



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Lane Line Detection

by

Jinwei Yu

Supervisor: Gordon Dickers

Second Marker:

Project submitted as part of the requirements for
the award of MSc Data and Artificial Intelligence

August 2022

DECLARATION

I, ...Jinwei Yu..... declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : Jinwei Yu.....

Date : 20/07/2022.....

FORM OF CONSENT

I Jinwei Yu..... hereby consent that my Project, submitted in candidature for the M.Sc. Data and Artificial Intelligence degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : Jinwei Yu.....

Date : 20/07/2022.....

Abstract

Due to the progressive growth of the transportation sector in recent years and the yearly rise in traffic accidents, society has become increasingly concerned about traffic safety. To lessen the frequency of traffic accidents and to promote driving safety, scholars have conducted a series of studies on vehicle-assisted driving. Research shows that image-based lane line detection and lane departure warning technologies have become the main research directions in vehicle-assisted driving systems. The thesis focuses on the problems of lane line detection in vehicle-assisted driving systems by studying a variety of complex road scenarios such as lane lines being blocked by other obstacles; road shadows produced by structures and trees on each side of the street; faded lane lines, inconsistent colors and uneven illumination, and other lane line detection algorithms. Different algorithms are used to fit different lane line types and scenarios.

The main points are as follows.

1. Enhancing lane characteristics in the vehicle camera's initial image in order to get ready for later lane line detection. The pre-processing process includes binarization, noise reduction for the target region, region of interest extraction, and then threshold segmentation based on the lane line edge features to distinguish the target lane line from the background.
2. The Hough transform technique is utilized for fitting in road sceneries with straight lane lines and standard lighting. Firstly, the Canny operator is used for edge detection, and then the Hough transform algorithm is used to fit the lane line edge pixels.
3. For complex road scenes with curved lane line types, large road shadows, and faded lane lines, a sliding window polynomial-based algorithm is used to fit their lane lines. The segmentation approach is enhanced in this study by combining many thresholds to distinguish the target from the background.; finally, the curve is fitted using a sliding

window polynomial fitting method.

4. For road scenarios with multiple lane lines and lane lines obscured by other obstacles, the multiple lane line detection problems are converted into a semantic segmentation problem, proposing to improve the hourglass module to adopt the structure of encoder-decoder plus stubbing layer, and selecting two hourglass modules stacked for feature extraction of the image, making the whole network more accurate in controlling local and global information, reducing the computational effort and improving the model detection speed. Compared with traditional image processing methods, the algorithm is more adaptable to complex scenarios such as lane lines being obscured and unclear lanes ahead, while improving robustness and having good detection accuracy.

Table of contents

Abstract	I
Table of contents	i
Chapter 1: Introduction.....	1
Chapter 2: Literature review	4
2.1 Lane-line detection based on traditional methods	5
2.1.1 Method for detecting lane lines with features.....	5
2.1.2 Method for detecting lane lines based on models	6
2.2 Deep learning-based lane-line detection.....	7
2.2.1 Using categorization techniques, identify lanes.....	7
2.2.2 Using target detection to determine lane lines.....	8
2.2.3 Lane line identification based on segmentation	9
Chapter 3 Methodology	12
3.1 Lane line fitting based on traditional methods	12
3.1.1 Picture greyscaling.....	13
3.1.2 Filtering and smoothing.....	15
3.1.3 Binarization.....	17
3.1.4 Detecting edges.....	19
3.1.5 Area of interest extraction.....	20
3.2 Deep learning-based lane line fitting.....	21
This section describes several important components of CNNs.	21
3.2.1 Convolutional layer.....	21
3.2.2 Pooling layer	23
3.2.3 Activation layer	24
3.2.4 Fully connected layer	27
3.2.4 Deconvolution layer	28
3.2.5 Several classical neural network models.....	28
Chapter 4: Results and discussion	33
4.1 Detection method based on the Hough transform model	33
4.1.1 Principle of the Hough linear detection algorithm	33

4.1.2 Analysis of Hough transform model detection results	34
4.2 Detection method based on polynomial fitting of sliding windows	37
4.2.1 Perspective transformation	38
4.2.2 Locating the lane line basepoint	39
4.2.3 Sliding window traversal and detection.....	40
4.3 Deep learning-based lane line detection method.....	41
4.3.1 The dataset.....	41
4.3.2 Overall network structure	42
4.3.3 Data pre-processing	46
4.3.4 Loss function selection	47
4.3.5 Training details.....	49
4.3.6 Evaluation indicators	50
4.3.7 The result	50
Chapter 5: Conclusion	53
Chapter 6 Reflection	55
Reference	58

Chapter 1: Introduction

The car sector has grown to become one of the largest and most important worldwide industries, significantly boosting the economies of Germany, Japan, the US, and other nations. The growth of other related industries and the evolution of the industrial structure are both heavily influenced by the car industry in every industrialized country. Due to the contradictory trends of quickly rising vehicle sales and a lack of essential road infrastructure, several nations are currently experiencing difficulties. In many different nations, undesirable negative occurrences including traffic delays, energy waste, motor vehicle accidents, and pollution are on the rise. According to the latest United Nations statistics, up to 50 million people are injured and approximately 1.26 million are killed per year in traffic collisions globally, resulting in an economic cost of nearly US\$1.9 trillion [1]. According to surveys, a motorist should take preventative action at least 0.5 seconds before a critical scenario in order to be able to avoid a traffic collision [2]. Therefore, reducing the number of traffic collisions is a continuing objective of action.

One of the areas of worry is autonomous driving technology, which has emerged as a result of the convergence of the conventional automotive sector and the rising information technology industry, together with the rapid advancement of artificial intelligence technology. [3]. Driverless technology's central component may be viewed as a sophisticated intelligent control system that combines a number of sub-functional modules, including mechanical control, path planning, and intelligent sensing. The vehicle may function autonomously under the control of a single system thanks to the cooperative operation of the numerous elements [4]. The creation of intelligent urban transportation is the goal of autonomous technology research and development. The technology provides solutions to a variety of issues, including accidents and snarled up city traffic, while ensuring that drivers have a safe, comfortable, and enjoyable driving experience. Therefore, the development of autonomous technology is highly important and has a very bright future [5].

Unmanned vehicles' ability to accurately locate lane lines in real-world driving scenarios is a critical component of their perception capability, which is made possible by a key technology in the realisation of driverless vehicles: the ability to perceive the surrounding driving scene while the vehicle is in motion. However, lane line detection accuracy is also crucial for the autonomous vehicle's decision-making and has a direct bearing on the safety of the car on the road [6]. Technology for lane line detection is also crucial for currently installed advanced driver assistance systems [7]. In order to ensure driver safety, advanced driver assistance systems for cars need a number of subsystems to work together. These subsystems include lane departure warning, forward collision warning, intersection assisted mobility, pedestrian collision prevention, lane departure warning, and left turn assist. By increasing lane line identification accuracy, cars may provide more accurate lane departure and lane change warnings, which lowers the likelihood of driver irresponsibility causing accidents and fewer personal injury incidents. Investigating lane line detection algorithms is very significant because high-quality lane line identification findings are a crucial assurance of driving safety.

Two major categories may be made for lane detecting algorithms.: those that extract features from images before relying on algorithm fitting [8], and those based on deep learning [9] methods to identify lane lines in driving scenes. In the former category, most algorithms require the extraction of picture characteristics including color [10] and edge data [11]. Then, a variety of additional algorithms are frequently utilized to assist in lane identification, such as the Randomly Sampled Consistent Algorithm (RANSAC) [12] and the Hough Transform method [13]. However, to achieve good results, such methods frequently rely on the selection of particular features in a given scene, which is challenging in challenging driving scenarios. Autonomous driving requires a high level of driving scenario knowledge, particularly in a range of complicated driving scenarios. In this context, lane line detection techniques based on deep learning [14] offer a great advantage. Deep learning has helped new discoveries in a growing number of driverless-

related research domains, including the segmentation of driving sceneries, the identification of drivable locations, the detection of driver visual saliency [15], and others. Feature processing of driving scenes in terms of spatial relationships by means of deep convolutional neural networks can provide sufficient feature information for different perception tasks. Therefore, deep learning-based lane line detection algorithms are a work of future development trend and research significance since they are more in line with the requirement for intelligent automobiles.

Chapter 2: Literature review

A variety of lane line detection methods have recently been offered by several academics both domestically and internationally. The various detection techniques can be divided into three categories: vision-based, LIDAR-based, and LIDAR and vision-sensing-based lane line detection. LIDAR-based detection systems use the LIDAR to rapidly fire millions of laser beams in order to correctly determine the distance and form of the surrounding objects by their difference in return time. As a result, a 3D rendering of the local surroundings is produced. Although LIDAR-based lane line detection algorithms can precisely range objects regardless of illumination conditions, their adoption is constrained by their high cost and weight, which make them difficult to integrate into a vehicle's interior. In order to completely perceive the lane scene, millimetre wave radar, ultrasonic radar, and low-cost LIDAR are included as supplements to camera detection, which is the fundamental component of detection systems that integrate visual perception with LIDAR. In situations when visual sensors fail, visual data can be combined with other sensor modalities to enhance estimations of the road scene (such as inclement weather or off-road driving). To locate lane boundaries, identify vehicles and other impediments like pedestrians, and fully comprehend the landscape around them, One camera and deep learning algorithms are used in vision sensor-based detection systems. This strategy differs from the other two in that it does not rely on pre-collected high-precision map data or external infrastructure. Vision sensors are frequently employed in the field of autonomous driving and have several advantages over other types of sensors, such as LIDAR, including excellent resolution, low power usage, cheap cost, simplicity of aesthetic integration, and non-intrusiveness. This research focuses on visual perception-based vision-aware lane line recognition methods.

In order to accurately locate each lane in a picture and determine the type of lane line, vision-based lane line recognition algorithms must be used. The bulk of techniques are lane line modelling and feature extraction methods. The lane modelling algorithm derives

a workable phrase employing linear or non-linear fitting methods for the spatial location of the lane lines, whereas the lane lines' borders, color, and texture are all captured by the feature extraction technique. Both conventional image processing-based and deep learning-based lane recognition techniques fall within the category of mainstream lane detection approaches.

2.1 Lane-line detection using conventional techniques

Feature-based lane line detection methods and model-based lane line detection methods are two categories for conventional lane line detection approaches. These techniques are typically based on models or hand-made features.

2.1.1 Method for detecting lane lines with features

Lane lines are often elongated rectangles that are white or yellow and considerably different from the road surface. Because of this, feature-based lane line identification methods frequently use an analysis of lane lines' edges, gradients of pixel points, color, and other visual characteristics to identify and locate them.

Niu et al. [16] present a two-stage feature extraction lane line detection approach. The lane line sample data are first fitted using the Gaussian Statistical Colour Model (GSCM), and a segmentation threshold is calculated to binarize the image with the desired false negative rate as the target. Next, the lane line region of interest is divided based on the position of the horizon, and the lane line edges are extracted using the Sobel operator in the chosen region. Finally, a modified Hough transform is used to identify the lane lines and vanishing points. The report also suggests using the relationship between the world coordinate system and the image coordinate system to determine the distance and yaw angle of the vehicle with respect to the lane line.

Bertozzi et al [17] suggested a lane detecting method based on color. The article makes

the case that the threshold of the edge detection algorithm value setting has a substantial influence on the amount of interference information that is still present in the binary picture produced. It also describes a technique for determining an appropriate threshold value by iterating. Since the lane lines' form and relative location in the image are independent of the resolution size, the algorithm's effectiveness may be increased by The efficiency of the method may be increased by decreasing the picture resolution since the relative location and form of the lane lines in the image are independent of resolution size.

In order to determine the texture direction of each pixel, Rasmussen et al. [18] first proposed a driving area created using line segments going in the same direction as the vanishing point, and lane lines are then extracted from it and estimated the vanishing point using Hough voting.

2.1.2 Method for detecting lane lines based on models

Lane lines have significant geometrical and architectural qualities in addition to the appearance. The lane lines often include parallel, broken, or continuous straight lines or curves, intersect at an infinite point, and may also have additional structural components. These characteristics are frequently used by model-based lane line detection techniques to extract ROIs before characterizing the structure of the lane lines with mathematical models. Common mathematical models include straight lines, parabolas, hyperbolas, and spline curves.

