A New Fast Median Filtering Algorithm Based on FPGA

Leiou Wang[1, 2] *,
[1] Digital System Integration Lab, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China
[2] University of Chinese Academy of Sciences, Beijing 100080, China
* Email: wangleiou@mail.ioa.ac.cn

Abstract
Edge detection plays an important role in the field of image processing. However, due to the random noise in images, image filtering should be taken in image preprocessing. The median filtering algorithm can suppress the noise in images, thus this algorithm is widely employed in many different fields. By analyzing the common 3 x 3 filtering window mathematical model and different median filtering algorithms, this paper proposes a fast median filtering algorithm based on Field Programmable Gate Array (FPGA). This algorithm can effectively reduce the compare operation and it is suitable to implement on FPGA. In addition, this paper also implements Sobel edge detection algorithm, which can accomplish edge detection from a median filtering image.

1. Introduction

Edge detection is one of the significant techniques in the field of image processing such as object tracking, image segmentation and feature extraction. Early edge detection algorithm used local operators to approximately compute the gradient of gray level of an image in the spatial domain. The position of the gradient local maximum is considered to be the edge points. Examples of the edge detection based on gradient are Prewitt and Sobel operators [1-2]. The Laplacian of Gaussian operator for edge detection has been proposed by Marr and Hildreth [3]. The Canny operator can give the edge information of both intensity and direction [4]. Considering the accuracy and simplicity of edge detection, this paper employs the Sobel operator. However, all these operators are sensitive to noise, so a filtering algorithm should be taken in the preprocessing.

It is well known that image may be contaminated with noise in the process of data transmission, for example the inherent defection of the image system, the working conditions and the capturing devices [5]. As a result, the image quality is decline and the accuracy of the following processing will be awfully affected, for instance edge detection, image segmentation, and feature extraction. It is necessary to eliminate the noise from the image by using an image filtering algorithm. But the noise removal is often accomplished at the expense of blurring or losing the image information. Therefore, as a spatial filtering technique, median filtering algorithm, compared with other filtering algorithms such as the mean filtering, can effectively eliminate impulse noise, salt and pepper noise, and keep the edge information of the image.

Image processing algorithms need to process very large amount of data, so software implementation will be very time consuming. For many systems requiring real-time processing, the implementation speed is often considered as a key factor. As a result, the image processing algorithm is suitable to be implemented in hardware. FPGA is reconfigurable device, and it is suitable for pipeline and parallel data processing. According to the rich hardware resources and the advantage of parallel processing, this paper implements the fast median filtering and Sobel edge detection algorithm on a FPGA.

The paper is organized as follow. In section 2, the principle of the fast median filtering algorithm is proposed and Sobel edge detection algorithm is introduced. In section 3, the implementation of this fast median filtering and Sobel edge detection algorithm are elaborated in detail, including the 3 x 3 window, the median filtering module and the edge detection module. In section 4, the experiment analysis is provided and some conclusions are derived in section 5.

2. The principle of algorithm

2.1 Related works

Median filtering is a popular algorithm of noise removal, employed extensively in applications involving signal and image processing. This non-linear technique has been proved to be a good alternative to linear filtering as it can be effectively suppress impulse noise while preserving the edge information.

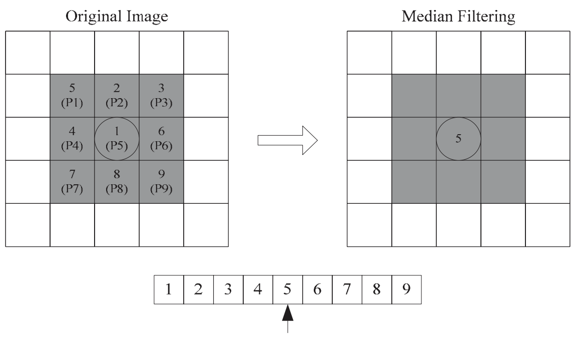

Figure 1. Median filtering algorithm

The median is defined as the middle element of a group of numbers after the numbers are sorted [6]. Note that the group should contain odd number of elements.

The algorithm can be explained with the following steps. Consider an n x n (3 x 3) window as shown in Figure.1.

The Standard Median Filtering (SMF) algorithm can be defined:

$$g(x, y) = med\{f(x-i, y-j)\} \; (i, j) \in W \qquad (1)$$

The g(x, y) is the output value and f(x-i, y-j) is the input pixel value, W is the n x n window.

The compare operation is very huge in the SMF algorithm. For an n x n window, it needs:

$$(n\times n-1)+..+[n\times n-1-(n\times n-1)]=\frac{n^2\times(n^2-1)}{2} \qquad (2)$$

The compare operation is O (n^4), if n = 3, the compare operation needs 36 times.

Zhang [7] proposed a median filtering algorithm that accelerated calculation by using the mean value. The Mean based fast Median Filtering (MMF) algorithm divided the elements of the n x n window into two groups, one was less than the mean value of the n x n window, another was greater than the mean value, and then sorted the elements in one of them. If n = 3, the compare operation needed about 25 times.

Zhu [8] proposed a median filtering algorithm that used a cross window. If n = 3, the compare operation needed 8 + 7 + 6 + 5 + 4 = 30 times in this Not full Standard Median Filtering (NSMF) algorithm.

Fu [9] proposed a median filtering algorithm that sorted the columns, rows and diagonal elements in a 3 x 3 window. If n = 3, the column compare operation needed 9 times, the row compare operation also needed 9 times, and the compare operation of the three elements in the diagonal needed 3 times, so the compare operation totally needed 21 times in this Diagonal Median Filtering (DMF) algorithm.

2.2 Fast median filtering algorithm

By analyzing above algorithms, this paper proposes a Fast Median Filtering (FMF) algorithm that needs less compare operation and makes full use of the parallel processing ability of FPGA.

There are three steps to realize the FMF algorithm.

First step: The pixels are sorted horizontally. The pixels of each row are sorted in ascending order. Generally, the compare operation needs 9 times. But due to FPGA parallel processing ability, the compare operation just needs 3 clock cycles.

(a) min $_{row1}$ = min [P1, P2, P3]; med $_{row1}$ = median [P1, P2, P3]; max $_{row1}$ = max [P1, P2, P3];

(b) min $_{row2}$ = min [P4, P5, P6]; med $_{row2}$ = median [P4, P5, P6]; max $_{row2}$ = max [P4, P5, P6];

(c) min $_{row3}$ = min [P7, P8, P9]; med $_{row3}$ = median [P7, P8, P9]; max $_{row3}$ = max [P7, P8, P9];

Second step: The pixels are sorted vertically. For obtaining the maximum pixel value of the first column, the compare operation only needs 2 times. Similarly, for obtaining the minimum pixel value of the third column, the compare operation also needs 2 times. The compare operation totally needs 7 times and 3 clock cycles.

(d) max $_{col1}$ = max [min $_{row1}$, min $_{row2}$, min $_{row3}$];

(e) med $_{col2}$ = median [med $_{row1}$, med $_{row2}$, med $_{row3}$];

(f) min $_{col3}$ = min [max $_{row1}$, max $_{row2}$, max $_{row3}$];

Third step: In third step, sort the maximum pixel value of the first column, the median pixel value of the second column and the minimum pixel value of the third column, and pick up the median from these three elements. This median is the median of the nine pixels. Discard all the other elements. The compare operation needs 3 times and 3 clock cycles.

(g) med $_{window}$ = median [max $_{col1}$, med $_{col2}$, min $_{col3}$];

From the above analysis, the FMF algorithm makes full use of the parallel processing ability of FPGA. The compare operations reduce to 19 times, while the clock cycles cut down to 9. So the FMF algorithm significantly improves the processing speed, as well as greatly increases the efficiency of image processing.

2.3 Sobel edge detection algorithm

Edge detection algorithm uses local operators to approximately compute the gradient of gray level of an image in the spatial domain. The position of the gradient local maximum is considered to be the edge points.

Mathematically, for an image F(x, y), the gradient magnitude G(x, y) and gradient direction Θ(x, y) are computed:

$$G(x, y) = \frac{\partial F(x, y)}{\partial x}\cos\theta + \frac{\partial F(x, y)}{\partial y}\sin\theta \qquad (3)$$

$$G(x, y) = G_x \cos\theta + G_y \sin\theta \qquad (4)$$

$$\theta(x, y) = \arctan(\frac{G_y}{G_x}) \qquad (5)$$

$$|G(x, y)| = \sqrt{G_x^{\,2} + G_y^{\,2}} \qquad (6)$$

Typically, an approximate magnitude is computed using the following equation [10-11]:

$$|G(x, y)| = |G_x| + |G_y| \qquad (7)$$

The edge detection algorithm has included the Roberts, Prewitt and Sobel operators. The Sobel operator is widely used by far and it is shown in Figure.2.

