Universidade Federal do Pará

Instituto de Tecnologia

Faculdade de Engenharia da Computação

EC01039 – Computação Gráfica e Processamento de Imagem

Prof.: Ronaldo de Freitas Zampolo

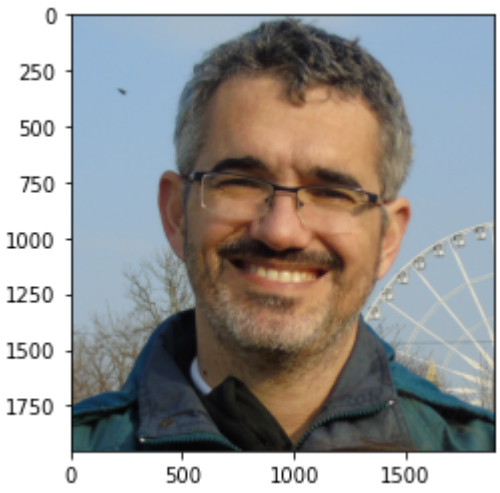
Aluno: ADRIANO MEDEIROS PINHEIRO

Mat.: 201806840060

Implementações 2

In [14]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

!wget https://github.com/AdrianoMedeirosPinheiro/PDI/blob/80239395b0d08466cfb47202c

img = mpimg.imread('C:/Users/Pichau/Pictures/zampolo.jpg')
imgplot1 = plt.imshow(img)
plt.show()
```

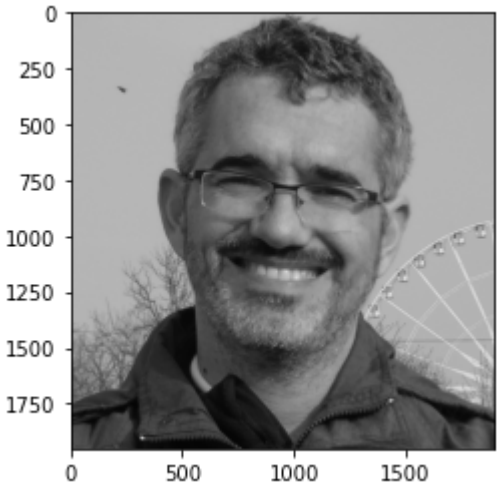
```
'wget' nÆo , reconhecido como um comando interno
ou externo, um programa oper vel ou um arquivo em lotes.
```



1. A partir de uma imagem RGB, implemente uma rotina para achar sua versão em tons de cinza. Considere nesta questão que o tom de cinza de cada pixel será simplesmente a média aritmética dos valores dos seus três canais cromáticos (R,G, e B). Ao final, arredonde os valores obtidos para os pixels sejam números inteiros.

In [6]:
```
img_gray = img.copy()
columms = np.shape(img)[0]
rows = np.shape(img)[1]
```

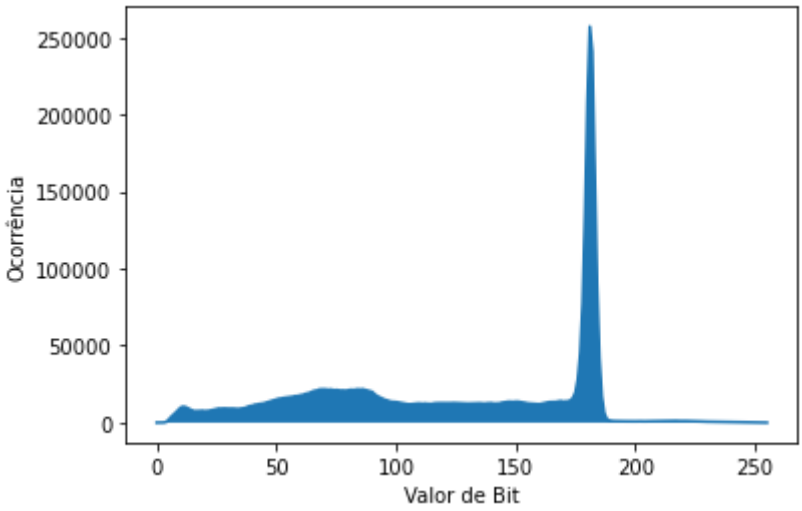
```
for x in range(columms):
  for y in range(rows):
