CZECH TECHNICAL UNIVERITY IN PRAGUE

FACULTY OF MECHANICAL ENGINEERING

DEPARTMENT OF INFORMATION AND AUTOMATION TECHNOLOGY

AUTONOMOUS DRIVING WITH THE HELP OF A WEB CAMERA

BACHELOR THESIS

STEVE SUNIL MATHEWS

August 2020

Supervisor: Doc. Ing. Martin Novak, Ph.D.

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Mathews Steve Sunil** Personal ID number: **464941**

Faculty / Institute: **Faculty of Mechanical Engineering**

Department / Institute: **Department of Instrumentation and Control Engineering**

Study program: **Bachelor of Mechanical Engineering**

Branch of study: **Information and Automation Technology**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Autonomous driving with web camera

Bachelor's thesis title in Czech:

Autonomní řízení s využitím webové kamery

Guidelines:

1) Review sensors suitable for a small autonomous car model
2) Prepare software for image processing from a web camera for obstacle detection. The software can run on a PC and send data to the car control system
3) Design several test tracks for algorithm testing
4) Prove experimentally the functionality of your algorithms, including theirs limitations

Bibliography / sources:

[1] Prateek Joshi, OpenCV with Python By Example: Build real-world computer vision applications and develop cool demos using OpenCV for Python, Packt Publishing (September 2015), ISBN-10: 9781785283932
[2] Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles (Lecture Notes in Control and Information Sciences) ,Springer; 1st ed. 2017 edition (December 2, 2017), ISBN-13: 978-3319553719

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Martin Novák, Ph.D., Division of electrotechnics, FME

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **30.03.2020** Deadline for bachelor thesis submission: **11.06.2020**

Assignment valid until: _____

doc. Ing. Martin Novák, Ph.D.	Head of department's signature	prof. Ing. Michael Valášek, DrSc.
Supervisor's signature		Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt _____ Student's signature _____

Declaration

I, Steve Mathews, declare that this thesis titled "**Autonomous Driving with the help of a Web Camera"** has been solely written by myself and has not been used or submitted before in any levels of education. This thesis has been done while being a student for a bachelor's degree at this University. Any sections of the thesis that has been submitted before or other publications or sites have been stated clearly. I have mentioned all works that have helped me in the writing of the thesis. The thesis is based on the work done by myself and the people who have helped, are also credited for in this thesis.

Abstract

An object detection algorithm is proposed through different methods and ideas to find the right approach. This object detection will be tested on a robotic model of a car which substitutes a real life car on the road, while receiving continuous frames. The proposed program picks up on objects that it is trained to detect and identifies them and shows the name of the object with location by specifying the coordinates with respect to the spatial field of the camera, in order for the maneuverability of the autonomous vehicles. Testing has been done on images and videos and then in real time. Results will be compared between the experiments and a final conclusion will be drawn to find out if the approach proposed, can be used in real life, and discussion about future improvements and areas to correct to make the approach much easier and more accurate to be used in reality.

Keywords: Object detection, Computer vision, Autonomous vehicles, Sensors

Acknowledgements

First and foremost, I would like to thank God for helping me complete this thesis and gain knowledge about the topic, even though it was outside my comfort zone. Secondly, I would like to thank my family for the support and encouragement they have provided me during this special and stressful circumstance. Special thanks to my supervisor, Doc. Ing. Martin Novak, for first of all helping me finalize on this topic, directing me to my objective, telling me what steps I have to take or what to learn and being available for consultation and helping me with the experiment ideas and for completing this thesis. Lastly, I would like to thank my companions for being there till the end and encouraging me in this period. I would also like to thank professor, Matousc, for his insight into the topic.

Table of Contents

List of Figures

AUTONOMOUS DRIVING WITH THE HELP OF A WEB CAMERA

1. Introduction

A couple of years ago, people thought that getting a computer to distinguish between a cat and a dog would be almost impossible, even with the advance of technology. Now with the present level of artificial intelligence, it is possible to do it at a level greater than 99 percent accuracy and it is called as image classification, for starters. Given an image, it is possible to identify the number of objects in an image and place a label on them, identifying what they are, which further on, works also on videos or real time video, which is also know an object detection.

One of the most sought-after inventions in transportation is the concept of autonomous driving. Cars equipped with this technology will have their own benefits. They will likely reduce crashes, energy consumption, and considerably less pollution. In the past decade, with the increasing interest in this field from companies such as BMW, Audi and Tesla to name a few, autonomous driving has gone from "assumptions and ideas on paper" to "definitely possible" to creations which can be seen road legal in the coming years. In December 2018, Waymo, the company that emerged from Google self-driving-car project, started its own commercial self-driving-car service in the suburbs of Phoenix [1]. An autonomous truck can be driven nearly 24 hours a day, whereas human-driven vehicles must take breaks for the driver taking rest, which thereby reduces efficiency and time taken for transportation and less strenuous on the human body and mind. With self-driving cars (SDC), the goal is to be able to operate a car normally, but without a human driver. From a user perspective, safety and convenience are always a major concern and also new vehicles should enable people to drive, even those who presently can't, due to the aging population. Current readings estimate that by 2025, we will be able to see over 600,00 self-driving cars on the road, and the number will keep jumping [2].

Using systems of cameras, lasers, radar, GPS, some cars can:

-Assist in parking

-Maintain speed related to vehicles or objects in front

-Reduce the risks of accidents

For example, Tesla cars can park themselves and can come back to receive you from shopping, all by using your smartphones.

2. Review

Autonomous driving is a very controversial and complex advancement in technology for many people. So, in order to simplify it, it is crucial to figure out how it works, as well as the different kind of sensors incorporated which help them to know where to drive and to perceive different objects on the road in order to prevent casualties. This technology can decrease transportation cost in the long run and increase accessibility to people with mobility issues and low-income households, if the technology is developed to the point where it is a necessity. Each component that is included in AV's (automatic vehicles) are a subject of intensive research in various fields and is also considered as a mix of many fields such as transportation, electrical engineering, IT, software and hardware engineering. [13]

The first attempt towards autonomous driving can be dated as far as the early 1920s [14].

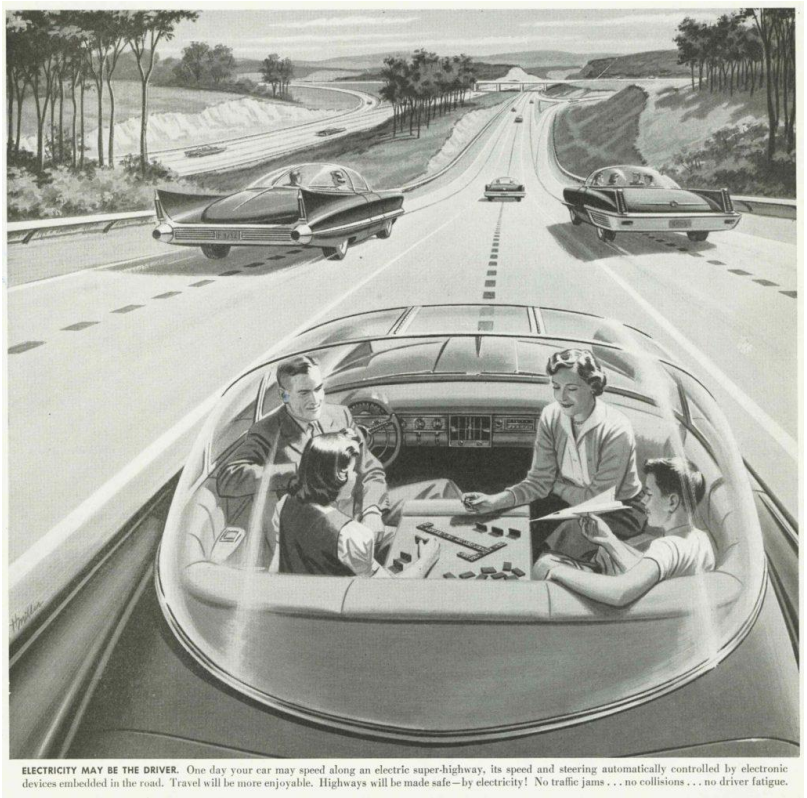

Figure 1: Driverless car of the future as envisioned on the Saturday Evening Post, 1950s. [14]

Though transportation is a means of development of the society, it is also coupled by negative issues such as pollution, accidents, costs and human casualties. AV technology is largely perceived to eliminate these existing negative impacts.

2.1 Literature Review

From the bibliography mentioned, there are two books which kickstarted this thesis. First the book on the libraries or modules that I would be focusing on and how to practice computer vision applications that are needed.[16] The second book depicts the use of these sensors or autonomous vehicles in different fields and their benefits with a peek in the mathematical theory and practical applications. I shall stick to the use of sensors for land vehicles or more preferably, road legal vehicles. Small autonomous vehicles can hold many small sensors but the capability to process different sets of data from all the sensors and combining them or 'fusing' them to one complete data, from which the vehicle make a decision is based on the processing power and different constraints that come with each sensor. To confront these questions, I looked at it through the biological point of view.[17] There are animals present that use two types of sensory inputs and move about. For example, the bat. Most of them use their vision and sonar. During low lighting, sonar proves to be capable in enabling them to catch their prey. One would understand the basis of sensor fusion, but in my case, it is assumed that the testing car can understand the data it receives from other sensors. Also, I shall focus on one of the sensors, which is the camera or the vision of the autonomous car itself. Visual sensors can amass huge amount of information through images or videos and hence used in machines or as robotic visions. Many applications that involve the use of cameras include vehicle tracking, monitoring the environment, inspecting infrastructure and many other applications that the involvement of human life just deems to be risky. [17] Numerous challenges are involved in the developing an object tracking system, which one of them is object tracking frame to frame, which relatively has a high error rate. In each frame, object measurements keep changing at a fast rate which would lead to errors in measurements. Also, the applications that are needed by the system varies due to the objects detected or the environment placed in. In this paper, we shall be choosing the right combination of applications or files required and develop an algorithm which is used to detect objects which are commonly present on the road and to perform with high accuracy in detection and with capable speeds.

2.2 Levels of Autonomous Vehicles

Ranging from assistance of drivers to fully autonomous cars, there are five generally accepted levels of self-driving cars. These levels have been developed by the Society of Automotive Engineers (SAE) International's standard J3016 and it depends on the level of human involvement in driving.[11] There are six levels actually, but Level 0 is basically full human involvement and no automation at all.

These are the levels of autonomous vehicles that are classified globally [4].

