock_t current = clock();
99          double time = (double)(current - start) / (CLOCKS_PER_SEC);
100         if(time >= lenght){
101             return 1;
102             }
103      }
104      return  0;
105 }
```

Listing 7.6: Pi motor code v2

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  #include <iostream>
6  #include <string>
7  #include <sstream>
8
9  #include "pi2c.h"
10
11 int contine(char dir);
12 int stopped(char dir, int time);
13
14 int delay(double lenght); // in seconds
15 int toInt(char* string);
16
17 Pi2c GBL_arduino(0x18); //Create a new object "arduino" using address "0x08"
18
19 int main(int argc, char** argv)
20 {
```

```
21      if (argc <= 1){
22          return -1;
23      }
24      char* command_string = argv[1];
25
26      char command_char = command_string[0];
27      printf("command:[%c]\n",command_char);
28
29      int time = 0;
30
31      if(argc >= 3){
32      char* time_char = argv[2];
33      time = toInt(time_char);
34      }
35
36      printf("delay:%d\n",time);
37
38      if(time > 0){
39
40          printf("not continuous\n");
41
42          stopped(command_char, time);
43
44          stopped('s', 20);
45
46
47      }else
48      {
49          printf("continuous\n");
50          contine(command_char);
51      }
52
53
54  }
55
56  int contine(char dir){
57      printf("sending dir\n");
58      GBL_arduino.i2cWriteArduinoInt(dir);
59
60      return 0;
61  }
62
63  int stopped(char dir, int time){
64      printf("sending dir\n");
65
66      char data[2] = {dir,time};
67
68      GBL_arduino.i2cWrite(data, 2);
69
70      return 0;
71  }
72
73  int toInt(char* string) {
74
75    std::string s = string;
76
77      int i =0;
78
79    try
80    {
81      i = std::stoi(s);
```

```
  }
  catch (std::invalid_argument const &e)
  {
        return 0;
  }
  catch (std::out_of_range const &e)
  {
        return 0;
  }

  return i;
}

int delay(double lenght){
    clock_t start = clock();
    while(1){
        clock_t current = clock();
        double time = (double)(current - start) / (CLOCKS_PER_SEC);
        if(time >= lenght){
            return 1;
            }
     }
     return  0;
}
```

Listing 7.7: Pi motor code v2.1

7.5 sensors

```
#include <Wire.h>

#include <Servo.h>

#include <NewPing.h>

Servo myservo;  // create servo object to control a servo

int sensor = -1;

#define SONAR_NUM 2      // Number of sensors.
#define MAX_DISTANCE 200 // Maximum distance (in cm) to ping.

#define BACK_SENSOR 0
#define LEFT_SENSOR 1

NewPing sonar[SONAR_NUM] = {   // Sensor object array.
  NewPing(2, 3, MAX_DISTANCE), // Each sensor's trigger pin, echo pin, and max
    distance to ping.
  NewPing(4, 5, MAX_DISTANCE)
};

int back_distance = 0;
int left_distance = 0;



void setup() {
  Wire.begin(8);                // join i2c bus with address #8
  Wire.onRequest(requestEvent); // register event
  Wire.onReceive(receiveEvent); // register event
  Serial.begin(9600);           // start serial for output
```

```
32
33    myservo.attach(6);  // attaches the servo on pin 9 to the servo object
34  }
35
36  void loop() {
37    left_distance = sonar[LEFT_SENSOR].ping_cm();
38    back_distance = sonar[BACK_SENSOR].ping_cm();
39  }
40
41  // function that executes whenever data is requested by master
42  // this function is registered as an event, see setup()
43  void requestEvent() {
44    Serial.print("request");
45
46    if (sensor != -1) {
47      switch (sensor) {
48        case BACK_SENSOR:
49            Serial.print("back:");
50
51          Wire.write(back_distance);
52          break;
53        case LEFT_SENSOR:
54            Serial.print("left:");
55
56          Wire.write(left_distance);
57          break;
58      }
59      sensor = -1;
60    }
61
62  }
63
64  // function that executes whenever data is received from master
65  // this function is registered as an event, see setup()
66  void receiveEvent(int howMany) {
67    char flag = 0;
68    int input[howMany - 1];
69    int current = 0;
70    while (Wire.available()) { // loop through all but the last
71      Serial.print("current:");
72      Serial.println(current);
73      if (current == 0) {
74        flag = Wire.read();
75        current++;
76      } else {
77        input[current] = Wire.read(); // receive byte as a character
78        Serial.println(input[current]);         // print the character
79        current++;
80      }
81    }
82
83    Serial.print("flag:{");
84    Serial.print(flag);
85    Serial.println("}");
86
87    if (flag == 's') {
88      Serial.println("setting servo:");
89      int val = input[1];
90      setServo(val);
91    } else  if (flag == 'b') {
92      Serial.println("getting back sensor:");
```

```
93      sensor = BACK_SENSOR;
94      delay(100);
95    } else  if (flag == 'l') {
96      Serial.println("getting left sensor:");
97      sensor = LEFT_SENSOR;
98    }
99
100 }
101
102 void setServo(int val) {
103   Serial.print("val: ");
104   Serial.println(val);
105
106   myservo.write(val);                  // sets the servo position
107   delay(15);                           // waits for the servo to get there
108 }
109
110 int getInt(int input[], int in_size) {
111   int mult = 1;
112   int sum = 0;
113   int in_int = 0;
114   int expon = 0;
115
116   for (int i = 0; i < in_size; i++) {
117     //    Serial.print("I: ");
118     //    Serial.println(i);
119
120     if (input[i] == 10) {
121       return sum;
122     }
123     expon = (in_size - 2) - i;
124     //    Serial.print("expon: ");
125     //    Serial.println(expon);
126
127     mult = power(10, expon);
128     //    Serial.print("mult: ");
129     //    Serial.println(mult);
130
131     in_int = input[i] - 48;
132     //    Serial.print("int: ");
133     //    Serial.println(in_int);
134
135     sum = sum + (in_int * mult);
136     //    Serial.print("sum: ");
137     //    Serial.println(sum);
138
139   }
140
141   return sum;
142
143 }
144
145 int power(int bace, int expon) {
146   if (expon == 0) {
147     return 1;
148   }
149   int sum = bace;
150   //Serial.println(sum);
151   for (int i = 1; i <= (expon - 1); i++) {
152     sum = sum * bace;
153     //Serial.println(sum);
```

```
154   }
155
156   return sum;
157 }
```

