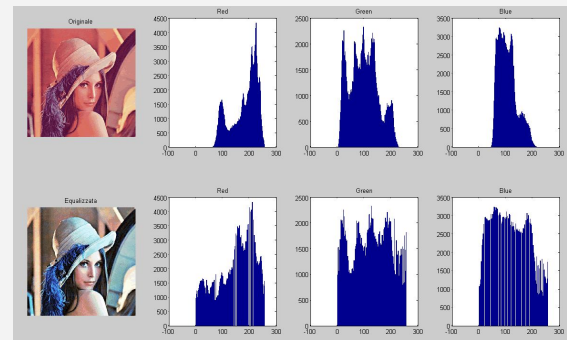


Processamento de Imagens

Anderson Fortes

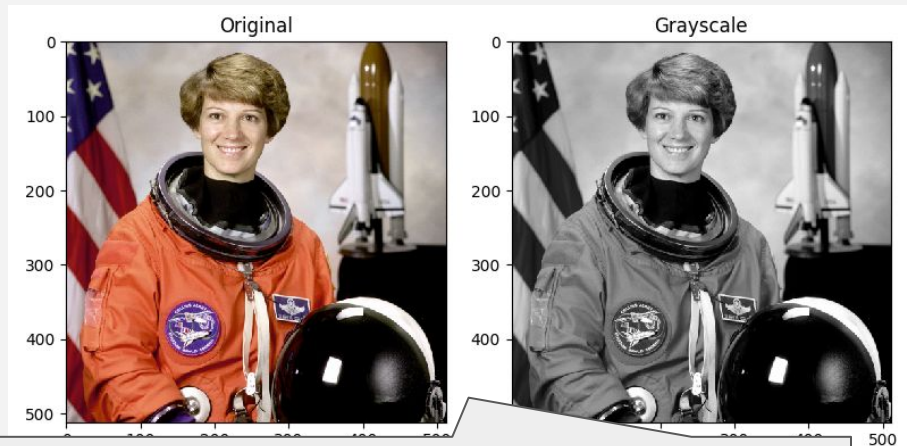


Na aula passada



Convertendo imagem RGB em tons de cinza

Por quê converter uma imagem colorida para tons de cinza?



Em algumas aplicações é necessário trabalhar com as cores. Veremos como trabalhar com cores e não ter o problema do processamento em 3 canais do RGB.

a conversão de
quase sempre

tons de cinza é uma tarefa

Como é convertida a imagem RGB em tons de cinza



Iremos converter a imagem *frutas.png* para tons de cinza. Crie uma função que recebe uma imagem e retorna ela em tons de cinza.



Espaço de cores HSV

Em sistemas de Visão Computacional e Processamento de Imagens, o espaço de cor HSV é um dos modelos principais e mais utilizados para a representação de cor.

Qual era o problema mesmo em processar imagens RGB?



Segmentando canais em uma imagem HSV



Assim como as imagens em RGB podem ser segmentadas em três canais distintos com a função `split` da biblioteca OpenCV, o mesmo pode ser feito com imagens representadas no espaço HSV.

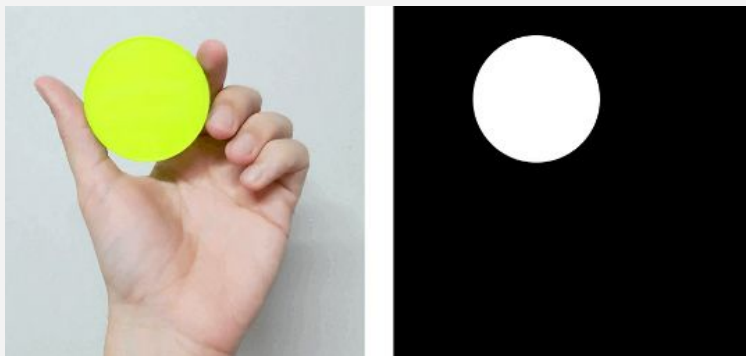
```
1  import cv2
2
3  img = cv2.imread("img/frutas.png")
4
5  img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
6
7  h,s,v = cv2.split(img_hsv)
8
9  cv2.imshow("matiz",h)
10 cv2.imshow("saturação",s)
11 cv2.imshow("valor",v)
12
13 img_rgb = cv2.merge((h,s,v))
14
15 img_rgb = cv2.cvtColor(img_rgb,cv2.COLOR_HSV2BGR)
16
17 cv2.imshow("RGB restored", img_rgb)
```

A importância do HSV

Rastrear objetos coloridos em movimento também é uma aplicação muito comum em sistemas baseados em Visão Computacional.

Esta técnica é frequentemente usada para acompanhar a movimentação de um robô, pessoas vestindo roupas de uma determinada cor, bolas coloridas em jogos e etc.

A imagem a seguir exibe um frame de uma captura em tempo real. Nesse frame, um objeto de interesse amarelo foi segmentado, possibilitando o rastreamento da sua posição na imagem.



