REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministry of MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département : Electronics

Rapport

Projet de Fin de Cycle (PFC) de Licence

Spécialité : Electronique Industriel

Par

- ***HADJ SAID YAHIA***
- ***CHIHANI ROSTOM***
- ***TOUMI ABDERRAHMANE***

Intitulé :

Détection de lignes par OpenCV pour une Voiture Autonome

Présenté le *: 22/06/2022*

Devant le Jury composé de :

Nom & Prénom	***Grade***	***Qualité***	***Etablissement***
SID AHMED S	***Dr.***	***Encadreur***	***Univ-BBA***
DJELLAL D	***M.***	***Président***	***Univ-BBA***
YOUSFI A	***Dr.***	***Examinateur***	***Univ-BBA***

Année Universitaire 2021/2022

Thanks

Praise be to ﷲ, Lord of the Worlds.

At the end of this work, we first want to

Express our deepest gratitude to our

supervisor:

Dr. Sid Ahmed Soumia

Who guided us throughout this work.

We thank everyone who contributed and

Facilitate the completion of this work in good

Terms.

Dedications

We dedicate this modest work

To our dear parents.

To all our families and friends.

To all those who encouraged us and

Supported.

Summary:

Computer Vision is a branch of artificial intelligence that aims to allow a machine to understand what it "sees" when it connects to one or more cameras. With the widespread use of digital images, Computer Vision techniques proved to be an indispensable tool in diverse applications such as video surveillance, video compression, medical imaging, robotics, human computer interaction. Our project focuses on studying a method for lane line detection in self-driving cars and doing the implementation in Raspberry Pi 4.

ملخص:

تعتبر الرؤية الحاسوبية أحد فروع الذكاء الاصطناعي الذي يهدف إلى السماح للآلة بفهم ما "تراه" عندما تتصل بكاميرا واحدة أو أكثر. مع الاستخدام الواسع النطاق للصور الرقمية، أثبتت تقنيات رؤية الكمبيوتر أنها أداة لا غنى عنها في تطبيقات متنوعة مثل المراقبة بالفيديو، وضغط الفيديو، والتصوير الطبي، والروبوتات، والتفاعل البشري مع الكمبيوتر. يركز مشروعنا على دراسة طريقة لاكتشاف خط الطريق في السيارات ذاتية القيادة والقيام بالتنفيذ باستعمال Raspberry Pi 4 .

Sommaire :

La vision par ordinateur est une branche de l'intelligence artificielle qui vise à permettre à une machine de comprendre ce qu'elle "voit" lorsqu'elle se connecte à une ou plusieurs caméras. Avec l'utilisation généralisée des images numériques, les techniques de vision par ordinateur se sont révélées être un outil indispensable dans diverses applications telles que la vidéosurveillance, la compression vidéo, l'imagerie médicale, la robotique, l'interaction homme-machine. Notre projet se concentre sur l'étude d'une méthode de détection de ligne de voie dans les voitures autonomes et la mise en œuvre dans Raspberry Pi 4.

TABEL OF CONTENTS:

CHPTER 02:

LIST OF FIGURES:

GENERAL INTRODUCTION:

Computer Vision is a branch of artificial intelligence (AI) that allows computers and systems to extract meaningful information from a single image or a sequence of images, it aims to automate the tasks that the visual system of a human can perform.

OpenCV (Open-Source Computer Vision) is a library of algorithms mainly aimed at real-time Computer Vision. With the advent of powerful machines, we are getting more processing power to work with. With its focus on real-time vision, OpenCV helps students and professionals to efficiently implement projects and jump-start research by providing them with a Computer Vision infrastructure that was previously available only in a few major research labs.

An autonomous car is a vehicle capable of detecting its environment and to operate without human intervention. Autonomous cars developers use artificial intelligence techniques and work on large quantities of data that include images from cameras on autonomous cars from which the car learns to identify the traffic lights, the trees, sidewalk borders, pedestrians, traffic signs and other parts of any driving environment given.

In this work, we are interested in writing and executing an algorithm using OpenCV library for line detection on Raspberry Pi 4. The lines detection algorithm intended for an autonomous car in order to keep the car on the right track using the OpenCV library.

Our project is divided into two main chapters:

CHAPTER 1:

In this project our input data are images, so in the theoretical chapter; we first dedicated a small portion to talk about images and their characteristics, then we introduced the Computer Vision domain and its application fields. lastly, we explained in a detailed manner the structure of the library OpenCV and why do we particularly need it to make the achievement of our goal easier.

CHAPTER 2:

The experimental chapter is divided into two parts; in the first one we have explained the steps we followed to detect the lane lines then is the second one we talked in more details how we implemented our program on Raspberry Pi 4.

CHAPTER 01

Introduction:

Computer Vision is a field of artificial intelligence (AI) that aims to develop techniques that help computers to capture, interpret, understand, and process the objects that are visually perceivable. This allows Computer Vision systems to react appropriately. In this research, we will use OpenCV which is a Computer Vision library but first we need to understand the field of image processing and the characteristics of digital images.

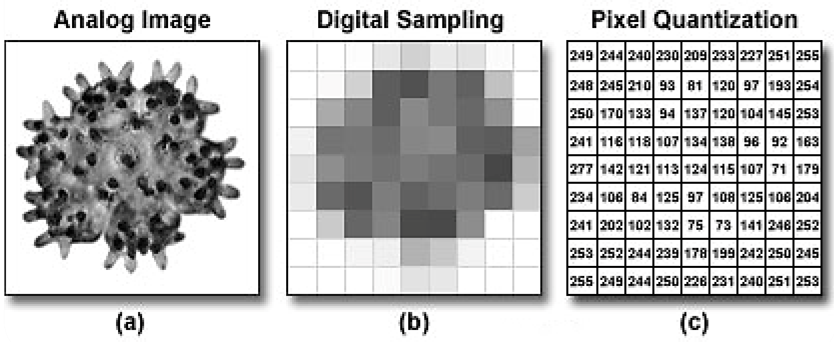
I. What is an image?

An image is a planar representation of a scene or an object located in general in a three-dimensional environment, it comes from the contact of light rays coming from objects forming the scene with a sensor (camera, scanner, X-rays, etc.). It's actually just a spatial representation of light. The image is considered as a set of points to which a physical quantity (luminance, color) is assigned.

I.1. Digital images:

A digital image is a representation of a real image as a set of numbers that can be stored and handled by a digital computer. In order to translate the image into numbers, it is divided into small areas called pixels (picture elements). For each pixel, the imaging device records a number, or a small set of numbers, that describe some property of this pixel such as its brightness (the intensity of the light) or its color. The numbers are arranged in an array of rows and columns that correspond to the vertical and horizontal positions of the pixels in the image [1].

1



249	244	240	230	209	233	227	251	255
248	245	210	93	81	120	97	193	254
250	170	133	94	137	120	104	145	253
241	116	118	107	134	138	98	92	183
277	142	121	113	124	115	107	71	179
234	106	84	125	97	108	125	106	204
241	202	102	132	75	73	141	248	252
253	252	244	239	178	199	242	250	245
255	249	244	250	228	231	240	251	253

Figure 1. Matrix representation of a digital image

[1] http://hamamatsu.magnet.fsu.edu/articles/images/digitalimagesfigure1.jpg

I.2. Image types:

Before explaining the difference between the three types of images, binary, gray and color images, we show in the figure below an example of the same image with different types:

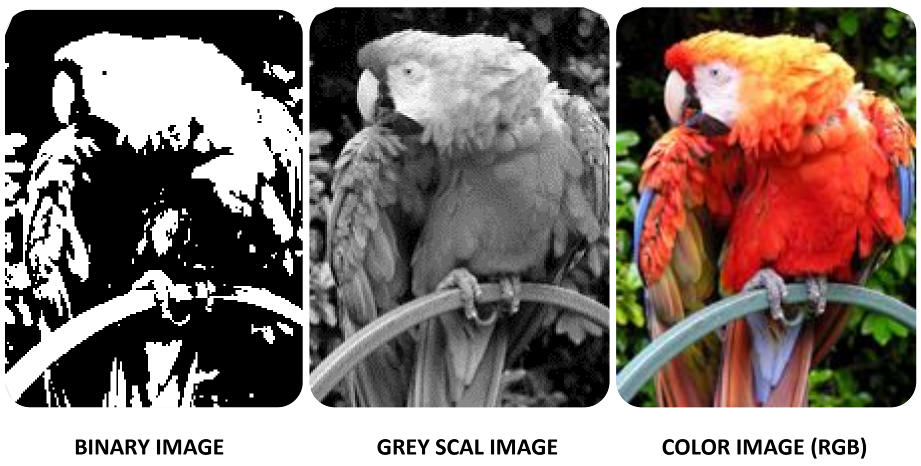


Figure 2. Three-different image type

I.2.1. Binary images:

It is the simplest type of image. The binary image consists of a 1-bit image and it takes only 1 binary digit to represent a pixel Black and White ‘0’ and ‘1’. Binary images are mostly used for general shapes or outlines.

Binary images are generated using the threshold operation. When a pixel is above the threshold value, then it is turned white ('1') and which are below the threshold value then they are turned black ('0').

[2]

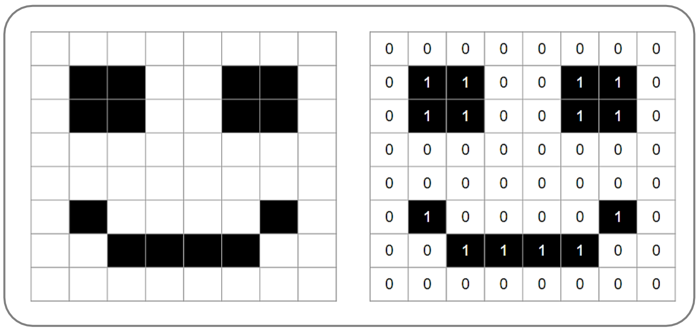


Figure 3. Binary image

I.2.2. Grayscale image:

Grayscale images are monochrome images, means they have only one color “shades of gray”.

[2] https://learnlearn.uk/binary/wp-content/uploads/sites/11/2017/01/step-1-binary-bitmap-bw.png

A normal grayscale image contains 8 bits/pixel data, which has 256 different grey levels.

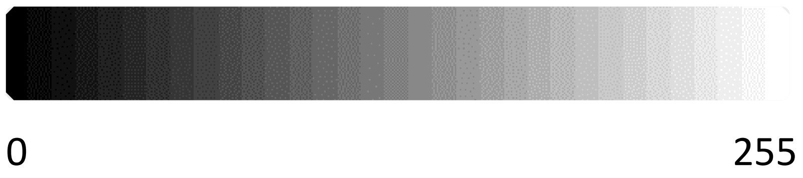
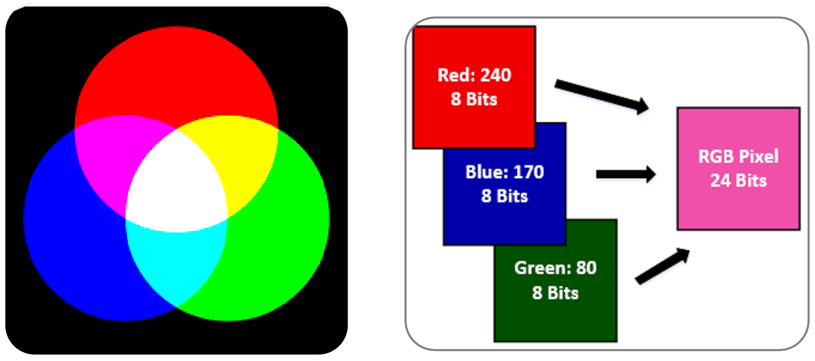


Figure 4. Grayscale [0-255].

