ACCURACY OF MEASUREMENTS IN TERMS OF FPGA BASED CANNY EDGE DETECTION

Dimitre Kromichev

Abstract: The paper focuses on the capabilities of Canny edge detection to be used in taking accurate measurements required for exercising reliable quality inspection on the production lines. Four new algorithms are proposed. They are scrutinized in terms of mathematical exactness as a tool for achieving the targeted veracity of detected contours particularly with respect to their being precise in shape and localisation. FPGA’s functionalities define the hardware platform and framework for the advanced computational characteristics of the presented algorithms in terms of their efficiency and applicability, thus enhancing the total performance of FPGA orientated Canny aimed at accomplishing the goal of conducting plausible quality inspection to guarantee an item’s compliance with the production standards.

Key words: Canny, FPGA, accuracy, measurement, algorithm, shape, localization

1. Introduction

Field Programmable Gate Array (FPGA) based Canny edge detection can efficiently be utilized in taking accurate measurements for the purpose of conducting reliable quality inspection on the production lines. Applying Canny to the digitized image of a surveyed item results in mapping the contours of a particular geometric figure which represents the input to a highly specialized computational algorithm measuring the figure for compliance with the technological specifications. That imposes the peremptory demand for total veracity of detected contours. The latter is a function of several parameters, the most outstanding among them being shape and localization. Both shape and localization require that the computational algorithms within the Canny modules should guarantee optimal mathematical exactness of results.

The objective of this paper is to present four new algorithms aimed at optimizing the mathematical precision of FPGA orientated Canny computations to be used as a reliable platform for accurate measurements on the production lines. These algorithms are focused on gradient magnitude and direction calculation, dynamic computation of high and low thresholds, and hysteresis thresholding. The task is to describe in detail the sequence of steps, to thoroughly analyze the mathematical reliability, to expose the characteristics, and to point out the applicability of the proposed algorithms in compliance with the specifics of FPGA implementation. A highly specialized software tool and a set of real life images are used to test the algorithms’ applicability. The targeted hardware is Altera FPGAs. Relevant to the analyses and conclusions arrived at in this paper are only gray-scale images.

2. Literature survey

The implementation of Pythagoras aimed at computing the gradient magnitude in the FPGA orientated Canny which is described in the literature [8][11][12][14][15] relies on approximation. Although fast, this technique is very inaccurate. So far, in the literature there has been no algorithm addressing both the requirement for mathematical exactness and the demand for speed.

To avoid the FPGA implementation of inverse tangent function, proposed are techniques relying entirely upon the vertical or horizontal values’ being positive or negative [13]. This approach does not provide the necessary correctness of gradient direction calculations.

For the high and low thresholds, in [7], uniformly quantized gradient magnitude histograms are computed on overlapped blocks. An intermediate classification threshold is calculated based on a set of pixels with gradient magnitudes larger than a defined value. The high threshold is computed on the basis of the histograms. In [9][10], the OTSU method is applied to adaptively calculate high and low thresholds by basically splitting all the image pixels into two classes, and computing the best threshold value through the variance maximum value between the two classes. The approach here is to select a threshold which minimizes the within-class variance or maximizes the between-class variance. In [1], a low-complexity 8-step non-uniform gradient magnitude histogram is used to compute block-based hysteresis thresholds. These approaches share the flaw of computing the high and threshold values in a sequential mode, thus impeding the pipelining efficiency.

For hysteresis thresholding, in the literature
described are few approaches addressing FPGA implementation. Hysteresis thresholding relies on a moving window, two comparators, FIFO buffer and OR gates [13][16]. All of them depend heavily on recursive computations.

3. Proposed algorithms

Canny edge detection includes five modules: Gaussian smoothing, computing the orthogonal gradients, computing gradient magnitude and direction, non-maximum suppression, hysteresis thresholding. In terms of accuracy, advanced algorithms for gradient magnitude and direction, non-maximum suppression and hysteresis thresholding are set forth.

3.1. Gradient magnitude algorithm

Mathematically, gradient magnitude \( G_M \) is computed using

\[
G_M = \sqrt{G_x^2 + G_y^2},
\]

(1)

where

\( G_x \) and \( G_y \) are the x and y gradients.

On FPGA, computing Pythagoras starts with multiplying \( |G_x| \) by \( |G_y| \) and \( |G_x| \) by \( |G_y| \). The largest positive pixel value being 255, the results of both multiplications are within the interval \([0, 65025]\).

In the next step, the multiplication results are added, and consequently the numbers to be square rooted are within \([0, 130050]\).

