A
Dissertation
On

Noise Reduction Using Fuzzy Filter

Submitted in Partial fulfillment of the requirements
For the award of Degree of

MASTER OF ENGINEERING
(Computer Technology and Application)
Delhi University, Delhi

Submitted By:
SOMENDRA PRAKASH
(University Roll No. 10195)

Under the Guidance of:
Dr. O.P VERMA.
Head of Department
Department Of Information Technology
Delhi College of Engineering, Delhi

DEPARTMENT OF COMPUTER ENGINEERING
DELHI COLLEGE OF ENGINEERING
UNIVERSITY OF DELHI
2008

CERTIFICATE

DELHI COLLEGE OF ENGINEERING
(Govt. of National Capital Territory of Delhi)
BAWANA ROAD, DELHI - 110042

Date:___________

This is certified that the work contained in this dissertation entitled **"Noise Reduction Using Fuzzy Filter"** by Somendra Prakash, College Roll no. 16/CTA/06, University Roll no. 10195, is the requirement of the partial fulfillment for the award of degree of Master of Engineering in Computer Technology & Application at Delhi College of Engineering. This work was completed under my direct supervision and guidance. He has completed his work with utmost sincerity and diligence.

The work embodied in this major project has not been submitted for the award of any other degree to the best of my knowledge.

Dr. O. P Verma
Head Of Department
Department of Information Technology
Delhi College of Engineering

ACKNOWLEDGEMENT

"If brain is the nucleus of thoughts, teacher is the source of energy to run the operation of solving cross puzzles of doubts that often poise the mind of students."

I am thankful to the Almighty because without his blessings this work was not possible. It is a great pleasure to have the opportunity to extent my heartfelt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor Dr. O.P Verma for his invaluable guidance, encouragement and patient reviews.
His continuous inspiration has made me complete this dissertation. He kept on boosting me time and again for putting an extra ounce of effort to realize his work.

I would also like to take this opportunity to present my sincere regards to my teachers Dr. Daya Gupta, Dr. M. Kulkarni, Mrs. Rajni Jindal, Mr. Manoj Sethi and Mr. Rajeev Kumar for their support and encouragement.

I am also thankful to my classmates for their unconditional support and motivation during this work.

Somendra Prakash
M.E.(Computer Technology and Application)
Department of Computer Engineering

ABSTRACT

Vision in general and images in particular have always played an important and essential role in human life. The field of image processing has numerous commercial, scientific, industrial and military applications.

One of the important issues is in image processing concern noise detection and reduction. Noise can occur in images because of several reasons, e.g. the circumstances of recording, the circumstances of transmission (damaged data), storage, copying, etc. Most common type of noise includes impulse noise (e.g. salt and pepper noise), additive noise (e.g. Gaussian noise) and multiplicative noise (e.g. speckle noise). The presence of noise can be very disturbing. Therefore the noise reduction is a well known problem in image processing. The reduction of noise in an image is a pre-processing step. Besides classical filter, fuzzy filter can be used to reduce the noise in an image; fuzzy filters use techniques from fuzzy set theory, and have the ability to incorporate the uncertainty that is involved in noise reduction. The objective of the thesis is development of fuzzy filters, which are adaptive in nature and remove different noises like salt and pepper and Gaussian noise.

Most of the classical filters that remove noise simultaneously also blur the edges, but fuzzy filters have the ability to combine edge-preservation and smoothing. Compared to other non-linear techniques, fuzzy filter are able to represent knowledge in a comprehensible way.

Contents

List of Tables :

1. Introduction

1.1General Concept

It is not a surprise that image processing is a growing research field. Vision in general and images in particular have always played an important and essential role in human life. The field of image processing has various commercial, scientific, industrial and military applications.

One of the important issues is in image processing concern noise detection and reduction. Noise can occur in images because of several reasons, e.g. the circumstances of recording, the circumstances of transmission (damaged data), storage, copying, etc. Most common type of noise includes impulse noise (e.g. salt and pepper noise), additive noise (e.g. Gaussian noise) and multiplicative noise (e.g. speckle noise). The presence of noise affects the accuracy of image processing. Therefore the noise reduction is a well known problem in image processing. The reduction of noise in an image is a pre-processing step. In order to reduce the noise, nonlinear filtering techniques are often employed because they provide better results than linear methods for noise removal and do not degrade the edges and the details of the image. Besides classical filter, fuzzy filter can be used to reduce the noise in an image; fuzzy filters use techniques from fuzzy set theory, and have the ability to incorporate the uncertainty that is involved in noise reduction. The objective of the thesis is development of fuzzy filters, which are adaptive in nature and remove different noises like impulse noise (e.g. salt and pepper noise), and additive noise (e.g. Gaussian noise).

Most of the classical filters that remove noise simultaneously also blur the edges, but fuzzy filters have the ability to combine edge-preservation and smoothing. Compared to other non-linear techniques, fuzzy filter are able to represent knowledge in a comprehensible way. The class of fuzzy filters makes use of techniques from fuzzy set theory to incorporate the intrinsic imprecision and uncertainty that comes along with noise detection.

1.2 Motivation

Image processing is concerned primarily with extracting useful information from images. It is also an important field for the automated visual inspection. In order to improve visual quality of image, the acquired images must pass through a stage of image processing to remove distracting and useless information (e.g. unwanted noise) from images. Therefore, preprocessing techniques can play an important role in increasing the accuracy of subsequent tasks.

Enhancement of noisy image data is a very challenging issue in many research and application areas. In the last few years, nonlinear filters based on fuzzy models have been shown to be very effective in removing noise without destroying the useful information contained in the image data. Our motive is to develop a fuzzy filter that is adaptive in nature and remove the noise from the image information.

1.3 Scope of the study

It is well known that fuzzy filters have better performance than classical filters. For example , most classical filters that remove noise simultaneously blur the edges, while fuzzy filters have the ability to remove noise with preservation of image information. Fuzzy filters are able to represent knowledge in a comprehensible way, as compared to other non-linear filtering techniques.

Median filters can eliminate impulse noise, have good edge preserving ability, and have moderate noise attenuation ability in the flat regions of image. However, the performance of a median filter is average for filtering random noise in an image. Random noise is one of the most common noises in images. Another nonlinear filtering techniques called moving average filters, which can smooth random noise, but can not suppress impulse noise, and can not preserve sharp edges of an image. In this, we define five fuzzy filters which apply fuzzy membership-type of weighted functions to the image pixel-values within a moving window. These fuzzy filters attempts to incorporate the feature of a median filter for filtering impulse noise and preserving edge, and/or the feature of a moving average filter for filtering random noise.

Another fuzzy filters that removes the Random Noise based on a parameter 'α' which is fixed for each pixel for the entire image is also presented.. In fact we require two values of 'α' ; one for smoothing and other for sharpening of the image. α_{sharp} has a value greater than α_{smooth}. We have taken largest possible value for α_{sharp} , which is L-1 or 255.

1.4 Objective of the thesis

The objectives of the thesis are as follows :

1. To develop fuzzy filters for filtering different types of noises like impulse noise (e.g. salt and pepper noise), additive noise (e.g. Gaussian noise).
2. To make the fuzzy filters adaptive which can remove the noises from image based on the information obtained from the image itself.

1.5 Organization

The remainder of thesis is organized as follows:

Chapter 2 provides a literature review of different noise models and classical filters.

Chapter 3 describes about fuzzy logic that includes introduction to fuzzy logic, fuzzy fundamentals, fuzzy image processing, and its advantages over classical filter, classification of fuzzy filter and their comparison.

Chapter 4 presents the implementation of fuzzy filters for noise reduction in images which includes Gaussian fuzzy filter with median centre(GMED), the symmetrical triangular fuzzy filter with median centre (TMED), the Gaussian fuzzy filter with moving average centre (GMAV), the symmetrical triangular fuzzy filter with moving average centre (TMAV) and two standard median filter(MED) and the moving average filter (MAV).

Chapter 5 present the method of development and implementation of fuzzy filter for removal of Gaussian Noise.

Finally conclusions about this thesis for noise reduction along with the suggestions for future work are given in chapter 6.

2. Image, Noise and Filtering

In this chapter, we present the model of image degradation process i.e. noise models, followed by classical filters, the noise include uniform noise, salt and pepper noise and Gaussian noise etc. The classical filters include linear filters, rank filters, and adaptive filters.

2.1 Digital images

A digital image may be defined as a discrete two-dimensional function, f(x, y), where x and y are spatial coordinates. The amplitude of f at any pair of coordinates (x, y) is called the gray level of the image at that point. It will be assumed that the image is rectangular, consisting of Y rows and X columns. The resolution of such an image is written as X x Y. By convention , f(0, 0) is taken to be the top left corner of the image, and f(X-1, Y-1) the bottom right corner. This is shown in figure 2.1.

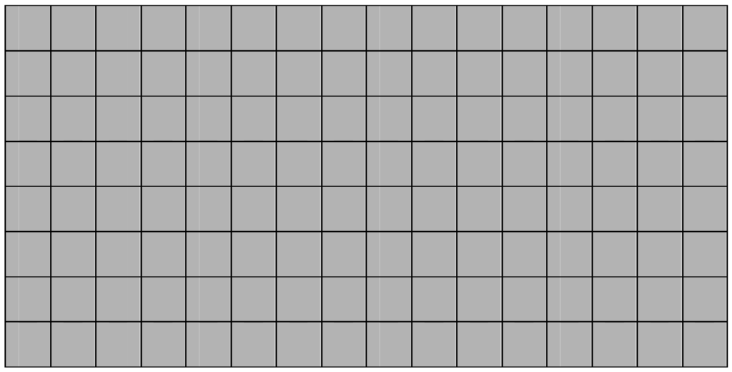


Figure 2.1 : A rectangle digital image of resolution 16 X 8.

Each distinct coordinate in an image is called a *pixel (*picture element). The nature of the output of *f*(x, y) for each pixel is dependent on the type of image. Most images are the result of measuring a specific physical phenomenon, such as light. A g*rayscale* image measures light intensity only. Each pixel is a scalar proportional to the brightness. The minimum brightness is called black, and the maximum brightness is called white. A typical example is given in Figure 2.2. A *color* image measures the intensity and chrominance of light.

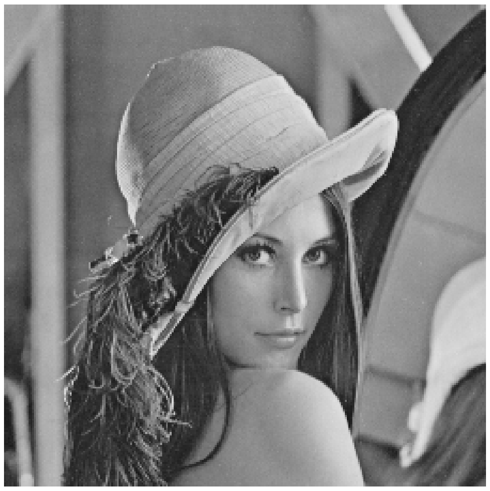


Figure 2.2 : A typical grayscale image Lena of resolution 256 X 256.

For storage purposes, pixel values need to be quantized. The brightness in grayscale images is usually quantized to *Z* levels, so *f*(x, y) E {0,1, - - -, Z – 1}. If Z has the form 2^{L}**,** the image is referred to as having *L bits per pixel*. Many common grayscale images use 8 bits per pixel, giving 256 distinct gray levels.

2.2 Image statistics

2.2.1 The histogram

A *histogram* plots the relative frequency of each pixel value that occurs in a grayscale image. The histogram provides a convenient summary of the intensities in an image, but is unable to convey any information regarding spatial relationships between pixels. Figure 2.3 shows the intensity histogram for the image from Figure 2.2. In this example, the image does not contain many very low or very high intensity pixels. It is possible that peaks in the histogram correspond to objects in the image.

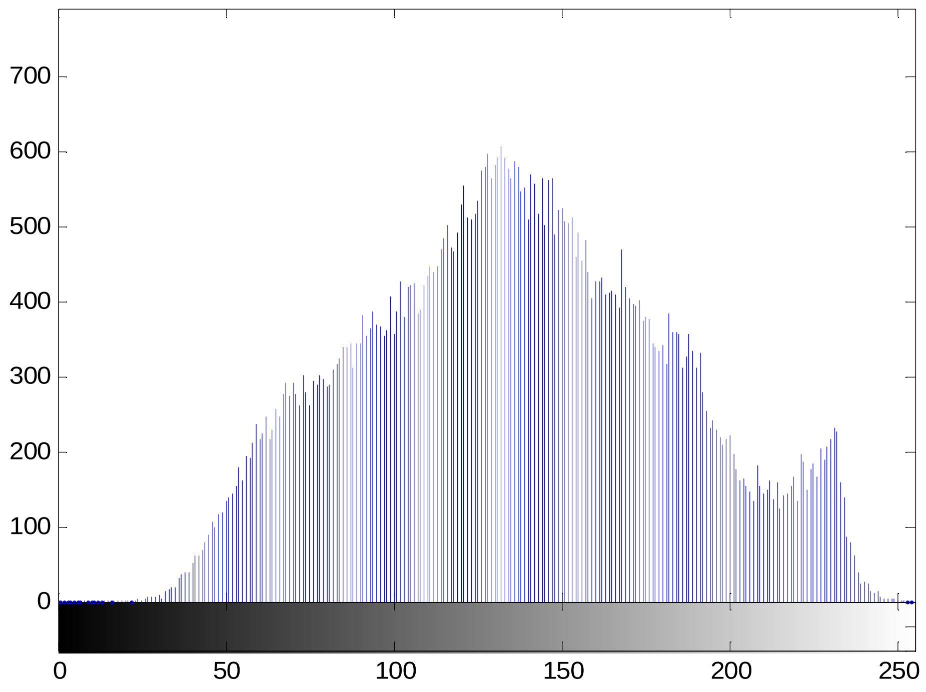


Figure 2.3 : The histogram for the grayscale image in figure 2.2

2.2.2 The mean

The average pixel value of an image is referred as image mean. For a grayscale image this is equal to the average brightness or intensity. Let the image $f(x, y)$ be referred as $\boldsymbol{f}$ of size Y x X**.** The mean of this image, $\boldsymbol{f}_{\text{mean}}$, may be calculated using Equation 2.1.

$$\boldsymbol{f}_{\text{mean}} = \frac{}{} \qquad (2.1)$$

2.2.3 The variance

The image variance, f_{variance}, gives an estimate of the spread of pixel values around the image mean. It can be calculated using equation 2.2. The standard deviation is simply $\sqrt{f_{\text{variance}}}$.

$$f_{\text{variance}} = — \tag{2.2}$$

2.3 Image operations

2.3.1 Image-scalar operations

Various useful arithmetical operations may be defined for images. Let $\otimes$ represent the binary operator for addition, subtraction, multiplication, or division. Equation 2.3 shows how to combine a scalar, **c**, and an image, **g**, to produce a new image f. This is a pixel-wise operation - each pixel in g is operated on using $\otimes$ with c, and the result put in f.

$$\mathbf{f} = \mathbf{g} \otimes c \qquad \equiv \qquad \forall(x, y) \quad f(x, y) = g(x, y) \otimes c \tag{2.3}$$

This equation can be used to enhance an image which is too dark. Consider the image in figure 2.4a which uses 8 bits per pixel (256 levels), but only contains pixels with intensities from 64 to 191. One may consider enhancing it to use the full intensity range using Equation 2.4., and floating point precision is used for all pixels during the calculation. The result is given in figure 2.4b.

$$\mathbf{f} = \left\lfloor \frac{\mathbf{g} - 64}{128} \times 255 \right\rfloor \tag{2.4}$$

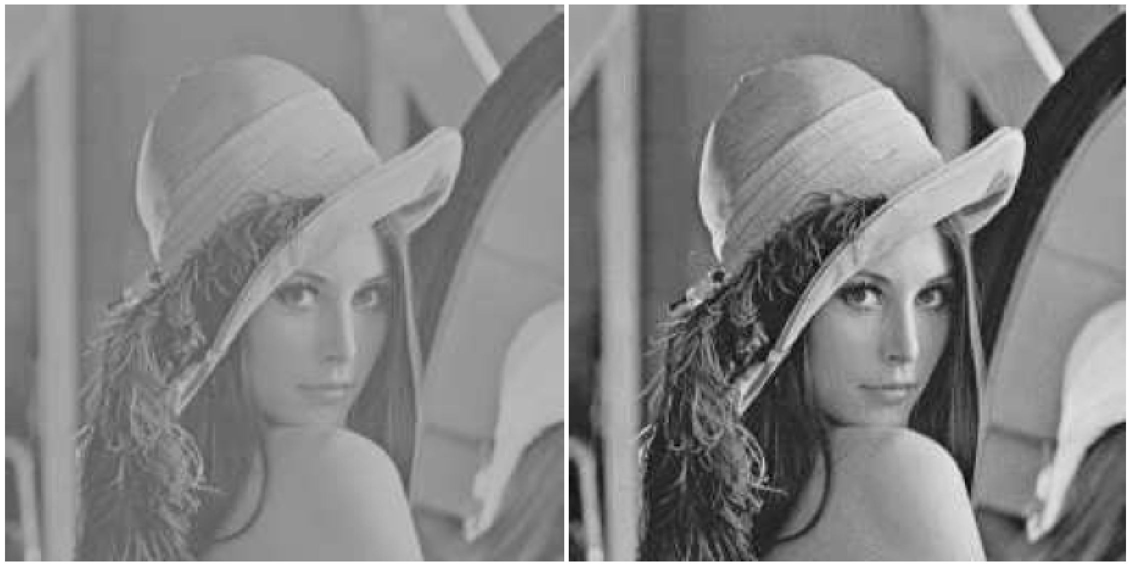


Figure 2.4 : (a) Low contrast image ; (b) after enhancement

2.3.2 Image-image operations

The image-scalar operation may be extended to the combination of two images, $\mathbf{g}_1$ and $\mathbf{g}_2$, having the same resolution. Instead of combining a scalar with each pixel, two pixels with the same coordinate in different images are used instead. Equation 2.5 describes this process.

$$\mathbf{f} = \mathbf{g_1} \otimes \mathbf{g_2} \qquad \equiv \qquad \forall (x,y) \;\; f(x,y) = g_1(x,y) \otimes g_2(x,y) \tag{2.5}$$

2.4 Image acquisition

Image acquisition is the process of obtaining a digitized image from a real world source. The random changes into the values of pixels in the digitized image may occur during the acquisition process. These random changes are called *noise*. Figure 2.5 shows the sources of noise introduced during the image acquisition process.

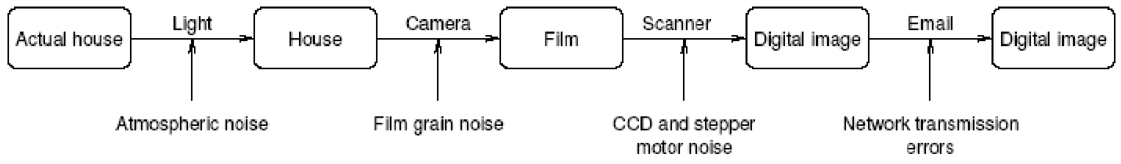


Figure 2.5. Introduction of noise during image acquisition process.

2.5 Images and Noise Models

Grayscale images are mathematically modeled as finite grids of numbers, where the numbers represents gray values. The number of gray values usually is 256 (i.e. the values range between 0 and 255), the number of rows and columns determines the size of the image (e.g. size of 256 X 256, 512 X 512, ..). A point in the grid is called a pixel (picture element).

In practice, images easily get corrupted with noise, e.g. due to the circumstances of recording, the circumstances of transmission (damaged data), storage, copying, etc. The properties of the noise introduced due to various circumstances are likely to vary. However, there are standard noise models which model well the types of noise encountered in most images includes additive noise, impulse noise, and multiplicative noise etc. Figure 2.6 shows how these types of noise affect a typical grayscale image.

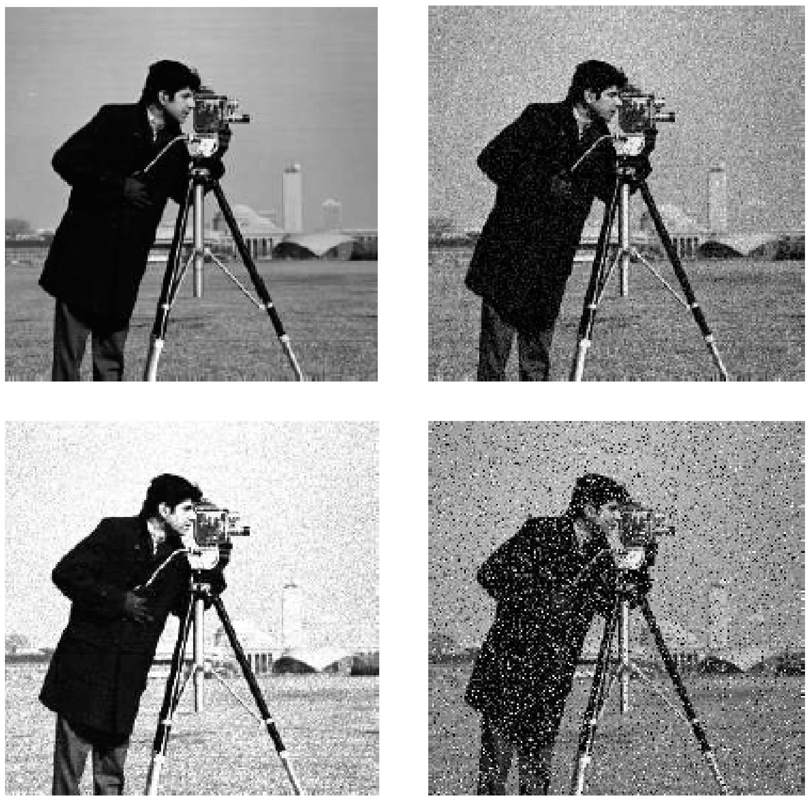


Figure 2.6: Different types of noise: (a) original image; (b) additive noise; (c) multiplicative noise; (d) impulse noise.

2.5.1 Additive noise

Let g(x, y) be the noisy image of the ideal image, f(x, y), and η(x, y) be a noise function which returns random values coming from an arbitrary distribution. Then additive noise can be described by Equation 2.6.

$$g(x, y) = f(x, y) + \eta(x, y) \qquad (2.6)$$

Additive noise is independent of the pixel values in the original image. Typically, additive noise η(x, y) is symmetric about zero. This has the effect of not altering the average brightness of the image. Thermal noise within photo-electronic sensors is a good model of additive noise.

The level of noise is generally expressed by its variance. For example, if a color image is digitized with RGB values in the range (0,.. 255), additive Gaussian noise with variance, $\sigma_n{}^2$ =1 would not be visible. A moderate noise level with $\sigma_n{}^2$ = 100 makes an image grainy, while, $\sigma_n{}^2$ =1000; the noise level obscures the image. Similar effects are observed with other noise distributions.

In order to compare the performance of the original input image, degraded image and processed images, some measures of error are necessary. The signal-to-noise ratio (SNR) is often used for the characterization of signal , which can be calculated using Equation (2.7).

$$SNR \ \ in \ \ dB = 10 \times \log_{10} \frac{\sigma_f^2}{\sigma_n^2} \tag{2.7}$$

Where $\sigma_f{}^2$ and $\sigma_n{}^2$ are the variances in the signal and noise respectively.

The mean square error (MSE) between the original image f(x,y) and the filtered image g(x,y) can be calculated using Equation (2.8)

$$\text{MSE}[f(x,y),g(x,y)] = \frac{\sum_{i=1}^{M1} \sum_{j=1}^{M2} [f(x,y)-g(x,y)]^2}{M1.M2} \tag{2.8}$$

Where M1 and M2 are image dimensions.

In practice, images easily get corrupted with noise, e.g. due to the circumstances of recording (e.g. dust on a lens, electronic noise in cameras and sensors, etc.), transmission (e.g. electromagnetic interaction with satellite images, transmission over a channel, etc.), storage, copying, scanning, etc. Video images transmitted via satellite are very susceptible to the electronic interference due to sunspot activity.

The classification of noise is based upon the shape of its probability density function (PDF). The mean and variance are important parameters to characterize the noise. Mean value $\overline{m}$, gives the average brightness of the noise and square root of variance σ gives the average peak-to-peak gray level deviation of the noise. The mean and variance are defined as

$$\overline{m} = \sum_{k=0}^{g_{\max}} k \quad p_n(k) \tag{2.9}$$

$$\sigma^2 = \sum_{k=0}^{g_{\max}} (kp_n(k) - \overline{m})^2 \tag{2.10}$$

Where, $p_n(k)$ is the frequency of occurrence of noise amplitude, k. Ideally, k varies from $-\infty$ to $+\infty$, however, since the pixel levels are limited in the range [0, L-1], the noise amplitude level k also lies in [0, L-1].

2.5.2 Impulse noise

The probability density function (PDF) of impulse noise is given by Equation (2.11):

$$p(z) = \begin{cases} p_a & \text{for } z=a \\ p_b & \text{for } z=b \\ 0 & \text{otherwise} \end{cases} \qquad (2.11)$$

Where $P_{a\,=}$ probability of salt noise, & $P_{b\,=}$ probability of salt noise.

In the image information, if *b>a*, gray level b will appear as light dot in the image. Conversely, level *a* will appear dark dot in the image. If either P_a or P_b is zero, the impulse noise is called unipolar. If neither probability is zero, especially if *a* and *b* are approximately equal, impulse noise will resemble salt and pepper granules randomly distributed over the image .For this reason, bipolar impulse noise is also called salt-and-pepper noise. Shot and spike noises also refer to this type of noise. Noise impulse can be positive or negative.

2.5.3 Multiplicative noise

Multiplicative noise, or speckle noise, is a signal dependent form of noise whose magnitude is related to the value of the original pixel. Multiplicative noise in the simple form can be described by Equation (2.12). Multiplicative noise is an approximation to the noise encountered in images recorded on film slides.

$$g(x, y) = f(x, y) + \eta(x, y)\, f(x, y) = f(x, y)[1 + \eta(x, y)] \qquad (2.12)$$

2.5.4 Uniform noise

Uniform noise produces noise values with equal probability in the range from g_{max} to g_{min}. The histogram of the uniform noise is given by Equation(2.13):

$$p_n(k) = \begin{cases} \dfrac{1}{g_{\max} - g_{\min}} \ ; \ g_{\min} < k < g_{\max} \\ 0 \ ; \ otherwise \end{cases} \tag{2.13}$$

The mean and standard daviation are computed using as follows :

$$\overline{m} = \frac{g_{\max} + g_{\min}}{2} \tag{2.14}$$

$$\sigma = \frac{g_{\max} - g_{\min}}{\sqrt{12}} \tag{2.15}$$

2.5.5 Gaussian noise

The Gaussian noise is most common type of noise that is found in an image, which is the result of many unknown noises from independent sources added together. The probability density function (PDF) of Gaussian noise is given by Equation (2.16):

$$p_n(k) = \frac{1}{\sigma\sqrt{2\pi}} \quad e^{-\frac{(k-\overline{m})^2}{\sigma^2}} ; \qquad -\infty < k < \infty \tag{2.16}$$

In the probability density function (PDF), mean is located at the peak, having highest probability of occurrence and the width is determined by the standard deviation. Gaussian noise is defined over infinite range; however, the digitized image has finite range. Hence the noise values that exceed the gray level range are deposited at the 0 and g_{max} points on the PDF. For 99.7% of the gray levels, the peak-to-peak gray level deviation is equal to 6σ . For example, consider an image containing Gaussian noise with a mean gray level of 128 and a standard deviation of 10. For 99.7% of the pixels in this image, the peak-to-peak gray level deviation will be 60. This results in the image’s gray levels varying between 98 and 158.

2.5.6 Quantization noise

Quantization noise is due to the quantization of pixel values during the analog to digital conversion. For example, imagine an analog image with brightness values ranging from 0 to 10. If it is quantized to accuracy 0.1, the digitized image will have 101 distinct gray levels. A given intensity, L, could have originally been anywhere in the range L ± 0.05. This uncertainty in the true value of L is called quantization noise.

2.5.7 Periodic noise

Periodic noise in an image can be introduced during image acquisition process from electrical or electromechanical interference. This is the only type of spatially dependent noise. Periodic noise can be reduced significantly via frequency domain filtering.

2.6 Local windows

The union of the pixel being processed and its neighbouring pixels may be collectively referred to as a window, a mask, or the local region surrounding the pixel. Local windows typically involve fewer than 50 pixels, on images with up to 10^2 pixels. The only unbiased window configuration is an isotropic one - symmetrical and centered on the pixel to be processed. A circular window meets this requirement, but because pixels reside on a rectangular grid, some pixels would cover an area only partially within the circle. Figure 2.7 shows five local windows which are commonly used in image processing. Each is an approximation to a circular window, with the square windows being simplest to implement.

