



School of Electrical, Electronic & Computer Engineering

Faculty of Engineering, Computing & Mathematics

The University of Western Australia

GENG5512: Engineering Research Project Part 2

Simulating Autonomous Vehicles

Final Paper

By

Lijie Liu

22034019

Academic Supervisor: Prof. Thomas Bräunl

Word Count: 6884

19th October 2018

Contents

Abstract	4
1. Introduction	5
1.1 Autonomous Vehicles	5
1.2 Simulating Autonomous Vehicles	8
1.3 Research Goals	9
1.4 Basic structure of this paper	9
2. Related Technology	10
2.1 Machine Learning	10
2.2 Convolutional neural network	10
2.2.1 AlexNet Model	12
2.2.2 YOLOv3	13
2.3 Edge Detection	15
2.4 Tesseract OCR	15
2.5 Hardware & Software Environment	16
3 Methods for Virtual Autonomous Vehicle	17
3.1 Driving in Traffic Lanes	17
3.1.1 Driving Behavior Training	18
3.2 Detecting Traffic Signs	20
3.2.1 Identify Circle	20
3.2.2 Identify Speed Number	21
3.3 Detecting Obstacles	21
3.3.1 Traffic Light	24
3.3.2 Vehicle	25
3.3.3 Pedestrian	26
4 Results for Virtual Autonomous Vehicle	27

4.1 Driving	27
4.2 Traffic Sign	29
4.3 Obstacles Detection	31
5 Future Work and Summary	34
Reference	35

Abstract

With the development of the economy, the numbers of vehicles in various cities around the world are increasing dramatically. At the same time, problems such as traffic congestion, traffic accidents, air and noise pollution are getting worse and worse in urban areas in different countries. Considering the current state of IT technology, the best solution for most of traffic problems at the moment is autonomous vehicle. The problem I am trying to solve in this research paper is developing a low-cost autonomous vehicle that is suitable for internal roads environment, for instance, university campus, industrial areas and large residential areas. For this research project, I use an open urban driving simulator Carla, which offered an internal roads environment, to train and test my autonomous vehicle.

Compared with other autonomous vehicle that based on high precision map and optical radar, the approaches I adopted to develop this low-cost autonomous vehicle are machine learning and computer vision. During the research, a few methods have been tried to identify traffic lanes, traffic lights, vehicles and pedestrians. Eventually, I chose to use AlexNet, one of Convolutional Neural Network (CNN) models for training driving behavior classifier, and use YOLOv3, a real-time object detection system for detecting obstacles, such as other vehicles, traffic lights and pedestrians. Moreover, edge detection and Tesseract OCR have been used for traffic sign detection. At the end of this research project, an autonomous vehicle in Carla simulating environment has been found that achieve human like self-driving, avoiding obstacles, and detecting traffic indication in real time.

Keywords: Autonomous vehicle, Computer Vision, Machine Learning, Edge Detection, OCR

Simulating Autonomous Vehicles

1. Introduction

1.1 Autonomous Vehicles

An autonomous vehicle is a ground vehicle that can receive its environment such as other vehicles, pedestrians, traffic safety instructions, and move automatically without human input. In order to detect surrounding environment, the autonomous vehicles will use a variety of techniques such as radar, laser light, GPS and Computer Vision. The control system in the autonomous vehicles use the sensory information to navigate the roads, avoid the obstacle and follow traffic rules.

The idea of automating driving has been found since at least 1920s, at that time, autonomous vehicle was called “Phantom motor car” (The Milwaukee Sentinel, 1926). In the 1980s, Navlab project from Carnegie Mellon University built a self-driving vehicle which analysis raw images from the road by neural networks and control the steering wheel in real time (Thorpe et al., 1988). In 2009, some vehicle manufactures such as Toyota, BMW and Ford offered automatic parking assistance for their products (Evans, 2009). In 2012, Tesla Model S was released, and Model S is considered the world’s first autonomous car (Tesla.com, 2012). Nowadays, almost all the vehicle manufactures and Internet Companies such as Google, Uber and Baidu are working on Autonomous Vehicles (Whitwam, 2014).

To identify autonomous vehicles, SAE International released a formal classification system for autonomous vehicles in 2016. There are 6 levels of driving mode for vehicle (SAE International, 2014).

Level 0: Automated system issues warnings and may momentarily intervene but has no sustained vehicle control.

Level 1(“hands on”): The driver and the automated system share control of the vehicle.

Level 2(“hands off”): The automated system takes full control of the vehicle. The driver must monitor the driving and be prepared to intervene immediately at any time if the automated system fails to respond properly.

Level 3(“eyes off”): The driver can safely turn their attention away from the driving tasks, the vehicle will handle situations that call for an immediate response. The driver must still be prepared to intervene within some limited time.

Level 4(“mind off”): As level 3, but no driver attention is ever required for safety.

Level 5(“steering wheel optional”): No human intervention is required.

The new Audi A8 is generally considered the first level 3 autonomous vehicle in the world (McAler, 2017).



Figure 1: The new Audi A8 First Level 3 Autonomous Vehicle

Compared with regular vehicles that can be found easily on the roads, there are four dominant potential advantages for autonomous vehicles. Firstly, and most importantly is safety. Autonomous vehicles are safer because the traffic accidents caused by human

errors, delayed reaction time and distracted will almost be eliminated. Moreover, autonomous vehicle has border view and shorter reaction time compared with human driver, and this benefit can make passengers safer (Anderson, 2016). Secondly, welfare. Autonomous vehicles make travel much more easy than regular vehicles. Seniors, minors, disabled person can use autonomous vehicles service individually without another people's help. And driver can do exactly the same things as passengers do in the autonomous vehicles (Stenquist, 2014). Thirdly is traffic. If all the vehicles on the roads are autonomous vehicles, traffic control department can increase the speed limits and decrease safety gaps, in this condition, roadway capacity will increase and traffic congestion will be minimized (Andra Keay, 2012). Lastly, due to safer driving and less traffic congestion, the traffic cost such as vehicle insurance and fuel consumption will reduce (Du, Ang & Rus, 2017).

In order to achieve the goals that autonomous vehicle has ability to avoid obstacles (such as other vehicles and pedestrians) and obtain traffic rules information (such as traffic lights and speed limits), it is important for autonomous vehicle to sense its environment. Perceived devices such as camera, radar and laser light have been utilized widely for autonomous vehicles to detect their surroundings. Figure 2 shows the sensors on autonomous vehicle.

