lab3-21035

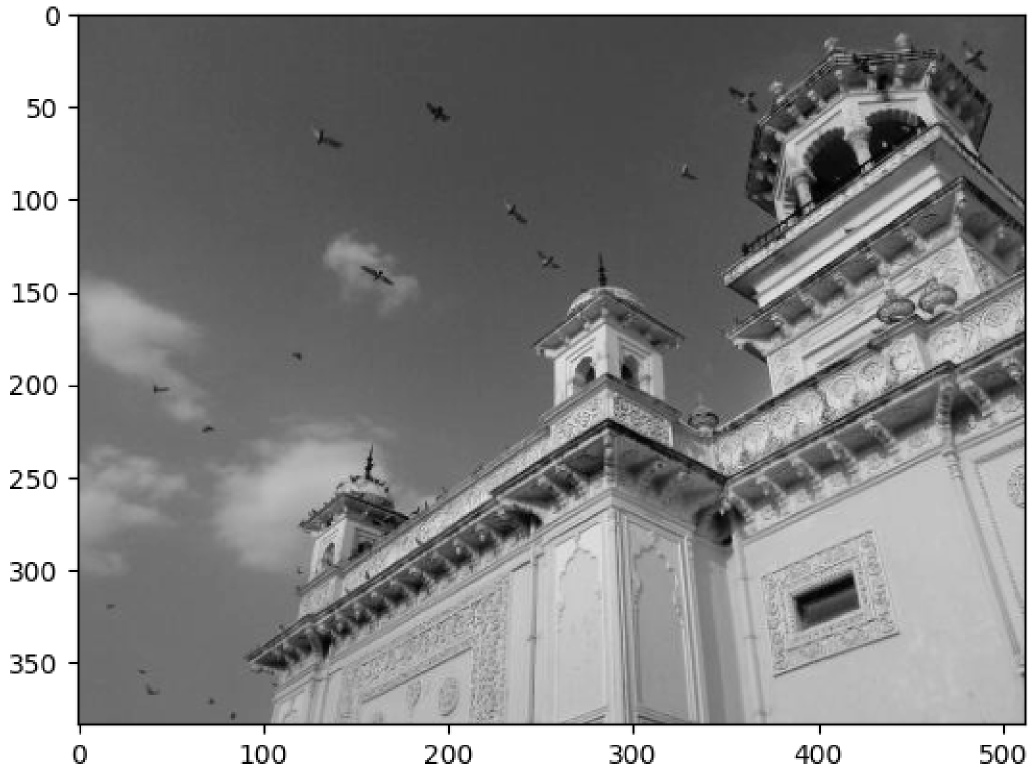
November 19, 2023

[34]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import math
```

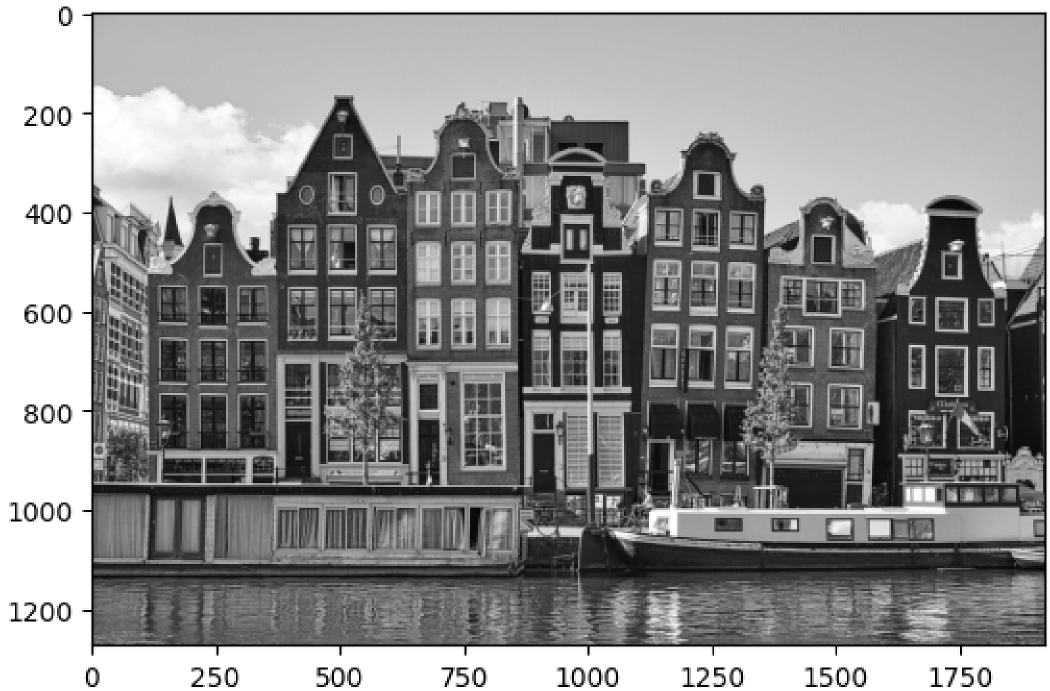
[35]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Read the image
img = cv2.imread('buildings.jpg')

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Show the image with matplotlib
plt.imshow(gray, cmap='gray')
plt.show()
```

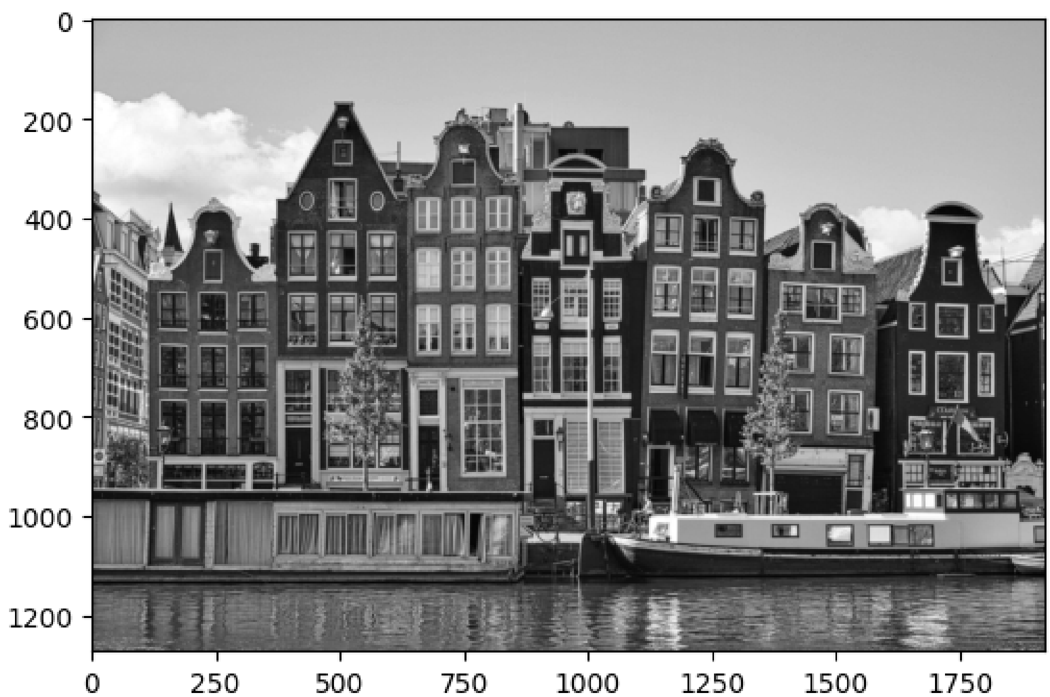
[36]:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Read the image
img = cv2.imread('home.jpg')

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Show the image with matplotlib
plt.imshow(gray, cmap='gray')
plt.show()
```



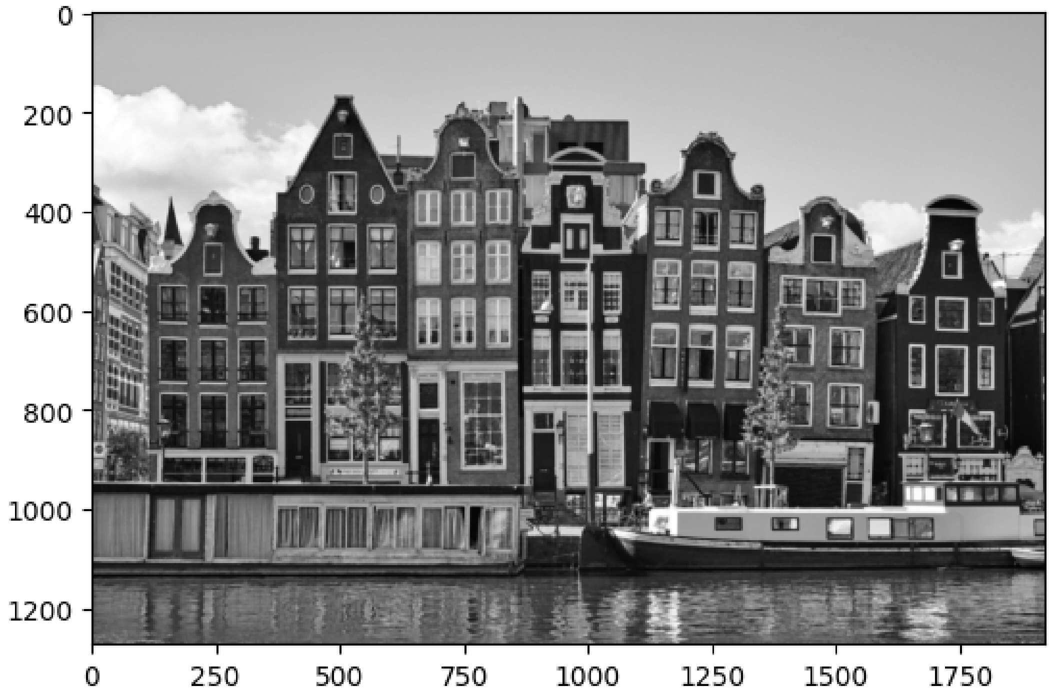
[37]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Read the image
img = cv2.imread('buildings.jpg')

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Show the image with matplotlib
plt.imshow(gray, cmap='gray')
plt.show()

# Mean filter
kernel = np.ones((3,3),np.float32)/9
dst = cv2.filter2D(gray,-1,kernel)
plt.imshow(dst, cmap='gray')
plt.show()
```


0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750


0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750

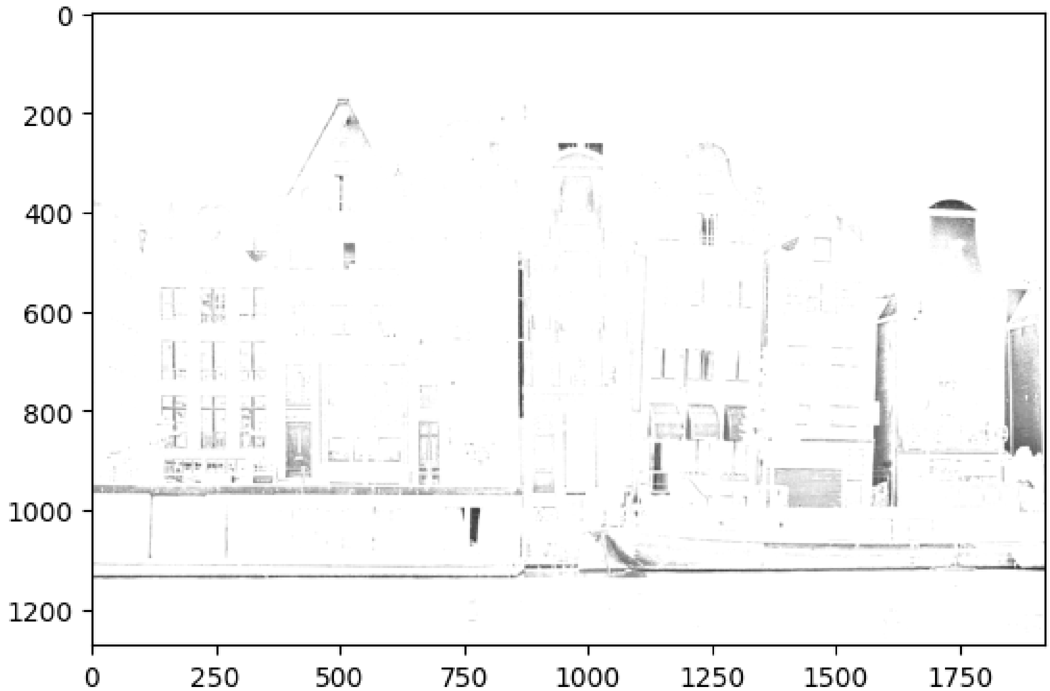
[38]:

```
#Median filter
median = cv2.medianBlur(gray,5)
plt.imshow(median, cmap='gray')
plt.show()
```

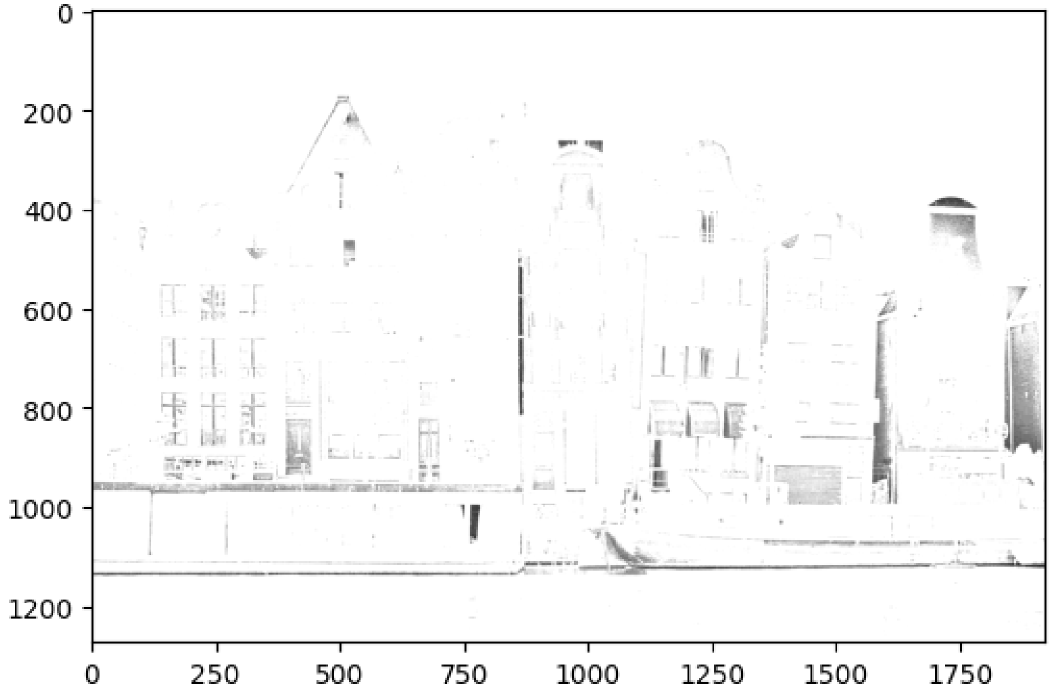


[39]:

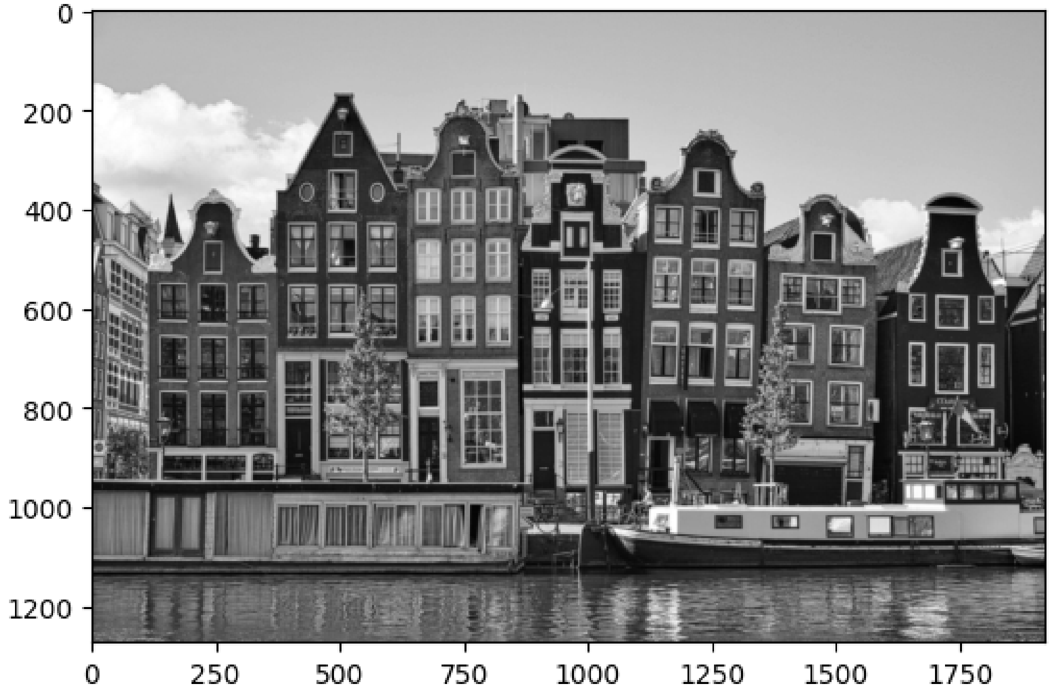
```
#max filter
kernel = np.ones((3,3),np.float32)
dst = cv2.filter2D(gray,-1,kernel)
plt.imshow(dst, cmap='gray')
plt.show()
```



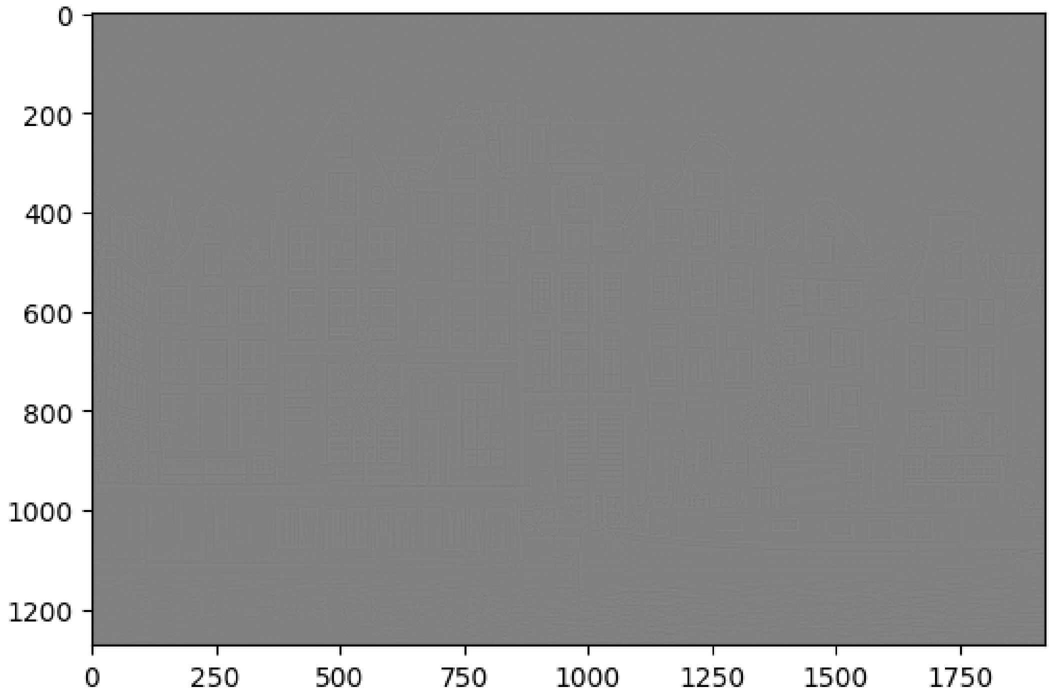
[40]:
```
#min filter
kernel = np.ones((3,3),np.float32)
dst = cv2.filter2D(gray,-1,kernel)
plt.imshow(dst, cmap='gray')
plt.show()
```



