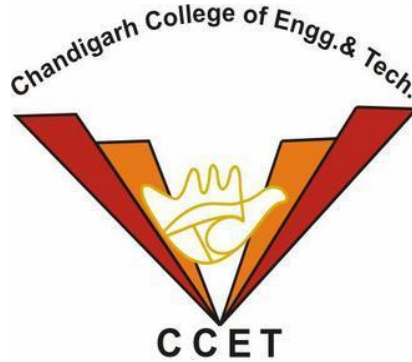


PROJECT REPORT

On Self Driving Car



**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE AWARD OF**

THE DEGREE OF

BACHELOR OF ENGINEERING

(Electronics & Communication Engineering)

SUBMITTED TO

**Chandigarh College of Engg and Technology, PANJAB UNIVERSITY,
CHANDIGARH**

SUBMITTED BY

**AMOL CHAUDHRY
ATHARV SHARMA
SAHIB BIR SINGH BHATIA**

**ROLL NO. :CO16503
ROLL NO. :CO16509
ROLL NO. :CO16547**

**SUPERVISED BY
Prof. Sarita Sharma
December 2019**

Acknowledgment

We would like to place on record our deep sense of gratitude to Prof. _____, HOD-Dept. of Electronics & Communication Engineering, CCET Chandigarh India for his generous guidance, help and useful suggestions.

We express our sincere gratitude to Prof. _____, Dept. of Electronics & Communication Engineering, CCET Chandigarh India, for her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

Signature of Students:

Amol Chaudhry

Atharv Sharma

Sahib Bir Singh Bhatia

ABSTRACT

Self-driving cars are a shared ambition among Google, Tesla, Apple, Uber and Lyft, among other automotive, tech and ridesharing companies. For Uber and Lyft specifically, it's a matter of cutting costs. However, fiscal expediency is not the main benefit of this emerging technology.

Roughly 94 percent of traffic accidents are caused by human error, and to many, autonomous vehicles (AVs) seem to be our only path toward lessening related fatalities.

In addition, driverless cars have other benefits, such as lower fuel consumption, lower CO2 emissions and a reduction in congestion.

Our objective for this project is to design and build an autonomous driving car (hereafter referred to as “self driving car”) that is capable of stable traversal through the self made track with automatic control using machine learning.

More specifically, our goal was to implement a Raspberry Pi microcontroller and an Arduino microcontroller in a master/slave configuration as the controllers and program them with code and also implement different techniques like machine learning and image processing.

INDEX

ACKNOWLEDGEMENT	ii
ABSTRACT	iii
LIST OF FIGURES	v
Chapter 1 Self Driving Car Essentials	6
1.1 Car Frame	
1.2 Motors	
1.3 Motor Drive Controller Board	
1.4 Battery	
1.5 Arduino Uno	
1.6 Raspberry Pi	
1.7 Raspberry Pi Camera Module	
1.8 Self Driving Car and Track Photos	
Chapter 2 Device Setups	12
2.1 Arduino Uno Setup	
2.2 RaspberryPi Setup	
2.3 OpenCv Software	
Chapter 3 Raspberry Pi Raspicam Setup	17
3.1 Camera Setup	
3.2 Capturing Images and Video	
3.3 Raspbian	
3.4 Image Processing	
Chapter 4 Device Communication	29
4.1 Master-Slave Communication	
4.2 Machine Learning	
References	38

LIST OF FIGURES

1.CHAPTER 1:

1. Figure 1.1 Frame
2. Figure 1.2 Frame Assembly
3. Figure 1.3 H Bridge
4. Figure 1.4 Battery
5. Figure 1.5 Arduino
6. Figure 1.6a Raspberry Pi , Figure 1.6b Raspberry Pi Pins
7. Figure 1.8a Side view, Figure1.8b Back view, Figure 1.8c Bird's Eye view

2.CHAPTER 2:

1. Figure 2.1a Forward Code, Figure 2.1b Backward Code
2. Figure 2.1c Right Code, Figure 2.1d Left Code

3.CHAPTER 3:

1. Figure 3.1a Camera Module
2. Figure 3.1b Camera installed
3. Figure 3.1c Select "Enable" and press "Enter".
4. Figure 3.1d Select "Yes" and press "Enter".Your Pi will reboot.
5. Figure 3.1e Reboot y/n
6. Figure 3.2a Camera Perspective, Figure3.2b Lane Detection
7. Figure 3.3 Raspbian OS
8. Figure 3.4 Representation of an image as a matrix

4.CHAPTER 4:

1. Figure 4.1a,b,c,d Arduino Code
2. Figure 4.2.1 Reinforcement learning
3. Figure 4.2.2 K-means Algorithm-The cluster centroids are depicted as crosses and training examples are depicted as dots. (a) Original dataset. (b) Random initial cluster centroids. (c-f) The demonstration of running 2 iterations of k-means. Each training example is assigned in each iteration to the cluster centroid that is closest and then, each cluster centroid is moved to mean of points assigned to it.
4. Figure 4.2.3 SVM

CHAPTER: 1

SELF DRIVING CAR HARDWARE ESSENTIALS

Every self driving car prototype requires, at a minimum, the following components: a frame, motors, motor speed controller, a battery, a camera, and microcontrollers.

This section will discuss the function of each component, why it is necessary, and what considerations to make when selecting that component.

1.1 Car Frame

The frame of the car provides the physical structure for the entire vehicle. It joins the motors to the rest of the car and houses all of the other components. The frame must be large enough, but must not be too large and therefore too heavy for the motors.

For our self driving car we chose a Frame Kit as seen in Figures which measures at width:6.1inches length:10.1inches across motors.

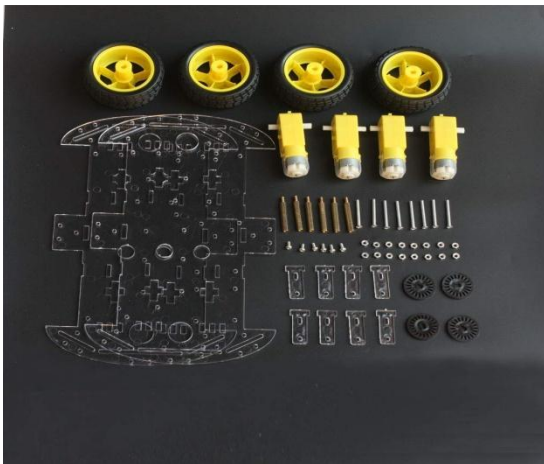


Figure 1.1 Frame

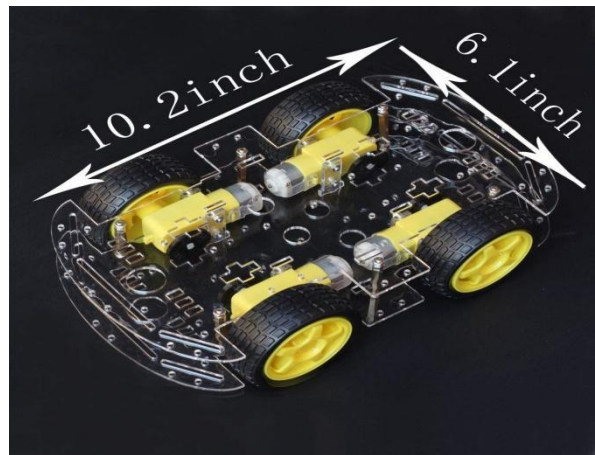


Figure 1.2 Frame Assembly

1.2 Motors

The motors spin the wheels to provide the car with forward thrust.

Self driving cars almost exclusively use brushless DC motors, as they provide thrust-to-weight ratios superior to brushed DC motors. However, they require more complex speed controllers.

Motors are typically given two ratings: Kv ratings and current ratings. The Kv rating indicates how fast the motor will spin (RPM) for 1 V of applied voltage.

The current rating indicates the max current that the motor may safely draw. For our project, we selected normal DC motors.

1.3 L298N Motor Drive Controller Board Module Dual H Bridge DC STEPPER

Dual-channel H-bridge driver working mode creates higher working efficiency, L298N as main chip. Can drive one 2-phase stepper motor, one 4-phase stepper motor or two DC motors. The L298N is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic level and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional Supply input is provided so that the logic works at a lower voltage.