I.2.3. Color image (RGB):

Color images are three band monochrome images in which, each band contains a different color and the actual information is stored in the digital image. The color images contain gray level information in each spectral band. The images are represented as red, green and blue (RGB images), And each color image has 24 bits/pixel means 8 bits for each of the three-color band (RGB) [2].



3

Figure 5. RGB COLOR

I.3.Main graphic formats:

I.3.1. JPEG (.jpg,.jpeg):

JPEG, which stands for Joint Photographic Experts Groups is a “lossy” format meaning that the image is compressed to make a smaller file. The compression does create a loss in quality but this loss is generally not noticeable.

I.3.2. GIF (.gif):

GIF or Graphics Interchange Format files are widely used for web graphics, because they are limited to only 256 colors, can allow for transparency, and can be animated. GIF files are typically small in size and are very portable.

[3] https://publicdomainvectors.org/photos/1304647802.png

I.3.3. PNG (.png):

PNG or Portable Network Graphics files are a lossless image format originally designed to improve upon and replace the GIFformat. PNG files are able to handle up to 16 million colors, unlike the 256 colors supported by GIF.

I.3.4. TIFF (.tif, .tiff):

TIFF or Tagged Image File Format are lossless images files meaning that they do not need to compress or lose any image quality or information (although there are options for compression), allowing for very high-quality images but also larger file sizes.

I.3.5. Bitmap (.bmp):

BMP or Bitmap Image File is a format developed by Microsoft for Windows. There is no compression or information loss with BMP files, which allows images to have very high quality, but also very large file sizes. Due to BMP being a proprietary format [3].

I.4. Characteristics of a digital image:

I.4.1. Pixel:

An image consists of a set of points called “pixel” (Picture element) thus represents the smallest constituent element of a digital image. For 3D images the "pixel" is then called a voxel, and represents a volume elementary. Examples of such images are found in medical images. The axial tomographic images are thus images constructed from several X-rays taken from different viewing angles.

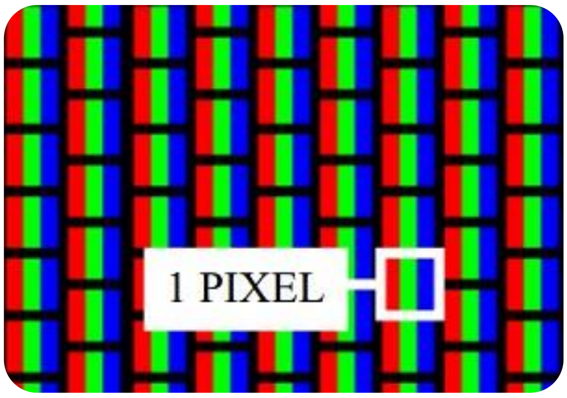


Figure 6. Representation of pixel in digital image

I.4.2. Definition:

The number of points (pixels) constituting an image is called definition; it is the number of columns of the image multiplied by its number of lines. An image with 10 columns and 11 lines will have a definition of 10 x 11, that is 110 pixels.

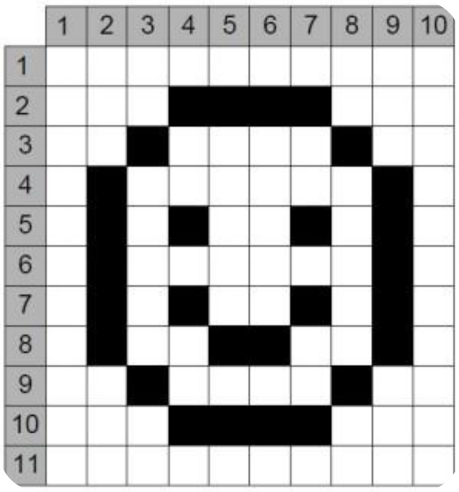
4 

Figure 7. Example of image definition

I.4.3. Dimension:

The dimension of an image corresponds to the measurements ($width\ x\ length$) of a digital image, for example 4608x3072 px. Usually, we start by giving the width. The Dimension of a graphic file is given in pixels.

I.4.4. Resolution:

The resolution of an image is the number of dots contained in a given length (in inches). It is expressed in dots per inch (DPI in French or DPI in English for Dots Per Inch).

An inch measures 2.54 cm, it is a British unit of measurement used in English-speaking countries.

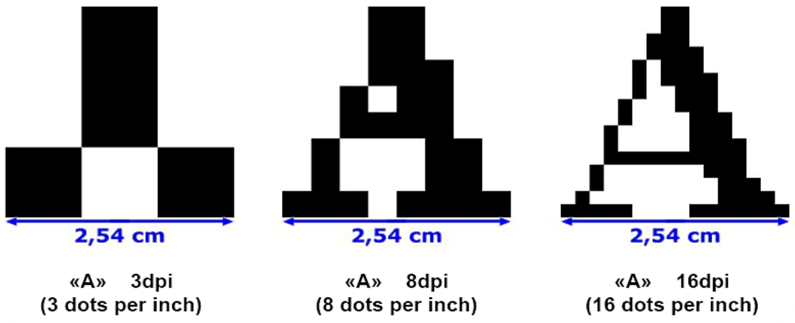


Figure 8. Example of image resolution

I.4.5. Noise:

Noise is a random variation of brightness or color information in images, and it is usually an aspect of electronic noise. It can be produced by the image sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the

[4] https://www.lossendiere.com/2016/08/31/caracteristiques-dune-image-numerique/

unavoidable shot noise of an ideal photon detector. Image noise is an undesirable by-product of image capture that obscures the desired information.

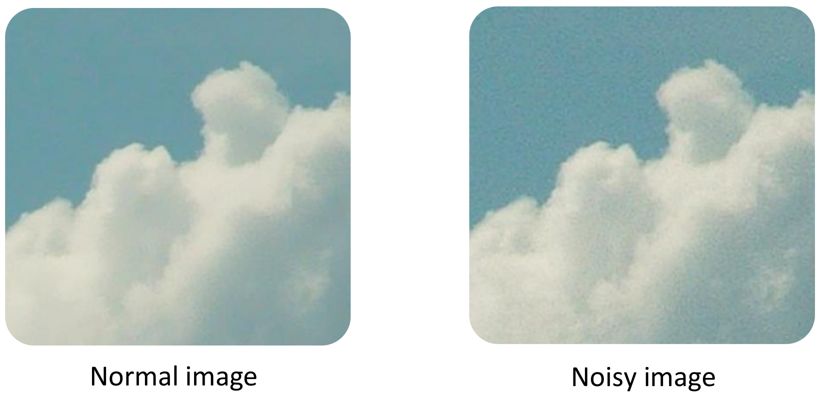


Figure 9. Example of a noisy image

I.4.6. Luminance:

Luminance is the luminous intensity projected on a given area and direction. Luminance typically describes the intensity of emitted light. In the case of our display profiling products, we measure the luminance of the cd/m2 as the unit of measure.

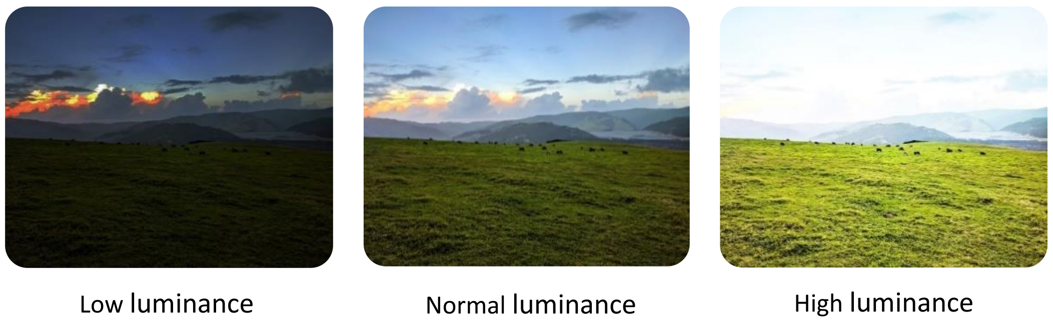


Figure 10. Example of three different luminance values of the same image

I.4.7.Contrast:

Contrast is the scale of difference between pure black and pure white. In most cases, when you take out your phone and take a selfie, you can get a medium contrast image. If you want to make the object stand out, you can use light and shadow to increase or decrease contrast.

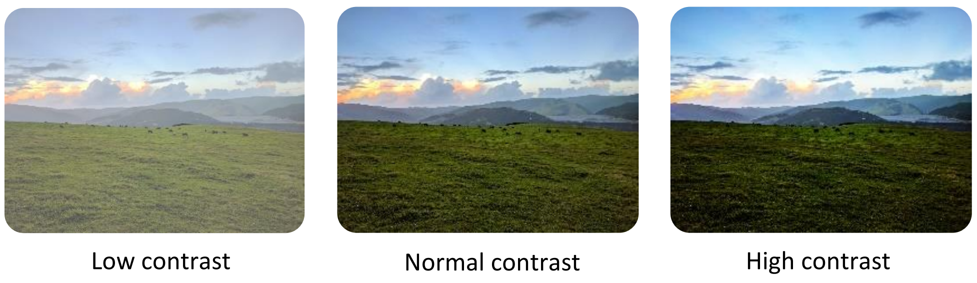


Figure 11. Example of three different contrast value of an image

I.4.8. Histogram

An image histogram is a graph of pixel intensity (on the x-axis) versus number of pixels (on the y-axis). The x-axis has all available gray levels, and the y-axis indicates the number of pixels that have a particular gray-level value.2 Multiple gray levels can be combined into groups in order to reduce the number of individual values on the x-axis.

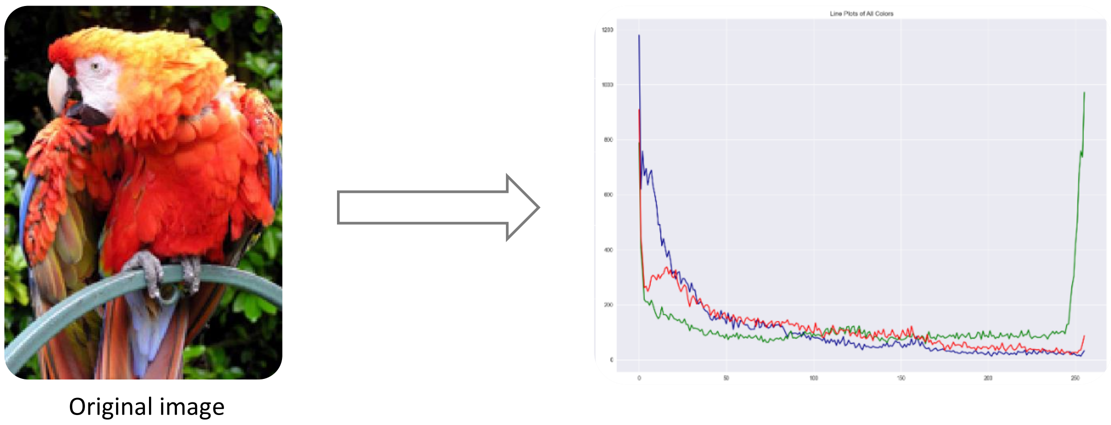


Figure 12. Example of an RGB image and its histogram

II.Computer Vision:

II.1. What is Computer Vision?

