Computing and Informatics, Vol. 38, 2019, 1320–1340, doi: 10.31577/cai_2019_5_1320

PARALLEL PEER GROUP FILTER FOR IMPULSE DENOISING IN DIGITAL IMAGES ON GPU

José Agustín Tortolero Osuna, Alberto Jorge Rosales Silva

Instituto Politécnico Nacional de México, ESIME Zacatenco
Av. IPN s/n, Edif. Z, Acceso 3, 3er Piso, SEPI-Electrónica
Col. Lindavista, 07738, CDMX, México
e-mail: arosaless@ipn.mx

Abstract. A new two-steps impulsive noise parallel Peer Group filter for color images using Compute Unified Device Architecture (CUDA) on a graphic card is proposed. It consists of two steps: impulsive noise detection, which uses a Fuzzy Metric as a distance criterion and a filtering step. For the needed ordering algorithm we are using the Marginal Median Filter with forgetful selection sort. Comparisons with other color filters for Graphics Processing Unit (GPU) architectures are presented, demonstrating that our proposal presents better performance in color preservation and noise suppression.

Keywords: Impulse denoising, color images, CUDA, parallel architecture, fuzzy metrics

1 INTRODUCTION

The growing demand for digital image processing algorithms for applications, i.e. in consumer electronics, industry, and medicine, creates the need to develop algorithms to be able to perform itself quickly and executed accordingly to the needs of each application field. All processing algorithms must possess, either hardware or software, denoising procedures to improve the image information obtained from the acquisition hardware device, by doing this the information provided is more significant for other post-processing stages (i.e. face recognition, augmented reality, computer vision, etc.). Noise affects the majority of image processing systems, being responsible for the 99 % of the failures [1, 2, 3, 4]. This clearly justifies the use of noise-suppressing algorithms incorporated within the systems. The incorporation

of a denoising algorithm as a pre-processing stage consumes hardware and software resources which take up processing time affecting other post-processing procedures.

Decades ago most of the designed filters were carried out serially, today the current processing needs lead to optimize resources and make processing times more efficient [5]. This points to parallel implementations; this type of processing leads to substantially improving computational time in all stages of an image processing system. In this way, it is proposed to exploit the potentials of parallelism and use them for noise suppression. Between the noise types that affect the digital images, the impulsive noise is one of the most important, this affects digital images in the acquisition or transmission stages and through malfunctioning sensors or communication channels [2, 3, 4], this is produced by human-made phenomena, such as car ignition systems, industrial machines in the vicinity of the receiver, switching transients in power lines, unprotected switches, natural causes, etc. Impulsive noise has been treated using parallel architectures, e.g. the Peer Group Family filters [6], presenting acceptable results preserving inherent characteristics of the digital images. Their main features are that they are fast and have low computational complexity, these characteristics are used in parallel systems for impulse noise removal.

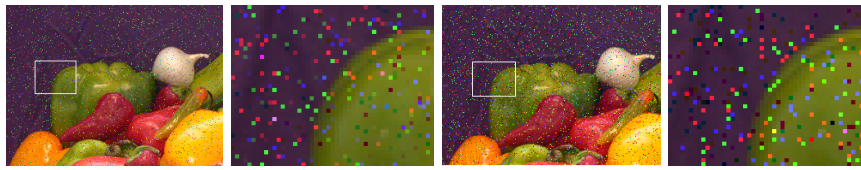
So, our efforts are focused on the design of a more robust and fast parallel filter for GPU architectures filter, that preserves inherent characteristics of the objects such as edges, details and chromatic content. We demonstrated the improved results against obtained from other states of the art algorithms reported such as the Peer Group Algorithm with Fuzzy Metric for GPU [6, 7, 8, 9].

Impulse noise in color images is modeled with random and fixed values, if $F^{\chi}_{(x,y)}$ denotes the input noisy image with χ denoting each color channel $\{R, G, B\}$, and $O^{\chi}_{(x,y)}$ denoting the original noise-free image for every (x, y) pixel position [2], this is described by Equation (1):

$$F^{\chi}_{(x,y)} = \begin{cases} O^{\chi}_{(x,y)}, & \text{with probability } 1-p, \\ \zeta^{\chi}_{(x,y)}, & \text{with probability } p, \end{cases} \tag{1}$$

where p is defined as the noise density for the random value of the impulsive noise, $\zeta^{\chi}_{(x,y)}$ is an identically distributed, independent random process with an arbitrary underlying probability density function, having values of impulsive noise in the interval $[0, \ldots, 255]$, and for the fixed values (Salt and Pepper noise) the values present are denoted by 0 or 255. In Figure 1, the corrupted Peppers image with random and fixed noise values are shown.

The structure of the paper is as follows: in Section 2 a brief review of the GPU architecture is presented; in Section 3 the Peer Group Algorithm is introduced using fuzzy metric to justify the novelty of our proposal; Section 4 describes the proposed filtering algorithm in a detailed manner; in Section 5 a comparative analysis of the read/write memory accesses between the proposed and the state-of-the-art filters is done. Finally, Sections 5 and 6 show the performance results.



a) b) c) d)

Figure 1. Peppers images with: a) random impulsive noise, c) salt and pepper impulsive noise, b) and d) the zoomed regions to denote impulsive distortions, respectively

2 GPU ACCELERATION CONSIDERATIONS FOR IMAGE PROCESSING

One of the main differences between CPU and GPU is the number of working threads. CPU can execute only up to two threads per core, while GPU chips can run thousands. Figure 2 shows an overview of the GPU card hardware consisting of Graphics Double Data Rate (GDDR) Memory (Global, Constant and Texture), Streaming Multiprocessors (SMs), and Streaming Processors (SPs) [10, 11].

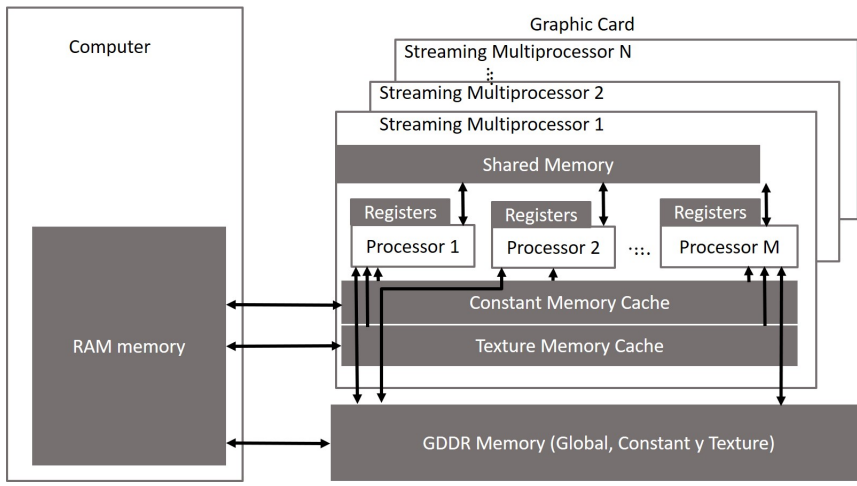


Figure 2. The GPU hardware [10]

The SPs are responsible for the mathematic operations done in the GPU chip. SMs are made of several Streaming Processors, up to 48 in modern GPUs; this is a crucial aspect for GPU scaling. A GPU has one or more SMs, where each has only one control unit; this unit seeks and decodes each instruction. The SMs on the GPU uses the Single Instruction Multiple Data (SIMD) architecture and the SP receives the same controlling signal and executes the same instruction set using different operands [12]. The optimum performance of parallel algorithms in graphic

cards lies in the understanding of how memory devices work [12], what care should be taken in the type of memory where data is temporarily stored [13], because a bad memory selection could produce poor parallel performance, i.e., the off-chip memory offers more storing space and global scope but its writing and reading latency is high.