Informações de dimensões da imagem



```
import cv2

img = cv2.imread("img/bolinhas.png")

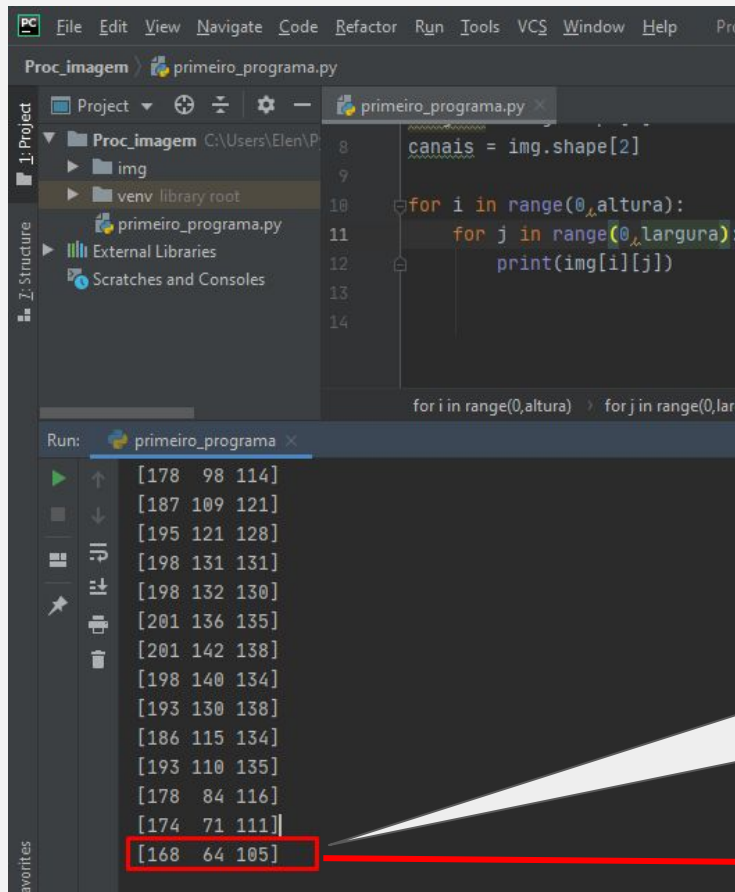
altura = img.shape[0]
largura = img.shape[1]
canais = img.shape[2]

print("A imagem é de tamanho {} de altura por {} de largura e contém {} canais.".format(altura, largura, canais))
```

```
C:\Users\Elen\PycharmProjects\Proc_imagem\venv\Scripts\python.exe C:/Users/Elen
A imagem é de tamanho 345 de altura por 619 de largura e contém 3 canais.
```

```
Process finished with exit code 0
```


Acessar cada um dos pixels

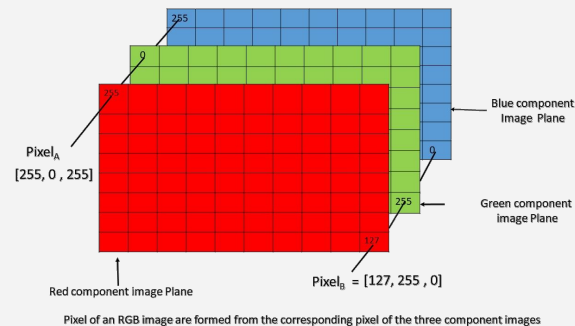


The screenshot shows an IDE with a project named 'Proc_imagem'. The file 'primeiro_programa.py' is open, showing a script that iterates over the height and width of an image to print its pixel values. The output window shows a list of pixel values, with the last one, `[168 64 105]`, highlighted by a red box. A red arrow points from this box to the text 'Em python é B G R'.

```
canais = img.shape[2]
for i in range(0, altura):
    for j in range(0, largura):
        print(img[i][j])
```

Run: primeiro_programa

[178 98 114]
[187 109 121]
[195 121 128]
[198 131 131]
[198 132 130]
[201 136 135]
[201 142 138]
[198 140 134]
[193 130 138]
[186 115 134]
[193 110 135]
[178 84 116]
[174 71 111]
[168 64 105]



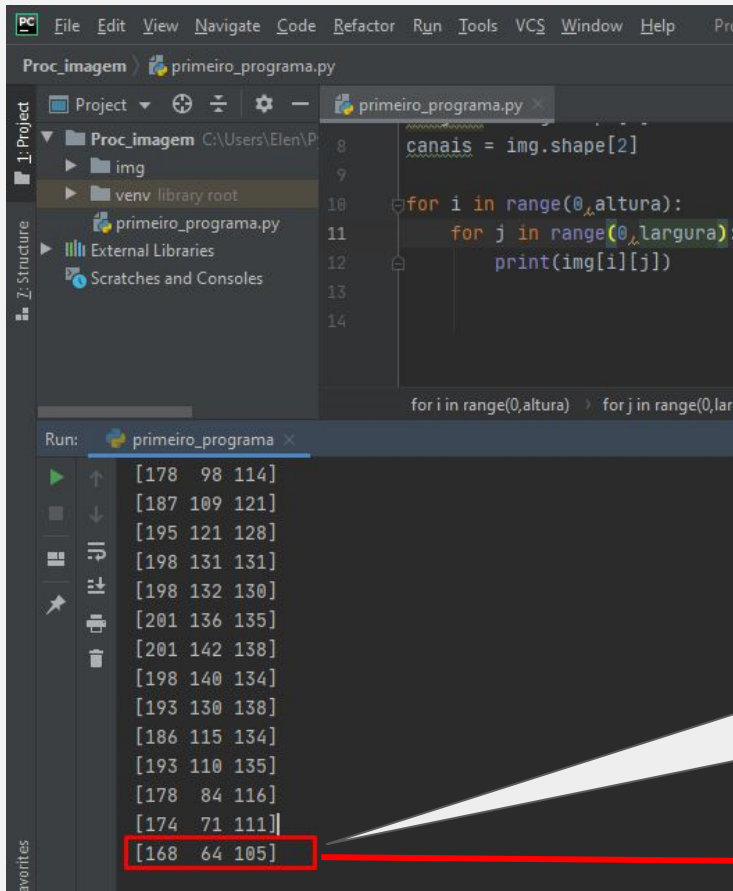
**Então podemos
acessar em cada
um dos canais...**

