COMP9517 Computer Vision Project Group Report

An Yan
School of Computer Science *and* Engineering(CSE)
The University of New South Wales (UNSW Sydney)
Sydney, Australia
z5142177@ad.unsw.edu.au

Hao Fu
School of Computer Science and Engineering(CSE)
The University of New South Wales (UNSW Sydney)
Sydney, Australia
z5253457@ad.unsw.edu.au

Kevin Wang
School of Computer Science and Engineering(CSE)
The University of New South Wales (UNSW Sydney)
Sydney, Australia
z5281453@ad.unsw.edu.au

Junyu Zhang
School of Computer Science and Engineering(CSE)
The University of New South Wales (UNSW Sydney)
Sydney, Australia
z5304897@ad.unsw.edu.au

Yunqi Wang
School of Computer Science and Engineering (CSE)
The University of New South Wales (UNSW Sydney)
Sydney, Australia
z5324194@ad.unsw.edu.au

***Abstract*—During the past few years, there has been lots of great progress in the field of autonomous and self-driving technology. People have been gained plenty of convenience due to this huge improvement. People have been used advanced control systems, artificial intelligence, and computer vision technologies to help vehicles steer themselves on roads without direct human assistance. In this article, Anaconda OpenCV and Visual Studio Code are used as a development environment and Python is used as a main computer language to talk about a new, simple visual module by using computer vision techniques learned in this course which can implement these functions for a self-driving car. These methods can meet most requirements and has good adaptability.**

Keywords—computer vision techniques, on-board camera, autonomous, OpenCV, Python, HOG, SVM, YOLO, Deep Learning.

I. INTRODUCTION

Since the 1980s, Intelligent Transportation System(ITS) has been developed rapidly. Surface with the increasing demand for information in modern transportation, ITS can not only use the significant power of computers to perform more scientific and efficient transportation planning and management at the level of traffic management, it can also obtain richer traffic information by utilizing advanced automatic equipment. During autonomous driving, whether it can perceive the dynamic environment of autonomous vehicles plays a necessary and important role. Over the past decade or in traditional, a variety of expensive sensors such as LiDAR or MMW(Millimeter-wave) radar and GPS were used to better perceive the surroundings and the information about the positions and motions of the autonomous cars. However, in this project, only cameras were given to replace all of these advanced sensors and there are three tasks for our group to implement.

For the first and the second task, we are supposed to implement some Python solutions which will be used to estimate the distance and the velocity of each recognized vehicle relative to the given camera in each testing image. In addition, the estimated distance and velocity are showed above the bounding box of each detected vehicle. As we all know, the development of the internet have promoted the accelerated implementation of artificial intelligence technology in vehicle identification scenarios. However, in real life, there are many problems when doing accurate identification of vehicles since all the vehicles on the roads are always in relatively complex environment. At first, there are many disturbances such as changeable backgrounds, occlusions and shadows in the natural environment. Secondly, weather factors such as cloudy skies, haze and light changes also play an important role. In addition, camera heights and shooting angles also effect the deformations and scale on vehicles. All of these factors we are considered to make sure we can have a good result by applying the method we used.

During autonomous driving, how to accurately locate the position of the vehicle is crucial. This is why we need to implement lane detection in the third task. Lane detection provides necessary localization information to the vehicle's system controls. We have implemented a Python method to precisely detect all the driving lanes in each image and overlay them on the input image. Humans can directly observe where the lane line is through our smart eyes, but it is such a difficult problem for camera to achieve this so we need to teach the camera to recognize and locate the location of the lane line from a single image. Also there are a lot of factors effect our experiment such as the color of the line, how dark is the color, the straightness of the line and how can we tell the camera to detect it. What else, the factors we mentioned in task1 and 2 such as the shadows of trees and buildings and vehicle occlusion still exist in this task. Above all, lane lines vary in degree of wear and the change of the width are also worth to considering. There are also different ways to solve these questions and we will show in the following contents.

TuSimple dataset are mainly used for these three tasks. There are a set of 1074 2-second-long video clips for us to train and a set of 2692-second-long video clips for us to test our experiment during the first and second task. The input is a clip that contains the 40th frame file path and the output should be a image contains the recognized vehicles with a bounding box including the distance and velocity. There is a little bit different for the last task, the Lane Detection dataset and the associated annotated frames are provided for us to conduct the tests. Also the input frame should be the 20th frame in a clip instead of the fortieth one. The camera height is 1.8 meters. What else, any available external data sources are also allowed to be tested.

II. Literature Review

A. A Simple Image-based Object Velocity Estimation Approach [1]

In this research, Chu, H. C., & Yang, H. [1] have talked about a feasible and practical method to estimate object velocity by just using a single camera instead of some expensive and professional equipment. There are two main processing modules: one is an object detection module(ODM) the other one is a background measurement module(BMM). The ODM captures an image of the object and the BMM estimates the movement distance of the object[1]. The experiment was performed by four steps—object detection, background measurement, moving duration and velocity estimation, each of them is applying the fundamental image processing technology and based on a simple webcam and a known length referred object.

The result of this experiment shows that it is a simple-based, practical and feasible object velocity estimation approach by just performed using a simple camera besides two phenomena which can reflected the results to some extent. The estimated error is increasing when detected object moved too fast and the camera was too far away from the background.

In general, this approach is a bit similar as our tasks, it is performed using a simple camera and applying some image processing methods that we learned in this course such as grayscale image, noise removal, median filter, geometric theory and some mathematical measurement methods which we can use part of these methods for our reference in our tasks.

