ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 6, Special Issue 5, March 2017

National Conference on Advanced Computing, Communication and Electrical Systems - (NCACCES'17)

24th - 25th March 2017

Organized by

C. H. Mohammed Koya
KMEA Engineering College, Kerala- 683561, India

Implementation of Canny Edge Detection Algorithm in FPGA Using HDL

Dhanalakshmi Renganathan [1], Riyas K S [2] , Anu George[3]

P.G. Student, Department of Electronics and Communication Engineering, Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India[1]

Associate Professor, Department of Electronics and Communication Engineering, Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India[2]

Assistant Professor, Department of Electronics and Communication Engineering, Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India[3]

ABSTRACT: Edge detection is the process of finding areas of an image where a large change in intensity occurs. Edge detection significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. An edge is the boundary between an object and the background, hence if the edges are identified accurately in an image all its objects can be located and basic properties such as area, perimeter and shape of an image can be measured. The Canny edge detection algorithm is known as the Optimal edge detector. The Canny edge detector is one of the most widely-used edge detection algorithms due to its superior performance. The data of edge detection is very large, so the achievement of high speed of image processing is a difficult task. Field Programmable Gate Array (FPGA) can overcome this difficult task and it is an effective device to realize real-time parallel processing for vast amounts of image and video data. The proposed work shows the implementation of Canny Edge detection algorithm on FPGA.

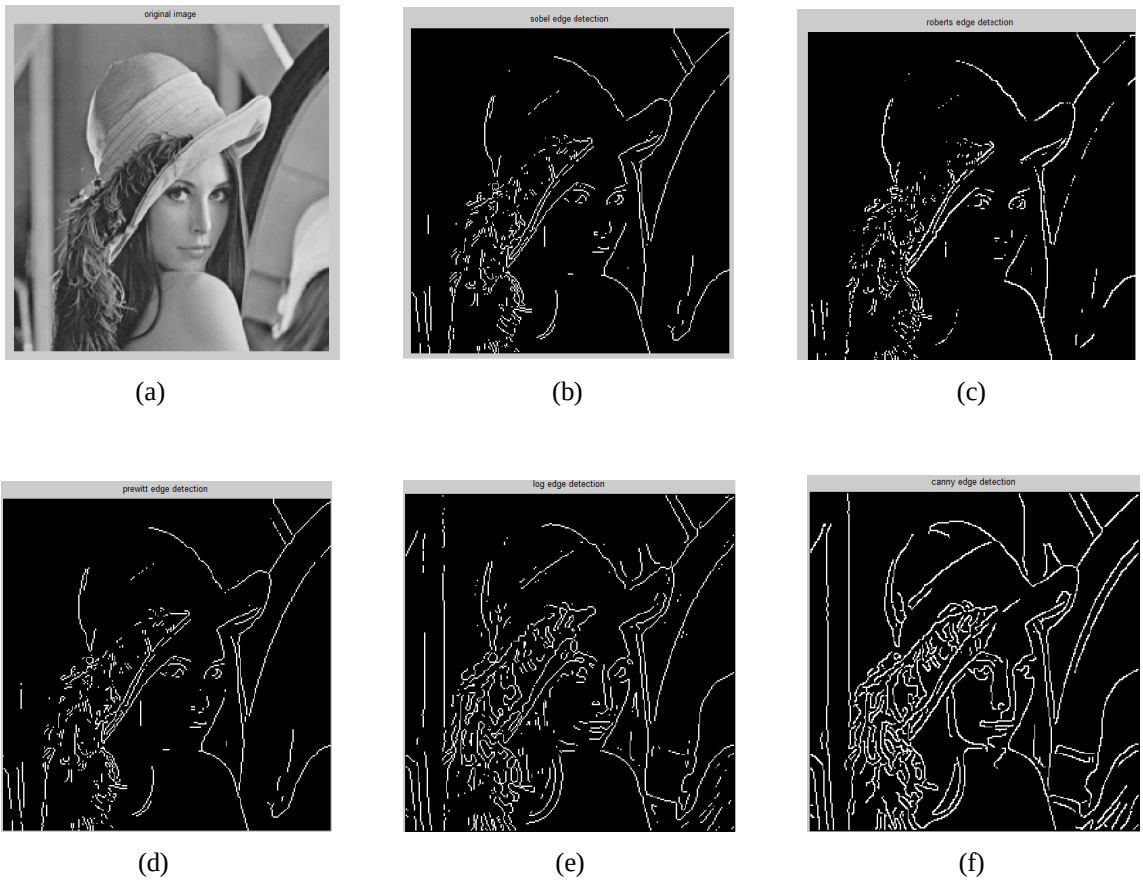
KEYWORDS: Image Processing, Canny Edge Detector, Xilinx, MATLAB, FPGA, VGA monitor.

