**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Subject Name**

**Digital Image Processing**

**Lab Number**

**5**

**SUBMITTED TO:**

**Dr. Asad Khan**

**LE Sundas Ashraf**

**SUBMITTED BY:**

**Student Name**

1. Wahaaj Nasir

**Reg#413238**

**DE- 44 Dept C&SE**

**Objectives:**

Filtering on Images and Histogram Calculations

**Related Topic/Chapter in theory class:**

Spatial Filtering

**Hardware/Software required:**

Hardware: PC

Software Tool: Pycharm

**Task 1:**

**Lab Task 1:**

**Apply the following filters with size 3, 15 and 31 on images Fig01.tif and Fig02.tif:**

**• Median**

**• Min**

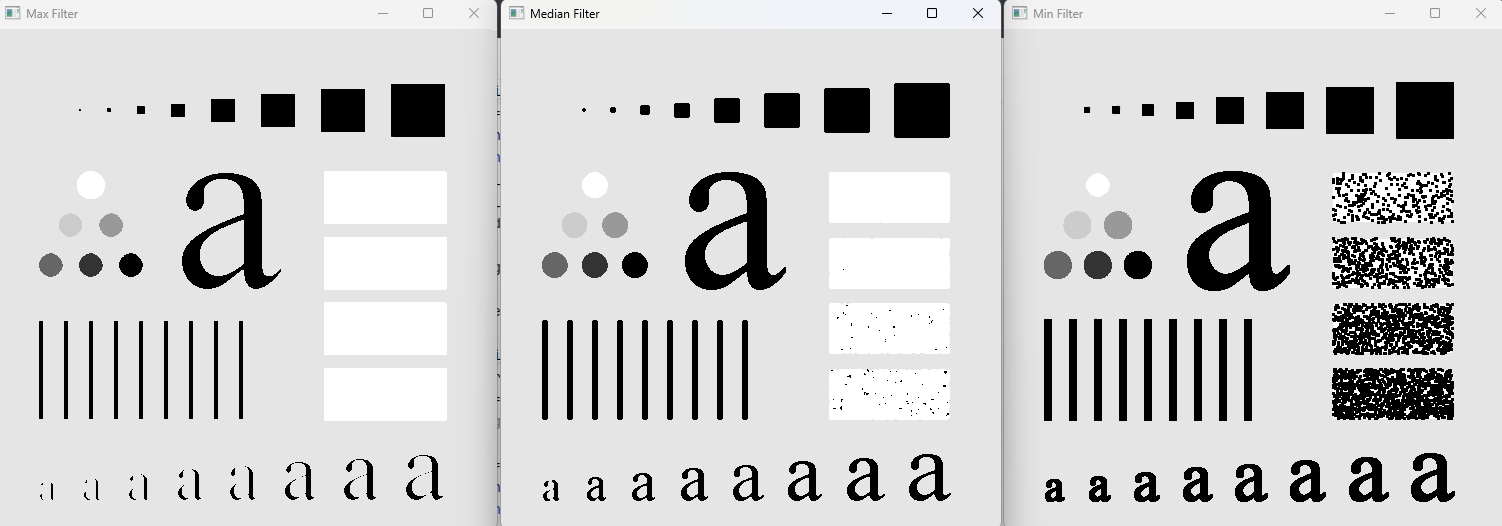
**• Max**

**Solution:**

import numpy as np  
import cv2 as cv  
  
def padding(pad, orig):  
 rows, cols = orig.shape  
 padded\_arr = np.ones((rows+ 2 \* pad, cols+ 2 \* pad), dtype = np.uint8)\*255  
  
 for i in range(rows):  
 for j in range(cols):  
 padded\_arr[i+pad][j+pad] = orig[i][j]  
  
 return padded\_arr  
  
def remove\_padding(padded\_img, pad):  
 rows, cols = padded\_img.shape  
 return padded\_img[pad:rows-pad, pad:cols-pad]  
  
def filter\_min(image, filter\_size):  
 pad = filter\_size//2  
 rows, cols = image.shape  
 filtered\_img = np.zeros((rows, cols), dtype = np.uint8)  
  
 padded\_img = padding(pad, filtered\_img)  
 for i in range(pad, rows-pad):  
 for j in range(pad, cols-pad):  
 sub\_img = image[i-pad:i+pad+1, j-pad:j+pad+1]  
 sub\_img\_val = np.min(sub\_img)  
 padded\_img[i][j] = sub\_img\_val  
  
