



Hardware Implementation of Hough Transform for the Application in Lane Detection in Smart Vehicles

Chessda Uttraphan^{1,3*}, Dhivaakar Ravindran¹, Chua Wee Heng¹, Kok Boon Ching², Nabihah Ahmad^{1,3}, A Arul Edwin Raj⁴

¹ Department of Electronic Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia

² Department of Electrical Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia

³ VLSI and Embedded System Technology (VEST) Research Group

⁴ Department of ECE, Saveetha Engineering College, Chennai, India

ABSTRACT

Lane detection is one of the important features of smart vehicles. It is used to assist drivers in achieving the best driving experience. Lane detection utilizes the line detection algorithm where there are many algorithms that are available, but the most effective algorithm is the Hough Transform because it is simple and can be applied in both software and hardware implementations. However, studies have shown that Hough Transform implementation of video in the software environment could result in sub-par performance because it requires extremely high computation resources and memory. Therefore, we propose hardware implementation of the Hough Transform for lane detection in this work. The targeted hardware is the Field Programmable Logic Array (FPGA) as its reconfigurable nature allows for rapid design. Hardware implementation of video processing enables parallel data processing, which reduces overall system latency. Furthermore, the hardware design can be optimized to reduce the number of logics that will lead to lower power consumption. The hardware logic was designed based on the Hough Transform equation by using the Verilog Hardware Description Language (HDL) in Intel Quartus Prime software. After the design is successfully completed and verified through simulation, the execution speed of the hardware implementation is then compared with the same design in the software environment (MATLAB). Results show that the hardware-based Hough Transform is over 100 times faster, with less than 1% of logic resources utilized, resulting in a lower power consumption at 146 mW.

Keywords:

Hough transform; straight line detection; lane detection; FPGA; hardware implementation

1. Introduction

Lane detection is a crucial component of modern autonomous driving systems, enabling vehicles to stay within their designated lanes and navigate safely on the road [1]. To achieve this, computer vision algorithms are employed to identify the lane markings on the road and track them

* Corresponding author.

E-mail address: chessda@uthm.edu.my

<https://doi.org/10.37934/araset.XX.X.XX>

in real-time. One popular technique used for this purpose is the Hough transform, which is a mathematical method that can detect lines in an image or video even if they are broken or distorted [2].

The Hough transform was first introduced by Paul Hough in 1962, and it has since been widely used in computer vision and image processing applications [3]. The technique works by transforming an image into a parameter space, where lines can be represented as points. By searching for clusters of points in this parameter space, the Hough transform can identify lines in the original image.

In lane detection, the Hough transform is used to detect the straight lines that make up the lane markings. These lines can then be used to determine the position and orientation of the vehicle within its lane. The Hough transform is particularly useful in this application because it can handle a variety of lane marking types, such as solid lines, dashed lines, and even curved lines [4].

In the previous study, Kumar *et al.*, [5] proposed effective and efficient vision-based real-time lane markings and tracking lane detection methods for straight and curved lane lines. They claimed that the proposed approaches are able to achieve real-time response and high accuracy for a vehicle in lane change assistant system on highways. A better technique was proposed in [6, 7] where shadows and low light conditions are taken into account. They used edge detection and gradient techniques to process the images before applying the Hough Transform. A more advanced approach was reported in [8] where the work focused in the pre-processing stage. They proposed smoothing and edge detection operators that applied on input frames to automatically obtain binary images, then, lane markings segmentation is carried out. After that, An Adaptive Region of Interest (AROI) is extracted to reduce the computational complexity. Other works on lane detection using Hough Transform can be found in [9-12]. Deep learning approaches for lane detection were also have been reported in [13-16]. However, this technique is not on our focus as it usually increases computation complexity.

FPGA devices can also achieve low latency, as they do not require the overhead of an operating system or additional software layers [17, 18]. This makes them suitable for applications where real-time processing is critical, such as lane detection in autonomous vehicles. By using FPGA to implement the Hough transform, the latency can be significantly reduced, resulting in more timely and accurate detection of lane markings. Furthermore, FPGA devices are known for their energy efficiency, as they can be programmed to consume only the power needed for the task at hand [19]. This is important for battery-powered devices, such as smart vehicles, where energy consumption is a critical consideration. By implementing the Hough transform on an FPGA, the power consumption can be minimized, resulting in longer battery life, and reduced operating costs.

As stated earlier, Hough Transform needs extremely high computation resources and huge memory space because the computation to calculate Hough parameters are repeated for each image pixel and for all angles [20]. Therefore, recently many works have proposed the hardware implementation of the Hough Transform in lane detection applications. Hardware implementation such as FPGAs are highly parallel and can perform multiple computations simultaneously. This makes them well-suited for processing image data, where multiple calculations need to be performed on large amounts of data in real-time. By using FPGA, the Hough transform algorithm can be parallelized and executed much faster than on traditional CPUs or GPUs, resulting in improved real-time performance and reduced latency. The programmability of FPGAs allows the system designers to configure the design to meet specific requirements. The Hough transform algorithm can be implemented on an FPGA by designing custom hardware circuits that are optimized for the algorithm's specific needs. This enables the algorithm to be tailored for a specific application or task, resulting in improved efficiency and accuracy.

For example, Hajjouji *et al.*, [21] proposed a novel FPGA implementation of Hough Transform for straight lane detection. The design in [22] is claimed to have less logics with reduced overall complexity. However, the algorithm is to detect the circle lines instead of straight line. Other hardware implementation of Hough Transform for lane detection were reported in [23-26]. Although there were many works that have explored the possibility of implementing Hough Transform on the hardware with promising performance improvement as compared to the software implementation counterparts. However, if we carefully analyze the previous works, there is still much room for improvement. In this work, we propose a unique design of the hardware architecture for executing the Hough Transform algorithm. The design can achieve faster speed and lower power consumption as compared to the software implementation.

This paper is organized as follows: Section 1 introduces the implementation of Hough Transform in the lane detection and discusses related works with the current implementation issues. Section 2 discusses the fundamentals of Hough Transform Algorithm with the aid of numerical examples together with the proposed designed based on the algorithm, while Section 3 provides the experimental works, results, and analysis. Finally, Section 5 concludes the proposed work and discusses the future direction of this work.

2. Methodology

The Hough Transform equation, Eq. (1) describes the relationship between ρ and θ , two essential parameters used to detect straight lines in a 2D plane. ρ represents the distance from the origin to the closest point on the identified line, while θ denotes the angle between the x-y axis and the line connecting the origin with that closest point as illustrated in Figure 1.

$$\rho = x \cdot \cos \theta + y \cdot \sin \theta \quad (1)$$

The linear Hough transform algorithm uses the two parameters that define a straight line, (ρ, θ) . Consider the 2-D plane in Figure 2, where the size is 7×6 pixels.

Suppose the edge pixels provided by the Sobel filter are as follows: (0, 5), (1, 4), (2, 3), (3, 2), (3, 5), (4, 1), (4, 3) and (5, 5). By using Eq. (1), ρ for θ between 0° to 180° of each edge pixel can be calculated. The example of the calculation for θ incremented by 45° is given in Table 1 (note that this is the simplified calculation, where in practice, the smaller the θ resolution, the more accurate the line detection). From Table 1, it can be observed that the most ρ value detected is 3.54, which indicates that the pixels (0, 5), (1, 4), (2, 3), (3, 2) and (4, 1) must be on the same straight line. All those pixels point to the same θ value, which is 45° . That's exactly what is illustrated in Figure 3. To implement this, the Hough Transform algorithm can now be created as given in Figure 4. In the algorithm, for each given pixel, ρ for θ from 0° to 180° will be calculated. The value of the calculated ρ will be counted in the accumulator. The most hit ρ value indicates the straight line at the pixel with specific θ angle.

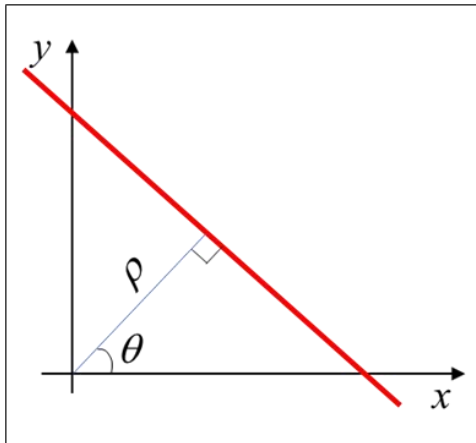


Fig. 1. A line on 2-D plane

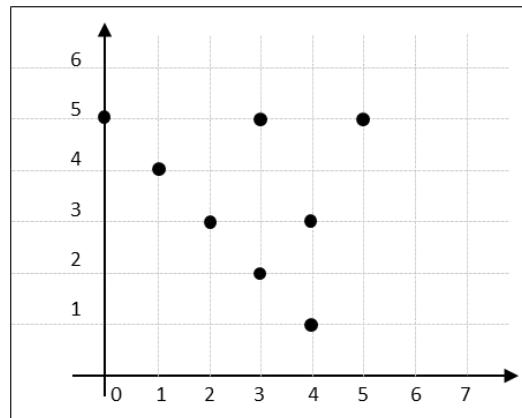


Fig. 2. Edge pixels of an image

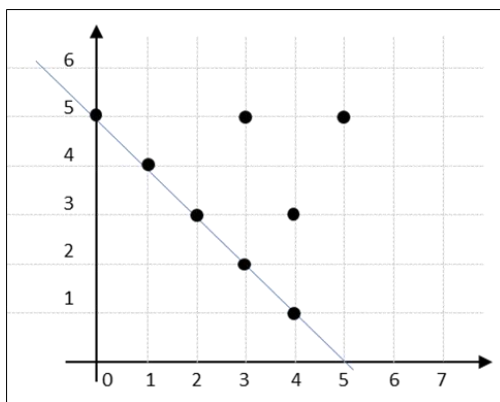


Fig. 3. Straight line construction by Hough Transform

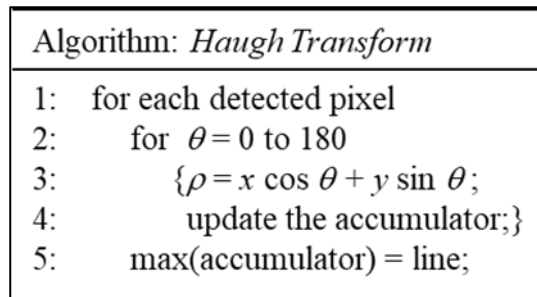


Fig. 4. Simplified Hough Transform algorithm

Table 1

Hough Transform calculation example

Pixel	0°	45°	90°	135°	180°
(0, 5)	0.00	3.54	5.00	3.54	0.00
(1, 4)	1.00	3.54	4.00	2.12	-1.00
(2, 3)	2.00	3.54	3.00	0.71	-2.00
(3, 2)	3.00	3.54	2.00	-0.71	-3.00
(3, 5)	3.00	5.66	5.00	1.41	-3.00
(4, 1)	4.00	3.54	1.00	-2.12	-4.00
(3, 3)	3.00	4.24	3.00	0.00	-3.00
(5, 5)	5.00	7.07	5.00	0.00	-5.00

The hardware logic implementation of the Hough Transform is based on Eq. (1) and the algorithm depicted in Figure 4. Figure 5 illustrates the proposed hardware design for the Hough Transform, which is modelled after the algorithm presented in Figure 4. The detected pixels, representing (x, y) coordinates, are stored in dedicated Random-Access Memories (RAMs) called X Pixel and Y Pixel, respectively. Additionally, the cosine and sine values required for θ calculations are stored in lookup tables (LUTs) implemented as Read-Only Memories (ROMs) named Cos and Sin. For each pixel, the corresponding ρ values are calculated for θ angles ranging from 0 to 180 degrees, and these values are accumulated in the ρ Accumulator RAM (Acc Rho). As demonstrated in the numerical example, the highest ρ value at a specific θ angle indicates the presence of a straight line,

as shown in Figure 3. To ensure synchronized operations in the proposed design, a controller is employed to generate the necessary control signals. The controller's arithmetic state machine (ASM), which models its behaviour, is provided in Figure 6.