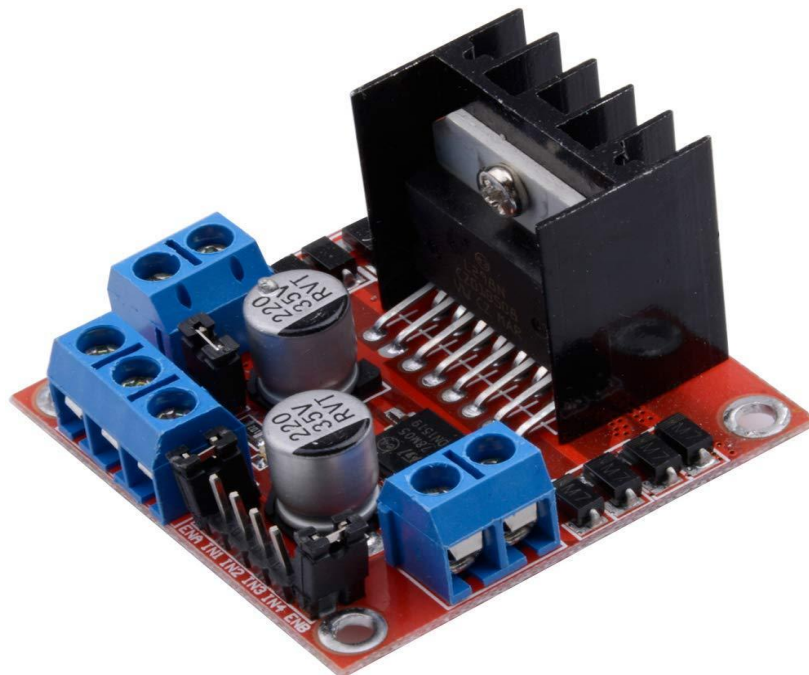


Figure 1.3 H Bridge

1.4 Battery



Figure 1.4 Battery



Figure 1.5 Arduino Uno

1.5 Arduino

The controller is the “brain” of the vehicle, and performs the necessary operations to keep the car stable and controllable.

It accepts user control commands from the RaspberryPi, and calculates the necessary motor output.

For our project however, we used an Arduino Uno as the vehicle controller, as we intended to program the car control software ourselves.

1.6 RaspberryPi

The Raspberry Pi 3 Model B is the earliest model of the third-generation Raspberry Pi. It replaced the Raspberry Pi 2 Model B in February 2016.

The Raspberry Pi 3 Model B+, the latest product in the Raspberry Pi 3 range has features:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI

- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

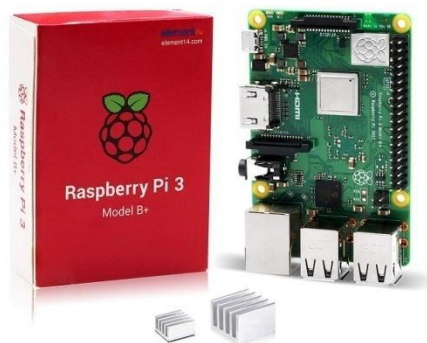


Figure 1.6a Raspberry Pi

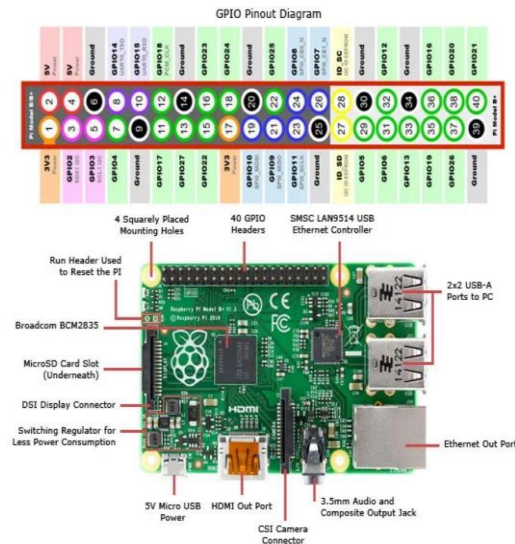


Figure 1.6b Raspberry Pi Pins

1.7 Raspberry Pi Camera Module V2-8 Megapixel,1080

The Camera Module can be used to take high-definition video, as well as still photographs.

It's easy to use for beginners, but has plenty to offer advanced users if you're looking to expand your knowledge. It supports 1080p30, 720p60 and VGA90 video modes, as well as still capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi.

The camera works with all models of Raspberry Pi 1, 2, 3 and 4. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library.

