EVALUATION OF STANDARD AND DIRECTIONAL MEDIAN FILTERS ALGORITHMS AND THEIR IMPLEMENTATION ON FPGA FOR MEDICAL APPLICATION

Talbi, F.; Alim-Ferhat, F.; Seddiki, S.; Hachemi, B.
Centre de Développement des Technologies Avancées
BP 17, Baba Hassen, Algiers 16303, Algeria
falim@cdta.dz

ABSTRACT

The standard and the directional median filters are effective methods for the removal of impulse-based noise from the images. The main advantage is being the preserving of edges as compared to the mean filter.
The main objective of this article is to implement the standard and the directional median filters on FPGA (Field Programmable Gate Array) in order to eliminate impulsive noise in the medical image. As a first step, two algorithms were developed and validated by Matlab tool.
Subsequently, two architectures are proposed and implemented using the Xilinx ISE 12.2 environment.

Index Terms: Impulsive noise, Median filter, Directional median filter, PSNR, FPGA.

1. INTRODUCTION

THE world of science has used in several image processing technique to improve image quality; indeed, this latter is often affected by noise during acquisition. The impulsive noise is one of the major problems in medical image processing. Impulse noise from digital images can be removed by using mean, standard median and directional filters. Impulsive noise is commonly appears during images acquisition; to eliminate it, median filters are used [1, 2].

A reliable and precise filtering of normal or pathological medical images from imaging systems remains an important objective in processing of medical information. This is because it represents the first step in the analysis chain leading to assisted diagnosis in studying internal structures morphology.

In the literature, there are many kinds of filters realizing this task. However, some of them have disadvantages such as degradation of the medical image quality, suppression of elements containing helpful information in the diagnosis, etc.

To increase the quality and precise filtering, standard and directional median filters are used in many industrial applications that go from machine vision to robotics going through medical imaging.

The standard and directional median filters are used widely in medical image processing since it preserves edges while removing noise.

Our objective is the elimination of impulsive noise in medical images. When noise intensity is increasing, a standard median or directional filter remains many shots unfiltered. To overcome the limitation of standard and directional median filters, some works have been proposed in the literature to improve the performances and the efficiently of these filters. Several works have been realized; among them, implementation based on FPGA [3, 4, 5]. To achieve our goal, we have organized this paper as follows.

The first section introduces filters and their various applications. The second section deals with the operating principle of the two filters studied in this paper, and their implementation in MATLAB software tool. The third section focuses on hardware implementation on FPGA programmable circuit, simulation results and temporal performances. The last section concludes the paper and draws-up perspectives and future works on the development of directional median filter.

2. STANDARD MEDIAN FILTER

The median filter operates according to the following steps [6,7] :

1) Sort the values in ascending order.
2) Replace the value of the central pixel by the value in the middle of the sorted values
3) Repeat this process for all image pixels.

Generally, the sorting is done in ascending order. The ordered elements were noted f_i, the ascending sort is characterized by:

$$f_1 < f_2 < \ldots < \frac{f_{n+1}}{2} < \ldots < f_{n-1} < f_n$$

The median element of the neighborhood is $\frac{f_{n+1}}{2}$.

2.1 Software implementation

To verify the efficiency and quality of the proposed filtering technique, we test this procedure on several MRI (Magnetic Resonance Images) images using MATLAB tool. The program was applied to medical images with different noise levels.

Fig. 2 below illustrates the application of the standard median filter on noisy MRI images with a rate of 20% with an application window of 3x3 filter.

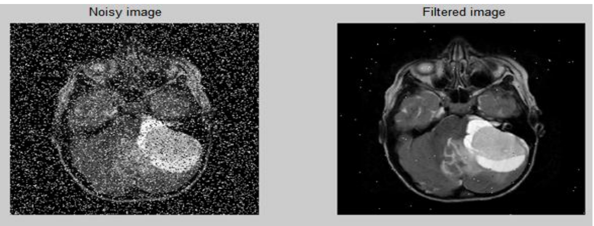


Figure 1 Application of the median filter on MRI images by MATLAB tool for the image

3. DIRECTIONAL MEDIAN FILTER

This filter uses a pulse detector, which is based on the differences between the current pixel and its neighbors aligned with four main directions.