Another consideration is with respect to the registers which allow memory access without latency, unfortunately there are only 255 registers per thread used in the GPUs Maxwell architecture and shared memory (which is faster than off-chip memory). This memory is slower than register memory, it should be taken into consideration that bank conflicts can occur if two threads in the SM try to access data stored in the shared memory, in such cases, the parallel process is serialized, which may cause a significant performance decrease [12, 13]. For this reason, it is justifiable to take into account the selection of the correct memory to be used. The GPU graphic card has its own Graphics Double Data Rate (GDDR) memory where read/write Global, read-only Constant and Texture memories are allocated; the last two memories are used for speeding up the reading from the GDDR memory. The register memory is the fastest in the graphic card, but it is limited in quantity (i.e., on Fermi each SMs has 32 k memory addresses). Shared memory is the second-fastest memory and can be read by all the threads within the SM. The Global memory is for general purposes and is the slowest memory in the graphic card. These facts are summarized in Table 1.

Memory Type	Registers	Shared Memory	Texture Memory	Constant Memory	Global Memory
Bandwidth	Highest	High	Low	Low	Low
Location	On-chip	On-chip	Off-chip	Off-chip	Off-chip
Scope	Thread	Threads in SM	All SMs	All SMs	All SMs

Table 1. Graphic card memory overview

CUDA programming model is the tool to be used to exploit parallelism capabilities of the GPU card, this tool introduces three key abstractions [14]: a hierarchy of thread groups, shared memories, and barrier synchronization; these abstractions are presented to the programmer as a minimal set of language extensions. This decomposition preserves language expressivity by allowing threads to cooperate when solving each sub-problem, and at the same time enables automatic scalability. Each block of threads can be scheduled concurrently or sequentially on any of the available multiprocessors within a GPU, so a compiled CUDA program can execute on any number of multiprocessors, and only the runtime system needs to know the physical multiprocessor count. A CUDA-enabled program, called "kernel", can be executed in any number of SMs, as illustrated by Figure 3 [14].

Graphic cards can solve time-consuming image processing algorithms specifically in image denoising. Our proposal offers good quality results exploiting new tech-

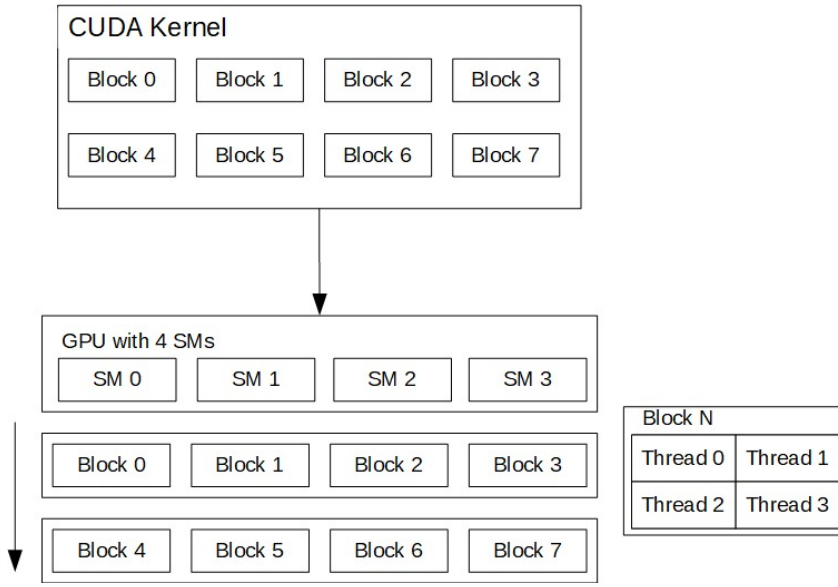


Figure 3. CUDA thread hierarchy

nologies capabilities and delivers optimum data pre-processed (data denoised) to other post-processing stages described before. Now in the next sections, we analyze, treat and discuss the proposal how the denoising algorithm is to be implemented in the GPU and its main characteristics to be taken into account to speed up the denoising stage.

3 PEER GROUP ALGORITHM

The Peer Group algorithm is a robust filter consisting in a processing window of $W = n \times n$ over a central pixel P_c^χ, the Peer Group $\mathcal{P}(P_c^\chi, \kappa, d)$ of such central pixel are the pixels satisfying $d(P_c^\chi, P_j^\chi) \leq \alpha$ where $d(P_c^\chi, P_j^\chi)$ is an appropriate distance criterion between P_c^χ and one of its neighbors, P_j^χ, $j = n^2 - 1$, where n is the size of the processing window, with $n = 3$, and α is a fixed threshold [6]. The distance criterion used is the Euclidean distance described by Equation (2):

$$d(P_c^\chi, P_j^\chi) = \sqrt{\sum_{\chi=\{R,G,B\}} \left(P_c^\chi - P_j^\chi\right)^2}. \tag{2}$$

For every pixel within the processing window, if $d(P_c^\chi, P_j^\chi) \leq \alpha$ is fulfilled, then the pixel is labeled as a noise-free pixel, otherwise it is noisy. In the filtering stage, the noisy pixels are substituted by the substitute obtained from the Arithmetic Mean Filer (AMF) [7] or the Vector Median Filter (VMF) [15] applied to the set

of free-noisy pixels. Sanchez et al. [7, 8, 9], propose the Peer Group algorithm with Fuzzy Metric (PGMF) for GPU architectures, where each stage is executed in separated kernels in serial sequence. This algorithm uses the Fuzzy Metric shown in Equation (3) [7, 8, 9, 16] instead of the Euclidean distance in the detection stage.

$$M_\infty = \prod_{\chi=1}^{3} \frac{\min(P_c^\chi, P_j^\chi) + K}{\max(P_c^\chi, P_j^\chi) + K}, \text{ with } K = 1\,024. \tag{3}$$

The kernel algorithm for detection and suppression of noise is shown in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 PGMF algorithm

```
1: function DetectionKernel ▷ Input: noisyImage ▷ Output: detectedNoise
2:     for each image pixel P_th its corresponding processing parallel thread th do
3:         a_c^χ ← {R,G,B} values of P_th from the global memory
4:         for all the P_j^χ pixels within W do
5:             b_j^χ ← {R,G,B} values of P_j^χ
6:             distance = Fuzzymetric(a_c^χ, b_j^χ)
7:             if distance ≥ α then
8:                 pixel b_j^χ ∈ P(P_c^χ, κ, α)
9:             end if
10:        end for
11:        if #P(P_c^χ, κ, α) ≥ (κ + 1) then
12:            P_th is labeled as noise free pixel
13:        else
14:            P_th is labeled as noisy pixel
15:        end if
16:    end for
17: end function
```

Algorithm 2 PGMF algorithm

```
1: function NoiseFilteringkernel ▷ Input: noisyImage ▷ Output:
   filteredImage
2:     for each image pixel P_th its corresponding processing thread th do
3:         for all the P_j^χ labeled as noise free pixels within W do
4:             FilteringArray_j^χ ← {R,G,B} values of P_j^χ
5:         end for
6:         Filter FilteringArray_j^χ use either the VMF or the AMF
7:     end for
8: end function
```

4 PROPOSED MODIFIED PEER GROUP FUZZY METRIC FILTER (MPGFMF) FOR GPU ARCHITECTURES

Our peer group filter proposal was designed by executing only one kernel instead of two as the filter described in Section 3. The noise detection stage is used to know if the central pixel is a noisy one or not, if the central pixel is noisy, the filtering stage uses all the pixels within W to obtain a denoised one. The main idea behind this is to reduce the read/write count to the GDDR memory outperforming the computational cost from the work reported by Sanchez et al. [7, 8, 9], in addition, to improve the filtering stage, it is suggested to use the Median Filter (MF) with forgetful selection sort [17] applied for each color channel. The forgetful selection sort algorithm illustrated in Figure 4 consists of four iterations for a window processing W with $n = 3$ for each RGB channel; in the "Iteration 1", from the first six of the nine elements array the minimum and the maximum intensity values are obtained, then they are discarded of the array; in the "Iteration 2" the max and min values are computed again of the remaining four pixels adding the seventh element of the original array. These steps are repeated until the final "Iteration 4" where the mean pixel intensity is obtained value for every color channel. Along with these ideas, the proposed filter takes the advantage of the register memory that resides inside the GPU chip instead of the slowest GDDR memory [17, 8] used in the related work. In the kernel Algorithm 3, the proposed Modified Peer Group Fuzzy Metric Filter (MPGFMF) is shown. In Section 5, a memory cost analysis and a quality comparison between the proposal and reported filters in the literature are presented.

5 MEMORY COST ANALYSIS AND COMPARATIVE

A memory cost comparison between the state-of-the-art method PGMF and the proposed MPGFMF filtering algorithm considers the computational cost in the detection stage, so the number of accesses C_D is delivered in Equation (4), in the filtering stage, the computational cost is measured denoting C_{NF} (noise free) and C_N (noisy) described in Equation (5) and (6), respectively, considering that the PGMF filter computational analysis encompasses two cases where the pixel is noisy or free of noise.

$$C_D = \chi \cdot n^2, \tag{4}$$

$$C_{NF} = 1 + 2 \cdot \chi, \text{ noise-free pixel}, \tag{5}$$

$$C_N = 1 + (n^2 - 1) + \chi \cdot (n^2 - 1) + \chi, \text{ noisy pixel}. \tag{6}$$

For the PGMF method, $n = 3$ and $\chi = 3$ are taken, the noise-free case requires 35 reading and writing accesses for detection and filtering, and the noisy pixel requires 61 reading and writing accesses for the detection and filtering, both for the GDDRAM accesses. For our proposal, the same parameter values for n and χ are proposed resulting in 30 reading and writing accesses for the detection and filtering

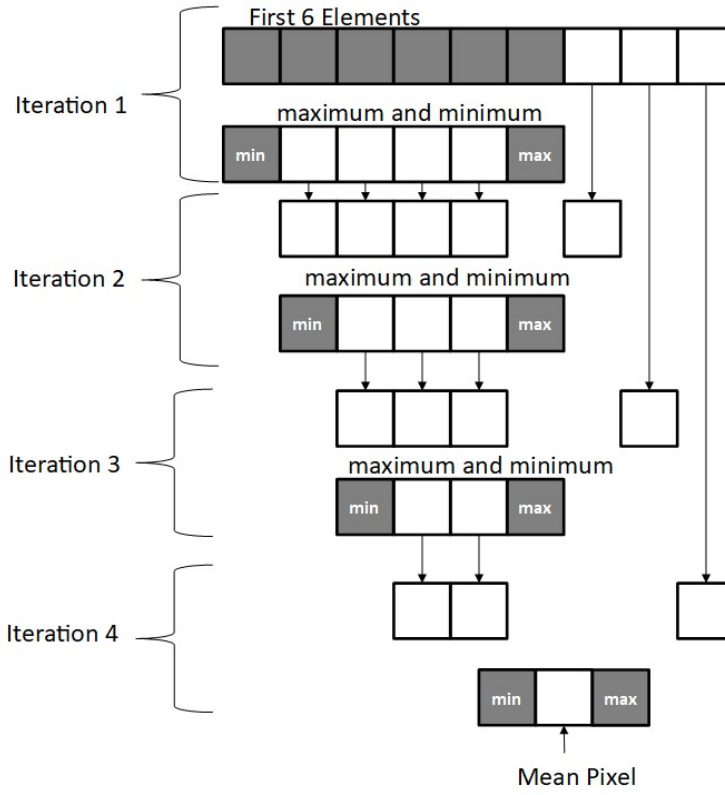


Figure 4. Forgetful selection algorithm

stage, also for the GDDRAM accesses. All the pixels within W for the proposed filtering algorithm are considered. Carrying out the count of readings and writings to the GDDR memory outperforms other parallel filters. The detection stage of the proposed filter needs:

$$C_D = \chi \cdot n^2. \tag{7}$$

For the filtering stage, it is proposed to reuse the memory in the detection stage, because one kernel is used and all the used variables in the present thread are still available for us. The difference from the PGMF filter which has two kernels and local variables and registers is that the kernels are scope-only (from Table 1). So, the proposed filter requires the same amount of readings and writings accesses no matter if the pixel within W is noisy or not, only needing χ writings for the pixel resulting from the filtering process, computed as in Equation (8):

$$C_N = C_{NF} = \chi, \tag{8}$$

Algorithm 3 Proposed MPGFMF

```
function MPGFMF                ▷ Input: noisyImage        ▷ Output: filteredImage
    for For each image pixel P_th its corresponding processing thread th do
        a_c^χ ← {R,G,B} values of P_th from global memory
        for all the P_j^χ pixels within W do
            b_j^χ ← {R,G,B} values of P_j^χ
            distance = Fuzzymetric(a_c^χ, b_j^χ)
            if distance ≥ α then
                pixel b_j^χ ∈ P(P_c^χ, κ, α)
            end if
        end for
        if #P(P_c^χ, κ, α) ≥ κ + 1 then
            P_th is labeled as noise free pixel
        else
            P_th is labeled as noisy
        end if
        if P_th is labeled as noisy pixel then
            Filter P_th all the pixels within W using the MMF with forgetful selection sort
        else
            P_th is the output pixel
        end if
    end for
end function
```

when $n = 3$, and $\chi = 3$, requires 30 readings and writings to the GDDRAM for "noise-free" and "noisy" cases. The following section shows the experimental results of the proposal and the method used as a comparison.

6 EXPERIMENTAL RESULTS

Here are compared the filters found in the scientific literature against our proposal using objective parameters. The objective parameters are defined as Peak Signal to Noise Ratio (PSNR) [18]:

$$PSNR = 10 \cdot \log \left[\frac{255^2}{MSE} \right] \tag{9}$$

where

$$MSE = \frac{1}{3 \times M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\left(R_{x,y}^{O} - R_{x,y}^{F} \right)^2 + \left(G_{x,y}^{O} - G_{x,y}^{F} \right)^2 + \left(B_{x,y}^{O} - B_{x,y}^{F} \right)^2 \right] \tag{10}$$

where $R^O_{x,y}$, $G^O_{x,y}$ and $B^O_{x,y}$ are the color components of the original image and $R^F_{x,y}$, $G^F_{x,y}$ and $B^F_{x,y}$ are the color components of the filtered one.

The Mean Chromaticity Error (MCRE) [19] is defined as follows:

$$MCRE = \frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \mu \left[O_{x,y}, F_{x,y} \right] \tag{11}$$

where $\mu[O_{x,y}, F_{x,y}]$ is the PP' distance in the Maxwell triangle between the pixel with coordinates x, y within the original image $O_{x,y}$ and the filtered pixel with coordinates x, y within the filtered one $F_{x,y}$.

