Transport and Communications Science Journal, Vol. 75, Issue 04 (05/2024), 1659-1672

Transport and Communications Science Journal

LANE DETECTION USING HOUGH TRANSFORMATION AND YOLOv8

Nguyen Viet Bach, Pham Xuan Tung*

University of Science and Technology of Hanoi, Vietnam Academy of Science and Technology, 18 Hoang Quoc Viet Street, Cau Giay, Hanoi, Vietnam

ARTICLE INFO

TYPE: Research Article
Received: 15/12/2023
Revised: 25/04/2024
Accepted: 07/05/2024
Published online: 15/04/2024
https://doi.org/10.47869/tcsj.75.4.15
* *Corresponding author*
Email: pham-xuan.tung@usth.edu.vn;

Abstract. Autonomous vehicles necessitate the integration of advanced technologies such as computer vision and deep learning to comprehend and navigate their surroundings. A crucial yet challenging component of this integration is the accurate detection of lanes, which can be influenced by a multitude of varying lane characteristics and conditions. This research undertakes a comparative analysis of lane detection methodologies, explicitly focusing on traditional image processing techniques and Convolutional Neural Networks (CNNs). The evaluation utilized a sample of 500 images from the CULane dataset, which encompasses a diverse range of traffic scenarios. Initially, a method incorporating Gaussian blurring, Canny edge detection, and Hough line transformation was examined. Despite its efficiency, operating at 30 frames per second, this approach exhibited a high error rate (average Mean Squared Error (MSE) of 0.537), which is attributable to the loss of critical image details during the preprocessing stage. Subsequently, the performance of a fine-tuned YOLOv8 model, trained on a reformatted version of the CULane dataset was assessed. The combination of object detection and subsequent Hough transformation yielded high accuracy, demonstrating the model's ability to learn and identify relevant lane features. The deep CNNs demonstrated superior performance over classical image processing techniques in terms of lane detection accuracy, thereby underscoring their potential applicability within the realm of autonomous vehicle technology.

Keywords: Image processing, deep learning, object detection, lane detection, AI, YOLO, OpenCV

1. INTRODUCTION

The pursuit of enhancing traffic systems has led to the quest for a mechanism that can adeptly aid drivers in navigating traffic. Traditional methods have found this task challenging due to the myriad of uncertainties and variables involved, which are nearly impossible to encapsulate within a single equation. From some of the research regarding such subjects [1], one can see that the mentioned task is an intricate and complex accomplishment which require a better alternative to the more conditions-dependent conventional methods. Consequently, a novel approach that can revolutionize how individuals engage with traffic is warranted, a role that appears befitting for Artificial Intelligence (AI) technology. The nascent field of object-detection self-driving vehicles has demonstrated its potential as a formidable instrument in aiding drivers and establishing an autonomous transportation system. As many of studies utilizing AI technology in traffic systems [2], [3] have shown, this new technology has the capable to revolutionize the field of autonomous vehicles. An object detection model can equip a vehicle with the capability to assimilate an extensive array of visual information and make rapid decisions, thereby paving the way for enhanced vehicular control. Such a system can potentially eliminate human errors and provide a superior analysis of external conditions, thereby fostering a safer and more efficient environment for traffic participants.

In many countries, including Vietnam, the prevalence of self-driving cars still needs to be improved. Despite the extensive promotion of this technology in recent years, consumers harbor numerous concerns regarding the cost and safety of self-driving vehicles. These issues are exacerbated in developing nations like Vietnam, characterized by lower living standards and inadequate road conditions for the safe and effective operation of such systems. Nonetheless, continual advancements in the field could potentially render this technology accessible to consumers, significantly influencing vehicular usage.

The deployment of computer vision technology in analyzing dashcam inputs from the perspective of an automobile has attracted considerable interest in recent times. While numerous studies have employed object detection to identify various elements a driver could encounter in traffic [4], lane detection remains a formidable challenge. Owing to the distinct characteristics of lanes, such as shape, color, texture, condition, and location, many scientists still rely on the traditional image processing method [5-10]. The demanding nature of deep learning models has resulted in limited research on this topic. Among the scant research studies employing deep learning approaches for road lane detection, a handful have harnessed the power of the YOLO object detection [11] deep learning model to strike a balance between accuracy and speed. Object detection technology holds immense promise in revolutionizing the way we navigate traffic. However, as a relatively new scientific field that has only gained prominence in the past decade, further developments are essential before it can be deemed indispensable. This has kindled our interest in delving deeper into this realm and identifying effective methods to extract visual information from vehicles' dash cams using deep learning technology.

This paper scrutinizes two lane detection methods: image processing in isolation and integrating deep learning techniques with Hough transformation. The deep learning techniques specifically discuss two models, namely YOLOv8 pose estimation and YOLOv8 object detection. These Convolutional Neural Network (CNN) models are trained using the CULANE dataset [12], and each approach is evaluated based on the Mean Average Precision (MAP) score, representing accuracy and speed. With the overarching goal of developing a

system capable of processing traffic inputs, the focus is not solely on achieving high precision; ensuring a minimal processing speed is equally imperative.

