Monitoring Safe Following Distances for Vehicles from Digital Images Using Deep Learning

Mingchen Mi

A project report submitted to the Auckland University of Technology
in partial fulfillment of the requirements for the degree of
Master of Computer and Information Sciences (MCIS)

2021

School of Engineering, Computer & Mathematical Sciences

Abstract

Automatic or auxiliary driving system is becoming popular in recent years. Safe following distance monitoring will help the driver to prevent traffic accident happens. There are a spate of ways to implement this system, traditional image process, or deep learning methods, we will discuss how deep learning can improve the system performance, in lane detection, following distance estimation, vehicle ego-speed estimation, vehicle tracking and re-identification, and finally, complete this safe following distance monitoring system.

Keywords: Lane Detection, Speed Estimation, Vehicle Re-Identification, Vehicle Distance Estimation, Deep Learning

Table of Contents

Chapter 1 Introduction 1

1.1 Background and Motivation 2

1.2 Research Questions 3

1.3 Contributions 4

1.4 Objectives of This Report 4

1.5 Structure of This Report 4

Chapter 2 Literature Review 6

2.1 Introduction 7

2.2 Lane Detection 7

2.3 Vehicle Following Distance Estimation 9

2.4 Vehicle Speed Estimation 11

2.5 Vehicle Tracking and Re-Identification 15

Chapter 3 Methodology 18

3.1 Dataset 19

3.2 ResNet 23

3.3 Lane Detection 23

3.4 Vehicle Tracking Based on Re-Identification 25

3.5 Image to Optical Flow to Speed 30

3.6 Program Implementation 33

Chapter 4 Results 35

4.1 Safe Following Distance Monitoring System Flow Chart 36

4.2 Performance Analysis 37

4.3 Limitations of the Research 42

Chapter 5 Analysis and Discussions 44

5.1 Analysis 45

5.2 Discussions 45

Chapter 6 Conclusion and Future Work 46

6.1 Conclusion 47

6.2 Future Work 47

References 50

List of Figures

Figure 2.1 Lane points prediction based on cell selecting ……...……....7

Figure 2.2 Horizontal triangulation……...……8

Figure 2.3 Use color to indicate dense optical flow…….……13

Figure 2.4 Histograms of vehicle images…….16

Figure 2.5 Global feature in re-id……..……...17

Figure 3.1 The examples of frames from KITTI……...19

Figure 3.2 GPS and IMU velocity in meter per frame…….…..20

Figure 3.3 Examples of speed frequency from KITTI……..…...……..…..21

Figure 3.4 The optical flow by RAFT, driving and stop……...…….....21

Figure 3.5 Image samples from Veri-776……..……...22

Figure 3.6 The ground truth format from TuSimple dataset……..……..22

Figure 3.7 ResNet residual block……..……...……23

Figure 3.8 Frame mask for Hough Transform(top) and deep learning (bottom)……...24

Figure 3.9 GrabCut foreground mask generator……..…..…27

Figure 3.10 Cross entropy loss of vehicle re-id.....27

Figure 3.11 Vehicle re-id ranking…….……28

Figure 3.12 Image to Optical flow to speed by convolutional neural network…….…31

Figure 3.13 Speed distribution for optical flow data…….…….…….…..…31

Figure 3.14 Speed distribution for train and test set…….……..……..……..…32

Figure 3.15 Huber loss of vehicle Speed estimation……...……..…35

Figure 4.1 Flowchart of the System.…….……...…….....37

Figure 4.2 Speed distribution in test set and predictions……..…….38

Figure 4.3 Mean of absolute speed difference on the test set…….……..39

Figure 4.4 Mean of absolute speed difference in percentage on the test set……..40

Figure 4.5 canny edge detection: original image, converted line drawing and mask……..……..40

Figure 4.6 Deep learning lane detection…..…….41

Figure 4.7 Vehicle re-id cosine distance, feature based and pixel based…….……..42

Figure 4.8 Demo of the safe following distance system…..……..43

Figure 4.9 Speed estimation failure cases…….…..……..43

Figure 4.10 Optical flow error by video transcoding…..…….……..44

List of Tables

Table 3.1 Velocity in KTITTI dataset……..……..……19

Table 3.2 Vehicle re-id performance……...……..28

Table 3.3 Vehicle re-id same camera and view…….……29

Table 3.4 Threshold of similarity for re-id…….…….……..……30

Table 3.5 Software dependencies…….34

Chapter 1
Introduction

This chapter is beginning of this report, it contains 5 parts: In the background and motivations, we explained the importance of keep a safe following distance. In related technological background, the research questions are asked, basically it provide what needs to be achieved to make improvement on safe following distance monitoring. Our contributions are detailed, including specified targets that need to be approached. Objective is related to with a brief plan of how to approach the targets. Structure of this report is presented for an overview of this report.

1.1 Background and Motivation

Safe following distance monitoring is essential in road safety for vehicle following and securely overtaking (Feng, 2010), to give the response time for the driver, and left the emergency brake distance for the vehicle, it is the predetermined minimum following distance between the leading and following vehicle (Takagi, 2011). Many countries have suggested the “3-seconds” or “4-seconds” rule in their road code, it can be explained by if the vehicle speed is 10 m/s, the safe following distance will be 30 or 40 meters.

Object seems smaller when it goes far is a common expressions of perspective, in a 2D image, the object distance can be estimated if it’s real size is known. There are many objects in the traffics that we can use as reference, such as the lane width was supposed to be constant in the same section of road.

Speed is relative, we perceive our driving speed by the object movement around us, and the acceleration acting on us, however, as human, most of us are very hard to identify our driving speed by just our eyes. In fact, many traffic accidents were started with incorrect speed estimation made by the driver. GPS is a very good channel to access the vehicle speed, but it could drop offline, or lose accuracy in urban canyon environments (Ge, 2003). Inertial Measurement Unit (IMU) acceleration data can be used for calculate the speed (Li, 2020), but it will produce cumulative error. We require an alternative method instead, to monitor our driving speed in order to compute the safe following distance. Simultaneous Localization and Mapping (SLAM) matches the corresponding image features (such as an unknown landmarks) in sequenced frames, and use these features to estimate the position of camera, and the relative motion between them in frames, but most SLAM technologies rely on stereovision algorithm or equipment (Md. Tanvir, 2018), which is not what we desired.

Optical flow was mainly used at visual odometry (Chen, 2017), for estimating the camera's ego-motion, or works combine with road geometric data to estimate the vehicle or aircraft’s relative position or trajectory.

Vehicle re-identification was more like vehicle brand recognition, most researches are focusing on how to achieve a better performance on the dataset, and ignoring the real scenario. It is technically impassable to compare two vehicle images that has no shared views, and those manually labeled part features are not obvious, or representative enough to distinguish vehicles in the same brand, therefore, vehicle re-id technology is mainly used for ranking to narrow the search scope, or as an alternative when license plate is not applicable.

However basically all vehicle re-id algorithm performs quite well when the images have the exactly same view and the light condition is identical, especially if there is only one view (just front, back or side) visible for the vehicle, with the vehicle's rectangular structural characteristic, we will have minimum background semantic information.

1.2 Research Questions

We need the vehicle speed to estimate the vehicle's position after 3 or 4 seconds, and lane and vehicle detection to confirm that if we are following a vehicle, also, vehicle re-identification to track the front vehicle, to find out that how long have we been following that car. And most importantly, we need to estimate our following distance. As it's a project about deep learning, we will discuss that how deep learning improve the methods in accuracy and universality, therefore, the research questions would be:

(1) What deep learning method can be implement for lane detection?

(2) How will deep learning be used at vehicle ego-speed estimation?

(3) How will deep learning solve the problems in front vehicle detection and tracking.

(4) How to estimate the current following distance?

The core idea of our work is to use different deep learning method to achieve the requirements in computing and monitoring the safe following distance for vehicle, therefore, we need to investigate, evaluate and compare these potential deep learning

methods, to choose the appropriate one in each purpose. When necessary, we also need to collect new data to train the model or to evaluate our system.

1.3 Contributions

The theme of this project is to compute and monitoring the safe follow distance by visual base on deep learning, it's a modularized program that can be divided into 4 parts, (1) lane detection, (2) following distance estimation, (3) vehicle ego-speed estimation, (4) vehicle re-identification and tracking.

1.4 Objectives of This Report

Firstly, this report till introduce the traditional methods to approach our modules: the lane detection, following distance and ego-speed estimation, and vehicle re-identification and tracking, we will demonstrate and discuss the behavior and limitation of these methods, and evaluate the advantages of deep learning based method on these research areas.

Then, a visual based safe distance monitoring system is proposed, by the data outcomes from our modules, and we can divide objectives of this report corresponding to the modules, to compute the optical flow field from image frames, and use CNN to estimate the real-world speed from it, use structural features to detect lanes from the image, and geometric perspective to convert the distance in pixel level measurement to real-world measurement. At last, detect the front vehicle and use its classification features to track it by re-identifying this vehicle. Algorithms were approached by Python.
And finally, we discussed visual scenarios that may collapse our system.

1.5 Structure of This Report

The structure of this report is described as follows:

- In Chapter 2, we will provide a literature review on lane detection, following distance estimation, vehicle ego-speed estimation, vehicle re-identification and tracking, we will separately discuss them in traditional methods and in deep

learning methods.

- In Chapter 3, we will implement the deep learning research methods and algorithm we choose. We will present our dataset, and how these methods satisfy the design purpose of each module of the system.
- In Chapter 4, we will demonstrate and evaluate the result of each system module and the entire system, and compare to the traditional methods. We will also state the limitations of these methods.
- In Chapter 5, we will analysis and discuss the experimental results.
- At last, in Chapter 6, we make a conclusion to summarize our research outcomes and planning for the feature works.

Chapter 2
Literature Review

In this chapter, we analysis the research questions and find feasible solutions in traditional and deep learning, present cross and inner comparison of them. The focus of this paper would be the four modules that construct vehicle safe following distance monitoring system.

2.1 Introduction

Monitoring vehicle safe following distance can prevent us from the rear-end accidents, there are many direct methods for distance detection, but they all has some weakness in driving scenarios, such as ultrasonic ranging method has a 4 to 5 meters of detection range, because ultrasonic speed can be easily influenced. Millimeter wave radar works well at long range detection, but it's too expensive and over complex as a vehicle device. And as kind of visible light, laser is just not suitable with traffic ranging.

Visual distance detection is based on computer vision, the challenges in this method are complex weather condition, illumination and road environment, and deep learning can adopt those difficulties.

2.2 Lane Detection

Traditionally, lane detection is about image processing, a pixel level segmentation, in purpose of highlighting these white and yellow lines that exist on the image. especially when there are vehicles in the lanes. another method is to assume the lanes by the camera position and steering wheel angle, which was mainly used for vehicle trail prediction in parking assist systems. But occlusions, road maintenance, lighting and other environmental condition would interference the detection.

We can convert the frame image into line drawing of objects' boundaries by Canny edge detection (Canny, 1986). First we convert the 3-chennel RGB image into grayscale image, the gray value of each pixel is between [0,255], this process can simply be done by OpenCV. Then we apply Gaussian blur on this grayscale image to filtering out the unwanted noises. After preprocessing, the Canny edge detector works as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2.1)$$

It starts with computing the horizontal G_x and vertical G_y gradient of each pixel, and respectively convolves with 3×3 gradient template. Then we can compute the magnitude and direction by the following equation:

$$M[u,\mathrm{v}] = \sqrt{P[u,\mathrm{v}]^2 + Q[u,\mathrm{v}]^2}$$
$$\theta[u,v] = \tan^{-1}\left(Q[u,v], P[u,v]\right) \tag{2.2}$$

The candidate edge points were trimmed by the non-maximum suppression method and we can use double thresholding method to reduces the number of false edges. The non-maximum suppression method remains only one-pixel width for every edge, which has the highest gradient in the gradient direction. It makes the edges sharper.

