ETRO VUB-DEPARTMENT OF ELECTRONICS AND INFORMATICS

VUB VRIJE UNIVERSITEIT BRUSSEL

Image processing

Exercises 2025

Adrian Munteanu (course titular)
Yangxintong Lyu, Xinxin Dai, Ran Zhao and Olivier Ducastel (TAs)

0 Introduction

0.1 Assistants - Coordinates

Yangxintong Lyu
VUB – Department of Electronics and Informatics (ETRO)
Pleinlaan 9, PL9.2.36
lyangxin@etrovub.be

Xinxin Dai
VUB – Department of Electronics and Informatics (ETRO)
Pleinlaan 9, PL9.2.35
xdai@etrovub.be

Olivier Ducastel
VUB – Department of Electronics and Informatics (ETRO)
Pleinlaan 9, PL9.2.36
oducaste@etrovub.be

Ran Zhao
VUB – Department of Electronics and Informatics (ETRO)
Pleinlaan 9, PL9.2.35
rzhao@etrovub.be

0.2 Contents

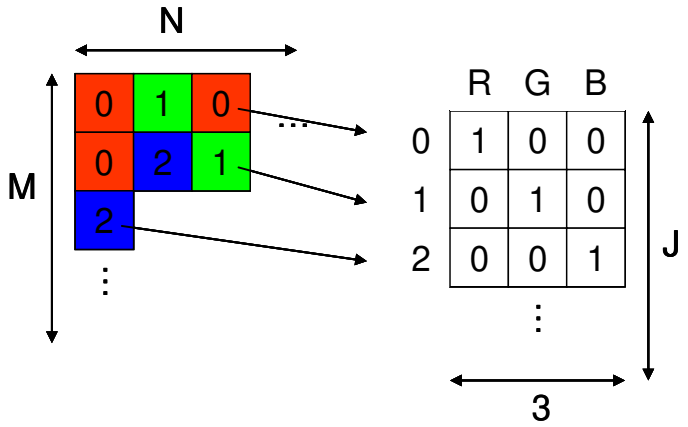
0.3 Matlab and image processing

During these exercises we will use the image processing toolbox available in Matlab. In the first session we will go over the basic commands of Matlab and of this toolbox.

0.3.1 Types of images

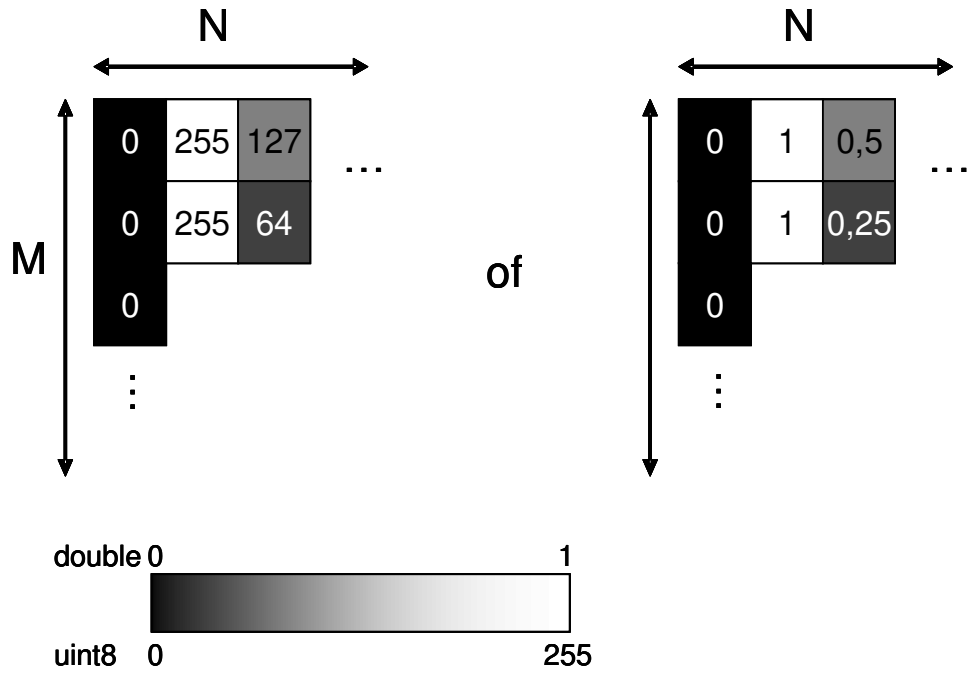
Matlab works with 4 types of images: indexed images, greyscale images, RGB images and binary/monochrome images. Indexed images are represented by two matrices, one matrix (with the same size as the image itself) containing the image data and another containing a colour table. In the image matrix a colour table index is stored for each pixel. For each index the colour table contains the red, green, and blue intensities in double format (intensity between 0 (black) and 1 (white)). For a colour table with J entries, this matrix has a size of Jx3.
The second type of image is the greyscale image. This is represented by a matrix with the same dimensions as the image itself, in which the intensity of each pixel is stored. This can be done in two ways: as a double value (values between 0 and 1) or as an UINT8 value (values between 0 and 255).

RGB images are stored in a three dimensional table (matrix). If the image has N x M pixels, the dimensions of the matrix shall be N x M x 3. For each pixel the R, G and B intensities that define the colour of that pixel are stored into that structure. The intensities can be stored as double or UINT8.
Finally, binary images are represented by a matrix where each pixel can have two possible values, on or off. This can be done with UINT8 or double values (0 is false, 1 is true). The figures below summarize this:



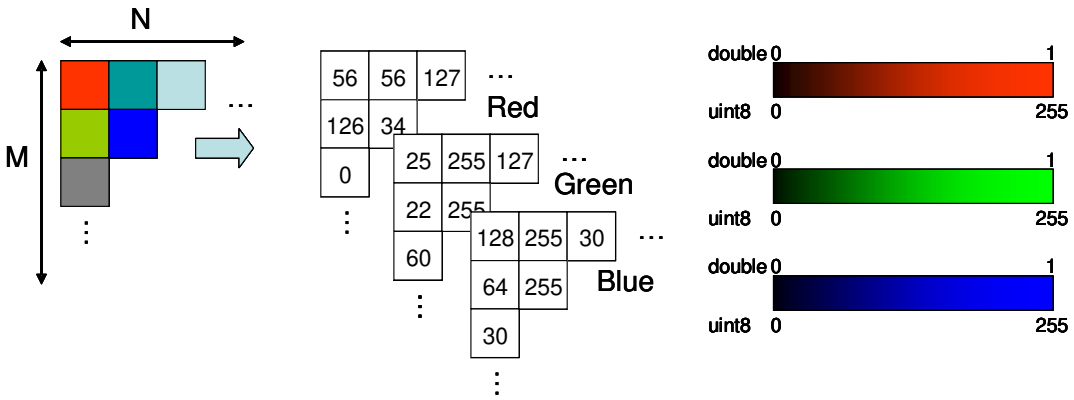
Two matrices: a first matrix with dimensions equal to the image dimensions (MxN) that contains the indexes between 0 and J-1. These indexes refer to the rows in the second matrix (color table) with dimensions Jx3 where the red, green and blue intensities of the indexed colors are stored (in double format, intensities between 0 and 1).

Figure 1: Indexed image.



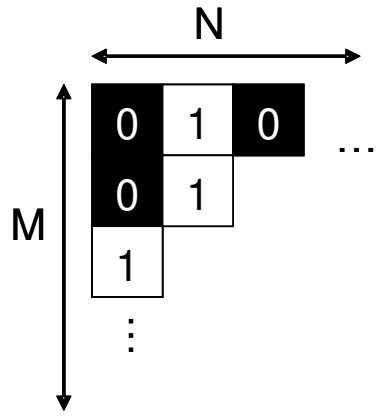
1 MxN matrix with intensity values in double or uint8 format.