    img_gray[x, y] = sum(img[x, y]) / 3
imgplot2 = plt.imshow(img_gray)
```



1. Crie uma rotina para calcular e exibir o histograma de uma imagem de tons de cinza.

In [8]:
```
def histogr(image):
  hist = np.zeros(256)
  for x in range(columms):
    for y in range(rows):
      hist[image[x, y, 0]] += 1
  return hist
hist1 = histogr(img_gray)
plt.plot(hist1)
plt.fill_between(range(len(hist1)), hist1)
plt.xlabel("Valor de Bit")
plt.ylabel("Ocorrência")
```

Out[8]:
```
Text(0, 0.5, 'Ocorrência')
```



1. Implemente a transformação de intensidade

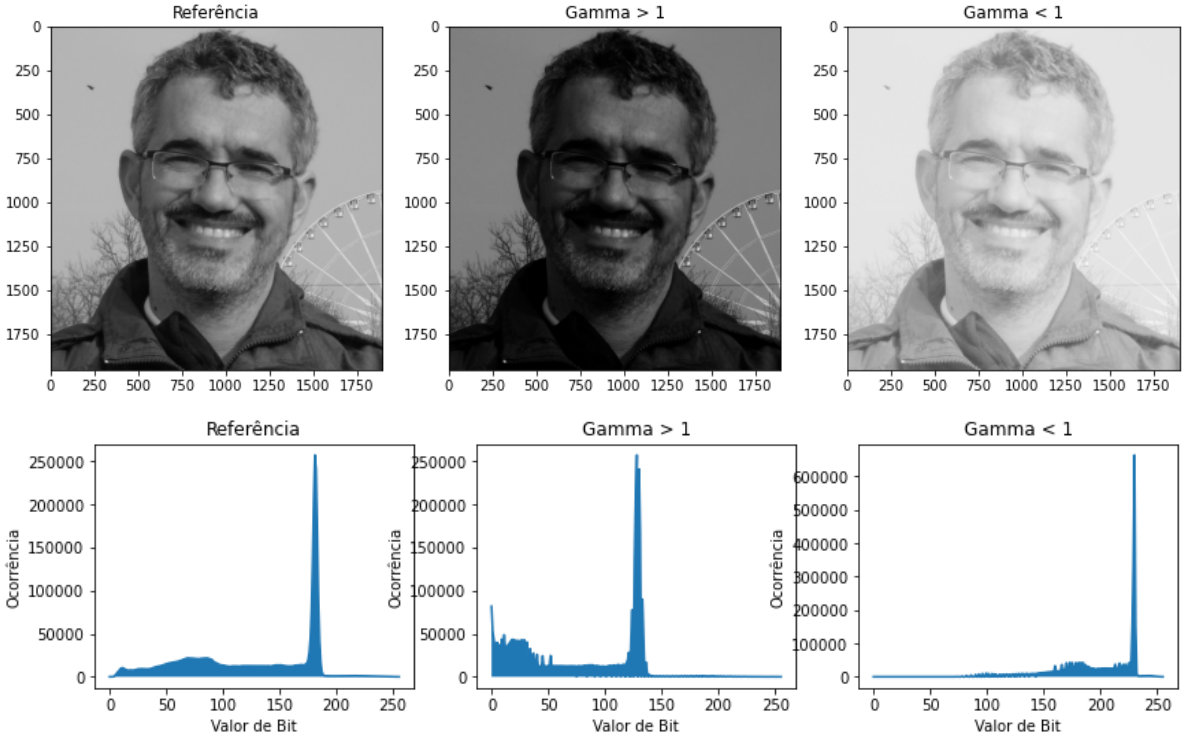
g(x, y) = k * [f(x, y)] ^ γ .

Ajuste o valor de k para que os valores dos pixels resultantes g(x, y) continuem no mesmo intervalo dos valores dos pixels da imagem de referência f(x, y), ou seja [0, 255]. Teste para

dois valores de γ (um maior que 1 e outro menor que 1) à escolha. Para cada γ, mostre a imagem de referência, a imagem resultante, o histograma da imagem de referência, o histograma da imagem resultante e gráfico das funções de mapeamento (as duas funções no mesmo gráfico, se quiser).

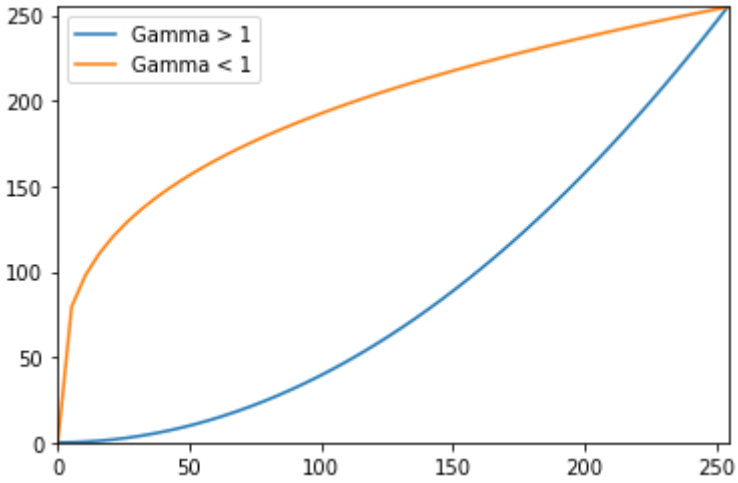
In [10]:
```
k1 = 0.003937
gamm1 = 2
k2 = 48.37227
gamm2 = 0.3
img2 = img.copy()
img3 = img.copy()

