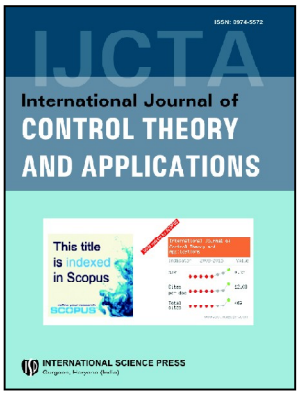

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 29 • 2017

A Review on Median Filters used for the Removal of Salt and Pepper Noise

S. Samsad Beagum[1] and S. Sheeja[2]

[1]Research Scholar, Department of Computer Science, Karpagam University, Karpagam Academy of Higher Education, Coimbatore, India
[2]Associate Professor and Head, Department of Computer Applications, Karpagam University, Karpagam Academy of Higher Education, Coimbatore, India

Abstract: *Background/Objectives:* The aim of this paper is to present a review on the various early and recent median filters available in literature for the removal of salt and pepper noise. *Methods/Analysis:* It presents the working of these filters with examples, merits and demerits. It presents a comparative analysis of the performance of these filters using peak signal to noise ratio PSNR, mean absolute error MAE, Image Enhancement Factor IEF and mean structural similarity index measure MSSIM. *Findings:* Salt and pepper noise affects digital images during their capture using digital cameras and cellular phones and during their transmission. Efficient algorithms are important to remove salt and pepper noise especially in devices like cellular phones that cannot implement hardware mechanism to remove salt and pepper noise. Median filtering is one of the efficient techniques used to remove salt and pepper noise. Still they fail to preserve edges at higher noise ratios. Improving the edge preservation capability of median filters is an important task. This review shows that median filters that use fixed window size execute faster than those that use adaptive window size. However, adaptive median filters produce better restoration results than the median filters using fixed size windows. Some recent adaptive median filters have achieved considerable improvement in preserving image details at very high noise densities giving an average of 0.8 MSSIM index at 90% noise density but at the cost of execution time. *Improvements/Applications:* This review shows that it is crucial to reduce the execution time of adaptive median filters at higher noise densities for their effective implementation in imaging devices.

Keywords: Median filters, Adaptive median filters, Salt and pepper noise, Fixed-valued impulse noise

1. INTRODUCTION

Salt and pepper noise happens due to analog-to-digital converter errors and bit errors in transmission[1, 2]. In digital cameras, pixels become hot due to photodiode leakage currents. When the exposure time is long or when shutter speed is slow, hot pixels appear as salt and pepper noise in captured images. Some cameras use dark frame subtraction to remove salt and pepper noise. A dark frame is an image captured in dark. It is subtracted from subsequent images to remove the salt and pepper noise. However, this mechanism requires a mechanical shutter. Some cameras and cellular phones do not have shutters and hence cannot use this method.

Median filtering and interpolation around the noisy pixels are the other two common methods used for removing salt and pepper noise. Median filters are very effective in removing salt and pepper noise as median is a good estimate of central measure eliminating the outliers but they replace every pixel with the median of the neighborhood and remove the image details. Median filters can be classified into two types – filters with fixed size filtering window and filters with adaptive or variable size filtering window. Adaptive median filter[3] is a very efficient variant of the median filter that preserves edges and image details at lower and medium noise densities but at higher noise densities, it fails to preserve edges and image details. Several variations of the adaptive median filter have been presented to improve the edge preserving capability of median filters[3-9, 16].

In this paper, we review the working and performance of various early and recent variants of the median filter[3, 6-7, 10, 11, 13, 16, 17] used in the removal of salt and pepper noise. The remaining paper is organized as follows. Section 2 provides the literature review of various median filters. Section 3 presents the experimental results and the comparison of the performance of the various filters discussed. Section 4 presents the conclusion.

2. REVIEW OF MEDIAN FILTERS

Let X be the original image of size $M \times N$ and Y denote the image corrupted with salt-and-pepper impulse noise. Let $X(i,j)$ denote the gray level at pixel location (i,j). Let $[S_{min}, S_{max}]$ be the dynamic range of X, i.e. $S_{min} \le X(i,j) \le S_{max}$ for all (i,j). The gray level at any pixel in the corrupted image Y is given by,

$$Y(i,j) = \begin{cases} S_{\min}, \text{with probability } p, \\ S_{\max}, \text{with probability } q, \\ X(i,j), \text{with probability } 1-p-q \end{cases} \quad (1)$$

where $p+q$ defines the noise level.

2.1. Standard Median Filter (SMF)

SMF[5] substitutes all the pixels in the image with the median of their neighborhood pixels. If the neighborhood, for instance, is chosen as 3×3, then for each pixel in the input image, the median filter sorts the pixels in its 3×3 neighborhood and replaces the pixel being processed with the middle value. Consider the following 3×3 neighborhood in Figure 1.

105	167	104
209	56	25
54	100	200

Figure 1: 3×3 Neighborhood

The median filter processes the center pixel with value 56 as follows.

Step 1: The 3×3 matrix is arranged in ascending order.

25, 54, 56, 100, 104, 105, 167, 200, 209

Step 2: The center pixel is replaced with the middle value 104 of the sorted array.

105	167	104
209	104	25
54	100	200

Figure 2: Resultant 3×3 kernel using SMF

The SMF is given by the following equation.

$$SMF\ (i, j) = \text{median}\ (\{Y\ (i - s)\ (j - t) \mid (s, t) \in Sw\})$$

where *Sw* is the filtering window. As every pixel is replaced by the median, SMF removes image details from the processed image.

2.2. Center Weighted Median Filter (CWMF)

A weighted median filter gives a weight to every pixel in the filtering window and finds the median. Center Weighted Median Filter[6] is a variation of the weighted median filter that uses fixed window size. It substitutes all the pixels in the image with the median of their neighborhood pixels after giving more weight to the center pixel. For the 3×3 neighborhood shown in Figure 1, if the weight given to the center pixel ω = 5, then the CWM processes the center pixel with value 56 as follows.

Step 1: The 3×3 matrix is arranged in ascending order.

25, 54, 56, 100, 104, 105, 167, 200, 209

Step 2: The center pixel with value 56 is duplicated w = 5 times.

25, 54, 56, 56, 56, 56, 56, 100, 104, 105, 167, 200, 209

Step 3: The center pixel is replaced with the middle value 56 of the array from step 2.

105	167	104
209	56	25
54	100	200

Figure 3: Resultant 3×3 kernel using CWMF

The CWM is summarized in the following equation.

$$C_{\omega}(i,j) = \text{median}(\{Y(i-s)(j-t), \omega \lozenge Y(i,j) \mid (s,t) \in Sw, (s,t) \neq (0,0)\} \quad (3)$$

In the above equation, *Sw* denotes the filtering window, w is the center weight and $\omega = 2k + 1$ where *k* is a non-negative integer. The operator ◊ denotes the repetition operation. $\omega \lozenge Y(i,j)$ produces ω copies of $Y(i,j)$. As CWM gives more weight to the center pixel, it produces better results than SMF but still it has the same drawback of removing the image details from the output image.

