Universidad de Valladolid

Department of Telecommunication Engineering

Final Degree Project

Autonomous vehicle control by 3D camera in a Twizy

Author:
Mohammad Kanaan

Tutors:
Dr. Juan Carlos Aguado Manzano
Mr. Adrian Mazaira Hernandez

VALLADOLID, JULY 2022

Table of Contents

Intro

The idea of the project is a continuation and taking a step further a previous project, the twizy line project, which is a car-sharing service project. Users through a mobile application can request a car and it shall drive itself (guided by lines on the ground) to the pick up zone, and they are done with the car, they park it at the leaving zone, and again it shall drive itself from the leaving zone to an appropriate place in the park zone.

Figure 1. TwizyLine logo.

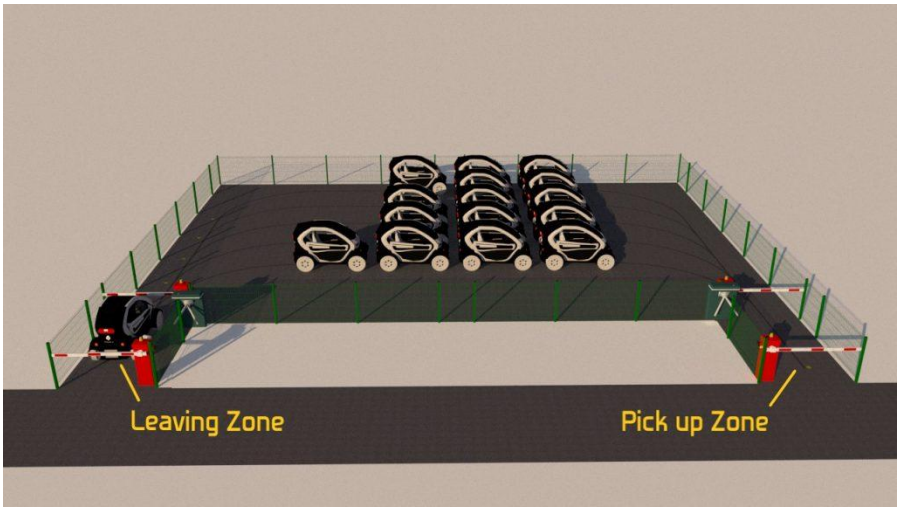


Figure 2. TwizyLine car park.

Break down of students previous work needs to be explained:

Using General Motors look on autonomous driving, autonomous vehicle systems can be split into three parts:

1. Perception: it uses sensors to capture needed data for the later stages.
2. Planning: it uses perceived data to decide the desired behavior for the vehicle.
3. Control: given a path, this part is responsible for following that path by controlling the longitudinal and transversal movement of the vehicle.

1. Perception:
   a. For line following a magnetic strip was used
   b. For obstacle avoidance ultrasonic sensors were used
   c. An odometer in the wheels to know their speed and to count the distance traveled by the vehicle.
   d. A GPS receiver was used so that the barrier opens when a vehicle is less than 15 meters from it.

e. RFID tags were deployed on different places in the park zone, so that a vehicle equipped with an RFID antenna would be able to read them and thus know its position inside the park zone.

2. Planning: planning was done on two levels: “long-term planning” and “short-term planning”.

a. long-term planning: it’s a general path planning that was done by the back end in which the back end chooses the best route for the car.

b. short-term planning: after the best route has been decided by the backend now the vehicle is responsible to move at a suitable speed or even when to stop and when to take a turn.

3. Control: longitudinal and lateral control.

a. longitudinal control: when the vehicle is on autonomous mode the signal coming from the pressure sensor in the accelerator pedal is ignored and similar signals are simulated and sent by an outer circuit.

b. As for the lateral control, it was done by “coupling gears that connect to an electric motor on the current steering column of the Twizy”.

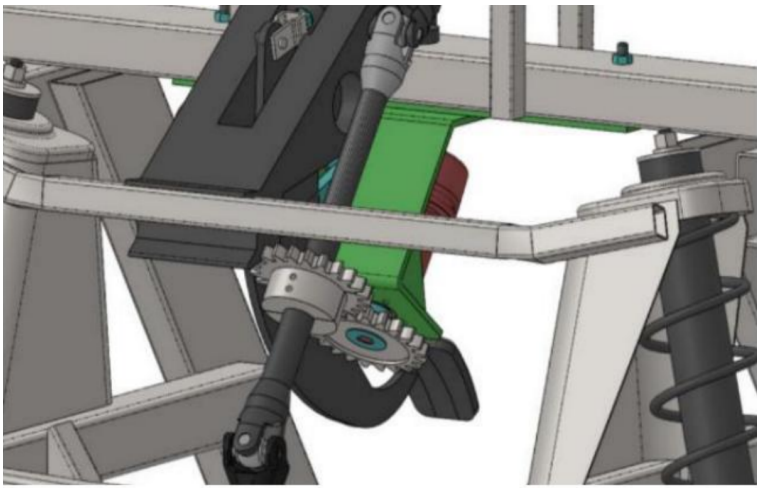


Figure 3. Control of the vehicle's steering wheel using an electric motor.

c. Processing: two Humming Board CBi were used as the car’s brain.

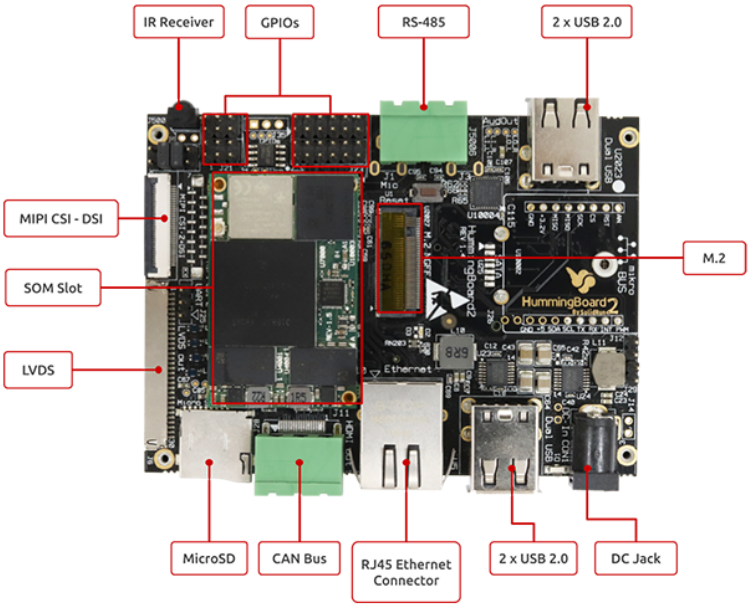


Figure 4. Interfaces available on the Humming Board CBi.

One board was used for communication purposes while the other was responsible for controlling the vehicle using the following configuration.

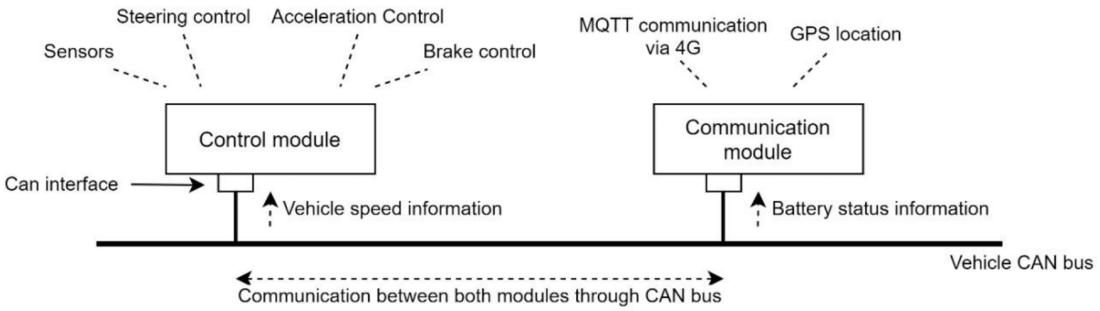


Figure 5. Tasks of the modules and interconnection.

Communication module responsibilities:

a. Getting GPS location from GPS receiver.
b. Communication with the MQTT server.
c. Getting battery status.

Control module responsibilities:

a. “Interpreting information from the sensors and from the car (such as speed)”.
b. Sending control commands to the actuators

- The two modules are connected by a can bus.

Project Objectives:
Things added to the project from my end:

1. Adding a stereo camera capable of:
    a. Road lines detection.
    b. General objects detection.
2. Adding a circuit capable of controlling the car's gears.
3. Replacing the throttle circuit as the previous one had a few problems.
4. Backing up the two humming boards.

Perception Sensors

Environment perception sensors play a vital role in autonomous driving as they are mainly responsible for localization, mapping, depth sensing and object detection.

1. **localization** is the process of determining where a mobile robot is located with respect to its environment. Localization is one of the most fundamental competencies required by an autonomous robot as the knowledge of the robot's own location is an essential precursor to making decisions about future actions.[1]

2. In the world of robotics the problem of **mapping** is that of acquiring a spatial model of a robot's environment. [2]

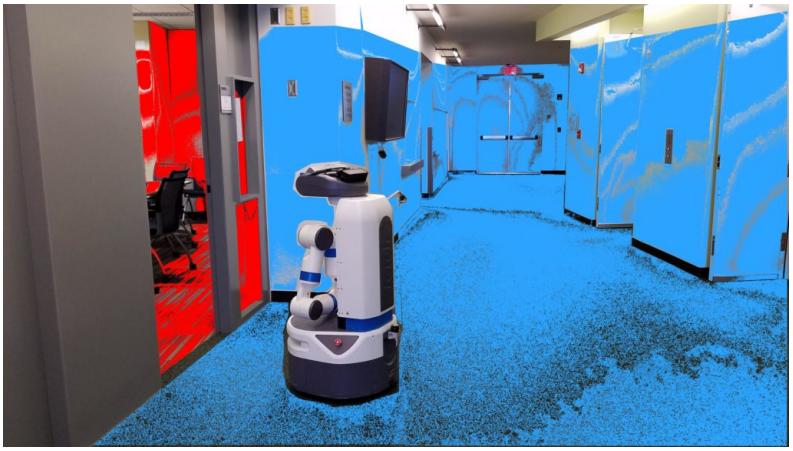


Figure 6. A robot mapping the surrounding environment.

3. **Depth sensing:**

Sensors like ultrasonic sensors they consist of transmitter which emits ultrasonic sound waves and depending on the time it needs to reach the receiver, the distance to the nearest obstacle in its field of view can be calculated. [3]

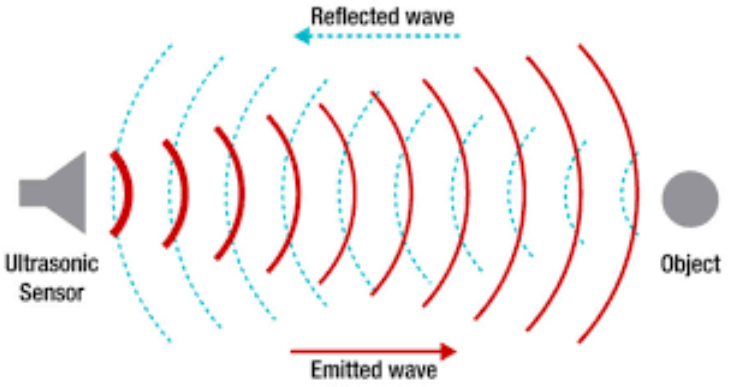


Figure 7. Ultrasonic sensor.

While sensors like stereo cameras and lidars can be used to generate depth maps which can be represented as grayscale images, the higher the intensity value (the whiter) a pixel has the closer it is to the origin point.

Figure 8. Example of a depth map.

They could even generate 3D point clouds where each pixel is assigned to it (x,y,z) values from the point of origin. [4]

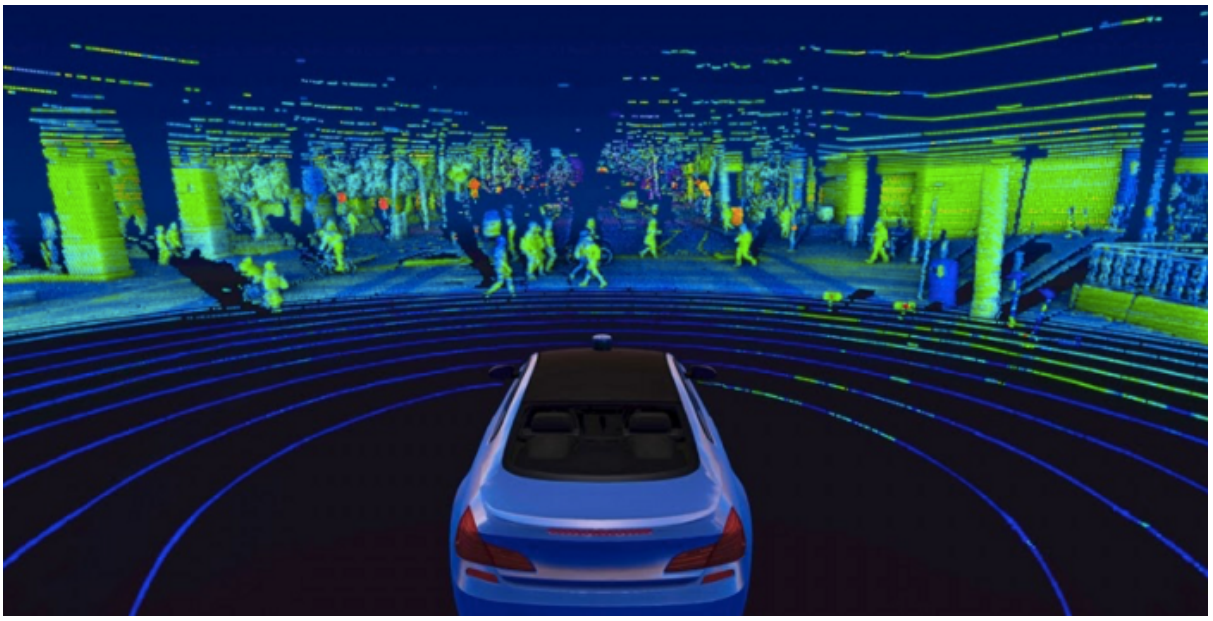


Figure 9. A vehicle capturing a 3D point cloud of the environment.

4. **Object detection** comes in many shapes but in general, its purpose is to classify objects present in the perceived area of the environment and output their locations.

Figure 10. Object detection vs Semantic segmentation.

- Object detection will be discussed in greater details in a later section.

Stereo Vision

Humans can estimate depth through their vision quite accurately and that's thanks to two main reasons, the first is monocular cues which are cues that can be sensed using only one eye like texture, relative size and motion parallax (how objects sizes change while moving), the second reason which is more accurate and more dependable is binocular cues, it can be simply explained by saying humans use their two eyes to perceive two slightly different images of the same environment and use this difference to perceive a three dimensional structure of the world. [5]

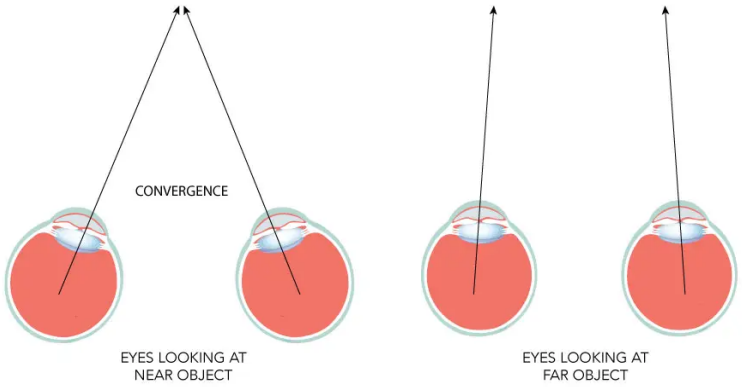


Figure 11. Human eyes looking at near and far obstacles.

