Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia
Codificação de Sinais Multimédia

2º Semestre de 2024/2025

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

1. Abra o ficheiro com a imagem `"lena.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("lena.tif")
   cv2.imshow("OriginalImage", 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 (por exemplo, 80 e 10).

   Verifique visualmente a qualidade das imagens assim como o tamanho dos ficheiros.

   Escreva código para calcular 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 que está na variável `x_img` para níveis de cinzento usando o método `cvtColor` e grave o resultado. Verifique também o tamanho do ficheiro e compare com o tamanho do ficheiro original.

   ```
   x_img_g = cv2.cvtColor(x_img,cv2.COLOR_BGR2GRAY)
   cv2.imshow("GrayImage",x_img_g)
   cv2.imwrite("file3.bmp",x_img_g)
   ```

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

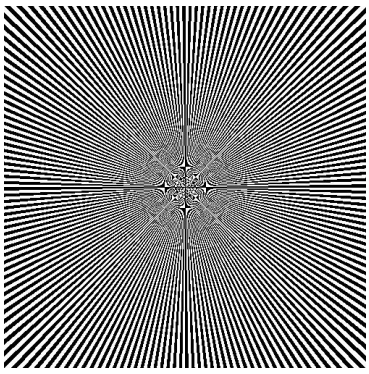
$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & x & 7/16 \\ 3/16 & 5/16 & 1/16 \end{bmatrix}$$

   Aplique esta função à imagem em tons de cinzento que está na variável `x_img_g`.

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

7. Construa uma função para gravar a matriz obtida na questão anterior (variável `y`) para um ficheiro binário. Verifique o tamanho do ficheiro inicial e do ficheiro final. Calcule a taxa de compressão e meça a SNR e a PSNR.

8. Crie uma função que apresente uma imagem com dimensão de $N \times N$ pixeis 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).



9. Neste exercício vamos realizar controlo de erros.

   a. Abra o ficheiro com a imagem `xpto3.jpeg` utilizando a instrução `cv2.imread("xpto3.jpeg",0)` para converter para níveis de cinzento[1]. Visualize a imagem e verifique as suas dimensões bem como o tipo dos seus elementos. Produza o histograma da imagem. Caso a imagem tenha valores que não sejam exclusivamente de "branco" ou "preto" quantifique a imagem a fim de que os valores passem a ser apenas de "branco" ou "preto". Para fazer a quantificação rescale a imagem para valores de intensidade entre 0 e 1. Confirme com um histograma que após a quantificação obteve uma imagem estritamente com valores 0 ou 1 em cada um dos píxeis. Visualize a imagem quantificada após rescalar novamente para valores da gama 0-255.

   Para a imagem quantificada, cujos píxeis têm os valores 0 ou 255, o display no notebook recorrendo à função `imshow()` do módulo `matplotlib.pyplot` apresenta (utilizando `cmap=gray`) vários píxeis a cinzento, o que não corresponde à realidade. Para visualizar o "preto e branco" de maneira fidedigna podemos usar o opencv ou então a biblioteca PIL. Para por exemplo visualizar uma imagem no notebook usando a biblioteca PIL pode fazer-se:

```
from PIL import Image
from IPython.display import display

img = Image.fromarray(image, mode='L') # image:  np.array que contem a imagem
display(img)
```

   b. Simule o efeito de um canal de transmissão ruidoso no qual se transmite a imagem a preto e branco obtida na alínea anterior (referimo-nos à imagem quantificada e cujos píxeis têm o valor 0 ou 1). Visto que a imagem consiste apenas de valores 0 e 1 pode ser entendida como

[1] Caso a imagem não esteja na diretoria do notebook, naturalmente deverá ter isso em conta.

uma sequência de bits. Neste exercício o efeito do canal consiste em produzir o *flip* de cada bit com determinada probabilidade f (a *probabilidade de erro de bit* no canal). Considere os seguintes valores de f: $0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5$.

i. Determine a taxa de erros de bit real para cada uma das imagens ruidosas.

ii. Faça o display de cada uma das imagens (original e ruidosas), rescalando-as previamente para valores entre 0 e 255 e convertendo para tipo `uint8`.

iii. Implemente a correção de erros através do código de repetição R_N. Considere diversos valores de N: 3, 5, 7, 9, 11, 61. Para tal considere que a sequência de bits corresponde à leitura das imagens por linha. Determine a taxa de erros de bit (BER) real após a correção de erros. Visualize as imagens corrigidas e compare com as imagens originais e ruidosas (antes da correção).

iv. Implemente a correção de erros através do código de Hamming $H(7, 4)$, utilizando a formulação matricial. Para esta experiência considere os seguintes valores de f: $0.0001, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05$. Determine a taxa de erros de bit (BER) real após a correção de erros. Visualize as imagens corrigidas e compare com as imagens originais e ruidosas (antes da correção).

v. Produza gráficos que permitam fazer uma análise abrangente dos resultados (poderá também calcular métricas como a SNR, por exemplo). Analise e discuta os compromissos entre robustez ao ruído e razão de código.