12-15-2006

Development of an Image Noise Estimation Method and a Sub-Imaging Based Wiener Method

Eric F. Smith
University of New Orleans

Development of an Image Noise Estimation Method and a Sub-Imaging Based Wiener Method

A Dissertation

Submitted to the Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Engineering and Applied Science

by

Eric F. Smith

A.S., Delgado Community College, 1996
B.S., University of New Orleans, 1988
M.S., University of New Orleans, 2002

December 2006

ACKNOWLEDGMENT

I would like to express sincere gratitude to my major advisor, Dr. George Ioup, for his guidance, which made it possible for me to complete my dissertation in the Department of Physics. Discussions with him were always very informative and I gained considerable knowledge and insights into the methods and problems of image restoration. I wish to thank the members of my dissertation committee, Dr. George Ioup (chair), Dr. Juliette Ioup, Dr. Ashok Puri, Dr. Paul Herrington, Dr. Curtis Outlaw and Dr. Weilin Houe for their pertinent discussions and comments and helpful suggestions. Finally, I would like to thank my parents for their love and guidance and support through my formative years.

TABLE OF CONTENTS

LIST OF FIGURES v

LIST OF TABLES vii

LIST OF ABBREVIATIONS vii

ABSTRACT x

1 INTRODUCTION 1

1.1 Preliminaries 3

1.2 Image Domain Direct Method 10

1.3 Fourier Method 11

1.4 Wiener Method 12

1.5 Van Cittert Method 14

1.6 Richardson Lucy Method 16

1.7 CLEAN Method 18

1.8 Comparison of Methods 20

2 NOISE SIGMA ESTIMATION METHOD 33

2.1 Wavelet Noise Reduction Method 34

2.2 Morrison Noise Reduction Method 40

2.3 Comparison of MNRM and WNRM 42

2.4 Noise Sigma Estimation Method 46

3 SUB-IMAGING METHOD 54

3.1 Sub-Imaging 55

3.2 Sub-Imaging Wiener Method 59

3.3 Comparison of SIWM and the Wiener Method 60

3.4 Comparison of SIWM and MatLab Wiener2 68

4 SUMMERY AND DISCUSSION 71

BIBLIOGRAPHY 74

COMPUTER CODE 76

VITA 96

LIST OF FIGURES

Figure 1.1. Image Contamination Model 1

Figure 1.2. Narrow & Wide PSF 3

Figure 1.3. B&W and Color Pixel Sample 4

Figure 1.4. Example of Image Contamination 7

Figure 1.5. Dr. Norbert Wiener 12

Figure 1.6. Flow Chart of Wiener Method 13

Figure 1.7. Flow Chart of VCM 14

Figure 1.8. Flow Chart of RLM 16

Figure 1.9. Flow Chart of CLEAN Method 18

Figure 1.10. Images used in this Dissertation 20

Figure 1.11. 6 Restoration Methods: Image 2 with PSF(11,0.33,0.33) & Noise(30) 22

Figure 1.12. 6 Restoration Methods: Image 2 with PSF(11,1.75,1.75) & Noise(0) 23

Figure 1.13. 6 Restoration Methods: Image 2 with PSF(11,1.,1.) & Noise(30) 24

Figure 1.14. 6 Restoration Methods: Image 5 with PSF(11,0.33,0.33) & Noise(30) 25

Figure 1.15. 6 Restoration Methods: Image 5 with PSF(11,1.75,1.75) & Noise(0) 26

Figure 1.16. 6 Restoration Methods: Image 5 with PSF(11,1.,1.) & Noise(30).....27

Figure 1.17. 6 Restoration Methods: Image 10 with PSF(11,1.,1.) & Noise(30) 28

Figure 1.18. 6 Restoration Methods: Image 1 with PSF(11,1.,1.) & Noise(30) 29

Figure 2.1. Flow Chart of Daubechies Coefficients 36

Figure 2.2. Flow Chart of WNRM.....37

Figure 2.3. WNRM and medium PSF with noise 38

Figure 2.4. Flow Chart of Morrison Noise Reduction Method (MNRM) 41

Figure 2.5. MNRM and WNRM, narrow width PSF: Image 3.....42

Figure 2.6. MNRM and WNRM, medium width PSF: Image 8 43

Figure 2.7. MNRM and WNRM, wide width PSF: Image 10 44

Figure 2.8. Flow Chart of Sigma Estimation Method (SIGEST) 46

Figure 2.9. Graph of Sigma actual vs Sigma estimated 48

Figure 2.10. MNRM with narrow PSF: Images 1 and 8 53

Figure 3.1. Flow Chart of CALC_ASI 55

Figure 3.2. Example of SIM applied to Image 2 (all sub-images).....56

Figure 3.3. Example of SIM applied to Image 2 (one sub-images) 57

Figure 3.4. Example of SIM applied to Image 1 58

Figure 3.5. Flow Chart of SIWM 59

Figure 3.6. SIM Applied to Image 9 61

Figure 3.7. SIM Applied to Image 14 62

Figure 3.8. SIM Applied to Image 1 63

Figure 3.9. SIWM vs Wiener 2: Image 13 68

Figure 3.10. SIWM vs Wiener 2: Image 9 69

Figure 3.11. SIWM vs Wiener 2: Image 9 69

Figure 3.12. Flow Chart of Complete SIWM 73

LIST OF TABLES

Table 1.1. Simulation results for PSF(11,1.0,1.0) and Noise(30).....30

Table 2.1. Noise Estimation Results for WNRM 39

Table 2.2. Noise Estimation Results for narrow width PSF 49

Table 2.3. Noise Estimation Results for medium width PSF 50

Table 2.4. Noise Estimation Results for wide width PSF 51

Table 3.1. SIWM Results for Images: 1, 2, 4, 7, 9 to 14, narrow PSF 64

Table 3.2. SIWM Results for Images: 1, 2, 4, 7, 9 to 14, meduim PSF 65

Table 3.3. SIWM Results for Images: 1, 2, 4, 7, 9 to 14, wide PSF 66

LIST OF ABBREVIATIONS

h	Recorded image data
h2	Image data processed by two iterations of the Morrison Noise Reduction Method
hN	Image noise data calculated by, hN = h – h2
PSF	Point spread function
b	Blurring function (PSF), also impulse response
n	Noise data
f	Actual image data (uncontaminated), in optics called object data
Psdr	Peak signal to distortion ratio
Rmse	Root mean square error
mse	Mean square error
**	Convolution operator
DFT	Discrete Fourier transform
IDFT	Inverse discrete Fourier transform
$b \supset B$	Fourier transform of b into B
$B \subset b$	Inverse Fourier transform from B to b
σ_f	Standard deviation of image data f
σ_n	Standard deviation of noise data n
σ_1	First calculated estimate of standard deviation of noise data n
σ_e	Estimate of standard deviation of noise data n
IDDM	Image Domain Deconvolution Method

FTM	Fourier Transform Method
VCM	Van Cittert Method
RLM	Richardson Lucy Method
MNRM	Morrison Noise Reduction Method
SIGEST	Sigma Estimation Method
SIM	Sub-Imaging Method
SIWM	Sub-Image Wiener Method
2D	Two dimensional

ABSTRACT

This research consists of three parts. The first part is an investigation of several popular image restoration techniques. The techniques are used to restore 2-D image data, f(x,y), that has been blurred by a known point spread function (PSF), b(x,y) and corrupted by an unknown amount of noise, n(x,y). Several sample images are restored using all of the techniques. Of the methods investigated the one which produces the best restoration results was determined to be the Wiener deconvolution method. The determination of the best method is based on the quality of the restored image and the required restoration time.

The second part of this research involves the development of a noise standard deviation (σ_n) estimation method. The method determines an estimate, σ_e, of σ_n based on the Morrison Noise Reduction Method (MNRM) and is therefore an iterative method. The results of the noise, σ_n, estimating method (SIGEST) developed are rather good. The error between σ_n and σ_e when average across several images all contaminated with a medium width or greater PSF and various amounts of noise, is less than 10 percent. Knowledge of σ_n is important for the application of Wiener deconvolution. All noise in this research is assumed to be uncorrelated noise.

The third part of this research involves the development of the Sub-Imaging Method, *SIM*. In the third part of this research, the h2 and the hN of image data h is defined as follows:

h2 = Image data h processed by two iterations of the MNRM

hN = h – h2

SIM divides an image's hN into several rectangular parts, calculates σ_e of each part by the method described previously, calculates the average of the σ_e's and selects the part with a σ_e

which is closest to the average of all the σ_e's. The part with a σ_e closest to the average is defined to be the average sub-image (asi). The following assertions concerning SIM are investigated:

1. The asi of an image can be used in the place of the whole image to determine σ_e of σ_n and used to restore the whole image. Therefore, the noise in a piece of an image can represent the noise in the whole image (provided it is the asi of the image's hN).
2. SIM can be combined with the Wiener image restoration method to restore contaminated image data without the σ_n of the data initially being known.

In this research, image and numerical results are provided which validate the two claims about SIM. The wiener method and SIM are combined to develop the Sub-Image Wiener Method (SIWM). In this research, image and numerical results are provided to show that SIWM is an effective method of restoring blur and noise contaminated image data. Image and numerical data are provided comparing SIWM to the Matlab function Wiener2. The results show that SIWM is faster and yields better results than the Wiener2 method.

CHAPTER 1

INTRODUCTION

"One picture is worth more than ten thousand words", is a familiar proverb that refers to the idea that complex stories can be told with just a single still image. A single image may be more informative and or influential than a substantial amount of written information. An image may be defined as a two-dimensional function, f(x,y), where x and y are spatial coordinates, and the amplitude of f at any location (x,y) is called the intensity or gray level of the image at that point, [8]. Joseph Nicéphore Niépce using a sliding wooden box camera made by Charles and Vincent Chevalier in Paris made the first permanent image (photograph) in 1826 or 1827. In many applications (e.g., satellite imaging, medical imaging, astronomical imaging, poor-quality family portraits) the imaging system introduces a slight distortion. Often images are slightly blurred and image restoration aims at deblurring the image.

Digital image processing is the manipulation of images by computer. One of the first applications of digital images was in the newspaper industry when pictures were first sent by submarine cable between London and New York in the 1920s. Digital image techniques in image restoration and enhancement had their first fruitful application at the Jet Propulsion Laboratory of the California Institute of Technology, [2]. In the 1960s, as part of the program to land a man on the moon, it was decided to land unmanned spacecraft initially, which would televise back images of the moon's surface and test the soil for later manned landings. Unfortunately, the limitations on weight and power supply made it impossible to launch a "perfect" TV camera system on the unmanned craft. Thus, JPL measured the degradation

properties of the cameras before they were launched and then used computer processing to remove, as well as possible, the degradations from the received moon images, [2].

From the 1960s until the present, the field of image processing has grown significantly. In addition to applications in medicine and the space program, digital image processing techniques now are used in a broad range of applications. Computer procedures are used to enhance the contrast or code the intensity levels into color for easier interpretation of X-rays and other images used in industry, medicine, and the biological sciences. Geographers use the same or similar techniques to study pollution patterns from aerial and satellite imagery. Image enhancement and restoration procedures are used to process degraded images of unrecoverable objects or experimental results too expensive to duplicate. In archeology, image processing methods have successfully restored blurred pictures that were the only available records of rare artifacts lost or damaged after being photographed. In physics and related fields, computer techniques routinely enhance images of experiments in areas such as high-energy plasmas and electron microscopy. Similarly, successful applications of image processing concepts can be found in astronomy, biology, nuclear medicine, law enforcement, defense and industrial applications [2]. Today, there is almost no area of technical endeavor that is not impacted in some way by digital image processing.

1.1 Preliminaries

Deconvolution is a process of recovering information which has been altered (contaminated) from its true original form. The contamination for many images has two basic forms: blurring, which is described by the convolution, and noise addition. Thus contamination can be modeled as follows:

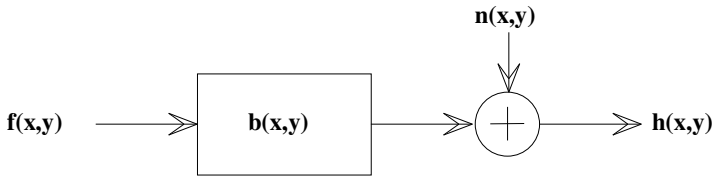


Figure 1.1

[1.1], $h(x, y) = b(x, y) ** f(x, y) + n(x, y).$

h = Recorded image data.

b = Blurring function or point spread function (PSF), also called the impulse response.

n = Noise

f = Actual input data (uncontaminated), referred to in optics as the object data.

The symbol ** denotes convolution between the PSF and the actual data.

Convolution describes the action of an observing instrument; it is generally a smoothing process. Mathematically, 2D convolution is defined (continuous and discrete variables respectively) as follows,

[1.2], $h(x, y) = b ** f = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} b(r,c) \cdot f((x-r),(y-c))\,dr\,dc.$

[1.3], $h(x, y) = b ** f = \sum_{r=-\infty}^{\infty}\sum_{c=-\infty}^{\infty} b(r,c) \cdot f((x-r),(y-c))\,\Delta r\,\Delta c.$

A numerical example of 2D convolution is shown in equation 1.4.

$$[1.4], \quad b**f = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} ** \begin{bmatrix} 2 & 3 & 1 \\ 5 & 4 & 7 \\ 3 & 1 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 7 & 7 & 2 \\ 11 & 31 & 30 & 18 \\ 18 & 39 & 45 & 40 \\ 9 & 15 & 22 & 24 \end{bmatrix} = h.$$

Noise is whatever distorts, deforms, prevents, interferes or changes the information being observed and or recorded other than that caused by a devices lens. In this paper all noise is considered to be normally distributed (gaussian) with a mean of zero. In my research I added noise to the various input image data by generating a random matrix using the Matlab function randn. The function *randn*, generates random normally distributed values with a standard deviation of 1, the values are stored in the noise matrix, *inoise*. The following equation is used to create the 2D noise matrix n(x,y) used in equation 1.1:

$$[1.5], \quad n(x, y) = inoise(x, y) \cdot [Max(f) - Min(f)] \cdot ns/100.$$

In equation 1.5, the value of *ns* is arbitrarily set from 0 to 100, so that the noise is set to have a σ_n which is ns% of the range of values of f, the original image data. To ensure the non-negativity of h calculated in equation 1.1 and that it not exceed the maximum pixel value, if h is less than zero then it's value is set equal to zero and if it is greater than 255 then it is set equal to 255.

Point spread function is a mathematical term for the impulse response of a 2D optical system. It is the impulse or point response of an optical system to a point input. A lens focuses a single point of light into a complex shape known as a point spread function (PSF). The shape of the PSF depends upon light wavelength, lens numerical aperture, and the optical aberration of the lens. In this paper all PSF's are 2D Gaussian types generated by the following equation:

[1.6], $$PSF(N_x, N_y, \sigma_x, \sigma_y) = \frac{1}{2 \cdot \pi \cdot \sigma_x^2 \cdot \sigma_y^2} \cdot \exp\left[\frac{-x^2}{2 \cdot \sigma_x^2} + \frac{-y^2}{2 \cdot \sigma_y^2}\right].$$

In equation 1.6, x runs from $-N_x/2$ to $N_x/2$ and y from $-N_y/2$ to $N_y/2$. The size of the matrix generated is N_x by N_y. The variables σ_x and σ_y are the standard deviation in the x and y direction respectively. For example, a 11 by 11 narrow PSF is given by, b = PSF(11,11,0.33,0.33), and is shown in figure 1.2a, a medium width PSF is given by b = PSF(11,11,1.0,1.0) shown in figure 1.2b and a 11 by 11 wide PSF is given by, b = PSF(11,11,1.75,1.75), shown is shown in figure 1.2c. All of the PSF's used in this paper are square in size, so hencforth all PSF's will be indicated by PSF(N_{xy}, σ_x, σ_y), with $N_{xy} = N_x = N_y$. Also from this point forward the amount of noise added to an image will be indicated by Noise(ns), the function which creates the noise matrix n(x,y) used in equation 1.1.

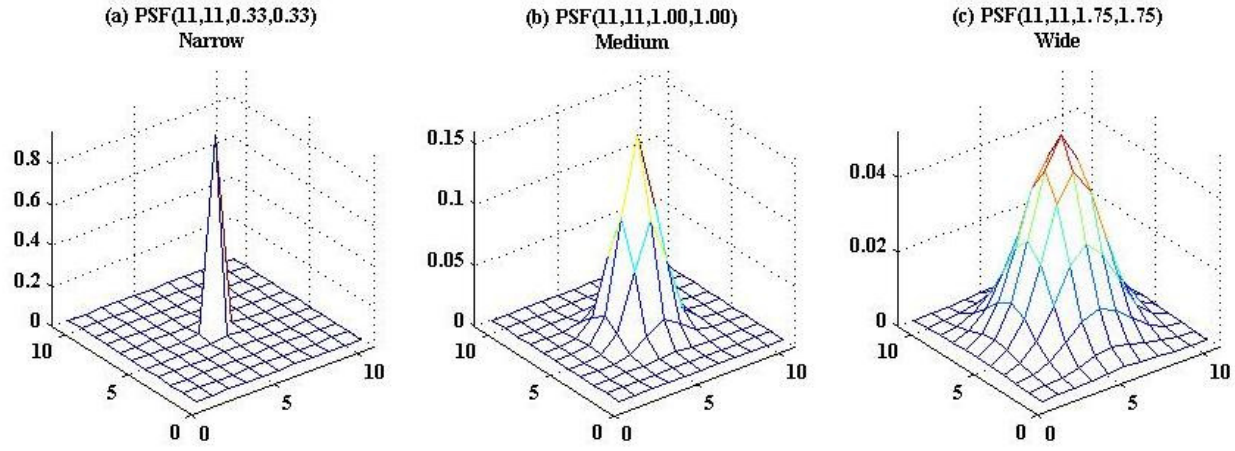


Figure 1.2

The purpose of deconvolution is to determine f by restoring the known h image data. For goal one of my research, the deconvolution techniques to be investigated will be coded in MatLab to calculate f based upon knowing h and b. The case where b is unknown is called *blind*

deconvolution, which will not be considered in this research, [14]. The MatLab code (m files) are shown in the *Code* section of this dissertation. Knowledge of the PSF b is possible because the blurring caused by a measuring instrument can be determined. The PSF is mostly a function of an instrument's lens for many optical systems.

The 2D image data contained in h and f specifies the light intensity measured from 0 through 255 for this research. An example of black and white image data stored in f1 is shown in equation 1.7. Each pixel in an image has an intensity value, [6]. The y and x index of the intensity corresponds to its position, e.g., f(2,4) = 220 means at position 2 units vertical and 4 units horizontal the light intensity is 220. The image generated by the f1 data is shown in figure 1.3a. An example of color image data f2 is shown in equation 1.8, each pixel is made of three separate intensities: red (f2(:,:,1)), green (f2(:,:,2)) and blue (f2(:,:,3)), [19]. The image generated by f2 is shown in figure 1.3b.

```
[1.7], f1 = [180,  0, 30,  0,220,  0, 30,  0,180
              0,  0,  0,220,120,220,  0,  0,  0
              0,  0,220,  0,120,  0,220,  0,  0
              0,220,  0,  0,120,  0,  0,220,  0
            220,120,120,120,255,120,120,120,220
              0,220,  0,  0,120,  0,  0,220,  0
              0,  0,220,  0,120,  0,220,  0,  0
              0,  0,  0,220,120,220,  0,  0,  0
            180,  0, 30,  0,220,  0, 30,  0,180]
```

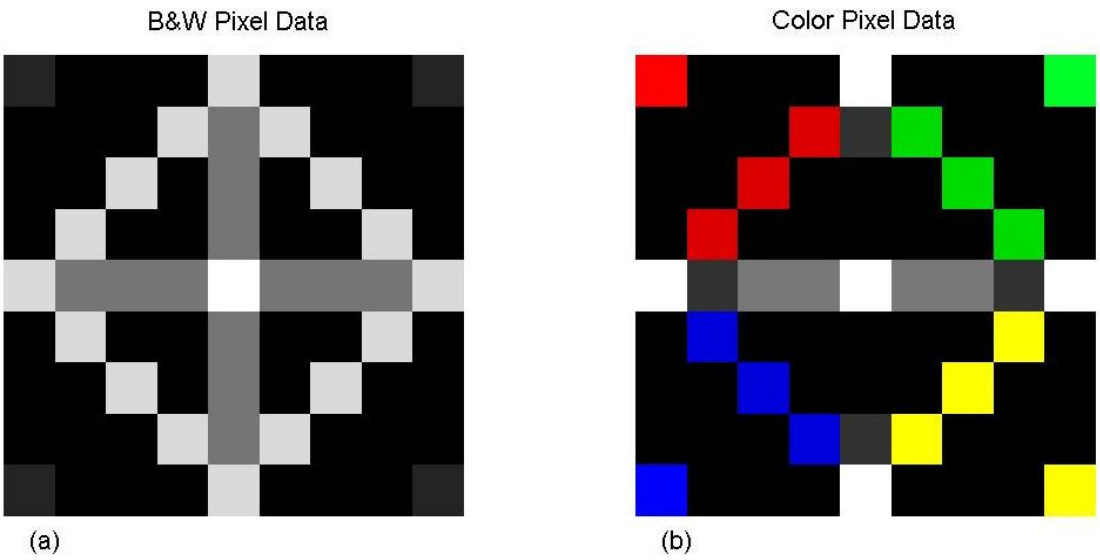


Figure 1.3

```
[1.8], f2(:,:,1) =[255,  0,  0,  0,255,  0,  0,  0,  0
                    0,  0,  0,220, 50,  0,  0,  0,  0
                    0,  0,220,  0,  0,  0,  0,  0,  0
                    0,220,  0,  0,  0,  0,  0,  0,  0
                  255, 50,120,120,255,120,120, 50,255
                    0,  0,  0,  0,  0,  0,  0,255,  0
                    0,  0,  0,  0,  0,  0,255,  0,  0
                    0,  0,  0,  0, 50,255,  0,  0,  0
                    0,  0,  0,  0,255,  0,  0,  0,255]

       f2(:,:,2) =[  0,  0,  0,  0,255,  0,  0,  0,255
                    0,  0,  0,  0, 50,220,  0,  0,  0
                    0,  0,  0,  0,  0,  0,220,  0,  0
                    0,  0,  0,  0,  0,  0,  0,220,  0
                  255, 50,120,120,255,120,120, 50,255
                    0,  0,  0,  0,  0,  0,  0,255,  0
                    0,  0,  0,  0,  0,  0,255,  0,  0
                    0,  0,  0,  0, 50,255,  0,  0,  0
                    0,  0,  0,  0,255,  0,  0,  0,255]

       f2(:,:,3) =[  0,  0,  0,  0,255,  0,  0,  0, 40
                    0,  0,  0,  0, 50,  0,  0,  0,  0
                    0,  0,  0,  0,  0,  0,  0,  0,  0
                    0,  0,  0,  0,  0,  0,  0,  0,  0
                  255, 50,120,120,255,120,120, 50,255
                    0,220,  0,  0,  0,  0,  0,  0,  0
                    0,  0,220,  0,  0,  0,  0,  0,  0
                    0,  0,  0,220, 50,  0,  0,  0,  0
                  255,  0,  0,  0,255,  0,  0,  0,  0]
```

Two criteria which are used to indicate how well the image g has been restored from the measured image data h are psdr and rmse. The peak signal to distortion (blur and noise) ratio (psdr), the root mean square error (rmse) and the mean square error (mse) are calculated as follows:

[1.9], $N_{rc} = N_r \cdot N_c$.

[1.10], $mse = \frac{1}{N_{rc}} \sum_{i}^{N_r} \sum_{j}^{N_c} \| f(i,j) - g(i,j) \|^2$.

[1.11], $rmse = \sqrt{mse}$.

[1.12], $psdr = 20 \cdot Log(Max(g)/rmse)$.

In equation 1.9, N_r and N_c are the number of pixels in each row and column of the image respectively, and N_{rc} is the total number of pixels in the image. Note that in equation 1.12 the psdr is given in dB. The smaller the value of rmse the closer g is to f by this measure. The greater the value of psdr the less noise and blurring are in the restored image g although the value of maximum g also enters the result.

The average pixel intensity of f is $\overline{p}$, the variance of image data f is σ_f^2, and the standard deviation of f is σ_f shown in equations 1.13, 1.14 and 1.15 respectively. The standard deviation of f is the statistical measure of spread or variability of the pixel intensities in f.

[1.13], $\overline{p} = \frac{1}{N_{rc}} \sum_{r=1}^{N_r} \sum_{c=1}^{N_c} f(r,c)$.

[1.14], $\sigma_f^2 = \frac{1}{(N_r - 1) \cdot (N_c - 1)} \sum_{r=1}^{N_r} \sum_{c=1}^{N_c} [f(r,c) - \overline{p}]^2$.

[1.15], $\sigma_f = \sqrt{\sigma_f^2}$.

The discrete Fourier transform (DFT) of f and the inverse discrete Fourier transform (IDFT) of F are defined in equations 1.16 and 1.17 respectively.

$$[1.16], \quad DFT(f) = F(j,k) = \sum_{j=0}^{N_r-1} \sum_{k=0}^{N_c-1} f(x,y) \cdot \exp[-2\pi i \cdot (x\,j / N_r - y\,k / N_c)].$$

$$[1.17], \quad IDFT(F) = f(x,y) = \frac{1}{N_{rc}} \sum_{j=0}^{N_r-1} \sum_{k=0}^{N_c-1} F(j,k) \cdot \exp[2\pi i \cdot (x\,j / N_r - y\,k / N_c)].$$

Figure 1.4 shows an example of blur, noise and blur plus noise applied to a sample image. Note how the addition of noise to an image produces a salt and pepper type appearance.

