Image Restoration

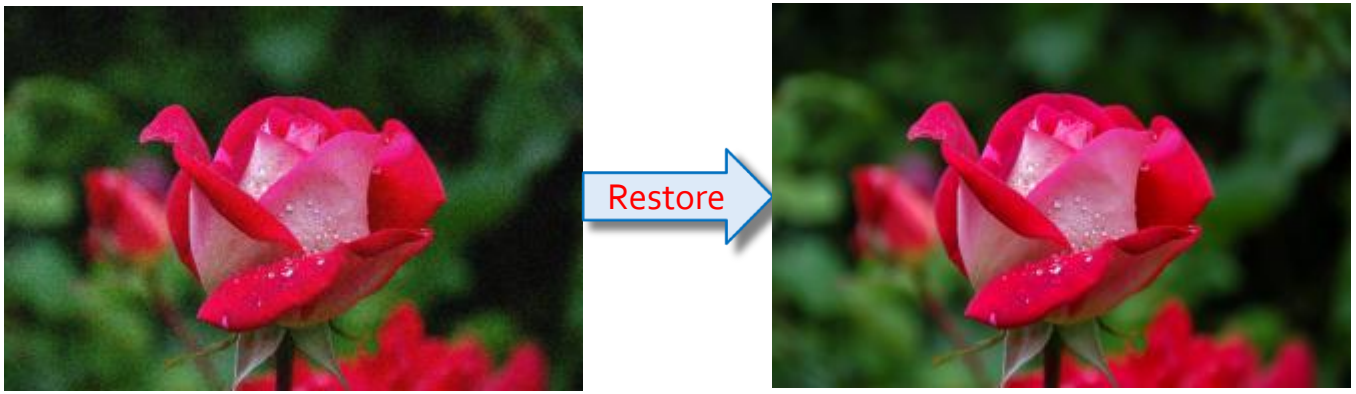
Dr. Su Su Maung
CEIT Department
Yangon Technological University

Outline of Lecture5

- Introduction
- A model of image degradation
- Different types of noise
- Cleaning different types of noise

What is Image Restoration?

- **Image Restoration** is the operation of taking a corrupt, noisy image and estimating the clean, original image.

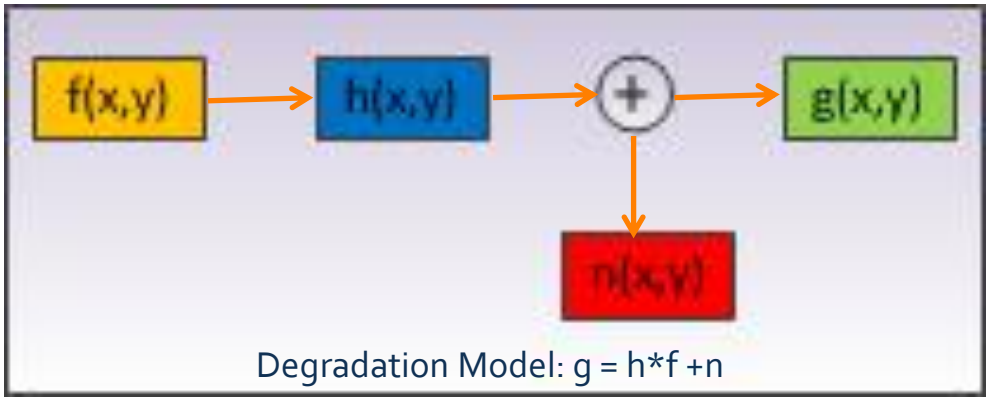


Introduction

- Image restoration concerns the removal or reduction of degradations occurred during the acquisition of the image.
- Such degradations may include noise, errors in the pixel values, or optical effects such as out of focus blurring, or blurring due to camera motion.
- Restoration techniques can be performed neighbourhood operations, while others require the use of frequency domain processes.

A model of image degradation

- In the spatial domain, we have an image f(x,y) , and a spatial filter h(x,y) for which convolution with the image results in some form of degradation.
- For example, if h(x,y) consists of a single line of ones, the result of the convolution will be a motion blur in the direction of the line.



Degradation Model: g = h*f +n

A model of image degradation

- Thus if n(x,y) represents random errors which may occur, we have as our degraded image:

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

- We can perform the same operations in the frequency domain, where convolution is replaced by multiplication, and addition remains as addition

$$G(i,j) = F(i,j)H(i,j) + N(i,j)$$

A model of image degradation

```
x=data.camera()
a=np.array([[0,0,0,0,0],[0,0,0,0,0],[1,1,1,1,1],[0,0,0,0,0],[0,0,0,0,0]])
b=ndi.convolve(x,a,mode='constant')
io.imshow(b)
```

Noise

- We may define noise any degradation in the image signal, caused by external disturbance.
- If an image is being sent electronically from one place to another, via satellite or wireless transmission, or through networked cable, we may expect errors to occur in the image signal.
- Depending on the type of disturbance, The errors will appear on the image output in different way.
- If we know what type of errors , then we know the type of noise on the image; hence we can choose the most appropriate method for reducing the effects.
- Cleaning an image corrupted by noise is an important area of image restoration.

different noise types

- Salt and pepper noise
- Gaussian noise
- Speckle noise
- Periodic noise

Salt and pepper noise

- Also called impulse noise, shot noise, or binary noise.
- This degradation can be caused by sharp, sudden disturbances in the image signal; its appearance is randomly scattered white or black (or both) pixels over the image.
- To add noise, we use the Python function noise.random_noise from the util module of skimage, which takes a number of different parameters. To add salt and pepper noise:

```
import skimage.util.noise as noise
imgn=noise.random_noise(img,mode = 's&p')
imgn2=noise.random_noise(img,mode = 's&p',amount=0.2)
```

- The above code would produce an image with 20% of its pixels corrupted by salt and pepper noise.

Salt and pepper noise (cont.)

Original image

Image with salt & pepper noise

```
import skimage.io as io
import skimage.util.noise as noise
img=io.imread('images/redrose.jpg')
imgn=noise.random_noise(img,'s&p',amount=0.2)
io.imshow(imgn)
```

Gaussian noise

- Gaussian noise is an idealized form of white noise, which is caused by random fluctuations in the signal.
- We can observe white noise by watching a television which is slightly mistuned to a particular channel.
- Gaussian noise is white noise which is normally distributed.
- If the image is represented as I, and the Gaussian noise by N, a matrix whose elements are normally distributed, then we can model a noisy image by simply adding the two:

$$I + N$$

```
import skimage.util.noise as noise
gg=noise.random_noise(img,'gaussian');
```

Gaussian noise (cont.)

Original image

Image with Gaussian noise

```
import skimage.io as io
import skimage.util.noise as noise
img=io.imread('images/jasmine.jpg')
imgn=noise.random_noise(img,'gaussian')
io.imshow(imgn)
```

Speckle noise

- speckle noise (or more simply just speckle) can be modelled by random values multiplied by pixel values, hence it is also called multiplicative noise.
- where I is the image matrix, and N consists of normally distributed values with mean o.

$$I(1 + N)$$

```
gs = noise.random_noise(g,'speckle')
```

- Speckle noise is a major problem in some radar applications.

Speckle noise n(cont.)

Original image

Image with speckle noise

Although Gaussian noise and speckle noise appear superficially similar, they are formed by two totally different methods, and require different approaches for their removal.

```
import skimage.io as io
import skimage.util.noise as noise
img=io.imread('images/jasmine.jpg')
imgn=noise.random_noise(img,'speckle')
io.imshow(imgn)
```

Periodic noise

- If the image signal is subject to a periodic, rather than a random disturbance, we might obtain an image corrupted by periodic noise.
- The effect is of bars over the image.
- it is quite easy to create our own, by adding a periodic matrix (using a trigonometric function), to our image:

```
r,c=img.shape
p=np.zeros(r,c)
x,y=np.mgrid[0:r,0:c].astype('float32')
p=np.sin(x/3+y/3)+1.0
imgp=(2*util.img_as_float(img)+p/2)/3
```

periodic matrix

- Salt and pepper noise, Gaussian noise and speckle noise can all be cleaned by using spatial filtering techniques. Periodic noise, however, requires the use of frequency domain filtering. This is because whereas the other forms of noise can be modelled as local degradations, periodic noise is a global effect.

Periodic noise (cont.)

Original image

The image corrupted by periodic noise

```
import skimage.util as util
img=io.imread('images/jasmine3.jpg')
r,c,z=img.shape
p=np.zeros((r,c,z))
x,y=np.mgrid[0:r,0:c].astype('float32')
p[:,:,0]=np.sin(x/3+y/3)+1.0
p[:,:,1]=np.sin(x/3+y/3)+1.0
p[:,:,2]=np.sin(x/3+y/3)+1.0
imgp=(2*util.img_as_float(img)+p/2)/3
```

Cleaning salt and pepper noise

- Low pass filtering
- Median filtering

Low pass filtering

- Pixels corrupted by salt and pepper noise are high frequency components of an image, and using a low-pass filter should reduce them.
- In Python, median filtering is implemented by the uniform_filter method in the ndimage module of scipy:

5x5 averaging

7x7 averaging

```
Import scipy.ndimage as ndi
Img5=ndi.uniform_filter(img,5)
Img7=ndi.uniform_filter(img,7)
```

Median filtering

- Median filtering seems almost tailor-made for removal of salt and pepper noise.
- The median of a set is the middle value when they are sorted. If there are an even number of values, the median is the mean of the middle two. A median filter is an example of a non-linear spatial filter.
- The operation of obtaining the median means that very large or very small values, noisy values, will end up at the top or bottom of the sorted list. Thus the median will in general replace a noisy value with one closer to its surroundings.

Median filtering (cont.)

- In Python, median filtering is implemented by the median_filter method in the ndimage module of scipy:

```
Img3=ndi.median_filter(img,3)
Img5=ndi.median_filter(img,5)
```

The result is a vast improvement on using averaging filters.

Cleaning salt & pepper noise with median filters of size 3x3 and 5x5

Cleaning Gaussian noise

- Image averaging
- Average filtering

Image averaging

- It may sometimes happen that instead of just one image corrupted with Gaussian noise, we have many different copies of it.
- An example is satellite imaging; if a satellite passes 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. In such a case a very simple approach to cleaning Gaussian noise is to simply take the average, the mean, of all the images.

Image averaging (cont.)

- To see why this works, suppose we have 100 copies of our image, each with noise; then the i-th noisy image will be:

$$M + N_i$$

- where M is the matrix of original values, and N_i is a matrix of normally distributed random values with mean 0. We can find the mean M′ of these images by the usual add and divide method:

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

Image averaging (cont.)

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

$$M' \cong M$$

```
# Image averaging to remove Gaussian noise
img = io.imread(images/jasmine.jpg')
x,y,z=img.shape
t=np.zeros((x,y,3,10))
for i in range(10):
   t[:,:,:,i]=noise.random_noise(img,'gaussian')
t=np.mean(t,3)
io.imshow(t)
io.imsave('jasmine1_img_avg10.jpg',t)
```

Image averaging (cont.)

5 images

10 images

100 images

Image averaging to remove Gaussian noise

Average filtering

- If the Gaussian noise has mean 0, then we would expect that an average filter would average the noise to 0.
- The larger the size of the filter mask, the closer to zero.
- Unfortunately, averaging tends to blur an image.

5x5 averaging

7x7 averaging

```
Import scipy.ndimage as ndi
Img5=ndi.uniform_filter(img,5)
Img7=ndi.uniform_filter(img,7)
```

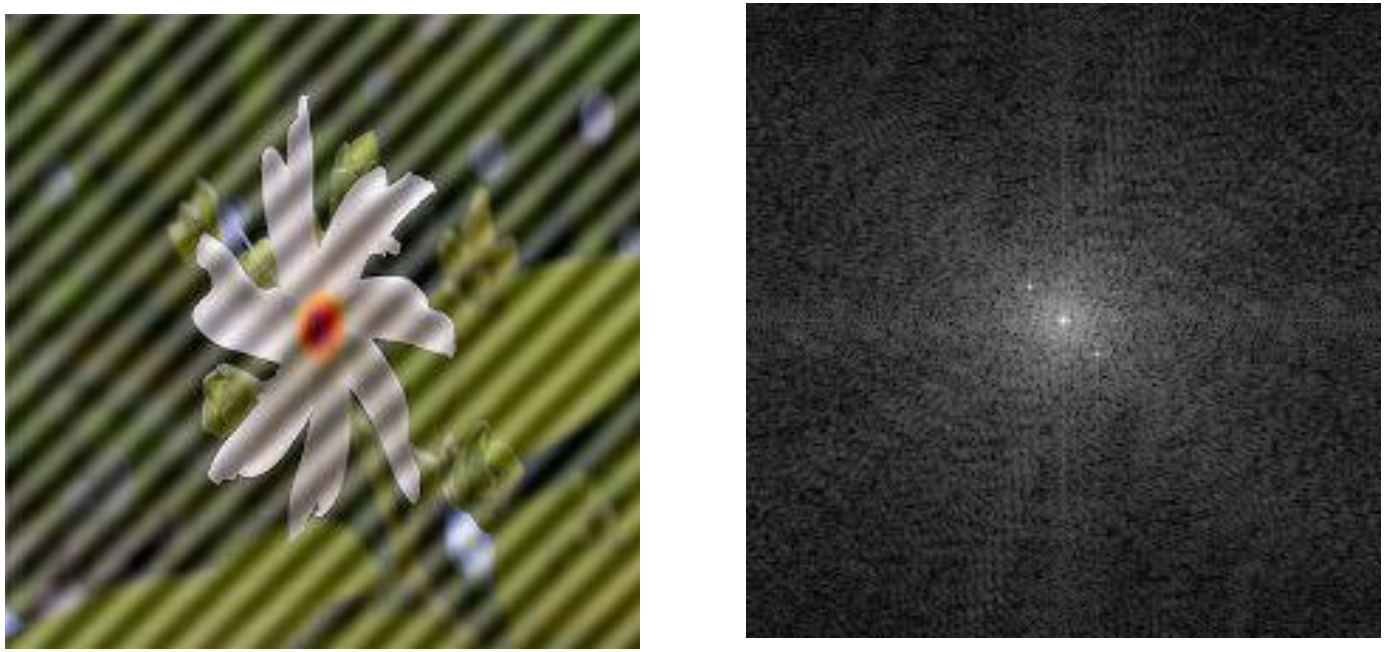
Removal of periodic noise

- Periodic noise may occur if the imaging equipment (the acquisition or networking hardware) is subject to electronic disturbance of a repeating nature, such as may be caused by an electric motor.
- We can easily create periodic noise by overlaying an image with a trigonometric function:

```
r,c=img.shape
p=np.zeros(r,c)
x,y=np.mgrid[0:r,0:c].astype('float32')
p=np.sin(x/3+y/3)+1.0
imgp=(2*util.img_as_float(img)+p/2)/3
```

original image

Removal of periodic noise (cont.)



The image (225 x 225) with periodic noise, and its Fourier transform

Removal of periodic noise (cont.)

- The extra two spikes away from the center correspond to the noise just added.
- In general the tighter the period of the noise, the further from the center the two spikes will be. This is because a small period corresponds to a high frequency (large change over a small distance), and is therefore further away from the center of the shifted transform.
- We will now remove these extra spikes, and invert the result. The first step is to find their position. Since they will have the largest maximum value after the DC coefficient, which is at position (112, 112)

Removal of periodic noise (cont.)

- we simply have to find the maxima of all absolute values except the DC coefficient., we find that the spikes have row, column values of(100,100) and (124,124). These have the same distance from the centre: 16.97.

```
from numpy.fft import ifft2,fft2,fftshift
imgp=co.rgb2hsv(imgp)
imgf=fftshift(fft2(imgp[:,:,2]))
temp=ex.rescale_intensity(np.log(1+np.abs(imgf)),out_range=(0,1))
imgf2=util.img_as_ubyte(temp)
imgf2[112,112]=0
i,j=np.where(imgf2==imgf2.max())
I,j
(array([100, 124], dtype=int64), array([100, 124], dtype=int64))
d=(i-112)**2+(j-112)**2
R=np.sqrt(d[1])
16.97056274847714n    % sqrt(288) = 16.97
```

Removal of periodic noise (cont.)

- There are two methods we can use to eliminate the spikes:
  - Band reject filtering
  - Criss-cross filetering

Band reject filtering

- We create a filter consisting of ones with a ring of zeroes; the zeroes lying at a radius of 16(sqrt(288) from the center:

```
z=np.sqrt((x-112)**2+(y-112)**2)
k=1
d=np.sqrt(d[1])
br=((z<np.floor(d1-k))|(z>np.ceil(d1+k)))
```

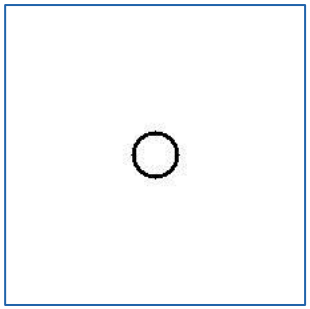
% z=15,16,17 will be zeros

- where z is the matrix consisting of distances from the origin. This particular ring will have a thickness large enough to cover the spikes. Then we multiply this by the transform:

cfr = cf * br

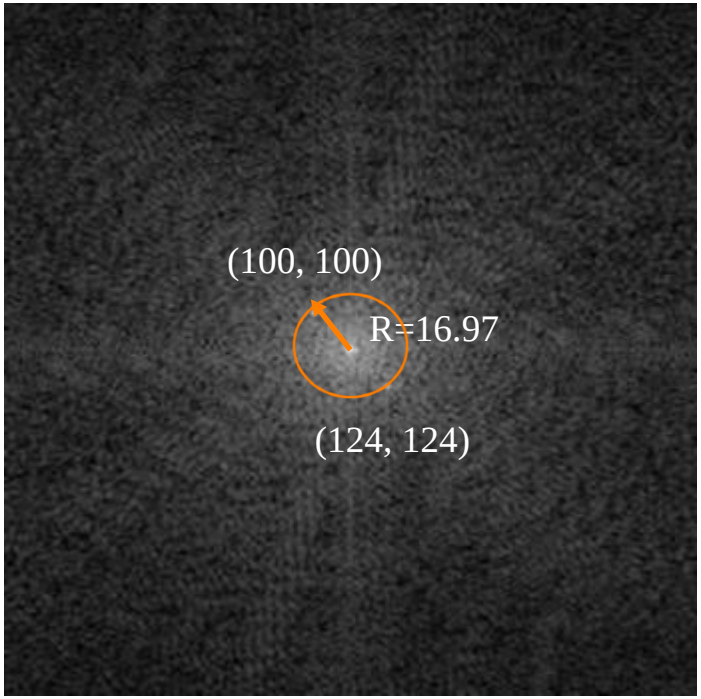
DFT of degraded image

ring



Band reject filter

Band reject filtering (cont.)



A band-reject filter

After inversion

Removing periodic noise with a band-reject filter

Criss-cross filetering

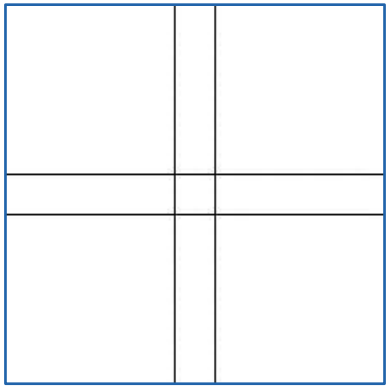
- With a criss-cross filter, we simply make the rows and columns of the spikes zero.

```
cf2 = np.copy(cf)
cf2[ l , : ] = 0
cf2[ : , j ] = 0
```

% entire row will be zero

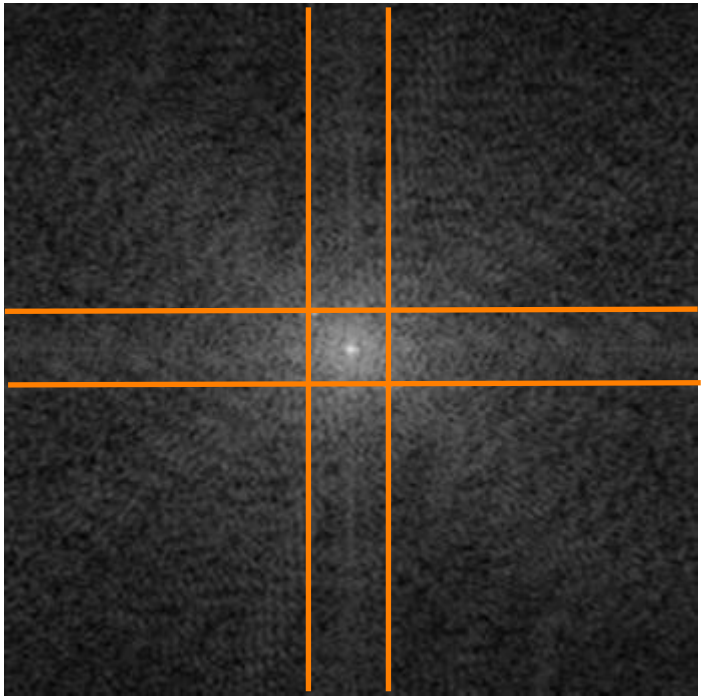
% entire column will be zero

- Much of the noise in the center has been removed. Making more rows and columns of the transform zero would result in a larger reduction of noise.



Criss-cross filter

Criss-cross filetering (cont.)



A criss-cross filter

After inversion

Removing periodic noise with a criss-cross filter

Next Week Lecture (Week6)

- Lecture6:Image Segmentation

Thank You