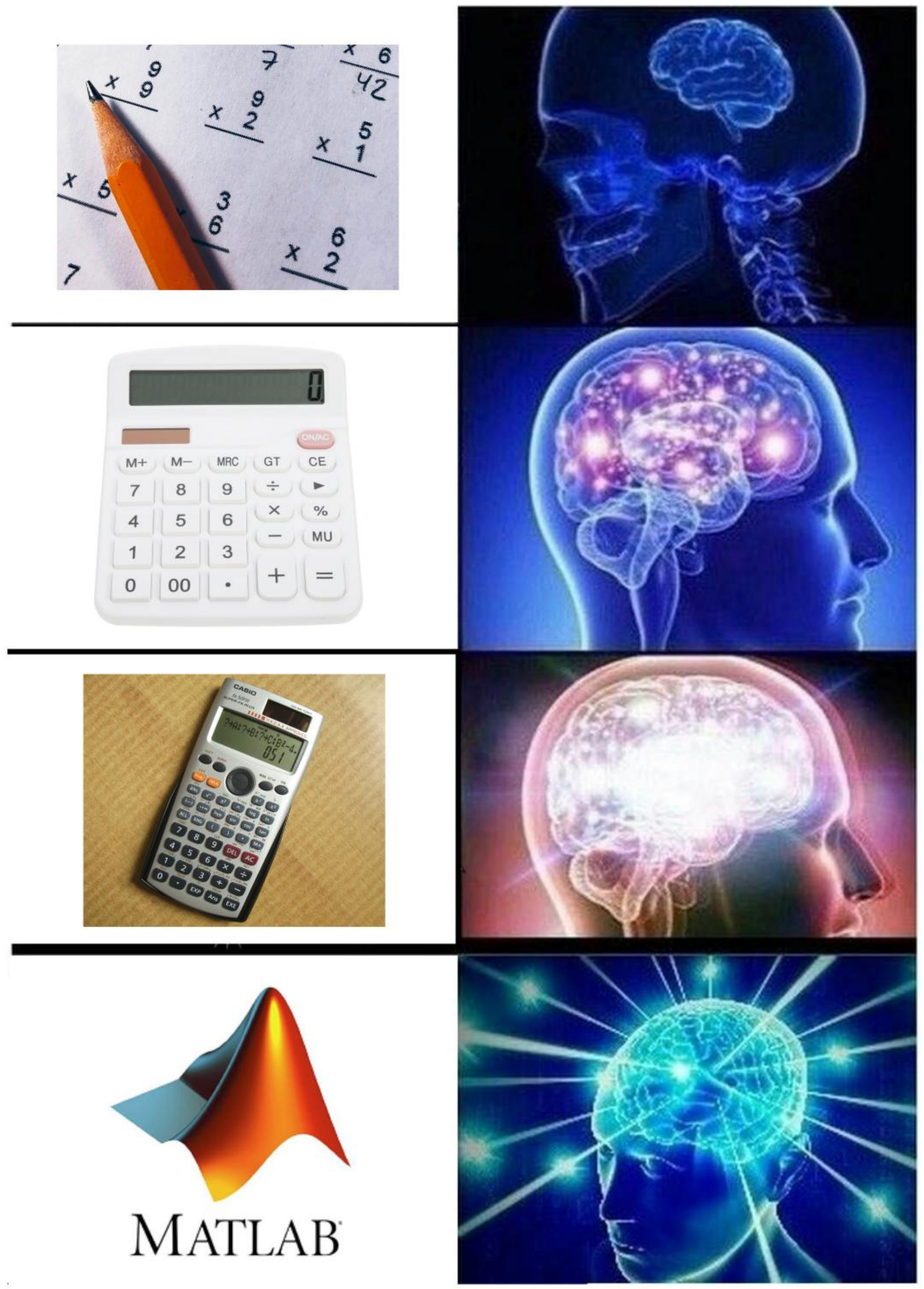
MATH2221 Mathematics Laboratory II

Lecture 10: Image and Video Processing Using MATLAB

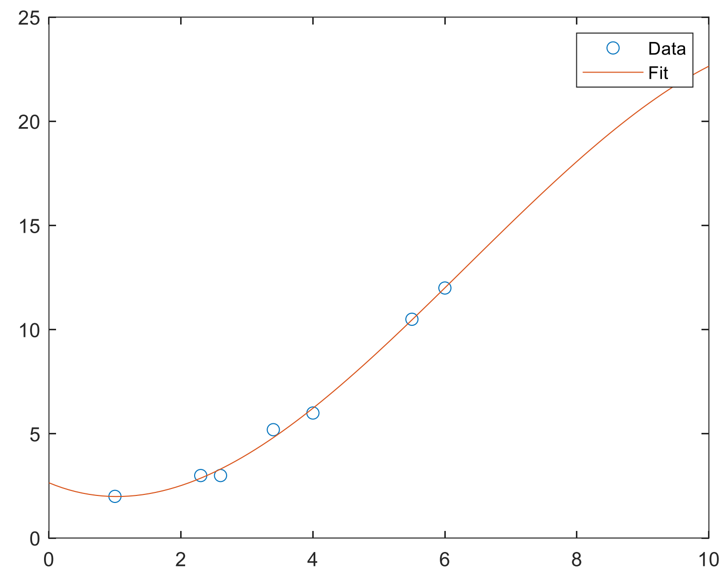
Gary Choi

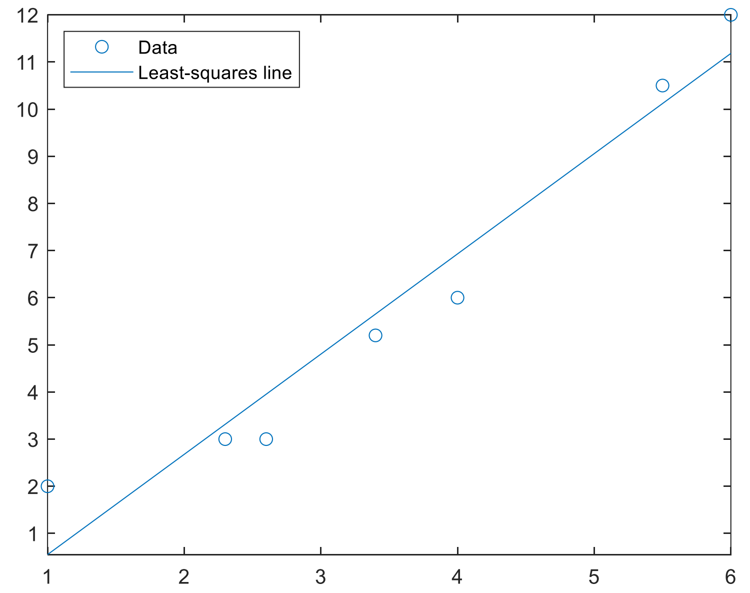
March 25, 2025



Recall: Data analysis using MATLAB

- Polynomial fitting:
  - p = polyfit(x,y,n)
  - yi = polyval(p, xi)
- Visualization of least-squares lines: lsline
- Other model fitting/multiple regression:
  - Formulate a matrix equation using the data points
  - Then solve for the best-fit coefficients using A\b

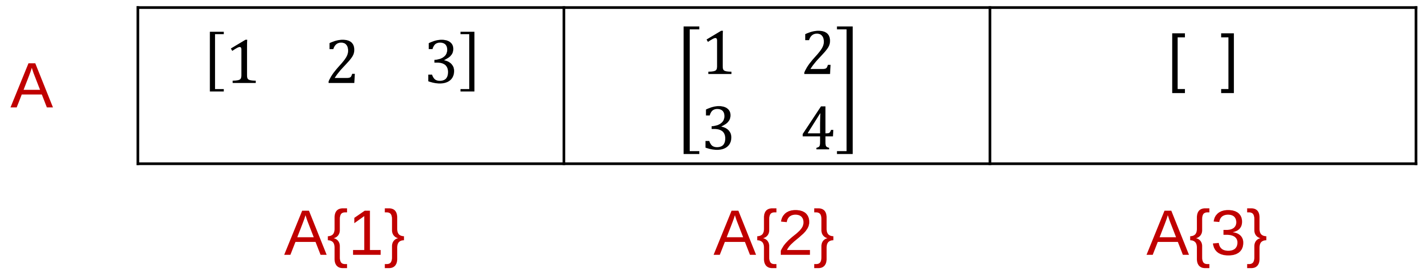




Recall: Advanced data structure in MATLAB

- **Cell array**

```
A = cell(1,3);
A{1} = [1,2,3];
A{2} = [1,2; 3,4];
```


A
$\begin{bmatrix}1 & 2 & 3\end{bmatrix}$
$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$
[ ]
A{1}
A{2}
A{3}

