LANE DETECTION AND LANE FOLLOWING USING **OPENCV**

- ABSTRACT: The project is about a Bot which does the Lane Detection and then follows the lane accordingly. **OpenCV** concept has been used as the platform of doing this task. OpenCV (open source computer vision) is a library of programming functions mainly aimed at **real-time computer vision**. This Bot is a prototype for defining what automation is all about.

- INTRODUCTION: While driving a car, the basic fundamental we follow is lane (which is somewhat different in India) .So here one step is taken toward making my Bot understand what is lane and where is has to move or let's say to make it an autonomous Bot.
  The Bot has a **piCamera** which captures the video which is processed using the libraries of the openCV and then the lane is being detection out of the image which the Bot follows whether it is a curved path or a straight path .It is a 3 wheel (steerable front wheel) Bot .
  The Bot is constructed using the basic component like H bridge motor driver, servo motor and DC motors.

Raspberry pi board is used which is programmed to control these components as per the requirement.
The piCamera is attached to the Raspberry pi so that it can capture the video and it can be processed by the code.

COMPONENTS DESCRIPTION:

RASPBERRY PI 3B

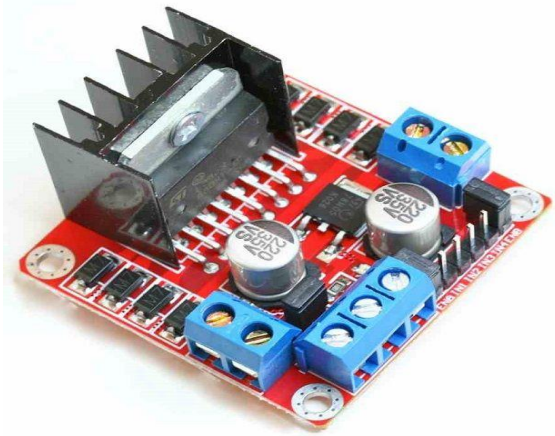
RASPBERRY PI: The **Raspberry Pi** is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.

Raspbian comes preloaded with Python, the official programming language of the Raspberry Pi and IDLE 3, a Python

Integrated Development Environment. We're going to show you now how to get started with IDLE and write your very first, albeit simple

Raspberry Pi is neither a microprocessor nor **microcontroller**, it is a single board computer which contains a SOC (System On Chip - Has multicore processor, GPU, ROM, I/O Peripherals inside it.), DDR RAM memory, Ethernet port, USB host, micro HDMI on it. The **Raspberry Pi** is slower than a modern laptop or desktop but is still a complete Linux computer and **can** provide all the expected abilities that implies, at a low-power consumption level.

MOTOR DRIVER



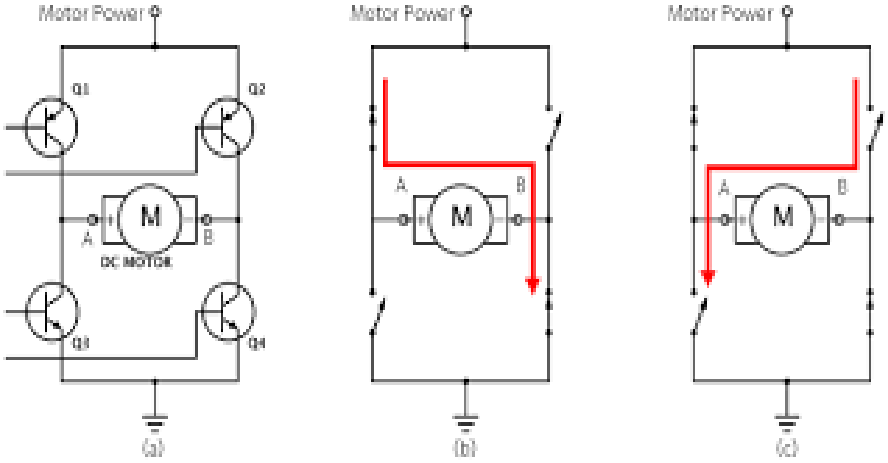
L298N Motor driver

A **motor driver** is a little current amplifier; the function of **motor drivers** is to take a low-current control signal and then turn it into a higher-current signal that can drive a **motor**.

The **L298** 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 levels and drive inductive loads such as relays, solenoids, DC and stepping **motors**

It is a H **bridge** motor driver .An **H bridge** is an electronic circuit that enables a voltage to be applied across a load in opposite direction. These circuits are often used in robotics and other applications to allow DC motors to run forwards or backwards.



H Bridge Circuit Diagram

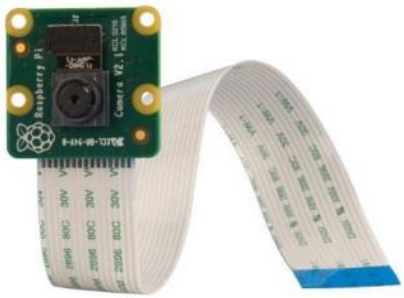
SERVO MOTOR:

A **servomotor** is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

LEVEL SHIFTER:

A **level shifter** in digital electronics, also called a **logic-level shifter**, is a circuit used to translate signals from one logic level or voltage domain to another, allowing compatibility between ICs with different voltage requirements .Here it is used to shift the voltage level from 5 v to 3.3v as only 3.3v can be taken from raspberry pi

PI CAMERA:



The Raspberry Pi camera module can be used to take high-definition video, as well as stills photographs.

Other components are:

- DC Motors
- 12v Battery

SOFTWARE REQUIRED:

OPENCV - OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine. It has C++, Python, Java and MATLAB interfaces and supports Windows. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE

Here in this project, Python is used for the interface. OpenCV has many inbuilt libraries which is required for the lane detection purpose which makes it easy to do.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

The challenge:

When we drive a car, we use our eye to decide where to go. The lines on the road that show us where the lane are act as our constant reference for steer the vehicle. So naturally one of the first thing we do is detecting the lanes using the algorithm.

OpenCV Libraries used:

Gaussian Blur: In image processing, a **Gaussian blur** (also known as **Gaussian smoothing**) is the result of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss). It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect

of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen.

Gaussian Blur is used to make the edges smoother.

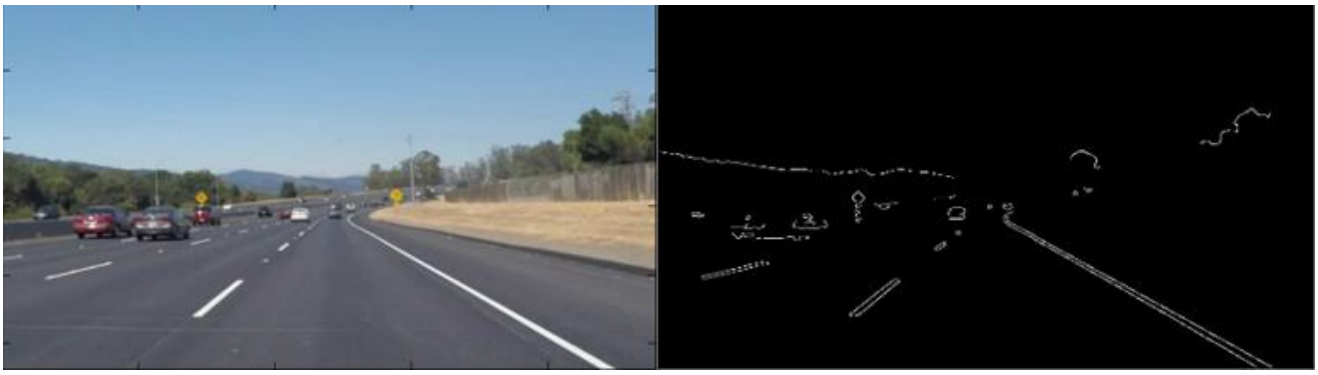
Gray Scaling

The images should be converted into gray scaled ones in order to detect shapes (edges) in the images. This is because the **canny edge detection** measures the magnitude of pixel intensity changes or gradients (more on this later).