2. MATERIALS AND METHOD

2.1. Dataset

All the training and evaluation are conducted on a portion of the CULane dataset [12], a large-scale, challenging data set for academic research on traffic lane detection. The CULane data set contains 133325 images worth more than 55 hours of videos captured by cameras mounted on six vehicles driven by different drivers in Beijing, with 88880 used for training, 9675 for validating, and 34680 for testing. All the images, sized 1640x590, are annotated in key points for each lane and with a maximum of 4 lanes. Key points are sampled along the y-axis and with step values of 10.

2.2. YOLOv8 models

In comparison, YOLOv8 offers more accuracy-speed optimization than previous models. As observed in Figure 1, YOLOv8 offers a great performance with both high accuracy-precision and fast processing time, making it a suitable model for various real-time detection tasks and having a wide range of applications.

In particular, two of the models used in this research are of object detection and pose estimation variants, yolov8n and yolov8n-pose respectively. Being light-weighted models, both models still offer a good performance with good mAP50-95 and high processing speed with the COCO [13] data set, making them suitable options for real-time detection tasks, similar to the one in this research.

YOLOv8 utilizes a CNN that can be divided into two main parts: the backbone and the head. The backbones of YOLOv8 are formed by a modified version of CSPDarkNet53, which has 53 convolutional layers and employs cross-stage partial connections to improve information flow between the different layers. The heads, which is responsible for predicting bounding boxes, objectness scores, and class probabilities for the objects detected in an image.

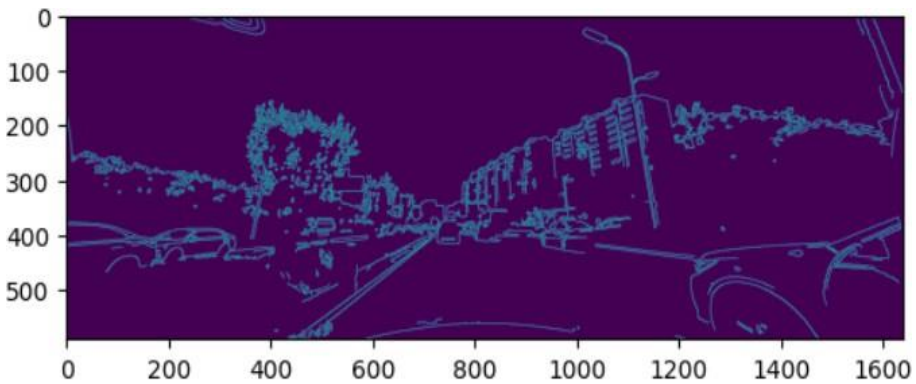
YOLOv8 utilizes a CNN that can be divided into two main parts: the backbone and the head. The backbones of YOLOv8 are formed by a modified version of CSPDarkNet53 (Fig. 4), which has 53 convolutional layers and employs cross-stage partial connections to improve information flow between the different layers.

YOLOv8 also has many features to enhance its capability: a self-attention system that focuses on different parts of the image and adjusts the importance of different features based on their relevance to the task; multi-scale object detection using a feature pyramid network is also utilized. It also features a non-maximum suppression stage, a lengthy post-processing process procedure, detecting an object's direct centre without needing a marked anchor. Hence, the number of prediction boxes is reduced. Not only is it making changes to the architecture, but YOLOv8 has also made new efforts in data augmentation to progress training results. Training samples are augmented online during training: the model randomly sees a slightly different variation of the images provided at each epoch. The model's performance can be improved by greatly enlarging the training sample's range and diversity.

2.3. Lane detection based on image processing

Due to their unique shapes, road lanes have the potential to be detected using methods of line detection, specifically in this application: Hough transforms for line detection. Images are subject to a pre-processing stage of Gaussian blurring and canny edges detection to make Hough transformation possible (Fig. 1).

(a) Original image



(b) Transformed image

Figure 1. Pre-processing stage.

After that, the image goes through line detection using Hough transform. For more accurate filtering, some of the crucial parameters required for all the positive lines can be toggled: minimum length, which is the smallest value a line measured in to be counted as positive (length measured in pixels), and maximum gap, which is the most significant number of pixels between two similar lines for them to be counted as one. The results are sets of 4 points representing the coordinates of the starting and end points, respectively.

Finally, the result is put through a filter to cross out lines that are sure not to be lanes. This is done by setting a range that the slopes of the output lines must fall into; all lines that do not qualify will be discarded. All the lines with slopes from -0.2 to 0.2 would be filtered out in this case (Fig. 2).

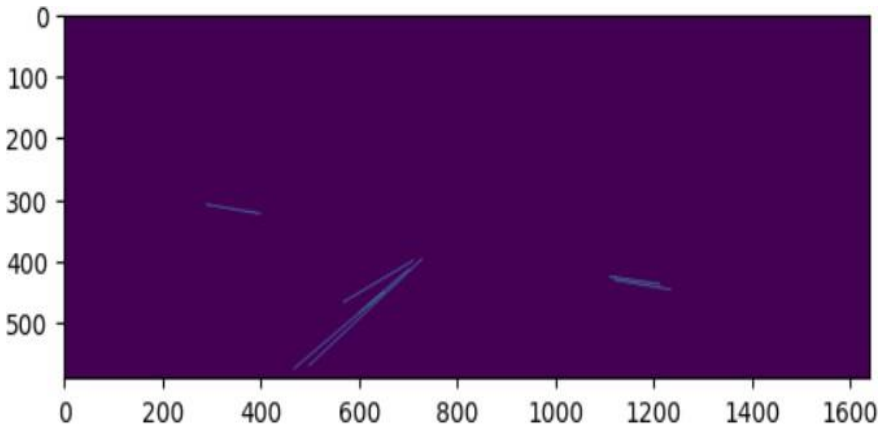


Figure 2. Lines obtained through Hough transform.

