Accelerating CORDIC for Hough Transform

S. Suchitra, R. K. Satzoda and T. Srikanthan
Centre for High Performance Embedded Systems
Nanyang Technological University
Singapore
{assuchitra,rksatzoda,astsrikan}@ntu.edu.sg

***Abstract*—CORDIC, a hardware efficient algorithm, is commonly employed to implement the linear Hough transform (HT) for detecting straight lines in images. In this paper, we exploit the properties of the CORDIC algorithm to accelerate the HT computations. Unlike the conventional approaches, where CORDIC is used for computing HT for a given set of angles, the proposed method uses CORDIC to define the angle set based on the required resolution. It is shown that with minimum number of iterations, the desired accuracy can be obtained and adjusted by just varying the register lengths. The proposed method is shown to provide 6$\times$ speed up without any additional hardware when compared with existing CORDIC based HT computation methods with $9\times$ lesser number of computations.**

***Index Terms*—CORDIC, Hough Transform, Acceleration.**

I. Introduction

The Hough Transform (HT) [1] [2] is a popular approach for detecting straight lines in images. The binary edge map of the image is sent in as input to the HT engine, and (1) is computed on all edge pixel coordinates.

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

where(x,y) is the pixel coordinate in the image, ρ is the length of the perpendicular from the origin to a line passing through (x, y), and θ is the angle made by the perpendicular with the X-axis. The output of (1) gives a sinusoidal curve, representing the parametric equivalent of that point (x, y).

Conventionally, (1) is computed for every edge pixel for each angle in the range, typically from 0^o to 180^o, with a fixed resolution, say 1^o [3], [4]. The resulting $\rho - \theta$ values are accumulated using a 2-D array, with the peaks in the array indicating straight lines in the image. Peak detection involves analysis of the $\rho - \theta$ array to detect straight lines.

One of the main issues with the use of HT is the underlying computational complexity. (1) involving trigonometric computations needs to be performed n_θ times for each edge pixel, where n_θ signifies the number of angles in the range. Depending on the resolution, n_θ could be as large as 180 or 360. This means, for a standard image size of 512×512 with 10% edge content and $n_\theta = 180$, the number of HT computations incurred would be equal to 4718592. Each HT computation involves computation of one cosine and sine operation, two multiplications and one addition.

A means to manage the computational complexity is the use of hardware efficient algorithms, such as CORDIC for implementing the HT equation (1) [4]–[7]. CORDIC (COordinate Rotation Digital Computer), developed by Volder [8] is an arithmetic computing algorithm that uses simple shifts and adds in an iterative manner to compute various mathematical functions [9]. Due to its simplicity in architecture and efficiency in computation, it has become a popular approach to perform trigonometric operations.

CORDIC has found extensive use in the hardware efficient HT implementations [7] [10] because the outputs of CORDIC engine can be directly mapped to the HT equations. CORDIC being an iterative engine would require n_c iterations to compute the HT on the coordinates of an edge pixel for each angle resulting in a total of $pm^2n_cn_\theta$ CORDIC operations to compute the HT of an $m \times m$ edge map with p fraction of edge pixels. Symmetry in CORDIC operation [4], [10] has been used to further reduce the number of CORDIC operations for HT computation by half [4]. Pipelining the CORDIC at the cost of area was also proposed in [7] to reduce the total computation time for HT computation.

In this paper, we propose an efficient use of CORDIC engine to reduce the total computation time of HT computation by orders of magnitude without compromising on area cost. Instead of using CORDIC to compute the HT of desired angles in the Hough space, the proposed method uses CORDIC iterations to generate the angle space. This method is shown to generate more accurate results at lower computation times and computation complexity as compared to the existing CORDIC based HT computation techniques. The rest of the paper is organized as follows. Some preliminaries of the CORDIC algorithm are discussed in Section II. The proposed method of computing the HT using CORDIC is described in detail in Section III. The computational efficiency and timing of the proposed HT computation technique are discussed in Section IV. The paper is concluded in Section V.

