Group Projet Report-Team Visual HD

* COMP 9517 Computer Vision

Haonan Zhang
z5151812
University of New South Wales
Sydney NSW Australia
line 5: z5151812@ad.unsw.edu.au

Hanrui Tao
z5237012
University of New South Wales
Sydney NSW Australia
z5237012@ad.unsw.edu.au

Fei Xu
z5239235
University of New South Wales
Sydney NSW Australia
z5239235@ad.unsw.edu.au

Yiyang Ke
z5244460
University of New South Wales
Sydney NSW Australia
z5244460@ad.unsw.edu.au

Dannie Zhao
z5266355
University of New South Wales
Sydney NSW Australia
z5266355@ad.unsw.edu.au

***Abstract*— Vehicle relative velocity estimation and lane recognition are the important issues in autonomous driving, and more and more projects have begun to explore them in depth. There are two goals of this project. Firs one is to estimate the relative velocity of a specific vehicle from a sequence of images and the second one is to detect all the driving lane in each image. lane detection. In this project, a light-weight approach by OpenCV was implemented to estimate the velocity based on the information labels created YOLO v5. Another contribution is to establish a serious of functions to detect driving lanes. The outputs of position and velocity estimation approach is able to annotate the parameters of relative position and velocity of each vehicle in video clips. The accuracy of lanes detection is 0.72, the false-positive is 0.35 and the false-negative is 0.56.**

Keywords— relative distance measurement, lanes detection, similar triangles, smoothing algorithm

I. INTRODUCTION

Currently, driverless cars are one of the hottest areas in the research field. These self-driving cars use computer vision and deep learning to solve car problems, including detecting the relative position and speed of the vehicle, and detecting lane lines. The current traffic system has been highly dependent on real-time traffic information obtained by video surveillance equipment. Based on video surveillance, it can automatically detect vehicle moving targets, extract vehicle velocity and other information [1-2]. Perceiving the dynamic environment of an autonomous vehicle is main task to realize autonomous driving. Information about the location and the movement of the agent in the environment around the vehicle plays an important role in the movement plan. Traditionally, this type of information is sensed by expensive distance sensors (such as LiDAR or MMW radar). Camera sensors provide a handy and powerful alternative to LiDAR or radar-based distance sensors. Although LiDAR systems can provide very accurate measurement results, they can also fail under adverse environmental conditions, such as fog, snow, rain or even exhaust gas [3-4]. However in this project, the camera is fixed which made the problem becomes less complicated, because angle measurements can be obtained using a calibrated camera system, and speed estimates can be easily determined from these measurements. On the other hand, other information such as the focal length of camera, foreground background segmentation and self-motion was required. The result of recent research [5] shows that it is feasible to estimate the ego-motion and disparity map in a single-camera image through the structure of motion. However, it still has some limitations. Semantic segmentation of scenes is a basic problem in computer vision, and deep neural networks have recently been used to solve them in MS COCO 2015 and 2016 segmentation competition [6-7]. For example, the information provided in the KITTI benchmark [8]. The object scene stream is designed to estimate dense 3D sports fields that carry extremely valuable information about the geometric constellation of a given scene in their temporal evolution.

In object detection field, traditional detection systems usually use classifiers to perform detection and evaluate it on the different positions and scales of the test image. For example, the Part Model (DPM) implement a sliding window approach, in which the classifier moves at evenly spaced positions on the whole picture [9].

State-of-the-art methods such as R-CNN involved a region proposal method to choose the potential bounding boxes, and then implement the classifier on those boxes. After finish classify, post-processing is used to streamline the bounding box, rescore the box and remove repeated detections relied on other objects in the image [10]. However, these the efficiency of these methods are low and do not have any space to improve because each part of the pipeline need to be trained respectively. In 2016, the YOLO was created which is a new object detection method [11]. The architecture convolutional neural of YOLO shows in Fig. 1. This model performed fast and easy to optimize on end-to-end because the entire inspection pipeline is a single network.

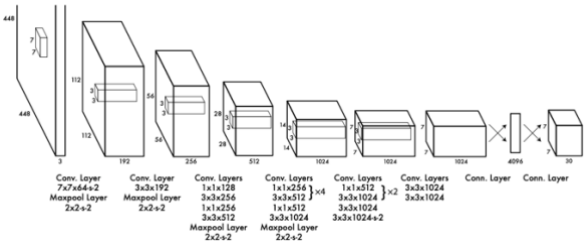


Fig.1. The architecture convolutional neural of YOLO

This report implements a two stages process for velocity estimation. The pre-trained YOLO v5 was implemented to detect vehicles.

