COM 3105

Processamento Digital de Imagens

LN05 - 2023

Processamento de Histogramas | Parte II

Prof. Eng. Marcos Cordeiro d'Ornellas, Ph.D.

Revisado em: 26.03.2023.

1 Histogramas de imagens com mais de 8 bits

Imagens em tons de cinza têm pixels que variam do preto ao branco, passando por uma série de etapas de intensidade intermediárias. **A maioria das imagens permite 256 níveis de intensidade diferentes, um número que surge dos 256 valores diferentes que um byte pode assumir. Como um byte é composto de 8 bits, essas imagens são conhecidas como imagens em tons de cinza de 8 bits.** Alguns scanners de última geração podem digitalizar com uma escala de intensidade mais precisa. Utilizam 12 ou 16 bits para representar os valores de intensidade da imagem, possibilitando o registro de 4.096 ou 65.536 diferentes níveis de cinza.

O número de bits usados para valores de intensidade é conhecido como profundidade de bits. Embora uma profundidade de bits alta pareça uma opção preferida, na prática, **os benefícios são limitados**. Aqui estão alguns pontos a serem considerados sobre imagens de 12 ou 16 bits:

- Eles ocuparão mais espaço.
- Eles podem ser salvos em formatos de arquivo limitados.
- Nem todos os softwares de imagem podem funcionar com imagens de 12 ou 16 bits. Mesmo se puderem, sua funcionalidade pode ser reduzida.
- **Nem todos os softwares podem funcionar com imagens de 12 ou 16 bits. Você pode ter que reduzir a imagem para 8 bits antes de digitalizar.**
- **Você não verá nenhuma diferença na tela do computador**, porque os monitores de computador convencionais são configurados para mostrar apenas até 256 tons de cinza. Mesmo se o monitor pudesse mostrar mais (isso é possível usando alguns truques de programação sofisticados), o olho humano não consegue discernir tantos.
- Você não verá melhor as áreas escuras da imagem. **O principal problema na varredura das imagens médicas, por exemplo, é discernir detalhes nas áreas escuras (por exemplo, sob o queixo, ao redor do osso hióide e pescoço). Este problema não será resolvido aumentando a profundidade de bits porque a limitação é baseada na densidade óptica do scanner e dos componentes eletrônicos.**

O exemplo a seguir utiliza o processo de binning em uma imagem de 12-bits extraída de um exame e convertida a partir do formato Dicom.

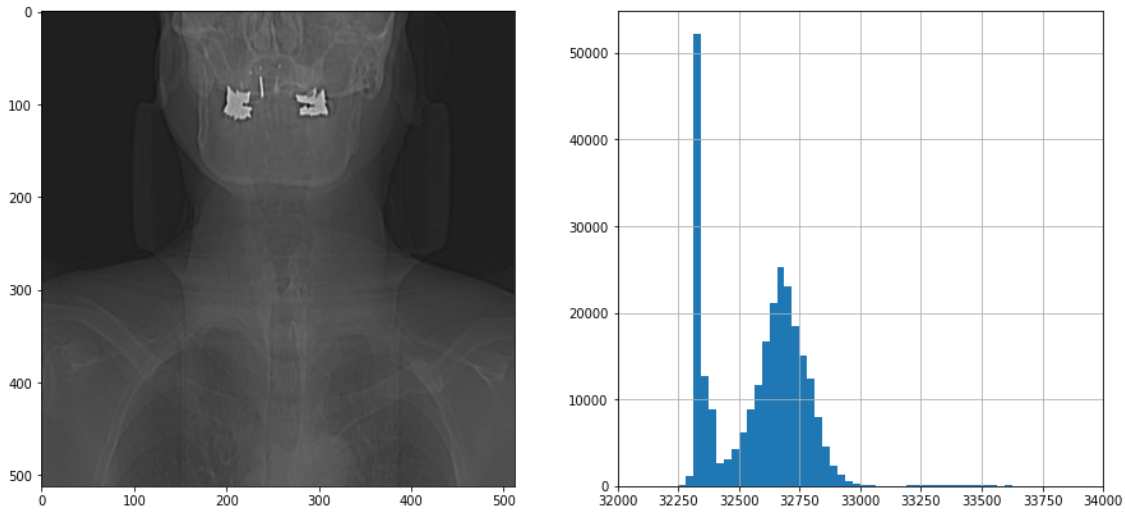
In [1]:

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

plt.figure(figsize=(15,15))

img = cv2.imread('book_folder/ct.dcm512.tif', cv2.IMREAD_ANYDEPTH)

plt.subplot(2,2,1)
plt.imshow(img,cmap='gray')

plt.subplot(2,2,2)
plt.grid(True)

plt.hist(img.ravel(),64,[32000,34000])
plt.xlim([32000,34000])
```

executed in 1.30s, finished 19:11:22 2023-03-26

Out[1]:

```
(32000.0, 34000.0)
```



2 Histogramas de imagens coloridas

2.1 Histogramas de intensidade

O histograma de intensidade ou luminância h_{Lum} de uma imagem colorida nada mais é do que o histograma da imagem em tons de cinza correspondente, portanto, naturalmente, todos os aspectos da discussão anterior também se aplicam a esse tipo de histograma. **A imagem em tons de cinza é obtida calculando a luminância dos canais individuais da imagem colorida. Ao calcular a luminância, não é suficiente simplesmente calcular a média dos valores de cada canal de cor; em vez disso, uma soma ponderada que leva em consideração a teoria da percepção de cores deve ser calculada.**

2.2 Histogramas de canal de cor individual

Mesmo que o histograma de luminância leve em consideração todos os canais de cores, erros de imagem que aparecem em canais únicos podem permanecer desconhecidos. Por exemplo, o histograma de luminância pode parecer limpo mesmo quando um dos canais de cor está supersaturado. **Em**

imagens RGB, o canal azul contribui apenas com uma pequena quantidade para o brilho total e, portanto, é especialmente sensível a esse problema.

Imagem no formato RGB **Histograma da imagem**

