Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
() : –

doi:10.3906/elk-2005-12

A novel hybrid decision-based filter and universal edge-based logical smoothing add-on to remove impulsive noise

Rajanbir Singh GHUMAAN, Prateek Jeet Singh SOHI, Nikhil SHARMA, and Bharat GARG*
Electronics and Communication Engineering, Thapar Institute of Engineering and Technology, Patiala, India

Received: 02.05.2020 • **Accepted/Published Online:** 14.04.2021 • **Final Version:** ..2021

Abstract: This paper presents a novel hybrid filter along with a universal extension to remove salt and pepper noise even at a very high noise density. The proposed filter initially specifies a threshold and then denoises the image using a combination of linear, nonlinear, and probabilistic techniques. Furthermore, to improve the quality, a universal add-on is presented which uses edge detection and smoothening techniques to brush out fine details from the restored image. To evaluate the efficacy, the proposed and existing filtering techniques are implemented in MATLAB and simulated with benchmark images. The simulation results show that the proposed filter is able to restore image details even at the extremely high noise density of 99%. Moreover, the proposed filter provides admirable results on natural as well as medical images from very low to very high noise density. Finally, it is observed that, on average, the proposed filter improves the PSNR by 11% over the state-of-the-art technique.

Key words: Salt and pepper noise, impulse noise, median filters, mean filter, image processing

1. Introduction

Images are often corrupted with random noise when it is transmitted through a noisy medium. The major constituent of this noise is salt and pepper or impulse noise. This is a phenomenon in which pixels in the image randomly obtain extreme values, e.g., 0 and 255 [1]. This in turn, results in transfer of false information. To overcome this issue, various linear and nonlinear filters have been introduced over the past three decades [2, 3]. Furthermore, it is observed that nonlinear median filters produce better outputs. This is because most pixels have higher correlation with their neighboring pixels, and the median filter utilizes this fact more beneficially than the mean filter [1].

The advance filtering techniques include value-weighted filtering that calculates weighted average based on the values gathered from the currently processing window [4]. Other techniques like interpolation-based filters are also presented, but their performance is not satisfactory at high noise densities because the number of pixels with original value (noise-free) is reduced [5, 6]. A combination of mean and median filters is also used; for example, based on noise density, the filter decides whether to apply the mean or median operation to restore the noisy pixel [7]. In most cases, mean operation is used at high noise densities, whereas median operation is used at low noise densities. However, these filters cannot provide better results at higher noise densities, e.g., 95% and 97% [7–10]. Therefore, to overcome this problem, a novel filtering technique is proposed with following features:

*Correspondence: bharat.garg@thapar.edu

1. We have integrated three major concepts, namely weighted median, noise density-based filtering, and iterative filtering, to improve the overall image quality of the filter images.

2. The proposed edge-based logical smoothener acts as a universal add-on that improves the quality of images obtained from various filtering techniques at high noise densities.

3 We have concatenated concepts such as canny edge detection, mean blur, and gaussian blur to improve image quality by using multiple thresholding criteria.

The rest of the paper is organized as follows. Section 2 provides detailed literature review while Section 3 presents the proposed filtering approach. The simulation results and comparative analysis are given in Section 4, and Section 5 concludes the paper.

2. Related work

A number of noise removal filters have been introduced to overcome the limitations of mean and median filters using a more complex combinational approach as summarized in Figure 1.

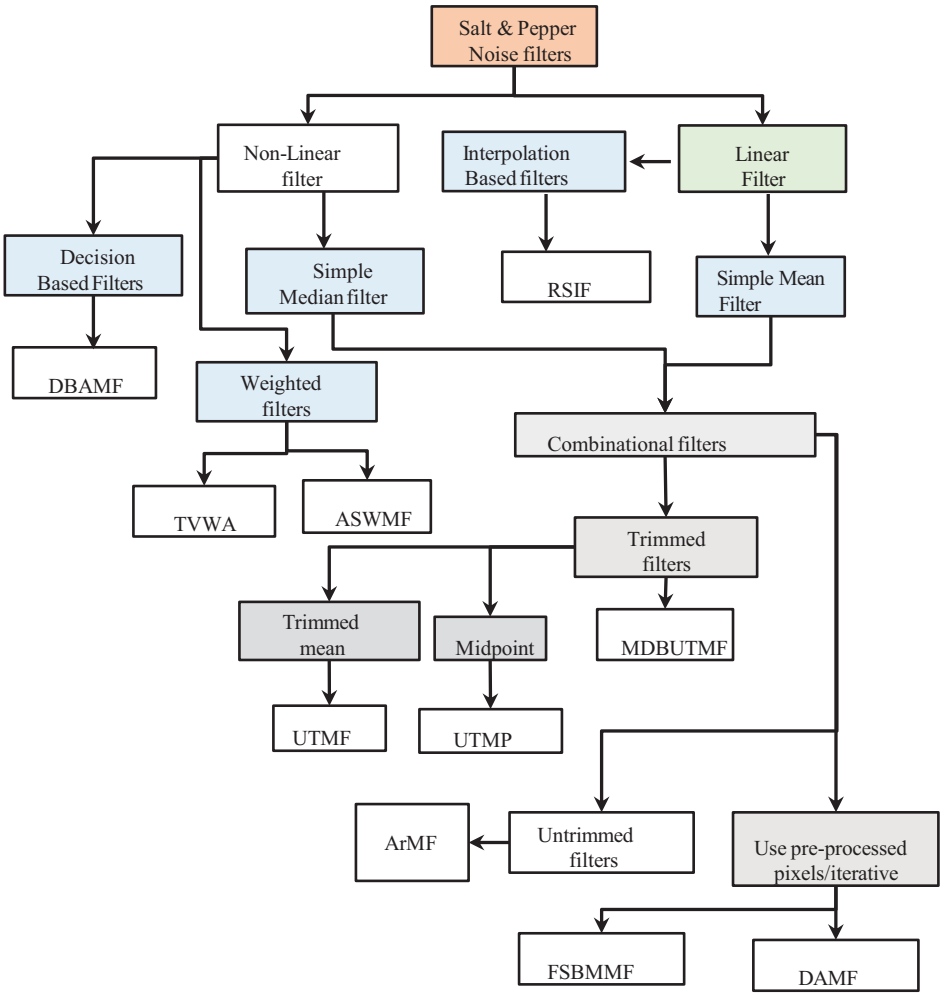


Figure 1. Flowchart of the related work.

Decision-based filters such as decision-based adaptive median filter (DBAMF) [11] apply sorting such that the central pixel is the median of the 3×3 window, if no noise-free pixel is found, it is replaced by the nearest nonnoisy pixel. However, at high noise densities, when the noise-free pixels are rare, the candidate noisy pixel is replaced by the nearest nonnoisy pixel. This results in repetition of the same pixel in the image. Some trimmed filters such as unsymmetric trimmed mean filter (UTMF) [12] of the trimmed mean category and unsymmetric trimmed midpoint filter (UTMP) of the trimmed midpoint subcategory are presented to overcome the shortcomings of DBAMF. The UTMP has a better performance at low noise densities, whereas UTMF performs better at high noise densities. The UTMF uses mean of nonextreme values of 3×3 window, whereas UTMP uses midpoint of nonextreme pixels. Both filters have good results in their respective domains, but they miserably fail if considered separately.

To overcome the abovementioned problems in the trimmed filter category, modified decision-based unsymmetric trimmed median filter (MDBUTMF) [8] that makes substantial use of trimming technique was presented. In this technique, the noisy pixel is replaced by the median of 3×3 trimmed window constructed around it. Furthermore, if a zero size matrix is obtained by trimming operation, the noisy pixel is replaced by the mean of untrimmed 3×3 window. The small window size ensures high levels of edge preservation. However, at high noise densities, a large amount of pixels get replaced by random values upon performing the mean operation. Another member of this family of combinational filters, belonging to the subcategory of untrimmed filters is fast switching-based mean median filter (FSBMMF) [7]. This filter is the first of its kind to make use of previously processed pixels for estimating pixel values that are otherwise left unprocessed during the median calculation phase. This filter denoises each noisy pixel $(x_{i,j})$ based on the following expression. Here, med3, med5, mean5, and ppp represent median operation with 3×3, 5×5, mean operation with 5×5 window, and previously processed pixel operation, respectively.

$$y_{i,j} = \begin{cases} med3(x_{i,j}), & \text{if } med3(x_{i,j}) \text{ is NFP.} \\ med5(x_{i,j}), & \text{elseif } med(x_{i,j}) \text{ is NFP.} \\ mean5(x_{i,j}), & \text{elseif, } x_{i,j} \text{ does not belong to boundary} \\ PPP(x_{i,j}), & \text{else, previously processed pixel of that row/col} \end{cases} \tag{1}$$

The interpolation-based filters such as recursive cubic spline interpolation filter (RSIF) [10] are very similar to MDBUTMF [8], the only difference being, instead of performing median operation on a 3×3 window, it takes a 3×3 window, and if all the pixels of this window are noisy, then the mean of this window replaces noisy pixels, else cubic spline interpolation is performed.

$$y_{i,j} = \begin{cases} mean3(nImg), & \text{if } w_{3\times3} \text{ contains noisy pixels only} \\ a_i \times x^2 + b_i \times x^2 + c_i \times x + d_i, & \text{Cubic spline interpolation} \end{cases} \tag{2}$$

Owing to the mathematical superiority of the estimation technique, this filter performs extremely well when compared to its predecessors mentioned before. On the contrary, it requires large execution time and provides blurred output image due to denoising using the mean value under very high noise density conditions.

