ABSTRACT

MATHEMATICAL ASPECTS OF IMAGE PROCESSING

by

Samantha Kirk

April, 2014

Chair: Dr. Gail Ratcliff

Major Department: Mathematics

In this thesis, image processing is explored from a mathematical point of view. After defining a digitized image, techniques for adjusting resolution are discussed. Image transformations defined on a neighborhood centered about a pixel and their relationships to convolution are considered. The Fourier transform and the discrete Fourier transform are introduced in both one and two dimensions. Properties of the Fourier transform are demonstrated with analysis of the power spectrum of an image. A degradation model is used to study image restoration, in the cases where distortion is due to noise and motion blur. Other approaches to image restoration employ the processes of inverse and Wiener filtering.

MATHEMATICAL ASPECTS OF IMAGE PROCESSING

A Thesis

Presented to

The Faculty of the Department of Mathematics

East Carolina University

In Partial Fulfillment

of the Requirements for the Degree

Master of Arts in Mathematics

by

Samantha Kirk

April, 2014

MATHEMATICAL ASPECTS OF IMAGE PROCESSING

by

Samantha Kirk

APPROVED BY:

DIRECTOR OF THESIS:

Dr. Gail Ratcliff

COMMITTEE MEMBER:

Dr. Chris Jantzen

COMMITTEE MEMBER:

Dr. Guglielmo Fucci

COMMITTEE MEMBER:

Dr. Alexandra Shlapentokh

CHAIR OF THE DEPARTMENT
OF MATHEMATICS:

Dr. Johannes Hattingh

DEAN OF THE
GRADUATE SCHOOL:

Dr. Paul Gemperline

ACKNOWLEDGEMENTS

I would especially like to thank my advisor Dr. Ratcliff for her guidance, patience, and the opportunity to learn something new together. I would like to thank Dr. Jantzen, Dr. Shlapentokh, and Dr. Fucci for donating their time to help me improve my writing. Finally, I would like to thank my family for their support and encouragement throughout my study.

TABLE OF CONTENTS

1 An Introduction to Image Processing 1

1.1 Defining an Image 2

1.2 The Resolution of a Digitized Image 3

2 Image Enhancement in the Spatial Domain 7

2.1 Point Processing 7

2.1.1 Image Negatives 8

2.1.2 Thresholding 9

2.1.3 Gamma Correction 10

2.1.4 Histogram Equalization 12

2.2 Spatial Filtering 15

2.2.1 Convolution 16

2.2.2 Convolving Two Images 18

2.2.3 Smoothing Masks 21

2.2.4 Sharpening Masks 24

3 Image Enhancement in the Frequency Domain 31

3.1 The Fourier Transform 31

3.2 The Discrete Fourier Transform 33

3.3 Fast Fourier Transform 35

3.4 Properties of the Fourier Transform and the Power Spectrum 37

3.4.1 Viewing the Power Spectrum 38

3.4.2 The Translation Property 40

3.4.3 The Rotation Property 41

3.4.4 The Wavelength Property 43

3.5 Filtering in the Frequency Domain 46

4 Image Restoration 49

4.1 Types of Noise 49

4.2 Noise Removal 52

4.3 Circulant and Block-Circulant Matrices 54

4.4 Diagonalization of Circulant Matrices 58

4.5 Diagonalization of Block-Circulant Matrices 61

4.6 Inverse Filtering 62

4.7 Wiener Filter 64

MATLAB codes 68

References 71

CHAPTER 1: An Introduction to Image Processing

How do we represent an image mathematically? How do we convert from a continuous image to a digital image taken with a camera? What are some transformations we can apply to an image to enchance it? In this thesis, we will answer these questions and others that occur when working in image processing. In Chapter 1, we start by defining an image and learn how to create a digital image. We then look at how to adjust the resolution (or quality) of a digital image by manipulating its dimensions and gray values.

In Chapter 2, we apply various transformations to an image. The first transformations we consider are ones that depend only on the gray values. Some of these transfomations include finding the negative of an image, using gamma correction to brighten an image, and taking a gray valued image into a black and white image. Next, we extend to transformations that involve a neighborhood of pixels. These transformations are called spatial filtering and can be done through the use of convolution. After we define the convolution of two images, we explore filtering techniques that lead to blurring, finding edges, and sharpening an image.

In Chapter 3, we apply our filtering techniques to the frequency domain. We open this chapter with a review of the continuous Fourier transform and some of its properties. We then discuss the discrete Fourier transform and the relationship between the continuous and discrete versions. Because of the amount of computation the discrete Fourier transform takes we also consider the Fast Fourier transform which image processing programs like MATLAB use to reduce calculation time. Next, we consider properties of the Fourier transform and the power spectrum. Finally, we apply some filters from the spatial domain to the frequency domain.

In Chapter 4, we consider images that are corrupted with blurring or noise and

techniques to restore them to their original quality. First, we give a degradation model that describes how an image is corrupted. We then look at images where only noise has been added. We find that if we have some knowledge about the type of noise then we can make decisions on which filtering technique to use. Next, we consider images with both blur and noise. We assume that the blurring is due to a convolution operator and, by writing the degradation model in matrix notation, we are led to filtering techniques such as inverse filtering and Wiener filtering that can help us find an estimate for the original image.

1.1 Defining an Image

A black-and-white image is a two-dimensional, real-valued function, denoted by f, where the value of $f(x, y)$ gives the intensity (or brightness) of the image at a point in space. Since light is a form of energy we have $0 \leq f(x, y) < \infty$. For an image to be represented in a camera f must be sampled both spatially and in amplitude. Digitization of the spatial coordinates is called "image sampling" and digitization of amplitude is called "gray-level quantization" [3].

Let $D = \{0, 1, ..., N-1\} \times \{0, 1, ..., M-1\}$ be the grid for the image and let $V = \{0, 1, ..., G-1\}$ be the range of gray values. A digital image $f : D \to V$ is created by taking equally spaced samples from an image and takes the form of an $N \times M$ matrix where

$$f = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & ... & f(0,M-1) \\ f(1,0) & f(1,1) & f(1,2) & ... & f(1,M-1) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ f(N-1,0) & f(N-1,1) & f(N-1,2) & ... & f(N-1,M-1) \end{bmatrix}.$$

Each element of the matrix is referred to as a **pixel**.

The resolution (or the amount of detail the image holds) depends on the number of samples that are taken from the original image and the number of gray values. Most images let $V = \{0, 1, 2, ..., 255\}$ which gives an image a total of 256 gray values ranging from 0 (black) to 255 (white). Another common grayscale is to allow 256 gray values to range between 0 (black) and 1 (white).

Color images are composed of three distinct images (or three two-dimenisional arrays) where each image represents the intensity of one of three different colors. For cameras and computer monitors the three colors used are red, green, and blue (the primary colors of light). These images are referred to as RGB images. Color printers use the secondary colors of light (or the primary colors of pigments) cyan, magenta, and yellow and are called CMY images.

We will focus our attention on black and white images.

1.2 The Resolution of a Digitized Image

The process of storing a digitized image requires decisions to be made about the values of N, M, and G. Common practice is to let these quantities be integer powers of two. That is,

$$N = 2^n, M = 2^m, G = 2^g$$

where $n, m, g \in \mathbb{Z}^+$. One bit of memory can store one of two choices, either 0 or 1. Thus the number of bits required to store an image is given by

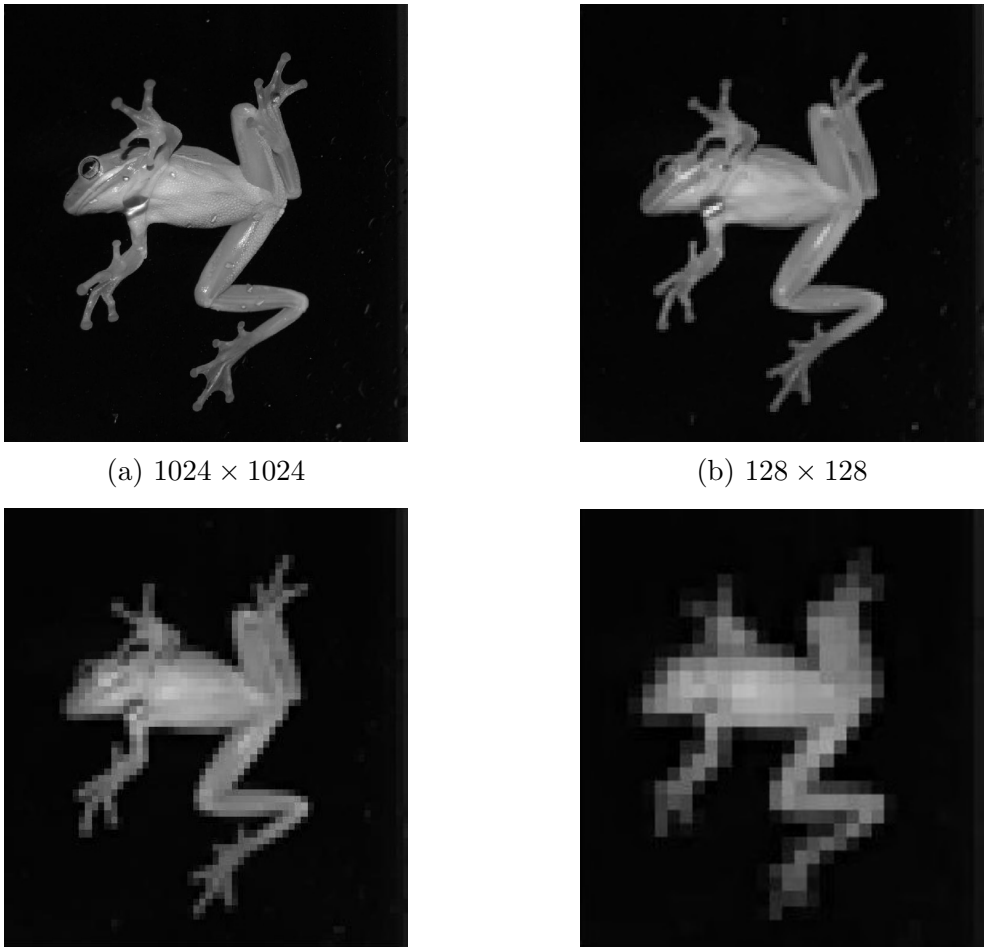
$$b = N \times M \times g.$$

Since computers have a limited amount of storage space it is important to use the least amount of memory possible while saving the image's resolution. To reduce the

amount of storage space we can change the dimensions of the image or we can change the number of gray values.

Figure 1.1 explores what happens when the dimensions of an image are changed to save space. In (a) we have a 1024×1024 pixel image of a frog with 256 gray values. This figure takes up 8,388,608 bits (or 1,048,576 bytes) of space. If we change the dimensions to 128×128 pixels then we have an image that takes 131,072 bits (or 16,384 bytes) of space. This change significantly decreases the amount of space used by the image (1/63 of the original) yet the reduction in size results in a loss of detail and an image that is an eighth of its original size. In (b) we have rescaled the 128×128 image in order to compare its quality to the original. To resize the image, pixels must be replicated to fill in the added columns and rows of the array. This process produces a "checkerboard effect" [3] which becomes more pronounced if we reduce our original image's dimensions to 64×64 pixels (c) or 32×32 pixels (d).

Figure 1.1



(a) 1024×1024 (b) 128×128

(c) 64×64 (d) 32×32

Another way to save memory is to change the number of gray values. In Figure 1.2 (a) we have a 400×400 rose with 256 gray values. In (b) we have reduced the number of gray values to 32 while keeping the dimensions of the image the same. This reduction has saved us 48,000 bytes of space. However, now there is an insufficient number of gray values to represent the smooth transitions in the image. The resulting jumps in gray values leave us with an effect called false contouring [3]. In (c) and (d) we have reduced the gray values to 8 and 4 which makes false contouring more noticable.

Figure 1.2

(a) 256 gray values

(b) 32 gray values

(c) 8 gray values

(d) 4 gray values

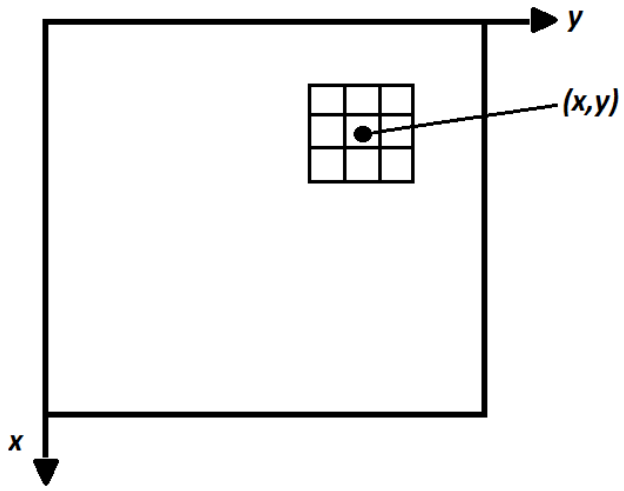
CHAPTER 2: Image Enhancement in the Spatial Domain

In this chapter we consider image transformations in the spatial domain. This refers to operations that directly affect the pixels of an image. Spatial domain processes are denoted by the expression

$$g = Tf \tag{2.1}$$

where f is the input image, g is the output image, and T is an operator on f defined over a specific square neighborhood centered about each pixel (x, y).

Figure 2.1



2.1 Point Processing

We begin by considering intensity transformations where T operates on a neighborhood of size 1×1 centered about the pixel (x, y). That is, the value of g at (x, y) depends only on the intensity value of f at (x, y) and no other pixels around it. Thus

we can simplify the equation in (2.1) to

$$s = T(r)$$

where r is the intensity of f and s is the intensity of our output image g.

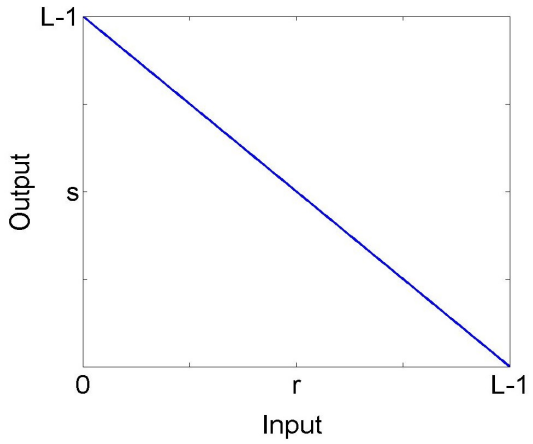
2.1.1 Image Negatives

Suppose we have an image f with L gray values. That is, let $r = 0, 1, ..L-1$. The negative of an image is found by using the transformation function

$$s = (L-1) - r$$

as shown in Figure 2.2 (a). If our image has 256 gray values then $s = 255 - r$. In Figure 2.2 (b) we have an image of a train with its negative (c).

Figure 2.2



(a)

(b)

(c)

2.1.2 Thresholding

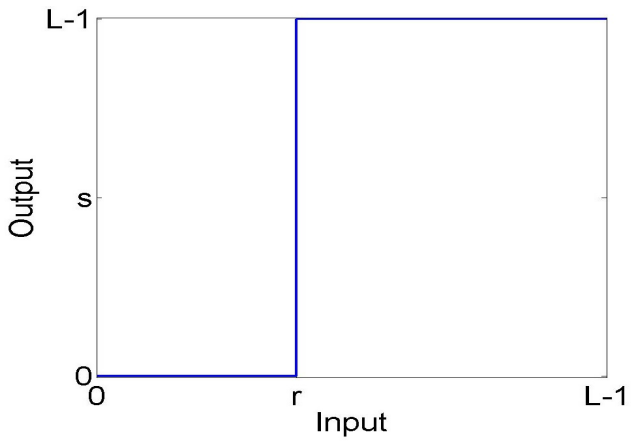
Thresholding is taking a grayscale image and converting it to an image with only two gray values (black and white). To threshold an image we choose a value $m \in [0, L-1]$ where

$$s = \begin{cases} L-1 & \text{if } r \geq m \\ 0 & \text{if } r < m \end{cases}.$$

In Figure 2.3 (a) we have the graph of the thresholding function and in (b) we have a parrot with 256 gray values. If we choose $m = 102$ then all gray values greater than 102 are mapped to white and all gray values less than 102 are mapped to black. We

can see the results of thresholding with the given m in (c). Thresholding can also be used to find edges of objects.

Figure 2.3



(a)

(b)

(c)

2.1.3 Gamma Correction

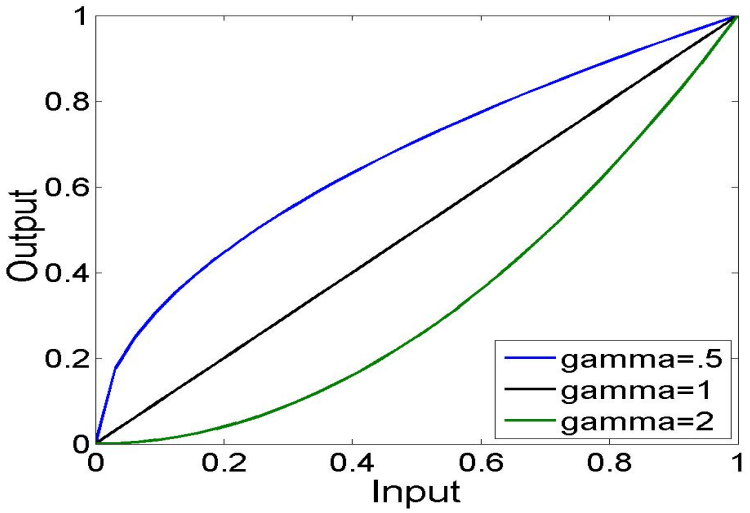
Sometimes we have an image that has been overexposed (too bright) or underexposed (too dark) to light. Gamma correction is a nonlinear intensity transformation that

allows us to change the brightness of an image. Assuming that are gray-values range from 0 to 1, the formula for gamma correction is:

$$s = cr^{\gamma}$$

where c is a constant and $\gamma > 0$. By adjusting the gamma value we can manipulate the contrast of images. In Figure 2.4 we explore what happens when $c = 1$ but γ changes. When $\gamma = 1$ we get the identity transformation $s = r$. Thus the input image is the same as the output image. When $\gamma = 0.5$ we have $s = r^{0.5}$ which increases all gray values and hence lightens the image. When $\gamma = 2$ we have $s = r^2$ and the gray values are decreased.

Figure 2.4



In Figure 2.5 (a) we have an underexposed image of a sailboat. If we let $\gamma = 0.5$, as in (b), our result is a brighter image that is much easier to see. However, if we let $\gamma = 2$ we get a result (c) that is much darker than our original image.

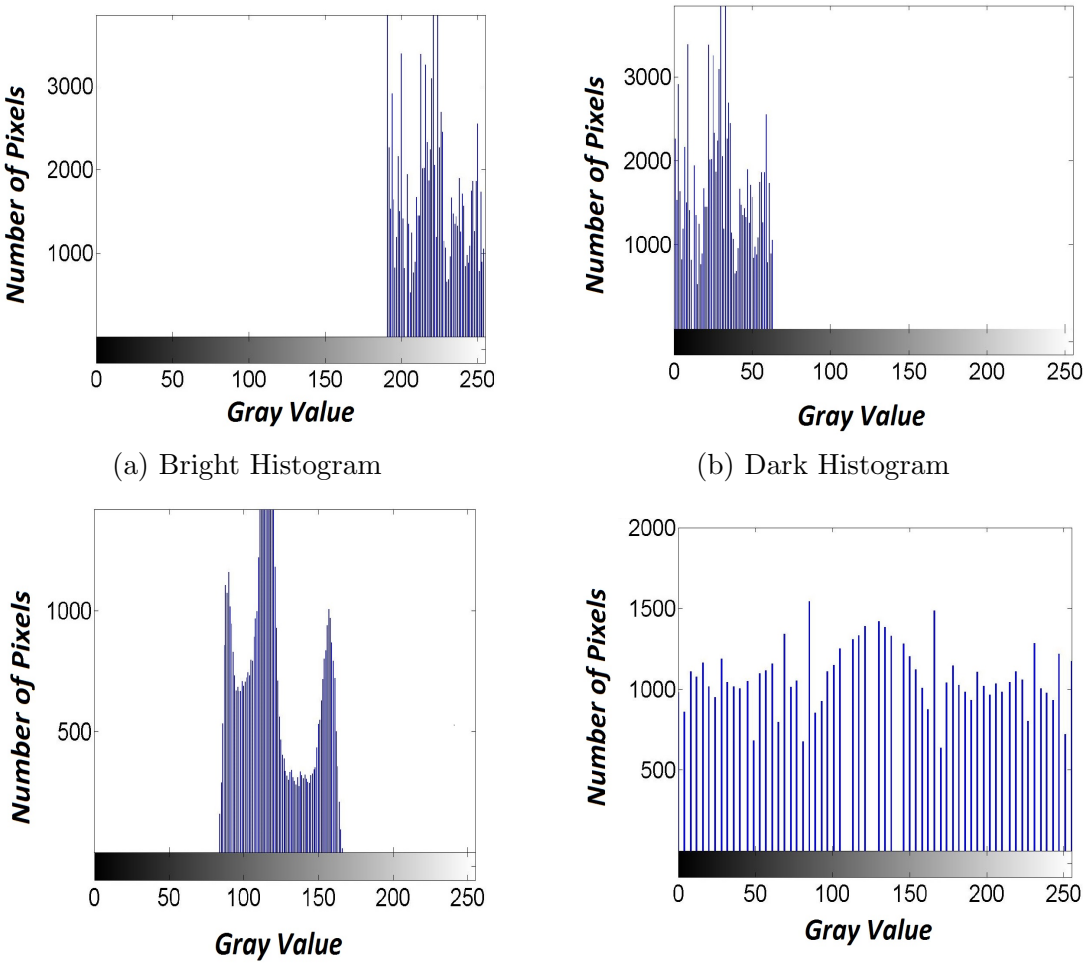
Figure 2.5

(a) Original image (b) γ= 0.5 (c) γ=2

2.1.4 Histogram Equalization

A histogram is a graph that shows the distribution of gray levels in an image. Histograms fall into four basic categories: dark, bright, low-contrast, or high-contrast as shown in Figure 2.6. Histogram equalization is a way to stretch a dark, bright, or low-contrast histogram into a high-contrast histogram.

Figure 2.6



(a) Bright Histogram

(b) Dark Histogram

(c) Low-Contrast Histogram

(d) High-Contrast Histogram

Let us start by assuming that T is a monotonically increasing continuous function from $[0, 1]$ to $[0, 1]$. Since $s = T(r)$ we can write $r = T^{-1}(s)$. If we are given a random variable X with probability distribution function f then the probability distribution function of $T(X)$ is

$$q(s) = p(r) \left| \frac{dr}{ds} \right| .$$

Consider the transformation function $s = \int_0^r p(w)dw$. Then

$$\frac{ds}{dr} = \frac{d}{dr}\int_0^r p(w)dw = p(r).$$

Thus

$$q(s) = p(r)\left|\frac{dr}{ds}\right| = p(r)\left|\frac{1}{p(r)}\right| = 1$$

which is a uniform density distribution. Therefore if we find the transformation function equal to the cumulative distribution function of X we will be able to produce a probability density function that is uniform [3].

For the discrete case we can perform histogram equalization by letting

$$p(r) = \frac{n(r)}{N}$$

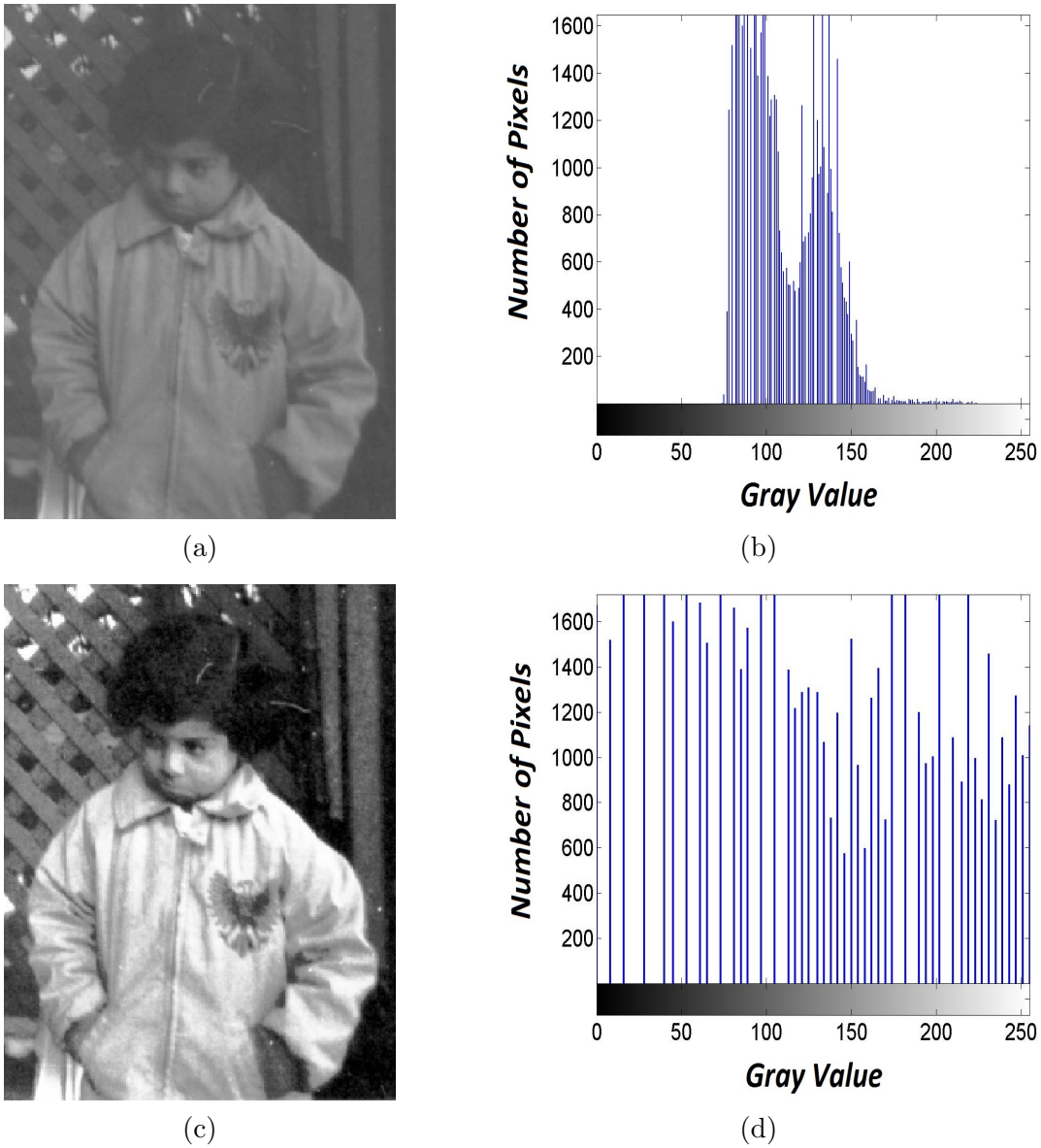
where $r \in \{0, 1, ..., L-1\}$, $n(r)$ is the number of occurrences of the gray level r and N is the total number of pixels. Then the cumulative distribution function is given by

$$s = T(r) = \sum_{j=0}^{k} p(r)$$

where $k = 0, 1, ..., L-1$.

In Figure 2.7 (a) we have a low-contrast image of a boy and in (b) we have the histogram of the image. When histogram equalization is applied to the image we obtain the result in (c). We can see in (d) that histogram equalization has stretched the orginal histogram and created an image which is more balanced and easier to see. Also notice that (d) is not quite uniform due to the discrete nature of the function [4].

Figure 2.7



2.2 Spatial Filtering

We now consider an operator defined on an 8-neighborhood centered about the pixel (x, y) as shown in Figure 2.8. The linear operations in this section consist of multiplying each pixel in the neighborhood by a defined coefficient and then summing the

results to obtain a response at the point (x, y). These coefficients are arranged in a matrix called a **mask** or **filter**.

Figure 2.8

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

To perform spatial filtering we apply the mask to the original image through convolution. Let us begin by defining convolution for continuous and discrete functions.

2.2.1 Convolution

The convolution of two continuous functions f and g is given by

$$(f * g)(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da.$$

For example, let

$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \text{ and } g(x) = \begin{cases} \frac{1}{2} & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}.$$

Then

$$\begin{aligned}(f*g)(x) &= \int_{\infty}^{0} f(a)g(x-a)da + \int_{0}^{2} f(a)g(x-a)da + \int_{1}^{\infty} f(a)g(x-a)da \\ &= \int_{\infty}^{0} 0da + \int_{0}^{1} f(a)g(x-a)da + \int_{2}^{\infty} 0da \\ &= \int_{0}^{1} f(a)g(x-a)da.\end{aligned}$$

Now when $0 < x \leq 1$ we have

$$\int_{0}^{1} f(a)g(x-a)da = \int_{0}^{x} f(a)g(x-a)da = \frac{1}{2}a|_{0}^{x} = \frac{1}{2}x.$$

When $1 < x \leq 2$ we have

$$\int_{0}^{1} f(a)g(x-a)da = \int_{1}^{x-1} f(a)g(x-a)da = \frac{1}{2}a|_{x-1}^{1} = -\frac{1}{2}x + 1.$$

Thus

$$(f*g)(x) = \begin{cases} \frac{1}{2}x & 0 < x \leq 1 \\ -\frac{1}{2}x + 1 & 1 < x \leq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Now let f and g be two periodic functions on a domain of size M. The discrete convolution of f and g is given by

$$(f*g)(x) = \sum_{m=0}^{M-1} f(m)g(x-m)$$

for $x = 0, 1, ...M-1$.

If we were to take M samples from the functions

$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases} \text{ and } g(x) = \begin{cases} \frac{1}{2} & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

and convolve them we would see that discrete and continuous convolution are basically the same with the exception that displacements take place which correspond to the space between the samples taken and that summation replaces integration [3].

We can extend both continuous and discrete convolution into two-dimensions. For the continuous case we have

$$(f * g)(x, y) = \int_{\mathbb{R}^2} f(a, b)g(x - a, y - b)da\, db$$

and for the discrete case we have

$$(f * g)(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)g(x - m, y - n).$$

Now we are ready to consider the convolution of two images f and g.

2.2.2 Convolving Two Images

Convolution of two images is the same as convolving two discrete periodic functions f and g on a domain of size M in the x direction and size N in the y direction. In spatial filtering we call the function g the filter. Since f and g are periodic we can compute $(f * g)(1, 1)$ by repositioning g in such a way that $g(0, 0)$ is in the $(1, 1)$ spot. Figure 2.9 shows an example of repositioning g when $M = N = 4$.

