# Segmentação de Imagens com Python de A à Z: Técnicas Clássicas de Processamento Digital de Imagens

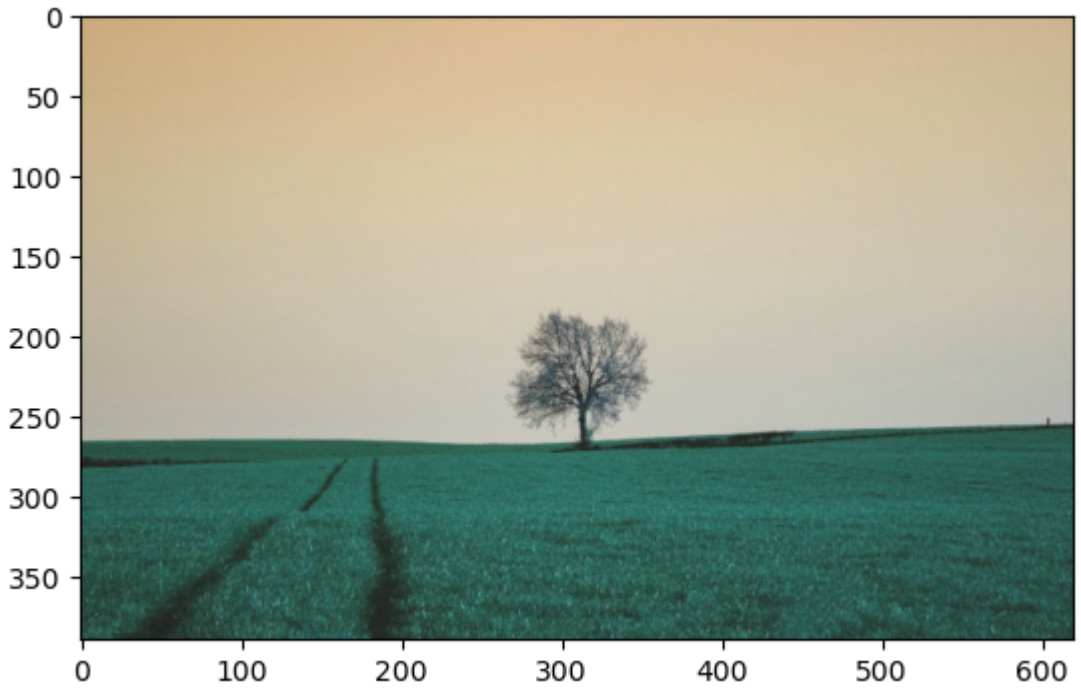
## Importação de bibliotecas

In [1]:
```
import cv2 # OpenCV
import numpy as np
from matplotlib import pyplot as plt
#from google.colab.patches import cv2_imshow #para google colab
```

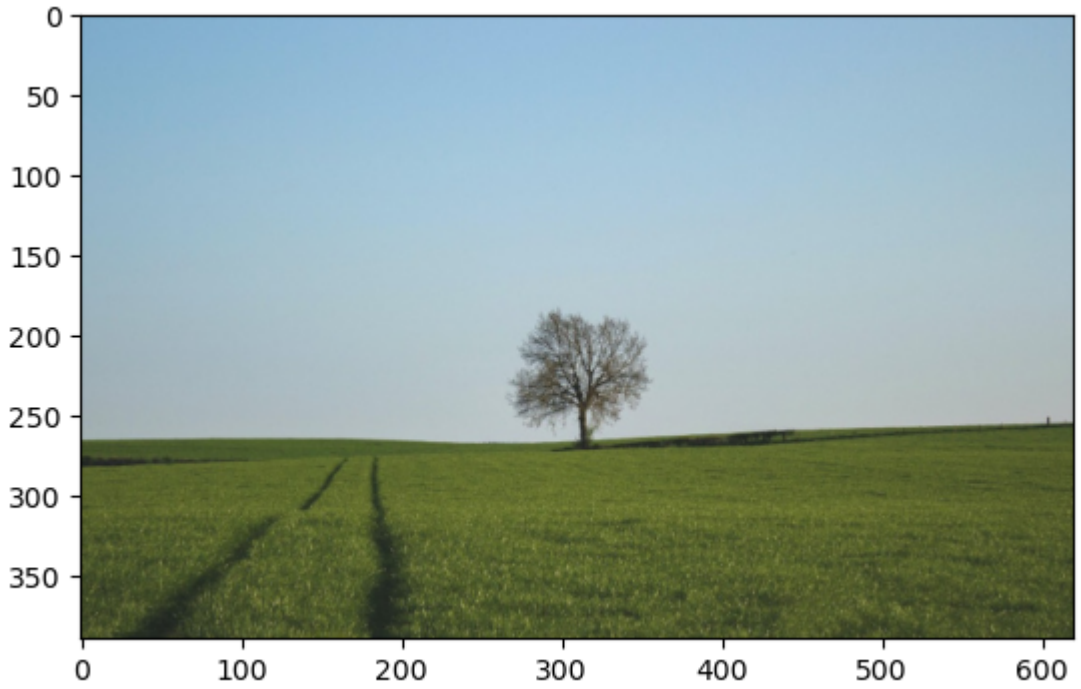
## Segmentação baseada em Limiarização (Thresholding)

### Limiarização Global (Global Thresholding)

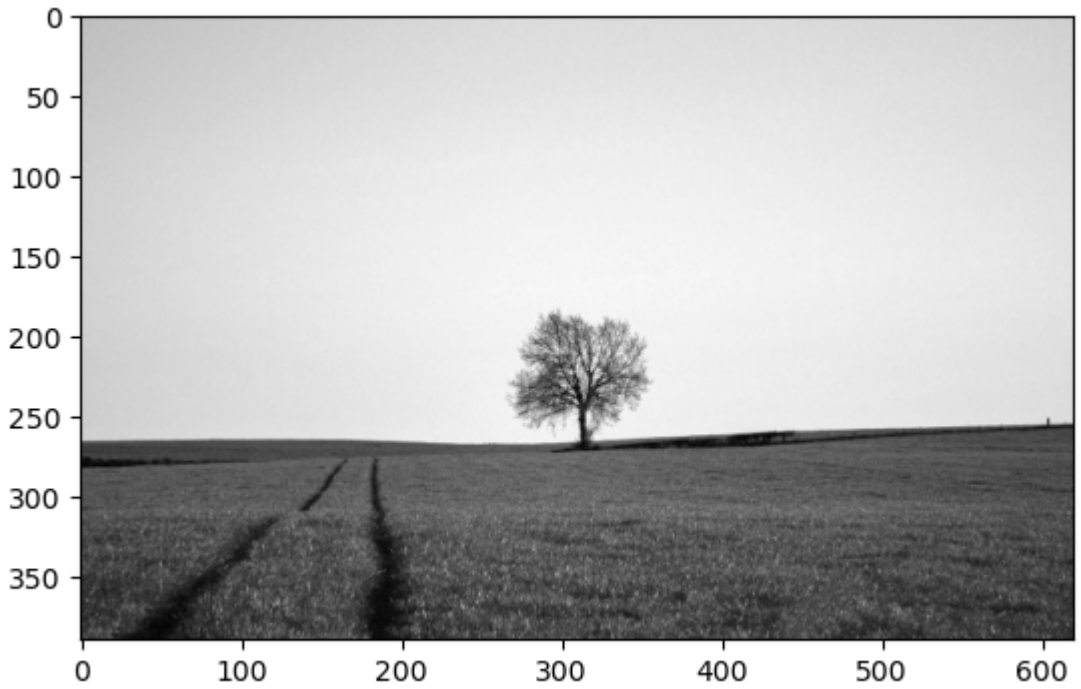
In [2]:
```
img = cv2.imread('./materials/imagens/paisagem01.jpg')
plt.imshow(img);
```




