Digital Image Processing

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)
```

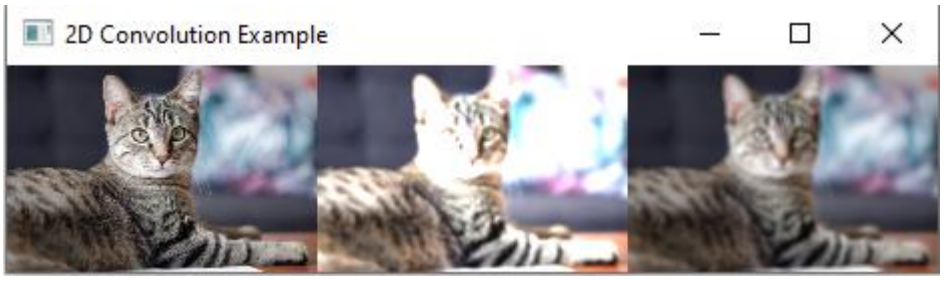
Task 1

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

Code:

```
import cv2
import numpy as np

path = r'C:\Users\Naeem\Desktop\Jahanzeb\DIP\DIP Lab\images\333.jpg'
img = cv2.imread(path, cv2.IMREAD_COLOR)
img = cv2.resize(img, (0, 0), None, .25, .25)

gaussianBlurKernel = np.array(([[1, 2, 1], [2, 4, 2], [1, 2, 1]]), 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)

horizontalStack = np.concatenate((img, gaussianBlur, meanBlur), axis=1)

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

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

Output:



We can see that the average filter changes the gray values (in this case, the image brightness increased too much). Whereas, the Gaussian blur smoothened the image without changing the brightness.

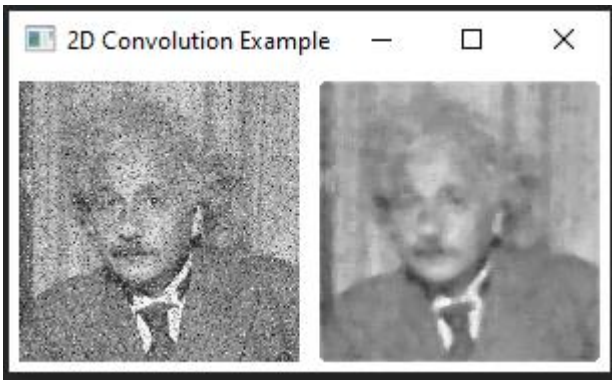
Task 2

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?

Code:

```
import cv2
import numpy as np

path = r'C:\Users\Naeem\Desktop\Jahanzeb\DIP\DIP Lab\images\SP.jpg'
img = cv2.imread(path, cv2.IMREAD_COLOR)
img = cv2.resize(img, (0, 0), None, .25, .25)

medianBlur = cv2.medianBlur(img, 5)

horizontalStack = np.concatenate((img, medianBlur), axis=1)

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

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

Output:



If rank 1 or 25 was used, it would use the white (in rank 25 case) or black noise (in rank 1 case) pixels to put in the center pixel place. Hence, the filter would fail to remove any noise.

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)
```

Code:

```
import cv2
import numpy as np

path = r'C:\Users\Naeem\Desktop\Jahanzeb\DIP\DIP Lab\images\moon.jpg'
img = cv2.imread(path, cv2.IMREAD_COLOR)
img = cv2.resize(img, (0, 0), None, .25, .25)

def Mylaplacian(img):
    kernel = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]], np.float32)
    dst = cv2.filter2D(img, -1, kernel)
    return dst

img2 = Mylaplacian(img)
cv2.imshow("Laplacian",img2)

Y = np.array([[-2,-5,-2], [0,0,0], [2,5,2]], np.float32)
X = np.array([[2,0,-2], [5,0,-5], [2,0,-2]], np.float32)

sobelX = cv2.filter2D(img,-1,X)
sobelY = cv2.filter2D(img,-1,Y)

final = abs(sobelY)+abs(sobelX)

horizontalStack = np.concatenate((sobelX, sobelY, final), axis=1)
cv2.imshow("Sobel", horizontalStack)

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

Output:

Due to the higher values in the Sobel filter, it can be seen that the edges are much brighter whereas, in Laplacian, the edges are quite dull. Moreover, Sobel detects both vertical and horizontal edges and then adds them: this helps sobel retain more information about the edges that Laplacian which on works on the