# Standard Deviation Based Impulse Noise Filter

**Major Project submitted in partial fulfillment of the requirements for the award of degree of**

**Master of Technology**
**In**
**Information Systems**


Submitted by:
**Shreya Arora**
**(15/IS/2010)**

Under the Guidance of:
**Dr. O. P. Verma**




**Department of Information Technology**
**Delhi Technological University**
**Bawana Road, Delhi – 110042**

**(2010-2012)**

# Certificate

This is to certify that Ms. Shreya Arora (15/IS/2010) has carried out the major project titled "Standard Deviation Based Impulse Noise Filter" as a partial requirement for the award of Master of Technology degree in Information Systems by Delhi Technological University.

This work is an authentic piece of work carried out and completed under my supervision and guidance during the academic session 2010-2012. The matter contained in this report has not been submitted elsewhere for the award of any other degree.

(Project Guide)
**Dr. O. P. Verma**
Head of Department
Department of Information Technology
Delhi Technological University
Bawana Road, Delhi-110042

# Acknowledgement

I would like to express my heartfelt thanks to my guide, Prof. O.P. Verma, for his guidance, support, and encouragement during the course of my master study at the Delhi Technological University, Delhi. I am especially indebted to him for teaching me both research and writing skills, which have proven to be beneficial for my current research and future career. Without his endless efforts, knowledge, patience, and answers to my numerous questions, this research would have never been possible. The experimental methods and results presented in this thesis have been influenced by him in one way or the other. I would like to thank sir, for his constructive criticism without which this project would not have been possible. It has been a great honor and pleasure for me to be able to work under his supervision.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Shreya Arora

Roll No. 15/IS/2010

M.Tech (Information Systems)

E-mail: shreya_arora@ymail.com

# Abstract

In the field of Digital Image Processing, noise removal plays a very crucial role. The output of any image processing algorithm will depend greatly on the quality of the image sent as input. Many images during acquisition, transmission and processing get corrupted by impulse noise. Thus impulse noise removal becomes an important pre processing step for image processing.

Impulse noise usually corrupts only some pixels, leaving a few pixels uncorrupted. Therefore impulse noise removal is normally a two step process. The first step involves classifying the image pixels into corrupted and uncorrupted pixels. The second step deals with restoration of these corrupted pixels.

This study introduces a novel approach for impulse noise removal, typically in the range of 10% to 80% noise density. The proposed scheme is a double stage filter, which removes impulse noise based on heuristic calculations of neighboring pixels. In the first stage, "*Detector*", the pixels are identified as noisy or noise-free using distance calculations on the neighboring pixels. An adaptive threshold is used to classify these pixels. Once the pixels are identified to be noisy "*Filtering*" is performed on the noisy pixels. During filtering the image pixels are assigned weight values and the final restoration is done using a weighted median.

In order to evaluate the performance of this proposed filter "*A Novel Approach for Salt and Pepper Noise Removal based on Heuristic Analysis of Neighboring Pixels*" (SPHN), various test images were used. The performance of the SPHN filter is also compared with several other popular techniques. It has been found that not only does the proposed filter work well on a variety of images but also produces results which are significantly better than many popular techniques.

# Contents

# List of figures

# List of Tables

# List of Abbreviations and Symbols

## Abbreviations

| | |
|---|---|
| SPHN | A Novel Approach for Salt and Pepper Noise Removal based on Heuristic Analysis of Neighboring Pixels |
| SPN | Salt and Pepper Noise |
| RVIN | Random Valued Impulse Noise |
| SM | Standard Median Filter |
| CWM | Center Weighted Median Filter |
| ACWM | Adaptive Centre Weighted Median Filter |
| SWM | Switching Median Filter |
| TSM | Tri-State Median Filter |
| DWM | Directional Weighted Median Filter |
| MSWM | Modified Switching Median Filter |
| AM | Adaptive Median Filter |
| PSM | Progressive Switching Median Filter |
| MSM | Multi-State Median Filter |
| PSNR | Peak signal to Noise Ratio |
| MSE | Mean Squared Error |

## Symbols

| | |
|---|---|
| X | Noisy Image |
| Y | Restored Image |
| $\delta_i$ | distance of $i^{th}$ pixel from the central pixel |
| $p_i$ | $i^{th}$ pixel |
| $\xi$ | central pixel in the 3×3 window |
| $\delta_{avg}$ | Average of all the distances $\delta_i$ |
| $\phi$ | Global Maxima of the image |
| $\Diamond$ | Repetition Operator |

# Chapter 1: Introduction

Image processing is one of the most rapidly growing areas in the computer world today. New advances in technology and decrease in the price of mass storage devices have lead to people switching from analog imaging to digital imaging. Classic fields like medicine, film and video production, photography, remote sensing and security monitoring produce huge volumes of data every day.

Digital image processing is primarily concerned with extracting useful information from images. It can be defined as the process of receiving and analyzing visual images by a digital computer [1].

An image may be described as a two-dimensional function *I*.

$$I = f(x, y) \tag{1.1}$$

Where $x$ and $y$ are spatial coordinates. Amplitude of $f$ at any pair of coordinates $(x, y)$ is called intensity $I$ or gray value of the image. When spatial coordinates and amplitude values are all finite, discrete quantities, the image is called *digital image* [2].

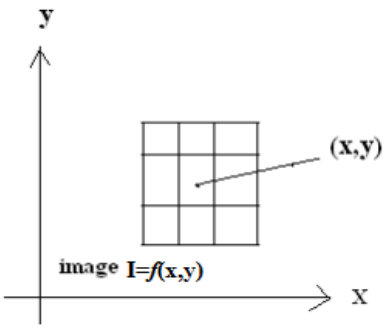



**Fig. 1.1**: Digital Image [2]

A gray scale image is represented as a single 2-D array; however a color image is represented as three 2-D arrays, one for each color(R, G, B).

Digital images often get corrupted, degraded during acquisition, transmission of images. An image thus has to be processed before further use. Digital image processing amongst other areas deals with *image restoration*. Image restoration is the removal or reduction of degradations that are incurred while the image is being obtained [3]. Degradation could be a result of blurring or noise. Blurring is a form of bandwidth reduction of the image caused by the imperfect image formation process such as relative motion between the camera and the original scene or by an optical system that is out of focus [4]. In addition to blurring, the images can be corrupted by noise. There are several techniques that insert noise in images depending on how the image is created. For example: If the image is a scanned photograph, noise can be inserted due to the film grain noise or scanner itself. Noise can also be introduced due to the electronic transmission media or the equipment used for gathering the noise [1].

There are primarily three levels where image processing algorithms are required. The first level is the lowest level; here the data is usually raw unprocessed noisy data. The techniques employed here are usually denoising and edge detection. At the next level are the algorithms which use low level results for further processing like segmentation and edge linking. At the highest level are algorithms which try to extract meaningful information from the data at hand, like optical character recognition and handwriting recognition.

## 1.1 Preliminaries

An image as defined above is a two dimensional function $I = f(x, y)$. Each distinct coordinate in the image is called a picture element or *pixel*. The nature of each element is dependent on the type of image.

### 1.1.1 Types of Images

a. Binary Images

In a binary image, each pixel assumes one of only two discrete values: 0 or 1 interpreted as black and white respectively. A binary image is stored as a logical array of 0s and 1s [5].

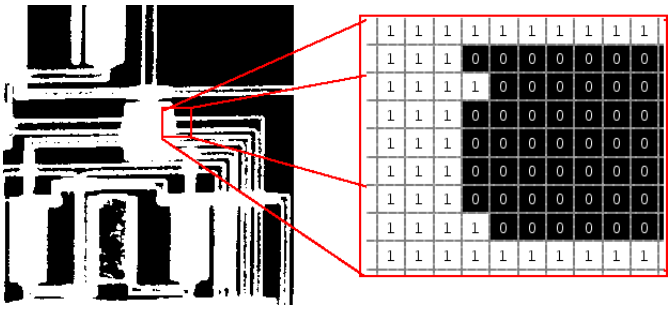



**Fig 1.2**: Pixel Values in a Binary Image [5]

b. Indexed Images

An indexed image consists of an array and a colormap matrix. The pixel values in the array are direct indices into a colormap. By convention, this documentation uses the variable name X to refer to the array and map to refer to the colormap.

The colormap matrix is an m-by-3 array of class double containing floating-point values in the range [0, 1]. Each row of map specifies the red, green, and blue components of a single color. An indexed image uses direct mapping of pixel values to colormap values. The color of each image pixel is determined by using the corresponding value of X as an index into map.

The relationship between the values in the image matrix and the colormap depends on the class of the image matrix. If the image matrix is of class single or double, it normally contains integer values 1 through p, where p is the length of the colormap. The value 1 points to the first row in the colormap, the value 2 points to the second row, and so on. If the image matrix is of class logical, uint8 or uint16, the value 0

points to the first row in the colormap, the value 1 points to the second row, and so on [5].

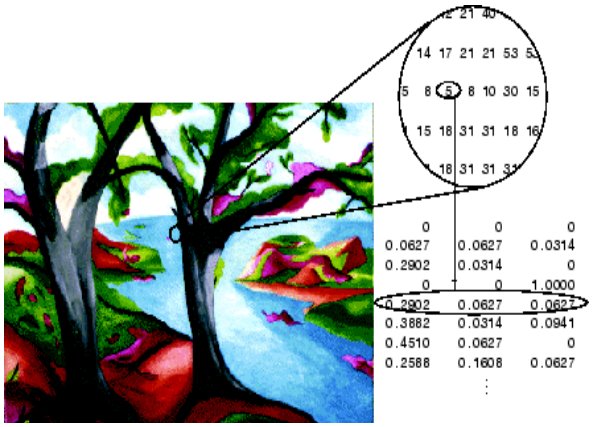



**Fig 1.3**: Indexed Image with colormap entries [5]

c. Grayscale Image

A grayscale image (also called gray-scale, gray scale, or gray-level) is a data matrix whose values represent intensities within some range. MATLAB stores a grayscale image as an individual matrix, with each element of the matrix corresponding to one image pixel.

The matrix can be of class uint8, uint16, int16, single, or double. While grayscale images are rarely saved with a colormap, MATLAB uses a colormap to display them. For a matrix of class single or double, using the default grayscale colormap, the intensity 0 represents black and the intensity 1 represents white [5].

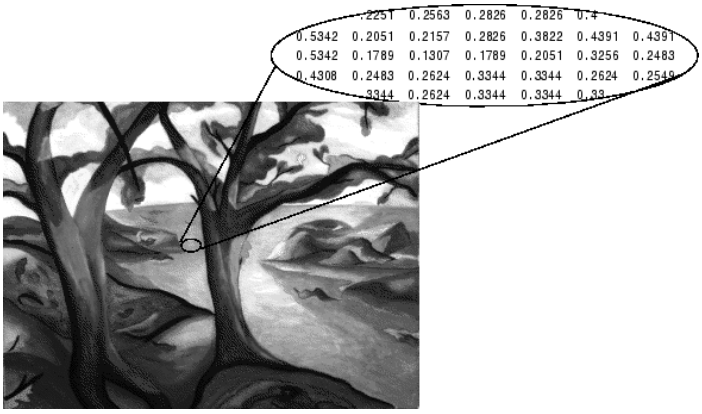



**Fig 1.4**: Grayscale Image with corresponding pixel entries [5]

d. Truecolor Image

A truecolor image is an image in which each pixel is specified by three values — one each for the red, blue, and green components of the pixel's color. MATLAB store truecolor images as an m-by-n-by-3 data array that defines red, green, and blue color components for each individual pixel. Truecolor images do not use a colormap. The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location.

Graphics file formats store truecolor images as 24-bit images, where the red, green, and blue components are 8 bits each. This yields a potential of 16 million colors. The precision with which a real-life image can be replicated has led to the commonly used term truecolor image.

A truecolor array can be of class uint8, uint16, single, or double. In a truecolor array of class single or double, each color component is a value between 0 and 1. A pixel whose color components are (0, 0, 0) is displayed as black, and a pixel whose color components are (1, 1, 1) is displayed as white. The three color components for each pixel are stored along the third dimension of the data array. For example, the red, green, and blue color components of the pixel (10,5) are stored in RGB(10,5,1), RGB(10,5,2), and RGB(10,5,3), respectively [5].


**Fig 1.5**: Truecolor Image with corresponding pixel entries [5]

### 1.1.2 Image Statistics

a. Histogram

The histogram of the gray scale image is a graph indicating the number of times a gray level exists in the image. The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function

$$h(r_k) = n_k \tag{1.3}$$

where

$r_k$ is the kth gray level and

$n_k$ is the number of pixels in the image having gray level $r_k$.

It is common practice to normalize a histogram by dividing each of its values by the total number of pixels in the image, denoted by n.

Thus, a normalized histogram is given by:

$$p(r_k) = n_k / n \quad \text{for k=0, 1, p , L-1} \tag{1.4}$$

Here $p(r_k)$ gives an estimate of the probability of occurrence of gray level $r_k$. The sum of all components of a normalized histogram is equal to 1[2].

A great deal of information can be inferred from the histogram of the image:

- In a dark image, the histogram is cluttered at the lower end
- In a uniformly bright image, the histogram is cluttered at the upper end.
- In a well contrasted image, the histogram is spread out across the intensity levels.

**Fig 1.6** (a): Image **Lena (Dark)**

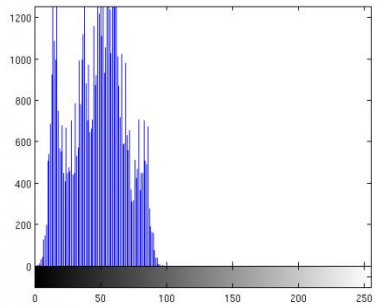



**Fig 1.6** (b): Corresponding Histogram
The peaks are concentrated at the lower end

**Fig 1.7** (a): Image **Lena (Contrast enhanced)**

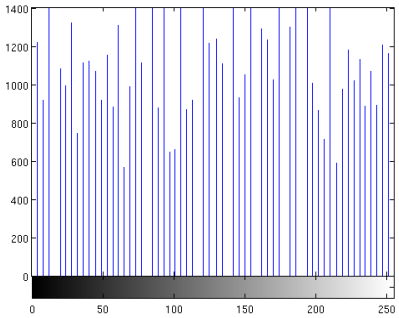



**Fig 1.6** (b): Corresponding Histogram

The peaks are distributed over the entire range

b. Mean

Image mean is the average pixel value of an image. For an image f(x, y) denoted as **f**, the mean $\bar{x}$ can be given by the following equation:

$$\bar{x}[f] = \frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1} f(x,y) \tag{1.5}$$

The mean of a grayscale image is the measure of average brightness or average intensity.

c. Variance and Standard Deviation

The *image variance*, $\sigma^2$ gives an estimate of the spread of pixel values around the image mean. The variance of an image **f** can be calculated using the following:

$$\sigma^2[f] = \bar{x}[f - \bar{x}[f]]^2 \tag{1.6}$$

$$= \frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}(f(x,y) - \bar{x}[f])^2$$

$$= \frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1}\left(f(x,y) - \frac{1}{YX}\sum_{y'=0}^{Y-1}\sum_{x'=0}^{X-1} f(x',y')\right)^2$$

The variance can also be expressed as:

$$\sigma^2[f] = \bar{x}[f^2] - \bar{x}[f]^2 \tag{1.7}$$

$$= \left(\frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1} f(x,y)^2\right) - \left(\frac{1}{YX}\sum_{y=0}^{Y-1}\sum_{x=0}^{X-1} f(x,y)\right)^2$$

The *standard deviation* $\sigma$ is the measure of dispersion of data. Lower value of standard deviation means less dispersion, i.e. values are similar, and a larger value indicates high dispersion. Standard deviation $\sigma$ is simply the square root of variance, i.e. $\sqrt{\sigma^2}$

d. Entropy

The histogram of an image is a probability distribution of pixel values. For a gray scale image with z levels the histogram can be denoted as p (z). The entropy for image f can thus be expressed as:

$$H(f) = -\sum_{z=0}^{Z-1} p(z)\log_2 p(z) \quad bits \tag{1.8}$$

When all intensities have equal frequency, the value of entropy attains its maximum value which is equal to $\log_2 z$ . Entropy measures the information content of an image.

## 1.2 Problem Formulation and Thesis Organization

The basic aim of this thesis is the estimation of uncorrupted image from the corrupted or noisy image without loss of information. This process is also referred to as "denoising". The technique used for denoising plays a major role in the quality of the resultant uncorrupted image. In this thesis a number of popular denoising techniques have been studied and each is implemented in Matlab 7.9[8]. Each method is compared in terms of its visual and quantitative performance. To evaluate the performance of denoising techniques, high quality standard test images such as "Lena", "Cameraman", "Pepper", "Pirate" and "Baboon" were taken, and some noise was added to them. This noisy image became the input to the denoising algorithms. To quantify the performance parameters such as Peak Signal to Noise Ratio (PSNR) and Mean Square Error (MSE) were used.

The rest of the thesis is organized as follows. Chapter 2 discusses the various noise types and presents a survey of various techniques available for denoising Impulse Noise in images. Chapter 3 introduces the proposed impulse noise removal algorithm “A Novel Approach for Salt and Pepper Noise Removal based on Heuristic Analysis of Neighboring Pixels” (SPHN). Chapter 4 presents the comparative study of popular denoising techniques along with the proposed filter (SPHN). It also tabulates the PSNR and MSE of these techniques. It further discusses the future scope of the work presented in the thesis.

# Chapter 2: Noise

In everyday language we use the word noise to mean unwanted sound. In physics and analog electronics noise means any unwanted addition to the signal. In other words noise can be defined as any quantity that disrupts the normal functioning of the system.

Image noise is any random variation in the brightness and intensity information in images. Noise can be introduced in the images during their acquisition, transmission, storage or retrieval. Thus the fundamental problem in image processing is noise removal. The nature of the noise removal algorithm depends on the type of noise in the image.

## 2.1. **Types of Noise in Images**

### 2.1.1. Impulse Noise

Fat-tail distributed or "impulsive" noise is sometimes called salt-and-pepper noise or spike noise [2]. Impulsive noise consists of relatively short duration of “on/off” noise pulses which are caused by a variety of sources, such as switching noise, adverse channel environments in a communication system etc. An impulsive noise filter can be used for enhancing the quality and intelligence of noisy signals, and for achieving robustness in pattern recognition and adaptive control systems [6].

In an image corrupted by impulse noise, some pixels are noisy while others remain unaffected by noise. The pixels are affected by some probability as shown in Eq. 2.1 and 2.2. In case of colored images either all or one or two of its components can be corrupted with impulse noise.

In case of *Salt & Pepper Noise* (SPN) or *Fixed Valued Impulsive Noise*, the noisy pixel value will be equal to either $n_{min}$ (0) or $n_{max}$ (255), whereas for *Random Valued*

*Impulsive Noise* (RVIN) the noisy pixel value can be any value in the range $n_{min}$ to $n_{max}$.

Salt and Pepper noise can be mathematically represented as in 2.1 :

$$y_{i,j} = \begin{cases} zero\ or\ 255 & with\ probability\ p \\ x_{i,j} & with\ probability\ 1\text{-}p \end{cases} \quad (2.1)$$

where $y_{i,j}$ represents the noisy image pixel, $p$ is the total noise density of impulse noise and $x_{i,j}$ represents the uncorrupted image pixel.

Random Valued impulse noise can be mathematically represented as in 2.2:

$$y_{i,j} = \begin{cases} n_{i,j} & with\ probability\ p \\ x_{i,j} & with\ probability\ 1\text{-}p \end{cases} \quad (2.2)$$

where $n_{i,j}$ is the gray level of the noisy image.

Figure 2.1 shows an image (Cameraman) contaminated by 10% Salt and Pepper Noise.

**Fig 2.1(a)**: Original Image (**Cameraman**)


**Fig 2.1(b)**: Image (**Cameraman**) contaminated by 10% Salt and Pepper Noise

## 2.1.2. Gaussian Noise

Gaussian noise is a form of additive noise. Additive noise is governed by the rule in equation 2.3. $w(x,y) = s(x,y) + n(x,y)$ (2.3)

where:

$s(x, y)$ is the original signal, $n(x, y)$ is noise introduced into the original signal to produce corrupted image $w(x, y)$ and $(x, y)$ is the pixel location.

Gaussian noise is evenly distributed over the signal [7]. This means that each pixel in the corrupted image is the sum of its original value and the random Gaussian distributed noise value.

Gaussian noise is so named because the noise follows a Gaussian distribution curve, which is a bell shaped curve given by equation 2.4

$$F(g) = \frac{1}{\sqrt{2\Pi\sigma^2}} e^{-(g-m)^2/2\sigma^2} \qquad (2.4)$$

Where $g$ is the gray level, $m$ is the mean or average of the function and $\sigma$ standard deviation of the noise. The Graph of the Gaussian distribution is a bell shaped curve, as shown in figure 2.2.