Computer Vision is a collection of algorithms that allow a computer to analyze an image and extract useful information. It is used in many applications, and it is rapidly becoming a part of everyday life.

II.2.How Computer Vision works:

It all starts with an image. The computer analyzes an image to identify lines, corners, and a broad area of colors. This process is called feature extraction, once the features are extracted, the computer can use this information for many different tasks.

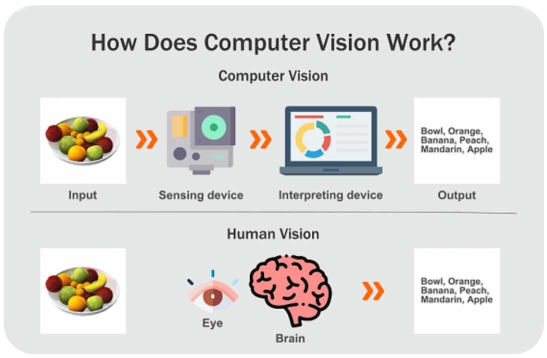

How Does Computer Vision Work?
Computer Vision
Input
Sensing device
Interpreting device
Bowl, Orange, Banana, Peach, Mandarin, Apple
Output
Human Vision
Eye
Brain
Bowl, Orange, Banana, Peach, Mandarin, Apple

Figure 13. How Computer Vision works?

II.3.Application fields of Computer Vision:

Computer Vision is a powerful feature that can be combined with many types of applications and sensing devices to support a number of practical use cases. Here are some of the different types of Computer Vision applications fields [4]:

II.3.1. Transportation:

The increasing demands of the transportation sector have propelled technological development in this industry, with Computer Vision at its center. As an example, the Self-driving cars use real-time object identification and tracking to gather information about what's happening around them and steer accordingly.

5

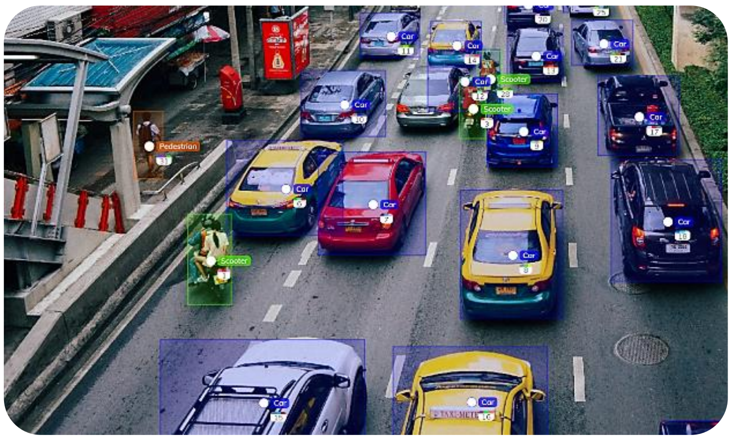


Figure 14. Example of opencv use in transportation

II.3.2. Medical Imaging:

Computer Vision has been used in various healthcare applications to assist medical professionals in making better decisions regarding the treatment of patients. Medical imaging or medical image analysis is a method that creates a visualization of particular organs and tissues to enable a more accurate diagnosis. With medical image analysis, it becomes easier for doctors and surgeons to glimpse the patient’s internal organs to identify any issues or abnormalities.

X-ray radiography, ultrasound, MRI, endoscopy, etc., are a few of the disciplines within medical imaging.

[5] https://www.v7labs.com/blog/computer-vision-applications#h1

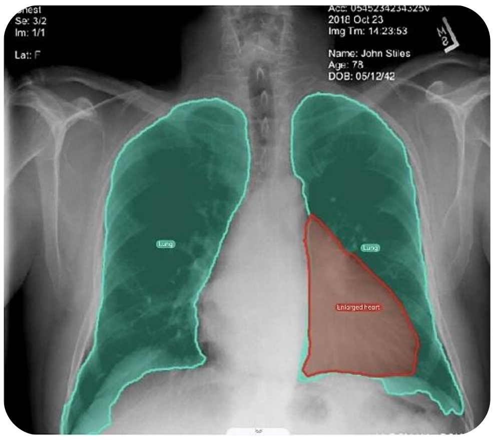


Figure 15. Image shows X-ray radiography

II.3.3. Safety / COVID-19 Pandemic Protective Measures:

As the Covid-19 pandemic hit the world in early 2020, it brought several manufacturing activities to a halt. Even as the activities resumed, several manufacturing plants made social distancing and wearing masks mandatory for workers' safety. Computer Vision applications are tremendously helpful in this case, as it makes it possible to effectively monitor the workforce to identify any violations of the Covid-19 protocol. In addition, the use of such Computer Vision models in manufacturing plants will ensure a safe working environment during and even after the pandemic.

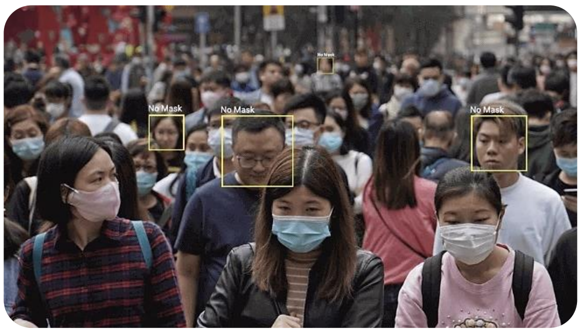


Figure 17. Image shows mask detecting

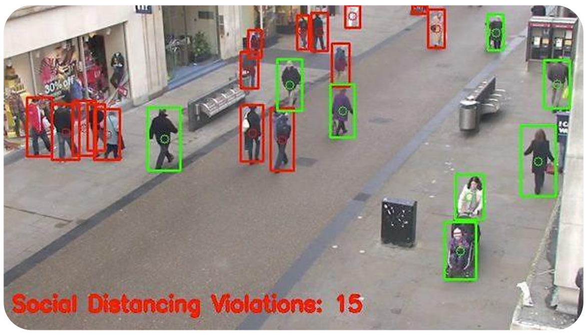


Figure 16. Image capturs social distancing violation

II.3.4. Agriculture:

The agricultural sector has witnessed a lot of contributions when it comes to Artificial Intelligence (AI) and Computer Vision in areas like plant health detection and monitoring, planting, weeding, harvesting, and advanced analysis of weather conditions.

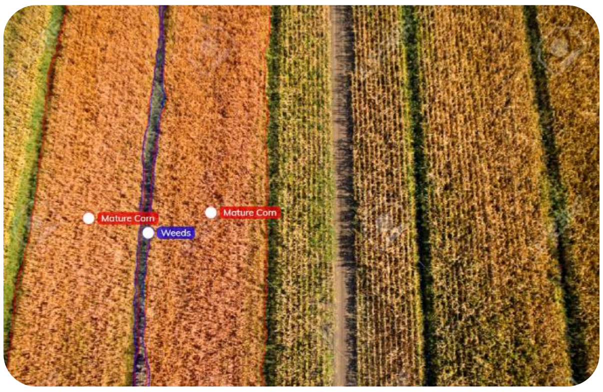


Figure 19. Crop image from drone

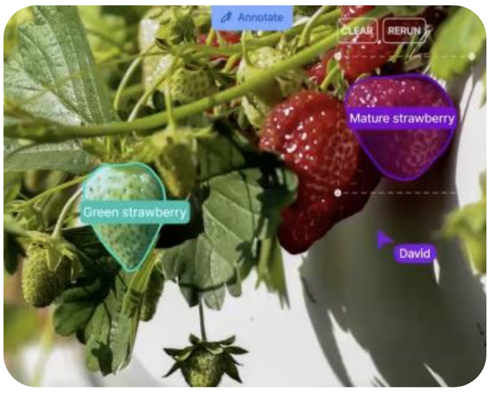


Figure 18. Agricultural crop monitoring

II.3.5. Manufacturing/Optimizing Supply Chains:

Optimizing the supply chain process is beneficial for manufacturing plants to reduce costs while enhancing customer satisfaction. Several manufacturing enterprises have turned to Computer Vision applications for tasks like warehouse management, managing inventories, and improving efficiency in the organization, For example, companies like Amazon and Walmart are working to implement drone systems to monitor warehouse inventories, real-time processing of camera streams is used to detect empty containers and optimize restocking.

6

Figure 20. Image shows optimizing supply chains

II.3.6. Security:

*Facial Rec*ognition cameras are an integral part of many smart cities. These cameras keep monitoring the city for criminal activities, can recognize moving objects, and detect wrongdoing. In addition, smart surveillance systems can send alerts when irregular activity occurs in the town. The signs are dispatched with the help of public safety agencies.

[6] https://esfam.auf.org/wp-content/uploads/2020/09/supply-chain-management-Illustration.jpg

Figure 21. Facial recognition application

II.3.7. Scientific Research:

Computer Vision can for example help in identifying invisible objects to the human eye in medical images, to prevent the diagnosis of diseases that are usually detected too late.

These were examples that show the wide uses of Computer Vision in different areas.

II.4. Presentation of the OpenCV library:

II.4.1. WHAT IS OPEN CV?

OpenCV (also known as Open-Source Computer Vision) is an open-source library for Computer Vision and Machine Learning. It has many functionalities for image processing and Computer Vision. It is a cross-platform library, and it works with many programming languages and operation systems. The library has more than 2,500 optimized algorithms for Machine Learning and Computer Vision tasks. It has a community of more than 47,000 Computer Vision professionals, and it has been downloaded more than 18 million times [5].

OpenCV

II.4.2. Principal modules of OpenCV 4.2.0:

OpenCV consists of two types of modules, main and additional modules [6]:

- **Main modules:** These modules are more or less the core modules of OpenCV and come by default with the packaged versions. They form core modules because they provide the core functionalities such as image processing tasks, filtering, transformation, and others.

- **Extra modules:** These modules do not come by default with the OpenCV distribution. These modules are related to additional Computer Vision functionalities such as text recognition.

Main Modules:

a) **Core:**

Includes all core OpenCV functionalities such as basic structures, Mat classes, and so on.

b) **Imgproc:**

Includes image-processing features such as transformations, manipulations, filtering, and so on.

c) **Imgcodecs**:

Includes functions for reading and writing images.

d) **Videoio:**

Includes functions for reading and writing videos.

e) **highgui:**

Includes functions for GUI creation to visualize results.

f) **Video:**

Includes video analysis functions such as motion detection and tracking, the Kalman filter, and the infamous CaM Shift algorithm (used for object tracking).

g) **calib3d:**

Includes calibration and 3D reconstruction functions that are used for the estimation of transformation between two images

h) **features2d:**

Includes functions for keypoint-detection and descriptor-extraction algorithms that are used in object detection and categorization algorithms.

i) **Objdetect:**

Supports object detection.

j) **dnn:**

Used for object detection and classification purposes, among others. The dnn module is relatively new in the list of main modules and has support for deep learning.

k) ml:

Includes functions for classification and regression and covers most of the Machine Learning capabilities.

l) Flann:

Supports optimized algorithms that deal with the nearest neighbor search of high-dimensional features in large data sets. Flann stands for Fast Library for approximate nearest neighbors (FLann).

m) Photo:

Includes functions for photography-related Computer Vision such as removing noise, creating HD images, and so on.

n) Stitching:

Includes functions for image stitching that further uses concepts such as rotation, estimation and image warping.

o) Shape:

Includes functions that deal with shape transformation, matching, and distance-related topics.

p) Superres:

Includes algorithms that handle resolution and enhancement.

q) Videostab:

Includes algorithms used for video stabilization.

r) Viz:

Display widgets in a 3D visualization window.

Conclusion:

In this chapter we have introduced some basics about images as they will be our data inputs, and we have given some details about the Computer Vision field to understand why we will use OpenCV which is a Computer Vision library, so we at least, understood what is Computer Vision and its applications. And to conclude we gave the structure of the OpenCV library.

CHAPTER 02

Introduction:

A self-driving car, also known as an autonomous vehicle, is a car that senses its environment and moves safely with little or no human interaction. Identifying lane lines is a very important task to keep the vehicle in the constraints of the lane. In our project we will describe a simple pipeline for lane detection and implement it on Raspberry Pi 4 that we mount it on a small robotic car, then put this latter on a track layout to see how it performs. But before we can detect lane lines in a video, we must be able to detect lane lines in a single image. Once we can do that, detecting lane lines in a video is simply repeating the same steps for all frames in a video. To do this in an easier way, we will use OpenCV-Python open-source library.

PART 1:

III. Lane line detecting steps:

In order to achieve our goal, we will follow these steps:

1. Load an Image.
2. Pre-processing of image:
   2.1. Convert to grayscale.
   2.2. Gaussian Blur.
3. Canny Edge Detection.
4. Region of interest.
5. Hough Transform.
6. Add the extrapolated lines to the input image.
7. Combine Line Segments into Two Lane Lines.
8. Steering angle.
9. Add pipeline to video.

Next, we explain how we implement each step using OpenCV.

III.1. Load an Image:

To read an image in OpenCV we use *imread* function:

Syntax: cv2.imread (“Path”, Flag).

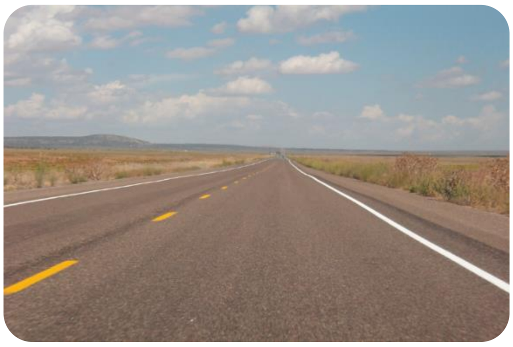


Figure 22. Example of an input image

III.2. Pre-processing of an image:

III.2.1.Convert original image to grayscale:

We take as an example the image below and apply all the detection steps on it. Our goal is to find the yellow and white lanes. To do so, as a first step we need to convert the original image to a grayscale one. The grayscale-weighted average is represented by the following equation:

Grayscale = 0.299 * R + 0.587 * G + 0.114 * B

Where R, G and B are integers representing red (R), green (G) and blue (B), with values in the range from 0 to 255 [7].

Examples:

🟥	Pure Red (255,0,0)	⬛	Equivalent Gray (76, 76, 76)
🟩	Pure Green (0, 255, 0)	▨	Equivalent Gray (150, 150, 150)
🟦	Pure Blue (0, 0, 255)	⬛	Equivalent Gray (29, 29, 29)

Figure 23. Converting RGB colors to grayscale

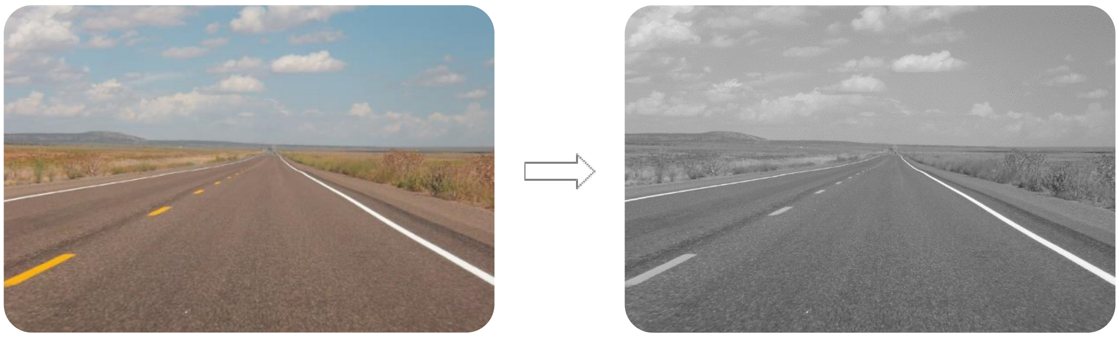


Figure 24. Converting to a grayscale image

III.2.2.Apply a slight Gaussian Blur:

In real systems the captured images are noisy images so it is necessary to filter them before detecting the edges in order to avoid detecting fake contours.

Gaussian Blur (Gaussian smoothing) is a pre-processing step used to reduce the noise from the image. This filter works by taking a pixel and calculating a value (similar to the mean, but with more bias in the middle). The filter is constructed based on normal distribution, which is shaped like a bell curve. The idea is that pixels closer to the center pixel have a larger weight than those further away [8].

To make it clearer, below an example of a gaussian kernel applied on a small matrix.

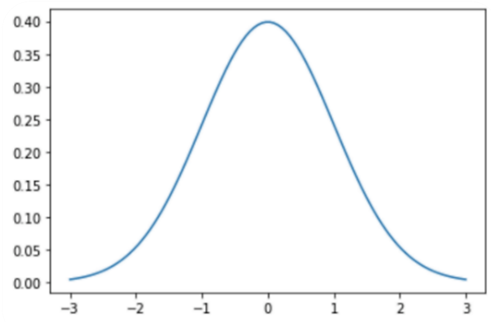
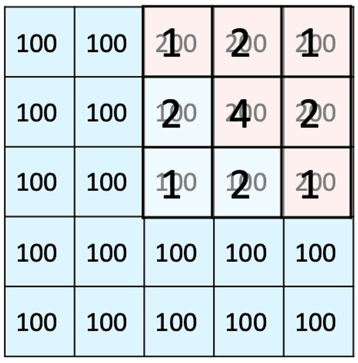


Figure 25. Gaussian kernel

1	2	1
2	4	2
1	2	1

Figure 26. Sample Gaussian filter

We apply the filter in Figure 26 on the matrix below.



100	100	1 200	2 200	1 200
100	100	2 100	4 200	2 200
100	100	1 100	2 100	1 200
100	100	100	100	100
100	100	100	100	100

To, to modify the $pixel(2{,}4)$:

$$4\ x\ 200\ +\ 2\ x\ (200\ +\ 200\ +\ 100\ +\ 100)\ +\ 1\ x\ (200\ +\ 200\ +\ 200\ +\ 100)\ =\mathbf{2700}.$$

$$2700/\ (4+\ 2\ x\ 4\ +\ 1\ x\ 4)\ =\ \mathbf{168.75}.$$

After applying the filter on this segment, the color will be between blue and pink but more on the pink side. This will likely create a gradient effect and smooth harsh edges.

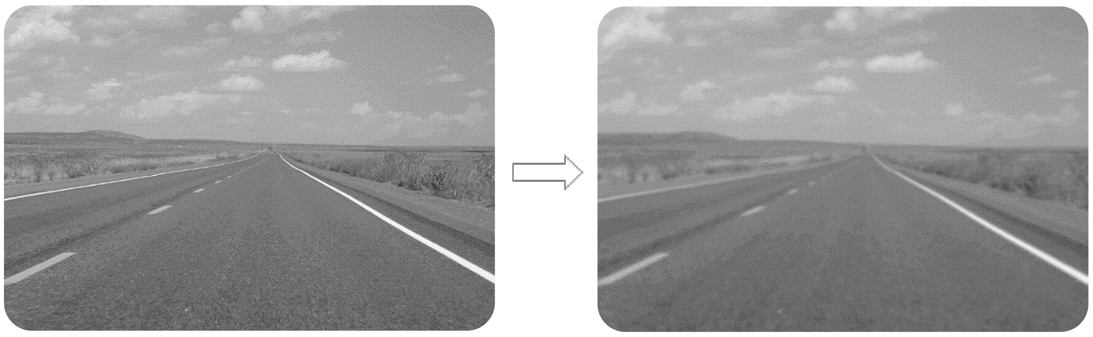


Figure 27. Adding Gaussian blur to a grayscale image

To do this in OpenCV we simply apply this function: $cv2.GaussianBlur()$.

III.3. Canny Edge Detection

The complicated conditions of road make the correct edge detection of lane markings become very challenging. In order to get an ideal edge of lane markings in a road image, we apply the Canny detector which is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images [9].

III.3.1.Gradient Calculation

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. The gradients can be determined by using Sobel filters for both directions (horizontal and vertical) where A is the image. An edge occurs when the color of an image changes, hence the intensity of the pixel changes as well.

$$Gx = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \boldsymbol{A} \quad ; \quad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \boldsymbol{A}$$

Then, we calculate the magnitude and angle of the directional gradients:

Magnitude: $|G| = \sqrt{Gx^2 + Gy^2}$

Angle: $\theta = \tan^{-1}(Gy/Gx)$

III.3.2.Non-maximum suppression:

Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges by scanning the entire image to get rid of pixels that might not be part of an edge. Non-maximum suppression works by finding pixels that are local maximum in the direction of the gradient (gradient direction is perpendicular to edges).

For example, in Figure 28 we have three pixels that are next to each other: pixels a, b, and c. Pixel b is larger in intensity than both a and c where pixels a and c are in the gradient direction of b. Therefore, pixel b is marked as an edge. Otherwise, if pixel b was not a local maximum, it would be set to 0 (i.e., black), meaning it would not be an edge pixel.

[7]

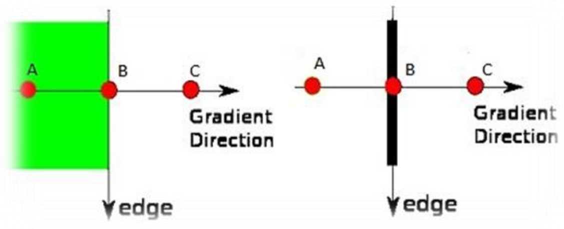


Figure 28. Example shows the principle of the non-maximum

[7] https://machinelearningknowledge.ai/wp-content/uploads/2020/06/Canny-Edge-Detection-Non-Max-Suppression.jpg

III.3.3.Thresholding:

Non-maximum suppression is not perfect because some edges might actually be noise and not real edges. To solve this, Canny Edge Detector goes one-step further and applies thresholding to remove the weakest edges and keep the strongest ones. Edge pixels that are borderline weak or strong are only considered strong if they are connected to strong edge pixels.

Thresholding sets two thresholds, a high and a low threshold. In this algorithm, we normalized all the values such as that they will only range from 0 to 1. Pixels with a higher value are most likely to be edges. For example, you might choose the high threshold to be 0.7, this means that all pixels with a value larger than 0.7 will be a strong edge. You might also choose a low threshold of 0.3, this means that all pixels less than it are not an edge and you would set it to 0. The values in between 0.3 and 0.7 would be weak edges.

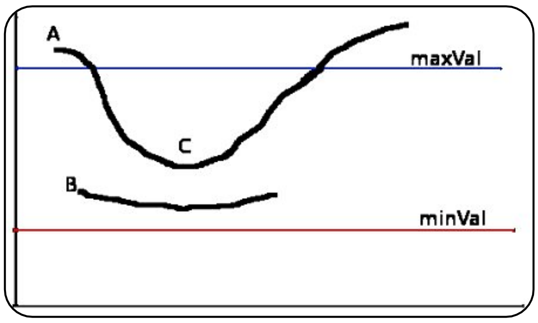


Figure 29. The double threshold step

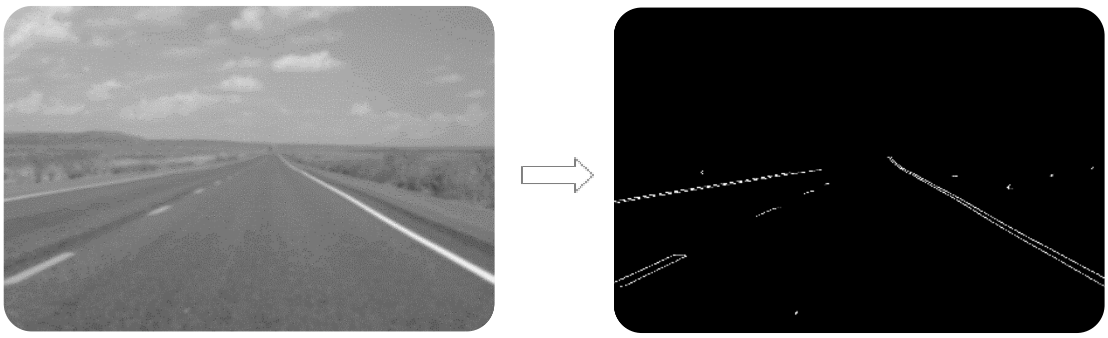


Figure 30. The original image and its output after applying the Canny filter

To do this in OpenCV we simply apply this function: $cv2.Canny()$.

III.4. Define Region of Interest:

The region of interest for the car’s camera is only the two lanes immediately in its field of view and not anything extraneous. We can filter out the extraneous pixels by making a polygon region of interest and removing all other pixels that are not in the polygon.

When the Mask object is added to the image area, only the non-zero area is visible, and all the pixel values in the Mask that overlap with the image will be invisible. That is to say, the shape and size of the Mask area directly determines what you see. The size and shape of the final image.

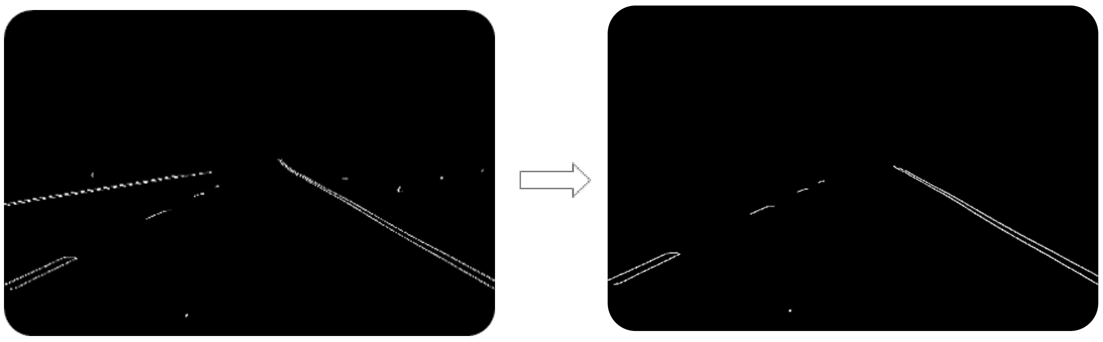


Figure 31. Applying a mask on the Canny output image

III.5. Hough Transform:

Now that we have detected edges in the region of interest, we want to identify lines, which indicate lane lines. This is where the Hough transform comes in handy. Before explaining how Hough transform works let us remind you that a straight line can be represented by two parameters.

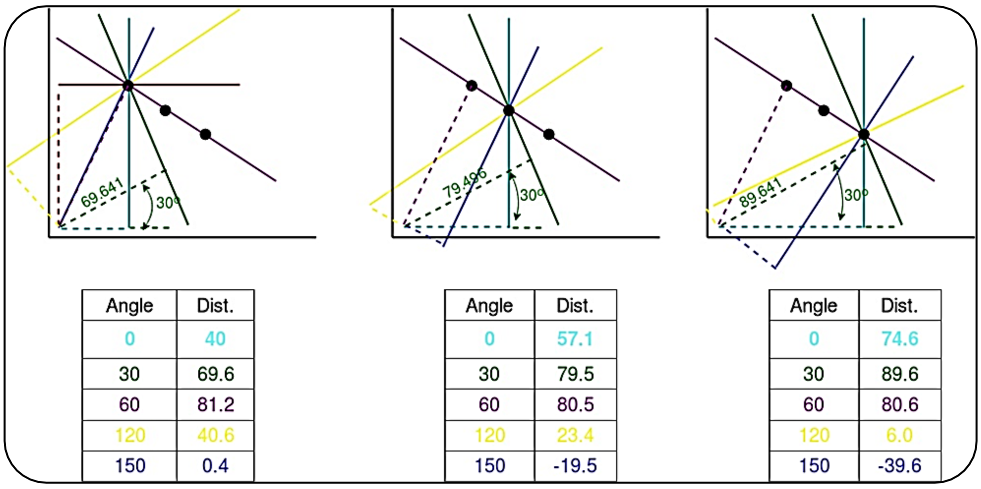
1. The line is then described as: $y = a\,x + b$, (a, b) correspond to slope and intercept.
2. or as $r = xcos\theta + ysin\theta$, r, is the shortest distance from the origin to the line (approaching the line perpendicularly). The second, θ, is the angle between x-axis and the distance line.

III.5.1. How does Hough Line method work

First, it creates a 2D array or accumulator (to hold values of two parameters) and it is set to zero initially. Let rows denote the r and columns denote the (θ) theta. Then, the size of the array depends on the accuracy you need. If you need the accuracy of the angles to be 1 degree, you then need 180 columns. For “r”, if you need the accuracy to be one pixel you then you take the maximum distance possible which is the diagonal length of the image. The example below explains better the idea behind it [10].

Consider an 100×100 image with a horizontal line at the middle. Take the first point of the line. You know its (x, y) values. Now in the line equation, put the values $\theta(theta) = 0,1,2, \dots, 180$ and check the r you get. For every $(r, 0)$ pair (r is measured in pixels and 0 is measured in radians) you increment value by one in the accumulator in its corresponding $(r, 0)$ cells. So now in accumulator, the cell $(50, 90) = 1$ along with some other cells. Now take the second point on the line. Do the same as above. Increment the values in the cells corresponding to $(r, 0)$ you got. This time, the cell $(50, 90) = 2$. We are actually voting the $(r, 0)$ values. You continue this process for every point on the line. At each point, the cell $(50, 90)$ will be incremented or voted up, while other cells may or may not be voted up. This way, at the end, the cell $(50, 90)$ will have maximum votes. Therefore, if you search the accumulator for maximum votes, you get the value $(50, 90)$ which says, there is a line in this image at distance 50 from origin and at angle 90 degrees.

To do this in OpenCV we simply apply this function: $cv2.HoughLines()$.



Angle	Dist.
0	40
30	69.6
60	81.2
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
120	23.4
150	-19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
120	6.0
150	-39.6

Figure 32. Example shows how Hough Line method works

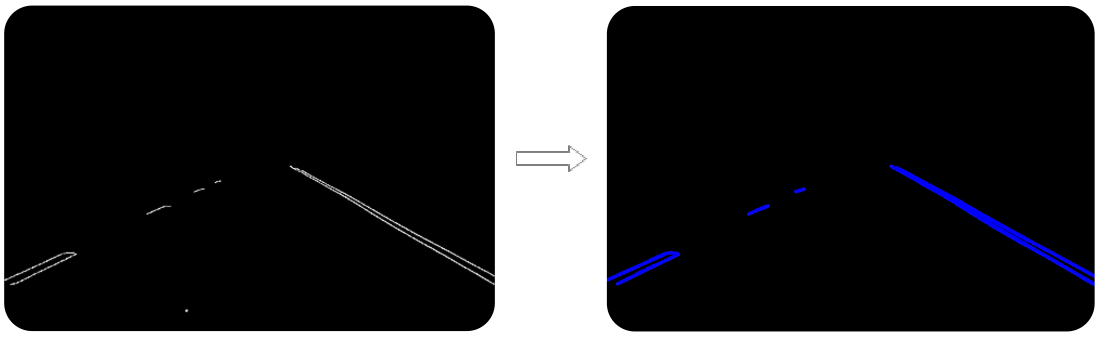


Figure 33. Applying Hough transform on masked image

III.6. Add the extrapolated lines to the input image

We then overlay the extrapolated lines to the input image. We do this by adding a weight value to the original image based on the detected lane line coordinates [11].

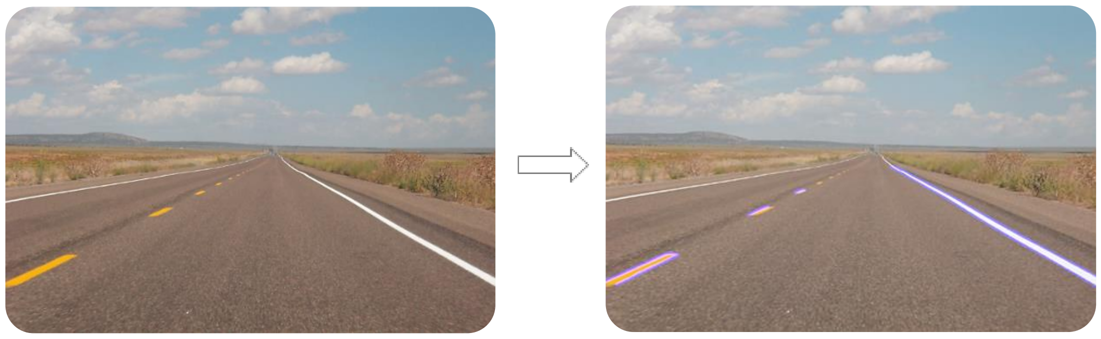


Figure 34. Showing the extrapolated lines on the input image

III.7. Combine Line Segments into Two Lane Lines:

Now that we have many small line segments with their endpoint coordinates (x1, y1) and (x2, y2), how do we combine them into just the two lines that we really care about, namely the left and right lane lines? One way is to classify these line segments by their slopes. We can see from the picture above that all line segments belong to the left lane line should be upward sloping and on the left side of the screen, whereas all line segments belong to the right lane line should be downward sloping and be on the right side of the screen. Once the line segments are classified into two groups, we just take the average of the slopes and intercepts of the line segments to get the slopes and intercepts of left and right lane lines.