for x in range(columms):
  for y in range(rows):
    img2[x, y] = k1 * (img_gray[x, y].astype(np.float) ** gamm1)
    img3[x, y] = k2 * (img_gray[x, y].astype(np.float) ** gamm2)

hist2 = histogr(img2)
hist3 = histogr(img3)

fig1 = plt.figure(figsize=(20, 20))
ax1 = fig1.add_subplot(2, 4, 1)
ax1.title.set_text('Referência')
plt.imshow(img_gray)
ax2 = fig1.add_subplot(2, 4, 2)
ax2.title.set_text('Gamma > 1')
plt.imshow(img2)
ax3 = fig1.add_subplot(2, 4, 3)
ax3.title.set_text('Gamma < 1')
plt.imshow(img3)

fig2 = plt.figure(figsize=(13, 3))
ax4 = fig2.add_subplot(1, 3, 1)
plt.plot(hist1)
plt.fill_between(range(len(hist1)), hist1)
plt.xlabel("Valor de Bit")
plt.ylabel("Ocorrência")
ax4.title.set_text('Referência')
ax5 = fig2.add_subplot(1, 3, 2)
plt.plot(hist2)
plt.fill_between(range(len(hist2)), hist2)
plt.xlabel("Valor de Bit")
plt.ylabel("Ocorrência")
ax5.title.set_text('Gamma > 1')
ax6 = fig2.add_subplot(1, 3, 3)
plt.plot(hist3)
plt.fill_between(range(len(hist3)), hist3)
plt.xlabel("Valor de Bit")
plt.ylabel("Ocorrência")
ax6.title.set_text('Gamma < 1')
```

```
C:\Users\Pichau\AppData\Local\Temp\ipykernel_2500\1548054258.py:10: DeprecationWar
ning: `np.float` is a deprecated alias for the builtin `float`. To silence this wa
rning, use `float` by itself. Doing this will not modify any behavior and is safe.
If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations
  img2[x, y] = k1 * (img_gray[x, y].astype(np.float) ** gamm1)
C:\Users\Pichau\AppData\Local\Temp\ipykernel_2500\1548054258.py:11: DeprecationWar
ning: `np.float` is a deprecated alias for the builtin `float`. To silence this wa
rning, use `float` by itself. Doing this will not modify any behavior and is safe.
If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations
  img3[x, y] = k2 * (img_gray[x, y].astype(np.float) ** gamm2)
```



In [12]:
```
r = np.linspace(0, 256)
s1 = k1 * (r ** gamm1)
s2 = k2 * (r ** gamm2)
plt.xlim([0, 255])
plt.ylim([0, 255])
plt.plot(r, s1, label="Gamma > 1")
plt.plot(r, s2, label="Gamma < 1")
plt.legend(loc='best')
```

Out[12]:
```
<matplotlib.legend.Legend at 0x209148328b0>
```



In [ ]: