Implementation and Comparison of Lane Line Detection Methods Based on Deep Learning and Machine Learning

QU CHENG 222012063
QIU ZEWEI 222012062
HUANG JUNRUI 221012019

Contents

Abstract

This report investigates lane line detection by replicating the Hough transform and Canny edge detection in a machine learning approach and 3D LaneNet in a deep learning approach. In the machine learning approach, we successfully detected straight lane lines by using Canny edge detection and the Hough transform and plotted the detection results in the image. However, this approach has limited effectiveness for curved lane lines and detection in complex environments. In the deep learning approach, we achieved higher accuracy by training a 3D LaneNet model with a large amount of data. However, the accuracy of the deep learning approach depends heavily on the duration of training and can produce significant errors under

disturbed conditions. It also discusses the limitations of machine learning and deep learning methods in lane line detection and points out the advantages of deep learning methods in adapting to different scenarios and real-time detection.

1. Introduction

Lane lane detection plays a key role in autonomous driving and driver assistance systems. In order to keep the car safe while driving, autonomous driving systems need to keep the car moving along the lane lines on the road, which requires accurate sensing of the lane lines. Lane detection plays an important role in autonomous driving systems, especially in Advanced Assisted Driving Systems (ADAS). Given a two-dimensional front-view image taken by an onboard camera, the aim of lane detection is to obtain the exact shape of each lane line on the road; that is, not only is it required to obtain the direction and shape of the lane line, but also to differentiate between each lane instance, and the lane detection task can be likened to an instance segmentation task. Due to the elongated shape of lane lines and the complexity of lane line changes, occlusion problems, lighting effects and semantic ambiguity in real scenarios, the need for instance-level recognition presents difficult challenges. The lane line detection algorithm is used in in-vehicle systems and requires the processing of real-time data, placing very high demands on the real-time performance of the algorithm, and how to improve the performance of the algorithm while reducing hardware requirements is another challenge for the lane line detection task. It is therefore crucial to develop lane detection methods rationally. Machine learning methods and deep learning methods are the two main directions of current lane line detection research. Machine learning methods are based on specific filters and algorithms such as edge detection filters, the Hough transform or the RANSAC algorithm. However, these methods have degraded performance in curved, occluded or high interference situations and have strict requirements for parameter settings that need to be manually adjusted to the specific driving environment. In contrast, deep learning methods automatically learn features by training neural networks and are able to handle complex road scenarios and changing lighting conditions. However, deep learning methods require large amounts of annotated data and high computational resources. The aim of this report is to compare and evaluate the performance of these two methods in lane line detection and to explore their strengths, weaknesses and limitations.

2. Related work

We summary some traditional lane detection methods, Deep learning methods and datasets as below table1, 2.

2.1. Machine Learning Methods

In the field of lane line detection, Machine Learning methods rely on specific filters and algorithms, such as edge detection filters, the Hough transform or the RANSAC algorithm. While these methods show good accuracy for straight lane line detection, performance degrades in the presence of curves, occlusions or high levels of interference. In addition, these methods have stringent requirements for parameter settings that need to be manually adjusted to the specific driving environment, making them a high workload and less robust. There are 5 methods in Machine Learning that we had researched.

- **Lane Detection Based on Hough Transformation:** This method uses edge detection algorithms (like the Canny operator) to extract all edges in the image and identifies straight lines using Hough transformation. The disadvantage of this method is that it has difficulty handling curved lane lines.
- **Lane Detection Based on LSD Lines:** LSD (Line Segment Detector) is an open image processing algorithm that effectively detects straight lines in images. However, like Hough transformation, LSD also has difficulty dealing with curved lane lines.
- **Lane Detection Based on Bird's-eye View Transformation:** This method converts the image to a bird's-eye view, making the lane lines appear straight in the image, and then performs lane detection. However, this method is seriously interfered under conditions such as occlusion.
- **Lane Detection Based on Fitting:** This method uses polynomials or other functions to fit the edge points of the detected lane lines. Although this method can handle curved roads, it is unstable and may be affected by noise and occlusion.
- **Lane Detection Based on Parallel Perspective Vanishing Point:** This method uses the fact that lane lines converge at a perspective vanishing point in the distance to assist in detecting lane lines. However, this method's operation has some specific requirements for the camera. It needs to adjust the image before transformation, and the installation of the camera and the tilt of the road itself will affect the transformation effect.