Figure 2: Greyscale or intensity image.



1 MxNx3 array with intensity values for red, green and blue in double or uint8 format.

Figure 3: RGB images.



1 MxN matrix with boolean values (0 or 1)

Figure 4: Monochrome image.

0.3.2 Conversion between different types of images

Matlab has several commands to convert an image to another type of image: To convert an indexed image to an RGB image we use the command *ind2RGB*.
e.g.: rgbcolor=ind2RGB(A,Amap);
To go from an indexed image to a greyscale image we use *ind2gray*.
e.g.: gray=ind2gray(A,Amap);
These are only two examples out of the wide variety of conversion functions available. See the Matlab help for more information.

0.3.3 Reading images from disk and displaying them on screen

The generic Matlab command to read images from disk is *imread*. To read greyscale images or RGB images the syntax is:
A=imread('filename',type)
For indexed images:
[A,Amap]=imread('filename',type)
A in this case is the image data, while Amap contains the colormap.
"type" can be one of the following values depending on the image format:

'jpg' or 'jpeg'	Joint Photographic Experts Group (JPEG)
'tif' or 'tiff'	Tagged Image File Format (TIFF)
'bmp'	Windows Bitmap (BMP)
'png'	Portable Network Graphics
'hdf'	Hierarchical Data Format (HDF)
'pcx'	Windows Paintbrush (PCX)
'xwd'	X Window Dump (XWD)

Except for the colour table of an indexed image, the *imread* function generates matrices with uint8 values. These image representations are not suitable for image processing since arithmetic operations are only possible with double values in Matlab. The conversion to intensities of the double type (between 0 and 1) can be done by using the im2double command.

To show an image on screen we use the *imshow* command.
For indexed images the following syntax applies:
imshow(A,Amap);
For RGB and greyscale images the syntax is:
imshow(A);
In both cases the image is shown in the last opened image window. If there aren't any windows open a new one is automatically opened. New image windows are opened with the following command:
figure;

0.3.4 Overview of the standard Matlab syntax

Matlab is based on computations with matrices. All variables are matrices. Matrices are indexed from 1 (and NOT from 0 as in C ☹). Avoid using variable names i and j as these names are used in the representation of complex numbers. Here is a table with useful commands:

A=[1 2 3 ; 4 5 6 ; 7 8 9]	Creates the matrix A=$\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}$. After each ; a new row is started, elements in the same row are separated by a space.
A=[1 2 3]	Creates the row vector A=$\begin{pmatrix}1&2&3\end{pmatrix}$.
A=[1 ; 2 ; 3]	Creates the column vector A=$\begin{pmatrix}1\\2\\3\end{pmatrix}$.
B=A'	' is the symbol to transpose a matrix. In this case B is the transposed matrix of A.

C=B*A	The multiplication of the matrices A and B
C=B.*A	Only possible when A and B have the same dimensions. The elements of C are calculated by multiplying the corresponding elements in A and B with each other.
C=B./A	Identical for division.
A=size(B)	Returns the dimension of B as a row vector with the first element representing the number of rows and the second element the number of columns.
A=zeros(M,N)	Creates a matrix A with M rows and N columns filled with zeros.
A=ones(M,N)	Creates a matrix A with M rows and N columns filled with ones.
det(A)	Computes the determinant for A.
A(k,l)	Returns the element on row k and column l in matrix A.
A(:,l)	Returns the l^{th} column of matrix A
A(k,:)	Returns the k^{th} row of matrix A
A(a:b,c:d)	Returns the submatrix of A specified by the rows a to b and the columns c to d
;	Every line closed with a ; does not give any output on screen
Ctrl-C	Cancel running command.
clc	Clear command window
who	Overview of registered variables.
help command1	Returns help about command1
clear	Clear all registered variables.

