
FIAP

Objetivos (CV)

Ao final da disciplina o aluno será capaz de:

- Compreender o conceitos de Processamento digital de imagens
- Conhecer as principais técnicas para detecção e segmentação de objetos
- Conhecer e aplicar técnicas de visão computacional

1.2 Introdução a processamento de imagens

Objetivos da aula:

- Conhecer o que é uma imagem digital
- Conhecer como fazer leitura e exibição de imagens
- conhecer algumas propriedades de imagens
- conhecer canais de cores de imagens

array de zero

- “VETOR de qualquer formato” contendo zeros

```
# Começamos importanto as bibliotecas
import cv2
import numpy as np
from matplotlib import pyplot as plt

zeros_array = np.zeros((3, 3))


plt.imshow(zeros_array)
print(zeros_array)
```

Desafio 5

- Crie uma array de zero com x linhas e x colunas. E escreva (desenhe) a primeira letra do seu nome ou grupo.
- Plote a imagem para visualizar o resultado.
- Dica: Use np.zeros() para criar o array, para facilitar faça em escala de cinza onde o valor de intensidade do pixel 0=branco e 255=preto.

array

```
img = np.zeros((10,10, 3), dtype=np.uint8
)
plt.imshow(img)
print(img)
```

```
img[0, 0] = [255, 0, 0]
plt.imshow(img)
print(img)
```

Objetivos da aula:

- Filtro negativo de imagem
- Recorte da imagem
- Segmentação por pixel

Exibindo uma imagem de uma única banda

Podemos abrir uma imagem de uma única banda ou podemos separar uma imagem multibanda (como as imagens coloridas)

Bibliotecas

```
import numpy as np
import cv2 as cv
import matplotlib as mpl
from matplotlib import pyplot
as plot
import math
```

```
#lê a imagem
img_BGR = cv.imread("Folha.jpg")
#altera a ordem das bandas de cores BGR
para RGB
img_RGB = cv.cvtColor(img_BGR,cv.COLOR_B
GR2RGB)
#exibe a imagem
plot.imshow(img_RGB)
```

SPLIT

```
#faz split das bandas da imagem RGB
R, G, B = cv.split(img_RGB)
plot.imshow(R, cmap='gray')
plot.show()
plot.imshow(G, cmap='gray')
plot.show()
plot.imshow(B, cmap='gray')
```

RGB para Grayscale (níveis de cinzas)

```
#obtem as dimensões da imagem (número de linhas e número
de colunas)
num_lin = img_RGB.shape[0]
num_col = img_RGB.shape[1]
#cria uma matriz preenchida com zeros -
 nela serão armazenados os níveis de cinzas das médias co
mputadas (R+G+B)/3
img_gray=np.zeros((num_lin,num_col), dtype="uint8")
for l in range (num_lin):
      for c in range (num_col):
            (r, g, b) = img_RGB[l,c]
            img_gray[l,c] = (int(r)+int(g)+int(b))//3
plot.imshow(img_gray,cmap='gray')
```

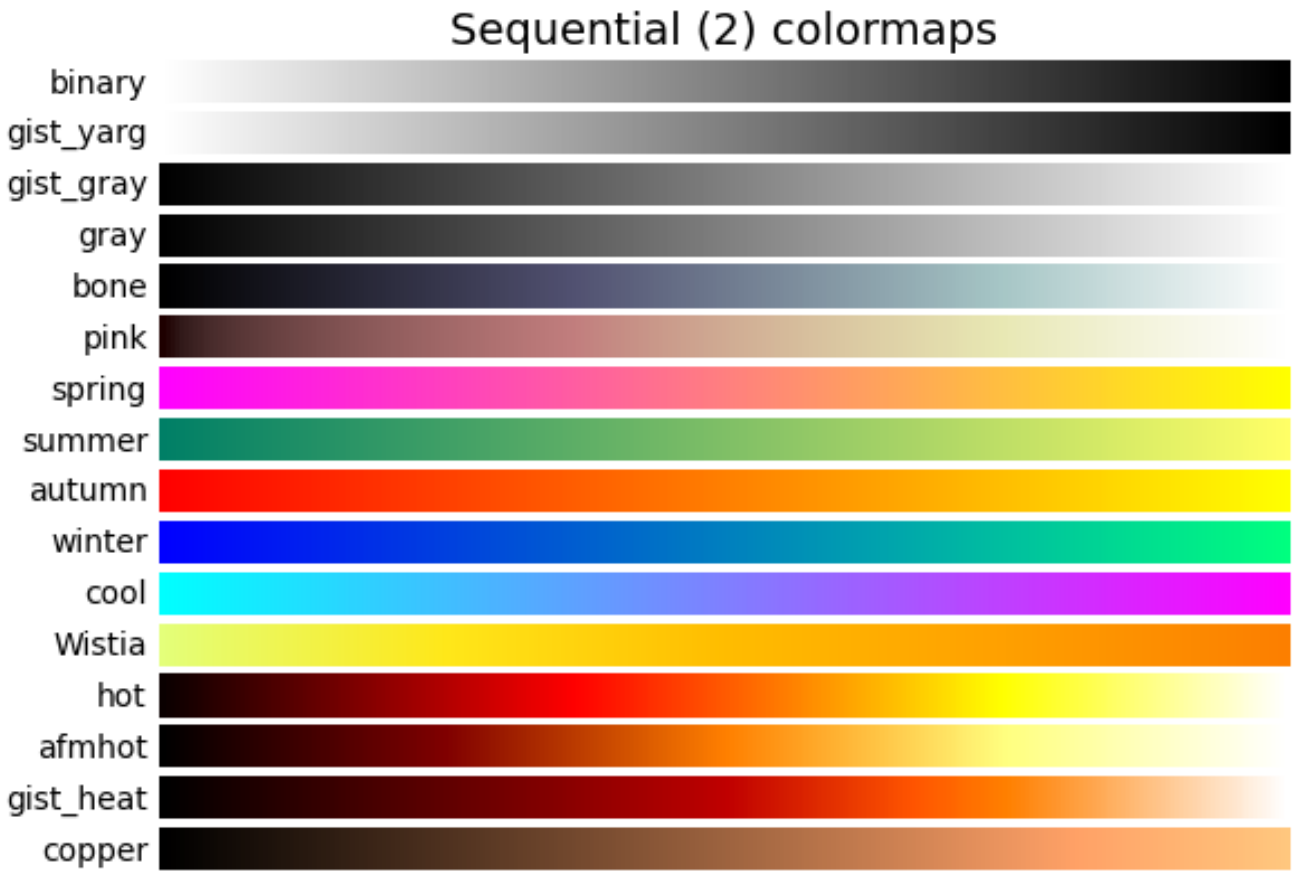
```
#a função abaixo converte diretamente uma imagem c
olorida para níveis de cinzas (pode ser BGR-
>GRAY ou RGB->GRAY)
img_gray=cv.cvtColor(img_RGB, cv.COLOR_RGB2GRAY)
plot.imshow(img_gray,cmap='gray')
```

Colorbar

```
imgB, imgG, imgR = cv.split(imgBGR)
plot.imshow(imgG, cmap='Greens')
plot.colorbar()
```

MODELOS DE MAPEAMENTOS

https://matplotlib.org/examples/color/colormaps_reference.html



DESAFIO 6

Utilize a imagem satelite.jpg e aplique uma paleta de cor que
destaque as diferentes temperaturas. Note que as intensidades
claras da imagem são baixas temperaturas e escuras são altas

Grayscale para Binária (2 níveis de cinzas 0 e 255)

```
#define o limiar
thresh = 23
#aplica a função de limiarização  usando o limiar (thresh) defin
ido
#forma 1
img_bin=cv.threshold(img_gray, thresh, 255, cv.THRESH_BINARY)[1]
#forma 2
#[thresh,img_bin] = cv.threshold(img_gray, thresh, 255, cv.THRES
H_BINARY)
#aplica a função de limiarização  usando o método de Otsu, que d
efine automaticamente o limiar
#[thresh, img_bin] = cv.threshold(img_gray, thresh, 255, cv.THRE
SH_OTSU)
plot.imshow(img_bin,cmap='gray')
print(img_gray.min())
```

Desafio 7

```
#lê a imagem
img_cor = cv.imread("Flor.jpg")
#faz split das bandas
B, G, R = cv.split(img_cor)
img_cinza=cv.cvtColor(img_cor, cv.COLOR_RGB2GRAY)
thresh = 20
img_bin = cv.threshold(R, thresh, 255, cv.THRESH_BINARY)[1]

plot.imshow(img_cor)
plot.show()
plot.imshow(img_bin, cmap='gray')
```

Transformação Negativo

```
img = cv.imread('Flor.jpg')
e=cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#converte o intervalo de níveis de cinzas de img
 para [0,1]
s = e.max()-e
plot.imshow(e, cmap='gray')
plot.show()
plot.imshow(s, cmap='gray')
```

Filtro negativo (Inverte imagem)

```
import numpy as np

from matplotlib import pyplot as plt
import cv2
imagem = cv2.imread("cogumelo.png")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)

plt.imshow(image, interpolation="none", cmap="gray")
plt.show()
```

Desafio 8

Faça uma implementação que inverte as cores de uma imagem em escala de cinza, com valores que vão de 0 ate 255. dica: a forma explicita de fazer uma inversão é: a = 255 - a

```
import cv2
imagem = cv2.imread("cogumelo.png")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)

for y in range(0, image.shape[0]):
  for x in range(0, image.shape[1]):
    if image[y, x] == 255:
      image[y,x] = 0
    else:
      image[y,x] = 255

plt.imshow(image, interpolation="none", cmap="gray")
plt.show()
```

Desafio 9

Faça o mesmo par uma imagem colorida, realize a inversão de cores dos canais R, G e B.
o resultado deve ser parecido com a imagem abaixo:

```
%matplotlib inline
# Importando a biblioteca OpenCV
import cv2

#import a biblioteca Numpy
import numpy as np

from matplotlib import pyplot as plt

imagem = cv2.imread("drone.jpg")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2
RGB)

for y in range(0, image.shape[0]):
  for x in range(0, image.shape[1]):
    image[y,x] = 255 - image[y,x]



plt.imshow(image, interpolation="none")
plt.show()
```

Recorte da imagem (crop)

O recorte de uma parte da imagem, ou crop, consiste em extrair da imagem uma região de interresse (ROI).

```
import cv2
imagem = cv2.imread("drone.jpg")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)

plt.imshow(image, interpolation="none")

plt.show()
```

```
image2 = image.copy()

#crop_img = img[y:y+h, x:x+w]
image2 = image[50:250,580:950]

plt.imshow(image2, interpolation="none
", cmap="gray")
plt.show()
```

Desafio 10

Ajude o nosso sayajin!!

A imagem foi dividida em 4 quadrantes aleatorios e precisamos organizar essa bagunça. Faça a reconstrução da imagem nas posições corretas.
Dica: Crie uma copia da imagem original (img2 = img.copy()), faça um crop da imagem 4 partes (crop1, crop2, crop3, crop4), junte as partes cortadas na ordem correta na img2. no final Salve a imagem (cv2.imwrite())

```
import cv2
imagem = cv2.imread("gokuinvertido.jpg")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)

altura = image.shape[0]
largura = image.shape[1]

print("altura: {} largura: {}".format(altura, largura))

plt.imshow(image, interpolation="none")

plt.show()

# para salvar imagem
#cv2.imwrite("gokunormal.jpg", cv2.cvtColor(image2, cv2
.COLOR_RGB2BGR))
```

Solução??

```
from matplotlib import pyplot as plt
import cv2


imagem = cv2.imread("gokuinvertido.jpg")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)

altura = image.shape[0]
largura = image.shape[1]
alt2 = int(image.shape[0]/2)
lar2 = int(image.shape[1]/2)

### crop da imagem
#crop1 = sup esq
#crop2 = sup dir
#crop3 = inf esq
#crop4 = inf dir

crop1 = image[0:alt2,0:lar2]
crop2 = image[0:alt2,lar2:largura]
crop3 = image[alt2:altura,0:lar2]
crop4 = image[alt2:altura,lar2:largura]

# faz uma cópia
img2 = image.copy()

##### remontar o quebra cabeça

img2[0:alt2,0:lar2] = crop4
img2[0:alt2,lar2:largura] = crop3
img2[alt2:altura,0:lar2] = crop2
img2[alt2:altura,lar2:largura] =crop1



#print(int(image.shape[0]/2), type(int(image.shape[0]/2)))

plt.imshow(img2, interpolation="none")
plt.show()
```

Segmentação de imagens

- Agora que sabemos como manipular pixel e como alterar seu valor e sua posição. Podemos fazer atividades mais complexas como conseguir realizar a segmentação de algum objeto ou item da imagem (vídeo), Como na imagem abaixo.

Desafio 11

- De forma intuitiva realize algumas mudanças no código e veja o efeito que causa na imagem. Este exercício é apenas um aperitivo de algumas técnicas que vamos estudar na próxima aula

```
# Entenda o codigo e faça as alterações que achar
 necessárias
import cv2
imagem = cv2.imread("drone.jpg")
image = cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB)
print(image.shape)

for y in range(0, image.shape[0]):
  for x in range(0, image.shape[1]):
    if image[y,x,1] > 170 :
      image[y,x]= (255,255,255)


plt.imshow(image, interpolation="none")
plt.show()
```