The first step is to estimate the relative distance between the camera to each vehicle in images. Based on the principle of pinhole (monocular) camera, the object and the image have a one-to-one relationship [12-13], the distance from the object

to the camera is able to be deduced by using fx, fy, cx, cy provided by Project Specification. In the second step, based on the result of first step, the velocity formula and the time of the video clips are employed to compute the velocity.

Lane line intelligent recognition is a crucial and basic technical component in the field of autonomous safe driving. Compute vision-based recognition technology captures road images through on-board cameras and uses algorithms to recognize lane lines in the image to remind the driver of vehicle changes conditions of the lanes to avoid dangers. Therefore, the aim of driving lanes detection is to find all the drive lanes in the images. This project implements Gaussian blur [14] which is a common methods of image pre-processing with a Gaussian function to reduce image noise and reduce detail. In 1986, three criteria of Canny were proposed for judging margins performance: SNR standards, localization accuracy standards and unilateral response standards, and derived the best Canny edge detection operator [15-16]. This project also involved Hough Transformation to get the straight line on the degree image after edge detection. Hough created a method which called Hough Transform to identified lines in the images efficiently in 1962[17]. In shape analysis, the Hough transform is considered a powerful tool, even in the presence of noise and occlusion, the Hough transform is able to provide good outcomes [18].

The database used in this project of estimation real distance and velocity is from TUSIMPL[19] which provided training dataset includes 3626 short driving sequences in freeway traffic, recorded by a single HD (1280x720p) camera. Each video split into 20 clips. However, the vehicles are only annotated on the 20th flames. An annotation contains the bounding box is a list consisting of four characteristic coordinates [left, right, top, bottom] specifying the vehicles border coordinates of the bounding box in pixels. The images are in good and medium weather conditions and different traffic conditions in different daytime. The number of highway roads are more than 2 lanes. Fig. 2 shows an example image from the data. The database used in driving lanes detection also has 3626 video clips with annotated frames in same surrounding condition. The annotation contains the information of the lane in each image which include a width values list of lanes and a height values of lanes.

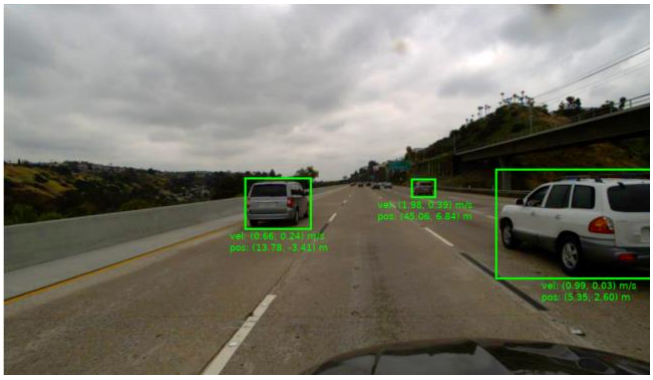


Fig.2. A sample image of location and velocity lables

II. METHODS

A. *Estimation of relative distance of other vehicles*

After running the YOLO v5 [20] model to detect vehicles, it will automatically generate a TXT file for each detected image which shows in Fig. 3 which contains location labels in XYWH format.

Two ways are widely used to represent bounding boxes (Fig.2), the first is using coordinates (x1, y1) and (x2, y2), the second is using center coordinate (X,Y) and W(width) and H(height) [21].

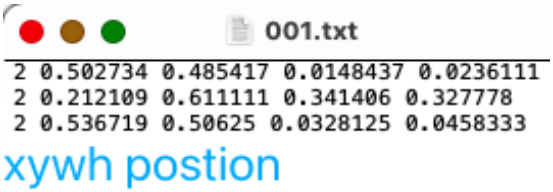


Fig.3. Principle of the method for dx

To compute the distance, the XYWH position should be converted to bounding box coordinates (x1,y1) and (x2,y2)[22].

$$x1 = (X - \frac{W}{2}) \times ImageWidth \quad (1)$$

$$x2 = (X + \frac{W}{2}) \times ImageWidth \quad (2)$$

$$y1 = (Y - \frac{H}{2}) \times ImageHeight \quad (3)$$

$$y2 = (Y + \frac{H}{2}) \times ImageHeight \quad (4)$$

After getting the bounding box coordinates of detected vehicles, the similar triangles [12] are used to obtain the distance between camera and cars in the images. Firstly, from the Fig. 4, the line on the left represents the image, and the line in the middle represents the camera. On the right is the height of the car. Distance X is the distance from car in the real world to the camera in X axis. Distance Y is that in Y axis. Right, Left, Bot, Top are the pixels position for the car box in the image. Fx and Fy are the focal length parameters. Cx and Cy are the principal point. Both Fx, Fy and Cx, Cy are represented in pixel. The triangle on the left and the triangle on the right

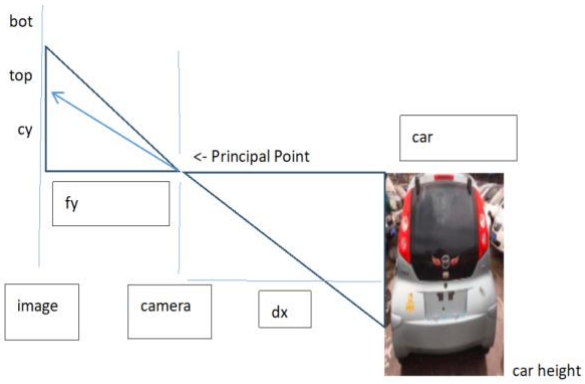


Fig.4. Principle of the method for dx

have the same proportions, so the principle of similar triangles can be used. It means that the radio of car height and the distance between Cx to bot in pixel can be calculated.

Therefore, we could the distance x using the ratio of these two similar triangles:

$$dx = \frac{Car\ height \times Fy}{|Bot - Cy|} \quad (5)$$

Next, we use the information in Fig. 5 to get the distance y value. We also use the ratio of similar triangles to Fig. out the results.

$$dy = \frac{(Right - Cx) \times dx}{Fx} \quad (6)$$

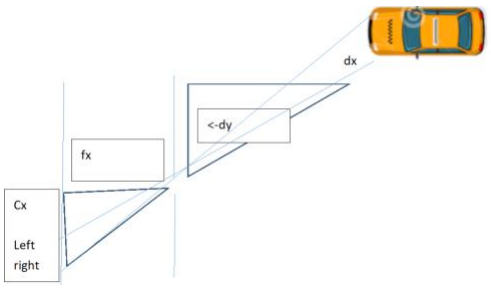


Fig.5. Principle of the method for dy

B. *Estimation of relative velocity of other vehicles*

Due to the dataset was recorded on highways [19], thus the velocity will not change much in 2 seconds duration. Fig. 6 shows a simple distance change between the 1st frame and the 40th frame.

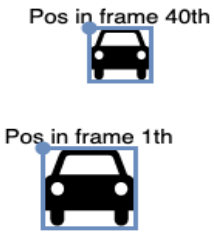


Fig.6. A simple distance change between the 1th frame and the 40th frame

After calculating the position of the last frame and the first frame using task1 method, the following formula can be used to compute velocity.

$$v = \frac{Pos\_40th - Post\_1th}{2s} \quad (7)$$

C. *Lanes Detection - Preprocessing of Dataset*

a) Region of interest (ROI): After observation, the images in the dataset. The structure of the region in each image has a similarity. In each image, from 0 – 300 pixels are normally sky and trees. The mask method has been chosen in this task, which is using black pixels to cover regions that are not the region of interest, then generate the region of interest image. The region of interest image is given following in Fig. 7.

b) Gaussian smoothing: In order to reduce image noises, we apply the Gaussian smoothing algorithm which is an

Fig.7. Region of Interest.

image process method. This algorithm is using the Gaussian function to compute the transformation applying to each pixel in the image. Equation (8) below illustrates the Gaussian function[23]:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (8)$$

In this task, using Gaussian blur to remove noise can avoid extracting useless features in the next steps. The following Fig.8 shows the image after the Gaussian smoothing process.

Fig.8. The result after Gaussian Smoothing

c) Color Space Selection: The combination of RGB and HSV color spaces is applied in task 3. This is because the light and texture in the picture environment are diverse, it is difficult to obtain ideal recognition results with only a single color channel. Through observation, most lanes in the image are either white or yellow. In some cases, white Lane lines and yellow lane lines exist in the same image. In this consideration, white color and yellow should be chosen from different color spaces. As Fig. 9 given below, white color from RGB color space and Yellow color from HSV color space have been selected with specific thresholding.

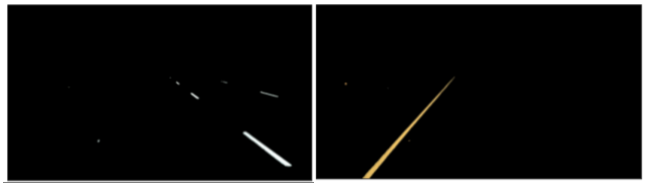


Fig.9 The results of RGB White Filter and HSV yellow filter

As shown above, lane lines in white and yellow color can be extracted clearly in this method. Then apply the

combination of RGB and HSV color spaces, the example of combination result given in Fig. 10.

Fig.10. The result of combined RGB and HSV

D. Lanes Detection - Detection

a) Canny Edge Detector: The Canny edge detector is implemented in the detection stage in task 3. The canny edge algorithm is patented by John F. Canny in 1986. This algorithm detects edges in a wide range of images by using a multiple level operation algorithm[24]. The Process of the Canny edge detection algorithm can be divided into 4 different steps: reduce noises, find the intensity gradients, non-maximum suppression, Hysteresis Thresholding. In this project, Chosen Gaussian Blur to reduce noises, which has been done in the pre-processing stage mentioned above. As for finding intensity gradient, the argument 'Sobel kernel size' has been set to 3. In the Hysteresis Thresholding step, two threshold values are needed - minVal and maxVal, these two values have been set to 60 and 100 in the experiment.

b) The canny edge detector is essential in this lane detection task since it can find the boundary position of the driving lane. Normally, the boundary of the driving lane is determined by the contrast between the road surface and the painted line. The canny edge detector performs excellently in finding driving lane edges[25]. As Fig. 11 given below, the method of canny edge detector can determine well the driving lane edge in the input image

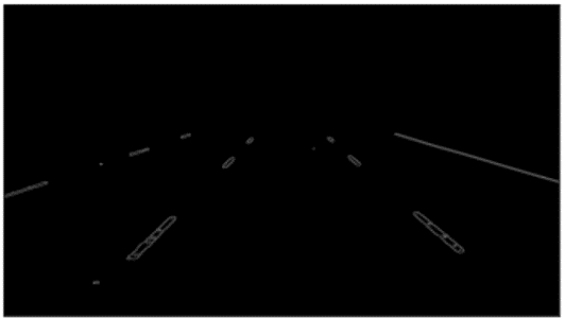


Fig.11. Edge Detection

c) Hough Transform: Hough transform is a feature extraction technique used in computer vision that was invented by Paul V. C. Hough[26]. In the Hough Transform algorithm, an edge detected image is required. In this task, the edge image generated from the previous canny detection step is used as binary image input. Then apply the Hough Transform to detect whether a line exists in the edge image. Fig .12 gives the example image that shows the lines determined by Hough Transform.

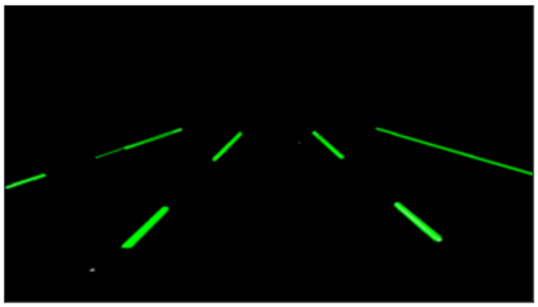


Fig.12. Hough Transform

E. Lanes Detection - Optimization

Average of slope and intercept: It is vital to apply an optimization algorithm since the result obtained from the detection stage are multiple lines for each driving lane, which should not be the final result. Averaging and extrapolating are chosen to optimize the results, the reason is that each driving lane lines in the image have a different slope, using this can grouping and dividing lines. First, calculate the slope by using (9) and the intercept by using (10) of each line, the formula of this function is given below:

$$\text{Slope} = \frac{y_2 - y_1}{x_2 - x_1} \tag{9}$$

$$\text{Intercept} = y_1 - (Slope * x_1) \tag{10}$$

Where (x_1, y_1) and (x_2, y_2) are the coordinate of the start point and the end point in each line separately. Then grouping these lines into right lane lines and left lane lines according to the slope. The last step is to calculate the average and intercept of the slope for all the lines in each lane. Draw the line with the average slope and the average intercept, as the final line for the lane. The results after optimization given below in Fig. 7.

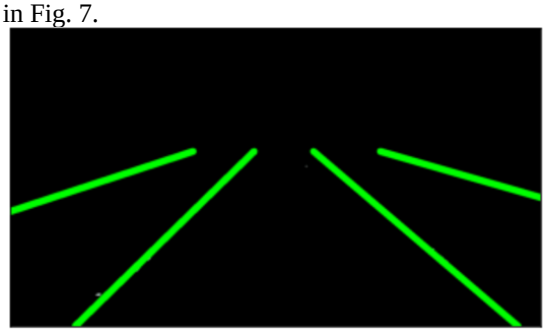


Fig.13. Optimized lane lines

III. EXPERIMENTAL SETUP

A. Estimation of relative distance of other vehicles

First step, using the YOLO v5[20] model to detect and label the cars in the images. Please note that the submitted code does not contain any function or code of the YOLO v5 model, the inputs of task1 and 2 are original images and the label txt files generated by yolov5.