Listing 7.8: arduino sensor code

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  #include <iostream>
6  #include <string>
7  #include <sstream>
8
9  #include "pi2c.h"
10
11 void set_servo(int pos);
12 int get_distance(char pos);
13
14 int delay(double lenght);
15
16 Pi2c GBL_arduino(0x08); //Create a new object "arduino" using address "0x18"
17
18 int main(int argc, char** argv)
19 {
20     if (argc <= 1){
21         return 0;
22     }
23     char* command = argv[1];
24
25     char com = command[0];
26     printf("command:[%c]\n",com);
27
28     switch(com){
29     case 's':
30         printf("setting servo\n");
31         set_servo(argv[2]);
32         break;
33     case 'b':
34         printf("get back sensor\n");
35         get_distance('b');
36         break;
37     case 'l':
38         printf("get left sensor\n");
39         get_distance('l');
40         break;
41     defualt:
42         printf("nothing\n");
43         break;
44     }
45     return 0;
46 }
47
48 void set_servo(int pos)
49 {
50     int data[5] = {};
51     data[0] = 's';
52
53     data[1] = pos;
54
55     printf("pos:%d",data[1]);
```

```
56
57      GBL_arduino.i2cWrite(data, 5);
58  }
59
60  int get_distance(char pos)
61  {
62      if  (pos == 'b') {
63      printf("getting back:");
64      GBL_arduino.i2cWrite('b', 1);
65    } else if (pos == 'l') {
66      printf("getting back:");
67      GBL_arduino.i2cWrite('l', 1);
68    }
69
70      int data = 0;
71      arduino.i2cRead(data,1);
72
73      return data;
74  }
75
76  int delay(double lenght){
77      clock_t start = clock();
78      while(1){
79          clock_t current = clock();
80          double time = (double)(current - start) / (CLOCKS_PER_SEC);
81          if(time >= lenght){
82              return 1;
83              }
84       }
85       return  0;
86      }
```

Listing 7.9: Pi sensor code

7.6 Line following

```
1  //EEEE1002 Bench 41 2019/20
2
3  #include "opencv2/core.hpp"
4  #include "opencv2/imgproc.hpp"
5  #include "opencv2/highgui.hpp"
6  #include "opencv2/videoio.hpp"
7
8  #include <sys/types.h>
9  #include <dirent.h>
10 #include <errno.h>
11 #include <vector>
12 #include <string>
13 #include <iostream>
14
15
16 using namespace std;
17 using namespace cv;
18
19 Mat LineFollowFunction(Mat image);
20
21 // int main()
22 // {
23 //      cout << "Built with OpenCV " << CV_VERSION << endl;
24 //      Mat image;
25 //      VideoCapture capture;
```

```
26 //      capture.open(0);
27 //      if(capture.isOpened())
28 //      {
29 //          cout << "Capture is opened" << endl;
30 //          for(;;)
31 //          {
32 //              capture >> image;
33 //              if(image.empty())
34 //                  break;
35 //              LineFollowFunction(image);
36 //              imshow("Sample", image);
37 //              if(waitKey(10) >= 0)
38 //                  break;
39 //          }
40 //      }
41 //      else
42 //      {
43 //          cout << "No capture" << endl;
44 //          image = Mat::zeros(480, 640, CV_8UC1);
45 //          //drawText(image);
46 //          imshow("Sample", image);
47 //          waitKey(0);
48 //      }
49 //      return 0;
50 // }
51
52 int main(int argc, char** argv )
53 {
54     if ( argc != 2 )
55     {
56         printf("usage: DisplayImage.out <Image_Path>\n");
57         return -1;
58     }
59     Mat image;
60     image = imread( argv[1], 1 );
61     if ( !image.data )
62     {
63         printf("No image data \n");
64         return -1;
65     }
66     namedWindow("Display Image", WINDOW_AUTOSIZE );
67     LineFollowFunction(image);
68     imshow("Display Image", image);
69     waitKey(0);
70     return 0;
71 }
72
73 Mat LineFollowFunction(Mat image)
74 {
75
76     //Crop the stream
77     Mat StreamCropped;
78     cv::Size StreamDimensions  = image.size();
79     int StreamWidth = StreamDimensions.width;
80     int StreamHeight = StreamDimensions.height;
81     StreamCropped = image(Rect(0,(StreamHeight/2),StreamWidth,(StreamHeight/2)));
82     //namedWindow ("Cropped Stream", CV_WINDOW_FREERATIO);
83     //imshow("Cropped Stream", StreamCropped);
84
85
86     //Convert to grayscale
```

```
87      Mat StreamGrayscale;
88      Mat StreamBlurred;
89      cvtColor(StreamCropped, StreamGrayscale, COLOR_BGR2GRAY);
90      //imshow("Stream Grayscale", StreamGrayscale);
91
92
93      //Gaussian Blur is added
94      GaussianBlur(StreamGrayscale, StreamBlurred, Size(5,5),0);
95      //namedWindow ("Stream Blurred", CV_WINDOW_FREERATIO);
96      //imshow("Stream Blurred", StreamBlurred);
97
98
99      //Threashold the stream
100     Mat StreamThresholded;
101     threshold(StreamBlurred, StreamThresholded, 60, 255, THRESH_BINARY_INV);
102     //namedWindow ("Stream Thresholded", CV_WINDOW_FREERATIO);
103     //imshow("Stream Thresholded", StreamThresholded);
104
105
106     //Contour the stream
107     std::vector<std::vector<cv::Point>>contours;// Variable for list of contours
108     std::vector<Vec4i>hierarchy;// Variable for image topology data
109     cv::findContours(StreamThresholded,contours,hierarchy,RETR_EXTERNAL,
    CHAIN_APPROX_SIMPLE,Point(0,0));// Calculate the contours and store them
110
111     /*  if (contours.size() < 1){
112           Mat symb = Mat::zeros(240, 320, CV_8UC3);
113           return symb;
114       } */
115
116     Mat StreamContoured = StreamThresholded;
117     for(int i =0; i < contours.size(); i++) // Loop through the contours
118     {
119         drawContours(StreamContoured,contours,i,Scalar(0,0,255),2,LINE_8,noArray
    (),0,Point());// Draw each in red
120     }
121
122
123     imshow("Contoured", StreamContoured);
124
125
126
127
128
129     //Bodge Below:
130
131     //Left side
132     Mat LeftStream;
133     Mat LeftStreamOutput;
134     int LeftStreamCount;
135     LeftStream = StreamThresholded(Rect(0,(StreamHeight/2),(StreamWidth/3),(
    StreamHeight/2)));
136     inRange(LeftStream, Scalar(150), Scalar(255), LeftStreamOutput);
137     LeftStreamCount = countNonZero(LeftStreamOutput);
138
139     //Centre
140     Mat CentreStream;
141     Mat CentreStreamOutput;
142     int CentreStreamCount;
143     CentreStream = StreamThresholded(Rect(0,(StreamHeight/2),(StreamWidth/3),(
    StreamHeight/2)));
```

