IJSAA International Journal of Systems , Algorithms & Applications

A Pipelined Architecture for Image Compression Based on Hough Transform

[1]R.Manikandarajan, [2]Anita Titus
[1,2]Department of ECE,
Easwari Engineering College, Chennai
e-mail: manikanrajanr1@yahoo.com[1], anita_titus@rediffmail.com[2]

Abstract - **The proposed system deals with compression of the line detected binary image obtained from Hough transform by the usage of Row-Column Reduction Coding(RCRC) algorithm. The input image is preprocessed and converted to binary image by thresholding. The binary image is further used for edge and line detection using Hough Transform. The line detected image is then compressed using Row-column reduction coder. The image is divided into 8x8 blocks, each 8x8 block is used for Row-column reduction. There are totally two reference vectors i.e Row Reference Vector (RRV) and Column Reference Vector (CRV). Initially row reduction is performed and RRV is obtained. Over the Row reduced block, column wise reduction is performed and CRV is found out. Finally reduced block is obtained. Compressed bit stream is created by concatenation of RRV,CRV and Reduced Block. The input binary image is reconstructed by reverse RCRC algorithm. The speed of the system is enhanced by employing Pipelining architecture. The pipeline has four stages: row compress, column compress, reduced block and concatenation. The proposed pipelined system reduces the actual time taken for compression from 6.746 ns to 4.272ns. This system is synthesized using Altera cyclone II as target device. The timing analysis is performed in QUARTUS II software.**

Keywords: *Column Reference Vector (CRV), Hough transform, Row Column Reduction Coding (RCRC), Row Reference Vector (RRV), Run-length encoder (RLE).*

I. INTRODUCTION

A) Hough Transform:

Hough Transform is a standard tool in image analysis and computer vision for feature extraction. It recognizes global patterns in an image space by identifying local patterns (ideally a point) in a transformed parameter space, i.e. Hough space. In this system, Hough transform is used for detecting straight lines in an image. The input image is preprocessed and converted into a binary featured image by thresholding. The pixels with value "1" are called feature points. The incrementing property of Hough transform is described and used to reduce resource requirement. In order to facilitate parallelism, the image is divided into blocks and the incrementing property is applied to pixels within a block and between blocks. Moreover, the locality of Hough transform is analyzed to reduce the memory access.

A line is one which passes through many features points. Imagining the lines are drawn with various angles passing through a feature point, each line can be represented by using a ρ,θ value, where ρ is the perpendicular distance of the line to the origin and θ is the angle between a normal to the line and the positive x-axis. Each time a line is drawn for a feature point, it will produce a (ρ, θ) value which can be considered as a "vote" for a specific line. After processing all of the feature points, the line that has the largest accumulated votes will correspond to the line that passes through the largest number of feature points, this process is termed as the voting process. Through Hough transform, the ρ for a line with an angle θ passing through a feature point at the image coordinate (x,y) can be calculated by

$$\rho, \theta\,(x,y) = x \cos\theta + y \sin\theta \qquad (1)$$

In this paper, the focus is mainly on the voting process of the Hough transform. The number of feature pixels in an image is usually much less than that of non-feature pixels. In order to perform pixel-level parallelism of Hough transform, a (W X H) image is divided into blocks with a block-size *M* by *N*, so that a relatively small processing element (PE) can process all of the pixels inside a block simultaneously. The blocks that do not contain feature pixels are called as non-feature blocks (i.e., all zero blocks). The performance can be significantly improved if the non-feature blocks are skipped. In [4] a novel parallel Hough transform computation method called the Additive Hough transform (AHT) method is used, where the image is divided using a *kxk* grid to reduce the total computation time by a factor of *kxk*. An efficient implementation of the AHT method is performed using a look-up table (LUT) and two-operand adder arrays for every angle. Edge detection is done using sobel gradient. In [5] ,coarse-grain and a fine-grain parallelization of a straight forward Hough Transform implementation on a 8-core machine is given. Due to parallelization overheads and memory requirements, these schemes do not fully utilize the computation power. In [6] a new Hough Transform implementation for parallelization is proposed. This Hough Transform algorithm (1) has better memory performance and (2) does not use locks. A real-time image processing algorithm based on run length encoding (RLE) for a vision-based intelligent controller of a Humanoid Robot system is also given in [6]. The RLE algorithms identify objects in the image, providing their size and position. The RLE Hough transform is also used for recognition of landmarks in the image to aid robot localization.

