An Efficiency Hardware Design for Lane Detector Systems

Duc Khai Lam
University of Information Technology, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam

***Abstract*—The Hough Transform (HT) algorithm is a popular method for lane detection based on the 'voting' process to extract complete lines. The voting process is derived from the HT algorithm and then executed in parameter space (ρ, θ) to identify the 'votes' with the highest count, meaning that image points with pairs of angle θ and distance ρ corresponding to those 'votes' lie on the same line. However, this algorithm requires significant memory and computational complexity. In this paper, we propose a new algorithm for the Hough Space (HS) by utilizing parameterization (Y-intercept, θ) instead of (ρ, θ) parameterization and lane direction. This simplifies the inverse LHT operation and reduces the accumulator's size and computational complexity compared to the standard LHT. We aim to minimize processing time per frame for real-time processing. Our implementation operates at a frequency of 250MHz, and the processing time for each frame with a resolution of 1024x1024 is 4.19ms, achieving an accuracy of 85.49%. This design is synthesized on the Virtex-7 VC707 FPGA.**

***Keywords*—FPGA; Hough transform; look up table; lane detector; autonomous vehicle**

I. Introduction

Lane detection is one of the crucial objectives in image processing and computer vision, extensively applied in industries such as vehicle guidance and Advanced Driver Assistance Systems (ADAS). It involves detecting white or yellow lane markings. In some vision applications for Lane Departure Warning Systems (LDWS), the Hough Transform algorithm is widely utilized for lane detection due to its robust and effective detection capability, even in environments with significant noise or multiple non-contiguous lines [1]–[3]. This method relies on the 'voting' process and extracts complete lines.

Recent studies have focused on enhancing the Voting method within the Hough Transform for real-time computation. In [4], the authors applied Parallel Voting on an FPGA, utilizing a 2D array accumulator for line computation in the Hough Space with parameter pairs (ρ, θ). By transforming the array into a 1D array and partitioning the Hough Space into parallel voting blocks, concurrent determination of lines and parameter computation (ρ, θ) was achieved. This accelerated the process, resulting in an average processing speed of 5.4ms per frame at a frequency of 200MHz, making video processing more feasible.

Similarly, subsequent authors proposed a hardware architecture for HT serving lane detection using the Parallel Voting method, implemented on FPGA in the scientific study [5]. Based on θ, the values in the Hough Space were parallelized. To detect edges of image frames in videos, computations of (ρ, θ) were performed to extract the highest voting value in the Hough Space to determine the lines. The achieved processing speed was approximately 135 frames/s when deployed on the FPGA kit, operating at a frequency of 50MHz, and the image transmission protocol was VGA (640x480).

In the study [6], the authors developed a new algorithm for the Hough Parameter Space (HPS), which significantly reduced memory requirements compared to the standard Hough Transform (HT) algorithm. This method also supported accelerated Inverse Hough Transform (IHT) and reduced the accumulator size for voting. The efficiency of the proposed architecture was demonstrated through hardware-software co-simulation on the Xilinx Virtex-5 ML505 platform. The architecture from reference [4] allowed for a processing time of 1.47ms per frame for an image size of 640x480 pixels and an operating frequency of 200MHz.

The Angular Regions - Line Hough Transform (AR-LHT) method, based on techniques from LHT, is a memory-efficient approach for line detection in images. Utilizing the Hough Parameter Space (HPS) with minimal dispersion reduces memory usage significantly, as reported in [7]. Authors employ two smaller memories: a 1-bit Bitmap Region (RBM) and a downsized HPS. RBM determines peak orientation precisely after the voting process. Results show a 48% decrease in RAM usage compared to standard LHT for images sized 1024x1024 pixels. FPGA processing time for one image is 9.03ms.

In the study [8], the authors propose an HT architecture that uses a Look Up Table (LUT) to store trigonometric values and use the value of orientation θ calculated in the Sobel Edge Detection algorithm instead of rotating small angles as the HT standard. The processing time per 1024x1024 image resolution frame is 6.17ms with an accuracy of 94%. This design is synthesized on the FPGA Virtex-7 VC707.

In the study [9], [10], the authors implemented a real-time single-camera lane detection. To address different lighting conditions, they employed vital algorithms such as the Otsu, Canny algorithms and the Hough transform for lane detection to determine the Region of Interest and minimize computational complexity; detecting vanishing points is crucial.