For driving environments that require real-time feedback, these methods are often unable to meet the real-time requirements. Although perspective transformation can enable multi-lane detection, its high requirements for cameras and road conditions add to the complexity of implementation.

2.2. Deep Learning Methods

Deep learning methods can be divided into four categories: segmentation-based methods, detection-based methods, parameter curve-based methods, and keypoint-based methods [3].

Deep Learning Approaches	Machine Learning Approaches
Segmentation-based methods	Hough Transform-based
Detection-based methods	LSD Line Detection
Parameter curve-based methods	Bird's Eye View Transformation
Key points-based methods	Fitting-based
-	Parallel Perspective Vanishing Point

Table 1. Overview of Deep Learning and Machine Learning Approaches for Lane Detection.

Dataset	No. of Images	Conditions/Features	Image Size
TuSimple	72k	Highway, Clear weather	1280x720
CULane	98k	Various difficult conditions	1640x590
Caltech	About 1.2k	Simple scenes	640x480
VPGNet	20k	Various lane types	-
BDD100k	120M	4 US regions, Various conditions	1280x720
ApolloScape	140k	-	3384x2710
CurveLanes	135k	Complex scenes	2650x144

Table 2. Overview of Various Datasets for Lane Detection.

- **Segmentation-based methods: SCNN [4], RESA [5], LaneNet [6]** These methods model lane line detection as a per-pixel classification problem, with each pixel classified as a lane line area or background. Although these methods are accurate, the models are usually large, processing speeds are slower, and performance may degrade in severe occlusion scenarios.

- **Detection-based methods: LineCNN [7], LaneATT [8]** These methods use a top-down approach to predict lane lines. The methods take advantage of the prior knowledge that lane lines extend from near to far in the driving view, and construct lane line instances. These methods can better handle occlusions and have better real-time performance, but the preset shape of the anchor may affect the flexibility of detection.

- **Parameter curve-based methods: PolyLaneNet [9], B´ezierLaneNet [10]** These methods use predefined parameter curves to detect the shape of the lane lines. The methods are highly efficient (e.g., PolyLaneNet can run at 115 FPS), but they may not have high accuracy.

- **Keypoint-based methods:FOLOLane [11], GANet [12]** These methods detect instances of lane lines directly and use post-processing to divide the instances. This type of method is both flexible and real-time. However,how to construct global information when handling occlusions is a problem to be considered.

3. Methodology

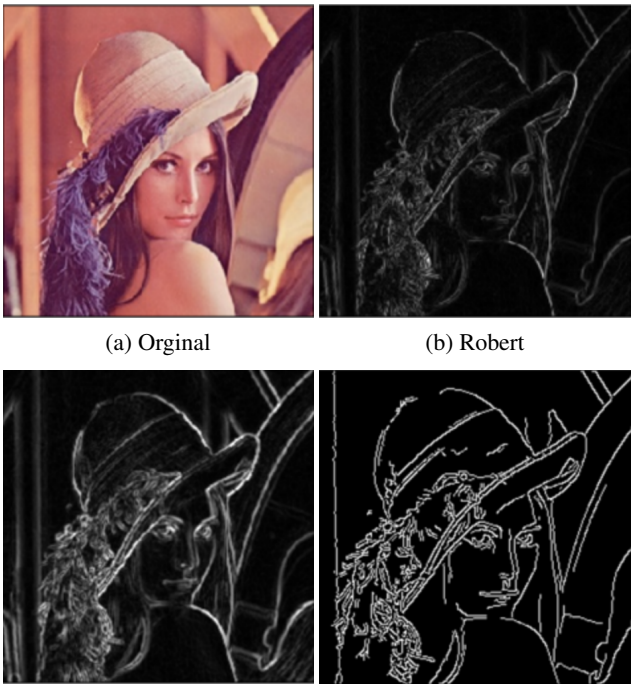
3.1. Machine Learning

The machine learning approach we reproduce the method of hough variation combined with canny operator edge detection, the detailed principles of which are as follows.

3.1.1 Edge Detection

In Machine Learning and Computer Vision, Edge Detection is an important feature extraction method that helps us to identify the outline of an object from an image. The basic principle of edge detection is to find places where the brightness of an image varies significantly, which usually correspond to the boundaries of an object, as shown in Fig.1.

According to the definition of an edge, to find a point with a significant change in brightness, it is only necessary to find a point with a large gradient, which is the partial derivative of the current pixel for the X and Y axes, so in the field of image processing it can be understood as the rate of change of the pixel's grey value. The differentiation of one of these two-dimensional functions (dealing with grey-scale



(a) Orginal (b) Robert

(c) Prewitt (d) Canny

Figure 1. Comparison of the effect of different edge detection operators.

maps) and the gradient are defined as follows:

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right).$$

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}.$$

$$\theta = \arg \left(\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x} \right).$$

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x+\epsilon, y) - f(x,y)}{\epsilon}.$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\epsilon \to 0} \frac{f(x, y+\epsilon) - f(x,y)}{\epsilon}.$$

3.1.2 Canny Edge Detector

In this experiment, we use the canny edge detection algorithm. Canny algorithm is a commonly used edge detection method with the following steps: [1]

1. **Gaussian filtering:** First, we need to smooth the image to remove noise. The Canny algorithm uses a Gaussian filter to achieve this step. The basic principle of Gaussian filtering is a weighted average of the grey values of each pixel point and its neighbouring pixel points, where the weights are determined by the Gaussian formula and the closer the pixel is to the centre the greater the weight.

2. **Calculating the gradient image and the angular image:** The aim of this step is to find the places in the image where the change in brightness is most pronounced, i.e. the edges. The Canny algorithm uses four filters (corresponding to four directions: horizontal, vertical and two diagonals) to calculate the gradient of each pixel point, i.e. the rate of change in brightness. Also, depending on the direction of the gradient, we can calculate the angular image.

3. **Non-maximum suppression:** The aim of this step is to keep only the points on the edges with the greatest luminance variation, in order to make the edges sharper. This is done by checking, for each pixel, whether it is a local maximum in the direction of the gradient and, if not, setting the brightness of that pixel to 0.

4. **Double-threshold edge joining:** The aim of this step is to determine the true edge, and the Canny algorithm sets two thresholds: a low threshold and a high threshold. For pixel points with a gradient value above the high threshold, we assume that they must belong to an edge. For pixels with a gradient value below the low threshold, we consider them to be definitely not edges. For pixels with a gradient value between the two thresholds, we further examine the eight surrounding pixels and if any of the pixels have a gradient value above the high threshold, then that pixel is also considered to belong to the edge.

3.1.3 Hough Transform

Hough Transform is a powerful algorithm for shape recognition in image processing, such as straight lines, circles and ellipses. The basic idea is to map from image space to parameter space.

Hough space is a two-dimensional plane where the horizontal axis represents the slope and the vertical axis represents the intercept of a line on an edge image, just like Fig.2. A line on an edge image is represented as $y = ax + b$. A line is represented as a point in Hough space, as the line is characterised by its slope a and intercept b. On the other hand, a point on an edge image can have an infinite number of lines crossing it, so a point is represented in Hough space as a line which takes the form $b = ax_i + y_i$.

We can use these equations to express the straight line in another form.

$$m = \frac{\text{rise}}{\text{run}} = \frac{y_2 - y_1}{x_2 - x_1}.$$

$$m = \text{slope}.$$

$$(x_1, y_1) = \text{first point}.$$

$$(x_2, y_2) = \text{second point}.$$

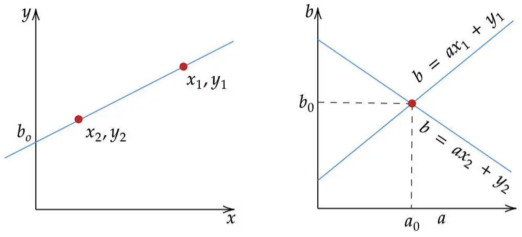


Figure 2. Hough Sapce 1.

There is a major problem with using the form $y = ax + b$ to represent a line and with using slope and intercept to represent a Hough space (see Figure 3): for vertical lines, the slope a is infinite, which means that infinite memory is needed in the computer to represent all possible values of a. To solve this problem, we use an alternative form of representing a straight line as a normal passing through the origin and perpendicular to the line. This normal takes the form $\rho = x\cos(\theta) + y\sin(\theta)$, where ρ is the length of the normal and θ is the angle between the normal and the x-axis. Thus, Hough space is now represented by ρ and θ, where the horizontal axis represents the value of θ and the vertical axis represents the value of ρ. This new linear representation eliminates the problem of infinite slope that arises when dealing with vertical lines.

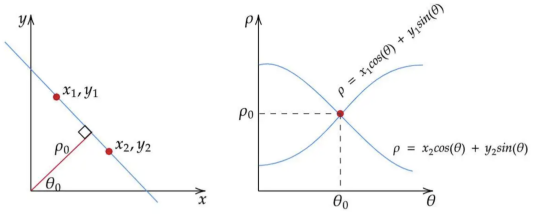


Figure 3. Hough Space 2.

The representation of edge points in Hough space is also changed. Instead of generating a straight line in Hough space, the edge point (x_i, y_i) now generates a cosine curve. In the Hough transform, straight line detection proceeds as follows (see Figure 4):

1 All edge points are mapped from the edge image to the Hough space, generating a large number of cosine curves.

2 If two or more edge points lie on the same line, then their corresponding cosine curves will cross on a particular (ρ, θ) pair. This means that the lines on the original image are represented as points in the Hough space.

3 sStraight lines are detected by finding (ρ, θ) pairs with a number of crossings greater than a certain threshold. Note that some pre-processing on the Hough space, such as neighbourhood suppression, is usually needed to remove edge maps in order to get the best results.

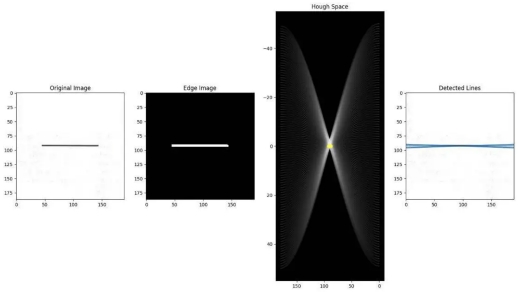


Figure 4. Hough Transform Steps.

3.2. Deep Learning Methods

3D LaneNet has made significant improvements and optimizations to previous research. Firstly, the algorithm optimized anchor representation in 3D LaneNet and used anchor prediction to obtain 3D Lane in a more reliable coordinate system. Secondly, by decoupling image segmentation and geometric encoding, the algorithm greatly reduces the demand for 3D annotation. This method introduces a new geometric guided lane anchor representation design in a new coordinate system, and applies specific geometric transformations to directly calculate the real 3D lane points from the network output. By aligning anchors with top view features, this method can be extended to unobserved scenes. It adopts a scalable two-stage framework that allows for independent learning of image segmentation subnets and geometric encoding subnets, greatly reducing the number of 3D labels required for learning.

3.2.1 Overall Pipeline

- **Image semantic segmentation:** Firstly, an image semantic segmentation network is used to predict lane line masks. This is achieved by identifying and determining whether each pixel in the image belongs to a lane line.

- **Image perspective transformation:**Then, use Inverse Perspective Mapping (IPM) to convert the mask to top view. IPM is a transformation method from the perspective view of the front camera of a vehicle to the bird's-eye view, which requires the camera's internal parameter matrix.

- **Lane line prediction:**Next, predict the lane line in the virtual aerial view. This is achieved by using the Deep Learning method to find the possible shape and location of lane lines in the given aerial image.

- **Geometric mapping:**The lane lines in the virtual aerial view are mapped back to the real world coordinates using the derived geometric relationship. This is achieved by converting the location and shape of lane lines in the virtual aerial view back to the original camera view.

3.2.2 Two stage framework

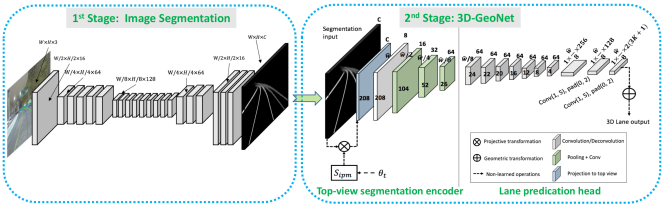


Figure 5. Two stage framework

As shown in figure 5 3D-LaneNet adopts a two-stage framework:

- **Stage1: Image semantic segmentation network**
  In this stage, a deep learning network is used to perform pixel level semantic segmentation of lane lines on the original input image, generating lane line masks.

- **Stage 2: GeoNet**
  In the second stage, based on the output of the segmentation network, the GeoNet network predicts the specific position of the lane line. GeoNet is another deep learning network that takes the output of the segmentation network as input and predicts the specific position of each lane line.

