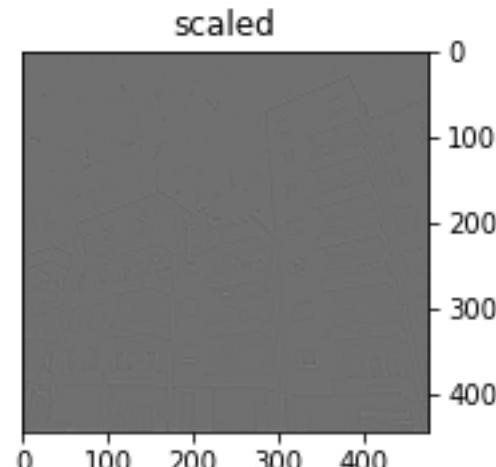
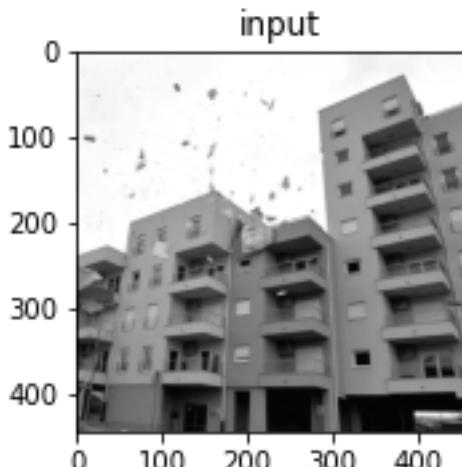


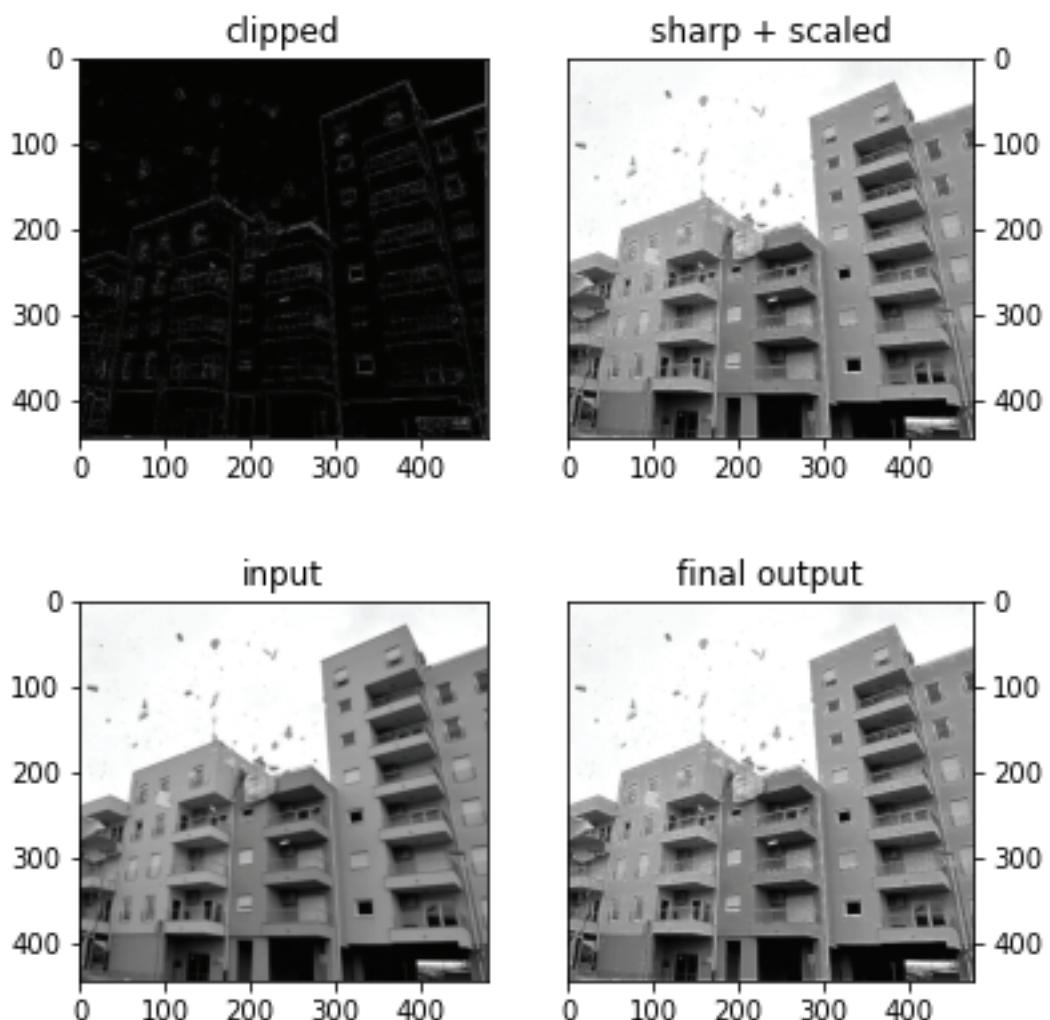
```
1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 # path to the input img
7 path = 'C:/Users/Raiyan/Desktop/building.jpg'
8
9 # reading img + converting from BGR to GRAY
10 img = cv.imread(path)
11 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
12
13 k_h = int(input("Enter kernel height: "))
14 k_w = k_h
15 k_size = (k_h,k_w)
16
17 # kernel with neg center value
18 kernel = np.array(([0,-1,0],
19                     [-1,+4,-1],
20                     [0,-1,0]), np.float32)
21
22 # img height
23 img_h = img.shape[0]
24 # img width
25 img_w = img.shape[1]
26 # kernel height // 2
27 a = kernel.shape[0] // 2
28 # kernel width // 2
29 b = kernel.shape[1] // 2
30
31 # empty op img
32 output = np.zeros((img_h,img_w), np.float32)
33
34 # conv
35 # visiting each pixel in the img
36 # m ta row img e ... for each row ...
37 for i in range(img_h):
38     # n ta coln img e ... for each coln ...
39     for j in range(img_w):
40         # empty var for storing all the values
41         values = []
42         # visiting each pixel in the kernel
43         # a ta row img e ... for each row ...
44         for x in range(-a,a+1):
45             # b ta coln img e ... for each coln ...
46             for y in range(-b,b+1):
47                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
48                     output[i][j] += kernel[a+x][b+y] * img[i-x][j-y]
49                 else:
50                     output[i][j] += 0
51
52 out_conv = output
53
54 # scaled
55 def scaled(image):
56     g_m = image - image.min()
57     g_s = 255*(g_m / g_m.max())
58     return g_s.astype(np.float32)
59
60 scaled = scaled(out_conv)
61
62 plt.imshow(scaled, 'gray')
63 plt.title('scaled')
64 plt.show()
65
66 output = out_conv
67
68 # val capping or clipping from 0 - 255
69 for i in range(img_h):
70     for j in range(img_w):
71         if output[i][j] <0 :
72             output[i][j] = 0
73         elif output[i][j] >255 :
74             output[i][j] = 255
75
76 clipped = output.astype(np.float32)
77
```

```

78 # center of kernel is (+)
79 sharpened = img + out_conv
80
81 output = sharpened
82 # sharpened + clipping from 0 - 255
83 for i in range(img_h):
84     for j in range(img_w):
85         if output[i][j] < 0 :
86             output[i][j] = 0
87         elif output[i][j] >255 :
88             output[i][j] = 255
89
90 sharpened_and_clipped = output.astype(np.float32)
91
92 # sharpened + scaled
93 g_m = sharpened - sharpened.min()
94 g_s = (g_m / g_m.max()) * 255
95 sharpened_and_scaled = g_s.astype(np.float32)
96
97
98 def show_images(images, image_title):
99     # displaying multiple images side by side
100    # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
using-matplotlib
101
102    # err : was giving weird colormap due to diff in the mechanism of reading img of
cv2 & matplotlib
103    # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
matplotlib
104    # running this once in the code will ALWAYS give gray op
105    plt.gray()
106
107    no_of_imgs = len(images)
108    f = plt.figure()
109    for i in range(no_of_imgs):
110
111        # Debug, plot figure
112        axes = f.add_subplot(1, no_of_imgs, i + 1)
113        # the last img will show y axis on the RHS instead of LHS(which is by
default)
114
115        if i==no_of_imgs-1:
116            axes.yaxis.tick_right()
117
118        plt.title(image_title[i])
119        plt.imshow(images[i])
120        # plt.rc('font', size=8)
121        plt.show(block=True)
122
123
124 show_images([img,scaled], ['input', 'scaled'])
125 show_images([clipped, sharpened_and_scaled], ['clipped', 'sharp + scaled'])
126 show_images([img,sharpened_and_scaled], ['input', 'final output'])
127
128
129
130

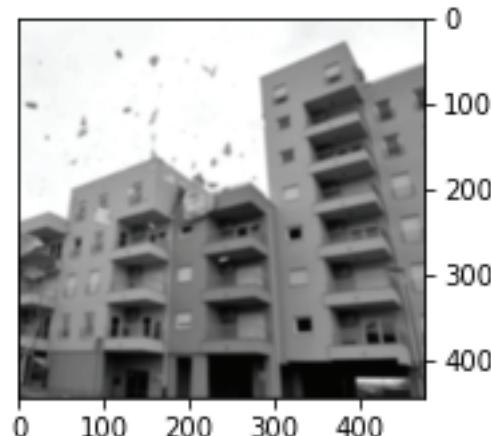
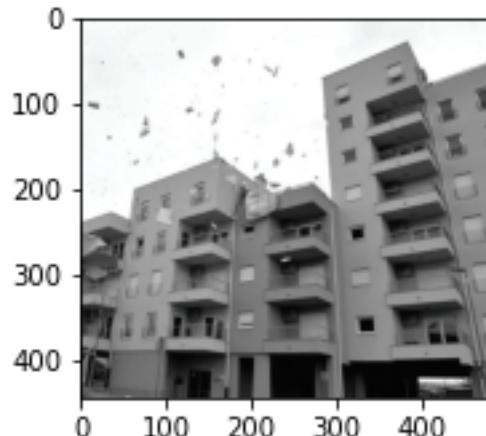
```





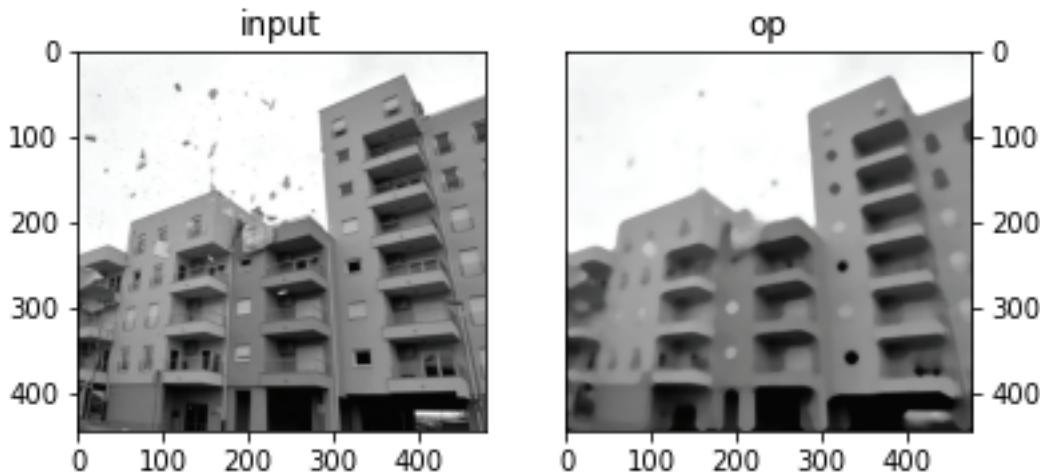
```
1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 def show_images(images):
7     # displaying multiple images side by side
8     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
9     # using-matplotlib
10    # err : was giving weird colormap due to diff in the mechanism of reading img of
11    # cv2 & matplotlib
12    # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
13    # matplotlib
14    # running this once in the code will ALWAYS give gray op
15    plt.gray()
16
17    n = len(images)
18    f = plt.figure()
19    for i in range(n):
20        # Debug, plot figure
21        axes = f.add_subplot(1, n, i + 1)
22        # the last img will show y axis on the RHS instead of LHS(which is by
23        # default)
24        if i==n-1:
25            axes.yaxis.tick_right()
26        plt.imshow(images[i])
27
28 # path to the input img
29 # path = "C:/Users/Raiyan/Desktop/img/03/Image-Processing-and-Computer-Vision-Lab/Lab
2/Average filter/Input.png"
30 path = 'C:/Users/Raiyan/Desktop/building.jpg'
31
32 # reading img + converting from BGR to GRAY
33 img = cv.imread(path)
34 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
35
36 k_h = int(input("Enter kernel height: "))
37 k_w = k_h
38 k_size = (k_h,k_w)
39
40 # avg kernel
41 kernel = np.ones( k_size, np.float32)
42
43 # img height
44 img_h = img.shape[0]
45 # img width
46 img_w = img.shape[1]
47 #kernel height
48 a = kernel.shape[0] // 2
49 # kernel width
50 b = kernel.shape[1] // 2
51
52 # empty op img
53 output = np.zeros((img_h,img_w), np.float32)
54
55 # sum of the values of the kernel
56 k_sum = kernel.sum()
57 # print(f'ksum is {ksum}')
58
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):
64
65         # empty var for calculating a summed value
66         value = 0
67         # visiting each pixel in the kernel
68         # a ta row img e ... for each row ...
69         for x in range(-a,a+1):
70             # b ta coln img e ... for each coln ...
71             for y in range(-b,b+1):
```

```
73         if 0 <= i+x < img_h and 0 <= j+y < img_w:  
74             value = value + kernel[a+x][b+y] * img[i+x][j+y]  
75         else:  
76             value = value + 0  
77     value = value / k_sum  
78     output[i][j] = value  
79  
80 show_images([img,output])  
81  
82
```



```
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 # avg kernel
19 kernel = np.ones( k_size, np.float32)
20
21 # img height
22 img_h = img.shape[0]
23 # img width
24 img_w = img.shape[1]
25 # kernel height // 2
26 a = kernel.shape[0] // 2
27 # kernel width // 2
28 b = kernel.shape[1] // 2
29
30 # empty op img
31 output = np.zeros((img_h,img_w), np.float32)
32
33 # sum of the values of the kernel
34 k_sum = kernel.sum()
35 # print(f'ksum is {ksum}')
36
37
38 # visiting each pixel in the img
39 # m ta row img e ... for each row ...
40 for i in range(img_h):
41     # n ta coln img e ... for each coln ...
42     for j in range(img_w):
43         # empty var for storing all the values
44         values = []
45         # visiting each pixel in the kernel
46         # a ta row img e ... for each row ...
47         for x in range(-a,a+1):
48             # b ta coln img e ... for each coln ...
49             for y in range(-b,b+1):
50                 if 0 <= i+x < img_h and 0 <= j+y < img_w:
51                     calculated_val = kernel[a+x][b+y] * img[i+x][j+y]
52                     values.append( calculated_val )
53                 else:
54                     values.append(0)
55         values.sort()
56
57         median_val = len(values) // 2
58
59         output[i][j] = values[median_val]
60         output[i][j] /= k_sum
61
62
63 def show_images(images):
64     # displaying multiple images side by side
65     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
using-matplotlib
66
67     # err : was giving weird colormap due to diff in the mechanism of reading img of
cv2 & matplotlib
68     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
matplotlib
69     # running this once in the code will ALWAYS give gray op
70     plt.gray()
71
72     no_of_imgs = len(images)
73     f = plt.figure()
```

```
74     for i in range(no_of_imgs):
75
76         # Debug, plot figure
77         axes = f.add_subplot(1, no_of_imgs, i + 1)
78         # the last img will show y axis on the RHS instead of LHS(which is by
79         # default)
80
81         if i==no_of_imgs-1:
82             axes.yaxis.tick_right()
83         if i==0 :
84             plt.title('input')
85         else :
86             plt.title('op')
87             plt.imshow(images[i])
88             # plt.rc('font', size=8)
89             plt.show(block=True)
90
91 show_images([img,output])
```



```

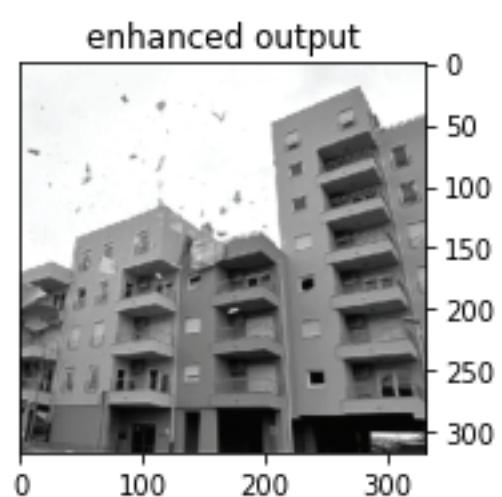
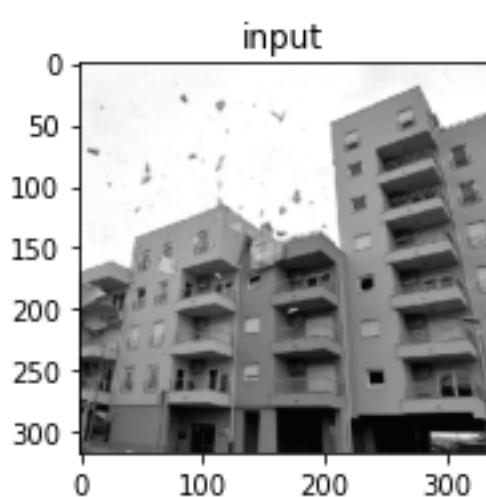
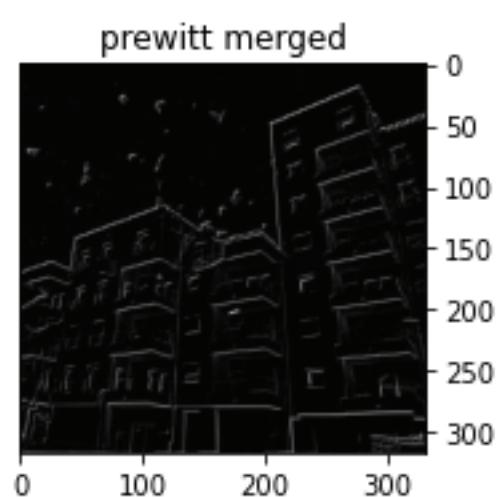
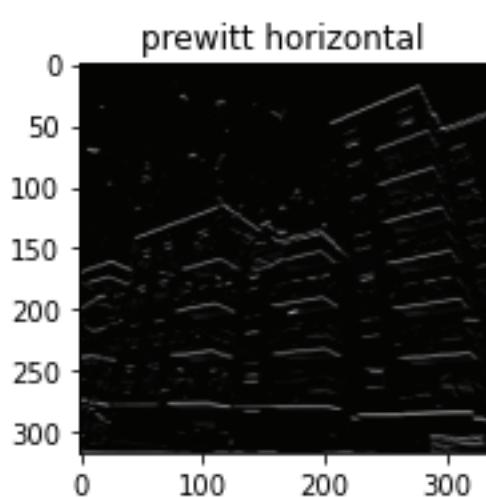
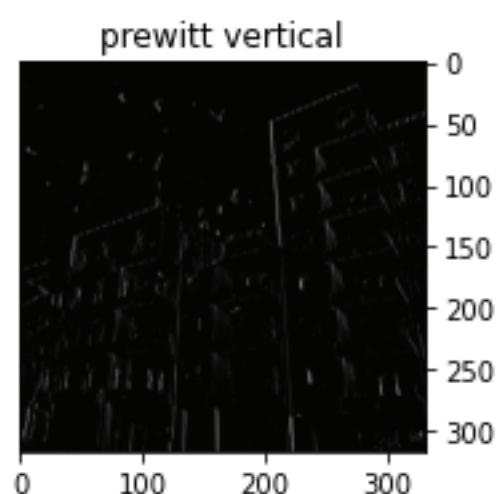
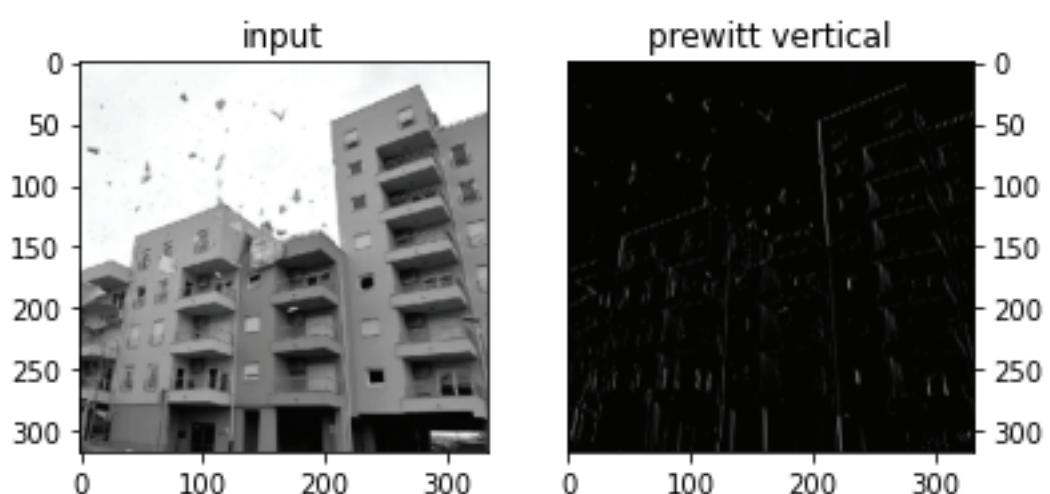
1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 # path to the input img
7 path = 'C:/Users/Raiyan/Desktop/building_332x317.jpg'
8
9 # reading img + converting from BGR to GRAY
10 img = cv.imread(path)
11 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
12 img1 = img
13
14 # prewitt-y kernel
15 kernel = np.array([[-1,0,1],
16                     [-1,0,1],
17                     [-1,0,1]], np.float32)
18
19 k_h = kernel.shape[0]
20 k_w = k_h
21 k_size = (k_h,k_w)
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         # sum of val to be calc
42         calc = 0
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                     calc += kernel[a+x][b+y] * img[i-x][j-y]
50                 else:
51                     calc += 0
52         calc = calc / (k_w*k_h)
53         output[i][j] = calc
54
55 prewitt_vertical = output
56 for i in range(img_h):
57     for j in range(img_w):
58         if prewitt_vertical[i][j] > 255:
59             prewitt_vertical[i][j] = 255
60         elif prewitt_vertical[i][j] < 0:
61             prewitt_vertical[i][j] = 0
62
63 output = np.zeros((img_h,img_w), np.float32)
64
65 # prewitt-x kernel
66 kernel = np.array([[-1,-1,-1],
67                     [0,0,0],
68                     [1,1,1]], np.float32)
69
70 # conv
71 # visiting each pixel in the img
72 # m ta row img e ... for each row ...
73 for i in range(img_h):
74     # n ta coln img e ... for each coln ...
75     for j in range(img_w):
76         # sum of val to be calc
77         calc = 0