A technique for identifying and following lanes using multi-frame spatiotemporal pictures is presented by Jung et al. in [19]. The technique detects two parallel lines that correspond to the lane edges using the Hough transform and the temporal consistency of lane width, and then fits the identified lane lines into a cubic curve model using a weighted least squares approach.

For the cubic spline model, Beriel et al. [20] provide a method for tracking lane markers based on particle filtering. The procedure starts by extracting the ROI from the inverse perspective picture, then utilizes an adaptive threshold to locate potential regions for lane lines, and lastly smooths the lane markers using particle filtering. For the lane line spline model, there are three control points.

A single curve model can fit lane lines at curves better but is much more computationally costly and cannot balance real-time and detection accuracy. In contrast, a single straight-line model is quick and accurate for fitting straight lane lines but cannot be used to detect lane lines at curves. Therefore, a mixed model detection approach has been suggested by numerous academics. In order to detect straight lines and curves, respectively, Zhou et al. [21] proposed a straight-curve model for curve detection algorithm. This algorithm first detects straight lines using an improved Hough transform, and then builds a polynomial curve model to detect curves based on the continuity of road lane lines and the tangent relationship between straight lines and curve models.

2.2 Lane-line detection based on deep learning

The domains of identification, classification, segmentation, and detection of still pictures have all benefited greatly from the usage of CNN, which are capable of learning the parameters of a network model by training on huge data sets. It has recently become commonly employed for lane detecting in the direction. The use of CNN for lane line recognition is more accurate and resilient than utilising classic lane detection algorithms since it can acquire multi-scale information of the picture. Because of this, it is more suited for challenging driving scenarios in everyday circumstances. Lane line recognition is becoming increasingly accurate as large-scale road traffic statistics are made accessible more often and computer hardware infrastructure is continually upgraded.

2.2.1 Using categorization techniques, identify lanes

Sorting various photographs into pre-established classifications is the aim of the image classification task. As a result, before lane identification can be accomplished using classification techniques, the position of the lane must be defined by using a prediction in lane line detection approaches based on classification methods.

When extracting features, conventional CNN network models look for global pictures. However, it also uses up hardware resources and sharply increases the interference of duplicate information. In the CNN framework, Kim J. et al.[22] presented a quick learning technique. In the CNN framework, a stacked ELM design is suggested to quickly compute the network weights between the output and hidden layers. network weights in a single cycle to speed up learning and produce reliable outcomes with less training data.

The region in the image that needed to be identified was initially split into two halves by Gurchian et al. [23] Since the on-board camera acquired lane lines were typically concentrated in the lower half of the image, the curve fitting algorithm was chosen to determine the parameters to fit the lane lines in that region of the image, and the complete lane lines were then fitted in accordance with the location of the vanishing point in conjunction with the lower part of the lane lines.

2.2.2 Using target detection to determine lane lines

One way to think of target detection is as a problem of localization and categorization for various targets. Another efficient technique for lane line recognition is regression of lane dimensions using a focus detection method. In the majority of cases, lane line based on target identification begins identifying each section of the lane line's feature points or boundary boxes before detecting the lane line using coordinate regression.

Huval et al. [24] suggested a dual-view convolutional neural network to increase the network's identification accuracy while utilising the top and front views to weed out

unnecessary data and preserve the road-like structure. To find the ideal arrangement of lane lines, a weighted cap filter and a global optimization function were developed. In the event of road obstruction, lane line breakdown, etc., the target detection approach has shown good results for lane line extraction. When the road is obstructed or the lane lines are damaged, it is impossible to efficiently remove the lane lines.

In the literature, a Spatial Convolutional Neural Network (SCNN) for lane line recognition has been suggested [25]. To transmit information across rows and columns of an image, feature mapping converts the conventional convolutional layer-by-layer connection form into a slice-by-slice convolutional form. The algorithm is quite effective in locating distant continuous-shaped objects as well as huge targets. It is especially effective at finding items like utility poles and traffic lanes that have strong spatial links but inadequate visual clues. In difficult settings and for object recognition in CityScapes, the system performs admirably.

For the purpose of lane line recognition, a Dual-View Convolutional Neural Network (DVCNN) architecture was put out in the literature [26]. A cap filter is proposed that can handle potential lane lines and smooth gradient texture distribution, reducing the likelihood of false detections; next, a new DVCNN architecture is designed, primarily consisting of a front view and a top view, using the front view to exclude moving cars, guardrails, and roadside objects, and the top view to filter out objects that resemble the shape of lane lines such as arrows;

2.2.3 Lane line identification based on segmentation

The categorization of an image at the pixel level is known as segmentation. Lane lines and background pixel points are frequently marked as belonging to separate classes in segmentation-based lane line identification, which subsequently uses semantic segmentation or instance segmentation to categorize the picture down to the pixel level

[27].

(1) A sample segmentation-based technique for detecting lane lines

For the purpose of detecting lane lines, Pan et al. [28] suggest the SCNN special neural network structure. The feature map's pixels may now convey spatial information across rows and columns. This structure is highly useful for locating things like lane lines and utility poles that have strong spatial links but few shape signals. The technique only manages a detection speed of 7.5 FPS due to the high computing cost required for spatial message transfer at the pixel level.

A lane line recognition system employing a mix of semantic segmentation and clustering approaches was proposed by Hou et al. in [29]. The feature information of lane lines was extracted using a complete convolutional network with an encoding-decoding structure, and the original convolution was replaced with a null convolution to broaden the network's perceptual range. The segmentation model is then directly relied on the network output to execute fitting and clustering operations to separate various lane lines into distinct instances.

(2) Lane line identification via semantic segmentation

Before using a semantic segmentation network, Neven et al. [30] changed the input picture to a top view viewpoint. For the purpose of extracting lane line features, the input picture is transformed to a top view viewpoint and then sent to a convolutional neural network model. To lessen the distortion brought on by the input image's spatial translation, the network learns the inverse perspective parameters in the inverse perspective transformation matrix during the training phase. The issue of the image's near and far ends having an unequal distribution of lane line pixels is successfully resolved. However, using this strategy necessitates the network model to perform more computations, which increases the amount of pixels on the road surface. Changes in the driving surface's slope have a little impact on the results of the inverse perspective transformation; as a result,

the results are unsatisfactory during the real inspection procedure. In the course of the real inspection, the results are not adequate.

Hsu et al. [31] describes a technique for segmenting lane lines on a highway using a deep neural network, which is an instance-level segmentation. The segmentation output is pixels per lane line, and the lane line parameters are then fitted using least squares. For semantic segmentation of lane line areas, the literature employs a network design with convolutional and deconvolutional symmetry.

In order to identify lane lines, Chang et al. [32] offer an instance segmentation method. A lane segmentation branch and a lane embedding branch make up the end-to-end network of the approach, which is built as a branching, multi-tasking network (Lane Net). Additionally, it suggests using a learning adaptive perspective transformation approach rather of a preset perspective transformation matrix. A neural network (H-Net) is trained to produce adaptive perspective transformation parameters, in contrast to a fixed transformation matrix. The approach also suggests a technique for adapting viewpoint transformation.

Chapter 3 Methodology

3.1 Lane line fitting based on traditional methods

Onboard cameras provide the data for lane line detection, and often, these images encompass a lot of irrelevant and distracting material, leading to erroneous lane line recognition. The collected pictures must be pre-processed in order to increase the detection effectiveness of lane line detection by emphasizing more valuable information and lowering the detection difficulty. The processed picture must make sure that both the finer details of the image and the essential lane edge information are preserved. Inaccurate pre-processing results will have a direct impact on the outcomes and analysis of the subsequent image processing, making it impossible to recognize the target lane lines precisely.

The image pre-processing technique employed in this study entails de-noising the picture by greying, filtering, and smoothing the camera-captured image; conducting gradient-based edge detection and image thresholding segmentation based on the color, edge, texture, and specific information of the lane lines; and then strengthening the edge features of the lane lines so as to identify the target lane lines from the backdrop, by area of interest selection. The image is pre-processed using image thresholding and gradient-based edge detection. In Figure 3.1, the pre-processing procedures are depicted.

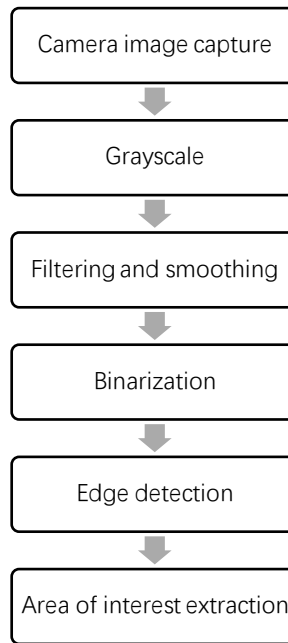


Figure 3.1 Image pre-processing steps

3.1.1 Picture greyscaling

The initial image captured by the car's camera is often in color. Each image's pixel points are composed of red, green, and blue components, and they can all have values in the range between 0 and 255 on these three channels. It would be impossible to achieve the real-time requirements without processing the majority of the worthless information in a color image. For this reason, it is required to greyscale pictures with a range of vivid colors into a single-channel image. By lowering the frequency of hardware operations, this decreases the volume of data while still ensuring real-time performance. Image greyscaling's function is to transform a color image with plenty of information into a greyscale image. The widely used greyscale techniques. The commonly used greyscale methods are:

(1) Component method

The luminance of a component of R, G and B is extracted separately and the color image is greyed out using the luminance.

(2) Maximum value method

For each pixel point of the different color channels take the maximum value on the 3 channels as the greyscale value of the pixel, as in Formula (3.1), the resulting greyscale map will be relatively brighter overall.

$$\text{gray}(i, j) = \max(R(i, j), G(i, j), B(i, j)) \quad \text{Formula (3.1)}$$

(3) Mean value method

The values of the three channels for each pixel point are averaged to obtain a grey-scale image, as in Formula (3.2).

$$\text{gray}(i, j) = \frac{1}{3}(R(i, j), G(i, j), B(i, j)) \quad \text{Formula (3.2)}$$

This thesis uses three different methods to convert the original image to a greyscale method, a comparison of which is shown in Figure 3.2.



(a) Original image



(b) Component method



(c) Maximum value method



(d) Mean value method

Figure 3.2 Comparison of each greyscale chart

From the analysis of the principle and effect of greyscale processing, all the above images can be converted into single-channel greyscale images with very clear road edges. In this thesis, the mean value method was chosen for the greyscale processing.

3.1.2 Filtering and smoothing

Due to the possibility of certain external surroundings on the real road disturbing the pictures taken by the vehicle-mounted camera, noise may be formed in specific parts of the image. This might result in poor picture quality and impair the detection and recognition of lane lines behind. The picture must be smoothed and filtered to remove isolated noise, while still maintaining the edge information and limiting the effect of outside noise on the detection results, in order to enhance the detection results. To accomplish linear filtering, two techniques are frequently used: mean filtering and Gaussian filtering.

(1) Mean value filtering

The principle of mean filtering is to select a small area of the image, add up the grey values of the pixels in that area and average them, then use the average value as the value of all the pixels in that area. A fixed template is usually set at 3×3 and the middle pixel and the 8 pixels around it form a template, which is then averaged over the 9 pixels to

give the pixel value of all the points in the area. According to its principle, mean filtering removes noise from the image by removing points that do not match the surrounding pixels.

(2) Gaussian filtering

Gaussian filtering is broadly similar to mean filtering in that it uses a fixed template to scan each pixel in the image, the difference being that Gaussian filtering multiplies each pixel by a different weight, and this weight is determined by a Gaussian function. The method is good at suppressing noise when the noise points generated by the image conform to a normal distribution. The function of Gaussian filtering is given in Formula (3.3), with the expression

$$f(x) = \frac{1}{\delta\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\delta^2}} \quad \text{Formula (3.3)}$$

where δ is the standard deviation of x and μ is the mean of x .

After introducing the filtering method, the noise generated by the camera was analyzed and a better filter was selected to filter out as much of the noise present on the road as possible while ensuring accuracy. A comparison of the filtering methods is shown in Figure 3.3.



(a) Mean value filtering



(b) Gaussian filtering

Figure 3.3 Comparison chart of the two filtering methods

From the analysis of the filtering principles and effects, the above images all remove noise very well, while the road edge information is very clear. In the thesis, the filtering method

chosen is Gaussian filtering in linear filtering.

3.1.3 Binarization

The lane line target must be differentiated from the road backdrop by binarization in order to extract the lane lines from the road picture. Binarization reduces all of the picture's pixels to a black and white image with only two grey values by starting with a grey-scale image with 256 brightness levels. The binarization procedure not only minimizes storage requirements and processing complexity, but also lessens interference from non-target elements.