2.3. Adaptive Center Weighted Median Filter (ACWMF)

ACWMF[7] is a variation of the Center Weighted Median Filter. It does not replace all the pixels. It replaces only the noisy pixels in the input image with the median of the neighborhood. It detects the noisy pixels using center weighted median. It determines a pixel as noisy by comparing the differences between the pixel and the center weighted medians with certain thresholds. For the 3×3 Neighborhood shown in Figure 1, the impulse detection procedure is explained as follows.

Step 1: Calculate the center weighted medians $C_1, C_3, C_5, .., C_{max_\omega}$ with center weights $\omega = 1, 3, 5, .., max_\omega$ respectively using equation (2). If the size of the filtering window is given by

$$W \times W = 2L+1, \quad (4)$$

$$\text{then } max_\omega = 2L - 1. \quad (5)$$

For the 3×3 matrix given in Figure 1, $W \times W = 2L+1 = 3 \times 3 = 9$ and hence $L = 4$ and $max_\omega = 7$. The center weighted medians C_1, C_3, C_5 and C_7 are calculated using equation (3).

C_1 = median(25, 54, 56, 100, 104, 105, 167, 200, 209) = 104.

C_3 = median(25, 54, 56, 56, 56, 100, 104, 105, 167, 200, 209) = 100.

C_5 = median(25, 54, 56, 56, 56, 56, 56, 100, 104, 105, 167, 200, 209) = 56.

C_7 = median(25, 54, 56, 56, 56, 56, 56, 56, 56, 100, 104, 105, 167, 200, 209) = 56.

Step 2: Calculate the absolute differences between the center pixel *Y(i, j)* and the center weighted medians from step 1.

$$d_1 = |C_1 - Y(i,j)| = 104 - 56 = 48$$

$$d_3 = |C_3 - Y(i,j)| = 100 - 56 = 44$$

$$d_5 = |C_5 - Y(i,j)| = 56 - 56 = 0$$

$$d_7 = |C_7 - Y(i,j)| = 56 - 56 = 0$$

Step 3: Calculate the absolute differences *dev* between the pixels in the filtering window and the median of the filtering window C_1. This gives the deviation of the pixels from the median of the window.

$$dev = \{|105 - 56|, |167 - 56|, |104 - 56|, |104 - 56|, |209 - 56|, |56 - 56|, |25 - 56|, |54 - 56|, |100 - 56|, |200 - 56|\}$$

$$dev = \{49, 111, 48, 153, 0, 31, 2, 44, 144\}$$

Step 4: Find the median *MAD* of *dev*. This is an estimate of the distribution of the pixels in the filtering window.

$$MAD = median(dev) = 48 \quad (6)$$

Step 5: Calculate a set of thresholds using the following equation.

$$T_i = t_{const} \times MAD + \delta_i \tag{7}$$

where $i = 1, 3, 5, .., max_\omega$ (eqn. 4), $0 \le t_{const} \le 0.6$ and $[\delta_1, \delta_3, \delta_5, \delta_7] = [55, 40, 25, 15]$ for a 3×3 filtering window. The authors[7] have given these constant values after simulation with various images.

In our example, if we take $t_{const} = 0.1$, then

$$T_1 = 0.1 \times 48 + 55 = 59.8 \tag{8}$$

Similarly, $T_3 = 44.8$, $T_5 = 29.8$, $T_7 = 19.8$

Step 6: The pixel being processed is detected as noisy if any difference from step 2 is greater than the corresponding threshold value i.e., $Y(i, j)$ is noisy, if for any $i = 1, 3, 5, .., \max_\omega$, $d_i > T_i$.

In our example, $Y(i,j)$ is not noisy, as $d_1 < T_1$, $d_3 < T_3$, $d_5 < T_5$ and $d_7 < T_7$

The pixel detected as noisy is then replaced with the median of the filtering window.

The ACWM filter is summarized in the following equation.

$$Z_{i,j} = \begin{cases} C_1, & if\ \exists i, d_i > T_i \\ Y(i, j), & otherwise \end{cases} \tag{9}$$

where C_1 is the median of the filtering window.

ACWM works well for smaller noise densities. It is efficient in preserving edges but noise suppression reduces gradually from noise densities greater than 35%. The problem with this algorithm is the selection of the threshold values used in noise detection.

2.4. Adaptive median Filter (AMF)

AMF[3] is a variation of the median filter designed to overcome the drawback of the standard median filter. It replaces only the noisy pixels in the image with the median of the filtering window. For every pixel in the input image Y, first it analyses if the pixel being processed is diverse from most of the neighborhood pixels by comparing it with the minimum and maximum values of the filtering window. If the pixel value is inside the range of minimum and maximum values, it considers the pixel as noise free and leaves unchanged. If the comparison fails, then the pixel is considered as noisy. Second for every noisy pixel, AMF analyses if the median value of the filtering window is noise free by comparing the median with the minimum and maximum values of the filtering window. If the median value is inside the range of minimum and maximum values, it is considered as noise-free and the noisy pixel is substituted with the median of the filtering window. If the median value is corrupted then AMF increases the size of the filtering window and analyses if the median value of the new window is noise free. The increase in the window size is repeated until it finds a noise-free median value or until the maximum allowed window size is reached.

If S_w denotes the filtering window of size $W \times W$ centered at pixel $Y(i,j)$ and $W_{max} \times W_{max}$ denotes the maximum window size used by the algorithm, then the steps involved in AMF is as follows. For each pixel in the input image Y,

Step 1: Initialize $W \times W = 3 \times 3$.

Step 2: Calculate the minimum S_{min}, median S_{med} and maximum S_{max} values of the $W \times W$ filtering window.

Step 3: If $S_{min} < S_{med} < S_{max}$, then the median S_{med} is noise-free and go to step 5; otherwise S_{med} is noisy and go to the next step.

Step 4: Increase the window size by 2, $W = W+2$. If $W \leq W_{max}$ repeat the process from step 2; otherwise replace $Y(i,j)$ with S_{med} and go to process the next pixel.

Step 5: If $S_{min} < Y(i,j) < S_{max}$, then $Y(i,j)$ is noise-free and hence leave it unchanged and go to process the next pixel; otherwise $Y(i,j)$ is a noisy pixel and replace it with S_{med} and process the next pixel.

In our example in Figure 1, $Y(i,j) = 56$, $S_{min} = 25$, $S_{max} = 209$, and $S_{med} = 104$.

