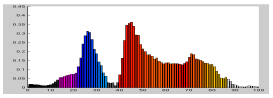
Dr. Qadri Hamarsheh



Digital Image Processing (750474)

Lecture 7

Outline of the Lecture

- Matlab Programming Examples
- Low-level processing (for loop structure)
- high-level processing (vectorization)
- Interactive I/O

Matlab Programming Examples

Example 7.1:

Write a Matlab function that computes the average intensity of an image.

The program should produce an error if the input is not a one-or two-dimensional array.

Hints:

1) 2-D array **f** can be converted to a column vector **v**, by letting

v= f (:)

(The function will be able to work with both vector and image inputs).

2) To return the size of the longest dimension of an array, we use

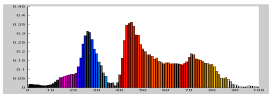
length (A).

3) Another way to obtain the **number of elements** in an array directly is to use function **numel**, with syntax:

n= numel (A)

Example 7.1 (Matlab Code)

```
function av = average (A)
% AVERAGE computes the average value of an array
% AV = AVERAGE (A) computes the Average value
% of input array, A, which must be a 1-D or
% 2-D array
% check the validity of the input
if ndims (A) >2
    error ('The dimensions of the input cannot exceed 2.')
end
% compute the Average
av = sum (A(:)) / length (A(:));
% or
% av = sum (A(:)) / numel (A);
% ndims function returns the number of dimensions.
```



Example 7.2:

Write an M-function (based on for loop) to extract a rectangular sub image from an image.

Solution:

- The inputs to the function:
  - ✓ *Original image*
  - ✓ *The size (number of rows and columns) of the sub image*
  - ✓ *Coordinates of the top left comer of the extracted image.*

Keep in mind
The image origin in Matlab is at (1,1)

Example 7.2 (Matlab Code)

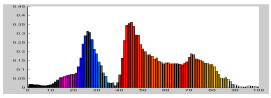
```
function s= subim (f, m, n, rx, cy)
% SUBIM Extract a sub image, s, from a
% given image f
% the subimage is of the size m-by-n. and
% the coordinates of its top, left corner are
% (rx, cy).
s = zeros (m, n); % preallocating matrix.
rowhigh = rx + m-1;
colhigh = cy + n-1;
xcount = 0;
for r = rx : rowhigh
    xcount = xcount +1;
    ycount = 0;
   for c= cy: colhigh
       ycount = ycount +1;
       s (xcount, ycount)= f (r, c);
   end
end
```

Example 7.3: color image planes:

The green and red color planes of image rgbimage.jpg are swapped

Example 7.3 (Matlab Code)

```
f = imread ('rgbimage.jpg');
red = f (:, :, 1); %red component
g (:, :, 1) = f (:, :, 2);
g (:, :, 2) = red ;
g (:, :, 3) = f (:, :, 3);
imshow (g);
```



Example 7.4: individual pixel processing:
The intensity of the red color channel of rgb image.jpg is divided by 2.
(Low-level processing)

Example 7.4 (Matlab Code)

```
f = imread ('football.jpg');
[M    N    D] = size (f);
g = uint8 (zeros (M, N, 3));
for x= 1:M
     for y= 1:N
          g (x, y, 1) = f (x, y, 1)/2;
          g (x, y, 2) = f (x, y, 2) ;
          g (x, y, 3) = f (x, y, 3);
    end ;
end ;
imshow (g);
```

Example 7.5
The intensity of the red color channel of rgb image.jbg is divided by 2 using high-level processing (array notation--- vectorization)

Example 7.5 (Matlab Code)

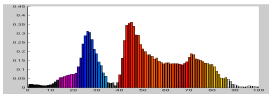
```
f = imread ('football.jpg');
g = f ;
g  (:, :, 1) = g (:, :, 1) /2 ;
imshow (g);
```

Comments for examples:

- Matlab is a programming language specifically designed for **array operations**; we can whenever possible to increase the computational speed (**code optimization**) using:

1) **Vectorizing :** simply means converting ***for*** and ***while*** loops to equivalent ***vector*** or ***matrix*** operations, the vectorized code (**example 7.5**) runs on the order of ***30 times*** faster than the implementation based on for loops(**example 7.4**).
2) **Preallocating Arrays**: to improve **code execution time**, preallocate the size of the array used in a program.

```
>> f = zeros (1024); % reduce memory fragmentation
>> g = zeros (1024);
>> img = unit8 (zeros (512, 1024))
% create a block image with
% width 1024 and height 512 of type unit8
>> img = unit8 (255 * ones (512, 1024)) % white image
>> img = double (ones (512, 1024)) % white image (double).
```



Example 7.6

Vectorization of program for extracting sub regions (example 7.2)

Example 7.6 (Matlab Code)

```
rowhigh = rx + m-1;
colhigh = cy + n-1;
s = f (rx: rowhigh, cy: colhigh);
%where f is the image from which the region is to be extracted.
```

Example 7.7

Convert the rgb color image football.jpg to a greyscale image:
(Using the method of computing the mean of the three color channels and then store it in file grayfootball.jpg ;

- Using low-level processing (for loop).
- Using high-level processing (vectorization).

Example 7.7 (Using low-level processing)

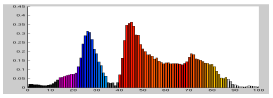
```
f = imread ('football.jpg');
[M  N  D] = size (f);
g = unit8 (zeros (M , N));
for x = 1:M
      for y = 1:N
           % g(x,y) = (f(x,y,1) + f(x,y,2) + f(x,y,3)) / 3 ;
           % The line above doesn't work.
           % overflow occurs, while processing unit8, because
          % the value range in the intermediate results
          % are limited to 255
            g (x, y) = (f (x,y,1) /3 + f(x,y,2)/3 + f(x,y,3)/3);
      end;
end;
imshow (g);
imwrite (g, 'grayfootball.jpg', Quality' , 100) ;
% The quality of the resulting image is set to 100 (no data loss;)
```

Example 7.7 (Using low-level processing)

```
f = imread ('football.jpg');
g = unit8 (mean (f, 3));
imshow (g)
imwrite (g, 'grayfootball.jpg' , 'Quality' , 100);
```

Home work1:

Write an M-function for **grayscale image blurring Method** (compute the mean of a **3*3** pixel environment and by setting the resulting center pixel to this value (mean)). Using the **vectorization** and **for** loop methods. The function also compares the **computational speed** time between two methods (use tic and toc Matlab function).



Interactive I/O

- To write **interactive M-functions** that display information and instructions to users and accept inputs from the keyboard
  - ✓ function **`disp`** is used to display information on the screen.

Syntax:

```
disp (argument).
```

Argument may be:

a) ***Array*** :

```
>> A = [1  2  ;  3  4];
>> disp (A);
```

b) ***variable contains string*** :

```
>> sc = 'Digital Image processing' ;
>> disp (sc)
```

c) ***string***:

```
>> disp ('This is another way to display text')
```

- ✓ function **`Input`**: is used for inputting data into an M-function.

Syntax:

I. Form 1

```
t = input ('message')
```

- This function outputs the words contained in message and waits for an input from the user, followed by a return, and stores the input in **`t.`**
- The input can be

a) ***Single number***.
b) ***Character string*** (single quotes)
c) ***Vector*** (enclosed by square brackets)
d) ***matrix*** (enclosed by square brackets) and rows separated by semicolons).

II. Form 2

```
t = input ('message','s')
```

- Outputs the contents of message and accepts a character string whose elements can be separated by commas.

- ✓ **`str2num (t)`** is used to ***convert*** strings into numbers of class double.