# Design of Block Level Edge Detector using Spatial Filter

**Nayina Ramapur[1] Sujata Pujari[2] Dr.T C.Thanuja[3]**
[1,2]M.Tech. Student (VLSI Design and Embedded Systems) [3]Professor
[1,2,3]Department of Electronics & Communication Engineering
[1,2,3]VTU, Belgaum

*Abstract*— The image processing plays a significant role in many applications such as medical and defense applications. Consequently, there is more demand for the research in computer vision and image processing field. Canny Edge detection algorithm is the standard for many years. But still there are some limitations like more computation time and the area utilization. As the existing algorithm requires the storage for the overlapping block generated from the non-overlapping block, the area utilization is more and also accessing delay increases. Design of block level edge detector using spatial mask filter is proposed to overcome these effects. In this proposed design the edge detection is applied at the block level. The creation of overlapping block from the non- overlapping is performed with help of shifting operation. Thus the area and delay can be reduced by performing the shifting operation which generates the overlapping blocks from the non-overlapping block without any storage requirement.

***Key words:*** Edge Detection, Edge Detection Algorithm, Block Level, Reduced Area

## I. INTRODUCTION

Edge detection is one of the significant sections of the image processing algorithm, which has many applications like image morphing, pattern recognition, image segmentation and image extraction etc. As the edge is one of the major information contributors to any image, hence the edge detection is a very important step in many of the image processing algorithms. It represents the contour of the image which could be helpful to recognize the image as an object with its detected edges. In the ideal case, by applying the edge detector to an image gives the different edges that are connected to form the outline of the object. Important property of the edge detection is the detection of the exact edges along with the good orientation of the object in the image. And the memory required to store the edges of an image is less compared to the whole image even though it contains all information of the shape and orientation of the object.

The definition of Edge is differing as the authors varied. One of them is "It is change either in the brightness or the color of an image ". It is found that edges are basically denoted into four kinds in any image. They are step edge, ramp edge, roof edge and line edge. Many edge detection algorithms are proposed by many researchers and they are mainly classified into two types based on the order of derivative used [1]

- Gradient Based
- Laplacian Based

The Gradient based edge detection is also known as the first order derivative based because the gradient is calculated by differentiating an image. The Laplacian method computes the second order derivative of an image for the edge detection.

When the existing canny edge detector is applied to the block levels of the image it may detect the false edges or sometimes it may drop the true edges. In order to improve the performance, the block level Edge Detection Using Spatial Filter is applied to the block level. There are many edge detection algorithms were proposed by different authors [2].They are Prewitt, Sobel, Laplace, Canny, etc. The Canny Edge Detection Algorithm is the most effective algorithm among the existing edge detection algorithms. But the computation time required is more.

To defeat this, the algorithm is implemented on the different hardware devices like ASIC and FPGA. The implementation on FPGA is better than ASIC. In [3] the canny edge detection architecture and algorithm uses the language called Handle-C for the modeling purpose and the implementation of Canny Edge detection is made using FPGA Virtex-3. To have the proper operation, the Nios-II embedded processor is used for hardware implementation and the DSP builder acts as the mediator between the Altera Quartus II design software and MATLAB/Simulink tools [4]. Although these are good edge detection algorithms but there is a tradeoff between the performance and flexibility.

To improve the performance in terms of area and processing rate, parallel design of a real-time Canny [5] is introduced and it uses Spartan-3E for implementation. In the Spartan-3E the area is consuming of 28% with a processing rate of 240 frames per second for the image of 1Mpixel. For Vertex-5 the area taken is 6% with a processing rate of 580 frames per second [5]. In order to overcome the time consummation and manual setting of threshold [6] is introduced and it has real time performance cyclone. As the existing algorithm requires the storage of overlapping block generated from the non-overlapping block, the area utilization is more and also accessing delay increases. Thus the area and delay can be reduced by performing the shifting operation which generates the overlapping blocks from the non-overlapping block without any storage requirement [7].

In [7] the filter mask used for smoothing is the Gaussian mask and in the proposed design the spatial filter mask is replaced with the Gaussian mask.

The paper is organized as follows. Section 2 describes methodology and implementation of the edge detection algorithms. Also explains the proposed edge detector using spatial filter at the block level is proposed. Section 3 describes the obtained results and Section 4 finally concludes the paper.

