

Atividade 2 - Alexandre de Taunay Voloch (12557532)

Para essa atividade vamos inicialmente usar a mesma imagem que utilizamos na última atividade. Vamos abri-la no Python e tbm ver ela no Jupyter

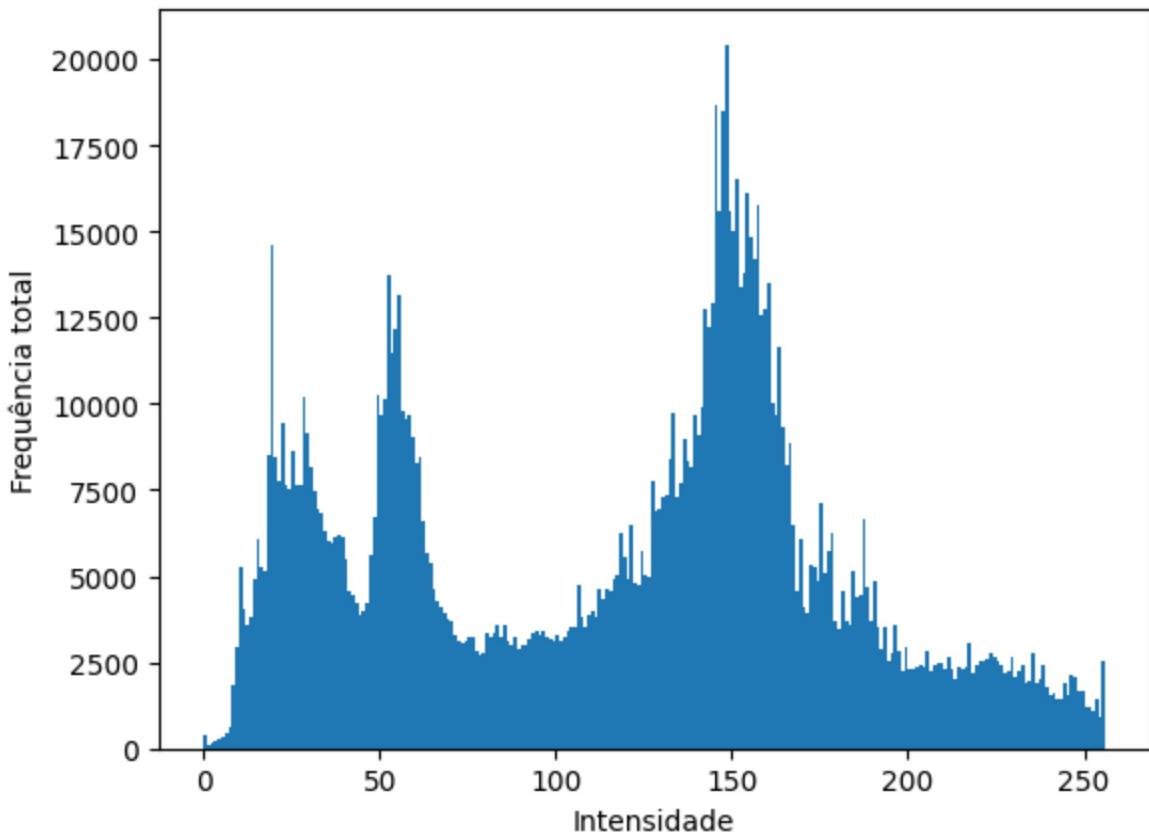
```
In [ ]: from PIL import Image  
  
imagem = Image.open("minha_imagem_grayscale.jpg")  
  
In [ ]: from IPython.display import Image, display  
  
display(Image('minha_imagem_grayscale.jpg', width=250))
```



1

Para fazer o histograma vamos usar o matplotlib

```
In [ ]: import numpy as np  
import matplotlib.pyplot as plt  
  
imagem_vetor = np.array(imagem)  
  
plt.hist(imagem_vetor.ravel(), bins=256, range=[0,256], histtype='bar')  
plt.xlabel("Intensidade")  
plt.ylabel("Frequência total")  
plt.show()
```



2

Agora para normalizar, poderíamos transformar usando o numpy, mas é um rolo, pois já existe uma biblioteca chamada opencv que tem funções para isso. Então vamos utilizá-la

```
In [ ]: import cv2
import matplotlib.pyplot as plt

imagem_cv2 = cv2.imread("minha_imagem_grayscale.jpg", cv2.IMREAD_GRAYSCALE)

# agora vamos normalizar o histograma - em inglês isso se chama equalize Histogram
imagem_normalizada = cv2.equalizeHist(imagem_cv2)

# agora vamos mostrar os dois histogramas e compará-los

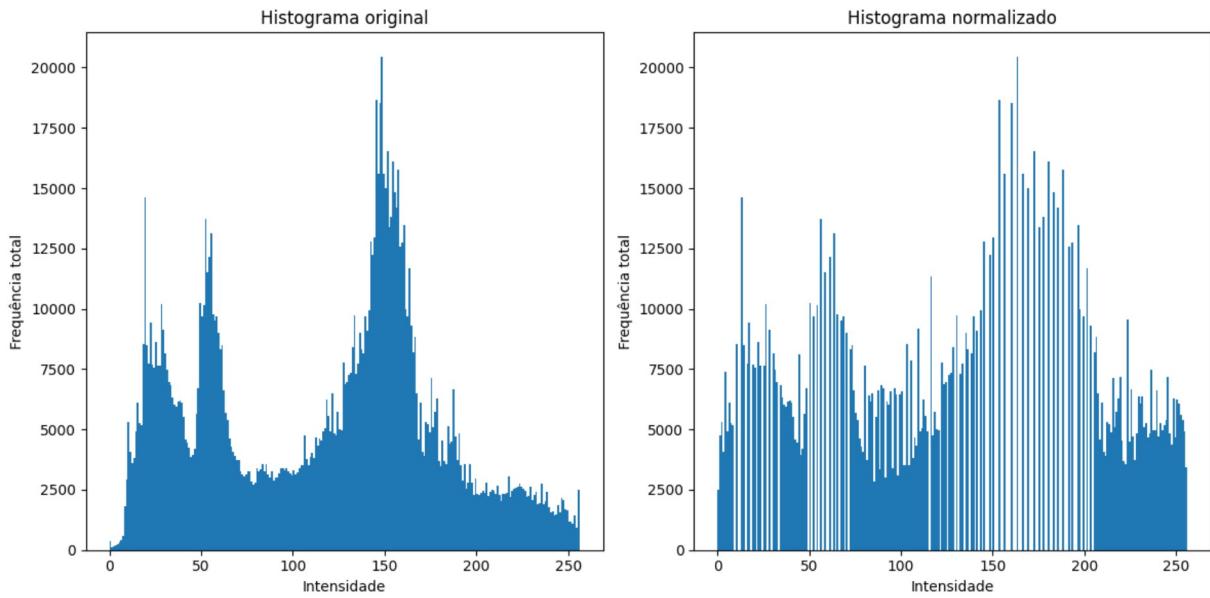
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.hist(imagem_cv2.ravel(), bins=256, range=[0,256], histtype='bar')
#plt.imshow(imagem_cv2, cmap='gray')
plt.xlabel("Intensidade")
plt.ylabel("Frequência total")
plt.title("Histograma original")

plt.subplot(1, 2, 2)
#plt.imshow(imagem_normalizada, cmap='gray')
```

```
plt.hist(imagem_normalizada.ravel(), bins=256, range=[0,256], histtype='bar')
plt.xlabel("Intensidade")
plt.ylabel("Frequência total")
plt.title("Histograma normalizado")

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



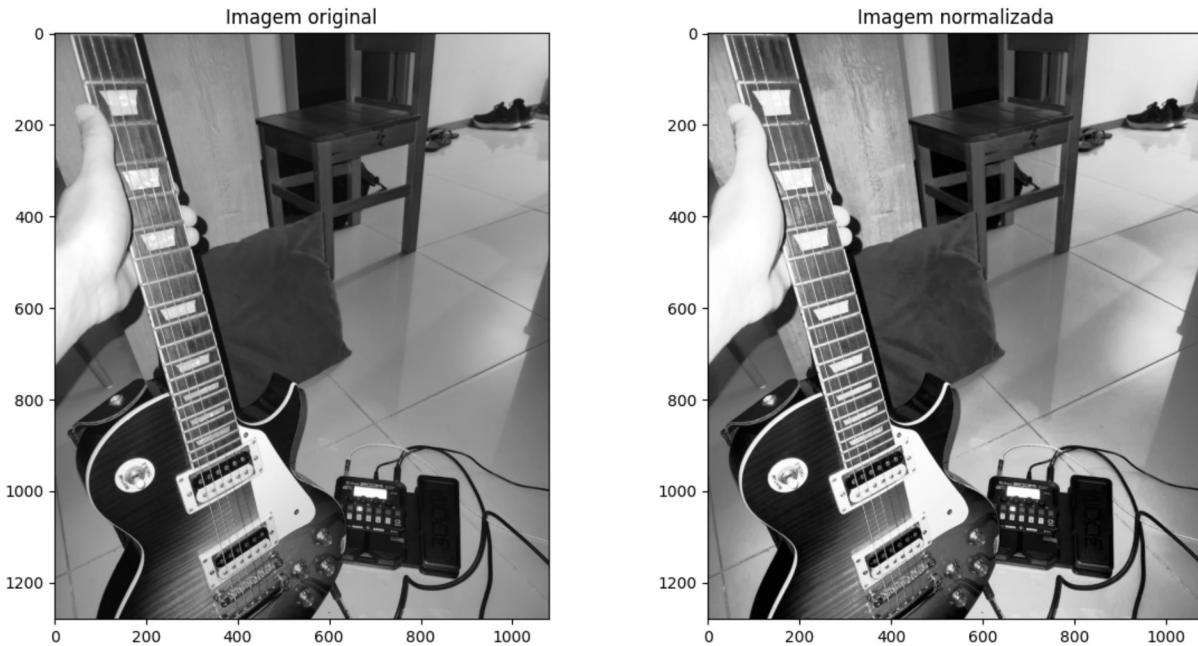
E agora vamos mostrar as duas imagens para compará-las, tbm usando o matplotlib.

```
In [ ]: plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(imagem_cv2, cmap='gray')
plt.title("Imagen original")

plt.subplot(1, 2, 2)
plt.imshow(imagem_normalizada, cmap='gray')
plt.title("Imagen normalizada")

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



Percebemos que o histograma e a imagem mudaram muito pouco. Ou seja, ela já estava razoavelmente normalizada antes. Isso faz sentido, porque a foto foi capturada usando uma câmera de celular, que de fato já tenta ajustar a imagem para ter o melhor histograma.

3

Agora, para escurecer a imagem, vamos usar uma simples transformação linear multiplicando todos os valores de intensidade por 0.5, assim efetivamente cortando a luminosidade pela metade.

```
In [ ]: # Como vamos mostrar várias imagens e histogramas Lado a Lado, vamos fazer uma função
def hist_imagem(imagem):
    plt.figure(figsize=(12,6))

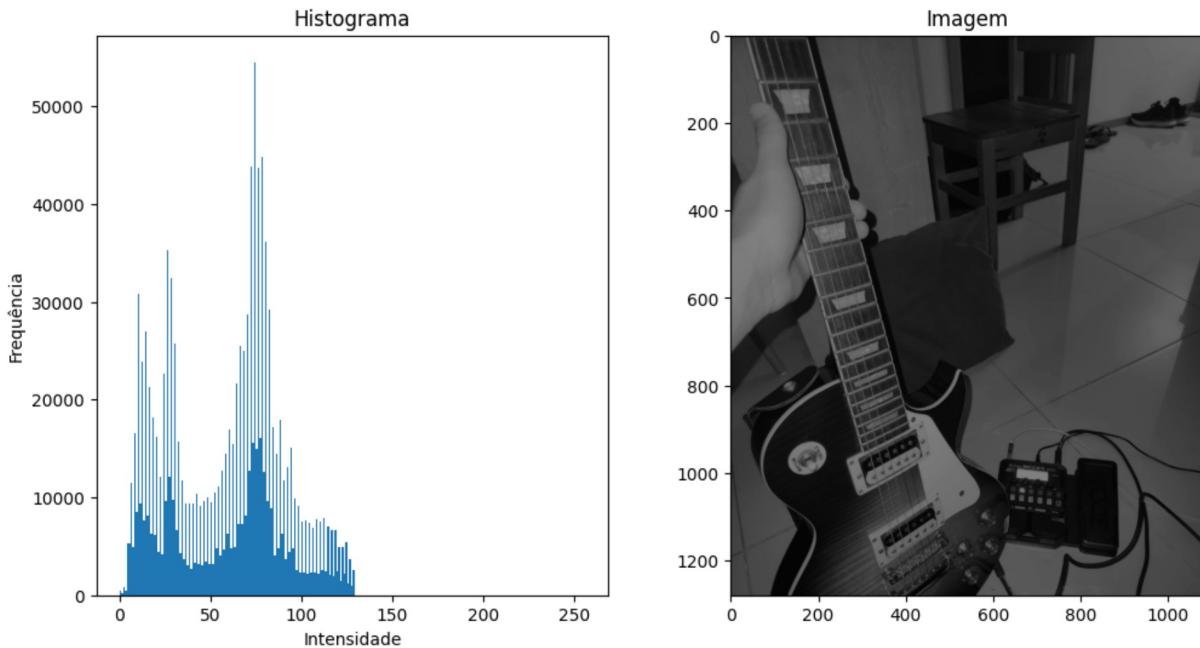
    plt.subplot(1,2,1)
    plt.hist(imagem.ravel(), bins=256, range=[0, 256], histtype='bar')
    plt.title('Histograma')
    plt.xlabel('Intensidade')
    plt.ylabel('Frequência')

    plt.subplot(1,2,2)
    # Perceba que precisamos manualmente colocar o vmin e vmax pro matplotlib. pois
    # ele percebe que o histograma "para" após 255*0.5, e portanto mostra a imagem
    # e portanto não muda nada na imagem em si
    plt.imshow(imagem, cmap='gray', vmin=0, vmax=255)
    plt.title("Imagen")

    plt.show()
```

```
In [ ]: imagem_escura = cv2.multiply(imagem_cv2, np.array([0.5]))

hist_imagem(imagem_escura)
```



4

Agora para reduzir o contraste podemos pensar num algoritmo que "puxa" os pixels para mais próximo de um valor médio da imagem, i.e.

$$I' = M + \alpha(I - M)$$

Onde M é o valor médio das intensidades da imagem:

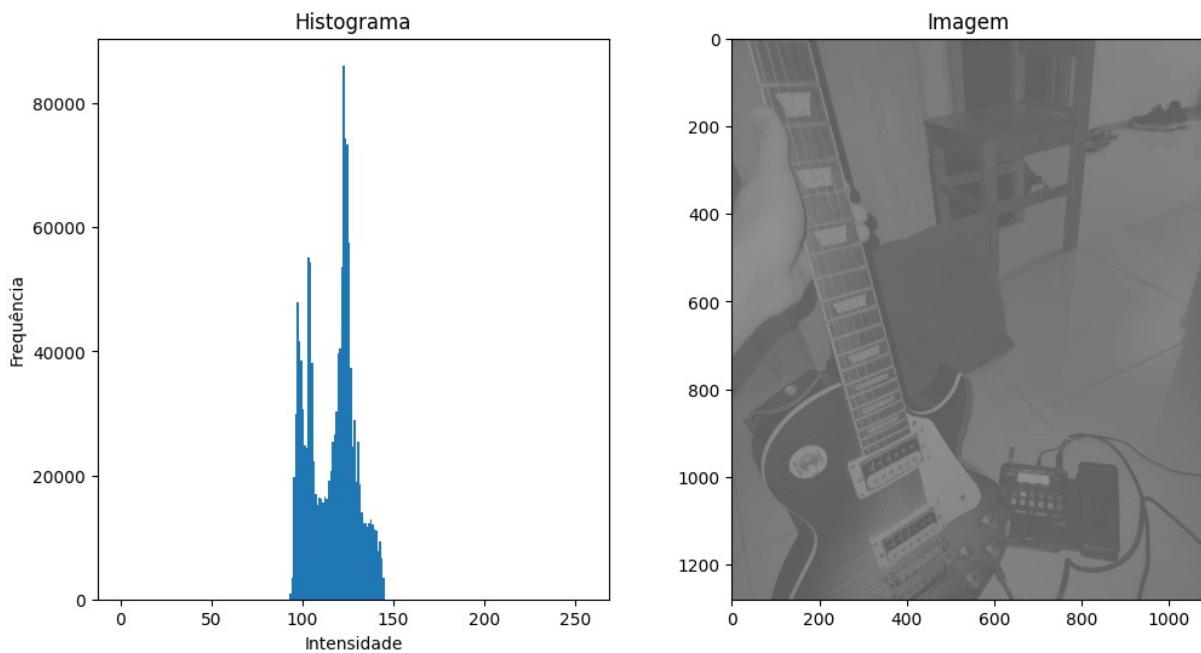
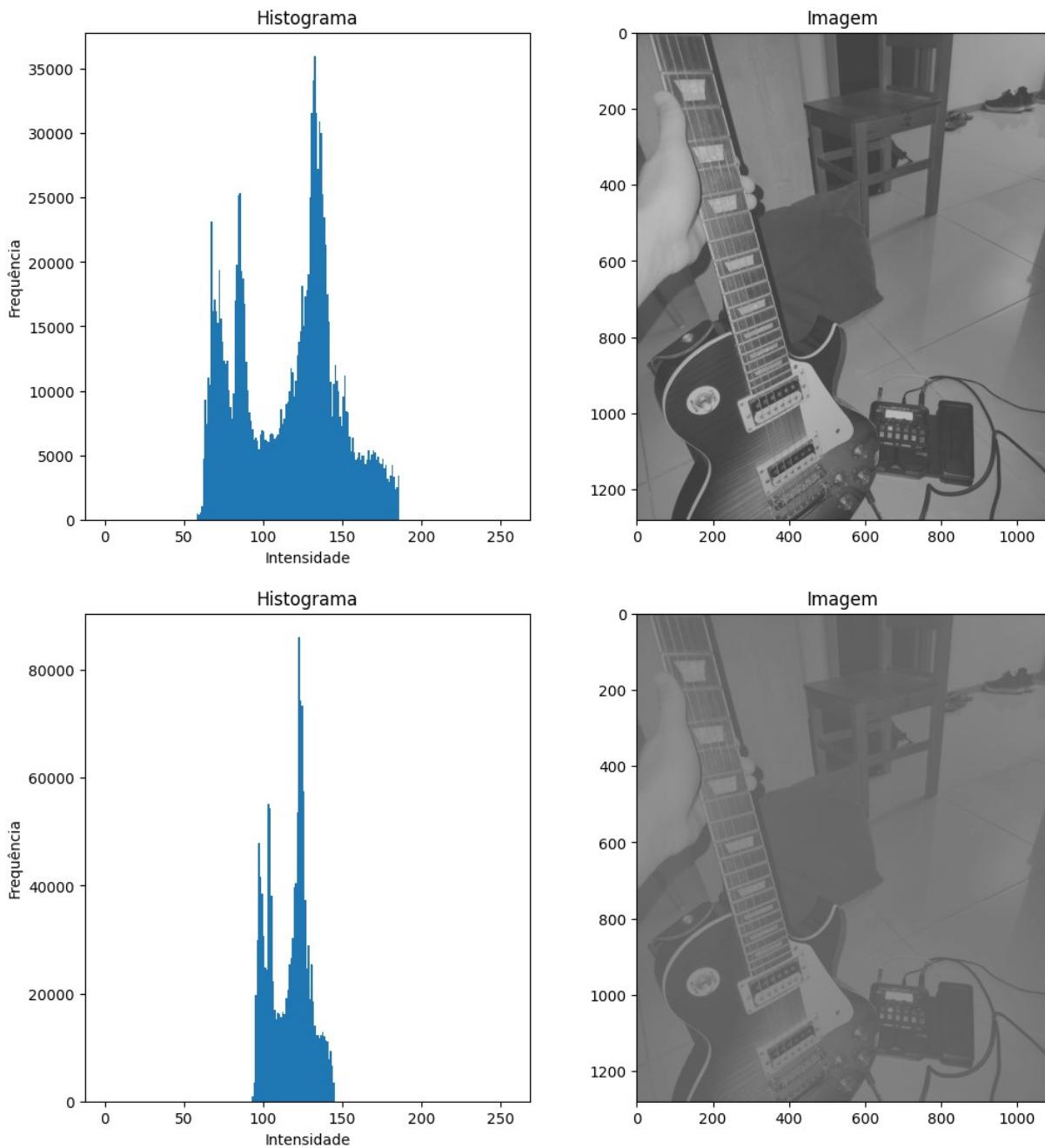
$$M = \sum_{n=1}^N \frac{I_n}{N}$$

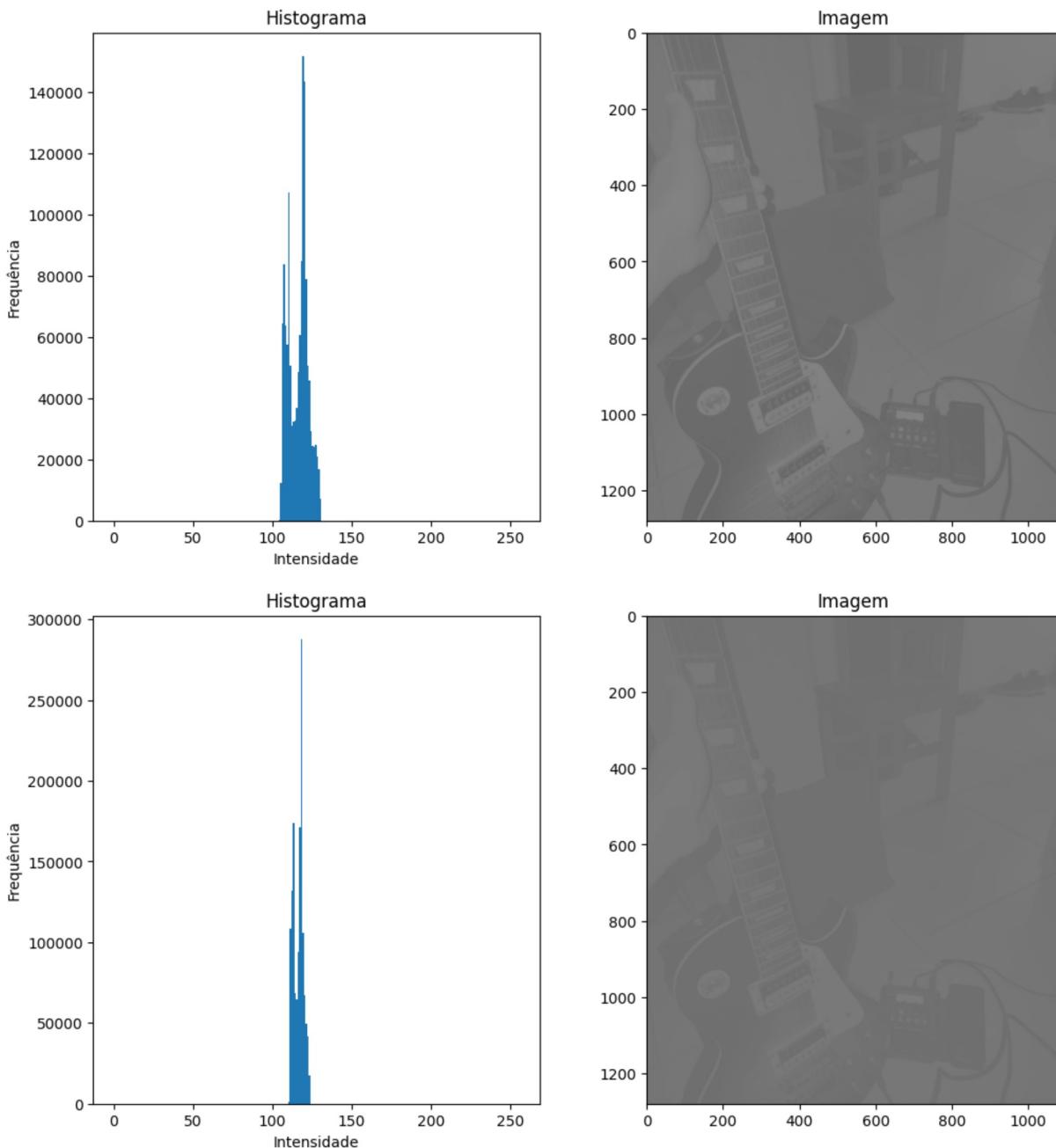
Vamos variar α entre 0.5, 0.2, 0.1, 0.05, só pra ver o efeito disso na imagem

```
In [ ]: media_imagem = np.mean(imagem_cv2)

for alpha in [0.5, 0.2, 0.1, 0.05]:
    imagem_nova = np.clip(media_imagem + alpha*(imagem_cv2 - media_imagem), 0, 255)
    # usamos clip para garantir que os pixels fiquem entre 0 e 255 (acima)

    hist_imagem(imagem_nova)
```



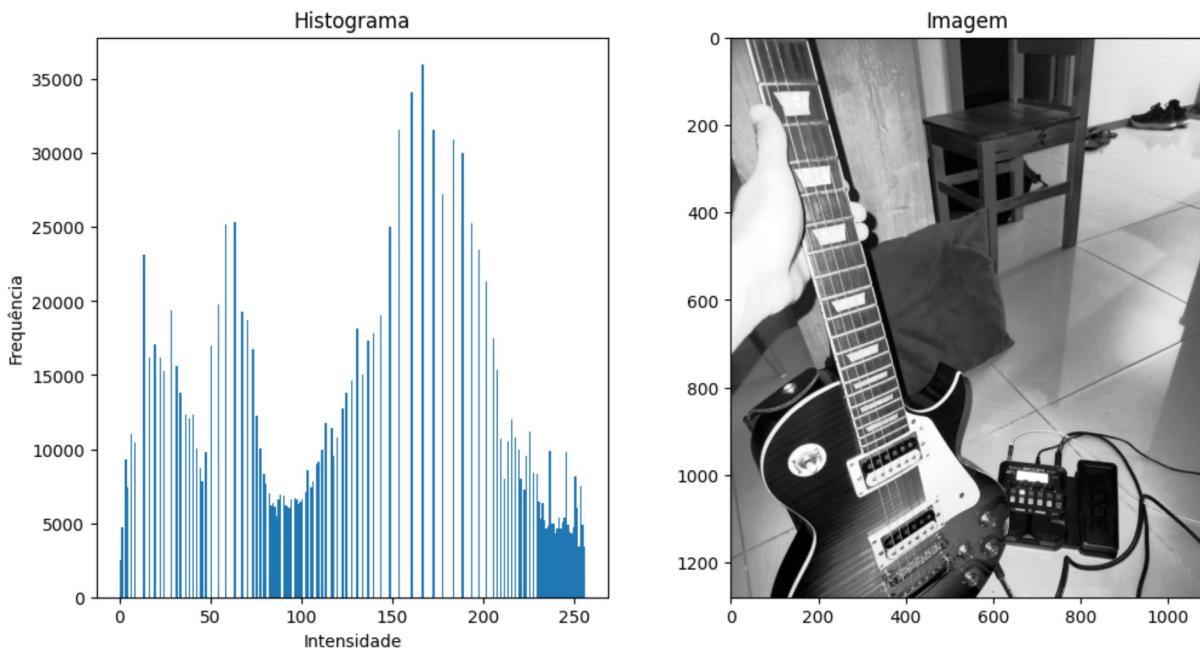


Como podemos ver ela vai ficando cada vez menos intensa e "visualizável".

5

Agora vamos pegar a imagem com $\alpha = 0.5$ e equalizá-la.

```
In [ ]: imagem_contraste = np.clip(media_imagem + 0.5*(imagem_cv2 - media_imagem), 0, 255).  
        imagem_c_normalizada = cv2.equalizeHist(imagem_contraste)  
        hist_imagem(imagem_c_normalizada)
```

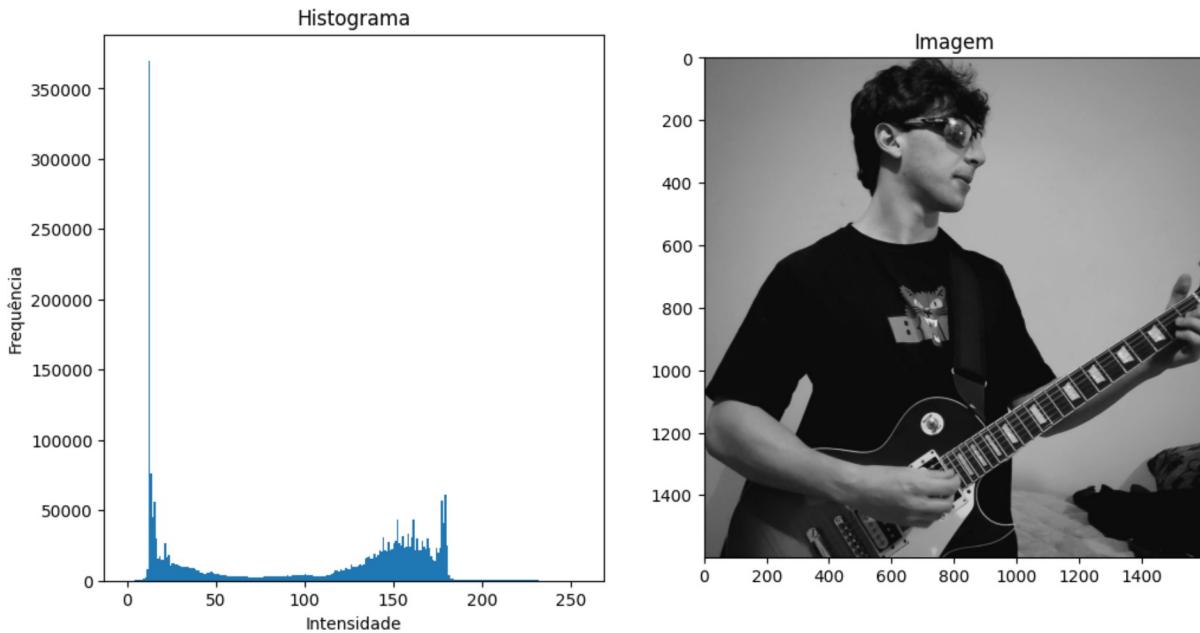


Percebemos que a imagem está novamente com "contraste" bom, embora agora ela tenha bem menos informação nas cores do que na imagem original - podemos ver isso no histograma, que está com mais "linhas" do que originalmente (onde as cores/intensidades eram bem mais uniformemente distribuídas.)

6

Agora vamos pegar uma nova imagem chamada "nova imagem.jpeg" e convertê-la para monocromática, e depois apresentá-la.

```
In [ ]: from PIL import Image  
nova_imagem = Image.open("nova_imagem.jpeg").convert("L")  
nova_imagem.save("nova_imagem_grayscale.jpg")  
  
nova_imagem_cv2 = cv2.imread("nova_imagem_grayscale.jpg", cv2.IMREAD_GRAYSCALE)  
  
hist_imagem(nova_imagem_cv2)
```



Podemos ver que essa imagem tem um fundo muito bem definido (a parede branca atrás de mim, no caso) e isso é percebido no histograma.

7

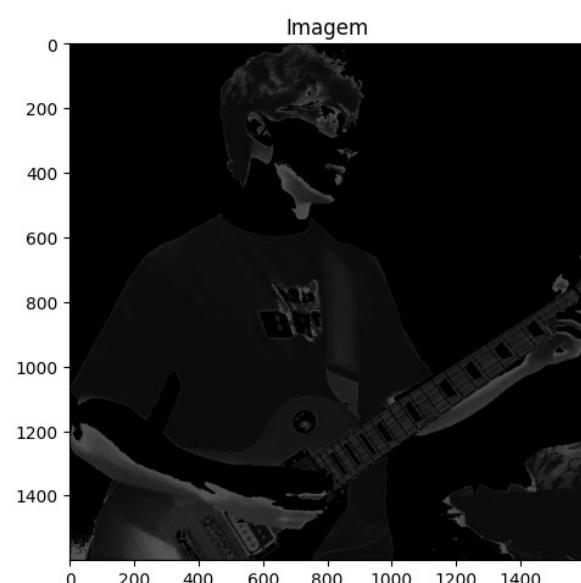
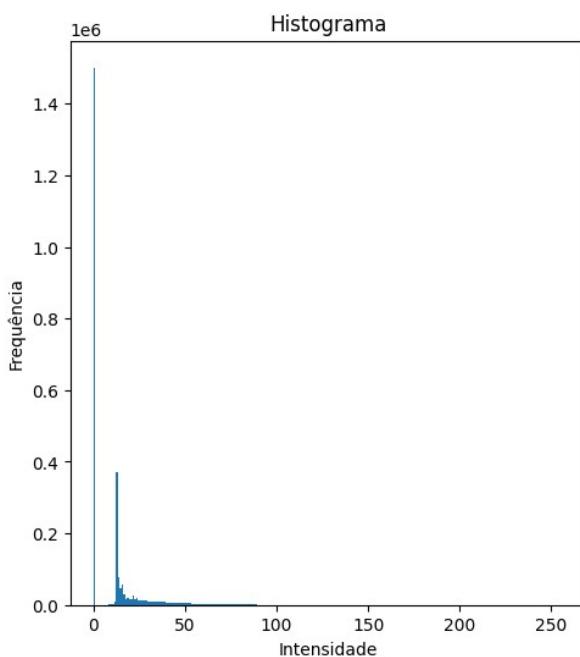
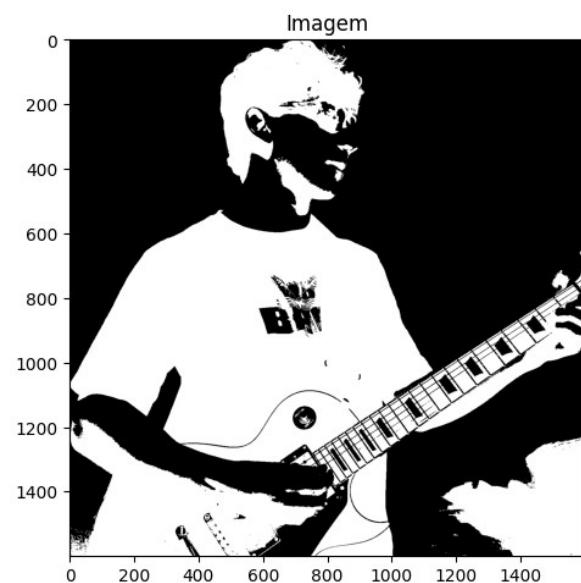
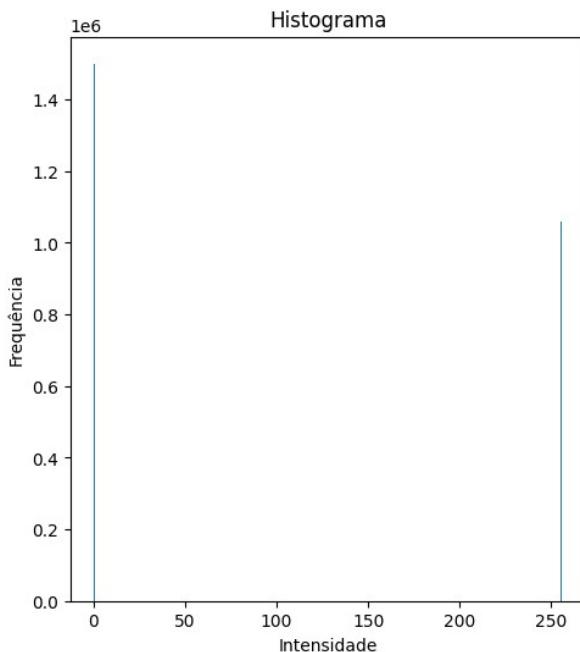
Para fazer a máscara vamos usar o método de Otsu, que foi passado na aula. Esse método já é incluso no OpenCV.

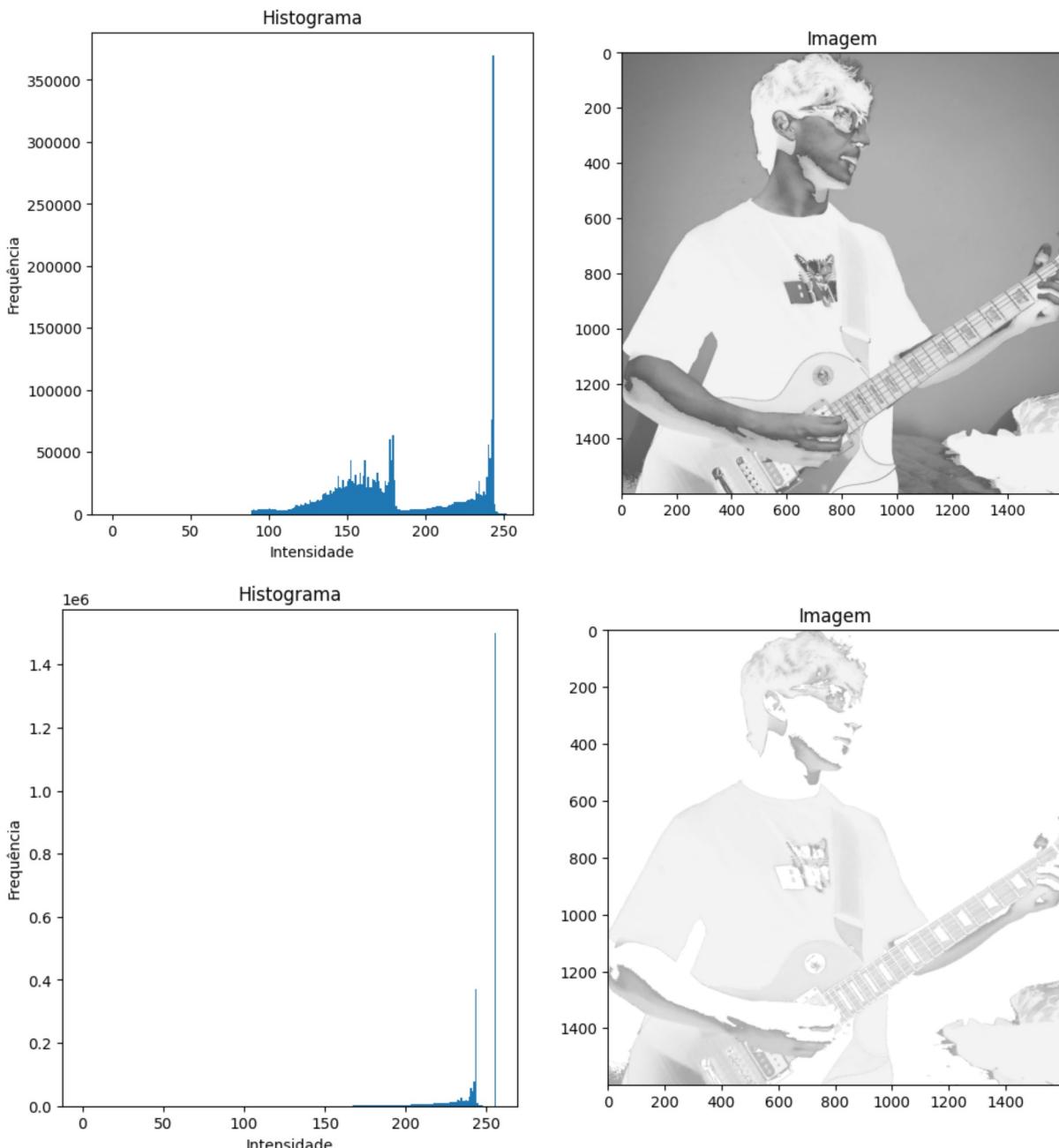
```
In [ ]: limiar, mascara = cv2.threshold(nova_imagem_cv2, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
print(limiar)
hist_imagem(mascara)

# agora aplicar a máscara na imagem
prim_plano = cv2.bitwise_and(nova_imagem_cv2, nova_imagem_cv2, mask=mascara)
seg_plano = cv2.bitwise_not(nova_imagem_cv2, nova_imagem_cv2, mask=mascara)
hist_imagem(prim_plano)
hist_imagem(seg_plano)

# vamos inverter a imagem pra ela ficar mais facil de visualizar
hist_imagem(cv2.bitwise_not(prim_plano))
```

88.0



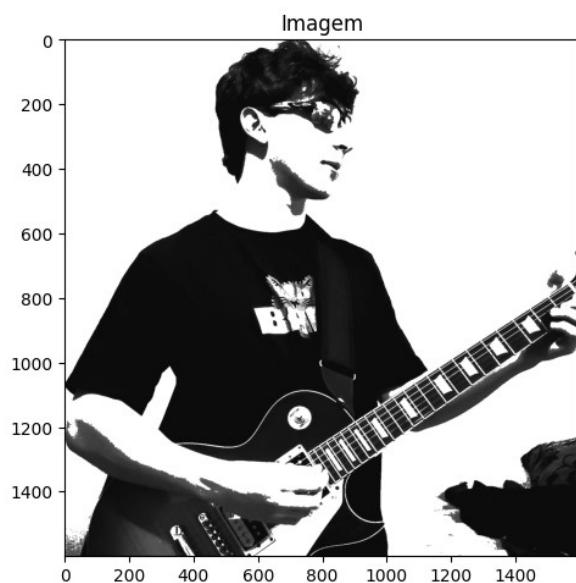
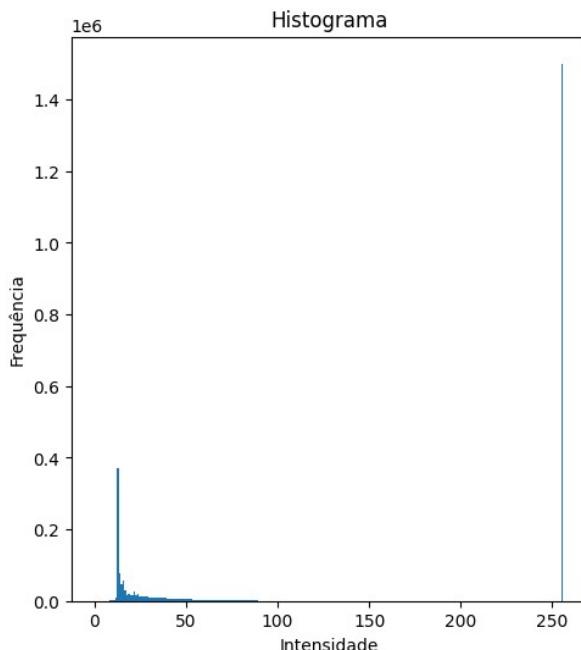


Vemos que a separação não ficou tão boa assim. Ela escolheu o limiar na intensidade 88, que talvez é um pouco baixa demais. Mas não podemos culpá-la, pois minha camiseta preta ocupa muitos pixels! Ela isolou bem esse "objeto" central (i.e. minha camiseta)

Quero tentar colocar a imagem com mascara, num fundo branco, pra ver se podemos representar melhor o que é primeiro plano e o que é segundo plano.

```
In [ ]: fundo_branco = np.ones_like(nova_imagem_cv2) * 255
```

```
# Onde a máscara é preta (0) utilizamos fundo_branco, senão, utilizamos prim_plano
final = np.where(mascara==0, fundo_branco, prim_plano)
hist_imagem(final)
```



Agora sim!!! 🎉