1. INTRODUCTION TO AUTONOMOUS VEHICLE SYSTEMS

1.1 Introduction

The interface of the autonomous car with the surroundings must be similar to that of human way of interaction[1]. Humans use their eyes as a source of vision and then processes the visual signals in his/her brain and takes the necessary action accordingly. Similarly the autonomous car uses a camera as a visual source to know its surrounding, path etc. and uses the image processing techniques on the images received from the camera. This processing takes place on a minicomputer (RASPBERRY PI)[2]. After image processing[3], control instructions are passed on to the driving motors which helps in steering the vehicle accordingly.

In the future, automated systems will help to avoid accidents and reduce congestion. The future vehicles will be capable of determining the best route and warn each other about the conditions ahead. Many companies[4] and institutions are working together in countless projects in order to implement the intelligent vehicles and transportation networks of the future.

1.2 Background

An autonomous vehicle is fundamentally defined as a passenger vehicle that drives by itself. An autonomous vehicle is also referred to as an autopilot, driverless car, auto-drive car, or automated guided vehicle (AGV)[5]. Most prototypes that have been built so far performed automatic steering that were based on sensing the painted lines in the road. Today's researchers are using sensors and advanced software together with other custom-made hardware in order to assemble autonomous cars. Although the prototypes[6] seem to be very successful, a fully autonomous car that is reliable enough to be on the streets has not been constructed yet. This is mostly because of the difficulties involved in controlling a vehicle in the unpredictable traffic conditions of urban areas.

1.3 History

The first known worthy attempt to build an autonomous vehicle was in 1977[8]. The project research was carried out by Tsukuba Mechanical Engineering Laboratory in Japan. The car functioned by following white street markers and was able to reach speeds of up to 20 mph on a dedicated test course.

The breakthrough in the development autonomous vehicles came in the 1980's with the work of Ernst Dickmanns[9] and his team at Bundeswehr Universität München.

Their prototype was able to achieve 60 miles per hour on the roads without traffic.This event stands as gaining the most media coverage of any Intelligent Transportation System activity in US until the 2005 DARPA[10] Challenge. Nowadays we are looking forward to see the next DARPA Challenge that will take place in an urban environment in November 2007.

1.4 OBJECTIVES:

- Liberating the driver from the responsibility of operating the vehicle, thus leading the society towards lesser number of accidents.
- To decrease the traffic congestion by optimal utilization of roads.
- To provide better transportation to persons with disabilities and elders.

References:

1. http://www.rand.org/pubs/research_reports/RR443-2.html
2. https://www.raspberrypi.org
3. http://docs.opencv.org/2.4/doc/tutorials/imgproc/table_of_content_imgproc/table_of_content_imgproc.html
4. http://whatis.techtarget.com/definition/driverless-car
5. https://www.iaeme.com/MasterAdmin/uploadfolder/IJMET_03_01_003/IJMET_03_01_003.pdf
6. https://www.cbinsights.com/blog/autonomous-driverless-vehicles-corporations-list/
7. https://web.wpi.edu/Pubs/E-project/Available/E-project-043007-205701/unrestricted/IQPOVP06B1.pdf
8. https://web.wpi.edu/Pubs/E-project/Available/E-project-043007
9. https://www.lifehacker.com.au/2016/02/creator-of-the-worlds-first-self-driving-cars-ernst-dickmanns/
10. http://archive.darpa.mil/grandchallenge05/

2. Hardware Description

2.1 RASPBERRY PI:

Raspberry Pi is a mini computer[1] with a size of credit card. It was designed by Raspberry Pi foundation in United Kingdom for teaching basic computer skills, hardware, programming to school students and other interested people. Raspberry Pi is manufactured by companies like Newark element14[2] (Premier Farnell), RS Components and Egoman. Raspberry Pi uses Micro-SD card instead of traditional built-in hard disk for booting and storage purposes. Raspberry Pi is also sold online for a price around Rs.3000[3] in india.

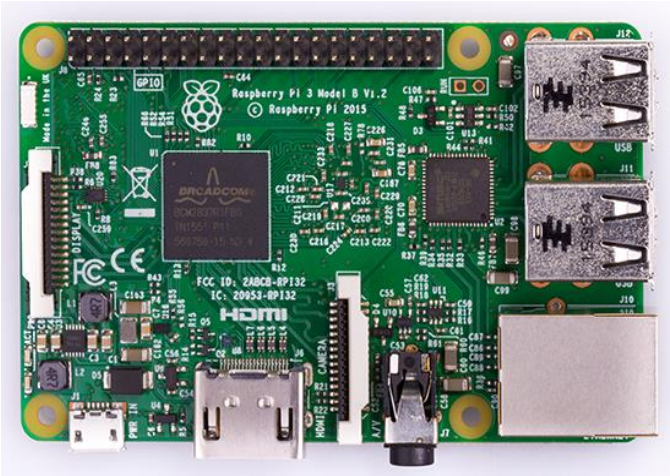


Fig.2.1. RaspberryPi[4]

2.1.1 Features of Raspberry Pi[5]:

- ARM v8 Quad core processor
- RAM of 1GB
- GPIO pins (40)
- 802.11n WLAN
- Four USB ports
- HDMI port
- Micro-SD slot
- CSI - Camera Serial Interface
- DSI – Display Serial Interface

2.1.2 A brief description of the components on the Pi:

1)Processor / SoC (System on Chip):

Raspberry Pi has a System on Chip module manufactured by Broadcom with ARMv8 processor.This SoC on the Raspberry Pi board is equivalent to the chip used by an old Android phone.

2) Power source

Raspberry Pi is powered using a Micro-USB charger. It requires 700mA of current for its operations. A smart phone charger can be used to provide required power to the Pi.

3) SD Card

Raspberry Pi uses Micro-SD card instead of traditional built-in hard disk for booting and storage purposes. A Micro-SD card slot is provided for inserting the SD card on which the operating system can be loaded using card reader on a computer

4) GPIO pins

GPIO stands for “General Purpose Input/Output”. GPIO pins are used as generic pins for Input or Output and these pins can be controlled by the user at the run-time.

[6]

Figure 2.2: GPIO connector on RPi

5) DSI Connector

The Display Serial Interface (DSI) is used to connect compatible LCD screen to the PI using some drivers to drive the display.

