International Journal of Computer Engineering Science (IJCES)
Volume 3 Issue 5 (May 2013) ISSN : 2250:3439
https://sites.google.com/site/ijcesjournal http://www.ijces.com/

Implementation of Distributed Canny Edge Detector on FPGA

T.Rupalatha[1],G.Rajesh[2], K.Nandakumar[3]
[1]II M.Tech, Chadalawada Ramanamma Engg. College, Tirupati
[2]Assoc.Professor, Dept.of.ECE, CREC, Tirupati
[3]Assistant Professor, Dept. of. ECE, CREC, Tirupati
rupalathareddy@gmail.com[1] ,rajesh.gundlapalli@gmail.com[2] .menanda.k@gmail.com[3]

Abstract: Edge detection is one of the basic operation carried out in image processing and object identification .In this paper, we present a distributed Canny edge detection algorithm that results in significantly reduced memory requirements, decreased latency and increased throughput with no loss in edge detection performance as compared to the original Canny algorithm. The new algorithm uses a low-complexity 8-bin non-uniform gradient magnitude histogram to compute block-based hysteresis thresholds that are used by the Canny edge detector. Furthermore, an FPGA-based hardware architecture of our proposed algorithm is presented in this paper and the architecture is synthesized on the Xilinx *Spartan-3E* FPGA. Simulation results are presented to illustrate the performance of the proposed distributed Canny edge detector. The FPGA simulation results show that we can process a 512×512 image in **0.28ms** at a clock rate of **100 MHz.**

Keywords: Canny Edge detector, Distributed Processing, Non-uniform quantization, FPGA.

1. Introduction

Edge detection serves as a pre-processing step for many image processing algorithms such as image enhancement, image segmentation, tracking and image/video coding. Typically, edge detection algorithms are implemented using software. With advances in Very Large Scale Integration (VLSI) technology.

Some approaches have been proposed for real-time edge detection. Alzahrani and Chen present an absolute different mask (ADM) edge detection algorithm and its pipelined VLSI architecture but the edge detector in [1] offers a trade-off between precision, cost and speed, and its capability to detect edges is not as good as the Canny algorithm(complex than other edge detection algorithms, such as Roberts, Prewitt and Sobel algorithms). There is another set of work on Deriche filters that have been derived using Canny's criteria. For instance, it was stated in [2] that a network with four transputers takes 6s to detect edges in a 256×256 image using the Canny- Deriche algorithm, far from the requirement for real-time applications .The approach of [3] operates on two rows of pixels at a time. This reduces the memory requirement at the expense of a decrease in the throughput. Furthermore, it is known that the original Canny edge detection algorithm needs two adaptive image-dependent high and low thresholds to remove false edges. However, the algorithm in [3] just fixes high and low thresholds in order to overcome the dependency between the blocks, which results in a decreased edge detection performance.

The hysteresis threshold calculation is a key element that greatly affects the edge detection results. However, the original Canny algorithm computes the high and low thresholds for edge detection based on the entire image statistics,which prevents the processing of individual blocks independently.

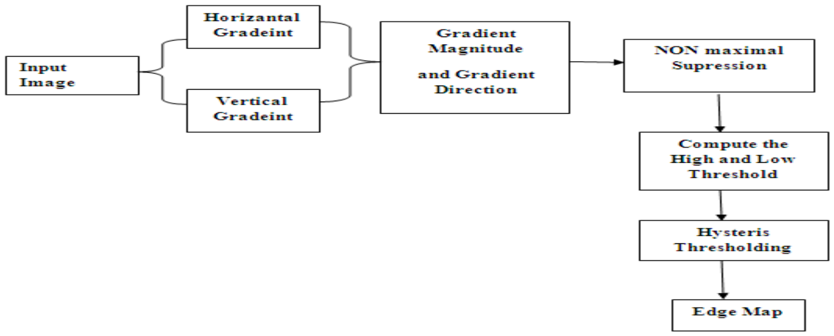


Fig.1. Block diagram of the Canny edge detection algorithm.

In [4], we proposed a new threshold selection algorithm based on the distribution of pixel gradients in a block of pixels to overcome the dependency between the blocks. However, in [4], the hysteresis thresholds calculation is based on a very finely and uniformly quantized 64-bin gradient magnitude histogram, which is computationally expensive and, thereby, hinders the real-time implementation. In this paper, a method based on non-uniform and coarse quantization of the gradient magnitude histogram is proposed. In addition, the proposed algorithm is mapped onto a reconfigurable hardware architecture.

2. CANNY EDGE DETECTION ALGORITHM

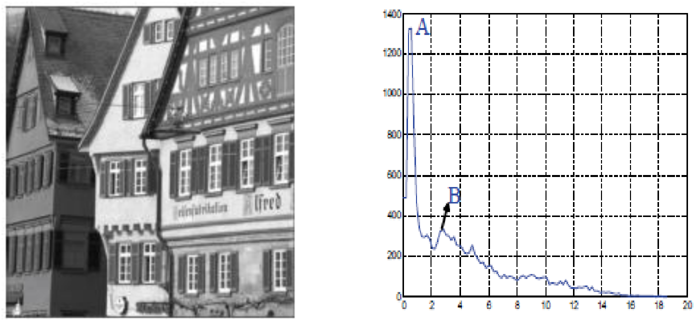
Canny developed an approach to derive an optimal edge detector based on three criteria related to the detection performance. The model was based on a step edge corrupted by additive white Gaussian noise. A block diagram of the Canny edge detection algorithm is shown in Fig. 1. The original Canny algorithm [5] consists of the following steps:

1. Smoothing the input image by Gaussian mask. The output smoothed image is denoted as I(x, y).
2. Calculating the horizontal gradient $G_x(x, y)$ and vertical gradient $G_y(x, y)$ at each pixel location by convolving the image I(x, y) with partial derivatives of a 2D Gaussian function.
3. Computing the gradient magnitude G(x, y) and direction $\theta_G(x, y)$ at each pixel location.
4. Applying non-maximum suppression (NMS) to thin edges.
5. Computing the hysteresis high and low thresholds based on the histogram of the magnitudes of the gradients of the entire image.
6. Performing hysteresis thresholding to determine the edge map.

3. PROPOSED DISTRIBUTED CANNY EDGE DETECTION ALGORITHM

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 [4], we proposed a distributed Canny edge detection algorithm, 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.

The input image is divided into m x m overlapping blocks. The adjacent blocks overlap by $(L - 1)/2$ pixels for a $L\times L$ gradient mask. However, for each block, only edges in the central $n\times n$ (where $n = m + L - 1$) non-overlapping region are included in the final edge map. Steps 1 to 4 and Step 6 of the distributed Canny algorithm are the same as in the original Canny algorithm except that these are now applied at the block level. Step 5, which is the hysteresis high and low thresholds calculation, is modified to enable parallel processing. In [4], a parallel hysteresis thresholding algorithm was proposed based on the observation that a pixel with a gradient magnitude of 2, 4 and 6 corresponds to blurred edges, psychovisually significant edges and very sharp edges, respectively. In order to compute the high and low hysteresis thresholds, very finely and uniformly quantized 64-bin gradient magnitude histograms are computed over overlapped blocks. If the 64-bin uniform discrete histogram is used for the high threshold calculation, this entails performing 64 multiplications and 64×Np comparisons.



(a) (b)

Fig.2.(a) Original 512×512 House image; (b) Histogram of the gradient magnitude after non-maximal suppression of the House image.

As in [6], it was observed that the largest peak in the gradient magnitude histograms after NMS of the Gaussian smoothed natural images occurs near the origin and corresponds to low-frequency content, while edge pixels form a series of smaller peaks where each peak corresponds to a class of edges having similar gradient magnitudes. Consequently, the high threshold should be selected between the largest peak and the second largest edge peak.