```

```

78     # visiting each pixel in the kernel
79     # a ta row img e ... for each row ...
80     for x in range(-a,a+1):
81         # b ta coln img e ... for each coln ...
82         for y in range(-b,b+1):
83             if 0 <= i-x < img_h and 0 <= j-y < img_w:
84                 calc += kernel[a+x][b+y] * img[i-x][j-y]
85             else:
86                 calc += 0
87         calc = calc / (k_w*k_h)
88         output[i][j] = calc
89
90 prewitt_horizontal = output
91 for i in range(img_h):
92     for j in range(img_w):
93         if prewitt_horizontal[i][j] > 255:
94             prewitt_horizontal[i][j] = 255
95         elif prewitt_horizontal[i][j] < 0:
96             prewitt_horizontal[i][j] = 0
97
98 prewitt_merged = prewitt_horizontal + prewitt_vertical
99 for i in range(img_h):
100    for j in range(img_w):
101        if prewitt_merged[i][j] > 255:
102            prewitt_merged[i][j] = 255
103        elif prewitt_merged[i][j] < 0:
104            prewitt_merged[i][j] = 0
105
106 img = img + prewitt_merged
107
108 for i in range(img_h):
109     for j in range(img_w):
110         if img[i][j] > 255:
111             img[i][j] = 255
112         elif img[i][j] < 0:
113             img[i][j] = 0
114
115
116 def show_images(images, image_title):
117     # displaying multiple images side by side
118     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
119     # using-matplotlib
120     # err : was giving weird colormap due to diff in the mechanism of reading img of
121     # cv2 & matplotlib
122     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
123     # matplotlib
124     # running this once in the code will ALWAYS give gray op
125     plt.gray()
126
127     no_of_imgs = len(images)
128     f = plt.figure()
129     for i in range(no_of_imgs):
130
131         # Debug, plot figure
132         axes = f.add_subplot(1, no_of_imgs, i + 1)
133         # the last img will show y axis on the RHS instead of LHS(which is by
134         # default)
135
136         if i==no_of_imgs-1:
137             axes.yaxis.tick_right()
138
139         plt.title(image_title[i])
140         plt.imshow(images[i], 'gray')
141         # plt.rc('font', size=8)
142         plt.show(block=True)
143
144 show_images([img1,prewitt_vertical],
145             ['input', 'prewitt vertical'])
146 show_images([prewitt_horizontal, prewitt_merged],
147             ['prewitt horizontal', 'prewitt merged'])
148 show_images([img1,img],
149             ['input', 'enhanced output'])
150

```

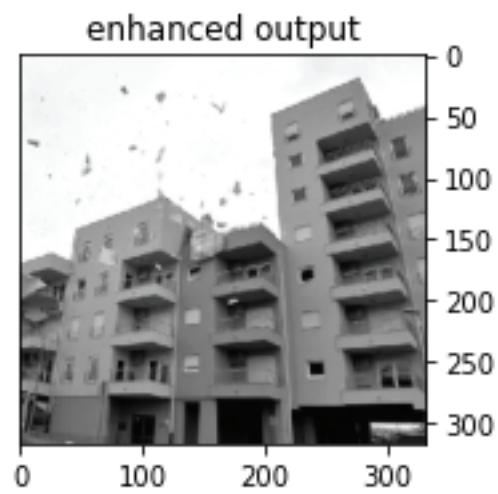
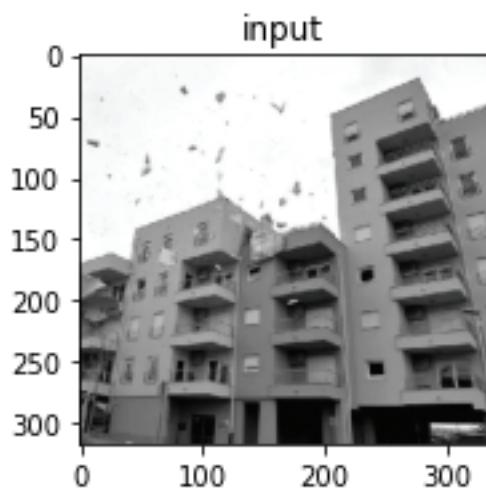
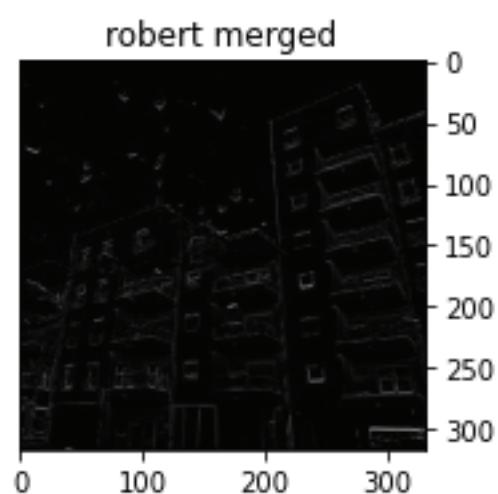
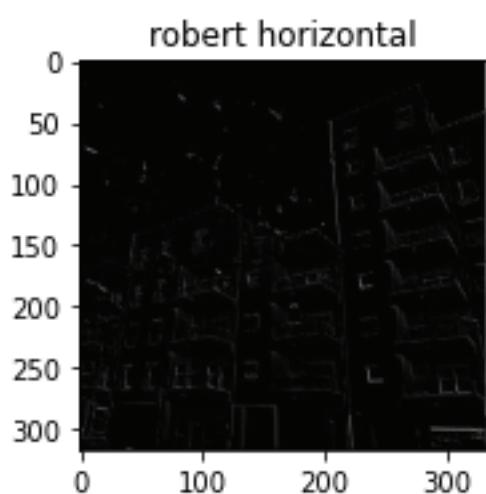
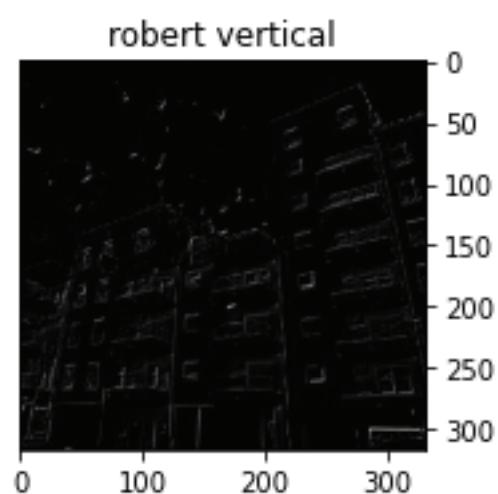
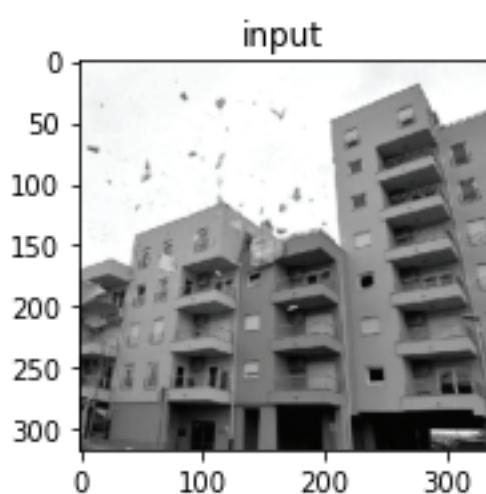


```
1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 # path to the input img
7 path = 'C:/Users/Raiyan/Desktop/building_332x317.jpg'
8
9 # reading img + converting from BGR to GRAY
10 img = cv.imread(path)
11 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
12 img1 = img
13
14 # robert-x kernel
15 kernel = np.array(([1,0],
16 [0,-1]), np.float32)
17
18 k_h = kernel.shape[0]
19 k_w = k_h
20 k_size = (k_h,k_w)
21
22 # img height
23 img_h = img.shape[0]
24 # img width
25 img_w = img.shape[1]
26 # kernel height // 2
27 a = kernel.shape[0] // 2
28 # kernel width // 2
29 b = kernel.shape[1] // 2
30
31 # empty op img
32 output = np.zeros((img_h,img_w), np.float32)
33
34 # conv
35 # visiting each pixel in the img
36 # m ta row img e ... for each row ...
37 for i in range(img_h):
38     # n ta coln img e ... for each coln ...
39     for j in range(img_w):
40         # sum of val to be calc
41         calc = 0
42         # visiting each pixel in the kernel
43         # a ta row img e ... for each row ...
44         for x in range(-a,a+1):
45             # b ta coln img e ... for each coln ...
46             for y in range(-b,b+1):
47                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
48                     if 0<= a+x < k_h and 0<= b+y < k_w:
49                         calc += kernel[a+x][b+y] * img[i-x][j-y]
50                 else:
51                     calc += 0
52         calc = calc / (k_w*k_h)
53         output[i][j] = calc
54
55 robert_vertical = output
56 for i in range(img_h):
57     for j in range(img_w):
58         if robert_vertical[i][j] > 255:
59             robert_vertical[i][j] = 255
60         elif robert_vertical[i][j] < 0:
61             robert_vertical[i][j] = 0
62
63 output = np.zeros((img_h,img_w), np.float32)
64
65 # robert-y kernel
66 kernel = np.array(([0,1],
67 [-1,0]), np.float32)
68
69 # conv
70 # visiting each pixel in the img
71 # m ta row img e ... for each row ...
72 for i in range(img_h):
73     # n ta coln img e ... for each coln ...
74     for j in range(img_w):
75         # sum of val to be calc
76         calc = 0
77         # visiting each pixel in the kernel
```

```

78     # a ta row img e ... for each row ...
79     for x in range(-a,a+1):
80         # b ta coln img e ... for each coln ...
81         for y in range(-b,b+1):
82             if 0 <= i-x < img_h and 0 <= j-y < img_w:
83                 if 0<= a+x < k_h and 0<= b+y < k_w:
84                     calc += kernel[a+x][b+y] * img[i-x][j-y]
85             else:
86                 calc += 0
87     calc = calc / (k_w*k_h)
88     output[i][j] = calc
89
90 robert_horizontal = output
91 for i in range(img_h):
92     for j in range(img_w):
93         if robert_horizontal[i][j] > 255:
94             robert_horizontal[i][j] = 255
95         elif robert_horizontal[i][j] < 0:
96             robert_horizontal[i][j] = 0
97
98 robert_merged = robert_horizontal + robert_vertical
99 for i in range(img_h):
100    for j in range(img_w):
101        if robert_merged[i][j] > 255:
102            robert_merged[i][j] = 255
103        elif robert_merged[i][j] < 0:
104            robert_merged[i][j] = 0
105
106 img = img + robert_merged
107
108 for i in range(img_h):
109     for j in range(img_w):
110         if img[i][j] > 255:
111             img[i][j] = 255
112         elif img[i][j] < 0:
113             img[i][j] = 0
114
115 def show_images(images, image_title):
116     # displaying multiple images side by side
117     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
118     # using-matplotlib
119     # err : was giving weird colormap due to diff in the mechanism of reading img of
120     # cv2 & matplotlib
121     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
122     # matplotlib
123     # running this once in the code will ALWAYS give gray op
124     plt.gray()
125
126     no_of_imgs = len(images)
127     f = plt.figure()
128     for i in range(no_of_imgs):
129
130         # Debug, plot figure
131         axes = f.add_subplot(1, no_of_imgs, i + 1)
132         # the last img will show y axis on the RHS instead of LHS(which is by
133         # default)
134
135         if i==no_of_imgs-1:
136             axes.yaxis.tick_right()
137
138         plt.title(image_title[i])
139         plt.imshow(images[i], 'gray')
140         # plt.rc('font', size=8)
141         plt.show(block=True)
142
143 show_images([img1,robert_vertical],
144             ['input', 'robert vertical'])
145 show_images([robert_horizontal, robert_merged],
146             ['robert horizontal', 'robert merged'])
147 show_images([img1,img],
148             ['input', 'enhanced output'])