 filtered\_img = remove\_padding(padded\_img, pad)  
  
 return filtered\_img  
  
def filter\_max(image, filter\_size):  
 pad = filter\_size//2  
 rows, cols = image.shape  
 filtered\_img = np.zeros((rows, cols), dtype = np.uint8)  
  
 padded\_img = padding(pad, filtered\_img)  
 for i in range(pad, rows-pad):  
 for j in range(pad, cols-pad):  
 sub\_img = image[i-pad:i+pad+1, j-pad:j+pad+1]  
 sub\_img\_val = np.max(sub\_img)  
 padded\_img[i][j] = sub\_img\_val  
  
 filtered\_img = remove\_padding(padded\_img, pad)  
  
 return filtered\_img  
  
def filter\_median(image, filter\_size):  
 pad = filter\_size//2  
 rows, cols = image.shape  
 filtered\_img = np.zeros((rows, cols), dtype = np.uint8)  
  
 padded\_img = padding(pad, filtered\_img)  
 for i in range(pad, rows-pad):  
 for j in range(pad, cols-pad):  
 sub\_img = image[i-pad:i+pad+1, j-pad:j+pad+1]  
 sub\_img\_val = np.median(sub\_img)  
 padded\_img[i][j] = sub\_img\_val  
  
 filtered\_img = remove\_padding(padded\_img, pad)  
  
 return filtered\_img  
  
*#-----------------  
# Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 5/Lab 5/Fig01.tif", 0)  
  
max\_img\_3 = filter\_max(image, 3)  
min\_img\_3 = filter\_min(image, 3)  
median\_img\_3 = filter\_median(image, 3)  
  
max\_img\_15 = filter\_max(image, 15)  
min\_img\_15 = filter\_min(image, 15)  
median\_img\_15 = filter\_median(image, 15)  
  
max\_img\_31 = filter\_max(image, 31)  
min\_img\_31 = filter\_min(image, 31)  
median\_img\_31 = filter\_median(image, 31)  
  
  
cv.imshow('Original', image)  
cv.imshow('Max Filter', max\_img\_3)  
cv.imshow('Min Filter', min\_img\_3)  
cv.imshow('Median Filter', median\_img\_3)  
cv.waitKey()  
  
cv.imshow('Original', image)  
cv.imshow('Max Filter', max\_img\_15)  
cv.imshow('Min Filter', min\_img\_15)  
cv.imshow('Median Filter', median\_img\_15)  
cv.waitKey()  
  
cv.imshow('Original', image)  
cv.imshow('Max Filter', max\_img\_31)  
cv.imshow('Min Filter', min\_img\_31)  
cv.imshow('Median Filter', median\_img\_31)  
cv.waitKey()