The main contribution of [11] is to present an efficient implementation of Hough Transform based on Gradient for line detection, using Xilinx Virtex-7 FPGA with digital signal processing (DSP) units and integrated RAM blocks. The architecture is implemented with a working frequency of 260.061MHz and $2n + (\sqrt{2} + 2)n + 232$ clock cycles for a grayscale image of size n * n.

For complicated conditions, such as rain and night illuminations, the new processing method, comprising four stages: Gaussian blur, grayscale conversion, Dark-Light-Dark threshold (DLD) algorithm, edge extraction using correlation filters, and Hough Transform according to [12], [13], has been developed. It effectively addresses lane complexities including arrows and text. Validation results demonstrate a maximum detection rate of 97.2%.

In [14], the authors present an algorithm for simple and user-friendly lane detection using the Line Segment Detector (LSD). This system maintains a throughput performance of 60 frames per second (fps) for VGA images (640×480) on the PYNQ-Z1 board with the Xilinx XC7Z020-1CLG400C FPGA.

To optimize the hardware resource for rho and theta calculations in the voting process of the line Hough Transform, the authors [15] propose an efficient memory design. This design is implemented in TSMC ASIC 90nm technology. It requires only 4174 MB RAM.

In this work, we focus on presenting an efficient hardware architecture design for the lane detection model to achieve real-time processing for large-resolution videos. We optimize hardware resources by reducing memory size and computational complexity. The lane detection core employs the Hough Transform algorithm. The proposed system includes the preprocessing process using the Gray Scale algorithm, Sobel Edge Detection, and the central processing algorithm - Hough Transform, implemented on an FPGA kit. The architecture focuses on accelerating hardware performance for the Hough Transform and Inverse Hough Transform algorithms by utilizing parameterization (Y-intercept, θ) in the Hough space and significantly reducing hardware resources by applying Region of Interest. Enhancements will concentrate on memory optimization of modules within the design and simplifying computational operations.

The rest of the paper is organized as follows. Section II presents the proposed hardware design architecture. Section III shows the evaluation and comparison results. Finally, Section IV gives the conclusions of this paper.

II. Proposed Hardware Design Architecture

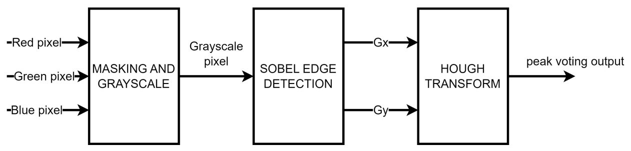

Fig. 1. Hough transform system.

Fig. 1 illustrates the architecture of the Lane Detection System using the Hough Transform, with the input being pixel values of an image and the output being the ρ and θ values after being voted in the Hough Transform module.

The proposed lane detection system consists of three parts, including the hardware architectures of the Gray Scale, Sobel Edge Detection, and Hough Transform algorithms. The functional blocks in this system are designed using the Verilog language. The available blocks of Gray Scale are referenced from the Masking module architecture, Gray Scale, Sobel Edge Detection, and the Hough Transform module is referenced from the study [6]. This paper will focus on utilizing the Region of Interest (ROI) to reduce resources. The system's ROI is determined through experimental measurements, as depicted in Fig. 2. The details of the improved and optimized functional modules will be described below.

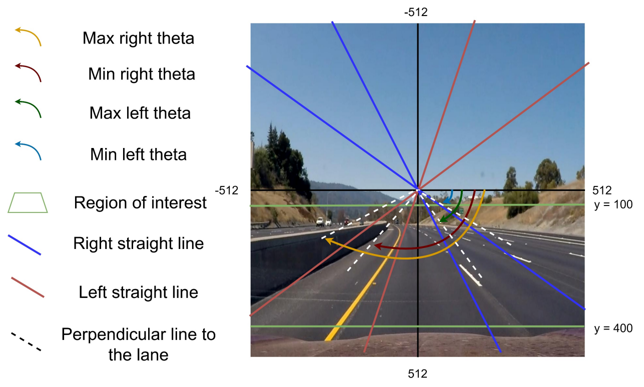

Fig. 2. Define the left and right lane boundaries' region of interest (ROI).

A. Sobel Edge Detection Module

The conventional structure is depicted in Fig. 3. In this structure, the image passes through Shift Registers to store the values of pixels in a row to form the matrix values of the Gx and Gy masks in the Gx, Gy Operator module. Then, it computes the orientation and edge intensity of the image through the Vectoring Cordic Module. Fig. 3 illustrates the architecture of this algorithm.

