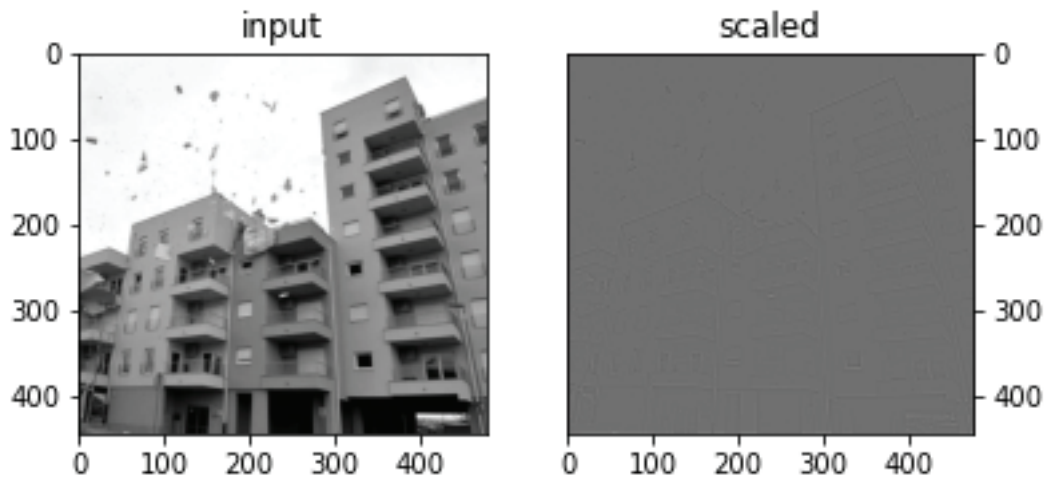
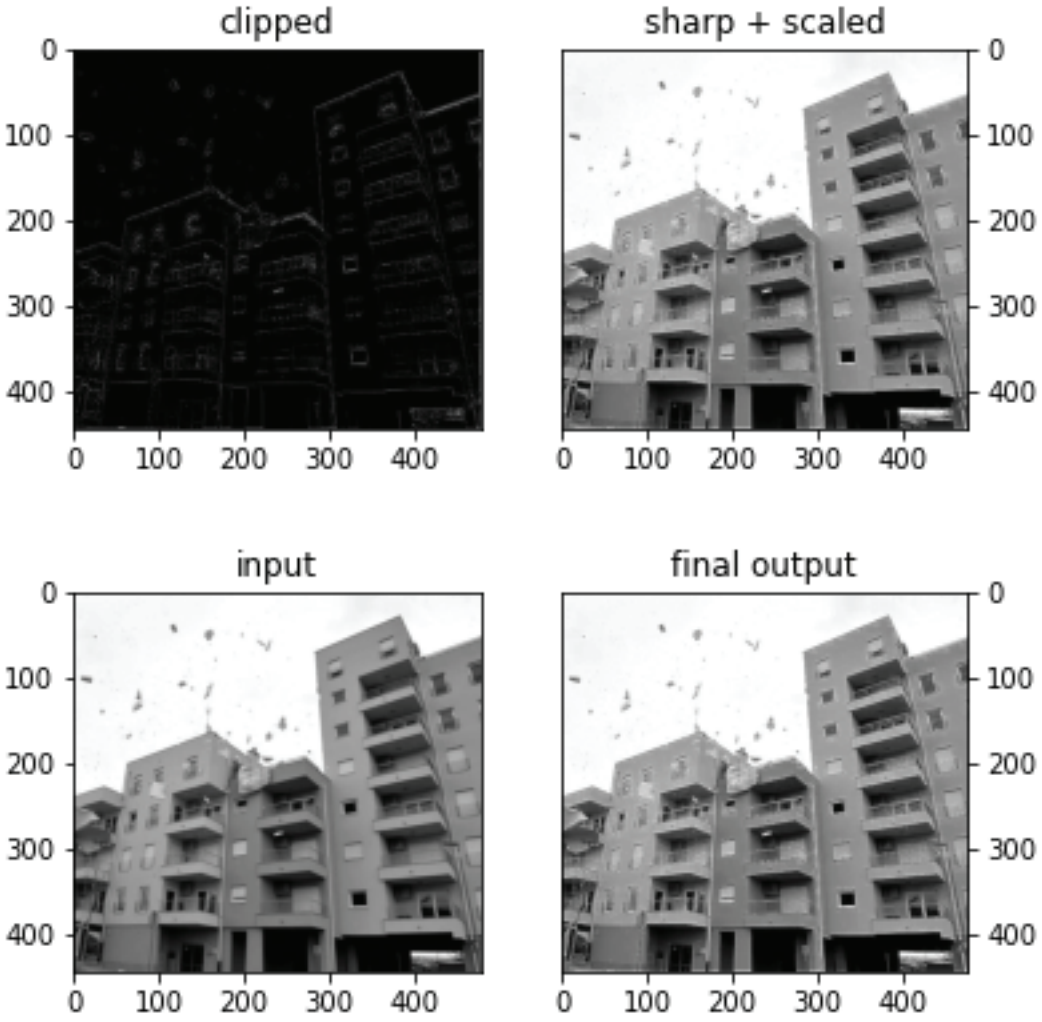
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# 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)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# kernel with neg center value
kernel = np.array(([0,-1,0],
                   [-1,+4,-1],
                   [0,-1,0]), np.float32)

# 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):
        # 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[a+x][b+y] * img[i-x][j-y]
                else:
                    output[i][j] += 0

out_conv = output

# scaled
def scaled(image):
    g_m = image - image.min()
    g_s = 255*(g_m / g_m.max())
    return g_s.astype(np.float32)

scaled = scaled(out_conv)

plt.imshow(scaled, 'gray')
plt.title('scaled')
plt.show()

output = out_conv

# val capping or clipping from 0 - 255
for i in range(img_h):
      for j in range(img_w):
          if output[i][j] <0 :
              output[i][j] = 0
          elif output[i][j] >255 :
              output[i][j] = 255