6) RCA Video

RCA is uused for video outputs and any television with RCA jack can be used as video output monitor.

.

Figure 2.3: RCA Video Connector[7]

7) Audio Jack

A 3.5mm audio jack is available on the Pi for audio output. Headphones or any 3.5mm audio cable can be used for audio output.

8) Status LEDs

Raspberry Pi has 5 status LEDs used to show the status of various activities like OK (SD card Access), POWER, FDX (Full Duplex), LNK (link),100Mbit (LAN).

9) USB 2.0 Port

Four USB 2.0 ports are provided on the Raspberry Pi board to connect other accessories like mouse or keyboard. If needed the number of USB ports can be increased by using USB hub.

10) Ethernet

Ethernet port is available on the Raspberry Pi board. It can be used to connect the Pi to the internet using a LAN cable.

11) CSI connector

CSI – Camera Serial Interface is a serial interface used to interface digital cameras with the PI.

12) HDMI

HDMI – High Definition Multimedia Interface usewd to connect with HDMI screens.

2.1.3 Operating System

Raspberry Pi is a micro-computer. All the modern day computers use operating system to perform their operations. Similarly Raspberry Pi also uses operating system. It uses an external Micro-SD card to store the operating system and directly boots the operating system from the SD card.A NOOBS installer is used to install the operating system in the SD card. A minimum of 8Gb SD card is required to install windows operating system. Different types of operating systems like Archlinux ARM, OpenELEC,Pidora, Raspbmc, RISC, Raspbian[8] etc. are available for Raspberry- Pi.

2.1.4 The NOOBS installer

Raspberry Pi package includes only the main board and it is not equipped with any operating system. Therefore an external SD card must be inserted in the slot provided on the main board and then the operating system must be installed onto the SD card using card reader on any computer.
Installing the operating system may be easy for few enthusiasts, but for the beginners it can be difficult. So to ease the process of installing an operating system, the Raspberry Pi foundation developed a software called NOOBS[9] – "New Out Of Box Software"

NOOBS installer is available on the official website and can be downloaded for free. After downloading NOOBS, copy it to the SD card and insert it in the SD card slot provided on the main board. On booting, NOOBS interface is loaded. User can select the operating system and install it. This is a convenient way of installing operating system for beginners.

2.2 WEB CAM:

A **webcam** is fundamentally a video camera that is used to feed or stream its image in real time to a computer or to a computer network. These images when "captured" by the computer, the video- stream can be saved or viewed or sent to other computer networks using internet. The web cam can be connected to computers and laptops using a USB cable provided with the web cam. Web cam's with various resolutions are available in the market. For example :640 x 480 pixels, RGB image [10].

Fig.2.4. Web Camera

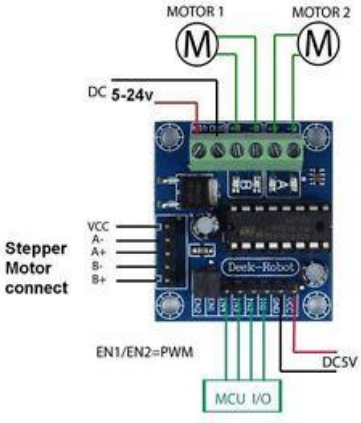
2.2.1 Features of webcam

- 500k pixels Resolution
- RGB24 Image Quality
- Auto or manual Exposure
- USB 2.0 interface
- 30 fps Frame rate
- 4cm to infinity focus range

2.3 L293D Motor driver IC:

2.3.1 Description

The L293D modules are used for driving the DC-motors. The L293D IC provides bidirectional drive currents of up to 1 A at voltages from varying from 4.5 V to 36 V. The L293D is characterized for operation in the temperature range of 0°C to 70°C.



[11]

Fig.2.5. L293D Motor Driver

2.3.2 Features

- Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- High Noise Immunity Inputs
- Output Current of 1A Per Channel (600 mA)
- Peak Output Current of 2A Per Channel (1.2 A)

References:

1. https://en.wikipedia.org/wiki/Raspberry_Pi
2. https://en.wikipedia.org/wiki/Farnell_element14
3. https://www.theguardian.com/technology/2016/feb/29/raspberry-pi-3-launch-computer-uk-bestselling
4. https://www.element14.com/community/community/raspberry-pi
5. http://www.wired.co.uk/article/raspberry-pi-three-wifi-bluetooth-release-price-cost
6. https://sites.google.com/site/semilleroadt/raspberry-pi-tutorials/gpio
7. https://thumb1.shutterstock.com/display_pic_with_logo/55860/55860,1134516380,4/stock-photo-audio-male-rca-connector-with-yellow-shell-809752.jpg
8. https://www.raspberrypi.org/downloads/raspbian/
9. https://www.raspberrypi.org/downloads/noobs/
10. http://www.qhmpl.com/QHM495LM-Web-Camera.html
11. http://i.ebayimg.com/images/g/Q6QAAOSwA3dYeotW/s-l300.jpg

3. Lane Detection and Bot Steering

3.1 Introduction:

Lane detection[1] is a process of recognizing the lanes on the road. Lane detection is the most important function for driving on roads. As there is rapid technology advancement in the field of Artificial intelligence in transportation, utilization of high calibrated sensors, efficient processing tools and algorithms which perform lane detection automatically.

