© Journal of the Technical University - Sofia
Plovdiv branch, Bulgaria
"Fundamental Sciences and Applications"
Vol. 25, 2019

# FPGA based Canny: advanced gradient direction algorithm focused on optimal speed and total mathematical accuracy

**Dimitre Kromichev**

Department of Marketing and International Economic Relations, University of Plovdiv, 24 Tzar Asen Street, Plovdiv 4000, Bulgaria

dkromichev@yahoo.com

**Abstract**. The paper presents an advanced algorithm guaranteeing optimal speed execution of FPGA based Canny gradient direction computations on the platform of total mathematical accuracy. The proposed algorithm's speed capabilities are experimentally tested and analyzed in terms of the two capital parameters: maximum clock frequency and minimum clock cycles required for obtaining a mathematically exact result at the highest clock rate. Proved is the algorithm's relevance to designing a novel organization of FPGA based Canny computations targeting optimal performance.

## 1. Introduction

The two basic requirements for FPGA based Canny are: optimizing execution speed and guaranteeing detected contours' precision. The former depends on both the maximum clock frequency Canny is capable of operating at and the minimum number of clock cycles needed to secure an exact result at the highest achievable clock rate. Mapped contours' genuineness is a function of the mathematical accuracy of calculations in every single Canny module.

Gradient direction has a key role in computing the exact local maxima. Therefore, all of the obtained gradient direction values should be mathematically accurate. The integer arithmetic utilized in the gradient direction computations should be up to the highest clock frequency FPGA based Canny is executed at. In FPGA based Canny gradient direction and gradient magnitude computations are executed in parallel. Non-maximum suppression is a neighbourhood operation whose pipelining efficiency depeds on the number of clock cycles required to calculate a gradient direction value.

The objective is to present an advanced algorithm capable of guaranteeing optimal speed execution of FPGA based Canny gradient direction computations on the platform of total mathematical accuracy. The task is to describe the proposed algorithm in terms of its computational mechanism and explore its performance with respect to the two capital speed parameters: maximum clock frequency and minimum clock cycles required for obtaining a mathematically exact result at the highest clock rate. Relevant to the conducted experiments and conclusions arrived at are only gray-scale images. The software tool employed to ascertain the differences between gradient direction exact mathematics and the available computational approaches is Scilab. The targeted hardware is Intel (Altera) FPGAs. The following ten Intel (Altera) FPGA families are used in the tests: 130 nm, 90 nm, 65 nm, 40 nm, 28nm Cyclone, Cyclone II, III, IV, V; 130 nm, 90 nm, 65 nm, 40 nm and 28nm Stratix, Stratix II, III, IV, V. Intel (Altera) Quartus, ModelSim and TimeQuest Timing Analyzer are utilized for exploring the feasibility and practicality of the proposed algorithm.

 Online ISSN 2603-459X

## 2. Survey of the available approaches

Inasmuch as there are eight surrounding pixels in the 3x3 neighborhood, there are four possible axes determining the gradient directions. Therefore, all calculation results are reduced to four values: 0, 45, 90, 135. They represent bisectors of eight angular sectors, each of them encompassing exactly 45°.

So far, there has been no mathematically accurate FPGA hardware inplementation of gradient direction. In the literature [3][7], it is generally stated that calculating gradient direction by using the exact mathematics

$$G_D = \tan^{-1}\left(\frac{G_y}{G_x}\right),$$

where

$Gy$ is the $y$ gradient, and $G_y \in [-255,255]$,
$Gx$ is the $x$ gradient, and $G_x \in [-255,255]$ (1)

is very difficult to inplement in hardware, and (1) is too slow to be used in real-time applications. On that basis, until now, in all of the described FPGA based Canny realizations (1) is replaced with approximation. The latter is represented by four main variants [1][2][4][5][6]. Their deviations from (1) for the four relevant directions: $x$- and $y$-axes, positive and negative diagonals, are obtained by using purposely designed programs written in Scilab. The results in both numerical and percentage terms are exhibited in the tables below.

**Table 1.** Difference between the total number of gradient direction values and the number of results calculated by utilizing (1) and approximation variants. ( ‘’-’’ denotes that the value is smaller and ‘’+’’ denotes that the value is larger than the number of all possible combinations).

| Approach | *Gx* | *Gy* | Total number of gradient direction values in [-255, 255] | Total number of results for gradient directions 0, 45, 90, 135 | Difference in total number of results in numerical terms | Difference in total number of results in percentage terms |
|---|---|---|---|---|---|---|
| Exact math | [-255, 255] | [-255, 255] | 261121 | 261121 | 0 | 0% |
| Approx. var. #1 | [-255, 255] | [-255, 255] | 261121 | 206659 | - 54462 | - 20.8% |
| Approx. var. #2 | [-255, 255] | [-255, 255] | 261121 | 261122 | + 1 | 0% |
| Approx. var. #3 | [-255, 255] | [-255, 255] | 261121 | 262144 | + 1023 | + 0.4% |
| Approx. var. #4 | [-255, 255] | [-255, 255] | 261121 | 106696 | - 154425 | - 59.1% |

**Table 2.** Difference between gradient direction values calculated by (1) and those computed by approximations. The number for direction 0 includes all cases wherein $Gy = 0$ and $Gx = 0$. (‘’-’’ denotes that the number of results calculated by (1) is larger and ‘’+’’ denotes that the number of results calculated by (1) is smaller than the number computed by using approximations).

| Approach | *Gx* | *Gy* | Total number of results | Ascertained number of values by directions | | | | Difference by directions in numerical terms | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 45 | 90 | 135 | 0 | 45 | 90 | 135 |
| Exact math | [-255, 255] | [-255, 255] | 261121 | 66046 | 65025 | 65025 | 65025 | - | - | - | - |
| Approx. var. #1 | [-255, 255] | [-255, 255] | 206659 | 49471 | 78736 | 0 | 78452 | - 3366 | + 349 | - 51816 | + 371 |
| Approx. Var. #2 | [-255, 255] | [-255, 255] | 261122 | 130562 | 511 | 129540 | 509 | + 77725 | - 77876 | + 77724 | - 77572 |
| Approx Var. #3 | [-255, 255] | [-255, 255] | 262144 | 130562 | 1021 | 129540 | 1021 | + 77725 | - 77366 | + 77724 | - 77060 |
| Approx. var. #4 | [-255, 255] | [-255, 255] | 106696 | 53349 | 511 | 52327 | 509 | + 512 | - 77876 | + 511 | - 77572 |

**Table 3.** Difference between gradient direction values calculated by (1) and those computed by approximations The values for direction 0 include all cases wherein *Gy*=0 and *Gx*=0. (‘’-’’ denotes that the number of values calculated by (1) is larger and ‘’+’’ denotes that the number calculated by (1) is smaller than the number computed by using approximation).

| Approach | *Gx* | *Gy* | Total count of results | Ascertained number of values by directions | | | | Difference by directions in percentage terms | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0 | 45 | 90 | 135 | 0 | 45 | 90 | 135 |
| Exact math | [-255, 255] | [-255, 255] | 261121 | 66046 | 65025 | 65025 | 65025 | - | - | - | - |
| Approx. var. #1 | [-255, 255] | [-255, 255] | 206659 | 49471 | 78736 | 0 | 78452 | - 6.4 % | + 0.5% | - 100% | + 0.5% |
| Approx. Var. #2 | [-255, 255] | [-255, 255] | 261122 | 130562 | 511 | 129540 | 509 | + 59.5% | - 99.4% | + 60% | - 99/4% |
| Approx. var. #3 | [-255, 255] | [-255, 255] | 262144 | 130562 | 1021 | 129540 | 1021 | + 59.5% | - 98.7% | + 60% | - 98.7% |
| Approx. var. #4 | [-255, 255] | [-255, 255] | 106696 | 53349 | 511 | 52327 | 509 | + 1 % | - 99.4% | + 1% | - 99/4% |

Tables 1, 2 and 3 show that all approximations demonstrate a huge shortage of calculated results, as well as sharply disproportional distribution of computed values by directions. They are completely inappropriate for demanding applications of FPGA based Canny.

## 3. The proposed gradient direction algorithm

The algorithm includes the following sequence of steps:
1) Determine all possible combinations between the signs of *Gy* and *Gx*. Their number is $2^2$.
2) The combinations from *step 1)* are divided into two sets.
3) One of the sets from *step 2)* includes the following relations:

$$Gy > 0 \;\&\; Gx > 0 \qquad Gy < 0 \;\&\; Gx < 0 \;. \tag{2}$$

4) The other set defined in *step 2)* includes the following relations:

$$Gy > 0 \;\&\; Gx < 0 \qquad Gy < 0 \;\&\; Gx > 0 \;. \tag{3}$$

5) Determine the two angles from the unit circle that define four angular intervals in which the *x* and *y* axes, the positive and negative diagonals are bisectors.. These angles are 22.5° and 67.5°.
6) Determine the exact numerical equivalent to angle 22.5°. The accurate representation of angle 22.5° is the fraction $\frac{99}{239}$ ($\tan^{-1}\left(\frac{99}{239}\right)$ = 22.500605394851°). Then, determine the exact numerical equivalent to angle 67.5°. The accurate representation of angle 67.5° is the fraction $\frac{169}{70}$ ($\tan^{-1}\left(\frac{169}{70}\right)$= 67.500605394851°). Figure 1 shows that.

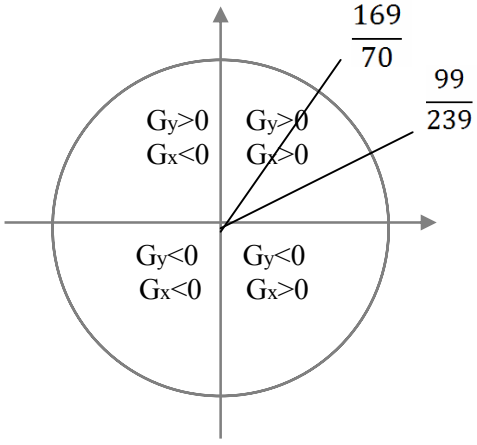

**Figure 1.** The two reference points and sign options for dividend and divisor

7) Takimg into account the set of sign relations defined in *step 3)*, the gradient direction calculation is based on

$$\begin{aligned} &Gy > 0 \ \& \ Gx > 0 \\ &Gy < 0 \ \& \ Gx < 0 \\ &\text{If} \quad |Gx|*99 \geq |Gy|*239 && G_D = 0. \\ &\text{If} \quad |Gx|*99 < |Gy|*239 \ \& \ |Gx|*169 > |Gy|*70 && G_D = 45. \\ &\text{If} \quad |Gx|*169 \leq |Gy|*70 && G_D = 90. \end{aligned} \tag{4}$$

8) With respect to the set of sign relations defined in *step 4)*, the gradient direction calculation is based on:

$$\begin{aligned} &Gy > 0 \ \& \ Gx < 0 \\ &Gy < 0 \ \& \ Gx > 0 \\ &\text{If} \quad |Gx|*99 \geq |Gy|*239 && G_D = 0. \\ &\text{If} \quad |Gx|*99 < |Gy|*239 \ \& \ |Gx|*169 > |Gy|*70 && G_D = 135. \\ &\text{If} \quad |Gx|*169 \leq |Gy|*90 && G_D = 90. \end{aligned} \tag{5}$$

In Intel (Altera)| FPGAs, comparison function is considerably slower than addition, subtraction and hard multiplication for equal input data widths. Therefore, to achieve maximum speed in FPGA, the comparison function in (8) and (9) is replaced with subtraction and checking the result for being greater than or equal to zero.

With respect to the very important fact that both Gy and Gx can be equal to zero, the comlete set of relations between the y- and x-gradients requires that the following conditions be fulfilled so that all the possible combinations within the interval [-255,255] could be correctly encompassed by the algorithm:

$$\begin{aligned} &\text{If} \quad Gy = 0 \ \& \ Gx \neq 0 && G_D = 0. \\ &\text{If} \quad Gy \neq 0 \ \& \ Gx = 0 && G_D = 0. \\ &\text{If} \quad Gy = 0 \ \& \ Gx = 0 && G_D = 0. \end{aligned} \tag{6}$$

The computational mechanism of the algorithm is presented in the figure below:

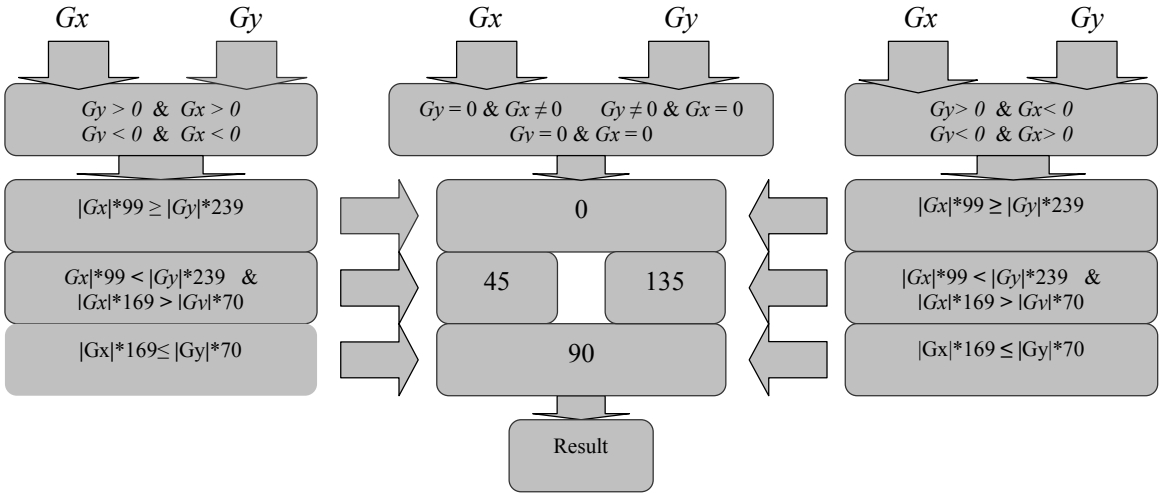


**Figure 2.** Functional model of the proposed gradient direction algorithm

## 4. Exploring the algorithm's mathematical accuracy

Testing mathematical accuracy of the proposed algorithm has to take into account both the values of the involved *x*- and *y*-gradients and the sign relations expressed in (2) and (3).

Check # 1

Gy = 50 Gx = 40

40*99 ≥ 50*239 (false)

40*99 < 50*239 & 40*169 > 50*70 (true)

40*169 ≤ 50*80 (false) .

Therefore, $G_D$ = 45 . Using the conventional method: $\tan^{-1}\left(\frac{50}{40}\right)$ = 51.3401917°.

Check # 2

Gy = -48 Gx = 55

55*99 ≥ |-48|*239 (false)

55*99 < |-48|*239 & 55*169 > |-48|*70 (true)

55*169 ≤ |-48|*70 (false).

Therefore, $G_D$ = 135 . Using the conventional method : $\tan^{-1}\left(\frac{-48}{55}\right)$ = - 41.1120904°.

The total mathematical accuracy of results is guaranteed for all integers within [-255,255]. The cases wherein dividend and divisor in (1) are equal to zero – 1021 in total, are to be considered separately according to (10). The whole bulk of calculated gradient direction values based on the positive and negative integers – 260100 in total, is quite equally distributed among the four intervals defining the four gradient directions. Each interval contains 65025 values resulting from the exact implementation of (1). Therefore, this proves that the two reference points – $\frac{99}{239}$ and $\frac{169}{70}$, representing the angles 22.5° and 67.5° are most appropriately selected. They are up to the goal of guaranteeing total mathematical accuracy.

## 5. Exploring the algorithm's speed capabilities in FPGA: results and analysis

Assessing the proposed algorithm's performance is based on the following methodology: 1) Select an FPGA device from each of the 10 targeted Intel (Altera) FPGA families using Quartus to conduct tests; 2) Write VHDL programs implementing the proposed algorithm and the approximation variants using the values within [-255,255]; 3) Analyze the algorithm's speed on a comparative basis using ModelSim and TimeQuest Timing Analyzer. The achieved results are exhibited in the table below.

**Table 4.** Capital speed parameters for the gradient direction calculation for the proposed algorithm and the available approximation variants.

| FPGA family | Values | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Capital speed parameters for the gradient direction calculation | | | | | | | | | |
| | Maximum clock frequency (MHz) | | | | | Total number of clock cycles required to secure result | | | | |
| | Proposed algorithm | Approx var. #1 | Approx var. #2 | Approx var. #3 | Approx var. #4 | Proposed algorithm | Approx var. #1 | Approx. var. #2 | Approx var. #3 | Approx var. #4 |
| Cyclone | 192 | 57 | 149 | 151 | 59 | 3 | 6 | 7 | 5 | 6 |
| Cyclone II | 242 | 68 | 190 | 192 | 71 | 3 | 6 | 7 | 5 | 6 |
| Cyclone III | 343 | 80 | 234 | 237 | 82 | 3 | 6 | 7 | 5 | 6 |
| Cyclone IV | 359 | 84 | 228 | 232 | 84 | 3 | 6 | 7 | 5 | 6 |
| Cyclone V | 367 | 88 | 231 | 235 | 88 | 3 | 6 | 7 | 5 | 6 |
| Stratix | 286 | 71 | 234 | 239 | 75 | 3 | 6 | 7 | 5 | 6 |
| Stratix II | 368 | 116 | 302 | 307 | 118 | 3 | 6 | 7 | 5 | 6 |
| Stratix III | 484 | 142 | 426 | 433 | 147 | 3 | 6 | 7 | 5 | 6 |
| Stratix IV | 522 | 162 | 442 | 453 | 167 | 3 | 6 | 7 | 5 | 6 |
| Stratix V | 557 | 188 | 464 | 471 | 192 | 3 | 6 | 7 | 5 | 6 |

In Table 5 presented are the resources required by the proposed algorithm to be executed in Intel (Altera) Cyclone V E 5CEBA4F17C6N Device**.** The exibited data proves that the algorithm's advanced computational mechanism guarantees its being very economical in utilizing the FPGA resources.

**Table 5.** Resources utilized by the proposed algorithm in Intel (Altera) Cyclone V E 5CEBA4F17C6N Device.

| FPGA Resource | Resource Counts | Used | Resource utilization |
|---|---|---|---|
| Logic utilization (in ALMs ) | 18480 | 75 | 0.4058 % |
| Total registers | - | 118 | - |
| Total block memory bits | 3168512 | 0 | 0 % |
| Total DSP blocks | 66 | 2 | 3.0303 % |

The analysis of the test results is as follows:

1) Multiplication being the slowest integer arithmetic in the proposed algorithm, the optimal clock frequency of its execution is defined by the 9x9 hard multiplier's performance.

2) The optimal speed of the entire FPGA based Canny is determined by the optimal 18x18 hard multiplier's performance, the range of magnitudes of Gaussian filters' coefficients having been taken into account. In the targeted Intel (Altera) FPGA families, the 9x9 hard multiplier is from 14.2% to 16.4% faster than the 18x18 hard multiplier. Therefore, under all test conditions the proposed algorithm is capable of working accurately at a clock frequency higher than the maximum clock rate of the entire FPGA based Canny. This proves that it is optimal in the clock frequency domain.

3) Approximation variants #1 and #4 use division. Therefore, their maximum clock frequencies are very low. Approximations #2 and #3 are considerably slowed down by the comparison function.

4) In the proposed algorithm all multiplicaton and subtraction operations are performed simultaneously. Thus, its execution requires only 3 clock cycles at the highest clock frequency.

The two capital speed parameters having been assessed, the proposed algorithm is from 4.6 to 8.2 times faster than the available approximation variants. Therefore, it is optimal in terms of speed.

## 6. Conclusion

Presented is an advanced gradient direction algorithm focused on optimal speed and total mathematical accuracy. The algorithm's computational mechanism is set forth in detail. Its total mathematical accuracy is proved. Scrutinized are the algorithm's speed capabilities in FPGA on a comparative basis. The test results for the two capital speed parameters prove that the proposed algorithm is the only computational technology which can work accurately at clock frequencies higher than the maximum clock frequencies the entire FPGA based Canny executes at in the targeted Intel (Altera) FPGA platforms. Therefore, the proposed algorithm is optimal in the speed domain.

## References

[1] Aravindh G. and Manikandababu C. S. 2015 Algprithm and Implementation of Distributed Canny Edge Detector on FPGA *ARPN Journal of Engineering and Applied Sciences* Vol. **10** (7), pp.3208-3216

[2] Chandrashekar N.S., K.R. Nataraj 2013 Image Smoothening Gradient Magnitude and Hystersis Calculation for Canny Edge Detector Using FPGA for Area Optimization *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)* Vol. **2** (2), pp. 5-9

[3] Divya. D, Sushmap S. 2013 FPGA Implementation of a Distributed Canny Edge Detector *International Journal of Advanced Computational Engineering and Networking*, Vol.**1** (5) pp.46-51

[4] Fangxin Peng, Xiaofeng Lu, Hengli Lu, Sumin Shen 2012 An improved high-speed canny edge detection algorithm and its implementation on FPGA *Proceedings of SPIE, Pattern Recognition and Basic* Technologies Conference Vol. **8350**

[5] Supraya K, 2017 Hardware Implementation Of Canny Edge Detection Algorithm With FPGA *Journal of Engineering Science and Technology* Vol. **12**, No. 9, pp. 2536-2550

[6] Pallavi Ramgundewar, Hingway S .P., Mankar K. 2015 Design of modified Canny Edge Detector based on FPGA for Portable Device *Journal of The International Association of Advanced Technology and Science* Vol. **16** (2), pp. 10-16

[7] Poonam S. Deokar and Anagha P. Khedkar 2015 Implementation of Modified Distributed Canny Edge Detector Algorithm Using FPGA *International Journal of Information Research and Review* Vol. **2** (8), pp. 999-1003