

CSE185

Introduction to Computer Vision

Lab 01: Image Processing in MATLAB

Instructor: Prof. Ming-Hsuan Yang

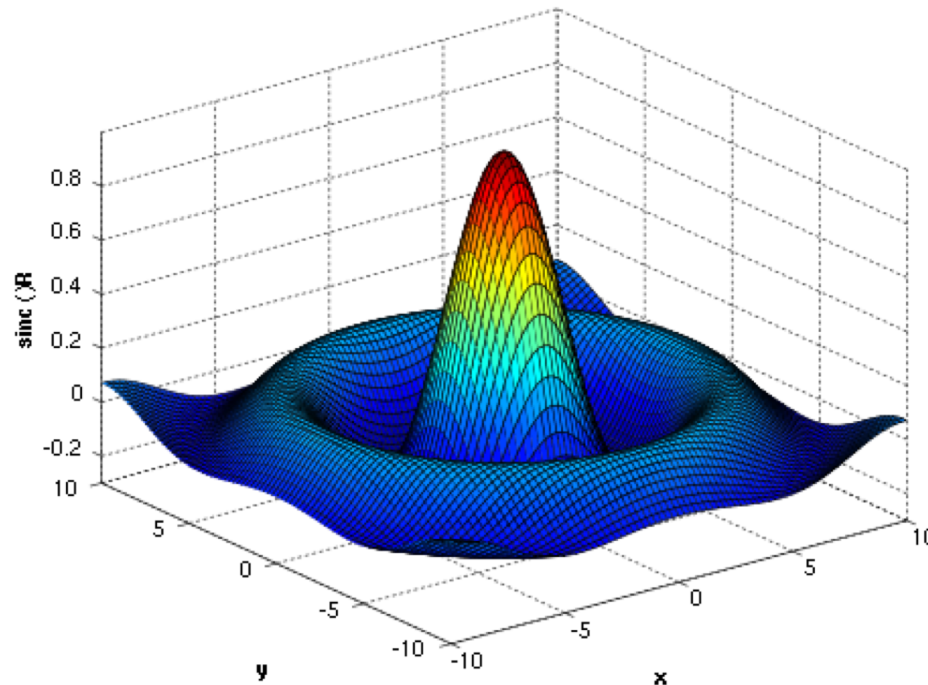
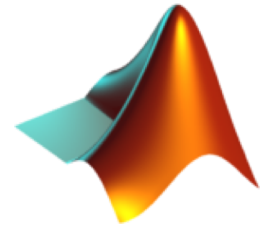
TA: Taihong Xiao & Tiantian Wang

Lecture and Labs

- Lecture:
 - Monday 15:00 – 16:15 @ COB2 130
 - Tuesday 15:00 – 16:15 @ COB2 130
- Labs:
 - 02L: Monday 16:30 – 19:20 @ COB1 281
 - 03L: Monday 19:30 – 22:20 @ COB1 281
 - 04L: Friday 10:30 – 13:20 @ SE1 100
 - 05L: Monday 07:30 – 10:20 @ COB1 281

Introduction to MATLAB

- MATLAB is a numerical computing environment
- Allow easy operation on matrix, image, N-D data
- Easily plot and visualize data
- Simple GUI
- Interface with other languages (e.g. C/C++, Python)



Variable

- Variable: no declaration, implicit type conversion

```
>> x = 10
x =
    10
>> x = 'test' (use single quote for string)
x =
    test
>> z = 10; y = z + 10
y =
    20
>> y = z^2
y =
    100
>> y = mod(z, 3)
y =
     1
```

Vector

- Vector: use `[]` or `init:step:end`

```
>> vec = [1, 100]
vec =
     1    100
>> vec = 1:2:10
vec =
     1     3     5     7     9
```

- Use `()` to access elements (index starts from 1):

```
>> vec(3)
ans = 5
```

- Access part of vector:

```
>> vec(2:4)
ans =
     3     5     7
```

Matrix

- Matrix: use semicolon to separate each row

```
>> A = [1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
```

- Access elements:

```
>> A(2, 3)
ans =
     6
```

- Access sub-matrix:

```
>> A(1:2, 2:3)
ans =
     2     3
     5     6
```

Multiplication

- Matrix-vector multiplication:

```
>> A = [1 2 3; 4 5 6; 7 8 9];  
>> x = [1; 1; 1];  
>> A * x
```

```
ans =  
     6  
    15  
    24
```

$$A * x = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 6 \\ 15 \\ 24 \end{pmatrix}$$

- Element-wise multiplication:

```
>> A = [1 2 3; 4 5 6; 7 8 9];  
>> B = [1 1 1; 2 2 2; 3 3 3];  
>> A .* B
```

```
ans =  
     1     2     3  
     8    10    12  
    21    24    27
```

$$A .* B = \begin{pmatrix} 1 \cdot 1 & 2 \cdot 1 & 3 \cdot 1 \\ 4 \cdot 2 & 5 \cdot 2 & 6 \cdot 2 \\ 7 \cdot 3 & 8 \cdot 3 & 9 \cdot 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 8 & 10 & 12 \\ 21 & 24 & 27 \end{pmatrix}$$

