Labsheet 5
Lab-5 Image Enhancement - 1

Ragja Palakkadavath
(SC18M003)

Sub: Image Processing Lab **Department**: Mathematics
Date of Lab sheet: September 06, 2018 **Date of Submission**: September 12, 2018

1. Negative of an image

Use the mammogram image. Find the negative of an image without using any inbuilt function.

Aim

To understand how negative of an image is obtained without using any inbuilt function

Theory/Discussion

Negative of an image, is obtained such that the lightest areas of the photographed subject appear darkest and the darkest areas appear lightest.

Algorithm

1. Read the Image
2. Convert the image to array
3. To find the negative of the image, subtract its intensity value from 255
4. Save the obtained matrix of pixels as image

Code

```
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt

def get_image(image_name):
    image_path="C:/Users/Student/Downloads/ragja_iplab/".format(image_name);
    img = Image.open('C:/Users/Student/Downloads/ragja_iplab/mammogram.tif').convert('L')
    img.load()
    data = np.asarray( img, dtype="float64" )
    negative=255
    negative_array=negative-data
    plt.figure()
    plt.imshow(negative_array,cmap='gray')
get_image('mammogram.tif')
```

Result

Negative of an Image

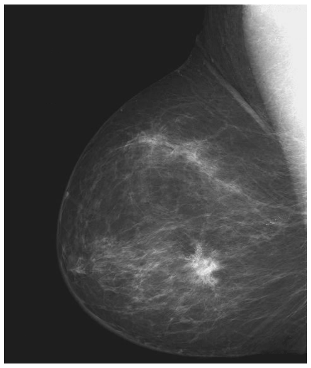

Figure 1: Original Mammogram Image

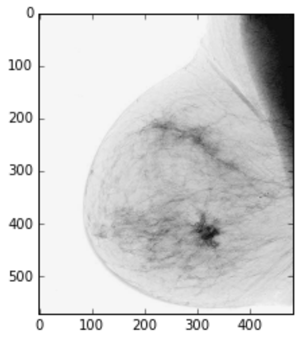

Figure 2: Negative of Mammogram Image

Inference

The negative of an image was obtained by subtracing the intensity of the pixels from 255(highest gray level in the image) Hence the darker parts of image was made lighter and vice versa.

2. Brightness Enhancement and Brightness Slicing

1. Use pollengrains image and increase the intensity value at each pixel location to obtain a brightness enhanced image.
2. Use kidney image and perform brightness enhancement only to pixels with intensity between 100 and 150 ; with and without background subtraction.

Aim

To understand how to increase brightness of an image as a whole, and also increasing brightness of specific intensities of specific pixels

Theory/Discussion

Brightness of an image is increased by increasing the intensity values of the pixels - where the max cap is the highest intensity gray level.

---

1. Read pollen grain Image
2. Convert the image to an array
3. For each pixel of the image, add an intensity value to make it brighter
4. If new intensity value of an image is greater than 255, set 255 as its intensity value
5. Read kidney Image
6. Convert the image to array. Modify on the pixels whose intensity values range from 100 to 150
7. For increasing brightness with background subtraction set all other intensity values as 0
8. For increasing brightness without background subtraction set all other intensity values remain unchanged

---

Code

Code for increasing brightness of pollen grain image

```
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt

def get_image():
    img = Image.open('C:/Users/Student/Downloads/ragja_iplab/1_pollen_grains.png').convert('
    L')
    img.load()
    data = np.asarray( img, dtype="uint8" )
    print(data)
    new_image=np.zeros((data.shape[0],data.shape[1]))
    for i in range(0,data.shape[0]):
        for j in range(0,data.shape[1]):
                final_intensity=data[i][j]+150
                if(final_intensity >255):
                    new_image[i][j]=255
                else:
                    new_image[i][j]=data[i][j]+150
    print(new_image)
    plt.figure()
    plt.imshow(new_image,cmap='gray')

get_image()
```

