Impulse Noise Detection and Removal Using Second Order Differential and Similar Nearest Neighbors

By

Saqib Rasheed
625-FBAS/MSCS/F10

Supervisor

Dr Ayyaz Hussain
Assistant Professor

Department of Computer Science & Software Engineering,
Faculty of Basic and Applied Sciences,
International Islamic University Islamabad, Pakistan

Impulse noise detection and removal using second order differential and similar nearest neighbors

Saqib Rasheed
625-FBAS/MSCS/F10

Supervisor
Dr Ayyaz Hussain
Assistant Professor,
Department of Computer Science & Software Engineering,
International Islamic University Islamabad, Pakistan

Department of Computer Science & Software Engineering,
Faculty of Basic and Applied Sciences,
International Islamic University Islamabad, Pakistan

Department of Computer Science and Software Engineering

International Islamic University, Islamabad

Dated: 29th November, 2012

FINAL APPROVAL

This is to certify that the department of Computer Science and Software Engineering, International Islamic university, Islamabad accepts this dissertation submitted by **Saqib Rasheed, Reg. No. 625-FBAS/MSCS/F10**, in its present form as satisfying the dissertation requirements for the degree of **MS in Computer Science.**

Committee:

SUPERVISOR

Dr. Ayyaz Hussain

Assistant Professor
Department of Computer Science & Software Engineering,
International Islamic University,
Islamabad

[illegible] 29/11/12

(Dr. Ayyaz Hussain)

INTERNAL EXAMINER

Dr. Rashid Bin Muhammad

Assistant Professor
Department of Computer Science & Software Engineering,
International Islamic University,
Islamabad.

[illegible]

(Dr. Rashid Bin Muhammad)

EXTERNAL EXAMINER

Dr. Zia-ul-Qayyum

Professor
Department of Computer Science,
Iqra University Islamabad

[illegible]

(Dr. Zia-ul-Qayyum)

Acknowledgement

I wish to express my sincere gratitude to my guide, assistant professor Dr. Ayyaz Hussain, for his guidance, support and encouragement during my study of Master of computer science at the Department of Computer Science and Software Engineering, International Islamic University, Islamabad, Pakistan. I am particularly indebted to him for teaching me the research and writing, which proved beneficial to my current research and future career. Without his support, efforts, knowledge, patience and answers to my many questions, this research would not have been possible. Experimental methods and results presented in this thesis have been influenced by him in one way or another. It was a great honor and pleasure for me to do research under his guidance and supervision. I would like to thank all the members of the Department of computer science and software engineering, who helped me by providing the necessary resources, and in various ways, in the completion of my work.

I thank to my father and especially to my wife for supporting me throughout my studies. Without their dedication, reliability, and I may not be able to follow my degree of MS(CS) from International Islamic University.

I want to dedicate this thesis to my sons Abdullah and Ahmad. In one word, they meant everything to me.

Abstract

Most of the image processing techniques do not perform well when images are corrupted with noise. Therefore restoring damaged images is considered as preprocessing step in most imaging applications. Detection and removal of impulse noise is an active area of research and we are proposing a novel technique which removes Random value impulse noise (RVIN) from digital gray scale images. The proposed method consists of two main modules. First module uses directional based statistics to detect noisy pixels from RVIN corrupted images. It calculates first and then second order difference in four main directions of 5x5 window. Second order difference is then used to produce a noise map based on a threshold. Detected noisy pixels are passed to the filtering scheme for estimation of their noise free values. Filtering scheme uses the edge statistics along the four main directions as well as background information to compute edge biased median for estimation of the non-noisy value of the pixel under consideration. Proposed technique has been compared using well known performance measure peak-signal-to-noise ratio. Simulations show that the proposed filter can provide excellent performance of suppressing impulse noise in all situations.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1

Introduction

We can define an image as a two dimensional function *f(x, y,,)* the spatial coordinates are defined as *x* and *y* and they have amplitude *f* at coordinates *(x, y)* which is known to be intensity of the points or we can say it is the gray level at those particular points. When these coordinates *x, y* and there amplitude *f* have finite value or discrete quantities than an image is known to be a digital image. Each digital image has a finite number of elements, having particular location. These elements are known an image elements, picture elements or pixels [1]. We can also say that an image is a two dimensional array, which can be manipulated through several techniques, like, mathematics, trend removal, convolution edge detection, filters, and image analysis. An image can be processed optically or digitally with a computer.

The entire image processing process is divided into three main stages as mention below:

1. ***Discretization:*** Visual information is converted into a discrete form, which can be used for further processing by computers.

2. ***Processing:*** Varying techniques are adopted to improve the quality of image know as filtering.

3. ***Analysis:*** Image features present in the image are extracted, like segmentation, recognition etc.

In the first stage, the input is an image taken by camera and the output will be a digital image. In the second stage, both the input and the output will be an image and the output will be an improved form of input image. A schematics diagram of different stages is shown in Figure 1.1. The figure is taken from the book specified in [2].

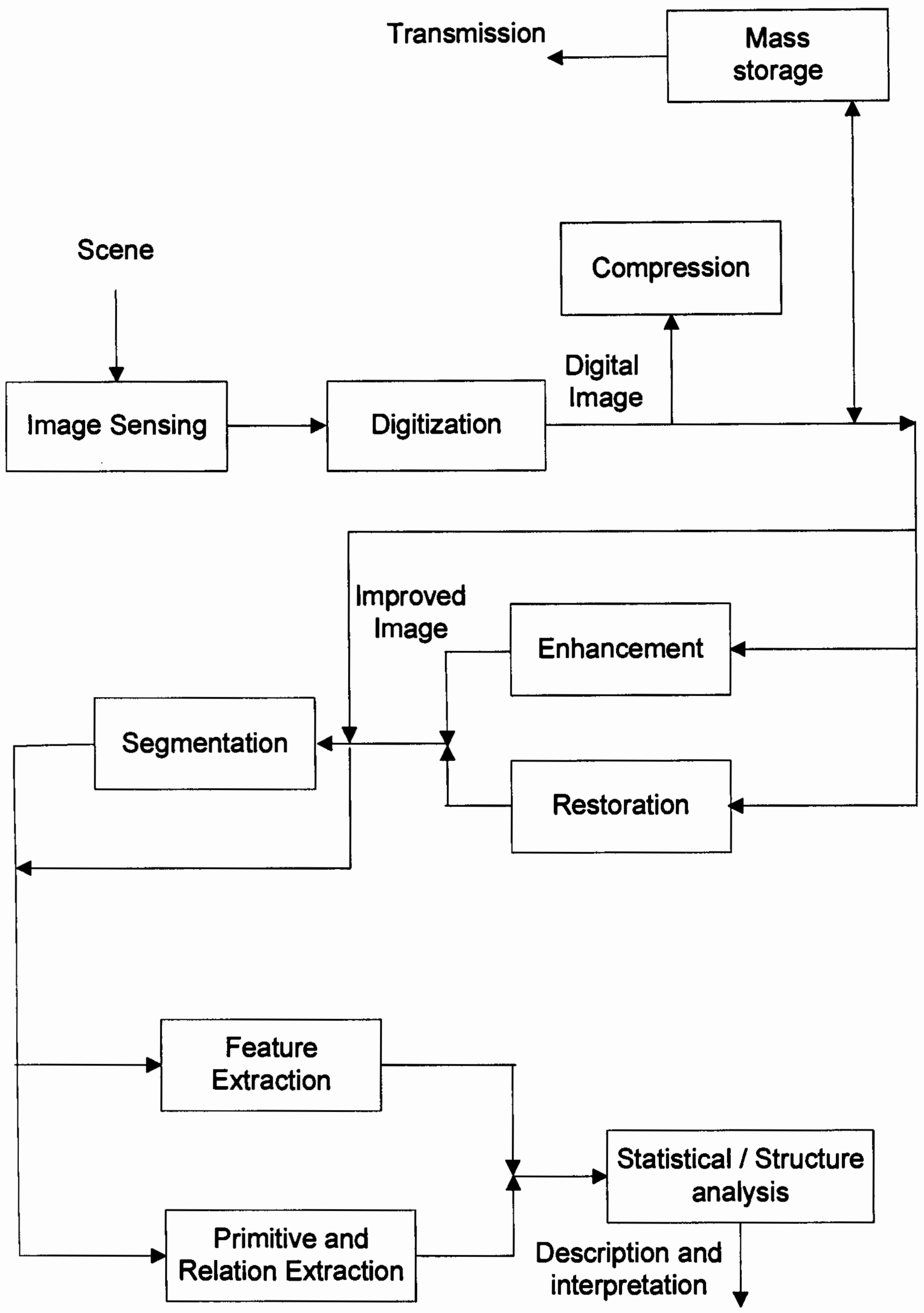


Figure 1.1 Image processing stages

1.1 Motivation

From the literature review and as it is mention in problem statement detecting and removing of Salt-n-pepper noise is easier as compared to Random value impulse noise. Most of the filters perform well when Salt-n-pepper is added to an image but when it comes to Random value impulse noise their performance decrease, both in detection as well as in removing the noise. The performance of any filter depends on the detection module, better the noise is detected more easily it is removed. The back bone behind detecting the noise in an image is threshold value used. The value of each pixel is compared with the threshold used and if any pixel value exceeds the threshold it will be considered as noisy pixel. The threshold value used in detection module is a constant value or it can be a set having four to five values in it. These threshold values are considered after conducting many experiments because these threshold value(s) can be optimal in one condition or environment or may not be able to perform well in different environment. Here by environment or condition means type of image, its characteristics and the ratio of noise added to it.

To boost the performance of detector module, a set of threshold is considered after conducting many experiments.

The facts which motivated us

- Work to improve the detection efficiency and to identify contaminated pixels.
- To devise a filter which identify sharp edges and preserve them.
- To decrease the computational complexity and to increase the computational efficiency of the filter.

1.2 Noise Models

Removing noise from digital images is very essential area for research. It is considered to be backbone process in image segmentation, analyses, pattern recognition etc. In digital images noise can be added during image acquisition (digitization) or it can be added during its transmission. Imaging sensors can be affected by ambient conditions, interference can be added to an image during transmission.

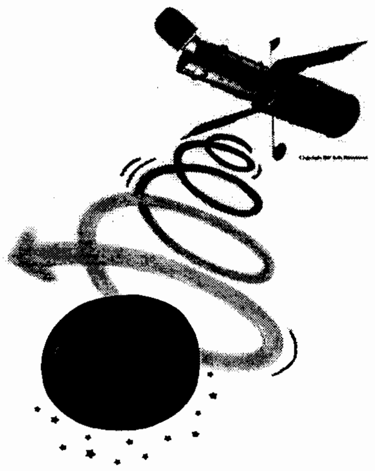


Figure 1.2: Image transmission from satellite

For example when an image is taken by a digital camera temperature and level of light are main factors which can produce noise in a digital image. Noise can also be added to a digital image during its transmission. May be damaged, for example, an image transmitted using a wireless network as a result of lightning or other disturbances in the atmosphere [1].

1.3 Image degradation model

The degradation process modeled is shown in the figure (1.3) below. It takes an input image *f(x,y)* and yields a degraded image known to be as *g(x,y)*. Now noise known to be *n(x,y)* is added along with *g(x,y)*. After adding noise it is passed to a restoration filter, the main objective of restoration filter is to remove the noise by estimating values for noisy pixels. These estimations are to be close as possible to the input image. The more we know about *H* and *n*, the closer $f'(x,y)$ to *f(x,y)*.

The whole process is shown in equation (1.1)

$$g(x,y) = h(x,y) \bigstar f(x,y) + n(x,y) \tag{1.1}$$

Where *h*(x,y) is the spatial representation of the degradation function and "★" is the convolution which is spatial domain in analogous to multiplication in the frequency domain[1].

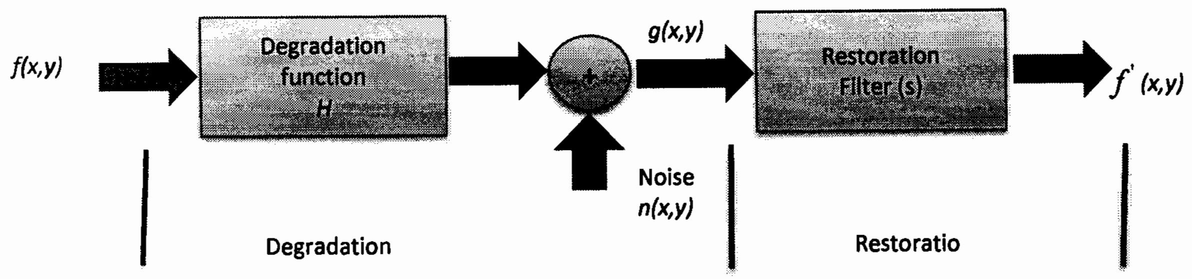


Figure 1.3: Image degradation model

1.4 Impulse Noise

In general terms impulse noise (IN) can be defined as intensity value of a single pixel, corrupted by any means and value of signal pixel can be dark or bright spots that are not authentic imagery. Impulse noise can be classified into two main categories namely as.

1. Salt and pepper noise (SPN).
2. Random value impulse noise (RVIN).

1.4.1 Salt & Pepper (SNP)

Salt & pepper noise is the form of noise in which image pixels are corrupted by black and white spots. These spots randomly appear in the image. If the image pixel is affected by salt noise, it will have 255 values which appear to be white and if the image pixel is corrupted by pepper noise it will have 0 values which appear to be black. Salt and pepper noise is shown in the equation below in equation 1.1.

$$x_{i,j} = \begin{cases} \text{O or 255} & \text{with probability } p \\ y_{i,j} & \text{with probability } 1-p \end{cases} \tag{1.1}$$

Where p represents total noise density in a single digital image. So the total distribution of salt & pepper noise in an image will be $p/2$. This means that the image will have 50% of salt noise which appears to be whit spots in the image and 50% pepper noise which appears to be black spots in an image. $x_{i,j}$ represents the noisy pixel and $y_{i,j}$ represents noise free pixel.

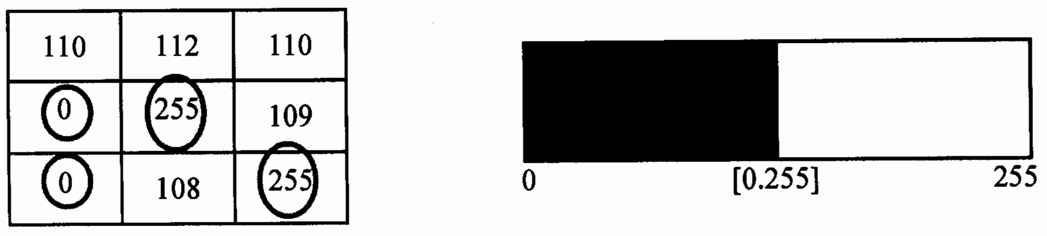
Salt & pepper noise can also have different noise densities $p1$ and $p2$ respectively than the equation will be something like this.

$$x_{i,j} = \begin{cases} 0 \text{ or } 255 & \text{with probability } p = (p1 + p2) \\ y_{i,j} & \text{with probability } 1 - (p1 + p2) \end{cases} \tag{1.2}$$

Salt & Pepper noise in digital images can be seen as black and white spots as shown in figure 1.4. Original image is shown in figure 1.4 (a) and 20% of salt and pepper noise is added into the image and shown in figure1.4(b). Whereas figure 1.4(c) show a 3 x 3 window of a digital image which show the value of nine pixels. Four out of nine pixels are corrupted with salt & pepper noise having o or 255 value. The corrupted pixels are circled. Two of the pixels are affected by Salt noise having 255 values and two of the pixels are corrupted by pepper noise having 0 values. The distribution of salt & pepper noise is shown in figure 1.4(d), which show the crisp discussion, either white or either black.

(a) Original Image

(b) 20% SNP included



(c) 3 x 3 win (d) SNP Color distribution

Figure 1.4: Example of Salt & pepper (SNP)

Salt & pepper noise is easier to detect because of its crisp values. It will either be 0 or it will be 255, where $x_{i,j} = S_{\min}$ or $S_{\max}$.

