5

Chapter Five
Image Degradation & Restoration

salt & pepper noise

Gaussian & Speckle noise

periodic noise

blur

5.1 Image Degradation Model

Image restoration concerns about remove or reduce a degradation which have occurred during image acquisition. The reasons of degradation may include:

- noise: which are errors in the image pixel value
- blur: which is the distortion or smearing in the image because of camera motion or out of focus.

The degraded image could be successfully restored using two domains,

- neighbourhood operations, or
- frequency domain processes

let $f(x, y)$ is an image and $h(x, y)$ is a spatial filter, so

$$g(x,y) = f(x,y) * h(x,y)$$

is a convolution formula that has some form of degradation. A noise must be considered in this formula which can be modelled as an additive function to the convolution formula as follows,

$$g(x,y) = f(x,y) * h(x,y) + n(x,y)$$

in frequency domain becomes,

$$G(U,V) = F(U,V) .H(U,V) + N(U,V)$$

5.2 Types of Noise:

Noise is defined as any degradation in the image signal caused by external disturbance such as:

- Wireless transmission
- Satellite transmission
- Network cable transmission

Errors appear on the image output in different ways depending on the type of disturbance in the signal. If we know what type of errors in the image, we can choose the most appropriate method for reducing the effects. Cleaning an image corrupted by noise is an important area of image restoration.

5.2.1 Salt and Pepper Noise

Also, called impulse noise, shot noise, or binary noise. This degradation can be caused by sharp and sudden disturbances in the image signal. It is randomly scattered white or black or both pixels over the image.

Ex//

```
> > t = imread( 'twin.tif');
>> tg = rgb2gray (t);
>> tn= imnoise ( tg , 'salt&pepper' , 0.2 );        % default noise is 10%
```

(a) Original image (b) With added salt & pepper noise

Fig. 5.1 Twin image is degraded by %20 of salt and pepper noise

5.2.2 Gaussian Noise

It is an idealized form of white noise, it is caused by random fluctuations in the signal, we can observe white noise by watching a T.V which is slightly mistuned to a particular channel. It is normally distributed and for example if I is an image matrix, N is a white or Gaussian noise, so

$I' = I + N$ *where, I' is a noisy image*

The "Gaussian" parameter can also take optional values giving the mean and variance of the noise, the default values are mean (= 0) and standard deviation ($\sigma = 0.01$)

```
>> tgn = imnoise ( tg , 'gaussian' )
```

5.2.3 Speckle Noise

It is also called a multiplicative noise and it is a major problem in some radar applications. Like Gaussian noise, speckle can be modelled by random values multiplied by pixel values, $I\,(1+N)$. So,

$I' = I + IN = I(1 + N)$

N: consists of normally distributed values with mean 0 and its default value is 0.04.

>> tsp = imnoise (tg , 'speckle')

(a) Gaussian noise (b) Speckle noise

Fig. 5.2 Twins image is degraded by Gaussian and Speckle noise

5.2.4 Periodic Noise

This type of degradation has a global effect, and it is not easy to remove or decrease its effect using traditional cleaning methods. If image signal is subjected to a periodic rather than random disturbance, we might obtain a corrupted image by periodic noise. Periodic noise may occur in the image because of:

1. Image equipment
2. Network equipment
3. External disturbance of repeating nature like electric motor

The effect is look like bars over the image. All other noises like; salt & peppers or Gaussian noise could be removed using spatial filters while periodic noise requires the use of frequency domain technique to decrease the degradation effect or remove it. This is because the other forms of noise can be

modelled as local degradations while periodic noise has a global effect. We can simulate the worst effect of periodic noise using any periodic function like, exponent, cosine, or sine function:

```
>> s = size ( tg )
>> [ x , y ] = meshgrid ( 1 : s(1 ),  1: s(2) )
>> P =  1 + sine ( x/3 + y/5)
>> tpk = ( im2double ( tg ) + P/2 ) / 2
```

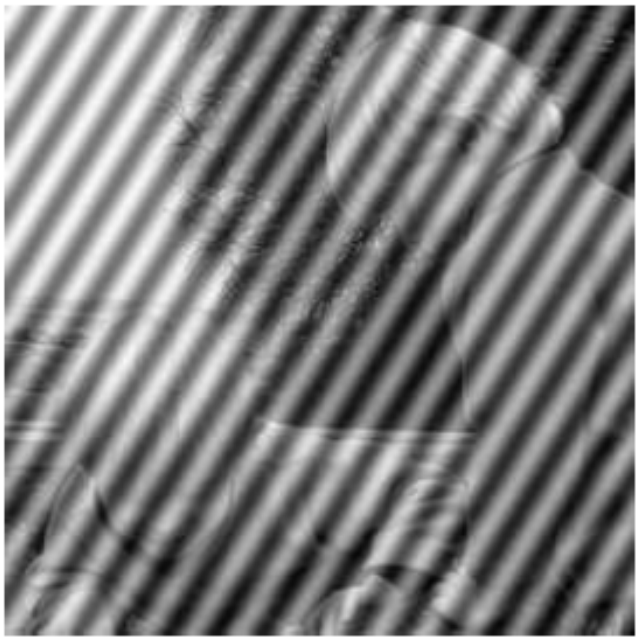


Fig 5.3 Twin image corrupted by periodic noise

5.3 Image Restoration: Noise Cleaning

5.3.1 Salt & pepper Noise Removal

The sudden change in the image pixel value for this kind of noise can be represented as a high frequency change and this high frequency can be blocked or filtered using ***low pass filters***. The next sections will discuss many filtering techniques for cleaning different noisy image

- **Mean Filter:** as shown in the figure 5.4, the filtered image is not well cleaned as the noise is smeared over the image. The effect is even more pronounced if we use a large averaging filter.

```
>> fav = fspcial ('average')
>> tav = filter2 ( fav, tn )
```

- **Median Filter:** a median filter is an example of a non-linear spatial filter. It is widely used as it is very effective at removing noise while preserving edges. The median is calculated by first sorting all the pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value. If window elements are even, the middle is computed using the mean of the adjacent two pixels.

50	65	52
63	**255**	58
61	60	57

50 52 57 58 **60** 61 63 65 255

>> tmdn = medfilt2 (tn)

(a) 3×3 averaging

(b) 7×7 averaging

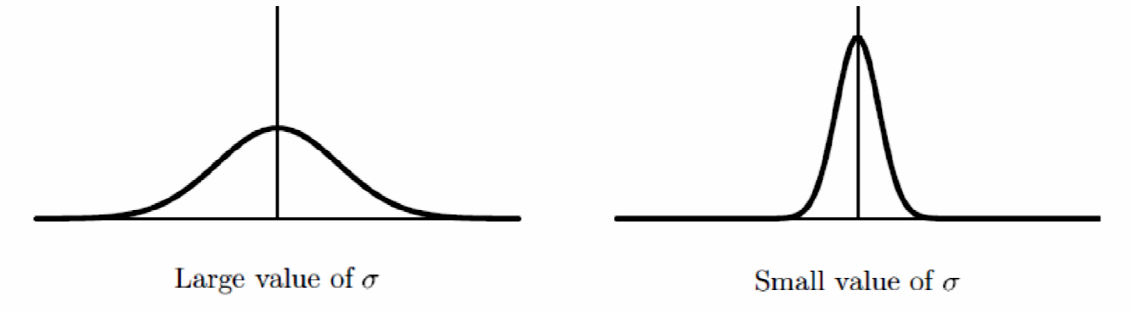
(a) Using `medfilt2` twice

(b) using a 5×5 median filter

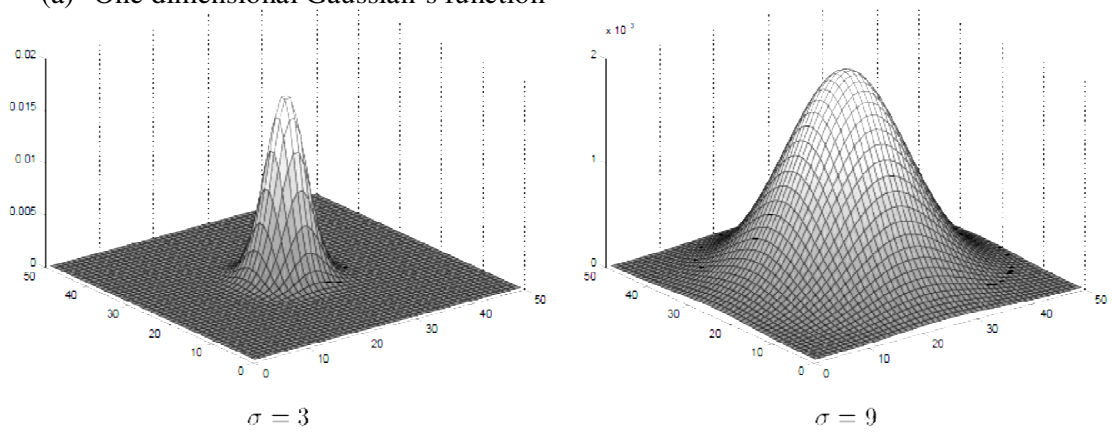
Fig 5.4 Noisy Twin image is cleaned using averaging and median filters

- **Gaussian Filter:** this filter is a class of low pass filters, and as LPFs does, it tend to smooth the image. It is based on the Gaussian probability distribution function, for two dimensions image,

$$f(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$



(a) One dimensional Gaussian's function



(b) Two dimensional Gaussian's function

Fig. 5.5 (a) and (b)

Ex//

g = fspecial('gaussian',3,5)

g = 0.1096 0.1118 0.1096
0.1118 0.1141 0.1118
0.1096 0.1118 0.1096

g = fspecial('gaussian',3,3)

g = 0.1070 0.1131 0.1070
0.1131 0.1196 0.1131
0.1070 0.1131 0.1070

g = fspecial('gaussian',4,3)

g = 0.0558 0.0623 0.0623 0.0558
0.0623 0.0696 0.0696 0.0623
0.0623 0.0696 0.0696 0.0623
0.0558 0.0623 0.0623 0.0558

$5 \times 5, \sigma = 0.5$ $5 \times 5, \sigma = 2$

$11 \times 11, \sigma = 1$ $11 \times 11, \sigma = 5$

Fig. 5.6 effects of different Gaussian filters on image

HW// The array $I_{8\times8}$ represents a small grayscale image. Compute the "valid" filtered images that result when the image is convolved with each of the mask (a) to (h) below.

$$\text{(a)}\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \text{(b)}\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix} \quad \text{(c)}\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{(d)}\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$

$$\text{(e)}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{(f)}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{(g)}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{(h)}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$I =$

20	20	20	10	10	10	10	10
20	20	20	10	10	10	10	20
20	20	20	10	10	10	10	20
20	20	10	10	10	10	20	10
20	10	10	10	10	10	20	10
10	10	10	20	10	10	20	10
10	10	10	10	10	10	10	10
20	10	20	20	10	10	20	20

Check your answers with MATLAB.

5.3.2 Gaussian Noise Removal

- **Mean of Multi-noisy Images:** With Gaussian noise instead of one corrupted image, we have many different Gaussian noise corruption images. If the satellite passed over the same spot many times, we will obtain many different images of the same place. Another example is in microscopy, we might take many different images of the same object.

Example:

Suppose we have 100 copies of Gaussian noisy image, so it's still true to express the formula,

M + N_i

where M is the matrix of original values, and N_i is a matrix of normally with mean 0. We can find the mean M' of these images by the usual ado

$$\begin{aligned} M' &= \frac{1}{100}\sum_{i=1}^{100}(M + N_i) \\ &= \frac{1}{100}\sum_{i=1}^{100} M + \frac{1}{100}\sum_{i=1}^{100} N_i \\ &= M + \frac{1}{100}\sum_{i=1}^{100} N_i \end{aligned}$$

Since N_i is normally distributed with mean 0, it can be readily shown tha will be close to zero—the greater the number of N_i's; the closer to zero.

$$M' \approx M$$

and the approximation is closer for larger number of images $M + N_i$.

```
>> s = size ( tg )
>> tg10 = zeros ( s(1) , s(2), 10 )
>> for i=1: 10
>> tg10 ( : , : , i) = imnoise ( tg , 'gaussian' )
>> end ;      tavrg = mean ( tg10, 3)          % See figure 5.7
```

- **Averaging Filter:** The Gaussian noise has ***mean = 0***, then we expect that mean filter would average noise to zero. The larger size of the filter mask, the closer to zero the value of the mean. Unfortunately mean filter tends to blur an image, however if we accept blurring against noise reduction. This method can be applied.

(a) 10 images (b) 100 images

Fig. 5.7 Gaussian noise removal using the mean of multi noisy images

(a) 4×3 averaging (b) 5×5 averaging

Fig. 5.8 Gaussian noise removal using mean filter

5.3.3 Periodic Noise Removal

- The Ring Filter

We have discussed how to create periodic noise bars and how to add them into an image using periodic function like ***sine***. In this section we shall remove or decrease the degradation effect of this noise by designing frequency domain filters. The filtering in frequency domain like LPF and HPF are already discussed in section . Periodic noise in spatial domain would be appeared as two shiny spikes in frequency domain. The locations of those two spikes are completely related to (X and Y) values in the periodic function, see figure 5.9. The next step is to design a Band Reject Filter, or commonly called Ring filter to reject the spikes. The next step is inversing the FD filtered image back it again into spatial domain. See the block diagram in Fig. 5.10.

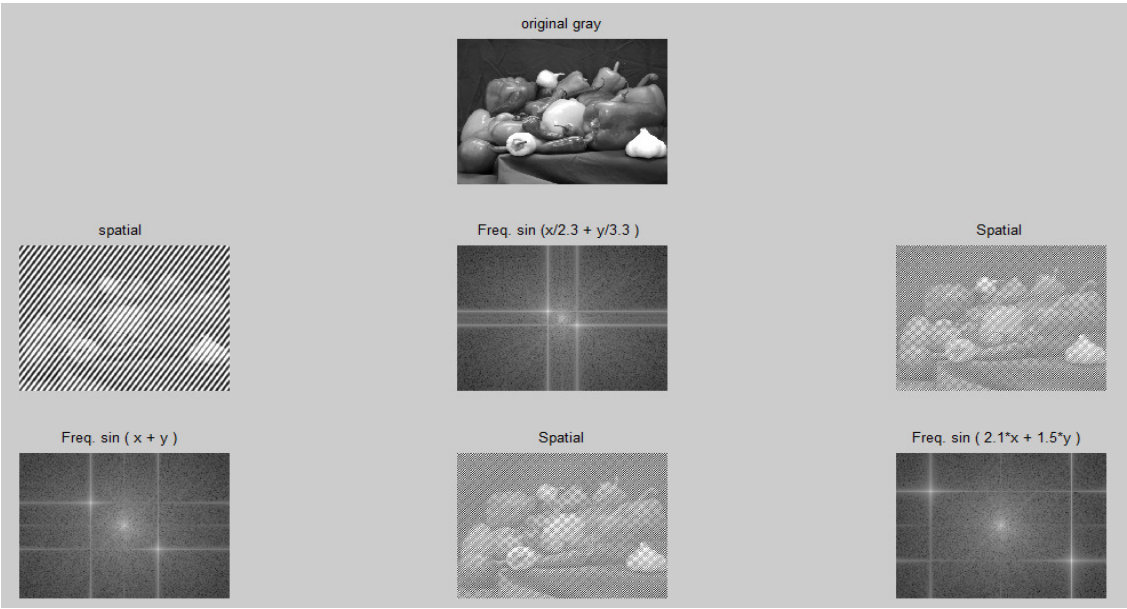


Fig. 5.9 different degradation images depending on sine (x + y) function

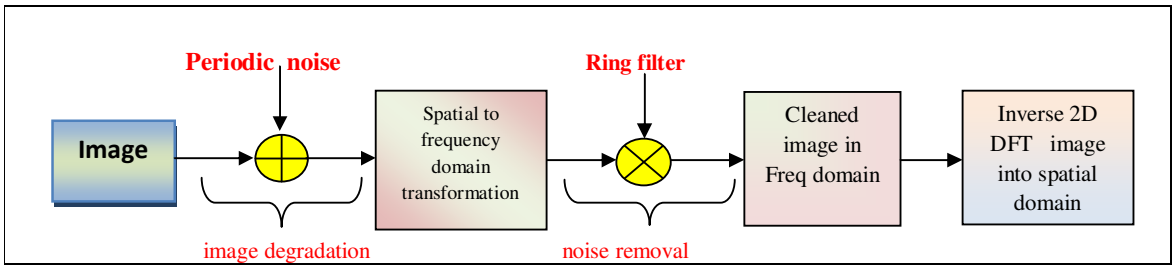


Fig. 5.10 image restoration/ periodic noise removal in frequency domain

Example: This example demonstrates periodic noise addition and removal using Band Reject Filter or commonly known a Ring Filter

```
clc; close all; clear all

 I =  imread ('C:\Program
              Files\MATLAB\R2008a\toolbox\images\imdemos\pears.png');
 I = rgb2gray(I);

       [r,c] = size (I);
       [x, y] = meshgrid(1:c, 1:r);


   p1 = 1 + sin ( x +  y );
   I2 =  im2double(I) + p1 ;
   tgpf = fftshift ( fft2 (I2) );


subplot(2,3,1); imshow(mat2gray (I*1.2)) ;title ('original gray')  ;

subplot(2,3,2); imshow((I2/2));title('noisy image in Spatial domain');

subplot(2,3,3);imshow (mat2gray( log ( abs(tgpf) ) ) ) ; title
                            ('noisy image in Freq. domain  ')  ;



    z = sqrt ( ( x - c/2 ).^2 + ( y - r/2 ).^2 );
    F= ( z < 135 | z > 145 );
    resf = tgpf .* F ;
    resi = ifft2 ( resf );


subplot(2,3,4);imshow (mat2gray( log (1+ abs(resf) ) ) ); title ('noisy
image X Ring filter  ')  ;
subplot  (2,3,5);imshow (mat2gray ( log (1+ abs(resi) ) ) ) ; title
('F= ( z < 135 | z > 145 );')  ;


    F2= ( z < 20 | z > 190 );
    resf2 = tgpf .* F2 ;
    resi2 = ifft2 ( resf2 );


subplot(2,3,6);imshow (mat2gray ( log (1+ abs(resi2) ) ) ) ; title
('F2= ( z < 20 | z > 190 ); ')  ;
```

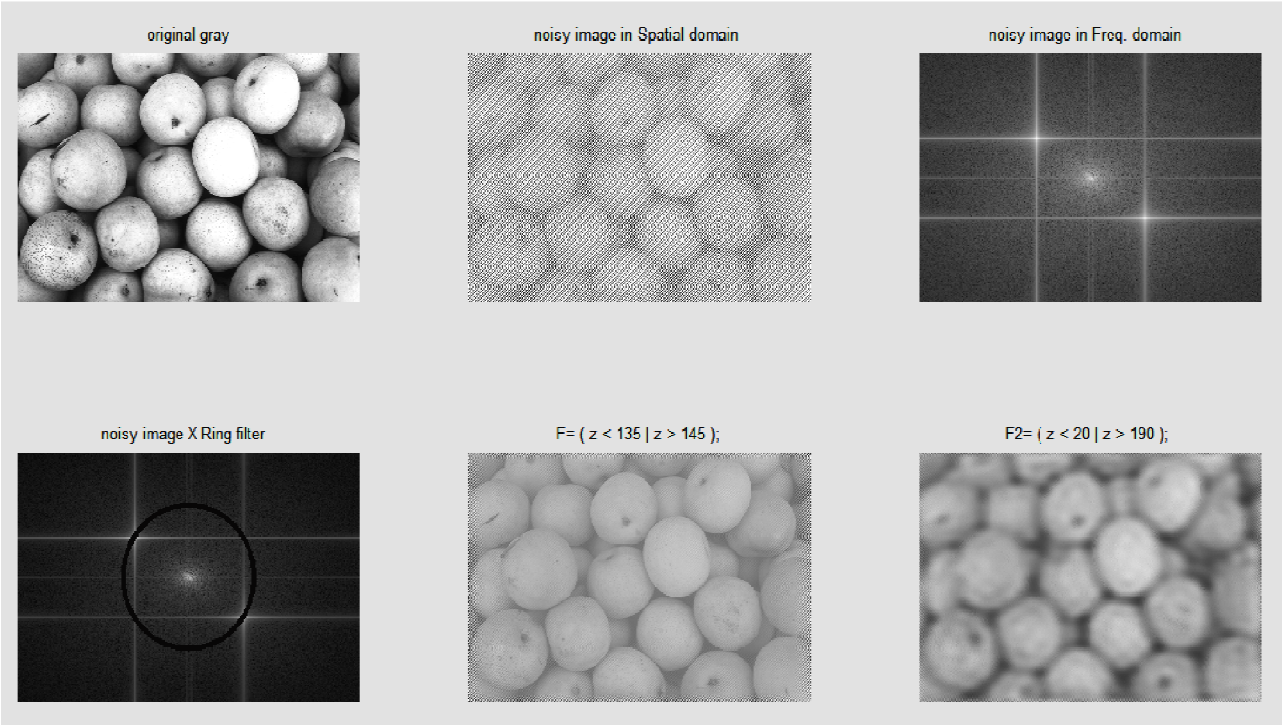


Fig. 5.11 Pears image is degraded by periodic noise and cleaned by Ring filter

- The Notch Filter
  Another method can be used to clean the noisy image in the frequency domain is by setting the spike intersection lines to zero, i.e.; the interested row(s) and column(s).

HW//
Apply Notch filter method to clean the pears noisy image above and compare the result with the ring filter.

5.3.4 Winner Filter:

We know that the Gaussian noise (*N*) is normally distributed with mean ($\mu = 0$). Here, if we have an image x is filtered with a filter F and corrupted by an additive or Gaussian noise, the linearity of the Fourier transform is,

$$Y(i,j) = X(i,j) \,.\, F(i,j) + N(i,j)$$

$$X(i,j) = [\, Y(i,j) - N(i,j) \,] \,/\, F(i,j)$$

The presence of noise can have a catastrophic effect on the inverse filtering where the noise can completely dominate the output and making the direct inverse impossible. Since we are dealing with the additive noise, the noisy image M' can be written as: $M' = M + N$ and if R represents the restored image, so our target is making R value close to M. In other words we want the result of this formula:

$$\sum (m_{i,j} - r_{i,j})^2$$

to be zero or close to zero. Filters which operates on this principle of Least Squares are called Winner filters, we can obtain x by

$$X(i,j) = \left[\frac{1}{F\,(i,j)} \frac{|F(i,j)|^2}{|F(i,j)|^2 + k}\right] Y(i,j)$$

Where k, is a constant and it is used to approximate the amount of noise. If the variance σ^2 of the noise is known, then $k = 2\sigma^2$ otherwise, k can be chosen by trail and error to yield to the best result. Note that when k = zero, then the equation will be reduced to

$$X(i,j) = \left[\frac{Y(i,j)}{F\,(i,j)}\right]$$

HW//
Write a MATLAB code to apply Winner formula on your Gaussian noisy image. Explain if the restored image is satisfied or not.

5.4 Motion Deblurring

Again the DFT of any filtered image could be expressed as:

$Y\ (i,j) = x(i,j)\ .\ F\ (i,j)$

So, if we are given the F(i,j) and Y(i,j),we should be able to recover the DFT of the original image X(i,j) by:

$x(i,j) = Y\ (i,j)\ /\ F\ (i,j)$

But as mentioned before the result, x(i,j) may be very large values and eventually will be dominated because of some small elements of filter F(i,j). There are two different methods to overcome this problem.

1st : Apply a low pass filter to the division

$$x(i,j) = \frac{Y\ (i,j)}{F\ (i,j)}\ L(i,j)$$

2nd : Using "division constrained" or threshold value d, so if the $|F(i,j)| < d$, we don't perform a division but just keep the original Y(i,j) value, thus

$$x(i,j) = \frac{Y\ (i,j)}{F\ (i,j)} \quad \text{..... if } |F(i,j)| \geq d$$

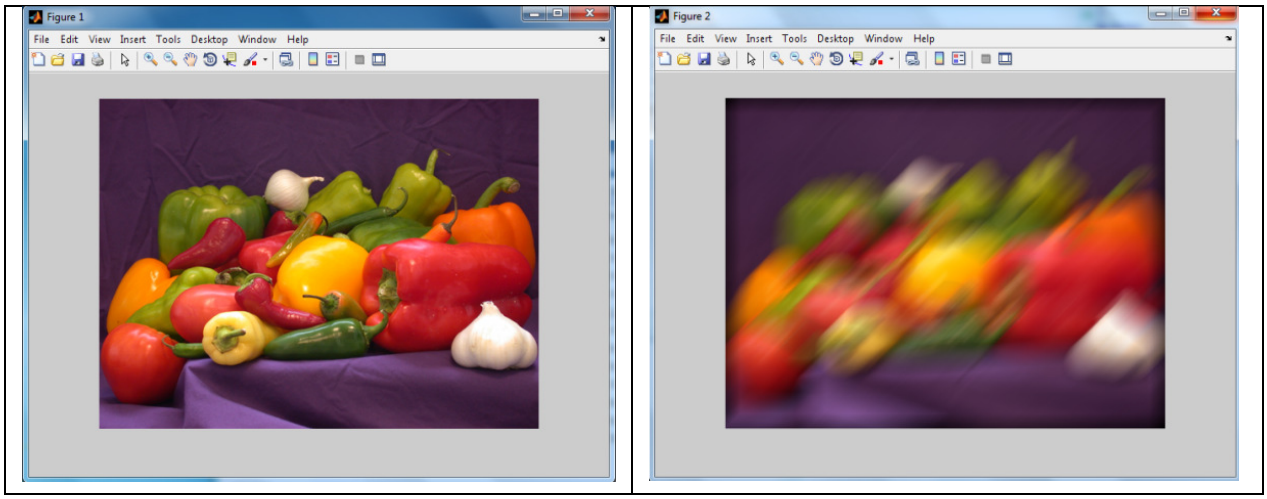
or,

$$x(i,j) = Y\ (i,j) \quad \text{..... if } |F(i,j)| < d$$

The removal of blur caused by motion would be a special case of filter inverse in frequency domain.

Ex:

```
originalRGB = imread ( 'peppers.png' );
h = fspecial( 'motion', 50, 45 );
filteredRGB = imfilter (originalRGB,h);
figure, imshow (originalRGB), figure,
imshow (filteredRGB)
```



EX: The second method can be implemented in MATLAB code below

$$x(i,j) = \frac{Y(i,j)}{F(i,j)} \quad \text{..... if } |F(i,j)| \geq d$$

or,

$$x(i,j) = Y(i,j) \quad \text{..... if } |F(i,j)| < d$$

```
clc, close all , clear all;
n=imread ('image.jpg');
ng= rgb2gray (n);
[r,c]= size (ng)

blur= fspecial ('motion', 15);
nb= imfilter (ng, blur);


fr= zeros(1:r, 1:c);
fr(1, 1:15) = blur;

d= 0.12;
frf= fft2( fr );
frf ( abs ( frf ) < d ) =1 ;
yi = ifft2 ( ( fft2( nb ) ) ./ frf  );
imshow uint8 ( yi ) ;
```

HW//

Write MATLAB code to implement the two methods of de-blurring on your selected image. Use different threshold values(d) and compare the result of the two methods.