II. Preliminaries: CORDIC Algorithm

We first give a brief overview of the CORDIC algorithm as it forms the basis for the proposed high speed Hough Transform engine. All trigonometric functions can be computed or derived using vector rotations on CORDIC [9], based on the rotation transform equations in (2). Given a coordinate (x, y), CORDIC rotation engine generates the rotated (x', y') as shown below:

$$\begin{aligned} x' &= x\ cos\theta - y\ sin\theta \\ y' &= y\ cos\theta + x\ sin\theta. \end{aligned} \tag{2}$$

, where (x, y) and (x', y') are the vectors before and after rotation respectively and θ is the angle of rotation. The low

computational complexity of the CORDIC algorithm is brought about from the underlying principle that any angle can be approximated as a summation of n micro angles of the form $tan^{-1}(2^{-i})$, i.e. $\theta \approx \sum_{i=1}^{n} \pm tan^{-1}(2^{-i})$. CORDIC performs a series of micro-rotations on a vector lying on the X-Y plane, which is equivalent to performing a rotation of the desired input angle θ. Having micro angles of this form reduces the multiplications in (1) into simple shift operations in hardware. The CORDIC equations are given below by

$$\begin{aligned} x_{i+1} &= k(x_i - \delta_i y_i 2^{-i}) \\ y_{i+1} &= k(y_i + \delta_i x_i 2^{-i}) \end{aligned} \tag{3}$$

where k is a fixed scaling factor. An n_c-iteration or n_c order CORDIC approximates any given input angle to one of the 2^{n_c} angles. From (3), it can be seen that for each of the iterations, the direction of rotation, given by δ_i, can be ± 1. So, for n_c micro-rotations, there are 2^{n_c} possible rotation combinations. Hence, an n-bit CORDIC engine would approximate any given input angle to one of the following 2^{n_c} target-angles: $\pm tan^{-1}2^0 \pm tan^{-1}2^{-1} \pm tan^{-1}2^{-2} \pm tan^{-1}2^{-3} \pm \pm tan^{-1}2^{-n}$. Three additions, two for x_{i+1} and y_{i+1} in (3) and one for the residue angles, are performed in every CORDIC iteration. Table I shows the CORDIC iterations for $20^o = 0.349066667$ radians. The third column shows the angle register which is updated in every iteration. It is initialized to 0.349066667 radians. In every iteration i, $tan^{-1}2^{-i}$ is added or subtracted from the value stored in the angle register based on the δ value. The value in the last column shows the actual angle obtained by $\pm tan^{-1}2^0 \pm tan^{-1}2^{-1} \pm tan^{-1}2^{-2} \pm tan^{-1}2^{-3} \pm \pm tan^{-1}2^{-i}$ at the end of the i-th iteration. In other words, the actual angle is being approximated to this angle at the end of the i-th iteration. In Table I, $20^o = 0.349066667$ radians is approximated to 0.349073139 radians after 16 iterations. Typically, an n-iteration or n-order CORDIC engine generates an output with n-bit accuracy.

TABLE I
CORDIC ITERATIONS FOR ANGLE = 20^o (0.349 RAD)

Iteration i	$tan^{-1}2^{-i}$	CORDIC angle accum.	$\delta = \pm 1$	Approx. angle
Initial		**0.349066667**	1	
1	0.463647609	-0.114580942	-1	0.463647609
2	0.244978663	0.130397721	1	0.218668946
3	0.124354995	0.006042726	1	0.34302394
⋮	⋮	⋮	⋮	⋮
14	6.10352E-05	-5.22485E-05	-1	0.349118915
15	3.05176E-05	-2.1731E-05	-1	0.349088398
16	1.52588E-05	-6.47217E-06	-1	**0.349073139**

III. PROPOSED HOUGH TRANSFORM TECHNIQUE

As discussed in the Section I, the HT equation (1) is conventionally computed for a set of angles $[\theta_{min}, \theta_{max}]$, where the number of these angles n_θ is given by the resolution. The resolution is decided based on the required accuracy of the line detection process. Typically, equally spaced angles within the range are considered. It can be seen from CORDIC rotation equations (2) that the outputs of CORDIC map directly to the HT equation in (1) for different angle ranges. For each angle $\theta \in [\theta_{min}, \theta_{max}]$ in the Hough space, (1) is computed by employing the CORDIC engine.

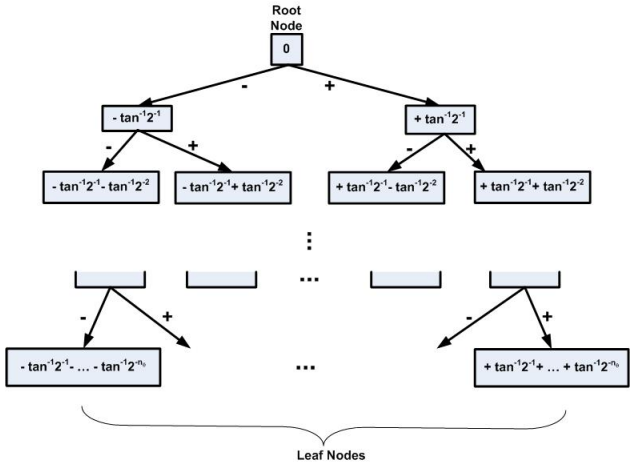

Fig. 1. Possible rotation angles of CORDIC represented as a binary tree.