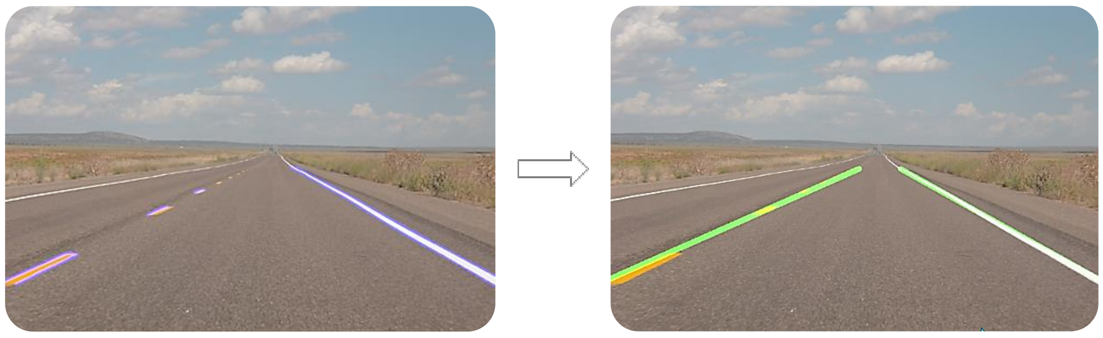


Figure 35. Combining line segments into two lane lines

III.8. Steering angle:

Now that we have the coordinates of the lane lines, we need to steer the car so that it will stay within the lane lines, even better, we should try to keep it in the middle of the lane. We need to compute the steering angle of the car, given the detected lane lines. In the Figure below the red line in the middle represents the steering angle of the car.

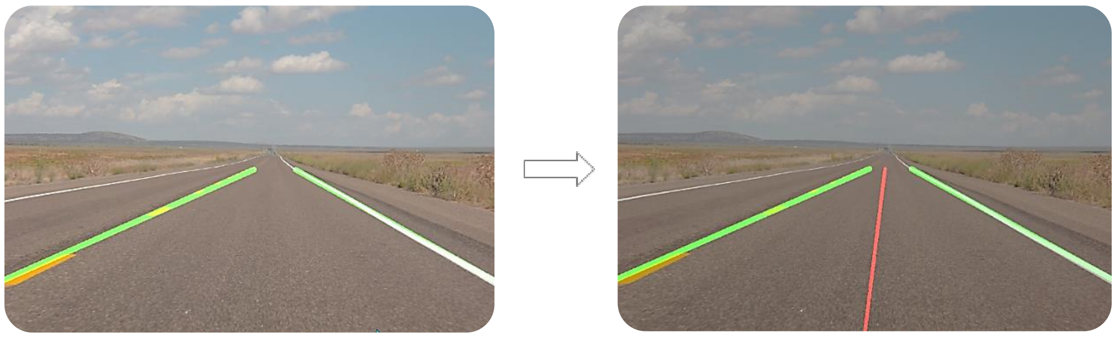


Figure 36. Representing the steering angle of the car

III.9. Add pipeline to video:

Once we have designed and implemented the entire car lane detection pipeline, we apply the transformation frame by frame. We need a video of car driving on lane lines to do this.

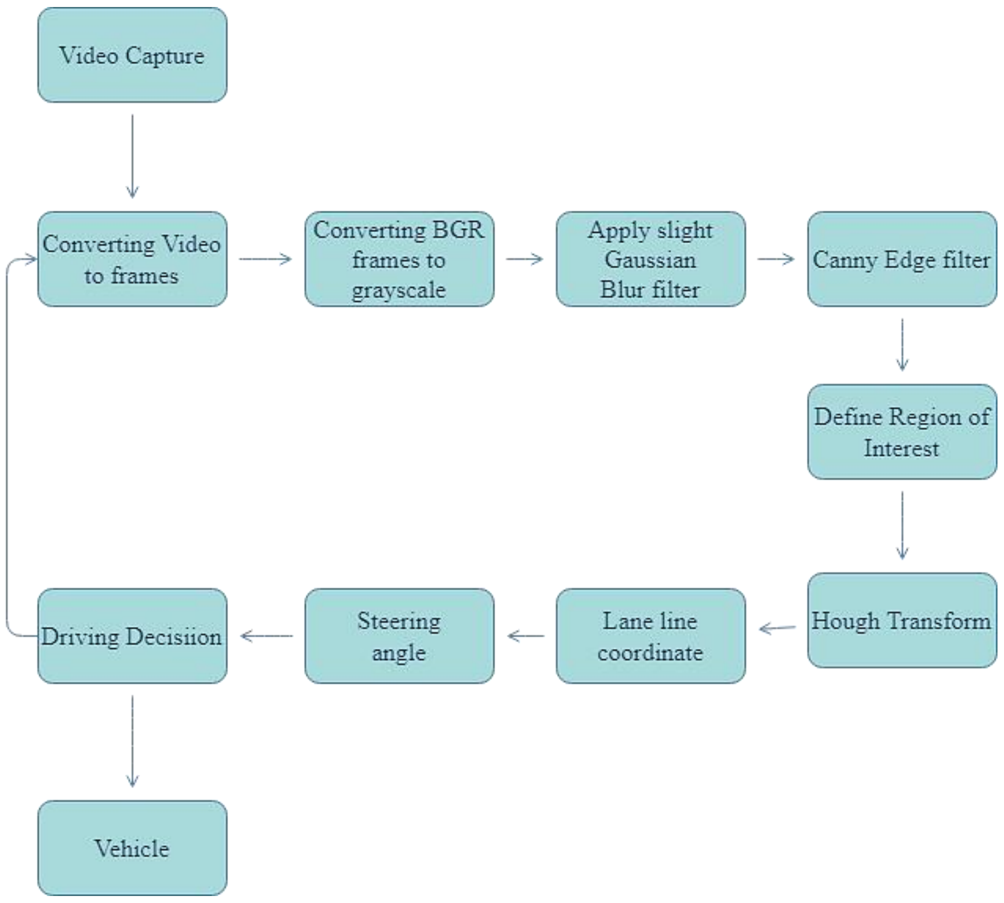
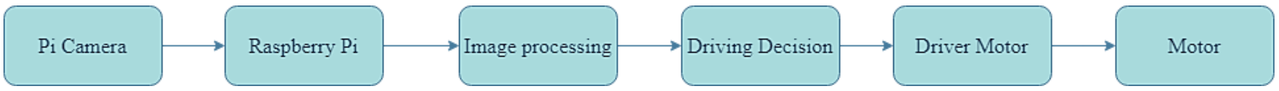

Video Capture
Converting Video to frames
Converting BGR frames to grayscale
Apply slight Gaussian Blur filter
Canny Edge filter
Define Region of Interest
Hough Transform
Lane line coordinate
Steering angle
Driving Decisiion
Vehicle

Figure 37. Lane Line Detection Algorithm

PART 2:

IV. Environment for Development:

We will describe all hardware and software we used in developing this project.



IV.1. HARDWARE :

IV.1.1. Car components:

- Raspberry Pi 4 B+.
- Raspberry Pi Camera Module (640*480).
- Power Banks (5v/3A); to power raspberry pi.
- 2×(DC Geared Motor 100 RPM).
- L298 Motor Driver.
- 7 ~ 12V Battery; to power DC motor.
- Wheels.
- Jumper wires.

IV.1.2. Raspberry Pi 4 B+:

A. Difference between Raspberry Pi and Arduino:

This table resume the main differences between Raspberry Pi and Arduino:

Basis	Arduino	Raspberry Pi
License	Arduino is an open-source project. Both its software and hardware design are open source.	Both hardware and software of Raspberry Pi are closed source.
Control Unit	From Atmega Family	From ARM Family
Clock Frequency	16 MHz (Arduino UNO)	Up to 1.5 GHz in Raspberry Pi 4 B
RAM	Requires less RAM (2kB)	Requires large RAM (more than 1 GB)
CPU Architecture	8-bit	64-bit
Logic level	Arduino’s logic level is 5V.	Raspberry Pi’s logic level is 3V.
Power Consumption	Consumes about 200 MW of power	Consumes about 700 MW of power
Based on	Arduino is a Microcontroller	Raspberry Pi is based on a microprocessor
Hardware Structure	Simple hardware structure	Complex hardware Structure
Software	Arduino boards are programmable using C/C++ languages.	Raspberry Pi supports its own Linux-based operating system Raspberry Pi OS. You can also install the OS you like.

Internet	Arduino does not have internet support. You need additional modules or shields to connect it to the internet.	Raspberry Pi has a built-in Ethernet port and Wi-Fi support.
Cost	Arduino boards are cheaper.	Raspberry Pi boards are expensive.
How they handle power drop	Arduino devices begin executing code when they are turned on. Therefore, when power is turned off, abruptly, you will not end up with a corrupt operating system or errors. The code will simply start again when plugged in.	Raspberry Pi requires the same care as a PC. You have to shut the operating system down properly.
Current drive strength	Higher current drive strength	Lower current drive strength
Capability	Arduino is generally used to perform single (and simple) tasks repeatedly.	Raspberry Pi can perform multiple tasks simultaneously.
Wireless connectivity	Arduino does not support Bluetooth or Wi-Fi.	Raspberry Pi supports Bluetooth and Wi-Fi.
Applications	Traffic light countdown timer, Parking lot counter, Weighing machines, etc.	Robot controller, Game servers, Stop motion cameras, etc.

In our project, we will control the car, based on the video captured by the camera, and process it using the OpenCV library, so the best option for this project is Raspberry Pi.

B. Characteristics of the used Raspberry Pi:

- Processor: Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz.
- Memory: 2GB.
- GPIO: Standard 40-pin GPIO header.
- Hard disk: 64 GB Micro SD card slot for loading operating system and data storage.

IV.1.3.Raspberry Pi Camera Module:

The camera is the eye of the car. The camera is interfaced with the Raspberry Pi and can continuously capture the images that represent the track layout.

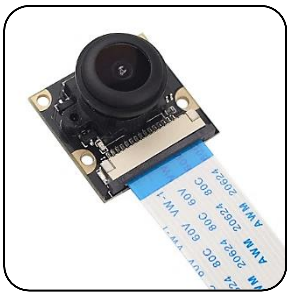


Figure 38. Pi Camera Module

IV.1.4.L298 Motor Driver:

The differential drive is a two-wheeled drive system with independent actuators for each wheel. The name refers to the fact that the motion vector of the robot is the sum of the independent wheel motions. The drive wheels are usually placed on each side of the robot and towards the front. Two motors at both sides are connected in parallel, so motor controller consider it as single unit. The Difference between voltages supplied to motors at both sides makes the car turn to a certain radius either to the right or to the left.

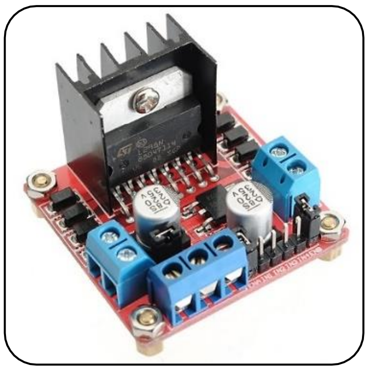


Figure 39. L298 Motor Driver

IV.1.5.Controlling Car Driving:

There are following control pins on L298N:

- Speed Control: ENA, ENB.
- Direction Control: IN1, IN2, IN3, IN4.