In the next step, the values from annotations files are used to calculate a suitable car height for this task. Because this model is going to get a box that's pretty much the same as the notation on the annotation. Fig. 14 is an example of Vehicle detection in yolov5 model. The camera parameters(Fx, Fy, Cx, Cy) have been provided. If we ignore the Cx and Cy, the value (Dx,Dy) is not reasonable. Combined with the pixel position and camera parameters of the vehicle in the image

and the assumed height of the vehicle, a relatively accurate distance between the vehicle and the camera can be obtained. Thus, for the same vehicle in an image, we want the position value to be as close as possible to the value in the annotation. velocity.

For Evaluation, mean square error and average difference are used in this task. Mean square error is calculated by taking the square of the difference between the position information of the same vehicle in the annotation.

Average difference is the sum of the difference between predicted values and their corresponding ground truths.

Mean square error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{n=1}^{n}(\mathrm{X}_i - x_i)^2 \qquad (11)$$

Average difference:

$$\mathrm{AD} = \frac{1}{n}\sum_{n=1}^{n}|\mathrm{X}_i - x_i|^2 \qquad (12)$$

X_i are the position values in annotations, x_i are the predicted position value.

Fig.14. Vehicle detection by yolov5

B. *Estimation of relative velocity of other vehicles*

In this task, the velocity estimation is based on position estimation. The first image and last image in a clip will be chosen first. The methods of the task of vehicle distance will be used to compute position for each detected vehicle because of different environments and cars in a clip, so if the first image is not suitable for calculation, the other images, such as the 20th or the 30th image will be used and the time between two images must be adjusted. velocity.

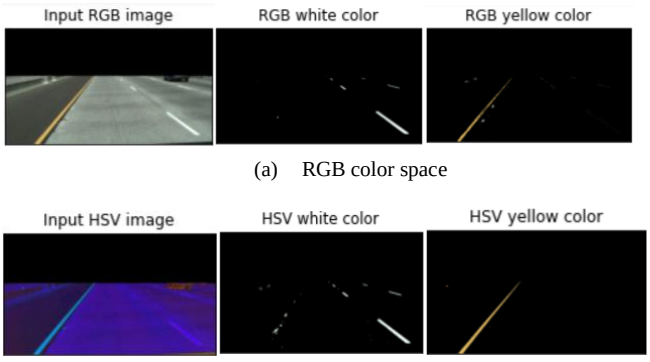
Evaluation for this task is the same as evaluating the distance; by calculating the mean square error and the average difference between the predicted value and the reference value in the annotation file, the accuracy of estimated velocity can be checked.

C. *Lanes detection*

a) Dataset: There are two datasets for this task, named train_set and test_set. For train_set, it includes 3626 video clips with 20 frames in each clip. It also contains 3626 annotations with ground truth data which are width and height data of each lane for the last frame of each clip. For the test_set, it only contains 2782 video clips. Since the test dataset does not include the ground truth lanes data, it is not easy to do quantitative evaluation for the results. Therefore, this experiment is conducted only on the train_set. After shuffling the clips data in the train_set, the first 200 clips will be used as the final test set.

b) Preprocessing: In the preprocessing part, after getting the region of interest images, the images should be processed to reduce image noises. This experiment tried three functions, which are Gaussian Blur, Mean Blur and Median Blur. According to the experimental results, the Gaussian Blur has the best performance of removing obvious noises.

In order to extract suitable color features, many experiments are also carried out. The lines that should be detected are all yellow or white color, so only the white and yellow color should be selected from the image. Using color features can easily extract lane features except vehicles, road and other useless features. The experiments show that the single-color space cannot extract two colors well at the same time, so combining multiple color spaces will be the solution for this weakness. From Fig. 8, it is obvious that the different performance on RGB and HSV color space when extracting different colors. Comparing Fig. 15(a) and Fig. 15(b), the RGB performs better than HSV on extracting white color, RGB white color image has less noisy points. And the HSV color space performs better on extracting yellow color. Therefore, this experiment decides to use RGB space to select white color and HSV to select yellow color and combine them. Moreover, after many attempts, the thresholds for extracting two colors are shown in the TABLE I.