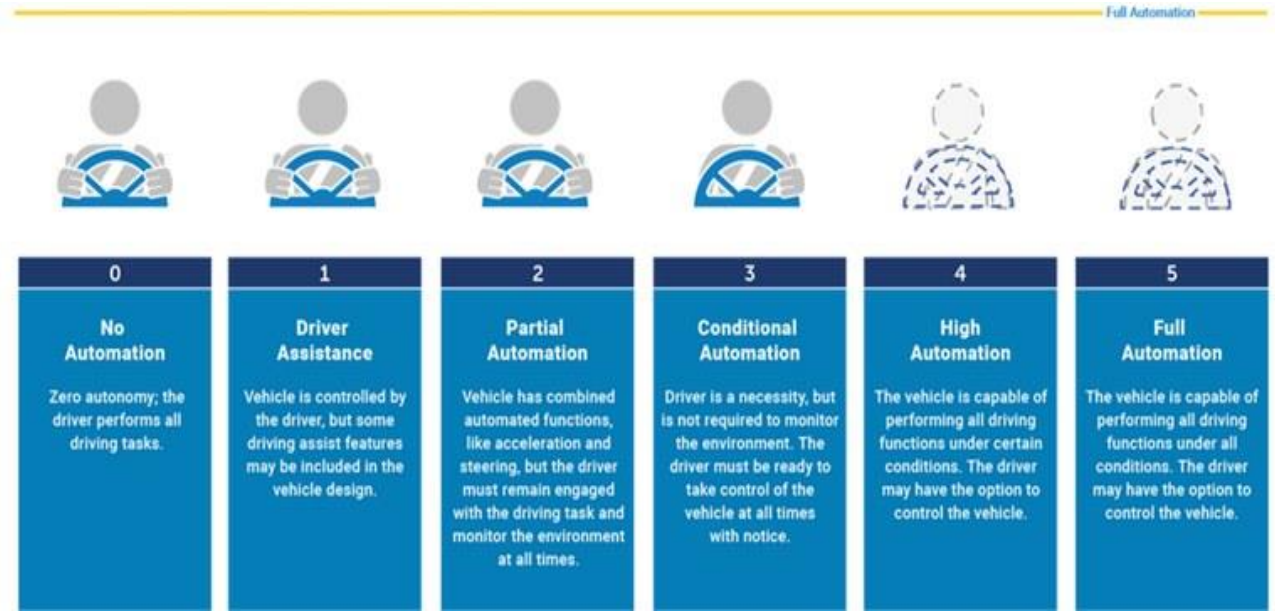

Figure 2: Classification between each Level. [4]

Level 0 – No Automation

This is full human dependent. One can accelerate, brake, steer, and go through traffic without and help from any technological devices. For example, if a car comes at you, you alone must decide how to react and proceed in a safe manner.

Level 1 – Driver Assistance

Adaptive cruise control and lane assist are present to help with driving fatigue. Adaptive cruise control helps keep safe distances between cars ahead using radars and/or cameras. Lane assist helps in bringing

the vehicles back to respective lane, if veered off a bit. 2018 Toyota Corolla and 2018 Nissan Sentra are examples of Level 1 autonomous technology.

Level 2 – Partial Automation

Level 2 automation assists in controlling speed and steering. It will help with stop-and-go traffic by maintaining distance between vehicles, while also providing steering assist. Tesla Autopilot and Volvo Pilot Assist are some of the examples.

Level 3 – Conditional Automation

Level 3 automation can drive themselves, but with limitations, such as limited access highways at a certain speed. Although hands are off, drivers are still required behind the wheel. 2019 Audi A8 is an example of Level 3 automation. The difference between Level 2 to Level 3 and above is that the vehicle is capable of monitoring its environment (LiDAR).

Level 4 – High Automation

Level 4 autonomous vehicles can drive themselves without human interactions, though regulations and obstacles slow the availability of this technology. The autonomous driving system would still notify the driver if the conditions around him are safe, and then only would it let the person switch into this mode. Waymo have developed and tested Level 4 vehicles in Arizona and have developed apps similar to Uber and Bolt but have autonomous cars driving them to their destination.

Level 5 – Full Automation

At Level 5, we reach true driverless cars. Level 5 are capable of monitoring and navigating through all road conditions and require no human interventions, eliminating the need for pedals and steering wheel. Though there exist artificial intelligence, that makes Level 5 possible, due to regulations and legal battles, it is still a work in progress. NVIDIA have announced an AI computer to help in achieving this level of autonomy, where the drivers just have to input their destination and sit back, which the car drives itself. [10]

2.3 Sensors Used In Autonomous Vehicles

To drive better than humans, autonomous vehicles must view the environment and process information better than humans. Object detection belongs to the core abilities of an autonomous system as they are required to perceive the surrounding environment (Computer vision). In computer vision for SDC, the goal is to be able to identify objects near the car, which is done by using image classification network called Convolutional Neural Networks [3]. To identify objects in an environment, SDCs use combination of sensors to achieve high accuracy and efficiency, without which autonomous vehicles would be impossible.

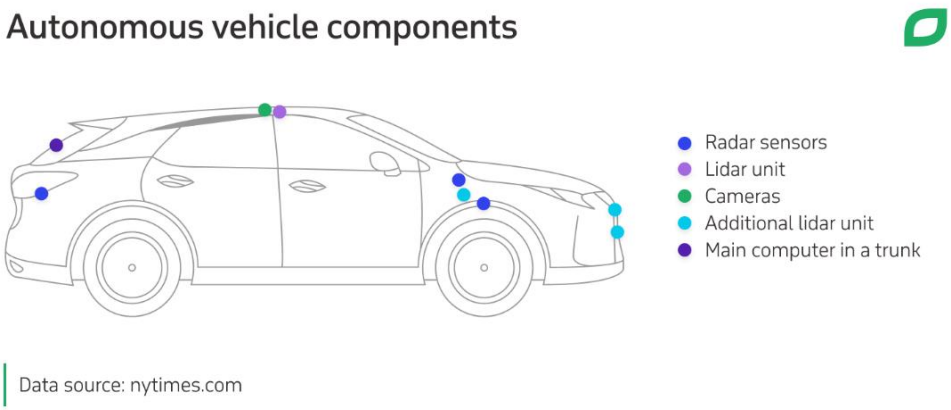

Figure 3: Location of sensors. [11]

Manufactures are generally faced with three primary sensor types -cameras, radar or lidar. When these sensors are coupled with a computing system, the vehicle can then attempt to map, understand and navigate through the environment. The primary sensors are also coupled with secondary sensors which increase their range. New sensors are also developed to maximize the potential of autonomous driving.

Figure 4: Different sensors used.[6]

a) Radar Vision

Radio detection, known simply as 'radar', was first developed before Second World War. It has been used to accurately locate the position, speed and direction of planes, boats and other moving objects. Radar works by firing radio waves at a target area and monitoring for reflections from any objects. They are crucial to the overall attribute of autonomous driving as they can gauge the distances and speed of the objects detected in relation to the vehicle in real time. There are two types of radar sensors used, based on their ranges. Short range (24Ghz) are used for maintaining lane-keeping assistance and parking aids. Long range (77GHz) uses include distance control and brake assistance. But unlike the camera sensor, they do not have an issue in functioning during fog and rain.

Today, it has been deployed to help keep our roads safe, with many modern cars using radar sensors for hazard detection and range- finding in features like advanced cruise control.[6] Radar sensors can aid camera vision in times of low visibility, like driving in the night and improve detection for self-driving cars. They are able to determine speed and distance; however, they can't distinguish between different types of vehicles.[7] Radar sensors correctly identify 90% to 95% of pedestrians, which is not enough to ensure safety on the roads. The widely used 2D radars are not able to accurately determine an object's height as they scan only horizontally, which can cause issues. 3D radars are currently being tested to combat these issues.

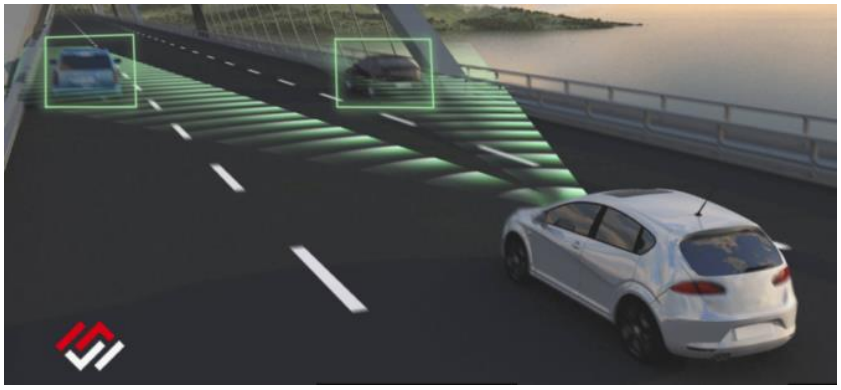

Figure 5: Radar sensor in use in the above picture. NVIDIA DRIVE partner Metawave delivers highly accurate radar sensing technology. [7]

b) LiDAR

Camera and radar are common sensors used in cars today. However, for full driverless capability, lidar, which stands for light detection and ranging, is a sensor that measures distances by pulsing lasers – that has proven to be incredibly useful. Lidar makes it possible for self-driving cars to have a deeper depth of view of their environment. It provides shape and depth to the environment it is scanning and can be configured to give a full 360-degree map around the vehicle. It also works just as well, in low-light conditions, like radars. Advantages which make Google, Uber and Toyota choose lidar systems. [11]

Vehicle only need lidar in a few key places to be effective. However, the sensors are more expensive to implement –10 times the cost of cameras and radars as rare earth metals are needed in order to produce lidar sensors and takes up a huge amount of processing power to process millions of measurements per second. [7] They are also very complex built up of many intricate moving parts, which are vulnerable to damage. [6] During snow and fog, the sensors can be blocked and can affect their ability to detect objects. [11]

Figure 6: Visualization of a Velodyne lidar sensor detecting objects with laser pulses.[7]

c) Camera

Cameras are reliable, cheap and easy to produce and are a widely understood piece of technology. Most of the vehicle applications that use cameras today include advanced driver assistance systems (ADAS), surround view systems (SVS) and driver monitoring systems (DMS). Coupled with infra-red lighting, they can perform to some extent at night also. Thermal cameras are mostly used for detecting people in the day or night (Tesla are mostly known in this field for their mix of cameras and radars, as Tesla CEO Elon Musk is not a LiDAR fan, as tesla vehicles don't have LiDAR and rely on radar, GPS and other cameras and sensors.) [5].

Equipping cars with cameras, the vehicles can maintain a 360-degree view of the environment, thereby giving a broader picture of the traffic ahead. In the present time, 3D cameras are available and are used for producing highly detailed images. For example, cameras can easily identify between cars, pedestrians, signals, bridges and other road markings.

But as reliable, cheap, and easy to produce, cameras often face limitations as we find with the human eye. They need a clear lens to see properly and they don't always give a crisp or reliable picture in bad weather. At night, they are as good as the vehicle's head lights, reducing their accuracy. Also, there are situations where the images produced are just not good enough for the processor to make a good decision about what is the next step to take. For example, when the color of the objects detected are similar to the background or 'camouflaged', the algorithm can fail. It also requires amount of training and processing power to use a camera for object detection with pre-set values, two factors which are inherently limited with an in-vehicle computer system [6]. Hence, cameras are mostly used as a fusion

with other sensors such as LiDAR. We shall be focusing on using the camera as our sensor for object detection.

