International Journal of Research in Engineering, Science and Management
Volume-2, Issue-5, May-2019
www.ijresm.com | ISSN (Online): 2581-5792

Edge Detection using Distributed CANNY Algorithm and Implementation in FPGA

R. Jayarani
Lecturer, Dept. of Electronics and Communication Engineering, Government Polytechnic College, Trichy, India

***Abstract*: Edge detection is the most common pre-processing step in digital image processing. Edge detection which significantly reduces the amount of data and filter out the unwanted information in an image. Canny edge detection is the most common edge detection algorithm and it is based on frame level statistics. Direct implementation of canny algorithm at the block level leads to excessive edges in smooth region and loss of significant edges in high detailed region. This cannot be used for real time implementation because it requires higher latency. To overcome this problem, a distributed canny edge detection algorithm is proposed and it is based on block classification. This resulting block based code will significantly reduce the latency. Hence, the proposed algorithm will yield a better performance than the other edge detection algorithm. Finally, this algorithm is mapped onto Xilinx Spartan-3(EDK) Embedded Development Kit FPGA platform**

***Keywords*: Distributed image processing, canny edge detector, high throughput, parallel processing, FPGA.**

1. Introduction

Edge detection is the most common pre-processing step in image processing such as image segmentation, image enhancement, tracking and image/video coding. The Canny edge detector is predominantly used due to its ability to extract significant edges. Edge detection is a basic operation in image processing. There are many edge detection algorithms, such as Robert detector, Prewitt detector, Kirsch detector, Gauss-Laplace detector and canny detector have been proposed. Among these algorithms, canny algorithm has been used widely in the field of image processing because of its good performance. The Canny edge detector is predominantly used in many real-world applications due to its ability to extract significant edges with good detection and good localization performance. The Canny edge detection algorithm contains extensive pre-processing and post processing steps and is more computationally complex than other edge detection algorithms. Furthermore, it performs hysteresis thresholding which requires computing high and low thresholds based on the entire image statistics. Finally, this algorithm is mapped onto Xilinx Spartan 3 (EDK) FPGA platform.

A. CANNY edge detection Algorithm

Canny developed an approach to derive an optimal edge detector based on three criteria for edge detection.

- Low error rate of detection. It should find all edges and nothing but edges.
- Localization of edges. The distance between actual edges in the image and the edges found by the algorithm should be minimized.
- Single response. The algorithm should not return multiple edge pixels when only a single edge exists.

The model was based on a step edge corrupted by additive white Gaussian noise. The original canny algorithm consists of following steps:

a) Smoothing the input image by Gaussian mask. This eliminates the high frequency components in the image. The output smoothed image is denoted as I(x, y).
b) Calculating the horizontal and vertical gradient Gx(x,y) and Gy(x, y) respectively at each pixel location by convolving the image I(x, y).
c) Computing the gradient magnitude G(x, y) and direction Θg(x, y) at each pixel location.
d) Applying non-maximum suppression (NMS) to thin the edges.
e) Computing the hysteresis high and low thresholds based on the histogram of the magnitudes of the gradients of the entire image [3][4].

2. Proposed distributed CANNY edge detection

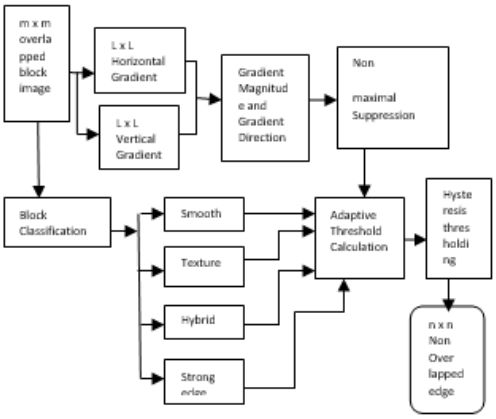


Fig .1. Block Diagram

Edge detection is the most common pre-processing step in many image processing algorithms such as image enhancement, image segmentation, tracking and image/video coding. Among

the existing edge detection algorithms, the canny edge detector has remained a standard for many years and has best performance. Canny edge detection algorithm cannot be used for real time application since it requires higher latency and decreased throughput. Due to the drawback, distributed canny edge detection algorithm is proposed which is based on block classification. Block classification is based on block type such as smooth, texture, edge/texture and strong edge. The proposed algorithm yields better edge detection results for both clean and noisy images.

