Efficient and real-time lane detection using CUDA-based implementation

Hoda El boussaki[1], *Rachid Latif*[1], and *Amine Saddik*[1,2]

[1]Laboratory of Systems Engineering and Information Technology LISTI, ENSA,Agadir, Morocco
[2]Faculty of Applied Sciences, Ait Melloul, Morocco

Abstract. Lane detection is an essential component of autonomous driving systems, enabling vehicles to accurately identify and follow road markings. In this paper, we look at an lane detection approach that integrates median filtering and the Hough transform. Median filtering is an essential pre-processing step for reducing noise and improving lane detection accuracy. However, given its high computational demands, optimization of this process is essential for real-time applications. To this end, we used CUDA for acceleration, taking advantage of its parallel computing capabilities to improve performance. We implemented and tested this optimised lane detection algorithm on the NVIDIA Jetson Nano and on a desktop, providing a comparative analysis of improvements in efficiency and speed. This approach highlights the potential of real-time path detection in embedded and high-performance computing environments.

1 Introduction

Lane detection is a crucial element in improving the safety and dependability of autonomous driving systems. By precisely recognizing lane boundaries and configurations, it allows self-driving vehicles to stay centred within their lanes, perform safe lane changes, and handle complex driving scenarios. This capability minimises the likelihood of unintentional lane departures, decreases the risk of accidents, and enhances overall traffic efficiency.

Lane detection is a crucial component of autonomous driving and advanced driver assistance systems (ADAS), with numerous methods being developed to improve its accuracy and reliability. Traditional techniques, such as the Canny edge detector and Hough Transform, have been widely used for detecting lane boundaries in structured road settings, as demonstrated by Pizzati et al. [1]. However, these conventional methods often face challenges under adverse conditions, such as low lighting or lane occlusions. To overcome these limitations, researchers have increasingly turned to machine learning and deep learning approaches. For example, He et al. [2] introduced a deep convolutional neural network (CNN) capable of detecting lanes with greater precision by learning features directly from raw image inputs. In addition, Behrendt et al. [3] proposed a hybrid model that integrates deep learning with traditional vision techniques, achieving superior performance in complex driving environments. These advancements highlight the transition from classical image processing to sophisticated AI-driven solutions, enhancing lane detection's robustness and accuracy in real-world scenarios.

In our paper, we present an algorithm for lane detection via the Hough transform. The lane detection algorithm utilises a median filter for pre-processing, adaptive thresholding, and the Hough Transform, all implemented using OpenCV. The algorithm was deployed on both a desktop environment and the Jetson Nano platform, and it is structured into several distinct processing blocks. Among these, the median filter was identified as the most computationally intensive component. To optimise performance, we accelerated this filter using CUDA, significantly reducing its processing time on the Jetson Nano. This enhancement not only improved the overall speed of the lane detection system but also demonstrated the feasibility of running complex vision algorithms on resource-constrained embedded platforms, making it suitable for real-time applications in autonomous driving.

The summary of our contribution is is as follows:

- The examination of the temporal constraints of the lane detection algorithm through a Hardware/Software Co-Design methodology.
- The implementation and optimisation using CUDA of the algorithm on the Jetson Nano platform.

This paper is structured as follows: the first section gives a state of the art on lane detection methods. The second section gives an overview of the Hough transform. The third part gives a description of the methodology used. The fourth part presents the results obtained. Lastly, we will conclude our paper.

2 State of the art

Lane detection has been extensively researched as it is pivotal for the advancement of autonomous driving and advanced driver assistance systems (ADAS). Early research focused on using classical computer vision techniques, such as the Sobel operator, Hough Transform, and RANSAC algorithm, for lane boundary detection [4-5]. For instance, Kaur and Kaur [6] used the Sobel operator for edge detection, followed by the Hough Transform for line detection, achieving satisfactory results on structured roads. However, these methods are susceptible to failure in complex environments, such as unmarked lanes, varying illumination, and occlusions. Zhang et al. [7] proposed an improved RANSAC algorithm to address these issues by increasing the robustness of lane model fitting under noisy conditions. These classical approaches, while computationally efficient, often struggle with the complexities of real-world driving scenarios, leading to the adoption of more advanced techniques.

Recent advancements have seen a significant shift towards deep learning-based methods for lane detection, which can learn high-level features from raw image data. Pan et al. [8] introduced the SCNN (Spatial CNN) model, which uses a sequential convolutional neural network to capture spatial dependencies between pixels, significantly improving lane detection performance in curved and challenging environments. Similarly, Hou et al. [9] proposed an instance segmentation approach that uses a row-based CNN model to classify each pixel as belonging to a specific lane line, achieving state-of-the-art accuracy. Zhao et al. [10] further enhanced lane detection using a multi-task learning approach that combines lane detection with road segmentation, demonstrating superior adaptability to varying road conditions. These approaches leverage the powerful feature extraction capabilities of deep learning, making them robust to diverse lane geometries and road conditions.

In addition to these advanced methods, hybrid approaches that combine deep learning with traditional vision techniques have shown promise in improving both accuracy and computational efficiency. For instance, Li et al. [11] proposed a hybrid framework that integrates a CNN for lane marking detection with a Kalman filter for lane tracking, achieving real-time performance on embedded platforms like the NVIDIA Jetson. Similarly, Ko et al. [12] utilised a deep learning model for initial lane marking detection, followed by a particle filter for robust tracking, enhancing the system's performance in occluded and dynamic environments. Wang et al. [13] introduced a deep-learning-based lane detection algorithm with a dedicated module for identifying challenging road conditions, such as faded lane markings and heavy traffic. These hybrid methods illustrate the ongoing efforts to balance the strengths of traditional and deep learning approaches, ensuring reliable and efficient lane detection in a wide range of driving scenarios.

Moreover, Tabelini et al. [14] used a transformer-based method, like LaneFormer, for their ability to model global contexts, offering improved performance over CNN-based models in complex driving environments. Zheng et al. [15] also used LaneFormer to address the limitations of traditional CNNs in lane detection by capturing global contextual information more effectively, particularly in complex environments with occlusions and sharp lane curvatures. Also, Wang et al. [16] proposed a hybrid CNN-Transformer architecture, combining the strengths of CNNs for local feature extraction with the global context modelling abilities of transformers, resulting in improved lane detection performance in challenging road scenarios.

3 THe Hough transform: an overview

In 1962, Paul Hough pioneered an innovative technique for the detection of straight lines in binary images, laying the groundwork for what we now refer to as the Hough Transform [17]. This technique operates by transforming image points into a parameter space, where each line can be represented as distinct points. Rather than directly seeking connected line segments in images that may be noisy or incomplete, Hough's methodology involves accumulating votes in an "accumulator space" for potential line parameters. The identification of lines occurs through peaks in this space, which indicate the most probable line candidates. This approach is particularly effective at mitigating the effects of noise and dealing with incomplete boundaries, allowing for the detection of lines even when significant portions are obscured. Over time, this foundational work has been adapted to recognize other geometric forms, such as circles and ellipses, solidifying its status as a crucial technique in the field of image processing

The Hough Transform is an advanced feature extraction technique that focuses on identifying shapes, even when the shape's boundaries are incomplete or corrupted by noise [18]. It achieves this by converting the detection problem from the image space to a parameter space, where a voting mechanism identifies the best candidate shapes. Each point in the image contributes votes to potential shapes, and the most likely candidates are found by locating local maxima in an accumulator space, which is explicitly constructed as part of the algorithm. The classical Hough Transform was initially developed to detect straight lines, representing them in a polar coordinate form:

$$R = x.cos(\theta) + y.sin(\theta) \quad (1)$$

where r is the distance from the origin and θ is the angle of the line's normal.

The relationship between the origin and angle of a straight line and the position of a straight line is shown in figure 1.

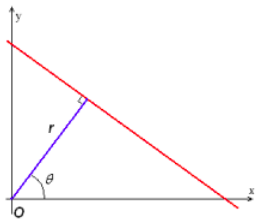

Fig. 1. The connection between the origin and angle of a straight line and the position of a straight line [19]

Each line can be associated with a pair of parameters, represented as (r, θ). This (r, θ) coordinate system is often referred to as Hough space and is utilised for defining two-dimensional straight lines. In this context, r indicates the distance from the origin to the line, while θ denotes the angle of the line's normal. The Hough Transform leverages this parameter space to efficiently identify lines in an image, making it a fundamental technique in image processing and computer vision.

One of the transform's major advantages is its ability to tolerate imperfections such as missing parts of the shape or noise, making it highly resilient for real-world applications. The Hough Transform has since been extended to detect other shapes like circles and ellipses by expanding the parameter space to include additional parameters like the radius in the case of circles. Each point in the image contributes to a vote in this expanded space, enabling robust shape detection even under challenging conditions [20].

4 Methodology

We used a 9-second road video. Figure 2 shows the lane detection algorithm. The algorithm is divided into two blocks: the pre-processing block and the Hough transform lane detection block. The first block is divided into 4 functional blocks. In the first functional block, the RGB image is converted to grayscale, then in the second the image is filtered using the median filter. The third functional block is adaptive thresholding, which is used to segment the image into binary. Finally, the fourth is used to extract the region of interest, which is the lower region of the image. The second block is divided into 2 functional blocks. The first applies the Hough transform and the second finds one line per side.

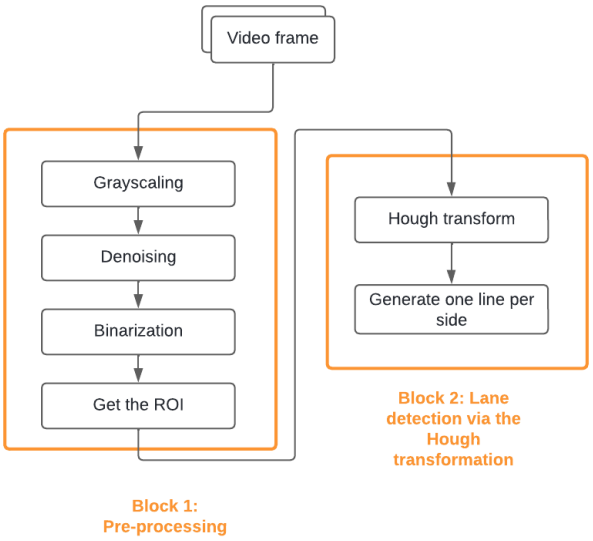

Fig. 2. Global algorithm

4.1 Pre-processing

In this step, the coloured image is transformed into a grayscale image, as the edge detection technique only works with grayscale images. Next, the method uses the median filter as a smoothing filter, as the median filter can effectively reduce noise while preserving salient edges. The median filter was found to be better than the gaussian filter when considering only the time parameter [21]. The median filter is recognized as a non-linear technique utilised in image processing for effectively mitigating noise while retaining edge details. This filter works by replacing the value of each pixel with the median of the values in its neighbouring pixels, making it especially useful for eliminating salt-and-pepper noise. Its nonlinear nature allows it to outperform many linear filtering methods in preserving important image features [22]. Next, adaptive thresholding is used where the threshold is calculated for each small region of the image based on local pixel values. The size of the region around each pixel for which the threshold is value 9 and the constant subtracted from the threshold calculation to refine the results is 2. For each pixel, a small square region of size blockSize is taken around that pixel. The threshold for this pixel is calculated based on the values of the pixels in this region, then adjusted by the constant. If the pixel value is greater than the calculated threshold, the pixel is set to 255. The next step is the extraction of the region of interest, focusing on the lower part where road lanes are most likely to appear in a lane detection system. The function block creates a triangular ROI that simulates the area where road lanes could be detected. First the height and width of the input image are calculated, then three points of a triangle are defined to represent the desired ROI. These points vary according to the fov value, allowing the width of the triangular area at the bottom of the image to be adjusted. A mask is created by filling this triangle with

white on a black background. This mask then leaves visible only those pixels within the triangular area of interest, all others being obscured.

4.2 Lane detection via the Hough transform

OpenCV's HoughLinesP function is used to detect line segments in an image using the probabilistic Hough transform, a variant of the standard Hough transform. The first parameter is the source image , which must be a binary image, usually obtained by edge detection. The second parameter is a vector that stores the detected lines, each line being represented by its start and end points (x1, y1, x2, y2). The third parameter defines the distance resolution of the accumulator in pixels, a smaller value increasing the accuracy of line detection. The fourth parameter defines the angular resolution in radians, generally set at CV_PI/180 for intervals of 1 degree. The fifth parameter is the threshold, which determines the minimum number of intersections in the accumulator for a line to be detected. Higher thresholds result in fewer lines being detected, concentrating on the most important ones. The sixth parameter specifies the minimum length of detected lines, ensuring that only lines longer than this value are taken into account, while the seventh parameter enables nearby line segments to be merged by defining the maximum distance between them. Together, these parameters enable precise control of the lines detected in the image, making the function adaptable to different scenarios.

The second function block processes a set of detected line segments, usually from a Hough transform, to group them into two primary lines representing the left and right lanes of a road. It first initialises vectors to store the x and y coordinates of these grouped lines. For each line segment, the block calculates its slope to determine whether it belongs to the left or right group; lines with a negative slope are grouped as left lines, and those with a positive slope are grouped as right lines. The block then uses a polynomial fitting technique (polyfit_boost) to estimate the linear representation of these grouped lines in the specified y-coordinate range (min_y to max_y). The final output is a vector containing the x-coordinates of the start and end points of the left and right lines, together with the corresponding y-coordinates, thus summarising the boundaries of the detected lanes. Figure 3 shows the tracks detected on an image.

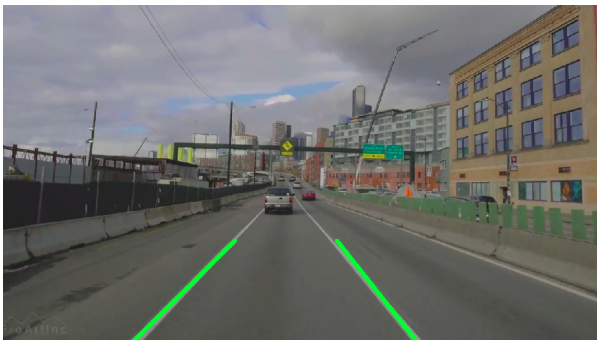

Fig. 3. Lane detected on an image

5 Hardware/Software results

5.1 Sequential implementation

The algorithm was implemented across various architectures using the C/C++ programming language, chosen for its efficiency and performance optimization capabilities. The implementation consists of two main processing blocks, each designed to handle specific tasks within the overall algorithm. For a comprehensive analysis, the processing time for each block was meticulously measured and recorded. This allows for a clearer understanding of the performance characteristics associated with each architectural configuration. The results of the time evaluation are summarised in Table 1, which provides a detailed breakdown of the processing times, while Figure 4 visually represents the comparative performance of the different architectures, facilitating an easy interpretation of the data.

The Critical Peak Power (CPP) is a measure used to assess the performance of implementations on different processing units. To calculate this metric, it is necessary to use the frequency (F) in hertz, the execution time (T), and the image resolution (R) [23]. It was determined using the following equation:

$$CPP = \frac{F \times T}{R} \quad (2)$$

Table 1. Processing time for each block.

Blocks	Desktop	Jetson Nano
Block 1	38.98 ms	121.56 ms
Block 2	3.575 ms	38.09 ms
Total	42.555 ms	159.65 ms
CPP	130	248

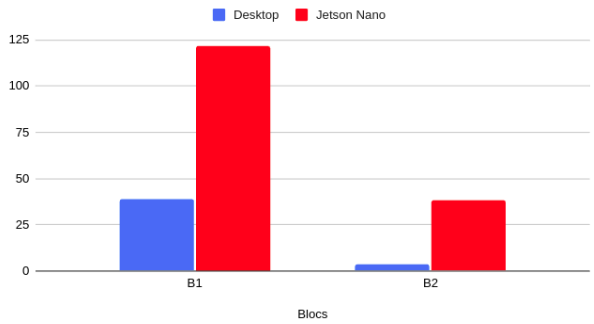

Fig. 4. Processing times for different blocks on the Desktop and Jetson Nano

A comparative time evaluation was conducted on two machines to assess their image processing capabilities. The results indicated that the desktop computer outperformed the alternative tools, achieving a total processing time of 42.555 ms for each image. In contrast, the Jetson Nano exhibited a significantly longer processing time of 159.65 ms per image. To enhance the efficiency of our image processing workflow, we attempted to decompose the first block of processing into four distinct functional blocks. The first functional block FB1 converts the RGB image to grayscale. This step reduces the computational complexity by eliminating colour information, which is unnecessary for lane detection. Grayscale conversion simplifies the image to a single channel, making it easier to process while retaining critical intensity information that highlights lane markings. FB2 applies the median filter. The purpose of this filter is to reduce noise while preserving the edges. In FB3, the image is binarized to determine if the pixel will be black or white. In this step, a threshold is applied to distinguish between the lane markings and the background. This binary representation simplifies the subsequent processing steps by reducing the complexity of the image and clearly separating potential lane lines from the surrounding environment. The last functional block FB4 selects the region of interest that is the lower half of the image. This segmentation aimed to optimise performance and improve the overall processing speed. The specifics of the time evaluation for these various functional blocks are summarised in table 2, providing a clear comparison of the processing times for each segment.

Table 2. Processing time for each functional block.

Blocks	Desktop	Jetson Nano
FB1	4.52 ms	2.35 ms
FB2	32.62 ms	107.93 ms
FB3	1.68 ms	9.8 ms
FB4	0.16 ms	1.48 ms
Total	38.89 ms	121.56 ms

5.2 CUDA-based implementation

In this section, we focus solely on the second functional block. We've opted for CUDA to exploit the advantages of an Nvidia GPU. The second functional block is the filtering block, using OpenCV's medianBlur function. Using CUDA to optimise the medianBlur function considerably accelerates image processing by taking advantage of the parallel computing power of GPUs (graphics processing units). The medianBlur function reduces noise by replacing each image pixel with the median of neighbouring pixels. However, this operation is computationally expensive, as it requires the pixels in the neighbourhood window to be sorted for each pixel in the image. By implementing this function on the GPU with CUDA, each pixel can be processed independently and simultaneously by thousands of threads, which reduces computation time compared with CPU execution. The image data is transferred to GPU memory, where the median calculation is performed in parallel for each block of pixels, and the results returned to main memory. This approach is particularly beneficial for large images or real-time applications, where processing speed is crucial. Table 3 summarises the time obtained using CUDA on the Desktop and Jetson Nano and table 4 shows the computed CPP after using CUDA.

Table 3. CUDA-based processing times for each functional block.

Blocks	Desktop	Jetson Nano
FB1	4.52 ms	2.35 ms
FB2	5.6 ms	18.54 ms
FB3	1.68 ms	9.8 ms
FB4	0.16 ms	1.48 ms
Total	38.89 ms	121.56 ms

Table 4. Computed CPP.

Blocks	Desktop	Jetson Nano
Total	48	110

After utilising CUDA, the Critical Peak Power (CPP) decreased significantly, dropping from 248 to 110 on the Jetson Nano and from 130 to 48 on the

desktop. These changes highlight the effectiveness of CUDA in optimising performance across both platforms. The substantial reduction in CPP on the Jetson Nano indicates that the implementation is now more power-efficient, even as it manages to handle intensive computational tasks. Similarly, the desktop's decreased CPP demonstrates enhanced efficiency through parallel processing, resulting in a lower peak power requirement. Overall, these results underscore the advantages of employing CUDA to maximise hardware performance and efficiency in various processing environments.

The results of our evaluation revealed that the total processing time for Block 1 was 11.96 milliseconds (ms) when executed on the desktop computer, while the Jetson Nano recorded a processing time of 32.17 ms for the same block. This indicates a significant performance improvement, with the desktop achieving a 2.7-fold enhancement in processing time compared to its sequential implementation, which originally took 42.555 ms. Similarly, the Jetson Nano demonstrated a 2.3-fold improvement over its sequential version, which required 159.65 ms for processing. These optimizations highlight the efficiency gains achieved through parallel processing techniques. The time evaluation for both blocks, following the implementation of CUDA, is illustrated in figure 5, which provides a visual comparison of the processing times before and after the optimization, further emphasising the enhancements in performance.

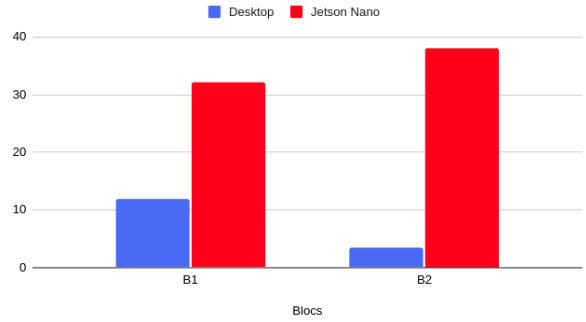

Fig. 5. Processing times for different CUDA-based blocks

6 Conclusion

In this work, we have studied a lane detection system using the median filter as a pre-processing step and the Hough transform to detect lane markings. The median filter, essential for noise reduction, was identified as the most computationally-intensive part of the process. To improve its performance, we optimised the median filter using CUDA and implemented it both on the NVIDIA Jetson Nano and on a Desktop. The Jetson Nano provided an efficient solution for real-time embedded applications, while the Desktop offered superior processing power for handling high-resolution images. This optimization considerably improved the overall efficiency and speed of the traffic lane detection system. Future perspectives include studying the algorithm's limitations in real-world scenarios and conducting tests under varying lighting conditions and different levels of noise.

Acknowledgment

We owe a debt of gratitude to the Ministry of National Education, Vocational Training, Higher Education and Scientific Research (MENF-PERSRS) and the National Center for Scientific and Technical Research of Morocco (CNRST) for their financial support (grant number: 27UIZ2022) and for the project Cov/2020/109.

References

1. F. Pizzati, A. Allodi, and F. Debattista, “Learning to detect and track visible and occluded lane markings using recurrent neural networks,” in 2018 IEEE Intelligent Vehicles Symposium (IV), Jun. 2018, pp. 1721-1726.
2. Y. He, H. Wang, and B. Zhang, “A deep learning approach for traffic lane detection and departure warning,” IEEE Intelligent Transportation Systems Magazine, vol. 8, no. 2, pp. 58-68, 2016.
3. K. Behrendt, L. Novak, and R. Botros, “A deep learning approach to lane detection using the LANet framework,” IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 2, pp. 448-458, 2019.
4. K. Jung and C. Kim, “Robust lane detection based on two-stage feature extraction,” in Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, 2007, pp. 23-28.
5. J. W. Lee and M. Lee, “Vision-based lane detection for two-wheeled robot,” in 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 300-305.
6. P. Kaur and M. Kaur, “A hybrid approach for lane detection using edge detection and Hough transform,” in Proceedings of the 2015 International Conference on Computing, Communication & Automation, 2015, pp. 1221-1226.
7. Q. Zhang, Y. Liu, D. Cao, and J. Zhang, “Robust lane detection under different road mark conditions,” Sensors, vol. 18, no. 6, p. 1817, 2018.
8. X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, “Spatial as deep: Spatial CNN for traffic scene understanding,” in Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 7276-7283.
9. Y. Hou, Z. Ma, C. Liu, and C. C. Loy, “Learning lightweight lane detection CNNs by self attention distillation,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1013-1021.
10. X. Zhao, K. Zhu, C. Zhao, and J. Gong, “Multi-task learning for lane detection and road scene segmentation,” in Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 291-296.

11. X. Li, J. Liu, W. Zhou, and W. Li, “Robust real-time lane detection for driver assistance systems,” in Proceedings of the 2018 International Conference on Computer Vision and Pattern Recognition, 2018, pp. 1572-1578.
12. Y. Ko, S. Lee, H. Jung, and W. Woo, “Hybrid lane detection using CNN and particle filter for self-driving car,” in Proceedings of the 2018 International Conference on Electronics, Information, and Communication (ICEIC), 2018, pp. 1-3.
13. H. Wang, Z. He, Q. Chen, and W. Zhang, “A lane detection algorithm based on deep learning,” in Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2018, pp. 1712-1717.
14. L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, “PolyLaneNet: Lane Estimation via Deep Polynomial Regression,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1242–1251, Jan. 2021.
15. Z. Zheng, Z. Yuan, Z. Tang, Y. Yu, Y. Li, and Y. Zhou, "LaneFormer: Object-Query Based End-to-End Lane Detection," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2022, pp. 8784-8793.
16. X. Wang, J. Guo, K. He, F. Wang, and X. Yang, "Hybrid CNN-Transformer for Lane Detection in Challenging Environments," in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV), 2023, pp. 1124-1133.
17. P. V. C. Hough, "Method and Means for Recognizing Complex Patterns", US Patent 3,069,654, Ser. No. 17,7156 Claims, 1962.
18. S. T. Karri, “Hough Transform - Surya Teja Karri - Medium,” Medium, Dec. 11, 2021. [Online]. Available: https://medium.com/@st1739/hough-transform-287b2dac0c70
19. X. Li, P. Yin, Y. Zhi, and C. Duan, “Vertical lane line detection technology based on Hough Transform,” IOP Conference Series Earth and Environmental Science, vol. 440, no. 3, p. 032126, Feb. 2020.
20. J.-Q. Zhang, H.-B. Duan, J.-L. Chen, A. Shamir, and M. Wang, “HoughLaneNet: Lane detection with deep hough transform and dynamic convolution,” Computers & Graphics, vol. 116, pp. 82–92, Aug. 2023.
21. A. Kumar and S. S. Sodhi, “Comparative analysis of Gaussian Filter, Median Filter and Denoise Autoenocoder,” 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), 2020.
22. S. A. Villar, S. Torcida, and G. G. Acosta, “Median Filtering: A New Insight,” Journal of Mathematical Imaging and Vision, vol. 58, no. 1, pp. 130–146, Dec. 2016.
23. A. Saddik, R. Latif, M. Elhoseny, and A. E. Ouardi, “Real-time evaluation of different indexes in precision agriculture using a heterogeneous embedded system,” Sustainable Computing Informatics and Systems, vol. 30, p. 100506, Jan. 2021.