LITERATURE SURVEY ON IMPULSE NOISE REDUCTION

Manohar Koli[1] and S.Balaji[2]

[1]Dept. of Computer Science, Tumkur University Tumkur-572103, Karnataka, India
[2]Dept. of CSE, City Engineering College Bangalore-560061, Karnataka, India

ABSTRACT

In every image processing algorithm quality of image plays a very vital role because the output of the algorithm depends on the quality of input image. Hence, several techniques are used for image quality enhancement and image restoration. Some of them are common techniques applied to all the images without having prior knowledge of noise and are called image enhancement algorithms. Some of the image processing algorithms use the prior knowledge of the type of noise present in the image and are referred to as image restoration techniques. Image restoration techniques are also referred to as image de-noising techniques. In such cases, identified inverse degradation functions are used to restore images. In this survey, we review several impulse noise removal techniques reported in the literature and identify efficient implementations. We analyse and compare the performance of different reported impulse noise reduction techniques with Restored Mean Absolute Error (RMAE) under different noise conditions. Also, we identify the most efficient impulse noise removing filters. Marking the maximum and minimum performance of filters helps in designing and comparing the new filters which give better results than the existing filters.

KEYWORDS

Impulse Noise, Image Restoration, Image Enhancement, Image De-Noising, Adaptive Filters.

1. INTRODUCTION

Noise is any unwanted signal present in the original signal. Complete classification of noise is shown in Figure 1. Noise is broadly classified into two main categories: blur noise and impulse noise.

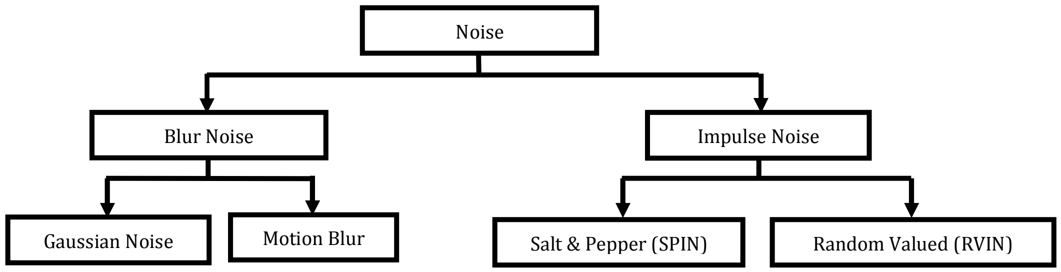
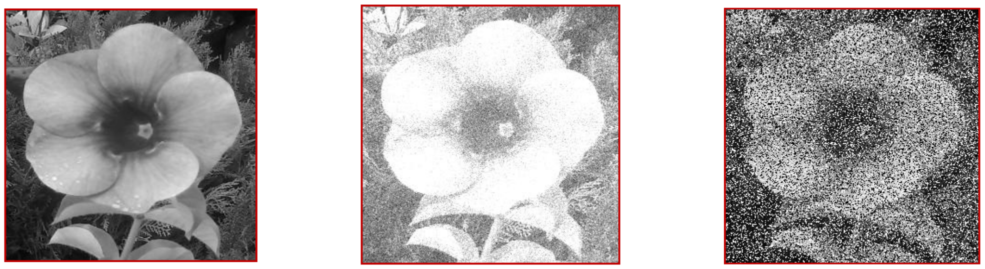


Figure 1: Classification of Image Noise Types

Blur noise is a uniform noise which makes the image blur. It modifies all the pixels present in the image by shifting pixel values towards low or high intensity level of the image (Gaussian blur). Sometimes, the noise blurs the image in a particular direction which is then called the motion blur. Main causes of blur noise are atmospheric turbulence, missed focus of camera lens, improper opening and closing of camera shutter and relative motion between camera and object. Impulse noise is a non-uniformly distributed noise. It modifies only select pixels from the image keeping the remaining pixels unchanged.

$$X_{ij} = \begin{cases} N_{ij} & \text{With Probability } p \\ S_{ij} & \text{With Probability } (1-p) \end{cases} \tag{1}$$

Where S_{ij} denotes the noiseless image pixel and N_{ij} the noise substituting for the original pixel.



(a) Original Image. (b) Original Image With 40% Gaussian Noise. (c) Original Image With 40% Impulse Noise.

Figure 2: Different Types of Noise

Impulse noise produces dot spots or patches on the image. Impulse noise has the property of either leaving pixels unmodified with probability $(\mathbf{1}-\mathbf{p})$ or replacing it all together with a probability of $\mathbf{p}$ as shown in equation (1). The sources of impulse noise are usually the result of error in transmission system, faulty sensors present in storage or capturing devices, and atmospheric or man-made disturbances. Outputs of both blur and impulse noises are shown in Figure 2.

$$X_{ij} = \begin{cases} 0 \text{ or } 255 & \text{Corrupted Pixel} \\ X_{ij} & \text{Non Corrupted Pixel} \end{cases} \tag{2}$$

$$X_{ij} = \begin{cases} 0 \text{ to } 255 & \text{Corrupted Pixel} \\ X_{ij} & \text{Non Corrupted Pixel} \end{cases} \tag{3}$$

In this paper, both fixed (salt and pepper) and variable (random valued) impulse noise models are considered for image de-noising. The Salt and Pepper Impulse Noise (SPIN) assumes a minimum value of 0 and a maximum value of 255 of noise, as shown in equation (2) and Random Valued Impulse Noise (RVIN) method assumes a noise value between the minimum value of 0 and the maximum value of 255 of noise, as shown in equation (3).

2. FILTERS

Impulse noise reduction algorithms are broadly classified in to two classes: linear and non-linear algorithms, as shown in Figure 3. In literature, many image de-noising algorithms for correcting the images corrupted by impulse noise are proposed. In a linear technique, the noise reduction method is applied linearly to all the pixels in the corrupted image without checking for the corrupted pixels, whereas in non-linear methods corrupted and non-corrupted pixels are determined first then the reduction techniques are applied for correcting the corrupted pixels only. Linear algorithms are time consuming and also blur the image and hence non-linear noise reduction algorithms are preferred. A set of conditions are considered in classifying corrupted and non-corrupted pixels in the non-linear algorithms.

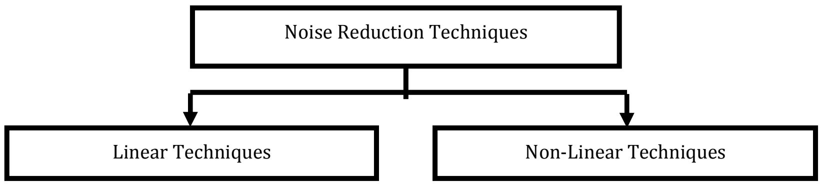


Figure 3: Classification of Reduction Techniques

2.1 Linear Filters

In linear techniques the noise reduction method is applied linearly to all the pixels in the corrupted image without checking for the corrupted pixels. Linear filters are simple filters with low computational complexities. Implementation of linear filters is simple compared to non-linear filters. Mean and median based filters are considered as most popular non-linear filters. Theoretically, mean based filter produce low mean square error whereas median based filter produce good noise tolerance.

2.1.1 Mean Filter (MMF)

In simple mean filters odd length fixed sized scanning window is used to get restored image from the corrupted image. With the help of the scanning window corrupted image pixels are scanned in horizontal and vertical directions. In each scan test pixel is replaced by the mean value of the scanning window pixels. For our comparative implementation of the mean filter a $(5x5)$ sized scanning window is used. In this filter mean value is calculated by taking ratio of sum of all pixels present in the scanning window and total number of pixels as shown in equation (4).

$$I(0,0) \quad = \quad \frac{\sum_{n=-K}^{K}\sum_{m=-K}^{K} I(m,n)}{(2K+1)\times(2K+1)} \qquad (4)$$

2.1.2 Weighted Mean Filter (WMMF)

Weighted mean filter is another modified basic mean filter. In this filter weights are assigned to scanning window pixels. Mean value of the products of pixel values and their corresponding weights are used as restoration value of the test pixel as shown in equation (5). Based on different parameters such as distance from the window center pixel, direction and position in ordered statistics, etc. different filters use different ways of weight calculations. Usually, window center pixel is assigned more weights compared to the other neighboring pixels of the window. Weighted mean filter is used when more than one parameter or feature of the window pixels influence the restoration value. For our comparative implementation of weighted mean filter a $(5x5)$ sized window and weight window shown in Figure 4 is used. Weighted window is designed based on the Euclidean distance of pixels from window center pixel.

$$I(0,0) \quad = \quad \frac{\sum_{n=-K}^{K}\sum_{m=-K}^{K} I(m,n)\times W(m,n)}{\sum_{n=-K}^{K}\sum_{m=-K}^{K} W(m,n)} \qquad (5)$$

1	1	1	1	1
1	2	2	2	1
1	2	3	2	1
1	2	2	2	1
1	1	1	1	1

Figure 4: Weight Window

2.1.3 Trimmed Mean Filter (WMMF)

Trimmed mean filter is an improved version of the mean filter where test pixel is replaced by trimmed mean value of the window pixels. To calculate the trimmed mean value of the pixels, trimming value u must be known. All available window pixels are arranged in ascending order of their intensity values. From ordered list of pixels first u and last u pixels are removed and the mean value of the remaining pixels is used as the restoration value for the corrupted pixels. The process of removing the first and the last set of pixels from an ordered list is called trimming. The trimming process increases the accuracy or sharpness of the calculated mean. Calculate the mean of the remaining ordered pixel list and this calculated mean will be used as a restoration value of the test pixel. For our comparative implementation of the trimmed mean filter a $(5x5)$ sized window and $u = 3$ are used.