1.8 SELF DRIVING CAR AND TRACK PHOTOS

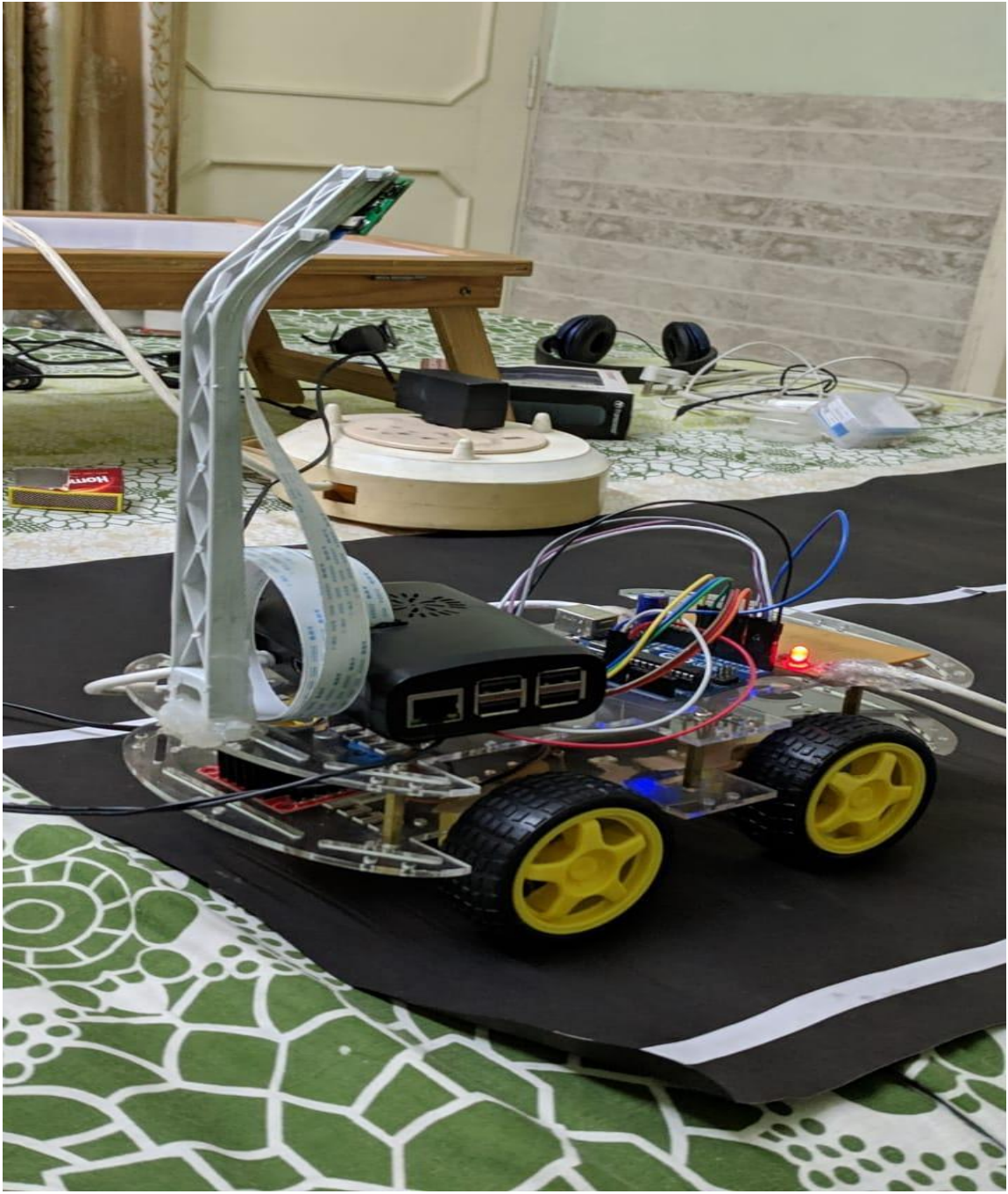


Figure 1.8a sideview

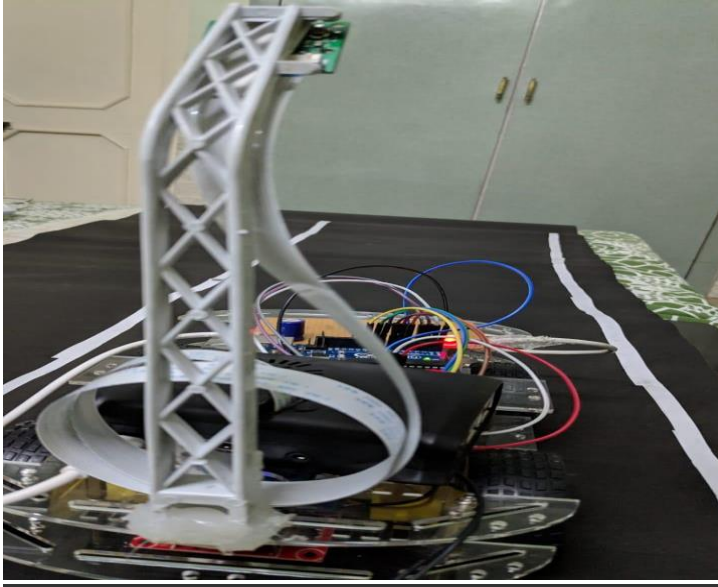


Figure1.8b Backview



Figure 1.8c Bird's Eye View

CHAPTER:2

DEVICE SETUPS

2.1 SLAVE DEVICE SET UP(ARDUINO UNO)

- In some situations, it can be helpful to set up two (or more!) Arduino and Genuino boards to share information with each other.
- The I2C protocol involves using two lines to send and receive data: a serial clock pin (SCL) that the Arduino or Genuino Master board pulses at a regular interval, and a serial data pin (SDA) over which data is sent between the two devices. As the clock line changes from low to high (known as the rising edge of the clock pulse), a single bit of information - that will form in sequence the address of a specific device and a command or data - is transferred from the board to the I2C device over the SDA line. When this information is sent - bit after bit -, the called upon device executes the request and transmits its data back - if required - to the board over the same line using the clock signal still generated by the Master on SCL as timing.
- Because the I2C protocol allows for each enabled device to have its own unique address, and as both master and slave devices take turns communicating over a single line, it is possible for your Arduino or Genuino board to communicate (in turn) with many devices, or other boards, while using just two pins of your microcontroller.
- There are four functions that we have designed for the arduinouno which will work as a slave device in our project, these are:

1.FORWARD FUNCTION:

```
Forward_Backward | Arduino 1.8.10
File Edit Sketch Tools Help

Upload

Forward_Backward

const int LowL =7;

const int EnableL = 10;
const int HighR = 8;    //RIGHT SIDE MOTOR
const int LowR =9;

void setup() {

  pinMode(EnableL, OUTPUT);
  pinMode(HighL, OUTPUT);
  pinMode(LowL, OUTPUT);

  pinMode(EnableR, OUTPUT);
  pinMode(HighR, OUTPUT);
  pinMode(LowR, OUTPUT);

}

void Forward()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,255);

}
```

Figure 2.1a Forward Code

2.BACKWARD FUNCTION:

```
void Backward()
{
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  analogWrite(EnableL,255);

  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnableR,255);

}
```

Figure 2.1b Backward Code

3. RIGHT FUNCTION:


```
Left_Right | Arduino 1.8.10
File Edit Sketch Tools Help

Left_Right$
void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 200);
}

void Right2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 160);
}

void Right3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 100);
}
```

Figure 2.1c Right Code

4.LEFT FUNCTION:

```
Left_Right
void Left1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 200);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 255);
}

void Left2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 160);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 255);
}

void Left3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 100);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 255);
}
```

Figure 2.1d Left Code

2.2 MASTER DEVICE SET UP(RASPBERRY PI)

- Installing Raspbian on the Raspberry Pi is pretty straightforward. We'll be downloading Raspbian and writing the disc image to a microSD card, then booting the Raspberry Pi to that microSD card. For this project, you'll need a microSD card (go with at least 16 GB), a computer with a slot for it, and, of course, a Raspberry Pi and basic peripherals (mouse, keyboard, screen, and power source). This isn't the only method for installing Raspbian (more on that in a moment), but it's a useful technique to learn because it can also be used to install so many other operating systems on the Raspberry Pi.
- We can connect the Raspberry Pi to the computer by the following three methods:
 1. Connecting Raspberry Pi to Personal Computer via Ethernet
 2. Connecting Raspberry Pi to Personal Computer via WIFI
 3. Raspberry Pi to Personal Computer via VNC Viewer
- For this project we are using VNC viewer/Remote Desktop to connect our Raspberry Pi to PC.
 In computing, **Virtual Network Computing (VNC)** is a graphical desktop-sharing system that uses the Remote FrameBuffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical-screen updates back in the other direction, over a network.
- VNC is platform-independent – there are clients and servers for many GUI-based operating systems and for Java. Multiple clients may connect to a VNC server at the same time. Popular uses for this technology include remote technical support and accessing files on one's work computer from one's home computer, or vice versa.
- VNC was originally developed at the Olivetti & Oracle Research Lab in Cambridge, United Kingdom. The original VNC source code and many modern derivatives are open source under the GNU General Public License.

- There are a number of variants of VNC which offer their own particular functionality; e.g., some optimised for Microsoft Windows, or offering file transfer (not part of VNC proper), etc. Many are compatible (without their added features) with VNC proper in the sense that a viewer of one flavour can connect with a server of another; others are based on VNC code but not compatible with standard VNC.

2.3 OPENCV SIMULATION SOFTWARE

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images.

What it can do :

1. Read and Write Images.
2. Detection of faces and its features.
3. Detection of shapes like Circle,rectangleetc in a image. E.g Detection of coin in images.
4. Text recognition in images. e.g Reading Number Plates/
5. Modifying image quality and colors e.g Instagram, CamScanner.
6. Developing Augmented reality apps.

Which Language it supports :

1. C++
2. Android SDK
3. Java
4. Python

CHAPTER:3

RASPBERRY PI RASPICAM SETUP

3.1 CAMERA SETUP

The Raspberry Pi camera module can be used to take high-definition video, as well as still photographs. It's easy to use for beginners, but has plenty to offer advanced users if you're looking to expand your knowledge. There are lots of examples online of people using it for time-lapse, slow-motion and other video cleverness. You can also use the libraries we bundle with the camera to create effects.

If you're interested in the nitty-gritty, you'll want to know that the module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi. It can be accessed through the MMAL and V4L APIs, and there are numerous third-party libraries built for it, including the Picamera Python library.

The camera module is very popular in home security applications, and in wildlife camera traps.

You can also use it to take snapshots.

Features

- 5MP sensor
- Wider image, capable of 2592x1944 stills, 1080p30 video
- 1080p video supported
- CSI
- Size: 25 x 20 x 9 mm

Camera Details

The camera consists of a small (25mm by 20mm by 9mm) circuit board, which connects to the Raspberry Pi's Camera Serial Interface (CSI) bus connector via a flexible ribbon cable. The camera's image sensor has a native resolution of five megapixels and has a fixed focus lens. The software for the camera supports full resolution still images up to 2592x1944 and video resolutions of 1080p30, 720p60 and 640x480p60/90. The camera module is shown below:

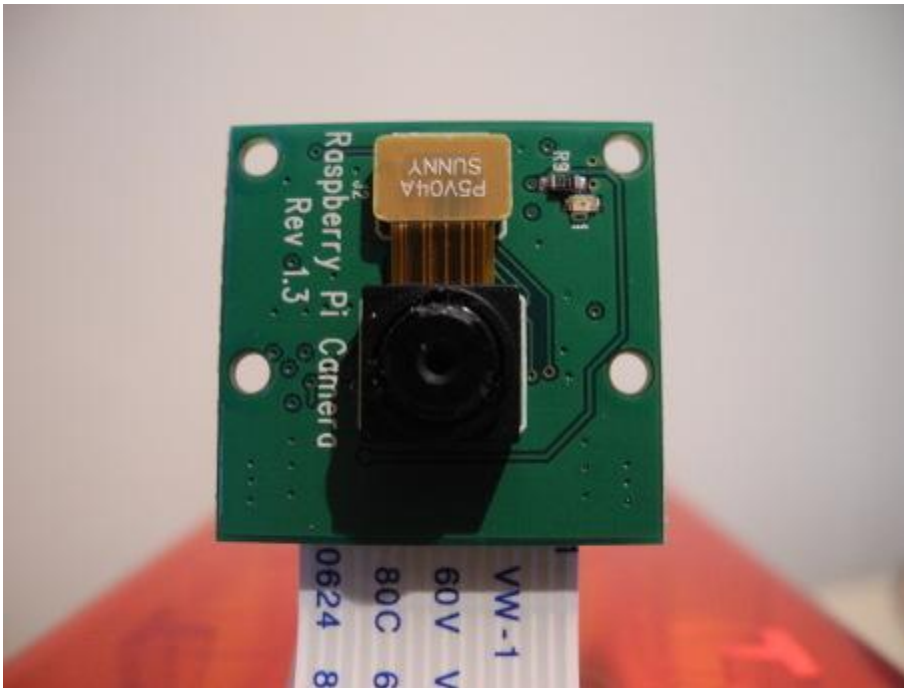


Figure 3.1a Camera Module

Installation involves connecting the ribbon cable to the CSI connector on the Raspberry Pi board. This can be a little tricky, but if you watch the videos that demonstrate how it is done, you shouldn't have any trouble.

When you purchase the camera, you will receive a small camera board and cable. You'll want to devise some method of supporting the camera in order to use it.

Some camera stands and Raspberry Pi cases are now available.

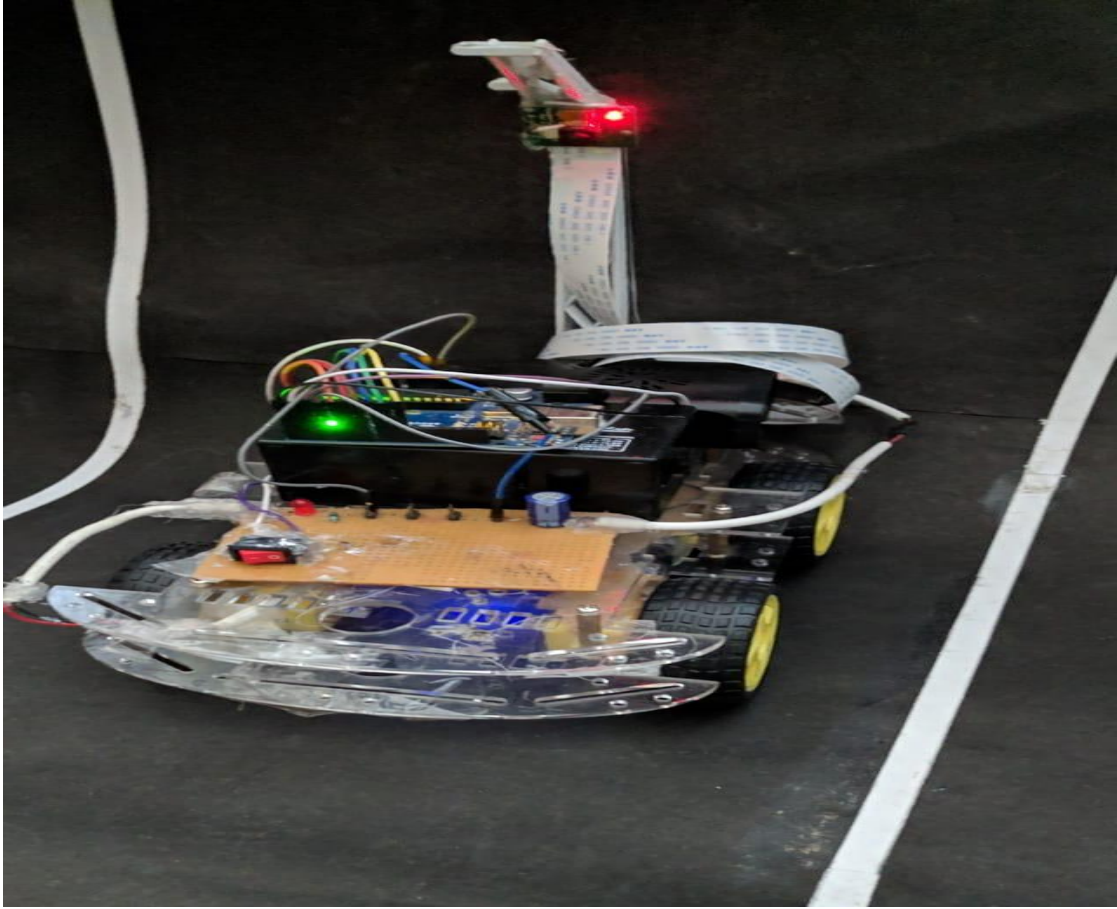


Figure 3.1b Camera Installed

Once the hardware is set up, you can move on to configuring the software.

Connect to the camera

- The flex cable inserts into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port. The flex cable connector should be opened by pulling the tabs on the top of the connector upwards then towards the Ethernet port. The flex cable should be inserted firmly into the connector, with care taken not to bend the flex at too acute an angle. The top part of the connector should then be pushed towards the HDMI connector and down, while the flex cable is held in place.

- Update the SD card

In order to use the camera you must be using a recent operating system that knows that the camera exists. The easiest way to do this is to grab the latest Raspbian image from the RaspberryPi.org site and create a fresh SD card.

- Enable camera in raspi-config settings

Reboot. If you are using a fresh image the raspi-config utility should load. If it doesn't then you can run it manually using `:sudo raspi-config`. Select the "Camera" option and press "Enter".



Figure3.1c Select "Enable" and press "Enter".

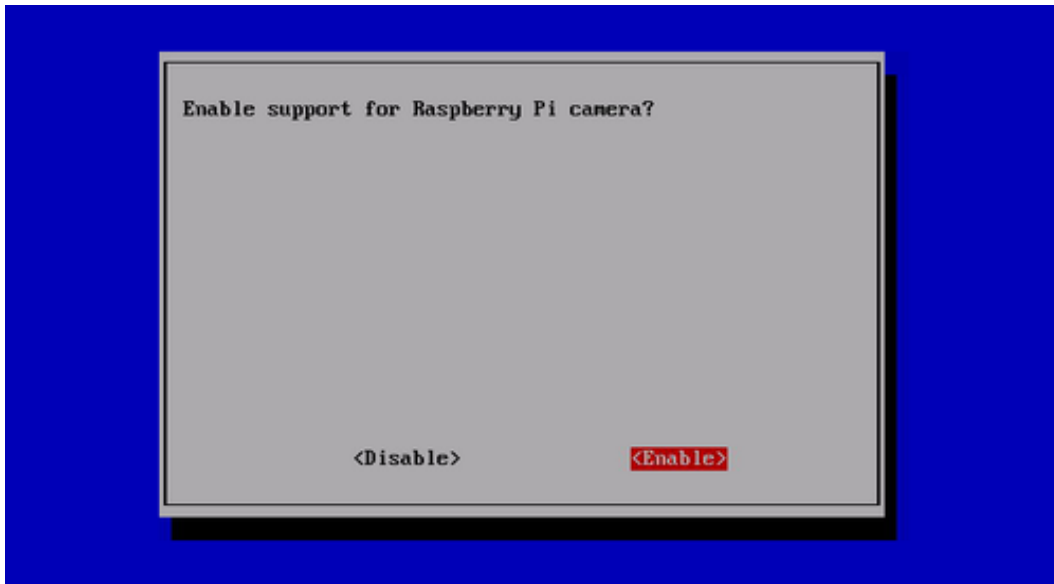


Figure3.1d Select “Yes” and press “Enter”. Your Pi will reboot.



Figure3.1d Reboot y/n

Updating your operating and enabling the camera using raspi-config did two things. It told your Pi that there is a camera attached and it added two command line utilities. Raspistill raspivid These allow you to capture still photos and HD video respectively.

Software

Since its inception, the camera is supported in the latest version of Raspbian, the preferred operating system for Raspberry Pi. The instructions in this blog post assume you are running Raspbian. The first step is to get the latest Raspberry Pi firmware, which supports the camera. You can do that from a console by running:

- `sudo apt-get update`
- `sudo apt-get upgrade`

You then need to enable the camera from the Raspberry Pi configuration program by running:

- `sudo raspi-config`

Choose "camera" from the program and then select "Enable support for Raspberry Pi camera". You should then reboot when prompted by the raspi-config program. The camera will be enabled on subsequent boots of the Raspberry Pi.

Several applications should now be available for the camera: the `rapistill` program captures images, `raspivid` captures videos, and `raspivid` takes uncompressed YUV format images. These are command line programs. They accept a number of options, which are documented if you run the commands without options. That reference also describes some more sophisticated things you can do, like streaming the video over the network and viewing it on another computer.

If you want to examine the source code for the programs, report bugs or compile them yourself, they are maintained at the project on github. You can either cross-compile or build the tools natively on the Raspberry Pi.

- **User Space V4L2 Driver**

The camera drivers are proprietary in the sense that they do not follow any standard APIs. That means that applications have to be written specifically for the Raspberry Pi camera. Under Linux, the standard API for cameras (including web cams) is V4L (Video for Linux), and a number of applications have been written that support any camera with a V4L driver. An independent developer has now written a user space V4L driver for the Raspberry Pi camera. With that driver, you can use generic Linux applications written for cameras. The driver has a few limitations: it is closed sourced, and can be a little slow because it runs as a user program rather than a kernel driver. The program worked reasonably well when I tested it and it is expected to continue to improve.

- **Official V4L2 Driver**

Recognizing that a V4L driver is needed, the Raspberry Pi Foundation reported that they were working with Broadcom to develop an official kernel V4L driver. As a kernel driver, it should be faster than the user space driver. The official driver became available in December 2013. The driver

is still quite new and not many people appear to have tried it yet. The latest Raspbian distribution and latest Raspberry Pi boot firmware is required for use. You will also need to build some code yourself. If you want to try it, some brief instructions showing the commands to build it are listed below (these commands should be run from a shell).

- **Get the latest Raspbian packages**

- `sudo apt-get update`
- `sudo apt-get upgrade`

- **Get the latest firmware**

- `sudo pi-update`

- **Get the source code for the V4L utilities**

- `git clone git`
- `CD v4l-utils`

Building the software should take about fifteen minutes. You need to have the camera enabled and sufficient Graphics Processing Unit (GPU) memory configured.

In theory, any camera application written to use the V4L APIs should work with the driver. I encourage you to try it out and report all positive or negative outcomes back to the developers.

The following are some basic commands:

1. **raspistill**: Capturing still photographs with the camera module
2. **raspivid**: Capturing video with the camera module
3. **Time-lapse**: Taking pictures at regular intervals and stitching them together into a video
4. **raspiyuv**: Capturing still photographs and generating raw unprocessed image files

3.2 CAPTURING IMAGES AND VIDEO

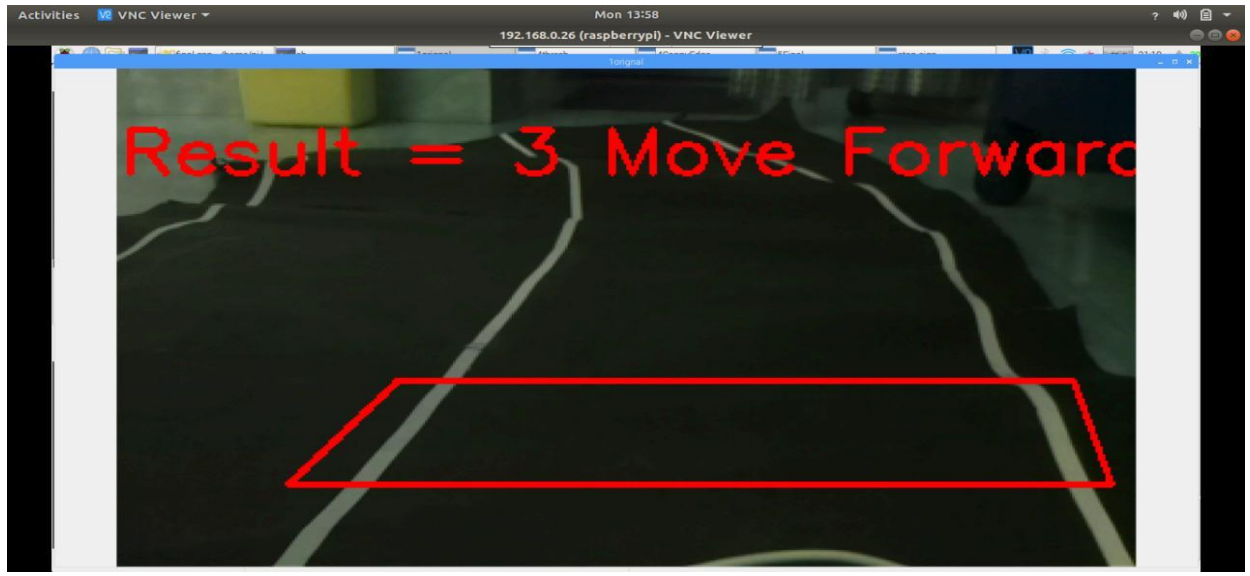


Figure 3.2a Camera Perspective

It is the process of locating a moving object (or multiple objects) over time using a camera. It has a variety of uses, some of which are: human-computer interaction, security and surveillance, video communication and compression, augmented reality, traffic control, medical imaging^[1] and video editing. Video tracking can be a time consuming process due to the amount of data that is contained in video. Adding further to the complexity is the possible need to use object recognition techniques for tracking, a challenging problem in its own right.

To perform video tracking an algorithm analyzes sequential video frames and outputs the movement of targets between the frames. There are a variety of algorithms, each having strengths and weaknesses. Considering the intended use is important when choosing which algorithm to use. There are two major components of a visual tracking system: target representation and localization, as well as filtering and data association.

Target representation and localization is mostly a bottom-up process. These methods give a variety of tools for identifying the moving object. Locating and tracking the target object successfully is dependent on the algorithm. For example, using blob tracking is useful for identifying human movement because a person's profile changes dynamically. Typically the computational complexity for these algorithms is low. The following are some common target representation and localization algorithms:

- **Kernel-based tracking** (mean-shift tracking): an iterative localization procedure based on the maximization of a similarity measure (Bhattacharyya coefficient).

- **Contour tracking:** detection of object boundary (e.g. active contours or Condensation algorithm). Contour tracking methods iteratively evolve an initial contour initialized from the previous frame to its new position in the current frame. This approach to contour tracking directly evolves the contour by minimizing the contour energy using gradient descent.

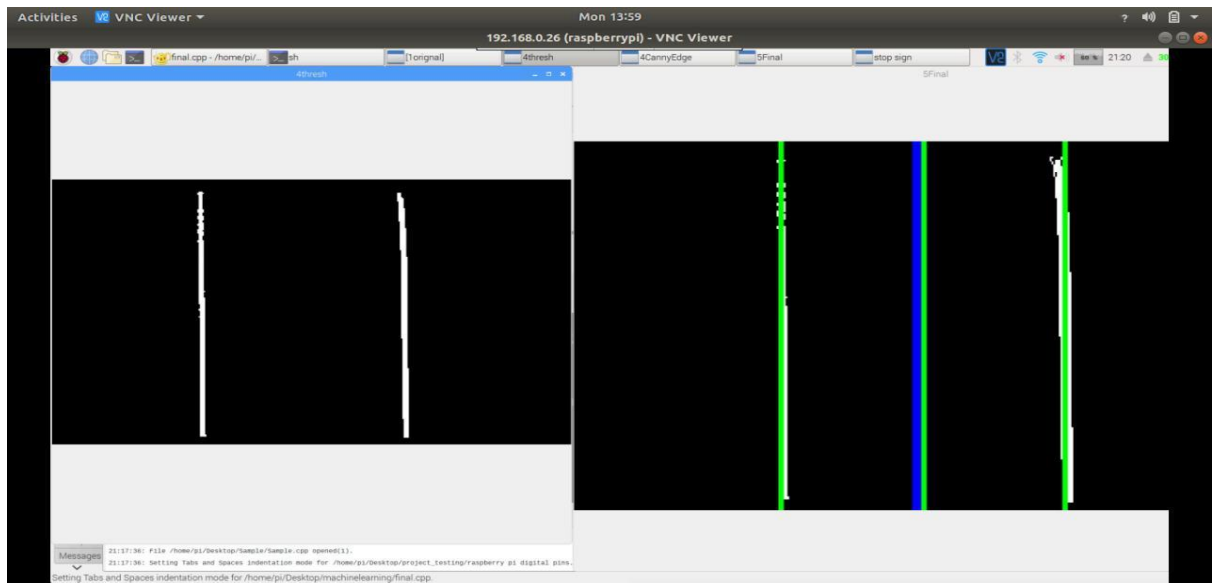


Figure3.2b Lane Detection

3.3 Raspbian

Raspbian is a Debian-based computer operating system for Raspberry Pi. There are several versions of Raspbian including Raspbian Buster and Raspbian Stretch. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers. Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012. The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs.

