Programming Assignment # 1

Course code: CSE- 6239

Course Tittle: Computer Vision

Submitted to:

Dr. Sk. Mohammad Masudul Ahsan

Professor

Department of Computer Science & Engineering

Khulna University of Engineering & Technology (KUET)

Submitted By:

Md. Safayet Hossain

Id:1907551

Department of CSE (KUET)

Submission date:18/09/2019

1. Write a function that convolves an image with a given convolution filter
function [output_Image]= myImageFilter(Input_image, filter) .
Your function should output image of the same size as that of input Image (use padding). Test your function (on attached images House1.jpg and House2.jpg) and show results on the following Kernels.

A. Average Kernel (3*3 and 5*5)

```
import cv2
from skimage import io, color
import matplotlib.pyplot as plt
import numpy as np

#input image from directory
img1=cv2.imread('House1.jpg', cv2.CV_8UC1)
img1 = color.rgb2gray(img1)

img2=cv2.imread('House2.jpg', cv2.CV_8UC1)
img2 = color.rgb2gray(img2)


def convolve2d(image,
kernel,kernel_height,kernel_width):
    output = np.zeros_like(image)
    for x in range(image.shape[1]-kernel_width+1):
        for y in range(image.shape[0]-kernel_height+1):

output[y,x]=(kernel*image[y:y+kernel_height,x:x+kernel_
width]).sum()/(kernel_width*kernel_height)
    return output

# Average kernel 3by3

kernel = np.array([[1,1,1],[1,1,1],[1,1,1]])
result1 = convolve2d(img1, kernel, 3, 3)
result2 = convolve2d(img2, kernel, 3, 3)

cv2.imwrite('Output
image/AveragekernelThreeByThreeHouse1.jpg',result1)
cv2.imwrite('Output
image/AveragekernelThreeByThreeHouse2.jpg',result2)
```

Output:

Figure 4 AveragekernelThreeByThreeHouse1.jpg

Figure 4 AveragekernelThreeByThreeHouse2.jpg

Figure 4 averagekernelfiveByfiveHouse1.jpg

Figure 4AveragekernelfiveByfiveHouse2.jpg

B. Gaussian Kernel (σ =1, 2, 3). Use (2σ +1)×(2σ +1) as size of Kernel (You may write a separate function to generate Gaussian Kernels for different values of σ.)

```
import cv2
from skimage import io, color
import matplotlib.pyplot as plt
import numpy as np

#input image
img1=cv2.imread('House1.jpg', cv2.CV_8UC1)
img1 = color.rgb2gray(img1)

img2=cv2.imread('House2.jpg', cv2.CV_8UC1)
img2 = color.rgb2gray(img2)


def convolve2d(image, kernel,kernel_height,kernel_width):
    output = np.zeros_like(image)
    for x in range(image.shape[1]-kernel_width+1):
        for y in range(image.shape[0]-kernel_height+1):

output[y,x]=(kernel*image[y:y+kernel_height,x:x+kernel_wi
dth]).sum()/(kernel_width*kernel_height)
    return output


def gussiankern(sigma):
    x, y = np.meshgrid(np.linspace(-1,1,2*sigma+1),
np.linspace(-1,1,2*sigma+1))
    d = np.sqrt(x*x+y*y)
    g = np.exp(-( d**2 / ( 2.0 * sigma**2 ) ) )
    return g

#gaussian kernel

sigma = 1
kernel = gussiankern(sigma)
result1 = convolve2d(img1,kernel,2*sigma+1,2*sigma+1)
result2 = convolve2d(img2,kernel,2*sigma+1,2*sigma+1)
```

```
cv2.imwrite('Output
image/gussiankernelHouse1.jpg',result1)
cv2.imwrite('Output
image/gussiankernelHouse2.jpg',result2)
```

Output:

Figure 6gussiankernelHouse1.jpg

Figure 6 GussiankernelHouse2.jpg

C. Sobel Edge Operators.

```
import cv2
from skimage import io, color
import matplotlib.pyplot as plt
import numpy as np

#input image
img1=cv2.imread('House1.jpg', cv2.CV_8UC1)
img1 = color.rgb2gray(img1)

img2=cv2.imread('House2.jpg', cv2.CV_8UC1)
img2 = color.rgb2gray(img2)

def convolve2d(image,
kernel,kernel_height,kernel_width):
    output = np.zeros_like(image)
    for x in range(image.shape[1]-kernel_width+1):
```

```
        for y in range(image.shape[0]-kernel_height+1):

output[y,x]=(kernel*image[y:y+kernel_height,x:x+kernel_
width]).sum()/(kernel_width*kernel_height)
    return output


#Sobel Edge Operator

# Sobel Edge  ---> x <----
kernel = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
result1 = convolve2d(img1,kernel,3,3)
result2 = convolve2d(img2,kernel,3,3)

cv2.imwrite('Output
image/sobel_edge_X_House1.jpg',result1)
cv2.imwrite('Output
image/sobel_edge_X_House2.jpg',result2)


# Sobel Edge  --->  y  <---
kernel = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
result1 = convolve2d(img1,kernel,3,3)
result2 = convolve2d(img2,kernel,3,3)

cv2.imwrite('Output
image/sobel_edge_Y_House1.jpg',result1)
cv2.imwrite('Output
image/sobel_edge_Y_House2.jpg',result2)
```

Output:

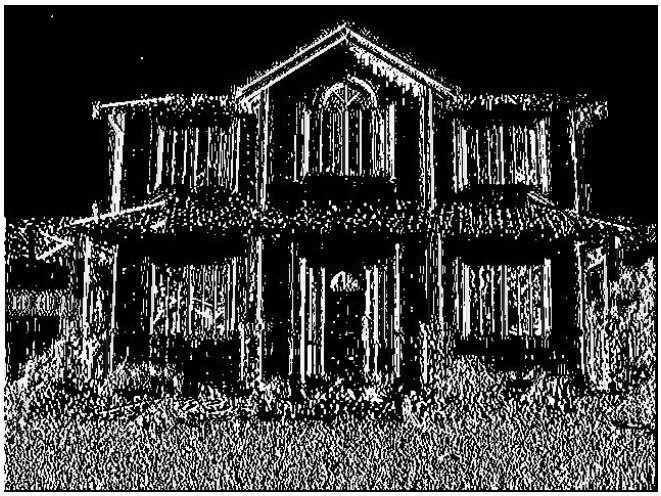

Figure 10 Sobel_edge_X_House1.jpg

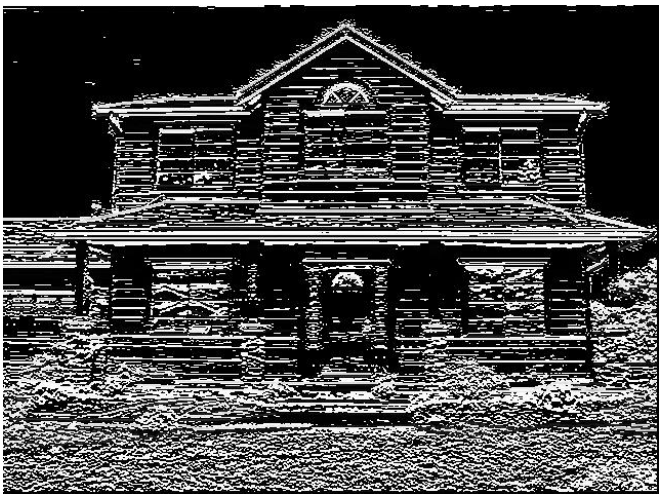

Figure 10 Sobel_edge_Y_House1.jpg

Figure 10 Sobel_edge_X_House2.jpg

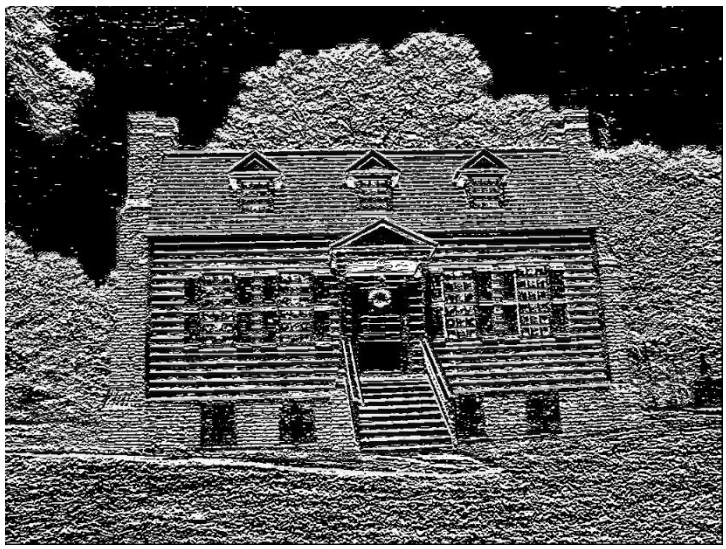

Figure 10 Sobel_edge_Y_House2.jpg

D. Prewitt Edge Operators.

```
import cv2
from skimage import io, color

import matplotlib.pyplot as plt
import numpy as np

#input image
img1=cv2.imread('House1.jpg', cv2.CV_8UC1)
img1 = color.rgb2gray(img1)

img2=cv2.imread('House2.jpg', cv2.CV_8UC1)
img2 = color.rgb2gray(img2)


def convolve2d(image,
kernel,kernel_height,kernel_width):
    output = np.zeros_like(image)
    for x in range(image.shape[1]-kernel_width+1):
        for y in range(image.shape[0]-kernel_height+1):

output[y,x]=(kernel*image[y:y+kernel_height,x:x+kernel_
width]).sum()/(kernel_width*kernel_height)
    return output


# Prewiit edge operator

# Prewiit Edge Operator ----> x <----
kernel = np.array([[+1,0,-1],[+1,0,-1],[1,0,-1]])
result1 = convolve2d(img1,kernel,3,3)
result2 = convolve2d(img2,kernel,3,3)

cv2.imwrite('Output
image/Prewiit_edge_X_house1.jpg',result1)
cv2.imwrite('Output
image/Prewiit_edge_X_house2.jpg',result2)
```

```
# Prewiit Edge Operator --->  y  <----
kernel = np.array([[+1,+1,+1],[0,0,0],[-1,-1,11]])
result1 = convolve2d(img1,kernel,3,3)
result2 = convolve2d(img2,kernel,3,3)

cv2.imwrite('Output
image/Prewiit_edge_Y_house1.jpg',result1)
cv2.imwrite('Output
image/Prewiit_edge_Y_house2.jpg',result2)
```

Output:

Figure 14 Prewiit_edge_X_house1.jpg

Figure 14 Prewiit_edge_Y_house1.jpg

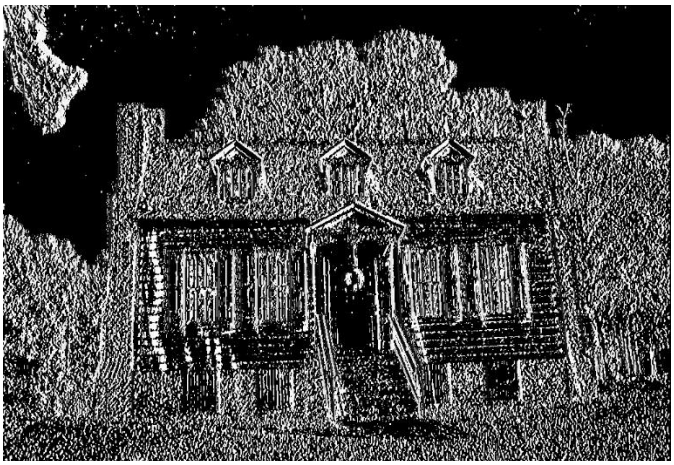

Figure 14 Prewiit_edge_X_house2.jpg

Figure 14 Prewiit_edge_Y_house2.jpg

2 Attached ‘Noisy image1’ and ‘Noisy image2’ are corrupted by salt and paper noise. Apply 5 by 5 Averaging and Median filter and show your outputs.

```
import cv2
from skimage import io, color
import numpy as np

#input image
img1=cv2.imread('Noisyimage1.jpg', cv2.CV_8UC1)
img1 = color.rgb2gray(img1)
```

```
img2=cv2.imread('Noisyimage2.jpg', cv2.CV_8UC1)
img2 = color.rgb2gray(img2)



def convolve2d(image,
kernel,kernel_height,kernel_width):
    output = np.zeros_like(image)
    for x in range(image.shape[1]-kernel_width+1):
        for y in range(image.shape[0]-
kernel_height+1):

output[y,x]=(kernel*image[y:y+kernel_height,x:x+kerne
l_width]).sum()/(kernel_width*kernel_height)
    return output


def median_filter(data, filter_size):
    temp = []
    indexer = filter_size // 2
    data_final = []
    data_final = np.zeros((len(data), len(data[0])))
    for i in range(len(data)):

        for j in range(len(data[0])):

            for z in range(filter_size):
                if i + z - indexer < 0 or i + z -
indexer > len(data) - 1:
                    for c in range(filter_size):
                        temp.append(0)
                else:
                    if j + z - indexer < 0 or j +
indexer > len(data[0]) - 1:
                        temp.append(0)
                    else:
                        for k in range(filter_size):
                            temp.append(data[i + z -
indexer][j + k - indexer])

            temp.sort()
```

```
            data_final[i][j] = temp[len(temp) // 2]
            temp = []
    return data_final




#average jernel 5 by 5

kernel =
np.array([[1,1,0,0,1],[1,1,1,1,0],[0,0,0,0,0],[0,1,1,
1,1],[1,0,1,1,0]])
result1 = convolve2d(img1,kernel,5,5)
result2 = convolve2d(img2,kernel,5,5)

cv2.imwrite('Output
image/noiseRemove1Byaveragefilter.jpg',result1)
cv2.imwrite('Output
image/noiseRemove2Byaveragefilter.jpg',result2)

result1 = median_filter(img1,3)
result2 = median_filter(img2,3)

cv2.imwrite('Output
image/noiseremoveByMedian1.jpg',result1)
cv2.imwrite('Output
image/noiseremoveByMedian2.jpg',result2)
```

Output:

Figure 18 NoiseRemove1Byaveragefilter.jpg

Figure 18 noiseremoveByMedian1.jpg

Figure 18 NoiseRemove2Byaveragefilter.jpg

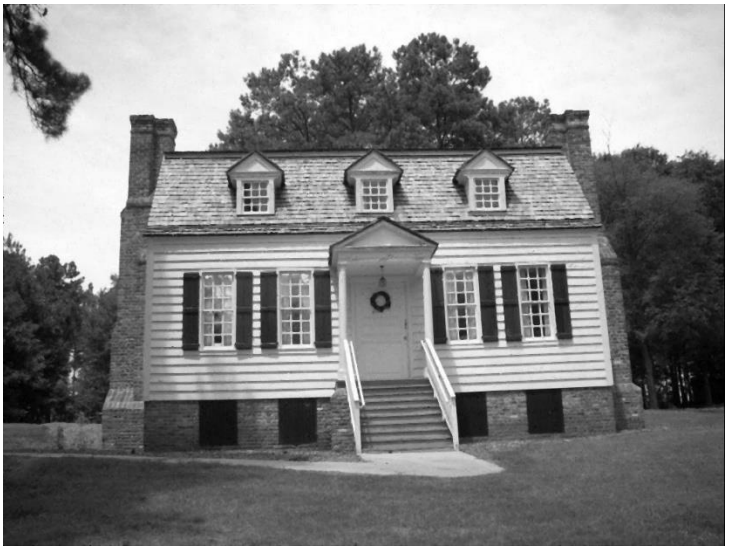

Figure 18 noiseremoveByMedian2.jpg

4 Load *walk_1.jpg* and *walk_2.jpg* images in openCV. Convert them to gray scale and subtract *walk_2.jpg* from *walk_1.jpg*. What is the result? Why?

```
import cv2

image_One = cv2.imread('walk_1.jpg')
image_two = cv2.imread('walk_2.jpg')

gray_image_one =
cv2.cvtColor(image_One,cv2.COLOR_RGB2GRAY)
gray_image_two =
cv2.cvtColor(image_two,cv2.COLOR_RGB2GRAY)

subtract_image = gray_image_one.copy()

for i in range(subtract_image.shape[0]):
    for j in range(subtract_image.shape[1]):

subtract_image.itemset((i,j),abs(int(gray_image_one[i][j]
) - int(gray_image_two[i][j])))


cv2.imwrite('output
image/subtract_image.jpg',subtract_image)
```

Output:

Figure 19 subtract_image.jpg

This result has been come because two images background are same but their objects are difference so one background subtract by another that's why our result every time come like this.