- **Structure array**

we use the "." operator: arrayname.field = value

```
>> student = struct; % create a new structure array
>> student.firstname = 'TaiMan';
>> student.lastname = 'Chan';
>> student.id = 123456;
```

Recall: Data input/output in MATLAB

- The MATLAB-specific .mat format
  - save('filename.mat'), save('filename.mat', 'a', 'b', 'c’)
  - load('filename.mat'), load('filename.mat', 'a', 'b', 'c’)

- Saving/loading with other file formats
  - Excel file (.xls or .xlsx)
    - xlsread, xlswrite
  - CSV file (.csv)
    - csvread, csvwrite
  - More general file formats (.txt, .dat, .csv, .xls, .xlsx)
    - readmatrix, writematrix, readcell, writecell

- Even more flexible input/output
  - fopen, fclose, textscan, fprintf

Changing the display format of data in MATLAB

- Use format style to change the display format of numeric values in MATLAB

Style	Result	Example
short (default)	Short, fixed-decimal format with 4 digits after the decimal point	3.1416
long	Long, fixed-decimal format with 15 digits after the decimal point for double values, and 7 digits after the decimal point for single values.	3.141592653589793
shortE	Short scientific notation with 4 digits after the decimal point.	3.1416e+00
longE	Long scientific notation with 15 digits after the decimal point for double values, and 7 digits after the decimal point for single values.	3.141592653589793e+00

- More options: https://www.mathworks.com/help/matlab/ref/format.html

- Examples:

```
>> format short
>> a = exp(1)
a =
   2.7183
```

```
>> format long
>> a
a =
   2.718281828459045
```

Changing the display format of data in MATLAB

- Simple display method: directly typing the variable a or disp(a)
- To control the display more precisely: use sprintf(format1,a1,format2,a2,...)

Format '%m.nL'	Description
%mi	Integer. Blanks added to the left, total number of characters is m.
%m.nf	Fixed-point notation, with n decimal places. Blanks added to the left, total number of space is m.
%m.ne	Like f with a signed 3-digit exponent notation (e.g. e+000). Total character count m includes five for the exponent.
%m.ng	Choose f or e, whichever is shorter
%ms	Print a string array or character vector. Blanks added to the left so that total number of characters is at least m.

Example Command	Result
sprintf('%1i',round(pi))	3 (1 character)
sprintf('%3i',round(pi))	_ _ 3 (_ is a space)
sprintf('%5.3f',pi)	3.142
sprintf('%8.3f',pi)	_ _ _ 3.142
sprintf('%12.3e',pi)	_ _ 3.142+000
sprintf('%6s','hello')	_hello
sprintf('%6s','hello123')	hello123

Reading/writing a file with specific data format

- One can also control the input/output data format using fprintf and fscanf

- fid = fopen('file',permission)
  - Open file and return the file identifier fid.
  - permission specifier can be: 'r' read, 'w' write, etc.

- fprintf(fid,format,a1,...)
  - Write variables a1,... according to format to the file identified by fid.
  - The format commands are the same as the ones for sprintf

