Indonesian Journal of Electrical Engineering and Computer Science
Vol. 16, No. 3, December 2019, pp. 1320~1326
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v16.i3.pp1320-1326

A fast-efficient parallel processing algorithm for straight line detection

Rasiq S.M.[1], Jeevan K.M.[2], S. Krishnakumar[3]
[1,3]School of Technology and Applied Sciences, Edappally, India
[2]GITAM School of Technology, Bangalore, India

Article Info

Article history:

Received Apr 2, 2019
Revised Jul 6, 2019
Accepted Jul 28, 2019

Keywords:

High speed
Line detection
Parallel processing
RK algorithm

ABSTRACT

This work presents a novel method for detecting straight lines in an image at a very high speed with optimum number of processors and their functionalities. The method can be used to extract straight lines directly from an image without noise removal and pre-processing. First the square image is converted to a binary edge image using a parallel edge detection mechanism. The parallel edge detection mechanism used in this work is capable of producing edge image within a short time. Then the binary square image is transferred to a system having large number of Processing Elements (PEs). A PE has only limited jobs such as pixel scanning, compare line length with nearby PEs and transmit data to the Main Control Unit (MCU). The MCU collects data from all PEs and evaluates straight lines. Even if the number of PEs is high, it is comparatively very much less than the parallel Hough Transform method and practically implementable using recent ULSI technologies.

Corresponding Author:

Rasiq S.M.,
School of Technology and Applied Sciences,
Edappally, Kochi-682024, India.
Email: rasiqsm2@gmail.com

1. INTRODUCTION

Knowledge about the lines in an image is useful in many applications such as unmanned vehicle guidance, robot navigation, medical image processing, object recognition, computer vision and artificial intelligence [1-12]. In a high-resolution image, thousands of lines in different angles are possible. Sequential computation is very time-consuming process to find all the lines present in the image. Hough Transform (HT) is the most widely used to detect lines.

Using HT Cartesian coordinate (X, Y) image is mapped to parameter space (ρ, Ø) as in the (1), in which each point represents a straight line called Accumulator and the content of Accumulator represents number of points included in the corresponding line.

$$\rho = X cos\emptyset + Y sin\emptyset \quad (1)$$

Since the computational complexity of HT is very large, many approaches are presented to speed up the computation process of HT. Parallel algorithms including Parallel HT are presented in several works in this area [13-19]. Recently some practical implementations of Parallel HT based on FPGA are carried out, but optimized results are not obtained to speed up the line extraction for some real time applications like robot navigation.

Ling and Honjia [15] carried out efficient work for an $n \times n$ square image and it requires mn^2 number of processors for 'm' angular variations. It is based on the linear array with reconfigurable pipeline bus system (LARPBS). The mn^2 number of processors are further divided to 'm' sub arrays where each sub

array represents an angle value. The angular variations are bounded and its value varies from 0^0 to 180^0. Implementation of this method is costly as it requires complicated hardware.

In the proposed work, straight lines are extracted from a high-resolution binary edge image using Array of Processors (AP) at a very high speed. This method requires minimum number of processors. The computation time is also minimum because separate Processing Elements (PE) is used for scanning unidirectional pixel arrays in the binary square edge image. In Section 2, the theory and methodology of line extraction is described and Section 3 gives simulated results and discussion. It is followed by Section 4 to conclude the discussion.

2. THEORY AND METHODOLOGY

This work is focused on a line extraction technique for finding suitable method with minimum transistor count and maximum frames per second. By observing possibilities of different line alignments in a binary image, it is clear that number of line alignments in an image is less than the number of processors required for Parallel HT [14, 15, 18, 19].

2.1. Line Alignments and Number of Processing Elements

The accuracy of the HT depends on accumulator cells and the bin size [20-22]. In this method, number of lines is calculated with maximum accuracy. An image has four sides AB, BC, CD and DA as shown in Figure 1. Figure 1 shows all possible line alignments from side AB to side BC. From the figure, it is clear that $(n-1)\ (n-1)\ +2$ line alignments are possible.

Similarly, the number of line alignments from the side AB to CD, AB to DA, BC to CD, BC to DA and CD to DA are calculated as $n\ (n-1)$, $(n-1)\ (n-2)$, $(n-1)\ (n-2)\ +1$,$(n-1)\ (n-2)$ and $(n-2)\ (n-2)$ respectively. The repeated line alignments are avoided while considering the number of line alignments from one side to other. Therefore, for an n x n image the total number of possible line alignments is the sum of the number of line alignments from each side of the image mentioned above. Therefore, the total number of possible line alignments is calculated as:

$$\text{Number of line alignments} \quad P_n = 6n^2 - 16n + 14 \tag{2}$$

As shown in (2) represents number of PEs (P_n) required for the proposed method to detect the entire straight lines in the image with maximum resolution and 'n' is the number of rows of a n x n image. In earlier works, the angular resolution is taken as $\Delta\theta = 1^0$ [20-22].