The double thresholding method sets two thresholds, maximum and minimum. Those greater than maximum threshold are directly detected as edges, while those lower than min are detected as not edges, and for these pixels in the middle range, if it is adjacent to an edge, we define it as an edge, otherwise it's not.

After Canny edge detection, to isolate the lanes from all objects' edges in the image, one common solution is using mask to segment the area of lanes, it could be defined by a 3 vertexes triangular mask, or a 4 vertexes trapezoidal mask. At last, use Hough Transform algorithm to detect straight lines and link them as the final lane detection result (Deng, 2018). One method is use the lane vanishing point to decide the mask for the segmentation of region of interest by fitting (Hongru, 2021), and deep learning (Lee, 2017).

There are two kinds of deep learning methods for lane detection, one treats it as a semantic segmentation or instance segmentation task such as LaneNet (Davy, 2018) and SCNN (Tang, 2018), the other uses visual features to predict sequential lane points. In "Ultra-Fast Structure-aware Deep Lane Detection" (Zequn, 2020), they have proposed a method that not based on dense prediction, but used row anchors and cells to convert it

into an image classification problem, which has greatly improved the time complexity in lane detection. This method achieved 96.06% accuracy on the Tusimple dataset. The main idea of this method is predefine the image into segmented rows, and selecting cells that classified as lane:

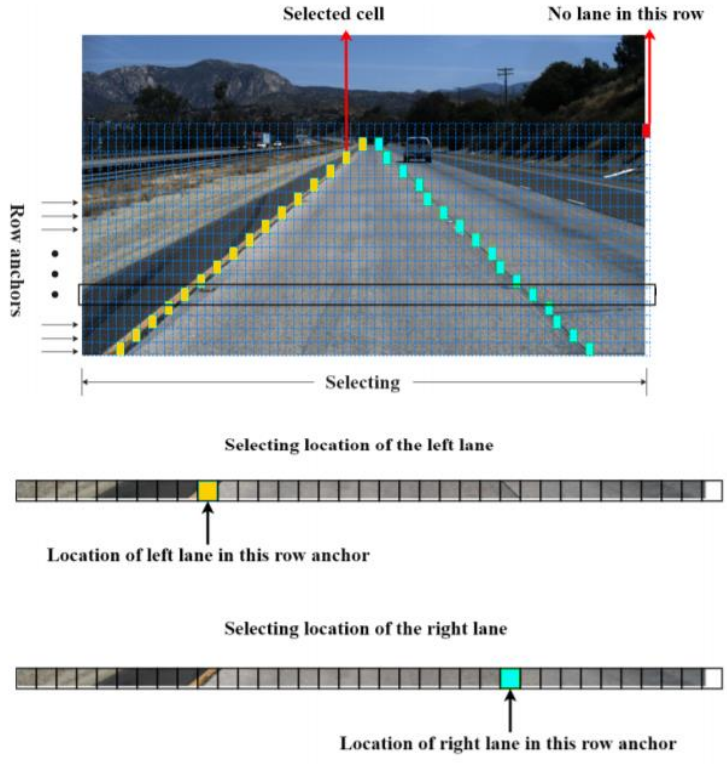


Figure 2.1 Lane points prediction based on cell selecting

It computes the probability distribution in each row, and use it to select the lane cell. The time complexity for pixel level segmentation is the image width×height, and when using row anchors, it becomes row number×image height, which is much faster.

2.3 Vehicle Following Distance Estimation

Perspective is the most popular method in distance estimation by monocular vision camera, on visual, the object size changes in inverse proportion to the distance, we can measure the distance if we know the object size and the camera focal length.

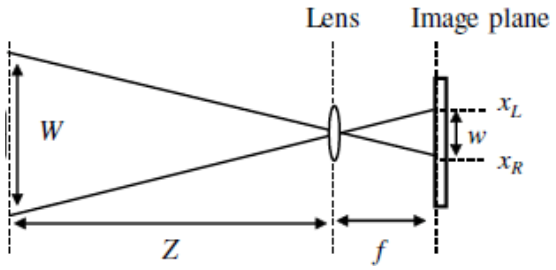


Figure 2.2 Horizontal triangulation

If Z is the distance, W is the real object width under the target vehicle, w is the pixel object width in the image, and f is the camera focal length, the following distance Z can be estimated by:

$$Z = \frac{fW}{w} \tag{2.3}$$

This triangular geometry theory does not consider the observation angle so it can not apply on deviate vehicles, and the error rate depends on the uniformity of the reference object width (Katsuyuki Nakamura, 2013).

License plate has the unified size, so it could be used as the standard measurement, however, the safe following distance can be over 60 meters, even the vehicle will seem so small in our visual in this far.

The width of vehicles is similar; 1.83 meters could describe 80% vehicles' width by hot-selling cars statistics in 2015 (Bao, 2016). In this method, to avoid using camera attributes, they have also created fitted curve function to explain the relationship between the vehicle width in pixel, and the distance in meters. Artificial neural network has been used for long distance detection (Karthika, 2020), which's more accurate with around 1% to 3% relative error. 3D detection can be used at distance estimation of vehicles in other lanes (Zhe, 2020). But vehicle width is not actually regulated, and small cars should not be just ignored.

Lanes have the unified width (Kang, 2017), we use it as the object of reference.

2.4 Vehicle Speed Estimation

Traditionally our vehicle uses photoelectric encoders to detect the spin of wheels to provide us a relatively accurate speed on the speedometer (Malvezzi, 2001), but when the vehicle is sliding by insufficient friction or the wheels' diameter changed, this method is no longer reliable, and many vehicles do not supply the port to access the speed reading. Usually a drive recorder or dash-cam receives the vehicle speed information from the GPS system, but GPS signal is not always reliable as it could be blocked or interfered. Therefore, to catch the relative motion between the camera and the scene, optical flow the pixel level motion between video frames can be used as an alternative method. If we know the length of a lane marking, we can simply calculate the speed by the time interval for them vanishing at the bottom of the frame (Han, 2016).

Optical flow can represent the relative motion between the camera and the scene between two frames, it is how our brain sense our ego-motion (Smith, 2008), which was constructed by the motion speed and direction of pixels from one frame to another. Optical flow can be used at moving object segmentation (Kun, 2016), Image registration (Alexandru, 2015), and human body action recognition (Jue, 2017).

Most optical flow algorithms assume that the brightness of images stay constant, images are sequenced with time continuity or the objects from adjacent images has only moved slightly.

We can estimate the optical flow between two frames from the projection of object motion in three-dimensional space on two-dimensional imaging plane, each pixel outcomes with a 2D vector, if the time interval between frames is very short, we consider this vector is the pixel instantaneous velocity in that frame, which's also called "optical flow vector", all these vectors constitute the optical flow field of the image, which describes the motion field in the real world for motion analysis. If we define $I(x, y, t)$ as the pixel intensity,

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.4}$$

where t is the time dimension, x, y are the 2D coordinate of this pixel in the frame, and

d is the deviation. We can apply Taylor expansion for the left side of equation,

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \varepsilon \tag{2.5}$$

where ε is an infinitesimal of second order, so we can just ignore it. Then we express the previous equation with it,

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \tag{2.6}$$

At last, divide this formula by dt to get the optical flow equation,

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{2.7}$$

where u and v are the horizontal and vertical pixel velocities respectively as they were calculated by distance divided by time, $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the horizontal, vertical and time gradients in image respectively. We get this question from eq. (3.9):

$$I_x u + I_y v + I_t = 0 \tag{2.8}$$

The variables *u* and *v* are still remains unknown; the equation cannot be solved with two unknown variables, therefore, we need constraint conditions, and there are four different methods to calculate the optical flow field:

(1) **Gradient based**. It uses the spatiotemporal differences in image gray scale to calculate the velocity vector of pixels.

(2) **Matching based**. The matching could be regional or on features, feature matching tracks the features in the image, which's robust to large scale motion and brightness changes of the target. And regional matching first locates the similar regions from two images, and calculate the motion vector by region movement.

(3) **Phase based.** Extract the phase information from images to calculate the motion vectors.

(4) Deep learning based. Treat the optical flow prediction problem as a supervised deep learning problem.

The estimated optical flow field could have different motion vector density according to the method. Sparse optical flow methods are represented by the Lucas Kanade method (Bruce, 1981), it's a gradient based method, which finds the extreme points in the image and retract those points from the next frame, to get the motion vector.

Dense optical flow computes movement on each pixel point from the current image to each pixel point in the image next, therefore it's time complexity is higher but also more accurate. We can use color to indicate the direction of optical flow, and brightness to indicate the speed:

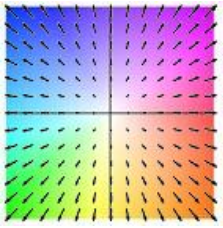


Figure 2.3 Use color to indicate dense optical flow

The evaluation of optical flow model is defined by average End-Point Error (EPE), the EPE is the average Euclidean distance on each vector in a frame. It's impossible to label the optical flow ground truth manually, usually they are point cloud data collected by laser scanner.

FlowNet (Alexey, 2015) algorithm has proposed to learn the concept of optical flow by CNN model, they have created the Flying Chairs artificial dataset, and proved that we can use an artificial dataset to train the model and estimate the optical flow in real world images, however this method was not very competitive to the existing highly fine-tuned methods. And FlowNet2 (Ilg, 2017) was improved from FlowNetSimple and FlowNetCorr with a better performance on large scale pixel displacements, and got an average EPE of 3.96 on the Sintel-clean dataset (Butler, 2012). Recurrent All-Pairs Field Transforms (RAFT), the average EPE is 1.609 on the Sintel-clean dataset. There are three parts in this network:

(1) **Feature Encoder**. Take the pixel shift from two images as feature.

(2) **Context Encode**. Take the contextual feature from one image, to remain the same position for the optical flow predicted in the image

(3) **Update**. Use the output of the two encoders, and compute the similarity in feature level by dot product similarity:

$$\text{similarity } = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (2.9)$$

BRAFT algorithm (Jia, 2021) was based on RAFT, but segments the image into blocks, and use the strength-weakness correlation in pixel pairs to calculate the blokes' visual similarity to clarify the edge of object in optical flow, which is powerful for moving object segmentation, but not very suitable for ego-motion optical flow.

However, in order to get the real vehicle speed from frames, we still require the depth of these vectors in the optical flow, which's impassable for civil vehicles. In visual odometry, monocular visual odometry can only detect 2-dimension pixel movement, which can directly use for estimate the ground speed of unmanned aerial vehicle with a known flight altitude (Chunhui, 2014), or the distance to the roof for indoor robot (Lin, 2019), and we can use features from the image to minimize the re-projection error for estimating the speed. And stereo visual odometry can detect the depth, but it requires two cameras and the depth detection only reliable in a limited visual range.

It has been proved that we can CNN model to predict the camera's ego-speed from optical flow alone (Alexey, 2015) (Memisevic, 2015), this method won't be extremely precise because the predictions were highly dependents on the theoretical universal depth distribution, and we can only make assumptions of the pixel's depth base on its shape, vector distribution, direction and intensity. It is impossible to find a dataset with all traffic conditions that we could encounter, but we could do some data augmentation in the existing dataset.

2.5 Vehicle Tracking and Re-Identification

Vehicle detection is a problem of classification and localization objects in the traffic, we require not only the existence of the car, but also the location of the car in the image, before deep learning, we use the image visual features (Ma, 2017), for example, use the Histogram of Oriented Gradient of the object as feature and a use classifier such as support vector machine to classify the image.

YOLOv5 (Glenn, 2020) is a deep learning algorithm that has the excellent performance in object detection, it is a lightweight algorithm and fast in object detection, which can help us to segment all vehicles as candidates from the image.

There are several ways to approach vehicle tracking:

(1) **Region Based tracking**. Use the location, size or the intensity histogram of the rectangular detection bounding boxes between consecutive frames, it's highly depends on the vehicle detection performance.

(2) **Model Based Tracking**. Create geometric model of vehicle by prior knowledge, vehicle motion, this method is accurate but also costly.

(3) **Feature Based Tracking**. Perform matching in consecutive frames using vehicle features, this method performs well in occluded conditions.

Kalman filter has been used for moving object tracking (Zeng, 2009) for decades, it locks on the given signal feature and track it. Image depth can be used for object tracking in 3-D scene (Gutev, 2019), it can be approached by binocular vision camera (Tian, 2011), and would be more accurate to works combine with range radar (Jie, 2021). But it is costly and not suitable with long range target.

The most identical feature of a vehicle is the license plate (Watcharapinchai, 2017) (Jung, 2018), and the annual inspection signs can identify the vehicle in similar outfits to a certain extent (Liu X. a., 2018), but our system requires to track the vehicle from a relatively far distance, so the outfit of the vehicle could be the only parameter we can get in most cases, the similarity of image can be compared by the pixel level difference, and

deep learning has the advantage to solve the change in background and view angle problem.

The image similarity at pixel level can be calculated with the histograms of two images (Mich, 2001).

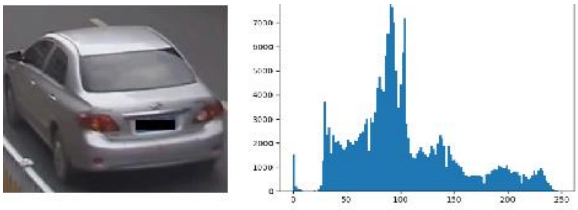


Figure 2.4 Histograms of vehicle images

It can be compared by the image structural similarly, which includes:

(1) **Brightness**. The average grayscale value.

(2) **Contrast ratio**. grayscale standard deviation.

(3) **Structure**. The structure of pixels.

The features could be got from the image, such as the layout of vehicle components, or beyond the image, such as the classification confidences of each class, which's widely used at vehicle re-identification region. Object part feature can be used in re-id for vehicle as well as for person (Sun, 2018), the fine-grained classification uses local features for vehicle, such as the vehicle logo, shape of driving mirror, lights and windows, cargo in the back for trucks, and the color of license plate (He, 2019). Their work was pretty much based on the image resolution, which has higher computation cost. Also, manually labeling vehicle parts has limited its generalization performance.

If the image for each vehicle is too few or imbalanced in the dataset, there will be inter-class problem, the model cannot learn enough feature from the vehicle class, it's a common classification problem. Spatial-temporal information of the vehicle has been used to restrict the matching range by where the target may possibly be. They have used two-stage framework, Siamese-CNN and path-LSTM, and achieved an average Jaccard Similarity (AJS) of 96.39% (Shen, 2017). However, using those extra vehicle information

requires additional computational load, and the time complexity was depending on a chain MRF model.

If we use global feature to conduct re-id, the model would still focus on the primary vehicle components such as the light.

Figure 2.5 Global feature in re-id

Most re-id algorithm would do re-ranking for post process, such as the "*k*-reciprocal Encoding" method (Zhong, 2017), it takes the candidate from the re-id result, compute the Mahalanobis distance for the initial ranking list, Jaccard distance for the *k*-reciprocal ranking list, if the target image appears in *k* rank, it is more likely to be a true match, and afterward, we can add weight to this candidate in re-ranking. spatial-temporal information can also be used at re-ranking (Jiang N. a., 2018). Vehicle post process is very important in vehicle ranking problem, but it takes huge amount of calculation to go over each sample in the category for every search target.

Chapter 3
Methodology

In this chapter, we are going to articulate the research methods for deep learning, and how to implement them to approach our objectives. We will also demonstrate the dataset we have used and the data structure at each stage.

3.1 Dataset

There is no dataset especially designed for speed estimation, but we found the ego-vehicle speed value exists in odometry datasets. The KITTI dataset (A. Geiger, 2012) consists 6 hours of real-world traffic driving recordings that captured in a mid-size city called Karlsruhe, it has included traffic situations of city, highway, residential and campus.

Videos in KITTI was recorded in 10 frames per second, and it gives more than 90,000 frames in total, we took the RGB image sequences, and converted them into optical flow data to train our CNN model, and there are .txt files that contain the vehicle's forward and leftward velocity, in consider of we only require the velocity for safe following, we choose to use the forward velocity as the ground truth label.

Figure 3.1 The examples of frames from KITTI

Image sequences from KITTI are short (from ten seconds to a few minutes) and discontinuous with each other. Velocity values in this dataset was recorded in by GPS (Global Positioning System) and IMU (Inertial Measurement Unit).

Table 3.1 Velocity in KTITTI dataset

vn	velocity towards north (m/s) by GPS
ve	velocity towards east (m/s) by GPS
vf	forward velocity, parallel to earth-surface (m/s) by IMU
vl	leftward velocity, parallel to earth-surface (m/s) by IMU
vu	upward velocity, perpendicular to earth-surface (m/s) by IMU

The GPS velocities were measured by the coordinate displacement (latitude and longitude), and IMU uses the acceleration records to calculate the velocity. Normally in a short scale distance of vehicle movement, IMU velocity is more accurate than GPS, we used the parameters from GPS and IMU to calculate the travel distance between frames at one sample from KITTIl.

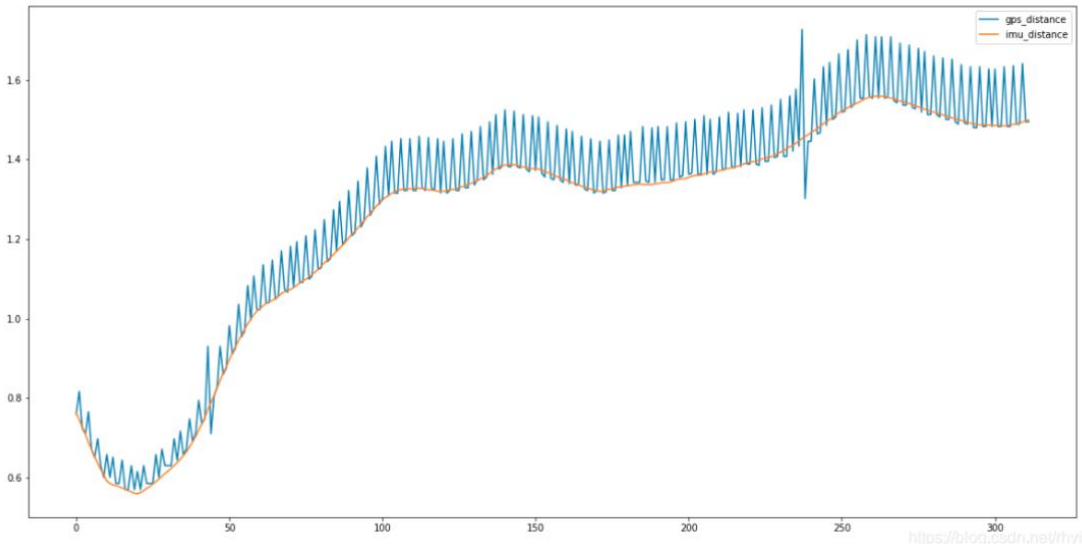


Figure 3.2 GPS and IMU velocity in meter per frame

The distance data that calculated with IMU data by integral method is smoother than the data recorded by GPS. Although there will be accumulative errors for IMU, we require only the accuracy on the velocity between each frames. As a conclusion, we choose to use the IMU forward velocity as the speed ground truth, and the speed frequencies for

some samples are shown below:

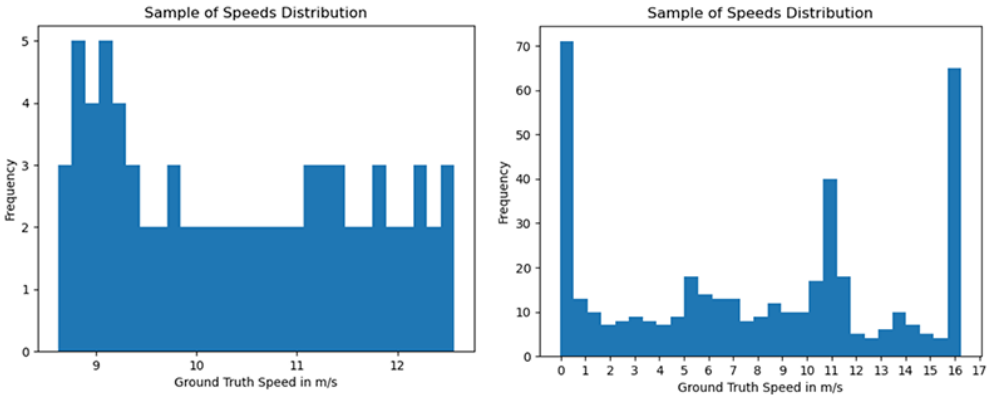


Figure 3.3 Examples of speed frequency from KITTI

One problem in this dataset is the missing speed value for static vehicle, when the vehicle is not moving, the speed reading is a negative number, we have reviewed the dataset, and there is no data for driving backwards, therefore we modified them to 0 m/s.

Figure 3.4 The optical flow by RAFT, driving and stop

Figure above is the optical flow produced by RAFT, we use color to represent the speed vector. For vehicle re-id, we have used the Veri-776 dataset (Liu X. a., 2016), it contains more than 49,357 cropped images of 776 vehicles, there are 37,778 images for training and 11,579 images for testing, each vehicle appears in only one of the training or test set.

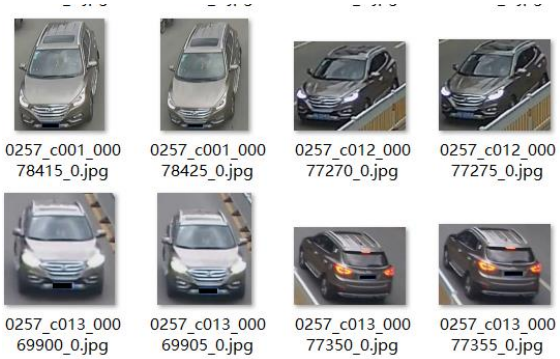
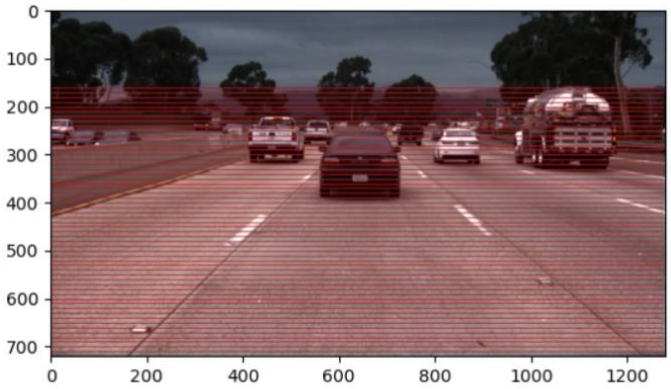


Figure 3.5 Image samples from Veri-776

Vehicles were photographed in environment of urban-area roads by 2 to 18 different cameras, images have different brightness, shooting angle, resolution and occlusion condition. In the version of Veri-776 dataset we have received, license plates were blocked by black rectangles.

In lane detection, we used the TuSimple dataset (TuSimple: Tusimple benchmark, 2021), it has 3626 video clips for train and 2782 video clips for test, each clip includes 20 image frames from 1 second video.

The ground truth was labeled as points in form of invariable y-axis set from 160 to 710 with an interval of 10 that stored as "*h_sample*", and corresponding uncertain x-axis that stored as "lanes" to represent the *x* position of lane. The "-2" value means at this y-axis; lane does not exist.



{"lanes": [[-2, -2, -2, -2, -2, -2, -2, -2, 597, 582, 562, 532, 503, 474, 439, 4
"h_samples": [160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270,
"raw_file": "clips/0531/1492636601694587103/20.jpg"}

Figure 3.6 The ground truth format from TuSimple dataset

Each lane label is actually a set of coordinates in the lane, not the lane region.