The evaluation process is conducted on 500 randomly selected images from the data set, with the data annotations of each image converted from a vast array of key points to 2 values that uniquely represent a line roughly similar to in Hough space. These two metrics are the degree of radiance with the x-axis and the distance to the line's origin. This enables all the metrics to be normalized and thus makes for a more illustrative evaluation process. Let (x1, y1) and (x2, y2) be the coordinate of two points in Cartesian space. The two parameters represent the line that crosses through both points and is formulated as follows:

$$\theta = \arctan \frac{y_1 - y_2}{x_1 - x_2} \tag{1}$$

$$h = \frac{\left| y_1 - x_1 \dfrac{y_1 - y_2}{x_1 - x_2} \right|}{\sqrt{1 + \left(\dfrac{y_1 - y_2}{x_1 - x_2} \right)^2}} \tag{2}$$

The output of the detection module is compared against the adjusted label to generate the MSE score and the total number of correct recalls. These metrics are used to examine how far the prediction deviates from reality and, therefore, give us an idea of the compatibility of this method in practical uses. The processing time of each of the images is also recorded for evaluation purposes.

2.4. Lane detection based on deep learning and image processing

This study will apply two deep-learning models in combination with image processing techniques to detect the lane (Fig. 3). They are YOLOv8 pose detection and YOLOv8 object detection [8], and their results are directly compared. The results of the deep learning model are then subjected to a process of line filtering using Hough transformation to achieve the final detections for lanes.

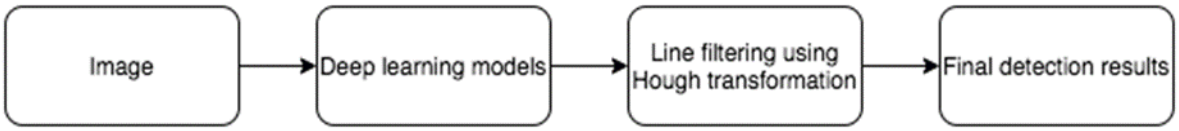


Figure 3. Diagram of the deep learning and image processing technique.

2.4.1. Pose detection-based model

Before the training is initiated, data representation must be considered to fit into the format of the deep learning model. With YOLOv8 pose detection, we can retain the labels' original key points format of the data set with little augmenting. The labels are truncated to accelerate the training process and increase the model's accuracy without much decrease in representing the lane: from 16-21 key points for each lane to 4 key points for each lane since most lanes are straight lines, as shown in Fig. 4.

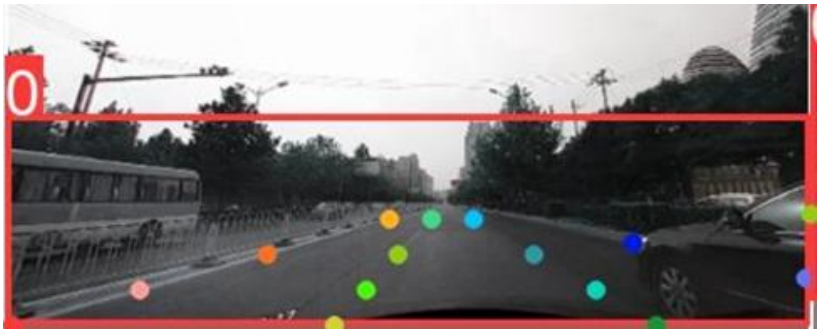


Figure 4. Data annotation for pose detection.

The model in use is pre-trained with the COCO dataset for increased effectiveness in training. The training process lasts for 50 epochs, with 17733 images used for training and 500 used for validation.

2.4.2. Object detection based

Even though using an object detection model to detect a straight lane that extends for a large proportion of the images is an unorthodox approach, it has enormous potential due to YOLOv8 detection's high accuracy and exceptional ability in object detection. The core idea of this method is to use line detection using Hough transform on a set of points that is the center of the bounding boxes.

Similar to using YOLOv8 pose detection, the data representation of CULane must be altered to initiate the model. However, due to extreme differences in how data is presented, YOLOv8 detection requires heavy label reformatting. Images and labels in the data set must also be relocated in a format similar to that used in YOLOv8 pose detection. Each key point is warped by a small bounding box with lane classification to reformat the labels. Therefore, instead of predicting key points, the model will try picking up "lanes" objects spread along the lanes. A bounding box (80, 40) surrounds each key point in the original data annotation, as shown in Fig. 5. Doing so can vastly increase the number of features in each "lane" object and subsequently raise the effectiveness of the object detection model. The model is trained for over 100 epochs.

Figure 5. Demonstration of data annotation for YOLOv8 object detection.