The Normalized Color Difference (NCD) described in [20] is defined in Equation (12), where $L^O_{x,y}$, $u^O_{x,y}$, $v^O_{x,y}$ and $L^F_{x,y}$, $u^F_{x,y}$, $v^F_{x,y}$ are the values of the perceived lightness and two chrominance values related to $O_{x,y}$ and $F_{x,y}$, respectively.

$$NCD = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \sqrt{(L^O_{x,y} - L^F_{x,y})^2 + (u^O_{x,y} - u^F_{x,y})^2 + (v^O_{x,y} - v^F_{x,y})^2}}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \sqrt{(L^O_{x,y})^2 + (u^O_{x,y})^2 + (v^O_{x,y})^2}}. \tag{12}$$

Structural Similarity [21] (SSIM), which characterizes the preservation of the structures of the image is given by:

$$SSIM = \frac{(2\mu_O \mu_F + C_1)(2\sigma_{OF} + C_2)}{(\mu_O^2 + \mu_F^2 + C_1)(\sigma_O^2 + \sigma_F^2 + C_2)} \tag{13}$$

where μ_O and μ_F are the averages of $O_{x,y}$ and $F_{x,y}$, respectively. σ_O^2 and σ_F^2 are the variances of the two images. σ_{OF} is the covariance of $O_{x,y}$ and $F_{x,y}$. $C_1 = (0.01L)^2$ and $C_2 = (0.03L)^2$ are two constants and $L = 2^8$ is the dynamic range of the pixel values.

The visual signal-to-noise ratio (VSNR) [22, 23] characterizes the Visual Fidelity of images taking into consideration the human visual system. These criteria consist of 3 stages, namely Preprocessing, Assess the Detectability of the Distortions and Compute the VSNR, where:

$$VSNR = 20 \cdot \log_{10} \left(\frac{C(O)}{\alpha d_{pc} + (1 - \alpha)(d_{gp}/\sqrt{2})} \right) \tag{14}$$

where $C(O)$ is the contrast distortion of the original image, d_{pc} is the contrast perceived of the distortions obtained from the filtered image respect to the original one, d_{gp} is the disruption calculation of global precedence and α is a suggested weight, in our case $\alpha = (0.04)^4$ [22].

Comparison of the filtering methods is achieved using the PC computer – its characteristics are listed in Table 2.

The test color images used are the well known "Lena" and "Mandrill", shown in Figure 5 a) and 5 b), respectively, the spatial resolution size is of 512×512. These

Characteristics	Value	Characteristics	Value
CPU	Intel Core i7-8700K @ 3.70 GHz	GPU	Nvidia GeForce RTX 2080
CPU architecture	Coffee Lake	GPU architecture	Turing
Cache	12 Mb	SMs	46
Core count	6 cores	SPs in each SM	64
System Memory	16 Gb DDR4	Memory	8 Gb GDDR6
OS	Microsoft Windows 10, 64-bit		

Table 2. Characteristics of the system used in the CPU and GPU

images were resized from 256×256 using Matlab, leaving grayscale images with intensity values from 0 to 255, and with 24-bit color images per pixel. The resized versions of the images of $1\,024 \times 1\,024$, $2\,048 \times 2\,048$, $4\,096 \times 4\,096$ and $8\,192 \times 8\,192$ pixels were used for the runtime tests. We also tested the performance of the filters using a 3D and a 2D image shown in Figures 5 c) and 5 d), both 8 bit grayscale images. The 3D grayscale image is a Magnetic Resonance Imaging (MRI) scan, with dimensions $256 \times 320 \times 192$ and was obtained from the dataset of I Do Imaging (file 1010_brain_mr_06.nii.gz) found in [24] in the Neuroimaging Informatics Technology Initiative NIfTI-1 Data Format. We used the Digital Imaging and Communications in Medicine (DICOM) Converter version: 1.10.5 from DICOM apps in order to obtain bitmap images for testing. For the PSNR, SSIM and VSNR experiments, we used the image number 96 of the 3D set. The 2D grayscale image is a computed tomography (CT) mammography scan of size $1\,024 \times 1\,024$ obtained from the mini-MIAS database of mammograms (file mdb001.pgm) [25]. Furthermore, the filter proposal was tested against a state-of-the-art Deep Learning Filter (DLF) for impulse noise found in [26]. The DLF methods are very promising techniques to be used in the digital image denoising, the efforts until now are focused in grayscale denoising because a lot of computational resources needs to be used. The quality results obtained from DLF are impressive, but they spend a lot of computational time, as shown in [26, 27]. The results for the PSNR and SSIM for the test grayscale images of 512×512 pixels are shown for the Lena, Barbara, Boat, and 256×256 Cameraman. Besides, the runtime is given as a criterion for the GPU.

7 DISCUSSION

It is demonstrated that our proposal has better numerical results compared with the other state-of-the-art GPU filters for the Mandrill image in all the noise density interval. In Tables 3 and 4 there are shown the denoising results of the Mandrill image corrupted with Random-value and Salt and Pepper impulsive noises from 0 through 40 %, with incremental steps of 5 %; evaluation criteria used are PSNR, MCRE, NCD, SSIM, and VSNR. It is possible to see that our filter outperforms in

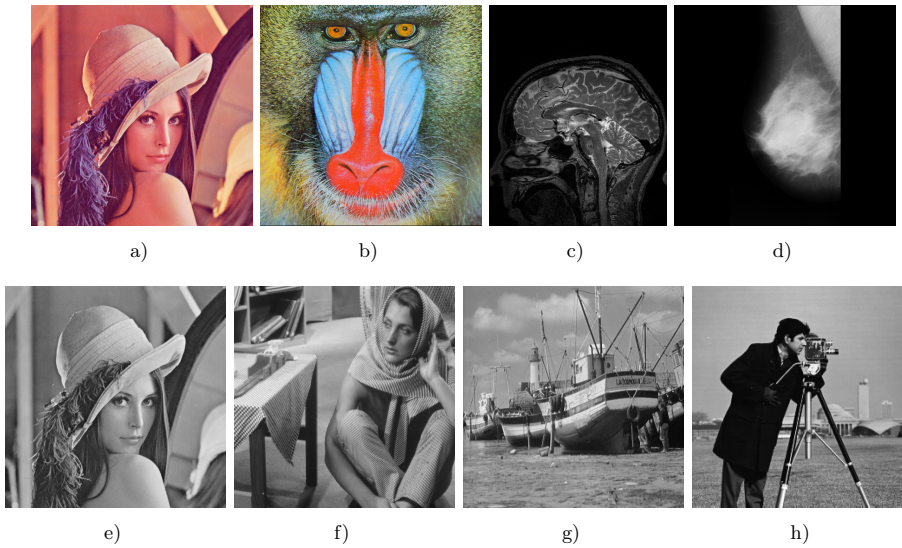


Figure 5. Digital Images used: a) Lena, b) Mandril c) Brain, d) Mammography, e) Lena Grayscale, f) Barbara, g) Boat, and h) Cameraman

the whole noise interval (0 to 40 %). As the noise density increases, it is possible to note that the difference in the numerical results increases, i.e. in Table 4, for 0 % $PGMF_{VMF}$ has a PSNR = 32.33 dB, $PGMF_{AMF}$ has a PSNR = 32.6 dB and our filter 32.71 dB, for 40 % we can see that $PGMF_{VMF}$ has a PSNR = 18.44 dB, $PGMF_{VMF}$ has a PSNR = 16.68 dB and our filter 19.51 dB.

In Table 5, the PSNR, MCRE, NCD, SSIM, and VSNR results concerning the Lena image corrupted with Random-value and Salt and Pepper impulsive noise from 0 through 40 % with steps of 5 % are presented. You can see in Table 5 that the proposed filter has similar good performance in the PSNR, MCRE, NCD, SSIM and VSNR criteria, outperforming the GPU filters ($PGMF_{\text{VMF}}$ and $PGMF_{\text{AMF}}$), that is, for 10 % we obtained by means of our filter proposal PSNR values of 26.11 dB, while $PGMF_{\text{VMF}}$ = 25.99 dB and $PGMF_{\text{AMF}}$ = 25.98 dB. In Table 6, there are shown similar results, that is, for example, by our proposal for 10 % of Salt and Pepper noise a PSNR result of 29.26 dB is obtained, while for comparative filters $PGMF_{\text{VMF}}$ = 25.38 dB and $PGMF_{\text{AMF}}$ = 25.42 dB were obtained in PSNR criterion, and so on for all the other criteria implemented.