After the pulse detection, it does not just replace the noisy pixels identified by median filter outputs but continues to use the information of the four directions to weigh the pixels in the window to preserve the details.

3.1 Method of operation

In this section, we describe the method of operation of the directional median filter following next steps [8]:

Step 1: Pulse detector

Fig. 2 shows the edges aligned with the four main directions

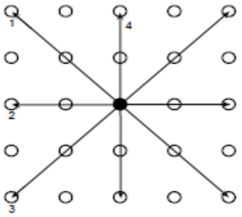


Figure 2 Graphical representations of the four directions

According to the previous figure, each of these four directions contains 05 pixels. S_k (k=1 .. 4) represents a set of coordinates aligned with the kth direction centered at (0, 0).

In the 5x5 window centered at (i, j), we calculate the index $d_{i,j}^{(k)}$for each direction; Knowing that $d_{i,j}^{(k)}$is the absolute sum of all the differences in gray levels values between $y_{i+s,j+t}$ and $y_{i,j}$with (s, t)ϵS_k^0 ,$d_{i,j}^{(k)}$is the direction index defined by:

$$d_{i,j}^{(k)} = \sum\nolimits_{(s,t)\ \epsilon S_k^0} |y_{i+s,j+t} - y_{i,j}| \quad 1 \le k \le 4$$

Each index direction is sensitive to edge aligned with a given direction. Then, the minimum of these four directions indices is used for pulse detection, which can be denoted by:

$r_{i,j} = \mathrm{mi\,n}\left\{d_{i,j}^{(k)} : 1 \le k \le 4\right\}$ The value of r(i, j) can be discussed in 3 cases:

- When the current pixel is a flat region without noise, $\boldsymbol{r_{i,j}}$,is small, because the four indices direction are small.
- When the current pixel is an edge pixel, $\boldsymbol{r_{i,j}}$ is also small, because at least one indice direction index is small.
- When the current pixel is a pulse, $\boldsymbol{r_{i,j}}$ is great, because the four indices directions are great.

Then we establish a threshold T to compare the coefficient $\boldsymbol{r_{i,j}}$ in order to determine if the pixel is noisy or not. The pulse detector is defined as follows:

Then we establish a threshold T to compare the coefficient $\boldsymbol{r_{i,j}}$in order to determine if the pixel is noisy or not.

The pulse detector is defined as follows:

$$y_{i,j} = \begin{cases} \text{noisy pixel} & \text{if } r_{i,j} > T \\ \text{pixel noiseless} & \text{if } r_{i,j} \le \mathrm{T} \end{cases}$$

Step 2: Calculation of the median value

After the pulse detection, the noisy pixels will be replaced by the calculated median values according to the direction which is the lowest standard deviation.

The first step is to calculate the standard deviation $\sigma_{i,j}^{(k)}$of gray levels values between $y_{i+s,j+t}$ and (s, t) ϵS_k^0 (k=1 à 4). The equation of $L_{i,j}$is given by:

$$L_{i,j} = \mathrm{argmin}\left\{\sigma_{i,j}^{(k)} : k = 1 \text{ to } 4\right\}$$

With:

$$\sigma_{i,j}^{(k)} = \sqrt{1/5 \sum_{(s,t)\ \epsilon S_k^0} (y_{i+s,j+t} - \mathrm{mean}^{(k)})^2]}$$

$$\mathrm{mean}^{(k)} = \mathrm{mean}\left(\text{the pixel values in the } k^{\text{th}}\text{direction}\right)$$

The argmin operation is used to calculate the function minimizer. The standard deviation shows how values are grouped in the average value of all pixels.