Adaptive switching median filter (ASWMF) [4] of the weighted filter subcategory has a different approach for removing salt and pepper noise. Here every pixel has to meet certain criterion to be classified as a noisy pixel. If the pixel is found to be noiseless, then it is left unprocessed. Otherwise, the pixel is estimated by taking the median of a variable sized window depending on the length of the window (even or odd). If the

trimmed window length is even, then the most reoccurring pixel of this window is repeated once again, and if there is no reoccurring element, then the nearest nonnoisy pixel is repeated. This makes the window length odd, and the median of this updated window is used to replace the corrupted pixel. An improved approach of the weighted filter subcategory is the three value weighted approach (TVWA) [13] which takes into consideration a variable size trimmed window. The pixels of this window are divided into three groups, namely minimum, maximum, and middle, based on their closeness to the same. The noisy pixel is then replaced by sum of products of weighting factors with their maximum, middle, and minimum values of respective groups. The size of the processing window is increased to a maximum size of 7×7 until a noise-free pixel is detected. The larger window sizes such as 7×7 and 5×5 lead to loss of edge details, which results in blurred images.

The adaptive Riesz mean filter (ARMF [25]) initially creates a binary bitmap based on whether a given pixel is noisy or nonnoisy, it assigns the pixel a value of 0 or 1, respectively. It takes into consideration a window of variable size, the size of which is determined by values in its corresponding bitmap. If all values in a given bitmap symbolize noisy pixels, then processing window size is increased to incorporate nonnoisy pixels. Once a window size is decided, the processing pixel is replaced by Riesz mean of the selected window.

Recently, different applied median filter (DAMF) [14] of the preprocessed/iterative subcategory has proven to be a state-of-the-art filter. DAMF distinguishes between corrupted and original values by equating corrupted pixels to logic 1 and original pixels to logic 0. This, in turn, results in a binary image. Then the pixels with logic 1 are processed by considering a 3×3 window, the size of this window is linearly increased until the point where at least one uncorrupted pixel is present. There is only one condition that the size of the mask must not exceed 7×7. Then the noisy pixel is replaced by the median of the trimmed window values. Once the entire image is filtered in this way, the resulting image is again processed by first converting it into a binary image. Then the entire image is reprocessed in a similar fashion as before, the only difference being that the window size is kept constant at 3×3. The major reason for this filter's success is its optimized use of preprocessed pixels unlike the FSBMMF filter where only four previously processed pixels are available for estimation. This filter provides eight preprocessed pixels during the second iteration. This technique also takes special care for lack of information present in the corner and edges by utilizing symmetric padding instead of zero padding. The only observable drawback is the use of large window sizes in the first step, which adversely affects the edges in the processed image, thus resulting in a blurred image. Along with these filters, various other filters have been proposed, and they had one of the issues above [1–30].

To resolve the abovementioned drawbacks, the next section presents an novel hybrid decision-based algorithm (HDBF) that effectively eliminates impulse noise while preserving the detailed image information. In addition to this, a universal add-on named edge-based logical smoothener (EBLS) is proposed for image enhancement at high noise densities.

3. Proposed hybrid decision-based filtering (HDBF) approach

The following subsections first present the flowchart of HDBF followed by its algorithm and finally the flowchart of EBLS.

3.1. Flowchart of the proposed filtering approach

The flowchart of the proposed filter is shown in Figure 2. Initially, the threshold variables and flags are initialized as shown in Figure 2, and the noise density (ND) is calculated using the formula given below. Here $nImg_{x,y}$

are the pixel values of the nImg, and M × N represents the image size:

$$= \begin{cases} 1, & nImg_{x,y} \in 0, 255 \\ 0, & Otherwise \end{cases} \tag{3}$$

$$ND = \frac{\sum_{x=N}^{x=M} \sum_{y=0}^{y=N} (x, y)}{M \times N} \times 100\% \tag{4}$$

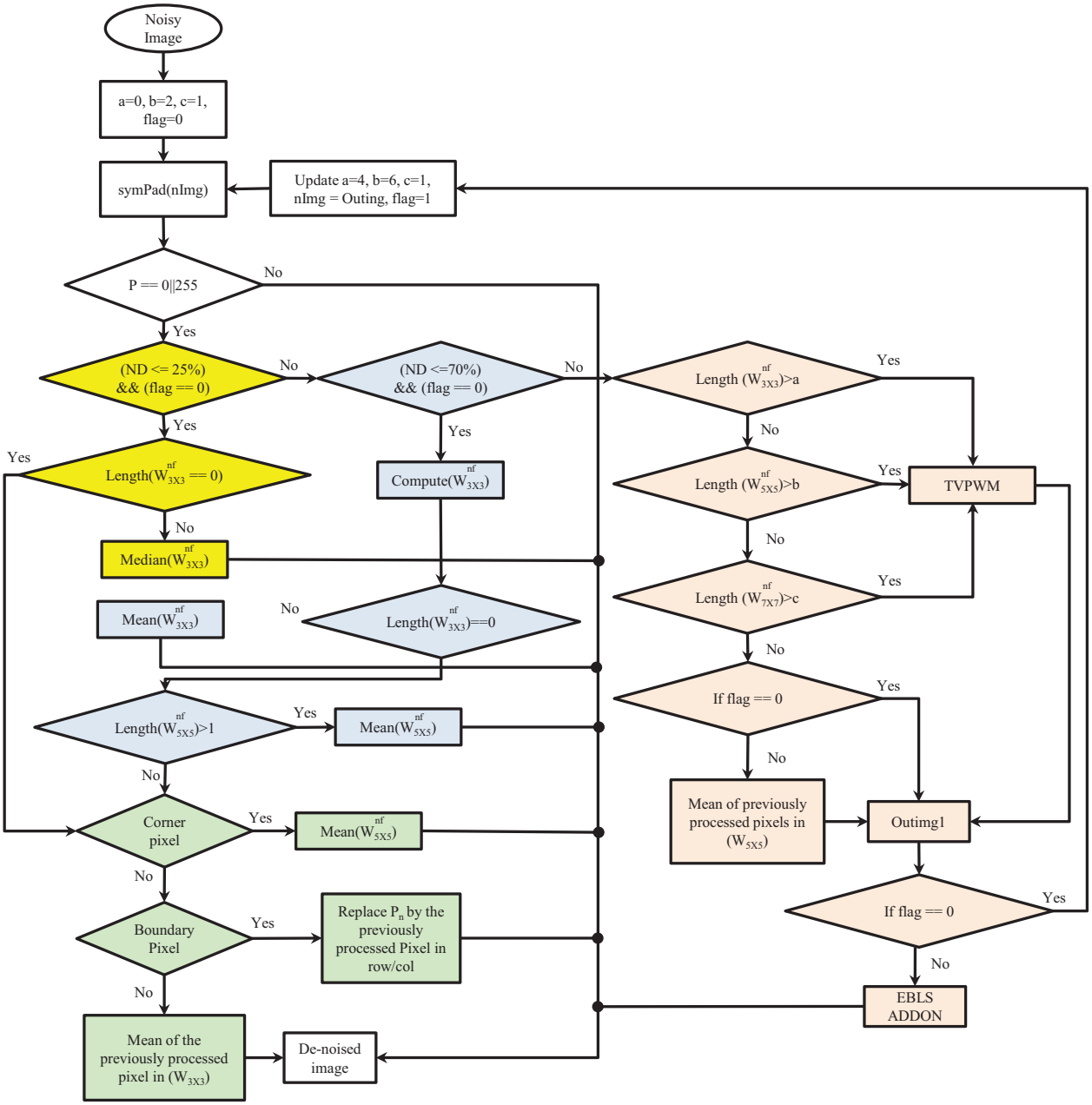


Figure 2. Flowchart of the proposed filter.

The proposed approach performs different operations under different noise density conditions as follows:

1. **ND $<$ 25% condition**: Under this condition, if the flag is zero (initially its value is zero), then the noisy pixel is replaced by the median of a 3×3 trimmed window. The process of obtaining a trimmed window from window $W_{3\times3}$ is explained in the example given below in Figure 3:

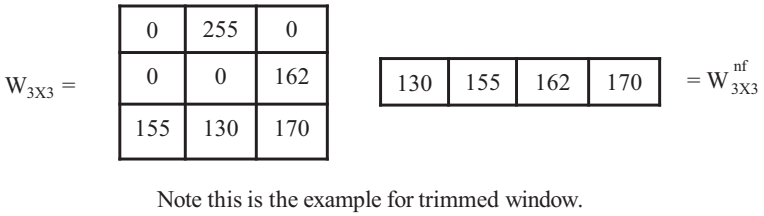


Figure 3. Example of a trimmed window.