B. Real-time vehicle distance estimation using single view geometry[2]

This paper shows a single view geometry-based approach to estimate the distance for vehicles which is lightweight, intuitive, independent of data. The authors found that real-time logging of all the information on the roads is very crucial at first and tried to applying active sensing since it is a minimal post processing. They had explored plenty of data-driven and geometric approaches in literature[2]. It is worth mentioning that Multi-View Stereo(MVS) which takes into account the geometry of the features in the scene which can provide depth for each pixel through matching multiple images. This method is such a new part for us by learning computer vision technology.

Three dataset were used in this research, Kitti, nuScenes and Lyft level5. In addition, they applied some deep-learning based method like ADAS module of the lane detection which is arose in our third task. As a huge project, it is not only directly measure the distances of the vehicles on the roads, nut also covered the lane detection part which is the third part for our project. More details about how to settle the camera and measure the distance are showed in the 3.2 part. Horizon estimation, image rectification and parallel line fitting are really useful and practical for our own tasks. For evaluation, monocular dashcam images are mainly applied in this experiment to estimate relative distance of the vehicles which is using a single image to gauge 3D measurements using perspective geometry[2].

It is an simple, data independent algorithm and one-time computation which means it costs no computational expense and it outperforms end-to-end DL approaches.

C. Lane detection using lane boundary marker network with road geometry constraints[3]

This paper presented an intuitive and practical algorithm of lane detection by using a lane boundary marker network based on road geometry constraints. The experiment results of this method on three different public datasets showed that remarkable improvements on the current methods such as using plenty of features from low-level to deep-level extracted from convolutional neural networks[3] and even some deep-learning algorithms. Especially the results on TuSimple dataset which is provided for our own tasks showed that the accuracy is around 98% and passing some DL based algorithms such as SCNN and SAD-Nets[3] by more than 1.5%.

Authors from this article found that it is critical to improve the ability of lane detection in identifying and ensuring safe. Besides, properly understanding the surroundings scene and road conditions is necessary for autonomous vehicles. However, the current methods such as ow-level edge features, histogram analysis, Linear Hough Transform or the combination of these are not able to solve the problem of occlusion. In addition, they are relatively easily fooled by non-lane edges or some false conditions such as road curbs, shadows appearing in the images[3]. These conditions also showed when we doing the third task, we are trying to solve these by applying more algorithms and doing more experiments,. Even for deep learning based solutions such as VPGNet, SCNN, JLBNet, although the results are more reliable than the current basic methods because these solutions tend to learn the background of the road and ignore the occlusions, it is still difficult for DL based algorithms to detect the lane in some conditions ,when there is no lane marking or the evidence are removed on the road.

Therefore, a novel algorithm was proposed that is to find the key points on the lane boundary by applying marker network and using road geometry to estimate an inverse perspective mapping. The main principle to validate the detected lanes is to combine equidistant and parallelism property of the lanes jointly and the fixed lane width to fit the lanes. To address such cases, they firstly used a Convolutional Neural Network(CNN) which is based on encoder-decoder to find the key points across the lane boundaries. Then applying scene geometry IPM(Inverse Perspective Mapping) method on the detected markers or key-points are fitted on the rectified points which is contained in the second step—Camera Initialization, it also includes horizon estimation and lane width initialization. After the above steps , it went to Post-Initialization step which include Equidistant Parallel Line/Curve Fitting and Detecting Weak or Missing Lane Boundaries. These steps are very useful and practical and it gave us a really good result when we applied them to our own task.

In conclusion, this algorithm is efficient and practical. The important thing is this method is modular so we can combine it with other lane detectors or other DL based lane detection methods to improve the ability and get better results. This also provides plenty of evidence, examples and models such as Linear Hough Transform, SCNN and Inverse Perspective Mapping for our group to do more research and improve our own task—lane detection.

D. Lane detection using Color-Based Segmentation[4]

This paper presented a new method of lane detection based on color information and color segmentation. This method is kind of easy by just applying a few steps but it is useful and easy to implement.

At the beginning, they chose a interested area to located the center line from the image and using lane marking to distinguish the lane from other features such as vehicles, trees and pedestrians. Then applied statistical methods in this color image to find out a threshold which was used to distinguish the lane boundaries. After getting the detected lane marking pixels, they were trying to figure out which one is lane boundary by using two main properties: vertical and horizonal. Finally, use a mathematical quadratic function to approach the lane boundaries. The form is : f_k: $a_k y^2 + b_k y + c_k = x$

This new simple lane detection algorithm is different from gradient method, it can easily get rid of the environmental influence such as sunshine, sunlight, pavement and obstacles. Using least square method is much more simple to approach the lane boundary without any previous information and with higher accuracy rather than use Hough Transform or other complicated models. Most important thing is this method can deal with the edge conditions better such as broken lines or curved line which is a good example for our group to learn and improve our own project.

E. Simple Robust Road Lane Detection Algorithm[5]

A simple robust road lane marker detection algorithm[5] was proposed to test the accuracy in real-time configuration. This method is mainly divided into three parts: pre-processing, post-processing and lane recognition[5] and consists of the optimization of two deep-learning algorithm Canny edge detection and Hough Transform. These algorithms are all useful and exercisable for our project and we applied them to optimization our processing.

There are several basic image processing technology applied during the first part—pre-processing such as gray-scale, dilation and erosion for getting much more useful information. Then they used Canny edge detection and Hough transform for post-processing. For the former, it recognized the lines and edges in the smoothen image. For the latter, it was used to connect the different and discontinuous lines. Hough transform is much easier to detect the straight lines and curves.

The formula is:

$$T = x \cos d + y \cos d$$

where T is the radius from the origin and d is the angle.

The last step is to figure out the possible both left and right lane markers and averaged them to get the results.

III. METHODS

A. Task 1

The monocular camera generates a one-to-one relationship between the object and the image. With this principle, the relationship between focal length (fx, fy), size of object in pixel (top, bottom, left, right), Principal Point Offset (cx, cy) and distance (distance x, distance y) can be deducted.

