

```

1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5 minval = 0
6 maxval = 255
7
8 # path to the input img
9 path = 'C:/Users/Raiyan/Desktop/cube.png'
10
11 # reading img + converting from BGR to GRAY
12 img = cv.imread(path)
13 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
14 # resizing image
15 img = cv.resize(img, (int(820/3.5),int(720/3.5)), interpolation = cv.INTER_AREA)
16 img = img/maxval
17 img1 = img
18
19 k_h = int(input("Enter kernel height: "))
20 k_w = k_h
21 k_size = (k_h,k_w)
22
23 # empty kernel
24 kernel = np.zeros( k_size, np.float32)
25
26 # img height
27 img_h = img.shape[0]
28 # img width
29 img_w = img.shape[1]
30 # kernel height // 2
31 a = kernel.shape[0] // 2
32 # kernel width // 2
33 b = kernel.shape[1] // 2
34
35 sigma = 60.0
36 normalizing_c = 1.0 / ( 2.0 * sigma * sigma )
37
38 # building kernel
39 for x in range(-a,a+1):
40     for y in range(-b,b+1):
41         dist = math.sqrt(x*x + y*y) * normalizing_c
42         val = math.exp( -dist ) / ( np.pi * 2.0 * sigma * sigma )
43         kernel[a+x][b+y] = val
44
45 # empty op img
46 output = np.zeros((img_h,img_w), np.float32)
47
48 # conv
49 # visiting each pixel in the img
50 # m ta row img e ... for each row ...
51 for i in range(img_h):
52     # n ta coln img e ... for each coln ...
53     for j in range(img_w):
54         # sum of val to be calc
55         calc = 0
56         # empty kernel for each iter
57         loop_ker = np.zeros( k_size, np.float32)
58         # visiting each pixel in the kernel
59         # a ta row img e ... for each row ...
60         for x in range(-a,a+1):
61             # b ta coln img e ... for each coln ...
62             for y in range(-b,b+1):
63                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
64                     dist = math.sqrt( np.power( img[i][j] - img[i-x][j-y], 2 ) ) *
normalizing_c
65                     val = math.exp( -dist ) / ( np.pi * 2.0 * sigma * sigma )
66                     loop_ker[a+x][b+y] = kernel[a+x][b+y] * val
67
68             for x in range(-a,a+1):
69                 for y in range(-b,b+1):
70                     if 0 <= i-x < img_h and 0 <= j-y < img_w:
71                         calc += kernel[a+x][b+y] * img[i-x][j-y]
72                     else:
73                         calc += 0
74                 calc = calc / ( loop_ker.shape[0] * loop_ker.shape[1] )
75                 output[i][j] = calc
76 output *= maxval

```

```
77
78
79 def show_images(images, image_title):
80     # displaying multiple images side by side
81     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
    using-matplotlib
82
83     # err : was giving weird colormap due to diff in the mechanism of reading img of
    cv2 & matplotlib
84     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
    matplotlib
85     # running this once in the code will ALWAYS give gray op
86     plt.gray()
87
88     no_of_imgs = len(images)
89     f = plt.figure()
90     for i in range(no_of_imgs):
91
92         # Debug, plot figure
93         axes = f.add_subplot(1, no_of_imgs, i + 1)
94         # the last img will show y axis on the RHS instead of LHS(which is by
    default)
95
96         if i==no_of_imgs-1:
97             axes.yaxis.tick_right()
98
99         plt.title(image_title[i])
100         plt.imshow(images[i])
101         # plt.rc('font', size=8)
102         plt.show(block=True)
103
104 show_images([img1,output], ['input', 'output'])
105
```