B) RCRC Algorithm:

The compression of the binary line detected image is performed using the Row-Column reduction algorithm. This algorithm is designed to deal with the internal blocks of the input image. The RCRC proceeds as follows: For each 8×8 block, a row reference vector (RRV), a column reference vector (CRV), and a reduced block (RB) are generated. There are two reference vectors i.e Row Reference Vector (RRV) and Column Reference Vector (CRV). The RRV is constructed as follows: (i) an empty 8×1 vector is created; (ii) the first row of bits in the block is compared with the second row. If they are identical (as a result of their XORing), a '1' is placed in the first bit and a '0' in the second bit locations in the RRV; (iii) similarly the third row is compared with the second row. If they are identical, a '0' is as-

IJSAA International Journal of Systems , Algorithms & Applications

signed for the third row. This process is continued until the last row of the block; (iv) if two consecutive rows are not identical, '1' is placed for both rows in their corresponding locations in the RRV; (v) remove all identical rows from the block to obtain a row-reduced block. A fast lossless image compression scheme is stated in [3], where the number of different colors of a given map image is determined. For each image, a separate bi-level data layer is created, one for the color and the second is for the background. Then, each bi-level layer individually is compressed based on symbol-entropy in conjunction with our row-column reduction coding algorithm. In [2] simultaneous row and column reductions of higher-order linear differential systems are given. In this paper, row - column reduced forms of higher-order linear differential systems are defined with power series coefficients and two algorithms are given, along with their complexities, for their computation.

II. HOUGH TRANSFORM

A) Run Length Encoder:

A run-length encoder encodes an input binary feature image into a zero-run-length symbol stream, in order to skip the non feature blocks. A symbol is represented as a triplet {*rb, code, zl*} where *rb* bit indicates the beginning of a block-row, *code* represents the pixel values in a block and *zl* is the number of successive zero-blocks after the current block. The architecture of Hough transform is as shown in figure 1.

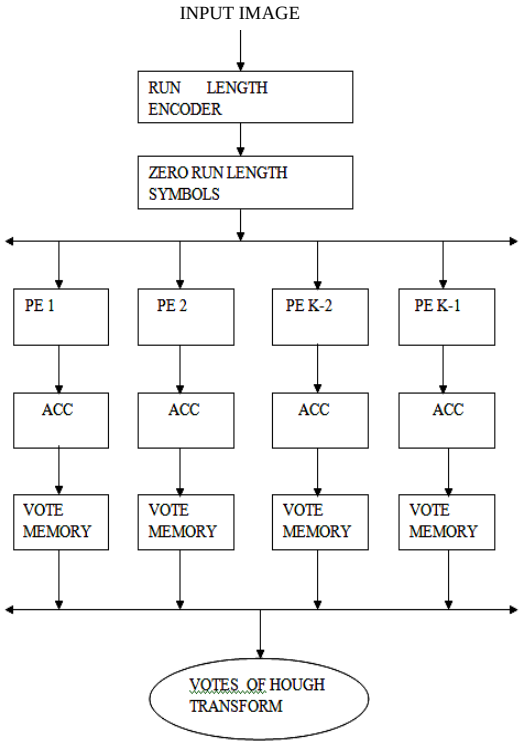


Figure 1. Hough transform block diagram.