Just like binocular vision in humans, cameras can be used in a pair to sense depth.

What's a stereo camera?

A stereo camera is a type of camera with two or more lenses with a separate image sensor or film frame for each lens. This allows the camera to simulate human binocular vision, and therefore gives it the ability to capture three-dimensional images, a process known as stereo photography. Stereo cameras may be used for making stereoviews and 3D pictures for movies, or for range imaging. [6]

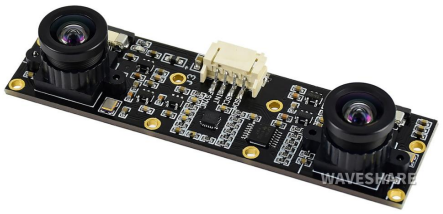


Figure 12. Simple stereo camera module

Since each camera takes a photo of the same environment. Look at the figure below, to find the distance between the stereo camera and the orange cone, the angle between the cone and the center of each camera is calculated, using these two angles and knowing the distance between the two lenses the distance between the stereo camera and the orange cone is then can be easily calculated.

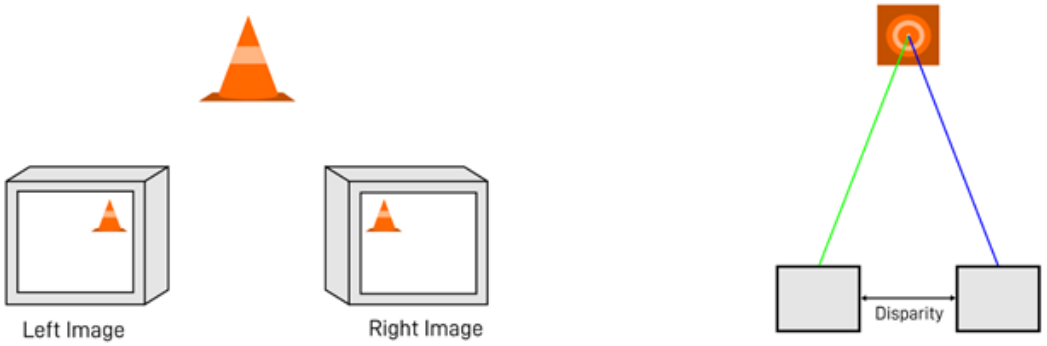


Figure 13. How a stereo camera calculates distance.

Using this method a depth map of the world could be generated, which is usually a grayscale image where the intensity of each pixel indicates the distance from the stereo camera.

Some problems arise from using the previous solution:

1. Occlusion: just like when you put your finger close to your face between your eyes there are parts of your finger your right eye can see but not your left, and since the whole idea of using two lenses is to find common pixels in these two lenses and only then those pixels could be assigned depth to them, pixels which are not common can’t have a depth value.

Look how in the depth map below there are some very black pixels around the tables, this indicates that it's an area that's not common between the two lenses and its depth can't be calculated using the previous method.

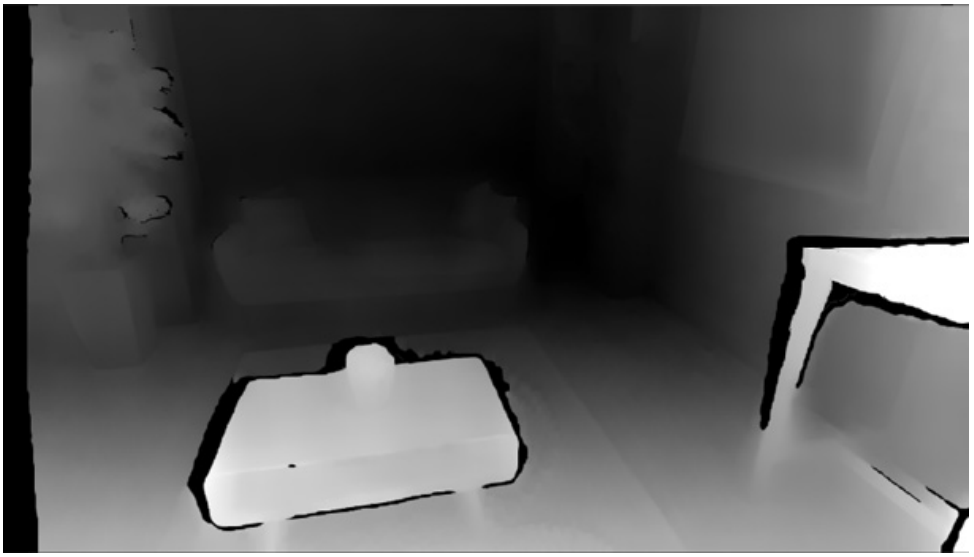


Figure 14. An example of stereo camera's occlusion.

To solve this problem there are a wide variety of solutions, the most effective is to use AI to estimate the depth value for those pixels.

2. Flat untextured surfaces: since the method looks for each pixel in one photo and the same pixel in the other picture it could be sometimes confusing to find those common pixels when there are untextured objects in the environment.

Figure 15. A Flat untextured wall.

That's why sometimes a projector is used along with the camera to project different shades of different colors to help with this problem so finding common pixels would be easy.

The more common and more feasible solution is to use an infrared sensor along with the dual lens setup to aid with those difficult shots.

Comparison Between Different Types of Perception Sensors

Table 1. Comparison between different types of perception sensors. [7] [8] [9]

	Range	Data resolution	Affected by weather	cost
Ultrasonic	Close range up to ten meters	Low resolution	No	Inexpensive
Radar	Long range (100 to 200 meters)	Low resolution	No	Inexpensive
Stereo Camera	Depends E.g.some up to a few meters some up to around 40	High resolution	Yes	Moderate
Lidar	Long range (up to around 200 meters)	High resolution	No	Expensive

Although stereo cameras can be affected by weather and it's true their depth sensing range isn't the best but the data they output is the optimal type of data for identifying objects as they not only provide information about how far an object is from the camera but they also output rgb color information that could be very important when identifying objects in the environment.

Now that the concept of stereo cameras is broadly explained, the next section will be an explanation of the exact camera that will be used.

Intel® RealSense™ Depth Camera D435i

The depth camera D435i is part of the Intel® RealSense™ D400 series of cameras, a lineup that takes Intel's latest depth-sensing hardware and software and packages them into easy-to-integrate products. Perfect for developers, makers, and innovators looking to bring depth sensing to devices, Intel® RealSense™ D400 series cameras offer simple out-of-the-box integration and enable a whole new generation of intelligent vision-equipped solutions.

Figure 16. RealSense D435i camera.

Table 2: RealSense D435i features

Depth range	.3 m to 3 m	RGB sensor resolution	2 MP
Depth FOV	87° × 58°	RGB FOV	69° × 42°
Depth resolution	1280 × 720	RGB resolution	1920 × 1080
Depth frame rate	Up to 90 fps	RGB frame rate	30 fps

Table 3: RealSense D435i physical features

Connectors	USB-C 3.1 Gen 1
Length × Depth × Height	90 mm × 25 mm × 25 mm
Mounting mechanism	– One 1/4-20 UNC thread mounting point. – Two M3 thread mounting points.

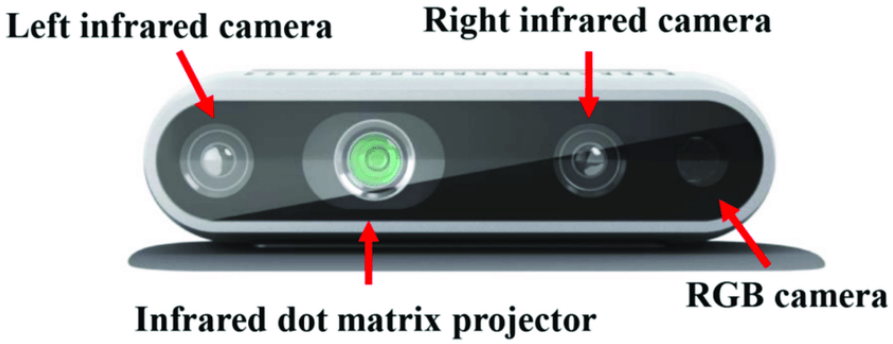


Figure 17. RealSense D435i key components

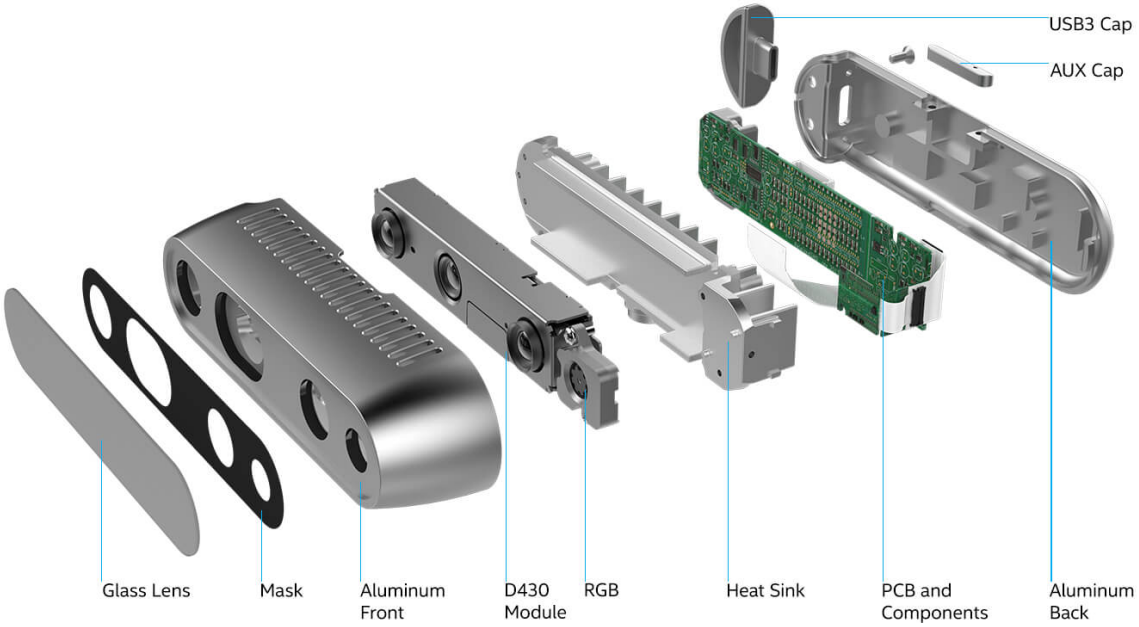


Figure 18. RealSense D435i teardown.

- One of the biggest disadvantages of using this camera is its very low range of depth 3 meters for a vehicle is unacceptable. That's why although it’s a big decision, it’s been decided not to use the depth capabilities of the camera and only use it as an RGB camera.

Gears Circuit

The Twizy car has three gear states: Drive, Neutral, Reverse.

Figure 19. Renault twizy gears buttons.

Switching between states is done as follows:

1. To enter “Drive” state: the upper button with ‘D’ on it should be pressed.
2. To enter “Reverse” state: the bottom button with ‘R’ on it should be pressed.
3. To enter “Neutral” state: both of the previous buttons should be pressed, the easiest way to do so is pressing the middle part of the gears remote which has ‘N’ printed over it.

Now let’s take a look at the wires going out of the gears remote:
Here’s how the connector looks like

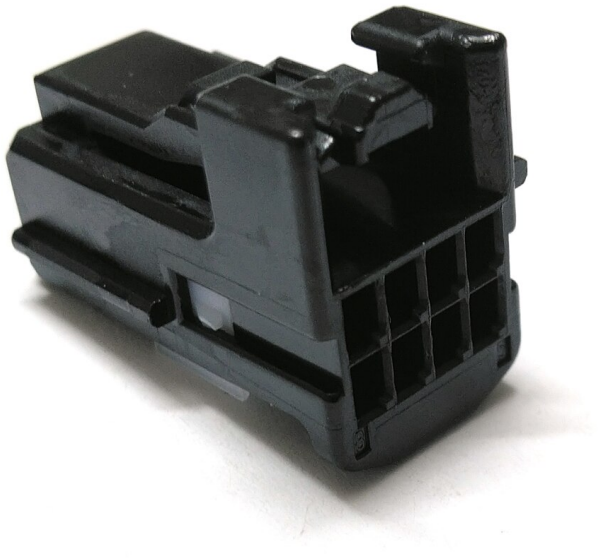


Figure 20. TYCO 0-1379659-1 connector.

Table 4: wires out of gears buttons.

Not connected*	Positive	Drive signal	Ground
Not connected*	Positive	Not connected	Reverse signal

* The not connected ones are just slots left empty; they don't have a wire sticking out of them.
Positive wires have a voltage of approximately 22.23V with ground.

With trial and error it's been found that whenever a signal wire is connected to either a positive or ground it's activated, for example if the Drive signals was to be connected to positive the car will go into "Drive" state, the same happens if it's connected to ground the car will go into "Drive" state, the same can be said about the Reverse signal.

In order to go into "Neutral" state however both of the Drive signal and the Reverse signals should be activated and that's by connecting both of them to positive, both of them to ground or one of them to positive and the other to ground.

To simulate the previous states the following circuit was built.

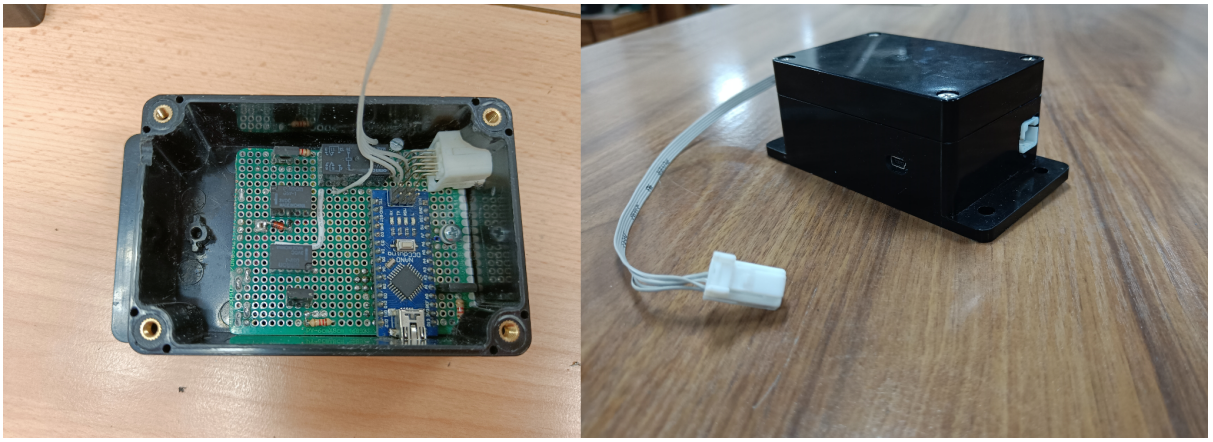


Figure 21. Gears circuit.

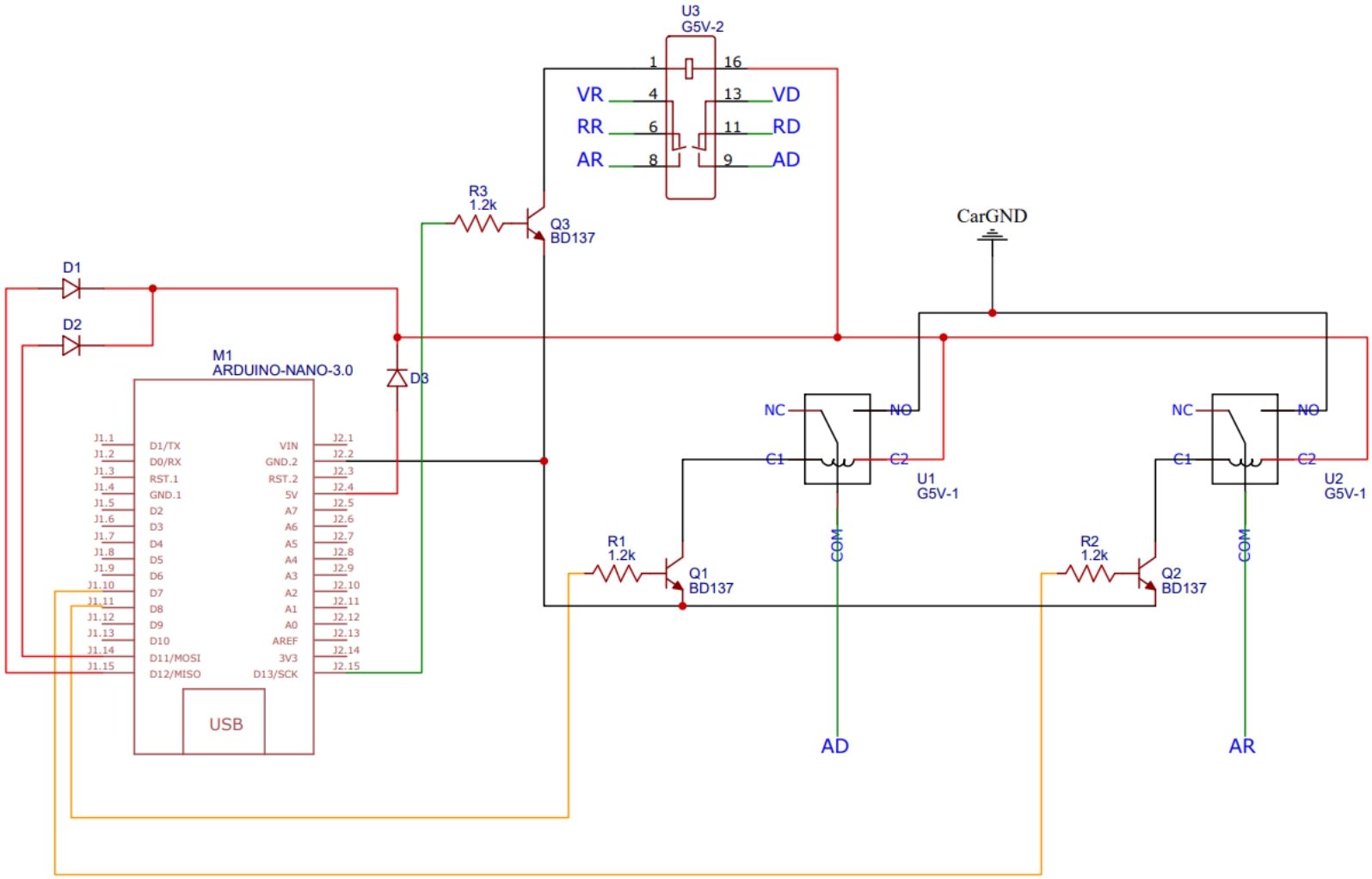


Figure 22. Gears circuit schematics.

AR/AD: Arduino reverse/Arduino drive.
RR/RD: Remote reverse/Remote drive.
VR/VD: Vehicle reverse/Vehicle drive.
Note: pins 11 and 12 have been used along with diodes to supply the relays with enough current.

Throttle Circuit

The throttle system in most vehicles is controlled by how much the accelerator pedal (or gas pedal in gas cars) is pressed and based on it the motor should supply power accordingly.

There are many ways to read pressure in the accelerator pedal, the simplest is to use a potentiometer that changes its value when the pressure on the pedal is changed.

However, relying on one potentiometer could be quite risky and that's why most companies decided to use two potentiometers so a single point of failure throttle system is avoided.

Building the throttle circuit:
In Renault twizy wires going to the pedal are as follows:

Figure 23. Wires going to the pedal.

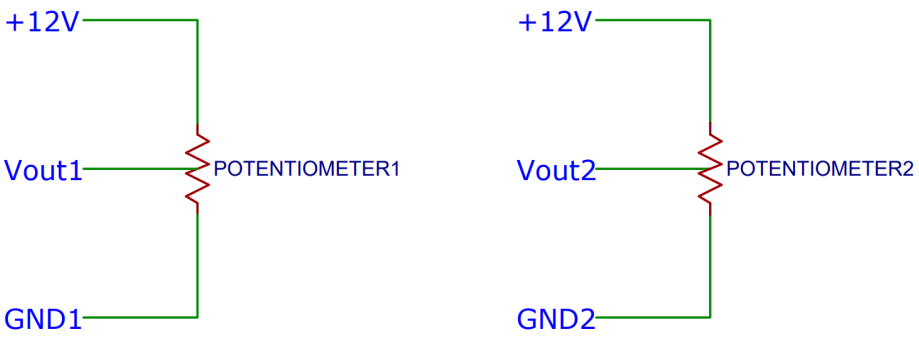


Figure 24. Wires going to the pedal with relation to the potentiometers.

It’s been found that to reach the wanted speed with the wanted acceleration the readings of the potentiometers should be as follows:

Table 5. Wanted potentiometers readings.

Potentiometer value	Vout1 (V)	Vout2 (V)
0%	0.96	4.06
10%	5.09	1.41
20%	6.30	1.86
30%	7.40	2.32
40%	8.80	2.78
50%	9.95	3.23
60%	10.71	3.69
70%	11.33	4.15
80%	11.74	4.59
90%	11.75	5.05
100%	11.75	5.56

Due to the supply shortage in the market only digital potentiometers with two terminals were available, so in order to solve this problem, fixed value resistors were used with this type of digital potentiometers to divide the voltage between them.

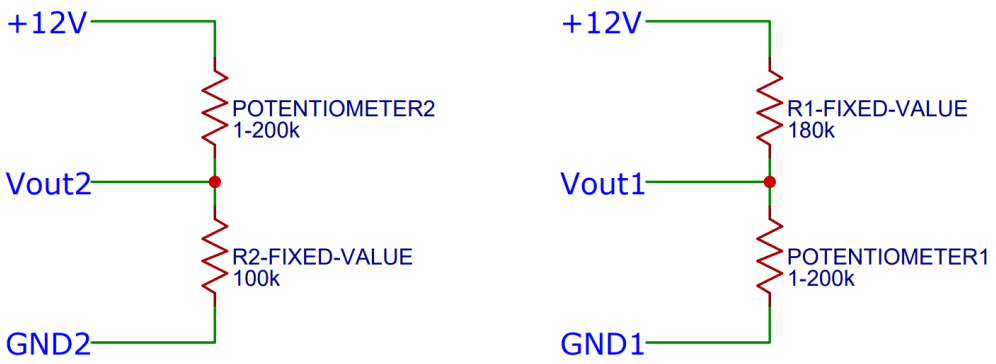


Figure 25. Needed circuit diagram.

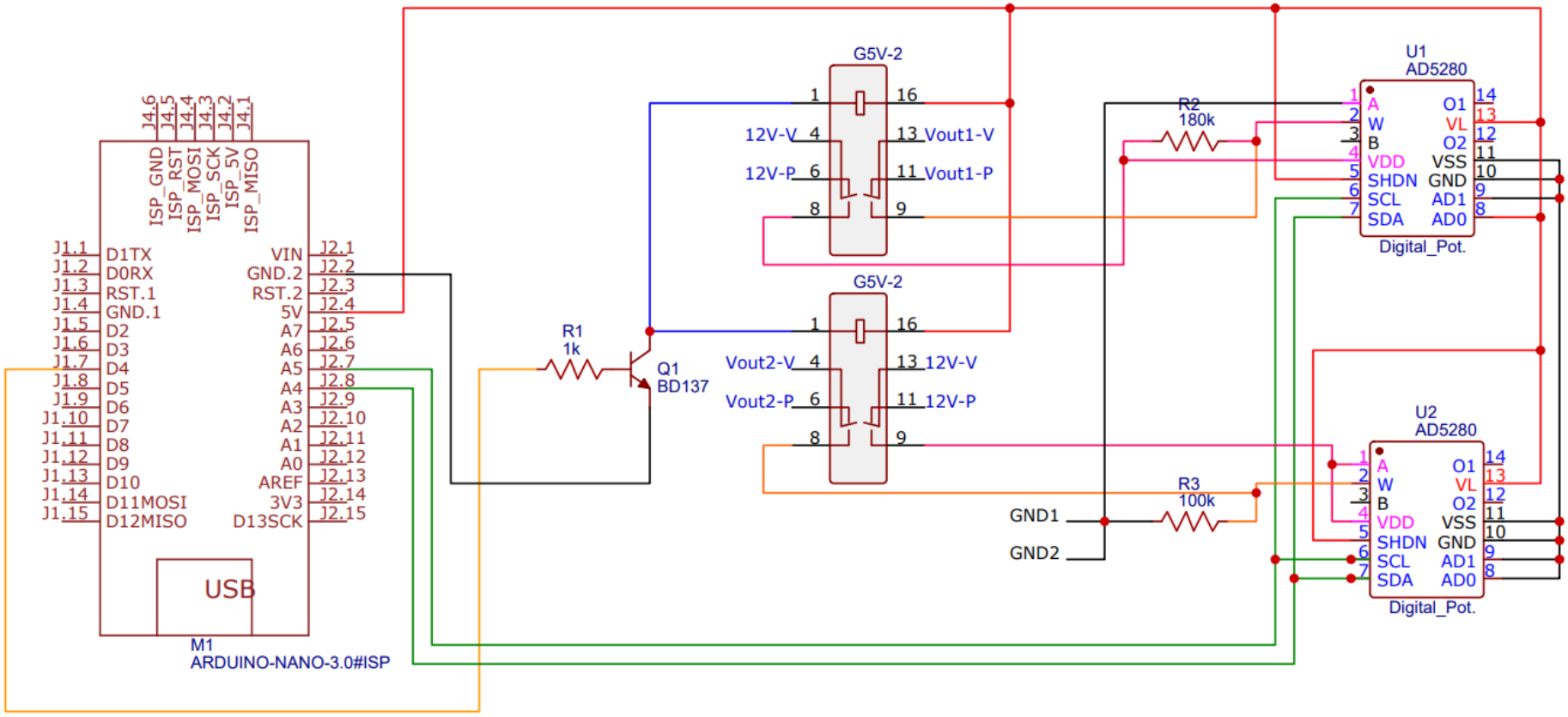


Figure 26. Throttle circuit schematics.

12V-V: 12V from the vehicle.
12V-P: 12V to the padel.
Vout-P: Vout from the padel.

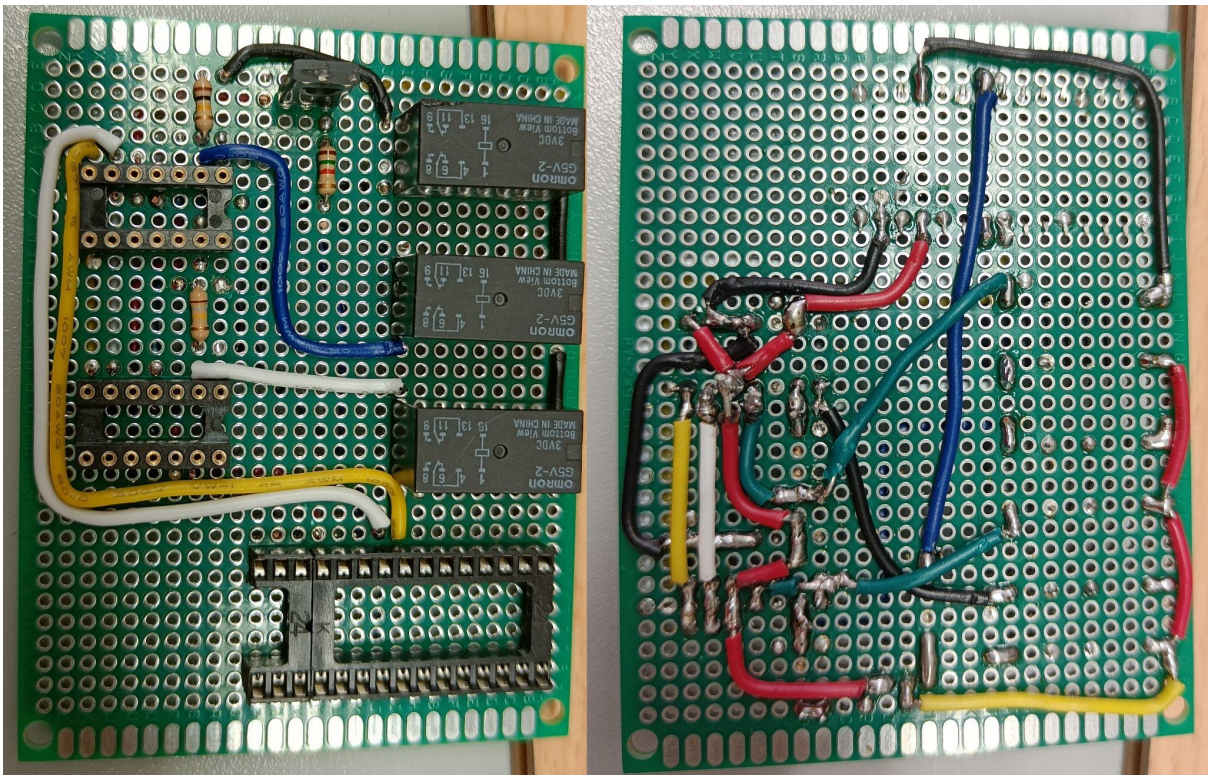


Figure 27. Throttle circuit upside and downside photos.

The bigger socket is where the arduino is placed, while smaller ones have AD5280 digital potentiometers placed in them.

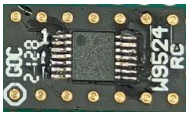


Figure 28. AD5280 digital potentiometer.

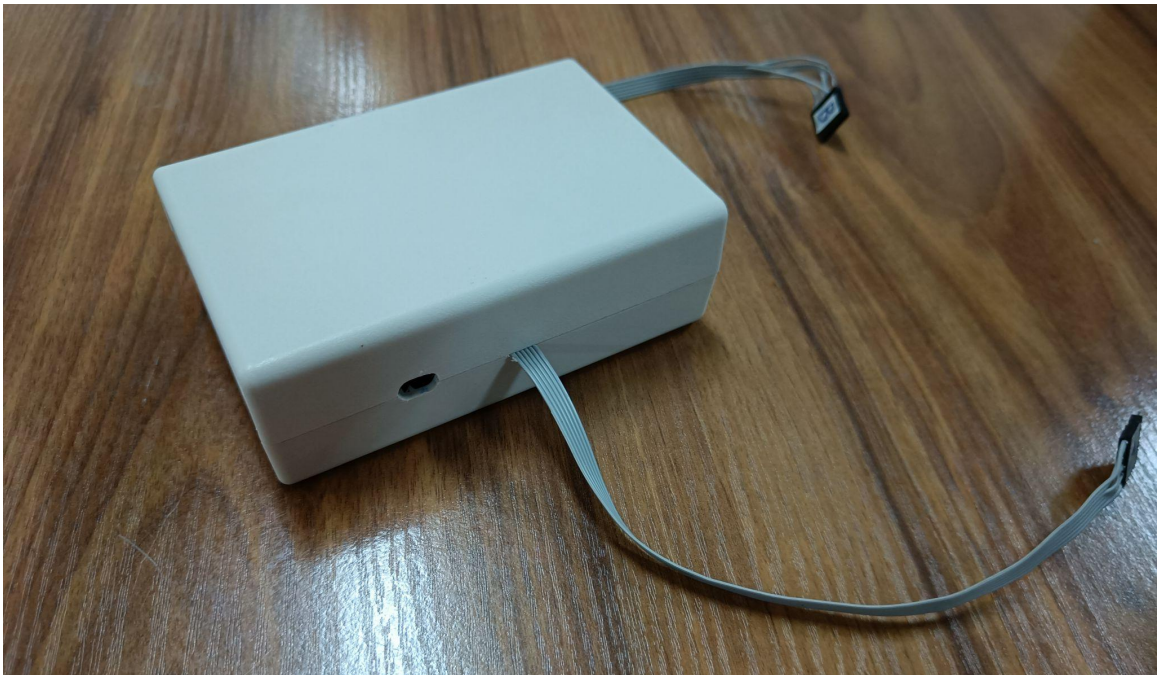


Figure 29. Throttle circuit box.

Road Line Detection

This section discusses a computer vision technique that takes an image of a road as an input and marks the possible paths.

In the case of the TwizLine project there are two types of road lines: curved lines and straight lines, as the figure below shows, orange lines represent curved lines while green one represent straight lines.

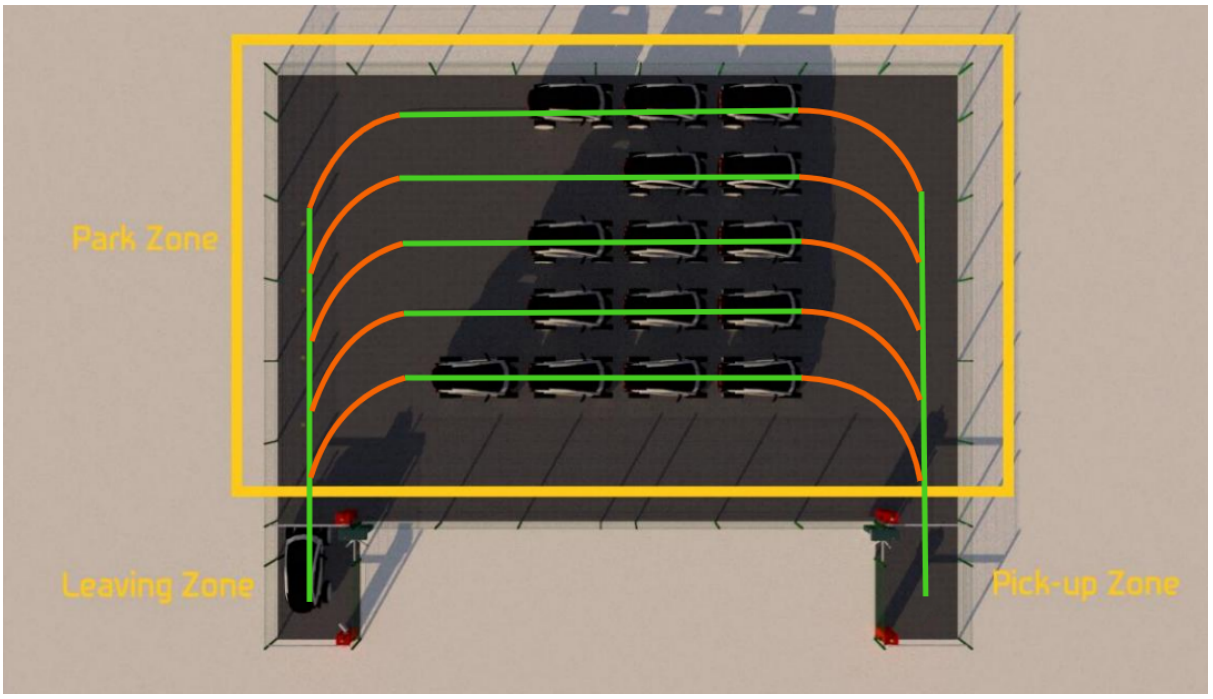


Figure 30. Types of road lines.

To mark road lines within an image, the image goes through multiple steps which can be described as follows:

1. Convert it into a suitable color space.
2. Keep only the pixels which are in the range of colors that of the road lines.
3. Grayscale the image.
4. Eliminate noises by blurring.
5. Detect those pixels which represent the outer side of the road lines.
6. Pass those pixels into Hough transform algorithm to get lines wherever there are pixels that happen to align.
7. draw the possible road paths using hough lines by taking the average slope of lines that are close to each other if the slope is below a certain value then it's a curved line otherwise it's a straight line.

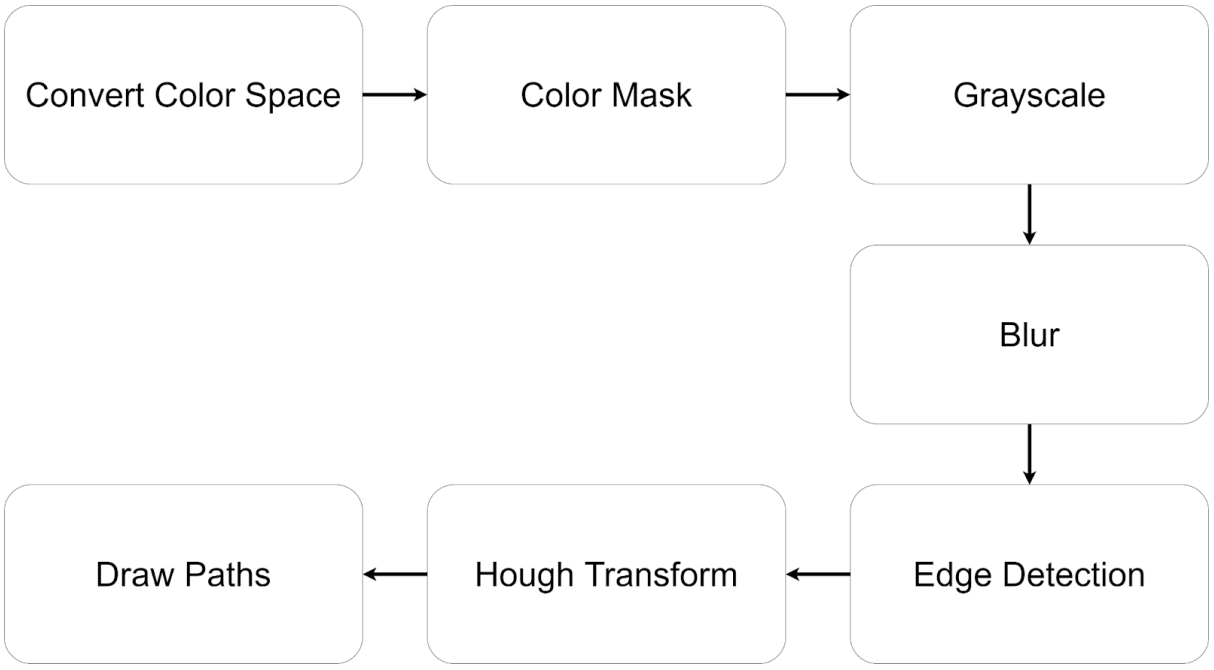


Figure 31. Used method for road Line detection diagram.

Tools used

Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. [10]

There are multiple reasons for using Numpy over standard python lists:

1. Working with numpy saves a lot of computational resources for the following reasons:

    a. It needs less memory and that's because for each element a list stores way more information about it while numpy by default stores only the value of that element as int32.

    b. It doesn't type check.

    c. it stores arrays as contiguous memory, meaning it doesn't scatter an array all over the memory wherever there's free space, instead an array's elements are stored one after the other without any different data separating those elements.

2. We can do more to it other than the basic operations: insertion, deletion, concatenating, … etc.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

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. [11]

Step 1 Convert the image into a suitable color space

Usually colors in images are represented in RGB or BGR, so here's a quick explanation of this model:

RGB Color Space

Each color in the RGB model has three values (R,G,B) with R representing how red a certain color is, G is for green and B is for blue.

For example in 8 bit RGB format R takes value from 0 to 255 with zero meaning that the color has no relation with red or that it has no red component and 255, the same goes for G and B.

This is an example of a color with RGB values of (255,161,51).

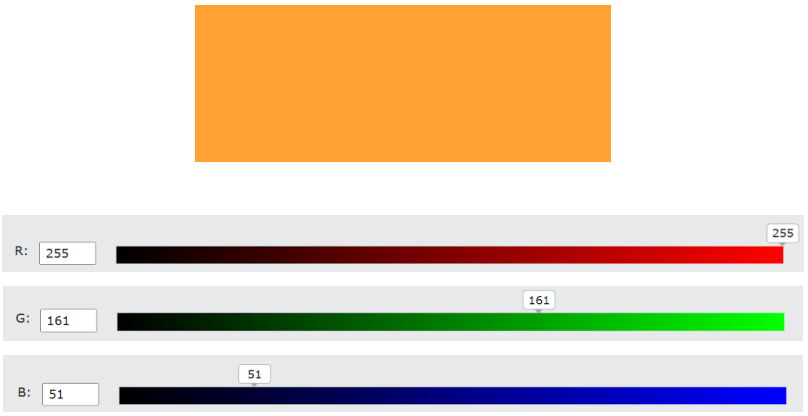


Figure 32. RGB colorspace.

Why not use RGB

Although RGB color space is one of the most common and convenient ways to represent colors, one important downside of using this kind of representation in color detection/thresholding is that it's not robust to minor changes in external lighting.

For instance, two shades of red might have very different RGB values making the range of color for the road line very wide which would result in passing more pixels that are not part of the road line.

One of the most common solutions for this problem is using a different color space like HSV or HSL.

HSV Color Space

Each color in the RGB model has three values (H,S,V).

H or hue is the color portion of the model, expressed as a number from 0 to 360 degrees:

- Red falls between 0 and 60 degrees.
- Yellow falls between 61 and 120 degrees.
- Green falls between 121 and 180 degrees.
- Cyan falls between 181 and 240 degrees.
- Blue falls between 241 and 300 degrees.
- Magenta falls between 301 and 360 degrees.

Figure 33. Hue spectrum in HSV colorspace.

S or saturation is the amount or the intensity of a color (dominance of Hue) (0-100%)

Figure 34. Saturation spectrum in HSV colorspace.

V or value is basically the brightness of the color. (0-100%)

Figure 35. Value spectrum in HSV colorspace.

Note that ranges for (H,S,V) in openCV are as follows:
H takes the values from 0 to 179 while S and V take values from 0 to 255.

Step 2 Color Mask

In the previous step the image has been converted from RGB to HSV, in this step however only pixels in the range of color of the road line will remain and all of the other pixels will be turned to absolute black.

In order to do that, the range of color of the road must be defined and to do that a number of pictures have been taken of the material that will be used as road lines in different lighting conditions.

After that, a mask will be created out of the image which in this case is a two dimensional array with each element holding a value of either 255 when the corresponding pixel has (HSV) values in the road line range, or 0 when it doesn't.

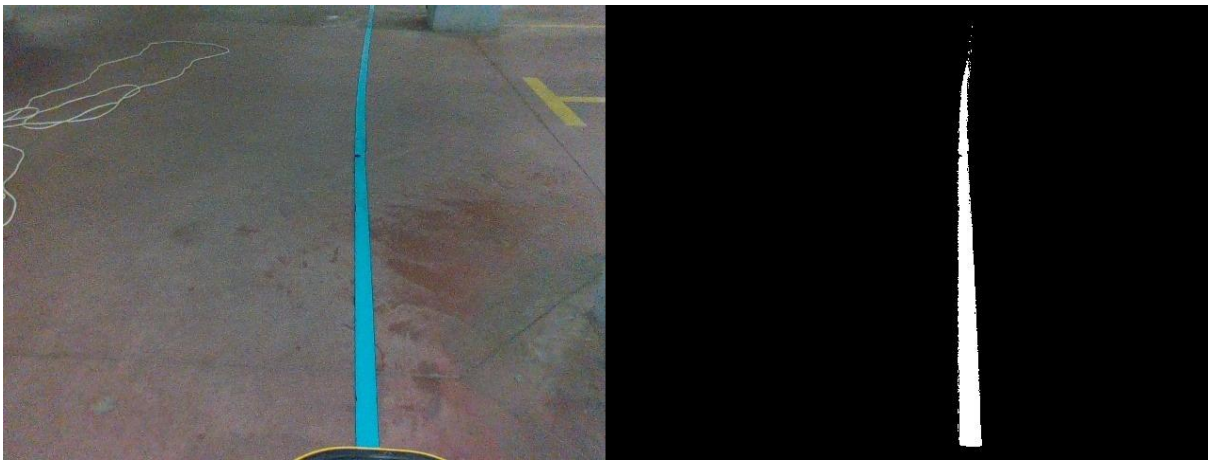


Figure 36. Color masking road line.

The original image and the mask are then anded bitwise resulting in the following image.

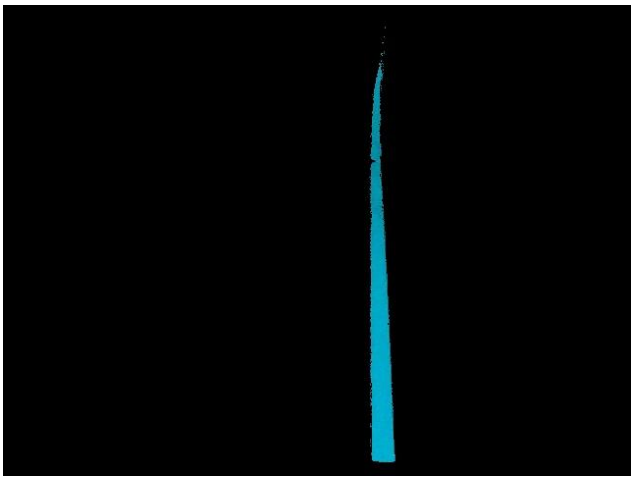


Figure 37 Original image and the mask anded bitwise.

Step 3 Grayscaling

What makes this step important is that the edge detection step (canny edge detection) can only work with grayscale images and that's because of the way it works which will be explained later.

In HSV color space each pixel has three values in order to represent its color, in grayscale on the other hand each pixel has only one value representing it's light intensity, usually in a grayscale image each pixel has an 8-bit value starting from 0 (completely dark or black pixel) to 255 (completely illuminated or white pixel).

Figure 38 image from step 2 grayscaled.

Step 4 Blurring

Since most edge-detection algorithms are sensitive to noise, using some kind of blurring to reduce image noise was a necessary step, gaussian blur was used in this method with kernel of size 15.

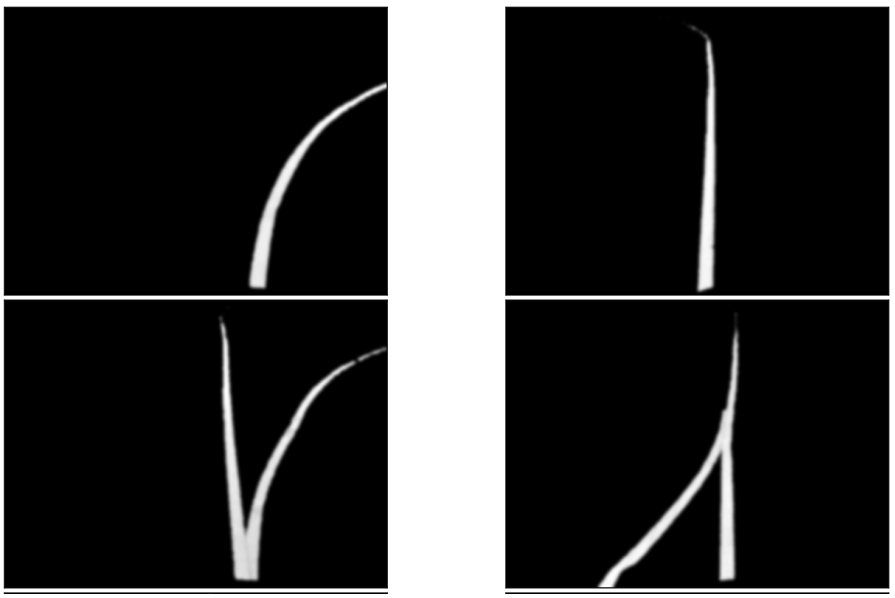


Figure 39. images blurred.

Step 5 Edge Detection

Using Canny edge detector only when there's a change in color intensity that's above the threshold a line is detected.

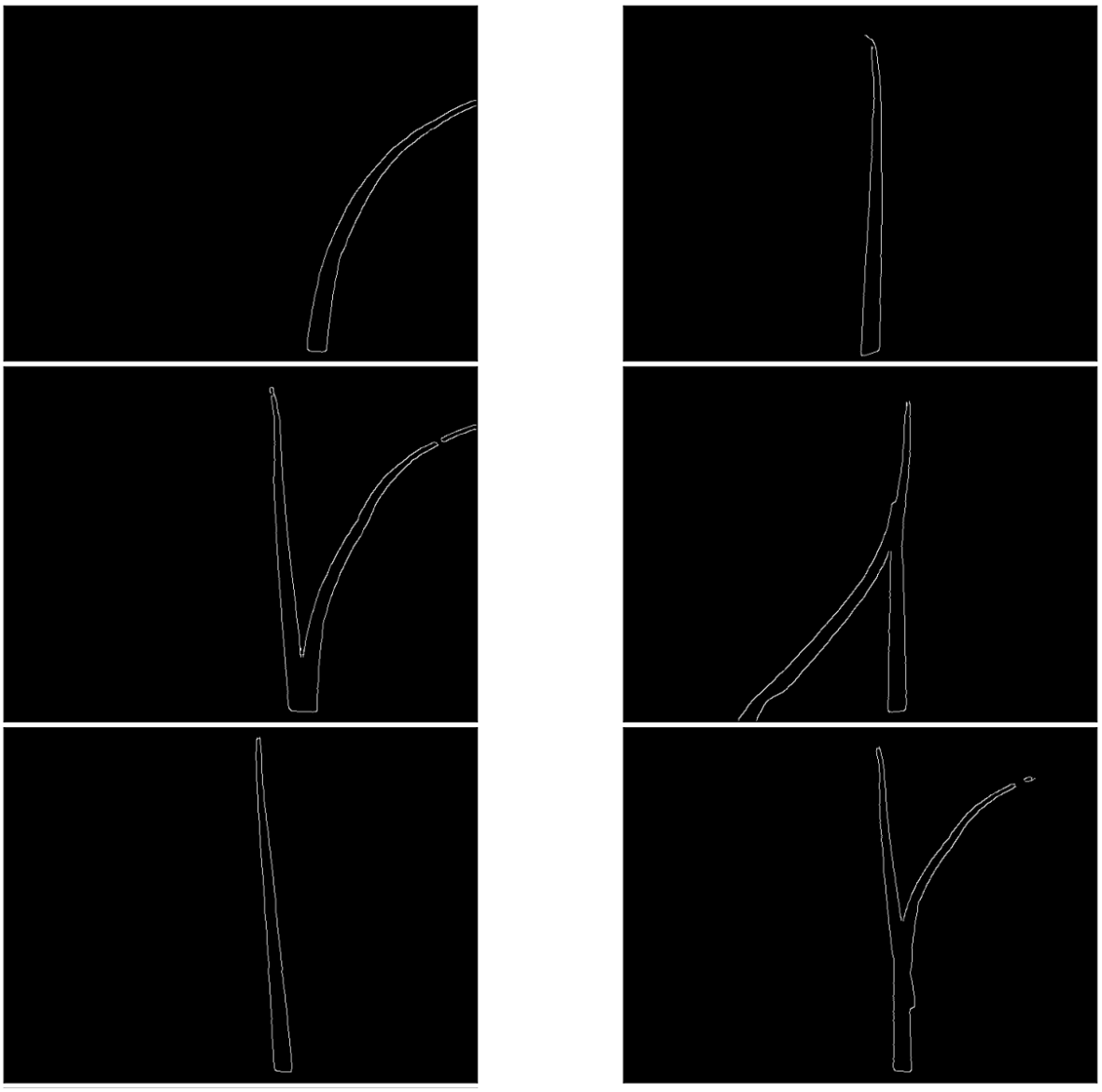


Figure 40. Images after canny edge detection.

Step 6 Hough Transform

To simply put it when enough pixels aline a line is drawn

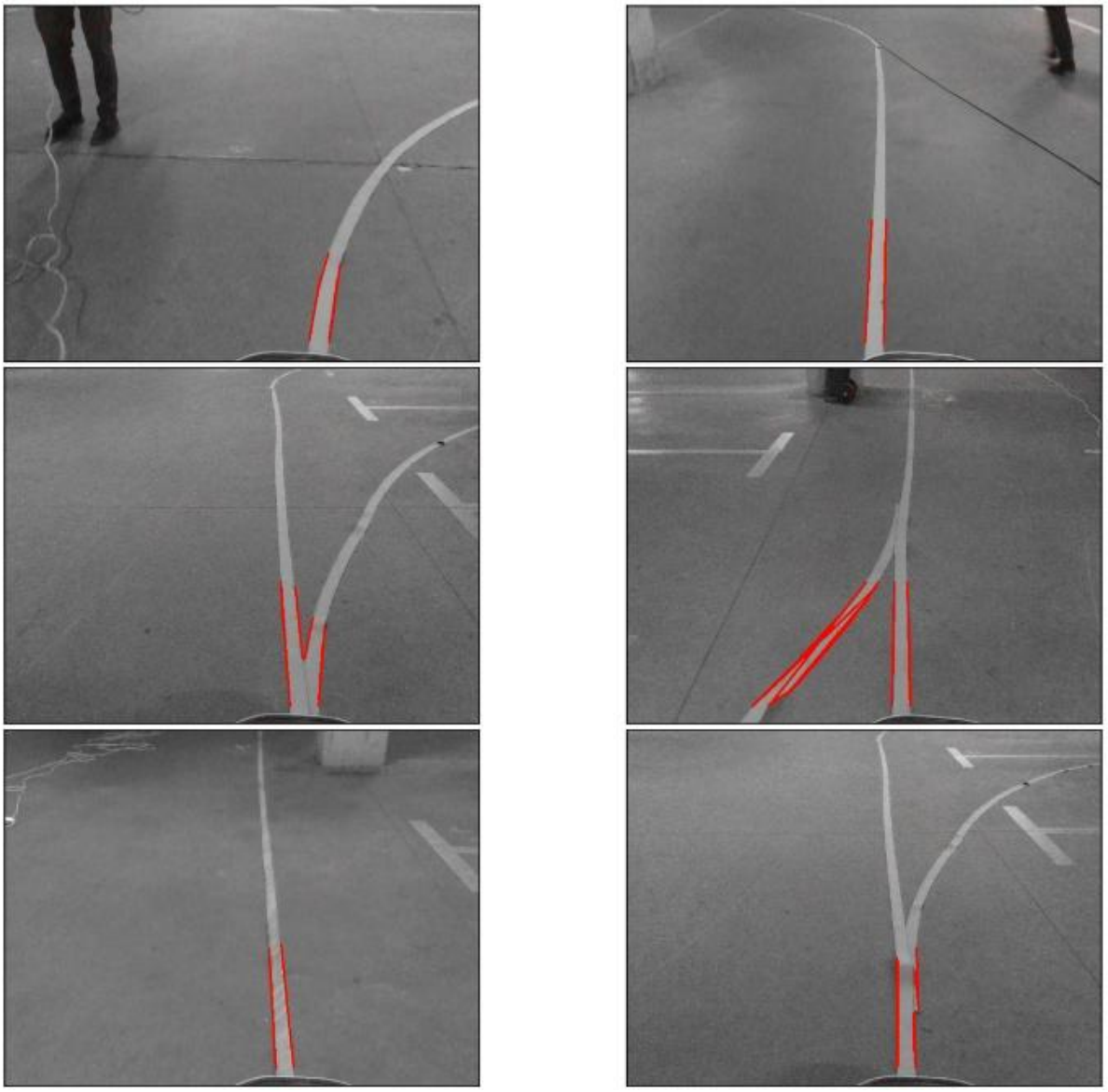


Figure 41. Images after Hough transform.

Step 7 Draw Paths

By taking the average slope of lines that are close to each other, if the slope is below a certain value then it's a curved line otherwise it's a straight line.

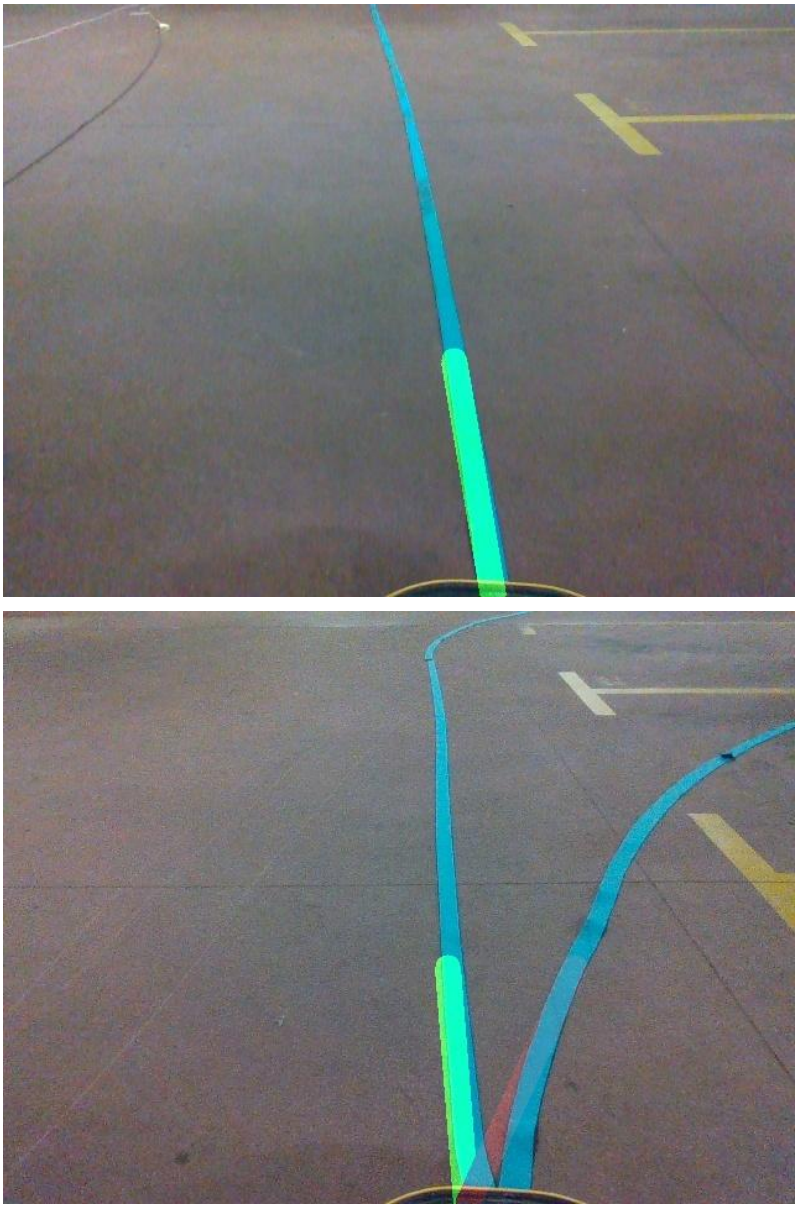


Figure 42. Road line detection.

Object Detection

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer. [12]

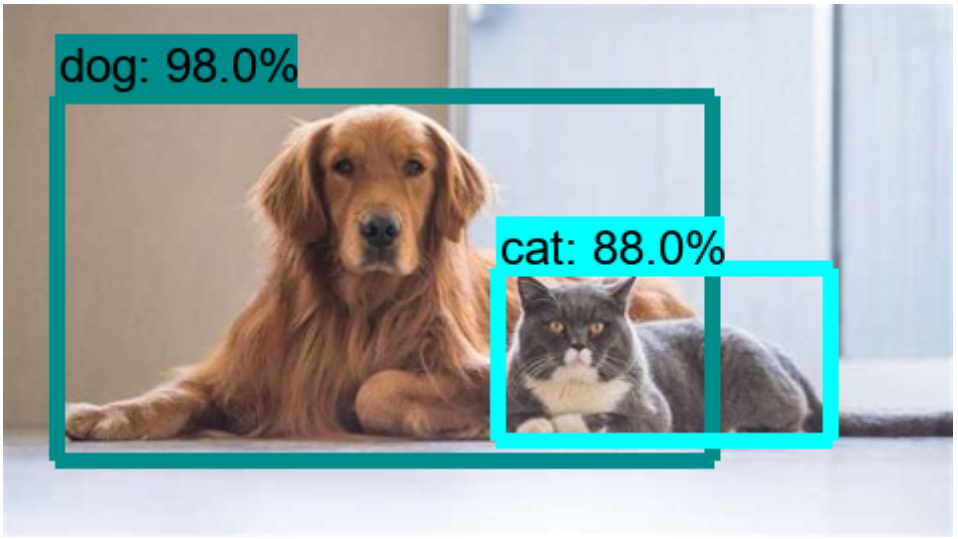


Figure 43. Object detection Example.

Types of machine learning

Broadly speaking, object detection can be broken down into machine learning-based approaches and deep learning-based approaches.

In more traditional ML-based approaches, computer vision techniques are used to look at various features of an image, such as the color histogram or edges, to identify groups of pixels that may belong to an object. These features are then fed into a regression model that predicts the location of the object along with its label.

On the other hand, deep learning-based approaches employ convolutional neural networks (CNNs) to perform end-to-end, unsupervised object detection, in which features don't need to be defined and extracted separately. For a gentle introduction to CNNs, check out this overview.

Because deep learning methods have become the state-of-the-art approaches to object detection, these are the techniques we'll be focusing on for the purposes of this project. [13]

To proceed further it must be clear what deep learning is.

Theory

Deep Learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. [14]

Machine Learning

“Machine Learning is the science (and art) of programming computers so they can learn from data.”
Aurélien Géron in Hands-On Machine Learning with Scikit-Learn and TensorFlow

Here are some very common definitions:

“Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.”
Arthur Samuel, 1959

And a more engineering-oriented one:

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.” Tom Mitchell, 1997

Meaning, if for example we had a system that could tell if a given image is either an apple or an orange, the task T could be the classification itself while E the experience is the images that you’d like the system to train on or what’s also known as the training data, and P can be any performance measure like in our case it could be the percentage of images that were correctly classified to all test images.

Types of Machine Learning

There are many ways to classify the types of machine learning there are, the most common is supervised\unsupervised which describe the amount of supervision the system is getting, generally there are four levels of supervision:

1. Supervised Learning: the training data is labeled for example the apple/orange system is given an amount of apple images that are labeled as apples and the system should find a pattern in these images and link it to apples, labeling can be done in a variety of ways like putting all of the apple images in a folder called apple or by naming the image as apple.

2. Unsupervised Learning: the training data here is unlabeled, an unsupervised learning system can have many purposes, the most common are: clustering, visualization and dimensionality reduction and association rule learning.

    a. Clustering: the system is given data, let's say information about the customers of a certain restaurant without telling the system how to group those customers, the system should be able to split customers into groups based on common information between them.
    b. Visualization and dimensionality reduction: the system here when given data is supposed to plot the data into a 2D or a 3D diagram, while dimensional reduction systems reduce the size of the given data without losing too much information.

    c. Association rule learning: the purpose of this type of systems is to find relation between certain elements of the data.

3. Semi-Supervised Learning: in this type of learning only some of the training data is labeled while a big portion of it is not.

4. Reinforcement Learning: this type of learning is somewhat different, the agent is put into the learning environment with defined rewards, this agent then keeps trying different actions to get the maximum amount of rewards in a strategy called policy.

Artificial Neural Networks

The name Artificial Neural Networks or ANNs came from the idea that ANNs simulate the way real neural networks in the brain work, both consist of a number of neurons or nodes connected together in some sort of way, not only that they are connected but the strength of the connection between neurons might vary from pair to pair, the strength of this connection is called “weight”.

An artificial neural network can be thought of as a usually complicated mathematical equation that takes an amount of inputs usually these inputs are in the range 0 to 1 and outputs one or more outputs.

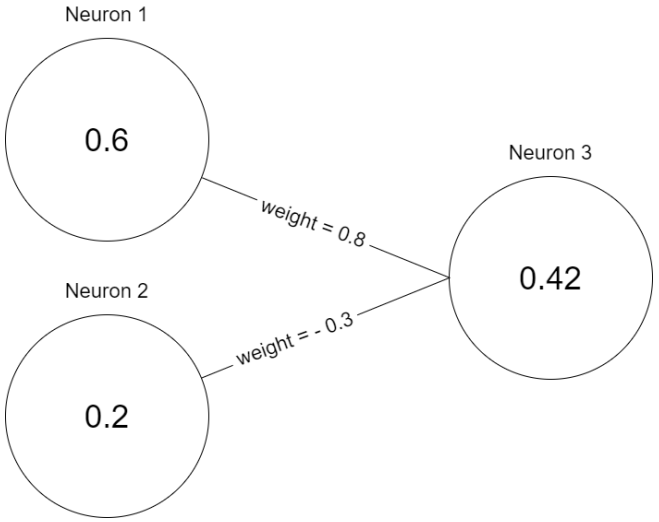


Figure 44. A very simple neural network.

As you can see the function for this neural network is:

activation of neuron1 * weight of the connection between neuron1 and neuron3
+
activation of neuron2 * weight of the connection between neuron2 and neuron3

= activation of neuron3

Note: the activation of a neuron is the value assigned to that neuron.

Although this equation is true for some neural networks, most neural networks aren't linear, no matter how many neurons the network has, if it followed the previous equation it'll always be linear. [15]

$A_{n1} * W1 + A_{n2} * W2 + A_{n3} * W3 + \ldots = A_{nn}$

What if the pattern the data has wasn't linear?

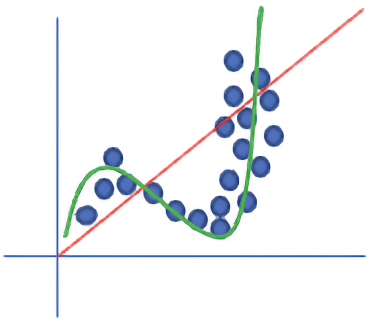


Figure 45. Linear regression vs nonlinear regression

To tackle this problem a concept called the activation function has been introduced, which simply takes the output of the previous layer's neurons as input and passes them through some sort of function which ensures nonlinearity.

$F_{activation}(A_{n1} * W1 + A_{n2} * W2 + A_{n3} * W3 + ….). = A_{nn}$

Most common activation functions

1. Sigmoid:

Sigmoid outputs a value between 0 and 1 but since it's exponential it's very power intensive, using the sigmoid many neurons could make the model slow.

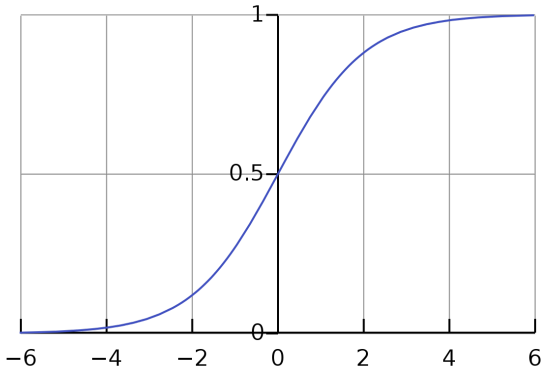


Figure 46. Sigmoid function.

$$f(x) = 1/1 + e^{-x}$$

2. Hyperbolic Tangent (Tanh)

The Tanh outputs a value t=between -1 and 1, it's also exponential and has the same problem as the previous one, but this one is zero centered making it easier to optimize the neurons after it.

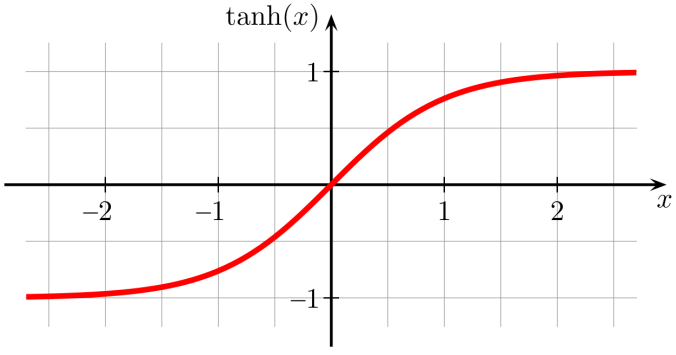


Figure 47. Hyperbolic Tangent function.

$$f(x) = e^{x} - e^{-x} / (e^{x} + e^{-x})$$

3. Rectified Linear Unit (ReLU)

What's so great about the ReLU is that it's so simple making it computationally efficient, but it has a downside that it's not zero-centered.

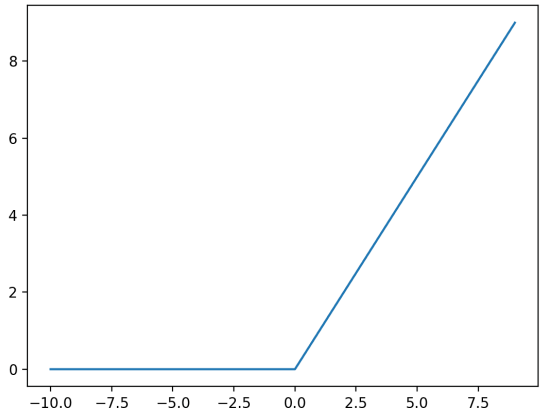


Figure 48. ReLU function.

$$f(x) = \{x\ x >= 0,\ 0\ x < 0\}$$

4. Leaky ReLU

The leaky ReLU has all of the advantages of the ReLU being so simple but it also has a leak value which makes it closer to being zero-centered.

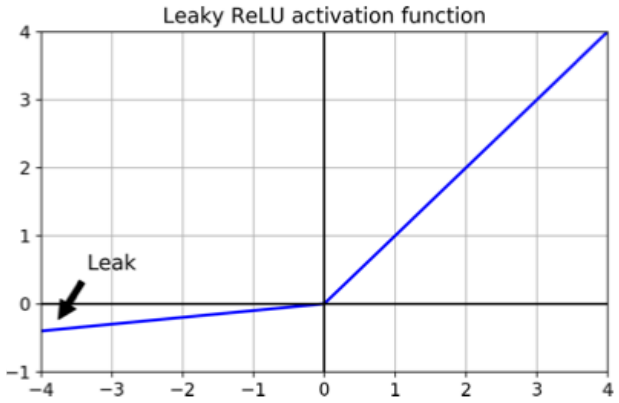


Figure 49. Leaky ReLU function.

$f(x) = \{x\ x >= 0,\ ax\ x < 0\}$ where 'a' is the slope of the leak value

General Structure of a Neural Network

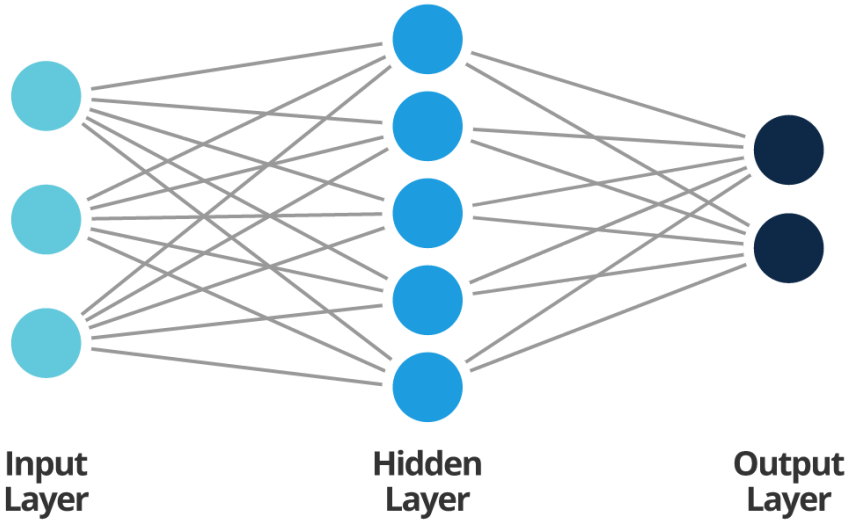


Figure 50. Simple Diagram of a neural network

Neural networks typically consist of:

1. Input layer: data goes in here.
2. Hidden layer(s): when there are more than one it's called deep learning, and in those layers is where the NN is learning the features.
3. Output layer: prediction is outputted here.

Layer Types:

The layers displayed in the previous diagram is an example of dense layers also known as fully connected layers where every neuron is connected to every neuron in the following layer.

Here are some of the most common layers, they'll be explained later in greater details:

1. Dense (fully connected)
2. Convolutional layer
3. Pooling layer
4. Recurrent layer

Supervised Learning

Is a subcategory of machine learning in which a system is supposed to output some kind of prediction after studying a labeled dataset.

An example of a labeled dataset is MNIST dataset which consists of 60,000 images of handwritten digits 0 to 9 with each digit given the corresponding label.

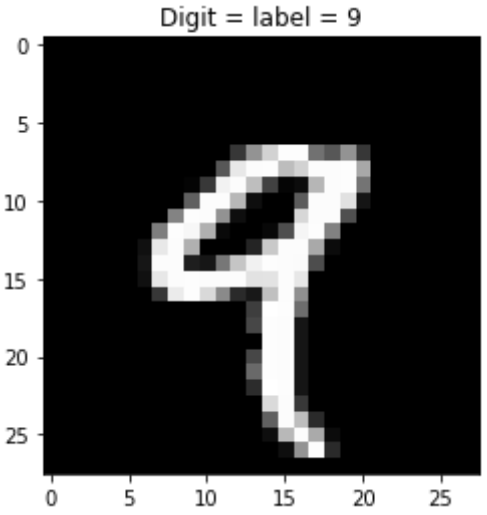


Figure 51. Sample Image from MNIST dataset.

As said earlier the purpose of using supervised learning is to give some kind of prediction after being given some input, and this prediction can be discrete values as in predicting if a given image contains a car or a truck or continuous values as in predicting the price of a house.

Regression

It is a type of supervised learning whose purpose is to give a continuous value prediction when given an input, and that's by training the model on the same kind of inputs so it finds a pattern between these inputs and the wanted output.

For example, predicting the price of a house after giving the system the number of rooms the house has, its area, how many floors,... etc.

Architecture of a regression model

Table 6. Architecture of a regression model

Hyperparameter	Typical value
Input layer shape	Shape for one input
Hidden layer(s)	Problem specific, min = 1, max = infinite
Neurons per hidden layer	Problem specific, usually from 10 to 100
Output layer shape	Same shape as desired prediction shape
Hidden activation	Usually ReLU
Output activation	None, ReLU, logistic/tanh
Loss function	MSE, MAE, Huber
Optimizer	SGD, Adam

Brief Description of How a Regression Model is Trained:

1. First the input also known as features is fed into the input layer which has a number of neurons equal to the number of features.

2. Assuming a dense layer is used, each neuron in the input layer is connected to each neuron in the next layer (hidden layer) each of these connections has a value called weight which indicates the strength of the connection, hidden layers could be one or more depending on the problem.

3. The loss is then calculated which describes how far the prediction is from the real value.

4. The optimizer then changes the weights in some way depending on the optimizer used to reduce the loss in what's called back propagation.

Steps in modeling:

1. Getting data ready (preprocessing)
2. Build or pick a pretrained model
3. Fit or train the model
4. Evaluate the model
5. Improve through experimentation
6. Save the model

Important terms:

Bias: Bias in Neural Networks can be thought of as analogous to the role of a constant in a linear function, whereby the line is effectively transposed by the constant value. [16]

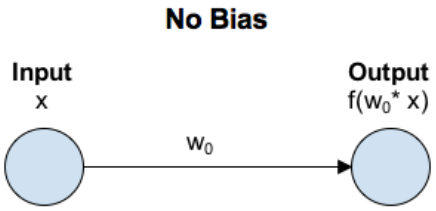


Figure 52. Neural network without biases.

In a scenario with no bias, the input to the activation function is 'x' multiplied by the connection weight 'w0'.

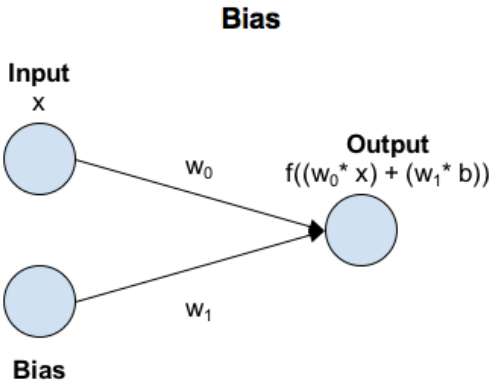


Figure 53. Neural network with biases.

In a scenario with bias, the input to the activation function is 'x' times the connection weight 'w0' plus the bias times the connection weight for the bias 'w1'. This has the effect of shifting the activation function by a constant amount (b * w1).

Depending on the sign of the bias it's used to increase or decrease the range of values that activates a neuron.

weights and biases together are called **learnable parameters.**

Over Fitting refers to a model that models the training data too well.

Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize. [17]

Regularization is a value added to the loss to decrease overfitting

Batch or mini-batch (two terms used interchangeably): is the number of samples passed at once during training. The main reason for why the training set is broken down into batches is that computational resources sometimes aren't enough to pass all of the data at once. [18]

One **epoc** means we trained the model on all of the training data one time.

Transfer Learning: The reuse of a pre-trained model on a new problem is known as transfer learning in machine learning. A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning. [19]

The reuse of a pre-trained model on a new problem is known as transfer learning in machine learning. A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning.

The knowledge of an already trained machine learning model could be transferred to a different but closely linked problem throughout transfer learning.

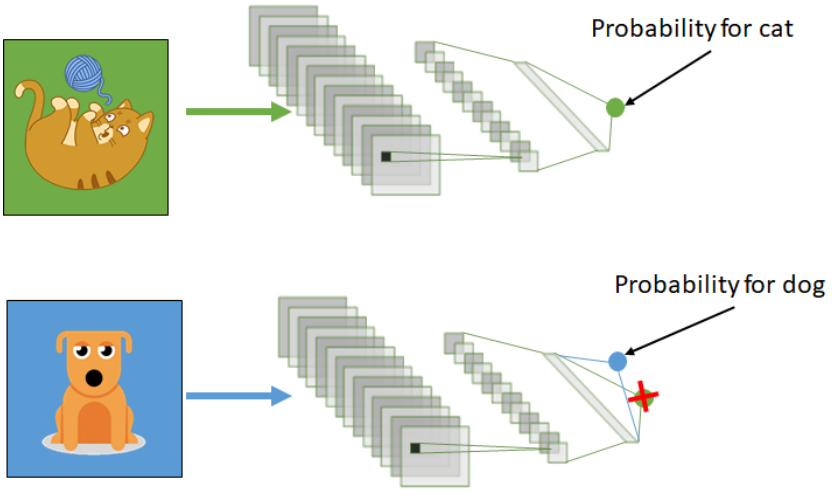


Figure 54. Transfer Learning.

Fine tuning: taking a pretrained model and adjusting the weights and biases based on the problem, some weights and biases can be made frozen meaning it won't be changed during training.

Normalization in deep learning normalization is making the features values range from zero to one

Standardization: subtracting the mean from each data point and dividing it by the standard deviation this forces the data to have mean of zero and std of one.

Classification [20]

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observations into a number of classes or groups. Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories.

There are two types of Classifications:

1. Binary Classifier: If the classification problem has only two possible outcomes, then it is called Binary Classifier.

Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.

2. Multi-class Classifier: If a classification problem has more than two outcomes, then it is called Multi-class Classifier.

Example: Classifications of types of crops, Classification of types of music.

Confusion matrix is a very popular measure used while solving classification problems. It can be applied to binary classification as well as for multiclass classification problems. An example of a confusion matrix for binary classification is shown in Table 7: [21]

Table 7. Confusion matrix.

	Actual Positive	Actual Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

Although it's possible to do image classification using only dense layers but doing so is somewhat of a tedious task since a good performing classification model built using dense layers only will have so many learnable parameters which will make the model converge

slow which increase the need for a bigger dataset and consumes more time, these were some of the main reasons why CNN (or convolutional neural networks) became so common when dealing with images in deep learning.

A convolution layer simply works as following:

1. A set of kernels with a defined size is initialized manually or randomly.
2. For each kernel, each element in the kernel is multiplied by the corresponding element in the image and then the sum is saved in a different matrix.
3. The kernel keep redoing step 2 while slides on all of the image starting from the top left to the bottom right.

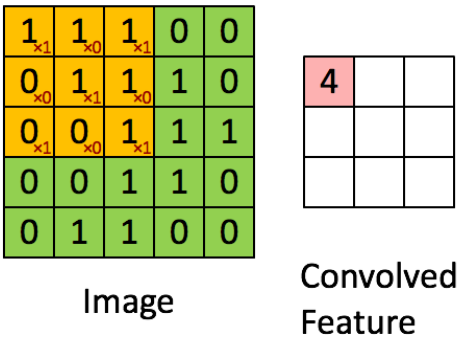


Figure 55. Convoluting a 5x5 image with a 3x3 kernel to get a 3x3 convolved feature. [22]

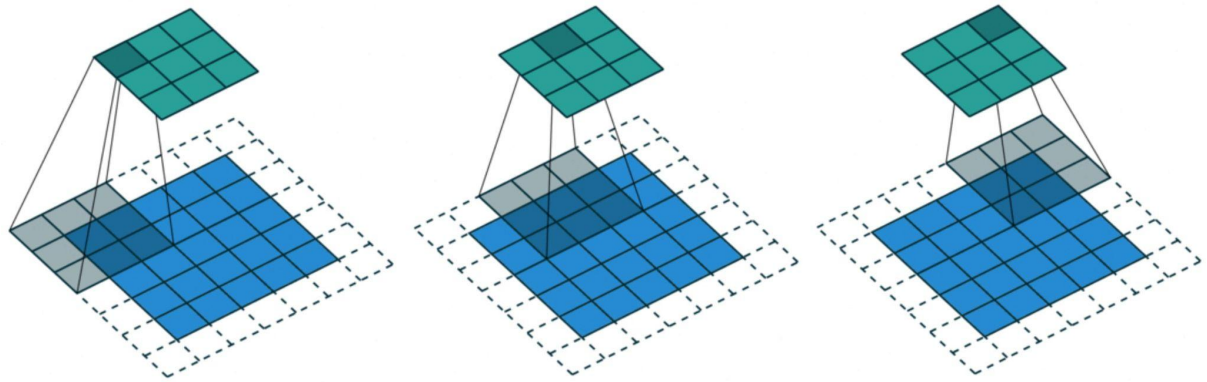


Figure 56. Convolution Operation. [22]

Convolutional layer parameters:

1. Number of kernels: the more kernels there are the more the convoluted features extracted from the image.
2. Size of the kernel: the smaller the kernel size the smaller the size of the features extracted from the image, it's recommended to use bigger kernel sizes in convolution layers early in the model.
3. Stride: stride here indicates the number of pixels the kernel will slide along one direction, When a convolutional layer has a stride of 1 the kernel moves 1 pixel at a time.
4. Padding: padding is the amount of empty pixels surrounding the image while doing the convolution operation. The two most common types of padding are: [23]
    a. Valid padding: which basically means there are no padding pixels added, but all pixels must have a valid value in them.
    b. Same padding: zero valued pixels (empty pixels) will be added around the outer side of the image and that's to make sure that the convoluted feature has the same number of rows and columns as the input image.

Pooling Layers

Usually a convolution layer is followed by a pooling layer, pooling layers could be max or average or min pooling, it works in a very similar way to convolutional except that instead of doing the convolution operation pooling can output the max or min or average value available in the sliding window.

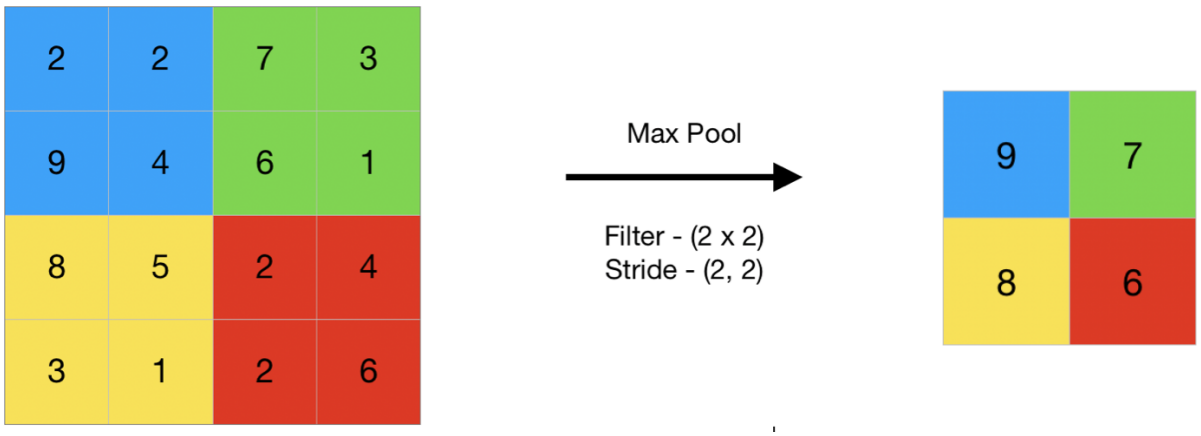


Figure 57. Max pooling operation.

Now that's the theory part is out of the way, the next section will discuss state of the art object detection methods.

State of The Art

The state-of-the-art methods can be categorized into two main types: one-stage methods and two stage-methods. One-stage methods prioritize inference speed, and example models include YOLO, SSD and RetinaNet. Two-stage methods prioritize detection accuracy, and example models include Faster R-CNN, Mask R-CNN and Cascade R-CNN. [24]

Basic structure [13]

Deep learning based object detection models usually consent of two parts:

1. An encoder which is typically a CNN, its purpose is to extract features from the image.
2. A decoder which could be as simple as a regression ANN which gives two outputs (x,y) of the top left corner of the bounding box and bottom right (x,y).

Problems with the aforementioned structure is that it assumes that there's an object of a certain class to exist in the image and only one instance of the object is present.

To solve some of these problems, improvements were made mainly in the decoder, instead of just two elements vector a longer vector could be outputted, look at the figure below, output vector in this example is 7 elements long;

1. Pc: is the probability that any object of any class to exist in the image.
2. Bx is the x coordinate of the center of the bounding box.
3. By is the y coordinate of the center of the bounding box.
4. Bw is the width of the bounding box.
5. Bh is the height of the bounding box.
6. C1 is the probability that there's an instance of cat class to exist in the image.
7. C2 is the probability that there's an instance of dog class to exist in the image.

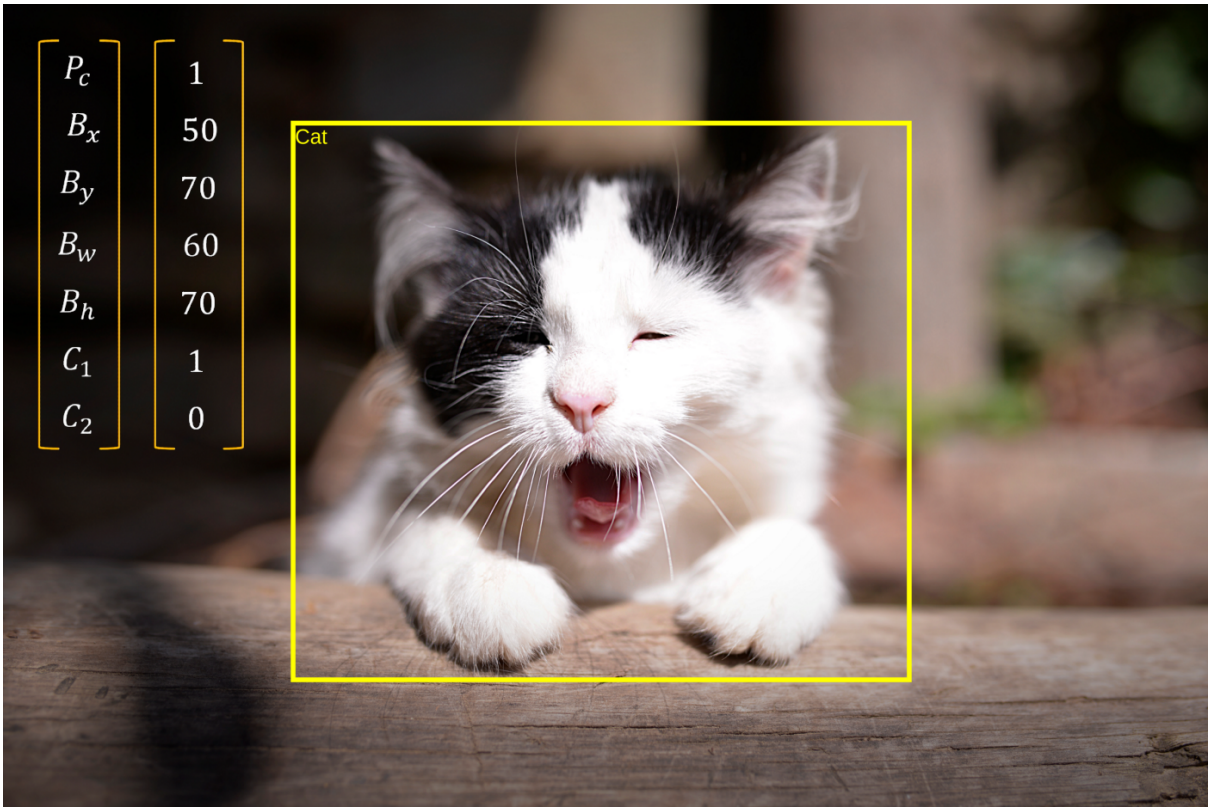


Figure 58. Beyond the basic structure.

Although this method solves the problem of detecting an object from multiple classes within an image, it is still not capable of detecting more than one object.

Sliding Window Method

It's also one of the early approaches to object detection, it's very similar to the previous method except it processes certain areas of the image one at a time until the whole image is covered.

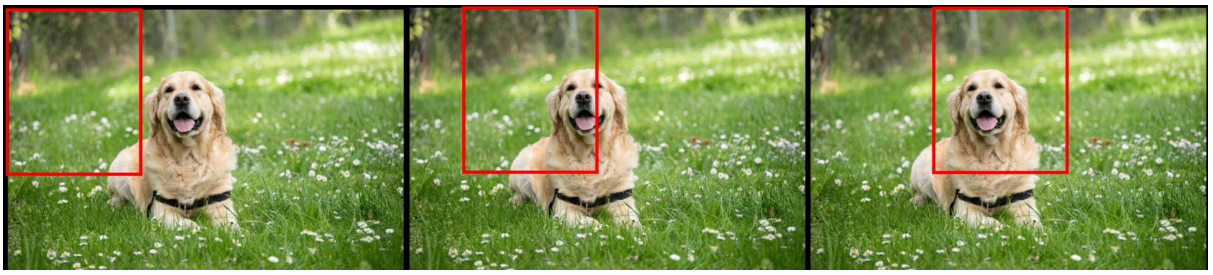


Figure 59. Sliding window method.

Problems with this method:

1. Computationally expensive.
2. Many bounding boxes for the same object.

Two-Stage Regional Based Networks

Let's explain R-CNN as it's the most famous two stage object detection method.

1. First it takes an image as an input
2. It uses some deterministic algorithm to selective search and extract region proposals that are highly likely to have surround an object approximately 2000 areas that are going to be fed into the later step.
3. Resize those areas one at a time to 24*24 pixels and run them through a CNN.
4. Classify those areas to get bounding boxes

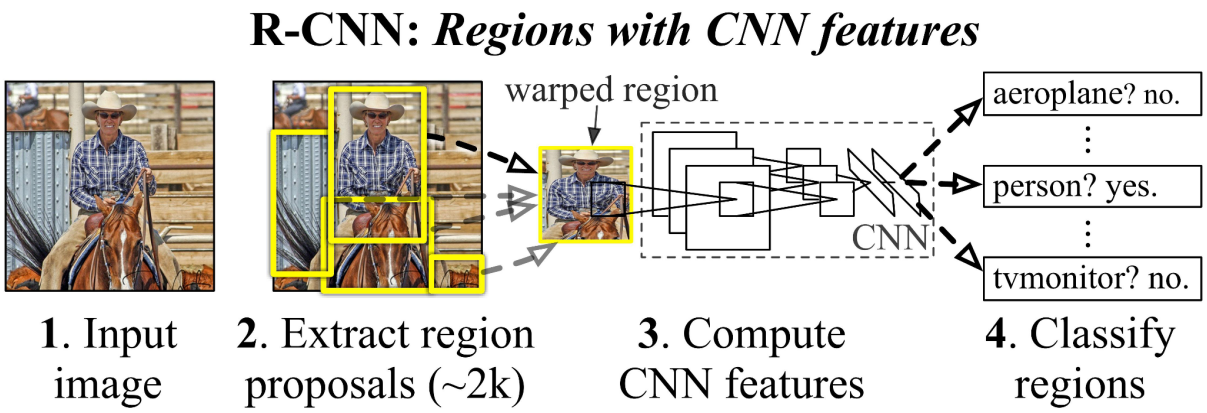


Figure 60. R-CNN method.

Problems solved with this method:

1. There are a lot less areas to search through compared to the sliding box algorithm.
2. There is no need for the model itself to calculate the size of these bounding boxes since the selective search algorithm assumed the areas (their location and size).

Although this method is faster than the sliding box method and although there has been many papers improving this method it's still nowhere near being used in real time applications, this is where YOLO comes in handy.

One-stage Object Detection Methods

Since YOLO (or you only look once) is by far the most popular one-stage object detection method, it'll be explained in this section.

1. It split the image into a grid.
2. Classify each cell into one of the predefined classes and get a vector output similar to the second method mentioned here in this document.
3. Get bounding boxes with high enough confidence.

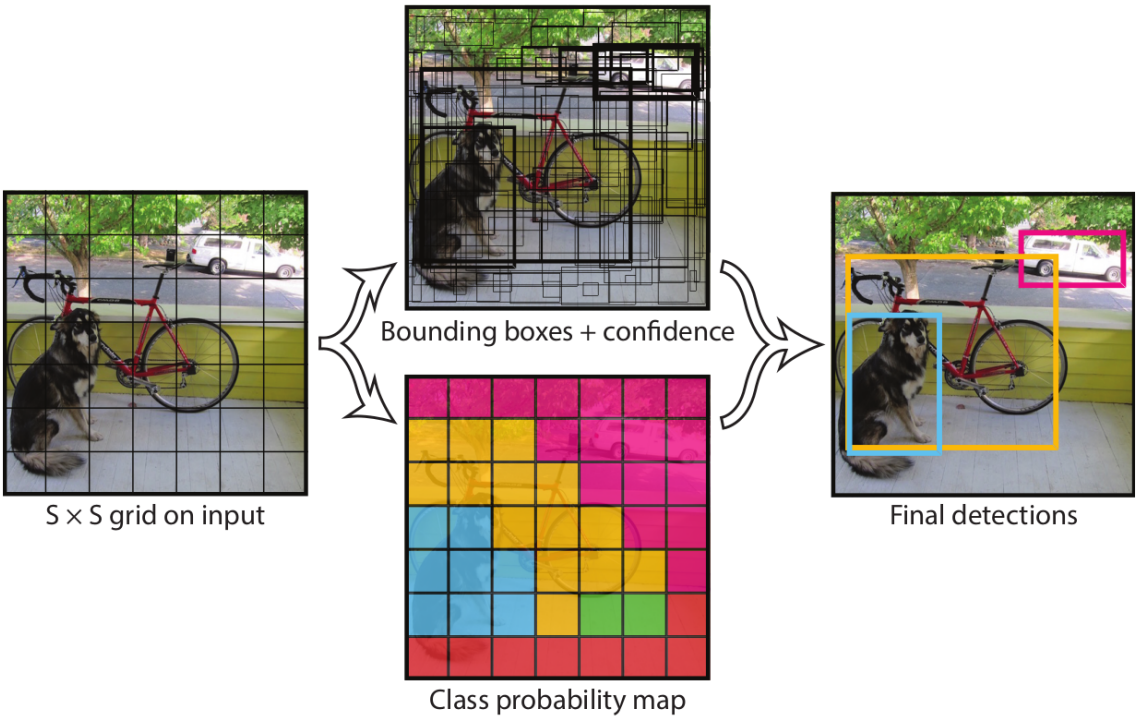


Figure 61 YOLO method.

One Might ask what if there were many bounding boxes for the same object all with very high confidence?

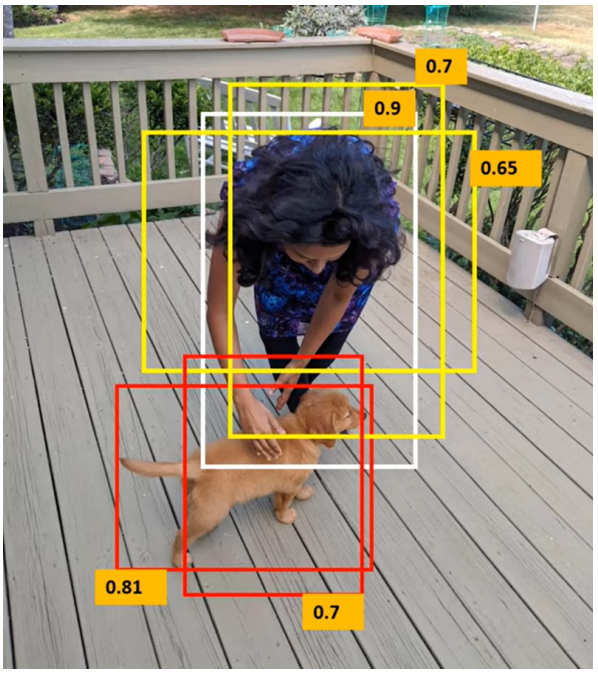


Figure 62. Many bounding boxes and few objects.

A method called Non max suppression is used, for overlapping bounding boxes Intersection over union is calculated:

IOU = intersect area/union area

Then if IOU was above a certain value, only the overlapping bounding box with the highest confidence stays and the rest are discarded.

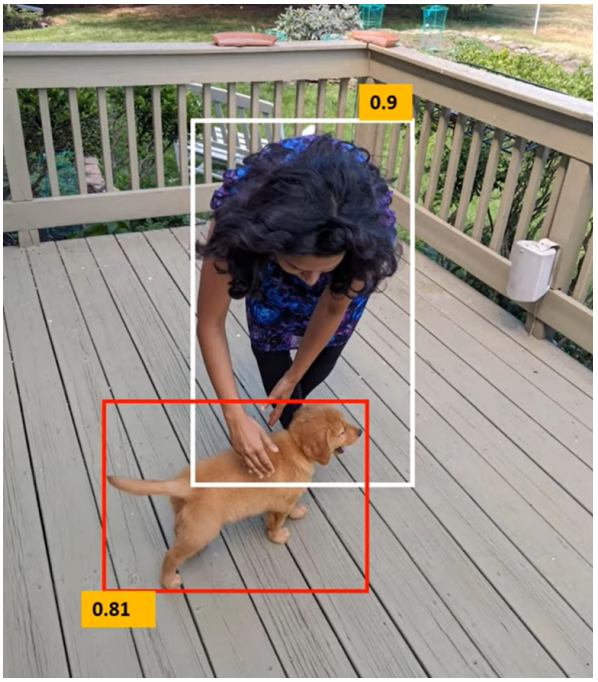


Figure 63. After non max suppression.

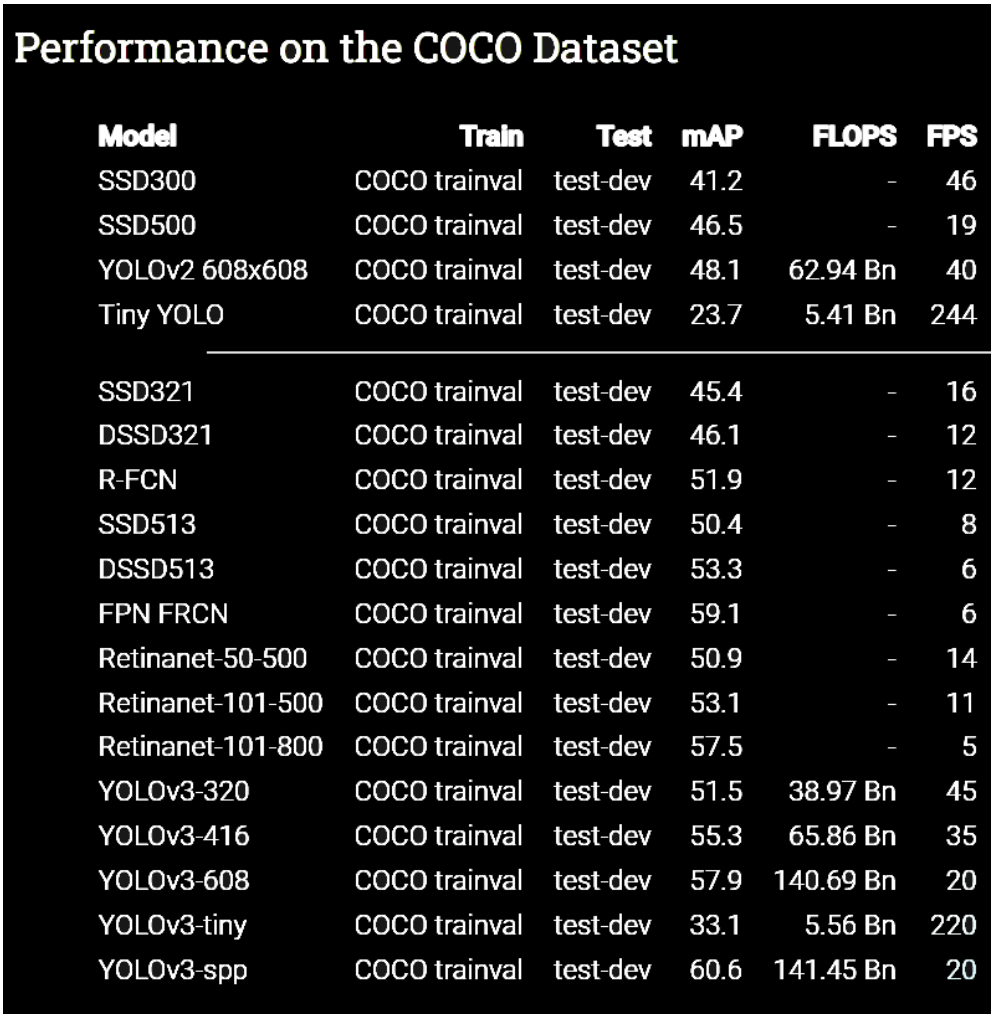
So how good exactly is YOLO?

in 2015 when the original paper for YOLO was published, although it wasn't the most accurate it carried with it some shocking results.

Table 8. YOLO benchmarked against some other state of the art OD algorithms 2016.

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img

Yes it’s not 2015 but YOLO is still getting updated and it is still one of the best object detection methods there are.



Performance on the COCO Dataset

Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20

Figure 64 . Recent YOLO versions benchmarked against other recent methods.[25]

It can be safely said that the YOLO algorithm combines both accuracy and relatively lower computational cost making it a good option for autonomous driving.

Conclusion

When choosing deep learning as a tool in any project the following questions arise: should I build my own model? Should I use a pretrained model and use it as it is? Or should I use a pretrained model and only use some of the early layers and add my own layers and train it some more.

Well, it all depends on the application, although using a pretrained model as it is should be the first option, sometimes the application in which the model is going to be used in is somewhat different than the original model's (the input, the output or both are different to a certain degree) in this situation modifying a model and training it some more is the best option, and sometimes the problem is so unique or when the solution is wanted to be carried out differently the only option left is to build your own model and sometimes your own data too, but it's so time intensive it should always be the last resort.

It's been decided to use YOLO as an object detection model but to estimate how far these objects are from the camera, for each object the average of the lowest 10% distance value of an image's pixels is calculated.

Regarding road lines detection as a future work; deep learning could also solve this issue in a different way increasing speed and resistance to changes in the environment like changing in lighting and the road lines getting dirty.

Bibliography

1. Shoudong Huang, Gamini Dissanayake. “Robot Localization: An Introduction” 15th August 2016, https://onlinelibrary.wiley.com/doi/full/10.1002/047134608X.W8318#:~:text=Robot%20localization%20is%20the%20process,making%20decisions%20about%20future%20actions
2. Sebastian Thrun. “Robotic Mapping: A Survey” February 2002, http://robots.stanford.edu/papers/thrun.mapping-tr.pdf Accessed:
3. Gerard Gibbs, Huamin Jia, Irfan Madani. “Obstacle Detection with Ultrasonic Sensors and Signal Analysis Metrics” January 2017, https://www.researchgate.net/publication/322913643_Obstacle_Detection_with_Ultrasonic_Sensors_and_Signal_Analysis_Metrics
4. Stereolabs. “3-D Point Cloud”, https://www.stereolabs.com/docs/depth-sensing/#3-d-point-cloud
5. Vidya Prabhu. “How Does Depth Perception Work” 19th November 2019, https://www.youngwonks.com/blog/How-Does-Depth-Perception-Work#:~:text=Among%20humans%2C%20depth%20perception%20takes,will%20be%20less%20than%20accurate
6. Wikipedia. “Stereo camera”, https://en.wikipedia.org/wiki/Stereo_camera
7. Kara Gremillion. “Choose The Right Sensors For Autonomous Vehicles”, 9th December 2021, https://semiengineering.com/choose-the-right-sensors-for-autonomous-vehicles/
8. Roderick Burnett. “Understanding How Ultrasonic Sensors Work” 24th March 2020, https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm
9. Motaz Khader, Samir Cherian. “An Introduction to Automotive LIDAR” May 2020, https://www.ti.com/lit/wp/slyy150a/slyy150a.pdf
10. NumPy community. “NumPy User Guide” 22nd June 2022, https://numpy.org/doc/stable/numpy-user.pdf
11. OpenCV. “About OpenCV”, https://opencv.org/about/
12. MathWorks. “What Is Object Detection?” https://www.mathworks.com/discovery/object-detection.html
13. Fritz Labs, “Object Detection Guide" https://www.fritz.ai/object-detection/
14. IBM Cloud Education, “Deep Learning” 1st May 2020 https://www.ibm.com/cloud/learn/deep-learning
15. Zack Brodtman, “The Importance and Reasoning behind Activation Functions” 15th November 2021, https://towardsdatascience.com/the-importance-and-reasoning-behind-activation-functions-4dc00e74db41
16. PICO Knowledgebase. “The role of bias in Neural Networks” https://www.pico.net/kb/the-role-of-bias-in-neural-networks
17. Jason Brownlee. “Overfitting and Underfitting With Machine Learning Algorithms” 21st March 2016 https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms

18. Bipin Krishnan P. “WHEN and WHY are batches used in machine learning ?” 19th Nov 2019 https://medium.com/analytics-vidhya/when-and-why-are-batches-used-in-machine-learning-acda4eb00763
19. Pranshu Sharma. “Understanding Transfer Learning for Deep Learning” 30th October 2021 https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning
20. JavaTpoint. “Classification Algorithm in Machine Learning” https://www.javatpoint.com/classification-algorithm-in-machine-learning
21. Ajay Kulkarni. Feras A. Batarseh “Foundations of data imbalance and solutions for a data democracy” 2020 https://www.sciencedirect.com/topics/engineering/confusion-matrix
22. Sumit Saha. “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way” 15th December 2018 https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
23. Yugesh Verma. “Guide to Different Padding Methods for CNN Models” 4th September 2021 https://analyticsindiamag.com/guide-to-different-padding-methods-for-cnn-models/
24. Paperswithcode. “Object Detection” https://paperswithcode.com/task/object-detection
25. Redmon, Joseph and Farhadi, Ali. “YOLOv3: An Incremental Improvement” 2018 https://pjreddie.com/darknet/yolo/