Available online at http://www.ijirr.com

ISSN: 2349-9141

International Journal of Information Research and Review
Vol. 2, Issue, 08, pp. 999-1003, August, 2015

OPEN ACCESS JOURNAL

Full Length Research Paper

IMPLEMENTATION OF MODIFIED DISTRIBUTED CANNY EDGE DETECTOR ALGORITHM USING FPGA

[*]Poonam S. Deokar and Anagha P. Khedkar

M.E Student, MCEORC, Nashik, India

***Corresponding Author**

Received 13th July 2015; Published 31st August 2015

Abstract

Edge can be defined as discontinuities in image intensity from one pixel to another. Edge detection is one of the most fundamental algorithms in digital image processing. Direct implementation of the canny algorithm has high latency and cannot be employed in real-time applications. To overcome these, an adaptive threshold selection algorithm is proposed. Distributed Canny Edge Detection using FPGA has high edge detection performance, but has high resource utilization. Here, to reduce the resource utilization we present modified distributed canny edge detection algorithm which uses approximate square root method. This algorithm divides the image into blocks and computes edges of the blocks in parallel. It is capable of removing excess edges in the image. Subjective test shows that performance of proposed algorithm is better than previous algorithm. Finally, it is implemented on Virtex-4 FPGA and synthesized using Xilinx ISE. The synthesized image uses 4 computation engines and obtains nearly 30% resource optimization in approximate calculation. To detect edges images from SIPI database are used.

Keywords: FPGA, Canny Edge Detector, Latency, Threshold.

***To cite this paper**:* 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, Issue, 08, pp. 999-1003. August, 2015.

INTRODUCTION

An edge of an image is jump in intensity. An ideal edge is the set of connected pixels each of which is located at an orthogonal step transition in grey level. The basic purpose of edge detection is to significantly reduce the amount of data in an image, while preserving the structural properties to be used for further image processing. Edge detection has been widely applied in various different computer vision systems. It is used to identify changes in luminosity of the image, changes in the intensity due to changes in scene structure. Using software, Edge detection algorithms are implemented, and their hardware acceleration is possible with Very Large Scale Integration (VLSI) technology, for real-time applications (Christos Gentsos, 2010). The Canny edge detector has better performance than previous algorithms. Canny edge detection algorithm is also known as the optimal edge detector (Wenhao He and KuiYuan, 2008). Canny's intentions were to enhance the many edge detectors in the image. Implementation of Canny algorithm with hardware has high latency and cannot be employed in real-time applications (Christos Gentsos, 2010). Another shortcoming of the commonly used Canny algorithm is that the computational cost of the algorithm is very high and it cannot be implemented in real time to meet the needs of mobile robot vision system.

Canny algorithm uses statistical analysis of complete image to calculate threshold values and it uses same high and low threshold value for entire image, which leads to some extra edges. For Canny algorithm performance can be improved by using self adaptive threshold calculation. In Distributed Canny algorithm, It divides image into different blocks and calculates threshold values for each block separately. All block can run in parallel, so reduces latency problem to some extent. But, Resource utilization increases accordingly. To overcome these shortcomings of this algorithm, a new approach is proposed in this paper, which computes the approximate value of gradient.

Approximate calculation reduces the cost of design as well as resource utilization is also optimized to great extent. Proposed algorithm is tested on the various images to analyze the performance of the algorithm. From the detail study of further literature survey, it has been identified that the performance of distributed Canny edge detector can be enhanced by using optimization algorithms viz. Genetic algorithm (GA) for resource optimization as well as for designing adaptive threshold. In addition to conventional GA, the novel GA operators viz. Basic twin operator, advanced twin operator may be incorporated to get better optimization results at reduced computational costs (Anagha Parag Khedkar, 2009; Anagha

Parag Khedkar and Dr. Shaila Subbaraman, 2010; Anagha Parag Khedkar and Dr. Shaila Subbaraman, 2010Anagha Parag Khedkar and Dr. Shaila Subbaraman, 2010).

