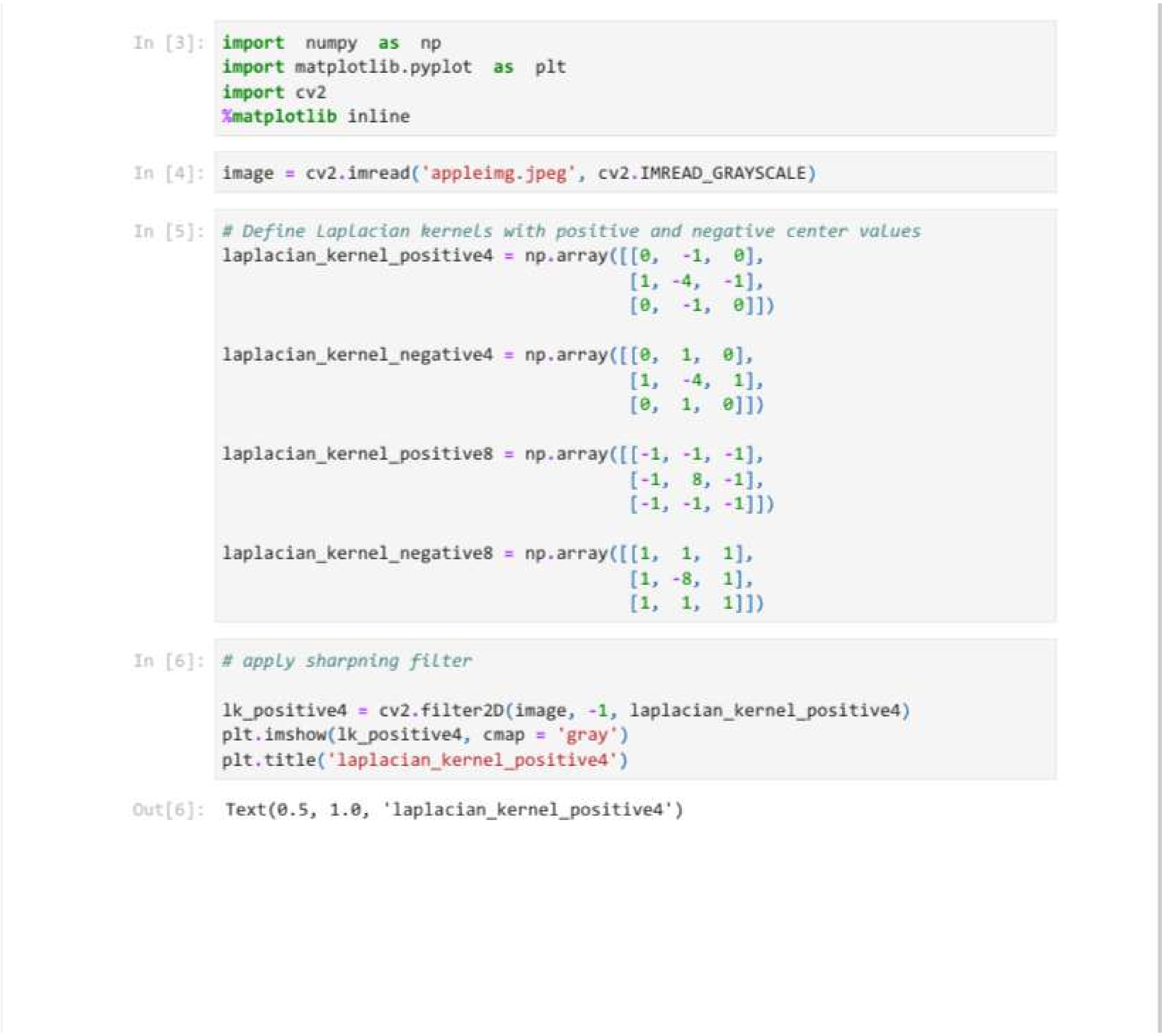


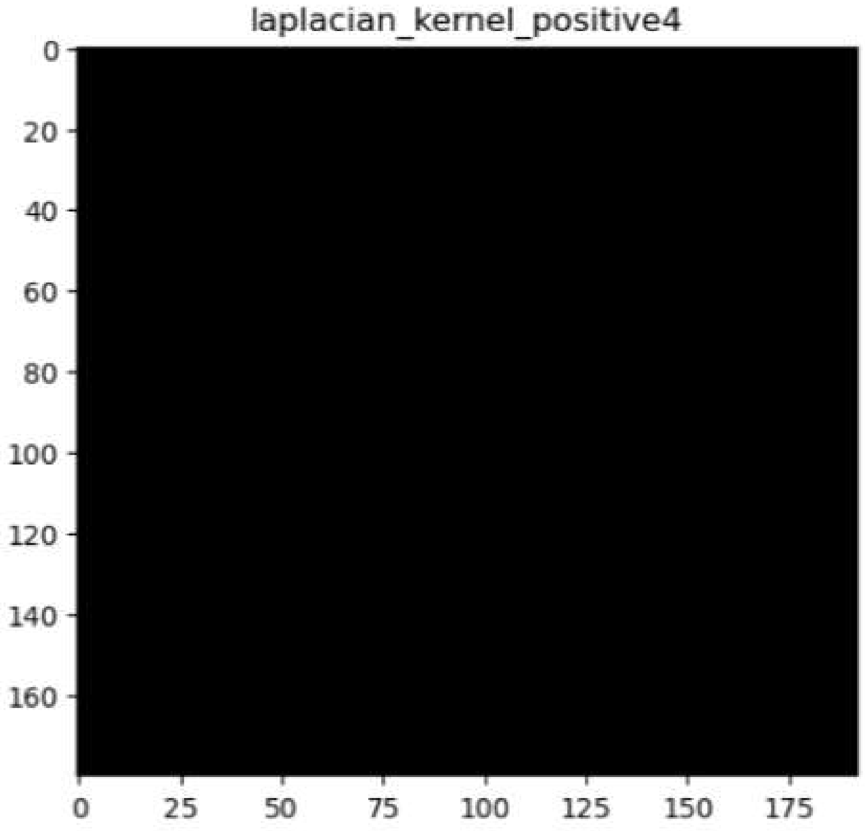
```
In [3]: import  numpy  as  np
        import matplotlib.pyplot  as  plt
        import cv2
        %matplotlib inline
```

```
In [4]: image = cv2.imread('appleimg.jpeg', cv2.IMREAD_GRAYSCALE)
```

```
In [5]: # Define Laplacian kernels with positive and negative center values
        laplacian_kernel_positive4 = np.array([[0,  -1,  0],
                                               [1, -4,  -1],
                                               [0,  -1,  0]])

        laplacian_kernel_negative4 = np.array([[0,  1,  0],
                                               [1,  -4,  1],
                                               [0,  1,  0]])

        laplacian_kernel_positive8 = np.array([[-1, -1, -1],
                                               [-1,  8, -1],
                                               [-1, -1, -1]])

        laplacian_kernel_negative8 = np.array([[1,  1,  1],
                                               [1, -8,  1],
                                               [1,  1,  1]])
```

```
In [6]: # apply sharpning filter

        lk_positive4 = cv2.filter2D(image, -1, laplacian_kernel_positive4)
        plt.imshow(lk_positive4, cmap = 'gray')
        plt.title('laplacian_kernel_positive4')
```

```
Out[6]: Text(0.5, 1.0, 'laplacian_kernel_positive4')
```

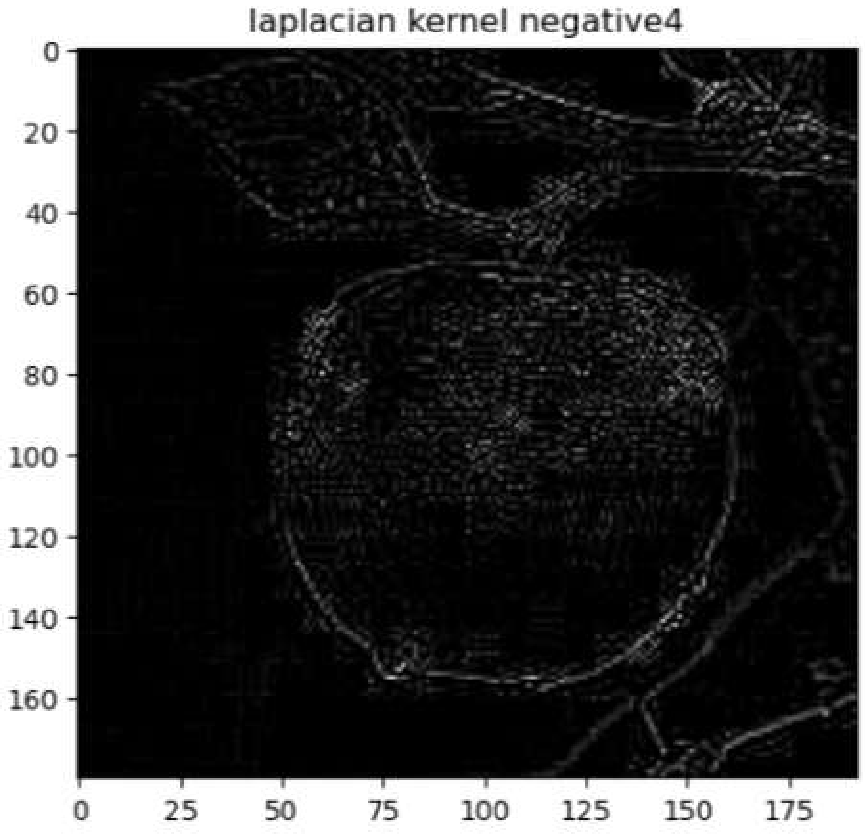


In [7]:

```
lk_positive4 = cv2.filter2D(image, -1, laplacian_kernel_negative4)
plt.imshow(lk_positive4, cmap = 'gray')
plt.title('laplacian kernel negative4')
```

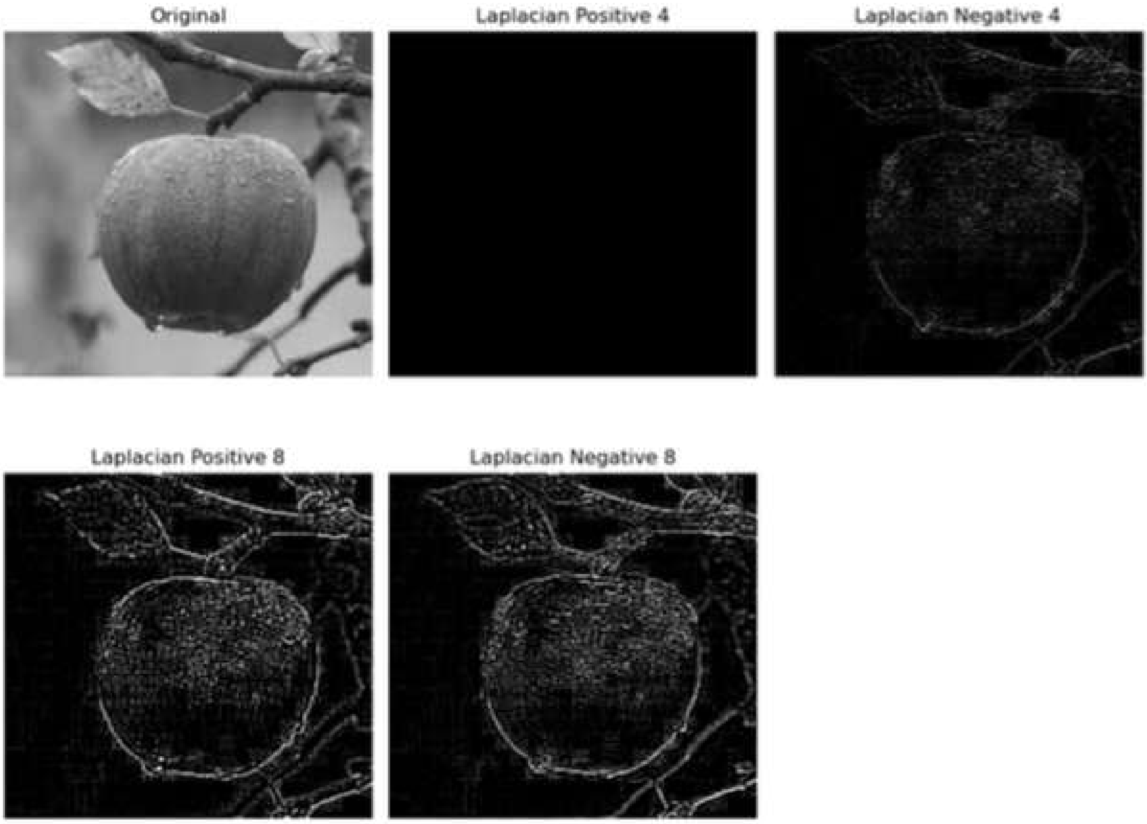
Out[7]:

```
Text(0.5, 1.0, 'laplacian kernel negative4')
```



