CIM 2023 Introductory Lab Assignment - Visual Part - basics of Matlab for visual signals

Assignment Name: **Basics of Matlab for visual signals manipulation**
Due Date: **March 7th, 2023.** No submission is required.

Purpose

This assignment intends to make known some of Matlab's functions for manipulating audiovisual signals. It is an introduction to the work to be done so that the student becomes familiar with Matlab. It merely presents some Matlab functions and proposes a number of operations so that the student performs some "hands-on" work.

Why is it important that multimedia students gain the listed skills and knowledge? Because Matlab is a scientific programming tool that has many functions developed specifically for performing a large number of operations on media signals. Knowing the basics of using Matlab for manipulating visual signals is an added value for experimenting easily with a number of operations that many times are performed as preparatory steps in visual signals compression.

Skills

In this assignment, you'll be learning how to:

1) Import, visualise and export images in Matlab;
2) Access images available in Matlab
3) obtain characteristics of the imported file
4) Access individual data or values of an image
5) Convert formats of images
6) Change properties of images, namely dimensions
7) Create an image histogram
8) Apply a filter to an image
9) Obtain help on commands and functions
10) Access demos and examples

Knowledge

In this assignment, you'll be learning about:

1) The way that Matlab represents images
2) Some of the basic functions that Malab offers in its Image Processing toolbox
3) The demonstration Matlab has available
4) The usefulness of the MathWorks repository as a source of information, clarification and inspiration

Overview

MatLab has several specific commands and functions to manipulate images and perform various image processing techniques. MatLab is particularly suited to image manipulation and processing since its basic data structure is the matrix and an image is nothing more than a matrix. An image is read in MatLab and stored as a matrix of type uint8 (integer value represented with 8 bits, therefore with 2^8=256 possible values).

In order to be able to use the values of this matrix the type "double" in arithmetic operations such as addition, subtraction, multiplication, and division, it is necessary to convert this matrix to the double type (double precision real values).

Here are some Matlab built-in functions for importing, exporting, and obtaining information:

imread - read image file

imwrite - write image to file

imshow - displays an image in a window

imageinfo - creates a tool that allows obtaining information about an image saved in a file

imattributes - list attributes of the image being displayed

image - visualizes any matrix as an image; the value of each matrix element specifies the color of the spatially corresponding image element.

im2frame and frame2im - allow converting from image format to movie format used by MATLAB and vice versa.

Availability of images in Matlab

Matlab has available a number of images with varying formats that we can import and work with in Matlab. We need simply type in their name without indicating the full path. When we are using our own images, then it is necessary to indicate the full path so that Matlab knows where to find them.

For example, we could import the image "peppers.png" listed below:

>> i=imread('peppers.png');

Or the image "test.jpg" located in C:\Users\Teresa Andrade\Documents\MATLAB\images:

>>j=inmread ('C:\Users\Teresa Andrade\Documents\MATLAB\images\test.jpg');

Note that it is possible to provide relative paths in respect to the directory where you are running Matlab. If you are running Matlab in C:\Users\Teresa Andrade\Documents\MATLAB, then the command to import the "test.jpg" image could be:

>>k=imread('images\test.jpg');

These are some of the available images, listed by format:

Uncompressed		Compressed
PNG	TIFF	JPEG
blobs.png	board.tif	baby.jpg
circlesBrightDark.png	cameraman.tif	car_1.jpg
coins.png	canoe.tif	car_2.jpg
coloredChips.png	circuit.tif	car1.jpg
DistortedImage.png	eight.tif	car2.jpg
hands1-mask.png	forest.tif	flamingos.jpg
kobi.png	kids.tif	foggyroad.jpg
liftingbody.png	m83.tif	foggysf2.jpg
lighthouse.png	mandi.tif	foosball.jpg
onion.png	moon.tif	football.jpg

Uncompressed		Compressed
PNG	TIFF	JPEG
pears.png	pout.tif	greens.jpg
peppers.png	tire.tif	hallway.jpg
saturn.png	trees.tif	hands1,jpg
tape.png	foosballraw.tiff	indiancorn.jpg
text.png		llama.jpg
toysflash.png		lowlight_1.jpg
		lowlight_2.jpg
		micromarket.jpg
		office_4.jpg
		parkavenue.jpg
		peacock.jpg
		sevilla.jpg
		sherlock.jpg
		strawberries.jpg
		trailer.jpg
		wagon.jpg
		yellowlily.jpg

In the Windows and Mac OS images should be located under (check the Matlab version):

/Applications/MATLAB_R2021a.app/toolbox/images/imdata

Hands-on work

Importing and visualising images

Type in the following commands on your Matlab workspace.

>> im = imread('praia.jpg'); % reads the “praia” image with the JPEG format into the matrix im (MxNx3)

>> [line,col]=size(im(:,:,1)); % reads the number of rows and columns of the image that was stored in the matrix im

% and stores it in the variables lin and col (any variables); note: the matrix im has 3 dimensions since the

% image that was read is a color image. Note: the command [line,col]=size(im) would give the same result.