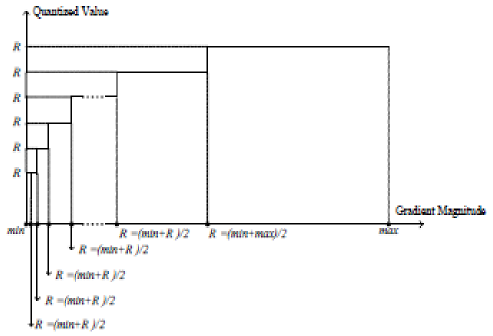


Fig 3. Reconstruction values and quantization levels; min and max representation.

A sample gradient magnitude histogram is shown in Fig. 2(b) for the 512×512 House image (Fig. 2(a)). Based on the above observation, we propose a non-uniform quantizer to discretize the gradient magnitude histogram. Specifically, the quantizer needs to have more quantization levels in the region between the largest peak A and the second largest peak B and few quantization levels in other parts. Fig. 3 shows the schematic diagram of the designed quantizer. Accordingly, n reconstruction levels can be computed as follows:

$$R_1 = (\min + \max)/2, \quad (1)$$

$$R_{i+1} = (\min + R_i)/2,\ i = 2, ..., n \quad (2)$$

where min and max represent, respectively, the minimum and maximum values of the gradient magnitude after NMS, and Ri is the reconstruction level. The proposed distributed thresholds selection algorithm is shown in Fig. 4. Let G_t be the set of pixels with gradient magnitudes greater than a threshold t, and let N_{Gt} for t = 2, 4, 6, be the number of corresponding gradient elements in the set G_t. Using N_{Gt}, an intermediate classification threshold C is calculated to indicate whether the considered block is high-detailed, moderately edged, blurred or textured, as shown in Fig. 4. Consequently, the set G_t = G_t=c can be selected for computing the high and low thresholds. The high threshold is calculated based on the histogram of the set Gc such that 20% of the total pixels of the block would be identified as strong edges. The lower threshold is the 40% percentage of the higher threshold as in the original Canny algorithm.

We compared the high threshold value that is calculated using the proposed distributed algorithm based on an 8-bin non-uniform gradient magnitude histogram with the value obtained when using a 16-bin non-uniform gradient magnitude histogram. These two high thresholds have similar values. Therefore, we use the 8-bin non-uniform gradient magnitude histogram in our implementation.

4. PROPOSED DISTRIBUTED CANNY ALGORITHM IMPLEMENTATION ON FPGA

In this section, we describe the hardware implementation of our proposed distributed Canny edge detection algorithm on the Xilinx *Spartan-3E* FPGA.

4.1. Architecture Overview

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 x m blocks. The proposed architecture, shown in Fig.4, 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 size divided by the SRAM size. 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. As shown in Fig.4, 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 processing time for an N×N image is reduced, in the best case, by a factor of p x q.

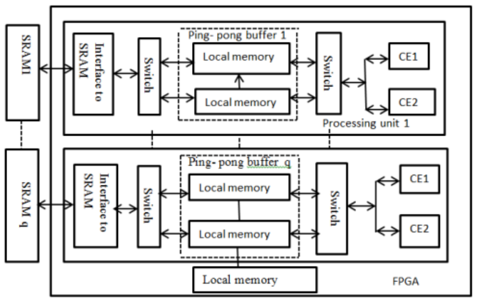


Fig.4. The architecture of the proposed distributed Canny algorithm.

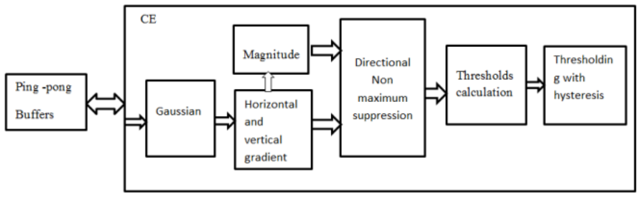

CE
Ping -pong Buffers
Gaussian
Magnitude
Horizontal and vertical gradient
Directional Non maximum suppression
Thresholds calculation
Thresholding with hysteresis

Fig.5. Block diagram of the CE (compute engine) for the proposed distributed Canny edge detection.

