FPGA Based Edge Detection Architecture for Embedded Applications

1st Lucas Caetano Pereira ,2nd Vagner Rosa,3rd Nelson Lopes Duarte Filho
Centro de Ciencias Computacionais - C3
Universidade Federal do Rio Grande - FURG
Rio Grande, Brazil
caetano02117@gmail.com, vsrosa@gmail.com, nelson.duartefilho@gmail.com

***Abstract*—Real-time image processing dependent applications are widely spread in the present industry and the academy. Such applications include inspection of industrial processes, object tracking, facial analysis and many others. In these kinds of applications detection of edges in the input image is a basic but fundamental step whereas this process remove unnecessary information and brings up the information needed in ensuing processing steps. A common approach to detect the edges of a image is to apply the Sobel filter allied with some noise reduction method. The Sobel filter is a gradient based method, that highlight the edge pixels of a image. FPGAs are a effective type of device to process in real time vast amounts of data such as the pixels of a image due it's fine-grain parallelism and re-configurable structures. This paper describes a hardware architecture for edge detection which consists of a Mean Filter for noise reduction and a Sobel Filter. Both method implemented for Field Programmable Gate Array (FPGA) technology as 8-bit architectures .**

I. Introduction

Modern manufacturing environments have nowadays vast and complex production lines that depend in many cases of real-time based inspection processes in order to ensure quality and the reproducibility of the final product and increase produtivity levels in the whole scope of the production routine.

Other field widely dependent of efficient image processing running at real-time is the autonomous cars research [1], where a good understanding of the scene of a image and the object present in it. In this kind of application the correct processing of the image information guarantee the safety inherently necessary for the well functioning of the autonomous car by reliably providing useful information of the image.

The noise present in a digital image, which is essentially a random variation of color and brightness can in many cases hide important information contain in the original image, making it difficult the correct extraction of certain image features. Therefore, ensure that only the important morphological information about the scene are highlighted and the noise present in the image be discarded is vital in any image dependent application.

The edges of image are considered to be most important image attributes that provide valuable morphological information of the object in the scene. The quality of the detected edges play a important role in more complex image processing routines and in the correct behavior of real-time computer/machine vision systems. Such be in manufacturing environments, autonomous vehicle research and many other applications. Thus edge detection is one of the most important processes in lower level image processing to understand the information contain in a image. The literature presents a series of edge detection methods, for the architecture presented in this paper, the chosen approach is to apply a mean filter to reduce the noise in the input image, next apply the Sobel operator to accentuate the edge pixels and then apply a threshold to let only those pixels visible.

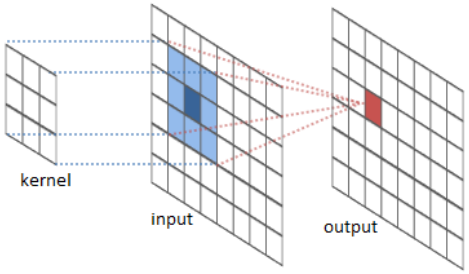

Fig. 1. Convolution of a image with a kernel [3]

FPGA are suitable for applications that require real-time processing of large data amounts due the high hardware customization and parallelization offered for this technology. In this research field FPGAs have shown very high performance although the low frequencies FPGA commonly operates. Such performance came from high need of parallelism in image processing and the possibility of instantiate large internal memories banks on FPGAs that can be accessed in parallel [2].

II. Important Concepts

A. Convolution of a Image

Convolution is the process of adjust the value of each pixel of a image based in a weighted value of it's neighbors through the passing of a kernel, Fig 1. The kernel slides over every pixel of the image and a new value to each pixel is calculated as a weighted sum of the nearby pixels.