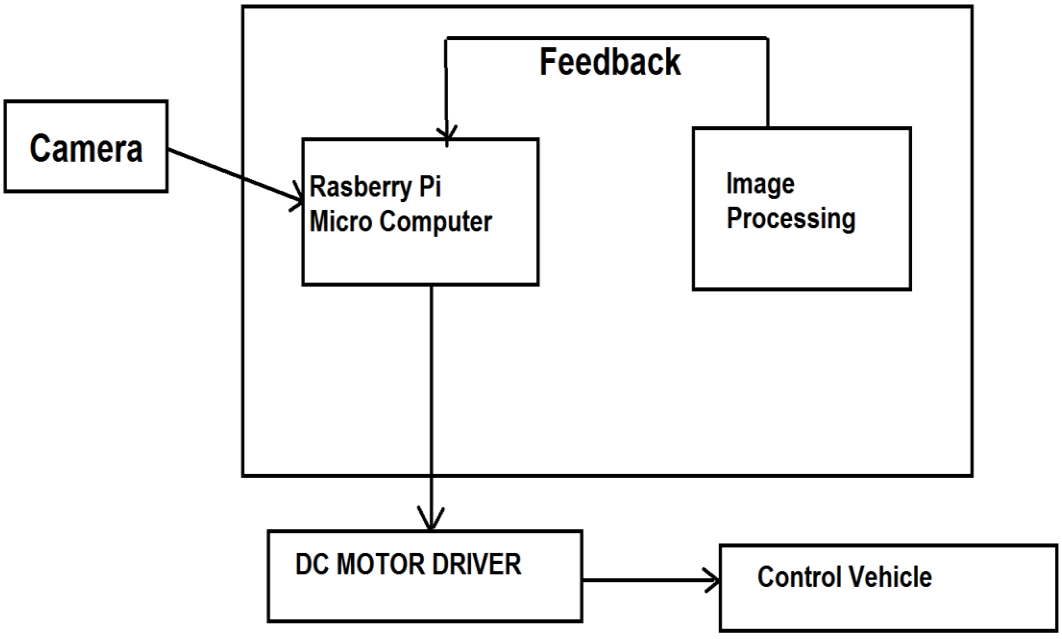


Figure 3.1 - Basic Block Diagram of Self Driving Vehicle

The control mechanism of an autonomous car consists of few main blocks as shown in **Figure 1**.

METHODOLOGY:

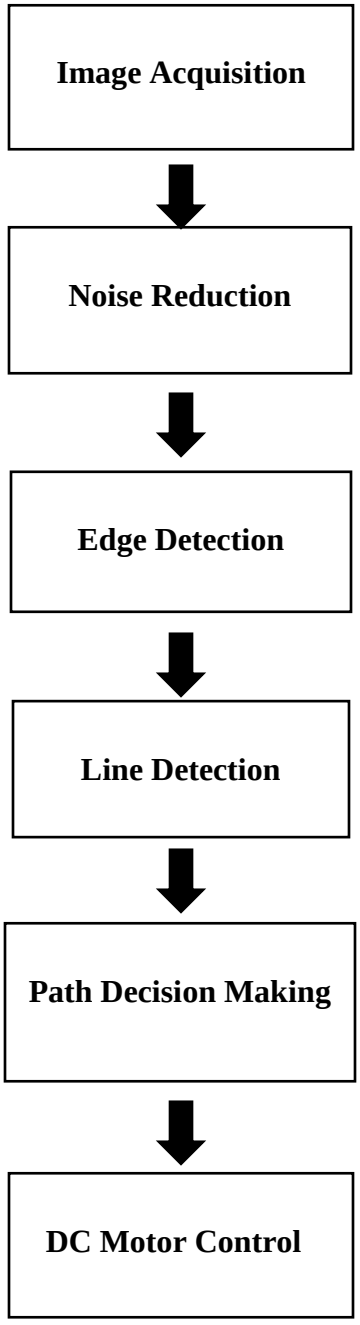


Fig 3.2 Steps involved in designing self-driving vehicle

Description:

- Webcam captures the Path images and gives input to Raspberry Pi for further processing.
- The color image from web camera is first converted into greyscale and then to Binary images using cvtColor ()[2] and threshold () respectively.
- Noise in the image is removed by using Erosion() and Dilation()[3] functions respectively.
- Edge Detection is performed by Canny () operator [4].
- Line detection is performed by using Hough Transform [5].
- Depending upon the lines we take our Steering Decision.

Consider a Baseline of width equal to width of vehicle. We steer the vehicle in a way such that detected lines of lanes never touch intersect the baseline.

3.2 Steps involved:

1. Crop the frame to Region of Interest.
2. Convert image into Grey Scale image
3. Convert image to Binary Image by selecting suitable threshold.
4. Do Image Processing to reduce noise.
5. Do edge detection in the image.
6. Apply Hough Transform to detect lines in the image
7. Draw lines in the image
8. Using Line Iterator from the center, find the first line on both left and right.
   Find the mid point of both the points. This gives center of lane.
9. Compare the mid point of the lane with camera center and steer the bot accordingly.

Step 1: Crop Region of Interest

Code:

```
int height = (img1.rows / 2), width = (img1.cols);
                Rect r(0, height, width, height - 1);
                ROI = img1(r);
```

Track generally present in the lower half of the image using these lines of code we crop the image to bottom part of it. This enables us to prevent detection of unwanted lines in the image.