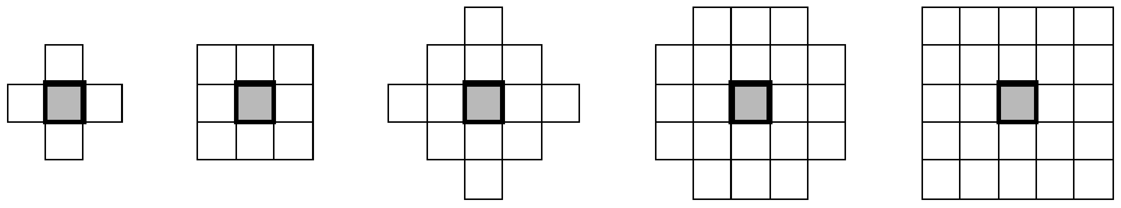


Figure 2.7: Common local neighbourhood configurations: 5, 9, 13, 21, 25 pixels.

2.7 Filtering

As explained above that, images easily get corrupted with noise due to the circumstances of recording, the circumstances of transmission (damaged data), storage, copying, etc. The types of noise encountered in most images includes additive noise, impulse noise, and multiplicative noise etc. This unwanted noise are removed by filters. Filters are mainly used to suppress either the high frequencies in the image, ***i.e., for*** smoothing the image, or the low frequencies, ***i.e.,*** for enhancing or detecting edges in the image.

Suppose that an image-processing operator F acts on the two input images A and B and produces output images C and D respectively. If the operator F is $linear$**,** then

$$F\ (a \times A + b \times B) = (a \times C) + (b \times D)$$

Where $\boldsymbol{a}$ and $\boldsymbol{b}$ are constants. This means that each pixel in the output of a $\boldsymbol{linear}$ operator is the weighted sum of a set of pixels in the input image.

If the image processing operator is nonlinear such as threshold operators, then for an input image, A(x, y), the threshold output image , B(x, y), is defined by Equation(2.17) :

$$B(x, y) = \begin{cases} 1 & if\ A(x, y) > T \\ 0 & if\ A(x, y) \leq T \end{cases} \qquad (2.17)$$

Suppose the threshold operator is *non-linear*, because individually, corresponding pixels in the two images A and B may be below the threshold, whereas the pixel obtained by adding A and B may be above threshold. Similarly, the absolute value operation is non-linear:

$$|-1+1| \neq |-1|+|1|$$

2.7.1 Temporal filtering

Temporal filtering averages multiple sampled versions of the same scene. If the true image is $\mathbf{f}$, and N samples, $\mathbf{g}_1 \ldots\ldots \mathbf{G}_N$, are taken, the temporally filtered image, $\mathbf{f'}$, can be calculated using Equation (2.18)

$$\mathbf{f'} = - \tag{2.18}$$

Temporal filtering is ideal if multiple version of the same image can be obtained. In practice, this does not usually happen, because the objects in the scene move, or the capturing equipment move. Even slight variations can displace the pixels in each sampled image, causing the whole average to become a blurred version of the original.

2.7.2 Spatial filtering

Temporal filtering uses multiple versions of each pixel value at the same position but different times, whereas spatial filtering uses pixels at the same time but at different spatial coordinates in the image. When temporally filtering a static scene, pixels from the same position but different times are expected to have the same noiseless value. When those pixels are not available, the best alternative is pixels near the current pixel. When only a single noisy image is available, spatial filtering is the only option. This is the criteria of nearly all noise reduction algorithms in image.

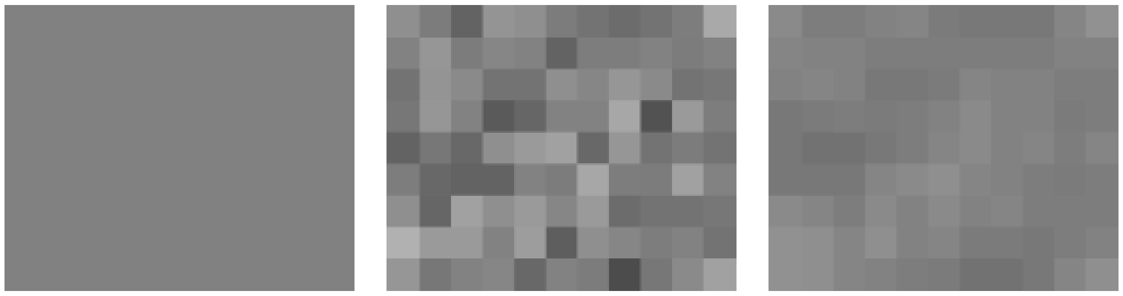


Figure 2.8: Application of the 3X3 box filter: (a) original; (b) noisy; (c) filtered.

The simplest spatial filter is the averaging filter which replaces a pixel with the average of itself and the pixels in the local neighbourhood. The most common averaging filter is the box filter, which gives equal weight to the 9 pixels in a 3X3 window. This choice of weights maximally reduces the noise variance when the noise is additive. Box filtering is similar to temporal filtering except that samples from adjacent pixel positions are used as approximations to multiple samples of the pixel at the same position. Example of 3X3 box filter is shown in Figure (2.8).

2.7.3 Linear filters

The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbors, including itself. This has the effect of eliminating pixel values, which are unrepresentative of their surroundings. Often a 3×3 square kernel shown in Figure 2.9 is used, although larger kernels (***e.g.***, 5×5 squares) can be used for more severe smoothing. (a small kernel can be applied more than once in order to produce a similar ,but not identical effect as a single pass produces with a large kernel)

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Figure 2.9: 3×3 averaging kernel used in mean filtering

Computing the straightforward convolution of an image with this kernel performs the mean filtering that is most commonly used as a simple method for reducing noise in an image.

There are two main problems with mean filtering, which are:

- A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighborhood.

- When the filter neighborhood straddles an edge, the filter will interpolate new values for pixels on the edge thereby blurring the edge. This may be a problem if sharp edges are required in the output.

2.7.4 Rank filters

For a linear filter, the output is a linear combination (i.e., weighted average) of neighboring pixels. Problem with linear filter is that edges are blurred by its application. To improve upon them non-linear filters are used.
The rank filters are a class of non-linear filters. With a rank filter operating in a window about a pixel ij , the N pixel within the neighborhood is ranked as :

$$g^1_{ij} \le \; g^2_{ij} \le \dots \; \le g^N_{ij}$$

Then ,the output value is chosen.

$$f_{ij} = R_k (g_{ij})$$

Where g is the input image, f is the output image and k_{th} rank value in the window is chosen. Three special rank filters are the $\min_{n}$, $\max_{n}$ and median filters.

$$\min_n (g) = R_1(g)$$

$$\max (g) = R_N(g)$$

$$\text{median}(g) = R_{\lceil N/2 \rceil}(g)$$

The net effect of three rank filters is to reduce the variance in the image. Well-known properties of median filters are :

- They preserve sharp edges
- They eliminate spike (salt and pepper) noise

2.7.4.1 Median filters

This involves sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value so as to calculate the median. If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used. Figure 2.10 illustrates calculation of median value.

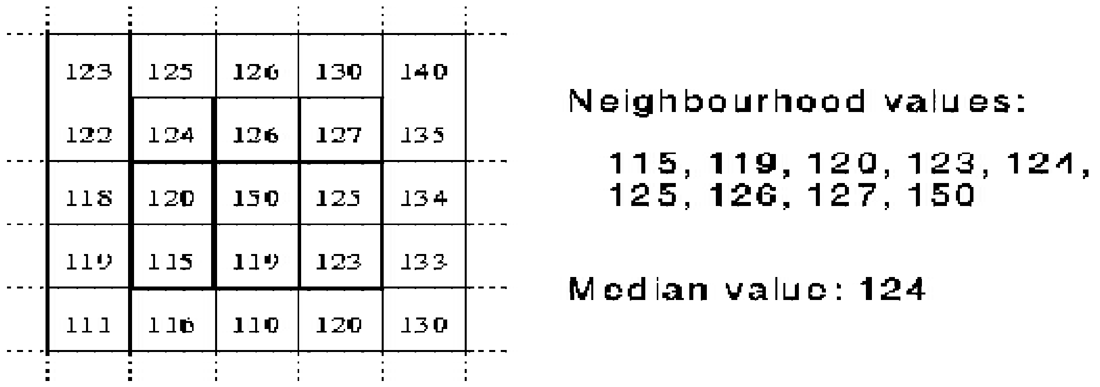


Figure 2.10 : The median value of a pixel neighborhood.

As can be seen, the central pixel value of 150 is rather unrepresentative of the surrounding pixels and is replaced with the median value: 124. A 3×3 square neighborhood is used here , however ,larger neighborhoods will produce more severe smoothing.

By calculating the median value of a neighborhood rather than the mean filter, the median filter has two main advantages over the mean filter:

- The median is more robust than the mean and so a single very unrepresentative pixel in a neighborhood will not affect the median value significantly.
- Since the median value must actually be the value of one of the pixels in the neighborhood, the median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason the median filter is much better at preserving sharp edges than the mean filter.

In General, the median filter allows a great deal of high spatial frequency detail to pass while remaining very effective at removing noise on images where less than half of the pixels in a smoothing neighborhood have been effected. (As a consequence of this, median filtering can be less effective at removing noise from images corrupted with Gaussian noise)

Unlike the mean filter, the median filter is non-linear. This means that for two images A (x) and B (x) ,we have

Median |A (x)+B (x) | ≠ Median |A (x)| + Median |B (x)|

2.7.5 Adaptive filters

Once selected, the filters are applied on image without regard for how the image characteristics vary from one point to another. The adaptive filter whose behavior changes is based on statistical characteristic of image inside the filter region defined by the m × n rectangular window. Adaptive filters are capable of performance superior to that of filters discussed previously. The price paid for improved filtering power is an increase in filter complexity.

2.7.5.1 Adaptive, local noise reduction filter

The simplest statistical measures of a random variable are its mean and variance. These are reasonable parameters on which to base an adaptive filter because they are quantities closely related to appearance of an image. The mean gives a measure of average gray level in the region over which the mean is computed, and the variance gives measures of average contrast in that region.

This filter is to operate on a local region, S_{xy}. The response of the filter at any point (x, y) on which the region is centered is to be based on following quantities :

(a) g(x, y), the value of noisy image at (x, y)

(b) σ^2_η, the variance of the noise corrupting f(x, y) to form g(x, y)

(c) m_L ,the local mean of the pixels in S_{xy}

(d) σ^2_L, the local variance of the pixels in S_{xy} .

An adaptive expression for obtaining restored image, **f’**(x, y), may be expressed in equation (2.19) as follows:

$$\mathbf{f'}(x, y) = g(x. y) - \frac{\sigma 2\eta}{\sigma 2L} [\, g(x, y) - m_L \,] \qquad (2.19)$$

2.7.5.2 Adaptive median filter

The adaptive median filter changes (increases) the size of S_{xy} during filter operation, depending on certain conditions. The output of the filter is a single value used to replace the value of the pixel at (x, y), the particular point on which the window S_{xy} is centred at a given time .

Consider the following notation:

Zmin = minimum gray level value in S_{xy}

Zmax = Maximum gray level value in S_{xy}

Zmed = Median of gray levels in S_{xy}

Zxy = gray level at coordinates (x, y)

Smax = maximum allowed size of S_{xy}

The adaptive median filtering algorithm works in two levels, denoted level A and level B, as follows:

```
Level A :  A1 = Zmed – Zmin
           A2 = Zmed – Zmax
           If A1 > 0 AND A2 < 0,    go to level B
           Else increase the window size
           If window size ≤ Smax    repeat level A
           Else output Zxy.
Level B :  B1 = Zxy – Zmin
           B2 = Zxy – Zmax
           If B1 >0 AND B2 < 0, output Zxy
           Else                 output Zmed.
```

The additional benefit of the adaptive median filter is that it seeks to preserve detail while smoothing non-impulse noise.

3. Fuzzy Logic in Image Processing

3.1 Introduction

The concept of Fuzzy Logic (FL) was invented by Prof. Lotfi A. Zadeh at the University of California in Berkeley in 1965 and presented as a way of processing data by allowing partial set membership rather than only full or non-membership. Basically, Fuzzy Logic (FL) is a multivalued logic, that allows intermediate values to be defined between conventional evaluations like true/false, yes/no, high/low, etc. He said " As complexity rises, precise statements lose meaning and meaningful statements lose precision."
Fuzzy logic is all about the relative importance of precision.

3.2. Fuzzy Set Theory

Fuzzy set theory is the extension of conventional (crisp) set theory .It handles the concept of partial truth (truth values between 1 (completely true) and 0 (completely false)). It is used to model the vagueness and ambiguity in complex systems.

The idea of fuzzy sets is simple and natural. For instance, we want to define a set of gray levels that share the property dark. In classical set theory, we have to determine a threshold, say the gray level 100. All gray levels between 0 and 100 are elements of this set; the others do not belong to the set. But the darkness is a matter of degree. So, a fuzzy set can model this property much better. To define this set, we also need two thresholds, say gray levels 50 and 150. All gray levels that are less than 50 are the full members of the set, all gray levels that are greater than 150 are not the member of the set. The gray levels between 50 and 150, however, have a partial membership in the set as shown in figure 3.1.(b).

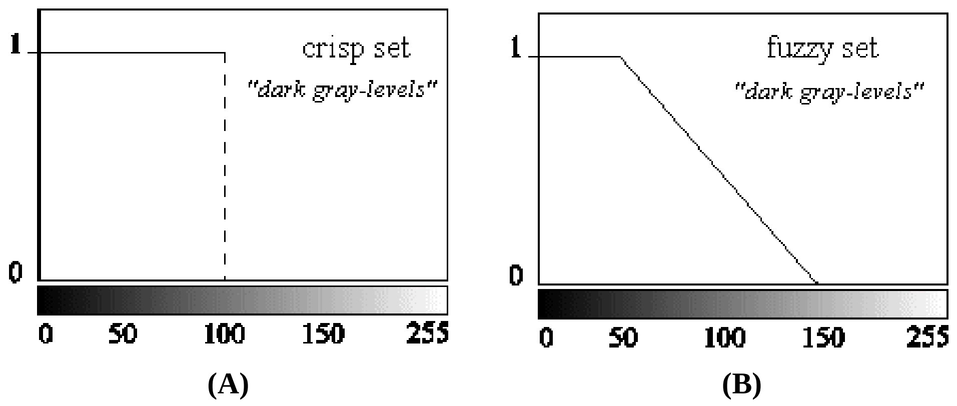


Figure 3.1.: Representation of "dark gray-levels (a) crisp set and (b) fuzzy set

•Fuzzy sets describe vague concepts

•A fuzzy set admits the possibility of partial membership in it.

•The degree an object belongs to a fuzzy set is denoted by a membership value between 0 and 1.

•A membership function associated with a given fuzzy set maps an input value to its appropriate membership value.

3.3 Fuzzy Logic

Fuzzy logic is a logic. The ultimate goal of fuzzy logic theory is to provide a foundation for approximate reasoning using imprecise propositions based on fuzzy set theory, in a way similar to the classical reasoning using precise propositions based on the classical set theory.

3.3.1 Membership Function in the Fuzzy Logic

A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is referred to as the universe of discourse. The only condition a membership function must really satisfy is that it must vary between 0 and 1.

A classical set might be expressed as
$A = \{x \mid x > 6\}$
A fuzzy set is an extension of a classical set. If X is the universe of discourse and its elements are denoted by *x*, then a fuzzy set A in X is defined as a set of ordered pairs:

$A = \{x, \mu_A(x) \mid x \ \varepsilon\ X\}$

$\mu_A(x)$ is called the membership function (or MF) of *x* in A. The membership function maps each element of X to a membership value between 0 and 1.

3.3.2 Why Use Fuzzy Logic?

Fuzzy logic is used because of the following observations:

- Fuzzy logic is conceptually easy to understand.
- Fuzzy logic is flexible.
- Fuzzy logic is tolerant of imprecise data.
- Fuzzy logic can model nonlinear functions of arbitrary complexity.
- Fuzzy logic can be built on top of the experience of experts.
- Fuzzy logic can be blended with conventional control techniques.
- Fuzzy logic is based on natural language.

3.4 Fuzzy Image Processing

Fuzzy image processing is the collection of all approaches that understand, represent and process the images, their segments and features as fuzzy sets. The representation and processing depend on the selected fuzzy technique and on the problem to be solved. The general structure of fuzzy image processing is shown in figure 3.2.
Fuzzy image processing has three main stages:

- Image fuzzification
- Modification of membership values
- if necessary, image defuzzification.

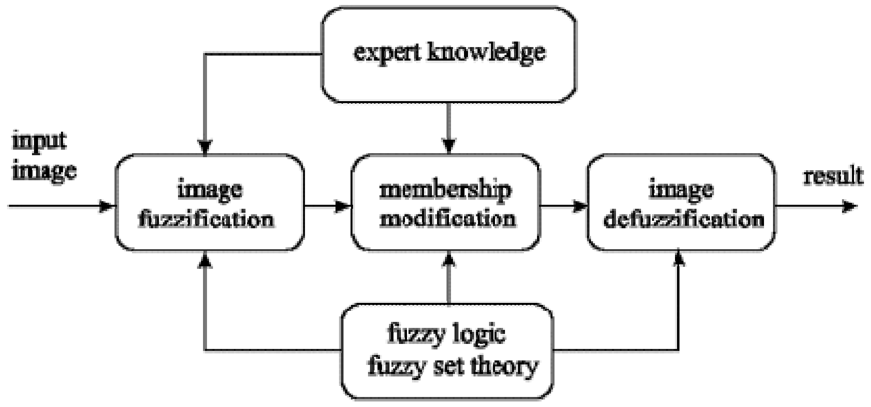


Figure 3.2 The general structure of fuzzy image processing

The main power of fuzzy image processing is in the middle step (modification of membership values). The fuzzification and defuzzification steps are due to the fact that we do not possess fuzzy hardware. Therefore, the coding of image data (fuzzification) and decoding of the results (defuzzification) are steps that make possible to process images with fuzzy techniques. After the image data are transformed from gray-level plane to the membership plane (fuzzification), appropriate fuzzy techniques modify the membership values. This can be a fuzzy clustering; a fuzzy rule-based approach, a fuzzy integration approach and so on. These steps are shown in Figure 3.3.

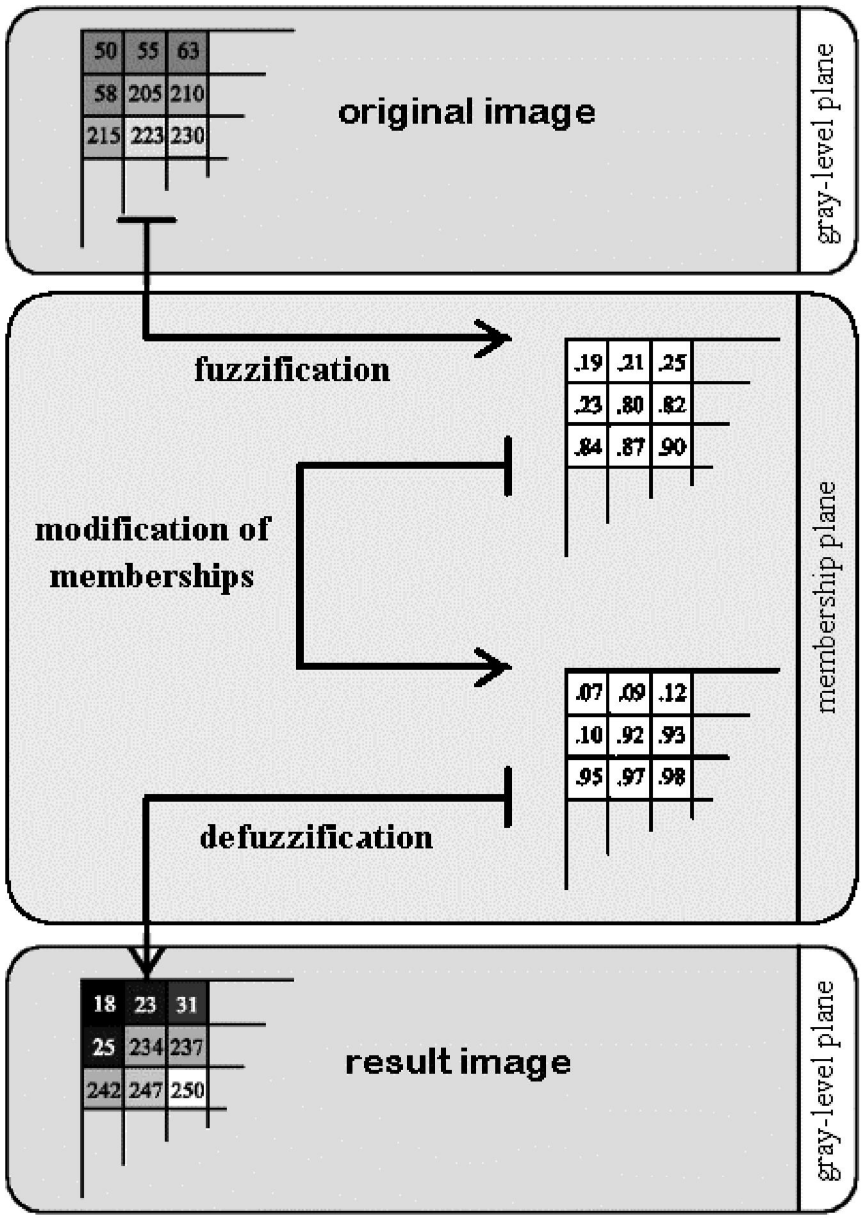


Figure 3.3: Steps in fuzzy image processing

3.5 Why Fuzzy Image Processing

Why we should use fuzzy techniques in image processing? There are many reasons to do this. The most important of them are as follows:

- Fuzzy techniques are powerful tools for knowledge representation and processing
- Fuzzy techniques deal with vagueness and ambiguity efficiently

In many image-processing applications, we have to use expert knowledge to overcome the difficulties. Fuzzy set theory and fuzzy logic offer us powerful tools to represent and process human knowledge in the form of fuzzy if-then rules. On the other side, many difficulties in image processing arise because the data/results are uncertain. This uncertainty, however, is not always due to the randomness but to the ambiguity and vagueness. Beside randomness, which can be managed by probability theory, we can distinguish between three other kinds of imperfection in the image processing as:

- Grayness ambiguity
- Geometrical fuzziness
- Vague (complex/ill-defined) knowledge

These problems are fuzzy in the nature. The question whether a pixel should become darker or brighter than it already is, the question where is the boundary between two image segments, and the question what is a tree in a scene analysis problem, all of these and other similar questions are examples for situations that a fuzzy approach can be the more suitable way to manage the imperfection.

The main difference to other methodologies in image processing is that input data X (histogram, gray level, etc,) will be processed in the so called membership plane where one can use great diversity of fuzzy logic ,fuzzy set theory and fuzzy measure theory to modify/aggregate the membership value ,classify data , or make decision using fuzzy inference. The new membership values are retransformed in the gray level plane to generate new histogram, modify gray levels, image segments, or class objects.

3.6 Fuzzy filters and their ability to incorporate uncertainty

3.6.1 The link between grayscale images and fuzzy sets

Grayscale images are mathematically modelled as finite grids of numbers, where the numbers represent grey values. The number of grey values usually is 256 (i.e., the values range between 0 and 255); the number of rows and columns determines the size of the image (e.g., size of 256 × 256, 512 × 512, …). In this way, a grayscale image can be regarded as a mapping from a certain universe X to a certain bounded and finite set of values.

It is interesting to observe that fuzzy sets are modelled in the same way. A fuzzy set in a universe X is modelled as a mapping from X to the unit interval [0,1], i.e., every element x of X is associated with a value in [0,1], called the membership degree of x in the considered fuzzy set. A high membership value of x indicates that x satisfies the property associated with the fuzzy set to a high degree, while a low membership value indicates the opposite. An example of a fuzzy set is figure 5.1(b). Consequently, fuzzy sets are suited to model imprecision (e.g., properties such as ‘high’, ‘low’, ‘far’, ‘many’, …). Furthermore, fuzzy logic can be applied to reason with this uncertainty.

Given the similarity, on an abstract level, between grayscale images and fuzzy sets, it follows that techniques from fuzzy set theory can be used in image processing. This has been done more often in the past years, and the results have shown that fuzzy techniques can offer an added value.

3.6.2 General outline of a fuzzy filter

The classical approach to reduce noise in a grayscale image mainly consists of replacing the gray value of a pixel with another value; the way in which the other value is determined depends on the filter that is applied. Quite often, all pixels are treated in the same way (the classical median filter is a typical example). It should be clear that this approach has some disadvantages:

- not all the pixels should be treated in the same way, because not all the pixels will be contaminated with noise in the same way
- one should try to find a more adaptive way to replace a pixel value (e.g., taking into account characteristics of a neighbourhood of the pixel).

Fuzzy set theory makes it possible to model and to reason with uncertainty. And uncertainty is what occurs when processing an image for noise reduction, because of the fact that one can distinguish degrees of contamination of a pixel in an image. Fuzzy set theory allows to model and to work with this uncertainty and to improve the quality of noise reduction. In general, a fuzzy filter used for noise reduction uses both numerical information (just as classical filters) and linguistic information (modelled by fuzzy set theory; e.g., 'small' and 'large' values). This information is processed by fuzzy rules (approximate reasoning; e.g., "if most of the gradient values are large, then assume that the pixel is noisy"), resulting in a (defuzzified) filter output. The general scheme of fuzzy filters is shown in Figure 3.4.

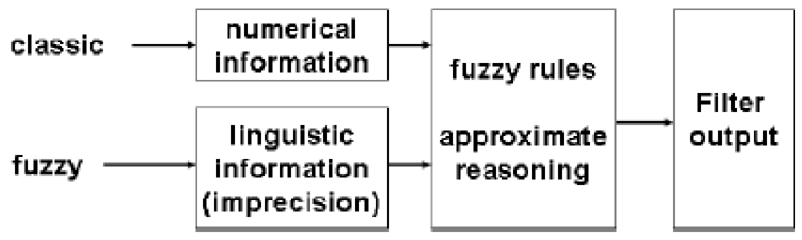


Figure 5.4 Fuzzy filters use both numerical and linguistic information to process an image

3.7 Filters for noise reduction

The variety of filters can be divided into three subclasses:

- classical filters
- fuzzy–classical filters, i.e., fuzzy-logic-based filters that are a modification or extension of classical filters
- fuzzy filters, i.e., filters that are purely based on fuzzy logic and have no straightforward connection with classical filters.

Based on above classification, the 38 different filters are as follows:

3.7.1 Classical filters

• MF: Median Filter
• WF: Weighted Filter
• AWF: Adaptive Weighted Filter
• WIENER: Wiener Filter
• GAUS: Gaussian Filter
• EMF: Extended Median Filter

3.7.2 Fuzzy–classical filters

• FMF: Fuzzy Median Filter
• TMED: Symmetrical Triangle Fuzzy Filter with Median Centre
• ATMED: Asymmetrical Triangle Fuzzy Filter with Median Centre
• GMED: Gaussian Filter with Median Centre
• FIDRM: Fuzzy Impulse Noise Detection and Reduction Method Filter
• WFM: Weighted Fuzzy Mean Filter
• FWM: Fuzzy Weighted Mean Filter
• AWFM1: First Adaptive Weighted Fuzzy Mean Filter
• AWFM2: Second Adaptive Weighted Fuzzy Mean Filter
• CK: Choi and Krishnapuram Filter
• FDDF: Fuzzy Decision-Directed Filter
• TMAV: Symmetrical Triangle Fuzzy Filter with Moving Average Centre
• ATMAV: Asymmetrical Triangle Fuzzy Filter with Moving Average Centre
• DWMAV: Decreasing Weight Fuzzy Filter with Moving Average Centre
• GMAV: Gaussian Fuzzy Filter with Moving Average Centre
• MPASS: Multipass Fuzzy Filter
• FMMF: Fuzzy Multilevel Median Filter

3.7.3 Fuzzy filters

• FIRE: Fuzzy Inference Ruled by Else-action Filter
• DSFIRE: Dual Step Fuzzy Inference Ruled by Else-action Filter

• PWLFIRE1: First (non-adaptive) Piecewise Linear Fuzzy Inference Ruled by Else-action Filter
• PWLFIRE2: Second (adaptive) Piecewise Linear Fuzzy Inference Ruled by Else-Action Filter
• IFCF: Iterative Fuzzy-Control-Based Filter
• MIFCF: Modified Iterative Fuzzy-Control-Based Filter
• EIFCF: Extended Iterative Fuzzy-Control-Based Filter
• SFCF: Smoothing Fuzzy-Control-Based Filter
• SSFCF: Sharpening Smoothing Fuzzy-Control-Based Filter
• GOA: Gaussian Noise Reduction Filter
• HAF: Histogram Adaptive Filter
• FSB1: First Fuzzy-Similarity-Based Noise Reduction Filter
• FSB2: Second Fuzzy-Similarity-Based Noise Reduction Filter
• FSB1R: First Recursive Fuzzy-Similarity-Based Noise Reduction Filter
• FSB2R: Second Recursive Fuzzy-Similarity-Based Noise Reduction Filter

3.8 Classification of fuzzy filters

For a meaningful classification of fuzzy filters, the following criteria can be considered:

(1) Noise type : for which noise type(s) is the filter designed?
(2) Degree of fuzziness : is the filter a pure fuzzy filter (i.e. is the output exclusively determined by fuzzy rules), or is it rather a modification of one or more classical filters (i.e. are fuzzy rules just an aid in the process)?
(3) Fuzzy rules: which criteria are used by the fuzzy rules to determine whether a pixel is a noise pixel or not (gray-value difference, …)?
(4) Algorithm: is the Algorithm iterative or not? Is it recursive or not?
(5) Complexity: how complex is the construction?