Em python é **B G R**

Acessar cada um dos pixels

```
primeiro_programa.py x
8  canais = img.shape[2]
9
10 for i in range(0, altura):
11     for j in range(0, largura):
12         for h in range(0, 3):
13             print(img[i][j][h])
14
```

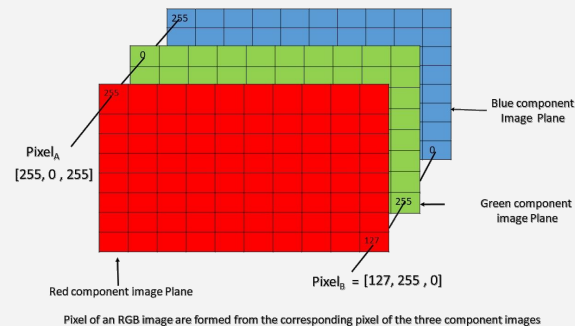
Acessar cada um dos pixels



The screenshot shows an IDE with a project named 'Proc_imagem'. The file 'primeiro_programa.py' is open, containing the following code:

```
8  canais = img.shape[2]
9
10 for i in range(0, altura):
11     for j in range(0, largura):
12         print(img[i][j])
13
14
```

The output window shows a list of pixel values. The last value, `[168 64 105]`, is highlighted with a red box. A red arrow points from this box to the text 'Em python é B G R'.



**Então podemos
acessar em cada
um dos canais...**

Em python é **B G R**

Acessar cada um dos pixels

```
primeiro_programa.py x
8  canais = img.shape[2]
9
10 for i in range(0, altura):
11     for j in range(0, largura):
12         for h in range(0, 3):
13             print(img[i][j][h])
14
```

Alterar valor de pixels



Na imagem *beira2.png*, os pixels em que o canal vermelho seja bem intenso (com valores altos - acima de 180) vamos transformar em um azul claro.

Alterar valor de pixels



Na imagem *beira2.png*, os pixels em que o canal vermelho seja bem intenso (com valores altos - acima de 240) vamos transformar em um azul claro.

```
1  import cv2
2  img= cv2.imread("img/beira3.png")
3
4  cv2.imshow("Beira rio", img)
5  cv2.waitKey(0)
6
7  azul = (231,172,65)
8  altura = img.shape[0]
9  largura = img.shape[1]
10
11  for i in range(0,altura):
12      for j in range(0,largura):
13          if img[i][j][2] >240:
14              img[i][j] = azul
15
16  cv2.imshow("Beira rio 2", img)
17  cv2.waitKey(0)
```

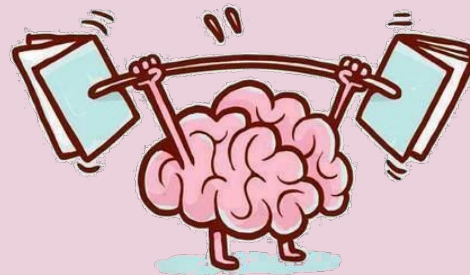
Atividade

Na pasta baixada, existe uma imagem chamada *arena.png*.

Você deve criar um código que abra e mostre essa imagem. A seguir, deve varrer os pixels dela e nos que a faixa azul for maior que 200, transformar em um vermelho forte.

Prosseguiremos após todos concluírem.

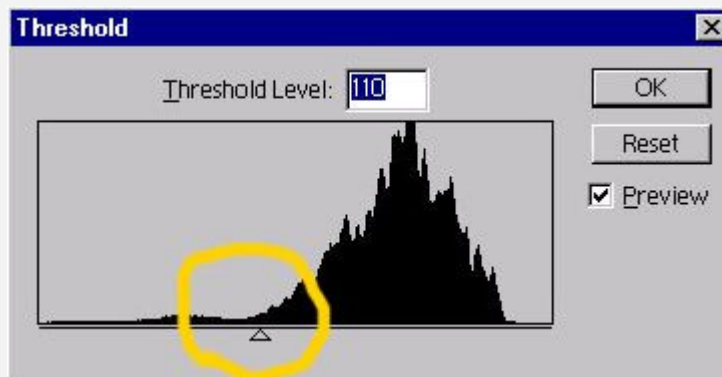
Realizar a entrega no classroom.



Histograma de cores

O histograma de cores de uma imagem é a distribuição de frequência dos níveis de cinza em relação ao número de amostras.

Essa distribuição nos fornece informações sobre a qualidade da imagem, principalmente no que diz respeito à intensidade luminosa e ao contraste.



Histograma de cores



Para entender, vamos ver o histograma de uma imagem binária:

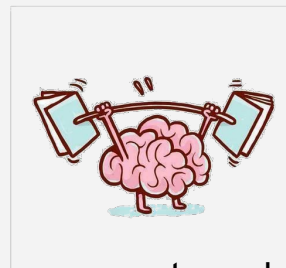
Por possuir pixels representados apenas pela cor preta ou branca, o histograma de cores de uma imagem binária pode ser facilmente obtido.

O total de pixels da imagem subtraído do total de pixels de uma determinada cor nos fornece a quantidade de pixels brancos ou pretos representados.

O total de pixels pretos ou brancos pode ser obtido percorrendo toda a matriz que representa a imagem, contando-os individualmente,

Histograma de uma imagem binária

Antes de mais nada, vamos exercitar o aprendizado. Para isso, vamos:



1 - transformar a imagem da folha colorida (*folha_color.png*) em uma imagem em tons de cinza (Use a função criada anteriormente).

2 - abrir a imagem cinza e transformar ela em uma imagem binária. Para isso, vamos determinar um limiar adequado. Devemos salvar a imagem binária como **.bmp**.

3 - Por fim, vamos analisar quantos pixels pretos e quantos brancos a imagem possui.



Histograma de uma imagem binária

Antes de mais

so, vamos:



1 - transformar
função criada a

em uma imagem em tons de cinza (Use a

2 - abrir a imag
determinar um

inária. Para isso, vamos
inária como **.bmp**.

3 - Por fim, vamos analisar

o pixels pretos e quantos brancos a imagem possui.

Note como a imagem resultante apresenta ruído. Esse ruído é normal quando se realiza a “binarização” de uma imagem. Na próxima aula veremos sobre como eliminar esse tipo de ruído.



Histograma de uma imagem binária



Histograma de uma imagem binária



A biblioteca **Matplotlib** possui a função `hist`, que, além de abstrair essa tarefa, gera a figura que ilustra o histograma de cores da imagem.

Vamos ver como funciona usando a imagem da folha que acabamos de binarizar.

```
import cv2
from matplotlib import pyplot

img = cv2.imread("img/folha_binaria.bmp", 0)

pyplot.hist(img.ravel(), 256, [0, 255])

pyplot.show()
```

parâm. 1: Imagem

parâm. 2: Quantidade de elementos distintos a serem representados

parâm. 3: Intervalo a ser mostrado no gráfico.

O método `ravel`, aplicado no primeiro parâmetro, tem a finalidade de transformar a imagem em um vetor.

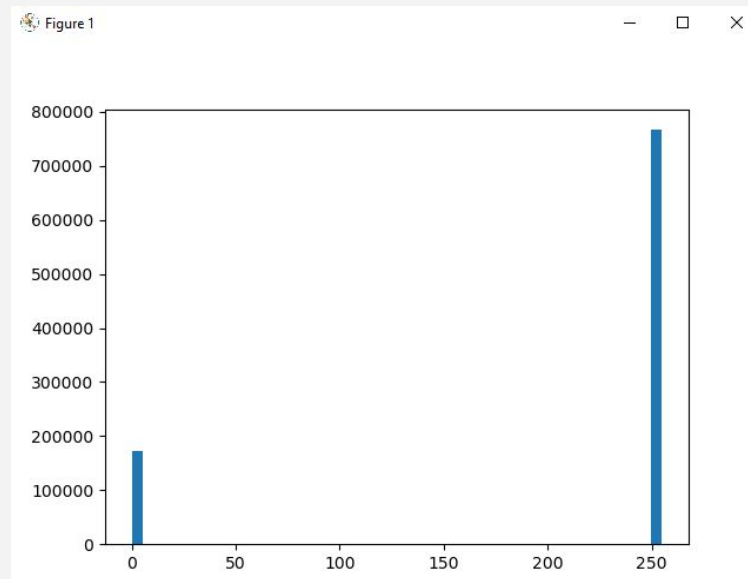
Histograma de uma imagem binária

```
import cv2
from matplotlib import pyplot

img = cv2.imread("img/folha_binaria.bmp", 0)

pyplot.hist(img.ravel(), 256, [0, 255])

pyplot.show()
```



Histograma de uma imagem em Tons de Cinza



Nas imagens em tons de cinza, por haver grande diversidade de tons, o histograma pode ser representado em classes que contabilizam os valores em determinados intervalos.

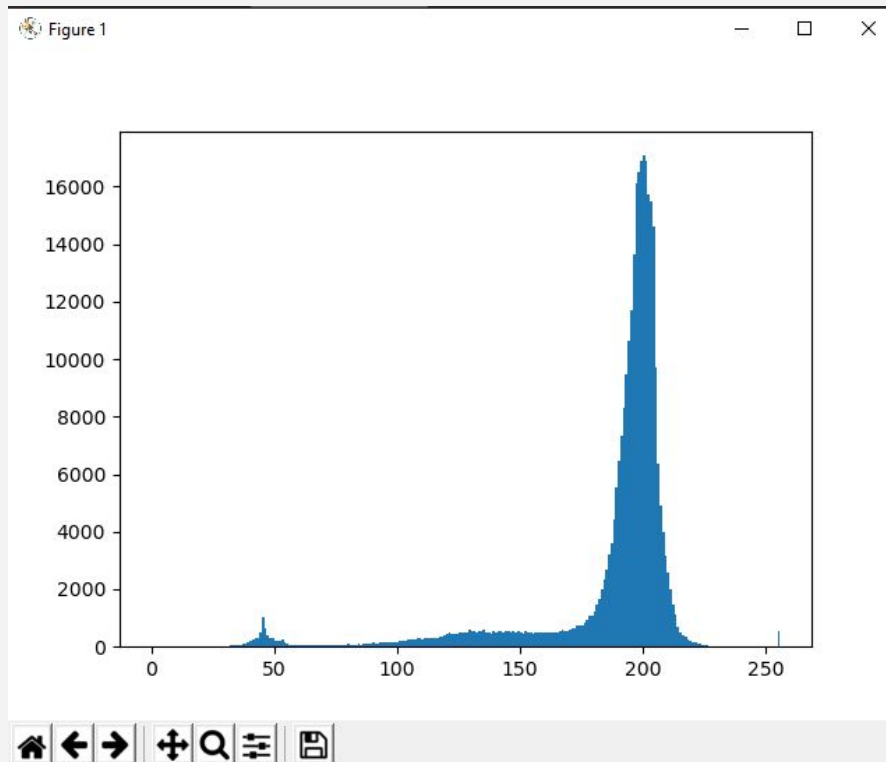
Quanto maior o número de classes, mais informações sobre a imagem podem ser representadas. Por se tratar de uma imagem em 8 bits, o número máximo de classes para representar os intervalos é 256.

