Research on Image Median Filtering Algorithm and Its FPGA Implementation

Yueli Hu Huijie Ji
School of Mechanical and Electronic Engineering, Shanghai University,
Key Laboratory of Advanced Display and System Applications (Shanghai University), Ministry of Education
Shanghai 200072, P.R. China
E-mail:huyueli@shu.edu.cn

Abstract

Image filtering plays an important role in image preprocessing. The median filters, including the standard median filter and the multi-level median filter, which can preserve image features and thin lines are introduced and discussed in detail. After that, the FPGA based solution for the algorithms of these two filters are presented. This paper also gives account to the FPGA implementation of the complete structure of a general filter including the filtering window generating module and the row-column counting module. RTL level simulation is performed in Modelsim to verify the functional correctness and system level simulation is performed in Matlab to compare the filtering effect.

1. Introduction

Due to the inherent defection of the imaging system, the transfer medium, the working conditions and the recording devices, images are subjected to noise corruption during its transmitting channels. As a result, the image quality is reduced and the effectiveness and accuracy of its subsequent processing courses (such as edge detect, image segmentation, feature extraction and pattern recognition) will be negatively affected. So it is necessary to remove the noise from the image using an image filter. But the effective removal of the noise is often accomplished at the expense of blurred or even lost features. Different filtering algorithms have been analyzed and compared in quite a few literatures and lots of improved algorithms have been put forward [1][2].

It is generally accepted that computational complexity required for the filtering applications is immense and complex. The complexity arises from the fact that it requires a huge amount of data to represent image information in digital format. Implementing such applications on a general purpose computer can be easier, but hardly time-efficient due to additional constraints on memory, I/O bandwidth and other peripheral devices. What's more, this solution definitely cannot meet the portability needs of these applications. Thanks to the parallelism and pipelining technologies, application specific hardware implementation can offer much greater speed than a software implementation. There are two types if technologies available for hardware design: full custom hardware design also well known as Application Specific Integrated Circuits (ASIC) and semi custom hardware devices, which are programmable devices like Digital Signal processors (DSPs) and Field Programmable Gate Arrays (FPGAs) [3]. The ASIC design can offer highest performance, but the complexity and the cost associated with the design are very high and these designs cannot be reconfigured. DSPs are well suited to extremely complex math intensive applications, but often face a constant struggle to find the balance between speed and generality. FPGAs are reconfigurable devices, and the parallelism and pipelining technologies can be implemented in it which is impossible in dedicated DSP designs. Implementing image processing algorithms on reconfigurable device minimizes the time-to-market cost, enables rapid prototyping of complex algorithms and simplifies debugging and verification. Moreover, the reprogrammable feature of FPGA devices provides the user with fast adaptation of system to current demands.

The paper is organized as follows. In section 2 we introduce the median filters: including the standard median filter and the multi-level median filter. In section 3 we present the details of median filter algorithms implemented in the FPGA and offer some simulation result and their resource utilization. In section 4 we provide the experiment results of these two median filters. And in section 5 we are to derive some conclusions and discuss the future work.

978-0-7695-3571-5/09 $25.00 © 2009 IEEE
DOI 10.1109/GCIS.2009.130

2. Filter algorithms description

In this section, we will firstly introduce the the standard median filter, and then we will proceed to the multi-level median filter, which has better performance in terms of edge-preserving.

2.1. Standard median filter

Standard median filter is one sort of non-linear filters. It sorts the pixels values in the neighborhood defined by the filter mask and then replaces the central pixel with the median value in the sorted group. Detailed discussions of the theoretical analysis of the median filters and their properties can be found in [4],[5].

With its fine detail preservation and impulsive noise removal characteristics, standard median filter has played an important role in image preprocessing applications. An important shortcoming of the filter is that its output is always constrained, by definition, to be one of the samples in the input window. We can see that in an $N\times N$ window, if the number of the polluted pixels is greater than $N(N+N)/2$, then the noise won't be removed. In such case, we have to increase the size of the window to improve the filtering efficiency. On the other hand, if the number of the pixels of edge features is smaller than $N(N+N)/2$, then the feature pixel will be replaced by other irrelevant pixels, and the features will be impaired. In such case, we have to reduce the size of the window to preserve image features. In order to strike a balance between noise removal and feature preservation, the size of the window has to be carefully chosen, and usually with contradictive effect.

