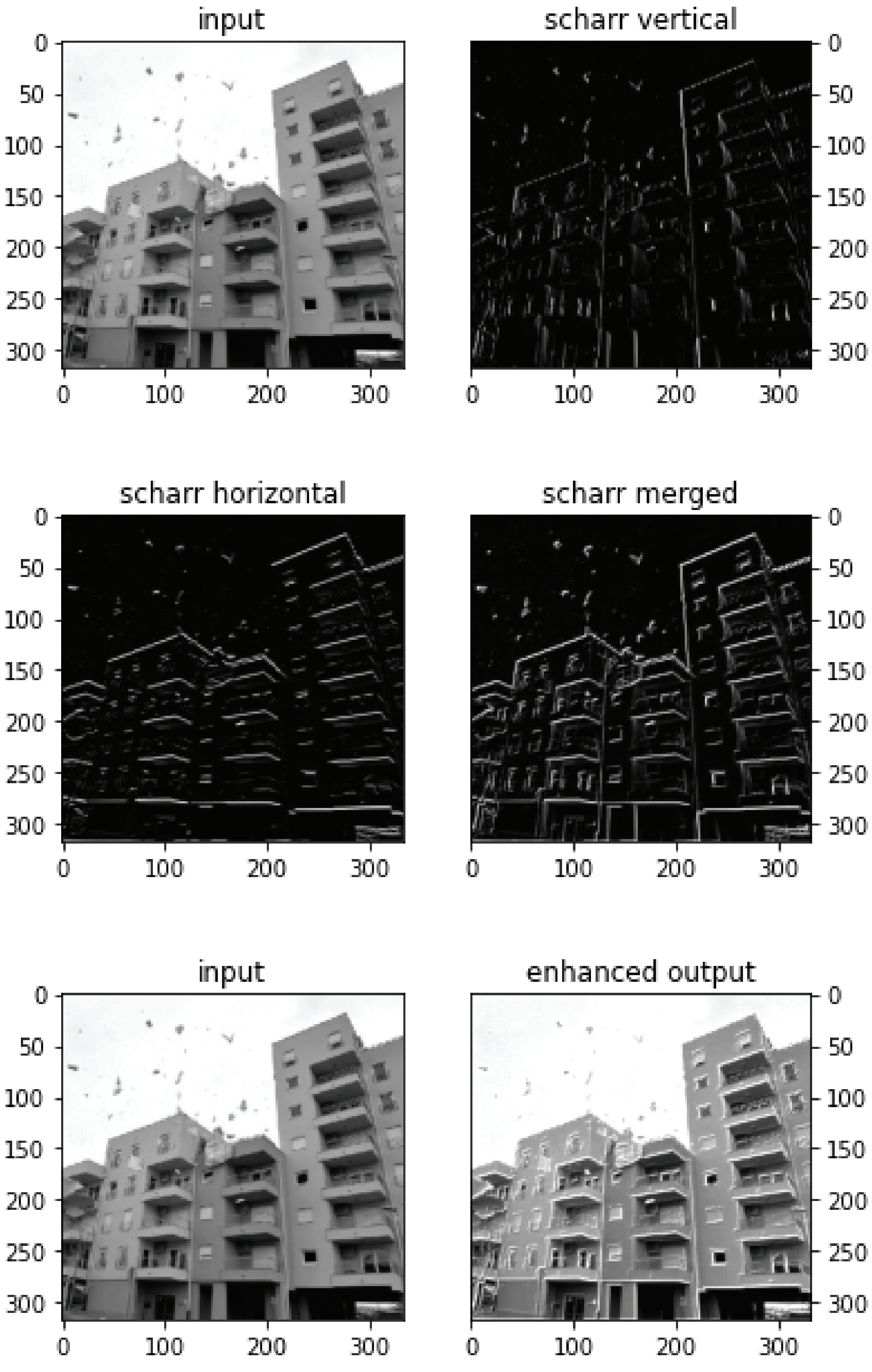
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# path to the input img
path = 'C:/Users/Raiyan/Desktop/building_332x317.jpg'

# reading img + converting from BGR to GRAY
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img1 = img

# scharr-y kernel
kernel = np.array(([-3,0,3],
                   [-10,0,10],
                   [-3,0,3]), np.float32)

k_h = kernel.shape[0]
k_w = k_h
k_size = (k_h,k_w)

# img height
img_h = img.shape[0]
# img width
img_w = img.shape[1]
# kernel height // 2
a = kernel.shape[0] // 2
# kernel width // 2
b = kernel.shape[1] // 2

# empty op img
output = np.zeros((img_h,img_w), np.float32)

# conv
# visiting each pixel in the img
# m ta row img e ... for each row ...
for i in range(img_h):
    # n ta coln img e ... for each coln ...
    for j in range(img_w):
        # sum of val to be calc
        calc = 0
        # visiting each pixel in the kernel
        # a ta row img e ... for each row ...
        for x in range(-a,a+1):
            # b ta coln img e ... for each coln ...
            for y in range(-b,b+1):
                if 0 <= i-x < img_h and 0 <= j-y < img_w:
                    calc += kernel[a+x][b+y] * img[i-x][j-y]
                else:
                    calc += 0
        calc = calc / (k_w*k_h)
        output[i][j] = calc

scharr_vertical = output
for i in range(img_h):
    for j in range(img_w):
            if scharr_vertical[i][j] > 255:
               scharr_vertical[i][j] = 255
            elif scharr_vertical[i][j] < 0:
                scharr_vertical[i][j] = 0

output = np.zeros((img_h,img_w), np.float32)

# scharr-x kernel
kernel = np.array(([-3,-10,-3],
                   [0,0,0],
                   [3,10,3]), np.float32)

# conv
# visiting each pixel in the img
# m ta row img e ... for each row ...
for i in range(img_h):
    # n ta coln img e ... for each coln ...
    for j in range(img_w):
        # sum of val to be calc
        calc = 0
        # visiting each pixel in the kernel
        # a ta row img e ... for each row ...
        for x in range(-a,a+1):
            # b ta coln img e ... for each coln ...
            for y in range(-b,b+1):
                if 0 <= i-x < img_h and 0 <= j-y < img_w:
                    calc += kernel[a+x][b+y] * img[i-x][j-y]
                else:
                    calc += 0
        calc = calc / (k_w*k_h)
        output[i][j] = calc

scharr_horizontal = output
for i in range(img_h):
    for j in range(img_w):
            if scharr_horizontal[i][j] > 255:
               scharr_horizontal[i][j] = 255
            elif scharr_horizontal[i][j] < 0:
                scharr_horizontal[i][j] = 0

scharr_merged = scharr_horizontal + scharr_vertical
for i in range(img_h):
    for j in range(img_w):
            if scharr_merged[i][j] > 255:
               scharr_merged[i][j] = 255
            elif scharr_merged[i][j] < 0:
                scharr_merged[i][j] = 0

img = img + scharr_merged

for i in range(img_h):
    for j in range(img_w):
            if img[i][j] > 255:
               img[i][j] = 255
            elif img[i][j] < 0:
                img[i][j] = 0


def show_images(images, image_title):
    # displaying multiple images side by side
    # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-using-matplotlib

    # err : was giving weird colormap due to diff in the mechanism of reading img of cv2 & matplotlib
    # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-matplotlib
    # running this once in the code will ALWAYS give gray op
    plt.gray()

    no_of_imgs = len(images)
    f = plt.figure()
    for i in range(no_of_imgs):

        # Debug, plot figure
        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 default)

        if i==no_of_imgs-1:
            axes.yaxis.tick_right()

        plt.title(image_title[i])
        plt.imshow(images[i], 'gray')
        # plt.rc('font', size=8)
    plt.show(block=True)

show_images([img1,scharr_vertical],
            ['input', 'scharr vertical'])
show_images([scharr_horizontal, scharr_merged],
            ['scharr horizontal', 'scharr merged'])
show_images([img1,img],
            ['input', 'enhanced output'])


```


input
scharr vertical
scharr horizontal
scharr merged
input
enhanced output
0
50
100
150
200
250
300
0
100
200
300