```
    inRange(CentreStream, Scalar(150), Scalar(255), CentreStreamOutput);
    CentreStreamCount = countNonZero(CentreStreamOutput);

    //Right side
    Mat RightStream;
    Mat RightStreamOutput;
    int RightStreamCount;
    RightStream = StreamThresholded(Rect(0,(StreamHeight/2),(StreamWidth/3),(StreamHeight/2)));
    inRange(RightStream, Scalar(150), Scalar(255), RightStreamOutput);
    RightStreamCount = countNonZero(RightStreamOutput);

    //Comparison
    if(CentreStreamCount > LeftStreamCount && CentreStreamCount > RightStreamCount)
    {
        printf("Forwards\n");
    }

    if(RightStreamCount > LeftStreamCount && RightStreamCount > CentreStreamCount)
    {
        printf("Turn Right\n");
    }

    if(LeftStreamCount > CentreStreamCount && LeftStreamCount > RightStreamCount)
    {
        printf("Turn Left\n");
    }

    return (StreamCropped);
}
```

Listing 7.10: Line following V1

```
#include <stdio.h>

#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/videoio.hpp"


using namespace cv;

int line_xval(Mat img);

int main(int argc, char** argv )
{
    if ( argc != 2 )
    {
        printf("usage: DisplayImage.out <Image_Path>\n");
        return -1;
    }
    Mat image;
    image = imread( argv[1], 1 );
    if ( !image.data )
    {
        printf("No image data \n");
        return -1;
    }
    namedWindow("Display Image", WINDOW_AUTOSIZE );
```

```
28      int value = line_xval(image);
29      printf("x_value:%d\n",value);
30      imshow("Display Image", image);
31
32      waitKey(0);
33      return 0;
34  }
35
36  int line_xval(Mat img){
37      //Crop the stream
38      Mat StreamCropped;
39      cv::Size StreamDimensions  = img.size();
40      int StreamWidth = StreamDimensions.width;
41      int StreamHeight = StreamDimensions.height;
42      StreamCropped = img(Rect(0,(StreamHeight/2),StreamWidth,(StreamHeight/2)));
43      //namedWindow ("Cropped Stream", CV_WINDOW_FREERATIO);
44      imshow("Cropped Stream", StreamCropped);
45
46      //Convert to grayscale
47      Mat StreamGrayscale;
48      cvtColor(StreamCropped, StreamGrayscale, COLOR_BGR2GRAY);
49      //imshow("Stream Grayscale", StreamGrayscale);
50
51
52      //Gaussian Blur is added
53      Mat StreamBlurred;
54      GaussianBlur(StreamGrayscale, StreamBlurred, Size(5,5),0);
55      //namedWindow ("Stream Blurred", CV_WINDOW_FREERATIO);
56      //imshow("Stream Blurred", StreamBlurred);
57
58
59      //Threashold the stream
60      Mat StreamThresholded;
61      threshold(StreamBlurred, StreamThresholded, 100, 255, THRESH_BINARY_INV);
62      //namedWindow ("Stream Thresholded", CV_WINDOW_FREERATIO);
63      imshow("Stream Thresholded", StreamThresholded);
64
65
66      //Contour the stream
67      std::vector<std::vector<cv::Point>>contours;// Variable for list of contours
68      std::vector<Vec4i>hierarchy;// Variable for image topology data
69      cv::findContours(StreamThresholded,contours,hierarchy,RETR_EXTERNAL,
    CHAIN_APPROX_SIMPLE,Point(0,0));// Calculate the contours and store them
70
71      /*  if (contours.size() < 1){
72            Mat symb = Mat::zeros(240, 320, CV_8UC3);
73            return symb;
74        } */
75
76      for(int i =0; i < contours.size(); i++) // Loop through the contours
77      {
78          drawContours(StreamCropped,contours,i,Scalar(0,0,255),2,LINE_8,noArray()
    ,0,Point());// Draw each in red
79      }
80
81      printf("contros:%d\n",contours.size());
82
83      std::vector<std::vector<cv::Point>>approxedcontours(contours.size());// Array
     for new contours
84
85      for(int i=0;i<contours.size();i++){
```

```
86          cv::approxPolyDP(contours[i],approxedcontours[i],10,true);// Approximate
    the contour
87      }
88
89      int max_area =0, max_index = 0,area =0;
90
91
92      for(int i =0;i <approxedcontours.size();i++)// Loop through the contours
93      {
94          area = contourArea(approxedcontours[i]);
95          if (area > max_area){
96              max_area = area;
97              max_index = i;
98          }
99      }
100
101     printf("corners:%d\n",approxedcontours[max_index].size());
102
103     int x_sum=0;
104
105     cv::Point pt1 = approxedcontours[max_index][1], pt2 = approxedcontours[
    max_index][2];
106     cv::line(StreamCropped, pt1, pt2, cv::Scalar(255, 0, 0), 3, 8, 0);
107     x_sum = x_sum + pt1.x + pt2.x;
108
109     printf("red_line: A: x[%d],y[%d] B: x[%d],y[%d]",pt1.x,pt1.y,pt2.x,pt2.y);
110
111     pt1 = approxedcontours[max_index][3]; pt2 = approxedcontours[max_index][0];
112     cv::line(StreamCropped, pt1, pt2, cv::Scalar(0, 0 , 255), 3, 8, 0);
113     x_sum = x_sum + pt1.x + pt2.x;
114
115     printf("blue_line: A: x[%d],y[%d] B: x[%d],y[%d]",pt1.x,pt1.y,pt2.x,pt2.y);
116
117     int x_avg = x_sum/4;
118
119     Point img_center;
120     img_center.x = x_avg;
121     img_center.y = 230;
122
123     circle(StreamCropped,img_center,5,cv::Scalar(0, 255 , 0), 1,8,0);
124     //circle(StreamCropped,line_center,5,cv::Scalar(0, 255 , 0), 1,8,0);
125
126     printf("total x: %d\n", StreamDimensions.width);
127     printf("line x: %d\n", x_avg);
128
129     int x_delta = x_avg - (StreamDimensions.width/2);
130
131
132     return x_delta;
133 }
```

Listing 7.11: Line following V2

7.7 Symbol detect

```
1 #include "opencv2/core.hpp"
2 #include "opencv2/imgproc.hpp"
3 #include "opencv2/highgui.hpp"
4 #include "opencv2/videoio.hpp"
5
6 #include <sys/types.h>
```

```
#include <dirent.h>
#include <errno.h>
#include <vector>
#include <string>
#include <iostream>


using namespace cv;
using namespace std;

String get_symbol(Mat img);

Mat get_img(Mat img);