If the trimmed window is found to be empty and the candidate pixel is a corner pixel, then the mean of trimmed 5×5 window replaces the noisy pixel. However, under the same circumstance, if the corrupted pixel is a boundary pixel, then the previously processed pixel of that very row or column replaces the noisy pixel. However, a pixel from the interior of the image is replaced by the mean of previously processed pixels in a 3×3 window. Under this condition, the denoising process is performed by the steps highlighted in yellow and green in Figure 2.

2. **25% $<$ ND $>$ 70% condition**: The corrupted pixel is replaced with the mean of trimmed 3×3 window constructed across it. If the window size is found to be zero, then the window size is increased to 5×5, and it is checked if its size is greater than one, if so the corrupted pixel is replace by its mean. Otherwise, the steps are followed as done for the noise density less than 25%. The denoising steps performed under this condition are shown in blue and green in Figure 2.

3. **ND $>$ 70% condition**: If the length of the trimmed 3×3 window is more than the first threshold variable ($length(W^{nf}_{3\times3}) > a$), then it executes the three valued probability-based weighted mean (TVPWM) function as per Algorithm 1 given in the next subsection. In case the first threshold condition is not met ($length(W^{nf}_{3\times3}) < a$), the window size is increased, and the length of trimmed 5×5 is checked against the second threshold variable ($length(W^{nf}_{5\times5}) > b$), and TVPWM is executed. Now if the second threshold condition is not met ($length(W^{nf}_{5\times5}) < b$), the same procedure is repeated for window of size 7×7, and the length of trimmed 7×7 is checked against the third threshold variable ($length(W^{nf}_{7\times7}) > c$). After the entire image has been processed, the threshold and flag values are updated. We then perform the same process as before. Now the image obtained at the end of the second iteration is passed through the proposed EBLS add-on logic presented in subsection 3.3 to further improve the edge information of the reconstructed image. For this condition, the steps are highlighted in orange in Figure 2.

3.2. Proposed TVPWM function

The pseudocode of the TVPWM function is shown in Algorithm 1. The algorithm divides the elements of the trimmed window of size $\alpha \times \alpha$ into three sets, where α is the window size obtained form the previous step.

Algorithm 1 TVPWM

Input $W_{\alpha\times\alpha}$ ▷ *Input Trimmed window across corrupted pixel of size* $\alpha \times \alpha$
Output Pix ▷ *Restored pixel value*
Initialize: $\gamma_1 \leftarrow 0; \gamma_2 \leftarrow 0;$
Initialize: $\delta_1 \leftarrow [\,]; \delta_2 \leftarrow [\,];$
for each P_i **do**
    $\gamma_1 \leftarrow |\beta_{max} - P_i|;$
    $\gamma_2 \leftarrow |\beta_{min} - P_i|;$
    if $\gamma_1 \leq \gamma_2$ **then**
        $\delta_1 \leftarrow [\delta_1, P_i];$
    else
        $\delta_2 \leftarrow [\delta_2, P_i];$
    end if
end for
$P_{\delta_1} \leftarrow \text{length}(\delta_1)/\text{length}(W_c);$
$P_{\delta_2} \leftarrow \text{length}(\delta_2)/\text{length}(W_c);$
$\beta_{mid} = P_{\delta_1} \times \beta_{max} + P_{\delta_2} \times \beta_{min};$
Return: $\beta_{mid};$
Initialize: $\gamma_1 \leftarrow 0; \gamma_2 \leftarrow 0; \gamma_3 \leftarrow 0;$
Initialize: $\delta_1 \leftarrow [\,]; \delta_2 \leftarrow [\,]; \delta_3 \leftarrow [\,];$
for each P_i **do**
    $\gamma_1 \leftarrow |\beta_{max} - P_i|;$
    $\gamma_2 \leftarrow |\beta_{min} - P_i|;$
    $\gamma_3 \leftarrow |\beta_{mid} - P_i|;$
    if $\gamma_1 \leq \gamma_2$ and $\gamma_1 \leq \gamma_3$ **then**
        $\delta_1 \leftarrow [\delta_1, P_i];$
    else if $\gamma_2 \leq \gamma_1$ and $\gamma_2 \leq \gamma_3$ **then**
        $\delta_2 \leftarrow [\delta_2, P_i];$
    else
        $\delta_3 \leftarrow [\delta_3, P_i];$
    end if
end for
$P_{\delta_1} \leftarrow \text{length}(\delta_1)/\text{length}(W_c);$
$P_{\delta_2} \leftarrow \text{length}(\delta_2)/\text{length}(W_c);$
$P_{\delta_3} \leftarrow \text{length}(\delta_3)/\text{length}(W_c);$
$Pix \leftarrow P_{\delta_1} \times \beta_{max} + P_{\delta_2} \times \beta_{min} + P_{\delta_3} \times \beta_{mid};$
Return: $Pix;$

Initially, the maximum and minimum values within the window are computed, and then on the basis of these values, the elements of the trimmed window are divided into two sets, where the values close to the maximum and minimum are stored in δ_1 and δ_2, respectively. The middle value of this set is calculated using Eq. 5.

$$\beta_{mid} = P_{\delta_1} \times \beta_{max} + P_{\delta_2} \times \beta_{min} \tag{5}$$

Here P_{δ_1} and P_{δ_2} are the probability of finding elements of δ_1 and δ_1, respectively, in the trimmed window, whereas β_{max} and β_{min} are maximum and minimum values, respectively. Now, based on the middle value β_{mid}, we divide the elements of the trimmed window into three groups $\delta_1, \delta_2, \delta_3$ where each of these groups store the value closest to the maximum, minimum, and middle value respectively. Finally, the processed

pixel is given by Eq. (6).

$$Pix = P_{\delta_1} \times \beta_{max} + P_{\delta_2} \times \beta_{min} + P_{\delta_3} \times \beta_{mid} \tag{6}$$

where P_{δ_1}, P_{δ_2}, and P_{δ_3} represent the probability of finding an element of $\delta_1, \delta_2, \delta_3$ in the trimmed window, respectively. To further improve the performance of the filter, an extension named edge-based local smoothener is proposed in the next subsection.

3.3. Illustration for the proposed EBLS add-on

The EBLS effectively calculates the edges of the image using canny edge detection [15] and then makes accurate decision to use different smoothener for different parts of the image. The flowchart of the proposed EBLS add-on is shown in Figure 4. Steps to perform the same are given below:

1. The image is symmetric padded.
2. Canny edge detection is used to extract the edges of the image.
3. Two thresholds for applying the canny method are initialized: a high and a low threshold for low and high edge sensitivity, respectively. The edge with lower sensitivity is started and then flattened to incorporate associate edge pixels from the high sensitivity result. This fills the gaps between the detected edges and provides an edged image. In this case, the default value of thresholds are chosen. The resulting image is stored in matrix E3.
4. Then image dilation [17] (used to expand the pixels in an image) is performed to bold out the edges and to extract most of the information out of the edges. To perform image dilation, 2D convolution of edged image is done with 5×5 Gaussian kernel with rethreshold of 0.2, and the result is stored in E4.

$$kernel = \begin{bmatrix} 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 2 \\ 4 & 8 & 16 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix} \tag{7}$$

5. Logical smoothing is performed on the dilated image where based on the pixel value 1 or 0 (edged region), the average or Gaussian smoothing is performed respectively on the image.

EBLS could be applied to almost every salt and pepper filter after some value of threshold. In the next section, the performance of the proposed EBLS is explained in a detail along with the table that shows improved performance in terms of PSNR. The next section presents a comparative analysis of simulation results to evaluate the efficacy of the proposed filter over the existing.

3.4. Illustrations of the proposed algorithm

How the filter works on different matrices at different noise densities can be seen in Figures 5–7. To understand how the proposed filtering algorithm works, let us consider the noisy pixel 255 located at $(1,1)$ in Figure 5. Since the trimmed 3×3 window around this noisy pixel does not have any noise-free pixels, window size of 5×5 is considered. The median of this 5×5 restores the noisy pixel. To denoise the pixel located at $(1,2)$, a 3×3 window is considered. Since the trimmed window around this pixel does not have any noise-free pixels and since this pixel belongs to the first row (i.e. boundary pixel), this pixel is restored by the previously processed pixel which is 162. Similarly, the proposed algorithm denoinses other noisy pixels as shown in Figures 5–7.