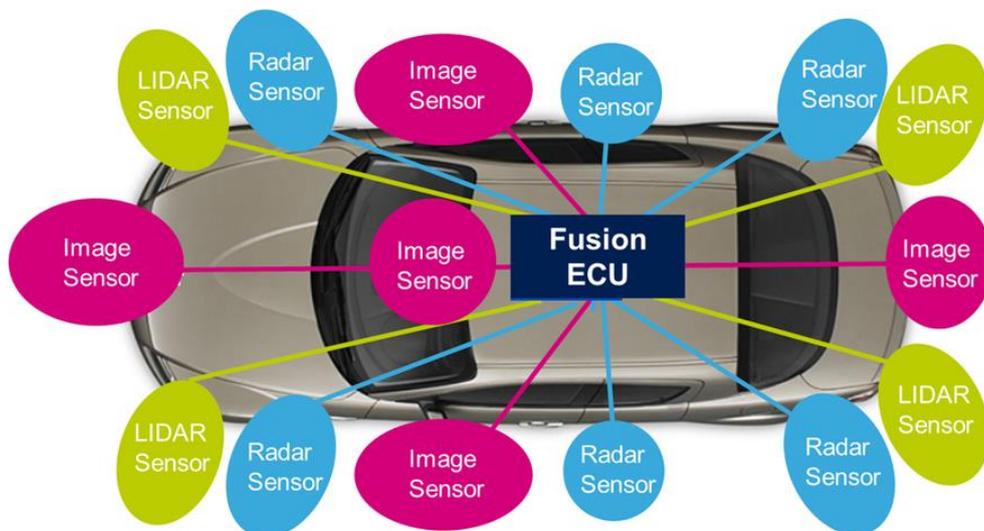


Figure 2: the image from "Sensor Online" shows the sensors on Autonomous Vehicle

1.2 Simulating Autonomous Vehicles

For any vehicle manufacture who wants to build an autonomous vehicle, the most important part for autonomous vehicle is that it can drive perfectly in a variety of real-world scenarios without any accident. However, according to a report by RAND (Kalra, 2016), autonomous vehicles would have to be driven hundreds of millions of miles and sometimes hundreds of billions of miles to demonstrate their reliability in terms of fatalities and injuries, and under even aggressive testing assumptions, existing fleets would take tens and sometimes hundreds of years to drive these miles – an impossible proposition if the aim is to demonstrate their performance prior to releasing them on the roads for consumer use. Table 1 shows how long time the autonomous vehicle require for safety from RAND.

Table 1: RAND Corporation for Autonomous Vehicle Reliability

Benchmark Failure Rate				
Statistical Question	How many miles (years*) would be driven...	(A) 1.09 fatalities per 100 million miles?	(B) 77 reported injuries per 100 million miles?	(C) 190 reported crashes per 100 million miles?
	(1) Without failure to demonstrate with 95% confidence that their failure rate is at most...	275 million miles (12.5 years)	3.9 million miles (2 months)	1.6 million miles (1 month)
	(2) To demonstrate with 95% confidence their failure rate to within 20% of the true rate of...	8.8 billion miles (400 years)	125 million miles (5.7 years)	51 million miles (2.3 years)
	(3) To demonstrate with 95% confidence and 80% power that their failure rate is 20 % better than the human driver failure rate of ,,,	11 billion miles (500 years)	161 million miles (7.3 years)	65 million miles (3 years)
*We assess the time it would take to complete the requisite miles with a fleet of 100 autonomous vehicles (larger than any known existing fleet fleet) driving 24 hours a day, 365 days a year, at an average speed of 25 miles per hour.				

In this situation, simulation is the best solution for building a reliable autonomous vehicle. To ensure that everything is taken into consideration, the simulation should have:

A variety of world scenarios, such as traffic situation, driver behavior, time, weather and road environment.

Multiple sensor suites/array, such as radar, camera and laser light.

With multiple, scalable random permutations.

Nowadays, a variety of vehicle simulator have released, Vires, CarSim, ROS and Carla. Moreover, the researchers from Princeton even use video game GTA V (Filipowicz, A., Liu, J., & Kornhauser, A. 2017) for simulating autonomous vehicles.

1.3 Research Goals

As demonstrated in the introduction, compared with regular vehicles, autonomous vehicles are much safer, cheaper and wider application. Moreover, for any vehicle manufactures that want to build their own autonomous vehicle, simulating autonomous vehicle is one of the most important parts which will directly influence the safety and reliability for autonomous vehicle. Subsequently, this research project' goal is that build an autonomous vehicle on Carla driving simulator, detect traffic light, traffic sign, other vehicles and pedestrians so that vehicle can drive with traffic instruction and avoid obstacles.

1.4 Basic Structure of This Paper

First Part is the introduction, which presents the motivation, status of this research project.

Second Part introduces related technology, machine learning and computer vision have been used for this research project.

Third Part presents the methods for this research project. Use machine learning on vehicles driving, object detection system for obstacles detection and edge detection plus Tesseract OCR for traffic sign detection.

Fourth Part is about the result of this research project.

Last Part is about the future works and the summary for this research projects.

2 Related Technology

2.1 Machine Learning

Machine learning (ML) (Samuel, 1959) is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed, it is a field of artificial intelligence that use statistical techniques to give computer systems the ability to “learn” form data.

The processes involved in machine learning are similar to that of data mining and predictive modeling (Mannila, n.d.1988). Both require searching through data to look patterns and adjusting program actions accordingly. Machine learning has been used in recommendation engine, fraud detection, spam filtering, network security threat detection, predictive maintenance and building news feeds.

With the development of Machine learning, numerous of machine learning approaches have been found these years, ("Machine learning", 2018) for example, Decision tree learning, Association rule learning, Artificial neural networks, Deep learning, Inductive logic programming, Support vector machines, Clustering, Bayesian networks, Representation learning, Similarity and metric learning, Sparse dictionary learning, Genetic algorithms and Rule-based machine learning. In this research project, the approach adopted is artificial neural network, especially Convolutional neural network.

2.2 Convolutional neural network

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data ("MNIST Demos on Yann LeCun's website", 1988). CNN is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. CNN uses a variation of multilayer perceptions designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural

networks (SIANN), based on their shared-weights architecture and translation invariance characteristics (Zhang, Itoh, Tanida & Ichioka, 1990).

A CNN has an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and ReLU (Rectified Linear Unit) layers. Convolutional layers are for convolution calculation layer, then linear product and summation. Pooling layers are for calculating maximum or average of the areas. ReLU layers are activate layer, ReLU is one of the activation functions. Fully connected layers are transfer image data to labels. The image shows how CNN classifier identify a vehicle. Figure 3 shows the structure of Convolutional neural networks.

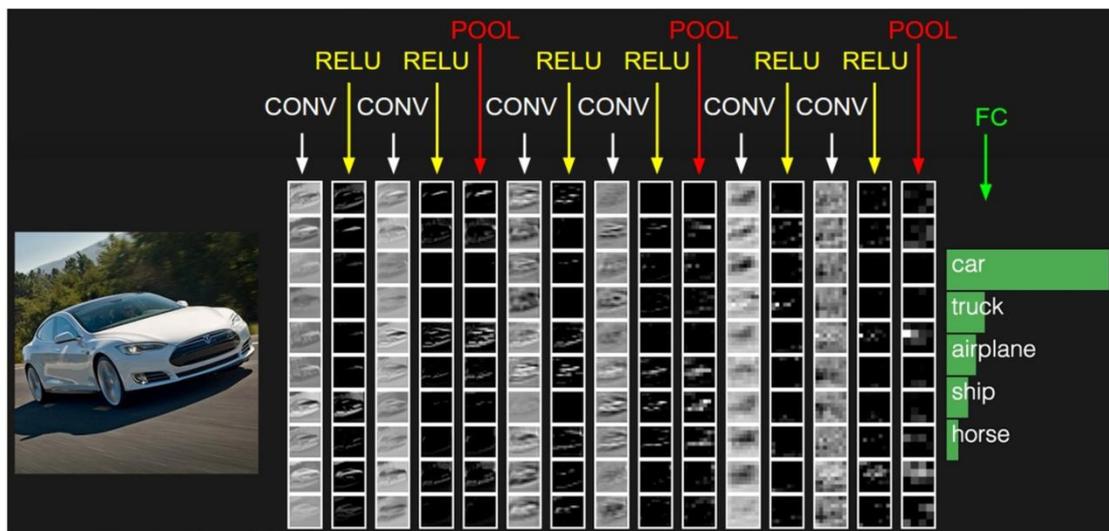


Figure 3: The structure for Convolutional neural networks (Image from Quora)

Based on the differences of the structure of CNN such as number of layers and type of layers, there are several CNN models that have been used widely, Lenet, AlexNet, GoogLeNet, VGG and Deep Residual Learning ("Five Class CNN Models: Lenet, Alexnet, Googlenet, VGG, DRL", 2017). Depends on the requirement and working environment, in this research project, I chose AlexNet model for training.

Moreover, in this research project, except for training classifier, CNN is also adopted for real-time object detection system. Object detection is a computer technology related

to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Methods for object detection generally fall into either machine learning-based approaches or deep learning-based approaches. For machine learning approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, deep learning techniques that are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural network (CNN). In this research project, You Only Look Once (YOLO) has been used for object detection.

2.2.1 AlexNet Model

AlexNet is the name of a convolutional neural network model, it contained eight layers (Krizhevsky, Sutskever & Hinton, 2017). The first five were convolutional layers, and the last three were fully connected layers. It used the non-saturating ReLU activation function, which showed improved training performance over tanh and sigmoid. Figure 4 shows the structure of AlexNet.

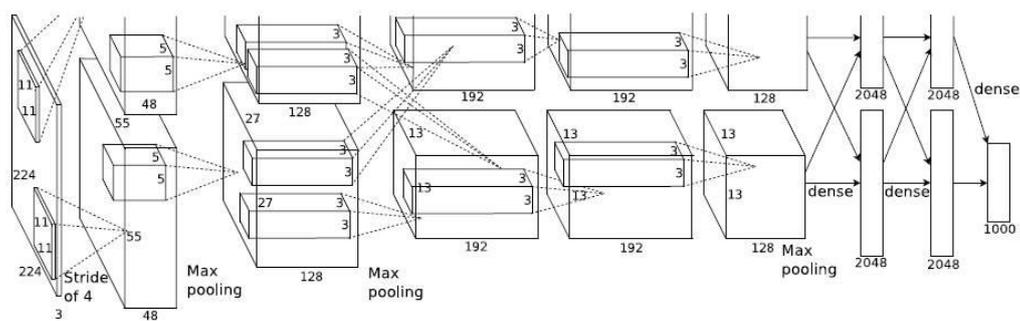


Figure 4: The structure of AlexNet

Compared with other CNN model such as Lenet, GoogLenet and VGG, the reason AlexNet model is selected for this research project is that AlexNet has less computation but good performance. It has only 8 layers, but models such as GoogLenet and VGG has over 30 layers. Then, AlexNet is first CNN model that can work on GPU, which is 40 times faster than running on CPU. Moreover, AlexNet is first CNN model that use

ReLU activation function.

One more feature for AlexNet is that there is a special calculating layer, LRN layer, which is used for smoothing operation for output data from pervious layer. The formula for this calculating layer is:

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$

2.2.2 YOLOv3

You Only Look Once (YOLO) is a real-time object detection system that based on darknet and Convolutional neural networks (Redmon, J., & Farhadi, A., 2018). Figure 5 shows the structure of YOLO. For YOLO algorithm, object detection problem is treated as a regression problem, using one convolutional neural network structure can directly predict the bounding box and class probability from the input image. Compared with other object detection system such as R-CNN, Fast R-CNN and SSD, there are several advantages:

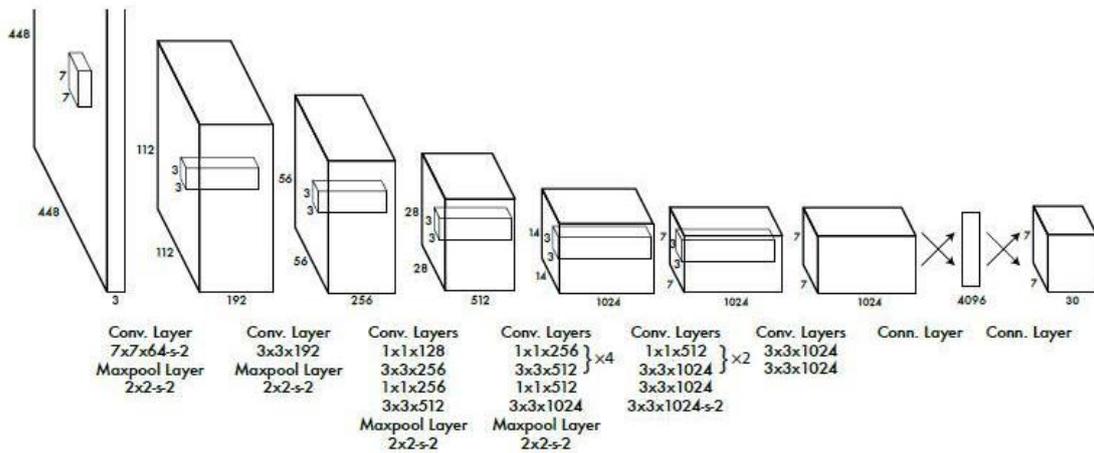


Figure 5: The structure of YOLO

Firstly, YOLOv3 is extremely fast and accurate. According to Figure 6, YOLOv3 is 1000 times faster than R-CNN and 100 times faster than Fast R-CNN.