Matlab is also a programming language. The most relevant statements are:

%bla bla	% indicates the beginning of a comment line (equivalent with // in C)
for k=1:5 ... end	for loop
while a>0 ... end	while loop
if a==b ... elseif b>c	if...then

... else end	
function z=dosomething(param1,param2) .... z=...	Function with return value *z* and parameters *param1* and *param2*. The name of the m-file where the function is saved has to be the same as the name of the function. The return value is set by an assignment in the function.

0.3.5 Exercise 1

Read in the indexed image *trees.tif* with the following command
[aind,amap] = imread('trees.tif','TIF');
and show it on screen with imshow(aind,amap). Convert the image to RGB values with the ind2rgb (also see Matlab help) command and check the transformation for 1 pixel position (warning: a value *i* in aind corresponds with the index *i+1* in the colormap because in Matlab arrays are indexed from 1 to N instead of from 0 to N-1). Next use ind2gray to convert the image to an intensity image. Use imshow to display both images. The conversion to greyscale is done by first converting to RGB values followed by a conversion to the YUV colour space. In de YUV colour space each colour consists of a luminance component Y (the grey value we are looking for) and two chrominance components U en V. The transformation matrix is as follows:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Check for 1 pixel if the conversion was correct (rounding error can be ignored).

0.3.6 Exercise 2

The effect of a colormap on an indexed image. Define the following matrices:

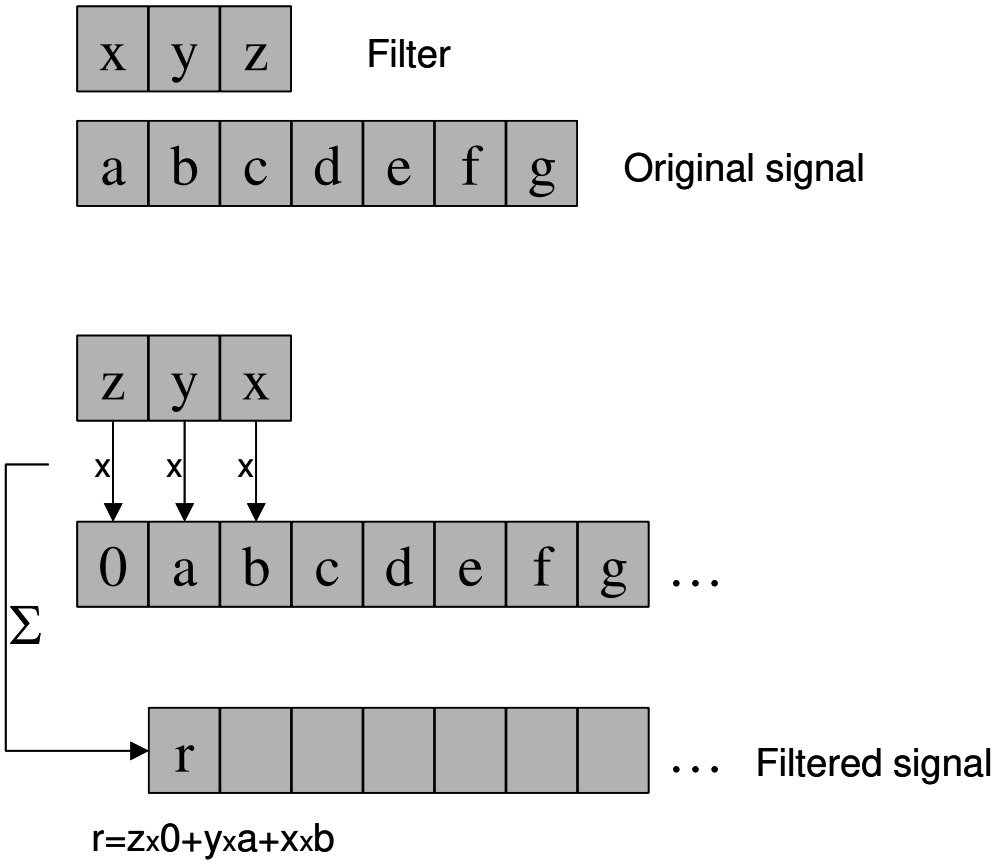
```
X=[1 2 3;3 1 2;2 3 1];
X_map=[1 0 0;0 1 0;0 0 1];
```

Look at the image with imshow(?,?,'InitialMagnification'). Change the colormap to turn the blocks into cyan, magenta and yellow. Show the image again with imshow. Change the colormap to turn the blocks black, grey and white.

0.3.7 Exercise 3

This exercise is meant to practice MATLAB programming. Write a module that performs filtering on a one dimensional signal for filters with an odd number of samples. Use zeros to fill in the borders. Next, specify an average filter with 3

samples (1/3[1 1 1]) and filter a random 1 D signal. The figure below describes the filtering operation:



Check you results with the Matlab conv command.

1 Image transformations

1.1 The DFT

For an image s with NxN pixels

Forward transformation:

$$coeff(k,l) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} s(i,j)\cdot e^{-j2\pi\frac{(ik+lj)}{N}}$$

Inverse transformation:

$$s(i,j) = \frac{1}{N^2}\sum_{l=0}^{N-1}\sum_{k=0}^{N-1} coeff(k,l)\cdot e^{j2\pi\frac{(ik+lj)}{N}}$$

Properties:

Linearity:

$$s1(i,j) + s2(i,j) \Leftrightarrow coeff\,1(k,l) + coeff\,2(k,l)$$

$$a\cdot s(i,j) \Leftrightarrow a\cdot coeff(k,l)$$

Scaling:

$$s(a\cdot i, b\cdot j) \Leftrightarrow \frac{1}{|a\cdot b|} coeff\left(\frac{k}{a},\frac{l}{b}\right)$$

Translations:

$$s(i-ishift, j-jshift) \Leftrightarrow coeff(k,l)\cdot e^{-j2\pi\left(\frac{ishift\cdot k + jshift\cdot l}{N}\right)}$$

$$coeff(k-kshift, l-lshift) = s(i,j)\cdot e^{j2\pi\left(\frac{kshift\cdot i + lshift\cdot j}{N}\right)}$$

Rotation:

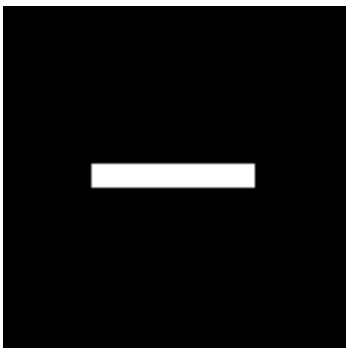
Rotation of s(i,j) over θ ⇔ rotation of coeff(k,l) over θ

Filtering:

$$f * s \Leftrightarrow coeff_f \cdot coeff_s$$

1.2 Exercise 4 - DFT

Create an image with the use of MATLAB commands that looks like this (128x128):



Look at the spectrum of the image with *surfc* and *imagesc* (use *fftshift* to get the low frequencies in the middle of the image). What do you notice about the energy present in the highfrequent components of the DFT image? Compare the horizontal and vertical components. What is the reason for the difference? Now

rotate the image over 45°, 60° and 90°(imrot). Compare the absolute value of the spectrum of the result with the absolute value of the original spectrum. What do you notice? What property of the DFT is hereby demonstrated?

1.3 Exercise 5

Read the grayscale images 'mandrill.tif' and 'zebra.tif'. Compute the DFT and then its modulus and phase values (use *abs* and *angle*) for each picture. Represent the log10 of the modulus of the DFT (use *log10* of 1+modulus, and *imshow(..., [])*). Do the same for the shifted variants (use *fftshift*).
Create two new DFTs by using the modulus of the DFT **of one image** with the phase information **of the other** (use *1i* for the imaginary unit). Then compute the inverse transform (use *ifft2*). Show the resulting pictures (use *round* on ifft coefficients, and *imshow(..., [])*) and comment on the results.