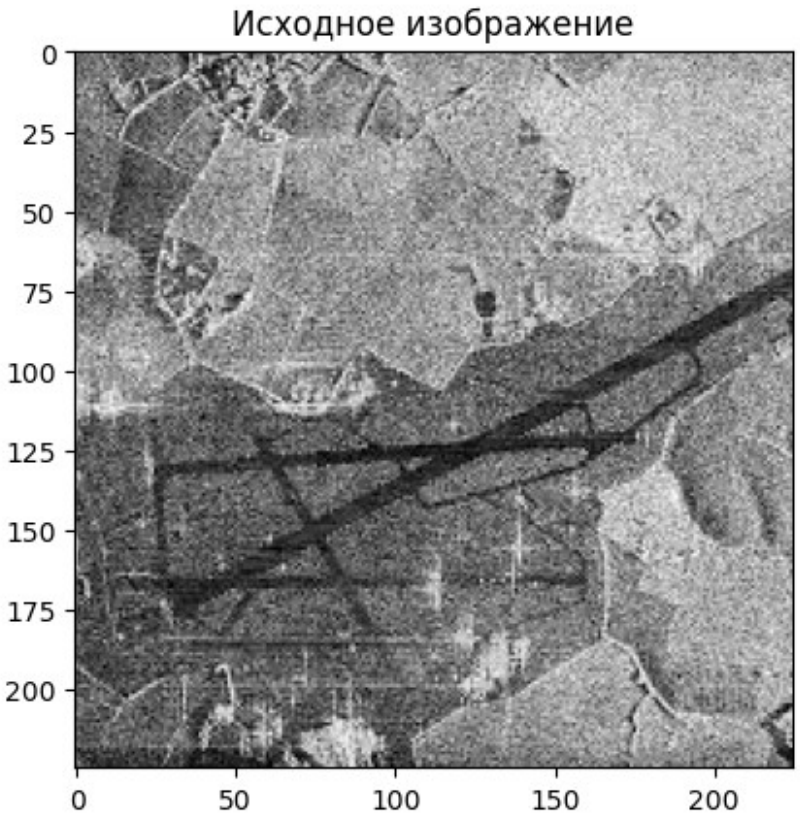
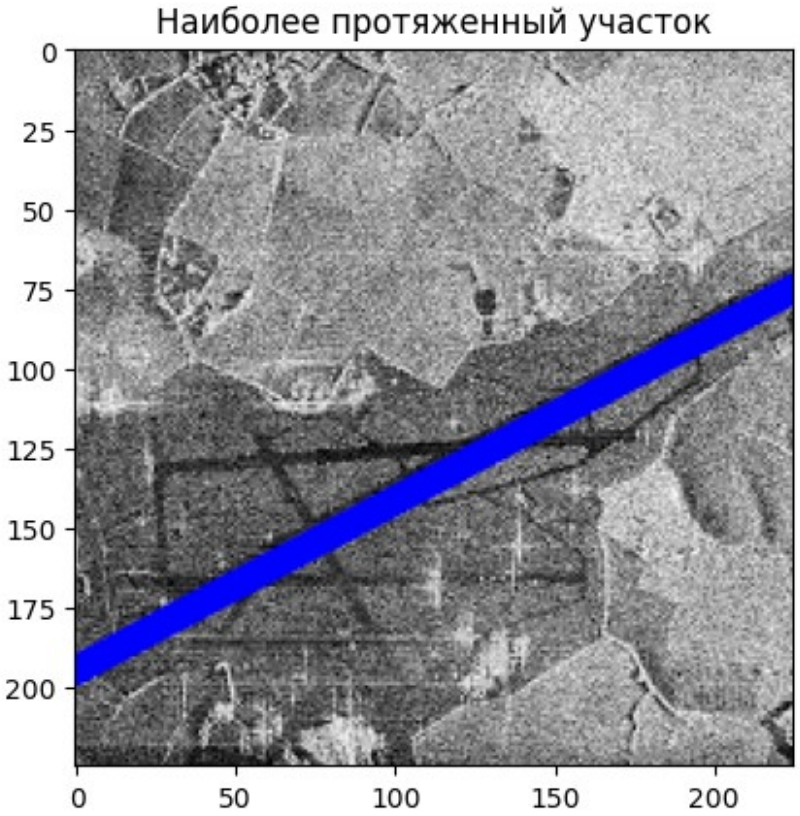
```
import math
import numpy as np
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('sar_3.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(image_gray, cmap="gray")
plt.title("Исходное изображение")
plt.show()
```




```
blur = cv2.GaussianBlur(image_gray, (5,5), 0)
canny = cv2.Canny(blur, 100, 200)
lines = cv2.HoughLines(canny, 1, np.pi / 180, threshold = 100)
image_line = image.copy()
if lines is not None:
    longest_line = None
    max_length = 0
    for rho, theta in lines[:, 0]:
        a = np.cos(theta)
        b = np.sin(theta)
        x0 = a * rho
        y0 = b * rho
        pt1 = (int(x0 + 1000 * (-b)), int(y0 + 1000 * (a)))
```

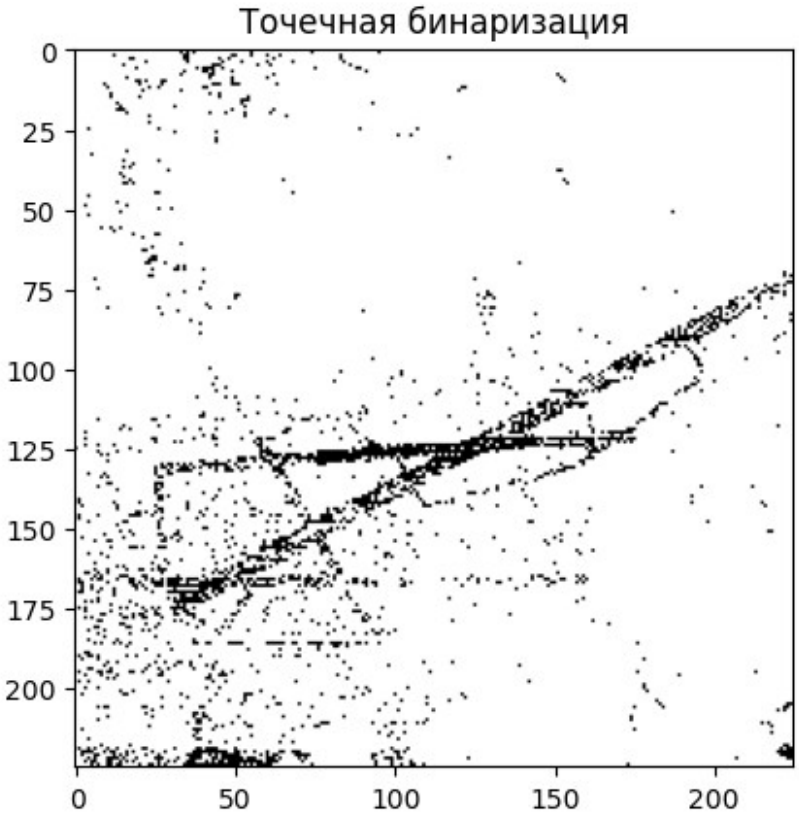
```
        pt2 = (int(x0 - 1000 * (-b)), int(y0 - 1000 * (a)))
        length = np.sqrt((pt1[0] - pt2[0])**2 + (pt1[1] - pt2[1])**2)
        if length > max_length:
            max_length = length
            longest_line = (pt1, pt2)
    if longest_line:
        cv2.line(image_line, longest_line[0], longest_line[1], (0, 0,
255), 7, cv2.LINE_AA)
else:
    print("Линии не обнаружены.")
print(f'Максимальная длина {max_length}')
plt.imshow(image_line)
plt.title("Наиболее протяженный участок")
plt.show()
```

```
Максимальная длина 1998.7668698475068
```

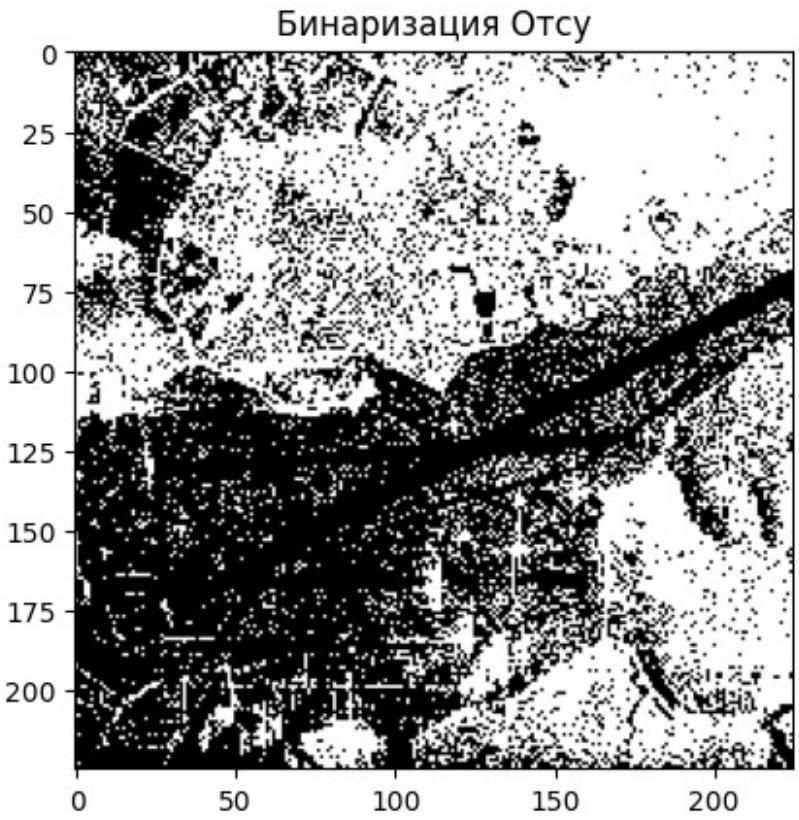



```
import copy
bin_img = copy.deepcopy(image_gray)
T  = 50
bin_img[image_gray < T] = 0
bin_img[image_gray >= T] = 255
```

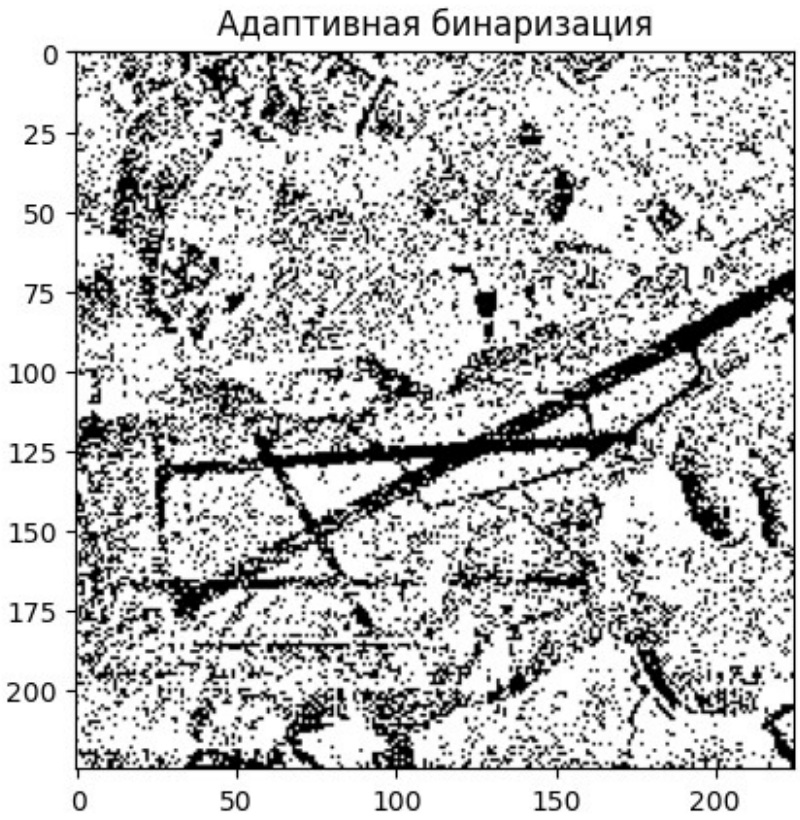
```
plt.imshow(bin_img, cmap="gray")
plt.title("Точечная бинаризация")
plt.show()
```



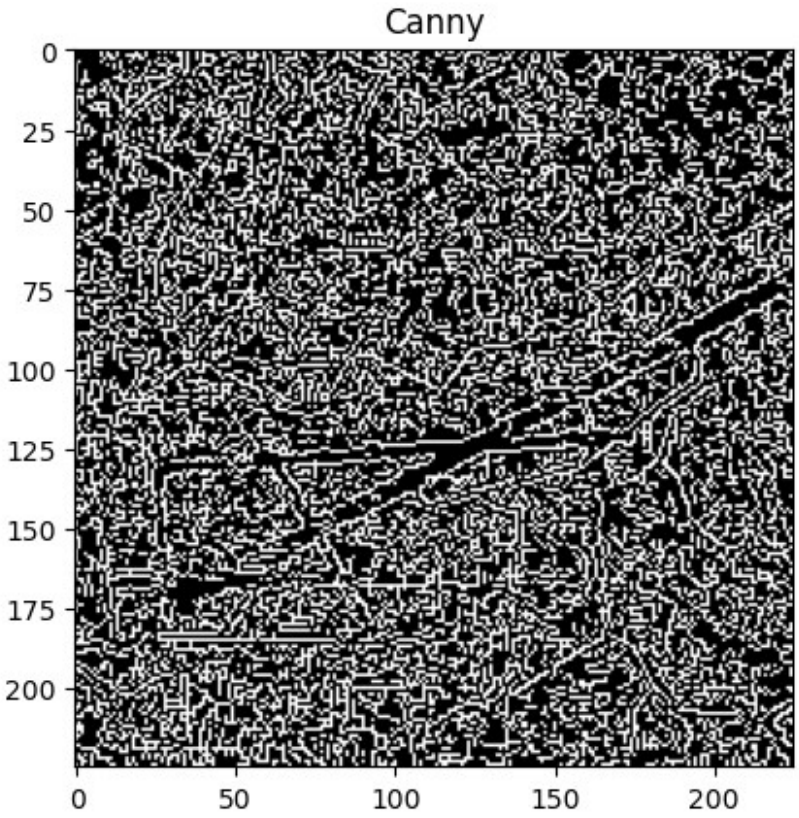

```
_,th2 =
cv2.threshold(image_gray,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

plt.imshow(th2, cmap = "gray")
plt.title("Бинаризация Отсу")
plt.show()
```




```
th3 = cv2.adaptiveThreshold(image_gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,71,21)

plt.imshow(th3, cmap = "gray")
plt.title("Адаптивная бинаризация")
plt.show()
```

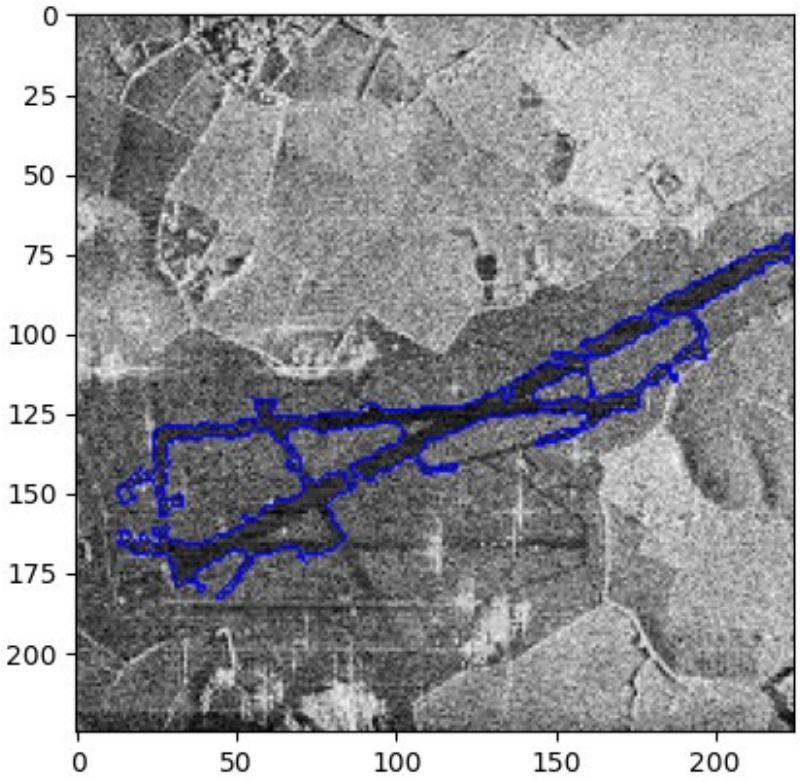



```
scale = 1
delta = 0
ddepth = cv2.CV_16S
grad_x = cv2.Sobel(image_gray, ddepth, 1, 0, ksize=3, scale=scale,
delta=delta, borderType=cv2.BORDER_DEFAULT)
grad_y = cv2.Sobel(image_gray, ddepth, 0, 1, ksize=3, scale=scale,
delta=delta, borderType=cv2.BORDER_DEFAULT)
grad = cv2.addWeighted(grad_x, 0.5, grad_y, 0.5, 0.0)

fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow((grad_x - grad_x.min()) * 255, cmap="gray")
axs[0].set_title("Оператор Собеля X")
axs[1].imshow((grad_y - grad_y.min()) * 255, cmap="gray")
axs[1].set_title("Оператор Собеля Y")
axs[2].imshow((grad - grad.min()) * 255, cmap="gray")
axs[2].set_title("Оператор Собеля")
plt.tight_layout()
plt.show()

edges = cv2.Canny(image_gray,100,200)

plt.imshow(edges, cmap="gray")
plt.title("Canny")
plt.show()
```




```
blur = cv2.GaussianBlur(bin_img, (5, 5), 0)
thresh = cv2.adaptiveThreshold(blur, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 55, 2)
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
line_image = image.copy()
if contours:
    for contour in contours:
        area = cv2.contourArea(contour)
        if area > 500:
            cv2.drawContours(line_image, [contour], -1, (0, 0, 255),
1)
plt.imshow(line_image, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x8935670>
```


0
25
50
75
100
125
150
175
200
0
50
100
150
200