I. INTRODUCTION

Edge is one of the prominent features in the image processing applications. The main goal of edge detection is to locate and identify sharp discontinuities [1] from an image. These discontinuities are due to abrupt changes in pixel intensity which characterizes boundaries of objects in a scene as well as give boundaries between different regions in the image. Such boundaries are used to identify objects for segmentation and matching purpose. These object boundaries are the first step in many of computer vision algorithms like edge based obstacle detection, edge based face recognition, edge based target recognition, image compression etc. So the edge detectors are required for extracting the edges. There are many edge detection operators available.

Strong intensity contrast in an image shows the edge of an image, and the edges will identify the boundary of an image. Edge detection is a very important first step in many algorithms used for image segmentation, tracking and image/video coding. The Canny edge detection is predominantly used due to its ability to extract significant edges. Edge detection, as a basic operation in image processing, has been researched extensively. A lot of edge detection algorithms, such as Robert detection, Prewitt detection, Kirsch detection, Gauss-Laplace detection and Canny detection 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 detection is predominantly used in many real-world applications due to its ability to extract significant edges with good detection and good performance.

Here followed a list of criteria to improve current methods of edge detection. The first and most obvious is low error rate. It is important that edges occurring in images should not be missed and that there be no responses to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge.The proposed Canny edge detection algorithm is designed using Verilog HDL and simulation is done using MATLAB and Xilinx ISE Design Suit. The input to the verilog code is in the form of text/pixel values. Image processing in verilog is a big topic .An image is almost always a 2D matrix. But processing a 2D image in FPGA might not be a good idea. It might lead to excessive delays and resources. So we convert the 2D image into a linear 1 D array. This data can be stored in a RAM or ROM. To get the most efficient memory module, its recommended that, we use the Block Memory Generator module available in coregen to do this. The edges of an image are determined by comparing the neighboring text pixels. The edge detected output is in the form of text/pixel values. MATLAB is used for the conversion of the input image to text/pixel values and output edge detected text/pixel output to the edged output. The proposed design can be implemented for any format of images like jpg, gif, bmp etc. The implementation of canny edge detection is done in Spartan 6 FPGA board.



(a) (b) (c)

(d) (e) (f)

Fig. 1: Different Edge Detectors; (a) Original Image (b) Sobel Edge Detection (c) Robert Edge Detection (d) Prewitt Edge Detection (e) Log Edge Detection (f) Canny Edge Detection.

II. LITERATURE SURVEY

R. Ponneela Vignesh et.al in [1] implemented the Canny edge detection algorithm in FPGA device, and it is applicable for image segmentation, image tracking, image coding etc. In this paper Canny edge algorithm reduces memory requirements, decreased latency, increased through output with no loss in edge detection performance and Canny edge detection algorithm use probability for finding error rate localization and response in various images. Chandrashekar N.S et.al in [2] explains the 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. Tejaswini H.R et.al in [3] explains that the edge detection is an eloquent step in image processing and in object recognition. The Canny edge detection is called as optimal detection due to its good performance. Samina Jafar et.al in [4] explainsthe edge detection is one of the key stages in image processing and objects reorganization. The Canny Edge Detection is one of the most widely used edge detection algorithm due to its good performance. T.Rupalatha et.al in [5] explains edge detection is one of the basic operation carried out in image

processing and object identification. In this paper, the 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. FPGAs are often used as implementation platforms for real time image processing applications because their structure is able to exploit spatial and temporal parallelism. Such parallelism is subject to the processing mode and hardware constraints of the system. FPGA have become the dominant form of programmable logic in comparison to devices like PLA (Programmable Array Logic) and Complex Programmable Logic Devices (CPLD). FPGA can implement far target logic functions FPGA provides designers with reconfigurable logic that can be reprogrammed on application specific basis. This increases the flexibility. Edge detection is one of the fundamental steps in image processing, image analysis, image pattern recognition and computer vision techniques.

III. PROPOSED METHODOLOGY

Our aim is to design a canny edge detector in FPGA using verilog and then it is compared with the result obtained from MATLAB.The Canny edge detectionin verilog consists of following steps:

- The input image is converted into text/pixel values using MATLAB program.
- The pixel values are stored in external text file, and are captured by Verilog HDL with help of commands like “sreadmemh” or “sreadmemb”.
- Edges are finding in the image using Verilog HDL with Xilinx software and final image value is stored in another file.

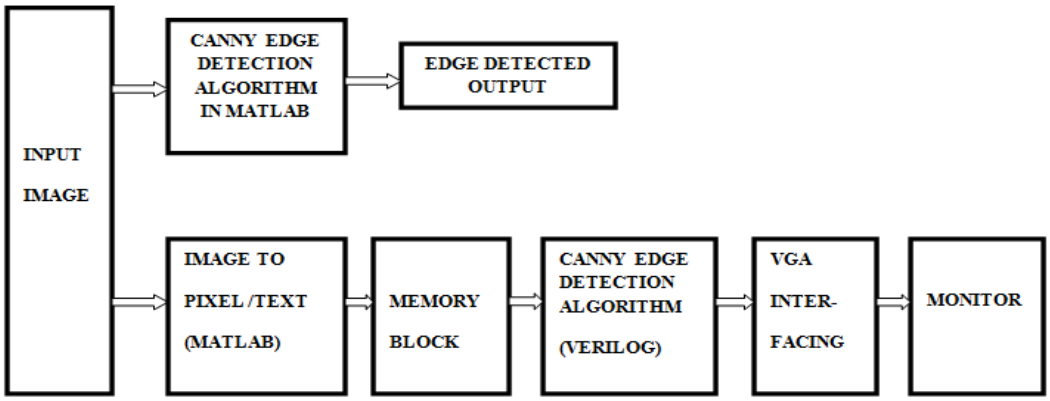


Fig. 2: Block Diagram of the Proposed Work

IV.CANNY EDGE DETECTION ALGORITHM

The popular canny edge detector uses the following steps to find contours presents in the image. The first stage is achieved using Gaussian smoothing. The resulting image is sent to the PC that sends it back to the gradient filter. After the gradient calculation non maximal suppression is used. After the edge directions are known, non maximum suppression is applied. Non maximum suppression is used to trace pixels along the gradient in the edge direction and compare the value. Finally, hysteresis is used as a means of eliminating streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 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 are also selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1.

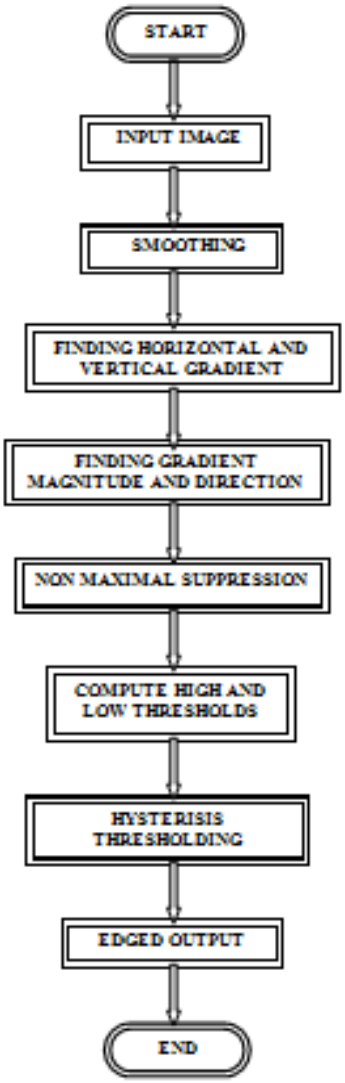


Fig. 3: Flow Chart of Canny Edge Detection

Step 1 Gaussian Smoothing

In order to implement the canny edge detector algorithm, a series of steps must be followed. The first step is to filter out any noise in the original image before trying to locate and detect any edges. And because the Gaussian filter can be computed using a simple mask, it is used exclusively in the Canny algorithm. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. Image smoothing is the first stage of the canny edge detection. The pixel values of the input image are convolved with predefined operators to create an intermediate image. This process is used to reduce the noise within an image or to produce a less pixilated image. Image smoothing is performed by convolving the input image with a Gaussian filter.