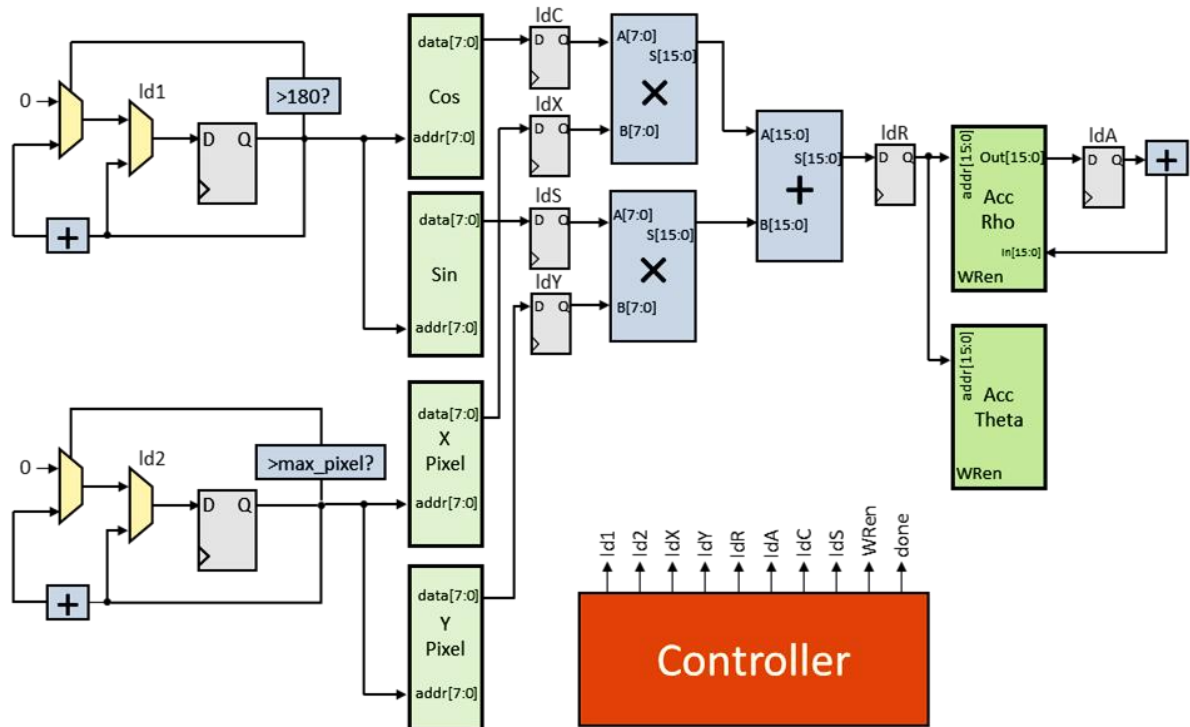


Fig. 5. The proposed hardware implementation of the Hough Transform

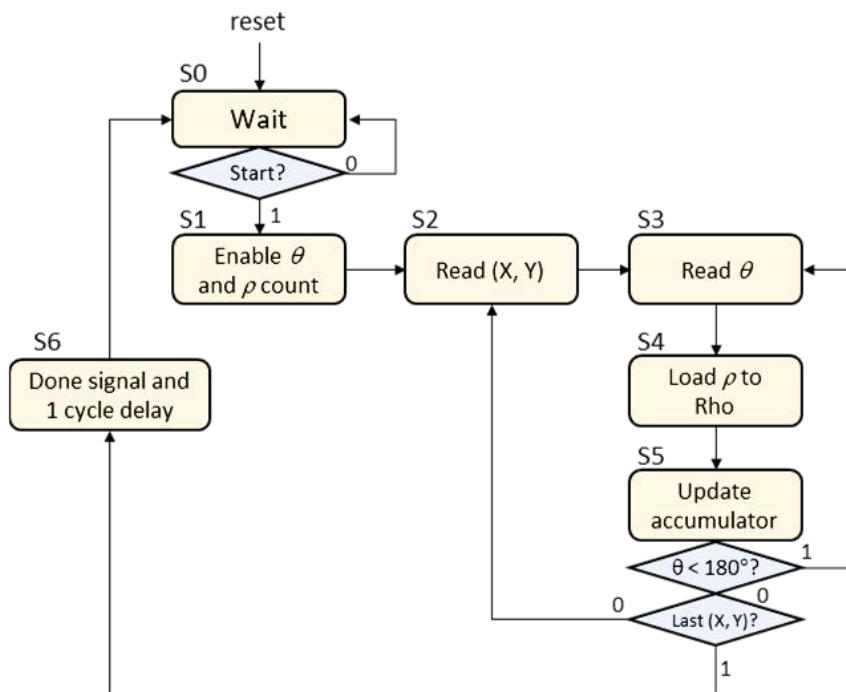


Fig. 6. ASM chart of the controller

The sine and cosine values, represented by 8-bit signed numbers, are initialized in the ROM using memory initialization files (*mif* files). The specific contents of the ROMs can be seen in Figure 7. Furthermore, the synthesized top-module level of the proposed design is provided in Figure 8.

Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13	+14	+15	+16	+17	+18	+19	+20	+21	+22	+23	+24	+25	+26	+27	+28	+29	+30	+31
0	00	02	04	07	09	0B	0D	0F	12	14	16	18	1A	1D	1F	21	23	25	27	29	2B	2E	30	32	34	36	38	3A	3C	3E	40	41
32	43	45	47	49	4B	4C	4E	50	52	53	55	57	58	5A	5B	5D	5E	60	61	63	64	65	67	68	69	6B	6C	6D	6E	6F	70	71
64	72	73	74	75	76	77	77	78	79	79	7A	7B	7B	7C	7C	7D	7D	7E	7E	7E	7E	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F	7F
96	7E	7E	7E	7D	7D	7D	7C	7C	7B	7B	7A	79	79	78	77	77	76	75	74	73	72	71	70	6F	6E	6D	6C	6B	69	68	67	65
128	64	63	61	60	5E	5D	5B	5A	58	57	55	53	52	50	4E	4C	4B	49	47	45	43	41	40	3E	3C	3A	38	36	34	32	30	2E
160	2B	29	27	25	23	21	1F	1D	1A	18	16	14	12	0F	0D	0B	09	07	04	02	00	00	00	00	00	00	00	00	00	00	00	
192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

(a)

Addr	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13	+14	+15	+16	+17	+18	+19	+20	+21	+22	+23	+24	+25	+26	+27	+28	+29	+30	+31
0	7F	7F	7F	7F	7F	7E	7E	7E	7D	7D	7D	7C	7C	7B	7B	7A	79	79	78	77	77	76	75	74	73	72	71	70	6F	6E	6D	
32	6C	6B	69	68	67	65	64	63	61	60	5E	5D	5B	5A	58	57	55	53	52	50	4E	4C	4B	49	47	45	43	41	40	3E	3C	3A
64	38	36	34	32	30	2E	2B	29	27	25	23	21	1F	1D	1A	18	16	14	12	0F	0D	0B	09	07	04	02	00	FE	FC	F9	F7	F5
96	F3	F1	EE	EC	EA	E8	E6	E3	E1	DF	DD	DB	D9	D7	D5	D2	D0	CE	CC	CA	C8	C6	C4	C2	C0	BF	BD	BB	B9	B7	B5	B4
128	B2	B0	AE	AD	AB	A9	A8	A6	A5	A3	A2	A0	9F	9D	9C	9B	99	98	97	95	94	93	92	91	90	8F	8E	8D	8C	8B	8A	89
160	89	88	87	87	86	85	85	84	84	83	83	83	82	82	82	81	81	81	81	81	81	81	00	00	00	00	00	00	00	00	00	00
192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