2.1.4 Trimmed Weighted Mean Filter (TWMF)

Trimmed weighted mean filter is a combination of both trimmed and weighted mean filters. This feature catch the power of trimmed filter sharped mean value and also power of utilizing more than one feature of window pixels. Combined effect of both filters produce improved performance. For our comparative implementation of weighted trimmed mean filter $(5x5)$ sized window, weight window and $u = 2$ are used. In this filter window pixels are arranged in sorted order and then first u and last u pixel values are removed from ordered list. For restoration weighted mean of remaining pixels is used.

2.1.5 Median Filter (MF)

A median filter is the most popularly used one because of its inherent high fault-tolerance. In this filter window pixels values are sorted in order and then the median value is used as the test pixel value or restoration value. Mean filter is effective in minimizing the mean square error value of the estimation. Median filter algorithm produces good visibility in the restored image. Many improved versions of median filters are proposed by adding new features to the existing filters. For our comparative implementation of median filter a $(5x5)$ window is used. In this filter all selected window pixels are arranged in sorted order and the median value is used for restoration as shown in equation (6-9).

$If\ n\ is\ odd$

$$X \;=\; Sort(X) \tag{6}$$

$$I(0,0) \;=\; X\left(\frac{n+1}{2}\right) \tag{7}$$

$If\ n\ is\ even$

$$X \;=\; Sort(X) \tag{8}$$

$$I(0,0) \quad = \quad \frac{X\left(\frac{n}{2}\right)+X\left(\frac{n+2}{2}\right)}{2} \qquad (9)$$

2.1.6 Weighted Median Filter (WMF)

In weighted median filter, weights are assigned to window pixels. All weighted window pixels are arranged in order and the weighted median value is considered as a replacement value of test pixel. For our comparative implementation of weighted median filter a $(5x5)$ window and weight as shown in Figure 4 is used.

2.1.7 Trimmed Weighted Median Filter (TWMF)

All window pixels are arranged in order and using trimming value u first u and last u values are removed. For the remaining values, the weighted median filter technique is applied to get the restoration value. For our comparative implementation a $(5x5)$ weight window as shown in Figure 4 and trimming variable u =3 are used.

2.2 Non-Linear Filters

In non-linear methods corrupted and non-corrupted pixels are determined first and then the reduction techniques are applied on the corrupted pixels thus detected. Linear algorithms are time consuming and also blur the image and hence non-linear noise reduction algorithms are preferred. A set of conditions are considered in identifying corrupted and non-corrupted pixels in the non-linear algorithms.

2.2.1 Adaptive Median Filter (AMF)

In the adaptive median filter [1], noisy pixels are replaced by the adaptive median value. Here, adaptive means suitable for the existing noise. According to AMF, median value must be present between minimum and maximum values of pixel list. If calculated median value is not adaptive median then the correct median value is selected by increasing the window size of the scanning window. The AMF consists of two levels. The first level tests for the presence of residual impulses in the median filter output. If the first level asserts that there is no impulse in the median filter output, then the second level tests whether the center pixel itself is corrupted by an impulse or not. If the center pixel is decided as uncorrupted, then AMF leaves it as it is without filtering. If not, the output of the AMF is replaced by the median filter output at the first level. On the other hand, if the first level asserts that there is an impulse in the median filter output, then we simply increase the window size for the median filter and repeat the first-level test.

2.2.2 Progressive Switching Median Filter (PSMF)

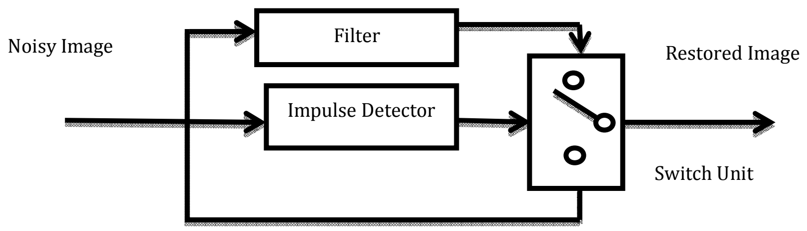


Figure 5: Block Diagram of Progressive Switching Median Filter

Progressive switching median filter [2] progressively or iteratively identifies the noisy pixels and replaces them iteratively. For the first iteration, a 3x3 window is used and, gradually, the window size is increased in each iteration. It is switching in nature because in the first stage it identifies the noisy pixels and it switches to the second phase of the algorithm for replacing the noisy pixels. First phase of the algorithm, namely, the iterative noise detection uses the threshold value to compare with the absolute difference of center pixel and the adaptive median value. If the difference is greater than the test pixel value the test pixel is considered as noisy pixel and is replaced by the adaptive median value. The noisy pixels processed in the one iteration are used to help the process of the other pixels in the subsequent iterations. The main advantage of this method is that the impulse pixels located in the middle of the large noise blotches can also be properly detected and filtered and, therefore, better restoration results are expected.

2.2.3 Tri-State Median Filter (TSMF)

Tri-state median filter [3], before applying filtering unconditionally, incorporates the Standard Median (SM) filter and Center Weighted Median (CWM) filter into the noise detection framework to determine whether the pixel is corrupted. Noise detection is realized by an impulse detector, which takes the outputs from the SM and CWM filters and compares them with the origin or center pixel value in order to make a tri-state decision (Figure 6). The switching logic is controlled by a threshold T and the output of TSM filter is obtained by equation (10).

$$Y^{TSM_{ij}} = \begin{cases} X_{ij} & T \geq D1 \\ Y^{CWM_{ij}} & D2 \leq T < D1 \\ Y^{SM_{ij}} & T < D2 \end{cases} \qquad (10)$$

where $Y^{CWM_{ij}}$ and $Y^{SM_{ij}}$ are the outputs of CWM and SM filters, respectively, and $d1 = |X_{ij} - Y^{SM_{ij}}|$ and $d2 = |X_{ij} - Y^{CWM_{ij}}|$.

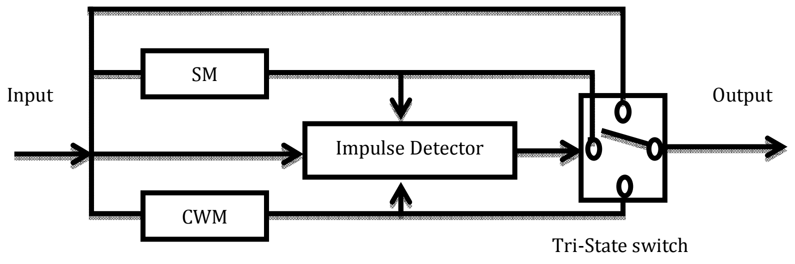


Figure 6: Block Diagram of Tri-State Median Filter

2.2.4 Adaptive Fuzzy Switching Filter (AFSF)

The adaptive fuzzy switching filter [4] is composed of three cascaded subunits. The first subunit aims at detecting impulse noise by considering grayscale distribution among neighboring pixels. The second subunit implements the grayscale estimation according to the information of neighboring pixels. The final subunit suitably modifies the value of this correction in order to further improve the detail preservation by fuzzy switching.

0	0	0	0	0
0	0	0	0	0
-1	-1	4	-1	-1
0	0	0	0	0
0	0	0	0	0

0	0	-1	0	0
0	0	-1	0	0
0	0	4	0	0
0	0	-1	0	0
0	0	-1	0	0

0	0	0	0	-1
0	0	0	-1	0
0	0	4	0	0
0	-1	0	0	0
-1	0	0	0	0

-1	0	0	0	0
0	-1	0	0	0
0	0	4	0	0
0	0	0	-1	0
0	0	0	0	-1

Figure 7: Four 5x5 Convolution Kernels

$$W(i,j) = min\{|I(i,j) * Kp|: p = 1-4\} \tag{11}$$

where Kp is a P^{th} kernel

$$M = Median\{r(.)\} \tag{12}$$

$$AVG = \frac{[Y(i-1,j-1)+Y(i-1,j)+Y(i-1,i+1)+Y(I,j-1)]}{4} \tag{13}$$

$$E(i,j) \quad = \quad \begin{cases} M & if\ K > 0 \\ AVG & K = 0 \end{cases} \tag{14}$$

where $r(.)$ denotes the pixels not discarded, K denotes the number of pixels having value not equal to maximum or minimum value of pixels in the window.

$M[W(i,j)]\ €[0,1]$ is the membership function of $W(i,j)$ that indicates how much a pixel looks like an impulse noise. This gives the following fuzzy rules:

$$[Rule\ 1]\ If\ W(i,j)\ is\ large, then\ M[W(i,j)]\ \ is\ large. \tag{15}$$

$$[Rule\ 2]\ If\ W(i,j)\ is\ small, then\ M[W(i,j)]\ is\ small. \tag{16}$$

According to the above rules, s-function is used to describe the membership function of the impulse noise corruption extent of the current pixel.

$$M[W(i,j)] \quad = \quad \begin{cases} 0 & if\ W(i,j) < \alpha \\ 2\left(\frac{W(i,j)-\alpha}{\gamma-\alpha}\right)^2 & if \alpha \le W(i,j) \le \beta \\ 1-2\left(\frac{W(i,j)-\gamma}{\gamma-\alpha}\right)^2 & if\ \beta \le W(i,j) \le \gamma \\ 1 & if\ W(i,j) > \gamma \end{cases} \tag{17}$$

where $\beta = \frac{(\alpha+\gamma)}{2}$

Hence, the filter based on adaptive fuzzy rules proposed produces the output value:

$$Y(i,j) = I(i,j) + M[W(i,j)]\ x\ [E(i,j) - I(i,j)] \tag{18}$$

Membership function $M[W(i,j)] = 0$ means that the current pixel $I(i,j)$ is a noise-free pixel and it needs no filtering. The filter will output the original pixel and preserve the image detail. If the membership function $M[W(i,j)] = 1$, then the current pixel $I(i,j)$ has been corrupted absolutely by impulse noise and it needs filtering. The filter will output $E(i,j)$ that is the estimation of the current pixel $I(i,j)$. If the membership function is such that $0 < M[W(i,j)] < 1$, then the current pixel has been corrupted somewhat by impulse noise. The filter will output the weighted average of $I(i,j)$ and $E(i,j)$. Figure 7 shows four convolution kernels used in our implementation and analysis.

2.2.5 Novel Impulse Noise Detection (NIND)

The novel impulse noise-detection algorithm [5] is based on the order statistics within a local window. Impulse detection algorithm is developed based on the following two assumptions. First, a noise-free image should be locally smoothly varying and is separated by edges. Second, a noise pixel takes a gray value substantially larger than or smaller than those of its neighbors. C denote the corrupted, noisy image of size $l1 \times l2$, and xij is its pixel value at position (i, j). In order to judge whether xij is an impulse pixel, first sort the 9 pixel values in the 3x3 window centered about it in ascending order. Suppose that the 9 sorted pixel values are $X(1) <= X(2) <= \cdots <= X(9)$, which is also called the order statistics. Next, calculate the mean t of the 3 pixel values in the middle, i.e., $t = 1/3(X(4) + X(5) + X(6))$. It is well known that a noisy pixel (an impulse) is usually located near one of the two ends in the ascending order. Therefore, only 3 pixel values in the middle are used in the computation of mean t. After applying the above procedure to each pixel in the noisy image, a noise map with pixel value $y(0)ij = |xij - t|$ at position (i, j) is generated. Similarly, apply the procedure above on noise map $y(0)\ ij$ to produce its corresponding noise map $y(1)\ ij$ and so on. Finally the generated noise map noise replacing algorithms are applied to restore the corrupted image. Figure 8 below illustrates this process.

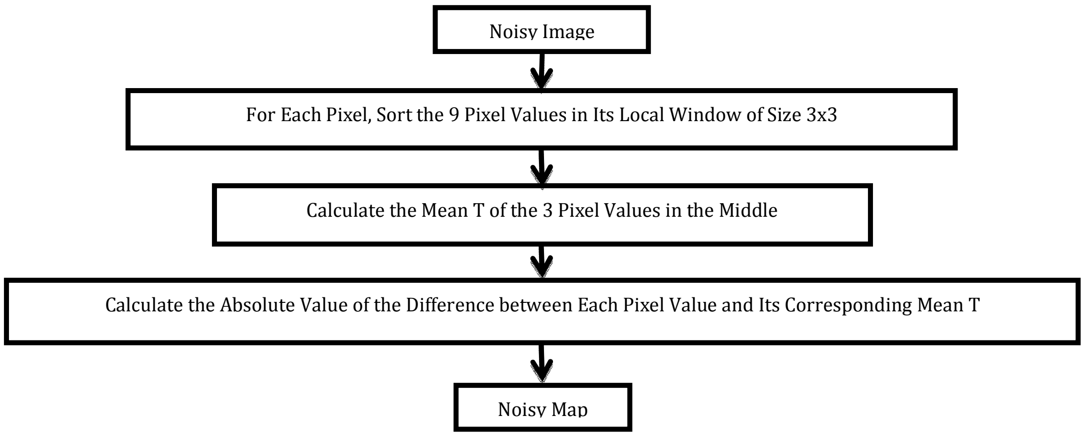

Noisy Image
For Each Pixel, Sort the 9 Pixel Values in Its Local Window of Size 3x3
Calculate the Mean T of the 3 Pixel Values in the Middle
Calculate the Absolute Value of the Difference between Each Pixel Value and Its Corresponding Mean T
Noisy Map

Figure 8: Generation of a Noise Map for NIDBOSF

2.2.6 An Efficient Algorithm for the Removal of Impulse Noise (AEAFRIN)

In this algorithm [6], noise pixels are identified using the assumption that noisy pixels are located near one of the ends in the ordered pixel list.

$$Nij = \begin{cases} 1 & xij \le X(u) \ or \ xij >= X(T-u+1) \\ 0 & Otherwise \end{cases} \tag{19}$$

where $T = (2Ld + 1)^2$, u is constant, and $1 <= u <= (T - 1)/2$. $Nij = 1$ indicates that xij is located near one of the two ends in the ordered samples of pixels $Wij\ (I)$. However, Nij alone is not enough to detect impulse noises accurately. The rank-ordered absolute differences (ROAD) combine with Nij to achieve superior impulse noise removal performance.

$$Tij = ROAD \times Nij \tag{20}$$

$$Bij = \begin{cases} 0 & Tij = W1 \\ \frac{(Tij - W1)}{(W2 - W1)} & W1 < Tij < W2 \\ 1 & Tij >= W2 \end{cases} \tag{21}$$

Where $W1$ and $W2$ are two pre-determined parameters. $Bij = 0$ indicates that the pixel xij is noise-free while $Bij = 1$ indicates that it is completely corrupted. When $0 < Bij < 1$, it measures how much the pixel xij is damaged. Calculate the membership function Bij for each pixel (i,j); the pixel value of xij is replaced by a linear combination of its original value xij and the median $mij\,(I\,)$ of the local window $Wij\,(I)$ with the current pixel xij excluded.

$$Yij = (1 - Bij\,) \times xij + Bij \times mij\,(I\,) \tag{22}$$

where Yij is the restored value of xij . It is clear that for a noise-free pixel $(Bij = 0)$, its value is unchanged, i.e.,$Yij = xij$. For a heavily corrupted pixel $(Bij = 1)$, its value is replaced by the median, i.e., $Yij = mij\,(I\,)$. For all other pixels $(0 < Bij < 1)$, the restored pixel value Yij is a linear combination of xij and $mij\,(I\,)$ as in equation of Yij.

2.2.7 Decision-Based Algorithm (DBA)

In DBA [7], the detection of noisy and noise-free pixels is decided by checking whether the value of a processed pixel element lies between the maximum and minimum values that occur inside the selected window. This is because the impulse noise pixels can take the maximum and minimum values in the dynamic range of (0, 255). If the value of the pixel processed is within the range, then it is an uncorrupted pixel and left unchanged. If the value does not lie within this range, then it is a noisy pixel and is replaced by the median value of the window or by its neighborhood values. If the noise density is high, there is a possibility that the median value is also a noisy value. In the latter case, the pixel processed is replaced by the previously processed adjacent neighborhood pixel value in place of the median value.

2.2.8 Improved Adaptive Median Filtering (IAMF)

In [8], the difference of current central pixel with median of local neighborhood pixels is used to classify the central pixel as noisy or noise-free. The noise is attenuated by estimating the values of the noisy pixels with a switching based median filter applied exclusively to those neighborhood pixels not labeled as noisy. The size of filtering window is adaptive in nature, and it depends on the number of noise-free pixels in the current filtering window.

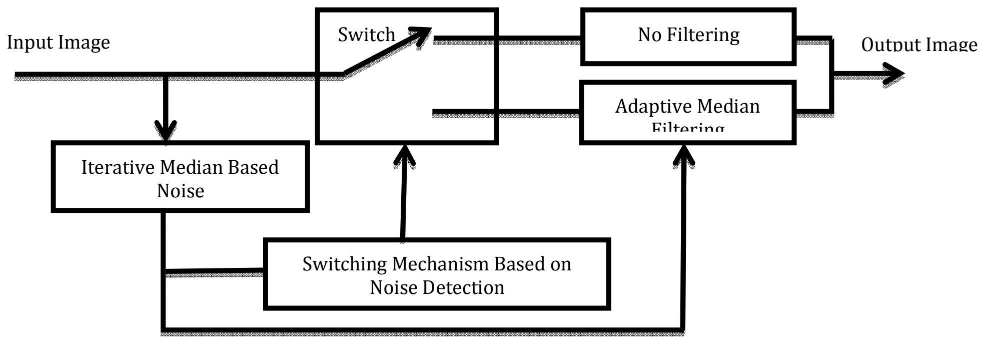

Input Image
Switch
No Filtering
Output Image
Adaptive Median Filtering
Iterative Median Based Noise
Switching Mechanism Based on Noise Detection

Figure 9: Block Diagram of Improved Adaptive Median Filter

Simulation results indicate that this filter is better able to preserve 2-D edge structures of the image and delivers better performance with less computational complexity as compared to other de-noising algorithms existing in literature.

2.2.9 Robust Statistics Based Algorithm (RSBA)

As in [9], let X denote the noise corrupted image and for each pixel $X\ (i,j)$ denoted as Xi,j a sliding or filtering window of size $(2L+1)\ X\ (2L+1)$ centered at Xi,j is defined as shown in Figure 10. The elements of this window are $Si,j\ =\{Xi-u,j-v\ ,-L<=u,v\ <=\ L\}$.