Mat transformPerspective(std::vector<Point> boundingContour, Mat frame, int
    x_size, int y_size);
Point findContourCentre(std::vector<cv::Point> contour);
float compareImages(Mat cameraImage, Mat librarySymbol);


void drawText(Mat & image, String text);

int main()
{
    cout << "Built with OpenCV " << CV_VERSION << endl;
    Mat image;
    VideoCapture capture;
    capture.open(0);
    if(capture.isOpened())
    {
        cout << "Capture is opened" << endl;
        for(;;)
        {
            capture >> image;
            if(image.empty()){
                break;
            }
            String symbol = get_symbol(image);
            drawText(image,symbol);

            imshow("Sample", image);
            if(waitKey(10) >= 0)
                break;
        }
    }
    else
    {
        cout << "No capture" << endl;
        image = Mat::zeros(480, 640, CV_8UC1);
        drawText(image,"hello");
        imshow("Sample", image);
        waitKey(0);
    }
    return 0;
}

// int main(int argc, char** argv )
// {
//     if ( argc != 2 )
//     {
```

```
67 //          printf("usage: DisplayImage.out <Image_Path>\n");
68 //          return -1;
69 //      }
70 //      Mat image;
71 //      image = imread( argv[1], 1 );
72 //      if ( !image.data )
73 //      {
74 //          printf("No image data \n");
75 //          return -1;
76 //      }
77 //      namedWindow("Display Image", WINDOW_AUTOSIZE );
78 
79 //      String symbol = get_symbol(image);
80 //      drawText(image,symbol);
81 
82 //      imshow("Display Image", image);
83 //      waitKey(0);
84 //      return 0;
85 // }
86 
87 String get_symbol(Mat img){
88     Mat image = get_img(img);
89 
90     string dir = "symbols";
91      vector<string> files = vector<string>();
92 
93     DIR *dp;
94     struct dirent *dirp;
95     if((dp  = opendir(dir.c_str())) == NULL) {
96         cout << "Error(" << errno << ") opening " << dir << endl;
97         return "Error";
98     }
99     string temp;
100 
101     while ((dirp = readdir(dp)) != NULL) {
102         temp = dirp->d_name;
103         if(temp.find(".PNG") != std::string::npos){
104             files.push_back(string(dirp->d_name));
105         }
106     }
107     closedir(dp);
108 
109     Mat symbol;
110 
111     float max_certanty =0,certanty;
112     int most_certain_file = 0;
113 
114     for (unsigned int i = 0;i < files.size();i++) {
115         //cout << files[i] << endl;
116 
117         String symbol_file = "symbols/" + files[i];
118 
119         symbol = imread(symbol_file,1);
120 
121         cv::Size img_sz = image.size();
122         int imageWidth = img_sz.width;
123         int imageHeight = img_sz.height;
124 
125         cv::Size sym_sz = symbol.size();
126         int symWidth = sym_sz.width;
127         int symHeight = sym_sz.height;
```

```
128
129         //printf("image: height: %d\t width: %d\n",imageWidth,imageHeight);
130         //printf("symbol [%s]: height: %d\t width: %d\n",symbol_file.c_str() ,
    symWidth,symHeight);
131
132         if(imageWidth > 0 && symWidth > 0 && imageHeight > 0 && symHeight > 0){
133
134             Mat image_BW;        // Convert the frame to HSV and apply the limits
135             cvtColor(image, image_BW, COLOR_BGR2HSV);
136             int low_hue = 150, high_hue = 180;
137             inRange(image_BW, Scalar(low_hue, 70, 80), Scalar(high_hue, (230),
    (255) ), image_BW);
138
139             Mat kernel=cv::getStructuringElement(MORPH_ELLIPSE, Size(5,5));
140             cv::morphologyEx(image_BW,image_BW,MORPH_CLOSE, kernel);
141
142             Mat symbol_BW;        // Convert the frame to HSV and apply the limits
143             cvtColor(symbol, symbol_BW, COLOR_BGR2HSV);
144             low_hue = 150, high_hue = 180;
145             inRange(symbol_BW, Scalar(low_hue, 0, 0), Scalar(high_hue, (255),
    (255) ), symbol_BW);
146
147             certanty  = compareImages(image_BW,symbol_BW);
148
149             if(certanty > max_certanty){
150                 max_certanty = certanty;
151                 most_certain_file = i;
152             }
153
154             //imshow("BW symb",symbol_BW);
155             //imshow("BW img",image_BW);
156
157         }
158     }
159
160     if(max_certanty < 50){
161         return "none";
162     }
163
164     String returnd = files[most_certain_file];
165
166     return returnd;
167
168 }
169
170 Mat get_img(Mat img){
171     Mat imgHSV;        // Convert the frame to HSV and apply the limits
172     cvtColor(img, imgHSV, COLOR_BGR2HSV);
173     int low_hue = 150, high_hue = 180;
174     inRange(imgHSV, Scalar(low_hue, 70, 80), Scalar(high_hue, (230), (255) ),
    imgHSV);
175
176     //imshow("pink cam",imgHSV);
177
178     Mat imgSmooth;
179
180     Mat kernel=cv::getStructuringElement(MORPH_ELLIPSE, Size(5,5));
181     cv::morphologyEx(imgHSV,imgSmooth,MORPH_CLOSE, kernel);
182
183     //imshow("smooth cam",imgSmooth);
184
```

```
185     std::vector<std::vector<cv::Point>>contours;// Variable for list of contours
186     std::vector<Vec4i>hierarchy;// Variable for image topology data
187     cv::findContours(imgSmooth,contours,hierarchy,RETR_EXTERNAL,
    CHAIN_APPROX_SIMPLE,Point(0,0));// Calculate the contours and store them
188
189     if (contours.size() < 1){
190         Mat symb = Mat::zeros(240, 320, CV_8UC3);
191         return symb;
192     }
193
194
195     for(int i =0;i < contours.size();i++)// Loop through the contours
196     {
197         drawContours(img,contours,i,Scalar(0,0,255),2,LINE_8,noArray(),0,Point())
    ;// Draw each in red
198     }
199
200     std::vector<std::vector<cv::Point>>approxedcontours(contours.size());// Array
     for new contours
201
202     for(int i=0;i<contours.size();i++){
203         cv::approxPolyDP(contours[i],approxedcontours[i],30,true);// Approximate
    the contour
204     }
205
206     for(int i =0;i <contours.size();i++)// Loop through the contours
207     {
208         drawContours(img,approxedcontours,i,Scalar(0,255,0),2,LINE_8,noArray(),0,
    Point());// Draw each in green
209     }
210
211     int max_area =0, max_index = 0,area =0;
212
213
214     for(int i =0;i <approxedcontours.size();i++)// Loop through the contours
215     {
216         area = contourArea(approxedcontours[i]);
217         if (area > max_area){
218             max_area = area;
219             max_index = i;
220         }
221     }
222
223     //int corners =  approxedcontours[max_index].size();
224
225     //printf("corners:%d",corners);
226
227
228
229     Mat symb = transformPerspective(approxedcontours[max_index],img,320,240);
230
231     // cv::Rect roi;
232
233
234     // roi.x = boundingRect(approxedcontours[max_index]).x;
235     // roi.y = boundingRect(approxedcontours[max_index]).y;
236     // roi.width = boundingRect(approxedcontours[max_index]).width;
237     // roi.height = boundingRect(approxedcontours[max_index]).height;
238
239     // Mat crop = img(roi);
240
```

