Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.039686
Article

An Optimized Implementation of a Novel Nonlinear Filter for Color Image Restoration

Turki M. Alanazi*

Department of Electrical Engineering, College of Engineering, Jouf University, Sakaka, Saudi Arabia
*Corresponding Author: Turki M. Alanazi. Email: tmanazi@ju.edu.sa

Received: 11 February 2023; Accepted: 11 April 2023; Published: 23 June 2023

Abstract: Image processing is becoming more popular because images are being used increasingly in medical diagnosis, biometric monitoring, and character recognition. But these images are frequently contaminated with noise, which can corrupt subsequent image processing stages. Therefore, in this paper, we propose a novel nonlinear filter for removing "salt and pepper" impulsive noise from a complex color image. The new filter is called the Modified Vector Directional Filter (MVDF). The suggested method is based on the traditional Vector Directional Filter (VDF). However, before the candidate pixel is processed by the VDF, the MVDF employs a threshold and the neighboring pixels of the candidate pixel in a 3 × 3 filter window to determine whether it is noise-corrupted or noise-free. Several reference color images corrupted by impulsive noise with intensities ranging from 3% to 20% are used to assess the MVDF's effectiveness. The results of the experiments show that the MVDF is better than the VDF and the Generalized VDF (GVDF) in terms of the PSNR (Peak Signal-to-Noise Ratio), NCD (Normalized Color Difference), and execution time for the denoised image. In fact, the PSNR is increased by 6.554% and 12.624%, the NCD is decreased by 20.273% and 44.147%, and the execution time is reduced by approximately a factor of 3 for the MVDF relative to the VDF and GVDF, respectively. These results prove the efficiency of the proposed filter. Furthermore, a hardware design is proposed for the MVDF using the High-Level Synthesis (HLS) flow in order to increase its performance. This design, which is implemented on the Xilinx Zynq XCZU9EG Field-Programmable Gate Array (FPGA), allows the restoration of a 256 × 256-pixel image in 2 milliseconds (ms) only.

Keywords: Nonlinear filter; impulsive noise; noise reduction; software/hardware optimization; color image; HLS; FPGA

1 Introduction

Noise frequently degrades image quality and can come from a variety of sources, including electrical disturbances, communication issues, noisy sensors, etc. This noise causes problems in digital image processing. Moreover, denoising and enhancement of images while preserving edge sharpness and image details are essential operations before using them for other image processing techniques.

Nonlinear filters [1] are useful for removing noise from images, as they can effectively preserve the edges and details of the image while smoothing out the noise. They are a type of image filtering technique that involves modifying the pixel values of an image based on their local environment. Some commonly used nonlinear filters include the median filter, the bilateral filter, and the adaptive filter. These filters are widely used in image processing applications such as computer vision, medical imaging, and remote sensing. The Vector Directional Filter (VDF) [2,3] is a type of nonlinear filter that is used to remove noise from color images. Unlike traditional color filters that process each color channel independently, VDF operates on the color vector space, taking into account the spatial correlations between different color channels. This filter employs the angle between the image vectors as an ordering criterion, which allows it to provide an optimal directional estimator. In this case, the vectors of the image with directions atypical in RGB space are removed, and the best estimations of the chromaticity are attained. But the VDF has some limitations, especially when some noisy pixels are not identified or non-noisy pixels are classified as noise. However, in the first scenario, the pixels remain unchanged. For the second scenario, the pixels will be replaced. As a result, these two scenarios may have negative effects, such as blurring fine details or destroying the edges of the reconstructed image.

In this context, to improve the performance of the VDF, a new filter is proposed, which is the Modified VDF (MVDF). Besides, a hardware architecture is suggested for this filter for real-time processing. Nevertheless, to design a hardware architecture, the Low-Level Synthesis (LLS) flow can be used through a hardware description language such as Verilog. But, the LLS needs a high level of skill and much time to design any hardware architecture, especially for complex algorithms [4–6]. For that, in the last decade, a High-Level Synthesis (HLS) flow was established to generate several hardware designs for any algorithm in a few times from a high-level programming language like C or C++ [7–9] and realize the space exploration to select the best design. Thereby, in this work, the HLS flow is used to create an efficient hardware architecture for the MVDF and implement it in the Zynq XCZU9EG Xilinx Field-Programmable Gate Array (FPGA).

The following is the structure of the paper: The related works are illustrated in Section 2. Section 3 gives an overview of the VDF and the theory of the proposed MVDF. In Section 4, several experiments are realized to select the best parameters for the proposed filter, and a comparative study is conducted. A real-time MVDF hardware architecture is designed in Section 5. The paper concludes with Section 6.

2 Related Works

A commonly encountered noise in color images is a “salt and pepper” impulsive noise [10]. However, in the literature, several filters are proposed to remove this noise. In fact, for reducing impulse noise from color images, a basic vector filter [11,12] is provided. The vector ordering approach is utilized in these filters. Furthermore, weighted vector filters are suggested in [13–17], where each pixel of the filter window is given a non-negative weight in order to eliminate outliers. In addition, [18,19] provide a hybrid vector filter, which employs several subfilters and produces an output that is a composite of the samples of the input vector. Moreover, fuzzy vector filters are introduced in [20–22]. In this kind of filter, the fuzzy functions created by various distance functions are utilized to determine the weights of a nonlinear adaptive filter. In recent years, popular techniques based on convolutional neural network architecture and deep learning have been suggested to improve the restoration of image quality [23–25].

The most important filters used to preserve the chromaticity of the restored color image and suitable for real-time implementation in the basic vector filter family are the VDF and Generalized VDF [26]. In order for these filters to restore the image, a filter window must be applied to each pixel in the image. This process is carried out without differentiating between a non-noisy pixel and a noisy pixel, which can lead to a poor-quality image. Moreover, these filters require a large amount of time to filter the image and have a high level of computational complexity.

However, in the literature, some solutions are proposed to reduce the computational complexity of these filters. In fact, the authors of [27] suggest a real-time hardware implementation for the VDF. Therefore, the nonlinear function in the VDF is approximated to provide a flexible hardware architecture. But, this approximation leads to a loss in the quality of the restored image. In [28], the authors propose a floating-point implementation of the VDF on the Texas Instruments DSP TMS320C6711. This solution allows filtering a 320 × 320 image in 1342 ms at 150 MHz. Reference [29] introduces some approximations to implement VDF on an Intel Pentium® processor running at 2.66 GHz. These approximations allow a reduction in the processing time but a decrease in the filtered image quality.

In this study, the MVDF is suggested to enhance the quality of the denoised image without increasing the computational complexity relative to the VDF and GVDF filters. In fact, the principle of the MVDF is to detect, through a threshold and neighborhood pixels of the central pixel in the 3 × 3 filter window, whether the selected pixel in the image is corrupted by noise or not before processing it by the VDF. Besides, for real-time execution of the MVDF, the hardware architecture is designed for this filter via the HLS flow.

3 Materials and Methods

3.1 VDF Overview

The VDF allows optimization of the vector direction variation related to the pixel chrominance. Hence, the VDF permits minimizing the angular distance.