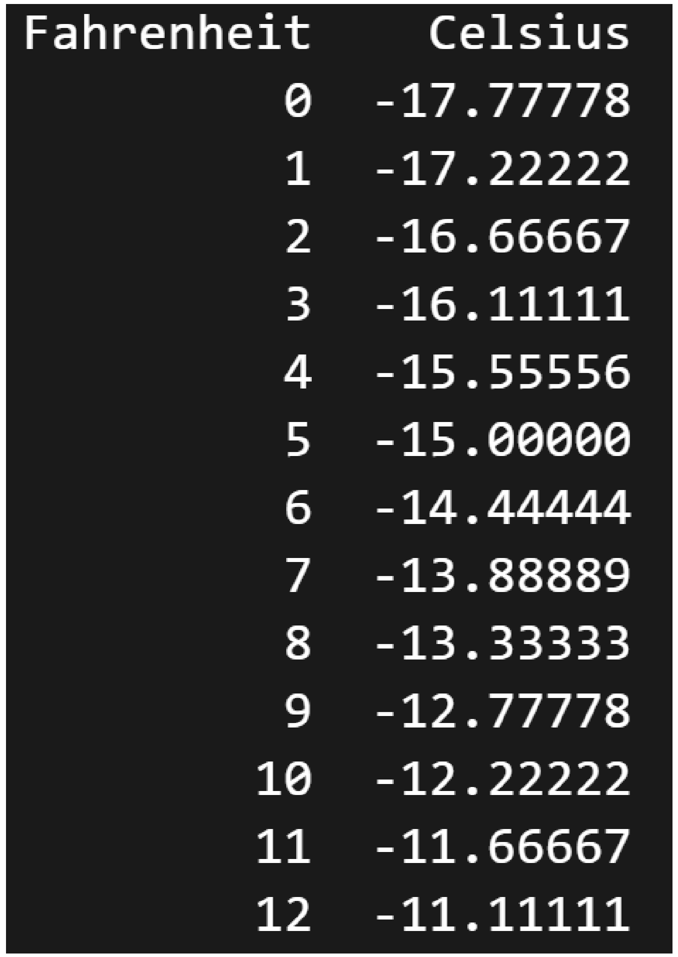
- fscanf(fid,format)
  - Read data using format from the file identified by fid.

- fclose(fid)
  - Close the file with identifier fid.

Reading/writing a file with specific data format

- Example:
  Writing a txt file with the conversion between Fahrenheit and Celsius.

```
% open the file table.txt
datafile=fopen('table.txt','w');

% print heading. \n means a new line
fprintf(datafile,' Fahrenheit    Celsius\n');

for i=0:100
  fprintf(datafile,'%11i',i);
  fprintf(datafile,'%11.5f\n',(i-32)*5/9);
end

% close the file table.txt
fclose(datafile);
```



```
 Fahrenheit    Celsius
          0  -17.77778
          1  -17.22222
          2  -16.66667
          3  -16.11111
          4  -15.55556
          5  -15.00000
          6  -14.44444
          7  -13.88889
          8  -13.33333
          9  -12.77778
         10  -12.22222
         11  -11.66667
         12  -11.11111
```

Path and file management

- By default, MATLAB can only access the files (.m or other data files) in the “current directory”
  - cd: Display the current directory
  - cd('newfolderpath') :Change the current directory to another folder
    - cd('..'): Go up one level
    - cd('some_subfolder_name'): Go to the subfolder (relative path)
    - cd('some_full_path') (e.g. C:\Users\...\mydata): Go to the specific folder (absolute path)

- Show all files in the current directory
  - dir: Display all files in current directory
  - dir *.txt: Display all files with .txt extension (can change to other formats)
  - dir Lec*.pdf: Display all .pdf files starting with Lec.

Path and file management

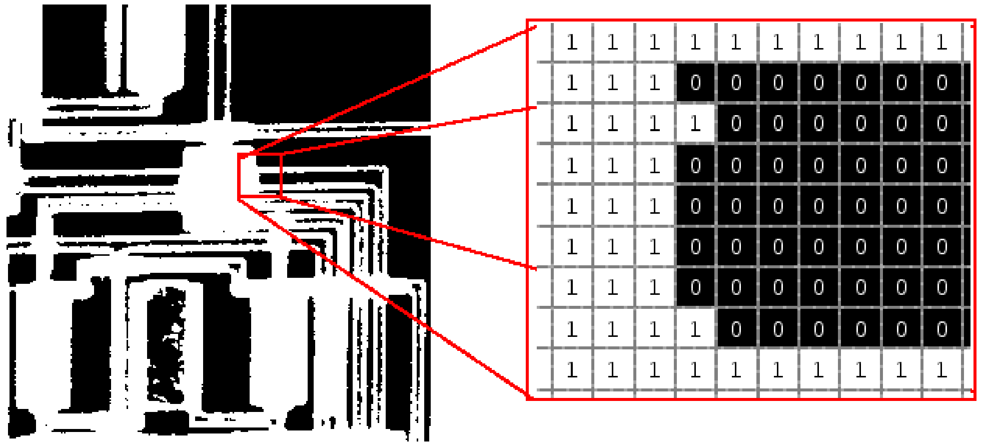
- Search path: a list of directories in which MATLAB looks for .m files and data
  - Include the current directory and some default MATLAB toolbox folders
  - Can examine the list using path