Comparing the picture $f(x,y)$ with each pixel point (x,y) and a predetermined threshold T is the fundamental concept behind the binarization process. All pixels are shown in white when the pixel segmentation threshold T is exceeded, and in black when T is exceeded. The image is binarized as shown in Formula (3.4).

$$s(x,y) = \begin{cases} 0, & f(x,y) < T \\ 255, & f(x,y) \geq T \end{cases} \quad \text{Formula (3.4)}$$

where $s(x,y)$ denotes the greyscale value of (x,y) after binarization; $f(x,y)$ denotes the greyscale map of (x,y) before binarization; and T denotes the segmentation threshold.

Threshold selection is the most important thing in binarization processing. If the threshold is chosen too small, fine noise in the image will be extracted; conversely, if the threshold is chosen too large, some of the target detail will be lost, resulting in target loss and affecting the final extraction result. Considering the changes in lighting and complex weather during lane line acquisition, as well as the shadows of buildings and trees around the lane, if the segmentation threshold is set to a fixed value, the binarization effect will not be satisfactory, so for different situations, different thresholds must be adopted for segmentation. The following methods of thresholding are commonly used, namely global thresholding, local thresholding and adaptive thresholding.

(1) Global thresholding principle

Global thresholding is the process of finding the best segmentation threshold by using global information about the whole image. By setting different feature thresholds, the pixel points in an image can be divided into several categories. By analyzing the greyscale and color characteristics of the input original image, the image can be divided into two parts by selecting the optimal segmentation threshold. In the thesis, the image is first converted into a greyscale image and then the optimal segmentation threshold is set to 127 by using the global information of the whole image. when a pixel exceeds the threshold, the part is not changed and the other part is set to 0.

(2) Local Thresholding Principle

Local thresholding is similar to the global thresholding principle, except that local thresholding first divides the image into blocks and then divides the image into blocks by setting different thresholds, noting that if the segmentation area set in this method is too small, the result will be poor.

(3) Adaptive thresholding principle

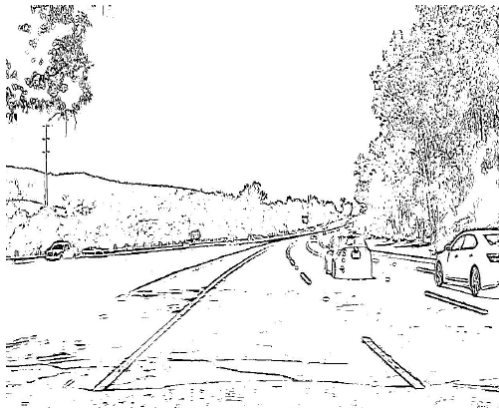
Adaptive threshold segmentation, i.e., segmentation using different thresholds for different scenes, is not a fixed value, but the threshold is automatically adjusted as the environment varies. The method has good applicability in different environments. The following thresholding methods were used to process the images, which are shown in Figure 3.4.



(a) Original image



(b) Global threshold segmentation



(c) Adaptive threshold segmentation

Figure 3.4 Comparison chart of threshold segmentation methods

As can be seen from Figure 3.4, the adaptive thresholding method was used to obtain a partially blurred road, while the global thresholding segmentation method not only fits accurately to the edge information of the lane lines but also filters out the noise extremely well.

3.1.4 Detecting edges

Edge detection algorithms are frequently used to extract edge information from the lane lines after creating a binary contour map of the lane. Edges are a class of points that differ significantly from the grey value of the adjacent pixels and are rich in information due to abrupt changes in information such as greyscale or structure.

The majority of edge detection techniques now in use are gradient-based algorithms, which employ differentiation operations in a picture to identify whether or not an edge is there. It is clear that the derivative's largest value identifies the location where the change is greatest. The Roberts, Prewitt, Sobel, Laplacian, and Canny operators are often used edge detection operators. The above operators are based on features with large variations in image edge information, and the various operator features are used to set corresponding thresholds to calculate gradient intensity variations to detect edges, and finally, based on the pixel point values the edge extraction is based on whether the pixel value exceeds the threshold or not.

3.1.5 Area of interest extraction

By looking at how the road scene is laid out in the image, the analysis reveals that the road information is complicated and varies when driving, increasing the complexity of lane line extraction. Non-lane line interference may be removed by extracting the target region of interest [33]. The road photos that were recorded are separated into two sections: (1) the target lane region that has to be identified, and (2) the background on either side of the lane and the sky region above. As illustrated in Figure 3.5, the non-target zone is intended to be removed by extracting the region of interest.



(a) Original image



(b) Region of interest

Figure 3.5 Region of interest extraction

As can be seen in Figure 3.5(b), the desired target area is obtained by removing invalid

information from the background and sky on both sides. This processing reduces the amount of information in the algorithm and makes subsequent processing simpler while improving detection efficiency.

3.2 Deep learning-based lane line fitting

Traditional lane recognition techniques need manual searches for specific lane attributes like geometry, color, gradient, etc. and primarily rely on strong a priori knowledge. These techniques are poorly resilient and ineffective, and they are vulnerable to noise like changes in brightness, moving cars, and broken surfaces. In today's era of big data, it is easy to access large amounts of image information, and only by combining the processing of large data volumes with machine vision can we stay at the forefront of the technology of our time. More and more academics are combining deep neural networks with machine vision because to the rapid advancement of neural networks.

Convolutional Neural Networks (CNN) have become a popular tool for solving machine vision tasks due to their excellent performance in machine vision tasks. In the case of lane line detection, the use of CNNs can reduce the influence of many distracting factors in complex road scenes, resulting in robust lane line prediction maps.

This section describes several important components of CNNs.

3.2.1 Convolutional layer

The essential component of a convolutional neural network is the convolutional layer. In image detection, the target features are mainly extracted from the underlying features such as greyscale and shape. Convolutional computation requires setting the size of a convolutional kernel, which is used to convolve the neighborhood of a pixel to obtain the neighborhood features, i.e., the Kernel starts from the top left corner of the image and moves horizontally and vertically in stride size until all pixels are traversed. During the

traversal process, the Kernel and the image pixels are multiplied and added together, and the sum replaces the central pixel value of the convolved region. The final convolution result is the Feature Map. The convolution is calculated separately for each color channel of the image, so if the input image is a three-channel RGB image, the Feature Map will be a three-channel feature map.

In the convolution process, the Kernel is equivalent to a filter, i.e., an image or feature map is filtered through a filter to achieve the filtering process, and the filtering result is the convolution result. In the training process of a CNN model, the initial values of the weight parameters of the convolution layer are random values, and the parameters are gradually adjusted during the iterative training of the target features, so that the parameters gradually converge and the model has the ability to detect the image features of the target.

The convolution calculation is shown in Figure 3.6. Usually, a padding operation is required to maintain the size of the input image, by filling in the edges of the image with "0" to keep the image size consistent.

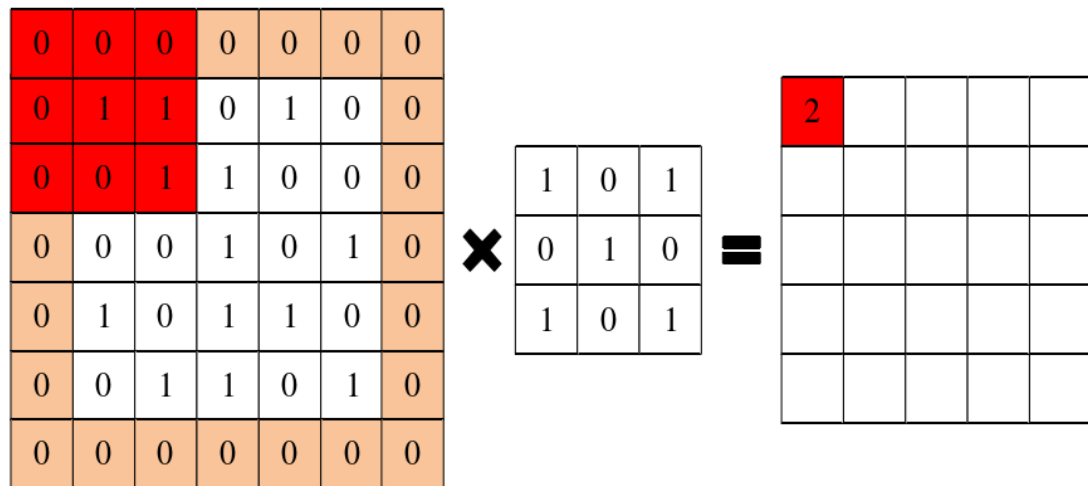


Figure 3.6 Convolution operations

When the convolution layer performs the computation, the feature maps extracted differ

depending on the Kernel. In image processing, there are many image processing methods done based on convolution operations, such as mean filtering, Gaussian filtering and edge sharpening, whose convolution kernels are shown in Figure 3.7.

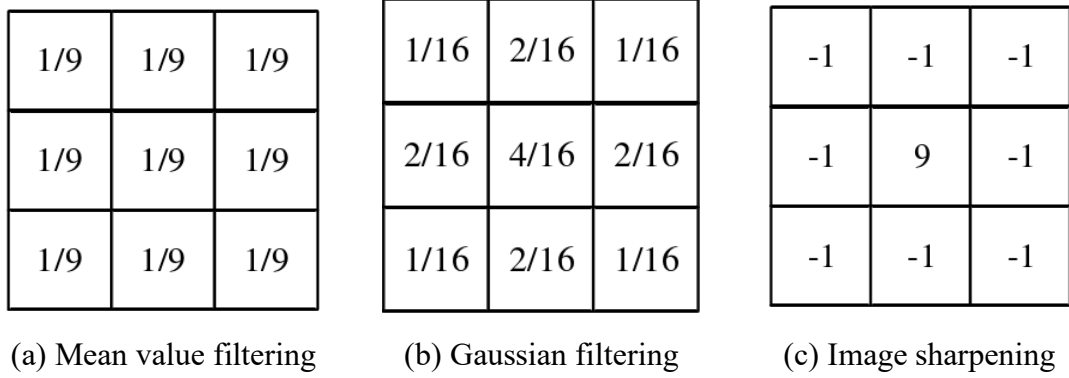


Figure 3.7 Convolution kernels under different image processing methods

3.2.2 Pooling layer

If the original amount of the input data is kept for convolutional neural network training, there will be challenges in training the model because of the enormous size of the network parameters, which reduces the algorithm's effectiveness and might result in overfitting as the training advances. Therefore, deep learning-based image semantic segmentation requires pooling to reduce the number of training parameters and the space size of the image or feature map during feature extraction. Usually, a pooling layer is added after the convolutional layer in order to reduce the number of parameters, maintain image feature invariance and increase the convolutional kernel field of perception to prevent overfitting, while preserving the main features and making the feature information extracted from the convolutional layer not severely lost.

The pooling layer is used without the need to set the learning parameters like a convolutional layer, it only needs to set the Kernel, Stride and the pooling method. The pooling methods are usually mean pooling and maximum pooling, but in the case of lane line detection, i.e., the corresponding target detection, the final result is a binary map.

Therefore, maximum pooling is used here to better perform the task of lane line detection, and the maximum pooling is calculated as shown in Figure 3.8.

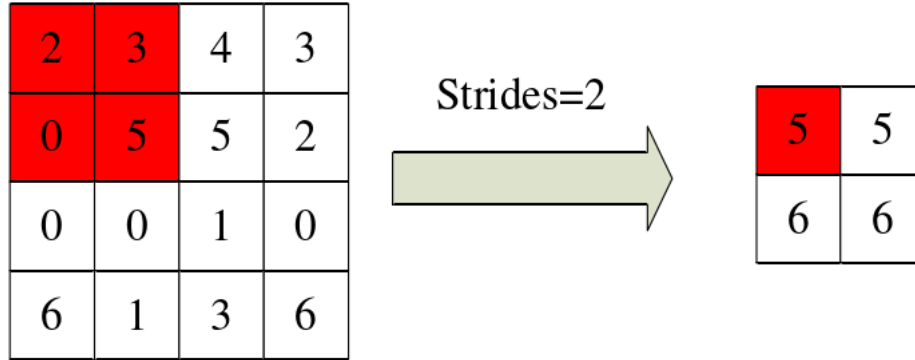


Figure 3.8 Maximum value pooling

3.2.3 Activation layer

In a neural network, a neuron's input determines its output in the upper layer, resulting in a linear connection between the input and output of each neuron node's upper and lower levels. The linearity of this relationship may be easily checked during neural network training. An activation function is therefore necessary in order to create a neural network model that can learn complicated input (such as images, audio, and video).

The linear connection between the input and output in a convolutional neural network limits the learning approximation capabilities of the neural network model without the inclusion of an activation function, even with a large number of hidden layers. Only with the addition of an activation function can a CNN model have the ability to learn to approximate complex target image features in a non-linear way. The three commonly used activation functions are described below.

(1) The Sigmoid function

The output of the Sigmoid function ranges from 0 to 1, regardless of the input. The Sigmoid function is a monotonically continuous function, differentiable everywhere and

easy to derive derivatives from, nonetheless, the Sigmoid function's left and right near-saturation zones often causes the gradient to vanish, and the inclusion of exponential operations in network learning and the fact that the output is not centered on 0 leads to slower convergence.

The expression for the Sigmoid function is:

$$f(x) = \frac{1}{(1 + e^{-x})} \quad \text{Formula (3.5)}$$

The image of the Sigmoid function is shown in Figure 3.9.

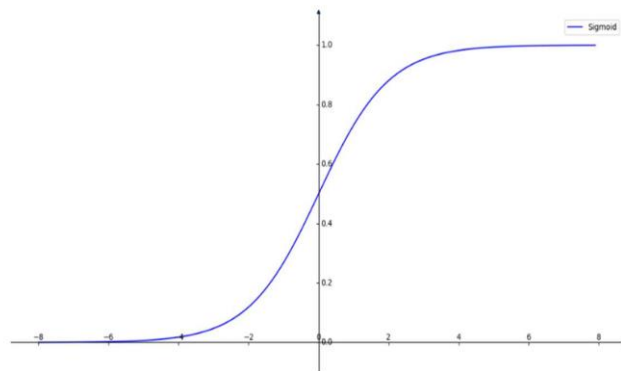


Figure 3.9 The Sigmoid function image

(2) The Tanh function

The Tanh function has an output between -1 and 1, regardless of the input. It is also a monotonically continuous function and is differentiable everywhere. The left-hand side of the function is negative and the right-hand side is positive. In network learning, the Tanh function can be used to have faster convergence because the slope is greater in its linear region around 0. However, the Tanh function has a larger range of near-saturation zones on the left and right than the Sigmoid function, and does not solve the problem of gradient disappearance.

The Tanh function expression is:

$$f(x) = \frac{e^x - e^{-x}}{(e^x + e^{-x})}$$

Formula (3.6)

The image of the Tanh function is shown in Figure 3.10.

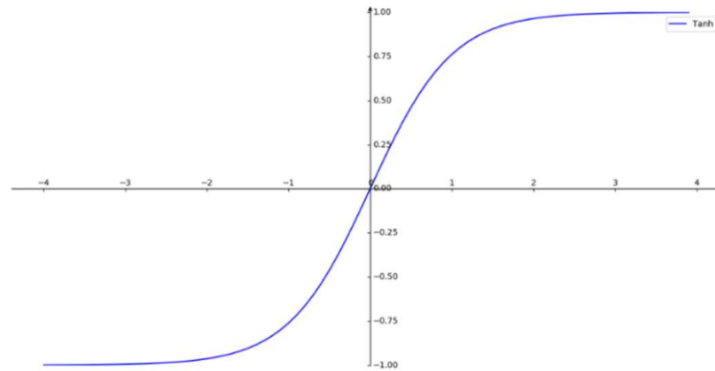


Figure 3.10 The Tanh function image

(3) The Relu Function

The left-hand side of the Relu function is suppressed, with all its values being 0, while the right-hand side is an increasing region. Also, for the ReLU function, its value does not saturate when $x > 0$, solving the problem of gradient disappearance. Therefore, in convolutional neural networks, the ReLU function is often used to increase the network's capacity for feature extraction.

The Relu function expression is:

$$f(x) = \max(0, x)$$

Formula (3.7)

The image of the Relu function is shown in Figure 3.11The image of the Relu function is shown in Figure 3.11:

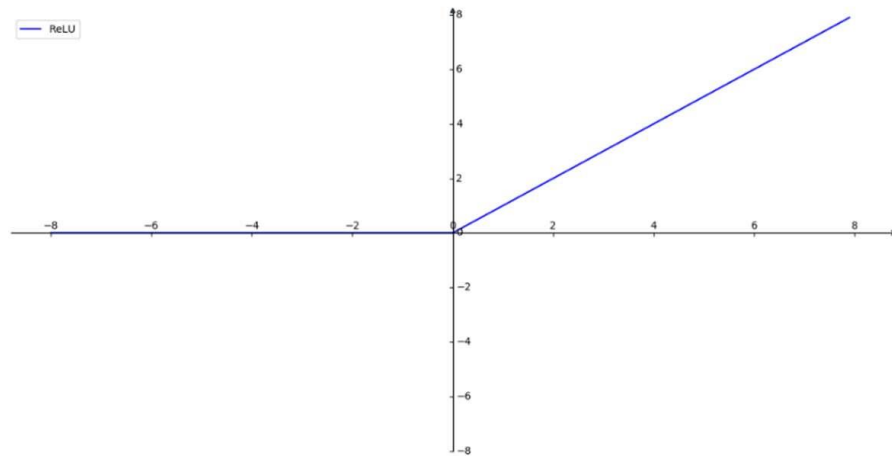


Figure 3.11 The Relu function image

3.2.4 Fully connected layer

Each neuron is fully connected in the Fully connected layers, which are shown in Figure 3.12.

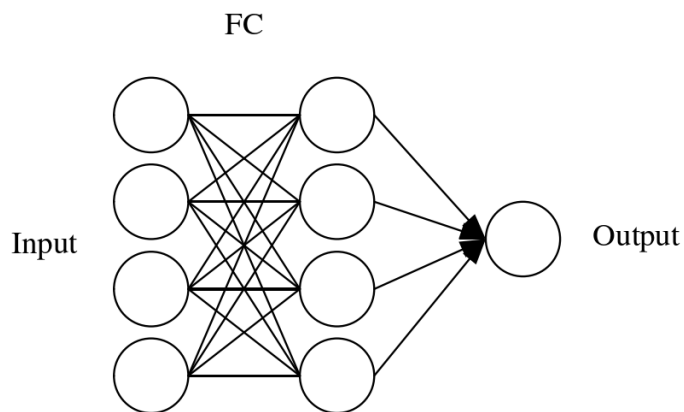


Figure 3.12 Diagram of the fully connected layer

Typically, a feature extraction element plus a feature classification part makes up CNN training of target features. The feature extraction part mainly consists of convolutional, activation and pooling layers, while the feature classification part is a fully-connected

layer that classifies the labeled features of the extracted targets.

3.2.4 Deconvolution layer

The deconvolution method is proposed in Literature [34], using a deconvolution network to visualize the extracted features by inverse pooling, inverse activation and inverse convolution of the extracted feature map to acquire a feature map that is compatible with the input image's resolution. In a neural network model, the presence of pooling layers leads to smaller and smaller feature map sizes. Therefore, it is often essential to map the low-resolution feature map to a high-resolution feature map using deconvolution layer in order to decrease the feature map to the original size of the input picture and accomplish end-to-end detection. The deconvolution is equivalent to the inverse of convolution, where the deconvolution operation only maximizes the recovery of the input detail features from the convolved feature values, and does not allow for full restoration.

The steps for deconvolution are as follows.

- (1) Invert the convolution kernel up and down.
- (2) Complement 0 according to the input dimension and step size.
- (3) Convolution with the inverted kernel after the 0's are filled.

3.2.5 Several classical neural network models

Given that lane lines have different characteristics and there is a great demand for real-time, the design of the neural network structure needs to take into account not only the detection accuracy but also the computational complexity. To assure the precision of lane recognition, it would be detrimental to sacrifice real-time speed. Therefore, a deep learning network topology that is appropriate for lane line identification must be designed.

Lane line detection based on deep learning can be classified as an image segmentation task. Image segmentation, as opposed to target recognition and image classification,

requires the detected lane line image features to be presented in an intuitive form. The image segmentation task can be considered using a semantic segmentation approach, where the semantic segmentation model is able to take the extracted feature map and achieve a classification for each pixel, so that each pixel in the image can be classified with a corresponding label. And since the lane line image segmentation task is a binary classification task, i.e., the lane line image is detected by the semantic segmentation model, each pixel will be detected as a background or lane line label to obtain a binarized image for lane line detection, and the binarized image is undoubtedly a very intuitive representation of the image features of the lane line.

Currently, some of the more mature and popular neural network models for semantic segmentation are FCN [35], SegNet [36], ResNet [37] and so on. While possessing high accuracy in these networks, all of them have the disadvantage of large network models and high computational effort. Lane line detection needs to consider real-time, so these models cannot be directly applied in lane line detection. In this study, we develop the following traditional semantic segmentation models as the foundation for our deep learning-based lane line semantic segmentation network model.

(1) FCN

The proposal of FCN opened up a path for semantic segmentation and laid a solid foundation for subsequent research on semantic segmentation. Before FCN, most convolutional neural network structures (e.g., VGG16, VGG19, etc.) used fully-connected layers to classify images, and the detection results were only the category probabilities of the images, which could not visualize the image features extracted by the neural network model. The FCN model is based on the classical CNN model VGG19, which replaces all the computationally simple convolutional layers with the more time-consuming fully-connected layers.

Due to the existence of pooling layers, which are unable to intuitively represent the picture

features, the initial feature map for the FCN during training is created following feature extraction with a low resolution. In order to achieve pixel-level classification, the initial feature map is reduced to the original input image resolution through the deconvolution layer, so that each pixel point has a corresponding label to achieve the semantic segmentation of the image. the FCN network model structure, as shown in Figure 3.13.

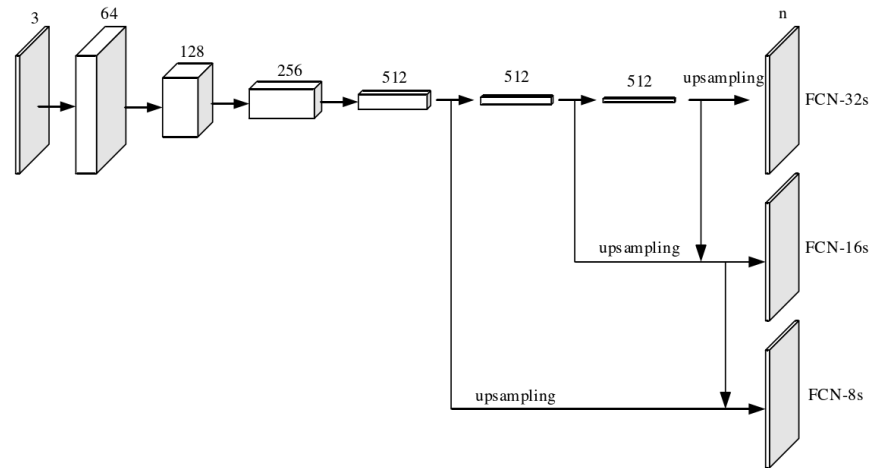


Figure 3.13 FCN structure

(2) SegNet

SegNet, a deep network structure based on the structure of the FCN, which may be separated into two primary components: the encoder and the decoder, is proposed for semantic segmentation of pictures for autonomous driving tasks. The convolutional neural network's encoder, which extracts features, and decoder, which reduces features, work together to perform semantic segmentation by upsampling the original feature map layer by layer in order to recover the input picture resolution.

Figure 3.14 displays a schematic illustration of the SegNet network concept.

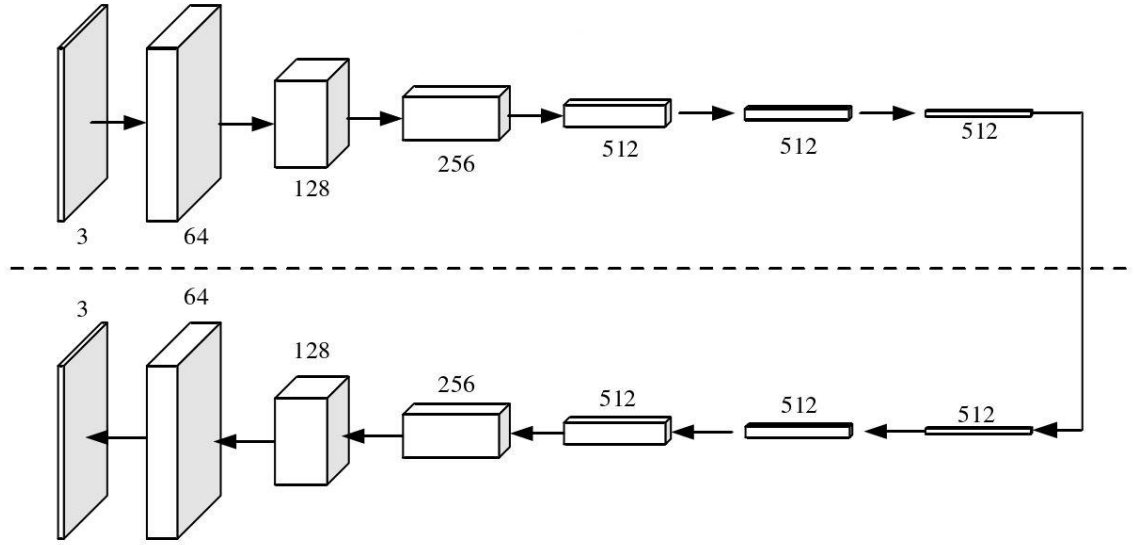


Figure 3.14 SegNet structure

(3) ResNet

Generally speaking, the deeper the network layer, the richer and more accurate the feature information detected. However, according to experimental evidence, the accuracy of the training data decreases as the network is deepened. This is not due to overfitting, which is a hallmark of the network, but rather to the high accuracy of the training and the low accuracy of the test set. However, the current problem is a drop in training accuracy, which, as can be seen through the literature [38], is due to the gradient explosion and gradient disappearance problems that occur with optimizers as the layers of the deep learning network deepen, leading to a drop in training accuracy and making it difficult to train deeper network structures.

There is a simple solution to this phenomenon - data normalization. By normalizing the data in the network, the network model is able to avoid this problem to a certain extent. However, this method only works for networks with a few dozen layers. Therefore, in order to make deeper networks with good training detection, Kai-Ming He et al. [39] proposed an idea: assuming that the existing model has reached saturation, adding a constant mapping layer after the model should not increase the error further and design a ResNet network model based on this. The structure of the ResNet network model is shown

in Figure 3.15.

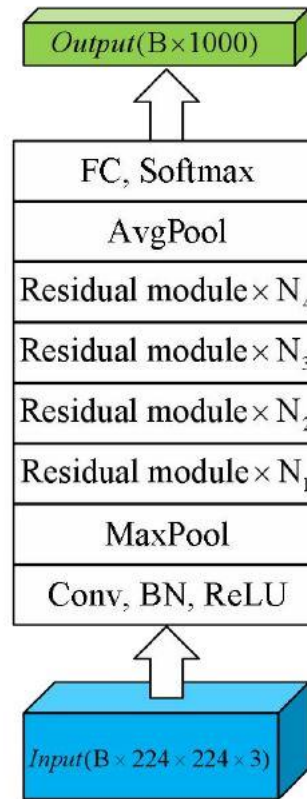


Figure 3.15 ResNet structure

Chapter 4: Results and discussion

4.1 Detection method based on the Hough transform model

The thesis employs a straight-line detection based on the Hough transform method, which is an effective algorithm for straight line recognition, for situations when the lane line type is straight and the road circumstances are typical. By translating the line's coordinates from the Cartesian coordinate system to another coordinate system where the parameter coordinates are retrieved, the detection method calculates the position of a straight line.

4.1.1 Principle of the Hough linear detection algorithm

Straight line detection is accomplished using the Hough transform technique. Its main goal is to use the pairwise property of points and lines to convert a specific curve from an image coordinate system to a polar coordinate system. The curve parameter problem in picture space is transformed into a peak problem using the voting statistics space by the fact that each curve's parameters correspond to a point in the parameter space. As seen in Figure 4.1, the various points on a particular line in the Cartesian coordinate system are labelled in the x-y coordinate system.

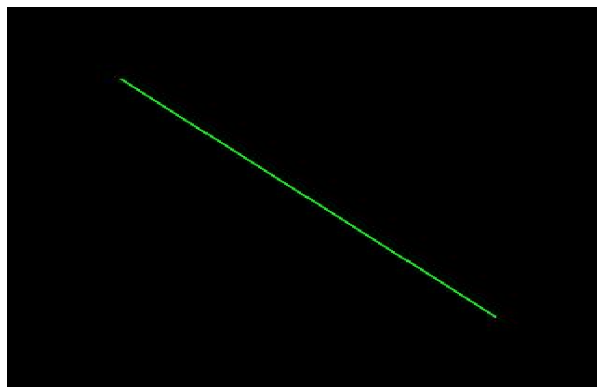


Figure 4.1 Straight line in the x-y coordinate system

According to the equation of the line as in Formula (4.1).

$$y = kx + b$$

Formula (4.1)

By converting the coordinates of the points on the line from x-y coordinates to Cartesian

k-b coordinates, the points in the x-y coordinate system are transformed into a line in the new coordinate system, and the equation of the line becomes as in Formula (4.2).

$$b = -xk + y \quad \text{Formula (4.2)}$$

The straight-line parameters are also changed from k, b to x, y. Mapping the points in the diagram to the new coordinate system results in an infinite number of straight lines, as shown in Figure 4.2.

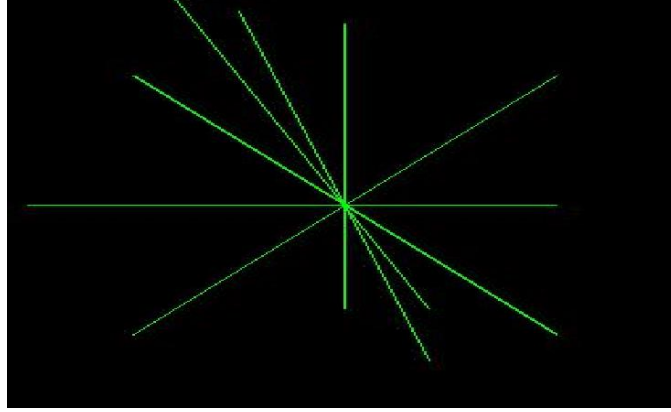


Figure 4.2 Cartesian coordinates k-b straight line mapping graph

The straight-line identification issue may be converted into a statistical problem of grouping coordinates in a parametric coordinate system based on this theory.

4.1.2 Analysis of Hough transform model detection results

The Hough straight line detection algorithm is similar to the Hough detection algorithm in that the straight-line detection is transformed into a coordinate mapping followed by a problem of clustering discrete points, with the coordinates of the clustered points corresponding to the parameters of the straight line, and for straight lanes, excellent results can be obtained using this method. The specific steps are:

(1) After the image is captured with the on-board camera, it is firstly greyed out by the `cv2.cvtColor(image,cv2.COLOR_RGB2GRAY)` function. The purpose of greying out is to obtain a feature map containing all the lane target information to reduce the number of subsequent calculations, as shown in Figure 4.3.



(a) The original image



(b) The grayscale image

Figure 4.3 Grayscale processing

(2) The grey-scale map is then filtered to ensure that the noise is removed without losing the original useful information. The thesis uses Gaussian filtering, which is a Gaussian filtering algorithm with a good noise removal effect. Gaussian filtering is used to suppress normally distributed noise by convolving a Gaussian kernel with the original image. In this thesis, Gaussian filtering is performed by the `cv2.GaussianBlur(gray,(kernel_size, kernel_size), 0)` function, where `gray` denotes the greyscale map and `kernel_size` denotes the Gaussian kernel, set to 5×5 . The recognition results are shown in Figure 4.4.



(a) The grayscale image



(b) Gaussian filtering

Figure 4.4 Filter processing

(3) Canny thresholding is then performed, as Canny is a two-threshold processing mechanism that requires two thresholds, a low threshold T_1 and a high threshold T_2 , to be set. The paper uses the `cv2.Canny(np.uint8(blur_gray), low_threshold, high_threshold)` function for edge extraction of lane line features, the two thresholds `low_threshold`,

high_threshold are 50, 150 respectively. threshold is 50 and 150 respectively, and the recognition results are shown in Figure 4.5.

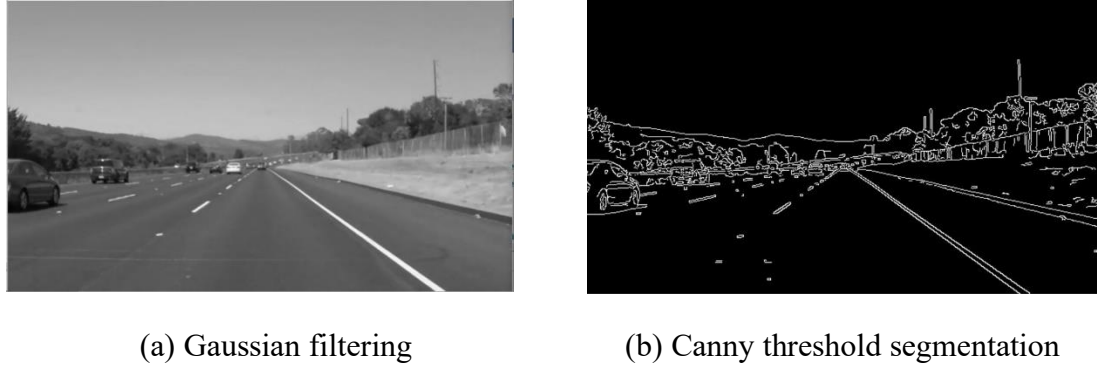


Figure 4.5 Threshold segmentation processing

Following the region of interest's extraction, the thesis's region of interest is selected primarily using the image's greyscale gradient information. The lane lines in the image have a gradient that is primarily horizontal. The thesis classifies the outcomes by region of interest (img, vertices), as shown in Figure 4.6, and the redundant image information is primarily concentrated below the sky area above and the bottom near automobile area captured by the on-board camera.

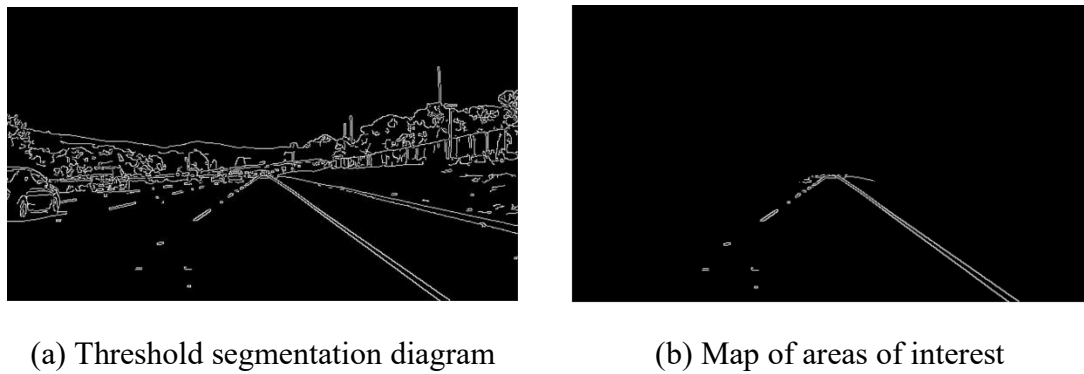
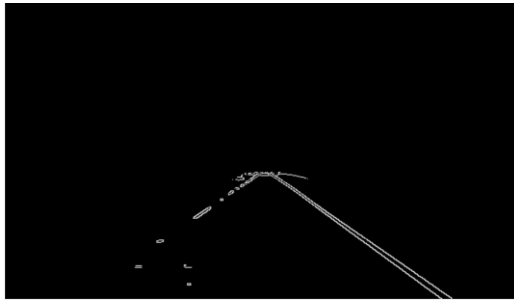


Figure 4.6 Area of interest extraction

As can be seen in Figure 4.6, the ROI segmentation method used in the thesis not only filters out the sky area above the road, but also preserves the redundant map of the near-vehicle area, allowing the region of interest in the lane line area to be detected to be

accurately obtained.

(5) The final Hough line detection and recognition was performed using the `hough_lines(img, rho, theta, threshold, min_line_len, max_line_gap)` function, where `img` is the original image, and the distance resolution and radian angle resolution of the Hough grid were set to 2, $\pi/180$ respectively. The final purpose of the straight-line detection algorithm is to count the spatial peaks, so it is necessary to set the minimum voting value, above which it is considered whether the identified straight line, the thesis is set to 15. Finally, the minimum number of pixels to form a straight line and the maximum pixel spacing between connectable line segments are set to 20, the final recognition results are shown in Figure 4.7.



(a) Threshold segmentation diagram



(b) Results graph

Figure 4.7 detection result

4.2 Detection method based on polynomial fitting of sliding windows

Although lane lines are predominantly straight lines, curves still exist in many road conditions and therefore curve detection is a critical part of lane line detection. For curve detection, due to its high complexity, a simplified curve algorithm is required to achieve the lane line detection requirements. A mathematical model for curved lane lines is proposed for the problem that the Hough transform is good for fitting straight lines but poor for curves. The original image is greyed out, filtered and thresholded for segmentation in the same way as in Section 4.1.

4.2.1 Perspective transformation

For better lane line detection, a top view image of the road surface is needed in order to obtain an approximately parallel lane line for further processing. This is because the image captured by the in-vehicle camera is the image information of the direction of travel directly in front of the vehicle. The paper uses the camera imaging principle to map the pixel points on the original image to the top view image using an inverse perspective transformation method.

It is known from image imaging principles that not all pixel points on the original map can be mapped into the overhead view. In theory, only the points below the vanishing point can be shown in the top view of the road. Based on the location of the previously detected vanishing point, the known camera parameters are used to determine the converted area, i.e., a part of the area of interest, to obtain the converted top view image and then to carry out the lane line detection and recognition. The conversion formula for the inverse perspective transformation is derived based on the coordinate system. So, the recognition needs to be converted to a two-dimensional spatial projection, i.e., the spatial coordinates (x,y,z) are converted to (l,m,n), as in Formula (4.3).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} l \\ m \\ n \end{bmatrix} \quad \text{Formula (4.3)}$$

Where $\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ is the perspective transformation matrix, $\begin{bmatrix} l \\ m \\ n \end{bmatrix}$ is the source

target point, $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ is the target point. The transformation matrix and the inverse

transformation matrix are obtained through the source target points and the target points.

The transformed image is obtained by inputting the image obtained after threshold segmentation and the transform matrix, as shown in Figure 4.8.



Figure 4.8 Perspective view

4.2.2 Locating the lane line basepoint

Firstly, the left and right lane lines need to be identified, i.e., the base point of the lane lines needs to be determined. As the pixels of the lane lines are concentrated somewhere on the horizontal axis, the image is divided into two parts, the thesis uses the pixel distribution of the bottom half of the perspective view as a basis, the peak of the pixel distribution on the horizontal axis is most likely the base point of the left and right lane lines. The location of the lane line pixel points was determined by first intercepting the bottom half of the image and adding the pixels by column, by plotting a histogram from which the peak of the sum of the pixels in each column can be seen. The left and right lane lines are spread at pixel peaks at 150 and 400, which represent the left and right peaks, respectively, as shown in Figure 4.9.

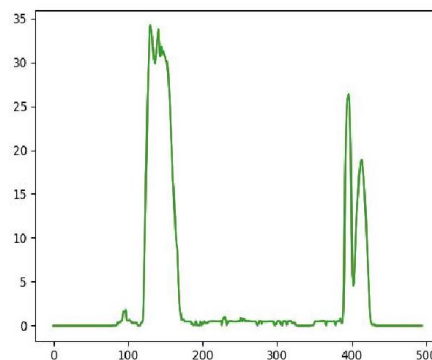


Figure 4.9 Histogram

4.2.3 Sliding window traversal and detection

After locating the base point through the perspective view, the histogram will obtain the left and right peaks so that the two peaks are used as the starting point for the left and right lane lines. To acquire the pixel indexes of the left and right lane lines, traversal and detection are done simultaneously using sliding windows. The thesis chooses the number of sliding windows to be 9, the width to be 100 and the condition for redrawing the sliding windows: the minimum number of pixels to be 50. variables and lists are also set to record the starting position of their sliding windows and the indexes of the left and right pixels.

The specific steps are to draw the sliding window by iterating through the coordinates of all non-zero pixels in the single channel image and saving them in a list. When the program iterates through the loop, the starting point of the left and right sliding window positions and the pixel indexes of the left and right lane lines are traversed and recorded, and then the number of sliding windows is started to be iterated through. The number of valid pixels in the window is set to determine if the number of valid pixels satisfies the condition for redrawing the sliding window, while the average of the horizontal coordinates of the valid pixels is used as the base coordinates for the next window, and the iterative detection is done in turn until the number of cycles is satisfied.

A second order polynomial was used to fit the left and right lane lines and plot the matching left and right lane lines using the position coordinates of the left and right lane line pixels that were acquired through sliding window fitting. Figure 4.10 depicts the outcomes of the polynomial fit.

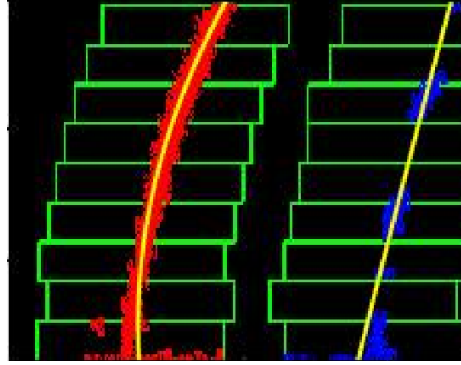


Figure 4.10 Fitting results

The drawn binary aerial view of the left and right lane lines is then transformed into the original image space using the inverse transformation matrix, and the result is combined with the original image. This process is completed by obtaining the inverse transformation matrix through the perspective transformation. Through mathematical relationships, the radius of curvature and the center offset location are also determined, and these are labelled on the graph. Figure 4.11 displays the final lane line recognition findings.



(a) Original image



(b) Detection results

Figure 4.11 Detection results

4.3 Deep learning-based lane line detection method

4.3.1 The dataset

The lane line dataset used in the thesis is TuSimple, which contains 3626 training images and 2782 test images, as well as various lane line images for different scenarios. test set. The raw_file is the path of the 20th frame of each data segment, the lanes and h_samples

are the specific annotations of the data, the $h_samples$ are the vertical coordinates and the lanes are the horizontal coordinates of each lane. The above data indicates that there are 4 lane lines and the first pixel of the first lane line is (632,240). After processing, the required annotation file was obtained and the image obtained after processing is shown in Figure 4.12.

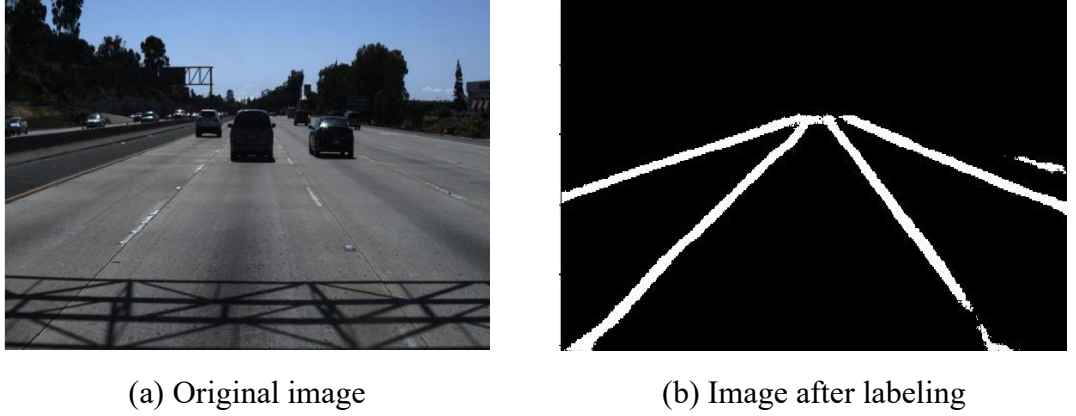


Figure 4.12 Dataset annotation

4.3.2 Overall network structure

The neural network model used in this paper is the Stacked Hourglass Network (SHN). The overall flow of the network is shown in Figure 4.13. The input image is passed through the image adjustment layer, pre-processed to compress the input size to 512×256 , and then passed to the feature extraction layer. In this study, the input data size of 512×256 is compressed to 64×32 by the convolutional layer and the maximum pooling layer, saving memory and computation time, and then the hourglass module is used to extract features from the input image. Next, the overall structure of the hourglass module is described in detail in this paper.

Stacked Hourglass networks are modular networks where the hourglass modules are essentially made up of a number of residual modules, which can be stacked into different hourglass modules depending on the number of residual modules. At the same time,

different numbers of hourglass modules can be selected for training according to the task requirements. As the main objective of this paper is to design a network model for lane line detection, it is necessary to take into account both the accuracy of detection and the real-time performance of detection, so two hourglass modules are selected to be stacked as the backbone network for training.

(1) Hourglass module

The feature extraction layer was selected as a stacked hourglass network, which achieved excellent performance in key point detection. The overall network structure contains two identical hourglass blocks for feature extraction. The overall network structure of the hourglass module is shown in Figure 4.13.

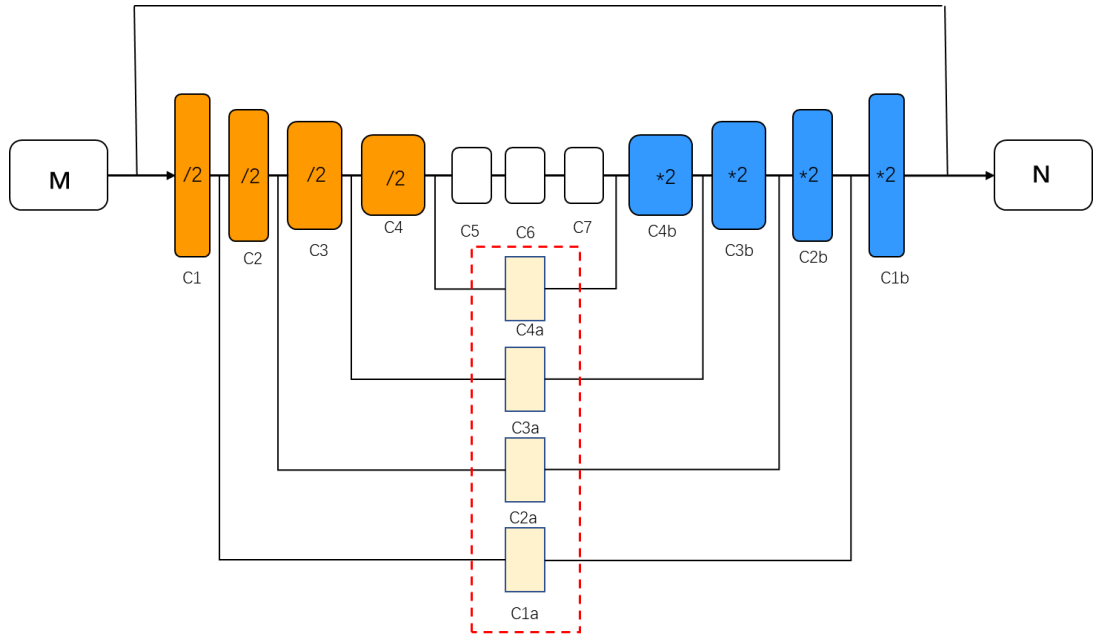


Figure 4.13 Hourglass module structure

The Hourglass module is the core component of the Stacked Hourglass network. It is shown in Figure 4.13. In the figure, the orange and blue parts are the basic residual module of the Hourglass module; the orange part $/2$ indicates the down-sampling operation of the input image by means of maximum pooling to extract deeper feature information; the

blue part*2 indicates the reconstruction of the feature map by means of up-sampling operation. The red dashed box marked section is the structure of the improved residual module. Depending on the complexity of the problem to be solved, different numbers of hourglass modules can be selected for stacking, the more the number of hourglass modules, the more accurate the results. However, too many hourglass modules will also increase the computation of the network and affect the detection speed of the model.