These two stages are independently trained, and the parameters of the semantic segmentation network and GeoNet are updated separately, and they do not share any parameters.

3.2.3 Loss function and training strategy

3D LaneNet uses a combined loss function, which includes segmentation loss and geometric loss. Segmentation loss is used to train semantic segmentation networks to optimize the prediction of lane line masks; Geometric loss is used to train GeoNet to optimize the prediction of lane line positions.

In terms of training strategy, as this is a two-stage framework, the semantic segmentation network and GeoNet are trained separately. When training semantic segmentation networks, we use segmentation loss, which can measure the difference between the predicted lane line mask and the actual mask. Once the segmentation network training is completed, we can use it to provide GeoNet with predicted lane line masks.

GeoNet's training uses geometric loss, which measures the difference between the predicted lane line position and the actual position. When training GeoNet, we fix the parameters of the segmentation network and only update the parameters of GeoNet. In this way, the training process of the two networks is decoupled and can be carried out independently, reducing the need for expensive 3D annotations.

It is worth noting that this two-stage training strategy is an important feature of 3D-LaneNet. Although this strategy lacks the elegance of end-to-end training, it reduces the dependence on 3D annotation, allowing the algorithm to achieve good performance even with fewer 3D annotation data. In addition, this strategy also allows us to optimize and improve the segmentation network and GeoNet separately, making them more flexible.

4. Experiment

4.1. Machine Learning

This experiment focuses on lane detection in images using a combination of the Canny edge detection and the Hough Transform.

4.1.1 Key Characteristics of Lane Detection

Prior to edge detection, it is crucial to understand the following features to improve detection rates:

- **Color:** Lane lines usually are light-colored (white or yellow), while the roads are dark (dark grey). Thus, grayscale images are preferred because the lanes can be easily segregated from the background.

- **Shape:** Lane lines are usually solid or dashed, differentiating them from other objects in the image. Edge detection algorithms like Canny can be used to find all edges/lines in an image, then further information can be used to decide which edges are likely to be lane lines.

- **Orientation:** Highway lane lines are more vertical than horizontal. Therefore, the slope of the detected lines in the image can be used to check if it could potentially be a lane.

- **ROI in the image:** In a regular road image taken by a dashcam, lane lines usually appear in the lower half of the image. Thus, the search area can be narrowed down to a region of interest to reduce noise.

4.1.2 Procedure

Our methodology consists of several steps, each of which is designed to incrementally identify the lane lines from the original images. We utilize the four key features mentioned