In our proposed approach, instead of fixing these angles in the Hough space, we exploit the properties of the CORDIC algorithm to derive the angle space that aids in simplifying the complexity of the overall HT and also improving the performance. Fig. 1 shows a binary tree obtained by the different permutations in which the micro rotations in CORDIC result in 2^{n_c} angles after n_c iterations. The binary tree in Fig. 1 has n_c stages, each stage refering to one iteration. The nodes (shown by rectangles) in the j-th stage represent the angles obtained from different summations dictated by the nodes till the j-th stage, i.e. $\sum_{i=1}^{j} \pm tan^{-1}2^{-(i-1)}$.

We propose to set the angles given by the leaf nodes in the last stage of the CORDIC binary tree as the set of θs in the Hough space for HT computations, rather than computing HT for predefined angles. The advantages of the proposed method are explained below with examples:

A. Non-redundant computations

By choosing the angle set based on the proposed method, we drastically reduce the redundancies in computations. To illustrate this, let us consider two angles in the Hough space for which the HT is to be computed, say 80^o and 60^o using a 3-iteration CORDIC. Fig. 2 shows the two paths A and B that will be taken by the CORDIC engine after three iterations. So a 3-iteration CORDIC would give the ρ values corresponding to 85^o and 57^o respectively for the two input angles. It can be seen that there are redundant computations in the two paths A and B. This is because both the paths go through the nodes 45^o and 71.565^o and these nodes are computed twice. On the other

hand, in the proposed method, the angles in the Hough space are set as the angles in the leaf nodes. In this case, for a crude angle resolution with $n_\theta = 8$, the angle set will be {85.601, 57.529, 32.471, 4.398, -4.398, -32.471, -57.529, -85.601}. The CORDIC engine, after three iterations and passing through all intermediate nodes, generates the accurate ρ values of all the 8 angles. This angle range can be expanded by increasing the number of iterations.

Thus, the conventional use of the CORDIC engine for HT computation leads to redundant computations for different angles. In the proposed method, this redundancy is avoided as all the branches are computed for the lesser number of nodes because the number of iterations is lesser than the conventional method.

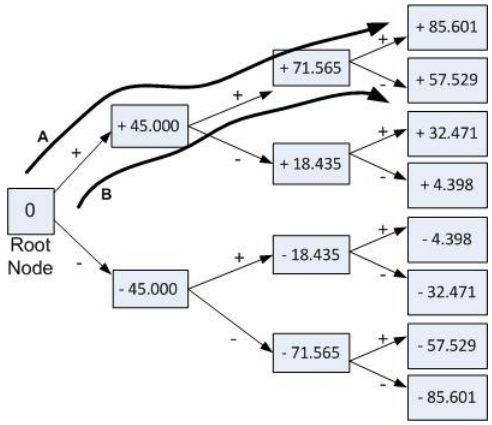

Fig. 2. Example illustrating the proposed approach.

B. Reduced number of iterations:

The proposed selection of CORDIC angles also reduces the number of iterations of CORDIC, and hence leads to further computational savings. Let the number of iterations in the proposed method and conventional method be denoted by n_p and n_c respectively. It is seen that for the same accuracy of ρ values, n_p is much lesser than n_c. This can be understood by first looking at the resulting errors from the CORDIC computations.