Consider x: $Z^a \rightarrow Z^b$ which presents the multichannel signal, where a is the image dimension and b is the color channel in the image. However, $a = 2$ and $b = 3$ for the RGB color image. Besides, consider $V = \left\{x_j \in Z^2; j = 1, \ldots, M\right\}$ which illustrates the size of the filter window. In fact, $x_1, x_2, \ldots x_M$ is the noisy pixels in the filter window. $x_{(M+1)/2}$ is the central pixel in the filter window and allows to determine its position. The angular distance for the input x_i of the filter window is determined by Eq. (1).

$$\propto_j = \sum_{i=1}^{M} Arccos\left(\frac{x_j x_i^T}{\|x_j\|.\|x_i\|}\right) \tag{1}$$

The *Arccos* function determines the angular distance between two channels $x_j = \left(R_j, G_j, B_j\right)$. and $x_i = (R_i, G_i, B_i)$.

Assuming that $\propto_1, \ \propto_2, \ldots, \ \propto_M$ are the sum of the aggregated angular distances, the function *Argmin*, presented in Eq. (2), permits to find the minimum angle in the filter window W allowing to minimize the angle with other pixels.

$$y_{VDF} = Argmin\left(\propto_j\right) \tag{2}$$

So, the output of the VDF filter is the vector $x_{(1)}$ from the input vectors in the filter window. Nevertheless, the vector $x_{(1)}$ is determined by $x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(M)}$which is related to the lowest weighted angular distance $\propto_{(1)} \in \left\{\propto_{(1)} \leq \propto_{(2)} \leq \ \ldots \ \leq \propto_{(M)}\right\}$

3.2 Proposed Filter

The problem with the VDF is that it works with a fixed filter window. This causes a loss in the details of the image and creates a blurred image while filtering the noisy pixels. For that, our purpose is to minimize error in the VDF output and increase its performance by estimating if the selected pixel in the image is corrupted by noise or not before it is processed by the VDF. For that, a threshold (T) and the relationship with neighborhood pixels ($RWNP$) of the candidate pixel, which is the central pixel $x_{(M+1)/2}$ in the filter window (W), are used to classify if the candidate pixel is corrupted by noise or free of noise. In fact, if the $RWNP$ is greater than or equal to T, then the central pixel $x_{(M+1)/2}$ is considered a corrupted pixel and should be processed by the VDF filter to determine the denoising pixel. Otherwise, the central pixel is considered free of noise and stays unchanged. This is the principle of our proposed MVDF.

But in the MVDF, there is an extra filter that checks if the central pixel is noisy or not. This filter is based on the computation of the $RWNP$ value. In fact, in the filter window, the minimum order pixels of the VDF output and their closest neighborhoods contain comparable values of the calculated angle relative to another input pixel. So, the $RWNP$ is computed as the distance between the central pixel $x_{(M+1)/2}$ and the lowest first (f) pixels $x_{(1)}, x_{(2)}, \ldots, x_{(f)}$ related to the lowest weighted angle $\propto_{(1)}, \propto_{(2)}, \ldots, \propto_{(f)}$ determined by the VDF in the filter window. Nevertheless, the f highest similarity input pixels ($x_{(1)}, x_{(2)}, \ldots, x_{(f)}$) provides a high measurement of the noise in the center pixel. Thus, the $RWNP$ is calculated by Eq. (3), where f should be less than or equal to M.

$$RWNP = \left\| \frac{1}{f} \sum_{j=1}^{M} x_j - x_{\frac{(M+1)}{2}} \right\| \tag{3}$$

So, the output pixel of the MVDF is specified per Eq. (4).

$$y_{MVDF} = \begin{cases} x_{(1)} & if\ RWNP \geq T \\ x_{(M+2)/2} & Otherwise \end{cases} \tag{4}$$

Indeed, from Fig. 1, we can see that for the 3×3 filter window nine angular distances are calculated. Then, these distances are arranged in ascending order to determine the f highest similarity input pixels $x_{(1)}, x_{(2)}, \ldots, x_{(f)}$. Based on these results, we compute the $RWNP$ value in order to determine the output RGB pixel of the MVDF. Thus, referring to Eq. (4), the output of the $\mathrm{y_{MVDF}}$ is equal to $x_{(1)}$when the $RWNP \geq T$. This means that the central pixel $x_{(M+1)/2}$ is probably corrupted by impulsive noise, and it is replaced by $x_{(1)}$. On the other hand, the output of the $\mathrm{y_{MVDF}}$ is equal to $x_{(M+1)/2}$ when the $RWNP < T$. This means that the central pixel $x_{(M+1)/2}$ is free of noise and can stay unchanged.

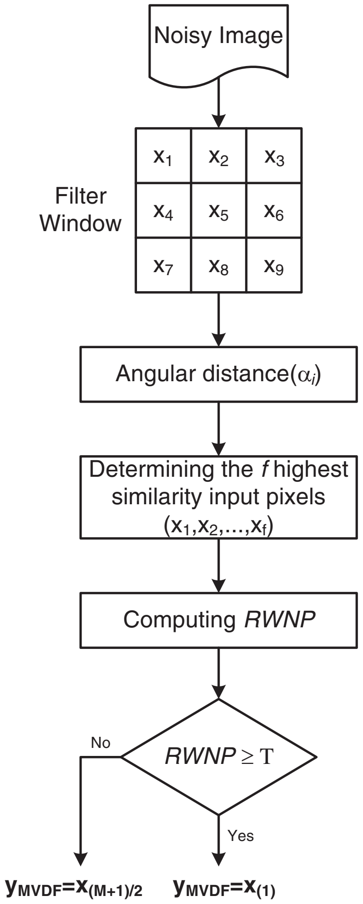


Figure 1: MVDF block diagram

4 Results and Discussion

The performance of our proposed MVDF depends on the f and T parameters. In fact, we have to select the correct values for these parameters to preserve the details of the filtered image and achieve a high visual quality in the restored color image, irrespective of the intensity of the impulsive noise. To realize this study, two standard images are used, which are Lena and Sailboat. These images have a resolution of 256 × 256 pixels and are contaminated by "salt and pepper" impulsive noise of intensities 10% and 20%. This noise is produced by the *imnoise* function given by the MATLAB tool. The NCD (Normalized Color Difference) and PSNR (Peak Signal-to-Noise Ratio) metrics are used to measure the color image quality objectively. Indeed, the PSNR and NCD metrics are computed through Eqs. (5) and (8), respectively.

$$PSNR = 10\log\left(\frac{255^2}{MSE}\right) \tag{5}$$

where the Mean Squared Error (MSE) for three image components R (Red), G (Green), and B (Blue) is calculated based on Eq. (6)

$$MSE = \frac{MSE_R + MSE_G + MSE_B}{3} \tag{6}$$

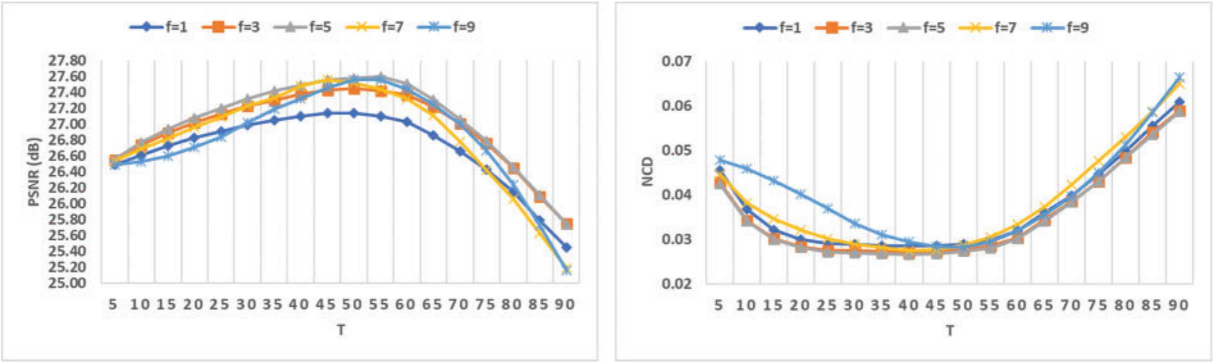
However, the MSE for each channel in the color image is determined by Eq. (7). j presents the image channel (R, G, and B). X and Y provide the height and the length of the image, respectively. But, O_j and I_j indicate the restored and the input images, respectively.