For the Brain image, the PSNR, SSIM, and VSNR criteria results for the proposal and comparative filters using Salt and Pepper and Random-value impulsive noise are presented. The proposal presents better results for the criteria from 0 through 40 % with steps of 5 % of noise density, as seen in Tables 7 and 8. In Table 7 the best results for the SSIM criterion can be perceived, for example, for 20 % of Random-value impulsive noise for MPGFMF the $SSIM = 0.571$ was obtained,

	PGMF$_{\text{VMF}}$					PGMF$_{\text{AMF}}$					MPGFMF				
%	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR
0	32.33	0.143	0.000	0.971	12.459	32.6	0.127	0.000	.972	12.399	**32.71**	**0.117**	0.000	**0.987**	**24.017**
5	26.82	4.415	0.006	0.909	12.887	27.12	4.241	0.006	0.912	12.839	**27.37**	**4.239**	**0.005**	**0.930**	**22.318**
10	24.42	8.261	0.011	0.849	12.656	24.75	**7.967**	0.011	0.854	12.622	**25.05**	7.977	**0.010**	**0.875**	**20.485**
15	22.78	11.860	0.017	0.790	12.351	23.12	11.489	0.016	0.796	12.355	**23.46**	**11.475**	**0.015**	**0.822**	**18.559**
20	21.59	15.254	0.021	.735	11.815	21.9	14.850	0.021	0.742	11.824	**22.29**	**14.766**	**0.019**	**0.773**	**17.106**
25	20.66	18.426	0.026	0.684	11.392	21.00	17.979	0.026	0.693	11.212	**21.43**	**17.800**	**0.023**	**0.729**	**15.997**
30	19.81	21.492	0.031	0.636	10.806	20.15	20.934	0.031	0.646	10.588	**20.71**	**20.569**	**0.027**	**0.689**	**14.872**
35	19.09	24.556	0.035	0.589	10.292	19.39	23.846	0.03	0.598	9.956	**20.05**	**23.310**	**0.031**	**0.649**	**14.067**
40	18.44	27.459	0.039	0545	9.795	18.68	26.479	0.042	0.551	9.128	**19.51**	**25.840**	**0.034**	**0.614**	**13.132**

Table 3. Criteria results for the Mandril image with random-value impulsive noise

	PGMF$_{\text{VMF}}$					PGMF$_{\text{AMF}}$					MPGFMF				
%	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR
0	32.33	0.143	0.000	0.971	12.459	32.6	0.127	0.000	0.972	12.399	**32.71**	**0.117**	0.000	**0.987**	**24.017**
5	26.16	4.306	0.007	0.901	12.779	26.65	3.963	0.007	0.907	12.765	**27.15**	**4.000**	**0.006**	**0.929**	**21.650**
10	22.81	9.0551	0.015	0.805	12.161	23.3	**8.500**	0.015	0.816	12.116	**24**	8.505	**0.013**	**0.852**	**18.725**
15	20.56	14.1079	0.024	0.717	11.204	20.94	13.406	0.024	0.717	10.874	**22.02**	**13.051**	**0.019**	**0.775**	**16.588**
20	18.64	20.363	0.035	0.596	10.121	18.64	19.077	0.041	0.596	9.039	**20.53**	**17.863**	**0.026**	**0.698**	**14.905**
25	16.81	28.405	0.047	0.483	9.176	16.31	25.439	0.007	0.455	6.866	**19.22**	**22.968**	**0.032**	**0.625**	**13.622**
30	15.13	38.836	0.064	0.379	8.166	14.05	31.293	0.129	0.318	4.607	**18.18**	**28.054**	**0.038**	**0.559**	**12.458**
35	13.44	52.492	0.086	0.279	7.278	11.95	36.951	0.231	0.198	2.894	**17.01**	**34.165**	**0.046**	**0.484**	**11.096**
40	12.03	66.670	0.670	0.204	6.667	10.37	40.575	0.378	0.120	2.168	**15.91**	**40.850**	**0.054**	**0.417**	**9.932**

Table 4. Criteria results for the Mandril image with Salt and Pepper impulsive noise

	PGMF$_{VMF}$					PGMF$_{AMF}$					MPGFMF				
%	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR
0	29.25	0.006	0.000	0.963	13.251	29.26	0.006	0.000	.963	13.255	**29.26**	**0.004**	0.000	**0.983**	**21.11**
5	27.48	2.638	0.003	0.902	13.518	27.46	2.647	0.003	0.882	13.395	**27.54**	**2.570**	**0.003**	**0.901**	**21.49**
10	25.99	5.184	0.007	0.822	13.541	25.98	5.318	0.007	0.807	13.469	**26.11**	7.977	**.006**	**0.826**	**21.11**
15	24.63	7.665	0.107	0.724	13.455	24.63	8.043	0.011	0.733	13.409	**24.79**	**7.553**	**0.010**	**0.752**	**20.33**
20	23.55	9.969	0.014	0.611	13.017	23.57	10.652	0.014	0.673	13.249	**23.80**	**9.837**	**0.0133**	**0.693**	**19.61**
25	22.58	12.352	0.017	0.494	12.043	22.57	13.440	0.019	0.616	12.867	**21.43**	**12.238**	**0.167**	**0.637**	**18.68**
30	21.72	14.790	0.021	0.379	11.041	21.71	16.242	0.023	0.568	12.487	**22.05**	**14.668**	**0.0201**	**0.588**	**17.81**
35	20.96	17.139	0.024	0.266	9.624	20.92	18.859	0.027	0.521	12.000	**21.42**	**16.842**	**0.234**	**0.547**	**17.03**
40	20.19	19.770	0.028	0.180	8.363	20.10	21.577	0.032	0.479	11.310	**20.76**	**19.159**	**0.026**	**0.509**	**16.10**

Table 5. Criteria results for the Lena image with Random-value impulsive noise

	PGMF$_{VMF}$					PGMF$_{AMF}$					MPGFMF				
%	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR	PSNR	MCRE	NCD	SSIM	VSNR
0	29.25	0.006	0.000	0.963	13.251	29.26	0.006	0.000	0.963	13.255	**29.26**	**0.004**	0.000	**0.983**	**21.11**
5	27.6	1.950	0.003	0.902	13.518	27.6	1.943	0.003	0.903	13.395	**27.73**	**1.829**	**0.003**	**0.901**	**21.49**
10	25.38	4.358	0.007	0.822	13.541	25.42	4.565	0.008	0.823	13.469	**25.68**	4.114	**0.006**	**0.826**	**21.11**
15	23.07	7.560	0.013	0.724	13.455	23.12	8.260	0.013	0.724	13.409	**23.58**	**7.002**	**0.010**	**0.752**	**20.33**
20	20.87	11.994	0.020	0.611	13.017	20.87	13.197	0.022	0.606	13.249	**21.83**	**10.354**	**0.015**	**0.693**	**19.61**
25	18.89	17.956	0.029	0.429	12.043	18.58	18.814	0.039	0.473	12.867	**20.53**	**13.764**	**0.020**	**0.637**	**18.68**
30	16.96	25.877	0.043	0.379	11.041	16.16	24.897	0.071	0.339	12.487	**19.34**	**17.498**	**0.024**	**0.588**	**17.81**
35	15.00	37.424	0.063	0.266	9.624	13.69	31.921	0.135	0.217	12.001	**18.07**	**22.368**	**0.030**	**0.547**	**17.03**
40	13.33	50.593	0.090	0.179	8.363	11.60	37.928	0.243	0.130	11.310	**16.85**	**28.273**	**0.038**	**0.509**	**16.10**

Table 6. Criteria results for the Lena image with Salt and Pepper impulsive noise

and for comparative ones, for PGMF_{VMF} the $SSIM = 0.538$ and for PGMF_{AMF} the $SSIM = 0.563$. And also, for the results in Table 8, for 20 % of Salt and Pepper impulsive noise for MPGFMF the $SSIM = 0.571$ is obtained, and for comparative ones, for PGMF_{VMF} the $SSIM = 0.539$ and for PGMF_{AMF} the $SSIM = 0.567$.