Distributed Canny edge detection algorithm is proposed. A diagram of the proposed algorithm is shown in Fig.1. In the proposed distributed version of the canny algorithm, the input image is divided into m × m overlapping blocks and the blocks are processed independent of each other. For an L × L gradient mask, the m × m overlapping blocks are obtained by first dividing the input image into n × n non-overlapping blocks and then extending each block by (L + 1)/2 pixels along the left, right, top, and bottom boundaries. This results in m × m overlapping blocks, with m = n + L + 1. Next horizontal and vertical gradient are calculated. NMS step involves computing the gradient direction at each pixel. If the pixel's gradient direction is one of 4 possible main directions (0°, 45°, 90°, 135°) the gradient magnitude of this pixel is compared with two of its immediate neighbours along the gradient direction and the gradient magnitude is set to zero if it does not correspond to a local maximum. For the gradient directions that do not coincide with one of the 8 possible main directions, an interpolation is done to compute the neighbouring gradients. Distributed Canny algorithms are the same as in the original canny algorithm except that these are now applied at the block level. Next step which is the hysteresis high and low thresholds calculation is modified to enable parallel block-level processing without degrading the edge detection performance.

A. The distributed canny edge detection algorithm consists of following steps

- Smoothing the input image by Gaussian mask. This eliminates the high frequency components in the image. The output smoothed image is denoted as G(x, y).
- Calculating the horizontal and vertical gradient Gx and Gy respectively at each pixel Location by convolving the image G(x, y).
- The gradient magnitude and direction at each pixel location.
- Applying non-maximum suppression (NMS) to thin the edges.
- Block classification.
- Adaptive threshold calculation.
- Computing the hysteresis high and low thresholds based on the histogram of the magnitudes of the gradients of the entire image.

1) Input Image:

2D Input image of size 512x512 in Fig. 2 is consider as input image and can be represented as G(x,y).

Fig. 2. Lena input image

2) Gaussian filtering to remove noise

The Gaussian filter is used to blur and remove unwanted detail and noise. By calculating a suitable 3x3 mask, the Gaussian smoothing can be performed using standard convolution method. A convolution mask is much smaller than the actual image. As a result, the mask is slide over the image, calculating every square of pixels at a time. Gaussian filter uses 2D distribution to perform convolution.

$$F_x(x,y) = -\frac{x}{\sigma^2} e^{-x^2+y^2/2\sigma^2} \quad (1)$$

$$F_y(x,y) = -\frac{y}{\sigma^2} e^{-x^2+y^2/2\sigma^2} \quad (2)$$

Where x and y represents horizontal and vertical values of a two dimensional image, sigma =0.1 is the standard deviation of the Gaussian function and G(x, y) represents the smoothened image. Sigma plays a major role in the selection of the coefficients of the Gaussian filter.

3) Gradient

Gradient is a directional change in the intensity or colour in an image. Image gradients may be used to extract information from images. Vertical edges can be detected by using a horizontal gradient operator followed by a threshold operation to detect the extreme values of the gradient. Horizontal edges produce a vertical gradient in the image, and can be enhanced with a vertical gradient detector. After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image [7]. The gradient can be calculated using two different masks such as horizontal and vertical gradient. Column of sobel operator is horizontal gradient (Di) and row is vertical gradient (Dj).Sobel operator is used to determine the gradient at each pixel of smoothened image. Sobel operators in i and j directions are given as,

$$D_i = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad D_j = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (3)$$

Sobel masks are convolved with smoothed image and giving gradients in i and j directions.

$$Gx = Di * G(x,y) \quad (4)$$

Gy=Dj*G(x,y) (5)

Where Gx and Gy are horizontal and vertical gradient. G(x,y) represents the image.