$X(i-1,j-1)$	$X(i-1,j)$	$X(i-1,j+1)$
$X(i,j-1)$	$X(i,j)$	$X(i,j+1)$
$X(i+1,j-1)$	$X(i+1,j)$	$X(i+1,j+1)$

Figure 10: A 3x3 Filtering Window with X(i,j) as Center Pixel

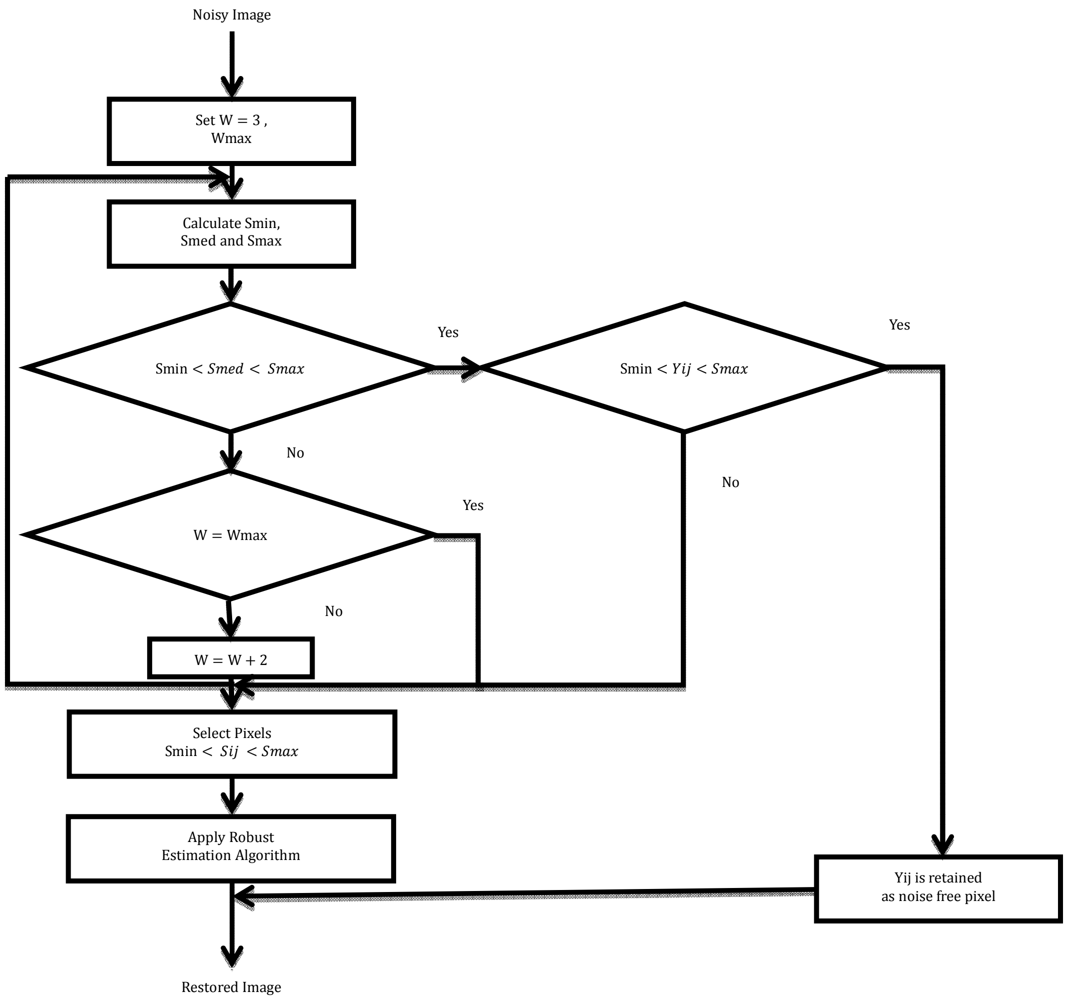

Noisy Image
Set W = 3 , Wmax
Calculate Smin, Smed and Smax
Smin < Smed < Smax
Yes
Smin < Yij < Smax
Yes
No
No
W = Wmax
Yes
No
W = W + 2
Select Pixels Smin < Sij < Smax
Apply Robust Estimation Algorithm
Yij is retained as noise free pixel
Restored Image

Figure 11: Block Diagram of Robust Statistics Based Algorithm

Algorithm 1

1. Set the minimum window size $w = 3$.
2. Read the pixels from the sliding window and store it in S.
3. Compute minimum $(Smin)$, maximum $(Smax)$ and median value $(Smed)$ inside the window.
4. If the center pixel in the window $X(i,j)$,is such that $Smin < X(i,j) < Smax$, then it is considered as uncorrupted pixel and retained. Otherwise go to step 5.
5. Select the pixels in the window such that $Smin < Sij < Smax$ if number of pixels is less than 1 then increase the window size by 2 and go to step 2 ,else go to step 6.
6. Difference of each pixel inside the window with the median value $(Smed)$ is calculated as x and applied to robust influence function.

$$f(x) = 2x/(2t^2 + x^2) \tag{23}$$

Where t is outlier rejection point, is given by,

$$t = \frac{ts}{\sqrt{2}} \tag{24}$$

Where ts is the maximum expected outlier and is given by,

$$ts = k(tN) \tag{25}$$

7. Where tN is the local estimate of the image standard deviation and k is a smoothening factor. Here $k = 0.3$ is taken for medium smoothening.

Pixel is estimated using equation (26) and (27),

$$S1 = \sum_{l \in L}(pixel\,(l) * f(x)/x) \tag{26}$$

$$S2 = \sum_{l \in L} f(x)/x \tag{27}$$

where l is number of pixels in the window, ratio of $S1$ and $S2$ gives the estimated pixel value.Figure 11 shows the block diagram of robust statistics based algorithm reported in [9].

2.2.10 Decision Based Adaptive Filter (DBAF)

The decision based adaptive filter [10] is two stage algorithm. In the first stage noise candidates are identified using rank ordered absolute difference (ROAD) value. In the second stage replacement is done by median of uncorrupted pixels in the filtering window. The filtering window is varied adaptively based on the number of uncorrupted pixels in the window. The ROAD value of the processing pixel is calculated by summing the five absolute difference values in the sorted array. A binary image is obtained by comparing the ROAD value with the pre-defined threshold value. This process is repeated for the entire image. In the binary image value 0 corresponds to noisy pixels and 1 corresponds to noise-free pixels. In the correction stage, the corrupted pixel is replaced by the median of the uncorrupted pixels in the $3x3$ window. The window size is increased if the number of uncorrupted pixels is less than three. This process is repeated for the entire image to get the restored image.

Noise Detection

1. A $3x3$ detection window centered at $X(i,j)$ is applied to the corrupted image. The absolute difference with the center pixel is obtained using equation (28).

$$D = \left|X_{ij} - X_{i+k,j+l}\right|, \quad -L_c \le k,l \le +L_c \tag{28}$$

2. The array D is sorted and the sum of five smallest absolute differences are calculated. This gives a $ROAD$ value for the current pixel as shown in equation (29).

$$ROAD = \sum_{n=1}^{5} D(n) \tag{29}$$

3. The $ROAD$ value is checked wirth the predefined threshold value. Based on the condition current pixel is detected as corrupted or uncorrupted pixel. For optimal performance threshold value is set as 40 for all images. The above steps are repeated for entire image and binary image B of size $M \times N$ is obtained using equation (30).

$$B = \begin{cases} 1 & ROAD(i,j) < 40 \\ 0 & ROAD(i,j) \geq 40 \end{cases} \quad (30)$$

Noise Filtering

1. A $W \times W$ correction window centered at $B(i,j)$ is applied to the flag image B, where $W = 2Lc + 1$ and initial Lc is 1. If $B(i,j) = 0$ a result image R is obtained by multiplying binary image segment with noisy image segment by using equation (31).

$$R = X_{i+k,j+l} \times B_{i+k,j+l}, \qquad L_c \leq k,l \leq L_c \quad (31)$$

2. In R , number of non-zero pixels are three and above, $X(i,j)$ is replaced by median of non-zero pixels in R. If numbers of non-zero pixels are less than three, Lc is incremented by one and step1 and step2 are repeated. The same process is repeated for entire image.

2.2.11 Min-Max Detector Filter (MDBF)[11]

In [11] the authors propose a min-max based simple and efficient algorithm to detect the impulse noise. A 3x3 scanning window is used to scan the given corrupted image and center pixel of window is compared with its 8 neighboring pixels present within the window. If test pixel is less than the minimum value present in its neighboring pixels or greater than maximum value then test pixel is considered as corrupted pixel and its value is replaced by median value of its neighboring pixels. Otherwise, test pixel is considered as non-corrupted pixel and it is kept as it is and the window is moved to the next pixel. The algorithm is outlined below.

MDB Filter Algorithm

1. Take corrupted image (x).
2. Take a 3x3 window (w). Let the center pixel be the test pixel.
3. Shift the window row wise then column wise to cover the entire pixel in the image and repeat step4 and step5.
4. If (test pixel< min (rest of the pixel in w) or (test pixel> max (rest of the pixel in w) then the test pixel is corrupted.
5. If the test pixel is corrupted apply median filter to the test filter in the window w.
6. Stop.

2.2.12 Detail Preserving Adaptive Filter (DPAF)

The paper [12] proposes an effective and efficient method of impulse noise removal which not only removes noise but also preserves the image details. The algorithm first classifies the pixels as noisy and noise-free based on its N8(p) (Figure 12) neighbours using averaging parameters introduced here and then replaces the noisy pixel by the adaptive median of the pixel. The algorithm uses adaptive median as it provides better de-noising. Since the proposed algorithm performs prior classification of pixels as noise and noise-free this method preserves image details. The algorithm is outlined below.