3.2 ResNet

Usually, the increase of depth of layer causes the degradation problem on the model, accuracy remains or even decreases, we expect the rest layers can remain all the features when it performs the best in our model, which was called “identity mapping” (Kaiming, Identity Mappings in Deep Residual Networks, 2016), but because of we used the nonlinear activation function Relu (Nair, 2010), we cannot reverse the information loss from the layer outputs. ResNet (Kaiming, Deep Residual Learning for Image Recognition, 2016) used residual learning to solve this problem. It takes the outputs from previous layer (or layers) by a shortcut connection, add to the output from this layer, and this this sum result as the input of the activation function.

$$O = \mathcal{F}(\mathrm{I}, \{Wi\}) + I \tag{3.1}$$

$$\mathcal{F} = W_2\sigma(W_1 I) \tag{3.2}$$

Where O is the output of residual block and I is the input, $\mathcal{F}$ is the residual function, σ is Relu, W_2 and W_1 are the weight from two layers. It’s compulsory to have least two layers in this block.

3.3 Lane Detection

Traditional lane detection is highly depending on the region of interest mask, and deep leaning method is much less relying on it, and mainly use it in purpose increase the structural similarity of the frame with the training set, or reduce detection range above the horizon to increase the detection speed.

Figure 3.7 Frame mask for Hough Transform(top) and deep learning (bottom)

We extract lanes by using the color of pixels (TsungYing, 2006), but all other vehicles, guardrails, street lamps, even the sky and shadows could make the detection fail, because in some cases, it's hard to distinguish the lanes and environment by a grey level threshold, particularly in scene of vehicles. "Ultra-Fast Structure-aware Deep Lane Detection" (Zequn, 2020), we used ResNet-34 as back bone, the image is divided into cell blocks in predefined row anchors, we use ResNet to verify if lane exist in each cell blocks and select them, finally outputs those selected cell coordinates.

One challenge in lane detection is the no-visual-clue problems (Jiang L. a., 2019), to predict the lane when it was occluded. We knew that lanes have the characters of smooth, rigid and continues, missing lane points can be estimated and filled by the location of selected cells, it is defined as the structure loss:

$$L_{str} = L_{sim} + \lambda L_{shp} \tag{3.3}$$

Which is contributed by the loss coefficient λ, similarity loss and shape loss:

$$L_{\text{sim}} = \sum_{i=1}^{C} \sum_{j=1}^{h-1} \left\| P_{i,j,:} - P_{i,j+1,i} \right\|_1 \tag{3.4}$$

For similarity loss, $P_{i,j}$ is the probability on j-th row anchor, and $\| \|_1$ means L1 norm, the sum of absolute difference.

$$\begin{aligned} L_{shp} = \sum_{i=1}^{C} \sum_{j=1}^{h-2} \| \left(Loc_{i,j} - Loc_{i,j+1}\right) \\ -\left(Loc_{i,j+1} - Loc_{i,j+2}\right) \|_1 \\ Prob_{i,j,:} = \mathrm{softmax}\left(P_{i,j,1:w}\right) \\ Loc_{i,j} = \sum_{k=1}^{w} k \cdot Prob_{i,j,k} \end{aligned} \tag{3.5}$$

where $Prob_{i,j,:}$ is the probability at each location, $P_{i,j,1:w}$ is vector in w dimension, $Prob_{i,j,k}$ is the probability of the i-th lane, the j-th row anchor, and the k-th location. The overall loss they proposed in this method was defined as,

$$L_{\text{total}} = L_{cls} + \alpha L_{str} + \beta L_{seg}. \tag{3.6}$$

where α and β are loss coefficients. And beside the structure loss, they have also used L_{cls} classification loss, which based on cross entropy loss with one extra dimension for "no lane" label, and one-hot label as the ground truth for cells in the row, and they have also used the cross entropy as auxiliary segmentation loss L_{seg} for training only.

Images from TuSimple dataset have been resized to 288×800 as the input to the ResNet for classification, after a fulling conduct layer and reshape to probabilities in row anchors, for cell selecting in group classification. Meanwhile during training, multi-scale features output from ResNet layers has been used by auxiliary segmentation to model local features. The performance of lane detection can be evaluated by accuracy in TuSimple dataset

$$accuracy = \frac{\sum_{\text{clip}} C_{\text{clip}}}{\sum_{\text{clip}} S_{\text{clip}}} \tag{3.7}$$

where C_{clip} is the lane points correctly predicted, and S_{clip} means the total amount of ground truth in each clip.

3.4 Vehicle Tracking Based on Re-Identification

Vehicle re-id in deep learning is based on classification, each vehicle will be treat as an

individual class, and we will use the confidence result from each class to generate the feature matrix, and give the similarity outcome by calculate the matrix distance. We have to assume that input images are vehicle images, as there are only class level features in that matrix. We used YOLOv5 to detect and produce bounding boxes to isolate all vehicle targets from the image, then find the closest one in our lane, and extract the features by ResNet-50. The image re-id preprocess is:

- **Resizing**. Resize the image to 200×200.
- **Random Horizontal Flipping**. For preventing model overfitting, randomly rotate the image horizontally.
- **Random Cropping.** To reduce the noise and increase the model's stability for missing value.
- **Normalizing.** To fit the deep learning network, mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225].

In addition, we applied random brightness on the vehicle by segment it from the background, and randomly adjust the brightness by -20% to +20%. The segmentation was done by foreground mask generator GrabCut from OpenCV:

Figure 3.8 GrabCut foreground mask generator

The model training part for re-identification is similar to classification, so we choose to cross entropy loss to evaluate the model.

$$H(p,q) = -\sum_x \left(p(x) log\ q(x) + (1-p(x)) log\ (1-q(x))\right) \tag{3.8}$$

where p is the expected probability distribution output, and q is the actual output, and x is the amount of information. In PyTorch, the cross entropy loss in calculated by combining the softmax, log, and NLLLoss:

$$H(p,q) = -\sum_x (p(x)log\ q(x) \tag{3.9}$$

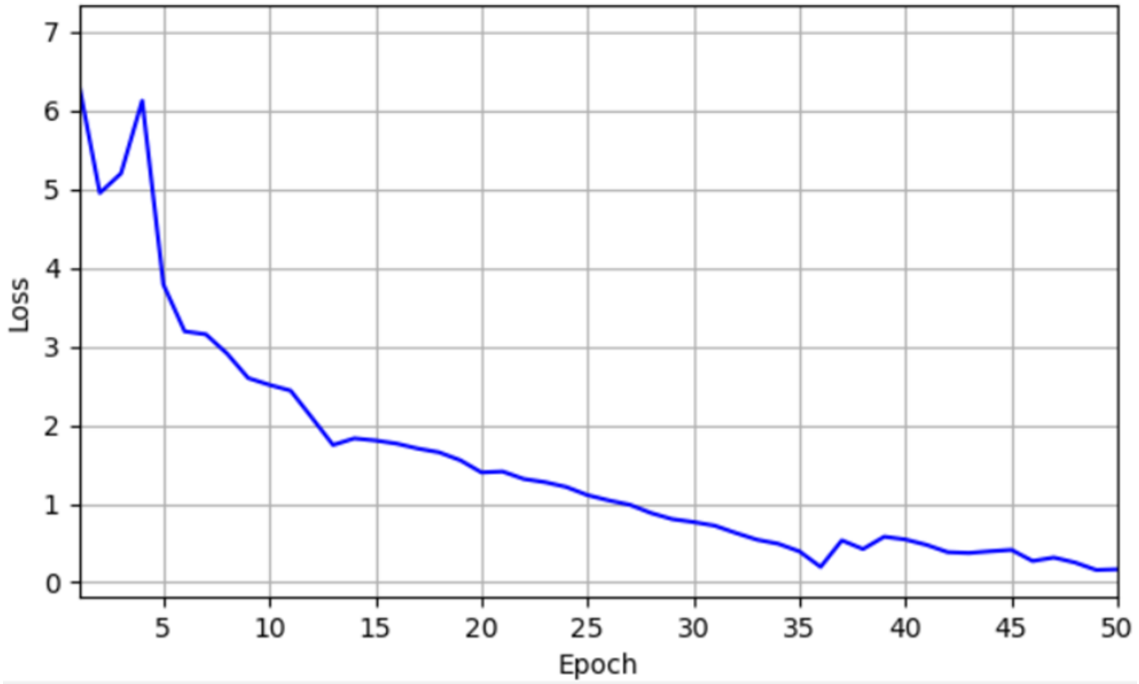


Figure 3.9 Cross entropy loss of vehicle re-id

A re-id model can be to evaluate by mean Average Precision (mAP):

$$mAP = \frac{1}{|Q_R|}\sum_{q\in Q_R} AP(q) \tag{3.10}$$

It's basically the sum of average precisions divide by the number of class, we have 576 classes in the Veri-776 training dataset, each class represent one vehicle. The re-id model can also be evaluated by Rank-N, it means in ranking, the probability in percentage of that the correct target can be found in the top N predictions.

Table 3.2 Vehicle re-id performance

mAP	Rank-1	Rank-5
42.41%	56.25	82.57.

Figure 3.10 Vehicle re-id ranking

The mAP of this model on Veri-776 is 42.41%, however, the evaluation of vehicle ranking requirement is stricter:

(1) **Cross view**. Compare the vehicle's front to its back, or a plan view to aerial view.

(2) **Different camera**. Images from the same camera must be excluded.

(3) **Long scale of time length**. Illumination or weather condition could be very different.

But to track the vehicle in our front, we don't have those restricts, all vehicles are expected to have the same orientation, under the sight of one same camera. The performance for the same camera and view is much better, it against the standard valuation criteria for re-id, but more close to our tracking needs.

Table 3.3 Vehicle re-id same camera and view

mAP	Rank-1	Rank-5
94.3%	98.2	99.7.

The outcome of re-identification is a matrix of confidence values for each class as the feature map, to compare them we can use matrix Euclidean distance:

$$Euclidean\ Distance\ (T,P) = \sqrt{\begin{bmatrix} T_1^2 & T_1^2 & \cdots & T_1^2 \\ T_2^2 & T_2^2 & \cdots & T_2^2 \\ \cdots & \cdots & \ddots & \cdots \\ T_M^2 & T_M^2 & \cdots & T_M^2 \end{bmatrix} + \begin{bmatrix} P_1^2 & P_2^2 & \cdots & P_N^2 \\ P_1^2 & P_2^2 & \cdots & P_N^2 \\ \cdots & \cdots & \ddots & \cdots \\ P_1^2 & P_2^2 & \cdots & P_N^2 \end{bmatrix} - 2 * TP^T} \tag{3.11}$$

Where *T* and *P* are the two matrix.

Euclidean distance works fine with similarity ranking, but we need to normalize the distance to determine if two images were the same vehicle. Cosine distance is another measurement for matrix distance:

$$Cosine\ Distance\ (T,P) = 1 - \frac{\sum_{i,j=1}^{n} t_{ik} p_{ij}}{\sqrt{\sum_{i,j=1}^{n} t_{ij}^2} \sqrt{\sum_{i,j=1}^{n} p_{ij}^2}} \tag{3.12}$$

It outcomes with a number from 0 to 1, which's more apposite for similarity comparison. We took the images from the test set of Veri-776 dataset that shoot from the same camera ID to find the threshold value of similarity to our vehicle tracking module, and found that when the threshold set to 0.97, the accuracy is the highest.

Table 3.4 Threshold of similarity for re-id

Cosine Similarity	Accuracy
1.000	0%

0.990	7.32%
0.980	57.58%
0.975	82.76%
0.970	97.51%
0.965	86.39%

The light reflation on vehicle may produce huge light spot in the image, usually those images will be considered as polluted and exclude from the re-id dataset, so we record the re-id result in the past 1 second, and take the highest on to represent them. In addition, we knew that even if the recall rate is only 1%, on average, error will still occur once every 100 frames, it could be as short as only 5 seconds for vehicle tracking when using a 20 fps camera, in this case, Rank-20 is much more reliable, it's acceptable for vehicle safe following distance monitoring when the update of Boolean value of vehicle following delays for 1 second. The vehicle can be defined by:

if (Cosine Similarity in last 20 frames > 0.97):

return True

else:

return False.