In [8]:

```
# Apply the Laplacian kernels to the image
output_positive4 = cv2.filter2D(image, -1, laplacian_kernel_positive4)
output_negative4 = cv2.filter2D(image, -1, laplacian_kernel_negative4)

output_positive8 = cv2.filter2D(image, -1, laplacian_kernel_positive8)
output_negative8 = cv2.filter2D(image, -1, laplacian_kernel_negative8)

# Create a list of titles and images for plotting
titles = ['Original', 'Laplacian Positive 4', 'Laplacian Negative 4',
          'Laplacian Positive 8', 'Laplacian Negative 8']
images = [image, output_positive4, output_negative4, output_positive8, output_ne

# Plot the images using Matplotlib
plt.figure(figsize=(10, 8))
for i in range(len(images)):
    plt.subplot(2, 3, i+1)
    plt.imshow(images[i], cmap='gray')
    plt.title(titles[i])
    plt.axis('off')

plt.tight_layout()
plt.show()
```



high boost filtering

In [10]:

```
# apply gaussian blur to image
blurred = cv2.GaussianBlur(image, (9,9), 20)

# calculate the mask(og image- blurred img)
mask = image- blurred

# high- boost filtering(image + k*mask)
k = 2   # adjust this value for stronger/weaker high boost effect
```

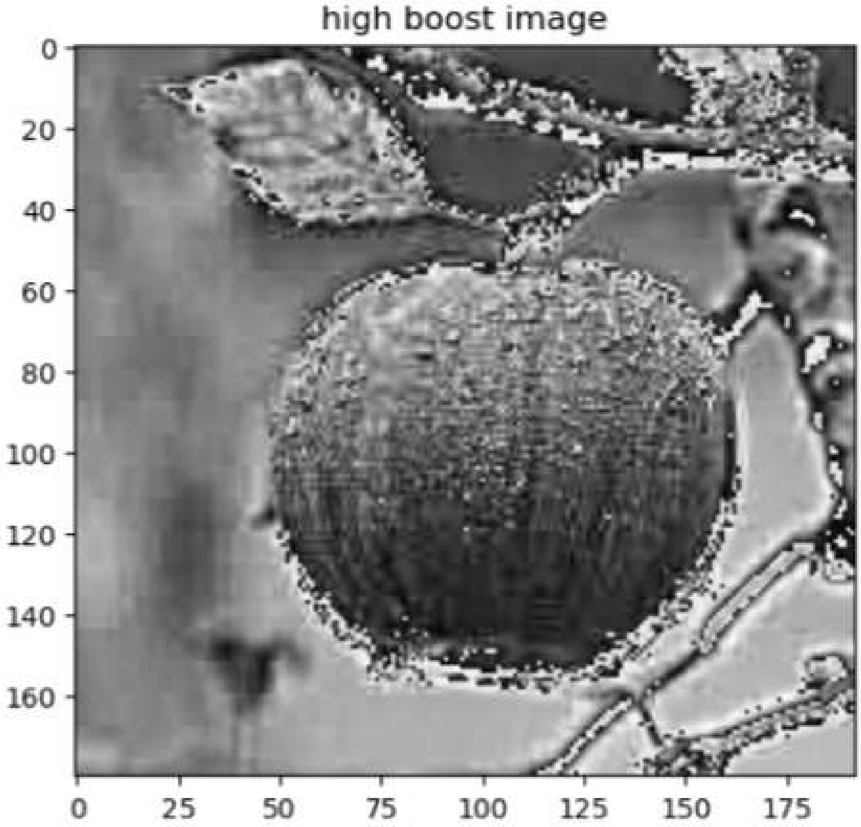
```
high_boost_image = image + k* mask

# clip the values to be in the valis range [0,255] and convert to unint8
high_boost_image = np.clip(high_boost_image, 0, 255).astype(np.uint8)
```

In [11]:

```
plt.imshow(high_boost_image, cmap = 'gray')
plt.title('high boost image')
```

Out[11]:

```
Text(0.5, 1.0, 'high boost image')
```

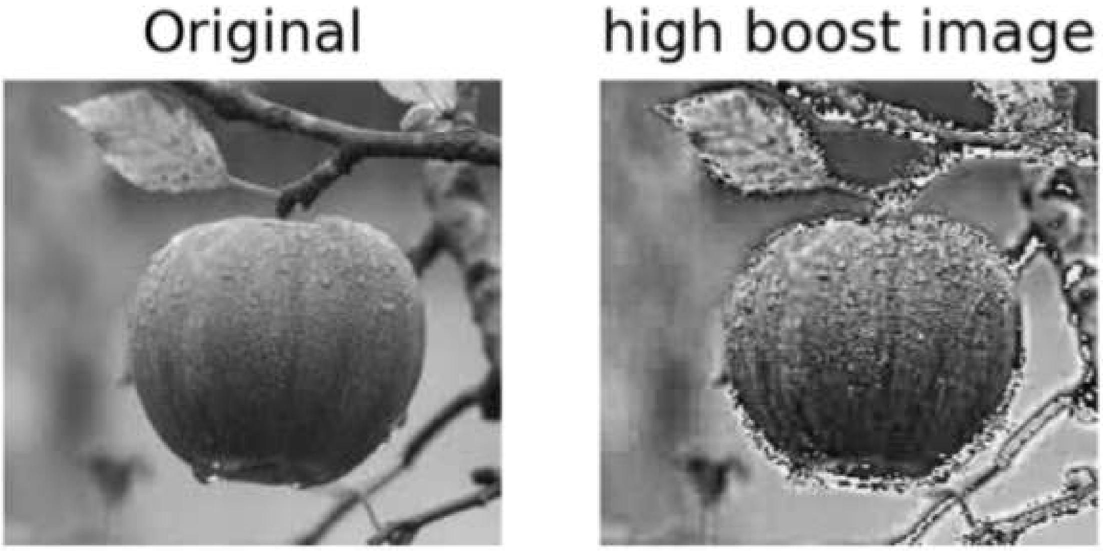


In [12]:

```
fig=plt.figure(dpi=300)

fig.add_subplot(1,3,1)
plt.imshow(image,cmap='gray')
plt.axis("off")
plt.title("Original")

fig.add_subplot(1,3,2)
plt.imshow(high_boost_image,cmap='gray')
plt.axis("off")
plt.title("high boost image")
```

Out[12]:

```
Text(0.5, 1.0, 'high boost image')
```



Original

high boost image

In [ ]: