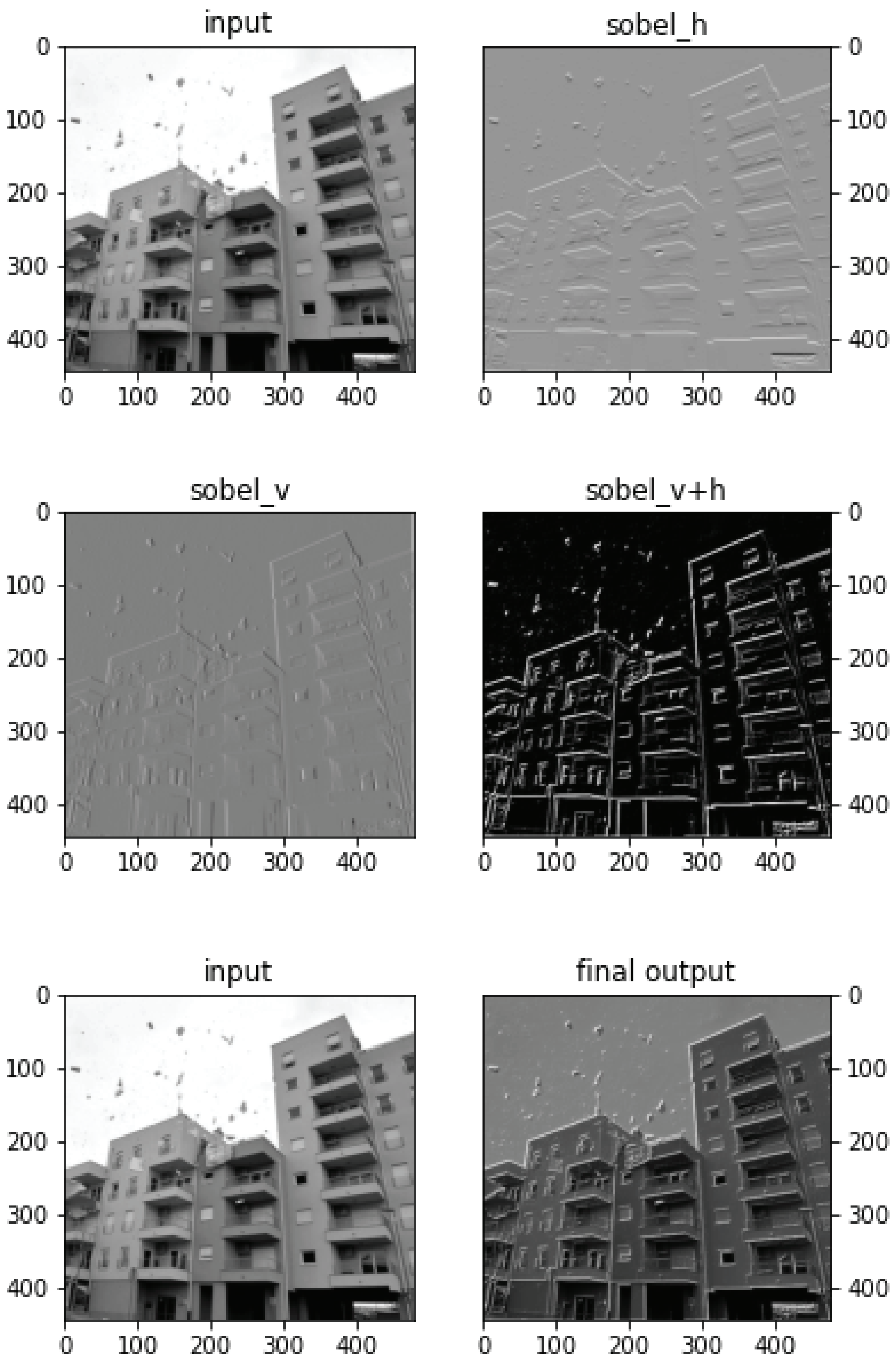
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math


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

# reading img + converting from BGR to GRAY
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
img1 = img
# sobel vertical
kernel_v = np.array(([-1,0,1],
                     [-2,0,2],
                     [-1,0,1]), np.float32)

# sobel horizontal
kernel_h = np.array(([-1,-2,-1],
                     [0,0,0],
                     [1,2,1]), np.float32)

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

k_h = kernel_v.shape[0]
k_w = k_h

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

def clipped_op(img):
    for i in range(img_h):
        for j in range(img_w):
            if result[i][j] > 255:
                result[i][j] = 255
            if result[i][j] < 0:
                result[i][j] = 0
    img = img.astype(np.float32)
    return img

# conv 1
# 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):
        # empty var for storing all the values
        values = []
        # 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:
                    output[i][j] += kernel_v[a+x][b+y] * img[i-x][j-y]
                else:
                    output[i][j] += 0

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

# conv 2
# 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):
        # empty var for storing all the values
        values = []
        # 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:
                    output[i][j] += kernel_h[a+x][b+y] * img[i-x][j-y]
                else:
                    output[i][j] += 0

output_h = output
result = output_v + output_h

result = clipped_op(result)

# plt.imshow(result, 'gray')
# plt.title("sobel_v+h")
# plt.show()

img = img.astype(np.float32)
img += result
img = clipped_op(img)

# plt.imshow(img, 'gray')
# plt.title("img+sobel")
# plt.show()

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])
        # plt.rc('font', size=8)
    plt.show(block=True)

show_images([img1,output_h], ['input', 'sobel_h'])
show_images([output_v,result], ['sobel_v', 'sobel_v+h'])
show_images([img1,img], ['input', 'final output'])



```


input
sobel_h
sobel_v
sobel_v+h
input
final output
0
100
200
300
400