$$MSE_j = \frac{1}{XY}\sum_{n=1}^{x}\sum\nolimits_{m=1}^{y}\left(O_j(n,m) - I_j(n,m)\right)^2 \tag{7}$$

$$NCD = \frac{\sum_{n=1}^{x}\sum_{m=1}^{y}\sqrt{(L_o(n,m) - L_i(n,m))^2 + (u_o(n,m) - u_i(n,m))^2 + (v_o(n,m) - v_i(n,m))^2}}{\sum_{n=1}^{x}\sum_{m=1}^{y}\sqrt{(L_o(n,m))^2 + (u_o(n,m))^2 + (v_o(n,m))^2}} \tag{8}$$

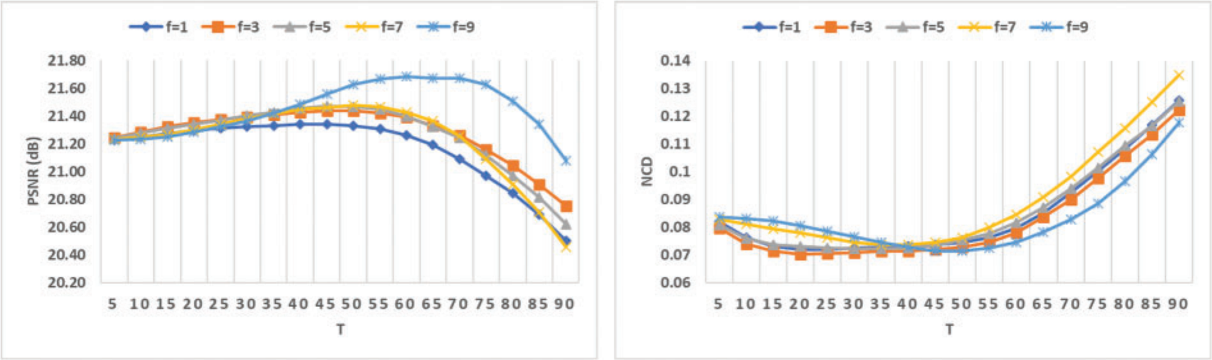
where, the L_o and L_i represent the luminance component of the restored and input images, respectively. But, (u_o, v_o) and (u_i, v_i) are the restored and the input images for the chrominance components, respectively.

The MVDF code is written in the programming language C, and it is made with the help of Microsoft Visual Studio Community 2019. The CoreTM i7-1165G7@2.80 GHz Intel processor is used to run the C code for MVDF on the computer. The *Timer* of the Intel processor is employed to calculate the processing time of our MVDF code. Figs. 2 and 3 illustrate the curves of the PSNR and NCD values in terms of T (from 5 to 90) and f (1, 3, 5, 7, and 9) for the Lena and Sailboat images, respectively. In fact, for each value of f, we have drawn a curve by varying the value of T. Our study was conducted with two intensity levels of impulsive noise, equal to 10% and 20%. Our goal is to study the behavior of the MVDF and select the best values of the f and T parameters in order to provide an excellent restoration of the noisy color image with our proposed MVDF.



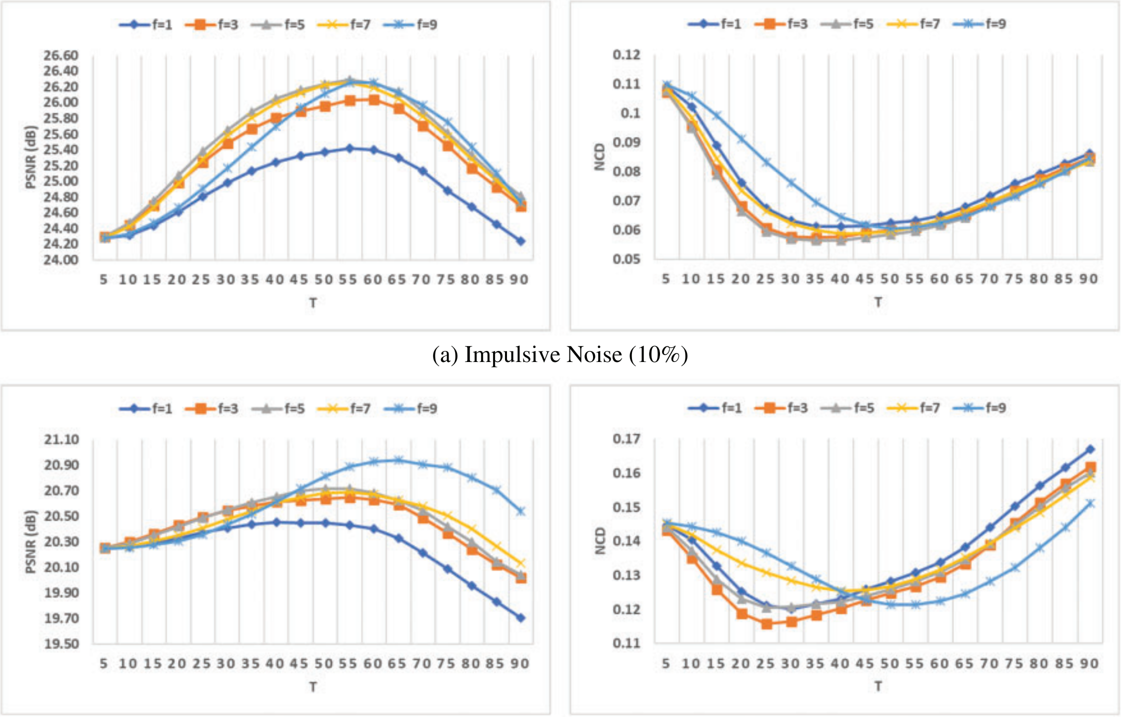
(a) Impulsive Noise (10%)

Figure 2: (Continued)



(b) Impulsive Noise (20%)

Figure 2: Measurement of the PSNR and NCD of the restored Lena image with MVDF for various f and T values and noise intensity (10% and 20%)



(a) Impulsive Noise (10%)

(b) Impulsive Noise (20%)

Figure 3: Measurement of the PSNR and NCD of the restored Sailboat image with MVDF for various f and T values and noise intensity (10% and 20%)

It is clear from Figs. 2a and 3a that the MVDF provides the best performance in terms of the PSNR and NCD when $f = 5$, 7, or 9 and $T = 50$ for Lena and Sailboat images, which are corrupted by 10% of impulsive noise. On the other hand, if we increase the level of noise to 20% for Lena and Sailboat images, we can see from Figs. 2b and 3b, respectively, that the MVDF provides the best performance when only $f = 9$ and $T = 50$. Thus, we can notice from the above findings that when the noise level is increased, it is worth using the nine input pixels $(x_{(1)}, x_{(2)}, \ldots, x_{(9)})$ from the 3×3 filter window to calculate the *RWNP* parameter for MVDF. This allows a high measurement of the noise in the center pixel and minimizes the error in the output of the MVDF. Consequently, to reduce the computational complexity of the MVDF and increase its performance to restore the color image with the best quality while preserving the image details, we select as values for *f* and *T* parameters 9 and 50, respectively. Accordingly, the block diagram of the MVDF is became as depicted in Fig. 4. However, in the beginning, the RWNP parameter is computed for $f = 9$. Then, we compare the *RWNP* with $T = 50$. In fact, if $RWNP < 50$, the candidate pixel is determined to be noise-free and then the MVDF result is equal to $x_{(M+1)/2}$. Otherwise, it is considered noise-corrupted, and the VDF is used to identify the filtered pixel that is equal to $x_{(1)}$ corresponding to minimum angular distance $(\propto_j)$.