Canny Edge Detection

The block diagram of the canny edge detection algorithm is shown in Fig. The Canny algorithm (Canny Edge Detection, 2009) consists of the following steps:

- Smoothing the input image.
- Calculating the horizontal gradient Gx and vertical gradient Gy at each pixel location.
- Computing the gradient magnitude G and direction θG at each pixel location.
- Applying Non-Maximal Suppression (NMS) to thin edges.
- Computing high and low thresholds based on the histogram of the gradient magnitude for the entire image.
- Performing hysteresis Thresholding.

Non-Maximal Suppression: Once the direction of the gradient is known, the pixel that has no local maximum gradient magnitude is eliminated. If the pixel's gradient direction is one of 8 possible main directions the gradient magnitude of this pixel is compared with two of its immediate neighbors along the gradient direction and the gradient magnitude is set to zero if it does not correspond to a local maximum. ng gradients.

Threshold Calculation: The high threshold is computed such that a percentage p1 of total pixel in the image would be classified as Strong edge. The high threshold corresponds to the point at which value of gradient magnitude is Cumulative distributive function (CDF) equals to 1- p1. The low threshold is calculated as percentage p2 of high threshold.

Hysteresis Threshold: If the gradient magnitude of pixel is greater than high threshold then this pixel is considered as strong edge.
If the gradient magnitude of pixel is between high and low threshold then this pixel is considered as weak edge. Hysteresis used to determine the edge map.

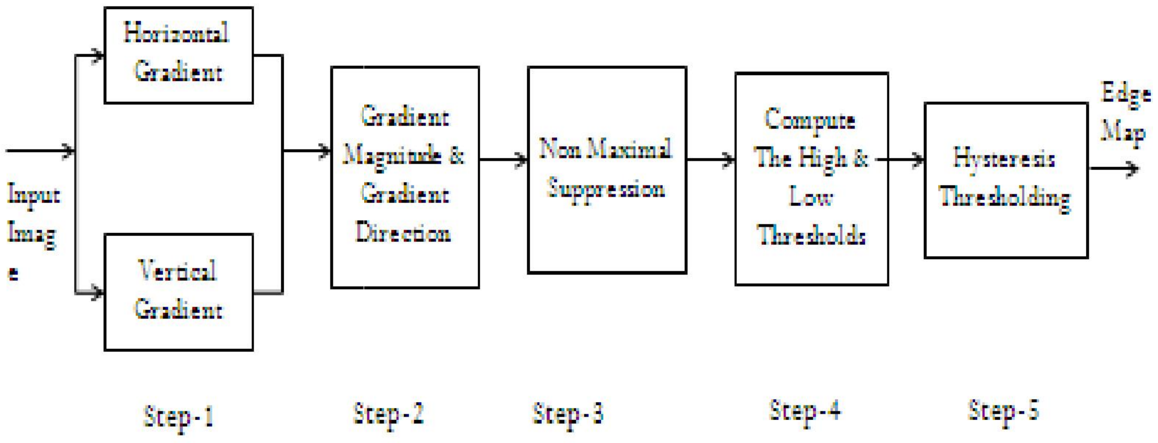


Fig. 1. Block Diagram of the Canny Edge Detection Algorithm

Smoothing: Smoothing of the image is achieved by median filter to remove noise.

Calculation of Gx and Gy: It is performed using the Sobel Operator. The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows).

Gradients and direction calculation: The approximate absolute gradient magnitude (edge strength) at each point can be found by the formula which is simpler to calculate compared to the equation of exact gradient magnitude.
Exact formula is given by,

$$G(magnitude) = \sqrt{Gx^2 + Gy^2}$$

Approximate calculation can be done by,

G(approximate)= 15\16 * max(Gx,Gy) + 0.4* min(Gx,Gy).

The formula for finding the edge direction is given below:

theta = invtan (Gy / Gx)

Distributed Canny Edge Detection

The Canny edge detection algorithm operates on the whole image and has a latency that is proportional to the size of the image. While performing the original canny algorithm at the block-level would speed up the operations, it would result in loss of significant edges in high-detailed regions and excessive edges in texture regions. Natural images consist of a mix of smooth regions, texture regions and high-detailed regions and such a mix of regions may not be available locally in every block of the entire image. In (Christos Gentsos, 2010), distributed canny edge detection algorithm is proposed, which removes the inherent dependency between the various blocks so that the image can be divided into blocks and each block can be processed in parallel.

