2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems

An Efficient Hardware Implementation of Canny Edge Detection Algorithm

Sangeetha D
Full-Time Research Scholar, Department of ECE
Government College of Technology
Coimbatore, India
e-mail: sangeetha.srit@gmail.com

Deepa P
Assistant Professor, Department of ECE
Government College of Technology
Coimbatore, India
e-mail: deepap05@gmail.com

***Abstract*— The edge detection is one of the key techniques in most image processing applications. The canny edge detection is proven to be able to significantly outperform existing edge detection techniques due to its superior performance. Unfortunately, the implementation of the systems in real-time is computationally complex, high hardware cost with increased latency. The proposed canny edge detection algorithm uses approximation methods to replace the complex operations; the pipelining is employed to reduce the latency. Finally, this algorithm is implemented on Xilinx Virtex-5 FPGA. When compared with the previous hardware architecture for canny edge detection, the proposed architecture requires fewer hardware costs and takes 1ms to detect the edges of 512x512 image.**

Keywords*— Pipelining, Canny Edge Detector, Less Hardware, FPGA.*

I. INTRODUCTION

In computer vision, machine vision and many imaging applications, edge detection plays a vital role as a preprocessing step. Edge defines the object boundaries within the image and occurs when discontinuities present in the pixel values. The implementation of an edge detection system in hardware is a challenging task because of the presence of interference in an image due to noise, various lighting conditions, high processing speed, and high accuracy. These challenges make edge detection becomes unreliable and more difficult.

Many researchers have proposed diverse techniques for edge detection [1]-[5]. There are various edge detection techniques, such as Robert detector [1], Prewitt detector [2], Sobel detector [3], Laplace of Gaussian detector [4], and canny edge detector [5]. Among several edge detection techniques canny based edge detection provides better performance by addressing the limitations of all other edge detectors, and thereby achieving greater improvement in terms of sharp edges and noise immunity [6]. However, the canny edge detection algorithm is computationally complicated and high latency for real-time applications.

For real-time applications, high-speed canny edge detection is necessary. So far, several papers have dealt with FPGA based canny edge detection algorithms for real-time applications [7-9]. Canny edge detection algorithm implementation on FPGA is proposed in [7] and [8] uses system-level hardware and software co-design tool results in performance degradation due to high latency. To reduce the latency parallel implementation is modeled in [9] and the pixels are processed in parallel leads to increase in memory access and reduced latency than [7] and [8]. In all these existing hardware implementation, the hysteresis threshold, a low and high threshold are constant in order to reduce the complexity and degrades the performance. To overcome this limitation, the thresholds are computed adaptively in [10] on Altera Cyclone and provides increased latency. Some recent works have tried to reduce the computational complexity and latency by employing absolute operations instead of square and square root operations to calculate the gradient magnitude to show a better reduction in computational complexity and latency as compared to [11] results in lower accuracy. The performance of the edge detection depends upon the threshold values. The computation of proper threshold with low latency is an important issue in the canny edge detection algorithm. Recently, the robust threshold computation method with high accuracy has been introduced in [12], and utilizes more resources with increased computational complexity.

The canny edge detection circuit are applied to an end-user camera equipment should have lower hardware cost. In previous implementation the lower hardware cost is achieved by employing rough calculations to replace the complicated operations and reduction in input data to meet the real-time applications. However, it results in lower accuracy. Thus, there is a trade-off between hardware cost and accuracy. To overcome the issue, an efficient canny edge detection algorithm was developed with adequate accuracy and reduced computational complexity. To increase the accuracy of the proposed canny edge detection algorithm with reduced hardware cost, the edge detection system adapts approximate methods instead of the complex operations. The goal of the approximate method is to use low hardware components without affecting the performance. To reduce the latency of the proposed algorithm pipelining technique was employed. The developed system uses less hardware and leads to a reduction in logic cells with high throughput and reduction in computation time. The key contributions of proposed design strategies are (a) approximate method for the computation of

978-1-4673-8700-2/16 $31.00 © 2016 IEEE
DOI 10.1109/VLSID.2016.68

gradient magnitude and orientation; (b) histogram based median filter architecture uses sliding window technique.