clipped = output.astype(np.float32)

```

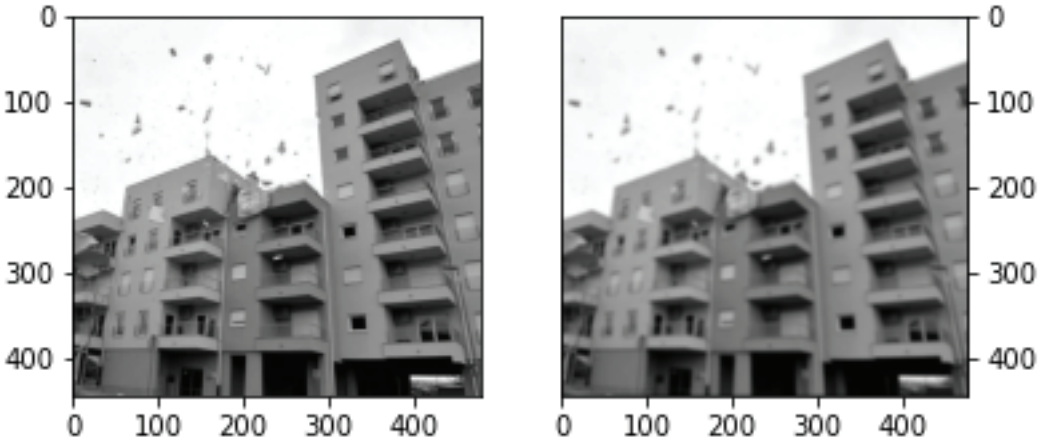
```
# center of kernel is (+)
sharpened = img + out_conv

output = sharpened
# sharpened + clipping from 0 - 255
for i in range(img_h):
      for j in range(img_w):
          if output[i][j] <0 :
              output[i][j] = 0
          elif output[i][j] >255 :
              output[i][j] = 255

sharpened_and_clipped = output.astype(np.float32)

# sharpened + scaled
g_m = sharpened - sharpened.min()
g_s = (g_m / g_m.max()) * 255
sharpened_and_scaled = g_s.astype(np.float32)


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([img,scaled], ['input', 'scaled'])
show_images([clipped, sharpened_and_scaled], ['clipped', 'sharp + scaled'])
show_images([img,sharpened_and_scaled], ['input', 'final output'])
```




clipped
sharp + scaled
input
final output
0
100
200
300
400

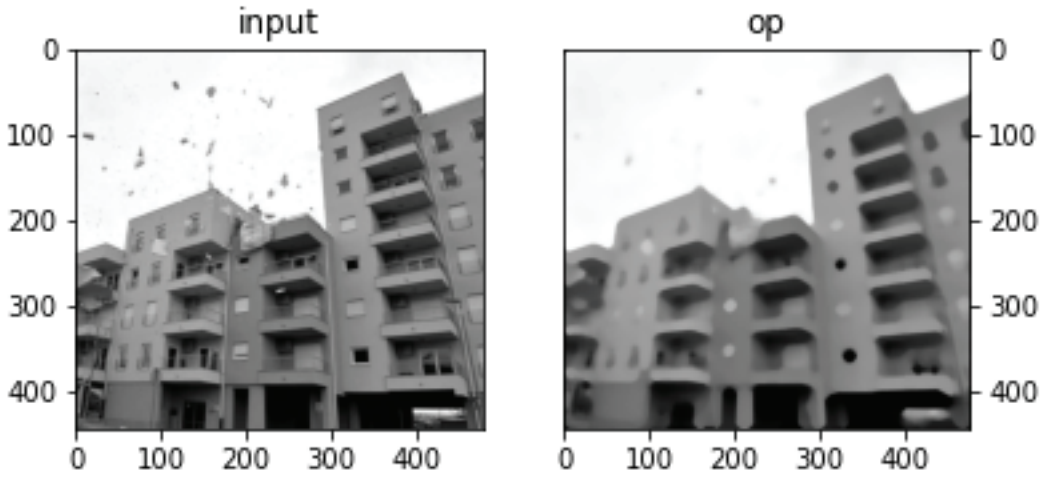
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

def show_images(images):
    # 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()

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

        # Debug, plot figure
        axes = f.add_subplot(1, n, i + 1)
        # the last img will show y axis on the RHS instead of LHS(which is by default)
        if i==n-1:
            axes.yaxis.tick_right()
        plt.imshow(images[i])

    plt.show(block=True)

# path to the input img
# path = "C:/Users/Raiyan/Desktop/img/03/Image-Processing-and-Computer-Vision-Lab/Lab 2/Average filter/Input.png"
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)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# avg kernel
kernel = np.ones( k_size, np.float32)

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

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

# sum of the values of the kernel
k_sum = kernel.sum()
# print(f'ksum is {ksum}')

# 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 calculating a summed value
        value = 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:
                    value = value + kernel[a+x][b+y] * img[i+x][j+y]
                else:
                    value = value + 0
        value = value / k_sum
        output[i][j] = value

show_images([img,output])


```



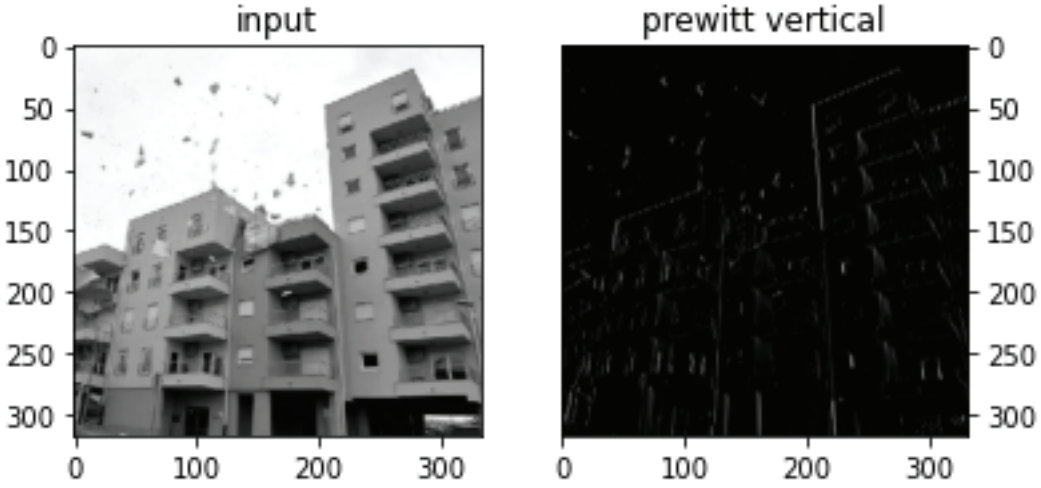
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# path to the input img
# path = "C:/Users/Raiyan/Desktop/img/03/Image-Processing-and-Computer-Vision-Lab/Lab 2/Average filter/Input.png"
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)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# avg kernel
kernel = np.ones( k_size, np.float32)