In [3]:
```
rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(rgb);
```




In [4]:
```
gray = cv2.cvtColor(rgb, cv2.COLOR_RGB2GRAY)
plt.imshow(gray, cmap='gray');
```



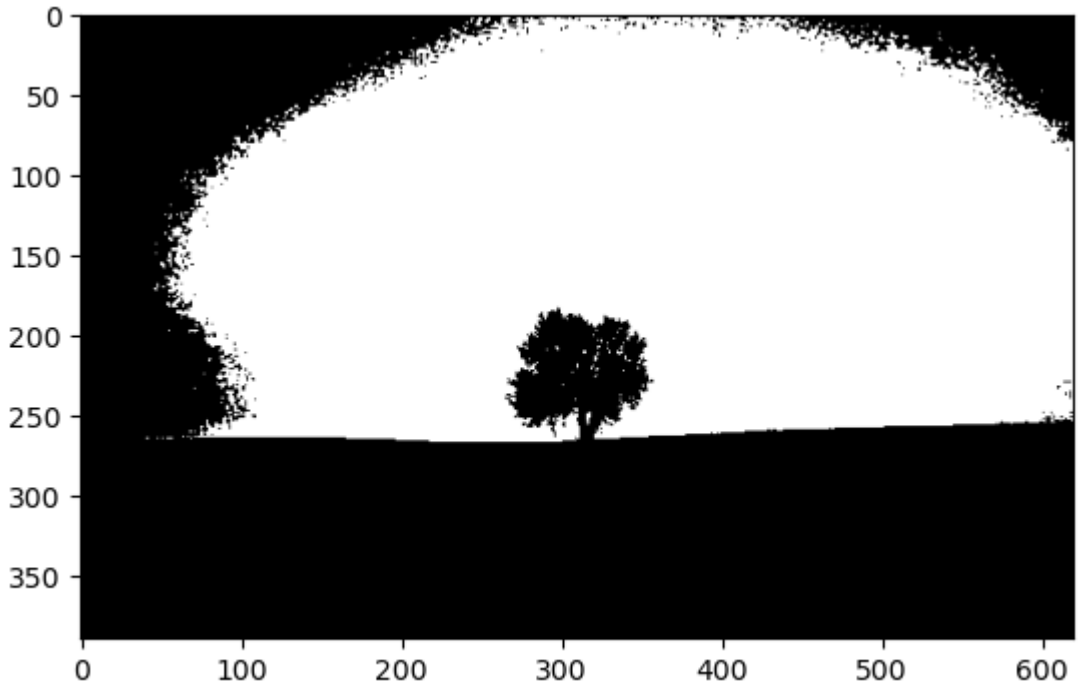

In [5]:
```
limiar = 180 # 0 - 255
```

In [6]:
```
val, thresh = cv2.threshold(gray,limiar, 255, cv2.THRESH_BINARY)
```

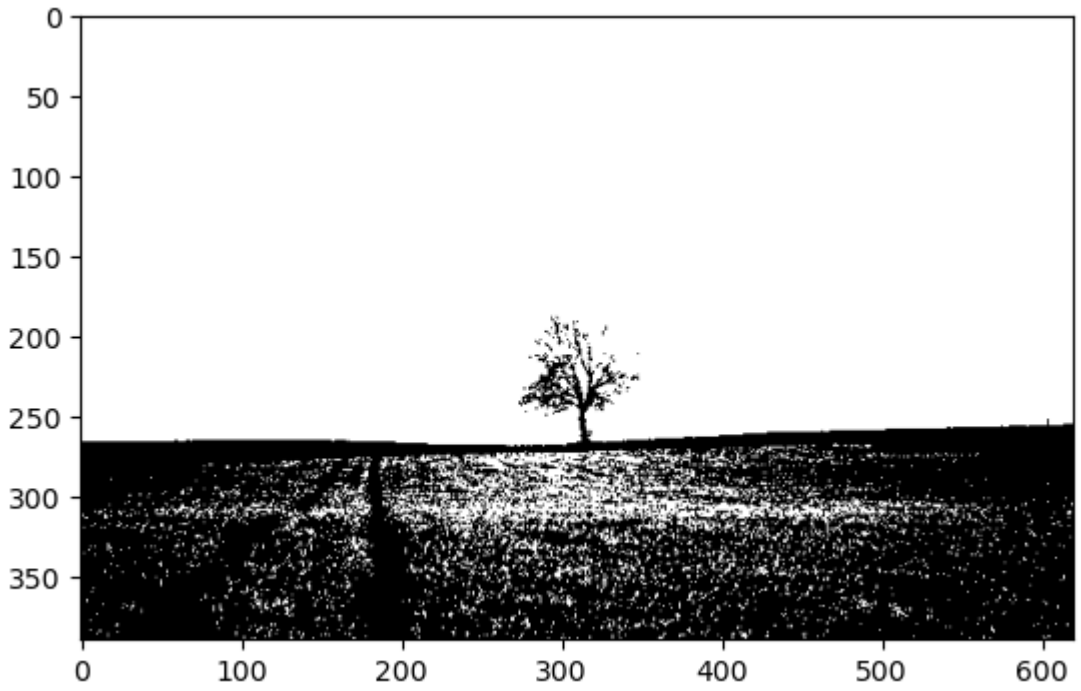
In [7]:
```
val
```

Out[7]:
```
180.0
```

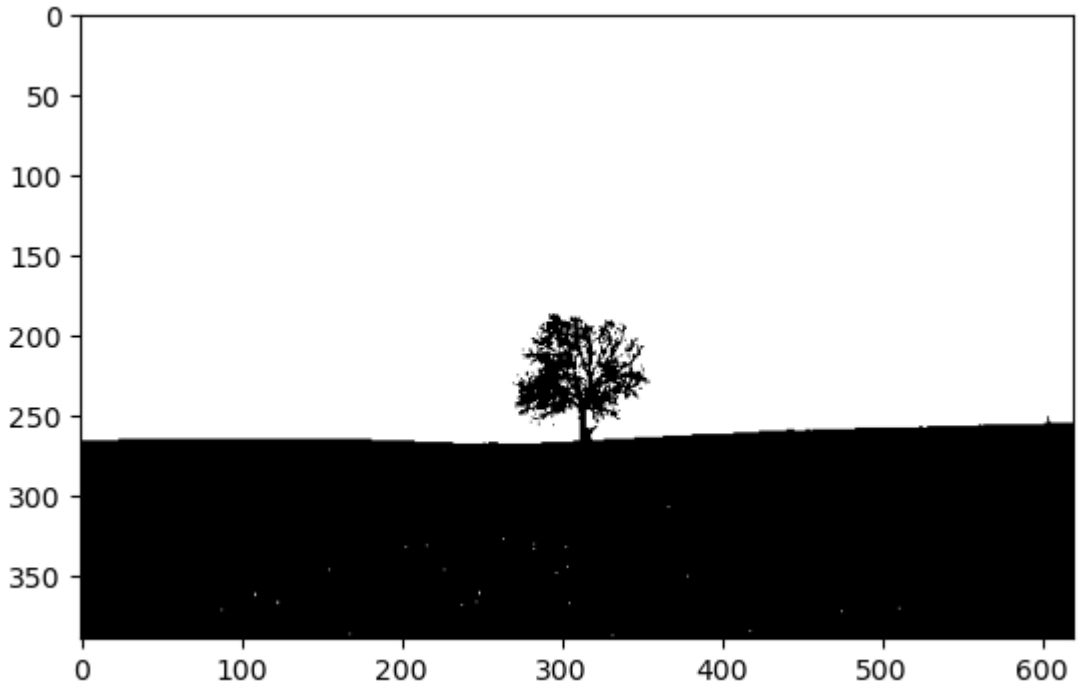
In [8]:
```
plt.imshow(thresh, cmap='gray');
```