In the distributed version of the Canny algorithm (Christos Gentsos, 2010; QianXu and Lina J. Karam, 2014), the input image is divided into m × n overlapping blocks, and the blocks are processed independent of each other. To prevent edge artifacts and loss of edges at the boundaries, adjacent blocks overlap by *(L-1)/2* pixels for L× L gradient mask. However, for each block, only edges in the central n × n (where n= m-L+1) non-overlapping region are included in the final edge map.

In the proposed algorithm, Steps 1 to 3 and Steps 5 to 7 are the same as in the original canny algorithm except that these are now applied at the block level.

The high and low gradient threshold selection step of the original Canny (Step 4) is modified to enable block-level processing. Analysis of natural images showed that a pixel with a gradient magnitude of 4 corresponds to a psycho-visually significant edge. Also, a pixel with a gradient magnitude of 2 and 6 corresponds to blurred edges and very sharp edges, respectively. The proposed threshold selection algorithm was designed based on these observations and is as shown below:

- Calculating the horizontal gradient $G^x$and vertical gradient G^yat each pixel location by convolving with gradient masks.
- Computing the gradient magnitude G and direction θ^Gat each pixel location.
- Applying Non-Maximal Suppression (NMS) to thin edges.
- Parallel block-level processing without degrading the edge detection performance.
- Performing hysteresis thresh holding to determine the edge map.

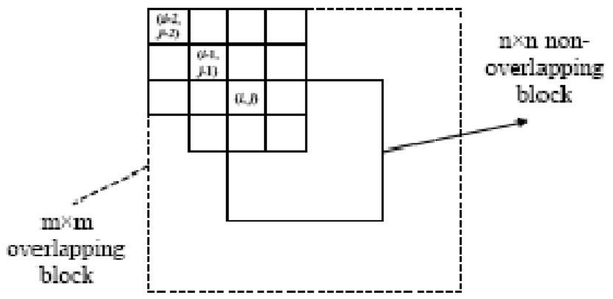


Fig. 2. Structure of m x m overlapping block

Natural images consist of a mix of smooth regions, texture regions and high-detailed regions and such a mix of regions may not be available locally in every block of the entire image. The input image is divided into m×m overlapping blocks. The adjacent blocks overlap by (L − 1)/2 pixels for a L × L gradient mask.

Modified Distributed Canny Edge Detection Algorithm Using Fpga

The system for implementing the Distributed canny edge detection algorithm based on an FPGA platform.

Our FPGA Architecture design follows the exact procedure of the Canny algorithm, therefore our implementation has 5 blocks (Canny Edge Detection, 2009)

- Smoothing
- Sobel Gradient calculation
- Non Maximum Suppression
- Thresholding
- Hysteresis

Smoothing

The digital image is first smoothened using median filter to remove noise. In this implementation 3x3 image matrix is used to achieve parallel filtering. Median filtering with FPGA needs 3 line FIFO ,6 D-type Flip-flops with controller.

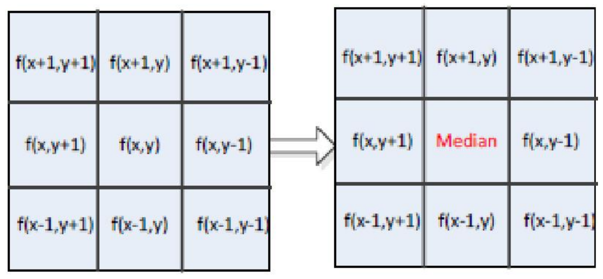


Fig. 4. Median Filtering

Sobel Gradient Calculations

To calculate horizontal and vertical gradient 3x3 image matrix is multiplied with the Sobel operator.