2.2. Multi-level median filter

As an improvement of the standard median filter, Nieminen and Neuvo put forward the multi-level median filter [6].

Let $f(i,j)$ be the pixel value of the image at position (i,j). Let W present the filtering window of size $(2N+1)\times(2N+1)$ which is centered at position (i,j).

Define four sub-windows in W as follows:

$$\begin{cases} W1(i,j)=\{f(i+m,j),-N\le m\le N\} \\ W2(i,j)=\{f(i+m,j+m),-N\le m\le N\} \\ W3(i,j)=\{f(i,j+m),-N\le m\le N\} \\ W4(i,j)=\{f(i+m,j-m),-N\le m\le N\} \end{cases} \quad (1)$$

Let:

$$\begin{cases} Zk(i,j)=med[f(*,*)\in Wk(i,j)] \\ T1(i,j)=min[Zk(i,j),1\le k\le 4] \\ T2(i,j)=max[Zk(i,j),1\le k\le 4] \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Where:

$med[\ ]$, $min[\ ]$ and $max[\ ]$ respectively denotes the mediation operation, the minimization operation and the maximization operation of the elements in the square brackets. Then the output of the multi-level median filter is:

$$y(i,j)=med[T1(i,j),T2(i,j),f(i,j)] \ldots\ldots\ldots\ldots (3)$$

The substructures $Zk(i,j)$, k=1,2,3,4 are used to estimate features of different orientation from the image. The substructure $T2(i,j)$, whose maximization operation is used to combine the directional signal estimates, can be used for the images in which the geometrical information is on a lower valued background. Similarly, substructure $T1(i,j)$, whose minimization operation is used to combine the directional signal estimates, can be used for the images in which the geometrical information is on a higher valued background. And the final mediation operation $y(i,j)=med[T1(i,j),T2(i,j),f(i,j)]$ improves the noise attenuation of the substructures. Since the multi-level median filter takes into account geometrical information on both lower and higher valued background, it can much more effectively preserve the edge features of an image than the standard median filter.

3. FPGA implementation

FPGAs have usually been configured using hardware languages (HDL), so we must convert the algorithm description to hardware description. The image filter comprises of three modules: the filtering window generating module, the row-column counting module and the filtering operation module.

3.1. Filtering window generating module

Image preprocessing algorithms usually deal with neighboring pixels. In order to obtain the $N\times N$ window, a set of First In First Out (FIFO) dual-port memories are used to generate a column of the window. These vectors are then used to build the pixel rows. At the output we have the field memory to store the filtering window. A 3×3 window generating module is depicted in Figure 1.

The depth of the FIFO corresponds to the width of the input image, which means each FIFO stores one line of an image. The six registers are used to store the column pixels in each line. By this way, we can get

every filtering window at the rate of pixel clock. And the pixels in the filtering window are sent to the filtering operation module in a parallel mode.

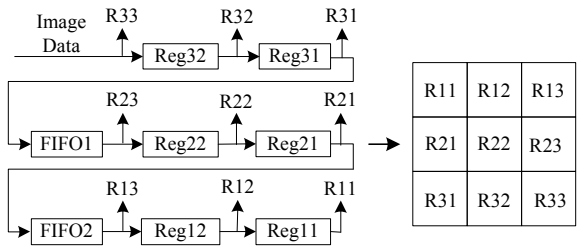
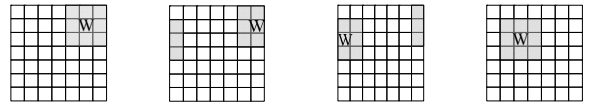


Figure 1. Window generating module

3.2. Row-Column counting module

As the filtering window moves across the image, the central pixel will move to the edge of the image and lead to form an invalid window.



a.window1 **b.window2** **c.window3** **d.window4**

Figure 2. Moving process of the filtering window

As shown in Figure 2, the central pixel in window2 and window3 are at the edge of an image, so these two windows are invalid.

We know that the edge of an image doesn't contain much useful information, so quite a few algorithms set the edge pixel value to zero. In order to mark the position of the window, a row-column counter is to be developed to calculate the current rows and columns of the window.