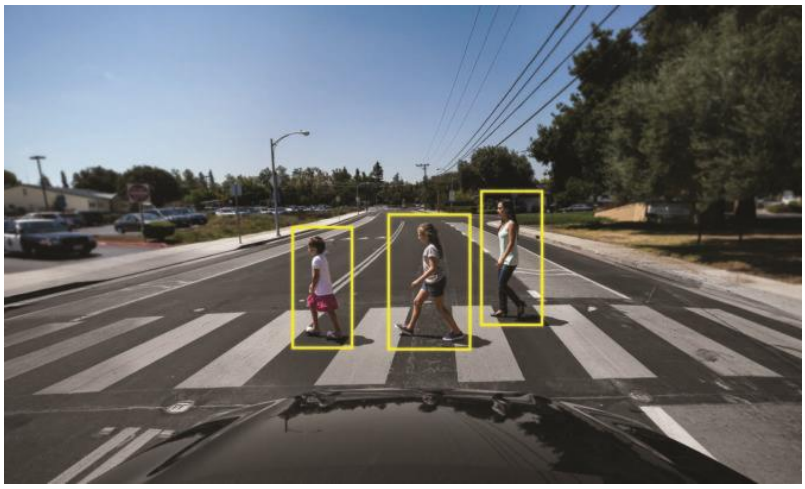

Figure 7: An autonomous vehicle uses camera data to perceive objects in its environment.[7]

d) Ultrasonic Sensors

Ultrasonic sensors are one of the secondary sensors which are used in combination with the primary sensors. Vehicles often use these sensors to detect obstacles in the immediate vicinity. They play a vital role in automated parking and we also see them in use but in small roles in present cars.
This technology imitates the navigation process of bats which emit ultrasonic waves to map out their surroundings. Likewise, these sensors send out sound waves and when they hit an object, they produce echoes, revealing the exact location of that object. By measuring the time, it takes for the echoes to arrive, the sensors measure the distance between the object and the vehicle. This type of sensor is present in the front and rear bumpers of the vehicle. The sensors relay information back to the driver, usually in the form of audio beeps, to reduce accidents and making parking that much simpler. They have a range of almost 2 meters.
Drawback with this sensor is that it can only be used at very low speeds. They are used for parking which is their intended purpose and at the moment there is no further development required, given with its existing purpose.

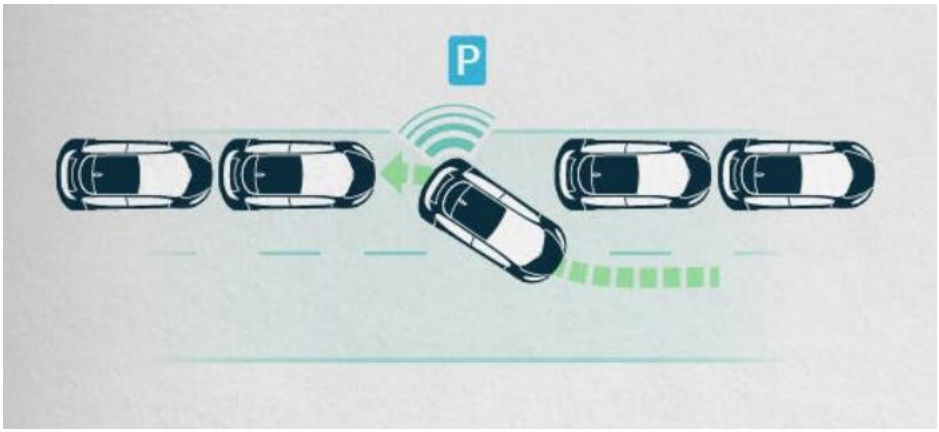

Figure 8 : Usage of Ultrasonic sensor in parking. [24]

e) Cloud

This type of "sensor" is a more advanced version which is basically a combination of sensors running together to receive data which is then stored in a place or cloud and this data can be drawn upon by any car and at the suitable location. It offers highly accurate real-time map data which is constantly being updated by the collective intelligence of the vehicles. For instance, reporting closed lanes or defective traffic lights. This data will be stored and will be used by other vehicles to plan their destination much easier and will avoid more vehicles stopping at closed lanes, giving them a fore warning and avoiding it. A vehicle can adjust to an upcoming traffic jam early on. This would decrease traffic and casualties and make driving more organized. It is a type of sensor providing the vehicle with an image of its surrounding. The vehicle's other sensors have a maximum reach of 250 meters. The cloud data basically allow vehicles to better anticipate what's ahead.

This is in its prototype stages which are being tested, but the map data is not yet precise enough for highways or rural areas. Using this kind of intelligence of vehicles can only be possible if there is a sizeable number of connected cars on the road.

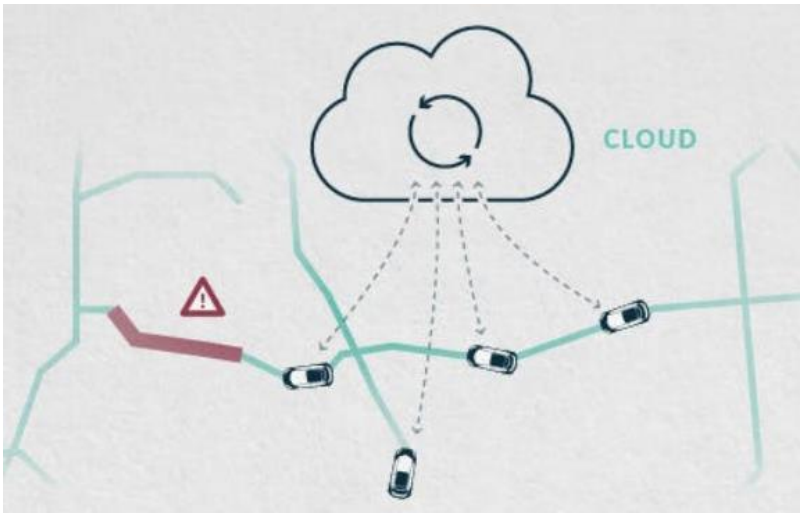

Figure 9 : Cloud technology being used. [24]

3. Goals to meet

This paper aims to answer the following questions:

1) Review sensors suitable for a small autonomous car model.
2) Prepare software for image processing from a web camera for obstacle detection. The software can run on a PC and send data to the car control system.
3) Design several test tracks for algorithm testing
4) Prove experimentally the functionality of your algorithms, including their limitations.

At the end of these tasks, the program coded would be tested on the car model and the results will be taken as the basis for object detection and determine how functional the program is.

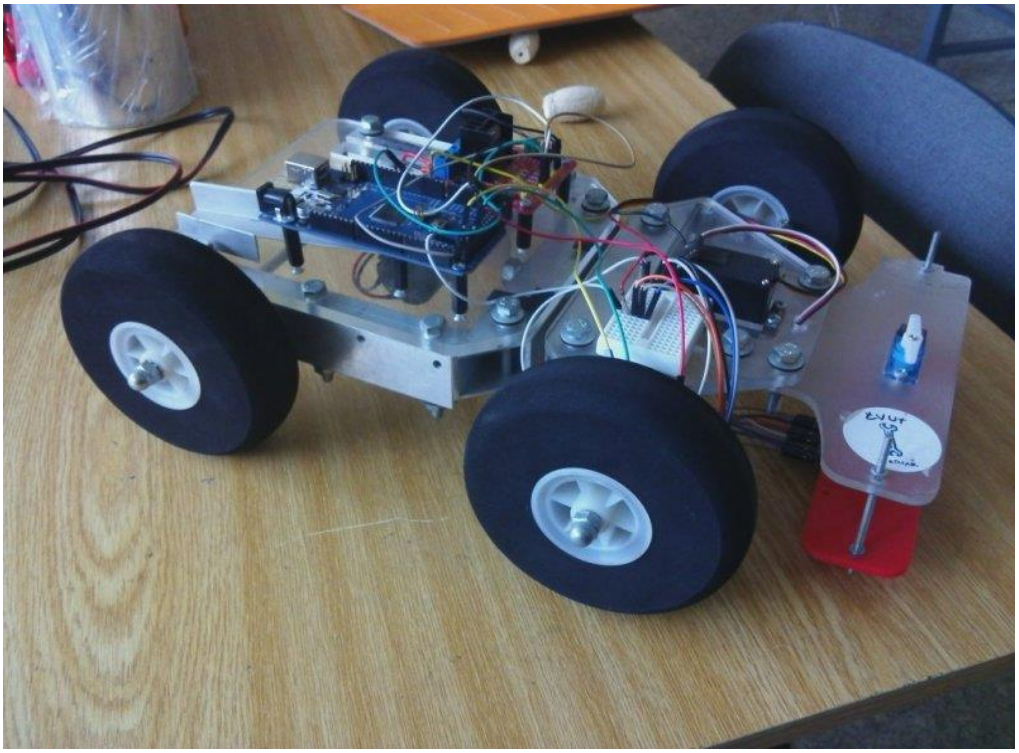

Figure 10 : Robotic car model which will be used for the experiment. [30]

4. Used methods

To proceed with Object detection from scratch, it would require coding language and specific libraries which makes object detection simple yet accurate. Low level languages such as C/C++ are native languages for heavy algorithms and are used to write other machine languages, but it is way too cumbersome. Other languages such as Python and Java can also be used and since Python is a high-level language, easy to understand and is used for image recognition because it has a lot of supporting libraries which make the task easier, it has been used for this thesis. Object detection, whether performed through deep learning or other computer vision techniques, builds on the root of image classification and finds out where in the image an object is located. Current state of the art object detection is due to the advancement of the neural networks. Currently, it can detect and recognize numerous objects and it is possible due to the possibility of training the program to detect the objects. In our case we are interested in the accuracy and speed of the program.

Next step are the libraries in Python which are used for this task. NumPy is one of the libraries in Python programming and provides support for arrays. An image is basically a standard NumPy array containing pixels of data points. By using basic NumPy operations, such as slicing, masking and making an array of zeroes, one can manipulate the pixels or create patterns. As it is a module or a library it can be installed into the python compiler using the command window present in the python compiler by using pip.

```
pip install numpy
```

To test if the if the library in correctly installed, we can import the library to use it in a sample code.

```
import numpy as np
```

Next module is the OpenCV package. It is very powerful and this useful Computer Vision library enables users to build Computer Vision models. Its focus lies on real time Computer Vision and its implementation. Using this module, one can read and write images, capture and save videos, process images, detect specific objects or analyze the video itself. This module can also be installed using the pip command and can be imported to see if it is successfully added in.

```
pip install opencv-python
```

As discussed, object detection is based on the principle of image classification. The difference in code between detecting objects in an image from a video or an image is that, an image is loaded and the computer in not required to record any frames, which is done using a loop. The first step is to try identifying objects in an image. There are many approaches to identifying an object. The first step was to identify an object is the base on the color. As humans, we perceive color by its attributes of brightness, hue and colorfulness. A computer identifies the color using the amounts of red, green and blue color required to match the color. Hence, we have made a trackbar which can be manipulated to track any combinations of the HSV space to obtain the necessary color. Here, as the computer records real-time frames, the frames are converted to HSV plane. In HSV, it is easier to represent a color than in RGB color-space. Steps are:

-Take each frame of the video

-Convert from BGR to HSV color-space

-Thresholding the HSV image for the range of a color, which in this case is red.

-And then extracting the red object alone and presenting it on another window.