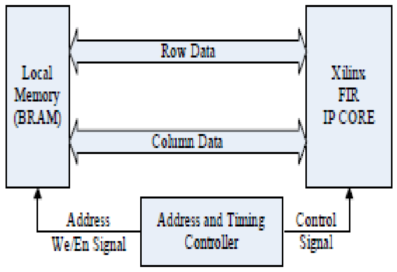
The specific values of p and q depend on the processing time of each PE, the data loading time from the SRAM to the local memory and the interface between FPGA and SRAM, such as total pins on the FPGA, the data bus width, the address bus width and the maximum system clock of the SRAM. In our application, we choose p = 2 and q = 8. In the proposed architecture, each CE consists of the following units, as shown in Fig.5:

4.2. Image Smoothening

The input image is smoothened using a 3×3 Gaussian mask, as shown in Fig. 6(a). The Gaussian filter (Fig. 6(a))is separable and, thus, the implementation of the 2-D convolution with the 3×3 Gaussian mask is achieved using row and column 1- D convolutions. The proposed architecture for the smoothening unit is shown in Fig. 6(b).

p^2	pq	p^2
pq	q^2	pq
p^2	pq	p^2

(a)



(b)

Fig6.(a) Mask for the low pass Gaussian filter with p = 0.0437; q = 0.9947; (b) Pipelined Image Smoothening Unit.

The main components of the architecture consists of a 1-D finite impulse filter (FIR) to process the data and the on-chip Block RAM (BRAM) to store the data. In our design, we adopt the Xilinx's pipelined FIR IP core, which provides a highly parameterizable , area-efficient, high-performance FIR filter utilizing the structure characteristics in the coefficient set, such as symmetry and conjugacy .By exploiting the symmetry of the Gaussian filter, the architecture uses two multipliers to perform the 1-D convolution using a 3-tap filter. The address controller fetches the input image data from the local memory into the FIR core and, after the computation, it stores the results back in the BRAM.

4.3. Gradients and Gradient Magnitude Calculation

This stage calculates the vertical and horizontal gradients using convolution kernels. The kernels vary in size from 3×3 to 9×9, depending on the sharpness of the image. The Xilinx FIR IP core, which can support up to 256 sets of coefficients with 2 to 1024 coefficients per set, is used to implement the kernels. The whole design is pipelined, and thus the output is generated every clock cycle. This is input to the magnitude calculation unit which computes, at each pixel location, the gradient magnitude from the pixel's horizontal and vertical gradients.

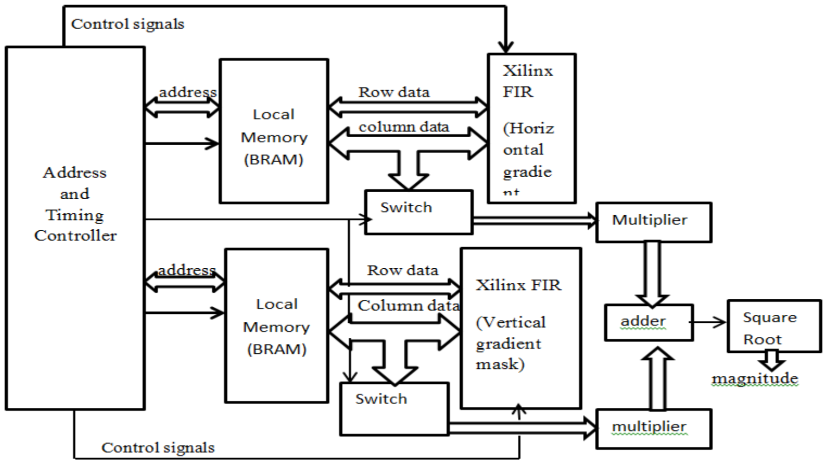


Fig7.Gradient and Magnitude Calculation Unit.

The architecture of this unit is shown in Fig.7. The gradient calculation architecture consists of two 1-D FIR models and the corresponding local memory. The filters for computing the horizontal and vertical gradient elements can process data in parallel. The magnitude computation consists of two multipliers and one square- root computation module, which are implemented by the Xilinx Math Function IP cores.

4.4. Directional Non Maximum Suppression

Fig. 8 shows the architecture of the directional non-maximum suppression unit. In order to access all the pixels' gradient magnitudes in the 3×3 window at the same time, two FIFO buffers are employed.

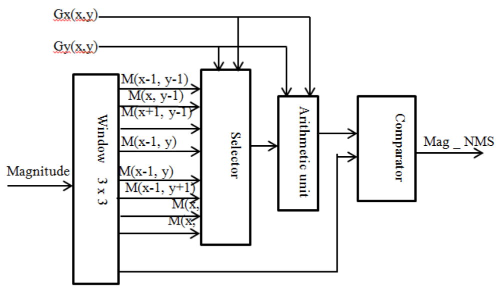


Fig8. Directional Non Maximum Suppression Unit.

The horizontal gradient G_x and the vertical gradient G_y control the selector which delivers the gradient magnitude (marked as M(x, y) in Fig.8) of neighbours along the direction of the gradient, into the arithmetic unit.

4.5. Calculation of the hysteresis thresholds

Since the low and high thresholds are calculated based on the gradient histogram, we need to compute the histogram of the image after it has undergone directional non-maximum suppression .As discussed in Section 3, an 8-step non-uniform quantizer is employed to obtain the discrete histogram for each processed block.

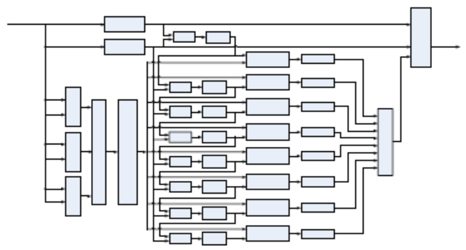


Fig 9. The architecture of the Threshold Calculation Unit.

4.6. Thresholding with hysteresis

Since the output of the non-maximum suppression unit contains some spurious edges, the method of thresholding with hysteresis is used. Two thresholds, high threshold Th_H and low threshold Th_L , which are obtained from the threshold calculation unit, are employed. Let f(x, y) be the image obtained from the non maximum suppression stage, $f_1(x, y)$ be the strong edge image and $f_2(x, y)$ be the weak edge image.

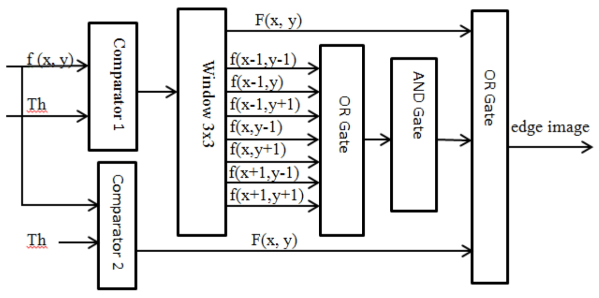


Fig 10.Pipelined architecture of the Thresholding unit

5. SIMULATION RESULTS

The algorithm performance was tested using a variety of512×512 natural images.

5.1. Mat lab Simulation Results

We conducted the following experiments to investigate the effectiveness of the new distributed Canny edge detector that is proposed in this paper.

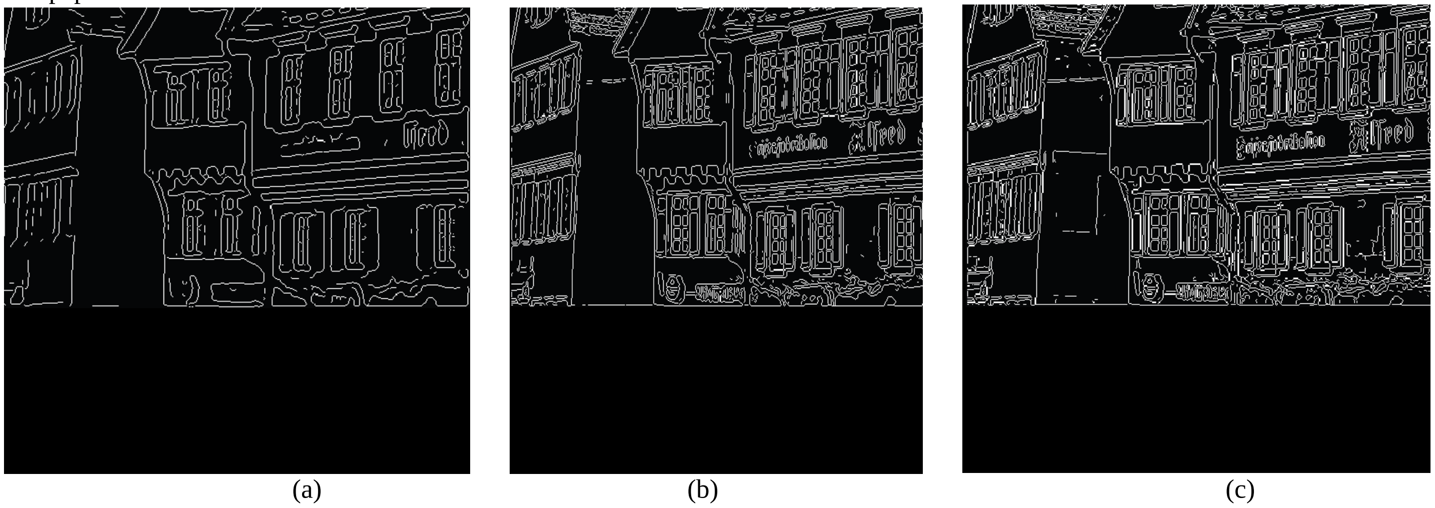


Fig.11. Floating-point Mat lab simulation results for the 512×512 House image: (a) Edge map of the original Canny edge detector; (b) Edge map of the algorithm of [4] with a 3×3 gradient mask and a block size of 64; (c) Edge map of our proposed algorithm with a 3×3 gradient mask and a block-size of 64.

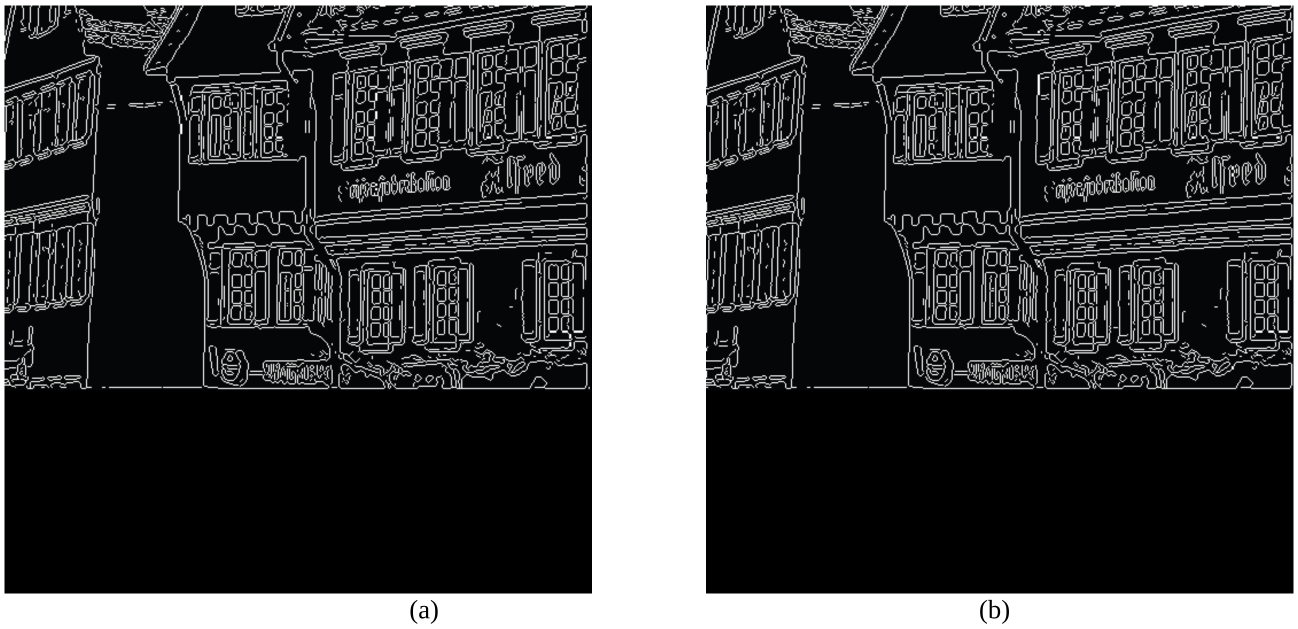


Fig.12. (a)Edge map of Mat lab implementation; (b) Edge map of FPGA implementation.

5.2. Fixed-point Mat lab and FPGA Simulation Results

Fig.12 shows the fixed-point Mat lab implementation software result and the FPGA implementation generated result for the 512×512 House image using the proposed distributed Canny edge detector with block size of 64×64 and a 3×3gradient mask. The FPGA result is obtained using Model Sim .It is clear that the *hardware implementation of our proposed algorithm can successfully detect significant edge*s and results in edge maps that are similar to the ones that are obtained using the fixed-point Mat lab simulation. Furthermore, for a **100MHz** clock rate, the total processing running time using the FPGA implementation is **0.28ms** for a 512×512 image.

6. CONCLUSION

We presented a novel distributed Canny edge detection algorithm that results in a significant speed up without sacrificing the edge detection performance. We proposed a **novel non uniform quantized histogram calculation method in order to reduce the computational cost of the hysteresis threshold selection** .As a result, the computational cost of the proposed algorithm is very low compared to the original Canny edge detection algorithm. The algorithm is mapped to onto a Xilinx *Spartan-3E* FPGA platform and tested using Model Sim. It is capable of supporting fast real-time edge detection for images and videos with various spatial and temporal resolutions including full-HD content.

REFERENCES

[1] F. M. Alzahrani and T. Chen, "A real-time edge detector algorithm and VLSI architecture," *Real-Time Imaging* , vol. 3, no. 5, pp. 363 – 78, 1997.
[2] L. Torres, M. Robert, E. Bourennane, and M. Paindavoine,"I implementation of a recursive real time edge detector using retiming techniques," *VLSI*, pp. 811 –816,Aug. 1995.
[3] D. V. Rao and M. Venkatesan, "An efficient reconfigurable architecture and implementation of edge detection algorithm using Handle-C," *ITCC*, vol. 2, pp. 843 – 847,Apr. 2004.
[4] S. Varadarajan, C. Chakrabarti, L. J. Karma, and J. M.Bauza , "A distributed psycho-visually motivated Canny edge detector," *IEEE ICASSP*, pp. 822 –825, Mar. 2010.
[5] J. Canny, "A computational approach to edge detection,"*IEEE Trans. PAMI*, vol. 8, no. 6, pp. 679 –698, Nov. 1986.
[6] W. He and K. Yuan, "An improved Canny edge detector and its realization on FPGA," *WCICA*, pp. 6561 –6564,Jun. 2008.