The decrease in angular resolution reduces the number of processors P_n. So, the (2) can be modified as (3). Here the value of 'k' depends on the angular resolution

$$P_n = 6(\tfrac{n}{k})^2 - 16\left(\tfrac{n}{k}\right) + 14 \tag{3}$$

Similarly, processing time, transistor count and accuracy of lines can also be optimized.

Different steps used in parallel HT are the calculation of 'ρ' using the (1) and calculation of cosine and sine values from look up tables. That is, in parallel HT, the processor has to perform multiplication, addition and cosine & sine value calculations. Therefore, each processor in parallel HT is complicated and it requires more processing time.

In this work, an efficient line detection method has been employed and which works at a very high speed and requires comparatively lesser number of PEs. It consists of mainly two units. First unit converts the grey level square image frame into a binary square edge image and the second unit consists of large number of PEs and a Main Control Unit (MCU).

In Figure 2 the line from the point A to the point E is drawn in Cartesian coordinate system. The relation between polar coordinate and Cartesian coordinate are shown:

$$i = \rho\, cos\emptyset \quad \text{and} \quad j = \rho\, sin\emptyset$$
$$\text{That is, } cos\emptyset = i/\rho \quad \text{and } sin\emptyset = j/\rho$$
$$\phi = sin^{-1}\left(\frac{j}{\rho}\right) \tag{4}$$

From the Figure 2, it is clear that the variable 'j' changes from 0 to n-1

$\rho^2 = n^2 + j^2$ Therefore, $\rho = \sqrt{n^2 + j^2}$

Substituting the value of ρ in (4)

$$\phi = sin^{-1}(\frac{j}{\sqrt{n^2+j^2}}) \quad (5)$$

From (5) it is clear that the angular variation in polar coordinate is not linear with respect to 'i' and 'j' in Cartesian coordinate system. Different parallel HT methods use angular resolution $\Delta\emptyset = 1^0$.

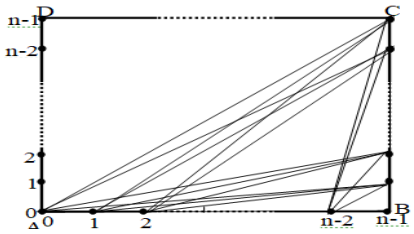


Figure 1. Possible lines from AB to BC resented

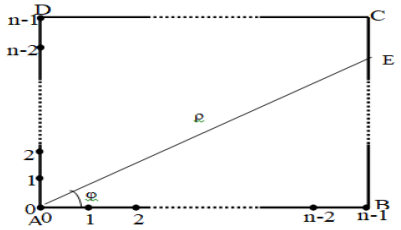


Figure 2. Line from A to E

2.2. Line Extraction

At first the input image has been transferred to a parallel edge detection mechanism which works based on the parallel edge detection method [23]. It consists of arrays of differentiators. The differentiated outputs have been transferred to an array of thresholders. The outputs of the thresholders give a binary square edge image. The number of edge pixels and the thickness of the edges are maxima when the threshold value is low. This method is used for reducing sophisticated hardware and no thinning process is needed. It is due to noises in the input image some unwanted low value pixels may present in the edges and it is rectified by a PE. Figure 3(a, b, c, d, e &f) show original image and the variation of edge width with respect to different threshold values 70, 50, 20, 10 and 5 respectively. From the figure, it is identified that the width of the edges increases with decrease in threshold value and which is encircled with a red color in Figure 3(f). The binary edge image is then transferred to a system which contains three n x n arrays. First array stores the binary edge image, second and third arrays store the (x, y) coordinates of each pixel in the binary pixel array. The pixels in the first array are the parts of a unidirectional arrays or line alignments, described in section 2.1.