Canny edge detection:

Canny edge detection which make use of the fact that edges has high gradients (how sharply image changes — for example, dark to white)

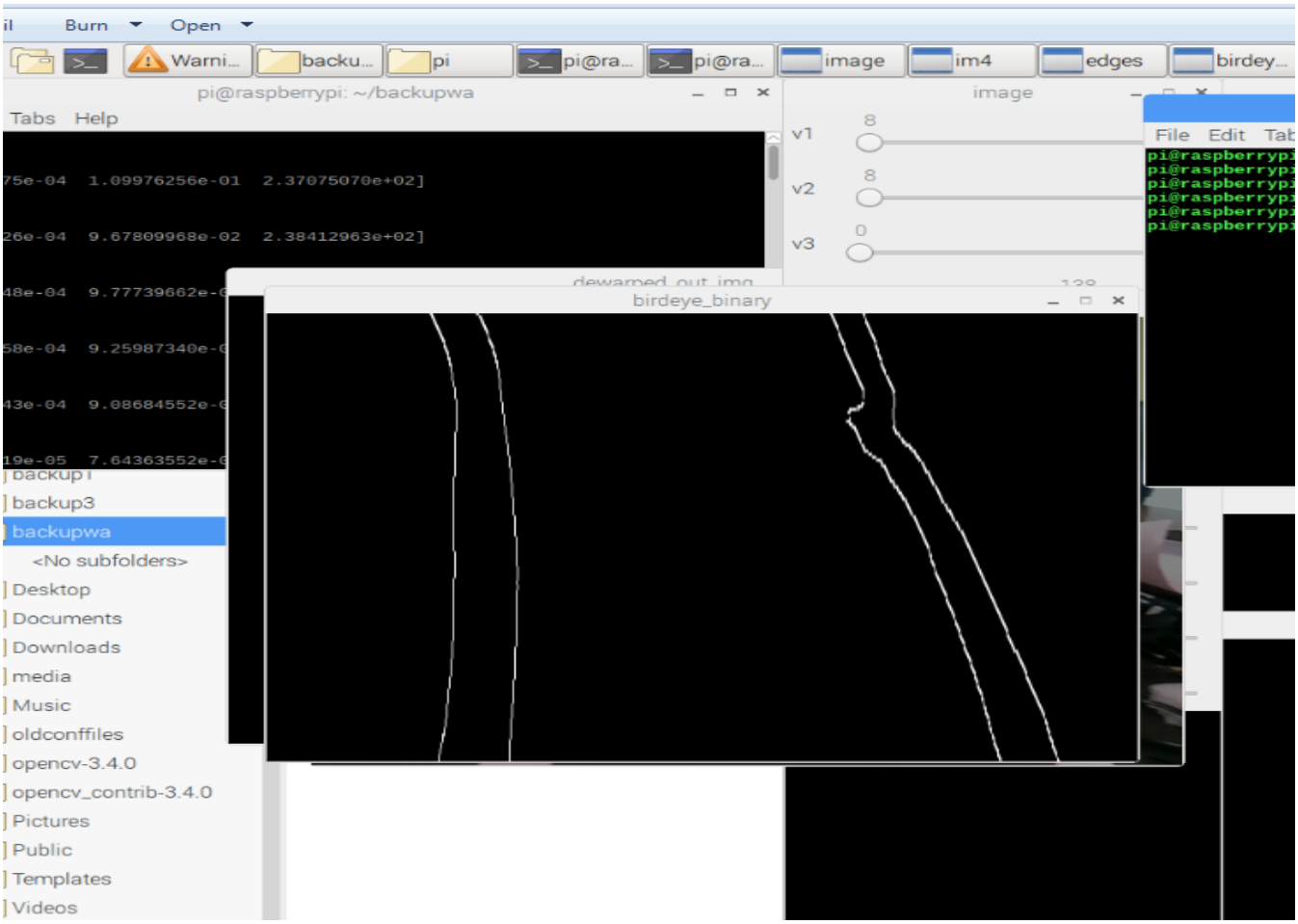
OpenCV functions for that: **cv.Canny()**



Region of interest: When finding lane lines, we are not interested in other environmental disturbances .So we mask the other regions as per our requirement.

