Implementation of Edge Detection Digital Image Algorithm on a FPGA

Issam Bouganssa, Mohamed Sbihi and Mounia Zaim

Laboratory of System Analysis, Information Processing and Integrated Management, High School of Technology SALE, Mohammed V University in Rabat, Morocco

Abstract. This paper presents the implementation of an adaptive contour detection filter on field programmable gate array (FPGA) using a combination of hardware and software components. The proposed system locates contours from a calculation of Gradient in preferred directions while quantifying the importance of the outline with a wise thresholding. The dedicated algorithm is implemented in an FPGA Xilinx Spartan 6, and results are displayed in a VGA monitor. The FPGA offers the necessary performance for image processing and video in real time, while maintaining the flexibility of the system to support an adaptive algorithm. Simulation results and synthesis proposed edge detection processor on FPGA chip demonstrate the efficiency of proposed architecture.

1 Introduction

The goal of edge detection is to produce something like a line drawing of an image. In practice we will look for places in the image where the intensity changes quickly. Observe that, in general, the boundaries of objects tend to produce sudden changes in the image intensity.

For example, different objects are usually different colors or hues and this causes the image intensity to change as we move from one object to another. In addition, different surfaces of an object receive different amounts of light, which again produces intensity changes. Thus, much of the geometric information that would be conveyed in a line drawing is captured by the intensity changes in an image. Unfortunately, there are also a large number of intensity changes that are not due to geometry, such as surface markings, texture, and specular reflections. Moreover there are sometimes surface boundaries that do not produce very strong intensity changes. Therefore the intensity boundary information that we extract from an image will tend to indicate object boundaries, but not always.

The detected edges are materialized by the intensity breakdown in the image in a given direction. Several methods exist to detect the break. Some of them are more or less complex. Others are more or less intensive computing.

In this work, we use the gradient measurement method for images followed by a judicious thresholding which allow to isolate the edge of the rest of the image. The algorithm is implemented in a FPGA device which is providing good performance of integrated circuit platform for research and development. Indeed, the FPGA technology has become an alternative for the implementation of software algorithms.

The results of edge detection algorithm implemented in FPGA are displayed on a VGA screen continuously. The image will be stored in a block memory in FPGA. Then, the algorithm reads the image of memory and stores the processed image back into memory. VGA controller designed reads the processed image from memory and displaying it.

We choose in our work the Xilinx Spartan6 circuit which offers the performance required in the real-time processing of image sequences, while retaining the flexibility of the system to support an adaptive algorithm.

2 Contour definition

In the image, a contour can be considered in different ways. Here we describe three main ways to consider a contour:

First, a contour can be seen as an abrupt change in the intensity of the image particularly for images in gray level [1], there are several types of variations (Figure 1).

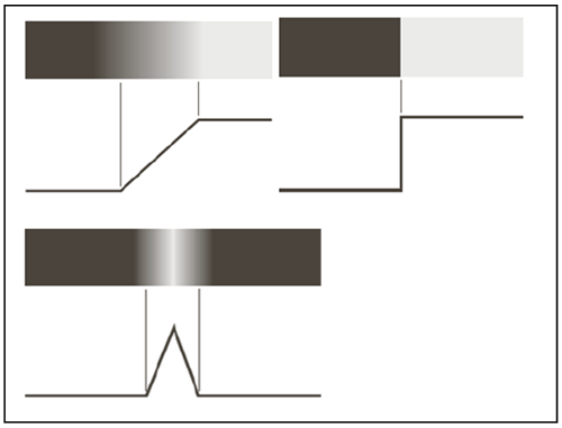


Figure 1. Contour profiles, ramp, doorstep, roof.

Secondly, a way very similar to that mentioned above, to consider the contour as a difference on color.

Thirdly, if we consider the image as a 2D signal, we can go in the frequency domain (Fourier transform or wavelet for example) [2]. In this case, a contour can be represented as the high frequency signal.

3 Edge detection principle with gradient

A local variation of intensity is a primary source of information in image processing. It's measured by ***the gradient*** [3,4] vector function of the pixel [i, j].

3.1 The gradient in a discrete image

In an orthogonal coordinate system (Oxy) where (Ox) is the horizontal axis and (Oy) the vertical axis, the image ***gradient*** (or rather the ***luminance f***) at any point or pixel coordinates (x, y) is denoted by:

$$Grad\, f = \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} \quad (1)$$

The ***module of the gradient*** quantifies the importance of the contour highlighted, that is to say the magnitude of the jump intensity observed in the image:

$$\|\nabla f\| = \sqrt{(\frac{\partial f^2}{\partial x}) + (\frac{\partial f^2}{\partial y})} \quad (2)$$

The ***direction α_o of the gradient*** determines the present edge in the image. Indeed, the gradient direction is orthogonal to that of the outline:

$$\alpha_o = \arctan(\frac{\partial f/\partial y}{\partial f/\partial x}) \quad (3)$$

The principle of ***edge detection by the use of the gradient*** is to calculate,in the first time, the gradient of the image in two orthogonal directions, then the gradient module. The next step is to make a selection of the most marked contours, that is to say the points of stronger contrast with adequate thresholding (see below).

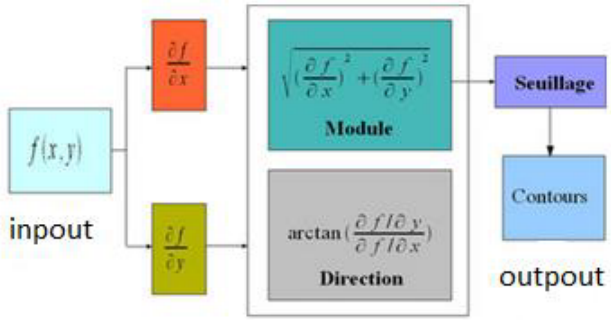
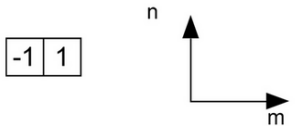


Figure 2. principle of edge detection.

3.2 Mask with two elements



The correlation of the mask with a luminance image [5], f (i, j) can be written:

$$\sum_{m=0}^{1}\sum_{n=0} h(m,n)\, f(m+i,n+j)$$
$$= h(0,0\, f(i,j) + h(1,0)\, f(i+1,j)$$
$$= f(i+1,j) - f(i,j) \quad (4)$$

3.3 Mask with three elements

The gradient calculation is achieved by means of two masks, the first performing a horizontal gradient, the second vertical gradient. The second mask is derived from the first by a rotation of $\frac{\pi}{2}$.

$$[\text{-1 0 1}] \text{ and} [-1\ 0\ 1]^{\mathrm{T}}$$

Compared to previous, [6], [7] this mask has the advantage of producing two effects. To calculate the gradient in one direction, and performs a smoothing in the orthogonal direction. This smoothing makes it a little less sensitive to noise than the previous mask. The origin of this mask is always the center pixel. The value of the constant C can take 1 or 2 to increase the effect of smoothing.

The output pixel obtained after filtering is:

$$\sum_{m=-1}^{1}\sum_{n=-1} h(m,n)\, f(m+i,n+j)$$
$$= \frac{1}{c+2}\{h(-1,1)\, f(i-1,j+1) + h(0,1)\, f(i,j+1)\}$$
$$+ \frac{1}{c+2}\{h(1,1)\, f(i+1,j+1) + h(-1,-1)\, f(i-1,j-1)\}$$
$$+ \frac{1}{c+2}\{h(-1,0)\, f(i-1,j) + h(-1,1)\, f(i-1,j+1)\}$$
$$\sum_{m=-1}^{1}\sum_{n=-1} h(m,n)\, f(m+i,n+j)$$
$$= \frac{1}{c+2}\{f(i-1,j+1) + c\, f(i,j+1)\, f(i+1,j+1)\}$$
$$-\frac{1}{c+2}\{f(i-1,j-1) + c\, f(i-1,j)\, f(i-1,j+1)\} \quad (5)$$

The equation reveals the double action with a horizontal middle ages of three pixels on the lines above and below the central pixel and calculate the vertical gradient between the two lines.