```
241     //printf("corners:%d",approxedcontours[max_index].size());
242 
243     //printf("contors:%d\n",approxedcontours.size());
244 
245     imshow("cam",img);
246     imshow("cam bw",imgSmooth);
247 
248     Size size = symb.size();
249 
250 
251     if(size.width != 0 && size.height != 0 ){
252         imshow("shift cam",symb);
253     }
254     return symb;
255 
256 }
257 
258 Mat transformPerspective(std::vector<Point> boundingContour, Mat frame, int
    x_size, int y_size)
259 {
260     if(boundingContour.size() != 4)
261     {
262         // Error the contour has too many points. Only 4 are allowed
263         Mat emptyMat;
264         return emptyMat;
265     }
266 
267     Mat symbol(y_size,x_size,CV_8UC1, Scalar(0));
268 
269     Point2f symbolCorners[4], boundingCorners[4];       // Create (and populate)
    variables containing the corner locations for the transform
270     symbolCorners[0] = Point2f(0,0);
271     symbolCorners[1] = Point2f(symbol.cols - 1,0);
272     symbolCorners[2] = Point2f(symbol.cols - 1,symbol.rows - 1);
273     symbolCorners[3] = Point2f(0,symbol.rows - 1);
274 
275     Point contourCentre = findContourCentre(boundingContour);    // To populate
    the contour corners we need to check the order of the points
276 
277     int point1, point2, point3, point4;
278 
279     if(boundingContour[0].x > contourCentre.x)
280     {
281         if(boundingContour[0].y > contourCentre.y)
282             point3 = 0;
283         else
284             point2 = 0;
285     }
286     else
287     {
288         if(boundingContour[0].y > contourCentre.y)
289             point4 = 0;
290         else
291             point1 = 0;
292     }
293 
294     if(boundingContour[1].x > contourCentre.x)
295     {
296         if(boundingContour[1].y > contourCentre.y)
297             point3 = 1;
298         else
```

```
299             point2 = 1;
300     }
301     else
302     {
303         if(boundingContour[1].y > contourCentre.y)
304             point4 = 1;
305         else
306             point1 = 1;
307     }
308
309     if(boundingContour[2].x > contourCentre.x)
310     {
311         if(boundingContour[2].y > contourCentre.y)
312             point3 = 2;
313         else
314             point2 = 2;
315     }
316     else
317     {
318         if(boundingContour[2].y > contourCentre.y)
319             point4 = 2;
320         else
321             point1 = 2;
322     }
323
324     if(boundingContour[3].x > contourCentre.x)
325     {
326         if(boundingContour[3].y > contourCentre.y)
327             point3 = 3;
328         else
329             point2 = 3;
330     }
331     else
332     {
333         if(boundingContour[3].y > contourCentre.y)
334             point4 = 3;
335         else
336             point1 = 3;
337     }
338
339     if(point1 + point2 + point3 + point4 != 6)
340     {
341         //Unable to reconstruct rectangle
342         Mat emptyMat;
343         return emptyMat;
344     }
345
346     boundingCorners[0] = boundingContour[point1];
347     boundingCorners[1] = boundingContour[point2];
348     boundingCorners[2] = boundingContour[point3];
349     boundingCorners[3] = boundingContour[point4];
350
351     Mat transformMatrix = cv::getPerspectiveTransform(boundingCorners,
    symbolCorners); // Calculate the required transform operation
352     Mat transformedSymbol(240,320,CV_8UC1,Scalar(0));
353     warpPerspective(frame, transformedSymbol, transformMatrix, cv::Size(symbol.
    cols,symbol.rows));  // Perform the transformation
354
355     return transformedSymbol;
356 }
357
```

```
358 Point findContourCentre(std::vector<cv::Point> contour)
359 {
360     Moments foundRegion;     // Variables to store the region moment and the
    centre point
361     Point centre;
362     foundRegion = moments(contour, false);       // Calculate the moment for the
    contour
363     centre.x = (foundRegion.m10/foundRegion.m00);  //Calculate the X and Y
    positions
364     centre.y = (foundRegion.m01/foundRegion.m00);
365
366     return centre;
367 }
368
369 float compareImages(Mat cameraImage, Mat librarySymbol)
370 {
371     float matchPercent = 100 - (100/((float)librarySymbol.cols*(float)
    librarySymbol.rows) * (2*(float)countNonZero(librarySymbol^cameraImage))); //
    An incorrect pixel has double weighting
372     return matchPercent;
373 }
374
375
376
377 void drawText(Mat & image, String text)
378 {
379     putText(image, text,
380             Point(20, 50),
381             FONT_HERSHEY_COMPLEX, 1, // font face and scale
382             Scalar(255, 255, 255), // white
383             1, LINE_AA); // line thickness and type
384 }
```

Listing 7.12: Symbol detect

7.8 Shape counter

```
1 #include "opencv2/core.hpp"
2 #include "opencv2/imgproc.hpp"
3 #include "opencv2/highgui.hpp"
4 #include "opencv2/videoio.hpp"
5
6 #include <sys/types.h>
7 #include <dirent.h>
8 #include <errno.h>
9 #include <vector>
10 #include <string>
11 #include <iostream>
12
13
14 using namespace cv;
15 using namespace std;
16
17 int count_shapes(Mat img);
18
19 Mat get_img(Mat img);
20
21
22 Mat transformPerspective(std::vector<Point> boundingContour, Mat frame, int
    x_size, int y_size);
23 Point findContourCentre(std::vector<cv::Point> contour);
```

