

2003

Denoising techniques - a comparison

Sarita Veera Dangeti

Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_theses



Part of the [Electrical and Computer Engineering Commons](#)

Recommended Citation

Dangeti, Sarita Veera, "Denoising techniques - a comparison" (2003). *LSU Master's Theses*. 3789.
https://digitalcommons.lsu.edu/gradschool_theses/3789

This Thesis is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Master's Theses by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.

DENOISING TECHNIQUES - A COMPARISON

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

in

The Department of Electrical and Computer Engineering

by

Sarita Dangeti

B.E., Andhra University College of Engineering, Visakhapatnam, India, 2000
May 2003

Acknowledgements

I would like to thank Dr. Suresh Rai, my major professor for his assistance and constant support during my Master' program. He came up with the very initial idea of this thesis. I greatly appreciate his invaluable guidance, which helped me complete this thesis.

I thank my committee members Dr. Kak and Dr. Trahan for their support during my work on this thesis. Special thanks to Rhett Smith for all software provided especially, Fraclab. My sincere gratitude to the faculty and staff of the Department of Electrical and Computer Engineering.

I thank my parents and my sister, Betina for their support and guidance, which gave me an opportunity to pursue higher studies. Special thanks to my husband Krishna who was very patient and provided me the encouragement needed to complete my thesis. I also thank Padma Maddi for helping me during my documentation of this thesis.

Last but not the least I would like to thank all my friends whose support was very valuable during my Master's study at LSU.

Table of Contents

ACKNOWLEDGEMENTS.....	ii
ABSTRACT.....	v
CHAPTER	
1 INTRODUCTION.....	1
1.1 Preliminaries.....	1
1.2 Problem Formulation and Thesis Layout.....	2
2 ADDITIVE AND MULTIPLICATIVE NOISES.....	5
2.1 Gaussian Noise.....	6
2.2 Salt and Pepper Noise.....	7
2.3 Speckle Noise.....	8
2.4 Brownian Noise.....	9
2.5 Summary.....	10
3 LINEAR AND NONLINEAR FILTERING APPROACH.....	11
3.1 Background.....	11
3.2 Linear Filtering.....	12
3.2.1 Mean Filter.....	12
3.2.2 LMS Adaptive Filter.....	15
3.3 Median Filter.....	18
3.4 Summary.....	20
4 WAVELET TRANSFORMS AND DENOISING.....	21
4.1 Discrete Wavelet Transform (DWT) – Principles.....	21
4.2 Properties of DWT.....	23
4.3 Mallat’s Algorithm.....	24
4.4 Wavelet Thresholding.....	28
4.4.1 VisuShrink.....	30
4.4.2 SureShrink.....	32
4.4.3 BayesShrink.....	33
4.5 Summary.....	36
5 MULTIFRACTAL IMAGE DENOISING.....	37
5.1 Introduction.....	37
5.2 Hölder Exponents.....	38
5.3 Image Denoising Using Multifractal Analysis.....	39
5.3.1 Multifractal Regularization.....	40
5.3.2 Multifractal Pumping.....	41
5.4 Summary.....	44

6	RESULTS AND CONCLUSION.....	45
6.1	Results.....	45
6.2	Conclusions and Future Work.....	47
	BIBLIOGRAPHY.....	49
	APPENDIX: MATLAB FUNCTIONS.....	52
	VITA.....	54

Abstract

Visual information transmitted in the form of digital images is becoming a major method of communication in the modern age, but the image obtained after transmission is often corrupted with noise. The received image needs processing before it can be used in applications. Image denoising involves the manipulation of the image data to produce a visually high quality image. This thesis reviews the existing denoising algorithms, such as filtering approach, wavelet based approach, and multifractal approach, and performs their comparative study. Different noise models including additive and multiplicative types are used. They include Gaussian noise, salt and pepper noise, speckle noise and Brownian noise. Selection of the denoising algorithm is application dependent. Hence, it is necessary to have knowledge about the noise present in the image so as to select the appropriate denoising algorithm. The filtering approach has been proved to be the best when the image is corrupted with salt and pepper noise. The wavelet based approach finds applications in denoising images corrupted with Gaussian noise. In the case where the noise characteristics are complex, the multifractal approach can be used. A quantitative measure of comparison is provided by the signal to noise ratio of the image.

Chapter 1

Introduction

1.1 Preliminaries

A very large portion of digital image processing is devoted to image restoration. This includes research in algorithm development and routine goal oriented image processing. Image restoration is the removal or reduction of degradations that are incurred while the image is being obtained [Ca79]. Degradation comes from blurring as well as noise due to electronic and photometric sources. Blurring is a form of bandwidth reduction of the image caused by the imperfect image formation process such as relative motion between the camera and the original scene or by an optical system that is out of focus [La91]. When aerial photographs are produced for remote sensing purposes, blurs are introduced by atmospheric turbulence, aberrations in the optical system and relative motion between camera and ground. In addition to these blurring effects, the recorded image is corrupted by noises too. A noise is introduced in the transmission medium due to a noisy channel, errors during the measurement process and during quantization of the data for digital storage. Each element in the imaging chain such as lenses, film, digitizer, etc. contribute to the degradation.

Image denoising is often used in the field of photography or publishing where an image was somehow degraded but needs to be improved before it can be printed. For this type of application we need to know something about the degradation process in order to develop a model for it. When we have a model for the degradation process, the inverse process can be applied to the image to restore it back to the original form. This type of image restoration is often used in space exploration to help eliminate artifacts generated by mechanical jitter in a spacecraft or to compensate for distortion in the optical system of a telescope. Image denoising finds applications in fields such as astronomy where the resolution limitations are severe, in medical imaging where the physical requirements for

high quality imaging are needed for analyzing images of unique events, and in forensic science where potentially useful photographic evidence is sometimes of extremely bad quality [La91].

Let us now consider the representation of a digital image. A 2-dimensional digital image can be represented as a 2-dimensional array of data $s(x,y)$, where (x,y) represent the pixel location. The pixel value corresponds to the brightness of the image at location (x,y) . Some of the most frequently used image types are binary, gray-scale and color images [Um98].

Binary images are the simplest type of images and can take only two discrete values, black and white. Black is represented with the value '0' while white with '1'. Note that a binary image is generally created from a gray-scale image. A binary image finds applications in computer vision areas where the general shape or outline information of the image is needed. They are also referred to as 1 bit/pixel images.

Gray-scale images are known as monochrome or one-color images. The images used for experimentation purposes in this thesis are all gray-scale images. They contain no color information. They represent the brightness of the image. This image contains 8 bits/pixel data, which means it can have up to 256 (0-255) different brightness levels. A '0' represents black and '255' denotes white. In between values from 1 to 254 represent the different gray levels. As they contain the intensity information, they are also referred to as intensity images.

Color images are considered as three band monochrome images, where each band is of a different color. Each band provides the brightness information of the corresponding spectral band. Typical color images are red, green and blue images and are also referred to as RGB images. This is a 24 bits/pixel image.

1.2 Problem Formulation and Thesis Layout

The basic idea behind this thesis is the estimation of the uncorrupted image from the distorted or noisy image, and is also referred to as image "denoising". There are various methods to help restore an image from noisy distortions. Selecting the appropriate method plays a major role in getting the desired image. The denoising methods tend to be problem specific. For example, a method that is used to denoise

satellite images may not be suitable for denoising medical images. In this thesis, a study is made on the various denoising algorithms and each is implemented in Matlab6.1 [Ma01]. Each method is compared and classified in terms of its efficiency. In order to quantify the performance of the various denoising algorithms, a high quality image is taken and some known noise is added to it. This would then be given as input to the denoising algorithm, which produces an image close to the original high quality image. The performance of each algorithm is compared by computing Signal to Noise Ratio (SNR) besides the visual interpretation.

In case of image denoising methods, the characteristics of the degrading system and the noises are assumed to be known beforehand (in two of the techniques considered in Chapters 3 and 4). The image $s(x,y)$ is blurred by a linear operation and noise $n(x,y)$ is added to form the degraded image $w(x,y)$. This is convolved with the restoration procedure $g(x,y)$ to produce the restored image $z(x,y)$.

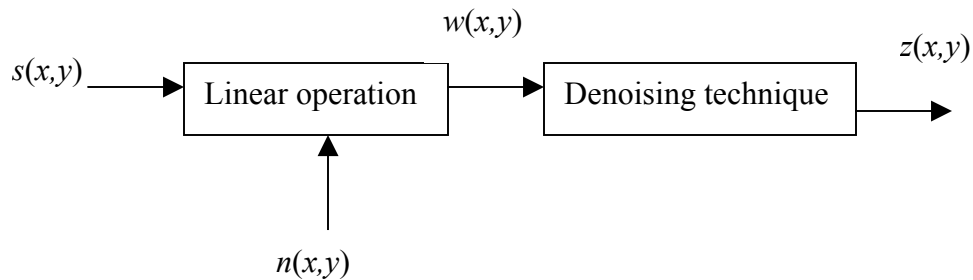


Figure 1.1: Denoising concept

The “Linear operation” shown in Figure 1.1 is the addition or multiplication of the noise $n(x,y)$ to the signal $s(x,y)$ [Im01] (Refer to Chapter 2 for a detailed discussion). Once the corrupted image $w(x,y)$ is obtained, it is subjected to the denoising technique to get the denoised image $z(x,y)$. The point of focus in this thesis is comparing and contrasting several “denoising techniques” (Figure 1.1).

Three popular techniques are studied in this thesis. Noise removal or noise reduction can be done on an image by filtering, by wavelet analysis, or by multifractal analysis. Each technique has its advantages and disadvantages. Denoising by wavelets

and multifractal analysis are some of the recent approaches. Wavelet techniques consider thresholding while multifractal analysis is based on improving the Hölder regularity of the corrupted image.

The rest of this thesis is organized as follows. Chapter 2 discusses noise types considered during the implementation of the various denoising algorithms. It gives the distribution of each type of noise and also presents their effect on an image. Chapter 3 discusses filtering approaches using a linear mean filter [Um98] and the adaptive Least Mean Square (LMS) filter [Li93] to help denoise images. A nonlinear approach based on median filtering is also described. In Chapter 4, we introduce discrete wavelet transforms along with the implementation of Mallat's algorithm [Ma89] and later discuss denoising of images using wavelets. Chapter 5 considers denoising based on multifractal analysis [Lu01]. In this chapter, a tool in Matlab called Fraclab [Ve00] is also introduced which demonstrates the denoising of digital images based on multifractals. Chapter 6 provides a comparative study of all these methods considered for denoising. The quantitative results of comparison are also tabulated by calculating the Signal to Noise Ratio [St01] of the output image. It also provides a future scope on the work described in the thesis.

Chapter 2

Additive and Multiplicative Noises

In this chapter we discuss noise commonly present in an image. Note that noise is undesired information that contaminates the image. In the image denoising process, information about the type of noise present in the original image plays a significant role. Typical images are corrupted with noise modeled with either a Gaussian, uniform, or salt and pepper distribution. Another typical noise is a speckle noise, which is multiplicative in nature. The behavior of each of these noises is described in Section 2.1 through Section 2.4.

Noise is present in an image either in an additive or multiplicative form [Im01]. An additive noise follows the rule

$$w(x, y) = s(x, y) + n(x, y),$$

while the multiplicative noise satisfies

$$w(x, y) = s(x, y) \times n(x, y),$$

where $s(x, y)$ is the original signal, $n(x, y)$ denotes the noise introduced into the signal to produce the corrupted image $w(x, y)$, and (x, y) represents the pixel location. The above image algebra is done at pixel level. Image addition also finds applications in image morphing [Um98]. By image multiplication, we mean the brightness of the image is varied.

The digital image acquisition process converts an optical image into a continuous electrical signal that is, then, sampled [Um98]. At every step in the process there are fluctuations caused by natural phenomena, adding a random value to the exact brightness value for a given pixel.