# 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)

# sum of the values of the kernel
k_sum = kernel.sum()
# print(f'ksum is {ksum}')


# 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:
                    calculated_val = kernel[a+x][b+y] * img[i+x][j+y]
                    values.append( calculated_val )
                else:
                    values.append(0)
        values.sort()

        median_val = len(values) // 2

        output[i][j] = values[median_val]
        output[i][j] /= k_sum


def show_images(images):
    # 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()
```

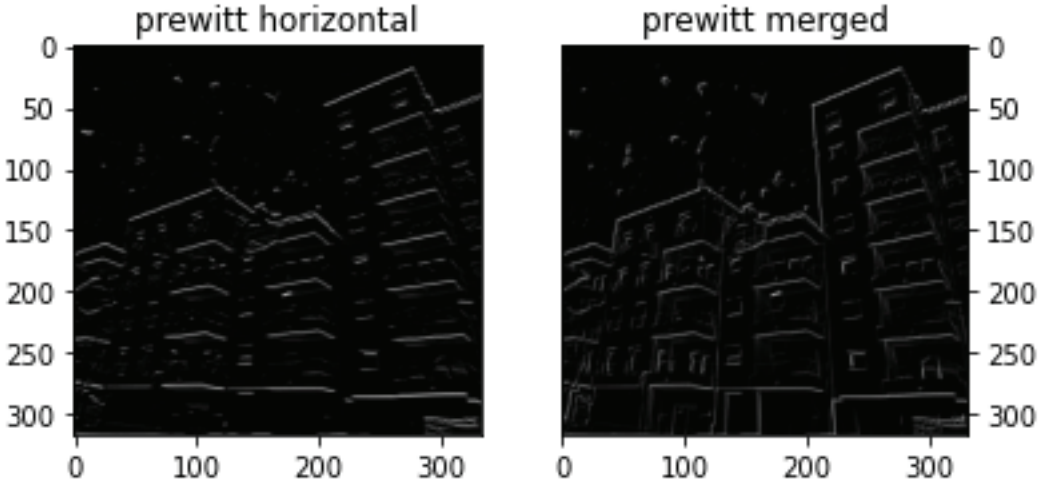
```
74      for i in range(no_of_imgs):
75
76          # Debug, plot figure
77          axes = f.add_subplot(1, no_of_imgs, i + 1)
78          # the last img will show y axis on the RHS instead of LHS(which is by
    default)
79
80          if i==no_of_imgs-1:
81              axes.yaxis.tick_right()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

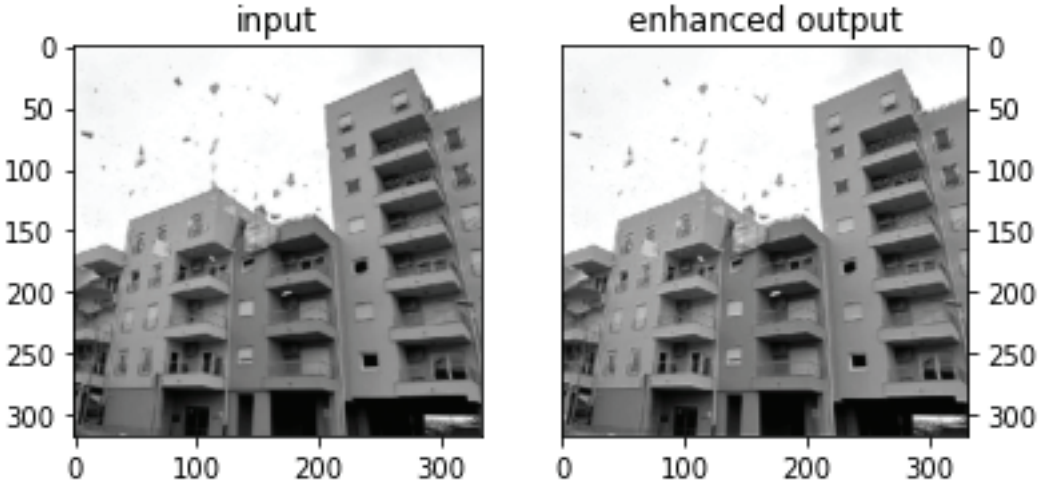


```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# 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

# prewitt-y kernel
kernel = np.array(([-1,0,1],
                   [-1,0,1],
                   [-1,0,1]), 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

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

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

# prewitt-x kernel
kernel = np.array(([-1,-1,-1],
                   [0,0,0],
                   [1,1,1]), 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

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

prewitt_merged = prewitt_horizontal + prewitt_vertical
for i in range(img_h):
    for j in range(img_w):
            if prewitt_merged[i][j] > 255:
               prewitt_merged[i][j] = 255
            elif prewitt_merged[i][j] < 0:
                prewitt_merged[i][j] = 0

img = img + prewitt_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,prewitt_vertical],
            ['input', 'prewitt vertical'])
show_images([prewitt_horizontal, prewitt_merged],
            ['prewitt horizontal', 'prewitt merged'])
show_images([img1,img],
            ['input', 'enhanced output'])
```

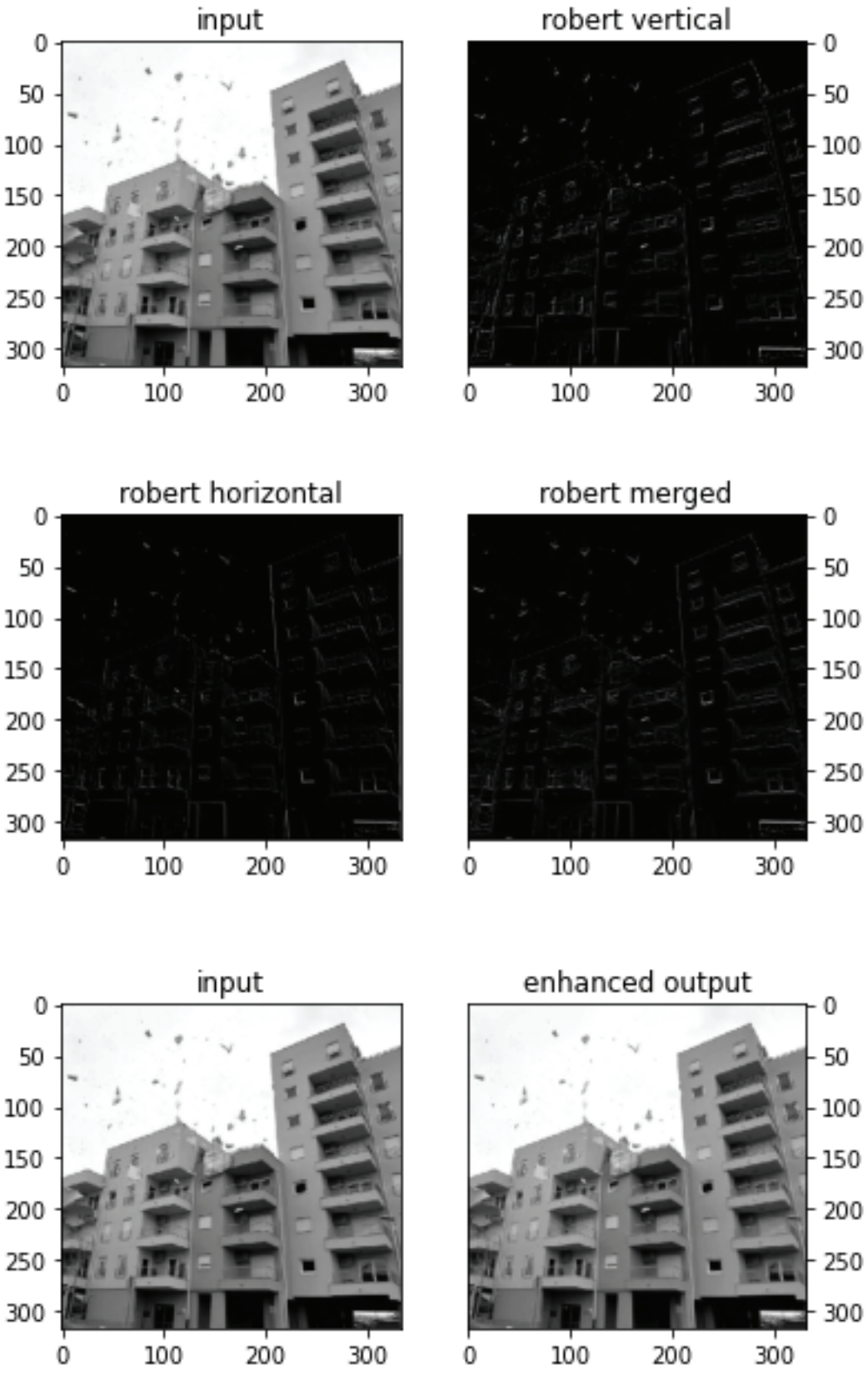

input
prewitt vertical
0
50
100
150
200
250
300
0
100
200
300


prewitt horizontal
prewitt merged
0
50
100
150
200
250
300
0
100
200
300


input
enhanced output
0
50
100
150
200
250
300
0
100
200
300

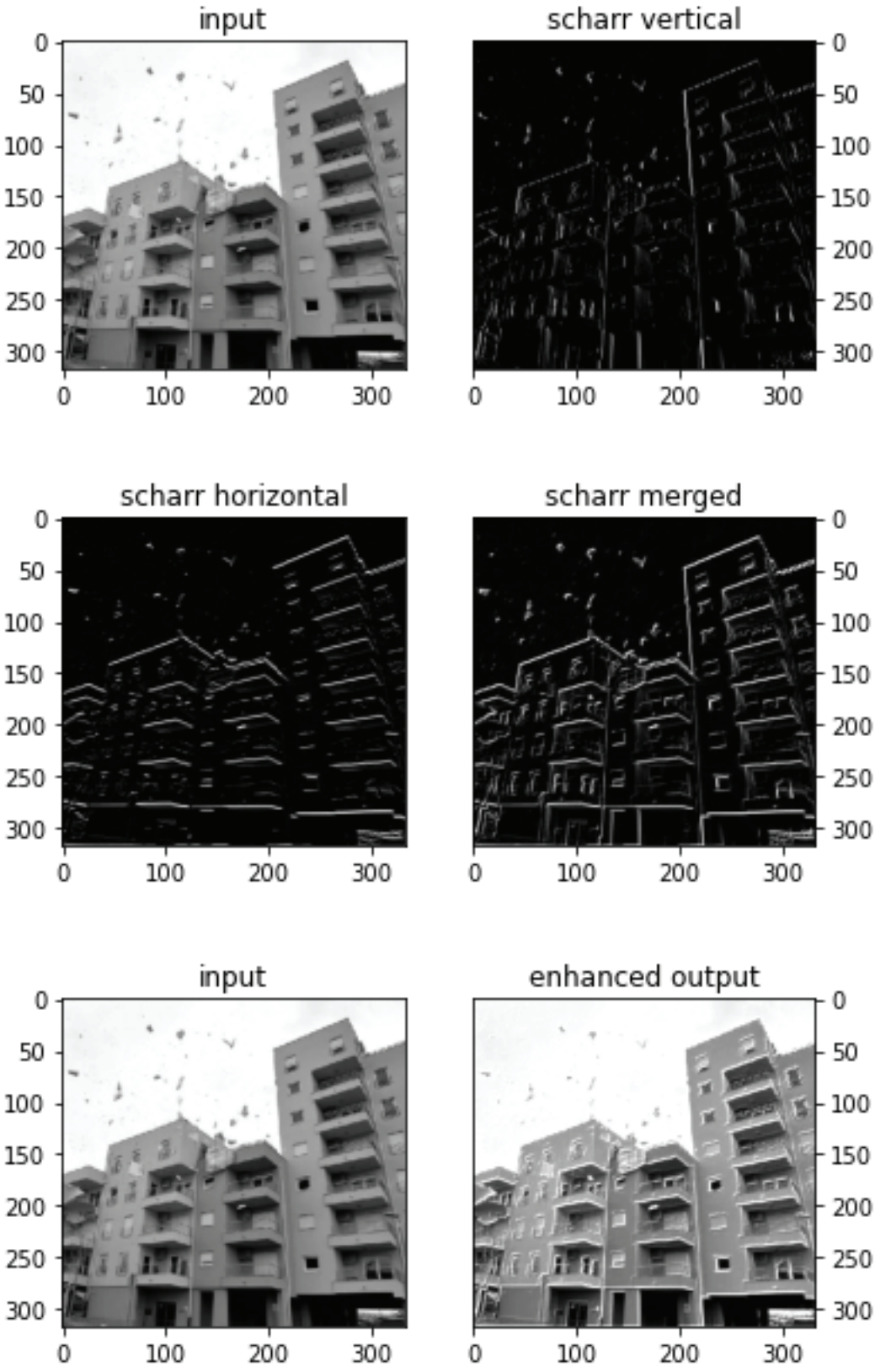
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# 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

# robert-x kernel
kernel = np.array(([1,0],
                   [0,-1]), 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:
                    if 0<= a+x < k_h and 0<= b+y < k_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

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

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

# robert-y kernel
kernel = np.array(([0,1],
                   [-1,0]), 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:
                    if 0<= a+x < k_h and 0<= b+y < k_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

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

robert_merged = robert_horizontal + robert_vertical
for i in range(img_h):
    for j in range(img_w):
            if robert_merged[i][j] > 255:
               robert_merged[i][j] = 255
            elif robert_merged[i][j] < 0:
                robert_merged[i][j] = 0

img = img + robert_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,robert_vertical],
            ['input', 'robert vertical'])
show_images([robert_horizontal, robert_merged],
            ['robert horizontal', 'robert merged'])
show_images([img1,img],
            ['input', 'enhanced output'])


```


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

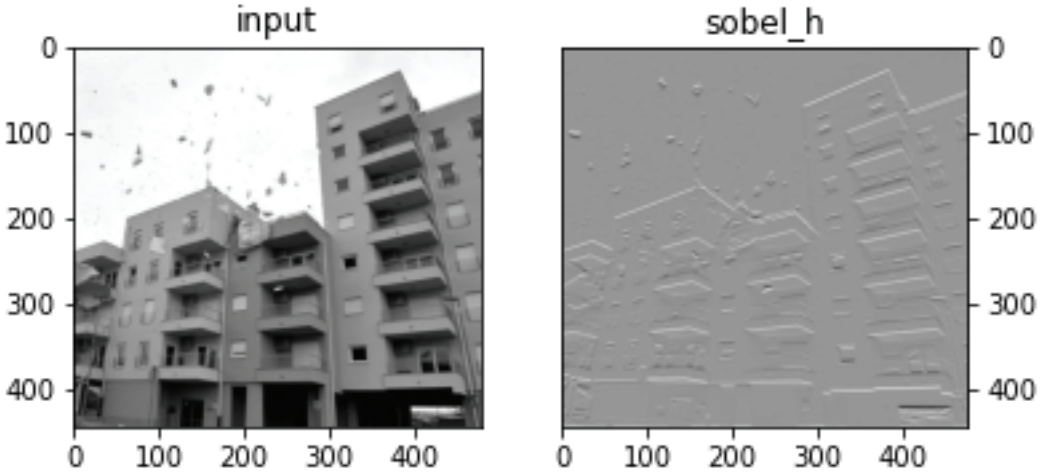
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# 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
```

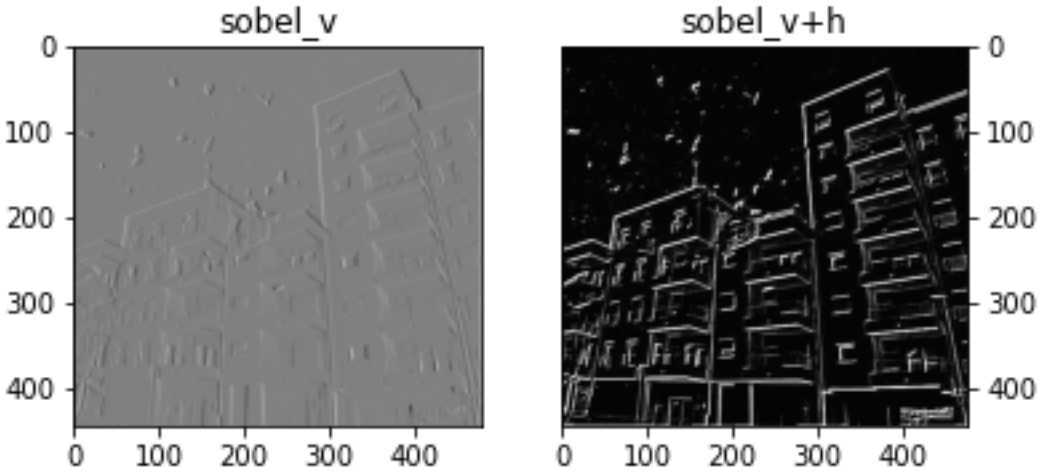
```
        # visiting each pixel in the kernel
        # a ta row img e ... for each row ...
        for x in range(-a,a+1):
            # b ta coln img e ... for each coln ...
            for y in range(-b,b+1):
                if 0 <= i-x < img_h and 0 <= j-y < img_w:
                    calc += kernel[a+x][b+y] * 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'])
```

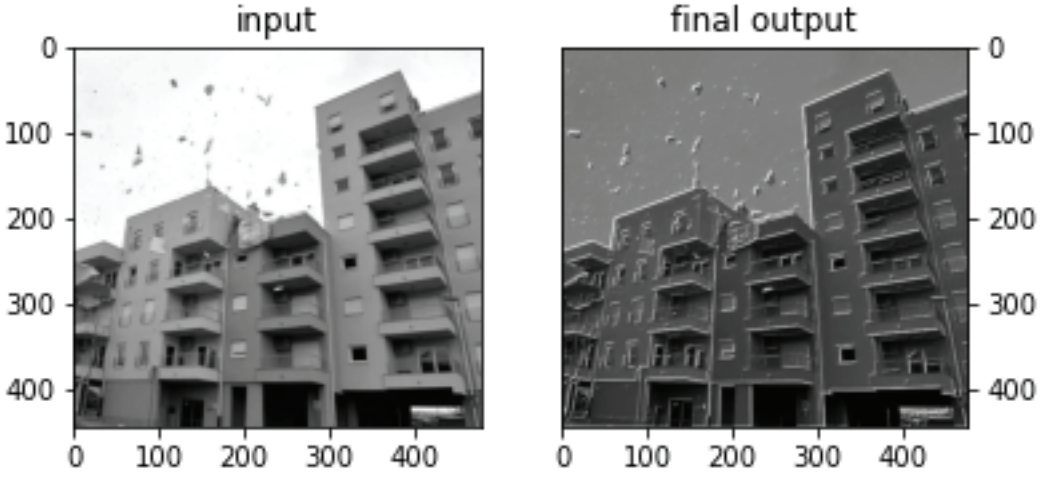

input
scharr vertical
scharr horizontal
scharr merged
input
enhanced output
0
50
100
150
200
250
300

```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'])
```

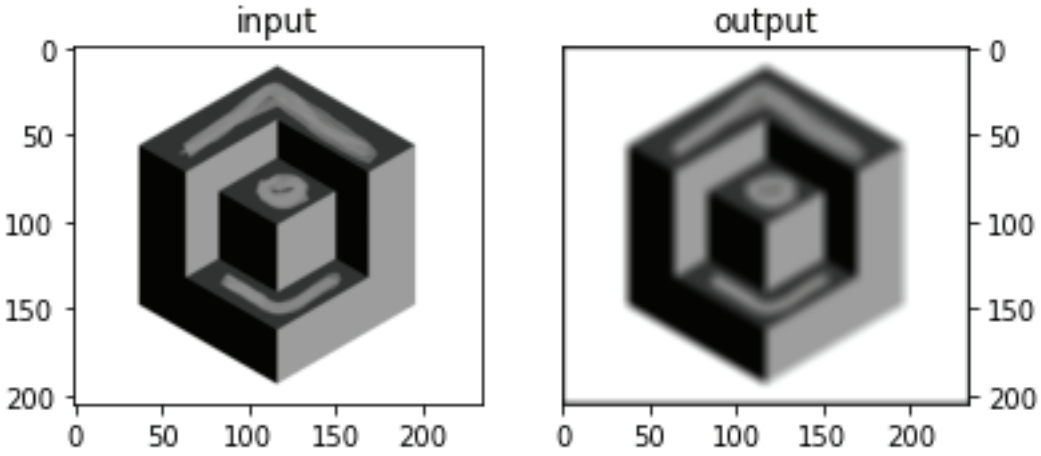

input
sobel_h
0
100
200
300
400


sobel_v
sobel_v+h
0
100
200
300
400


input
final output
0
100
200
300
400

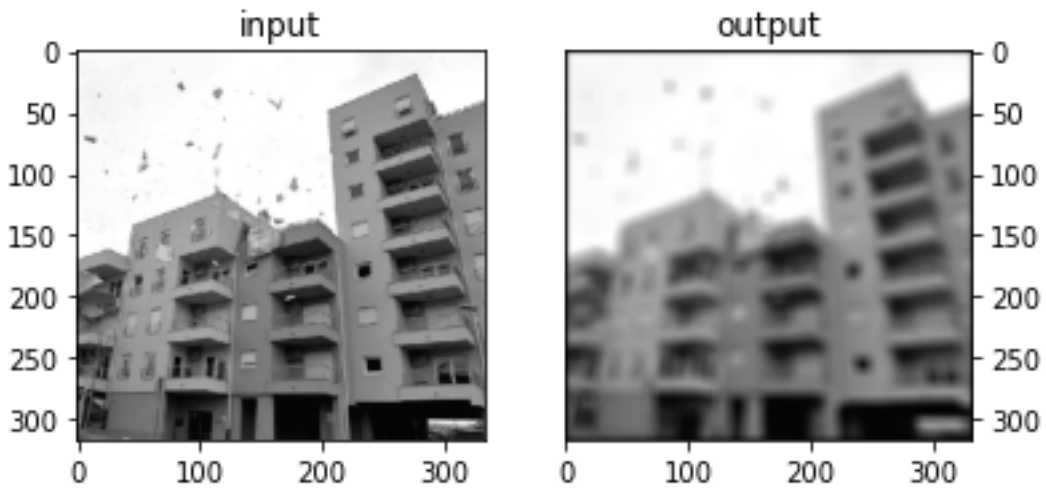
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math
minval = 0
maxval = 255

# path to the input img
path = 'C:/Users/Raiyan/Desktop/cube.png'

# reading img + converting from BGR to GRAY
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
# resizing image
img = cv.resize(img, (int(820/3.5),int(720/3.5)), interpolation = cv.INTER_AREA)
img = img/maxval
img1 = img

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# empty kernel
kernel = np.zeros( k_size, np.float32)

# 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

sigma = 60.0
normalizing_c = 1.0 / ( 2.0 * sigma * sigma  )

# building kernel
for x in range(-a,a+1):
    for y in range(-b,b+1):
        dist = math.sqrt(x*x + y*y) * normalizing_c
        val = math.exp( -dist ) / ( np.pi * 2.0 * sigma * sigma )
        kernel[a+x][b+y] = val

# 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
        # empty kernel for each iter
        loop_ker = np.zeros( k_size, np.float32)
        # 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:
                    dist = math.sqrt( np.power( img[i][j] - img[i-x][j-y], 2 ) ) *
normalizing_c
                    val = math.exp( -dist ) / ( np.pi * 2.0 * sigma * sigma )
                    loop_ker[a+x][b+y] = kernel[a+x][b+y] * val

        for x in range(-a,a+1):
            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 / ( loop_ker.shape[0] * loop_ker.shape[1] )
        output[i][j] = calc
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
```



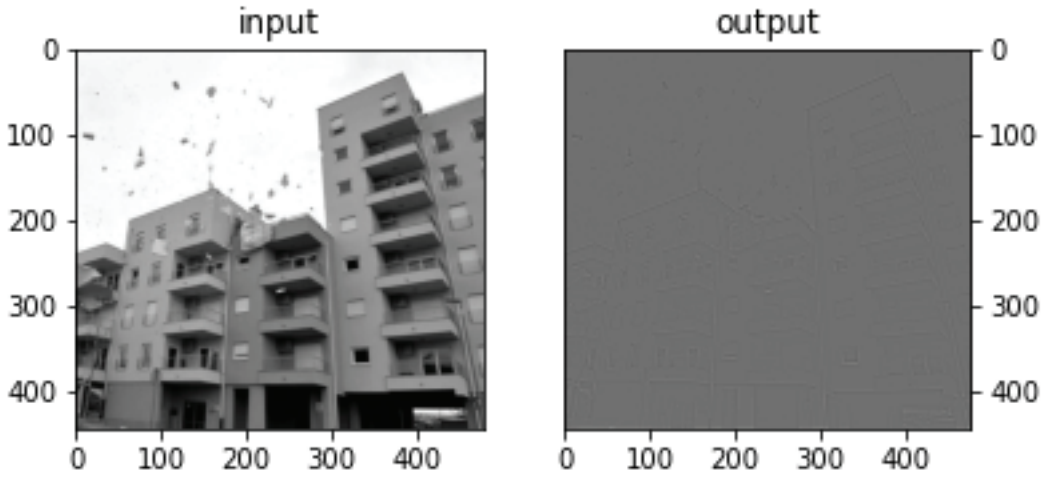
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# 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)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# empty kernel
kernel = np.zeros( k_size, np.float32)

# 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

sigma = 60.0
normalizing_c = 1.0 / ( 2.0 * sigma * sigma * np.pi )

# building gauss kernel
for x in range(-a,a+1):
    for y in range(-b,b+1):
        dist = math.sqrt(x*x + y*y) * normalizing_c
        val = math.exp( -dist ) * normalizing_c
        kernel[a+x][b+y] = val

# 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
        temp = 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:
                    temp += kernel[a+x][b+y] * img[i-x][j-y]
                else:
                    temp += 0
        temp = temp / (k_w*k_h)
        output[i][j] = temp


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):
```

```
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
    default)
79
80          if i==no_of_imgs-1:
81              axes.yaxis.tick_right()
82
83          plt.title(image_title[i])
84          plt.imshow(images[i], 'gray')
85          # plt.rc('font', size=8)
86      plt.show(block=True)
87
88  show_images([img,output], ['input', 'output'])
89
```



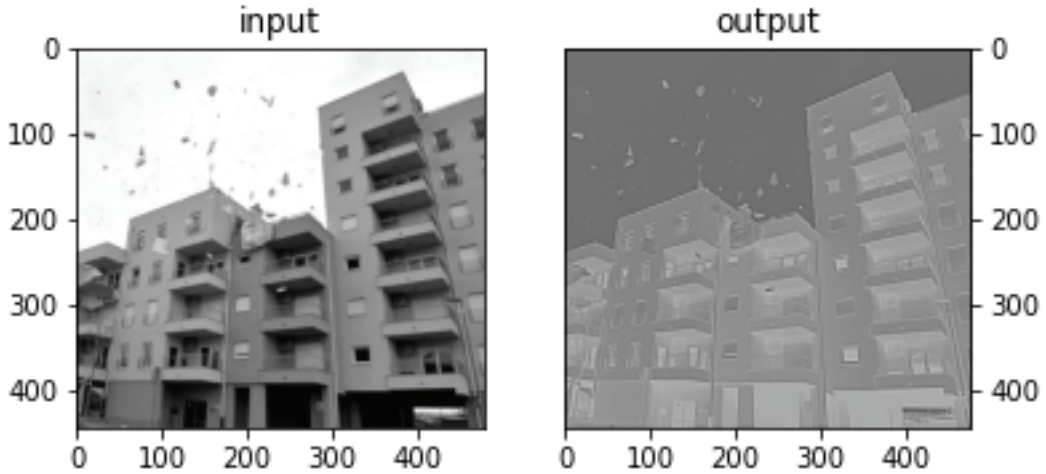
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# 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)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# avg kernel
kernel1 = np.zeros( k_size, np.float32)
kernel2 = np.zeros( k_size, np.float32)
kernel3 = np.zeros( k_size, np.float32)

# img height
img_h = img.shape[0]
# img width
img_w = img.shape[1]
# kernel height // 2
a = kernel1.shape[0] // 2
# kernel width // 2
b = kernel1.shape[1] // 2

pi=3.1416
sigma1 = 1.0
normalizing_c = 1.0 / ( 2.0 * sigma1 * sigma1 * pi )

# building kernel1
for x in range(-a,a+1):
    for y in range(-b,b+1):
        r = math.exp( -(x*x + y*y) / (2.0 * sigma1 * sigma1) )
        r = r* normalizing_c
        kernel1[a+x][b+y] = r

sigma2 = 2.5
normalizing_c = (1.0 / ( 2.0 * sigma2 * sigma2 * pi ))

# building kernel2
for x in range(-a,a+1):
    for y in range(-b,b+1):
        r = math.exp( -(x*x + y*y) / (2.0 * sigma2 * sigma2) )
        r = r* normalizing_c
        kernel2[a+x][b+y] = r

# subtracting kernel1 from kernel2
kernel2 = kernel2 - kernel1

# 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):
        # 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] += kernel2[a+x][b+y] * img[i-x][j-y]
                else:
                    output[i][j] += 0



def show_images(images, image_title):
    # displaying multiple images side by side
```

```
78      # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-
    using-matplotlib
79
80      # err : was giving weird colormap due to diff in the mechanism of reading img of
    cv2 & matplotlib
81      # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-
    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
    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
```



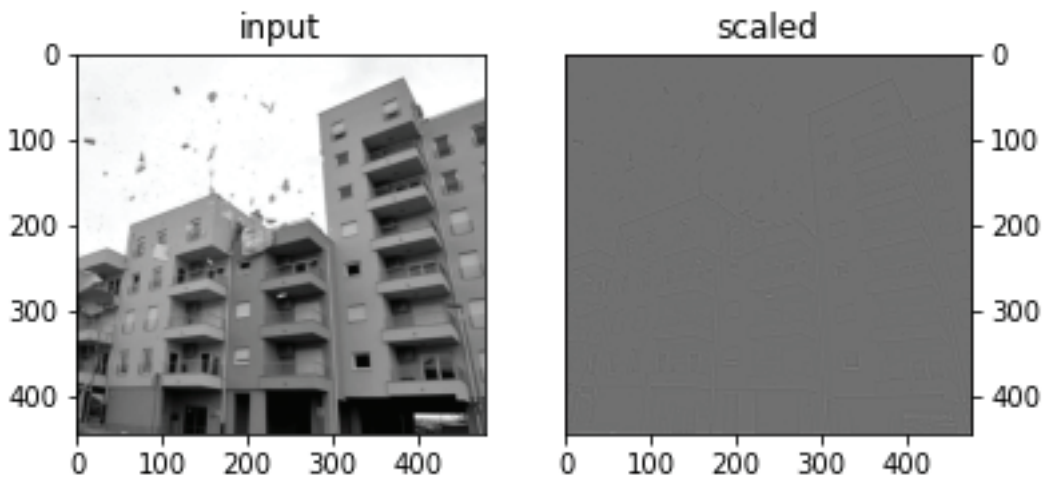
```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# 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)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# avg kernel
kernel = np.zeros( k_size, np.float32)

# 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

pi=3.1416
sigma = 1.0
normalizing_c = - 1.0 / ( sigma * sigma * sigma * sigma * pi )

# building kernel1
for x in range(-a,a+1):
    for y in range(-b,b+1):
        val1 = math.exp( -(x*x + y*y) / (2.0 * sigma * sigma) )
        val2 = 1 - ((x*x + y*y) / (2.0 * sigma * sigma))
        val1 = val1 * val2 * normalizing_c
        kernel[a+x][b+y] = val1

# 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):
        # 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[a+x][b+y] * img[i-x][j-y]
                else:
                    output[i][j] += 0

out_conv = output
# scaled
def scaled(image):
    g_m = image - image.min()
    g_s = 255*(g_m / g_m.max())
    return g_s.astype(np.float32)


# val capping or clipping from 0 - 255
for i in range(img_h):
      for j in range(img_w):
          if output[i][j] <0 :
              output[i][j] = 0
          elif output[i][j] >255 :
              output[i][j] = 255

output = output.astype(np.float32)

```

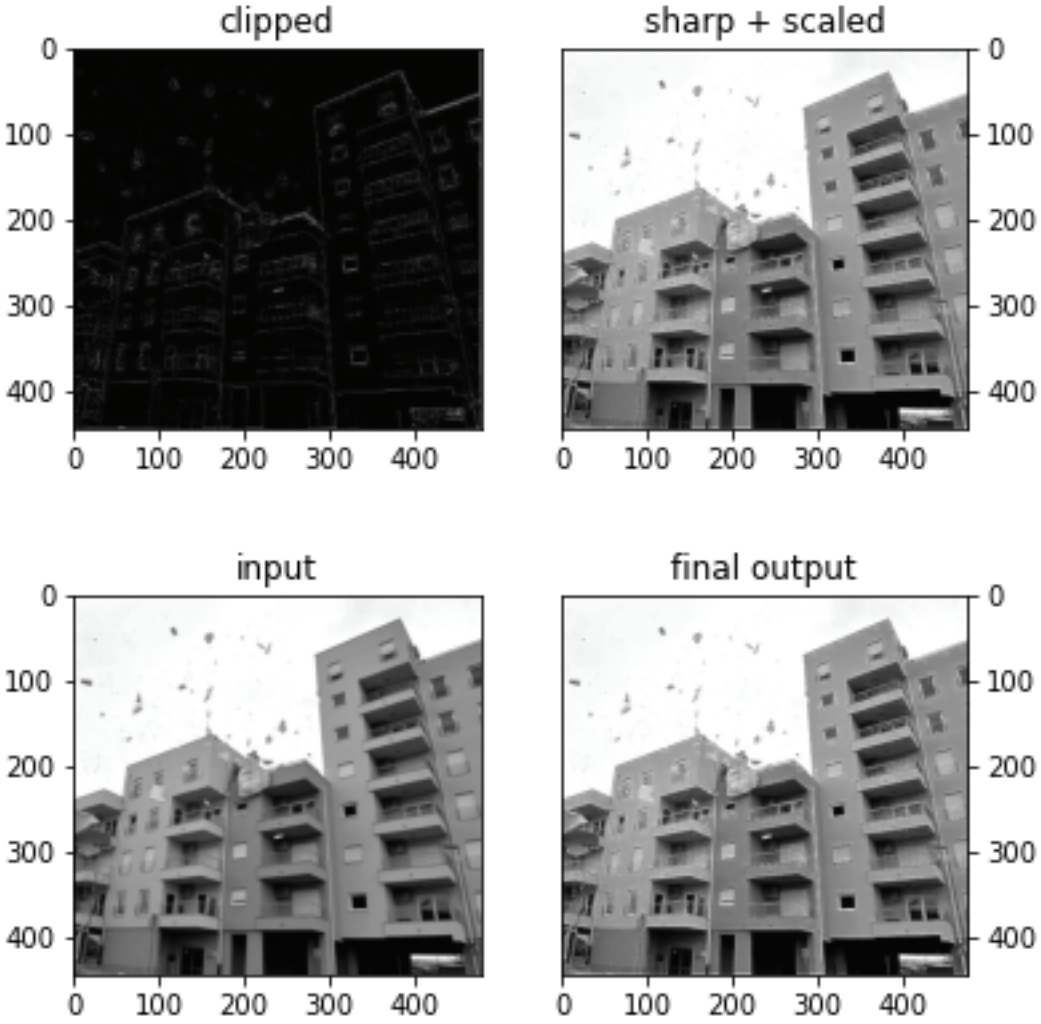
```
clipped = output
scaled_output = scaled(out_conv)
img1 = img
##########
img = img.astype(np.float32)
img -= clipped

# val capping or clipping from 0 - 255
for i in range(img_h):
      for j in range(img_w):
          if img[i][j] <0 :
              img[i][j] = 0
          elif img[i][j] >255 :
              img[i][j] = 255

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,img], ['input', 'output'])

```



```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/img/03/Image-Processing-and-Computer-Vision-Lab/Lab 2/Average filter/Input.png"
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)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# kernel with neg center value
kernel = np.array(([0,-1,0],
                   [-1,-4,-1],
                   [0,-1,0]), np.float32)

# 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):
        # 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[a+x][b+y] * img[i-x][j-y]
                else:
                    output[i][j] += 0

out_conv = output

# scaled
def scaled(image):
    g_m = image - image.min()
    g_s = 255*(g_m / g_m.max())
    return g_s.astype(np.float32)

scaled = scaled(out_conv)

plt.imshow(scaled, 'gray')
plt.title('scaled')
plt.show()

output = out_conv

# val capping or clipping from 0 - 255
for i in range(img_h):
      for j in range(img_w):
          if output[i][j] <0 :
              output[i][j] = 0
          elif output[i][j] >255 :
              output[i][j] = 255

```

```
clipped = output.astype(np.float32)


# center of kernel is (-)
sharpened = img - out_conv

output = sharpened
# sharpened + clipping from 0 - 255
for i in range(img_h):
      for j in range(img_w):
          if output[i][j] <0 :
              output[i][j] = 0
          elif output[i][j] >255 :
              output[i][j] = 255

sharpened_and_clipped = output.astype(np.float32)

# sharpened + scaled
g_m = sharpened - sharpened.min()
g_s = (g_m / g_m.max()) * 255
sharpened_and_scaled = g_s.astype(np.float32)


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([img,scaled], ['input', 'scaled'])
show_images([clipped, sharpened_and_scaled], ['clipped', 'sharp + scaled'])
show_images([img,sharpened_and_scaled], ['input', 'final output'])
```




clipped
sharp + scaled
input
final output
0
100
200
300
400