2.4.3. Line filtering through Hough transformation process

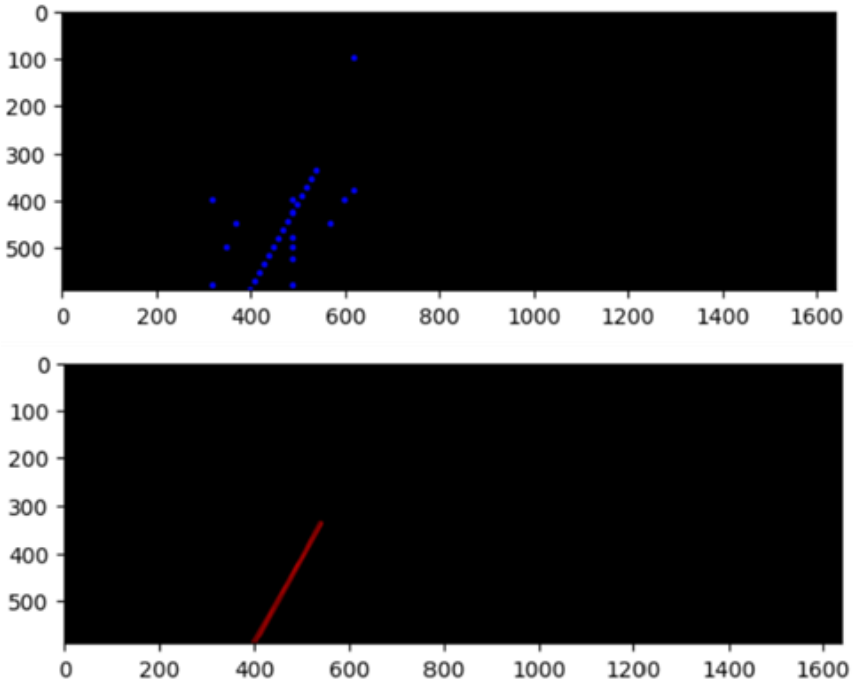


Figure 6. Demonstrates the effect of the Hough transform on filtering errors.

Finally, as mentioned before, all the centers of the detection boxes will go through a lane-detecting process by Hough transformation. For the lack of a dedicated function for Hough transform for a set of points, the post-processing stage consists of drawing the center point of each detected object on a blank image and then applying OpenCV's function for Hough transform to find the best-fit lines (Fig. 6).

The result obtained from the Hough transformation process is a set of lines with many redundancies and minor deviations. Therefore, any two lines with a variation of less than 5% are considered to be similar.

3. EXPERIMENTAL RESULTS AND DISCUSSION

3.1. Experiments setup

Due to the cumbersome size of the data set and the extensive hardware and processing power requirements, this project is conducted on Kaggle. The hardware used for this project is

as follows: CPU: Intel(R) Xeon(R) CPU @ 2.00GHz, GPU: GPU P100, RAM: 12 GB.

3.2. Image processing

Through a process of testing, the result of run-time evaluation for the lane detection method using solely OpenCV is shown in Fig. 7 below:

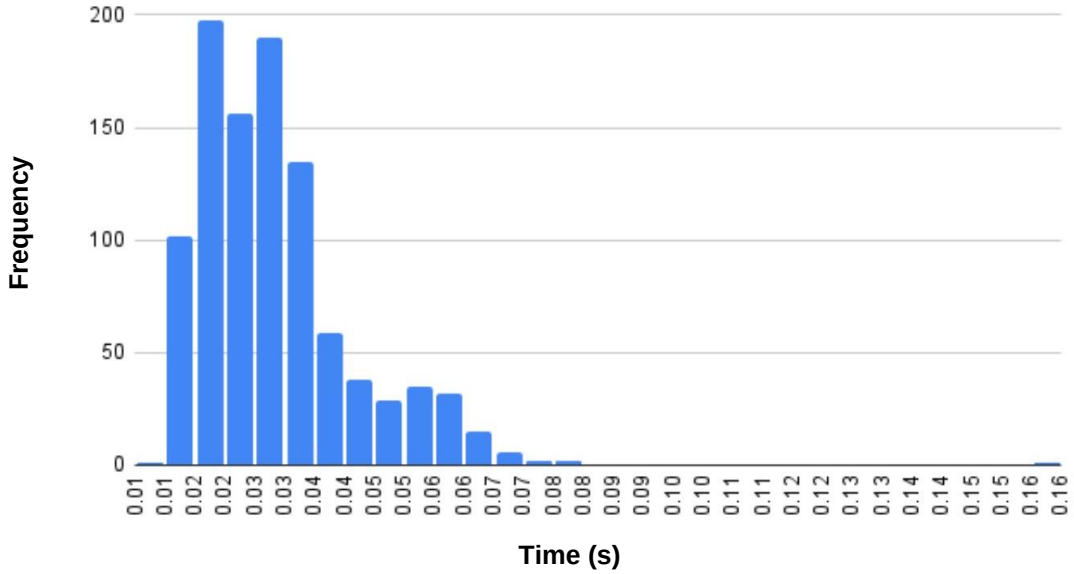


Figure 7. OpenCV method's run-time distribution.

As we can see from the diagram above, this method shows good speed with a mean processing time per image of 0.0327 seconds, resulting in about 30 frames per second on average. However, the algorithm needs to improve accuracy with the test set, with the average MSE at 0.537, falling well above the tolerable amount (Fig. 8).

