IMPLEMENTATION OF SOBEL EDGE DETECTOR IN TO FPGA

Tomáš Kováčik[1], Daniela Ďuračková [1], Vladimír Sedlák[1], Marcel Černák [1]

Roman Záluský[1]

[1] Institute of Electronics and Photonics, Faculty of Electrical Engineering and Information Technology, Ilkovicova 3,
812 19 Bratislava, Slovakia
E-mail: tomas_kovacik@stuba.sk

Received 30 April 2014; accepted 12 May 2014

1. Abstract

This article deals with feasibility of the implementation of Sobel edge detector into FPGA. Constant increase of image processing speed either static image or real time video processing requires searching for new opportunities to implement image processing algorithm in different platforms. Since FPGAs provide sufficient size, speed and parallel programming, they are suitable platform for image processing and computer vision [1].

In the first part of article, we describe theoretical background of the edge detection field. Second part focuses on Sobel edge detector implementation into FPGA and the last section presents the results.

2. Sobel edge detector

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. Classical methods of edge detection involve convolving the image with an operator (a 2-D filter), which is constructed to be sensitive to large gradients in the image while returning values of zero in uniform regions. There is an extremely large number of edge detection operators available, each designed to be sensitive to certain types of edges. Variables involved in the selection of an edge detection operator include:

- Edge orientation**:** The geometry of the operator determines a characteristic direction in which it is most sensitive to edges. Operators can be optimized to look for horizontal, vertical, or diagonal edges.

- Noise environment**:** Edge detection is difficult in noisy images, since both the noise and the edges contain high-frequency content. Attempts to reduce the noise result in blurred and distorted edges. Operators used on noisy images are typically larger in scope, so they can average enough data to discount localized noisy pixels. This results in less accurate localization of the detected edges.

- Edge structure: Not all edges involve a step change in intensity. Effects such as refraction or poor focus can result in objects with boundaries defined by a gradual change in intensity. The operator needs to be chosen to be responsive to such a gradual change in those cases. Newer wavelet-based techniques actually characterize the nature of the transition for each edge in order to distinguish, for example, edges associated with hair from edges associated with a face

Edge detection is a fundamental tool used in most image processing applications to obtain information from the frames as a precursor step to feature extraction and object segmentation. This process detects outlines of an object and boundaries between object and the background in the image. An edge-detection filter can also be used to improve the appearance blurred and anti-aliased video stream [2].

Commonly used method for detecting edges is to apply derivative operators on images. Derivative based approaches can be categorized into two groups, namely first and second order derivative methods. First order derivative based techniques depend on computing the gradient several directions and combining the results of each gradient. The value of the gradient magnitude and orientation is estimated using two differentiation mask.

Part of edge detection tools is also Sobel edge detector [3]. It uses a 3 x 3 convolution table. For a given Image I, horizontal and vertical gradient of each pixel (x, y), are calculated as show in Fig.1.

In each pixel (x, y), the approximations of the horizontal and vertical gradients are combined as follows, to obtain an approximation of the gradient value:

$$G(x,y) = \sqrt{Gx(x,y)^2 + Gy(x,y)^2} \tag{1}$$

Typically, an approximate magnitude is computed using:

$$G(x,y) = |Gx(x,y)| + |Gy(x,y)| \tag{2}$$

The orientation of gradient is calculated as follow:

$$\theta(x,y) = Arctg\,\frac{Gy(x,y)}{Gx(x,y)} \tag{3}$$

$$G_x(x,y) = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I(x,y)$$

$$G_y(x,y) = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I(x,y)$$

Fig.1: *Calculate horizontal and vertical gradient of each pixel (x, y) of image I*

3. Implementation

As described above, complete edge detection consists of compute gradient in horizontal and vertical direction. Great advantage is that computation in horizontal and vertical directions are not interdependent. It allows us to use a big advantage of FPGA which is parallelism [4]. As we can see in Fig.2 in the second part of the design, the signal path is divided into two. First sub-path enters the part to compute horizontal branch and second enters vertical branch. Both consist from DSP48E cores. These cores are based on a real DSP and are implemented into FPGA as a hardware part. In FPGA they are used mainly for fast computing. In our design, there are several cores. Since all calculations are created that result is available after one clock impulse and whole the design is divided into the levels, so we can also use a pipelining to achieve even better results. First part of design is large shifter which has one serial input and nine outputs. We feed input with image data and shifter shifts the data to output. We collect the data in different locations within the shifter. This way we create 3x3 window which moves within the picture. We convolve this window with convolution table. This window selects a part of image which is currently being processed.

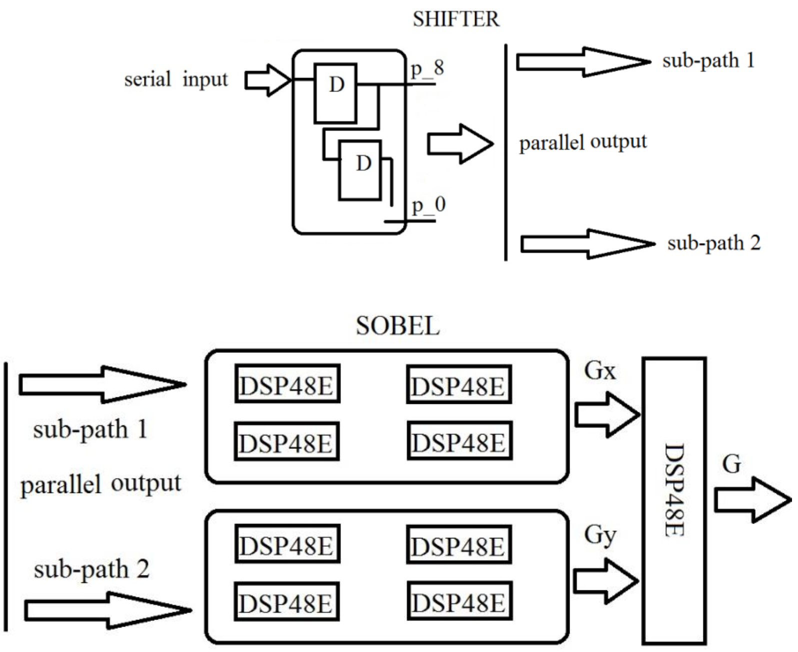

Fig.2: *Sobel edge detector*

4. Simulation and results

Input for our simulation was the text file which contained original image Fig.3. This file was sent to our testbench and applied to Sobel filter. The output represented three text files which contained horizontal gradient image Fig.4, vertical gradient image Fig.5 and final image Fig.6.