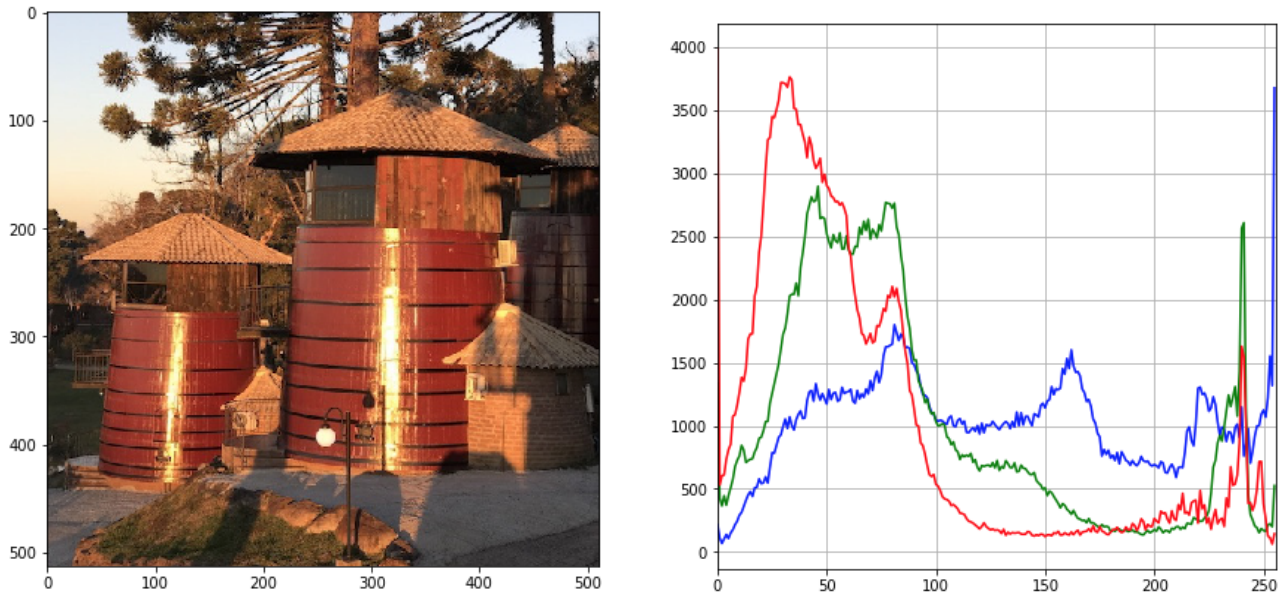


Figura 1: Histograma da imagem colorida.

2.3 Histogramas de cores combinadas

Os histogramas de luminância e os histogramas de componentes fornecem informações úteis sobre a iluminação, contraste, faixa dinâmica e efeitos de saturação relativos aos componentes de cor individuais. É importante lembrar que eles não fornecem informações sobre a distribuição das cores reais na imagem, pois são baseados nos canais de cores individuais e não na combinação dos canais individuais que formam a cor de um pixel individual. Considere, por exemplo, quando h_R, o histograma do componente para o canal vermelho, contém a entrada $h_R(200) = 24$.

Então, sabemos apenas que a imagem tem 24 pixels com um valor de intensidade de vermelho de 200. A entrada não nos diz nada sobre os valores de verde e azul desses pixels, que poderiam ser qualquer valor válido ($*$); isso é,

$$(r, g, b) = (200, *, *).$$

Suponha ainda que os três histogramas de componentes incluam as seguintes entradas:

$$h_R(50) = 100,\ h_G(50) = 100,\ h_B(50) = 100.$$

Podemos concluir que a imagem contém 100 pixels com a combinação de cores

$$(r, g, b) = (50, 50, 50)$$

ou que essa cor ocorre em tudo? Em geral, não, porque não há como verificar a partir desses dados se existe um pixel na imagem em que todos os três componentes têm o valor 50. A única coisa que realmente podemos dizer é que o valor da cor (50, 50, 50) pode ocorrer no máximo 100 vezes nesta imagem.

