**IERG4190 / IEMS5707**

**Multimedia Coding and Processing**

**Week 6: Image Processing (II) – Coding practice**

**2021**

# Introduction on Matlab

Matlab is a numerical computing environment that is designed for easy implementation of matrix operations, data plotting and numerical analysis.

Matlab is a very simple language with very simple syntax. It is meant to allow users to convert mathematical operations into commands easily and explore data interactively.

Code examples are available at <https://github.com/zhoubolei/matlabexample>

Run command line:

git clone <https://github.com/zhoubolei/matlabexample>

More detailed tutorial:

**https://www.mathworks.com/help/matlab/getting-started-with-matlab.html**

When you start up matlab, you will see a console called "Command Window" in the center of the matlab window.

Graphical user interface, application

Description automatically generated

To start learning Matlab, all you need to do is to type commands into the Command Window first.

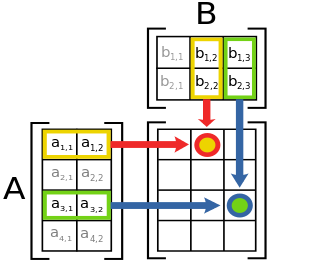
|  |
| --- |
| You can follow the instruction to install matlab: <https://www.itsc.cuhk.edu.hk/tc/all-it/procurement-support/campus-wide-software/matlab-and-simulink/>  It is also accessible in mainland. Please access the mainland site for mainland access. Please contact your TA for any technical assistance.  If you prefer an offline software, you can download Octave (<https://www.gnu.org/software/octave/>), an open source alternative of Matlab. The language is almost the same for our use. |

# Matlab Basics

The following table lists some basic commands for Matlab:

|  |  |
| --- | --- |
| Declare a variable / assign value to a variable | a = 100 |
| Querying the value of a variable | a |
| Perform a mathematical operation | a = a + 1 |
| Declare a m-by-n matrix | b = zeros(3,4) // create a matrix of all zeros  c = ones(3,4) // create a matrix of all ones  d = rand(3,4) // create a matrix of uniformly distributed random numbers  **Note the order of row and column** |
| Access a matrix element | d(1,2)  **In matlab index starts with 1, not 0!** |
| Access a row | e = d(1,:)  **Here ":" is an important symbol that denotes "all elements"** |
| Access a column | f = d(:,1) |
| Change the row/column of a matrix | d(2,:) = e  d(:,3) = f |
| Operating on a matrix | d = d \* 2  g = f \* e  **Try g = e \* f**  **What happens? Why?** |

Notice that there are many ways to perform the same operation in Matlab. The more advanced you are, the less commands you need. However, we do not require expertise in this course, so we will stick to "slow" and "simple" methods.

Just in case you forgot matrix multiplication, here is a helper diagram from Wikipedia:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SELF EXERCISE 1**   1. Create a matrix m with the following values:  |  |  |  | | --- | --- | --- | | 1 | 2 | 3 | | 1 | 2 | 3 | | 1 | 2 | 3 | | 1 | 2 | 3 |  1. Create an array v with the follow values:  |  |  |  |  | | --- | --- | --- | --- | | 5 | 5 | 5 | 5 |  1. What is the result of m \* v’? what happened here? |

# 

# Image as Matrix

Images can be read in as a matrix into Matlab. Once read into Matlab, we can perform our image processing operations.

To work with files, you need to set your working directory, or upload files onto the online drive. If you are using offline version of Matlab, in the left panel of your Matlab window, go to your home directory and right click on it (**click on exactly the words**) and select "Add to Path". If you are using the online version, use the **Upload** button in the top toolbar.

|  |  |
| --- | --- |
| Reading an image | I = imread('peppers.png')  I = im2double(I)  **Download the test images from Blackboard. Put them into your working directory/Upload to your online drive.**  **Type 'dir' in the Command Window to see if you see the file. It is a good practice to convert an image to double after loading it in - it will be in range of 0.0 to 1.0 and is good for further processing.** |
| Displaying an image | imshow(I)  **Once you have an imshow window, it will display your image upon further commands.** |
| Extracting different channels from the image | **Now use ‘peppers.png'.** This image is a m-by-n-by-k matrix, where k is the number of channels, which is usually 3 because we use **RGB model**.  To get the values for the red channel, for instance, we type  r = I(:,:,1)  To get the values of the green and blue channel, we type  g = I(:,:,2)  b = I(:,:,3) |
| Showing size of an image (or matrix) | size(I) |
| Combining images (or matrices) | J = cat(1,I,I)  **This means combine I and I in the first dimension.**  **Type imshow to see the result. To combine in another dimension, use** cat(2,....) |
| Cropping images (or matrices) | K(:,5,:) = []  **Here [] means empty matrix - we have set the 5th column to be empty, thus cropping it** |
| Saving image | imwrite(I, 'output.jpg') |

|  |
| --- |
| **SELF EXERCISE 2**   1. Load in the test image and modify the image such that the left 1/3 region is completely black; 2. For the same test image, modify the image such that the upper 1/3 region is completely black; 3. Concatenate the two images horizontally. |

# 

Programs and Flow Control

Advanced Matlab users usually minimize the amount of for-loops used in their computations. The many mathematical functions provided by Matlab are often powerful and highly optimized, and are usually preferred. However, in our course we will recommend you to stick to simple for loops unless you are experienced in matlab.

To create an executable script in matlab, you simply create a ".m" file in your working directory. The script can be invoked when you type the file name. For instance, if you create a "test.m" in your working directory, then you can run your commands inside the script "test.m" by typing "test" in your Command Window.

Notice that if a statement has ";" behind it in the script, then output to screen is suppressed, e.g.

a = 1;

Otherwise, if ";" is missing, the console will feedback with the output:

a = 1

That may be important in your debugging.

|  |  |
| --- | --- |
| For Loop | I = imread('peppers.png');  I = im2double(I);  for i=1:5  I(i,i,:) = 0;  end  imshow(I);  **Here, i will loop from 1 to 5, creating a diagonal line in the image. Note the added ";" at the end of each line, and the 'end' that denotes end of for loop.** |
| If statement | I = imread('peppers.png');  I = im2double(I);  for i=1:5  if I(i,i,1) < 0.5  I(i,i,:) = 1.0;  end  end  imshow(I); |
| elseif and else | I = imread('peppers.png');  I = im2double(I);  for i=1:5  if I(i,i,1) < 0.4  I(i,i,:) = 1.0;  elseif I(i,i,1) > 0.7  I(i,i,:) = 0.0;  else  I(i,i,:) = 0.5;  end  end  imshow(I); |
| Other looping statements | <http://www.mathworks.com/help/matlab/matlab_prog/loop-control-statements.html> |

Convolution

To use frequency domain methods we will need to use 2D convolution. Luckily this can be done easily in Matlab. We will now use the sample image **test.jpg**

|  |  |
| --- | --- |
| Creating a filter kernel | F = [1,1,1;1,1,1;1,1,1]  F = double(F)  **This is a convenient way to create a 3-by-3 matrix. Note we always change matrices to double in our case.** |
| Perform convolution | # read the image file  J = imread('**test.jpg**');  J(:,:,1) = conv2(J(:,:,1), F, 'same')  J(:,:,2) = conv2(J(:,:,2), F, 'same')  J(:,:,3) = conv2(J(:,:,3), F, 'same')  **Here we perform convolution on each channel of the image J. The parameter 'same' asks Matlab to produce a convolution result in the same size as the input.** |
| Normalization | **If you use imshow to display the output now the image is washed out. What should we do to get a proper output?** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SELF EXERCISE 3**   1. Please now use **test.jpg** 2. Create a script called 'test.m'. In the script, load in the image, convert it to double, create a filter and convert it to double as well, and apply the filter as follows  |  |  |  | | --- | --- | --- | | 5 | 0 | -3 | | 5 | 0 | -3 | | 5 | 0 | -3 |  1. Many image processing results are out of display range though. In the case here, you are recommended to shift the output by 15 and normalize by 30 before you show it. |

ffmpeg tutorial: <https://github.com/zhoubolei/matlabexample/blob/main/ffmpeg.md>

More advanced examples:

* 1. Image saliency (pdf: <https://github.com/zhoubolei/matlabexample/blob/main/cvpr2007.pdf> ): <https://github.com/zhoubolei/matlabexample/blob/main/example_image_saliency.m>
  2. Video saliency (pdf: <https://github.com/zhoubolei/matlabexample/blob/main/accv2010.pdf> ): <https://github.com/zhoubolei/matlabexample/blob/main/example_video_saliency.m>
  3. Hybrid image (pdf: <https://github.com/zhoubolei/matlabexample/blob/main/siggraph06.pdf> ): <https://github.com/zhoubolei/matlabexample/blob/main/example_hybrid_image.m>