Figure 1.4

1.2 Image Domain Direct Method

The Image Domain Direct method, IDDM, is a direct method that solves equation 1.1, with zero noise, algebraically for f [13]. The original image data f, is solved for by executing equation 1.18 for all x from 1 to N_x and y from 1 to N_y.

$$[1.18], \quad f(x,y) = \frac{h(x,y) - \sum_{r=1}^{J}\sum_{c=1}^{K} b(r,c) f((x-r+1),(y-c+1))}{b(1,1)}.$$

Where,

$$[1.19], \quad J = Min(Rows(b),(x-1)).$$

$$[1.20], \quad K = Min(Cols(b),(y-1)).$$

Advantages of the method:

1. Restores f very well, with b known and zero noise.

Disadvantages of the method:

1. Is extremely sensitive to noise.
2. Is computationally expensive and slow.
3. If b(1,1) equals zero, PSF must be shifted to solve for f.

This method restores the image data exactly provided the image contains zero noise and has a known PSF. I have written MatLab code in file "iddm.m" which calculates f by this method. The code is shown in the code section of this dissertation and several images restored by this method are shown in the *Comparison of Methods* section.

1.3 Fourier Transform Method

The Fourier transform method is a direct method that solves equation 1.1, with zero noise, for f by first calculating the Fourier transform of $h \supset H$ and of $b \supset B$, [13]. By the convolution theorem,

[1.21], $b ** f = h \supset H = B \cdot F.$

with B · F, in equation 1.21, the point wise product of matrices B and F. Thus,

[1.22], $F = H / B,$ and

[1.23], $f = IDFT(F).$

Advantages of the method:

1. Restores f very well, with zero noise.
2. Is computationally fast (especially compared to the direct method).

Disadvantages of the method:

1. Is sensitive to noise.

This method restores the image data very well, provided the image contains zero noise and has a known PSF. The MatLab code for this method is in file "ftm.m". The code is shown in the code section of this dissertation and several images restored by this method are shown in the *Comparison of Methods* section.

1.4 Wiener Method

The Wiener method is an optimum linear noise and deconvolution filter based in the Fourier domain. It was invented by mathematician Norbert Wiener, [3]. Dr. Wiener first published the Wiener filter method in 1949. The method is a direct method that solves for f by using the power spectra of the noise, P_n, and image data, P_f, by the following equations, which also show the approximations used in this dissertation, [15] [5] [17]:

[1.24], $P_n = \frac{N \cdot N^*}{N_{rc}}.$

[1.25], $P_f = \frac{F \cdot F^*}{N_{rc}}.$

[1.26], $\hat{P}_n \approx \sigma_n^2$, for white noise.

[1.27], $\hat{P}_f \approx \frac{H \cdot H^*}{N_{rc}} - \sigma_n^2.$

[1.28], $G = \frac{B^*}{|B|^2 + P_n / P_f} \approx \frac{B^*}{|B|^2 + \hat{P}_n / \hat{P}_f}.$

Dr. Norbert Wiener 1894-1964
Figure 1.5

[1.29], $F = G \cdot H.$

[1.30], $f = IDFT(F).$

Advantages of the method:

1. Can restore f with noise present.

Disadvantages of the method:

1. Requires an initial knowledge of P_f or an approximation as by equation 1.27.

2. Requires knowledge of P_n or an approximation such as by equation 1.26, which then requires knowledge of σ_n or an estimate of it.

This method restores the image data with a known PSF and noise present. A flowchart of the method is shown in figure 1.5 and the MatLab code for this method is in file "dwiener.m". The code is shown in the code section of this dissertation and several images restored by this method are shown in the *Comparison of Methods* section. The Wiener method is known to be one of the best linear filters if P_f and the contaminating noise are known. It is derived to be the optimum filter in the least squares sense. Of course in reality neither the P_n nor P_f are known, but for white noise they can be approximated very well by equations 1.26 and 1.27.

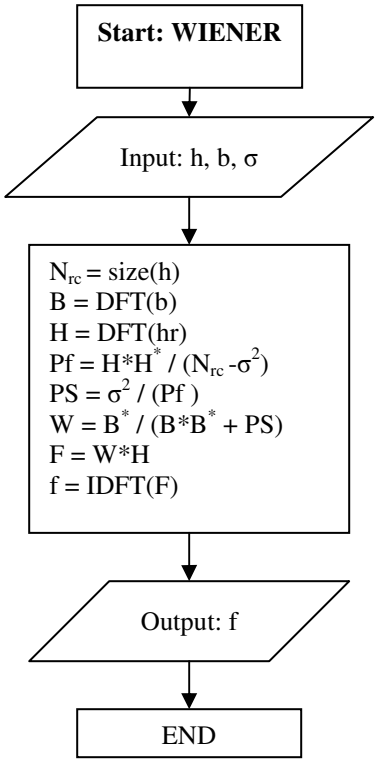


Figure 1.6

1.5 Van Cittert Method

The Van Cittert method is an iterative deconvolution method, [13]. It approximates f in equation 1.1, by setting an initial value f_0 = f then iterating equation 1.31 an arbitrary maximum set number of times (Nitr) or until the change in rmse is less than some set tolerance (Tol), [10].

[1.31], $f_{k+1} = f_k + [h - b * * f_k]$.

A flowchart of the method is shown in figure 1.7.

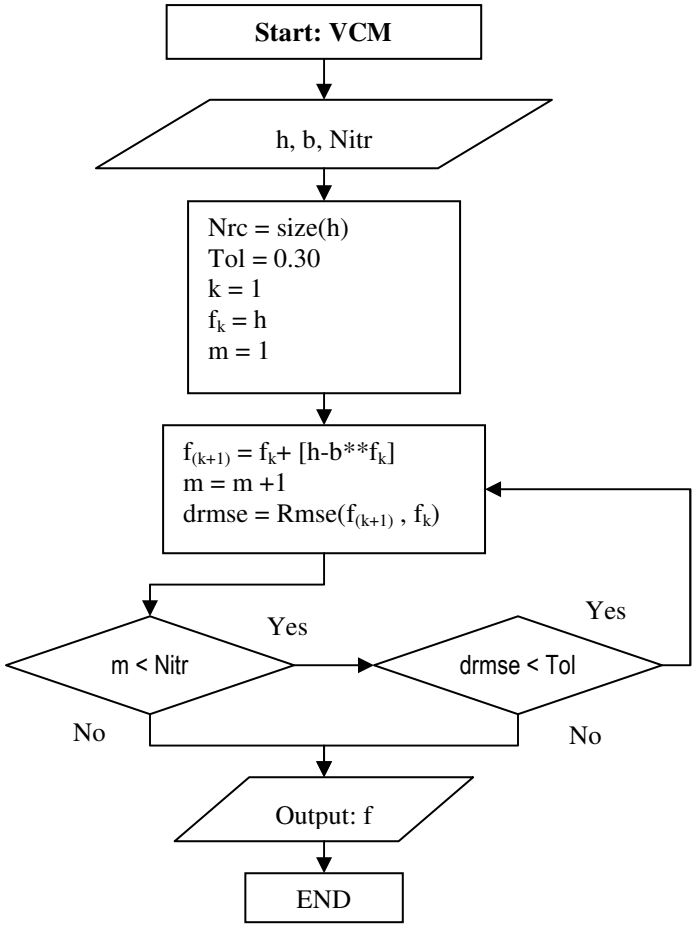


Figure 1.7

Advantages of the method:

1. Can restore f with noise present.

Disadvantages of the method:

1. Number of iterations is by trial or estimate and or the value of the stopping tolerance is arbitrary.

This method restores the image data with a known PSF and noise present. The MatLab code for this method is in file “vcm.m”. The code is shown in the code section of this dissertation and several images restored by this method are shown in the *Comparison of Methods* section.

1.6 Richardson Lucy Method

The Richardson Lucy method is an iterative deconvolution method, [13]. This method was developed from maximum likelihood theory and is modeled with Poisson statistics. It approximates f, with noise, by setting an initial value of f then iterating one of the following equations Nitr times, or until the change in rmse is less than some set tolerance (Tol):

[1.32], $f_{k+1} = f_k * [(-b) ** \{h / (b ** f_k)\}]$, Poisson Noise Model.

[1.33], $f_{k+1} = f_k * [\frac{b ** h}{(b ** f_k) ** b}]$, Gaussian Noise Model.

A flowchart of the method, for gaussian noise, is shown in figure 1.8.

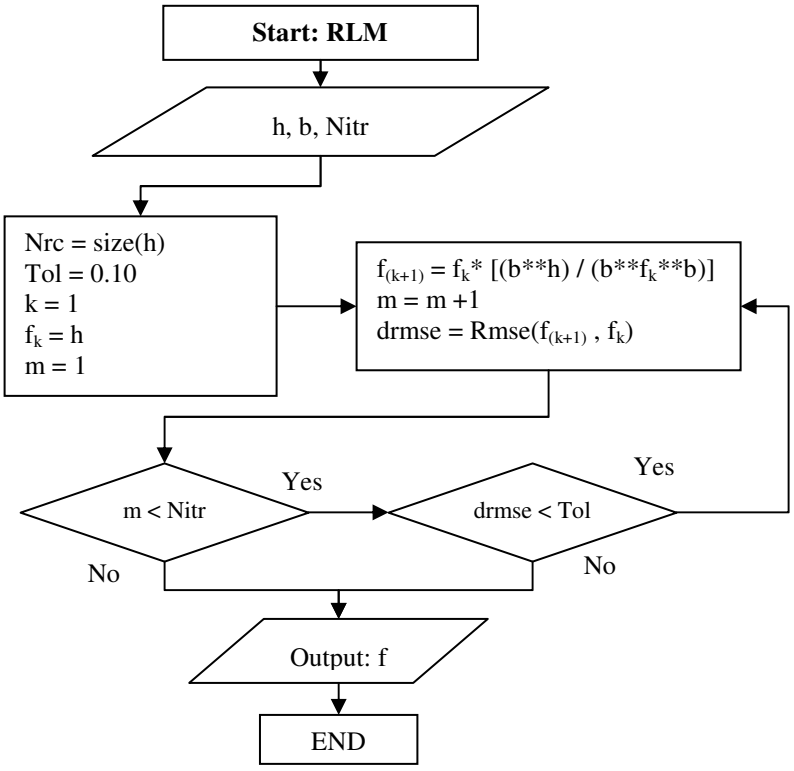


Figure 1.8

Advantages of the method:

1. Can restore f with noise present.

Disadvantages of the method:

1. Number of iterations is by trial or estimate and or the value of the stopping tolerance is arbitrary.

This method restores the image data with a known PSF and noise present. The MatLab code for this method is in file “rlmg.m”. The code is written only for the case of gaussian noise. It is shown in the code section of this dissertation and several images restored by this method are shown in the *Comparison of Methods* section. Note that the division indicated in equations 1.32 and 1.33 is element by element division.

1.7 CLEAN Method

The CLEAN method is an iterative deconvolution method. CLEAN was developed by J. D. Hogbom in 1974, [21]. The method is nonlinear and deconvolves b, referred to as the "dirty beam", from h, referred to as the "dirty image". The algorithm involves the creation of a residual image R, and a density image, D, [22] [23]. The CLEAN algorithm is described by the flowchart in figure 1.9.

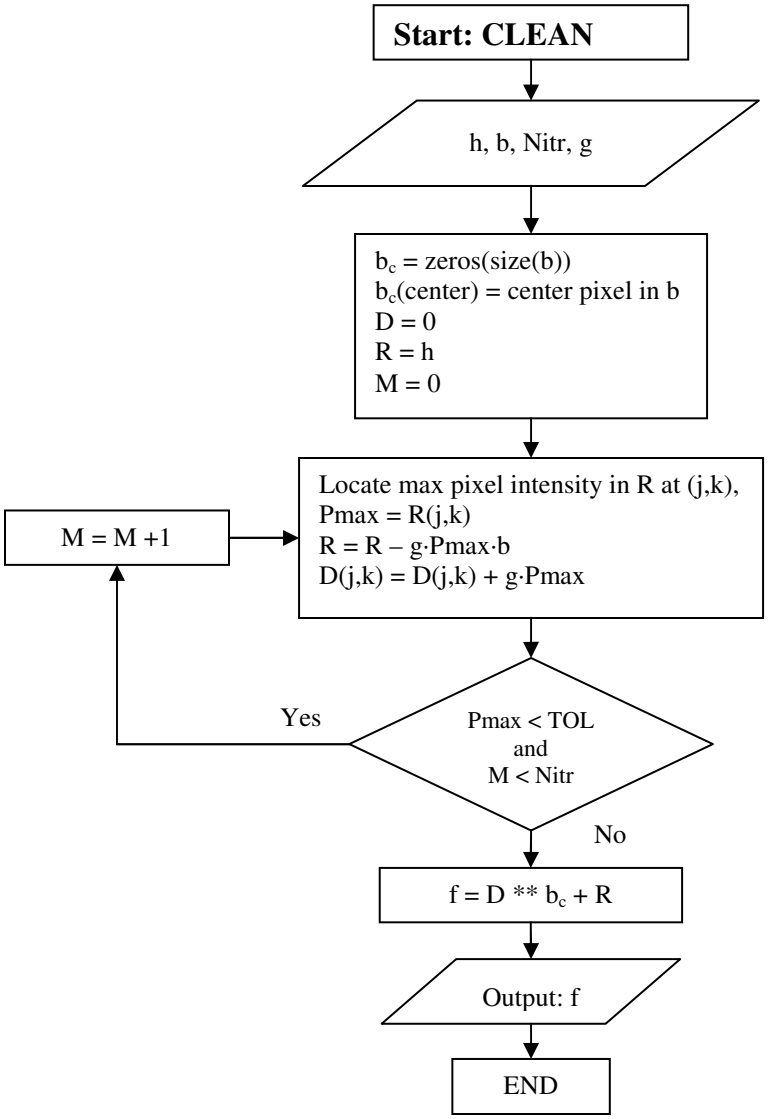


Figure 1.9

The method assumes that the image (sky) brightness is essentially an ensemble of point sources (the sky is dark, but full of stars).

Advantages of the method:

1. Can restore f with noise present.

Disadvantages of the method:

1. Requires a large number of iterations.
2. Number of iterations is by trial or estimate.

This method restores the image data with a known PSF and noise present. The MatLab code for this method is in file "clean.m". The code is shown in the code section of this dissertation and several images restored by this method are shown in the *Comparison of Methods* section.

1.8 Comparison of Methods

In this dissertation the images shown in figure 1.10 shall be used to evaluate the various image restoration methods and techniques.

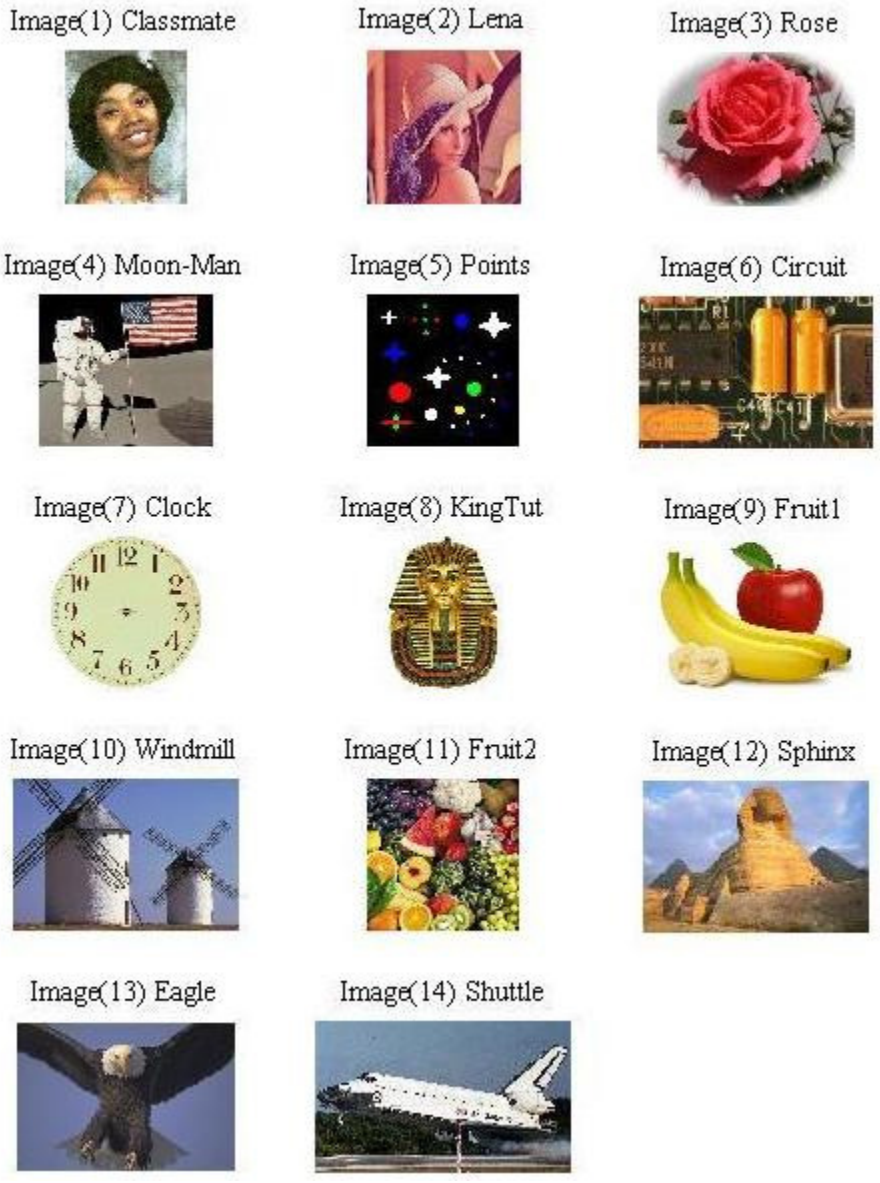
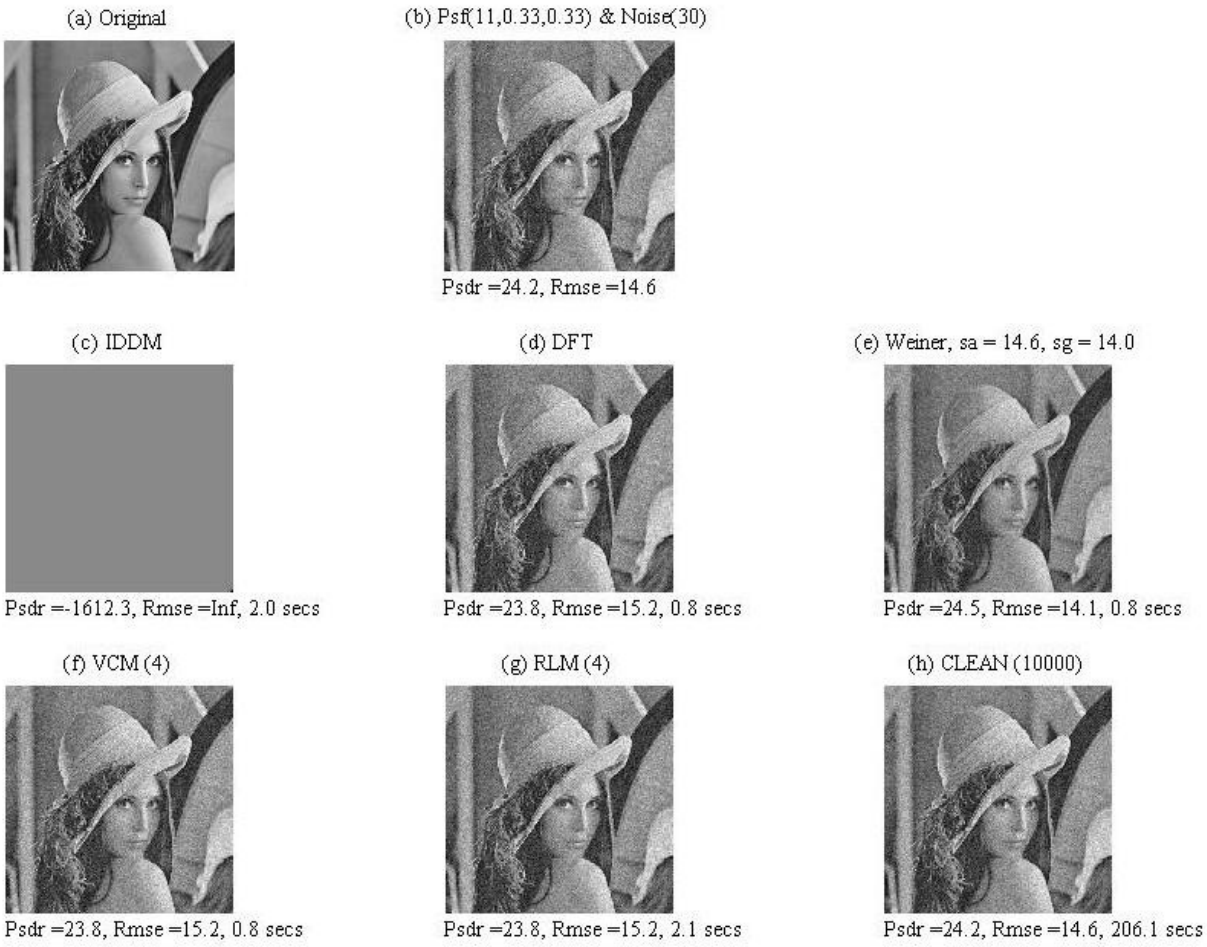


Figure 1.10

The main objective of this section is to determine the best image restoration method of the six described in sections 1.1 through 1.6. The method selected must be able to restore an image that has been contaminated with blur and noise. The test comparison of the six methods is done using several of the images shown in figure 1.10. The images are contaminated in three different ways:

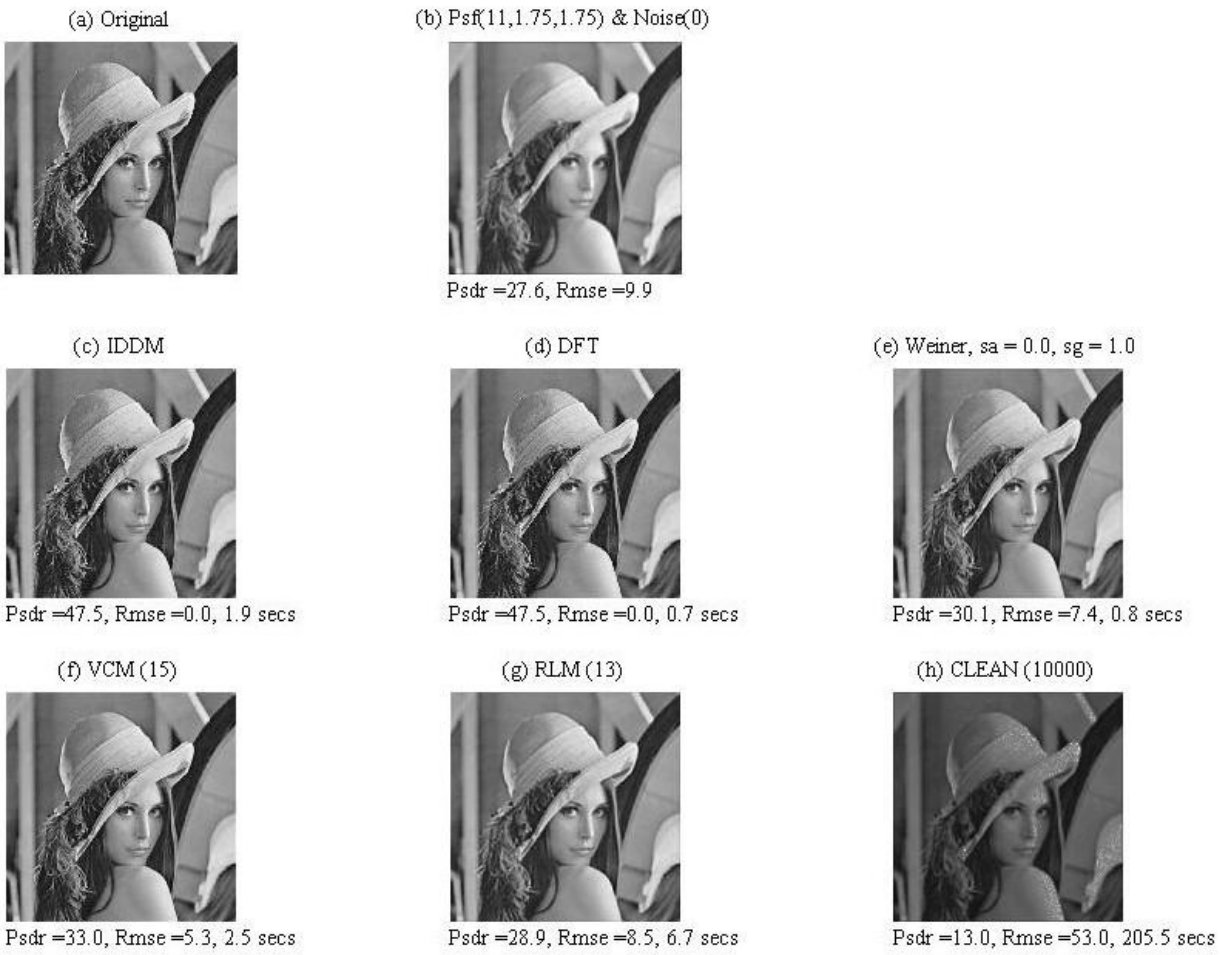
1. Blurred by a narrow width PSF(11,0.33,0.33) and Noise(30)
2. Blurred by a medium width PSF(11,1.75,1.75) and Noise(0)
3. Blurred by a wide width PSF(11,1.00,1.00) and with Noise(30)

The first contamination (simulation) represents contamination mostly due to noise. The second simulation represents contamination due to blurring with no noise. The third simulation represents a moderate amount of contamination due to blurring and noise. The third simulation is done using images 2, 5, 10 and 1, and the first and second is done using images 2 and 5. The results of the simulation are shown in figures 11 through 17 and a summary of the third simulation results is shown in table 1.1. For all of the figures, images restored with the Wiener method, *sa* in the title above the image is the actual σ_n and *sg* is a guessed value based on looking at the contaminated image. For all images restored with RLM, VCM and CLEAN method, the values in parenthesis are the number of iterations used. Below each restored image the psdr, rmse and execution time are shown.