I1	I2	I3
I4	I5	I6
I7	I8	I9

m1	m2	m3
m4	m5	m6
m7	m8	m9

Fig. 5. Image matrix and mask

$$y = i1m1 + i2m2 + i3m3 + \cdots \ldots + i9m9$$

$$y = \sum_{a=1}^{9} iama$$

To implement horizontal and vertical gradient blocks in FPGA (Canny Edge Detection, 2009), we require Adder, Subtractor, Multiplier, Comparator. Input to this blocks are image pixels.

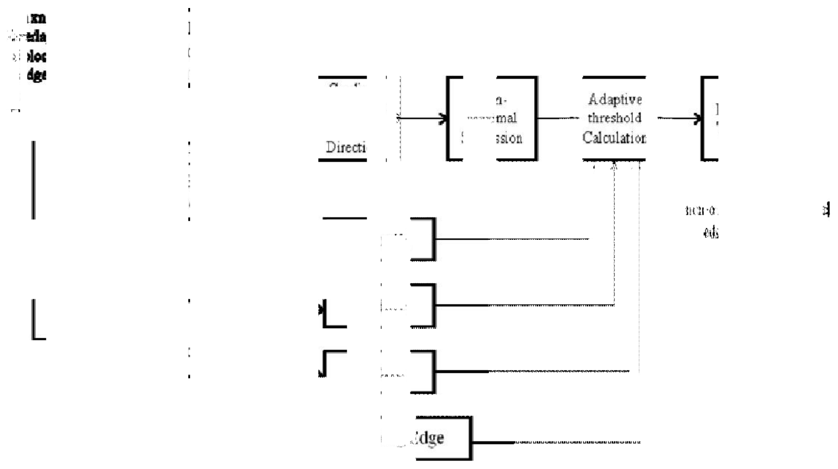


Fig. 3. Distributed Canny Edge Detection Algorithm Block Diagram

Output of this comparator is used to calculate the Gradient magnitude. It calculates approximate root value. FPGA hardware requires multiplier, comparator to calculate the gradient value.
Exact formula is given by,

$$G(magnitude) = \sqrt{Gx^2 + Gy^2}$$

Approximate calculation can be done by,

G(approximate)= 15\16 * max(Gx,Gy) + 0.4* min(Gx,Gy).

Non-Maximal Suppression

The formula for finding the edge direction is given below:
theta = invtan (Gy / Gx)

Horizontal and vertical gradient magnitudes are fetched from memory and used as input to NMS unit.

It computes gradient direction at each pixel. Gradient magnitudes of gradient magnitudes of four nearest neighbors along the direction are selected to compute two intermediate gradients. Gradient magnitudes of four nearest neighbors along the direction are selected to compute two intermediate gradients. The final gradient magnitude after directional NMS (marked as Mag_NMS (x, y) is stored back into local memory and used as the input for the hysteresis thresholding unit.

Block Classification

Block classification unit divides image into different blocks such as smooth, edge/texture, uniform, strong edge and medium edge block. It consists of two parts: 1) pixel classifier, 2) block classifier. Input to block classification unit is image blocks. Pixel classification, the local variance of each pixel is utilized and the variance is calculated as follows:

Var = 1/8 $\sum_{i=1}^{9}(xi - xm)^2$

Where,

Xi =value of the current pixel
Xm =mean of the 3 X 3 template of pixel.

The pixels in the 3×3 windows are fetched from the local memory and stored in one FIFO buffer to compute the local variance. Then, the local variance is compared with T_u and T_e (Threshold Values: Tu =Upper Threshold, Te=Lower Threshold) [17] in order to determine pixel type. Then two counters are used to get the total number of pixels for each pixel type. The output of counter 1, the number of uniform pixels, while the output of counter 2, the number of edge pixels. The block classification stage is initialized once the counter values are available.

Adaptive Thresholding

Adaptive threshold estimator calculates the threshold for each block. High threshold is given as 1-P1 and low threshold is 40% of high threshold. It selects threshold value according to the type of block. Block type is used as select line for multiplexer.P1 is percentage of edge pixels in block.