If statement

- If statement

```
if EXPRESSION
    ...
end
```

- If-else statement

```
if EXPRESSION
    ...
else
    ...
end
```

```
if EXPRESSION
    ...
elseif EXPRESSION
    ...
else
    ...
end
```


Loop

- For loop

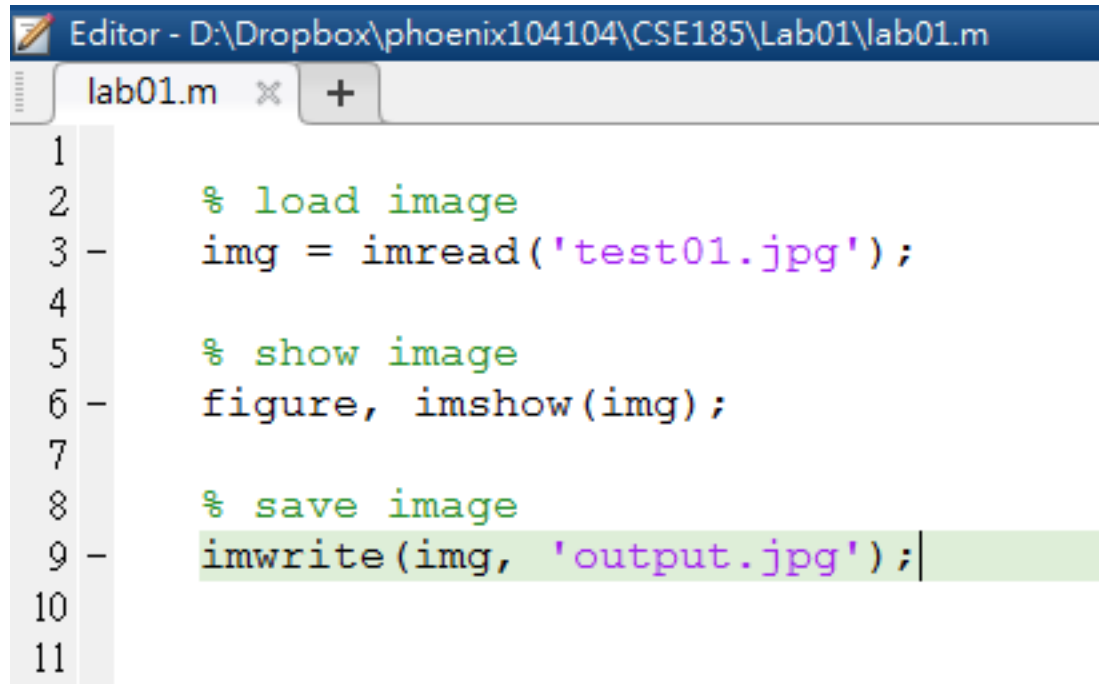
```
for i = 1:10  
    ...  
end
```

- While loop

```
while EXPRESSION  
    ...  
end
```

M file

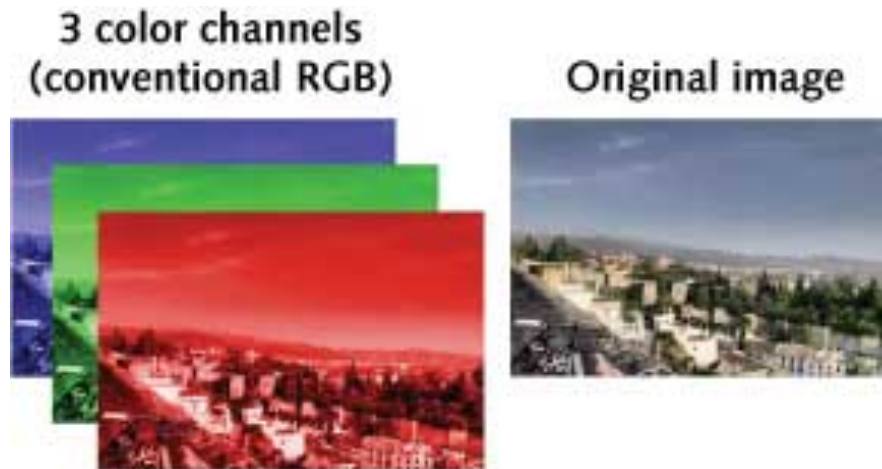
- MATLAB executable file/script, function: *.m file



```
Editor - D:\Dropbox\phoenix104104\CSE185\Lab01\lab01.m
lab01.m x +
1
2 % load image
3 - img = imread('test01.jpg');
4
5 % show image
6 - figure, imshow(img);
7
8 % save image
9 - imwrite(img, 'output.jpg');
10
11
```