## II. METHODOLOGY AND IMPLEMENTATION

For the Edge detection, image is given as input and the edge map is obtained as output. The project uses the Verilog coding for the design. But the Xilinx doesn't allow the image data as input, as the Xilinx can work on any number systems like decimal, hexadecimal, etc. So, the image has to be first converted to the any of the number system then it can be fed to the Xilinx as input. The conversion of image to

 

hexadecimal is done using the matlab. Image consists of a number of pixels; each pixel value indicates the intensity at that point.

The matlab can accept the gray images only for processing purpose. Thus first the given input RGB (Red Green Blue) image has to be converted to gray image and then image is resized, a text file is created. The pixel values are stored in that text files as equivalent hexadecimal number, in a single column for easy processing. The main idea behind the edge detection of an image is detecting the change of intensity of the pixel in an image; it is between 0 to 255. The dark portion of the image is treated as '0' and the bright portion is treated as '255'. First the obtained pixel values from the conversion of input image to the text is given as input to the shifter which will divide the obtained number of bits into 8-bit pixels and is fed to the horizontal filter that computes both horizontal gradient($G_x$) and vertical gradients($G_y$). Before the process of shifting first the smoothing operation is employed this will be done as follows:

In this design, the spatial filter mask m×n matrix of size w with the m= 2x+1 and n= 2y+1 where x, y are non-zero positive integers. For x=1 and y=1 the smallest possible mask size is 3×3. The coordinate arrangement of the mask coefficients is shown in the following figure fig. 1.

| | | |
|---|---|---|
| $w(-1,-1)$ | $w(-1,0)$ | $w(-1,1)$ |
| $w(-0,-1)$ | $w(0,0)$ | $w(0,1)$ |
| $w(1,-1)$ | $w(1,0)$ | $w(1,1)$ |

Fig. 1 Arrangement of Mask coefficients

Since mask used is of size 3×3 hence the image is also selected in the block of 3×3. To satisfy the criterion of the homogeneous pixel concept and to make differentiation of the homogenous pixels boundary, the pixels have to be classified first. In the proposed design first the color image is converted to gray image and then the window is applied. All the coefficients of the window are made as 1 except the central pixel. The central pixel is made as ×. The arrangement of window is shown in the following figure fig. 2.

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | × | 1 |
| 1 | 1 | 1 |

Fig. 2 Arrangement of the window

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 2-D spatial gradient uses the Sobel operator on an image to find the gradients in both the directions. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found.

The obtained $G_x$ and $G_y$ are given input to the shift registers which intern fed to the respective first in first out buffers. The obtained pixels are indicating the gradients in both directions and they are used to produce the Gradient magnitude and direction of that pixel. Then they are stored in the buffer and then the non-maximal suppression is made. By using the Adaptive Thresholding the pixels are compared to the different thresholds depending on the respective blocks. Finally the hysteresis thresholding is employed which will give the connectivity between the weak edges and the strong edges depending on the continuity. Thus the edge pixels are obtained. Then that are converted to the image in the matlab using Text to image conversion code.

### *A. Block Diagram and Flow chart Of the Block Level Edge Detector using Spatial Filter:*

The Block Level Edge Detector Using Spatial Filter and its flow chart are shown in fig. 3 and 4 respectively. In the block diagram it is shown that first the 8 bit pixel of the input image is given as input to the filter. This filter computes the Gradient values of the each pixel in the x- & y- directions ($G_x$ &$G_y$). Then both are given to the shifters which will generate the overlapping blocks then they are passed through the FIFO (First In First Out). Then the Gradient magnitude its direction (G & $\Theta_G$) are calculated. Further the Non maximal suppression, adaptive thresholding are employed to get the significant edges. Hysteresis is applied to remove the false weak edges and consider the weak edge connected to the strong edges.