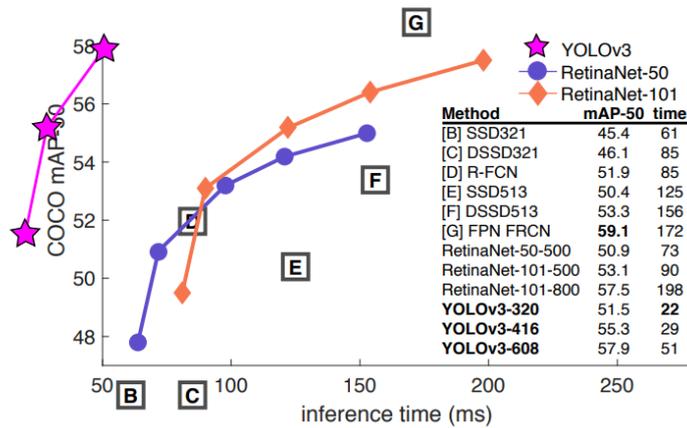


Figure 6: line chart of inference time for different object detection

Secondly, compared with old version YOLO, such as YOLOv2, the performance of detecting small objects in YOLOv3 is much better. Detecting small objects ability is important for autonomous vehicle because find objects in distances is necessary for vehicle so that vehicle has enough time to react about the finding obstacles. Figure 7 shows the comparison between YOLOv2 and YOLOv3 one small objects.

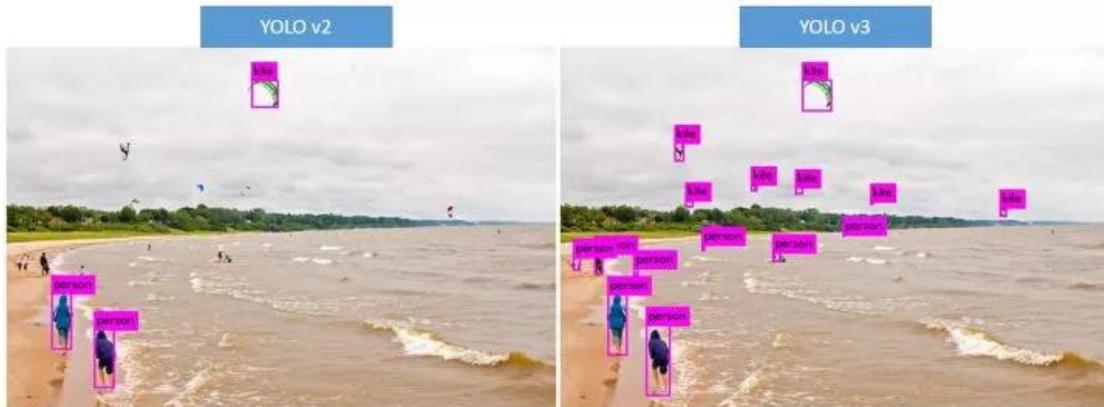


Figure 7: Compared between YOLOv2 and YOLOv3

Thirdly, Generalization. According to the image below, Skull is not in the training dataset, but with YOLO's powerful generalization ability, it is automatically classified into human. Generalization is important for this research project because this object detection system is adopted in virtual environment. Figure 8 shows that YOLOv3 detect skull as person.

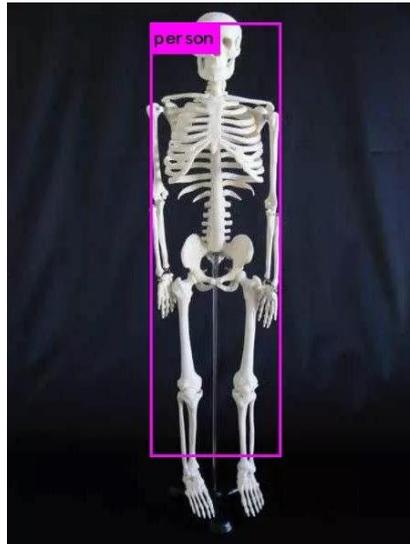


Figure 8: Detect skull

2.3 Edge Detection

Edge detection includes a variety of mathematical methods that aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities (Memon, 2014). The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges. Approaches such as Canny edge operator, Sobel edge operator and Prewitt edge operator are widely used for edge detection. In this research project, Canny edge operator has been used for traffic sign detection because in Carla environment, traffic sign is red circle.

2.4 Tesseract OCR

Tesseract OCR is an optical character recognition engine for various operating systems. ("tesseract-ocr", 2018) It is considered one of the most accurate open source optical character recognition engines. In this research project, Tesseract OCR is used for detecting the digital numbers on traffic signs.

2.5 Hardware & Software Environment

Hardware Environment

This research project is running on a personal computer with

CPU: Intel(R) Core (TM) i7-4770 CPU @ 3.40Hz

GPU: Nvidia GTX GeForce TITAN X, 12Gb

Installed Memory (RAM): 8Gb

Software Environment

This research Environment is running with

Operating System: Windows 10 Pro 64-bit

Programming Language: Python 3.5.6

Master Libraries Included: Tensorflow-gpu, Keras, Pillow, OpenCV-Python, AlexNet, YOLOv3, Tesseract

3 Methods for Virtual Autonomous vehicle

For Carla environment, traffic lanes, traffic lights, traffic signs, vehicles and pedestrians are included in the simulating urban environment. To achieve the goals that autonomous vehicles can drive without breaking traffic rules or hitting obstacles in Carla environment, three major parts should be solved and these three parts are driving in traffic lanes, detecting traffic signs, detecting traffic lights, other vehicles and pedestrians (traffic light can be considered as an obstacle because vehicle needs to stop when traffic light is in red). The following part will focus on these three parts.



Figure 9: Screenshot from Carla environment

3.1 Driving in Traffic Lanes

To achieve driving in traffic lanes, at the beginning of this research project I decided to use edge detection method. My initial plan was detecting double yellow solid lines at the middle and the boundary of traffic line and sidewalk as the driving traffic lane, and make sure the central point of autonomous vehicle is at the middle of these two lines. This method can work in some special condition but it can not be the general method because in some conditions this method can not work. For example, in Carla environment, vehicles need to go through the intersections and, in these areas, there is

no double yellow solid lines. This means that edge detection is not suitable for roads with intersections. Secondly, the performance is not good enough for finding the line of the double yellow solid lines, especially in rainy or overcast weather. At that situation, left side boundary of traffic lane and sidewalk will be identified as the left line and vehicle will drive across the double yellow solid lines. Overall, edge detection method is not suitable for detecting traffic lanes in Carla Environment.

Since Carla is an environment with closed internal roads, which means that the length of roads is limited, I can manually drive the vehicle in Carla environment and follow the rules of driving in my side of roads, in the same time, record my operations and camera image from the vehicle. With all this collected information, I can “teach” vehicle how to drive in its own side of the roads, which I called “Driving Behavior Training”.

3.1.1 Driving Behavior Training

Like most racing games that controlled by keyboard, vehicle in Carla can be controlled by WASD keys, W means Throttle, S means Brake, A means Left and D means Right. To collect data for training driving behavior classifier, I wrote a python script that can save keyboard control and Carla screen in the same time. For example, Figure 10 shows that at that situation, vehicle at the middle of the intersection and decide to go right, the controller should press W and D in the same time, which means the operation of go forward right.



Figure 10: Training dataset

For this method of controlling vehicle, there are 9 different operations, Forward, Brake, Turn Left, Turn Right, Forward Left, Forward Right, Retard Left, Retard Right and No key. The following chart shows how these operations means in keyboard control and training label.