(b)

Fig. 7. 8-bit signed values in ROMs (a) Sine (b) Cosine

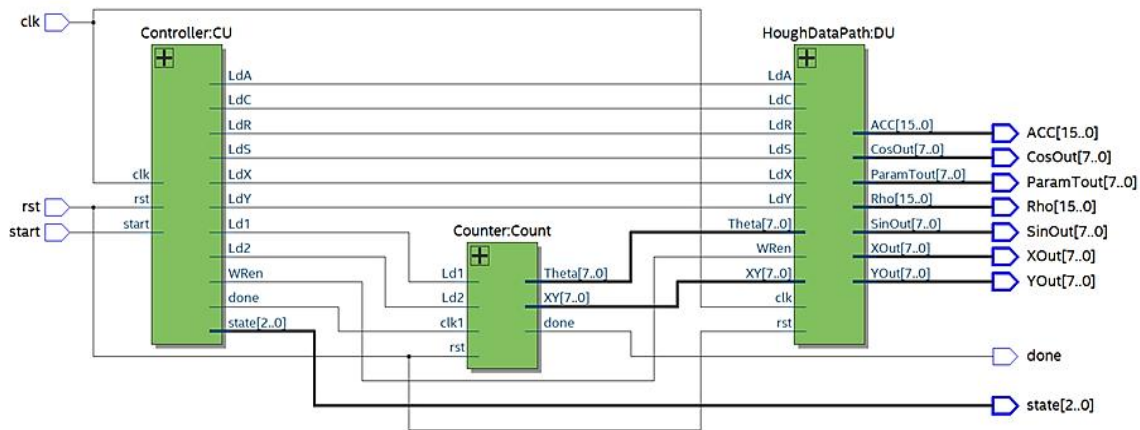


Fig. 8. The synthesized top-level module of the proposed hardware Hough Transform

3. Results

3.1 Experiment Setup

The proposed hardware design depicted in Figure 5 was implemented in Verilog HDL using the Intel Quartus Prime development tool. The design was synthesized using Quartus, which allowed us to record important metrics such as the total number of logic elements, timing information including the maximum operating frequency (F_{max}), and the total power consumption. To evaluate the performance of the design, we conducted simulations using ModelSim. The execution speed of the design was determined by referring to the obtained F_{max} . Multiple tests were carried out, each with different image (line) settings. It is worth noting that, in this initial phase of the work, we utilized constructed images in which the edge pixels were manually generated, as illustrated in Figure 7. This approach was chosen instead of using actual images as the latter would require preprocessing using modules such as the Sobel Filter. The inclusion of such preprocessing modules is planned for our future work. The objective of this study was primarily to demonstrate the efficient hardware implementation of the Hough Transform. Once the design was successfully simulated and the execution speed was obtained, we compared this speed with the execution time

of the same settings in MATLAB. This comparison allowed us to assess the effectiveness and efficiency of our hardware implementation.

3.2 Experimental Results and Analysis

3.2.1 Verification results

To ensure the correctness of the design, a thorough verification process was conducted using an input image depicted in Figure 3. The simulation results of the verification test for the proposed design are presented in Figure 9, with the corresponding output values tabulated in Table 2. In the simulation results, the sine and cosine values were represented as 8-bit signed values. For instance, for $\theta = 45^\circ$, the sine value ($\sin \theta$) and cosine value ($\cos \theta$) both equal 90 or 0x5A in the 8-bit representation. Analyzing the obtained results, specifically for the pixel coordinates $(x, y) = (0, 5)$, we observe the calculated ρ values for θ angles of 0, 45, 90, 135, and 180 degrees to be 0, 450, 635, 450, and 0, respectively. Examining the calculated ρ values for all pixels, it becomes evident that the ρ value of 450 is the most frequently stored value in the RAM. Based on this observation, we can conclude that pixels (0, 5), (1, 4), (2, 3), (3, 2), and (4, 1) lie on the same straight line, while pixels (3, 5), (4, 3), and (5, 5) do not. This outcome provides compelling evidence that the proposed design operates correctly, aligning with the principles of the Hough Transform algorithm.

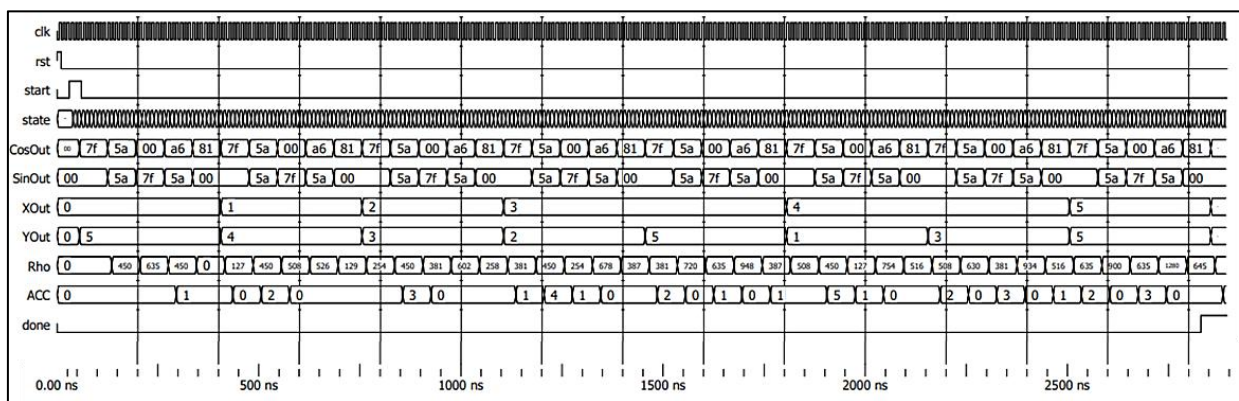


Fig. 9. Simulation result of the verification test

Table 2

Output of the proposed hardware Hough Transform

θ	$\cos(\theta)$	$\sin(\theta)$	x	y	ρ	x	y	ρ	x	y	ρ	x	y	ρ	x	y	ρ
0	127	0	0	5	0	1	4	127	2	3	254	3	2	381	3	5	381
45	90	90	0	5	450	1	4	450	2	3	450	3	2	450	3	5	720
90	0	127	0	5	635	1	4	508	2	3	381	3	2	254	3	5	635
135	-90	90	0	5	450	1	4	526	2	3	602	3	2	678	3	5	948
180	-127	0	0	5	0	1	4	129	2	3	258	3	2	387	3	5	387

Table 2. Continued

Output of the proposed hardware Hough Transform

θ	$\cos(\theta)$	$\sin(\theta)$	x	y	ρ	x	y	ρ	x	y	ρ
0	127	0	4	1	508	4	3	508	5	5	635
45	90	90	4	1	450	4	3	630	5	5	900
90	0	127	4	1	127	4	3	381	5	5	635
135	-90	90	4	1	754	4	3	934	5	5	1280

3.2.2 Experimental results

The proposed hardware implementation of the Hough Transform was subjected to testing using various constructed images. These images were created using PIXILART software, with each image set to a size of 100×100 pixels. In Table 3, we present both the original images and the output generated by the proposed hardware design, showcasing successful construction of the straight lines in each image. To compare the performance of the hardware implementation with the software counterpart, the execution times for each image were recorded for both MATLAB and the hardware design. These results are summarized in Table 4. It should be noted that while the execution time for the hardware implementation remains fixed, the software execution time can vary due to the computer load during algorithm execution. Across all the test cases, it is evident that the hardware execution outperforms the software execution in terms of speed, achieving up to a 100-fold improvement. This is remarkable considering that the hardware implementation operates at a much lower frequency of 50 MHz, in contrast to the 3.2 GHz clock speed of a personal computer. These findings affirm the superior efficiency and effectiveness of the proposed hardware design for the Hough Transform.

Table 3
 Experimental results for the test in different constructed images

Original image	Line constructed	Original image	Line constructed

Table 3. Continued

Experimental results for the test in different constructed images

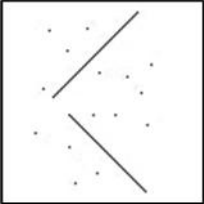

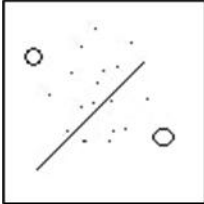
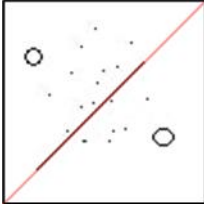
Original image	Line constructed	Original image	Line constructed
			

Table 4

Execution time test results

Image	Execution time (s) in MATLAB	Execution time (s) in hardware implementation
Image D1	0.083178	0.000803
Image D2	0.082436	0.000803
Image D3	0.080723	0.000803
Image D4	0.081263	0.000803
Image D5	0.078437	0.000803
Image D6	0.080891	0.000803
Image D7	0.077648	0.000803
Image D8	0.086015	0.000803
Image D9	0.079168	0.000803
Image D10	0.079433	0.000803

Table 5 presents a benchmark comparison of our work with references [24-26]. Notably, our proposed Hough Transform hardware design showcases the lowest utilization of logic elements, amounting to only 237 4-input lookup tables (LUTs). Consequently, it achieves the lowest total power consumption, measured at 146.29 mW. Regarding the maximum operating frequency (F_{max}), our design achieves a clock rate of up to 128.9 MHz. It is important to note that direct comparisons with [24, 25] are not feasible due to the use of different FPGA devices in those works, thereby influencing the obtained F_{max} values. Memory utilization is dependent on the built-in memory employed within our design. As such, it is not directly comparable to the memory utilization in the referenced works.

Table 5

Performance comparison with other works

Performance metrics	Proposed work	[24]	[25]	[26]
Maximum operating frequency	128.9 MHz	69.2 MHz	226.7 MHz	200 MHz
Total logic elements	237	278	5,717	15,704
Total registers	87	431	6,010	13,727
Total memory bits	1,581,056	38,280	-	3,052,544
Total thermal power dissipation	146.29 mW	-	-	640.89 mW

4. Conclusions

In this study, we have introduced a novel implementation of the Hough Transform algorithm on an FPGA. The results demonstrate that the proposed design achieves a significant speedup of up to 100 times compared to the software implementation in MATLAB, despite running at a lower clock speed on a slower platform. This is due to the parallel execution of the computation modules,

which results in faster processing times. Moreover, the design consumes low power at 146 mW, utilizing less than 1% of the total logic elements of the FPGA device. We also show that the proposed design utilizes the lowest logic elements as compared to other similar works. However, due to the large size of the RAM accumulator, RAM for storing image pixels, and LUTs (ROMs) for θ , sine, and cosine values, the design utilized a total of 66% memory bits. While the improved speed was achieved, the proposed design still requires multiple iterations of calculating ρ . In future work, we aim to explore other logic design configurations to allow for more parallel computation by applying proper operation scheduling and constraints resources allocation. We also plan to optimize the design by using logic transformation techniques, such as replacing multiplication with shift operations, to reduce the overall cost. These efforts will further enhance the performance of the proposed design and make it suitable for a broader range of applications in computer vision and image processing.

Acknowledgement

This research was funded by a grant from Universiti Tun Hussein Onn Malaysia (UTHM) through GPPS (vot. Q288).

References

- [1] Ma, Long Yang, Hao Zhu, and Hong Duan. "A method of multiple lane detection based on constraints of lane information." In *2021 China Automation Congress (CAC)*, p. 4059-4064. IEEE, 2021. <https://doi.org/10.1109/CAC53003.2021.9727491>
- [2] Jiang, Yan, Feng Gao, and Guoyan Xu. "Computer vision-based multiple-lane detection on straight road and in a curve." In *2010 International Conference on Image Analysis and Signal Processing*, p. 114-117. IEEE, 2010. <https://doi.org/10.1109/IASP.2010.5476151>
- [3] Fokkinga, Maarten. "The hough transform." *Journal of functional programming* 21, no. 2 (2011): 129-133. <https://doi.org/10.1017/S0956796810000341>
- [4] Dharsini, Ms Visnu, K. Karthik, M. Gopichandd, Jorige Venkatesh, and S. Sabarish. "Advanced road lane line detection." In *2022 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI) 1*, p. 1-5. IEEE, 2022. <https://doi.org/10.1109/ICDSAAI55433.2022.10028920>
- [5] Kumar, Sunil, Manisha Jailia, and Sudeep Varshney. "An efficient approach for highway lane detection based on the Hough transform and Kalman filter." *Innovative infrastructure solutions* 7, no. 5 (2022): 290. <https://doi.org/10.1007/s41062-022-00887-9>
- [6] Weganofa, Riza, Ayu Liskinasih, Gunadi Harry Sulisty, and Punadji Setyosari. "Bridging the expectation and actual learning." *International Education & Research Journal* 5, no. 1 (2019): 40-41.
- [7] Istiningrum, Astika, Umi Salamah, and Nurcahya Pradana Taufik Prakisy. "Lane detection with conditions of rain and night illumination using hough transform." In *2022 5th International Conference on Information and Communications Technology (ICOIACT)*, p. 429-434. IEEE, 2022. <https://doi.org/10.1109/ICOIACT55506.2022.9972068>
- [8] Marzougui, Mehrez, Areej Alasiry, Yassin Kortli, and Jamel Baili. "A lane tracking method based on progressive probabilistic Hough transform." *IEEE access* 8 (2020): 84893-84905. <https://doi.org/10.1109/ACCESS.2020.2991930>
- [9] Lin, Yancong, Silvia-Laura Pintea, and Jan van Gemert. "Semi-supervised lane detection with deep Hough transform." In *2021 IEEE International Conference on Image Processing (ICIP)*, p. 1514-1518. IEEE, 2021. <https://doi.org/10.1109/ICIP42928.2021.9506299>
- [10] Qiu, Dong, Meng Weng, Hongtao Yang, Weibo Yu, and Keping Liu. "Research on lane line detection method based on improved hough transform." In *2019 Chinese Control And Decision Conference (CCDC)*, p. 5686-5690. IEEE, 2019. <https://doi.org/10.1109/CCDC.2019.8833139>
- [11] Jiang, Libiao, Jingxuan Li, and Wandong Ai. "Lane line detection optimization algorithm based on improved Hough transform and R-least squares with dual removal." In *2019 IEEE 4th advanced information technology, electronic and automation control conference (IAEAC) 1*, p. 186-190. IEEE, 2019. <https://doi.org/10.1109/IAEAC47372.2019.8997573>
- [12] Syed, Mohammad Haider, and Santosh Kumar. "Road lane line detection based on roi using hough transform algorithm." In *Proceedings of Third International Conference on Computing, Communications, and Cyber-Security:*