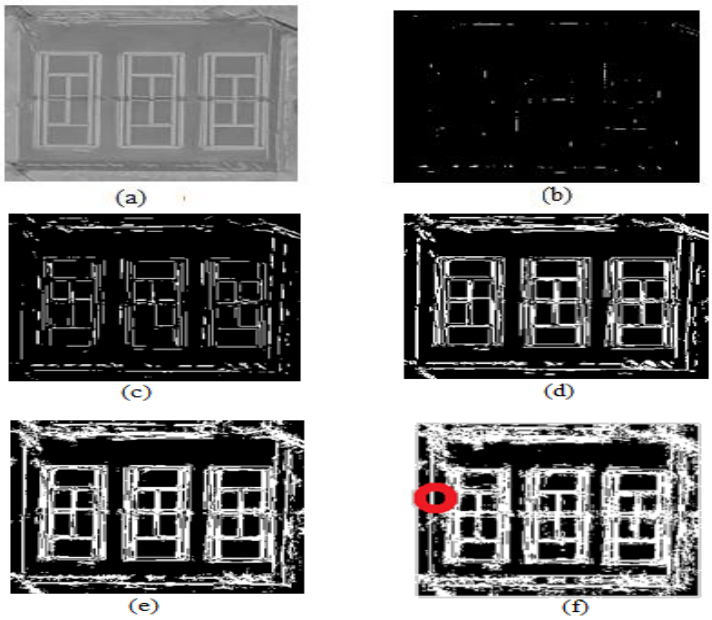


Figure 3. Edge Image for different threshold values (a) Original Image, (b) Edge Image with threshold 70, (c) Edge Image with threshold 50, (d) Edge Image with threshold 20, (e)Edge Image with threshold 10 and (f) Edge Image with threshold 5

The AP consists of number of PEs as specified in (3) and an MCU. Each PE processes a unidirectional array by performing the following functions:

a) PE scans through a line alignment of a unidirectional pixel array for consecutive high value pixels in the edge image using an N-to-1 Multiplexer. The unwanted low value pixels are eliminated in a unidirectional array by adjusting a small threshold value 'th1'. While scanning a unidirectional array if PE detects consecutive low value pixels between high value pixel arrays and if the number of low value pixels is less than threshold value 'th1', the PE consider the low value pixels as incorrect pixels and they are counted as high values. The PE fetches the starting and ending points of a high value array since the (x, y) values are attached with each pixel. The starting and ending points of a line detected by a particular PE_i are represented as (x_i, y_i) and (x_i', y_i') and the length of high value array is represented as l_i. These parameters are stored in the registers of the PE_i.
b) The comparators in the PE_i compare the line length l_i with nearby processors' (PE_{i-1} and PE_{i+1}) line lengths $l_{i-1}$and $l_{i+1,}$ only if they are having approximately same starting and ending points. The starting and ending points of the line of PE_{i-1} are represented as ($x_{i-1,}$ y_{i-1}) and (x_{i-1}', y_{i-1}') respectively and the starting and ending points of the line of PE_{i+1} are represented as ($x_{i+1,}$ y_{i+1}) and (x_{i+1}' ,y_{i+1}') respectively. The PE_iperforms the following operations

$if(x_i \approx x_{i-1} \approx x_{i+1})$ & ($y_i \approx y_{i-1} \approx y_{i+1}$)&$(x_i' \approx x_{i-1}' \approx x_{i+1}')$&$(y_i' \approx y_{i-1}' \approx y_{i+1}')$&$(l_i \geq l_{i-1})$& $(l_i \geq l_{i+1})$

{

A line is detected by the corresponding PE and the line data l_i, $(x_i, , y_i)$ and (x_i', y_i') are transferred to MCU

}

else

{

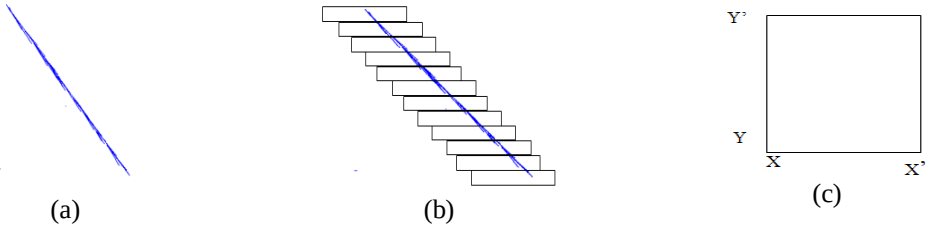
No line is detected

}

The symbol '≈ ' indicates 'approximately equal to and this function is performed by RK algorithm [24]. This algorithm is implemented in a device called 'RK device' [24].