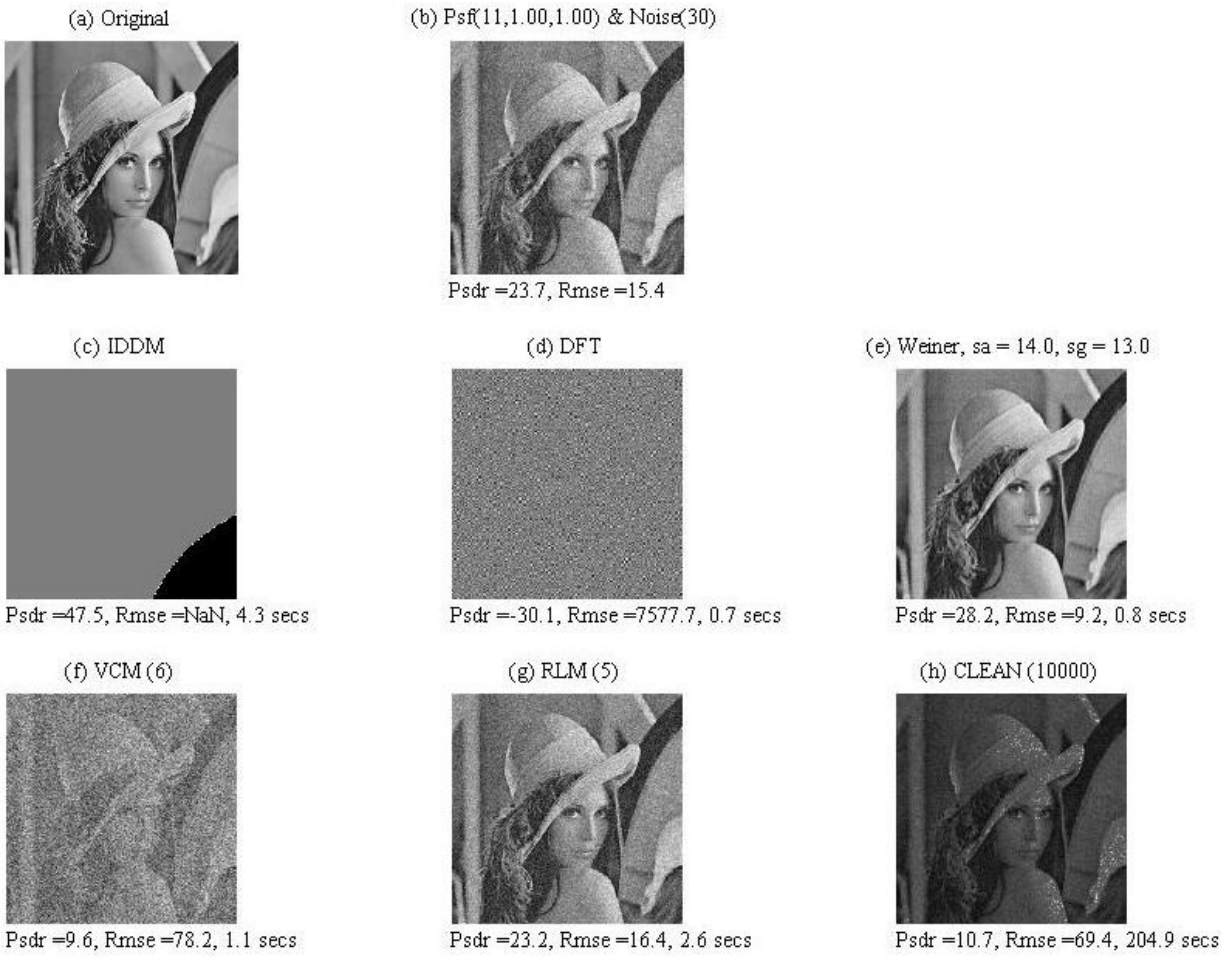
Restoration results of an image contaminated with a small amount of blur and Noise(30).

Figure 1.11



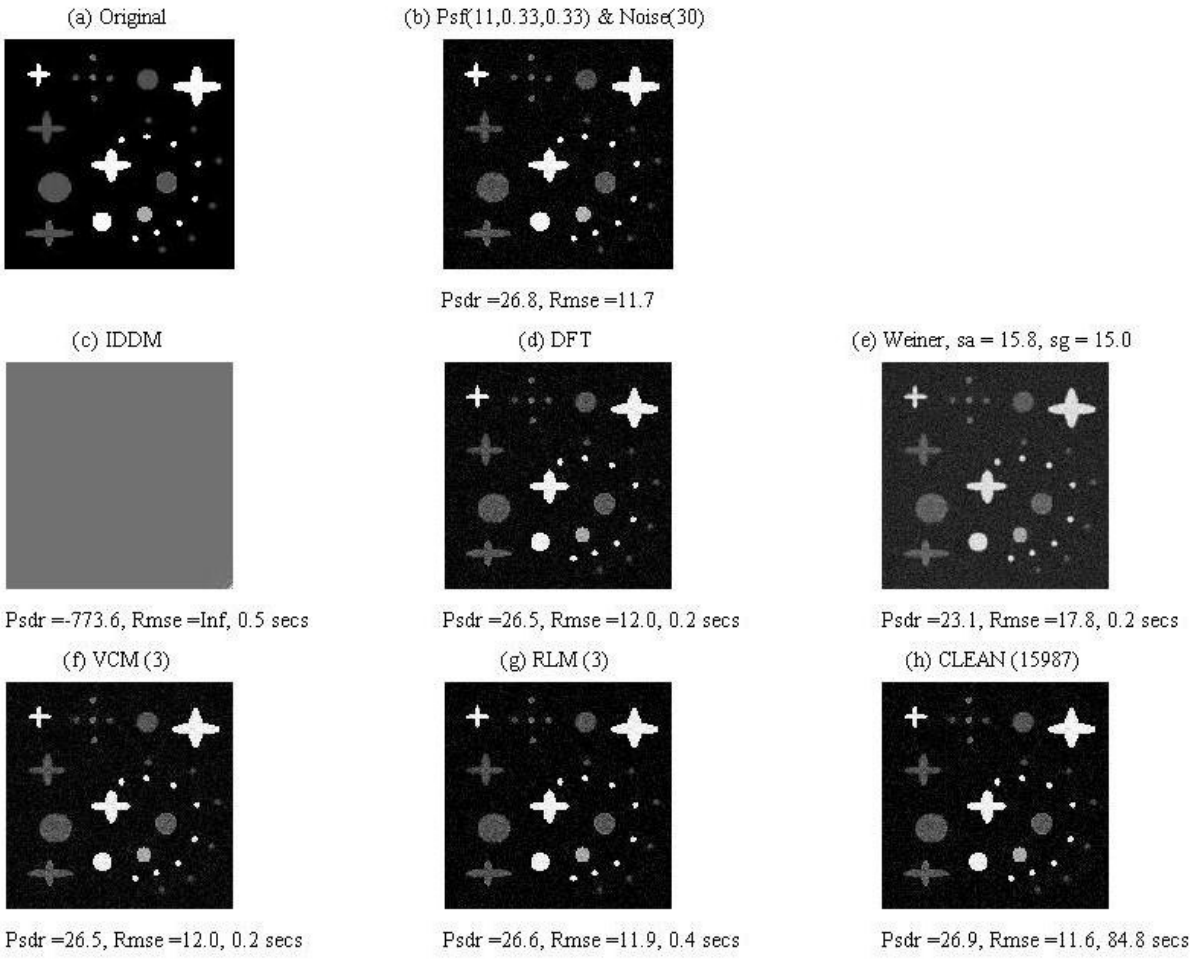
Restoration results of an image contaminated with no noise and a large amount of blur.

Figure 1.12



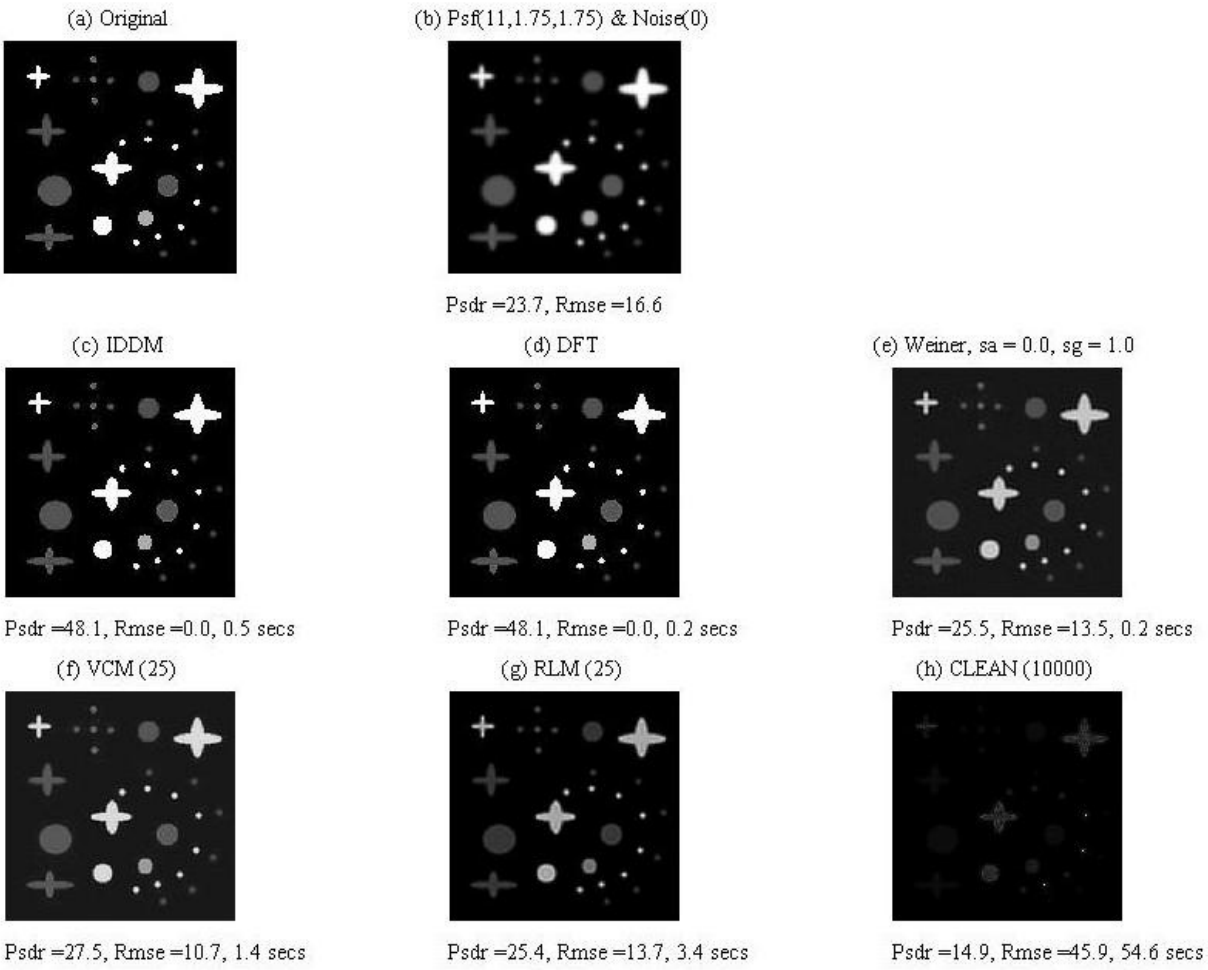
Restoration results of an image contaminated with Noise(30) and blur.

Figure 1.13



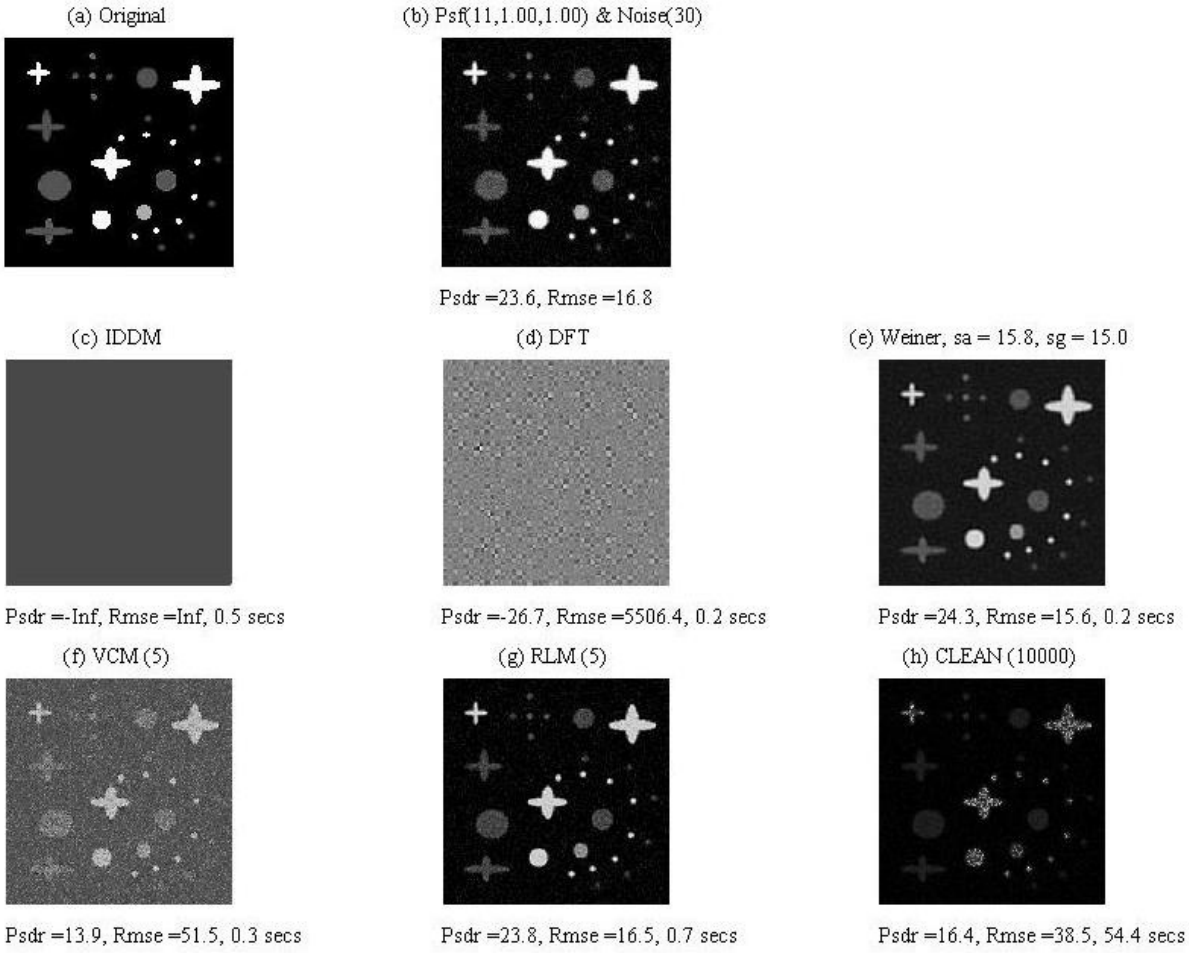
Restoration results of an image contaminated with a small amount of blur and Noise(30).

Figure 1.14



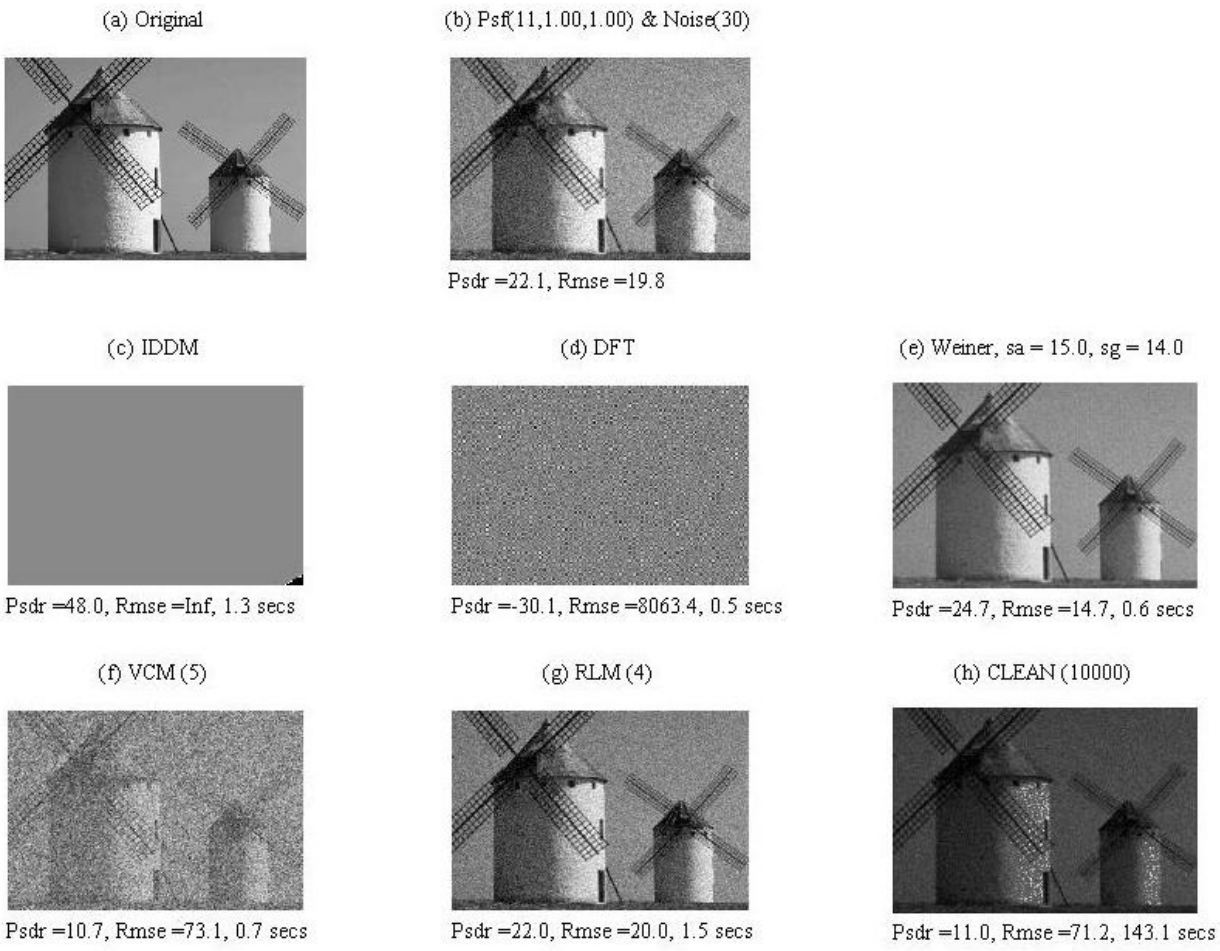
Restoration results of an image contaminated with no noise and a large amount of blur.

Figure 1.15



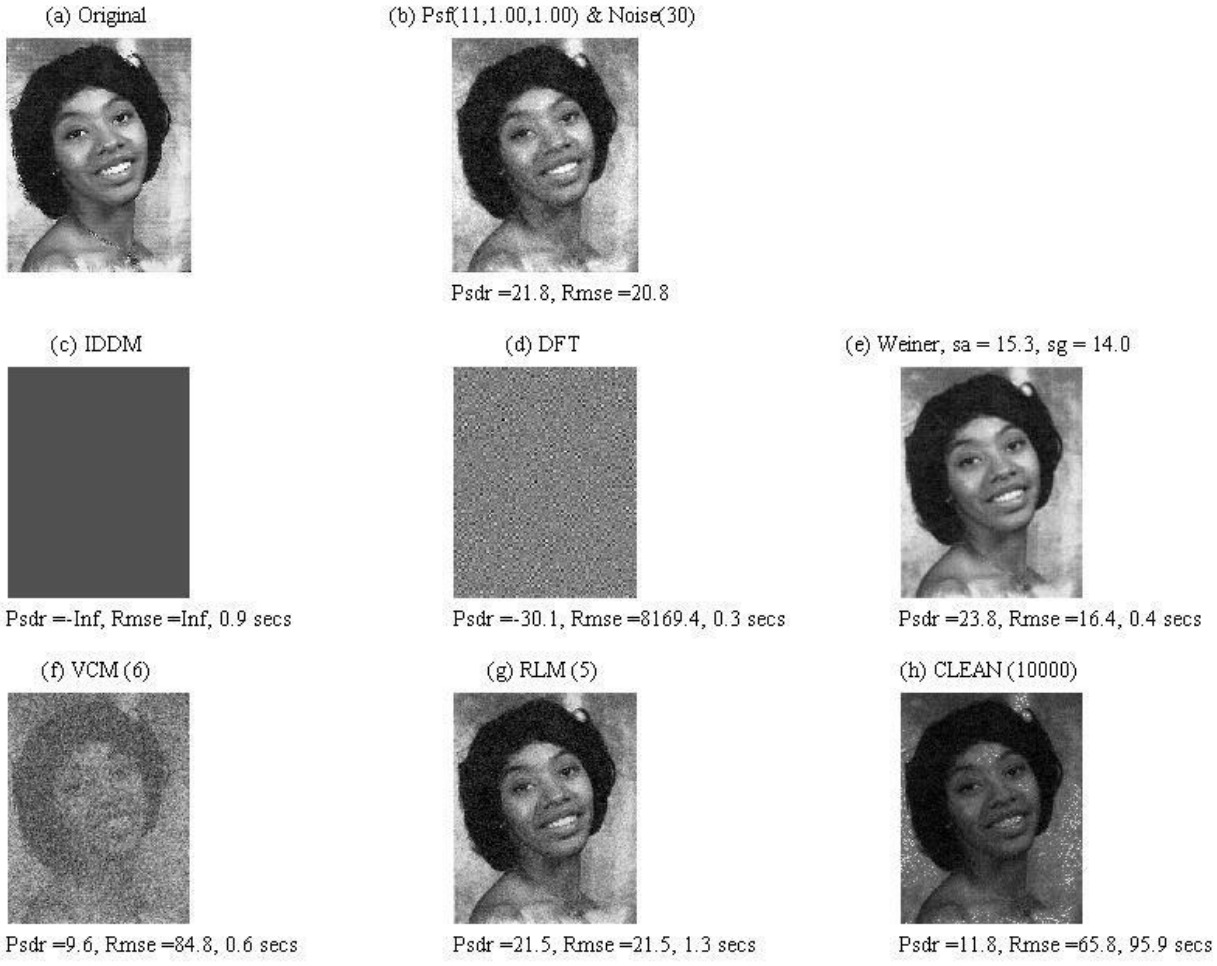
Restoration results of an image contaminated with Noise(30) and blur.

Figure 1.16



Restoration results of an image contaminated with Noise(30) and blur.

Figure 1.17



Restoration results of an image contaminated with Noise(30) and blur.

Figure 1.18

Summary: Comparison Restoration Methods Results PSF(11,1.0,1.0) Noise(30)

Image (No.)	Method	Psdr	Rmse	Time sec	Iterations	Restored	Best Method
Lena	Image data	23.7	15.4				
(2)	IDDM	-Inf	Inf	4.3	1	No	
	DFT	-30.1	7577.7	0.7	1	No	
	Weiner	28.2	9.2	0.8	1	Yes	**Yes**
	VCM	9.6	78.2	1.1	6	No	
	RLM	23.2	16.4	2.6	5	Yes	
	CLEAN	10.7	69.4	204.9	10000	No	
Points	Image data	23.6	16.8				
(5)	IDDM	-Inf	Inf	0.5	1	No	
	DFT	-26.7	5506.4	0.2	1	No	
	Weiner	24.3	15.6	0.2	1	Yes	**Yes**
	VCM	13.9	51.5	0.3	5	No	
	RLM	23.8	16.5	0.7	5	Yes	
	CLEAN	16.4	38.5	54.4	10000	No	
Windmill	Image data	22.1	19.8				
(10)	IDDM	-Inf	Inf	1.3	1	No	
	DFT	-30.1	8063.4	0.5	1	No	
	Weiner	24.7	14.7	0.6	1	Yes	**Yes**
	VCM	10.7	73.1	0.7	5	No	
	RLM	22	20	1.5	4	Yes	
	CLEAN	11	71.2	143.1	10000	No	
Classmate	Image data	21.8	20.8				
(1)	IDDM	-Inf	Inf	0.9	1	No	
	DFT	-30.1	8169.4	0.6	1	No	
	Weiner	23.8	16.4	0.4	1	Yes	**Yes**
	VCM	7.7	104.9	0.8	8	No	
	RLM	21.5	21.5	1.3	5	Yes	
	CLEAN	11.8	65.8	95.9	10000	No	

Table of simulation results for PSF(11,1.0,1.0) and Noise(30)

Table 1.1

The data in figures 1.11 through 1.18 indicate that the best direct method is the Wiener method. The Wiener method restored all of the test images, (although some better than others) for every type of PSF and amount of noise tested. If there is no noise contamination then the best method for giving the consistently lowest rmse is the IDDM, however the data show that it is one the slowest methods and real world data will almost always contain noise. The IDDM on average required 3 times more time than the next best non-noise handling method, the DFT method. If keeping the restoration time to a minimum is important then the DFT is a better method than the IDDM. The data show that the DFT can handle some noise unlike the IDDM, which did not restore any of the images with noise. However the DFT failed to restore images with moderate noise and a medium width PSF, as shown by figures 1.13, 1.16, 1.17 and 1.18. The data in table 1.1 shows that for the worst simulation case (moderate amounts of blur and noise) the best method based on highest PSDR and smallest RMSE is the Wiener and the second best is RLM.