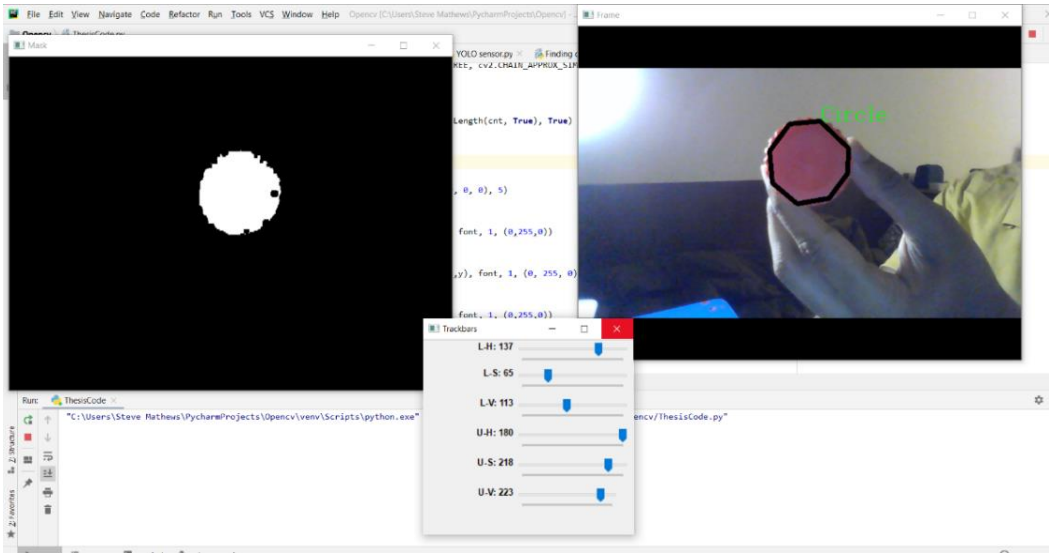

Figure 11: Detection of red objects using HSV space

Here, one can observe that it converts color-space and based on the HSV value of red, it draws contours over the shape of red and depending on the number of contours, it prints out a shape, which in this case

is a circle. The benefit of this approach is the resistance to noise and any spatial deformations. The advantage of improved recognition rate works out only for images, as there is no movement or extra illumination, whereas for moving objects, illumination from the sun or any light source can change the amount of color of the object being detected, which we can see below.

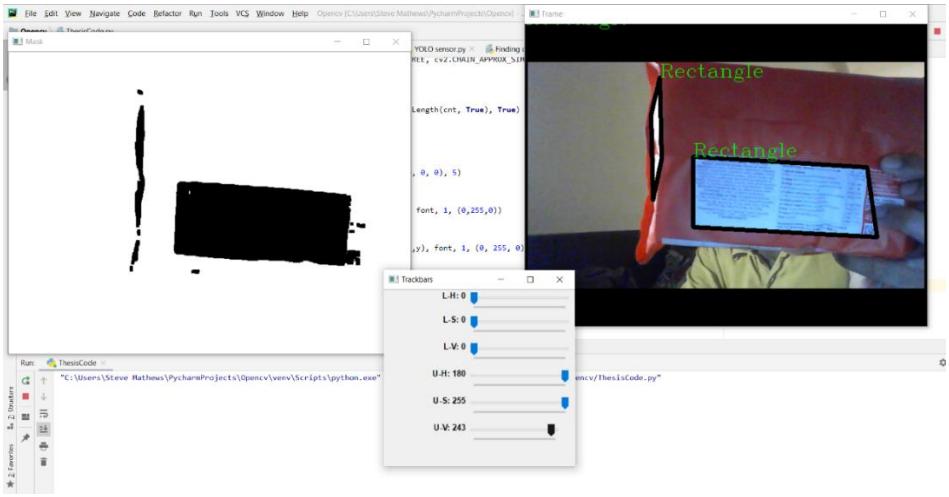

Figure 12: Detection of shapes using contours

Here we try identifying the white rectangle box on a packet of biscuits. As seen above, it is not very accurate and not reasonable enough to implement this approach for autonomous driving. Due to the presence of light, other surfaces which are not the color of white gets fader and might appear white to the camera as seen as above and will get detected. Also, in real time, there will be a multitude of different color objects to identify from and it isn't possible to just focus on one color at a time, and specific HSV color value combinations are required to identify a particular color, whereas in real time, there will be many shades of one color itself.

From the above trial, color is not the ideal criteria to solely focus on. Instead of the color, we can focus on the edges of an object or in other terms edge detection. This method extracts the objects and identifies it in terms of edges. Now using this method is advantageous than the previous method as it is not affected by illumination conditions or any variations of the color on one object. They also represent the object boundaries with which, the information when passed on can be used to find out the area of the object detected to drive the car away from the objects in order to avoid collisions.

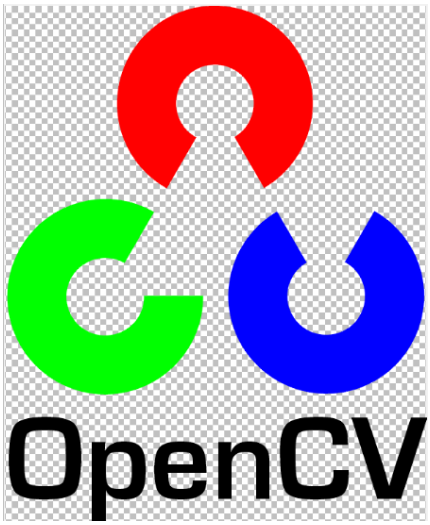

Figure 13: The trial image used for edge detection. [28]

```
"C:\Users\Steve Mathews\PycharmProjects\
Number of contours=9
```

The number of contours are displayed above, about that specific image.

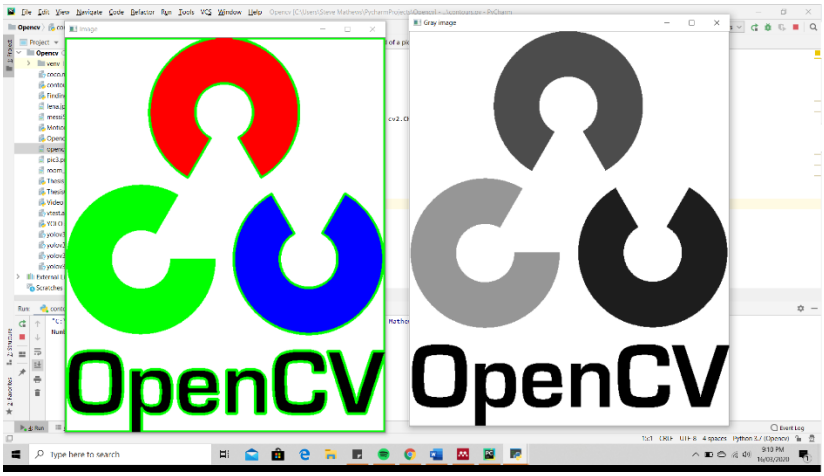

Figure 14: The contours are drawn over the shapes and letters.

Drawback is the labelling and the number of objects in the environment. We can't see the use in the necessary task at hand that is autonomous driving. On video, due to the blur present it cannot accurately detect roughly all the edges and would draw its own edges based on its detection, though there are no objects present in that detection. As in the previous approach, we could use trackbars to focus on one color to reduce the noise, whereas for edge detection there is no manipulation as it directly focuses on the edges. Also, edge detection gives us the correct shape of the object. For example, when driving, the

car in front gets detected through this algorithm. The algorithm can be manipulated to give the spatial area of the object. But the area that would be displayed would be precise and therefore the car cannot use this for avoiding cars as there would not be a margin which can be accepted for the car to have an idea of how much to move to the left or right. Bounding boxes can be used for this method. Finding the central point of the object and then placing it under a box, with which the area would be sufficient for the car to make a decision but would make the edge detection pointless in this field. Since we are trying to build an object detection software to be used for driving, motion of the objects can be used as a feature to be detected, though there is a drawback to this approach as discussed below.

The next approach is based on the area of the objects, bounding boxes are drawn over these objects, identifying them from a video and tracking them till they're in frame. Now as it is a video, where the frame is recorded continuously, the Gaussian blur will be applied to help reduce the noise. As the frames are being recorded, we apply Gaussian blur to one and find the difference between the video for edge detection. On the main frame, we apply the boxes and reveal the status of motion of objects detected.

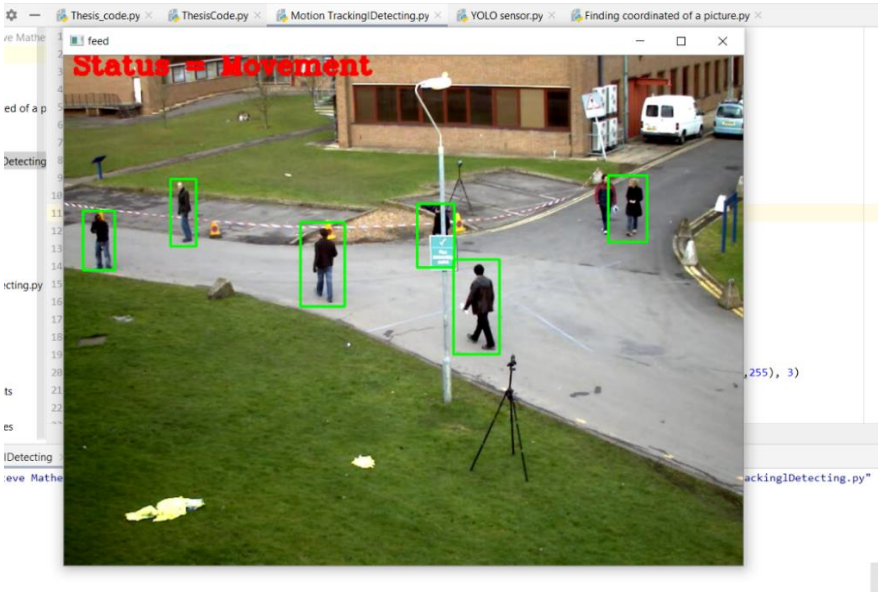

Figure 15: Detection of moving objects in a video. Video taken from the web.[29]

As shown above, boxes are placed on the people and it tracks them till they're on the video and on the top left the status of the object is shown. The drawback to this method is, it only tracks moving objects. As we can see the lamppost, the cars and the tripod stand are not detected as they are not moving.

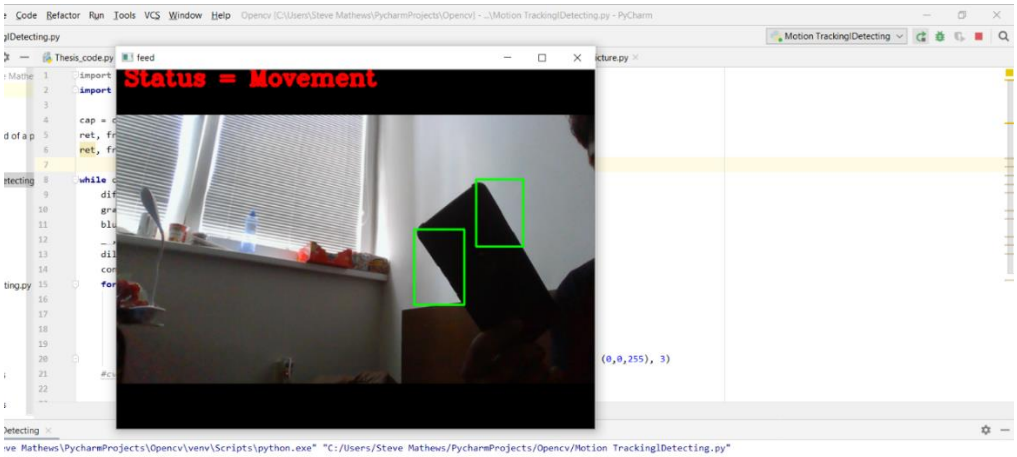

Figure 16: Detection of phone when moving it .

As the phone moves, it gets detected and status is shown.