Code for increasing kidney image brightness with background subtraction

```
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt

def get_image():
    img = Image.open('C:/Users/Student/Downloads/ragja_iplab/3_kidney.png').convert('L')
    img.load()
    data = np.asarray( img, dtype="uint8" )
    print(data)
    new_image=np.zeros((data.shape[0],data.shape[1]))
    for i in range(0,data.shape[0]):
        for j in range(0,data.shape[1]):
            if data[i][j]>=100 and data[i][j]<=150:
                final_intensity=data[i][j]+150
                if(final_intensity>255):
                    new_image[i][j]=255
                else:
                    new_image[i][j]=data[i][j]+150
            else:
                new_image[i][j]=0
    print(new_image)
    plt.figure()
    plt.imshow(new_image,cmap='gray')
get_image()
```

Code for increasing kidney image brightness without background subtraction

```
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt

def get_image():
    img = Image.open('C:/Users/Student/Downloads/ragja_iplab/3_kidney.png').convert('L')
    img.load()
    data = np.asarray( img, dtype="uint8" )
    print(data)
    new_image=np.zeros((data.shape[0],data.shape[1]))
    for i in range(0,data.shape[0]):
        for j in range(0,data.shape[1]):
            if data[i][j]>=100 and data[i][j]<=150:
                final_intensity=data[i][j]+150
                if(final_intensity>255):
                    new_image[i][j]=255
                else:
                    new_image[i][j]=data[i][j]+150
            else:
                new_image[i][j]=data[i][j]
    print(new_image)
    plt.figure()
    plt.imshow(new_image,cmap='gray')
get_image()
```

Result

Increase brightness of the Image

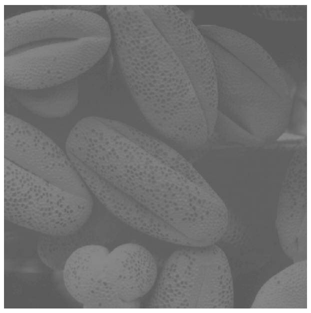

Figure 3: Original Pollen Grain Image

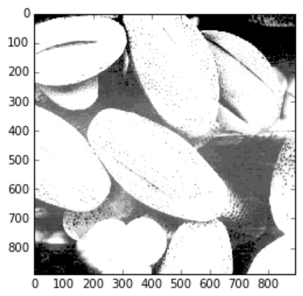

Figure 4: Brighter Pollen Grain Image

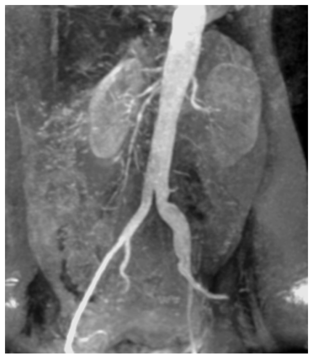

Figure 5: Original Kidney Image

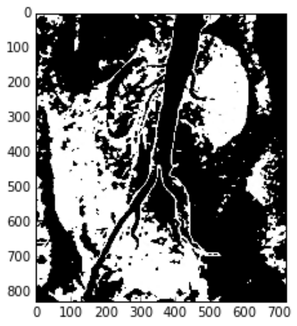

Figure 6: Kidney Image selective brightness with background subtraction

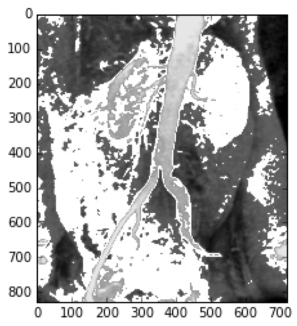

Figure 7: Kidney Image selective brightness without background subtraction

Inference

Brightness enhancement was done for the entire image. Selective brightness was also performed for the required intensity values. Both were performed with/without background subtraction

3. Histogram of an image

1. Plot the histogram of image.
2. Find the cumulative distribution also.
3. Is the histogram unimodal or multimodal?

Aim

To generate the histogram of an image and the CDF of the image

Theory/Discussion

An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value.

Algorithm

```
1. Read an Image
2. Plot the histogram of the image by the hist function in python by setting a value for the bin size
3. Compute the Cumulative Density Function by making cumulative attribute true in the hist function
4. Plot both the histograms for visualization
```

Code

Result

Histogram of Image

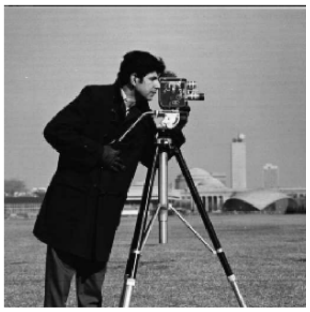

Figure 8: Original Cameraman Image

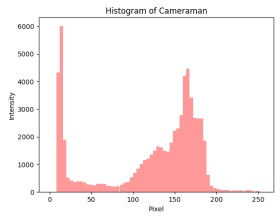

Figure 9: Histogram of Cameraman Image with bin size=64

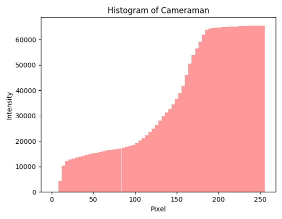

Figure 10: CDF of Cameraman Image

Inference

From the histogram of the image we can conclude that the histogram is multimodal as there are several peaks in the histogram Histogram and CDF of the cameraman image were plotted

4. Reducing high frequency noise

Use the cameraman image and perform the following

1. Add impulse noise and Gaussian noise to the image using inbuilt functions
2. Perform median filtering on both the images with filter size 3x3 , 5x5 and 11x11
3. Perform average filtering on both the images with filter size 3x3 , 5x5 and 11x11
4. Comment on the results obtained. Which filtering is best for which noise?

Aim

To learn how to add/denoise impulse and gaussian noises on an image

Theory/Discussion

Images taken with both digital cameras and conventional film cameras will pick up noise from a variety of sources. To use these images further, the noise should be (partially) removed â for aesthetic purposes as in artistic work or marketing, or for practical purposes such as computer vision. In salt and pepper noise, pixels in the image are very different in color or intensity from their surrounding pixels; the defining characteristic is that the value of a noisy pixel bears no relation to the color of surrounding pixels. In Gaussian noise, each pixel in the image will be changed from its original value by a (usually) small amount

Algorithm

1. Read the image Image
2. Apply gaussian and impulse noise to the cameraman image via random noise function in python
3. Average or mean filtering is performed by convoluting the image with a low pass filter
4. For median filtering, based on the kernel size, a part of the image is selected. The median is found among the selected intensity values. Then the central pixel is replaced with the median intensity value.

Code

```
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
import skimage.util as skp
import math
from scipy import signal
from scipy.misc import imsave
def get_image():
    img = Image.open('C:/Users/Student/Downloads/ragja_iplab/cameram
    an.tif').convert('L')
    img.load()
    data = np.asarray( img, dtype="uint8" )
    noise_creation_gaussian(data)

def noise_creation_gaussian(data):
    noise=skp.random_noise(data,mode='gaussian')
    imsave('gaussian_noise.png',noise)

def noise_creation_impulse(data):
    noise=skp.random_noise(data,mode='s&p')
    imsave('impulse_noise.png',noise)

def get_image2():
    img = Image.open('C:/Users/Student/Downloads/ragja_iplab/impulse_noise.png').convert('L
    ')
    img.load()
    data = np.asarray( img, dtype="uint8" )
    noise_removal_median_filter(data,3)
    #noise_removal_mean_filter(data,11)

def noise_removal_median_filter(data,size):
    radius=size-2
    filtered=np.zeros(shape=(data.shape[0],data.shape[1]))
    for y in range(0,data.shape[0]-1):
        top = max(y - radius, 0)
        bottom = min(y + radius,data.shape[0]-1)
        for x in range(0,data.shape[1]-1):
            left = max(x - radius, 0)
            right = min(x + radius, data.shape[1]-1)
            values = list()
            for v in range(top, bottom):
                for u in range(left,right):
                    values.append(data[v][u])
            filtered[y][x] = np.median(values)
    plt.figure()
    plt.imshow(filtered,cmap='gray')
    imsave('impulse_noise_removed_median_3-.png',filtered)

def noise_removal_mean_filter(data,size):
    h=np.ones(shape=(size,size))
    h=np.float_(h)
    h=h/size*size
    h=np.array(h)
    image=signal.convolve2d(data,h)
    plt.figure()
    plt.imshow(image,cmap='gray')
    imsave('gaussian_noise_removed_mean_11.png',image)

get_image2()
```

Result

Denoising Gaussian Noise

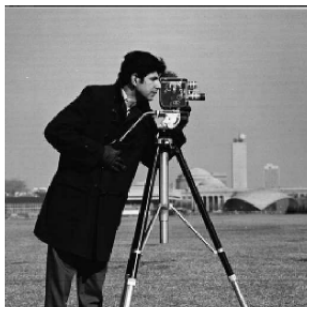

Figure 11: Original Cameraman Image

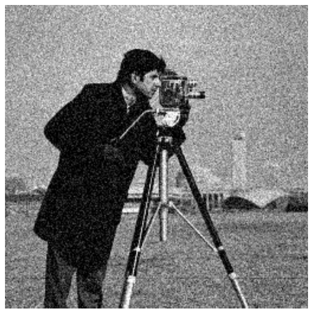

Figure 12: Cameraman Image with Gaussian Noise

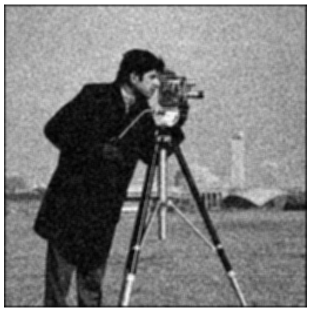

Figure 13: Gaussian Noise Removed by 3x3 Average Filter

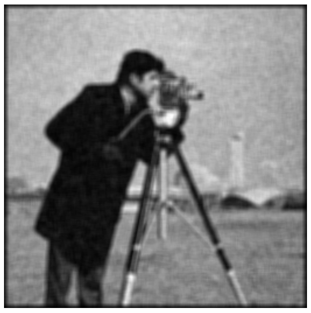

Figure 14: Gaussian Noise Removed by 5x5 Average Filter

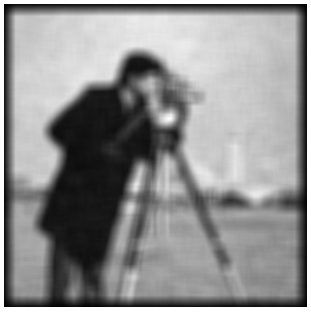

Figure 15: Gaussian Noise Removed by 11x11 Average Filter

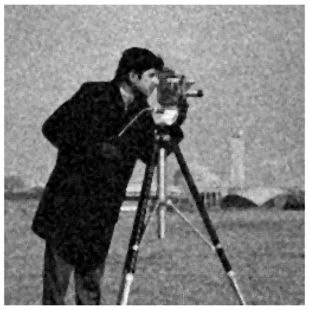

Figure 16: Gaussian Noise Removed by 3x3 Median Filter

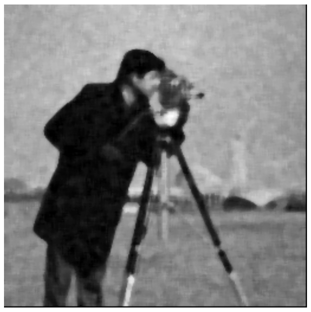

Figure 17: Gaussian Noise Removed by 5x5 Median Filter

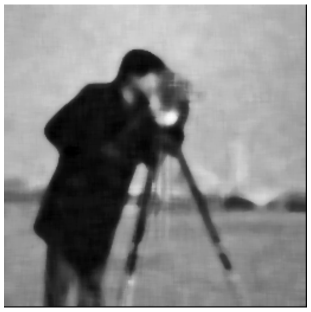

Figure 18: Gaussian Noise Removed by 11x11 Median Filter

Denoising Impulse Noise

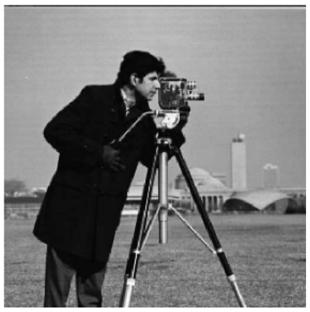

Figure 19: Original Cameraman Image

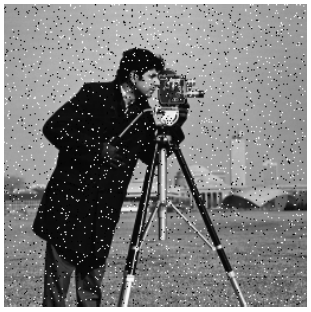