	PGMF_{VMF}			PGMF_{AMF}			MPGFMF		
%	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR
0	41.87	0.994	36.16	**42.76**	0.994	36.67	42.50	**0.994**	**37.40**
5	31.05	0.901	27.93	**31.55**	0.907	**28.53**	31.52	**0.907**	28.51
10	26.48	0.792	25.17	27.04	0.806	26.08	**27.08**	**0.808**	**26.34**
15	23.61	0.671	24.93	24.30	0.692	25.22	**24.35**	**0.694**	**25.27**
20	20.92	0.538	25.59	21.61	0.563	25.77	**21.69**	**0.566**	**26.04**
25	19.23	0.446	24.99	19.81	0.471	25.06	**19.89**	**0.474**	**25.34**
30	17.65	0.358	24.54	18.19	0.381	24.57	**18.27**	**0.384**	**24.88**
35	16.3	0.290	23.38	16.88	0.312	23.18	**16.95**	**0.314**	**23.46**
40	15.24	0.258	**24.84**	15.77	0.259	24.45	**15.81**	**0.261**	24.63

Table 7. Criteria results for the Brain image with Random-value impulsive noise

	PGMF_{VMF}			PGMF_{AMF}			MPGFMF		
%	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR
0	41.87	0.994	36.16	**42.76**	0.994	36.67	42.50	**0.994**	**37.40**
5	30.28	0.942	24.45	31.18	0.950	25.34	**31.19**	**0.950**	**28.55**
10	24.15	0.846	22.37	24.84	0.862	23.27	**24.88**	**0.863**	**23.16**
15	20.50	0.706	21.49	21.03	0.727	22.00	**21.10**	**0.729**	**21.87**
20	17.22	0.539	22.01	17.76	0.567	22.17	**17.82**	**0.571**	**22.21**
25	15.03	0.383	21.17	15.47	0.408	21.62	**15.53**	**0.412**	**21.58**
30	13.16	0.264	20.99	13.54	0.285	21.26	**13.61**	**0.290**	**21.41**
35	11.74	0.176	20.95	12.09	0.192	21.21	**12.16**	**0.196**	**21.27**
40	10.40	0.115	20.71	10.71	0.127	20.93	**10.77**	**0.130**	**21.19**

Table 8. Criteria results for the Brain image with Salt and Pepper impulsive noise

Tables 9 and 10 show the results for the Mammography image. Our filter presents the best results for the criteria implemented from 0 through 40 % with steps of 5 % of noise density. In Table 9, results for the Brain image corrupted with Random-value impulse noise are presented, the proposal MPGFMF for the VSNR results for 30 % of noise density is of $VSNR = 25.12$, while for the comparative ones, PGMF_{VMF} presents $VSNR = 24.75$, and for PGMF_{AMF} has $VSNR = 23.05$. The same tendency is perceived in Table 10 for all the criteria implemented and for all the noise percentage levels, showing that our proposal outperforms the comparative ones.

In Figure 6 there is presented a visual comparison between the three filters using a zoomed region of Lena image corrupted with 5 %, 15 % and 25 % of Salt and Pepper impulsive noise. It is denoted that for 5 % and 15 % the differences between the three

	$PGMF_{VMF}$			$PGMF_{AMF}$			MPGFMF		
%	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR
0	26.01	0.978	48.41	26.01	0.978	48.41	**26.01**	**0.978**	**48.41**
5	25.35	0.841	41.85	25.39	0.843	41.85	**25.40**	**0.844**	42.31
10	23.908	0.689	35.57	24.09	0.697	35.57	**24.12**	**0.700**	**36.16**
15	22.00	0.523	31.69	22.35	0.535	31.69	**22.39**	**0.541**	**32.20**
20	20.05	0.370	28.27	20.48	0.384	28.27	**20.52**	**0.391**	**28.99**
25	18.39	0.254	26.16	18.86	0.268	24.75	**18.91**	**0.273**	**26.51**
30	16.95	0.167	24.75	17.44	0.178	23.05	**17.48**	**0.183**	**25.12**
35	15.71	0.112	23.05	16.21	0.1212	22.14	**16.24**	**0.124**	**23.53**
40	14.72	0.077	22.14	15.20	0.084	22.13	**15.22**	**0.086**	**22.44**

Table 9. Criteria results for the Mammography image with Random-value impulsive noise

	$PGMF_{VMF}$			$PGMF_{AMF}$			MPGFMF		
%	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR	PSNR	SSIM	VSNR
0	26.01	0.978	48.41	26.01	0.978	48.41	**26.01**	**0.978**	**48.41**
5	25.21	0.931	33.63	25.30	0.931	34.03	25.30	**0.933**	**33.98**
10	22.585	0.818	26.36	22.86	0.824	27.05	**22.89**	**0.826**	**27.21**
15	19.45	0.635	21.76	19.82	0.635	22.36	**19.86**	**0.650**	**22.26**
20	16.79	0.438	19.42	17.16	0.438	19.81	**17.20**	**0.456**	**19.76**
25	14.56	0.263	17.64	14.90	0.263	18.09	**14.94**	**0.280**	**18.21**
30	12.85	0.144	16.78	13.17	0.144	17.04	**13.21**	**0.151**	**17.08**
35	11.37	0.074	16.02	11.67	0.074	16.24	**11.71**	**0.083**	**16.31**
40	10.17	0.038	15.41	10.43	0.038	15.71	**10.46**	**0.046**	**15.81**

Table 10. Criteria results for the Mammography image with Salt and Pepper impulsive noise

filters are imperceptible only with numerical results. However, in the case of 25 % of impulsive noise, the proposal filter presents better noise suppression quality in visual representation denoted by the noisy clusters zones. The $PGMF_{AMF}$ filter propagates the noise between the pixels, this way the $PGMF_{VMF}$ filter does not suppress the noisy pixels formed in clusters well enough compared to our filter proposal. Our filter proposal (MPGFMF) does not propagate the noise and the noisy pixels in clusters are processed in a better way improving performance. Table 13 shows the computational cost performance of the three filters using the Lena image with resolutions of 512×512, $1\,024 \times 1\,024$, $2\,048 \times 2\,048$, $4\,096 \times 4\,096$ and $8\,192 \times 8\,192$. Superior filtering quality and fast execution in the proposal filter is due to changes proposed in the detection and filtering stages, the algorithm decreases the reading and writing counts, dealing better with noisy pixels in consequence of pixel clusters. The forgetful selection algorithm requires low register accesses inside the SMs and does not depend on the magnitude calculations between color pixels, performing better than the Vector Median Filter.