```
24 float compareImages(Mat cameraImage, Mat librarySymbol);
25
26
27 void drawText(Mat & image, String text);
28
29
30 int main(int argc, char** argv )
31 {
32     if ( argc != 2 )
33     {
34         printf("usage: DisplayImage.out <Image_Path>\n");
35         return -1;
36     }
37     Mat image;
38     image = imread( argv[1], 1 );
39     if ( !image.data )
40     {
41         printf("No image data \n");
42         return -1;
43     }
44     namedWindow("Display Image", WINDOW_AUTOSIZE );
45
46     int shapes = count_shapes(image);
47     string print = "num: " + to_string(shapes);
48     drawText(image, print);
49     imshow("Display Image", image);
50     waitKey(0);
51     return 0;
52 }
53
54 int count_shapes(Mat img){
55
56     cv::Size img_sz = img.size();
57     int imageWidth = img_sz.width;
58     int imageHeight = img_sz.height;
59
60
61     Mat image = get_img(img);
62
63     img_sz = image.size();
64     imageWidth = img_sz.width;
65     imageHeight = img_sz.height;
66
67     //image = Mat::zeros(240, 320, CV_8UC3);
68
69     Mat img_shapes;       // Convert the frame to HSV and apply the limits
70     cvtColor(image, img_shapes, COLOR_BGR2HSV);
71     int low_hue = 150, high_hue = 180;
72     inRange(img_shapes, Scalar(low_hue, 70, 80), Scalar(high_hue, (230), (255) ),
       img_shapes);
73
74     imshow("shapes only" , img_shapes);
75
76     std::vector<std::vector<cv::Point>>contours;// Variable for list of contours
77     std::vector<Vec4i>hierarchy;// Variable for image topology data
78     cv::findContours(img_shapes,contours,hierarchy,RETR_EXTERNAL,
      CHAIN_APPROX_SIMPLE,Point(0,0));// Calculate the contours and store them
79
80     for(int i =0;i < contours.size();i++)// Loop through the contours
81     {
82         drawContours(image,contours,i,Scalar(0,0,255),2,LINE_8,noArray(),0,Point
```

```
    ());// Draw each in red
83      }
84
85      if(contours.size() < 1){
86          return 0;
87      }
88
89      std::vector<std::vector<cv::Point>>approxedcontours(contours.size());// Array
        for new contours
90
91      for(int i=0;i<contours.size();i++){
92          cv::approxPolyDP(contours[i],approxedcontours[i],10,true);// Approximate
        the contour
93      }
94
95      int num =0;
96
97      for(int i =0;i <contours.size();i++)// Loop through the contours
98      {
99          drawContours(image,approxedcontours,i,Scalar(0,255,0),2,LINE_8,noArray()
        ,0,Point());// Draw each in green
100         if(approxedcontours[i].size() >= 3){
101             num++;
102         }
103     }
104
105     //printf("num:%d\n",num);
106
107     imshow("shapes_contored" , image);
108
109     return num;
110
111 }
112
113 Mat get_img(Mat img){
114
115
116     cv::Size img_sz = img.size();
117     int imageWidth = img_sz.width;
118     int imageHeight = img_sz.height;
119
120
121     Mat imgHSV;         // Convert the frame to HSV and apply the limits
122     cvtColor(img, imgHSV, COLOR_BGR2HSV);
123     int low_hue = 100, high_hue = 130;
124     inRange(imgHSV, Scalar(low_hue, 70, 80), Scalar(high_hue, (230), (255) ),
        imgHSV);
125
126     imshow("blue cam",imgHSV);
127
128     Mat imgSmooth;
129
130     Mat kernel=cv::getStructuringElement(MORPH_ELLIPSE,Size(5,5));
131     cv::morphologyEx(imgHSV,imgSmooth,MORPH_CLOSE,kernel);
132
133     //imshow("smooth cam",imgSmooth);
134
135     std::vector<std::vector<cv::Point>>contours;// Variable for list of contours
136     std::vector<Vec4i>hierarchy;// Variable for image topology data
137     cv::findContours(imgSmooth,contours,hierarchy,RETR_EXTERNAL,
        CHAIN_APPROX_SIMPLE,Point(0,0));// Calculate the contours and store them
```

```
138
139     if (contours.size() < 1){
140         Mat symb = Mat::zeros(240, 320, CV_8UC3);
141         return symb;
142     }
143
144
145     for(int i =0;i < contours.size();i++)// Loop through the contours
146     {
147         drawContours(img,contours,i,Scalar(0,0,255),2,LINE_8,noArray(),0,Point())
    ;// Draw each in red
148     }
149
150     std::vector<std::vector<cv::Point>>approxedcontours(contours.size());// Array
     for new contours
151
152     for(int i=0;i<contours.size();i++){
153         cv::approxPolyDP(contours[i],approxedcontours[i],30,true);// Approximate
    the contour
154     }
155
156     for(int i =0;i <contours.size();i++)// Loop through the contours
157     {
158         drawContours(img,approxedcontours,i,Scalar(0,255,0),2,LINE_8,noArray(),0,
    Point());// Draw each in green
159     }
160
161     int max_area =0, max_index = 0,area =0;
162
163
164     for(int i =0;i <approxedcontours.size();i++)// Loop through the contours
165     {
166         area = contourArea(approxedcontours[i]);
167         if (area > max_area){
168             max_area = area;
169             max_index = i;
170         }
171     }
172
173     //int corners =  approxedcontours[max_index].size();
174
175     //printf("corners:%d",corners);
176
177     if (approxedcontours[max_index].size() != 4){
178         Mat symb = Mat::zeros(240, 320, CV_8UC3);
179         //imshow("board", symb);
180         return symb;
181     }
182
183     Mat symb = transformPerspective(approxedcontours[max_index],img,320,240);
184
185     //imshow("cam",img);
186     //imshow("board", symb);
187
188
189     return symb;
190
191 }
192
193 Mat transformPerspective(std::vector<Point> boundingContour, Mat frame, int
    x_size, int y_size)
```

```
{
    if(boundingContour.size() != 4)
    {
        // Error the contour has too many points. Only 4 are allowed
        Mat emptyMat;
        return emptyMat;
    }

    Mat symbol(y_size,x_size,CV_8UC1, Scalar(0));

    Point2f symbolCorners[4], boundingCorners[4];       // Create (and populate) variables containing the corner locations for the transform
    symbolCorners[0] = Point2f(0,0);
    symbolCorners[1] = Point2f(symbol.cols - 1,0);
    symbolCorners[2] = Point2f(symbol.cols - 1,symbol.rows - 1);
    symbolCorners[3] = Point2f(0,symbol.rows - 1);

    Point contourCentre = findContourCentre(boundingContour);   // To populate the contour corners we need to check the order of the points

    int point1, point2, point3, point4;

    if(boundingContour[0].x > contourCentre.x)
    {
        if(boundingContour[0].y > contourCentre.y)
            point3 = 0;
        else
            point2 = 0;
    }
    else
    {
        if(boundingContour[0].y > contourCentre.y)
            point4 = 0;
        else
            point1 = 0;
    }

    if(boundingContour[1].x > contourCentre.x)
    {
        if(boundingContour[1].y > contourCentre.y)
            point3 = 1;
        else
            point2 = 1;
    }
    else
    {
        if(boundingContour[1].y > contourCentre.y)
            point4 = 1;
        else
            point1 = 1;
    }

    if(boundingContour[2].x > contourCentre.x)
    {
        if(boundingContour[2].y > contourCentre.y)
            point3 = 2;
        else
            point2 = 2;
    }
    else
    {
```