4) Gradient magnitude and direction:

Gradient magnitude: Gradient magnitude provides the information about the strength of the edge. The horizontal and vertical gradient values obtained in the gradient unit are the inputs to the magnitude block, these gradient values are used for finding the magnitude of the input image. Magnitude of the gradient is calculated using the formula,

$$Magnitude = \sqrt{G_x^2 + G_y^2} \quad (6)$$

Gradient Direction: The direction of gradient is always perpendicular to the direction of the edge. Finding the edge direction is trivial once the gradient in the x and y directions are known.

$$Direction = \tan^{-1}(G_y/G_x) \quad (7)$$

Gy represents the vertical direction. Gx represents the horizontal direction

5) NMS (Non Maximum Supperssion)

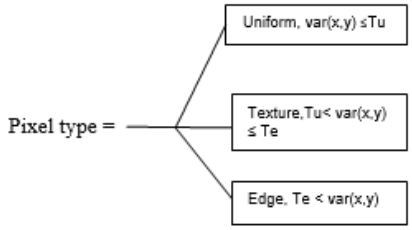
Non-maximum suppression is an edge thinning technique. Given estimates of the image gradients, is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. In some implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the centre pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbours in the gradient direction.

6) Block Classification

The block classification method based on local variance of pixel using 3x3 window that is centered around the considered pixel in order to label the size it as of type edge, texture. Each block is classified based on the total number of edges, texture and uniform pixel in the block. Here the input image is divided into block and each block can be classified according to classification techniques [1]. Each block size of an input image is 64x64.

Pseudo codes for block classification:

STEP1: Pixel Classification:



STEP 2: Block classification:

Table 1
Block Classification

Block type	No. of pixels of pixel type	
	N uniform	N edge
Smooth	≥ 0.3 Total pixel	0
Texture	< 0.3 Total pixel	0
Edge/texture	< 0.65(Total_ pixel- N edge)	(>0)&(<0.3 Total_ pixel)
Strong edge	≤0.7 Total_ Pixel	≥0.3 Total_ Pixel

var (x, y): the local(3x3) variance at pixel (x, y)
Tu and Te: two thresholds
Total pixel: the total numbers of pixels in the block
N uniform: the total number of uniform pixels in the block
N edge: the total number of edge pixels in the Block

7) Hysteresis Thresholding

Large intensity gradients are more likely to correspond to edges than small intensity gradient. Hysteresis thresholding is used to determine the edge map. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. Any pixel in the image that has a value greater than t1(0.25) is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to this edge pixel and that have a value greater than t2 (0.8).

3. Simulation output


Fig. 3. Input image


Fig. 4. Canny edge detection


Fig. 5. Proposed distributed canny edge detection

A. Performance evaluation

This validates the observation that sensitivity for noisy image and clean image, the edge detection performance of the proposed distributed canny algorithm is better than the original frame-based canny algorithm.

$$Sensitivity = \frac{TN}{TN+FP} \quad (8)$$

TN=True Negative, FP=False positive

Table 2
Performance Evaluation

Image type	Canny Algorithm	Distributed Canny Algorithm
Clean image	90.5890%	99.0706%
Noise image	92.2551%	98.9970%

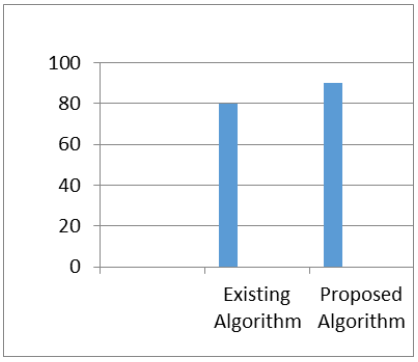


Fig. 6. Comparison Graph

4. Implementation in FPGA

In order to demonstrate the parallel efficiency of the proposed distributed canny edge detection algorithm, an FPGA based hardware implementation of the proposed algorithm is used in this section [5]. Architecture: Depending on the available FPGA resources, the image needs to be partitioned into q sub images and each sub-image is further divided into p m×m blocks. The proposed architecture, shown in Figure-6 consists of q processing units in the FPGA and some Static RAMs (SRAM) organized into q memory banks to store the image data, where q equals to the image site divided by SRAM size.

