

Tópicos de Ciência de Dados

Trabalho Laboratorial Preparatório

Introdução ao numPy e matPlotLib



Introdução

Período de execução: 15 dias (2 aulas práticas laboratoriais)

Esforço extra aulas previsto: 4h

Data de Entrega: esta ficha não é entregue

Objectivo: Pretende-se que o aluno se familiarize com as bibliotecas numPy, sciPy e matplotlib.

Trabalho Prático

A . Elaboração de um conjunto de scripts e funções para manipulação de som

1. Crie um script e grave-o com o nome **'mainAudio.py'**. Este script será utilizado na chamada de todas as funções indicadas abaixo (alíneas 2 a 11).
2. Leia o ficheiro wave **'saxriff.wav'**. Para tal, utilize a função **'read'** do **scipy.io**:
 - `from scipy.io import wavfile`
 - `[fs, data] = wavfile.read(filename);` Para mais detalhes, consulte a ajuda do **scipy**.
3. Escute o sinal áudio, com recurso à função **play** da biblioteca **sounddevice**.
 - `import sounddevice as sd`
 - `sd.play(data, fs)`
 - `status = sd.wait() # Wait until file is done playing`
4. Crie a função **apresentarInfo('nomeFicheiro', fs, nrBitsQuant)** que apresente no ecrã informações sobre o ficheiro. Utilize a função **'print'**
 - Exemplo do resultado da função:

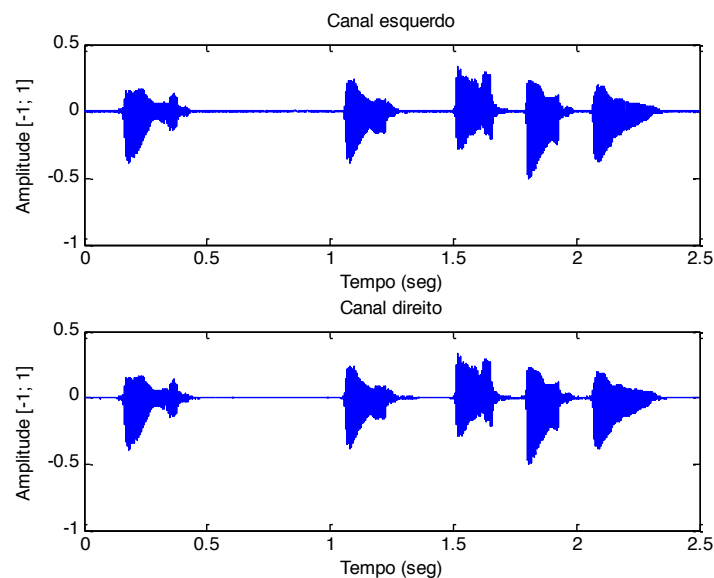
Informação sobre o ficheiro Nome do ficheiro: saxriff.wav Taxa de amostragem: 22.050 kHz Quantização: 16 bits
--

5. Elabore a função **visualizacaoGrafica(sinal, fs)** que apresente o sinal áudio num gráfico 2D.
 - 5.1. Converta o índice de cada amostra para o valor temporal respectivo; sugestão: intervalo de amostragem $T_s = 1/fs$; `duracao = nrAmostras * T_s` .

5.2. No caso de se tratar de um sinal monoaural, apresente o sinal num único plot; caso o sinal seja stereo (i.e., 2 canais), apresente o canal esquerdo no topo e o direito no plot inferior); utilize as funções *plot*, *subplot*, *figure*, *close*, *xlabel*, *ylabel*, e *title* da biblioteca *matplotlib.pyplot* (ex:

- Exemplo:
 - `import matplotlib.pyplot as plt`
 - `plt.subplot(212)`
 - `plt.plot(tempo,canalDir,'b')`
 - `plt.xlabel('Tempo (seg)')`
 - `plt.ylabel('Amplitude [-1:1]')`
 - `plt.title('Canal Direito')`

Dê nomes aos eixos e um título a cada plot realizado.



