University of Diyala
Computer Science Department
Image Processing
3rd Class
Lecturer: Dr. Jumana Waleed Salih

- -

Image Processing

معالجة صور

4 th lecture

Introduction to M-Function Programming

1 M-Files

M-files are created using a text editor and are stored with a name of the form `filename.m`, such as `average.m` and `filter.m`. The components of a function M-file are

- The function definition line
- The H1 line
- Help text
- The function body
- Comments

The *function definition line* has the form

```
function [outputs] = name(inputs)
```

The *H1 line* is the first text line. It is a single *comment* line that follows the function definition line.

The *function body* contains all the MATLAB code that performs computations and assigns values to output arguments.

2 Operators

Arithmetic Operators

Function	Description
`imadd`	Adds two images; or adds a constant to an image.
`imsubtract`	Subtracts two images; or subtracts a constant from an image.
`immultiply`	Multiplies two images, where the multiplication is carried out between pairs of corresponding image elements; or multiplies a constant times an image.
`imdivide`	Divides two images, where the division is carried out between pairs of corresponding image elements; or divides an image by a constant.
`imabsdiff`	Computes the absolute difference between two images.
`imcomplement`	Complements an image. See Section 3.2.1.
`imlincomb`	Computes a linear combination of two or more images. See Section 5.3.1 for an example.

■ Suppose that we want to write an M-function, call it fgprod, that multiplies two input images and outputs the product of the images, the maximum and minimum values of the product, and a normalized product image whose values are in the range [0, 1]. Using the text editor we write the desired function as follows:

```
function [p, pmax, pmin, pn] = improd(f, g)
%IMPROD Computes the product of two images.
%  [P, PMAX, PMIN, PN] = IMPROD(F, G)† outputs the element-by-
%  element product of two input images, F and G, the product
%  maximum and minimum values, and a normalized product array with
%  values in the range [0, 1]. The input images must be of the same
%  size.  They can be of class uint8, unit16, or double. The outputs
%  are of class double.

fd = double(f);
gd = double(g);
p = fd.*gd;
pmax = max(p(:));
pmin = min(p(:));
pn = mat2gray(p);
```

Note that the input images were converted to double using the function double instead of im2double because, if the inputs were of type uint8, im2double would convert them to the range [0, 1]. Presumably, we want p to contain the product of the original values. To obtain a normalized array, pn, in the range [0, 1] we used function mat2gray. Note also the use of single-colon indexing, as discussed in Section 2.8.

Suppose that f = [1 2; 3 4] and g = [1 2; 2 1]. Typing the preceding function at the prompt results in the following output:

```
>> [p, pmax, pmin, pn] = improd(f, g)
p =
    1    4
    6    4
pmax =
    6
pmin =
    1
```

3 Flow Control

Statement	Description
`if`	`if`, together with `else` and `elseif`, executes a group of statements based on a specified logical condition.
`for`	Executes a group of statements a fixed (specified) number of times.
`while`	Executes a group of statements an indefinite number of times, based on a specified logical condition.
`break`	Terminates execution of a `for` or `while` loop.
`continue`	Passes control to the next iteration of a `for` or `while` loop, skipping any remaining statements in the body of the loop.
`switch`	`switch`, together with `case` and `otherwise`, executes different groups of statements, depending on a specified value or string.
`return`	Causes execution to return to the invoking function.
`try...catch`	Changes flow control if an error is detected during execution.