Step2 Gradient Calculation

After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. In this stage, the blurred image obtained from the image smoothing stage is convolved with a 3x3 Sobel operator. The Sobel operator is a discrete differential operator that generates a gradient image. Sobel operators used to calculate the horizontal and vertical gradients. To obtain the gradient image, a smoothened image from the first stage is convolved with the horizontal and vertical Sobel operators.

Step3 Gradient Magnitude and Direction

Gx and Gy are images with pixel values that are the magnitude of the horizontal and vertical gradient, respectively. The magnitude, or edge strength, of the gradient is then approximated using the formula,

$$|G| = |G_x| + |G_y| \quad (1)$$

The direction of the edge is computed using the gradient in the x and y directions. Edge direction is defined as the direction of the tangent to the contour that the edge defines in 2- dimensions. The edge direction of each pixel in an edge direction image is determined using the arctangent

$$\theta = \arctan\left(G_y / G_x\right) \quad (2)$$

Step4 Non Maximum Suppression

The edge strength for each pixel in an image obtained from equation is used in non maximum suppression stage. The edge directions obtained from equation are rounded off to one of four angles 0 degree, 45 degree, 90 degree or 135 degree before using it in non-maximum suppression. 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 (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image. Non-maximum suppression (NMS) is used normally in edge detection algorithms. It is a process in which all pixels whose edge strength is not maximal are marked as zero within a certain local neighborhood. This local neighborhood can be a linear window at different directions of length 5 pixels.

Step5 Thresholding and Hysterisis

Thresholding with hysteresis is the last stage in canny edge detection, which is used to eliminate spurious points and non-edge pixels from the results of non-maximum suppression. The input image for thresholding with hysteresis has gone through Image smoothing, calculating edge strength and edge pixel, and the Non-maximum suppression stage to obtain thin edges in the image. Results of this stage should give us only the valid edges in the given image, which is performed by using the two threshold values, T1 (high) and T2 (low), for the edge strength of the pixel of the image. Edge strength which is greater than T1 is considered as a definite edge. Edge strength which is less than T2 is set to zero. The pixel with edge strength between the thresholds T1 and T2 is considered only if there is a path from this pixel to a pixel with edge strength above T1. The path must be entirely through pixels with edge strength of at least T2. This process reduces the probability of streaking. As the edge strength is dependent on the intensity of the pixel of the image, thresholds T1 and T2 also depend on the intensity of the pixel of the image. Hence, the thresholds T1 and T2 are calculated by the canny edge detectors using adaptive algorithms. Thus all the edges in an image are detected using the canny edge detection.

V.RESULTS

Canny Edge Detection in MATLAB

The input image is converted to a gray scale image and then resized as a 256x256 image .A 3x3 Gaussian filter is designed and it is used to filter the image in order to remove the noise present in the image. For that the image is convolved with the filter designed. Then horizontal as well as vertical gradient is calculated. The magnitude or the strength of the image as well as direction is found. Non maximum suppression is then done. Finally thresholding is also done. Te output obtained from MATLAB is given.

Canny Edge Detection in Verilog

An image is almost always a 2D matrix. But processing a 2D image in FPGA might not be a good idea. It might lead to excessive delays and resources. So we convert the 2D image into a linear 1 D array. This data can be stored in a RAM or ROM. To get the most efficient memory module, it is recommended that, we use the Block Memory Generator module available in coregen to do this.

In brief the steps are:

- Create a .coe file with the image pixels data.
- Use coregen in Xilinx ISE to create a simple single port ROM of the required size and load the ROM with the data in steps 1.
- Use coregen in Xilinx ISE to create a simple single port RAM of the same size as ROM.
- Write the code for Canny Edge Detection where both these RAM and ROM are initiated as components and a process is written to get the transpose of the image stored in ROM.
- To verify that the RAM contains the edge detected image, read its contents one by one.

We have already done Steps 1,2 and 3.Image is converted to its pixel values and these pixel values are stored ina .coe file.Then using IP core generator in Xilinx ISE Design suit a single port ROM as well as single port RAM for the pixel file.The next step is to write the verilog code for Canny Edge Detection. This step is not yet done.Further working is in progress.

