Implementation of SOBEL, PREWITT, ROBERTS Edge Detection on FPGA

F. Alim.Ferhat, L. Ait mohamed, O. Kerdjidj, K. Messaoudi, A. Boudjelal, S. Seddiki
Country Centre for Development of Advanced Technologies, Algiers, Algeria
{falim , laitmohamed, okerdjidj, kmessaoudi, aboudjelal, sseddiki}@CDTA.dz

Abstract - *The proposed work presents FPGA based architecture for Edge Detection using different operators of gradient: Sobel, Roberts, Prewitts. These different operators were chosen for its resistance to noise. Gray image is converted into file *.Coe format using MATLAB. The proposed gradient Edge Detection is modeled using Parallel Architecture and implemented in VHDL, then the result is reconverted by Matlab. This work is a first step in our chain of segmentation.*

Keywords: Gradient Operator, Edge Detection, Digital Image Processing, FPGA.

1 Introduction

FPGA has become an alternative for the implementation of algorithms that unique structure permitted the technology used in many applications, video surveillance and medical imaging. FPGA is an integrated circuit with a large scale that can be reprogrammed. The term "field programmable" refers to the ability to change the functioning of the device. Edge detection is one of the most in the processing of low-level image; the quality of detected contours has a very important role in the realization of complex automated computer vision/machine [1]. Many algorithms of edge detection are available in the literature and give different detection result on the same image input. Edge detection by Sobel operator is very used compared to the simple gradient operator because of its resistance to noise and facilitated its implementation. [2].

1.1 Diagram of the system of edge detection

The proposed work is represented as follows:

- Converting the image into a vector in a text file extension *.Coe using the tool MATLAB.
- Loading the file *. Coe in Rom FPGA.
- Detecting image contours with different operators of gradient Sobel, Robert, Prewitt in VHDL.
- Conversion by MATLAB of text file generated by the simulation tool ISE.

The synoptic is shown in Figure.1.

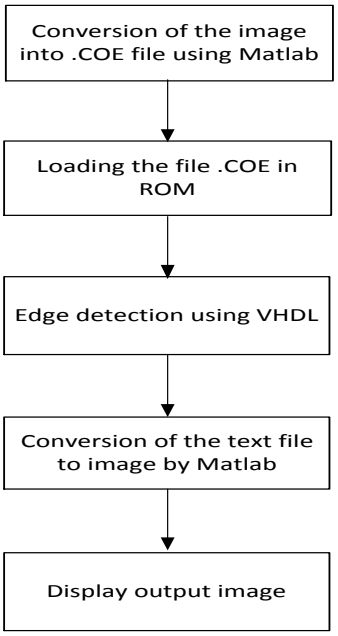


Fig 1: Synoptic of Edge detection system

2 EDGE DETECTION

Edge detection is a preliminary step in many applications of image analysis. The contours are indeed rich indices, as well as points of interest for any subsequent interpretation of the image. Contours in an image derived from:

- Discontinuities of reflectance function (texture, shadow)
- Depth discontinuities (object edges), the latter are characterized by discontinuities of the intensity function in the images.

The principle of edge detection is thus based on the study of the derivative of the intensity function from the image: the local extrema of the function gradient intensity and the zero crossings of the Laplacian. The difficulty resides in the presence of noise in the images.

2.1 Detection by gradient

This approach is based on research of image points with high gradient which corresponds to the points where the gradient magnitude is maximal. The gradient of an image I(x,y)is defined as the vector [3]

$$\vec{\nabla} I(x,y) = \frac{\partial I(x,y)}{\partial x}\vec{I}_x + \frac{\partial I(x,y)}{\partial y}\vec{I}_y. \quad (1)$$

where: $\vec{I}_x$, $\vec{I}_y$ are a unit vector along X and Y and $\vec{\nabla}$ is the gradient is a vector which can be calculated in different ways.