The proposed algorithm is implemented in a security system. To emphasis security, automated personal identification system based on biometrics has been used. Recently, Iris recognition has been used with high reliability for personal identification. In Iris recognition, the canny edge detection algorithm is used for iris segmentation [18]. The architecture has been synthesized on the Xilinx Virtex-5 FPGA. It occupies less hardware and takes 1ms to detect edges of 512×512 images in the Iris database when clocked at 100 MHz.

II. Canny Edge Detection Algorithm

This section gives an overview of the conventional canny edge detection for an optimal edge detector [5]. Fig.1 summarizes the computation of canny edge detection. It can be divided into the following steps.

A. Gradient computation in x and y-direction

The gradients in horizontal and vertical directions, $f_x(x,y)$ and $f_y(x,y)$ respectively, for each pixel(x,y) is calculated by convolving the images with gradient masks.

B. Gradient Magnitude and Orientation computation

After obtaining $f_x(x,y)$ and $f_y(x,y)$, each pixel calculates the magnitude and direction. The magnitude defines the strength at the edge, and direction defines the orientation of the edge present at each pixel.

The magnitude $mag(x,y)$ can be given as,

$$mag(x,y)=\sqrt{f_x(x,y)^2+f_y(x,y)^2} \tag{1}$$

The direction $\theta(x,y)$ can be given as,

$$\theta(x,y)=\arctan\frac{f_y(x,y)}{f_x(x,y)} \tag{2}$$

C. Non-Maximum suppression

The edges are made thin by converting the blurred edges of the image gradients into sharp edges. This can be done by suppressing the minima (preserving local maxima) in the gradient image. This step involves,

(i) Quantization is performed over 0°-360° and it is split into eight bins (0°, 45°, 90°, 135°, 180°, 225°, 270° and 315°).

(ii) The current pixel is assumed as one if the gradient magnitude of the current pixel is greater than the magnitude of 8-D neighbors along the gradient orientation of the current pixel; otherwise, it is assumed as zero.

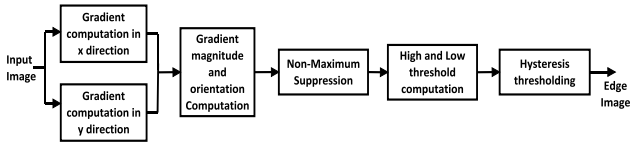

Fig. 1. Implementation of canny edge detection algorithm.

D. High and low threshold calculation

The potential edges are determined by high and low threshold. These values are calculated based on gradient magnitude histogram for the whole image. The high threshold value is chosen based on P_1 percentage of the total number of pixels. The lower threshold value is P_2 percentage of a high threshold value. The percentage values are calculated by using the cumulative distribution function (CDF). The values of P_1 are P_2 are typically chosen as 20 % and 40% respectively.

E. Hysteresis thresholding

Finally, the hysteresis thresholding creates the continuous edge map by removing (i) the edge discontinuities present within an image, and (ii) the edges caused due to noise and illumination variation. The gradient magnitude of the each pixel is compared with the high and low threshold values. If the gradient magnitude is greater than the high threshold value, then it is marked as a strong edge. If the gradient magnitude is lower than the low threshold value, then it is marked as a weak edge. If the gradient magnitude value is in between high and low threshold value, then further investigation is done with eight neighbors values. If they are connected, then it is interpreted as a strong edge otherwise it is taken as a weak edge.

III. Proposed Design Strategy

In proposed design strategy the following are the key contributions:

i) Approximate methods for the computation of gradient magnitude and orientation.

ii) Cumulative histogram based median filter architecture for sliding window.

A. Gradient computation based on Sobel operator

The simple Sobel operator [3] is used initially for edge detection. There is one motivation for the employment of the Sobel operator rather than other operators mentioned throughout the literature [1]-[3]. The first is that the more weight is allocated to the pixel intensities around the edges. Fig. 2 shows the Sobel operator kernels in horizontal(x) and vertical(y) direction. For each pixel located at coordinate (x,y), the gradients in x-axis and y-axis, which are denoted by $f_x(x,y)$ and $f_y(x,y)$ needs to be computed and given in (3) & (4),

$$f_x(x,y)=(p_2-p_0)+2(p_5-p_3)+(p_8-p_6) \tag{3}$$

$$f_y(x,y)=(p_6-p_0)+2(p_7-p_1)+(p_8-p_2) \tag{4}$$

where, $p_0=f(x\text{-}1,y\text{-}1)$, $p_1=f(x,y\text{-}1)$, $p_2=f(x+1,y\text{-}1)$, $p_3=f(x\text{-}1,y)$, $p_5=f(x+1,y)$, $p_6=f(x\text{-}1,y+1)$, $p_7=f(x,y+1)$ and $p_8=f(x+1,y+1)$.