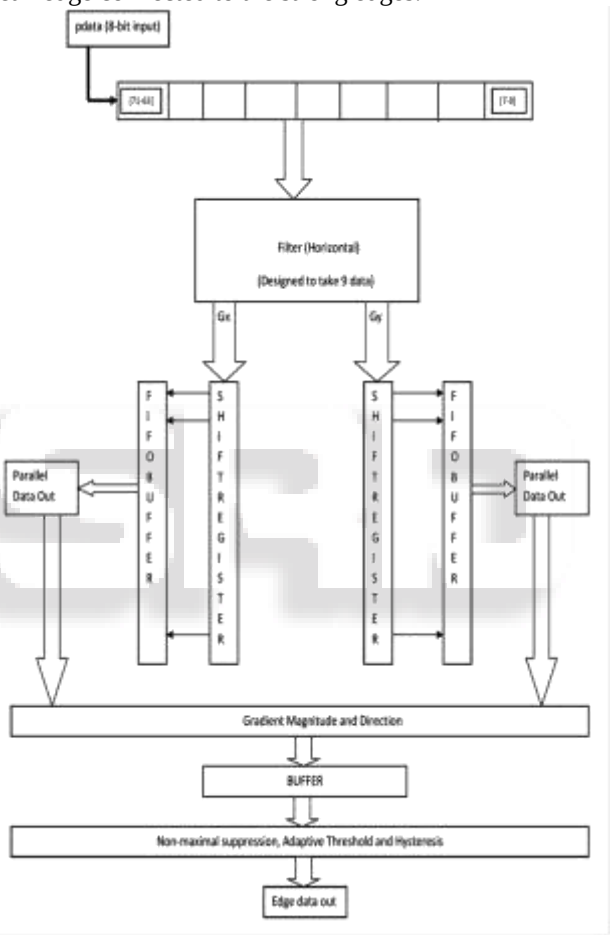


Fig.3 Block diagram of Block level Edge Detector Using Spatial Filter

The flow chart is shown in fig.4 which will provide the flow of the proposed design.

As mentioned in the flow chart, the processing of image in Xilinx platform is not possible directly, before processing colored image is converted to gray image which has the pixel value range between 0 and 255($2^8$-1). Then the pixel values of gray image are made to store in a single column in the text file. This text file is given as input to the Xilinx after all the processes mentioned in the above flow chart, the edge pixels are generated which has values like 0 for non-edge, 255 for strong edge and the in between values indicate the weak edges. Depending on the connection to the strong edge, the weak edge is either considered in the edge map else it is dropped. The detailed explanation of each step is described in the further part of the paper.

 

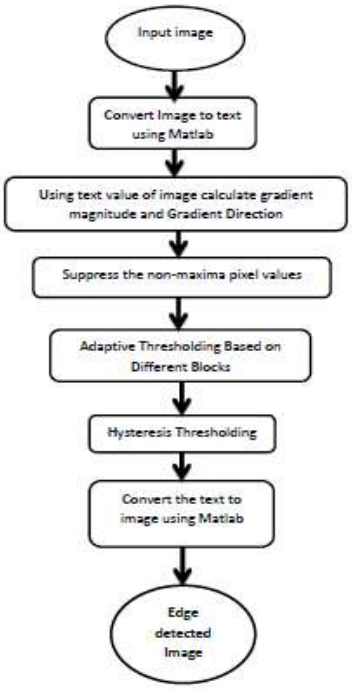


Fig. 4: Flow chart of the Block Level Edge Detector Using Spatial Filter

*B. Overlapping Block Creation:*

The conversion of overlapping block from the non-overlapping block is made [7] in as follows. In order to include all significant edges first the image is divided into n×n non overlapping blocks, and then they are extended in all the four directions by (l+1)/2 if the gradient mask size is lxl. The overlapping block is obtained from the non-overlapping block is in order to not to miss any of the true edges. An example of it is shown in fig. 5 with the Gradient mask of 3X3 implies the extension in all the four directions is (3+1)/2 =2.

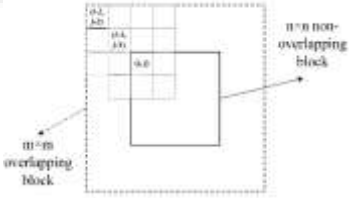


Fig. 5: Obtaining m×m overlapping block from the n×n non-overlapping Block with the Gradient mask 3×3

The overlapping block of the image obtained from non-overlapping block is made previously using the memory which increases the amount of memory required. Thus in this proposed block level Edge Detection Using Spatial Filter, the cache system is used with shift operation to get overlapped block from the non-overlapped block. Here there is no extra memory requirement as in the previous algorithm. The obtained overlapping block is directly given for the processing without storing it to any of the storage element. Thus the latency is reduced and throughput is increased. This shifting operation is made in window transformation.

