Image and Video processing
Laboratory 1
Getting started with Matlab, Quantization, Sampling, Filtering, 2DFT, Weber law

Prof. Touradj EBRAHIMI

Reports submission – !!! read carefully !!!

Students will have to produce and submit a report two weeks after completion of each laboratory session. An electronic version of the report is submitted via Moodle's interface for uploading assignments. There will be a hard deadline for each submission. Late submissions will have to be justified and explained to Ivan Ivanov by email. A structure of the report is explained below. In addition to the report, the source code produced should also be submitted in electronic form. The report and the source code should be submitted in **ONE ZIP** file using Moodle platform.

Structure of the reports and m-files

- the name of the final ZIP file will be: `lab_`*number-of-lab*`_`*your-surname*`.zip`
- the final ZIP file will contain
  - final m-file called `run_lab_`*number-of-lab*`_`*your-surname*`.m`
  - all created m-files and functions which are necessary to obtain the results
  - all source images and data needed for successful running of the final m-file
  - an electronic version of your report in **PDF** format
- the structure of the final m-file
  - the final m-file will contain all necessary commands and functions, by running this m-file one must get all results[1]
  - submitted m-files will be commented
  - each figure will be properly titled
  - the final m-file will be divided into cells[2] according to the example bellow
- the parts of program codes should not be included in the final report
- description of the results should be a part of the final report
- don't forget to answer all the questions in your report
- check whether your final m-file can be launched without problems - only then submit your report

[1]of course it is upon you whether you want to include everything in the final m-file, or create different functions which you will call in the final m-file

[2]those who do not know how to use the cells in Matlab to create the program code more transparent and easier to read, they should look at `http://www.mathworks.com/demos/matlab/developing-code-rapidly-with-cells-matlab-video-tutorial.html`

An example of final m-file

```
% Lab (number of lab) - your name and date

%% Exercise (number of exercise) - "Name of exercise"

% your own program code with your coments
a = imread('picture.tif');
% all figures will be properly titled
figure('name','Name Of The Figure')
imshow(a,[]);

%% Exercise (number of exercise) - "Name of exercise"

% your own program code with your coments
[output1, output2, ..., outputN] = function_name (input1, input2, ..., inputN);
```

1 Introduction

1.1 Images in matlab

Matlab can use three types of data to store images and four formats.

The three types of data are:

uint8 Pixel values are coded as unsigned 8 bits integers. Possible values range from 0 to 255. This format is not suitable for algebraic operations.

uint16 Pixel values are coded as unsigned 16 bits integers. Possible values range from 0 to 65535. This format is similar to the pervious one but it offers a wider range of values. it's rarely used.

double Pixel values are coded in the floating point format using 64 bits. They can therefore take all possible reel values. This format uses more memory than the previous 2 formats. It enables us to perform efficiently all algebraic operations.

The four image formats are:

indexed The image is stored in a 2D matrix containing indeces to an associated color table (*colormap*). For data of type `uint8` and `unit16` the value 0 corresponds to the first line of the *colormap*, 1 to the 2nd, and so on. For data of type `double` the value 1.0 corresponds to the first line of the *colormap*, 2 to the 2nd, and so on.

RGB or truecolor The image is stored in a 3D matrix of dimension $m \times n \times 3$. The 3rd dimension gives the R, G, B values of the pixel.

intensity The image is stored in a 2D matrix. Each element of the matrix corresponds to the grey level of the pixel.

binary The image is stored in a 2D matrix with binary (0 or 1) coefficients. 0 corresponds to black and 1 corresponds to white. Only data storage in `uint8` and `double` are possible for this format.

colormap The table of colors associated with an image. It's a matrix of type `double` of m lines and 3 columns. Each line gives the R, G, B values of a color. The coefficients take values between 0.0 and 1.0.

Note: The functions `double`, `uint8`, `rgb2ind`, `gray2ind`, etc., are used to convert from one format to another.

1.2 Matlab online help

The online Matlab help is available on the web `http://www.mathworks.com/access/helpdesk/help/helpdesk.html`. For each command, `help` gives detailed information in the Matlab terminal.

1.3 Functions and Matlab scripts

In Matlab, we can define functions or execute commands using a *script*. Use the online help for information on `function` and `script`. The commands `diary`, `load` and `save` might be useful too.

2 Exercises

2.1 Images and color tables

1. read and show the following images:
   - `trees.tif` (image with *indexed* format)
   - `lena.tif` (image with *truecolor* format)
2. Show the images in gray level and invert them (i.e. show the negative). For `trees.tif` do this by modifying its color table.
3. Using a function, modify the color table according to the following rule:

$$\begin{aligned} r' &= r^\gamma \\ g' &= g^\gamma \\ b' &= b^\gamma \end{aligned}$$

   where r, g, b are the original color values, r', g', b' are the new values and γ is the correction factor.

   Show the modified images for different values of γ (typically between 0.5 and 2) and deduce the utility of the gamma correction.
4. Create the image of a chess board (8x8) with blue and yellow squares for the 2 following formats :*indexed* and *truecolor*. Save them to the *TIFF* format.

Note: In this exercise, we can use the functions `imread`, `imwrite`, `image`, `imshow`, `imagesc`, `truesize`, `colormap`, `repmat`, `zeros`, `ones`, `end`, etc.

2.2 Image quantization

Uniform quantization[3] of a sample x is defined as $Q(x) = \lfloor x/\Delta \rfloor$, where Δ is the *quantization step size*. The operator $\lfloor y \rfloor$ (`floor`) rounds y to the closest integer smaller than y itself (i.e. rounds to $-\infty$). The reconstructed value is $\hat{x} = Q(x)\Delta + \Delta/2$.

Apply uniform quantization to the image `lena-y.png` in the grey level. Adjust the quantization step to obtain successively 128, 64, 32, 16, 8, 4 and 2 gray levels. Show the images to determine the number of grey levels after which the phenomenon of "false contours" appears.

[3]this definition is convenient for an even number of quantization levels.

2.3 Filtering

We have the following 5×5 separable 2D filter:

$$\begin{bmatrix}0.0357\\0.2411\\0.4464\\0.2411\\0.0357\end{bmatrix}\begin{bmatrix}0.0357 & 0.2411 & 0.4464 & 0.2411 & 0.0357\end{bmatrix} =$$

$$\begin{bmatrix}0.00127449 & 0.00860727 & 0.01593648 & 0.00860727 & 0.00127449\\0.00860727 & 0.05812921 & 0.10762704 & 0.05812921 & 0.00860727\\0.01593648 & 0.10762704 & 0.19927296 & 0.10762704 & 0.01593648\\0.00860727 & 0.05812921 & 0.10762704 & 0.05812921 & 0.00860727\\0.00127449 & 0.00860727 & 0.01593648 & 0.00860727 & 0.00127449\end{bmatrix}$$

Calculate and show the frequency response (magnitude only) of this filter. Then, apply the filter to the image `gold-text.png` (convolution between the filter and the image) and show the result.

Now consider the 3×3 filter

$$\frac{1}{6}\begin{bmatrix}-1 & -4 & -1\\-4 & 26 & -4\\-1 & -4 & -1\end{bmatrix}$$

Calculate and show it's frequency response, Then, apply the filter to the resulting image in the previous step. What are your conclusions?

Note: In this exercise, we can use the following functions `freqz2`, `imfilter`, `conv2`, `freqspace`, `mesh`, `meshgrid`, etc.

2.4 Correlation

The `gold-text.png` is a natural image in which a text is inserted. Find the position of the letter **g** using the image `g-letter.png` and the correlation operator. Implement the correlation function in the spatial domain and the frequency domain.

Repeat the operation by adding a noise to the natural image with a normal distribution with standard deviations 5, 10, 25, 40 and 50 (for a dynamic range $[-128, 127]$). What is the shape of the correlation product? What are your conclusions?

N.B.1: Apply the correlation function between images with zero nominal average . i.e. with a symmetric dynamic range with respect to zero.(ex. $[-128, 127]$, or $[-0.5, 0.5]$).

N.B.2: Don't forget to take into account the translation introduced by the calculation of correlation in the Fourier domain.

Note: In this exercise, we can use the following functions `conv2`, `fft2`, `real`, `max`, `randn`, `pixval`, `find`, etc.

2.5 Resampling

Down sample the image `sub4.tif` by a factor 2 in each direction, by taking one pixel out of two. Show the resulting image. Repeat the operation to the resulting image to obtain an image down-sampled by a factor of 4.

Describe the results?

2.6 Phase and magnitude of the 2DFT

1. Calculate the 2D Fourier transform (2DFT) of the image `lena-y.png`. Then, calculate the inverse 2DFT by deleting the imaginary part . Repeat the operation by deleting the real part this time. Show the images and explain the results by only using image operations in the spatial domain (i.e. not passing by operations in the Fourier domain).

2. Calculate the 2DFT of the image `lena-y.png`. Then, calculate the inverse DFT by setting the phase of the 2DFT to zero. Show the result (real part). Repeat the operation by setting the magnitude to 1 and maintaining the phase as it is. Give an interpretation of the results.

Note: The 2DFT of an image $x(k,l)$, of dimension $K \times L$, is defined as

$$X(m,n) = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} x(k,l)e^{-2j\pi(\frac{mk}{K}+\frac{nl}{L})} = \sum_{k=0}^{K-1}\sum_{l=0}^{L-1} x(k,l)W_K^{mk}W_L^{nl}$$

for $m = 0,\dots,K-1$; $n = 0,\dots,L-1$ and or $W_N \triangleq e^{-\frac{j2\pi}{N}}$.

2.7 Weber law

1. Create a Matlab function named `weber` taking three parameters of intensity L_1, L_2 and L_b. This function returns a 2D matrix, described by figure 1.

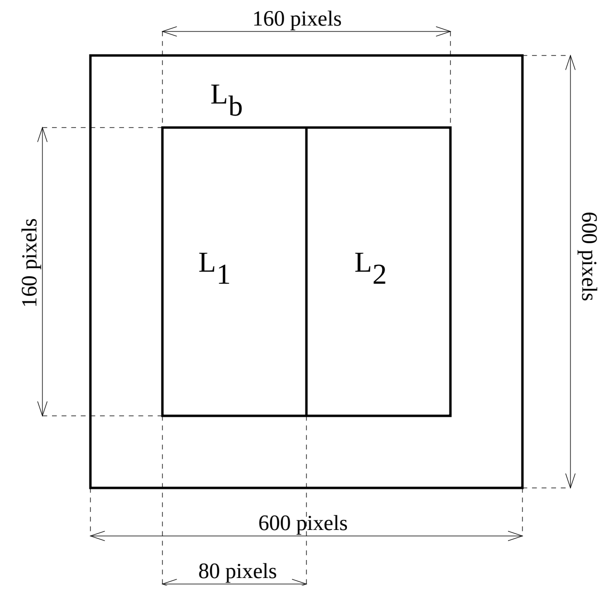

Figure 1: Test image for Weber constant determination

2. Determine your Weber constant. Start by taking the value $L_b = 10$. By varying the intensities L_1 and L_2, determine the weber constant experimentally.

$$\alpha \approx \frac{\Delta L}{L_1}, \tag{1}$$

where $\Delta L = L_2 - L_1$ is the minimal intensity difference we can perceive.

3. Same question, but by taking this time $L_b = 200$. Comment the results.

Note: it's important to perform the whole weber experience under the same ambient illumination conditions and the same monitor settings.