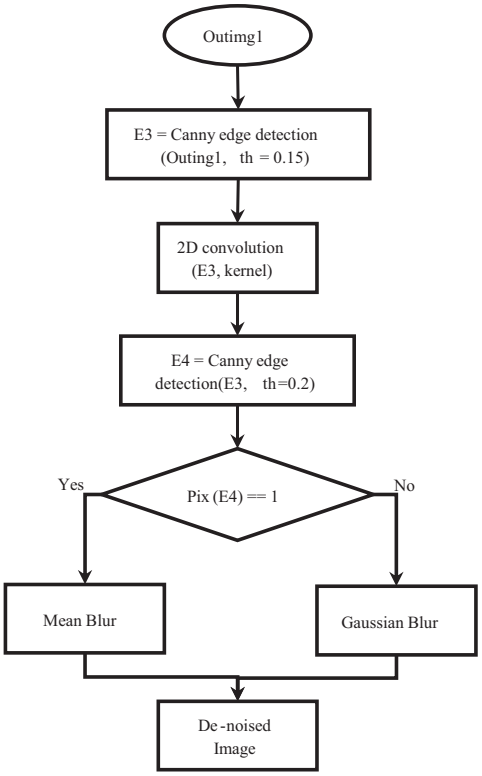


Figure 4. Flowchart of the proposed EBLS add-on.

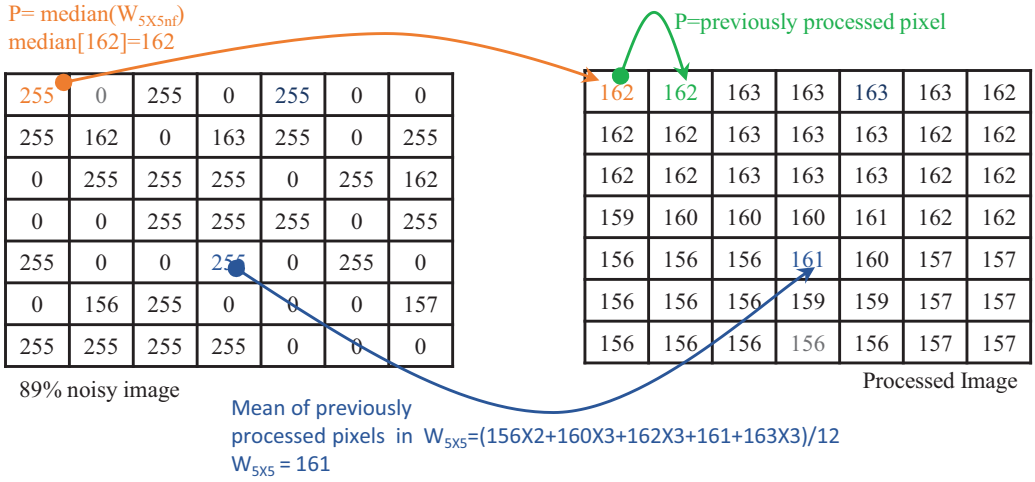


Figure 5. Operation at low noise density ($\leq$ 25).

4. Simulation

The simulation is performed for the proposed and other state-of-the-art filters, namely ArMF[25], MDBUTMF, FSBMMF, RSIF, TVWA, ASWMF, and DAMF. Different benchmark images are considered for analysis. Initially, the salt and pepper noise-corrupted images with varying noise densities (10% to 99%) are generated

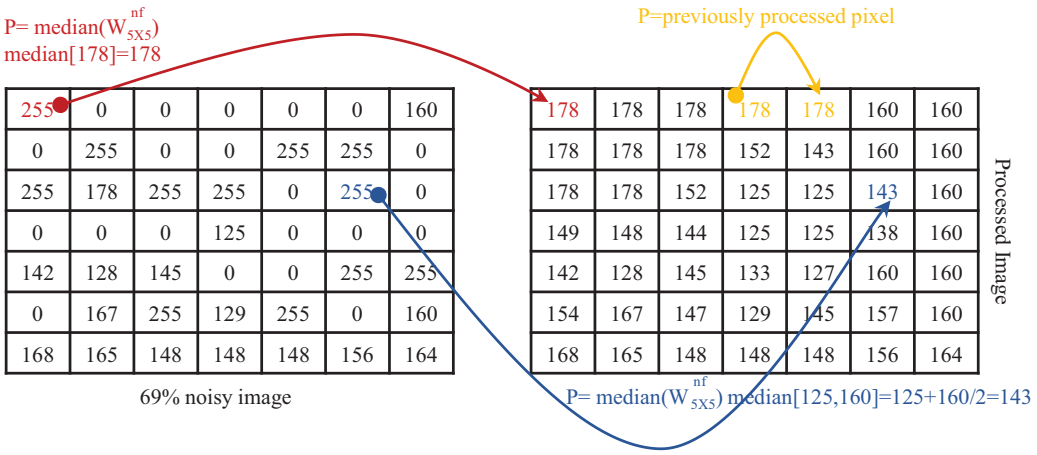


Figure 6. Operation at high noise density (25 < noise density < 70).

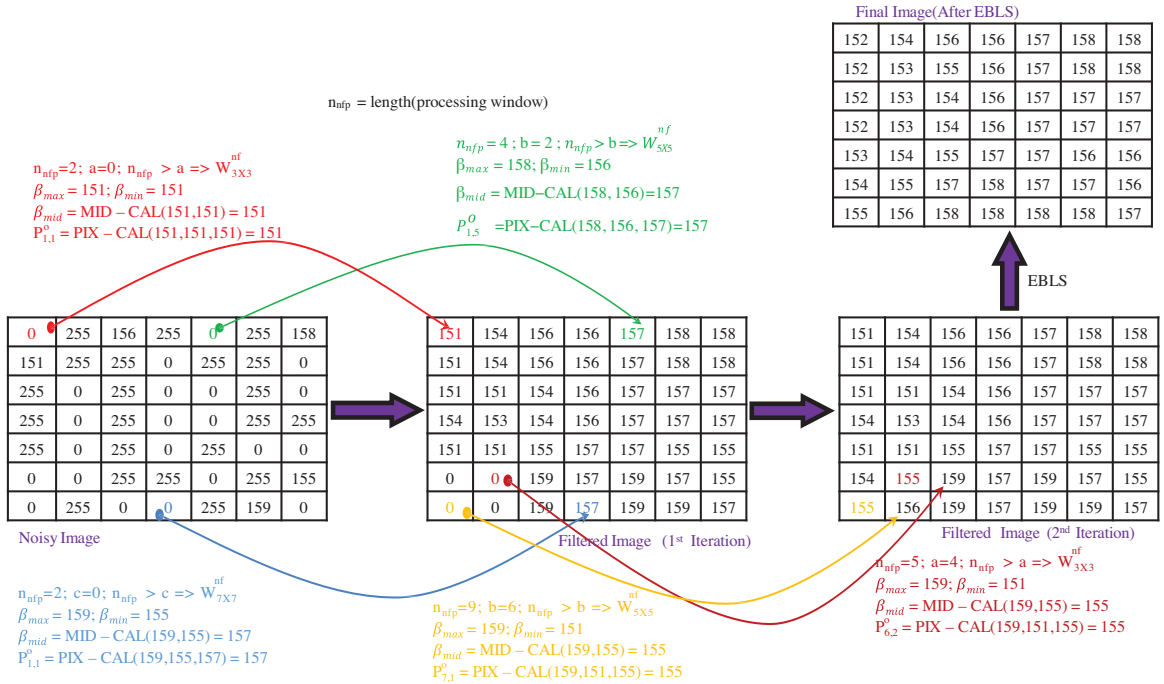


Figure 7. Operation at higher noise density (≥ 70).

and then filtered using the proposed and existing filters. For the quantitative analysis, the peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) values are calculated. The mathematical expression of PSNR is given by Eq. (8).

$$PSNR = 10 \times log_{10} \frac{MAX^2}{MSE} \tag{8}$$

where MSE is the mean square error and Max is the max pixel value residing in the image. In case of 8-bit

images, the value of Max is 255.

$$MSE = \frac{1}{M \times N} \times \sum_{i=1}^{M} \sum_{j=1}^{N} (x_{i,j} - y_{i,j}) \tag{9}$$

Parameters M and N represent dimensions of the input and output image x and y. Here $x_{i,j}$ and $y_{i,j}$ represent the pixel values at positions (i, j), respectively.

$$SSIM(x, y) = \frac{(2 \times \mu_x \times \mu_y + C_1) \times (2 \times \sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) \times (\sigma_x^2 + \sigma_y^2 + C_2)} \tag{10}$$

$$\text{Standard Deviation}(\sigma) = \sqrt{\frac{1}{N} \times \sum_{i=1}^{N} (x_i - \mu)^2} \tag{11}$$

where μ_x (σ_x) and μ_y (σ_y) represent the mean (standard deviation) value in x and y directions, respectively. Furthermore, C_1 and C_2 represent the constants which are considered to limit the SSIM value to 1.

The following subsections provide the qualitative and quantitative analyses of the proposed filter over the existing with different benchmark images.