```

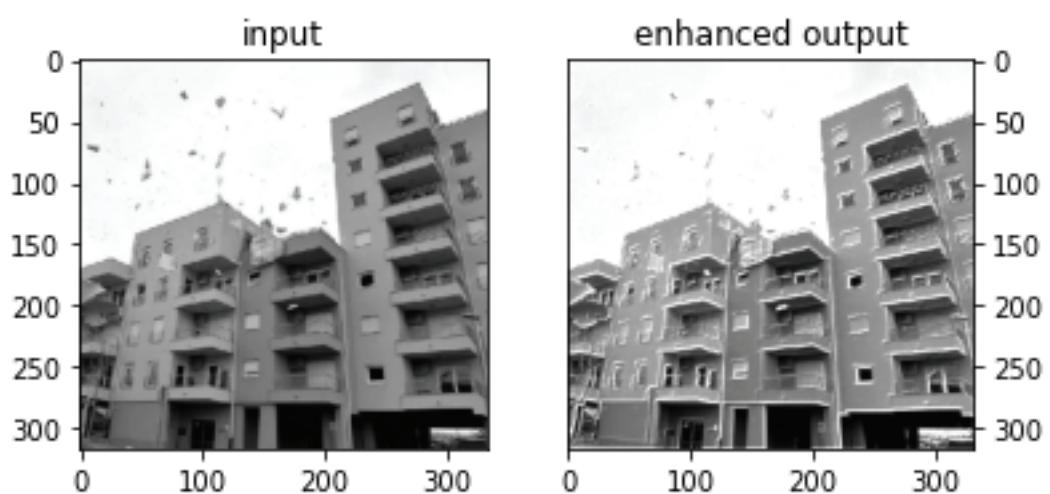
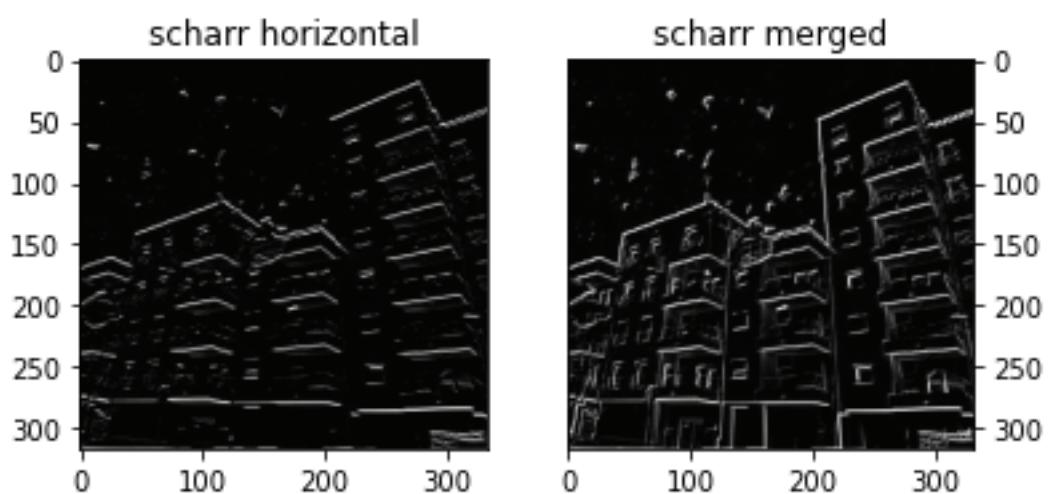
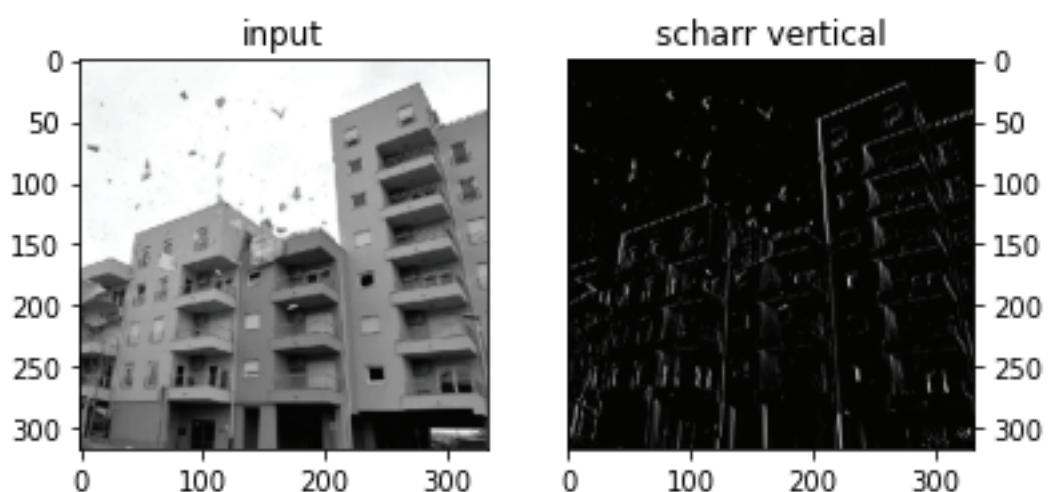


```
1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 # path to the input img
7 path = 'C:/Users/Raiyan/Desktop/building_332x317.jpg'
8
9 # reading img + converting from BGR to GRAY
10 img = cv.imread(path)
11 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
12 img1 = img
13
14 # scharr-y kernel
15 kernel = np.array([[-3,0,3],
16                   [-10,0,10],
17                   [-3,0,3]], np.float32)
18
19 k_h = kernel.shape[0]
20 k_w = k_h
21 k_size = (k_h,k_w)
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         # sum of val to be calc
42         calc = 0
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                     calc += kernel[a+x][b+y] * img[i-x][j-y]
50                 else:
51                     calc += 0
52         calc = calc / (k_w*k_h)
53         output[i][j] = calc
54
55 scharr_vertical = output
56 for i in range(img_h):
57     for j in range(img_w):
58         if scharr_vertical[i][j] > 255:
59             scharr_vertical[i][j] = 255
60         elif scharr_vertical[i][j] < 0:
61             scharr_vertical[i][j] = 0
62
63 output = np.zeros((img_h,img_w), np.float32)
64
65 # scharr-x kernel
66 kernel = np.array([[-3,-10,-3],
67                   [0,0,0],
68                   [3,10,3]], np.float32)
69
70 # conv
71 # visiting each pixel in the img
72 # m ta row img e ... for each row ...
73 for i in range(img_h):
74     # n ta coln img e ... for each coln ...
75     for j in range(img_w):
76         # sum of val to be calc
77         calc = 0
```

```

78 # visiting each pixel in the kernel
79 # a ta row img e ... for each row ...
80 for x in range(-a,a+1):
81     # b ta coln img e ... for each coln ...
82     for y in range(-b,b+1):
83         if 0 <= i-x < img_h and 0 <= j-y < img_w:
84             calc += kernel[a+x][b+y] * img[i-x][j-y]
85         else:
86             calc += 0
87     calc = calc / (k_w*k_h)
88     output[i][j] = calc
89
90 scharr_horizontal = output
91 for i in range(img_h):
92     for j in range(img_w):
93         if scharr_horizontal[i][j] > 255:
94             scharr_horizontal[i][j] = 255
95         elif scharr_horizontal[i][j] < 0:
96             scharr_horizontal[i][j] = 0
97
98 scharr_merged = scharr_horizontal + scharr_vertical
99 for i in range(img_h):
100    for j in range(img_w):
101        if scharr_merged[i][j] > 255:
102            scharr_merged[i][j] = 255
103        elif scharr_merged[i][j] < 0:
104            scharr_merged[i][j] = 0
105
106 img = img + scharr_merged
107
108 for i in range(img_h):
109     for j in range(img_w):
110         if img[i][j] > 255:
111             img[i][j] = 255
112         elif img[i][j] < 0:
113             img[i][j] = 0
114
115
116 def show_images(images, image_title):
117     # displaying multiple images side by side
118     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
using-matplotlib
119
120     # err : was giving weird colormap due to diff in the mechanism of reading img of
cv2 & matplotlib
121     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
matplotlib
122     # running this once in the code will ALWAYS give gray op
123     plt.gray()
124
125     no_of_imgs = len(images)
126     f = plt.figure()
127     for i in range(no_of_imgs):
128
129         # Debug, plot figure
130         axes = f.add_subplot(1, no_of_imgs, i + 1)
131         # the last img will show y axis on the RHS(which is by
default)
132
133         if i==no_of_imgs-1:
134             axes.yaxis.tick_right()
135
136         plt.title(image_title[i])
137         plt.imshow(images[i], 'gray')
138         # plt.rc('font', size=8)
139     plt.show(block=True)
140
141 show_images([img1,scharr_vertical],
142             ['input', 'scharr vertical'])
143 show_images([scharr_horizontal, scharr_merged],
144             ['scharr horizontal', 'scharr merged'])
145 show_images([img1,img],
146             ['input', 'enhanced output'])
147
148