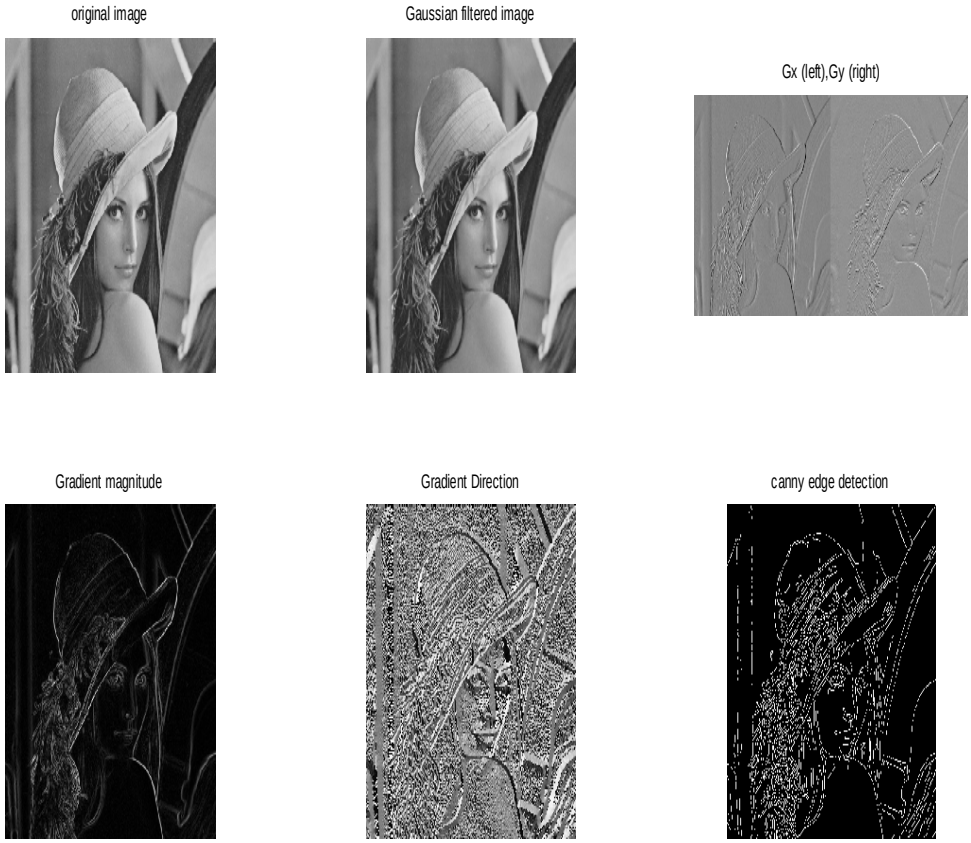


Fig 4.MATLAB experimental result of Canny Edge Detection

output - Notepad

File Edit Format View Help

```
MEMORY_INITALIZATION_RADIX=10;
MEMORY_INITIALIZATION_VECTOR=
162     162     160     162     163     160     159
162     162     160     162     163     160     159
163     160     160     160     161     159     157
160     158     159     157     160     159     154
155     157     157     157     158     157     157
156     158     156     153     157     157     156
157     157     157     156     157     156     155
158     157     157     157     155     156     156
157     157     156     153     156     156     156
156     156     158     155     157     156     157
156     156     156     156     156     157     158
157     155     154     157     157     157     158
155     154     155     158     159     159     158
155     155     157     156     158     160     159
158     158     158     157     158     159     159
157     158     158     158     159     159     160
157     158     158     157     158     158     160
160     159     158     158     158     160     162
159     159     159     159     163     164     162
161     161     160     158     161     161     160
164     162     161     160     160     163     161
160     161     162     161     159     160     162
161     159     161     161     160     161     163
163     161     163     160     159     162     164
163     162     161     160     159     163     169
161     160     159     159     160     165     171
161     159     160     161     163     168     170
160     161     160     164     167     168     169
159     160     162     165     170     171     168
```