```
253         if(boundingContour[2].y > contourCentre.y)
254             point4 = 2;
255         else
256             point1 = 2;
257     }
258 
259     if(boundingContour[3].x > contourCentre.x)
260     {
261         if(boundingContour[3].y > contourCentre.y)
262             point3 = 3;
263         else
264             point2 = 3;
265     }
266     else
267     {
268         if(boundingContour[3].y > contourCentre.y)
269             point4 = 3;
270         else
271             point1 = 3;
272     }
273 
274     if(point1 + point2 + point3 + point4 != 6)
275     {
276         //Unable to reconstruct rectangle
277         Mat emptyMat;
278         return emptyMat;
279     }
280 
281     boundingCorners[0] = boundingContour[point1];
282     boundingCorners[1] = boundingContour[point2];
283     boundingCorners[2] = boundingContour[point3];
284     boundingCorners[3] = boundingContour[point4];
285 
286     Mat transformMatrix = cv::getPerspectiveTransform(boundingCorners,
    symbolCorners); // Calculate the required transform operation
287     Mat transformedSymbol(240,320,CV_8UC1,Scalar(0));
288     warpPerspective(frame, transformedSymbol, transformMatrix, cv::Size(symbol.
    cols,symbol.rows));  // Perform the transformation
289 
290     return transformedSymbol;
291 }
292 
293 Point findContourCentre(std::vector<cv::Point> contour)
294 {
295     Moments foundRegion;     // Variables to store the region moment and the
    centre point
296     Point centre;
297     foundRegion = moments(contour, false);       // Calculate the moment for the
    contour
298     centre.x = (foundRegion.m10/foundRegion.m00);  //Calculate the X and Y
    positions
299     centre.y = (foundRegion.m01/foundRegion.m00);
300 
301     return centre;
302 }
303 
304 float compareImages(Mat cameraImage, Mat librarySymbol)
305 {
306     float matchPercent = 100 - (100/((float)librarySymbol.cols*(float)
    librarySymbol.rows) * (2*(float)countNonZero(librarySymbol^cameraImage))); //
    An incorrect pixel has double weighting
```

```
307     return matchPercent;
308 }
309
310
311
312 void drawText(Mat & image, String text)
313 {
314     putText(image, text,
315             Point(20, 50),
316             FONT_HERSHEY_COMPLEX, 1, // font face and scale
317             Scalar(255, 255, 255), // white
318             1, LINE_AA); // line thickness and type
319 }
```

Listing 7.13: Shape counter

7.9 Stop light detection

```
1  #include "opencv2/core.hpp"
2  #include "opencv2/imgproc.hpp"
3  #include "opencv2/highgui.hpp"
4  #include "opencv2/videoio.hpp"
5
6  #include <sys/types.h>
7  #include <dirent.h>
8  #include <errno.h>
9  #include <vector>
10 #include <string>
11 #include <iostream>
12
13
14 using namespace cv;
15 using namespace std;
16
17
18 Mat get_img(Mat img, int hue);
19
20 int numPixels(Mat img,int low_hue,int high_hue,int low_sat,int high_sat,int
       low_val,int high_val);
21
22
23
24 Mat transformPerspective(std::vector<Point> boundingContour, Mat frame, int
       x_size, int y_size);
25 Point findContourCentre(std::vector<cv::Point> contour);
26 float compareImages(Mat cameraImage, Mat librarySymbol);
27
28 void drawText(Mat & image, String text);
29
30 int main(int argc, char** argv )
31 {
32     if ( argc != 2 )
33     {
34         printf("usage: DisplayImage.out <Image_Path>\n");
35         return -1;
36     }
37     Mat image;
38     image = imread( argv[1], 1 );
39     if ( !image.data )
40     {
41         printf("No image data \n");
```

```
42          return -1;
43      }
44
45      Mat green_light =  get_img(image , 60);
46      Mat red_light = get_img(image, 170);
47
48      int green_bright = numPixels(green_light , 0, 180, 0, 255, 200, 255);
49      int red_bright = numPixels(red_light , 0, 180, 0, 255, 200, 255);
50
51      if (green_bright > red_bright)
52      {
53          drawText(image,"GREEN");
54      }else if (green_bright < red_bright)
55      {
56          drawText(image,"RED");
57      }else
58      {
59          drawText(image,"NEITHER");
60      }
61
62      imshow("Display Image", image);
63
64      imshow("green Image", green_light);
65      imshow("red Image", red_light);
66
67      waitKey(0);
68      return 0;
69  }
70
71
72  Mat get_img(Mat img, int hue )
73  {
74
75      cv::Size img_sz = img.size();
76      int imageWidth = img_sz.width;
77      int imageHeight = img_sz.height;
78
79      int upper_hue = (hue - 20), lower_hue = (hue + 20);
80
81      Mat imgHSV;         // Convert the frame to HSV and apply the limits
82      cvtColor(img, imgHSV, COLOR_BGR2HSV);
83      inRange(imgHSV, Scalar(upper_hue, 130, 90), Scalar(lower_hue, 255, 255),
        imgHSV);
84
85      //imshow("green cam",imgHSV);
86
87      Mat imgSmooth;
88
89      Mat kernel=cv::getStructuringElement(MORPH_ELLIPSE, Size(5,5));
90      cv::morphologyEx(imgHSV,imgSmooth,MORPH_CLOSE, kernel);
91
92      //imshow("smooth cam",imgSmooth);
93
94      std::vector<std::vector<cv::Point>>contours;// Variable for list of contours
95      std::vector<Vec4i>hierarchy;// Variable for image topology data
96      cv::findContours(imgSmooth,contours,hierarchy,RETR_EXTERNAL,
        CHAIN_APPROX_SIMPLE, Point(0,0));// Calculate the contours and store them
97
98      if (contours.size() < 1){
99          Mat symb = Mat::zeros(240, 320, CV_8UC3);
100         return symb;
```