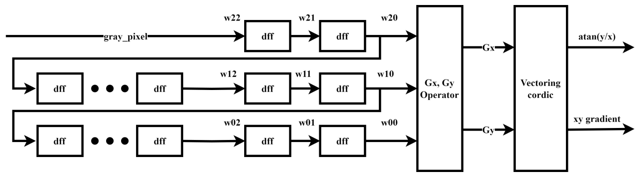

Fig. 3. Sobel edge detection.

However, utilizing Shift Registers as a First-In-First-Out (FIFO) mechanism could result in resource consumption and potentially suboptimal efficiency. As illustrated in Fig. 4, the Registers typically employed for retaining pixel rows are substituted with Memory components within the proposed Sobel Edge Detection method. Leveraging Memory resources on the FPGA offers the advantage of mitigating the routing complexities inherent in the design, consequently enabling elevated processing frequency within this module.

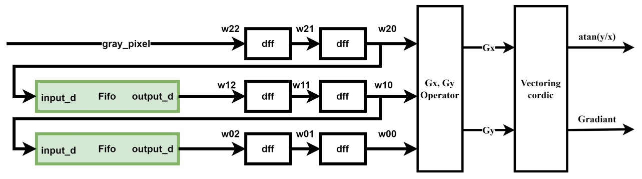

Fig. 4. Proposed Sobel Edge Detection.

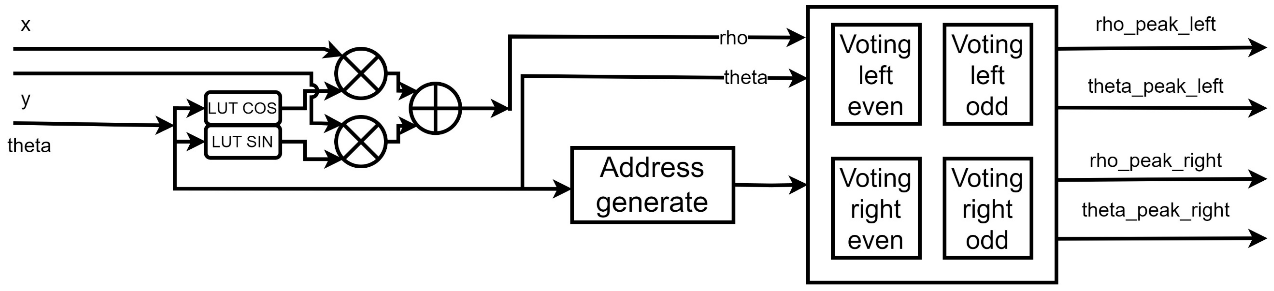

Fig. 5. Hough Transform module.

B. Hough Transform Module

In the conventional hardware design of the Hough Transform module, the authors employed the conventional method of Hough Transform computation, depicted in Fig. 5. This method involves computing ρ (rho) using the formula:

$$\rho = x * cos(\theta) + y * sin(\theta) \tag{1}$$

And utilizing the (ρ, θ) value system throughout calculations and as results. However, this approach does not include mechanisms for accelerating computations for the Inverse Hough Transform, necessitating additional processing steps after obtaining results from the hardware design. To reduce computational complexity, cos and sine trigonometric functions are computed via Look-Up Tables (LUTs) for both functions, as illustrated in Fig. 6. Notably, Sin and Cosin LUTs encompass data beyond the Region of Interest experimented with, resulting in memory consumption to store unnecessary values. Furthermore, the Voting module in the conventional design uses a Dual Port RAM as an accumulator for each image pixel identified as part of a straight line, as depicted in Fig.7. Subsequently, the (ρ, θ) values are directed to the Address generate block to furnish address values for the Voting module, thereby facilitating the voting process.

COS SIN LUT		
θ	**COS**	**SIN**
0	1	0
1	0.9998	0.0174
2	0.0993	0.0348
…	…	…
30	0.866	0.5
…	…	…
179	-0.9998	0.0174

Fig. 6. Sin LUT and Cos LUT.

Within the Voting Module, the Dual-port RAM performs concurrent accumulation and voting tasks, leveraging its versatile operational modes that seamlessly alternate between reading and writing operations. This integrated functionality optimizes resource utilization and enhances overall processing throughput within the module.