-1	0	1
-2	0	2
-1	0	1

G_x operator

1	2	1
0	0	0
-1	-2	-1

G_y operator

Figure 2. Sobel operator

The horizontal gradient G_x and the vertical gradient G_y can be represented as below:

$$G_x = (P3 + 2P6 + P9) - (P1 + 2P4 + P7) \quad (8)$$

$$G_y = (P1 + 2P2 + P3) - (P7 + 2P8 + P9) \quad (9)$$

Then G_x and G_y are computed from these pixel values, and Pi is the pixels intensity in i position as it appears in Figure.1.

The Sobel operator has distinct advantages, although it is slightly more complex than other methods. It gives an estimate of edge direction as well as edge magnitude at a point. What's more, the Sobel operator is still relative easy to implement in hardware, especially by using a pipeline approach.

3. The implementation of algorithm

3.1 3 x 3 window

Image processing algorithms usually deal with neighbor pixels. In order to obtain a 3 x 3 window, two First In First Out (FIFO) memories are needed to save image pixels. A 3 x 3 window structure is shown in Figure.3.

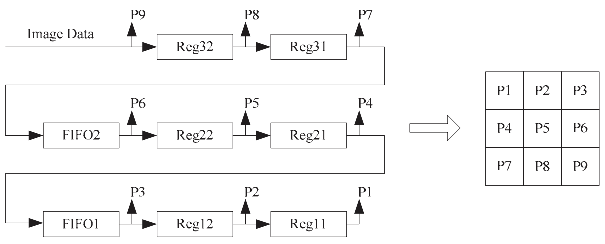

Figure 3. 3 x 3 window structure

The depth of the FIFO corresponds to the row width of the input image. These six registers are used to save the column pixels in different rows. By this way, the 3 x 3 window can be easily obtained, and all these pixels in the window are sent to the median filtering module.

3.2 Median filtering module

The median filtering module is basically composed of the comparison unit and the delay unit. In order to keep the synchronization of the entire module, the comparison unit is completed by using synchronous circuits. If some pixels do not execute the compare operation in a clock cycle, these pixels can save in the delay unit, and then they can participate in the next clock cycle. The median filtering module structure is shown in Figure.4.

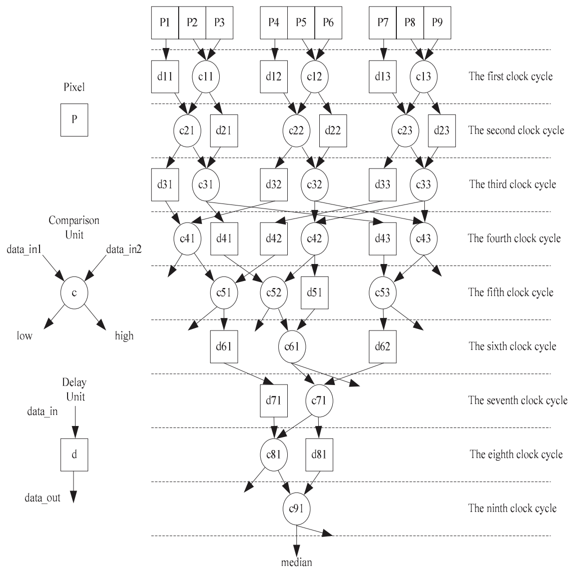

Figure 4. Median filtering module structure

This median filtering module includes 19 comparison units, and the processing divide into 9 clock cycles. The first step of the FMF algorithm is accomplished from the first clock cycle to the third clock cycle, and the pixels are sorted horizontally. Due to the parallel processing ability of FPGA, although compare operation needs 9 times, the compare operation can complete in 3 clock cycles. Therefore, the processing efficiency is significantly improved. The second step and the third step are similar to the first step.

3.3 Sobel edge detection module

The Sobel edge detection module is mainly composed of the add unit, the subtract unit and absolute unit. The add unit and subtract unit are synchronous circuits. The Sobel edge detection module structure is shown in Figure.5.

The structure corresponds to equation (8) and (9). This module employs the pipeline approach, and the delay from input ports to output port is 4 clock cycles. The G_x computation needs 6 add units and 1 subtract unit. The entire module needs 13 add units, 2 subtract units and 2 absolute units.