>> imshow(im); % displays the image that is in the matrix im

>> im2 = double(im); % converts the image that is in the im array to double precision real type and saves

% in the new matrix im2

>> im2 = im2 + 10; % increases the brightness of the image that is in the im2 matrix

>> im2 = uint8(im2); % converts image back to integer type

>> imwrite(im2, 'name.bmp'); % saves the new image in the file “nome.bmp” with bitmap format

>> imwrite(im2, 'name2.jpg'); % saves the new image in the file “nome2.jpg” with jpeg format

Using images in bitmap format (.bmp)

For images stored in the “bmp” format (24-bit bitmap) 3-dimensional matrices or planes are created to store each of the fundamental color components R, G, B. Each dimension/plane contains a matrix MxN (number of lines times # of columns) which corresponds to the matrix of one of the RGB color components (in exactly that order). To separate each component, the following code can be used:

Type in the following commands on your Matlab workspace.

% read 24-bit RGB image, that is, each image element is represented by 24 bits, 8 bits for

% the red color R, 8 bits for the green color (G) and 8 bits for the blue color (B):

>> im = imread('praia.bmp');

% check that it is a matrix of n pixels by m columns with depth 3, that is a matrix of 3

% dimensions. depth 3 indicates that in practice there is a matrix for each of the primary colors

% RGB.

>>size(im)

% assign all rows and all columns in the foreground of the matrix im to the variable r, those in the background

% plane to the variable g and those of the third plane to the variable b

>> r = im(:, :, 1);

>> g = im(:, :, 2);

>> b = im(:, :, 3);

% get the dimensions of each of the matrices

>> [lin, col, plane] = size(im);

% create a new black image (matrix of zeros) with 24 bits per element, using the dimensions of the

% image read:

>> im2 = uint8(zeros(lin, col, plane)); % created a zero-only image

% Each plane of this new matrix receives one of the r, g or b planes of the other image, incremented and % decremented by a given value

>> im2(:, :, 1) = r+50;

>> im2(:, :, 2) = g-50;

```
>> im2(:, :, 3) = b+100;
```

```
% view and compare the two images that are stored in im and im2 matrices:
>>figure; imshow(im);
>>figure; imshow(im2);
```

Other functions for changing representations, values and filtering

rgb2gray - 24-bit conversion to grayscale

im2bw - conversion to black and white

rgb2ind - 24-bit to 256-color conversion

rgb2hsv - conversion between RGB and HSV color spaces

rgb2ycbcr - conversion between RGB and YCbCr color spaces

imresize - change the dimensions of an image

imrotate - rotates an image from a certain angle

imhist - calculates the histogram of an image (recording the number of times each value occurs in an image)

filter2, imfilter - applies a filter to an image and returns an array of real values with the filtered image

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

The “imresize” function above returns an array that is M times larger (or smaller) than image A, where 'method' can be one of:

nearest = nearest neighbor

bilinear = bilinear interpolation

bicubic = bicubic interpolation

```
B = imrotate(A, Angle, 'method');
```

The “imrotate” function above returns an array that is a rotated version of Angle of image A, where 'method' can be one of [nearest, bilinear, bicubic].

Examples:

```
>> A = imread('fruit.bmp');
>> figure(1); imshow(A); title('original image');
>> B = imresize(A, 0.5, 'nearest');
>> figure(2); imshow(B); title('image resized');
>> B = imrotate(A, 45, 'nearest');
```

```
>> figure(3); imshow(B); title('rotated image');
% filter an image; it is necessary to round and convert to uint8 to be able to visualize the image
% filtered; fspecial is used to create the desired filter type (in the example below three types are created,
% filter: average, motion, and edge enhancement
>> im = imread('nenufares.bmp');
>> im = rgb2gray(im);
>> subplot(2,2,1);imshow(im);title('Original grayscale image');
>> h = fspecial('average', 5);
>> im2 = uint8(round(filter2(h, im)));
>> subplot(2,2,2);imshow(im2);title('Image filter media');
>> h= fspecial('motion',20,45);
>> im3= imfilter(im,h,'replicate');
>> subplot(2,2,3);imshow(im3);title('Motion filter image');
>> h=fspecial('unsharp');
>> im4 = imfilter(im,h,'replicate');
>> subplot(2,2,4);imshow(im4);title('Sharper image');
```

Built-in demos

There are a number of demonstrative scripts in Matlab that you can run and also examine the code to see how certain function work. Run the bellow command in the Matlab workspace to obtain a listing of built-in demos. You can also find many examples on the MathWorks website.

```
>>help imdemos
```

MathWorks

Info on Matlab built-in functions

Use the MathWorks Web site to obtain more information on the Matlab built-in functions of the Image Processing Toolbox. You may also obtain more explanation for each function by using the 'help' command followed by the name of the function in the Matlab workspace.

https://www.mathworks.com/help/images/referencelist.html?type=function&category=getting-started-with-image-processing-toolbox&s_tid=CRUX_topnav

By selecting a category, a list of relevant functions appears. By selecting one of such functions you get access to full documentation but also to some examples.

- Select the category "Image Filtering and Enhancement" to obtain a list of related built-in functions.
    - Select the 'imfilter" function. By selecting it we obtain access to its documentation but also to available examples on how to use it. Analyse the documentation and the example inline commands. Then select "Examples" on the top and try the example "Filter Grayscale and Truecolor (RGB) Images Using imfilter Function"
    - Repeat for the "fspecial" function. Go through the documentation and inline example commands.

- Select the category “Import, Export and Conversion”. Under the group “Color space Conversion” select “rgb2ycbcr”. Analyse the documentation and then run the examples.

MathWorks examples

You can also find many examples on the MathWorks website

https://www.mathworks.com/help/overview/examples.html?category=image-processing-and-computer-vision&s_tid=CRUX_topnav