Within the Voting Port A of the Dual-port RAM, its functionality extends to retrieving the Accumulator value, subsequently updating it by assigning the value A (ρ_i, θ_i) = A (ρ_i, θ_i) + 1 into Port B. Activating wr_en permits the writing operation to Port B contingent upon detecting an edge pixel within the Sobel Edge Detection module. Upon completion of the mapping and accumulation stages for the candidate (ρ, θ) pairs, the most prominent vote within the accumulator ensemble is determined through a straightforward comparison method characterized by its simplicity and minimal computational overhead.

Upon surpassing predefined threshold criteria during the voting process, the (ρ, θ) values garnered at the output undergo comparison and are archived within a flip-flop, effectively becoming the benchmark for comparison to ascertain the maximum value among the addresses within the dual-port RAM. This selection process culminates in the derivation of the outcome. Notably, each Voting module functions autonomously in delineating between the left and right lane lines. Nevertheless, each Voting module is singularly capable of executing voting operations for individual lane lines, necessitating the deployment of two Voting modules to detect a single lane line.

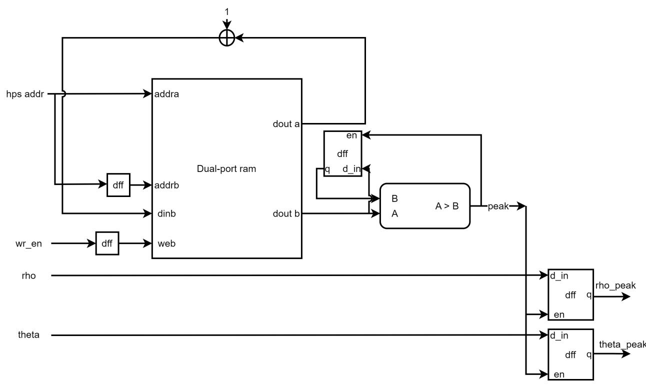

Fig. 7. Voting module.

RAM Allocation in [6]		b					
		MinB	**…**	**…**	**…**	**…**	**maxB**
θ	**0**	🟩	🟩	🟩	🟩	🟩	🟩
	1	🟩	🟩	🟩	🟩	🟩	🟩
	2	🟩	🟩	🟩	🟩	🟩	🟩
	…	🟩	🟩	🟩	🟩	🟩	🟩
	177	🟩	🟩	🟩	🟩	🟩	🟩
	178	🟩	🟩	🟩	🟩	🟩	🟩
	179	🟩	🟩	🟩	🟩	🟩	🟩

Fig. 8. Allocation Dual Port RAM.

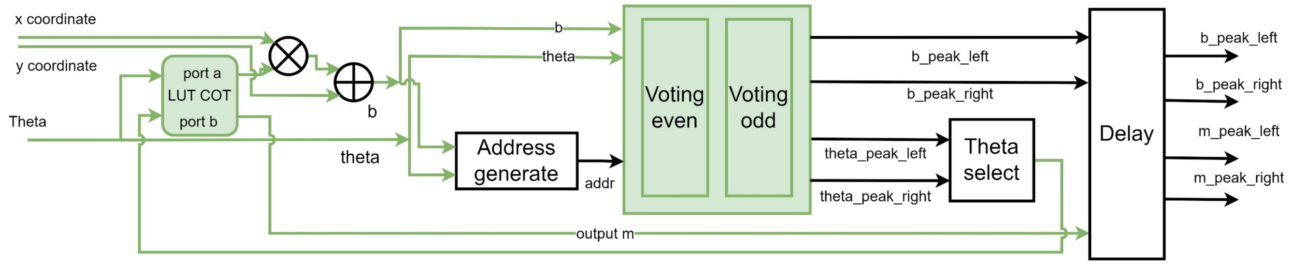

Fig. 9. Proposed Hough Transform.

The dual-port RAM in the conventional design has a memory space size from 0 to 179 * MaxRHO with MaxRHO = 750 for an image size of 1024x1024, as shown in Fig. 8. The memory space size of the dual-port RAM includes all addresses from the smallest to the largest of the (ρ, θ) value pairs and contains values that are not within the ROI, leading to unnecessary resource consumption.

Even and odd voting blocks are used alternately, with the Even Voting performing the voting while the Odd Voting is in the reset process for the next voting round, and vice versa. This allows the voting process to be carried out continuously without interruption to reset the accumulator values. Therefore, four voting modules are required to perform the Voting and reset the accumulator values alternately to continuously obtain the output of the left and right lane lines. In the conventional formula Eq. (1) of the Hough Transform algorithm, the processing of the inverse transformation, Inverse Hough Transform, has not been performed and needs to be processed to obtain the equation of the line:

$$b = x * m + y \tag{2}$$

In the proposed design, the computation of ρ will be replaced by b:

$$m = cot\theta \tag{3}$$

$$b = \rho / sin(\theta) \tag{4}$$

$$(2), (3), (4) \rightarrow b = cot(\theta) * x + y \tag{5}$$

Using this method, the hardware design also accelerates the Inverse Hough Transform. The proposed Hough Transform algorithm is described in Fig. 9. Compared to the conventional design, the Hough Transform module computes the value of b according to formula Eq. (5) to jointly enter the selection by pairs of values (b, θ) for the line. The design can compute for the process of Inverse Hough Transform with the final output being a linear equation Eq. (2). After selecting the line, the θ value is used to retrieve the (b, m) pair of results from the Dual port ROM of the COTAN trigonometric function to output. The final output value will be in the format of a linear equation. Moreover, the COTAN trigonometric function will only utilize 1 LUT instead of both Sin and Cosine trigonometric functions.

Furthermore, ROI will be utilized to reduce the resource usage of Cotan LUT, as shown in Fig. 10. The values stored inside the Cot LUT are limited to those experimented in the conventional design. The LUT will compute the values of θ from 30 degrees to 53 degrees and 130 degrees to 153 degrees to optimize resources and eliminate unnecessary values outside the ROI.

COT LUT	
θ	**COT**
30	1.732
…	…
53	0.7535
130	-0.839
…	…
153	-1.9626

Fig. 10. Cotan LUT.

In the proposed hardware design of the Hough Transform, the Voting module is reduced to 2 Voting modules: Voting even and Voting odd. Each Voting module can perform Voting for the left and right lane lines by adding a condition to differentiate the peak values for the left and right lanes. This is illustrated in Fig. 11.

In the proposed Voting module, a comparison operation is added to compare the rotation angle of the pixel. If the pixel has a rotation angle greater than 90 degrees, then it belongs to the left side, otherwise if it is less than 90 degrees, then it belongs to the right side of the lane. This way, each Voting module will handle both the right and left lanes, reducing the number of Voting Modules needed by half.

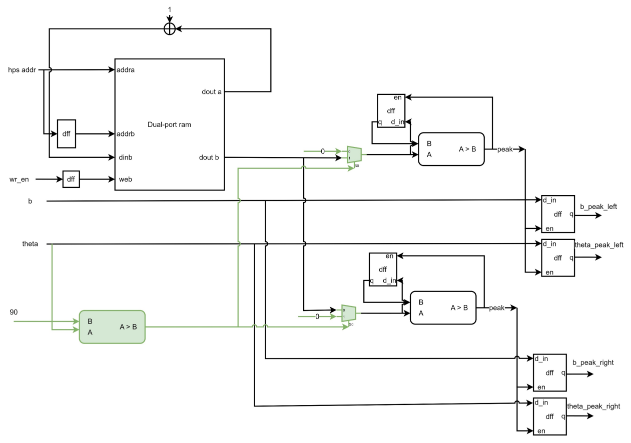

Fig. 11. Proposed voting module.

After changing from the Hough Space (ρ, θ) to the parameter space (b, θ), the maximum value in the space also changes,

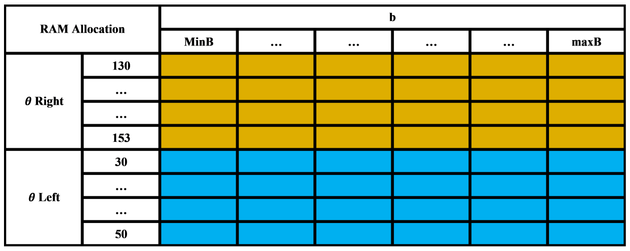

Fig. 12. Size of the proposed Dual Port RAM.

with the max being 1550. However, when applying ROI to the Hough Transform Module, the b value will range from $-860 \leq b \leq 400$ based on formula Eq. (5). The offset needs to be added to max = 400 to obtain the computation address 860, and the upper limit of b is 1260. Therefore, the Dual-port ram in the proposed design will have a memory region size of ((53-30) * (bmax+boffset) + ((153-130) * (bmax+boffset)) as shown in Fig. 12 and eliminate unused address regions outside the ROI.

The amount of memory used by the Hough Transform will be reduced to only 2% compared to the conventional design, and in the HOUGH TRANSFORM Module of the proposed design, only 2 VOTING modules are required. This reduces the overall RAM size of the proposed design to approximately 1.044% compared to the conventional design.

