International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.2, Mar-Apr 2012 pp-552-554 ISSN: 2249-6645

Performance and Analysis of Edge detection using FPGA Implementation

R. Ponneela Vignesh[1], R. Rajendran[2]
[1]*PG Student/VLSI Design, SNS College of Technology, Coimbatore, Tamilnadu.*
[2]*Dept.of ECE, SNS College of Technology, Coimbatore, Tamilnadu.*

Abstract

Edge detection serves as a pre-processing step for many image processing algorithms such as image enhancement, image segmentation, tracking and image/video coding. The edge detection is one of the key stages in image processing and object recognition. This paper present a Canny edge detection algorithm that results in significantly reduced memory requirements, decreased latency and increased throughput with no loss in edge detection performance. This edge detection algorithm is based on MATLAB simulation and FPGA implementation.

Keywords: *Canny edge detector, MATLAB and FPGA*

I. Introduction

Edge detection is a basic operation in image processing, it refers to the process identifying and locating sharp discontinuities in an image, the discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. It is a very important first step in many algorithms used for segmentation, tracking and object recognition [1].

There are an extremely large number of edge detection operators available, each designed to be sensitive to edges, typically it reduces the memory size and the computation cost[2] the edge detection algorithms are implemented using software, with advance in very large scale integration(VLSI) technology, their hardware implementation become an attractive alternative, especially for real-time applications, Many edge detection algorithms such as Robert detector, Prewitt detector, Kirsch detector, Gauss-Laplace detector and Canny edge detector have been proposed[3], among these algorithms has be widely used in the field of image processing because of its good performance[3]. It operates on two rows of pixels at a time this reduces the memory requirement at the expense of a decrease in the throughput [2]. In order to reduce memory requirements, decrease latency and increased throughput is proposed in [1]. An absolute different mask edge detection algorithm and its pipelined VLSI architecture for real-time application. But the edge detector in offers a trade-off between precision, cost and speed, and its capability to detect edges is not as good as the Canny algorithm [4].

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 reconfigurable hardware architecture. The original Canny edge detection algorithm computes the high and low thresholds for edge detection based on the entire image statistics which prevents the processing on individual block independently.

II. Canny edge detection algorithm

The Canny edge detection algorithm is known to many as the optimal edge detector. The first and most obvious is low error rate. It is the important that edges occurring in images should not be missed and that there are no responses to non-edges. The second criterion is that the edge points be well localized. A third criterion is to have only one response to a single edge.

This was implemented because the first two were not substantial enough to completely eliminate the possibility of multiple responses to an edge. Based on these criteria, the canny edge detector first smoothes the image to eliminate and noise. It then finds the image gradient to highlight regions with high spatial derivatives After the edge directions are known, non maximum suppression now has to be applied Non-maximum suppression is used to trace along the edge in the edge direction and suppress any pixel value that is not considered to be an edge. This will give a thin line in the output image.

The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a non edge). To avoid this hysteresis uses 2 thresholds, a high and a low. If the magnitude is above the high threshold, it is made an edge.
This model was based on a step edge corrupted by additive white Gaussian noise. Canny edge detection developed an approach to derive an optimal edge detector based on three criteria related to the detection performance. The Canny edge detection algorithm block diagram is shown in below.

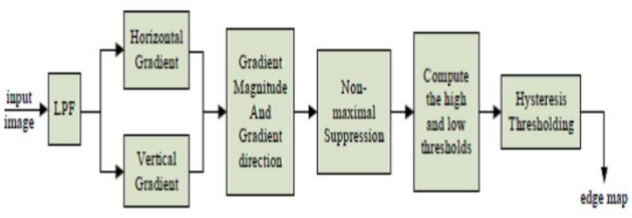

Fig 1 : Block of Canny edge detection

The Canny edge detection algorithm consists of the following steps:

- Calculating the horizontal gradient and vertical gradient at each pixel location by convolving the image with partial derivatives of a 2D Gaussian function.

- Smoothing the input image by Gaussian mask. The output smoothed the image.
- Computing the gradient magnitude and direction at each pixel location.
- Applying non-maximum suppression (NMS) to thin edge.
- Computing the hysteresis high and low thresholds based on the histogram of the magnitudes of the gradients of the entire image.
- Performing hysteresis thresholding to determine the edge map.
- The low pass filtering is achieved by taking the average pixel values.

It can be shown that convolving an image with a symmetric 2D Gaussian and then differentiating in the direction of the gradient forms simple and effective directional operator, which meets the three criteria mentioned above

Suppose the $G(x, y)$ is a 2D Gaussian and $I(x, y)$ is the image, then the smoothed image $H(x, y)$ as

$$H(x, y) = G(x, y) * I(x, y)$$

Where the $H(x, y)$ is denoted as horizontal gradient, the $G(x, y)$ is denoted as gradient magnitude and $I(x, y)$ as smoothed image.

The data obtains usually contains some spurious responses in the non maximal suppression (NMS). This is called the streaking problem and is quite common in the edge detection problem. These streaking can be eliminated by using a threshold with hysteresis.

The Canny edge detector will be carried out in following four steps:

1. Image smoothing
2. Gradient calculation and directional Non-Maximum suppression
3. Calculating thresholds
4. Thresholding with hysteresis.

1. Smooth the image with a two dimensional Gaussian. In most cases the computation of a two dimensional Gaussian is costly, so it is approximated by two one dimensional Gaussians, one in the x direction and the other in the y direction.

2. Take the gradient of the image. This shows changes in intensity, which indicates the presence of edges. This actually gives two results, the gradient in the x direction and the gradient in the y direction.

3. Non-maximal suppression. Edges will occur at points the where the gradient is at a maximum. Therefore, all points not at a maximum should be suppressed. In order to do this, the magnitude and direction of the gradient is computed at each pixel. Then for each pixel check if the magnitude of the gradient is greater at one pixel's distance away in either the positive or the negative direction perpendicular to the gradient. If the pixel is not greater than both, suppress it.

4. Edge Thresholding. The method of thresholding used by the Canny Edge Detector is referred to as "hysteresis". It makes use of both a high threshold and a low threshold. If a pixel has a value above the high threshold, it is set as an edge pixel. If a pixel has a value above the low threshold and is the neighbour of an edge pixel, it is set as an edge pixel as well. If a pixel has a value above the low threshold but is not the neighbour of an edge pixel, it is not set as an edge pixel. If a pixel has a value below the low threshold, it is never set as an edge pixel.

First the edge detection algorithm is implemented by applying a Gaussian filter on the whole image. Then the processed image data are used as the input to calculate the gradient at each pixel level and these gradient values are used to calculate the direction and the magnitude at a certain pixel.

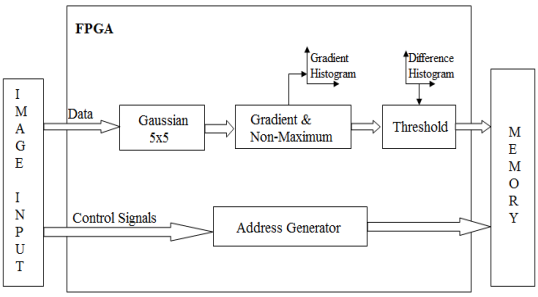

Fig 2 : Design flow of Canny edge detection

In order to implement the Canny edge detector algorithm, some steps must be followed. The following steps are executed sequentially:

- The first step is to filter out any noise in the original image before trying to locate and detect any edges.
- The Gaussian filter mask is used exclusively in Canny edge detection algorithm.
- The Gaussian smoothing can be performed using standard convolution methods. a convolution mask is much smaller than the actual image
- The larger width is the Gaussian mask and the lower is the detector sensitivity.
- The localization error in the detected edges also increases slightly as the Gaussian width is increased. The Gaussian in implementation is shown below.

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Fig 3: Discrete approximation to Gaussian function

- The second step is to find the edge strength by taking the gradient of the image.
- Then, the approximate absolute gradient magnitude (edge strength) at each point can be found.
- It uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns).
- The other convolution masks estimating the gradient in the y-direction (rows). They are shown below:

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

Gx

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

Gy

The magnitude or edge strength of gradient is then approximated using the formula:

$$|G| = |Gx| + |Gy|$$

The direction of the edge is computed using the gradient in the x and y directions. Whenever the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, Formula for finding edge direction:

$$\theta = \tan^{-1} (Gy / Gx)$$

If GY has a value of zero, the edge direction will equal to 0 degrees. Otherwise the edge direction is equal to 90 degrees.

III Simulation results

The algorithm performance was tested using a variety of images. The simulations results are obtained in MATLAB. Here the results are tested by two images.

(a). Finger print and

(b). Eye

Simulation output:

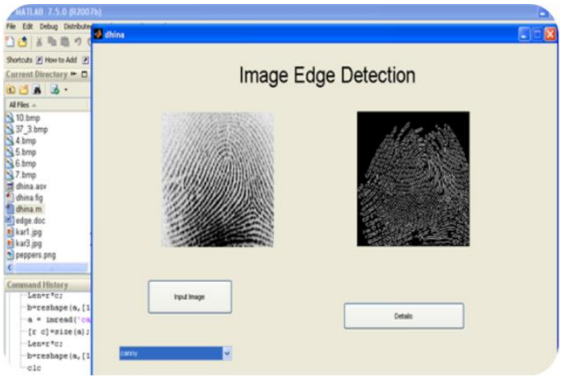

Fig 4: Matlab simulation result for Finger print image

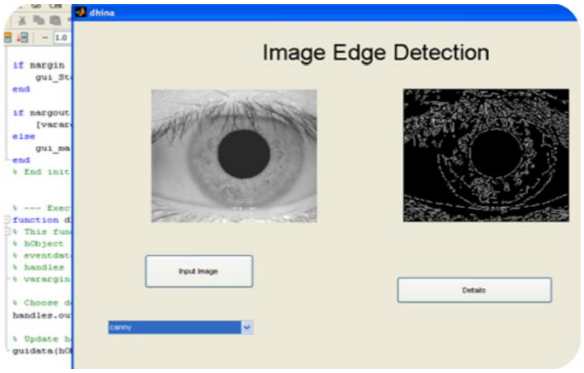

Fig 5: Matlab simulation result for Eye image

CONCLUSION

The result of the image can be obtained by using MATLAB.Thus compared to other edge detection algorithm, Canny edge detection algorithm use probability for finding error rate localization and response in various images. Thus even blurred image and images with noise can be detected accurately using this Canny edge detection algorithm as there is a specification of improving signal to noise ratio. It results in a significant speedup and successfully detects the edges of the images.

FUTURE WORK

Thus as a future enhancement, the implementation of Canny edge detection algorithm can be perform on FPGA platform and tested in system C. Thus tested in system C which can overcome the major disadvantage of canny edge detection algorithm of complex computation and time consuming. It is capable of supporting fast real time edge detection for images and videos.

REFERENCES

[1] S. Varadarajan, C. Chakrabarti, L. J. Karam, and J. M.Bauza, "A distributed psycho-visually motivated Canny edge detector," IEEE ICASSP, pp.882-825, Mar.2010.

[2] 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.

[3] W. He and K. Yuan, "An improved Canny edge detector and its realization on FPGA," WCICA, pp. 6561 –6564, Jun. 2008.

[4] 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.

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

[6] D. Marr and E. Hildreth, "Theory of edge Detection," Proc. Royal Soc. of London, series B, vol. 207, pp. 187-217, 1980.

[7] D. Demigny, and T. Kamle," A discrete expression of Canny's criteria for step edge detector performances evaluation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(6), pp. 1199-1211, 1997.

[8] L. Torres, M. Robert, E. Bourennane, and M. Paindavoine, "Implementation of a recursive real time edgedetector using retiming techniques," *VLSI*, pp. 811 –816, Aug. 1995.