Os histogramas de componentes fornecem informações adicionais sobre a distribuição de intensidade nos canais de cores individuais. Ao calcular histogramas de componentes, cada canal de cor é considerado uma imagem de intensidade separada e cada histograma é calculado

independentemente dos outros canais. O resultado do código a seguir mostra o histograma de três componentes h_R, h_G e h_B de uma imagem colorida RGB típica.

> Embora os histogramas convencionais (intensidade ou componente) de imagens de cores representem propriedades importantes, eles não fornecem nenhuma informação útil sobre a composição das cores reais em uma imagem.

Na verdade, **uma coleção de imagens coloridas pode ter histogramas de componentes muito semelhantes e ainda conter cores totalmente diferentes.** Isso leva ao tópico interessante do histograma combinado, que usa informações estatísticas sobre os componentes de cor combinados na tentativa de determinar se duas imagens são aproximadamente semelhantes em sua composição de cores. Os recursos calculados a partir desse tipo de histograma costumam formar a base dos métodos

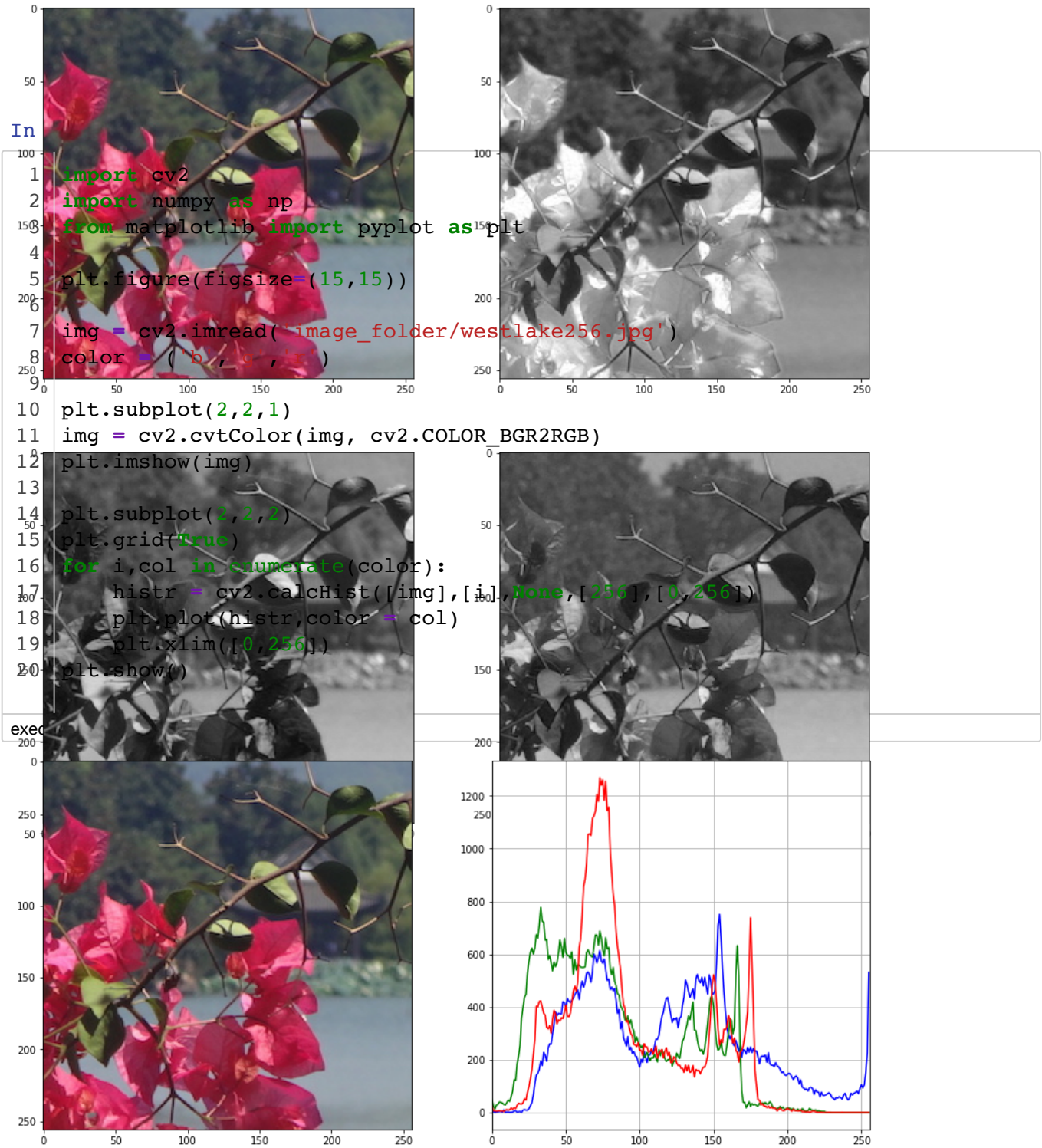
In [2]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

plt.figure(figsize=(15,15))

bgr_img = cv2.imread('image_folder/westlake256.jpg')
b,g,r = cv2.split(bgr_img)       # get b,g,r
#r,g,b = cv2.split(bgr_img)
rgb_img = cv2.merge([r,g,b])     # switch it to rgb

plt.subplot(2,2,1)
#img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(rgb_img)

plt.subplot(2,2,2)
plt.imshow(r,cmap='gray')

plt.subplot(2,2,3)
plt.imshow(g,cmap='gray')

plt.subplot(2,2,4)
plt.imshow(b,cmap='gray')