1. Colored Peppers image of size 512×512.
2. Medical image Thigh (256×256) and Lungs (425×425).
3. Average of Lena, Boat, and Zelda images (512×512) and Kodak images.
4. Performances of other filters using EBLS on Lena.

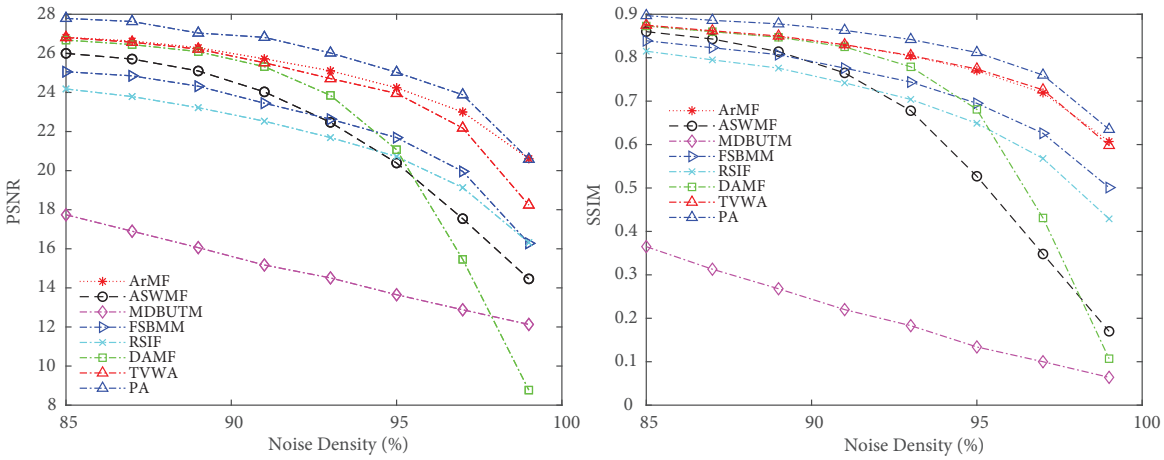
4.1. Simulation result analysis with colored Peppers image

Table 1 shows PSNR and SSIM values of colored Peppers image (512×512) filtered using the proposed and existing filters with noise densities varying from 85% to 99%. From the results, we can confirm that the proposed algorithm gives better results even for colored images with an average of 0.5 dB and 0.015 increase in PSNR and SSIM values, respectively. For colored images too, at 97% and 99%, the performance of the filter is exceptionally well. This shows the filter's efficacy for all kind of images with very high noise densities. Figures 8a and 8b present the plot for PSNR and SSIM of the Peppers image for varying noise densities, respectively. Figures 9a and 9b represent original nonnoisy and noisy (with 85% noise density) images, respectively. Figures 9c–9j provide restored Peppers images for visual analysis using different filters at 85% noise density. It is evident that the proposed filter can precisely restore edges throughout the entire range of noise densities.

To check the computational cost, the execution time of all the filters are computed with 10% to 90% noise density. From Table 1, we can infer that at lower noise densities, the time complexity of the filter is comparable with most of the other filters. At higher noise densities, the algorithm takes a slightly longer time, but better performance is achieved at the expense of higher computational cost.

Table 1. PSNR and SSIM of the filtered colored images using the proposed and existing filters.

Metric	Noise (%)	ArMF	ASWMF	MDBUTM	FSBMM	RSIF	DAMF	TVWA	PA
PSNR	85	26.82	26.00	17.74	25.06	24.18	26.68	26.81	27.79
	87	26.63	25.71	16.90	24.85	23.79	26.45	26.57	27.63
	89	26.29	25.10	16.06	24.32	23.22	26.09	26.22	27.04
	91	25.72	24.03	15.17	23.45	22.53	25.32	25.52	26.82
	93	25.11	22.47	14.51	22.63	21.69	23.85	24.71	26.02
	95	24.24	20.39	13.65	21.69	20.72	21.07	23.95	25.03
	97	23.00	17.54	12.88	19.96	19.13	15.46	22.18	23.88
	99	20.61	14.46	12.13	16.28	16.32	8.77	18.24	20.58
SSIM	85	0.873	0.860	0.365	0.839	0.815	0.873	0.875	0.897
	87	0.860	0.843	0.313	0.823	0.795	0.860	0.862	0.886
	89	0.848	0.814	0.268	0.807	0.776	0.847	0.850	0.878
	91	0.829	0.765	0.220	0.776	0.742	0.825	0.830	0.863
	93	0.804	0.678	0.183	0.744	0.704	0.779	0.805	0.842
	95	0.770	0.527	0.134	0.695	0.649	0.681	0.774	0.812
	97	0.719	0.348	0.100	0.626	0.568	0.431	0.726	0.760
	99	0.607	0.170	0.064	0.501	0.429	0.107	0.598	0.635
Execution time (s)	10	0.242	1.852	0.186	0.123	3.176	0.194	0.493	0.359
	20	0.138	2.322	0.200	0.091	4.167	0.345	0.534	0.420
	30	0.247	3.584	0.359	0.158	5.336	0.277	0.556	0.515
	40	0.294	3.179	0.346	0.150	6.594	0.291	0.470	0.517
	50	0.217	3.569	0.346	0.156	8.281	0.304	0.437	0.504
	60	0.214	4.157	0.394	0.153	7.627	0.344	0.482	0.550
	70	0.253	3.931	0.438	0.232	8.384	0.386	0.586	1.882
	80	0.342	4.401	0.474	0.200	9.576	0.446	0.678	1.747
	90	0.447	4.082	0.532	0.204	10.335	0.603	0.784	2.232
	Average	0.266	3.453	0.364	0.163	7.053	0.354	0.558	0.970



(a) PSNR (b) SSIM

Figure 8. PSNR and SSIM plots of colored Peppers image filtered using various filters at different noise densities.

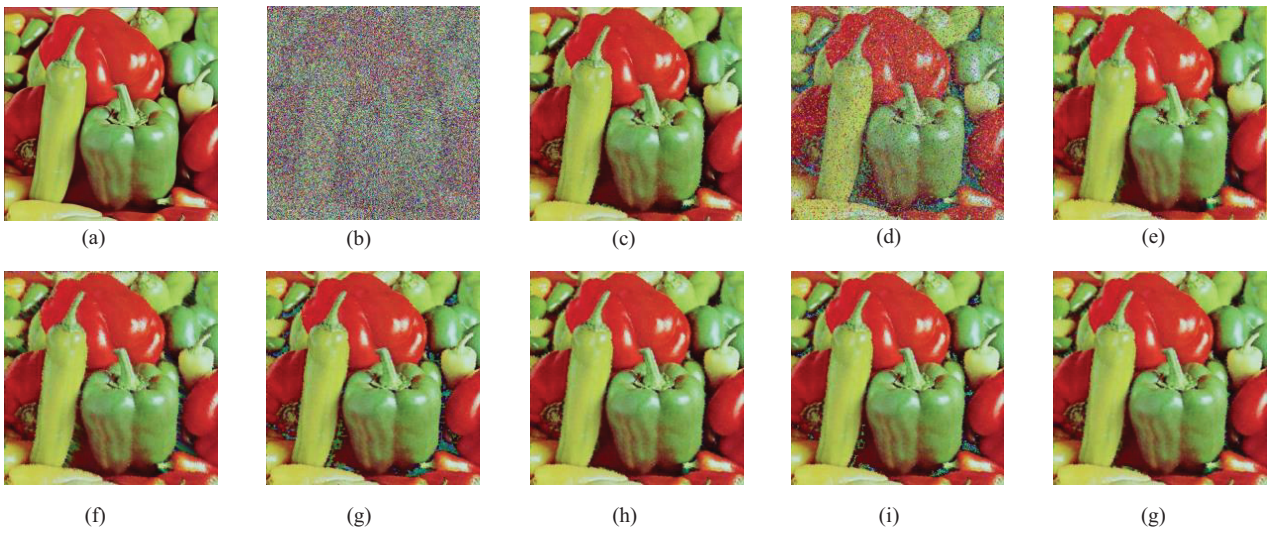


Figure 9. Colored Peppers image (a) original, (b) with 85% noise density, and filtered images using: (c) ARmF, (d) MDBUTM, (e) FSBMM, (f) RSIF, (g) TVWA, (h) ASWMF, (i) DAMF, and (j) the proposed filter.

4.2. Analysis on medical images

This subsection consists of quantitative analysis on a medical thigh image (256×256) in terms of PSNR and SSIM and visual representation of lungs image (425×425) at noise density of 97%. Table 2 clearly depicts that the proposed filter produces improved results over a wide range of noise densities (i.e. 10% to 90%). Figures 10a and 10b provide a graphical representation for the same. Figures 11a and 11b represent the original nonnoisy and noisy (with 95% noise density) images, respectively. Figures 11c–11j show the visual analysis of restored lungs image at 97% noise density using various other filters and the proposed filter. The proposed algorithm produces the best quality restored images with better feature preservation, lower streaking, and significantly lower blurring.

4.3. Comparative analysis of grey-scale images

Table 3 encapsulates the average PSNR and SSIM values for the proposed and the existing state-of-the-art filters for Lena, Boat, and Zelda images (512×512) with noise density shifting from 75% to 99%. The simulation outcomes demonstrate that the proposed algorithms have significantly performed better in terms of PSNR and SSIM as compared to the other filters even at high noise densities like 97% and 99% with 5% average improvement in PSNR. Figures 12a and 12b provide the plot for PSNR of the same. The trend evidently depicts that the proposed filter surpasses the PSNR and SSIM of other methods. Figures 13a and 13b represent the original nonnoisy and noisy (with 91% noise density) images, respectively. Figures 13c–13j show pictorial representation of the Lena image filtered using the proposed as well as the existing filters at 91% salt and pepper noise. Therefore, the proposed filter can retrieve the details of an image better than the existing filters and is very efficient in noise removal at high noise densities.

To compare the performance on larger dataset, images are tested on kodak benchmark dataset containing 24 natural images of size 512 $\times$ 768 and 768 $\times$ 512 shown in Table 4. In this case too, the proposed HDBF

Table 2. Quality metrics of the filtered X-ray images using the proposed and existing filters.

Metrics	Noise (%)	ARmF	ASWMF	MDBUTM	FSBMMF	RSIF	DAMF	TVWA	PA
PSNR	10	35.82	35.30	33.05	36.80	35.69	30.71	35.26	37.82
	20	34.16	32.57	29.44	33.31	33.49	29.79	33.84	34.95
	30	32.74	29.99	26.54	31.04	31.00	27.52	32.00	32.92
	40	31.63	28.36	23.93	29.18	29.38	26.43	31.15	32.04
	50	30.43	26.69	21.83	28.01	27.87	24.72	30.16	31.22
	60	28.88	25.21	19.31	26.90	26.34	23.99	28.69	28.94
	70	27.30	23.12	16.89	25.87	24.68	22.34	27.27	27.80
	80	25.44	21.13	13.98	24.44	22.33	20.74	25.50	26.76
	90	22.92	17.68	11.20	22.44	19.72	17.97	23.12	24.53
SSIM	10	0.977	0.976	0.948	0.979	0.976	0.976	0.975	0.981
	20	0.970	0.963	0.918	0.962	0.966	0.967	0.967	0.976
	30	0.961	0.944	0.872	0.943	0.950	0.952	0.955	0.959
	40	0.949	0.928	0.804	0.920	0.931	0.939	0.943	0.949
	50	0.935	0.907	0.733	0.893	0.903	0.920	0.929	0.939
	60	0.914	0.875	0.618	0.862	0.868	0.894	0.906	0.910
	70	0.880	0.830	0.464	0.820	0.812	0.860	0.875	0.882
	80	0.831	0.758	0.281	0.762	0.724	0.809	0.825	0.835
	90	0.737	0.581	0.137	0.658	0.569	0.693	0.740	0.771

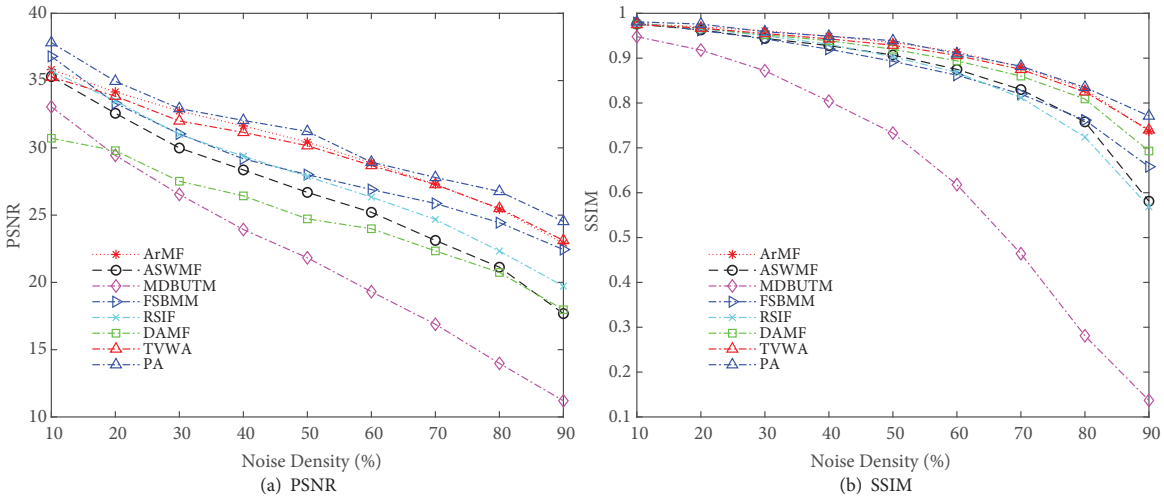


Figure 10. PSNR and SSIM plots for the medical image Thigh, each of size 256×256.

outperforms the other existing filters with an average improvement of around 1 dB in PSNR and 0.1 in SSIM. Moreover, average standard deviations of the proposed algorithm on 24 images are also provided in Table 4, which shows the filters' consistent performances on multiple images as none of the value is greater than 1 dB.

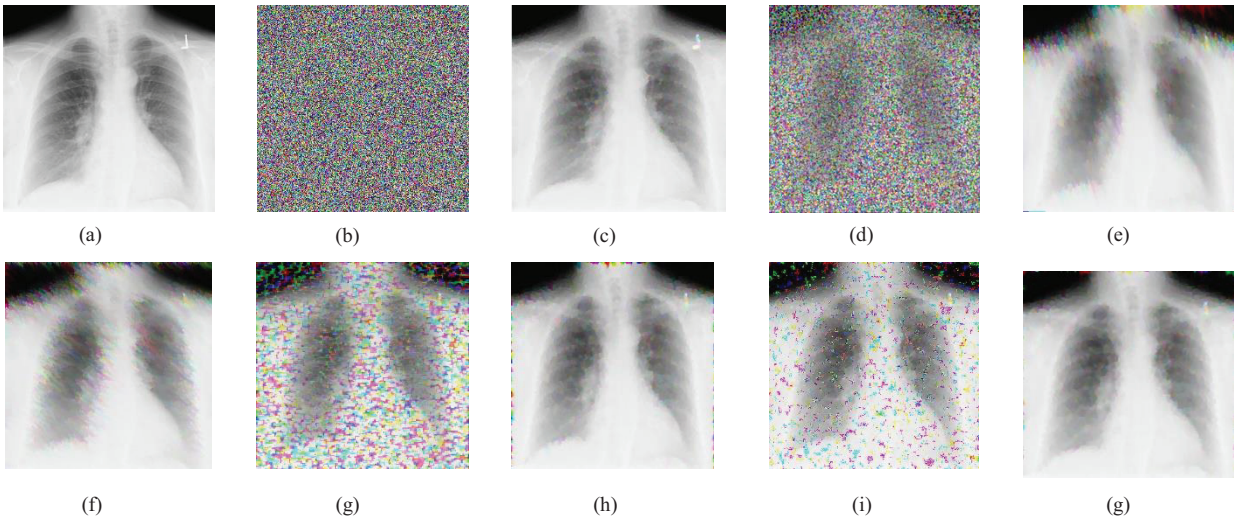


Figure 11. X-ray (Chest) images (a) original, (b) with 95% noise density, and filtered images using: (c) ARmF, (d) MDBUTM, (e) FSBMM, (f) RSIF, (g) TVWA, (h) ASWMF, (i) DAMF, and (j) the proposed filter.

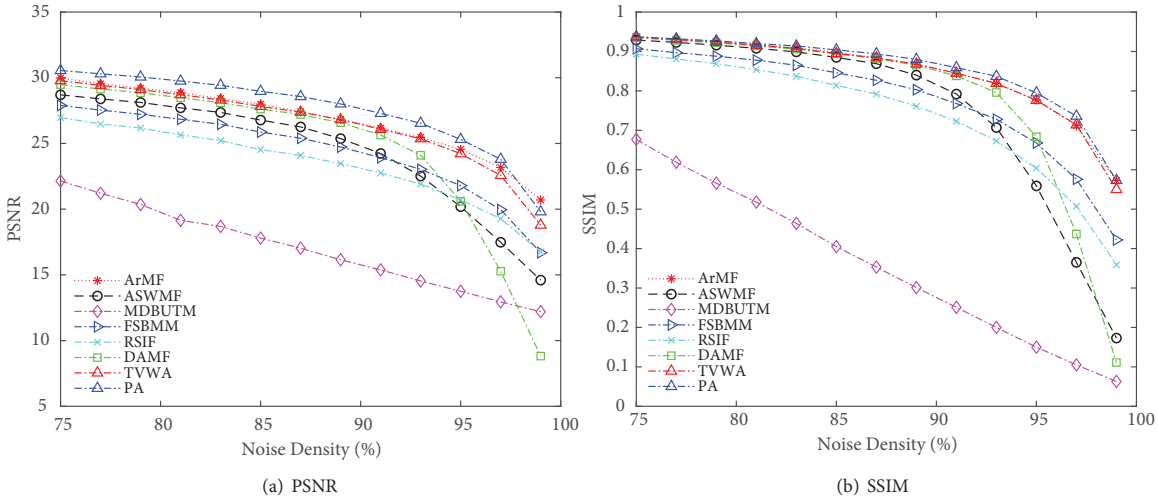


Figure 12. PSNR for different filters with varying noise densities using Lena image (512×512).

4.4. Analysis of the proposed EBLS add-on

Table 5 shows the performance of the filters, namely DAMF [14], ASWMF [4], and TVWA [13] in terms of PSNR. To evaluate these metrics, the Lena (512×512) image is processed with and without the use of universal EBLS add-on at varying noise densities. Variation in noise density is dependent on the threshold value for respective filters. Threshold is defined as the value beyond which the extension produces significantly better results for a given filter respectively. Thus, the table suggest that there is improved performance in each filter after some threshold. The table shows, on an average, 14.8%, 2.23%, and 5.25% improvement in PSNR values of DAMF, TVWA, and ASWMF, respectively. Figures 14a–14c shows the plot of these filters before and after the application of the proposed EBLS add-on. From the plot, we can infer that each filter after some value of threshold noise density gives significant increase in the performance in terms of PSNR.

Table 3. Average PSNR and SSIM of filtered Lena, Zelda, and Boat images using the proposed and existing filters.

Metric	Noise (%)	ARmF	ASWMF	MDBUTM	FSBMM	RSIF	DAMF	TVWA	PA
PSNR	75	29.98	28.70	22.15	27.89	26.97	29.48	29.78	30.55
	77	29.54	28.39	21.21	27.54	26.48	29.14	29.42	30.30
	79	29.21	28.12	20.36	27.22	26.17	28.86	29.10	30.06
	81	28.87	27.71	19.15	26.84	25.65	28.48	28.71	29.75
	83	28.42	27.36	18.69	26.46	25.22	28.10	28.31	29.43
	85	27.99	26.78	17.80	25.87	24.53	27.62	27.82	28.97
	87	27.42	26.26	17.03	25.40	24.09	27.20	27.38	28.57
	89	26.84	25.37	16.15	24.73	23.47	26.58	26.81	28.01
	91	26.17	24.24	15.38	23.92	22.75	25.64	26.08	27.29
	93	25.46	22.52	14.54	23.03	21.91	24.10	25.36	26.54
	95	24.55	20.19	13.75	21.79	20.71	20.60	24.22	25.31
	97	23.18	17.48	12.95	19.96	19.28	15.28	22.57	23.80
	99	20.70	14.60	12.20	16.69	16.74	8.82	18.79	19.80
SSIM	75	0.938	0.929	0.677	0.907	0.893	0.935	0.937	0.937
	77	0.931	0.923	0.619	0.897	0.881	0.928	0.930	0.932
	79	0.925	0.916	0.566	0.888	0.869	0.922	0.924	0.927
	81	0.917	0.908	0.518	0.878	0.854	0.914	0.916	0.920
	83	0.909	0.899	0.464	0.865	0.837	0.906	0.908	0.914
	85	0.898	0.885	0.405	0.846	0.814	0.894	0.895	0.904
	87	0.883	0.869	0.353	0.827	0.792	0.882	0.884	0.894
	89	0.867	0.840	0.301	0.803	0.761	0.865	0.868	0.880
	91	0.844	0.792	0.251	0.768	0.723	0.838	0.845	0.859
	93	0.819	0.707	0.200	0.728	0.673	0.796	0.819	0.837
	95	0.777	0.559	0.150	0.668	0.604	0.684	0.776	0.795
	97	0.712	0.365	0.105	0.576	0.507	0.437	0.714	0.736
	99	0.573	0.173	0.063	0.422	0.359	0.111	0.550	0.573

Figure 13. Lena images (a) original, (b) with 91% noise density, and filtered images using: (c) ARmF, (d) MDBUTM, (e) FSBMM, (f) RSIF, (g) TVWA, (h) ASWMF, (i) DAMF, and (j) the proposed median filters.

Table 4. Average PSNR and SSIM of filtered Kodak-24 bench mark images using proposed and existing filters.

Metric	Noise (%)	ARmF	ASWMF	MDBUTM	FSBMM	RSIF	DAMF	TVWA	PA
PSNR	75	25.38	20.55	24.21	23.67	24.76	25.39	23.12	25.84
	77	25.12	19.81	23.97	23.41	24.49	25.08	22.81	25.60
	79	24.82	19.10	23.76	23.21	24.25	24.82	22.53	25.44
	81	24.54	18.34	23.51	22.94	23.96	24.52	22.21	25.21
	83	24.18	17.60	23.21	22.63	23.67	24.10	21.88	24.88
	85	23.87	16.84	22.98	22.35	23.36	23.83	21.51	24.68
	87	23.46	16.10	22.65	22.07	23.01	23.46	21.13	24.35
	89	23.10	15.34	22.26	21.72	22.63	23.04	20.65	24.04
	91	22.61	14.58	21.83	21.31	22.03	22.56	20.16	23.63
	93	22.11	13.84	21.33	20.77	21.00	21.92	19.51	23.10
	95	21.43	13.10	20.51	20.08	18.75	21.25	18.74	22.51
	97	20.64	12.36	19.51	19.17	14.40	20.25	17.59	21.52
	99	19.07	12.73	16.85	16.77	8.64	17.01	15.02	17.89
SSIM	75	0.891	0.604	0.851	0.840	0.883	0.891	0.824	0.881
	77	0.882	0.554	0.840	0.828	0.875	0.882	0.810	0.875
	79	0.872	0.505	0.828	0.816	0.865	0.871	0.795	0.867
	81	0.860	0.453	0.814	0.800	0.853	0.859	0.776	0.858
	83	0.847	0.403	0.798	0.784	0.841	0.847	0.757	0.847
	85	0.832	0.353	0.779	0.766	0.826	0.831	0.734	0.835
	87	0.813	0.305	0.758	0.743	0.808	0.814	0.707	0.820
	89	0.793	0.259	0.731	0.717	0.786	0.791	0.675	0.803
	91	0.767	0.214	0.700	0.686	0.756	0.766	0.637	0.781
	93	0.734	0.173	0.661	0.646	0.707	0.733	0.589	0.753
	95	0.691	0.133	0.611	0.597	0.596	0.690	0.530	0.716
	97	0.630	0.094	0.543	0.529	0.362	0.630	0.442	0.658
	99	0.522	0.068	0.252	0.220	0.087	0.312	0.192	0.325

Table 5. Average PSNR of three existing filters before and after application of add-On.

Noise density	DAMF		TVWA		ASWMF	
	W/o EBSL	With EBSL	W/o EBSL	With EBSL	W/o EBSL	With EBSL
50%	NULL	NULL	NULL	NULL	32.31	32.35
55%	NULL	NULL	NULL	NULL	31.58	32.02
60%	31.73	31.96	NULL	NULL	30.81	31.66
65%	30.99	31.62	31.42	31.55	29.97	31.20
70%	30.30	31.19	30.53	31.05	29.18	30.65
75%	29.38	30.50	29.76	30.50	28.42	30.10
80%	28.43	29.76	28.70	29.65	27.73	29.56
85%	27.31	28.69	27.75	28.69	26.48	28.42
90%	25.69	27.10	26.52	27.37	24.47	26.43
95%	20.56	24.00	24.59	25.12	20.12	22.00
97%	15.34	20.79	23.10	23.56	17.61	19.00
99%	9.01	15.44	20.26	20.56	14.76	15.44