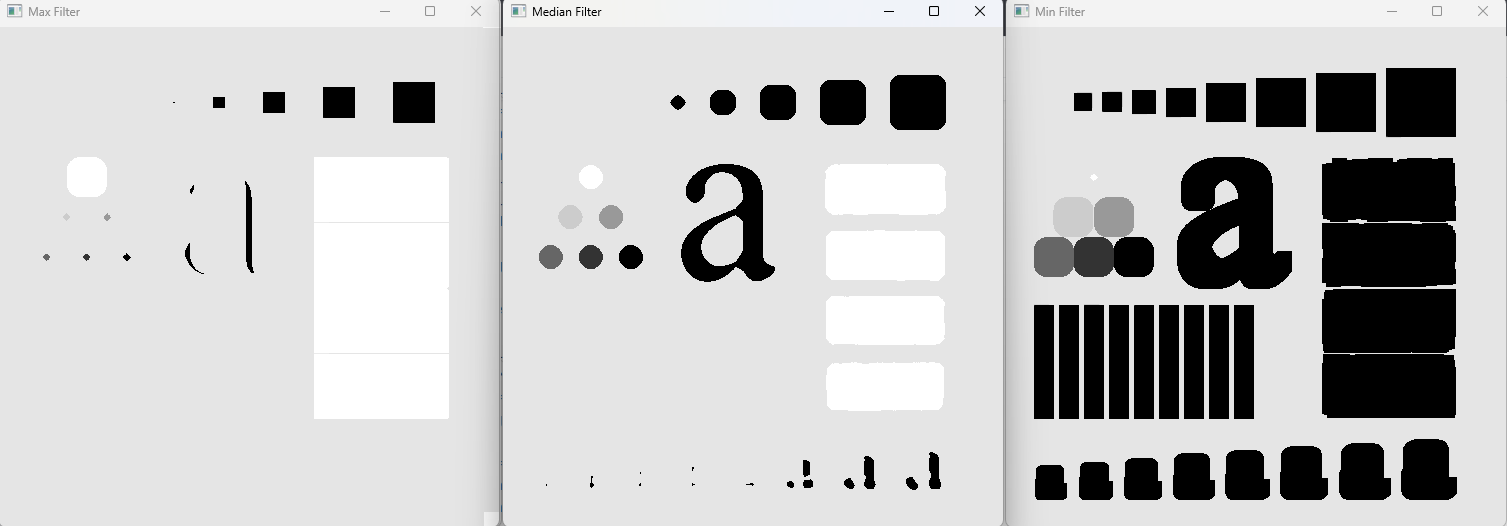
**Output:**

**3x3**



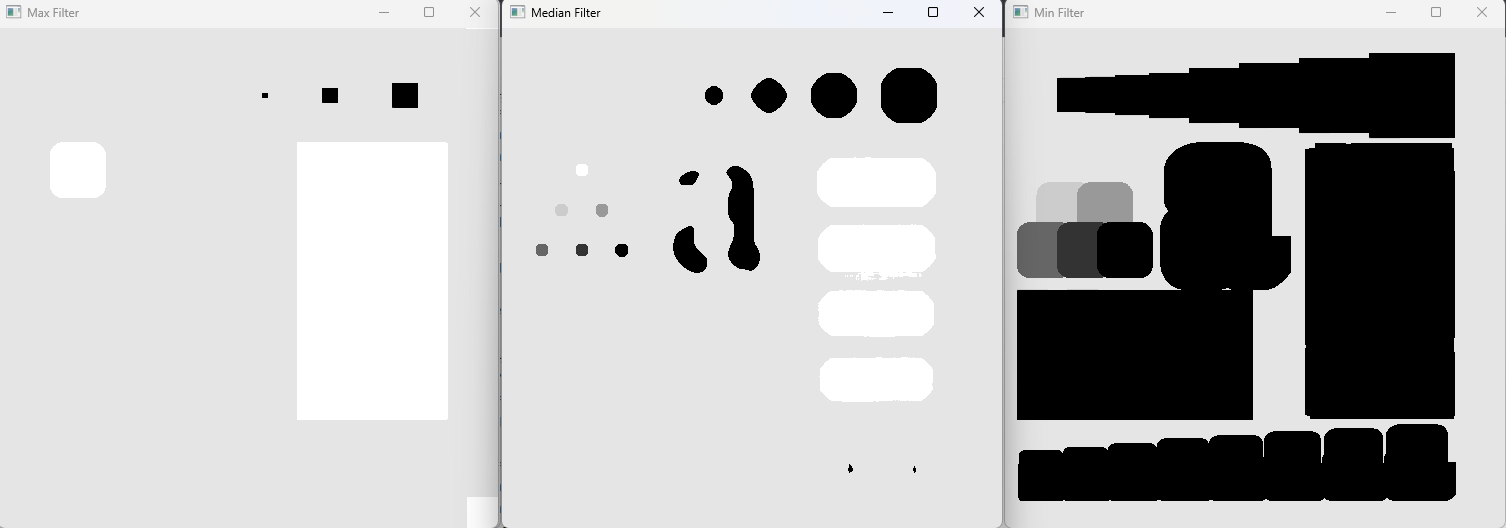
**Max Median Min**

**15x15**

****

**Max Median Min**

**31x31**

****

**Max Median Min**

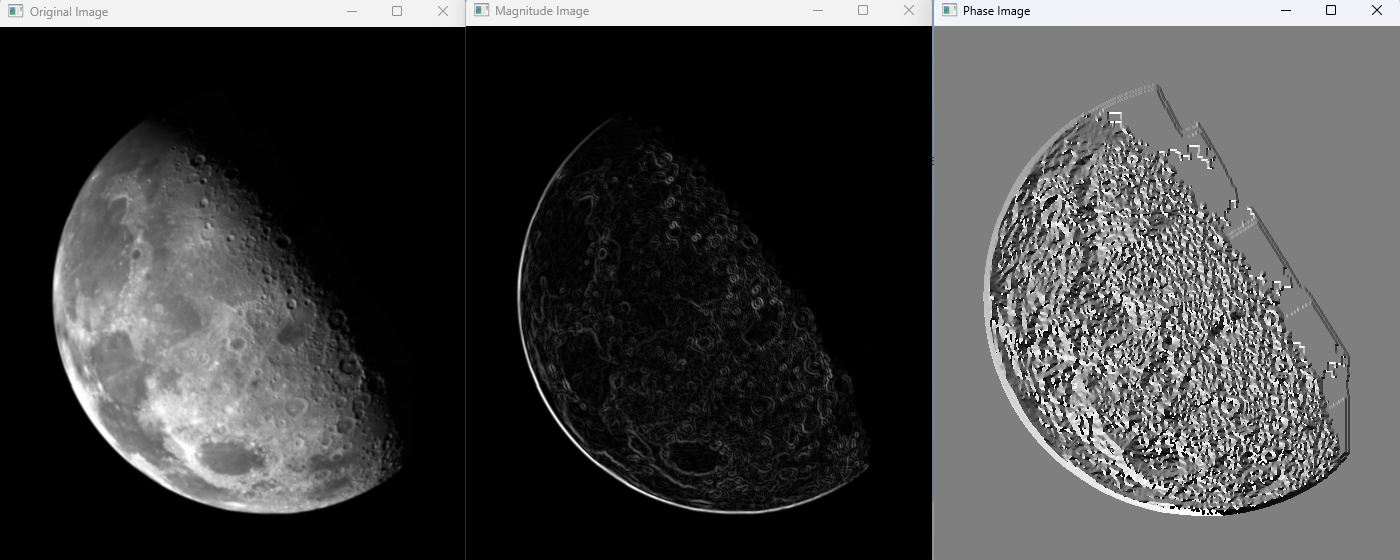
**Task 2:**

**Apply Horizontal Sobel and Vertical Sobel separately on the Fig03.tif. Display the results. Then add the images obtained by Horizontal Sobel and Vertical Sobel together and display the resultant images with all the edges.**

**Solution**

import numpy as np  
import cv2 as cv  
  
def padding(pad, orig):  
 rows, cols = orig.shape  
 padded\_arr = np.ones((rows+ 2 \* pad, cols+ 2 \* pad), dtype = np.uint8)\*255  
  
 for i in range(rows):  
 for j in range(cols):  
 padded\_arr[i+pad][j+pad] = orig[i][j]  
  
 return padded\_arr  
  
def remove\_padding(padded\_img, pad):  
 rows, cols = padded\_img.shape  
 return padded\_img[pad:rows-pad, pad:cols-pad]  
  
def filter\_sobel(image, filter\_size):  
 pad = filter\_size//2  
 rows, cols = image.shape  
 mag\_img = np.zeros((rows, cols), dtype = np.float32)  
 phase\_img = np.zeros((rows, cols), dtype=np.float32)  
  
 filter\_x = np.zeros((3, 3), dtype=int)  
 filter\_y = np.zeros((3, 3), dtype=int)  
  
 *# Filter\_X* filter\_x[0][0] = -1  
 filter\_x[0][1] = -2  
 filter\_x[0][2] = -1  
 filter\_x[1][0:3] = 0  
 filter\_x[2][0] = 1  
 filter\_x[2][1] = 2  
 filter\_x[2][2] = 1  
  
 *# Filter\_Y* filter\_y[0][0] = -1  
 filter\_y[0:3][1] = 0  
 filter\_y[0][2] = 1  
 filter\_y[1][0] = -2  
 filter\_y[1][2] = 2  
 filter\_y[2][0] = -1  
 filter\_y[2][2] = 1  
  
 for i in range(pad, rows-pad):  
 for j in range(pad, cols-pad):  
 sub\_img = image[i-pad:i+pad+1, j-pad:j+pad+1]  
 sobel\_x = np.sum(np.multiply(sub\_img, filter\_x))  
 sobel\_y = np.sum(np.multiply(sub\_img, filter\_y))  
 mag = np.sqrt(sobel\_x\*\*2 + sobel\_y\*\*2)  
 phase = np.arctan2(sobel\_y, sobel\_x)  
 mag\_img[i][j] = mag  
 phase\_img[i][j] = phase  
  
 mag\_max = np.max(mag\_img)  
 mag\_min = np.min(mag\_img)  
 mag\_img = ((mag\_img - mag\_min)/(mag\_max - mag\_min)) \*255  
 phase\_img = ((phase\_img - np.min(phase\_img))/(np.max(phase\_img) - np.min(phase\_img))) \*255  
  
