ESTI019 - 3QS2021 - CSM - Minami

Lab3 - Codificação de Imagem por DCT e Animação

Objetivos:

1. Produzir um vídeo de animação para o Grupo com Blender
2. Efetuar conversões entre espaços de cores
3. Comparar arquivos comprimidos JPEG
4. Efetuar compressão de imagem com DCT

1. *Produzir um vídeo de animação para o Grupo com Blender*

Vejam o vídeo de como efetuar uma animação básica usando o Blender (instalação recomendada 2.8):

https://youtu.be/xz4t1j2-1gs

2. *Efetuar conversões entre espaços de cores*

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


from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

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

```
total 9549
-rw------- 1 root root  480760 Sep 25 21:58 avatares.png
-rw------- 1 root root  854682 Sep 25 22:05 fotos.jpg
-rw------- 1 root root    3838 Sep 24 01:27 Lab2_Anexo1_Cap_img_colab_x.ipynb
-rw------- 1 root root  467627 Sep 24 01:26 Lab2_Anexo2_Cap_vid_colab_x.ipynb
-rw------- 1 root root 3277943 Sep 28 22:13 Lab2_Imagem_Video_v2.ipynb
-rw------- 1 root root   13770 Oct  2 19:46 Lab3_Cod_Imagem_por_DCT_e_Animacao_v2.ipynb
-rw------- 1 root root  786486 Oct  1 13:28 lena.bmp
-rw------- 1 root root   72937 Oct  1 21:47 messi5.jpg
-rw------- 1 root root  474418 Oct  9  2019 Pikachu.mp4
-rw------- 1 root root 2630370 Sep 27 02:31 video_lento.mp4
-rw------- 1 root root  712299 Sep 27 02:29 video_rapido.mp4
Resolução:  548  x  342 PIXELS.  3  camadas.
```

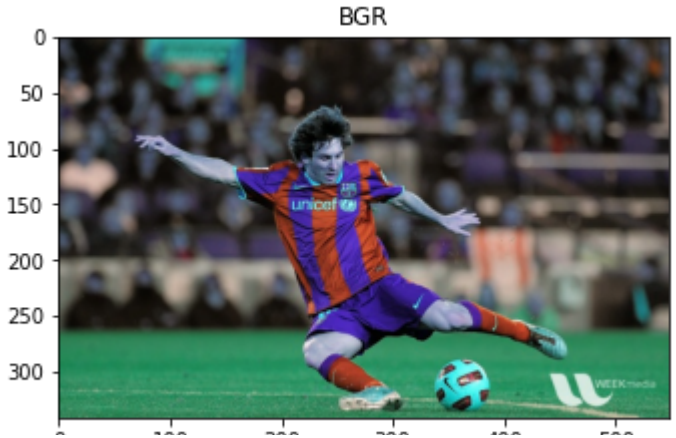
Separa os canais e re-arranja para formar imagem RGB

```
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')
```

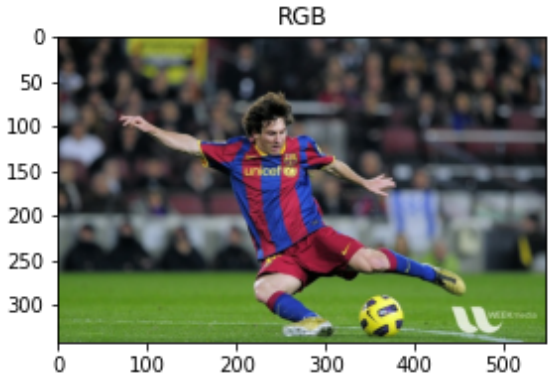
```
Text(0.5, 1.0, 'RBG')
```

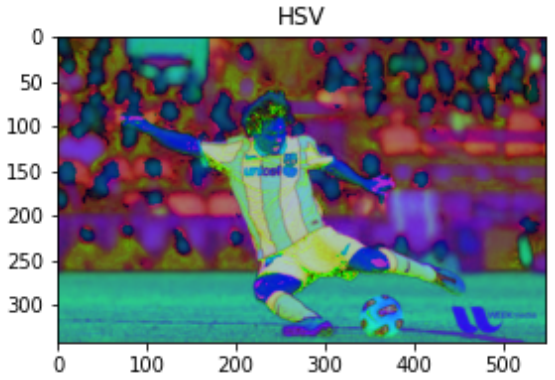


Converte para os formatos YCrCb e HSV

```
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')
```

```
Text(0.5, 1.0, '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')
```