Using the intrinsic matrix of camera, focal length and Principal Point Offset are known. With the bounding box detected, the size of object can also be settled. Therefore, with the principle of similar triangles, distance x and distance y can be obtained.

These can be expressed below:

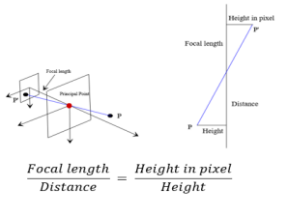

Figure 1. similar triangles

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 2. intrinsic matrix

B. Task2

To estimate the velocity of vehicles, two milestones need to be achieved, track the vehicle from different frames in a particular clip and estimate the velocity based on the relative distance.

1) Vehicle Tracking

From individual part, we have already detected and bounded the regions which predicted as contain cars. Nevertheless, the problem is, to estimate the distance of a car moved, we need to predict which bounding boxes refer to the same car. In order words, we need to trace a particular car from different frames.

a) Peak signal-to-noise ratio

Peak signal-to-noise ratio (PSNR) could be used to compute the difference between images. It is defined via the mean square error. If there given a noise-free a noise-free m ×n monochrome image and its noisy approximation K, MSE is defined as $MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}[I(i,j) - K(i,j)]^2$ and PSNR could be defined as $PSNR = 10 \cdot \log_{10}\left(\frac{{MAX_I}^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$. As shown in the above formulas, the less value of MSE is, the higher the value of PSNR. So the more considerable value of PSNR indicates the more negligible difference between the target image and the origin. In this task, we tried to calculate values by comparing all predicted bounding boxes. By selecting the pair bounding box with the most significant PSNR value in two sequential frames, we could verdict that what bounding boxes contain is the exact vehicle since they have the slightest difference. More details and the specific process of this part will be showed in the section Results and Discussion.

b) DeepSort Algorithm

Unlike PSNR, DeepSort algorithm put more emphasis on feature extracting. DeepSort use a 8-dimension vector $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ to describe the status of a particular locus in a time interval. u and v indicate the center of predicted bounding box. γ means the radio of length and width. The rest of four represent the velocity details in corresponding space.

DeepSort considered relationship of both motion and appearance. It use Mahalanobis distance to describe the relationship of the given bounding box and the another predicted by tracker, combined with creating a gallery for each target object to restore the feature vectors to judge whether a

pair of bounding box contain the same car. We will show the details of implementing method in the following section.

c) Overlap Ratio

By observing all images from the train set, we found that in a sequence of frames, the position of each detected car is not changing much. For this method, we try to calculate the overlap ratio of bounding boxes in two sequential frames since if two bounding boxes contain the same car, it is likely that the position of the next bounding box is near.

C. Task 3

1) Traditional methods—Canny edge detection and Hough Transform:

After looking through all the given papers and anatomize all the experiments and methods, our group decided to use Canny edge detection and Hough Transform as our traditional methods since these two algorithms are the traditional and practical ones for lane detection part. Thus, we tried these two methods first and try to optimize them by solving some extant problems. For Canny edge detection, it works with monochromatic images which means basic image processing such as gray-scaling, noise reduction and image smoothing are necessary. This method is easily influenced by some environment conditions such as sunlight, the color of the road and other obstacles, which is the main disadvantages of this method. After extracting the features of the road, Tough Transform was applied. The main advantage of this method is to detect the straight lines and curves, which also show its shortcomings at the same time, when there are crooked lanes, or the lanes are covered by something else such as wheels or pedestrians. More details and the specific process of this part will be showed in the fourth section—Experimental Setup.

For the third task, we also implemented another traditional algorithm to detect the lane which is based on color information. This method is to abstract the features first to distinguish from other obstacles by using lane markers. Then segment lane marking with histogram-based method. This method can deal with complex environment. It is robust in the presence of noise, shadows, the absence of the lanes and other obstacles. However, as a traditional method, there is still some problems that it cannot solve. For example, if there are no white lanes in the image, this method will not work anymore.

2) Deep-Learning methods—SCNN

Spatial CNN(SCNN) generalizes traditional deep layer-by-layer convolutions to slice-by-slice convolutions within feature maps[6]. It could catch more spatial information on rows and columns than the traditional way. It convolves alone the four directions sequentially and does convolution results accumulation on the slices in each directions.

The entire network architecture is showed below in figure 3.

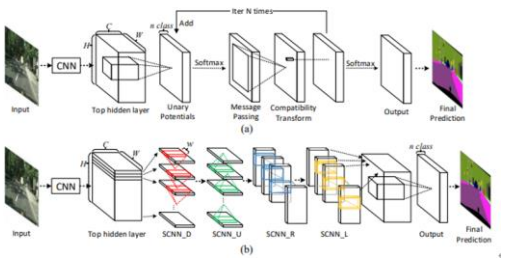

Figure 3

Figure 3(a) is the MRF/CRF based method. Figure 3(b) is the implementation of Spatial CNN[12].

3) Deep-Learning methods—Ultra

For the previous methods, there are some problems that these traditional methods are not able to solve them so that here we also implement Deep-Learning method to improve our result.

With the development of more and more lane detection research, we can find that traditional methods are not enough to deal with some conditions especially "different colors of the lanes ". Thus for this research area, more demand is based on track retrieval on the semantic layer which can make lane detection more robust and can also maintain a good recognition rate on roads with multiple occlusions, fast light changes and complex rad conditions.

Ultra [9] is a faster and more practical method since it defines the lane detection as a collection of finding the position of tracking lines in certain rows in the image which is based on "row-based classification". According to the experiments, this method can apparently accelerate the detection process. More details will show in the following sections.

The overall of the architecture of this method is showed in the following figure 4.