The performance of our proposed MVDF is compared with that of VDF and GVDF. Indeed, Fig. 5 illustrates the performance comparison in terms of PSNR and NCD metrics for VDF, GVDF, and MVDF. This study is realized for indoor scenes such as Lena (Fig. 5a) and outdoor scenes such as Sailboat (Fig. 5b). These images are contaminated with 3%, 5%, 10%, 15%, and 20% of "salt and pepper" impulsive noise. Nevertheless, we can notice from Fig. 5 that the MVDF provides a better performance compared to the VDF and GVDF in terms of PSNR and NCD metrics, especially for low and medium levels of noise. Besides, we can observe from Fig. 6, which illustrates the execution time in milliseconds (ms) of the VDF, GVDF, and MVDF, that the MVDF reduces the computational complexity relative to the VDF and GVDF. In fact, the MVDF gives a reduction in the execution time of 64.088% and 67.172% relative to the VDF and GVDF, respectively. In light of the above findings, our proposed MVDF is superior to other filters in terms of the quality of the restored image and the reduction of its computational complexity.

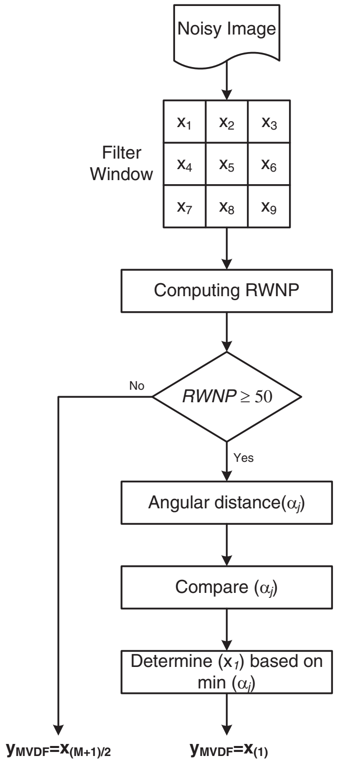
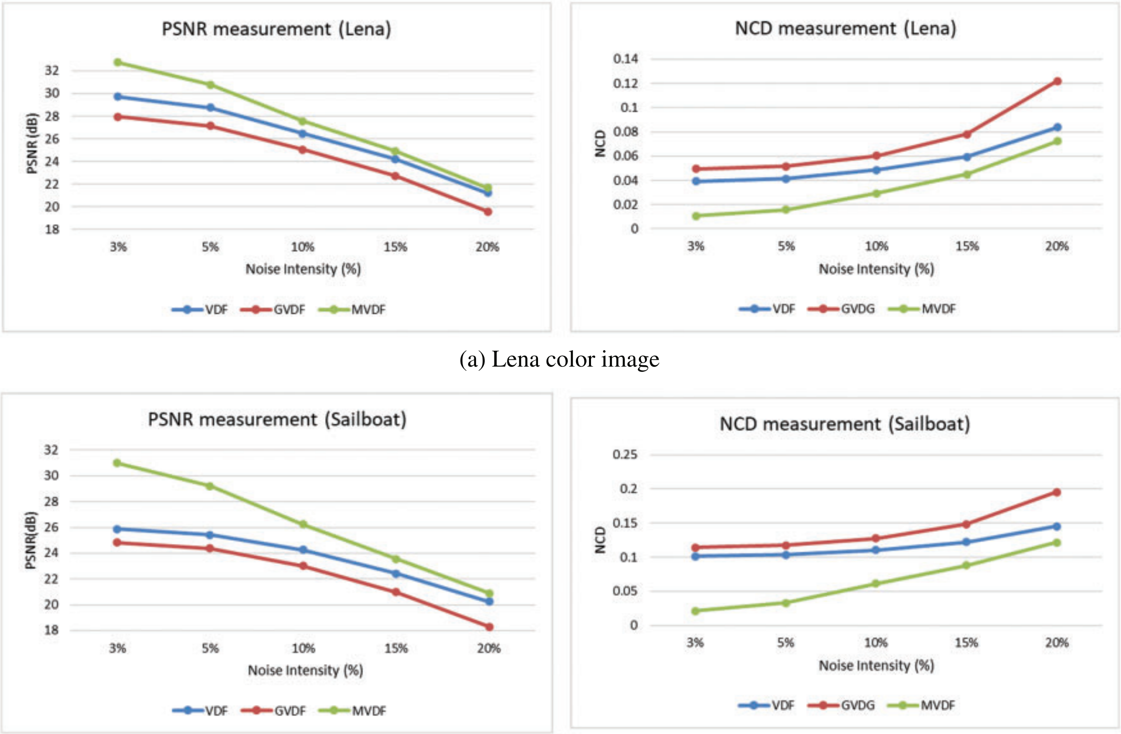


Figure 4: Block diagram of the MVDF for $f = 9$ and $T = 50$



(a) Lena color image

(b) Sailboat color image

Figure 5: PSNR and NCD comparison of the restored images ((a) Lena and (b) Sailboat) by the VDF, GVDF, and MVDF

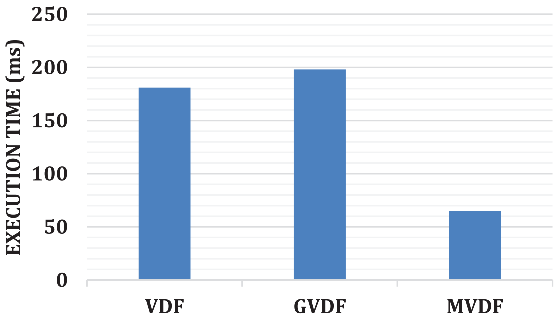


Figure 6: Comparison of the execution time (ms) between VDF, GVDF, and MVDF

To evaluate the efficiency of our proposed MVDF, we extended our study by using more standard test images. By the way, Table 1 presents the performance comparison of the VDF, GVDF, and MVDF for five test images (Lena, Sailboat, Flower, Barbara, and Peppers). This comparison is realized in terms of objective measurement through the PSNR and NCD metrics for impulsive noise intensity between 3% and 20%. Furthermore, Figs. 7 and 8 depict the performance comparison of the subjective measurement of the restored image by the VDF, GVDF, and MVDF for 3% and 10% of the impulsive noise level, respectively. From Table 1, it is worth noting that the MVDF enhances the PSNR by 6.554% and 12.624% and reduces the NCD by 20.273% and 44.147% relative to the VDF and GVDF, respectively. These findings are supported by the subjective evaluations shown in Figs. 7 and 8, which clearly show that the visual quality of the restored image by MVDF is superior to that restored by VDF and GVDF. In fact, we can see that the MVDF preserves the image details and provides a sharper image. But the VDF and GVDF provide a blurred image.