From step 3, $S_{min} < S_{med} < S_{max}$, $25 < 104 < 209$, the median of the filtering window S_{med} is noise-free and hence step 5 is processed. Otherwise, the window size is increased and step 3 is again performed until S_{med} is noise-free or the window size reaches the maximum.

From step 5, $S_{min} < Y(i,j) < S_{max}$, $25 < 56 < 209$, the center pixel $Y(i,j) = 56$ is noise-free and left unchanged.

AMF works well in terms of noise suppression and edge preservation for low to medium noise densities. For higher noise densities, it suppresses noise but fails to preserve image details and edges. In addition, as it uses adaptive window size, it consumes more time than median filters with fixed window size.

2.5. Progressive Switching Median Filter (PSMF)

PSMF[10] works in two stages – the impulse detection stage and the filtering stage. The impulse detection stage identifies the noisy pixels and the filtering stage replaces only those pixels identified as noisy. This two-stage process is referred to as a switching scheme. Both the stages are iterative procedures using a fixed window size.

2.5.1. Impulse Detection Stage

Let the size of the detection window be D. Let Y denote the noisy image of size $M'N$ and let $Y_k(i, j)$ be the resultant matrix after k^{th} iteration. Let B denote the noise matrix of size $M \times N$ whose values are all zeroes initially. After each iteration k of the impulse detection stage,

$$B(i, j) = \begin{cases} 1, if\ \ Y_{k-1}(i, j)\, is\, noisy \\ 0, otherwise \end{cases} \tag{13}$$

For each pixel $Y_{k-1}(i, j)$ in the noisy image Y_{k-1}, the iteration is as follows.

Step 1: Find the median *med* of the detection window.

Step 2: If $|Y_{k-1}(i, j) - med| \geq T_D$ *and* $B(i, j) \neq 1$, then the pixel is noisy and hence set $B(i, j) = 1$ and $Y_k(i, j) = med$; otherwise $Y_k(i, j) = Y_{k-1}(i, j)$. T_D. is a pre-defined threshold value and is defined by the authors[10] as

$$TD = 65 + (-50) * \text{noiselevel} \tag{14}$$

Step 3: Process the next pixel.

The impulse detection process is iterated 3 times. The variable *noiselevel* is calculated from the noisy image before the restoration process.

Calculation of noise-level: Let *NCount* denote the number of noisy pixels in the noisy image whose initial value is 0. For each pixel $Y(i, j)$ in the input noisy image Y, calculate the median of its 3×3 neighborhood window as *med*. If $|\text{med} - Y(i, j)| < 40$, increment *NCount*. After processing all the pixels, *noiselevel* is calculated as

$$noiselevel = \frac{N\,Count}{M \times N} \tag{15}$$

The size of the detection window D is set to the value 3, if *noiselevel* < 0.25; otherwise it is set to 5.

For example, if *noiselevel* = 0.20 then $T_D = 65 + (-50) * 0.20 = 55$.

If Figure 6 (b) gives the 3×3 impulse detection window of matrix B for a 3×3 neighborhood in the noisy image Y_{k-1} after the k^{th} iteration as shown in Figure 6 (a), then

105	167	104
56	209	5
54	100	200

(a)

0	0	0
0	0	1
0	0	0

(b)

Figure 6: (a) 3×3 filtering window in noisy image Y_{k-1}. (b) Corresponding values in matrix B

$Y_{k-1}(i, j) = 209$ and $med = 104$. Step 2 of the impulse detection stage checks if $|Y_{k-1}(i, j) - \text{med}| \geq T_D$. $|Y_{k-1}(i, j) - \text{med}| = 209 - 104 = 105$ and $105 \geq 55$. Hence $Y_{k-1}(i, j)$ is an impulse and $Y_k(i, j)$ is set to 104 and $B(i,j)$ is set to 1. The output is shown in Figure 7.

105	167	104
56	104	5
54	100	200

(a)

0	0	0
0	1	1
0	0	0

(b)

Figure 7: (a) Resultant 3×3 matrix in the noisy image. (b) Corresponding values in matrix B

2.5.2. Filtering Stage

Let $Z = Y_{k=3}$ be the noisy image given as input to the filtering stage. This stage uses a fixed filtering window of size 3×3. For each pixel $Z(i, j)$ in the input image Z, if $B(i, j) = 1$, then

Step 1: Find the number of pixels *NGood* in the 3×3 neighborhood, whose corresponding value in the B matrix is 0.

Step 2: If *NGood* = *0*, then process the next pixel; otherwise find the median *medGood* of the good pixels in the 3×3 filtering window whose corresponding value in the B matrix is 0. Replace with *medGood* and set

After processing all the pixels, the procedure is repeated until all the values in B matrix become 0. For the example shown in Figure 7, *NGood* = 7 , *medGood* = median(54, 56, 100, 104, 105, 167, 200) = 104. The output $Z(i, j) = medGood = 104$ and $B(i, j) = 0$.

PSMF consumes more time compared to many algorithms and also removes the image details considerably at medium to higher noise densities. The pre-threshold determination is a disadvantage of this algorithm requiring additional processing to determine the *noiselevel* from the noisy image.

2.6. Decision Based Algorithm (DBA)

DBA[11] uses a fixed window size of 3×3 for restoration. If a pixel is noisy, it replaces it with the median of the filtering window if the median is noise-free. If the median is noisy, it replaces the noisy pixel with the previously processed noise-free pixel. For an input image whose pixel values range from 0 to 255, let $S_{min} = 0$ and $S_{max} = 255$. For each pixel $Y(i, j)$, in the input noisy image Y, DBA performs the following.

Step 1: If $S_{min} < Y(i, j) < S_{max}$, then it is noise-free and hence process the next pixel; otherwise it is a noisy pixel and go to step 2.

Step 2: Find the median *med* of the filtering window.

Step 3: If $S_{min} < med < S_{max}$, then replace $Y(i, j)$ with med; otherwise replace $Y(i, j)$ with the previously processed pixel. Go to process the next pixel.

For the 3×3 filtering window shown in Figure 8, $Y(i, j) = 255 = S_{max}$ and hence a noisy pixel. The median of the filtering window $med = 255 = S_{max}$ is also noisy. Hence $Y(i, j)$ is replaced with the previous noise-free pixel value 56.

105	255	255
56	255	255
54	209	255

Figure 8: 3×3 filtering window

The author of DBA[11] also proposes a fast way of finding median value of filtering window which reduces its execution time significantly.

1. Sort the rows of the 3×3 filtering window.
2. Sort the columns of the resultant 3×3 filtering window.
3. Sort the right diagonal elements of the 3×3 filtering window. Now the center element of the window is its median.

DBA is a very fast and simple algorithm efficient for very low noise ratios but it produces streaky effect in the restored images as the noise ratio goes high.