The data in figures 1.11 through 1.18 indicate that the best iterative method is the RLM because it produced the best looking restored image and best psdr and rmse numbers as compared to the other iterative methods. The CLEAN only produced an improved image in figure 1.14, the case of a narrow PSF and moderate amount of noise, which is not surprising since the method is designed to restore images of bright points against a dark background. The only image that is close to being in the category of bright spots on a dark background is the Points image 5 – which is used in figures 1.14 through 1.16. The Clean method is the slowest of the three iterative methods. It required thousands of iterations to produce an improved image – if it improved the image at all. I think the CLEAN method is best suited for use with astronomical

type images, which is what it was originally designed for (recorded by an imaging system with a narrow PSF).

The data in figures 1.11 through 1.18 show the RLM and VCM to be better image restoration methods than the CLEAN method. In addition, the figures show that the RLM tends to converge to a solution in fewer iterations than the VCM and for the same number of iterations the VCM is the faster of the two. However the RLM produced better results for wide PSF's than the VCM and has better noise handling abilities.

To summarize, the data in figures 1.11 through 1.18 and table 1.1 indicate that the best image restoration method in the presence of noise and blur, of the six investigated, is the Wiener method. The Wiener method in almost every situation produced an improved restored image in terms of a higher PSDR and lower RMSE and by visual inspection. The biggest problem with the method is determining an estimate of the noise and thus the noise standard deviation, σ_n. In the next chapter I will develop and investigate a method to automate the determination of σ_n using a noise reduction method. It will be used in conjunction with the Wiener method.

CHAPTER 2

NOISE SIGMA ESTIMATION METHOD

The first goal of this chapter is to investigate the effects of using noise reduction to enhance the appearance of contaminated image data. There are currently various noise reducing methods, in this dissertation two of them will be tested and evaluated. The first method of noise reduction is the Wavelet Noise Reduction Method (WNRM), [9]. The second method is the Morrison Noise Reduction Method (MNRM), [11]. The best method based on quality of results and speed of execution will be determined. The last but most important goal of this chapter is to develop a method, which by use of the best noise reduction method selected (WNRM or MNRM), can be used to determine an estimate of σ_n of the noise in contaminated image data, without the noise being known.

2.1 Wavelet Noise Reduction Method

Wavelets are mathematical functions that cut up data into different frequency components, and then study each component with a resolution matched to its scale. They have advantages over traditional Fourier methods in analyzing physical situations where the signal contains discontinuities and sharp spikes. Wavelets were developed independently in the fields of mathematics, quantum physics, electrical engineering, and seismic geophysics. Interchanges between these fields during the last twenty years have led to many new wavelet applications such as image compression, turbulence, human vision, radar, and earthquake prediction. In this research, wavelets are used for de-noising 2-d image data.

The de-noising procedure of a noisy image can be divided into three stages, [25] [24]:

1) The wavelet transform of an image.
2) Thresholding of wavelet coefficients.
3) The inverse wavelet transform.

In the above stages, the vital key is stage (2), how to threshold the wavelet coefficients.

For stage 1, the coefficients of the chosen wavelet must be calculated. I have chosen to use Daubechies wavelets, [4]. The coefficients, a_k, are solved for from the following three conditions

$$[2.1], \quad \sum_{k=1}^{N} a_k = 2$$

$$[2.2], \quad \sum_{k=1}^{N} (-1)^{k-1} a_k k^{m-1} = 0$$

$$[2.3], \quad \sum_{k=1}^{N} a_k a_u = 2\delta(0, m-1)$$

where, u = k +2*(m-1) and, m = 1,2…(N/2). Next a forward transform matrix from the coefficients is constructed, [18]. For example for D4 wavelets the transform matrix is

[2.4], $$W = \begin{vmatrix} a_1 & a_2 & 0 & 0 \\ b_1 & b_2 & 0 & 0 \\ 0 & 0 & a_1 & a_2 \\ 0 & 0 & b_1 & b_2 \end{vmatrix}$$

[2.5], $$b_k = (-1)^{k-1} a_{(N-k+1)}$$

[2.6], $$F = W h$$

For stage (2), thresholding is applied to F, that is the values in F below λ are set equal to zero, this is called hard thresholding and is shown in equation 2.8. Soft thresholding is shown in equation 2.9 where the sgn(F_{xy}) is the sign of F_{xy}, [9] [7].

[2.7], $$\lambda = STD(h)\sqrt{2 \cdot \log(M)}$$

[2.8], $$if \mid F_{x,y} \mid \geq \lambda \ then \ F_{x,y} = F_{x,y} \ else \ F_{x,y} = 0$$

[2.9], $$F_{x,y} = \text{sgn}(F_{x,y}) \cdot Min([0, (\mid F_{x,y} \mid -\lambda)])$$

STD = Standard deviation, and M = height and width of W, (W is a square matrix). Next the de-noised image, f, is calculated using the inverse wavelet transform matrix which is equal to the transpose of the wavelet transform matrix.

[2.10], $$f = W^{'} F$$

A flow chart of the implementation of the WNRM method used in this dissertation is shown in figures 2.1 and 2.2. The Matlab code implementing the method is shown in the Code section of this paper.

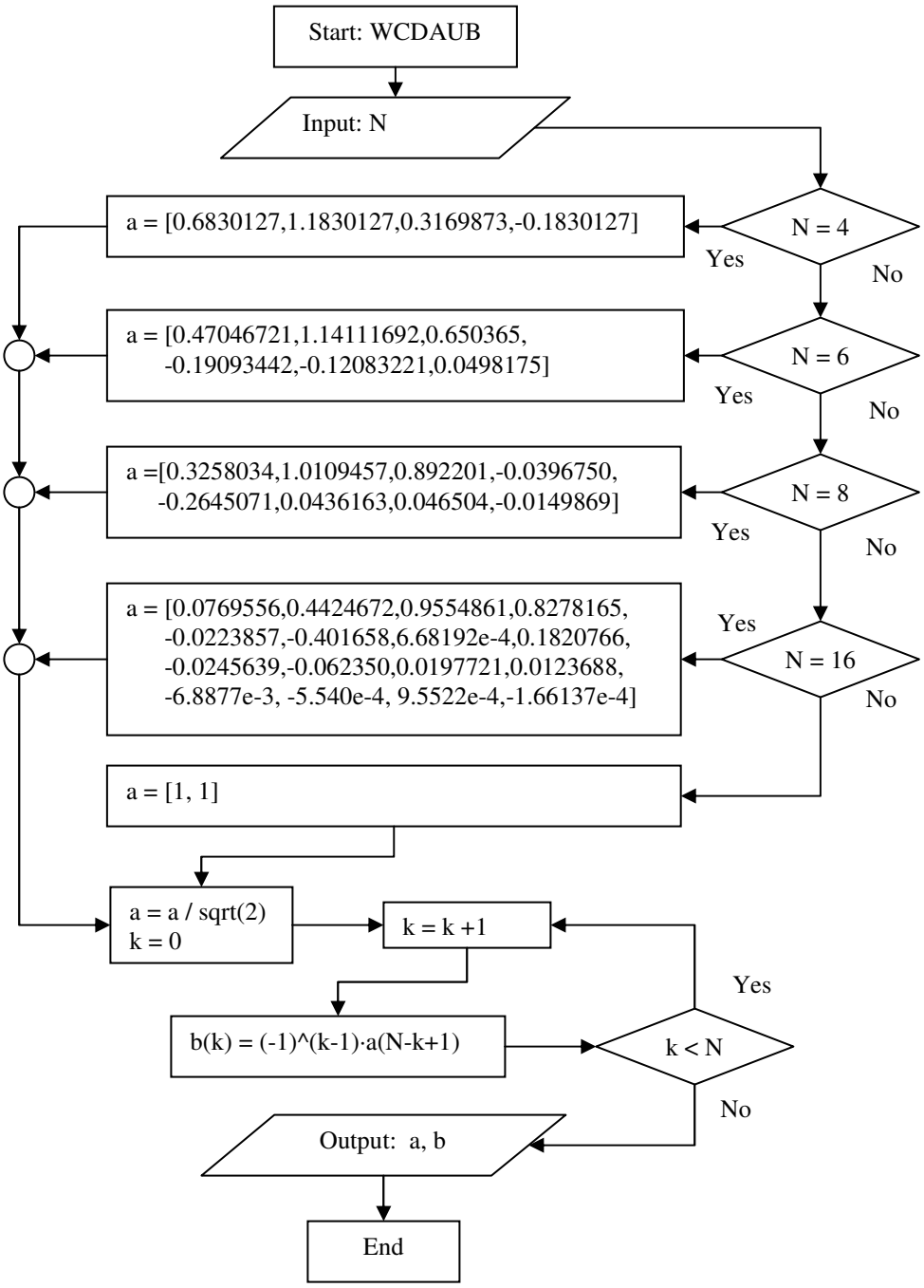

Start: WCDAUB
Input: N
N = 4
Yes
No
a = [0.6830127,1.1830127,0.3169873,-0.1830127]
N = 6
Yes
No
a = [0.47046721,1.14111692,0.650365,
-0.19093442,-0.12083221,0.0498175]
N = 8
Yes
No
a =[0.3258034,1.0109457,0.892201,-0.0396750,
-0.2645071,0.0436163,0.046504,-0.0149869]
N = 16
Yes
No
a = [0.0769556,0.4424672,0.9554861,0.8278165,
-0.0223857,-0.401658,6.68192e-4,0.1820766,
-0.0245639,-0.062350,0.0197721,0.0123688,
-6.8877e-3, -5.540e-4, 9.5522e-4,-1.66137e-4]
a = [1, 1]
a = a / sqrt(2)
k = 0
k = k +1
b(k) = (-1)^(k-1)·a(N-k+1)
k < N
Yes
No
Output: a, b
End

Figure 2.1

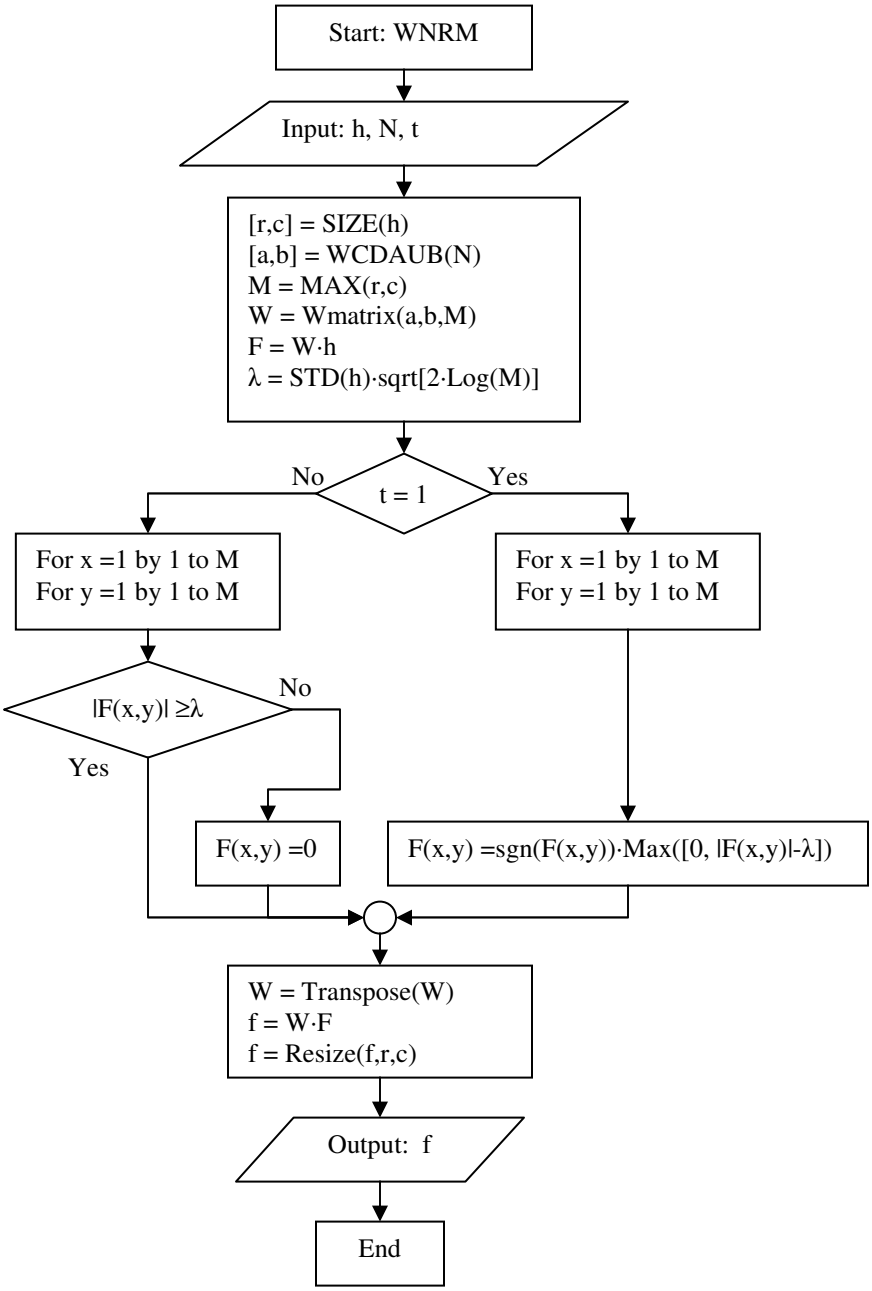
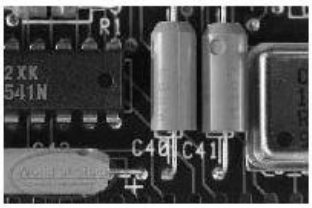
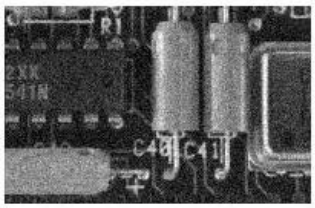

Start: WNRM
Input: h, N, t
[r,c] = SIZE(h)
[a,b] = WCDAUB(N)
M = MAX(r,c)
W = Wmatrix(a,b,M)
F = W·h
λ = STD(h)·sqrt[2·Log(M)]
No
t = 1
Yes
For x =1 by 1 to M
For y =1 by 1 to M
For x =1 by 1 to M
For y =1 by 1 to M
|F(x,y)| ≥λ
No
Yes
F(x,y) =0
F(x,y) =sgn(F(x,y))·Max([0, |F(x,y)|-λ])
W = Transpose(W)
f = W·F
f = Resize(f,r,c)
Output: f
End

Figure 2.2

(a) Original

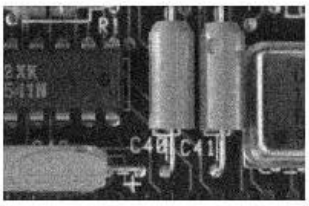


(b) Psf(11,1.00,1.00) & Noise(30)



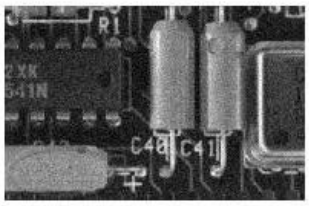
Psdr = 23.4, Rmse = 17.0

(c) WNRM (D4), Soft T



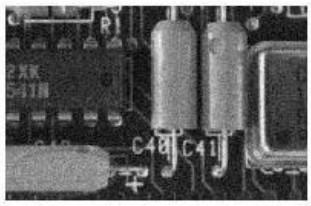
Psdr = 22.6, Rmse = 18.7, 2.5 secs

(d) WNRM (D8), Soft T



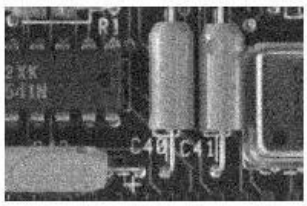
Psdr = 22.7, Rmse = 18.6, 2.4 secs

(e) WNRM (D16), Soft T



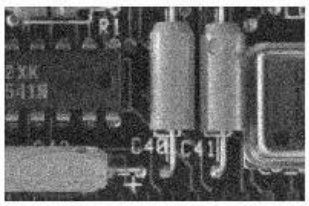
Psdr = 22.6, Rmse = 18.6, 2.4 secs

(f) WNRM (D4), Hard T



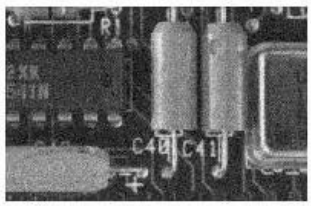
Psdr = 23.8, Rmse = 16.2, 1.7 secs

(g) WNRM (D8), Hard T



Psdr = 23.9, Rmse = 16.2, 1.7 secs

(h) WNRM (D16), Hard T



Psdr = 23.9, Rmse = 16.1, 1.7 secs

Results of the WNRM applied to image data contaminated by a medium width PSF and Noise(30).

Figure 2.3

(a) Lena PSF(11,1,1) NS(30)

Method	NI/WL	PSDR	RMSE	Time
Actual		23.729	15.429	
WNRM Soft-T	D4	22.795	17.180	1.06
WNRM Soft-T	D8	22.864	17.044	1.05
WNRM Soft-T	D16	22.809	17.151	1.02
WNRM Hard-T	D4	24.228	14.567	0.31
WNRM Hard-T	D8	24.255	14.521	0.31
WNRM Hard-T	D16	24.235	14.554	0.33

(b) Rose PSF(11,1,1) NS(30)

Method	NI/WL	PSDR	RMSE	Time
Actual		23.289	17.462	
WNRM Soft-T	D4	21.384	21.744	1.84
WNRM Soft-T	D8	21.404	21.693	1.81
WNRM Soft-T	D16	21.356	21.814	1.89
WNRM Hard-T	D4	23.910	16.257	1.19
WNRM Hard-T	D8	23.933	16.214	1.17
WNRM Hard-T	D16	23.863	16.345	1.19

(c) Circuit PSF(11,1,1) NS(30)

Method	NI/WL	PSDR	RMSE	Time
Actual		23.429	16.980	
WNRM Soft-T	D4	22.582	18.720	2.45
WNRM Soft-T	D8	22.650	18.573	2.41
WNRM Soft-T	D16	22.628	18.622	2.42
WNRM Hard-T	D4	23.828	16.217	1.73
WNRM Hard-T	D8	23.855	16.168	1.73
WNRM Hard-T	D16	23.870	16.140	1.72

Table 2.1, Medium width PSF and with time in seconds.

The data in figure 2.3 and table 2.1 show that WNRM using hard thresholding produces better restoration results than using soft thresholding. The data also show that the best image restoration results, using D4 through D16 wavelet lengths, occurred most often when using a D8 wavelet. However, the results for all three wavelet lengths were very close. Therefore to compare the WNRM to the Morrison Noise Reduction method, which will be discussed next section, only hard thresholding D8 wavelets will be used.

2.2 Morrison Noise Reduction Method

The Morrison Noise Reduction method (MNRM) is an iterative noise reduction, [11]. It is often used to reduce the noise in data before it is processed by another image restoration technique. The method smoothes and reduces the noise in the known image data, h, by convolving the image h with the PSF b then iterating equation 2.12, shown below, a specified number of times to restore back to h gradually but without incompatible noise, [11] [12].

[2.11], $h_1 = h**b, \quad k = 1.$ [2.12], $h_{k+1} = h_k + [h - h_k]**b.$

One way to determine when to stop the iterations is to calculate the change in RMSE of $(h - h_k)$, if it is below some set tolerance then the process is stopped. Stopping the method when the change in RMSE reaches a set tolerance or the maximum number of set iterations is reached are the two stopping criteria I use in my implementation of the MNRM. One criterion for the method to give a significantly improved image (in terms of reduced noise) is the PSF must be of sufficient width. In my research I have performed many simulations with various images and have determined numerically that the method will give a significantly improved image if equations 2.13 and 2.14 are satisfied.

[2.13], $\sigma_x \geq 0.8.$ [2.14], $\sigma_y \geq 0.8.$

In equations 2.3 and 2.4 σ_x and σ_y are the same variables defined in equation 1.6 for the PSF. The MNRM does not require that the PSF used be the actual PSF. Thus if the given PSF is narrow, i.e., does not meet the conditions in equations 2.13 and 2.14, then another (wider) PSF can be substituted. In my implementation of the MNRM I have set the PSF used to be one of medium width. This is the same as the PSF shown in figure 1.2b.

A flow chart of the implementation of the MNRM method used in this dissertation is shown in figure 2.4. The Matlab code implementing the method is shown in the Code section of this paper. Note in figure 2.4 if the variable w = 1 then the given PSF, bi, is narrow, if w = 2 then bi is of moderate width and if w = 3 then bi is a wide width PSF.

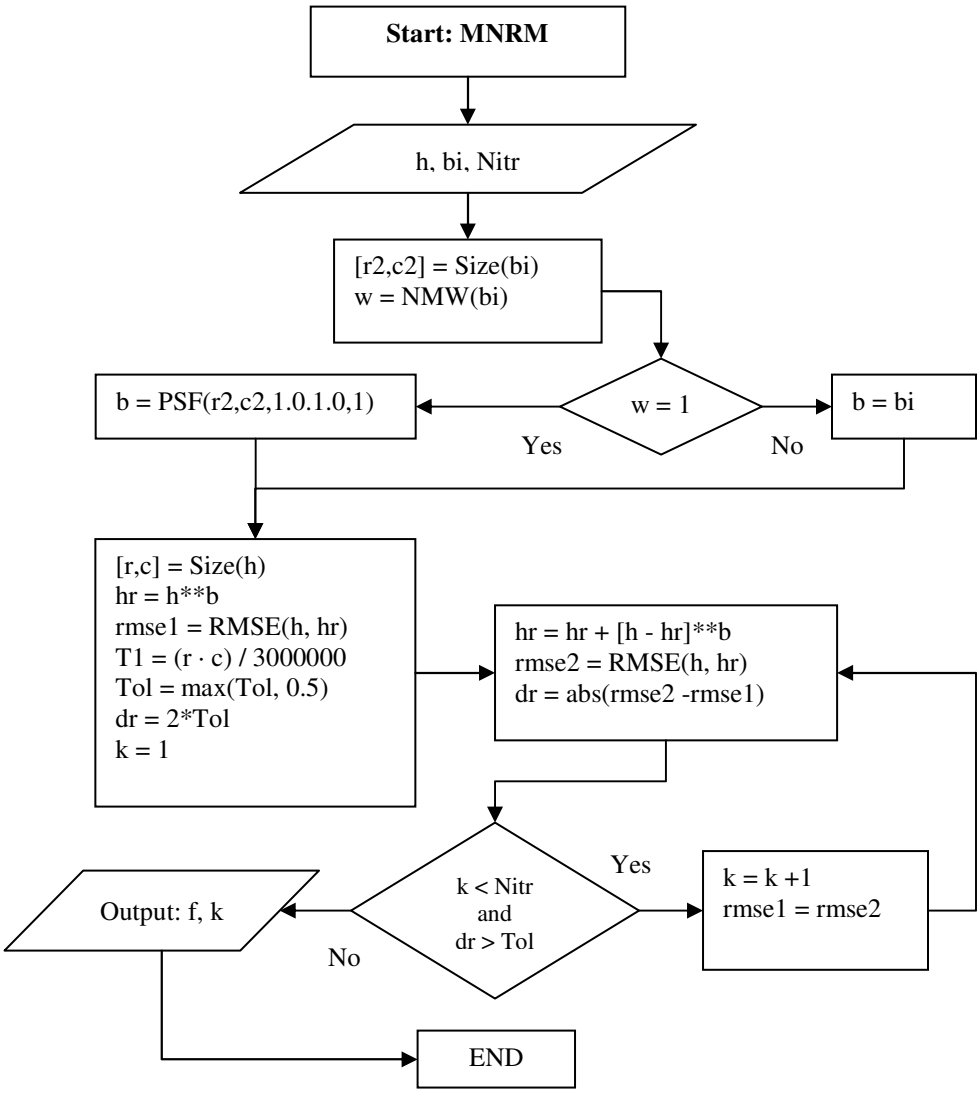
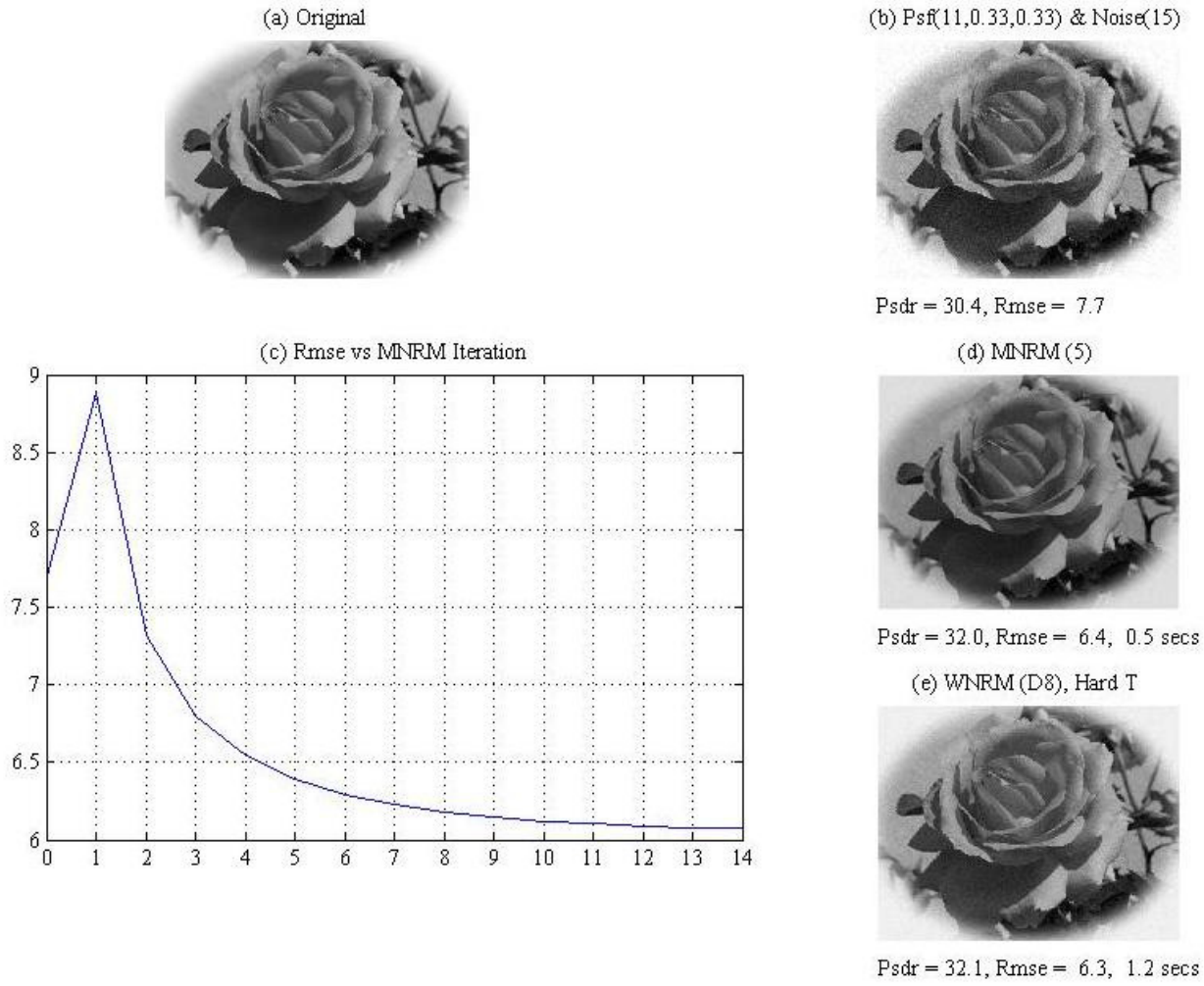


Figure 2.4

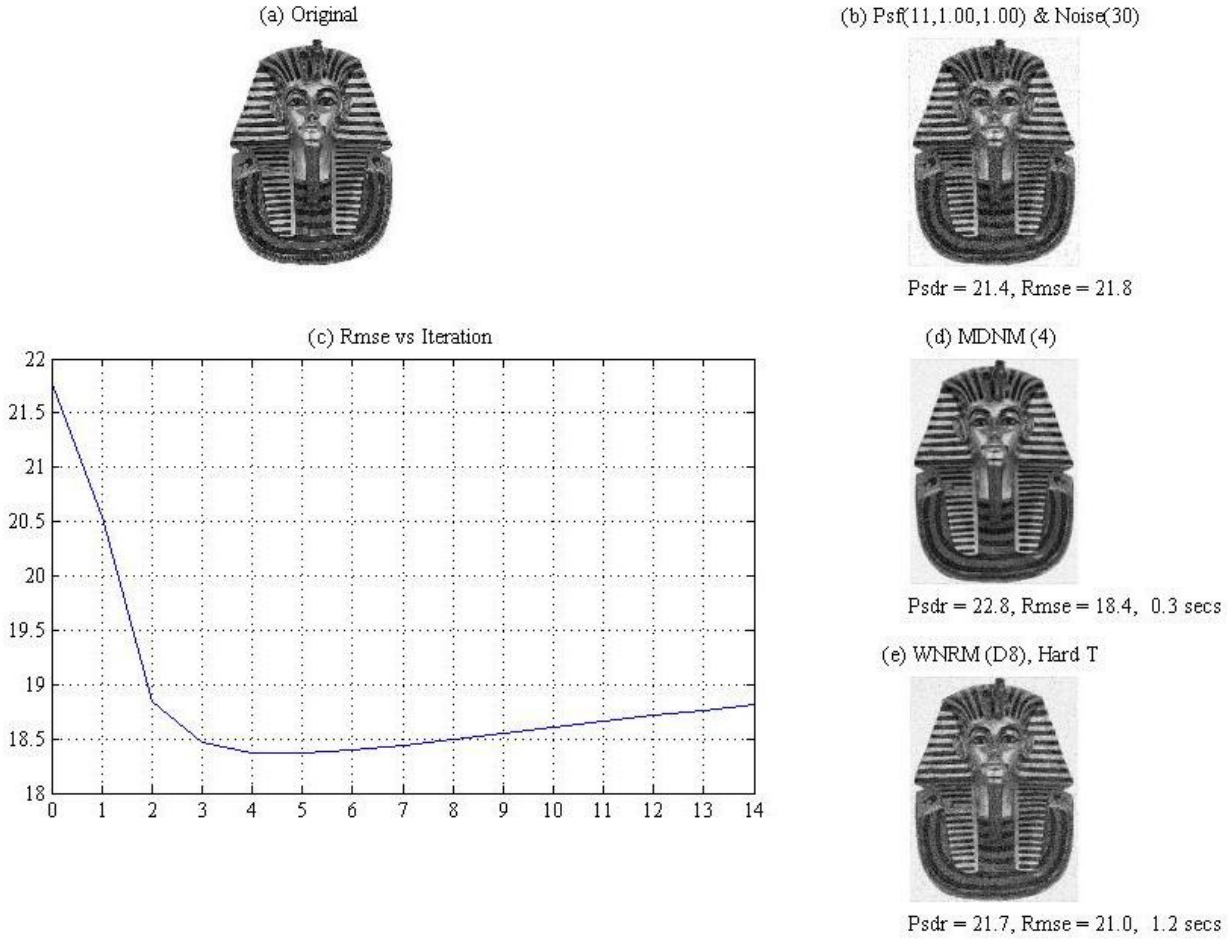
2.3 Comparison of MNRM and WNRM

The Rose image was corrupted with a small amount of contamination, the King Tut image with a moderate amount of contamination and the Windmill image with a large amount of noise and blur contamination. The WNRM and the MNRM were used to restore the three images. The restoration results are shown, respectively, in figures 2.5 through 2.7.



Results of MNRM and WNRM applied to image data contaminated by a narrow width PSF and Noise(15).

Figure 2.5



Results of MNRM and WNRM applied to image data contaminated by a medium width PSF and Noise(30).

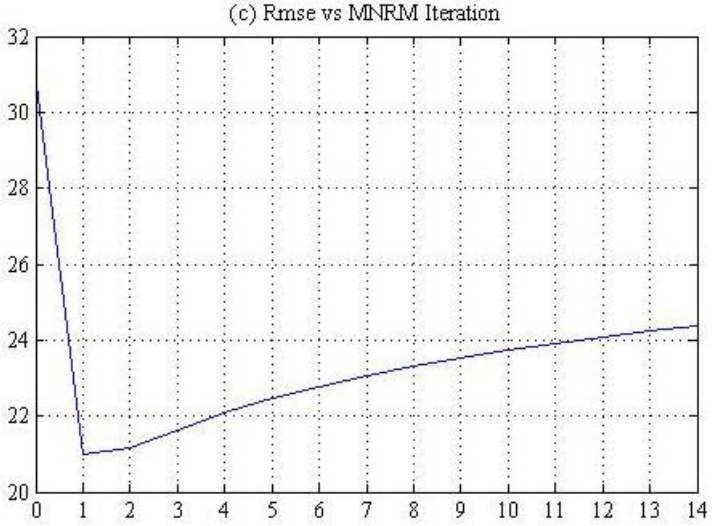
Figure 2.6

(a) Original

(b) Psf(11,1.75,1.75) & Noise(50)

Psdr = 18.2, Rmse = 30.9

(c) Rmse vs MNRM Iteration



(d) MNRM (6)

Psdr = 20.9, Rmse = 22.8, 0.5 secs

(e) WNRM (D8), Hard T

Psdr = 18.4, Rmse = 30.3, 1.9 secs

Results of MNRM and WNRM applied to image data contaminated by a wide width PSF and Noise(50).

Figure 2.7

The data in figures 2.5 through 2.7 show that my implementations of the MNRM and WNRM (using hard thresh-holding) reduces the noise in the given noise and blur contaminated images, because in all figures the resulting image has an increased PSDR and a decreased RMSE. The data in parts (c) and (d) shows that the chosen tolerance in the MNRM algorithm yields a close to optimum number of iterations for the Morrison method (figure 2.4). In all cases the number of Morrison iterations executed coincides fairly close to the minimum of the RMSE vs MNRM iteration graph. The results in figure 2.5 indicate that the WNRM does a slightly better job of restoring the contaminated image than the MNRM. Thus I conclude for small amounts of contamination the WNRM is a slightly better choice than the MNRM, but the MNRM is a very close second. Figures 2.6 and 2.7 show that for moderate and lager amounts of image contamination, the MNRM produces better image restoration than the WNRM. Additionally, the data in all three of the figures indicate that MNRM is faster than the WNRM. This is surprising since the MNRM is an iterative method and the WNRM is a direct method. It seemed that the WNRM should be faster – but the results, for the images I have run simulations on, show otherwise. I do think that for some image sizes and amounts of contamination the WNRM could yield faster results than the MNRM – it just did not do so for the image and contamination combinations I investigated.

The MNRM and the WNRM unlike the Wiener method (chosen as the best restoration method in chapter 1) only reduce image noise contamination and not blur contamination. Because of their noise reducing abilities, I conclude the MNRM or the WNRM can be used as part of a noise estimation method to determine the σ_n of noise-contaminated image data. The noise estimator, utilizing the MNRM, will be the subject of the next section.

2.4 Noise Sigma Estimation Method

I have developed the Sigma Estimation Method (SIGEST) to estimate the σ_e of the noise, σ_n, in contaminated image data without the noise actually being known. SIGEST is based upon using the Morrison Noise Reduction Method (MNRM). A flowchart of the SIGEST algorithm is shown in figure 2.8. Note that in figure 2.8 the variable *w*, is defined as in figure 2.4.

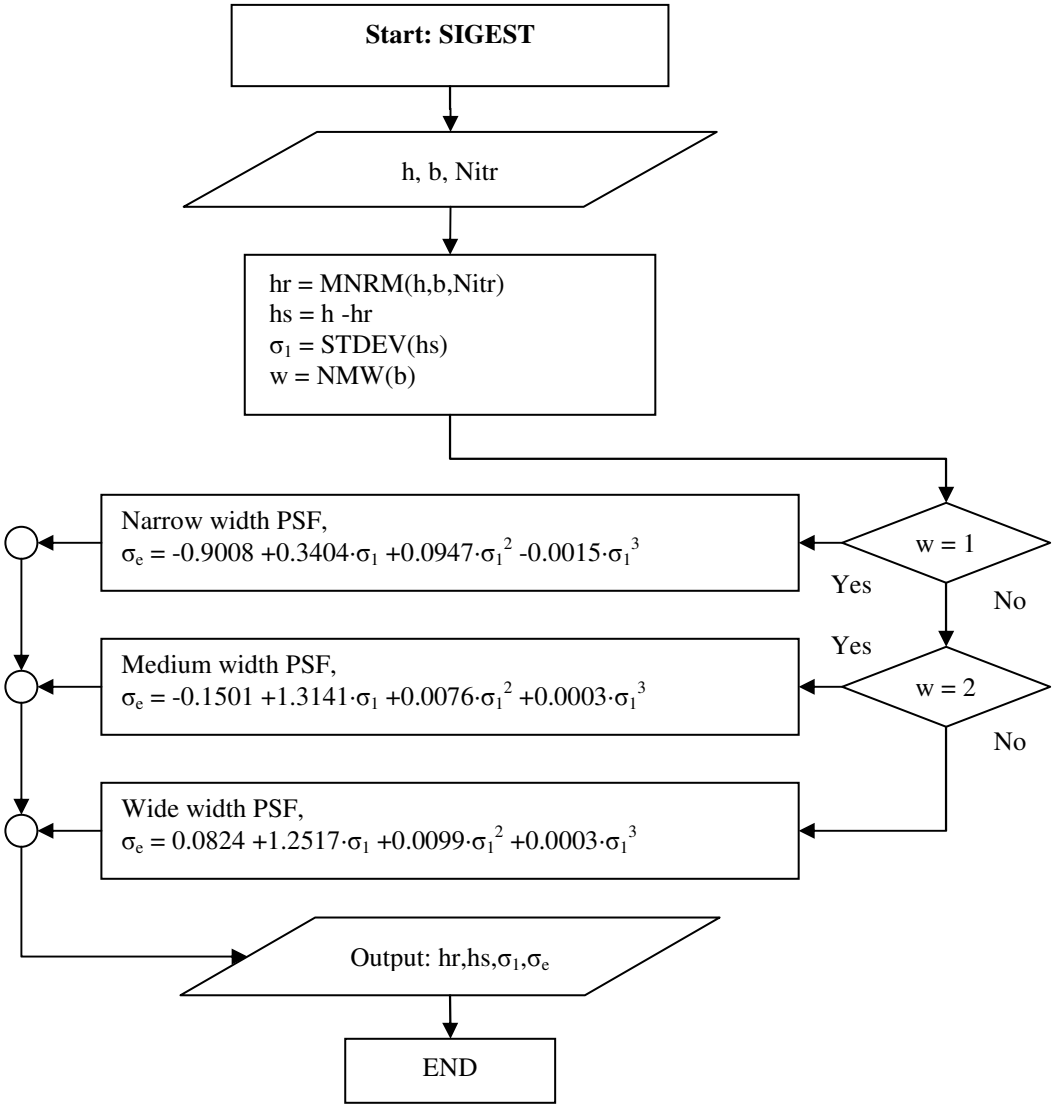


Figure 2.8

SIGEST determines a numerical estimate of σ_n of noise contaiminated image data h by calculating, via MNRM, a reduced noise version of h, *hr*. To calculate the first estimated noise standard deviation, σ_1, the method simply calculates the standard deviation of (h –hr). The differenence between h and hr should give a good estimate of the noise in h. As stated earlier in this dissertation, the MNRM does not remove all of the noise contamination from an image. Therfore I have incorporated in to SIGEST a correction equation to the value of σ_1 to yield a more accurate estimated noise standard deviation, σ_e. To develop the correction equations for σ_e, σ_1 was calculated for NS(x) with x from 0 to 100 in steps of 5. The calculations were done for the following images: Classmate, Moon-Man, Fruit-1 and Windmill. Figure 2.9 shows a graph of the resulting data for the Fruit-1 image. Notice that σ_1 for all PSF widths used is almost always less than the actual σ_n, and the shape of the σ_1 data is a curved line rather than a straight line as the shape of σ_n is. The shape of the σ_1 line for all PSF widths were very similar for the Classmate, Moon-Man and Windmill images as those shown in figure 2.9 for the Fruit-1 image. Thus I have selected the following equation to correct σ_1 to σ_n (i.e., to calculate σ_e),

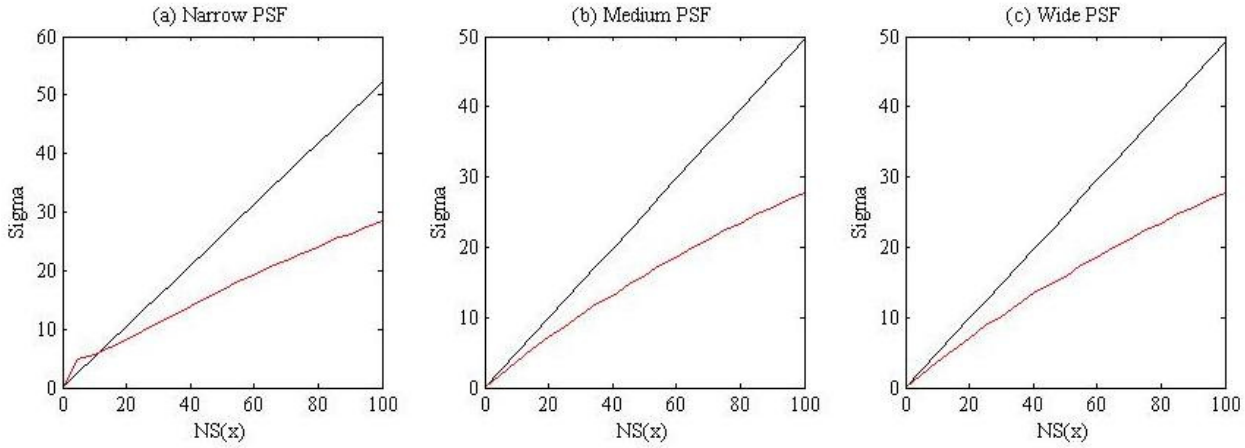
[2.15], $\sigma_e = c_1 + c_2 \cdot \sigma_1 + c_3 \cdot \sigma_1^2 + c_4 \cdot \sigma_1^3$.

The values of c_1 through c_4 were determined using regression analysis, [20] [1], and calculated using Matlab code that I wrote. For all four of the previously named images the value of the c's are given by,

[2.16], $X(r) = [1, \sigma(r), \sigma^2(r), \sigma^3(r)]$, $r \; from \; 0 \; to \; 20 \; by \; 1$.

[2.17], $c = \left(X^T \cdot X\right)^{-1} \cdot X^T \cdot \sigma_n$.

Note that in equation 2.16, X is a matrix calculated from the determined values of σ_e, some of which are shown in figure 2.9. The value of the c's for narrow, medium and wide width PSFs are shown in the SIGEST method flow chart shown in figure 2.8.



Results of SIGEST applied to image data contaminated by a narrow, medium and wide width PSF all with noise NS(x), with x varied from 0 to 100. The actual sigma is in black the calculated first sigma is shown in red.

Figure 2.9

I have tested the method on the first ten of the images shown in figure 1.10, for noise levels of NS(10), NS(30) and NS(60) (recall equation 1.5). The results of testing SIGEST are shown in tables 2.2 through 2.4. In the tables the x in MNRM(x) is the acutal number of iterations done by the MNRM and the terms Dif(Sa,Se) and Dif(Sa,S1) are defined as follows:

[2.18], $$Dif(Sa,Se) = \frac{(Sa - Se)}{Sa} \cdot 100.$$

[2.19], $$Dif(Sa,S1) = \frac{(Sa - S1)}{Sa} \cdot 100.$$

(a) Results for NS(10) PSF(11,0.33,0.33)

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	6	5.2681	10.0417	10.5477	90.61%	100.22%
2	Lenna	4	4.8772	5.3505	3.4019	9.70%	30.25%
3	Rose	5	5.2682	7.1238	5.7878	35.22%	9.86%
4	Moon-Man	5	5.2239	5.7142	3.8566	9.39%	26.17%
5	Points	5	5.2827	7.9009	6.9604	49.56%	31.76%
6	Circuit	4	5.2022	5.3351	3.3830	2.55%	34.97%
7	Clock	5	5.2554	6.5724	5.0012	25.06%	4.84%
8	KingTut	7	5.2709	10.0697	10.5978	91.04%	101.06%
9	Fruit1	5	5.2690	7.3134	6.0671	38.80%	15.15%
10	Windmill	6	5.1939	7.3610	6.1378	41.72%	18.17%
					Average =	**39.37%**	**37.25%**

(b) Results for NS(30) PSF(11,0.33,0.33)

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	7	15.8042	13.7720	17.8306	12.86%	12.82%
2	Lenna	5	14.6317	11.1497	12.5881	23.80%	13.97%
3	Rose	6	15.8045	11.9900	14.2092	24.14%	10.09%
4	Moon-Man	5	15.6717	11.2847	12.8445	27.99%	18.04%
5	Points	5	15.8480	10.3669	11.1345	34.59%	29.74%
6	Circuit	5	15.6066	11.6376	13.5221	25.43%	13.36%
7	Clock	5	15.7662	11.8267	13.8894	24.99%	11.90%
8	KingTut	7	15.8128	13.6725	17.6224	13.54%	11.44%
9	Fruit1	6	15.8071	11.3539	12.9766	28.17%	17.91%
10	Windmill	7	15.5817	12.1302	14.4853	22.15%	7.04%
					Average =	**23.76%**	**14.63%**

(c) Results for NS(60) PSF(11,0.33,0.33)

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	9	31.6083	21.0550	34.2471	33.39%	8.35%
2	Lenna	7	29.2633	20.1997	32.2523	30.97%	10.21%
3	Rose	8	31.6090	20.3195	32.5315	35.72%	2.92%
4	Moon-Man	8	31.3433	19.1949	29.9164	38.76%	4.55%
5	Points	6	31.6960	15.5393	21.6275	50.97%	31.77%
6	Circuit	8	31.2132	20.1261	32.0809	35.52%	2.78%
7	Clock	8	31.5324	19.2404	30.0218	38.98%	4.79%
8	KingTut	9	31.6256	20.3618	32.6303	35.62%	3.18%
9	Fruit1	8	31.6141	18.9057	29.2468	40.20%	7.49%
10	Windmill	9	31.1633	20.6564	33.3171	33.72%	6.91%
					Average =	**37.38%**	**8.29%**

Table 2.2, Narrow width PSF

(a) **NS(10)** **PSF(11,1.,1.)**

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	3	5.0968	4.1479	5.4528	18.62%	6.98%
2	Lenna	3	4.6791	3.6957	4.8254	21.02%	3.13%
3	Rose	3	5.2682	4.0906	5.3731	22.35%	1.99%
4	Moon-Man	3	4.9613	3.7352	4.8800	24.71%	1.64%
5	Points	3	5.2827	2.9991	3.8674	43.23%	26.79%
6	Circuit	3	5.0965	4.0100	5.2610	21.32%	3.23%
7	Clock	3	5.2554	3.9413	5.1656	25.00%	1.71%
8	KingTut	4	5.2709	3.8720	5.0694	26.54%	3.82%
9	Fruit1	3	5.2690	3.8041	4.9753	27.80%	5.57%
10	Windmill	3	5.0074	4.0962	5.3809	18.20%	7.46%
					Average =	**24.88%**	**6.23%**

(b) **NS(30)** **PSF(11,1.,1.)**

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	4	15.2903	11.3087	16.1164	26.04%	5.40%
2	Lenna	4	14.0374	10.4929	14.8219	25.25%	5.59%
3	Rose	4	15.8045	11.3235	16.1402	28.35%	2.12%
4	Moon-Man	4	14.8840	10.2602	14.4569	31.07%	2.87%
5	Points	3	15.8480	8.1171	11.1778	48.78%	29.47%
6	Circuit	4	15.2894	11.3275	16.1466	25.91%	5.61%
7	Clock	4	15.7662	10.9566	15.5549	30.51%	1.34%
8	KingTut	4	15.8128	11.0272	15.6671	30.26%	0.92%
9	Fruit1	4	15.8071	10.4189	14.7057	34.09%	6.97%
10	Windmill	4	15.0223	11.1743	15.9016	25.62%	5.85%
					Average =	**30.59%**	**6.61%**

(c) **NS(60)** **PSF(11,1.,1.)**

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	7	30.5806	19.9963	31.5645	34.61%	3.22%
2	Lenna	7	28.0748	19.2344	30.0724	31.49%	7.12%
3	Rose	7	31.6090	20.3188	32.2051	35.72%	1.89%
4	Moon-Man	7	29.7680	18.5183	28.6962	37.79%	3.60%
5	Points	5	31.6960	14.7052	21.7715	53.61%	31.31%
6	Circuit	7	30.5788	20.1190	31.8077	34.21%	4.02%
7	Clock	7	31.5324	19.3301	30.2581	38.70%	4.04%
8	KingTut	7	31.6256	19.7564	31.0916	37.53%	1.69%
9	Fruit1	7	31.6141	18.8416	29.3144	40.40%	7.27%
10	Windmill	7	30.0446	20.2906	32.1489	32.47%	7.00%
					Average =	**37.65%**	**7.12%**

Table 2.3, Medium width PSF

(a) **NS(10)** **PSF(11,1.75,1.75)**

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	3	5.0599	3.9325	5.1761	22.28%	2.30%
2	Lenna	2	4.5967	3.7576	4.9414	18.25%	7.50%
3	Rose	3	5.2681	3.9950	5.2601	24.17%	0.15%
4	Moon-Man	3	4.9180	3.5781	4.7016	27.24%	4.40%
5	Points	3	5.2826	2.8874	3.7863	45.34%	28.33%
6	Circuit	3	4.8873	3.7910	4.9863	22.43%	2.03%
7	Clock	3	5.2553	3.8365	5.0472	27.00%	3.96%
8	KingTut	3	5.2709	3.8958	5.1267	26.09%	2.74%
9	Fruit1	3	5.2689	3.7041	4.8699	29.70%	7.57%
10	Windmill	3	4.9860	3.8755	5.0995	22.27%	2.28%
					Average =	**26.48%**	**6.12%**

(b) **NS(30)** **PSF(11,1.75,1.75)**

No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	4	15.1797	11.2537	15.8500	25.86%	4.42%
2	Lenna	4	13.7901	10.3242	14.3906	25.13%	4.35%
3	Rose	4	15.8043	11.3768	16.0459	28.01%	1.53%
4	Moon-Man	4	14.7539	10.2123	14.2171	30.78%	3.64%
5	Points	3	15.8478	8.2119	11.1949	48.18%	29.36%
6	Circuit	4	14.6620	10.8953	15.2833	25.69%	4.24%
7	Clock	4	15.7660	11.0049	15.4561	30.20%	1.97%
8	KingTut	4	15.8126	11.0973	15.6021	29.82%	1.33%
9	Fruit1	4	15.8068	10.5268	14.7058	33.40%	6.97%
10	Windmill	4	14.9580	11.1300	15.6538	25.59%	4.65%
					Average =	**30.27%**	**6.24%**

(c) **NS(60)** **PSF(11,1.75,1.75)**

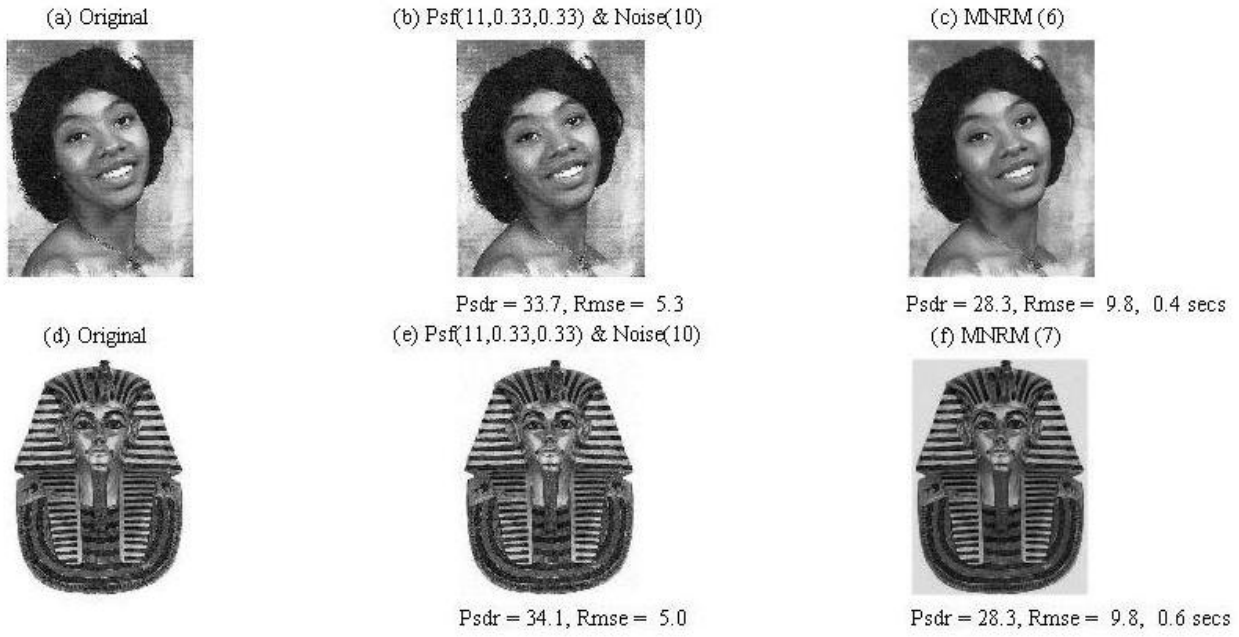
No.	Image	MDNM(x)	Sigma Actual	Sigma 1	Sigma Estimated	|Dif(Sa,S1)|	|Dif(Sa,Se)|
1	Classmate	7	30.3593	19.9901	31.4566	34.15%	3.61%
2	Lenna	7	27.5801	18.9434	29.3859	31.31%	6.55%
3	Rose	7	31.6085	20.4502	32.3860	35.30%	2.46%
4	Moon-Man	7	29.5078	18.4969	28.5207	37.32%	3.35%
5	Points	5	31.6956	14.9153	21.9497	52.94%	30.75%
6	Circuit	7	29.3240	19.4579	30.3962	33.65%	3.66%
7	Clock	7	31.5320	19.4661	30.4124	38.27%	3.55%
8	KingTut	7	31.6251	20.0304	31.5374	36.66%	0.28%
9	Fruit1	7	31.6137	19.0410	29.5764	39.77%	6.44%
10	Windmill	7	29.9159	20.2887	32.0583	32.18%	7.16%
					Average =	**37.16%**	**6.78%**

Table 2.4, Wide width PSF

I define a successful noise estimate method to be one that gives results that are within 10% of the actual noise, when averaged over several different images. In this case the images are the ten shown in tables 2.2, 2.3 and 2.4. The data in tables 2.2, 2.3 and 2.4 shows that for narrow, medium and wide width PSFs and for all noise levels tested the SIGEST successfully estimates the actual σ_n – for all but one contamination combination. The one contamination combination that the SIGEST method fails to yield a good average σ_e, is for a narrow PSF and noise of Noise(10), which is the smallest amount of contamination tested. However if the biggest outliners were removed (Classmate and King Tut) then the average error would be 17.1%. Depending on the need and or use of σ_e, an error of 17.1% may be acceptable – as will be the case for image restoration using the Wiener method in chapter 3. I think the SIGEST method does not calculate a good σ_e in table 2.2a (the smallest amount of contamination) because the images have little noise and blur contamination in them and thus there is very little noise that the MNRM can remove from the image. However the MNRM does yield an improved restoration of the images even though they contain little contamination. The images shown in figure 2.10, which have the worst two σ_e results in table 2.2a, show that although the PSDR and RMSE are not improved from the contaminated images the restored images do look better! I think the visual improvement in the images is due to MNRM reducing the noise in the images while increasing the amount of blur in them.

The data in tables 2.2, 2.3 and 2.4, except for 2.2a, shows that the σ_1 values are lower than the actual noise values because the Morrison method does not remove all of the noise in an image. However the results indicate that using a good stopping tolerance in MNRM (see figure 2.4) results in the Morrison method removing a significant amount of noise contamination from an image. I developed the stopping tolerance shown in figure 2.4 after trying various stopping

criteria. The stopping criteria in figure 2.4 produced the best results when used with the images shown in figure 1.10 for various amounts of noise. Also note that the correction equations developed do a very good job at calculating σ_e from the value of σ_1, as the majority of the data in the average percent difference cells of tables 2.2, 2.3 and 2.4 clearly show.



Results of MNRM applied to image data containing little contamination, i.e., a narrow width PSF and Noise(10).

Figure 2.10

One possible way the SIGEST method could be made more accurate is to adjust the value of Tol based upon small, medium and large amounts of noise but this would require more user input to the method. One of my requirements for the SIGEST method is that it require very little user input thus the value of Tol will be calculated as shown in figure 2.4. The SIGEST method will be used by the sub-imaging restoration method described in the next chapter.

CHAPTER 3

SUB-IMAGING METHOD

The goal of this chapter is to develop an algorithm using the Wiener method that automates the determination of a good noise σ for a given set of contaminated image data, h, and restores the image data. For the case of whole image noise contamination, the method improves the speed of the noise estimation method (SIGEST) by using sub-images. The method is named the Sub-Image Wiener Method (SIWM).

3.1 Sub-Imaging Method

I define the h2 of an image to be the resulting image data produced by two iterations of the MNRM applied to the image i.e,

[3.1], $h2 = MNRM\,(h,2)$.

I define the hN of an image to be the estimated noise in the image resulting from two iterations of the Morrison Method. The calculation of hN is shown in equation 3.2.

[3.2], $hN = h - h2$.

The Sub-Imaging Method (SIM) divides the hN of an image into several pieces and selects the piece with an estimated σ_e of σ_n which is closest to the average of the standard deviation of all of the pieces. I define the selected sub-image to be the average sub-image (asi). A flow chart showing the calculation of an image's asi is shown in Figure 3.1.

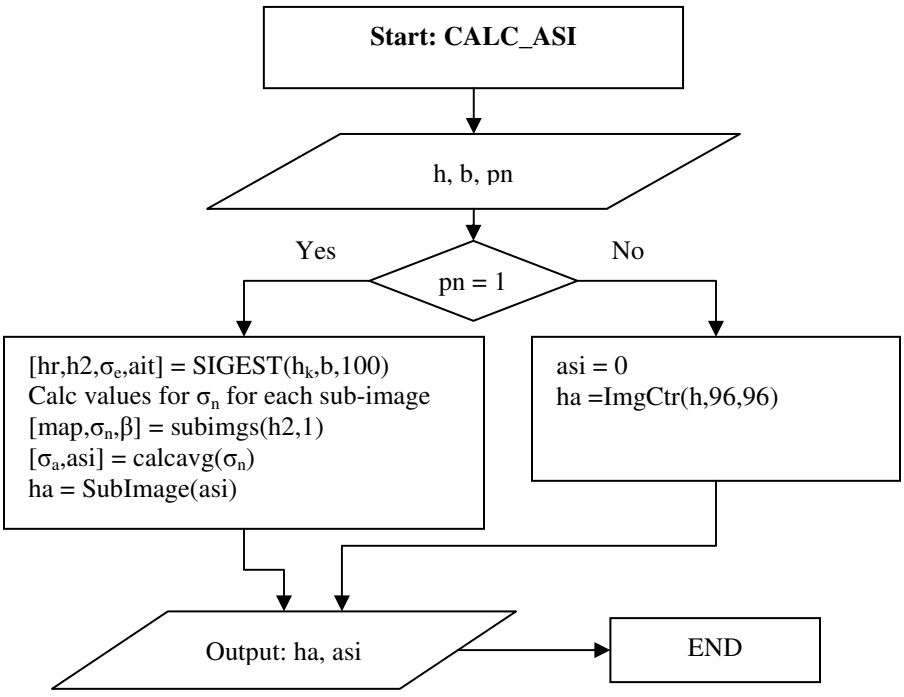


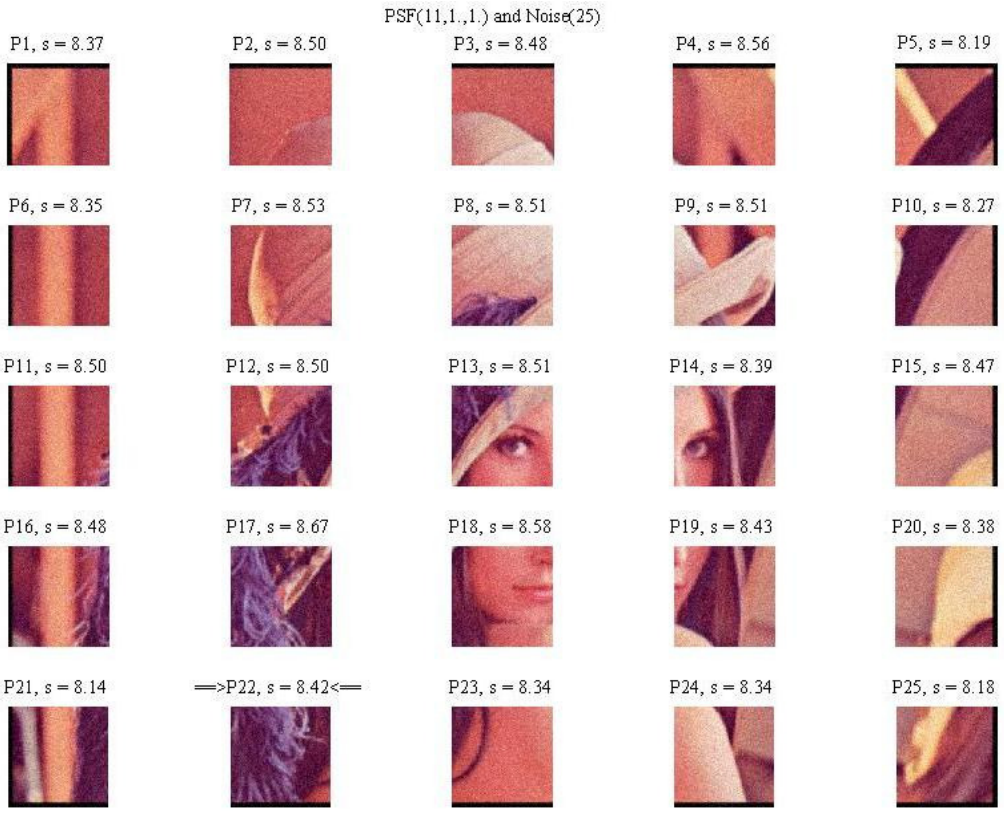
Figure 3.1

For image data that are uniformly noise contaminated throughout the image pn is set to 0, (meaning no partial noise). An example of this is shown in figure 3.2 and its' asi is shown in figure 3.3. For this noise case, the function Calc_Asi does the following:

1. Set asi = 0
2. Set ha equal sub-image of the given image equal to a 96 by 96 pixel image copied from the center of the contaminated image, returns ha and asi.

For image data that are partially noise contaminated as in figure 3.4, pn is set to 1 (meaning partial noise is present). For this noise case, the function Calc_Asi does the following:

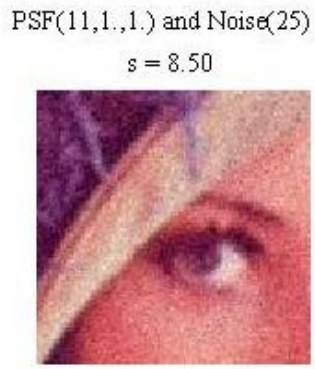
1. Divide the image into sub-images each having a size as close as possible to 96 by 96 pixels.
2. Calculate the sigma of each sub-image. Select the sub-image with a sigma closest to the average of all the sigmas, this is the *asi*.
3. Set ha equal to the asi and return ha and its' sub-image number.



Blur and nose contaminated Lena image divided into 5 rows by 5 columns of sub-images. Note that part 22 is the asi of the image.

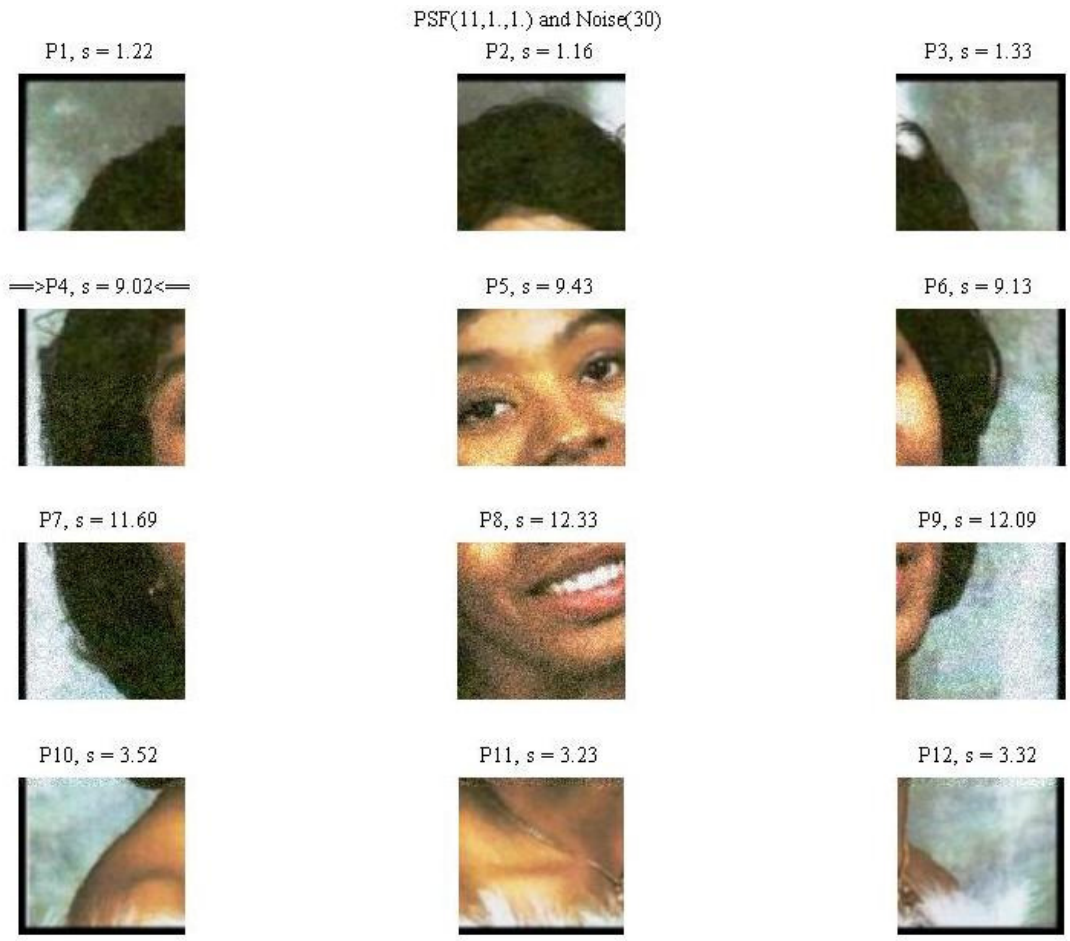
Figure 3.2

Note figure 3.2 shows the results of setting pn equal to 1, in Calc_Asi, however since the image is uniformly noise contaminated, the average of the sub-images is approximately equal to all of the sub-images as is the shown in the figure. Thus in figure 3.2 the best choice is to set pn equal to 0 when using Calc_Asi and the resulting sub-image is shown in figure 3.3. Figure 3.4 shows the results of a partially noise contaminated image with pn equal to 1.



Blur and nose contaminated Lena image with center sub-image extracted. Note that the noise sigma is approximately the same as that of the asi in figure 3.2.

Figure 3.3



Blur and partially noise contaminated Classmate image divided into 4 rows by 3 columns of sub-images. The asi of the image is part 4.

Figure 3.4

3.2 Sub-Imaging Wiener Method

The following assertions are made concerning SIM:

1. The asi of an image can be used in the place of the whole image to determine σ_e of σ_n and used to restore the whole image. Therefore, the noise in a piece of an image can represent the noise in the whole image (provided it is the asi of the image's hN).
2. SIM can be combined with SIGEST and the Wiener method to restore contaminated image data without the noise σ of the data being known. This combination I define to be the Sub-Image Wiener Method (SIWM).

A flow chart of the SIWM is shown in figure 3.5.

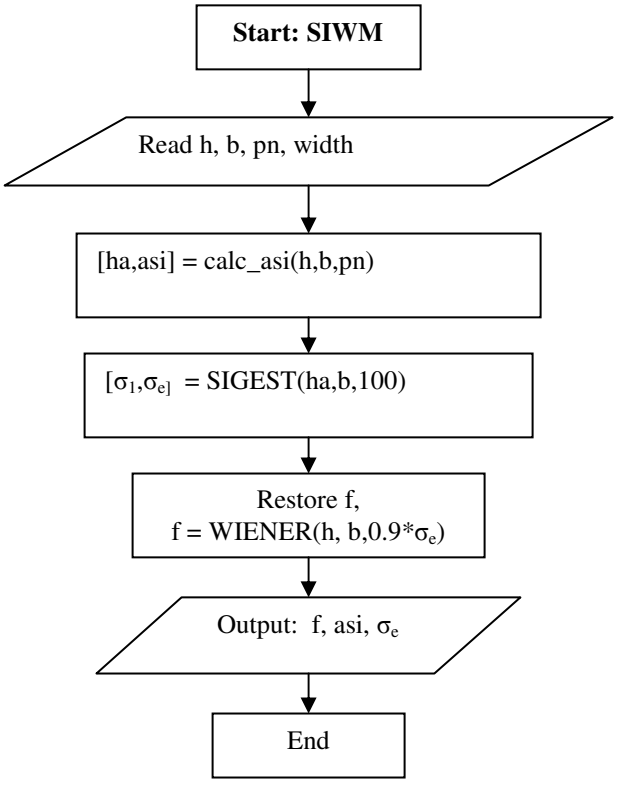


Figure 3.5

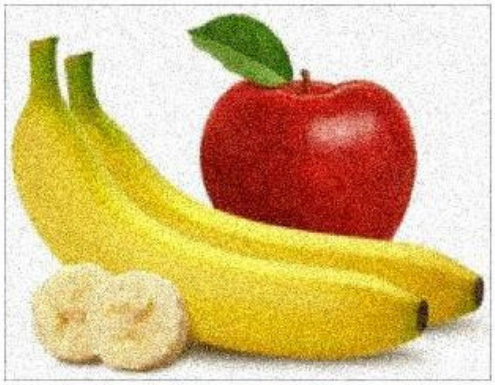
3.3 Comparison of the SIWM and the Wiener Method

Note that in the SIWM the sigma used in the Weiner part of the method is σ_1, which as shown in chapter 2 is typically 10% to 25% less than the actual sigma, σ_n. I have chosen to use σ_e*0.9 instead of σ_e because in executing the many simulations with the SIWM and Wiener methods the results lead to an increased PSDR and decreased RMSE. Using a smaller value of sigma also leads to a better looking restored image because the smaller value causes more detail and noise to remain in the restored image. However the human eye has filtering abilities, thus the eye filters out the extra noise and more detail is seen. Nevertheless the value of σ_e is returned because the true estimated value of sigma may be needed for other reasons.

The SIWM was applied to images 1, 2, 4, 7, 9, 10, 11, 12, 13 and 14. Some of the restored image results are shown in figures 3.6 through 3.8 and all are summarized in table 3.1. In figures 3.6 through 3.8, in the title of image (c) the value of *sigma* given is the actual σ_n of the image noise. In the title of image (d) the calculated values of σ_1 and σ_e are shown as $s = \sigma_1$ and σ_e and the x in SIWM(x) is the actual number of iterations of the MNRM used to calculated σ_1. Note that the values of PSDR and RMSE are shown below image (b) in figures 3.6 through 3.8 and the values of PSDR, RMSE and the amount of time in seconds for execution of the SIWM is shown at the bottom of image (d). Similar information is shown below image (c), except the time shown is for execution of the Wiener method.

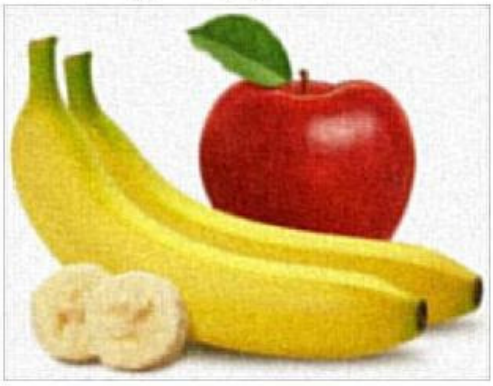
(a) Original

(b) Psf(11,1.00,1.00) & Noise(35)



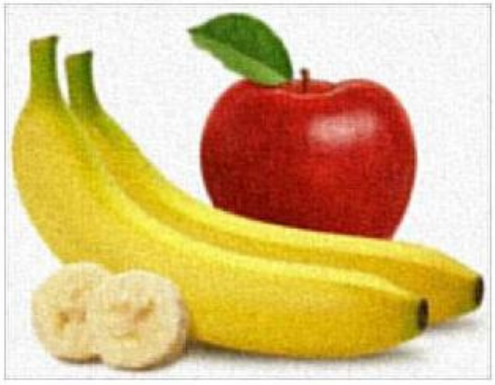
Psdr =23.2, Rmse =17.7

(c) Wiener, sigma =17.65



Psdr =27.4, Rmse =10.9, 1.4 secs

(d) SIWM(2), 1 x 1, asi =0, s =12.10, 17.39



Psdr =27.5, Rmse =10.8, 2.1 secs

Results of SIWM applied to Fruit-1 image contaminated with Noise(35).
Figure 3.6

(a) Original

(b) Psf(11,1.00,1.00) & Noise(70)

Psdr =16.8, Rmse =36.9

(c) Wiener, sigma =36.80

Psdr =21.5, Rmse =21.6, 3.1 secs

(d) SIWM(4), 1 x 1, asi =0, s =19.26, 30.11

Psdr =21.2, Rmse =22.2, 4.0 secs

Results of SIWM applied to Shuttle image contaminated with Noise(70). Note that although the image is contaminated by a very large amount of noise the method still noticeably improves the image.

Figure 3.7

(a) Original

(b) Psf(11,1.00,1.00) & Noise(40)

Psdr =22.4, Rmse =19.3

(c) Wiener, sigma =13.48

Psdr =24.0, Rmse =16.1, 1.0 secs

(d) SIWM(5), 4 x 3, asi =6, s =11.41, 16.27

Psdr =23.9, Rmse =16.3, 1.5 secs

(e) SIWM(2), 1 x 1, asi =0, s =14.82, 21.98

Psdr =23.5, Rmse =17.0, 1.2 secs

Results of SIWM applied to Classmate image partially contaminated with Noise(40). In 3.8(d) the asi is used and in 3.8(e) it is not, note the restoration in 3.8(e) is degraded from 3.8(d) – the asi is needed!

Figure 3.8

(a) PSF(11,11,0.33,0.33)

Image	Method	Sigma	PSDR	RMSE	Time (sec)
Classmate	Contaminated	21.13	21.99	20.27	
Noise(40)	SIWM(4,15.54)	21.63	14.21	49.65	1.19
	Wiener	21.13	14.18	49.84	0.91
Lena	Contaminated	13.06	25.24	12.97	
Noise(30)	SIWM(2,9.87)	10.25	16.36	36.04	2.56
	Wiener	13.06	16.35	36.10	1.98
Moon-Man	Contaminated	18.39	23.45	17.08	
Noise(35)	SIWM(3,13.19)	16.63	13.25	55.27	1.77
	Wiener	18.39	13.20	55.55	1.28
Clock	Contaminated	22.54	21.90	20.49	
Noise(45)	SIWM(3,15.64)	21.84	14.19	49.78	3.41
	Wiener	22.54	14.05	50.59	2.66
Fruit1	Contaminated	17.65	24.53	15.15	
Noise(35)	SIWM(2,12.46)	15.14	14.14	50.07	1.83
	Wiener	17.65	14.00	50.88	1.44
Windmill	Contaminated	31.26	18.35	30.46	
Noise(60)	SIWM(7,19.70)	31.09	14.79	45.93	1.83
	Wiener	31.26	14.57	47.07	1.45
Fruit2	Contaminated	13.08	26.00	12.73	
Noise(25)	SIWM(3,14.09)	18.49	15.39	43.19	4.25
	Wiener	13.08	15.35	43.41	3.06
Sphinx	Contaminated	9.76	28.05	9.70	
Noise(20)	SIWM(2,7.04)	5.67	18.65	28.63	1.80
	Wiener	9.76	18.76	28.26	1.45
Eagle	Contaminated	14.95	24.73	14.79	
Noise(30)	SIWM(2,11.22)	12.73	14.06	50.56	1.75
	Wiener	14.95	14.00	50.89	1.38
Shuttle	Contaminated	36.80	17.54	33.84	
Noise(70)	SIWM(5,19.17)	29.86	13.53	53.73	4.11
	Wiener	36.80	13.11	56.35	3.22

Table 3.1

(b) PSF(11,11,1.00,1.00)

Image	Method	Sigma	PSDR	RMSE	Time (sec)
Classmate	Contaminated	21.13	20.21	24.89	
Noise(40)	SIWM(2,14.82)	21.98	23.29	17.45	1.19
	Wiener	21.13	23.28	17.47	0.95
Lena	Contaminated	13.06	24.28	14.48	
Noise(30)	SIWM(2,9.45)	13.20	28.51	8.90	2.52
	Wiener	13.06	28.51	8.90	2.02
Moon-Man	Contaminated	18.39	22.11	19.92	
Noise(35)	SIWM(2,13.10)	19.05	25.36	13.71	1.63
	Wiener	18.39	25.26	13.86	1.27
Clock	Contaminated	22.54	21.12	22.42	
Noise(45)	SIWM(3,15.51)	23.17	25.87	12.97	3.41
	Wiener	22.54	25.82	13.04	2.64
Fruit1	Contaminated	17.65	23.18	17.69	
Noise(35)	SIWM(2,12.10)	17.39	27.46	10.81	1.81
	Wiener	17.65	27.38	10.91	1.50
Windmill	Contaminated	31.26	17.59	33.27	
Noise(60)	SIWM(4,20.02)	31.61	22.61	18.67	1.83
	Wiener	31.26	22.66	18.56	1.44
Fruit2	Contaminated	13.08	22.15	19.83	
Noise(25)	SIWM(2,9.31)	12.98	23.32	17.33	4.23
	Wiener	13.08	23.25	17.47	3.08
Sphinx	Contaminated	9.76	26.47	11.64	
Noise(20)	SIWM(2,6.49)	8.78	29.95	7.79	1.80
	Wiener	9.76	30.01	7.74	1.45
Eagle	Contaminated	14.95	22.81	18.46	
Noise(30)	SIWM(2,10.88)	15.43	26.81	11.64	1.72
	Wiener	14.95	26.84	11.61	1.38
Shuttle	Contaminated	36.80	16.79	36.90	
Noise(70)	SIWM(4,19.26)	30.11	21.20	22.21	4.11
	Wiener	36.80	21.46	21.56	3.19

Table 3.2

(c) PSF(11,11,1.75,1.75)

Image	Method	Sigma	PSDR	RMSE	Time (sec)
Classmate	Contaminated	21.13	19.40	27.33	
Noise(40)	SIWM(2,14.90)	21.93	22.58	18.96	1.20
	Wiener	21.13	22.55	19.02	0.92
Lena	Contaminated	13.06	23.20	16.39	
Noise(30)	SIWM(2,9.46)	13.06	27.01	10.57	2.45
	Wiener	13.06	26.92	10.69	2.00
Moon-Man	Contaminated	18.39	20.59	23.74	
Noise(35)	SIWM(2,13.13)	18.90	23.33	17.31	1.61
	Wiener	18.39	23.23	17.51	1.30
Clock	Contaminated	22.54	20.23	24.83	
Noise(45)	SIWM(3,15.52)	23.02	24.56	15.09	3.36
	Wiener	22.54	24.47	15.25	2.61
Fruit1	Contaminated	17.65	21.98	20.30	
Noise(35)	SIWM(2,12.15)	17.29	25.88	12.96	1.83
	Wiener	17.65	25.76	13.14	1.42
Windmill	Contaminated	31.26	16.93	35.89	
Noise(60)	SIWM(4,20.25)	31.99	21.68	20.78	1.81
	Wiener	31.26	21.64	20.85	1.44
Fruit2	Contaminated	13.08	20.93	22.81	
Noise(25)	SIWM(2,9.30)	12.82	22.36	19.36	4.16
	Wiener	13.08	22.30	19.50	3.06
Sphinx	Contaminated	9.76	25.15	13.55	
Noise(20)	SIWM(2,6.50)	8.72	28.43	9.28	1.80
	Wiener	9.76	28.30	9.43	1.41
Eagle	Contaminated	14.95	21.18	22.27	
Noise(30)	SIWM(2,10.88)	15.25	26.40	12.21	1.73
	Wiener	14.95	26.35	12.27	1.38
Shuttle	Contaminated	36.80	16.20	39.50	
Noise(70)	SIWM(4,19.40)	30.29	20.93	22.91	4.09
	Wiener	36.80	20.82	23.22	3.20

Table 3.3

The data in figures 3.6 through 3.8 and tables 3.1, 3.2 and 3.3, show that SIWM in every case restores the given image data, i.e., increases the PSDR and decreases the RMSE of the contaminated image. The data show that the σ_n estimation method, SIGEST, developed in chapter 2, works fairly well. In every case, of whole image noise contamination, the method determined an estimate of σ_n that is within 18% of the actual σ_n. The tabulated data show that as the number of sub-images is increased the accuracy of the calculated σ_n remains fairly constant (up to a limit in the number of sub-images chosen) and restoration time is reduced. Thus σ_e remains close enough to σ_n for SIWM to restore the image data. In addition, increasing the number of sub-images reduces the amount of time required to restore the image. Therefore the two assertions about SIM are validated for these data. However there are a few instances where an increase in the number of sub-images causes a significant increase in the percent difference between σ_n and σ_e. The increase is an indication that there is a limit to the number of sub-images that can be used beyond which the restoration results are significantly degraded. The data also shows that for the case of whole image noise contamination it is not necessary to use the images asi, any sub-image can be used. The difference in results between figures 3.8(d) and 3.8(e) show for the case of partial noise contamination the asi should be used for best restoration results.

3.4 Comparison of the SIWM and MatLab Wiener2

The MatLab function Wiener2, performs a lowpass filter on an image that has been contaminated by constant power additive noise. It uses a pixel-wise adaptive Wiener method based on statistics estimated from a local neighborhood of each pixel and it calculates an estimate of the noise contained in the image data, [16]. Wiener2 estimates the local mean, μ, and variance, σ^2, around each pixel and they are calculated using the following equations:

[3.3], $\mu = \frac{1}{N \cdot M} \sum_{\eta_1, \eta_2 \in \eta} h(\eta_1, \eta_2).$ [3.4], $\sigma^2 = \frac{1}{N \cdot M} \sum_{\eta_1, \eta_2 \in \eta} h^2(\eta_1, \eta_2) - \mu^2.$

In this section the wiener2 function is compared to the SIWM. In figure 3.9 the methods are applied to an image containing with very little contamination, in figure 3.10 a moderate amount of contamination and in figure 3.11 a large amount of distortion.

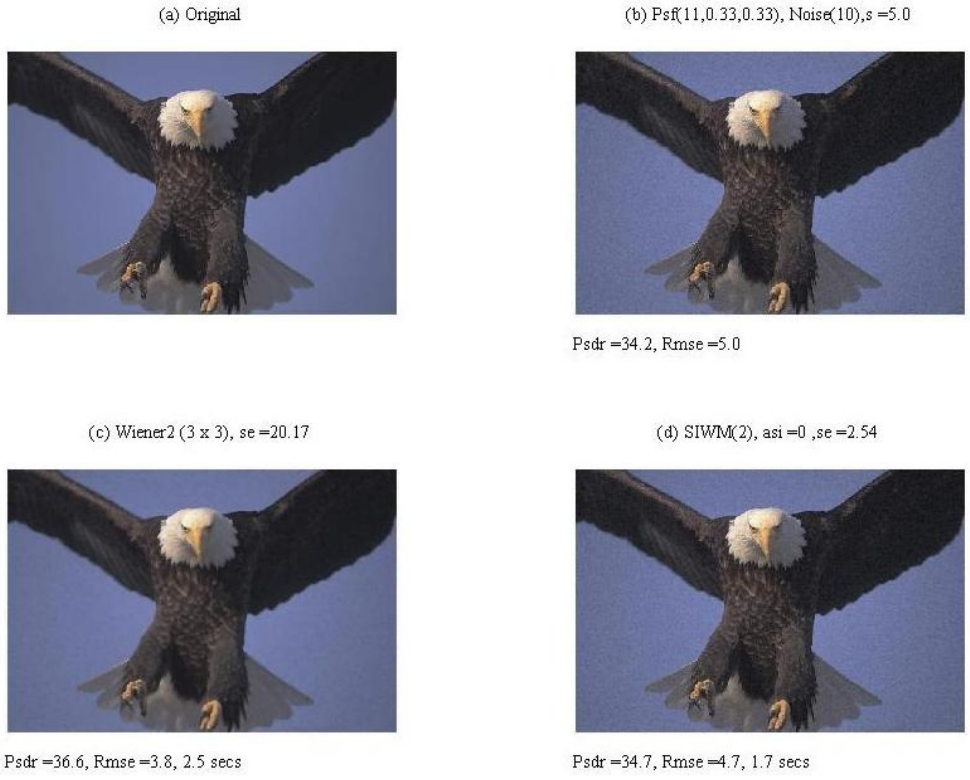


Figure 3.9

(a) Original

(b) Psf(11,1.00,1.00), Noise(30),s =15.6

Psdr =23.6, Rmse =16.9

(c) Wiener2 (3 x 3), se =17.33

Psdr =27.2, Rmse =11.1, 2.6 secs

(d) SIWM(2), asi =0 ,se =15.68

Psdr =28.0, Rmse =10.2, 1.6 secs

Figure 3.10

(a) Original

(b) Psf(11,1.75,1.75), Noise(40),s =21.1

Psdr =19.4, Rmse =27.3

(c) Wiener2 (3 x 3), se =20.89

Psdr =21.9, Rmse =20.4, 1.8 secs

(d) SIWM(2), asi =0 ,se =21.93

Psdr =22.6, Rmse =19.0, 1.2 secs

Figure 3.11

The results shown in figure 3.9 shows that both the SIWM and the Wiener2 methods are capable of restoring images which contain very little distortion, with the Wiener2 method producing slightly better results numerically. Notwithstanding, in my opinion, the SIWM result looks better. Figure 3.9 also shows that the SIWM is faster than the Wiener2 method. Figures 3.10 and 3.11 show that the SIWM does a better job of restoring image data with moderate and large amounts of contamination – and does it faster than the Wiener2 method. Lastly all three of the figures show that the SIWM produces a more accurate estimate of the image contaminating noise standard deviation, σ_n.

Summary and Discussion

In this dissertation six well know image restoration methods have been investigated. Three of the methods are direct and three are iterative. For chapter 1 several contaminated images were processed using all six of the investigated methods. The results are – the Wiener method is the best method (of the six) for the cases investigated. In chapter 2 the Morrison De-Nosing method (MNRM) is described and is shown to be a viable noise reducing method. The MNRM is used to develop a method of estimating the standard deviation of noise, σ_n, contained in contaminated image data, h. The method was tested on the first ten images shown in figure 1.10 and is found to yield good results. The sigma estimation method (SIGEST) is shown in figure 2.4.

In chapter 3, the Sub-Imaging Method (SIM) is introduced and two assertions about it are made:

1. The asi of an image can be used in the place of the whole image to determine σ_e of σ_n and used to restore the whole image. Therefore, the noise in a piece of an image can represent the noise in the whole image (provided it is the asi of the image's hN).
2. SIM can be combined with the Wiener method to restore contaminated image data without the noise σ of the data being known.

An algorithm utilizing SIM and the Wiener method was developed and tested using 10 of the images shown in figure 1.10. The results validate both assertions concerning SIM for these images. However two exceptions were discovered. There is a limit to the number of sub-images that an image can be divided into and have assertion two remain valid. The second is that as shown by figure 3.2 compared to figure 3.3, it is not necessary that the sub-image chosen be the asi of the whole image if the whole image is completely contaminated with noise. If the image is partially contaminated with noise then its asi must be used for best restoration results as shown

by figure 3.8. Additionally the more sub-images used, the less time required by the SIM restoration method.

The first conclusion of this dissertation is that the method SIGEST, based on the MNRM can be used to determine a good estimate of the standard deviation of noise contained in image data, σ_n. The second conclusion is that SIM combined with the Wiener method is a viable restoration method that enables blur and noise contaminated image data to be restored without the noise being known (using σ_n as determined by SIGEST). The final conclusion is that SIM works best as a restoration method when applied to images which contain a moderate amount, and more, of blur and noise contamination. The SIWM is least accurate for narrow width PSF images, so one possible improvement would be to use another filter to restore images with a narrow PSF. Another improvement in the method would be to develop an algorithm that adjusts the sub-image size of 96 by 96 pixels used in the Calc_Asi function (figure 3.1). The sub-image size has a great affect on partially noise contaminated images. I think further research that resolves the two previously mentioned weakness in the SIWM, would substantially improve the method. However I think the SIWM as developed and coded in this work, is an effective image restoration tool. But as with any other method the user must review the results and perhaps reapply the method, changing the inputs as needed. A fairly complete flow chart of the whole SIWM is shown in figure 3.12.

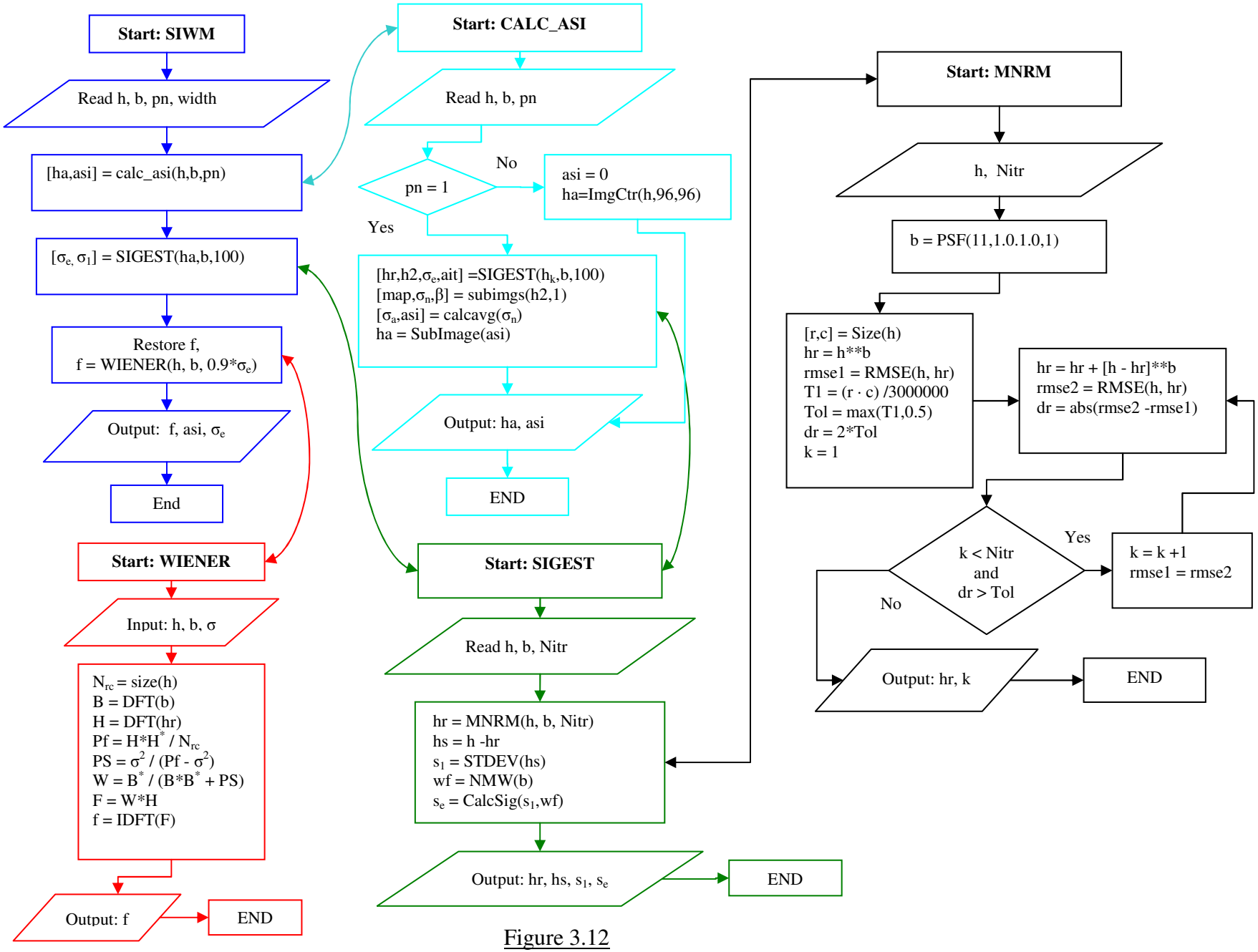

Start: SIWM
Read h, b, pn, width
[ha,asi] = calc_asi(h,b,pn)
[σe, σ1] = SIGEST(ha,b,100)
Restore f,
f = WIENER(h, b, 0.9*σe)
Output: f, asi, σe
End
Start: WIENER
Input: h, b, σ
Nrc = size(h)
B = DFT(b)
H = DFT(hr)
Pf = H*H* / Nrc
PS = σ2 / (Pf - σ2)
W = B* / (B*B* + PS)
F = W*H
f = IDFT(F)
Output: f
END
Start: CALC_ASI
Read h, b, pn
pn = 1
No
asi = 0
ha=ImgCtr(h,96,96)
Yes
[hr,h2,σe,ait] =SIGEST(hk,b,100)
[map,σn,β] = subimgs(h2,1)
[σa,asi] = calcavg(σn)
ha = SubImage(asi)
Output: ha, asi
END
Start: SIGEST
Read h, b, Nitr
hr = MNRM(h, b, Nitr)
hs = h -hr
s1 = STDEV(hs)
wf = NMW(b)
se = CalcSig(s1,wf)
Output: hr, hs, s1, se
END
Start: MNRM
h, Nitr
b = PSF(11,1.0.1.0,1)
[r,c] = Size(h)
hr = h**b
rmse1 = RMSE(h, hr)
T1 = (r · c) /3000000
Tol = max(T1,0.5)
dr = 2*Tol
k = 1
hr = hr + [h - hr]**b
rmse2 = RMSE(h, hr)
dr = abs(rmse2 -rmse1)
k < Nitr
and
dr > Tol
Yes
No
k = k +1
rmse1 = rmse2
Output: hr, k
END

Figure 3.12

BIBLIOGRAPHY

[1] Michael Patrick Allen, Understanding Regression Analysis. Plenum Publishing Corporation, New York, 2004, pages 71-80.

[2] H. C. Andrews and B. R. Hunt, Digital Image Restoration, Prentice-Hall, Inc, New Jersey, 1977, pages 12 – 23 and 211 - 222.

[3] Daniel Shannon Briggs, High Fidelity Deconvolution of Moderately Resolved Sources, PhD thesis, The New Mexico Institute of Mining and Technology, Socorro, New Mexico, March 1995, page 182.

[4] Richard L. Burden and J. Douglas Faires, Numerical Analysis 5th ED., PWS-Kent Publishing Company, Boston, 1993, p 555 to 557.

[5] I. Busko, Wiener Image Restoration, Bulletin of the American Astronomical Society, Vol. 26, 05/1994, pages1012 - 1014.

[6] Kenneth R. Castleman, Digital Image Processing, Prentice-Hall, Inc, New Jersey, 1979, pages 3 – 13 and 250 - 253.

[7] Richard E. Crandall, Projects In Scientific Computation, Springer Verlag New York, Inc, September 1994, pages 205 – 210.

[8] Rafael C. Gonzalez & Richard E. Woods, Digital Image Processing 2nd Edition, Prentice Hall, New Jersey, 2002, pages 289 – 295.

[9] Amara Graps, An Introduction to Wavelets, IEEE Computational Science and Engineering, Summer 1995, vol. 2, num. 2, published by IEEE Computer Society, pages 1 - 15.

[10] N. R. Hill, and George E. Ioup, Convegence of the Van Cittert Iterative method of Deconvolution, Journal. of the Optical Society of America, Vol 66, No 7, pages 487 – 489, 1976.

[11] George E. Ioup and Juliette W. Ioup, Iterative Deconvolution, Geophysics Vol 48, No. 9, Sept. 1983, pages 1287 – 1290.

[12] Juliette W. Ioup, and George E. Ioup, Optimum use of Morrison's iterative noise removal method prior to linear deconvolution, Bulletin America. Physics Soc. Vol. 26, 1981, page 1213.

[13] Peter A. Jansson, Deconvolution of Images and Spectra 2nd ED., Academic Press, Inc., 1997, San Diego, CA.

[14] D. Kundur and D. Hatzinakos, Blind Image Deconvolution, IEEE Signal Processing Magazine, 13, 1996, pages 43-64.

[15] R.L. Lagendijk and J. Biemond, Basic Methods for Image Restoration and Identification, Handbook of Image and Video Processing 2nd edition, ISBN 0-12-119792-1, Elsevier Academic Press, Burlington, MA, 2005, pages 167-181.

[16] Jae S. Lim, Two-Dimensional Signal and Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1990, pages 536-540.

[17] L. D. Marks, Weiner filter enhancement of HREM images, Ultramicroscopy 62, 1996, pages 43-52.

[18] Dwight F. Mix, Kraig J. Olejniczak, Elements of Wavelets for Engineers and Scientists. John Wiley & Sons, New York, Incorporated, 2003, pages 145-157.

[19] William K. Pratt, Digital Image Processing, John Wiley & Sons, New York, 1978, pages 736 – 740.

[20] Fred Ramsey and Daniel Schafer, The Statistical Sleuth: a course in methods of data analysis. Wadsworth Group, Duxbury: Thomson Learning, Inc, 2002, pages 240-250.

[21] A. Segalovitz & B. Roy Frieden, A "CLEAN"-type Deconvolution Algorithm, Astronomy and Astrophysics 70, 1978, pages 335 – 343.

[22] J.L. Starck, Entropy and astronomical data analysis: Perspectives from multiresolution analysis, Astronomy and Astrophysics 368, 2001, pages 730 - 746.

[23] D. G. Steer, P. E. Dewdney & M. R. Ito, Enhancements to the deconvolution algorithm "CLEAN", Astronomy and Astrophysics 137, 1984, pages 159 – 165.

[24] Mitsuyoshi Tomiya and Akihiro Ageishi, Registration of Remote Sensing Image Data Based on Wavelet Transform, ISPRS Archives, Vol. XXXIV, Part 3/W8, Munich, Sept. 2003, pages 17 –19.

[25] Martin Vetterli, Adaptive Wavelet Thresholding for Image Denoising and Compression, IEEE Transactions On Image Processing, Vol. 9, No. 9, Sept. 2000, pages 1542 – 1546.