Perspective transform:

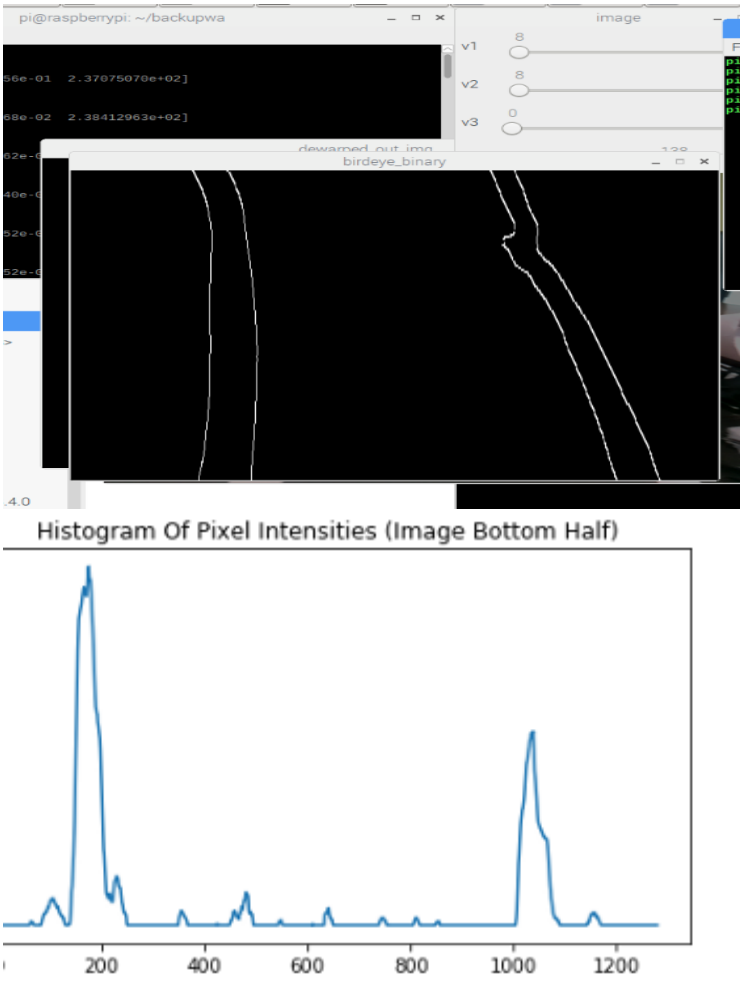
We now need to define a trapezoidal region in the 2D image that will go through a perspective transform to convert into a bird's eye view. It's the change in the perspective when u see the image from side and when u see it as top-down view.



Histogram

We then compute a histogram of our binary thresholded images in the y direction, on the bottom half of the image, to identify the x positions where the pixel intensities are highest.

Here the frame is divided into two parts, left and right sides and that is how we get the exact indices of the frame. The operation is done on both sides to detect the lane on both the sides.

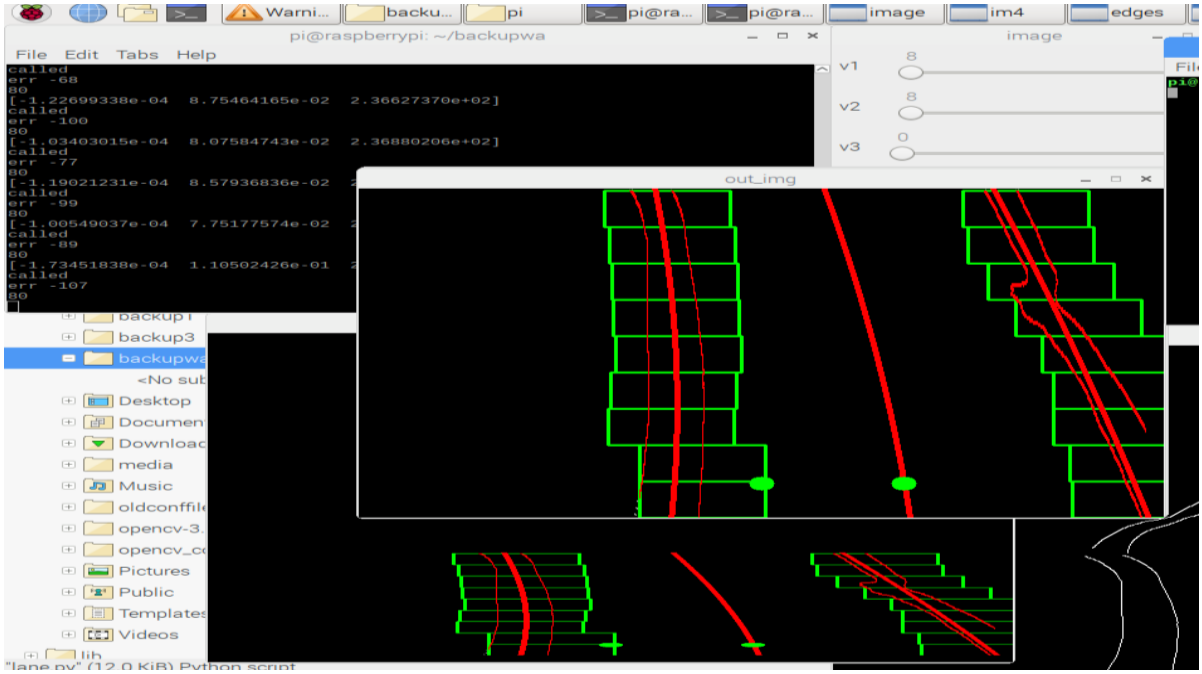


Top: Bird's eye view binary representation of lane. Bottom: histogram of pixel intensities of the image.

Sliding window:

Since we now know the starting x position of pixels (from the bottom of the image) most likely to yield a lane line, we run a sliding windows search in an attempt to "capture" the pixel coordinates of our lane lines.

From then, we simply compute a second degree polynomial, via numpy's polyfit, to find the coefficients of the curves that best fit the left and right lane lines.



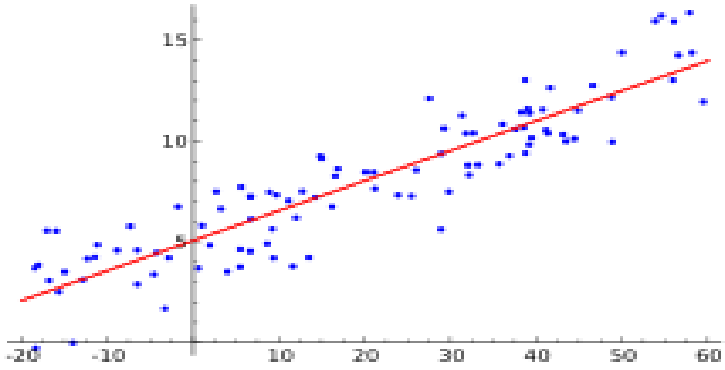
sliding windows :Green boxes

Use of polynomial equation:

We know for a straight line y=mx+c

As the degree of equation increases it is makes a curved line Y=ax^2+bx+c.

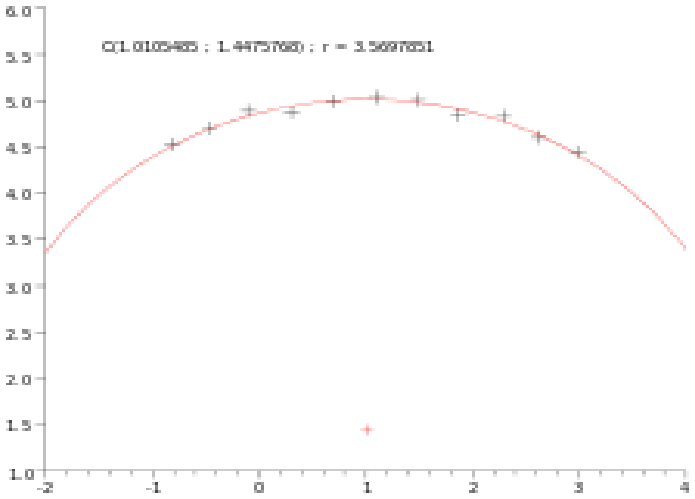
Regression:



If x coordinates are independent coordinates then y are dependent coordinates .So we have to find out the line which can be fitted to all the points using the root mean square method to minimize the error

So this is for the straight line to minimize the error we can go for a curve line that is second degree of polynomial.

But if degree is increased much it increase the probability of error can increase.



Some function are available to find the coefficients and the coordinates of the dependent variable. Such as:

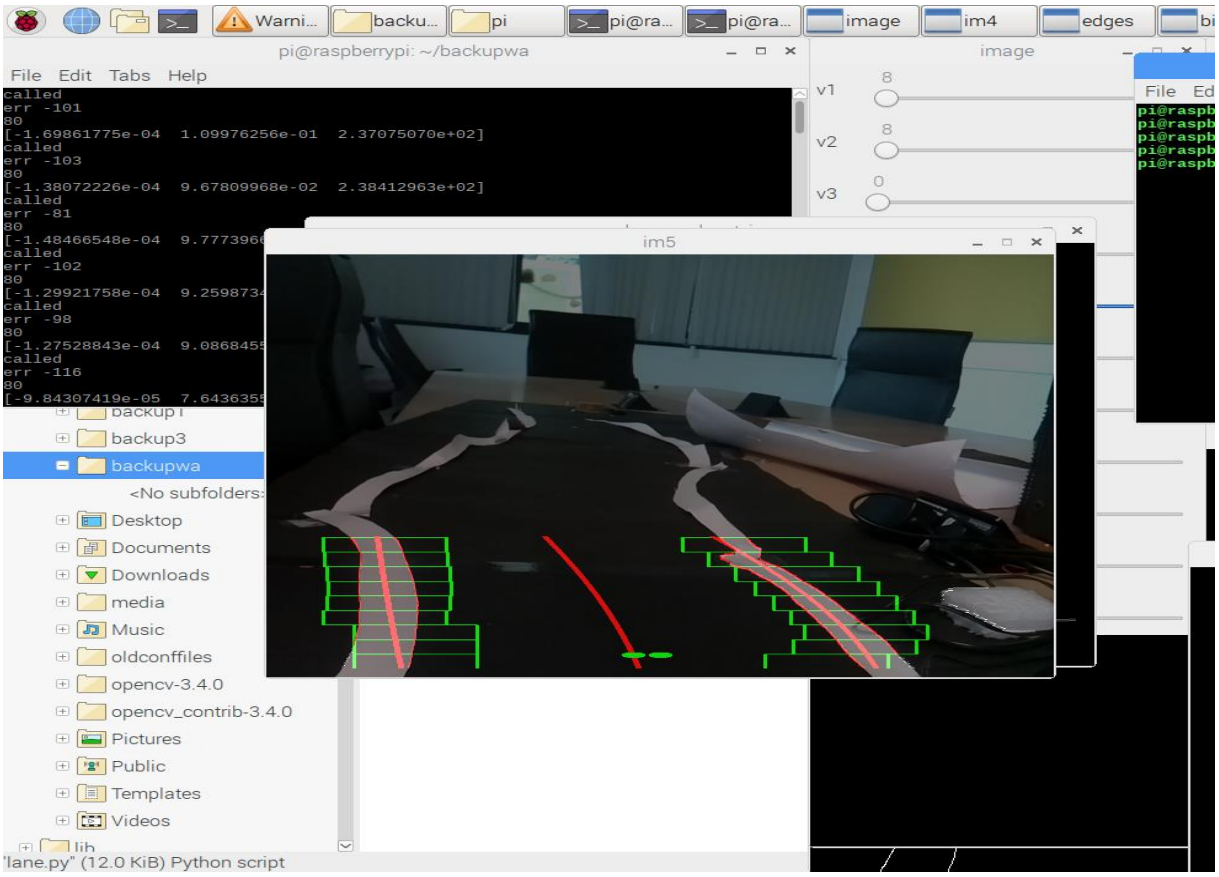
Polyfit():`numpy.polyfit`(*x*, *y*, *deg*, *rcond=None*, *full=False*, *w=None*, *cov=False*)