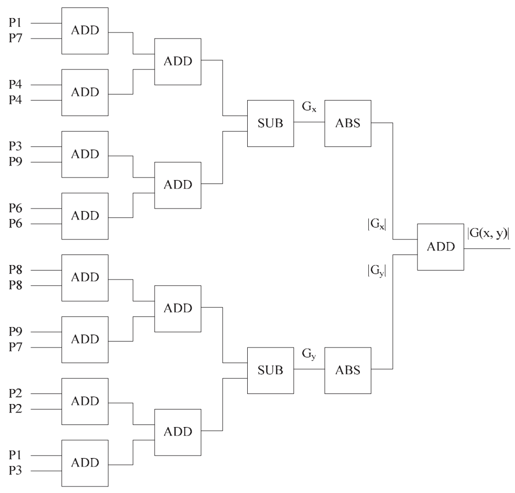

Figure 5. Sobel edge detection module structure

4. The experiment analysis

In order to implement the FMF and Sobel edge detection algorithm, this paper performs experiments under Altera Stratix IV EP4SE820F43C3 FPGA and employs Quartus II 12.0, Matlab 7.1 as the main development software. The experimental results are shown in Figure.6.

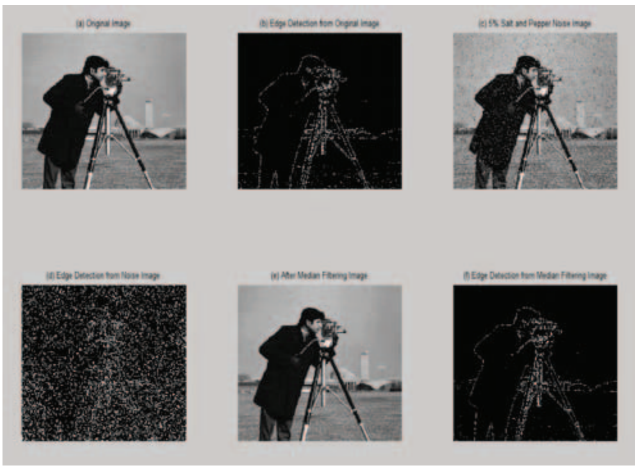
(a) Original Image
(b) Edge Detection from Original Image
(c) 5% Salt and Pepper Noise Image
(d) Edge Detection from Noise Image
(e) After Median Filtering Image
(f) Edge Detection from Median Filtering Image

Figure 6. Experimental results

The (a) is the original image. The (b) is edge detection image from (a). The (c) is contaminated image with 5% salt & pepper noise. The (d) is edge detection image from (c). The (e) is the image after median filtering. The (f) is edge detection image from (e). From the experimental results, the good performance of the FMF and Sobel edge detection algorithm can be easily seen. The efficiency of the FMF algorithm is shown in the Table 1.

Table 1. The efficiency of the FMF algorithm

algorithm	compare operation times
SMF	36 (3 x 3 window)
MMF	25 (3 x 3 window)
NSMF	30 (3 x 3 window)
DMF	21 (3 x 3 window)
FMF	19 (3 x 3 window)

The experiment results show that the FMF algorithm has dramatically improvement in compare operation. Each comparison unit costs 17 logic elements, so the FMF algorithm saves 289 logic elements comparing with SMF. What's more, the FMF algorithm doesn't lost edge detection accuracy comparing with other algorithms.

5. Conclusion

By analyzing the common 3 x 3 filtering window mathematical model and different median filtering algorithms, the FMF algorithm is proposed in this paper. What's more, this paper implements the FMF and the Sobel edge detection algorithm based on a FPGA. Experiment results show that the FMF algorithm has a good performance in elimination noise and it can keep the edge information of the original image. At the same time, the FMF algorithm can dramatically reduce the compare operation from 36 times to 19 times. By using the parallel processing ability, the clock cycle can be also cut down to 9.

References

[1] A. Rosenfel, IEEE Transactions on Biomedical Engineering, p.83 (1989).
[2] I. Sobel, Computer Graphics and Image Processing, p.127 (1978).
[3] D. Marr and E. C. Hildreth, Proceedings of the Royal Society, 207B, p.187 (1980).
[4] J. Canny, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6), p.679 (1986).
[5] Dong Hanlei, Xu Liping and Gao Yingmin, Communication technology, p.171 (2009).
[6] Li Leiming, Zhang Huanchun and Zhang Bo, Electronic Engineer, p.48 (2004).
[7] Zhang Li, Chen Zhiqiang, Gao Wenhuan and Kang Kejun, J Tsinghua Yniv (Sci & Tech), p.1157 (2004).
[8] Zhu Jie, Zhu Xiaojuan and He Ming, Computer Measurement & Control, p.798 (2007).
[9] Fu Yiqiang, Master Dissertation of NanChang University, (2006).
[10] Zhengyang Guo, Wenbo Xu and Zhilei Chai. 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, p.169-171 (2010).
[11] I. Yasri, N. H. Hamid and V. V. Yap. 2008 International Conference on Electronic Design, (2008).