Computer Code

This section is a listing of all of the major computer code I used in performing the image restoration methods and various parameter calculations in this dissertation. All of the code is written in Matlab version 7.1.

```
function [f,dt] = iddm(oi,b1,inoise,ns,Show)
tic;
% All equations developed involve dividing by b(1,1) so thus,
if abs(b1(1,1)) < 0.001,
  disp('First element of b can not be zero or nearly zero.');
  disp('Will adjust PSF upward by 0.5 and normalize b(1,1) to one.');
  b1 = b1 + 0.5;
end
b   = double(ceil(b1/b1(1,1)));
[h] = blur1(oi,inoise,b,ns,0);  % Blur the given image.
[m3,n3] = size(h); % size of h
[m1,n1] = size(b); % size of b
m2 = m3 -m1 +1;  % number of rows of f
n2 = n3 -n1 +1;      % number of cols of f
f  = double(zeros(m2,n2));
%-------------------------------------------------------
for col = 1:1:n2,
  for row = 1:1:m2,
  %---------------------------------------------
  tot = double(0.0);
  c = 1;        j = min(m1,row);        k = min(n1,col);
    while c <= k,
      r = 1;
            while r <= j,
        tot = tot +b(r,c)*f((row+1-r),(col+1-c));
        r = r +1;
      end
       c = c +1;
        end
  f(row,col) = (h(row,col) -tot) / b(1,1);
    %---------------------------------------------
  end
end
%-------------------------------------------------------
dt = toc;
%-----------------------------------------------------------------
% Plot the images; Original, Contaminated and Restored.
%-----------------------------------------------------------------
if Show == 1,
  if ns > 0,  xtxt2 = sprintf('Blurr +noise');
  else        xtxt2 = sprintf('Blurr');
  end
  xtxt3 = sprintf('Restored; Direct');
  showi3(oi,h,f,dt,'Original',xtxt2,xtxt3);
end
return
```

```
function [f,dt] = ftm(oi,b,inoise,ns,Show)
tic;
[h] = blur1(oi,inoise,b,ns,0);  % Blur the given image.
[r1,c1] = size(b);   % size of b is (r1,c1)
[r3,c3] = size(h);   % size of h is (r3,c3)
r2 = r3 -r1 +1;      % size of image is (r2,c2)
c2 = c3 -c1 +1;
f  = zeros(r2,c2);
row = 1; col = 1; r = 1; c = 1; j = 1; k = 1;
% Use the convolution Thm; Convolution in one domain is
% equal to multiplication in the other domain.
%-----------------------------------------------------------
B  = fft2(b,r3,c3);
H  = fft2(h,r3,c3);
% inverse filter
Filter = (abs(B) > 0.0001) .* (1./B);
f  = ifft2(H.*Filter);    % F = H / B, and f = ifft2(IMAGE)
f  = real(f);
%-----------------------------------------------------------
% Plot the images,
dt = toc;
if Show ~= 0,
   showims(oi,h,f,ns,'Restored; FFT',dt,Show);
end
return
```

```
function [f,dt] = dwiener(oi,b,inoise,ns,sigma)

tic;
[h] = blur1(oi,inoise,b,ns,0);  % Blur the given image.
% Solve for h, by the Wiener Method.
 [r,c] = size(h);
B     = fft2(b,r,c);
H     = fft2(h,r,c);
Pf    = conj(H).*H /(r*c);  % Estimate f = h
Pn    = ones(r,c) * sigma^2;
% Calculate the wiener filter,
PS  = Pn ./ (Pf -sigma^2);
W   = (conj(B)) ./ (conj(B).*B + PS);
% Restore the image.
F   = W.*H;
f   = real(ifft2(F));
dt = toc;
return
```

```
function [f,dt,j] = dvcm(oi,b,inoise,ns,Nitr,Show)

dt = cputime;
[h]  = blur1(oi,inoise,b,ns,0);  % Blur the given image.
[r,c]= size(h);
Nrc   = r * c;
Tol = 0.30;
f     = h;            % Initial guess for the image.
fprintf(1,'Tolerence for stopping = %f\n',Tol);
% Iterate m times using Van Citter's Method.
%-------------------------------------------------------
j   = 1;
rmse1 = 1;
rmse2 = 10 * rmse1;
while (abs(rmse2 -rmse1) > Tol) & (j <= Nitr),
   R  = (h -conv2(f,b,'same'));
   fj = f + R;
   rmse2 = rmse1;
   rmse1 = sqrt(abs(mserror(f,fj)));
   fprintf(1,'%2d, Rmse1 = %.6f, Rmse2 = %f\n',j,rmse1,rmse2);
   j = j +1;
   f = fj;
end
j = j -1;
%-------------------------------------------------------
f  = xtract(oi,f,0);
h  = xtract(oi,h,0);
dt = cputime - dt;
return
```

```
function [f,dt,j] = drlmg(oi,inoise,b,ns,Nitr,Show)

[h] = blur1(oi,inoise,b,ns,0);  % Blur +noise the given image.
dt  = cputime;
[xr,xc] = size(h);
[r2,c2] = size(b);
Nrc     = xr * xc;
Tol = 0.10;
fprintf(1,'Tolerence for stopping = %f\n',Tol);
j     = 1;
rmse1 = 1;
rmse2 = 10 * rmse1;
% For the blurred +noise image data, h;
%============================================================
  fi  = h;          % Initial reference image.
  f2  = h ./2;    % Initial guess of deconvoled image.
  % Determine if 'b' has a zero in it.
  %-----------------------------------------------------
  HasZero = 0;
  for r = 1:1:r2,
    for c = 1:1:c2,
      if abs(b(r,c)) < 0.01,
        HasZero = 1;
      end
    end
  end
  %-----------------------------------------------------

    % Iterate nitr times using Richardson Lucy Method (gaussian noise).
    %-------------------------------------------------------------------
  if HasZero == 0,
  while (abs(rmse2 -rmse1) > Tol) & (j <= Nitr),
    s1 = conv2(f2,b,'same');
    s2 = conv2(s1,b,'same');
    s3 = conv2(h,b,'same') ./ s2;
    f2 = f2 .* conv2(s3,b,'same');
      rmse2 = rmse1;
      rmse1 = sqrt(abs(mserror(f2,fi)));
      fprintf(1,'%d, Rmse1 = %.6f, Rmse2 = %f\n',j,rmse1,rmse2);
    fi = f2;
    j  = j +1;
  end   end
    %-----------------------------------------------
  if HasZero == 1,
  while (abs(rmse2 -rmse1) > Trmse) & (j <= Nitr),
    s1 = conv2(f2,b,'same');
```

```
      s2 = conv2(s1,b,'same');
      [r3,c3] = size(s2);
      s3 = s2;
      for r = 1:1:r3,
        for c = 1:1:c3,
          s3(r,c) = s2(r,c);
           if s2(r,c) == 0,
             s3(r,c) = 1;
           end
         end
       end
      s4 = conv2(h,b,'same') ./ s3;
      f2 = f2 .* conv2(s4,b,'same');
        rmse2 = rmse1;
        rmse1 = sqrt(abs(mserror(f2,fi)));
      fprintf(1,'%d, Rmse1 = %.6f, Rmse2 = %f\n',j,rmse1,rmse2);
      fi = f2;
      j  = j +1;
    end  end
      fprintf(1,'..... Finished! .....\n');
    %--------------------------------------------------------------------
j = j -1;
%f   = f2;
f  = xtract(oi,f2,0);
h  = xtract(oi,h,0);
dt = cputime-dt;
```

```
function [f,R,D,dt,steps] = clean(oi,inoise,b,ns,gamma,Pmax,itr,dn,Show)

[h,noise]  = blur1(oi,inoise,b,ns,0);  % Blur the given image.
[row,col]  = size(h);
[row2,col2]= size(b);
Nrc        = row * col;
rr  = max([floor((row2 -1)/2 +1), 1]);
cc  = max([floor((col2 -1)/2 +1), 1]);
tic;
if (dn > 0) & (ns > 0),  % Do pre de-noising.
   disp('... De-Noising the imagage ...');
   hn = h;
     h = ioupdn(hn,b,dn,0);
end
% Clean beam, bc,
bmax  = max(max(b)) * 1;
bc    = gaussmtx(row2,col2,0.01,0.01,0,0);
R     = h;              % Residual image
D     = zeros(row,col);  % Density image
Tol   = max(max(h)) * Pmax;
Imax  = 2 * Tol;
steps = 0;
%
%-------------------------------------------------------------------
while (Imax > Tol) & (steps < itr),
   [v1,m1] = max(R);      % Row of max values, μ
   [Imax,m2] = max(v1);   % Max pixel value
   j  = m1(m2);           % Location of max pixel, R(j,k)
   k  = m2;
   gp = gamma * Imax;
   r1 = j -rr +1;
   c1 = k -cc +1;
   r2 = j +rr -1;
   c2 = k +cc -1;
   W  = zeros(row,col);
   % Copy center of (gp * b) to W at (j,k).
   W(r1:r2,c1:c2) = gp * b;
   R  = R -W;
   D(j,k) = D(j,k) + gp;
   steps  = steps +1;
end
f  = conv2(D,bc,'same');
f  = D + R;
% Scale max intensity in f to match that in h, the given data.
sf = min(max(max(h)),255) / max(max(f));
f  = f * sf;
```

```
%-------------------------------------------------------------------
dt = toc;
fprintf(1,'------ CLEAN Method finished, iterations = %d ------\n\n',steps);

% Plot the original, contaminated, Residual & Restored images.
%-------------------------------------------------------------------
if Show == 1,
  xtxt2 = sprintf('Psf(%d,%.1f,%.1f) & Noise(%.1f)\r\n',nb,sbx,sby,ns);
  xtxt3 = sprintf('Residual, g =%.2f, tol =%.2f',gamma,Pmax);
  if dn ~= 0,
    xtxt4 = sprintf('CLEAN +PNR, itrs = %d',steps);
  else
    xtxt4 = sprintf('CLEAN, itrs = %d',steps);
  end
  showi4b(oi,h,R,f,dt,dt,'Original',xtxt2,xtxt3,xtxt4);
end

% Plot the original, contaminated, Density & Restored images.
%-------------------------------------------------------------------
if Show == 2,
  xtxt2 = sprintf('Psf(%d,%.1f,%.1f) & Noise(%.1f)\r\n',nb,sbx,sby,ns);
  xtxt3 = sprintf('Density, g =%.2f, tol =%.2f',gamma,Pmax);
  if dn ~= 0,
    xtxt4 = sprintf('CLEAN +PNR, itrs = %d',steps);
  else
    xtxt4 = sprintf('CLEAN, itrs = %d',steps);
  end
  showi4b(oi,h,D,f,dt,dt,'Original',xtxt2,xtxt3,xtxt4);
end
return
```

```
% Function description:
% Blurrs the given 2D B&W image with the given Point Spread Function
% 'psf' and adds noise if ns > 0. Resulting 2D image is returned as h.
% The model is;
%             h = b(*)image + noise
%=================================================================
function [h] = blur1(image,inoise,b,ns,Show)
h = conv2(b,image); % conv2(image,b,'same');
[r,c] = size(h);
[ny2,nx2] = size(inoise);
if (r <= ny2) & (c <= nx2), % Use the given noise if possible.
  noise = inoise(1:r,1:c);
else
  noise = randn(r,c);
end
% Det min and max values of h so that if noise is to be added;
% it is added as a percentage of (max -min).
%-------------------------------------------------------
if ns <= 0,  noise = noise * 0;  end
if ns > 0,
  vmin = min(min(h));
  vmax = max(max(h));
  magnitude = vmax -vmin;
  noise = noise * magnitude * (ns / 100.0);
  h = h + noise;
  % Ensure that all pixel values are in the range of 0 to 255. All values outside of the allowed
  % range must be truncated to 0 or 255, because an imaging system will not record values
  % outside of this range, some detail is lost but in actuality this is what occurs.
  for j = 1:1:r,
    for k = 1:1:c,
      if h(j,k) < 0,   h(j,k) = 0;    end
      if h(j,k) > 255, h(j,k) = 255;  end
    end
  end
end
return
```

```
% Function description:
% Blurrs the given color 2D image with the given Point Spread Function
% 'psf' and adds noise if ns > 0. Resulting 2D image is returned as h.
% The model is;      h = b(*)image + noise
%=============================================================
function [h,noise] = cblur(image,inoise,b,ns,Show)
[hr,hg,hb] = color2rgb(image);
hr = conv2(b,hr);     hg = conv2(b,hg);     hb = conv2(b,hb);
[ny,nx] = size(hr);
[ny2,nx2] = size(inoise);
if (ny <= ny2) & (nx <= nx2), % Use the given noise if possible.
  noise = inoise(1:ny,1:nx);
else
  noise = randn(ny,nx);
end
[rows,cols] = size(hr);
% Det min and max values of h so that if noise is to be added; add as a percentage of (max -min).
%-------------------------------------------------------
if ns <= 0, noise = noise * 0; end
if ns > 0,
  hbw = color2bw(image);
  vmin = min(min(hbw));
  vmax = max(max(hbw));
  magnitude = vmax -vmin;
  noise = noise * magnitude * (ns / 100);
  hr = hr + noise;
  hg = hg + noise;
  hb = hb + noise;
  % Ensure that all pixel values are in the range of 0 to 255. All values outside of the allowed
  % range must be truncated to 0 or 255, because an imaging system will not record values
  % outside of this range, some detail is lost but in actuality this is what occurs.
    for j = 1:1:ny,
      for k = 1:1:nx,
        if hr(j,k) < 0,   hr(j,k) = 0;    end
        if hr(j,k) > 255, hr(j,k) = 255;  end
        if hg(j,k) < 0,   hg(j,k) = 0;    end
        if hg(j,k) > 255, hg(j,k) = 255;  end
        if hb(j,k) < 0,   hb(j,k) = 0;    end
        if hb(j,k) > 255, hb(j,k) = 255;  end
      end
    end
end
h(:,:,1)  = hr;
h(:,:,2)  = hg;
h(:,:,3)  = hb;
return
```

```
% Function description: Morrison Noise Reduction Method
% De-noises the given 2-D image data by the Morrison method:
%   h = b (*) io + noise
% Resulting 2D image is returned as 'hr'.
%   f     = The original (un-contaminated) image data
%   h     = The measured (contaminated) image data
%   bi    = Actual blurring funct, Impulse response (Point Spread Function)
%   bm    = Is the actual PSF used so, bm = b
%   Nitr  = Max number of iters of Morrison method
%   Show  = 1, De-noised image is displayed.
%   Show  = 2, De-noised and Original images are displayed.
%   j     = The actual number of iterations done.
% The algorithm is;
%             hr(j) = hr(j-1) +[(h-hr(j-1))(*)b]
%=========================================================================
function [hr,j,bm] = mnrm(f,h,bi,Nitr,Show)

[r,c]   = size(h);    [rb,cb] = size(bi);   [wf,na] = nmw(bi);
% Morrison method works well using a medium width PSF.
% If the given PSF is narrow or wide then substitute a medium width of the same size.
%-------------------------------------------------------------------
if (wf ~= 2),
   bm = gaussmtx(rb,cb,1.0,1.0,1,0);
else
   bm = bi;
end
tol2  = (r * c) * 2e-007;  % Allow only a little error for each pix.
Tol   = max([tol2, 0.5]);
hr    = conv2(h,bm,'same');
rmse1 = sqrt(abs(mserror(h,hr)));   amin = 0;   j = 1;   xrmse(j) = rmse1;
% Iterate max of Nitr times using Morrison Method to reduce the noise in the given image data.
%-----------------------------------------------------------
while (amin == 0) & (j <= Nitr),
  j  = j +1;
  hr = hr +conv2((h -hr),bm,'same');
  rmse2 = sqrt(abs(mserror(h,hr)));
  % Located a min if set TOL being reached.
  if (abs(rmse2 -rmse1) < Tol),
     amin = 1;
  else
     rmse1 = rmse2;     xrmse(j) = rmse1;
  end
end
% Plot the images.
if Show == 1,     showi2(f,fr,'De-noised image');   end
return
```

```
% Function description: Wavelet Noise Reduction Method
% Reduces the noise in the given image by applying the DWT constructed from,
%   a = The scaling function coefficients, and,
%   b = The wavelet function coefficients
% a and b are calculated for the given Daubechies length N.
% Next the indicated thresholding method is applied as follows;
%   thold = 1 => Soft threshold, and,
%   thold = 2 => Hard threshold
%   N     = Length of wavelet to use.
%   f     = The De-noised image
%=================================================================
function [f] = wnrm(image,N,thold,Show)

[h,y1,x1,y2] = sqimage(image); % Image must be square!
x2 = y2;
[ny1,nx1]    = size(h);
M     = max([ny1,nx1]);
[a,b] = wcdaub(N);
% Construct the DN forward transform matrix;
%-----------------------------------------------------------------
W  = zeros(M,M);
dc = 1;
for r=1:2:(M-1),
  k  = 1;
  c2 = min([(dc+N-1),M]);
  for c=dc:1:c2,
    W(r+0,c) = a(k);
    W(r+1,c) = b(k);
    k = k +1;
  end
  dc = dc +2;
end
%-----------------------------------------------------------------
% Apply W to the given image.
%-----------------------------------------------------------------
F1 = W * h;
%-----------------------------------------------------------------
% Apply the indicated thresholding to F1 of the image.
%-----------------------------------------------------------------
sigma  = std(reshape(h,ny1^2,1));
lambda = sqrt(sigma)*sqrt(2*log10(M));
F2 = zeros(M,M);
if thold == 1,  % Soft thresholding
  for r=1:1:M,
    for c=1:1:M,
      F2(r,c) = sign(F1(r,c))*max([0,(abs(F1(r,c))-lambda)]);
```

```
    end
  end
end
if thold == 2,  % Hard thresholding
  for r=1:1:M,
    for c=1:1:M,
      if abs(F1(r,c)) > lambda,
        F2(r,c) = F1(r,c);
      else
        F2(r,c) = 0;
      end
    end
  end
end
%---------------------------------------------------------------------
% Apply the IWT to F2 to recover the image.
%---------------------------------------------------------------------
f = W' * F2;
f = f(1:y1,1:x1); % Reset f to the orginal image size.
%---------------------------------------------------------------------
```

```
%=============================================================
% Function description: Determines if the matrix data (PSF) width is,
%     wf = 1 => narrow
%     wf = 2 => medium
%     wf = 3 => wide.
%
% EFS,  © 2006
%=============================================================
function [wf,Na] = nmw(A2)

% First normalize A2,
Amax = max(max(A2));
A    = A2 / Amax;
Amax     = max(max(A));
[r1,c1]  = size(A);
Nrc = r1 * c1;
Amax1 = Amax * 0.50;
Na  = 0;
% Determine the number of elements in A >= Amax1.
for r = 1:1:r1,
  for c = 1:1:c1,
    if (A(r,c) >= Amax1),
      Na = Na +1;
    end
  end
end
if Na < 1,
  wf = -1;
  return
end
Rab = Na / Nrc;
if Rab < 0.02,
  wf = 1;   % Narrow PSF
end
if Rab >= 0.02 & Rab < 0.09,
  wf = 2;   % Medium PSF
end
if Rab >= 0.09,
  wf = 3;   % Wide PSF
end
return
```

```
%==========================================================
% [hr,hs,sig1,sigmae,Ait] = sigest(h,b,Nitr)
% ENAS 7050
% Image Deconvolution Research - IV
%
% Function description: Sigma Estimation Method
% Calculates an estimate of the noise in 2-D image data h.
%   h     = The noise contaminated image data
%   b     = The PSF of the image data
%   Nitr  = Max number of iterations to use in Morrison De-Noising method
% Returns:
%   hr    = Reduced noise version of h
%   hs    = The estimated noise data in h
%   sigmae = Estimated standard deviation of noise in the h
%   sig1   = The un corrected est std of the noise in h
%   Ait    = Actual number of iterations used in MDNM
%
% EFS,  © 11/2006
%==========================================================
function [hr,hs,sig1,sigmae,Ait] = sigest(h,b,Nitr)

[r1,c1]  = size(h);
[hr,Ait] = mnrm(h,h,b,Nitr,0);
hs       = (h -hr);
sig1     = std(reshape(hs, 1, (r1 * c1)));
sigmae   = sigact(sig1,b);
return
```

```
%=============================================================
% function [siga] = sigact(sige,psf)
% ENAS 7050
% Image Deconvolution Research - IV
% Function description: Calculates the actual noise sigma, siga, given
% the estimated noise sigma, sige, and the image PSF data.
%
% EFS,  © 11/2006
%=============================================================
function [siga] = sigact(sige,psf)

% First determine if the PSF is narrow, medium or wide width.
[wf,Naa] = nmw(psf);
siga = sige;
if wf == 1,  % Narrow width PSF.
   siga = -0.9008 +0.3404*sige +0.0947*sige^2 -0.0015*sige^3;
end
if wf == 2,  % Medium width PSF.
   siga = -0.1501 +1.3141*sige +0.0076*sige^2 +0.0003*sige^3;
end
if wf == 3,  % Wide width PSF.
   siga = 0.0824 +1.2517*sige +0.0099*sige^2 +0.0003*sige^3;
end
siga = siga * 1.00;  % Final adjustments to sigma can be made here.
return
```

```
%=============================================================
% [ha,asi,uniform,rows,cols] = calc_asi(h,b,pn,show);
% Function description:
% Calculates the asi of the given noise contaminated image data, h, return;
%   ha      = The sub-image of h for asi
%   asi     = The index to the average sub-image
%   uniform = 0 => Noise is uniform else is not uniform.
%
% For a 3 by 3 the sub-imaging the indexing is as follows;
%    | 1 | 2 | 3 |
%    | 4 | 5 | 6 |
%    | 7 | 8 | 9 |
% EFS,  © 11/2006
%=============================================================
function [ha,asi,uniform,rows,cols] = calc_asi(h,b,pn,show)

% Calculate hN from h2 (h2 is calc'ed in sigest), (2 iterations of the Morrison Method) of h.
[hr,hN,sig1,sigmae,Ait] = sigest(h,b,2);
if pn == 0,
% Partition the image into 1 center sub-image.
   rows = 1;       cols = 1;   uniform = 0;
   [ha,hac,alpha,asi] = imgcenter(hN,hN,0);
else
%-------------------------------------------------------------------
% Partition the image into sub-images of (rows, cols).
   [map,alpha,beta,rows,cols] = subimgs(hN,0);
   % Calc the average std of hN, and the asi to it.
   [av,asi] = calcavg(alpha);
   if show == -1,
     [av,asi] = calcavg3(alpha);  % min
   end
   if show == -2,
     [av,asi] = calcavg2(alpha);  % max
   end
   r1 = map(asi,1);   c1 = map(asi,2);   r2 = map(asi,3);   c2 = map(asi,4);
   ha = h(r1:r2,c1:c2);
   % Determine if the noise is uniform.
   smin = min(alpha);      smax = max(alpha);
   stol = 0.1 * av;
   sdel = abs(smax -smin);
   if sdel > stol,      uniform = 1;
   else                 uniform = 0;
   end
%-------------------------------------------------------------------
end
return
```

```
%=============================================================
% [f1,f2,hc,sigma,asi,dt,ait,rc] = siwm(oi,inoise,nb,sbx,sby,ns,pn,Show)
% Function description: By the Sub Image Wiener Method
% Blurrs the original 2D image 'oi' and adds 'noise' to it by;
%   h = b(*)f + noise
% Deconvolves color f from blurred and noisy (ns ~= 0), h.
% Resulting 2D image is returned as 'f'.
%   oi    = The original image
%   inoise= Input noise matrix
%   f1 =f = Calculated original image based on estimated sigma
%   f2    = Calculated original image based on actual sigma
%   hc    = The blurred image
%   b     = Blurring funct, impulse response (Point Spread Function)
%   ns    = Indicates if h contains noise, ns = 1.
%   Show  = 1, Original, Contaminated, Restored SIIWM,
%           Restored with actual noise sigma, PSNR, RMSE and sigma.
% EFS
% © 2006
%=============================================================
function [f1,f2,hc,sigma,asi,dt,ait,rc] = siwm(oi,inoise,nb,sbx,sby,ns,pn,Show)

% Create the PSF.
b = gaussmtx(nb,nb,sbx,sby,1,0);
[row2,col2] = size(b);
[hc,noise] = cblur(oi,inoise,b,ns,0);  % Blur the given image.
dt(1) = cputime;
[r,c,z1] = size(hc);
%Nsi = rows * cols;  % Calc the number of sub-images.
if ns ~= 0,
   sigma(3) = std(reshape(noise,1,r*c));   % Actual noise sigma(3).
else
   sigma(3) = 0;
end
% Partition the image into sub-images of (rows, cols), and determine
% the asi of the h2 of the image data. Each image color contains the
% same amount of noise, only 1 color is needed. Arbitrarily red is used.
[ha,asi,uniform,rc(1),rc(2)] = calc_asi(hc(:,:,1),b,pn,Show);
%-------------------------------------------------------------------

% Calculate the estimated noise std and number of Morrison iterations
% for sub-image ha.
%-------------------------------------------------------------------
[hr,hs,sigma(1),sigma(2),ait] = sigest(ha,b,100); % <== More sub-images saves time here!
if uniform == 0,
  fprintf(1,'Noise is uniform.\n');
else
```

```
  fprintf(1,'Noise is not uniform.\n');
end
fprintf(1,'M(%d), sigma act = %.4f, est = %.4f, one = %.4f\n',...
    ait,sigma(3),sigma(2),sigma(1));
%---------------------------------------------------------------------

% Determine if the PSF is narrow (wf == 1) or not (wf ~= 1).
%---------------------------------------------------------------------
[wf,Na] = nmw(b);

% Restore the image using the the est one sigma(1), by the Wiener Method.
%---------------------------------------------------------------------
for rgb =1:1:3,
    f1(:,:,rgb) = dwiener2(hc(:,:,rgb),b,sigma(2)*0.9);
end
dt(1) = cputime -dt(1);
%---------------------------------------------------------------------

% Restore the image using the the actual sigma(3), by the Wiener Method.
%---------------------------------------------------------------------
dt(2) = cputime;
for rgb =1:1:3,
    f2(:,:,rgb) = dwiener2(hc(:,:,rgb),b,sigma(3));
end
dt(2) = cputime -dt(2);

%---------------------------------------------------------------------
% Resize h, f1 and f2 to be the same size as the original image 'oi'.
% Size(h) =  Size(oi) + Size(b) -1.
%---------------------------------------------------------------------
for rgb =1:1:3,
    hc1(:,:,rgb) = xtract(oi(:,:,rgb),hc(:,:,rgb),0);
   f11(:,:,rgb) = xtract(oi(:,:,rgb),f1(:,:,rgb),1);
   f21(:,:,rgb) = xtract(oi(:,:,rgb),f2(:,:,rgb),1);
end
hc = hc1;   f1 = f11;   f2 = f21;
fprintf(1,'---------- Image Restoration Done ----------\n\n');
```

VITA

The author Eric F. Smith was born in Tylertown, Mississippi, to Willie and Sarah Smith on September 16, 1965. He graduated from John F. Kennedy High School in New Orleans, Louisiana in 1983. He earned his first degree in Mechanical Engineering from the University of New Orleans (UNO) in 1988. After graduating with a degree in Engineering he worked as a HVAC Engineer in the ship building industry for ten years. While working as an Engineer he earned his second degree in Electronics from Delgado Community College in 1996. He is a certified TV and Radio Technician, and repairs TV's as a hobby. In 1999 he left his engineering job to work as a math teacher in the New Orleans Public School System for about half a year. In 2000 he started working as a Computer Programmer. As a programmer he wrote scientific applications in the areas of mechanics and heat transfer. While employed as a programmer he earned a master's degree in Applied Physics from UNO in 2002. He has recently returned to the engineering field working as an Engineer for a local consulting firm. In the spring of 2003 he began his doctoral studies at UNO in Engineering and Applied Science, specifically in Applied Physics. His current graduate studies are concentrated in the area of computational physics. He is currently doing research in the area of numerical image restoration and enhancement.