- To let MATLAB access more files (without changing your current directory or moving the files into your current directory), use:
  - addpath('directory'): Add directory to the current set of search paths
  - rmpath('directory'): Remove directory from the current set of search paths

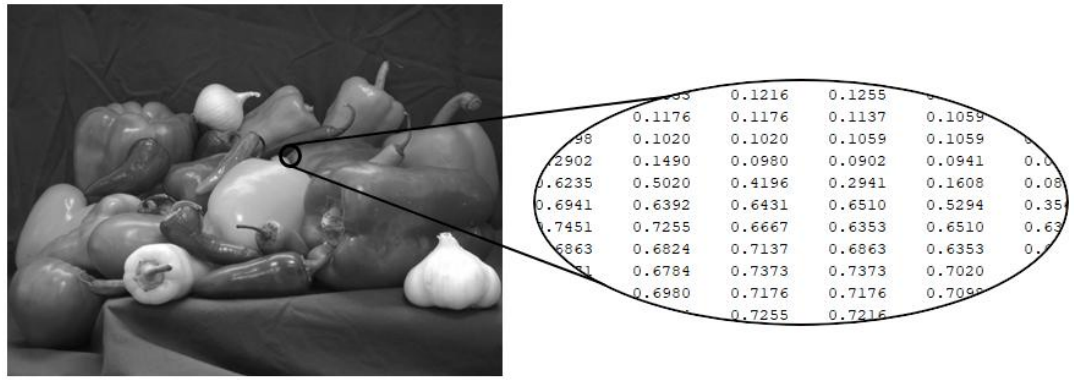
- Remarks:
  - Can keep different code and data files in different folders for better organization
  - Use addpath/rmpath to include/exclude them when needed
  - Recommended to avoid ambiguity/repetition in the file names

Next topic: Images

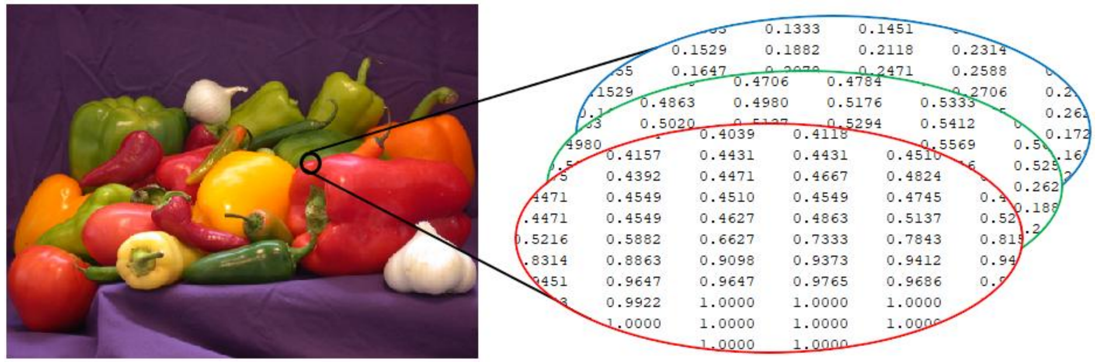
- Images are represented by matrices
- Binary (black and white) image
  - $m \times n$
  - All entries are 0 or 1
- Grayscale (intensity) image
  - $m \times n$
  - All entries are in [0, 1] (double) or [0, 255] (uint8, 8-bit unsigned integers)
- RGB (truecolor) image
  - $m \times n \times 3$
  - All entries are in [0, 1] (double) or [0, 255] (uint8, 8-bit unsigned integers)



1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1



	0.1216	0.1255	
0.1176	0.1176	0.1137	0.1059
0.1020	0.1020	0.1059	0.1059
0.1490	0.0980	0.0902	0.0941
0.5020	0.4196	0.2941	0.1608
0.6392	0.6431	0.6510	0.5294
0.7255	0.6667	0.6353	0.6510
0.6824	0.7137	0.6863	0.6353
0.6784	0.7373	0.7373	0.7020
0.6980	0.7176	0.7176	
	0.7255	0.7216	



How to process images in MATLAB?