No	Operations	Keyboard Input	Training Label
1	Forward	W	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
2	Break	S	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
3	Turn Left	A	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
4	Turn Right	D	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
5	Forward Left	W A	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
6	Forward Right	W D	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
7	Retard Left	S A	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
8	Retard Right	S D	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
9	No Key	-	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Table 2: Label for Training dataset

After driving for over 3 hours, over 500,000 datasets had been collected. After data preprocessing such as image resize, image transfer, data balancing and data randomly, I put all this processed data into AlexNet model, with epochs of 10, this model had finished training for 30 hours on computer with single GTX GeForce TITAN X.

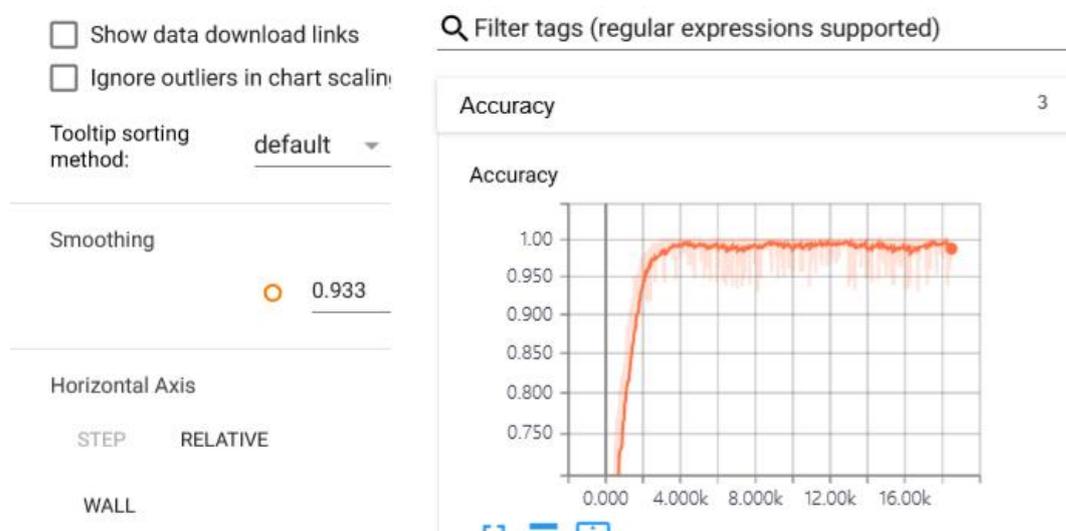


Figure 11: Tensorboard for classifier

Figure 11 is the tensorboard for classifier. It shows that the currency for this classifier close to 100%.

3.2 Detecting Traffic Signs

In Carla environment, there is only one type of traffic signs, which is speed limit sign. This sign is combined with a red circle, and speed number inside the circle. The following image shows the sample of speed limit traffic sign. Figure 12 shows the sample speed limit traffic sign.



Figure 12: Sample of speed limit traffic sign

To detect this traffic sign, the approach that been adopted is detecting circle first, then identify the number inside the circle.

3.2.1 Identify Circle

In this research project, edge detection was adopted for identifying circles. To achieve circle identification, Figure 13 is a flow chart that indicates the necessary operations.

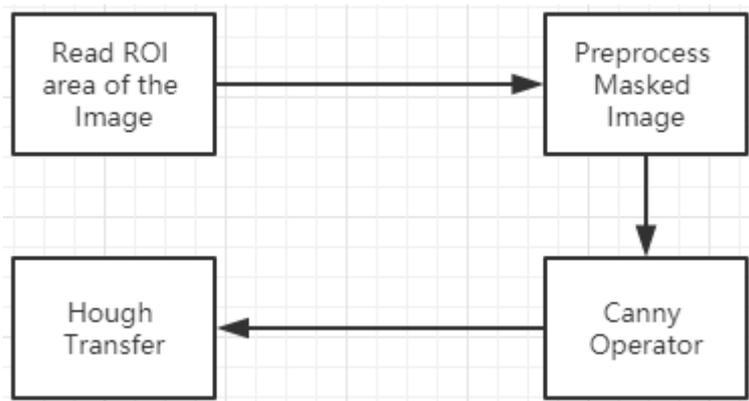


Figure 13: The process of edge detection

ROI (Region of Interest) for this research project is the right side of the image. Set ROI can reduce the amount of calculation, which can make identification faster. Moreover,

ROI can avoid detecting circles that are not necessary. For example, the following image shows the situation that there are traffic signs in both sides of the street, but only right side one is useful. Using ROI can avoid detecting any traffic signs in the left side of the street.



Figure 14: Screenshot of Carla environment with traffic signs in both side of the roads

3.2.2 Identify Speed Number

Once the circle for traffic sign has been found, we can use Tesseract OCR (Optical Character Recognition) to detect the speed number on this traffic sign. Tesseract OCR is considered one of the most accurate open source optical character recognition engines. Pytesseract is the python library for Tesseract OCR that can use python script on Tesseract OCR. In this research project, we can set Tesseract OCR as only detecting digital numbers.

3.3 Detecting Obstacles

In this research project, three objects have been considered as obstacles, traffic lights, other vehicles and pedestrians. At the very beginning, I planned to train three different classifiers for these three obstacles. To collect training dataset, I manually drove vehicle

in Carla Environment and save the images from Carla. Then, find the required areas from images and crop to small images that only have designed objects. The following images shows the dataset for traffic lights in Carla Environment.