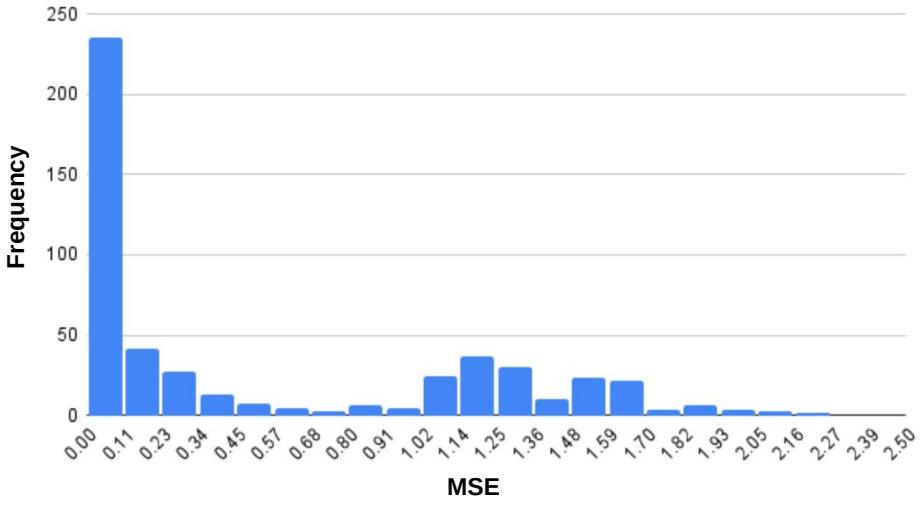
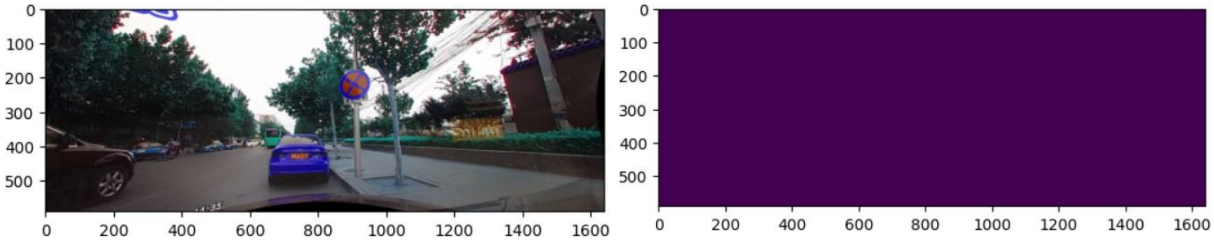


Figure 8. MSE distribution.

This poor performance can be traced back to the fact that this method is mainly affected by external conditions such as poor lighting, crowded traffic, blurry or unclear road markings,

etc. Unfortunately, the pre-processing technique that makes the whole method possible also removes crucial information from the pictures that can help significantly in those cases. With such conditions being prevalent in traffic, it can be easily seen in Fig. 9 that the method of using OpenCV for lane detection is usually unreliable.



(a) original image (b) image only shows detected lines

Figure 9. The Performance of using OpenCV method is mainly affected by external conditions.

Therefore, despite the excellent processing speed, lane detection using OpenCV must be improved in practical circumstances. New improvements and approaches must be made before this method becomes viable.

3.3. Results of pose-based model

As can be observed from Table 1 above, this method shows a breakneck mean processing speed of approximately 12.2ms, as recorded by YOLOv8's built-in function. However, the accuracy of this model, illustrated by Fig. 10, remains much to be desired, with Pose PR curve and the F1 curve showing moderate performances with low AUC, which hints at a common interpretation of Precision and Recall. MAP50 and MAP50-95 also retain only 0.597 and 0.238, respectively. Figure 11 shows that the predicted key points only mildly represent the lanes with major misplaced vital points. It needs to be more sufficient to give a reliable prediction of the road lanes.

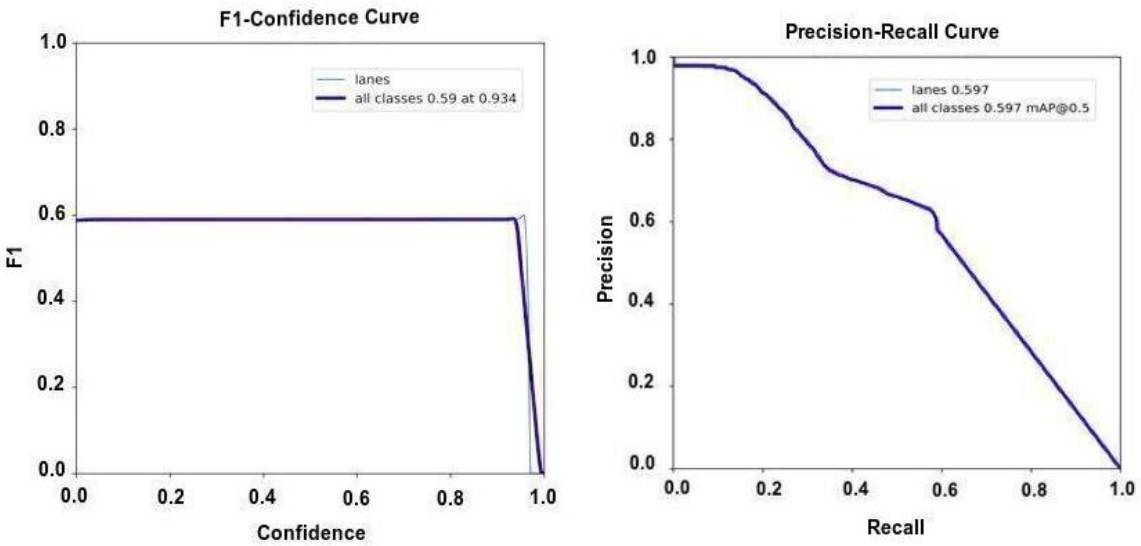


Figure 10. Result of pose-based model.