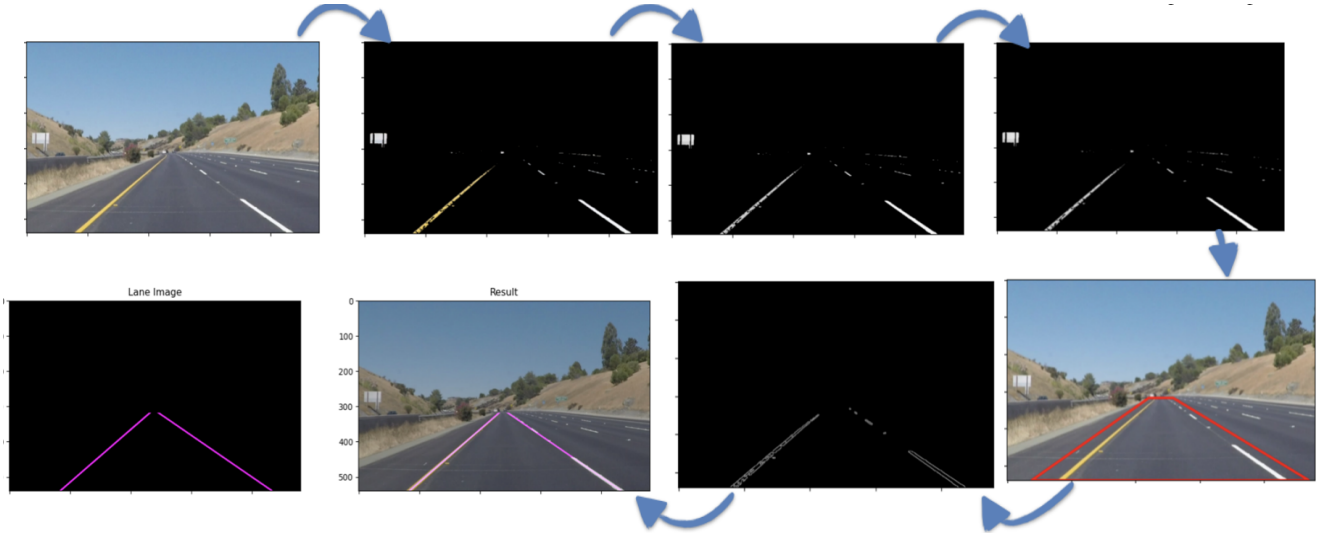


Figure 6. Machine Learning stages

earlier to facilitate the detection.The procedure for lane detection is as follows(Fig.6):

1 **Color Conversion:** The first step is converting the input color images to grayscale. This simplification allows us to more easily separate the lanes from the road as lanes are typically light-colored (white or yellow) while the road is dark-colored (black or gray). The conversion is done using the OpenCV function `cvtColor()` with the `COLOR_RGB2GRAY` argument.

2 **Gaussian Smoothing:** The grayscale images are then smoothed using a Gaussian blur with a kernel size of 5. This step reduces high-frequency noise in the image that can lead to false edge detection in the subsequent Canny operation. The kernel size of the blur, which determines the amount of smoothing, is a parameter that can be adjusted to optimize the detection performance.

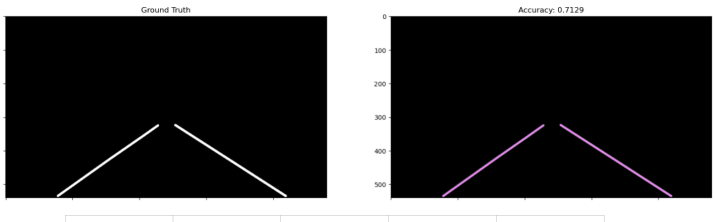
3 **Region of Interest Selection:** Given that the lane lines are expected to appear in a certain area in the image, we can ignore the areas of the image where lanes are not expected. By defining a region of interest based on the image size, we can focus our analysis and reduce the noise from other objects and irrelevant areas of the image.

4 **Canny Edge Detection:** Edges are detected in the smoothed grayscale images using the Canny Edge Detection technique. This technique identifies areas of the image where the intensity changes rapidly, which correspond to edges of objects - in our case, the lanes. We used a low threshold of 50 and a high threshold of 200 to detect strong edges as well as some weaker edges that are connected to the strong ones.

5 **Hough Transform:** The Hough Transform is applied to detect lines in the edge-detected image. The Hough Transform identifies points in the image that form a line and groups them together. The parameters for the transform can be adjusted to control the minimum length of a line, the maximum gap between segments of a line, and the number of votes needed to agree on a line.

6 **Drawing Lanes:** The final step is to draw lanes on the images based on the lines detected from the Hough Transform. We draw these lines back onto the original image to see our final result. The lines are drawn as overlays on the original image, allowing us to visually verify the accuracy of our lane detection.

4.1.3 Evaluation and Results



MSE	Precision	Recall	F1 Score	Accuracy
1.0084	0.9706	0.7128	0.8220	0.7128

Figure 7. Result and Evaluation in ML.

The experimental results(7) show that the combination of the Canny edge detection algorithm and the Hough transform can successfully detect lane lines in a given image. The detected lane lines are overlaid on the original image

for visualisation. And we evaluate them by comparing them with hand-drawn lane lines, including MSE, Precision, Recall, F1 Score, Accuracy. It can be seen that the method can obtain highprecision values in straight line detection.

The function `process_image()` in code was created to apply the aforementioned steps to each frame in a video. It takes an image and returns a copy of the image with the lanes drawn on it. The rest result images and videos are saved in the code package.

4.1.4 Problem and Thinking

While this approach is effective for detecting straight lane lines, it struggles with **curved lanes** (as in bends or turns) since the Hough Transform primarily identifies straight lines.

Figure 8. Incorrect Lines in ML.

Additionally, this approach may detect incorrect lines in **complex or busy environments** with lots of shadows, signs, or other vehicles(8. This is because the Canny edge detection algorithm identifies all the edges in the image, not just lane lines. Although the region of interest selection helps mitigate this issue, it cannot completely eliminate it. For the wrong line, we first change the ROI area, and then add one filter to remove horizontal, vertical and other lines with unusual slopes.