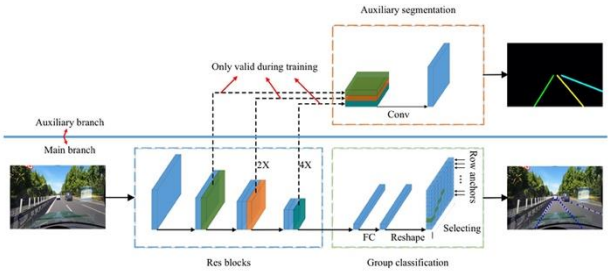

Figure 4. Overall of the architecture of Ultra[13]

IV. Experimental Setup

A. Detection

1) HOG and SVM

HOG is histogram of oriented gradient. It is the initial way that we used in the individual part. In this method, we uses HOG to extract positive and negative features and feed these features into the SVM classifiers to predict whether this feature is car's feature. Then, when we got the classifier, we used it to detect the testing dataset by using multi-scale sliding windows. The main drawbacks are the accuracy and speed. HOG could not get enough features to detect and sliding windows is a slow way to search cars. Figure 5 is the flow of the algorithm. This method needs to change the parameters for every pictures if we want to detect cars accurate. If fixed parameter values are set, the results are unstable.

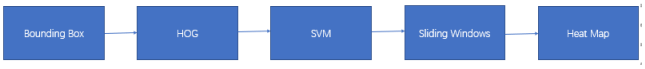

Figure 5. The flow of HOG and SVM

2) AlexNet and SVM

To get high accuracy, we changed the way of extracting features from hog to AlexNet which is the backbone of RCNN. AlexNet could extract features with 4096

dimensions. Figure 6 is the flow of this algorithm. However, the results are worse than using HOG. The reason is that it's hard to get enough features from the given training dataset.

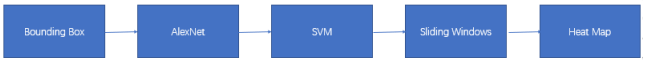

Figure 6. The flow of AlexNet and SVM

3) YOLOv3

We changed the way of extracting features from AlexNet and SVM to YOLOv3. We changed the format of the dataset since YOLO needs fixed format and chose the given supplementary dataset as the training dataset. Figure 7 is the steps of using YOLOv3 to detect cars. The YOLOv3 codes are based on [11]. The code for reorganizing the dataset were written by us (generate_dataset.py). The code of VeloEval class is based on the [10].

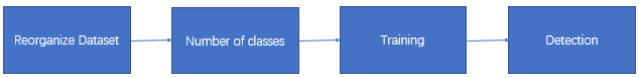

Figure 7. The flow of using YOLOv3

B. Task 1—Traditional methods experimental setup:

Firstly, we use the similar triangle to get the distance in both x-axis and y-axis, the specific diagram is showed in figure 8 and figure 9.

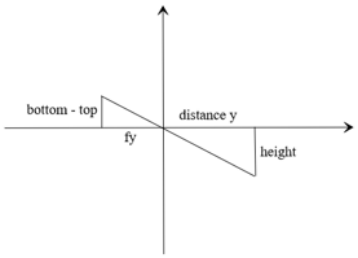

Figure 8.

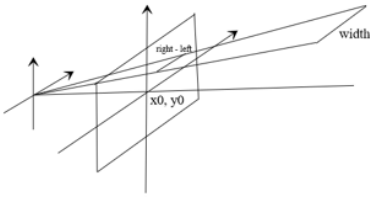

Figure 9.

For the distance in y-axis, we used a similar triangle to compute the result with the two triangles showed in figure 9,

we have:

$$\frac{distancey}{f_y} = \frac{height}{bottom - top}$$

Formula 1.

Which can be wrote as:

$$distancey = \frac{height * f_y}{bottom - top}$$

Formula 2.

For the distance in x-axis, our first thought was calculating it by a similar formula as distance in y-axis. However, the error of this method was large.

$$\frac{distancex}{f_x} = \frac{width}{right - left}$$

Formula 3.

Thus, we changed our plan and use another method. The middle point of the bounding box was applied to compute the result. As showed in figure 0, the specific value between real-world distance and the distance in the detected image is as same as the specific value between width in real-world and pixel value, which can be wrote as the formula:

$$\frac{distancex}{middle_x - cx} = \frac{width}{right - left}$$

Formula 4.

Which equals to:

$$distancex = \frac{width * (middle\_x - cx)}{right - left}$$

Formula 5.

To reduce the estimation error as much as possible, we used multiscale method. For the vehicles which are far away from the camera, the width was set to 1.6. For the ones that near the camera, the width was set to 2.2.

C. Task 2—Traditional methods experimental setup:

In this task, Tusimple dataset is passed for training and estimation. By collecting results from task one we gathered all bounding boxes detected as contain a vehicle in it. Note that the result of YOLO stores the center coordinate on x-axis and y-axis respectively, which means we need to transform it into the specific position of top, bottom, left and right edge.
For PSNR ratio calculation, what we need to feed is two partial image regard as image in bounding boxes retrieved from the origin picture. PSNR requires us to pass images with same size, so as a preprocessing we resized these partial images to a same scale and collect the ratio as result. Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, provided the bit depth is 8 bits, where higher is better. The processing quality of 12-bit images is considered high when the PSNR value is 60 dB or higher. For 16-bit data typical values for the PSNR are between 60 and 80 dB. Acceptable values for wireless transmission quality loss are about 20 dB to 25 dB [6].

For DeepSort Algorithm, which is different as PSNR calculating, it takes bounding box presented with coordinate of center of x and y axis respectively and the width and length of bounding box as input, to extract features from the origin image.