*C. Horizontal and Vertical Gradient Computation:*

The text file data is considered for the Gradient calculation in both horizontal and vertical direction. To obtain the gradients in both directions the text data has to be convolved with the gradient operators in those directions. The gradient operator used is Sobel operator, as it gives better gradient values. The sobel operator is an isotropic 3×3 image Gradient operator. It is an image intensity function for gradient, which approximates the values of gradients.

The magnitude and direction of the gradients are calculated as:

$$\text{Magnitude(G)}=\sqrt{{G_x}^2+{G_y}^2};\ \text{Direction}(\theta_G)=\tan^{-1}\left(\frac{G_y}{G_x}\right)$$

The direction is measured in terms of the degree. And it is obtained by dividing $G_y$ by $G_x$ then applying the inverse of tangent operation to that.

$\theta_G = \text{invtan}(G_y/G_x)$.

*D. Non Maximal Suppression:*

In this step the non-local maximals are eliminated. To eliminate, first the neighboring pixel values has to be compared with the pixel under consideration. Once the direction ($\theta_G$) of the edges are known the next step is to link to the direction of the image. This can be done in this step. So if the pixels of a 5x5 image are aligned as in fig. 6.

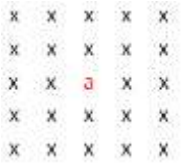

Fig. 6 Alignment of example of a 5×5 image

Just by observing the above figure that indicates the alignment of the image, the possibilities of the directions to the "a" is any of the four directions. They are:

- 0 degrees (in the horizontal direction)
- 45 degrees (along the positive diagonal)
- 90 degrees (in the vertical direction)
- 135 degrees (along the negative diagonal)

Approximation is made depending on the nearer direction, the edge direction is decided. The approximation is made such that if the obtained direction is 3° then it can be approximated to the 0°. In the similar way the approximation of the direction is made as in the following semicircle. The semicircle indicates the different colors of five sections which indicate the range of degrees to be approximated.

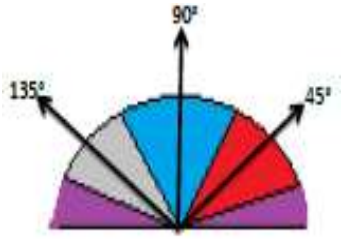


Fig. 7 Sections of approximating the degree of direction

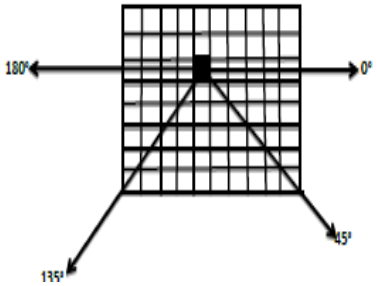


Fig. 8 Non Maximal Suppression mechanism

 

From the above figure the each color indicates a range of degrees are approximated to a single direction. That is the purple color specifies the range of 0 to 22.5 & 157.5 to 180 degrees is set to 0 degrees. Any edge direction coming in the red range 22.5 to 67.5 degrees is set to 45 degrees. Any edge direction specified in the blue range of 67.5 to 112.5 degrees is set to 90 degrees. And lastly, any edge direction falling within the gray range of 112.5 to 157.5 degrees is set to 135 degrees. All this approximation is done in the non-maximal suppression block. After the edge directions are known, non-maximum suppression now has to be applied. Non-maximum suppression and threshold 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.

The observation of neighboring pixel is made in the 0°, 45° and 135° are considered as shown in fig. 10; the black box represents the pixel under consideration. The neighboring pixels are compared in the all three directions mentioned before. Thus by comparing if the considered pixel is having higher value than its neighbors then it is treated as local maxima and is considered for edge map else it is eliminated from the edge map.

### *E. Adaptive Thresholding Computation*

In this step the image is divided into blocks as smooth, textured, hybrid and strong edge blocks based on the No. of Uniform pixels (Nu) and No. of Edge pixels (Ne). The process based on the following steps [8]:

1) If Nu >=0.3*(Total_Pixel) and Ne =0
   Smooth Block
2) If Nu<0.3*(Total_Pixel) and Ne=0
   Texture
3) If Nu<0.65(Total_Pixel-Ne) and Ne>0 &Ne<(0.3*Total_Pixel)
   Edge/Texture
4) If Nu>=0.65(Total_Pixel-Ne) and Ne>0&Ne<0.3*Total_Pixel)
   Medium Edge
5) If Nu<=0.7*(Total_Pixel) and Ne>=0.3*(Total_Pixel)
   Strong Edge

Where the $N_u$ indicates the number of uniform pixels and Ne represents the total number of edge pixels in the block. Depending on the values of the Nu and Ne the pixels are indicated to which block it belongs. In this design an adaptive thresholding block is used which will provide different threshold for different blocks.