- Loading an image into MATLAB: A = imread(filename)
- Displaying the image: imshow(A)
- Example:

```
>> A = imread('cuhk.jpg');
% A: 1080 × 1920 × 3 uint8
% Too large, “Cannot display summaries of variables with more than 524288 elements” in the workspace
>> figure; imshow(A);
```

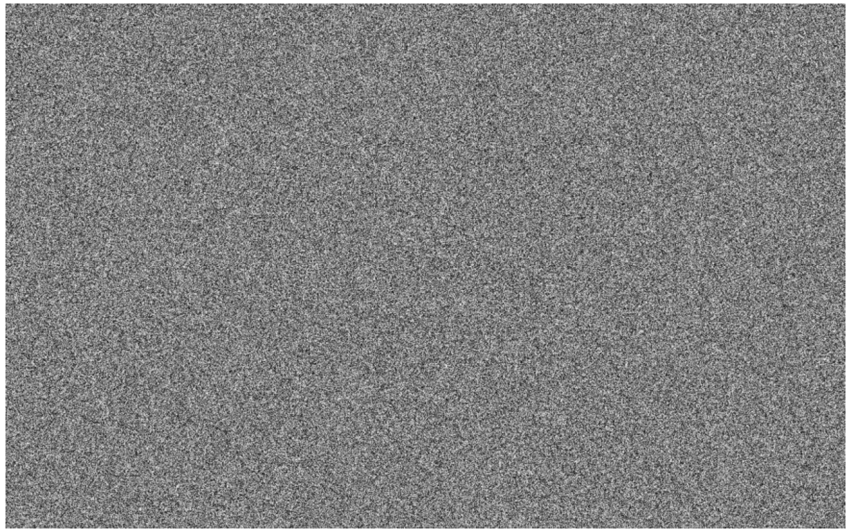
- Some built-in test images:
  - cameraman.tif, coins.png, peppers.png, rice.png, ...

https://60.cuhk.edu.hk/download/

How to process images in MATLAB?

- Writing an image from a matrix: imwrite(A,filename)

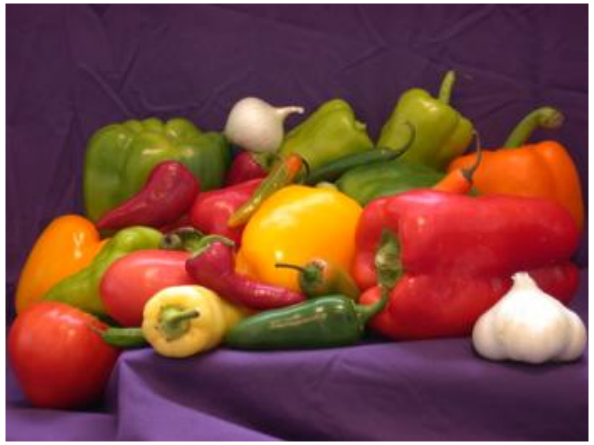
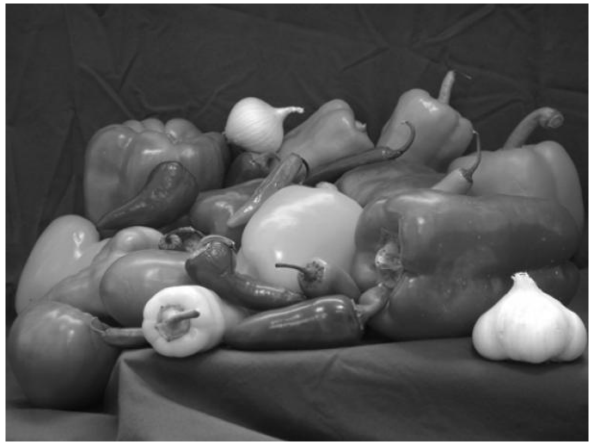
- Example: Create a grayscale image with width 1280 and height 800.
  A = rand(800,1280);
  imwrite(A,"myGray.png");



- Example: Create a RGB image with width 100 and height 100.
  B = rand(100,100,3);
  imwrite(B,"myRGB.png");

Image type conversion

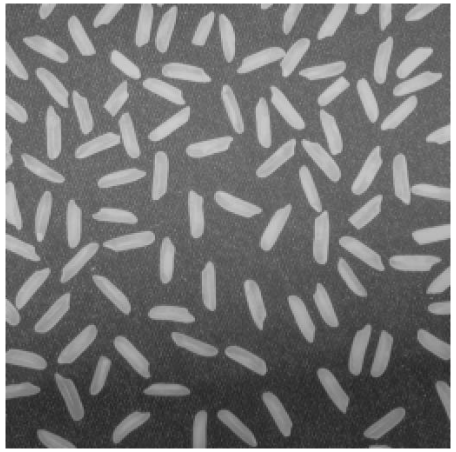
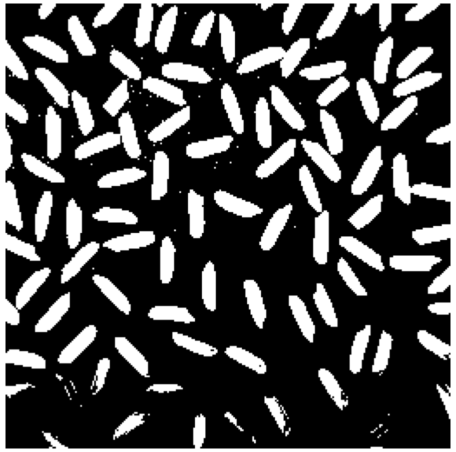
- Converting RGB image to grayscale:
  - I = rgb2gray(RGB)

Example:

```
RGB = imread("peppers.png");
figure; imshow(RGB)
I = rgb2gray(RGB);
figure; imshow(I)
```

- Converting grayscale image to a binary image:
  - BW = imbinarize(I)

Example:

```
I = imread('rice.png');
figure; imshow(I);
BW = imbinarize(I);
figure; imshow(BW);
```

See also: https://www.mathworks.com/help/images/import-export-and-conversion.html

