

A princípio, o dataset utilizado foi o **EuroSat Dataset**, podendo ser verificado no seguinte link: <https://www.kaggle.com/datasets/apollo2506/eurosat-dataset>

Foram selecionadas um total de 10 imagens **com baixo contraste** que necessitavam de um processamento digital para uma respectiva melhoria a nível visual.

```
In [2]: !pip install opencv-python
!python -m pip install -U scikit-image

Requirement already satisfied: opencv-python in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.21.2 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from opencv-python) (2.2.4)
[notice] A new release of pip is available: 24.3.1 -> 25.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Requirement already satisfied: scikit-image in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (0.25.2)
Requirement already satisfied: numpy>=1.24 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (2.2.4)
Requirement already satisfied: scipy>=1.11.4 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (1.15.2)
Requirement already satisfied: networkx>=3.0 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (3.4.2)
Requirement already satisfied: pillow>=10.1 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (11.1.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (2025.3.30)
Requirement already satisfied: packaging>=21 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in c:\users\caio\appdata\local\programs\python\python313\lib\site-packages (from scikit-image) (0.4)
[notice] A new release of pip is available: 24.3.1 -> 25.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
In [3]: import os
import cv2
from matplotlib import pyplot as plt
import numpy as np
from IPython.display import Image
from skimage import io
from skimage.exposure import match_histograms
```

```
In [4]: images = []
image_path = 'C:\\\\Users\\\\Caio\\\\AtividadePDI'

for i in range(1, 11):
    file_path = os.path.join(image_path, f"AnnualCrop_{i}.jpg")
    image = cv2.imread(file_path)
    if image is not None:
        images.append(image)
    else:
        print(f"Failed to load: AnnualCrop_{i}.jpg")
```

Com as imagens devidamente armazenadas, é necessário realizar o que se solicita na

atividade:

1. Dada uma imagem de satélite com baixa visibilidade, aplicar equalização de histograma para realce de contraste;
2. Aplicar especificação de histograma para adequar o estilo tonal de uma imagem a uma referência (por exemplo, uma imagem capturada em dia ensolarado);
3. Exibir e comparar os histogramas antes e depois do processamento;
4. Aplicar as técnicas apenas em determinadas regiões da imagem, como áreas agrícolas, usando uma máscara

Para satisfazer o que é proposto no item 1, serão utilizadas as imagens 1, 2 e 3.

Porém, note que as imagens são coloridas, ou seja, é necessário fazer os seguintes procedimentos: -Transformar as imagens para o espaço de cores HSV. -Realizar a equalização de histograma apenas no canal V (Valor). -Transformar as imagens de volta para o espaço de cores RGB.

```
In [5]: #Carregar as imagens
img1 = cv2.imread('C:\\Users\\Caio\\AtividadePDI\\AnnualCrop_1.jpg')
img2 = cv2.imread('C:\\Users\\Caio\\AtividadePDI\\AnnualCrop_2.jpg')
img3 = cv2.imread('C:\\Users\\Caio\\AtividadePDI\\AnnualCrop_3.jpg')
```

```
In [6]: # Converter para o espaço de cores HSV
hsv_img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
hsv_img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
hsv_img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2HSV)
```

```
In [7]: # Dividir os canais HSV
h1, s1, v1 = cv2.split(hsv_img1)
h2, s2, v2 = cv2.split(hsv_img2)
h3, s3, v3 = cv2.split(hsv_img3)
```

Agora, é necessário equalizar somente o canal **v**.

```
In [8]: # Equalizar o canal V (valor)
v1_equalized = cv2.equalizeHist(v1)
v2_equalized = cv2.equalizeHist(v2)
v3_equalized = cv2.equalizeHist(v3)
```

```
In [9]: # Mesclar os canais HSV de volta, com o canal V equalizado
hsv_img1 = cv2.merge([h1, s1, v1_equalized])
hsv_img2 = cv2.merge([h2, s2, v2_equalized])
hsv_img3 = cv2.merge([h3, s3, v3_equalized])
```

```
In [10]: # Converter de volta para o espaço de cores RGB
img1_equalized = cv2.cvtColor(hsv_img1, cv2.COLOR_HSV2RGB)
img2_equalized = cv2.cvtColor(hsv_img2, cv2.COLOR_HSV2RGB)
img3_equalized = cv2.cvtColor(hsv_img3, cv2.COLOR_HSV2RGB)
```

Essa parte do código exibe as imagens equalizadas

```
In [11]: # Supondo que você já tenha as imagens originais (em RGB) e equalizadas
```

```
# Cria uma figura com 3 linhas (para cada imagem) e 2 colunas (original vs equal
fig, axs = plt.subplots(nrows=3, ncols=2, figsize=(10, 15))

# Exibe a primeira imagem
axs[0, 0].imshow(img1) # Substitua 'img1_original' pela sua variável da imagem
axs[0, 0].set_title('Imagen 1')
axs[0, 0].axis('off')

axs[0, 1].imshow(img1_equalized)
axs[0, 1].set_title('Equalizada 1')
axs[0, 1].axis('off')

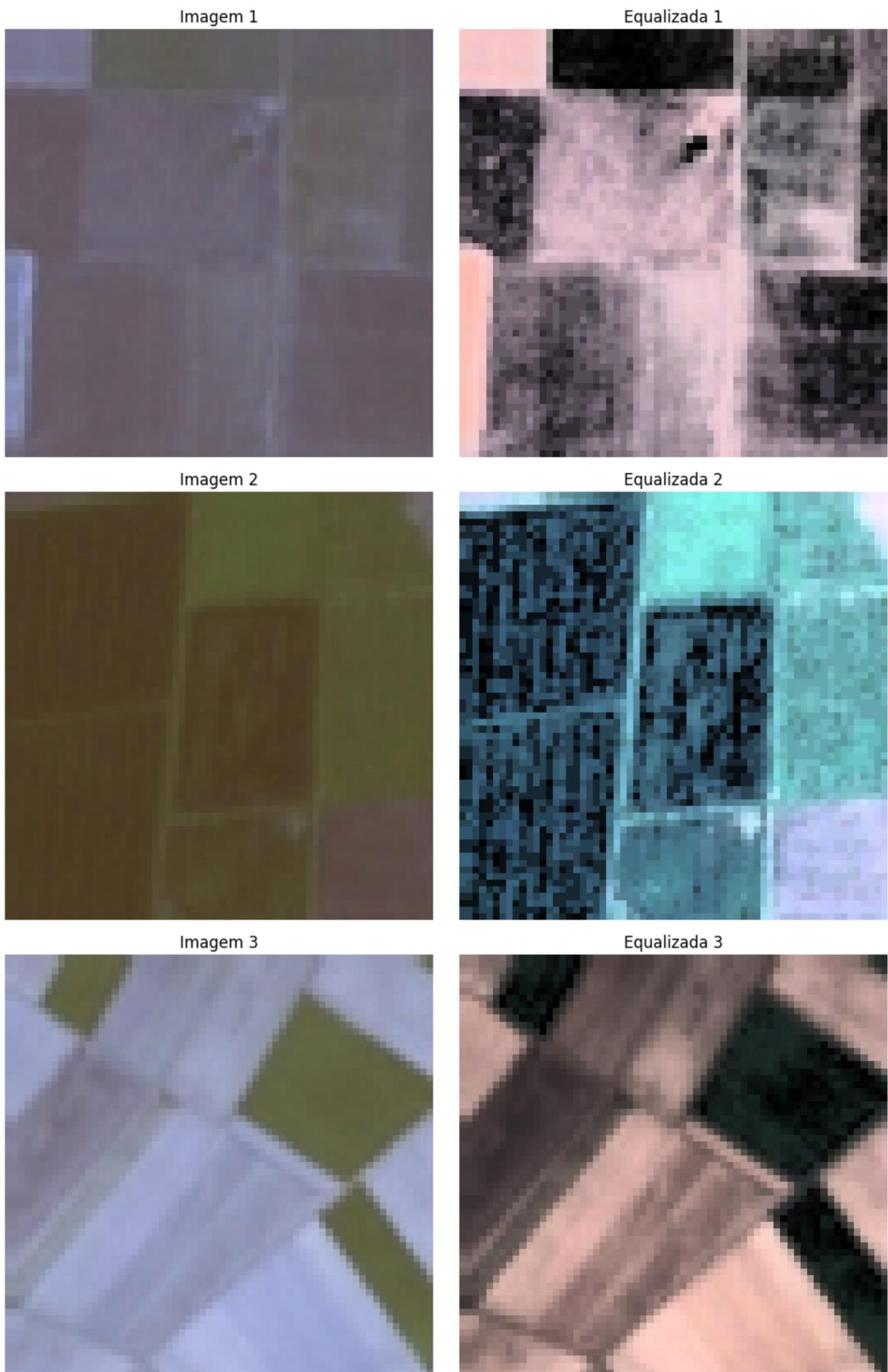
# Exibe a segunda imagem
axs[1, 0].imshow(img2) # Substitua 'img2_original' pela sua variável
axs[1, 0].set_title('Imagen 2')
axs[1, 0].axis('off')

axs[1, 1].imshow(img2_equalized)
axs[1, 1].set_title('Equalizada 2')
axs[1, 1].axis('off')

# Exibe a terceira imagem
axs[2, 0].imshow(img3) # Substitua 'img3_original' pela sua variável
axs[2, 0].set_title('Imagen 3')
axs[2, 0].axis('off')

axs[2, 1].imshow(img3_equalized)
axs[2, 1].set_title('Equalizada 3')
axs[2, 1].axis('off')

# Ajusta o Layout e mostra
plt.tight_layout()
plt.show()
```



Essa outra parte exibe os histogramas das imagens (original e equalizado)

```
In [12]: fig, axs = plt.subplots(nrows=3, ncols=4, figsize=(20, 12))
plt.subplots_adjust(wspace=0.3, hspace=0.3)
```

```

# Função para plotar histograma do canal V
def plot_histogram(v_channel, ax, title):
    ax.hist(v_channel.flatten(), bins=256, range=[0, 256], color='gray', alpha=0.5)
    ax.set_title(title)
    ax.set_xlim(0, 255)

# Processamento para cada imagem
for i in range(3):
    # --- Imagem Original ---
    # Converte BGR para RGB e HSV
    original_bgr = locals()[f'img{i+1}']
    original_rgb = cv2.cvtColor(original_bgr, cv2.COLOR_BGR2RGB)
    hsv_original = cv2.cvtColor(original_bgr, cv2.COLOR_BGR2HSV)
    v_original = hsv_original[:, :, 2]

    # Exibe imagem original
    axs[i, 0].imshow(original_rgb)
    axs[i, 0].set_title(f'Original {i+1}')
    axs[i, 0].axis('off')

    # Histograma original
    plot_histogram(v_original, axs[i, 1], f'Hist. Original {i+1}')

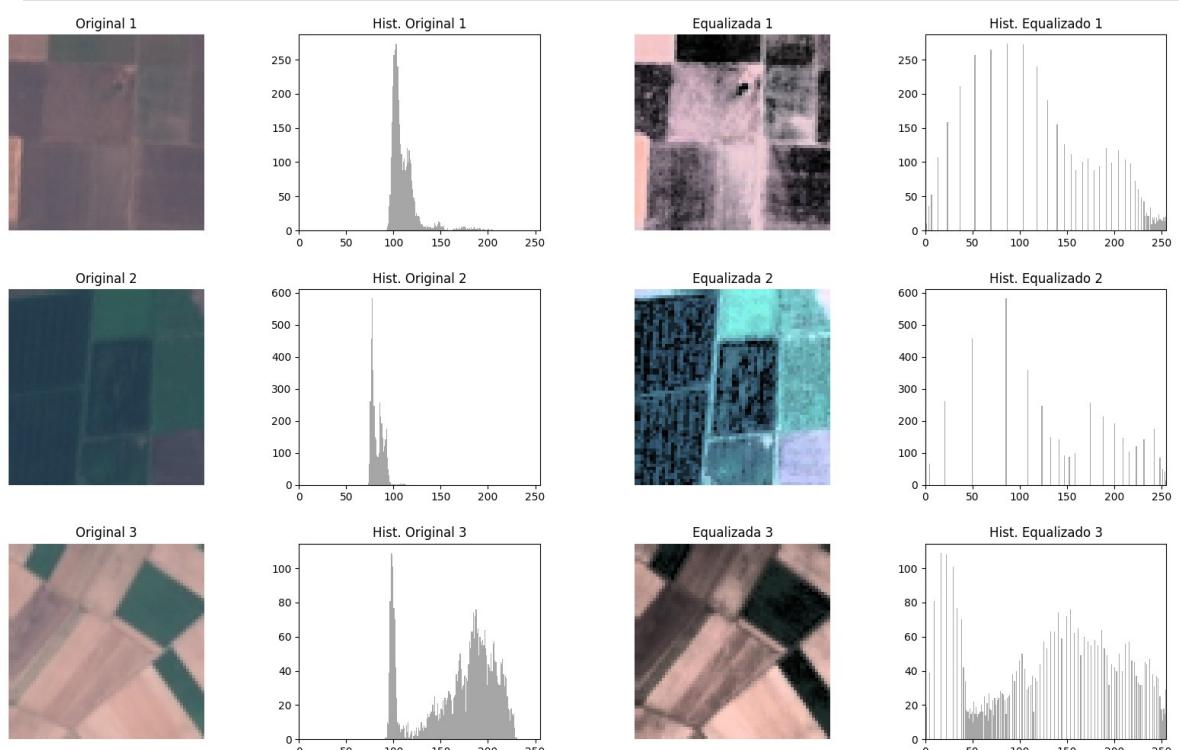
    # --- Imagem Equalizada ---
    # Canal V equalizado (já está em hsv_img1, hsv_img2, hsv_img3)
    v_equalized = locals()[f' hsv_img{i+1}'][:, :, 2]
    equalized_rgb = locals()[f'img{i+1}_equalized']

    # Exibe imagem equalizada
    axs[i, 2].imshow(equalized_rgb)
    axs[i, 2].set_title(f'Equalizada {i+1}')
    axs[i, 2].axis('off')

    # Histograma equalizado
    plot_histogram(v_equalized, axs[i, 3], f'Hist. Equalizado {i+1}')

plt.show()