Thus, all the possible values at the input of the square root function are 130051. And all the possible values at its output are \(2^8\), taking into account that only numbers in the interval \([0, 255]\) are relevant for pixels in the gray scale matrix. Consequently, the exact mathematical results of the square root function can be determined by 256 closed intervals, each of them representing a single integer from the interval \([0, 255]\). These intervals contain all the 130051 values calculated by adding the squared gradients.

Defining the smallest and the largest in each interval is based on the fact that the difference between squares of two consecutive integers is equal to the smaller integer multiplied by 2, and then 1 is added to the result. Thus, the smallest of all consecutive values contained in an interval is determined through:

1) squaring the gray scale image pixel value represented by this particular interval;
2) the gray scale image pixel value represented by this particular interval is decremented, and the result is subtracted from the result computed in 1).

The largest of all consecutive values in an interval is determined through:

1) squaring the gray scale image pixel value represented by this particular interval;
2) the gray scale image pixel value represented by this particular interval is subtracted from the result computed in 1). The only exception is the value in the interval representing 255 – here the largest possible number is 130050.

Apart from ensuring a hundred per cent mathematical accuracy, the algorithm also guarantees maximum speed, taking into account that the comparisons of all values for computing the gradient magnitude are executed simultaneously on FPGA [2][3][4][5][6], utilizing one of the most significant capabilities of hardware implementation – parallel computations.

3.2. Gradient direction algorithm

Gradient direction is calculated by means of

\[
g_D = \tan^{-1}\left(\frac{G_y}{G_x}\right),
\]

(2)

where

\( G_x \) is the y gradient, and \( G_y \in [-255, 255] \), \( G_x \) is the x gradient, and \( G_x \in [-255, 255] \).

There are 4 possible axes determining the gradient directions. Consequently, all the calculation results are reduced to 4 values: 0, 45, 90, 135. Thus, (2) is replaced with the ratio between \( G_y \) and \( G_x \).

The ratio is calculated on the basis of two reference points and the signs of \( G_y \) and \( G_x \). One of the reference points is used to represent an angle of 22.5°, and the other - 67.5°. The most appropriate approximation to angle 22.5° is the fraction \( \frac{2}{5} \) (\( \tan^{-1}\left(\frac{2}{5}\right) = 21.8014094° \)), and to angle 67.5° - the fraction \( \frac{2}{5} \) (\( \tan^{-1}\left(\frac{2}{5}\right) = 68.1985905° \)).

The computational algorithm is based on:

1) \( G_y > 0 \) & \( G_x > 0 \)
   \( G_y < 0 \) & \( G_x < 0 \)
   
   \[ G_D = 0 \]

2) \( |G_y|^2 < |G_x|^5 \)
   \( |G_x|^2 > |G_y|^5 \)
   \( G_D = 45 \)

3) \( |G_y|^5 \) & \( |G_x|^2 \)
   \( G_D = 90 \)

4) \( G_y > 0 \) & \( G_x < 0 \)
   \( G_y < 0 \) & \( G_x > 0 \)

124
If $|G_x|^2 |G_y|^5 = 0$  
If $|G_x|^2 < |G_y|^5$  
If $|G_x|^5 > |G_y|^2$  
$G_D = 0$  
$G_D = 135$  
$G_D = 90.0$

(3)

The algorithm ensures optimal mathematical accuracy and guarantees high speed performance taking into account that all comparisons and multiplications are executed simultaneously on FPGA.

3.3. Algorithm for dynamic computation of high and low thresholds

The algorithm covers the following sequence of steps:

1) Defining the range of consecutive positive values the high threshold calculation will be based upon. It is within a closed interval whose minimum and maximum values are selected in such a way that the difference between the maximum decremented by 1 and the minimum is a positive number whose factor is 7. The minimum is as many as three times smaller than the largest pixel value.

2) The closed interval from step 1) is divided into equal subintervals whose number is as many as three times bigger than 7. Each subinterval encompasses 7 consecutive values, and is associated with one reference value which is equal to the fourth out of the seven consecutive values contained in this subinterval.

3) Each of the reference values defined in step 2) is paired with a table value that is 2.8 times smaller than the reference value.

4) Defining a set of counters. Each counter is related to one of the subintervals from step 2).

5) Starting with image pixel #1 being processed, each pixel is assessed with respect to its value falling/not falling within one of the subintervals from step 2). If the pixel value is within an interval, the counter associated with this particular interval is incremented by 1.

6) With all the pixels gone through the assessment procedure from step 5), the values in each of the 16 counters defined in step 4) are compared to select the largest among them. On the basis of this largest value, one of the subintervals, and the reference value associated with it, as defined in step 2), is further selected. This reference value is the high threshold.

7) The table value defined in step 3) that is paired with the reference value as calculated in step 6) is selected. This table value is the low threshold.

The algorithm ensures optimal mathematical accuracy and maximum speed in terms of its dynamic computational parameters aimed at enhancing pipelining efficiency on FPGA.

3.4. Hysteresis thresholding algorithm

The high threshold $T_1$ and the low threshold $T_2$ having been calculated, the hysteresis thresholding computations are focused mainly upon the connectivity/non-connectivity determination with respect to $T_1$. The algorithm is presented by the following sequence of steps:

1) The value of each pixel is tested for being an edge (bigger or equal to $T_1$), not being an edge (less or equal to $T_2$), and being a possible edge (smaller than $T_1$ but bigger than $T_2$).

2) A 3x3 window centered around each pixel satisfying the condition for being a possible edge is applied. All the neighbouring pixels – maximum eight, are tested in reference to $T_1$.

3) If there is a value bigger than $T_1$, the central pixel from step 2) is defined as an edge and its coordinates in the image matrix are stored in a separate memory buffer to avoid redundant double check of up to five magnitudes when the window moves on to the next pixel.

4) If none of the neighbouring magnitudes is bigger than $T_1$, but at least one falls between $T_1$ and $T_2$, only peripheral magnitudes of a 5x5 window are checked for values bigger than $T_1$.

5) If such a value is available, then it is an edge pixel.

6) If one or more of these sixteen peripheral magnitudes in the 5x5 window falls between $T_1$ and $T_2$, then its (their) coordinates are stored in another memory buffer.

7) If such a magnitude is not available, then the central pixel from 2) is not an edge.

8) The procedure encompassing steps 1) through 7) is repeated for the next pixel in the image row, taking into account that from the second pixel downwards the contents of the two memory buffers are constantly checked. The second buffer contains those pixels which can eventually be determined as edge pixels on the basis of indirect connectivity.

4. Assessing the algorithms applicability in terms of accuracy

A software simulation using Scilab version 5.4.1, and Scilab Image and Video Processing Toolbox version is 0.4.3, is the appropriate tool to test the proposed advanced algorithms’ reliability.
and efficiency in terms of mathematical accuracy of computational results and veracity of detected contours.

The Gaussian filter used in this simulation has $\sigma = 1.4$. A set of real-life images is utilized to demonstrate the edge detection characteristics of the proposed algorithms. All the images are of size 512x512, and the format is JPG. One of them is presented to show the simulation results.

1) The real life input image:

![Image 1]

2) The edge detected image:

![Image 2]

5. Assessing the detected contours

The mapped contours’ analyses exhibit the achievement of the following targeted characteristics:

1) Optimal precision of detected contours in all respects and particularly in terms of shape and localization which are the two most important quality parameters for demanding industrial applications.

2) Uncompromising mathematical accuracy.

3) Veracity of calculations as the indispensable basis for edges’ plausibility.

4) Appropriateness of the employed algorithms with respect to computational organization and efficiency.

6. Conclusions

Presented are four new algorithms aimed at the mathematical precision optimization of FPGA orientated Canny to be used as a reliable platform for accurate measurements on the production lines. These algorithms are focused on gradient magnitude and direction calculation, dynamic computation of high and low thresholds, and hysteresis thresholding. Described in detail is the sequence of steps, and the mathematical reliability is thoroughly analyzed. Exposed is the proposed algorithms’ applicability in compliance with the specifics of FPGA implementation. A set of real life images is used for evaluation. The targeted mathematical precision is assessed within the framework of detected contours’ quality.

7. References


[10] Mei Fang, Guang Xue Yue, Qing Cang Yu. The Study on An Application of Otsu Method in Canny Operator, Proceedings of the Inter-
national Symposium on Information Processing (ISIP’09), 2009, pp.109-112


Author:

Dimitre Zhivkov Kromichev, MSc in Computer Systems and Technologies Engineering, Assistant Professor in the Department of Marketing and International Economic Relations, Faculty of Economic and Social Sciences, Plovdiv University Paisii Hilendarski, Bulgaria. Scientific field and interests: digital image processing, Field Programmable Gate Arrays.

Workplace: Plovdiv University Paisii Hilendarski, Faculty of Economic and Social Sciences, Department of Marketing and International Economic Relations, Mailing address Plovdiv 4000 24 Tzar Asen Street. Web address: www.uni-plovdiv.bg E-mail address: dkromichev@yahoo.com