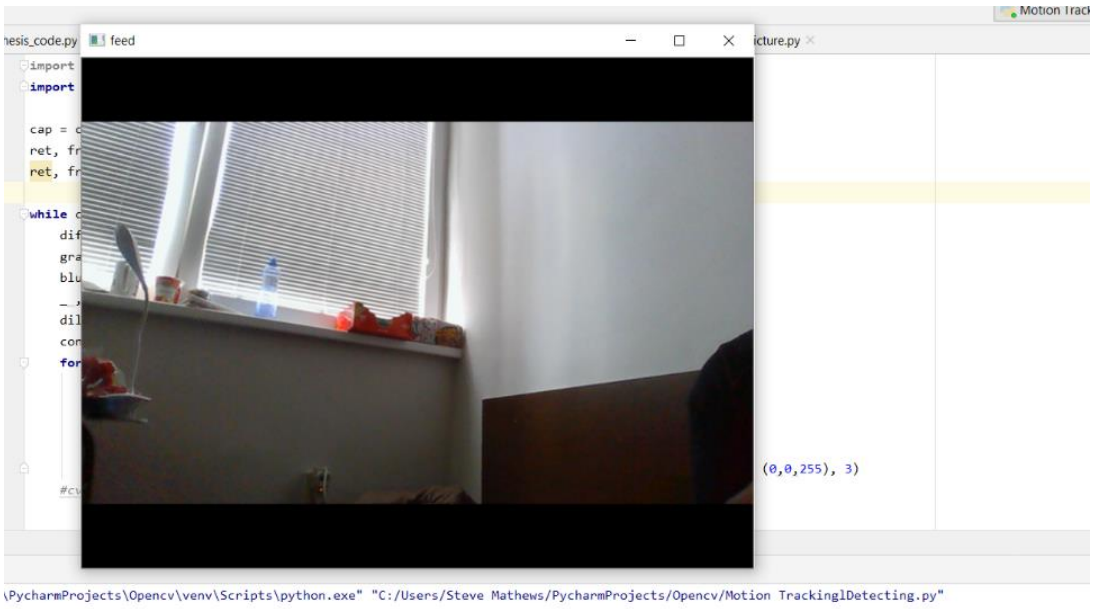

Figure 17: Detection does not take place, as there are no moving objects.

Here there are no moving objects, hence no detection and no status. This method will also not work, as we must detect even stationary objects such as traffic signals or the stop signs and so on and it also necessary to show what object is detected so that we will know what object it is. But we can use its concept of bounding boxes and try 'training' it to identify necessary stationary objects.

In summary, the program would need to detect any or all objects whether moving or stationary, with their names and size of the objects so as to send this information to the processor which in turn can maneuver the car away to avoid obstacles or stop at a red light and so on.

For this level of object detection described, we use pre-set weight files and configuration files, which are already trained to detect objects and shows the labels of the objects. The principle is the same, we record frames and identify objects. We first read our configuration files and load it into the code, and using these files, the program immediately identifies the objects and places a label on it and how much "confident" it is, what it describes it as. We will be using YOLO files (You Only Look Once) proposed by Joseph Redmond [9], as our base files and try reading it from the darknet. Darknet is an open-source framework that supports Object Detection and Image Classification tasks written in C/Cuda. It is a framework to train neural networks. It is mainly known for the implementation of the YOLO algorithm and for training it.[8] It performs state of the art object detection at ease. Previous detection systems apply their model to an image at multiple locations and whichever regions 'highly scored' are considered detections. But in YOLO, we apply a single neural network to the full image and can detect the objects in an image in one look.

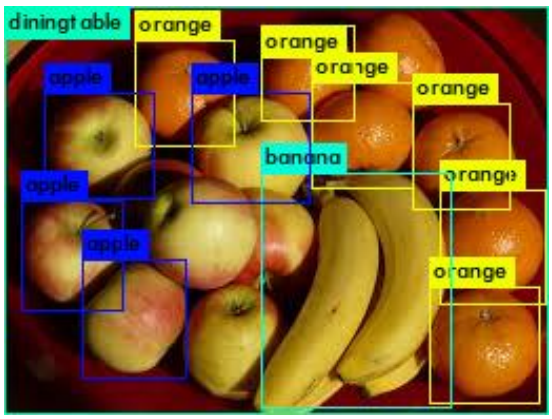

Figure 18: YOLO detecting most of the fruits in a basket. Image from the web[8].

This network splits up the image into regions and places bounding boxes and percentages for each region, and using a condition, we can show the most confident predictions. The major revolution YOLO brought was the capability to perform detections in one go. This approach is much faster than previous approaches and other detection methods, as we use one neural network instead of multiple. Also, YOLO learns the general representation of objects. When trained with the same set of images and when tested, YOLO outperforms detection methods like R-CNN by a huge gap and since it has a general idea of objects detected, it is less likely to fail when applied to new images or new inputs. But as advantageous

it may be, it still lags behind the state-of-the-art detection systems in accuracy. Though it can quickly pick up the objects from an image, it struggles to accurately focus on objects, like the small objects.[12] It is a tradeoff between speed and accuracy. We can run the algorithm using the CPU or the GPU (if there exists a dedicated graphics card). Difference between the CPU and the GPU while running the code, is the speed or the frame per second.

As discussed above, that YOLO was first proposed by Joseph Redmond, which was YOLO v1 version which came about May 2016 and it set the foundations of the algorithm and the subsequent versions which came afterwards were improvements to the original one. The algorithm works by dividing the image into grids, in which each grid had their confidence scores predicted, alongside the class probabilities.

YOLO v2 came out in December 2016, which was to improve the first version. Basically, the fact that the first version couldn't detect close up objects accurately and would perform few errors in placing the labels. It further generalizes better over any image size as it uses the mechanism of resizing the images to its liking. More and more convolutional layers were added to make detection accurate. A faster version of YOLO was introduced which only had 9 layers, which would take less time to compare grids and make detections. [15]

YOLO v3, which came about April 2018, which is based on the network Darknet, has expanded to 53 convolutional layers. All the different versions of YOLO and its pre-trained weights and configurational files which are required are all found in the original repository, by J Redmon in his website. [9] In this site, there are comparisons between each version based on their accuracy and fps, and depending on the versions, there are pre-trained files and instructions on how to access it and run it on one's computer.

YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45	cfg	weights
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35	cfg	weights
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20	cfg	weights
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220	cfg	weights
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20	cfg	weights

Figure 19: Different yolo versions with specifications found on the site, ready to be used. [15]

We will be running two pre-set files, yolov3 and yolov3-tiny and check the results through experiments and testing. Since it is the latest (at the time of experimentation), the weights which are needed and a model which is already trained on the COCO dataset, which comprises of 80 classes, out of which, the

required objects to be detected and more are found. While running the Yolov3 set files on CPU, we get about 1.5 to 3 frames per second, which is slow to implement, but is highly accurate, while, running the Yolov3-tiny set files on CPU, we get about 14 to 17 frames per second which is decently fast but less accurate than the previous set files. Another way to tackle this issue, is to run it on GPU. Other object detection methods such as TensorFlow or Keras can also be used but will reach the same speed if run on CPU.

Running it on GPU is a very meticulous process due to strenuous installation of Graphic card related files, which are not easy on Windows (current operating system) as it is not user-friendly as compared to Linux or Mac in creating specific environments which help us access the GPU. The 'make' files which can be downloaded easily has to manipulated based on the GPU and on what operating software. For example, on Linux, one can easily download the make files and run it as it is, cause it is compatible.

Now that we have downloaded the pre-set files, we shall test it on an image with multiple objects.

```
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
```

Figure 20: A snip of the code, where we read the set files.

Above, we can see we read the set files (yolov3.weights, yolov3.cfg) and we initialize an empty set called classes which is used to read the names of objects that can be detected from file 'coco.names'. OpenCV's new deep neural network (dnn) module contains two functions that can be used for preprocessing images and preparing them for classification via pre-trained deep learning models. We will be testing these files on an image of a room.

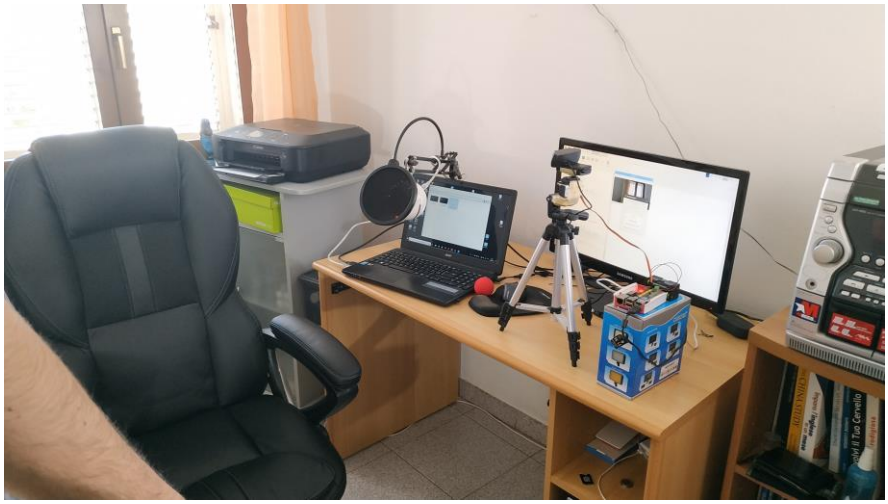

Figure 21: An image which contains objects that can be tested taken from the web. [21]

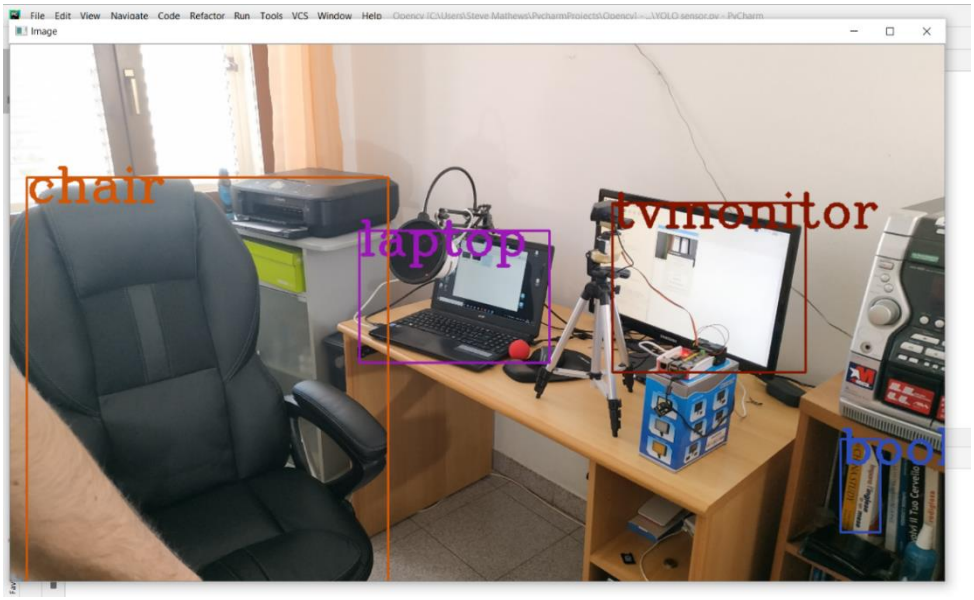

Figure 22: Objects detected using coco.name files.