In [9]:
```
limiar = 100 # 0 - 255
val, thresh = cv2.threshold(gray,limiar, 255, cv2.THRESH_BINARY)
plt.imshow(thresh, cmap='gray');
```




In [10]:
```
limiar = 140 #@param {type: "slider", min: 0, max:255, step: 1}
val, thresh = cv2.threshold(gray,limiar, 255, cv2.THRESH_BINARY)
plt.imshow(thresh, cmap='gray');
```




In [11]:
```
fig = plt.gcf()
fig.set_size_inches(18,6)
plt.imshow(thresh, cmap='gray')
plt.axis('off')
plt.show()
```

In [12]:
```
def mostrar (imagem):
    fig = plt.gcf()
    fig.set_size_inches(18,6)
    plt.imshow(imagem, cmap='gray')
```

```
    plt.axis('off')
    plt.show()
```

In [13]:
```
mostrar(thresh)
```

## Salvando o resultado em um arquivo de imagem

In [14]:
```
cv2.imwrite("./materials/resultados_bento/resultado_threshold.jpg", thresh)
```

Out[14]:
```
True
```

## Tipos de Limiarização

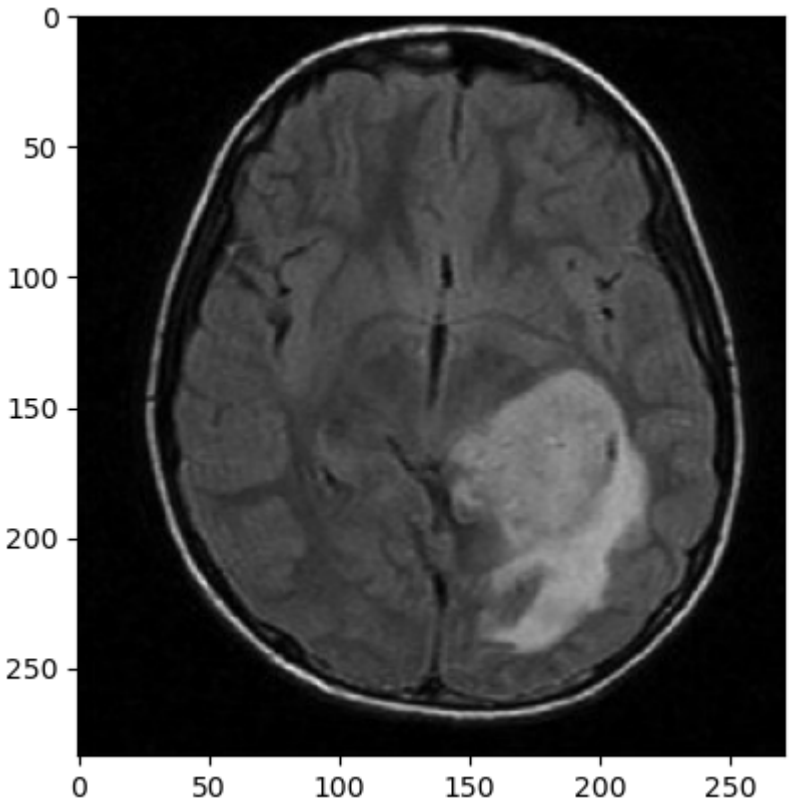
In [15]:
```
#_, thresh = cv2.threshold(gray, limiar, 255, cv2.THRESH_BINARY)
#_, thresh = cv2.threshold(gray, limiar, 255, cv2.THRESH_BINARY_INV)
#_, thresh = cv2.threshold(gray, limiar, 255, cv2.THRESH_TRUNC)
#_, thresh = cv2.threshold(gray, limiar, 255, cv2.THRESH_TOZERO)
_, thresh = cv2.threshold(gray, limiar, 255, cv2.THRESH_TOZERO_INV)

mostrar(thresh)
```

In [16]:
```
def exibir_limearizacao (img, limiar = 127):
    _, thresh_binary = cv2.threshold(gray, limiar, 255, cv2.THRESH_BINARY)
    _, thresh_binary_inv = cv2.threshold(gray, limiar, 255, cv2.THRESH_BINARY_INV)
    _, thresh_trunc = cv2.threshold(gray, limiar, 255, cv2.THRESH_TRUNC)
    _, thresh_to_zero = cv2.threshold(gray, limiar, 255, cv2.THRESH_TOZERO)
    _, thresh_to_zero_inv = cv2.threshold(gray, limiar, 255, cv2.THRESH_TOZERO_INV)

    titulos = ["Imagem original","Binary", "Binary inv", "Trunc", "To zero", "To zero
    imagens = [img, thresh_binary, thresh_binary_inv, thresh_trunc, thresh_to_zero, th

    fig = plt.gcf()
    fig.set_size_inches(18,12)

    for i in range(6):
        plt.subplot(2, 3, i+1)

        plt.imshow(cv2.cvtColor(imagens[i], cv2.COLOR_BGR2RGB), cmap='gray')
        plt.title(titulos[i])
        plt.xticks([]), plt.yticks([])

    plt.show()
```

