View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositorio Institucional de la Universidad de Alicante

PARAMETRIZED ARCHITECTURE FOR HOUGH TRANSFORM RECURSIVE EVALUATION

Juan Manuel García Chamizo
Mª Teresa Signes Pont
Higinio Mora Mora
Gregorio de Miguel Casado

Departamento de Tecnología Informática y Computación-Universidad de Alicante
{juanma, teresa, hmora, demiguel}@dtic.ua.es

ABSTRACT

The Hough Transform (HT) is a useful technique in image segmentation, concretely for geometrical primitive detection. A Convolution-Based Recursive Method (CBRM) is presented for function evaluation. In this generic approach, calculations are carried out by an unique parametric formula which provides all function points by successive iterations. The case of combined trigonometric functions involved in the calculation of the HT is analyzed under this scope. An architecture for reconfigurable FPGA-based hardware, using Distributed Arithmetic (DA) implements the design. The CBRM implementation provides improvements such as memory and hardware resources saving, as well as a good balance between speed and error-dependable precision.

1. INTRODUCTION

Proposed in 1962, the Hough Transform (HT) has become a widely used technique in image segmentation: plane curve detection [1], object recognition [2], air picture vectorization [3], etc. The HT is very suitable because of its robustness, although the great amount of temporary and spatial resources that requires has moved it away from real time applications. This way, the investigation efforts in HT have dealt with the design of fast algorithms and parallel or ad-hoc architectures. As the HT consists of function evaluation using arithmetic operations different algorithmic approaches have been developed: piece-lineal (PLHT) [4], combinatory (CHT) [5], binary (BHT) [6], adaptive (RHT) [7] and fast (FHT) [8]. There are also implementations of the CORDIC algorithm for applications that demand high speed and precision, such as digital signal and image processing and algebra [9][10]. However, their drawback is the lower degree of regularity and parallelism capabilities when comparing with the traditional algorithm. The parallelism that underlies in the traditional algorithm allows the implementation of architectures with shared or distributed memory (lineal array, mesh, hypercube [11] and binary tree) as well as specific HT systolic ones [12].

This paper presents the HT calculation under the scope of a Convolution - Based Recursive Method (CBRM), providing a suitable approach for the evaluation of the combined elementary functions involved, sine and cosine, together with systematic addition and multiplication operations. Compared with other proposals, CBRM offers good performance in speed, memory requirements and trade-off between precision and error. The Distributed Arithmetic (DA) implementation provides simplicity to the circuit and also good results as for error and speed.

The paper is structured in six parts: the CBRM fundamentals and its application to the calculation of the HT (Sec. 2); DA-architecture design (Sec. 3); implementation, experiments and results obtained (Sec. 4); comparison with other proposals (Sec. 5) and the conclusions drawn from this work (Sec. 6).

2. CONVOLUTION-BASED RECURSIVE METHOD (CBRM)

In this section, the fundamentals of the method and its application to the HT are exposed.

2.1. Fundamentals

The CBRM is introduced in order to improve function evaluation performance in computational operations.

The conceptual foundation of the method resides in that the convolution provides a framework for evaluating functions with independence of its structural peculiarities. In the case of two discrete functions f and g: $N \rightarrow R$ the convolution Ψ can be written as:

$$\begin{aligned} &\Psi(0) = f * g(0) = f(0)\cdot g(0) \\ &\Psi(1) = f * g(1) = f(0)\cdot g(1) + f(1)\cdot g(0) \\ &\ldots \\ &\Psi(p) = f * g(p) = f(0)\cdot g(p) + \ldots + f(p)\cdot g(0) \end{aligned} \qquad (1)$$

Using mathematical transformations (1) can be expressed recursively, achieving a simpler formulation (2).

$$\Psi_{q+1} = \alpha\, \Psi_q + \beta\, G_q \qquad q \in [0, I] \subset \mathbb{N} \qquad (2)$$

CBRM is based on a mathematical expression in which the first term is a Ψ value computed at every q iteration, which allows the computation of the next value at iteration $q+1$, and the second term is a value calculated using a function called G. The function G can be obtained like Ψ or just as a function of Ψ. The contributions of Ψ and G are weighted by the parameters α and β; the values of the parameters and the contribution of the function G are particular for each Ψ.

Reciprocally, it can be demonstrated that any discrete variable function that can be expressed in recursive form (2) with α different from zero, can be split up into two convolving functions f and g, $f(q) = \alpha^q$, $g(q) = \beta \cdot G_q$.

On these theoretical bases, the CBRM (2) supports the evaluation of a wide group of common functions. The computational interest of the method is related to the compactness of the recursive evaluation pattern. It takes advantage of pre-calculated values of the function to calculate successive ones by using a repetitive and simple operation. This specially becomes even more powerful when calculating combinations of elementary functions and when a lot of function points are required.

2.2. Application of the CBRM to the HT

The geometric primitive detection using the HT implies three stages: image outline creation by using an edge detector, application of the HT to each point of the image and a voting process in the Hough domain in order to extract the geometric primitives.

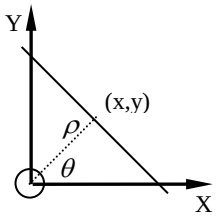

Fig 1. HT parameters for the line detection case.

If the geometric primitive to be detected is a straight line, the HT transforms each point $P(x,y)$ in the Cartesian domain in a point (ρ,θ) in the Hough domain, and vice versa. So, the Hough domain is complete and unique for $0 \le \rho < \Pi$ line representation.

The Hough domain can be interpreted as a voting grid. Each point in the Cartesian domain votes for a set of lines that intersect it and that stand for a grid point (ρ,θ). A local maximum point in the voting grid represents the best adjusted line detected. The increment of the grid points $\Delta\rho$ y $\Delta\theta$ establish both distance and angular difference between lines in the Cartesian domain, respectively.

The HT is a robust technique since the voting process is not affected by isolated noise points because wrong votes do not affect the local maximum. The HT also manages successfully line occlusion problems, because the distance between points is not relevant.

The parametrized space is discretized in N_θ levels, from 0 to Π and N_ρ levels, from ρ_{min} to ρ_{max}. The HT calculates the ρ values for all the angles in $[0,\Pi[$ and for every pixel in the image. The direct calculation has $O(N^2)$ complexity and the global amount of operations is $N^2 \cdot N_\theta$. If $[0,\Pi[$ is considered as $[0,\Pi/2[\cup[\Pi/2,\Pi[$, the HT for every pixel (x_i,y_j) in the image can be written as:

$$\begin{aligned} (\rho_I)_k &= x_i \cdot cos\theta_k + y_j \cdot sen\theta_k \quad 0 \le \theta_k < \Pi/2 \\ (\rho_{II})_k &= y_j \cdot cos\theta_k - x_i \cdot sen\theta_k \quad \Pi/2 \le \theta_k < \Pi \end{aligned} \tag{3}$$

If:

$$\begin{aligned} &\theta_k = \theta_{k-1} + \Delta\theta \\ &cos\theta_k = cos(\theta_{k-1} + \Delta\theta) \\ &sen\theta_k = sen(\theta_{k-1} + \Delta\theta) \\ &cos\Delta\theta = \alpha, \quad sen\Delta\theta = \beta \end{aligned} \tag{4}$$

When substituting (4) in (3) we have that:

$$\begin{aligned} &(\rho_I)_k = \alpha \cdot (\rho_I)_{k-1} + \beta \cdot (\rho_{II})_{k-1} \\ &(\rho_{II})_k = \alpha \cdot (\rho_{II})_{k-1} + \beta \cdot (\rho_I)_{k-1} \\ &with \quad \alpha^2 + \beta^2 = 1 \end{aligned} \tag{5}$$

As shown in (5), $(\rho_I)_k$ and $(\rho_{II})_k$ can be evaluated by the CBRM, by using $G(k) = (\rho_I)_k$ when evaluating $(\rho_{II})_k$ and vice versa; $(\rho_I)_0$ y $(\rho_{II})_0$ should be initialized with the value of the coordinates of each pixel in the image.