2.7. Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF)

MDBUTMF[13] also uses a fixed size filtering window of size 3×3. For each pixel $Y(i, j)$, in the input noisy image Y, if it is noisy i.e., $Y(i, j) == 0$ or $Y(i, j) == 255$ then,

Step 1: If all the neighborhood pixels in the filtering window are either 0 or 255 i.e., if all neighborhood pixels are noisy, then replace $Y(i, j)$ with the mean of the filtering window and go to process the next pixel; otherwise go to step 2.

Step 2: Find the good pixels with values > 0 and < 255 in the 3×3 neighborhood. Find their median and replace $Y(i, j)$ with this median.

For the 3×3 filtering window shown in Figure 8, $Y(i, j) = 255$ is a noisy pixel and it is replaced with the median of the good pixels in the neighborhood. After processing $Y(i, j)$ = median (54, 56, 105, 209) $= \frac{(56+105)}{2}$ = 81. If all the pixels in the 3×3 filtering window are noisy as shown in Figure 9, then = mean(255, 255, 255, 255, 255, 255, 0, 0, 255) = 198.

255	255	255
255	255	255
0	0	255

Figure 9: 3×3 filtering window with all noisy pixels

MDBUTMF is proposed as an improvement to DBA for removing the streaky effect produced by DBA. It is as fast as DBA producing better restoration at lower and medium noise densities. However, at higher noise densities, the restored images have a blurred effect.

2.8. Haidi's Adaptive Median Filter (AMF_Haidi)

AMF_Haidi[16] is a variation of adaptive median filter that uses variable filtering window size. It sets the maximum filtering window size to 39. It works in two stages – the impulse detection stage and restoration stage.

The impulse detection stage detects a pixel as noisy if its value = S_{min} or S_{max}, the minimum and maximum intensity values of an image. The output of this stage is an impulse detection matrix B whose value is 1 if the corresponding pixel in the input image Y is noisy, otherwise 0.

The restoration stage restores only the pixels identified as noisy. For each pixel $Y(i, j)$ identified as noisy, the restoration algorithm is as follows.

Step 1: Set the initial filtering window size as 3×3.

Step 2: Count the number of good pixels *NGood* in the filtering window. Good pixels refer to those pixels whose value in the B matrix is 0.

Step 3: If *NGood* < 8, increase the filtering window size by 2 and repeat step 2; otherwise replace $Y(i, j)$ with the median of the good pixels and go to process the next pixel.

0	78	56	**0**	210
67	105	**255**	**255**	58
255	56	255	**0**	**255**
67	54	209	**255**	180
45	58	98	**255**	99

Figure 10: 5×5 filtering window

For the 3×3 window shown inside Figure 10, $Y(i, j) = 255$ and the number of good pixels $NGood = 4$. Since $NGood < 8$, the window size is increased to 5×5. In the 5×5 filtering window, $NGood = 15$ and hence $Y(i, j)$ = median (45, 54, 56, 56, 58, 58, 67, 67, 78, 98, 99, 105, 180, 209, 210) = 67.

AMF_Haidi produces better edge preservation than MDBUTMF even at higher noise ratios.

2.9. Adaptive Weighted Mean Filter (AWMF)

AWMF[17] is an adaptive filter using variable filtering window size but it uses mean filter to restore a noisy pixel. It sets the maximum filtering window size W_{max} to 39. For each pixel in the input image Y, the algorithm works as follows.

Step 1: Set the initial filtering window size $W \times W$ as 3×3.

Step 2: Find the minimum W_{min} and maximum W_{max} values of the filtering window.

Step 3: Count the number of good pixels $NGood$ in the filtering window. Good pixels refer to those pixels whose value satisfies the following condition.

$$W_{min} < Y(i, j) < W_{max} \tag{16}$$

Step 4: If $NGood > 0$, find the mean of the good pixels as W_{mean}; otherwise set $W_{mean} = -1$.

Step 5: Find the minimum $W1_{min}$ and maximum $W1_{max}$ values of the next filtering window $(W+2)\times(W+2)$ centered at $Y(i, j)$.

Step 6: If $W_{min} = W1_{min}$ and $W_{max} = W1_{max}$ and $NGood > 0$, go to step 8; otherwise increase window size by 2, $W = W + 2$, and go to step 7.

Step 7: If $W \leq W_{max}$, go to step 2; otherwise replace $Y(i, j)$ with W_{mean} and go to process the next pixel.

Step 8: If $W_{min} < Y(i, j) < W_{max}$, $Y(i, j)$, is a noise-free pixel and hence go to process the next pixel; otherwise replace $Y(i, j)$ with W_{mean} and go to process the next pixel.

For the 3×3 window shown inside Figure 10, $Y(i, j) = 255$, $W_{min} = 0$, $W_{max} = 255$ and the number of good pixels ($W_{min} < 54, 56, 105, 209 < W_{max}$) $NGood = 4$. Since $NGood > 0$ the mean of the good pixels is calculated to be W_{mean} = mean(56, 105, 209) = 123. Then from step 5, the minimum and maximum values of 5×5 window are

calculated. $W1_{min} = 0$, $W1_{max} = 255$. Since $W_{min} = W1_{min} = 0$ and $W_{max} = W1_{max} = 255$, and $NGood > 0$, step 8 is done. Otherwise, window size is increased and the process is repeated. In step 8, $Y(i, j) = W_{max} = 255$ and hence it is a noisy pixel and replaced with $W_{mean} = 123$. Step 5 of calculating minimum and maximum values for the next window is done to speed up the process.

This algorithm performs better than all the other algorithms discussed above especially in preserving image details at very high noise ratios but it consumes more time than median filters of fixed-size filtering window.

3. EXPERIMENTAL SETUP AND RESULTS

3.1. Setup

All the algorithms are implemented in MATLAB 64-bit R2015a, installed in a laptop with 2.30GHz Intel Core i5 processor and 6GB RAM. The algorithms are tested with several standard 8-bit gray scale images of size 512×512, with dynamic range [0, 255] including the Lena, Bridge, CameraMan, LivingRoom and Mandril.