```

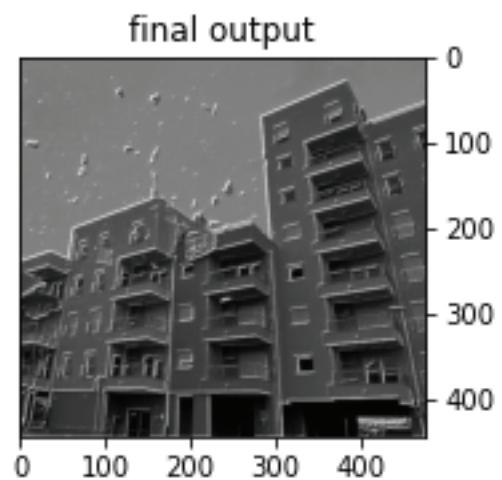
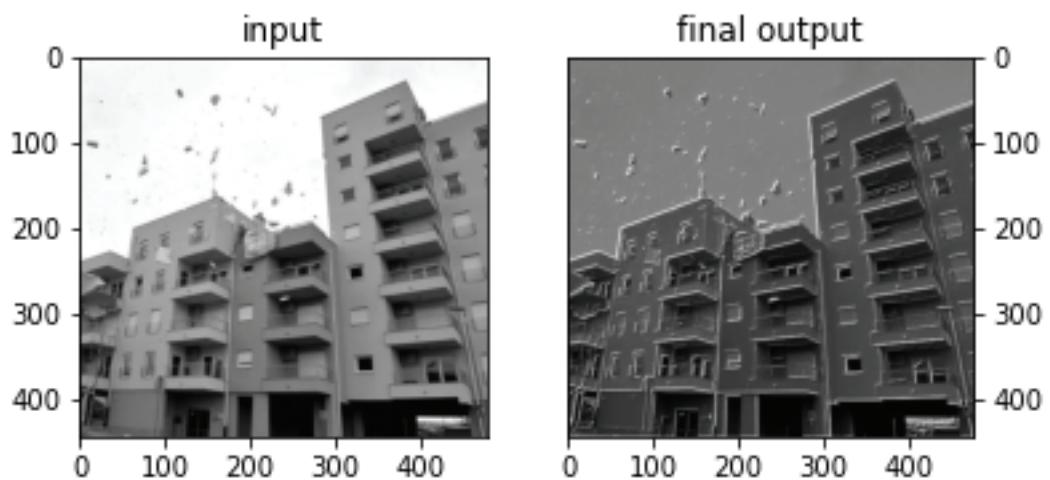
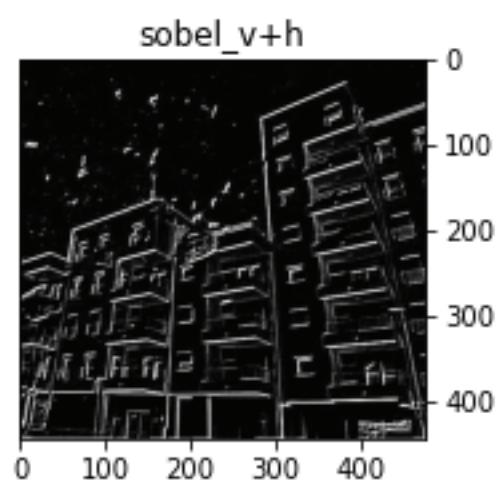
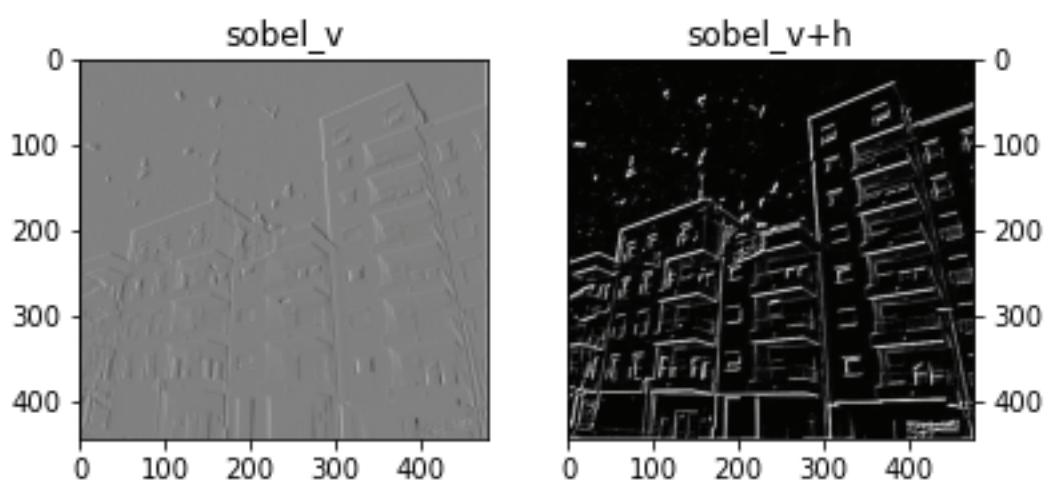
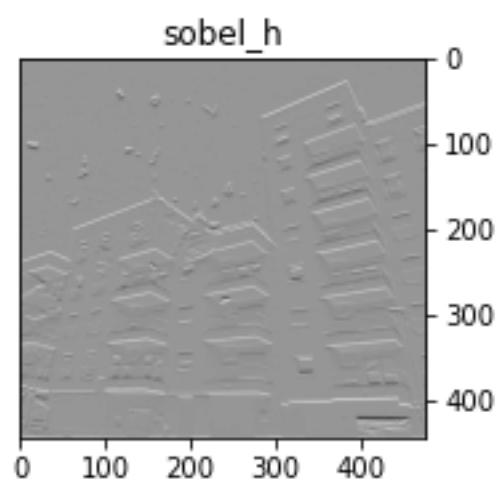
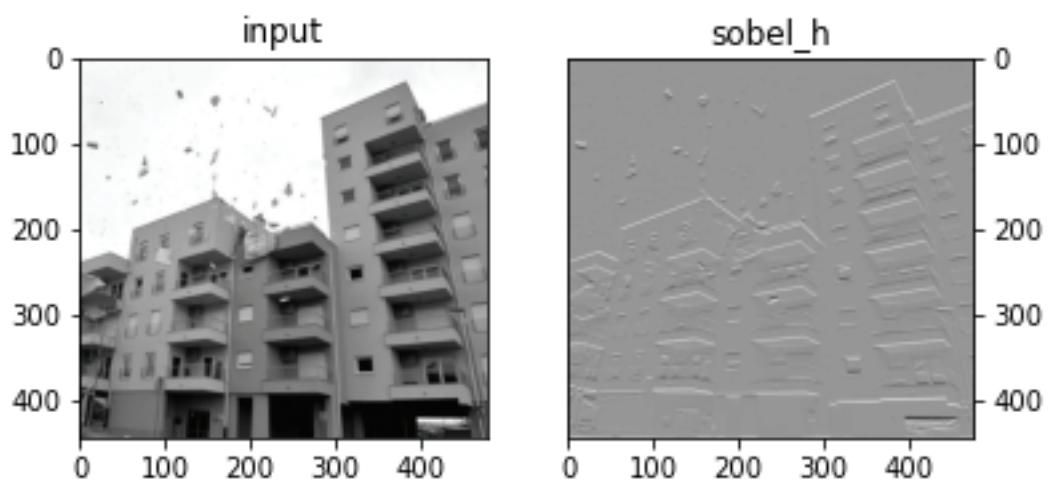


```

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):

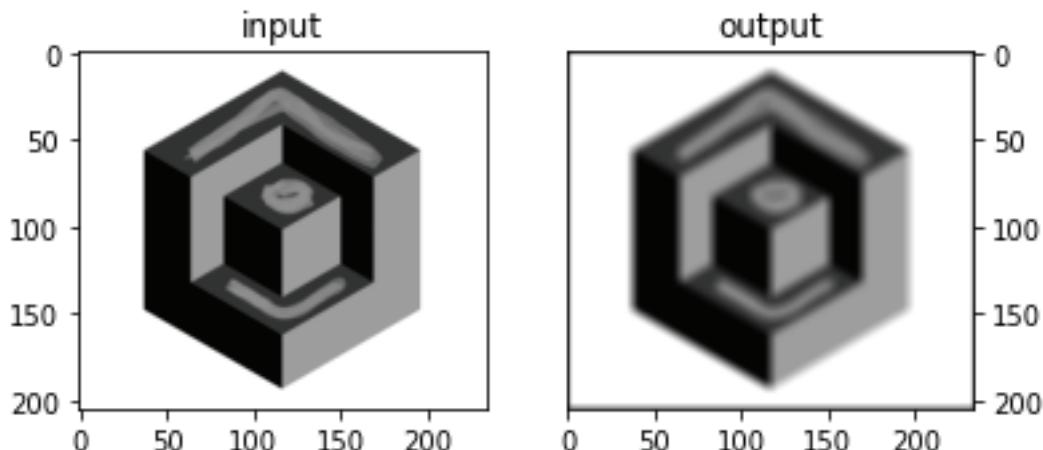
```

```
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
112     # cv2 & matplotlib
113     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-matplotlib
114     # running this once in the code will ALWAYS give gray op
115     plt.gray()
116
117     no_of_imgs = len(images)
118     f = plt.figure()
119     for i in range(no_of_imgs):
120
121         # Debug, plot figure
122         axes = f.add_subplot(1, no_of_imgs, i + 1)
123         # the last img will show y axis on the RHS instead of LHS(which is by default)
124
125         if i==no_of_imgs-1:
126             axes.yaxis.tick_right()
127
128         plt.title(image_title[i])
129         plt.imshow(images[i])
130         # plt.rc('font', size=8)
131         plt.show(block=True)
132
133 show_images([img1,output_h], ['input', 'sobel_h'])
134 show_images([output_v,result], ['sobel_v', 'sobel_v+h'])
135 show_images([img1,img], ['input', 'final output'])
136
137
```



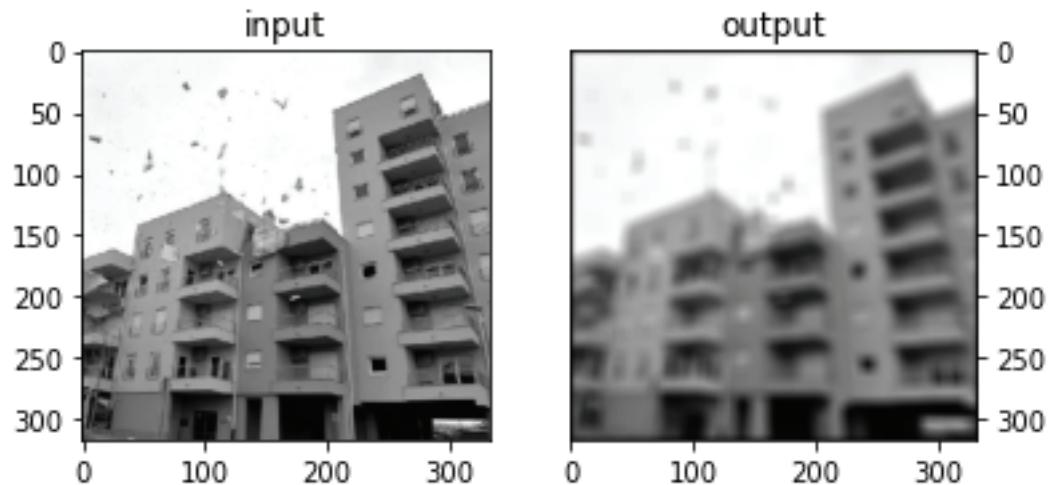
```
1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5 minval = 0
6 maxval = 255
7
8 # path to the input img
9 path = 'C:/Users/Raiyan/Desktop/cube.png'
10
11 # reading img + converting from BGR to GRAY
12 img = cv.imread(path)
13 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
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-
82     # using-matplotlib
83     # err : was giving weird colormap due to diff in the mechanism of reading img of
84     # cv2 & matplotlib
85     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
86     # matplotlib
87     # running this once in the code will ALWAYS give gray op
88     plt.gray()
89
90     no_of_imgs = len(images)
91     f = plt.figure()
92     for i in range(no_of_imgs):
93
94         # Debug, plot figure
95         axes = f.add_subplot(1, no_of_imgs, i + 1)
96         # the last img will show y axis on the RHS instead of LHS(which is by
97         # default)
98
99         if i==no_of_imgs-1:
100             axes.yaxis.tick_right()
101
102         plt.title(image_title[i])
103         plt.imshow(images[i])
104         # plt.rc('font', size=8)
105         plt.show(block=True)
```



```
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/building_332x317.jpg'
8
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)
16
17 # empty kernel
18 kernel = np.zeros( k_size, np.float32)
19
20 # img height
21 img_h = img.shape[0]
22 # img width
23 img_w = img.shape[1]
24 # kernel height // 2
25 a = kernel.shape[0] // 2
26 # kernel width // 2
27 b = kernel.shape[1] // 2
28
29 sigma = 60.0
30 normalizing_c = 1.0 / ( 2.0 * sigma * sigma * np.pi )
31
32 # building gauss kernel
33 for x in range(-a,a+1):
34     for y in range(-b,b+1):
35         dist = math.sqrt(x*x + y*y) * normalizing_c
36         val = math.exp( -dist ) * normalizing_c
37         kernel[a+x][b+y] = val
38
39 # empty op img
40 output = np.zeros((img_h,img_w), np.float32)
41
42 # conv
43 # visiting each pixel in the img
44 # m ta row img e ... for each row ...
45 for i in range(img_h):
46     # n ta coln img e ... for each coln ...
47     for j in range(img_w):
48         # sum of val to be calc
49         temp = 0
50         # visiting each pixel in the kernel
51         # a ta row img e ... for each row ...
52         for x in range(-a,a+1):
53             # b ta coln img e ... for each coln ...
54             for y in range(-b,b+1):
55                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
56                     temp += kernel[a+x][b+y] * img[i-x][j-y]
57                 else:
58                     temp += 0
59         temp = temp / (k_w*k_h)
60         output[i][j] = temp
61
62
63 def show_images(images, image_title):
64     # displaying multiple images side by side
65     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-using-matplotlib
66
67     # err : was giving weird colormap due to diff in the mechanism of reading img of
68     # cv2 & matplotlib
69     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
70     # matplotlib
71     # running this once in the code will ALWAYS give gray op
72     plt.gray()
73
74     no_of_imgs = len(images)
75     f = plt.figure()
76     for i in range(no_of_imgs):
```

```
75
76     # Debug, plot figure
77     axes = f.add_subplot(1, no_of_imgs, i + 1)
78     # the last img will show y axis on the RHS instead of LHS(which is by
79     # default)
80
81     if i==no_of_imgs-1:
82         axes.yaxis.tick_right()
83
84     plt.title(image_title[i])
85     plt.imshow(images[i], 'gray')
86     # plt.rc('font', size=8)
87     plt.show(block=True)
88
89 show_images([img,output], ['input', 'output'])
```

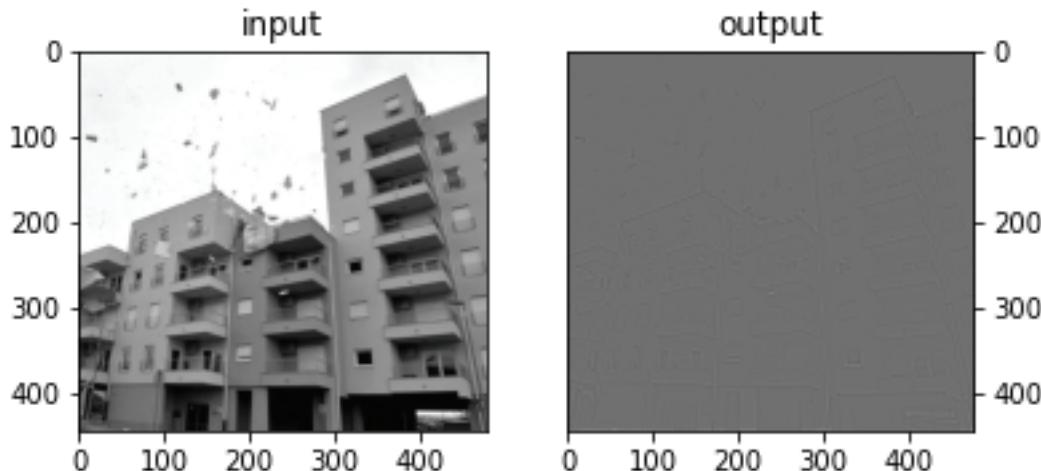


```

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/building.jpg'
8
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)
16
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)
21
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
30
31 pi=3.1416
32 sigma1 = 1.0
33 normalizing_c = 1.0 / ( 2.0 * sigma1 * sigma1 * pi )
34
35 # building kernel1
36 for x in range(-a,a+1):
37     for y in range(-b,b+1):
38         r = math.exp( -(x*x + y*y) / (2.0 * sigma1 * sigma1) )
39         r = r* normalizing_c
40         kernel1[a+x][b+y] = r
41
42 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):
48         r = math.exp( -(x*x + y*y) / (2.0 * sigma2 * sigma2) )
49         r = r* normalizing_c
50         kernel2[a+x][b+y] = r
51
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):
64         # visiting each pixel in the kernel
65         # a ta row img e ... for each row ...
66         for x in range(-a,a+1):
67             # b ta coln img e ... for each coln ...
68             for y in range(-b,b+1):
69                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
70                     output[i][j] += kernel2[a+x][b+y] * img[i-x][j-y]
71                 else:
72                     output[i][j] += 0
73
74
75
76 def show_images(images, image_title):
77     # displaying multiple images side by side

```

```
78     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
79     # using-matplotlib
80
80     # err : was giving weird colormap due to diff in the mechanism of reading img of
81     # cv2 & matplotlib
81     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
81     # 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)
91         # 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')
98         # plt.rc('font', size=8)
99         plt.show(block=True)
100
101 show_images([img,output], ['input', 'output'])
102
```



```

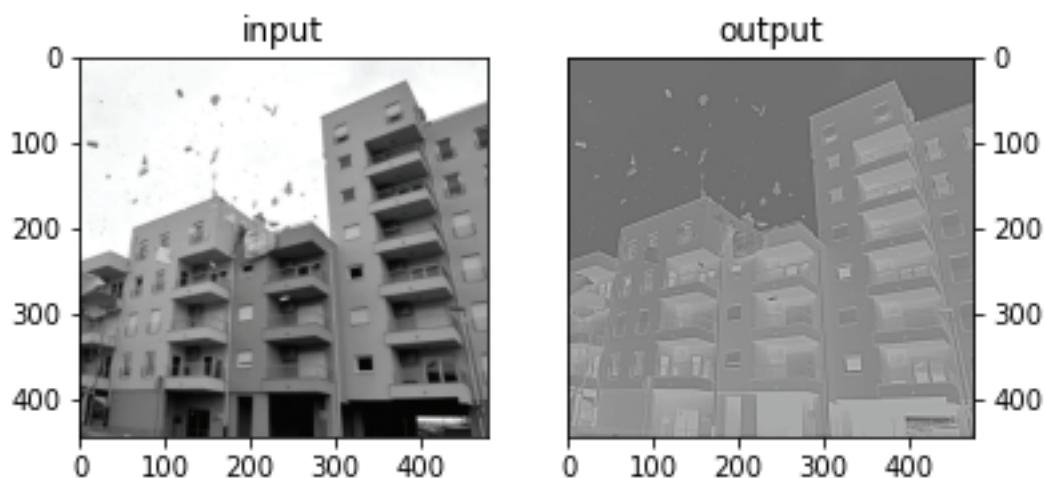
1 import cv2 as cv
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math
5
6 # path to the input img
7 path = 'C:/Users/Raiyan/Desktop/building.jpg'
8
9 # reading img + converting from BGR to GRAY
10 img = cv.imread(path)
11 img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
12
13 k_h = int(input("Enter kernel height: "))
14 k_w = k_h
15 k_size = (k_h,k_w)
16
17 # avg kernel
18 kernel = np.zeros( k_size, np.float32)
19
20 # img height
21 img_h = img.shape[0]
22 # img width
23 img_w = img.shape[1]
24 # kernel height // 2
25 a = kernel.shape[0] // 2
26 # kernel width // 2
27 b = kernel.shape[1] // 2
28
29 pi=3.1416
30 sigma = 1.0
31 normalizing_c = - 1.0 / ( sigma * sigma * sigma * sigma * pi )
32
33 # building kernel1
34 for x in range(-a,a+1):
35     for y in range(-b,b+1):
36         val1 = math.exp( -(x*x + y*y) / (2.0 * sigma * sigma) )
37         val2 = 1 - ((x*x + y*y) / (2.0 * sigma * sigma))
38         val1 = val1 * val2 * normalizing_c
39         kernel[a+x][b+y] = val1
40
41 # empty op img
42 output = np.zeros((img_h,img_w), np.float32)
43
44 # conv
45 # visiting each pixel in the img
46 # m ta row img e ... for each row ...
47 for i in range(img_h):
48     # n ta coln img e ... for each coln ...
49     for j in range(img_w):
50         # visiting each pixel in the kernel
51         # a ta row img e ... for each row ...
52         for x in range(-a,a+1):
53             # b ta coln img e ... for each coln ...
54             for y in range(-b,b+1):
55                 if 0 <= i-x < img_h and 0 <= j-y < img_w:
56                     output[i][j] += kernel[a+x][b+y] * img[i-x][j-y]
57                 else:
58                     output[i][j] += 0
59
60 out_conv = output
61 # scaled
62 def scaled(image):
63     g_m = image - image.min()
64     g_s = 255*(g_m / g_m.max())
65     return g_s.astype(np.float32)
66
67
68 # val capping or clipping from 0 - 255
69 for i in range(img_h):
70     for j in range(img_w):
71         if output[i][j] <0 :
72             output[i][j] = 0
73         elif output[i][j] >255 :
74             output[i][j] = 255
75
76 output = output.astype(np.float32)
77

```

```

78 clipped = output
79 scaled_output = scaled(out_conv)
80 img1 = img
81 #####
82 img = img.astype(np.float32)
83 img -= clipped
84
85 # val capping or clipping from 0 - 255
86 for i in range(img_h):
87     for j in range(img_w):
88         if img[i][j] < 0 :
89             img[i][j] = 0
90         elif img[i][j] >255 :
91             img[i][j] = 255
92
93 def show_images(images, image_title):
94     # displaying multiple images side by side
95     # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
96     # using-matplotlib
97     # err : was giving weird colormap due to diff in the mechanism of reading img of
98     # cv2 & matplotlib
99     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
100     # matplotlib
101     # running this once in the code will ALWAYS give gray op
102     plt.gray()
103
104     no_of_imgs = len(images)
105     f = plt.figure()
106     for i in range(no_of_imgs):
107
108         # Debug, plot figure
109         axes = f.add_subplot(1, no_of_imgs, i + 1)
110         # the last img will show y axis on the RHS instead of LHS(which is by
111         # default)
112
113         if i==no_of_imgs-1:
114             axes.yaxis.tick_right()
115
116         plt.title(image_title[i])
117         plt.imshow(images[i], 'gray')
118     plt.show(block=True)
119
120 show_images([img1,img], ['input', 'output'])
121

```



```

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
105|     # cv2 & matplotlib
106|     # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-matplotlib
107|     # running this once in the code will ALWAYS give gray op
108|     plt.gray()
109|
110|     no_of_imgs = len(images)
111|     f = plt.figure()
112|     for i in range(no_of_imgs):
113|
114|         # Debug, plot figure
115|         axes = f.add_subplot(1, no_of_imgs, i + 1)
116|         # the last img will show y axis on the RHS instead of LHS(which is by default)
117|
118|         if i==no_of_imgs-1:
119|             axes.yaxis.tick_right()
120|
121|         plt.title(image_title[i])
122|         plt.imshow(images[i])
123|         # plt.rc('font', size=8)
124|         plt.show(block=True)
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'])

```

