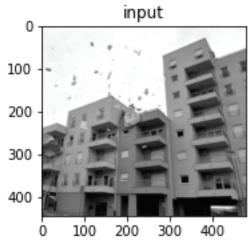
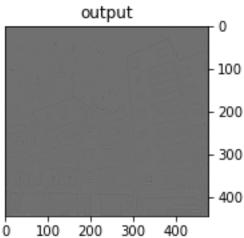
```
1 import cv2 as cv
 2 import matplotlib.pyplot as plt
 3 import numpy as np
 4 import math
 6 # path to the input img
 7 path = 'C:/Users/Raiyan/Desktop/building.jpg'
 9 # reading img + converting from BGR to GRAY
10 img = cv.imread(path)
11 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
12
13 k h = int(input("Enter kernel height: "))
14 k_w = k_h
15 k_{size} = (k_h, k_w)
17 # avg kernel
18 kernel1 = np.zeros( k_size, np.float32)
19 kernel2 = np.zeros( k_size, np.float32)
20 kernel3 = np.zeros( k_size, np.float32)
22 # img height
23 img_h = img.shape[0]
24 # img width
25 img_w = img.shape[1]
26 # kernel height // 2
27 a = kernel1.shape[0] // 2
28 # kernel width // 2
29 b = kernel1.shape[1] // 2
31 pi=3.1416
32 | sigma1 = 1.0
33 normalizing_c = 1.0 / ( 2.0 * sigma1 * sigma1 * pi )
35 # building kernel1
36 for x in range(-a,a+1):
37
       for y in range(-b,b+1):
           r = math.exp( -(x*x + y*y) / (2.0 * sigma1 * sigma1) )
38
39
           r = r* normalizing_c
40
           kernel1[a+x][b+y] = r
42 \text{ sigma2} = 2.5
43 normalizing_c = (1.0 / ( 2.0 * sigma2 * sigma2 * pi ))
44
45 # building kernel2
46 for x in range(-a,a+1):
47
       for y in range(-b,b+1):
           r = math.exp(-(x*x + y*y) / (2.0 * sigma2 * sigma2))
48
           r = r* normalizing_c
49
           kernel2[a+x][b+y] = r
52 # subtracting kernel1 from kernel2
53 kernel2 = kernel2 - kernel1
54
55 # empty op img
56 output = np.zeros((img_h,img_w), np.float32)
57
58 # conv
59 # visiting each pixel in the img
60 # m ta row img e ... for each row ...
61 for i in range(img h):
62
       \# n ta coln img e ... for each coln ...
63
       for j in range(img_w):
           # visiting each pixel in the kernel
64
65
           # a ta row img e ... for each row ...
66
           for x in range(-a,a+1):
               \# b ta coln img e \dots for each coln \dots
67
68
               for y in range(-b,b+1):
                   if 0 <= i-x < img_h and 0 <= j-y < img_w:
69
                       output[i][j] += kernel2[a+x][b+y] * img[i-x][j-y]
70
71
                   else:
72
                       output[i][j] += 0
73
74
75
76 def show_images(images, image_title):
77
       # displaying multiple images side by side
```

localhost:4649/?mode=python 1/2

```
# https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
 78
    using-matplotlib
 79
 80
        # err : was giving weird colormap due to diff in the mechanism of reading img of
    cv2 & matplotlib
 81
        # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
    matplotlib
 82
        # running this once in the code will ALWAYS give gray op
 83
        plt.gray()
 84
 85
        no_of_imgs = len(images)
 86
        f = plt.figure()
 87
        for i in range(no_of_imgs):
 88
 89
            # Debug, plot figure
 90
            axes = f.add_subplot(1, no_of_imgs, i + 1)
            # the last img will show y axis on the RHS instead of LHS(which is by
 91
    default)
 92
 93
            if i==no_of_imgs-1:
 94
                axes.yaxis.tick_right()
 95
 96
            plt.title(image_title[i])
 97
            plt.imshow(images[i], 'gray')
            # plt.rc('font', size=8)
 98
 99
        plt.show(block=True)
100
101 show_images([img,output], ['input', 'output'])
102
```





localhost:4649/?mode=python 2/2