The performance of the algorithms is measured quantitatively by Peak Signal-to-Noise Ratio (PSNR) measure, mean absolute error (MAE) measure, Image Enhancement Factor (IEF) and Mean Structural Similarity Index Measure (MSSIM) which are defined as

$$PSNR = 10 \log_{10} \frac{255^2}{\frac{1}{MN} \Sigma_{i,j} (z_{i,j} - X_{i,j})^2} \tag{17}$$

$$MAE = \frac{1}{MN} \Sigma_{i,j} | Z_{i,j} - X_{i,j} | \tag{18}$$

$$IEF = \frac{\Sigma_{i,j} (Y(i, j) - X(i, j))^2}{\Sigma_{i,j} (Z(i, j) - X(i, j))^2} \tag{19}$$

$$MSSIM(X, Z) = \frac{1}{NW} \Sigma_i SSIM(x_j, z_j) \tag{20}$$

where X, Y and Z are the original, noisy and restored images respectively; $M \times N$ is the size of the image; NW is the number of windows used in MSSIM calculation; x_j and z_j are the portions of the original and restored images at window j; and

$$SSIM(x, z) = \frac{(2 \times avg(x) \times avg(z) + C_1)(2 \times \text{cov}(x, z) + C_2)}{(avg(x)^2 + avg(z)^2 + C_1)(\sigma^2(x) + \sigma^2(z) + C_2)} \tag{21}$$

where avg refers to average; σ^2 refers to variance and *cov* refers to covariance; C1 = $(0.01 \times S_{max})^2$ and C2 = $(0.03 \times S_{max})^2$ by default.

The maximum window size used by AMF for different noise levels[3] is shown in table 1. CWMF is implemented with 3×3 filtering window with center weight 3.

Table 1
Maximum Window Size Used in AMF

Noise level	$W_{max} \times W_{max}$
< 25%	5×5
25% to 40%	7×7
41% to 60%	9×9
61% to 70%	13×13
71% to 80%	17×17
81% to 85%	25×25
86% to 90%	39×39

3.2. Results

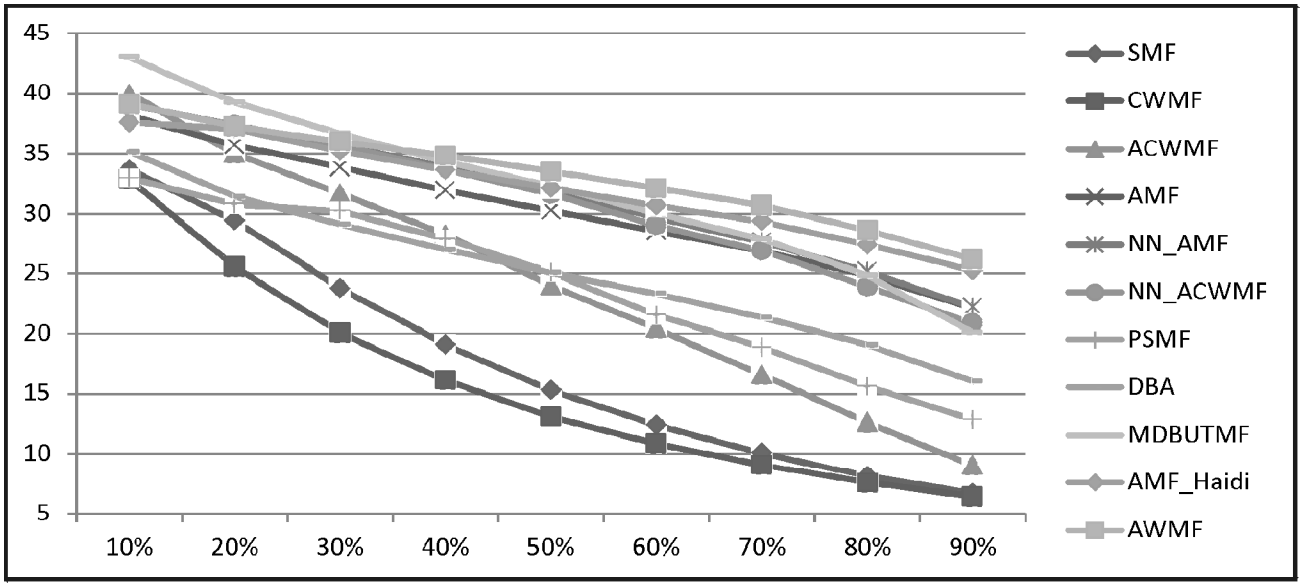

Figure 11: PSNR values given by the filters at 10% to 90% noise densities for Lena image

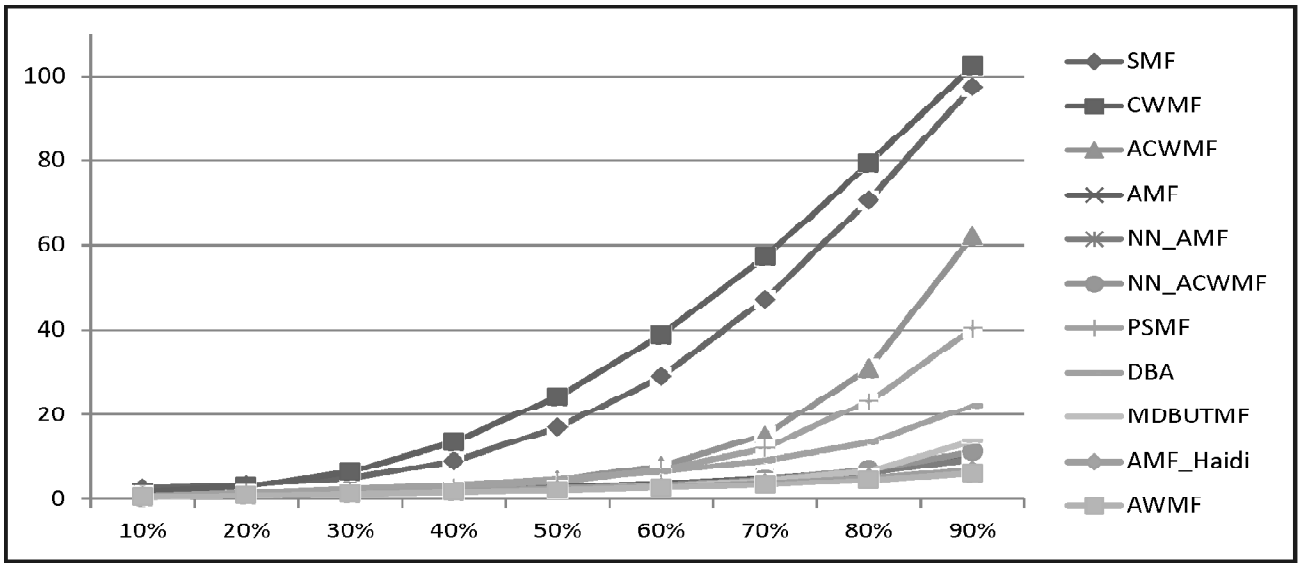

Figure 12: MAE values given by the filters at 10% to 90% noise densities for Lena image

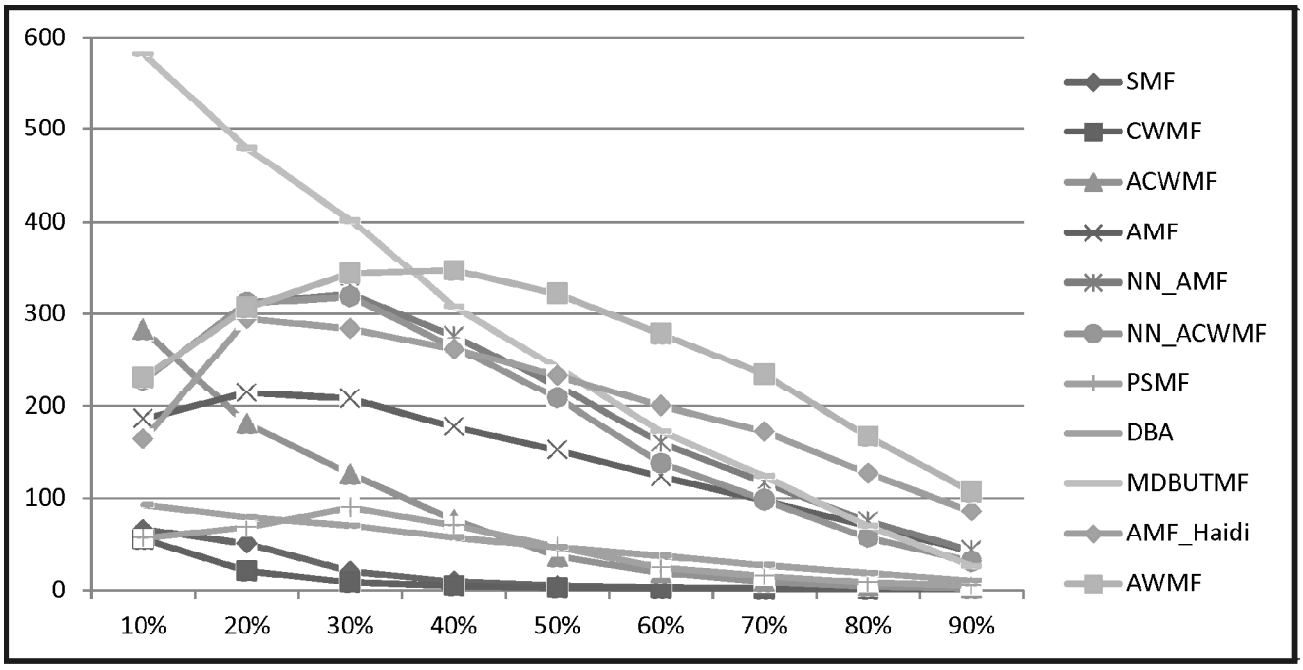

Figure 13: IEF values given by the filters at 10% to 90% noise densities for Lena image

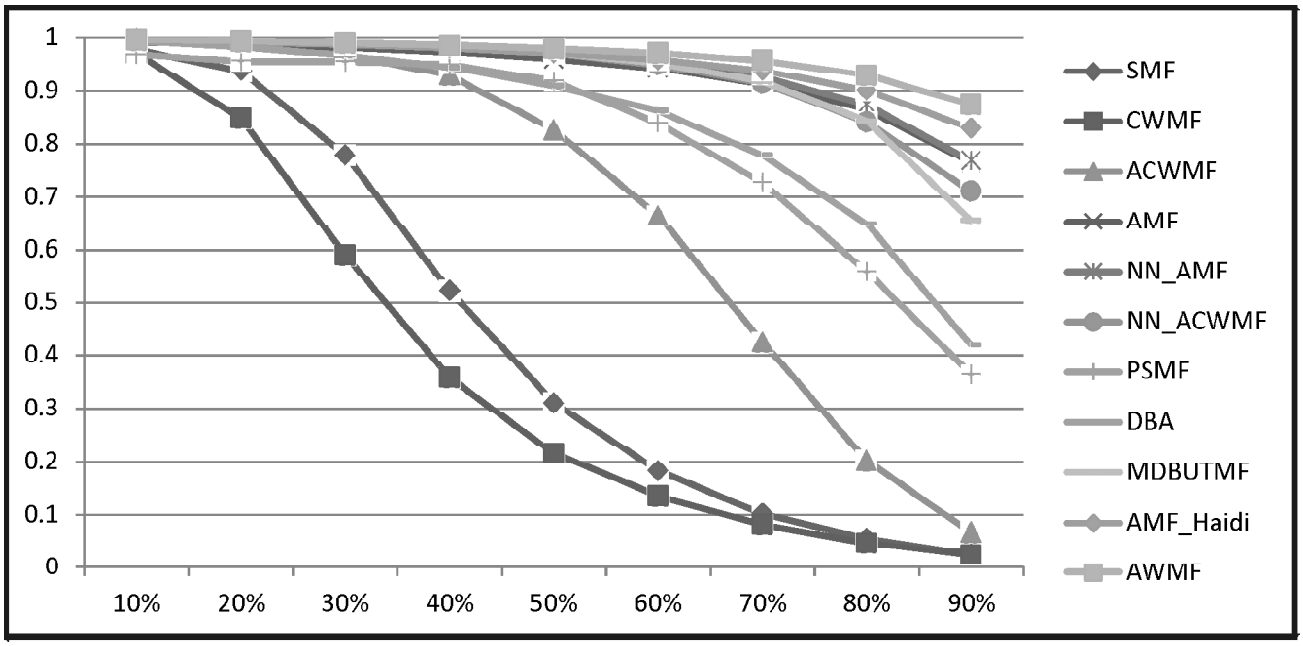

Figure 14: MSSIM values given by the filters at 10% to 90% noise densities for Lena image

The PSNR, MAE, IEF and MSSIM measures given by the various algorithms for the Lena image at various noise densities are shown as graphs in figures 11 to 14 respectively. The graphs of all test images show that all the filters produce good results for lower noise densities. As the noise density increases, the performance of the filters decreases. The 3´3 SMF and 3x3 CWMF with center weight 3 have given the least PSNR, MAE, IEF, and MSSIM measures. The 3´3 filter MDBUTMF gives the best PSNR and IEF measures until 35% whereas above 35% the adaptive filter AWMF produces the best measures. In case of MAE and MSSIM measures, AWMF produces the best results.

Figure 16 shows the CameraMan image restored by the various filters at 80% noise density. The restored test images show that the SMF, CWMF and ACWMF have failed to suppress noise from noise ratios > 30% and the degree of noise suppression decreases with increasing noise ratios.