```

executed in 726ms, finished 19:11:23 2023-03-26

Out[2]:

```
<matplotlib.image.AxesImage at 0x7f93f812d2e0>
```



3 Exemplos de aplicações

3.1 O histograma cumulativo

O histograma cumulativo, que é derivado do histograma comum, é útil ao realizar certas operações de imagem envolvendo histogramas; por exemplo, equalização do histograma O histograma cumulativo H é definido como

$$H(i) = \sum_{j=0}^{i} h(j) \tag{1}$$

for $0 \leq i < K$.

Um valor particular $H(i)$ é, portanto, a soma de todos os valores $h(j)$, com $j \leq i$, no histograma original. Alternativamente, podemos definir H recursivamente

$$H(i) = \begin{cases} h(0) & \text{for } i = 0 \\ H(i-1) + h(i) & \text{for } 0 < i < k \end{cases} \tag{2}$$

O histograma cumulativo $H(i)$ é uma função monotonicamente crescente com um valor máximo

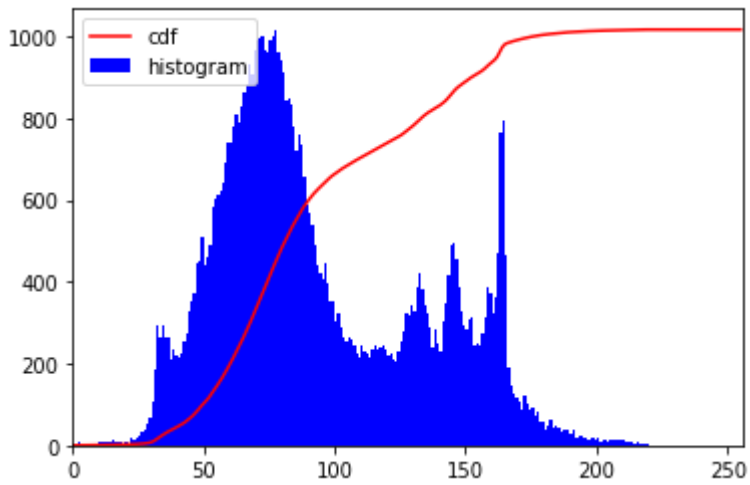
$$H(K-1) = \sum_{j=0}^{K-1} h(j) = M \cdot N \tag{3}$$

isto é, o número total de pixels em uma imagem de largura M e altura N. O próximo código mostra um exemplo concreto de um histograma cumulativo. O histograma cumulativo é útil não principalmente para visualização, mas como uma ferramenta simples e poderosa para capturar informações estatísticas de uma imagem.

In [4]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('image_folder/westlake256.jpg',0)

hist,bins = np.histogram(img,256,[0,256])

cdf = hist.cumsum()

cdf_normalized = cdf *hist.max()/ cdf.max() # this line not necessary.

plt.plot(cdf_normalized, color = 'r')
plt.hist(img.flatten(),256,[0,256], color = 'b')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```

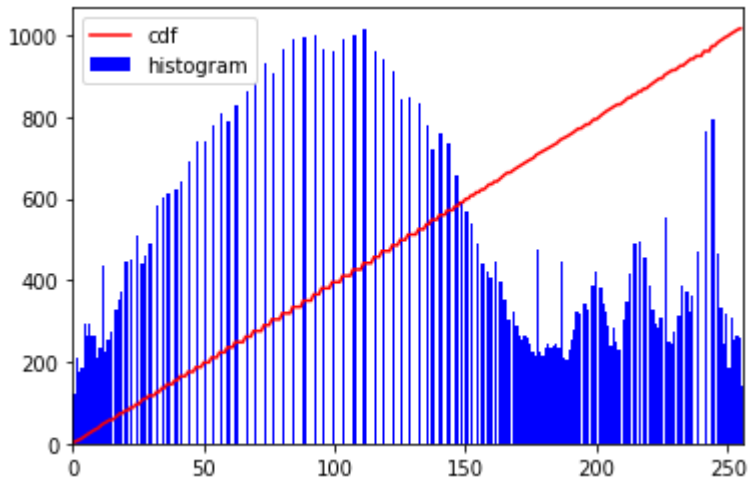
executed in 422ms, finished 19:11:24 2023-03-26



In [5]:

```
equ = cv2.equalizeHist(img)


hist,bins = np.histogram(equ.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()

plt.plot(cdf_normalized, color = 'r')
plt.hist(equ.flatten(),256,[0,256], color = 'b')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```

executed in 393ms, finished 19:11:24 2023-03-26



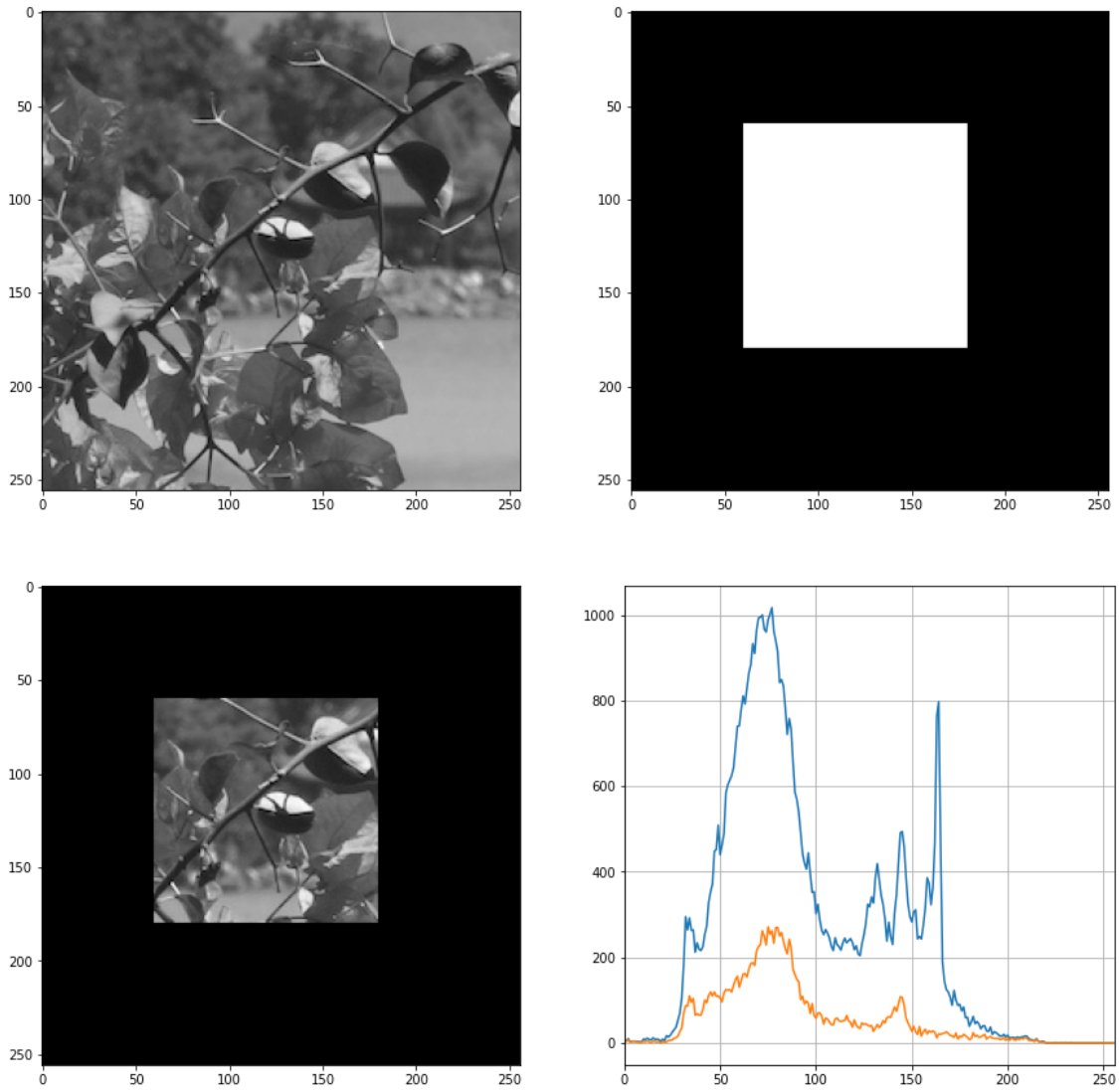
3.2 Histograma de uma máscara particular

O histograma de uma máscara particular ou região de interesse (ROI) pode ser útil em alguns casos para avaliar o histograma local em um dada área. O código s seguir define uma máscara e calcula o histograma para a máscara em particular.

In [6]:

```
plt.figure(figsize=(15,15))


img = cv2.imread('image_folder/westlake256.jpg',0)

# create a mask
mask = np.zeros(img.shape[:2], np.uint8)
mask[60:180, 60:180] = 255
masked_img = cv2.bitwise_and(img,img,mask = mask)