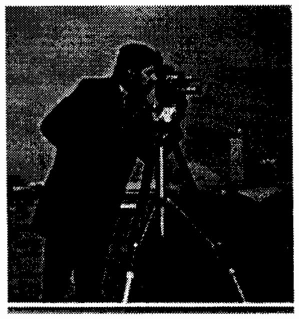
1.4.2 Random valued impulse noise (RVIN)

Random value impulse noise (RVIN) also known as uniform noise is a type of noise in which pixel value can be closer to the neighboring pixel value. RVIN is harder to detect and remove due to its characteristics.

$$x_{i,j} = \begin{cases} n_{i,j} & \text{with probability } p \\ y_{i,j} & \text{with probability } 1-p \end{cases} \tag{1.3}$$

Where $n_{i,j}$ is the gray level of noisy pixel. In SPN the pixel value can be 0 or 255 [$S_{\min}$ or $S_{\max}$] value which is easily to traceable. Where as in case of RVIN the pixels gets random values from 0 to 255, [$S_{\min}, S_{\max}$] which gets hard to identify, is it an original pixel or is it a noise in the image as shown in figure 1.5(c). In SPN the noisy pixel gets completely different values as compared to its neighboring pixels. Either it gets a large value '255' [$S_{\max}$] as compared to its neighboring pixel or it gets very small values '0' [$S_{\min}$] as compared to its neighboring pixel but in case of RVIN the noisy pixel can have minimum '0' [$S_{\min}$] or maximum '255' [$S_{\max}$] value or it can have same or closer value as compared to its neighboring pixels as shown in figure 1.5(c).

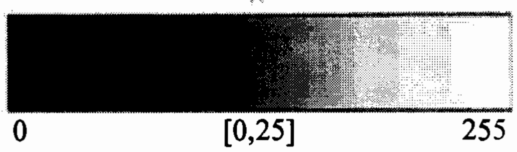
RVIN will have any value range from 0 to 255. It will have black and white spots along with gray spots in an image. Gray spots distribution will be different, it will be dark gray, medium gray, light gray, etc. Figure 1.5(a) shows the original image, in figure 1.5(b) 30% RVIN is added, figure 1.5(c) show a matrix of 3 x 3 win which has RVIN identified in circles and figure 1.5(d) shows the distribution of RVIN[2].


(a) Original Image

(b) 30 % RVIN included

118	116	115
20	250	111
119	122	255

(c) 3 x 3 win



(d) RVIN Color distribution

Figure 1.5: Example of Random value impluse noise (RVIN)

1.5 Similar Nearest Neighbor (SNN)

Similar nearest neighbor (SNN) pixels form different sizes of clusters of a digital image. For example cluster of 3 x 3 win, cluster of 5 x 5 win, cluster of 7 x 7 win and so on. In these cluster some pixels have same values and some have different values. Based on these values, image is considered as noisy image or noise free image. Normally in each cluster some pixels have same values and is considered as noisy free cluster as shown in figure 1.6(a). The reason is that any object in the image will have two or more pixels having same value. If in any cluster the values of pixels are not same than this cluster will be considered as noisy cluster as shown in figure 1.6(b). Another scenario is that in a cluster some pixel values will be very large or small from the rest of the pixels, this cluster will also be considered as noisy cluster but not all of the time but most of the time as shown in figure 1.6 (c). Sometimes an edge lies in a cluster and the values of this edge pixel will be quite different from the other pixels[3]. All of the three scenarios are shown in the figure 1.6.

120	115	115
121	117	117
119	116	115

(a) Noise free cluster

120	116	115
121	110	111
119	122	114

(b) Noisy cluster with all different values

118	116	115
5	**250**	115
118	122	**255**

(c) Noisy cluster with very large or small values

Figure 1.6: Similar Nearest Neighbor 3 x 3 windows

1.6 Problem domain

In this thesis Random value impulse noise (RVIN) is handled. As we have learned earlier that it is very hard to remove random value impulse noise because of its random values of pixels as compared to salt-and-pepper. In previous filters which are considered as state of art filters perform very well on salt-n-pepper but when comes to random value impulse noise there performance decreases. Especially when images have fine details, like boundaries and sharp edges. Median filter is considered as sate of art filter which provides ,mean value of a block to the noisy pixel. This strategy works very good when the image has smooth regions but when there are objects in an image it fail to estimate a good reasonable value for the noisy pixel, which result in image blur. There are many filters which detect boundaries and preserve them but again they perform well in salt-n-pepper noise. So we are proposing a novel method which will remove random value impulse noise and at the same time it will preserve the boundaries of an object present in an image.

1.7 Contribution

The novel method consists of two main modules. One is the detection of noise in the image and second part will remove the noise. The detection of noise plays an important part in removing the noise. It can't be wrong if we say that removal of noise depends on detection of noise. The better the noise pixels are detected the better they are filtered. This proposed directional based similar nearest neighbour filter (DBSNNF) detect the noise using first order derivative and second order derivative. The noisy image is sent to calculate first order derivative which keeps good track of thick edges in an image and then the output image is sent to calculate second order derivative which keeps good track of thin edges. After calculation of first order and second order a noise map is constructed showing which pixels is noisy and which one is noise free.

Now by considering the noise map the pixels having noise is evaluated a new value by passing it to the directional based similar nearest neighbour filter. The filter works in four directions and it also considered the background pixels, which can be helpful to identify the thick and thin edges and estimating a reasonable value. The decision made to consider which direction is based on standard deviation value of that direction. The direction having smaller standard deviation in a 5 x 5 window is considered to estimate a new value for the noisy pixel. In this way the whole image is divided into 5 x 5 block and each pixel is evaluated iteratively.

1.8 Thesis Layout

The layout of thesis is as chapter 2 consist of literature review, chapter 3 has problem statement, chapter 4 proposed technique to detection and removal of random value impulse noise, chapter 5 has simulation results, chapter 6 has conclusion and future works and in the last we have references.

Chapter 2

Literature review

Literature review
Spatial filtering
Order-Statistics filters
Adaptive filters
Median filters

Chapter 2

Literature Review

In digital images noise can be added during image acquisition (digitization)or it can be added during its transmission. Imaging sensors can be affected by ambient conditions, interference can be added to an image during transmission. So removing the noise from digital images is an emerging field in research. Much research has been done and many algorithms have been proposed to enhance the image details. Some of the algorithms detects the noise and produces a noise map and then repair the effected pixel and keeping other intact. Whereas some of the algorithms filters all the pixels irrespective of corruption.

There are many state of art filters that detect and remove Random value impulse noise (RVIN) from digital images. As discussed earlier RVIN is hard to detect and remove. Those algorithms which filters without detecting noise, use a window mask of size $(2N+1)^2$ where N > 0. In window mask the centre pixel is of interest as shown in figure (2.1) the centred pixel is circled.

1	1	1
1	0	1
1	1	1

3 x 3 window mask

1	1	1	1	1
1	1	1	1	1
1	1	0	1	1
1	1	1	1	1
1	1	1	1	1

5 x 5 window mask

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

7 x 7 window mask

Figure 2.1.Different window mask which run on image

The window mask will be moved from left top corner of a digital image. They work as overlapping mask. Disadvantage of this overlapping mask is that every pixel is treated irrespective to corruption.

Those algorithms in which detection is followed by filtering, firstly they notify noisy pixels and make a noise map and then only these affected pixels goes for filtering keeping other intact. In these algorithms filtering depends on detection of noisy pixels. Poor detection of noisy pixels results in poor filtering and vice versa.

The most well-known filters among all is median [4-5] filter. Median filter is considered to be the backbone filter in image enhancement, which perform excellent in smooth regions but it has poor performance when there is more detail in image like edges or boundaries.

2.1 Order-Statistics Filters

Order-statistics filters are well known non-linear spatial filter. These filter are based on ordering (ranking) of gray levels in the neighborhood which is defined by filter mask. The best known filter of this type is *median filter.*

2.1.1 Median Filter [4-5]

Median Filter is a state of art filter in the category of order-statistics filter. It has excellent noise reduction capabilities specially in uni-polar and bi-polar impulse noise. It replaces the corrupted value of a pixel by taking the median of the neighboring pixels and can be represented as

$$\hat{f}(x,y) = Median_{(s,t)\epsilon S_{xy}}\{g(s,t)\} \tag{2.1}$$

Where $\hat{f}(x,y)$ and $g(s,t)$ are restored and noisy images respectively and S_{xy} represent the set of coordinates in a window of size $m \times n$.

2.1.2 Weighted median filter [6]

This filter allows you to filter the corrupted image by the weighted non-linear method known to be median filter. This filter works for monochrome, 8-bit pixel and 24 bits per pixel images. Weighted median filter is different from the median filter that specified pixels in a local neighborhood is given more weight and is repeated a given number of times for calculation of a median value. It is defined as

Weighted Median (A) = Median [Repeat StrMask (i,j) times {A (x + i , y + j)}]

Where (x, y) coordinate of image A and the coordinate (i, j) is defined over the structuring mask of StrFunc structuring function.

2.1.3 Center Weighted Median Filter [7]

This filter gives more weight to a central value or pixel of each window. The Image details are preserved while suppressing impulsive type noise or additive noise.

2.1.4 Max and Min filter [8]

Max filter is very useful in finding the brightest points and remove pepper noise while the Min-filter helps you find the darkest points and removing salt noise in an image. This filter works very good in salt-and-pepper noise.

2.1.5 Adaptive Median Filter [9]

This filter can handle impulse noise with larger probabilities and preserves the details while smoothing non-impulse noise. It has three main purposes:

1. Removes *salt andpepper* noise
2. Smoothing for non-impulse noise
3. Reduce distortion (Thing or thickening of object boundaries)

Many state of art filter are proposed to remove random-valued impulse noise from digital images. Some of them well know filters are discussed below.

2.2 Tri-State Median Filter [10]

Tri state filter was proposed to preserve image details. It is based on two states of art filter known as standard median filter (SMF) and the Center weighted median filter (CWMF). Firstly noise detection is incorporated by an impulse detector. The filter first detects the noisy pixels in the image and then sends to filter those pixels to Tri state filter keeping other intact. Impulse detector is used to realized noise in an image, which takes the outputs from the CWM and SM filters, after that it compares them with the center pixel value to make a tri-state decision. Switching logic depends on threshold values. The output of the Tri-state filter is obtained by the equation 2.2 shown below.

$$Y_{i,j}^{TSM} = \begin{pmatrix} x_{i,j} & T > \bar{d}_1 \\ y_{i,i}^{CWM} & d_2 \leq T < d_1 \\ y_{i,j}^{SM} & T < d_2 \end{pmatrix} \tag{2.2}$$

Where $y_{i,i}^{CWM}$ and $y_{i,j}^{SM}$ are the outputs that are received from CWM and SM filters.

2.3 Signal-Dependent Rank Order Mean Filter (SDROM) [11]

This filter is a non-linear algorithm used for the detection of impulse noise and used to preserve image details. The method is applicable to all impulse noise models, such as salt-and-pepper noise know to be fixed value and random value impulse noise know to be having dynamic range. The algorithm is based on a screening assessment strategy. If a pixel is detected to have a noise, a fair value based on neighborhood information is replaced, otherwise the pixel remain unchanged. This filter achieves excellent technical exchanges between suppression noise and preserving details and edges without unnecessary increase in computational complexity. SD-ROM filter works in a 3 x 3 window. The noisy pixel is kept in the center and remaining eight pixels known to be the neighboring pixels are used to estimate the value for the centered corrupted pixel based on four sets of thresholds. T_1, T_2, T_3, T_4. Where each threshold is less than the other likely, $T_1 < T_2 < T_3 < T_4$. The estimated value for the corrupted pixel is based on the estimation of true value, which depends on the neighboring pixels. The performance of SD-ROM can be increased by adopting recursion.

2.4 Adaptive Impulse Detection Using Center Weighted Median Filter [12]

It invents new operator adjustment, which is based on estimation values between the current output pixel and outputs the cog average filters with variable center weights. Employs changing system based on detection mechanisms trigger. Using center-weighted median filter uses the weight is changed to a more general operator, which filters out specific outputs and anxiety Cwm differences between the current pixels using a defined pulse, creates. Median and the final production of the current pixel are off.

2.5 Advanced Impulse Detection Based on Pixel-Wise MAD (PWMAD) [13]

It is a powerful estimator of the variance, the (average absolute deviation from mean) MAD, modified and be used effectively to distinguish noise pixels from image data. The algorithm is very different standards, requires no training or customization, and successfully removes all types of impulse noise. Pixel-wise MAD concept is simple and low in complexity. MAD mean absolute deviation of the image data is used to estimate the presence of the separation of noisy pixels to detect. Median of the MAD (PWMAD) ensures reliable iterative pixel-wise changes.

2.6 Directional Weighted Median Filter for RVIN [14]

Another method for removing impulse noise directional random Rated weighted average filter (DWM) is. This filter uses a pulse detector, the current pixel and its neighbors based on the difference between coalitions of the four directions. First, the idea of a 5 x 5 window. Horizontal, vertical and two diagonals: Consider now the four directions. Each pixel is 5 units. In each direction, and then calculates the weighted difference takes a minimum. Phase filtering, calculates the standard deviation of the four directions. Description of the standard deviation around the average value of how well the total pixel values of four pixels to another is aligned with the direction. Therefore, the central value is coming. Calculate weighted average, where the direction in which direction the standard deviation is small, with an average price at the extra weight of noise pixels. It is an iterative method. This process was repeated 8 to 10 times. The new directional weighted median filter for removing impulse noise random value gives better performance when the noise level is very high.

2.7 Fuzzy Impulse Noise Reduction Methods for Color Images [15]

The reduction or elimination of noise in a color image is an essential part imaging as the definitive information for human perception or a self-testing and analysis. Besides all the classic filters based on noise reduction, have inspired many fuzzy filters have been developed over the last years. However, it is very difficult to improve the quality of the various filters. ? Who are these forms of noise, how in relation to each other, there are some filters that clearly surpass the others Clears numerical results with the visual effects; This paper answers these questions for color images corrupted by impulse noise.

2.8 Histogram-Based Fuzzy Filter for Image Restoration [16]

This article present a new approach for restoring noisy images, highly impulsive noise is particularly effective in removing while preserving image detail. It is a member of fuzzy smoothing filter for a series of works produced by the basic parameters of the histogram entry is achieved by making. The principle of maintaining a dynamic histogram integrated statistical inputs to the original parameters to adjust. A reference intensity and defuzzification of the production process to reduce the deviation. The median filter (MF), the proposed filter has the advantage that it is simple and does not assume any prior knowledge of the input data, but it is traditional for the full range of possible impulse noise filter (MF) including more than display appears. Unlike many neuron-fuzzy or neuron-fuzzy filter the random strategy, the long-time education, proposed to be used for the initial membership fees selecting functions.

2.9 Minimum-maximum exclusive mean filter (MMEM) [17]

Proposed MMEM filter works in a window size n x n. The damaged pixels kept center. The proposed filter is applied to the original noisy image pixel to estimate a gray value for the damaged pixel value is replaced. In other words, for each pixel find the maximum and minimum values of gray window a n x n. Through all the value whose pixels is equal minimum activity and maximum gray level in a window n x n. If all pixels are rejected then return to pick a new window and do all the steps of what calculate the average (original) gray value of pixels refused, and call this average value AVG. Use the average production filtered values of the four neighboring pixels as the average is greater than the threshold otherwise keep the original pixel intact.

2.10 Adaptive nearest neighborhood filter (ANNF) [18]

Adaptive nearest neighbor filter detects and filter salt-and-pepper noise only. Its performance is remarks able in SPN but when comes to resolve Random value impulse noise it fails to perform well. The details on ANNF is mention below

2.10.1 Noise Detection

Non overlapping blocks are of size 3 x 3 is made and mean of each block is calculated $\bar{x}$. Than first order derivative is calculated by subtracting every pixel of 3 x 3 windows from mean as shown in figure 2.2.

$$\underset{X}{\begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix}} \xrightarrow{\bar{x}-x_i} \underset{Dx}{\begin{pmatrix} dx_1 & dx_2 & dx_3 \\ dx_4 & dx_5 & dx_6 \\ dx_7 & dx_8 & dx_9 \end{pmatrix}} \xrightarrow{d(dx_i)} \underset{D^2x}{\begin{pmatrix} d^2x_1 & d^2x_2 & d^2x_3 \\ d^2x_4 & d^2x_5 & d^2x_6 \\ d^2x_7 & d^2x_8 & d^2x_9 \end{pmatrix}} \xrightarrow{Th_i} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

(a) First Order, Second Order & Noise Map Construction

$$\begin{aligned} d^2x_1 &= dx_1 - dx_2 \\ d^2x_2 &= dx_1 - dx_3 \\ d^2x_3 &= dx_2 - dx_3 \end{aligned}$$

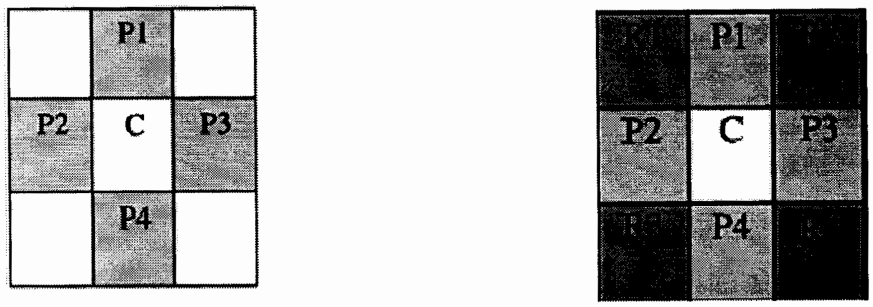
(b) Second order derivative calculation

Figure 2.2: Noise detection Process

Now Second order derivative is calculated from first order derivative, now based on threshold a noise map is constructed. The detail of the whole process is shows in the figure 2.2(a). Figure 2.2(b) shows the process of calculation of second order derivative. The noise map having '1' means the pixel value is corrupted and it is sent to Adaptive nearest neighborhood filter (ANNF) to estimate a value for the corrupted pixel. The noise map having '0' means the pixel value is not corrupted so it is left as it is.