B) Incrementing Property of Hough Transform:

Parallel PEs are used to compute different pixels in parallel form. To further improve the computation speed of Hough transform, pixel level parallelism is performed. For a specific θ value and two points in the image with coordinates given as (x , y) and ($x+dx$, $y+dy$), one could directly calculate ρ, θ values as,

$$\rho, \theta\,(x+dx\,,\, y+dy\,) = \rho, \theta\,(x,y) + d\,x\cos\theta + dy\sin\theta \qquad (2)$$

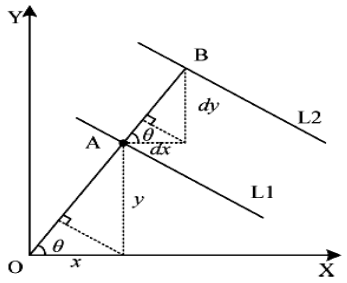
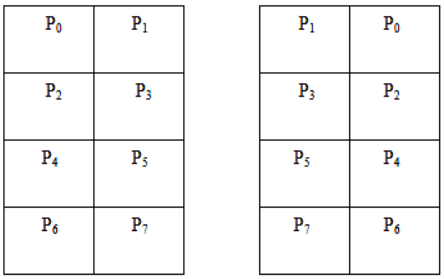


Figure 2. Incrementing property.

In Figure 2, L1 and L2 are lines with angle θ passing through points A (x , y) and B ($x+dx$, $y+dy$) respectively. The lines L1 and L2 are the distances OA and OB. OB is directly calculated or computed by adding OA and AB .



a) $0^0 < \theta < 90^0$ b) $90^0 < \theta < 180^0$

Figure 3. Pixels in a block.

In Figure 3.a the smallest ρ,θ value lies in the upper leftmost pixel P_0. In Figure 3.b, the smallest ρ,θ value lies in the upper rightmost pixel P_0.

C) Locality of Hough Transform:

The differences among the pixels in the same block are small. Also, pixels in the same block may contribute votes to the same ρ, θ in the parameter space. This property is called as locality of Hough transform. Instead of individually accumulating the votes from each feature pixel in the parameter space, pixels giving the same value can be jointly accumulated. Let ρ, θ (P_0) = $i0 + f0$, where $i0$ is the integer part and $f0$ is the residual fractional part. Let P_1 be another pixel in the same block with coordinate as ($x+dx$, $y+dy$) . The integer part can be derived by

$$i1 = i0 + \lfloor\, f0 + dx\cos\Theta + dy\sin\Theta\,\rfloor$$

where,

$$\lfloor\, f0 + dx\cos\Theta + dy\sin\Theta\,\rfloor$$

is called the vote offset from $i0$.

III. ROW-COLUMN REDUCTION CODING

The second part of the proposed system is an improved Row-Column Reduction Coding (RCRC) algorithm. The RCRC proceeds as follows: For each 8×8 block a row reference vector (RRV), a column reference vector (CRV) and a reduced block (RB) are generated.

The construction of RRV is explained in section I, the same procedure is followed for the construction of CRV from the Row Reduced block. The 1×8 column reference vector (CRV) is formed by removing zero-corresponded columns from the image block.

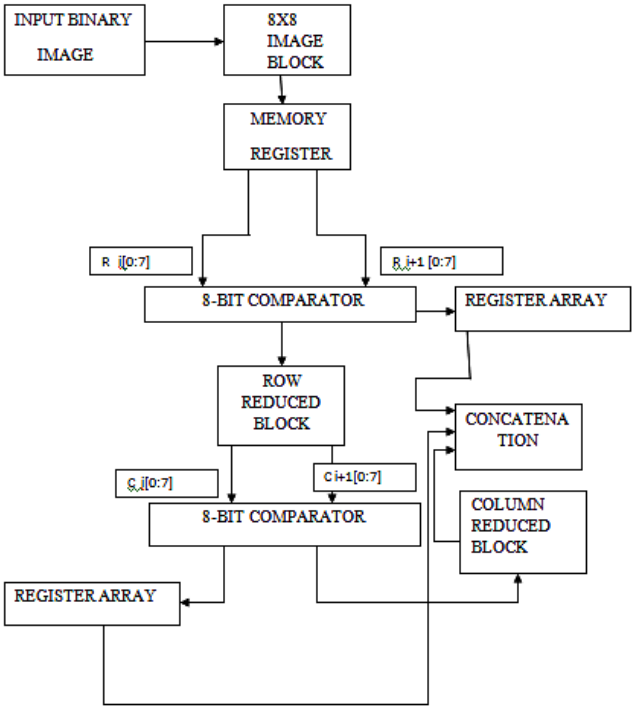


Figure 4. Proposed RCRC architecture.