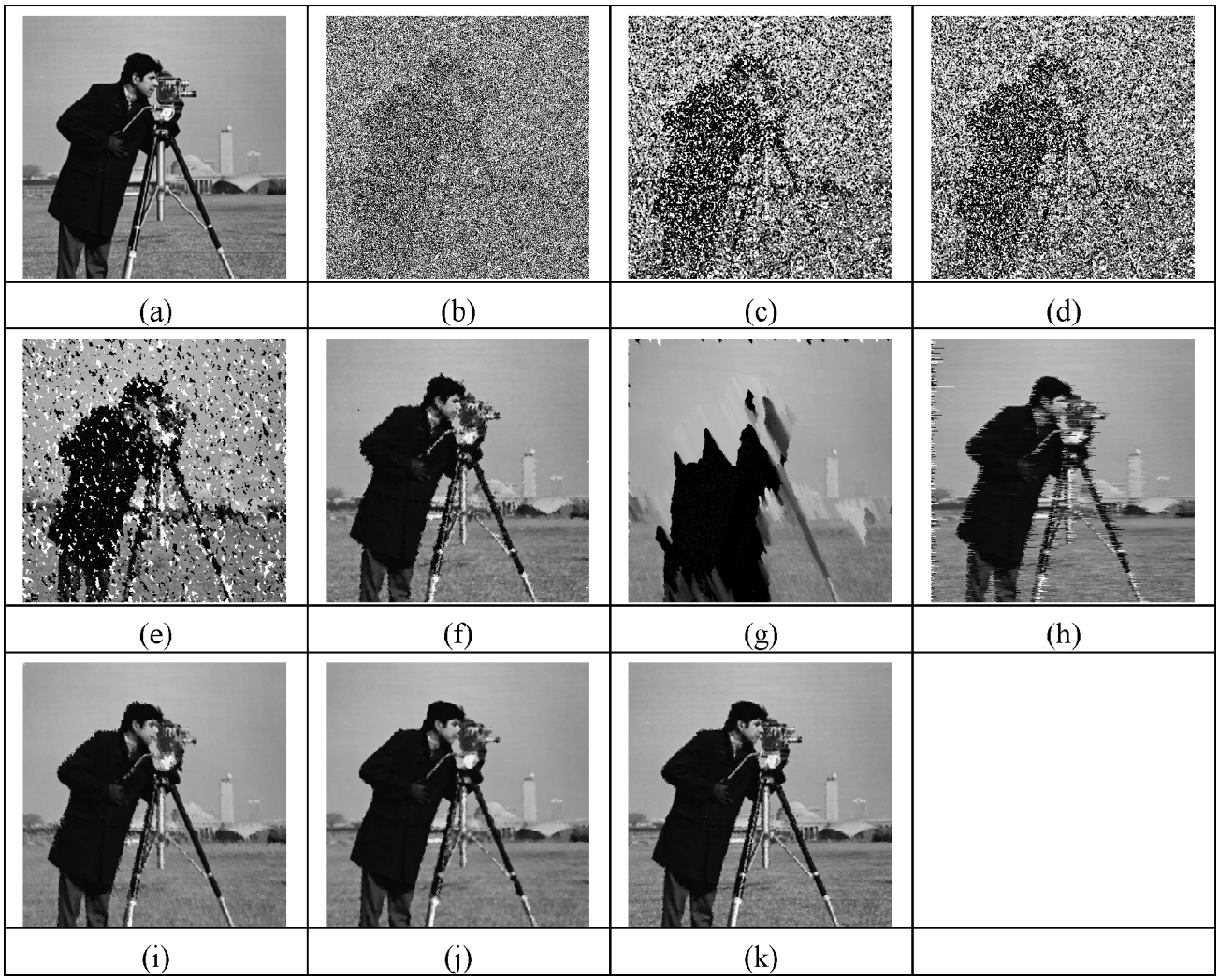

Figure 16: (a) Original CameraMan image (b) Corrupted with 80% salt and pepper noise. Image restored using (c) 3×3 SMF (d) 3×3 CWMF with center weight 3 (e) ACWMF (f) AMF (g) PSMF (h) DBA (i) MDBUTMF (j) AMF_Haidi (k) AWMF

PSMF removes the image details considerably from noise ratios > 45%. DBA produces lines on restored images from noise ratios > 35%. The adaptive filters AMF_Haidi and AWMF have produced better detail preservation than all the other filters even at higher noise ratios and AWMF has given the best quality restoration. Tables 2 and 3 show the PSNR and MSSIM results given by the various filters at 90% noise density for 5 test images. The results show that AWMF achieves the best restoration at a very high noise density of 90%

Figure 17 shows the plot of time taken in seconds by the various filters to restore the Lena image at 90% noise density. The time graphs for all test images show that 3×3 filters namely SMF, CWMF, DBA, and MDBUTMF take considerably less time to execute and the execution time remains approximately the same at all noise ratios. DBA and SMF takes the least execution time compared to all the filters. Among the adaptive filters, AMF_Haidi takes very low execution time than all the other adaptive filters. At noise densities lower than 30%

Table 2
PSNR values given by various filters at 90% noise density for 5 test images

	SMF	*CWMF*	*ACWMF*	*AMF*	*PSMF*	*DBA*	*MDBUTMF*	*AMF_ Haidi*	*AWMF*
Lena	6.68	6.43	9.05	22.14	12.83	16.03	20.08	25.26	26.19
Bridge	6.40	6.19	8.34	18.48	11.98	16.20	17.79	20.87	21.27
CameraMan	6.25	6.04	8.38	20.80	10.67	15.97	19.10	23.15	25.11
LivingRoom	6.74	6.50	8.91	20.35	13.70	17.19	19.59	22.58	23.33
Mandril	6.81	6.58	8.95	19.30	13.05	16.50	19.86	21.27	21.95

Table 3
MSSIM values given by various filters at 90% noise density for 5 test images

	SMF	*CWMF*	*ACWMF*	*AMF*	*PSMF*	*DBA*	*MDBUTMF*	*AMF_ Haidi*	*AWMF*
Lena	0.03	0.02	0.07	0.76	0.36	0.42	0.66	0.83	0.87
Bridge	0.03	0.03	0.06	0.57	0.15	0.42	0.41	0.60	0.73
CameraMan	0.03	0.03	0.05	0.80	0.41	0.60	0.70	0.82	0.90
LivingRoom	0.03	0.03	0.06	0.63	0.27	0.46	0.52	0.68	0.77
Mandril	0.03	0.03	0.06	0.56	0.20	0.35	0.47	0.55	0.72