2.10.2 Noise Removal

Adaptive nearest neighbor filter replace the contaminated pixel value by its neighboring pixel. It has two strategies in replacing the value of contaminated pixel. Firstly it will replace the contaminated pixels 'C' with P1 or P2 or P3 or P4 as shown in figure 2.3 (a). This filter is known as Adaptive nearest neighbor filter (ANNF). If all the pixels P1, P2, P3, P4 are corrupted than it will look into the pixels which are located in diagonal locations R1, R2, R3, R4 as shown in figure 2.3 (b). This method is known as Modified adaptive nearest neighborhood filter (MANNF).



(a) ANNF (b) MANNF

Figure 2.3 : Adaptive nearest neighbor filter

If these pixels R1, R2, R3, R4 are also corrupted than this pixel 'C' is left as it is and filter moves to next 3 x 3 block and filters the contaminated pixels. In this iterative and repeated way it will remove the contaminated pixels.

TH-9415

2.10.3 Limitations

I. Only removes SPN and does not consider RVIN which is difficult to detect and remove

II. Does not preserve boundary details well because of not using any directional statistics in noise detection process

2.11 Similar neighbor criterion [19]

A similar neighbor criterion filter detects random value impulse noise and removes it by comparing it with the certain thresholds values.

2.11.1 Noise Detection

The noise detection process works in different size windows 3 x 3, 5 x 5 and 7 x 7 respectively. Each window is passed through three different phases which are as under.

Phase I: If a block have similar pixels than noise-free pixel block as shown in figure 2.4 (a).
Phase II: If a block have no similar pixels than noisy pixel block as shown in figure 2.4 (b).
Phase III: If a block have very large or small values as compared to other pixels than noisy pixel

block. The pixels which are circled are considered to be noisy pixels as shown in figure 2.4 (c).

120	115	115
121	117	117
119	116	115

(a) Phase I

120	116	115
121	110	111
119	122	114

(b) Phase II

120	116	115
121	250	111
119	122	9

(c) Phase III

Figure 2.4: Noise Detection of similar neighbor criteria

2.11.2 Noise Removal

The restoration process works in 3 x 3, 5 x5, 7 x 7 windows. In order to restore the corrupted image replace each detected noisy pixel that was flagged as 0 in noise map with a normalized weighted sum is taken of good neighboring pixels in the filtering window of 5 x 5 size keeping other pixels intact.

2.11.3 Limitations

I. Does not preserve image details well because each pixel is considered as an original pixel and it has to satisfy the threshold(s), which result in poor detection specially in RVIN.

II. Does not preserve boundary details well because of not using any directional statics in noise detection process.

2.12 Multi-stage Directional Median Filter (MDM) [20]

Multi-stage directional median filter removes the noise using four different directions in a 5 x 5 window.

2.12.1 Noise Removal

This filter removes the noise by keeping the corrupted pixel in the center and estimate a value by getting the median value of all the four directions. The direction having smallest median values is considered and replaced as shown in figure 2.5.

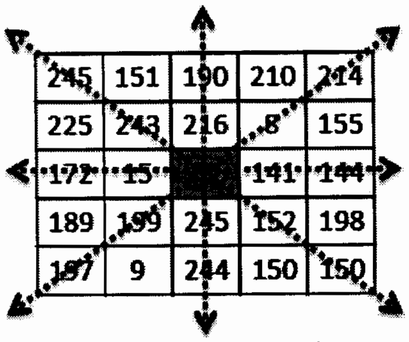


Figure 2.5: Four directions

2.12.2 Limitations

I. Only removes SPN and does not consider RVIN which is difficult to detect and remove.

II. Restore thin directions and neglect thick directions, does not use directional statistics in proper manner.

2.13 Summary

In this chapter, basics of spatial filtering, smoothing filters, order statistics filters and adaptive filters are discussed. Smoothing or average filters are used in pre-processing steps for blurring and noise reduction. Order statistics filters are based on ordering of gray levels in the neighborhood defined by filter mask. Adaptive filters are much better in performance as compared to smoothing and order-statistics filters. This chapter also describes in detail about the related work in the form of literature review.

Chapter 3

Problem statement

Chapter 3

Problem statement

As we know that Random Value Impulse Noise (RVIN) is hard to detect and remove as compared to Salt-and-pepper (SPN). In SPN the pixel value can be 0 or 255 [$S_{\min}$ or $S_{\max}$] value which is easily to traceable. Where as in case of RVIN the pixels gets random values from 0 to 255, [$S_{\min}$, $S_{\max}$] which gets hard to identify, is it an original pixel or is it a noise in the image. In SPN the noisy pixel gets completely different values as compared to its neighboring pixels. Either it gets a large value '255' [$S_{\max}$] as compared to its neighboring pixel or it gets very small values '0' [$S_{\min}$] as compared to its neighboring pixel but in case of RVIN the noisy pixel can have minimum '0' [$S_{\min}$] or maximum '255' [$S_{\max}$] value or it can have same or closer value as compared to its neighboring pixels. RVIN will have any value range from 0 to 255. It will have black and white spots along with gray spots in an image. Gray spots distribution will be different, it will be dark gray, medium gray, light gray, etc. The difference between RVIN and SPN can be best described by the figure 3.1

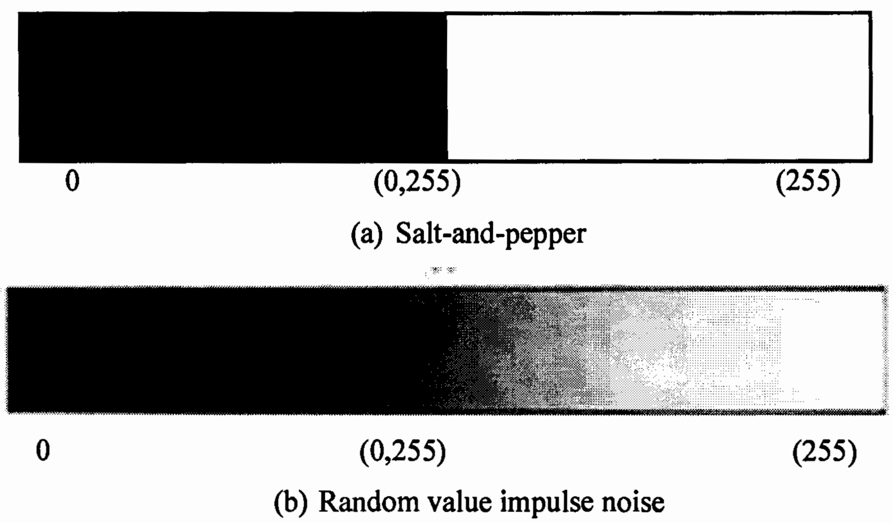


Figure 3.1: Comparison of RVIN & SPN

- ❑ **Problem 1:** Somasundram et.al's[12] only removes SPN and does not consider RVIN which is difficult to detect and remove as shown in figure 3.2(b)

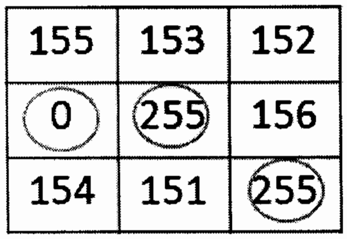
- ❑ **Problem 2:** Said et. al's [13] does not preserve image details well because each pixel is considered as an original pixel and it has to satisfy the threshold(s) and if it fail to satisfy the threshold, which result in poor detection specially in RVIN

- ❑ **Problem 3:** Both [12-13] technique does not preserve boundary details because of not using any directional statistics in noise detection process as shown in figure 3.2

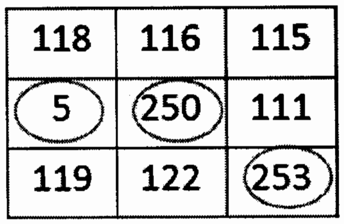
- ❑ **Problem 4:** Zong and Le [14] detects only thin edges, but when comes to thick edges it does not retain them.

- ❑ **Problem 5**: The filters discussed above [12, 14] removes salt-n-pepper noise from digital images they do not entertain random value impulse noise. Whereas Said et. al's [13] handles RVIN but without any directional statistics.

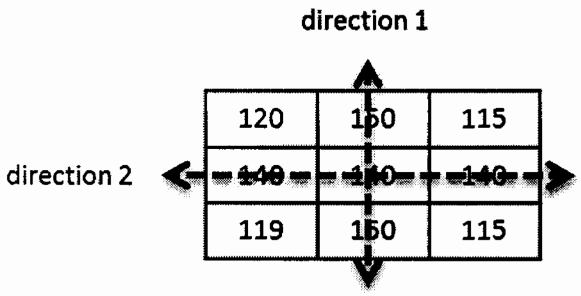
The problems discussed above handles salt-n-pepper very well except one (Problem 2) but when comes to random value impulse noise these filters do not detect noise very well and fail to preserve images having fine details. The reason is that they do not use any directional statistics in noise detection process or in noise removal. They just use the weighted median value of the neighboring pixels. The median filter performs well as long as the spatial density of the impulse noise is not large. This strategy works very well when images have smooth regions but when comes fine details it don't preserve boundaries of an object in an image especially when we are taking about thin edges in an image. The last filter problem 5 do handles thin edges but when we are taking about thick edges it do not handle thick edges and also remove salt-n-pepper noise. It do not consider random value impulse noise.



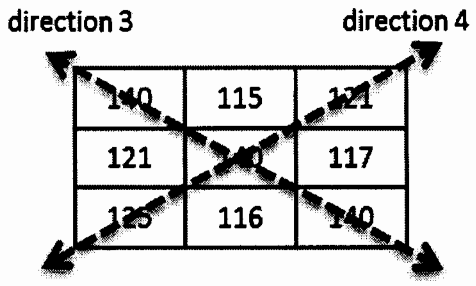
(a) Salt & Pepper Noise



(b) Random value impulse noise



(c) Horizontal & vertical direction



(d) Diagonal directions

Figure 3.2: Problem statement

Chapter 4

Proposed solution

Chapter 4

Proposed solution

4.1 Efficient Impulsive noise detection scheme

The main challenge in the removal of impulse noise is to suppress the noise and to preserve the details of the image like edges, objects, boundaries etc. Median filter for removing impulse noise is used as a base. A pulse detector with several filters are proposed to remove impulse noise, and some of them are described in chapter 2. We proposed a novel method for detection and removal of random value impulse noise from images. The scheme works in two phases, namely a novel detection method of contaminated pixels followed by filtering these contaminated pixels keeping other intact. The scheme works in two phases, namely a novel detection of contamination pixels followed by filtration of the contaminated pixels and keeping others intact. The detection scheme uses first-order difference followed by second-order difference of pixels in a 5 x 5 test window to detect noise and build a noise map. The filtering scheme is a median filter making a decision in choosing a direction using standard deviation to preserve edges and image details, this filter also runs in 5 x 5 windows.

4.1.1 Sharpening Spatial Filters

The main purpose of the loops on the fine details of the image or highlight to detail is to improve blurred, either accidentally or as a natural consequence of a particular method of acquiring of the images. Uses of image enhancement vary and include applications ranging from medical imaging to industrial autonomous inspection and to electronic printing and guidance military systems. Image blur could be done in the spatial domain by averaging the pixels in a neighborhood. Since the average is analogous integration, it is logical to conclude that the approach could be achieved by spatial differentiation. This indeed is the case, and discussion in this section discusses an efficient way to define and apply the operators to sharpening of digital differentiation. Fundamentally, the strength of the response of an operator derivative is proportional to the derivative operator tearing at the point at which the operator is applied. Thus, by taking image differential enhances discontinuities (such as noise) and the edges keeping less emphasis on areas with slowly varying gray level values.

4.1.2 First & Second order derivative

The behavior of first order & second order derivatives in areas of constant gray level (flat segments) is of interest at the beginning and end of discontinuities along gray level ramps. The derivatives of a digital function are defined in terms of differences.

A first order derivative function for one dimension is defined in equation (4.1).

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{4.1}$$

A partial derivative is used here to keep the same notation as when considering an image function of two variables $f(x,y)$.
Similarly, we define a second-order derivative as the difference in equation (4.2)

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \tag{4.2}$$

Where $f(x)$ is a pixel, $f(x+1)$ is the next pixel to it and $f(x-1)$ is the previous pixel to it.
Figure 4.1 shows the importance of first order and second order derivatives. Both of the derivatives are used to detect edges in flat regions and discrimination of noise from images. If we compare first order and second order derivatives we get to know that

- First-order derivative is generally used to detect thick edges present in an image.
- Second order derivatives have greater response to images having fine details, like image having fine lines and isolated points present.
- First order derivatives shows high response to gray level present in an image.
- Second derivatives produce shows higher response to gradual changes in gray level.

For similar changes present in a gray level, the response of second order derivatives in very strong to a line than to a step and to a point than to a line. So we will be using both first order derivative as well as second order derivative to detect noise and edges present in an image. Figure 4.1 (a) shows a single image that contains multiple solid objects, a line and a point sound. Figure 4.1 (b) shows a gray level horizontal profile (scanning line) of the image along the center and that includes noise point. This profile is the function of a dimension to be used for illustrations regarding the figure.

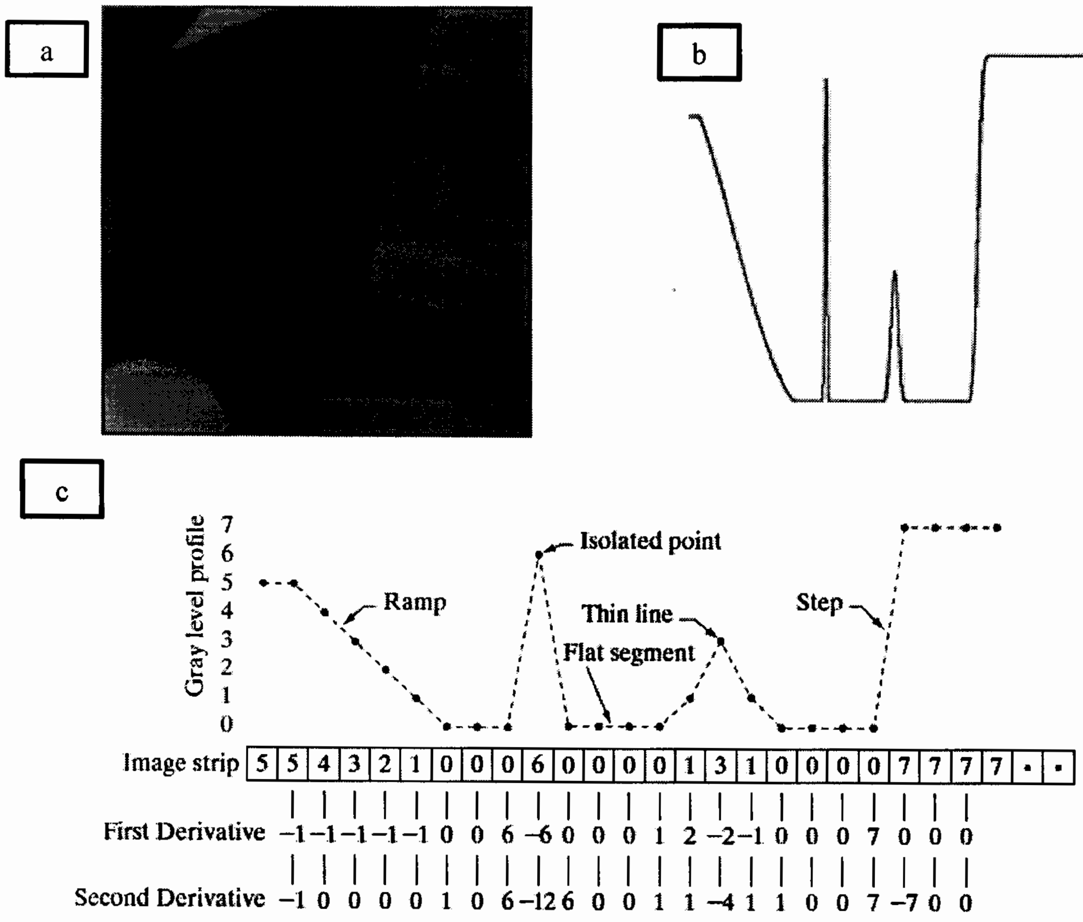

a
b
c
Gray level profile
7 6 5 4 3 2 1 0
Isolated point
Ramp
Thin line
Flat segment
Step
Image strip 5 5 4 3 2 1 0 0 0 6 0 0 0 0 1 3 1 0 0 0 0 7 7 7 7 • •
First Derivative −1 −1 −1 −1 −1 0 0 6 −6 0 0 0 1 2 −2 −1 0 0 0 7 0 0 0
Second Derivative −1 0 0 0 0 1 0 6 −12 6 0 0 1 1 −4 1 1 0 0 7 −7 0 0

Figure 4.1 (a) True image. (b) Isolated noise point along with 1-D horizontal gray level profile. (c) Calculation of FOD & SOD.

Figure 4.1 (c) shows a simplified profile, with enough members or numbers to make it possible for us to analyze how first order and second order derivatives behave as found with a point noise, a line, or the edge of an object. In our simplified diagram of the transition ramp spans four pixels, the noise is about one pixel, the line is three pixels thick, and the transition to the gray level crossing takes place between adjacent pixels. The number of gray levels was simplified to only eight. Considering the properties of first order derivative, is that it is not zero over the entire ramp, while the second-order derivative is nonzero only at the beginning and end of the ramp as shown in figure 4.1(c). Because the edges of an image resembles the same type of transition, we conclude that first order derivatives is to produce "thick" edges and second order derivatives, produces much finer thin edges[1].

4.2 Proposed algorithm

The proposed technique directional based similar nearest neighbor filter (DBSNNF) is divided into two main modules. (I) Noise detection module, (II) Noise removal module. Both of the parts play an important role in removing the noise. Failure of any one module will result in blur image having noise in it. Proposed algorithm is shown below.

DBSNNF Algorithm:

```
[1] Input image.
[2] Noise detection process.
    (2.1) First order derivative.
    (2.2) Second order derivative.
[3] Noise map construction.
[4] Noise removal process.
    (4.1) if noisy pixel found than
    (4.2) Passed the noisy pixel to directional based similar nearest neighbor filter.
    (4.3) else juts keep the pixel value intact.
[5] Final restored image.
```

4.3 Proposed architecture

Architecture of proposed algorithm is shown in figure (4.2) which shows the flow of the algorithm. A noisy image is taken in as a input image, it is then then passed to noise detection process which yields a noise map. Now considering the noise map, each pixel is checked one by one, if the pixel is noisy it is passed to the directional based similar nearest neighbor filter which estimate a new value for that pixel and if it is noise free than original value of pixel is grabbed and a new image is restored as shown in figure (4.2).

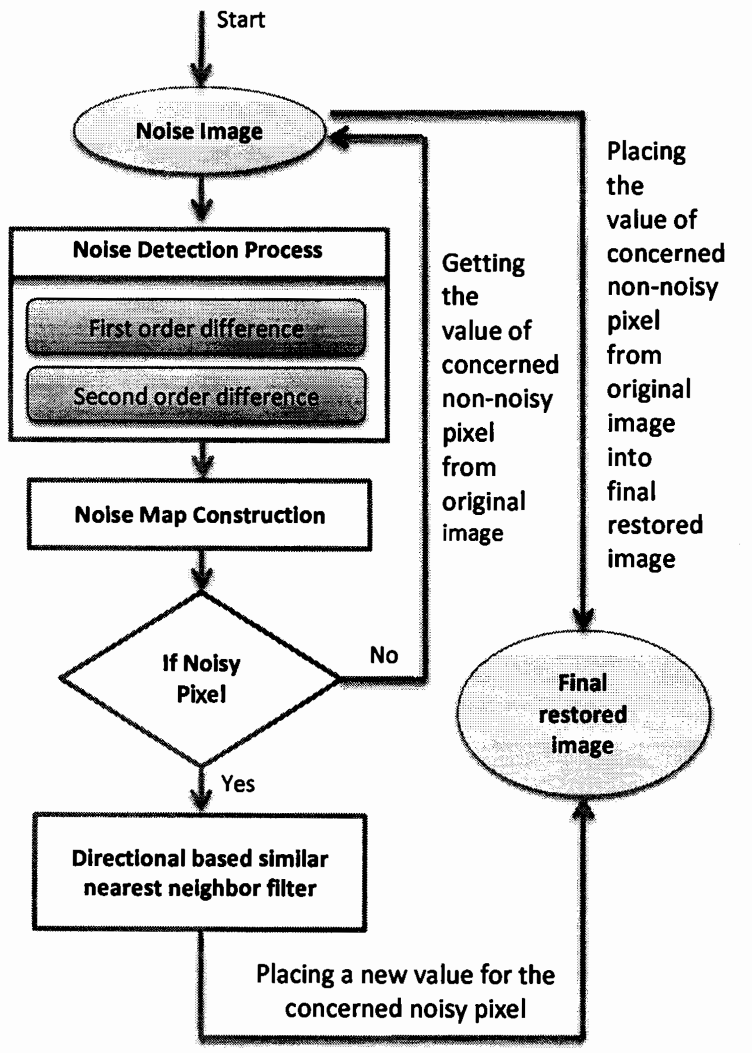

Start
Noise Image
Noise Detection Process
First order difference
Second order difference
Noise Map Construction
If Noisy Pixel
No
Yes
Directional based similar nearest neighbor filter
Getting the value of concerned non-noisy pixel from original image
Placing the value of concerned non-noisy pixel from original image into final restored image
Final restored image
Placing a new value for the concerned noisy pixel

Figure 4.2
Proposed solution architecture

Seventeen pixels are considered in a 5 x5 window for four directions, whereas the pixels that lies in these four directions are

- $d_1 = \{P1, P2, P7, P8\}$
- $d_2 = \{P3, P4, P5, P6\}$
- $d_3 = \{D1, D3, D6, D8\}$
- $d_4 = \{D2, D4, D5, D7\}$

Where '*C*' is the pixel whose value is to be evaluated and it is kept in center. All the twenty five pixels in a 5x 5 window are evaluated one by one for noise.

The equations for the above four directions (d_1, d_2, d_3, d_4) can be derived by getting impressed from Laplacian operator, which is a very good detector in detecting sharp edges when pixel gray levels change rapidly. But when there is a slow change in gray levels from dark to bright, than gradient operation produce a very wide edge. Laplacian operator is based on four pixels values as shown in figure 4.4

$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$
$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$
$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$

Figure 4.4: Laplacian operator

Laplacian operator is shown in equation 4.3.

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{4.3}$$

Where as in case of proposed DBSNNF algorithm four pixels are considered in one direction and the fifth pixel is the centered pixel. The equations for each direction is shown in equation 4.4

$$\left.\begin{aligned}
d_1 &= \{| f(x-2,y)+f(x-1,y)+f(x+1,y)+f(x+2,y)-4f(x,y)|\} && (4.4.1)\\
d_2 &= \{| f(x,y-2)+f(x,y-1)+f(x,y+1)+f(x,y+2)-4f(x,y)|\} && (4.4.2)\\
d_3 &= \{| f(x-2,y-2)+f(x-1,y-1)+f(x+1,y+1)+f(x+2,y+2)-4f(x,y)|\} && (4.4.3)\\
d_4 &= \{| f(x-2,y+2)+f(x-1,y+1)+f(x+1,y-1)+f(x+2,y-2)-4f(x,y)|\} && (4.4.4)
\end{aligned}\right\} \quad (4.4)$$

These four equations are derived from figure 4.5. Thus impulse detection can be represented using the minimum value of these four directions. The equation for impulse detection is shown as.

$$d = \min\{d_k : 1 \le K \le 4\} \tag{4.5}$$

After getting the value of *d* which is minimum among four directions, a decision is made to construct a noise map by comparing with a threshold *T*.

1. If value of d is small enough than test pixel $f(x,y)$ is a noise-free pixel or we can say it's a flat region pixel as all the four direction differences are small. So zero (0) will be placed for the concerned pixel.

2. A test pixel $f(x,y)$ when falls on an edge will have smallest second order derivative which will satisfy the threshold. So the test pixel $f(x,y)$ will be considered noise free pixel. So zero (0) will be placed again at the place of concerned pixel.

3. A large value of d implies that the test pixel $f(x,y)$ is noisy as its directions yields a

large value. So one (1) will be placed for the concerned pixel.

All the concerned four directions are shown in the figure 4.5. This will produce a Noise map which will identify noisy pixel and noise free pixel.

$f(x-2,y-2)$	$f(x-2,y-1)$	$f(x-2,y)$	$f(x-2,y+1)$	$f(x-2,y+2)$
$f(x-1,y-2)$	$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$	$f(x-1,y+2)$
$f(x,y-2)$	$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$	$f(x,y+2)$
$f(x+1,y-2)$	$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$	$f(x+1,y+2)$
$f(x+2,y-2)$	$f(x+2,y-1)$	$f(x+2,y)$	$f(x+2,y+1)$	$f(x+2,y+2)$

(a) Direction 1, vertical pixels, (represents equation 4.4.1)

$f(x-2,y-2)$	$f(x-2,y-1)$	$f(x-2,y)$	$f(x-2,y+1)$	$f(x-2,y+2)$
$f(x-1,y-2)$	$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$	$f(x-1,y+2)$
$f(x,y-2)$	$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$	$f(x,y+2)$
$f(x+1,y-2)$	$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$	$f(x+1,y+2)$
$f(x+2,y-2)$	$f(x+2,y-1)$	$f(x+2,y)$	$f(x+2,y+1)$	$f(x+2,y+2)$

(b) Direction 2, horizontal pixels, (represents equation 4.4.2)

$f(x-2,y-2)$	$f(x-2,y-1)$	$f(x-2,y)$	$f(x-2,y+1)$	$f(x-2,y+2)$
$f(x-1,y-2)$	$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$	$f(x-1,y+2)$
$f(x,y-2)$	$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$	$f(x,y+2)$
$f(x+1,y-2)$	$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$	$f(x+1,y+2)$
$f(x+2,y-2)$	$f(x+2,y-1)$	$f(x+2,y)$	$f(x+2,y+1)$	$f(x+2,y+2)$

(c) Direction 3, diagonal pixels, (represents equation 4.4.3)

$f(x-2,y-2)$	$f(x-2,y-1)$	$f(x-2,y)$	$f(x-2,y+1)$	$f(x-2,y+2)$
$f(x-1,y-2)$	$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$	$f(x-1,y+2)$
$f(x,y-2)$	$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$	$f(x,y+2)$
$f(x+1,y-2)$	$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$	$f(x+1,y+2)$
$f(x+2,y-2)$	$f(x+2,y-1)$	$f(x+2,y)$	$f(x+2,y+1)$	$f(x+2,y+2)$

(d) Direction 4, diagonal pixels, (represents equation 4.4.4)

Figure 4.5: Four directions for calculation of SOD

4.5 Efficient Impulsive noise removal scheme

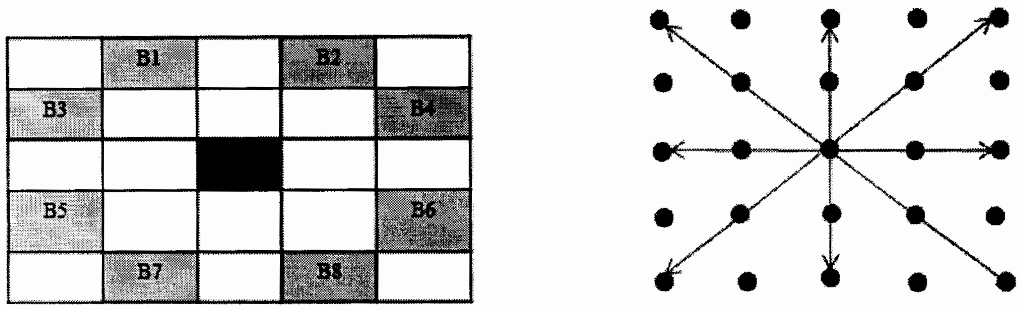
4.5.1 Noise Correction

To remove the noise, each pixel is checked, if it has noise means 1 (one) than it will be passed to Directional based similar nearest neighbor filter to remove the noise and if it does not have noise means zero (0) than the value of concerned pixel from the noisy image (original image) will be taken as it is and placed at the concerned location to make a final restored image.

4.6 DBSNNF noise removal algorithm

```
[1] If pixel is not noisy then
    (1.1) Just place that pixel value from input image at concerned place in restored image.
[2] else if pixel is noisy then
    (2.1) Make a 5 x5 window on input image.
    (2.2) Place the processing pixel at the center of 5 x 5 window.
    (2.3) Get the pixel values from four directions from input image (excluding center pixel).
    (2.4) Calculate median of four directions. ($MedianDir_i$)
    (2.5) Get the values of eight background pixels left in 5 x 5 window from input image.
    (2.6) Calculate median of eight background pixels. ($MedianBGP$)
    (2.7) Calculate standard deviation of four directions.
    (2.8) if any two or more standard deviations are equal then
        (2.8.1) Calculate difference (there exist 2 or more directions)
                $diff_i = MedianBGP - MedianDir_i$
        (2.8.2) if any two differences are equal then
                (2.8.2.1) There exist two edges.
                (2.8.2.2) Get the median value of that direction and place in the centered pixel.
    (2.9) else if any directions standard deviation equal to zero.
        (2.9.1) Its possible there exists a single edge.
        (2.9.2) Get the median value of that direction and place in the centered pixel.
    (2.10) else if no standard deviation equal to zero means no edge exist
        (2.10.1) Find direction with smallest standard deviation.
        (2.10.2) Combine that direction vertically to make a 5 x 6 window.
        (2.10.3) Calculate median of 5 x 6 window excluding the processing pixel.
        (2.10.4) Place the median value of 5 x 6 window in the effected pixel.
```

Where *MedianBGP* is the median value of eight background pixels as shown in figure 4.6(a)(b), $MedianDir_i$ is the median value of four directions, $diff_i$ is the difference calculated and i = *1,2,3,4.* The background pixels are shown in figure 4.6(a)(b).



(a) (b)

Figure 4.6: Eight background pixels

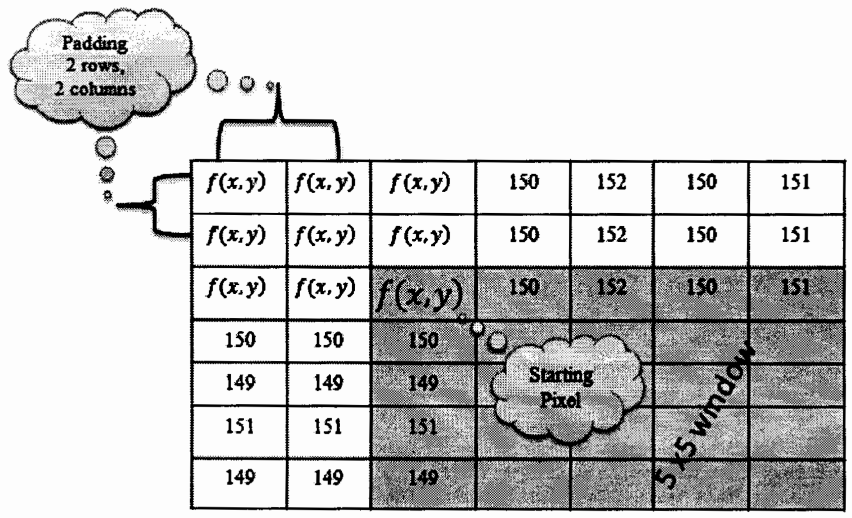
Where B1,B2,B3,B4,B5,B6,B7,B8 are the eight background pixels. Background information can be very useful in detection of edges. Median value of these eight background pixels will represents background information.

4.6.1 Detail explanation of algorithm

The noise map is constructed by noise detection module, which identifies noise as ***one*** and noise free as ***zero***. If the first pixel is noise free, its concerned value is picked from input image and placed at concerned place in restored image. Now second pixel is checked and if it is noisy pixel than a 5 x5 window is made from input image. Starting from the first pixel $f(x,y)$, keeping it in center and by padding two rows and two columns as shown in figure 4.6 four directions values are extracted from input image. (This pad array is used to get pixels values for four directions when checking first two rows and last two rows in the same way first two columns and last two columns) Median of each direction is calculated. Then eight background pixels are considered which are useful in the identification of edges. Median value of these eight pixels represents background information $B(x,y)$. Standard deviation $S_k(x,y)$ $(k = 1, 2, 3, 4)$ of four directions is calculated because standard deviation (∂) shows much dispersion or much variation from the average. Small value of ∂ tends to shows that data value are very close to the mean value. Whereas in case of high value of ∂ indicates that data points are spread over a large area. While in case of DBSNNF noise removal algorithm if there is one or more directions having same standard deviation $S_1 = S_2 = S_3 = S_4 = 0$ than difference between median value of background pixels ($MedianBGP$) and median value of each direction ($MedianDir_i$) is calculated which is shown in equation 4.6.

$$diff_i = | MedianBGP - MedianDir_i | \tag{4.6}$$

Whereas *i* ranges from 1 to 4. Since edges always differ from background, bigger difference of any direction yields a possible edge direction.



$f(x,y)$	$f(x,y)$	$f(x,y)$	150	152	150	151
$f(x,y)$	$f(x,y)$	$f(x,y)$	150	152	150	151
$f(x,y)$	$f(x,y)$	$f(x,y)$	150	152	150	151
150	150	150				
149	149	149				
151	151	151				
149	149	149				

Figure 4.7: Padding two rows & two columns

For example if difference between $MedianBGP$ and $MedianDir_1$ is bigger than difference of other three directions $\{ MedianDir_i \ (i = 2,3,4)\}$. It indicates that d_1 has an edge. If we have any two differences equal to each other than it means that there exist two edges, replacing median value of any direction will repair the edge pixel.

For example: if

$$diff_1 = diff_2 \,||\, diff_1 = diff_3 \,||\, diff_1 = diff_4 \text{ replace the noisy pixel with } MedianDir_1$$

in the same way if

$$diff_2 = diff_3 \,||\, diff_2 = diff_4 \text{ replace the noisy pixel with } MedianDir_2$$

and if

$$diff_3 = diff_4 \text{ replace the noisy pixel with } MedianDir_3 \text{ or } MedianDir_4$$

If all the Standard deviation $S_k(x, y)$ has different values that means there exist no edge than find the direction with minimum standard deviation with the equation 4.7.

$$S = \min\{S_k : 1 \le k \le 4\} \tag{4.7}$$

Now direction having small standard deviation is found. Combine the direction having small standard deviation vertically with 5 x 5 window to make a 5 x 6 window as shown in figure 4.8(b). Now calculate the median value of 5 x 6 window excluding the center noisy pixel and place that value in the center pixel colored in red as shown in figure 4.8(b). The center pixel is excluded because we estimate a value for noisy pixel by taking a median of noise free pixels. In this way all the pixels are analyzed for noise detection and passed to DBSNNF noise removal algorithm one by one which yields restored image.

245	151	190	210	214
225	243	216	8	155
172	15	255	141	144
189	199	245	152	198
197	9	244	150	150

(a) 5 x5 window with d_2 having small ∂

245	151	190	210	214	172
225	243	216	8	155	15
172	15	■	141	144	255
189	199	245	152	198	141
197	9	244	150	150	144

(b) 5 x 6 window by replicating d_2 values

Figure 4.8: Replicating values of direction having smaller ∂

4.7 Repeated architecture

Once the DBSNNF retrieves a restored image its peak signal noise ratio (PSNR) is calculated and compared with the previous value of PSNR of filtered image. For the first time PSNR value will be initialized to zero and when first restored image is retrieved its PSNR value is calculated and compared with previous value which is zero. So the condition satisfy, new PSNR value is greater than previous value so restored image is sent again to the filter as shown in figure 4.9. This process will continue until PSNR value of the restored image start decreasing. When the condition does not satisfy $previous_{PSNR} < new_{PSNR}$ than system exit. Which means that filter has restored the best image. So we will consider the PSNR value which is highest among all. The image with high PSNR value will be considered the best image filtered. Normally when the noise is less the filter gives best PSNR value at first iteration. As the noise ration increases the filter gives best PSNR value at second iteration and so on. When the noise ratio exceeds 80% then the filter gives best PSNR at fourth iteration, which is the last threshold value. Setting the threshold value plays an important role in detection of noise. These threshold values are considered after performing many experiments on different image with different noise ratio. More detail about threshold is discussed in chapter 5.

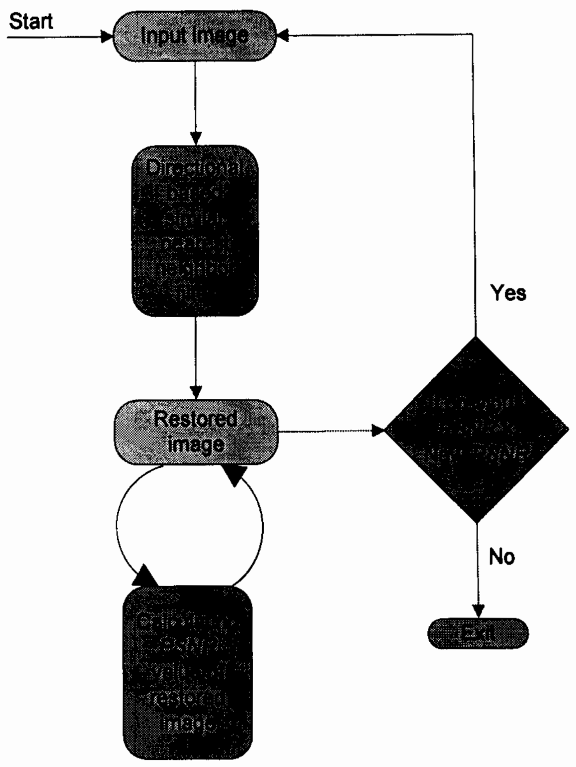


Figure 4.9: Repeated flow of the system

4.8 Standard Deviation

Standard deviation is used to tell how the values or data points are distributed. The standard deviation is represented by the Greek symbol σ. If data points are spread over a wide region then standard deviation value will be high showing that all the values are different from each other and if data points are close to each other than value of standard deviation will be small, showing that values are same or almost same to each other. Standard Deviation plays a key role in making decision based on the data points. The common definition of standard deviation S of a data vector X is shown in equation 4.8.

$$s=\left(\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^{\frac{1}{2}} \tag{4.8}$$

$$\bar{x}=\frac{1}{n}\sum_{i=1}^{n}x_i \tag{4.9}$$

Where n is the number of elements in the sample, $\bar{x}$ is the mean.

4.8.1 Example of standard deviation

Finding the heights of dogs (in millimeters) as shown in figure 4.10.

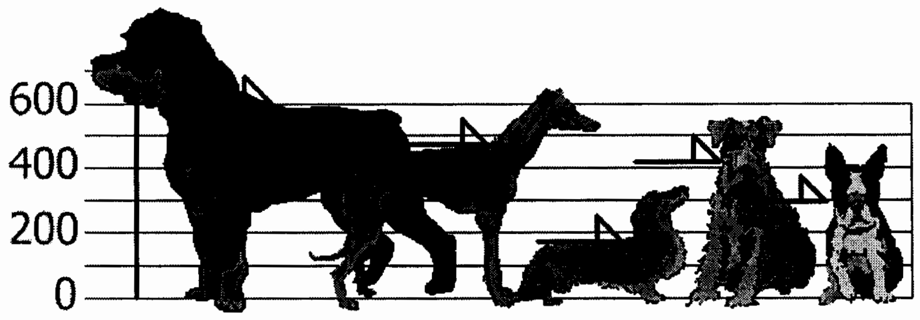


Figure 4.10: Example figure to calculate standard deviation

The height of each dog (at the shoulders) is: 600mm, 470mm, 170mm, 430mm and 300mm.

In order to calculate the standard deviation finds Mean, Variance, and Standard Deviation.

The mean is calculated by adding all dogs height divided by total number of dogs as shown in the calculation below.

$$\text{Mean} = \frac{600 + 470 + 170 + 430 + 300}{5} = \frac{1970}{5} = 394$$

The mean (average) height is 394 mm. Plotted as shown in the figure 4.11.

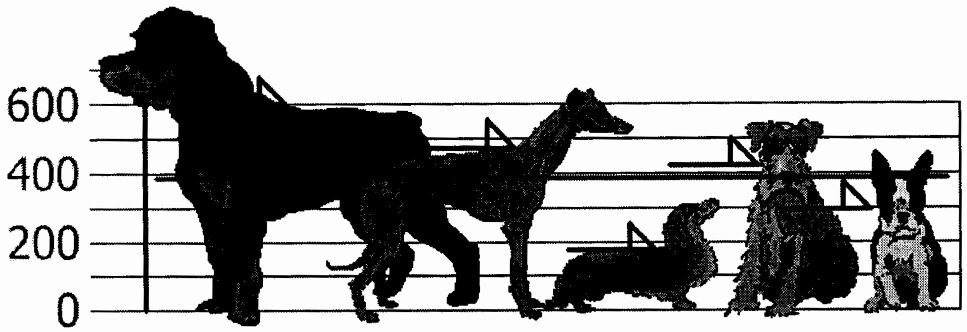


Figure 4.11: Showing average height

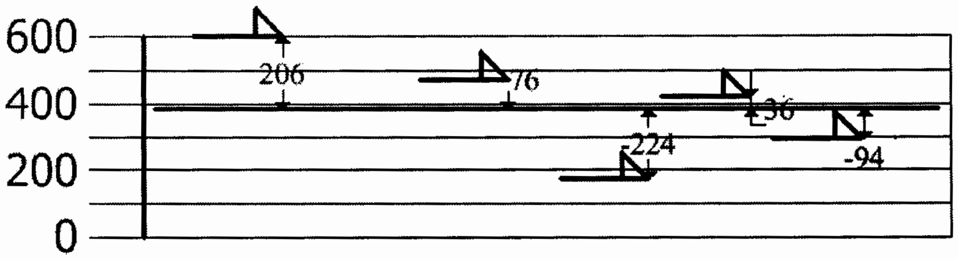


Figure 4.12: Showing difference from mean

Now calculating each dog difference by subtracting mean with its actual shoulder height as shown in figure 4.12.In order to calculate the Variance, take difference of each dog, square it, and then average the result:

$$\begin{aligned}\text{Variance: } \sigma^2 &= \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5} \\ &= \frac{42,436 + 5,776 + 50,176 + 1,296 + 8,836}{5} \\ &= \frac{108,520}{5} = 21,704\end{aligned}$$

The Variance calculated is 21,704. Now in order to find the Standard Deviation just square root the Variance value that would be

Standard Deviation: $\sigma = \sqrt{21,704} = 147.32... = 147$ (to the nearest mm).

Now in this way we can see that which heights are within one Standard Deviation (147mm) of the Mean, as shown in figure 4.13.

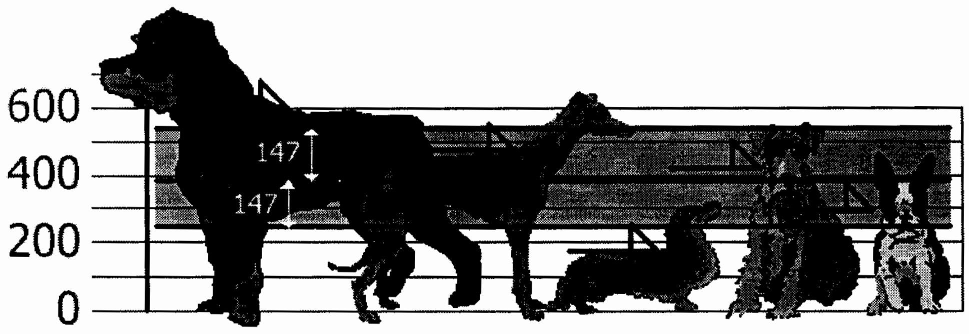


Figure 4.13: Showing values with in standard deviation

4.9 Summery

Proposed solution for random valued impulse noise removal is based on two separate phases, detection phase and filtering phase. In detection phase, a window of size 5x5 is made and scanned across the noisy image. In order to identify that whether the central pixel is corrupted or having impulse noise or not a novel technique having two modules, noise detection and filtration process is used. In first step, first order difference is calculated and then this image is passed to calculate order difference which is then used to develop a noise map. Pixel which are considered as noisy pixels are the candidates for noise removal, are sent to noise removal filter. The proposed filtering technique is based on directional median (MED) filter using standard deviation which works in four different directions respectively. Whereas to preserve edges and noise-free pixels the whole process is repeated and the restored image will be the input image for the second time processing. This process will continues until the PSNR value of the restored image start decreasing.

Chapter 5

Simulation results

Chapter 5

Simulation Results

5.1 Peak signal noise ratio

To study and analyze the performance and working of proposed method, results have been compared with many states of the art filters proposed earlier. The images were corrupted with different noise ratio in a size of 512 x 512. The peak signal noise ratio is defined in equation 5.1, which is used for the quality of measure of the images.

$$PSNR = 20\log_{10}\left(\frac{MAX_f}{\sqrt{MSE}}\right) \tag{5.1}$$

Whereas Mean square error (MSE) is defined as:

$$MSE = \frac{1}{mn}\sum_{0}^{m-1}\sum_{0}^{n-1}|f(i,j)-g(i,j)|^2 \tag{5.2}$$

Where f represents the matrix data of our original image, g represents the matrix data of our degraded image, m represents the numbers of rows of pixels of the images, i represents the index of that row, n represents the number of columns of pixels of the image and j represents the index of that column, MAX_f is the maximum signal value that exists in our original "known to be good" image. The mean squared error (MSE) for our practical purposes allows us to compare the "true" pixel values of our original image to our degraded image. The MSE is the average of the squares of "errors" between our refined image and our noisy image or corrupted image. The error is the amount by which the values of the noisy image different from the refined image.

The proposal is that the higher the PSNR, the better degraded image was reconstructed to match the original image and the best of the reconstruction algorithm. This can happen because we want to minimize the MSE between the images with respect to the maximum value of the image signal.

5.2 Experimental Results

Figure 5.1(a) shows the original image of size 21 × 21 blocks. Figure 5.1(a) is corrupted with 30% random value impulse noise ration as shown in figure 5.1(b). Figure 5.1(c) shows the restoration of image with median filter. Figure 5.1(d) shows the image restored by directional weighted median filter. Figure 5.1(e) show screen shot of adaptive median filter. Figure 5.1(f) shows the proposed filter result. If we compare the PSNR value of these filters, it clearly shows that proposed filter perform well against state of art filters. PSNR value for median filter is 16.69 dB, in the same way PSNR value is 15.56 dB for directional weighted median filter, for adaptive median filter PSNR is 19.77 dB and in the last PSNR for the proposed filter is 22.84 dB.