Moreover, this approach assumes that the camera position is fixed and the road is relatively flat, which may not be the case in real-world situations. It also does not account for changes in lighting conditions. For example, lane detection at night or in heavy rain or fog may be more challenging. So for different environment, the Machine Learning Method needs to be fine-tuned.

4.2. Deep Learning Methods

The deep learning part of the project unsing 3D GenLaneNet model training and prediction lane lines and centerlines. In the reproduction, we used the Kaggle cloud platform to train the model and retained the best model parameters based on loss. The results will be included in the submitted compressed package.

Due to the limitations of the Kaggle cloud platform, we can only train continuously for up to 10 hours. After multiple attempts, the model can train up to 50 epochs, so the effect may not be optimal. If we go through more training, the model may be better.

Figures 9 and 10 respectively show the model prediction results after training 20 epochs and the model prediction results after training 50 epochs. It is evident from the figure that the results generated by the model after training 50 epochs are closer to the real label.

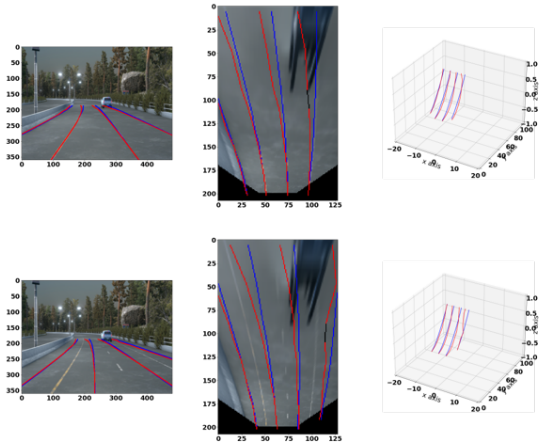


Figure 9. Training after 20 epochs

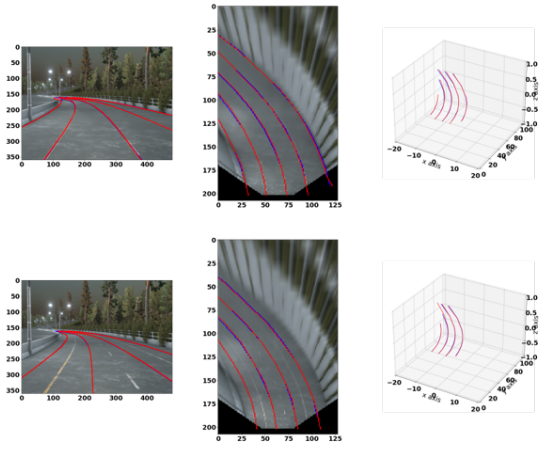


Figure 10. Training after 50 epochs

At the same time, the accuracy of predicting lane lines and lane centerline, as well as the training loss data, were also retained in the training, as shown in Figures 12 and 11.

It is obvious that deep learning models are more dependent on large amount of data training. Compared to machine learning models, deep learning models trained with a large

Figure 11. Loss

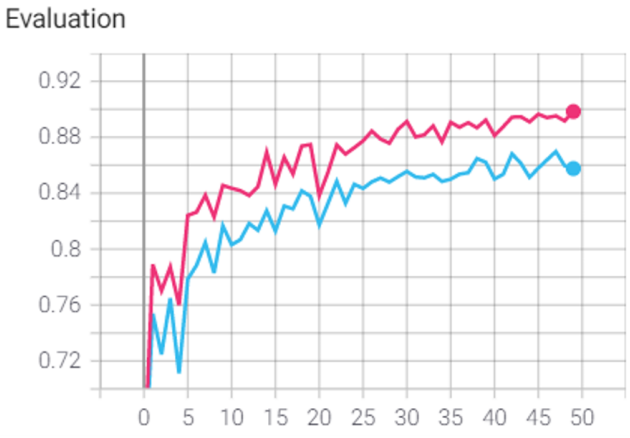


Figure 12. Accuracy

amount of data typically have higher accuracy and better performance.

5. Limitations

In our project, we have discovered several limitations in both machine learning and deep learning methods for lane detection.

For the Hough Transform method, although it can achieve high accuracy in detecting straight lane lines, it performs poorly in detecting curves. It also became evident that machine learning methods are highly environment dependent and require fine-tuning the model in different driving environments, making them unsuitable for real-time driving requirements.