Table 1: Performance evaluation of VDF, GVDF, and MVDF in terms of PSNR and NCD

		VDF		GVDF		MVDF	
Image	Noise	PSNR	NCD	PSNR	NCD	PSNR	NCD
Lena	3%	29.716	0.039	27.954	0.049	32.763	0.010
	5%	28.750	0.041	27.131	0.051	30.781	0.015
	10%	26.475	0.049	25.047	0.060	27.552	0.029
	15%	24.214	0.059	22.737	0.078	24.921	0.044
	20%	21.224	0.083	19.573	0.122	21.670	0.072
	Average	**26.076**	**0.054**	**24.488**	**0.072**	**27.537**	**0.034**
Sail boat	3%	25.868	0.101	24.830	0.114	30.985	0.020
	5%	25.410	0.103	24.380	0.117	29.201	0.032
	10%	24.268	0.110	23.017	0.127	26.252	0.060
	15%	22.411	0.122	20.991	0.148	23.553	0.088
	20%	20.247	0.145	18.304	0.195	20.891	0.121
	Average	**23.641**	**0.116**	**22.304**	**0.140**	**26.176**	**0.064**
Flower	3%	27.200	0.035	25.437	0.045	30.325	0.014
	5%	26.466	0.037	24.794	0.048	28.813	0.020
	10%	24.790	0.045	23.303	0.059	26.228	0.034
	15%	22.723	0.059	20.926	0.084	23.616	0.051
	20%	20.386	0.083	18.300	0.136	20.948	0.077
	Average	**24.313**	**0.052**	**22.552**	**0.074**	**25.986**	**0.039**
Barbara	3%	26.412	0.029	25.243	0.038	30.542	0.013
	5%	25.868	0.030	24.842	0.040	29.000	0.020
	10%	24.798	0.037	23.676	0.047	26.512	0.035
	15%	23.002	0.049	21.566	0.069	23.974	0.049
	20%	20.504	0.077	18.830	0.123	21.117	0.077
	Average	**24.117**	**0.044**	**22.831**	**0.063**	**26.229**	**0.039**
Peppers	3%	32.286	0.024	29.744	0.038	34.577	0.017
	5%	30.857	0.026	28.769	0.041	31.983	0.028
	10%	27.841	0.035	25.841	0.054	28.332	0.048
	15%	24.819	0.050	22.629	0.079	25.282	0.065
	20%	21.010	0.081	18.792	0.139	21.283	0.096
	Average	**27.363**	**0.043**	**25.155**	**0.070**	**28.291**	**0.051**

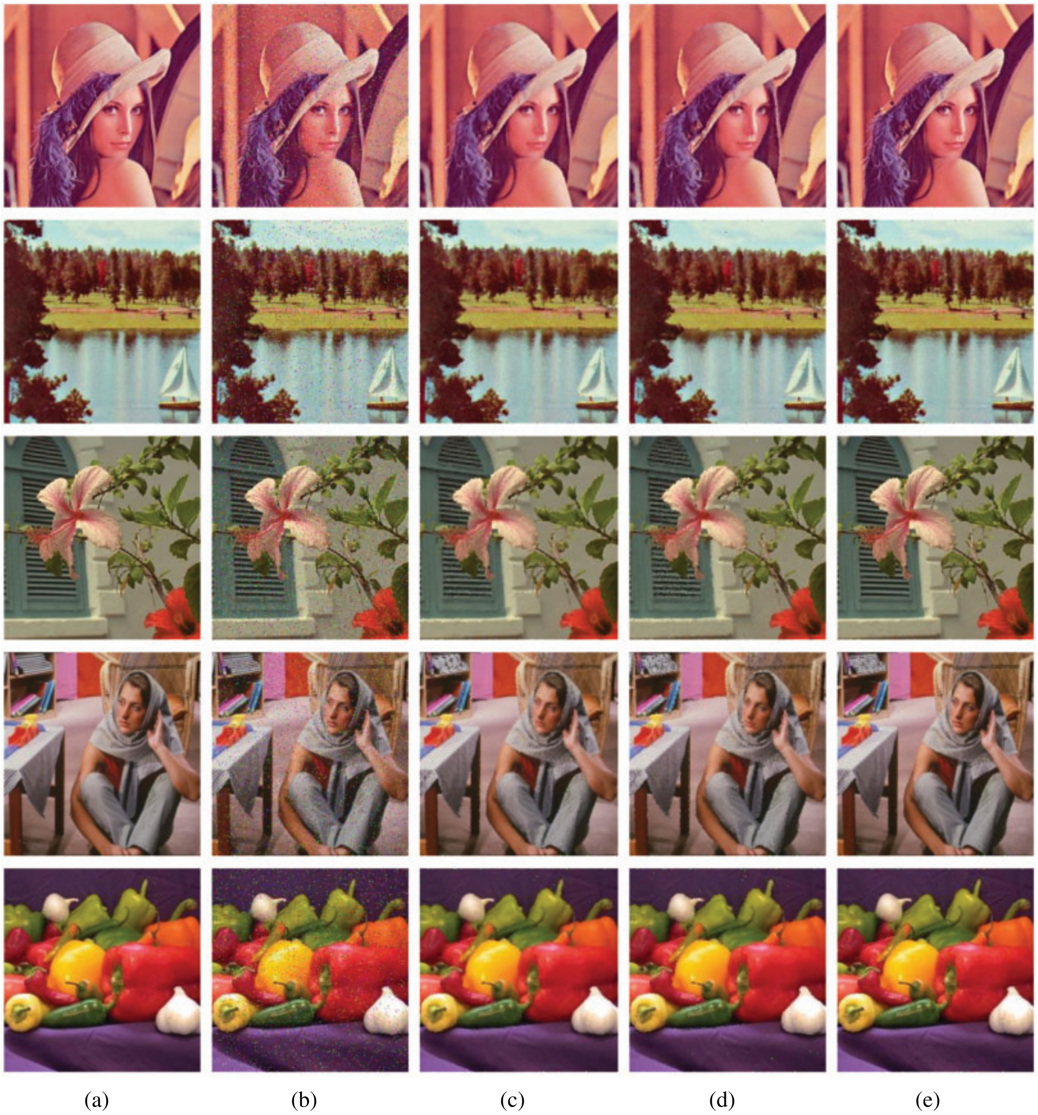


Figure 7: Comparison of the visual image quality between (a) original image, (b) contaminated image with 3% of impulsive noise and restored image with (c) VDF, (d) GVDF and (e) MVDF

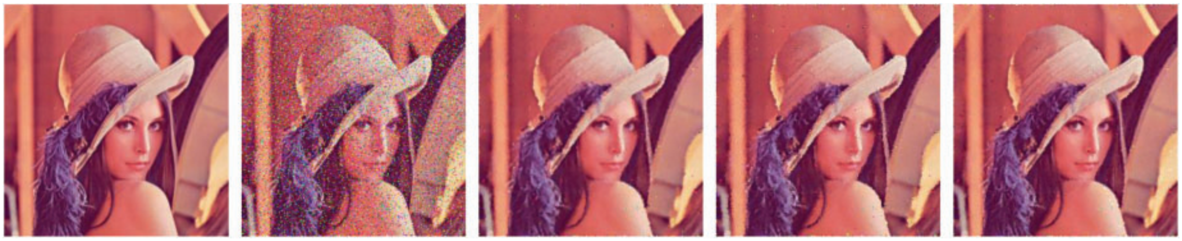


Figure 8: (Continued)