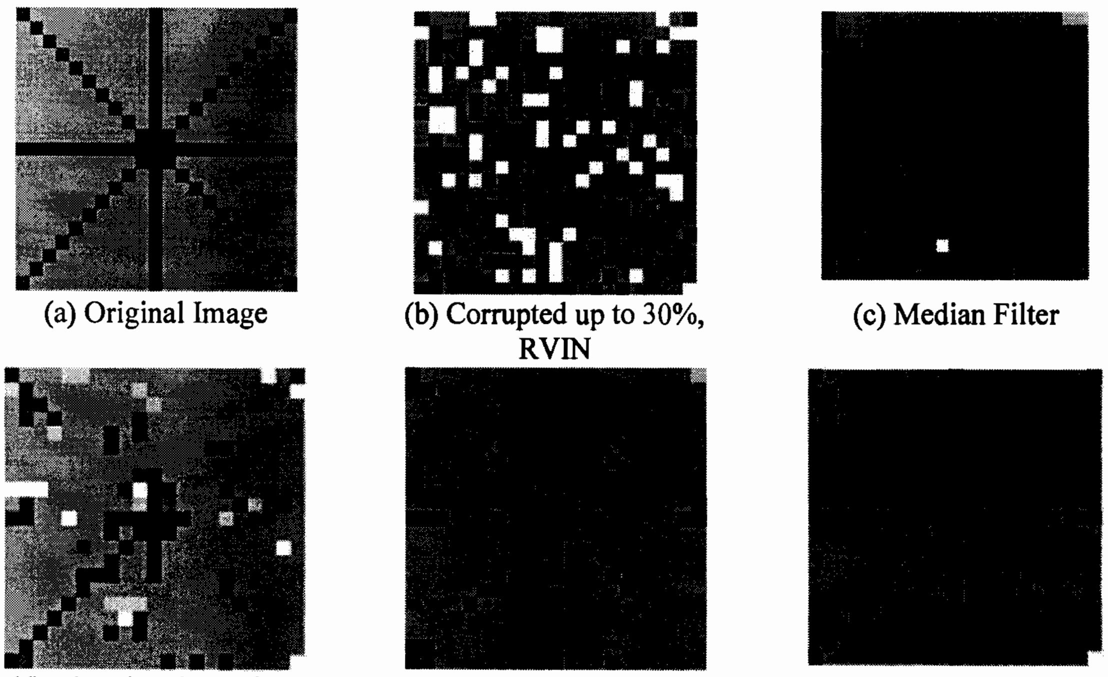

(a) Original Image (b) Corrupted up to 30%, RVIN (c) Median Filter
(d) Directional weighted median filter (e) RORVIN median filter (f) Proposed filter

Figure 5.1: Image having four directions of 21 x 21 block

The result shows that the proposed filter is the best. Adaptive median filter performance comes in second, than comes directional weighted median filter, conventional median filter performs poor in detection of edges.

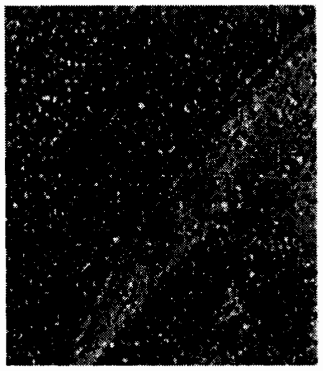
5.2.1 Results of Lena image

Table 5.1: Comparison of ***Lena*** image with different filters corrupted with varying strength of RVIN

Methods	5%	10%	15%	20%	25%	30%	40%	45%	50%
SD-ROM	39.22	35.89	34.09	32.48	31.05	29.86	27.32	24.96	22.35
ACWM	35.72	34.47	33.41	32.44	31.35	30.40	27.86	25.66	22.51
PWMAD	36.46	34.86	32.69	30.58	28.01	25.94	22.41	19.42	17.08
DWM	36.05	35.15	34.48	33.81	33.09	32.43	30.64	29.14	26.57
RORVIN	41.16	38.34	36.42	**34.96**	33.48	32.50	30.98	29.56	**28.76**
Proposed Method	**46.39**	**40.51**	**37.23**	34.54	**33.55**	**32.55**	**31.15**	**30.04**	26.05

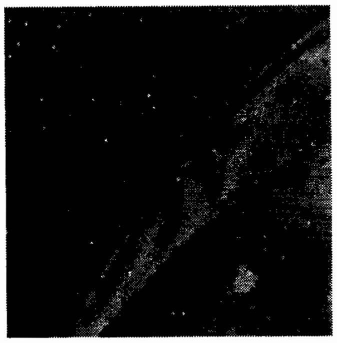
In our experiments Lena image is corrupted with Random value impulse noise (RVIN) (5% to 50% of noise) and is subjected to different filtering schemes as discussed above. The entire well known filters of RVIN are compared with the proposed method.

(a) True image

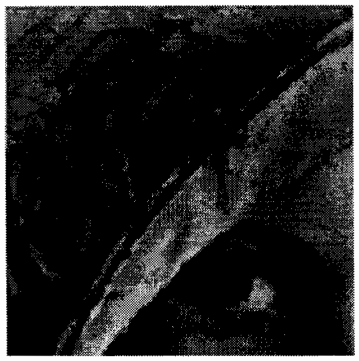

(b) Noisy image

(c) SDROM

(d) ACWM


(e) PWMAD

(f) DWM


(g) RORVIN

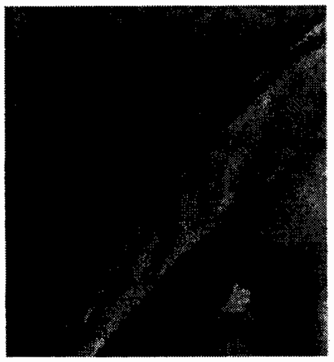

(h) Proposed method

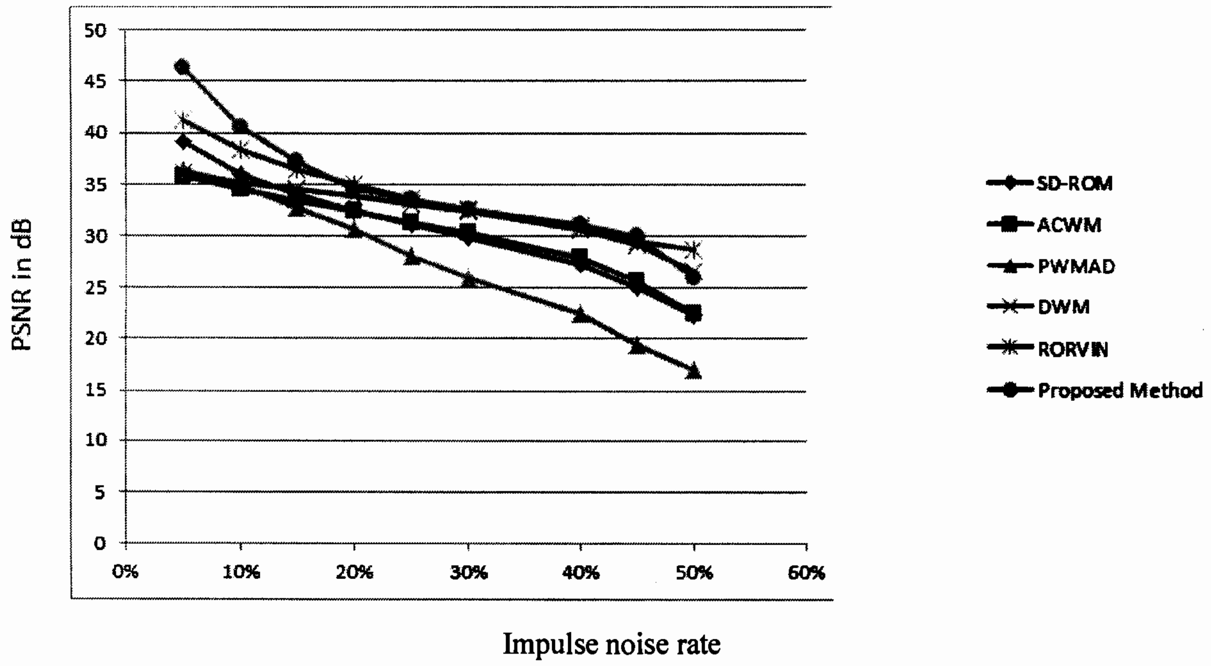
Figure 5.2: Lena image degraded with 30% of RVIN and restored by different methods.

Experiments have shown that the proposed method has performed well till 45% as compared to the other filters for Lena image. The proposed method perform very well in 5%, 10% and 15% noise but when 20% noise is added to the image, Removal of random value impulse noise (RORVIN) PSNR value is a bit higher than proposed method but the proposed method performs better than the other filters as shown in table 1. As the noise increases from 20% to 25%, 30%, 40%, 45% again the proposed method perform well from all the filters, even RORVIN on Lena image.

Figure 5.2(a) shows the original image of Lena, In figure 5.2(b) it is corrupted with 30%

RVIN and figure 5.2(c) shows Signal dependent rank ordered mean (SD-ROM) filter. Figure 5.2(d) shows Adaptive center weighted median filter. Figure 5.2(e) shows Pixel wise MAD filter. Figure 5.2(f) show Directional weighted median filter. Figure 5.2(g) show Removal of random value impulse noise filter and figure 5.2(h) shows the refine image obtained after applying the proposed filter. As we can see from figure 5.2(g) the proposed filter perform well and remove noise more efficiently than any other filter.

Figure 5.3 show the graphical representation of Table 1. We can clearly see that PSNR is high when the RVIN is low [5% to 20%] in Lena image, as the noise increases the PSNR values start to decrease because of random value which is very hard to detect. When the RVIN is 20%, proposed method performance is a bit lower than RORVIN but it performs better than rest of the methods. The proposed method keeps on performing better than all other filters until RVIN is reached to 50% in Lena image. The PSNR value of proposed method is lower than RORVIN & DWM when 50% noise is added but it performs better the other standard filters as shown in figure 5.3.



Impulse noise rate

Figure 5.3: Graphical chart representation of Lena image with different methods

5.2.2 Results of Bridge Image

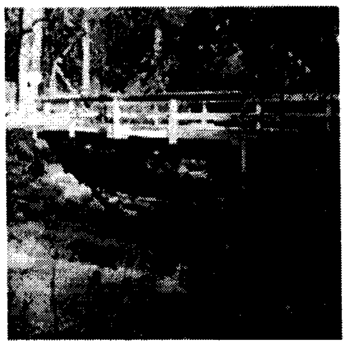
Now we will be showing the comparison of bridge image. The table 2 shows that proposed method performs better from all other well know filters. There was low performance in Lena image when 20% noise is added from RORVIN and the proposed method performs well till 45% and its performance start decreasing from 50% noise. Where as in case of bridge image proposed method performance is better right from start till 60% except from DWM filter as shown in table 2.

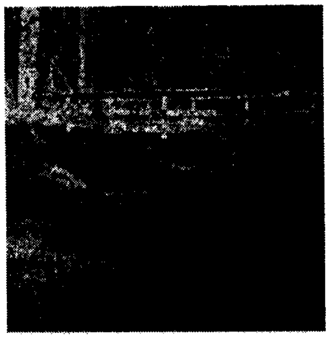
Table 5.2: Comparison of ***Bridge*** image with different filters corrupted with varying strength of RVIN

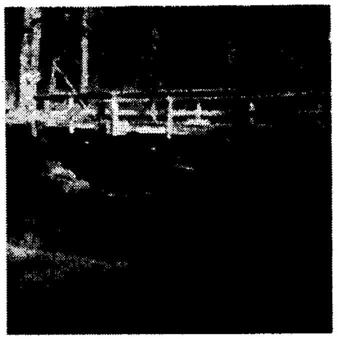
Methods	10%	20%	30%	40%	50%	60%
SD-ROM	26.62	26.35	24.89	23.03	21.18	19.21
ACWM	25.89	25.14	23.99	22.61	20.88	19.09
PWMAD	25.89	25.22	22.91	20.27	17.86	15.77
DWM	26.02	26.50	24.87	24.09	23.08	**21.41**
RORVIN	27.80	27.20	24.91	23.73	22.14	20.02
Proposed Method	**36.03**	**30.98**	**27.81**	**25.40**	**23.36**	21.32

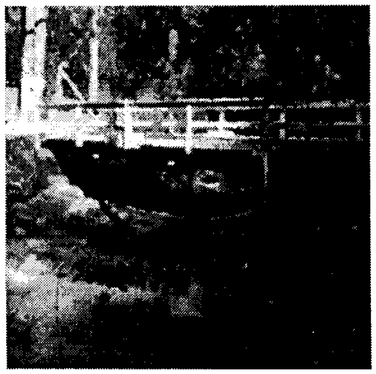
Figure 5.4(a) shows original Bridge image 5.4(b) Noisy image corrupted 30% with RVIN and. and figure 5.2(c) shows Signal-dependent rank-ordered-mean (SD-ROM) filter. Figure 5.2(d) shows Adaptive center weighted median filter. Figure 5.2(e) shows Pixel wise MAD filter. Figure 5.2(f) show Directional weighted median filter. Figure 5.2(g) show Removal of random value impulse noise filter and figure 5.2(h) shows the refine image obtained after applying the proposed filter.

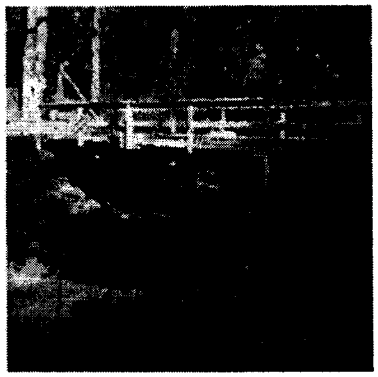
The performance of proposed filter is better on bridge image because bridge image have more boundaries than Lena image. As said earlier the proposed filter is best in detecting boundaries and retaining them.

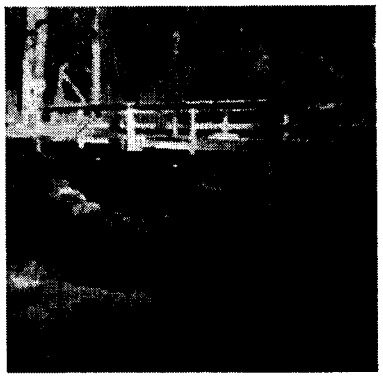

(a) True Image

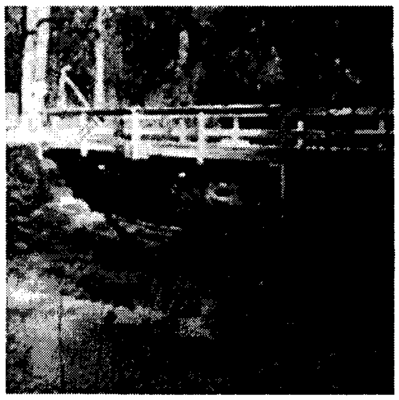

(b) Noisy Image


(c) SD-ROM


(d) ACWM


(d) PWMAD


(e) DWM


(f) RORVIN

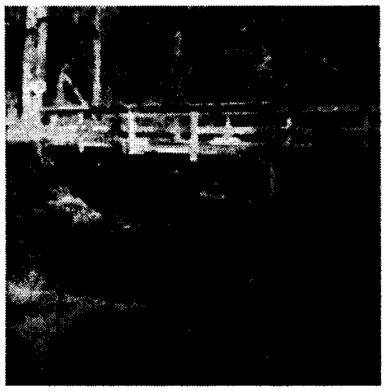

(g) Proposed Method

Figure 5.4: Bridge image degraded with 30% of RVIN and restored by different methods.

The proposed filter detects edges in four possible directions, and then replace the damaged pixel noise about the noise of the free edge average cost. The simulation showed that the proposed filter provides excellent performance suppressing impulsive noise in any situation, even if there is one edge in the image. The graphical representation of bridge image is shown in figure 5.5. The graph clearly shows that the proposed method performs better than all well know filters. The value of PSNR remains high throughout the experiments. As said earlier proposed method has performed better than any other method on those images which contain many edges. The reason is that we are considering the background details also in detecting the edges which gives a kick back in the performance of detection and estimating a value for noisy pixel, especially in case of a single edge or multiple edges from the rest of the methods.

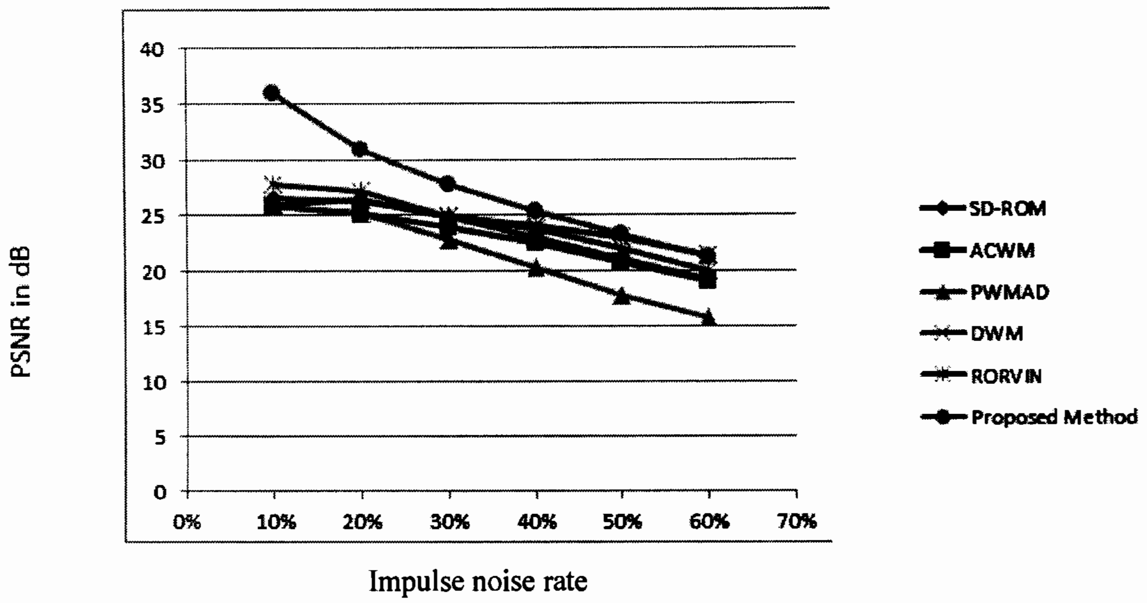


Figure 5.5: Graphical chart representation of Bridge image with different methods

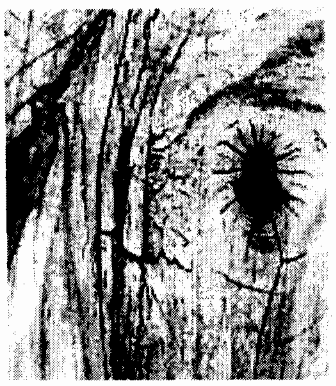
5.2.3 Results of wood image

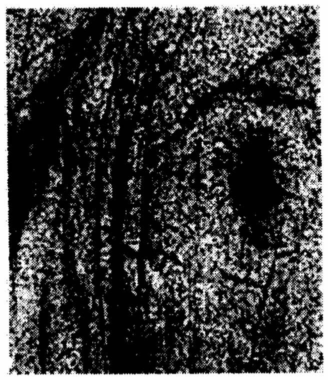
The proposed directional based similar nearest neighbor filter (DBSNNF) is compared with the results taken from a paper published in "*International Journal of Signal Processing*" in 2009. The paper ***"Multi-stage Directional Median Filter"*** by Zong Chen and Li Zhang perform the simulation on "*wood.jpg*" image. In this paper they have added salt-and-pepper noise (SPN), whereas we have added random value impulse noise (RVIN). As mention earlier that salt-n-pepper is easy to detect and remove as compared to random value impulse noise but still the performance of proposed Directional based similar nearest neighbor filter (DBSNNF) is better than Multi stage directional median filter (MDM) as shown in table 3.

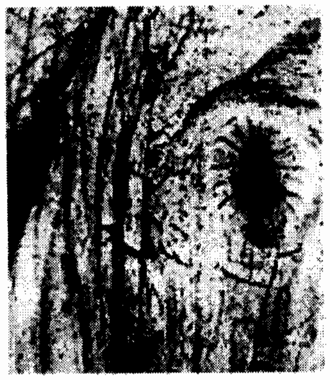
Table 5.3: Comparison of ***Wood*** image with different filters corrupted with varying strength of RVIN

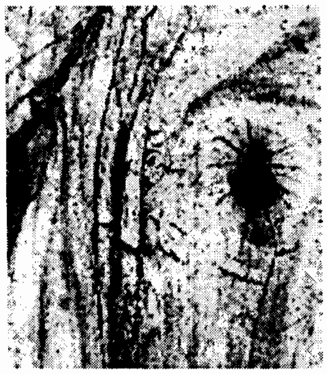
Noise level	CM	DWM	AM	MDM	**Proposed DBSNNF**
10%	17.48	20.63	20.66	24.87	**30.62**
20%	16.79	17.00	19.89	22.28	**26.27**
30%	15.73	14.46	18.97	20.50	**23.28**
40%	14.01	12.22	17.90	18.87	**21.19**
50%	12.12	10.40	16.90	17.40	**19.26**
60%	10.14	8.84	15.83	15.91	**17.20**
70%	8.42	7.56	14.74	14.43	**15.07**
80%	6.89	6.40	13.05	12.63	**13.18**
90%	5.53	5.38	9.59	9.95	**11.59**

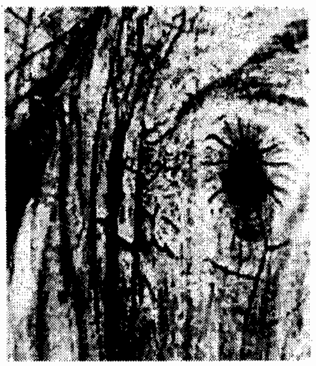
The performance of proposed DBSNNF increases as the image has more edges in it as compared to other state of art filters. Figure 5.6 show the wood.jpg image filtered by different state of art filters shown in table 3. Figure 5.6(a) is the original wood image which is corrupted with 30% random value impulse noise as show in figure 5.6(b). Conventional median filter is shown in figure 5.6(c). Directional weighted median filter result is shown in figure 5.6(d). Adaptive median filter result is shown in figure 5.6(e). Multi-stage directional median filter result is shown in figure 5.6(f) and proposed Directional based similar nearest neighbor filter is shown in figure 5.6(g). Proposed method performs best and then on second number multi stage median filters [14]. Both of the methods Multi stage median filter and proposed directional based similar nearest neighbor filter are designed in such a way to detect edges in an image, even the small edges. The point where DBSNNF has an edge over MDMF is that the writer does not mention how they have detected noise and constructed a noise map and in removing the noise they haven't handled the entire scenario like if there is two or more edges or if there is a single edge or there exist no edge in a 5 x5 window. The filter dose not handle this situation where as in case of DBSNNF first thick sharp edges are handled by calculating first order derivative and then it is sent to calculate second order derivative which identifies think edges. In removing the noise and preserving the edges all four direction are checked in a 5 x 5 window for two or more edges and then it checks if there exist a single edge and in the last if there exist no edge in a 5x 5 window. So this is the main point where DBSNNF has edge than all other sate of art filters.

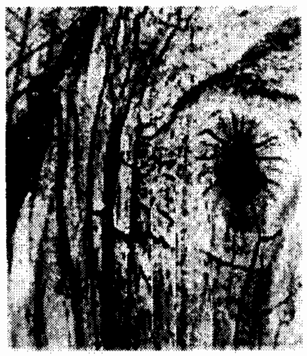

(a) Original image


(b) 30% RVIN added


(c) CM filter


(d) DWM filter


(e) AM filter


(f) MDM filter

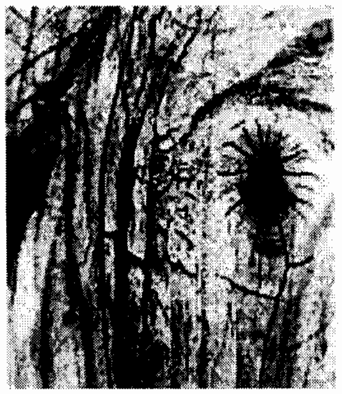

Proposed DBSNN filter

Figure 5.6: Wood image degraded with 30% of RVIN and restored by different methods.

Figure 5.7 shows clearly when there are both thick and think edges are present in an image the DBSNNF performs better than all other state of art filters. When the noise is less the DBSNNF performs outclass than any other filter but when the noise 70% to 80% AMF and MDM filters performance come close to DBSNNF but still there PSNR values are less than DBSNNF but as the noise is increased from 80% to 90% DBSNNF perform better by raising its PSNR value as shown in figure 5.7.

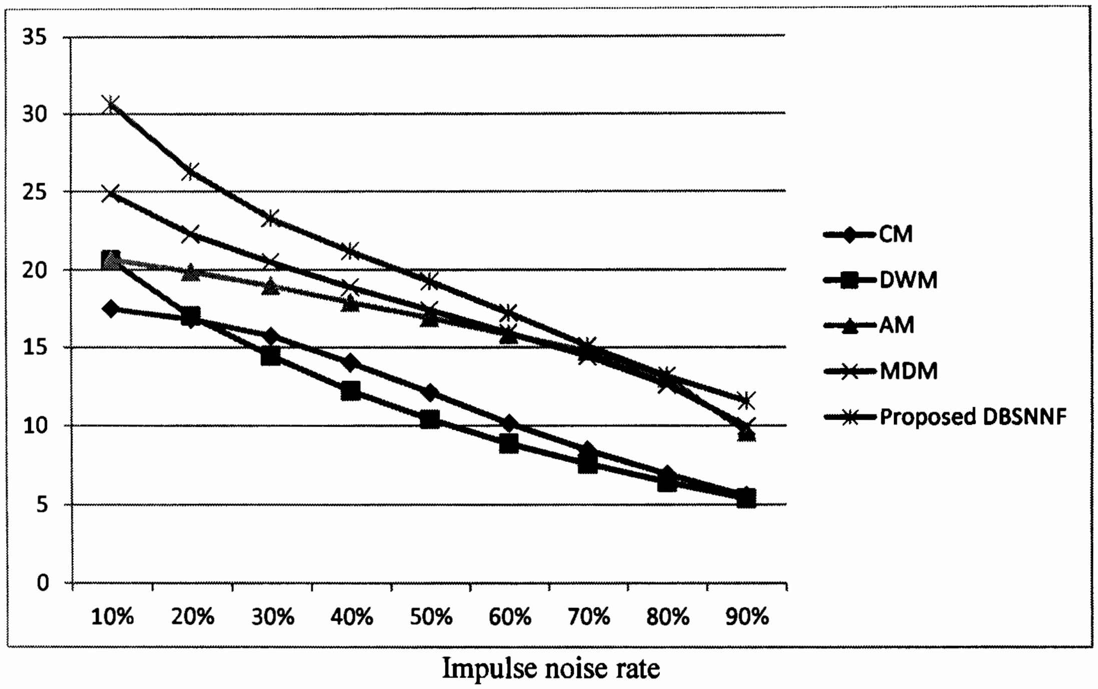


Figure 5.7: Graphical chart representation of wood image along with different methods

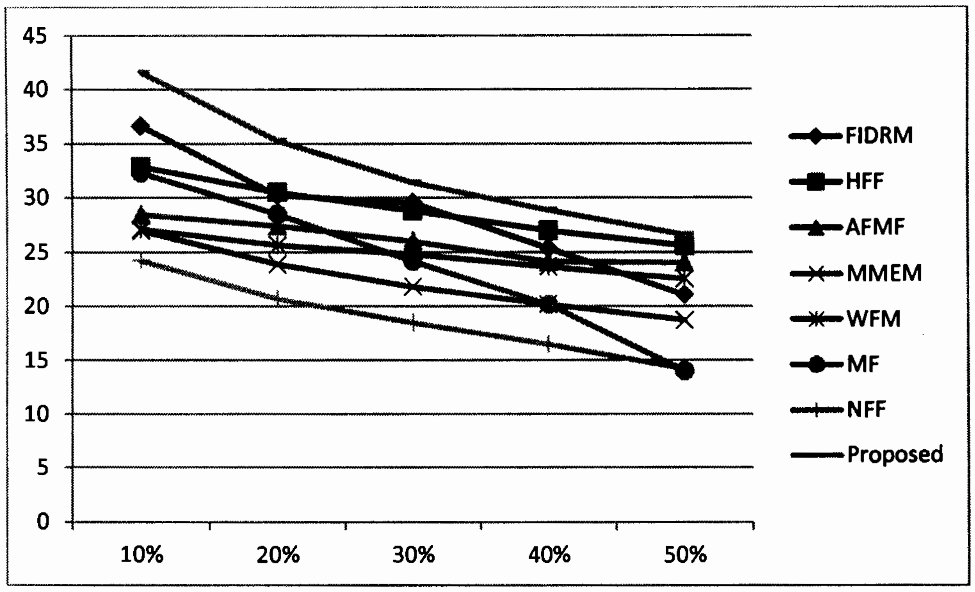
5.2.4 Results of Salt-n-pepper

The comparison of PSNR results among several state-of-the-art filters i.e., FIDRM, HFF, AFMF, MMEM, WFM, MF, NFF and proposed (AFRDM) filter for a 350 × 350 "Pepper" image corrupted with random-valued impulse noise densities ranging from 10% to 50% is given in Table 4.

Table 5.4: Comparison of ***Salt-n-pepper*** image with different filters corrupted with varying strength of RVIN

Noise / Method	10%	20%	30%	40%	50%
FIDRM	36.62	30.20	29.49	25.34	21.05
HFF	32.85	30.47	28.80	26.96	25.60
AFMF	28.42	27.38	26.00	24.12	24.05
MMEM	26.97	23.87	21.80	20.17	18.70
WFM	27.08	25.63	24.80	23.66	22.54
MF	32.27	28.45	24.16	20.15	14.00
NFF	24.21	20.66	18.44	16.47	14.21
Proposed	**41.58**	**35.28**	**31.39**	**28.83**	**26.59**

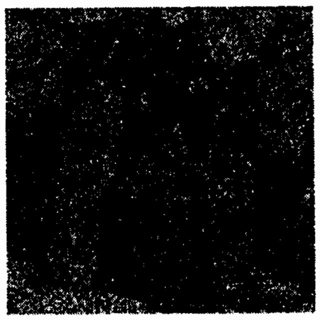
The graphical presentation of proposed DBSNNF is shown in figure 5.8. The graph clearly show that proposed filter performs very well as compared to other state of art filters. One thing to mention hear is that when noise is low the proposed filter performs extra ordinary but when RVIN increases, its performance start to decrease dramatically. The reason behind this is first order derivative which perform well to detect thick edges but when noise ratio is increased it performances decreases.



Impulse noise rate

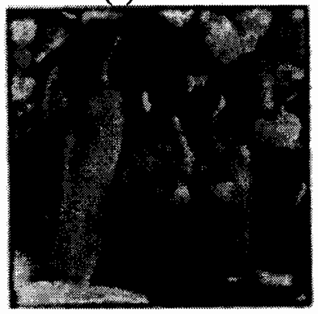
Figure 5.8: Graphical chart representation of salt-n-pepper image along with different methods

(a)) Original Pepper image


(b) Corrupted with 50% random-valued impulse noise

(c) WFM

(d) MMEM,


(e) MF

(f)) HFF

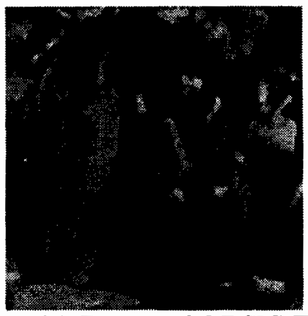

(g) Proposed DBSNNF

Fig. 5.9: Salt-n-pepper image degraded with 50% of RVIN and restored by different methods

5.3 Threshold Setting

To ensures better performance of proposed filter it is applied recursively and iteratively. Threshold values increases as per iteration. Setting the threshold value plays an important part in the detection of noise and making a noise map. Experiments has shown, keeping very small threshold allows corrupted pixels affected by RVIN to pass as they satisfy the threshold, which causes poor performance of the filter. Performing many simulations it has been observed that the threshold set mention below gives better results.

$$[TH1\ \ TH2\ \ TH3\ \ TH4] = [90\ \ 160\ \ 190\ \ 210]$$

The proposed filter works very well on second iteration when RVIN is low [5% to 20%]. It gives high PSNR value on second iteration where Threshold TH2 is 160. As the RVIN increases the proposed filter perform better on third iteration [25% to 40%] and as the RVIN increases from 40% the proposed filter perform better on fourth iteration. Different values of PSNR are yield on different noise ration as shown in Table 5.5 and its graphical representation is shown in figure 5.10 and figure 5.11.

Table 5.5: Different PSNR values on different RVIN of Lena & Bridge image

Iterations	1	2	3	4
Threshold	90	160	190	210
Lena 5%	40.09	46.64	46.28	45.71
Lena 10%	33.76	40.18	40.30	39.91
Lena 15%	30.21	36.54	37.04	36.72
Lena 20%	27.65	33.74	34.54	34.33
Lena 30%	25.88	31.40	32.57	32.55
Lena 40%	24.42	29.46	30.97	31.15
Lena 50%	19.56	24.04	25.65	25.99

Table 5.5.1: Lena Image *PSNR* values on different Thresholds

Iterations	1	2	3	4
Threshold	90	160	190	210
BR 5%	37.41	40.27	39.51	30.02
BR 10%	31.97	36.03	35.70	35.30
BR 20%	26.23	30.65	30.98	30.77
BR 30%	22.85	27.07	27.81	27.75
BR 40%	20.41	24.35	25.31	25.40
BR 50%	18.52	22.03	23.14	23.36
BR 60%	16.87	19.92	21.03	21.32

Table 5.5.2: Bridge Image *PSNR* values on different Thresholds

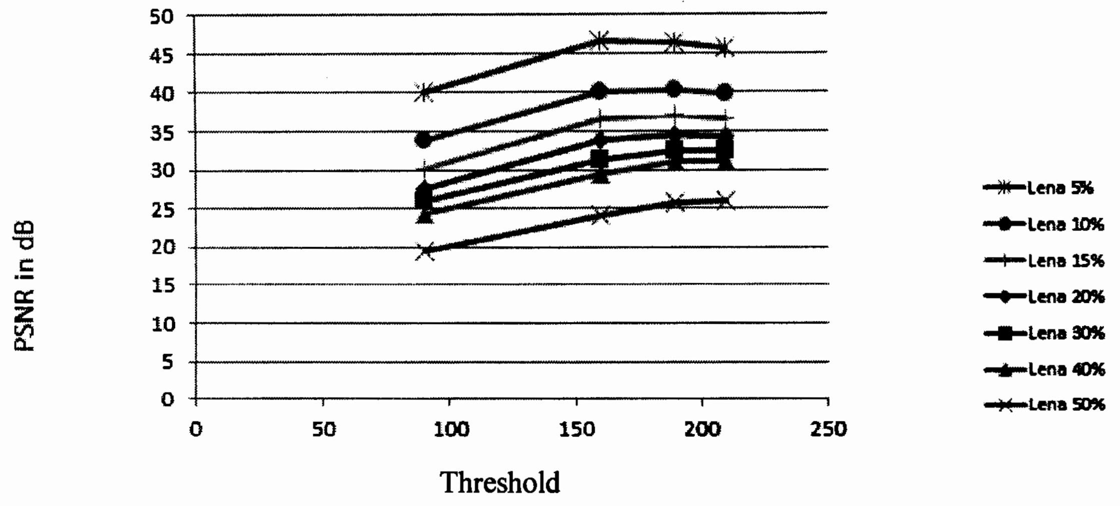


Figure 5.10: Different PSNR values on different threshold of Lena image

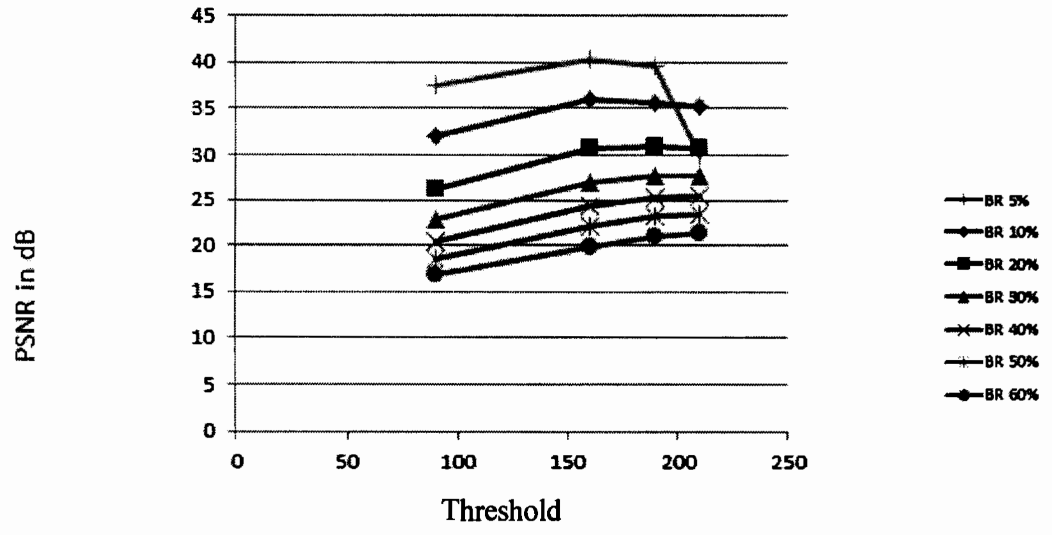


Figure 5.11: Different PSNR values on different threshold of Bridge image

In the same way the performance of proposed DBSNNF on wood and salt-n-pepper is shown in table 5.6. When random value impulse noise is low in wood image like 10%, it perform very well at first iteration and as the noise increase from 10% and onwards it start performing well in second and then in third iteration and so on. But in salt-n-pepper image when noise is low the proposed filter perform well in second iteration and as the noise increases it started to perform well in third and fourth iteration. The graphical presentation of both images is shown in figure 5.12 and 5.13.

Table 5.6: Different PSNR values on different RVIN of Wood & Salt-n-pepper image.

Iterations	1	2	3	4
Threshold	90	160	190	210
Wood 10%	30.64	27.18	26.50	25.75
Wood 15%	26.73	26.23	25.11	24.49
Wood 20%	24.58	24.66	23.82	23.27
Wood 30%	21.37	22.35	21.93	21.55
Wood 40%	18.82	20.46	20.37	20.12
Wood 50%	16.51	18.60	18.89	18.81
Wood 60%	14.30	16.71	17.40	17.51

Table 5.6.1: Wood Image *PSNR* values on different Thresholds

Iterations	1	2	3	4
Threshold	90	160	190	210
SNP 10%	29.73	40.41	38.07	37.83
SNP 15%	26.16	36.25	38.07	37.83
SNP 20%	23.74	32.96	35.16	35.10
SNP 30%	20.33	28.38	31.12	31.39
SNP 40%	18.05	25.11	28.09	28.77
SNP 50%	16.08	22.40	25.54	26.56
SNP 60%	13.98	19.23	22.36	23.76

Table 5.6.2: Bridge Image *PSNR* values on different Thresholds

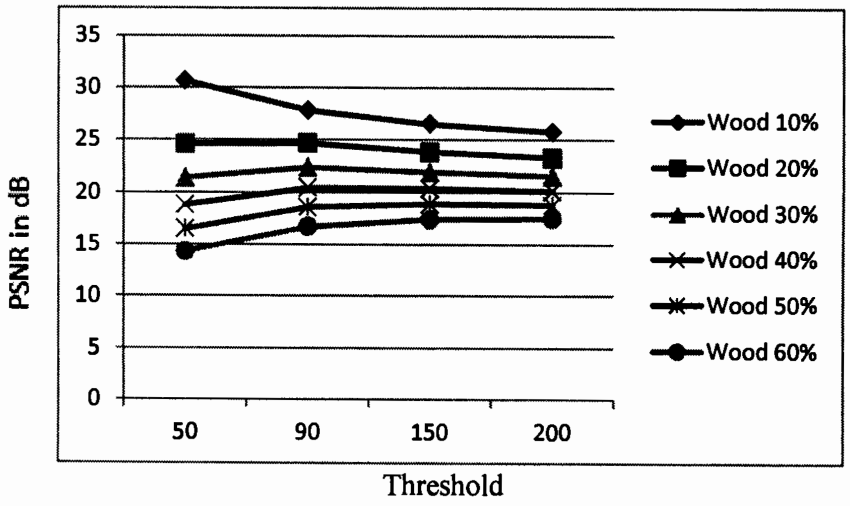


Figure 5.12: Different *PSNR* values on different threshold of Wood image

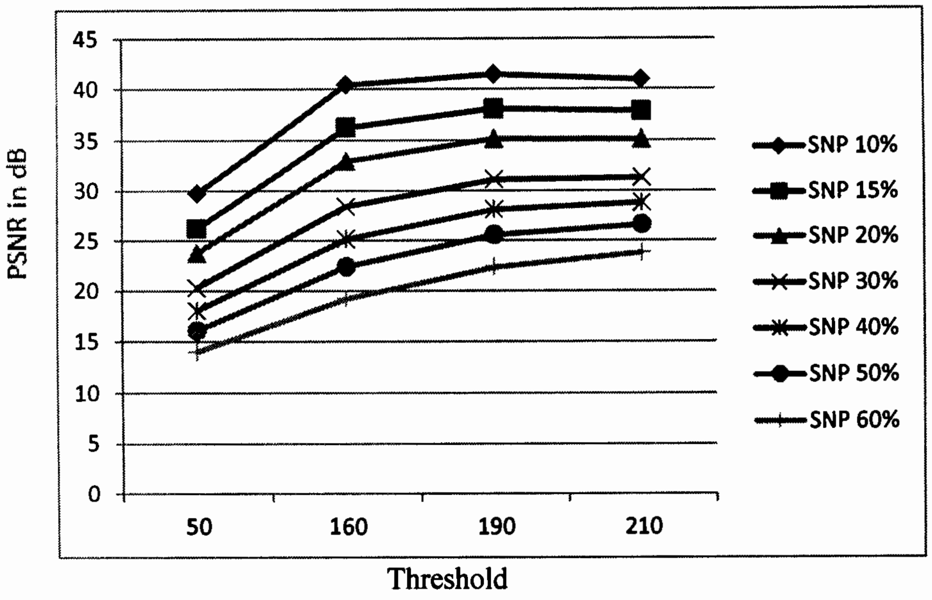


Figure 5.13: Different *PSNR* values on different threshold of Wood image

5.4 Summary

In this chapter, simulations have been performed on various images adding different noise ratio to Lena, Boat, Bridge & Wood image. The results are evaluated by considering Peak signal noise ratio and compared with different state of art filters. Simulation results shows that the proposed filter perform well against other state of art filters when images have fine details and sharp edges. A set of threshold values is considered and image is passed iteratively to same process again and again unless the Peak signal noise ratio starts decreasing.

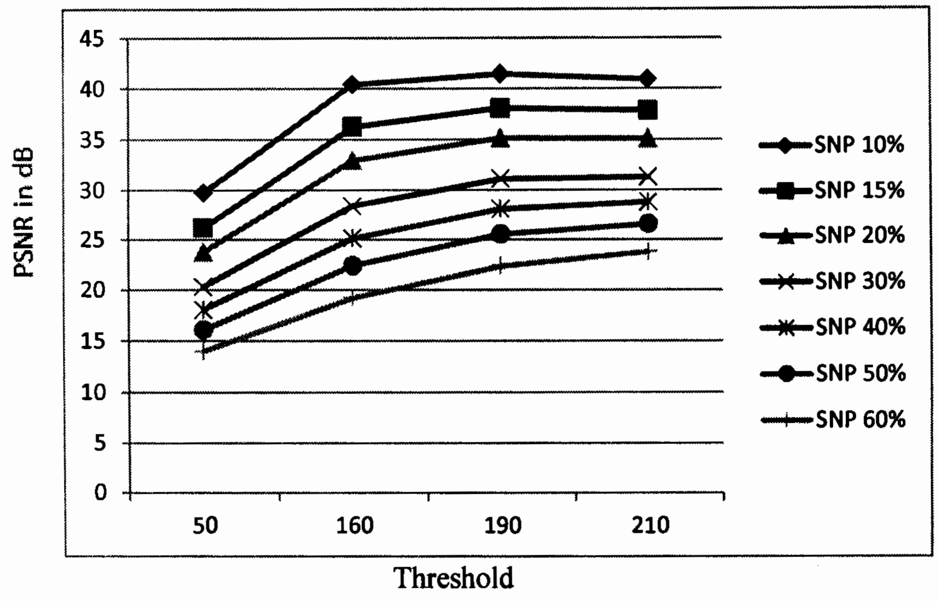


Figure 5.13: Different *PSNR* values on different threshold of Wood image

5.4 Summary

In this chapter, simulations have been performed on various images adding different noise ratio to Lena, Boat, Bridge & Wood image. The results are evaluated by considering Peak signal noise ratio and compared with different state of art filters. Simulation results shows that the proposed filter perform well against other state of art filters when images have fine details and sharp edges. A set of threshold values is considered and image is passed iteratively to same process again and again unless the Peak signal noise ratio starts decreasing.

Chapter 6

Conclusion & future works

consumed by proposed filter depends on noise ration and image details. The maximum time consumed when 90% noise is added to wood image was 95 seconds.

6.3 Future Work

As the proposed filter performs very well than other state of art filters in detection and removal of random value impulse noise. Better results can be achieved by making the threshold adaptive. Fixed threshold does not perform very well when different noise ratio is added to different images as compared to adaptive threshold. So the threshold can be made adaptive to achieve more better results. As the literature and previous research shows that adaptive threshold adjust its value according to the neighboring pixels. Secondly we can add fuzzy based directional median filter along with standard deviation for estimating the value for noisy pixel.

References

[1] Rafel C.Gonzalez, Richard E.Wood, " *Digital Image Processing*", 3rd edition, Engewood Cliffs, New Jersey: Prentice Hall, (2008).

[2] B Chandra and D Dutta Majumder,"*Digital Image Processing and Analysis*", Prentice-Hall, India, first edition, 2007.

[3] A.Said, H.Man, "*Similar neighbor criterion for impulse noise removal in images*", International Journal of Electronic & Communication, 64 (2010) 904–915

[4] Tukey. l.W, "*Exploratory Data Analysis,* Addison-Wesley, Reading, Mass, 1974.

[5] Pratt, W.K. "*Digital Image Processing*", lohn Wiley and Sons, Inc., New York, 1978.

[6] D. R. K.Brownrigg, "*The Weighted Median Filter*", Journal of Communications of the ACM, Volume 27, Number 8, August, 1984,

[7] Sung-Jea KO and Yong Hoon Lee,"*Center Weighted Median Filters and Their Applications to Image Enhancement*", IEEE Transaction and Circuits and Systems, Vol. 38, NO.9, September, 1991

[8] P.W Verbeek, H.A Vrooman, L.J Van Vliet "*Low-level image processing by max-minfilters*", Signal Processing, An International Journal, Volume 15, Issue 3, October 1988, Pages 249–258

[9] H. Hwang and R. A. Haddad, " *Adaptive Median Filtering*", IEEE Transaction on Image Processing, Vol. 4, No. 4, April 1995

[10] Tao Chen, Kai-Kuang Ma, and Li-Hui Chen, "*Tri-State Median Filter for Image Denoising*", IEEE Transactions on image processing, Vol. 8, 12, December 1999

[11] E. Abreu and S.K. Mztra, "*A Signal-Dependent Rank Ordered Mean (Sd-Rom) Filter - A New Approach For Removal Of Impulses From Highly Corrupted Images*", International Conference on Acoustics, Speech, and Signal Processing, Pages 2371- 2374, Vol.4, 1995

[12] Tao Chen and Hong Ren Wu, " *Adaptive Impulse Detection Using Center-Weighted Median Filters*", Signal Processing Letters, IEEE, Page(s): 1-3, Vol. 8, Jan, 2001

[13] Vladimir Crnojevic,Vojin Senk, Zeljen Trpovski,"*Advanced Impulse Detection Based on Pixel-Wise MAD*", IEEE Signal Processing Letters, vol. 11, no. 7, July, 2004

[14] Yiqiu Dong and Shufang Xu, "*A New Directional Weighted Median Filter for Removal of Random-Valued Impulse Noise*", IEEE Signal Processing Letters, Vol. 14, No. 3, March 2007

[15] Stefan Schulte, Mike Nachtegael, Valérie De Witte, Dietrich Van der Weken and Etienne Kerre,*"Fuzzy Impulse Noise Reduction Methods for Color Images"*, International Conference on Computational Intelligence, Theory And Applications, Part 25, 711-720, 2006

[16] Jung-Hua Wang, Wen-Jeng Liu, and Lian-Da Lin, *"Histogram-Based Fuzzy Filter for Image Restoration"*, IEEE Transactions On Systems, Man, and Cybernetics, Vol. 32, No. 2, April, 2002

[17] Wei-Yu Han and Ja-Chen Lin, *"Minimum-maximum exclusive mean (MMEM) filter to remove impulse noise from highly corrupted images"*, Journals & Magazines of Electronics Letters, Vol. 33 16th January 1997

[18] K.Somasundaramomasundaram, P. Shanmugavadivu, "*Impulsive noise detection by second-order differential image and noise removal using adaptive nearest neighborhood filter*", International Journal of Electronic and Communication, 472 – 477, 2009

[19] A.Said, H.Man, "*Similar neighbor criterion for impulse noise removal in images*", International Journal of Electronic and Communication, 904 – 915, 2010

[20] Zong Chen, Li Zhang."*Multi-stage Directional Median Filter*" International Journal of Signal Processing , 2009