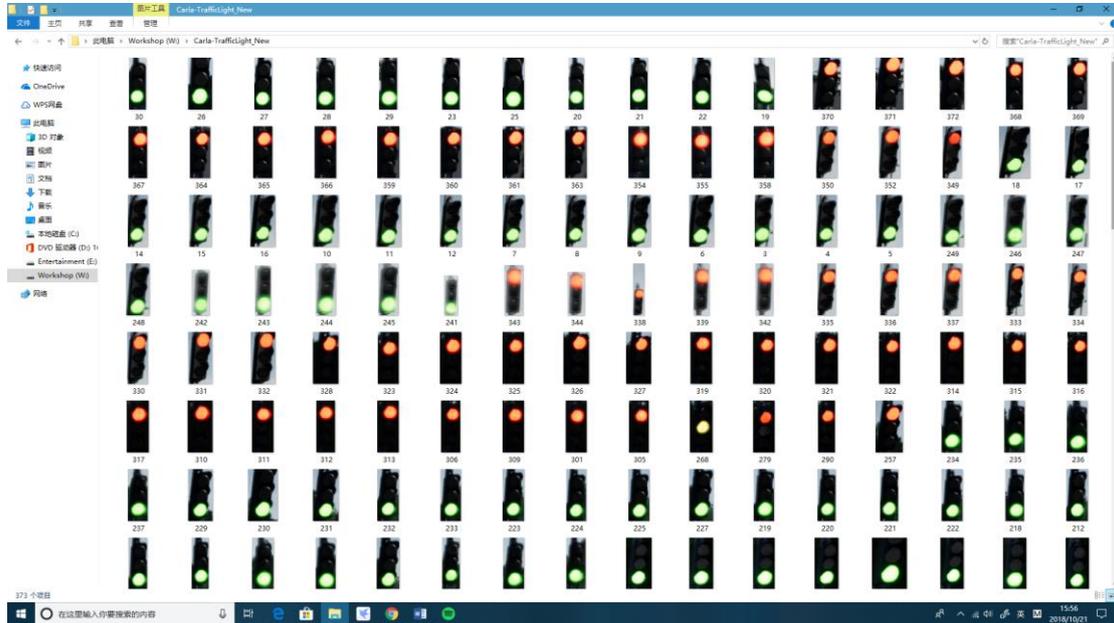


Figure 15: Collected dataset for traffic lights

After the first classifier finished, which is traffic light classifier, I realized that this approach was not working. The reason is that running three classifiers in the same time is not realistic, this method takes up too much graph memory. Then, I decided to train one classifier by three objects, which means traffic lights, other vehicles and pedestrians can be classified by one classifier. However, this method can't work, either. The most important reason why this method is not working is that the size and shape for these three objects are different. For, example, when randomly select one image for vehicle, one image for traffic light and one image for pedestrian, the size of vehicle is 492×243 , the size of traffic light is 85×198 , the size of pedestrian is 69×216 .



Figure 16, 17, 18 Selected images for vehicle, traffic light and pedestrian

From the images above, the ratio of width and height for the images of traffic lights and pedestrians can be considered as 1:3, but the ratio of width and height for the images of other vehicles is approximately 2:1. It is impossible to resize all the image into same designed size. And if I resize all the image with the ratio of width and height as 1:3, this classifier can't identify vehicles accurately.

To find the solution that using only one classifier to identify all the obstacles accurately, I decided to use real-time object detection system YOLOv3.



Figure 19: The demo for YOLOv3

There are over 100 classes of objects that can be identified by YOLOv3, such as car, person, train, plane, motorbike etc. But in this research project, only traffic light, car, bus and person need to be found in Carla environment.

3.3.1 Traffic Light

Two attributes for traffic light are important for autonomous vehicle, current light state and the distance. To identify the distance between the traffic light and autonomous vehicle, I selected to calculate the size of the box that covered traffic light. If the size of the box is larger than a designed number, for example 2000 (this number is determined by the size of Carla window), this means the traffic light is close to autonomous vehicle and vehicle should respond accordingly to this traffic light.

To identify which light is on for one traffic light, I chose to calculate the average RGB in the founding box of traffic light. Average RGB method means add up R, G, B respectively while traversing the image, then for total R, G, B, divided by the size of the image, the result is the average RGB for this image. For example, the following three images are traffic lights that I randomly selected. The first is in green, and the average RGB is (80, 95, 87). The second is in red, and the average RGB is (78, 62, 56). The third is in yellow, and the average RGB is (32, 32, 22).



Figure 20, 21, 22 Traffic light in green (80, 95, 87), red (78, 62, 56) and yellow (32, 32, 22)

If the green light is on, the image of traffic light is close to green, so the average RGB for this image, G is the maximum. If the red light is on, the image of traffic light is close to red, so the average RGB for this image, R is the maximum, If the yellow light is on, the image of traffic light is close to yellow, and yellow is made up of red and green, so the average RGB for this image, R and G is close (absolute value is less than 2), B is smaller than R and G.

Overall, if autonomous vehicle detects that there is a traffic light that is close to the vehicle, and the red light is on for this traffic light, the vehicle will stop in this situation.

3.3.2 Vehicle

When identify the vehicle, which include car and bus in Carla environment, except identifying the distance between detected vehicle and autonomous vehicle, which is same as traffic light, we also need the relative position of detected vehicle for autonomous vehicle. For example, the vehicles in opposite lane are not important for autonomous vehicle no matter how close it is. If the vehicle is in front of the autonomous vehicle, and the position of the detected vehicle will be reached later for autonomous vehicle, autonomous vehicle should respond accordingly to this detected vehicle.

To determine whether detected vehicle is necessary for autonomous vehicle to respond or not, I create a region on Interest (ROI), which is a trapezoid in front of the autonomous vehicle. The following image shows the ROI for detecting vehicle.



Figure 23: ROI area for vehicle and pedestrian detection

Overall, if autonomous vehicle detects that there is a vehicle that is close to the vehicle, and the central point of detected vehicle is in the area of ROI, the vehicle will stop in this situation.

Due to the generalization of YOLOv3, the hood of autonomous vehicle will be detected as vehicle, and this “detected vehicle” are always in the ROI area, to avoid that, it is necessary to crop the bottom before detection.

3.3.3 Pedestrian

The operation for detecting pedestrians is exactly the same as detecting other vehicles. If autonomous vehicle detects that there is a person that is close to the vehicle, and the central point of detected person is in the area of ROI, the vehicle will stop in this situation.

The way for determining the distance between pedestrian or vehicle with autonomous vehicle are the same as traffic light, which is the size of the box that covered vehicle or pedestrian. It is obvious that the size of vehicle is much larger than pedestrian, so the designed size for vehicle is 25000 and designed size for person is 5000 (this number is determined by the size of Carla window).

4 Results for Virtual Autonomous Vehicle

Unfortunately, due to the performance issue of the computer, especially the performance of GPU is not enough, three parts of classifiers and detectors can't run in the same time. When running Carla Simulator, this will take up a certain amount of graph card performance, the driving behavior classifier, real-time object detection and traffic sign detection will all take up graph card performance when running them, so when too much graph memory has been occupied, CUDA boost is disable at that situation and AlexNet and YOLOv3 can't run without CUDA. To show the result of this research project, I will show every part of detection individually.

4.1 Driving

After training for over 30 hours by AlexNet, a model had been created as the classifier for driving in Carla Environment. To use this driving classifier, it is necessary to launch Carla by windowed mode, because windowed mode Carla requires less on graph memory. Then, set window size of Carla window as 800×600 , and put Carla window at upper left corner of the desktop screen, this is the area python script will read.

Trained driving classifier will read Carla window and take this as input image, then give predicted operation and output on the screen. The following images are the screenshots about driving. Left is Carla window, right side is python script result, it shows the current operation at this image and current fps.

This image shows that the autonomous vehicle is at the middle of the road and the front of the vehicle is straight road, so the operation is W, which means go forward.



Figure 24: Result 1 for driving

When the autonomous vehicle is approaching the intersection, it needs to reduce the speed, otherwise, the vehicle will get out of control at the curved road. At this situation, the operation is S, which means brake, or no keys.



Figure 25: Result 2 for driving

The image below shows the situation that the autonomous vehicle is passing the right turn curved road. The operations are D, which means turn right, and WD, which means go forward right.