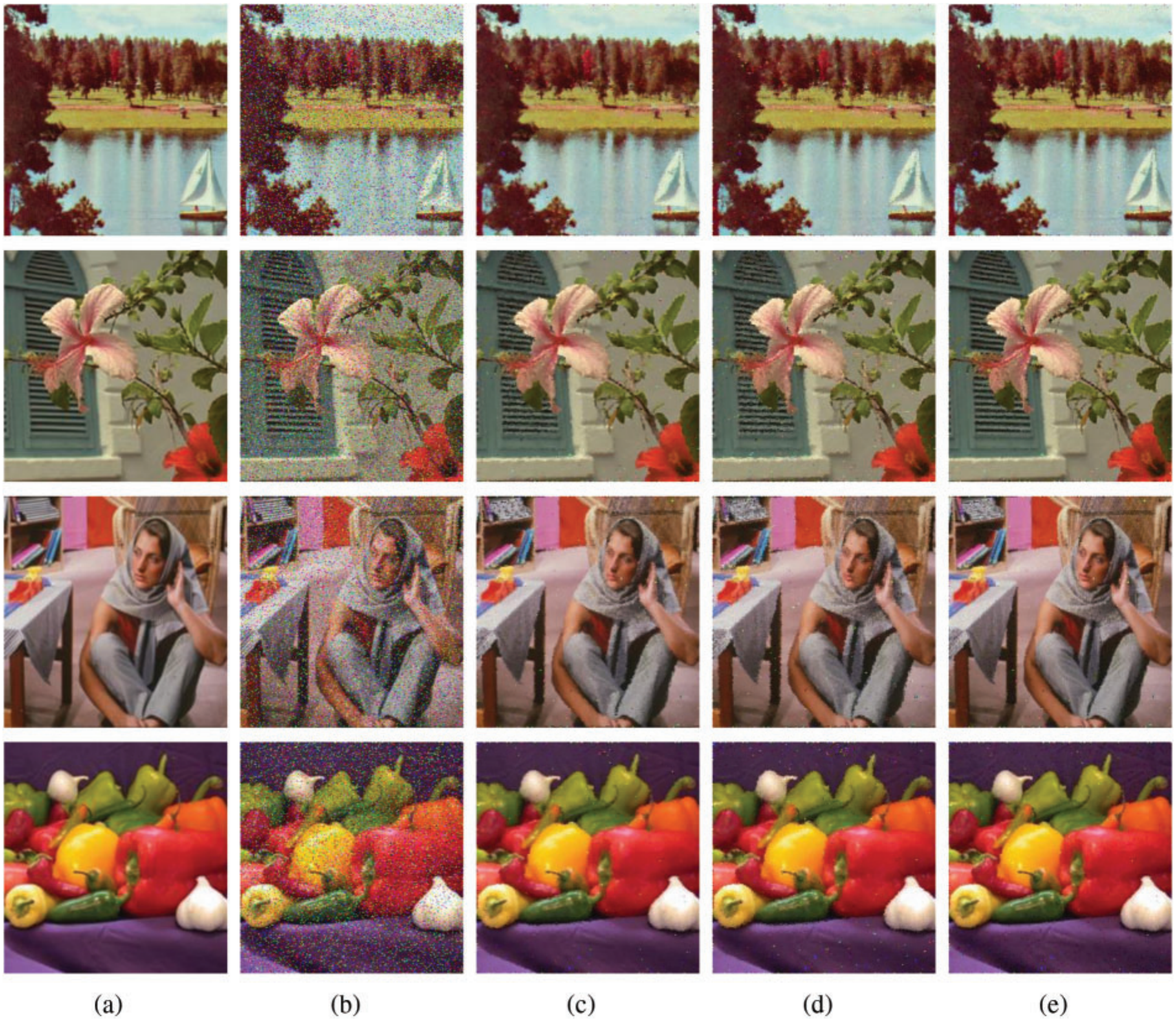
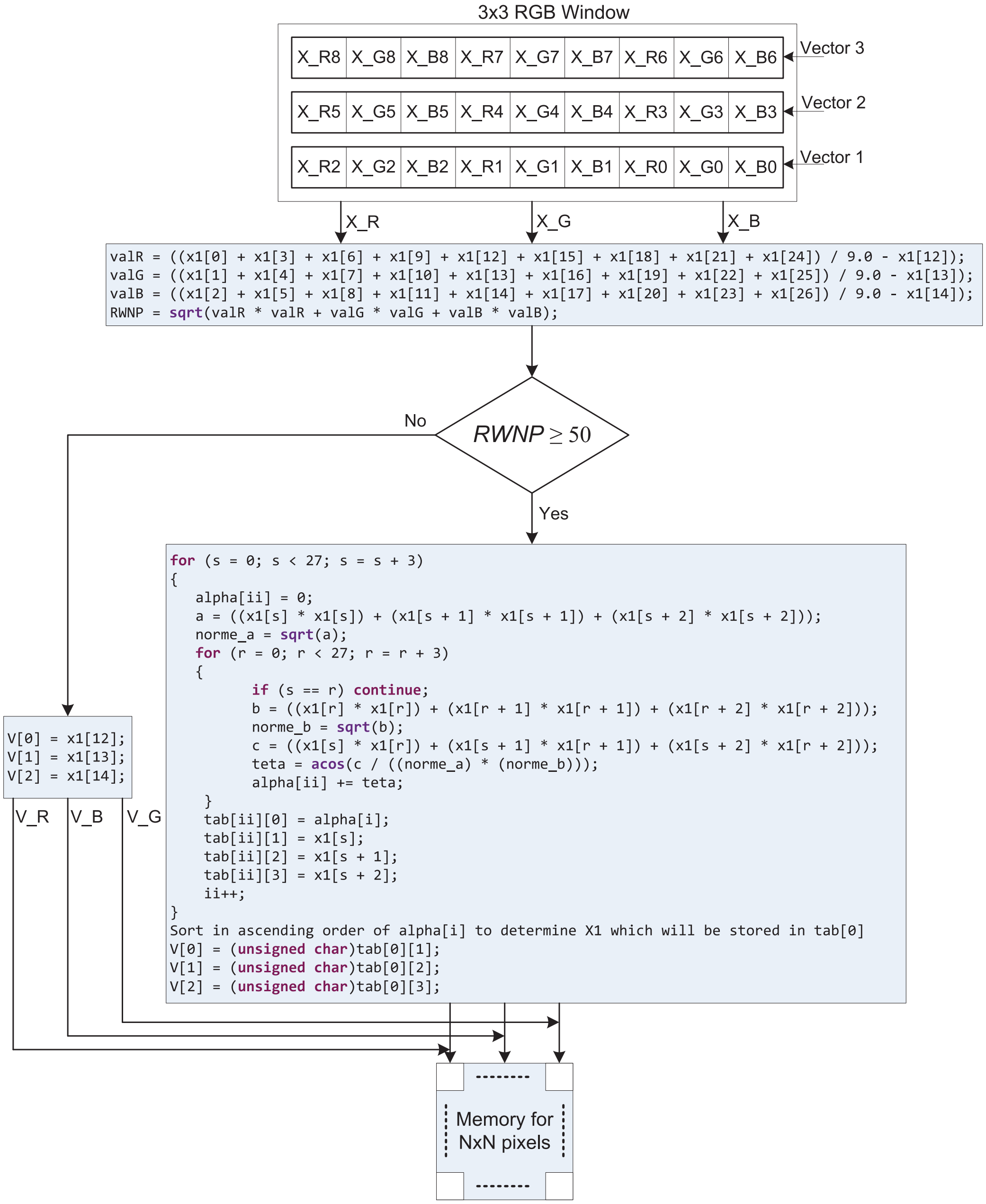