On the other hand, while the deep learning method demonstrated higher accuracy after extensive training, we observed that its precision was largely dependent on the duration of training, and the model would produce significant errors under disturbed conditions.

In particular, the 3D Gen-LaneNet model used in our project has some specific limitations. The main limitation is that the algorithm has a high requirement for camera parameters, and when the camera parameters are not accurate, the detection result may deteriorate. Hence, it may not be suitable for all onboard camera systems.

6. Conclusion

In conclusion, both traditional image processing methods and deep learning methods for lane detection have their own advantages and disadvantages. The choice of method is dependent on the specific application scenario and requirements.

However, deep learning methods have demonstrated several noticeable advantages. They can adapt better to different types of roads and environments, due to their ability to learn from a large amount of data. They can effectively handle complex scenarios such as curved roads, variable lighting conditions, and obstructions. Additionally, deep learning methods typically perform better in real-time detection and have been widely adopted in recent years.

Comparatively, traditional image processing methods are less computationally intensive and can handle smaller datasets, but their performance and adaptability to different scenarios are significantly inferior to deep learning methods.

With the advancement of deep learning technology and the increase in available data, deep learning methods have been widely used in the field of lane detection and show great prospects. However, further research and development are required to overcome their current limitations, especially in terms of reliance on extensive training data and the high computational requirements of these models.

References

[1] https://aishack.in/tutorials/canny-edge-detector/ 5

[2] How to implement straight line detection with the Hough transform algorithm, https://www.infoq.cn/article/3KfCkf67yZQHuiFp0x3Q

[3] Overview of lane line detection, https://blog.csdn.net 3

[4] Pan, Xingang, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Spatial as Deep: Spatial CNN for Traffic Scene Understanding." Proceedings of the AAAI Conference on Artificial Intelligence 32, no. 1 (April 27, 2018). https://ojs.aaai.org/index.php/AAAI/article/view/12301 4

[5] Zheng, Tu, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. "RESA: Recurrent Feature-Shift Aggregator for Lane Detection." ArXiv:2008.13719 [Cs], March 25, 2021. http://arxiv.org/abs/2008.13719 4

[6] Neven, Davy, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. "Towards End-to-End Lane Detection: An Instance Segmentation Approach." In 2018 IEEE Intelligent Vehicles Symposium (IV), 286–91, 2018. https://doi.org/10.1109/IVS.2018.8500547 4

[7] Li, Xiang, Jun Li, Xiaolin Hu, and Jian Yang. "Line-CNN: End-to-End Traffic Line Detection With Line Proposal Unit." IEEE Transactions on Intelligent Transportation Systems 21, no. 1 (January 2020): 248–58. https://doi.org/10.1109/TITS.2019.2890870 4

[8] Tabelini, Lucas, Rodrigo Berriel, Thiago M. Paixão, Claudine Badue, Alberto F. De Souza, and Thiago Oliveira-Santos. "Keep Your Eyes on the Lane: Real-Time Attention-Guided Lane Detection." ArXiv:2010.12035 [Cs], November 17, 2020. http://arxiv.org/abs/2010.12035 4

[9] Tabelini, Lucas, Rodrigo Berriel, Thiago M. Paixão, Claudine Badue, Alberto F. De Souza, and Thiago Oliveira-Santos. "PolyLaneNet: Lane Estimation via Deep Polynomial Regression." ArXiv:2004.10924 [Cs], July 14, 2020. http://arxiv.org/abs/2004.10924 4

[10] Feng, Zhengyang, Shaohua Guo, Xin Tan, Ke Xu, Min Wang, and Lizhuang Ma. "Rethinking Efficient Lane Detection via Curve Modeling." ArXiv:2203.02431 [Cs], March 4, 2022. http://arxiv.org/abs/2203.02431 4

[11] Qu, Zhan, Huan Jin, Yang Zhou, Zhen Yang, and Wei Zhang. "Focus on Local: Detecting Lane Marker from Bottom Up via Key Point." arXiv, May 28, 2021. https://doi.org/10.48550/arXiv.2105.13680 4

[12] Wang, Jinsheng, Yinchao Ma, Shaofei Huang, Tianrui Hui, Fei Wang, Chen Qian, and Tianzhu Zhang. "A Keypoint-Based Global Association Network for Lane Detection." ArXiv:2204.07335 [Cs], April 15, 2022. http://arxiv.org/abs/2204.07335 4