In the discrete case, each component, calculated using a convolution gives the value of the gradient in any direction as shown in the following relationship:

$$\nabla_x = \frac{\partial I(x,y)}{\partial x} = I(x,y) * h_1(x,y).$$
$$\nabla_y = \frac{\partial I(x,y)}{\partial y} = I(x,y) * h_2(x,y). \quad (2)$$

These operators can be applied to digital images "discrete case", the directional derivatives according to the horizontal and vertical directions to the site [i, j] are approximated by a simple finite differences:

$$\frac{\partial I}{\partial y} \approx \frac{\Delta I}{\Delta i} = I_i[i,j] = I[i+1,j] - I[i,j] \quad (03)$$

$$\frac{\partial I}{\partial x} \approx \frac{\Delta I}{\Delta j} = I_j[i,j] = I[i,j+1] - I[i,j] \quad (04)$$

The gradient magnitude is given by:

$$\left|\nabla I[i,j]\right| = \sqrt{I_j^2[i,j] + I_i^2[i,j]} \quad (05)$$

$$\left|\nabla I[i,j]\right| \approx \max\left\{\left|I_j[i,j]\right|, \left|I_i[i,j]\right|\right\} \quad (06)$$

$$\left|\nabla I[i,j]\right| = \frac{\left|I_j[i,j]\right| + \left|I_i[i,j]\right|}{2} \quad (07)$$

The direction angle of the vector $\vec{\nabla} I$ at (x,y) is

$$\theta = Arc\tan\left(\frac{\nabla_y}{\nabla_x}\right) \quad (08)$$

h1 is the convolution mask along x, and h2 the convolution mask along y.

There are various masks [4] of convolution, we give:

- Sobel operators:

$$h_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- Prewitt operators :

$$h_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

- Roberts operators:

$$h_1 = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

3 Proposed Architecture of FPGA Implementation for Edge Detection

We used in this architecture a 3×3 convolution karnels processing 256 × 256 gray scale image.

The architecture is shown in Fig. 2.The system is divided into four modules: 3×3 pixel address generation, Memory ROM, convolution 3×3 structure and gradient operators. We have in this architecture 3 inputs and three outputs, we find a clock signal Clk, a reset signal and Sel_op signal for the selection of the gradients operators, Dout1, Dout2, Dout3 are the results of edge detection with different operators (Sobel, Robert, Prewitt].

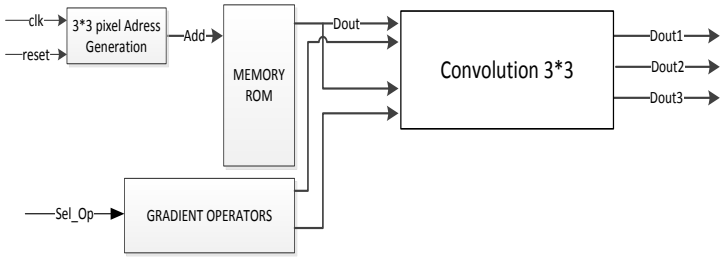


Fig 2: Architecture

The function and structure of each module are as follows:

- The structure of 3×3 pixel address generation module is shown in Fig.3. This module allows us to generate destine 9 addresses to the ROM for the operation of convolution Kernels [4] [5].

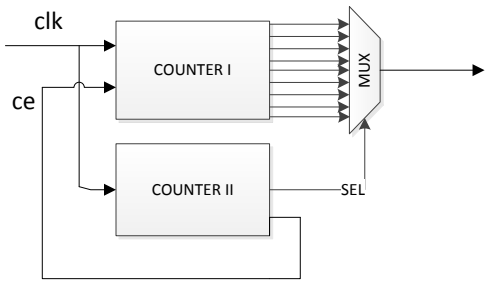


Fig 3: 3×3 Pixel Address Generation Module

- The structure of Memory ROM is shown in Fig.4: This module allows us to store the file *.Coe of image converted by Matlab.

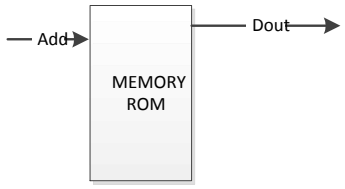


Fig 4: Memory ROM

- The structure of gradients operators is shown in Fig.5: This module permits to choose the coefficients of the different horizontal and vertical operators (Sobel, Prewitt, Roberts).

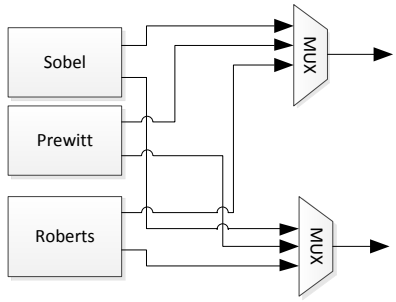


Fig 5: Gradient Operators

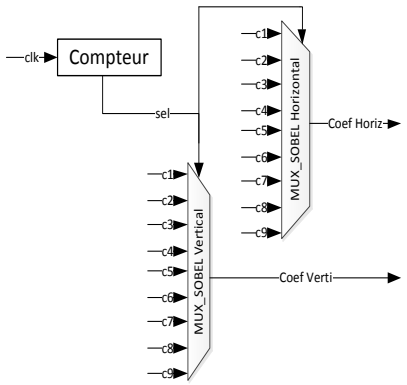


Fig 6: Unit generating the coefficients of the Sobel operator

- The structure of 3×3 convolution gradient is shown in Fig.6: This module done the convolution of pixels read by the ROM with the coefficients horizontal and vertical.

