

```

1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6
7 # path to the input img
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 img1 = img
14 # sobel vertical
15 kernel_v = np.array([[-1,0,1],
16                      [-2,0,2],
17                      [-1,0,1]], np.float32)
18
19 # sobel horizontal
20 kernel_h = np.array([[-1,-2,-1],
21                      [0,0,0],
22                      [1,2,1]], np.float32)
23
24 # img height
25 img_h = img.shape[0]
26 # img width
27 img_w = img.shape[1]
28 # kernel height // 2
29 a = kernel_v.shape[0] // 2
30 # kernel width // 2
31 b = kernel_v.shape[1] // 2
32
33 k_h = kernel_v.shape[0]
34 k_w = k_h
35
36 # empty op img
37 output = np.zeros((img_h,img_w), np.float32)
38 output_v = np.zeros((img_h,img_w), np.float32)
39 output_h = np.zeros((img_h,img_w), np.float32)
40
41 def clipped_op(img):
42     for i in range(img_h):
43         for j in range(img_w):
44             if result[i][j] > 255:
45                 result[i][j] = 255
46             if result[i][j] < 0:
47                 result[i][j] = 0
48     img = img.astype(np.float32)
49     return img
50
51 # conv 1
52 # visiting each pixel in the img
53 # m ta row img e ... for each row ...
54 for i in range(img_h):
55     # n ta coln img e ... for each coln ...
56     for j in range(img_w):
57         # empty var for storing all the values
58         values = []
59         # visiting each pixel in the kernel
60         # a ta row img e ... for each row ...
61         for x in range(-a,a+1):
62             # b ta coln img e ... for each coln ...
63             for y in range(-b,b+1):
64                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
65                     output[i][j] += kernel_v[a+x][b+y] * img[i-x][j-y]
66                 else:
67                     output[i][j] += 0
68
69 output_v = output
70 output = np.zeros((img_h,img_w), np.float32)
71
72 # conv 2
73 # visiting each pixel in the img
74 # m ta row img e ... for each row ...
75 for i in range(img_h):
76     # n ta coln img e ... for each coln ...
77     for j in range(img_w):

```

```

78     # empty var for storing all the values
79     values = []
80     # visiting each pixel in the kernel
81     # a ta row img e ... for each row ...
82     for x in range(-a,a+1):
83         # b ta coln img e ... for each coln ...
84         for y in range(-b,b+1):
85             if 0 <= i-x < img_h and 0 <= j-y < img_w:
86                 output[i][j] += kernel_h[a+x][b+y] * img[i-x][j-y]
87             else:
88                 output[i][j] += 0
89
90 output_h = output
91 result = output_v + output_h
92
93 result = clipped_op(result)
94
95 # plt.imshow(result, 'gray')
96 # plt.title("sobel_v+h")
97 # plt.show()
98
99 img = img.astype(np.float32)
100 img += result
101 img = clipped_op(img)
102
103 # plt.imshow(img, 'gray')
104 # plt.title("img+sobel")
105 # plt.show()
106
107 def show_images(images, image_title):
108     # displaying multiple images side by side
109     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
    using-matplotlib
110
111     # err : was giving weird colormap due to diff in the mechanism of reading img of
    cv2 & matplotlib
112     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
    matplotlib
113     # running this once in the code will ALWAYS give gray op
114     plt.gray()
115
116     no_of_imgs = len(images)
117     f = plt.figure()
118     for i in range(no_of_imgs):
119
120         # Debug, plot figure
121         axes = f.add_subplot(1, no_of_imgs, i + 1)
122         # the last img will show y axis on the RHS instead of LHS(which is by
    default)
123
124         if i==no_of_imgs-1:
125             axes.yaxis.tick_right()
126
127         plt.title(image_title[i])
128         plt.imshow(images[i])
129         # plt.rc('font', size=8)
130     plt.show(block=True)
131
132 show_images([img1,output_h], ['input', 'sobel_h'])
133 show_images([output_v,result], ['sobel_v', 'sobel_v+h'])
134 show_images([img1,img], ['input', 'final output'])
135
136
137

```