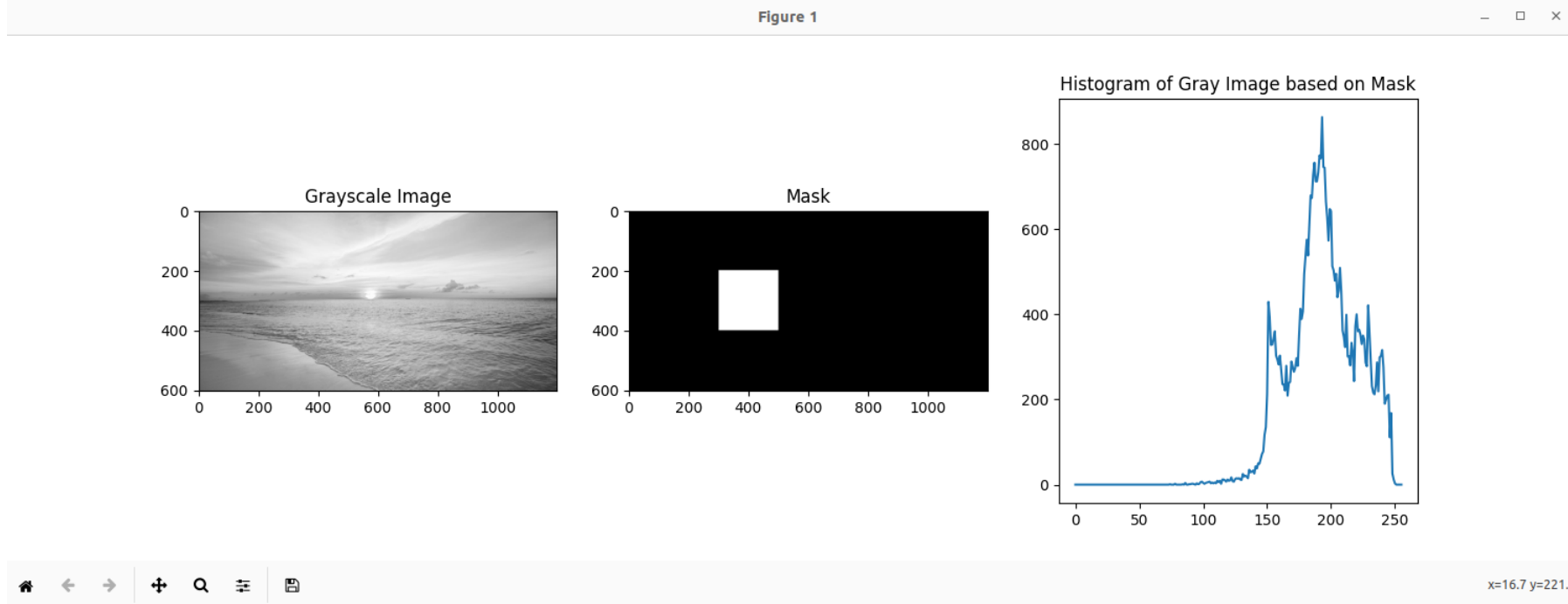
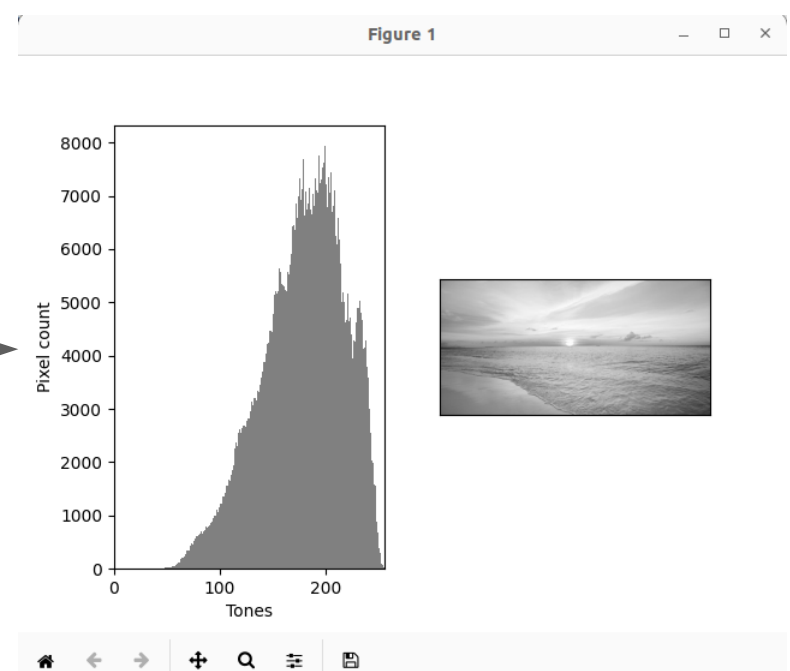