(a) RGB color space

(b) HSV color space

Fig.15. Comparison of RGB and HSV color space

TABLE I. THRESHOLDS FOR EXTRACTING COLOR

Color Space	Lower threshold	Upper threshold
RGB	[160, 160, 160]	[255, 255, 255]
HSV	[10, 10, 170]	[34, 255, 255]

c) Detection: For the edge detection, firstly, the Sobel operator is used to select edge parts. This experiment also calculates the gradients for both the X and Y directions. And the Sobel operator is applied in both the horizontal and vertical directions. Fig. 16 is the output images. According to the outputs, the Sobel operator can detect edges but still have lots of noise, and these image noise create false edges. Some edges are also marked several times.

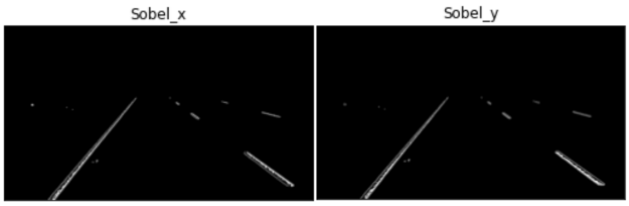


Fig.16. Sobel Operator

Therefore, the Canny edge function is tried to improve the above results. Canny edge function could ensure that a given edge in the image should only be marked once, and most of the false edges are removed as Fig. 10 shown.

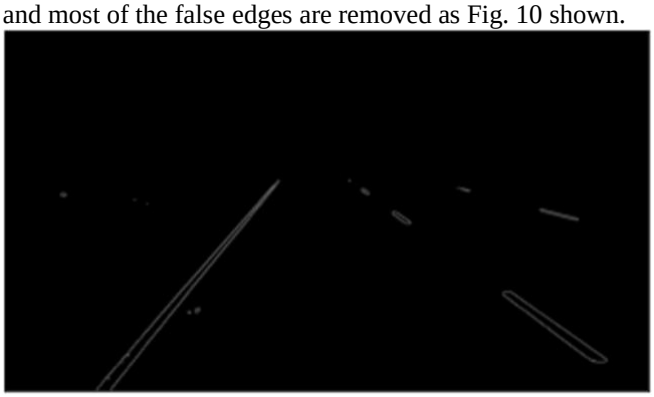


Fig.17. Canny Edge Detector

a) Optimization

After conducting the Hough Transform, there are many duplicated and overlapped lines on each lane line. The main target of the optimization step is to transform multiple lines to a single line for each lane. Therefore, firstly, the lines which have similar slopes could be classified into the same group. Most of the lanes can be divided into two groups, which are left lane and right lane. If the slope of a line is more than zero, this line belongs to the right line group, and if the slope of a line is less than zero, this line belongs to the left line group. Furthermore, there are some images has more than two lane line, so the group divided by slope will be more detailed, such as if there are four lanes in the image, the slope of first lane could be in range of (0, 0.5), the slope of first lane could be in range of (0.5, 1), the third slope could be (-0.5, 0), and the last one could be (-1, -0.5). The following step is to calculate the average slope and intercept values of each group and get the final single lines for each lane.

b) Metrics

In this task – Lane Detection, we take accuracy, false positive rate(FP) and false negative rate(FN) as three metrics. The metric accuracy measures the difference between the ground-truth and our prediction, the equation is in (13):

$$\text{Accuracy} = \frac{\sum_{clip} C_{clip}}{\sum_{clip} S_{clip}} \quad (13)$$

where the clip is the sum of clips, the clip is the number of correct points in the 20nd frame of the clip, the clip is the number of requested points in the 20nd frame of the clip.

The metrics false positive rate (FP) measures the ratio of wrong detected lanes. This means the lane is predicted but not matched with any lane in ground-truth. The equation is in (14):

$$\text{FP} = \frac{F_{pred}}{N_{pred}} \quad (14)$$

where the Fpred means the number of wrong predicted lanes, Npred means the number of all predicted lanes.

The metrics false negative (FN) measures the ratio of missed detected lanes. This means the lane is in the ground-truth but not matched with any lane in the prediction. The equation is in (15):

$$\text{FN} = \frac{M_{pred}}{N_{gt}} \quad (15)$$

Where Mpred is the number of missed ground-truth lanes in the predictions, Ngt is the number of all ground-truth lanes.