This procedure will result in a row-column-reduced block or simply the reduced block, (RB). In short, the output of the RCRC algorithm will be the RRV, CRV, and the RB. The proposed VLSI architecture for the RCRC algorithm is shown in Figure 4. The architecture consists of the memory registers for the storage of the 8x8 image block and memory vectors for the storage of the RRV and CRV. This architecture also employs an 8-bit comparator for comparison of bits in rows and columns of an image

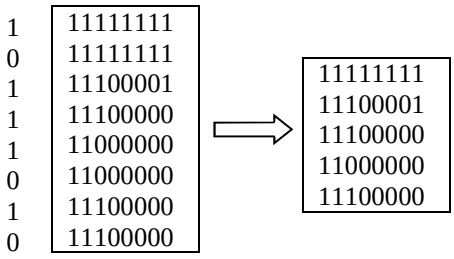


Figure 5. Row Reduction operation.

The Figure 5. illustrates the row reduction operation, where the entire 8x8 block is reduced based on the RRV value. The number of 1's in RRV corresponds to the number of non-identical rows in the block.

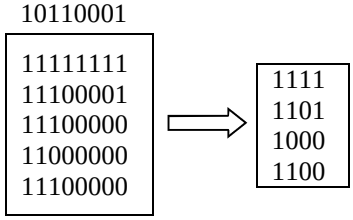


Figure 6. Column Reduction operation.

The column reduction operation is illustrated in Figure 6. The column reduction operation reduces the row reduced block further into reduced block based on the number of 1's in the CRV. Hence, the concatenated- compressed bit stream is obtained as

{ 10111010101100011111110110001100 }

The reverse RCRC process is performed to reconstruct the image from compressed bit streams. The architecture for reverse RCRC is given in Figure 7 as follows.

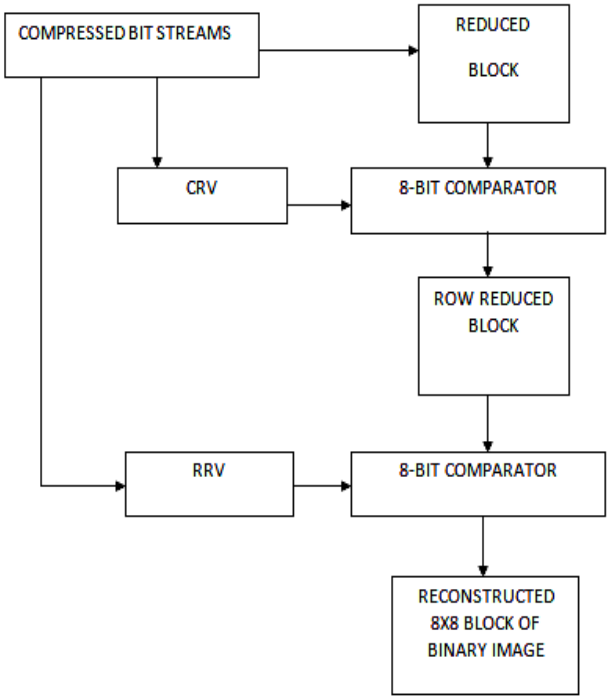


Figure 7. Architecture for reconstruction of image.

The reverse RCRC architecture contains the memory vectors for creation of CRV and RRV. It also has an 8 bit comparator for comparing the CRV and Reduced Block. The Row Reduced Block is obtained as a result of the comparison. The Column based reconstruction operation is performed as shown in the Figure 8.