To extract lane line feature information using the Hourglass module, an image is an input and after pre-processing operations on the image it is passed to the Stacked Hourglass module. The features obtained from C4 through the bottleneck module are defined as C4a and the result of upsampling C7 is defined as C7_up. The extracted features are then fused.

$$C4b=C(C7_up, C4a) \quad \text{Formula (4.3)}$$

Formula (4.3) is the fusion operation for the C7_up, C4a feature layer, i.e., the two feature maps are added in the same dimension to obtain the new feature map C4b. Then, we will fuse the different depth features in turn to obtain the shallow, medium and deep layers, as shown in Formula (4.4), (4.5) and (4.6), to enrich the final detection feature layer. Table 4.1 shows the details of each layer in the hourglass module.

$$C3b=C(C4b_up, C3a) \quad \text{Formula (4.4)}$$

$$C2b=C(C3b_up, C2a) \quad \text{Formula (4.5)}$$

$$C1b=C(C2b_up, C1a) \quad \text{Formula (4.6)}$$

	Layer	Type	Filter	Size/Stride
Encoder	Down-bottleneck	Conv	32	1/1
		Conv	32	3/2
		Conv	128	1/1
		Conv	128	3/2
	bottleneck	Conv	32	1/1
		Conv	32	3/1
		Conv	128	1/1
		Conv	128	1/1
Decoder	Up-bottleneck	Conv	32	1/1
		ConvTranspose	32	3/2
		Conv	128	1/1
		ConvTranspose	128	3/2
Output	Conv+relu+bn		64	3/1
branch	Conv+relu+bn		32	3/1
	Conv		C	1/1

Table 4.1 Hourglass module structure

(2) Encoding-decoding structure

The encoder and decoder are the two main components of the Hourglass module's symmetrical overall structure. The encoder part extracts the image features by convolution and pooling of the input image data, while the decoder part restores the precise details of the image lost during the encoding process due to downsampling operations. As shown in Figure 4.13, in the middle of the hourglass module, the first half of it is downsampled by 4 times maximum pooling to extract deeper feature information in the image. However, when detecting the lane lines, it is very easy to lose the boundary information of the lane lines through multiple downsampling operations, which affects the accuracy of the final localization and detection of the lane lines. Therefore, the feature map is reconstructed in the second half of the hourglass module by up-sampling the same

number of times as down-sampling to bring it back to the size of the original input image.

4.3.3 Data pre-processing

The quality of the input data directly determines how well the model is able to predict and generalize. Therefore, pre-processing of the input images pre-processing is necessary. The image pre-processing methods selected in this paper specifically include:

(1) Image scaling

As the images in the Tusimple dataset chosen for this project have a high resolution, they cannot be trained directly. Because the higher the resolution of the images, the more memory resources are required for training. Setting the batch size too large will easily exceed the memory and lead to the model not being trained; while setting the batch size too small will lead to the slow convergence of the model. Secondly, the higher the resolution of the input image, the larger the perceptual field is required to extract information, and the usual use of downsampling to increase the perceptual field will lead to an increase in model complexity. Therefore, all the images contained in the Tusimple dataset are resized images, i.e., the original input image is resized to 512*256 and the corresponding points are matched. By compressing the image size to save as much memory space as possible, the detection speed is accelerated.

(2) Data normalization

The process of normalizing data entails dividing up the entering information into discrete intervals, which facilitates data training and hastens the convergence of the model during training. In this study, we normalize the input photos using a straightforward scaling technique. The RGB formatted pictures in the Tusimple dataset have pixel values that are evenly spaced over the range [0,255].

(3) Data Augmentation

Data augmentation is the process of generating more identical (and equally valid) data from current data to enhance the distribution of training data. Through data augmentation,

a significant volume of training data may be supplied for the model to be trained, increasing both the resilience and accuracy of the model's learning parameters. Additionally, data augmentation may be employed to enhance the model's generalizability and prevent overfitting issues. Several widely used data augmentation techniques have been used in this study to improve the training set data. Figure 4.3 presents the findings. The first row displays the training set's original picture; the second row, from left to right, depicts the training image's shading, Gaussian noise, and flipping; and the third row, from left to right, depicts the training image's panning, rotation, and intensity adjustment.



Figure 4.14 Examples of data enhancement

4.3.4 Loss function selection

In the lane line semantic segmentation problem studied in this paper, the number of positive sample pixels is much smaller than the number of negative sample pixels, and

the category distribution is extremely unbalanced. In this case, the ability to select a suitable loss function is crucial to the network prediction accuracy. At present, the more commonly used loss functions are as follows.

(1) Cross-entropy loss function

When calculating the cross-entropy loss function, the cross-entropy loss is first calculated for each pixel point in the image, and then the average of the losses of all pixel points is calculated. This is done to achieve a pixel-by-pixel classification of the image, which is the goal of the semantic segmentation problem. Formula 4.7 provides the formula for calculating the cross-entropy loss.

$$L = \frac{1}{N} \sum_i L_i = -\frac{1}{N} \sum_i \sum_c y_c \log(p_c) \quad \text{Formula (4.7)}$$

For the binary classification problem, this is calculated as shown in Formula 4.8.

$$L = \frac{1}{N} \sum_i L_i = -\frac{1}{N} \sum_i [y_i \log(p_c) + (1 - y_i) \cdot \log(1 - p_c)] \quad \text{Formula (4.8)}$$

where y_i denotes the label value of sample point i , with positive samples being 1 and negative samples being 0; p_c denotes the probability of predicting that sample i belongs to category c , and N represents the total number of sample points.

(2) Weighted cross-entropy loss function

The segmentation of the lane line versus background problem in this chapter has an obvious drawback where negative sample loss can dominate the total loss, leading to subpar network performance. The cross-entropy loss function is relatively simple and can be applied to most semantic segmentation problems. By adjusting the weights, the weighted cross-entropy loss function thus solves the issue of imbalance in the quantity of lane line pixels vs background pixels. Its formula is.

$$L = \frac{1}{N} \sum_i L_i = -\frac{1}{N} \sum_i \sum_c w_c y_c \log(p_c) \quad \text{Formula (4.9)}$$

Where w_c represents the category weight and is calculated as

$$w_c = \frac{1}{\ln(q + p_c)} \quad \text{Formula (4.10)}$$

The weighted cross-entropy loss function is chosen for the model in this paper and the hyperparameter q is set to 1.02.

4.3.5 Training details

Before any photos were put into the network, they were all normalized using the mean (0.397,0.502,0.388) and standard deviation (0.230,0.232,0.226) of the three channels of the images. This improved training efficiency. A momentum batch gradient descent (MBGD) optimization solver was used to train the network. The parameter update formula for the small batch gradient descent method is:

$$W_{t1} = W_t - \eta_t g_{d\theta} \quad \text{Formula (4.11)}$$

where W is the model parameter, W_t is the learning rate and $g_{d\theta}$ represents the gradient at the current moment.

The optimization approach updates the parameters quickly by using the gradients of a small subset of the data each time, which is significantly quicker than the conventional gradient descent method, which updates the parameters using the gradients of all the samples. Additionally, it is more reliable than stochastic gradient descent (SGD), which chooses a training sample at random and computes the gradient at each update round. However, MBGD still suffers from a certain degree of oscillation, and the choice of `batch_size` also affects the training effect of the network, with too small a batch size causing severe oscillation and too large a batch size making the training time too long. Therefore, considering both requirements, a batch size of 20 is chosen for the training of the network.

4.3.6 Evaluation indicators

The evaluation metrics in this paper follow the original evaluation scheme of the Tusimple dataset, giving the final detection accuracy and comparing it with the accuracy of the Tusimple dataset, defined as a detected lane line and its corresponding labelled true value. Points where the distance from the detected lane line to the corresponding lane labeled true value is less than the threshold specified in the dataset are considered to be correctly detected points. The proportion of successfully recognized points to those with lane lines labelled with the real value is the accuracy. Thus, Formula 4.12 shows the final expression achieved.

$$accuracy = \sum_{clip} \frac{C_{clip}}{S_{clip}} \quad \text{Formula (4.12)}$$

where C_{clip} indicates the number of points correctly predicted by the network model for a given image, and S_{clip} indicates the number of true value points corresponding to it. False negatives and false positives are also given by the following equations.

$$FP = \frac{F_{pred}}{N_{pred}} \quad \text{Formula (4.13)}$$

$$FN = \frac{M_{pred}}{N_{gt}} \quad \text{Formula (4.14)}$$

Where F_{pred} is the number of predicted wrong lanes, N_{pred} indicates the number of predicted lanes, M_{pred} indicates the number of missing lanes and N_{gt} indicates the real lanes.

4.3.7 The result

Figure 4.15 and Figure 4.16 show the changes in accuracy and loss function of the network model during the training process, respectively.

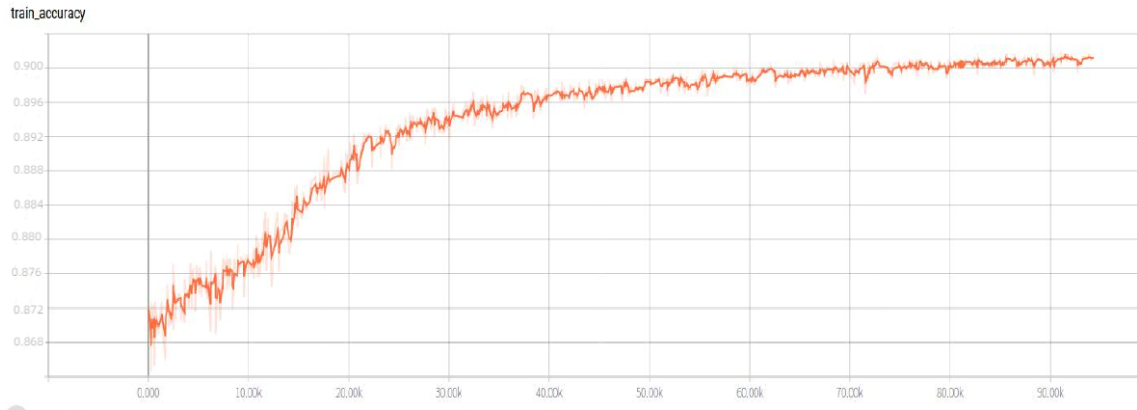


Figure 4.15 Detection accuracy during network model training

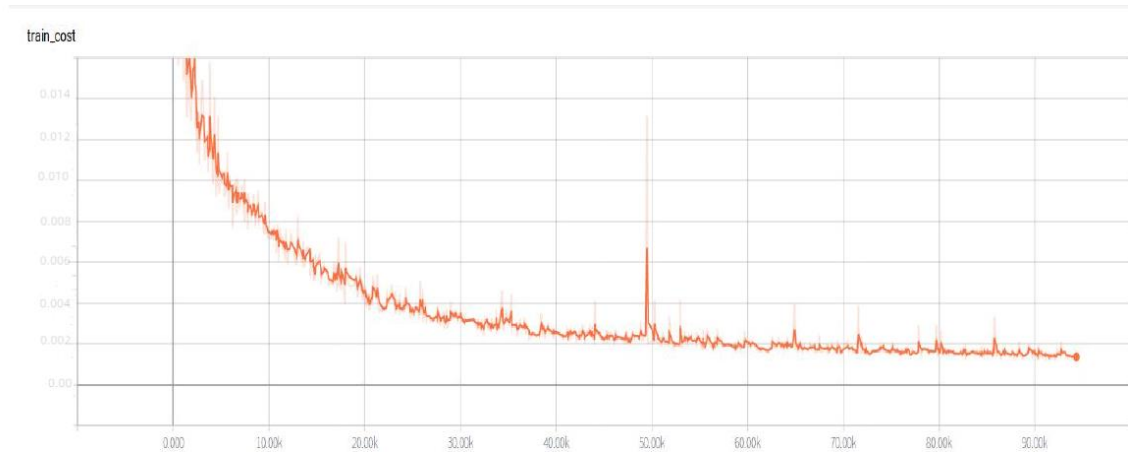


Figure 4.16 Total loss function during network model training

The detection results of our model on the TuSimple dataset are shown in Table 4.2.

Accuracy	FP	FN
90.32%	0.0935	0.0473

Table 4.2 Results

The detection outcomes of the model suggested in this study for various circumstances in the Tusimple dataset are shown in Figure 4.17. The graphic demonstrates how, even with significant lane line curvature, the network model can recognize the lane lines at the far end of the image. Despite the presence of many cars on the road and clouded lane lines,

the network model is also capable of reliably detecting lane lines.