Figure 20: Cameraman Image with Impulse Noise

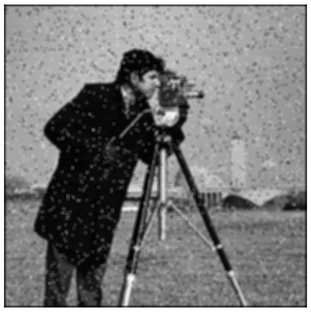

Figure 21: Impulse Noise Removed by 3x3 Average Filter

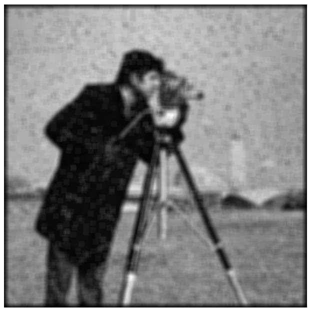

Figure 22: Impulse Noise Removed by 5x5 Average Filter

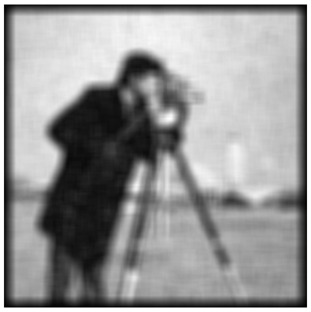

Figure 23: Impulse Noise Removed by 11x11 Average Filter

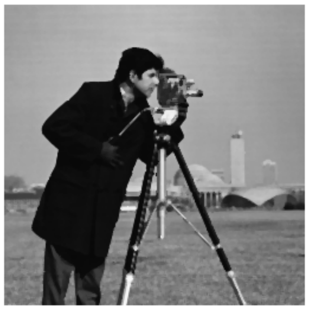

Figure 24: Impulse Noise Removed by 3x3 Median Filter

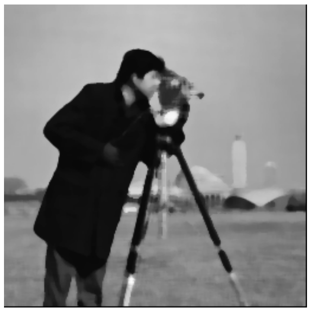

Figure 25: Impulse Noise Removed by 5x5 Median Filter

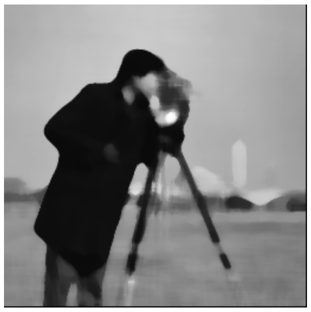

Figure 26: Impulse Noise Removed by 11x11 Median Filter

Inference

Noise was added to the image. Subsequently denoising was performed by average filter and median filter. It was found that average filter worked well for gaussian noise(low pass filtering to remove white noise) and median filter was better performing on impulse noise