Fig 5. Pixel Values of Image

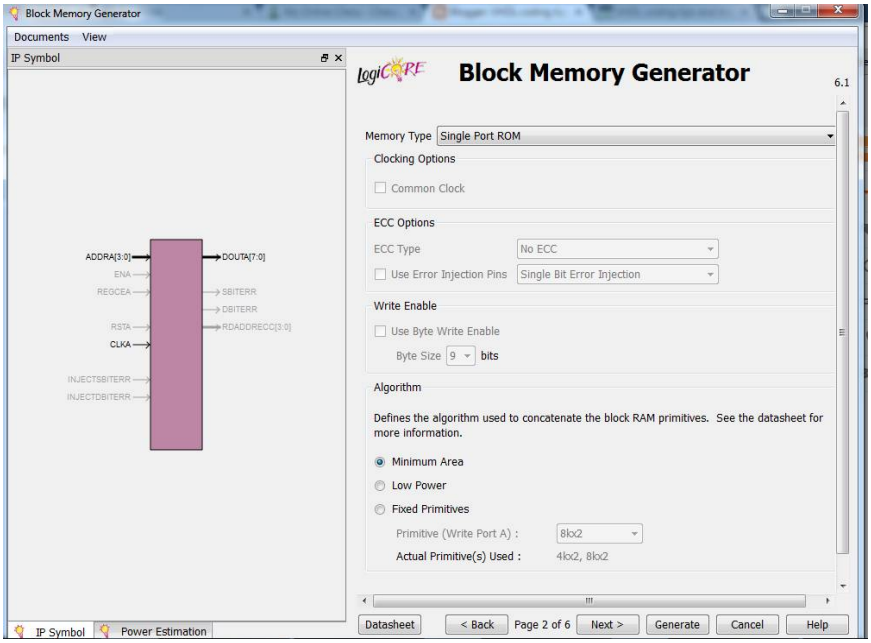


Fig 6. Block Memory Generator in Xilinx

VI .CONCLUSION

The implementation of Canny Edge Detection Algorithm on FPGA is described. Canny Edge Detector is designed using MATLAB. For writing the verilog code for Canny Edge Detection the image is first converted to text file using MATLAB. It can be given as the input for HDL program. Next step is to write the program for the Canny Edge detection in Verilog HDL. Canny Edge Detector is the Optimal Edge Detector and hence expecting better performance compared with other edge detectors . Since the FPGA method uses parallel processing the time required to detect the edges will be minimal when compared to conventional method.

REFERENCES

[1] R. Ponneela Vignesh, R. Rajendran, "*Performance and Analysis of Edge Detection Using FPGA Implementation*". International Journal of Modern Engineering Research (IJMER) Vol.2, Issue.2, Mar-Apr 2012 pp-552-554 ISSN: 2249-6645.
[2] Chandrashekar N.S, Dr. K.R. Nataraj, "*A Distributed Canny Edge Detection and Its Implementation on FPGA*" InternationalJournal of Computational Engineering Research (ijceronline.com) Vol. 2 Issue.7. Issn 2250-3005(online) November 2012.
[3] Tejaswini H.R, Vidhya N, Swathi R Varma,Santhosh B, "*An Implementation of Real Time Optimal Edge Detection and VLSI Architecture*". International conference on electronics and communication engineering, 28th april-2013, bengaluru, isbn: 978-93- 83060-04-7.
[4] Samina Jafar, Anupsingh Ramprakashsingh Rajput, "*Improved Distributed Canny Edge Detection In VHDL*". VSRD International Journal of Electrical, Electronics & Communication Engineering, Vol. 3 No. 6 June 2013.
[5] T. Rupalatha**,** Mr. C. Leelamohan, Mrs. M. Sreelakshmi, "*Implementation of Distributed Canny Edge Detection On FPGA*". International Journal of Innovative Research in Science, Engineering and Technology, Vol. 2, Issue7, July 2013.
[6] Divya. D, Sushma P. S, "*FPGA Implementation of a Distributed Canny Edge Detection*". International Journal of Advanced Computational Engineering and Networking, ISSN: 2320-2106 Volume- 1, Issue- 5.
[7] Mallavarapu. Ramu, T. V. S. Adinarayana, "*A Hardware/Software Co-Design Architecture Implementation of Canny Edge Detection Using FPGA and MATLAB*". International Journal of software& hardware research in engineering.
[8] Raman Maini, Dr. Himanshu Aggarwal "*Study and Comparison of Various Image Edge Detection Techniques*" IJIP, Volume (3): Issue (1).