```



Agora, é necessário realizar o seguinte procedimento: -Aplicar **especificação de histograma** para adequar o estilo tonal de uma imagem a uma referência (por exemplo, uma imagem capturada em dia ensolarado)

A imgem utilizada como referênci para o processo de especificação será:

```
In [21]: Image("maxresdefault.jpg", width=300)
```

Out[21]:



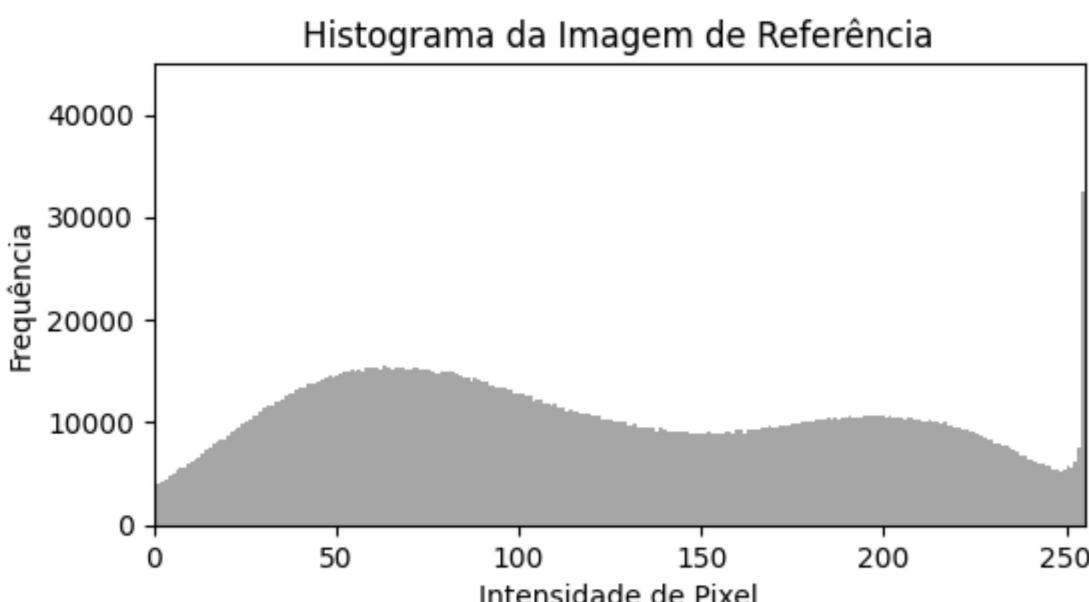
O respectivo histograma da imagem:

```
In [25]: # Leitura da imagem de referência
image = cv2.imread("maxresdefault.jpg")
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Converter de BGR para RGB

# Plotando o histograma tradicional
plt.figure(figsize=(6, 3))

# Histograma para todos os canais (transformando a imagem em uma única linha)
plt.hist(image_rgb.ravel(), bins=256, color='gray', alpha=0.7)
plt.title('Histograma da Imagem de Referência')
plt.xlim([0, 255]) # Intensidade de pixel
plt.xlabel('Intensidade de Pixel')
plt.ylabel('Frequência')

plt.show()
```