Thresholding with Hysteresis

Strong edges are interpreted as "certain edges", and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. Hysteresis is basically a procedure of comparing the value of the pixel at hand with the values of three neighboring pixels above it and the one directly on the left. If the pixel is a possible edge and one of the aforementioned neighboring pixels is a definite edge, then the pixel becomes a definite edge. Otherwise it is left as is.

RESULTS

Experimental results include the testing on images on MATLAB to select the proper operator for gradient calculation. Proposed algorithm is tested on FPGA Platform Virtex-4 is used.

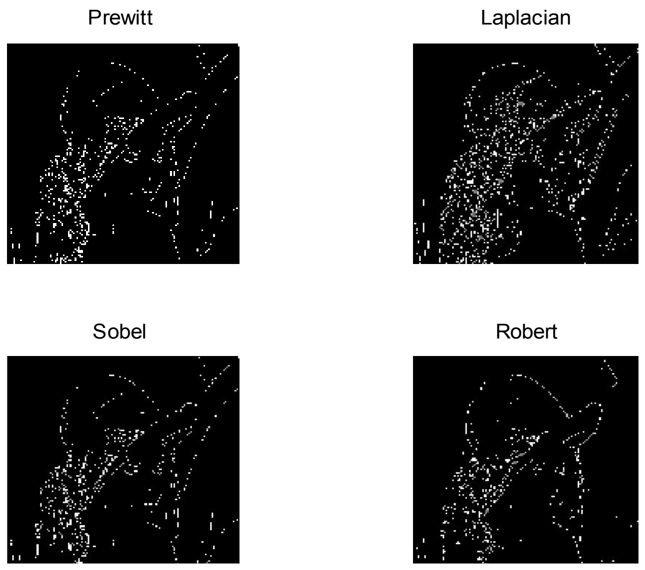


Fig. 6. Lena image output with different operator

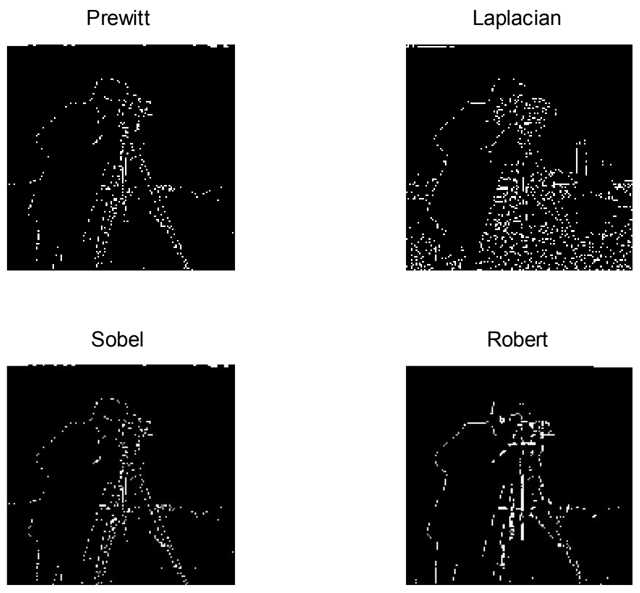


Fig. 7. Cameraman image output with different operator

MATLAB RESULTS

Different operators like Prewitt, Robert, Sobel, and Laplacian are applied on the Lena and Cameraman images to check the performance. Simulation Results shows the performance of Sobel operator is better, hence it is selected for gradient calculation.

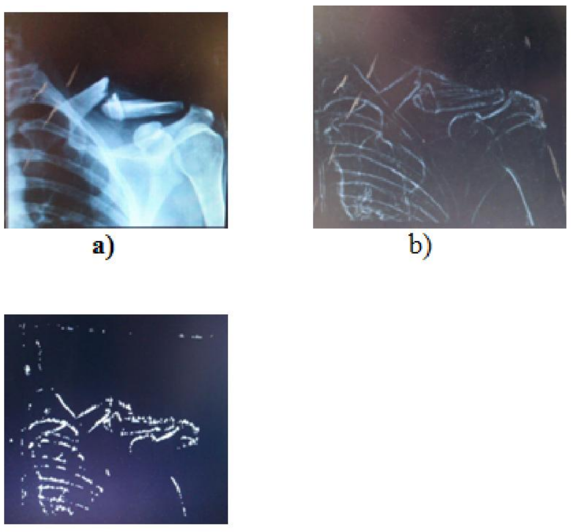
FPGA Implementation Results

Table shows the resource utilization summary for gradient calculation. Present Canny algorithm uses FIR Filter IP for the calculation of gradient, proposed canny algorithm implementation uses the approximate gradient calculation. So, it is observed that resource utilization is reduced by almost 30%, ultimately cost of the design is also reduced.

Table 1. Resource Utilization Summary

Logic Utilization	Using IP Core	Discrete Implementation	Optimization	% Optimization
Number of Slice Flip Flops	317	230	87	27
Number of occupied Slices	268	183	85	32
4 input LUTs	217	147	70	32

Image Results



a) b) c)

Fig.8: a) Original Image b) Canny Edge Output Image c) Proposed Algorithm Output Image

Conclusion

In this paper, modified distributed canny algorithm with the approximate calculation of the gradient is presented. This proposed method is implemented and tested on FPGA platform. Test images are taken from standard database. It is observed that algorithm has better edge detection performance with reduced latency. As system involves approximate calculation it increases speed of operation, also hardware required for the calculation is also optimized. It reduces computation cost as compared to original edge detection algorithm. The extension of this work is the effective use of GA for better optimization of resources yielding the cost efficient distributed canny edge detector.

REFERENCES

Christos Gentsos, Calliope- LouisaSotiropoulou and Spiridon Nikolaidis Nikolaos Vassiliadis ,"Real- Time Canny Edge Detection Parallel Implementation for FPGAs" 978- 1-4244-8 157 -6/ 1 0/$26.00 ©2010 IEEEICECS 20 10 499-502.

QianXu, Lina J. Karam,"A Distributed Canny Edge Detector: Algorithm and FPGA Implementation", IEEE Transactions on Image Processing DOI 10.1109/TIP.2014.2311656.

Wenhao He and KuiYuan, 2008. "An Improved Canny Edge Detector and its Realization on FPGA" Proceedings of the 7th World Congress on Intelligent Control and Automation June 25 - 27, Chongqing, China.

Christos Gentsos, 2010. Calliope- LouisaSotiropoulou and Spiridon Nikolaidis Nikolaos Vassiliadis "Real- Time Canny Edge Detection Parallel Implementation for FPGAs" 978- 1-4244-8 157 -6/ 1 0 ©20 10 IEEEICECS 499-502.

"Canny Edge Detection", 09gr820, March ,2009.

Caixia Deng, Weifeng Ma, Yin Yin ," An Edge Detection Approach of Image Fusion Based on Improved Sobel Operator", 2011 4th International Congress on Image and Signal Processing.

Li song and Kang He, 2013 "Parallel Hough Transform-Based Straight Line Detection and Its FPGA Implementation in Embedded Vision",Sensor,13,9223-9247.

Shashidhar Ram Joshi, Roshan Koju, "Study and Comparison of Edge Detection Algorithm", 978-1-4673-2590-5/12©2012 IEEE.

Anagha Parag Khedkar and Dr. Shaila Subbaraman, April 2009. "Novel Approach of Varying Mutation Probability in Genetic Algorithm", Int. J. Systemics, Cybernetics and Informatics, Hyderabad, India, pp. 65-68.

Anagha Parag Khedkar and Dr. Shaila Subbaraman, March 2010. "Novel operator for Genetic Algorithm", Proc. of 3rd International conference on Data Management: Innovations and Advances in Computer Science and Engineering, Gaziabad IMT, pp. 111-117.

Anagha Parag Khedkar and Dr. Shaila Subbaraman, 2010. "Effect of Advanced Novel Operator on the Performance of Genetic Algorithm", International Journal of Engineering Research & Technology IJERT, vol. 3, pp. 721-731.

****