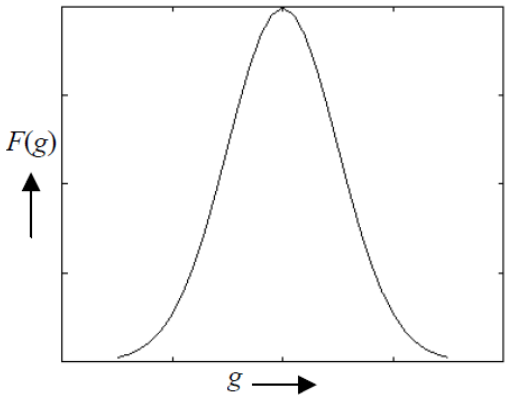



**Fig 2.2**: Gaussian distribution

Figure 2.3 shows image (Cameraman) corrupted by Gaussian noise with zero mean and variance equal to 0.05.

**Fig 2.3(a)**: Original Image (**Cameraman**)

**Fig 2.3(b)**: Image (**Cameraman**) contaminated with Gaussian noise with zero mean and variance 0.05

### 2.1.3. Speckle Noise

*Multiplicative Noise* or *speckle noise* is a signal dependent form of noise whose magnitude is related to the value of the original pixel [8]. Speckle noise is encountered in some imaging applications such as ultrasound imaging and synthesis aperture radar (SAR) imaging. The speckle noise is a signal dependent noise; this means that if the intensity of pixel is high then the noise intensity is also high.

It is given according to the equation 2.5.

$$n_{SN}(t) = \eta(t).S(t) \qquad (2.5\ a)$$

or, $$X_{SN}(m,n) = X(m.n) + \eta(m,n).X(m,n) \qquad (2.5\ b)$$

where $\eta(t)$ is random variable and $S(t)$ is the magnitude of the signal.

An image when corrupted by Speckle Noise appears as shown in figure 2.4.

**Fig 2.4(a)**: Original Image (**Cameraman**)

**Fig 2.4(b)**: Image (**Cameraman**) contaminated with Speckle Noise

## 2.2. **Literature Survey**

Images are often contaminated by a variety of noises. The efficiency of image processing algorithms greatly depends on the quality of the images provided as input. Low quality, corrupted images lead to bad results. Thus noise removal becomes an essential pre processing step for any image processing algorithm. Several noise removal techniques have been proposed in the scientific literature. Impulse noise is one of the most common types of noise found to contaminate images. Impulse noise removal techniques can be broadly classifies into *Linear* and *Non Linear* methods.

Linear Filters are the simplest type of filters; they apply the denoising algorithms to all image pixels irrespective of whether the pixel being processed is noisy or noise-free. Non Linear filters, however, first classify pixels into noisy and noise-free pixels, applying the denoising technique to only the noisy pixels. Thus Non Linear filters generally provide better results. Linear filters cause blurring of fine image details such as textures and edges, due to indiscriminate filtering; non linear filters are thus better suited due to their improved performance in terms of noise removal and edge/ detail preservation.

One of the earliest types of filters proposed is the Moving Average Filter [2], the Mean Filter [2] and the Median Filter [2]. These filters form the basis of several techniques.

- Moving Average Filter[2]

  The Moving Average Filter involves a sliding window of size (2K+1) × (2K+1), where K varies from 1 to n. this window scans the entire image row wise and column wise, each time replacing the value of the central pixel with the average of all the pixels within the window.

$$Y_{i,j} = \frac{1}{(2K+1)^2} \sum_{(u,v)\in S} X_{u,v} \qquad (2.6)$$

where:

X is the noisy image, Y is the restores image and S represents the sliding window. It performs is poorly in terms of both visual and quantitative results.

- Standard Median (SM) Filter[2]

  The Median filter is one of the most popular filters. It is very easy to implement and also relatively affective. The main functioning of the filter is same as the average filter but here the central pixel in the window is replaced by the median of all the neighboring pixels within the window.

  $$Y_{i,j} = Median_{(u,v)\in S}(X_{u,v}) \quad (2.7)$$

  Several types of median filters based on the window size have been implemented. Some of the most common window sizes in use are: (3×3) and (5×5).

However both the averaging filter and the median filter face some serious drawbacks. These filters cause the blurring of edges and other fine image details and are effective at only low noise densities. At noise densities exceeding 50% the output images of these filters show loss of details.

The classical median filter paved a way for the development of several other filters. These modified forms of the median filter attempt to remove the drawbacks faced by the median filter. Some of the popular median based filtering techniques are weighted median (WM) filter [9], center weighted median (CWM) filter [10],adaptive centre weighted median (ACWM) filter[11],switching median [SWM] filter [12], tri-state median (TSM) filter [13], directional weighted median (DWM) filter [14], modified switching median (MSWM) filter [15], adaptive median (AM) filter [16] ,progressive switching median (PSM) filter [17] and multi-state median (MSM) filter [18].

- Weighted Median (WM) Filter[9] and Centre Weighted Median (CWM) Filter[10]

  The WM filter is an extension of the Standard Median filter, in the WM filter Brownrigg modified the SM filter to include certain weighing parameters, based on which the filtering was performed. The CWM filter is an extension of WM filter, which gives more weight to the centre values within the window. The CWM filter allows certain degree of control over the smoothing behavior typical to median filters through the weights that can be set. This technique uses two steps. It first classifies the pixels into noisy and noise-free and in the second step it performs the filtering operation only on the corrupted pixels.

- Adaptive Centre Weighted Median (ACWM) Filter[11]

  It devises a novel adaptive operator, which forms estimates based on the differences between the current pixel and the outputs of centre-weighted median (CWM) [10] filter with varied centre weights. This filter designs an adaptive operator which detects the impulse noise corrupted pixels by using the difference between the outputs of the CWM filter and the current pixel. The filtering operation consists of replacing the corrupted pixel with the median.

- Tri-State Median (TSM) Filter [13]

  This filter has been proposed to enable noise suppression while preserving image details. Tri-State Median filter incorporates both the Standard Median filter [2] and the Centre Weighted Median filter [10] in order to make an informed decision about the pixel under consideration. In order to classify the pixel into corrupted and uncorrupted pixels, the TSM filter compares the outputs of the SM filter and CWM filter with the current pixel to reach a tri-state decision. The switching is controlled by a threshold, which ultimately affects the performance.

- Directional Weighted Median (DWM) Filter [14]

  Directional Weighted Median Filter uses a detector based on the differences between the central pixel and the pixels along the four main directions. It uses the weighted differences in a 5×5 window and finds the minimum value. This minimum weighted directional difference is compared to a fixed threshold based on which the pixel is classified as noisy or noise-free. In the filtering stage it assigns weights to the pixels of the four directions based on the standard deviation of these directions and replaces the corrupted pixel with the weighted median. This algorithm when performed iteratively is suited for highly corrupted images.

- Adaptive Median (AM) Filter [16]

  The adaptive median filter (AMF) [16] uses a varying window size in order to be able to remove noise. It uses the two major decisions, one involving detection of corrupted pixels and the other decision is to check whether the correct value of the median has been reached and hence whether or not to increase the window size. If the value of the median calculated is less than the minimum pixel value or greater than the maximum pixel value within the window, then this median is termed as incorrect and hence the window size is increased. The window size is increased till we obtain the correct median or the upper limit for size is reached.

- Progressive Switching Median (PSM) Filter [17]

  The progressive switching median filter is a two phase filter; in the first phase, detection of corrupted pixels take place using a fixed window size of 3×3. In the second phase the corrupted pixels are replaced by the median value calculated similar to the AM filter [16].

- Multi-State Median (MSM) Filter [18]

  This filter presents a generalized framework of median based switching schemes. It uses threshold to switch among a group of CWM filters [10] having different centre weights.

  Several other filters like "A New Adaptive Switching Median (ASWM) filter [19], Fuzzy Switching Median (FSM) filter [20] and Noise Adaptive Fuzzy Switching Median (NAFSM) Filter [21] are also based on the standard median[2] filter. ASWM proposed by Smail et al, calculates the weighted mean and weighted standard deviation to filter out impulse noise. Fuzzy switching median (FSM) filter [20] and noise adaptive fuzzy switching median (NAFSM) filter [21], use fuzzy reasoning along with SM filter to carry out impulse noise removal. These filters are an improvement over the standard median filter, but have various shortcomings like large computation time, poor denoising in high density noise, loss of image details, blurring of edges etc.

  Apart from these classical filtering techniques, fuzzy filters are another important class of filters. Fuzzy filters are realized by means of simple fuzzy rules that define some type of noise. several fuzzy filters have been proposed, such as Fuzzy Inference Rules by Else action (FIRE) filters by Russo [22]-[26]. Due to good performance of FIRE filters, these have been used by several authors such as Ville [27]. A multilevel fuzzy filter was developed by Russo [28], which involves three cascaded blocks of FIRE Filters. Jiu [29] also proposed a multilevel filter in fuzzy domain.

  A number of other techniques such as those that utilize histograms of the images have also been proposed [30]-[33]. Another class of filters [34]-[37], use neural networks along with fuzzy rules for the purpose of denoising.

There are a number of techniques available for the removal of impulse noise in images, but the purpose here is not to provide an exhaustive list of all techniques available but to provide a brief introduction of some of them. The selection of the technique depends on the characteristics of the image processing algorithm to be followed. The selection can be based on a variety of factors such as the quantitative measures like Peak signal to Noise Ratio (PSNR) and Mean Squared Error (MSE), the time taken by the technique to produce the desired results, the noise density of the corrupted image.

## 2.3. Performance Measures

The metric used for evaluating and comparing the performances of different filters are defined below:

**a. Mean Squared Error (MSE) :**

In statistics, the mean squared error (MSE) of an estimator is one of many ways to quantify the amount by which an estimator differs from the true value of the quantity being estimated.

For image processing, it is the cumulative squared error between the restored and the original image. A lower value of MSE means lesser error. It can be defined as in 2.8.

$$MSE(f, I) = \frac{\sum_{z=1}^{3}\sum_{x=1}^{L}\sum_{y=1}^{M}\left[I(x,y,z) - f(x,y,z)\right]^2}{3 \times L \times M} \qquad (2.8)$$

Where

$L, M$ are the Image dimensions

$I(x, y, z)$ = the pixel values of restored image

$f(x, y, z)$ = the pixel values of original image

**b. Peak Signal To Noise Ratio (PSNR) :**

PSNR is the measure of peak error and estimates the quality of the reconstructed image with respect to the original image. A higher value of PSNR indicates better reconstruction. It can be mathematically defined as in 2.9:

$$PSNR(f, I) = 10 \log\left(\frac{1}{MSE(f, I)}\right) \tag{2.9}$$

**c. Subjective or Qualitative Measure :**

Along with the quantitative measure namely MSE and PSNR, it is important to take under consideration the subjective performance i.e. the image quality performance measure subjective assessment is also required. The visual quality is related to the preference and judgement of the observer or the performance of an operator for some specific task. However perceptual quality evaluation is not a deterministic process.