As we can see it is successful and can now be tried on real time video frames. The process is the same as previous versions of the code. We read the pre-set files and the name list file. We use the blob function to return a blob which refers to a group of connected pixels in a binary image. Then we compare the blobs to the set files and depending on the 'confidence' of the object or how confident the code identifies the objects; we place a rectangular box around said object with its name and confidence percentage. Now the confidence condition in a code changes depending on the type of input. For example, in the above

picture, the confidence condition to detect an object was if its confidence was more than 0.5 or 50%. As it is an image and does not involve motion or unwanted illumination, the code can detect more accurately. When using videos, the confidence condition decreases, as the objects in a video mostly move and since its recording real time frames, we can face the same issue which we faced in detecting objects based on color, the unwanted illumination, which can 'confuse' the code into placing another object's label on it. When tested, it proved to be highly accurate, and shows potential when used to scan real objects or to test its performance (which are discussed under Experimental Results). Now that we can identify objects and label it, the next step is to find the area of the object, which is very easy, because of the bounding box feature. In this program, we use the YOLO files to identify the objects, the COCO dataset which has all the classes, to label the objects, and the bounding box to place over the object. The area of the bounding box is enough to get an idea how much spatial area the object takes and the coordinates of the object wherever it is on the frame or image. Another benefit for this approach that it is still learning or trainable. All we need is its data from when it was first trained, and it can be trained even further to detect new objects depending on the environment or task at hand. As discussed, that running it on CPU, is a bit slow, but this is basically because the program compares the images to its trained files or 'layers' to make a prediction. This can be made faster if we focus on one aspect for testing. For example, when we look at the COCO dataset,

```
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
    print(classes)
```

Figure 23: Printing all the classes in COCO data set.

It prints classes like 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', etc. If we remove the classes that are not required for a specific environment or testing, it will make it easier and faster for the program to analyze and detect objects accurately or as discussed , running on GPU.

5. Experimental Analysis

In this section, we are going to perform some experiments and simulations to test the accuracy and speed of detection of the two weight files proposed previously. First series of tests will be detecting generic images or videos which are found on the road on a daily basis. From this test, we will obtain the frame rate, how "confident" are the objects detected and the time required to detect the objects and how accurate each weight file is. Second series of test will be simulations. These simulations will be carried out either by building a basic layout for a car to go through with certain objects placed for detection or by real life simulation by running the program when driving on the roads and to compare both files precision when trying to detect moving objects.

5.1 Test 1- Images

In the first test, there are going to be two images which differ by the number of vehicles to detect. I will be using both versions of datafiles to test precision on images.

Figure 24: Image of road for testing with less vehicles taken from the web [25]

Figure 25: Image of road for testing with a lot of vehicles taken from the web[26]

Result – yolov3

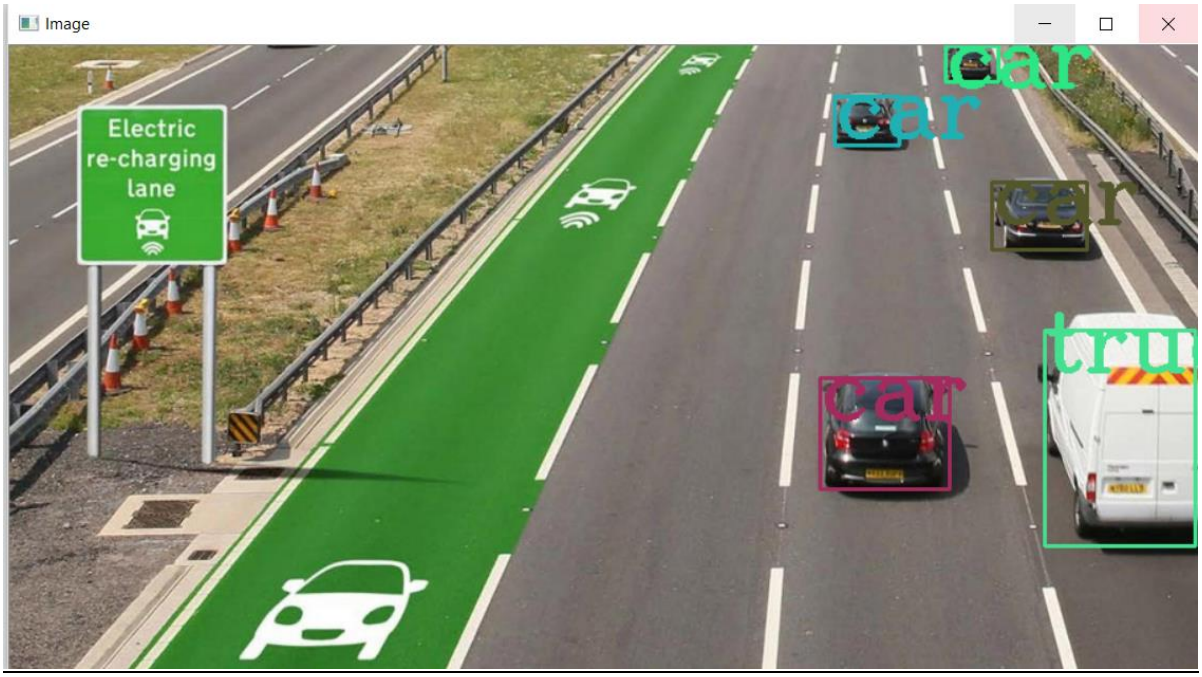

Figure 26: Results after running the code on test image 1 from yolov3

As seen, yolov3 file is successful in detecting all 5 vehicles, which makes this very accurate, no matter where the vehicles are located.

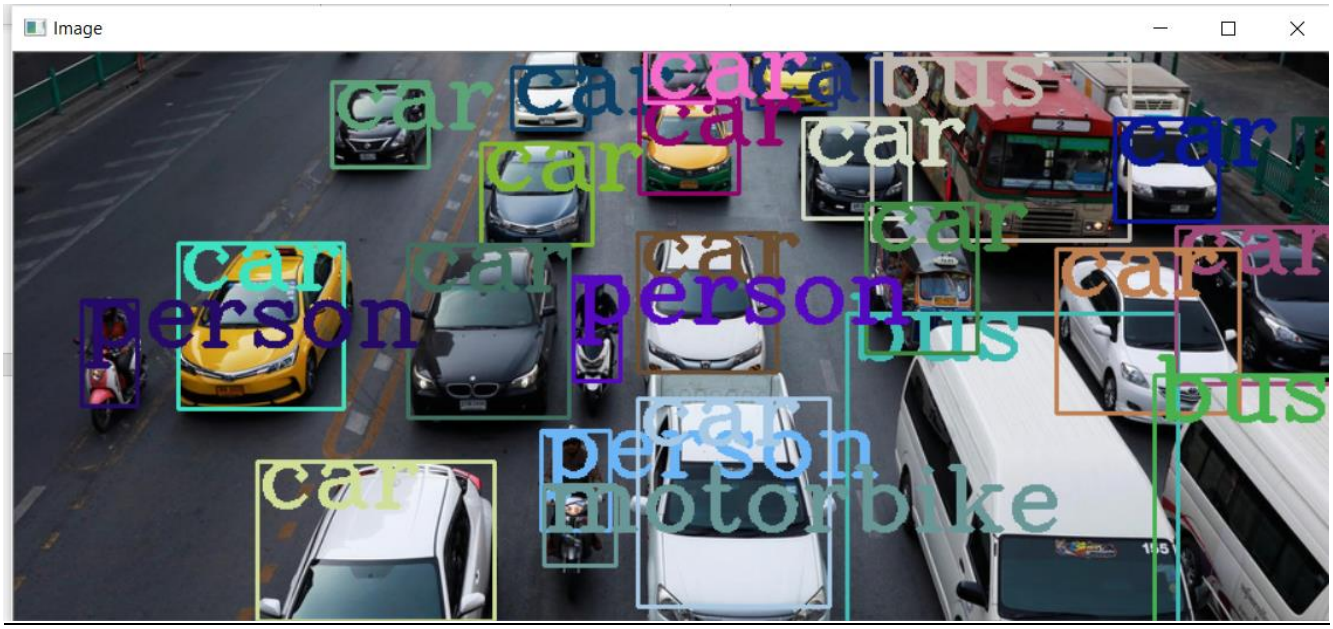

Figure 27: Results after running the code on test image 2 from yolov3

As seen above, number of objects to detect varies the level of accuracy in the image. What is seen above is that, most of the vehicles are detected and it even identifies the people which are on motorbikes (which is also detected). It has also detected the buses and a person on the right most side of the image though it has not been fully shown.
Even though it has detected most to all the vehicles shown, it has mistaken one truck by identifying it as a car by just focusing on the front of the truck in the top right. Also, on the left side, it has identified the person but not the bike. This may be due to the program concentrating on the person based on the similar color of both the bike and person and using non-max suppression it has focused on the person because it is more confident of the person than the bike. The auto-rickshaw has been detected as a car, because it has not been trained to be detected and it was labeled as a car as it was the closest resemblance.

This can be rectified by training the datafile by feeding more images so that the program can "learn" to distinguish and based on how "trained" it is, it can detect no matter where the object is detected in theory. It also depends on the quality of said image or video, higher the resolution of the image/video, better is the detection.

Image 1

Vehicles	Number	Detection
Cars	4	4
Trucks	1	1

Image 2

Vehicles	Number	Detection
Cars	14	14
Bikes	3	1
People	4	4
Bus	1	1
Mini vans	2	2 detected as buses
Truck	1	1 detected as car

Result – yolov3-tiny

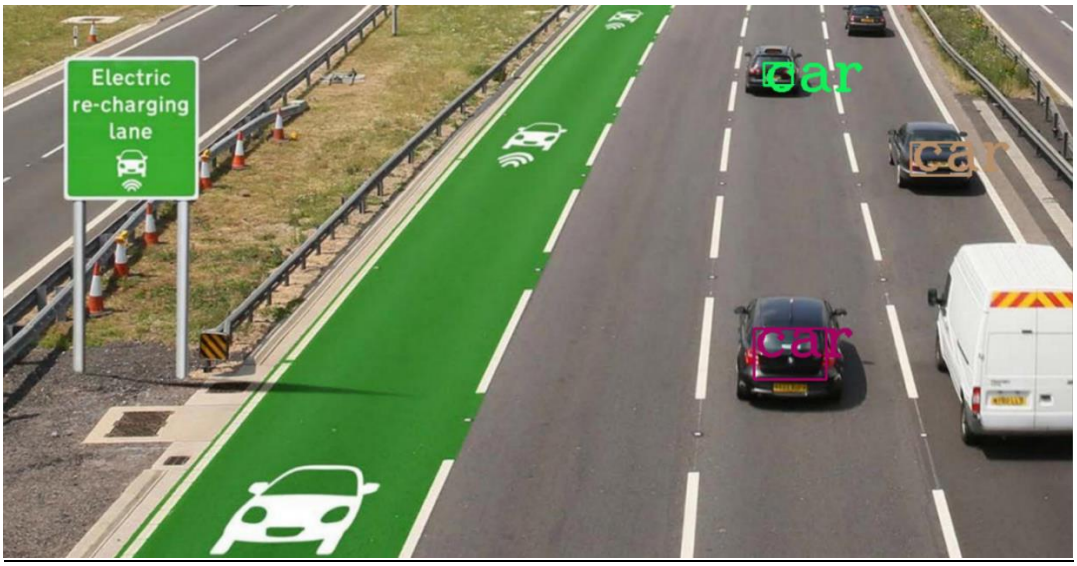

Figure 28: Results after running the code on test image 1 from yolov3-tiny

As observed, not all the vehicles are being detected by the yolov3-tiny weight files. Though it works, it is not very accurate.

Figure 29: Results after running the code on test image 2 from yolov3-tiny

As observed, it has only detected 3 vehicles, which is almost a drop of **86.95** percentage of accuracy, which is obviously not acceptable to be used on the roads.

This is due to the yolov3-tiny weight file. It has been trained to be faster but at the expense of accuracy.

Image 1

Vehicles	Number	Detection
Cars	4	3
Truck	1	0

Image 2

Vehicles	Number	Detection
Cars	14	2
Bikes	3	0
People	4	0
Bus	1	1
Mini vans	2	0
Truck	1	0

Attached below are graphs, which show the number of objects detected by both the set files and on both images. On the y-axis, it depicts the number of vehicles it has detected from image 1, x-axis depicts the vehicles it as detected.

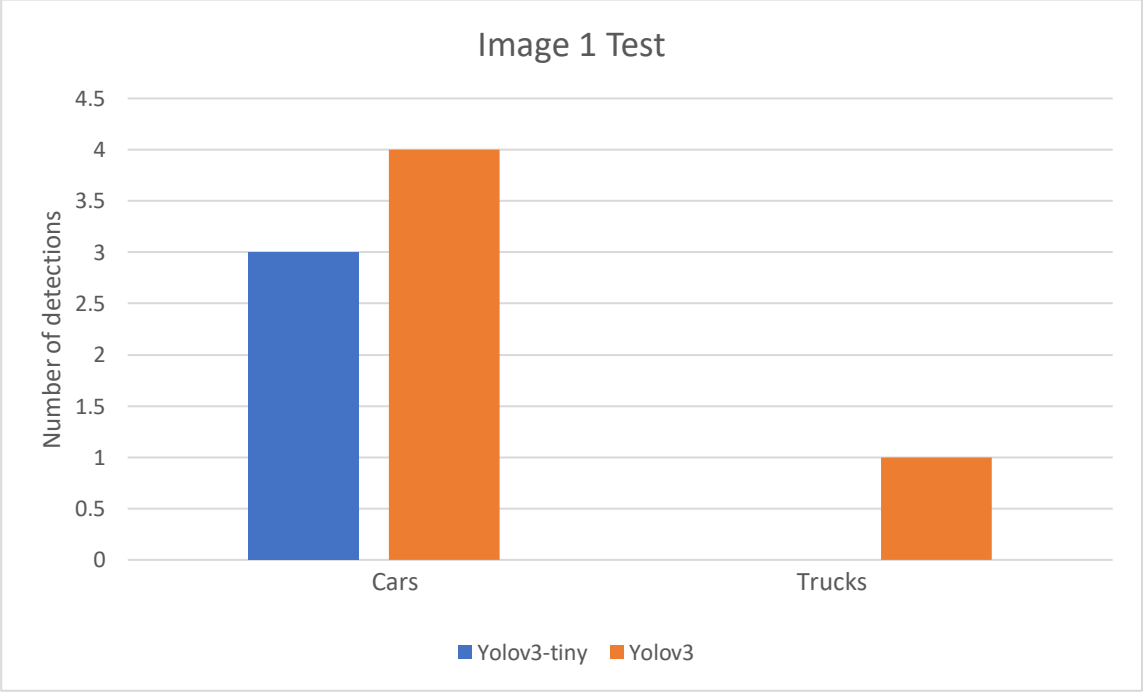

Figure 30: Graph comparing the performance of both test files on test image 1.

Figure 31: Graph comparing the performance of both test files on test image 2.

From the graphs above, one can clearly see that the yolov3 file far surpasses the detection capability of the yolov3 tiny file based on the number of detections. These tests are done on the same images, hence detection speed or frame rate cannot be discussed here but in the next set of tests which include moving images or video files, those factors come in play.

5.2 Test 2 – Video

In this test, I shall be playing a video file which contains few minutes of a road with vehicles going past. Again I shall be using both weight files, and the results will include accuracy the speed of detection, frame rate of the video and to see how long an object gets detected while moving and if it maintains the same label it is categorized under. I will be taking one frame from the video test and that will be the basis of the comparison.

Result – yolov3

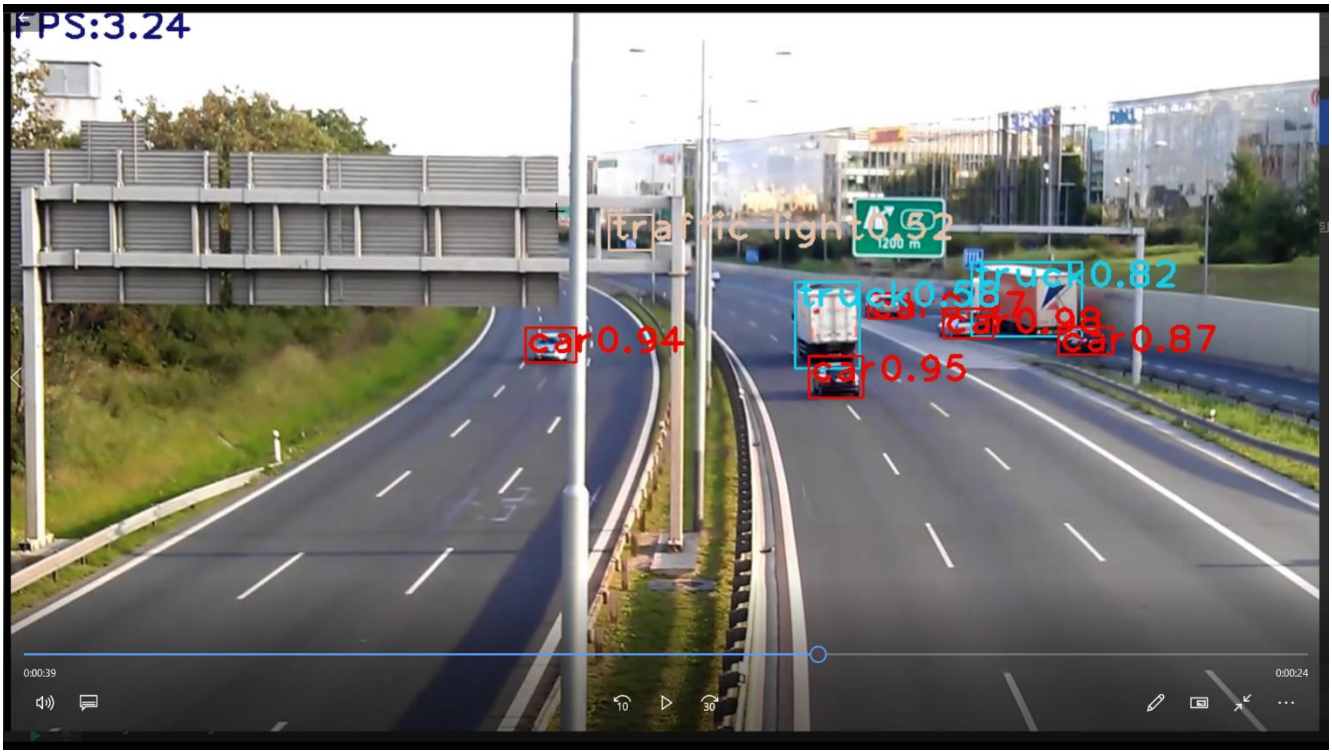

Figure 32: Yolov3 result on a video file taken from the web[27]

As seen above, we have fps and rate of detection and accuracy as parameters to compare. In this frame, all vehicles present are identified and labelled correctly with the "confidence" level of its detection and it also identifies the traffic light which is further away in the frame. It is very accurate but the downside is the fps. As it is run on CPU (Intel i7 chip), one can expect this much fps or just a bit faster. The video when played is processed much slower and then accurate detection takes place. It shows potential to be used on the roads but it has to be with much faster frames almost equal to the actual motion of vehicles around it the sensor.

Result – yolov3-tiny

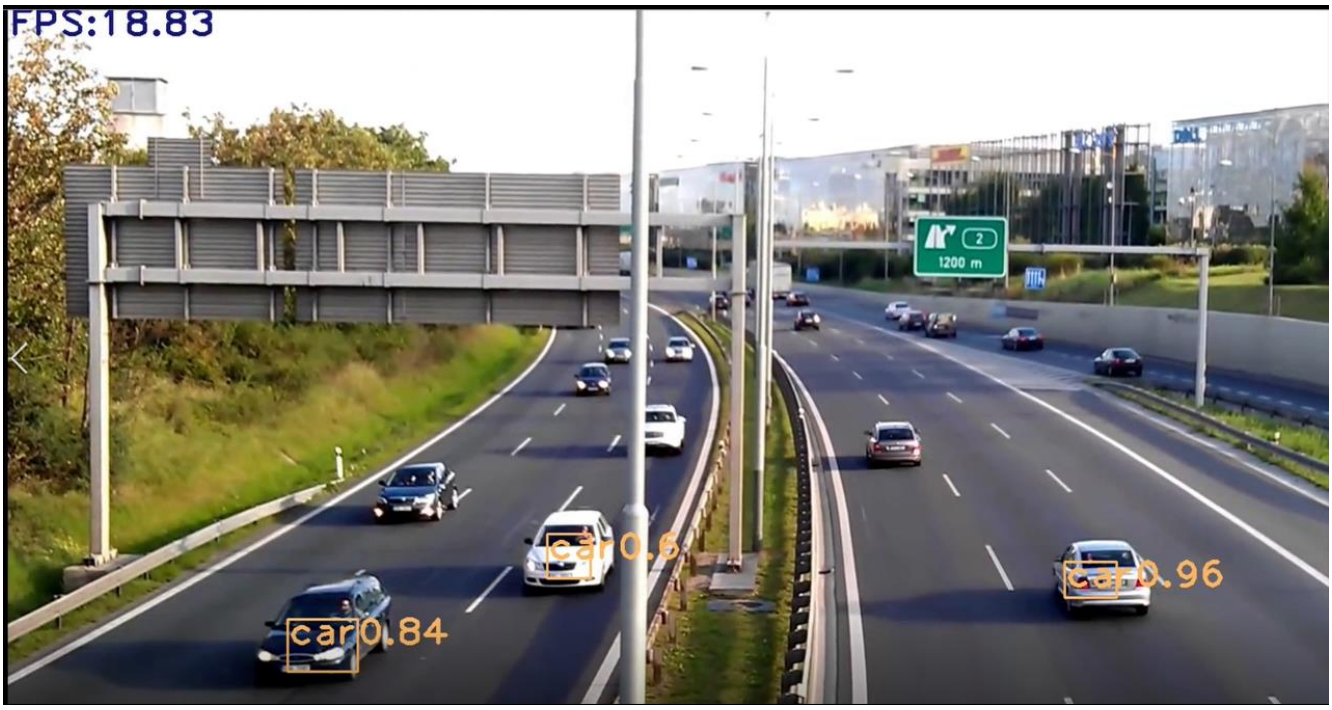

Figure 33: Yolov3 tiny result on video file taken from the web[27]

As seen above, most of the vehicles are not detected as seen in the previous test. Fps is quite stable and does not lag unlike the other configuration file. The accuracy is very low in this test, and vehicles nearby a particular distance from the camera gets detected and after the vehicle is present outside that distance, it is unable to get detected. The only advantage seen using this preset file is the fps and nothing more in this test.

5.3 Test 3 – Real time testing

In this test, I shall be taking it outside on the roads of Prague, where both files configuration files will be used. Again, the fps, the "confidence" of detection and the accuracy will be compared between both the files. As the end result will be videos, I will be taking again a frame as our result to compare. The web camera used in this is a 640 x 480 resolution camera, hence the frames recorded are not as clear.