# Calculate histogram with mask and without mask
# Check third argument for mask
hist_full = cv2.calcHist([img],[0],None,[256],[0,256])
hist_mask = cv2.calcHist([img],[0],mask,[256],[0,256])
plt.subplot(221), plt.imshow(img, 'gray')
plt.subplot(222), plt.imshow(mask,'gray')
plt.subplot(223), plt.imshow(masked_img, 'gray')
plt.subplot(224), plt.grid(True), plt.plot(hist_full), plt.plot(hist_mask)
plt.xlim([0,256])
plt.show()
```

executed in 533ms, finished 19:11:25 2023-03-26



3.3 Projeção do histograma (posterior)

É usado para segmentação de imagem ou localização de objetos de interesse em uma imagem. Ele cria uma imagem do mesmo tamanho (mas de canal único) que a da nossa imagem de entrada, onde cada pixel corresponde à probabilidade desse pixel pertencer ao nosso objeto. A imagem de saída terá nosso objeto de interesse mais claro em comparação com a parte restante. simples). **A retroprojeção do histograma é usada com o algoritmo camshift, por exemplo.**

O processo parte com a **criação de um histograma de uma imagem contendo nosso objeto de interesse.** O objeto deve preencher a imagem tanto quanto possível para melhores resultados. E um histograma de cores é preferível ao histograma de tons de cinza, porque a cor do objeto é a melhor maneira de definir o objeto do que sua intensidade em tons de cinza. Em seguida, **projetamos de volta este histograma sobre nossa imagem de teste, onde precisamos encontrar o objeto, ou seja, calculamos a probabilidade de cada pixel pertencente ao solo e o mostramos.** A saída resultante na limiarização adequada nos dá o aterramento sozinho.

OpenCV fornece uma função *cv2.calcBackProject()*. Seus parâmetros são quase iguais aos da função *cv2.calcHist()*. Um de seus parâmetros é o histograma, que é o histograma do objeto e temos que encontrá-lo. Além disso, **o histograma do objeto deve ser normalizado antes de passar para a função de backproject**. Ele retorna a imagem de probabilidade.

Em seguida, envolvemos a imagem com um núcleo de disco e aplicamos o limiar. Abaixo está meu código e saída:

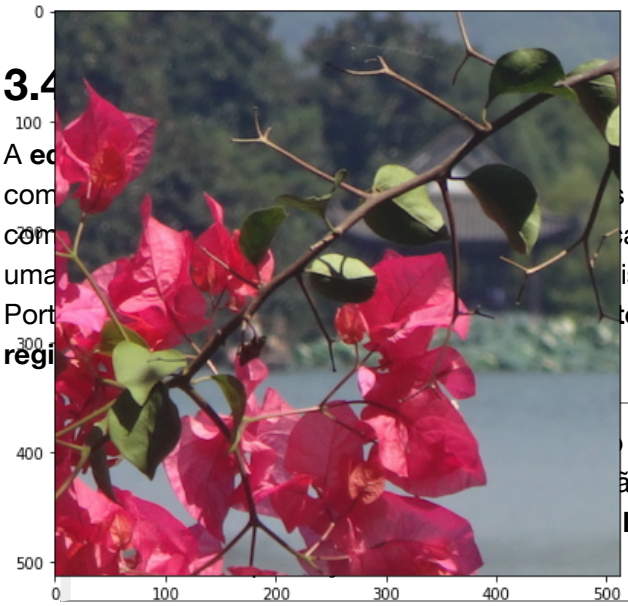
In [7]:

```
import cv2
import numpy as np

plt.figure(figsize=(15,15))

roi = cv2.imread('book_folder/westlake_red_patch.png')
hsv = cv2.cvtColor(roi,cv2.COLOR_BGR2HSV)

target = cv2.imread('book_folder/westlake512.png')
hsvt = cv2.cvtColor(target,cv2.COLOR_BGR2HSV)

# calculating object histogram
roihist = cv2.calcHist([hsv],[0, 1], None, [180, 256], [0, 180, 0, 256] )

# normalize histogram and apply backprojection
cv2.normalize(roihist,roihist,0,255,cv2.NORM_MINMAX)
dst = cv2.calcBackProject([hsvt],[0,1],roihist,[0,180,0,256],1)