Least squares polynomial fit.

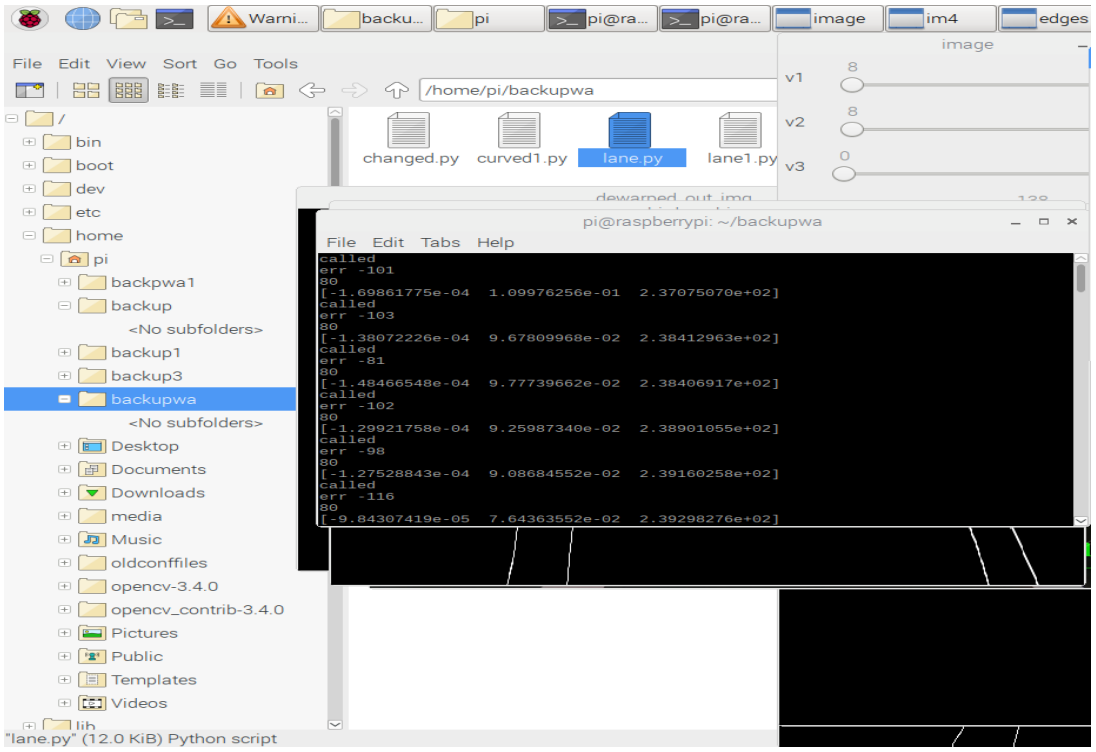
Polyval():`polyval`(*p*, *x*)

Evaluate a polynomial at specific values.

Here from the function polyfit() we get the coefficients of the polynomial equation and we have the coordinates of X that we got from the sliding window so we can get all the Y coordinates and then draw the proper curve line



Curve line drawn on the lane with sliding windows



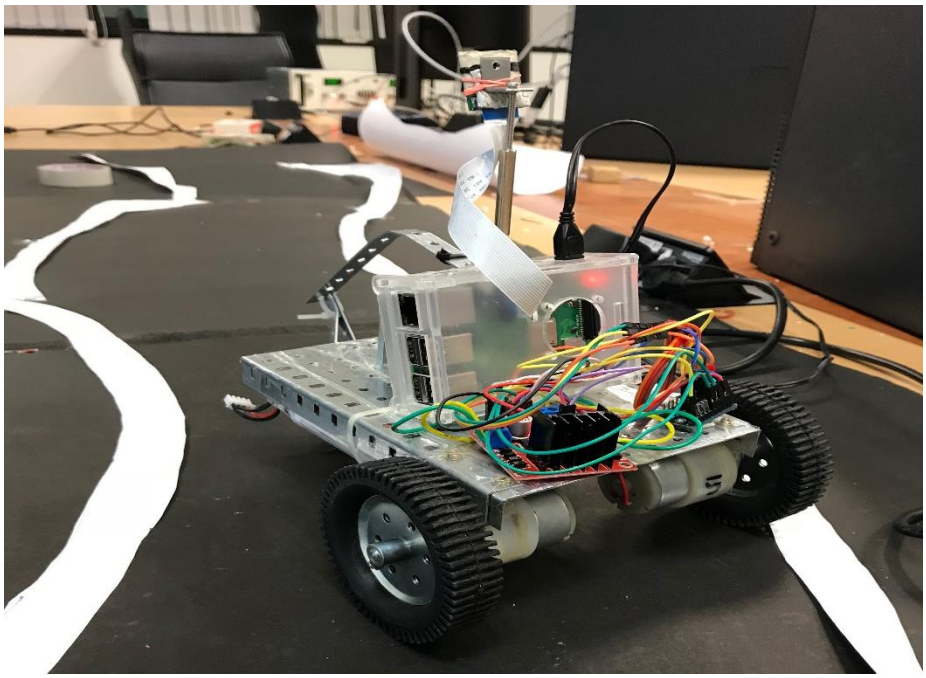
Coordinates of the y axis

Actuation:

Now we know the coordinates of the y axis of both the left and right lane so we can fine the middle of the indices that will be the main reference point that will be used for the calibration further.

So we draw a reference circle on the mid of the lane and the other point will be the Bot's position which is the mid of the camera frame.

Here we will find out the difference between both the points and that difference is calibrated with the servo motor angle which will steer the Bot



Challenges faced:

- First of all, it seems a basic thing of installing a cv2 version. Installation of it in raspberry pi can take a 8-9hours .So be sure of the memory in S D card which should be around 16gb.
- For the straight lane detection it was comparatively easy but for the curve lane the concept is mathematical

- The hardware setting is a difficulty as DC motor's speed can vary and factors can make the Bot's movement haywire.
- Raspberry pi is delicate, the base should be insulated to prevent it from getting short circuited.

Scope of improvement:

- The speed can be increased as I have done all the processing and video streaming in raspberry .As the raspberry pi processing is slower than if we stream it in the laptop.
- This Bot can only detect one lane we can improve it for the multiple lanes also.
- PID controller can be implemented to make decrease the error rate and also to increase the accuracy
- Sometimes if it recognizes only one lane then it can be made so that it can make its own path that may be done by restoring the previous middle lane coordinates.

Bibliography

- https://towardsdatascience.com/teaching-cars-to-see-advanced-lane-detection-using-computer-vision-87a01de0424f
- https://towardsdatascience.com/finding-lane-lines-on-the-road-30cf016a1165
- www.wikipedia.com