Color Image

- Color image is a 3-D matrix in MATLAB: $Height \times Width \times 3$

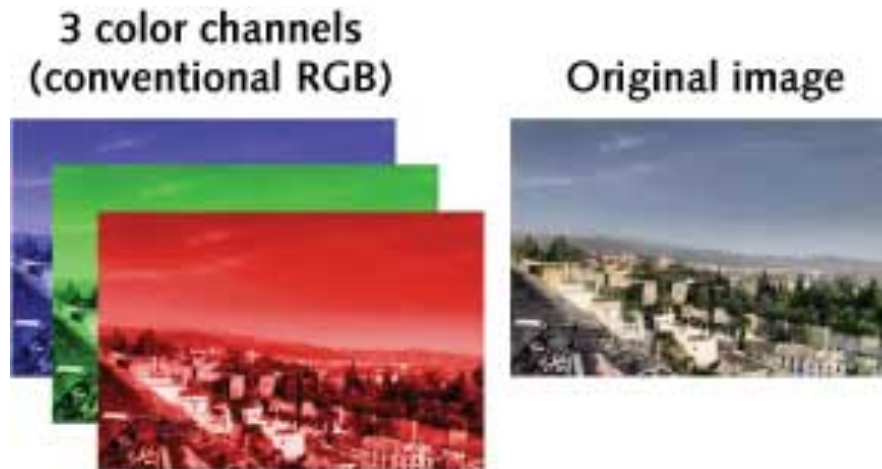


- Read image: `I = imread(filename);`
- Show image: `figure, imshow(I);`
- Save image: `imwrite(I, filename);`

Hint: type `help imread` in command window, or press F1 on function name to see the usage of the function

Color Image

- Color image is a 3-D matrix in MATLAB: $Height \times Width \times 3$



- $I(:, :, 1)$ is red channel
- $I(:, :, 2)$ is green channel
- $I(:, :, 3)$ is blue channel



: means select all elements in this dimension

Gray-scale Image

- Gray-scale image is a 2-D matrix in MATLAB: *Height* × *Width*



Gray-scale image



Color Image

- Use `size()` to check matrix dimension

```
>> I = imread('01_gray.jpg');  
>> size(I)  
ans =  
    300    400
```

Pixel Range and Type

- When loading image to MATLAB:
 - Default data type is uint8
 - Each pixel/element has a value between 0 and 255 (8 bits)
- Use `im2double()` to convert data type to double:
 - pixel range is between 0 and 1

```
>> I = imread('01.jpg');  
>> I(1, 1)  
ans =  
    34  
>> I = im2double(I); The same as I = double(I) / 255.0;  
>> I(1, 1)  
ans =  
    0.1333
```

Image Processing in MATLAB

- Set the value of green channel to zero



Image Processing in MATLAB

- Convert RGB to Y (gray-scale)

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B$$



- Do NOT use `rgb2gray()` function in MATLAB

Reference: RGB to YUV <https://en.wikipedia.org/wiki/YUV>

Image Processing in MATLAB

- Rotate image 90 degree: use `imrotate()`



Image Processing in MATLAB

- Crop image boundary: extract sub-matrix given the upper left and lower right positions



Image Processing in MATLAB

- Horizontally flip image: use `flip()`



Image Processing in MATLAB

- Combine 4 images into one image with 2 x 2 grid



Image Processing in MATLAB

- Combine 4 images into one image with 2 x 2 grid
- Hint: use `zeros(Height, Width, 3, 'uint8')` to create a canvas/matrix first, and consider each image as a sub-matrix of the canvas
- The size of our testing image is $300 \times 400 \times 3$, use 10 pixels for separations:

```
I1 = imread('01.jpg');  
canvas = zeros(300 * 2 + 10, 400 * 2 + 10, 3, 'uint8');  
canvas(1:300, 1:400, :) = I1;
```

Image Processing in MATLAB

- Use (:) to convert image/matrix to vector
 - matrix in MATLAB is column-major

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad A(:) = \begin{pmatrix} 1 \\ 4 \\ 7 \\ 2 \\ \vdots \\ 3 \\ \vdots \\ 9 \end{pmatrix}$$

- Use reshape () to convert vector to image/matrix

```
>> I = imread('01.jpg'); ⇐ The shape of image is 300×400×3
>> I = I(:);
>> size(I)
ans =
    360000         1
>> I = reshape(I, 300, 400, 3);
>> size(I)
ans =
    300    400     3
```


Lab 01 Assignment

1. Set green channel to zero, and save as **green.jpg**
2. Convert image from RGB to gray scale without using built-in functions, and save as **gray.jpg**
3. Rotate 90 degree, and save as **rotate.jpg**
4. Crop the image given the upper left point (30, 100) and lower right point (270, 300), and save as **crop.jpg**
5. Horizontally flip the image, and save as **flip.jpg**
6. Combine 4 images (01.jpg ~ 04.jpg) into one matrix with 2 x 2 grid and 10 pixels separations, and save as **combine.jpg**
7. Convert images (05.jpg and 06.jpg) to vectors, average them, and save as **average.jpg**
8. Upload all output images and your **lab01.m**

* For problem 1 – 5, choose 01.jpg as the input only.

Reference

- MATLAB: <http://www.mathworks.com/products/matlab/>
- Octave: <https://www.gnu.org/software/octave/>
- Introduction to MATLAB with Image Processing
http://www.slideshare.net/Sutanshu_Raj/introduction-to-matlab-with-image-processing-5495912