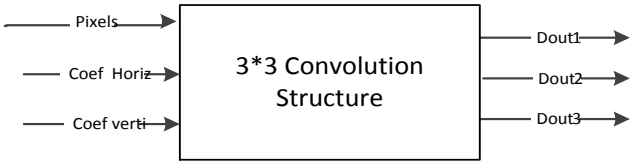
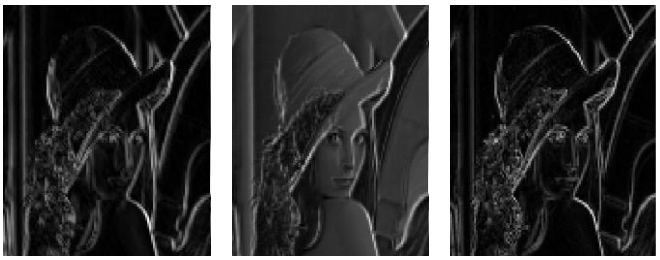


Fig 5: Convolution 3×3 Structure.

4 Experimental Results

All results for image edge detection in Matlab and its comparison in VHDL are represented in this section:

(a)



(b) (c) (d)

(e) (f) (g)

Fig 6: (a) original image, (b) Sobel Gradient using VHDL, (c) Prewitt Gradient using VHDL, (d) Roberts Gradient using VHDL, (e) Sobel Gradient Software, (f) Prewitt Gradient Software, (g) Roberts Gradient Software.

5 Synthesis

Synthesis consists in transforming the algorithms studied for a logic functions. It is done by the tool XST (Xilinx Synthesis Tool) integrated into ISE. We obtain a report that contains the occupancy rate (number of CLBs, SLICE, ...).

We implemented the proposed architecture on the circuit Virtex 4 device XC4VLX200 package FF1513. Resources used in the FPGA are given in Table 1.

TABLE I. RESOURCES OF PROPOSED ARCHITECTURE

Number of resources	Occupancy Rate
Number of Slices: 8653 out of 89088	9%
Number of Slice Flip Flops: 109 out of 178176	0%
Number of 4 input LUT: 17208 out of 178176	9%
Number of bonded IOBs: 61 out of 964	6%
Number of GCLKs: 1 out of 32	3%

6 Conclusion

The obtained results are simulated using ISE simulator. VHDL cannot use the standard image formats for that we converted her into binary file *.coe format using MATLAB. The binary file is applied as a vector, it is converted and displayed in MATLAB. The execution time of the entire program to edges detect of an image 256×256 is a few seconds. No operator is perfect for detecting contours. In practice, we obtain incomplete contours, there are pixels unnecessary gaps, errors in position and orientation of pixels contours. Everyone seems to have a preference for one method or another all depends on the application. Operator edge detection is only the first step in the chain of segmentation.

In order to improve speed and efficiency of edge detection a pre-filtering of the images is required. Linear filtering for noise with zero mean (Gaussian white noise, Gaussian filter). Non-linear filtering for impulse noise (median filter for example). Possible segmentation by watershed will give a better result .To improve the speed and efficiency pipeline and reconfigurable architecture can be further done in edge detection, also segmentation can be implemented on hardware using watershed or contour active methods .

7 Reference

[1] Raman Maini, Dr. Himanshu Aggarwal, “Study and Comparison of Various Image Edge Detection Techniques”, International Journal of Image Processing (IJIP), Volume (3)

[2] Tian Qiu, Yong Yan* FIEEE, Gang Lu SMIEEE, 2011. “A New Edge Detection Algorithm for Flame Image Processing”, IEEE.

[3] Maitine Bergounioux 2010 Quelques Méthodes de Filtrage en Traitement d'image

[4] Steve Kilts, Advanced FPGA Design: Architecture Implementation, and Optimization, John Tiley&Sons.

[5] Arrigo Benedetti, Andrea Prati, Nello Scarabottolo. “Image convolution on FPGAs: the implementation of a multi-FPGA FIFO structure”. Euromicro Conference, 1998.