IV. Results and Discussion

A. *Result of distanc and velocity estimation*

The result of estimation distance, there are three cars belonging to different images were counted in Fig. 18, and the information in the prediction results and annotations were compared.

```
mse_x:11.61
average_diff_x:1.97m
mse_y:0.22
average_diff_y:0.27m
mse__velo_x:3.02
average_diff_velo_x:1.0m
mse_velo_y:0.03
average_diff_velo_y:0.1m
```

Fig.18. Three counted vehicles

The result is that the average error of the three vehicles on the X-axis is only 1.97 meters and that on the Y-axis is only 0.26 meters. By comparing the position information with the annotation, the error for X is no more than 5 meters, and the error for Y-axis a no more than 2 meters. Fig. 19 is one of the results for vehicles position and velocity.

Fig.19. One of the final results

B. *Result of Lane detection*

There are 200 clips as the test dataset. After applying the algorithm to these test data, some visual results in Fig. 20 can demonstrate the performance of this algorithm.

Fig.20. Three examples of good performance

Fig.21. Three examples of bad performance

In order to accurately evaluate the performance and accuracy of detected lane lines, three quantitative evaluation metrics which mentioned above are used, and results shown in the TABLE II.

TABLE II. RESULTS OF EVALUATION METRICS

Metrics	**Accuracy**	**FP**	**FN**
Value	0.72	0.35	0.56

C. Discussion

a) Discussion of distance and velocity estimation: Distance and velocity estimation are highly dependent on vehicle detection, thus yolov5, a powerful deep learning model, was used to improve detection precision. Car height parameter could affect the estimation accuracy, by comparing and calculating, a suitable car height was chosen, which has a good performance.

In some cases, the number of detected vehicles of frame1 and frame40 is unequal. Vehicles may disappear or occur during the 2 seconds. In Fig.22, when the detected numbers do not match, we will choose another frame and time duration to calculate the velocity.

Fig.22 The same car in different frame

b) Discussion of distance and velocity estimation: According to Fig. 21, most of the results perform well. In these good examples, all of the number and position of driving lane lines from the input image can be detected accurately. However, this bad example shows that it has both wrong predictions and missed predictions. The reason may be the complexity of the environment, for example, there is a deep shade of the huge truck which may affect detection. And the green belt in the left of the image, which has a linear character and is next to the driving lane. When images are well lit, this yellow soil can easily be mistaken for yellow lines. From the evaluation results in TABLE Ⅱ, the average accuracy is about 72 percent, which is quite high. And the best accuracy is about 97 percent. These values can present the good performance of our algorithms. The False Positive and Negative rate is 35 percent and 56 percent, which means our prediction still missed many lanes.

There are some weaknesses of our implementation, firstly, the performance of the result relies on the image environment, for example, the shade, absence of lines, and weather will all affect the detection results. Secondly, the performance of detecting curve lines is not good enough. The implementation cannot calculate the slope of the curve driving lanes. To improve the results, some further methods are considered. Deep learning methods could be tried in further experiments. Moreover, trying other functions to improve preprocessing which may be helpful to reduce the influence of the environment

V. CONCLUSION

This report records the process and method of vehicle distance, vehicle velocity estimation and lane line recognition. In general, the approaches work well. This report proposes a relatively simple method that is able to directly track the speed of the vehicle and identify most of the lane lines from the trajectory of the video sequence. The method has real-time capability, and the accuracy is within a reasonable range. Future work should focus on the accuracy of the model to deal with more complex environments.

A. Summaries of implementation

This report records the process and method of vehicle distance, vehicle velocity estimation and lane line recognition. In general, the approach of distance and velocity works well, and a well-performed image lane-detection algorithm has been presented. This report proposes a relatively simple method that is able to directly track the speed of the vehicle and identify most of the lane lines from the trajectory of the video sequence. Series of pre-processing methods improve the detection results, and the detection methods used in this paper worked quite great in this project. The method has real-time capability, and the accuracy is within a reasonable range. While in some complex environments such as partial deep shade on the road, disturbances of other linear feature objects, this algorithm has some limitations.

B. Further Work

For further improvements, deep learning methods could be tried to improve the implementation, other pre-processing methods could be applied to reduce the interference of complex environments. The combination of other detectors may also contribute to the performance improvements.

VI. CONTRIBUTION OF GROUP MEMBERS

- Haonan Zhang: Programming and evaluation (Task 3 – Lane Detection), Report (Task 3 - Method, Results and Discussion, Conclusion), Demo Presentation (Task 3).
- Hanrui Tao: Programming and evaluation (Task 3 - Lane Detection), Report (Task 3 - Experiment Setup and Results and Discussion), Demo Presentation (Task 3).
- Fei Xu: yolov5 model parameter setting, XYWH coordinates conversion, improvements and discussion for velocity estimation, demo preparation.
- Yiyang Ke: Position and velocity estimations, final program integration and evaluation for task1 and task 2, improvement and discussion for position estimation.
- Dannie Zhao in charge of the introduction and conclusion part of this report.