The MCU fetches the data of lengths, starting and ending points of different lines from each PE ready to transfer data. For reducing the hardware complexity of a PE, the communications of a PE have been reduced to only nearby two PEs described as above. Figure 4(a, b & c) represent actual line with a number of coincided lines, RK block arranged in a coincided line segment and single RK block with two variable limits respectively. The data about a single line can be sent by multiple PEs and the actual line can be coincided by a number of lines as shown in Figure 4(a). The MCU collects data from all the PEs and computation becomes time consuming. In order to reduce the computation time, the MCU uses a large number of parallel RK blocks. A line segment consists of several RK blocks as shown in Figure 4(b). If the starting and ending points of a line are inside these blocks, then that line is deleted from the line data base. Thus, all the coincided lines, except the actual line which has the highest length, are deleted at a very high speed.

The RK block is the combination of two RK devices (Parallel Processors) [24]. The RK device processes a single variable. If a particular variable x is within a range X and X' the output of RK device is a logical 1; otherwise the output is logical 0. The values X and X' are learned from its previous experiences [24]. The outputs of two RK devices are logically ANDed gives an RK block as shown in Figure 4(c). The RK block processes two variables and those are the (x y) coordinate. If x and y values are within the RK block, its output becomes logical 1.



(a) (b) (c)

Figure 4. (a**)** Actual line with a number of coincided lines, (b) RK blocks arranged in coincided line a segment, (c)A single RK blocks with two variable limits

3. RESULTS AND DISCUSSION

Figure 5 (a, c, e & g) are input images and 5 (b, d, f & h) are the extracted straight lines. The figures show some simulated results using MATLAB. It is simulated with maximum resolution and got highly accurate lines. It is evident from Figure 5. Line Path is the path of a line from a starting pixel to an end pixel. By using the error elimination method and adjusting the value of threshold 'th1' some useful lines have been extracted, it is shown in Figure 5(a) and 5(b).

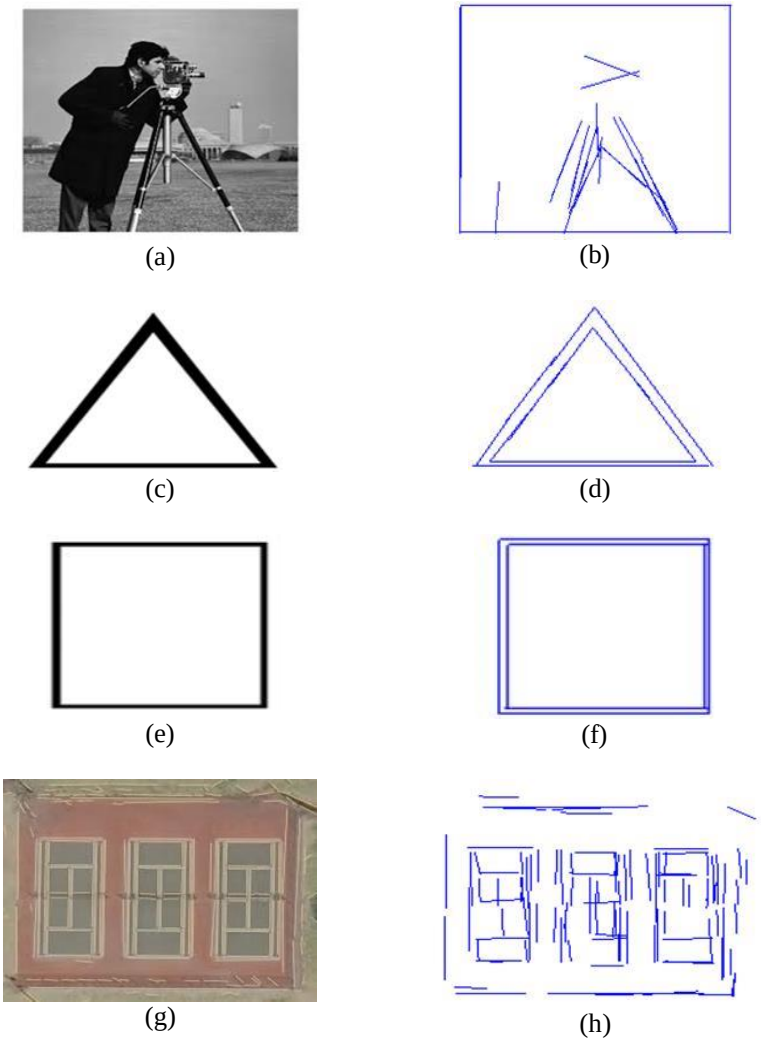


Figure 5. (a), (c), (e) and (g) are input images and (b), (d), (f) and (h) are the extracted straight lines

When the resolution is increased, the value of 'k' decreased and the number of processors P_n increased according to the (3). For different Parallel HT methods, the angle of resolution used is $1^{0.}$ Now consider the (5).

$$\phi = sin^{-1}(\frac{j}{\sqrt{n^2+j^2}})$$

Put Ø=1^0 and 'n' with different values, the corresponding values of 'j' are obtained as in the Table 1. It means that the value of 'n' in the (2) can be divided by these 'j' values and they can be substituted as 'k' in (3).