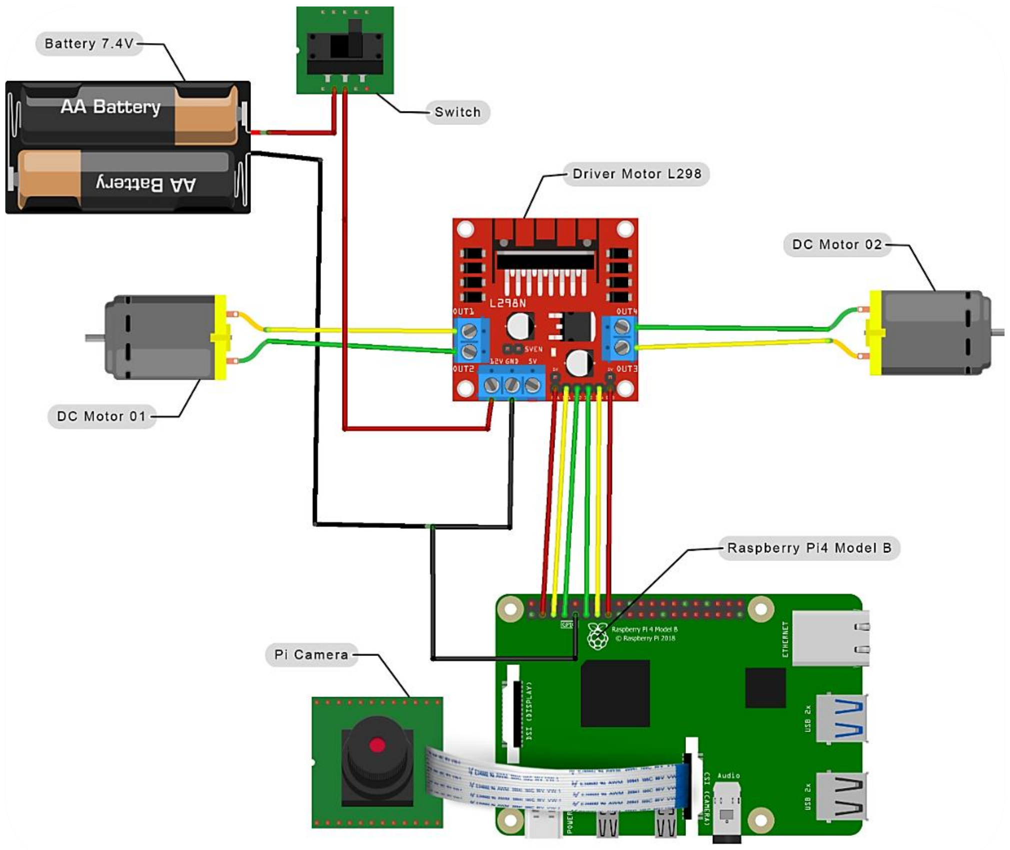


Figure 40. Electronic circuits

IV.2. SOFTWARE:

IV.2.1. Installing Raspbian on Raspberry Pi:

Step (1): First, we download and install Raspberry Pi Imager to a computer with an SD card reader. Then we put the SD card that we will use with our Raspberry Pi into the reader and run Raspberry **Pi Imager**.

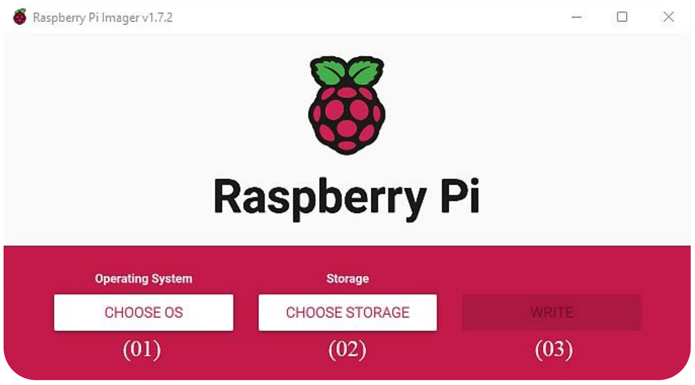


Figure 41. Raspberry Pi Imager

Step (2): After we run the Raspberry imager we choose the operating system, the SD card, and finally write and verify the SD card.

Step (03): We insert the microSD card we have just flashed with the Raspberry Pi imager to the microSD card slot of our Raspberry Pi 4. Then, we connect a USB keyboard, a USB mouse, and a micro-HDMI cable of our monitor to our Raspberry Pi 4.

Finally, we connect the USB Type-C power cable to our Raspberry Pi 4 and power it on.

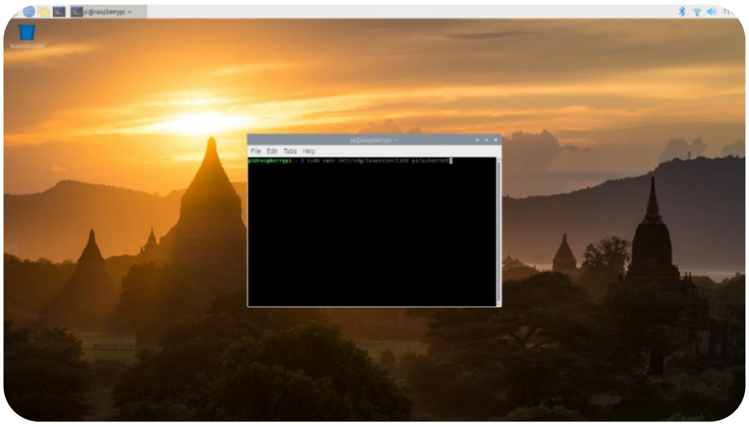


Figure 42. Raspberry Pi desktop with terminal

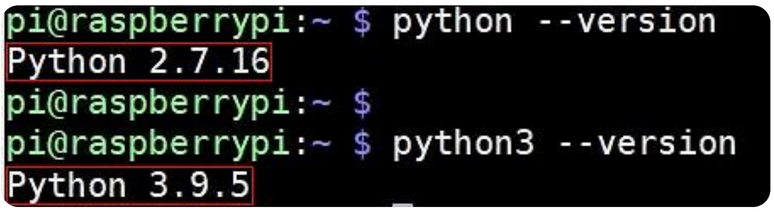
IV.2.2. Programing language:

A. Python:

In our project we used Python as it is a widely used general-purpose, high-level programming language. Its syntax allows us to express concepts in fewer lines of code if we compare it with other languages like C, C++or java.

B.Checking Python Version on Raspberry Pi:

To check python version on raspberry pi, we type these commands on the terminal:



```
pi@raspberrypi:~ $ python --version
Python 2.7.16
pi@raspberrypi:~ $
pi@raspberrypi:~ $ python3 --version
Python 3.9.5
```

Figure 43. Python version

Note: we have installed the latest version of **Python 3.9.5.**

IV.2.3. Libraries:

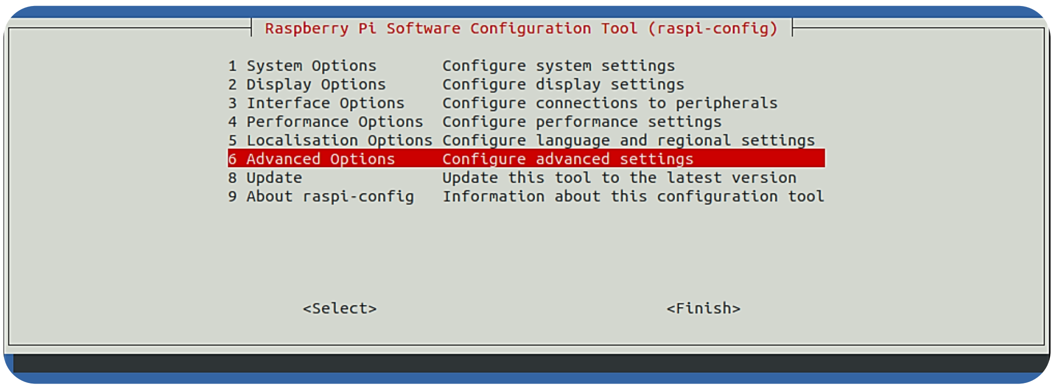
A. Install OpenCV on Raspberry Pi:

To install OpenCV on Raspberry Pi, we follow these steps:

Step (01): Expand the File system, by typing this command on Raspbian terminal:

```
sudo raspi-config
```

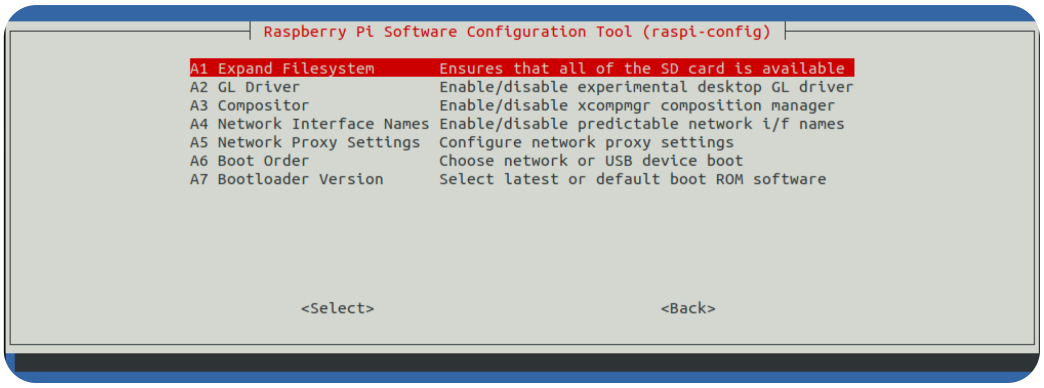
This will open the "Raspberry PI configuration tool" window. Then we Select *"Advanced Options"*, and hit Enter (Figure 44).



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options       Configure system settings
2 Display Options      Configure display settings
3 Interface Options    Configure connections to peripherals
4 Performance Options  Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

          <Select>                          <Finish>
```

Figure 44. Raspberry PI configuration tool window

On the new window appears, we select "*A1 Expand Filesystem*" and hit Enter (Figure 45).



```
Raspberry Pi Software Configuration Tool (raspi-config)

A1 Expand Filesystem         Ensures that all of the SD card is available
A2 GL Driver                 Enable/disable experimental desktop GL driver
A3 Compositor                Enable/disable xcompmgr composition manager
A4 Network Interface Names   Enable/disable predictable network i/f names
A5 Network Proxy Settings    Configure network proxy settings
A6 Boot Order                Choose network or USB device boot
A7 Bootloader Version        Select latest or default boot ROM software

            <Select>                        <Back>
```

Figure 45. Raspberry PI configuration tool window 2

Step (02): Update the system, by executing the commands below on Raspbian terminal:

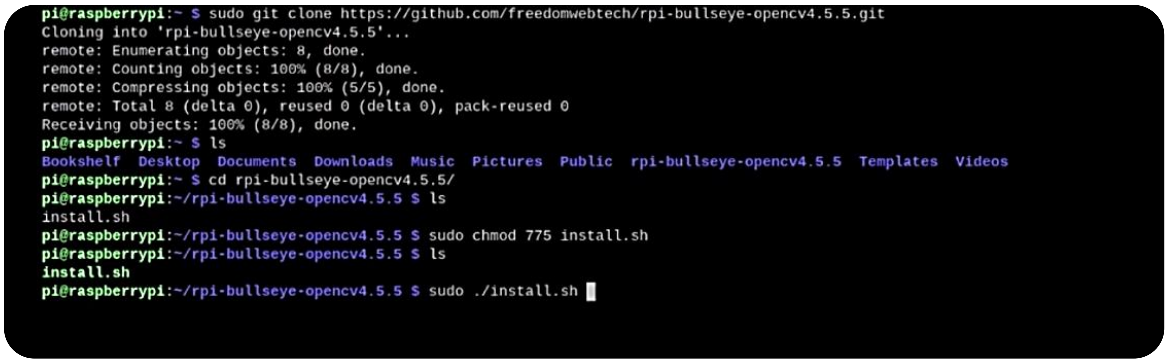
```
sudo apt update

sudo apt upgrade
```

Step (03): We go to GitHub repositories and copy the code (Figure 46). Then we paste the code on Terminal with this command:

```
sudo git clone https://github.com/freedomwebtech/rpi-bullseye-opencv4.5.5.git
```

After that, we write these following commands below.

```
Ls

cd rpi-bullseye-opencv4.5.5/

Ls

sudo chmod 775 install.sh

Ls

Sudo ./install.sh
```



```
pi@raspberrypi:~ $ sudo git clone https://github.com/freedomwebtech/rpi-bullseye-opencv4.5.5.git
Cloning into 'rpi-bullseye-opencv4.5.5'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
pi@raspberrypi:~ $ ls
Bookshelf  Desktop  Documents  Downloads  Music  Pictures  Public  rpi-bullseye-opencv4.5.5  Templates  Videos
pi@raspberrypi:~ $ cd rpi-bullseye-opencv4.5.5/
pi@raspberrypi:~/rpi-bullseye-opencv4.5.5 $ ls
install.sh
pi@raspberrypi:~/rpi-bullseye-opencv4.5.5 $ sudo chmod 775 install.sh
pi@raspberrypi:~/rpi-bullseye-opencv4.5.5 $ ls
install.sh
pi@raspberrypi:~/rpi-bullseye-opencv4.5.5 $ sudo ./install.sh
```

Figure 46. How to paste the code on terminal

Step (04): To check OpenCV Version we type these commends:

```
python3

>>> import cv2

>>> cv2.__version__
```

B. Install Python package on Thonny IDE (editor):

To install Python package on Thonny IDE, we must follow these steps:

- First, we open Thonny IDE, then we select "*Manage Packages*" from the tool's menu (Figure 47).

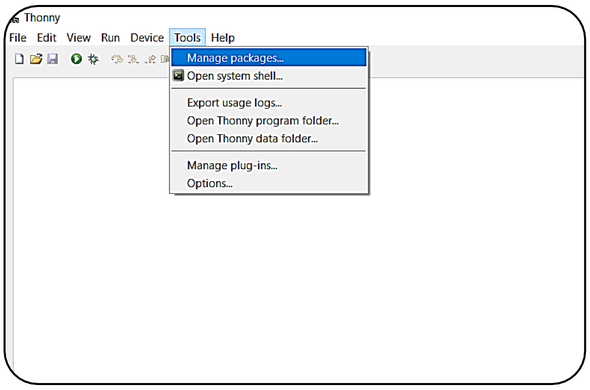
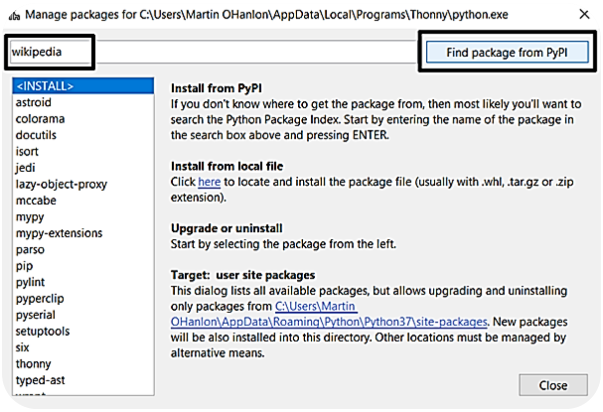


Figure 47. Install Python package on Thonny

- Enter the name of the package we want to install. Click on *"Find package from PyPI"* (Figure 48).


Manage packages for C:\Users\Martin OHanlon\AppData\Local\Programs\Thonny\python.exe

wikipedia

Find package from PyPI

<INSTALL>
astroid
colorama
docutils
isort
jedi
lazy-object-proxy
mccabe
mypy
mypy-extensions
parso
pip
pylint
pyperclip
pyserial
setuptools
six
thonny
typed-ast

Install from PyPI
If you don't know where to get the package from, then most likely you'll want to search the Python Package Index. Start by entering the name of the package in the search box above and pressing ENTER.

Install from local file
Click here to locate and install the package file (usually with .whl, .tar.gz or .zip extension).

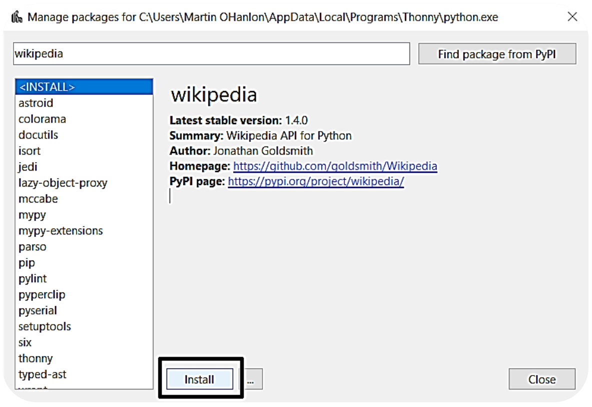
Upgrade or uninstall
Start by selecting the package from the left.

Target: user site packages
This dialog lists all available packages, but allows upgrading and uninstalling only packages from C:\Users\Martin OHanlon\AppData\Roaming\Python\Python37\site-packages. New packages will be also installed into this directory. Other locations must be managed by alternative means.

Close

Figure 48. Find package from PyPI

- Click on *"Install"* to install the package. Finally, when the package has been installed, we click *"close"* (Figure XX).



Manage packages for C:\Users\Martin OHanlon\AppData\Local\Programs\Thonny\python.exe

wikipedia

wikipedia

Latest stable version: 1.4.0
Summary: Wikipedia API for Python
Author: Jonathan Goldsmith
Homepage: https://github.com/goldsmith/Wikipedia
PyPI page: https://pypi.org/project/wikipedia/

Figure 49. install the package

IV.3. Results:

IV.3.1. Prototype of Robot model:

The figure represents the final form of our robot model:

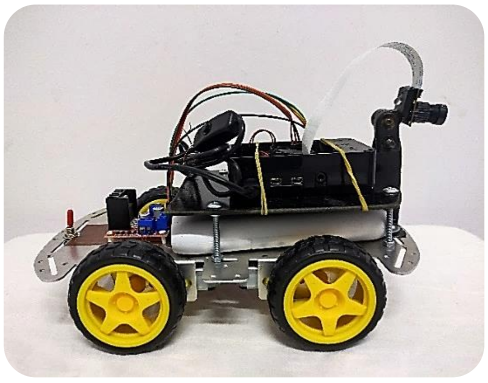


Figure 50. Side Perspective of the Car

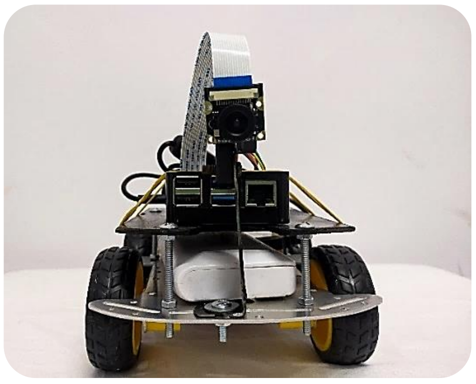


Figure 51. Front Perspective of the Car

IV.3.2. The track:

The following figure represents the track layout of our robot model:

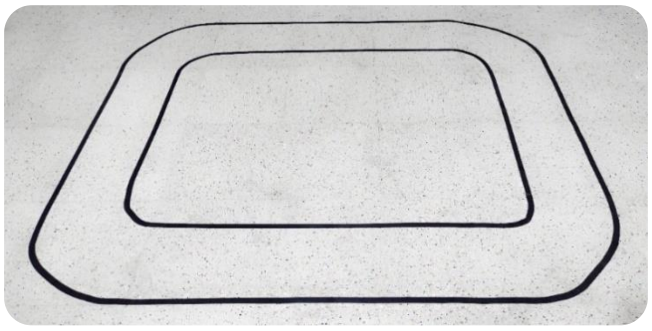


Figure 52. Track Layout

After we released our car on the track layer it was capable of moving forward but unfortunately there were a problem in the corners, we will try to solve it in our future work.

Conclusion:

In this chapter, we have seen how to set up a detection system for lane lines, which can be used for self-driving cars. The algorithm we used gave us good results. However, it is not always perfect; for example, our track couldn't follow the lines in the corners of the track layout so we need to optimize the driving decision step.

GENERAL CONCLUSION:

We have come a long way since chapter 1, where we managed to build our own car that is able to detect the lane line of the road.

To recap, in chapter 1, we introduced the three types of an image and its characteristics, then we touched on the definition of Computer Vision and how it works, as well as some of its application fields in real life, and then we detailed the OpenCV library and its principal modules.

In chapter 2, we divided this chapter into two parts. In part one, we tried to write an algorithm that helps us define the lane line of the road, after several steps from processing the captured images to eliminating any noise in them, to applying canny method and Hough transform, and finally finding the steering angle, which we directed the car along the road based on it.

In part two, we saw the hardware and software needed to develop this project; we began with the hardware where we introduced the component of our car: Raspberry Pi, Pi camera module, l289 motor driver and others. Then the software where we started working with the Raspberry Pi by installing the Raspbian operating system and configuring it. Then installing OpenCV and other libraries. Finally, we showed the result we have obtained.

This project has introduced us to a new field of Computer Vision that we really want to exploit alongside Deep learning in our future project.

REFERENCES:

BOOKS and PROJECT REPORT:

[1] Raspberry Pi Computer Vision Programming | by ***Ashwin Pajankar***

[2] Beginning Robotics with Raspberry Pi and Arduino | by ***Jeff Cicolani***

[3] SELF DRIVING CAR | by ***Mit Patel***

[4] Learn Computer Vision Using OpenCV | by ***Sunila Gollapudi***

[5] Programming Computer Vision with Python | by ***Jan Erik Solem***

WEBOGRAPHY:

[1] https://www.encyclopedia.com/computing/news-wires-white-papers-and-books/digital-images#:~:text=A%20digital%20image%20is%20a,called%20pixels%20(picture%20elements). Consulted on 02/05/2022

[2] https://www.javatpoint.com/dip-types-of-images Consulted on 10/05/2022

[3] https://guides.lib.umich.edu/c.php?g=282942&p=1885348 Consulted on 10/05/2022

[4] https://www.v7labs.com/blog/computer-vision-applications Consulted on 15/05/2022

[5] https://opencv.org/about/ Consulted on 15/05/2022

[6] https://docs.opencv.org/4.x/index.html Consulted on 12/05/2022

[7] https://cutt.us/kqyyX /Good Calculators / Consulted on 22/05/2022

[8] https://iq.opengenus.org/gaussian-blur/ Consulted on 26/05/2022

[9] https://justin-liang.com/tutorials/canny/ Consulted on 26/05/2022

[10] https://cutt.us/nu5Ia /Geeks for Geeks / Consulted on 05/06/2022

[11] https://cutt.us/kZ7e6 / Consulted on 19/05/2022