For an image with M rows and N columns, the row-column counter module is depicted in Figure 3.

In Figure 3, when the row value i belongs to $(2, M\text{-}1)$ and the column value j belongs to $(2, N\text{-}1)$, the output value is from the filtering operation module; otherwise the output is directly set to zero.

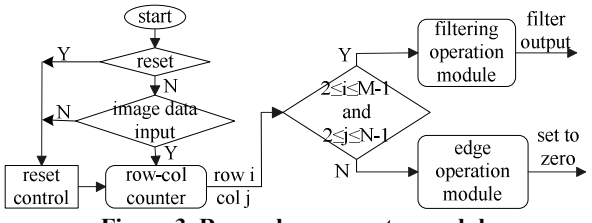


Figure 3. Row-column counter module

As for real-time video data, the row counter is disabled and reset during vertical blanking interval and the column counter is disabled and reset during horizontal blanking interval.

3.3. Filtering operation module

Despite of the same window generating module and the row-column counting module, the two filters employ different operation modules, whose FPGA implmetation will be described respectively in the following text.

3.3.1. Standard median filter. For 3×3 window, hardware implementation is depicted in Figure 4.

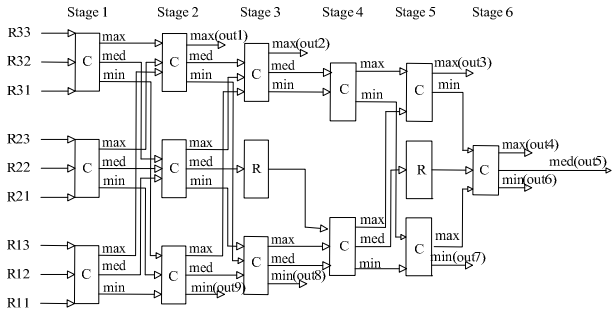


Figure 4. Structure of standard median filter

In Figure 4, the input data of the filter is the output of the filtering window generating module. The filter is composed of 6 stages of compare circuits. Stage 1 comprises of three tri-input comparators (C) whose outputs are organized in descending order. The maximum value of each comparator is put together as a group, and that is the same with the median value and the minimum value of each comparator. The three groups of data are the input of stage 2 which functions the same as the first stage. From this stage, we can get the maximum value out1 and the minimum value out9 of the original inputs. The two extrema are left aside and the rest are sent to stage 3 which comprises of two tri-input comparators and one register (R). Similarly, we get $out2$ and $out8$ as the maximum value and the minimum value of the seven input data, which are also left aside from the next stage. Stage 4, which comprises of one tri-input comparator and one bi-input comparator, takes the other five data of stage 3 as its inputs. The rest stages operate with the similar mechanism as the previous stages do. In this way, we can get the median value $out5$ of the nine data after six stages. The outputs $out1, out2, ..., out8, out9$ are the original input data in their descending order. It should be noted that in order to keep synchronization, the registers in stage three and stage five are employed to hold the data which don't participate in the compare operation of their respective stage.

3.3.2. Multi-level median filter. The multi-level median filter with a 3×3 window is composed of four stages of compare stage, as depicted in Figure 5.

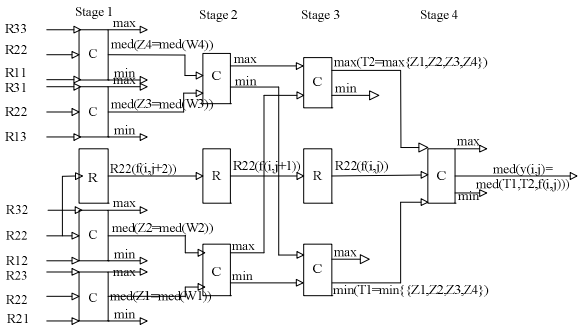


Figure 5. Structure of multi-level median filter

As shown in Figure 5, the first stage comprises of four tri-input comparators C and one register R. Each comparator takes into the data of a sub-window and yields the median value $Zk(i,j)$. The second stage as well as the third stage is comprised of two bi-input comparators and one register. The maximum value $T2$ and the minimum value $T1$ of $Zk, k=1,2,3,4$, which obtained from the third stage, are send to the fourth stage together with $f(i,j)$. The fourth stage has just one tri-input comparator and yields the resulting value of the filter. Registers in Figure 5 are of the same function as those in Figure 4.