Figure 26: Result 3 for driving

4.2 Traffic Sign

In order to identify traffic sign in Carla Environment, it is necessary to launch Carla by windowed mode, because windowed mode Carla requires less on graph memory. Then, set window size of Carla window as 800×600 , and put Carla window at upper left corner of the desktop screen, this is the area python script will read. This python script will display the area it read, then find and display the circle in this area. Once circle has been found, Tesseract OCR will identify the speed number that inside the circle. The following images are the screenshots about traffic sign detection. Left is Carla window, right side is detected window that created by python script, it shows the found circle and detected speed number.

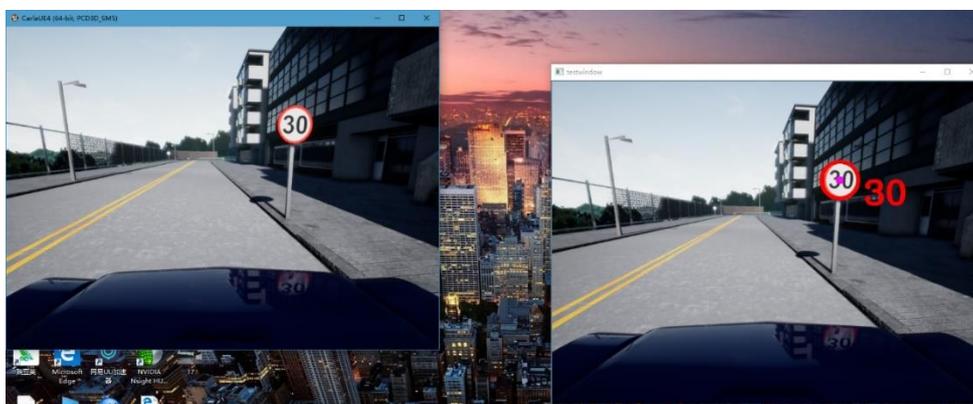


Figure 27: Result1 for traffic sign



Figure 28: Result 2 for traffic sign



Figure 29: Result 3 for traffic sign

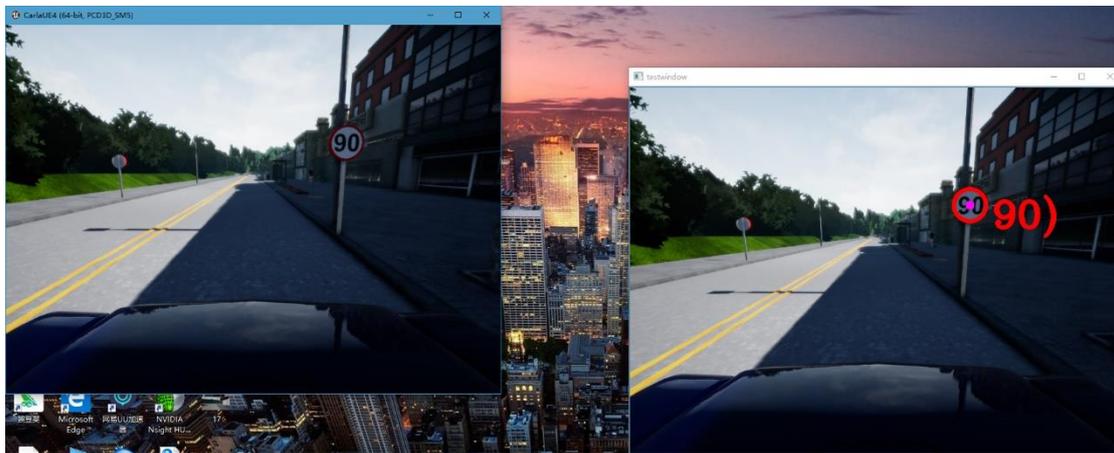


Figure 30: Result 4 for traffic sign

At the second image and last image, there is a traffic sign at the left side of the road, the circle will not be identified because of ROI, ROI is the right side of the image, so all the traffic signs that for opposite roads will not be identify.

For the last image, right part of the outside circle was identified as right parenthesis. This issue can be solved with regular expression.

4.3 Obstacles Detection

YOLOv3 real-time object detection system has been adopted for obstacles detection. To use this obstacles detection, same as driving classifier and traffic sign detection, it is necessary to launch Carla by windowed mode, because windowed mode Carla requires less on graph memory. Then, set window size of Carla window as 800×600 , and put Carla window at upper left corner of the desktop screen, this is the area YOLOv3 object detection will read.

Obstacles detection will read Carla window and take this as input image, then this python script will create a window that display the reading image, with detected objects in this reading image. At the upper left corner of the window, fps will be showed there. If a vehicle or a person is at the ROI area, which is in front of the autonomous vehicle, there will be a warning at the upper left corner of the windows, and the type of warning will be given. If a traffic light is detected and the red light is on, there will be a “stop” on the area of traffic light. The following images are the screenshots about object detection. Left is Carla window, right side is detected window, it shows the detected objects in Carla window.

The image below shows the situation that there is a vehicle in front of autonomous vehicle, and the distance between two vehicles is close. At the upper left corner, there is a warning for this situation. In the same time, the red vehicle far away from the vehicle has been identify.



Figure 31: Result 1 for detection

In this image, two vehicles, a person and the traffic light have been found. And the white vehicle in front of autonomous vehicle is close to the vehicle, so there is a warning at the upper left corner.



Figure 32: Result 2 for detection

In this image, one vehicle, one person and one traffic light have been found. The traffic light is close to autonomous vehicle, and it is in red, so there is a “stop” in the detected area of traffic light. The vehicle in front of autonomous vehicle is close to the vehicle, so there is a warning at the upper left corner.



Figure 33: Result 3 for detection

For the image below, in this situation, there is a warning showed at the top when there is no vehicle in front of the autonomous vehicle. It is because of the SUV behind the yellow mini cooper. At this situation, autonomous vehicle is heading slightly left of the road, and the central point of SUV is in the ROI area and this vehicle is close to autonomous vehicle, that's the reason for the warning.



Figure 34 Result 4 for detection

In this image, vehicle behind the jump has been detected even though over half of the vehicle has been blocked by jump. The traffic light is in red and close to autonomous vehicle, so there is a “stop” in the detected area of traffic light.

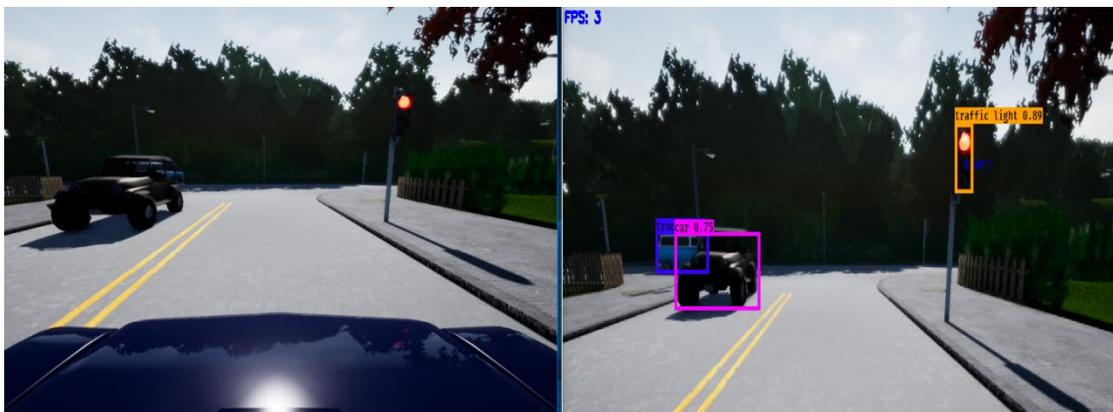


Figure 35: Result 5 for detecting

According to the fps information that showed at the upper left corner of these five images, the fps is always less than 5, but regular fps for YOLOv3 on GTX GeForce TITAN X is approximately 45, this showed that the GPU performance is not good enough for this method.

5 Future Work and Summary

At the end of this research paper, I will summarize this research project, and list the future work of this research project.

Based on the research results that presents above, we can make the conclusion that: firstly, it is available that use human operation and autonomous vehicle's camera for driving behavior training, and the obtained classifier can make a vehicle drive like human in closed internal roads environment such as Carla environment. Secondly, it is working that using edge detection and Tesseract OCR for traffic sign detection if there is only speed limit traffic sign. Thirdly, using YOLOv3, object detection for obstacles detection is possible in virtual environment, and the real-time performance is very good. Finally, the approach of using average RGB for determining the light state of traffic lights is working. In conclusion, if the GPU's performance is good enough, based on all the approaches above, an autonomous vehicle in Carla environment is achieved that can self-driving like human and detecting other vehicles, pedestrians, light state of traffic lights and traffic sign content in real time.

Moreover, there are some parts of this research project that could improve in future works: Firstly, and most importantly, find a way to reduce the requirement of GPU. Based on the research from Larabel (2016), they achieved that using CPU on training AlexNet, and it is available to use Tensorflow object detection system that works on CPU version Tensorflow. Secondly, find a way to control the speed of vehicle when use driving classifier for driving. There is a defect from this method that single image can not represent current speed of the vehicle, which is an important element for vehicle operation. To control the speed of vehicle, the ideal method is using PID controller. Finally, adopt the regular expression approach on speed number detection. Even though I have set on Tesseract OCR that detect digital number only, but sometime there is still symbol, such as right parenthesis that in figure 30.

References

- Ackerman, E. (2012). Study: Intelligent Cars Could Boost Highway Capacity by 273%. Retrieved from <https://spectrum.ieee.org/automaton/robotics/artificial-intelligence/intelligent-cars-could-boost-highway-capacity-by-273>
- Anderson, J. (2016). Self-Driving Vehicles Offer Potential Benefits, Policy Challenges for Lawmakers. Retrieved from https://www.rand.org/pubs/research_reports/RR443-2.html
- Du, X., Ang, M., & Rus, D. (2017). Car detection for autonomous vehicle: LIDAR and vision fusion approach through deep learning framework. *2017 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*. doi: 10.1109/iros.2017.8202234
- Evans, P. (2009). Ford to introduce Active Park Assist in 2009. Retrieved from <https://newatlas.com/ford-to-introduce-active-park-assist-to-automate-parallel-parking-in-2009/10642/>
- Hawkins, A. (2017). Waymo is first to put fully self-driving cars on US roads without a safety driver. Retrieved from <https://www.theverge.com/2017/11/7/16615290/waymo-self-driving-safety-driver-chandler-autonomous>
- Kalra, N. (2016). Test-Driving Alone Cannot Demonstrate the Safety of Autonomous Vehicles. Retrieved from https://www.rand.org/pubs/research_reports/RR1478.html
- McAleer, M. (2017). Audi's self-driving A8: drivers can watch YouTube or check emails at 60km/h. Retrieved from <https://www.irishtimes.com/life-and-style/motors/audi-s-self-driving-a8-drivers-can-watch-youtube-or-check-emails-at-60km-h-1.3150496>
- Model S | Tesla. (2012). Retrieved from <https://www.tesla.com/models>
- SAE International. (2014). Retrieved from <http://templatelab.com/automated-driving/>
- Stenquist, P. (2014). In Self-Driving Cars, a Potential Lifeline for the Disabled. Retrieved from <https://www.nytimes.com/2014/11/09/automobiles/in-self-driving-cars-a-potential-lifeline-for-the-disabled.html>
- The Milwaukee Sentinel. (1926). 'Phantom Auto' Will Tour City.
- Thorpe, C., Hebert, M., Kanade, T., & Shafer, S. (1987). Vision and Navigation for the Carnegie-Mellon Navlab. *Annual Review Of Computer Science*, 2(1), 521-556. doi: 10.1146/annurev.cs.02.060187.002513

- Whitwam, R. (2014). How Google's self-driving cars detect and avoid obstacles - ExtremeTech. Retrieved from <https://www.extremetech.com/extreme/189486-how-googles-self-driving-cars-detect-and-avoid-obstacles>
- Filipowicz, A., Liu, J., & Kornhauser, A. (2017). *Learning to recognize distance to stop signs using the virtual world of Grand Theft Auto 5* (No. 17-05456).
- Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal Of Research And Development*, 3(3), 210-229. doi: 10.1147/rd.33.0210
- Mannila, H. Data mining: machine learning, statistics, and databases. Proceedings Of 8Th International Conference On Scientific And Statistical Data Base Management. doi: 10.1109/ssdm.1996.505910
- Machine learning. (2018). Retrieved from https://en.wikipedia.org/wiki/Machine_learning#cite_note-5
- MNIST Demos on Yann LeCun's website. (1988). Retrieved from <http://yann.lecun.com/exdb/lenet/>
- Zhang, W., Itoh, K., Tanida, J., & Ichioka, Y. (1990). Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied Optics*, 29(32), 4790. doi: 10.1364/ao.29.004790
- Five Class CNN Models:Lenet, Alexnet, Googlenet, VGG, DRL. (2017). Retrieved from http://www.sohu.com/a/134347664_642762
- Krizhevsky, A., Sutskever, I., & Hinton, G. (2017). ImageNet classification with deep convolutional neural networks. *Communications Of The ACM*, 60(6), 84-90. doi: 10.1145/3065386

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Memon, B. (2014). Kresic N, Mikszewski A | Hydrogeological conceptual site models: data analysis and visualization. *Environmental Earth Sciences*, 72(2), 623-623. doi: 10.1007/s12665-014-3268-8

tesseract-ocr. (2018). Retrieved from <https://github.com/tesseract-ocr>

Larabel, M. (2016). Running Caffe AlexNet/GoogleNet On Some CPUs Compared To NVIDIA CUDA - Phoronix.

Retrieved from https://www.phoronix.com/scan.php?page=news_item&px=Caffe-CPU-GPU-Fun