Figure 8: Comparison of the visual image quality between (a) original image, (b) contaminated image with 10% of impulsive noise and restored image with (c) VDF, (d) GVDF and (e) MVDF

5 MVDF Hardware Architecture

The HLS represents a powerful tool allowing to develop a hardware architecture through high-level languages like C and C++ [30]. In this context, the Vivado HLS is developed by Xilinx to aid engineers in exploring design architecture in the shortest amount of time. In fact, this exploration is realized through the selection of the appropriate Paragma such as PARTITION, PIPELINE, RESOURCE provided by the Vivado HLS environment.

By the way, an optimized hardware architecture for the MVDF is designed through the Vivado HLS 18.1 tool. Fig. 9 depicts the structure of the C code provided as input to this tool. This code reads three image lines to construct the 3 × 3 RGB filter window, which is used to compute the *RWNP* value. From this value, the MVDF code can keep the central pixel unchanged or determine the filtered pixel through the process presented in Fig. 9. The provided RGB pixel is stored in the internal memory to restore the filtered image.


3x3 RGB Window
X_R8 X_G8 X_B8 X_R7 X_G7 X_B7 X_R6 X_G6 X_B6
Vector 3
X_R5 X_G5 X_B5 X_R4 X_G4 X_B4 X_R3 X_G3 X_B3
Vector 2
X_R2 X_G2 X_B2 X_R1 X_G1 X_B1 X_R0 X_G0 X_B0
Vector 1
X_R
X_G
X_B
No
RWNP ≥ 50
Yes
V_R
V_B
V_G
Memory for NxN pixels

```
valR = ((x1[0] + x1[3] + x1[6] + x1[9] + x1[12] + x1[15] + x1[18] + x1[21] + x1[24]) / 9.0 - x1[12]);
valG = ((x1[1] + x1[4] + x1[7] + x1[10] + x1[13] + x1[16] + x1[19] + x1[22] + x1[25]) / 9.0 - x1[13]);
valB = ((x1[2] + x1[5] + x1[8] + x1[11] + x1[14] + x1[17] + x1[20] + x1[23] + x1[26]) / 9.0 - x1[14]);
RWNP = sqrt(valR * valR + valG * valG + valB * valB);
```

```
V[0] = x1[12];
V[1] = x1[13];
V[2] = x1[14];
```

```
for (s = 0; s < 27; s = s + 3)
{
   alpha[ii] = 0;
   a = ((x1[s] * x1[s]) + (x1[s + 1] * x1[s + 1]) + (x1[s + 2] * x1[s + 2]));
   norme_a = sqrt(a);
   for (r = 0; r < 27; r = r + 3)
   {
         if (s == r) continue;
         b = ((x1[r] * x1[r]) + (x1[r + 1] * x1[r + 1]) + (x1[r + 2] * x1[r + 2]));
         norme_b = sqrt(b);
         c = ((x1[s] * x1[r]) + (x1[s + 1] * x1[r + 1]) + (x1[s + 2] * x1[r + 2]));
         teta = acos(c / ((norme_a) * (norme_b)));
         alpha[ii] += teta;
    }
    tab[ii][0] = alpha[i];
    tab[ii][1] = x1[s];
    tab[ii][2] = x1[s + 1];
    tab[ii][3] = x1[s + 2];
    ii++;
}
Sort in ascending order of alpha[i] to determine X1 which will be stored in tab[0]
V[0] = (unsigned char)tab[0][1];
V[1] = (unsigned char)tab[0][2];
V[2] = (unsigned char)tab[0][3];
```

Figure 9: Structure of the MVDF C code

The MVDF C code is synthesized without adding any Pragma through the Vivado HLS 18.1 tool to be implemented on the Zynq XCZU9EG Xilinx FPGA. Table 2 illustrates the HLS synthesis results of our code. From this table, we can notice that Design 1 uses 8% of Look-up-Table (LUTs), 2% of Flip-Flops (FFs), 7% of BRAM blocks, and 1% of DSP blocks, and needs 1722 ms to restore a

256 × 256 color image at 125 MHz. Therefore, the processing time should be reduced to reach a real-time throughput with the MVDF design. The PARTITION and PIPELINE Pragmas are applied to the C code to process the data in the pipeline. After adding these pragmas, we can observe that the processing time of Design 3 is reduced by 99% relative to Design 1 and that it can restore a 256 × 256 color image in 1 ms at 125 MHz. But, Design 3 uses more FPGA resources, with 88% of LUTs, 84% of FFs, and 89% of DSPs compared to Design 1. Thus, we have employed the RESOURCE Paragma to optimize the FPGA resources. Consequently, we can conclude from Table 2 that Design 4 through the RESOURCE Paragma permits us to reduce the number of LUTs by 44% and FFs by 34% compared to Design 3 and restore a 256 × 256 color image in 2 ms at 125 MHz. However, from these results, it is clear that Design 4 presents high data throughput compared to the CoreTM i7-1165G7@2.80 GHz intel processor, which needs 65 ms to restore a 256 × 256 color image. These results are given for the same image quality between the software and hardware implementations of the MVDF.

Table 2: HLS synthesis results on the Zynq XCZU9EG FPGA

	LUTs	FFs	BRAM_18K	DSP48E	Cycles	Pragmas
Design1	24233 (8%)	11242 (2%)	138 (7%)	49 (1%)	2152098822	Without
Design2	25462 (9%)	13282 (2%)	108 (5%)	55 (2%)	2140477378	PARTITION
Design3	196005 (71%)	71357 (13%)	122 (6%)	459 (18%)	1299687	PIPELINE
Design4	109246 (39%)	47101 (8%)	121 (6%)	439 (17%)	2470088	RESOURCE

6 Conclusion

In this paper, a new nonlinear filter is proposed for efficient color image restoration. This filter is called MVDF, and it allows to check if a pixel in the image is corrupted or not by noise before changing it to a denoising pixel. For that, the MVDF uses a threshold and the neighborhood pixels of the central pixel in 3 × 3 filter window to make the decision whether to change the central pixel or not. One of the important features of this filter is that, compared to VDF and GVDF, it provides a smooth and clear image without increasing the computational complexity or execution time. Moreover, it can be used for a wide range of applications, like satellite images, character and face recognition, medical imaging, and so on. In fact, the experimental results show that the proposed filter gives better image quality than VDF and GVDF in terms of both objective (PSNR and NCD) and subjective measures, with an average decrease of 65.63% in the execution time. These results are given in particular for color images with lower and moderate "salt and pepper" impulsive noise densities ranging from 3% to 20%. For real-time processing, the hardware architecture is proposed for the MVDF through HLS flow, which allows to restore the image in 2 ms at 125 MHz instead of 65 ms under the CoreTM i7-1165G7@2.80 GHz Intel processor.