 mag\_img = mag\_img.astype(np.uint8)  
 phase\_img = phase\_img.astype(np.uint8)  
  
 return mag\_img, phase\_img  
  
  
*#---------------------  
# Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 5/Lab 5/Fig03.tif", 0)  
  
cv.imshow('Original Image', image)  
cv.waitKey()  
  
mag\_img, phase\_img = filter\_sobel(image, 3)  
  
cv.imshow('Magnitude Image', mag\_img)  
cv.imshow('Phase Image', phase\_img)  
cv.waitKey()

**Output:**



**Task 3:**

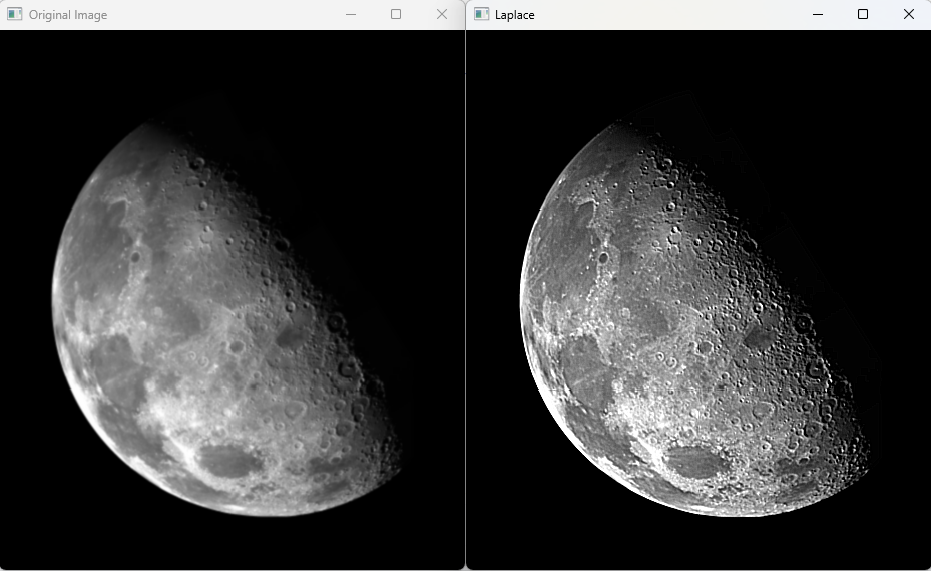
**Apply the following Laplacian mask to image Fig03.tif. Then add the filtered image to the**

**original image for a sharpening effect.**

**Solution**

import numpy as np  
import cv2 as cv  
  
def padding(pad, orig):  
 rows, cols = orig.shape  
 padded\_arr = np.ones((rows+ 2 \* pad, cols+ 2 \* pad), dtype = np.uint8)\*255  
  
 for i in range(rows):  
 for j in range(cols):  
 padded\_arr[i+pad][j+pad] = orig[i][j]  
  
 return padded\_arr  
  
def remove\_padding(padded\_img, pad):  
 rows, cols = padded\_img.shape  
 return padded\_img[pad:rows-pad, pad:cols-pad]  
  
def filter\_laplace(image, filter\_size):  
 pad = filter\_size//2  
 rows, cols = image.shape  
 laplace\_filter = np.array([[-1, -1, -1], [-1, 8, -1], [-1, -1 , -1]], dtype = int)  
 laplace\_img = np.zeros((rows, cols), dtype = np.float32)  
  
 padded\_img = padding(pad, image)  
 for i in range(pad, rows-pad):  
 for j in range(pad, cols-pad):  
 sub\_img = padded\_img[i-pad:i+pad+1, j-pad:j+pad+1]  
 value = np.sum(np.multiply(sub\_img, laplace\_filter))  
 laplace\_img[i][j] = value  
  
 laplace\_img = np.clip(image.astype(np.int16) + laplace\_img, 0, 255)  
 laplace\_img = laplace\_img.astype(np.uint8)  
  
 return laplace\_img  
  
  
*#---------------------  
# Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 5/Lab 5/Fig03.tif", 0)  
  
cv.imshow('Original Image', image)  
cv.waitKey()  
  
laplace = filter\_laplace(image, 3)  
cv.imshow('Laplace', laplace)  
cv.waitKey()

**Output:**

****