3.5 Image to Optical Flow to Speed

In this section, we used RAFT to convert the image frames into optical flow data, and then use CNN to get the speed value. The model is shown as Figure 3.11.

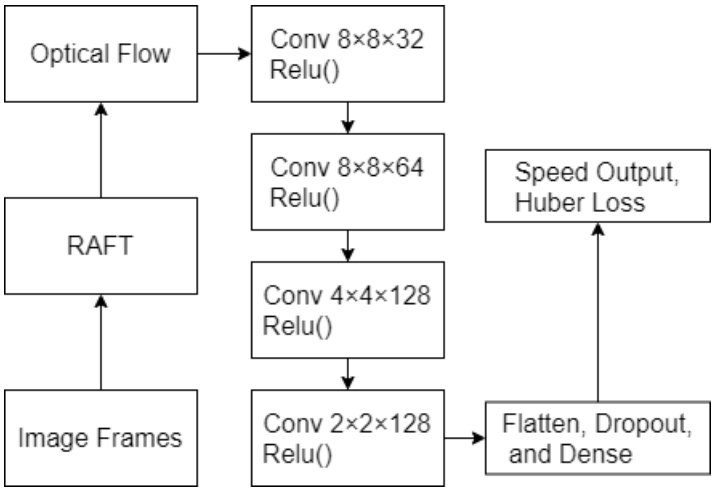


Figure 3.11 Image to Optical flow to speed by convolutional neural network

We chose to use the “City” and “Road” category from the KITTI dataset to train our model, generate optical flow requires pair of image, therefore we removed the last vehicle speed record for every image sequence, finally there are 15123 optical flow data in total.

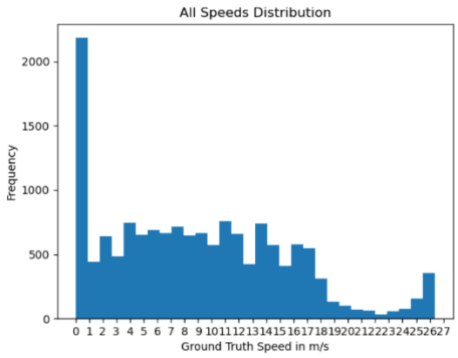


Figure 3.12 Speed distribution for optical flow data

We split the data into train set 80% and test set 20%:

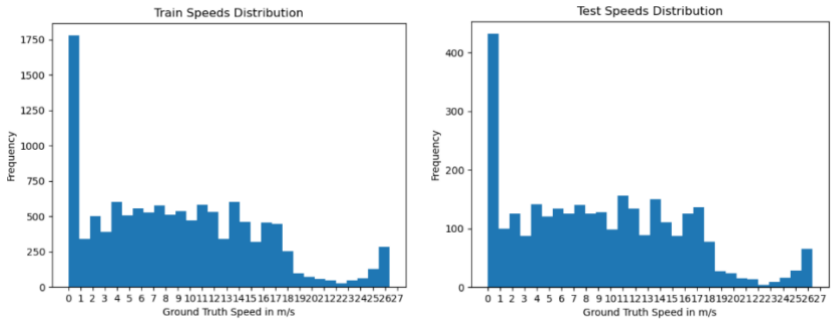


Figure 3.13 Speed distribution for train and test set

In training, the system takes 1242×375 RGB image from the KITTI dataset as input, then we use RAFT to convert the pair of RGB images into optical flow vector fields, in form of vector matrix that has the same width and height, but stored the horizontal vertical speed information in each pixel instead RGB channel value. In principle optical flow vector field is a speed image, but there was no existing standard file format to save them, so we just saved the array by NumPy (Numerical Python). and before the data input to the CNN model, we preprocess the optical flow between two consecutive frames by:

(1) **Horizontal flipping**. Use mirrored image to extend the train set.

(2) **Random rotating**. Randomly rotate the consecutive frames for from -5 to 5 degrees, and add 0 to fill the gap

This is for data augmentation, we did the flipping and rotation on the optical flow matrix during training, when each time they were picked from the training set.

Then the optical flow vectors go into the convolutional layers that connected by RELU activation function to extract features, followed by flatten, dropout and dense to reduce overfitting and prepare for output.

The choice of loss function is determined by the usage and output of the model, unlike classification tasks output with a discrete list of class probabilities, this is a regression task, which outputs only the predicted vehicle speed value, and we need to compare this value with the ground truth speed, the performance is basically how close these two numbers are, and Mean Squared Error (MSE) was commonly used in regression tasks in deep learning, which was defined as,

$$MSE = \frac{1}{m}\sum_{i=1}^{m} \left(y_i - f(x_i)\right)^2 \tag{3.13}$$

where m the number of samples, for sample i, y_i is the predictions and $f(x_i)$ are the ground truth values. However, outliers, such as error value in the ground truth, or extreme circumstance will cause serious influence to the MSE loss. Particularly for this project, the optical flow data was produced by deep learning method RAFT, so it contains some

errors and at some speed range, there were too few images in our dataset, on another hand, the speed ground truth from this dataset was detected by the Inertial Measurement Unit (IMU) installed on the vehicle, it could also produce outliers.

In this case, Mean Absolute Error (MAE) is less sensitive than MSE, as it calculates the absolute value of the distance from predation and the ground truth：

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |y_i - f(x_i)| \tag{3.14}$$

But for gradient solving, MAE is more complex than MSE, which's detrimental for model convergence and learning. Huber Loss propose has been proposed as an equilibrium (Wang, 2020):

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & |y - f(x)| > \delta \end{cases} \tag{3.15}$$

where δ is a hyper-parameter, when $|y - f(x)| \le \delta$, Huber Loss is equal to MSE, when $|y - f(x)| > \delta$, Huber Loss becomes alike to MAE. Commonly we set δ = 0.5 to achieve the balance between those two loss functions, and it could be adjusted or optimized by cross validation in future. The benefit of using Huber loss to train the model are that it's more stable than using MSE, and faster than MAE. At last, we used Adam optimizer (Diederik, 2017) for the optimization. We stopped the training at 20 epochs as the test loss curve became smooth.

3.6 Program Implementation

PyTorch is an open-source, end-to-end deep learning framework in Python, it has sample, concise API ports and the balance between usability and speed, although it may not perform as good as Tensorflow in accuracy, but the training time is shorter and the user experience is also much better, which makes it more suitable for research purpose.

Open Source Computer Vision (OpenCV) is a powerful and cross platform image processing and computer vision library, we used it to read the frames from video or

sequenced images, preprocess them and display the detection outcomes by its image editing module.

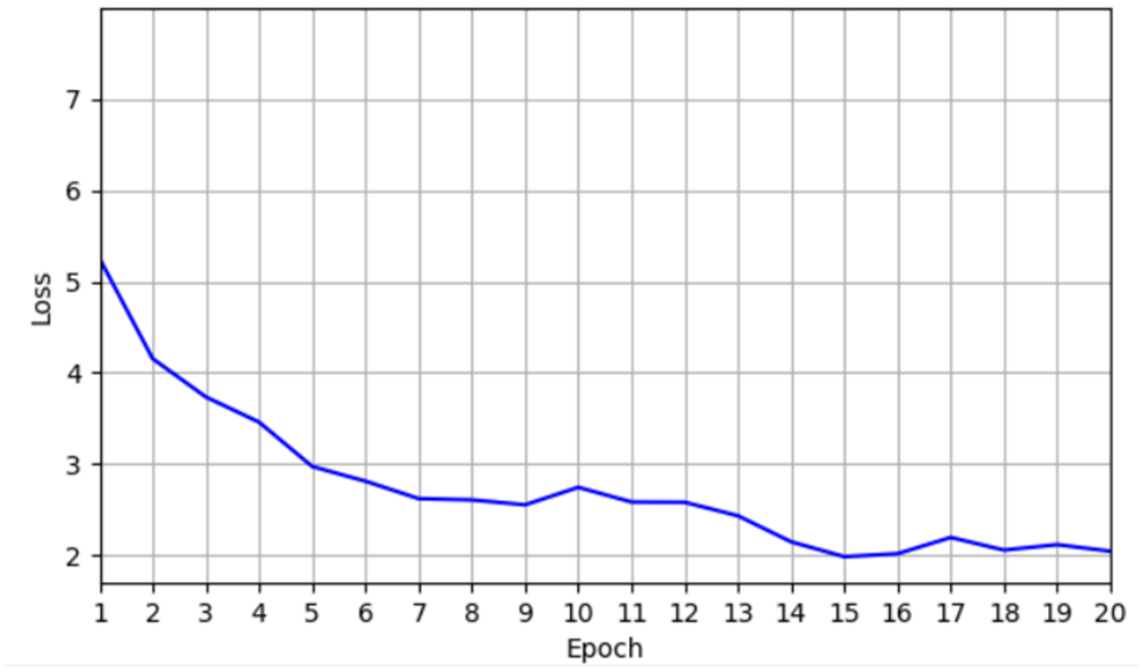


Figure 3.14 Huber loss of vehicle Speed estimation

Table 3.5 Software dependencies

Dependency	**Version**
Python	3.8
OpenCV-Python	4.4.0.44
PyTorch	1.9.0+cu102

Chapter 4
Results

In this chapter we will demonstrate our system and evaluate the experimental result for each module by traditional method and deep learning method. We will also present the limitations or issues in the system.

4.1 Safe Following Distance Monitoring System Flow Chart

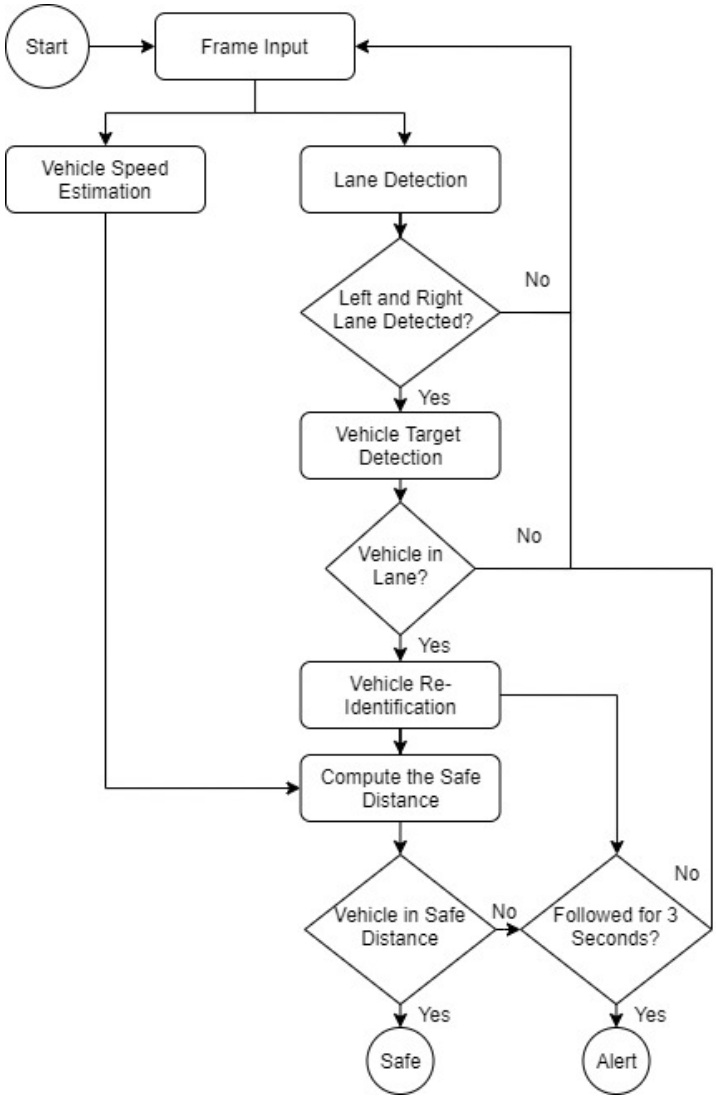


Figure 4.1 Flowchart of the system

When the system takes the first frame, there's nothing we can do, because there's no way to tell the speed from one single image, the system will be initiated since the second frame input, and the speed estimation was done independently, because it uses optical flow field as input data.