D. Task 3—Traditional methods experimental setup:

We used TuSimple dataset to do this lane detection experiment. The description of the TuSimple dataset is shown in table 1. The main environment is highway which is an easy scenario to detect with. It does not have many obstacles on the road, such as pedestrians, trees, and parked cars. The training dataset has 3626 video clips, each clip has 20 frames. The testing dataset has 2782 video clips which have same frames with training dataset.

Frames	*Train*	*Test*	*Resolution*	*Lane*	*Environment*
TuSimple	72020	55640	1280*720	<6	highway

Table I. Data Description

1) Canny+Hough

For task three, to evaluate our Canny edge detection and Hough Transform Algorithm, Tusimple dataset was used to do the experiments. To simplify our random access memory and save time as much as possible, and at the same time it is not necessary to have high resolution in lane detection part, we firstly shrink out images into the half size. Then we ran the demo codes on the groundtruth to have a cursory understanding of the results and what the output should be like.

Now some image processing were applied to the detect images. Gray-scaling, noise reduction, Gaussianblur are all commonly used in this pre-processing part. On the one hand, these steps can help us extract the features of the detected images, On the other hand, the images after doing these processing are easily for later process. After finishing all the pre-processing steps, Canny edge detection was used to detect the lane and figure out the lane marking as showed in figure10.

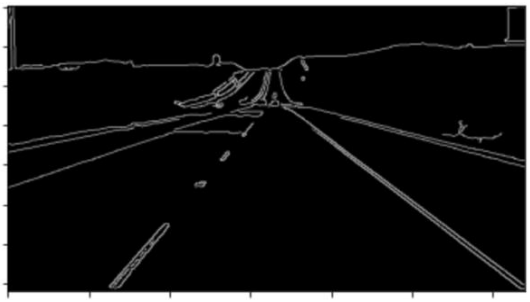

Figure 10. canny edge extraction result

Then we tried to use the mask firstly but there were a lot of factors that highly influence the results such as the position of our detecting camera and the position of other vehicles. In addition, the noise from other obstacles such as trees and fences. The result is showed in figure 11.

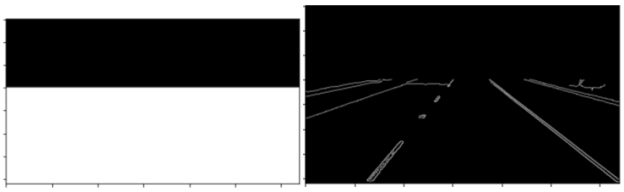

Figure 11. mask shape and the masked image

As a result, we tried a better method to extract the region of interest in the next work (see color based for details).

Hough Transform was used to provide useful information for the experiment. It can detect the lanes and then connect discontinuous lines. Here we will show several main steps of Hough Transform and the relative merits.

Figure 12 shows the basic processing of Hough Transform.

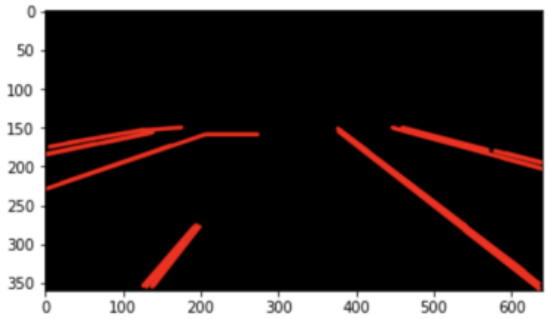

Figure 12. Hough lines

Then we sift out some hough lines that with improper slope to get the suitable lane lines. Next, we extended the filtered Hough lines to fit the lanes better. The result is showed in figure 13.

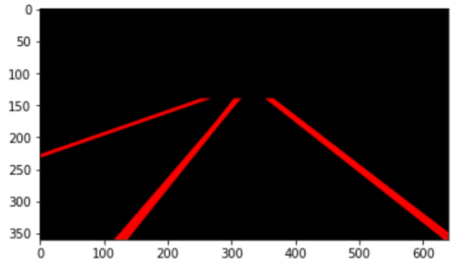

Figure 13. extending Hough lines

Then we narrow the lines into one pixel so we are already to generate the lane ground truth labels. Here is the result in figure 14.

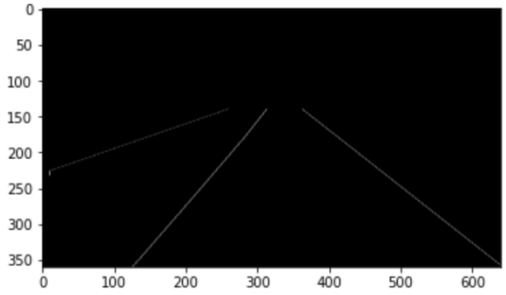

Figure 14. narrowed lines

Finally, generate the ground truth points shows in figure 15.

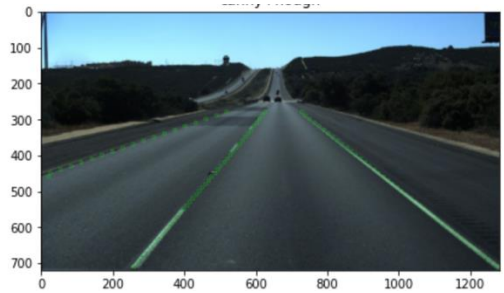

Figure 15. pred-ground truth

2) Color-based segmenation

Another algorithm we implemented for this task is Color-based method for lane detection. As the first algorithm, the pre-processing step is also necessary. While doing this step, there is one thing need to be pointed is to compare different channels to make both the white lines and yellow lines most obvious. As them shows in figure 16, the white lines and

yellow lines are both obvious in Blue channel. As a result, we chose the blue channel to do the next operations.

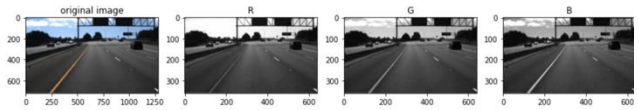

Figure 16. comparison between different channels

Then, by performing the gray-scaling histogram, we are able to find a an appropriate threshold to separate other objects and the brighter lanes and sky areas.

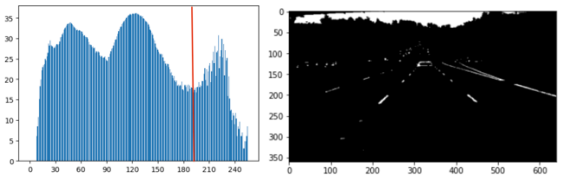

Figure 17. threshold and the binary image

Next, we performed another gray-scale histogram to show the white areas that based on the height of the image. Since the sky area is much bigger than lanes, by finding the height of the sky on the image and removing them, we can retain the image of the road to the maximum extent.

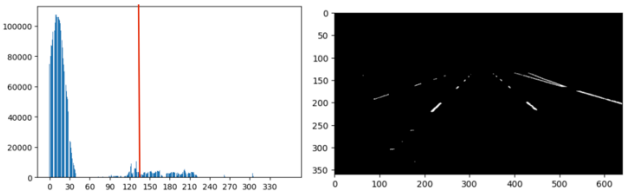

Figure 18. Region of interest

The next following work is to apply the extracted region of interest to the previous Canny + Hough method, but not continue the color based. Because the color-based method can only extract the colored lanes, which is not suitable for the TUSimple dataset (there are lots of lanes made up of gaps instead of white and yellow lines). But the interest area extraction can make up for the shortcomings of previous mask-method.

E. Task 3 Deep Learning methods Experimental Setup:

To improve existing problems of traditional methods and to get a good result, our group has implemented the other two Deep-Learning methods Ultra and SCNN for task three.

1) SCNN

The principle of SCNN as the information showed above in Deep-Learning methods—SCNN part. In this method, the training model and prediction process are showed in figure 19. The author of SCNN used LargeFOV network as the backbone and inserted the Spatial CNN in it. It feeds the image into LargeFOV network firstly. Then, feeding the obtained feature map into Spatial CNN. Finally, it could get the probmaps and each lane mark. This experiment is implemented on Pytorch1.8.1. The SCNN codes are based on the [12].

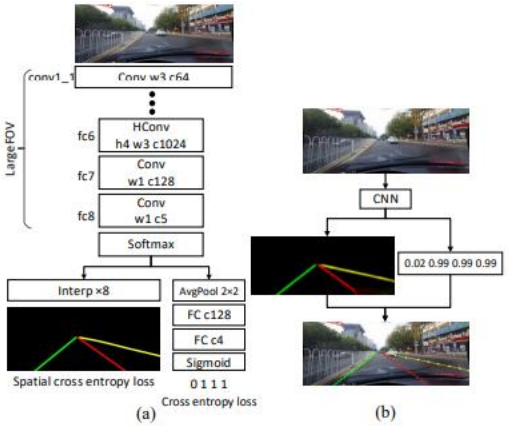

Figure 19. The Structure of SCNN and prediction

2) Ultra

Ultra is a method based on the row-based selecting, combines with contextual and global information, which makes this method works more intelligent, more robustness than traditional methods. Moreover, it is also faster than many mainstream Deep Learning Lane detection methods. Firstly, Ultra is selecting the correct locations of lanes on each predefined rows using global features instead of segmenting every pixel of lanes based on local receptive filed. In this way, the computation complexity is reduced to a minute range. (see details in figure 20)

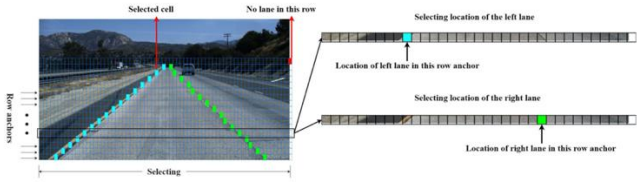

Figure 20. Illustration of selecting lanes and the selecting of rows

Apart from that, Ultra also purposed a pre-constraint method to fit the Lanes better. Ultra uses a new Lane structural loss that combines with smoothness and shape features:

$$L_{sim} = \sum_{i=1}^{C}\sum_{j=1}^{h-1}\left\|P_{i,j:} - P_{i,j+1:}\right\|_1,$$

Formula 6.

$$L_{shp} = \sum_{i=1}^{C}\sum_{j=1}^{h-2}\left\|\left(Loc_{i,j} - Loc_{i,j+1}\right) - \left(Loc_{i,j+1} - Loc_{i,j+2}\right)\right\|_1$$

Formula 7.

By row-based selecting and constraining features into high-level semantics and low-level visual information, Ultra can get better running speed and non-visual-clue Lanes.

The testing code of Ultra-method is based on [13]. The pre-trained model is also provided in[13]. The source code for testing model we used are: ‘evaluation/ eval_wrapper’, ‘model/ parsingNet’, ‘utils/ dist_utils’, ‘utils/ dist_print’ and ‘test’.(but we submited all the source code in files).

V. RESULTS AND DISCUSSION

A. *Task 1:*

The estimation error of first method, evaluated by the DEMO code provided in project specification.:

Position Estimation error (Near): 54.11109

Position Estimation error (Medium): 411.23172

Position Estimation error (Far): 948.52168

Position Estimation error total: 471.28817

The estimation error of second distance calculation method is shown blow, the evaluation by using DEMO code provided in project specification.:

Position Estimation error (Near): 9.16053

Position Estimation error (Medium): 44.48934

Position Estimation error (Far): 91.81101

Position Estimation error total: 48.48696

Moreover, the information on the detected image is showed below (only shows one bounding box for recognizable).

Figure 21. Visual result of distance

B. *Task 2:*

1) Vehicle tracking

By taking PSNR method, due to the distortion on the car far from the camera, the bounding box are not distinctly different from other regions we bounded. On conclusion, we cannot judge whether two bounding boxes contain a same car.

Figure 22 PSNR value between bounding boxes

By calculating the overlap ratio, most of detected cars could be related.

Figure 23 Red boxes indicates the start position and green boxes indicates the end position

By taking the Deep-Sort algorithm, it also works well on tracking these vehicles and correctly label them.

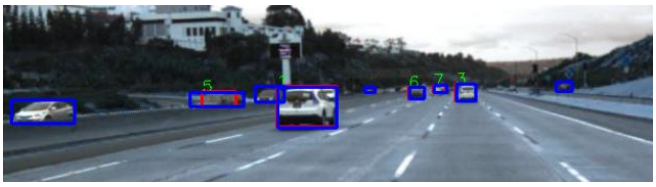

Figure 24 Labeled Vehicles generated by Deep-Sort Algorithm

2) Velocity Estimating

Based on the tracking result from part 1, we divide the relative position by the time passed after the origin frame to the end. The relative step could be calculated by the same method implemented in task 1. As a pre-condition, there is some assumption we taken:

- The relative acceleration of each car is unchanging.
- No affine transformation applies on camera.

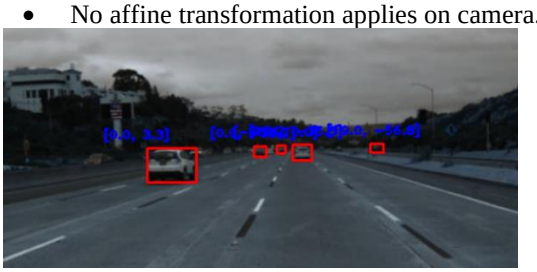

Figure 25 Velocity Estimation Result.

The evaluation formula could be defined as

$$accuracy = 1 - \frac{|S_{ground-truth} - S_{estimation}|}{S_{ground-truth}}$$

Formula 8.

As a result. We have got 0.63427682349743 as accuracy on average when compares to the groundtruth annotations.

C. *Task 3:*

1) Visual results

To better reflect the differences of various methods in Task3, we tested the methods above on four same images. The visual results show down below:

a) Traditional method

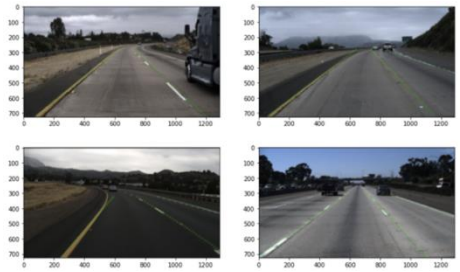

Figure 26. Visual results of Traditional method

b) Ultra-method

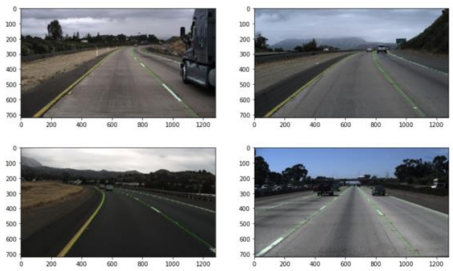

Figure 27. Visual results of Ultra-method

c) SCNN method

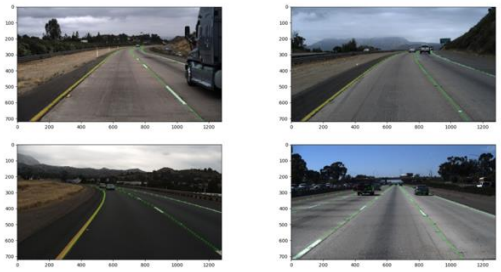

Figure 28. Visual results of SCNN

Moreover, we also tested the traditional method (Canny+Hough) and Deep-Learning method (Ultra-method) in different Lane conditions:

d) Straight lane:

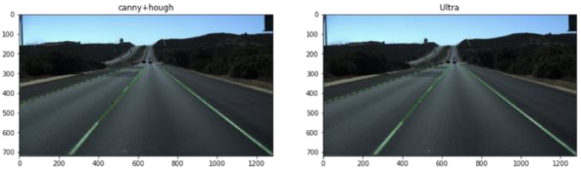

Figure 29. Straight lane result

a) Curved lanes:

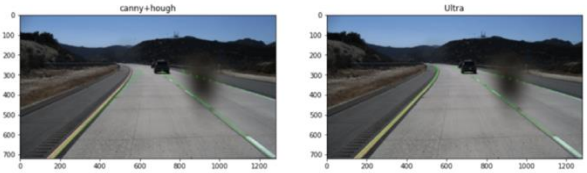

Figure 30. Curved lane result

b) Complex compound lanes:

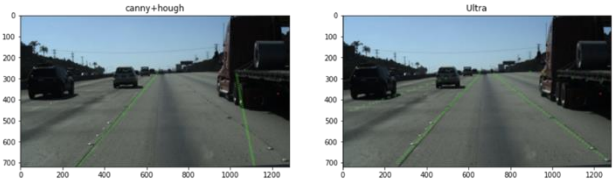

Figure31. Complex compound line

From above we can see that the traditional method (Canny+Hough) for curve lanes and complex lanes is relatively poor. Because it cannot effectively drop the influence of noises on the lane and the lanes of non-visual clue. However, based on the semantic understanding and learning, the deep learning methods can effectively predict the lanes position especially in the case of occlusions.

2) Metrics and numerical results

The evaluation formula is:

$$accuracy = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}},$$

Formula 9.

$$FP = \frac{F_{pred}}{N_{pred}},$$

Formula 10.

$$FN = \frac{M_{pred}}{N_{gt}}$$

Formula 11.

Where C_{clip} is the number of correct points in the last frame of the clip in TUSimple dataset? S_{clip} is the number of requested points in the last frame of the clip in TUSimple dataset. F_{pred} is the number of wrong predicted lanes and N_{pred} is the number of all predicted lanes. M_{pred} is the number of missed ground-truth lanes in the predictions, and N_{gt} is the number of all ground-truth lanes.

	Methods		
	Traditional	***Ultra***	***SCNN***
Accuracy	0.7319	**0.9545**	0.946
FP	0.3306	0.1339	0.063
FN	0.4367	0.0089	0.068

Table II. The Evaluation Results of Three Methods

Table 2 shows the evaluation results of three ways by using TUSimple dataset. After applying the Deep-Learning methods, the experiment result is improved a lot. It also shows that Ultra performs better than SCNN.

VI. CONCLUSION

A. Task 1:

From the experiment, we can find the error is acceptable but not that ideal. Error in estimating cars from medium and far distance is considerable higher than the vehicles closed. This method is practical and reasonable, but if there are another better algorithms of preprocessing the image such as generating a bird view or solvePnP , we will get a better result.

B. Task 2:

Both calculating overlap ratio and applying Deepsort could generate a well tracking result. But they have disadvantages respectively:

1) If a vehicle moves toward a different direction from camera, calculating the overlap ratio may cannot track the target since it changes a distinct position compares to the origin.

2) We must set a general threshold to filter all the ratios on overlap, this action could be vulnerable since the correct standard to judge whether two bounding boxes belong to a same car is different.

3) The ground-truth provided by Tusimple didn't provide all vehicles on the image, which means a set of features is abandoned, result in DeepSort could hardly track vehicles far from camera.

C. Task 3:

From the experiment and the comparison of traditional methods and Deep-Learning methods, it is clear that the traditional methods are not enough robust and it showed in these particular points:

1) If there are obvious obstacles on the roads, the traditional methods cannot clearly and directly recognize the lane lines. Sometimes, misidentification appeared.

2) For straight lines, traditional methods can mostly identify the lines. However, fitting process is not applied well when there are some curved lines.

3) As for the field of objective detection, methods need to combine total context and high-level semantics to learn in order to get better robustness.

4) Although Deep-Learning methods may also have some disadvantages, such as over reliance on training data. Compared with the traditional methods, it still has a great improvement in the recognition effect.

To sum up, the Deep-Learning methods have a huge and non-substitutable advantage and it will continue to be one of the mainstream directions of the object detection based on artificial intelligence in the nearest future.

VII. Contribution of Group Members

At the beginning, in order to make sure we can give a rigorous report, our group decided to separate the whole project into four parts, which means each of us is able to think about the task independently and finish their work efficiently.

Yunqi Wang-z5324194:1.The experiment, implementation and evaluation of traditional methods (Canny +Hough method and Color-based method). 2. Testing and evaluating the Ultra model on TUSimple.

Kevin Wang-z5281453: Task1: analyze, experiment set up, implementation and evaluation.

An Yan-z5142177:Implementing PSNR, overlap ratio and applying DeepSort Algorithm and evaluate the result in task 2.

Hao Fu- z5253457: Most of the preparatory work such as the introduction part and literature review. Also dispose and collate most of the writing part for the report.

Junyu Zhang – z5304897: Developing two new ways to detect vehicles, including AlexNet and SVM, YOLOv3. Testing and evaluating the SCNN model on TUSimple in task3.

VIII. References

[1] Chu, H. C., & Yang, H. (2014, April). A simple image-based object velocity estimation approach. In Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control (pp. 102-107). IEEE.

[2] Ali, A., Hassan, A., Ali, A. R., Khan, H. U., Kazmi, W., & Zaheer, A. (2020). Real-time vehicle distance estimation using single view geometry. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 1111-1120).

[3] Khan, H. U., Ali, A. R., Hassan, A., Ali, A., Kazmi, W., & Zaheer, A. (2020). Lane detection using lane boundary marker network with road geometry constraints. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 1834-1843).

[4] Chiu, K. Y., & Lin, S. F. (2005, June). Lane detection using color-based segmentation. In IEEE Proceedings. Intelligent Vehicles Symposium, 2005. (pp. 706-711). IEEE.

[5] Low, C. Y., Zamzuri, H., & Mazlan, S. A. (2014, June). Simple robust road lane detection algorithm. In 2014 5th International Conference on Intelligent and Advanced Systems (ICIAS) (pp. 1-4). IEEE.

[6] Pan, Xingang, et al. "Spatial as deep: Spatial cnn for traffic scene understanding." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.

[7] Faragallah, Osama S.; El-Hoseny, Heba; El-Shafai, Walid; El-Rahman, Wael Abd; El-Sayed, Hala S.; El-Rabaie, El-Sayed M.; El-Samie, Fathi E. Abd; Geweid, Gamal G. N. (2021). A Comprehensive Survey Analysis for Present Solutions of Medical Image Fusion and Future Directions. IEEE.

[8] Wojke. N, Bewley. A & Paulus (March, 2017). Simple online and realtime tracking with a deep association metric. IEEE

[9] Qin, Z., Wang, H., & Li, X. (2020). Ultra fast structure-aware deep lane detection. arXiv preprint arXiv:2004.11757.

[10] TuSimple <https://github.com/TuSimple/tusimple-benchmark.git>

[11] YOLO.<https://github.com/FLyingLSJ/PyTorch-YOLOv3-master.git >

[12] Spatial CNN.< https://github.com/harryhan618/SCNN_Pytorch.git >

[13] Ultra-Fast-Lane-Detection. < https://github.com/cfzd/Ultra-Fast-Lane-Detection.git >

[14] DeepSort <git@github.com:xiaorun2345/yolov5-deepsort.git