O histograma de uma máscara particular ou região de interesse (ROI) pode ser útil em alguns casos para avaliar o histograma local em uma área específica.

```
code # mask_hist.py ~
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 img = cv2.imread('images/tennis.jpg', 0)
6
7 # mask = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
8 mask = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
9 mask[200:400, 200:300] = 255
10
11 # Histogram based on mask
12 hist = cv2.calcHist([img], [0], mask, [256], [0, 256])
13
14 # plot image, mask, histogram
15 fig, ax = plt.subplots(1, 3, figsize=(15, 5))
16
17 ax[0].imshow(img, cmap=gray)
18 ax[0].set_title('Grayscale Image')
19
20 ax[1].imshow(mask, cmap=gray)
21 ax[1].set_title('Mask')
22
23 ax[2].plot(hist, color='r')
24 ax[2].set_title('Histogram of Gray Image based on Mask')
25
26 plt.show()
```



```
code # hist.py ~
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Load the image in grayscale
6 img = cv2.imread('images/tennis.jpg', cv2.IMREAD_GRAYSCALE)
7
8 # Calculate histogram
9 hist, bins = cv2.histogram(img, flatten(), 256, [0, 256])
10
11 # Plot the histogram
12 plt.subplot(121)
13 plt.plot(hist, flatten(), [0, 256], color='gray')
14 plt.xlabel('bins')
15 plt.ylabel('Pixel count')
16
17 # Display the image
18 plt.subplot(122)
19 plt.imshow(img, cmap=gray)
20 plt.title('Histogram of Gray Image')
21
22 # Show the plot
23 plt.show()
```



Conceito

Os histogramas são utilizados para apresentar as informações estatísticas de uma imagem em um formato visualmente interpretável. Com o auxílio de um histograma, é possível identificar tipos específicos de problemas em uma imagem.

O histograma mostra como as tonalidades de cinza são distribuídas na imagem e pode ajudar a identificar se a faixa radiométrica da imagem está sendo usada da melhor maneira possível. É como uma imagem da distribuição das tonalidades de cinza, o que ajuda a identificar onde estão os pontos mais claros e mais escuros da imagem.

Histograma de imagem é uma representação gráfica da distribuição dos valores de intensidade de pixels em uma imagem. Ele exibe o número de pixels em cada nível de intensidade, ajudando a visualizar o contraste, brilho e alcance tonal geral de uma imagem.

Ao observar o histograma e analisar a distribuição espacial dos valores, é possível ter uma ideia geral do brilho de uma imagem. Se os valores do histograma estiverem concentrados à esquerda, a imagem será mais escura. Se eles estiverem concentrados à direita, a imagem será mais clara.

É importante observar que um histograma contém apenas informações estatísticas sobre a distribuição de pixels e não fornece informações espaciais sobre onde esses pixels estão localizados na imagem. Portanto, não é possível recriar uma imagem apenas a partir de seu histograma.

Histogramas de Imagem

Histogramas com mais de um byte por pixel

Embora alguns scanners modernos possam usar 12 ou 16 bits para capturar valores de intensidade mais precisos, os benefícios de profundidades de bits mais altas são limitados e podem causar problemas de compatibilidade com alguns softwares e formatos de arquivo. Além disso, os olhos humanos não conseguem discernir mais do que 256 níveis de cinza em uma tela de computador, portanto, o uso de profundidades de bits mais altas não necessariamente resulta em uma melhor qualidade de imagem.

Embora o aumento de profundidade possa não ser necessário para aplicações cotidianas, há certas áreas, como medicina e astronomia, onde é crucial. Nessas áreas especializadas, uma maior profundidade permite medições e análises mais precisas, levando a resultados mais precisos e descobertas potencialmente salvadoras de vidas.