# Chapter 3: The Proposed Algorithm for Impulse Noise Removal

Noise Removal is an important aspect of image processing. Impulse noise occurs due to errors in the communication channel, and hence is one of the most commonly found noise types afflicting images. The most important challenge faced when denoising an image corrupted by impulse noise is to preserve the image details and prevent blurring. Impulse noise, unlike Gaussian noise is not evenly distributed over the image. Thus, in an image corrupted by impulse noise, we can find several corrupted pixels along with several uncorrupted ones. Thus it is of paramount importance to not only be able to filter out the noise but to be able to detect pixels corrupted by noise.

A number of impulse noise removal techniques have already been proposed in the scientific community, some of which were described in the previous chapter. This chapter presents a new algorithm for removal of impulse noise in images "**A Novel Approach for Salt and Pepper Noise Removal based on Heuristic Analysis of Neighboring Pixels**" **(SPHN)**. This scheme works in two stages, the first stage identifies the noisy and noise-free pixels using the distance measures. The second stage restores the noisy pixel by first assigning weights to the pixels and finally replacing the noisy value with weighted median.

## 3.1The Proposed Technique

SPHN is a double stage filter; the first stage identifies the noisy and noise-free pixels, the second stage performs filtering only on the noisy pixels. Not subjecting the noise-free pixels to filtering process helps preserve fine details and avoids blurring. The details of the two stages are as follows:

A. *Detection Stage:*

In the detection stage the corrupted image X is subjected to a 3×3 window $W_{i,j}$ centered at pixel $x_{i,j}$. The current pixel is compared to the maximum and minimum values $l_{max}$ and $l_{min}$ respectively within the current window; if the pixel is equal to either of the two values it is a noisy candidate.

$$x_{i,j} = \begin{cases} noise-free & if\ l_{\min} < x_{i,j} < l_{\max} \\ noise\ candidate & if\ x_{i,j} = l_{\min}\ or\ x_{i,j} = l_{\max} \end{cases} \quad (3.1)$$

where:

$l_{max}$ = local maxima within the current window

$l_{min}$ = local minima within the current window

Once identified as a noise candidate it is important to ascertain that the pixel is an isolated maximum or minimum value i.e. noise, and not a part of any edges. To do this, calculate the distance of all pixels in the window with the central pixel as below

$$\delta_i = abs(p_i - \xi) \quad (3.2)$$

where:

$\delta_i$ = distance of $i^{th}$ pixel from the central pixel

$p_i$ = $i^{th}$ pixel

$\xi$ =central pixel in the 3×3 window

Next we calculate the average $\delta_{avg}$ of all the distances $\delta_i$

By comparing this average distance $\delta_{avg}$ with the global maxima $\phi$, we can classify are pixel into noisy or noise free according to equation 3.3

$$x_{i,j} = \begin{cases} Noisy\ Pixel & if\ \delta_{avg} \geq 90\%\ of\ (\phi/2) \\ Noise-Free & otherwise \end{cases} \quad (3.3)$$

If according to Eq. 3.3 pixel is noisy, we process it further.

The next step is to find the value of pixels $p_1$ *and* $p_2$ in the neighborhood $\xi$ having the least distance $\delta_i$ (from $\xi$) and calculate their average $\rho_{avg}$.

Finally we classify the pixel as noisy or noise-free according to equation 3.4.

$$x_{i,j} = \begin{cases} Noisy\ Pixel & if\ \rho_{avg} \geq 90\%\ of\ (\phi / 2) \\ Noise-Free & otherwise \end{cases} \quad (3.4)$$

If the central pixel is noisy it undergoes the filtering stage, else if it is noise-free no further processing is needed.

B. *Filtering Stage*:

When the noisy pixels are identified, they are subjected to filtering.

In the filtering stage the first step is to calculate the mean $\eta_i$ of pixels in all directions $\eta_1$ (L to R), $\eta_2$ (T to B), $\eta_3$ (TL to BR) and $\eta_4$ (TR to BL).

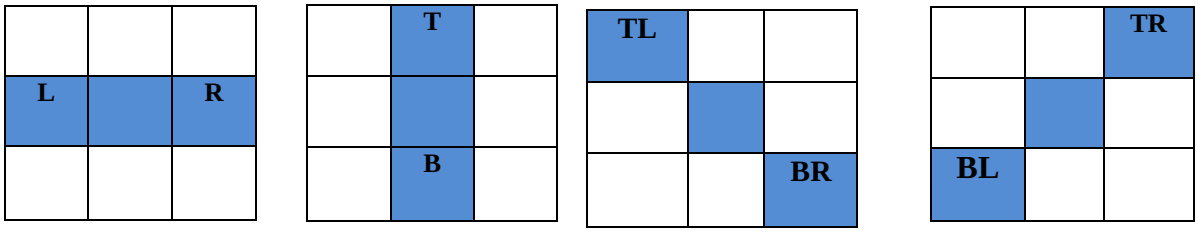



**Fig. 3.1**: The direction used to calculate the distances.

Next the distance of each pixel $p_i$ from its mean $\eta_i$ is calculated. Now the average of the distances $\eta_{avg_i}$ for each direction i as shown in figure 3.1 is calculated.

The next step is to assign weights ($w_1$ *to* $w_8$). Pixels in the direction with minimum $\eta_{avg_i}$ gets highest weight (4). Pixels in the direction of second minimum $\eta_{avg_i}$ gets weight 3. Similarly we assign weights 2 and 1 to directions with 3rd and 4th minimum $\eta_{avg_i}$ respectively.

After assigning the weights the restoration of the corrupted pixels is done according to Eq. 3.5

$$y_{i,j} = \begin{cases} median\{w_i \lozenge x_{i,j}\} & if\ x_{i,j}\ is\ noisy \\ x_{i,j} & if\ x_{i,j}\ is\ noise-free \end{cases} \quad (3.5)$$

where:

$w_i$ = weight assigned to the pixel

$\lozenge$ = repetition operator

## 3.2 Algorithm

The proposed filter (SPHN) has been presented in the previous section; it can be summarized as below:

*Detection Stage:*

1. Subject the corrupted image X to a 3×3 window $W_{i,j}$ centered at pixel $x_{i,j.}$
2. Compare the current pixel $x_{i,j}$ to the maximum and minimum values $l_{max}$ and $l_{min}$ respectively within the current window; the pixel is classified as a noise candidate or a noise-free pixel according to equation (3.1).
3. Now in order to ascertain that the noise candidate is indeed a noisy pixel and not an edge pixel, calculate the distance of all pixels in the window with the central pixel according to Eq. 3.2
4. The next step is to calculate the average $\delta_{avg}$ of all the distances $\delta_i$.
5. In order to segregate noise-free pixels from noisy pixels we compare the average distance $\delta_{avg}$ with the global maxima $\phi$. The classification of noisy and noise free pixels is done according to Eq. 3.3. Further processing is done on the noisy pixel.
6. The next step is to find the value of pixels $p_1$ *and* $p_2$ in the neighborhood $\xi$ having the least distance $\delta_i$ (from $\xi$) and calculate their average $\rho_{avg}$.
7. Finally the pixel is classified as noisy or noise-free according to equation 3.4.

The pixels identified as noisy pixels are subjected to the filtering stage.

*Filtering Stage:*

1. In the filtering stage the first step is to calculate the mean $\eta_i$ of pixels in all directions $\eta_1$ (W to E), $\eta_2$ (N to S), $\eta_3$ (NW to SE) and $\eta_4$ (NE to SW).
2. Next the distance of each pixel $p_i$ from its mean $\eta_i$ is taken and corresponding average $\eta_{avg_i}$ calculated as shown in figure 3.1 is calculated.
3. Next the weights ( $w_1$ *to* $w_8$ ) are assigned. Pixels in the direction with minimum $\eta_{avg_i}$ gets highest weight (4). Pixels in the direction of second minimum $\eta_{avg_i}$ gets weight 3. Similarly we assign weights 2 and 1 to directions with 3rd and 4th minimum $\eta_{avg_i}$ respectively.
4. After assigning the weights the restoration of the corrupted pixels is done according to Eq. 3.5

Section 3.3 illustrates the algorithm with the help of an example and section 3.4 gives its detailed Flow Chart.

## 3.3 Illustration

### 3.3.1 The Detector

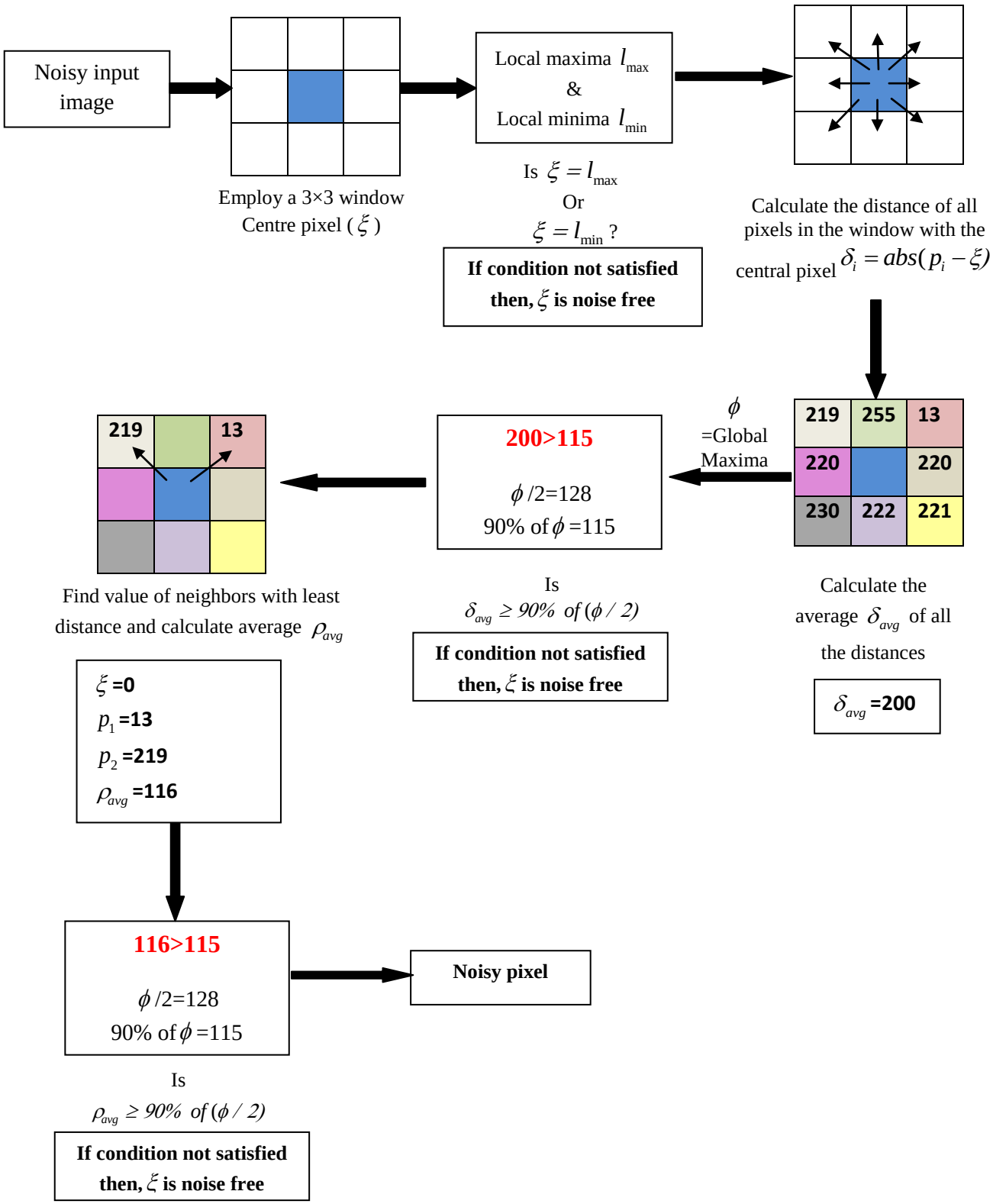
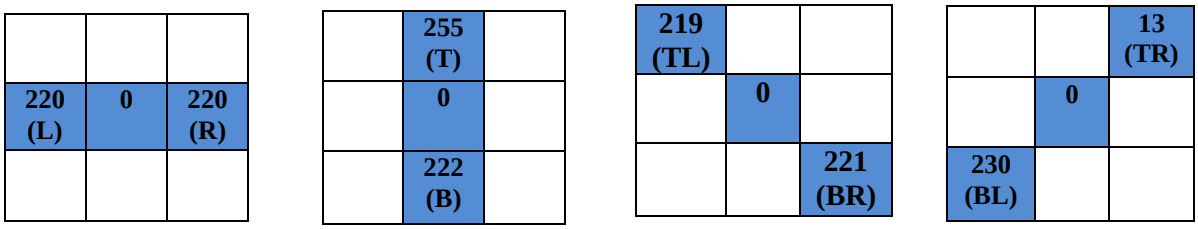



Fig 3.2: Illustration of the proposed algorithm.

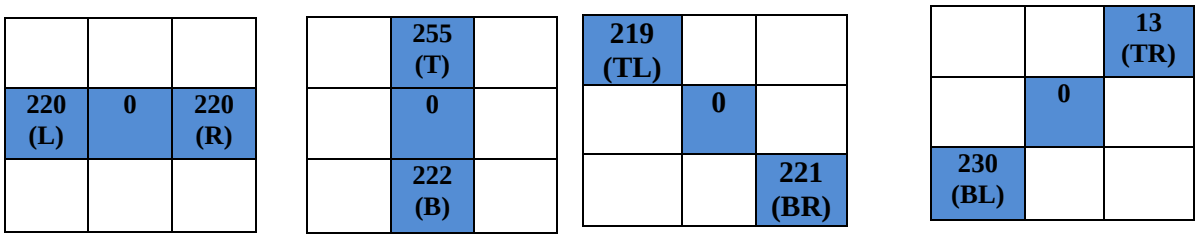
a) The detector

### 3.3.4 The Filter



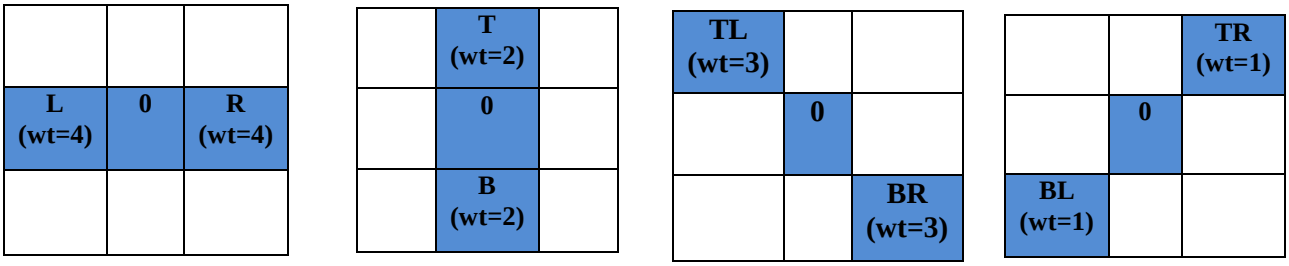

**Calculate the mean of pixels in all directions.**

$$\eta_1 = (220 + 220)/2 = 220$$
$$\eta_2 = (255 + 222)/2 = 239$$
$$\eta_3 = (219 + 221)/2 = 220$$
$$\eta_4 = (230 + 10)/2 = 120$$




**Calculate distance of every pixel from its corresponding mean and take its average $\eta_{avg_i}$**

$$\eta_{avg1} = ((220 - 220) + (220 - 220))/2 = 0$$
$$\eta_{avg2} = ((255 - 239) + (239 - 222))/2 = 17$$
$$\eta_{avg3} = ((220 - 219) + (221 - 220))/2 = 1$$
$$\eta_{avg4} = ((120 - 10) + (230 - 120))/2 = 115$$




**Pixels in direction with minimum $\eta_{avg}$ gets the highest weight**

For pixels
L,R; wt=4
TL, BR; wt=3
T, B; wt=2;
TR, BL; wt=1

**Finally using these weights; restore the pixel using weighted median.**

Fig 3.2: Illustration of the proposed algorithm.
b) The filter

## 3.4 Flow Chart for the Proposed Algorithm

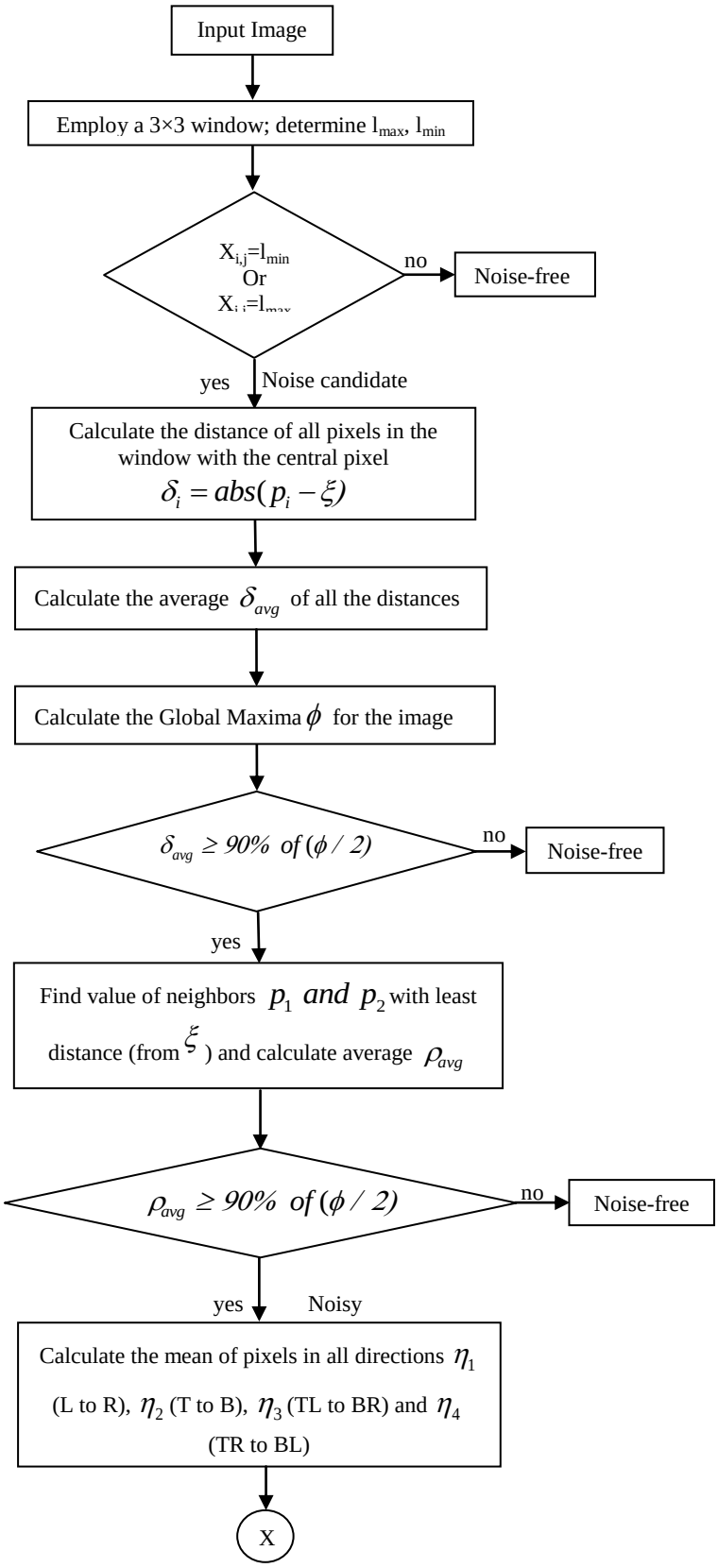



**Fig 3.3**: Flow chart of the proposed algorithm (SPHN)

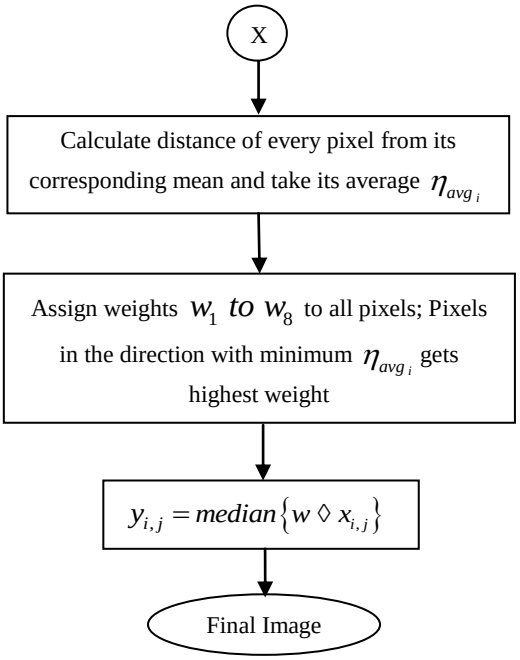



**Fig 3.3** contd.: Flow chart of the proposed algorithm (SPHN)

Where:

$l_{max}$ = local maxima in the current window

$l_{min}$ = local minima in the current window

$\xi$ = central pixel of the current window

$\delta_i$ = distance of pixel I from central pixel

$\eta_1, \eta_2, \eta_3, \eta_4$ = mean of pixels in all directions

$w_1$ to $w_8$ = weights assigned to pixels

$p_i$ = $i^{th}$ pixel

$\delta_{avg}$ = average of all distance values

$\phi$ = global maxima for the image

$\rho_{avg}$ = average of pixels nearest to central pixel $\xi$

$\eta_{avg}$ = distance average

$\lozenge$ = repetition operator

## 3.4 Summary

This chapter presents the proposed SPHN algorithm. This two stage algorithm uses distance calculations to identify noisy pixels. It then assigns weights to the pixels and finally restores them using the weighted median.

The advantage of this technique lies within the fact that it does not require any prior calculations. It uses an adaptive threshold for classifying pixels or restoring them. Moreover the algorithm is easy to understand and implement.

The next chapter discussed the quantitative and qualitative results for the proposed algorithm It also compares the SPHN algorithm with existing techniques.

# Chapter 4: Simulations, Results and Comparisons

The previous chapter presents a new technique for impulse noise removal based on heuristic analysis of neighboring pixels. The algorithm uses distance calculations to identify noisy pixels. It then assigns weights to the pixels and finally restores them using the weighted median.

To evaluate the performance of the algorithm, high quality standard test images such as "Lena", "Cameraman", "Pepper", "Pirate" and "Baboon" were taken, and some noise was added to them. This noisy image became the input to the denoising algorithm. To quantify the performance, parameters such as Peak Signal to Noise Ratio (PSNR) and Mean Square Error (MSE) are used. The PSNR and MSE values at noise densities ranging from 10% to 90% for all the test images have been tabulated in table 4.1.

**Table 4.1:** PSNR and MSE values for various test images at varying noise intensities for the proposed algorithm (SPHN)

| **Type of image** | **Noise in %** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **10** | **20** | **30** | **40** | **50** | **60** | **70** | **80** | **90** |
| **PSNR(dB)** | | | | | | | | | |
| **Peppers** | 87.84 | 80.55 | 77.60 | 70.95 | 69.00 | 66.91 | 64.40 | 65.17 | 65.54 |
| **Lena** | 72.01 | 71.65 | 69.70 | 68.95 | 66.26 | 64.03 | 62.68 | 62.92 | 63.33 |
| **Baboon** | 77.13 | 76.29 | 69.98 | 64.90 | 61.97 | 60.42 | 59.70 | 58.52 | 60.01 |
| **Pirate** | 85.16 | 75.62 | 71.06 | 68.27 | 65.85 | 64.58 | 61.47 | 61.53 | 63.26 |
| **Cameraman** | 69.56 | 67.42 | 66.78 | 65.93 | 65.67 | 65.15 | 63.70 | 63.71 | 63.72 |
| **MSE** | | | | | | | | | |
| **Peppers** | 0.01 | 0.05 | 0.07 | 0.08 | 0.09 | 0.13 | 0.23 | 0.19 | 0.18 |
| **Lena** | 0.03 | 0.04 | 0.07 | 0.08 | 0.15 | 0.25 | 0.35 | 0.33 | 0.35 |
| **Cameraman** | 0.07 | 0.14 | 0.17 | 0.18 | 0.19 | 0.19 | 0.27 | 0.25 | 0.27 |
| **Baboon** | 0.01 | 0.02 | 0.03 | 0.05 | 0.07 | 0.09 | 0.13 | 0.12 | 0.13 |
| **Pirate** | 0.07 | 0.08 | 0.09 | 0.13 | 0.16 | 0.22 | 0.46 | 0.45 | 0.47 |

Higher the value of PSNR the better is the restoration performance. Similarly a lower value of MSE indicates lower error rate.

As can be observed from table 4.1, the proposed SPHN algorithm works well at both high and low noise densities. It has the highest value of PSNR equal to 87.84 dB for image "Peppers" at 10% noise. At 10% noise the lowest PSNR of 69.56 dB is observed for image "Cameraman" From noise density equal to 90 % the PSNR value for "Peppers" is 65.54 and for "Cameraman" is 63.72 dB. For other images "Baboon", "Lena" and "Pirate" the PSNR values at 90% noise are 60.01, 63.33 and 63.26 respectively.

The lowest value of MSE equal to 0.01 is observed at 10% noise for image "Peppers". For image "Pirate" and "Cameraman" the MSE value is 0.07. This value increases to 0.18 for image "Peppers" and 0.47 for "Pirate" and to 0.27 for "Cameraman". The MSE values for images "Lena" and "Baboon" vary from 0.03 and 0.01 respectively at 10% noise to 0.30 and 0.13 respectively at 90% noise

Thus it can be concluded that even for a varied selection of images, the proposed algorithm works well at both high and low noise densities.

The following pages present the noisy as well as denoised images at different noise densities. Fig 4.1 presents the original test images used. Fig 4.2 shows the test image "Cameraman" at 30 % noise and the corresponding denoised image. Fig 4.3 and Fig 4.4show "Cameraman" at 40% and 50% noise respectively, along with the denoised image. Fig 4.5, 4.6 and 4.7 show image "Peppers" at 30%, 40% and 50% noise along with denoised images. Fig 4.8, 4.9, 4.10, 4.11, 4.12 and 4.13show images "Lena" and "Pirate" corrupted by 30%, 40% and 50% noise respectively, along with the denoised image. Fig 4.14, 4.15 and 4.16 show the noise corrupted and denoised image for "Baboon".


(a)


(b)


(c)


(d)

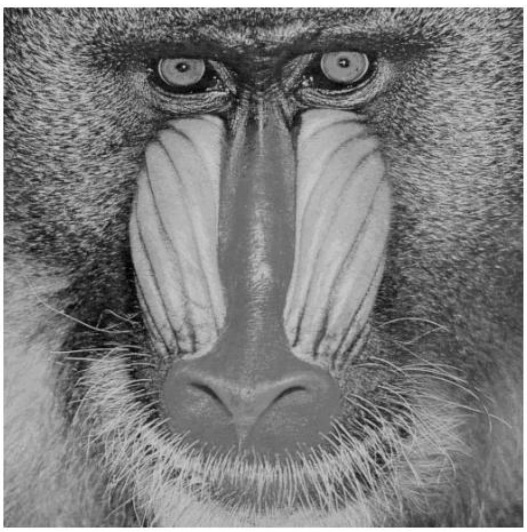

(e)

