

```

1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 # path to the input img
7 # path = "C:/Users/Raiyan/Desktop/img/03/Image-Processing-and-Computer-Vision-
  Lab/Lab 2/Average filter/Input.png"
8 path = 'C:/Users/Raiyan/Desktop/building.jpg'
9
10 # reading img + converting from BGR to GRAY
11 img = cv.imread(path)
12 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
13
14 k_h = int(input("Enter kernel height: "))
15 k_w = k_h
16 k_size = (k_h, k_w)
17
18 # kernel with neg center value
19 kernel = np.array([[0,-1,0],
20                   [-1,-4,-1],
21                   [0,-1,0]], np.float32)
22
23 # img height
24 img_h = img.shape[0]
25 # img width
26 img_w = img.shape[1]
27 # kernel height // 2
28 a = kernel.shape[0] // 2
29 # kernel width // 2
30 b = kernel.shape[1] // 2
31
32 # empty op img
33 output = np.zeros((img_h, img_w), np.float32)
34
35 # conv
36 # visiting each pixel in the img
37 # m ta row img e ... for each row ...
38 for i in range(img_h):
39     # n ta coln img e ... for each coln ...
40     for j in range(img_w):
41         # empty var for storing all the values
42         values = []
43         # visiting each pixel in the kernel
44         # a ta row img e ... for each row ...
45         for x in range(-a, a+1):
46             # b ta coln img e ... for each coln ...
47             for y in range(-b, b+1):
48                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
49                     output[i][j] += kernel[a+x][b+y] * img[i-x][j-y]
50                 else:
51                     output[i][j] += 0
52
53 out_conv = output
54
55 # scaled
56 def scaled(image):
57     g_m = image - image.min()
58     g_s = 255*(g_m / g_m.max())
59     return g_s.astype(np.float32)
60
61 scaled = scaled(out_conv)
62
63 plt.imshow(scaled, 'gray')
64 plt.title('scaled')
65 plt.show()
66
67 output = out_conv
68
69 # val capping or clipping from 0 - 255
70 for i in range(img_h):
71     for j in range(img_w):
72         if output[i][j] < 0 :
73             output[i][j] = 0
74         elif output[i][j] > 255 :
75             output[i][j] = 255
76

```

```

77 clipped = output.astype(np.float32)
78
79
80 # center of kernel is (-)
81 sharpened = img - out_conv
82
83 output = sharpened
84 # sharpened + clipping from 0 - 255
85 for i in range(img_h):
86     for j in range(img_w):
87         if output[i][j] < 0 :
88             output[i][j] = 0
89         elif output[i][j] > 255 :
90             output[i][j] = 255
91
92 sharpened_and_clipped = output.astype(np.float32)
93
94 # sharpened + scaled
95 g_m = sharpened - sharpened.min()
96 g_s = (g_m / g_m.max()) * 255
97 sharpened_and_scaled = g_s.astype(np.float32)
98
99
100 def show_images(images, image_title):
101     # displaying multiple images side by side
102     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
    using-matplotlib
103
104     # err : was giving weird colormap due to diff in the mechanism of reading img of
    cv2 & matplotlib
105     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
    matplotlib
106     # running this once in the code will ALWAYS give gray op
107     plt.gray()
108
109     no_of_imgs = len(images)
110     f = plt.figure()
111     for i in range(no_of_imgs):
112
113         # Debug, plot figure
114         axes = f.add_subplot(1, no_of_imgs, i + 1)
115         # the last img will show y axis on the RHS instead of LHS(which is by
    default)
116
117         if i==no_of_imgs-1:
118             axes.yaxis.tick_right()
119
120         plt.title(image_title[i])
121         plt.imshow(images[i])
122         # plt.rc('font', size=8)
123         plt.show(block=True)
124
125
126 show_images([img,scaled], ['input', 'scaled'])
127 show_images([clipped, sharpened_and_scaled], ['clipped', 'sharp + scaled'])
128 show_images([img,sharpened_and_scaled], ['input', 'final output'])

```