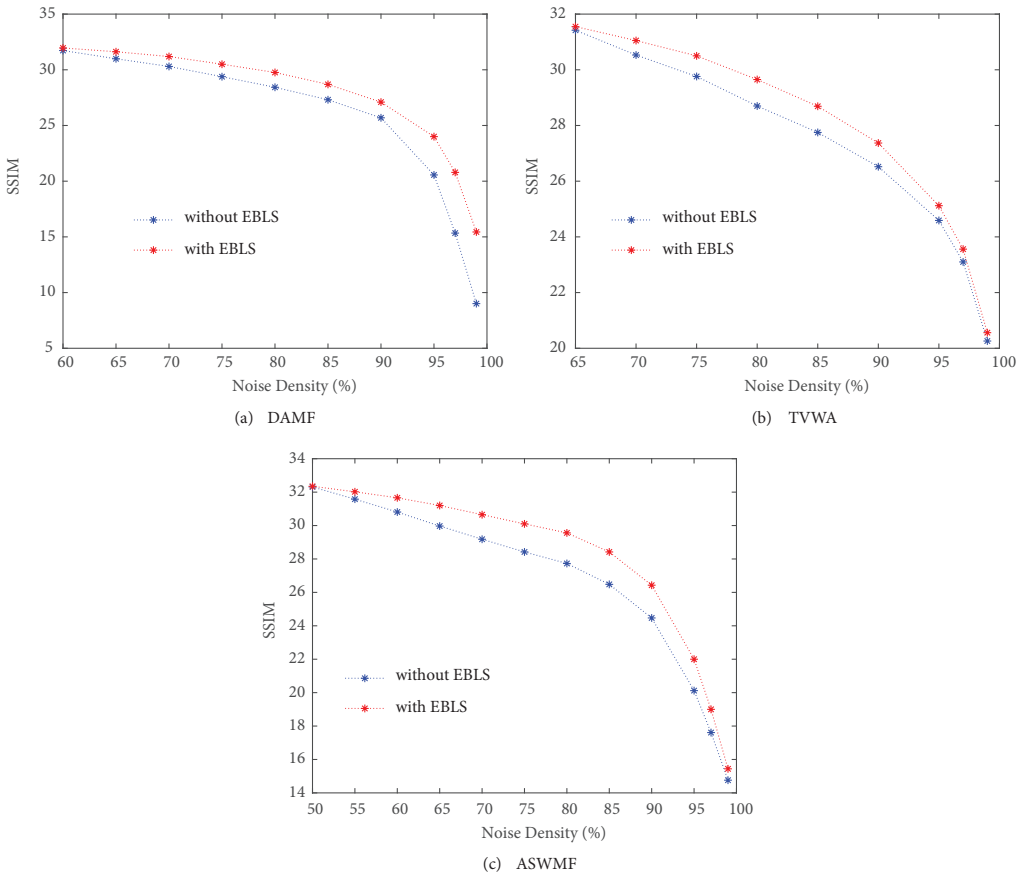


Figure 14. PSNR plots of different filters before and after the application of EBLS add-on.

5. Conclusion

In this paper, a new hybrid decision-based filter is proposed which uses variable size processing windows to remove noise at different noise densities. Furthermore, a universal edge-based logical smoothener add-on is also proposed that can be utilized with the proposed or any other existing salt and pepper noise removal filter to improve the performance. The proposed filter can preserve the edges of the image even at 97% and 99% noise densities and hence has better PSNR and SSIM. The simulation results show that the proposed filter provides stable high performance on grey-scale, colored, and medical benchmark images. Finally, it is observed that the proposed add-on improves the performance of all the existing filters by improving their results to a good extent. Moreover, the application of this filter in medical images could be used as an asset.

References

[1] Gonzalez RC. Digital image processing. 2002 2nd ed. Hoboken, NJ, USA: Prentice Hall.

[2] Wang Z, Zhang D. Progressive switching median filter for the removal of impulse noise from highly corrupted images. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 1999: 46 (1): 78-80.

[3] Yung NHC, Lai AHS, Poon KM. Modified CPI filter algorithm for removing salt-and-pepper noise in digital images. In: Visual Communications and Image Processing. 1996: 2727: 1439-1449.

[4] Faragallah OS, Ibrahem HM. Adaptive switching weighted median filter framework for suppressing salt-and-pepper noise. AEU-International Journal of Electronics and Communications. 2016: 70 (8): 1034-1040.

[5] Varatharajan R, Vasanth K, Gunasekaran M, Priyan M, Gao XZ. An adaptive decision based kriging interpolation algorithm for the removal of high density salt and pepper noise in images. Computers & Electrical Engineering. 2018: 70: 447-461.

[6] Kalyoncu C, Toygar O, Demirel H. Interpolation-based impulse noise removal. IET Image Processing. 2013: 7: 777-785.

[7] Vijaykumar V, Mari GS, Ebenezer D. Fast switching based median–mean filter for high density salt and pepper noise removal. AEU-International journal of electronics and communications. 2014: 68 (12): 1145-1155.

[8] Esakkirajan S, Veerakumar T, Subramanyam AN, PremChand C. Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter. IEEE Signal processing letters. 2011: 18 (5): 287-290.

[9] Li Z, Liu G, Xu Y, Cheng Y. Modified directional weighted filter for removal of salt & pepper noise. Pattern Recognition Letters 2014: 40: 113-120.

[10] Veerakumar T, Esakkirajan S, Vennila I. Recursive cubic spline interpolation filter approach for the removal of high density salt-and-pepper noise. Signal, Image and Video Processing. 2014: 8 (1): 159-168.

[11] Srinivasan K, Ebenezer D. A new fast and efficient decision-based algorithm for removal of high-density impulse noises. IEEE Signal Processing Letters 2007: 14 (3): 189-192.

[12] Balasubramanian S, Kalishwaran S, Muthuraj R, Ebenezer D, Jayaraj V. An efficient non-linear cascade filtering algorithm for removal of high density salt and pepper noise in image and video sequence. In: IEEE International Conference on Control, Automation, Communication and Energy Conservation, Perundurai, India. 2009: 1-6.

[13] Lu CT, Chen YY, Wang LL, Chang CF. Removal of salt-and-pepper noise in corrupted image using three-values-weighted approach with variable-size window. Pattern Recognition Letters 2016: 80: 188-199.

[14] Erkan U, Gökrem L, Enginoğlu S. Different applied median filter in salt and pepper noise. Computers & Electrical Engineering. 2018: 70: 789-798.

[15] Qian RJ, Huang TS. Optimal edge detection in two-dimensional images. IEEE Transactions on Image Processing 1996: 5 (7): 1215-1220.

[16] Lu CT, Chen MY, Shen JH, Wang LL, Hsu CC. Removal of salt-and-pepper noise for x-ray bio-images using pixel-variation gain factors. Computers & Electrical Engineering. 2018: 71: 862-876.

[17] Sternberg SR, Herteg G, Koskella MP, Berla TS. Apparatus and method for implementing transformations in digital image processing. U.S. Patent 4,665,551, issued May 12, 1987.

[18] Hidalgo MG, Massanet S, Mir A, Aguilera DR. Improving salt and pepper noise removal using a fuzzy mathematical morphology-based filter. Applied Soft Computing 2018: 63: 167-180.

[19] Erkan U, Thanh DNH, Enginoğlu S, Memi S. Improved adaptive weighted mean filter for salt-and-pepper noise removal. In: International Conference on Electrical and Computer and Communication Engineering (ICECCE), İstanbul, Turkey, 2020: 1-5.

[20] Sohi PJS, Sharma N, Garg B, Arya KV. Noise density range sensitive mean-median filter for impulse noise removal. In: Innovations in Computational Intelligence and Computer Vision. Advances in Intelligent Systems and Computing, Jaipur, India. 2021: 150-162.

[21] Fu B, Zhao X, Song C. A salt and pepper noise image denoising method based on the generative classification. Multimedia Tools Application 2019: 78: 12043-12053.

[22] Thanh DNH, Prasath VBS, Thanh LT. Total variation l1 fidelity salt-and-pepper denoising with adaptive regularization parameter. In: IEEE 5th NAFOSTED Conference on Information and Computer Science (NICS), Ho Chi Minh City, Vietnam. 2018: 400-405.

[23] Thanh DNH, Hien NN, Kalavathi P, Prasath VBS. Adaptive switching weight mean filter for salt and pepper image denoising. In: International Conference on Computing and Network Communications (CoCoNet'19),Trivandrum, India, 2020: 171: 292-301.

[24] Erkan U, Enginoğlu S, Thanh D, Minh HL. Adaptive frequency median filter for the salt and pepper denoising problem. IET Image Processing. 2020: 14: 1291-1302.

[25] Enginoğlu EU, Erkan U,Memiş S. Pixel similarity-based adaptive riesz mean filter for salt-and-pepper noise removal. Multimedia Tools Application. 2019: 78: 35401-35418.

[26] Erkan U, Thanh DNH, Hieu LM, Engínoğlu S. An iterative mean filter for image denoising. IEEE Access. 2019: 7: 167847-167859.

[27] Thanh DNH, Thanh LT, Prasath S, Erkan U. An improved bpdf filter for high density salt and pepper denoising. In: IEEE International Conference on Computing and Communication Technologies (RIVF), Danang, Vietnam. 2019: 1-5.

[28] Thanh D, Hai N, Prasath S, Minh HL, Tavares J. A two-stage filter for high density salt and pepper denoising. Multimedia Tools and Applications 2020: 79: 21013-21035.

[29] Erkan U, Gökrem L. A new method based on pixel density in salt and pepper noise removal. Turkish Journal of Electrical Engineering & Computer Sciences 2018: 26: 1-7.

[30] Sharma N, Sohi PJS, Garg B. An adaptive weighted min-mid-max value based filter for eliminating high density impulsive noise. Wireless Personal Communication 2021: 1-6. doi: 10.1007/s11277-021-08314-5