Agora, o processo de especificação de histograma será feito.

```
In [17]: # Leitura das imagens originais
image4 = io.imread("C:/Users/Caio/AtividadePDI/AnnualCrop_4.jpg")
image5 = io.imread("C:/Users/Caio/AtividadePDI/AnnualCrop_5.jpg")
image6 = io.imread("C:/Users/Caio/AtividadePDI/AnnualCrop_6.jpg")

# Leitura da imagem de referência
reference = io.imread("maxresdefault.jpg")

# Aplicação da correspondência de histograma
matched4 = match_histograms(image4, reference, channel_axis=-1)
matched5 = match_histograms(image5, reference, channel_axis=-1)
matched6 = match_histograms(image6, reference, channel_axis=-1)

# Listas para iteração
originals = [image4, image5, image6]
matched = [matched4, matched5, matched6]
titles = ['Imagen 4', 'Imagen 5', 'Imagen 6']

# Loop de visualização unificada
for i in range(3):
    fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10, 6))

    # Remove eixos
    for ax in axes.ravel():
        ax.set_axis_off()

    # Mostrando imagens
    axes[0, 0].imshow(originals[i])
    axes[0, 0].set_title('Original - ' + titles[i])

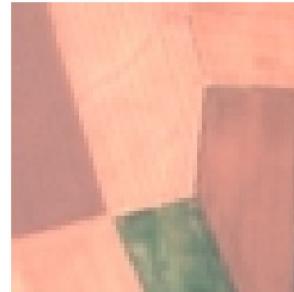
    axes[0, 1].imshow(matched[i])
    axes[0, 1].set_title('Correspondida')

    # Histogramas
    axes[1, 0].clear()
    axes[1, 0].hist(originals[i].ravel(), bins=256, color='gray', alpha=0.7)
    axes[1, 0].set_title('Histograma - ' + titles[i])
    axes[1, 0].set_xlim([0, 255])
    axes[1, 0].set_axis_on()

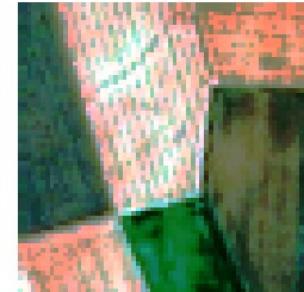
    axes[1, 1].clear()
    axes[1, 1].hist(matched[i].ravel(), bins=256, color='gray', alpha=0.7)
    axes[1, 1].set_title('Histograma Correspondido')
    axes[1, 1].set_xlim([0, 255])
    axes[1, 1].set_axis_on()

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

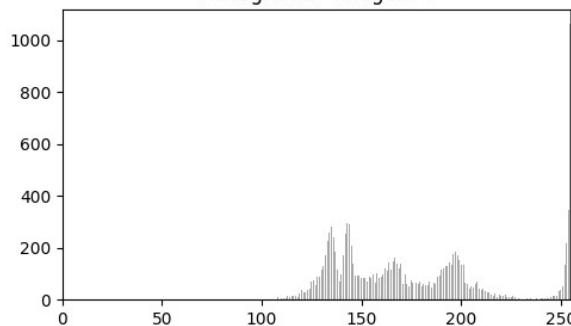
Original - Imagem 4



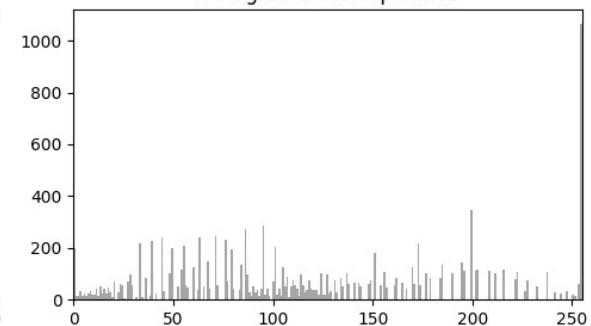
Correspondida



Histograma - Imagem 4



Histograma Correspondido



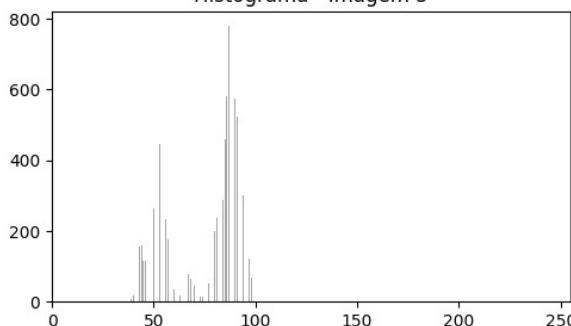
Original - Imagem 5



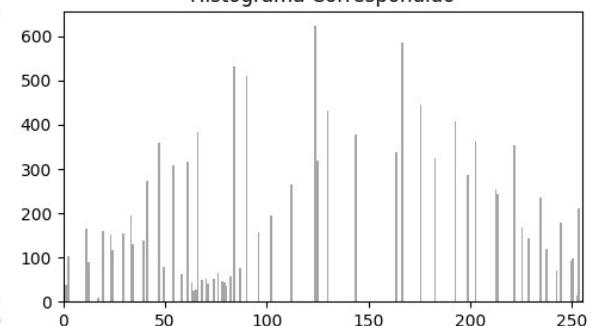
Correspondida

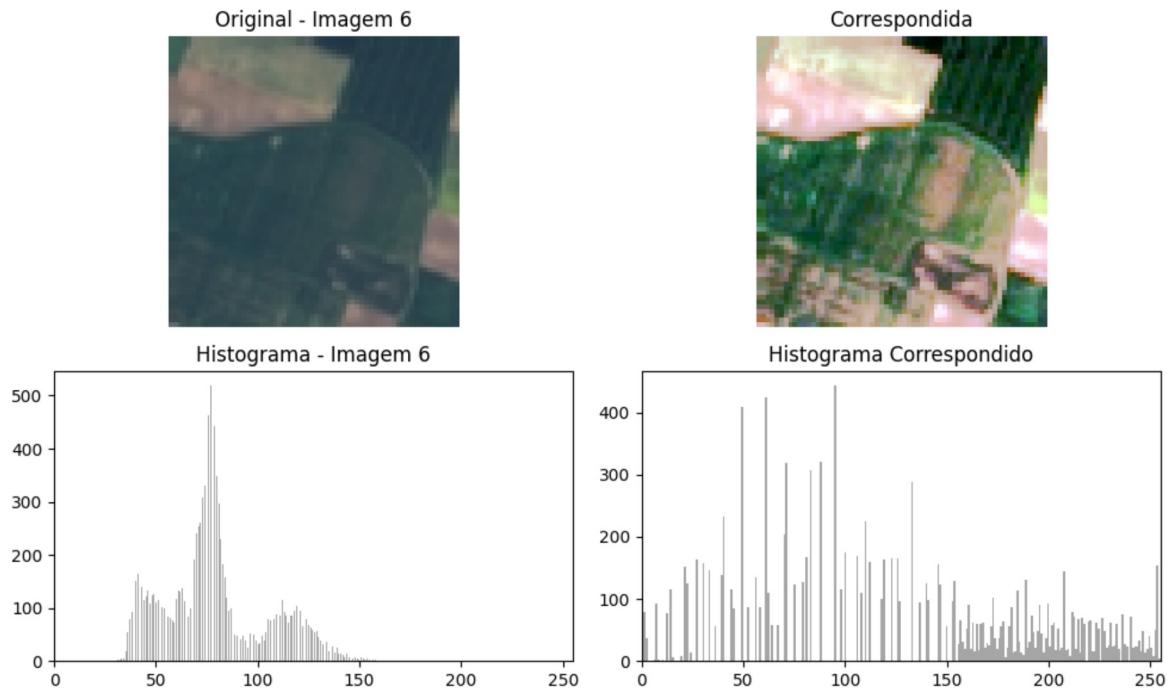


Histograma - Imagem 5



Histograma Correspondido





Agora, será feito o uso de uma **máscara** para melhorar o contraste de regiões específicas das imagens **7, 8, 9, 10**.

```
In [35]: import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, exposure
%matplotlib inline

# Configurações
image_files = ['AnnualCrop_7.jpg', 'AnnualCrop_8.jpg', 'AnnualCrop_9.jpg', 'AnnualCrop_10.jpg']
hsv_low = np.array([25, 40, 40])
hsv_high = np.array([100, 255, 255])

for file in image_files:
    # 1. Carregar e processar imagem
    img = cv2.imread(file)
    rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # 2. Criar máscara de vegetação
    mask = cv2.inRange(hsv, hsv_low, hsv_high)
    kernel = np.ones((7,7), np.uint8)
    mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel, iterations=2)
    mask = cv2.GaussianBlur(mask.astype(np.float32), (15,15), 0)

    # 3. Equalização CLAHE apenas na vegetação
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8,8))
    l_clahe = clahe.apply(l)
    lab_clahe = cv2.merge((l_clahe, a, b))
    enhanced = cv2.cvtColor(lab_clahe, cv2.COLOR_LAB2BGR)

    # 4. Aplicar com transição suave
    mask_normalized = mask[..., np.newaxis]/255.0
    result = (img * (1 - mask_normalized) + enhanced * mask_normalized).astype(np.uint8)
```

```
result_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)

# 5. Configurar plotagem no estilo desejado
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 8))

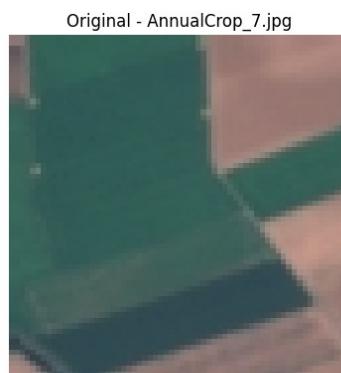
# Desligar eixos inicialmente
for ax in axes.ravel():
    ax.set_axis_off()

# Mostrar imagens
axes[0,0].imshow(rgb)
axes[0,0].set_title(f'Original - {file}')
axes[0,1].imshow(result_rgb)
axes[0,1].set_title('Processada')

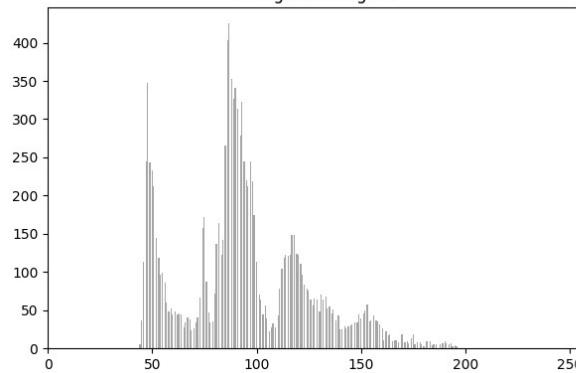
# Histogramas no estilo do seu exemplo
axes[1,0].clear()
axes[1,0].hist(rgb.ravel(), bins=256, color='gray', alpha=0.7)
axes[1,0].set_title('Histograma Original')
axes[1,0].set_xlim([0, 255])
axes[1,0].set_axis_on()

axes[1,1].clear()
axes[1,1].hist(result_rgb.ravel(), bins=256, color='gray', alpha=0.7)
axes[1,1].set_title('Histograma Processado')
axes[1,1].set_xlim([0, 255])
axes[1,1].set_axis_on()

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



Histograma Original



Histograma Processado