$L_{i,j}$ Indicates that the pixels are on the same direction as the central pixels are closer to each other. So to reconstruct the noisy pixel, the following equation is used:

$$m_{i,j} = \mathrm{median}\ \{W_1 * y_{i+s,j+t} : (s,t)\epsilon\Omega^3\}$$

$$\text{With } W_1 = \begin{cases} 2, & (s,t)\epsilon S_{li,j}^0 \\ 1, & \text{elsewhere} \end{cases}$$

According to the equations listed above, the output of the directional median filter is given by:

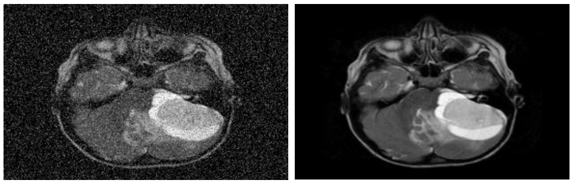
$$u_{i,j} = \alpha_{i,j} y_{i,j} + (1 - \alpha_{i,j}) m_{i,j}$$

With

$$\alpha_{i,j} = \begin{cases} 0 & \text{si} \quad r_{i,j} > T \\ 1 & \text{si } r_{i,j} \le \mathrm{T} \end{cases}$$

3.2 Software implementation:

Fig. 3 below shows the result of the application of directional median filter on noisy MRI images with a noise rate of 20% under the MATLAB tool.



(a1) (a2)

Figure 3 (a1) Medical image with 20% impulsive noise, (a2) filtered image with the directional median filter

3.3 Comparison of the two filters PSNR parameter

According to the results of the calculation of the PSNR for the median filter and the directional median, we make a comparison in order to justify our choice for directional median filter [9].

$$\mathrm{EQM} = \frac{\sum(M(i,j) - T(i,j))^2}{m * n}$$

$$\mathrm{PSNR} = 10 * \log \frac{255^2}{\mathrm{EQM}}$$

Fig. 4 s shows the different values of the PSNR for the two filters:

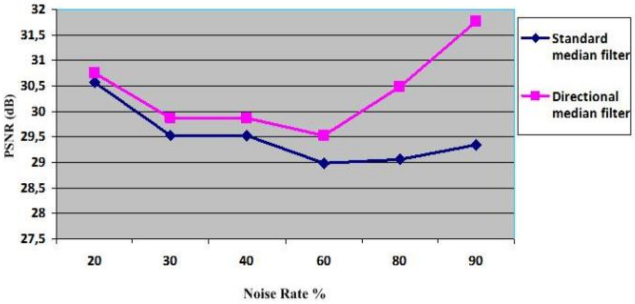


Figure 4 Variation of the value of the PSNR for both filters

Results discussion

Indeed, the above graph shows that the difference range between the two values in the same image and the same noise rate begins to widen even more once the noise rate reaches or exceeds 60%.

4. HARDWARE IMPLEMENTATION OF THE STANDARD AND THE DIRECTIONAL MEDIAN FILTER

In this part, we present the architecture, the simulation and the implementation of both median filters: Standard and directional median filters.

4.1 Description of the global architecture of the standard median filter

The global architecture of the standard median filter is composed of two different blocks which are:

1) Memory block.
2) Sort block.

The architecture of the memory block is represented by Fig.5:

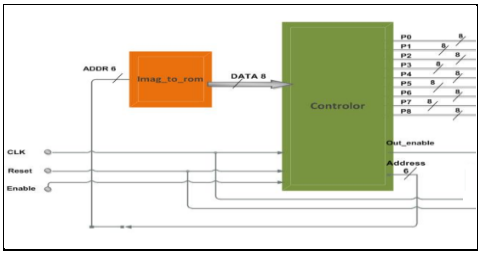


Figure 5 Architecture of the memory block

The memory block is constituted by:

1) RAM (Random Access Memory).
2) A controller.

The Sort block is realized by a set of comparators. A 3x3 sliding window algorithm is used as the base for filter operation, the sort is accomplished by using 36 comparators, according the following equation:

$$N = c_{ws}^2$$

C: combination; *N*: comparators numbers; *ws*: window size.

A comparison is made between two values (8 bits) to define the higher value and the lower value as shown in Figure 6.