The system requires the existence of lanes as well, we will find the left and right lane by their inclined angle, when both lanes detected, we will lock on the vehicle between these two lanes. If there is such a vehicle in front of us, we use the re-identification method to count that how long we have been following it, when it has reached a given threshold

time, such as 3 seconds, the following distance alert system turns on, otherwise it will compute and display that distance in another color.

When the following time is enough, and our distance to the front vehicle is beyond the safe line, the alert will be sent.

4.2 Performance Analysis

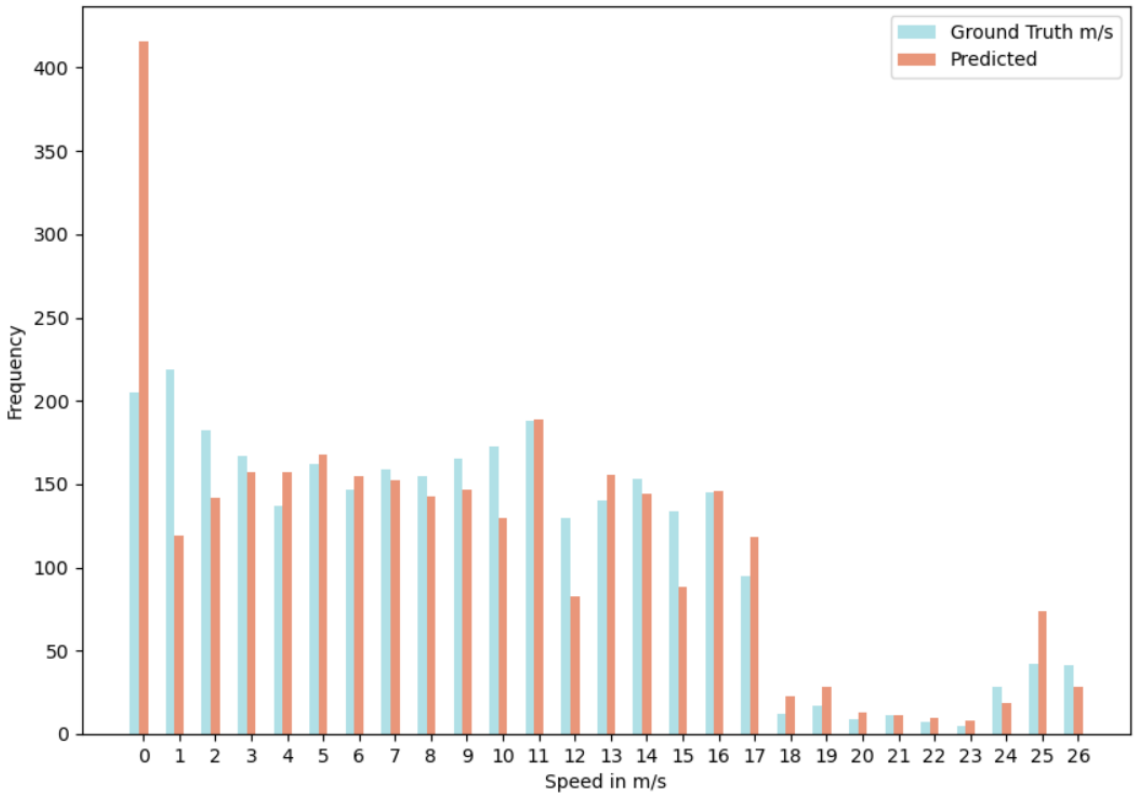


Figure 4.2 Speed distribution in test set and predictions

Obviously this model is not sensitive with very low speed, only less than half predictions for speed between zero and one were correct, it may because of when the vehicle is turning, the speed could be very low, but the visual was not static like when it stops at the intersection. And for high speed driving, it seems there wasn't pretty enough training data in speed above 17 m/s, and those frames were mainly coming from the "road" category, and the environment of highway were trends to be constant and "duplicated".

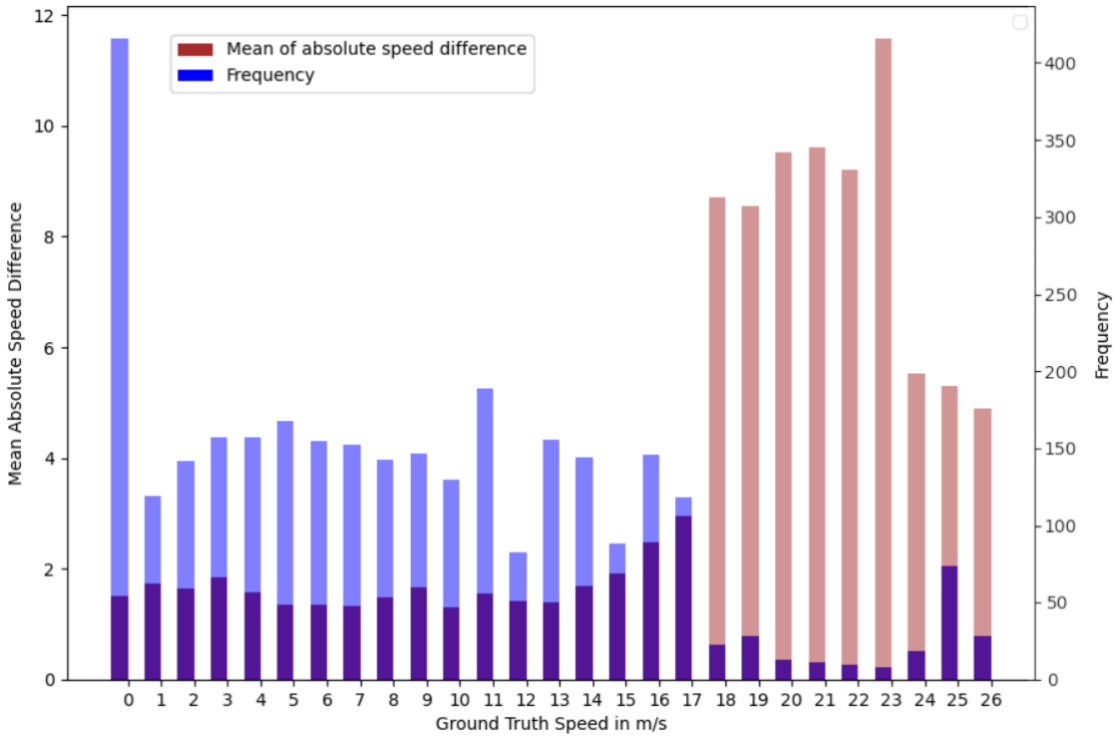


Figure 4.3 Mean of absolute speed difference on the test set

The global mean absolute speed difference is more than 6 m/s, but if we only look at speed from 2 m/s to 16 m/s, the error becomes less than 2 m/s.

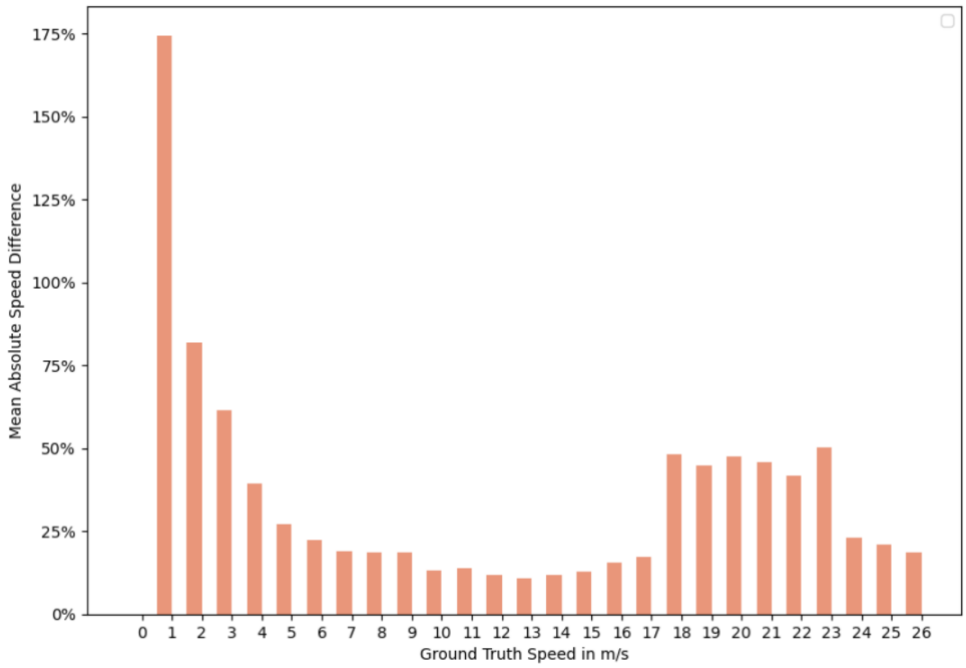


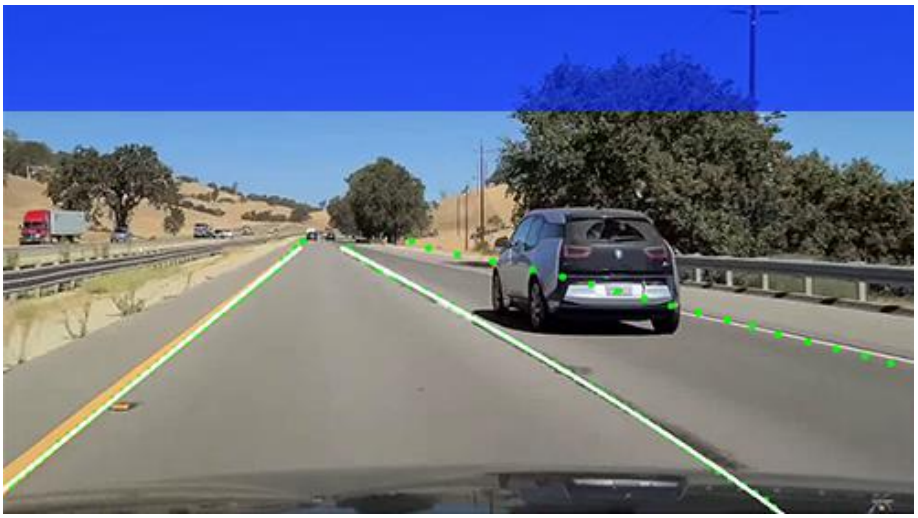
Figure 4.4 Mean of absolute speed difference in percentage on the test set

The model performs not very well at high speed, and for very low speed and 0 speed, the error was not too high, so it's unlikely to cause false alert in slow driving. Moreover, vehicle speed from 10 km/h to 60 km/h is representative for urban road driving, and when we apply "3-4 seconds rule" when driving at 20 m/s speed, the safe following distance will be above 60 or 80 meters, which is also beyond the robust detection range for lanes and following distance. And apparently the driver would not rely on this assist system when driving very slowly, the safe following distance can be flexible in traffic jam scenarios. Firstly, we apply Canny edge detection on the image, and use mask to extract the area of lanes

Figure 4.5 canny edge detection: original image, converted line drawing and mask

Then, we apply Hough transform to get all the straight lines, and poly fit them to get two continues lines as detection result, and at last, use slope to distinguish left and right lanes. The weakness of this method is the segmentation of the region of interest, it's limited by the influence of manual selection or post-processing technology.

In deep learning lane detection method, the functionality of mask is to crop the frame image, make the structure of image closer to those from the training dataset:



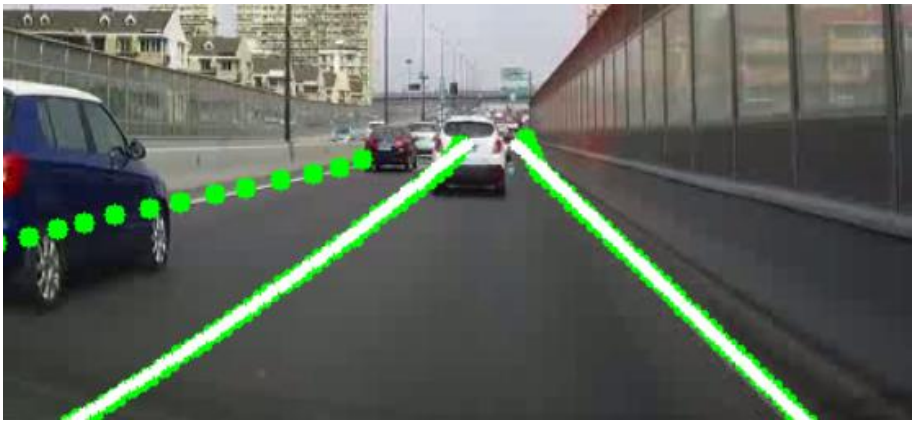


Figure 4.6 Deep learning lane detection

Corpping and resizing the image to 288×800 is all the preprocessing method we used for deep learning method in lane detection, it has solved the generalization problem, and general occlusion in traffic such as the front hood or other vehicles cannot influence the detection. And most importantly, this method still performs a very fast detection speed. For vehicle re-id, the similarity is defined by cosine distance:

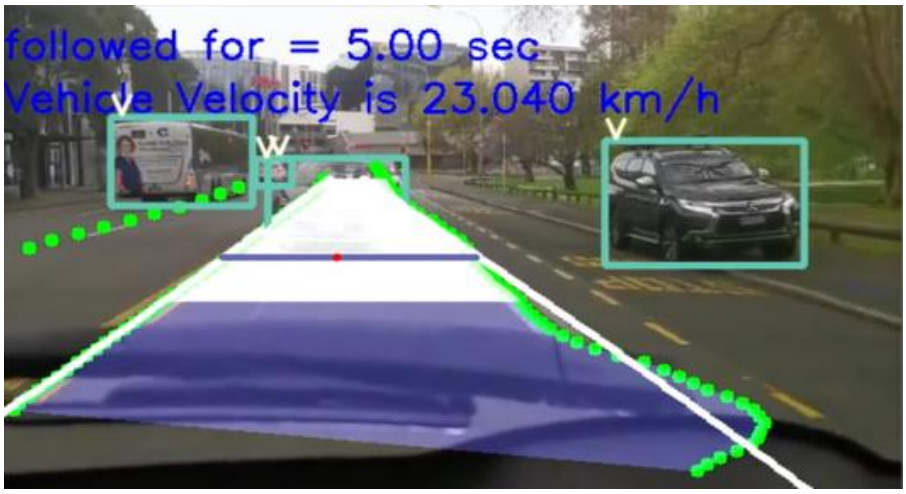
Figure 4.7 Vehicle re-id cosine distance, classification feature based and pixel based.

The image in left middle is the tracking target, and next two are the same vehicle in ideal condition, in forth image the brightness was dropped by 15%, fifth image is another vehicle, and last image is a non-vehicle object.

Pixel similarity performs well in ideal condition, and the classification feature similarity is more reliable in brightness fluctuations, and non-vehicle objects. Generally, YOLO will narrow the background, this deep learning advantage is not very obviously.

(a)

(b)



(c)

Figure 4.8 Demo of the safe following distance system

We test the entire system on a dash camera video, the white area in our lane shows us the safe following distance, which changes according to our driving speed, and for

vehicles following us, because of we don't have backward driving videos in the training dataset, we just switched the sequence of this frame and the previous frame image to produce the optical flow, so technique this speed is the "forward" speed in previous frame.

4.3 Limitations of the Research Work

We verified this speed estimation module on some real dash camera videos, and we found some typical failure cases:

(a) Predict: 9.4 m/s, GT: 17 m/s. (b) Predict: 6.57 m/s, GT: 3.1 m/s. (c) Predict: 18.2 m/s, GT: 30 m/s.

Figure 4.9 Speed estimation failure cases

In the first case, there is a huge occlusion in this image, and the luminance is going to surges substantially. And the second case, the vehicle is changing lane and moving slowly. The third case, there is an object that casting a reflection on the windshield, and the environment was too humdrum, the optical flow features were insufficient to express such speed.

(a) (b) (c)

Figure 4.10 optical flow error by video transcoding. The frame image (a), optical flow by transcode frame (b), and optical flow by original frame(c).

One interesting fact for optical flow is that when we use video transcoding software on the videos from internet, the fluctuation of video definition caused by transcoding has badly affected the optical flow estimation, it results in global random pixel movement between frames, and the outcome looks like a disaster. Other limitations are:

(1) Currently our method works only at straight lanes or Low angle curve lanes, as the distance estimation was based on the principle of perspective.

(2) The dataset, Veri-776 is sufficient for single view vehicle tracking, but KITTI dataset did not supply us enough sample of driving scenarios to distinguish the surrounding environment without extra depth information. Also, if there are unrecognized lane mark such as but station, it could interference the lane detection.

(3) The speed estimation was based on visual, if there are too many other traffic participants nearby, such as in a traffic jam, there's no way to sense if we are moving. However, it's also unusual to drive fast under crowded traffic environment.

(4) Actually this system cannot tell if the vehicle is moving forward or backward, because there were no negative speed values in the dataset.

(5) The lane detection used prior knowledge that the lanes should be at both side of the center of the X-axis, image need to be preprocessed to fit it.

(6) Special lane, such as double lane or reversible lane were not existing in our training dataset, there could be a deviation on the special lanes detected, and the road markings, for example, bus station road marking could influence our lane detection.

(7) The front vehicle tracking method will loss the target, if there are too large scale of change in illumination or occlusion that continues for more than the buffering time.

Chapter 5
Analysis and Discussions

In this chapter, we will analysis and discuss the experimental results from deep learning method compare to traditional method.

5.1 Analysis

In summary, our system will more stabilize when we can access the vehicle speed from the GPS, it becomes quite unstable in high speed driving, but it will still give us a referencing speed value with around 15% error range under speed between 4 m/s to 16 m/s, it's not as good as those methods that have used depth information, but acceptable for our safe following distance estimation. The deep learning lane detection method performs 96% accuracy on the test dataset, and it doesn't rely on the exacting mask to restrict region of interest to remove environmental interferences.

The distance detection is based on the lane width, it requires the focal length of camera in order to apply geometrical perspective method, and we could use the lane width and measured distance to create a fitting curve equation.

Vehicle similarity by classification feature is more stable in image brightness and background than pixel level similarity, ResNet performs 94.3 mAP for same camera re-id, and extremely high accuracy in finding the target vehicle in the top 20 candidates, which takes only one second time for following confirmation, it will also prevent the influence caused by a blinding flash of light.

5.2 Discussions

Deep learning method in lane detection is more adaptable and can be equally fast, and the accuracy of following distance depends on the lane width detection. And it is more reliable for vehicle tracking and re-id. Vehicle speed estimation from optical flow field can be used as an alternative, when other car-carried speed system are offline.

Chapter 6
Conclusion and Future Work

In this chapter, we will summarize the method we have used, answering the question of "How deep learning can improve the methods". And we planned our future work to address the weaknesses in this project.

6.1 Conclusion

The safe following distance monitoring system we proposed in this paper includes 40r modules. For lane detection, the traditional Canny edge and Hough transform method requires us to remove the environmental noise, and deep learning method doesn't have this region of interest issue, and performs a very fast detection speed.

We used the camera focal length, lane width at the bottom of frame and under the target vehicle as the parameters to apply the horizontal triangulation algorithm, which can calculate the actual distance to the target vehicle. For vehicle ego-speed estimation, RAFT can predict the pixel motion field between contiguous frames, and CNN can estimate the speed value from pixel motion to real-world motion with 2 m/s error rate in speed between 10 km/h to 60 km/h. and the following time counting is based on vehicle detection and re-id, YOLOv5 is very accurate in detection, and we set the cosine difference of ResNet classification confidence matrix similarity threshold to 97%, and we get 97.51% accuracy in same camera and same view point re-identification, it's very reliable when we use 20 frames to confirm the result, and it's more rubber with environmental noises.

6.2 Future Work

We will add scene recognition in our speed estimation module, for example, city, highway, or village road, and do some targeted training on our model and expend the dataset to fit more urban style. And try to remove those disturbed optical flow that caused by other moving vehicles.

The lane width could be affected by the camera distortion we will use depth estimation algorithms such as Monodepth2 (Clément, 2019), within the depth information, we can directly calculate the ego-speed of vehicle.

The dataset for speed estimation has too few samples from high speeds, and the distribution for other speeds was also not unified, which caused inherent bias in our model, we should find more data from other autonomous driving dataset, such as the BDDV dataset (Berkeley Deep Rrive Website, 2021) created in Berkeley, which included more

weather and illumination conditions. And the Apolloscape dataset (Apolloscape, 2021), which created in a modern city, it has the depth information for frames.

Currently the vehicle re-id performance is sufficient for tracking, but we can improve by using better network such as the Part-based Convolutional Baseline (Sun, 2018), and focus on the vehicle front and back side feature extraction, VehicleID dataset (Hongye, 2016) has 221,763 images for 26,267 vehicles which was much larger than Veri776 we used, and the image resolution is also higher, so there are more detailed visual features for each vehicle.

The optical inference time for RAFT is rather slow, although it's average end-point error is much higher, we have to balance the accuracy and system performance to achieve a real-time monitoring.

References

Alexandru, E. (2015). Dynamic 3D avatar creation from hand-held video input. ACM Transactions on Graphics, pp. 1 - 14.

Alexey, D. (2015). Flownet: Learning optical flow with convolutional networks. IEEE Int. Conference on Computer Vision (ICCV).

An, N., Yan, W. (2021) Multitarget tracking using Siamese neural networks. ACM Transactions on Multimedia Computing, Communications and Applications.

An, N. (2020) Anomalies Detection and Tracking Using Siamese Neural Networks. Master's Thesis. Auckland University of Technology, New Zealand.

Bao, D. (2016). Vehicle distance detection based on monocular vision. International Conference on Progress in Informatics and Computing (PIC), 187-191.

Bruce, D. (1981). An iterative image registration technique with an application to stereo vision. International Joint Conference on Artificial intelligence, pp. 674-679.

Butler, D. (2012). A naturalistic open source movie for optical flow evaluation. European Conf. on Computer Vision (ECCV), pp. 611--625.

Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 679-698.

Chen, W. (2017). A dense optical flow-based feature matching approach in visual odometry. International Conference on Intelligent Human-Machine Systems and Cybernetics, pp. 343-348.

Chunhui, Z. (2014). Visual odometry and scene matching integrated navigation system in UAV. International Conference on Information Fusion (FUSION), pp. 1-6.

Davy, N. (2018). Towards end-to-end lane detection: An instance segmentation approach. IEEE Intelligent Vehicles Symposium, p. 5.

Deng, G. (2018). Double lane line edge detection method based on constraint conditions Hough transform. International Symposium on Distributed Computing and Applications for Business Engineering and Science, pp. 107-110.

Feng, G. (2010). Modelling and simulation for safe following distance based on vehicle braking process. International Conference on E-Business Engineering, pp. 385-388.

Ge, Y. (2003). Autonomous Vehicle Positioning with GPS in Urban Canyon

Environments. IEEE Transactions on Robotics and automation, pp. 15–25.

Geiger, P. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354-3361.

Gowdra, N., Sinha, R., MacDonell, S., Yan, W. (2021) Maximum Categorical Cross Entropy (MCCE): A noise-robust alternative loss function to mitigate racial bias in Convolutional Neural Networks (CNNs) by reducing overfitting. *Pattern Recognition.*

Gowdra, N. (2021) *Entropy-Based Optimization Strategies for Convolutional Neural Networks.* PhD Thesis, Auckland University of Technology, New Zealand.

Gu, Q., Yang, J., Kong, L., Yan, W., Klette, R. (2017) Embedded and real-time vehicle detection system for challenging on-road scenes. *Optical Engineering*, 56 (6), 063102.

Gu, Q., Yang, J., Yan, W., Klette, R. (2017) Integrated multi-scale event verification in an augmented foreground motion space. Pacific-Rim Symposium on Image and Video Technology (pp.488-500)

Gu, Q., Yang, J., Yan, W., Li, Y., Klette, R. (2017) Local Fast R-CNN flow for object-centric event recognition in complex traffic scenes. Pacific-Rim Symposium on Image and Video Technology (pp.439-452)

Gutev, A. (2019). Exploiting depth information to increase object tracking robustness. International Conference on Smart Technologies, pp. 1-5.

Han, I. (2016). Car speed estimation based on cross-ratio using video data of car-mounted camera. Science International, 89–96.

He, B. (2019). Part-regularized near-duplicate vehicle Re-identification. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3992-4000.

He, K. (2016). Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778.

Herrera, A., Beck, A., Bell, D., Miller, P., Wu, Q., Yan, W. (2008) Behavior analysis and prediction in image sequences using rough sets. *International Machine Vision and Image Processing Conference* (pp.71-76)

Hongru, H. (2021). An effective method for lane detection in complex situations. International Symposium on Next Generation Electronics (ISNE), 1-4.

Hongye, L. (2016). Deep relative distance learning: Tell the difference between similar

vehicles. IEEE Conference on Computer Vision and Pattern Recognition, pp. 2167-2175.

Ilg, E. (2017). FlowNet 2.0: Evolution of optical flow estimation with deep networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Jia, D. (2021). BRAFT: Recurrent all-pairs field transforms for optical flow based on correlation blocks. IEEE Signal Processing Letters, pp. 1575-1579.

Jiang, L. (2019). Lane line detection optimization algorithm based on improved Hough transform and R-Least squares with dual removal. IEEE Advanced Information Technology, Electronic and Automation Control Conference, pp. 186-190.

Jiang, N. (2018). Multi-attribute driven vehicle re-identification with spatial-temporal re-ranking. IEEE International Conference on Image Processing (ICIP), pp. 858-862.

Jiao, Y., Weir, J., Yan, W. (2011) Flame detection in surveillance. Journal of Multimedia 6 (1).

Jie, B. (2021). Robust detection and tracking method for moving object based on radar and camera data fusion. IEEE Sensors, 21(9), pp. 10761-10774.

Jue, W. (2017). Ordered pooling of optical flow sequences for action recognition. IEEE Winter Conference on Applications of Computer Vision, pp. 168-176.

Jung, S. (2018). License plate detection and recognition in unconstrained scenarios. ECCV, 593-609.

Kang, S. (2017). Traffic lane estimation using road width information. IEEE International Conference on Consumer Electronics, pp. 53-54.

Karthika, K. (2020). Distance estimation of preceding vehicle based on mono vision camera and artificial neural networks. International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-5).

Kieran, D., Yan, W. (2010) A framework for an event-driven video surveillance system. *Advanced Video and Signal Based Surveillance* (AVSS).

Kun, Y. (2016). Motion flow segmentation based on optical flow angle. International Conference on Wireless Communication and Sensor Networks, pp. 592-597.

Lee, S., Kim, J., Yoon, J., et al (2017). VPGNet: Vanishing point guided network for lane and road marking detection and recognition. IEEE International Conference on Computer Vision (ICCV), pp. 1965-1973.

Li, B. (2020). A robust odometry algorithm for intelligent railway vehicles based on data fusion of encoder and IMU. IEEE IECON, pp. 2749-2753.

Li, F., Zhang, Y., Yan, W., Klette, R. (2016) Adaptive and compressive target tracking based on feature point matching. *International Conference on Pattern Recognition* (ICPR), (pp.2734-2739).

Li, P. (2018) *Rotation Correction for License Plate Recognition*. Master's Thesis, Auckland University of Technology, New Zealand.

Li, P., Nguyen, M., Yan, W. (2018) Rotation correction for license plate recognition. *International Conference on Control, Automation and Robotics*.

Li, Y., Ming, Y., Zhang, Z., Yan, W., Wang, K. (2021) An adaptive ant colony algorithm for autonomous vehicles global path planning. *International Conference on Computer Supported Cooperative Work in Design*.

Lin, Q. (2019). Mobile robot self-localization using visual odometry based on ceiling vision. IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1435-1439.

Liu, X. (2016). A deep learning-based approach to progressive vehicle re-identification for urban surveillance. IEEE International Conference on Multimedia and Expo (ICME).

Liu, X. (2018). A region-aware deep model for vehicle re-identification. IEEE International Conference on Multimedia and Expo.

Liu, X., Nguyen, M., Yan, W. (2019) Vehicle-related scene understanding using deep learning. *Asian Conference on Pattern Recognition.*

Liu, X. (2019) *Vehicle-related Scene Understanding Using Deep Learning.* Master's Thesis, Auckland University of Technology, New Zealand.

Liu, X., Yan, W. (2020) Vehicle-related scene segmentation using CapsNets. *International Conference on Image and Vision Computing New Zealand.*

Liu, X., Yan, W. (2021) Traffic-light sign recognition using Capsule network. *Springer Multimedia Tools and Applications.*

Ma, S. (2017). Research on automatic parking systems based on parking scene recognition. IEEE Access, pp. 21901-21917.

Malvezzi, M. (2001). Train speed and position evaluation using wheel velocity measurements. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 220-224.

Tanvir, H. (2018). A new vehicle localization scheme based on combined optical camera communication and photogrammetry. Mobile Information Systems.

Memisevic, K. (2015). Learning visual odometry with a convolutional network. International Conference on Computer Vision Theory and Applications.

Mehtab, S., Yan, W. (2021) FlexiNet: Fast and accurate vehicle detection for autonomous vehicles-2D vehicle detection using deep neural network. *International Conference on Control and Computer Vision*.

Mehtab, S., Yan, W. (2022) Flexible neural network for fast and accurate road scene perception. *Multimedia Tools and Applications.*

Mehtab, S. Yan, W., Narayanan, A. (2022) 3D vehicle detection using cheap LiDAR and camera sensors. *International Conference on Image and Vision Computing New Zealand.*

Mich, R. B. (2001). Histograms analysis for image retrieval. Pattern Recogn, 34, pp. 1625-1637.

Ming, Y., Li, Y., Zhang, Z., Yan, W. (2021) A survey of path planning algorithms for autonomous vehicles. *International Journal of Commercial Vehicles*.

Nair, V. (2010). Rectified linear units improve restricted Boltzmann machines. Proceedings of ICML, pp. 807-814.

Nakamura, K., Ishigaki, K., Ogata, T., Muramatsu, S. (2013). Real-time monocular ranging by Bayesian triangulation. IEEE Intelligent Vehicles Symposium (IV), pp. 1368-1373.

Pan, C., Yan, W. (2018) A learning-based positive feedback in salient object detection. International Conference on Image and Vision Computing New Zealand.

Pan, C., Yan, W. (2020) Object detection based on saturation of visual perception. *Multimedia Tools and Applications*, 79 (27-28), 19925-19944.

Pan, C., Liu, J., Yan, W., Zhou, Y. (2021) Salient object detection based on visual perceptual saturation and two-stream hybrid networks. *IEEE Transactions on Image Processing.*

Qin, Z., Yan, W. (2021) Traffic-sign recognition using deep learning. *International Symposium on Geometry and Vision.*

Shen, D., Xin, C., Nguyen, M., Yan, W. (2018) Flame detection using deep learning. *International Conference on Control, Automation and Robotics.*

Shen, H., Kankanhalli, M., Srinivasan, S., Yan, W. (2004) Mosaic-based view enlargement for moving objects in motion pictures. *IEEE ICME'04*.

Shen, Y., Yan, W. (2019) Blind spot monitoring using deep learning. *International Conference on Image and Vision Computing New Zealand*.

Shen, Y. (2017). Learning deep neural networks for vehicle Re-ID with visual-spatio-temporal path proposals. IEEE International Conference on Computer Vision (ICCV), pp. 1918-1927.

Smith, M. B. (2008). The representation of egomotion in the human brain. Current Biology, 191–194.

Sun, Y. (2018). Beyond part models: Person retrieval with refined part pooling and a strong convolutional baseline. Lecture Notes in Computer Science, pp. 501–518.

Takagi, R. (2011). Synchronisation control of trains on the railway track controlled by the moving block signalling system. Electrical Systems in Transportation, pp. 130-138.

Tang, X. P. (2018). Spatial as deep: Spatial CNN for traffic scene understanding. AAAI.

Tian, L. (2011). Binocular vision system design and its active object tracking. International Symposium on Computational Intelligence and Design, pp. 278-281.

TsungYing, S. (2006). HSI color model based lane-marking detection. IEEE Intelligent Transportation Systems Conference, pp. 1168-1172.

Wang, J., Yan, W., Kankanhalli, M., Jain, R., Reinders, M. (2003) Adaptive monitoring for video surveillance. International Conference on Information, Communications and Signal Processing.

Wang, J., Kankanhalli, M., Yan, W., Jain, R. (2003) Experiential sampling for video surveillance. *ACM SIGMM International Workshop on Video surveillance* (pp.77-86).

Wang, J., Yan, W. (2016) BP-neural network for plate number recognition. International *Journal of Digital Crime and Forensics* (IJDCF) 8 (3), 34-45.

Wang, J. (2016) *Event-driven Traffic Ticketing System*. Master's Thesis, Auckland University of Technology, New Zealand.

Wang, J., Ngueyn, M., Yan, W. (2017) A framework of event-driven traffic ticketing system. *International Journal of Digital Crime and Forensics* (IJDCF) 9 (1), 39-50.

Wang, J., Bacic, B., Yan, W. (2018) An effective method for plate number recognition.

Multimedia Tools and Applications, 77 (2), 1679-1692.

Wang, L. (2020). Robust regression model based on low rank representation. Computer Engineering, 46, pp. 74-79.

Watcharapinchai, N. (2017). Approximate license plate string matching for vehicle re-identification. IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).

Xing, J., Yan, W. (2021) Traffic sign recognition using guided image filtering. *International Symposium on Geometry and Vision.*

Xing, J., Nguyen, M., Yan, W. (2022) The improved framework of traffic sign recognition by using guided image filtering. *Springer Nature Computer Science.*

Xing, J. (2022) *Traffic Sign Recognition from Digital Images Using Deep Learning.* Master's Thesis, Auckland University of Technology, New Zealand.

Yan, W., Kankanhalli, M., Wang, J., Reinders, M. (2003) Experiential sampling for monitoring. *ACM SIGMM Workshop on Experiential Telepresence*, 70-72.

Yan, W., Jain, R. (2008) An event-based web monitoring environment. *International Conference on Communications and Networking in China.*

Yan, W. (2019) *Introduction to Intelligent Surveillance: Surveillance Data Capture, Transmission, and Analytics*. Springer London.

Yan, W. (2021) *Computational Methods for Deep Learning: Theoretic, Practice and Applications*. Springer London.

Zeng, W. (2009). Point matching estimation for moving object tracking based on Kalman filter. IEEE/ACIS International Conference on Computer and Information Science, pp. 1115-1119.

Zequn, Q. (2020). Ultra fast structure-aware deep lane detection. ECCV 2020, pp. 276-291.

Zhe, T. (2020). Inter-vehicle distance estimation method based on monocular vision using 3D detection. IEEE Transactions on Vehicular Technology, 69(5), 4907-4919.

Zhong, Z. (2017). Re-ranking person re-identification with *k*-reciprocal encoding. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3652-3661.