*This work was supported by CNPq – National Counsel of Technological and Scientific Development

All authors are with the Center of Computer Science, Federal University of Rio Grande, Rio Grande - RS, Brazil

B. Mean filter

The mean filter, or average filter is a convolution based method that calculates a new value for pixel in the center of the kernel based in the average value of the pixels inside the kernel window. This process smooth the image, working as a Low-pass filter and reducing the noise in it. The size of the kernel can change dramatically the results since the information of each pixel is affected by increasingly distant pixels as the kernel grows, the shape on the other hand can by any. The usual approach for this filter in most applications is a 3x3 square kernel, this combination of size and shape require that each position of the kernel be 1/9 in order to correct calculate the average value of the pixels. This configuration is show in Eq.1

$$K = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \tag{1}$$

C. Sobel Filter

The Sobel filter is a widely used method that accentuate the pixels on the edges of a image by calculating a approximation of the intensity gradient of the image in each pixel. This calculus is done through the convolution of two 3x3 Kernels with the image [4]. The two kernels represented in Eq.3 and Eq.2 are each used in separated convolution processes with image to generate the gradients *Gx* and *Gy*.

$$K_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2}$$

$$K_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \tag{3}$$

Then the absolute values of both gradients are added, reaching the value of the resultant gradient Eq.4

$$G = |G_x| + |G_y| \tag{4}$$

After the final gradient G, or transient image, have been calculated, each pixel *P* of this new image is compared with a predefined threshold value *T* in order to determine if the pixel is an edge pixel or not. If a pixel is greater than the threshold value, that pixel is considered to be a edge pixel and its final value *P'* is set to 1, else ways the pixel is considered a non edge pixel and its value is set to 0, Eq.5.

$$P' = \begin{Bmatrix} 1, if P > T \\ 0, else \end{Bmatrix} \tag{5}$$

III. Methodology

The architecture proposed in this work has been developed using VHSIC Hardware Description Language (VHDL) and its behavioral simulation was made with the ModelSim software by Altera. A testbench were implemented also in VHDL language to stimulate the digital design containing the filters

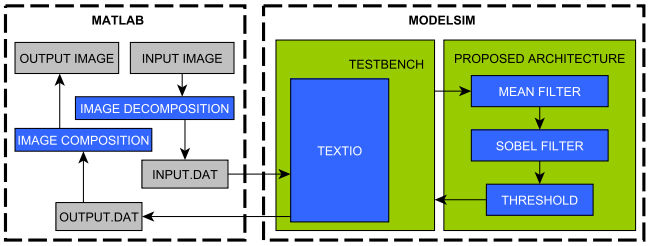

Fig. 2. Simulation Methodology

and interact with the input and output files. In order to convert the image in a format more easy to work with in the VHDL context, two scrips were made with the Matlab software, the full simulation enviroment can be se in Fig.2.

The first step in the simulation cycle is performed by one of the script cited previously. The procedure is to transform the input RGB image into a gray scale image, decompose this image into a bit vector containing each pixel of the image and create a .dat file to be read by the testbench. During the simulation process a TEXTIO based application following the implementation in [5] whitin the testbench file reads the .dat file and stimulate the implemented architecture with each pixel of the image sequentially.

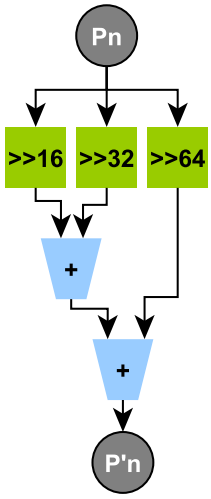

Fig. 3. Approximation of the 1/9 ratio.

After those stages, the proposed design operates on the given image, first with the Mean filter to smooth the image, removing noise. As stated in previous sections the classic Mean filter present in literature uses a kernel full with the value 1/9 to fulfill its function. But the division operation can be high costly in a hardware context, so a valuable approximation has to be found in order to simplify the implementation of the Mean Filter.

$$\frac{1}{9} = 0.111111 \tag{6}$$

$$\frac{1}{16} + \frac{1}{32} + \frac{1}{64} = 0.109375 \tag{7}$$

$$\frac{1}{16} + \frac{1}{32} + \frac{1}{64} \cong \frac{1}{9} \tag{8}$$

As show through Eq. 6, 7 and 8 the values of 1/9 and 1/16 + 1/32 + 1/64 can be considered as almost equivalent. This new value is much more suitable to be implemented in VHDL once its composed by sums and divisions by powers of 2, witch are essentially shifts, a much more fundamental operation in a hardware context. This new value was used to develop the digital circuit that apply the Mean filter over the given input image. Let P_n be a pixel in the input image, by shifting the value of this pixel three times, four times and five times right the result values are the value of the pixel divided by 16, 32 and 64 respectively. Those values are sum to obtain the weighted value of the pixel P'_n, Fig. 4.

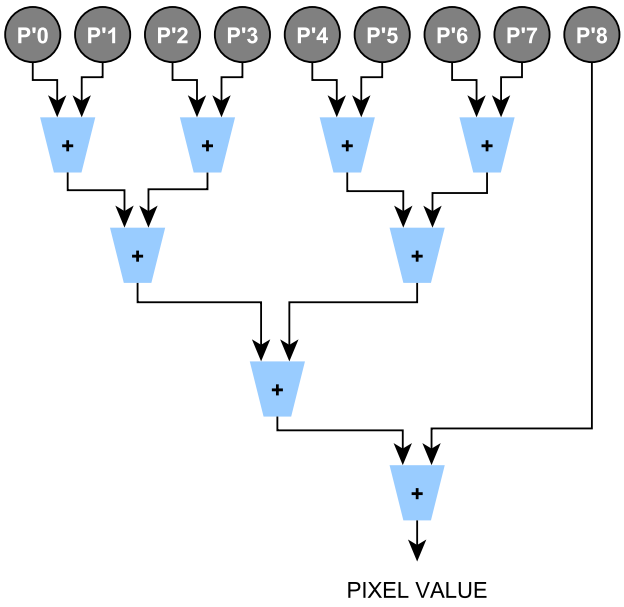

Fig. 4. Implemented Architecture of the Mean Filter.

After the weighted values of each pixel P' in the kernel are calculated, all values are sum in order to reach the final value of the pixel at the center of the kernel in the new image Fig.1.

The next process is the applying of the Sobel Filter over the image. The digital circuit implemented to applying the Sobel Method in the input image is a combinational 8-bit architecture that follows the design described in [6]. As show in Fig. 5 the implemented design shifts the value of each pixel in involved twice right, this operations is done to keep the 8-bit architecture of a possible overflow.

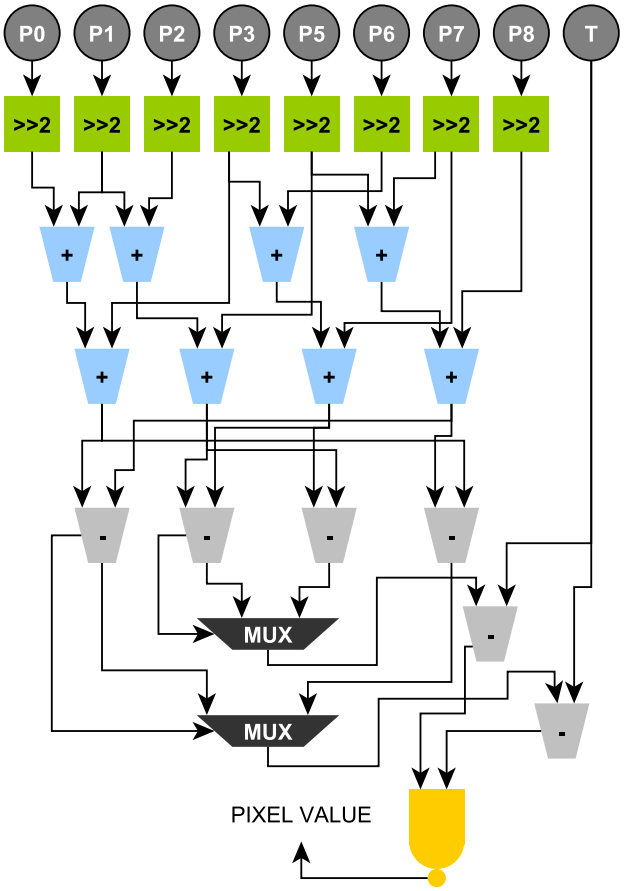

Fig. 5. Implemented Architecture of the Sobel method and Threshold applying.

Then the adjusted values are sum and subtracted in order to find the values of the gradients Gx and Gy. Two multiplexers are used to get the absolute value of each of the gradients. Next the values of each gradient are compared with the threshold throught the overflow signal of two subtraction blocks. If the threshold if greater than both the gradients, the pixel is considered not a edge pixel and its value is set o 0, otherwise the value of the pixel is set to 1.

As each pixel pass through the entire architecture, being processed by each filter and having passed through the application of the threshold, those pixels are delivered by the simulated architecture to the TEXTIO application in the testbench to be written in a output .dat file. With the second script developed with Matlab the output .dat file is assembled into the final image.

IV. Results

The implemented architectures for the application of the selected filters successfully extract the edges of a given input image. The Fig. 6 presents a input image at all processing stages, in the left upper corner the original image is displayed, followed by the image after the applying of the Mean Filter in the right upper corner. In lower segment of the figure, the

Fig. 6. Input image, the image after the sobel method and after the threshold apply

version of the image after the Sobel Method be applied can be see, before the threshold been applied in the left and after in the right.

V. Conclusion and Future Works

This result demonstrates that fundamental image processing procedures can be brought to the hardware and that FPGA devices are employable in implementing these basic blocks. Allowing image processing dependent applications to balance the processing load between their hardware and software components. This also enables these applications to enjoy the high scalability and parallelism offered by FPGA devices.

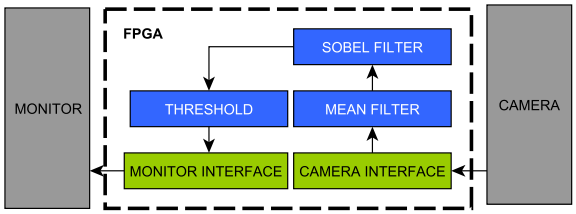

Fig. 7. Future Embedded Architecture

In future works this architecture will be embedded in a DE0-NANO industrial board developed by Altera to be fully integrated with the camera TRDB-D5M using the VHDL modules implemented in [7] and a interface with a monitor will also be implemented, resulting in the architecture demonstred in Fig. 7

Hereafter other filters and basic blocks commonly used in image processing will be implemented in VHDL and simulated in order to compare its results with the presented in this work and others.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3354–3361.

[2] T. Saegusa, T. Maruyama, and Y. Yamaguchi, "How fast is an fpga in image processing?" in *2008 International Conference on Field Programmable Logic and Applications*, Sept 2008, pp. 77–82.

[3] C. Olah. Understanding convolutions. [Online]. Available: http://colah.github.io/posts/2014-07-Understanding-Convolutions/

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[5] S. Zhang, J. Zhu, and C. Wang, "Application of textio in the simulation of fpga image processing algorithm," in *Information Science and Control Engineering (ICISCE), 2016 3rd International Conference on*. IEEE, 2016, pp. 235–238.

[6] N. Nausheen, A. Seal, P. Khanna, and S. Halder, "A fpga based implementation of sobel edge detection," *Microprocessors and Microsystems*, vol. 56, pp. 84–91, 2018.

[7] S. C. da Silva Filho, "Sistema embarcado para mapeamento de chanfros em chapas metalicas usando fpga e cmera," Rio Grande, 2015.