$X(i-1,j-1)$	$X(i-1,j)$	$X(i-1,j+1)$
$X(i,j-1)$	$X(i,j)$	$X(i,j+1)$
$X(i+1,j-1)$	$X(i+1,j)$	$X(i+1,j+1)$

Figure 12: N8(p) of A Pixel

$$A_H = \frac{1}{2}\sum_{i=1}^{2} \Delta_{ih} \quad (32)$$

where $\Delta_{1h} = |X_{i,j} - X_{i,j-1}|$ and $\Delta_{2h} = |X_{i,j} - X_{i,j+1}|$

$$A_V = \frac{1}{2}\sum_{i=1}^{2} \Delta_{iv} \quad (33)$$

where $\Delta_{1v} = |X_{i,j} - X_{i-1,j}|$ and $\Delta_{2v} = |X_{i,j} - X_{i+1,j}|$

$$A_{MD} = \frac{1}{2}\sum_{i=1}^{2} \Delta_{imd} \quad (34)$$

where $\Delta_{1md} = |X_{i,j} - X_{i-1,j-1}|$ and $\Delta_{2md} = |X_{i,j+1} - X_{i,j+1}|$

$$A_{AD} = \frac{1}{2}\sum_{i=1}^{2} \Delta_{iad} \quad (35)$$

where $\Delta_{1ad} = |X_{i,j} - X_{i-1,j+1}|$ and $\Delta_{2ad} = |X_{i,j} - X_{i+1,j-1}|$

$$\varphi = Mean\{A_h, A_v, A_{md}, A_{ad}\} \quad (36)$$

Algorithm

```
1. Input noisy image.
2. For all pixels in the image check
   Xij ∈ s if min [w_m] < xij < max[w_m]
   Xij ∈ nc if min[w_m] = xij or xij = max [w_m]
3. For all pixels in nc calculate ah , av , amd , aad and φ and check
   If 0.90 ≤ φ ≤ 1 then xij ∈ n,Xij ∈ s otherwise
4. For all pixels in n,Calculate m_ad. replace each pixe l in n by its m_ad.
```

2.2.13 Universal De-Noising Framework (UDF)

In the filter reported in [13], new detector based on new statistics called Robust Outlyingness Ratio (ROR) is used to measure how much a pixel looks like an impulse noise. Based on the ROR, the pixels are divided into four different clusters. Then, different decision rules are adopted to detect the impulse noise in each cluster. During the detection process, the from-coarse-to-fine strategy is used. In addition, the detection process contains two stages, i.e., the coarse stage followed by the fine stage. Different thresholds are used in the two stages. Finally, the detection procedure is iteratively adopted. In order to calculate the ROR, the window size must be given. In this paper, the authors use a window size of 5x5. Since the ROR measures the outlyingness of the pixels, i.e., how impulse like, all pixels are divided into four levels (clusters) according to the ROR. The four clusters are: the most likely cluster ROR; the second likely cluster ROR; the third likely cluster ROR; and the fourth likely cluster ROR. The lower the ROR, the lower impulse like of the pixel in its neighbors.

The Coarse Stage:

1. Choose the algorithm parameters, i.e., coarse thresholds tc1, tc2 , and tc4; window size n (the actual size is (2n+1)x(2n+1)); iterations mc=1 ; and initial j=1.
2. Initialize the detection flag matrix map as zeros, where “0s” and “1s” represent good and noisy pixels, respectively.
3. Calculate the ROR of the current pixel. If the ROR is in the fourth level, treat it as a good pixel, or calculate the absolute deviation d between the current pixel and the median of its local window. Then, compare d with tck threshold according to its ROR value. If d is larger than tck , it is a noisy pixel, or it is a good pixel. Update the flag map according to the result.
4. Get the median-based restored image i according to the detection result. If the flag is 1, represent the pixel with the median of its local window, or do not change.
5. if j<=mc, j=j+1 , then go to step 2, or the coarse stage is completed.

The Fine Stage:

1. Choose the algorithm parameters, i.e., fine thresholds tf1, tf2, tf3, and tf4; window size n (the actual size is (2n+1)x(2n+1)); iterations mf ; and initial j=1.
2. Initialize the detection flag matrix map as zeros, where “0s” and “1s” represent good and noisy pixels, respectively.
3. Calculate the ROR of the current pixel and the absolute deviation between the current pixel and the median of its local window. Then, compare d with threshold tfk according to its ROR value. If is d larger than tfk, it is a noisy pixel, or it is a good pixel. Update the map flag according to the result.
4. Get the median-based restored image with i the detection result. If the flag is 1, represent the pixel with the median of its local window, or do not change.
5. If j<=mf, j=j+1, then go to step 2, or the fine stage is completed.

3. PERFORMANCE MEASUREMENTS

To evaluate the performance of the impulse noise reduction algorithms the performance measure RMAE (Restored Mean Absolute Error) is used. The RMAE is measured using unit decibel (dB) and it is the amount of Mean Absolute Error (MAE) recovered by an algorithm. Mean absolute error gives the difference between the given two input images, as shown in equation (37). Low value of MAE indicates more similarity between the given images and vice versa. MAE value changes from 0 to 255. Zero value of MAE indicates that both images look exactly the same. As MAE the value increases towards 255 similarities between the images decreases. RMAE is calculated as the percentage ratio of the difference of corrupted image mean absolute error and restored image mean absolute error, as shown in equation (38). RMAE is the percentage amount of noise restored by an algorithm. Maximum value of RMAE is 100, indicating that both original and restored image are exactly the same meaning the restoration is 100%; that is, the algorithm has successfully restored all corrupted pixels. Sometimes, the algorithm returns a negative value of RMAE indicating that the algorithm is increasing the noise ratio instead of restoring the image. RMAE value is 100% if the restored image MAE is the same as the corrupted image MAE; all corrupted pixels are restored in this case. For good restoration algorithms, restored image MAE is less than the corrupted image MAE else we get negative RMAE value indicating bad restoration. MAE value increases and RMAE decreases with increase in noise ratio.

$$MAE = \frac{\sum_i^M \sum_j^N (X_{ij} - R_{ij})}{(M \times N)} \quad (37)$$

$$RMAE = 1 - \frac{(MAE\ of\ Corrupted\ Image - MAE\ of\ Restored\ Image)}{MAE\ of\ Corrupted\ Image} \quad (38)$$

Where

X	-	Original Image.	R	-	Restored Image
M X N	-	Size of Image.	MAE	-	Mean Absolute Error.
RMAE	-	Restored Mean Absolute Error.			

4. SIMULATION AND RESULTS

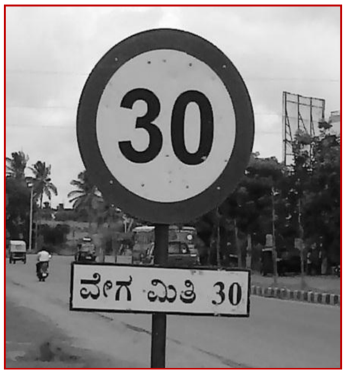
We have implemented most popular efficient linear and nonlinear algorithms and analyzed the results. Tables 1 show RMAE values of different filters for SPIN images 2 and the outputs are shown in Figures 13. The results are graphically represented in Figures 15 which was used to identify the most effective algorithms for SPIN. Tables 3 show RMAE values of different filters for RVIN images 3 and the outputs are shown in Figures 14; the results are graphically represented in Figures 16.

TABLE 1: RMAE OF FILTERS FOR SPIN IMAGE-2 (300X300)

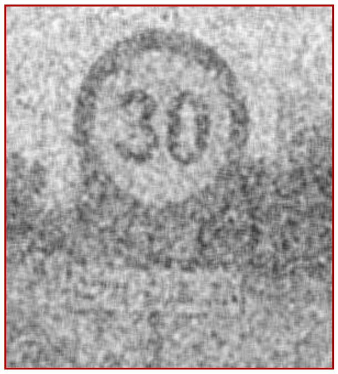
NOISE RATIO ▶ / FILTERS ▼	10	20	30	40	50	60	70	80	90
MMF (5X5)	-19.8	13.5	25.15	30.04	32.93	34.78	35.15	35.88	36.96
WMMF(5X5)	-14.56	15.65	26.04	30.16	33	34.14	35.53	36.38	36.86
TMMF(5X5)	28.2	44.33	41.9	39.25	37.65	35.82	35.35	34.03	33.95
WTMMF(5X5)	29.17	47.27	47.87	46.3	44.17	40.98	39.3	37.47	36.04
MF(5X5)	52.06	73.42	78.91	77.2	64.1	39.87	9.66	-6.56	-6.62
WMF(5X5)	21.2	42.06	42.04	38.42	29.41	13.41	-3.27	-11.28	-7.43
WTMF(5X5)	50.4	73.33	79.82	78.96	68.82	44.96	17.38	-0.46	-4.33
AMF	89.29	92.3	92.69	92.12	90.7	83.87	56.48	28.7	9.82
PSMF	74.36	81.68	83.38	79.97	66.11	35.55	7.34	-3.01	-3.94
TSMF	24.34	61.28	72.35	73.76	64.11	40.34	13.16	-3.69	-4.52
AFSF	77.21	85.48	87.89	87.89	85.39	80.83	72.33	62.05	47.39
NIND	80.87	86.98	86.8	78.08	51.02	13.83	-11.44	-15.78	-8.75
AEAFRIN	72.31	79.39	75.1	63.69	47.89	30.48	15.36	4.86	-0.12
DBA	95.28	95.06	94.11	93.17	92.19	89.95	85.91	70.64	21.19
IAMF	74.82	80.69	81.66	74.88	43.1	15.05	0.02	-10.97	-9.41
RSBA	89.33	91.24	90.87	89.62	88.2	84.41	57.55	23.38	6.63
DBAF	77.62	80.03	71.82	54.93	33.77	15.61	0.1	-7.65	-6.25
MDBF	89.29	90.54	84.94	75.97	61.85	46.27	31.3	18.27	7.99
DPAF	89.41	91.94	92.2	91.51	90.4	83.8	57.34	27.04	9.7
UDF	62.16	78.69	84.01	86.76	87.54	83.15	56.07	18.18	-2.86

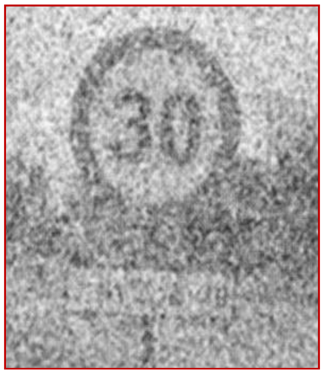
TABLE 2: RMAE OF FILTERS FOR RVIN IMAGE-3 (300X300)

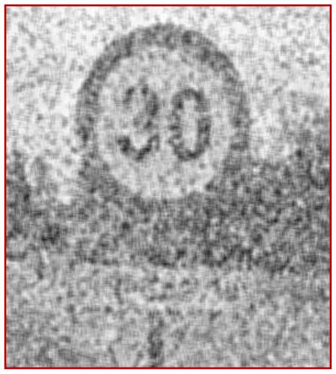
NOISE RATIO ▶ / FILTERS ▼	10	20	30	40	50	60	70	80	90
MMF (5X5)	-80.69	-21.44	-1.46	9.05	14.13	17.45	20	21.47	22.95
WMMF(5X5)	-70.68	-15.3	1.64	9.75	14.77	18.34	19.89	21.83	23.19
TMMF(5X5)	-29.01	17.48	27.12	28.82	28.48	27.8	26.91	26.49	25.72
WTMMF(5X5)	-25.38	18.13	28.09	29.55	28.82	28.78	27.95	27.11	26.33
MF(5X5)	4.6	48.83	62.06	65.68	63.22	54.57	43.75	33.84	26.62
WMF(5X5)	-47.3	19.78	40.19	47.76	48.91	41.87	32.87	24.09	16.15
WTMF(5X5)	-0.09	46.73	61.2	65.55	63.08	54.98	45.04	35.41	27.85
AMF	56.43	53.92	46.77	39.43	32.32	26.07	20	16.08	12.1
PSMF	49.38	67.3	73.51	74.85	71.49	64.08	51.88	37.45	26
TSMF	-34.03	29.27	51	59.05	57.36	50.77	40.51	32.25	25.88
AFSF	40.14	60.61	62.72	58.94	51.68	43.47	34.77	28.03	22.05
NIND	58.86	74.84	79.28	80.83	79.88	74.12	61.15	43.49	23.24
AEAFRIN	39.82	61.8	65.93	62.33	55.95	46.45	37.4	28.54	22.45
DBA	0.76	0.79	0.88	0.91	0.86	0.69	0.8	0.69	0.64
IAMF	52.95	67.3	71.08	71.68	65.6	40.38	17.67	6.94	2.41
RSBA	57.01	53.48	46.57	38.96	32.34	26.37	20.71	15.55	11.93
DBAF	49.44	66.79	68.95	65.04	56.96	47.63	37.63	29.47	22.28
MDBF	57.7	54.9	47.5	39.82	32.85	26.17	20.29	15.65	12.25
DPAF	56.7	54.63	48.18	40.07	32.25	26.3	20.17	16.08	12.4
UDF	17.04	54.92	66.02	70.93	71.57	66.6	54.59	41.67	32.56

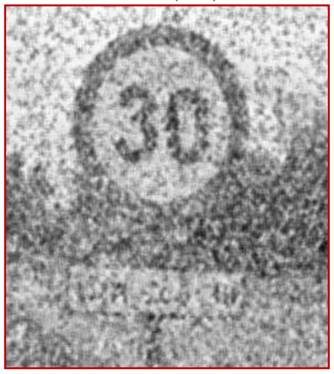


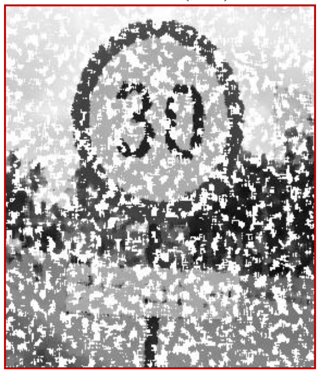
ORIGINAL IMAGE -2

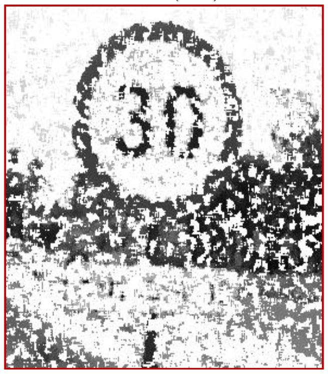

MMF (5X5)

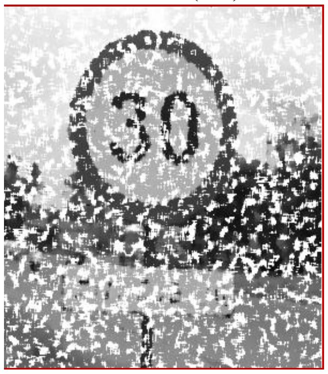

WMMF(5X5)

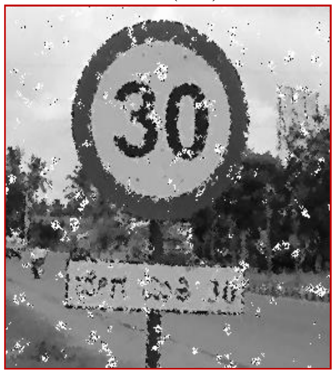

TMMF(5X5)

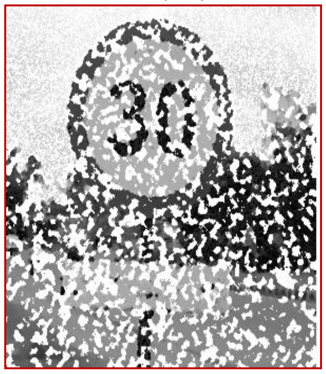

WTMMF(5X5)

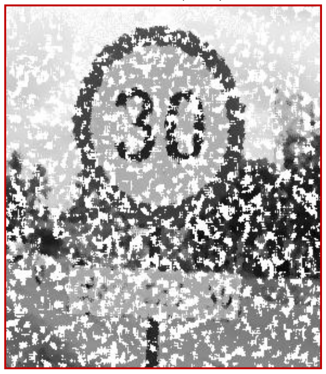

MF(5X5)

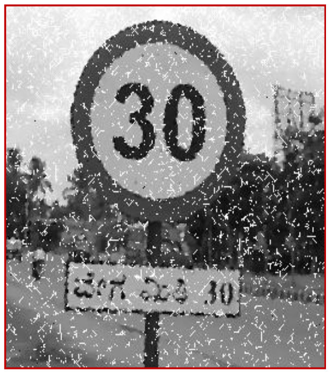

WMF(5X5)

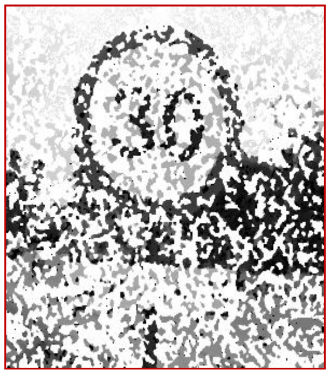

WTMF(5X5)

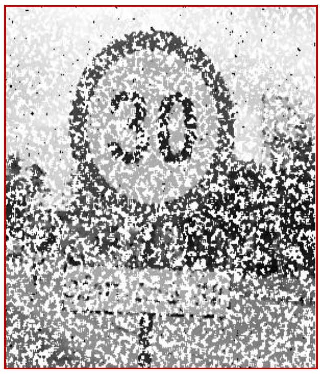

AMF

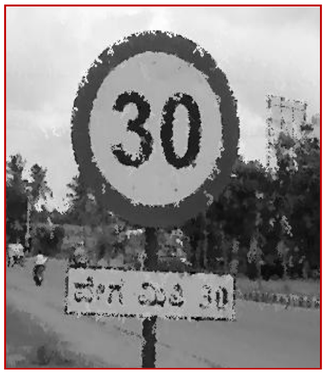

PSMF

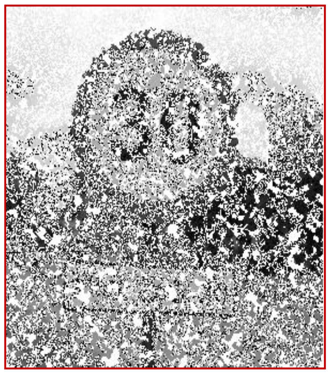

TSMF

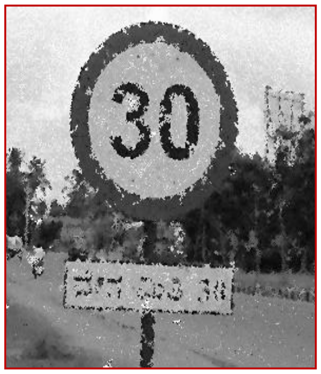

AFSF

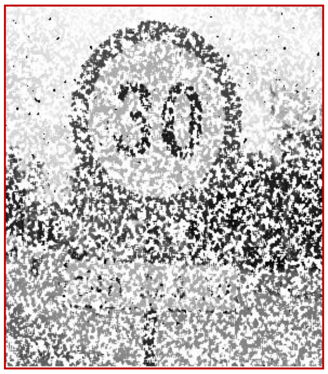

NIND

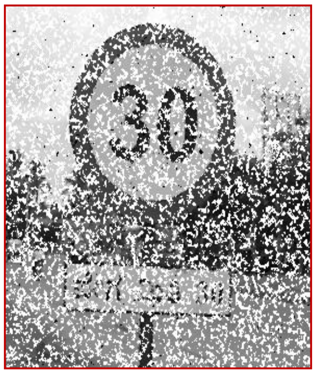

AEAFRIN

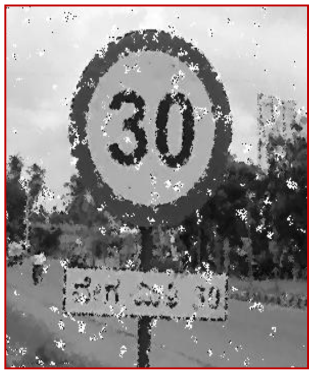

DBA


IAMF


RSBA


DBAF


MDBF


DPAF

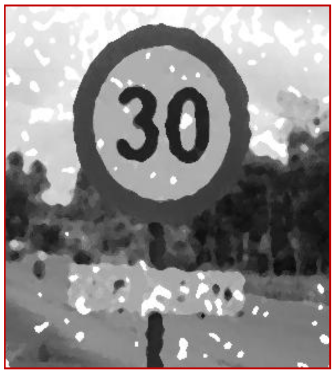
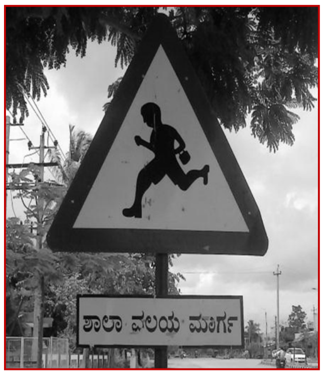
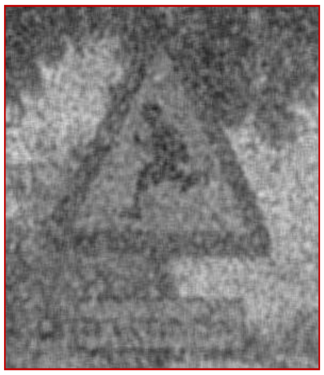
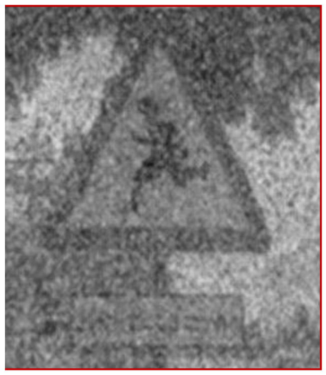

UDF

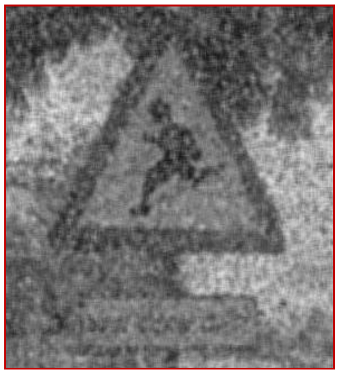
Figure.13 Results Of Filters For Image-2 (300x300) With 60% SPIN

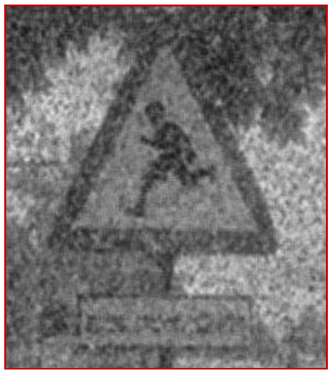


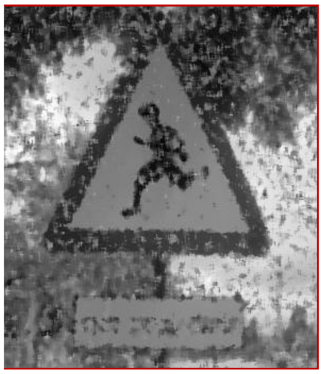
ORIGINAL IMAGE -3

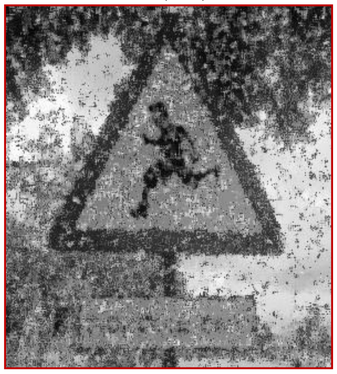

MMF (5X5)

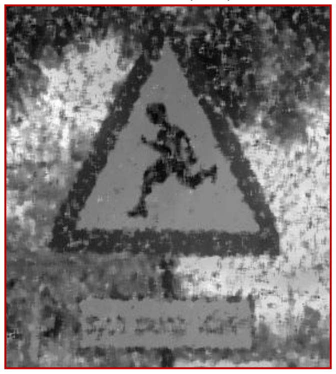

WMMF(5X5)

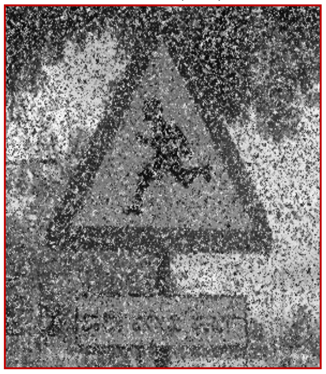

TMMF(5X5)

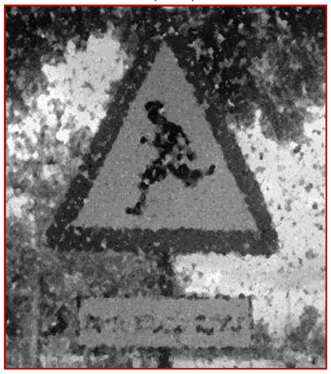

WTMMF(5X5)

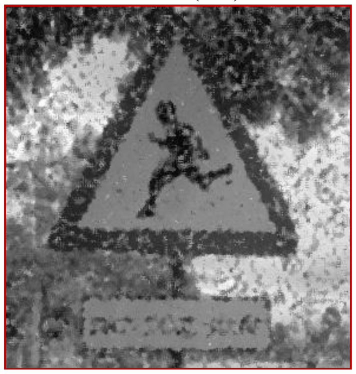

MF(5X5)

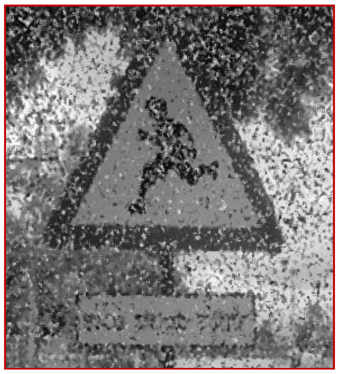

WMF(5X5)

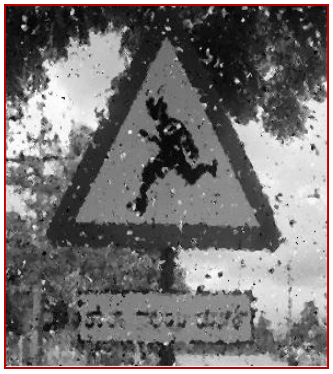

WTMF(5X5)

