
FIAP

1. Relacionamento e operações entre imagens

Objetivos da aula:

- Compreender e praticar relacionamento e operações entre imagens

Operações aritméticas

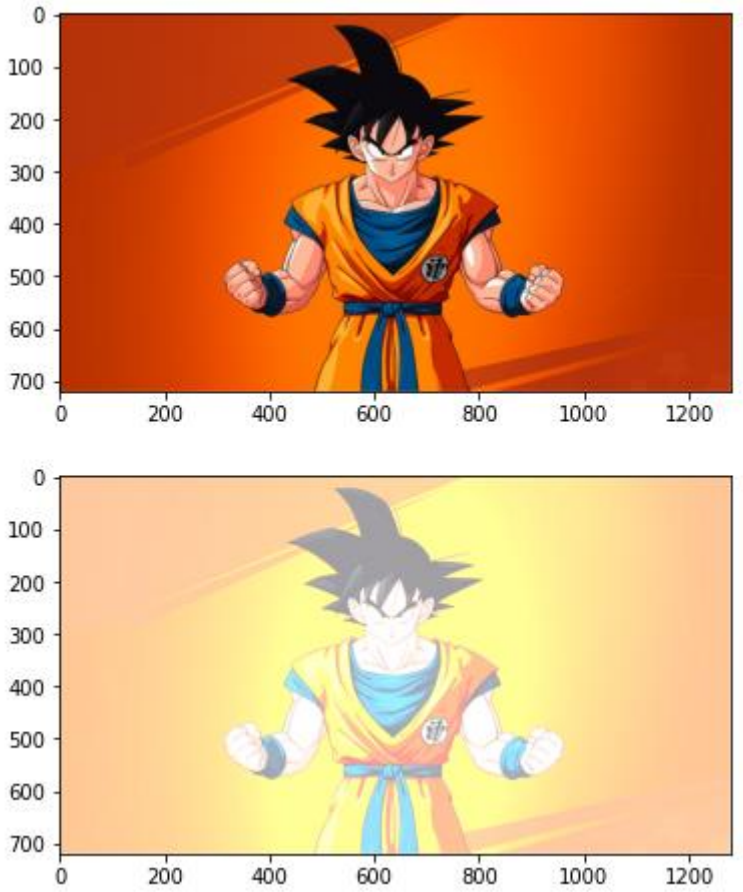
Sim, por que não?? lembre-se que uma imagem nada mais é que uma matriz. Logo podemos aplicar as mais diversas operações matemáticas.

Soma e Subtração

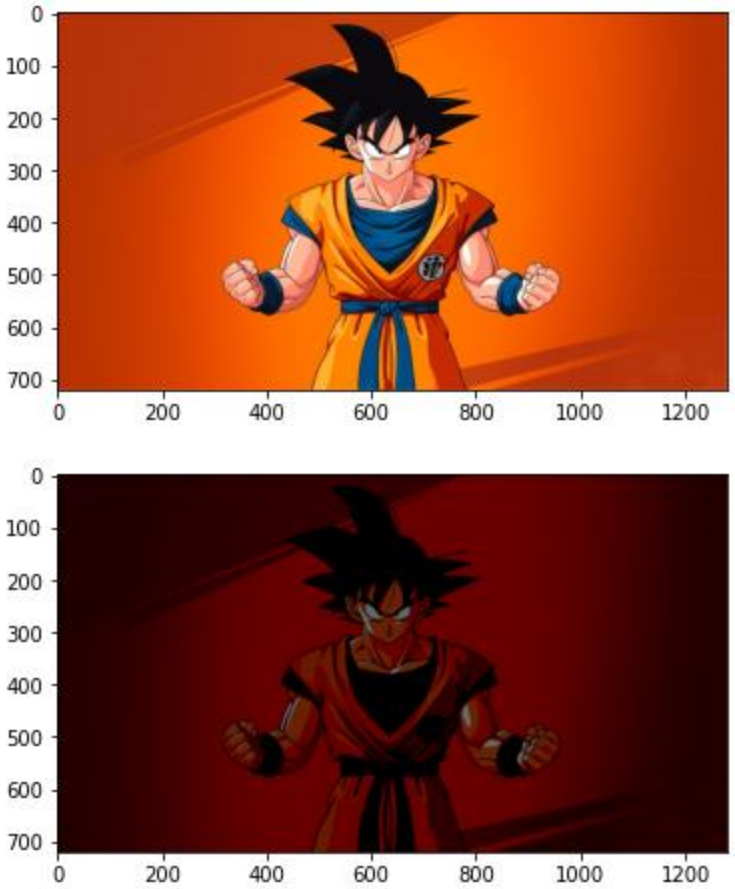
Podemos aplicar operações matemáticas para alterar o brilho e contraste de uma imagem. Na OpenCV usamos as funções ***cv2.add()*** e ***cv2.subtract()***

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


img = cv2.imread("goku.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# cria uma matriz com o mesmo dimensional da imagem original, mas com valores 100
matriz = np.ones(img.shape, dtype="uint8") * 150



img2 = cv2.add(img, matriz)


plt.imshow(img);
plt.show()
plt.imshow(img2);
```

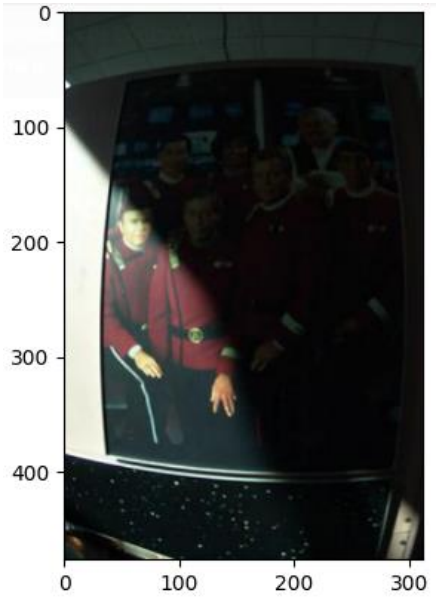
Exibir a imagem

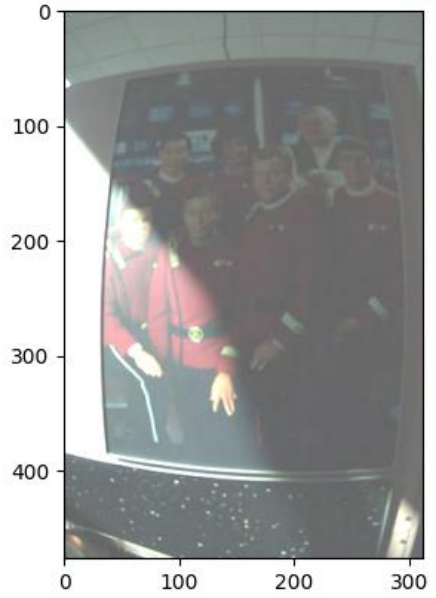


```
img = cv2.imread("goku.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# cria uma matriz com o mesmo dimensional da imagem original, mas com valores 100
matriz = np.ones(img.shape, dtype="uint8") * 150



img2 = cv2.subtract(img, matriz)


plt.imshow(img);
plt.show()
plt.imshow(img2);
```


0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

```
img = cv2.imread("jornada.png")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# cria uma matriz com o mesmo dimensional da imagem original, mas com valores 100
matriz = np.ones(img.shape, dtype="uint8") * 100



img2 = cv2.add(img, matriz)


plt.imshow(img);
plt.show()
plt.imshow(img2);
```


0
100
200
300
400
0
100
200
300


0
100
200
300
400
0
100
200
300

Sobreposição de imagens

Para realizar a sobreposição de imagens, ou blending, a OpenCV possui a função cv2.addWeighted(). Neste caso precisamos ponderar a porcentagem de cada imagem na imagem final.

```
src1 = cv2.imread('lena.jpg')
src1_rgb = cv2.cvtColor(src1, cv2.COLOR_BGR2RGB)

src2 = cv2.imread('rocket.jpg')
src2_rgb = cv2.cvtColor(src2, cv2.COLOR_BGR2RGB)

# 50% de transparencia para cada imagem
dst = cv2.addWeighted(src1_rgb, 0.5, src2_rgb, 0.5, 0)

plt.imshow(dst);
plt.show()
```

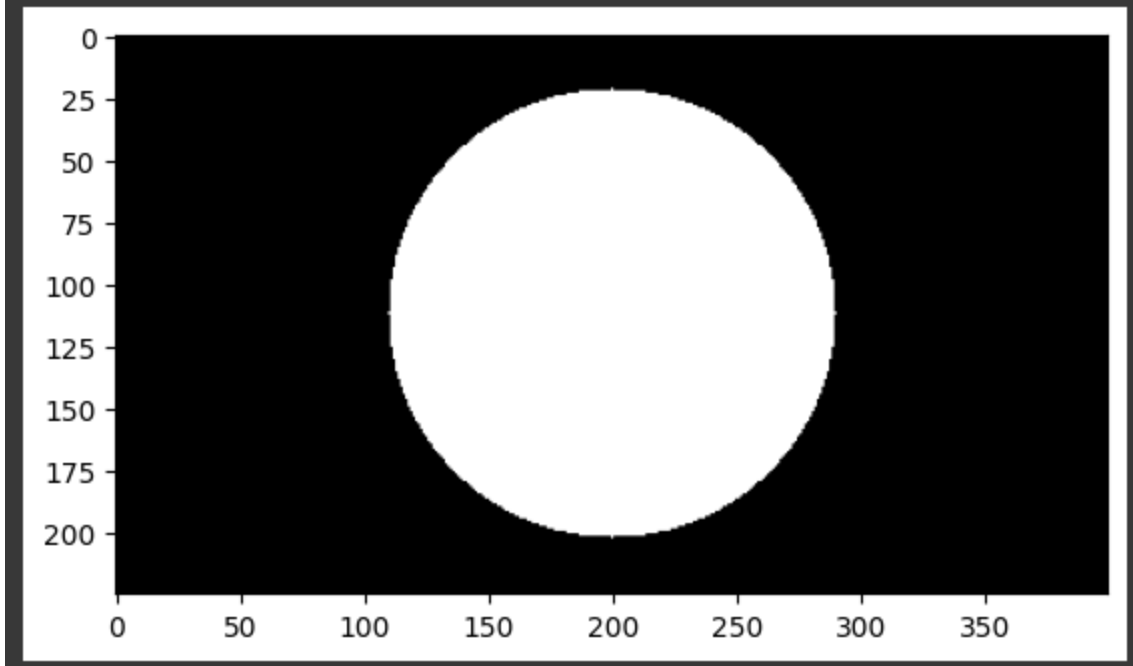
Operações Lógicas

Podemos executar nas imagens operações lógicas, como as mais usuais: NOT, AND, OR e XOR

```
img = cv2.imread('lena.jpg',0)


plt.imshow(img, cmap="gray")
plt.show()
```

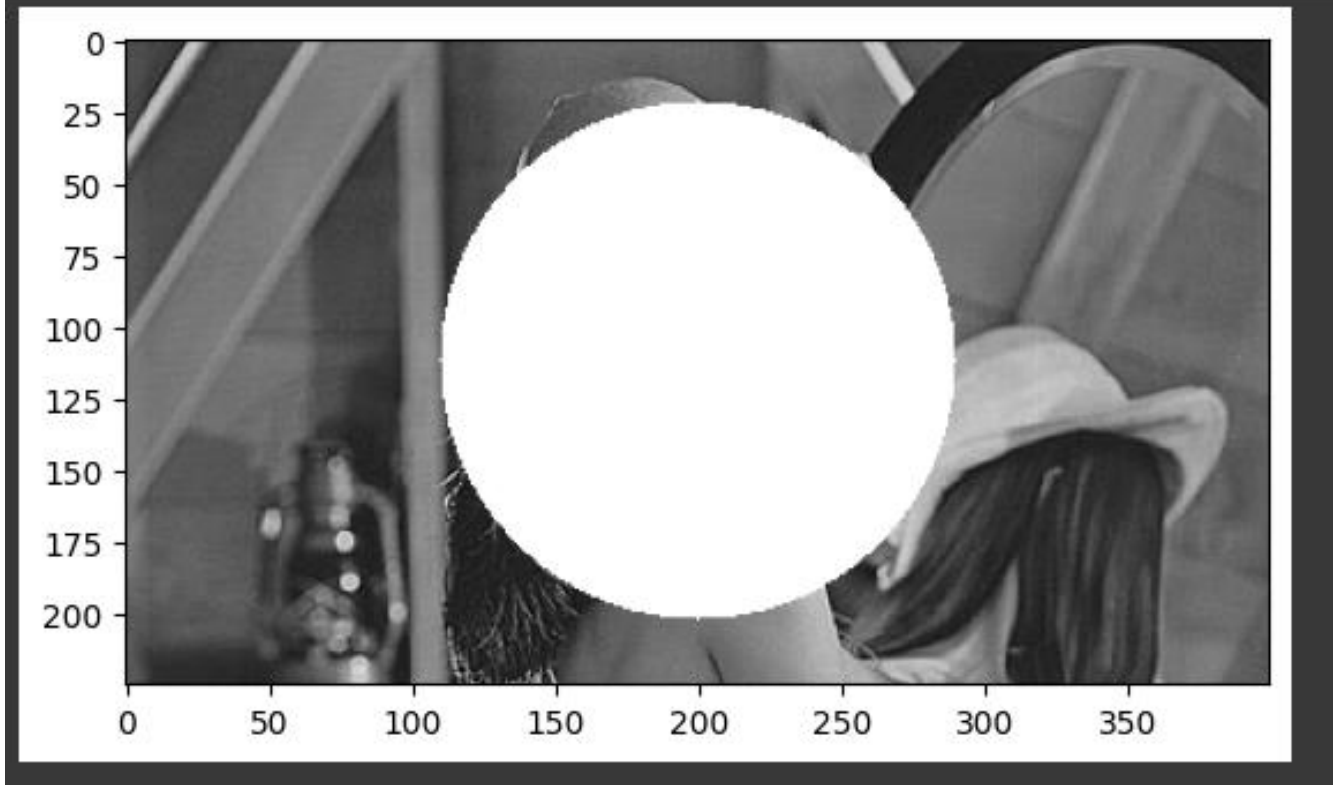
```python
# Aplicando NOT
m_not = cv2.bitwise_not(img)

plt.imshow(m_not, cmap="gray")
plt.show()
```

```python
# Criando uma imagen inicial preta
mask = np.zeros((img.shape[0], img.shape[1]), dtype="uint8")

# Coordenada central
center = (int(img.shape[1]/2), int(img.shape[0]/2))

# Tamanho do raio
radius = int((img.shape[0]/2)*0.8)

# Desenhando circulo braco
cv2.circle(mask, center, radius, 255, -1)

plt.imshow(mask, cmap="gray")
plt.show()
```



```python
# Aplicando AND
m_and = cv2.bitwise_and(img,mask)
plt.imshow(m_and, cmap="gray")
plt.show()
```

```
# Aplicando OR
m_or = cv2.bitwise_or(img,mask)
plt.imshow(m_or, cmap="gray")
plt.show()
```



```
# Aplicando XOR
m_xor = cv2.bitwise_xor(img,mask)
plt.imshow(m_xor, cmap="gray")
plt.show()
```