There are two kinds of errors that are associated with the CORDIC engine. The first error is the *approximation error* resulting because of the inherent iterative convergence property of CORDIC. In other words, any given angle can only be approximated to one of the 2^n_θ angles of the CORDIC. Typically, a CORDIC engine of order n_θ (i.e. an n-iteration CORDIC engine) gives an accuracy of n-bits [9]. At every iteration, the accuracy is improved by 1-bit [9] and the approximation error can be controlled by changing the order of the CORDIC engine. In Fig. 2, for a 3-order CORDIC, the resultant ρ values for 80^o and 60^o are accurate to three bits due to this error. In the proposed approach, this error does not creep in because there is no approximation of angles. The angles for which the ρs are being computed are the angles given by the CORDIC binary tree itself. By setting a suitable value of n_p, the angle resolution can be managed efficiently. For example, with an 8-order CORDIC, i.e. $n_p = 8$, $2^8 = 256$ angles are obtained in the last stage of the binary tree from -99.43^o to 99.43^o with an angle resolution of 0.89. Hence in 8-iterations it is possible to get accurate ρs for 256 angles in this range. The only source of error in the proposed method is the second kind of error - *quantization error* that arises because of the fixed register length used in CORDIC engine. This error is due to carry propagation resulting from the iterative additions. This can be resolved by setting the register length as $n_p + \lceil log_2 n_p \rceil$ to accomodate the additional carry bits. It should be noted that the conventional approach also suffers from this error. Thus, given a fixed register width, the proposed approach can generate a more accurate result with lesser number of iterations than the conventional usage of CORDIC for HT.

In summary, the proposed technique exploits the micro rotations of CORDIC algorithm to define the angle space of the HT. This reduces the number of iterations to get the ρ values of angles with less than 1^o angle resolutions.

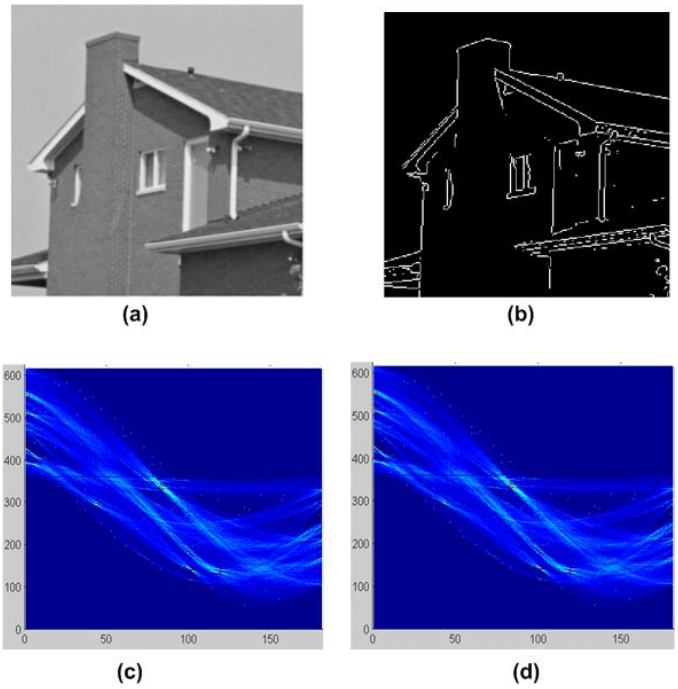

Fig. 3. (a) One of the benchmark images (b) Edge map (c) Hough space generated using the proposed technique (d) Hough space from the conventional approach

IV. Discussion

The proposed method was applied on a series of images. Fig. 3(a) and (b) show one of the benchmark images and its edge map. The Hough spaces obtained by applying the proposed method and the conventional approach on the edge map are shown in Fig. 3 (c) and (d) respectively. It can be seen that the Hough space with less than 1^o resolution in the proposed method captures all the information and more as compared to the conventional method with 1^o resolution. An 8-order CORDIC ($n_p = 8$) was considered in the proposed

method whereas 12-order CORDIC ($n_c = 12$) was considered for the conventional approach.

A. Computational Efficiency

The proposed approach reduces the computation efficiency by orders of magnitude. This is because of the tremendous reduction in the number of CORDIC mirco rotations. Conventional usage of CORDIC would require three additions per iteration, one each for x_{i+1}, y_{i+1} and residue angles. Thus the total number of addition operations to compute the HT of n_θ angles using an n_c-order CORDIC in the conventional way for $p < 1$ fraction of m^2 pixels is given by:

$$N_{con\_Adds} = (pm^2\ n_\theta n_c) \times 3. \tag{4}$$

As discussed in the previous section, the proposed method gives the HTs of 2^{n_p} angles for an n_p-order CORDIC. This is obtained by computing x_{i+1} and y_{i+1} in (3) for all the nodes in Fig. 1. The total number of additions involved in the proposed method is equal to:

$$N_{prop\_Adds} = (pm^2 \sum_{i=1}^{n_p} 2^i) \times 2. \tag{5}$$

From (4) and (5) the ratio of the number of additions of the proposed method against conventional method is given by:

$$\alpha = \frac{N_{prop\_Adds}}{N_{con\_Adds}} = \frac{2^{n_p+1}}{3n_c} \tag{6}$$

For the example considered in Fig. 3, the total number of addition operations per every edge pixel required to generate the ρ values of 256 angles in $[-99.43^o, 99.43^o]$ using the proposed method with 8-order CORDIC is equal to 1020. The conventional approach with 12-order CORDIC takes $3 \times 12 \times 256 = 9216$ additions per edge pixel. This is 9 times computationally more than the proposed approach.

B. Timing

Table II compares the timing (in terms of number of cycles) of the proposed method in different configurations of CORDIC against the conventional usage of CORDIC for HT computations. If an iterative CORDIC is used for computing the HT using conventional way [7], every iteration would require one cycle resulting in $n_c n_\theta$ cycles for every edge pixel. In the proposed scheme, 2^i cycles are required in the i-iteration to compute the x_{i+1} and y_{i+1} of every node in the binary tree. This results in a total of $\sum_{i=1}^{n_p} 2^i$ cycles for every edge pixel. In the pipelined mode, both methods of computing the HT would require a similar number of cycles. The initial pipeline latency is however lesser in the proposed method to fill the n_p stages because $n_p < n_c$. For comparison purposes the same number of angles is considered. If $n_c = 8$, $n_p = 12$ and $n_\theta = 2^{n_p} = 256$, the proposed method with iterative CORDIC gives 6 times speed up compared to its iterative counterpart of the conventional HT computation approach. Similarly, the pipelined version gives 1.015 times speed up.

TABLE II
TIMING COMPARISON

	CORDIC configuration	Timing (cycles)
Proposed	Iterative	$pm^2 \sum_{i=1}^{n_p} 2^i$
	Pipelined	$pm^2(n_p + n_\theta)$
Conventional [7]	Iterative	$pm^2\ n_\theta n_c$
	Pipelined	$pm^2(n_c + n_\theta)$

V. CONCLUSION

In this paper we propose a HT computation technique using the CORDIC algorithm. The CORDIC micro rotations of an n_p-order CORDIC are exploited to generate the ρ values for 2^{n_p} angles in the Hough space. The angles are shown to be decided by the CORDIC micro rotations. The accuracy of the ρ values obtained from the proposed technique is greater than the conventional usage of the CORDIC with lesser number of iterations. The accuracy in the proposed method is shown to be affected by the register length only and the number of iterations affect the angle resolution only of the Hough space. It is also shown that with just 8 iterations, an angle resolutions of $0.89^o < 1$ is achienved. The total number of additions were found to be 9 times lesser than the conventional CORDIC for HT. The reduction in computations is shown to be the result of lower redundancy in the proposed technique. Further work will be done to study the VLSI metrics of the proposed technique when implemented on an FPGA or ASIC platform.

REFERENCES

[1] P. Hough, "Methods and means for recognizing complex patterns," *US Patent 3 069 654*, 1962.

[2] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice Hall, 2002.

[3] M.-Y. Chern and Y.-H. Lu, "Design and integration of parallel Hough-transform chips for high-speed line detection," *Proceedings of 11th International Conference on Parallel and Distributed Systems*, vol. 2, pp. 42 – 46, July 2005.

[4] F. Zhou and P. Kornerup, "A high-speed Hough transform using CORDIC," *Proceedings of International Conference of VLSI Digital Signal Processing*, pp. 482–487, June 1995.

[5] D. Timmermann, H. Hahn, and B. J. Hosticka, "Hough Transform using CORDIC method," *Electronics Letters*, vol. 5, no. 3, pp. 205 – 206, February 1989.

[6] A. K. Majumdar, "Design of an asic for straight line detection in an image," *Proceedings of Thirteenth International Conference on VLSI Design*, pp. 128 – 133, January 2000.

[7] F. Zhou and P. Kornerup, "A high speed Hough Transform using CORDIC," Tech. Rep. PP-1995-27, 1, 1995. [Online]. Available: citeseer.ist.psu.edu/zhou95high.html

[8] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions Electron. Comput.*, vol. 8, pp. 330 – 334, 1959.

[9] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," *Proceedings of 1998 ACM/SIGDA Sixth International Symposium on FPGA*, pp. 191 – 200, February 1998.

[10] S. M. Karabernou and F. Terranti, "Real-time FPGA implementation of Hough Transform using gradient and CORDIC algorithm," *Journal of Image and Vision Computing*, vol. 23, pp. 1009 – 1017, July 2005.