BAHRIA UNIVERSITY, ISLAMABAD
Department of Computer Science

CEN 444
Digital Image Processing
Lab Journal 8

Student Name: _M MUNEEB AHMED KIANI

Enrolment No.: _01-135212-063

Title: Spatial Filtering

Objectives: to introduce to the process of Filtering. To understand functions for Smoothing and Sharpening filters.

Tools Used: Python

Procedure: Open IDLE and perform the following tasks

```
import cv2
import numpy as np

img = cv2.imread("Your_image.jpg")
img = cv2.resize(img, (0, 0), None, .25, .25)

gaussianBlurKernel = np.array(([[1, 2, 1], [2, 4, 2], [1, 2, 1]]), np.float
32)/9
sharpenKernel = np.array(([[0, -1, 0], [-1, 9, -1], [0, -
1, 0]]), np.float32)/9
meanBlurKernel = np.ones((3, 3), np.float32)/9

gaussianBlur = cv2.filter2D(src=img, kernel=gaussianBlurKernel, ddepth=-1)
meanBlur = cv2.filter2D(src=img, kernel=meanBlurKernel, ddepth=-1)
sharpen = cv2.filter2D(src=img, kernel=sharpenKernel, ddepth=-1)

horizontalStack = np.concatenate((img, gaussianBlur, meanBlur, sharpen), ax
is=1)
```

```
cv2.imwrite("Output.jpg", horizontalStack)

cv2.imshow("2D Convolution Example", horizontalStack)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

The Filter2D operation convolves an image with the kernel. You can perform this operation on an image using the Filter2D() method of the imgproc class. Following is the documentation of this method –

filter2D(src, dst, ddepth, kernel)

This method accepts the following parameters –

- **src** – A Mat object representing the source (input image) for this operation.
- **dst** – A Mat object representing the destination (output image) for this operation.
- **ddepth** – A variable of the type integer representing the depth of the output image.
- **kernel –** A Mat object representing the convolution kernel.

Average Filtering with 5x5 kernel

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('/content/sample_data/images/cameraman.png')
kernel = np.ones((5,5),np.float32)/25
dst = cv2.filter2D(img,-1,kernel)
plt.figure(figsize=(10,10))
plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.subplot(122),plt.imshow(dst),plt.title('Averaging')
plt.show()
```

Using blur Function

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('/content/sample_data/images/cameraman.png')

blur = cv2.blur(img,(5,5))
plt.figure(figsize=(10,10))
plt.subplot(121),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur),plt.title('Blurred')
plt.xticks([]), plt.yticks([])
plt.show()
```

Some other functions:

cv2.GaussianBlur().

cv2.getGaussianKernel().

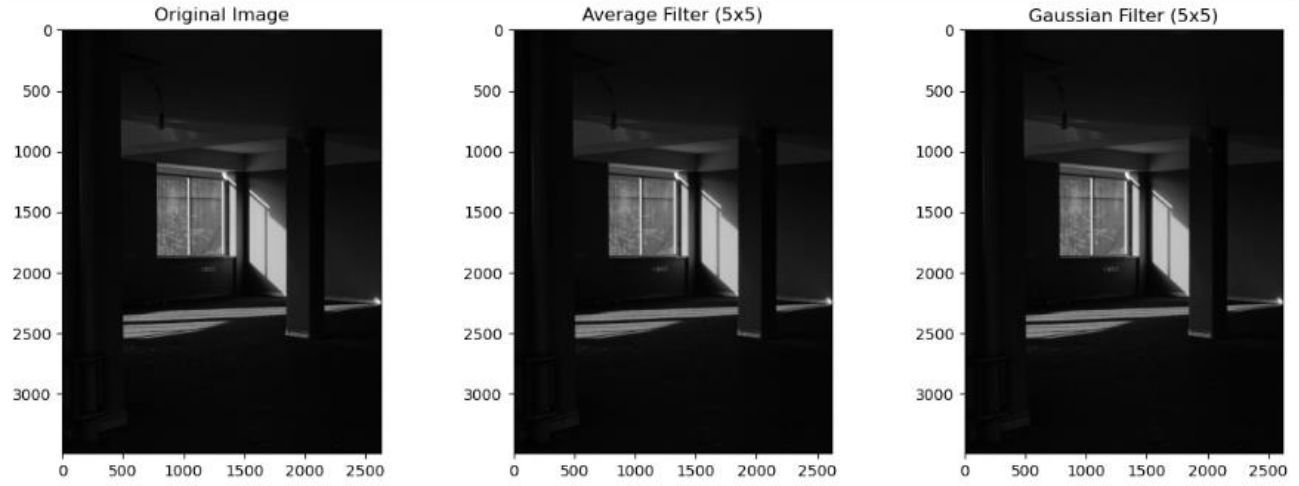
cv2.medianBlur()

```
median = cv2.medianBlur(img,5)
```

Task 1

Read an image of your choice. Apply average and gaussian filters of size 5x5 individually and identify the differences b/w their results.

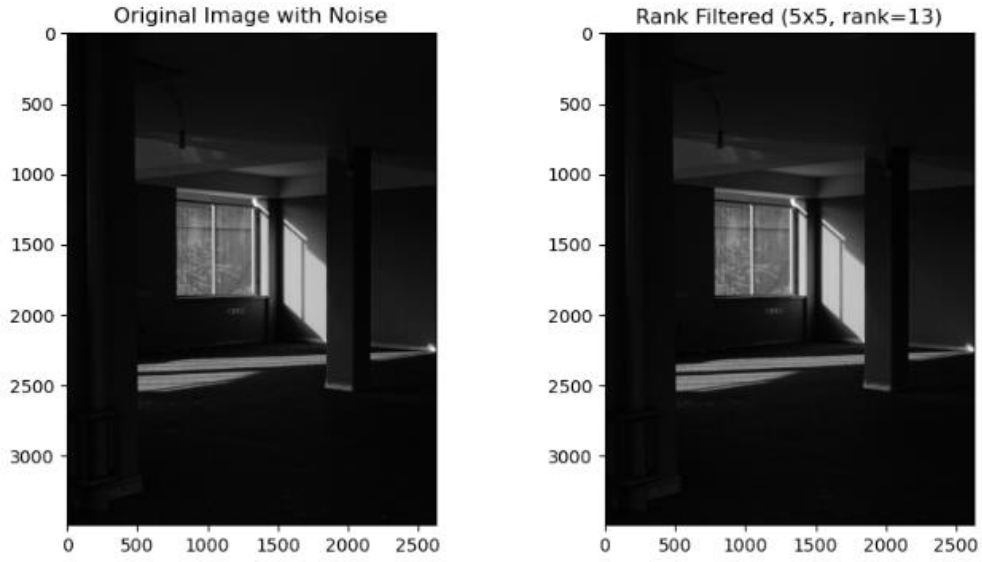
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
image = cv2.imread('pavel-moiseev-hFmxJMnvECc-unsplash.jpg', cv2.IMREAD_GRAYSCALE)
average_filtered = cv2.blur(image, (5, 5))
gaussian_filtered = cv2.GaussianBlur(image, (5, 5), 0)
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')
plt.subplot(1, 3, 2)
plt.title('Average Filter (5x5)')
plt.imshow(average_filtered, cmap='gray')
plt.subplot(1, 3, 3)
plt.title('Gaussian Filter (5x5)')
plt.imshow(gaussian_filtered, cmap='gray')
plt.show()
```



Task 2

Read an image of your choice which has salt and pepper noise. Apply rank filter of size 5x5 using rank = 13. What is the other name of this filtering? If you use rank = 1 or 25, will the noise increase or decrease?

```
from scipy.ndimage import rank_filter
import cv2
image = cv2.imread('pavel-moiseev-hFmxJMnvECc-unsplash.jpg', cv2.IMREAD_GRAYSCALE)
rank_filtered = rank_filter(image, rank=13, size=(5, 5))
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image with Noise')
plt.imshow(image, cmap='gray')
plt.subplot(1, 2, 2)
plt.title('Rank Filtered (5x5, rank=13)')
plt.imshow(rank_filtered, cmap='gray')
plt.show()
```



Task 3

Read the image. Write a function named 'mylaplacian' to MANUALLY code/implement 2nd order derivate of above read image in order to extract horizontal and vertical edges, collectively. Also, compare your results with 'Sobel' filter and state your findings.

[Code Hint]: You need to perform filtering with the following masks.

Vertical Edges:

```
g(x, y) = f(x + 1, y) + f(x – 1, y) – 2f(x, y)
```

Horizontal Edges:

```
g(x, y) = f(x , y + 1) + f(x, y - 1) – 2f(x, y)
```

```python
import numpy as np
import cv2
from matplotlib import pyplot as plt
def mylaplacian(image):
    vertical_edges = np.zeros_like(image, dtype=np.float32)
    horizontal_edges = np.zeros_like(image, dtype=np.float32)
    for x in range(1, image.shape[0] - 1):
        for y in range(1, image.shape[1] - 1):
            vertical_edges[x, y] = (
                image[x + 1, y] + image[x - 1, y] - 2 * image[x, y]
            )
            horizontal_edges[x, y] = (
                image[x, y + 1] + image[x, y - 1] - 2 * image[x, y]
            )
    laplacian_result = np.sqrt(vertical_edges**2 + horizontal_edges**2)
    return np.clip(laplacian_result, 0, 255).astype(np.uint8)
image = cv2.imread('Picture1.jpg', cv2.IMREAD_GRAYSCALE)
laplacian_result = mylaplacian(image)
sobel_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)
sobel_result = np.sqrt(sobel_x**2 + sobel_y**2)
sobel_result = np.clip(sobel_result, 0, 255).astype(np.uint8)
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.title('Original Image')
plt.imshow(image, cmap='gray')
plt.subplot(1, 3, 2)
plt.title('Manual Laplacian Filter')
plt.imshow(laplacian_result, cmap='gray')
plt.subplot(1, 3, 3)
plt.title('Sobel Filter')
plt.imshow(sobel_result, cmap='gray')
plt.show()
```