- IC4S 2021, p. 567-580. Singapore: Springer Nature Singapore, 2022. https://doi.org/10.1007/978-981-19-1142-2_45
- [13] Ren, Chao, Xiuling Huang, and Harutoshi Ogai. "Lane detection based on deep learning and SSIM method." In *2022 14th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, pp. 48-51. IEEE, 2022. <https://doi.org/10.1109/IHMSC55436.2022.00020>
- [14] Yalabaka, Srikanth, Ch Rajendra Prasad, P. Sanjana, B. Lokesh, T. Shradha, and Srinivas Samala. "Lane detection using deep learning techniques." In *2022 8th International Conference on Signal Processing and Communication (ICSC)*, p. 412-416. IEEE, 2022. <https://doi.org/10.1109/ICSC56524.2022.10009637>
- [15] Gopal, K. Venu, Ch Rohith, D. Siddhartha, and Shubhangi Mahule. "Lane detection on roads using computer vision." *International journal of engineering technology and management sciences* 6, no. 4 (2022): 8-15. <http://dx.doi.org/10.46647/ijetms.2022.v06i04.002>
- [16] Jiang, Yan, Feng Gao, and Guoyan Xu. "Computer vision-based multiple-lane detection on straight road and in a curve." In *2010 International Conference on Image Analysis and Signal Processing*, p. 114-117. IEEE, 2010. <https://doi.org/10.1109/IASP.2010.5476151>
- [17] Bailey, Donald G. "Image processing using FPGAs." *Journal of Imaging* 5, no. 5 (2019): 53. <https://doi.org/10.3390/jimaging5050053>
- [18] Zakaria, Fazrul Faiz, Asral Bahari Jambek, Norfadila Mahrom, Rafikha Aliana A. Raof, Mohd Nazri Mohd Warip, Phak Len Al Eh Kan, and Muslim Mustapa. "Tuberculosis classification using deep learning and FPGA inferencing." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 29, no. 3 (2023): 105-114. <https://doi.org/10.37934/araset.29.3.105114>
- [19] Ibro, Marsida, and Galia Marinova. "Review on low-power consumption techniques for FPGA-based designs in IoT technology." In *2021 16th International Conference on Telecommunications (ConTEL)*, p. 110-114. IEEE, 2021. <https://doi.org/10.23919/ConTEL52528.2021.9495970>
- [20] Solod, Panadda, Nattha Jindapetch, Kiattisak Sengchuai, Apidet Booranawong, Pakpoom Hoyingcharoen, Surachate Chumpol, and Masami Ikura. "Memory optimization for accelerating hough transform on fpga using high level synthesis." In *2019 IEEE International Circuits and Systems Symposium (ICyS)*, p. 1-4. IEEE, 2019. <https://doi.org/10.1109/ICyS47076.2019.8982398>
- [21] El Hajjouji, Ismaïl, Salah Mars, Zakariae Asrih, and Aimad El Mourabit. "A novel FPGA implementation of Hough Transform for straight lane detection." *Engineering Science and Technology, an International Journal* 23, no. 2 (2020): 274-280. <https://doi.org/10.1016/j.jestch.2019.05.008>
- [22] Orlando, Chuquimia, Pinna Andrea, Marsala Christophel, Dray Xavier, and Bertrand Granado. "FPGA-based real time embedded Hough transform architecture for circles detection." In *2018 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, p. 31-36. IEEE, 2018. <https://doi.org/10.1109/DASIP.2018.8597174>
- [23] He, Wenhao, and Kui Yuan. "An improved Hough Transform and its realization on FPGA." In *2011 9th World Congress on Intelligent Control and Automation*, p. 13-17. IEEE, 2011. <https://doi.org/10.1109/WCICA.2011.5970530>
- [24] Khai, Lam Duc, and Trinh Viet Hoang. "A road self-guided hardware-based demo system." In *2021 15th International Conference on Advanced Computing and Applications (ACOMP)*, p. 156-161. IEEE, 2021. <https://doi.org/10.1109/ACOMP53746.2021.00028>
- [25] Dong, Ziwei, Tingting Hu, Ryuji Fuchikami, and Takeshi Ikenaga. "Encoding-free incrementing Hough transform for high frame rate and ultra-low delay straight-line detection." In *2021 17th International Conference on Machine Vision and Applications (MVA)*, p. 1-4. IEEE, 2021. <https://doi.org/10.23919/MVA51890.2021.9511359>
- [26] Lu, Xiaofeng, Li Song, Sumin Shen, Kang He, Songyu Yu, and Nam Ling. "Parallel Hough Transform-based straight line detection and its FPGA implementation in embedded vision." *Sensors* 13, no. 7 (2013): 9223-9247. <https://doi.org/10.3390/s130709223>