In practice, Sobel operator cannot identify the edges with high accuracy. As shown in (3) and (4), the subtraction operations need to be performed to calculate the gradients. These subtraction operations are performed by using two's

complement subtraction. The overflow bit indicates the sign of the resultant number. If overflow occurs, then it is a positive value. If there is no overflow, then it is a negative value. The correct output is produced only when the width input operands made to equal to the width of the resulting value. In our implementation, two's complement subtraction and shift based multiplication operations are used to reduce the hardware cost. Fig. 3 shows the architecture of $f_x(x,y)$ and $f_y(x,y)$.

B. Gradient Magnitude and orientation computation

1) Gradient Magnitude calculation

The magnitude calculation of a single pixel is given in (3), requires two-squared and one-square root operation. The costs paid to realize the above-mentioned operations in hardware are high. In our design, low-cost hardware implementation technique named square root approximation suggested in [13] is adopted to compute the magnitude $mag(x,y)$. It is given as,

$$mag(x,y) \approx max((0.875a+0.5b),a) \quad (5)$$

where, $a = max(f_x(x,y), f_y(x,y))$

$b = min(f_x(x,y), f_y(x,y))$

Fig. 4 shows the architecture of square root approximation, where ADD block computes the addition of two inputs. By this technique, shift units are employed to avoid the multiplication operation.

2) Gradient orientation calculation

The orientation $\theta(x,y)$ of each pixel lies between 0°-180°. Calculation of each angle leads to high hardware cost. To reduce the hardware cost the orientations are spaced uniformly and it is divided into nine bins, as shown in Fig. 5.

In hardware implementation, the trigonometric functions are calculated by COordinate Rotation DIgital Computer (CORDIC) module by rotating the vectors. The transformation in the coordinates is performed by using a series of shift-and-add operations [14], [15]. The down-side of the CORDIC module is that it requires many iterations eventually it leads to the utilization of high hardware cost to achieve acceptable precision.

-1	0	1
-2	0	2
-1	0	1

a) Sobel operator kernel in x direction

-1	-2	-1
0	0	0
1	2	1

b) Sobel operator kernel in y direction

Fig. 2. Sobel Operator.

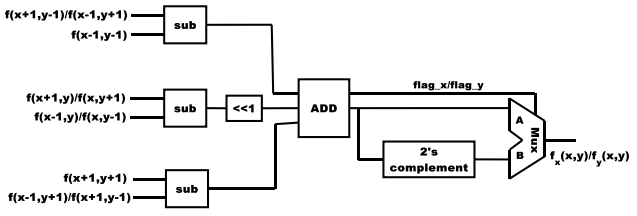

Fig. 3. Architecture of $f_x(x,y)$ and $f_y(x,y)$.

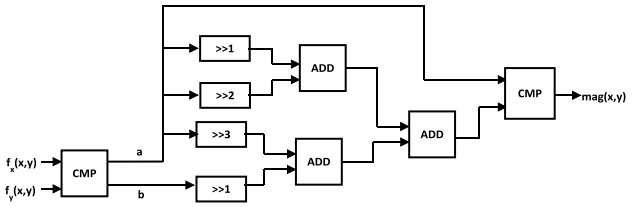

Fig. 4. Architecture of square root approximation.

To find the bin for $\theta(x,y)$ belongs, is calculated by modifying $\theta(x,y)$ as $\tan\theta(x,y)$ and the particular bin is defined between the two angle values. For simplification (2) can also be written as,

$$\tan\theta(x,y) = \frac{f_y(x,y)}{f_x(x,y)} \quad (6)$$

To reduce the hardware-consuming component, the shift based orientation calculation method is employed. The approximated values based on the shift operations are listed in Table I.