3.4. Simulation results and resource utilization

3.4.1. Simulation results. The RTL level simulation is performed and verified by the Modelsim simulator. The result of the standard median filter and the multi-level median filter that derived from Modelsim is shown in Figure 6 and Figure 7.

The architecture of both median filters is pipelined. And after the pipeline delay, say six clock cycles for standard median filter and four clock cycles for multi-level median filter, we can obtain the output of each filter in every clock cycle. Thanks to the pipelined structure, the algorithms can be completed in a very short time compared with software realization, and is satisfactory to a real-time image processing system.

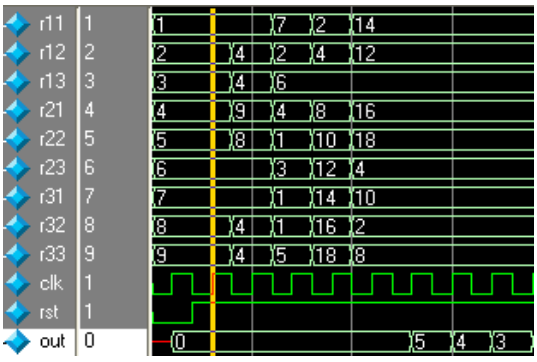


Figure 6. standard median filter

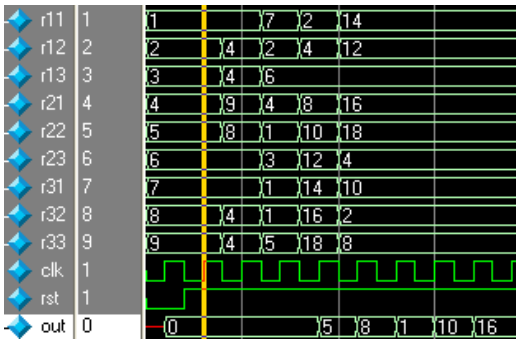


Figure 7. Multi-level median filter

3.4.2. Resource utilization. The FPGA implementation and test is achieved using Virtex-II Pro development board for xc2vp30ff896-6 device. Table 1 lists the resource utilization of the above two median filters. Table 1 shows that the standard median filter consumes nearly two times of the resources as the multi-level median filter does. The difference in the resource utilization is due to the algorithm complexity and the structural formation of each filter. As shown in Figure 4 and Figure 5, the former filter comprises of thirteen comparators and two registers while the latter one comprises of nine comparators and three registers.

Table 1. Utilization of FPGA resource

	Slices	*Slice Flip Flops*	*4 input LUT*
standard median filter	414	240	788
multi-level median filter	199	104	381

4. Experimental results

The two median filters are tested in the system level using Matlab in order to compare and analyze their effects. The experimental results are shown in Figure 8.

The image in Figure 8 is the test board of an 8-bit MCU chip developed by Shanghai University Microelectronics R&D Center. Figure 8 includes four sub-images, where a is the original image, b is the image polluted with 5% pepper and salt noise, c is the image processed with the standard median filter and d is the image processed with the multi-level median filter.Compare Figure 8.c and Figure 8.d we can make conclusions as follows.

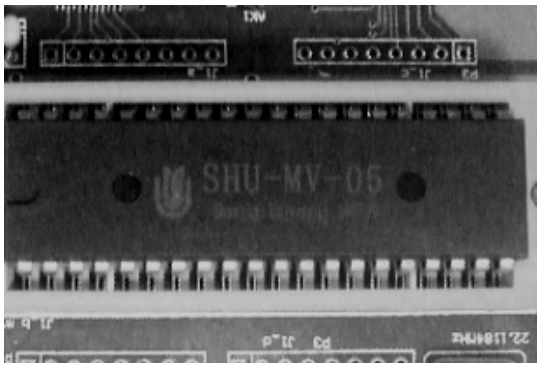
The standard median filter has removed the noise efficiently, but the whole image is blurred: symbol "22" at the right bottom becomes indistinct, and the via holes and the printed circuits are obscure.