Figure 11. Pose predictions on a validation image.

Table 1. Average run time for YOLOv8 pose estimation.

	Pre-process	Inference	Post-processing	Total
Processing time	1.7 ms	8.7 ms	1.8 ms	12.2 ms

3.4. Results of object detection-based model

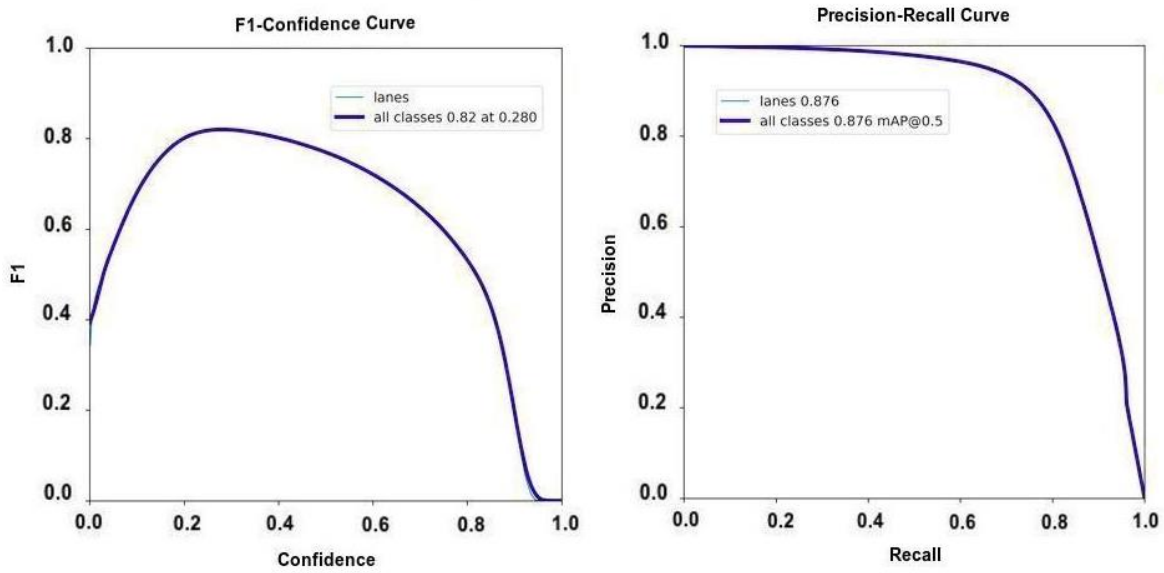
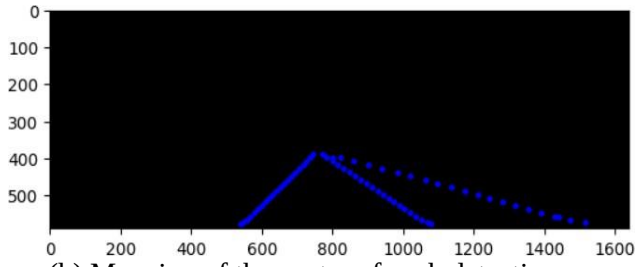


Figure 12. Results from object detection-based model.

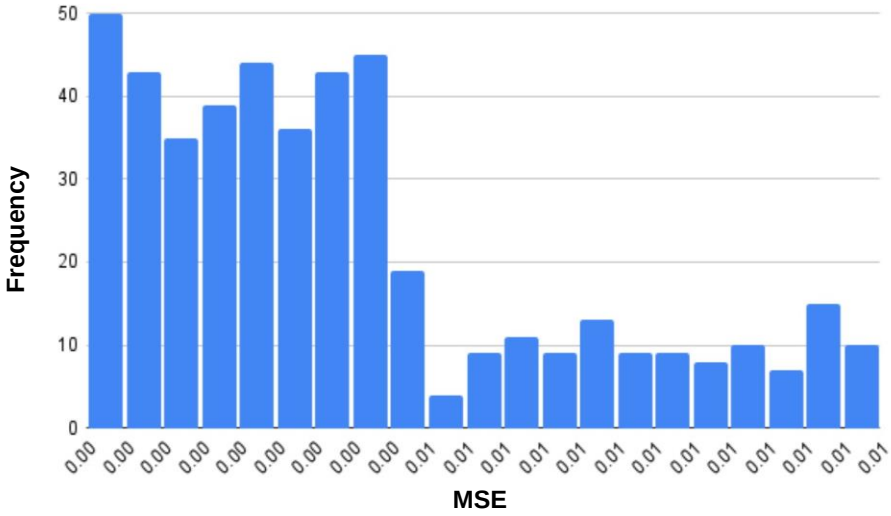
Figure 12 shows the YOLOv8 object detection results in detecting "lane" objects. The deep learning model can achieve a high MAP50 score of 0.89 and MAP50-95 of 0.68; the F1 curve and AUC of the PR curve also show excellent performance in Precision and Recall rate. This is a perfect foundation for the Hough transformation to take place. Moreover, the object detection model shows a substantial increase in processing time compared to the pose estimation model, from 12.2ms on average to 9.3ms (Table 2). It can be seen that the object detection model performs much better than the pose estimation model and, thus, makes for an excellent foundation for the Hough transformation to take place.