**Fig 4.1**: Test images (a) Cameraman (b) Peppers (c) Pirate (d) Lena (e) Baboon

**Fig 4.2**: Image (a) Cameraman corrupted with 30% noise (b) Corresponding denoised image

**Fig 4.3**: Image (a) Cameraman corrupted with 40% noise (b) Corresponding denoised image

**Fig 4.4**: Image (a) Cameraman corrupted with 50% noise (b) Corresponding denoised image

**Fig 4.5**: Image (a) Peppers corrupted with 30% noise (b) Corresponding denoised image

**Fig 4.6**: Image (a) Peppers corrupted with 40% noise (b) Corresponding denoised image

**Fig 4.7**: Image (a) Peppers corrupted with 50% noise (b) Corresponding denoised image

**Fig 4.8**: Image (a) Pirate corrupted with 30% noise (b) Corresponding denoised image

**Fig 4.9**: Image (a) Pirate corrupted with 40% noise (b) Corresponding denoised image

**Fig 4.10**: Image (a) Pirate corrupted with 50% noise (b) Corresponding denoised image

**Fig 4.11**: Image (a) Lena corrupted with 30% noise (b) Corresponding denoised image

**Fig 4.12**: Image (a) Lena corrupted with 40% noise (b) Corresponding denoised image

**Fig 4.13**: Image (a) Lena corrupted with 50% noise (b) Corresponding denoised image

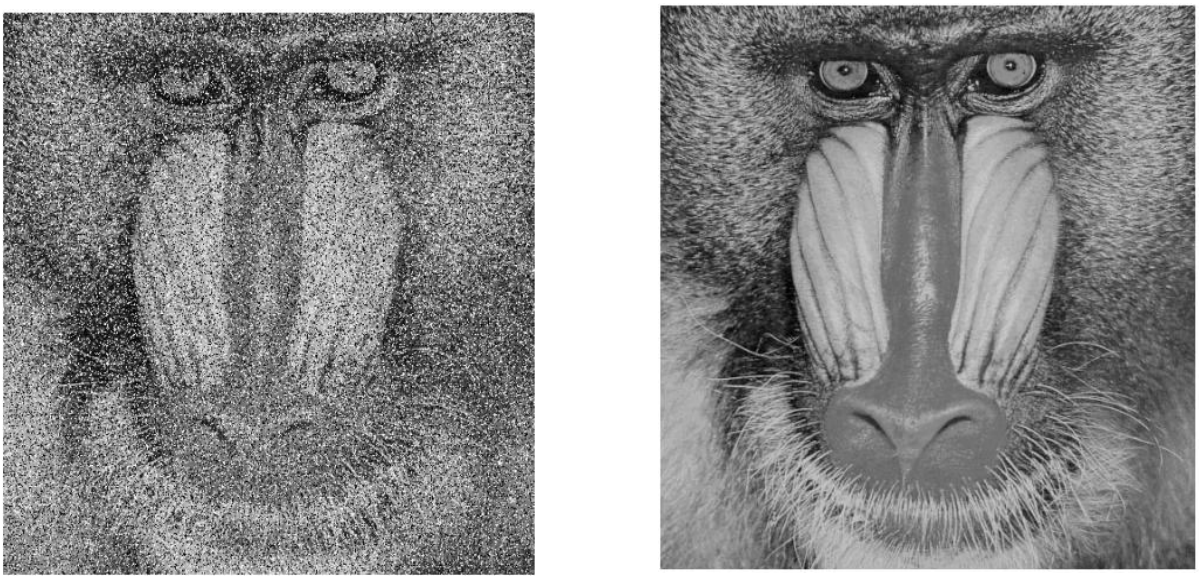


**Fig 4.14**: Image (a) Baboon corrupted with 30% noise (b) Corresponding denoised image

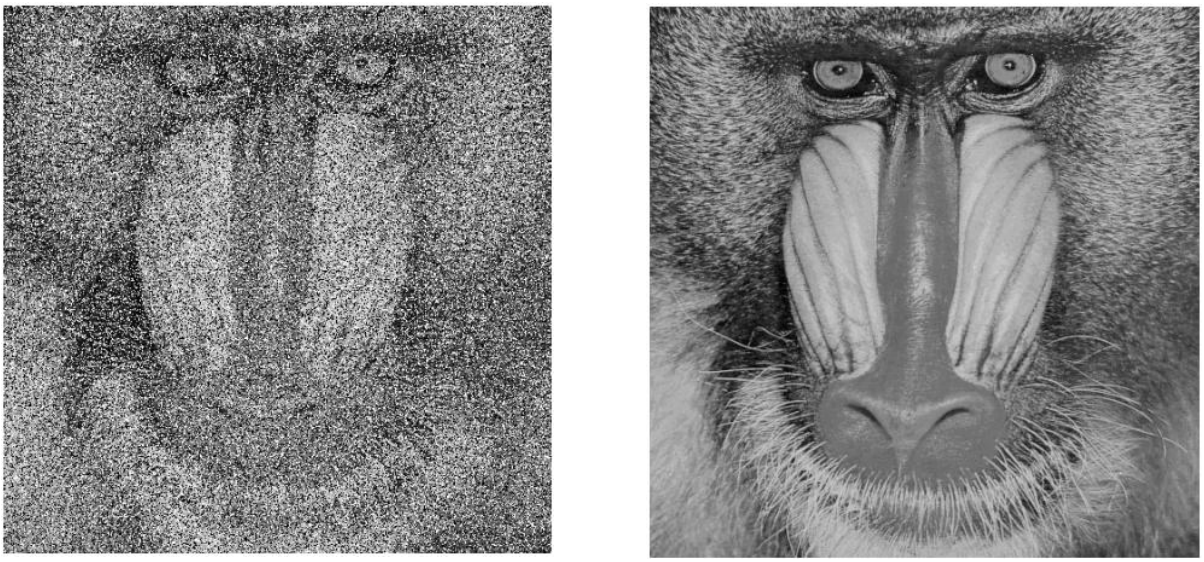


**Fig 4.15**: Image (a) Baboon corrupted with 40% noise (b) Corresponding denoised image

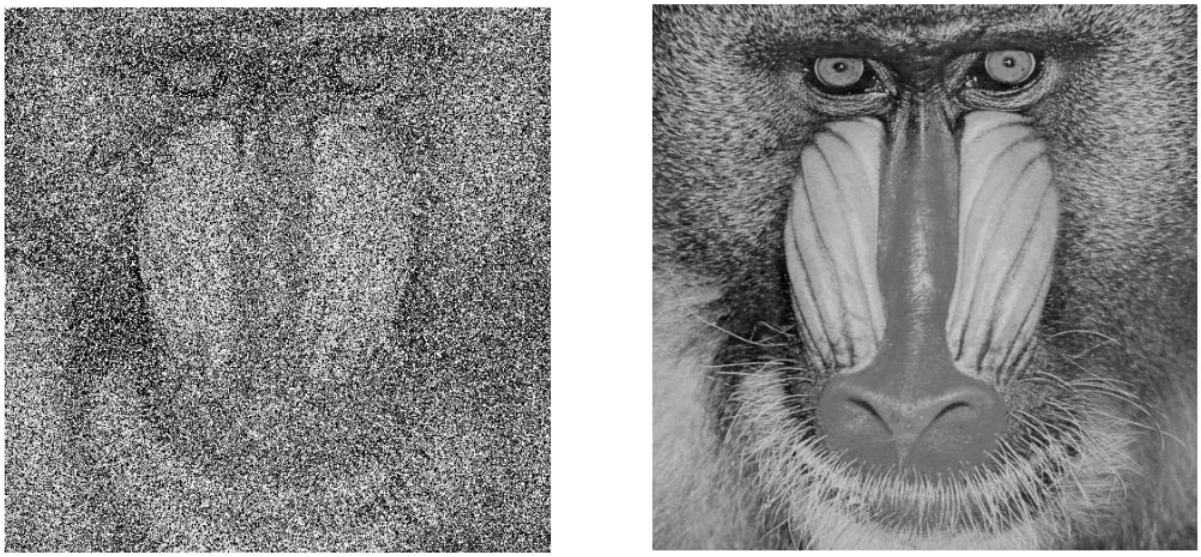


**Fig 4.16**: Image (a) Baboon corrupted with 50% noise (b) Corresponding denoised image

For the purpose of comparison the several standard techniques such as SDOD [39], ASWM [19], SM [2] filter, CWM [10] filter, CEF[41] and EEPA [40] were implemented in Matlab 7.9[8], the image "Lena" was provided as input and the outputs were observed.

The PSNR and MSE values of the SPHN filter are compared against the existing algorithms such as SDOD [39], ASWM [19], SM [2] filter, CWM [10] filter, CEF[41] and EEPA [40] by varying the noise density from 10% to 90% and are shown in Table 4.2 and Table 4.3.

**Table 4.2**: Comparisons of Restoration Results in PSNR (dB) for Seven Reference Algorithms Corrupted By Varying Noise Intensities for Test Image "Lena"

| **Noise in %** | **PSNR(dB)** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SDOD** | **ASWM** | **SM** | **CWM** | **CEF** | **EEPA** | **SPHN** |
| **10** | 25.83 | 33.9 | 36.50 | 36.98 | 39.05 | 41.26 | **72.01** |
| **20** | 25.60 | 33.57 | 34.53 | 35.23 | 37.56 | 38.95 | **71.65** |
| **30** | 25.22 | 33.02 | 31.42 | 32.45 | 35.42 | 37.39 | **69.70** |
| **40** | 25.00 | 32.22 | 28.08 | 29.12 | 32.87 | 35.03 | **68.95** |
| **50** | 25.00 | 31.29 | 25.51 | 26.23 | 31.95 | 34.79 | **66.26** |
| **60** | 24.45 | 27.95 | 22.01 | 23.33 | 31.15 | 34.23 | **64.03** |
| **70** | 24.32 | 29.26 | 20.51 | 20.14 | 29.90 | 33.45 | **62.68** |
| **80** | 24.16 | 28.27 | 17.16 | 18.67 | 28.75 | 32.36 | **62.93** |
| **90** | 24.08 | 27.36 | 12.01 | 13.21 | 27.63 | 31.45 | **63.33** |

**Table 4.3**: Comparisons of Restoration Results in MSE for Seven Reference Algorithms Corrupted By Varying Noise Intensities for Test Image "Lena"

| Noise in % | MSE | | | | | | |
|---|---|---|---|---|---|---|---|
| | **SDOD** | **ASWM** | **SMF** | **CWMF** | **CEF** | **EEPA** | **SPHN** |
| **10** | 89.90 | 26.26 | 26.12 | 23.78 | 25.20 | 30.60 | **0.03** |
| **20** | 90.23 | 28.53 | 45.34 | 36.12 | 26.60 | 32.55 | **0.04** |
| **30** | 92.55 | 32.40 | 90.23 | 78.34 | 27.80 | 37.42 | **0.07** |
| **40** | 95.05 | 38.96 | 118.78 | 120.53 | 29.41 | 45.67 | **0.08** |
| **50** | 96.12 | 48.20 | 235.65 | 250.68 | 29.79 | 57.18 | **0.15** |
| **60** | 96.78 | 104.1890 | 312.45 | 345.15 | 30.01 | 70.60 | **0.25** |
| **70** | 97.34 | 77.0421 | 500.89 | 511.78 | 30.76 | 85.54 | **0.35** |
| **80** | 98.23 | 96.6521 | 825.67 | 789.68 | 31.13 | 100.53 | **0.33** |
| **90** | 98.65 | 119.2638 | 875.12 | 890.50 | 31.78 | 112.26 | **0.35** |

From the Tables 4.3 and 4.4, it is observed that the performance of the proposed algorithm (SPHN) is better than the existing algorithms at both low and high noise densities.

The highest PSNR in the table 72.01 dB at 10% noise is for SPHN algorithm. The next in line is 41.26 dB for EEPA. The proposed filter has the lowest MSE value of 0.03 at 10% noise. This value increases to 0.35 at 90% noise. It can thus be concluded that the propose SPHN filter works well on high and low noise density images.

A plot of PSNR and MSE values against noise densities for Lena" image is shown in Fig. 4.17.

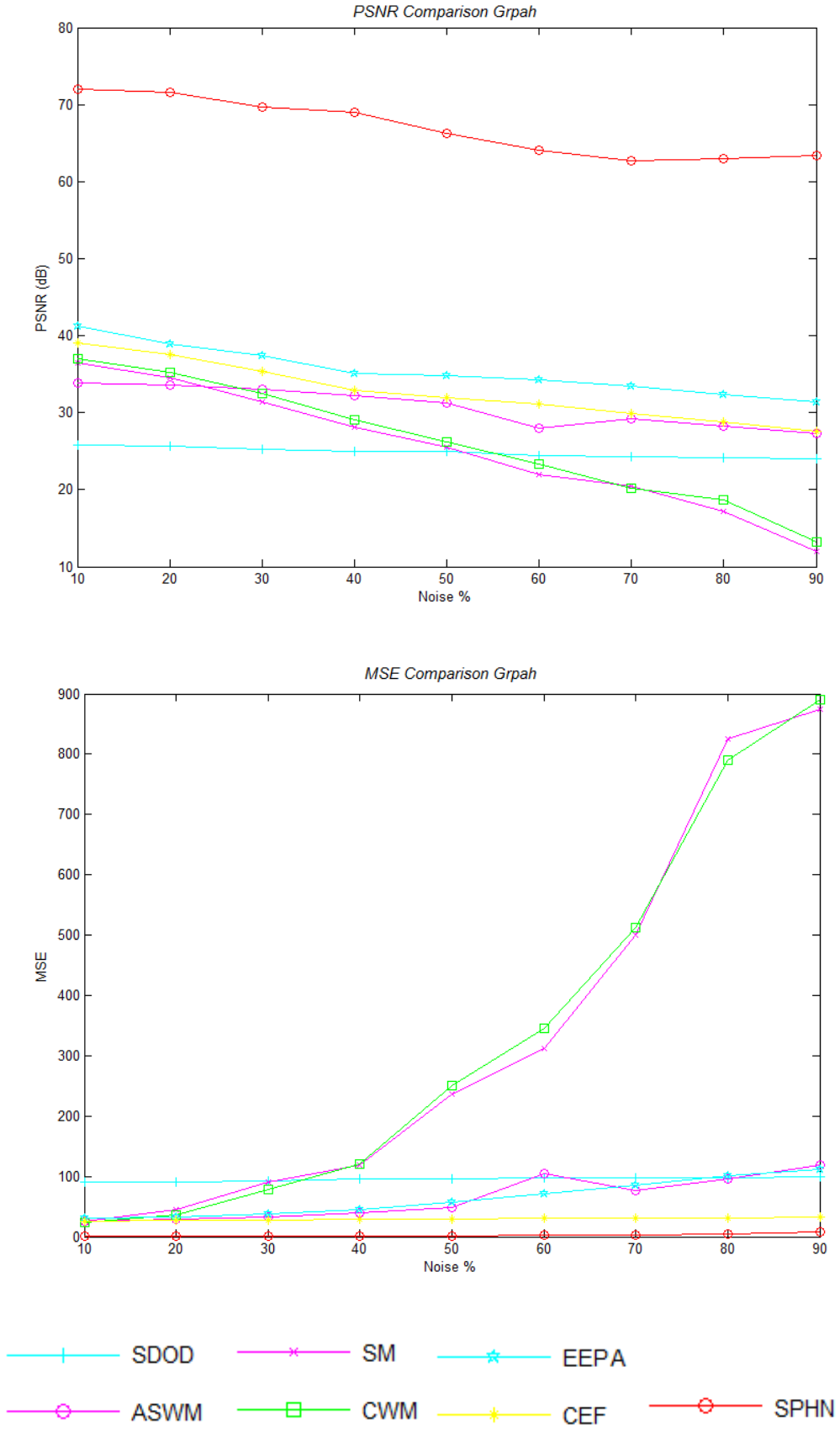



**Fig.4.17** Comparison graph of PSNR and MSE at different noise densities for '**Lena'** image

In order to explore the visual quality, the reconstructed images of different denoising methods in restoring 30% corrupted image “Lena” is shown in Fig: 4.18


(a) (b) (c)

(d) (e) (f)

(g) (h)

**Fig 4.18**: Restoration results of different methods in restoring corrupted image “Lena.” (a) corrupted image with 30% impulse noise, (b) SDOD, (c) ASWM (d) SM (e) CWM (f) CEF (g) EEPA (h) SPHN

We can see from the plots of PSNR and MSE with Noise % that the SPHN filter works better than the already existing techniques. Thus observing the visual and quantitative results we can come to a conclusion that the proposed filter works better from many existing techniques.