Raspbian uses PIXEL, Pi Improved X-Window Environment, Lightweight as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Openbox stacking window manager with a new theme and few other changes. The distribution is shipped with a copy of computer algebra program Mathematica and a version of Minecraft called Minecraft Pi as well as a lightweight version of Chromium as of the latest version.

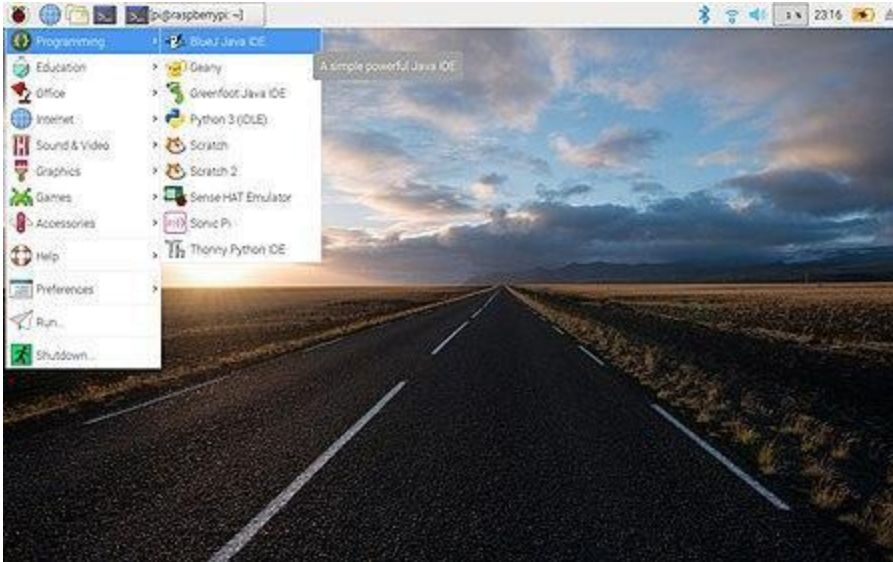


Figure3.3 Raspbian OS

3.4 IMAGE PROCESSING

Image processing is a method to perform some operations on an image, in order to extract some useful information from it. An image is nothing more than a two dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function $f(x,y)$ where x and y are the two coordinates horizontally and vertically. The value of $f(x,y)$ at any point gives the pixel value at that point of an image, the pixel value describes how bright that pixel is, and/or what color it should be.

For grayscale images the pixel value is a single number that represents the brightness of that pixel, the most common pixel format is the byte image, which is stored as an 8-bit integer giving a range of possible values from 0 to 255. As a convention is taken to be black, and 255 is taken to be white the values in between make up the different shades of gray.

To represent color images, separate red, green and blue components must be specified for each pixel (assuming an RGB color model), and so the pixel 'value' becomes a vector of three numbers. Often the three different components are stored as three separate 'grayscale' images known as color planes (one for each of red, green and blue).

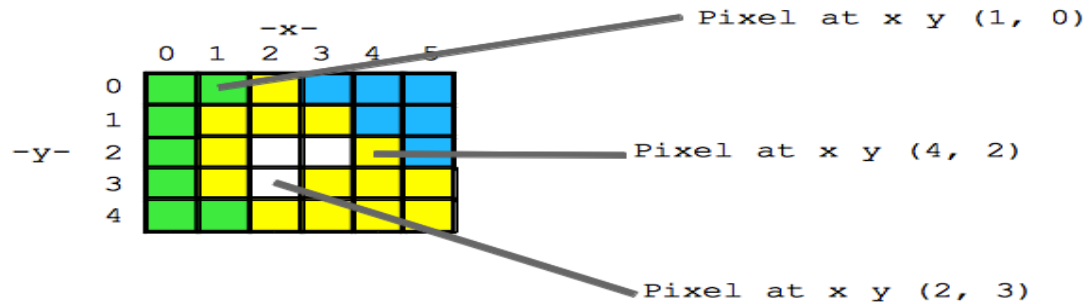


Figure 3.4 Representation of an image as a matrix

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components. When this model is associated with a precise description of how the components are to be interpreted (viewing conditions, etc.), the resulting set of colors is called color space.

Once known how the images could be represented, let's focus on the image processing side and specifically with OpenCV and python.

Sub-tasks in image processing could be categorized as follows :

- **Image acquisition, storage, transmission:** digitization/quantization, compression, encoding/decoding.
- **Image Enhancement and Restoration:** for improvement of pictorial information.
- **Information Extraction:** for further computer analysis .

Image Acquisition

OpenCV gives the flexibility to capture image directly from a pre-recorded video stream, camera input feed, or a directory path.

- **Taking input from a directory path**
- `img = cv2.imread('C:\Users\USER\Desktop\image.jpg',0)`
- **Capturing input from a video stream**
- `cap = cv2.VideoCapture(0)`

CHAPTER:4

DEVICE COMMUNICATION

4.1 MASTER-SLAVE DEVICE COMMUNICATION:

```
test
const int EnableL = 5;
const int HighL = 6;    // LEFT SIDE MOTOR
const int LowL = 7;

const int EnableR = 10;
const int HighR = 8;    //RIGHT SIDE MOTOR
const int LowR = 9;

const int D0 = 0;      //Raspberry pin 21   LSB
const int D1 = 1;      //Raspberry pin 22
const int D2 = 2;      //Raspberry pin 23
const int D3 = 3;      //Raspberry pin 24   MSB

int a,b,c,d,data;
```

Figure 4.1a,b,c Arduino Code

```
void Data()
{
    a = digitalRead(D0);
    b = digitalRead(D1);
    c = digitalRead(D2);
    d = digitalRead(D3);

    data = 8*d+4*c+2*b+a;
}
```

```
test
void loop()
{
    Data();
    if(data==0)
    {
        Forward();
    }

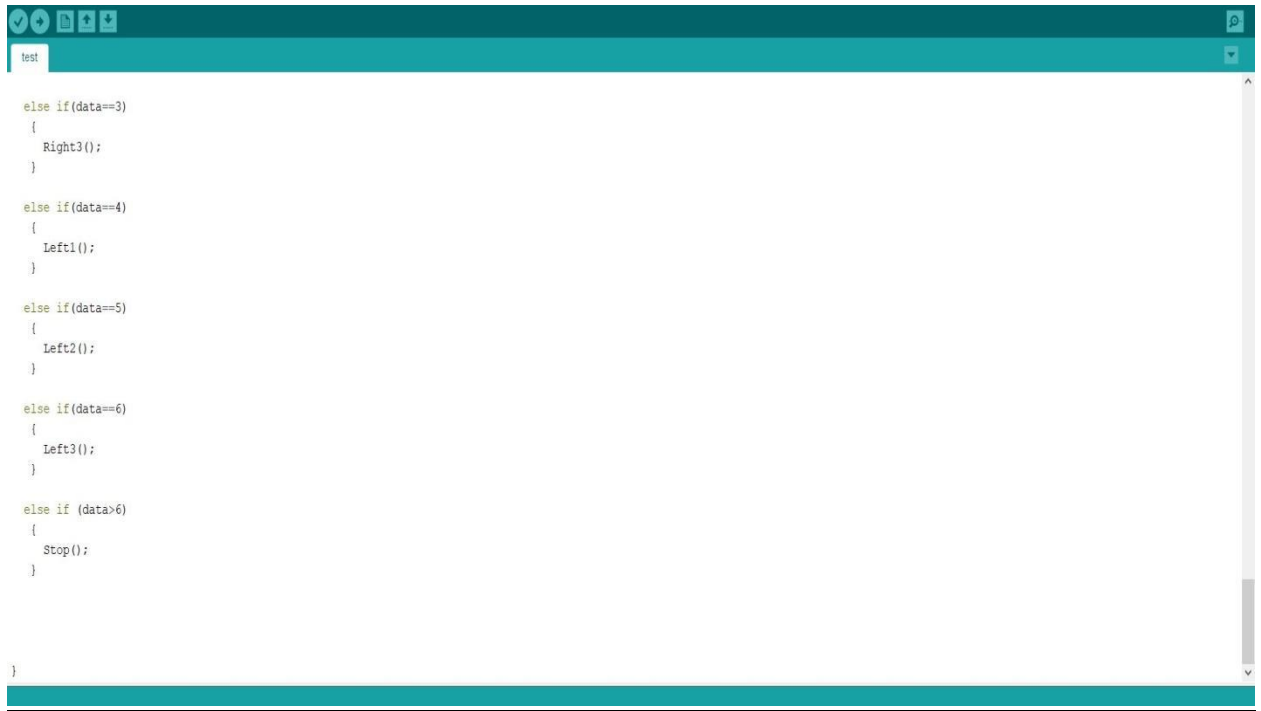
    else if(data==1)
    {
        Right1();
    }

    else if(data==2)
    {
        Right2();
    }

    else if(data==3)
    {
        Right3();
    }

    else if(data==4)
    {
        Left1();
    }

    else if(data==5)
    {
```



```
test

else if(data==3)
{
  Right3();
}

else if(data==4)
{
  Left1();
}

else if(data==5)
{
  Left2();
}

else if(data==6)
{
  Left3();
}

else if (data>6)
{
  Stop();
}

}
```

Figure 4.1d Arduino Code

4.2 Machine Learning

Basic terminologies:

Definition 1. A true positive test result is one that detects the condition when the condition is present.

Definition 2. A true negative test result is one that does not detect the condition when the condition is absent.

Definition 3. A false positive test result is one that detects the condition when the condition is absent.

Definition 4. A false negative test result is one that does not detect the condition when the condition is present.

Definition 5. Sensitivity measures the ability of a test to detect the condition when the condition is present. Thus, $\text{Sensitivity} = \text{TP}/(\text{TP}+\text{FN})$.

Definition 6. Specificity measures the ability of a test to correctly exclude the condition (not detect the condition) when the condition is absent. Thus, $\text{Specificity} = \text{TN}/(\text{TN}+\text{FP})$.

Definition 7. Predictive value positive is the proportion of positives that correspond to the presence of the condition. Thus, $\text{Predictive value positive} = \text{TP}/(\text{TP}+\text{FP})$.

Definition 8. Predictive value negative is the proportion of negatives that correspond to the absence of the condition. Thus, $\text{Predictive value negative} = \text{TN}/(\text{TN}+\text{FN})$.

Cascade Trainer GUI is a program that can be used to train, test and improve cascade classifier models. It uses a graphical interface to set the parameters and make it easy to use OpenCV tools for training and testing classifiers.

To test your classifiers, go to Test tab from the tab bar at the top and set the options as described below and finally press Start.

Select your cascade classifier using the Browse button at the top. You can also manually enter the path to cascade XML file in the Cascade Classifier XML field. This cascade classifier will be used for detection in images and/or videos.

You can select one of the following for Input Settings and you need to set the path according to this option:

- **Single Image:** A single image will be used as the scene in which detection will be done. In this case Path should point to a single image file.(Only supported images can be selected.)
- **Images in a Folder:** A folder containing many images will be used for testing. In this case Path should be set to a folder that contains one or more scene images.
- **Video:** A video file will be used for testing the classifier. In this case Path should point to a video file that will be used as input.

After setting the input settings it is time for output settings. Output type and path should be set according to what was chosen in input settings. See below:

- **Images in a Folder:** Detected objects will be saved to multiple images inside a folder. In this case Path should be set to a folder. Note that this option can be used with all of the possible options in input.
- **Video:** A video file with the detected objects highlighted with a red rectangle will be created. In this case Path should point to a video file that will be created after the test completes. Note that this option will only work if Video is selected in the input settings.

RANDOM FOREST CLASSIFIER

Random Forest Classifier is ensemble algorithm. In next one or two posts we shall explore such algorithms. Ensembled algorithms are those which combines more than one algorithms of same or different kind for classifying objects. For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object.

Suppose training set is given as : [X1, X2, X3, X4] with corresponding labels as [L1, L2, L3, L4], random forest may create three decision trees taking input of subset for example,

1. [X1, X2, X3]
2. [X1, X2, X4]
3. [X2, X3, X4]

So finally, it predicts based on the majority from each of the decision trees made.

This works well because a single decision tree may be prone to noise, but an aggregate of many decision trees reduce the effect of noise giving more accurate results.

The subsets in different decision trees created may overlap.

Basic parameters to Random Forest Classifier can be total number of trees to be generated and decision tree related parameters like minimum split, split criteria etc.

As part of their construction, random forest predictors naturally lead to a dissimilarity measure among the observations. One can also define a random forest dissimilarity measure between unlabeled data: the idea is to construct a random forest predictor that distinguishes the “observed” data from suitably generated synthetic data.

The observed data are the original unlabeled data and the synthetic data are drawn from a reference distribution. A random forest dissimilarity can be attractive because it handles mixed variable types very well, is invariant to monotonic transformations of the input variables, and is robust to outlying observations. The random forest dissimilarity easily deals with a large number of semi-continuous variables due to its intrinsic variable selection;

for example, the "Addcl 1" random forest dissimilarity weighs the contribution of each variable according to how dependent it is on other variables. The random forest dissimilarity has been used in a variety of applications, e.g. to find clusters of patients based on tissue marker data.

In the autonomous car, one of the major tasks of a machine learning algorithm is continuous rendering of surrounding environment and forecasting the changes that are possible to these surroundings. These tasks are classified into 4 sub-tasks:

- 1.The detection of an Object
- 2.The Identification of an Object or recognition object classification
- 3.The Object Localization and Prediction of Movement

The machine learning algorithms are loosely divided into 4 classes: decision matrix algorithms, cluster algorithms, pattern recognition algorithms and regression algorithms. One category of the machine learning algorithms can be utilized to accomplish 2 or more subtasks. For instance, the regression algorithms can be utilized for object localization as well as object detection or prediction of

the

movement.

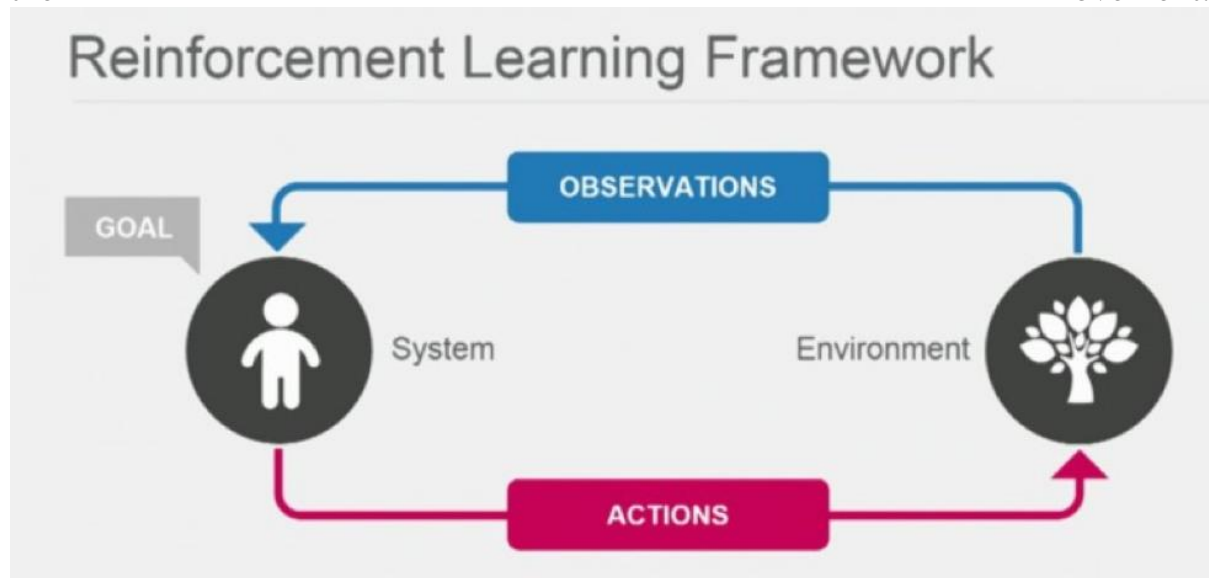


Figure 4.2.1 Reinforcement Learning Model

Decision Matrix Algorithms

The decision matrix algorithm systematically analyzes, identifies and rates the performance of relationships between the sets of information and values. These algorithms are majorly utilized for decision making. Whether a car needs to brake or take a left turn is based on the level of confidence these algorithms have on recognition, classification and prediction of the next movement of objects. The decision matrix algorithms are models composed of various decision models trained independently and in some way, these predictions are combined to make the overall prediction, while decreasing the possibility of errors in decision making. AdaBoosting is the most commonly used algorithm.

Clustering Algorithms

Sometimes, the images acquired by the system are not clear and it becomes difficult to locate and detect objects. Sometimes, there is a possibility of classification algorithms missing the object and in that case, they fail to categorize and report it to the system. The possible reason could be discontinuous data, very few data points or low-resolution images. The clustering algorithm is specialized in discovering the structure from data points. It describes the class of methods and class of problem like regression. The clustering methods are organized typically by modeling the approaches like hierarchical and centroid-based. All methods are concerned with utilizing the inherent structures in data to

organize the data perfectly into groups of maximal commonality. K-means, Multi-class Neural Network is the most commonly used algorithm.

K-means

K-means is a famous clustering algorithm. K-means stores k centroids that it utilizes for defining the clusters. A point is said to be in a specific cluster if it is closer to the centroid of that cluster than any other centroid. By alternating between choosing the centroids depending on the current assignment of data points to clusters and assigning the data points to clusters depending on current centroids.

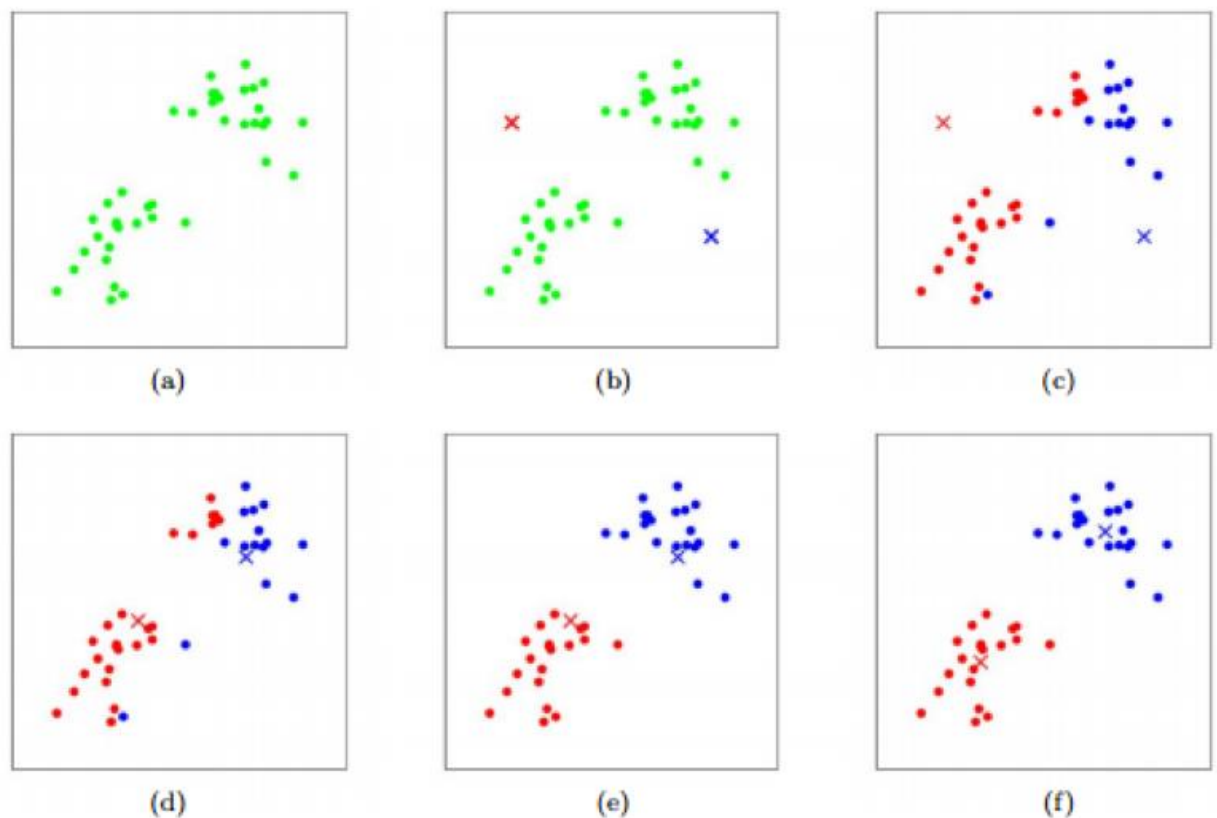


Figure4.2.2 K-means Algorithm – The cluster centroids are depicted as crosses and training examples are depicted as dots. (a) Original dataset. (b) Random initial cluster centroids. (c-f) The demonstration of running 2 iterations of k-means. Each training example is assigned in each iteration to the cluster centroid that is closest and then, each cluster centroid is moved to mean of points assigned to it.

Pattern Recognition Algorithms (Classification)

The images obtained through sensors in Advanced Driver Assistance Systems (ADAS) consists of all kinds of environmental data; filtering of the images is needed to determine the instances of an object category by ruling out the data points that are irrelevant. Before classifying the objects, the recognition of patterns is an important step in a dataset. This kind of algorithms are defined as data reduction algorithms.

The data reduction algorithms are helpful in reducing the dataset edges and polylines (fitting line segments) of an object as well as circular arcs to edges. Till a corner, the line segments are aligned with the edges and a new line segment will begin after this. The circular arcs align with the line segments' sequences that is similar to an arc. In various ways, the features of the image (circular arcs and line segments) are combined to form the features that are utilized for determining an object.

With the PCA (Principal Component Analysis) and HOG (Histograms of Oriented Gradients), SVM (Support Vector Machines) are the commonly used recognition algorithms in ADAS. The K nearest neighbor (KNN) and Bayes decision rule are also used.

Support Vector Machines(SVM)

SVM are dependent on the decision planes concept that define the decision boundaries. The decision plane separates the object set consisting of distinct class memberships. A schematic example is illustrated below. In this, the objects belong to either RED or GREEN class. A boundary line of separation separates the RED and GREEN objects. Any new object that falls to the left is labeled as RED and it is labeled as GREEN if it falls to the left.

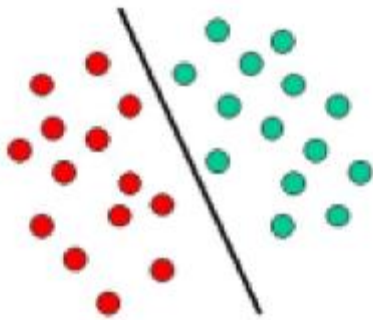


Figure 4.2.3 SVM

Regression Algorithms

This kind of algorithm is good at predicting events. The Regression Analysis evaluates the relation between 2 or more variables and collate the effects of variables on distinct scales and are driven mostly by 3 metrics:

- 1.The shape of regression line.
- 2.The type of dependent variables.
- 3.The number of independent variables.

The repeatability of the environment is leveraged by regression algorithms to create a statistical model of relation between the given object's position in an image and that image. The statistical model, by allowing the image sampling, provides fast online detection and can be learned offline. It can be extended furthermore to other objects without the requirement of extensive human modeling. An object's position is returned by an algorithm as the online stage's output and a trust on the object's presence.

The regression algorithms can also be utilized for short prediction, long learning. This kind of regression algorithms that can be utilized for self-driving cars are decision forest regression, neural network regression and Bayesian regression, among others.

REFERENCES

- Raspberry Pi Camera Module Documentation
- Arduino Uno Documentation
- MIT 6.S094 Machine Learning for Self Driving Car
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.