Considering P1 as the percentage of pixels in a block

1) Step1:If Smooth Block
   P1=0; i.e No Edges
   else if Texture Block
   P1=0.03; i.e Few Edges
   else if Medium Edge Block
   P1=0.2; i.e Medium Edges
   else
   P1=0.4; i.e Many Edges
2) Step2:Calculating the 8-bin non uniform gradient magnitude histogram and corresponding function is F(G)
3) Step3:Calculate High_threshold as F(High_threshold)=(1-P1)
4) Step4:Calculating Low_threshold =0.4*High_threshold

As mentioned in the above steps, the blocks are classified, and the blocks will represent unique status of the edges. If the block is smooth then there were no edges and the threshold is set to zero.

### *F. Hysteresis Thresholding:*

Finally to remove the streaking effect, hysteresis thresholding is employed. The streaking effect is arises due to the shifting of the output of the operator above and below the threshold. This method is used to have the continuity of the contour of the image. The discontinuity in the outline of the image is caused due to the varying values of the pixels around the threshold value. If only one threshold, T1 is applied to an image, and if an edge has an average strength that is same as T1, then due to noise, there will be cases where the edge drops below the threshold or the edge may be extended above the threshold.

To avoid this effect two thresholds high (T1) and low (T2) are employed in the hysteresis. If the edge is higher than the high threshold (T1) treated as the strong edge. If edge is lower than the low threshold (T2) then it is defined as non-edge. And if the edge value is in between the T1 and T2 threshold then that will be treated as weak edge. The process is continued as the edge pixel value is greater than the T2 and it will continue until the all pixels having the value less than the T1. Depending on the relation between the weak edge and the neighboring edges, either the weak edge is considered in the edge map else it will be treated as non-edge and eliminated from the edge map. This block will connect the weak edges with the strong edge. There are many weak edges were detected but all the weak edges are not true edges. The decision of considering the weak edge in the edge map only if that weak edge is connected to strong edge, else the weak edge is dropped from the edge map.

## III. SIMULATION RESULTS

The performance of the Block level edge detector using spatial filter algorithm is evaluated by simulating a different resolution of image such as 128×128, 256×256 using Matlab and Xilinx.

The thresholding is based on the type of the block. For example, if the considered image block has strong edge, then the threshold is set as 102. If the considered pixel is greater than this threshold, it is treated as strong edge and made as 255 else it will be treated as weak edge. Similarly for the blocks having medium edges and few edges the threshold will be set as 7.65 and 25 respectively. The above explained procedure is repeated to make the decision of edge or non-edge of the considered pixel. Since the smooth block does not have edges the threshold will be set either 2.5 or 0.

Subjective Performance evaluation of the algorithm is considered. Image is Read through the matlab and the color image is converted to binary and then to text. Fig 9 (a) shows the output of color image of size 128×128. Fig. 9(b) shows the gray converted image and Fig. 9(c) shows the text

 

file generated corresponding to the input image. Text file is read through the Xilinx software.


Fig. 9: Matlab input image & output text result

The Simulation results of the edge detector are shown in the fig. 10. In that the input pixel data is shown which contains the various pixel intensity of the image. The pixel intensity value is shown the binary form that varies from 0 to 255.That is in the binary 00000000 to 11111111. The clock and synchronization frequency is applied. The edge detected output is having the values in the same range but the non-edges are indicated with the all zero that is "00000000". The strong edges are indicated with all ones that is "11111111".

Fig. 10: Simulation of Edge detector

After getting the simulation results the generated text file is stored in the output text file mentioned in the test bench. Then to get edge mapped output the matlab code that converts text to image is used.

The edge detected image obtained in the matlab is shown in fig. 11

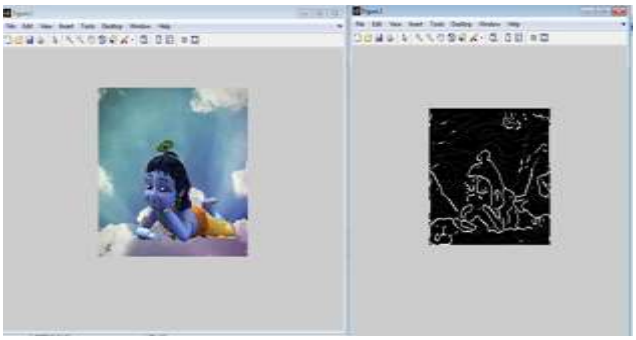

Fig. 11: Output Obtained in Matlab

For different images namely bird and flower of size 128×128, the canny edge detection and block level detection using spatial filter is applied. The images of the edge detection output are shown in the figures 12 and 13.

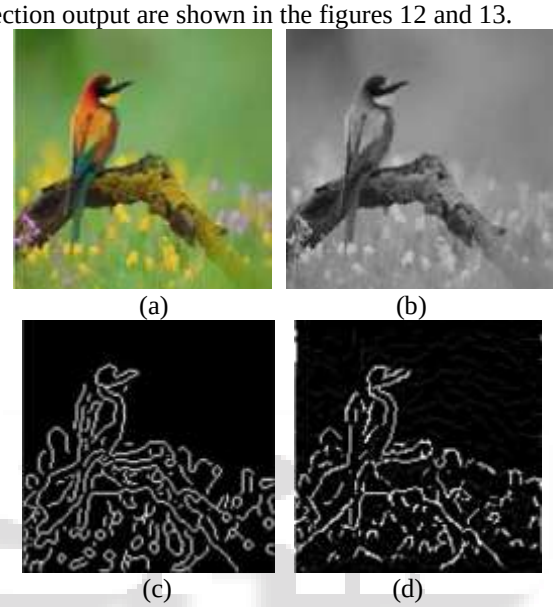


Fig. 12 Bird (a) Color image (b) Gray image (c) Canny Edge detected output (d) Block level Edge detected image Using Spatial Filter

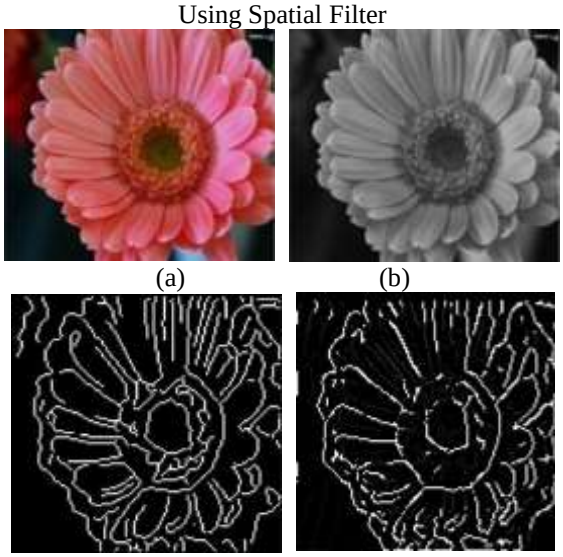


Fig. 13 Flower (a) Color image (b) Gray image (c) Canny Edge detected output (d) Block level Edge Detected image using spatial filter

The images considered for simulation to evaluate the performance are chosen based on the variation in the number of edges. Fig. 14, 15 & 16 shows the images with the more, medium and less edges of resolution 256×256.

 

Each figure contains the input image and the corresponding edge detected image.


Fig. 14 Input image and block level edge detected using spatial filter image of picture

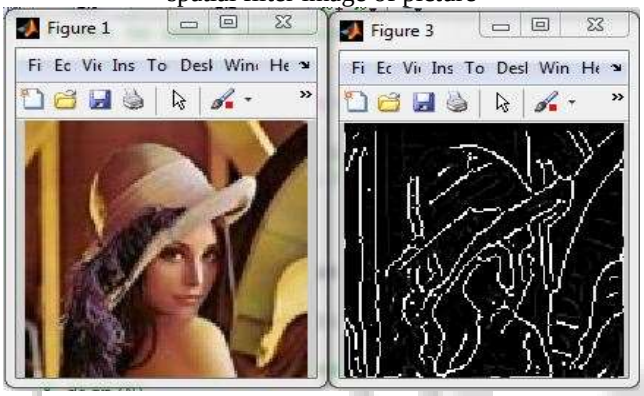


Fig. 15 Input image and block level edge detected using spatial filter image of lena

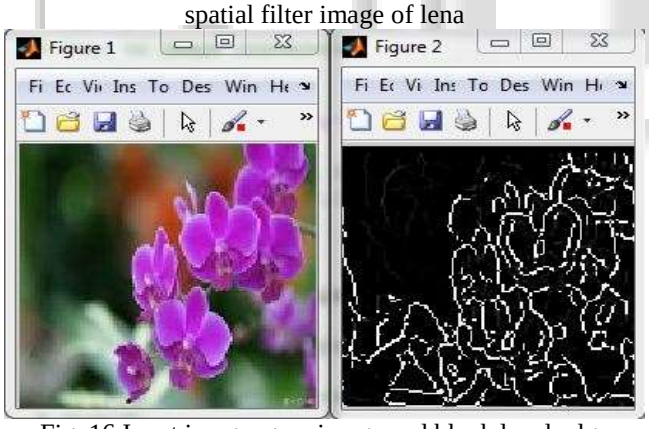


Fig. 16 Input image, gray image and block level edge detected using spatial filter image of spring flower

The Table 1 shows the number of slices utilized out of available for the Canny Edge Detector and the Block level Edge detector Using Spatial Filter.

| | Canny Edge Detector[4] | Proposed Block Level Edge Detector Using Spatial filter |
|---|---|---|
| Total Number of Slices Available | 27,288 | 27,288 |
| Number of Slices Utilized | 4560 | 3123 |
| Delay | 4.9652ns | 4.7097ns |

Table 1 Area Occupation and Operating Frequency comparison On Spartan-6 FPGA

From the above table the area utilization of the Canny Edge detector and Proposed Block Level Edge Detector Using Spatial Filter is mentioned in terms of number of Slice used. It is observed that the proposed design uses less area compared to the canny edge detector. That is the area is reduced by 31.51%. The frequency of operation is also more than the previous implementation, thus the speed is increased compared to the canny edge detector. And also the delay is reduced by 0.2ns.

## IV. Conclusion

The paper describes the Canny Edge Detection Algorithm that used to detect the edge of any image as a complete image without dividing it into blocks. The proposed Block level Edge detector using spatial filter has overcome the limitation of existing edge detection algorithms by reducing the delay and area. The design of Block Level Edge Detector Using Spatial Filter is coded in Verilog HDL language. The simulation and synthesis of the design is carried out using Xilinx ISE 14.5 tool. The delay and area of the proposed design is reduced by 31.51%. The delay is reduced by 0.2ns.

In the future for the different image processing applications the proposed method can be used to improve the performance.

## References

[1] James Clerk Maxwell, 1868 Digital Image Processing Mathematical and Computational Methods.
[2] Raman Maini, Dr. Himanshu Aggarwal "Study and Comparison of Various Image Edge Detection Techniques", International Journal of Image Processing (IJIP), Volume (3)
[3] D. V. Rao and M. Venkatesan, "An efficient reconfigurable architecture and implementation of edge detection algorithm using handle-C," in Proc. IEEE Conf. ITCC, vol. 2. Apr. 2004, pp. 843–847.
[4] H. Neoh and A. Hazanchuck, "Adaptive edge detection for real-time video processing using FPGAs," Altera Corp., San Jose, CA, USA, Application Note, 2005.
[5] C. Gentsos, C. Sotiropoulou, S. Nikolaidis, and N. Vassiliadis, "Real time canny edge detection parallel implementation for FPGAs," in Proc. IEEE ICECS, Dec. 2010, pp. 499–502.
[6] W. He and K. Yuan, "An improved canny edge detector and its realization on FPGA," in Proc. IEEE 7th WCICA, Jun. 2008, pp. 6561–6564.
[7] Q. Xu, C. Chakrabarti, and L. J. Karam, "A distributed Canny edge detector and its implementation on FPGA," in Proc. DSP/SPE), Jan. 2011, pp. 500–505.
[8] W. He and K. Yuan, "An improved canny edge detector and its realization on FPGA," in Proc. IEEE 7th WCICA, Jun. 2008, pp. 6561–6564.

 