Histogramas de Imagens Coloridas

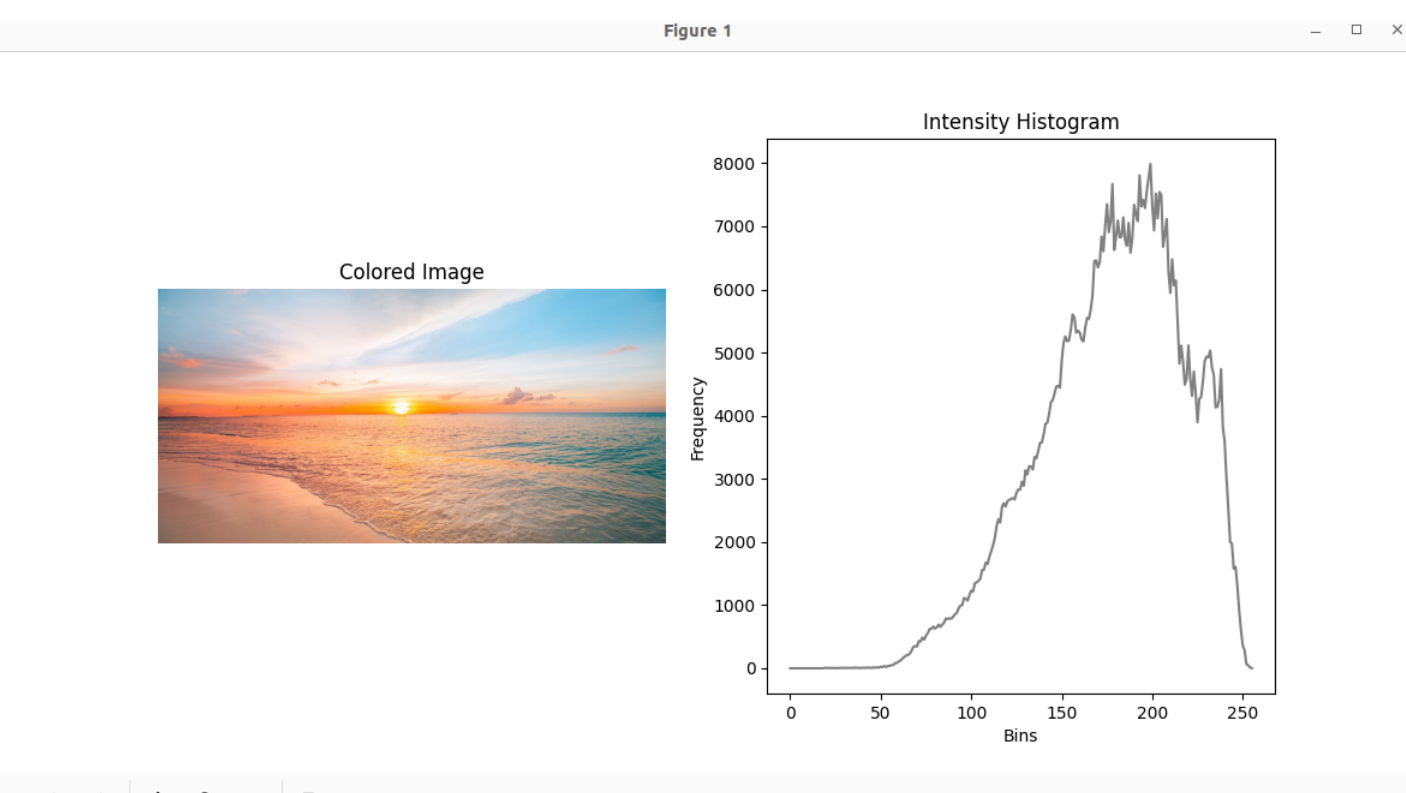
Histogramas de cada canal de cor

Mesmo que o histograma de luminância leve em conta todos os canais de cor, os erros de imagem que aparecem em canais individuais podem passar despercebidos.

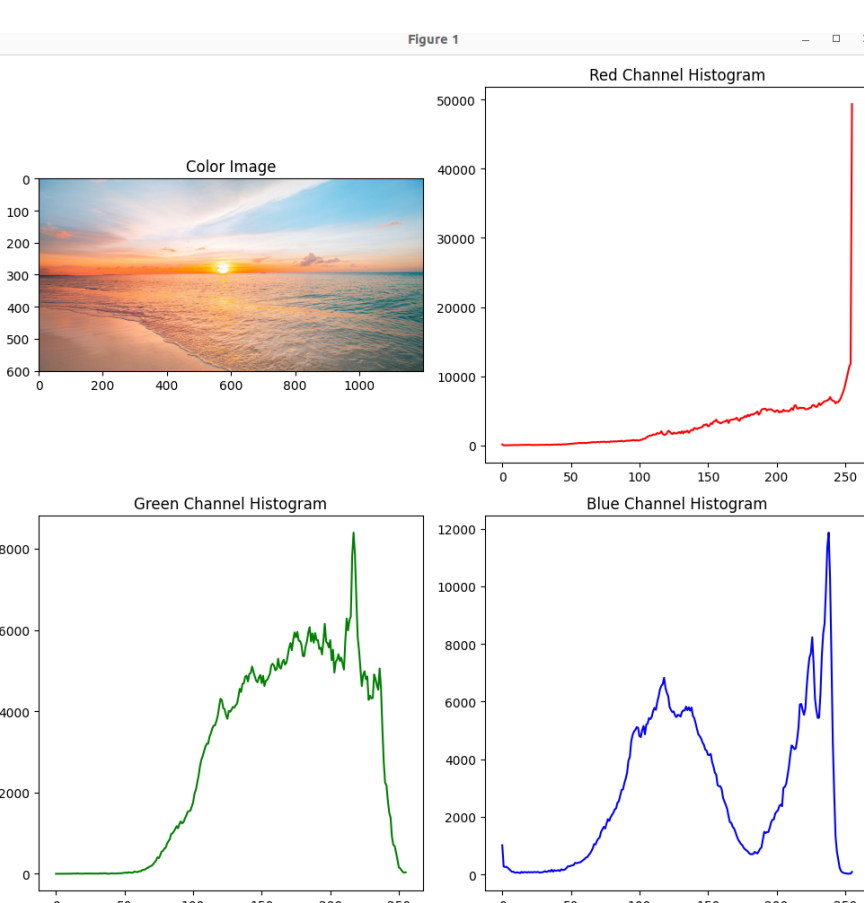
Os histogramas de luminância em cada componente fornecem informações úteis sobre a iluminação, contraste, faixa dinâmica e efeitos de saturação relacionados aos componentes de cor individuais. No entanto, é importante lembrar que eles não fornecem informações sobre a distribuição de cor real na imagem, pois são baseados em canais de cor individuais e não na combinação de canais individuais que formam a cor de um pixel individual.

```
code # intensity_hist.py ~
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Load colored image
6 img = cv2.imread('images/tennis.jpg')
7
8 # Get the gray image based on the color one
9 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
10
11 # Calculate histogram
12 hist = cv2.calcHist([img], [0, 1, 2], None, [256], [0, 256])
13
14 # Plot
15 fig, ax = plt.subplots(1, 3, figsize=(15, 5))
16
17 ax[0].imshow(img, cmap=gray)
18 ax[0].set_title('Color Image')
19
20 ax[1].imshow(gray, cmap=gray)
21 ax[1].set_title('Grayscale Image')
22
23 ax[2].plot(hist, color='r')
24 ax[2].set_title('Intensity Histogram')
25
26 plt.show()
```

Em resumo, o histograma de intensidade ou luminância de uma imagem colorida é o mesmo que o histograma da imagem em escala de cinza correspondente. A imagem em escala de cinza é obtida calculando a luminância dos canais individuais da imagem colorida usando uma soma ponderada que leva em consideração a teoria da percepção de cores.



```
code # color_hist.py ~
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Load color image
6 img = cv2.imread('images/tennis.jpg')
7
8 # Split image into three color channels
9 b, g, r = cv2.split(img)
10
11 # Calculate histograms for each color channel
12 hist_b = cv2.calcHist([img], [0], None, [256], [0, 256])
13 hist_g = cv2.calcHist([img], [1], None, [256], [0, 256])
14 hist_r = cv2.calcHist([img], [2], None, [256], [0, 256])
15
16 # Plot color image and histograms in one figure
17 fig, axes = plt.subplots(1, 4, figsize=(15, 10))
18 plt.suptitle('Color Image and Histograms')
19
20 axes[0].imshow(img, cmap=gray)
21 axes[0].set_title('Color Image')
22
23 axes[1].plot(hist_b, color='b')
24 axes[1].set_title('Blue Channel Histogram')
25
26 axes[2].plot(hist_g, color='g')
27 axes[2].set_title('Green Channel Histogram')
28
29 axes[3].plot(hist_r, color='r')
30 axes[3].set_title('Red Channel Histogram')
31
32 # Tight layout
33 plt.tight_layout()
34
35 plt.show()
```