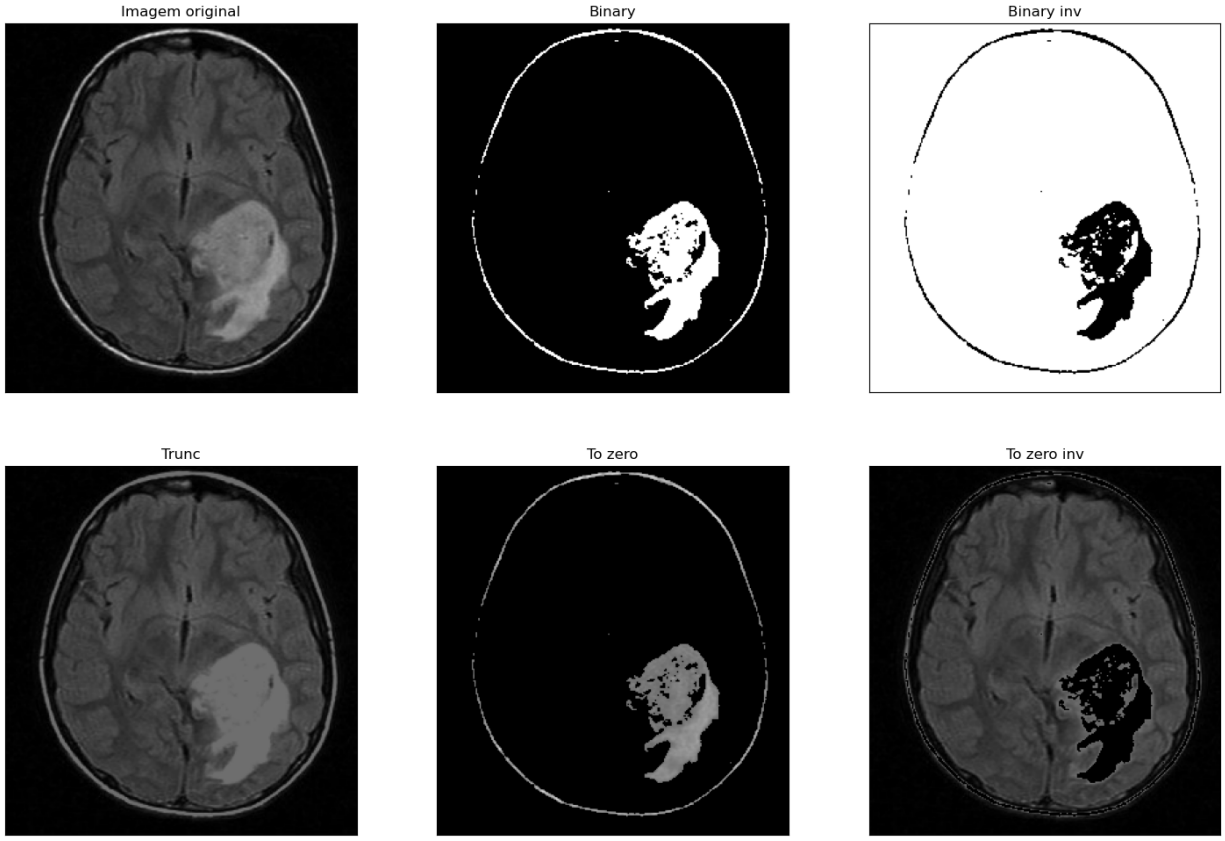
In [17]:
```
exibir_limearizacao(gray, limiar)
```


## Exemplo com tomografia computadorizada

In [18]:
```
img = cv2.imread('./materials/imagens/ct-scan.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray');
```

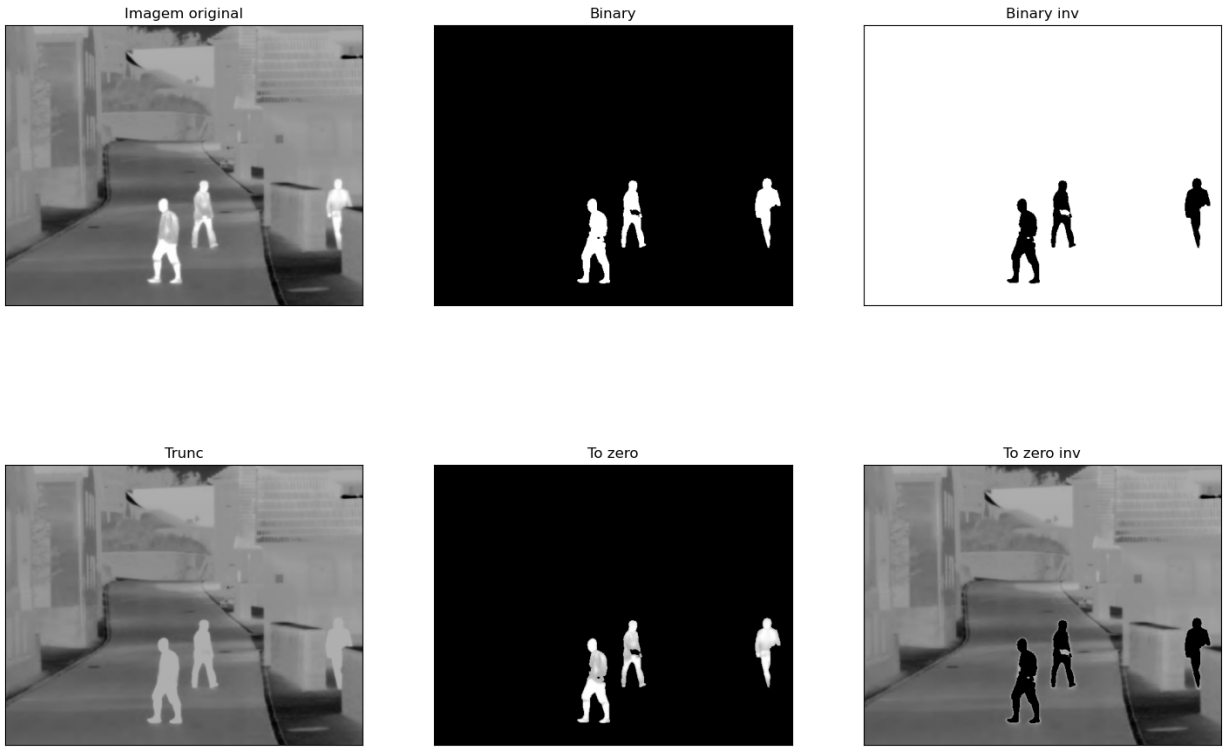



In [19]:
```
limiar = 110
exibir_limearizacao(gray, limiar)
```

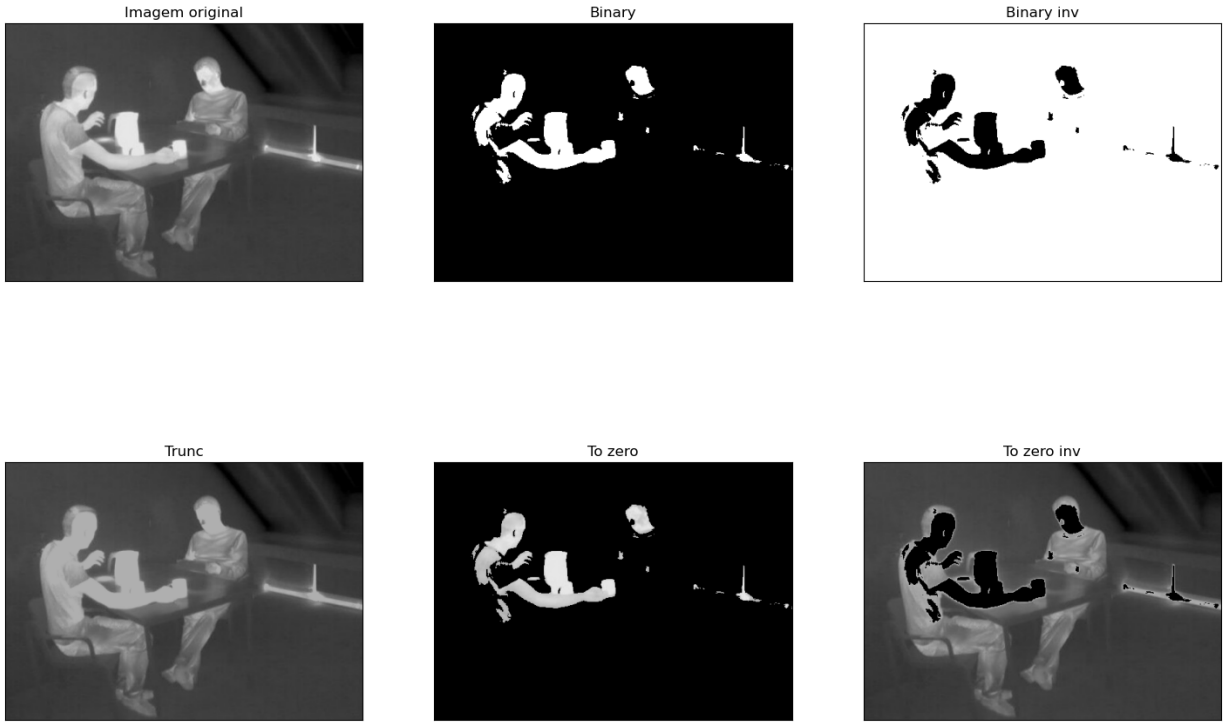



## Exemplo com imagem térmica infravermelha

In [20]:
```
img = cv2.imread('./materials/imagens/thermal01.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray');
limiar = 175
exibir_limearizacao(gray, limiar)
```

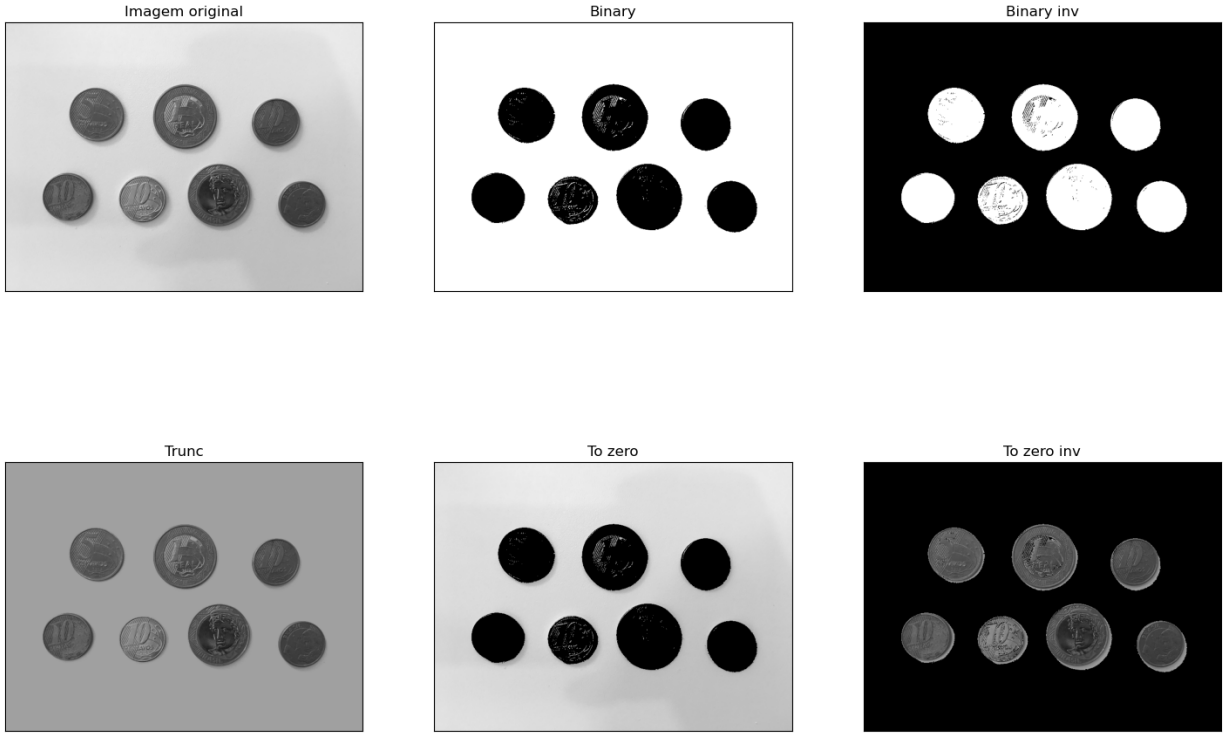


In [21]:
```
img = cv2.imread('./materials/imagens/thermal02.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray');
limiar = 175
exibir_limearizacao(gray, limiar)
```



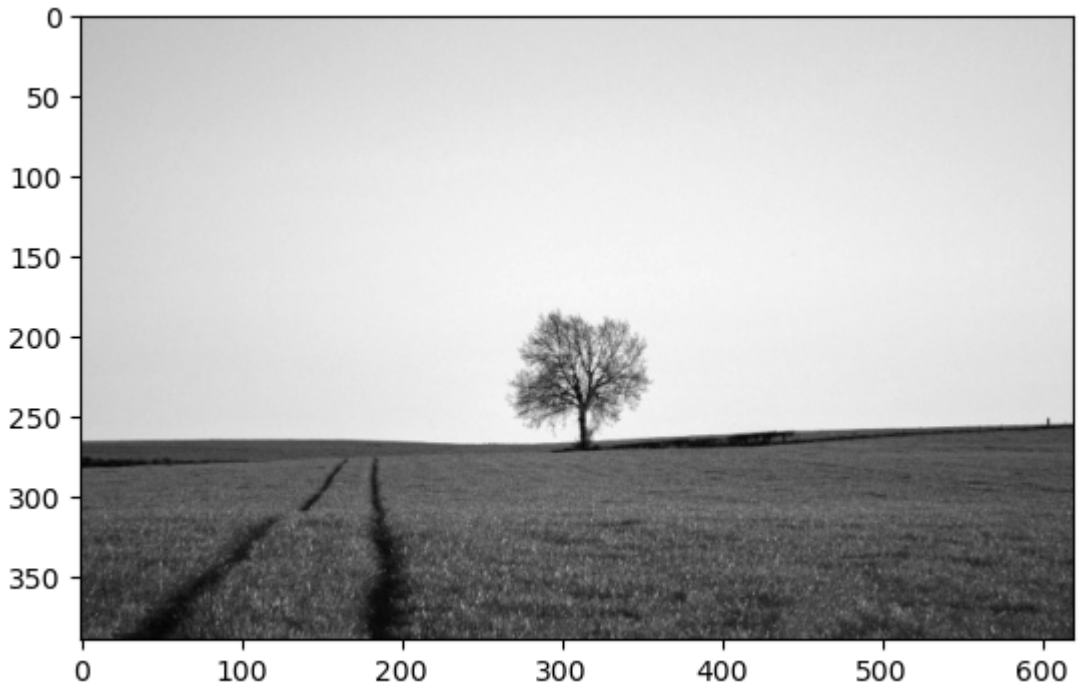

## Exemplo de separação do fundo do objeto

In [22]:
```
img = cv2.imread('./materials/imagens/moedas01.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray');
limiar = 160
exibir_limearizacao(gray, limiar)
```




# Método de Otsu (Otsu's method)

In [23]:
```
img = cv2.imread('./materials/imagens/paisagem01.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap='gray');
```




In [24]:
```
valor, otsu = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2. THRESH_OTSU)
print("Valor de limiar Otsu: ", valor)
```

```
Valor de limiar Otsu:  136.0
```

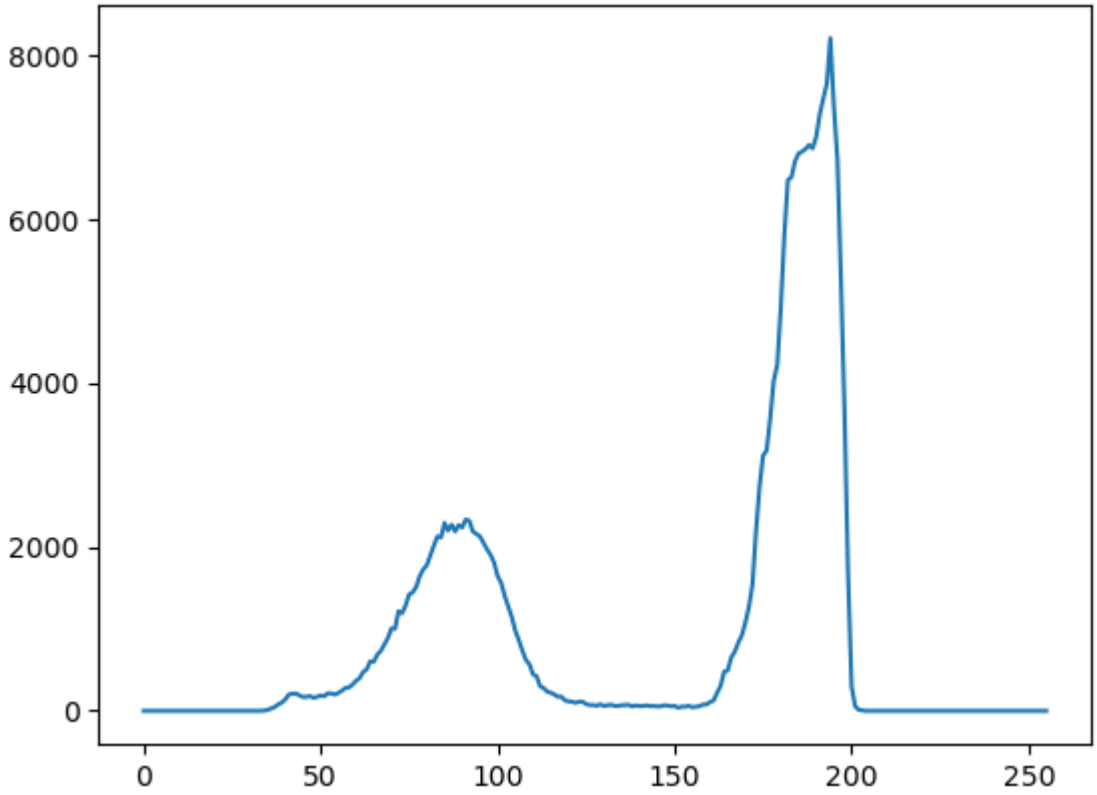
In [25]:
```
mostrar(otsu)
```

## Histogramas

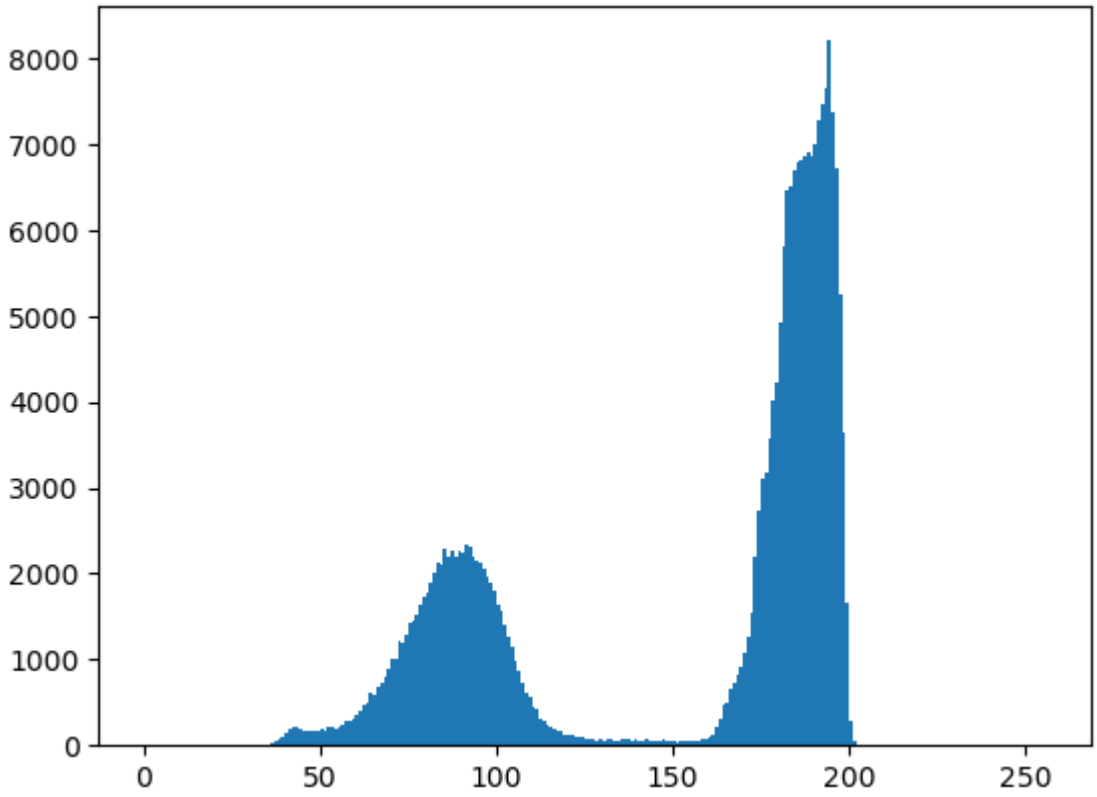
In [26]:
```
histograma, bins = np.histogram(gray, 256, [0,256])
histograma
```

Out[26]:
```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    1,
          0,    3,   11,   29,   46,   78,  100,  139,  198,  209,  210,
        188,  168,  170,  180,  158,  171,  187,  177,  213,  212,  198,
        218,  244,  279,  285,  319,  366,  399,  474,  504,  603,  599,
        685,  731,  807,  888, 1005, 1000, 1215, 1190, 1283, 1420, 1443,
       1513, 1643, 1726, 1775, 1897, 2017, 2128, 2114, 2292, 2198, 2270,
       2188, 2265, 2235, 2333, 2317, 2189, 2156, 2128, 2048, 1970, 1902,
       1808, 1643, 1562, 1402, 1271, 1145,  976,  862,  732,  620,  567,
        444,  423,  299,  279,  234,  221,  200,  170,  174,  128,  115,
        112,   97,  111,  111,   82,   70,   70,   62,   79,   60,   68,
         74,   62,   59,   68,   70,   73,   54,   65,   58,   59,   67,
         57,   60,   54,   55,   67,   64,   55,   60,   36,   51,   50,
         61,   43,   50,   56,   77,   78,  107,  125,  219,  305,  477,
        492,  651,  720,  832,  921, 1078, 1267, 1539, 2191, 2735, 3112,
       3176, 3564, 4025, 4216, 4918, 5802, 6474, 6506, 6699, 6796, 6819,
       6852, 6901, 6859, 7001, 7272, 7465, 7645, 8205, 7367, 6727, 5241,
       3654, 1666,  296,   57,   12,    6,    0,    1,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0], dtype=int64)
```

In [27]:
```
plt.plot(histograma);
```



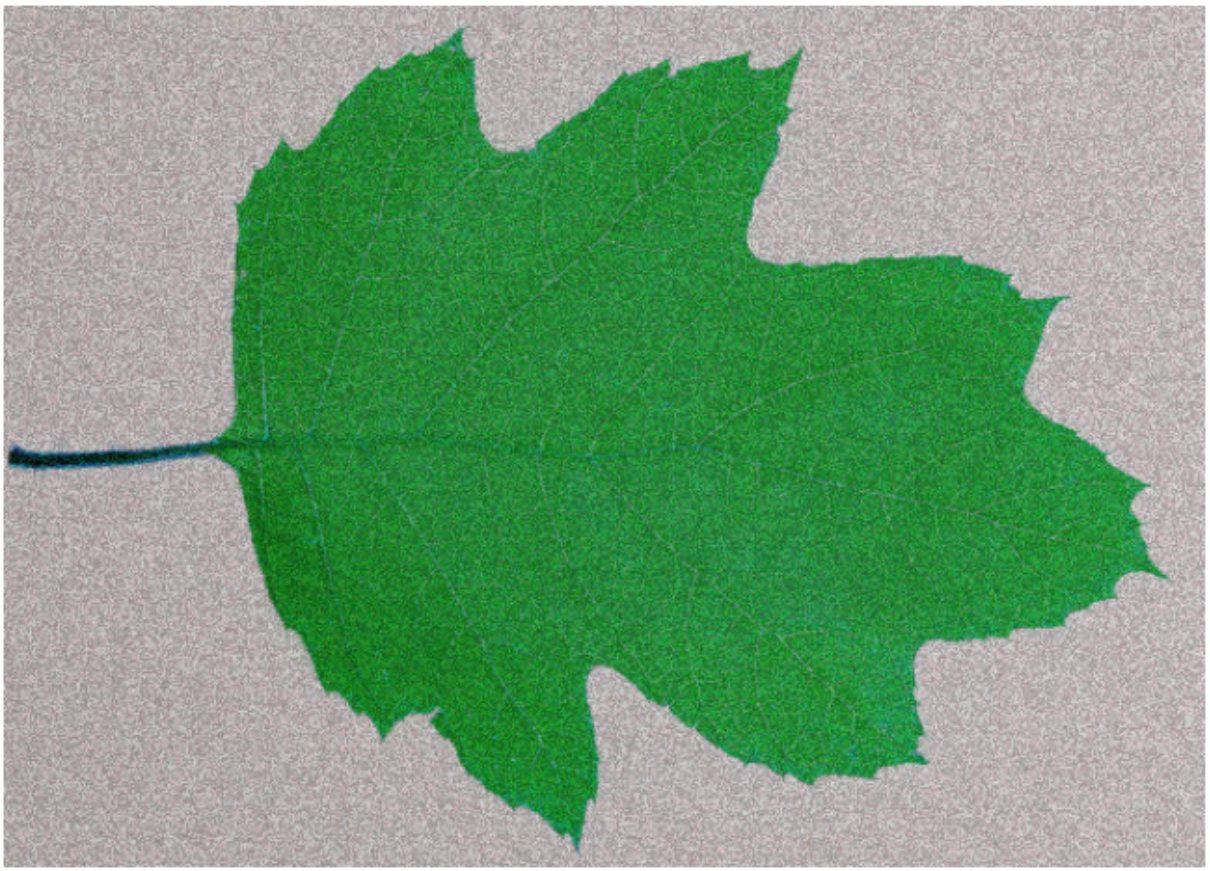

In [28]:
```
plt.hist(gray.ravel(), 256,[0,256]);
```

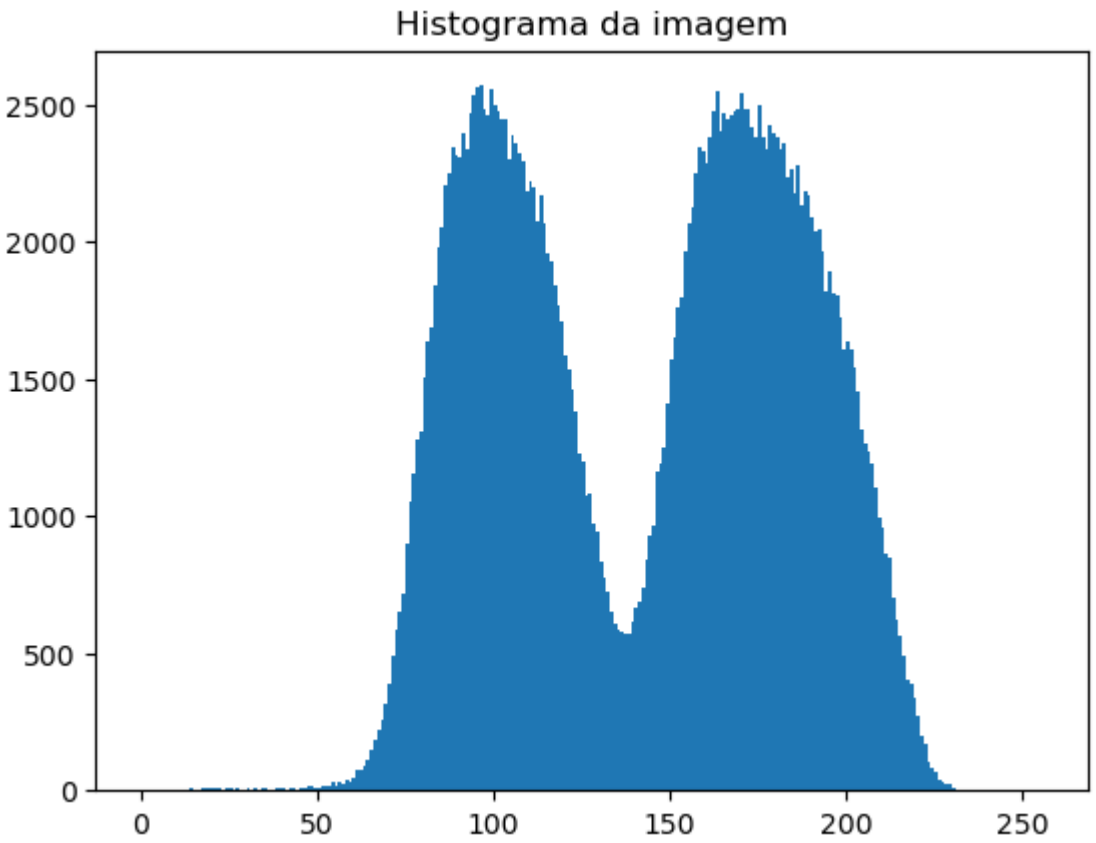



## Melhorando a segmentação em imagens com ruídos

In [29]:
```
img = cv2.imread('./materials/imagens/folha_ruido.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
mostrar(img)
plt.hist(gray.ravel(), 256, [0,256])
plt.title("Histograma da imagem")
plt.show()
```

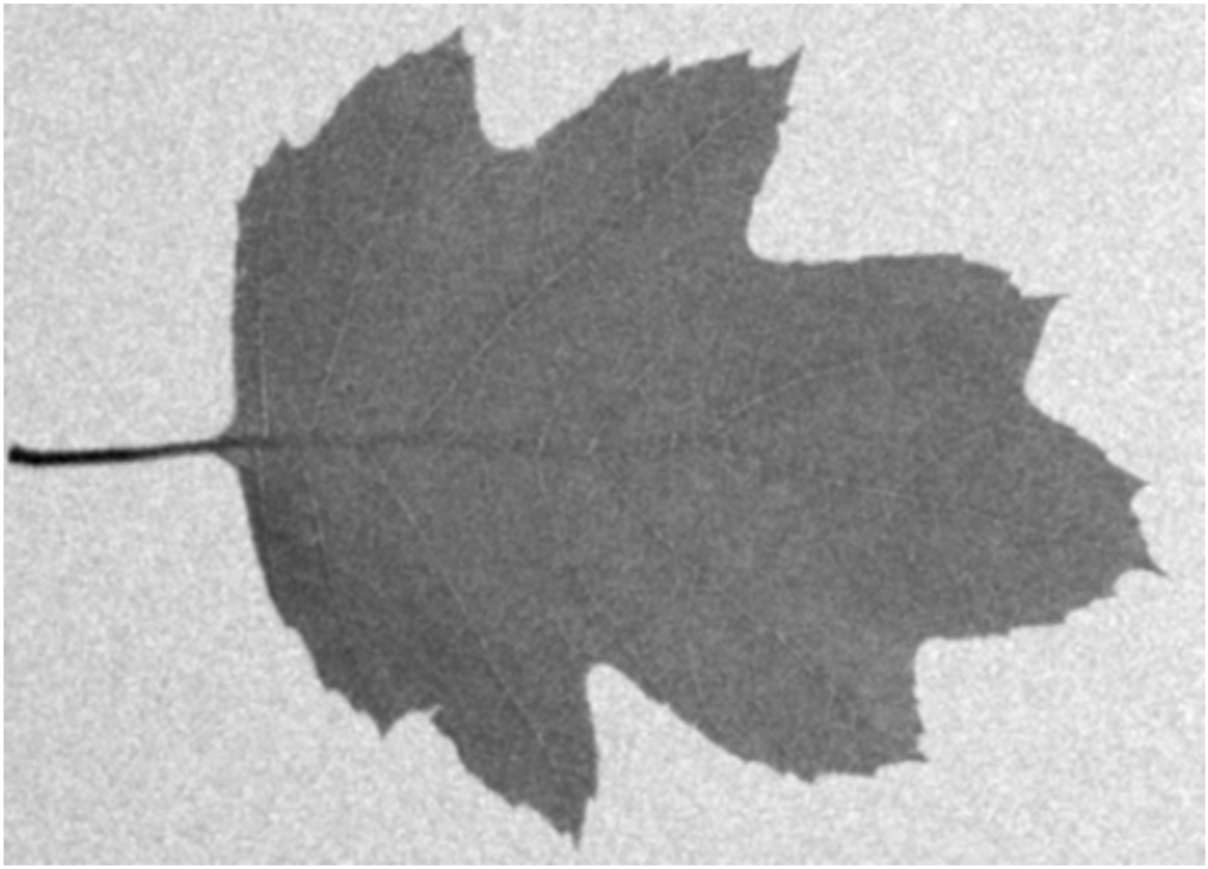




In [30]:
```
valor, otsu = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
print("valor do limiar", valor)
mostrar(otsu)
```

```
valor do limiar 139.0
```

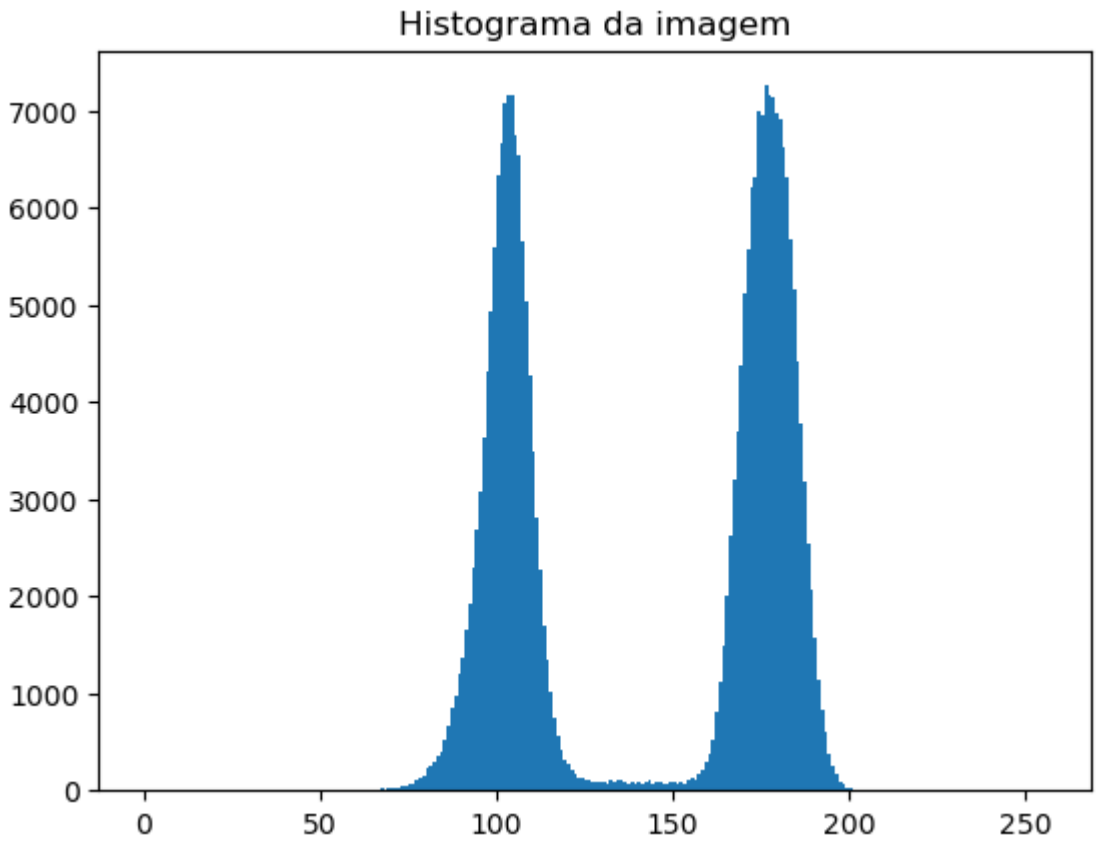
In [31]:
```
desfoque = cv2.GaussianBlur(gray, (5,5), 0)
mostrar(desfoque)
```



In [33]:
```
valor, otsu = cv2.threshold(desfoque, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
print("valor do limiar", valor)
mostrar(otsu)
```

```
valor do limiar 139.0
```

In [34]:
```
plt.hist(desfoque.ravel(), 256, [0,256])
plt.title("Histograma da imagem")
plt.show()
```


Histograma da imagem
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250