The tabulated values are lies in the first quadrant, the other orientations till tan 180° lies in the second quadrant has the negative sign, but the values are similar to the first quadrant. The quadrant value is calculated by using the quadrant_flag reduces the resource utilization. The quadrant_flag computation shown in Fig. 6 with input flag_x and flag_y are calculated as described in Fig. 3 during the computation of gradient values.

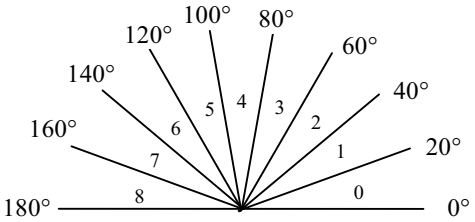

Fig. 5. Nine bins of Gradient orientation.

TABLE I
SHIFT-BASED TANGENT VALUE COMPUTATION

Tangent	Approximate value
tan 0°	0
tan 20°	$2^{-2}+2^{-3}$
tan 40°	$2^{-1}+2^{-2}+2^{-4}$
tan 60°	$1+2^{-1}+2^{-2}$
tan 80°	$1+2^{2}+2^{-1}+2^{-3}+2^{-5}$

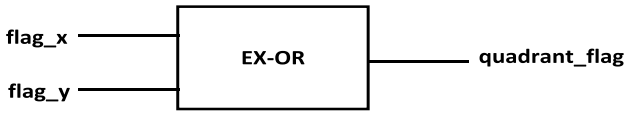

Fig. 6. Computation of quadrant_flag.

The value of $\tan\theta(x,y)$ can be expressed as,

$$\tan\theta_j(x,y) < \tan\theta(x,y) \leq \tan\theta_{j+1}(x,y) \quad (7)$$

Rewriting $\tan\theta(x,y) = \frac{f_y(x,y)}{f_x(x,y)}$, it becomes

$$\tan\theta_j(x,y) < \frac{f_y(x,y)}{f_x(x,y)} \leq \tan\theta_{j+1}(x,y) \quad (8)$$

The bin value is calculated by satisfying the given condition. The condition is derived from (8).

$$f_x(x,y) * \tan\theta_j(x,y) < f_y(x,y) \leq f_x(x,y) * \tan\theta_{j+1}(x,y) \quad (9)$$

Fig. 7 illustrates architecture for the computation of, $f_x(x,y) * \tan 20°$ and Fig. 8 is the architecture used to find the bin.

C. Non-Maximum suppression

The Non-Maximum Suppression (NMS) identifies that the current pixel is local maxima in the direction of a gradient. It requires 8-neighbors (3x3 window) for the determination of local maxima. If the maxima is present in the current pixel, then it is taken as edge pixel otherwise it is suppressed. It helps to make the edges thin. The selector is used to find the pixels that are having the direction equivalent to current pixel direction. Fig. 9 shows the architecture of Non-Maximum suppression.

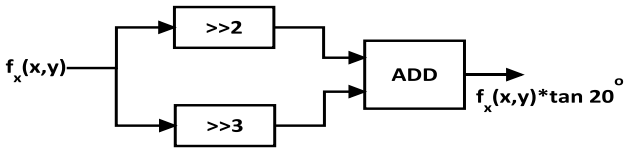

Fig. 7. Architecture for the computation of $f_x(x,y) * \tan 20°$.

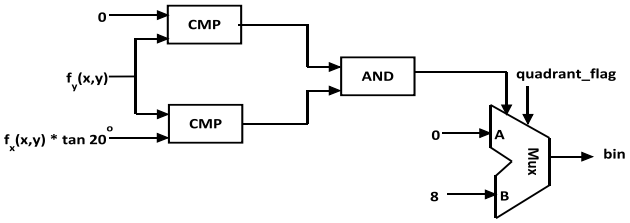

Fig. 8. Architecture to calculate the bin.

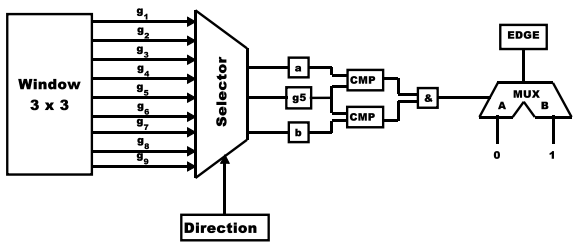

Fig. 9. Architecture of Non-maximum suppression.

D. Low and High Threshold Computation

The performance of the canny edge detector depends on the value of the thresholds. These thresholds need to be calculated to high precision. The median value of the gradient magnitude helps to find the value of thresholds and is calculated based on the histogram of gradient magnitude. Fig. 10 shows the Histogram based median filter architecture. The median value is computed by sliding window technique. Due to sliding window technique, the hardware cost is reduced. By using the rule of thumb [17], the low and high threshold values are calculated. The low and high threshold values are given as

$$\begin{aligned} &\text{Low threshold } (Th_L) = 0.33 * \text{Median value} \\ &\text{High threshold } (Th_H) = 1.66 * \text{Median value} \end{aligned} \quad (10)$$

The threshold values are computed by employing shift-based techniques. Fig. 11 (a) and (b) shows the architecture for Low and High threshold values calculation.

E. Hysteresis thresholding

Non-maximum suppression technique finds the strong edges. The edges may be affected by noise and illumination changes. Those edges are preserved by this technique. If the computed gradient magnitude is greater than the high threshold, then it is the strong edge pixel and those edges are preserved. If gradient magnitude is lower than the low threshold, those edges are suppressed. If the gradient magnitude value lies between low and high threshold then further investigation is done. The current pixel is considered as a strong edge pixel if any of the neighbors of the current pixel is strong edge pixel; otherwise it is considered as a weak edge pixel and it is suppressed. Fig. 12 shows the architecture of hysteresis thresholding.

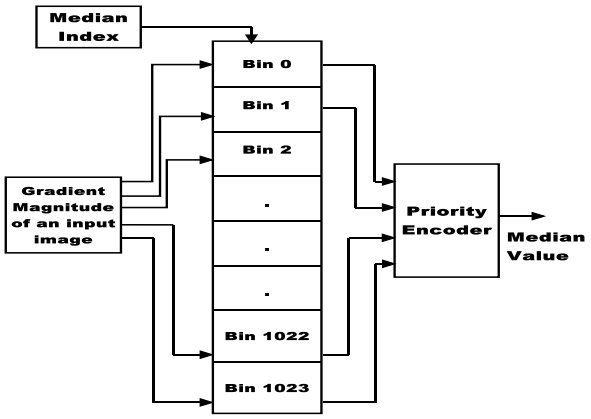

Fig. 10. Histogram based median filter architecture.

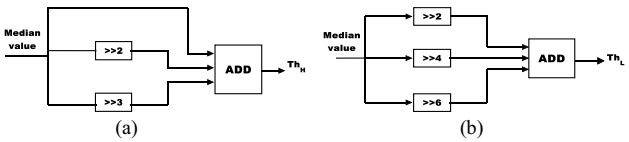

Fig. 11 Architecture for (a) Th_H (b) Th_L calculation.

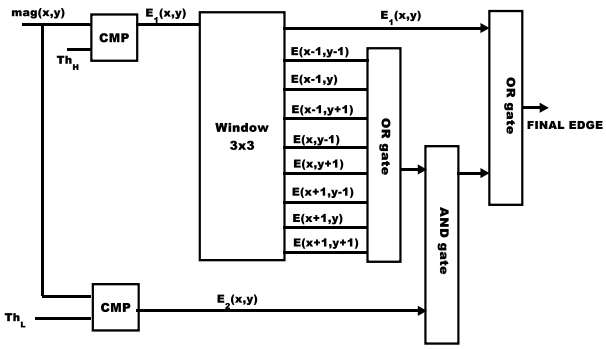

Fig. 12 Architecture of hysteresis thresholding.

F. Pipelined Architecture

The architecture consists of 6 major blocks as shown in Fig. 1. The edge detection algorithm adopts 3x3 mask. In order to speed up the operation of the VLSI hardware a 3 stage pipelined architecture has been proposed for 4 blocks: Gradient computation in x and y direction, Magnitude and orientation computation, Non- Maximum Suppression, Bin for threshold calculation as shown in Fig. 13(a).

In this design, each function requires one clock cycle. As shown in Fig. 13(b) to process the pixel covered by 3x3 mask requires 15 clock cycles instead of 36 clock cycles that helps to reduce the output latency. Thus the pipelined architecture increases the speed of operation and are implemented using Xilinx. The implementation results are summarized in the next section.

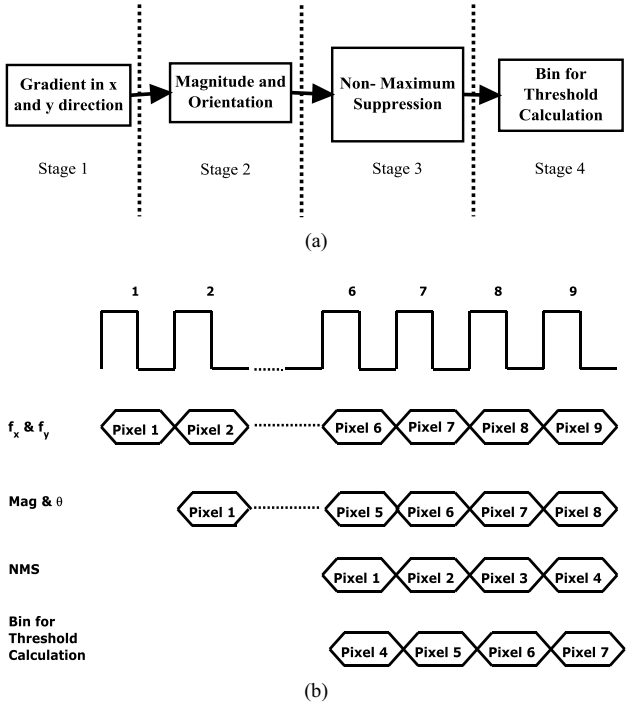

Fig. 13 Pipelining Stages (a) VLSI architecture for detecting edges (b) Clock used for processing the pixels.

IV. Experimental Results

To evaluate the performance of the developed system, the iris images are chosen for a test application. The Iris images [16] of size 512x512 are considered as an input data. The proposed canny edge detector is applied to the input data, and the result is shown in Fig. 14. As shown in Fig. 14 the proposed Canny edge detection algorithm yields comparable edge detection results as compared with the conventional Canny edge detector.

Furthermore, the VLSI architecture is implemented and verified on FPGA Xilinx Virtex-5 board. Table II lists the resource utilization of proposed architecture, and it is compared with other existing techniques. The result shows that it utilizes less cost compared to the existing architectures due to approximation method and sliding window technique for histogram based median filter architecture. The proposed architecture is efficient because of low hardware cost and visualization result.

Fig. 13(b) shows the pipeline implementation of edge detection algorithm. Such a pipeline design increases the throughput. To detect the image of size 512x512 totally 45,634 clock cycles are required. For the operating frequency of 100MHz, the overall processing time include memory read/write for fetching the data/storing the edge results and processing of edge detection algorithm is about 1ms to detect edges of the image, which is sufficient for real-time applications.

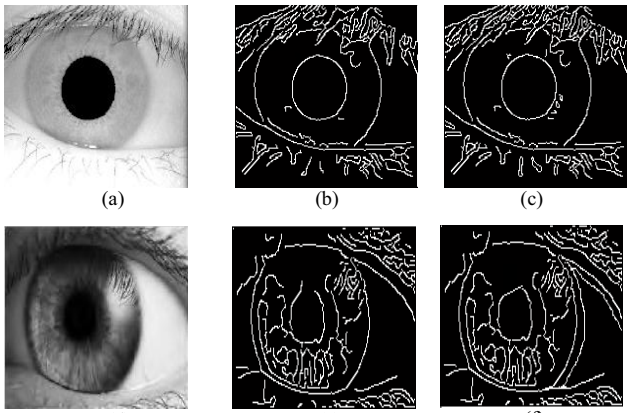

Fig. 14. (a), (d) 512x512 Iris image; edge map of (b), (e) the conventional canny edge detector [5] and (c), (f) the proposed canny detection algorithm.

TABLE II
Comparison of Resource Utilization and Time

Edge Detection Methods	Image Size	FPGA Device	Used Slices	Norm. Time (ms)
[7]	256x256	Xilinx Vertex-E	-	2.688
[8]	256x256	Altera Stratix II	1530/48532	2.64
[9]	512x512	Xilinx Virtex-5	4553/71680	1.669
[10]	360x280	Altera Cyclone	-	0.72
[11]	512x512	Xilinx Virtex-5	-	0.47
[12]	512x512	Xilinx Virtex-5	23904/37440	0.372
Proposed	512x512	Xilinx Virtex-5	5980/28495	0.456

V. CONCLUSION

The proposed canny edge detector uses the pipelining technique and handles the multiple blocks at the same time. It helps to achieve fastest working speed. By using the approximation methods, the hardware cost is reduced. The histogram based median filter architecture uses sliding window technique and results in low hardware utilization. The proposed algorithm is scalable and has high accuracy. It can be able to detect all psycho-visually important edges of the image. It takes only 1ms to detect the edges of 512x512 images. Hence, the proposed architecture supports real-time edge detection with less hardware utilization. In future, this system will be implemented to test noisy images in ASIC-based platforms for real time applications.

REFERENCES

[1] L. G. Roberts. "Machine perception of 3D solids," Optical and Electro-Optical Information Processing. MIT Press, 1965.

[2] R. C. Gonzalez, R. E. Wood. "Digital image processing," Second Edition. Prentice Hall, 2002.

[3] N. Kanopouls, N. Vasanthavada, R.L.Baker, "Design of an image edge detection filter using the sobel operator", IEEE journal of Solid-State Circuits, Apr.1988, pp.358-367.

[4] E. R. Davies. "Constraints on the design of template masks for edge detection," Pattern Recognition Lett., vol. 4, pp. 111-120, Apr,1986.

[5] J. F. Canny, "A computation approach to edge detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 769-798, November 1986.

[6] Y. Luo and R. Duraiswami, "Canny edge detection on nvidia cuda," Computer Vision and Pattern Recognition Workshop, vol. 0, pp. 1–8, 2008.

[7] D. V. Rao and M. Venkatesan, "An efficient reconfigurable architecture and implementation of edge detection algorithm using Handle-C," IEEE Conference on Information Technology: Coding and Computing (ITCC), vol. 2, pp. 843 – 847, Apr. 2004.

[8] H. Neoh, A. Hazanchuck, "Adaptive edge detection for real-time video processing using FPGAs," Application notes, Altera Corporation, 2005.

[9] C. Gentsos, C. Sotiropoulou, S. Nikolaidis, N. Vassiliadis, "Real-Time canny edge detection parallel implementation for FPGAs," IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 499-502, Dec. 2010.

[10] W. He and K. Yuan, "An improved canny edge detector and its realization on FPGA," IEEE Proceedings of the 7th World Congress on Intelligent Control and Automation (WCICA), pp. 6561 –6564, June 2008.

[11] Xiaoyang Li, Jie Jiang, Qiaoyun Fan, "An improved real-time hardware architecture for canny edge detection based on FPGA", in Third international conference on Intelligent Control and Information Processing, July 2012.

[12] Qian Xu, Srenivas Varadarajan, Chaitali Chakrabarti and L. J. Karam, "A distributed canny edge detector: algorithm and FPGA implementation", IEEE Transactions on Image Processing, 2014, pp. 1-16.

[13] D. D. Gajski, Principles of Digital Design. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.

[14] H. T. Ngo and V. K. Asari, "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images," IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 3, pp. 436–444, Mar. 2005.

[15] J. D. Bruguera, N. Guil, T. Lang, J. Villalba, and E. L. Zapata, "CORDIC based parallel/pipelined architecture for the Hough transform," J. VLSI Signal Process., vol. 12, no. 3, pp. 207–221, Jan. 2001.

[16] http://biometrics.idealtest.org/dbDetailForUser.do?id=4.

[17] Kerry. D. Wong, "Canny edge detection autothersholding", May 2009.

[18] Y. Daanial Khan, S. Afzal Khan, Farooa Ahmad, Saeed Islam, "Iris recognition using image moments and k-means algorithm", The Scientific World Journal, vol. 2014, Article ID 723595.