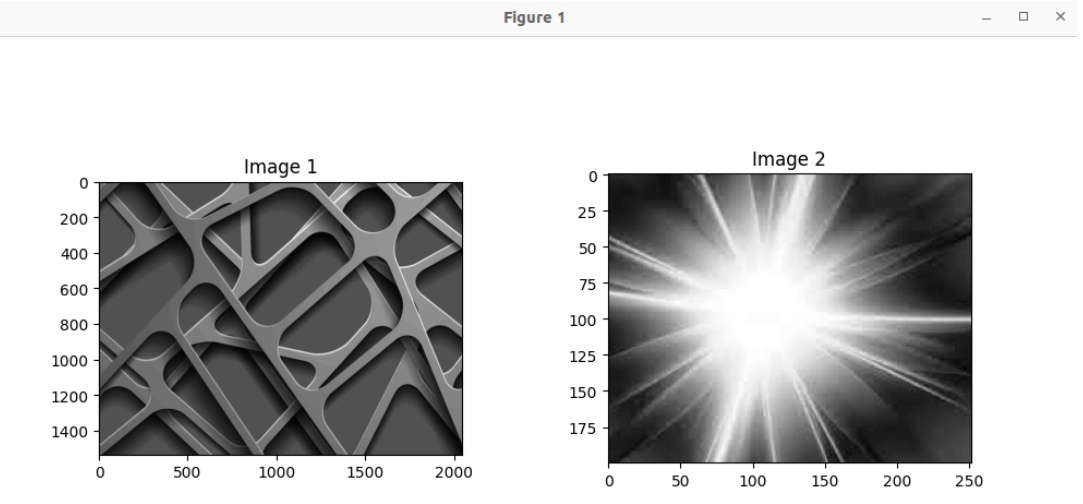


Fotografia



Fotógrafos profissionais que usam câmeras de alta qualidade sempre têm acesso a um histograma ao vivo da imagem sendo capturada. Isso os ajuda a ajustar as configurações da câmera e utilizar o canal de cor ou escala de cinza da melhor maneira possível para melhorar a exibição da imagem. O histograma fornece aos técnicos de aquisição de imagens uma visão de como a luz que entra na câmera foi capturada pelo sensor da câmera. Por exemplo, a imagem está muito clara ou escura?

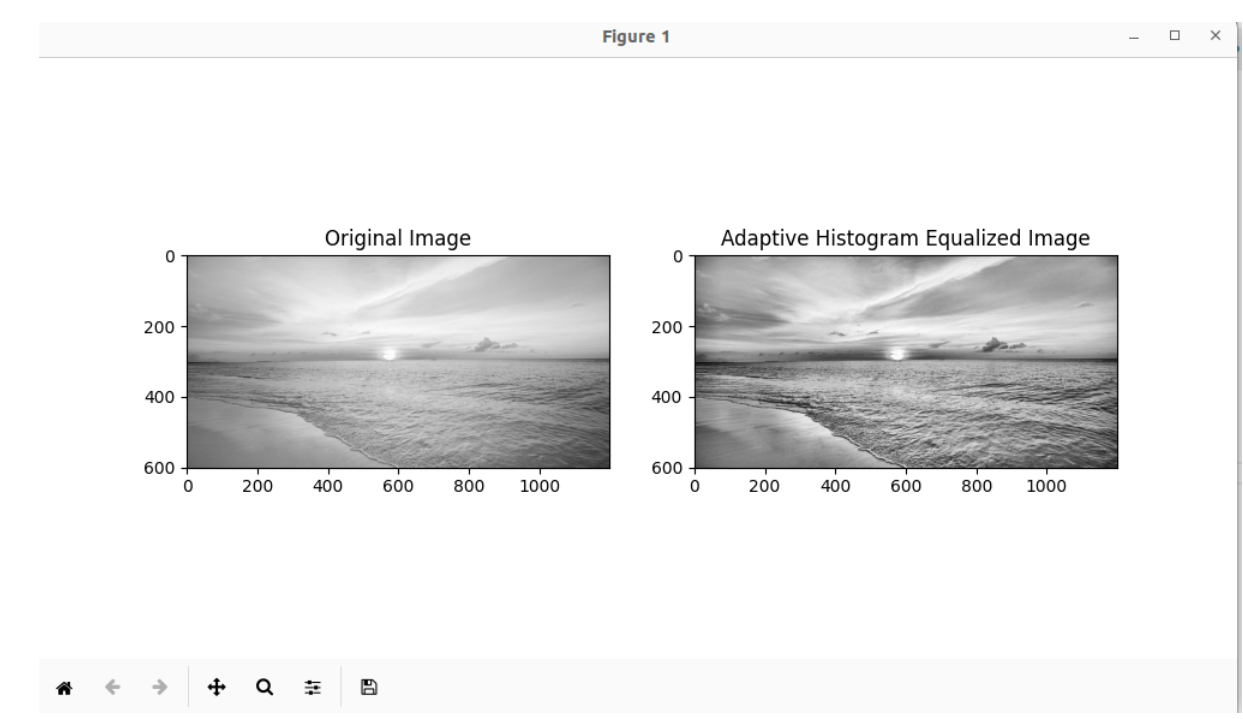
```
code # hist.py ~
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Load image in grayscale mode
6 img = cv2.imread('images/tennis.jpg', cv2.IMREAD_GRAYSCALE)
7
8 # Calculate histogram
9 hist, bins = cv2.histogram(img, flatten(), 256, [0, 256])
10
11 # Plot
12 fig, ax = plt.subplots(1, 2, figsize=(15, 5))
13
14 ax[0].imshow(img, cmap=gray)
15 ax[0].set_title('Original Image')
16
17 ax[1].plot(hist, color='r')
18 ax[1].set_title('Histogram of Image 1')
19
20 plt.show()
```



Histograma Adaptativo

O histograma adaptativo é uma técnica de processamento de imagem que tem como objetivo melhorar o contraste e o brilho de uma imagem ajustando o histograma de forma local. Ao contrário da equalização de histograma global, a equalização de histograma adaptativo divide a imagem em regiões menores e aplica a equalização de histograma em cada região de forma independente, permitindo uma melhor preservação do contraste local e evitando a sobre-amplificação de ruído que pode ocorrer no método global. Essa técnica é particularmente útil para imagens com iluminação não uniforme ou com regiões de iluminação significativamente diferentes.

```
code # adaptive_hist.py ~
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Load image in grayscale mode
6 img = cv2.imread('images/tennis.jpg', cv2.IMREAD_GRAYSCALE)
7
8 # Adaptive histogram
9 clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
10 climg = clahe.apply(img)
11
12 # Plot
13 fig, ax = plt.subplots(1, 2, figsize=(15, 5))
14
15 ax[0].imshow(img, cmap=gray)
16 ax[0].set_title('Original Image')
17
18 ax[1].imshow(climg, cmap=gray)
19 ax[1].set_title('Adaptive Histogram Equalized Image')
20
21 plt.show()
```



Histograma Acumulativo

A histograma cumulativo, que é derivado do histograma regular, é útil ao realizar determinadas operações de imagem envolvendo histogramas, como a equalização de histograma. O histograma cumulativo é definido como a soma de todos os valores do histograma até um determinado nível de intensidade. Ele é denotado por $H(i)$, onde i é o nível de intensidade e $H(i)$ é o valor do histograma para o nível de intensidade i . O histograma cumulativo é uma ferramenta útil em processamento de imagem para melhorar o contraste e ajustar o brilho.

Equalização de Histograma

Equalização de histograma é uma técnica utilizada no processamento de imagens para melhorar o contraste de uma imagem através do ajuste das intensidades de seus pixels. O principal objetivo da equalização de histograma é obter uma distribuição mais uniforme das intensidades de pixel na imagem, o que pode resultar em uma melhoria na qualidade geral da imagem e tornar detalhes que eram anteriormente difíceis de ver mais visíveis.

```
code # equalize_hist.py ~
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 # Load image in grayscale mode
6 img = cv2.imread('images/tennis.jpg', cv2.IMREAD_GRAYSCALE)
7
8 # Apply histogram equalization
9 eq_img = cv2.equalizeHist(img)
10
11 # Plot original and equalized image side by side
12 fig, axes = plt.subplots(1, 2, figsize=(15, 5))
13
14 axes[0].imshow(img, cmap=gray)
15 axes[0].set_title('Original')
16
17 axes[1].imshow(eq_img, cmap=gray)
18 axes[1].set_title('Equalized')
19
20 plt.show()
```