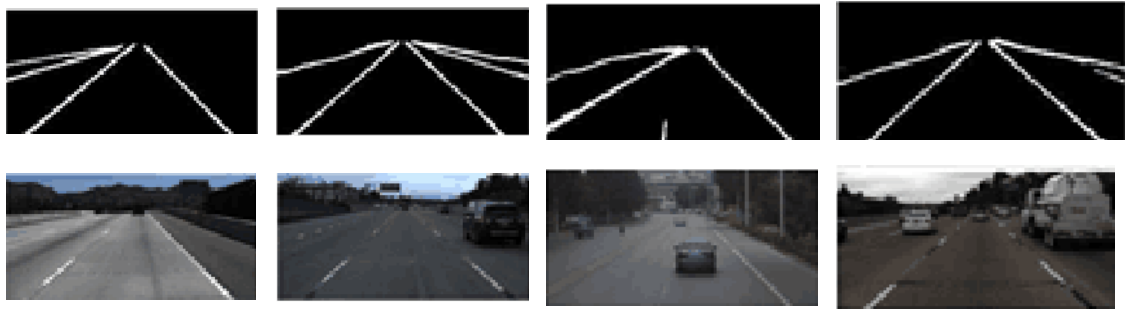


Figure 4.17 Results of lane line detection for different scenarios

Chapter 5: Conclusion

As a key technology in the vision system of smart driving cars, the thesis starts from two directions for lane line recognition. The first direction is based on traditional image processing methods to classify lane lines into two categories: straight lane line recognition in normal road conditions and curved lane line recognition in complex road conditions. For normal road conditions where the lane type is straight, the thesis uses a fit based on the Hough transform; the curved case in complex road conditions is fitted with a sliding window polynomial based on multiple thresholds. The thesis suggests a deep learning-based technique to transform the lane line identification issue into a semantic segmentation problem, with the primary work completed as follows. For scenarios with numerous lane lines and road scenarios where the lane lines are masked by other obstacles.

- (1) For selecting the best solution to the issue of lane line recognition in various scenarios, the development status of lane line detection technology is examined both domestically and internationally. This technology can be roughly divided into two directions: traditional methods and deep learning methods.
- (2) After the lane line picture is obtained by the camera, the image is edge-enhanced by pre-processing to highlight the lane line edges; then threshold segmentation is performed, and the thesis performs edge extraction by combining a variety of threshold segmentation methods with two channel information in Sobel and HSL, and the curve is fitted with a sliding window polynomial method.
- (3) To address the current problem that lane line detection is susceptible to complex environmental factors, the paper proposes the use of an improved hourglass module to extract lane line feature information. The Hourglass model is an encoder-decoder plus stub layer structure system, in which the deep-level information of the image is extracted by multiple downsampling in the encoder part, and the shallow-level information of the

image is fused with the deep-level information by using the stub layer. At the same time, two identical hourglass modules are stacked to extract features from the input image, thus providing good control of both global and local information, and helping to improve lane detection accuracy.

Chapter 6 Reflection

The lane line detection algorithm proposed in the current thesis still has some shortcomings. Reviewing the whole lane line algorithm process, the following shortcomings were found: (1) In the traditional lane line recognition part: although the lane lines are divided into two modules: straight lines and curves, the two types of lane line recognition are separated for pre-processing and threshold segmentation, without designing a pre-processing and threshold segmentation module that can be integrated together. (2) In the deep learning recognition part: the dataset used in the thesis is a small sample, which is difficult to cover comprehensively, so it cannot cope with all lane line scenarios and needs to be extended and enhanced to increase comprehensiveness and diversity. At the same time, the lane line images with severe lane wear, missing lanes and difficult lane line images under bad weather conditions need to be further enhanced and improved.

Acknowledgement

As I finish this dissertation, I am nearing the end of my study abroad experience, and I have a lot of feelings about this year of postgraduate study at the University of Wales Trinity Saint David. It was my first time in Europe and my first time living outside of China, which was a very different experience from travelling and made me very uncomfortable for a long time. I was already a difficult person to integrate into a new environment, especially as the English-only language environment made it even more overwhelming for someone like me who barely passed English, and I was also worried that there would be discrimination against Asians, as there is on the internet. However, once I had lived in Swansea for a while, all my worries disappeared. The people I met were friendly and welcoming; the restaurant staff would offer to recommend dishes when I couldn't understand the menu and show me how to eat and how to match the sauces; the teachers at the school were patient when I didn't understand something and answered my emails after class; my British classmates would drag me along to I still remember the grandfather who took me down the hill and brought me back to my flat when I was lost. One of the words I said the most in the UK was "cheers", which I only knew was used when drinking, but I didn't realize it was so common around me, and that people were so polite that they would show their appreciation when I did something as small as holding a door or giving way for someone else. It was no wonder that before I left China, the general impression of British people was that they were "gentlemen".

As I am coming to the end of another journey in my life, I just feel that I have gained so much, not only from completing my postgraduate studies in the UK and gaining new knowledge, but also from experiencing a culture first hand, which has fulfilled my main goal of studying abroad. As the old Chinese saying goes, 'hearing is believing and seeing is believing', I didn't want to learn about other countries from the internet and from what my friends told me, I wanted to come and study and live in the country for a while to learn about the local food culture and habits. This experience was more like my own Gap

Year, allowing me to get away from the hectic pace of life and study in China for a while and giving me time to think deeply about my future.

Finally, I would like to express my gratitude to my parents Hongliang Yu and Zhongli Sun for their financial and moral support after I decided to study abroad, to my girlfriend Miao Li for staying with me in a foreign relationship, for being with me when I was depressed and for being with me 10,000 kilometers across the time difference, to Gordon, Nic, David and other teachers for their help in my studies and life, and to Gordon, Nic and David for their help in my studies. I would like to thank Gordon, Nic, David and other teachers for their help in my studies and life, Zihao Liu, Chaoxing Luo, Zheng Chang, Hui He, and Linzhi Peng for living and cooking with me, and all the strangers I have met who have helped me. Finally, I would like to thank myself for having the courage to choose to make a new life and study hard for my language skills.

If I had the chance in the future, I would love to come back to the UK, to Swansea and to University of Wales Trinity Saint David. It is a place I will never forget.

Reference

- [1] Mozaffari, Al-Jarrah, O. Y., Dianati, M., Jennings, P., & Mouzakitis, A. (2019). Deep Learning-based Vehicle Behaviour Prediction For Autonomous Driving Applications: A Review. <https://doi.org/10.1109/TITS.2020.3012034>
- [2] Z. Xu, B.S. Shin. Accurate and robust line segment extraction using minimum entropy with Hough transform[J]. IEEE Trans, 2015, 24(3): 813-822
- [3] Yurtsever E, Lambert J, Carballo A, et al. A Survey of Autonomous Driving: Common Practices and Emerging Technologies[J]. arXiv: Robotics, 2019, 8(2): 3-29.
- [4] B. Fardi, G. Wanielik. Hough transformation based approach for road border detection in infrared images[C]. IEEE Intelligent Vehicles Symposium, Parma, 2004, 549-554
- [5] A. Abdelhamid, B. Azzedine. Lane detection and tracking system based on the MSER algorithm[C]. Proceedings of the 17th ACM international conference on modeling, New York, 2014, 259-266
- [6] Y. Zhen, G. Li. Robust lane marking detection using boundary-based inverse perspective mapping[C]. 2016 IEEE International Conference on AcousticsInt, Shanghai, 2016, 1921-1925
- [7] Yue L, Abdel-Aty M A, Wu Y, et al. The Practical Effectiveness of Advanced Driver Assistance Systems at Different Roadway Facilities: System Limitation, Adoption, and Usage[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 21(9): 3859-3870.
- [8] Hillel A B, Lerner R, Levi D, et al. Recent Progress in Road and Lane Detection: A Survey[J]. Machine Vision and Applications, 2014, 25(3): 727-745.
- [9] LeCun Y, Bengio Y, Hinton G. Deep Learning[J]. Nature, 2015, 521(7553): 436-444.
- [10] Kuang P, Zhu Q, Chen X. A Road Lane Recognition Algorithm Based on Color Features in AGV Vision Systems[C]. International Conference on Communications, Circuits and Systems, Guilin, China, 2006. IEEE, 2006, 1: 475-479.
- [11] Wang J, Zhang Y, Chen X, et al. A Quick Scan and Lane Recognition Algorithm Based on Positional Distribution and Edge Features[C]. International

Conference on Image Processing and Pattern Recognition in Industrial Engineering. International Society for Optics and Photonics, Xi'an, China, 2010. 7820: 78202F.

[12] Tan H, Zhou Y, Zhu Y, et al. A Novel Curve Lane Detection Based on Improved River Flow and RANSA[C]. International IEEE Conference on Intelligent Transportation Systems, Qingdao, China, 2014. IEEE, 2014: 133-138.

[13] Niu J, Lu J, Xu M, et al. Robust Lane Detection Using Two-Stage Feature Extraction with Curve Fitting[J]. Pattern Recognition, 2016, 59: 225-233.

[14] Chen P R, Lo S Y, Hang H M, et al. Efficient Road Lane Marking Detection with Deep Learning[C]. International Conference on Digital Signal Processing (DSP), Shanghai, China, 2018. IEEE, 2018: 1-5.

[15] Deng T, Yan H, Qin L, et al. How Do Drivers Allocate Their Potential Attention? Driving Fixation Prediction via Convolutional Neural Networks[J]. IEEE Transactions on Intelligent Transportation Systems, 2019, 21(5): 2146-2154.

[16] Niu J, Lu J, Xu M, et al. Robust lane detection using two-stage feature extraction with curve fitting[J]. Pattern Recognition, 2016, 59: 225-233.

[17] Bertozzi M, Broggi A. GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection[J]. IEEE transactions on image processing, 1998, 7(1): 62-81.

[18] Rasmussen C, Korah T. On-vehicle and aerial texture analysis for vision-based desert road following[C]//2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops. IEEE, 2005: 66-66.

[19] Jung S, Youn J, Sull S. Efficient lane detection based on spatiotemporal images[J]. IEEE Transactions on Intelligent Transportation Systems, 2015, 17(1): 289-295.

[20] Berriel R F, de Aguiar E, de Souza Filho V V, et al. A particle filter-based lane marker tracking approach using a cubic spline model[C]//2015 28th SIBGRAPI Conference on Graphics, Patterns and Images. IEEE, 2015: 149-156.

[21] Zhou S, Jiang Y, Xi J, et al. A novel lane detection based on geometrical model and gabor filter[C]//2010 IEEE Intelligent Vehicles Symposium. IEEE, 2010: 59-64.

[22] Kim J, Lee M. Robust lane detection based on convolutional neural network and random sample consensus[C]//International conference on neural information processing.

Springer, Cham, 2014: 454-461.

[23] Gurghian A, Koduri T, Bailur S V, et al. Deeplanes: End-to-end lane position estimation using deep neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016: 38-45.

[24] Huval B , Wang T , Tandon S , et al. An Empirical Evaluation of Deep Learning on Highway Driving[J]

[25] Lee S, Kim J, Shin Yoon J, et al. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition[C]//Proceedings of the IEEE international conference on computer vision. 2017: 1947-1955.

[26] Zhu L, Ji D, Zhu S, et al. Learning Statistical Texture for Semantic Segmentation[J]. 2021.

[27] Brabandere B D, Neven D, Gool L V. Semantic Instance Segmentation with a Discriminative Loss Function[J]. 2017.

[28] Pan X, Shi J, Luo P, et al. Spatial as deep: Spatial cnn for traffic scene understanding[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2018, 32(1).

[29] Hou Y, Ma Z, Liu C, et al. Learning lightweight lane detection cnns by self attention distillation[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 1013-1021.

[30] Neven D, De Brabandere B, Georgoulis S, et al. Towards end-to-end lane detection: an instance segmentation approach[C]//2018 IEEE intelligent vehicles symposium (IV). IEEE, 2018: 286-291.

[31] Hsu Y C, Xu Z, Kira Z, et al. Learning to cluster for proposal-free instance segmentation[C]//2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018: 1-8.

[32] Chang D, Chirakkal V, Goswami S, et al. Multi-lane detection using instance segmentation and attentive voting[C]//2019 19th International Conference on Control, Automation and Systems (ICCAS). IEEE, 2019: 1538-1542.

[33] Kaur G , Chhabra A . Curved Lane Detection using Improved Hough Transform and

CLAHE in a Multi-Channel ROI[J]. International Journal of Computer Applications, 2015, 122(13): 32-35

[34] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, 2014.

[35] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.

[36] Badrinarayanan V, Kendall A, Cipolla R. Segnet: a deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(12): 2481-2495.

[37] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs,” 2014.

[38] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs,” 2014.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep Residual Learning for Image Recognition,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.