Figure 2.9

g(0,0)	g(0,1)	g(0,2)	g(0,3)	g(0,4)
g(1,0)	g(1,0)	g(1,2)	g(1,3)	g(1,4)
g(2,0)	g(2,1)	g(2,2)	g(2,3)	g(2,4)
g(3,0)	g(3,1)	g(3,2)	g(3,3)	g(3,4)
g(4,0)	g(4,1)	g(4,2)	g(4,3)	g(4,4)

(a) g

g(-1,-1)	g(-1,0)	g(-1,1)	g(-1,2)	g(-1,3)
g(0,-1)	**g(0,0)**	g(0,1)	g(0,2)	g(0,3)
g(1,-1)	g(1,0)	g(1,1)	g(1,2)	g(1,3)
g(2,-1)	g(2,0)	g(2,1)	g(2,2)	g(2,3)
g(3,-1)	g(3,1)	g(3,2)	g(3,3)	g(3,4)

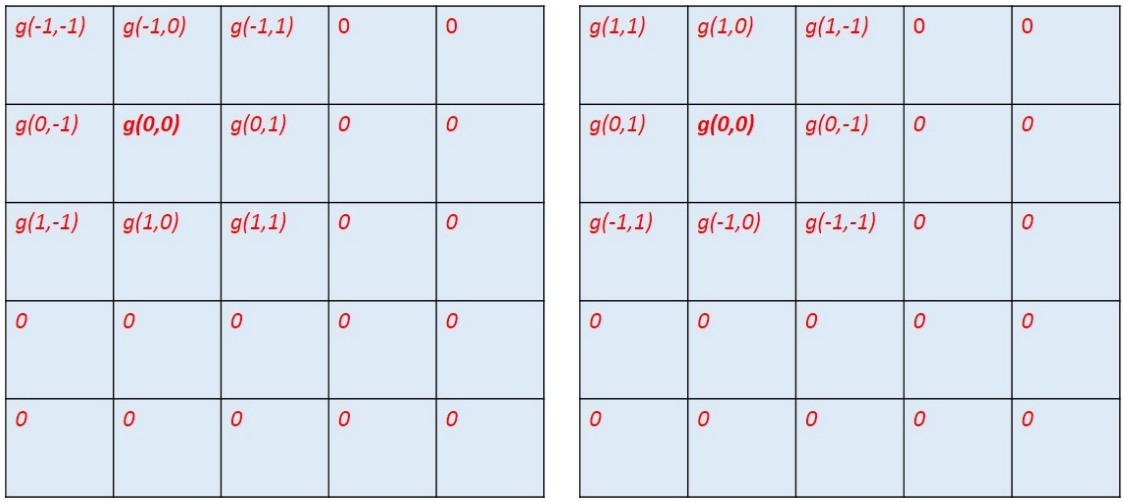
(b) Repositioned g

Our focus will be when g has only 9 non-zero entries arranged in a 3×3 grid as shown in Figure 2.10 (a). For 2-D convolution we have:

$$w(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n)g(x-m, y-n) \tag{2.2}$$

Now to convolve f with g we need to first find $g(x-m, y-n)$. This corresponds to a flip in both the x and y direction. Notice that we can use the periodicity of g again to obtain the result shown in Figure 2.10 (b).

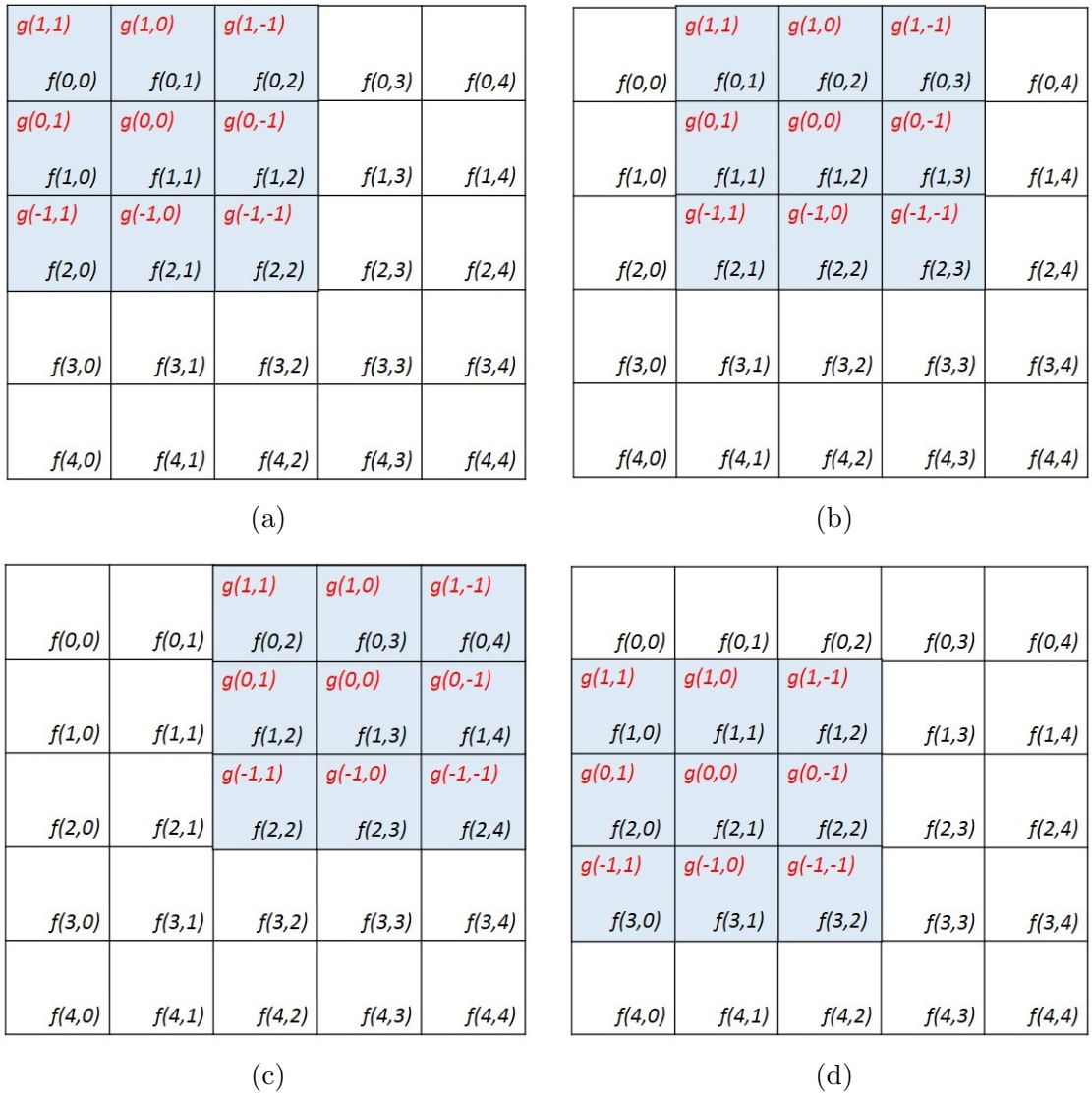
Figure 2.10



(a) (b)

Now to convolve the two functions we place our mask g on top of our image f in such a way that $g(0,0)$ is on top of $f(1,1)$. We will then multiply all numbers that are on top of one another and add them together. Our result will be the entry in our output image $w(1,1)$. Notice that since g has only 9 non-zero entries we do not need to worry about the rest of g multiplying f. We can just focus our on attention on the 3×3 grid (Figure 2.11 (a)). By using the periodicity of g we can shift our non-zero entries one unit to the right as shown in (b). We then multipy the numbers that overlap, add them together, and put our result in $w(1,2)$. We continue to shift our mask to the right one unit at a time, multiplying overlapping entries together and summing them, until we get to the end of the first row. We then shift our mask's nonzero entries down one row, where $g(0,0)$ is placed on top of $f(2,1)$, and repeat the process until we reach the end of the image.

Figure 2.11



We will now consider the different types of masks that are useful in spatial image enhancement.

2.2.3 Smoothing Masks

Sometimes we want to remove fine details from an image or make a noisy image easier to see. We can use smoothing masks to accomplish this goal. To smooth an image we

need to use a mask with only positive coefficients whose sum equals one. The fillter

$$g = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

is an example of a smoothing mask that takes a 3×3 neighborhood of pixels and averages them. If we want to increase the amount of blurring in an image we can increase the size of the neighborhood. For instance, we can average a 5×5 neighborhood of pixels by letting

$$g = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

If we want to average a 7×7 neighborhood then

$$g = \frac{1}{49}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

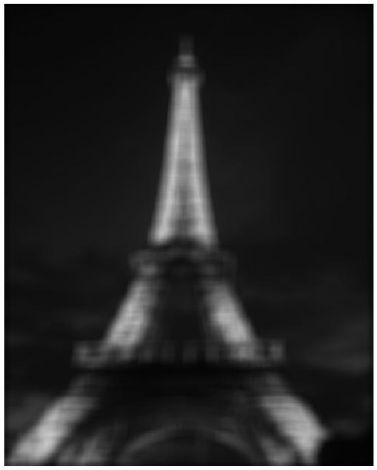
In Figure 2.12 we can compare the amount of blurring in an image to the size of the averaging filter. In (a) we have an image of the Eiffel Tower [7]. In (b) a 3×3 averaging filter is used to blur the image. Visually there is not much of a difference between (a) and (b) but if we increase the size of the averaging filter to 7×7 (c) or 15×15 (d) the blurring is more pronounced.

Figure 2.12

(a) Original Image

(b) 3×3

(c) 7×7



(d) 15×15

If we want to clear an image of noise (unwanted intensities in an image brought on by camera defects or the aging of an image) but reduce the amount of blurring we can give pixels that are closer to the center more weight than pixels that are farther away. For instance, we can let

$$g = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

This is called a weighted average filter or a Gaussian filter.

Another way to increase the amount of blurring (rather than change the dimensions of the filter) is to give heavier weights to pixels that are further away from the center than pixels that are closer. An example of this is the filter

$$g = \frac{1}{16}\begin{bmatrix} 3 & 1 & 3 \\ 1 & 0 & 1 \\ 3 & 1 & 3 \end{bmatrix}.$$

2.2.4 Sharpening Masks

Certain types of filters can find edges and sharpen images. In order to find an edge we need to find transitions in image intensity. These transitions can be detected by what is called a derivative filter. A derivative filter finds the rate of change between neighboring pixels. One way to estimate this change is to let

$$\frac{\partial f}{\partial x} \cong f(x+1, y) - f(x, y) \text{ and } \frac{\partial f}{\partial y} \cong f(x, y+1) - f(x, y).$$

In terms of a filter this would be

$$\frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } \frac{df}{dy} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

If we want to find edges in both the x and y direction we can use the magnitude of the gradient

$$|\bigtriangledown f| \cong \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

Another way to estimate the derivative is to subtract $f(x, y-1)$ from $f(x, y+1)$ for all x values in a 3×3 neighborhood. This derivative can be written in the form

of a mask (called a Prewitt filter) where

$$\frac{df}{dx} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}. \tag{2.3}$$

Similarly, we can let

$$\frac{df}{dy} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

First order derivatives tend to produce thicker edges than second order derivatives. Second order derivatives also tend to have a better response to fine detail which make them better for sharpening images. We can estimate the second derivative by the formula:

$$\frac{d^2 f}{dx^2} \cong f(x+1, y) + f(x-1, y) - 2f(x, y).$$

To find the edges in both the x and y direction we can use the Laplacian which is given by

$$\triangle f = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2}.$$

We could write this as a filter where

$$\triangle f \cong \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

In Figure 2.13 we can compare the effects of finding edges with the Prewitt gradient filter (b) against the Laplacian (c) on an image of a panda [6].

Figure 2.13

(a) Original

(b) Prewitt

(c) Laplacian

We can use edge detection methods to sharpen an image. Once we find the edges of an image we can subtract them from the original image to create a sharper result. We can also extend the Laplacian to the diagonal neighbors by using the filter

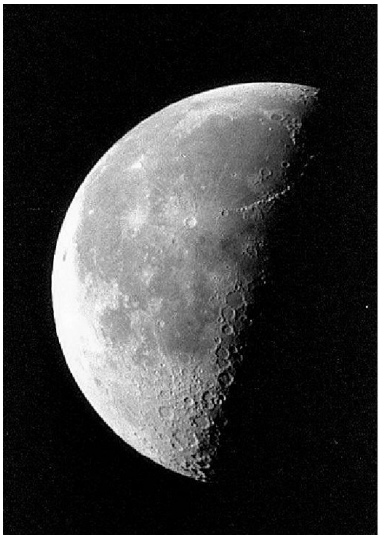
$$g = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

One other way to boost the amount of sharpening is to multiply the Laplacian by a positive constant c before subtracting it from the original image. The right c to choose depends on the viewer but a constant too small will have little to no effect while a constant that is too big may oversharpen the image and create an unwanted result.

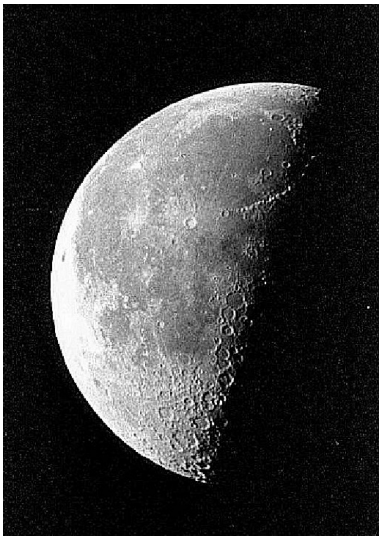
In Figure 2.14 (a) we have an image of the moon. In (b) we filtered the image with the Laplacian and subtracted the result from the original image. In (c) we subtracted the image filtered with the extended Laplacian from the original image. Notice that using the extended Laplacian provides us with a sharper image than (b) since it takes into account the diagonal edges. Finally, in (d) we have multiplied the Laplacian by 5 and then subtracted it from the original image. This has increased the sharpening effect but it has also enhanced the noise in the image (the white specks surrounding the moon). Since derivative filters find transitions in image intensity values it makes sense that the noise would be amplified. In order to reduce the noise we can choose a smaller c or blur the original image with a smoothing filter before applying the Laplacian (or extended Laplacian) filter.

Figure 2.14

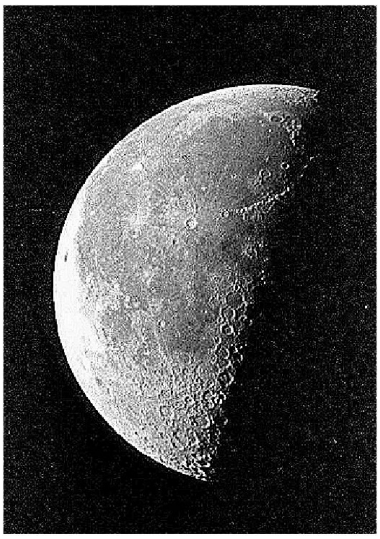
(a) Original Image



(b) Laplacian



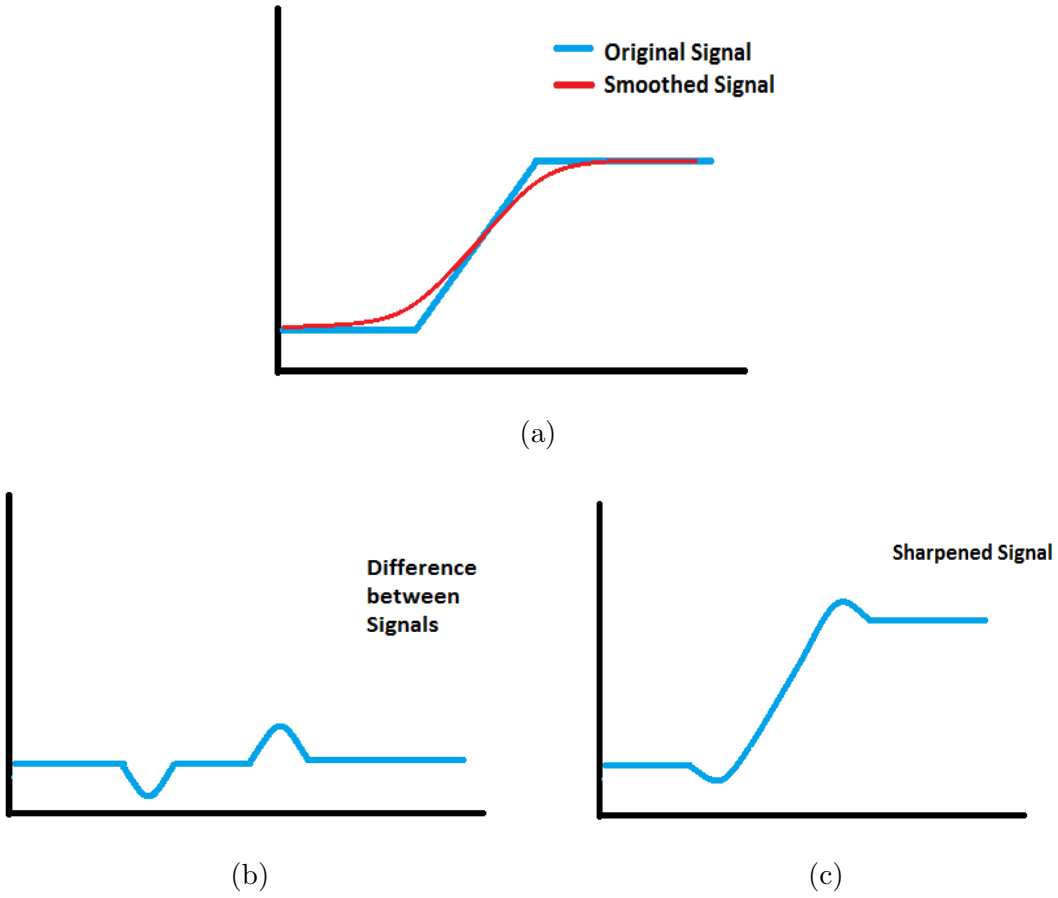
(c) Extended Laplacian



(d) Laplacian Boost

We can also sharpen an image by using a process called unsharp filtering. When we blur an image with a smoothing filter we decrease the fine details of the original image. Thus subtracting the blurred image from the original will leave us with edges and fine details. We can take those edges and add them back to the original to create a sharper image as shown in Figure 2.15 [8].

Figure 2.15: Unsharp Masking

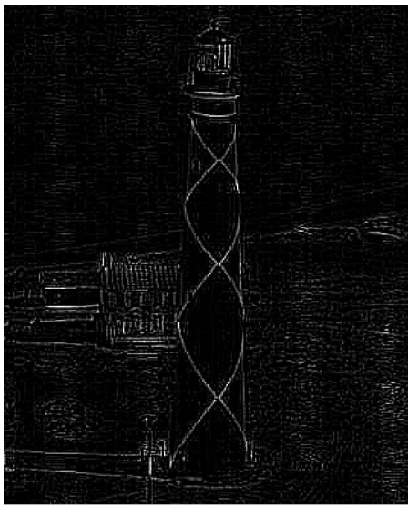


In Figure 2.16 we have applied unsharp filtering to an image of the Cape Lookout Lighthouse [5]. In (a) we have the original image and in (b) we have a blurred image obtained by using a 3×3 averaging filter. In (c) we have the difference between the original image and the blurred image (multiplied by 1.5 for the edges to be easier to see). Finally in (d) we have added the edges back to the original image and we have a sharper effect.

Figure 2.16

(a)

(b)



(c)

(d)

CHAPTER 3: Image Enhancement in the Frequency Domain

All the filtering processes we have seen so far in the spatial domain can also be applied to an image's frequency domain. Let us start with a review of the Fourier Transform and some of its properties.

3.1 The Fourier Transform

Let f be an integrable function on $\mathbb{R}$. The Fourier transform of f is defined by the equation

$$F(u) = \mathcal{F}f(u) = \int_{-\infty}^{\infty} f(x)e^{-2\pi iux}dx.$$

The Fourier transform extends from $L^1 \cap L^2$ to L^2. The inverse Fourier transform:

$$f(x) = \int_{-\infty}^{\infty} F(u)e^{2\pi iux}du$$

extends to L^2 where $\mathcal{F}$ is an isometry. The following theorem lists some basic properties of the Fourier transform [2].

Theorem 3.1. *Suppose $f \in L^1(\mathbb{R})$.*
(1) For any $a \in \mathbb{R}$, let $(T_a f)(x) = f(x-a)$ and $(M_a f)(x) = e^{2\pi iax} f(x)$. Then

$$\mathcal{F}(T_a f) = M_{-a}(\mathcal{F}f) \text{ and } \mathcal{F}(M_a f) = T_a(\mathcal{F}f).$$

(2) If $g \in L^1$ then

$$\mathcal{F}(f * g)(u) = F(u)G(u).$$

Proof. For the first part of (1) we have

$$\mathcal{F}(T_a f)(u) = \int_{-\infty}^{\infty} e^{-2\pi i u x} f(x-a)dx.$$

Let $z = x - a$. Then $dz = dx$ and

$$\mathcal{F}(T_a f)(u) = \int_{-\infty}^{\infty} e^{-2\pi i u(z+a)} f(z)dz = e^{-2\pi i a u} \int_{-\infty}^{\infty} e^{-2\pi i u z} f(z)dz = M_{-a}(\mathcal{F}f)(u).$$

For the second part of (1), we have

$$\mathcal{F}(M_a f)(u) = \int_{-\infty}^{\infty} e^{-2\pi i u x} e^{2\pi i a x} f(x)dx = \int_{-\infty}^{\infty} e^{-2\pi i x(u-a)} f(x)dx = T_a(\mathcal{F}f)(u).$$

For (2),

$$\begin{aligned}
\mathcal{F}(f * g)(u) &= \int_{-\infty}^{\infty} e^{-2\pi i u x} \left(\int_{-\infty}^{\infty} f(a) g(x-a) da \right) dx \\
&= \int_{-\infty}^{\infty} e^{-2\pi i u x} \left(\int_{-\infty}^{\infty} e^{-2\pi i u a + 2\pi i u a} f(a) g(x-a) da \right) dx \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-2\pi i u a} f(a) e^{-2\pi i u(x-a)} g(x-a) da dx.
\end{aligned}$$

Let $z = x - a$. Then

$$\begin{aligned}
\mathcal{F}(f * g)(u) &= \int_{-\infty}^{\infty} e^{-2\pi i u a} f(a) da \int_{-\infty}^{\infty} e^{-2\pi i u z} g(z) dz \\
&= F(u)G(u).
\end{aligned}$$

□

We can extend the Fourier transform for a vector $\vec{x} \in \mathbb{R}^n$ where:

$$F(\vec{u}) = \int_{\mathbb{R}^n} f(\vec{x}) e^{-2\pi i <\vec{u},\vec{x}>} d\vec{x}$$

and the inverse Fourier transform is:

$$f(\vec{x}) = \int_{\mathbb{R}^n} F(\vec{u}) e^{2\pi i <\vec{u},\vec{x}>} d\vec{u}$$

3.2 The Discrete Fourier Transform

Let f be a real-valued function that is N-periodic. The discrete Fourier transform is defined by

$$\mathcal{F}f(u) = F(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) e^{-2\pi i \frac{ux}{N}}$$

for $u = 0, 1, 2, ..., N-1$. The inverse discrete Fourier transform is given by

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} F(u) e^{-2\pi i \frac{ux}{N}}$$

for $x = 0, 1, 2, ..., N-1$.

The discrete Fourier Transform shares a number of properties with the continuous Fourier transform. Like the continuous Fourier transform, the discrete Fourier transform converts discrete convolution into multiplication.

Theorem 3.2. *Let f and g be two real-valued functions that are N-periodic on $\mathbb{Z}$. Then $\mathcal{F}(f * g) = \sqrt{N} F(u) G(u)$.*

Proof. By definition of discrete convolution, $(f * g)(x) = \sum_{l=0}^{N-1} f(l)g(x-l)$. Thus

$$\begin{aligned}
\mathcal{F}(f*g)(u) &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{-2\pi i \frac{ux}{N}} (f*g)(x) \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{-2\pi i \frac{ux}{N}} (\sum_{l=0}^{N-1} f(x-l)g(l)) \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{l=0}^{N-1} e^{-2\pi i \frac{ux}{N}} f(x-l)g(l) \\
&= \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} g(l) \sum_{x=0}^{N-1} e^{-2\pi i \frac{ux}{N}} f(x-l).
\end{aligned}$$

Now let $z = x - l$. Then

$$\begin{aligned}
\mathcal{F}(f*g)(u) =& \frac{1}{N} \sum_{l=0}^{N-1} g(l) \sum_{z=-l}^{N-1-l} e^{-2\pi i \frac{u(z+l)}{N}} f(z) \\
=& \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} e^{-2\pi i \frac{ul}{N}} g(l) \sum_{z=-l}^{N-1-l} e^{-2\pi i \frac{uz}{N}} f(z) \\
=& \frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} e^{-2\pi i \frac{ul}{N}} g(l) \sum_{z=0}^{N-1} e^{-2\pi i \frac{uz}{N}} f(z) \\
=& \sqrt{N} (\frac{1}{\sqrt{N}} \sum_{l=0}^{N-1} e^{-2\pi i \frac{ul}{N}} g(l)) (\frac{1}{\sqrt{N}} \sum_{z=-l}^{N-1-l} e^{-2\pi i \frac{uz}{N}} f(z)) \\
=& \sqrt{N} G(u) F(u).
\end{aligned}$$

□

Similarly to the continuous Fourier transform, the discrete Fourier transform can be extended to the two-variable case. The discrete Fourier transform pair is

$$F(u,v) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i(\frac{ux}{M} + \frac{vy}{N})}$$

for $u = 0, 1, 2, ..., M-1, v = 0, 1, 2, ..., N-1$ and

$$f(x,y) = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{y=0}^{N-1} F(u,v) e^{2\pi i(\frac{ux}{M} + \frac{vy}{N})}$$

for $x = 0, 1, 2, ..., M-1, y = 0, 1, 2..., N-1$.

3.3 Fast Fourier Transform

Consider again the one dimensional discrete Fourier transform

$$F(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) e^{-2\pi i \frac{ux}{N}}$$

for $u = 0, 1, 2, ..., N-1$. In practice, the discrete Fourier transform takes considerable time to compute. Let us define an "elementary operation" to be a multiplication of two complex numbers followed by the addition of two complex numbers.To compute each $F(u)$ there are $N-1$ elementary operations. Since u ranges from 0 to $N-1$ the computational time is proportional to N^2. If N is large then N^2 is enormous and the transform may become computationally unmanageable. One way to reduce the computation time is to rearrange and reduce the discrete Fourier transform in a certain way called the fast Fourier transform.

To reach the fast Fourier transform, let us begin by assuming $N = 2^n$ where n is a positive integer. Doing so allows us to express N as $N = 2M$ where $M = 2^{n-1}$. We use the notation F_N when performing the discrete Fourier transform on a space of N

points. Substituting $N = 2M$ into the discrete Fourier transform we get

$$\begin{aligned} F_{2M}(u) &= \frac{1}{\sqrt{2M}} \sum_{x=0}^{2M-1} f(x) e^{-2\pi i \frac{ux}{2M}} \\ &= \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} f(2x) e^{-2\pi i \frac{u(2x)}{2M}} + \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} f(2x+1) e^{-2\pi i \frac{u(2x+1)}{2M}} \right] \\ &= \frac{1}{\sqrt{2}} \left[\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} f(2x) e^{-2\pi i \frac{ux}{M}} + \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} f(2x+1) e^{-2\pi i \frac{ux}{M}} e^{-\pi i \frac{u}{M}} \right] \end{aligned}$$

Define

$$F_M^{even}(u) = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} f(2x) e^{-2\pi i \frac{ux}{M}}$$

and

$$F_M^{odd}(u) = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} f(2x+1) e^{-2\pi i \frac{ux}{M}}$$

for $u = 0, 1, ..., M-1$. Then

$$F(u) = \frac{1}{\sqrt{2}} \left[F_M^{even}(u) + F_M^{odd}(u) e^{-\pi i \frac{u}{M}} \right] \tag{3.1}$$

where $u = 0, 1, .., N/2 - 1$. Since $e^{-2\pi i \frac{(u+M)}{M}} = e^{-2\pi i \frac{u}{M}}$ and $e^{-2\pi i \frac{(u+M)}{2M}} = -e^{-\pi i \frac{u}{M}}$ we have

$$F(u+M) = \frac{1}{\sqrt{2}} \left[F_M^{even}(u) - F_M^{odd}(u) \right] \tag{3.2}$$

where $u + M = N/2, ..., N$. Thus the N-point fast Fourier transform can be obtained from using both (3.1) and (3.2). The number of operations required to use the fast Fourier transform is proportional to $N \log_2 N$. Comparing this to the discrete Fourier transform, we can see that the fast Fourier transform has a computational advantage of $\frac{N}{\log_2 N}$.

To find the inverse fast Fourier transform of

$$f(x) = \frac{1}{\sqrt{N}} \sum_{u=0}^{N-1} F(u) e^{2\pi i \frac{ux}{N}}$$

notice that

$$\overline{f}(u) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \overline{F}(x) e^{-2\pi i \frac{ux}{N}} \tag{3.3}$$

which is in the form of the fast Fourier transform. Thus we can take the complex conjugate of F, apply the fast Fourier transform, and take the complex conjugate of f to find the inverse fast Fourier transform.

As in the one dimensional case, we can decompose the discrete Fourier transform for two dimensions by using

$$F(u,v) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i (\frac{ux}{M} + \frac{vy}{N})}$$

for $u = 0, 1, 2, ..., M-1, v = 0, 1, 2, ..., N-1$. Notice that we can separate the transform into

$$F(u,v) = \frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} e^{-2\pi i \frac{ux}{M}} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i \frac{vy}{N}}.$$

Thus we can apply the fast Fourier transform first to the row vectors and then to the column vectors.

3.4 Properties of the Fourier Transform and the Power Spectrum

In our study of image processing we would like to visually understand the effects the Fourier transform has on an image. In order to apply the Fourier transform, an image

can be extended to a periodic function on $\mathbb{Z} \times \mathbb{Z}$. Since we cannot display complex-valued functions as a single image we can instead display the **power spectrum** $|F|$. Analysis of the power spectrum of an image gives us some interesting properties.

3.4.1 Viewing the Power Spectrum

We can find the complex conjugate of the Fourier transform F by

$$\overline{F}(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \overline{f}(x,y)e^{2\pi i(ux+vy)}dx.$$

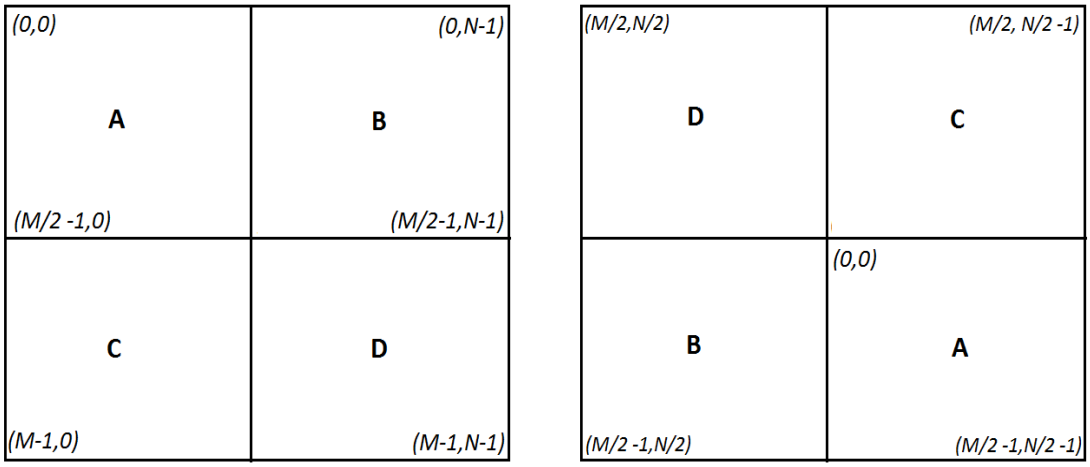
Since f is a real-valued function, we have

$$\begin{aligned}\overline{F}(u,v) &= \int_{-\infty}^{\infty} f(x,y)e^{2\pi i(ux+vy)}dx \\ &= \int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(-ux-vy)}dx \\ &= F(-u,-v).\end{aligned}$$

Thus $|F(u,v)| = |\overline{F}(u,v)| = |F(-u,-v)|$, which implies that the power spectrum is symmetric about the origin. We also have $|F(u,v)| \to 0$ as $(u,v) \to \infty$.

In the discrete case, $|F(u,v)| = |F(u+M, v+N)|$ and in general, we expect that as u, v increase then $|F(u,v)|$ becomes small. Thus the values of the discrete Fourier transform are the largest around the origin and decrease as we approach the middle of its period in the x and y direction. Since we defined an image using matrix notation, we need to rearrange the matrix of the power spectrum as shown in Figure 3.1 to better analyze its properties. That is, we place $(0,0)$ in the center of the image and use periodicity to arrange the other entries.

Figure 3.1



(a) Original Image (b) Power Spectrum

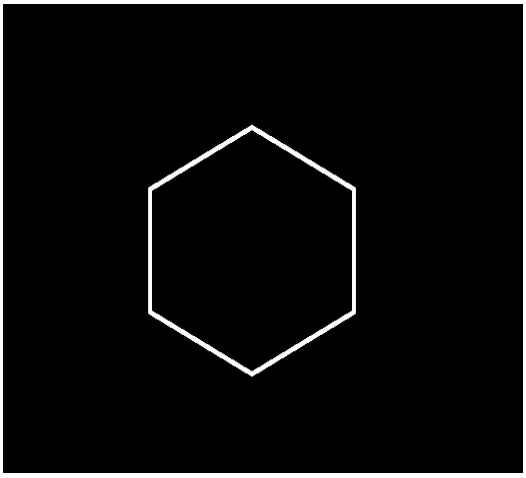
The range of values for the power spectrum is so large that only the brightest parts are visible in the image. To fix this we can use a transformation function G where

$$G(u, v) = c \log(1 + |F(u, v)|)$$

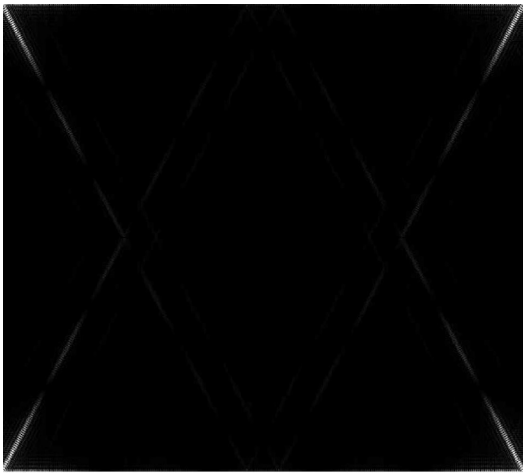
and c is a scaling constant.

In Figure 3.2 (a) we have an image of a hexagon. In (b) we have the image of the power spectrum of the hexagon and in (c) we have repositioned the power spectrum so that the origin is in the center. Finally in (d) we have applied the function $G(u, v) = \log(1 + |F(u, v)|)$ to the transform. This exercise illustrates the phenomenon that the Fourier transform of an image tends to have its largest values around the origin.

Figure 3.2

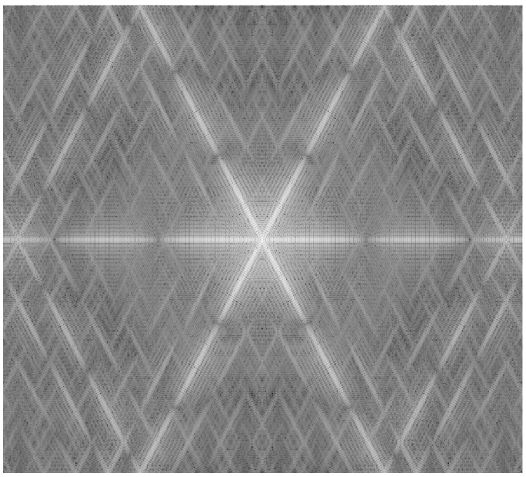


(a) Original Image



(b) Power Spectrum

(c) Recentered Power Spectrum



(d) Logged Power Spectrum

3.4.2 The Translation Property

We saw in the one-dimensional continuous case that $\mathcal{F}(T_a f) = M_{-a}(\mathcal{F}f)$. If we take the absolute value of $\mathcal{F}(T_a f)$ we get

$$|\mathcal{F}(T_a f)(u)| = \left|e^{-2\pi i a u}\right| \left|\int_{-\infty}^{\infty} e^{-2\pi i u z} f(z) dz\right| = |F(u)|.$$

Therefore if we translate an image and take the absolute value of the Fourier transform we end up with the same image of the power spectrum as if we did not translate the image at all.

This property also holds for the discrete two-dimensional case. In Figure 3.3 (a) and (b) we have an image of a rectangle and its logged power spectrum. In (c) we have translated the rectangle and in (d) we show its power spectrum. Notice that (b) and (d) are the exact same image.

3.4.3 The Rotation Property

Let x and u be vectors in $\mathbb{R}^2$. Let A be a 2×2 rotational matrix. Then $A^T = A^{-1}$ and $\det A = 1$. Thus

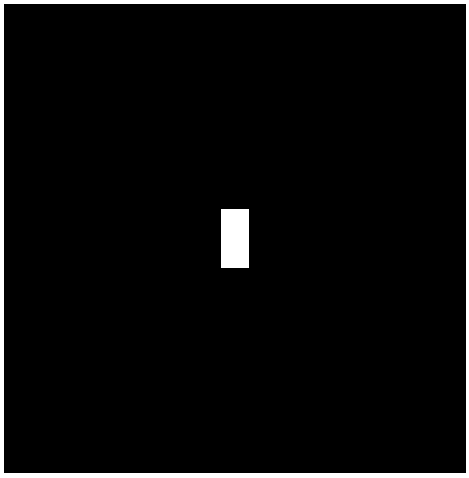
$$\begin{aligned}\mathcal{F}(fA)(u) &= \int_{\mathbb{R}^2} e^{-2\pi i u x} f(Ax)dx \\ &= \int_{\mathbb{R}^2} e^{-2\pi i u A^T Ax} f(Ax)dx\end{aligned}$$

Here we use a change of variables, letting $z = Ax$ and $dz = |\det A|dx$. Thus

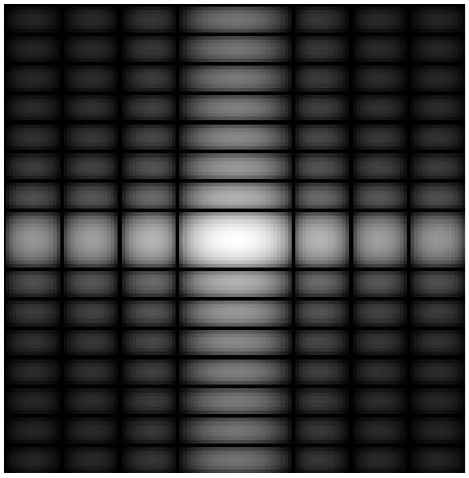
$$\begin{aligned}\mathcal{F}(fA)(u) &= \int_{\mathbb{R}^2} e^{-2\pi i u A^T z} f(z)dz \\ &= \int_{\mathbb{R}^2} e^{-2\pi i A u z} f(z)dz \\ &= \mathcal{F}f(Au)\end{aligned}$$

To illustrate the rotation property, in Figure 3.3 (e) we applied a rotation matrix to our image of a square and then took the Fourier transform (f). If we compare (f) with (b), we notice that the Fourier transform is also rotated.

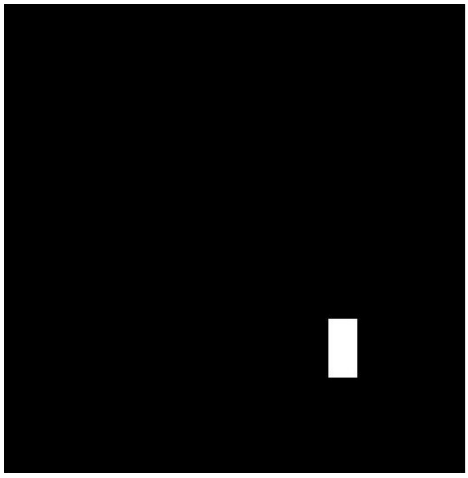
Figure 3.3



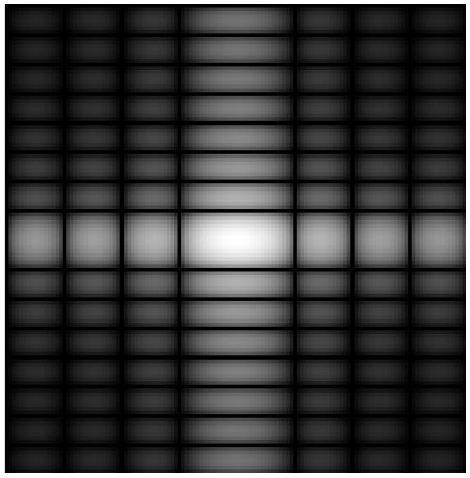
(a) Original Image



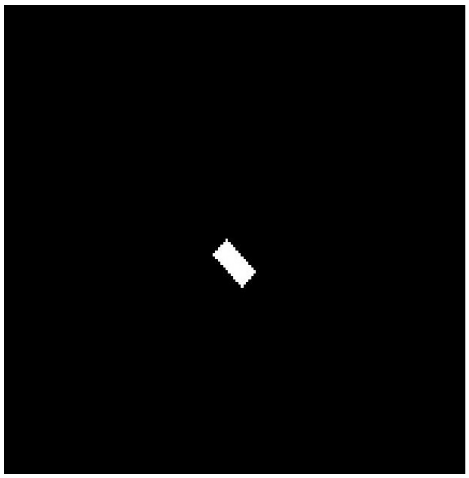
(b) Power Spectrum



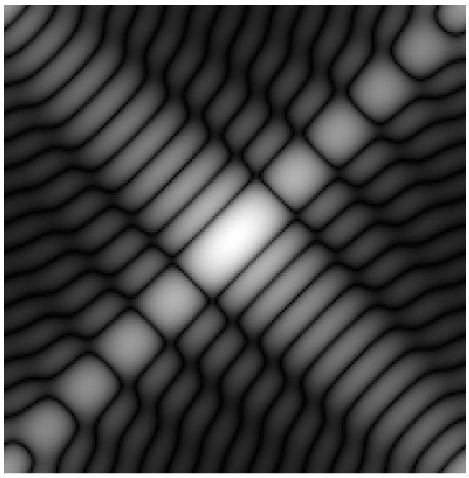
(c) Translated Image



(d) Translated Image's Power Spectrum



(e) Rotated Image

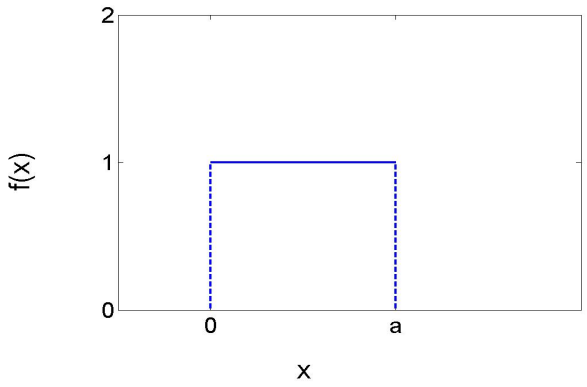


(f) Rotated Image's Power Spectrum

3.4.4 The Wavelength Property

Let us find the Fourier transform of the function $f(x) = 1$ where $0 < x < a$ (as pictured in Figure 3.4).

Figure 3.4



Then

$$\begin{aligned}\mathcal{F}[f(x)] &= \int_{-\infty}^{\infty} e^{-2\pi iux} f(x)dx = \int_{0}^{a} e^{-2\pi iux} dx \\ &= -\frac{1}{2\pi iu} e^{-2\pi iux}|_0^a = \frac{-1}{2\pi iu}(e^{-2\pi iua} - 1) \\ &= \frac{-1}{2\pi iue^{\pi iau}}(e^{-\pi iua} - e^{\pi iau}) = \frac{1}{\pi ue^{\pi iau}} \frac{(e^{\pi iua} - e^{-\pi iau})}{2i}.\end{aligned}$$

Since $\sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$, we can see that

$$\begin{aligned}\mathcal{F}[f(x)] =& \frac{1}{\pi ue^{\pi iau}} \frac{(e^{\pi iua} - e^{-\pi iau})}{2i} \\ =& \frac{1}{\pi ue^{\pi iau}} \sin(\pi au) \\ =& \frac{a}{e^{\pi iau}} \frac{\sin(\pi au)}{\pi au} = \frac{a}{e^{\pi iau}} \operatorname{sinc}(au).\end{aligned}$$

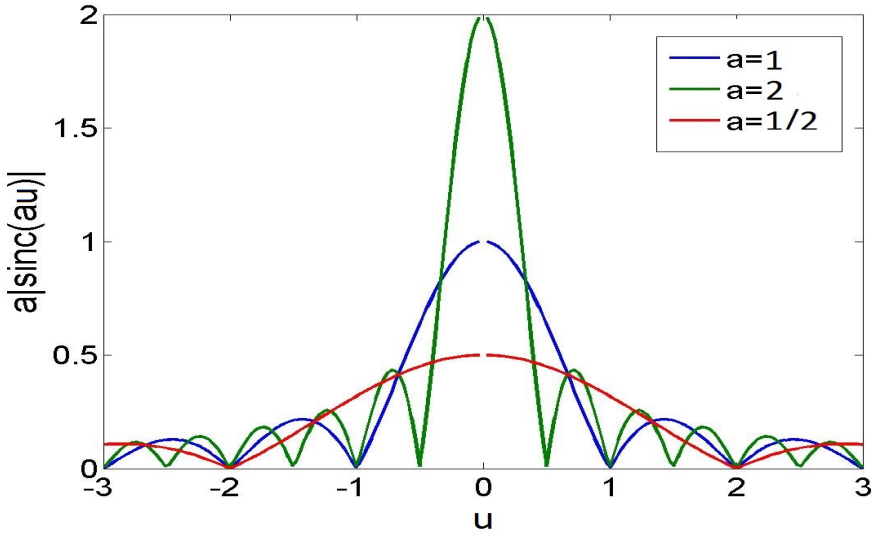
Now to see the power spectrum, we need to take the magnitude of the Fourier trans-

form. That is,

$$|\mathcal{F}f(u)]| = \left|\frac{a}{e^{\pi i a u}}\operatorname{sinc}(u)\right| = a|\operatorname{sinc}(au)|.$$

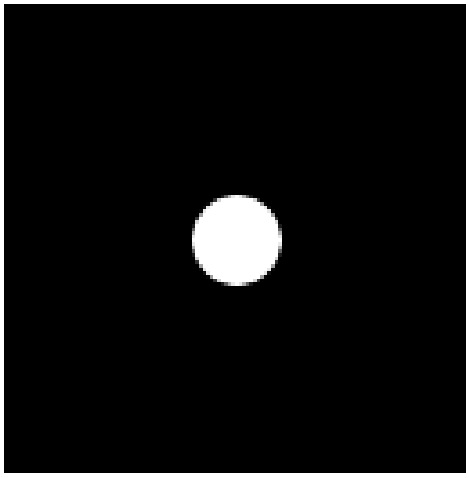
In Figure 3.5, we show three cases where $a = 1, a = 2$, and $a = 1/2$. If we let the wave length be denoted by λ, we can see that $\lambda = \frac{1}{a}$.

Figure 3.5

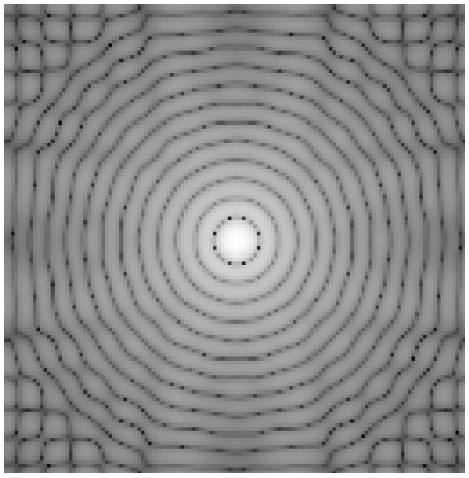


In Figure 3.6 (a) we demonstrate the wavelength property with a 128×128 pixel image of a circle. Notice that we have two gray values, where 0 = black and 1 = white. Therefore we have a function f where $f(x, y) = 1$ when $(x - 64)^2 + (y - 64)^2 < r^2$ (where r is the radius of the circle) and $f(x, y) = 0$ otherwise. The length of our interval $a = 2r$ is the same in each direction. In (b) we have the image of the centered power spectrum. In (c) we have a circle but with a smaller radius and in (d) we have the centered power spectrum. Notice the wavelength between (b) and (d) has lengthened since r has decreased.

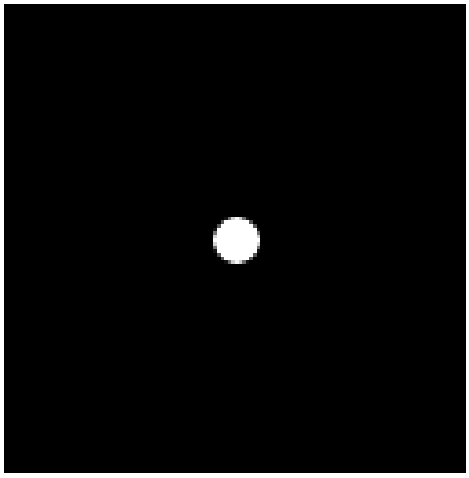
Figure 3.6



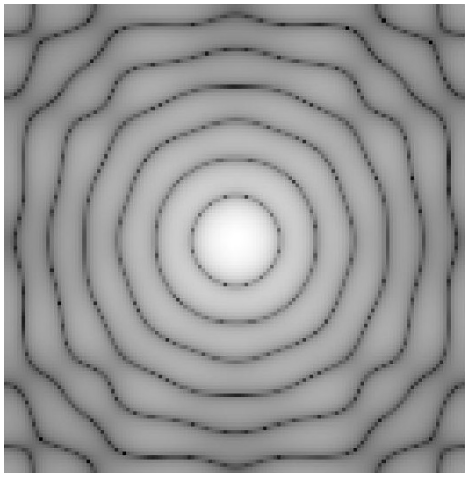
(a) Original Image



(b) Power Spectrum



(c) Circle with Reduced Radius



(d) Power Spectrum

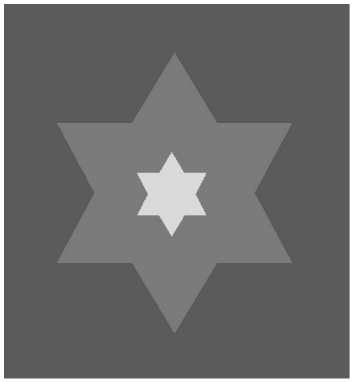
3.5 Filtering in the Frequency Domain

By using the fact that convolution is multiplication on the frequency side we can blur an image using one of the averaging filters we saw earlier. In Figure 3.7 (a) we have an image made of 3 gray-values and in (b) we have the logged power spectrum. In (c) we take the 15×15 averaging filter

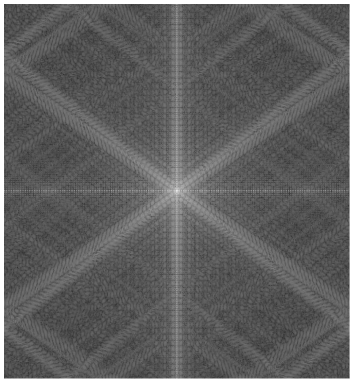
$$g(x,y) = \frac{1}{225}\begin{bmatrix} 1 & 1 & ... & ... & 1 \\ 1 & 1 & ... & ... & 1 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & 1 & ... & ... & 1 \end{bmatrix}$$

and find the centered power spectrum of the Fourier transform G. In (d) we multiply the Fourier transform of f with g and then take the absolute value. Lastly, we take the inverse Fourier transform and obtain the image seen in (e).

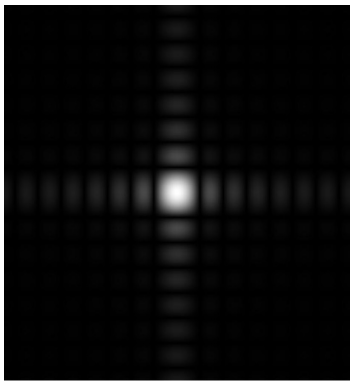
Figure 3.7



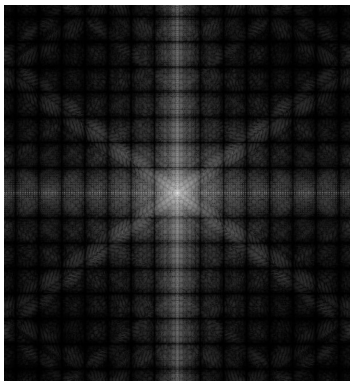
(a) Original Image



(b) $|F|$



(c) $|G|$



(d) $|FG|$

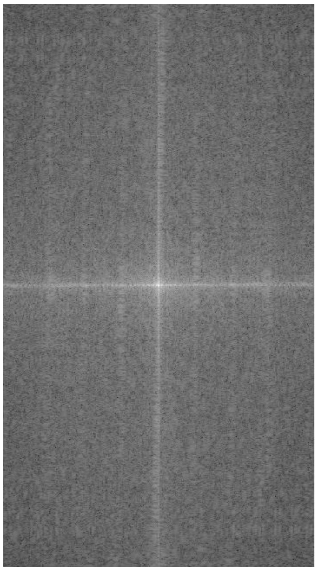
(e) Blurred Image

In Figure 3.8 we apply the Prewitt filter we saw in equation (2.3) to an image of Big Ben [1]. In (a) we have the original image, in (b) we have the power spectrum of the image, in (c) we have the power spectrum of the Laplacian. In (d) we multiply the

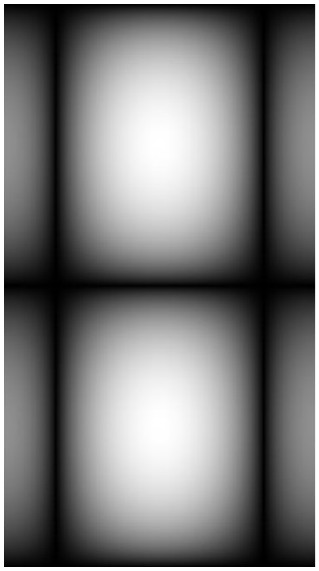
Fourier transform of f with g and show the power spectrum of that result. Finally in (e) we take the inverse Fourier transform of F times G and obtain the edges of the original image.

Figure 3.8

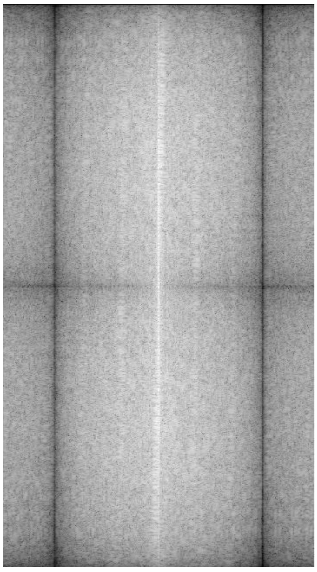
(a) Big Ben



(b) $|F|$



(c) $|G|$



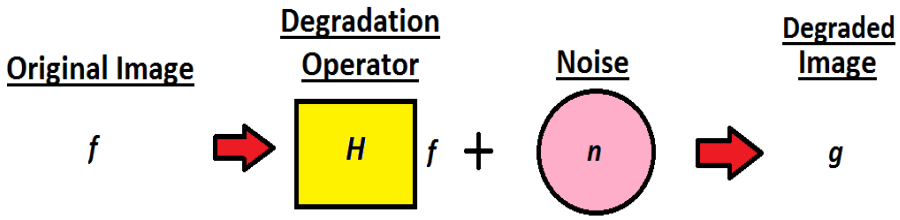
(d) $|FG|$

(e) Edges of Big Ben

CHAPTER 4: Image Restoration

In this chapter we take degraded images and try to restore them as closely as possible to their original quality. The idea behind image degradation is that we have some operator H that operates on an image f along with a noise term n that is added to produce a degraded image g as shown in Figure 4.1 [8]. It is assumed that we do not know the exact original image but we try to form an estimate $\hat{f}$ that is as close as possible to the original image. For simplification purposes, we assume that H is a linear, translation invariant operator. Nonlinear and space variant operators often have no known solution or are computationally unmanageable [8]. We first consider images with degradation due to noise only (H is the identity operator) and then consider images with both noise and blurring.

Figure 4.1



4.1 Types of Noise

Let us consider image restoration where the only degradation present is noise. That is, the operator acting on f is the identity operator

$$g = If + n.$$

Two common types of noise are Gaussian noise and impulse noise. Gaussian noise is easy to work with mathematically and a good model for camera sensor noise that comes from poor lighting and high temperatures. The probability distribution function of Gaussian noise is

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}},$$

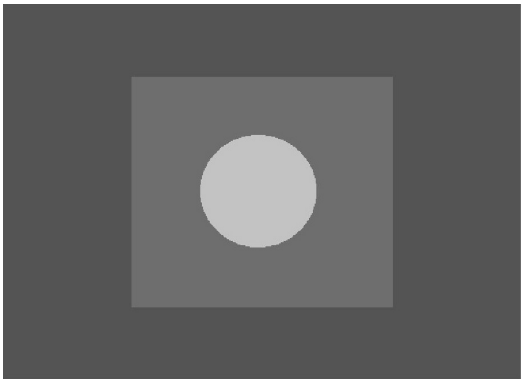
where $-\infty < x < \infty$. Impulse noise occurs when there is malfunctioning of the camera's sensor cells, memory failure, or errors in image digitization [8]. This type of noise produces one of two colors, black or white. Often referred to as "salt and pepper noise," the probability distribution function is given by

$$P(x) = \begin{cases} P_a & \text{for } x = a \\ P_b & \text{for } x = b \\ 0 & \text{otherwise,} \end{cases}$$

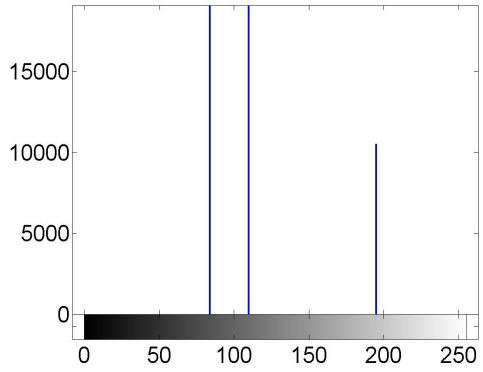
where $b > a$ and $P_a + P_b = 1$.

By looking at the histogram of a noisy image we can often determine the type of noise that has been added to an image. In Figure 4.2 (a) and (b) we see an image and its histogram. Notice that the histogram shows that the image has three gray values. In (c) and (d) we have added Gaussian noise to the image and show its histogram. We can see that now the histogram takes on a Gaussian curve. Finally, in (e) and (f) we have the original image with added impulse noise and its histogram. In (f) we have the three original gray values and two additional gray values (or impulses) where 0 = black and 255 = white.

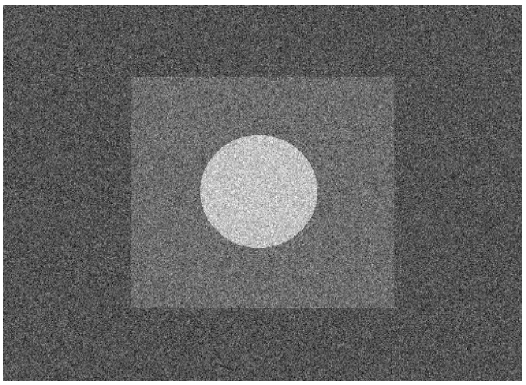
Figure 4.2



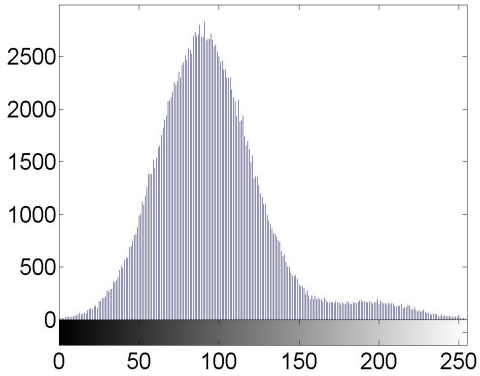
(a)



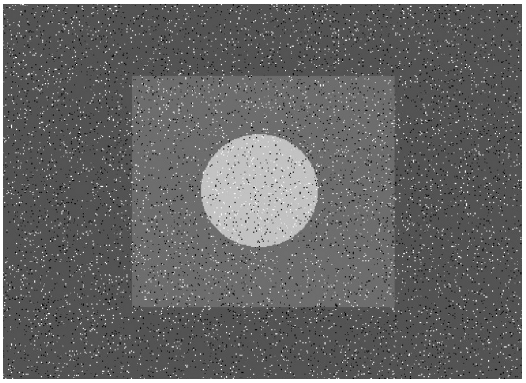
(b)



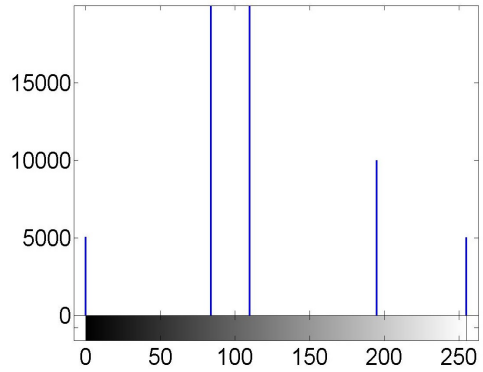
(c)



(d)



(e)



(f)

4.2 Noise Removal

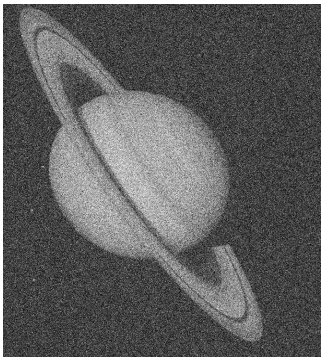
If we have a set of N still images that contain noise we can reduce the noise effects by averaging the images. The formula for image averaging is

$$g_N(x,y) = \frac{1}{N}\sum_{i=1}^{N} f_i(x,y).$$

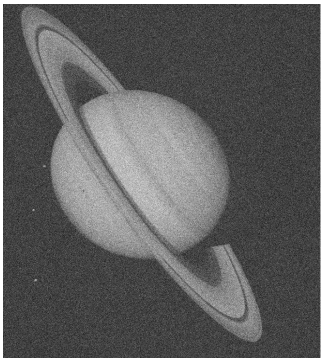
As N increases, the variance of the noise in g_N decreases by $\frac{1}{N}$. In Figure 4.3 (a) we have one image of Saturn that has been subjected to random Gaussian noise where $\mu = 0$ and $\sigma^2 = 0.5$. After averaging five of these noisy images we have the result given in (b). If we average 20 or 100 noisy images, as in (c) and (d), we obtain a clearer image.

Figure 4.3

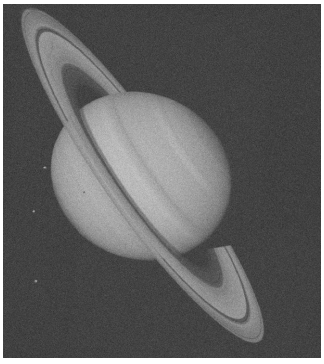
(a) Noisy Image



(b) Average of 5 images



(c) Average of 20 images



(d) Average of 100 images

Knowing the noise distribution of an image can help us choose the best filter to reduce the noise. An image corrupted with Gaussian or uniform noise responds well to averaging filters while an image with impulse noise responds best to the median filter. In Figure 4.4 (a) we have an image of a cameraman and in (b) 20% of the image hase been corrupted with impulse noise. In (c) a 3×3 averaging filter has been applied to try to clean the image while in (d) a 3×3 median filter is used. A median filter takes a 3×3 neighborhood of pixels and selects the median. It is obvious that the median filter does a better job of filtering when it comes to impulse noise.

Figure 4.4

(a) Original Image

(b) Corrupted Image

(c) 3×3 Averaging

(d) 3×3 Median Filtering

Another impulse reducing filter is to take the minimum value of a 3×3 window if we want to reduce only salt noise (or take the maximum value if we want to reduce pepper noise). We can also find both the maximum and mininum in a 3×3, add them together and divide by two. This type of filter is called the midpoint filter.

When an image has several gray values, sometimes we cannot clearly see the noise distribution. In this case we look for areas of pixels in an image that have near uniform gray values. In each of these sections we find the histogram and compute the mean and standard deviation. We then average the means and the deviations to find a noise model that matches closely to the histogram. Once we find our noise model we can then choose the best filter to reduce the noise. Another option is to use trial and error when selecting filters.

4.3 Circulant and Block-Circulant Matrices

We have defined the degradation model of an image by the formula

$$g = Hf + n.$$

The linear operator H is translation-invariant if and only if H is a convolution operator. Suppose that we have two real-valued functions f and h that are M-periodic (notice that if f has a shorter period than h then we can extend f with zeros until its period is equal to the period of h). We define the convolution of the functions f and h by

$$g(x) = (f * h)(x) = \sum_{m=0}^{M-1} = f(x)h(x-m)$$

for $x \in \{0, 1, 2, ..., M-1\}$.

We can use matrix notation to represent the convolution

$$g = Hf + n$$

where

$$f = \begin{bmatrix} f(0) \\ f(1) \\ . \\ . \\ . \\ f(M-1) \end{bmatrix}, \quad g = \begin{bmatrix} g(0) \\ g(1) \\ . \\ . \\ . \\ g(M-1) \end{bmatrix} \text{ and } n = \begin{bmatrix} n(0) \\ n(1) \\ . \\ . \\ . \\ n(M-1) \end{bmatrix}$$

and

$$H = \begin{bmatrix} h(x_1-0) & h(x_1-1) & h(x_1-2) & ... & h(x_1-M+1) \\ h(x_2-0) & h(x_2-1) & h(x_2-2) & ... & h(x_2-M+1) \\ h(x_3-0) & h(x_3-1) & h(x_3-2) & ... & h(x_3-M+1) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ h(x_M-0) & h(x_M-1) & h(x_M-2) & ... & h(x_M-M+1) \end{bmatrix}$$

where $x_i \in \{0, 1, 2, ..M-1\}$ and $i \in \{1, 2, 3, ...M\}$. That is,

$$H = \begin{bmatrix} h(0) & h(-1) & h(-2) & ... & h(-M+1) \\ h(1) & h(0) & h(-1) & ... & h(-M+2) \\ h(2) & h(1) & h(0) & ... & h(-M+3) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ h(M-1) & h(M-2) & h(M-3) & ... & h(0) \end{bmatrix}.$$

Since h is periodic, we have $h(x) = h(x+M)$ and can write

$$H = \begin{bmatrix} h(0) & h(M-1) & h(M-2) & ... & h(1) \\ h(1) & h(0) & h(M-1) & ... & h(2) \\ h(2) & h(1) & h(0) & ... & h(3) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ h(M-1) & h(M-2) & h(M-3) & ... & h(0) \end{bmatrix}.$$

We call this matrix a circulant matrix since the rows are related by a circular shift to the right. That is, the last element in each row is the first element in the following

row.

We can extend the one-dimensional degradation model to two-dimensions. Let f and h be two digitized images of size $M \times N$. Then the convolution of these functions is given by

$$g(x,y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} = f(m,n)h(x-m, y-n)$$

for $x \in \{0,1,2,...,M-1\}$ and $y \in \{0,1,2,...,N-1\}$. Here we can use matrix notation to represent the degradation model

$$g = Hf + n$$

where we stack the rows of the $M \times N$ matrix

$$f = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & ... & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & ... & f(1,M-1) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & ... & f(M-1,N-1) \end{bmatrix},$$

$$g = \begin{bmatrix} g(0,0) & g(0,1) & g(0,2) & ... & g(0,N-1) \\ g(1,0) & g(1,1) & g(1,2) & ... & g(1,M-1) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ g(M-1,0) & g(M-1,1) & g(M-1,2) & ... & g(M-1,N-1) \end{bmatrix}$$

and

$$n = \begin{bmatrix} n(0,0) & n(0,1) & n(0,2) & ... & n(0,N-1) \\ n(1,0) & n(1,1) & n(1,2) & ... & n(1,M-1) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ n(M-1,0) & f(M-1,1) & n(M-1,2) & ... & n(M-1,N-1) \end{bmatrix}$$

in such a way that

$$f = \begin{bmatrix} f(0,0) \\ f(0,1) \\ . \\ . \\ . \\ f(0,N-1) \\ f(1,0) \\ f(1,1) \\ . \\ . \\ . \\ f(1,N-1) \\ . \\ . \\ . \\ . \\ . \\ f(M-1,N-1) \end{bmatrix}, \quad g = \begin{bmatrix} g(0,0) \\ g(0,1) \\ . \\ . \\ . \\ g(0,N-1) \\ g(1,0) \\ g(1,1) \\ . \\ . \\ . \\ g(1,N-1) \\ . \\ . \\ . \\ . \\ . \\ g(M-1,N-1) \end{bmatrix}, \text{ and } n = \begin{bmatrix} n(0,0) \\ n(0,1) \\ . \\ . \\ . \\ n(0,N-1) \\ n(1,0) \\ n(1,1) \\ . \\ . \\ . \\ n(1,N-1) \\ . \\ . \\ . \\ . \\ . \\ n(M-1,N-1) \end{bmatrix}.$$

To determine the structure of H, let us fix the x value. Since we need to sum over all y values, let

$$H_j = \begin{bmatrix} h(j,y_1-0) & h(j,y_1-1) & h(j,y_1-2) & ... & h(j,y_1-N+1) \\ h(j,y_2-0) & h(j,y_2-1) & h(j,y_2-2) & ... & h(j,y_2-N+1) \\ h(j,y_3-0) & h(j,y_3-1) & h(j,y_3-2) & ... & h(j,y_3-N+1) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ h(j,y_N-0) & h(j,y_N-1) & h(j,y_N-2) & ... & h(j,y_N-M+1) \end{bmatrix}$$

where $y_i \in \{0,1,2,...N-1\}$ and $i \in \{1,2,3,...,N\}$. Substituting our y_i values and taking advantage of the periodicity of h we obtain:

$$H_j = \begin{bmatrix} h(j,0) & h(j,N-1) & h(j,N-2) & ... & h(j,1) \\ h(j,1) & h(j,0) & h(j,N-1) & ... & h(j,2) \\ h(j,2) & h(j,1) & h(j,0) & ... & h(j,3) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ h(j,N-1) & h(j,N-2) & h(j,N-3) & ... & h(j,0) \end{bmatrix}.$$

a circulant matrix. Since convolution sums over all x values,

$$H = \begin{bmatrix} H_0 & H_{M-1} & H_{M-2} & ... & H_1 \\ H_1 & H_0 & H_{M-1} & ... & H_2 \\ H_2 & H_1 & H_0 & ... & H_3 \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ H_{M-1} & H_{M-2} & H_{M-3} & ... & H_0 \end{bmatrix}.$$

Thus the dimension of H is $MN \times MN$. We call each H_j a block and H is a circulant matrix. Therefore we call H a block-circulant matrix.

4.4 Diagonalization of Circulant Matrices

Let us consider again our $M \times M$ circulant matrix

$$H = \begin{bmatrix} h(0) & h(M-1) & h(M-2) & ... & h(1) \\ h(1) & h(0) & h(M-1) & ... & h(2) \\ h(2) & h(1) & h(0) & ... & h(3) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ h(M-1) & h(M-2) & h(M-3) & ... & h(0) \end{bmatrix}.$$

Let us also define a scalar function λ by

$$\lambda(k) = h(0) + h(M-1)e^{k\frac{2\pi i}{M}} + h(M-2)e^{2k\frac{2\pi i}{M}} + ... + h(1)e^{(M-1)k\frac{2\pi i}{M}}$$

and a vector w by

$$w(k) = \begin{bmatrix} 1 \\ e^{k\frac{2\pi i}{M}} \\ e^{2k\frac{2\pi i}{M}} \\ . \\ . \\ . \\ e^{(M-1)k\frac{2\pi i}{M}} \end{bmatrix}$$

for $k = 0, 1, 2, ..., M-1$. Then

$$
\begin{aligned}
Hw(k) &= \begin{bmatrix} h(0) & h(M-1) & h(M-2) & ... & h(1) \\ h(1) & h(0) & h(M-1) & ... & h(2) \\ h(2) & h(1) & h(0) & ... & h(3) \\ . & . & . & ... & . \\ . & . & . & ... & . \\ h(M-1) & h(M-2) & h(M-3) & ... & h(0) \end{bmatrix} \begin{bmatrix} 1 \\ e^{k\frac{2\pi i}{M}} \\ e^{2k\frac{2\pi i}{M}} \\ . \\ . \\ . \\ e^{(M-1)k\frac{2\pi i}{M}} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{j=0}^{M-1} h(M-j)e^{jk\frac{2\pi i}{M}} \\ \sum_{j=0}^{M-1} h(M-j+1)e^{jk\frac{2\pi i}{M}} \\ \sum_{j=0}^{M-1} h(M-j+2)e^{jk\frac{2\pi i}{M}} \\ . \\ . \\ . \\ \sum_{j=0}^{M-1} h(M-j-1)e^{jk\frac{2\pi i}{M}} \end{bmatrix} \\
&= \begin{bmatrix} (1)(\sum_{j=0}^{M-1} h(M-j)e^{jk\frac{2\pi i}{M}}) \\ (e^{k\frac{2\pi i}{M}})(\sum_{j=0}^{M-1} h(M-j+1)e^{(j-1)k\frac{2\pi i}{M}}) \\ (e^{2k\frac{2\pi i}{M}})(\sum_{j=0}^{M-1} h(M-j+2)e^{(j-2)k\frac{2\pi i}{M}}) \\ . \\ . \\ . \\ (e^{(M-1)k\frac{2\pi i}{M}})\sum_{j=0}^{M-1} h(M-j-1)e^{(j-M+1)k\frac{2\pi i}{M}} \end{bmatrix} = \begin{bmatrix} (1)\lambda(k) \\ (e^{k\frac{2\pi i}{M}})\lambda(k) \\ (e^{2k\frac{2\pi i}{M}})\lambda(k) \\ . \\ . \\ . \\ (e^{(M-1)k\frac{2\pi i}{M}})\lambda(k) \end{bmatrix} \\
&= \lambda(k)w(k)
\end{aligned}
$$

This implies that $w(k)$ is an eigenvector of H and $\lambda(k)$ is the corresponding eigenvalue. Let us form a matrix W whose columns are $w(k)$ for all $k \in \{0, 1, ..., M-1\}$. That is, let

$$W = \frac{1}{\sqrt{M}} \begin{bmatrix} w(0) & w(1) & w(2) & ... & w(M-1) \end{bmatrix}.$$

We can find a particular element $w(k, j)$ in the matrix by letting

$$w(k, j) = \frac{1}{\sqrt{M}} e^{kj\frac{2\pi i}{M}}$$

for $k, j \in \{0, 1, 2, ..., M-1\}$. The inverse of W is given by

$$W^{-1}(k,j) = \frac{1}{\sqrt{M}} e^{-kj\frac{2\pi i}{M}}.$$

Note that W is symmetric and unitary, hence $W^{-1} = \overline{W}$. For example, suppose $k, j \in \{0, 1, 2\}$. Then

$$W = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{\frac{2\pi i}{3}} & e^{2(\frac{2\pi i}{3})} \\ 1 & e^{2(\frac{2\pi i}{3})} & e^{4(\frac{2\pi i}{3})} \end{bmatrix}$$

and

$$W^{-1} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{-\frac{2\pi i}{3}} & e^{-2(\frac{2\pi i}{3})} \\ 1 & e^{-2(\frac{2\pi i}{3})} & e^{-4(\frac{2\pi i}{3})} \end{bmatrix}.$$

Due to the orthogonality of the complex numbers, we know that $e^{jk\frac{2\pi i}{M}} * e^{-kj\frac{2\pi i}{M}} = 1$. Notice that $WW^{-1} = I$ where I is the identity matrix.

We can express H as

$$H = W\Lambda W^{-1} \tag{4.1}$$

or we can write

$$\Lambda = W^{-1}HW,$$

where Λ is a diagonal matrix with $\Lambda = \text{diag}[\lambda(0), \lambda(1), ..., \lambda(N-1)]$ which shows that H can be diagonalized.

Recall that

$$g = h * f + n = Hf + n$$

and we know

$$\mathcal{F}g = (\mathcal{F}h)(\mathcal{F}f) + \mathcal{F}n$$

when we apply the discrete Fourier transform. Since $g = Hf + n$, we can replace H by (4.1) to get

$$g = W\Lambda W^{-1}f + n.$$

So

$$W^{-1}g = \Lambda W^{-1}f + W^{-1}n.$$

Notice that W^{-1} is in fact the discrete Fourier transform, so the diagonalization of the matrix H is equivalent to the discrete Fourier transform taking convolutions to products.

4.5 Diagonalization of Block-Circulant Matrices

We can diagonalize block-circulant matrices in a similar fashion, where W is a matrix of size $MN \times MN$ such that

$$W(j,m) = \frac{1}{\sqrt{MN}} w_M(j,m) w_N(k,n)$$

where

$$w_M(j,m) = e^{jm\frac{2\pi i}{M}} \text{ and } w_N(k,n) = e^{kn\frac{2\pi i}{N}}.$$

We can also define the inverse matrix W^{-1} by

$$W^{-1}(j,m) = \frac{1}{\sqrt{MN}} w_M^{-1}(j,m) w_N^{-1}(k,n)$$

where

$$w_M^{-1}(j,m) = e^{-jm\frac{2\pi i}{M}} \text{ and } w_N^{-1}(k,n) = e^{-kn\frac{2\pi i}{N}}.$$

Just like the case with the circulant matrices, we can show $WW^{-1} = I$ where I is the identity matrix and that the elements of W^{-1} are the elements of the discrete two-dimensional Fourier transform.

4.6 Inverse Filtering

We said that the degradation model of an image is

$$g = Hf + n$$

where H is a translation-invariant operator that operates on an unknown function f, n is the noise, and g is the output function. Our objective is to find a function f_0 that closely approximates f. In practice, H is assumed to be a convolution operator. When H is unknown it can be estimated using a point source. When n is unknown we can sample uniform areas of the image and find the standard deviation and mean. Statistics for the noise give us the L^2-norm $||n||^2 = E(n^2) = \sigma^2 + \mu^2$. We want to find a function f_0 that reduces the noise as much as possible. That is, since

$$g - Hf = n$$

we want to find an f_0 that minimizes

$$||g - Hf_0||^2 = ||n||^2$$

where

$$||n||^2 = n \cdot n \text{ and } ||g - Hf_0||^2 = (g - Hf_0) \cdot (g - Hf_0).$$

We can think of this as minimizing the function T where

$$T(f) = ||g - Hf||^2.$$

If we assume f_0 minimizes $T(f)$ then

$$(\partial_f T)(f_0) = \partial_f (g - Hf)(g - Hf)|_{f=f_0} = 0.$$

Notice that

$$\partial_f H|_{f_0} = \lim_{t \to 0} \frac{H(f_0 + tf) - H(f_0)}{t}$$

and since H is linear we have $\partial_f H = \lim_{t \to 0} H\left(\frac{(f_0 + tf - f_0)}{t}\right) = H(f)$. So

$$\begin{aligned} \partial_f T(f_0) =& \partial_f (g - H\hat{f})^T (g - H\hat{f})|_{f=f_0} \\ =& -2H^T(g - Hf_0). \end{aligned}$$

Since $\partial_f T(f_0) = 0$, we have

$$\begin{aligned} 0 =& -2H^T(g - Hf_0) \\ =& g - Hf_0 \end{aligned}$$

and therefore

$$f_0 = H^{-1}g.$$

In equation (4.1) we saw that H can be written as $H = W\Lambda W^{-1}$. So

$$f_0 = H^{-1}g = (W\Lambda W^{-1})^{-1}g = W\Lambda^{-1}W^{-1}g$$

thus

$$W^{-1}f_0 = \Lambda^{-1}W^{-1}g.$$

Since the elements of W^{-1} are the elements in the discrete Fourier transform and $\Lambda = H$ we can re-express this formula as

$$F_0 = \frac{G}{\mathcal{F}h}.$$

This image restoration approach is referred to as inverse filtering. Note that computational difficulties occur when H is close to zero. Since our degradation model (with the discrete Fourier transform applied) is

$$G = \mathcal{F}h + N$$

then

$$\frac{G}{\mathcal{F}h} = F + \frac{N}{\mathcal{F}h}$$

and

$$F_0 = F + \frac{N}{\mathcal{F}h}.$$

If $\mathcal{F}h$ is close to zero the term $\frac{N}{H}$ would dominate the restoration result $\mathcal{F}^{-1}\hat{F}$ [3].

4.7 Wiener Filter

In this section we reconsider the degradation model

$$g = Hf + n$$

where we want to find a f_0 that estimates the image subject to the constraint

$$||g - Hf_0||^2 = ||n||^2$$

and a smoothing condition which minimizes

$$||Pf_0||^2 = ||p * f_0||^2$$

where p is a Laplacian filter. This optimization problem is solved using Lagrange multipliers. For simplicity we present the solution in one dimension. We need $\bigtriangledown_f(||Pf||^2) - \lambda \bigtriangledown_f (||g - Hf_0||^2) = 0$ at $f = f_0$. Thus each directional derivative of $(||Pf||^2 - \lambda||g - Hf_0||^2)$ must be zero at f_0. Computing directional derivatives of a linear operator P at f_0, we obtain

$$\begin{aligned}
\partial_f(||Pf||^2)|_{f_0} &= \frac{\partial}{\partial t}\bigg|_0 ||P(f_0 + tf)||^2 \\
&= \frac{\partial}{\partial t}\bigg|_0 (P(f_0 + tf) \cdot P(f_0 + tf)) \\
&= \frac{\partial}{\partial t}\bigg|_0 ((Pf_0 + tPf) \cdot (Pf_0 + tPf) \\
&= \frac{\partial}{\partial t}\bigg|_0 (Pf_0 \cdot Pf_0 + 2tPf_0 \cdot tPf + t^2 Pf \cdot Pf) \\
&= 2Pf_0 \cdot Pf
\end{aligned}$$

Here we regard f, f_0 as vectors and P as a matrix. Thus the Lagrange multiplier condition becomes

$$Pf_0 \cdot Pf - \lambda(g - Hf_0) \cdot Hf = 0$$

for all f. Thus

$$(P^T P f_0 - \lambda H^T g + \lambda H^T H f_0) \cdot f = 0$$

for all f and hence

$$(P^T P + \lambda H^T H) f_0 - \lambda H^T g = 0.$$

Thus

$$(H^T H + K P^T P) f_0 = H^T g$$

where K is a constant. Apply the Fourier transform, using $Hf = h * f$ and $Pf = p * f$

$$(\overline{\mathcal{F}h}\mathcal{F}h + K\overline{\mathcal{F}p}\mathcal{F}p)F_0 = \overline{\mathcal{F}h}G$$

we obtain the Wiener filter

$$F_0 = \frac{\overline{\mathcal{F}h}G}{|\mathcal{F}h|^2 + K|\mathcal{F}p|^2}.$$

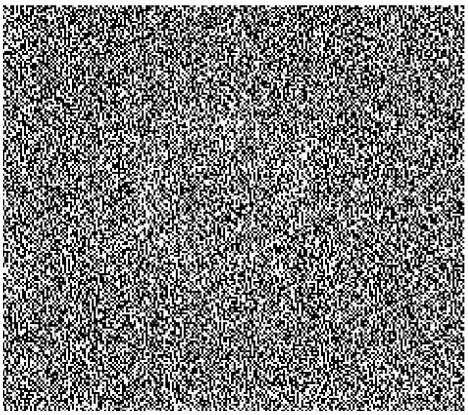
Now K is chosen to minimize the noise $||n||^2 = ||\mathcal{F}n||^2 \cong ||G - (\mathcal{F}h)F_0||^2$ using an iterative technique or through experimentation.

In Figure 4.5 (a) we have a picture of a tire. In (b) a motion blur is simulated through convolution with Gaussian noise that has $\mu = 0$ and $\sigma^2 = 0.0001$. In (c) inverse filtering is applied to restore the image. In (d) the Wiener filter is applied, producing a much clearer result.

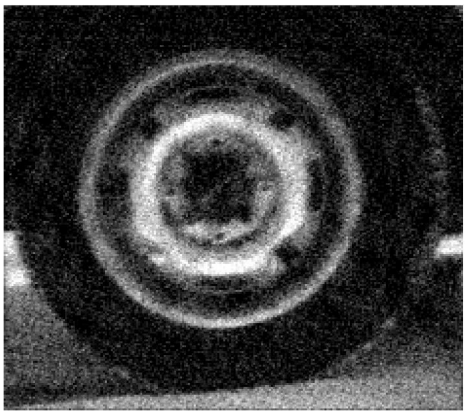
Figure 4.5

(a) Original Tire

(b) Tire with Noise and Motion Blur



(c) Tire Restored with Inverse Filtering



(d) Tire Restored with Wiener Filtering

MATLAB codes

1. **Image Negative**
   i=imread('tire.tif');
   i=255-i;
   figure; imshow(i2);

2. **Thresholding**
   i=imread('saturn.png');
   imshow(i);
   i=rgb2gray(i); % use if you need to covert an RGB image to black and white.
   figure; imshow(i)
   i=mat2gray(i); %coverts image from [0, 255] to [0,1]
   level=.7; % thresholding value you can adjust
   i2=im2bw(i, level);
   figure; imshow(i2);

3. **Gamma Correction** i=imread('cameraman.tif');
   imshow(i);
   i=mat2gray(i);
   I=imadjust(i, [ ], [ ], 1); % identity function where gamma=1
   figure; imshow(I);
   I2=imadjust(i, [ ], [ ], .5); % gamma= 0.5 brightens image
   figure; imshow(I2);
   I3=imadjust(i, [ ], [ ], 2); % gamma=2 darkens image
   figure; imshow(I3)

4. **Histogram Equalization**
   i=imread('pout.tif');
   imshow(i);
   figure; imhist(i);
   figure; histeq(i);

5. **Smoothing Filter**
   i=imread('coins.tif');
   imshow(i);
   f=(1/9)*[1 1 1; 1 1 1; 1 1 1]
   conv=imfilter(i,l);
   figure; imshow(conv);

6. **Laplacian Filter**
   i=imread('moon.tif');
   i=im2double(i);

```
imshow(i);
f=[0 1 0; 1 -4 1; 0 1 0]
conv=imfilter(i,f);
figure; imshow(conv, [ ]); title('Edges found by the Laplacian');
I=i-conv;
figure; imshow(I); title('Sharpened Image');
f2=[1 1 1; 1 -8 1; 1 1 1]
conv2=imfilter(i,f2);
figure; imshow(conv2, [ ]); title('Edges found by the Extended Laplacian');
I2=i-conv2;
figure; imshow(I2); title('Sharpened Image 2');
f3=f*7;
conv3=imfilter(i, f3);
figure; imshow(conv2, [ ]); title('Edges found by Laplacian Boost');
I3=i-conv3;
figure; imshow(I3); title('Over Sharpened Image');
```

7. Fourier Transform of an Image

```
i=imread('coins.png');
imshow(i); title('Original Image');
I=fft2(i);
S=abs(I);
figure; imshow(S, [ ]); title('Fourier Transform');
Ic=fftshift(I);
Sc=abs(Ic);
figure; imshow(Sc,[ ]); title('Centered Fourier Transform');
S2=log(1+Sc);
figure; imshow(S2, [ ]); title('Logged Centered Fourier Transform');
```

8. Adding Noise to an Image

```
f=imread('cameraman.tif');
g=imnoise(f, 'gaussian', 0, 0.02); % first number is the mean, second number is
the variance
g2=imnoise(f, 'salt & pepper,' .1); % .1 corresponds to the percent of the image
affected with noise
imshow(f);
figure; imhist(f); title('Original Histogram');
figure; imshow(g); title('Gaussian Noise');
figure; imshow(g2); title('Salt & Pepper Noise');
figure; imhist(g); title('Noisy Gaussian Histogram');
figure; imhist(g2); title('Noisy Salt & Pepper Histogram');
```

8. Wiener Filter

```
close all;
C=imread('tire.tif');
C=im2double(C);

%%Adding Motion Blur%%
LEN =40;
THETA = 11;
PSF = fspecial('motion', LEN, THETA);
blur = imfilter(C, PSF, 'conv', 'circular');
%%Using Weiner to Restore Image%%
wnr1 = deconvwnr(blur, PSF, 0);

%%Adding Noise to Blurred Image%%
noise_mean = 0;
noise_var = 0.0001;
blurred_noisy = imnoise(blur, 'gaussian', ... noise_mean, noise_var);

%%Using Weiner to Restore Image with K=0%%
wnr2 = deconvwnr(blurred_noisy, PSF);

%%Restoring Image with Estimated K%%
signal_var = var(C(:));
wnr3 = deconvwnr(blurred_noisy, PSF, noise_var / signal_var);

%%Images without Noise%%
imshow(C); title('Tire');
figure; imshow(blur); title('Motion-Blurred Image');
figure; imshow(wnr1); title('Restored Image');
i=abs(wnr1-C);
figure; imshow(i);
%%Images with Noise%%
figure; imshow(blurred_noisy); title('Motion-Blurred Image with Added Noise');
figure; imshow(wnr2); title('Restored Image with K=0');
figure; imshow(wnr3); title('Restored Image with Estimated K');
```

REFERENCES

[1] Diliff (2006). *Clock Tower-Palace of Westminster, London-September 2006*. Wikimedia Commons, http://commons.wikimedia.org/wiki/File:Clock_Tower_-_Palace_of_Westminster,_London_-_September_2006.jpg.

[2] Folland, G. (1992). *Fourier Analysis and its Applications*. Wadsworth & Brooks/Cole Mathematics. Brooks/Cole Publishing Company, California.

[3] Gonzalez, R. and Woods, R. (1993). *Digital Image Processing*. Addison-Wesley Publishing Company.

[4] Gonzalez, R. and Woods, R. (2004). *Digital Image Processing using MATLAB*. Prentice Hall.

[5] Guard, U. S. C. (2013). *USCG Cape Lookout*. Public Domain. Wikimedia Commons, http://commons.wikimedia.org/wiki/File:USCGCape_Lookout.jpg.

[6] Jcwf (2006). *Panda Closeup*. Public Domain. Wikimedia Commons, http://commons.wikimedia.org/wiki/File:Panda_closeup.jpg.

[7] Murugappan, S. (2012). *Eiffel Tower at Night*. Public Domain. Wikimedia Commons, http://commons.wikimedia.org/wiki/File:Eiffel_tower_at_night_WLM.JPG.

[8] Sapiro, G. (2013). *Image and Video Processing: From Mars to Hollywood with a Stop at the Hospital*. Duke University.