III. Verification

A. Synthesized Result

The architecture is synthesized on the Virtex-7 VC707 FPGA platform using the Vivado Design Suite program based on the proposed architecture method. Table I shows the resource consumption for the proposed hardware architecture of the Hough Transform, benefiting from method simplification, transforming the Hough Space into (b, θ), and applying ROI rigorously. The architecture utilizes 0.82% LUT and 482 Kbit memory (approximately 0.37%) for the Cotan Look-up table. BRAM usage accounts for about 3.5% of the Voting process.

TABLE I. Resource Consumption of the Proposed Architecture

Resource	Synthesis	Implementation
Board	Virtex-7 VC707	Virtex-7 VC707
LUTs	2498/303600 (0.82%)	2483/303600 (0.82%)
LUTRAM	482/130800 (0.37 %)	482/130800 (0.37 %)
FF	3243/607200 (0.53%)	3222/607200 (0.53%)
BRAM	36 /1030 (3.5%)	36/1030 (3.5%)
DSP	1/2800 (0.04%)	1/2800 (0.04%)
IO	89/700 (12.71 %)	89/700 (12.71 %)
BUFG	1/32(3.13%)	1/32(3.13%)

The achieved processing speed after synthesis (see Table II):

- Image resolution: 1024x1024
- Processing frequency: 250 MHz
- Processing speed (ms/frame): 4.19 ms

Table III illustrates a detailed comparison of resource utilization between our FPGA implementation system and other studies. Although our architecture utilizes different devices compared to other studies, overall, the hardware resources of LUTs we use are relatively small. The architecture of [4] employs a small FPGA generation, with 0.8% LUTs, 9.36% memory, and 3.72% DSP. In [5], resource usage comprises 17% LUTs, 49% memory, and 3% FF. However, it utilizes more resources for DSP, at 59.77%. Authors in the study [6] implement the Hough space using (Y-intercept, θ) and require 6.9% LUTs, 5.6% memory, and 15.54% slices without using DSP. Additionally, 5.18% LUTs, 9.34% slices, and 3.71% FF of resources are utilized in [7]. This system uses a significant number of resources for BRAM, at 66.67%. In [8], 5.83%, 3% slices, and 0.75% memory are utilized. Regarding BRAM, it uses 31%, 0.67% FF, and only 0.07% DSP.

B. Verification and Evaluation

To validate the hardware design of the Hough Transform system, a simulation model was implemented using the Verilog hardware description language, and the Hough Transform IP was simulated on the Vivado Design Suite, as shown in Fig. 13. The evaluation results will be based on the same validation

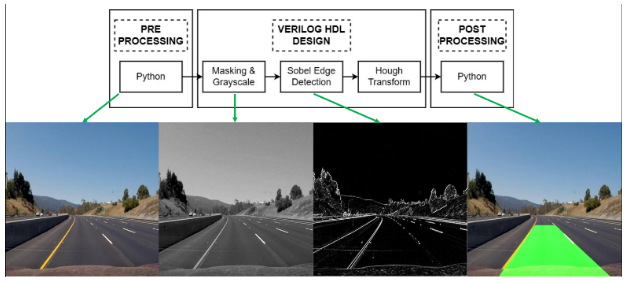

Fig. 13. Verification model.

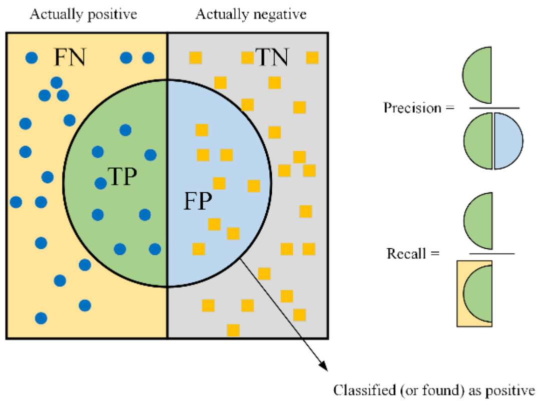

Fig. 14. Precision and recall.

dataset used in the study [8]. The videos are processed and converted into text files, then simulated using the Vivado Design Suite application. The resulting data is saved in text files, and Python processes the output values. These output values will be used to draw the lane lines.

This paper utilizes the Precision, Recall, and F1-score evaluation systems to ensure comprehensive and objective

TABLE II. Results of our Work Comparison with Different Architectures

	[4]	[5]	[6]	[7]	[8]	**Our architecture**
Image Resolution	1024x768	640x480	640x480	1024x1024	1024x1024	1024x1024
Fmax (MHz)	200	50	200	145	170	250
Processing Speed (ms/frame)	5.4	7.4	1.47	9.03	6.17	4.19
Normallized Speed (ns/pixel)	6.8	24.08	4.78	8.61	5.88	4

TABLE III. Resource Requirements of our System Comparison with Different Architectures

Resource usage	[4]	[5]	[6]	[7]	[8]	**Our architecture**
Device	Altera Stratix IV	Cyclone II FPGA	Virtex-5 ML505	Xilinx xc7z001-1	Virtex-7 VC707	Virtex-7 VC707
LUTs	1459	5460	1996	911	4551	2483
Slices	5115	–	1119	411	2275	1200
Memory (Kbit)	1604	1985	1625	–	986	482
BRAM	–	–	–	40	320	36
FF	–	5781	–	1307	4215	3222
DSP	48	52	0	0	2	1

TABLE IV. The Accuracy Results of the Simulation

		[8]			Our work		
Road type	Number of frame	Precision	Recall	F-score	Precision	Recall	F-score
Normal	1260	96.65%	98.46%	97.55%	93.14%	90.67%	90.54%
Poor condition	1802	97.31%	98.12%	97.65%	87.18%	98.53%	91.50%
Urban road	1810	83.61%	96.44%	88.41%	63.55%	96.37%	74.43%
Total	**4872**	**92.53%**	**97.67%**	**94.54%**	**81.29%**	**95.19%**	**85.49%**

evaluation. Precision, known as positive predictive value, measures the accuracy of the positive predictions. Recall, also referred to as sensitivity in binary classification, measures the proportion of actual positives that are correctly identified. F1-score is the harmonic mean of Precision and Recall (assuming both values are non-zero). Its value ranges from 0 to 1 and is defined as follows:

$$Precision = \frac{TN}{TN + FN} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8}$$

where, True Positive (TP) is the result where the model correctly predicts the positive class, True Negative (TN) is the result where the model correctly predicts the negative class. FP (False Positive) results in the model incorrectly predicting the positive class. FN (False Negative) is the result where the model incorrectly predicts the negative class. Fig. 14 illustrates an example of Precision and Recall results.

The evaluation results performed on the dataset including three videos, are depicted in Table IV. Testing results were conducted on multiple videos under various lighting and road conditions, including urban streets, highways, road conditions, coverage, poor road markings, day and night scenes. By comparing images, it is evident that the successfully deployed architecture accurately detects straight lanes.

The results evaluated on the dataset from study [6], consisting of 3 videos, are presented in Table IV. The testing results on multiple videos with varying lighting and road conditions, including urban streets, highways, road conditions, coverage, poor road markings, day and night scenes, were conducted. By comparing images, it can be observed that the successfully deployed architecture detects straight lanes. The comparative results indicate that the conventional architecture accurately detects straight lanes under different lighting and road conditions using metrics derived from these four results. Their average accuracy rates are approximately 92.53%, 97.67%, and 94.54%, respectively. Our hardware architecture's accuracy rates are 81.29%, 95.19%, and 85.49%, respectively. It is noted that there is a significant decrease in accuracy in our proposed architecture.

IV. Conclusions

This paper introduces a lane detection system for autonomous vehicles. The algorithm utilizes Gray Scale, Sobel Edge Detection, and Hough Transform methods. The hardware architecture is designed using the Verilog hardware description language. The proposed architecture implements a rigorous Region of Interest (ROI) approach to reduce hardware resource usage and algorithmic enhancements to reduce processing load for Inverse Hough Transform. The research aims to achieve fast processing speed through ROI implementation, which is capable of processing approximately 4.19ms per frame with a resolution of 1024x1024 and a frequency of 250MHz. When synthesized on the Virtex-7 VC707 FPGA board, the system achieves an accuracy of 85.49%. Although the real-time processing speed is achieved, the detection rate is relatively low. Therefore, we will explore how to improve the detection rate by applying the learning machine in our system for the future work.

Acknowledgment

This research is funded by Vietnam National University Ho Chi Minh City (VNU-HCM) under grant number DS2023-26-02.

References

[1] Yam-Uicab, R., Lopez-Martinez, J.L., Trejo-Sanchez, J.A. et al. A fast Hough Transform algorithm for straight lines detection in an image using GPU parallel computing with CUDA-C. J Supercomput 73, 4823–4842 (2017). https://doi.org/10.1007/s11227-017-2051-5.

[2] D. Qiu, M. Weng, H. Yang, W. Yu and K. Liu, ”Research on Lane Line Detection Method Based on Improved Hough Transform,” 2019 Chinese Control And Decision Conference (CCDC), Nanchang, China, 2019, pp. 5686-5690, doi: 10.1109/CCDC.2019.8833139.

[3] S. Luo and X. Zhao, ”Application of improved Hough transform in lane line detection,” 2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, China, 2022, pp. 1717-1721, doi: 10.1109/ITAIC54216.2022.9836543.

[4] Guan, Jungang, Fengwei An, Xiangyu Zhang, Lei Chen, and Hans Jürgen Mattausch. 2017. ”Real-Time Straight-Line Detection for XGA-Size Videos by Hough Transform with Parallelized Voting Procedures”, Sensors, vol. 17, no. 2: 270. https://doi.org/10.3390/s17020270.

[5] Guan, J., F. An, X. Zhang, Lei Chen and H. Mattausch. “Energy-Efficient Hardware Implementation of Road-Lane Detection Based on Hough Transform with Parallelized Voting Procedure and Local Maximum Algorithm.”, IEICE Transactions on Information and Systems, 2019. vol. E102.D, no. 6, pp. 1171-1182. https://doi.org/10.1587/transinf.2018EDP7279.

[6] El Hajjouji, Ismaïl Mars, Salah Asrih, Zakariae El Mourabit, ”A novel FPGA implementation of Hough Transform for straight lane detection,” International Journal of Engineering Science and Technology, 2019, vol. 23, https://doi.org/10.1016/j.jestch.2019.05.008.

[7] D. Northcote, L. H. Crockett and P. Murray, ”FPGA Implementation of a Memory-Efficient Hough Parameter Space for the Detection of Lines,” 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 2018, pp. 1-5, doi: 10.1109/ISCAS.2018.8351115.

[8] Lam, D.K., Dinh, P.T.L., Ngoc Diem Nguyen, T. (2023). Hardware-Based Lane Detection System Architecture for Autonomous Vehicles. In: Dao, NN., Thinh, T.N., Nguyen, N.T. (eds) Intelligence of Things: Technologies and Applications. ICIT 2023. Lecture Notes on Data Engineering and Communications Technologies, vol 188. Springer, Cham. https://doi.org/10.1007/978-3-031-46749-3_4.

[9] Y. Kortli, M. Marzougui, B. Bouallegue, J. S. C. Bose, P. Rodrigues and M. Atri, ”A novel illumination-invariant lane detection system,” 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), Abha, Saudi Arabia, 2017, pp. 166-171, doi: 10.1109/Anti-Cybercrime.2017.7905284.

[10] Y. Wang, L. Shi, J. Lausanne and D. Zhong, ”Straight lane line detection based on the Otsu-Canny algorithm,” 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2022, pp. 27-30, doi: 10.1109/ITOEC53115.2022.9734320.

[11] X. Zhou, Y. Ito and K. Nakano, ”An Efficient Implementation of the Gradient-Based Hough Transform Using DSP Slices and Block RAMs on the FPGA,” 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, Phoenix, AZ, USA, 2014, pp. 762-770, doi: 10.1109/IPDPSW.2014.88.

[12] Zhang, Z.C. and Ma, X., ”Lane Recognition Algorithm Using the Hough Transform Based on Complicated Conditions,” Journal of Computer and Communications, vol.7, pp. 65-75, https://doi.org/10.4236/jcc.2019.711005.

[13] A. Istiningrum, U. Salamah and N. P. Taufik Prakisya, ”Lane Detection With Conditions of Rain and Night Illumination Using Hough Transform,” 2022 5th International Conference on Information and Communications Technology (ICOIACT), Yogyakarta, Indonesia, 2022, pp. 429-434, doi: 10.1109/ICOIACT55506.2022.9972068.

[14] T. Manabe et al., ”Autonomous Vehicle Driving Using the Stream-Based Real-Time Hardware Line Detector,” 2019 International Conference on Field-Programmable Technology (ICFPT), Tianjin, China, 2019, pp. 461-464, doi: 10.1109/ICFPT47387.2019.00093.

[15] K. V. Pachkor and V. Arunachalam, ”Memory Efficient ASIC Implementation of Line Hough Transform,” 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2018, pp. 718-723, doi: 10.1109/RTEICT42901.2018.9012298.