Next, in Table 14 a comparison with the state-of-the-art Deep Learning Filter (DLF) [26] for Random-value impulse noise for PSNR and SSIM for noise densities

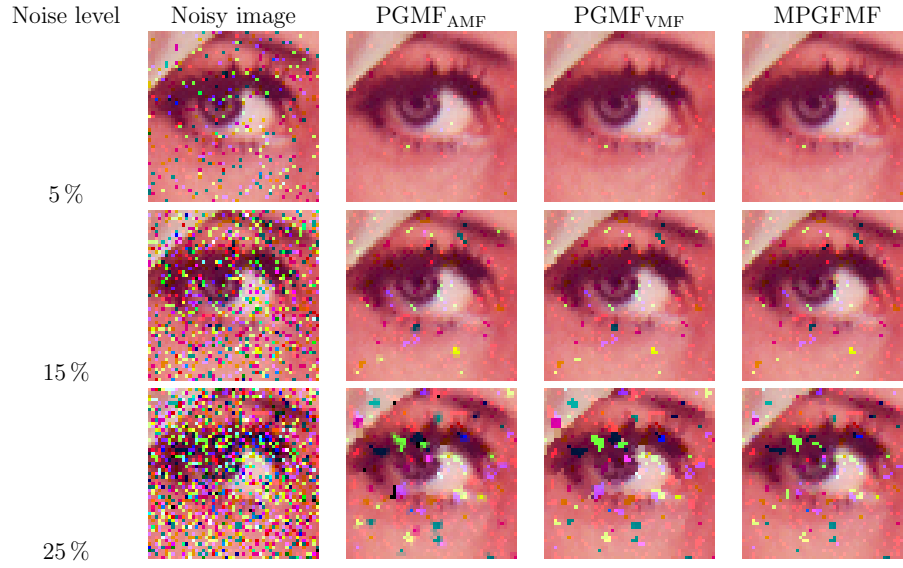


Figure 6. Visual comparative results between algorithms studied

Image Size	Filter	Time (ms)	Image Size	Filter	Time (ms)
	PGMF$_{AMF}$	0.739		PGMF$_{AMF}$	38.408
512 × 512	PGMF$_{VMF}$	1.730	4 096 × 4 096	PGMF$_{VMF}$	86.419
	MPGFMF	0.841		MPGFMF	41.568
	PGMF$_{AMF}$	2.807		PGMF$_{AMF}$	153.944
1 024 × 1 024	PGMF$_{VMF}$	6.594	8 192 × 8 192	PGMF$_{VMF}$	348.540
	MPGFMF	3.172		MPGFMF	160.628
	PGMF$_{AMF}$	9.422			
2 048 × 2 048	PGMF$_{VMF}$	21.731			
	MPGFMF	10.213			

Table 11. Runtime of the three filters using the Lena image corrupted with 10 % Salt and Pepper impulsive noise

10, 20, 30, 40, 50 and 60 % is made. From Table 14 we can perceive the poor results compared to a DLF filter, in PSNR and SSIM terms for 10, 20, 30, 40, 50 and 60 %.

For the runtime tests, as in [26], we used Cameraman (256 × 256), Sailing Boats (512 × 512) and Lena (512 × 512) corrupted with 10 %, 30 % and 50 % of Random-value impulse noise. In [26], the authors used an Intel i7 CPU based system and an Nvidia GeForce 960 GPU. Their obtained results and our results for the same images are shown in Table 15. Table 15 shows that our filter outperforms the Deep Learning filtering algorithm, i.e., for the Lena image with 10 % the runtime of filter found in [26] is 0.97 seconds. Our filter proposal has a running time of 0.0003 seconds. These results lead to a speed-up of 3 233.33 times.

Image Size	Filter	Time (ms)	Image Size	Filter	Time (ms)
$256 \times 320 \times 192$	$PGMF_{AMF}$	20.544	$1\,024 \times 1\,024$	$PGMF_{AMF}$	0.769
	$PGMF_{VMF}$	55.296		$PGMF_{VMF}$	0.763
	MPGFMF	19.968		MPGFMF	0.713

Table 12. Runtime of the three filters using the MRI Brain 3D image (image number 96) and the CT Mammography with 10 % of Salt and Pepper impulse noise

Image Size	Filter	Time (ms)	Image Size	Filter	Time (ms)
512×512	$PGMF_{AMF}$	0.739	$4\,096 \times 4\,096$	$PGMF_{AMF}$	38.408
	$PGMF_{VMF}$	1.730		$PGMF_{VMF}$	86.419
	MPGFMF	0.841		MPGFMF	41.568
$1\,024 \times 1\,024$	$PGMF_{AMF}$	2.807	$8\,192 \times 8\,192$	$PGMF_{AMF}$	153.944
	$PGMF_{VMF}$	6.594		$PGMF_{VMF}$	348.540
	MPGFMF	3.172		MPGFMF	160.628
$2\,048 \times 2\,048$	$PGMF_{AMF}$	9.422			
	$PGMF_{VMF}$	21.731			
	MPGFMF	10.213			

Table 13. Runtime of the three filters using the Lena image corrupted with 10 % Salt and Pepper impulsive noise

In our literature review, we found that little work has been done regarding image color filtering of impulse noise in parallel systems. We refocus previous ideas found in the literature, i.e. Peer Group Filtering and Median filtering with forgetful selection sort for grey-scale images, in order to improve performance of impulse noise filtering of color images further. In all experimental intervals we obtained better numerical results and qualitative results. We found that it is possible to run Parallel Peer Group Filters in real-time (< 33 ms per frame) up to $1\,024 \times 1\,024$ pixels in an Nvidia 2080 RTX card.

Criterion	PSNR (dB)						SSIM (r.u.)					
Noise (%)	10	20	30	40	50	60	10	20	30	40	50	60
Lena												
DLF	43.47	40.73	38.57	36.5	33.98	30.46	0.985	0.975	0.963	0.948	0.927	0.886
MPGFMF	27.67	26.87	25.97	24.7	23.36	22.12	0.823	0.765	0.718	0.658	0.588	0.517
Cameraman												
DLF	38.88	35.42	32.9	30.59	28.52	26.06	0.977	0.956	0.934	0.908	0.878	0.826
MPGFMF	24.34	23.5	22.6	21.38	20.22	19.01	0.773	0.702	0.642	0.555	0.483	0.418
Barbara												
DLF	43.44	40.14	37.61	35.51	33.2	30.82	0.990	0.978	0.959	0.932	0.878	0.797
MPGFMF	23.53	22.92	22.27	21.51	20.22	19.01	0.766	0.704	0.650	0.583	0.512	0.443
Boat												
DLF	42.51	39.58	37.56	35.86	34.22	32.48	0.973	0.953	0.931	0.904	0.866	0.800
MPGFMF	27.67	26.87	25.97	24.7	23.36	22.12	0.824	0.753	0.698	0.635	0.570	0.506

Table 14. PSNR and SSIM results applied by Lena, Cameraman, Barbara and Boat images corrupted by 10, 20, 30, 40, 50 and 60 % Random-value impulsive noise

Images	Cameraman			Sailing Boats			Lena		
Noise Density (%)	10	30	50	10	30	50	10	30	50
DLF	0.24	0.23	0.23	1.05	0.97	0.97	0.97	0.97	0.97
MPGFMF	0.000103	0.000103	0.000103	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003

Table 15. Runtime (secs.) comparison of a Deep Learning algorithm [26] against our proposal by 10, 30, and 50 % of Random-value impulsive noise

8 CONCLUSIONS

A novel GPU filtering algorithm for impulsive denoising applied to color images was proposed presenting better performance in preserving edges, details, structures, and chromaticity properties than those used as comparative. The criteria used to provide these validation quantitative results show that the *MPGFMF* preserves better the properties inherent to the images in suppressing the Random-value and Salt and Pepper impulsive noises. The execution runtime shows that the proposal is the second-fastest impulsive noise filter, comparing its rendering respect to the fastest PGMF_{AMF}, this filter is the worst in impulsive denoising, denoting that our proposal is the best option to denoise suppression and computational cost. For the MRI and CT images we found that our filter performs slightly better than the other GPU filters and for these types of images, our filter is the fastest. Compared with a state-of-the-art Deep Learning impulse noise filter our filter is not good in PSNR and SSIM terms for grayscale images, but in execution time, our filter is much faster ($> 3\,000$ times). To further improve the proposed filter, it is necessary to study and implement existing ordering algorithms or design a new one. We found that little work has been done regarding parallel color filtering and some research is needed, i.e. 3D color filtering of video sequences.

Acknowledgements

The authors are grateful to the Instituto Politécnico Nacional de México and the CONACyT de México for their support to this research work.

REFERENCES

[1] Salomon, C.—Breckon, T.: Fundamentals of Digital Image Processing. Wiley-Blackwell, 2011.

[2] Ponomaryov, V.—Montenegro, H.—Rosales, A.—Duchen, G.: Fuzzy 3D Filter for Color Video Sequences Contaminated by Impulsive Noise. Journal of Real-Time Image Processing, Vol. 10, 2015, pp. 313–328, doi: 10.1007/s11554-012-0262-9.

[3] Lukac, R.—Smolka, B.—Martin, K.—Plataniotis, K. N.—Venetsanopoulos, A. N.: Vector Filtering for Color Imaging. IEEE Signal Processing Magazine, Vol. 22, 2005, No. 1, pp. 74–86, doi: 10.1109/MSP.2005.1407717.

[4] Schulte, S.—De Witte, V.—Nachtegael, M.—Van Der Weken, D.—Kerre, E. E.: Fuzzy Two-Step Filter for Impulse Noise Reduction From Color Images. IEEE Transactions on Image Processing, Vol. 15, 2006, No. 11, pp. 3567–3578, doi: 10.1109/TIP.2006.877494.

[5] Owens, J. D.—Houston, M.—Luebke, D.—Green, S.—Stone, J. E.—Phillips, J. C.: GPU Computing. Proceedings of the IEEE, Vol. 96, 2008, No. 5, pp. 879–899, doi: 10.1109/JPROC.2008.917757.

[6] Smolka, B.: Fast Impulsive Noise Removal in Color Images. IEEE International Conference on Image Processing, 2013, pp. 1212–1216, doi: 10.1109/ICIP.2013.6738250.

[7] Sánchez, M. G.—Vidal, V.—Arnal, J.—Vidal, A.: Image Noise Removal on Heterogeneous CPU-GPU Configurations. Procedia Computer Science, Vol. 29, 2014, pp. 2219–2229, doi: 10.1016/j.procs.2014.05.207.

[8] Sánchez, M. G.—Vidal, V.—Bataller, J.—Arnal, J.: A Parallel Method for Impulsive Image Noise Removal on Hybrid CPU/GPU Systems. Procedia Computer Science, Vol. 18, 2013, pp. 2504–2507, doi: 10.1016/j.procs.2013.05.429.

[9] Sanchez, M. G.—Vidal, V.—Bataller, J.: Peer Group and Fuzzy Metric to Remove Noise in Images Using Heterogeneous Computing. In: Alexander, M. et al. (Eds.): Euro-Par 2011: Parallel Processing Workshops. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7155, 2011, pp. 502–510, doi: 10.1007/978-3-642-29737-3_55.

[10] Cook, S.: CUDA Programming: A Developer's Guide to Parallel Computing with GPUs. Morgan Kaufmann, Amsterdam, Boston, 2013, doi: 10.1016/C2011-0-00029-7.

[11] Tuning CUDA Applications for Maxwell. `http://docs.nvidia.com/cuda/maxwell-tuning-guide/index.html`.

[12] Kirk, D. B.—Hwu, W. W.: Programming Massively Parallel Processors: A Hands-on Approach. 1st Edition. Morgan Kaufmann, Burlington, MA, 2010.

[13] Couturier, R.: Designing Scientific Applications on GPUs. Chapman and Hall, Boca Raton, Florida, 2013.

[14] CUDA C Programming Guide. Available at: `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`.

[15] Smolka, B.: Adaptive Truncated Vector Median Filter. IEEE International Conference on Computer Science and Automation Engineering, Vol. 4, 2011, pp. 261–266, doi: 10.1109/CSAE.2011.5952847.

[16] Morillas, S.—Gregori, V.—Peris-Fajarnés, G.—Latorre, P.: A Fast Impulsive Noise Color Image Filter Using Fuzzy Metrics. Real-Time Imaging, Vol. 11, 2005, No. 5-6, pp. 417–428, doi: 10.1016/j.rti.2005.06.007.

[17] Perrot, G.—Domas, S.—Couturier, R.: Fine-Tuned High-Speed Implementation of a GPU-Based Median Filter. Journal of Signal Processing Systems, Vol. 75, 2014, No. 3, pp. 185–190, doi: 10.1007/s11265-013-0799-2.

[18] Bovik, A.: Handbook of Image and Video Processing. Academic Press, 2005, doi: 10.1016/B978-0-12-119792-6.X5062-1.

[19] Trahanias, P. E.—Karakos, D.—Venetsanopoulos, A. N.: Directional Processing of Color Images: Theory and Experimental Results. IEEE Transactions on Image Processing, Vol. 5, 1996, No. 6, pp. 868–880, doi: 10.1109/83.503905.

[20] Lukac, R.: Adaptive Vector Median Filtering. Pattern Recognition Letters, Vol. 24, 2003, pp. 1889–1899, No. 12, doi: 10.1016/S0167-8655(03)00016-3.

[21] Wang, Z.—Bovik, A. C.—Sheikh, H. R.—Simoncelli, E. P.: Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing, Vol. 13, 2004, No. 4, pp. 600–612, doi: 10.1109/TIP.2003.819861.

[22] Chandler, D. M.—Hemami, S. S.: VSNR: A Wavelet-Based Visual Signal-to-Noise Ratio for Natural Images. IEEE Transactions on Image Processing, Vol. 16, 2007, No. 9, pp. 2284–2298, doi: 10.1109/TIP.2007.901820.

[23] Poynton, C.: The Rehabilitation of Gamma. In: Rogowitz, B. E., Pappas, T. N. (Eds.): Human Vision and Electronic Imaging III. Proceedings of the SPIE, Vol. 3299, 1998, pp. 232–249.

[24] I Do Imaging. `https://wiki.idoimaging.com/index.php?title=Sample_Data`, accessed on 11/1/2019.

[25] The Mini-MIAS Database of Mammograms. `http://peipa.essex.ac.uk/info/mias.html`, accessed on 11/1/2019.

[26] Jin, L.—Zhang, W.—Ma, G.—Song, E.: Learning Deep CNNs for Impulse Noise Removal in Images. Journal of Visual Communication and Image Representation, Vol. 62, 2019, pp. 193–205, doi: 10.1016/j.jvcir.2019.05.005.

[27] Turkmen, I.: The ANN Based Detector to Remove Random-Valued Impulse Noise in Images. Journal of Visual Communication and Image Representation, Vol. 34, 2016, pp. 28–36, doi: 10.1016/j.jvcir.2015.10.011.

José Agustín Tortolero Osuna studies in the Instituto Politécnico Nacional de México. He is currently pursuing his Ph.D. degree focusing on image processing. His current interests are high-performance computing and image enhancement.

Alberto Jorge Rosales Silva works in the Instituto Politécnico Nacional de México. He obtained his Ph.D. in communications and electronics (2008), Master of Science in telecommunications engineering (2004) and Bachelors degree in engineer communications and electronics (1999) from ESIME-Culhuacan in Instituto Politécnico Nacional de México. Topics currently being developed in image processing, multispectral and multichannel video in real-time, visual computing, as well as medical image processing.