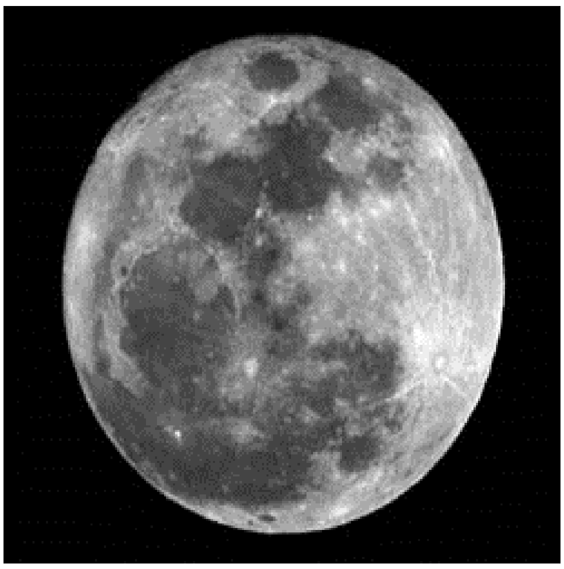

Fig.3: *Original image*

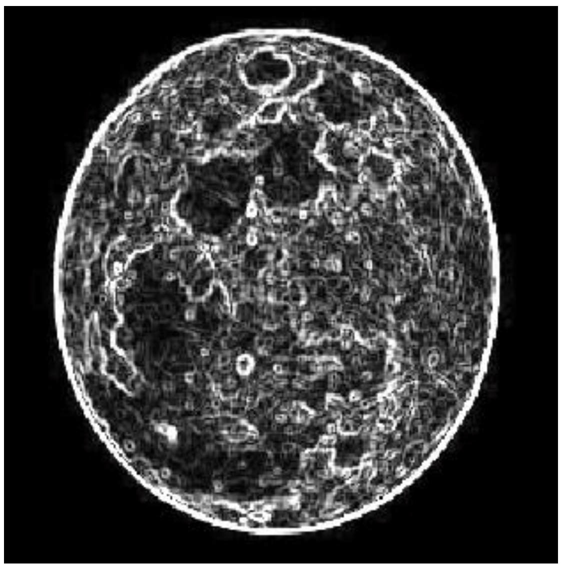

Fig.6: *Final image*

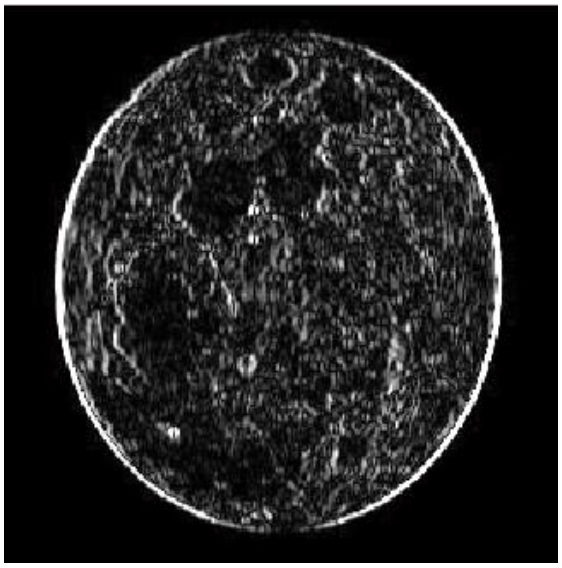
Fig.5: *Vertical image*

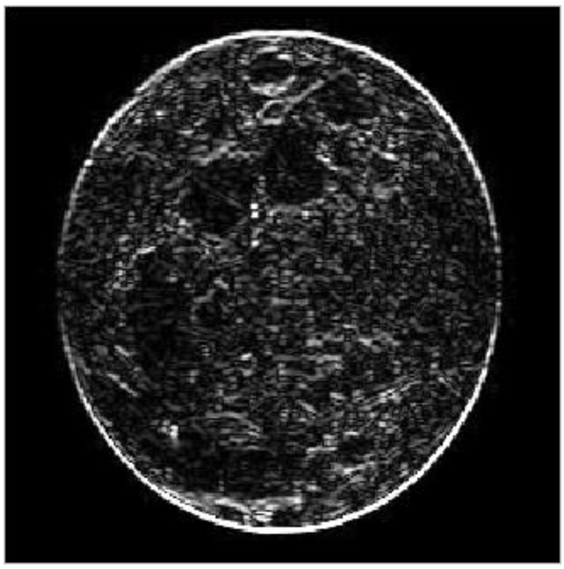
Fig.4: *Horizontal image*

Acknowledgement

This work is resulting from the project VEGA 1/0987/12 sponsored by Ministry of Education, Slovak Republic.

5. Conclusion

This paper presents a brief survey on implementation of edge detector in to FPGA. Advantage of this way is that we can create design which is greatly parallel. In proper implementation this can be equal or more faster compared with CPU or GPU. In regard to size of FPGA we can applied either simply filters or complex system to image processing and computer vision. Disadvantage of this approach is that we have to create system for ensuring data flow and store. This system takes a big part of chip field and make image processing slower.

References:

[1] Said, Y.; Saidani, T.; Smach, F.; Atri, M., "Real time hardware co-simulation of Edge Detection for video processing system," *Electrotechnical Conference (MELECON), 2012 16th IEEE Mediterranean* , vol., no., pp.852,855, 25-28 March 2012

[2] A. H. Neoh, A. Hazanchuk, “Adaptive Edge Detection for Real-Time Video Processing using FPGAs”, *Global Signal Processing (2004)*

[3] Shrivakshan, G.T., "An analysis of SOBEL and GABOR image filters for identifying fish," *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on* , vol., no., pp.115,119, 21-22 Feb. 2013

[4] Ye Li; Qingming Yao; Bin Tian; Wencong Xu, "Fast double-parallel image processing based on FPGA," *Vehicular Electronics and Safety (ICVES), 2011 IEEE International Conference on* , vol., no., pp.97,102, 10-12 July 2011 A.