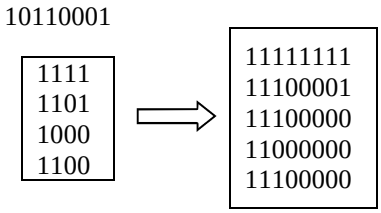


Figure 8. Column based reconstruction operation.

The CRV indicates the comparator that column 1 is repeated once, column 2 is unique, column 3 is repeated thrice and column 4 is unique.

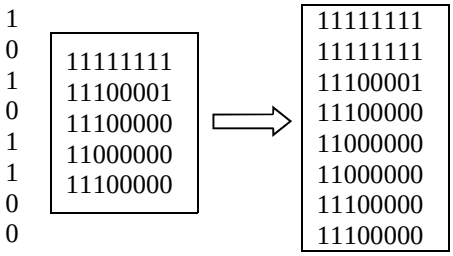


Figure 9. Row reconstruction operation.

IJSAA International Journal of Systems , Algorithms & Applications

The Row based reconstruction process is performed on the row reduced block to get back the input image block as shown in Figure 9. The RRV indicates the comparator that row 1& 2 are repeated once, row 3 is unique, row 4 & row 5 are also repeated once.

IV. PIPELINING ARCHITECTURE

Figure 10. illustrates the timing order for the pipelined architecture. The pipeline has four stages: row compress, column compress, reduced block and concatenation. Due to this pipelining style, the system consumes one 8x8 block per cycle. After the pipeline sets up, streams of compressed code outflows for every clock cycle.

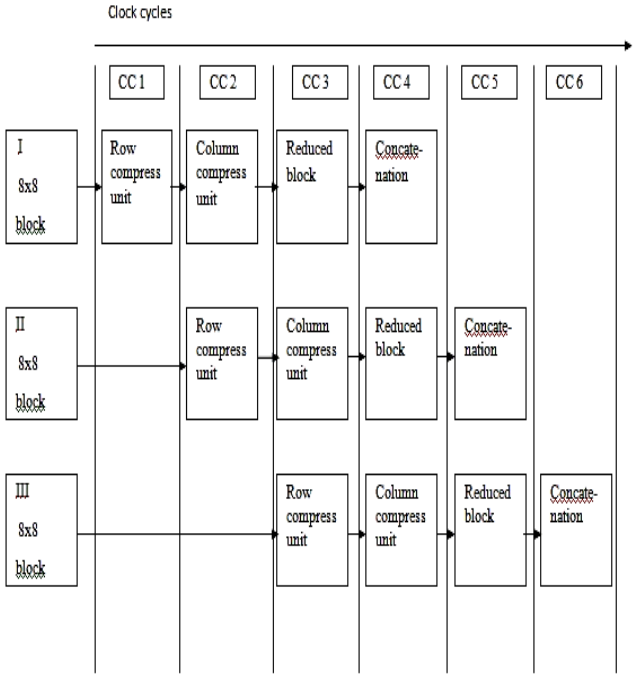


Figure 10. Timing order of RCRC algorithm.

V. RESULTS AND DISCUSSION

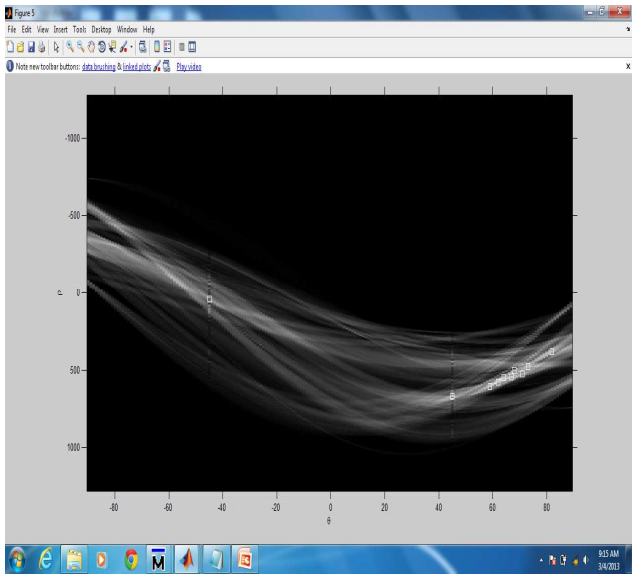


Figure 11. Line detected image.

Figure 11. shows the line detected image obtained from Hough transform. The image is converted to text file using MATLAB and compressed using RCRC algorithm.

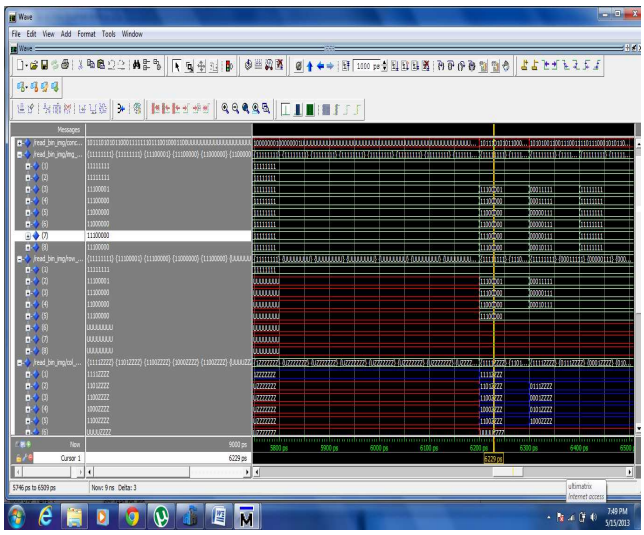


Figure 12. Reduced Block.

This system is simulated using ModelSIM Altera EDITION 6.5e software. The reduced block output is shown in Figure 12. ,where the reduced block along with the concatenated bit streams are obtained.

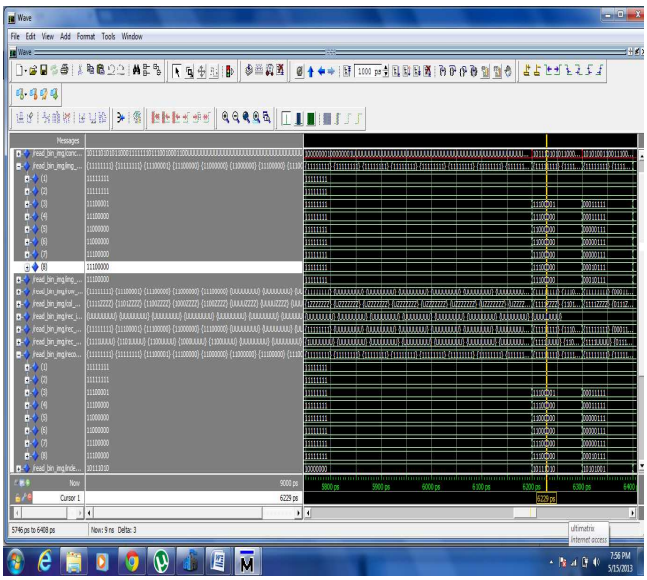


Figure 13 Reconstruction of image.

Figure 13. shows the reconstruction of original image block from obtained compressed bit streams. The reconstruction is performed using reverse RCRC algorithm. The timing analysis for the the pipelined RCRC algorithm is shown below in Figure 14.

The timing analysis of RCRC algorithm is performed in QUARTUS II software. The computational time taken by the RCRC algorithm is 6.746ns, whereas the computational time taken by the pipelined RCRC algorithm is 4.272ns.

Table 1. Comparison of RCRC with pipelined architecture.

Parameter	RCRC	Pipelined
Computational time	6.746 ns	4.272 ns
Frequency of operation	148.23 MHZ	234.08 MHZ

IJSAA International Journal of Systems , Algorithms & Applications

Table 1. shows the comparison of RCRC algorithm with pipelined architecture, where computational time and maximum frequency of operation are compared.

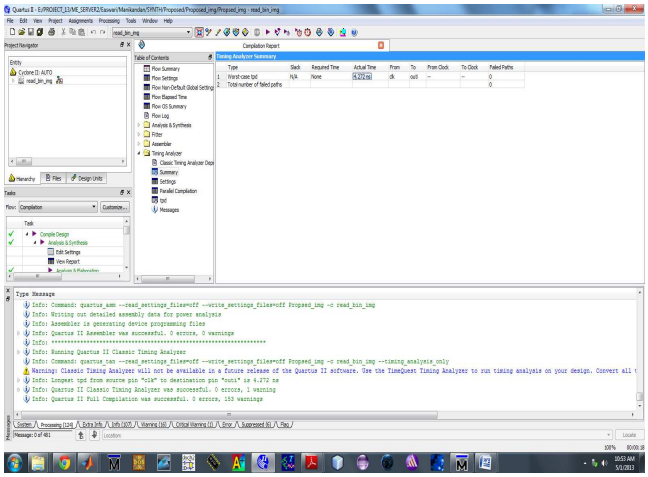


Figure 14. Timing analysis for RCRC.

VI. CONCLUSION

Hough transform is a standard tool for detecting straight lines in an image. The obtained line detected image is compressed using the Row-Column Reduction Coding algorithm. The compressed bit streams are obtained by concatenation of reduced blocks. Reverse RCRC is performed for reconstructing the image. In the RCRC algorithm, a pipelined architecture is employed for increasing the speed of the system. The computational time is reduced by reducing the number of cylcles required for operation.

REFERENCES

[1] Zhong-Ho Chen, Alvin W. Y. Su, and Ming-Ting Sun, "Resource-Efficient FPGA Architecture and Implementation of Hough Transform", IEEE transactions on very large scale integration (vlsi) systems, vol. 20, no. 8, august 2012

[2] Moulay A. Barkatou, Carole El Bacha, George Labahn, Eckhard Pflügel "On simultaneous row and column reduction of higher-order linear differential systems" Elsevier, journal of symbolic computation (2012) doi:10.1016/j.jsc.2011.12.016

[3] Saif Zahir and Arber Borici "A Fast Lossless Compression Scheme for Digital Map Images Using Color Separation," in Proc IEEE Int conf. Graphics and Image Processing Lab, Computer Science,2010 pp 1318-1321

[4] S. S. Sathyanarayana, R. K. Satzoda, and T. Srikanthan, "Exploiting inherent parallelisms for accelerating linear Hough transform," IEEE Trans. Image Process., vol. 18, no. 10, pp. 2255–2264, Oct. 2009.

[5] Y.-K. Chen, W. Li, J. Li, and T. Wang, "Novel parallel Hough transform on multi-core processors," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2008, pp. 1457–1460

[6] C. H. Messom, G. Sen Gupta, and S. N. Demidenko, "Hough transform run length encoding for real-time image processing," IEEE Trans. Instrum. Meas., vol. 56, no. 3, pp. 962–967, Jun. 2007.

[7] M. Kovac and N. Ranganathan, "JAGUAR: A fully pipelined VLSI architecture for JPEG image compression standard," Proc. IEEE, vol. 83, no. 2, pp. 247–258, Feb. 1995.

[8] Prashant Chaturvedi, Tarun Verma, Rita Jain L.N.C.T Bhopal "Design of Pipelined architecture for jpeg image compression with 2D-DCT and Huffman Encoding," International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE) Volume 2, Issue 1, January 2013

[9] Michael I. Gordon,William Thies, and Saman Amarasinghe "Exploiting Coarse-Grained Task, Data, and Pipeline Parallelism in Stream Programs", ASPLOS'06 October 21–25, 2006, San Jose, California, USA.

[10] Kopylov, P., and Franti, P. "Compression of map images by multilayer context tree modeling", IEEE Trans. On Image Processing, 14 (1), 2005, pp. 1-11.