(a) Visualization of validation result on one particular image

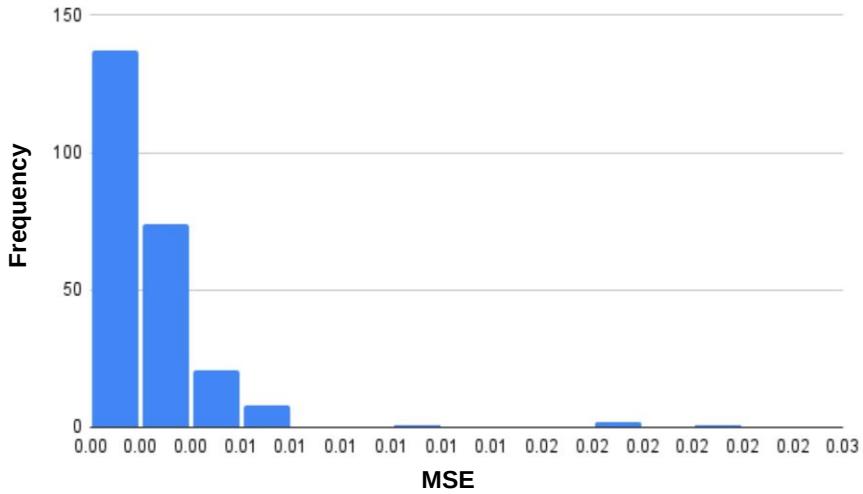


(b) Mapping of the center of each detection of the image

Figure 13. Visualization of the detection results.



(a) In normal condition



(b) at night time

Figure 14. Result of MSE distribution histogram.

Subsequently, after the results of the object detection model are put through Hough transformation, the lines obtained show substantial resemblances to the ground truth lines representing the lanes (Fig. 13). As observed from Fig. 14 below, the average MSE score of 0.0034 shows the effectiveness of the Hough transformation as a post-processing stage. Moreover, the core function of this deep learning method makes it less vulnerable to external conditions. The algorithm can still maintain its performance even in unfavorable circumstances. As demonstrated below, a great interpretation displayed by a very small average MSE of 0.0027 is still upheld even when evaluated at night.

On the other hand, we can see that by adding a post-processing stage, the run-time is weighted down a lot. Despite the deep learning model having a very short average run time of 9.5ms (Table 2), post-processing can more than double the entire run time with an average of 12.43ms per image (Fig 15). Therefore, compared to the other two methods, this approach has an average processing time per image of 0.022 seconds, which makes the average FPS (Frame per second) 45.

Table 2. Average run time for YOLOv8 object detection.

	Pre-process	**Inference**	**Post-processing**	**Total**
Processing time	1.5 ms	6.4 ms	1.4 ms	9.3 ms

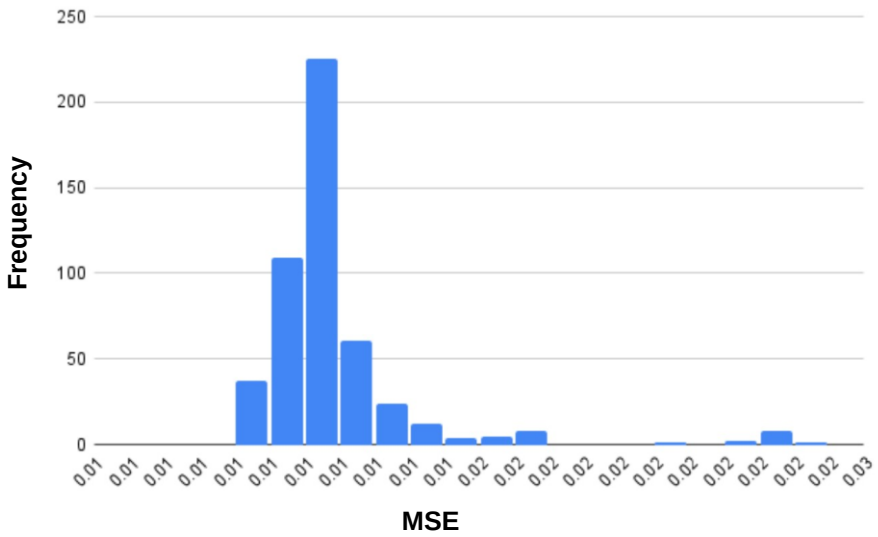


Figure 15. Rune time distribution for line filtering process.

3.5. Discussion

Despite having a reasonable processing time and being easy to carry out, lane detection using OpenCV remains unreliable because of its proneness to error. This method performs well when the conditions are ideal, which rarely happens in real-life traffic. In contrast, it is significantly worse in other less-than-ideal circumstances, which, unfortunately, are very common. This demerit can be traced to the pre-processing technique, which makes the whole process possible. A substantial amount of information intrinsic to the image is discarded by

preserving only the image edges, thereby rendering the method prone to erroneous predictions.

Regarding the performance of the YOLOv8 pose detection model, the result shows potential with excellent run-time and moderate accuracy. However, within the same data set and training time available, the result obtained from this method is worse than that of the YOLOv8 object detection model, which is illustrated by worse precision, recall, and MAP. YOLOv8 detection, in this case, has advantages over YOLOv8 pose detection since the presence of a bounding box means we can set an area with features representative of road lanes for the model to learn from; therefore, better prediction results can be obtained. Consequently, object detection is the more suitable method, resulting in more reliable detection for the Hough transformation process to take place.

Despite cumbersome post-processing stage weighting down the processing time, using object detection and Hough transformation still demonstrated prominence in accuracy and speed. Although the method of combining YOLOv8 object detection with line filtering using Hough transformation yields great result in this study, it is not without its limitations. The line filtering, despite acting as an excellent followed-up stage for the object detection process, makes the detected lanes only limited to being straight lines. Moreover, this approach may also encounter major difficulty in categorizing types of lanes due to the nature of the line fitting process, especially with dashed lanes. However, many features need improvement before this becomes a strong candidate for real-life usage. One of the first tasks that can be done is to optimize further. This module proves to be adequate on a solid system similar to that provided by cloud services like Google Colab or Kaggle. Tests conducted on an embedded system may yield different results. Moreover, all the training and testing are performed on one specific data set collected with certain camera qualities and POV, making the evaluation results only indicative of this data set. For images of different rates, the result may vary. To fix this problem in the future, we can improve the span of the data set by adding a wide range of images of different properties. In addition, complementary processes can be introduced to negate the mentioned restrains of this method in the future. For example, a curve fitting process can tremendously improve the result by solving the line-exclusivity problem of this approach; and additional categorizing stages can help identifying different types of lanes.

4. CONCLUSION

Through this project, we have explored the advantages and disadvantages of using image processing and deep learning models with the addition of Hough transformation. All the methods were tested with images from the extensive data set CULane. We can see the potential of applying deep learning to lane detection with the results obtained. Deep learning models generally show better accuracy and speed without needing heavy pre-processing stages that remove much information from the images, with the best results obtained from combining deep learning for object detection with image processing techniques for post-processing stages. The mentioned method can give accurate predictions and totals in a sufficiently short run time. With that in mind, deep learning has the massive potential to solve the problem of detecting lanes for self-operating automobiles.

REFERENCES

[1]. D. Vighnesh, S. Ganesh, K. Hritish, D. Gaurav, Lane Detection Techniques using Image, in Processing of ITM Web of Conferences, August 2021. http://dx.doi.org/10.1051/itmconf/20214003011
[2]. Z. Wang, W. Ren, Q. Qiu, LaneNet: Real-Time Lane Detection Networks for Autonomous Driving, (2018). https://doi.org/10.48550/arXiv.1807.01726
[3] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, Q. Wang, Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks, IEEE Transactions on Vehicular Technology, 69 (2020) 41-54. https://doi.org/10.1109/TVT.2019.2949603
[4]. R. Kulkarni, S. Dhavalikar, S. Bangar, Traffic Light Detection and Recognition for Self-Driving Cars Using Deep Learning, Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (2018) 1–4. https://doi.org/10.1109/ICCUBEA.2018.8697819
[5]. Dong-Joong Kang, Mun-Ho Jung, Road Lane segmentation using dynamic programming for active safety vehicles, Pattern Recognition Letters, 24 (2003) 3177-3185. https://doi.org/10.1016/j.patrec.2003.08.003
[6]. J. M. Collado, C. Hilario, A. Escalera, J. M. Armingol, Adaptative Road Lanes Detection and Classification, Lecture Notes in Computer Science, (2006). https://doi.org/10.1007/11864349_105
[7]. Wentao Yao, Zhidong Deng, Robust Real-Time Lane Marking Detection for Intelligent Vehicles in Urban Environment, Lecture Notes in Electrical Engineering, v122 (2011) 421-428. https://doi.org/10.1007/978-3-642-25553-3_52
[8]. Q. Lin, Y. Han, H. Hahn, Real-Time Lane Departure Detection Based on Extended Edge-Linking Algorithm, 2nd International Conference on Computer Research and Development, (2010). https://doi.org/10.1109/ICCRD.2010.166
[9]. U. Suddamalla, S. Kundu, S. Farkade, A. Das, A Novel Algorithm of Lane Detection Addressing Varied Scenarios of Curved and Dashed Lanemarks, (2015) 87-92. https://doi.org/10.1109/IPTA.2015.7367103
[10]. V. Devane, G. Sahane, H. Khairmode, G. Datkhile, Lane Detection Techniques using Image Processing, ITM Web of Conferences 40 (2021). https://doi.org/10.1051/itmconf/20214003011
[11]. Jocher Glenn, Ayush Chaurasia, Jing Qiu, Ultralytics YOLOv8. Version 8.0.0. https://github.com/ultralytics/ultralytics (accessed 25 January 2024).
[12]. Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, Xiaoou Tang, Spatial as Deep: Spatial CNN for Traffic Scene Understanding, Proceedings of the AAAI Conference on Artificial Intelligence, (2018) 7276–7283. https://doi.org/10.48550/arXiv.1712.06080
[13]. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick, Microsoft COCO: Common Objects in Context, 2014. https://doi.org/10.1007/978-3-319-10602-1_48