At the output of the block, the nine pixels are sorted in descending order. So the value of the central pixel (P4) of the 3x3 matrix must be replaced by the value in the middle of the sorted values (Median). This operation is repeated for all pixels of the image.

Finally, the architecture which has been developed of the standard median filter is shown in Fig. 6:

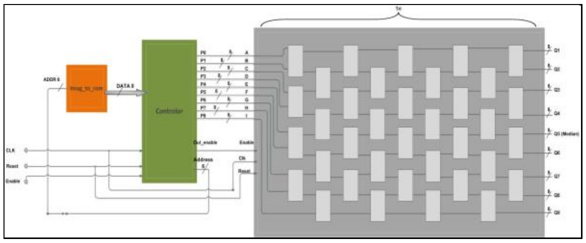


Figure 6 Standard median filter developed architecture

4.2 Description of the global architecture of the directional median filter

The block diagram of the filter is composed of five blocks defined as follows:

1) Block memory.
2) Calculation block of the directional indices and the impulse detector.
3) Calculation block of a standard deviation.
4) Calculation block of minimizer.
5) Calculation block of the median value.

Fig. 7 describes the global proposed architecture:

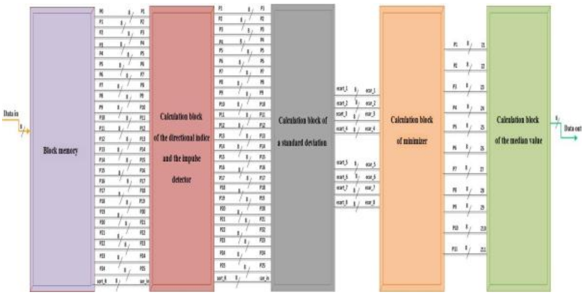

Data in
Block memory
Calculation block of the directional indice and the impulse detector
Calculation block of a standard deviation
Calculation block of minimizer
Calculation block of the median value
Data out

Figure 7 Block diagram of the directional median filter

The memory block is as the same as the previously described block; but, the operation to generate the pixels must be repeated 25 times to have at the output 25 pixels.

In the calculation block of the directional index and the impulse detector, we need 25 pixels as the block input (Direction); so, a 5x5 matrix to calculate the 4 indices direction outputs, these outputs are introduced at the input of the block (TRI), this block is realized by 6 comparators, according to the above equation. Each comparator must do a comparison between two directions indices to select the smallest direction which is the output that represents the value of the pulse detector, this value must be compared with a threshold to determine if the pixel is noisy or not.

The calculation block of a standard deviation calculates the standard deviation in four directions; so, we need 25 pixels at the input of block **(sous_ecart**). This block is realized by this equation:

$$\boldsymbol{sust} = [\mathbf{1/5} \sum_{(s,t)\,\epsilon S_k^0} (\boldsymbol{y_{i+s,j+t}} - \boldsymbol{mean^{(k)}})^2]$$

According to architecture below (Figure 8), **Sust_i** realize the direction i, (i= 1 to 4).

The second part is the block (**Squart_root**), this latter used core generator based on CORDIC algorithm which calculates the square root of the equation above.

The minimizer block determines the direction that represents the smallest standard deviation. When this direction is found we will take a matrix of 3X3 size centered on the pixel (i, j) and performs a duplicate operation of the two pixels of the selected direction.

In the calculation block of the median value, the sorting is accomplished through a number of comparators 5x5. Each comparator must make a comparison between two 8-bit values to define the maximum value and the minimum value.

At the end of this operation the eleven pixels are sorted in descending order. So the value of the central pixel of the 5X5 matrix must be replaced by the value located in the middle of the sorted values, and this operation is repeated for all pixels of the image.