For the detection of vertical edges, other mask is used. The output pixel obtained after filtering can be put in the form:

$$\sum_{m=-1}^{1}\sum_{n=-1} h(m,n)\, f(m+i,n+j) = \frac{1}{c+2}\{f(i+1,j+1) - f(i-1,j+1)\} + \frac{1}{c+2}\{c\, f(i+1,j) - c\, f(i-1,j)\} +$$
$$\frac{1}{c+2}\{f(i+1,j-1) - f(i-1,j-1)\} \quad (6)$$

This equation shows the dual action: calculate on three lines the horizontal gradient then applied a vertical smoothing.

Note:

The main interest of these masks is the ease of implementation and the speed of processing. Their disadvantage is their sensitivity to noise. However the detected edges are often quite large. According to the filters of three elements are the most used in industrial applications requiring real-time constraints.

4 Requirements image processing material

Treatment of live image requires high computing power. For example, the standard image.jpg with a dimension of 640*480 approximately 0.25 mega pixels per image, with a calculated power of 100 frames per second for a 25Mhz processor. The size of the image can be greater [8,9], the amount of processing required per pixel depends on the image processing algorithm.

The edge detection algorithm by the concentration gradient is a combination of four steps, the horizontal gradient, the vertical gradient, their modules and finally a suitable thresholding.

Depending on the size of the masks, the processing requirements for the first two convolution steps may to be change. Using the assumption of a mask size of 2*1 or 3*1 for the partial derivative, both steps require more operations per pixel. This implementation requires less complexity than most typical 2-D convolution process because of the symmetry characteristics and separable Gaussian masks. The steps of the calculation module and thresholding requires fewer operations per pixel.

Such as high resolution images become more frequent, the processing requirements will increase. The high resolution images of the standards generally have ten times more pixels per image. The computational load is approximately ten times higher. They require more DSP or a single, very expensive high-end DSP. In this scenario, FPGAs offer a real-time alternative platform image processing. FPGA effectively supports high levels of parallel processing data flow structures (Figure 4), which are important for the effective implementation of image processing algorithms.

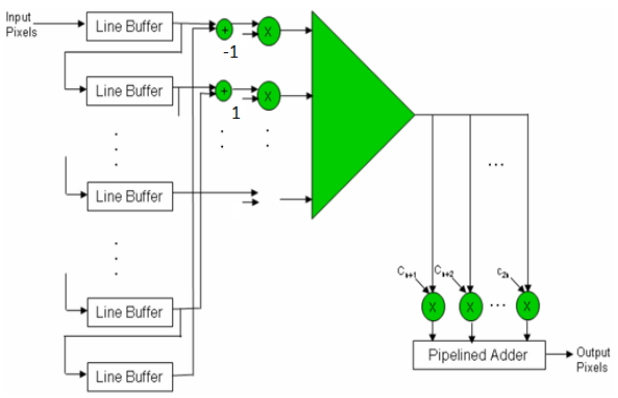
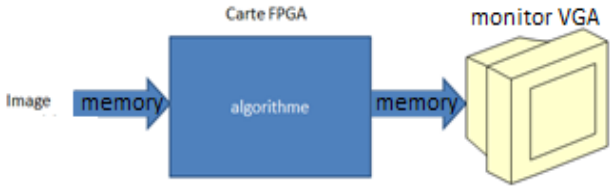


Figure 3. processing of data streams in parallel.

5 Hardware implementation on FPGA of the Xilinx family

For the implementation of our algorithm, we used the platform of Xilinx nexys3 Spartan-6 FPGA and VGA monitor to display results.



The algorithm was developed on Xilinx ISE interface and all the blocks are programmed in VHDL with the following block diagram:

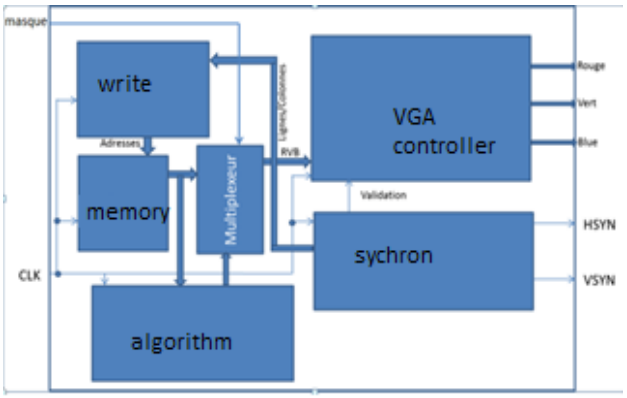


Figure 4. block diagram of programs.

Our project consists of several blocks, which are other than VHDL programs:

5.1 Block program "Memory"

The ***"Memory"*** block that exists in the party IP (intellectual property) to the Xilinx software is configured according to the size of the image that will store, each memory cell contains 8 bits (8 bits is the value which is coded each RGB pixel).

5.2 Block program "Memory Playback"

Since the image is not necessarily the same size as the screen, the program is necessary for the correct positioning of the image on the screen.

5.3 Block program "VGA Screen"

The program's role to define. The VGA monitor is controlled by five 10-bit coded signals: red (3 bits), green (3 bits), blue (2 bits), horizontal sync and vertical sync 1 bit 1 bit. The three color signals, collectively known as the RGB signal are used to control the color of a pixel at a location on the screen. In order to produce other colors, each color analog signal is to be supplied with a voltage between 0.7 and 1.0 volts for varying the intensities of the colors.

5.4 Block program "Synchronization"

The program lets you synchronize the scanning of pixels on the screen of a horizontally and vertically. Signals are used to control the synchronization of the scanning speed. The horizontal synchronizing signal determines the time to scan one line, while the vertical synchronizing signal determines the time to scan the entire screen. By manipulating these signals, the images are formed on the monitor screen.

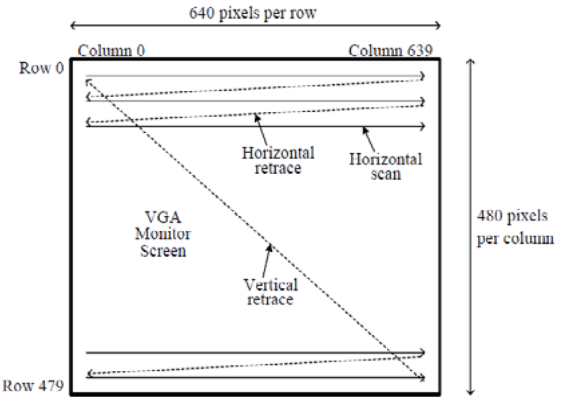


Figure 5. Operating principle of VGA controller.

5.5 Block program "algorithm"

It's the same principle previously defined in Part 3 for edge detection. The pixels after the conversion are always less than the total number of pixels. The upper limit guarantees that the operations depending on the FIFO list can be completed during the real-time processing period of a frame. FIFOs are implemented using multiple integrated M4K memory blocks, which are a size of 4 kbps each. The image of the edge detection result is stored in the blocks of RAM built in the Spartan-6 device. These large blocks of integrated RAM can be used to store temporarily the edge image when 2 bits are allocated to each pixel location.

Following diagram shows the different blocks implemented on the Xilinx ISE software:

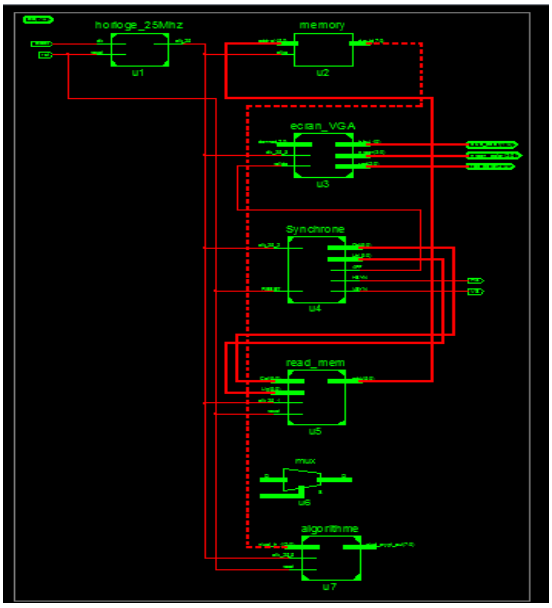


Figure 6. The blocks implemented on Xilinx ISE.

Preliminary results show the FPGA design clocked at 100MHz Spartan-6 targeting (Figure 7).

Figure 7. The image results before treatment, image after treatment.

We used a test image with a resolution of 200x200. The module is fully pipelined where a resulting pixel is calculated at each clock cycle [10]. With this rate and the clock frequency, it is possible to treat more than 400 image frames at 200x200 resolutions.

The design is scalable to handle high-resolution images while maintaining the clock frequency used to treat standard resolution video at 30 frames per second, as well as computer vision applications at high speeds, which require more 80 fps.

It is possible to increase the frequency above 100 MHz by introducing more pipeline stages at the expense of increased resource utilization [11]. But the uses of high resolution images require a more powerful processor that exceeds the range Spartan, in talking about the Virtex range, high performance at frequencies exceeding 500MHz.

6 Conclusion

In this paper, we propose implementation of edge detection algorithms on a FPGA. The presented method is based on detection of the maximum gradient of the horizontal and vertical image, on the module of them and on the thresholding to select the strongest contours.

The results obtained show is the clear presence of all the contours that exist in the images (horizontal and

vertical contours, circular forms and writings), the use of an FPGA-circuit spartan6 show its ability to manage images high resolution VGA output, while maintaining the clock frequency used (100Mhz) for processing image sequences with high resolution at 30 frames per second, as well as computer vision applications at high speeds, which require more than 80 fps.

References

1. Marimont, D., Rubner Y., "Un probabiliste Frameword pour Edge Détection et Échelle de sélection", Actes de la Conférence internationale de l'IEEE sur la vision par ordinateur, 207-214, Janvier (1998)
2. T. Lindeberg, "EdgeDetection et détection Ridge avec la sélection Echelle automatique", International Journal of Computer Vision, vol. **30**, numéro 2, 117-154, (1998)
3. Maar, D., E. Hildreth, "Théorie de la détection de bord", Actes royale Soc. Londres, vol. **207**, 187-217, (1980)
4. Technique Advanced Edge Detection: (Techniques de Vision informatique)
5. J. Malik, S. Belongie, T. Leung and J. Shi, "Contour and texture analysis for image segmentation", IJCV, vol. **43**, no. 1, (2001)
6. Duncan D-Y Po et Minh N. Do. - "Directionnel multiscale modeling of images using the contourlet" - Coordinated Science Lab and Beckman Institute, University of Illinois at Urbana-Champaign - Urbana IL 61801, (2003)
7. M. N. Do et M. Vetterli, "The contourlet transform: an efficient directional multiresolution image representation". IEEE Transactions Image Processing, (2003)
8. http://www.xilinx.com/products/design-tools/ise-design-suite.html
9. http://store.digilentinc.com/fpga-programmable-log
10. Wang, R., Cohen, G., Stiefel, K. M., Hamilton, T. J., Tapson, J., and van Schaik, A. An FPGA Implementation of a Polychronous Spiking Neural Network with Delay Adaptation. frontiers in Neuroscience, (2013)
11. Nazari, S., Amiri, M., Faez, K., and Amiri, M. Multiplier-less digital implementation of neuron–astrocyte signalling on FPGA. Neurocomputing, (2015)