Our future work will focus on using deep learning techniques to improve image quality restoration, especially for high levels of "salt and pepper" impulsive noise [31].

Funding Statement: This work was funded by the Deanship of Scientific Research at Jouf University (Kingdom of Saudi Arabia) under Grant No. DSR-2021-02-0393.

Conflicts of Interest: The author declares that he has no conflicts of interest to report regarding the present study.

References

[1] B. B. Ahamed, D. Yuvaraj and S. S. Priya, “Image denoising with linear and non-linear filters,” in *Proc. ICCIKE*, Dubai, United Arab Emirates, pp. 806–810, 2019.

[2] P. E. Trahanias and A. N. Venetsanopoulos, “Vector directional filters-a new class of multichannel image processing filters,” *IEEE Transactions on Image Processing*, vol. 2, no. 4, pp. 528–534, 1993.

[3] R. Lukac, B. Smolka, K. N. Plataniotis and A. N. Venetsanopoulos, “Selection weighted vector directional filters,” *Computer Vision and Image Understanding*, vol. 94, no. 1, pp. 140–167, 2004.

[4] A. Ben Atitallah, I. Abid, A. Boudabous and H. Loukil, “A new hardware architecture of the adaptive vector median filter and validation in a hardware/software environment,” *International Journal of Circuit Theory and Applications*, vol. 49, no. 8, pp. 2329–2347, 2021.

[5] M. Kthiri, B. Le Gal, P. Kadionik and A. Ben Atitallah, “A very high throughput deblocking filter for H. 264/AVC,” *Journal of Signal Processing Systems*, vol. 73, no. 2, pp. 189–199, 2013.

[6] A. Ben Atitallah, M. Kammoun and R. Ben Atitallah, “An optimized FPGA design of inverse quantization and transform for HEVC decoding blocks and validation in an SW/HW environment,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 3, pp. 1656–1672, 2020.

[7] M. Kammoun, A. Ben Atitallah, K. M. Ali and R. Ben Atitallah, “Case study of an HEVC decoder application using high-level synthesis: Intra prediction, dequantization, and inverse transform blocks,” *Journal of Electronic Imaging*, vol. 28, no. 3, pp. 033010, 2019.

[8] M. A. Ben Atitallah, R. Kachouri, A. Ben Atitallah and H. Mnif, “An efficient HW/SW design for text extraction from complex color image,” *CMC-Computers, Materials & Continua*, vol. 71, no. 3, pp. 5963–5977, 2022.

[9] A. Ben Atitallah, M. Kammoun, K. M. A. Ali and R. Ben Atitallah, “An FPGA comparative study of high-level and low-level combined designs for HEVC intra, inverse quantization, and IDCT/IDST 2D modules,” *International Journal of Circuit Theory and Applications*, vol. 48, no. 8, pp. 1274–1290, 2020.

[10] T. R. Chanu, T. R. Singh and K. M. Singh, “A survey on impulse noise removal from color image,” *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 13, pp. 4274–4295, 2021.

[11] B. Smolka, “Robust sharpening vector median filter,” in *Proc. CACS*, Taoyuan, Taiwan, pp. 1–6, 2018.

[12] B. Smolka, “Robustified vector median filter,” in *Proc. ICCSE*, Vancouver, BC, Canada, pp. 362–367, 2014.

[13] R. Lukac, “Another generalisation of vector filters,” in *Proc. VIPROM*, Zadar, Croatia, pp. 273–278, 2002.

[14] R. Lukac, “Adaptive color image filtering based on center-weighted vector directional filters,” *Multidimensional Systems and Signal Processing*, vol. 15, no. 2, pp. 169–196, 2004.

[15] Z. Ma, H. R. Wu and D. Feng, “Partition-based vector filtering technique for suppression of noise in digital color images,” *IEEE Transactions on Image Processing*, vol. 15, no. 8, pp. 2324–2342, 2006.

[16] B. Smolka, “Efficient modification of the central weighted vector median filter,” in *Pattern Recognition*, Berlin, Heidelberg: Springer, 2002.

[17] B. Smolka, “Rank order weighted vector median filter,” in *Proc. IWSSIP*, Rio de Janeiro, Brazil, 2010.

[18] L. Khriji and M. Gabbouj, “Adaptive fuzzy order statistics-rational hybrid filters for color image processing,” *Fuzzy Sets and Systems*, vol. 128, no. 1, pp. 35–46, 2002.

[19] I. Abid, A. Boudabous and A. Ben Atitallah, “A new adaptive vector median rational hybrid filter for impulsive noise suppression,” in *Proc. STA*, Monastir, Tunisia, pp. 417–421, 2015.

[20] G. Wang, H. Zhu and Y. Wang, “Fuzzy decision filter for color images denoising,” *International Journal for Light and Electron Optics*, vol. 126, no. 20, pp. 2428–2432, 2015.

[21] V. Gregori, S. Morillas, B. Roig and A. Sapena, “Fuzzy averaging filter for impulse noise reduction in colour images with a correction step,” *Journal of Visual Communication and Image Representation*, vol. 55, pp. 518–528, 2018.

[22] A. Roy, L. Manam and R. H. Laskar, “Region adaptive fuzzy filter: An approach for removal of random-valued impulse noise,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 9, pp. 7268–7278, 2018.

[23] Y. Sato, N. Yamamoto, N. Inagaki, Y. Iesaki, T. Asamoto *et al.,* “Deep learning for bone mineral density and T-score prediction from chest X-rays: A multicenter study,” *Biomedicines*, vol. 10, no. 9, pp. 2323, 2022.

[24] A. A. Ramadhan and M. Baykara, “A novel approach to detect COVID-19: Enhanced deep learning models with convolutional neural networks,” *Applied Sciences*, vol. 12, no. 18, pp. 9325, 2022.

[25] N. Sharma, L. Saba, N. N. Khanna, M. K. Kalra, J. S. Suri *et al.,* “Segmentation-based classification deep learning model embedded with explainable AI for COVID-19 detection in chest X-ray scans,” *Diagnostics*, vol. 12, no. 9, pp. 2132, 2022.

[26] P. E. Trahanias, D. Karakos and A. N. Venetsanopoulos, “Directional processing of color images: Theory and experimental results,” *IEEE Transactions on Image Processing*, vol. 5, no. 6, pp. 868–880, 1996.

[27] A. Ben Atitallah, A. Boudabous, L. Khriji and N. Masmoudi, “Reconfigurable architecture of VDF filter for multidimensional data,” *International Journal of Circuit Theory and Applications*, vol. 41, no. 10, pp. 1047–1058, 2012.

[28] L. C. Dominguez and V. I. Ponomaryov, “Nonlinear filters for colour imaging implemented by DSP,” in *Proc. ICEEE*, Mexico City, Mexico, pp. 81–84, 2005.

[29] M. Emre Celebi, “Real-time implementation of order-statistics-based directional filters,” *IET Image Processing*, vol. 3, no. 1, pp. 1–9, 2009.

[30] T. M. Alanazi, A. Ben Atitallah and I. Abid, “An optimized SW/HW AVMF design based on high-level synthesis flow for color images,” *CMC-Computers Materials & Continua*, vol. 68, no. 3, pp. 2925–2943, 2021.

[31] C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo *et al.,* “Deep learning on image denoising: An overview,” *Neural Networks*, vol. 131, pp. 251–275, 2020.