# Chapter 5: Conclusions and Future Work

The basic aim of this thesis is to understand the concept of impulse noise removal and to design an algorithm which works better than existing techniques. Impulse noise removal is an important area in the field of digital image processing. It is an important pre processing step to implement any image processing algorithm.

The first chapter introduces the concept of image processing. It explains in depth about the types of images, and the operations that can be applied to the images. The second chapter introduces the concept of noise. It tells about the different types of noise that can affect an image. Concentrating on impulse noise removal, it enlists a variety of existing techniques that deal with this area. Next it tells about the performance measures that can be used to quantitatively study the efficiency of the techniques.

The third chapter presents the proposed filter "**A Novel Approach for Salt and Pepper Noise Removal based on Heuristic Analysis of Neighboring Pixels**" (SPHN). It also gives its algorithm and flow chart. It is a two step algorithm, which uses the distance calculations of the neighborhood in order to classify noisy and noise-free pixels. In the filtering stage, it assigns weights to the pixels according to some calculations and an adaptive threshold. It then filters the noisy pixels using the weighted median.

To analyze the performance the algorithm was implemented using Matlab 7.9[8] and a number of test images namely "Cameraman", "Peppers", "Pirate", "Lena" and "Baboon" were used. The fourth chapter analyzes the performance of the proposed filter on various images at varying noise levels and also compares it with many existing filters. It concludes that the proposed filter works well at both low and high noise densities and also in comparison to some popular techniques.

This thesis is limited to applying the algorithm on images in gray scale only. In future the algorithm can also be implemented on color images. Future work can be extended to reduce the false and miss detection count to improve the detector capability.

# References


[1]. B. Chanda and D. Dutta Majumder. Digital Image Processing and Analysis. Prentice-Hall of India, 1st edition, 2002.

[2]. R. C. Gonzalez and R. E. Woods. Digital Image Processing. Addison Wesley, 2nd edition, 1992.

[3]. Castleman Kenneth R, *Digital Image Processing*, Prentice Hall, New Jersey, 1979.

[4]. Reginald L. Lagendijk, Jan Beimond, *Iterative Identification and Restoration of Images*, Kulwer Academic, Boston, 1991.

[5]. Mathworks, 2012, Image Types in the Toolbox

[6]. Saeed V. Vaseghi. Advanced Digital Signal Processing and Noise Reduction. 2nd Edition. John Wiley & Sons Ltd.2000

[7]. Scott E Umbaugh, *Computer Vision and Image Processing*, Prentice Hall PTR, New Jersey, 1998.

[8]. Matlab 7.9, "Matlab," http://www.mathworks.com/, 2009.

[9]. D.R.K.Brownrigg "The Weighted median filter," *Commun*.ACM, vol.27, no.8, pp.807-818, Aug 1984.

[10]. T. Chen and H. R. Wu. Adaptive impulse detection using center-weighted median filters. IEEE Signal Process. Lett., 8(1):1–3, January 2001.

[11]. S. J. KO and Y. H. Lee, "Center weighted median filters and their applications to image enhancement," IEEE Trans. Circuits Syst., vol. 38, pp. 984–993, 1991.

[12]. T. Sun and Y. Nuevo, "Detail preserving median based filters in image processing," Pattern Recognition. Lett, vol. 15, pp. 341–347, 1994.

[13]. T. Chen, K. K. Ma, and L. H. Chen, "Tri-state median filter for image denoising," IEEE Trans. Image Process., vol. 8, pp. 1834–1838, Dec.1999

[14]. Y. Dong and S. Xu, “A new directional weighted median filter for removal of random-value impulse noise,” IEEE Signal Process. Lett, vol.14, pp. 193–196, Mar. 2007.

[15]. C. C. Kang and W. J. Wang, “Modified switching median filter with one more noise detector for impulse noise removal,” Int.J.Electron.Commun.No. DOI: 10.1016/j.aeue.2008.08.009, 2008.

[16]. H.Hwang and R.A.Haddad, “Adaptive Median Filters: New Algorithms and Results,” IEEE Trans. Image Processing, vol.4, no.4, pp.499-502, 1995.

[17]. Z. Wang and D.Zhang, “Progressive Switching Median filter for the removal of Impulse Noise from Highly corrupted images,” IEEE trans. on circuits and systems part II: Analog and Digital signal processing, vol.46, no.1, pp.78-80, 1999.

[18]. T. Chen and H. R.Wu. Space variant median filters for the restoration of impulse noise corrupted images. *IEEE Trans. Circuits Syst. II*, 48(8):784–789, August 2001.

[19]. Smaïl Akkoul, Roger Lédée, Remy Leconge, and Rachid Harba, “A New Adaptive Switching Median Filter”, IEEE Signal Processing letters, pp. 587-590. 2010.

[20]. K. K. V. Toh, H. Ibrahim, and M. N. Mahyuddin, “Salt-and-pepper noise detection and reduction using fuzzy switching median filter,” IEEE Trans. Consumer Electron., vol. 54, no. 4, pp. 1956–1961, Nov. 2008.

[21]. Kenny Kal Vin Toh,”Noise adaptive fuzzy switching median filter for salt-and-pepper noise reduction” IEEE signal processing letters, VOL. 17, NO. 3 pp 281-244, Mar. 2010.

[22]. Russo, F, “A FIRE filter for detail-preserving smoothing of images corrupted by mixed noise,” IEEE fuzzy syst., vol.2, pp 1051-1055, July 1997.

[23]. Russo F., Noise Cancellation Using Nonlinear Fuzzy Filters, in: IEEE Instrumentation and Measurement Technology Conf., 1997,pp.772-777

[24]. Russo F., FIRE operators for image processing, in: IEEE Fuzzy Sets and Systems, Vol. 103, 1999, pp. 265-275.

[25]. Russo F. & Ramponi G., A fuzzy filter for images corrupted by impulse noise, in: IEEE Signal proceedings letters, Vol.3, No. 6, 1996, pp. 168- 170.

[26]. Russo F. & Ramponi G., Removal of impulse noise using a FIRE filter, in: IEEE Proceedings, 1996, pp. 975-978.

[27]. Kwan H.K., Fuzzy Filters for noise reduction in images, in: Fuzzy Filters for Image Processing (Nachtegael M., Van der Weken D., Van De Ville D. & Etienne E.E., editors), Springer-Verlag, 2002, pp. 25-53

[28]. Russo F. & Ramponi G., A noise smoother using cascade FIRE Filters, in: Proceedings of the 4th fuzzy-IEEE Conference, 1995, pp. 351-358.

[29]. Jiu J.Y., Multilevel median filter based on fuzzy decision, DSP IC Design Lab E.E. NTU., 1996.

[30]. Wang J.-H. & Chiu H.-C., HAF: An adaptive fuzzy filter for restoring highly corrupted images by histogram estimation, in: Proc. Natl. Sci.Counc. ROC (A), Vol. 23, No. 5, 1999. pp. 630-643.

[31]. Khodambashi, S.; Moghaddam, M.E.; "An impulse noise fading technique based on local histogram processing "; Signal Processing and Information Technology (ISSPIT), 2009 IEEE International Symposium 2009 , pp 95 – 100.

[32]. Jung-Hua Wang; Wen-Jeng Liu; Lian-Da Lin; "Histogram-based fuzzy filter for image restoration ", Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions Volume: 32 Issue: 2, 2002,pp: 230 – 238.

[33]. Yi Wan; Qiqiang Chen; Yan Yang , "Robust Impulse Noise Variance Estimation Based on Image Histogram"; Signal Processing Letters, IEEE ; 2010 , pp: 485 – 488.

[34]. Russo F., Nonlinear Filtering of Noisy Images Using Neuro-Fuzzy Operators, 1997, pp: 412-415.

[35]. Yuksel, M.E.; Basturk, A.; Besdok, E.; Yildirim, M.T.; "Detail-preserving restoration of impulse noise corrupted images by a switching median filter controlled by a neuro-fuzzy network"; Signal Processing and Communications Applications Conference, 2004 ,pp :87 – 90.

[36]. Yuksel, M.E.; "A hybrid neuro-fuzzy filter for edge preserving restoration of images corrupted by impulse noise";IEEE Transactions on image Processing 2006 pp: 928-936.

[37]. Yuksel, M.E.; Besdok, E.; "A simple neuro-fuzzy impulse detector for efficient blur reduction of impulse noise removal operators for digital images"; IEEE Transactions on Fuzzy Systems, 2004 , pp: 854 – 865.

[38]. K. K. Singh, A. Mehrotra, Dr. K. Pal, Dr. M.J. Nigam, "A N(8) Detail Preserving Adaptive Filter for Impulse Noise Removal" in *Int. Conf. on Image Imformation Processing*, 2011.

[39]. Ali S. Awad," Standard Deviation for Obtaining the Optimal Direction in the Removal of Impulse Noise" IEEE Signal Process. Lett, vol. 18, no.7. pp. 407–410, July 2011.

[40]. P. Y. Chen and C. Y. Lien, "An efficient edge-preserving algorithm for removal of salt-and-pepper noise," IEEE Signal.

[41]. Umesh Ghanekar, Awadhesh Kumar Singh, and Rajoo Pandey," A Contrast Enhancement-Based Filter for Removal of Random Valued Impulse Noise" IEEE Signal Process. Lett, vol. 17, no.1. pp. 47–50, Jan. 2010.S