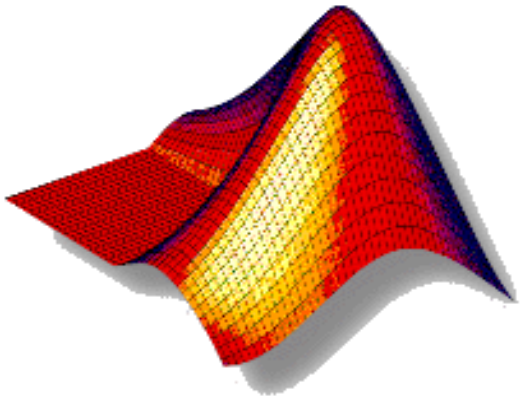
Teoria da Informação

Trabalho Laboratorial Preparatório

Introdução ao Matlab



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 o ambiente e programação em Matlab.

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.m'**. Este script será utilizado na chamada de todas as funções indicadas abaixo (alíneas 2 a 11).
   - Se desejar fazer uma pausa entre a execução de cada ponto, poderá utilizar o comando *pause* do Matlab. Para apresentar uma mensagem explicativa, utilize a função *disp*('mensagem').

2. Leia o ficheiro wave **'saxriff.wav'**. Para tal, utilize a função '*audioread*' do Matlab:
   - [sinal, fs] = audioread ('nomeFicheiroWav')[1]; Para mais detalhes, consulte a ajuda do Matlab. Esta função normaliza a amplitude do sinal para o intervalo [-1; 1]

3. Escute o sinal áudio, com recurso às funções *audioplayer* e *play* (consulte a ajuda do Matlab para mais detalhes).

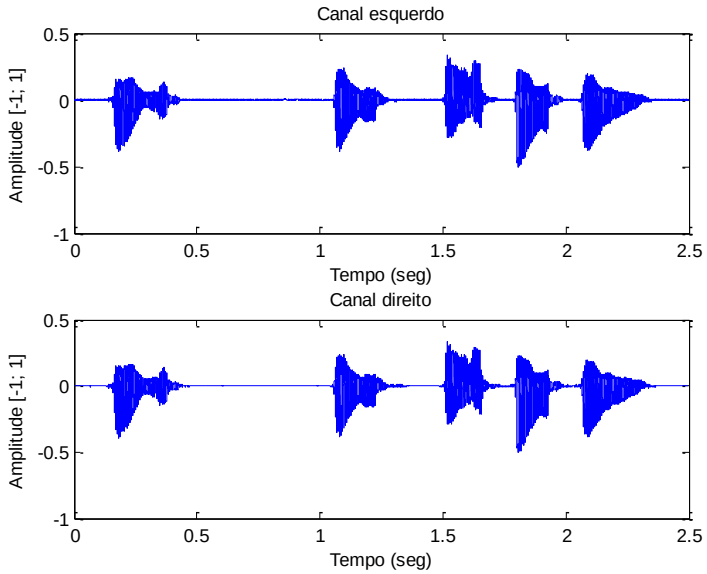
4. Crie a função **apresentarInfo('nomeFicheiro', fs, nrBitsQuant)** que apresente no ecrã informações sobre o ficheiro. Utilize as funções '*sprintf*' e '*disp*'. Para determinar o número de bits de quantização (ou bits por amostra) consulte a função *audioinfo*.

   - Exemplo do resultado da função:

[1] Nota: O formato wav (abreviação de *Waveform Audio Format*) é um standard de armazenamento de áudio. Na sua versão mais comum, os ficheiros .wav contêm áudio não-comprimido no formato PCM (Pulse-Code Modulation). Nesta representação digital, o sinal analógico original é *amostrado* em intervalos uniformes (a frequência de amostragem, f_s, típica, utilizada em ficheiros com qualidade de CD, é de 44.1 kHz) e depois *quantizado* para um conjunto de símbolos num código binário (é implementada habitualmente quantização de 16 bits, i.e., cada amostra é representada com recurso a 16 bits). São estes os parâmetros de saída da função wavread. Para além de ficheiros .wav, o Matlab permite ler ficheiros .au.

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 Ts = 1/fs; duracao = nrAmostras * Ts.
   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* e *size/length*. Dê nomes aos eixos e um título a cada plot realizado.



6. 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 a variável do Matlab *nargin*.

7. Execute o mesmo procedimento do ponto anterior, agora com a função *axis* do Matlab.
   - Nota: a restrição da zona de visualização é conseguida de forma mais simples através da função *axis*([xmin xmax ymin ymax]).

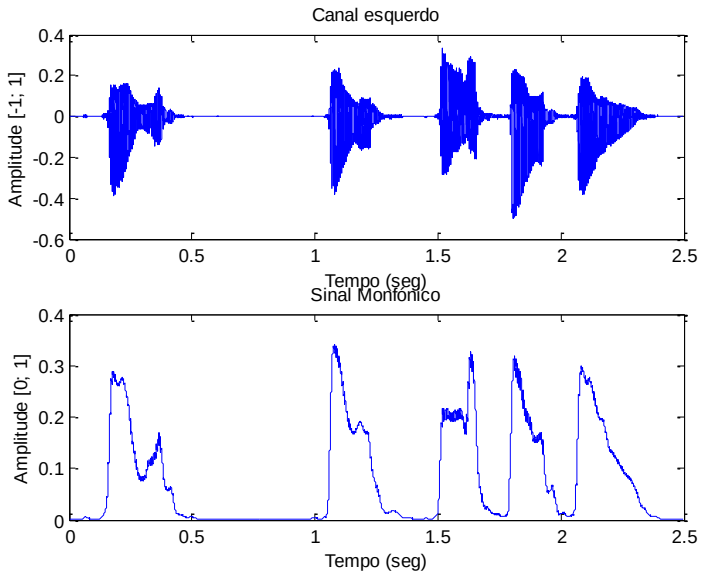
8. Crie uma função que adicione ruído uniforme ao sinal. Esta função deverá receber o sinal original e a amplitude do ruído e devolver o sinal original com ruído.
   - Recorra à função *rand* do Matlab, a qual gera uma matriz de tamanho especificado contendo valores aleatórios no intervalo [0, 1]. Deverá,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

9. 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 Matlab.

10. 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.
    10.1. 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.
    10.2. No seu script, faça soar o novo sinal. Verifique que o som de cada sinal é enviado por uma coluna diferente.
    10.3. Visualize o novo sinal recorrendo à função anteriormente implementada
    10.4. 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.
    10.5. Guarde os resultados do novo sinal áudio. Utilize a função *wavwrite*

11. Implemente uma função que devolva o contorno de amplitude de um sinal[2]. O resultado deverá ser algo do género:

[2] 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.



11.1. Implemente a função **ca = contornoAmplitude(x, W)**, em que W (ímpar) determina a dimensão de uma janela deslizante necessária à determinação do contorno.

11.2. O cálculo do contorno começa por uma operação designada por "rectificação de meia-onda", definida segundo a equação (1):

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

- Utilize a função *find* do Matlab para procurar os índices negativos de x (Exemplo: x = [1 12 4 6 2 15 17]; ind = find (x > 5) devolve os índices 2, 4, 6 e 7; se os quiser multiplicar por 2, basta fazer: x(ind) = x(ind) * 2;

11.3. De seguida, o contorno é determinado pela média móvel dos valores do sinal. i.e., **ca(i) = mean(xr(i-W/2 : i + W/2));** Dado que W é ímpar, utilize a função *floor*, a qual arredonda o valor em causa para o inteiro exactamente anterior.

a) Faça um ciclo para implementar a média móvel
b) De forma a lidar-se com índices negativos ou para além do comprimento do vector, é usual executar uma operação designada por "zero-padding", a qual consiste em adicionar zeros ao início e fim do sinal; deste modo, para um vector coluna, faça: **xr = [zeros(W/2, 1); xr; zeros(W/2, 1)];** Tal como anteriormente, utilize a função *floor*.
c) Finalmente, normalize o contorno obtido, de forma a que a sua energia seja igual à do sinal inicial (utilize a função desenvolvida anteriormente).
d) Visualize os resultados para diversos valores de W, e.g., 5, 7, 21, 25, 75, 255 e 355.

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

1. Crie um script e grave-o com o nome **'mainImage.m'**. 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*' do Matlab: Y = imread ('nomeFicheiro');
   - Esta função permite ler imagens numa série de formatos, entre os quais bmp, gif, jpg, etc.
   - Em imagens monocromáticas (i.e., níveis de cinzento), o resultado (em Y) é uma matriz WxH, em que W é o número de pixeis correspondentes à largura da imagem e H e o número de pixeis 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. Apresente informação sobre o ficheiro, com recurso à função *imfinfo* do Matlab. Verifique qual a dimensão da imagem e o método de codificação.

4. Visualize a sua matriz com recurso à função *image* do Matlab. Para retirar os valores dos eixos, utilize *axis off*.

image008.jpg

5. 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

6. 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 – floor(W/2); j + 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 a função *ones* do Matlab. 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 double, i.e., utilize *double*(x).
   - Sugestão 2: utilize dois ciclos while (*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

7. 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:

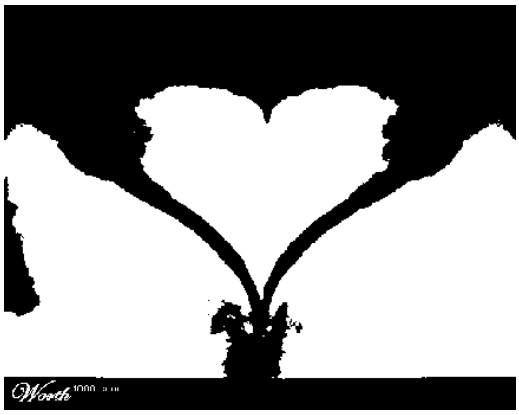
$$Y_{cinza}[i,j] = 0.2978R[i,j] + 0.5870\mathrm{G}[i,j] + 0.1140B[i,j]$$

- O resultado será uma matriz WxH, com valores entre 0 e 255;
- Para visualizar a imagem, deverá utilizar um mapa de cores adequado, neste caso de tons de cinza; para esse efeito, execute *colormap(gray(256))* antes de visualizar a imagem. Para restaurar o mapa de cores original faça *colormap('default')*.
- Ilustração do resultado:

Níveis de Cinzento

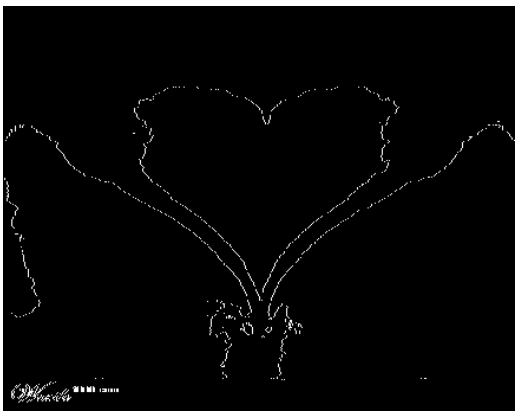
8. Binarize agora a imagem obtida em tons de cinza. Para tal, intensidades superiores a um dado limiar serão convertidas para branco (i.e., 255), sendo os valores inferiores ao limiar convertidos para preto (i.e., 0 → ausência de cor). Crie uma função que receba uma imagem em tons de cinza e o limiar 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



9. 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 pixeis ficarão a 0. A figura abaixo ilustra os resultados com base na imagem binarizada anterior.

Contorno da Imagem



10. Grave esta imagem em formato *png* utilizando a função *imwrite* do Matlab.
    - 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.