Table 1. Image Size and the Number of Processors Required for 1^0 Angle of Resolution

Image size	j	k in (3)	Pn
128 x 128	2	2	23566
256 x 256	4	4	23566
512 x 512	8	8	23566
1024 x 1024	16	16	23566
2048 x 2048	32	32	23566

From the Table 1, it is clear that, for an angular resolution of 1^0, the number of processors required to detect the line for any image of size 'n x n' is 23566. It shows that the accuracy and line resolution of this method.

Using this method, the edges can be extracted thicker than other methods. This method reduces the processing time very much. Figure 6 shows the comparison of different parallel straight-line detection algorithms with angular resolution 1^0. The horizontal axis shows the size n of an 'n x n' square image and the vertical axis shows Log_{10} (Number of processors required). It is sure that the required numbers of processors are lesser compared to earlier works [14, 15, 18, 19]. The computational complexity of each PE is also less in this method. Inter processor communication through reconfigurable mesh are required for the other methods. Also, the processors calculate cosine and sine values for different values of angles and perform arithmetic operations like addition and multiplication in the other methods [13-22].

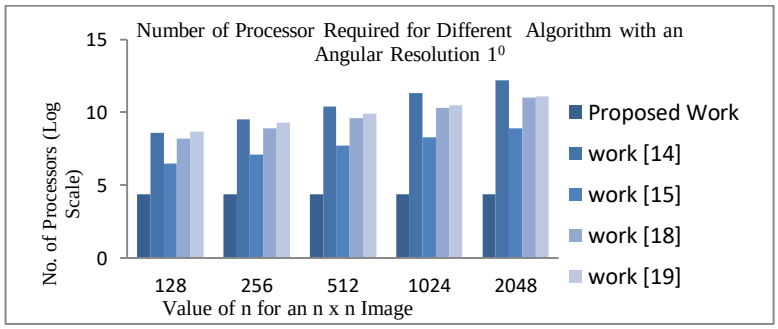


Figure 6. Comparison of number of processors required for different Parallel Algorithms with an angular resolution 1^0

The resolution of lines again can be increased by modifying edge detection method and increasing PEs. A raw image can be given as an input image to reduce processing time. RK algorithm is one of the important features of this work. It helps to reduce the processing time further. RK device is fast, easy to implement and requires very less transistor count. It is used to remove large number of coincided lines in simulation. In hardware implementation, it is a helpful tool for fast processing. The transistor count of the system can be reduced by using double-gatecarbon nanotube field effect transistor (DG-CNTFETs) based ALUs and logical units [25] in the MCU.

4. CONCLUSION

The method required only $6(n/k)^2 + 16(n/k) + 14$ number of PEs, here 'n' is the number of rows of an n x n square image and value of 'k' depends on the resolution of line extraction. Each PE has only some limited functionalities such as scanning unidirectional pixel array in a line alignment, noise elimination circuit, starting and ending points approximations using RK algorithm and comparison of line length with nearby processors. It is due to the simplicity of each PE, processing time and transistor count are less.

It is possible to make all the functions in a single chip with the ability to make high-resolution and high-speed line extraction at a very high speed. Now, ULSI technologies are advanced and even a commercially available single chip has a transistor count more than 50 billion. When the line accuracy or resolution increases the number of processors requirement increases in the order of n^2 for an n x n square image. By increasing the value of 'k' in the above equation, the number of PEs can be reduced.

By changing the scanning Multiplexer (N to 1) in a PE such as N-to-1 to N -to-m, the speed of line extraction increases almost 'm' times and each PE must be added a logical unit. The value of 'm' must be 2^r where r = 1,2 …Log_2 (N / 2).

REFERENCES

[1] N. N. S. A. Rahman, et al., "Shape and Level Bottles Detection Using Local Standard Deviation and Hough Transform," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, pp. 5032-5040, 2018.

[2] S. Ismae, et al., "Hardware/software co-design for a parallel three-dimensional bresenham's algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, pp. 148-156, 2019.