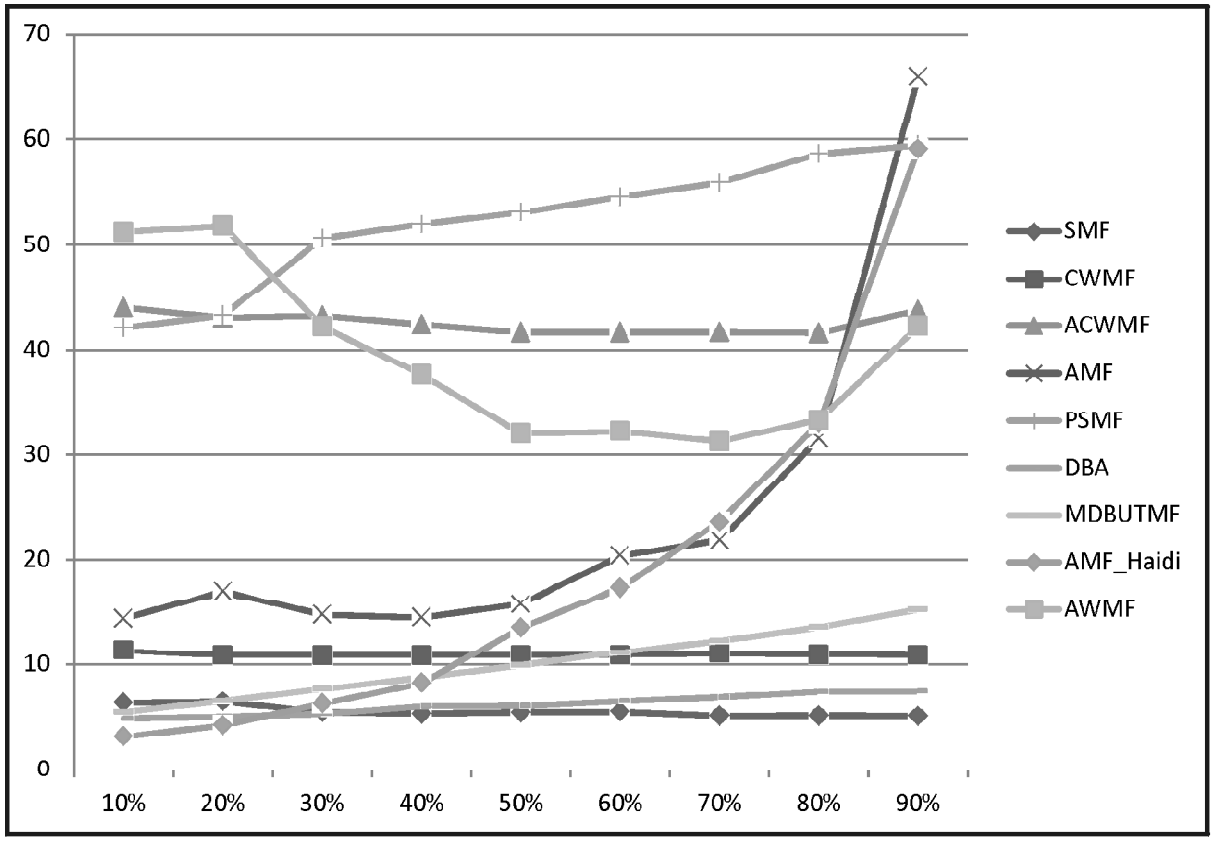

Figure 17: Time in seconds taken by the filters at 10% to 90% noise densities for Lena image

the execution time of AMF_Haidi is lower than the 3´3 filters. AWMF which has given best restoration results takes approximately 40 seconds to execute at all noise densities.

The adaptive median filters particularly AMF_Haidi and AWMF have shown best restoration results with better noise suppression and edge preservation even at very high noise densities whereas 3×3 filters take less execution time when compared to adaptive filters. AMF_Haidi is an exception in that it is an adaptive filter taking lesser execution time than the 3×3 filters at lower noise densities. MDBUTMF is the best filter for lower noise densities until 40% with best restoration results and consuming less execution time whereas AWMF is the best filter for better restoration at higher noise ratios and AMF_Haidi is best for higher noise ratios in terms of lesser execution time and better restoration results than all the other filters except AWMF.

4. CONCLUSION

This paper presented the working of various median filters with fixed and adaptive filtering window size used in the removal of salt and pepper noise. It is shown that 3×3 filters work faster than filters with adaptive window size. However, the adaptive filters give better restoration with detail preservation especially at higher noise densities. The review shows that achieving restoration with detail preservation at reduced execution time is vital in the removal of higher density salt and pepper noise.

REFERENCES

[1] Shapiro LG, Stockman GC. Computer Vision. Prentice-Hall; 2001.

[2] Bovik AC. Handbook of Image and Video Processing. Academic Press; 2005.

[3] Lin HM, Wilson Jr. AN. Median filters with adaptive length. IEEE Transactions on Circuits and Systems. 1988 June; 35: 675-690.

[4] Hwang H, Haddad RA. Adaptive Median Filters: New Algorithms and Results. IEEE Transactions on Image Processing. vol. 4, no. 4, pp. 499-502, Apr 1995.

[5] Rosenfeld A, Kak AC. Digital picture processing. vol. 1, Elsevier, 1982.

[6] Ko S-J, Lee YH. Center Weighted Median Filters and Their Applications to Image Enhancement. IEEE Transactions on Circuits and Systems, vol. 38, no. 9, pp. 984-993, Sep 1991.

[7] Chen T, Wu HR. Adaptive impulse detection using center-weighted median filters. IEEE Signal Processing Letters, vol. 8, pp. 1–3, 2001.

[8] Beagum SS, Hundewale N, Sathik MM. Improved adaptive median filters using nearest 4-neighbors for restoration of images corrupted with fixed-valued impulse noise. IEEE Internationl Conference on Computational Intelligence and Computing Research. 2015.

[9] Sun T, Neuvo Y. Detail-preserving median based filters in image processing. Pattern Recognition Letters. 1994;15:341–347.

[10] 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. 1999;46:78–80.

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

[12] Aiswarya K, Jayaraj V, Ebenezer D. A new and efficient algorithm for the removal of high density salt and pepper noise in images and videos. Second International Conference on Computer Modeling and Simulation. 2010; 409–413.

[13] Esakkirajan S, Veerakumar T, Subramanyam AN, PremChand CH. Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter. IEEE Signal Processing Letters. 2011;18:287–290.

[14] Gupta V, Gandhi DK, Yadav P. Removal of fixed value impulse noise using improved mean Filter for image enhancement. IEEE Nirma University International Conference in Engineering. 2013.

[15] Chaitanya NK, Sreenivasulu P. Removal of salt and pepper noise using advanced modified decision based unsymmetric trimmed median filter. International Conference on Electronics and Communication Systems. 2014.

[16] Ibrahim H, Kong NSP, Ng F. Simple adaptive median filter for the removal of impulse noise from highly corrupted images. IEEE Transactions on Consumer Electronics. 2008;54(4):1920–1927.

[17] Zhang P, Li F. A new adaptive weighted mean filter for removing salt-and-pepper noise. IEEE Signal Processing Letters. 2014; 21 (10): 1280–1283.