3. ARCHITECTURE

Distributed Arithmetic (DA) is used in order to multiply using a look-up table based scheme. Figure 2 shows the complete architecture and Figure 3 the DA module.

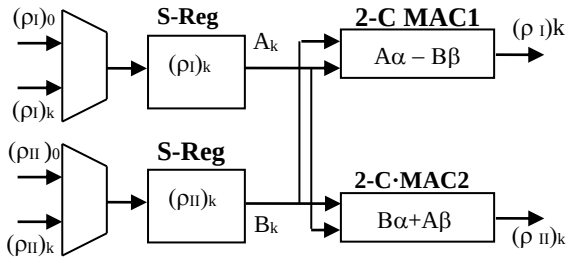

Fig 2. DA architecture for the CBRM.

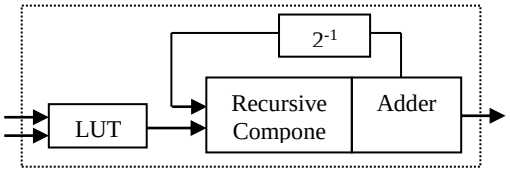

Fig 3. Structure of a 2-C·MAC.

First, shift-registers are initialized with $(\rho_I)_0$, $(\rho_{II})_0$ and next, feedbacked with the output values. The values of $(\rho_I)_{i-1}$, $(\rho_{II})_{i-1}$ are shifted to MACs one bit a time, starting

with the lower less significant bit. MAC blocks multiply $A\cdot\alpha - B\cdot\beta$ and $B\cdot\alpha + A\cdot\beta$ in serial mode in n cycles. These are signed expressions which depend on the sign of *A* and *B*, as we suppose without loss of generality that α and β are positive. Every cycle, the only possible values for A_i and B_i are (00), (01), (10) and (11); so, partial results of the expressions are stored in a look-up table (LUT) (Table 1). The sum of partial products is performed in the scaled accumulator.

The LUT can be addressed using four bits: two bits represent the input pair (A_i, B_i) and the others are the sign of *A* and *B* (the most significant bit). For each column, the left subcolumn represents LUT1 and the right one LUT2.

	A>0 y B>0 (00)		**A>0 y B<0 (01)**		**A<0 y B>0 (10)**		**A<0 y B<0 (11)**	
(00)	0	0	0	0	0	0	0	0
(01)	-β	α	β	−α	−β	α	β	−α
(10)	α	β	α	β	−α	−β	−α	−β
(11)	α−β	α+β	α+β	−α+β	−α−β	α−β	−α+β	−α−β

Table 1. LUT addressing.

Input data must be signed-value so that to address the LUT. Parameters α and β are real numbers so they might be represented in fixed point two's complement and only the entire part will be considered. Partitions are done as follows: input values of $(\rho_I)_k$ and $(\rho_{II})_k$ have 1 bit for sign, *n/2 –1* bits for entire part, n/2 bits for fractionnary part; parameter values α and β have 1 bit for sign, 1 bit for entire part, *n-2* bits for fractionnary part.

4. IMPLEMENTATION AND EXPERIMENTS

Implementation has been done for FPGA (V300 BG352-6 Xilinx, Virtex E). Simulation in VHDL provides critical path estimations. The error performed by the CBRM with regard to the direct calculation has been carried out by using calculating $(\rho_I)_k$ $(\rho_{II})_k$ using C language.

4.1. Calculation time estimation.

The architecture (Fig. 2) has been implemented with two identical subcircuits that calculate $(\rho_I)_k$ $(\rho_{II})_k$ in parallel. VHDL describes both of them as a composition of two modules: one devoted to the selection of the *n* partial products and another one that performs the sum of these partial products. Table 2 shows CBRM time estimations:

bits	**module 1**	**module 2**	**Overall**
8	0.468ns	28.188ns	28.656ns
16	0.468ns	40.620ns	41.088ns
32	0.468ns	139.180ns	139.648ns

Table 2. Calculation time of CBRM versus size of data.

4.2. Error calculation

In order to study the relative error and the number of erroneous bits in the result (5), the direct calculation has been taken as reference (3). Increasing data size from 16 to 32 bits rebounds favorably in general improvements. When rising α, the error increases as the rotated angle becomes bigger, while when rising ß both increase as the number of calculated values grow. Furthermore, as the rotated angle and/or the number of iterations rise, the erroneous bits increase less steeply (logarithmic growth) until a constant value is reached that can be notably lowered by controlling the parameters. The lineal growth of the relative error is the worst aspect outlined by this analysis, although it also can be controlled by tuning the parameters or reducing iterations and/or the rotated angle.

5. COMPARISON WITH OTHER PROPOSALS

Two Fast HT CORDIC architectures are considered for comparison: a segmented reconfigurable implementation [13] and a parallel one [14].

5.1. CBRM versus segmented CORDIC.

The segmented CORDIC architecture and the CBRM, have been tested on the same platform (V300 BG352) using a *128x128* image with 128 discrete angles; the times invested have been picked up in tables 3 and 4.

bits	**Overall**	**Frames/s**
8	28.656ns · 128 · 128 · 64 =0.03005 s	33
16	41.088ns · 128 · 128 · 64 =0.04308 s	23
32	139.648ns · 128 · 128 · 64 =0.14643 s	6

Tabla 3. Processing times for CBRM.

bits	**Operation +/-**	**Overall**	**Complete image**	**Frames/s**
8	11.076 ns	132.912 ns	0.139 s	7
16	25.308 ns	303.696 ns	0.318 s	3
32	53.772 ns	1720.704 ns	1.804 s	0.5

Table 4. Times obtained for segmented CORDIC.

As shown in Tables 2 and 3, CBRM provides good results when comparing with segmented CORDIC according to the experiments carried out with the HT for line detection. In the segmented CORDIC implementation, the global error ($E = 2N \cdot 2^{-(n-1/2)} + 2^{-M} \cdot n$) falls with the number of bits of the fractional part M and grows with the number of iterations, *n*, when *n>16*. According to [13], the absolute error for *N=128* is of 0.135, for 16 bits data (with 8 fractional) and 12 iterations. It is the same than having 6 erroneous fractional bits in the result. That represents an approximate relative error of $2^{-5}=3\%$. The CBRM has a relative error smaller than 1% for the same precision and values so, therefore presents a better behaviour.

The amount of hardware resources used by the segmented CORDIC architecture is higher than that of the CBRM, which involves two multiplexers, two displacement registers, 2 MACs of two inputs and 2 16n bit LUTs (Fig. 2 and 3). The segmented CORDIC implementation with 12 iterations needs 24 adder-substracter circuits, 24 fixed displacement registers, 24 registers and a ROM table containing the microrotations, plus two supplementary adder circuits to carry out the serial compensation of the scale factor.

5.2. CBRM versus parallel CORDIC.

Both architectures have been implemented on the same platform (V300 BG352) for comparison. An *NxN* image needs $N^2 \cdot N_\theta / 4m$ calculation cycles, where N_θ is the number of rotation angles and *m* the number of processors. The stage-time, without considering the multiplexer time, is that of the add/subtraction stage. Table 5 shows the time taken to transform a *128x128* image with 128 rotation angles and *m* processors. An inverse proportion exists between number of processors and calculation time; the number of processors considered aims for reaching the CBRM times (Table 3).

The comparison as for hardware resources is highly unfavourable in the case of the parallel implementation since it demands many more resources to obtain an equivalent speed to the implementation CBRM. The mentioned work does not provide data on the error performed by the implementation presented.

It suits to mention that the implementation of the CBRM can achieve even more speed when proposing four functions $(\rho_I)_k$ $(\rho_{II})_k$ $(\rho_{III})_k$ $(\rho_{IV})_k$: the angle to rotate would be the half, and, for the same increment, the number of iterations would be divided by two. This improvement would provide additional time-saving by means of duplicating the hardware used.

bits	Add/ Subst.	10 stage cycle	Complete image	Proc (m)
8	11.076ns	0.111ms	128·32·32·0.111/m=58.196s/m	1937
16	25.308ns	0.253ms	128·32·32·0.253/m=132.654s/m	3079
32	53.772ns	0.538ms	128·32·32·0.538/m=282.067s/m	1926

Table 5. Times obtained for the paralell CORDIC architecture.

6. CONCLUSIONS

A function evaluation method that provides calculation improvements for the HT has been presented (CBRM). It outlines the interest of convolution as a powerful tool that increases computational capabilities, essentially when applied to massive calculations. The evaluation of the HT has been carried out with a DA-based architecture. Compared with other well-known proposals, namely segmented and parallel CORDIC, it has been confirmed that the CBRM provides memory and hardware resource saving as well as speed improvements according to the experiments carried out with the HT. It also offers good results in dependable-precision error. These encouraging partial conclusions make quite reasonable to study in depth the capabilities of convolution to provide a more complete set of recursive evaluating patterns in order to extend the CBRM.

7. REFERENCES

[1] Muamar, H.K and Nixon, M., "Tristage Hough Transform for multiple ellipse extraction," *IEEE Proc. Part E: Computer and Digital Techniques*, Vol 138 nº 1, 1991.

[2] Haule, D.D and Malowany, A.S., "Object Recognition using fast adaptative HT," *IEEE Comp. Pacific Conf. On Communication, Compiler and Signal Processing*, pp. 91-94, 1989.

[3] da Silva, I., "Vectorization from aerial photographs applying the HT method," *Proc. SPIE*, Vol 1395, Pt2, pp. 956-963, 1990.

[4] Koshimizu, H. and Numada, M., "On fast Hough Transform method PLHT based on piecewise-linear Hough function," *J. System Computer in Japan*, Vol 21 nº5, pp. 62-73, 1990.

[5] Ben-Tzvi, D. and Sandler, M. "A Combinatorial Hough Transform". *J.P. Recognition Letters*, Vol 11,pp. 167-174, 1990

[6] da Fontura, L. and Sandler, M.B., "A binary HT and its efficient implementation in a systolic array architecture," *J.P. Recognition Letters*, Vol. 10, pp. 329-334, 89.

[7] Walther, J.S., "A unified algorithm for elementary functions," *Proc. Spring Joint Computers Conf.*, pp. 379-385, 1971.

[8] Li, H.F., Lavin, M.A. and Le Master, R.J., "Fast Hough Transform: a hierarchical approach," *J. Computer Vision Graphics Image Processing*, Vol.36, pp. 139-161, 1986.

[9] Hu, Y.H., "CORDIC-based VLSI architectures for Digital Signal Processing," *IEEE Signal Processing Magazine*, nº 7 pp. 16-35, 1992.

[10] Villalba, J., "Diseño de Arquitecturas CORDIC multidimensionales," *Tesis Doctoral Dept. de Arquitectura de Computadores. Universidad de Málaga*. Nov.1995.

[11] Shankar, R.V. and N. Asokan., "A parallel implementation of the Hough Transform method to detect lines and curves in pictures," *IEEE 32th Midwest Symp. On Circuits and Systems*, pp. 321-324, 1990.

[12] Chuang ,H.Y.H. and Li, C.C. "A systolic processor for straight line detection by modified HT". *IEEE Conf. on Computer Architecture for Pattern Analysis and Image Database Management*, pp. 300-304. 1995.

[13] Deng, Dixon, D.S and El Gindy H., "High speed Parametrizable HT using reconfigurable hardware," *Pan-Sydney Area Workshop an Visual Information Processing (VIP)*, 2001.

[14] Bruguera, J.D. and Guil, N. "CORDIC-based parallel/pipelined architecture for the HT," *J. of VLSI Signal Processing*, Vol 12 nº 3, pp. 207-221, 1996.