Executando o mesmo código apresentado no tópico anterior, carregando a imagem em tons de cinza da folha, obtemos o histograma ilustrado na figura a seguir.

Uma das informações que ele nos fornece é o nível de contraste da imagem.

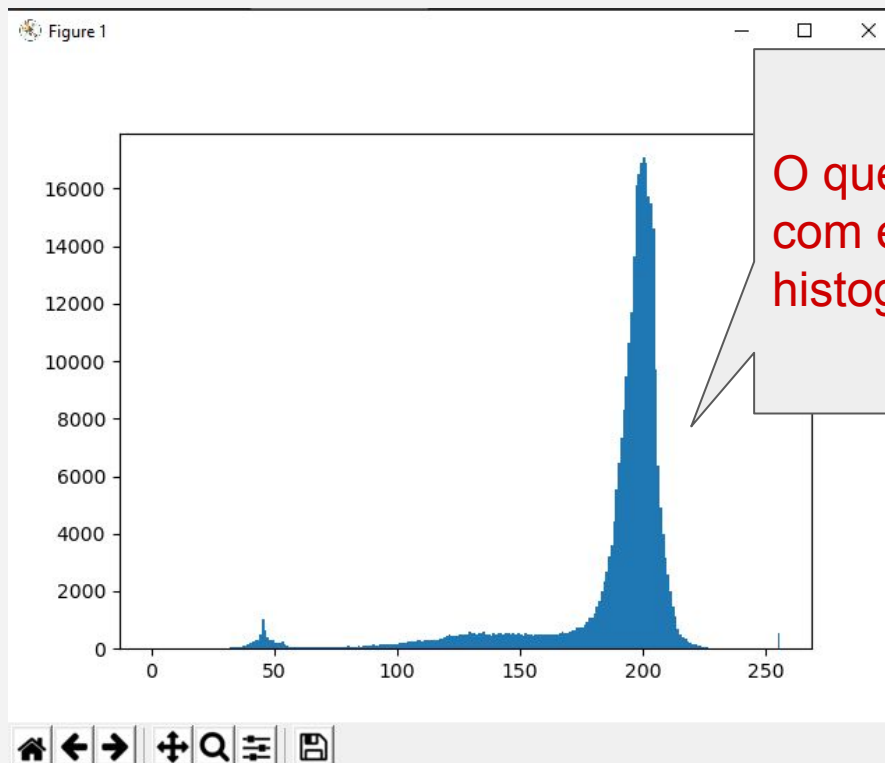
Histograma de uma imagem em Tons de Cinza

Vamos transformar a imagem ***folha_color.png*** em uma imagem **cinza** e verificar seu histograma.



Histograma de uma imagem em Tons de Cinza

Vamos transformar a imagem ***folha_color.png*** em uma imagem **cinza** e verificar seu histograma.

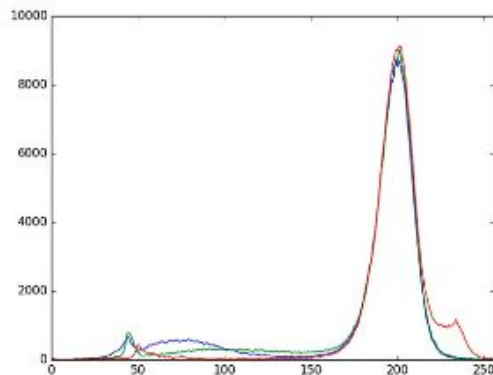


O que podemos notar com esse histograma?

Histograma de uma imagem colorida

Uma imagem colorida possui um histograma para cada canal de cor. Uma vez que cada canal é representado por pixels em tons de cinza, o mesmo método estudado nos tópicos anteriores pode ser utilizado para plotar esses histogramas.

Vamos usar o mesmo método que utilizamos para imagem em tons de cinza.



Observe o histograma do canal vermelho. Ele possui uma curva entre os valores 200 e 250 que se diferencia consideravelmente do canal azul e verde. Isso ocorre porque a folha ilustrada na fotografia, além de possuir tons avermelhados, ocupa uma área significativa da imagem.

Atividade 1 - Entregar ao final da aula



Abrir a imagem *chaves1.jpeg* e realizar a separação dos seus 3 canais RGB.

Após, mostrar o histograma de cores de cada um dos canais.

Utiliza **pyplot.figure("Nome da figura")** para gerar uma figura nova para cada um dos 3 histogramas.

Histogramas são muito importantes



Os histogramas tem outra importante finalidade: Eles podem ajudar a realçar os objetos, tornando-os mais nítidos, utilizando uma técnica chamada **Equalização de Histograma**.

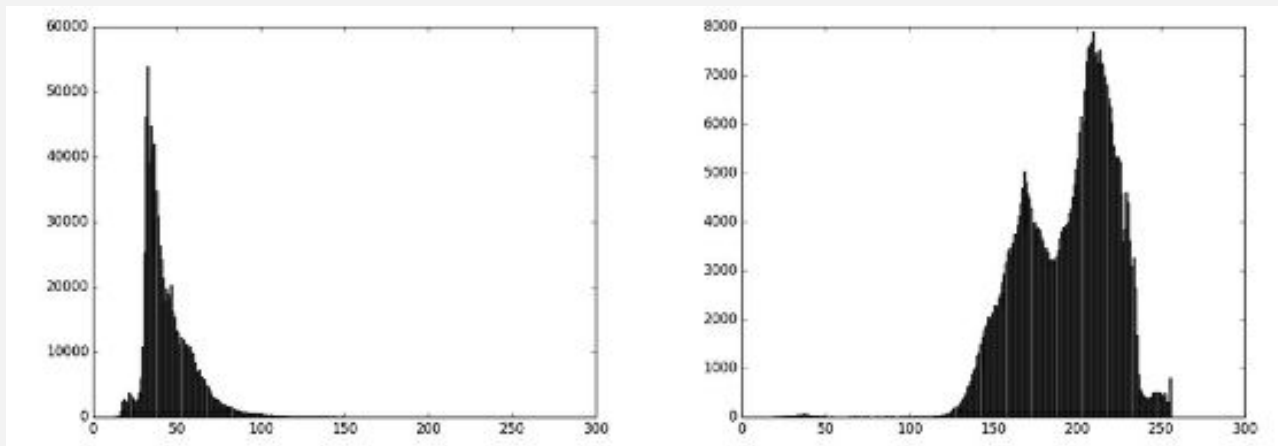
A exposição à luz da cena capturada e o nível de contraste da fotografia são dados importantes que podem ser extraídos de um histograma.

Histogramas são muito importantes



Imagens **superexpostas** (ou seja, com alto nível de luminosidade) geralmente apresentam histogramas com a maior parte dos elementos **concentrados à direita**.

Histogramas que possuem maior parte dos elementos **concentrados à esquerda** tendem a representar imagens subexpostas, com **baixo nível de luminosidade**.



Histogramas

Os histogramas tem ou
mais nítidos, utilizando

A exposição à luz da ce
podem ser extraídos de

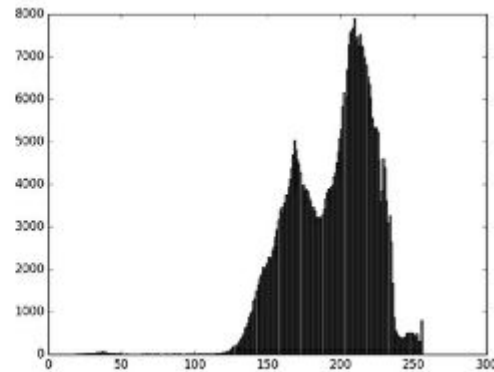
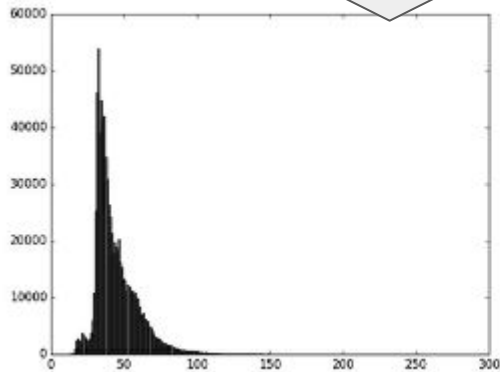
Imagens superexpostas
com a maior parte dos
parte dos elementos co
nível de luminosidade.

Assim como podemos olhar para uma fotografia observando suas cores, tonalidades e contrastes, os dados tabulados de um histograma permitem através dos números **que computadores também possam enxergar essas características em uma imagem.**

nando-os

ntes que

istogramas
sueem maior
com baixo



Histograma de uma imagem colorida



Imagens com baixo nível de contraste apresentam menor nitidez, sendo caracterizadas por **histogramas estreitos**, nos quais os elementos estão concentrados em **intervalos menores**.

Do contrário, imagens com alto nível de contraste apresentam **histogramas largos**, com **elementos distribuídos por toda faixa de tons de cinza disponível**.

Dependendo do ambiente onde a captura da imagem é realizada, o objeto de interesse pode não apresentar a nitidez desejada.

Atividade 2

Plotar o histograma das imagens *chaves1.jpeg*, *chaves2.jpeg* e *chaves3.jpeg* verificar se os histogramas permitem notar a diferença na iluminação das imagens.



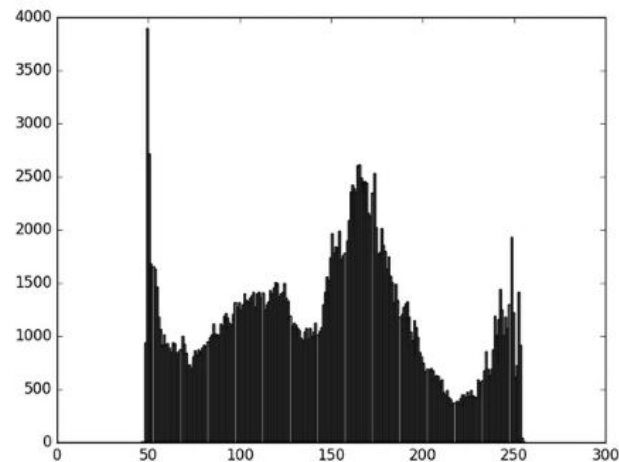
Equalização de Histograma

Imagens com baixo nível de contraste apresentam menor nitidez, sendo caracterizadas por **histogramas estreitos**, nos quais os elementos estão concentrados em **intervalos menores**.

Do contrário, imagens com alto nível de contraste apresentam **histogramas largos**, com elementos **distribuídos por toda faixa de tons de cinza disponível**.

Dependendo do ambiente onde a captura da imagem é realizada, o objeto de interesse pode não apresentar a nitidez desejada.

Vamos imaginar um software que precise ler os ponteiros dessa máquina.

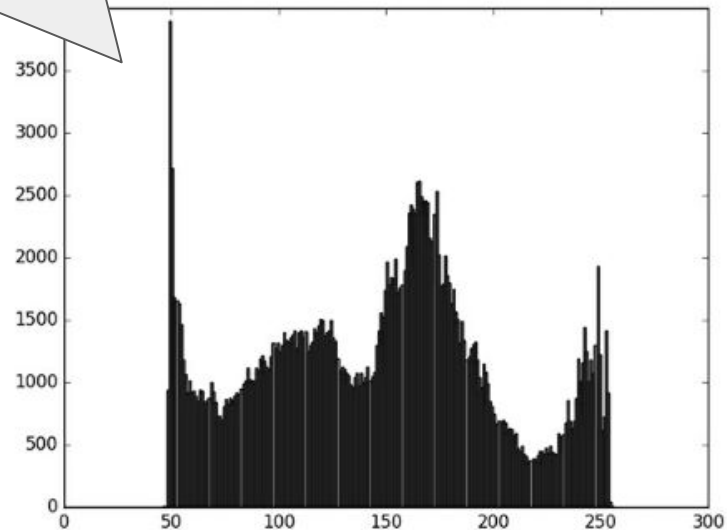


No eixo horizontal nenhum elemento foi representado em tons de cinza variando de 0 a 50. Isso significa que os elementos não estão distribuídos por toda a faixa de tons disponível, condição ideal para representar uma fotografia com nível de contraste apropriado. O objetivo da equalização de histograma é justamente modificar a tonalidade dos pixels da imagem, com intuito de redistribuir o histograma.

zadas por histogramas

os, com elementos

Dependendo do ambiente
apresentar a nitidez de



Equalização de Histograma



A função **equalizeHist** permite equalizar histogramas de imagens. Ela possui como único parâmetro obrigatório a matriz que representa a imagem carregada. Executando essa função, uma nova imagem com o histograma equalizado será retornada.

Vamos ver o que acontece com a imagem dos ***ponteiros.png***

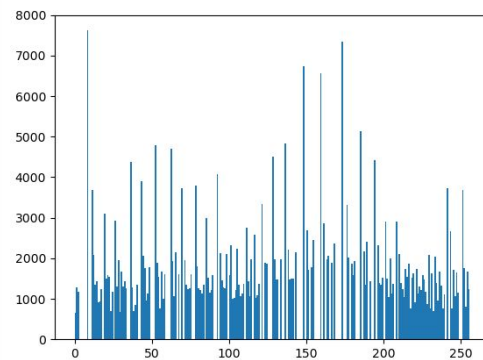
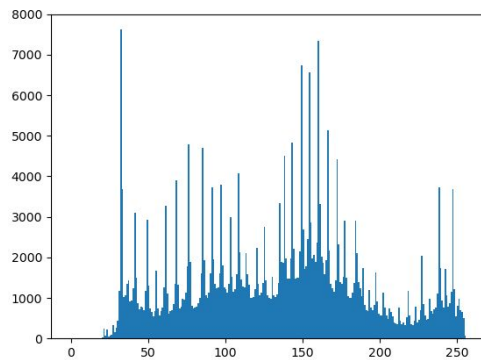
Equalização de Histograma