3.8.1 Noise type

The considered fuzzy filters are designed for impulse and / or gaussian noise. Impulse noise and gaussian noise are the most common types of noise. However, there are also other types of noise, such as speckle noise, uniform noise, etc. we have not encountered any fuzzy filter that is specifically designed to cope with the latter types of noise.
Note that, although they are both based on the classical adaptive weighted mean filter, the WFM filter is designed for impulse noise while the FWM filter is designed for gaussian noise.

3.8.2 Degree of fuzziness

Several fuzzy filter are modification, i.e. extensions, generalizations or refinements, of existing classical filters. These modifications are realized by using fuzzy rules. This group of filter can be divided in two classes:

(1) The output of the fuzzy filters depends on the output of one or more classical filters: in this case the weight of the classical filter(s) is determined using fuzzy rules. Examples: FMF, FDD and CK filters.

(2) The output of the fuzzy filter is obtained by using a classical formula: in this case the specific parameters of the formula are determined using fuzzy rules. Examples: FWM, WFM, AWFM1 and AWFM2 filters.

The other considered fuzzy filters are pure fuzzy filters, i.e. filters where the output is exclusively determined using fuzzy rules, independents of any classical filter. Examples: FIRE, DS-FIRE, PWL-FIRE, IFC, MIFC, EIFC, SFC, SSFC AND GOA Filters.

3.8.3 Fuzzy Rules

For every considered fuzzy filter it holds that the fuzzy rules determine the degree to which a pixel is a noise pixel or not. In general, this is done by investigating to which degree the processed pixel differs from its neighbours.

Most fuzzy filters explicitly use the gray-value differences between the processed pixel and its neighbours. This is the case for the FWM (which also takes distances into account), FIRE, DS-FIRE, PWL-FIRE, IFC, MIFC, EIFC, SFC, SSFC, GOA (where, with the purpose of edge detection, other gray-value differences are considered as well) and FDD filters. It is also the case for the CK filter (where distances are also taken into account), although this is done in a more complicated and less explicit way.

The FM, WFM, AWFM1 and AWFM2 filters are an exception: the fuzzy median filter uses the gray-value difference between the processed pixel and the median filtered version of this pixel; the WFM, AWFM1 and AWFM2 filters take the homogeneity of the neighbourhood into account in a different way (namely by using fuzzy sets dark, medium and bright).

3.8.4 Algorithm

Some of the fuzzy filters are applied iteratively or recursively. Note that an iterative filter also has a recursive character: after each iteration the algorithm is applied on the filtered pixels.

Notes: (1) Non-iterative filters can always be made interactive (one just applies the algorithm more than once); (2) Non-recursive filters can always be made recursive (after a pixel is filtered, one just replaces the original pixel by its filtered version); (3) the interactivity of a filter has a great impact on the computing time of that filter (the algorithm has to be applied several times), while this is not the case for recursive filters.

3.8.5 Complexity

The complexity of the considered fuzzy filters can be derived by analyzing the corresponding algorithms and by performing some experiments. In order to quantify the complexity of the filters, we will divide them in three classes: low, medium and high complexity.

(1) High complexity (computing time in experiments: above 10000 seconds): FWM filter. The high complexity is due to the calculation of the weight: (i) the weights are calculated iteratively until convergence is attained; (ii) some of the required parameters are also very complex to determine; (iii) the calculation have to be performed for each separate pixel.

(2) Low complexity (computing time in experiments: between 100 and 500 seconds): FM, FDD and PWL-FIRE filters. For the FM and FDD filters this follows from the construction (only gray-value differences are used, and straightforward formulas are available); the PWL-FIRE has a low complexity compared to the FIRE and DS-FIRE filter because, besides the relatively simple formulas, less different patterns of pixels are considered.

(3) Medium complexity (computing time in experiments: between 1000 and 3500 seconds): the formulas with the other fuzzy filters are more extended or complicated, which requires

more computing time. This is mainly due to the high number of function evaluations, resulting from the membership functions of the fuzzy sets involved.

Classification of fuzzy filter can be summarized in table 5.1. The different criteria can take the following values:

Noise type	: i = Impulse, g = Gaussian, b = both;
Degree of fuzziness	: m = mixed, p = pure;
Fuzzy rules	: d = differences(when based on gay-value differences), o = other;
Algorithm	: i = iterative, r = recursive, s = standard;
Complexity	: l = low, m = medium, h = high.

Table 3.1

Filter	Noise Type	Degree of Fuzziness	Fuzzy Rules	Algorithm	Complexity
FWM	g	m	d	s	h
FM	i	m	o	s	l
WFM	i	m	o	s	m
AWFM1	i	m	o	s	m
AWFM2	i	m	o	s	m
FIRE	i	p	d	r	m
DS-FIRE	i	p	d	r	m
PWL-FIRE	i	p	d	r	l
IFC	b	p	d	i	m
MIFC	b	p	d	i	m
EIFC	b	p	d	i	m
SFC	b	p	d	s	m
SSFC	b	p	d	s	m
GOA	g	p	d	i	m
FDD	i	m	d	s	l
CK	b	m	d	s	m

Based on these five criteria , the 16 considered fuzzy filters can be divided into 10 classes as follows:

(a) WFM, AFWM1 and AFWM2 (impulse, mixed, other, standard, medium);

(b) FIRE and DS-FIRE (impulse, pure, differences, recursive, medium);

(c) IFC, MIFC and EIFC (both, pure, differences, iterative, medium);

(d) SFC and SSFC (both, pure, differences, standard, medium);

(e) FWM (gaussian, mixed, differences, standard, high);

(f) FM (impulse, mixed, other, standard, low);

(g) PWL-FIRE (impulse, pure, differences, recursive, low);

(h) GOA ((gaussian, pure, differences, iterative, medium);

(i) FDD (impulse, mixed, differences, standard, low);

(j) CK (both, mixed, differences, standard, medium).

3.9 Conclusion

Regarding salt & pepper noise, the best filters are always fuzzy filters (DS-FIRE and PWL-FIRE for low noise levels, AWFM2 for higher noise levels). They clearly outperform the considered classical filters.

The best performing classical filter is the median filter. Note however that the fuzzy median filter (FM) performs much better than its classical counterpart, certainly for lower noise levels.

From the visual point of view, we have the following results:

(1) for low noise levels, most filters perform good, except the FWM, GOA, FDD, CK, mean, wiener and gaussian filters; (2) for higher noise level, the best visual results are obtained by the WFM, AWFM1, AWFM2, FM, DS-FIRE and PWL-FIRE and median filters.

Regarding gaussian noise, the best performing filters are fuzzy filters. It should be noted that the classical wiener (low noise levels) and the classical mean (high noise levels) are also among the best performing filters from a numerical point of view. In contrast to the case of salt & pepper noise, the FM filter is now outperformed by its classical counterpart.

From the visual point of view, the results are not so good compared to the case of salt & pepper noise. This is due to the more complicated nature of Gaussian noise. The AWFM2 and GOA filters generate the best visual results, both for low and high noise.

Based on the above five criteria , the fuzzy filters have been classified. This classification gives us a good insight in the technical differences between the considered filters, and can be a useful tool in selection procedures.

On the application level, the best performing filters are fuzzy filters. This clearly shows the usefulness of a fuzzy approach in the construction of filters for image noise reduction.

4. Implementation of fuzzy filters for image filtering

5.1 Introduction

In this chapter, six fuzzy filters for filtering images contaminated with random and impulse noises are introduced and implemented. Impulse noise is defined by noise density. The random noise is expressed in terms of its mean and variance values. Noise can be generated during image capture, transmission, storage, as well as during image copying, scanning, and display. For example, impulse noise can be generated through the T.V. broadcasting and due to the information losses; and the random noise can be generated during film exposure and development. Noise reduction in images has been one of the important tasks in image processing. For the case of impulse noise, most part of an original image is unaltered, and the image is characterized by some corrupted samples that vary drastically. As compared to impulse noise, random noise is a more challenging type of noise, the methods that are able to reduce random noise effectively in images are of interest.

Median filters can eliminate impulse noise, have good edge preserving ability, and have moderate noise attenuation ability in the flat regions of an image. The operations of a classical median (MED) filter involve the application of a window to move over an image and to replace the value at the centre pixel with the median of all the pixel values with in the window. In so doing, a pixel with a distinct intensity (in the case of impulse) as compared to those of its predefined neighbours will be eliminated. The implementation of a standard median filter is simple and the filter can process an image in a fast manner. However, the performance of a median filter is average for filtering random noise in an image.
Another nonlinear filtering technique called moving average (MAV) filters, which can smooth random noise, but can not suppress impulse noise, and can not preserve sharp edges of an image. The idea of standard moving average filter is to replace its centre pixel by the average value of its predefined neighbouring pixels, which can be easily implemented.

Each of the four fuzzy filters, applies a weighted membership function to an image with in a window to determine the centre pixel is introduced and implemented in this chapter include Gaussian fuzzy filter with median centre(GMED) , the symmetrical triangular fuzzy filter with median centre (TMED), the Gaussian fuzzy filter with moving average centre (GMAV), and the symmetrical triangular fuzzy filter with moving average centre (TMAV).

4.2 Definitions of Fuzzy Filters

Let x(i, j) be the input of a 2-dimentional fuzzy filter , the output of the fuzzy filter is defined as:

$$y(i,j) = \frac{\sum_{(r,s)\in A} F[X(i+r,\ j+s)].X(i+r,\ j+s)}{\sum_{(r,s)\in A} F[X(i+r,\ j+s)]} \qquad (4.1)$$

where F[x(i, j)] is the general window function and A is the area of the window. For a square window of dimensions N X N, the range of r and s are: $-R \leq r \leq R$ and $-S \leq s \leq S$, where $N = 2R + 1 = 2S + 1$.

With the definitions of different window functions, we can define six fuzzy filters, which we shall call the Gaussian fuzzy filter with median centre(GMED) , the symmetrical triangular fuzzy filter with median centre (TMED), the Gaussian fuzzy filter with moving average centre (GMAV), and the symmetrical triangular fuzzy filter with moving average centre (TMAV). The standard median filter(MED) and the moving average filter (MAV) are special cases of the fuzzy filters and we shall define them as follows.

4.2.1 Median Filter

In the case of a standard median filter, the window function is defined as:

$$F_{med} = \begin{cases} \mathbf{1\ for\ } \boldsymbol{X(i+r, j+s) = X\,med(i, j)} \\ \boldsymbol{o\ otherwise} \end{cases} \tag{4.2}$$

Such that the output value at the centre of a window y(i, j) is replaced by the median value X_{med}(i, j) among all the input values x(i + r, j + s) for r, s ε A in the window A at the discrete indexes (i, j).

4.2.2 Moving Average Filter

In a standard moving average filter, the window function is defined as:

$$F_{mav}[x(i + r, j + s)] = 1 \quad \text{for } r, s\ \varepsilon\ A \tag{4.3}$$

The moving average filter is equivalent to a 2-dimensional rectangular-shape fuzzy filter covering all the input values x(i + r, j + s) for r, s ε A in the window A .

4.2.3 Gaussian fuzzy filter with median centre

The Gaussian fuzzy filter with the median value within a window chosen as the centre value is defined as:

$$F_{gmed}\,[x(i + r, j + s)] = e^{-\frac{1}{2}\left[\frac{x(i+r,j+s)-Xmed(i,j)}{\sigma(i,j)}\right]^2} \quad \text{for } r, s\ \varepsilon\ A \tag{4.4}$$

Where X_{med}(i, j) and σ(i, j) represent, respectively, the median value and the variance value of all the input values x(i + r, j + s) for r, s ε A in the window A at the discrete indexes (i, j).

4.2.4 Symmetrical triangle fuzzy filter with median centre

The symmetrical triangle fuzzy filter with the median centre value within a window chosen as the centre value is defined as:

$$F_{tmed}[x(i+r, j+s)] = \begin{cases} \mathbf{1 - \frac{|X(i+r,j+s) - Xmed(i,j)|}{Xmm(i,j)}\ for|X(i+r, j+s) - Xmed(i,j)|} \\ \mathbf{1\ for\ Xmm = 0} \end{cases} \Big\}$$

(4.5)

Where,

Xmm(i, j) = max[Xmax(i, j) – Xmed(i, j), Xmed(i, j) - Xmin(i, j)]

Xmax(i, j), Xmin(i, j) and Xmed(i, j) are, respectively, the maximum value, the minimum value, and the median value of all the input values x(i + r, j + s) for r, s ε A within the window A at the discrete indexes (i, j).

4.2.5 Gaussian fuzzy filter with moving average centre

The Gaussian fuzzy filter with the moving average value within a window chosen as the centre value is defined as:

$$F_{\boldsymbol{gmav}}[x(i+r, j+s)] = e^{-\frac{1}{2}\left[\frac{x(i+r,j+s) - Xmav(i,j)}{\sigma(i,j)}\right]^2} \quad \text{for } r, s\ \varepsilon\ A \qquad (4.6)$$

Where X*mav*(I,j) and σ(I,j) represent, respectively, the moving average value and the variance value of all the input values x(i + r, j + s) for r, s ε A in the window A at the discrete indexes (i, j).

4.2.6 Symmetrical triangle fuzzy filter with moving average centre

The symmetrical triangle fuzzy filter with the moving average value within a window chosen as the centre value is defined as:

$$Ftmav[x(i+r,j+s)] = \left\{ \begin{array}{ll} 1 - \dfrac{|x(i+r,j+s) - Xmav(i,j)|}{Xmv(i,j)} & \\ \quad for\ |x(i+r,j+s) - Xmav(i,j)| \leq Xmv(i,j) & \\ & \\ 1 \qquad for\ Xmv = 0 & \end{array} \right\}$$

(4.7)

Where,

Xmv(i, j) = max[Xmax(i, j) - Xmav(i, j), Xmav(i, j) - Xmin(i, j)]

Xmax(i, j), Xmin(i, j) and Xmav(i, j) represent, respectively, the maximum value, the minimum value, and the moving average value of *x(i + r, j + s)* with in the window A at discrete indexes *(I,j)*.

4.3 Experimental Results

In this section, we present experimental results with multiple commonly used greyscale test images to assess the performance of above fuzzy filters. In all the computer simulations, two 8-bit mono images of dimensions M1XM2 (=256X256) pixels are used. In each of the images, the pixels s(i ,j) for $1 \leq i \leq M1$ and $1 \leq j \leq M2$, are corrupted by adding two kinds of noise namely, impulse(salt & peppers) and random(Gaussian) noise. The standard two test images are Lena and Red house are used as shown in figure 5.1-5.2. Low, medium, and high levels of impulse noise, with respective density values of 0.03, 0.15, and 0.3 are added to each of these two images as shown in figure5.3-5.5 for Lena image, figure5.45-5.47 for Red house image. Also, low, medium, and high levels of random noise, each has a mean value of 0.0 and a respective variance value of 0.0052, 0.021, and 0.106 is added to each of two images as shown in figure5.6-5.8 for Lena image, figure5.48-5.50 for Red house image.

In all the computer simulations, square window of dimensions 3X3 pixels are used. The mean squared error (MSE) is used to compare the relative filtering performance of above filters. The MSE between the filtered output image y(i, j) and the original image s(i, j) of dimensions M1XM2 pixels is defined as:

$$\text{Mean Squared Error (MSE)} = \frac{\sum_{i}^{M1} \sum_{j}^{M2} [y(i,j)-s(i,j)]^2}{M1.M2} \qquad (4.8)$$

The Mean Squared Error (MSE) of the original and filtered noisy Lena, and Red house images for the 3 levels of impulse noise and the 3 levels of random noise for a square window of dimensions 3X3 are respectively summarized in Tables 5.1-5.2 and Tables 5.3-5.4,

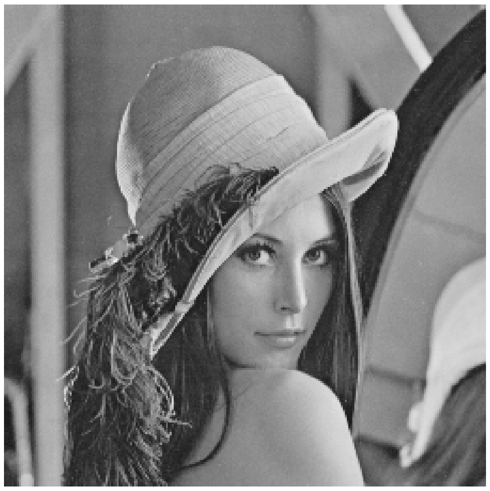


Fig.5.1. Original Lena image

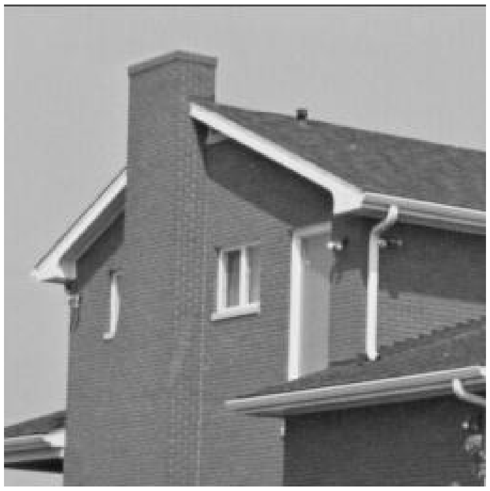


Fig.5.2. Original Red House image

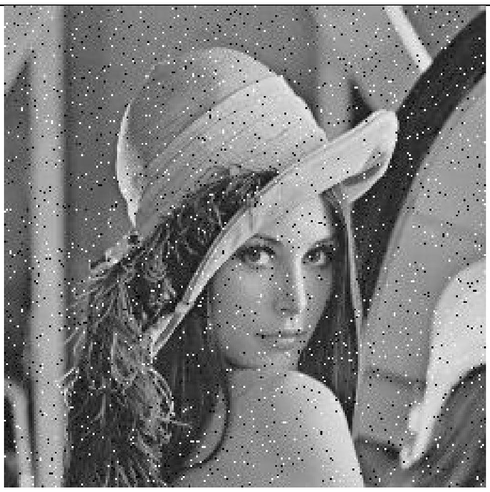


Fig.5.3 Lena with low impulse noise

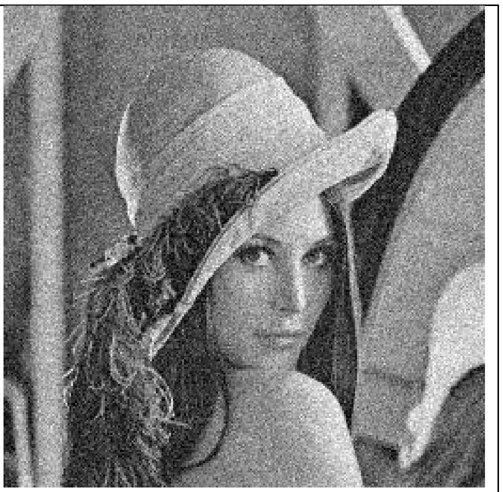


Fig5.6 Lena with low random noise

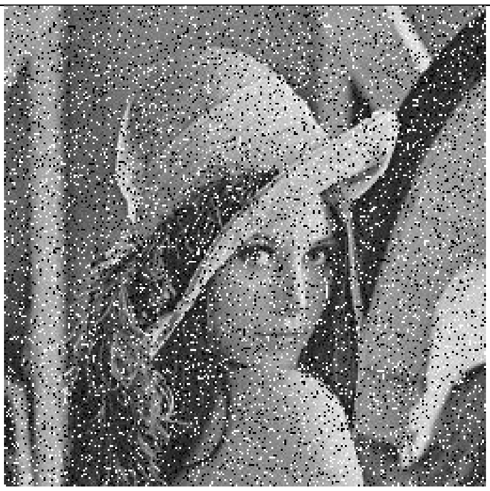


Fig.5.4 Lena with medium impulse noise

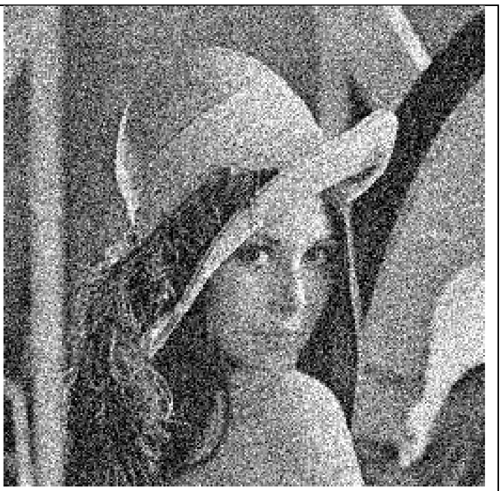


Fig5.7 Lena with medium random noise

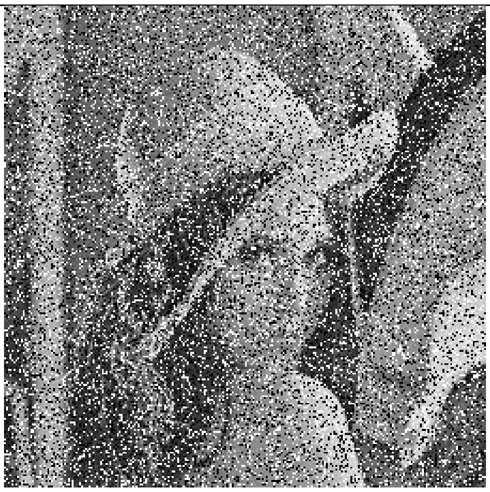


Fig.5.5 Lena with high impulse noise

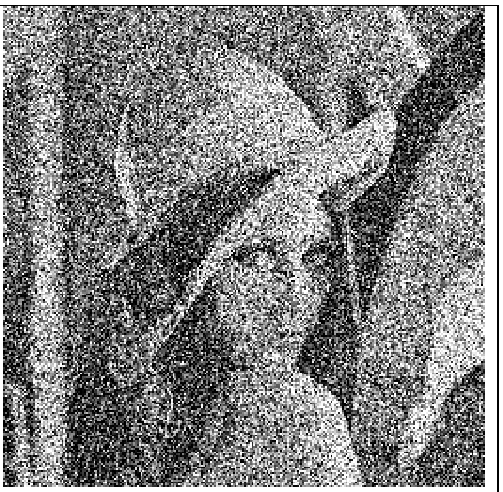


Fig5.8 Lena with high random noise

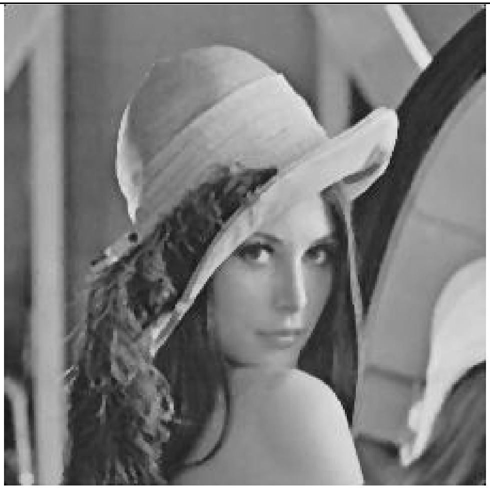


Fig.5.9 MED - low impulse noise

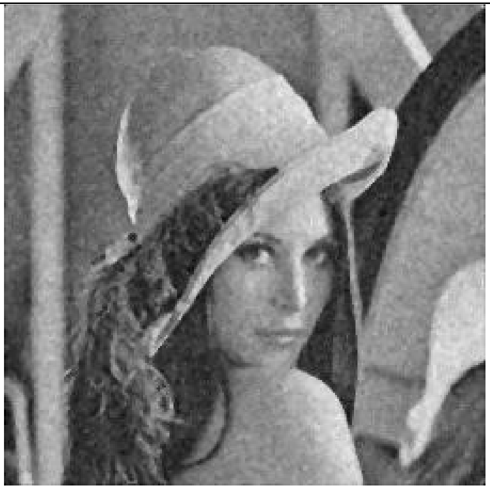


Fig5.10 MED - low random noise

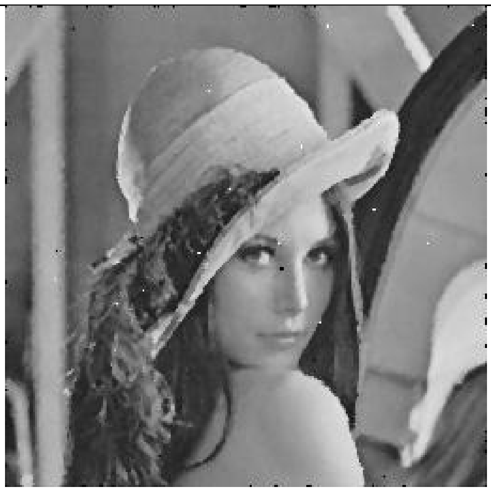


Fig.5.11 MED - Medium impulse noise

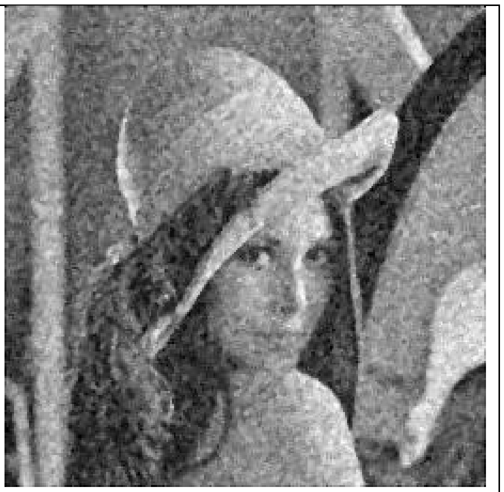


Fig5.12 MED - Medium random noise

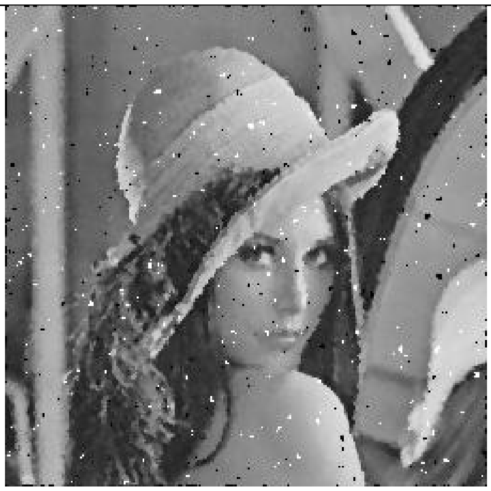


Fig.5.13 MED - High impulse noise

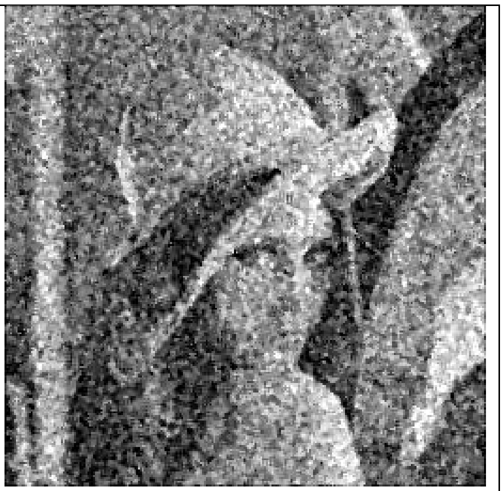


Fig5.14 MED - High random noise

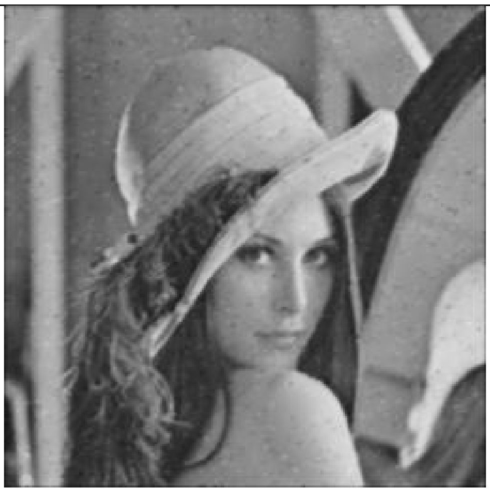


Fig.5.15 GMED - Low impulse noise

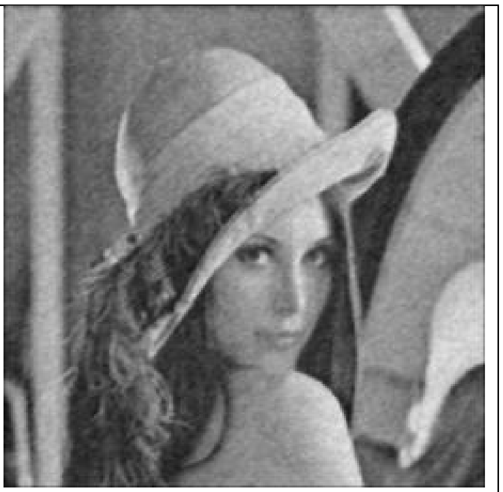


Fig5.16 GMED - Low random noise

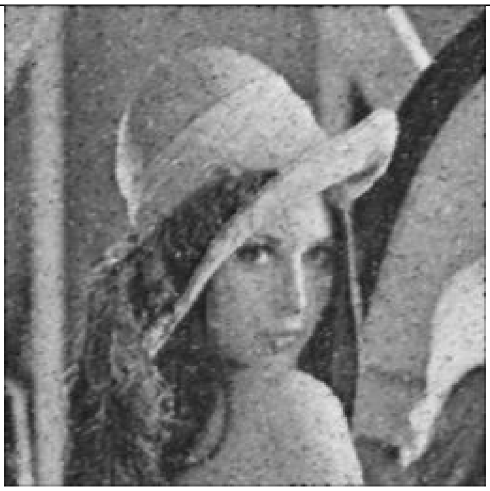


Fig.5.17 GMED - Medium impulse noise

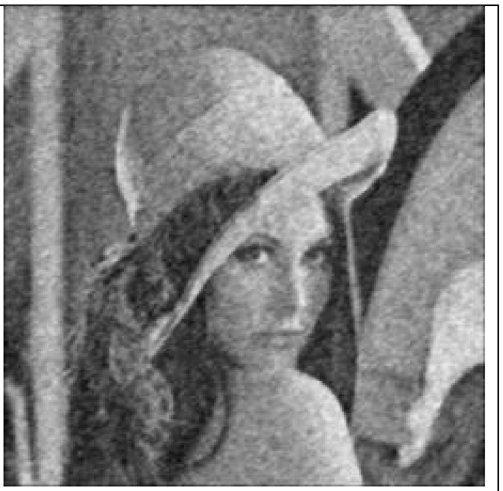


Fig5.18 GMED - Medium random noise

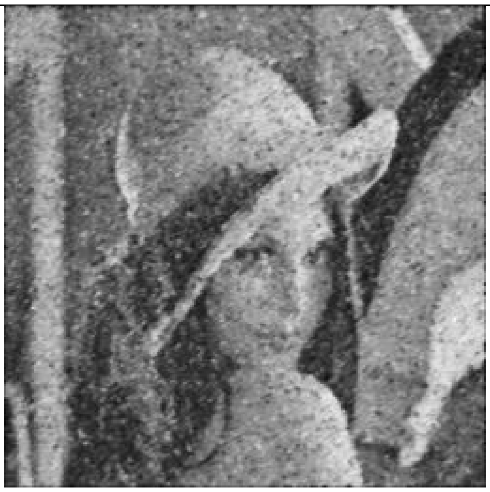


Fig.5.19 GMED - High impulse noise

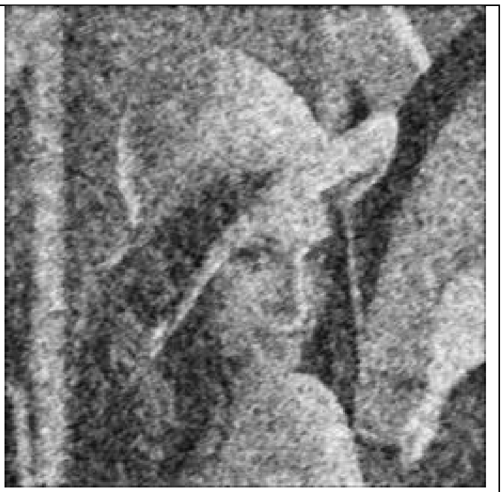


Fig5.20 GMED - High random noise

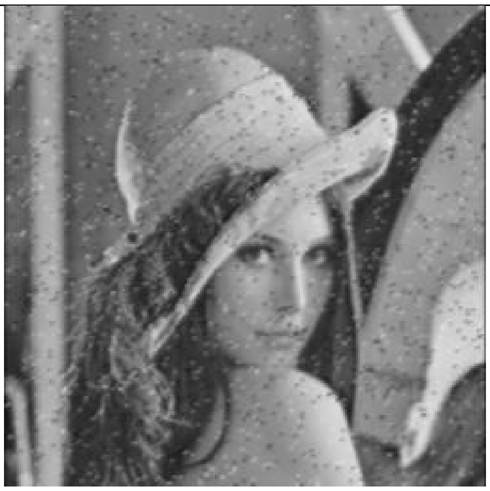


Fig.5.21 TMED - Low impulse noise

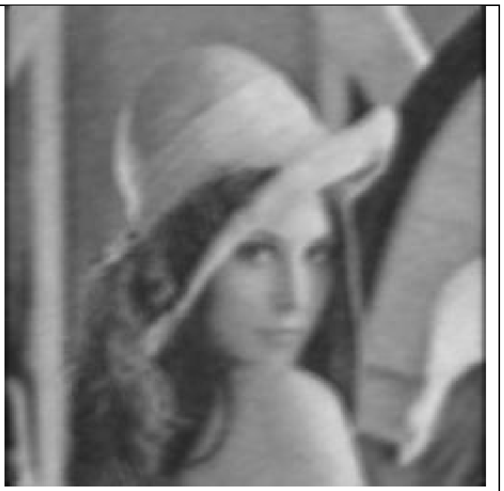


Fig5.22 TMED- Low random noise

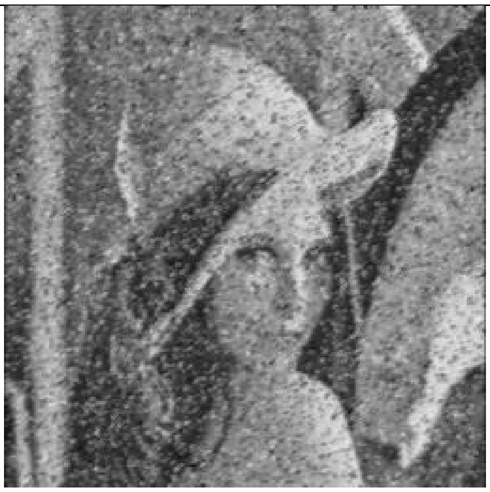


Fig.5.23 TMED – Medium impulse noise

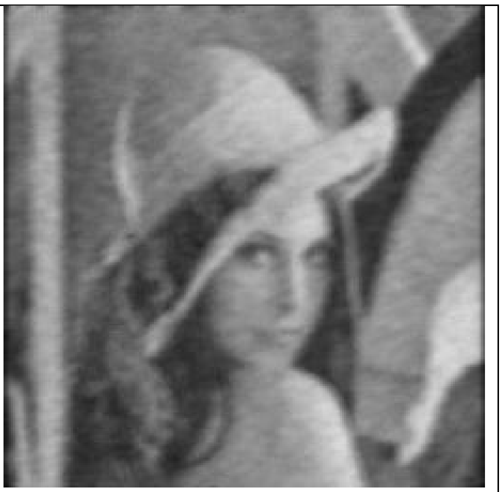


Fig5.24 TMED- Medium random noise

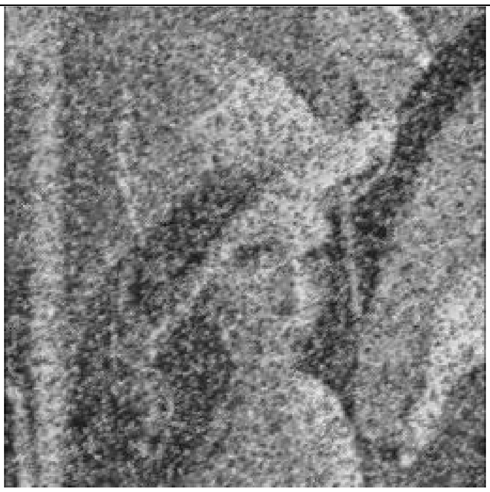


Fig.5.25 TMED - High impulse noise

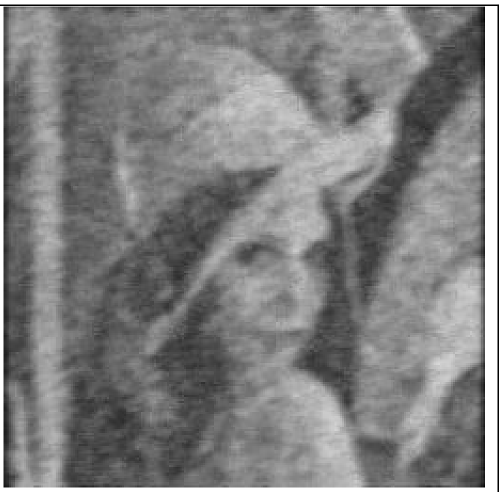


Fig5.26 TMED- High random noise

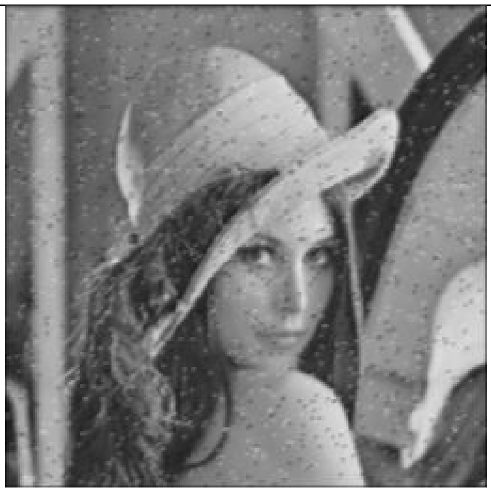


Fig.5.27 MAV - Low impulse noise

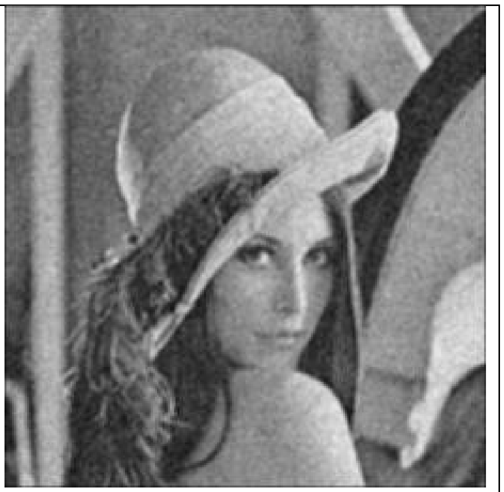


Fig5.28 MAV - Low random noise

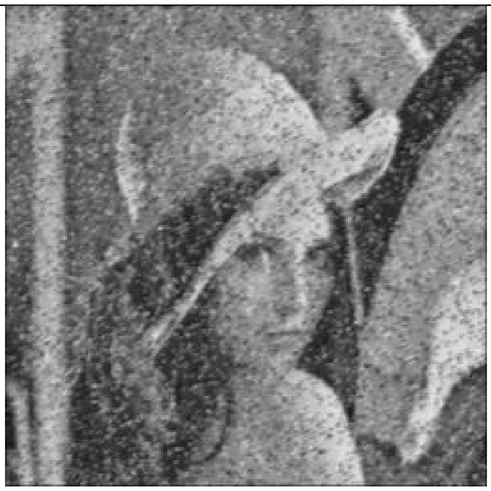


Fig.5.29 MAV - Medium impulse noise

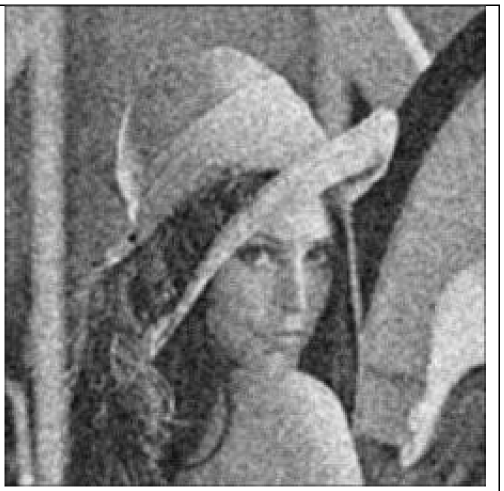


Fig5.30 MAV - Medium random noise

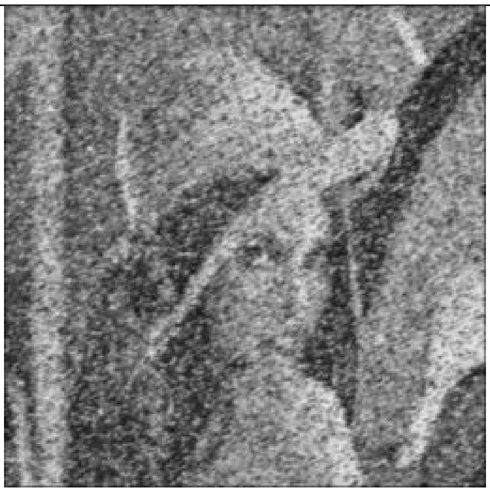


Fig.5.31 MAV - High impulse noise

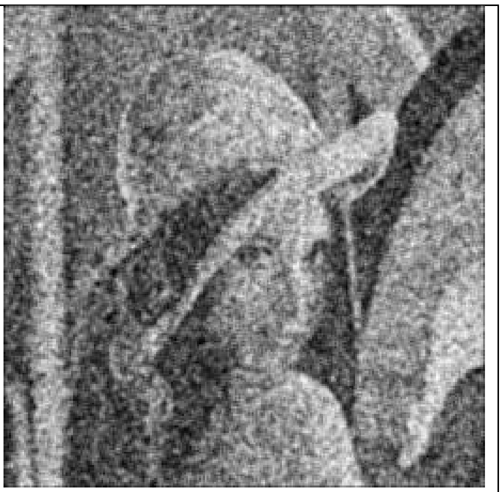


Fig5.32 MAV - High random noise

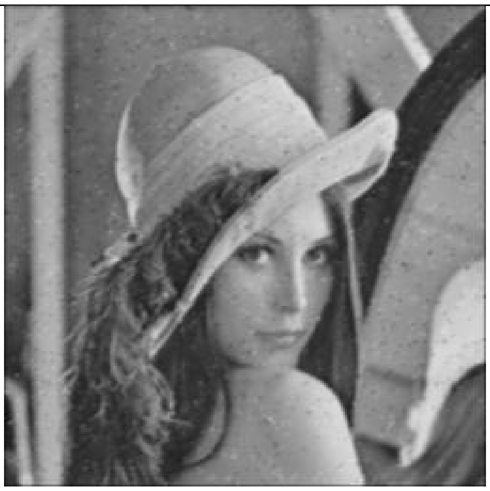


Fig.5.33 GMAV - Low impulse noise

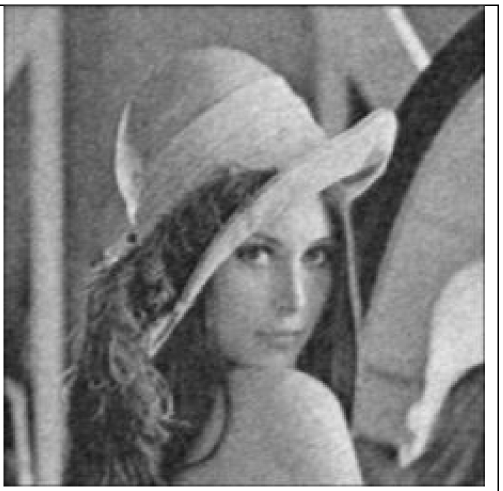


Fig5.34 GMAV - Low random noise

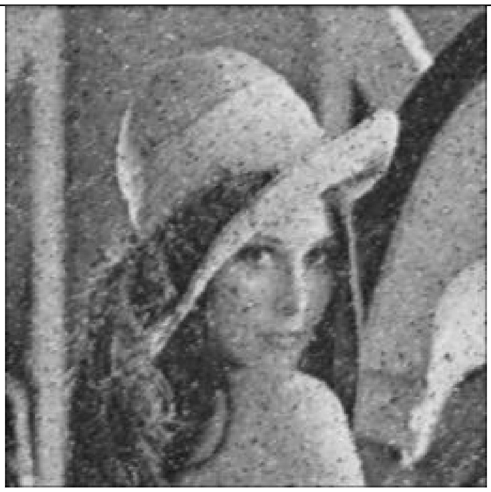


Fig.5.35 GMAV - Medium impulse noise

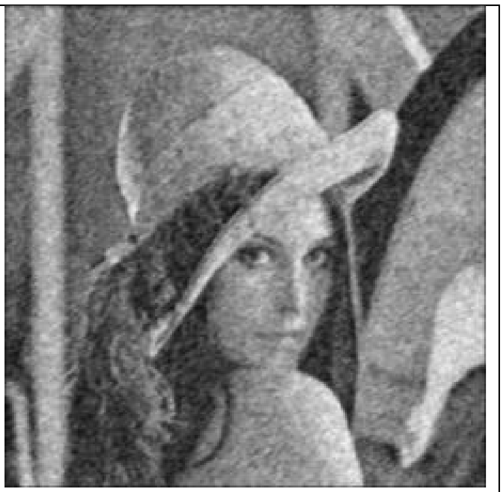


Fig5.36 GMAV - Medium random noise

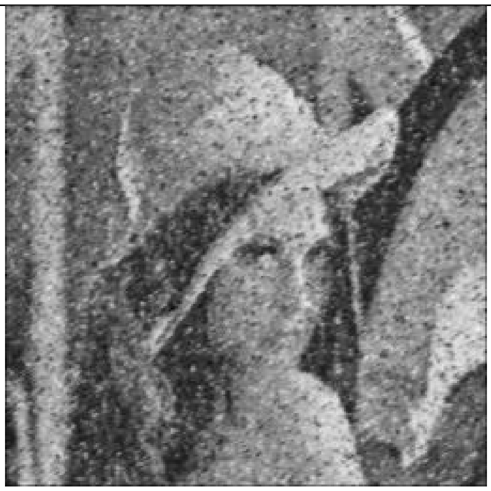


Fig.5.37 GMAV - High impulse noise

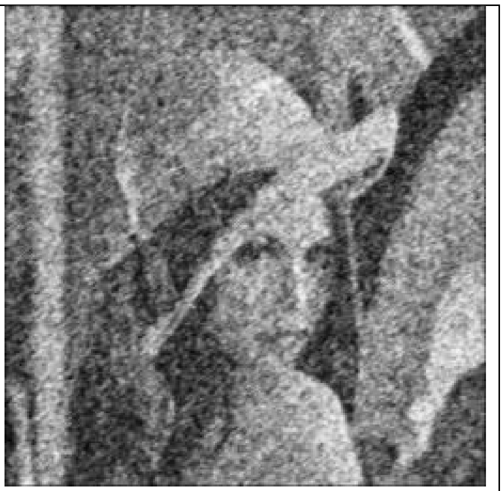


Fig5.38 GMAV - High random noise

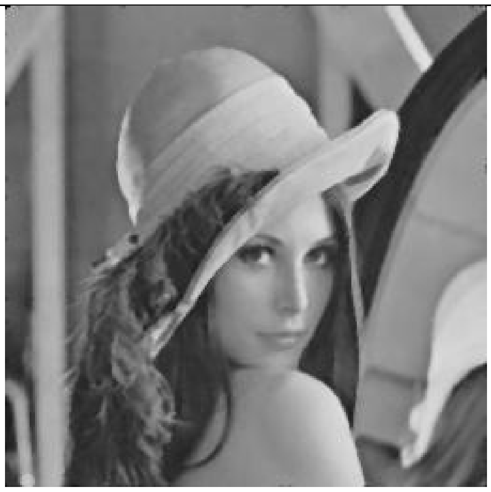


Fig.5.39 TMAV - Low impulse noise

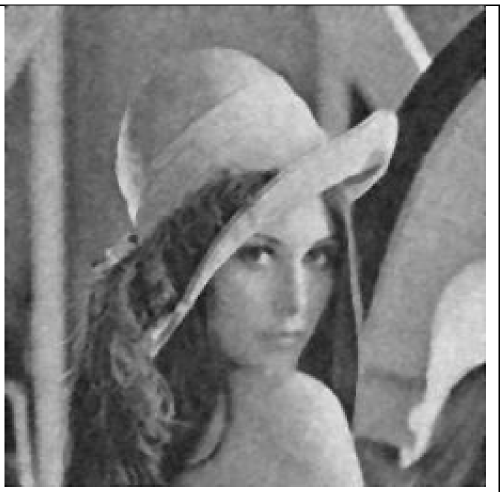


Fig5.40 TMAV - Low random noise

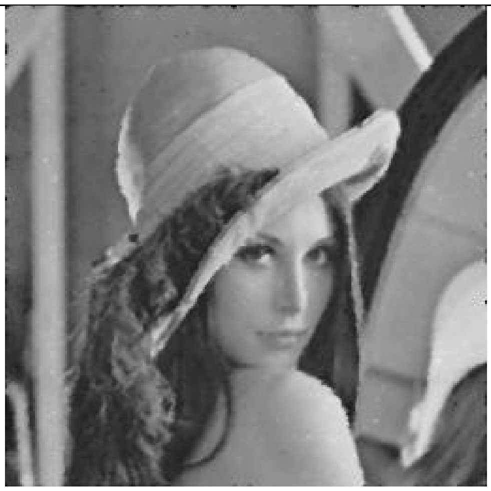


Fig.5.41 TMAV - Medium impulse noise

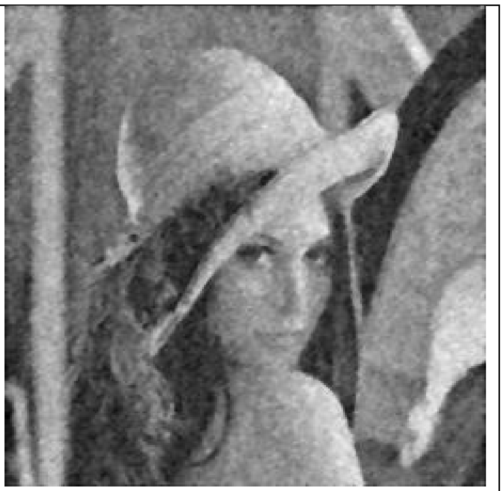


Fig5.42 TMAV - Medium random noise

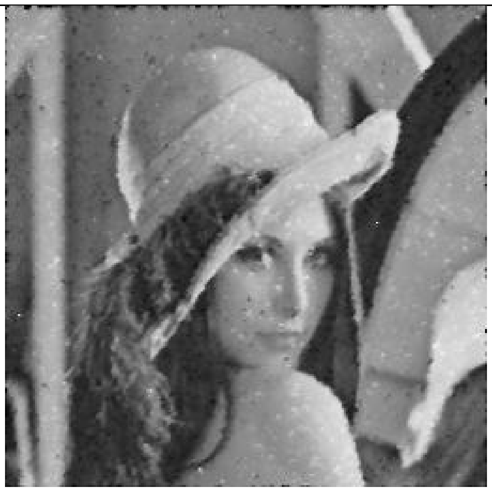
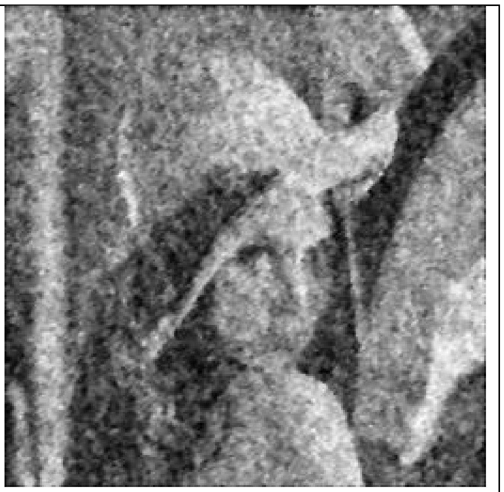



Fig.5.43 TMAV - High impulse noise Fig5.44 TMAV - High random noise

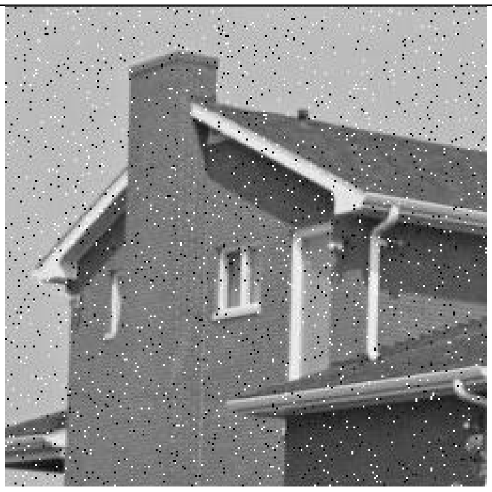
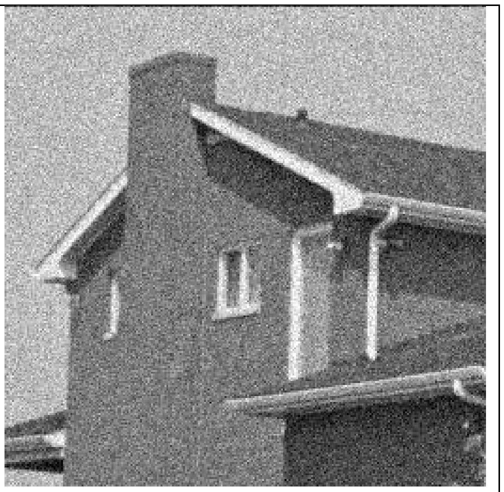



Fig.5.45 Red house with low impulse noise Fig5.48 Red house with low random noise

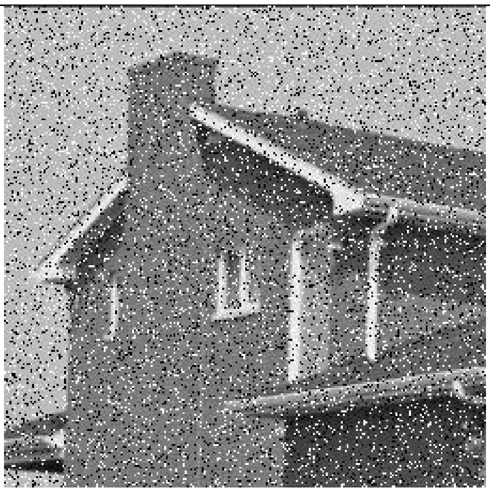


Fig.5.46 Red house with medium impulse noise

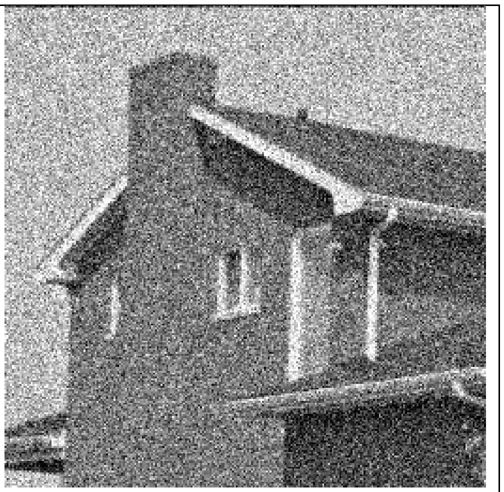


Fig5.49 Red house with medium random noise

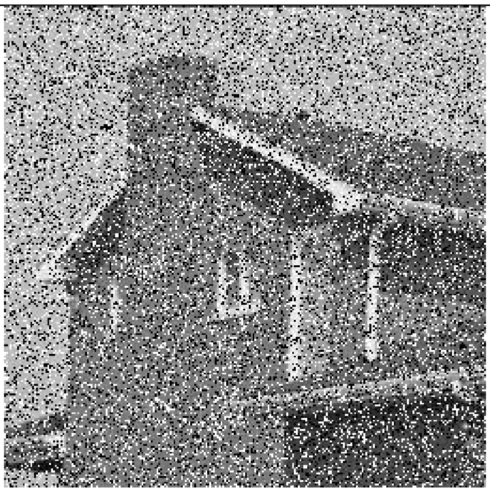


Fig.5.47 Red house with high impulse noise

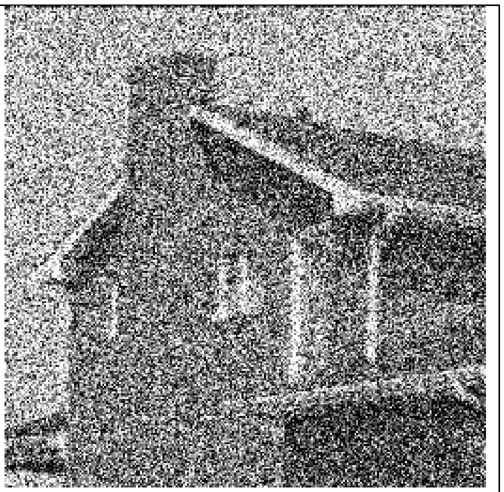


Fig5.50 Red house with high random noise

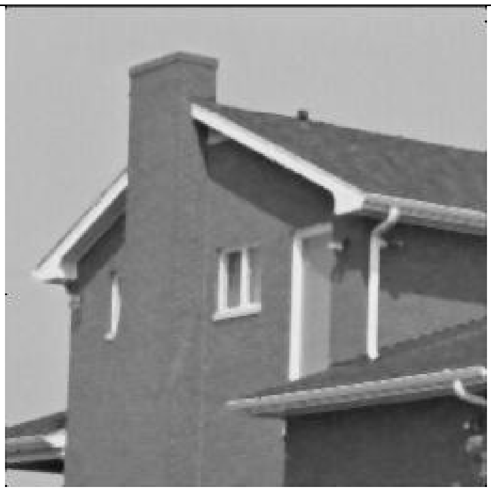


Fig.5.51 MED - Low impulse noise

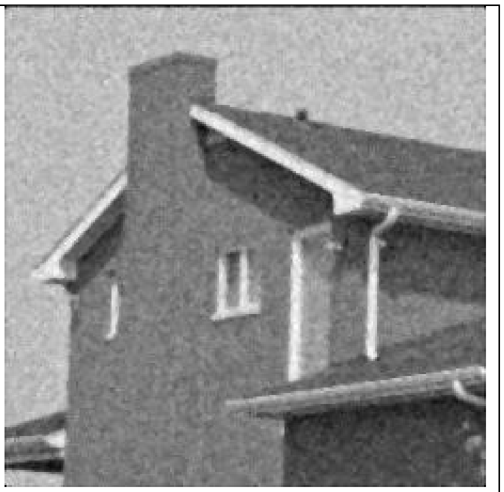


Fig5.52 MED - Low random noise

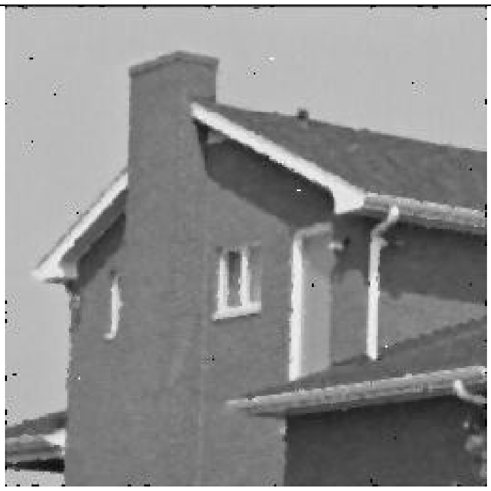


Fig.5.53 MED – Medium impulse noise

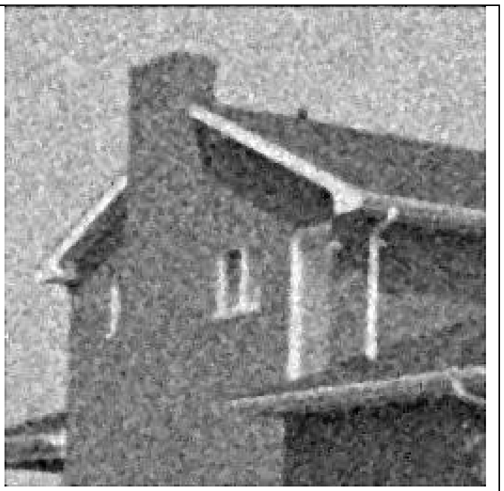


Fig5.54 MED - Medium random noise

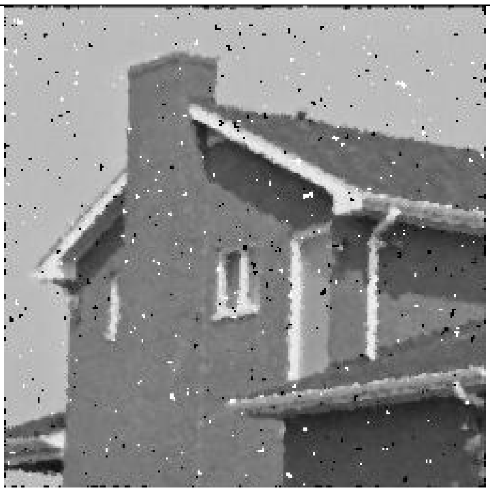


Fig.5.55 MED – High impulse noise

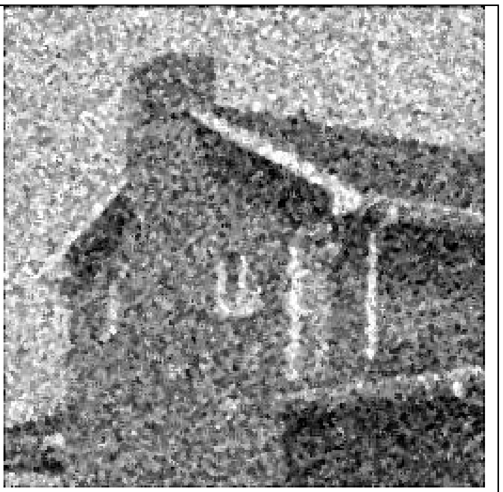


Fig5.56 MED - high random noise

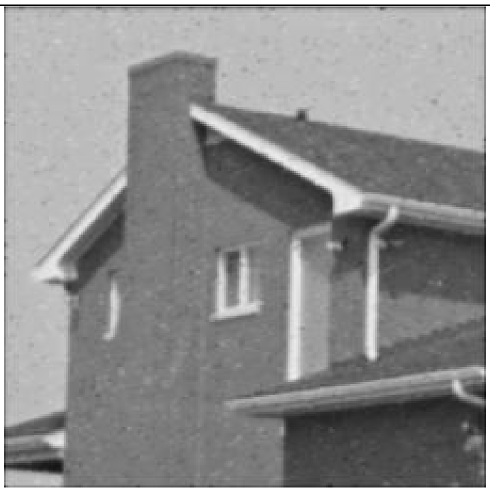


Fig.5.57 GMED – Low impulse noise

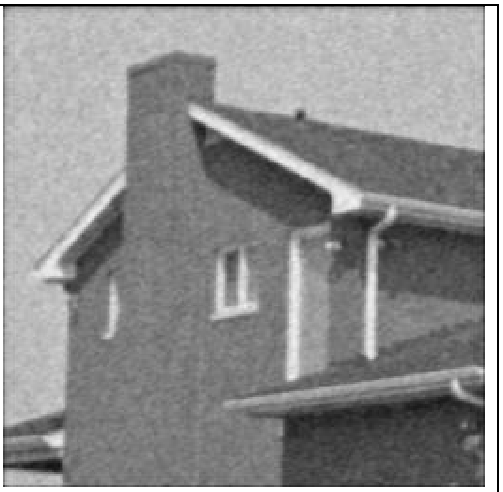


Fig5.58 GMED - Low random noise

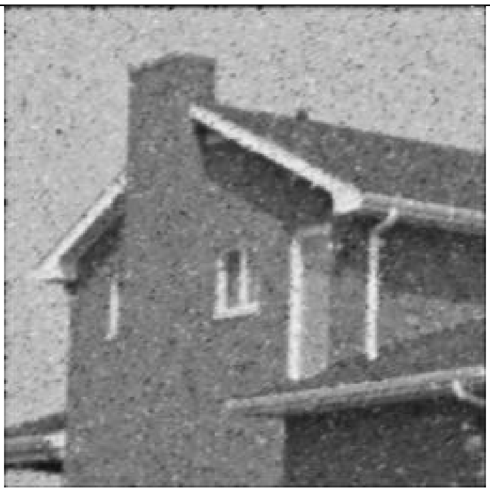


Fig.5.59 GMED – Medium impulse noise

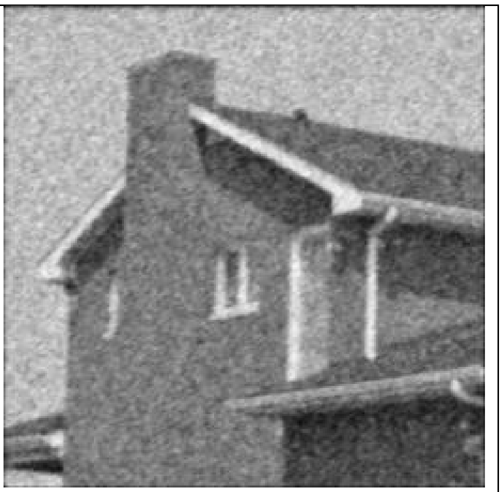


Fig5.60 GMED - Medium random noise

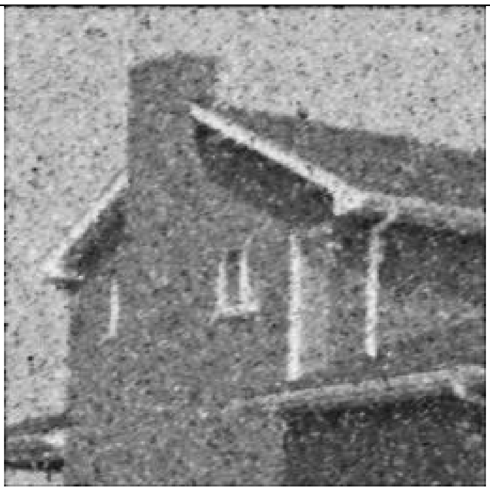


Fig.5.61 GMED – High impulse noise

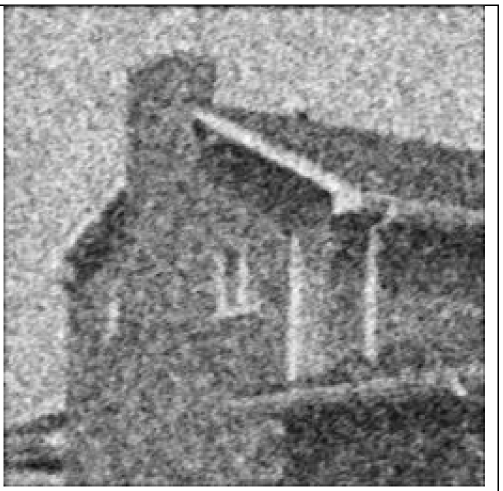
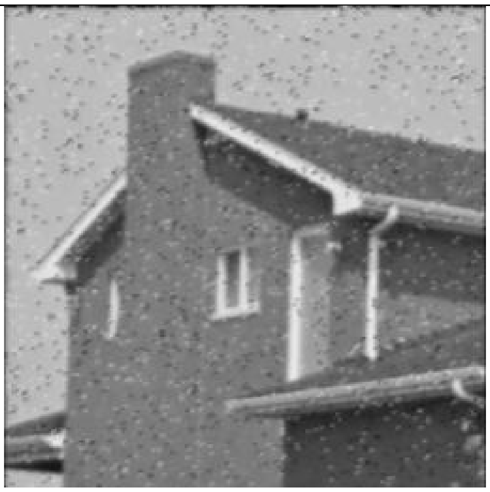


Fig5.62 GMED - High random noise



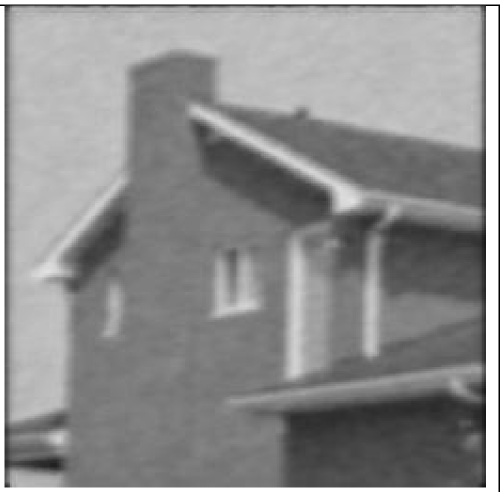
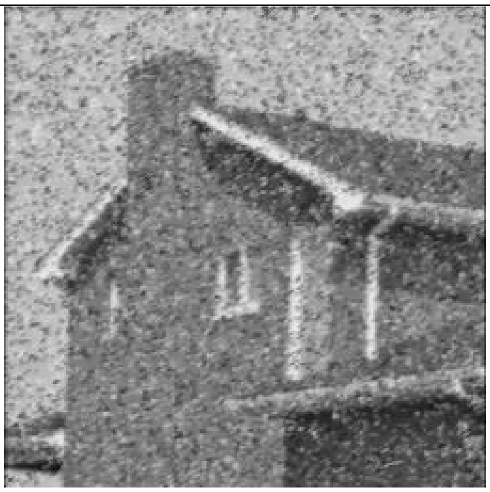


Fig.5.63 TMED – Low impulse noise   Fig5.64 TMED - Low random noise



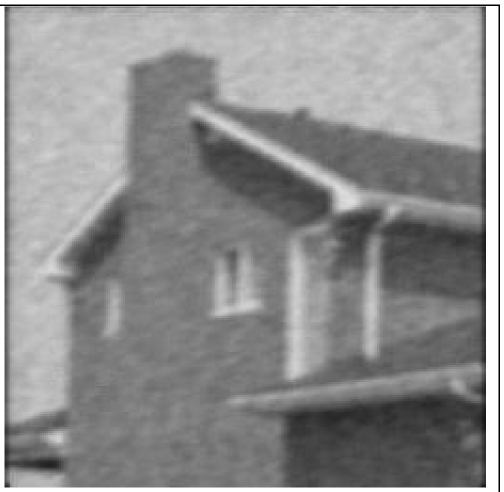


Fig.5.65 TMED – Medium impulse noise   Fig5.66 TMED - medium random noise

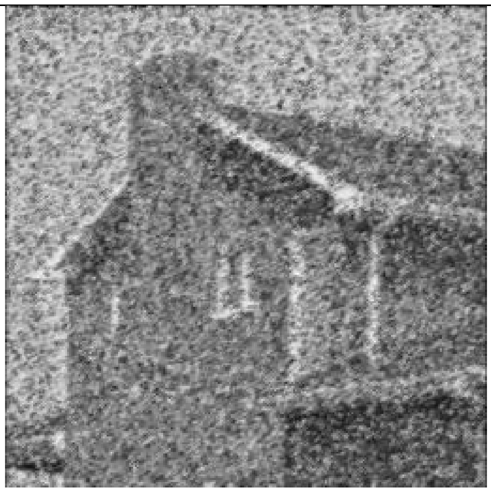


Fig.5.67 TMED – High impulse noise

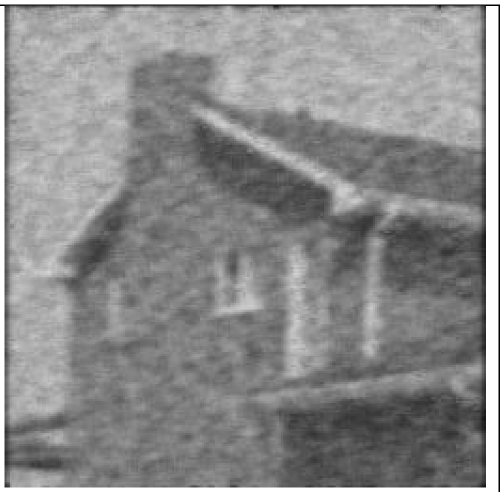


Fig5.68 TMED - High random noise

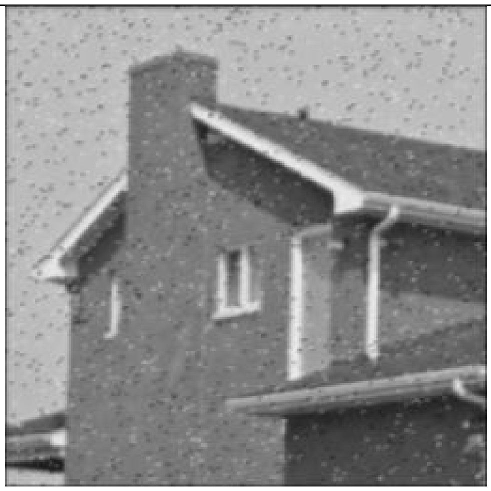


Fig.5.69 MAV – Low impulse noise

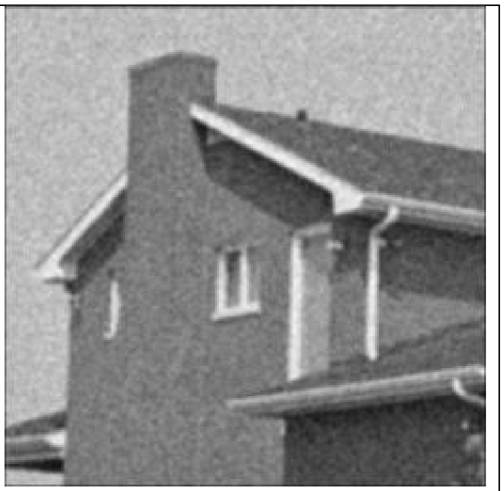


Fig5.70 MAV - Low random noise

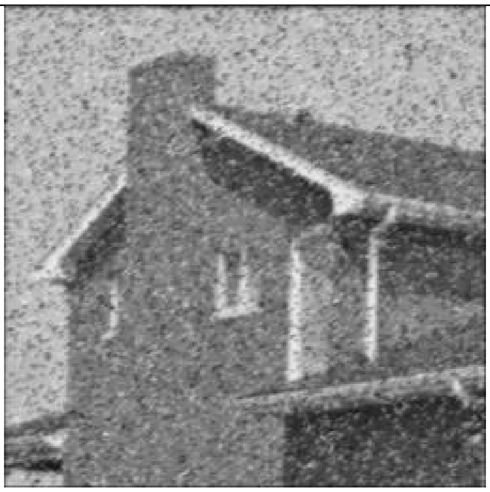


Fig.5.71 MAV – Medium impulse noise

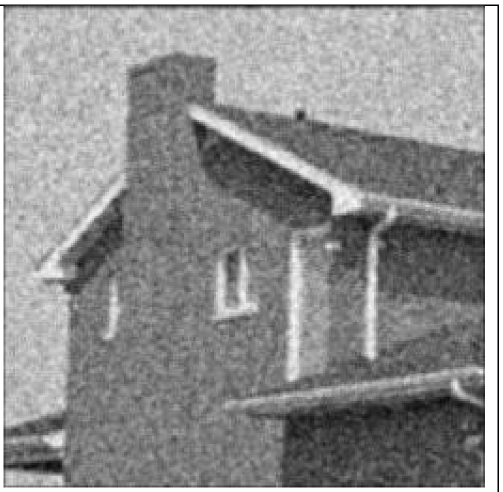


Fig5.72 MAV - Medium random noise

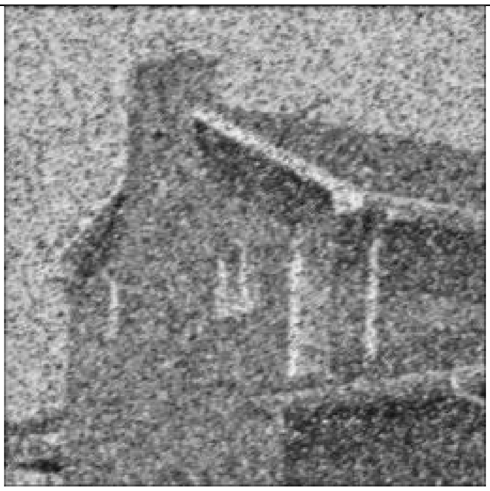


Fig.5.73 MAV – High impulse noise

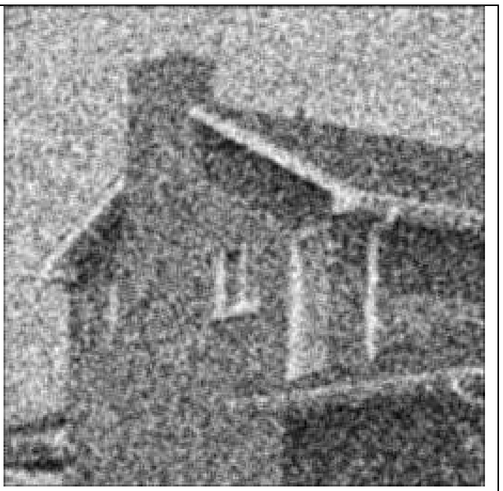


Fig5.74 MAV - High random noise

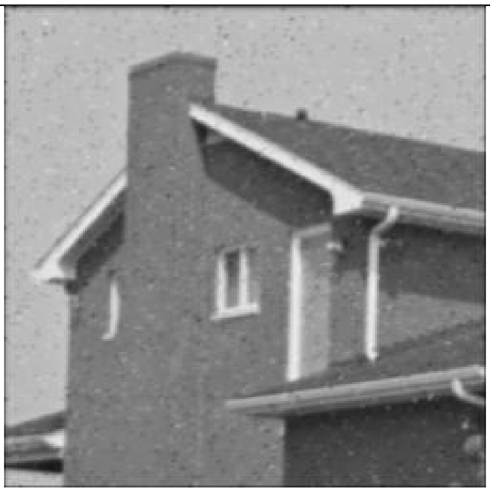


Fig.5.75 GMAV – Low impulse noise

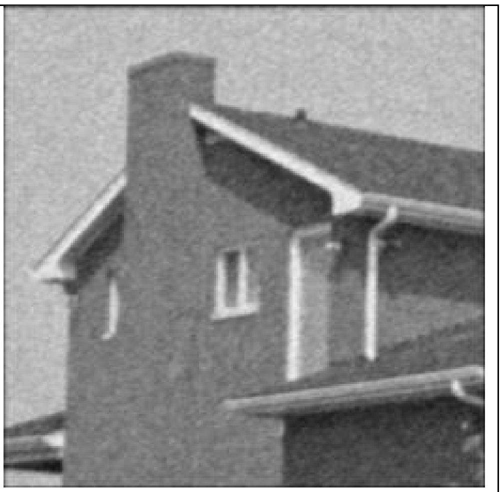


Fig5.76 GMAV - Low random noise

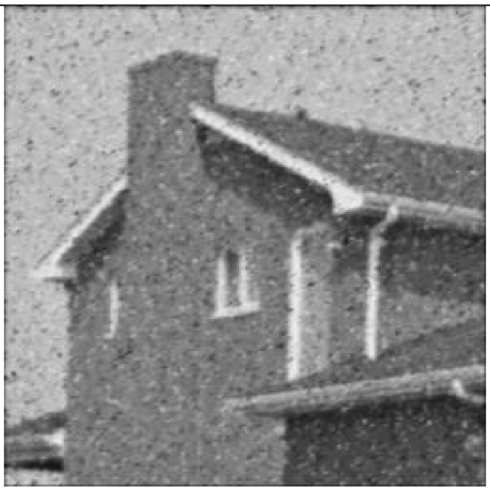


Fig.5.77 GMAV – Medium impulse noise

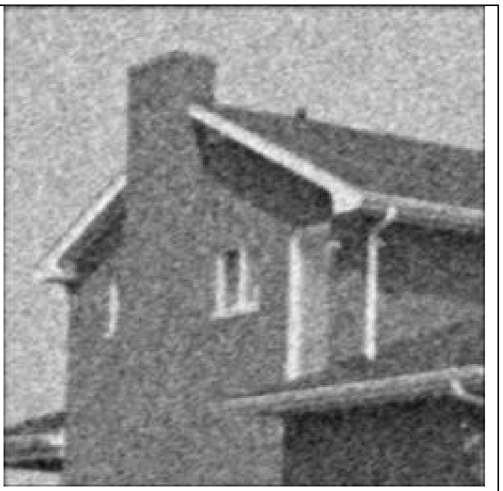


Fig5.78 GMAV - Medium random noise

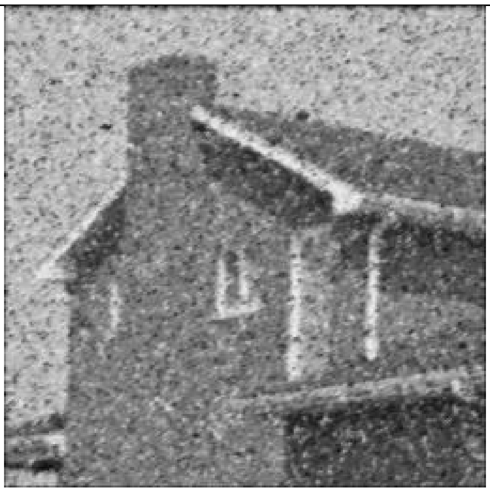


Fig.5.79 GMAV – High impulse noise

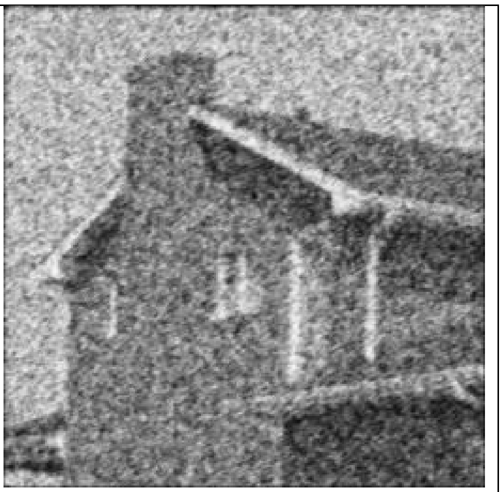


Fig5.80 GMAV - High random noise

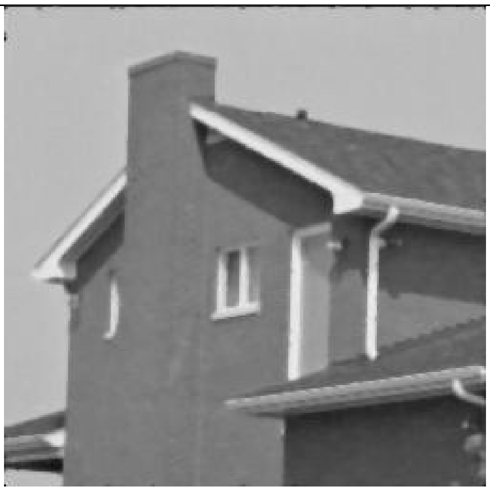


Fig.5.81 TMAV – Low impulse noise

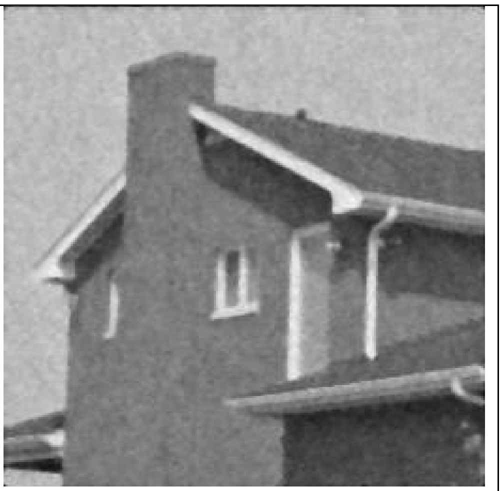
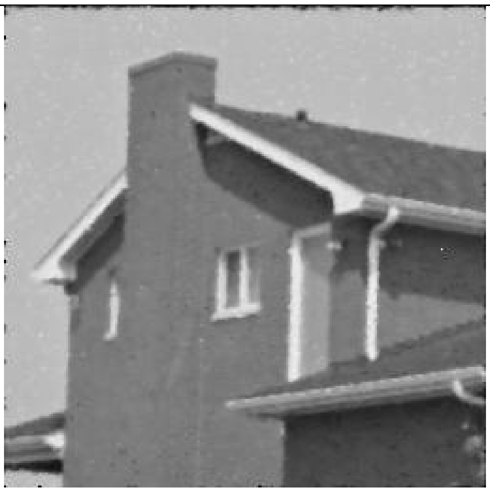


Fig5.82 TMAV - Low random noise



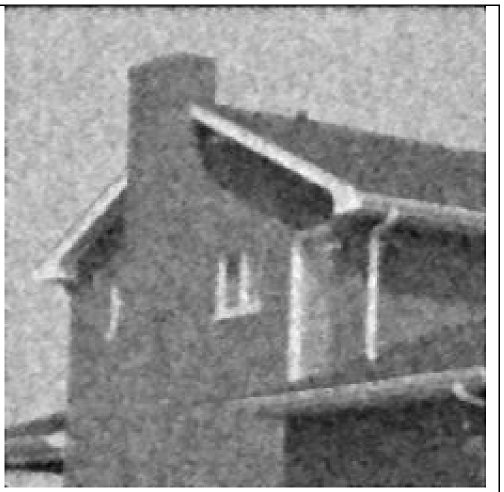
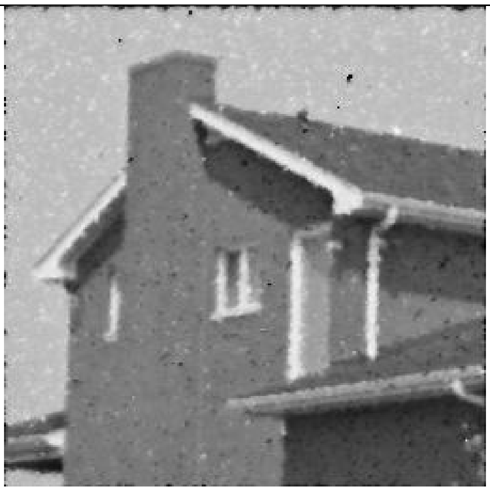


Fig.5.83 TMAV – Medium impulse noise   Fig5.84 TMAV - Medium random noise



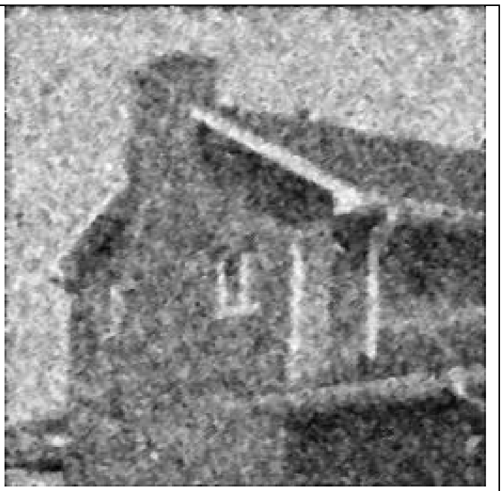


Fig.5.85 TMAV – High impulse noise   Fig5.86 TMAV - High random noise

Table 4.1

MSE of original and filtered noisy Lena images contaminated with impulse noise.

Filter	Density of Impulse Noise		
	Low=0.03	Medium=0.15	High=0.3
Noisy Image	540.5311	2794.0	5541.9
MED	70.0980	129.3456	383.4166
GMED	123.3544	223.3185	435.1084
TMED	191.1272	539.5587	1085.8
MAV	191.9377	478.4932	897.6409
GMAV	145.6020	287.8949	574.0851
TMAV	83.5318	118.8555	206.7520

Table 4.2

MSE of original and filtered noisy Lena images contaminated with random noise.

Filter	Variance of Random Noise		
	Low=0.0052	Medium=0.021	High=0.106
Noisy Image	336.3140	1305.4	4955.5
MED	136.4663	332.8569	1260.3
GMED	144.1659	239.5822	630.9635
TMED	355.4011	379.2538	603.9722
MAV	155.1720	262.1830	731.7963
GMAV	158.1746	250.4558	654.8163
TMAV	34.1012	57.3798	86.3799

Table 4.3

MSE of original and filtered noisy Red house images contaminated with impulse noise.

Filter	Density of Impulse Noise		
	Low=0.03	Medium=0.15	High=0.3
Noisy Image	572.0305	2809.8	5575.8
MED	29.4451	78.0516	377.4860
GMED	77.6550	183.5677	400.1558
TMED	169.4636	529.7220	1021.0
MAV	148.9322	435.9344	868.3545
GMAV	102.2067	235.0774	529.3206
TMAV	37.0743	68.9708	161.3956

Table 4.4

MSE of original and filtered noisy Red house images contaminated with random noise.

Filter	Variance of Random Noise		
	Low=0.0052	Medium=0.021	High=0.106
Noisy Image	338.7111	1308.2	5003.1
MED	94.3424	281.7866	1217.8
GMED	98.7944	196.4579	566.7820
TMED	300.5441	320.7529	538.3957
MAV	110.8082	222.1876	693.8959
GMAV	112.8209	207.6157	620.5429
TMAV	27.0361	50.2798	86.8388

Table 4.5

MSE ranking of filtered Lena images contaminated with Low-Medium-High level of impulse noise.

Filters	Low	Medium	High
MED	1	2	2
GMED	3	3	3
TMED	4	6	6
MAV	5	5	5
GMAV	6	4	4
TMAV	2	1	1

Table 4.6

MSE ranking of filtered Lena images contaminated with Low-Medium-High level of random noise.

Filters	Low	Medium	High
MED	2	5	6
GMED	3	2	3
TMED	6	6	2
MAV	4	4	5
GMAV	5	3	4
TMAV	1	1	1

Table 4.7

MSE ranking of filtered Red house images contaminated with Low-Medium-High level of impulse noise.

Filters	Low	Medium	High
MED	1	2	2
GMED	3	3	3
TMED	6	6	6
MAV	5	5	5
GMAV	4	4	4
TMAV	2	1	1

Table 4.8

MSE ranking of filtered Red house images contaminated with Low-Medium-High level of random noise.

Filters	Low	Medium	High
MED	2	5	6
GMED	3	2	3
TMED	6	6	2
MAV	4	4	5
GMAV	5	3	4
TMAV	1	1	1

5. Gaussian noise reduction using fuzzy filter combining sharpening and noise reduction

5.1 Introduction

Images are frequently corrupted by noise due to errors generated in noisy sensors and communication channels. Improving the quality of sensor data is a key issue in image based instrumentation. Indeed, preprocessing techniques can play a very relevant role in increasing the accuracy of subsequent tasks such as parameter estimation and object recognition. In this respect, contrast enhancement is often necessary in order to highlight important features embedded in the image data. The enhancement of noisy data, however, is a very critical process because the sharpening operation can significantly increase the noise.

In this chapter, a new nonlinear method for the enhancement of noisy data is presented. The proposed approach consists in a multiple output processing system that adopts fuzzy networks in order to combine contrast enhancement and noise reduction. Indeed, the fuzzy paradigm is well suited to address conflicting tasks such as detail sharpening and noise smoothing. In addition the multiple-output structure increases the overall performance of the fuzzy processing because the operation can repeatedly be applied to the image data.

The proposed technique significantly improves performance in the enhancement of images corrupted by Gaussian noise without using complicated tuning of fuzzy set parameters. The nonlinear behavior of the enhancement system is very easily controlled by one parameter only.

5.2 Fuzzy network with Filter Design

Fuzzy networks have been shown to be very effective for image processing applications such as noise removal. The basic operation of a fuzzy network is explained as follows:.

5.2.1 Smoothing Network

Let us consider a digitized image having L gray levels. Let $X_{i,j}$ be the pixel luminance at location [i,j] , ($0 <= X_{i,j} <= L - 1$) , as represented in Fig.5.1.

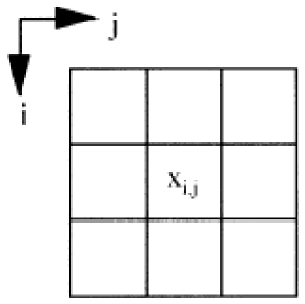


Figure 5.1: Showing 3×3 window with pixel to be processed

The structure of a simple fuzzy network operating on a 3X3 window is represented in Fig. 5.2. The network is composed of two symmetrical sub-network. The network output is an estimate of the noise amplitude.

The noise amplitude estimate is computed by considering (fuzzy) relations between the central pixel and its neighbours. For the sake of simplicity, let A denote the set of

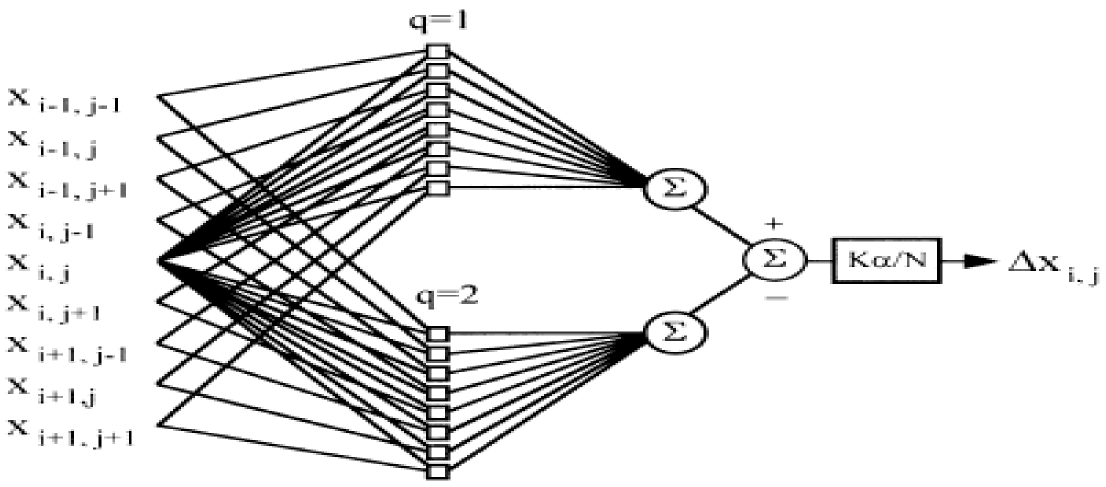


Fig. 5.2 Basic structure of fuzzy network

neighbouring pixel. The output is yielded by the following relationship:

$$\Delta X_{i,j} = \left(\frac{K_1\ \alpha_{\text{smooth}}}{N}\right)\left[\sum_{X_{m,n}\ \varepsilon\ A} \mu_{R1}\left(X_{i,j}, X_{m,n}, \alpha_{\text{smooth}}\right) - \sum_{X_{m,n}\ \varepsilon\ A} \mu_{R2}\left(X_{i,j}, X_{m,n}, \alpha_{\text{smooth}}\right)\right]$$

(5.1)

Where $K_1=1$ & Rq (q = 1,2) represents the class of fuzzy relations described by the parameterized membership function.

$$\mu_{Rq}(u,v,\alpha) = \begin{cases} \text{MAX}\{1-(|u-v-\alpha|)/(2 \times \alpha)\ ,0\} & ;\ q=1 \\ \text{MAX}\{1-(|u-v+\alpha|)/(2 \times \alpha)\ ,0\} & ;\ q=2 \end{cases} \qquad (5.2)$$

It should be observed that, by varying the value of α_{smooth} $(0 < \alpha_{smooth} <= L-1)$, different nonlinear behaviours can be obtained. For example, let us focus on fuzzy relation R_1. When α_{smooth} is large, the fuzzy relation represents "u is much larger than v". Conversely, when α_{smooth} is small, the fuzzy relation becomes "u is close to v." The (possibly) noise-free value $Y_{i,j}$ of the pixel luminance at location [i,j] is then obtained by subtracting the noise estimate $\Delta X_{i,j}$ from the original pixel luminance $X_{i,j}$.

$$Y_{i,j} = X_{i,j} - \Delta X_{i,j} \qquad (5.3)$$

The processing defined by (1)–(3) performs data smoothing and is typically applied to a noisy input image. Large values of α_{smooth} increase the noise cancellation at the price of a possible increase of the detail blur. The optimal choice depends on the amount of noise corruption and is typically a tradeoff between noise removal and detail preservation.

5.2.2 Sharpening Network

Now, let us consider a noise-free image. A sharpening effect can easily be implemented by using the same fuzzy network operations defined by (1) and (2). In this case, we choose

$$\mathbf{K_2 = 1}$$

$$\boldsymbol{\alpha_{sharp} = L - 1}$$

and we add the corresponding network output $\Delta Z_{i,j}$ to the original pixel luminance $X_{i,j.}$

$$Y_{i,j} = X_{i,j} \oplus \Delta Z_{i,j} \qquad (5.4)$$

where $\Delta Z_{i,j}$ is given by

$$\Delta Z_{i,j} = \left(\frac{K_2 \alpha_{sharp}}{N}\right) \left[\sum_{X_{m,n} \varepsilon A} \mu_{R1}\left(X_{i,j}, X_{m,n}, \alpha_{sharp}\right) - \sum_{X_{m,n} \varepsilon A} \mu_{R2}\left(X_{i,j}, X_{m,n}, \alpha_{sharp}\right) \right]$$

Symbol $\oplus$ denotes the bounded sum:

$$a \oplus b = \min\{a + b, L - 1\}$$

In fact, we can think of the sharpening effect as the opposite of the smoothing action. Notice that the bounded sum is formally required in order to limit the output value as follows:

$$Y_{i,j} \quad < \; = \quad L - 1$$

The parameter settings are based on a heuristic approach.

5.2.3 Combining Smoothing & Sharpening network

If the input image is noisy, we can combine a sharpening and a smoothing network in the same processing system. The former aims at increasing the luminance difference between the central pixel and its neighbourhood, while the latter aims at reducing the noise increase.

$$Y_{i,j} = X_{i,j} \oplus (\Delta Z_{i,j} - \Delta X_{i,j}) \qquad (5.5)$$

An appropriate choice of in the smoothing network permits us to remove noise in the uniform regions of the image, where the effect is more annoying from the point of view of the human perception.

5.3 Combining of fuzzy networks

More fuzzy networks can be adopted in the same structure in order to increase the enhancement effect. The proposed multiple-output system includes three fuzzy networks and deals with a 4x 4 neighbourhood, as shown in Figs (5.3 – 5.5) . Each processing step involves two different operations dealing with an appropriate choice of pixel patterns. The first operation performs smoothing only. This operation evaluates the output $Y_{i,j}$ by processing the luminances in the pixel pattern A (Fig. 3). This processing is performed by fuzzy network 1 (Fig. 4). According to the mechanism described in the previous section, the output $Y_{i,j}$ is given by the following relationship:

$$Y_{i,j} = X_{i,j} - (\Delta X_{i,j})^{A} \qquad (5.6)$$

Where,

$$\left(\Delta X_{i,j}\right)^{A} = \left(\frac{K_1\, \alpha_{smooth}}{N}\right)\left[\sum_{X_{m,n}\,\varepsilon\,A} \mu_{R1}\left(X_{i,j}, X_{m,n}, \alpha_{smooth}\right) - \sum_{X_{m,n}\,\varepsilon\,A} \mu_{R2}\left(X_{i,j}, X_{m,n}, \alpha_{smooth}\right)\right] \qquad (5.7)$$

The processing is recursive, i.e., the new value $Y_{i,j}$ is immediately assigned to $X_{i,j}$ and reused for further processing.

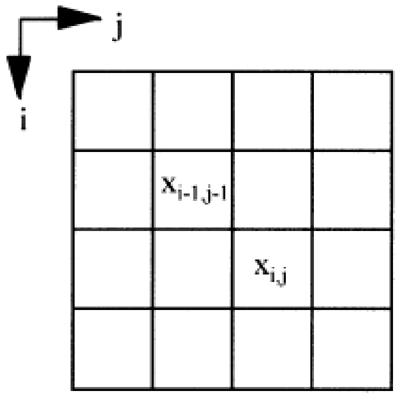


Figure 5.3: 4X 4 window

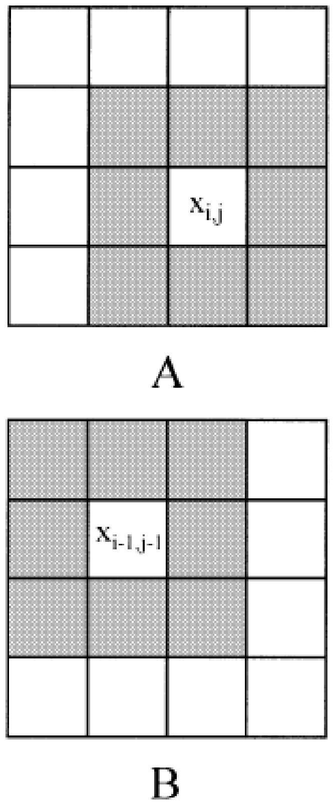


Fig. 5.4: Different pixel patterns for the multiple-output processing

The second operation is performed by fuzzy networks 2 & 3.It evaluates the output $Y_{i-1,j-1}$ by processing the luminances in the pixel pattern B (see Fig. 3). It should be observed that these luminances represent the results of the first operation because the processing is recursive and the window scans the image from left to right and from top to bottom. Since the second operation acts on prefiltered data, the effectiveness of the image enhancement process increases. The output is evaluated by the following relations:

$$Y_{i-1,j-1} = X_{i-1,j-1} \oplus \left((\Delta Z_{i-1,j-1})^{B} - (\Delta X_{i-1,j-1})^{B} \right) \qquad (5.8)$$

Where,

$$\left(\Delta X_{i-1,j-1}\right)^{B} = \left(\frac{K_1 \alpha_{smooth}}{N}\right)\left[\sum_{X_{m,n}\,\varepsilon\, B} \mu_{R1}\left(X_{i-1,j-1}, X_{m,n}, \alpha_{smooth}\right) - \sum_{X_{m,n}\,\varepsilon\, B} \mu_{R2}\left(X_{i-1,j-1}, X_{m,n}, \alpha_{smooth}\right)\right]$$

(5.9)

$$\left(\Delta Z_{i-1,j-1}\right)^{B} = \left(\frac{K_1 \alpha_{sharp}}{N}\right)\left[\sum_{X_{m,n}\,\varepsilon\, B} \mu_{R1}\left(X_{i-1,j-1}, X_{m,n}, \alpha_{sharp}\right) - \sum_{X_{m,n}\,\varepsilon\, B} \mu_{R2}\left(X_{i-1,j-1}, X_{m,n}, \alpha_{sharp}\right)\right]$$

(5.10)

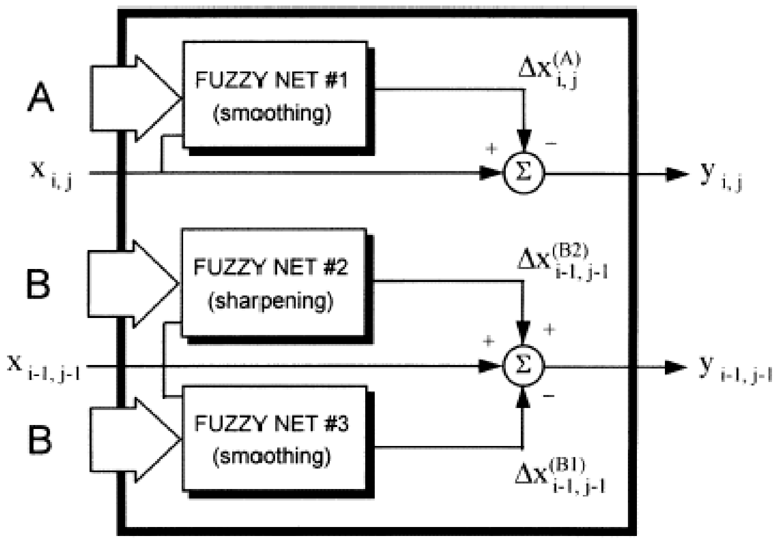


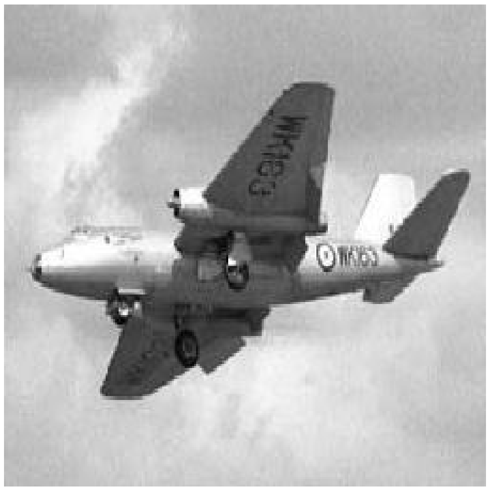
Figure 5.5 Block diagram of multiple output system

It is worth pointing out that the nonlinear behaviour of the overall system is easily controlled by the value of parameter α only ($0 < \alpha < = L-1$). In fact, according to (10), the strength of the sharpening action is assigned. On the contrary, the effectiveness of the smoothing effect depends on [see relations (8) and (9)]. A large value increases the noise removal. A small value decreases this effect.

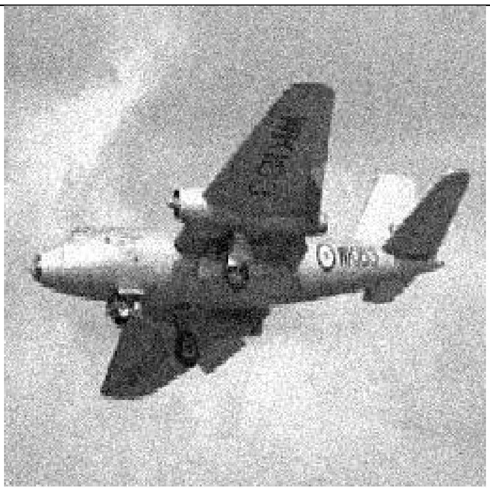
5.4 Experimental results and analysis

5.4.1 Result set 1

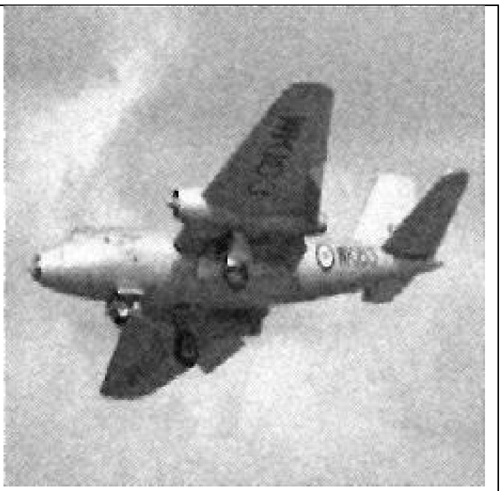
This filter was tested with different values of Gaussian noise for different images. The results are as follows:



(A)

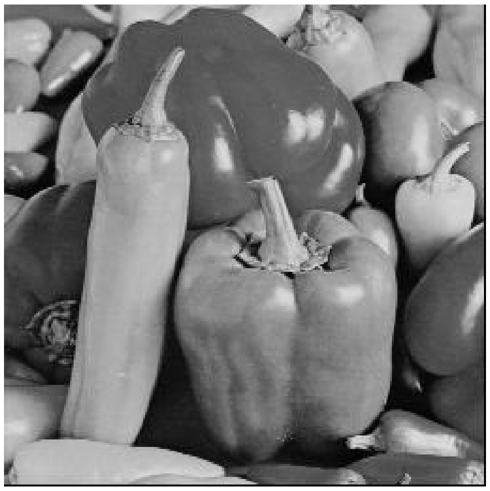


(B)

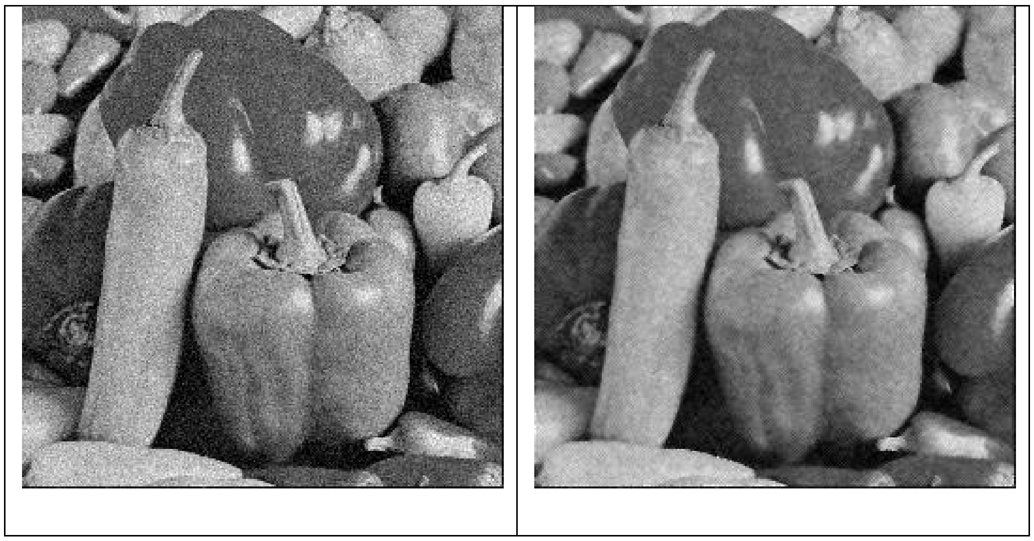


(C)

Fig 5.6: (a) Original Aircraft Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.004) (c) Image Corrected by the proposed method

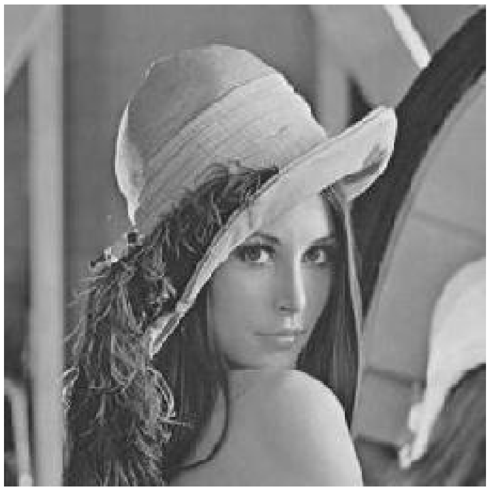


(A)



(B) (C)

Fig 5.6: (a) Original Peppers Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.004) (c) Image Corrected by the proposed method


(A)

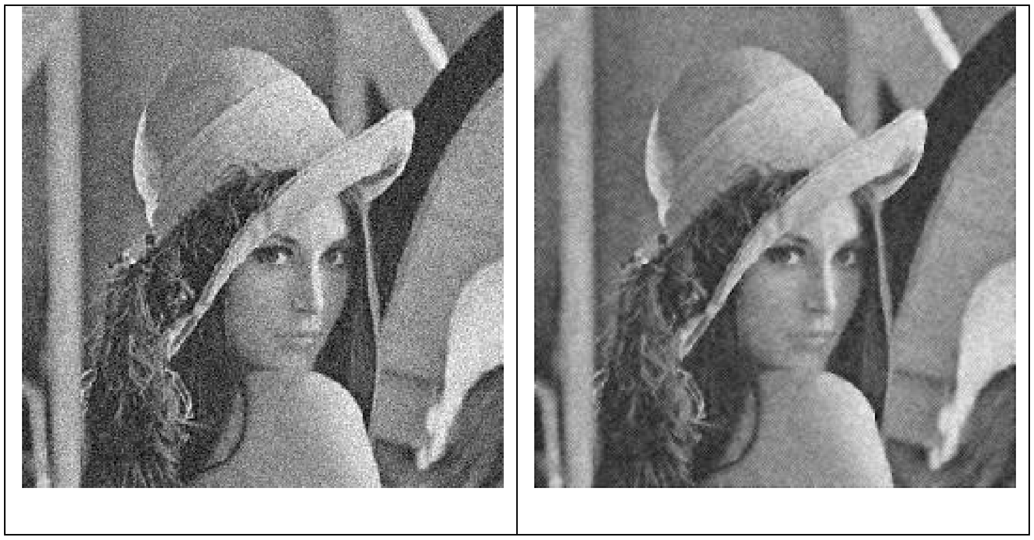
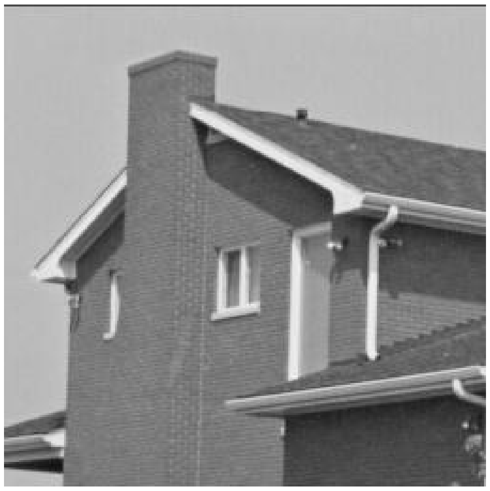

(B) (C)

Fig 5.7: (a) Original lena Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.004) (c) Image Corrected by the proposed method


(A)

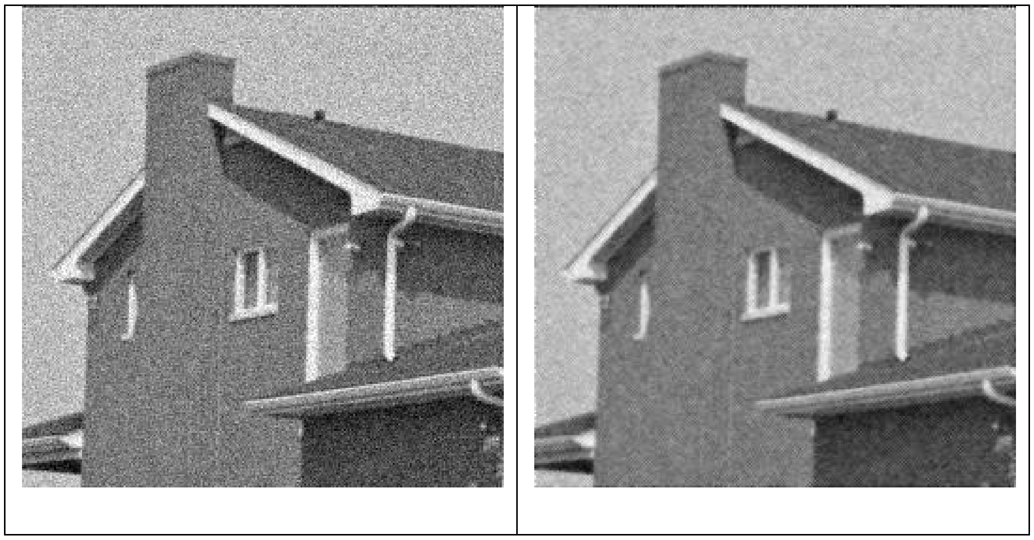
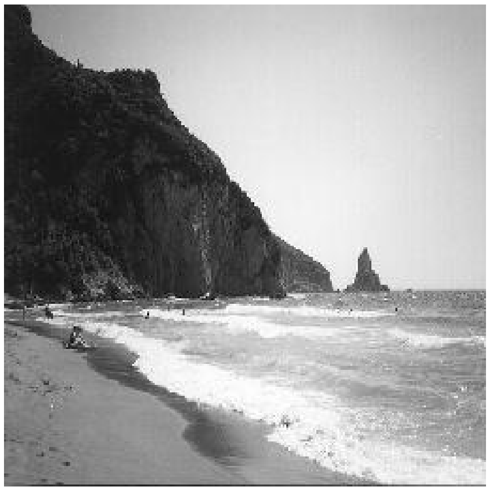

(B) (C)

Fig 5.8: (a) Original Red House Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.005) (c) Image Corrected by the proposed method


(A)

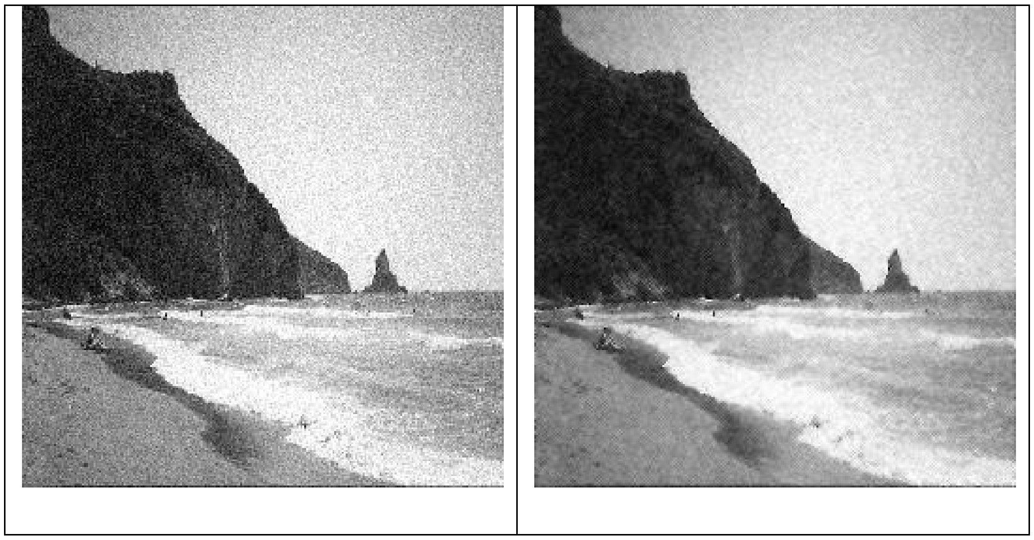
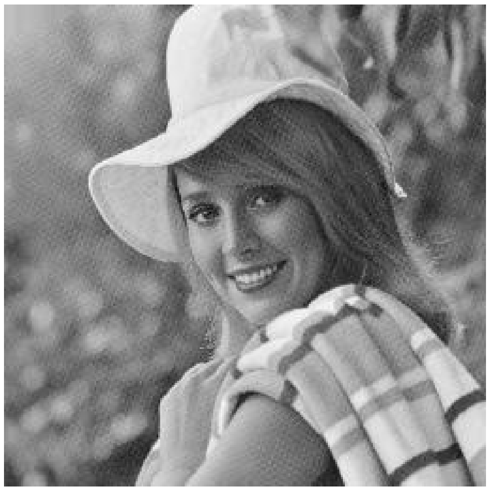

(B) (C)

Fig 5.9: (a) Original Gordios Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.005) (c) Image Corrected by the proposed method


(A)

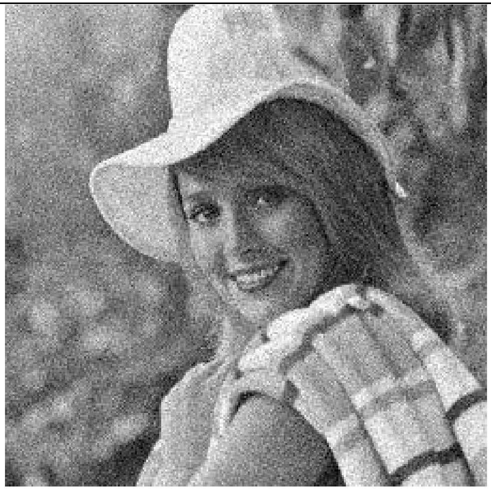
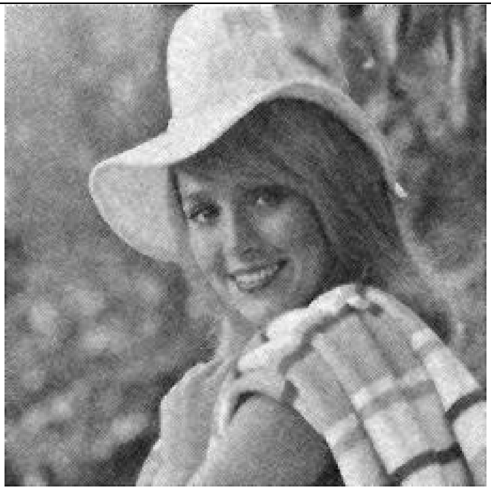
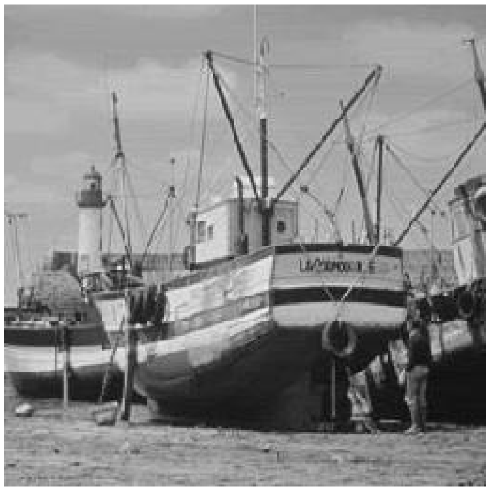


(B) (C)

Fig 5.10: (a) Original Elaine Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.005) (c) Image Corrected by the proposed method


(A)

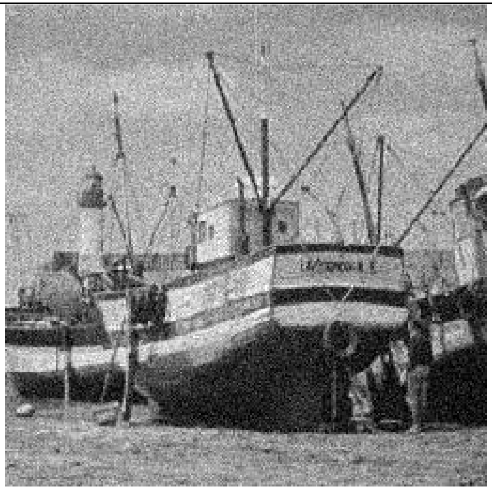
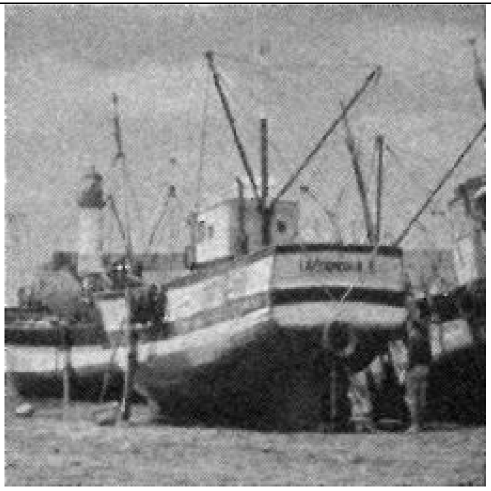
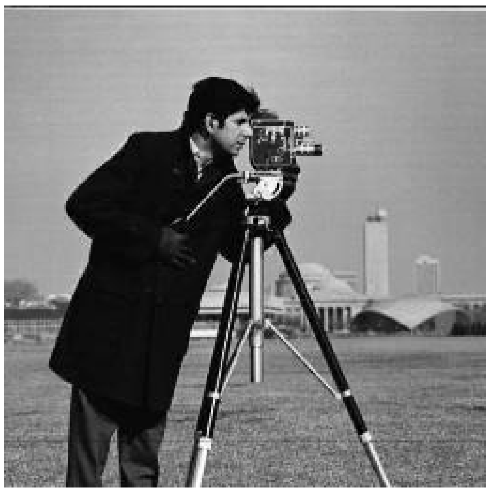
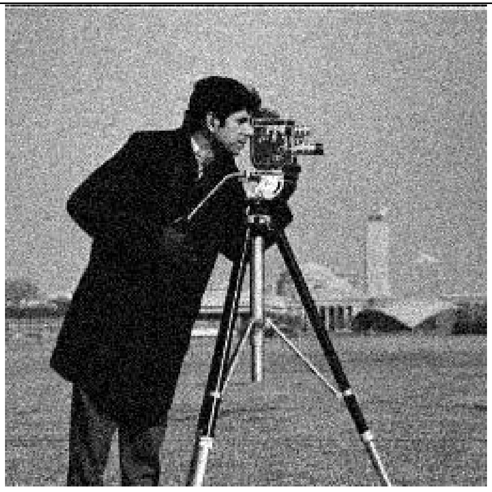
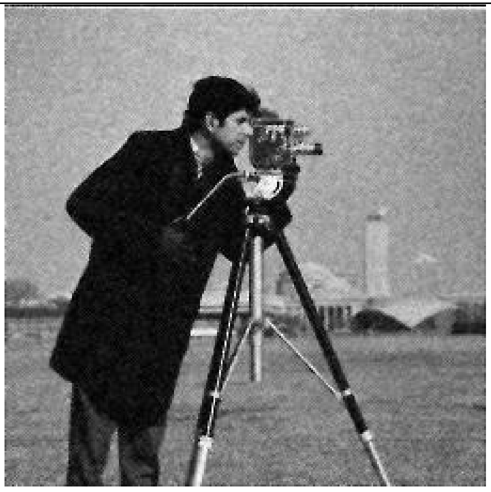


(B) (C)

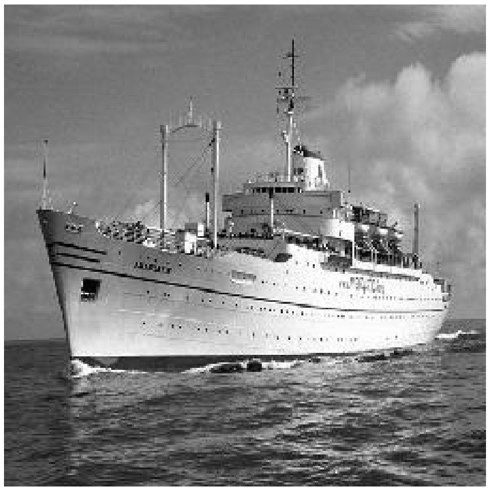
Fig 5.11: (a) Original boat Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.006) (c) Image Corrected by the proposed method


(A)

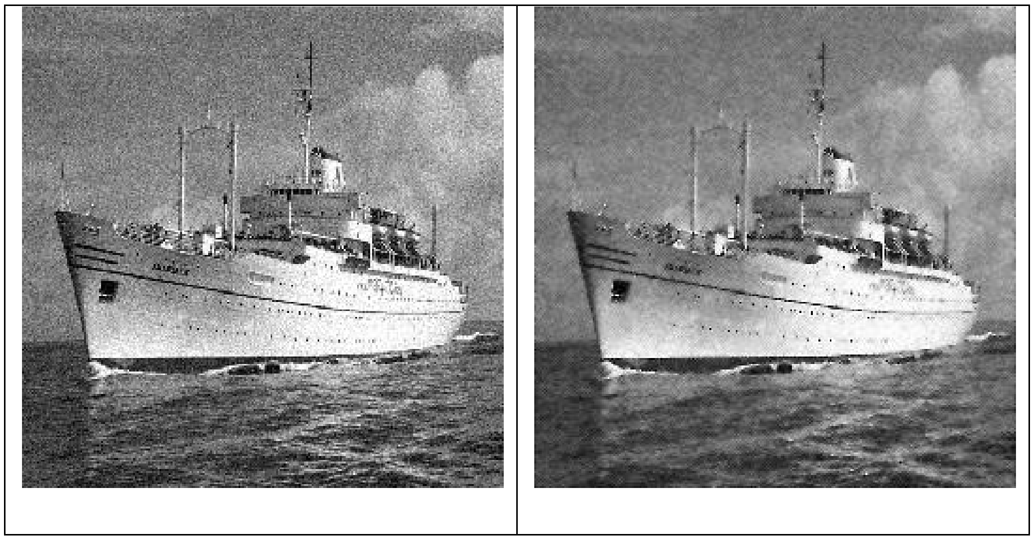



(B) (C)

Fig 5.12: (a) Original cameraman Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.006) (c) Image Corrected by the proposed method



(A)



(B) (C)

Fig 5.13: (a) Original ship Image (b) Image corrupted with Gaussian noise (mean = 0 and variance = 0.006) (c) Image Corrected by the proposed method

Table No. 5.1

Mean Squared Error (MSE) is used to compare the performance of the purposed filter with mean = 0; varying values of variance and keeping α_{smooth} = 90.

S.No.	Image	Size	Variance	MSE (Noisy Image)	MSE (Filtered Image)
1	Aircraft	256X256	0.004	251.0831	104.2658
2	Boat	256X256	0.004	259.3157	135.3408
3	Bridge	256X256	0.004	255.0955	194.9100
4	Cameraman	256X256	0.004	246.2198	138.1808
5	Elaine	256X256	0.004	256.5713	110.5588
6	Flintstones	256X256	0.004	245.4118	202.6412
7	Gordios	256X256	0.004	244.7941	114.9298
8	House	256X256	0.004	239.9612	120.9608
9	Lena	256X256	0.004	259.7132	113.9019
10	Peppers	256X256	0.004	255.9148	117.1757
11	Red House	256X256	0.004	256.8870	109.1357
12	Ship	256X256	0.004	254.4116	186.9299
13	Aircraft	256X256	0.005	312.4955	126.7148
14	Lena	256X256	0.005	327.5999	135.8023
15	Barbara	512X512	0.005	323.7253	174.9330
16	Peppers	512X512	0.005	315.5871	117.6468
17	Boat	256X256	0.006	384.4437	177.7596
18	Lena	256X256	0.006	387.2345	155.2740
19	Elaine	512X512	0.006	385.0884	137.2927
20	Peppers	512X512	0.006	380.0016	139.5670

5.4.2 Analysis from result set 1 & table 5.1

After carefully analysing from the figures and table (5.1), we can make out that the filtering performance of this filter is not same for all the images which are passed through this. We get best results for Aircraft and Red House images and worst results for Bridge and Flintstones images.

Table 5.2

Comparison of Mean Squared Error (MSE) values for different values of α_{smooth} ($0 < \alpha_{smooth} <= L-1$) ; with mean = 0; variance = 0.005.Test applied on famous Lena Image.

S.No.	α_{smooth}	MSE (Filtered Image)
1	5	805.4720
2	10	734.9650
3	15	612.8606
4	25	349.9498
5	35	221.3505
6	50	153.1185
7	60	140.6630
8	75	135.0264
9	80	135.6162
10	85	135.2196
11	90	135.8023
12	95	135.4634
13	100	137.3042
14	105	138.7020
15	120	139.8553
16	135	143.1158
17	150	143.4425
18	175	145.5496
19	200	146.2027
20	225	147.7474
21	240	146.6628
22	250	145.7695
23	255	147.0622

5.4.3 Analysis from table 5.2

The Mean Squared Error (MSE) value for the corrupted image with mean = 0 and variance = 0.005 Gaussian Noise applied on 256 X 256 Lena Image is 327.5999.

From the table 5.2, we can observe that for small values of α_{smooth} we don't get any filteration; in fact on passing the noisy image from the filter the noise is enhanced. As we increase the value of α_{smooth} the noise starts decreasing; it attains best value but on further increase in α_{smooth} the noise further starts increasing. We get best results for $\alpha_{smooth} = 90$.

The Mean Squared Error (MSE) value does not change significantly for α_{smooth} in the range of 75 to 255.

Table No. 5.3

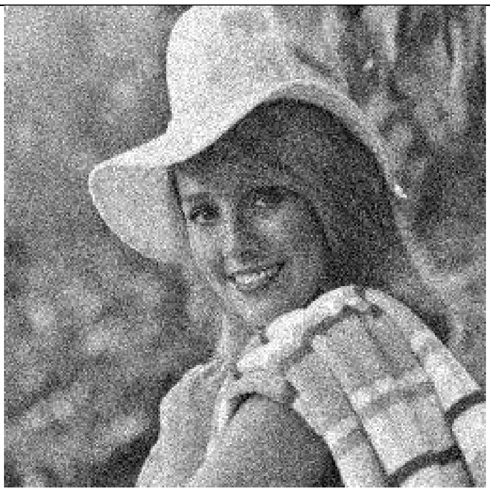
Comparison of Mean Squared Error (MSE) value for different values of variance; keeping mean = 0, and $\alpha_{smooth} = 90$ applied on 256 X 256 Lena Image.

S.No.	Variance	MSE (Noisy Image)	MSE (Filtered Image)
1	0.001	65.4982	48.6020
2	0.002	130.0615	70.3463
3	0.003	196.8356	92.1311
4	0.004	259.1617	112.9969
5	0.005	323.9015	135.5515
6	0.006	388.0177	157.1854
7	0.007	447.9888	176.8380
8	0.008	516.6158	202.0042
9	0.009	579.4248	223.3195
10	0.01	649.3800	248.3419
11	0.02	1253.2	495.5771
12	0.03	1837.9	814.50
13	0.04	2355.4	1181.4
14	0.05	2847.6	1597.5

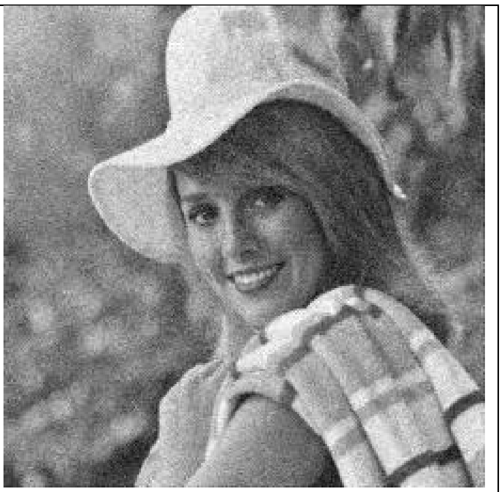
From above table, it is clear that as we increases the noise (variance), the performance of filter is affected. For larger values of variance the output image is highly noisy. In that case we should go for multiple pass filtering.

5.4.5 Result set 2 of multi pass filtering

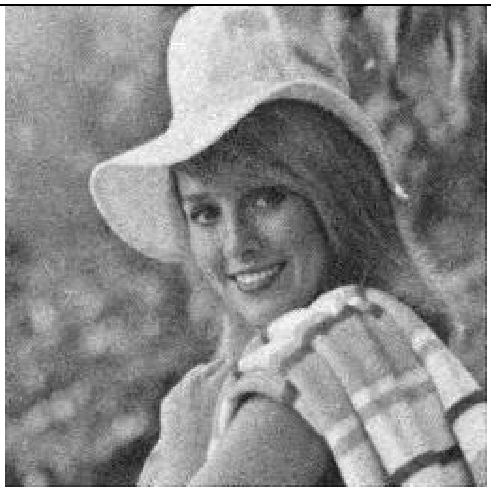
(A) Original Elaine Image

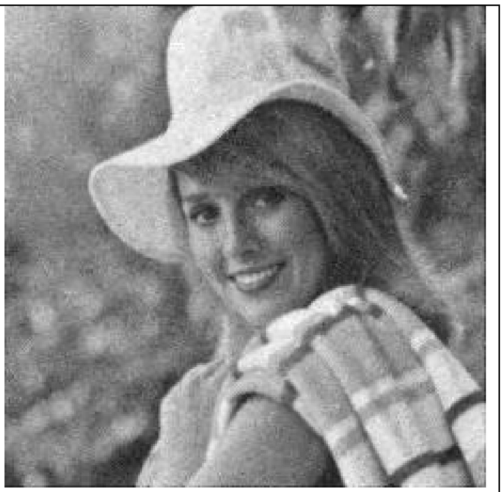


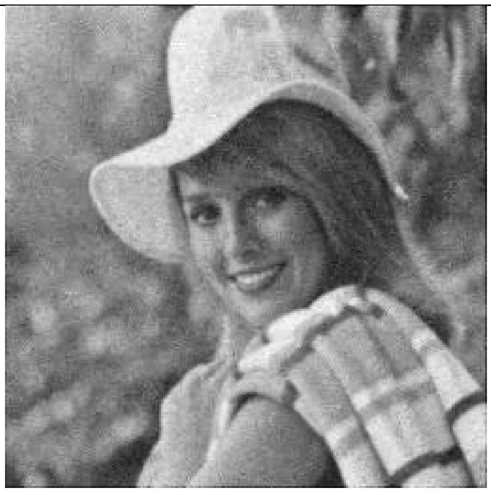
(B) Noisy Image



(C) First Iteration


(D) Second Iteration


(E) Third Iteration


(F) Fourth Iteration

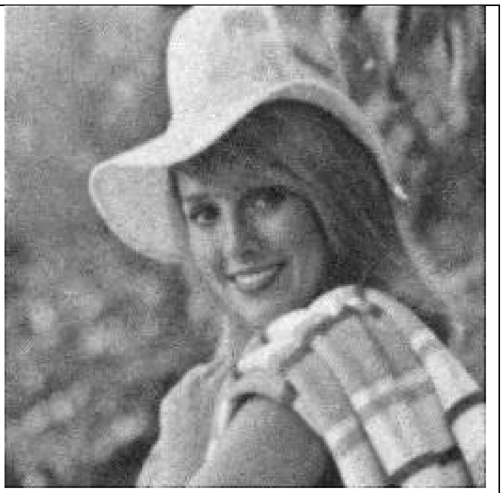

(G) Fifth Iteration

Fig 5.14 (a) Original Elaine Image (b) Noisy Image with Gaussian Noise (mean = 0 & variance = 0.006) (c) Image after first iteration (d) Image after second iteration
(e) Image after third iteration (f) Image after fourth iteration (g) Image after fifth iteration

Table 5.4

Comparison of Mean Squared Error (MSE) value after different passes through the filter keeping $\alpha_{smooth} = 90$ & with Gaussian noise of mean = 0, variance = 0.006 applied on 512 X 512 Elaine image.

Pass	MSE
1	137.2398
2	84.2753
3	86.4371
4	95.4737
5	106.1502

5.4.6 Analysis from above result set 2 & table 5.4

It is clear from above result set 2 and table 5.4 that multiple passing the image from the filter decreases the Mean Squared Error (MSE) . But, if we further keep on passing the image from the filter blurring starts taking place and Mean Squared Error (MSE) starts to increase . We get best results after second pass. Hence, there is no point passing the image beyond second pass.

6. Conclusion and suggestion for future work

6.1 Conclusion

In chapter four, we have implemented the four fuzzy filters to remove impulse and random noise. These are simple filters to implement and each of these filters applies a weighted membership function to an image with in a window to determine the centre pixel and no complex tuning of fuzzy set parameters is required. These includes Gaussian fuzzy filter with median centre(GMED) , the symmetrical triangular fuzzy filter with median centre (TMED), the Gaussian fuzzy filter with moving average centre (GMAV), and the symmetrical triangular fuzzy filter with moving average centre (TMAV).

In chapter five, we have implemented a fuzzy filter to remove random (e.g. Gaussian) noise. It is a simple filter and no complex tuning of fuzzy set parameters is required. In fact, the overall nonlinear behavior of the enhancement system is very easily controlled by one parameter ‘α’ only .

6.2 Future work

In chapter five, we have presented the filter for removal of random (e.g. Gaussian) noise. The filter is based on a parameter ‘α’ only which is fixed for each pixel for the entire image . This parameter ‘α’ which is fixed for each pixel for the entire image can be make variable for each pixel and calculated from image information and can be applied to remove impulse noise.

Bibliography

[1] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 2nd ed. Englewood Cliffs, New Jersey: Prentice-Hall 2001.

[2] K. Arakawa, "Median filters based on fuzzy rules and its application to image restoration", Fuzzy Sets Syst., vol. 77, pp. 3-13, 1996.

[3] F. Russo and G. Ramponi, "A fuzzy filter for images corrupted by impulse noise", IEEE Signal Process. Lett., vol. 3, no. 6, pp. 168-170, 1996.

[4] Choj, Y.S.. and Krishnapuram, R., “A robust approach to image enhancement based on fuzzy logic”, /€E€ Trcrns. Lnrtrge Procersing 6(6), 808-825. 1997.

[5] C. S. Lee, Y. H. Kuo, and P. T. Yu, “Weighted fuzzy mean filters for image processing," Fuzzy Sets and Systems, vol. 89, no. 2, pp.157-180, 1997.

[6] M. Nachtegael, D. Van der Weken, D. Van de Ville, and E. E. Kerre, Fuzzy Filters for Image Processing, 1st ed. Heidelberg, Germany: Physica Verlag, 2003, vol. 122.

[7] D. Van De Ville, M. Nachtegael, D. Van derWeken, E. E. Kerre, andW. Philips, “Noise reduction by fuzzy image filtering,” IEEE Trans. Fuzzy Syst., vol. 11, no. 4, pp. 429–436, Aug. 2001.

[8] M. Nachtegael, D. Van der Weken, and E. E. Kerre, “Fuzzy techniques in image processing: three case studies,” Int. J. Comput. Anticipatory Syst., vol. 12, pp. 89–104, Aug. 2002.

[9] H. K. Kwan, “Fuzzy filters for noise reduction in images,” in Fuzzy Filters for Image Processing, 1st ed, M. Nachtegael, D. Van der Weken, D. Van De Ville, and E. E. Kerre, Eds. Heidelberg, Germany: Physica Verlag, 2003, vol. 122, pp. 25–53.

[10] D. Van De Ville, M. Nachtegael, D. Van der Weken, W. Philips, I. Lemahieu, and E. E. Kerre, “A new fuzzy filter for Gaussian noise reduction,” Proc. SPIE Visual Communication and Image Processing, pp. 1–9, 2001.

[11] E. E. Kerre and M. Nachtegael, “Fuzzy Techniques in Image Processing,” 1st ed., vol. 52, Heidelberg, Physica Verlag, 2000, 429 pages.

[12] IEEE website: www.ieee.org

Program : Gaussian Noise reduction using fuzzy filter

```
clc
close all;
clear all;
I=imread('C:\Desktop\testimage\elaine.jpg');
figure,imshow(I);
J=imread('C:\filteredimage\test4.jpg');
J=imnoise(I,'gaussian',0,0.006);
figure,imshow(J);
 imwrite(I,'C:\inputimage\org.jpg');
 imwrite(J,'C:\\noisyimage\noisy.jpg');
in_image=size(I);
row=in_image(1);
col=in_image(2);
M=double(J);
K1=1;
N=8;
K2=1;
a1=90;
a2=255;
      for i= 1:row
    for j=1:col
        O(i,j)=0;
    end
end
 for i=1:row
    for j= 1:col
        if i==1 & j==1
            window1=double([0,0,0]);
            window2=double([0,M(i,j),M(i,j+1)]);
            window3=double([0,M(i+1,j),M(i+1,j+1)]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);

    elseif i==1 & j==col
            window1=double([0,0,0]);
            window2=double([M(i,j-1),M(i,j),0]);
            window3=double([M(i+1,j-1),M(i+1,j),0]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);

    elseif i==1 & j~=1 & j~=col
            window1=double([0,0,0]);
            window2=double([M(i,j-1),M(i,j),M(i,j+1)]);
            window3=double([M(i+1,j-1),M(i+1,j),M(i+1,j+1)]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);

        elseif i==row & j==1
            window1=double([0,M(i-1,j),M(i-1,j+1)]);
            window2=double([0,M(i,j),M(i,j+1)]);
            window3=double([0,0,0]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);

        elseif i==row & j==col
            window1=double([M(i-1,j-1),M(i-1,j),0]);
            window2=double([M(i,j-1),M(i,j),0]);
            window3=double([0,0,0]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);

        elseif i==row & j~=1 & j~=col
            window1=double([M(i-1,j-1),M(i-1,j),M(i-1,j+1)]);
            window2=double([M(i,j-1),M(i,j),M(i,j+1)]);
            window3=double([0,0,0]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);

        elseif j==1 & i~=row & i~=1
            window1=double([0,M(i-1,j),M(i-1,j+1)]);
            window2=double([0,M(i,j),M(i,j+1)]);
            window3=double([0,M(i+1,j),M(i+1,j+1)]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);

        elseif j==col & i~=row & i~=1
            window1=double([M(i-1,j-1),M(i-1,j),0]);
            window2=double([M(i,j-1),M(i,j),0]);
            window3=double([M(i+1,j-1),M(i+1,j),0]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);
        else
            window1=double([M(i-1,j-1),M(i-1,j),M(i-1,j+1)]);
            window2=double([M(i,j-1),M(i,j),M(i,j+1)]);
            window3=double([M(i+1,j-1),M(i+1,j),M(i+1,j+1)]);
            window=[window1;window2;window3];
            x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
            O(i,j)=M(i,j)-x(i,j);
         end

         M(i,j) = O(i,j);
```

```
      if (i > 1) & (j > 1)
        i=i-1;
        j=j-1;

        if i==1 & j==1
           window1=double([0,0,0]);
           window2=double([0,M(i,j),M(i,j+1)]);
           window3=double([0,M(i+1,j),M(i+1,j+1)]);
           window=[window1;window2;window3];
           x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
           y(i,j)=K2*a2/N*(windowprocess1(window,a2)-windowprocess2(window,a2));
           O(i,j)=min(M(i,j)+y(i,j)-x(i,j),255);


    elseif  i==1 & j~=1
           window1=double([0,0,0]);
           window2=double([M(i,j-1),M(i,j),M(i,j+1)]);
           window3=double([M(i+1,j-1),M(i+1,j),M(i+1,j+1)]);
           window=[window1;window2;window3];
           x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
           y(i,j)=K2*a2/N*(windowprocess1(window,a2)-windowprocess2(window,a2));
           O(i,j)=min(M(i,j)+y(i,j)-x(i,j),255);

       elseif j==1 & i~=1
           window1=double([0,M(i-1,j),M(i-1,j+1)]);
           window2=double([0,M(i,j),M(i,j+1)]);
           window3=double([0,M(i+1,j),M(i+1,j+1)]);
           window=[window1;window2;window3];
           x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
           y(i,j)=K2*a2/N*(windowprocess1(window,a2)-windowprocess2(window,a2));
           O(i,j)=min(M(i,j)+y(i,j)-x(i,j),255);

       else
           window1=double([M(i-1,j-1),M(i-1,j),M(i-1,j+1)]);
           window2=double([M(i,j-1),M(i,j),M(i,j+1)]);
           window3=double([M(i+1,j-1),M(i+1,j),M(i+1,j+1)]);
           window=[window1;window2;window3];
           x(i,j)=K1*a1/N*(windowprocess1(window,a1)-windowprocess2(window,a1));
           y(i,j)=K2*a2/N*(windowprocess1(window,a2)-windowprocess2(window,a2));
           O(i,j)=min(M(i,j)+y(i,j)-x(i,j),255);

       end
       M(i,j)=O(i,j);

      i=i+1;
     j=j+1;

  end

     end
end
```

```
S=uint8(O);
figure,imshow(S);
imwrite(S,'C:\\filteredimage\test5.jpg');
O=double(J);
Q=double(I);
C=0;
D=0;

   for a=1:row
    for b=1:col
        C=C + power(M(a,b)-Q(a,b),2);
        D=D + power(O(a,b)-Q(a,b),2);
    end
  end
mse1=(D/(row*col))
mse2=(C/(row*col))
```

Program: Window process1

```
function[num]=window(M,a);
sum=0;
u=M(2,2);
for i=1:3
    for j=1:3
        if i~=2 & j~=2
            v=M(i,j);
            sum =sum+max(1-abs(u-v-a)/(2*a),0);
        end
    end
end
num=sum;
```

Program: Window process2

```
function[num]=window(M,a);
sum=0;
u=M(2,2);
for i=1:3
    for j=1:3
        if i~=2 & j~=2
            v=M(i,j);
            sum =sum+max(1-abs(u-v+a)/(2*a),0);
        end
    end
end
num=sum;
```

Program: Moving average filter(MAV)

```
clc;
close all;
clear all;
n=imread('C:\testimages\lena.jpg');
n1=double(n);
a=im2double(n);
figure;imshow(a);
title('original image');
b=imnoise(a,'salt & pepper',0.3);
p=im2uint8(b);
pi=double(p);
imwrite(p,'C:\impimages\lena.jpg');
b1=imnoise(a,'gaussian',0,0.106);
b2=im2uint8(b1);
b3=double(b2);
imwrite(b2,'C:\gauimages\lena.jpg');

win=3;
w=(win-1)/2;
g=padarray(b,[w w]);
e=size(g);
r=e(1);
c=e(2);
i=w+1;
  while i<=(r-w)
    j=w+1;
     while j<=c-w
         fi=g(i-w:i+w,j-w:j+w);
         mav=sum(sum(fi))/(win*win);
       x=1;
     while x<=win
       y=1;
       while y <= win
           fn=mav;
           y=y+1;
       end
       x=x+1;
     end
     g(i,j)=fn;
     j=j+1;
     end
     i=i+1;
  end
  f=g(1+w:r-w,1+w:c-w);
  f1=im2uint8(f);
  imwrite(f1,'C:\filterimp\lena.jpg');
  f2=double(f1);

  win=3;
  w=(win-1)/2;
```

```
g=padarray(b1,[w w]);
e=size(g);
r=e(1);
c=e(2);
i=w+1;
  while i<=(r-w)
    j=w+1;
     while j<=c-w
         fi=g(i-w:i+w,j-w:j+w);
         mav=sum(sum(fi))/(win*win);
       x=1;
     while x<=win
       y=1;
       while y<=win
           fn=mav;
           y=y+1;
       end
       x=x+1;
     end
     g1(i,j)=fn;
     j=j+1;
     end
     i=i+1;
  end
  f3=g1(1+w:r-w,1+w:c-w);
  f4=im2uint8(f3);
  imwrite(f4,'C:\filtergau\lena.jpg');
  f5=double(f4);

row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
      r1(i,j)=((n1(i,j)-pi(i,j))^2);
  end
 end

k1=sum(sum(r1));
mse1=k1/(size(n1,1)*size(n1,2))

row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
      r2(i,j)=((n1(i,j)-b3(i,j))^2);
  end
 end

k2=sum(sum(r2));
mse2=k2/(size(n1,1)*size(n1,2))

row1=size(n1,1);
col1=size(n1,2);
```

```
for i=1:row1
   for j=1:col1
      r3(i,j)=((n1(i,j)-f2(i,j))^2);
  end
 end

k3=sum(sum(r3));
mse3=k3/(size(n1,1)*size(n1,2))

row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
      r4(i,j)=((n1(i,j)-f5(i,j))^2);
  end
 end

k4=sum(sum(r4));
mse4=k4/(size(n1,1)*size(n1,2))
```

Program: median filter.

```
clc;
close all;
clear all;
n=imread('C:\testimages\redhouse.jpg');
n1=double(n);
a=im2double(n);
figure;imshow(a);
title('original image');
b=imnoise(a,'salt & pepper',0.03);
p=im2uint8(b);
imwrite(b,'C:\impimages\redhouse.jpg');
pi=double(p);
b1=imnoise(a,'gaussian',0,0.0052);
b2=im2uint8(b1);
imwrite(b1,'C:\gauimages\redhouse.jpg');
b3=double(b2);
k1=3;
m1=medfilt2(b,[k1,k1]);
imwrite(m1,'C:\filterimp\redhouse.jpg');
p1=im2uint8(m1);
p2=double(p1);
m2=medfilt2(b1,[k1,k1]);
imwrite(m2,'C:\filtergau\redhouse.jpg');
p3=im2uint8(m2);
p4=double(p3);

row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
```

```
      r1(i,j)=((n1(i,j)-pi(i,j))^2);
  end
 end

k2=sum(sum(r1));
mse1_impulse=k2/(size(n1,1)*size(n1,2))

for i=1:row1
   for j=1:col1
      r2(i,j)=((n1(i,j)-b3(i,j))^2);
  end
 end

k3=sum(sum(r2));
mse2_gaussian=k3/(size(n1,1)*size(n1,2))

for i=1:row1
   for j=1:col1
      r3(i,j)=((n1(i,j)-p2(i,j))^2);
  end
 end

k4=sum(sum(r3));
mse3_filt_imp=k4/(size(n1,1)*size(n1,2))

for i=1:row1
   for j=1:col1
      r4(i,j)=((n1(i,j)-p4(i,j))^2);
  end
 end

k5=sum(sum(r4));
mse4_filt_gau=k5/(size(n1,1)*size(n1,2))
```

Program: Gaussian fuzzy filter with median centre

```
clc;
close all;
clear all;
n=imread('C:\testimages\redhouse.jpg');
n1=double(n);
a=im2double(n);
figure;imshow(a);
title('original image');
b=imnoise(a,'salt & pepper',0.3);
p1=im2uint8(b);
p2=double(p1);
imwrite(p1,'C:\impimages\redhouse.jpg');
b1=imnoise(a,'gaussian',0,0.106);
b2=im2uint8(b1);
b3=double(b2);
imwrite(b2,'C:\gauimages\redhouse.jpg');
row1=size(n1,1);
```

```
col1=size(n1,2);

for i=1:row1
   for j=1:col1
      r1(i,j)=((n1(i,j)-p2(i,j))^2);
  end
 end

k1=sum(sum(r1));
mse1=k1/(size(n1,1)*size(n1,2))

for i=1:row1
   for j=1:col1
      r2(i,j)=((n1(i,j)-b3(i,j))^2);
  end
 end

k2=sum(sum(r2));
mse2=k2/(size(n1,1)*size(n1,2))

g=padarray(b,[1 1]);
e=size(g);
r=e(1);
c=e(2);
i=1;
t=exp(ones(1,1));
i=2;
while i<=(r-1)
    j=2;
    while j<=c-1
        k=[g(i-1,j-1);g(i-1,j);g(i-1,j+1);g(i,j-1);g(i,j);g(i,j+1);g(i+1,j-1);g(i+1,j);g(i+1,j+1)];
        x=median(k);
        y=sqrt(std2(k));
        if (y~=0)
         l=i-1;
         while l<=(i+1)
             m=j-1;
             while m<=(j+1)
             z=-(((g(l,m)-x)/y).^2)/2;
             o(l,m)=realpow(t,z);
             m=m+1;
             end
             l=l+1;
         end
        end

k=[g(i-1,j-1)*o(i-1,j-1);g(i-1,j)*o(i-1,j);g(i-1,j+1)*o(i-1,j+1);g(i,j-1)*o(i,j-
1);g(i,j)*o(i,j);g(i,j+1)*o(i,j+1);g(i+1,j-1)*o(i+1,j-1);g(i+1,j)*o(i+1,j);g(i+1,j+1)*o(i+1,j+1)];
sum1=sum(k);
l=[o(i-1,j-1);o(i-1,j);o(i-1,j+1);o(i,j-1);o(i,j);o(i,j+1);o(i+1,j-1);o(i+1,j);o(i+1,j+1)];
sum2=sum(l);

if (sum2~=0)
```

```
  g(i,j)=sum1/sum2;
end
j=j+1;
    end
i=i+1;
end

f=g(2:r-1,2:c-1);
f1=im2uint8(f);
imwrite(f1,'C:\filterimp\redhouse.jpg');
f2=double(f1);
for i=1:row1
   for j=1:col1
      r3(i,j)=((n1(i,j)-f2(i,j))^2);
  end
 end
k3=sum(sum(r3));
mse3=k3/(size(n1,1)*size(n1,2))

g=padarray(b1,[1 1]);
e=size(g);
r=e(1);
c=e(2);
i=1;
t=exp(ones(1,1));
i=2;
while i<=(r-1)
    j=2;
    while j<=c-1
        k=[g(i-1,j-1);g(i-1,j);g(i-1,j+1);g(i,j-1);g(i,j);g(i,j+1);g(i+1,j-1);g(i+1,j);g(i+1,j+1)];
        x=median(k);
        y=sqrt(std2(k));
        if (y~=0)
         l=i-1;
         while l<=(i+1)
             m=j-1;
             while m<=(j+1)
             z=-(((g(l,m)-x)/y).^2)/2;
             o(l,m)=realpow(t,z);
             m=m+1;
             end
             l=l+1;
         end
        end

k=[g(i-1,j-1)*o(i-1,j-1);g(i-1,j)*o(i-1,j);g(i-1,j+1)*o(i-1,j+1);g(i,j-1)*o(i,j-
1);g(i,j)*o(i,j);g(i,j+1)*o(i,j+1);g(i+1,j-1)*o(i+1,j-1);g(i+1,j)*o(i+1,j);g(i+1,j+1)*o(i+1,j+1)];
sum1=sum(k);
l=[o(i-1,j-1);o(i-1,j);o(i-1,j+1);o(i,j-1);o(i,j);o(i,j+1);o(i+1,j-1);o(i+1,j);o(i+1,j+1)];
sum2=sum(l);

if (sum2~=0)
  g(i,j)=sum1/sum2;
```

```
end
j=j+1;
    end
i=i+1;
end

f3=g(2:r-1,2:c-1);
f4=im2uint8(f3);
imwrite(f4,'C:\filtergau\redhouse.jpg');
f5=double(f4);
for i=1:row1
   for j=1:col1
      r4(i,j)=((n1(i,j)-f5(i,j))^2);
  end
 end
k4=sum(sum(r4));
mse4=k4/(size(n1,1)*size(n1,2))
% psnr4=10*log10(255^2/mse3)
```

Program: Gaussian fuzzy filter with moving average centre

```
clc;
close all;
clear all;
n=imread('C:\testimages\redhouse.jpg');
n1=double(n);
a=im2double(n);
figure;imshow(a);
title('original image');
b=imnoise(a,'salt & pepper',0.03);
p1=im2uint8(b);
p2=double(p1);
imwrite(p1,'C:\Documents and Settings\prakash\Desktop\impimages\redhouse.jpg');
b1=imnoise(a,'gaussian',0,0.0052);
b2=im2uint8(b1);
b3=double(b2);
imwrite(b2,'C:\Documents and Settings\prakash\Desktop\gauimages\redhouse.jpg');
row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
      r1(i,j)=((n1(i,j)-p2(i,j))^2);
  end
 end

k1=sum(sum(r1));
mse1=k1/(size(n1,1)*size(n1,2))

for i=1:row1
   for j=1:col1
      r2(i,j)=((n1(i,j)-b3(i,j))^2);
  end
 end
```

```
k2=sum(sum(r2));
mse2=k2/(size(n1,1)*size(n1,2))

g=padarray(b,[1 1]);
% g=padarray(g,[1,1]);
e=size(g);
r=e(1);
c=e(2);
i=1;
t=exp(ones(1,1));
i=2;
while i<=(r-1)
    j=2;
    while j<=c-1
        k=[g(i-1,j-1);g(i-1,j);g(i-1,j+1);g(i,j-1);g(i,j);g(i,j+1);g(i+1,j-1);g(i+1,j);g(i+1,j+1)];
        x=mean2(k);
        y=sqrt(std2(k));
        if (y~=0)
          l=i-1;
         while l<=(i+1)
             m=j-1;
             while m<=(j+1)
             z=-(((g(l,m)-x)/y).^2)/2;
             o(l,m)=realpow(t,z);
             m=m+1;
             end
             l=l+1;
         end
        end

k=[g(i-1,j-1)*o(i-1,j-1);g(i-1,j)*o(i-1,j);g(i-1,j+1)*o(i-1,j+1);g(i,j-1)*o(i,j-
1);g(i,j)*o(i,j);g(i,j+1)*o(i,j+1);g(i+1,j-1)*o(i+1,j-1);g(i+1,j)*o(i+1,j);g(i+1,j+1)*o(i+1,j+1)];
sum1=sum(k);
l=[o(i-1,j-1);o(i-1,j);o(i-1,j+1);o(i,j-1);o(i,j);o(i,j+1);o(i+1,j-1);o(i+1,j);o(i+1,j+1)];
sum2=sum(l);

if (sum2~=0)
  g(i,j)=sum1/sum2;
end
j=j+1;
    end
i=i+1;
end

f=g(2:r-1,2:c-1);
f1=im2uint8(f);
imwrite(f1,'C:\Documents and Settings\prakash\Desktop\filterimp\redhouse.jpg');
f2=double(f1);
for i=1:row1
   for j=1:col1
      r3(i,j)=((n1(i,j)-f2(i,j))^2);
  end
 end
```

```
k3=sum(sum(r3));
mse3=k3/(size(n1,1)*size(n1,2))

g=padarray(b1,[1 1]);
e=size(g);
r=e(1);
c=e(2);
i=1;
t=exp(ones(1,1));
i=2;
while i<=(r-1)
    j=2;
    while j<=c-1
        k=[g(i-1,j-1);g(i-1,j);g(i-1,j+1);g(i,j-1);g(i,j);g(i,j+1);g(i+1,j-1);g(i+1,j);g(i+1,j+1)];
        x=mean2(k);
        y=sqrt(std2(k));
        if (y~=0)
          l=i-1;
         while l<=(i+1)
             m=j-1;
             while m<=(j+1)
             z=-(((g(l,m)-x)/y).^2)/2;
             o(l,m)=realpow(t,z);
             m=m+1;
             end
             l=l+1;
         end
        end

k=[g(i-1,j-1)*o(i-1,j-1);g(i-1,j)*o(i-1,j);g(i-1,j+1)*o(i-1,j+1);g(i,j-1)*o(i,j-
1);g(i,j)*o(i,j);g(i,j+1)*o(i,j+1);g(i+1,j-1)*o(i+1,j-1);g(i+1,j)*o(i+1,j);g(i+1,j+1)*o(i+1,j+1)];
sum1=sum(k);
l=[o(i-1,j-1);o(i-1,j);o(i-1,j+1);o(i,j-1);o(i,j);o(i,j+1);o(i+1,j-1);o(i+1,j);o(i+1,j+1)];
sum2=sum(l);

if (sum2~=0)
  g(i,j)=sum1/sum2;
end
j=j+1;
    end
i=i+1;
end

f3=g(2:r-1,2:c-1);
f4=im2uint8(f3);
imwrite(f4,'C:\ filtergau\redhouse.jpg');
f5=double(f4);
for i=1:row1
   for j=1:col1
      r4(i,j)=((n1(i,j)-f5(i,j))^2);
  end
 end
k4=sum(sum(r4));
```

```
mse4=k4/(size(n1,1)*size(n1,2))
```

Program: Triangle fuzzy filter with median centre

```
clc;
close all;
clear all;
n=imread('C:\testimages\redhouse.jpg');
n1=double(n);
a=im2double(n);
figure;imshow(a);
title('original image');
b=imnoise(a,'salt & pepper',0.3);
p1=im2uint8(b);
p2=double(p1);
imwrite(p1,'C:\impimages\redhouse.jpg');
b1=imnoise(a,'gaussian',0,0.106);
b2=im2uint8(b1);
b3=double(b2);
imwrite(b2,'C:\gauimages\redhouse.jpg');
w=3;
z=(w-1)/2;
g=padarray(b,[z z]);
e=size(g);
r=e(1);
c=e(2);
row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
      r1(i,j)=((n1(i,j)-p2(i,j))^2);
  end
 end

k1=sum(sum(r1));
mse1=k1/(size(n1,1)*size(n1,2))

for i=1:row1
   for j=1:col1
      r2(i,j)=((n1(i,j)-b3(i,j))^2);
  end
 end

k2=sum(sum(r2));
mse2=k2/(size(n1,1)*size(n1,2))

i=z+1;
while i<=(r-z)
  j=z+1;
  while j<=c-z
    k=g(i-z:i+z,j-z:j+z);
    m=median(k);
    xmax = max(k);
    xmin = min(k);
    n=[xmax-m;m-xmin];
    xmv=max(n);
    v1=0;
    v2=0;
    p=-z;
    while p<=z
     q=-z;
     while q<=z
       x=abs(g(i+p,j+q)-m);
       if(x<=xmv & xmv~=0)
         f=1-x/xmv;
       else
         f=1;
       end
       v1=v1+f*g(i+p,j+q);
       v2=v2+f;
       q=q+1;
     end
     p=p+1;
    end
    g(i,j)=v1/v2;
    j=j+1;
  end
  i=i+1;
end
f=g(z+1:r-z,z+1:c-z);
f1=im2uint8(f);
imwrite(f1,'C:\filterimp\redhouse.jpg');
f2=double(f1);
for i=1:row1
   for j=1:col1
      r3(i,j)=((n1(i,j)-f2(i,j))^2);
  end
 end

k3=sum(sum(r3));
mse3=k3/(size(n1,1)*size(n1,2))
w=7;
z=(w-1)/2;
g=padarray(b1,[z z]);
e=size(g);
r=e(1);
c=e(2);
i=z+1;
while i<=(r-z)
  j=z+1;
  while j<=c-z
    k=g(i-z:i+z,j-z:j+z);
    m=median(k);
    xmax = max(k);
    xmin = min(k);
    n=[xmax-m;m-xmin];
    xmv=max(n);
    v1=0;
    v2=0;
    p=-z;
    while p<=z
     q=-z;
     while q<=z
       x=abs(g(i+p,j+q)-m);
       if(x<=xmv & xmv~=0)
         f=1-x/xmv;
       else
         f=1;
       end
       v1=v1+f*g(i+p,j+q);
       v2=v2+f;
       q=q+1;
     end
     p=p+1;
    end
    g(i,j)=v1/v2;
    j=j+1;
  end
  i=i+1;
end
f3=g(z+1:r-z,z+1:c-z);
f4=im2uint8(f3);
imwrite(f4,'C:\filtergau\redhouse.jpg');
f5=double(f4);
for i=1:row1
   for j=1:col1
      r4(i,j)=((n1(i,j)-f5(i,j))^2);
  end
 end

k4=sum(sum(r4));
mse4=k4/(size(n1,1)*size(n1,2))
```

Program: Symmetrical triangle fuzzy filter with moving average centre

```
clc;
close all;
clear all;
n=imread('C:\testimages\redhouse.jpg');
n1=double(n);
a=im2double(n);
figure;imshow(a);
title('original image');
b=imnoise(a,'salt & pepper',0.3);
p1=im2uint8(b);
p2=double(p1);
imwrite(p1,'C:\ impimages\redhouse.jpg');
b1=imnoise(a,'gaussian',0,0.106);
b2=im2uint8(b1);
b3=double(b2);
```

```
imwrite(b2,'C:\gauimages\redhouse.jpg');
row1=size(n1,1);
col1=size(n1,2);

for i=1:row1
   for j=1:col1
      r1(i,j)=((n1(i,j)-p2(i,j))^2);
  end
 end

k1=sum(sum(r1));
mse1=k1/(size(n1,1)*size(n1,2))
for i=1:row1
   for j=1:col1
      r2(i,j)=((n1(i,j)-b3(i,j))^2);
  end
 end

k2=sum(sum(r2));
mse2=k2/(size(n1,1)*size(n1,2))
win=3;
w=(win-1)/2;
g=padarray(b,[w w]);
e=size(g);
r=e(1);
c=e(2);
i=w+1;
  while i<=(r-w)
    j=w+1;
     while j<=c-w
         fi=g(i-w:i+w,j-w:j+w);
         mav=sum(sum(fi))/(win*win);
         maxx=max(max(fi));
         minn=min(min(fi));
         if((maxx-mav)>(mav-minn))
            mv=double(maxx-mav);
         else
            mv=double(mav-minn);
         end
         numer=0;
         denom=0;
         x=1;
       while x<=win
       y=1;
       while y <= win
           xy=double(fi(x,y));
           if(mv==0)
              fn=1;
           else
            if(xy>mav)
              fn=1-(xy-mav)/mv;
            else
              fn=1-(mav-xy)/mv;
            end
           end
              numer=numer+fn*xy;
              denom=denom+fn;

           y=y+1;
       end
       x=x+1;
     end
     g(i,j)=numer/denom;
     j=j+1;
     end
     i=i+1;
  end

  f=g(1+w:r-w,1+w:c-w);
  f1=im2uint8(f);
  imwrite(f1,'C:\filterimp\redhouse.jpg');
  f2=double(f1);
row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
      r3(i,j)=((n1(i,j)-f2(i,j))^2);
  end
end
 k3=sum(sum(r3));
mse3=k3/(size(n1,1)*size(n1,2))
win=3;
w=(win-1)/2;
g=padarray(b1,[w w]);
e=size(g);
r=e(1);
c=e(2);
i=w+1;
  while i<=(r-w)
    j=w+1;
     while j<=c-w
         fi=g(i-w:i+w,j-w:j+w);
         mav=sum(sum(fi))/(win*win);
         maxx=max(max(fi));
         minn=min(min(fi));
         if((maxx-mav)>(mav-minn))
            mv=double(maxx-mav);
         else
            mv=double(mav-minn);
         end
         numer=0;
         denom=0;
         x=1;
       while x<=win
       y=1;
       while y <= win
```

```
          xy=double(fi(x,y));
          if(mv==0)
             fn=1;
          else
           if(xy>mav)
             fn=1-(xy-mav)/mv;
           else
             fn=1-(mav-xy)/mv;
           end
          end
             numer=numer+fn*xy;
             denom=denom+fn;

          y=y+1;
      end
      x=x+1;
    end
    g(i,j)=numer/denom;
    j=j+1;
    end
    i=i+1;
  end

  f3=g(1+w:r-w,1+w:c-w);
  f4=im2uint8(f3);
  imwrite(f4,'C:\filtergau\redhouse.jpg');
  f5=double(f4);
row1=size(n1,1);
col1=size(n1,2);
for i=1:row1
   for j=1:col1
      r4(i,j)=((n1(i,j)-f4(i,j))^2);
  end
end
 k4=sum(sum(r4));
mse4=k4/(size(n1,1)*size(n1,2))
```