```
101     }
102
103
104     for(int i =0;i < contours.size();i++)// Loop through the contours
105     {
106         drawContours(img,contours,i,Scalar(0,0,255),2,LINE_8,noArray(),0,Point())
    ;// Draw each in red
107     }
108
109     int max_area =0, max_index = 0,area =0;
110
111
112     for(int i =0;i <contours.size();i++)// Loop through the contours
113     {
114         area = contourArea(contours[i]);
115         if (area > max_area){
116             max_area = area;
117             max_index = i;
118         }
119     }
120
121     Rect boundRect;
122
123     boundRect = boundingRect( contours[max_index] );
124
125
126     Mat image_roi = img(boundRect);
127
128
129     return image_roi;
130
131 }
132
133 Mat transformPerspective(std::vector<Point> boundingContour, Mat frame, int
    x_size, int y_size)
134 {
135     if(boundingContour.size() != 4)
136     {
137         // Error the contour has too many points. Only 4 are allowed
138         Mat emptyMat;
139         return emptyMat;
140     }
141
142     Mat symbol(y_size,x_size,CV_8UC1, Scalar(0));
143
144     Point2f symbolCorners[4], boundingCorners[4];       // Create (and populate)
    variables containing the corner locations for the transform
145     symbolCorners[0] = Point2f(0,0);
146     symbolCorners[1] = Point2f(symbol.cols - 1,0);
147     symbolCorners[2] = Point2f(symbol.cols - 1,symbol.rows - 1);
148     symbolCorners[3] = Point2f(0,symbol.rows - 1);
149
150     Point contourCentre = findContourCentre(boundingContour);    // To populate
    the contour corners we need to check the order of the points
151
152     int point1, point2, point3, point4;
153
154     if(boundingContour[0].x > contourCentre.x)
155     {
156         if(boundingContour[0].y > contourCentre.y)
157             point3 = 0;
```

```
158         else
159             point2 = 0;
160     }
161     else
162     {
163         if(boundingContour[0].y > contourCentre.y)
164             point4 = 0;
165         else
166             point1 = 0;
167     }
168 
169     if(boundingContour[1].x > contourCentre.x)
170     {
171         if(boundingContour[1].y > contourCentre.y)
172             point3 = 1;
173         else
174             point2 = 1;
175     }
176     else
177     {
178         if(boundingContour[1].y > contourCentre.y)
179             point4 = 1;
180         else
181             point1 = 1;
182     }
183 
184     if(boundingContour[2].x > contourCentre.x)
185     {
186         if(boundingContour[2].y > contourCentre.y)
187             point3 = 2;
188         else
189             point2 = 2;
190     }
191     else
192     {
193         if(boundingContour[2].y > contourCentre.y)
194             point4 = 2;
195         else
196             point1 = 2;
197     }
198 
199     if(boundingContour[3].x > contourCentre.x)
200     {
201         if(boundingContour[3].y > contourCentre.y)
202             point3 = 3;
203         else
204             point2 = 3;
205     }
206     else
207     {
208         if(boundingContour[3].y > contourCentre.y)
209             point4 = 3;
210         else
211             point1 = 3;
212     }
213 
214     if(point1 + point2 + point3 + point4 != 6)
215     {
216         //Unable to reconstruct rectangle
217         Mat emptyMat;
218         return emptyMat;
```

```
219     }
220
221     boundingCorners[0] = boundingContour[point1];
222     boundingCorners[1] = boundingContour[point2];
223     boundingCorners[2] = boundingContour[point3];
224     boundingCorners[3] = boundingContour[point4];
225
226     Mat transformMatrix = cv::getPerspectiveTransform(boundingCorners,
    symbolCorners); // Calculate the required transform operation
227     Mat transformedSymbol(240,320,CV_8UC1,Scalar(0));
228     warpPerspective(frame, transformedSymbol, transformMatrix, cv::Size(symbol.
    cols,symbol.rows));  // Perform the transformation
229
230     return transformedSymbol;
231 }
232
233 Point findContourCentre(std::vector<cv::Point> contour)
234 {
235     Moments foundRegion;     // Variables to store the region moment and the
    centre point
236     Point centre;
237     foundRegion = moments(contour, false);       // Calculate the moment for the
    contour
238     centre.x = (foundRegion.m10/foundRegion.m00);  //Calculate the X and Y
    positions
239     centre.y = (foundRegion.m01/foundRegion.m00);
240
241     return centre;
242 }
243
244 float compareImages(Mat cameraImage, Mat librarySymbol)
245 {
246     float matchPercent = 100 - (100/((float)librarySymbol.cols*(float)
    librarySymbol.rows) * (2*(float)countNonZero(librarySymbol^cameraImage))); //
    An incorrect pixel has double weighting
247     return matchPercent;
248 }
249
250 int numPixels(Mat img,int low_hue,int high_hue,int low_sat,int high_sat,int
    low_val,int high_val)
251 {
252
253     Mat frameHSV;        // Convert the frame to HSV and apply the limits
254     cvtColor(img, frameHSV, COLOR_BGR2HSV);
255     inRange(frameHSV, Scalar(low_hue, low_sat, low_val), Scalar(high_hue,
    high_sat, high_val), frameHSV);
256
257     int pixels = 0;
258
259     pixels = countNonZero(frameHSV);
260
261     return pixels;
262
263 }
264
265 void drawText(Mat & image, String text)
266 {
267     putText(image, text,
268             Point(20, 50),
269             FONT_HERSHEY_COMPLEX, 1, // font face and scale
270             Scalar(0, 0, 0), // white
```

```
271             1, LINE_AA); // line thickness and type
272     putText(image, text,
273             Point(20, 100),
274             FONT_HERSHEY_COMPLEX, 1, // font face and scale
275             Scalar(255, 255, 255), // white
276             1, LINE_AA); // line thickness and type
277 }
```

Listing 7.14: Stop light detection

Bibliography

[1] R. pi foundation, "Raspberry pi 3 model b." https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed on 9/05/2020.

[2] the pi Hut, "Raspberry pi 3 model b product page." https://thepihut.com/products/raspberry-pi-3-model-b?src=raspberrypi. Accessed on 9/05/2020.

[3] Elinux, "Rpi bcm2835 gpios." https://elinux.org/RPi_BCM2835_GPIOs. Accessed on 27/04/2020 , Published on 6/04/2020.

[4] R. pi foundation, "Raspberry pi 3 model b+ (reduced schematic)." https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3bplus_1p0_reduced.pdf. Date created: 19/03/2018.

[5] C. Basics, "Build a great sounding audio amplifier from the lm386." https://www.circuitbasics.com/build-a-great-sounding-audio-amplifier-with-bass-boost-from-the-lm386/. Accessed on 27/04/2020.

[6] R. Components, "Rs components product page." https://uk.rs-online.com/web/p/miniature-speakers/7243100/. Accessed on 27/04/2020.

[7] Arduino, "I2c bi-directional level shifter." https://playground.arduino.cc/Main/I2CBi-directionalLevelShifter/. Accessed on 21/04/2020.

[8] R. pi foundation, "Camera module v2." https://www.raspberrypi.org/products/camera-module-v2/. Accessed on 9/05/2020.

[9] J. Sheppard, "Raspberry pi c++ library for easy i2c communication to and from an arduino." https://github.com/JohnnySheppard/Pi2c. Accessed on 9/05/2020 , Last edited on 3/11/2014.

[10] M. circuits, "H-bridges – the basics." https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/. Accessed on 09/05/2020.