# Now convolute with circular disc
disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
cv2.filter2D(dst,-1,disc,dst)

# threshold and binary AND
ret,thresh = cv2.threshold(dst,50,255,0)
thresh = cv2.merge((thresh,thresh,thresh))
res = cv2.bitwise_and(target,thresh)
res = np.vstack((target,thresh,res))

#cv2.imwrite('book_folder/result.jpg',res)

plt.subplot(2,2,1)
plt.imshow(cv2.cvtColor(target, cv2.COLOR_BGR2RGB))

plt.subplot(2,2,2)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

executed in 471ms, finished 19:11:25 2023-03-26

Out[7]:

```
<matplotlib.image.AxesImage at 0x7f94391b88e0>
```



3.4

A **ec** é uma técnica de proces agem de com imagens. Ele difere da ec istograma com alcula vários histogramas espondendo a uma istribuir os valores de lum nagem. Port **te local e realçar as defi las em cada regi**

ruído em regiões relativa meas de ão adaptativa do histogra **limitada pelo contraste** isso ao

O CLAHE implementa o filtro "Contraste Limited Adaptive Histogram Equalization". A implementação suporta apenas imagens e saídas não assinadas de 16 bits e sempre gera uma imagem no intervalo 0 - 65535. Para combinar a imagem CLAHE com a imagem original, ela deve ser redimensionada para o intervalo mínimo/máximo da imagem original em um pós-processamento Passo.

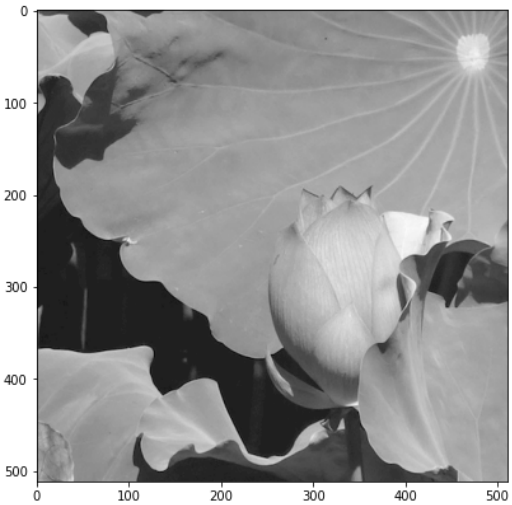
In [8]:

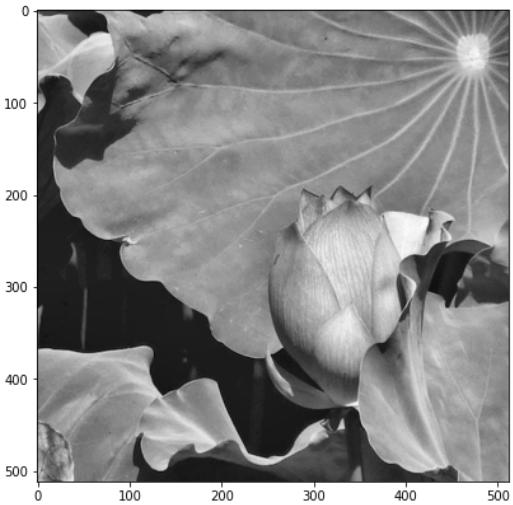
```
import numpy as np
import cv2
from matplotlib import pyplot as plt

plt.figure(figsize=(15,15))

img = cv2.imread('image_folder/lotus512.png',0)


# create a CLAHE object (Arguments are optional).
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)

##cv2.imwrite('clahe_2.jpg',cl1)

plt.subplot(2,2,1)
plt.imshow(img, 'gray')

plt.subplot(2,2,2)
plt.imshow(cl1, 'gray')

plt.subplot(2,2,3)
plt.grid(True)

plt.hist(img.ravel(),256,[0,256]);
plt.xlim([0,256])

plt.subplot(2,2,4)
plt.grid(True)

plt.hist(cl1.ravel(),256,[0,256])
plt.xlim([0,256])
```

executed in 1.13s, finished 19:11:26 2023-03-26

Out[8]:

```
(0.0, 256.0)
```


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500