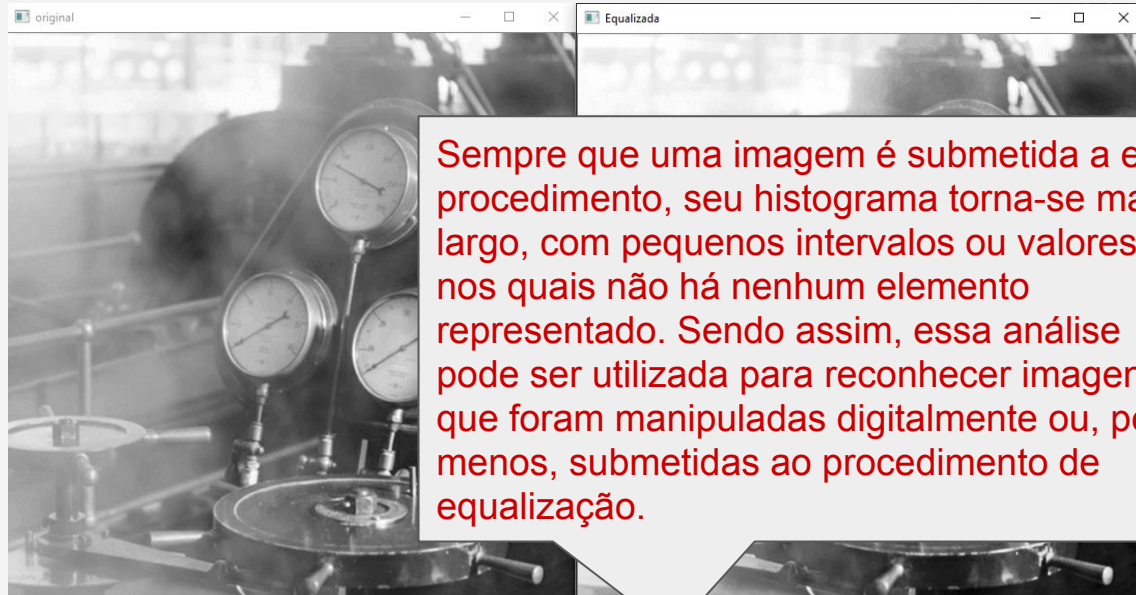


A função **equalizeHist** permite equalizar histogramas de imagens. Ela possui como único parâmetro obrigatório a matriz que representa a imagem carregada. Executando essa função, uma nova imagem com o histograma equalizado será retornada.

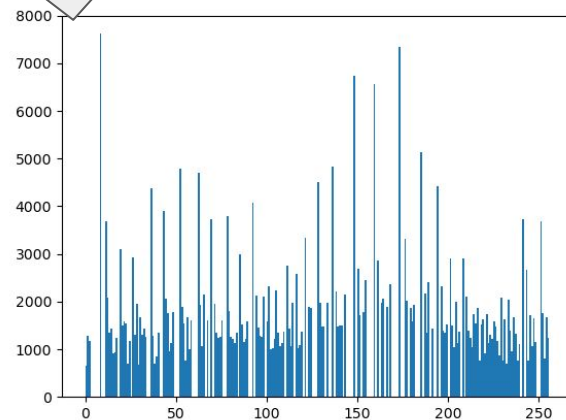
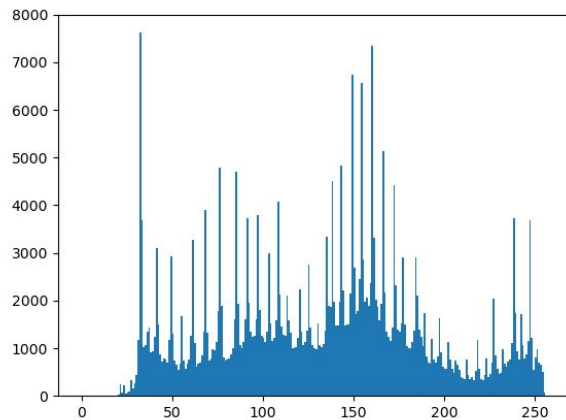
Vamos ver o que acontece com a imagem dos *ponteiros.png*

```
1 import cv2
2 from matplotlib import pyplot
3
4 img = cv2.imread("img/ponteiros.png", 0)
5
6 equa = cv2.equalizeHist(img)
7
8 cv2.imshow("original", img)
9 cv2.imshow("Equalizada", equa)
10 cv2.waitKey(0)
11
12 pyplot.hist(img.ravel(), 256, [0, 256])
13 pyplot.figure()
14 pyplot.hist(equa.ravel(), 256, [0, 256])
15
16 pyplot.show()
```





Sempre que uma imagem é submetida a esse procedimento, seu histograma torna-se mais largo, com pequenos intervalos ou valores nos quais não há nenhum elemento representado. Sendo assim, essa análise pode ser utilizada para reconhecer imagens que foram manipuladas digitalmente ou, pelo menos, submetidas ao procedimento de equalização.



Equalização de Histograma - Cores



O método que acabamos de ver não funciona para imagens coloridas RGB.

Os valores que representam os pixels, em uma imagem em RGB, determinam tanto a intensidade quanto à cromaticidade do pixel. Aplicando esse mesmo método nelas, as cores reais sofreriam distorções, e o conteúdo cromático da imagem seria afetado.

Solução:

Convertê-la para o espaço de cor HSV. Nesse espaço, os canais H e S representam as informações relacionadas à cor; já o canal V representa somente a intensidade de luz referente ao pixel. Dessa maneira, após a conversão da imagem original em RGB para o espaço HSV, basta segmentar os canais, aplicar a função de equalização ao canal de intensidade (canal V) e, por fim, converter o resultado novamente para o espaço de cor RGB.

Atividade 3



Realizar a melhora de contraste da imagem *chaves3.jpeg*. Para isso, são necessárias as seguintes etapas:

- 1 - convertê-la para HSV
- 2 - separar os canais HSV suas 3 faixas (h, s, v)
- 3 - aplicar a **equalização de histograma** apenas no canal de intensidade de cores, o canal V.
- 4 - juntar novamente os canais HSV
- 5 - transformar para RGB e comparar com a imagem original RGB.
 - Confira se a equalização do histograma deixou a imagem mais nítida.
- 6 - salvar como *chaves3_equaliz.png*
- 7 - Enviar o arquivo .py na atividade do google classroom.