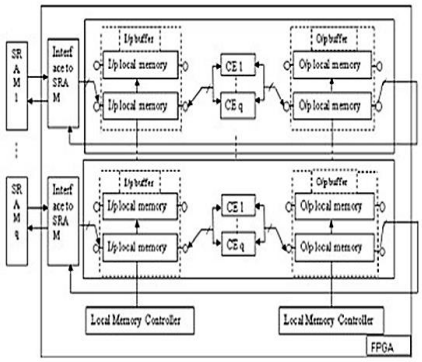


Fig. 7. Architecture of proposed distributed canny edge detector

Each processing unit processes a sub-image and reads/writes data from to the SRAM through ping-pong buffers, which are implemented with dual port Block RAMs (BRAM) on the FPGA. Each processing unit (PU) consists of p computing engines (CE), where each CE detects the edge map of an m×m block image. Thus, p×q blocks can be processed at the same time and the specific values of p and q depend on the processing time of each PE, the data loading time from the SRAM to the local memory and the interface between FPGA and SRAM, such as total pins on the FPGA, the data bus width, the address bus width and the maximum system clock of the SRAM. Each CE consists of the following 6 units, 1. Smoothening unit using Gaussian filter. 2. Vertical and horizontal gradient calculation unit. 3. Magnitude calculation unit. 4. Directional non-maximum suppression unit. 5. Block classification. 6. Thresholding with hysteresis unit.

5. Simulation output

Fig. 8. Input image

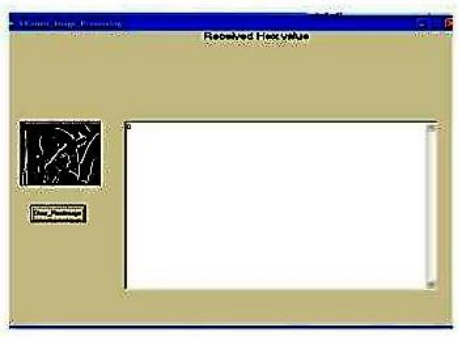


Fig. 9. Edge detected image

6. Conclusion

A novel distributed Canny edge detection algorithm that results in a significant speed up without sacrificing the edge detection performance. As a result, the computational cost of the proposed algorithm is very low compared to the original canny edge detection algorithm. This algorithm is mapped to onto a Xilinx Spartan-3(EDK) FPGA platform.

References

[1] Arbelaez. P, C. Fowlkes, and D. Martin. (2013). The Berkeley Segmentation Dataset and Benchmark http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

[2] Bao. P, L. Zhang, and X. Wu, "Canny edge detection enhancement by scale multiplication," IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 9, pp. 1484– 1490, Sep. 2004.

[3] Canny. J. F, "A computation approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 8, no. 6, pp. 769–798, Nov. 1986.
[4] Gentsos. C, C. Sotiropoulou, S. Nikolaidis, and N. Vassiliadis, "Realtime canny edge detection parallel implementation for FPGAs," in Proc. IEEE ICECS, Dec. 2010, pp. 499–402.
[5] He. W and K. Yuan, "An improved canny edge detector and its realization on FPGA," in Proc. IEEE 7th WCICA, Jun. 2008, pp. 6461–6464.
[6] Lourenco. L. H. A, "Efficient implementation of canny edge detection filter for ITK using CUDA," in Proc. 13th Symp. Comput. Syst., 2012, pp. 33–40.
[7] Luo. Y and R. Duraiswami, "Canny edge detection on NVIDIA CUDA," in Proc. IEEE CVPRW, Jun. 2008, pp. 1–8.
[8] Narvekar. N. D and L. J. Karam, "A no-reference image blur metric based on the cumulative probability of blur detection (CPBD)," IEEE Trans. Image Process., vol. 20, no. 9, pp. 2678–2683, Sep. 2011.