ESTI019 - QS2020 - CSM - Minami

Lab3 - Codificação de Imagem por DCT e Animação

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

```
from google.colab import drive
!ls -l "drive/My Drive/Colab Notebooks"   # verifica se montou o drive e se os arquivos es
bgr1 = cv.imread('drive/My Drive/Colab Notebooks/messi5.jpg') # leitura no formato BGR!
altura, largura, camadas = bgr1.shape
print("Resolução: ", largura, " x ", altura, "PIXELS. ", camadas, " camadas.")
```

Separa os canais e re-arranja para formar imagem RGB

```
b1, g1, r1 = cv.split(bgr1)
rgb2 = cv.merge([r1,g1,b1])
# Q1 - O que foi feito aqui?
```

Imprime cores trocadas (BGR) e reais (RGB)

```
plt.figure(figsize=[12, 5])
plt.subplot(121); plt.imshow(bgr1); plt.title('BGR')
plt.subplot(122); plt.imshow(rgb2); plt.title('RBG')
```

Converte para os formatos YCrCb e HSV

```
ycrcb = cv.cvtColor(bgr1, cv.COLOR_BGR2YCrCb)
hsv = cv.cvtColor(bgr1, cv.COLOR_BGR2HSV)
plt.figure(figsize=[15,6])
plt.subplot(131); plt.imshow(rgb2); plt.title('RGB')
plt.subplot(132); plt.imshow(ycrcb); plt.title('YCrCb')
plt.subplot(133); plt.imshow(hsv); plt.title('HSV')
```

Separação das Camadas RGB individualmente

```
imageR = rgb2.copy()
imageR[:, :, 1:3] = 0
imageG = rgb2.copy()
imageG[:, :, 0] = 0; imageG[:, :, 2] = 0
imageB = rgb2.copy()
```

```
imageB[:, :, 0:2] = 0
# Q2 - O que foi feito aqui?

plt.figure(figsize=[15,6])
plt.subplot(131); plt.imshow(imageR); plt.title('RGB_Camada R')
plt.subplot(132); plt.imshow(imageG); plt.title('RGB_Camada G')
plt.subplot(133); plt.imshow(imageB); plt.title('RGB_Camada B')
```

Separação dos Canais YCbCr

```
y1, cr1, cb1 = cv.split(ycrcb)
imageCR = ycrcb.copy()
imageCR[:, :, 0] = 0
imageCR[:, :, 2] = 0
Cr = cv.cvtColor(imageCR, cv.COLOR_YCrCb2RGB)

imageCB = ycrcb.copy()
imageCB[:, :, 0] = 0
imageCB[:, :, 1] = 0
Cb = cv.cvtColor(imageCB, cv.COLOR_YCrCb2RGB)

plt.figure(figsize=[15, 5])
plt.subplot(141); plt.imshow(rgb2); plt.title('RGB original')
plt.subplot(142); plt.imshow(y1, cmap='gray'); plt.title('YCrCb_Y')
plt.subplot(143); plt.imshow(Cr); plt.title('YCrCb_Cr')
plt.subplot(144); plt.imshow(Cb); plt.title('YCrCb_Cb')
```

Com as Imagens do Grupo:

1. Faça o mesmo com uma imagem de cada integrante do grupo e
2. Com a foto montagem de todos os do grupo, lembrando das roupas com cores diferentes, preferencialmente (R, G e B).

COMPRESSÃO DE IMAGENS COM PERDAS

```
==========================================================================
===
```

- O formato JPEG permite compressão da imagem ao salvá-la num arquivo com o comando imwrite().
- A compressão afeta a qualidade da imagem, sendo controlada pelo parâmetro IMWRITE_JPEG_QUALITY entre 0-100, sendo que quanto maior, melhor a qualidade. O

default é 95.

```
bgr = cv.imread('drive/My Drive/Colab Notebooks/lena.bmp')  # formato BGR

# salva com menor qualidade, fatores 25 e 5
cv.imwrite('lena25.jpg', bgr, [cv.IMWRITE_JPEG_QUALITY, 25])
cv.imwrite('lena05.jpg', bgr, [cv.IMWRITE_JPEG_QUALITY, 5])

# leitura para visualização e conversão para acertar a cor
rgb = cv.cvtColor(bgr, cv.COLOR_BGR2RGB)
bgr25 = cv.imread('lena25.jpg'); rgb25 = cv.cvtColor(bgr25, cv.COLOR_BGR2RGB)
bgr05 = cv.imread('lena05.jpg'); rgb05 = cv.cvtColor(bgr05, cv.COLOR_BGR2RGB)

plt.figure(figsize=[15,6])
plt.subplot(131); plt.imshow(rgb); plt.title('RGB Original')
plt.subplot(132); plt.imshow(rgb25); plt.title('JPEG fator 25')
plt.subplot(133); plt.imshow(rgb05); plt.title('JPEG fator 05')
```

COM AS FOTOS DO GRUPO

1. Repita o procedimento para cada uma das fotos dos integrantes do grupo e para a foto-montagem do grupo todo
2. Leia o tamanho dos arquivos (em bytes) e faça uma tabela comparando os tamanhos originais e os comprimidos e calcule a porcentagem de compressão de cada arquivo destes tamanhos na tabela construída

---

TRANSFORMADA DISCRETA COSSENO

Nesta parte calcule a DCT em bloco de 8x8 da imagem, referente à bola

```
img = cv.imread('drive/My Drive/Colab Notebooks/messi5.jpg')
alt, larg, cam = img.shape

ycbcr = cv.cvtColor(img, cv.COLOR_BGR2YCrCb)
y, cr, cb = cv.split(ycrcb)

bola = y[280:340, 330:390]
h, w = bola.shape

cx = round(w/2)
cy = round(h/2)

# Escolhendo um pedaço da imagem "BOLA"
bloco8x8 = bola[cx-4:cx+4, cy-4:cy+4]
print("(1)"); print("Matriz 8x8: componente Y original")
```

```
print( (1) ); print( Matriz 8x8: componente Y original )
print(bloco8x8)

bloco8x8f = np.float32(bloco8x8)/255.0  # conversão para float
dct8x8f = cv.dct(bloco8x8f)   # calcula a DCT
dct8x8 = np.int64( (dct8x8f*255.0)) # coversão para inteiro

print("(2)"); print("Imagem Y 8x8 (formato ponto flutuante)")
print( np.around(bloco8x8f, decimals = 2) )

print("(3)"); print("DCT de Y (ponto flutuante)")
print( np.around(dct8x8f, decimals = 2) )

print("(4)"); print("DCT de Y (formato inteiro)")
print(dct8x8)
```

ZERANDO manualmente da diagonal da DCT as componentes AC

```
dct8x8fz = dct8x8f.copy()
dct8x8fz[0,7] = 0
dct8x8fz[1,6:8] = 0
dct8x8fz[2,5:8] = 0
dct8x8fz[3,4:8] = 0
dct8x8fz[4,3:8] = 0
dct8x8fz[5,2:8] = 0
dct8x8fz[6,1:8] = 0
dct8x8fz[7,0:8] = 0
print( np.around(dct8x8fz, decimals = 2))
```

Bloco Original e Reconstruído com Zeros das componentes AC da diagonal para baixo zerados

```
bloco8x8recz = cv.idct(dct8x8fz)

plt.subplot(121); plt.imshow(bloco8x8,'gray'); plt.title('Bloco Original')
plt.subplot(122); plt.imshow(bloco8x8recz,'gray'); plt.title('Bloco Reconstruído com Zeros
```

Escolha outro bloco de 8x8 da imagem e:

1. refaça este procedimento zerando mais DUAS DIAGONAIS ACIMA DA PRINCIPAL além destas
2. Compare e comente as imagens do bloco original e reconstruída