**Instituto Superior de Engenharia de Lisboa**
Licenciatura em Engenharia Informática e Multimédia
**Codificação de Sinais Multimédia**

**2º Semestre de 2018/2019**

O ficheiro com o relatório do trabalho (em formato PDF) e os ficheiros de código implementado devem ser encapsulados num ficheiro ZIP com o número do grupo e submetidos no Moodle até 24 de Março.

---

A biblioteca OpenCV (Open Source Computer Vision Library) tem um conjunto de funções que permite processar imagens em Python. Esta ainda permite trabalhar com vários formatos de ficheiros de imagem.

1. Abra o ficheiro com a imagem "lenac.tif" e apresente a imagem. Verifique para que servem os métodos "dtype" e "shape":

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

x_img = cv2.imread("lenac.tif")
cv2.imshow('Original Image', x_img)

print x_img.dtype
print x_img.shape

cv2.waitKey(0)
cv2.destroyAllWindows()
```

2. Grave a mesma imagem, mas agora em formato "JPEG" com diferentes qualidades. Verifique visualmente a qualidade das imagens assim como o tamanho do ficheiro. Calcule a taxa de compressão, a SNR e a PSNR.

```
cv2.imwrite('file1.jpg',x_img,(cv2.IMWRITE_JPEG_QUALITY,80))
cv2.imwrite('file2.jpg',x_img,(cv2.IMWRITE_JPEG_QUALITY,10))
```

3. Converta a imagem para níveis de cinzento, usando o método "cvtColor" e grave a imagem. Este método aplica a transformação $Y = R*299/1000+G*587/1000+B*114/1000$, justifique a utilização desta equação. Verifique também o tamanho do ficheiro e compare-o com o ficheiro original.

```
x_img_g = cv2.cvtColor(x_img,cv2.COLOR_BGR2GRAY)
cv2.imshow('Gray Image',x_img_g)
cv2.imwrite('file3.bmp',x_img_g)
```

4. Apresente o histograma da imagem em tons de cinzento, verifique quantos níveis de cinzento tem a imagem.

```
plt.hist(x_img_g.ravel(),256,[0,256])
```

5. Nos próximos trabalhos será necessário realizar operações com os valores de cada pixel. Para este efeito pode-se transformar a imagem para um array. O código seguinte representa o pixel mais significante da imagem. Apresente oito imagens, cada uma com o valor de cada bit para todos os pixeis.

```
y = x_img_g > 128
cv2.imshow('BW', y*1.0)
```

6. Grave uma imagem que contém apenas a informação dos 4 bits mais significantes da imagem.

```
y= ...
cv2.imwrite('lena_4.bmp',y)
```

7. Construa uma função que realize o algoritmo de dithering Floyd Steinberg. Esta função recebe uma matrix (com os pixeis em tons de cinzento) e devolve uma matrix com valores a preto e branco. Este algoritmo aproxima cada pixel da imagem ao valor mais próximo (preto ou branco) e o erro é difundido para os pixeis adjacentes seguindo o método:

$$\begin{bmatrix} . & . & . \\ . & x & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix}$$

```
y = dither(x_img_g)
```

8. Construa uma função para gravar a matriz para um ficheiro binário. Verifique o tamanho do ficheiro inicial e do ficheiro final. Calcule a taxa de compressão e meça o SNR e o PSNR.

9. Crie uma função que apresente uma imagem ($100 \times 100$) como se apresenta na figura. O ângulo de cada sector é dado por parâmetro passado à função (o ângulo é um valor inteiro entre 0 e 360 graus).