Fig. 8 shows the architecture of a standard deviation:

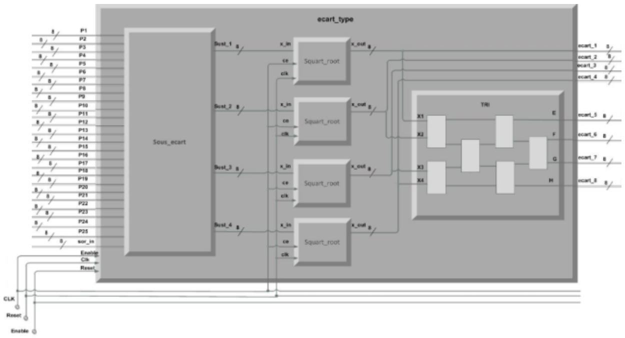


Figure 8 Standard deviation block

4.3 Comparison resources of the two architectures

The resources performances of the two architectures are shown in table.1:

Table 1 Comparison resources of the two median filters

	Number of Slice Registers	Number of Slice LUTs	Number used as logic	Number of occupied Slices	Number of bonded IOBs	Frequency (MHZ)	Execution times (s)
Median standard	514 (1%)	481 (1%)	481 (1%)	227 (3%)	83 (17%)	368.216	$1.755.10^{-7}$
Directional median	1.397 (4%)	1.563 (5%)	1.551 (5%)	601 (8%)	91 (18%)	107.202	$2.274.10^{-7}$

From the above comparisons, we notice that the standard median filter uses less execution time and few resources with satisfactory performance; however, directional median filter is preferable in terms of precision; despite the number of resources which is very large.

5. CONCLUSION

Image filtering is still a very large area of research. The objective of our work is devoted to the filtering of cerebral medical images from MRI « Magnetic Resonance Images ».

The image filtering is the heart of many problems in medical imagery because often it constitutes the first step of a real image processing flow of image depending on application selected.

We have implemented both architectures (i) standard and (ii) directional median filter under ISE tool of XILINX.

First step is devoted to the software implementation of these filters, under MATLAB tool, applied to MRI noisy. The results achieved at the simulation show the efficiency of the directional median filter algorithm.

The second step, we have presented the methodology of hardware design of the architectures developed of the standard and the directional median filters. The C5VLX50T 3FF1136-circuit of VIRTEX-5 family used for hardware implementation: Performances were justified in terms of the hardware resources used, and the execution time.

In order to integrate this work in a processing chain such as medical image segmentation, we must improve in the future:

The optimizing of the memory block to have fast access to the pixels stored in RAM. The other improvement consists of increasing of the calculation accuracy.

6. REFERENCES

[1] Sadri AR, Zekri M, Sadri S, Gheissari N, “Impulse noise cancellation of medical images using wavelet networks and median filters”, J Med Signals Sens. 2012 Jan;2(1):25-37.

[2] Toprak A, Güler I, “Suppression of impulse noise in medical images with the use of Fuzzy Adaptive Median Filter”, J Med Syst. 2006 Dec;30(6):465-71

[3] Sarala Singh, Prof. Ravimohan,"A VHDL Implementation of Effective Impulse Noise Filtration Technique", International Journal of Computer Technology and Electronics Engineering (IJCTEE), Volume 3, Issue 3, June 2013.

[4] Manju Chouhan, C.D Khare, "Implementation of Median Filter for CI Based on FPGA", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 12, December 2013, ISSN: 2278 – 1323.

[5] S.A. Fahmy, P.Y.K. Cheung W. Luk3, "High-throughput one-dimensional median and weighted median filters on FPGA", Published in IET Computers & Digital Techniques, ISSN 1751-8601.

[6] Hanifi Djamel, "Générateur d'architectures pour plate-forme reconfigurable dédié au traitement d'image", Thèse de Doctorat, Université d'Evry Val d'Essonne, Evry, France, 2000.

[7] Chikh Mohammed Tahar, "Amélioration des images par un modèle de réseau de neurones (Comparaison avec les filtres de base) ", thèse de Master en Informatique, 28 Septembre 2011.

[8] J.K. Mandal, AparnaSarkar, "A Modified Weighted Based Filter for Removal of Random Impulse Noise (MWB)", Second International Conference on Emerging Applications of Information Technology, 2011.

[9] J. K. Mondal, Somnath Mukhopadhyay" New Directional Weighted Median Filter for Removal of Random-Valued Impulse Noise", ICCS – 2010, November 19-20, 2010.