[3] M. Chandraker, et al., "Moving in Stereo: Efficient Structure and Motion using Lines," *IEEE 12th International Conference on Computer Vision (ICCV)*, pp. 1741-1748, 2009.

[4] X. Zhang and J. Zhang "Line segments matching algorithm combining msld Description and Corresponding Points Constraint," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W7, 2017.

[5] Jei X. Z. et al., "A line segments matching method based on Epipolar-line constraint and line segment featrues," *Journal of Software*, 2011.

[6] Zhang Y. S., et al., "A Hierarchical Stereo Line Matching Method Based on a Triangle Constraint," *Geomatics and Information Science of Wuhan University*, vol. 38, pp. 522-525, 2013.

[7] Liang Y., et al., "Linear Feature Matching Method Based on Local Affine Invariant and Epipolar Constraint for Close-range Images," *Geomatics and Information Science of Wuhan University*, vol. 39, pp. 88-90, 2014.

[8] H. I. Koo and N. I. Cho, "Robust skew estimation using straight lines in document images," *Journal of Electronic Imaging*, vol. 25, pp. 033014–033014, 2016.

[9] Varun R., et al., "Face Recognition using Hough Transform based Feature Extraction," *International Conference on Information and Communication Technologies (ICICT 2014) Procedia Computer Science*, vol. 46, pp. 1491-1500, 2015.

[10] A. Pathak, et al., "Line Follower Robot for Industrial Manufacturing Process," *International Journal of Engineering Inventions*, vol. 6, pp. 10-17, 2017.

[11] S. M. Aung, et al., "Live and Dead Cells Counting from Microscopic Trypan Blue Staining Images using Thresholding and Morphological Operation Techniques," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, 2019.

[12] M. Rmili, et al., "A New Approach to the Detection of Mammogram Boundary," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, 2018.

[13] L. Fisher and P. T. Highnam, "Computing the Hough transform on a scan line array processor," *IEEE Transactions on PAMI*, vol. 11, pp. 262-265, 1989.

[14] K. L. Chung and H. Y. Lin, "Hough transform on reconfigurable meshes," *Computer Vision and Image Understanding*, vol. 61, pp. 278-284, 1995.

[15] L. Chen and H. Chen, "A fast efficient parallel hough transform algorithm on LARPBS," *The Journal of Supercomputing*, vol. 29, pp. 185-195, 2004.

[16] R. K. Satzoda, et al., "Parallelizing the Hough Transform Computation," *IEEE Signal Processing Letters*, vol. 15, 2008.

[17] V. Bhatnagar, et al., "Hough Transform: Serial and Parallel Implementations," *Physics Department*, Panjab University, Chandigarh, India, 2011.

[18] Y. Pan, "Constant-time Hough transform on a 3D reconfigurable mesh using fewer processors," *Proceedings of Reconfigurarble Architectures Workshop (RAW2000)*, vol. 1800, pp. 966-973, 2000.

[19] Y. Pan, et al., "An improved constant time algorithm for computing the Radon and Hough transforms on a reconfigurable mesh," *IEEE Transactions on Systems, Man, and Cybernetics (Part A)*, vol. 29, pp. 417-421, 1999.

[20] Chen Z. H., et al., "Resource-efficient FPGA architecture and implementation of Hough tansform," *IEEE Trans Very Large Scale Integr Syst*, vol. 20, pp. 1419-1428, 2012.

[21] J. Guan and Fengwei A., "Real-Time Straight-Line Detection for XGA-Size Videos by Hough Transform with Parallelized Voting Procedure," *Sensors*, vol. 17, pp. 20, 2017.

[22] X. Lu and L. Song, "Parallel Hough Transform-Based Straight-Line Detection and Its FPGA Implementation in Embeded Vision," *Sensors*, pp. 9223-9247, 2013.

[23] Rasiq S. M. and S. Krishnakumar, "Parallel Processing Technique for Multiple Color Object Recognition," *International Journal of Pure and Applied Mathematics*, vol. 118, 2018.

[24] Rasiq S. M. and S. Krishnakumar, "Parallel Processing Technique for High Speed Object Recognition," *International Journal of Computer Applications*, vol. 99, pp. 23-27, 2014.

[25] H. Ghabri, et al., "New Optimized Reconfigurable ALU Design Based on DG-CNTFET Nanotechnology," *International Journal of Reconfigurable and Embedded Systems*, vol. 7, pp. 195-202, 2018.