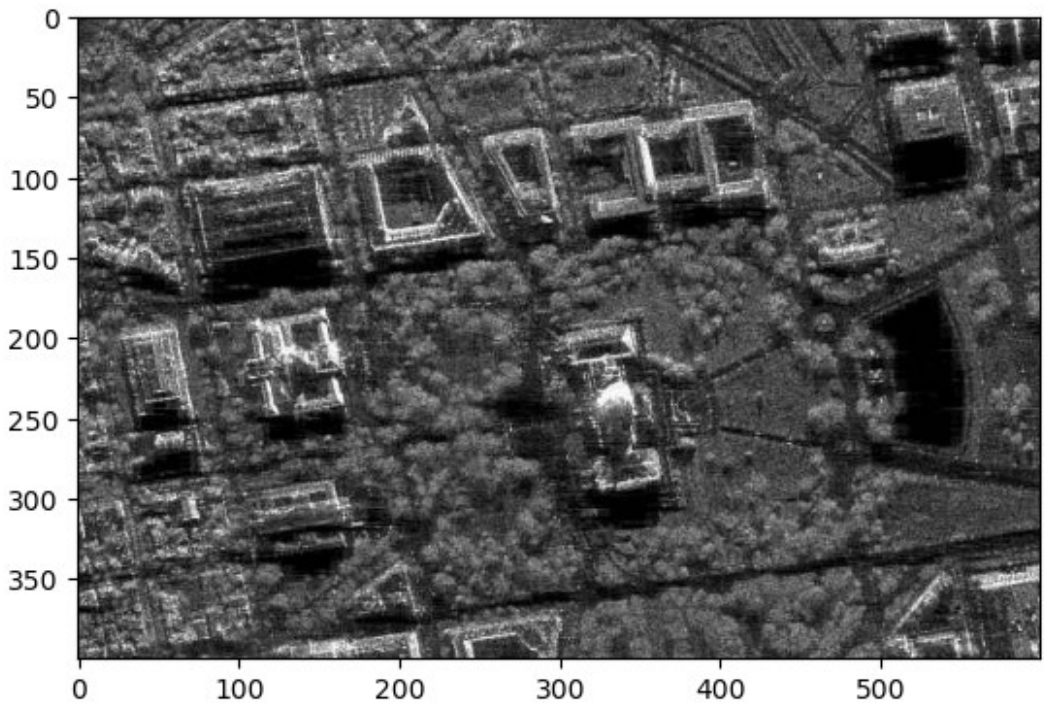
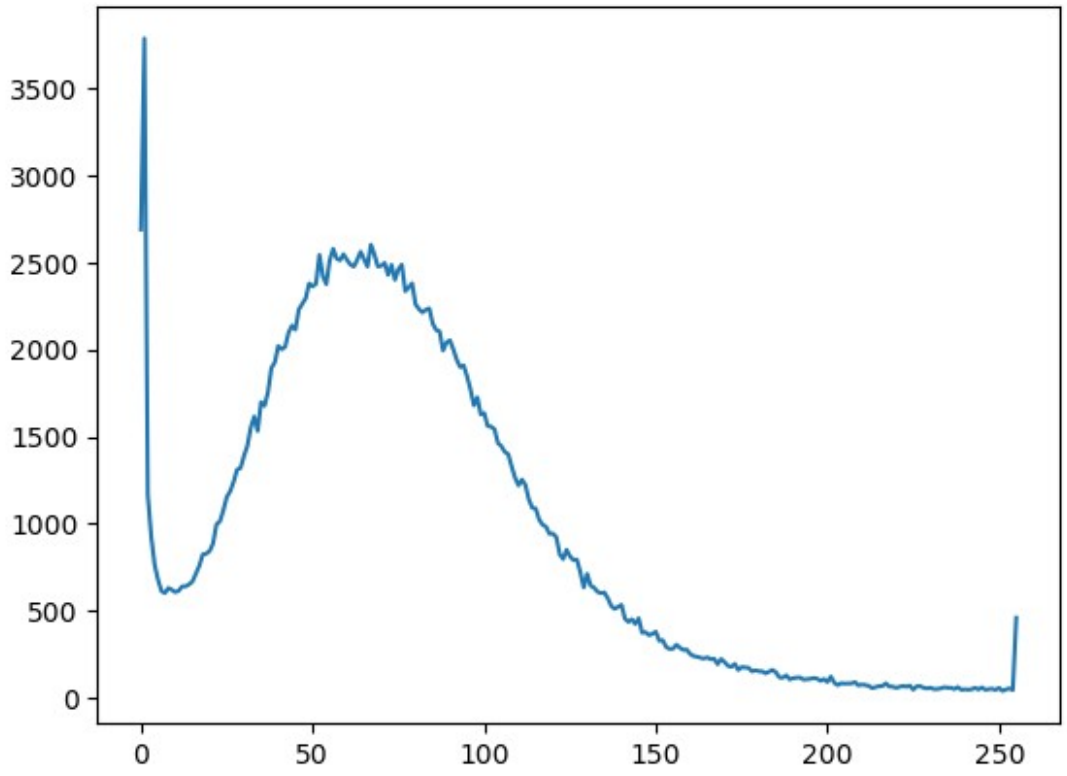
```
import numpy as np
import cv2
import matplotlib.pyplot as plt


# 1. Загрузите изображение в оттенках серого sar_1_gray.jpg.

image = cv2.imread('sar_1_gray.jpg')

image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

image1 = cv2.imread('sar_2_color.jpg')

image1_gray = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)

plt.imshow(image)

<matplotlib.image.AxesImage at 0xd7897e8>
```

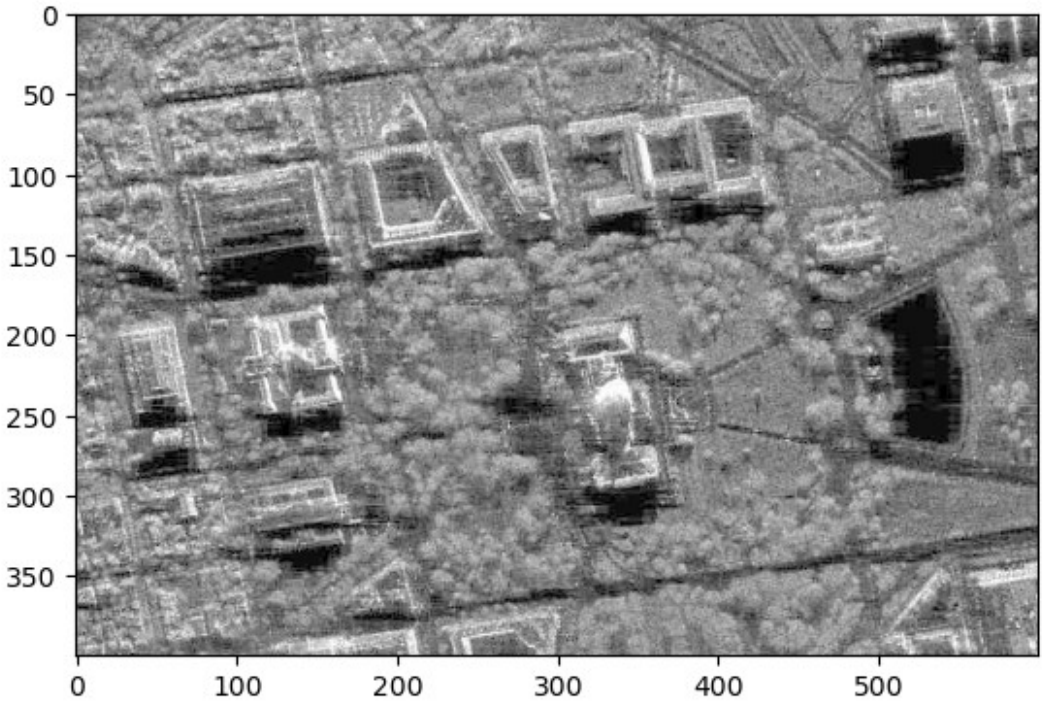



```
# 2. постройте гистограмму

hSize = 256
hRange = [0, 256]

h = cv2.calcHist([image], [0], None, [hSize], hRange,
accumulate=False)
```

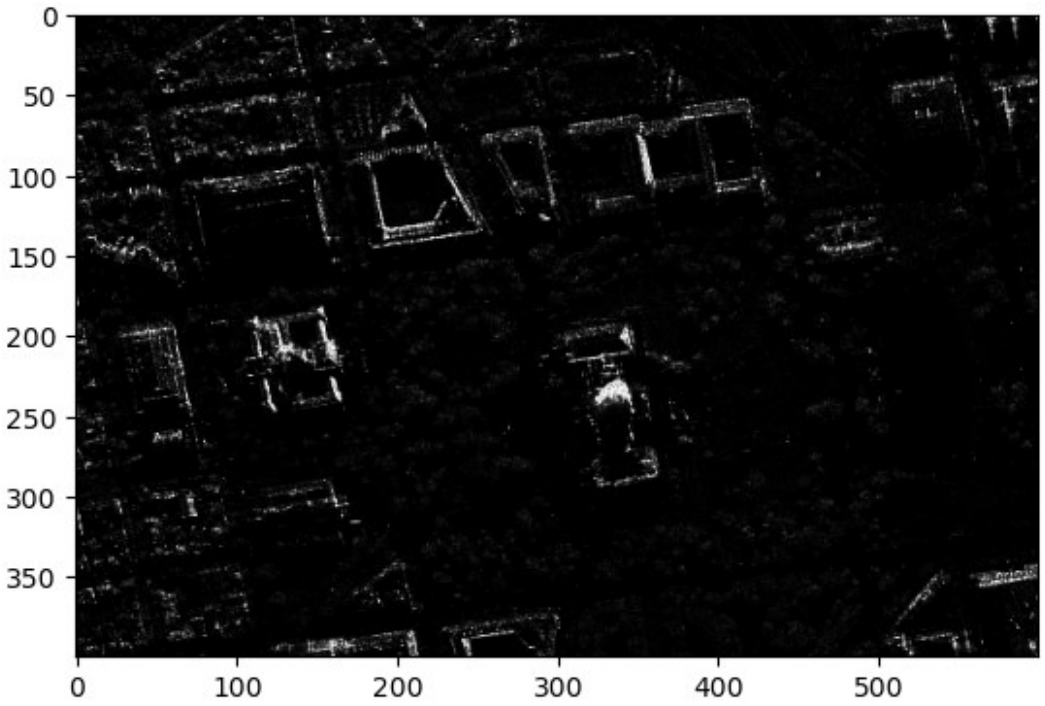
```
plt.plot(h);
```




```
# 3. реализуйте алгоритм гамма коррекции с параметром гамма <1, >1.

rMax = 255;
gamma1 = 0.5
im1_gray = image_gray;
r1 = [((i / rMax) ** gamma1) * rMax for i in range(256)];
r1 = np.array(r1, np.uint8);
im1_gray = cv2.LUT(im1_gray, r1)

rMax = 255;
gamma2 = 4.0;
im2_gray = image_gray;
r2 = [((i / rMax) ** gamma2) * rMax for i in range(256)];
r2 = np.array(r2, np.uint8);
im2_gray = cv2.LUT(im2_gray, r2)

plt.imshow(im1_gray, cmap='gray');
```




```
plt.imshow(im2_gray, cmap='gray');
```



```
# 4. Сравните исходное изображение, скорректированное при помощи гамма-фильтра. MSE, SSIM.

from skimage.metrics import structural_similarity, mean_squared_error

# MSE

mse = mean_squared_error(image_gray, im1_gray)

mse
```

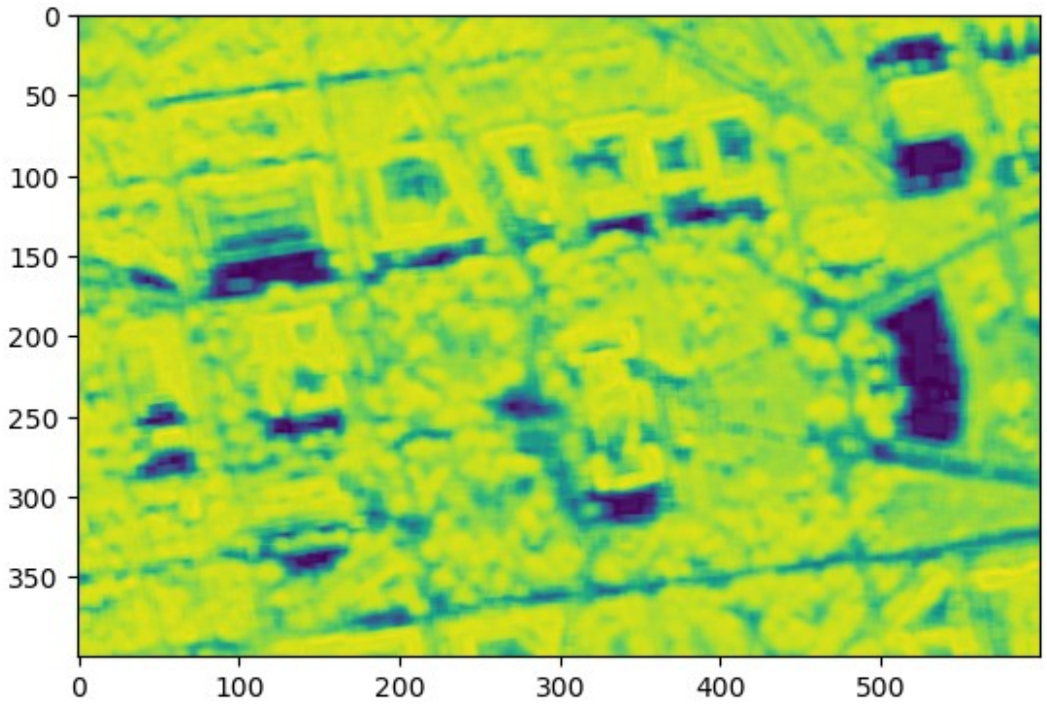
```
3250.429145833333
```

```
# SSIM

(ssim, diff) = structural_similarity(image_gray, im1_gray, full=True)
diff = (diff * 255).astype("uint8")
print("SSIM: {}".format(ssim))
```

```
SSIM: 0.7875008686792753
```

```
plt.imshow(diff)
```

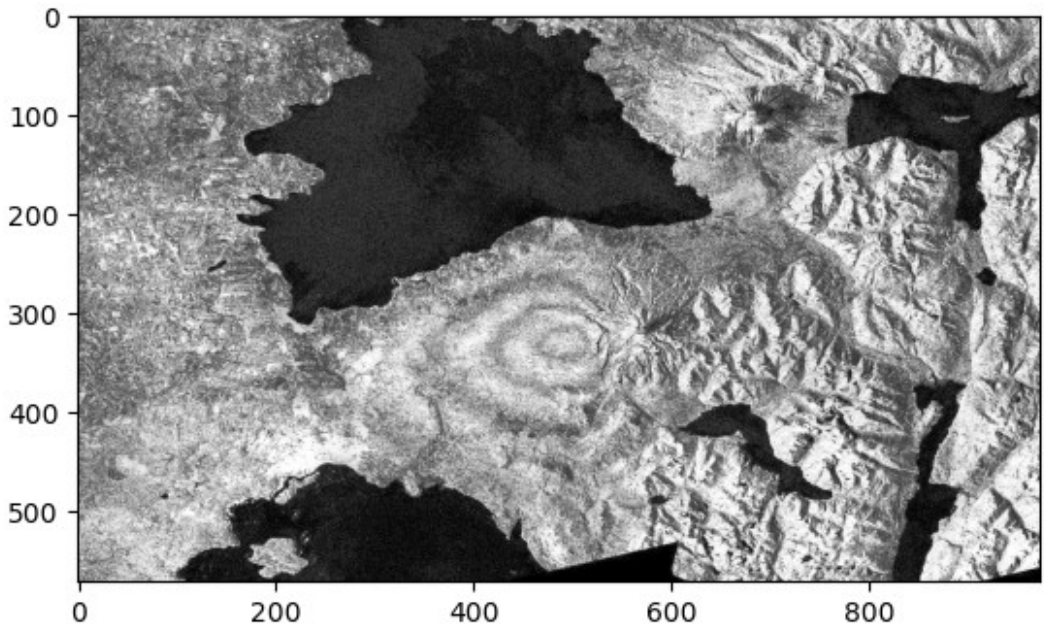
```
<matplotlib.image.AxesImage at 0xd76d7e0>
```




```
# 5. реализуйте алгоритм статистической цветокоррекции на основе статистики eq_gray.
```

```
eq_gray = cv2.equalizeHist(image1_gray)
image_gray = cv2.equalizeHist(image_gray)

plt.imshow(eq_gray, cmap="gray")
```

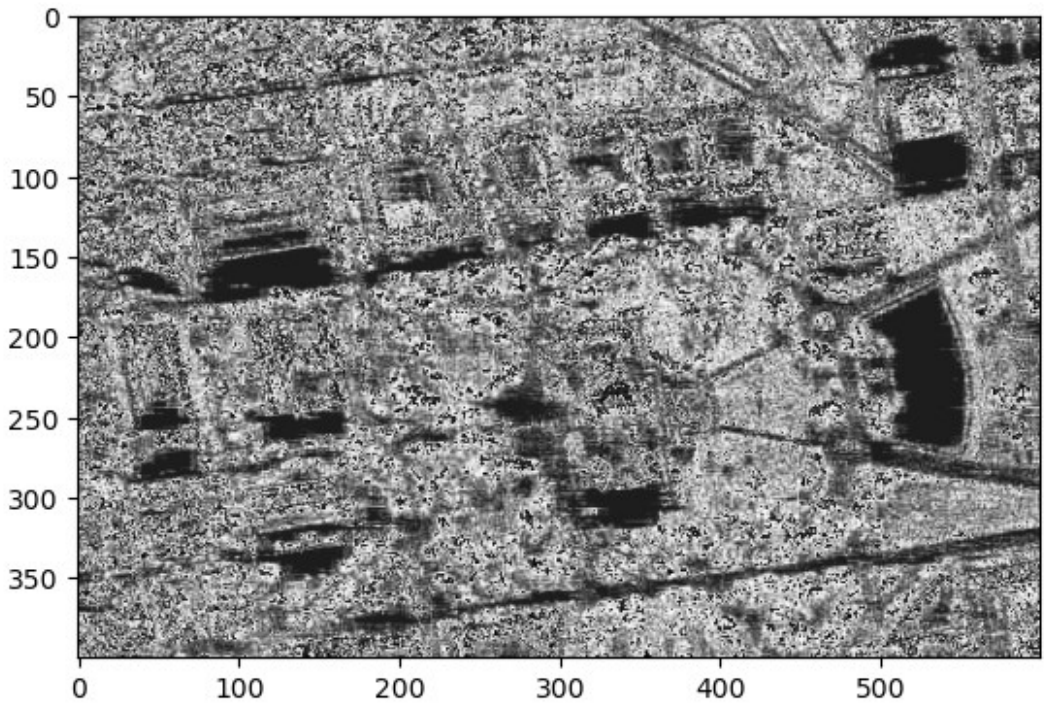
```
<matplotlib.image.AxesImage at 0xe154cc0>
```




```
e1 = np.sum(np.sum(eq_gray, axis = 0), axis = 0) # мат ожидание
изображения eq_grey
e = np.sum(np.sum(image_gray, axis = 0), axis = 0) # мат ожидание
изображения исходного

rows,cols = eq_gray.shape
rows1,cols1 = image_gray.shape

e1 /= (rows*cols)
e /= (rows1*cols1)

sum = 0
for i in range(rows):
    for j in range(cols):
        k   = eq_gray[i,j]
        sum += ((k-e1)**2);

d = (sum / (rows*cols))**0.5 # дисперсия изображения eq_gray

sum1 = 0
for i in range(rows1):
    for j in range(cols1):
```

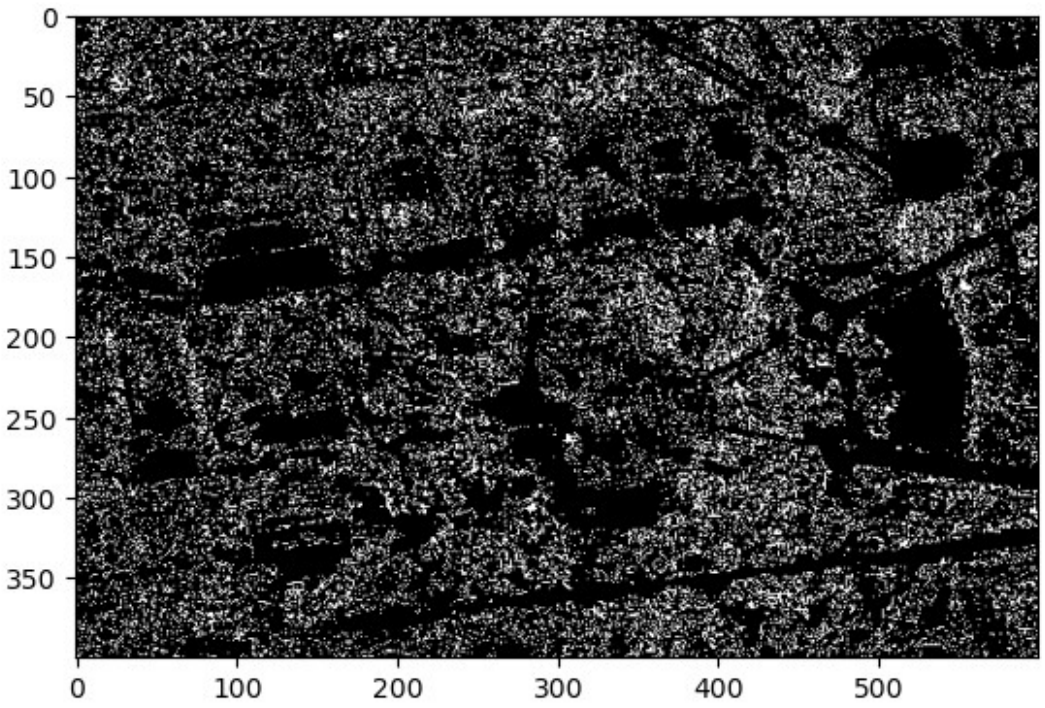
```
        k1 = image_gray[i,j]
        sum1+=((k1-e)**2)

d1 = (sum1/(rows1*cols1))**0.5 # дисперсия изображения исходного

for i in range(rows1):
    for j in range(cols1):
        image_gray[i,j] = (image_gray[i,j]-e)*(d/d1)+e1

plt.imshow(image_gray, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0xd8ee900>
```




```
# 6. Протестируйте работу алгоритмов пороговой фильтрации с различными параметрами.

_,thresh1 = cv2.threshold(image_gray,200,255,cv2.THRESH_BINARY)

plt.imshow(thresh1, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0xc3c2a80>
```



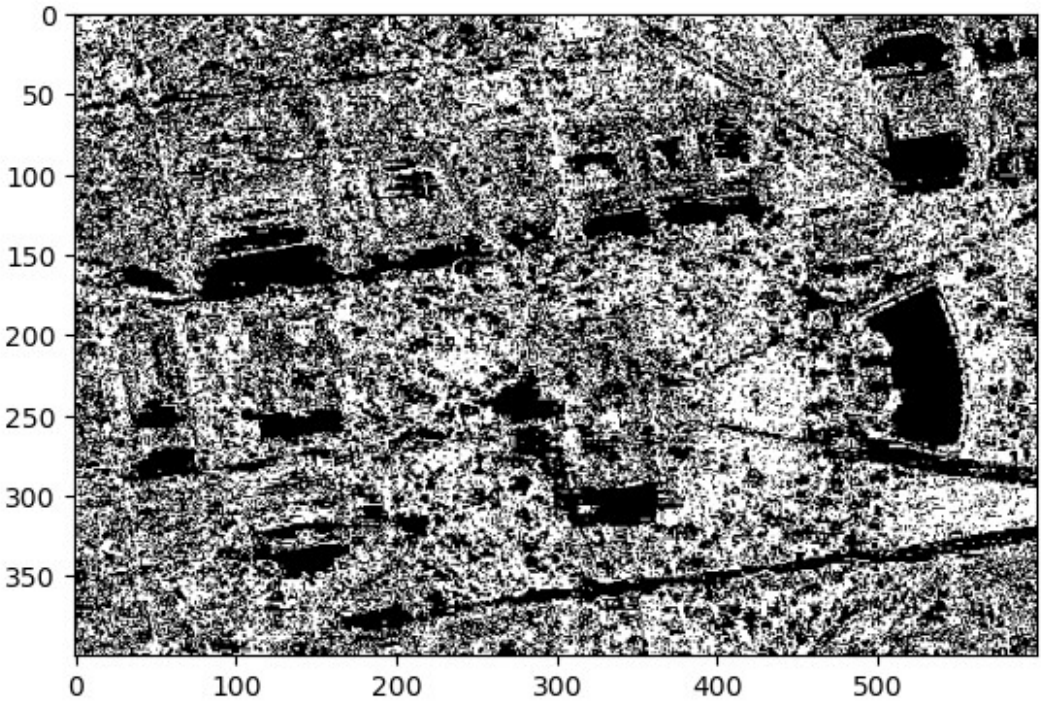

```
thresh1[thresh1==100].sum()
```

```
0
```

```
_,thresh2 = cv2.threshold(image_gray,100,255,cv2.THRESH_BINARY)
```

```
plt.imshow(thresh2, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0xc0c1568>
```




```
_,thresh3 = cv2.threshold(image_gray,10,255,cv2.THRESH_BINARY)
plt.imshow(thresh3, cmap='gray')
<matplotlib.image.AxesImage at 0xc0d34f0>
```