2.1 Gaussian Noise

Gaussian noise is evenly distributed over the signal [Um98]. This means that each pixel in the noisy image is the sum of the true pixel value and a random Gaussian distributed noise value. As the name indicates, this type of noise has a Gaussian distribution, which has a bell shaped probability distribution function given by,

$$F(g) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(g-m)^2/2\sigma^2},$$

where g represents the gray level, m is the mean or average of the function, and σ is the standard deviation of the noise. Graphically, it is represented as shown in Figure 2.1. When introduced into an image, Gaussian noise with zero mean and variance as 0.05 would look as in Image 2.1 [Im01]. Image 2.2 illustrates the Gaussian noise with mean (variance) as 1.5 (10) over a base image with a constant pixel value of 100.

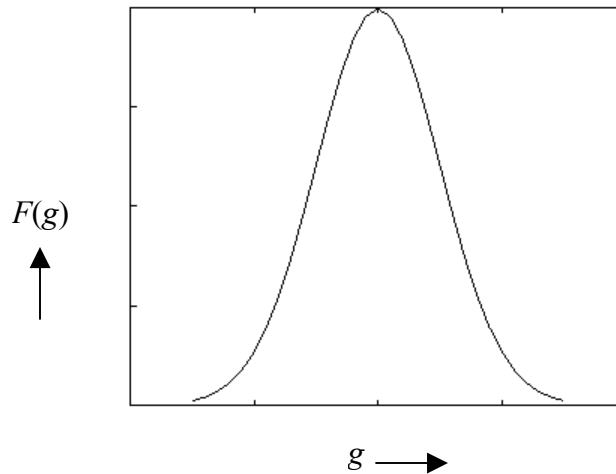


Figure 2.1: Gaussian distribution

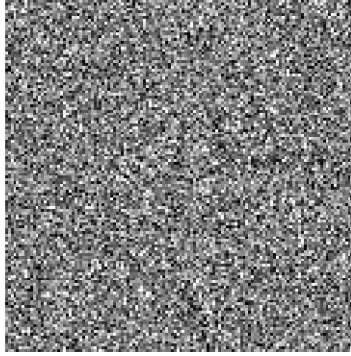


Image 2.1: Gaussian noise
(mean=0, variance 0.05)

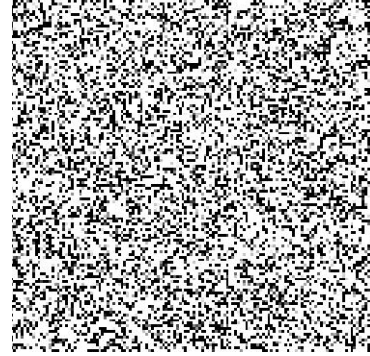


Image 2.2: Gaussian noise
(mean=1.5, variance 10)

2.2 Salt and Pepper Noise

Salt and pepper noise [Um98] is an impulse type of noise, which is also referred to as intensity spikes. This is caused generally due to errors in data transmission. It has only two possible values, a and b . The probability of each is typically less than 0.1. The corrupted pixels are set alternatively to the minimum or to the maximum value, giving the image a “salt and pepper” like appearance. Unaffected pixels remain unchanged. For an 8-bit image, the typical value for pepper noise is 0 and for salt noise 255. The salt and pepper noise is generally caused by malfunctioning of pixel elements in the camera sensors, faulty memory locations, or timing errors in the digitization process. The probability density function for this type of noise is shown in Figure 2.2. Salt and pepper noise with a variance of 0.05 is shown in Image 2.3 [Im01].

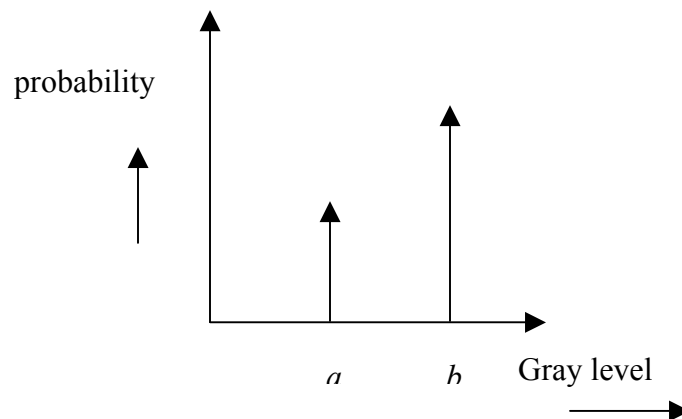


Figure 2.2: PDF for salt and pepper noise

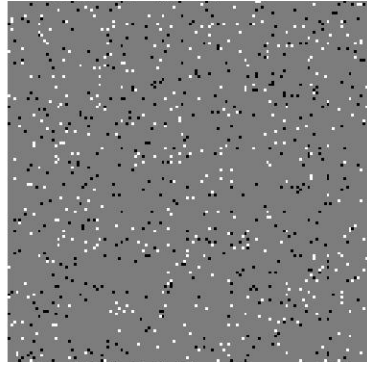


Image 2.3: Salt and pepper noise

2.3 Speckle Noise

Speckle noise [Ga99] is a multiplicative noise. This type of noise occurs in almost all coherent imaging systems such as laser, acoustics and SAR(Synthetic Aperture Radar) imagery. The source of this noise is attributed to random interference between the coherent returns. Fully developed speckle noise has the characteristic of multiplicative noise. Speckle noise follows a gamma distribution and is given as

$$F(g) = \frac{g^{\alpha-1}}{(\alpha-1)!a^\alpha} e^{-\frac{g}{a}},$$

where variance is $a^2\alpha$ and g is the gray level.

On an image, speckle noise (with variance 0.05) looks as shown in Image 2.4 [Im01]. The gamma distribution is given below in Figure 2.3.

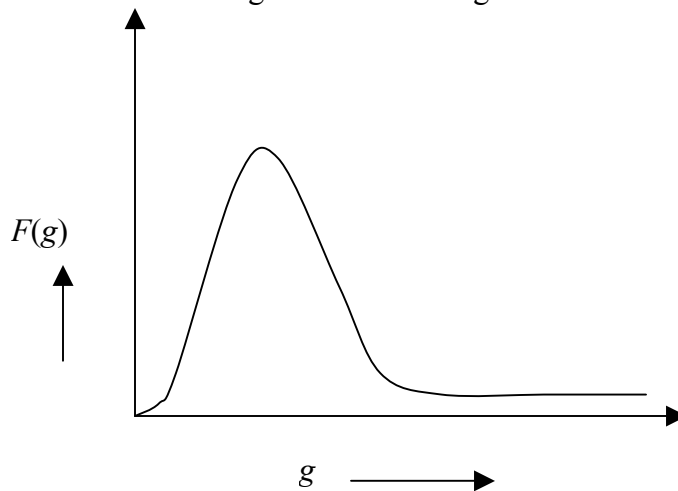


Figure 2.3: Gamma distribution

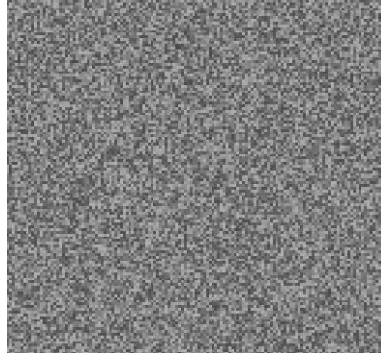


Image 2.4: Speckle noise

2.4 Brownian Noise

Brownian noise [Fr99] comes under the category of fractal or $1/f$ noises. The mathematical model for $1/f$ noise is fractional Brownian motion [Ma68]. Fractional Brownian motion is a non-stationary stochastic process that follows a normal distribution. Brownian noise is a special case of $1/f$ noise. It is obtained by integrating white noise. It can be graphically represented as shown in Figure 2.4. On an image, Brownian noise would look like Image 2.5 which is developed from Fraclab [Ve00].

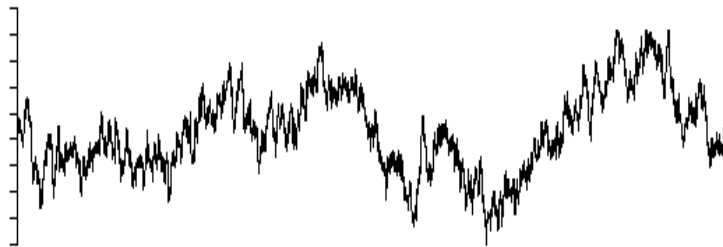


Figure 2.4: Brownian noise distribution

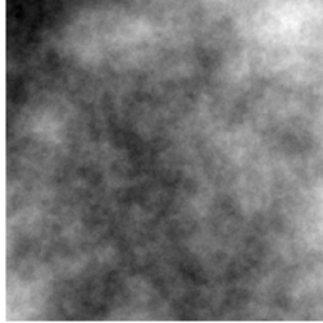


Image 2.5: Brownian noise

2.5 Summary

In this chapter, we have discussed various types of noises considered in the thesis along with their distributions. Gaussian noise, salt and pepper noise, and speckle noise can be generated from the Matlab 6.0 Image Processing tool box function library. Brownian noise is generated using Fraclab [Ve00], a tool in Matlab 6.0, and is added to the image. Based on the background provided so far, the main body of the thesis is discussed in Chapters 3 through 5. Chapter 3 first discusses the filtering approach in denoising.

Chapter 3

Linear and Nonlinear Filtering Approach

Linear filtering using mean filter and Least Mean Square (LMS) adaptive filter and nonlinear filtering based on median filter are discussed in this chapter. Further, the process image denoising is illustrated considering Matlab 6.1 [Ma01] implementations.

3.1 Background

Filters play a major role in the image restoration process. The basic concept behind image restoration using linear filters is digital convolution and moving window principle [Ni86]. Let $w(x)$ be the input signal subjected to filtering, and $z(x)$ be the filtered output. If the filter satisfies certain conditions such as linearity and shift invariance, then the output filter can be expressed mathematically in simple form as [Ni86]

$$z(x) = \int w(t)h(x-t)dt,$$

where $h(t)$ is called the point spread function or impulse response and is a function that completely characterizes the filter. The integral represents a convolution integral and, in short, can be expressed as $z = w * h$.

For a discrete case, the integral turns into a summation as

$$z(i) = \sum_{-\infty}^{+\infty} w(t)h(i-t). \quad (3.1)$$

Although the limits on the summation in Equation (3.1) are ∞ , the function $h(t)$ is usually zero outside some range. If the range over which $h(t)$ is non-zero is $(-k, +k)$, then the above Equation (3.1) can be written as

$$z(i) = \sum_{i-k}^{i+k} w(t)h(i-t). \quad (3.2)$$

This means that the output $z(i)$ at point i is given by a weighted sum of input pixels surrounding i where the weights are given by $h(t)$. To create the output at the next pixel

$i+1$, the function $h(t)$ is shifted by one and the weighted sum is recomputed. The total output is created by a series of shift-multiply-sum operations, and this forms a discrete convolution. For the 2-dimensional case, $h(t)$ is $h(t,u)$, and Equation (3.2) becomes

$$z(i, j) = \sum_{t=i-k}^{i+k} \sum_{u=j-l}^{j+l} w(t,u)h(i-t, j-u).$$

Values of $h(t,u)$ are referred to as the filter weights, the filter kernel, or filter mask. For reasons of symmetry $h(t,u)$ is always chosen to be of size $m \times n$ where m and n are both odd (often $m=n$). In physical systems, the kernel $h(t,u)$ must always be non-negative which results in some blurring or averaging of the image. If the coefficients are alternating positive and negative, the mask is a filter that returns edge information only. The narrower the $h(t,u)$, the better the system in the sense of less blurring. In digital image processing, $h(t,u)$ maybe defined arbitrarily and this gives rise to many types of filters. The weights of $h(t,u)$ may be varied over the image and the size and shape of the window can also be varied. These operations are no longer linear and no longer convolutions. They become moving window operations. With this flexibility, a wide range of linear, non-linear and adaptive filters may be implemented.

3.2 Linear Filtering

3.2.1 Mean Filter

A mean filter [Um98] acts on an image by smoothing it; that is, it reduces the intensity variation between adjacent pixels. The mean filter is nothing but a simple sliding window spatial filter that replaces the center value in the window with the average of all the neighboring pixel values including itself. By doing this, it replaces pixels, that are unrepresentative of their surroundings. It is implemented with a convolution mask, which provides a result that is a weighted sum of the values of a pixel and its neighbors. It is also called a linear filter. The mask or kernel is a square. Often a 3×3 square kernel is used. If the coefficients of the mask sum up to one, then the average brightness of the image is not changed. If the coefficients sum to zero, the average brightness is lost, and it returns a dark image. The mean or average filter works on the shift-multiply-sum

principle [Ni86]. This principle in the two-dimensional image can be represented as shown below (refer to Figure 3.1).

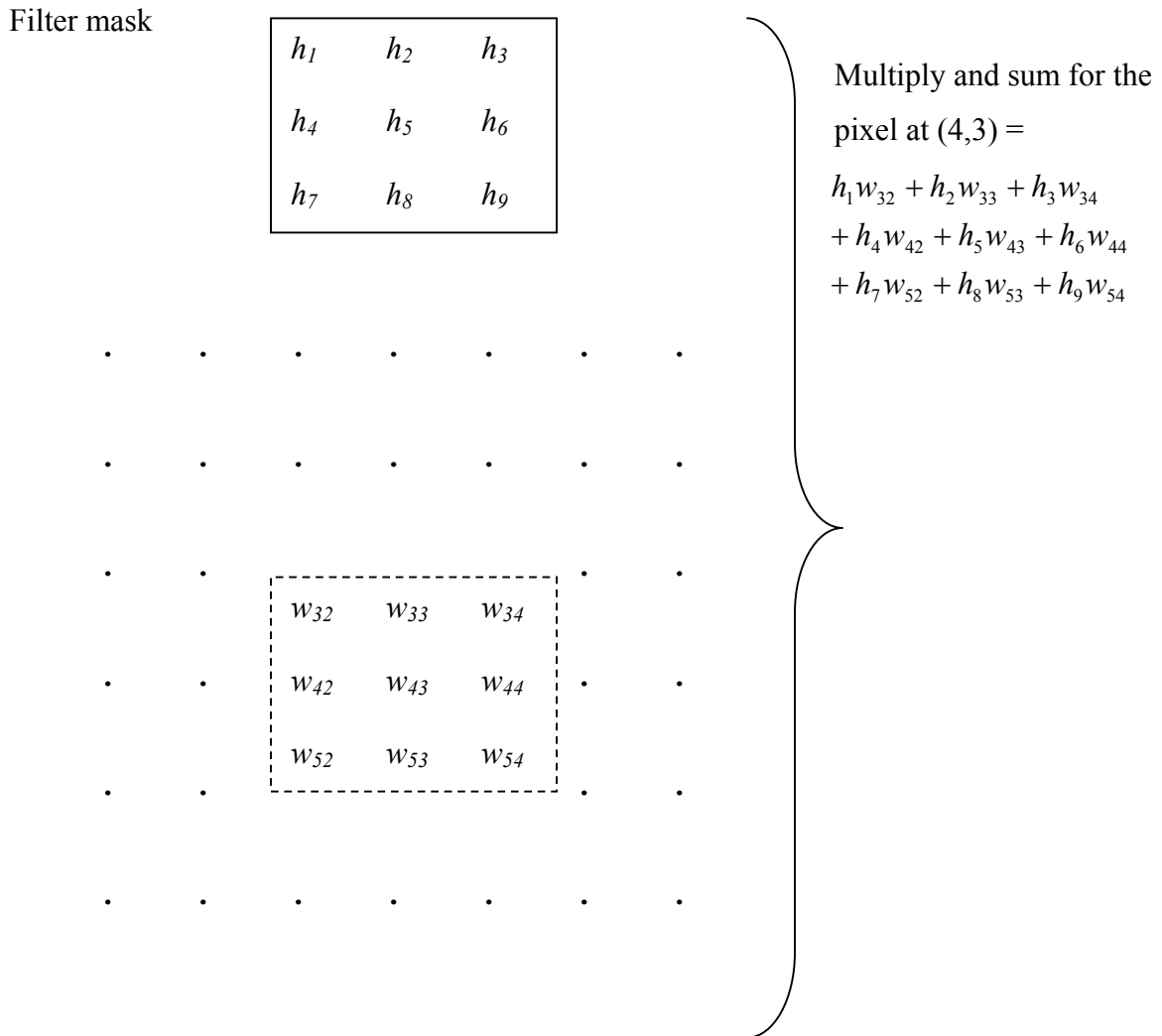


Figure 3.1: Multiply and sum process

The mask used here is a 3×3 kernel shown in Figure 3.2. Note that the coefficients of this mask sum to one, so the image brightness is retained, and the coefficients are all positive, so it will tend to blur the image.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Figure 3.2: A constant weight 3×3 filter mask

Example 3.1: For the following 3×3 neighborhood, mean filtering is applied by convoluting it with the filter mask shown in Figure 3.2.

$$\begin{bmatrix} 70 & 58 & 63 \\ 66 & 200 & 62 \\ 61 & 57 & 65 \end{bmatrix} \times \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

This provides a calculated value of 78. Note that the center value 200, in the pixel matrix, is replaced with this calculated value 78. This clearly demonstrates the mean filtering process.

Computing the straightforward convolution of an image with this kernel carries out the mean filtering process. It is effective when the noise in the image is of impulsive type. The averaging filter works like a low pass filter, and it does not allow the high frequency components present in the noise to pass through. It is to be noted that larger kernels of size 5×5 or 7×7 produces more denoising but make the image more blurred. A trade off is to be made between the kernel size and the amount of denoising.

The filter discussed above is also known as a constant coefficient filter because the weight matrix does not change during the whole process. Mean filters are popular for their simplicity and ease of implementation. We have implemented the averaging filter using Matlab 6.1 [Ma01]. The pixel values of an image “cameraman.tif” are read into the program by using the function **imread()** [Appendix]. This image is of size 256×256 . Salt and pepper noise is added to this image by using the function **imnoise()** [Appendix]. The pixel values of this corrupted image are copied into a 2-dimensional array of size 256×256 . A 3×3 weight matrix is initialized. Selecting a 3×3 window over the 256×256 pixel matrix, the weighted sum of the selected window is computed. The result replaces the center pixel in the window. For the next iteration, the window moves by one column

to the right. The window movement is considered in the horizontal direction first and then in the vertical direction until all the pixels are covered. The modified pixel matrix is now converted to the image format with the help of the function **imwrite()** [Appendix].

Image 3.1 is the one corrupted with salt and pepper noise with a variance of 0.05. The output image after Image 3.1 is subjected to mean filtering is shown in Image 3.2. It can be observed from the output that the noise dominating in Image 3.1 is reduced in Image 3.2. The white and dark pixel values of the noise are changed to be closer to the pixel values of the surrounding ones. Also, the brightness of the input image remains unchanged because of the use of the mask, whose coefficients sum up to the value one.

The mean filter is used in applications where the noise in certain regions of the image needs to be removed. In other words, the mean filter is useful when only a part of the image needs to be processed.



Image 3.1: Input to mean filter corrupted with salt and pepper noise



Image 3.2: Image after mean filtering

3.2.2 LMS Adaptive Filter

An adaptive filter does a better job of denoising images compared to the averaging filter. The fundamental difference between the mean filter and the adaptive filter lies in the fact that the weight matrix varies after each iteration in the adaptive filter while it remains constant throughout the iterations in the mean filter. Adaptive filters are capable of denoising non-stationary images, that is, images that have abrupt changes in

intensity. Such filters are known for their ability in automatically tracking an unknown circumstance or when a signal is variable with little a priori knowledge about the signal to be processed [Li93]. In general, an adaptive filter iteratively adjusts its parameters during scanning the image to match the image generating mechanism. This mechanism is more significant in practical images, which tend to be non-stationary.

Compared to other adaptive filters, the Least Mean Square (LMS) adaptive filter is known for its simplicity in computation and implementation. The basic model is a linear combination of a stationary low-pass image and a non-stationary high-pass component through a weighting function [Li93]. Thus, the function provides a compromise between resolution of genuine features and suppression of noise.

The LMS adaptive filter incorporating a local mean estimator [Li93] works on the following concept. A window, W , of size $m \times n$ is scanned over the image. The mean of this window, μ , is subtracted from the elements in the window to get the residual matrix W^r .

$$W^r = W - \mu \quad (3.3)$$

A weighted sum [Ni86] \tilde{z} , is computed in a way similar to the mean filter using

$$\tilde{z} = \sum_{(i,j) \in W} h(i,j)W^r \quad (3.4)$$

where $h(i,j)$ represents elements of the weight matrix shown in Figure (3.2). A sum of the weighted sum, \tilde{z} , and the mean, μ , of the window replaces the center element of the window. Thus, the resultant modified pixel value is given as

$$z = \tilde{z} + \mu \quad (3.5)$$

For the next iteration, the window is shifted over one pixel in row major order and the weight matrix is modified. The deviation e is computed by taking the difference between the center value of the residual matrix and the weighted sum as Equation (3.6).

$$e = W^r - \tilde{z} \quad (3.6)$$

The largest eigenvalue λ of the original window is calculated from the autocorrelation matrix of the window considered. The use of the largest eigenvalue in computing the modified weight matrix for the next iteration reduces the minimum mean squared error [Tk99]. A value η is selected such that it lies in the range $(0, 1/\lambda)$. In other words,

$$0 < \eta < 1/\lambda.$$

The new weight matrix h_{k+1} is

$$h_{k+1} = h_k + \eta \times e \times W^r \quad (3.7)$$

where h_k is the weight matrix from the previous iteration. The weight matrix obtained this way is used in the next iteration. The process continues until the window covers the entire image.

The LMS adaptive filter incorporating a local mean estimator is implemented in Matlab 6.1 [Ma01]. The pixel values of an image “cameraman.tif” are read into the program by using the function **imread()** [Appendix]. This image is of size 256×256 . Salt and pepper noise is added to this image by using the function **imnoise()** [Appendix]. The pixel values of this corrupted image are copied into a 2-dimensional array of size 256×256 . A 3×3 weight matrix, given in Figure 3.2, is initialized. A 5×5 window is scanned over the pixel matrix but the operations from Equations (3.3) through (3.6) are done on a 3×3 . The remaining pixels in the 5×5 window are used in the calculation of the autocorrelation matrix of the 3×3 window. The largest eigenvalue of the autocorrelation matrix is obtained with the help of the function **max(eig())** [Appendix]. The modified weight matrix is now computed based on Equation (3.7). The window traverses right by one column and the procedure is repeated with “**for**” loops until the window covers the entire image. The modified pixel matrix is now converted to the image format with the help of the function **imwrite()** [Appendix].

When the image is corrupted with salt and pepper noise, it looks as shown in Image 3.3. When Image 3.3 is subjected to the LMS adaptive filtering, it gives an output image shown in Image 3.4. Similar to the mean filter, the LMS adaptive filter works well for images corrupted with salt and pepper type noise. But this filter does a better denoising job compared to the mean filter.



Image 3.3: Input to LMS adaptive filter corrupted with salt and pepper noise



Image 3.4: Image after LMS adaptive filtering

3.3 Median Filter

A median filter belongs to the class of nonlinear filters unlike the mean filter. The median filter also follows the moving window principle similar to the mean filter. A 3×3 , 5×5 , or 7×7 kernel of pixels is scanned over pixel matrix of the entire image. The median of the pixel values in the window is computed, and the center pixel of the window is replaced with the computed median. Median filtering is done by, first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. Note that the median value must be written to a separate array or buffer so that the results are not corrupted as the process is performed. Figure 3.3 illustrates the methodology.

123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighborhood values:

115,119,120,123,124,125,126,127,150

Median value: 124

Figure 3.3: Concept of median filtering

The central pixel value of 150 in the 3×3 window shown in Figure 3.3 is rather unrepresentative of the surrounding pixels and is replaced with the median value of 124.

The median is more robust compared to the mean. Thus, 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. These advantages aid median filters in denoising uniform noise as well from an image.

The image processing toolbox [Im01] in Matlab 6.1 [Ma01] provides the **medfilt2()** [Appendix] function to do median filtering on an image. The input image and the size of the window are the parameters the function takes. As mentioned earlier, the image “cameraman.tif” is corrupted with salt and pepper noise with the **imnoise()** [Appendix] function after loading the image using **imread()** [Appendix]. Image 3.5 is the image corrupted with salt and pepper noise and is given to the function **medfilt2()** for median filtering. The window specified is of size 3×3 . Image 3.6 is the output after median filtering. It can be observed that the edges are preserved and the quality of denoising is much better compared to the Images 3.2 and 3.4.



Image 3.5: Input to median filter



Image 3.6: Output from median filter

3.4 Summary

In this chapter, we have focused on the denoising of images using the linear and nonlinear filtering techniques where linear filtering is done using the mean filter and the LMS adaptive filter while the nonlinear filtering is performed using a median filter. These filters are good for removing noise that is impulsive in nature. The mean filters find applications where a small region in the image is concentrated. Besides, implementation of such filters is easy, fast, and cost effective. It can be observed from the output Images (3.2) and (3.4) that the filtered images are blurred. The median filter provides a solution to this, where the sharpness of the image is retained after denoising. From our experimentation, it has been observed that the filtering approach does not produce considerable denoising for images corrupted with Gaussian noise or speckle noise. Wavelets play a very important role in the removal of the noise, especially when it is of the Gaussian type. We consider this technique in Chapter 4.

Chapter 4

Wavelet Transforms and Denoising

4.1. Discrete Wavelet Transform (DWT) - Principles

Wavelets are mathematical functions that analyze data according to scale or resolution [Gr95]. They aid in studying a signal in different windows or at different resolutions. For instance, if the signal is viewed in a large window, gross features can be noticed, but if viewed in a small window, only small features can be noticed.

Wavelets provide some advantages over Fourier transforms. For example, they do a good job in approximating signals with sharp spikes or signals having discontinuities. Wavelets can also model speech, music, video and non-stationary stochastic signals. Wavelets can be used in applications such as image compression, turbulence, human vision, radar, earthquake prediction, etc. [Gr95].

The term “wavelets” is used to refer to a set of orthonormal basis functions generated by dilation and translation of scaling function ϕ and a mother wavelet ψ [An01]. The finite scale multiresolution representation of a discrete function can be called as a discrete wavelet transform [Wa01]. DWT is a fast linear operation on a data vector, whose length is an integer power of 2. This transform is invertible and orthogonal, where the inverse transform expressed as a matrix is the transpose of the transform matrix. The wavelet basis or function, unlike sines and cosines as in Fourier transform, is quite localized in space. But similar to sines and cosines, individual wavelet functions are localized in frequency.

The orthonormal basis or wavelet basis is defined as [Ti92]

$$\psi_{(j,k)}(x) = 2^{j/2} \psi(2^j x - k). \quad (4.1)$$

The scaling function is given as [Ti92]

$$\phi_{(j,k)}(x) = 2^{j/2} \phi(2^j x - k), \quad (4.2)$$

where ψ is called the wavelet function and j and k are integers that scale and dilate the wavelet function. The factor ‘ j ’ in Equations (4.1) and (4.2) is known as the scale index, which indicates the wavelet’s width. The location index k provides the position. The wavelet function is dilated by powers of two and is translated by the integer k . In terms of the wavelet coefficients, the wavelet equation [Ti92] is

$$\psi(x) = \sum_k^{N-1} g_k \sqrt{2} \phi(2x - k), \quad (4.3)$$

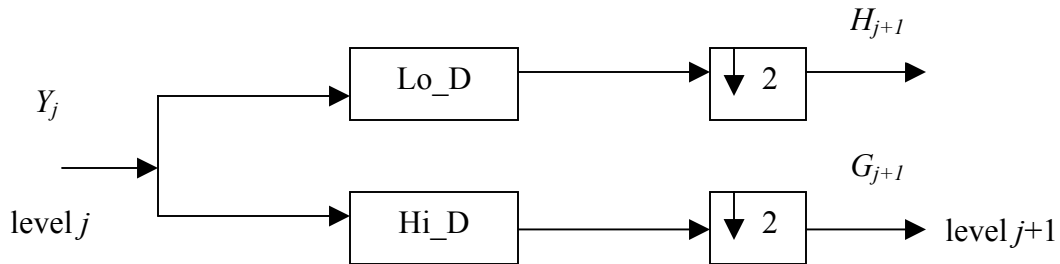
where g_0, g_1, g_2, \dots are high pass wavelet coefficients. Writing the scaling equation [Ti92] in terms of the scaling coefficients as given below, we get

$$\phi(x) = \sum_k^{N-1} h_k \sqrt{2} \phi(2x - k). \quad (4.4)$$

The function $\phi(x)$ is the scaling function and the coefficients h_0, h_1, h_2, \dots are low pass scaling coefficients. The wavelet and scaling coefficients are related by the quadrature mirror relationship, which is

$$g_n = (-1)^n h_{1-n+N}$$

The term N is the number of vanishing moments [Ti92]. A graphical representation of DWT is shown in Figure 4.1. Note that, Y_0 is the initial signal.



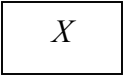
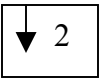
where  convolve with filter X and
 downsampling

Figure 4.1: A 1-Dimensional DWT - Decomposition step

As mentioned earlier, the wavelet equation produces different wavelet families like Daubechies, Haar, coiflets, etc. [Wa01]. Wavelets are classified into a family by the number of vanishing moments N . Within each family of wavelets there are wavelet subclasses distinguished by the number of coefficients and by the level of iterations. The filter lengths and the number of vanishing moments for four different wavelet families are tabulated in Table 4.1.

Table 4.1: Wavelet families and their properties [Ma01]

Wavelet Family	Filters length	Number of vanishing moments, N
Haar	2	1
Daubechies M	$2M$	M
Coiflets M	$6M$	$2M-1$
Symlets	$2M$	M

4.2 Properties of DWT

Some of the properties of discrete wavelet transforms are listed below [Gr95, Vi99].

- DWT is a fast linear operation, which can be applied on data vectors having length as integer power of 2.
- DWT is invertible and orthogonal. Note that the scaling function ϕ and the wavelet function ψ are orthogonal to each other in $L^2(0, 1)$, i.e., $\langle \phi, \psi \rangle = 0$.
- The wavelet basis is quite localized in space and frequency.
- The coefficients satisfy some constraints

$$\sum_{i=0}^{2N-1} h_i = \sqrt{2} \quad (4.5)$$

$$\sum_{i=0}^{2N-1} h_i h_{i+2l} = \delta_{l,0} \quad (4.6)$$

Here δ is the delta function and l is the location index.

$$\sum_{i=0}^{2N-1} (-1)^i i^k h_i = 0 \quad (4.7)$$

In all the above relations, N represents the number of vanishing moments.

- The wavelet coefficients of a fractional Brownian motion (fBm) supports Stationarity, i.e., $g_j(k) = g_j(0), \forall k$.
- Wavelet coefficients exhibit Gaussianity:
 $g_j(k) \sim N(0, \sigma_\psi 2^{2jH})$, where σ_ψ is a constant depending on ψ and H , the Hurst parameter for fBm. This property aids wavelets in the removal of Gaussian noise from images.
- The wavelet coefficients are almost decorrelated,
 $E[g_j(k)g_{j'}(k')] \approx |2^{-j}k - 2^{-j}k'|^{2(H-N)}$, where N refers to the number of vanishing moments.

Equations (4.5) through (4.7) are used to compute the scaling and wavelet coefficient values of the corresponding wavelet family. For Haar wavelet transform,

$h_0 = h_1 = 1/\sqrt{2}$ and $g_0 = -g_1 = 1/\sqrt{2}$. In the case of Daubechies 2 wavelets,

$$h_0 = \frac{1 + \sqrt{3}}{4\sqrt{2}}, h_1 = \frac{3 + \sqrt{3}}{4\sqrt{2}}, h_2 = \frac{3 - \sqrt{3}}{4\sqrt{2}}, h_3 = \frac{1 - \sqrt{3}}{4\sqrt{2}} \text{ and}$$

$$g_0 = \frac{1 - \sqrt{3}}{4\sqrt{2}}, g_1 = \frac{-(3 - \sqrt{3})}{4\sqrt{2}}, g_2 = \frac{3 + \sqrt{3}}{4\sqrt{2}}, g_3 = \frac{-(1 + \sqrt{3})}{4\sqrt{2}}.$$

The Gaussianity property exhibited by wavelets aids in denoising images corrupted with additive Gaussian noise. The decorrelation exhibited by the wavelet coefficients is important because it explains a Karhunen-Loeve-like expansion that is implicitly performed for $1/f$ processes using orthogonal wavelet bases.

4.3 Mallat's Algorithm

Mallat's algorithm [Ma68] is a computationally efficient method of implementing the wavelet transform. It calculates DWT wavelet coefficients for a finite set of input data, which is a power of 2. This input data is passed through two convolution functions, each of which creates an output stream that is half the length of the original input. This procedure is referred to as down sampling [Wi92]. The convolution functions are filters. One half of the output is produced by the low pass filter function defined by Equation

(4.4) and the other half is produced by the high pass filter function defined by Equation (4.3). The low pass outputs contain most of the information of the input signal and are known as “coarse” coefficients. The outputs from the high pass filter are known as “detail” coefficients.

The coefficients obtained from the low pass filter are used as the original signal for the next set of coefficients. This procedure is carried out recursively until a trivial number of low pass filter coefficients are left. The final output contains the remaining low pass filter outputs and the accumulated high pass filter outputs. This procedure is termed as decomposition.

In certain applications, some form of processing is done to the wavelet coefficients obtained after the DWT. Once the processing is done, the data vector is built back from the coefficients. This processes of reconstruction is referred to as the inverse Mallat’s algorithm.

In the reconstruction procedure, quadrature mirror filters Equation (4.3) and Equation (4.4) are supplied with the coarse coefficients and the accumulated detail coefficients. The so obtained outputs of the two filters are summed and are treated as the coarse coefficients for the next stage of reconstruction. This procedure is continued until the data vector is obtained. The numerical example below demonstrates Mallat’s algorithm and the inverse Mallat’s algorithm.

Example 4.1: Consider the one dimensional signal

$$Y = [1 \quad 0 \quad -3 \quad 2 \quad 1 \quad 0 \quad 1 \quad 2]$$

Applying the Haar wavelet transform to the above signal, with

$$H = \frac{Y_{2k} + Y_{2k+1}}{\sqrt{2}}, \text{ which gives the coarser approximation coefficients and}$$

$$G = \frac{Y_{2k} - Y_{2k+1}}{\sqrt{2}}, \text{ which gives the detail coefficients.}$$

$$h_0 = h_1 = 1/\sqrt{2}, \text{ the low pass filter coefficients}$$

$$g_0 = -g_1 = 1/\sqrt{2}, \text{ the high pass filter coefficients.}$$

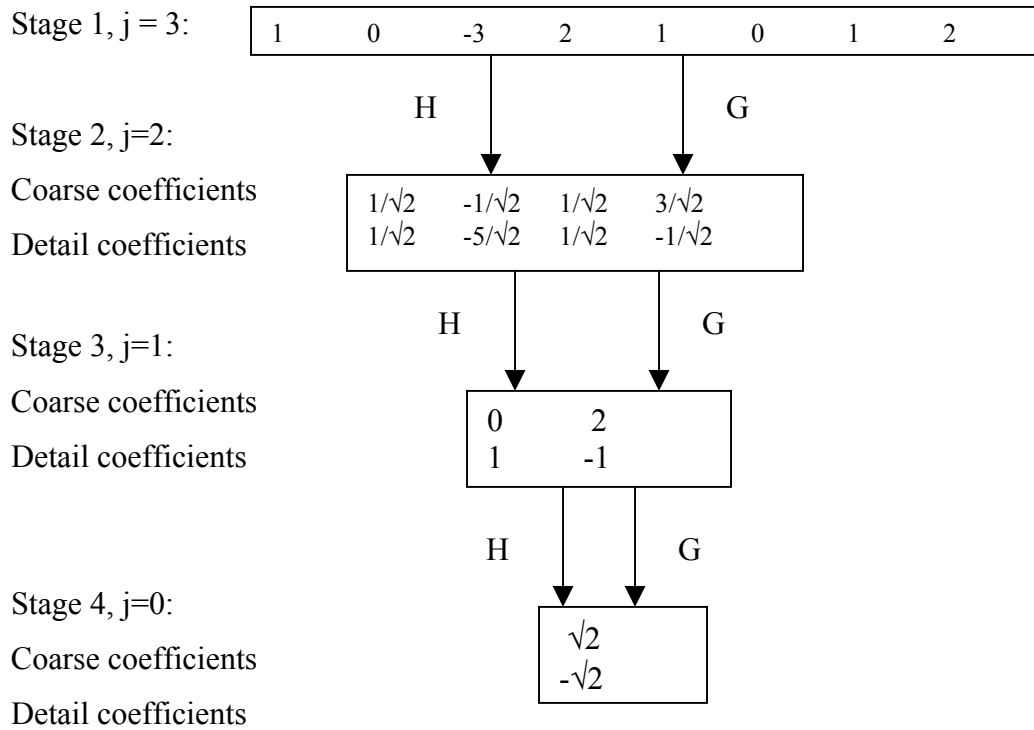


Figure 4.2: Mallat's algorithm (decomposition phase)

The original signal containing 8 elements is decomposed to the final stage containing only two elements by applying the Mallat's algorithm. For reconstruction, consider these two elements. Figure 4.3 illustrates the reconstruction procedure. Note that the final output of the reconstruction algorithm is the original data vector.

4.4 Wavelet Thresholding

Donoho and Johnstone [Do94] pioneered the work on filtering of additive Gaussian noise using wavelet thresholding. From their properties and behavior, wavelets play a major role in image compression and image denoising. Since our topic of interest is image denoising, the latter application is discussed in detail. Wavelet coefficients calculated by a wavelet transform represent change in the time series at a particular resolution. By considering the time series at various resolutions, it is then possible to filter out noise.

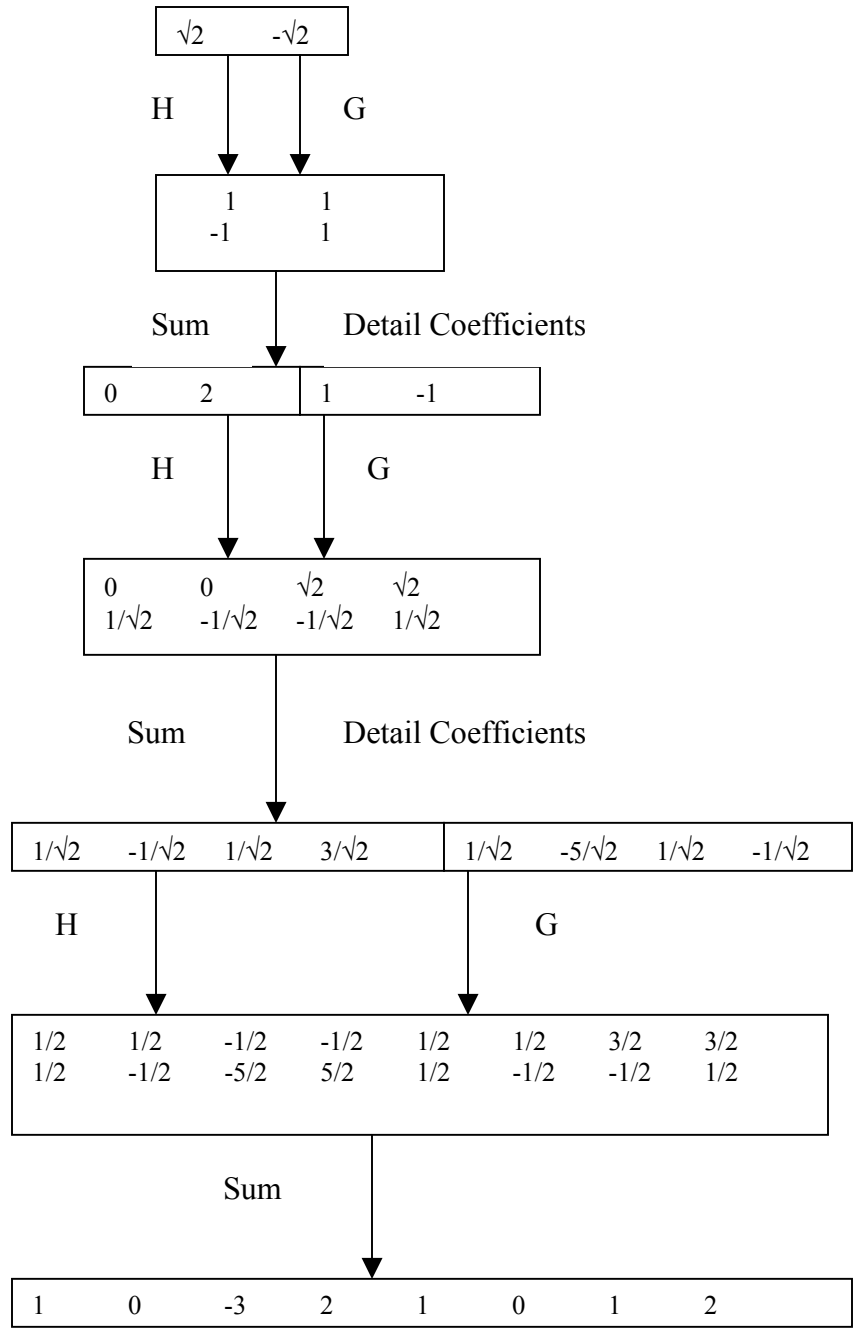


Figure 4.3: Inverse Mallat's algorithm (reconstruction phase)

The term wavelet thresholding is explained as decomposition of the data or the image into wavelet coefficients, comparing the detail coefficients with a given threshold value, and shrinking these coefficients close to zero to take away the effect of noise in the data. The image is reconstructed from the modified coefficients. This process is also known as the inverse discrete wavelet transform. During thresholding, a wavelet coefficient is compared with a given threshold and is set to zero if its magnitude is less than the threshold; otherwise, it is retained or modified depending on the threshold rule. Thresholding distinguishes between the coefficients due to noise and the ones consisting of important signal information.

The choice of a threshold is an important point of interest. It plays a major role in the removal of noise in images because denoising most frequently produces smoothed images, reducing the sharpness of the image. Care should be taken so as to preserve the edges of the denoised image. There exist various methods for wavelet thresholding, which rely on the choice of a threshold value. Some typically used methods for image noise removal include VisuShrink, SureShrink and BayesShrink [An01, Ch00, Do94].

Prior to the discussion of these methods, it is necessary to know about the two general categories of thresholding. They are hard- thresholding and soft-thresholding types. The hard-thresholding T_H can be defined as [Do92]

$$T_H = \begin{cases} x & \text{for } |x| \geq t \\ 0 & \text{in all other regions.} \end{cases}$$

Here t is the threshold value. A plot of T_H is shown in Figure 4.4.

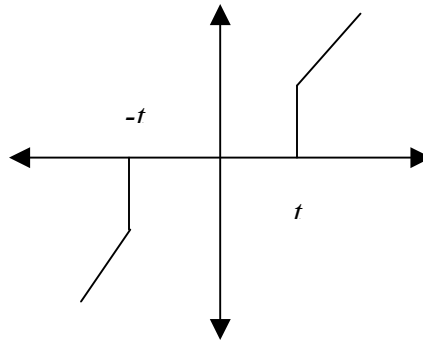


Figure 4.4: Hard thresholding

Thus, all coefficients whose magnitude is greater than the selected threshold value t remain as they are and the others with magnitudes smaller than t are set to zero. It creates a region around zero where the coefficients are considered negligible.

Soft thresholding is where the coefficients with greater than the threshold are shrunk towards zero after comparing them to a threshold value. It is defined as follows [Do92],

$$T_s = \begin{cases} \text{sign}(x)(|x| - t) & \text{for } |x| > t \\ 0 & \text{in all other regions.} \end{cases}$$

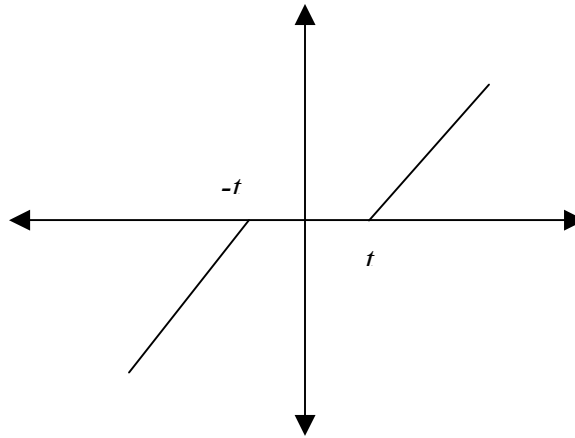


Figure 4.5: Soft thresholding

In practice, it can be seen that the soft method is much better and yields more visually pleasant images. This is because the hard method is discontinuous and yields abrupt artifacts in the recovered images. Also, the soft method yields a smaller minimum mean squared error compared to hard form of thresholding.

Now let us focus on the three methods of thresholding mentioned earlier. For all these methods the image is first subjected to a discrete wavelet transform, which decomposes the image into various sub-bands. Graphically it can be represented as shown in Figure 4.6.

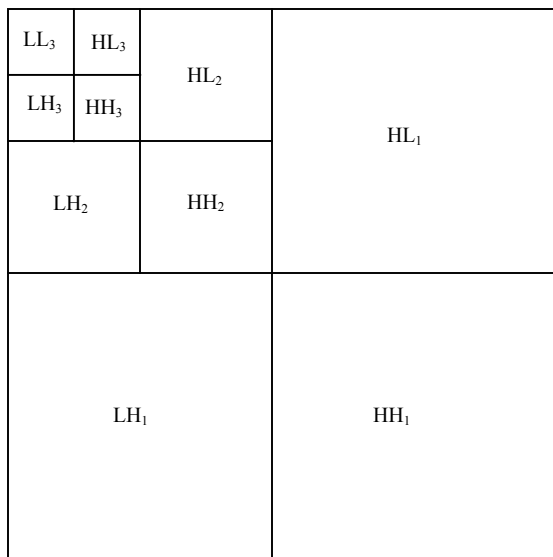


Figure 4.6: DWT on 2-dimensional data

The sub-bands $HH_k, HL_k, LH_k, k = 1, 2, \dots, j$ are called the details, where k is the scale and j denotes the largest or coarsest scale in decomposition. Note, LL_k is the low-resolution component. Thresholding is now applied to the detail components of these sub bands to remove the unwanted coefficients, which contribute to noise. And as a final step in the denoising algorithm, the inverse discrete wavelet transform is applied to build back the modified image from its coefficients.

4.4.1 VisuShrink

VisuShrink was introduced by Donoho [Do92]. It uses a threshold value t that is proportional to the standard deviation of the noise. It follows the hard thresholding rule. It is also referred to as universal threshold and is defined as

$$t = \sigma \sqrt{2 \log n} \tag{4.8}$$

σ^2 is the noise variance present in the signal and n represents the signal size or number of samples. An estimate of the noise level σ was defined based on the median absolute deviation [Do94] given by

$$\hat{\sigma} = \frac{\text{median}\left(\left\{|g_{j-1,k}| : k = 0, 1, \dots, 2^{j-1} - 1\right\}\right)}{0.6745} \tag{4.9}$$

where $g_{j-l,k}$ corresponds to the detail coefficients in the wavelet transform.

VisuShrink does not deal with minimizing the mean squared error [Ch00]. It can be viewed as general-purpose threshold selectors that exhibit near optimal minimax error properties and ensures with high probability that the estimates are as smooth as the true underlying functions [Do92]. However, VisuShrink is known to yield recovered images that are overly smoothed. This is because VisuShrink removes too many coefficients. Another disadvantage is that it cannot remove speckle noise. It can only deal with an additive noise. VisuShrink follows the global thresholding [An01] scheme where there is a single value of threshold applied globally to all the wavelet coefficients.

The VisuShrink algorithm has been implemented using Matlab 6.1 [Ma01]. The images “cameraman.tif” and “moon.tif” are read using the **imread()** function. Noise is added to the image using **imnoise()**. The threshold value using Equation (4.8) is computed from the function **ddencmp()**. A global thresholding is applied over the wavelet coefficients with **wdencmp()**. The modified image is obtained with **imwrite()** function. The images shown from Image 4.1 through 4.4 exhibit the effect of VisuShrink thresholding. Note that all Matlab 6.1 functions are given in Appendix.



Image 4.1: Image corrupted with Gaussian noise, variance 0.005



Image 4.2: Image after application of VisuShrink

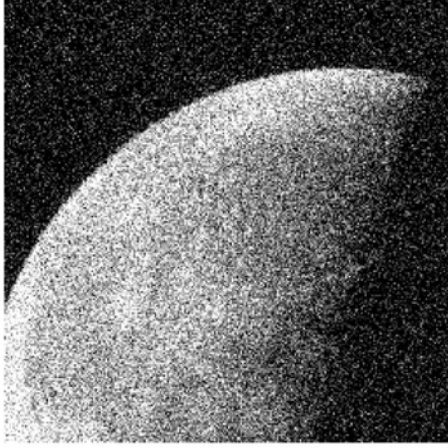


Image 4.3: Image corrupted with Gaussian noise, variance 0.05

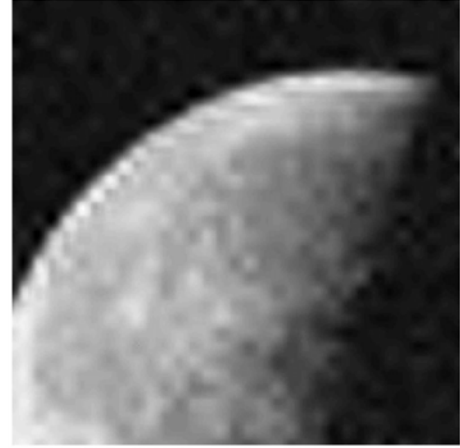


Image 4.4: Image after application of VisuShrink

4.4.2 SureShrink

A threshold chooser based on Stein's Unbiased Risk Estimator (SURE) was proposed by Donoho and Johnstone [Do94] and is called as SureShrink. It is a combination of the universal threshold and the SURE threshold. This method specifies a threshold value t_j for each resolution level j in the wavelet transform which is referred to as level dependent thresholding [An01]. The goal of SureShrink is to minimize the mean squared error, defined as [Ch00]

$$\text{MSE} = \frac{1}{n^2} \sum_{x,y=1}^n (z(x,y) - s(x,y))^2,$$

where $z(x,y)$ is the estimate of the signal while $s(x,y)$ is the original signal without noise and n is the size of the signal. SureShrink suppresses noise by thresholding the empirical wavelet coefficients. The SureShrink threshold t^* is defined as

$$t^* = \min(t, \sigma\sqrt{2\log n}),$$

where t denotes the value that minimizes Stein's Unbiased Risk Estimator, σ is the noise variance computed from Equation (4.9), and n is the size of the image. SureShrink follows the soft thresholding rule. The thresholding employed here is adaptive, i.e., a threshold level is assigned to each dyadic resolution level by the principle of minimizing the Stein's Unbiased Risk Estimator for threshold estimates. It is smoothness adaptive,

which means that if the unknown function contains abrupt changes or boundaries in the image, the reconstructed image also does.

SureShrink is implemented using Matlab 6.1 [Ma01]. With the function **imread()**, the image “moon.tif” is loaded into the Matlab workspace. This image is corrupted with Gaussian noise with the help of **imnoise()** function. Daubechies wavelet decomposition is done on the corrupted image with **wavedec2()** function. The threshold values are computed using the function, **wdcbm2()**. Sure thresholding is done on the detail coefficients with **wdencmp()** specifying the parameter, ‘**lvd**’ in the function which means that level dependent thresholding is done on the coefficients.

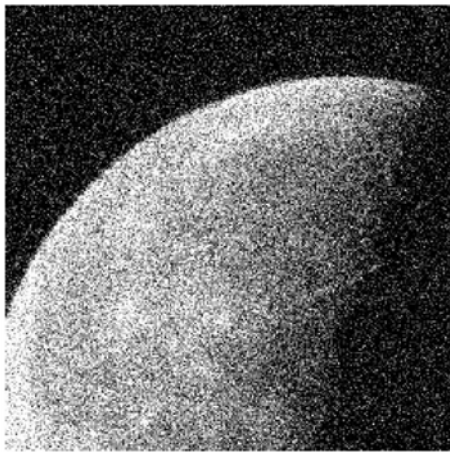


Image 4.5: Input corrupted with Gaussian noise



Image 4.6: Image after SureShrink thresholding

Image 4.5 is the image corrupted with Gaussian noise. Image 4.6 is the one obtained after Image 4.5 is subjected to SureShrink thresholding. The modified image is obtained using **imwrite()**. All Matlab 6.1 functions used here are listed in Appendix.

4.4.3 BayesShrink

BayesShrink was proposed by Chang, Yu and Vetterli [Ch00]. The goal of this method is to minimize the Bayesian risk, and hence its name, BayesShrink. It uses soft thresholding and is subband-dependent, which means that thresholding is done at each band of resolution in the wavelet decomposition. Like the SureShrink procedure, it is smoothness adaptive. The Bayes threshold, t_B , is defined as

$$t_B = \sigma^2 / \sigma_s^2 . \quad (4.10)$$

where σ^2 is the noise variance and σ_s^2 is the signal variance without noise. The noise variance σ^2 is estimated from the subband HH₁ in Figure 4.6 by the median estimator shown in Equation (4.9). From the definition of additive noise we have

$$w(x, y) = s(x, y) + n(x, y) .$$

Since the noise and the signal are independent of each other, it can be stated that

$$\sigma_w^2 = \sigma_s^2 + \sigma^2 .$$

σ_w^2 can be computed as shown below:

$$\sigma_w^2 = \frac{1}{n^2} \sum_{x,y=1}^n w^2(x, y) .$$

The variance of the signal, σ_s^2 is computed as

$$\sigma_s = \sqrt{\max(\sigma_w^2 - \sigma^2, 0)} . \quad (4.11)$$

With σ^2 and σ_s^2 , the Bayes threshold is computed from Equation (4.10). Using this threshold, the wavelet coefficients are thresholded at each band shown in Figure 4.6.

Matlab 6.1 [Ma01] is used for the implementation of BayesShrink. The image “cameraman.tif” is loaded into the workspace by using **imread()**. This image is corrupted with Gaussian noise using the **imnoise()** function. The image obtained is subjected to a discrete wavelet transform using Daubechies wavelets with the help of the **dwt2()** function. This function generates wavelet coefficients for the corrupted image. There are four bands namely, cA, cH, cV and cD, where cA corresponds to the approximation coefficients, while cH, cV, and cD are the detail coefficients over which thresholding is done. The noise variance for each band is computed using Equation (4.9) and the signal variance is computed using Equation (4.11). With these two values, the threshold value is computed from Equation (4.10). Thresholding of the wavelet coefficients is brought about using the function **wthresh()**. Inverse wavelet transform using **idwt2()** is done on the modified wavelet coefficients to get the signal. The image is built from this signal using **imwrite()** function. All the Matlab 6.1 functions used here are listed in Appendix. BayesShrink has been experimented to remove Gaussian noise (mean=0, variance = 0.05) and speckle noise (variance = 0.05). The input and output images after applying BayesShrink can be seen in the Images 4.7 through 4.10.

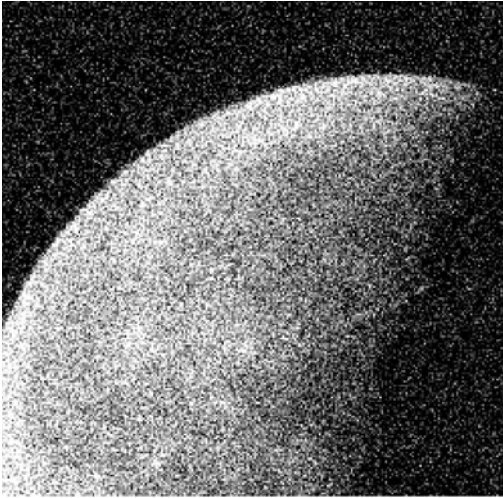


Image 4.7: Image corrupted with Gaussian noise

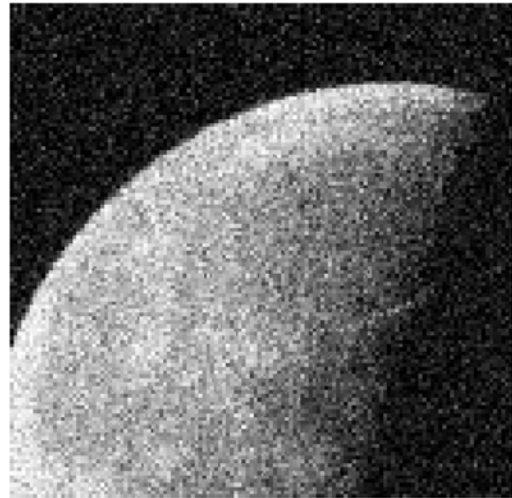


Image 4.8: Image subjected to BayesShrink



Image 4.9: Image corrupted with speckle noise



Image 4.10: Image subjected to BayesShrink

4.5 Summary

Denoising of images using VisuShrink, SureShrink and BayesShrink using Matlab 6.1 is discussed in this chapter. All these methods are based on the application of wavelet transforms. Each of these methods is compared in terms of the signal to noise ratio discussed in Chapter 6 of this thesis. Chapter 5 deals with denoising of images using the multifractal approach.

Chapter 5

Multifractal Image Denoising

This chapter deals with one of the most recent techniques in image denoising, known as multifractal analysis. A brief introduction on multifractals is given first before we describe the use of multifractals in image noise removal. The denoising algorithms are implemented in Matlab [Ma01] using Fraclab [Ve00] tools.

5.1 Introduction

The phenomenon of multifractals was first described by B. B. Mandelbrot in the context of fully developed turbulence [Ga96]. Multifractal structures are generated by the multiplicative cascade of random processes, while additive processes generally produce simple fractals or monofractals. Functions that are everywhere continuous but nowhere differentiable are called fractals. They are objects of a complex structure, which exhibit the scaling property, that is, they exhibit the same properties at different scales. A fractal describes the local singularity and is usually measured using the Hurst parameter. The fractal dimension is the basic notion for describing structures that have scaling symmetry and is closely related to Hölder regularity (see Section 5.2). Fractal dimension is a non-integer value. Multifractal analysis gives a compact representation of the spectral decomposition of a signal into parts of equal strength of regularity [Ri98]. This property makes multifractals very useful in image denoising. Other applications of multifractals are in the fields of turbulence, rainfall, dynamical systems and in earthquake modeling [Ha01].

Denoising by multifractal analysis is based on the fact that signal enhancement is equivalent to increasing the Hölder regularity at each point [Ve01]. It is well adapted to the case where the signal to be recovered is very irregular and nowhere differentiable, a property relevant to fractal or self-similar structures. The local regularity of a function is measured by the local Hölder exponent, which is a local notion. Since the Hölder

exponent is a local notion, this scheme is valid for signals that have sudden changes in regularity like discontinuities. To any continuous function we can associate its Hölder function, which gives the value of the Hölder exponent of the function at every point. In image denoising using multifractal analysis, the Hölder regularity of the input signal is manipulated so that it is close to the regularity of the desired signal (see section 5.3.1 and 5.3.2). The regularity of a function can be determined by geometrical and analytical ways. In the geometrical case, the regularity is obtained by computing the fractional dimensions of its graph. The analytical way considers a family of nested functional spaces and determines the ones to which the function actually belongs [Ve01]. Generally, the second method is more practical and, hence, popular.

Denoising by multifractal analysis makes no assumptions on the type of noise present in the signal. Also, noise is considered to be independent of the signal. This procedure is suitable for signals, that are everywhere irregular, and the regularity of the original signal may vary rapidly in time or space.

Section 5.2 describes the concept of Hölder exponent and provides the reason for its importance in image denoising.

5.2 Hölder Exponents

Holder exponent α_x of a function f at x can be defined as [Ma94]

$$\alpha_x = \liminf_{\epsilon \rightarrow 0} \left\{ \frac{\log |f(x) - f(y)|}{\log |x - y|} \mid y \in B_\epsilon(x) \right\},$$

where $B_\epsilon(x)$ denotes a ball of radius $\epsilon > 0$ centered at x . The Hölder exponent is a widely used tool for measuring the pointwise regularity of signals. The regularity characterizations are widely used in fractal analysis because they have direct interpretations in various applications. Computing the Hölder exponent at each point in an image gives an idea of its structure, especially of the edges [Ve97]. There are two types of Hölder exponents, namely, pointwise Hölder exponent and local Hölder exponent. The mathematical definitions of each of these terms is discussed in [Ve01].

Since our point of interest is image denoising using the multifractal analysis, the detailed mathematical treatment is not given here.

The pointwise Hölder exponent characterizes the regularity of a function under consideration at any given point. It is represented as α_p , and it corresponds to the auditive perception of smoothness for voice signals [Ve01]. The pointwise Hölder exponent is not stable under the action of differential operators, and this exponent is not sufficient to predict the pointwise Hölder exponent of its derivative. The local Hölder exponent is related to the regularity of a function under consideration around any given point. It is always smaller than the pointwise Hölder exponent. It is stable under differentiation and integration, unlike the pointwise exponent [Ve01]. Fraclab [Ve00] provides various methods for the estimation of these two exponents. In the following sections where the Hölder regularity is used for image denoising, the local Hölder exponent α_l and the pointwise Hölder exponent α_p are assumed to be the same, ie., $\alpha_l = \alpha_p$. To illustrate Hölder exponent, consider the wavelet coefficients that behave like $g_j(k) \approx 2^{j(\alpha+1/2)}$ as j tends to $-\infty$. Here, α represents the Hölder exponent and quantifies local variation, $g_j(k)$ are the detail coefficients obtained from the wavelet transform at resolution j . At a particular point t_0 , the Hölder exponent $\alpha(t_0)$ behaves like $(\delta t)^{\alpha(t_0)}$ as $\delta t \rightarrow 0$ in an interval $[t_0, t_0 + \delta t]$ of length δt . Informally, signals with $\alpha(t_0) = H$ (H is the Hurst parameter) at all instants t_0 are called monofractals while signals with nonconstant Hölder exponent $\alpha(t_0)$ are termed multifractals.

5.3 Image Denoising Using Multifractal Analysis

A lot of research is going on in the use of multifractal analysis in image denoising. Denoising is done based on factors such as spectrum shift value and Hölder exponent shift of the input signal. Two methods for image denoising using multifractal analysis are considered here. They are multifractal regularization [Ve01] and multifractal pumping [Ve00]. Each of these methods is discussed in detail in Sections 5.3.1 and 5.3.2, respectively.

Following the notations of previous chapters, the original signal is represented as $s(x,y)$, noise as $n(x,y)$, observed signal as $w(x,y)$, and (x,y) as the pixel location. The Hölder regularity of $w(x,y)$ represented as α_w will be less than the Hölder regularity of

$s(x,y)$ α_s . This is because $s(x,y)$ denotes the image without noise. The goal therefore, is to increase α_w . If the Hölder regularity of $s(x,y)$ be known, it can be used as a target. But in most practical cases, it is unknown. In such circumstances, it is estimated from $w(x,y)$. Let $z(x,y)$ be the estimate of the signal after regularization that has a regularity α_z which is close to α_s . Assuming that the regularity of the original signal without noise is unknown, we choose a positive parameter δ such that

$$\alpha_z = \alpha_w + \delta , \tag{5.1}$$

where α_z is the Hölder regularity of the estimated signal. The next step would be to estimate the local Hölder exponent of a signal from discrete observations. A wavelet-based procedure discussed in Chapter 4, Section 4.1 is used for estimating and controlling the Hölder exponent. It is to be noted that regularity is an abstraction and is valid only asymptotically. So the true value of Holder regularity is not the point of interest but only a resultant value, that is greater than that of the input signal is of interest [Ve01].

5.3.1 Multifractal Regularization

Multifractal regularization is a process by which the Hölder regularity of the input image is increased by the use of a wavelet-based approach. As mentioned above, according to the functional analysis point of view, no assumptions about the noise structure are made. A regularized version of the observed data is obtained that fulfills some constraints. These constraints [Ve01] are as follows.

1. The signal estimate, $z(x,y)$ is close to the observed signal $w(x,y)$ in the L^2 sense which means that $\|z(x, y) - w(x, y)\|_{L^2}$ is minimum.
2. The local Holder function of $z(x,y)$ is prescribed.

Applying the wavelet-based procedure, let $\{\psi_{j,k}\}$ be the orthonormal wavelet basis where j denotes scale and k the position. It is assumed that $\{\psi_{j,k}\}$ and has sufficiently many vanishing moments. The wavelet coefficients behave like $g_j(k) \approx 2^{j(\alpha+1/2)}$ as j tends to $-\infty$. Here, α represents the Hölder exponent and quantifies local variation. It is obtained from the regression of the logarithm of the wavelet coefficients of $z(x,y)$ above any point i

with respect to scale, which is $-(\alpha(i) + \frac{1}{2})$ [Lu01]. There are two points noteworthy during this estimation.

1. The estimation is obtained through a regression on a finite number of scales, defined as a subset of the scales available on the discrete data. In particular, it is possible to express the Hölder function of the noisy signal $w(x, y) = s(x, y) + \text{Gaussian white noise}$ as a function of α_s , and thus estimate conversely α_s from α_w [Lu01].
2. The use of (orthonormal) wavelets allows performing the reconstruction in a simple way. This reconstruction or the inverse discrete wavelet transform is discussed in Section 4.3.

A direct implementation of this can be done using the Fraclab [Ve00] tool in Matlab 6.1 [Ma01]. Using this toolbox, an image “cameraman.tif” which is corrupted with Gaussian noise with the help of **imnoise()** [Appendix] function is loaded into the Fraclab workspace with the help of “scan workspace” option present in the interface. The loaded image can be viewed by clicking on the “view” button. Once the image is loaded, it can be denoised by selecting the “multifractal regularization” option in the denoising menu present in the interface. This option opens a window where the Hölder exponent shift value can be specified over a range of [-5, +5]. The regularized image obtained when the corrupted Image 5.1 is subjected to multifractal regularization is shown in Image 5.2. The Hölder exponent shift specified is 2.5. Typically the shift value is around 2 [Ve00].

5.3.2 Multifractal Pumping

Multifractal pumping is a procedure by which the Hölder exponent of a received signal is increased so that the regularity of the signal is improved and the signal is close to the desired signal. In this method, initially, a wavelet transform is applied and the image is decomposed into its wavelet coefficients. The wavelet coefficients obtained from the wavelet transform at scale j are multiplied by $2^{-\delta j}$. (Here, δ refers to the user-



Image 5.1: Image corrupted with Gaussian noise



Image 5.2: Image after multifractal regularization

defined parameter in Equation (5.1).) This results in increasing the Hölder exponent by an amount δ . This roughly amounts to performing a fractional integration of order δ . Also, the local Hölder exponent is related to a notion of local fractional derivative.

Using Fraclab [Ve00], multifractal pumping has been experimented on a Synthetic Aperture Radar (SAR) image. This image namely “sar.tif” is loaded into the Fraclab workspace by using the “**load**” option, provided in the interface. When this image is viewed using the “**view**” option present in the Fraclab interface, it looks as shown in Image 5.3. It can be noticed from the input image that it is very irregular and no details are visible. This image is subjected to multifractal pumping which is selected from the “multifractal pumping” option from the denoising menu present in the interface. The value of δ can be specified here. It is referred to as the spectrum shift value and varies over the range -5 to $+5$. A value of 1.5 is specified for this image, and multifractal pumping is done on Image 5.3 by hitting the “**compute**” button. The resultant image is shown in Image 5.4. It can be observed from Image 5.4 that the inverted ‘V’ shaped river which is cannot be seen in Image 5.3 can be seen in the output image 5.4.

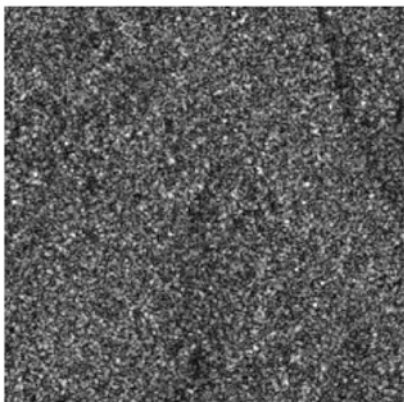


Image 5.3: Input image to multifractal pumping

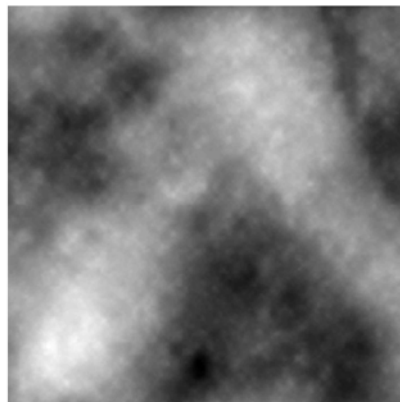


Image 5.4: Image after multifractal pumping (spectrum shift value of 1.5)

With the Fraclab toolbox, we get an opportunity to observe the effect of decreasing the regularity of an image by specifying negative values for the spectrum shift value. This effect can be observed in Image 5.5 where the spectrum shift value specified is -2.0 . The input image is degraded further has become more irregular. On the other hand, if a large positive value for the spectrum shift value is given, the input image gets too blurred. This effect can be seen in Image 5.6. The details of the image cannot be read from the output. Typical value for spectrum shift is around 0.5 [Ve01].

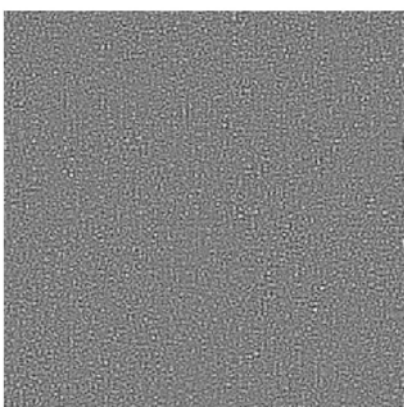


Image 5.5: Output of multifractal pumping (spectral shift value -2.0)

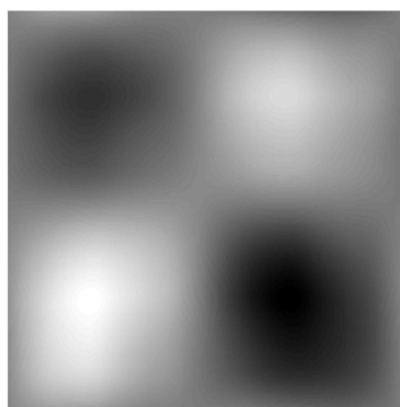


Image 5.6: Output of multifractal pumping (spectral shift value $+4.0$)

5.4 Summary

This chapter considered one of the latest techniques in denoising of images, i.e., multifractal analysis. This method is very helpful for the removal of noise from an image that has a complex and irregular nature. This method finds applications in denoising of Synthetic Aperture Radar (SAR) images. It is also observed that multifractal pumping is more effective than multifractal regularization. The next chapter provides a comparative study of all the techniques discussed so far. A quantitative result is given by the computation of signal to noise ratio of the output image.

Chapter 6

Results and Conclusion

This chapter deals with the comparison of the denoising techniques, namely, linear and non-linear filtering, wavelet based denoising, and denoising by multifractal analysis. The signal to noise ratio of the output image is calculated which acts as a quantitative standard for comparison.

6.1 Results

The selection of the denoising technique is application dependent. So, it is necessary to learn and compare denoising techniques to select the technique that is apt for the application in which we are interested.

By far there is no criterion of image quality evaluation that can be accepted generally by all. A technique to calculate the signal to noise ratio in images has been proposed which can be used with some approximation [St01]. This method assumes that the discontinuities in an image are only due to noise. For this reason, all the experiments are done on an image with very little variation in intensity. A test image where all pixel values having a magnitude of 100 is created and noise is added to it with the **imnoise()** function. Denoising is carried out following the techniques discussed in Chapter 3 through 5. Signal to Noise Ratio (SNR) for each of these outputs is computed.

The SNR is defined as

$$\text{SNR} = 20 \log_{10} \left(\frac{a_{\max} - a_{\min}}{s_n} \right) \quad (6.1)$$

The variable a_{\max} refers to the pixel value with maximum intensity while a_{\min} refers to the pixel value with minimum intensity in the image of interest. Variable s_n is the standard deviation of the noise defined as

$$s_n = \sqrt{\frac{1}{\Lambda-1} \sum_{(m,n) \in R} (a[m,n] - m_a)^2},$$

where m_a is the sample mean of the pixel brightness in the region R which is the entire image in all the experiments done in this thesis. The parameter Λ refers to the number of pixels in the region R and $a[m,n]$ is the pixel value. Sample mean is computed as

$$m_a = \frac{1}{\Lambda} \sum_{(m,n) \in R} a[m,n].$$

Tables 6.1 and 6.2 shows the SNR of the input and output images for the filtering approach and wavelet transform approach, respectively.

Table 6.1: SNR values for filtering approach

Method	SNR of input image	SNR of output image	Noise type and variance, σ
Mean filter	18.88	27.43	Salt and pepper, 0.05
Mean filter	13.39	21.24	Gaussian, 0.05
LMS adaptive filter	18.88	28.01	Salt and pepper, 0.05
LMS adaptive filter	13.39	22.40	Gaussian, 0.05
Median filter	18.88	47.97	Salt and pepper, 0.05
Median filter	13.39	22.79	Gaussian, 0.05

Table 6.2: SNR values for the wavelet transform approach

Method	SNR of input image	SNR of output image	Noise type and variance, σ
VisuShrink	13.39	31.17	Gaussian, 0.05
VisuShrink	18.88	19.01	Salt and pepper, 0.05
SureShrink	13.39	36.46	Gaussian, 0.05
SureShrink	18.88	40.67	Salt and pepper, 0.05
BayesShrink	13.39	30.98	Gaussian, 0.05
BayesShrink	18.88	18.92	Salt and pepper, 0.05

From Tables 6.1 and 6.2, it can be seen that the mathematical results obtained from the SNR computation and the experimental results shown in the image outputs in Chapters 3 through 5 match closely. For the multifractal denoising, the SNR computation is not compatible because, the brightness of the output image has been decreased.

6.2 Conclusions and Future Work

From the experimental and mathematical results it can be concluded that for salt and pepper noise, the median filter is optimal compared to mean filter and LMS adaptive filter. It produces the maximum SNR for the output image compared to the linear filters considered. The LMS adaptive filter proves to be better than the mean filter but has more time complexity. From the output images shown in Chapter 3, the image obtained from the median filter has no noise present in it and is close to the high quality image. The sharpness of the image is retained unlike in the case of linear filtering. In the case where an image is corrupted with Gaussian noise, the wavelet shrinkage denoising has proved to be nearly optimal. SureShrink produces the best SNR compared to VisuShrink and BayesShrink. However, the output from BayesShrink method is much closer to the high quality image and there is no blurring in the output image unlike the other two methods. VisuShrink cannot denoise multiplicative noise unlike BayesShrink. It has been observed that BayesShrink is not effective for noise variance higher than 0.05. Denoising salt and pepper noise using VisuShrink and BayesShrink has proved to be inefficient. When the noise characteristics of the image are unknown, denoising by multifractal analysis has proved to be the best method. It does a good job in denoising images that are highly irregular and are corrupted with noise that has a complex nature. In the two methods considered, namely multifractal regularization and multifractal pumping, the second method produces visually high quality images.

Since selection of the right denoising procedure plays a major role, it is important to experiment and compare the methods. As future research, we would like to work further on the comparison of the denoising techniques. If the features of the denoised signal are fed into a neural network pattern recognizer, then the rate of successful classification should determine the ultimate measure by which to compare various denoising procedures [Ta99]. Besides, the complexity of the algorithms can be measured

according to the CPU computing time flops. This can produce a time complexity standard for each algorithm. These two points would be considered as an extension to the present work done.

Bibliography

- [An01] Anestis Antoniadis, Jeremie Bigot, “Wavelet Estimators in Nonparametric Regression: A Comparative Simulation Study,” *Journal of Statistical Software*, Vol 6, I 06, 2001.
- [Ca79] Castleman Kenneth R, *Digital Image Processing*, Prentice Hall, New Jersey, 1979.
- [Ch00] S. Grace Chang, Bin Yu and Martin Vetterli, “Adaptive Wavelet Thresholding for Image Denoising and Compression,” *IEEE Trans. Image Processing*, Vol 9, No. 9, Sept 2000, pg 1532-1546.
- [Do92] David L. Donoho, “De-noising by soft-thresholding,” <http://citeseer.nj.nec.com/cache/papers/cs/2831/http:zSzzSzwww-stat.stanford.edu/SzreportszSzdonohozSzdenoiserelease3.pdf/donoho94denoising.pdf>, Dept of Statistics, Stanford University, 1992.
- [Do94] David L. Donoho and Iain M. Johnstone, “Adapting to Unknown Smoothness via Wavelet Shrinkage,” *Journal of American Statistical Association*, 90(432):1200-1224, December 1995.
- [Fr99] $1/f$ noise, “Brownian Noise,” <http://classes.yale.edu/99-00/math190a/OneOverF.html>, 1999.
- [Ga96] B.M.Gammel, “Multifractals,” <http://www1.physik.tu-muenchen.de/~gammel/matpack/html/Mathematics/Multifractals.html>, September 1996.
- [Ga99] Langis Gagnon, “Wavelet Filtering of Speckle Noise-Some Numerical Results,” *Proceedings of the Conference Vision Interface 1999*, Trois-Riveres.
- [Gr95] Amara Graps, “An Introduction to Wavelets,” *IEEE Computational Science and Engineering*, summer 1995, Vol 2, No. 2.
- [Ha01] David Harte, *Multifractals Theory and applications*, Chapman and Hall/CRC, New York, 2001.
- [Im01] Matlab 6.1, “Image Processing Toolbox,” <http://www.mathworks.com/access/helpdesk/help/toolbox/images/images.shtml>

- [La91] Reginald L. Lagendijk, Jan Biemond, *Iterative Identification and Restoration of Images*, Kulwer Academic, Boston, 1991.
- [Li93] J.N. Lin, X. Nie, and R. Unbehauen, "Two-Dimensional LMS Adaptive Filter Incorporating a Local-Mean Estimator for Image Processing," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol 40, No.7 July 1993, pg. 417-428.
- [Lu01] Jacques Lèvy Vèhel and Evelyne Lutton, "Evolutionary signal enhancement based on Hölder regularity analysis," *Project Fractales-INRIA*, 2001.
- [Ma68] Mandelbrot, B., and Wallis, J., "Noah, Joseph and operational hydrology," *Water Resources Research* 4, 909-918, 1968.
- [Ma89] Mallat S.G, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattn Anal. Mach. Intell.*, 11, 674-693, 1989.
- [Ma94] Peter R. Massopust, *Fractal Functions, Fractal Surfaces, and Wavelets*, Academic press, San Diego, 1994.
- [Ma01] Matlab6.1, "Matlab," <http://www.mathworks.com/>, May 2001.
- [Ni86] Wayne Niblack, *An Introduction to Digital Image Processing*, Prentice Hall, New Jersey, 1986.
- [Ri98] Rudolf H. Riedi, "Multifractals and Wavelets: A potential tool in Geophysics," *SEG Expanded Abstracts*, Rice University, Houston, Texas, 1998.
- [St01] Image Processing Fundamentals-Statistics, "Signal to Noise ratio," <http://www.ph.tn.tudelft.nl/courses/FIP/noframes/fip-Statisti.html>, 2001.
- [Ta99] Carl Taswell, "The What, How, and Why of Wavelet Shrinkage Denoising," <http://www.toolsmiths.com/docs/CT199809.pdf>, *Technical Report*, Stanford, CA, 1999.
- [Ti92] Tim Edwards, "Discrete Wavelet Transforms: Theory and Implementation," *Discrete Wavelet Transforms*, Stanford University, Draft #2, June 4, 1992
- [Tk99] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, San Diego, CA, 1999.

- [Um98] Scott E Umbaugh, *Computer Vision and Image Processing*, Prentice Hall PTR, New Jersey, 1998.
- [Ve97] Jacques Lèvy Vèhel, Bertrand Guiheneuf, “Multifractal image denoising,” *Project Fractales-INRIA*, April, 1997.
- [Ve01] Jacques Lèvy Vèhel, “Signal Enhancement Based on Hölder Regularity Analysis,” *Project Fractales-INRIA*, 2001.
- [Ve00] Jacques Lévy Véhel, “Fraclab,” www-rocq.inria.fr/fractales/, May 2000
- [Vi99] B. Vidakovic, *Statistical modeling by wavelets*, John Wiley and Sons, Inc. New York, 1999.
- [Wa01] Matlab 6.1, “Wavelet tool box,” <http://www.mathworks.com/access/helpdesk/help/toolbox/wavelet/wavelet.shtml>

Appendix: Matlab Functions

Matlab function	Description	Usage	Location
ddencmp()	Returns the Default threshold values for denoising and compression	[THR,SORH,KEEPAPP]= ddencmp (IN1,'wv',X)	Wavelet toolbox
double()	Convert data to double precision	B = double (A)	Image processing toolbox
dwt2()	Performs single level discrete 2-D wavelet transform	[cA,cH,cV,cD] = dwt2 (X,'wname')	Wavelet toolbox
eig()	Returns matrix eigen values and eigen vectors	lambda = eig (A)	Symbolic Math toolbox
idwt2()	Performs single level inverse discrete 2-D wavelet transform	X = idwt2 (cA,cH,cV,cD,'wname')	Wavelet toolbox
imnoise()	Adds noise to an image	J = imnoise (I,type)	Image processing toolbox
imread()	Read image from graphics files	[A] = imread (filename)	Image processing toolbox
imshow()	Displays an image	imshow (A)	Image processing toolbox
imwrite()	Writes image to graphics file	imwrite (A,filename)	Image processing toolbox
medfilt2()	Performs two-dimensional median filtering	B = medfilt2 (A,[m n])	Image processing toolbox
uint8()	Converts data to unsigned 8-	B = uint8 (A)	Image processing

	bit integers		toolbox
wavedec2()	Performs multilevel 2-D wavelet decomposition	$[C,S] = \mathbf{wavedec2}(X,N,'wname')$	Wavelet toolbox
wdcbm2()	Returns 2-D threshold value based on SureShrink	$[THR,NKEEP] = \mathbf{wdcbm2}(C,S,ALPHA)$	Wavelet toolbox
wdencmp()	Performs De-noising or compression using wavelets	$[XD] = \mathbf{wdencmp}('gbl',X,'wname',N,THR,SORH,KEEPAPP)$ $[XC] = \mathbf{wdencmp}('lvd',C,L,'wname',N,THR,SORH)$	Wavelet toolbox
wthresh()	Performs hard or soft thresholding	$Y = \mathbf{wthresh}(X,SORH,T)$	Wavelet toolbox

Vita

Sarita Dangeti was born in Kakinada, India, on August 22nd, 1978. She received the degree of Bachelor of Engineering in Electronics and Communications Engineering from Andhra University College of Engineering, Visakhapatnam, India, in May 2000. This was where she felt the need to pursue higher studies in her chosen field and decided to do a graduate program in the Department Electrical and Computer Engineering at Louisiana State University in Fall 2000 semester. She is a Graduate Assistant with School of the Coast and Environment at Louisiana State University (LSU). She expects to receive the degree of Master of Science in Electrical Engineering in May 2003.