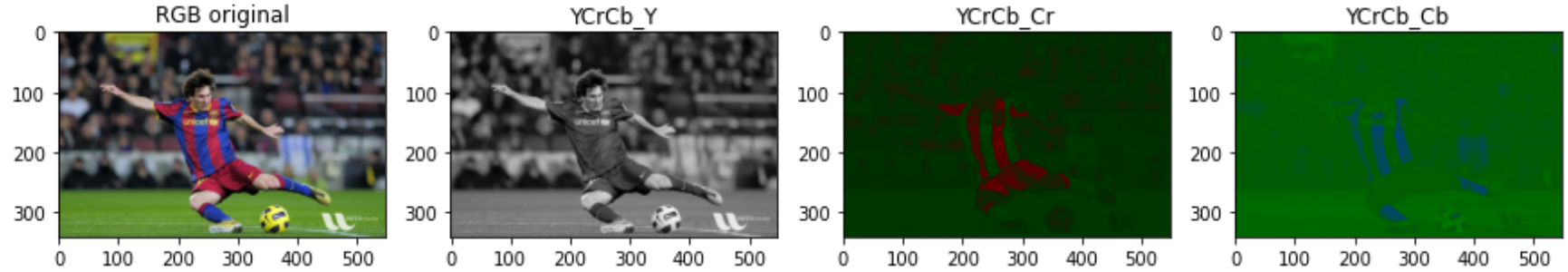
```
Text(0.5, 1.0, 'RGB_Camada B')
```

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')
```

```
Text(0.5, 1.0, '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).

```
bgr1 = cv.imread('drive/My Drive/Colab Notebooks/Multimidia/fotos.jpg') # leitura no formato BGR!
altura, largura, camadas = bgr1.shape
print("Resolução: ", largura, " x ", altura, "PIXELS. ", camadas, " camadas.")
```

```
Resolução:  1200  x  1200 PIXELS.  3  camadas.
```

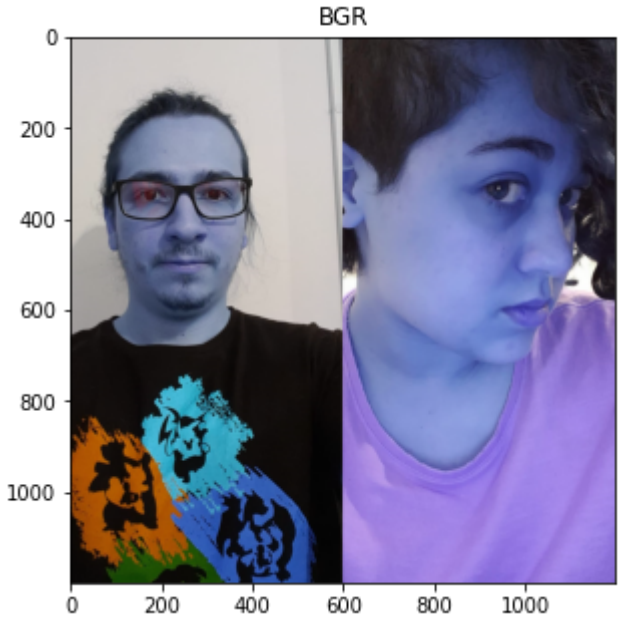
Separa os canais e re-arranja para formar imagem RGB

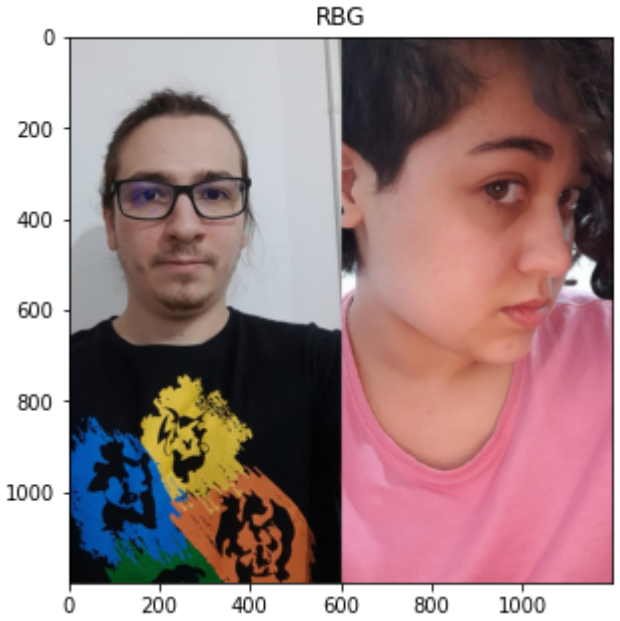
```
b1, g1, r1 = cv.split(bgr1)
rgb2 = cv.merge([r1,g1,b1])
# Q1 - O que foi feito aqui?
```

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')
```

```
Text(0.5, 1.0, 'RBG')
```



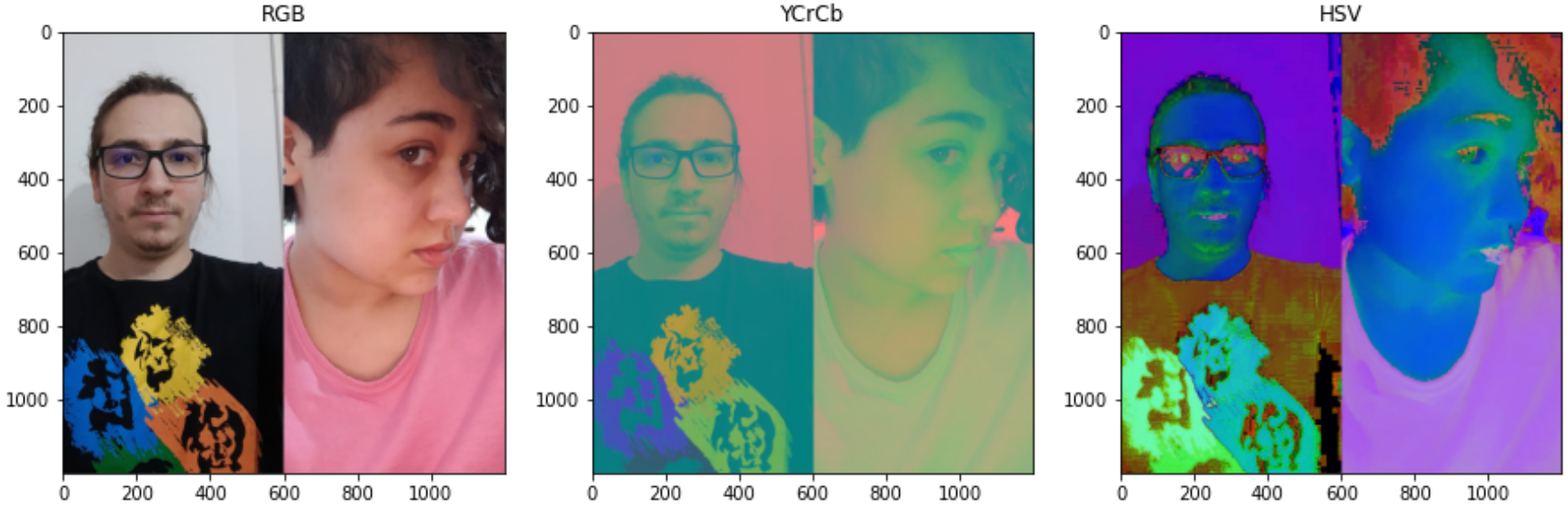


Converte para os formatos YCrCb e HSV

```
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')
```

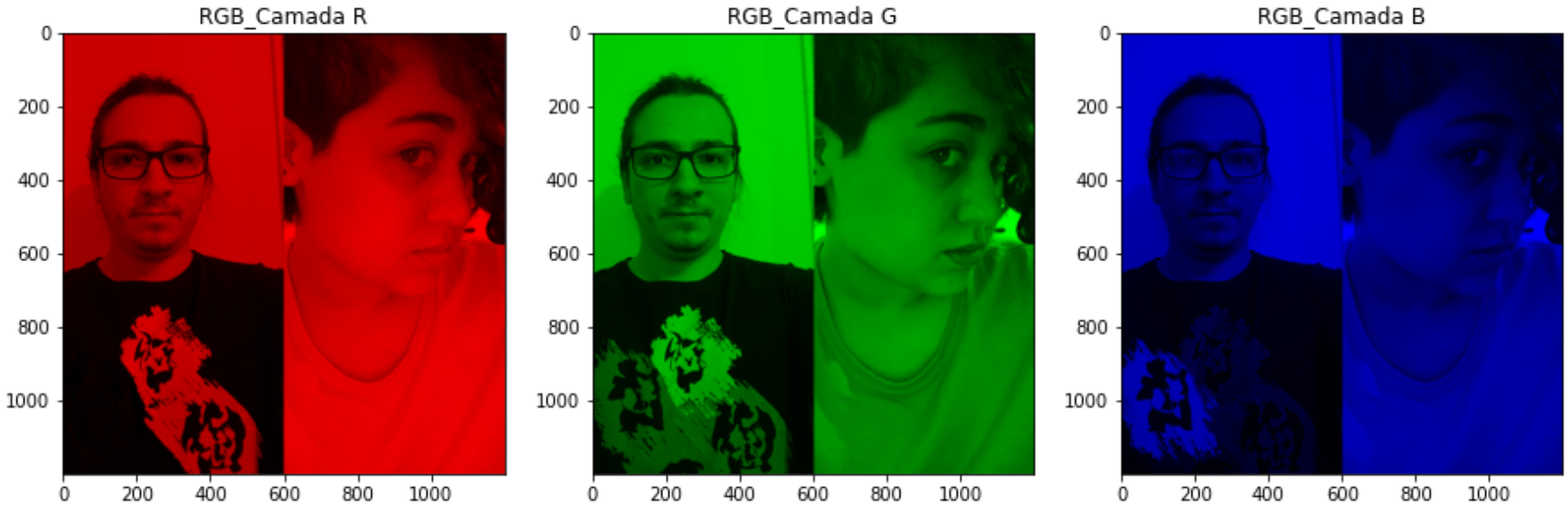
```
Text(0.5, 1.0, 'HSV')
```



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')
```

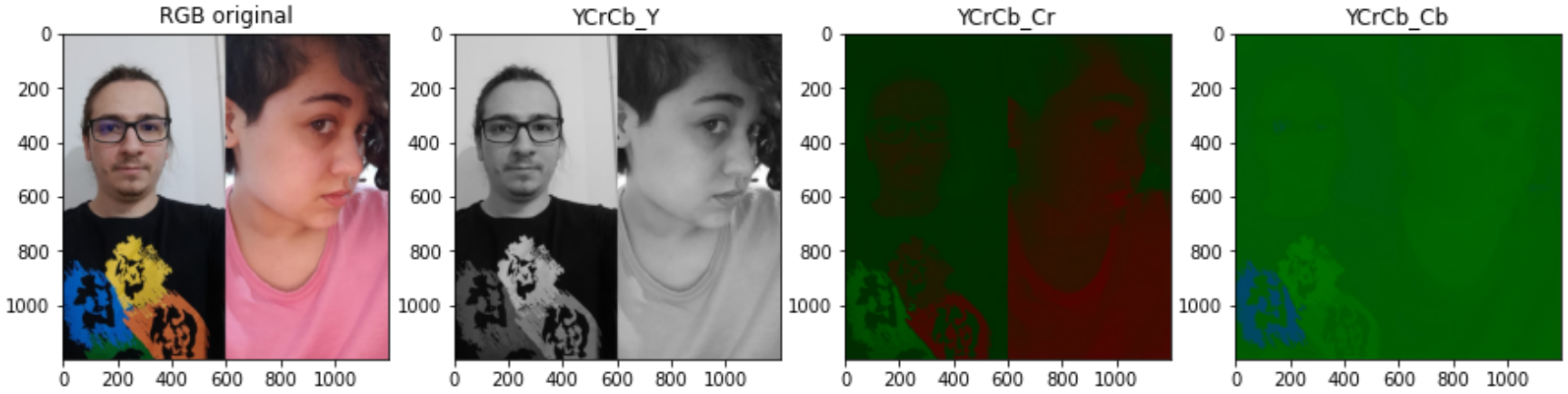
```
Text(0.5, 1.0, 'RGB_Camada B')
```



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')
```

```
Text(0.5, 1.0, 'YCrCb_Cb')
```



---

3. *Comparar arquivos comprimidos JPEG*

COMPRESSÃO DE IMAGENS COM PERDAS

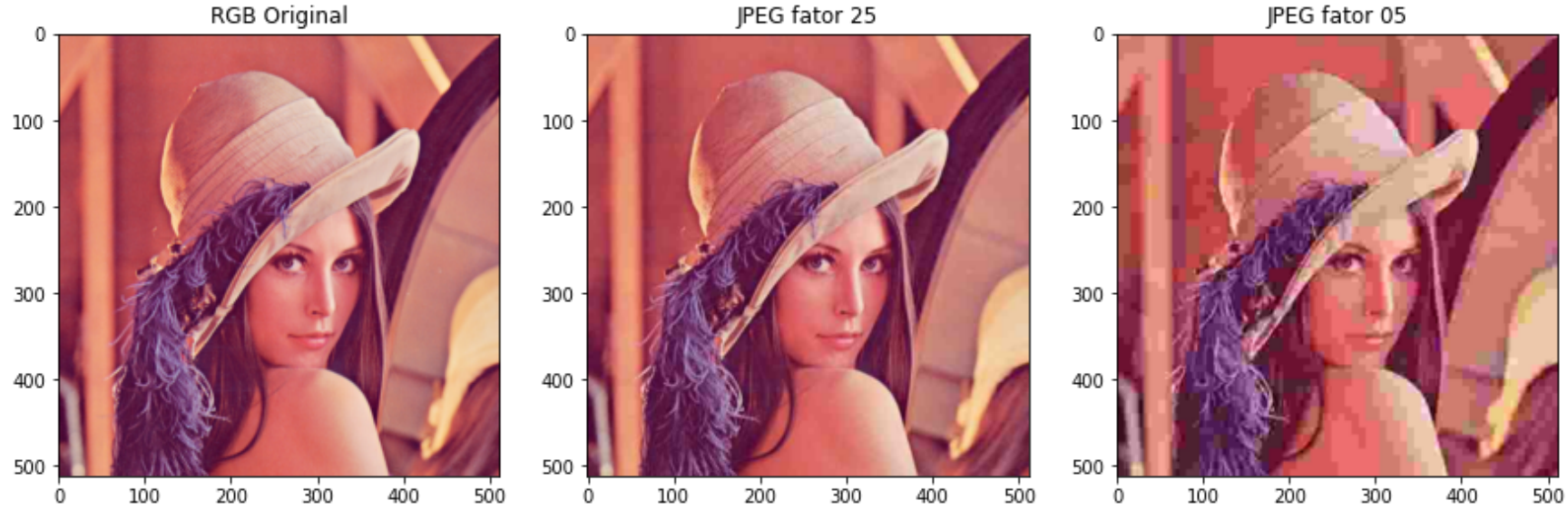
==========

- 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/Multimidia/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')
```

```
Text(0.5, 1.0, 'JPEG fator 05')
```

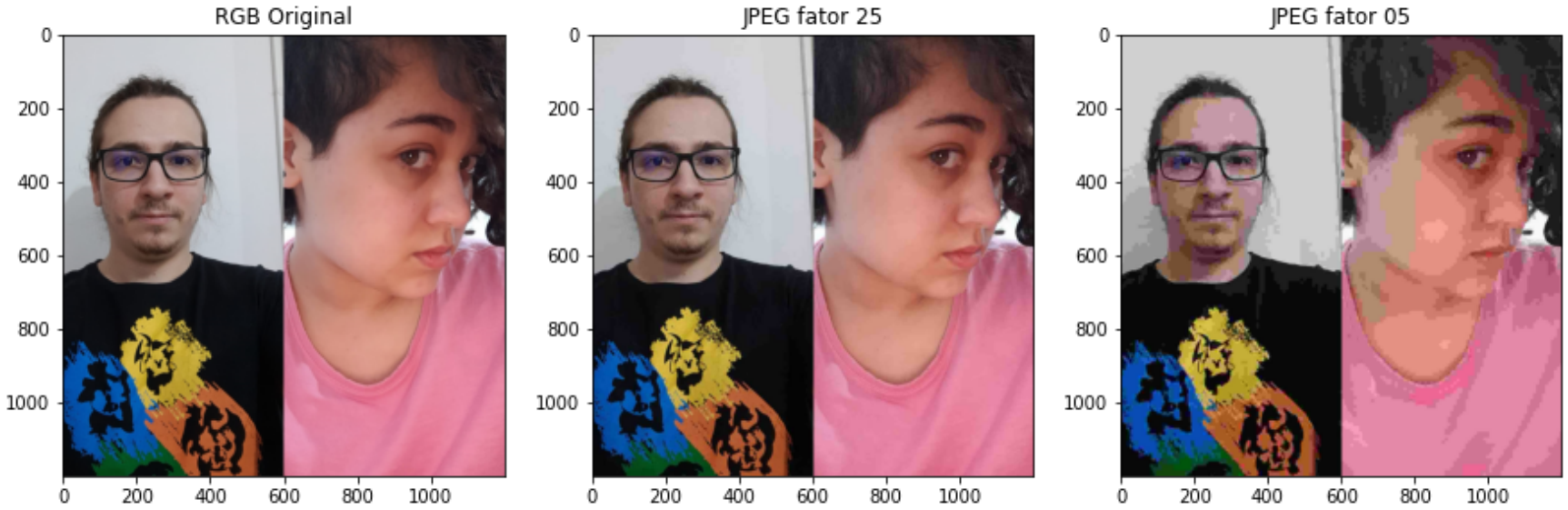


COM AS FOTOS DO GRUPO

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

```
bgr = cv.imread('drive/My Drive/Colab Notebooks/Multimidia/fotos.jpg')  # formato BGR

# salva com menor qualidade, fatores 25 e 5
cv.imwrite('fotos25.jpg', bgr, [cv.IMWRITE_JPEG_QUALITY, 25])
cv.imwrite('fotos05.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('fotos25.jpg'); rgb25 = cv.cvtColor(bgr25, cv.COLOR_BGR2RGB)
bgr05 = cv.imread('fotos05.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')
```

```
Text(0.5, 1.0, 'JPEG fator 05')
```



TRANSFORMADA DISCRETA COSSENO

4. Efetuar compressão de imagem com DCT

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

```
img = cv.imread('drive/My Drive/Colab Notebooks/Multimidia/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(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)
```

```
(1)
Matriz 8x8: componente Y original
[[ 91  84  84  89  96 109 133 157]
 [ 63  57  60  93  86  81  92 114]
 [100 101 103  97  91  77  74  79]
 [ 68  62  51  53  61  63  68  71]
 [ 35  36  36  41  39  37  35  33]
 [ 26  27  28  30  27  27  27  29]
 [ 13  14  14   9   9  10  11  13]
 [ 43  42  42  43  43  45  47  46]]
(2)
Imagem Y 8x8 (formato ponto flutuante)
[[0.36 0.33 0.33 0.35 0.38 0.43 0.52 0.62]
 [0.25 0.22 0.24 0.36 0.34 0.32 0.36 0.45]
 [0.39 0.4  0.4  0.38 0.36 0.3  0.29 0.31]
 [0.27 0.24 0.2  0.21 0.24 0.25 0.27 0.28]
 [0.14 0.14 0.14 0.16 0.15 0.15 0.14 0.13]
 [0.1  0.11 0.11 0.12 0.11 0.11 0.11 0.11]
 [0.05 0.05 0.05 0.04 0.04 0.04 0.04 0.05]
 [0.17 0.16 0.16 0.17 0.17 0.18 0.18 0.18]]
(3)
DCT de Y (ponto flutuante)
[[ 1.8  -0.12  0.06 -0.03  0.05  0.   -0.01 -0.01]
 [ 0.87 -0.16  0.06 -0.03  0.06  0.   -0.01 -0.01]
 [ 0.21 -0.16  0.06 -0.03  0.02  0.01  0.01 -0.  ]
 [-0.21 -0.12  0.04 -0.01 -0.    0.01  0.   -0.01]
 [ 0.15 -0.08  0.06  0.03 -0.02 -0.   -0.    0.01]
 [-0.    0.03  0.07  0.01 -0.03 -0.02  0.01  0.02]
 [ 0.24  0.08  0.   -0.01 -0.03 -0.01  0.01  0.01]
 [ 0.    0.1  -0.03 -0.02 -0.02 -0.    0.01  0.01]]
(4)
DCT de Y (formato inteiro)
[[458 -30  14  -7  13   0  -2  -2]
 [222 -39  15  -8  15   0  -1  -2]
 [ 52 -40  14  -7   4   2   1   0]
 [-53 -30   9  -2   0   1   1  -1]
 [ 37 -20  16   8  -5   0   0   1]
```

```
[  0   6  18   3  -8  -4   2   5]
[ 60  21   0  -2  -6  -1   2   3]
[  0  24  -7  -4  -5  -1   2   3]]
```

ZERANDO manualmente da diagonal da DCT as componentes AC

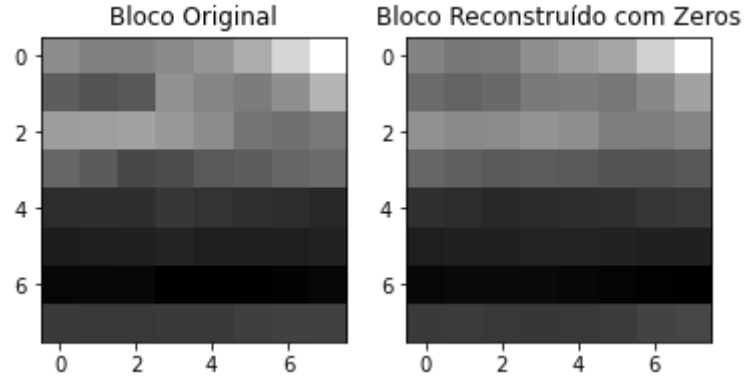
```
dct8x8fz = dct8x8f.copy()
dct8x8fz[0,7] = 0
dct8x8fz[1,6:8] = 0
dct8x8fz[2,5:8] = 0
dct8x8fz[3,4:8] = 0
dct8x8fz[4,3:8] = 0
dct8x8fz[5,2:8] = 0
dct8x8fz[6,1:8] = 0
dct8x8fz[7,0:8] = 0
print( np.around(dct8x8fz, decimals = 2))
```

```
[[ 1.8  -0.12  0.06 -0.03  0.05  0.   -0.01  0.  ]
 [ 0.87 -0.16  0.06 -0.03  0.06  0.    0.    0.  ]
 [ 0.21 -0.16  0.06 -0.03  0.02  0.    0.    0.  ]
 [-0.21 -0.12  0.04 -0.01  0.    0.    0.    0.  ]
 [ 0.15 -0.08  0.06  0.    0.    0.    0.    0.  ]
 [-0.    0.03  0.    0.    0.    0.    0.    0.  ]
 [ 0.24  0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.  ]]
```

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')
```

```
Text(0.5, 1.0, '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

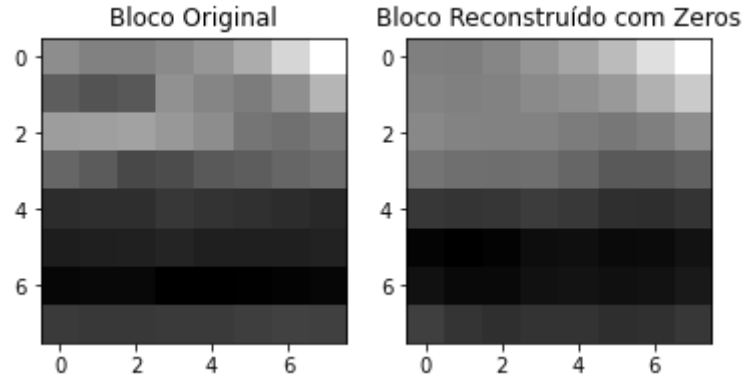
```
dct8x8fz = dct8x8f.copy()
dct8x8fz[0,5:8] = 0
dct8x8fz[1,4:8] = 0
dct8x8fz[2,3:8] = 0
dct8x8fz[3,2:8] = 0
dct8x8fz[4,1:8] = 0
dct8x8fz[5,0:8] = 0
dct8x8fz[6,0:8] = 0
dct8x8fz[7,0:8] = 0
print( np.around(dct8x8fz, decimals = 2))
```

```
[[ 1.8  -0.12  0.06 -0.03  0.05  0.    0.    0.  ]
 [ 0.87 -0.16  0.06 -0.03  0.    0.    0.    0.  ]
 [ 0.21 -0.16  0.06  0.    0.    0.    0.    0.  ]
 [-0.21 -0.12  0.    0.    0.    0.    0.    0.  ]
 [ 0.15  0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.  ]
```

```
       [ 0.    0.    0.    0.    0.    0.    0.    0.  ]
       [ 0.    0.    0.    0.    0.    0.    0.    0.  ]]
```

```
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')
```

```
Text(0.5, 1.0, 'Bloco Reconstruído com Zeros')
```



-X-X-X-