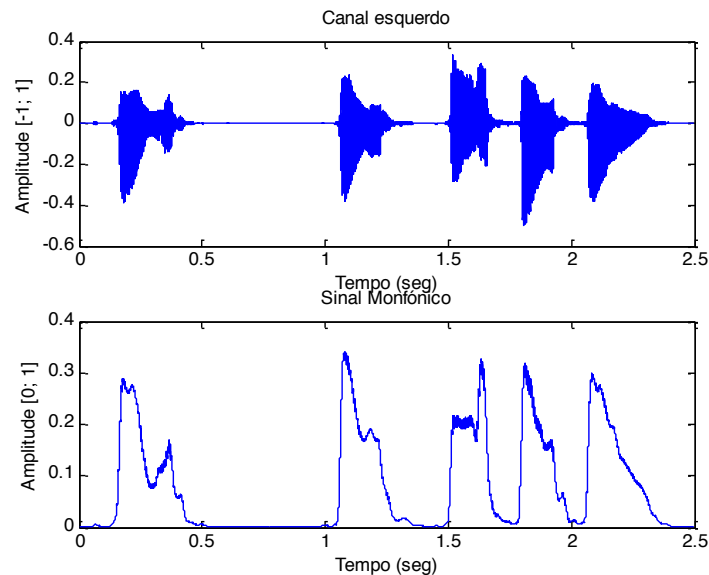
- Torne a função anterior mais genérica, através da definição do intervalo de tempo de graficação, i.e., **visualizacaoGrafica(sinal, fs, tIni, tFim)**
 - No caso de apenas serem introduzidos os dois primeiros argumentos, o funcionamento deverá ser idêntico ao do ponto 5.
 - No caso de se introduzir também o instante inicial, deverá fazer o plot desde aí até ao fim do sinal.
 - Caso sejam introduzidos os 4 parâmetros, o plot deverá ser restringido ao intervalo [tIni, tEnd]
 - Nota 1: utilize **args*.
- 6. Execute o mesmo procedimento do ponto anterior, agora com a função *axis* do *matplotlib.pyplot*.
 - Nota: a restrição da zona de visualização é conseguida de forma mais simples através da função *axis([xmin,xmax,ymin,ymax])*.

7. Crie uma função que adicione ruído uniforme ao sinal. Esta função deverá receber o sinal original e a amplitude do ruído e devolver o sinal original com ruído.
 - Recorra à função *rand* do `numpy.random`, a qual gera uma matriz de tamanho especificado contendo valores aleatórios no intervalo `[0, 1]`. Deverá, portanto, normalizar os valores resultantes de acordo com a amplitude especificada.

8. Implemente uma função para o cálculo da energia do sinal; a função em causa deverá receber o sinal (mono ou stereo) e devolver a sua energia (ou a energia de cada canal, no caso de sinais stereo), calculada segundo a fórmula: $\sum_{i=1}^N x^2[i]$, onde *x* representa o sinal em análise. Utilize a função *sum* do `numpy`. Observe que para evitar overflows nos cálculos poderá converter os seus dados para float usando, por exemplo, o método *astype* (*type*) do `numpy`
 - o Por exemplo:
 - o `import numpy as np`
 - o `v=np.sum(sinal.astype(np.float32)**2,axis=0)`

9. Elabore uma função que substitua o canal direito do sinal original por outro sinal áudio (`beats.wav`). Assuma que os dois sinais têm a mesma frequência de amostragem.
 - a. A função deverá receber o sinal original, o nome do ficheiro com o novo sinal e o instante temporal em que o mesmo deverá ser adicionado (exemplo, 0.5 seg depois do início do primeiro sinal); deverá devolver uma matriz com duas colunas contendo cada um dos sinais;
 - Sugestão 1: adicione zeros ao início do novo sinal, de forma a deslocá-lo para o instante desejado;
 - Sugestão 2: garanta que a dimensão dos dois canais é a mesma, através da adição de zeros ao final de um deles, em conformidade com os comprimentos respectivos.
 - Sugestão 3: no caso do segundo sinal ser stereo, utilize o seu canal esquerdo.
 - b. No seu código, faça soar o novo sinal. Verifique que o som de cada sinal é enviado por uma coluna diferente.
 - c. Visualize o novo sinal recorrendo à função anteriormente implementada
 - d. Misture agora os sinais esquerdo e direito. Para tal, adicione os seus valores e substitua os resultados, mantendo agora apenas uma coluna (i.e., sinal mono);
 - a) Escute o resultado e verifique que o som de ambas as colunas é igual.
 - b) Visualize o novo sinal recorrendo à função anteriormente implementada.

10. Implemente uma função que devolva o contorno de amplitude de um sinal¹. O resultado deverá ser algo do género:



- Implemente a função **ca = contornoAmplitude(x, W)**, em que W (ímpar) determina a dimensão de uma janela deslizante necessária à determinação do contorno.
- O cálculo do contorno começa por uma operação designada por “rectificação de meia-onda”, definida segundo a equação (1):

$$x_r[t_i] = \begin{cases} x[t_i] & , x[t_i] \geq 0 \\ 0 & , x[t_i] < 0 \end{cases} \quad (1)$$

- Utilize o método `nonzero()` dos Arrays do `numpy` para procurar os índices negativos de x
 - (Exemplo: **(i,j)=(sinal<=0).nonzero()**; se os quiser multiplicar por 2, basta fazer: **sinal(i,j) = sinal(i,j) * 2**;
- De seguida, o contorno é determinado pela média móvel dos valores do sinal. i.e., **ca(i) = np.mean(xr[i-W/2 : i + W/2])**; Dado que W é ímpar, utilize o método `floor` do `numpy`, a qual arredonda o valor em causa para o inteiro exactamente anterior.
 - a) Faça um ciclo para implementar a média móvel
 - b) Finalmente, normalize o contorno obtido, de forma a que a sua energia seja igual à do sinal inicial (utilize a função desenvolvida anteriormente).
 - c) Visualize os resultados para diversos valores de W, e.g., 5, 7, 21, 25, 75, 255 e 355.

¹ A determinação do contorno de amplitude tem diversas aplicações no processamento de sinal. A título de exemplo, em aplicações de análise de conteúdos musicais esse contorno é utilizado na detecção de inícios de notas musicais e na detecção de batidas rítmicas. Neste exercício, o contorno será determinado de forma simples, havendo obviamente mecanismos mais eficazes e eficientes de o obter.

B . Elaboração de um conjunto de scripts e funções para manipulação de imagem

1. Crie um script e grave-o com o nome **'mainImage.py'**. Este script será utilizado na chamada de todas as funções indicadas abaixo.
2. Leia a imagem no ficheiro 'image008.jpg'. Para tal, utilize a função 'imread' da package matplotlib.image:
 - Exemplo:
 - o `import matplotlib.image as mpimg`
 - o `img=mpimg.imread(file)`
 - Em imagens monocromáticas (i.e., níveis de cinzento), o resultado (em Y) é uma matriz WxH, em que W é o número de pixels correspondentes à largura da imagem e H é o número de pixels relativos à altura.
 - Em imagens policromáticas (i.e., coloridas), o resultado é uma matriz tri-dimensional de dimensão WxHx3, em que a última dimensão refere-se ao espaço de cores RGB (red, green, blue). A imagem é formada pela mistura de vermelho, verde e azul, com intensidades diferentes.
 - O número de bits utilizado na representação da intensidade varia com o formato do ficheiro e com a qualidade de gravação. No âmbito deste trabalho, poderá assumir valores de intensidade entre 0 e 255.
3. Visualize a sua matriz com recurso à função *image* do Matlab. Para retirar os valores dos eixos, utilize *axis('off')* da package *matplotlib.pyplot*.

image008.jpg



4. Crie uma função que realce a componente vermelha da imagem. Para isso, multiplique os valores relativos à componente R (i.e., $Y(:, :, 1)$) por um dado factor. Valores superiores a 255 deverão ser limitados a esse máximo. A função deverá receber a matriz inicial e o factor de multiplicação e devolver a matriz final. A imagem abaixo foi gerada com factor 2.
 - Nota: não utilize ciclos

Realce de Vermelho



5. Implemente uma função que dê o “efeito de mosaico” à sua imagem. Defina a largura, W , do mosaico. Em cada linha, os pixels no intervalo $[j - \text{floor}(W/2); j + \text{floor}(W/2)]$ receberão todos a intensidade do pixel j (para cada um dos canais R, G e B). A função deverá receber a imagem original e a largura do mosaico (ímpar). Visualize os resultados para $W = 11, 25$ e 35 .
- Nota 1: mosaicos nos bordos da imagem com dimensão inferior a W deverão ser tratados convenientemente;
 - Sugestão 1: na atribuição de valores a um dado intervalo, utilize o método `ones` do `numPy`. Tenha em consideração que o tipo de dados na matriz poderá não ser `standard`, pelo que neste caso a multiplicação não será possível. Deste modo, antes da multiplicação converta os valores da matriz para `float32`, i.e., utilize `astype(numPy.float32)`.
 - Sugestão 2: utilize dois ciclos `for` (*nested*);
 - A figura seguinte apresenta os resultados para $W = 11$. Embora não seja visível neste enunciado, o último mosaico é mais estreito que os restantes.

Efeito de Mosaico



6. Crie uma função para conversão da imagem para níveis de cinzento. Para esse efeito, deverá calcular a luminância monocromática pela combinação dos valores RGB de acordo com o standard NTSC:

$$Y_{cinza}[i, j] = 0.2978R[i, j] + 0.5870G[i, j] + 0.1140B[i, j]$$

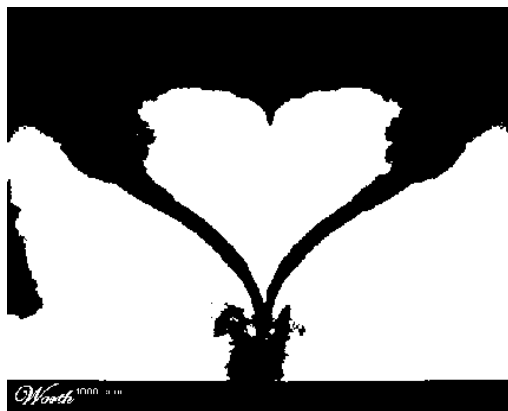
- O resultado será uma matriz $W \times H$, com valores entre 0 e 255;
- Os três canais R,G e B deverão conter a mesma intensidade
- Ilustração do resultado:

Níveis de Cinzento



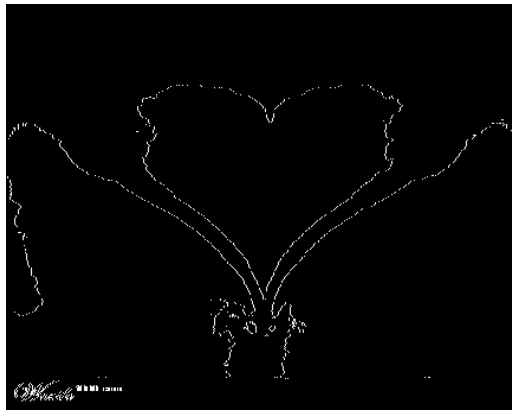
7. Binarize agora a imagem obtida em tons de cinza. Para tal, intensidades superiores a um dado limiar serão convertidas para branco (i.e., 255), sendo os valores inferiores ao limiar convertidos para preto (i.e., 0 → ausência de cor). Crie uma função que receba uma imagem em tons de cinza e o limiar de binarização, e devolva a imagem a dois tons.
 - Nota: não utilize ciclos;
 - A figura abaixo ilustra os resultados para um limiar de 100. Visualize os resultados com vários limiares.

Imagem Binarizada



8. Implemente uma função que receba uma imagem binarizada e devolva outra imagem onde os contornos estejam definidos. Deste modo, transições preto→branco ou branco→preto deverão receber intensidades iguais a 255, sendo que os restantes pixels ficarão a 0. A figura abaixo ilustra os resultados com base na imagem binarizada anterior.

Contorno da Imagem



9. Grave esta imagem em formato *bmp* utilizando o método *imsave* da package `matplotlib.image`.
 - Verifique que a dimensão do ficheiro correspondente à imagem final, apesar de conter apenas duas cores, é substancialmente superior à do ficheiro inicial em formato *jpeg*. De facto, este é um formato de compressão que permite reduzir de forma marcada a dimensão dos ficheiros de imagem. No entanto, trata-se de um método de compressão destrutiva.