[41]:
```
#Guassian filter
blur = cv2.GaussianBlur(gray,(5,5),0)
plt.imshow(blur, cmap='gray')
plt.show()
```



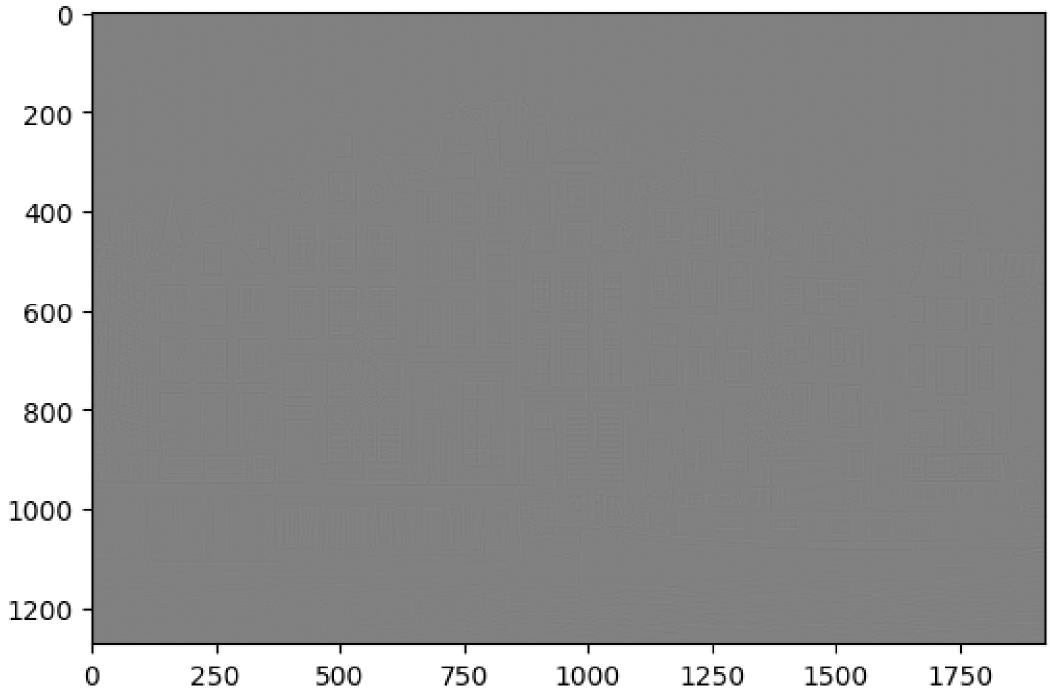
[42]:
```
#laplacian filter
laplacian = cv2.Laplacian(gray,cv2.CV_64F)
plt.imshow(laplacian, cmap='gray')
plt.show()
```

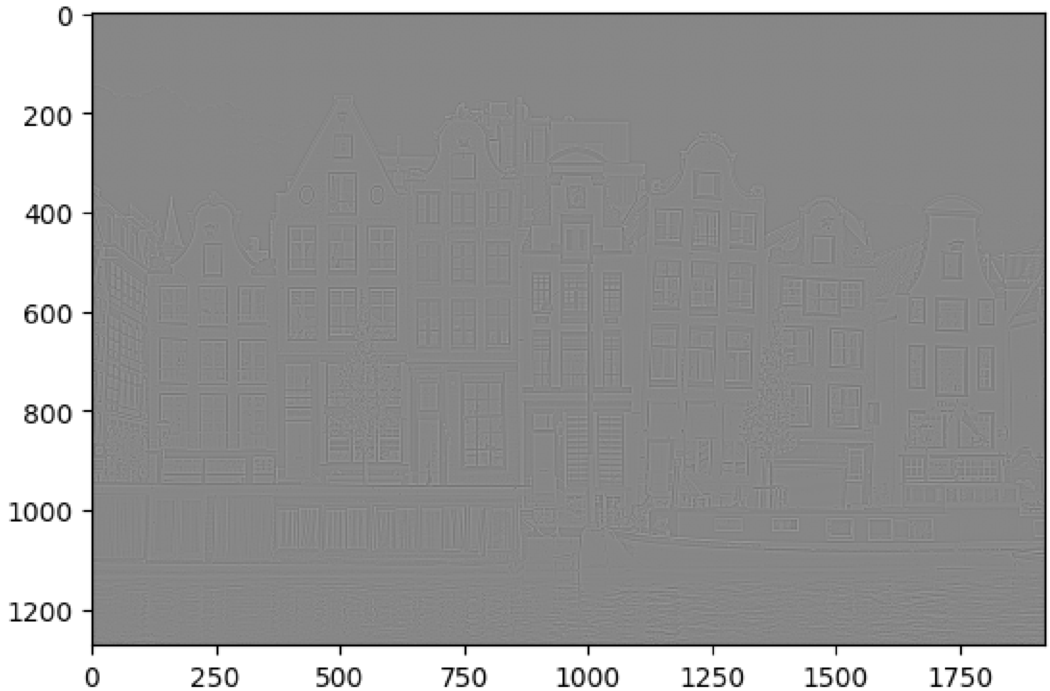


3.a.Apply an edge detector to the image and observe the intensity of the resulting output. Subsequently, employ a 3×3 mean filter on the initial image and reapply the edge detector. Comment on the difference. Investigate the effects of using a 5×5 or a 7×7 filter in this context

[43]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Read the image
img = cv2.imread('buildings.jpg')

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#laplacian filter
laplacian = cv2.Laplacian(gray,cv2.CV_64F)
plt.imshow(laplacian, cmap='gray')
plt.show()

#Mean filter
kernel = np.ones((3,3),np.float32)/9
dst = cv2.filter2D(gray,-1,kernel)
plt.imshow(dst, cmap='gray')
```

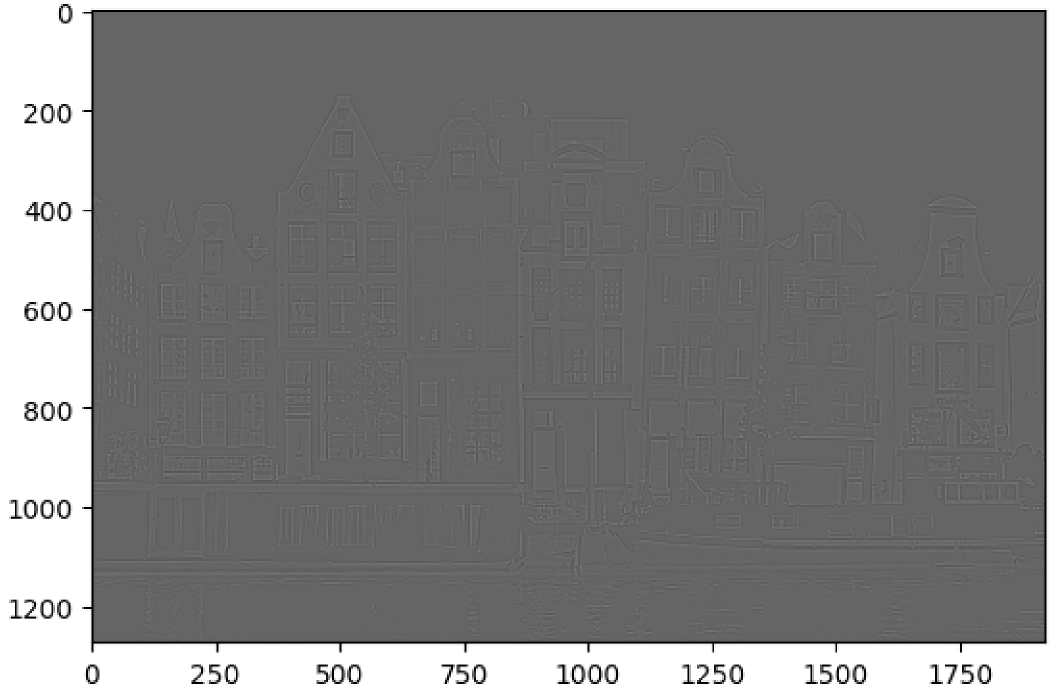
```
#laplacian filter
laplacian = cv2.Laplacian(dst,cv2.CV_64F)
plt.imshow(laplacian, cmap='gray')
plt.show()
```



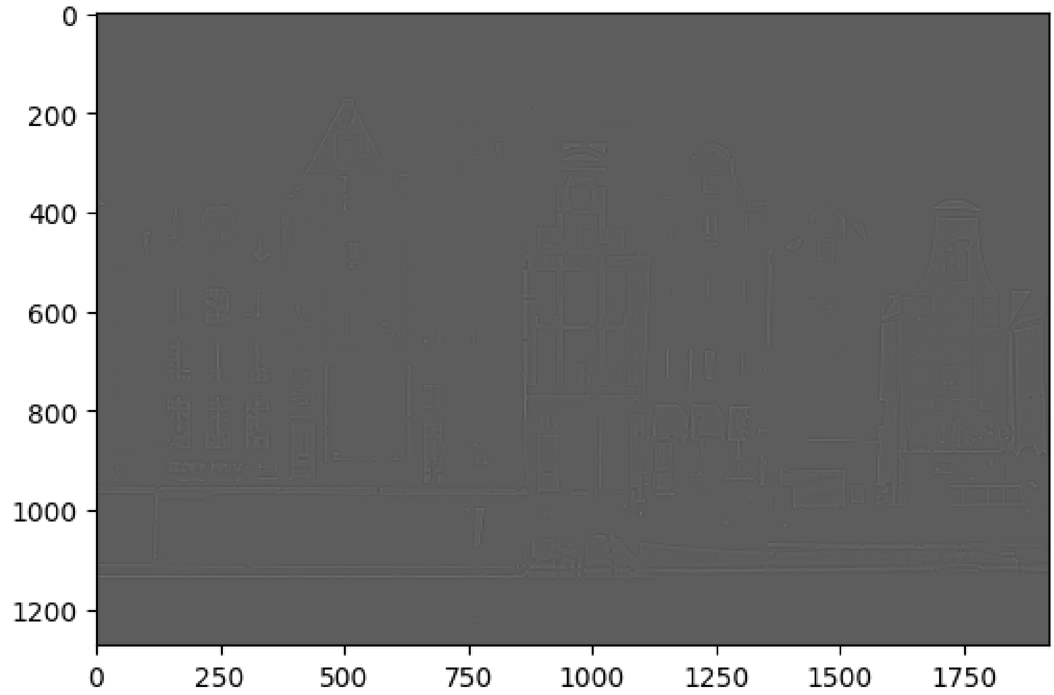


[44]:

```
#Mean filter
kernel = np.ones((5,5),np.float32)/9
dst = cv2.filter2D(gray,-1,kernel)
plt.imshow(dst, cmap='gray')

#laplacian filter
laplacian = cv2.Laplacian(dst,cv2.CV_64F)
plt.imshow(laplacian, cmap='gray')
plt.show()
```



[45]:
```
#Mean filter
kernel = np.ones((7,7),np.float32)/9
dst = cv2.filter2D(gray,-1,kernel)
plt.imshow(dst, cmap='gray')

#laplacian filter
laplacian = cv2.Laplacian(dst,cv2.CV_64F)
plt.imshow(laplacian, cmap='gray')
plt.show()
```



#The difference between above three is :

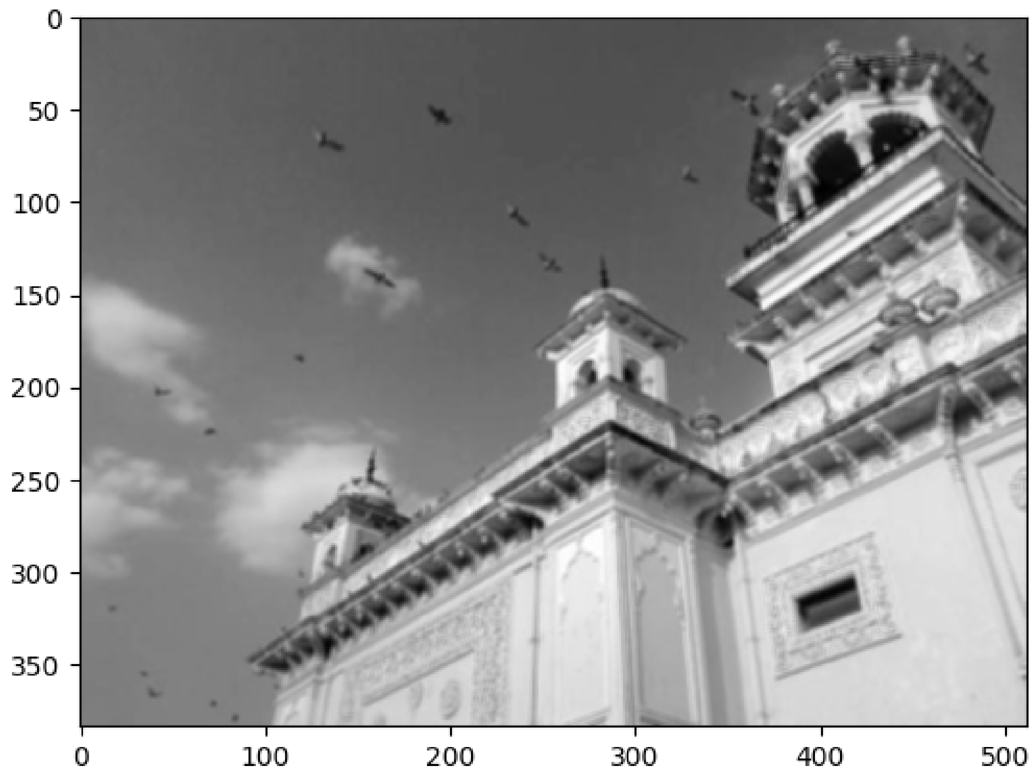
#The edges are more visible in the first image than the second and third image. #The edges are more visible in the second image than the third image.

3.b.Assess the comparative speed of mean and median filters when employing identical-sized neighbourhoods and images. Analyse how the efficiency of each filter is influenced by variations in the image size and the size of the neighbourhood

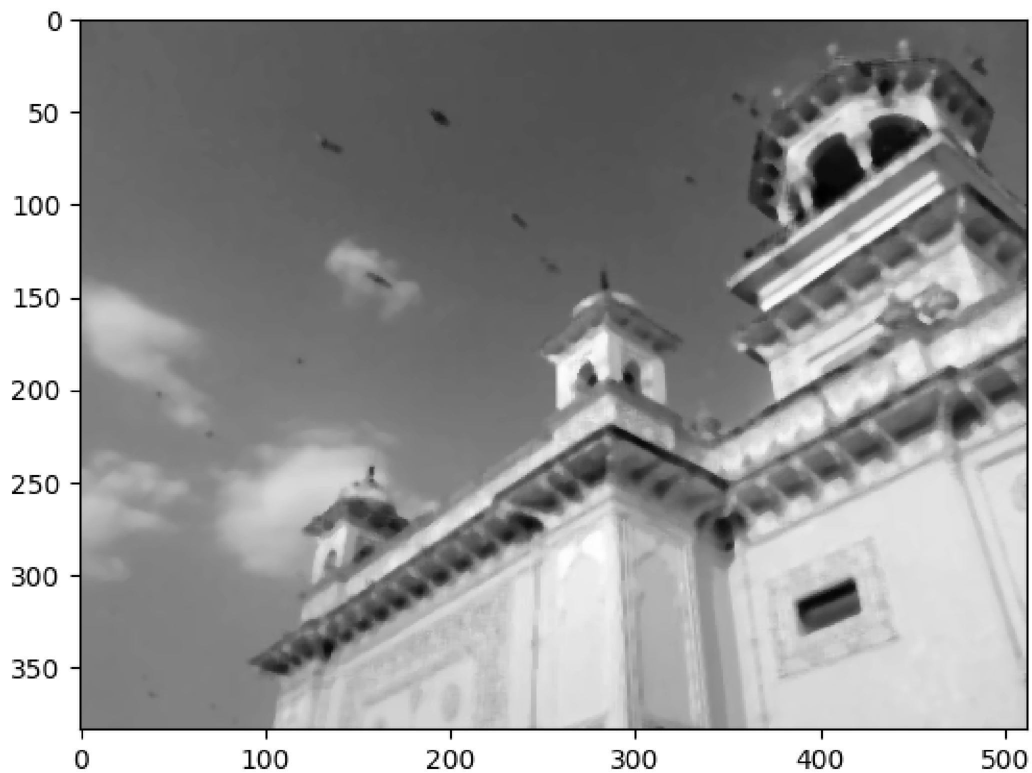
[46]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import time

# Read the image
img = cv2.imread('home.jpg')
# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#Mean filter
start = time.time()
kernel = np.ones((3,3),np.float32)/9
dst = cv2.filter2D(gray,-1,kernel)
end = time.time()
```

```
print(end - start)
plt.imshow(dst, cmap='gray')
plt.show()

#Median filter
start = time.time()
median = cv2.medianBlur(gray,5)
end = time.time()
print(end - start)
plt.imshow(median, cmap='gray')
plt.show()
```

0.0003190040588378906

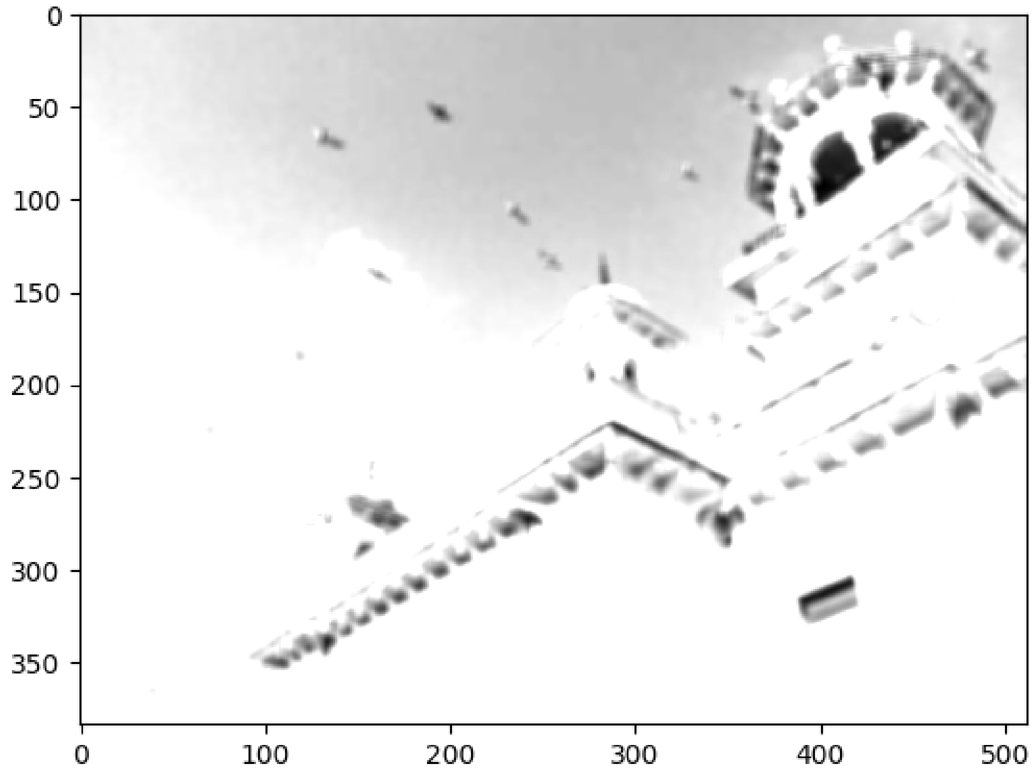


0.0006730556488037109

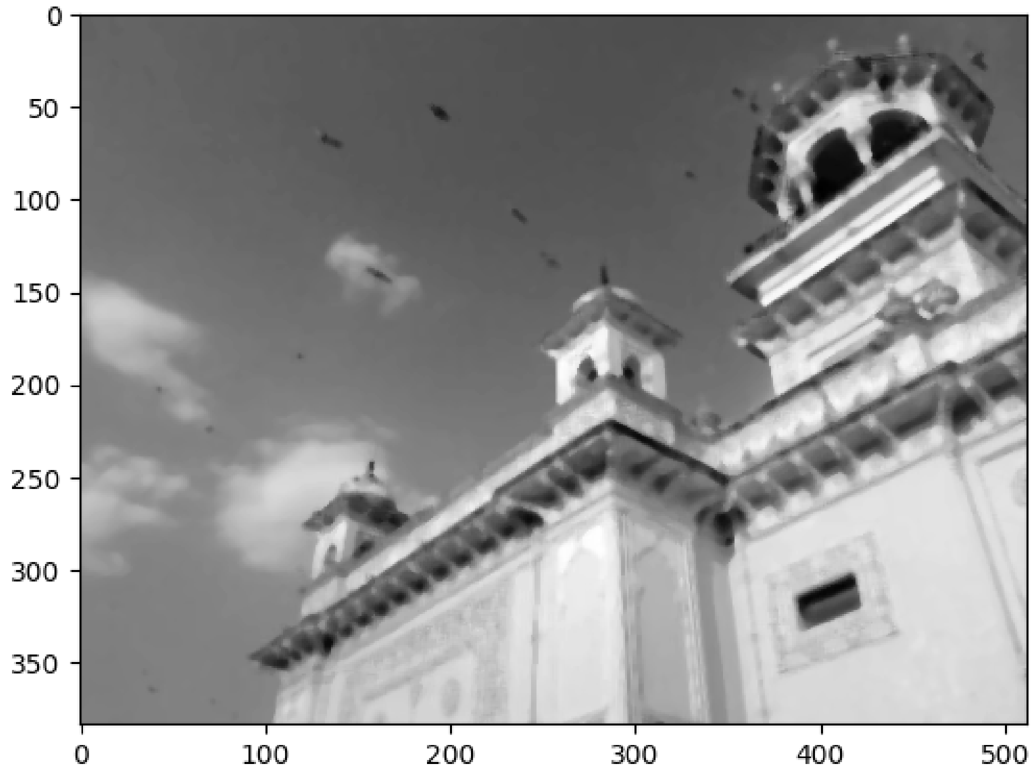


[47]:
```
#Mean filter
start = time.time()
kernel = np.ones((5,5),np.float32)/9
dst = cv2.filter2D(gray,-1,kernel)
end = time.time()
print(end - start)
plt.imshow(dst, cmap='gray')
plt.show()

#Median filter
start = time.time()
median = cv2.medianBlur(gray,5)
end = time.time()
print(end - start)
plt.imshow(median, cmap='gray')
plt.show()
```

```
0.0005409717559814453
```



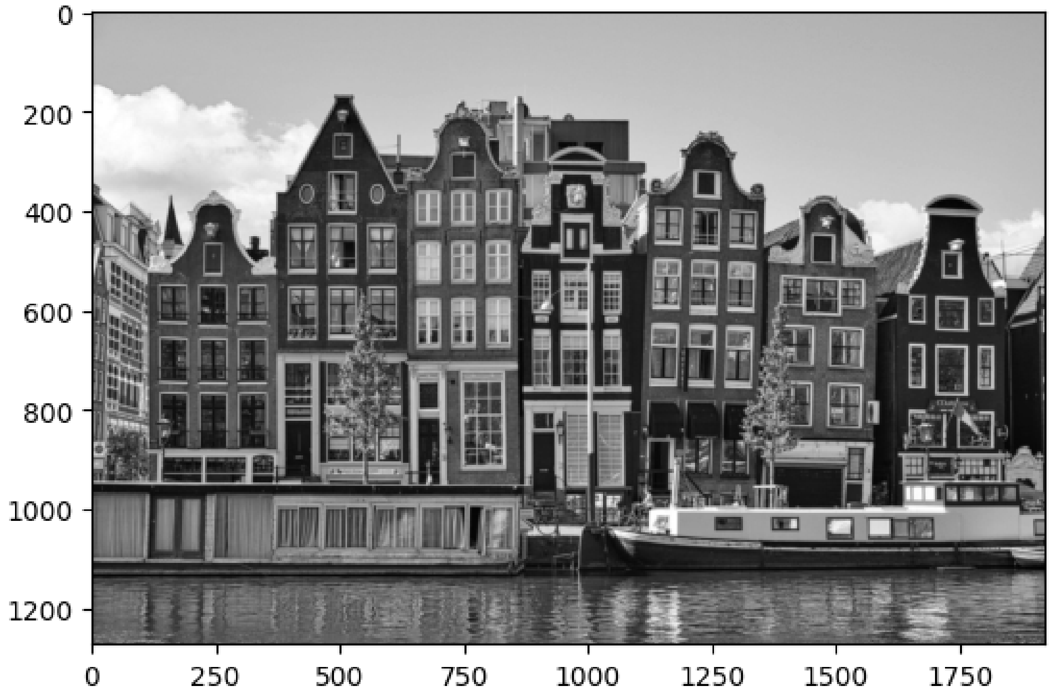
0.0008330345153808594

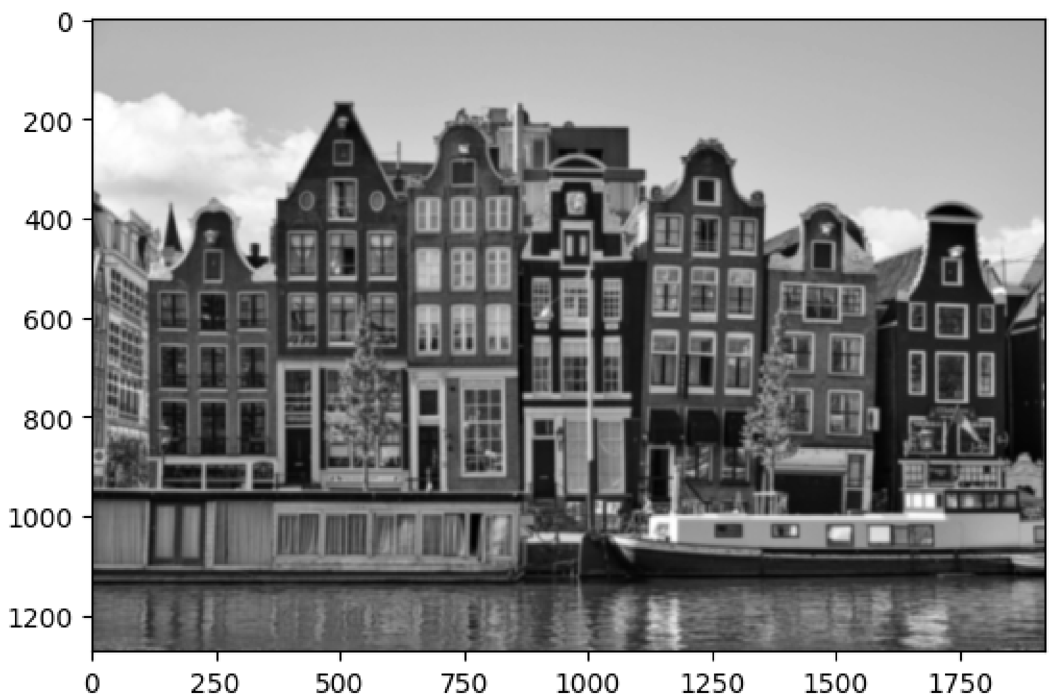


3.c.Experiment with the Gaussian filter using various sigma values and evaluate each one on the basis of noise removal and preservation of image details.

[48]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Read the image
img = cv2.imread('buildings.jpg')

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#Guassian filter
blur = cv2.GaussianBlur(gray,(5,5),0)
plt.imshow(blur, cmap='gray')
plt.show()

#Guassian filter
blur = cv2.GaussianBlur(gray,(15,15),0)
plt.imshow(blur, cmap='gray')
```

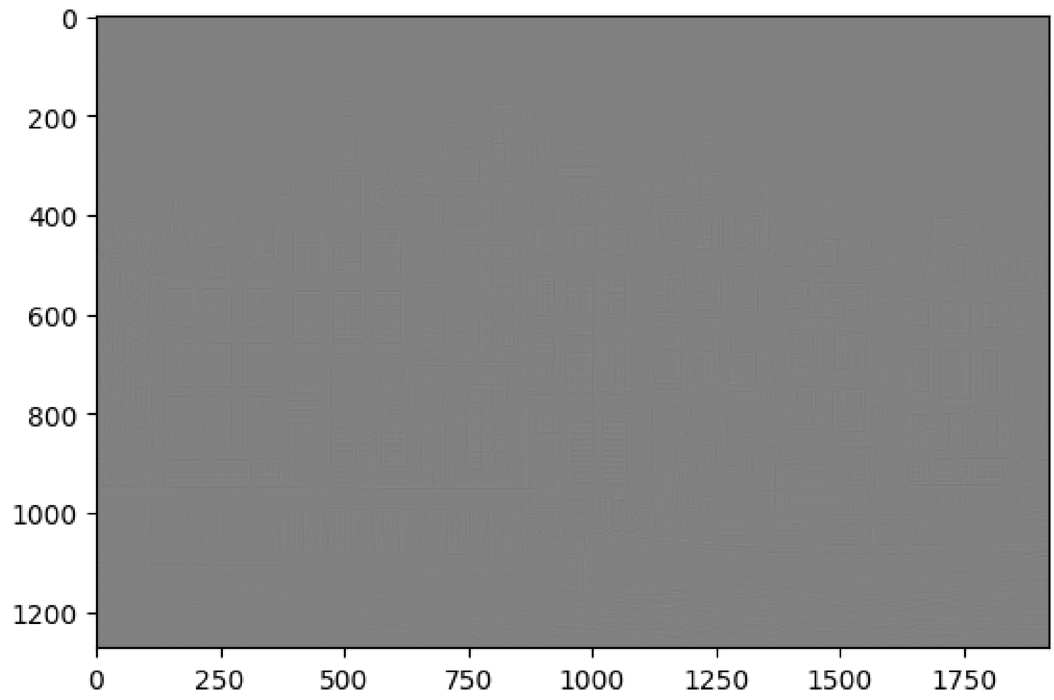
```
plt.show()
```


0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750


0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750

3.d.Apply following Laplacian filters over the image and compare the results

[49]:
```python
#Apply following Laplacian filters over the image

import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Read the image
img = cv2.imread('buildings.jpg')

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#laplacian filter
laplacian = cv2.Laplacian(gray,cv2.CV_64F)
plt.imshow(laplacian, cmap='gray')
plt.show()
```



#Results:

The edges are more visible in the first image than the second image.

1 what will happen if the kernel weights are negated ?

If you negate the kernel weights, you change the way the convolution operation works. Instead of emphasizing certain features, the convolution operation will now suppress them. This is because convolution is a linear operation, and flipping the sign of the kernel weights is equivalent to changing the phase of the operation.