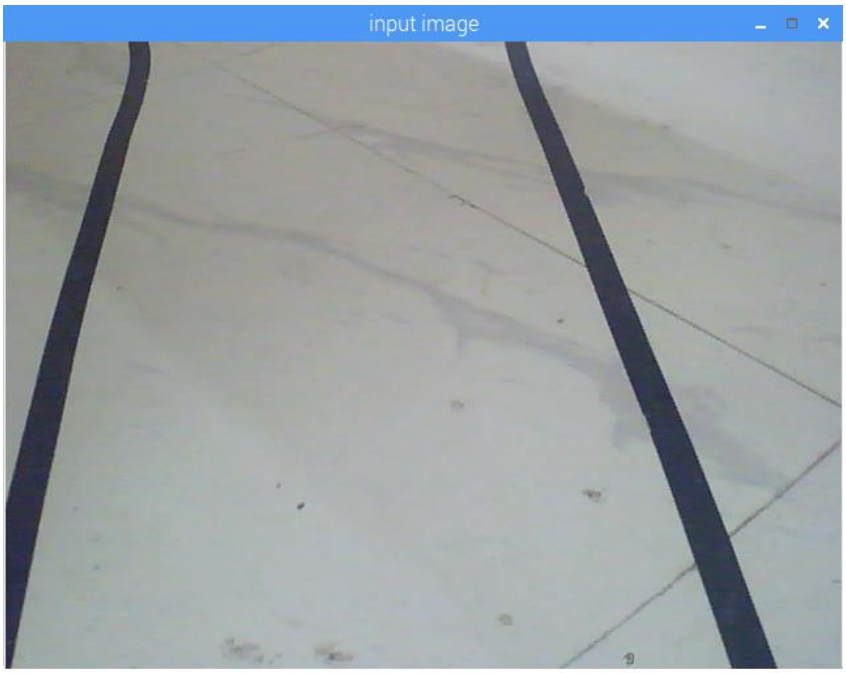


Fig:3.3 Input Image

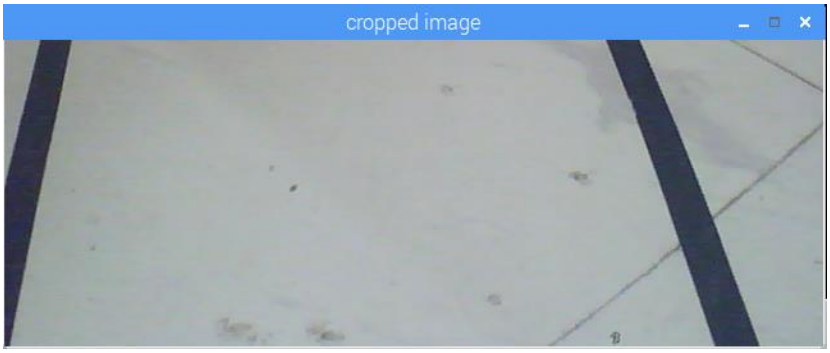


Fig:3.4 Cropped Region of Interest

Step 2: Convert Grey Scale[6]

Code:

```
cvtColor(ROI, grey, CV_RGB2GRAY);
```

Using the above function, **cvtColor()** we convert the colour image to Grey Scale. By converting we increase processing speed.

- **cvtColor(source image, destination image, conversion property):**

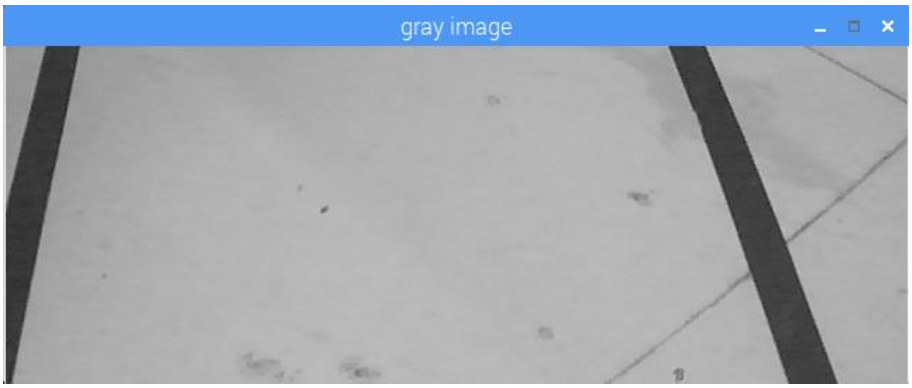


Fig:3.5 Gray Scale Image

Step 3: Convert to Binary Image[7]

Code:

```
threshold(grey, binary, bin_threshold, 255, THRESH_BINARY);
```

Using the above function, **threshold()** we convert the Grey image to Binary Image. By converting we increase processing speed.

- **Threshold (source image, destination image, double threshold, double max_val, int type)**

The function is used to get an image where all the pixels are either white(255) or black(0). Every pixel in the image is compared with threshold value given, if the pixel value is greater than threshold, the pixel is converted to white and if the pixel value is less than threshold, the pixel is converted to black.

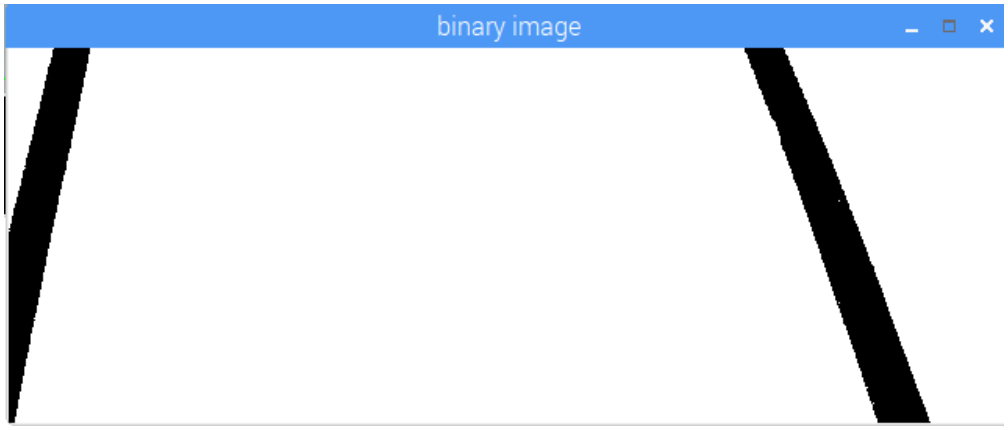


Fig:3.6 Binary Image

Step 4: Image Processing to reduce noise

Code:

```
dilate(binary, dil, Mat(), Point(-1, -1), 1);
erode(dil, erod, Mat(), Point(-1, -1), 5);
```

Using the above function, **dilate()**[8] we compare the pixel value with all eight neighboring pixel values and replaces the pixel value with maximum value in the neighborhood[4].

- **Dilate (source image, destination image, Input_Array kernel_matrix, Point_ anchor=Point(-1,-1), int it=1)**

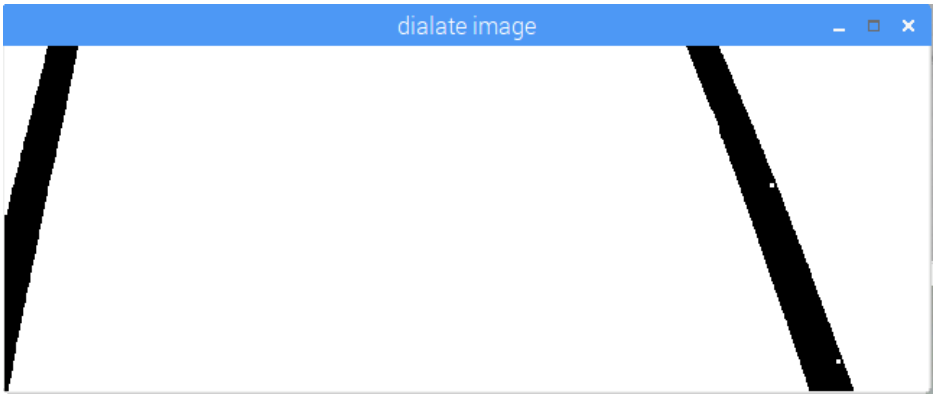


Fig:3.7 Dilated Image

Using the above function, **erode()[9],**we compare the pixel value with all eight neighboring pixel values and replaces the pixel value with minimum value in the neighborhood [5].

- **Erode (source image, destination image, Input_Array kernel, Point_ anchor=Point(-1,-1), int it=1)**

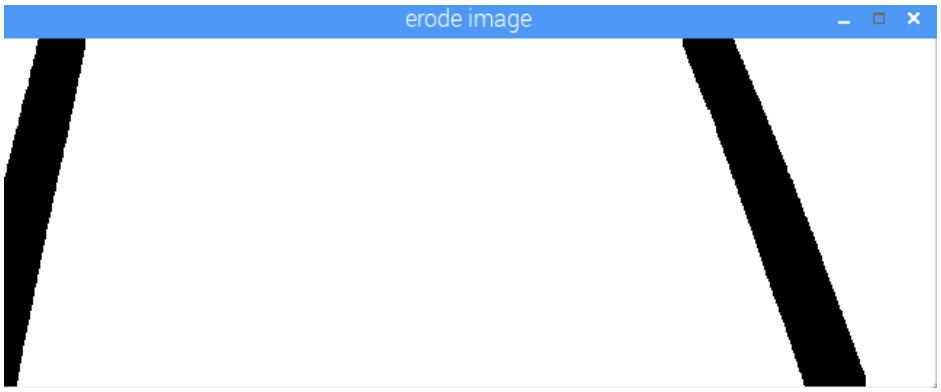


Fig:3.8 Eroded Image

Step 5: Edge Detection

Code:

```
Canny(erod, cany, low, high);
```

Using the above function, **Canny()[10],** we find edges throughout the image. We used Canny function because it has Low error rate, Good localization, Minimal response.

- **Canny (source image, destination image, double threshold_value1, double threshold_value2)**

Fig:3.9 Edge detected Image

Step 6: Line detection in image using Hough Transform

Code:

```
HoughLines(cany, lines2, 1, CV_PI / 180, thresholdd, 0, 0);
```

Hough Transform[11]:

1. "Hough Line Transform" is a transform used to detect "Straight Lines".
2. Before applying Hough Transform, an edge detection pre-processing is desirable.

- **void HoughLines (Input_Array image, Output_Array lines, Double rho, Double theta, int threshold, Double srn=0, Double stn=0)**

Step 7: Draw Lines in the image

Code:

```
if (lines.size())
                {
                        for (size_t i = 0; i < lines.size(); i++)
                        {
                                Vec4i l = lines[i];
                                line(ROI, Point(l[0], l[1]), Point(l[2], l[3]),
Scalar(0, 0, 255), 10);

                        }
                }
```

Fig: Line detected via Hough Transform

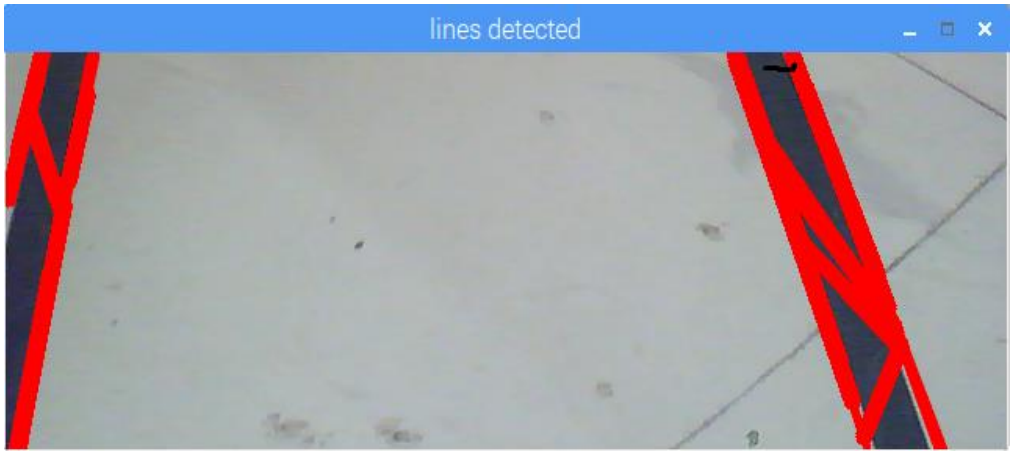


Fig 3.10 :Lines detected image

Step 8: Using Line Iterator[12] for Detection of lines on left and right from the center

Code:

```
buf[0] = Point(10, ROI.rows-10);
		buf[1] = Point(630,ROI.rows-10);
		/*buf[3] = Point(50,ROI.rows-10);
		buf[4] = Point(590, ROI.rows-10);*/
		LineIterator it(ROI, Point(320, ROI.rows-10), Point(0, ROI.rows-
10), 8);
		LineIterator it2(ROI, Point(320, ROI.rows-10), Point(640,
ROI.rows-10), 8);

		for (int i = 0; i < it.count; i++, ++it)
		{
			Vec3b val1 = ROI.at<Vec3b>(it.pos());
			//cout << (int)val1(2) << "  " << it.pos() << endl;

			if ((int)val1(2) == 255)
			{
				buf[0] = it.pos();
				break;
				//cout << it.pos() << endl;
			}
		}
		// alternative way of iterating through the line
		for (int i = 0; i < it2.count; i++, ++it2)
		{
			Vec3b val2 = ROI.at<Vec3b>(it2.pos());
			//cout << (int)val2(2)<<"  "<<it2.pos()<<endl;
			if ((int)val2(2) == 255)
			{
				buf[1] = it2.pos();
				break;
				//cout << it2.pos() << endl;
			}
		}
		buf[2].x = (buf[0].x + buf[1].x) / 2;
		buf[2].y = (buf[0].y + buf[1].y) / 2;
		line(ROI, Point(ROI.cols / 2, 0), Point(ROI.cols / 2, ROI.rows-
10), Scalar(0, 255, 0), 1);
		line(ROI, Point(0, ROI.rows-10), Point(640, ROI.rows-10),
Scalar(0, 255, 0), 1);
		line(ROI, buf[2], buf[2], Scalar(255, 0, 0), 10);
		//imshow("final", ROI);
```

LineIterator it (input image, Starting point, Ending point, gap between each pixel check);

Process to detect nearest left and right lines:

1. From centre check the first red pixel in the image to left. This gives the nearest line from the centre to the left.
2. From centre check the first red pixel in the image to right. This gives the nearest line from the centre to the right.

In the above code buf[0] contains nearest left red pixel co-ordinates and buf[1] has right nearest.We calculate buf[2], the mid point of buf[0] and buf[1].This represents the centre of the lane.

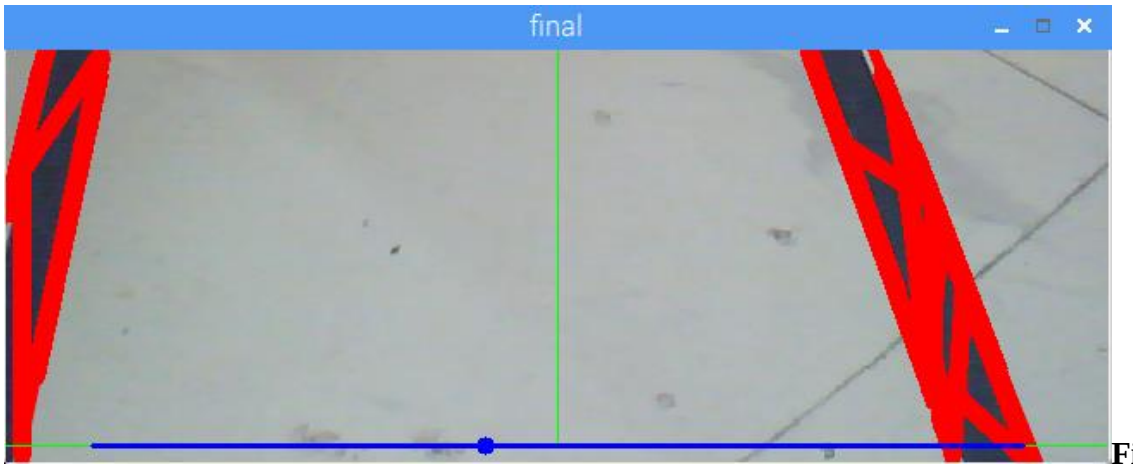


Fig 3.11 Final Image.

Step 9: Compare the mid point with center of image and make the steering decision.

Code:

```
if (buf[2].x>320)
    {right();
    exit(0);
    }
else if(buf[2].x<320)
 {
 left();
 exit(0);
 }
else{
straight();
}
```

Decision making:

1. If midpoint of the lane is in the left half of the image, we steer the bot to right.
2. If midpoint of the lane is in the right half of the image, we steer the bot to leftt.
3. Else straight.

References:

1. https://hal.inria.fr/hal-00781274/file/RT-433.pdf
2. http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html
3. http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilation.html
4. http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html
5. http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html
6. http://docs.opencv.org/2.4/modules/ocl/doc/image_processing.html?highlight=cvtcolor
7. http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?highlight=threshold#cv2.threshold
8. http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate#dilate
9. http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=erode#erode
10. http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html?highlight=canny
11. http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html
12. http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html?highlight=lineiterator

4. DC motor control

Using wiring pi libraries we control the input output pins of the raspberry pi. These are given to L293d motor driver IC which controls the motors accordingly.

4.1 WiringPi Introduction :-

WiringPI is a pin based General Purpose Input Output(GPIO) access library in C and C++ used in the Raspberry Pi digital pin control[1] . General input output pin controlling was done in Python language but we are implementing image processing in c++, Therefore to control IO pins in c++ we use wiringPi**.** Using wiring pi library, we can control the input output pins of the raspberry pi. Raspberry pi is directly connected to L293d motor driver IC which controls the motors accordingly.

4.2 Steps for robot motors setup :-

- **Defining raspberryPi GPIO pins for motors**

```
#definemr1    0
#definemr2    1
#defineml1    2
#defineml2    3
```

- **Initial setup for PIN configuration for input/output operation.**

```
wiringPiSetup () ;
pinMode (mr1, OUTPUT) ;
pinMode (mr2, OUTPUT) ;
pinMode (ml1, OUTPUT) ;
pinMode (ml2, OUTPUT) ;
```

- **Bot Steering Functions**

```
void straight()
{
digitalWrite (mr1, HIGH) ;
digitalWrite (mr2, LOW) ;
digitalWrite (ml1, HIGH) ;
digitalWrite (ml2, LOW) ;
}
void right()
{
digitalWrite (mr1, LOW) ;
digitalWrite (mr2, LOW) ;
digitalWrite (ml1, HIGH) ;
digitalWrite (ml2, LOW) ;
}
void left()
{
digitalWrite (mr1, HIGH) ;
digitalWrite (mr2, LOW) ;
digitalWrite (ml1, LOW) ;
digitalWrite (ml2, LOW) ;
}
```

These functions change the corresponding pins to either high or low enabling the motor driver to turn the motors according to the command.

References :-

1. http://wiringpi.com/

5. Object Detection

Object detection refers to a method of identifying and finding the object of certain class through out the image. Here we dealing with some image processing tools to discover the required object. There are many classifiers and machine learning tools available for object detection, we are using Haar cascade technique which has high detection rate[1].

5.1 Haar cascade :-

Haar-cascade classifier includes rectangular regions which are in neighborhood at a particular location in a specified window, it aggregates the pixel intensities in all regions and finds gap between these totals. This calculated difference is used to differentiate sub parts of given image. Face detection, Eye detection, Pattern detection in the image are some of the examples. In all the above examples, 2 neighboring rectangle regions are compared.

Haar-cascade system finds the object in the image by a moving window across the image. The classifier contains a list of cascade stages, where every stage contains weak learners. At every stage of classifier, a specific image part is defined by the location of window differentiating the image part as either positive or negative. Positive means that object was found and negative meaning that required object is missing in the whole image.

- If classifier labelling gives a negative result, therefore the required object is not found at that specific region and hereby the window location is shifted to respective next location.
- If classifier labelling gives a result of positive, therefore the object is found in the image and the classifier is shifted to next stage.
- If the classifier labelling gives a verdict of result positive, only when all stages finds the required object in the image.

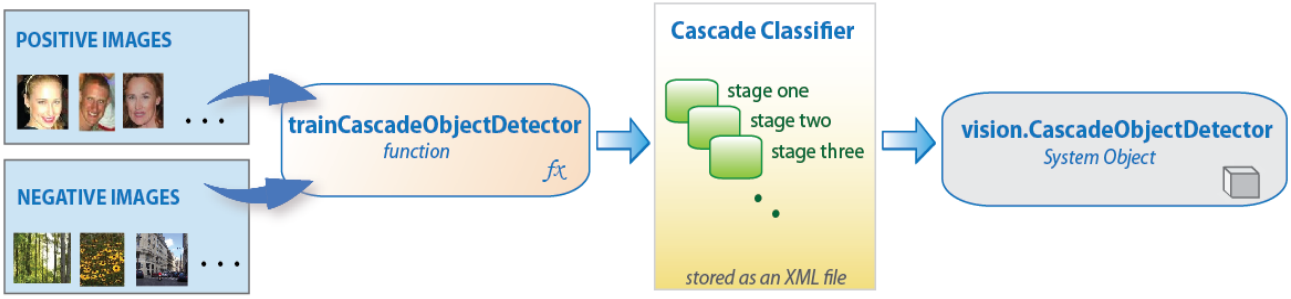


Fig.,5.1 Haar cascade working Process

5.2 Steps involved in Haar training[2] :-

- Positive samples Description.
- Negative samples Description.
- Creating a vec file from positive samples.
- Haar Cascade classifier training.
- Creating xml from cascade training.
- Object detection by using the generated xml file.

5.3 Implementing object detection[3]:

Training cascade:

- We have taken 20 positive images which consists of Stop Road sign

Fig.5.2 Positive sample images

- We have taken 30 negative images which does not consists of Stop Road sign and trained our cascade for 10 iterations.

Creating vector file:

"opencv-createsamples -info positive_samples.txt -vec vector_file.vec -w 30 -h 40"

Creating xml file

"opencv-haartraining -data haar -vec vector_file.vec -bg negative_samples.txt -nstages 15 -mem 2000 -mode all -w 30 -h 40"

- Final result is exported to xml file to use in detection

5.4 Object Detection Code:

```
void detect(Mat frame)

{
	std::vector<Rect> stopsign;
	Mat frame_gray;

	cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
	equalizeHist(frame_gray, frame_gray);
	//-- Detect faces

	stop_cascade.detectMultiScale(frame_gray, stopsign, 1.1, 2, 0 |
CV_HAAR_SCALE_IMAGE, Size(10,10));
	for (size_t i = 0; i < stopsign.size(); i++)
	{
		Point center(stopsign [i].x + stopsign [i].width / 2, stopsign [i].y + stopsign
[i].height / 2);
		ellipse(frame, center, Size(stopsign [i].width / 2, stopsign [i].height / 2), 0, 0,
360, Scalar(255, 0, 255), 2, 8, 0);
		imwrite("stop1.jpg",frame);
		stop();
		delay(5000);
	}

}
```

detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size())

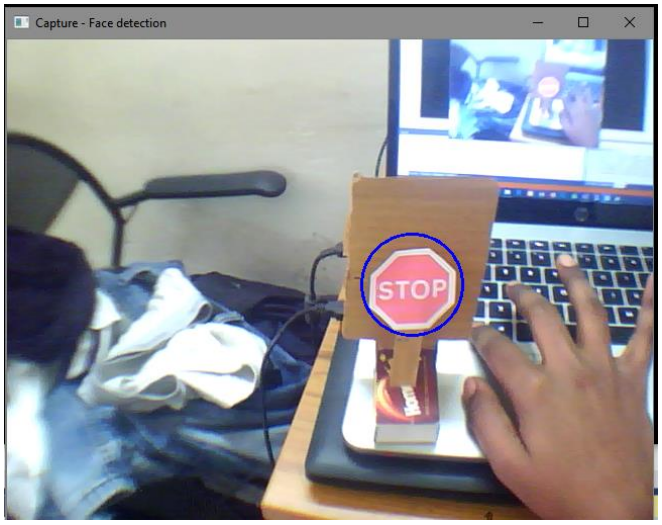


Fig.5.3 Sign Detected

Working of Code:

- After the completion of Classifier training, it is applied to either a part of an image or to a whole input image[4].
- When haar cascade classifier gives an output “1”, if the required object is found in the input image.
- When haar cascade classifier gives an output “0”, if the required object is found missing in the input image.
- In search of required object, the window is moved across the whole image to reach every location of image.
- Haar cascade designs classifier in such a way that it is resized easily, therefore different sized objects are detected efficiently.
- It implements scanning procedure at various scales and also done many times to find positive samples throughout the image.

References:

1. http://ds.cs.ut.ee/Members/artjom85/2014dss-course-media/Object%20detection%20using%20Haar-final.pdf
2. https://achuwilson.wordpress.com/tag/object-marker/
3. https://achuwilson.wordpress.com/2011/07/01/create-your-own-haar-classifier-for-detecting-objects-in-opencv/#more-31
4. http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

6. Final Implemented code

```
#include<iostream>
#include<math.h>
#include <opencv2/opencv.hpp>
#include <stdio.h>
#include <wiringPi.h>
#include <sys/types.h>
#include <unistd.h>
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#definemr1     0
#definemr2     1
#defineml1     2
#defineml2     3


using namespace std;
using namespace cv;

void detectAndDisplay(Mat frame);

String stop_cascade_name = "cascade.xml";
CascadeClassifier stop_cascade;
string window_name = "Capture - Stop detection";
int fps=60;



void stop()
{
digitalWrite (mr1, LOW) ;
digitalWrite (mr2, LOW) ;
digitalWrite (ml1, LOW) ;
digitalWrite (ml2, LOW) ;
}

void s (void)
{
  wiringPiSetup () ;
  pinMode (mr1, OUTPUT) ;
   pinMode (mr2, OUTPUT) ;
    pinMode (ml1, OUTPUT) ;
     pinMode (ml2, OUTPUT) ;
```

```
    }
void straight()
{
digitalWrite (mr1, HIGH) ;
digitalWrite (mr2, LOW) ;
digitalWrite (ml1, HIGH) ;
digitalWrite (ml2, LOW) ;

//cout<<"straight"<<endl;
}
void right()
{
digitalWrite (mr1, LOW) ;
digitalWrite (mr2, LOW) ;
digitalWrite (ml1, HIGH) ;
digitalWrite (ml2, LOW) ;
delay(fps);
straight();

//cout<<"d right";
}
void left()
{
digitalWrite (mr1, HIGH) ;
digitalWrite (mr2, LOW) ;
digitalWrite (ml1, LOW) ;
digitalWrite (ml2, LOW) ;
delay(fps);
straight();
//cout<<"d left ";
}

int low = 18, high = 135, thresholdd = 100,thresholdp=2, linelength = 50, maxlinegap =
200, bin_threshold = 80;


void detectAndDisplay(Mat frame)
{
        std::vector<Rect> faces;
        Mat frame_gray;

        cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
        equalizeHist(frame_gray, frame_gray);
        //-- Detect faces
        face_cascade.detectMultiScale(frame_gray, faces, 1.1, 2, 0 |
CV_HAAR_SCALE_IMAGE, Size(10,10));
```

```
        for (size_t i = 0; i < faces.size(); i++)
        {
                Point center(faces[i].x + faces[i].width / 2, faces[i].y + faces[i].height / 2);
                ellipse(frame, center, Size(faces[i].width / 2, faces[i].height / 2), 0, 0, 360,
Scalar(255, 0, 255), 2, 8, 0);
                imwrite("stop1.jpg",frame);
                stop();
                delay(5000);
        }
        //-- Show what you got
        //imshow(window_name, frame);
}


int main()
{
   s();
        Mat img1, ROI, grey, binary, dil, erod, blur, cany;
        face_cascade.load(face_cascade_name);
        vector<Vec4i> lines;
        vector<Point> buf(3);
   //string imagedirectory =
"C:/Users/praveen_gadi/Documents/IMG_20170202_195649.jpg";
        //string imagedirectory = "D:/opencv/WIN_20170203_16_59_09_Pro.mp4";
        //string ip="track.mp4";
        VideoCapture cap(0);
        while (1){

                cap >> img1;
                //imshow("input image",img1);                detectAndDisplay(img1);
                //pid_t pid1=fork();
                //if(pid1==0){
                int height = (img1.rows / 2), width = (img1.cols);
                Rect r(0, height, width, height - 1);
                ROI = img1(r);
      //imshow("cropped image",ROI);
                cvtColor(ROI, grey, CV_RGB2GRAY);
                //imshow("gray image", grey);
                threshold(grey, binary, bin_threshold, 255, THRESH_BINARY);
                //GaussianBlur(grey, blur, Size(-1, -1), 2,2);
                //imshow("binary image", binary);
                dilate(binary, dil, Mat(), Point(-1, -1), 1);
                //imshow("dialate image", dil);
                erode(dil, erod, Mat(), Point(-1, -1), 5);
                //imshow("erode image", erod);
                Canny(erod, cany, low, high);
```

```
		//imshow("edge detected cany", cany);
		HoughLinesP(cany, lines, 1, 3.14 / 180, thresholdp, linelength,
maxlinegap);
		if (lines.size())
		{
			for (size_t i = 0; i < lines.size(); i++)
			{
				Vec4i l = lines[i];
				line(ROI, Point(l[0], l[1]), Point(l[2], l[3]), Scalar(0, 0,
255), 10);

			}
		}
		//imshow("lines detected", ROI);

		buf[0] = Point(10, ROI.rows-10);
		buf[1] = Point(630,ROI.rows-10);
		/*buf[3] = Point(50,ROI.rows-10);
		buf[4] = Point(590, ROI.rows-10);*/
		LineIterator it(ROI, Point(320, ROI.rows-10), Point(0, ROI.rows-10), 8);
		LineIterator it2(ROI, Point(320, ROI.rows-10), Point(640, ROI.rows-10),
8);

		for (int i = 0; i < it.count; i++, ++it)
		{
			Vec3b val1 = ROI.at<Vec3b>(it.pos());
			//cout << (int)val1(2) << "  " << it.pos() << endl;

			if ((int)val1(2) == 255)
			{
				buf[0] = it.pos();
				break;
				//cout << it.pos() << endl;
			}
		}
		// alternative way of iterating through the line
		for (int i = 0; i < it2.count; i++, ++it2)
		{
			Vec3b val2 = ROI.at<Vec3b>(it2.pos());
			//cout << (int)val2(2)<<"  "<<it2.pos()<<endl;
			if ((int)val2(2) == 255)
			{
				buf[1] = it2.pos();
				break;
				//cout << it2.pos() << endl;
			}
```

```
                    }
                    buf[2].x = (buf[0].x + buf[1].x) / 2;
                    buf[2].y = (buf[0].y + buf[1].y) / 2;
                    line(ROI, Point(ROI.cols / 2, 0), Point(ROI.cols / 2, ROI.rows-10),
Scalar(0, 255, 0), 1);
                    line(ROI, Point(0, ROI.rows-10), Point(640, ROI.rows-10), Scalar(0, 255,
0), 1);
                    line(ROI, buf[2], buf[2], Scalar(255, 0, 0), 10);
                    //imshow("final", ROI);
       pid_t pid=fork();
       if(pid==0){
       //cout<<"xxxxx"<<endl;
       if (buf[2].x>320)
          {right();
          exit(0);
          }
      else if(buf[2].x<320)
        {
        left();
        //cout<<"left"<<endl;
        exit(0);
        }


       else{
       straight();
       //cout<<"straight"<<endl;
       }
         exit(0);
         }

                    //}
   waitKey(10);
          }

return 0;
}
```