Iniciação à manipulação de sinais visuais com o MATLAB

Objectivos: Este trabalho pretende dar a conhecer algumas das funções do Matlab para manipulação de sinais audiovisuais. É uma introdução ao trabalho a realizar para que o estudante se familiarize com o Matlab.

Introdução

O MatLab possui diversos comandos específicos para manipular imagens e para executar diversas técnicas de

processamento de imagens. O MatLab adequa-se particularmente à manipulação e processamento de imagens, uma vez que a sua estrutura básica de dados é a matriz e uma imagem não é mais do que uma matriz. Uma imagem é lida no MatLab e armazenada como uma matriz do tipo uint8 (valor inteiro representado com 8 bits, logo com 28=256 valores possiveis).

Para se poder utilizar os valores dessa matriz em operações de números do tipo double como adição, subtração, multiplicação e divisão, é necessário efectuar a conversão dessa matriz para o tipo double (valores reais de precisão dupla)

Agumas funções built-in do Matlab para importar, exportar e obter informação:

imread - lê ficheiro de imagem

imwrite - grava imagem num ficheiro

imshow - apresenta uma imagem numa janela

imageinfo - cria uma ferramenta que permite obter informação sobre uma imagem gravada em ficheiro

imattributes - lista atributos da imagem que está a ser visualizada

image - visualiza uma matriz qualquer como sendo uma imagem; o valor de cada elemento da matriz especifica a cor do elemento de imagem espacialmente correspondente.

im2frame e frame2im - permitem converter de formato de imagem para formato de filme usado pelo MATLAB e vice-versa.

Exemplo de utilização:

```
>> im = imread ('praia.jpg'); % lê a imagem "praia" com o formato JPEG para a matriz im (MxNx3)
>> [line,col]=size(im(:,:,1)); % lê o número de linhas e colunas da imagem que foi armazenada na matriz im
% e armazena nas variáveis lin e col (variáveis quaisquer); nota: a matriz im tem 3 dimensões já que a
% imagem que foi lida é uma imagem a cores. Nota: o comando [line,col]=size(im) daria o mesmo resultado.
>> imshow (im); % visualiza a imagem que está na matriz im
```

```
>> im2 = double(im); % converte a imagem que está na matriz im para tipo real de precisão dupla e guarda
% na nova matriz im2
>> im2 = im2 + 10; % aumenta o brilho da imagem que está na matriz im2
>> im2 = uint8(im2); % converte a imagem de volta ao tipo inteiro
>> imwrite (im2, 'nome.bmp'); % guarda a nova imagem no ficheiro "nome.bmp" com o formato bitmap
>> imwrite (im2, 'nome2.jpg'); % guarda a nova imagem no ficheiro "nome2.jpg" com o formato jpeg
```

imagens com o formato bitmap (.bmp)

Para imagens armazenadas no formato "bmp" (bitmap com 24 bits) são criadas matrizes a 3 dimensões ou planos para armazenar cada uma das componentes fundamentais de cor R, G, B. Cada dimensão/plano contém uma matriz MxN (nº de linhas vezes nº de colunas) que corresponde à matriz de uma das componentes de cor RGB (exactamente nessa ordem). Para separar cada componente, o seguinte código pode ser utilizado:

```
%  ler imagem RGB de 24 bits, isto é, cada elemento de imagem é representado por 24 bits, sendo 8 bits para
 % a cor vermelha R, 8 bits para a cor verde (G) e 8 bits para a cor azul (B):
>> im = imread('praia.bmp');

 % verificar que se trata de uma matriz de n pixels por m colunas com profundidade 3, isto é uma matriz a 3
% dimensões. a profundidade 3 indica que na prática existe uma matriz para cada uma das cores primárias
% RGB.
>>size(im)

% atribuir todas as linhas e todas as colunas do primeiro plano da matriz im à variável r , as do segundo
% plano à variável g e as do terceiro plano à variável b
>> r = im(:, :, 1);
>> g = im(:, :, 2);
>> b = im(:, :, 3);
% obter as dimensões de cada uma das matrizes
>> [lin, col, plano] = size(im);

% criar uma nova imagem preta (matriz de zeros) com 24 bits por elemento, usando as dimensões da
% imagem lida:
>> im2 = uint8(zeros(lin, col, plano)); % criada uma imagem só de zeros

% Cada plano dessa nova matriz recebe um dos planos r, g ou b da outra imagem, incrementados e %
decrementados de um dado valor
>> im2(:, :, 1) = r+50;
>> im2(:, :, 2) = g-50;
>> im2(:, :, 3) = b+100;

% visualizar e comparar as duas imagens que estão armazenadas nas matrizes im e im2:
>>figure; imshow(im);
>>figure; imshow(im2);
```

Outras funções para alterar representações, valores e fazer filtragens

rgb2gray - conversão de 24 bits para tons de cinza
im2bw - conversão para preto-e-branco
rgb2ind - conversão de 24 bits para 256 cores
rgb2hsv - conversão entre os espaços de cor RGB e HSV
rgb2ycbcr - conversão entre os espaços de cor RGB e YCbCr
imresize - altera as dimensões de uma imagem
imrotate - roda uma imagem de um determinado ângulo

imhist - calcula o histograma de uma imagem (registo do nº de vezes que cada valor ocorre numa imagem)
filter2, imfilter - aplica um filtro a uma imagem e devolve uma matriz de valores reais com a imagem filtrada

B = imresize(A, M, 'method')

A função "imresize" acima retorna uma matriz que é M vezes maior (ou menor) que a imagem A, onde 'method' pode ser um de:

nearest = vizinho mais próximo
bilinear = interpolação bilinear
bicubic = interpolação bicúbica

B = imrotate(A, Ângulo, 'method');

A função "imrotate" acima retorna uma matriz que é uma versão rodada de Ângulo da imagem A, onde 'method' pode ser um de [nearest, bilinear, bicubic].

Exemplos:

```
>> A = imread ('fruta.bmp');
>> figure(1); imshow(A); title('imagem original');
>> B = imresize (A, 0.5, 'nearest');
>> figure(2); imshow(B); title('imagem redimensionada');
>> B = imrotate (A, 45, 'nearest');
>> figure(3); imshow(B); title('imagem rodada');
% filtrar uma imagem; é necessário arredondar e converter para uint8 para se poder visualizar a imagem
% filtrada; usa-se fspecial para criar o tipo de filtro desejado (no exemplo abaixo são criados três tipos de,
% filtro: de média, de movimento e de realce de contornos
>> im = imread('nenufares.bmp');
>> im = rgb2gray(im);
>> subplot(2,2,1);imshow(im);title('Imagem original escala cinza');
>> h = fspecial ('average', 5);
>> im2 = uint8(round(filter2(h, im)));
>> subplot(2,2,2);imshow(im2);title('Imagem filtro media');
>> h= fspecial('motion',20,45);
>> im3= imfilter(im,h,'replicate');
>> subplot(2,2,3);imshow(im3);title('Imagem filtro motion');
>> h=fspecial('unsharp');
>> im4 = imfilter(im,h,'replicate');
>> subplot(2,2,4);imshow(im4);title('Imagem mais nitida');
```