REFERENCES

[1] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu.Automatic traffic surveillance system for vehicletracking and classification.IEEE Transactions onIntelligent Transportation Systems, 7(2):175–187,2006.

[2] B. Coifman, D. Beymer, P. McLauchlan, and J. Ma-lik. A real-time computer vision system for vehi-cle tracking and traffic surveillance.TransportationResearch Part C: Emerging Technologies, 6(4):271–288, 1998.

[3] S. Hasirlioglu, A. Riener, W. Ruber, and P. Winters- berger. Effects of exhaust gases on laser scanner data quality at low ambient temperatures. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 1708– 1713. IEEE, 2017.

[4] R. Rasshofer, M. Spies, and H. Spies. Influences of weather phenomena on automotive laser radar sys- tems. Advances in RadioScience, 9(B. 2):49–60, 2011.

[5] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe.Unsupervised learning of depth and ego-motionfrom video. InProc. CVPR, 2017.

[6] J. Dai, K. He, and J. Sun. Instance-aware semanticsegmentation via multi-task network cascades. InProc. CVPR, pages 3150–3158, 2016

[7] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convo-lutional instance-aware semantic segmentation. InProc. CVPR, 2016.

[8] Geiger, P. Lenz, and R. Urtasun. Are we readyfor autonomous driving? the kitti vision benchmarksuite. InProc. CVPR, 2012.

[9] P.F.Felzenszwalb,R.B.Girshick,D.McAllester,andD.Ra- manan. Object detection with discriminatively trained part based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich fea- ture hierarchies for accurate object detection and semantic segmentation. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 580–587. IEEE, 2014.

[11] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

[12] Xiaoming, Liu, Qin Tian, Chen Wanchun, and Yin Xingliang. "Real-time distance measurement using a modified camera." In 2010 IEEE Sensors Applications Symposium (SAS), pp. 54-58. IEEE, 2010.

[13] Simek, Kyle. "Dissecting the camera matrix, part 3: The intrinsic matrix." URL http://ksimek. github. io/2013/08/13/intrinsic (2013).

[14] Hummel, Robert A., B. Kimia, and Steven W. Zucker. "Deblurring gaussian blur." Computer Vision, Graphics, and Image Processing 38, no. 1 (1987): 66-80.

[15] J. Canny, “A computational approach to edge detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.8,no.6 pp.679–698, 1986.

[16] William McIlhagga, “The Canny Edge Detector Revisited, ” International Journal of Computer Vision, no.91,pp.251–261,2011.

[17] P.V.C. Hough, A Method and Means for Recognizing Complex Patterns, U.S. Patent 3,069,654, December 18, 1962.

[18] Illingworth, John, and Josef Kittler. "A survey of the Hough transform." Computer vision, graphics, and image processing 44, no. 1 (1988): 87-116.

[19] T. liu, C. Li, K. Zhou and M. Wang, "TuSimple/tusimple-benchmark", GitHub, 2021. [Online]. Available: https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/velocity_estimation. [Accessed: 21- Apr- 2021].

[20] Lan, Wenbo, Jianwu Dang, Yangping Wang, and Song Wang. "Pedestrian detection based on YOLO network model." In *2018 IEEE international conference on mechatronics and automation (ICMA)*, pp. 1547-1551. IEEE, 2018.

[21] S. Pokhrel, "Image Data Labelling and Annotation—Everything you need to know", Medium, 2021. [Online]. Available: https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1. [Accessed: 22- Apr- 2021].

[22] P. Pavithran, "How to label custom images for YOLO - YOLO 3 | CloudxLab Blog", CloudxLab Blog, 2021. [Online]. Available: https://cloudxlab.com/blog/label-custom-images-for-yolo/. [Accessed: 22- Apr- 2021].

[23] Mark S. Nixon and Alberto S. Aguado. Feature Extraction and Image Processing. Academic Press, 2008, p. 88.

[24] J. Canny, “A Computational Approach to Edge Detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679-698, November 1986.

[25] Kristijan Ma˘cek, Brian Williams, Sascha Kolski,” A lane Detection vision module for driver assistance”, EPFL, 1015Lausanne, 2002, pp. 1-6.

[26] Shapiro, Linda and Stockman, George. "Computer Vision", Prentice-Hall, Inc. 2001.