Result – yolov3

Figure 34: Yolov3 result in real time recording.

Figure 35: Yolov3 result on the roads from real time recording

As seen above, like in previous tests, vehicles in the sight of the camera gets detected, thereby making it very accurate. The downside noted during this experiment is the very low fps which is shown on the top left. As the frames are being recorded, I could observe a 1 second lag between reality and the program. Though the detection is very good, the fps has to be increased to be used viable for future uses.

Result – yolov3-tiny

Figure 36: Yolov3-tiny result on the road in real time

When used in real time, the yolov3-tiny file has actually been very accurate and stable as compared to its results in the previous tests. As the car is moving with respect to its surroundings, the camera is able to detect almost all the objects in its vicinity with no lag as compared to the yolov3.

The experiments carried out above, were carried out during the day with adequate sunlight, illuminating all the vehicles. Hence, the camera can detect most of the objects. Drawbacks of camera is similar to the drawbacks of the eye, which is the presence of light. Under cloudy conditions or rain, range of detection would decrease, but with the help of the headlights, nearby objects get illuminated which is then picked up by the camera. It tends to become harder during the night, as the only source of illumination are the headlights or the street lamps. In these cases, cameras are not highly favored and a variation of the camera is used, which is infrared or thermal cameras. They work by recognizing the heat signatures of all the objects and then proceeds to make a layout of the position of objects around the car.

6. Conclusion

This work focuses on the concept of autonomous driving by giving a brief introduction. We also focus on the main part of the concept, by talking about the sensors one would find in an autonomous vehicle, and by comparing its features and drawbacks and different classifications of automation. As there are a lot of sensors as discussed, I have focused on the camera.

Goals have been set, which needed to be met and discussed the stages taken to reach the final product. I have experimented with different aspects of an object deciding what would be the ideal attribute of an object to use as the basis of our detection. Drawbacks of each step is also discussed such as the color, shape detection and even detecting objects by their movement in a frame.

Finally, I code a program with respective software which uses neural networks to help in detection as it is much faster than the previous approaches to detection, easily trainable to detect any object according to the desired situation. I have focused on YOLO pre-set files of which there are two versions, whose differences have been mentioned. By accessing the YOLO set files, I have proceeded with couple of experiments to determine its usability on an actual road.

We find out that the YOLOv3 file has been trained to detect more objects than its counterpart yolov3-tiny. Though it is very successful and accurate, it comes with the expense of speed. This was why the tiny version of Yolov3 was trained. It was trained to detect less objects and thus would be much faster in detection as seen in the experiments. These tests were all done using the CPU to run the programs, which have their limitations when running neural networks which require a lot of processing speed. One method to make the YOLOv3 much more usable in by running it, using GPU as it would cut the processing speed by more than a half. Running the code on GPU would require a variation of the code by enabling GPU counters and different configuration files and settings which makes it strenuous and time-consuming.

The last experiment was to test the program on the before mentioned robotic model car, which a camera would be attached to, and would be made to go around a maze with several objects which one would observe in their day to day life on the road. But due to the present situation, it was not possible to test it with the model car. Hence the last experiment was modified to real life testing in real traffic situations, where a web camera would be kept on the dashboard and made to perceive objects in front of us and we would observe the results on the screen.

As seen in the last experiment, while the car was in motion, YOLOv3-tiny did remarkably well as compared to its previous attempts. It processed frames without any lag and identified objects very fast with respect to the object's distance to the car, making it seem more practical to use than the YOLOv3, which would detect all the objects no matter the distance, but with almost a 1 second lag, which could prove to be a problem. Possible solutions to the problems are discussed with ideas which could further improve the detection capability of the software.

7. Future improvements

After handling the experiments, there are factors that can be worked on, to make this program more viable. At the time of experiment, Yolov3 is the latest pre-set file used, but due to the advancement of technology, it will just be a matter of time when the next improved set files may be released which would surpass the old ones mainly through the speed or fps. Next factor would be the camera, which acts as the eyes to the vehicles. A sharper camera would result in more crisp and quality images which makes it easier for detection of objects. During the day or well lit situations, the camera present would be sufficient enough, but if the program was made to run during the night for example, the results wouldn't be as convincing during the day. In these scenarios, coupling it with another sort of camera or other sensors would be advantageous. As this thesis is about the camera, a thermal camera coupled with the normal camera would make it useful during low light situations. Also, instead of using CPU which is used for normal work, GPU in this case would make the program much faster as it uses more cores for the working of the program, which means more accurate confidence values at faster processing times. These are some of the few areas which can be improved in the future, which would help in making the program more practical.

Works Cited

[1] Davies, A. (2018, December 13). What Is a Self-Driving Car? The Complete WIRED Guide. Retrieved January 29, 2020, from https://www.wired.com/story/guide-self-driving-cars/

[2] Ibañez-Guzmán, J., Laugier, C., Yoder, J., & Thrun, S. (1970, January 01). Autonomous Driving: Context and State-of-the-Art. Retrieved February 20, 2020, from https://link.springer.com/referenceworkentry/10.1007/978-0-85729-085-4_50

[3] Law, W. (2019, September 12). An Introduction to Autonomous Vehicles. Retrieved January 25, 2020, from https://towardsdatascience.com/an-introduction-to-autonomous-vehicles-91d61ff81a40

[4] The 5 Levels of Autonomous Vehicles. (2020, January 24). Retrieved January 20, 2020, from https://www.truecar.com/blog/5-levels-autonomous-vehicles/

[5] Crowe, S. (2019, December 23). Researchers back Tesla's non-LiDAR approach to self-driving cars. Retrieved January 18, 2020, from https://www.therobotreport.com/researchers-back-teslas-non-lidar-approach-to-self-driving-cars/

[6] Cameras, radar or lidar? Which is best for autonomous vehicles? *VIA Technologies, Inc.* [online]. 27 September 2019. [Accessed 11 February 2020]. Available from: https://www.viatech.com/en/2019/09/which-sensors-are-best-for-autonomous-vehicles-cameras-radar-or-lidar/

[7] Burke, K. (2019, April 16). How Does a Self-Driving Car See?: NVIDIA Blog. Retrieved February 27, 2020, from https://blogs.nvidia.com/blog/2019/04/15/how-does-a-self-driving-car-see/

[8] Brummelen, J., O'Brien, M., Gruyer, D., & Najjaran, H. (2018, March 07). Autonomous vehicle perception: The technology of today and tomorrow. Retrieved March 20, 2020, from https://www.sciencedirect.com/science/article/pii/S0968090X18302134?via=ihub

[9] Redmon, J. (2018). YOLO: Real-Time Object Detection. Retrieved April 14, 2020, from https://pjreddie.com/darknet/yolo/

[10] Harner, I. (2020, January 21). The 5 Autonomous Driving Levels Explained. Retrieved February 20, 2020, from https://www.iotforall.com/5-autonomous-driving-levels-explained/

[11] Khvoynitskaya, S. (2020, February 11). 3 types of autonomous vehicle sensors in self-driving cars. Retrieved March 20, 2020, from https://www.itransition.com/blog/autonomous-vehicle-sensor

[12] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas. NV, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[13] Bagloee, S. A., Tavana, M., Asadi, M., & Oliver, T. (2016, August 29). Autonomous vehicles: Challenges, opportunities, and future implications for transportation policies. Retrieved March 20, 2020, from https://link.springer.com/article/10.1007/s40534-016-0117-3

[14] Weber, M. (2020, May 08). Where to? A History of Autonomous Vehicles. Retrieved January 20, 2020, from https://computerhistory.org/blog/where-to-a-history-of-autonomous-vehicles/

[15] Pugliese, M. (2018, October 05). A very shallow overview of YOLO and Darknet. Retrieved April 2, 2020, from https://martinapugliese.github.io/recognise-objects-yolo/

[16] Prateek Joshi, OpenCV with Python By Example: Build real-world computer vision applications and develop cool demos using OpenCV for Python, Packt Publishing (September 2015), ISBN-10: 9781785283932

[17] Sensing and Control for Autonomous Vehicles: Applications to Land, Water and Air Vehicles (Lecture Notes in Control and Information Sciences), Springer; 1st ed. 2017 edition (December 2, 2017), ISBN -13 :978-3319553719

[18] The role of camera technology in driverless cars. (2019, September 20). Retrieved February 20, 2020, from https://www.exeros-technologies.com/the-role-of-camera-technology-in-driverless-cars/

[19] Phillips, D. (1995). *Image processing in C*. Lawrence, Kansas: BPB Publications.

[20] Rosebrock, A. (2020, April 18). Real-time object detection with deep learning and OpenCV. Retrieved February 24, 2020, from https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/

[21] Canu, S. (2019, June 27). YOLO object detection using Opencv with Python. Retrieved April 25, 2020, from https://pysource.com/2019/06/27/yolo-object-detection-using-opencv-with-python/

[22] Synced. (2017, August 20). Deep Learning in Real Time - Inference Acceleration and Continuous Training. Retrieved April 20, 2020, from https://medium.com/syncedreview/deep-learning-in-real-time-inference-acceleration-and-continuous-training-17dac9438b0b

[23] Gilbertsen, C. (2017, March 27). Here's How The Sensors in Autonomous Cars Work. Retrieved April 20, 2020, from https://www.thedrive.com/tech/8657/heres-how-the-sensors-in-autonomous-cars-work

[24] Jefferson, A. (2018, March 19). When Cars Park Themselves: A Guide to Parking Assist Technologies. Retrieved April 20, 2020, from http://www.proctorcars.com/when-cars-park-themselves-a-guide-to-parking-assist-technologies/

[25] Chester, T. (2015, August 17). Electric car charging - UK [Digital image]. Retrieved April 24, 2020, from https://mondrian.mashable.com/2015%252F08%252F17%252F4d%252FDWPTphoto_c.ecd10.jpg%252F1200x627.jpg?signature=YtJzRb86JnxUw0c0DZiAIPS625s=

[26] Tun, S. Z. (2018, December 18). Vehicles are seen stuck in traffic along a road in Bangkok, Thailand [Digital image]. Retrieved April 25, 2020, from https://stock.adobe.com/cz/239430425?as_campaign=TinEye&as_content=tineye_match&epi1=239430425&tduid=39bf74a92c6351c0b6e29274bfae5445&as_channel=affiliate&as_campclass=redirect&as_source=arvato

[27] Youtube. 2011. *HTC Sensation FULL HD 1080P Video Sample (Highway Traffic)*. [online] Available at: <https://www.youtube.com/watch?v=wWLAc6mdJrs> [Accessed 18 May 2020].

[28] Shavit, A. (2006, January 1). OpenCV Logo with text svg version [Digital image]. Retrieved March 8, 2020, from https://cs.m.wikipedia.org/wiki/Soubor:OpenCV_Logo_with_text_svg_version.svg

[29] Puttemans, S. (Director). (2016, October 25). *Vtest* [Video file]. Retrieved February 2, 2020, from https://github.com/opencv/opencv/blob/master/samples/data/vtest.avi

[30] Novak, M. (2016, May 19). Robotic car. Retrieved April 12, 2020, from https://control.fs.cvut.cz/en/srd/Robotic_car

Appendix

Contents of the attached CD

Description
The PDF file of the thesis
The Python source code of the program
Yolov3 and Yolov3-tiny files with configuration files
Test images of Experiment 1
Test video file of Experiment 2
Resultant video of Experiment 2
Resultant video recording of Experiment 3 (real life testing)