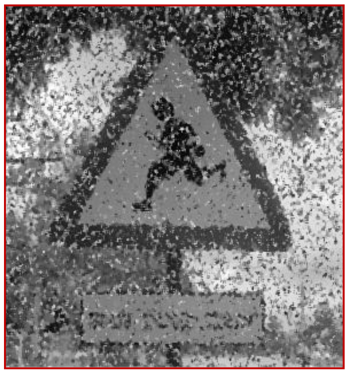

AMF

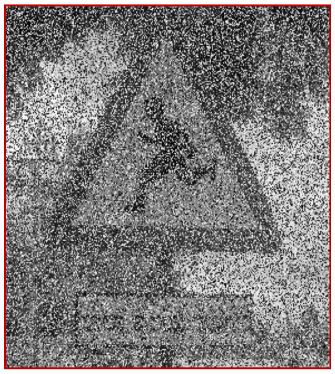

PSMF


TSMF


AFSF


NIND


AEAFRIN


DBA

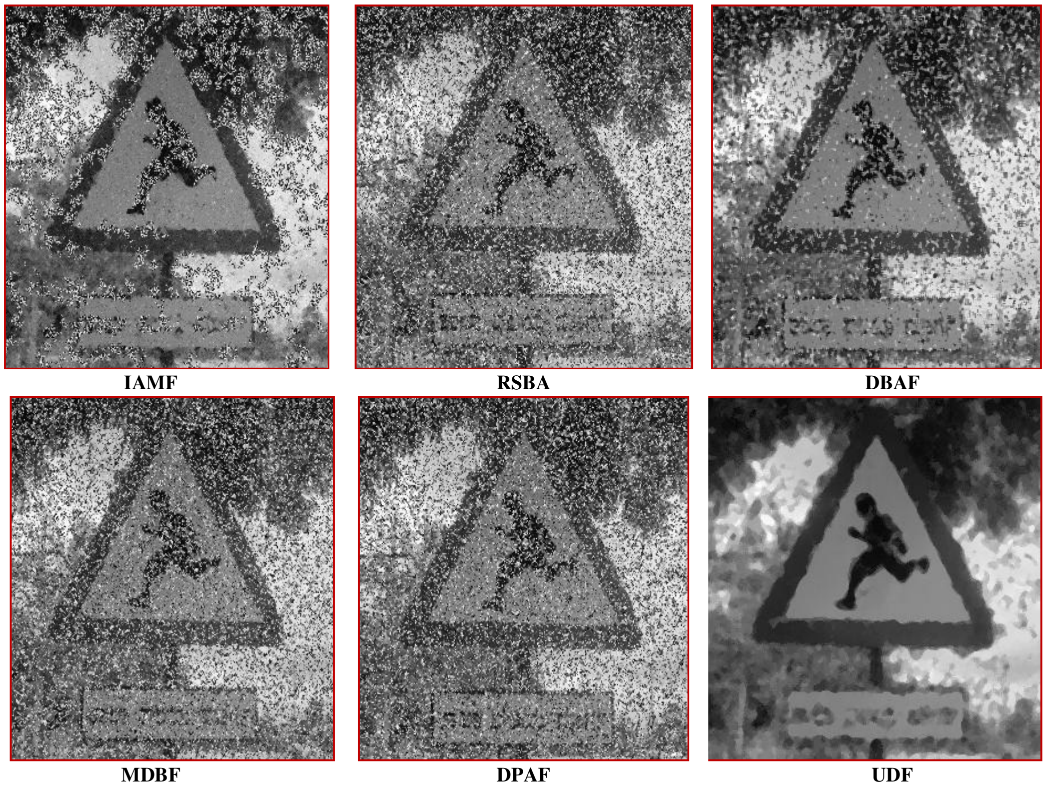


Figure 14: Results of Filters for Image-3 (300x300) with 60% RVIN

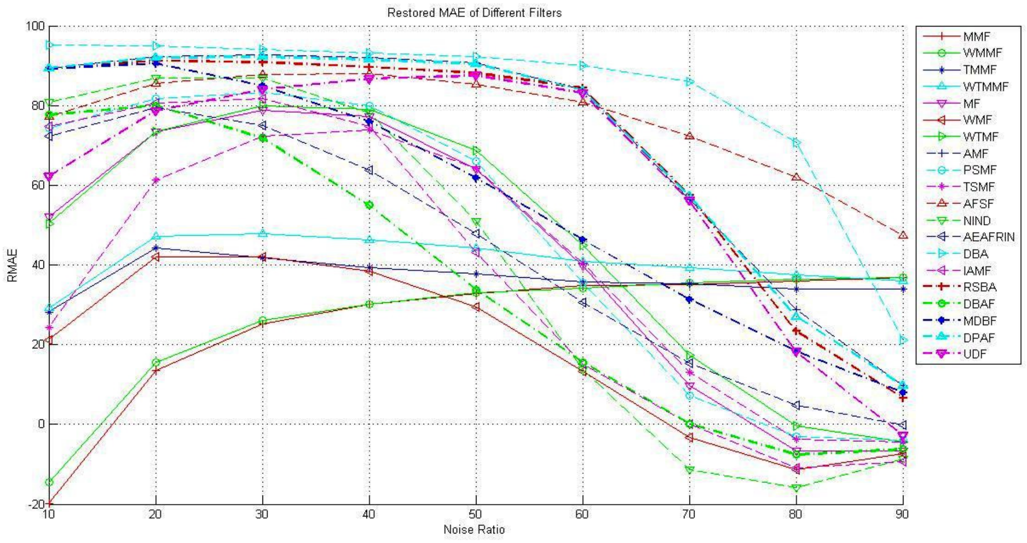


Figure 15: Performance of Filters for Image-2 SPIN

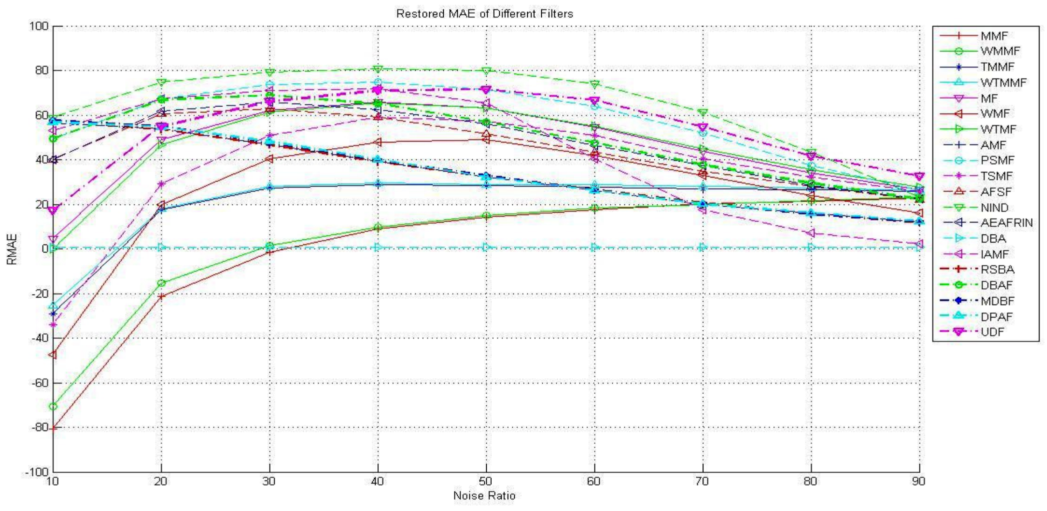


Figure 16: Performance of Filters for the Image-3 RVIN

5. CONCLUSIONS

We have implemented most popular and efficient linear and non-linear algorithms for impulse noise reduction. This paper presents the results and our analysis using different images corrupted with SPIN and RVIN. Our experimental results show that the non-linear filters produce better results compared to the linear filters. Further, our analysis also shows that handling SPIN is easier than RVIN. For SPIN noise, the DBF, AMF and DPAF filters show comparatively better results while for RVIN noise, the NIBOFS, UDF and WMF filters show comparatively better results.

REFERENCES

[1] H. Hwang and R. A. Haddad "Adaptive Median Filters: New Algorithms and Results" IEEE Transactions on Image Processing, Vol. 4, No. 4, April 1995, pp 499-502.

[2] Zhou Wang and David Zhang "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, Vol. 46, No. 1, January 1999, pp 78-80.

[3] Tao Chen, Kai-Kuang Ma, Li-Hui Chen "TriSstate Median Filter for Image Denoising" IEEE Transactions on Image Processing, Vol. 8, No. 12, December 1999, pp 1834-1838.

[4] Haixiang Xu, Guangxi Zhu, Haoyu Peng, Desheng Wang "Adaptive Fuzzy Switching Filter for Images Corrupted by Impulse noise" Pattern Recognition Letters 25 (2004) pp 1657–1663.

[5] Wenbin Luo "A New Impulse Detector Based on Order Statistics" Intl. J. Electronincs Communication (aeü) 60 (2006) pp 462–466.

[6] Wenbin Luo "An Efficient Algorithm for the Removal of Impulse Noise from Corrupted Images" Intl. J. Electron. Commun. (aeü) 61 (2007) pp 551 – 555.

[7] K. S. Srinivasan, D. Ebenezer "A New Fast and Efficient Decision-Based Algorithm for Removal of High-Density Impulse Noises" IEEE Signal Processing Letters, Vol. 14, No. 3, March 2007, pp 189-192.

[8] Mamta Juneja, Rajni Mohana "An Improved Adaptive Median Filtering Method for Impulse Noise Detection" International Journal of Recent Trends in Engineering, Vol. 1, No. 1, May 2009, pp 274-278.

[9] V.R.Vijaykumar, P.T.Vanathi, P.Kanagasabapathy, D.Ebenezer "Robust Statistics Based Algorithm to Remove Salt and Pepper Noise in Images" International Journal of Information and Communication Engineering 5:3 2009, pp 164-173.

[10] V.R.Vijaykumar, Jothibasu "Decision Based Adaptive Median Filter to Remove Blotches, Scratches, Streaks, Stripes and Impulse Noise in Image" Proceedings of 2010 IEEE 17th International Conference on Image Processing, September 26-29, 2010, Hong Kong, pp 117-120.

[11] S. K. Satpathy, S. Panda, K. K. Nagwanshi, C. Ardil "Image Restoration in Non-linear Filtering Domain Using MDB Approach" International Journal of Information and Communication Engineering 6:1 2010, pp 45-49.

[12] Krishna Kant Singh, Akansha Mehrotra, Kirat Pal, M.J.Nigam "A n8(p) Detail Preserving Adaptive Filter for Impulse Noise Removal" 2011 International Conference on Image Information Processing (ICIIP 2011).

[13] Bo Xiong, D. Zhouping Yin "A Universal Denoising Framework with a New Impulse Detector and Non-local Means" IEEE Transactions on Image Processing, Vol. 21, No. 4, April 2012, pp 1663-1675.

Authors Profile

Manohar Koli received a Bachelor's degree in Computer Science and Engineering from Visveswaraya Technological University, Belgaum and a Master's degree in Computer Science and Engineering from Kuvempu University, Shimoga. He is currently a research student at Tumkur University, Tumkur and holds the position of Assistant Professor in Computer Science and Engineering department of University BDT College of Engineering, Davangere. His research interests include image processing, pattern recognition and data mining.

Balaji S. received a Bachelor's degree in Mathematics from Madras University, Chennai, a Master's degree in Applied Mathematics from Anna University, Chennai and a Ph.D. degree in Computer Science and Engineering from Indian Institute of Science, Bangalore. He has served Indian Space Research Organization for more than two decades before taking up teaching. He is currently a Professor at City Engineering College, Bangalore. His research interests include fault-tolerant computing, real-time systems, mobile computing, image processing and computer vision, pattern recognition, data mining, energy efficient computing, energy management, energy harvesting, and wireless sensor networks.