The multi-level median filter preserves the image features quite well: the digital 22 at the right bottom

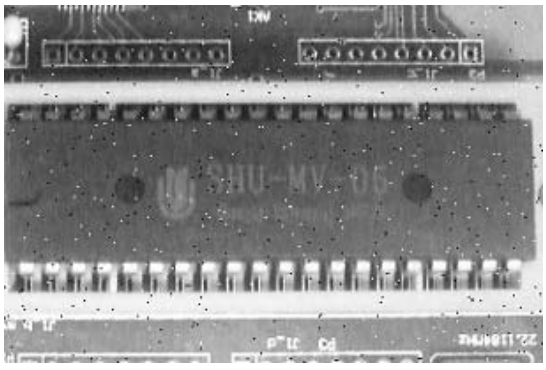
can be recognized easily, and the via holes and the printed circuits are clearer than that of the standard median filter. But the side effect is that the pepper and salt noise hasn't been totally removed, as there are still several white and black speckles left in the image.

5. Conclusions and future work

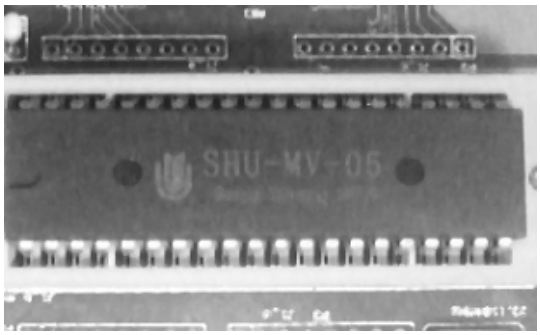
Image filtering is a necessary course in image processing. In order to effectively remove noise, image filters have to be carefully designed. Different filters have different effects and should be administered to specific application to achieve best result. After analyzing some filtering algorithms, an FPGA based solution for the image filter is presented. The solution is of general purpose and can be modified easily for various usages, that is to say, without changing the filtering window generating module and the row-column counting module, other neighboring algorithms such as edge detect and mathematics morphology can be performed in stead of the filtering algorithm. As the neighboring algorithms are performed in FPGA, the operation speed is greatly enhanced, which promises an extensive and significant usage of FPGAs in real-time image processing.



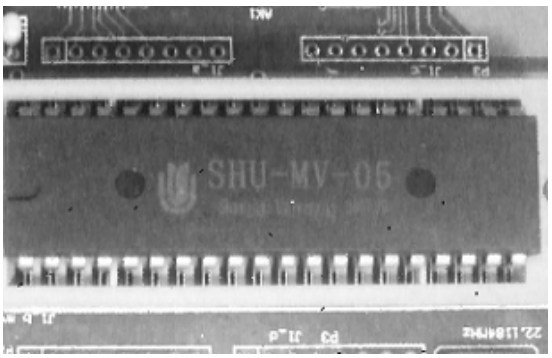
a. Original image



b. 5% Pepper and salt polluted image



c. Result of standard median filter



d. Result of multi-level median filter

Figure 8. Comparison between standard median filter and multi-level median filter

6. References

[1] Chang J J. Modified 2D median filter for impulse noise suppression in a real-time system. IEEE Transactions on Consumer Electronics, 1995, 41(1). pp. 73-80.

[2] NG P E and MA K K. A switching median filter with boundary discriminative noise detection for extremely corrupted images. IEEE Transactions on Image Processing, 2006, 15(6). pp. 1506-1516.

[3] Krupnova H and Saucier G. FPGA technology snapshot: current devices and design tools. Proceedings on Rapid System Prototyping, 21-23 June 2000. pp.200- 205.

[4] N C Gallagher and G L Wise. A theoretical analysis of the properties of median filters. IEEE Trans. Acoust., Speech, Signal Processing. vol.29, Dec. 1981. pp. 1136-1141.

[5] T A Nodes and N C Gallagher. Median filters: Some modifications and their properties. IEEE Trans. Acoust., Speech, Signal Processing. 30(5), Oct. 1982. pp. 739-7.

[6] Nieminen A and Neuvo Y. Comments on theoretical analysis of the Max/median filter. IEEE Trans. Acoustics, Speech, Signal Processing, 1988, 36(5). pp. 826-827.