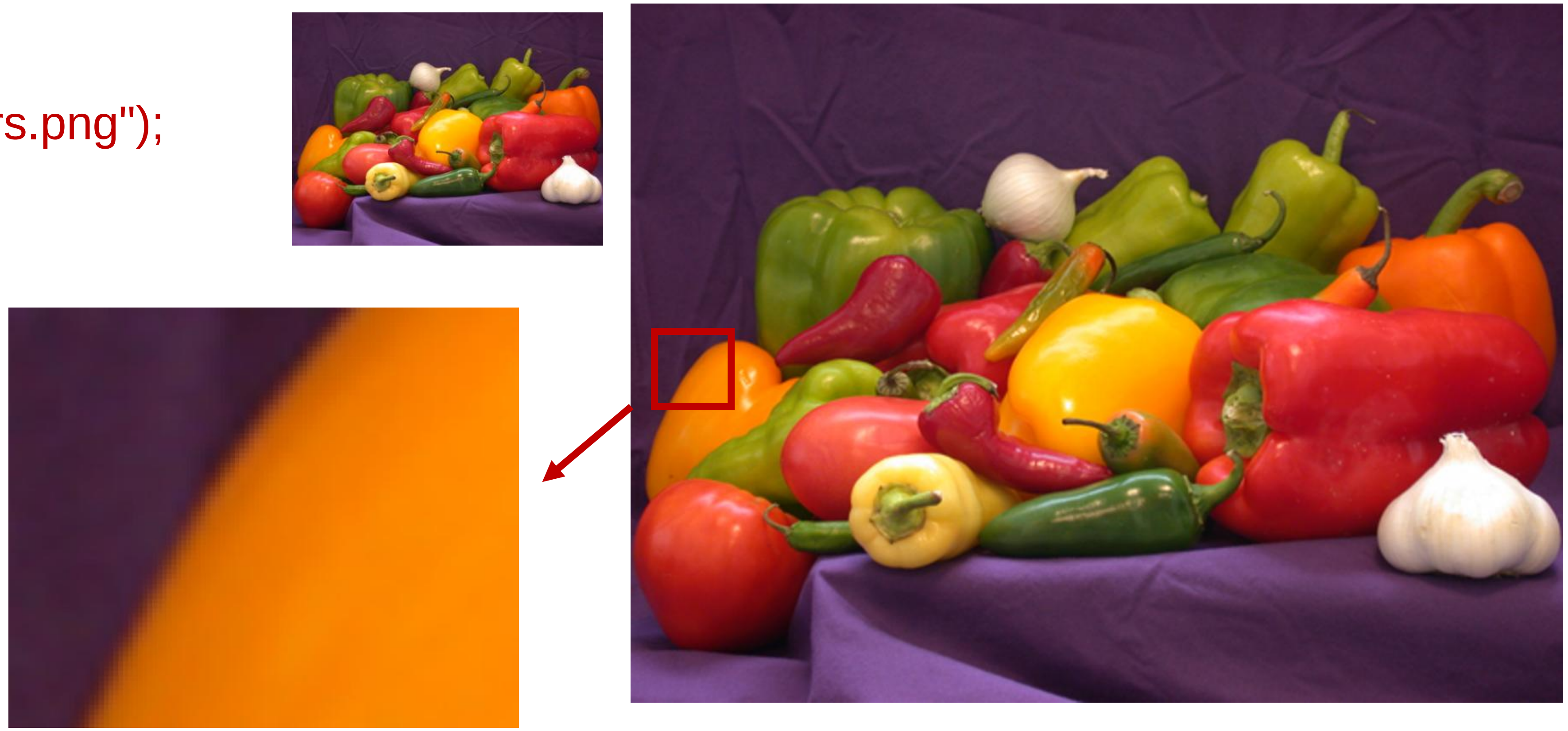
Resizing an image

- imresize: Resize a BW/grayscale/RGB image by interpolation
  - B = imresize(A,scale): specify the scaling factor
  - B = imresize(A,[numrows, numcols]): specify the target number of rows and columns
  - B = imresize(A,..., 'Method',method): specify the interpolation method as "bicubic" (default), "bilinear", "nearest" etc.

- Example:

```
RGB = imread("peppers.png");
figure; imshow(RGB)
```

```
I = imresize(RGB,3);
figure; imshow(I)
```



Cropping an image

- imcrop: Crop an image to keep a specific region only
  - [J,rect] = imcrop(I): an interactive Crop Image tool
  - J = imcrop(I,rect): crops the image according to the position and dimensions specified in the crop rectangle rect of the form [xmin, ymin, width, height] .
- Example:

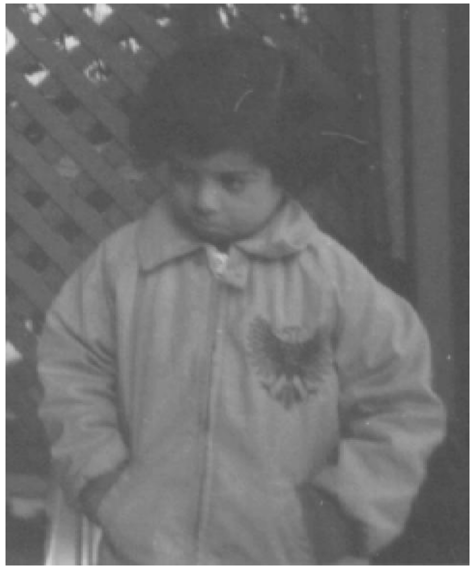
```
I = imread('coins.png');
[J, rect] = imcrop(I);
% use the mouse cursor to select a rectangular region
% then right click to save the cropped image as J
figure;
imshow(J);
```

Adjusting the image intensity values

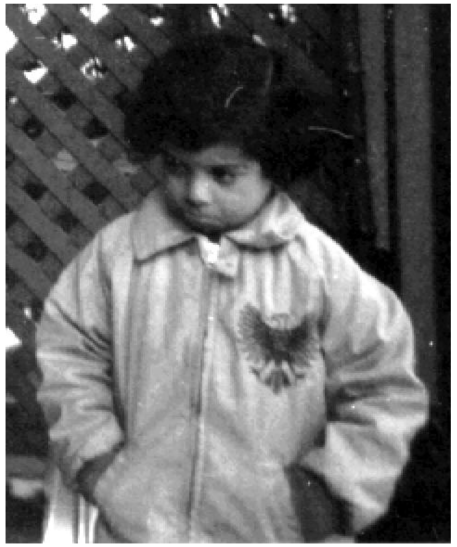
- imadjust: mapping the intensity values in grayscale image I to new values to increase the image contrast
  - J = imadjust(I):
    - saturate the bottom 1% and the top 1% of all pixel values
    - rescale everything to [0, 1] linearly
  - J = imadjust(I,[low_in, high_in]):
    - rescale values between low_in and high_in to [0, 1] linearly
  - J = imadjust(I,[low_in, high_in],[low_out, high_out]):
    - rescale values between low_in and high_in to [low_out, high_out] linearly

- Example:

```
I = imread('pout.tif');
figure; imshow(I);
```



```
J = imadjust(I);
figure; imshow(J);
```



Adjusting the image intensity values

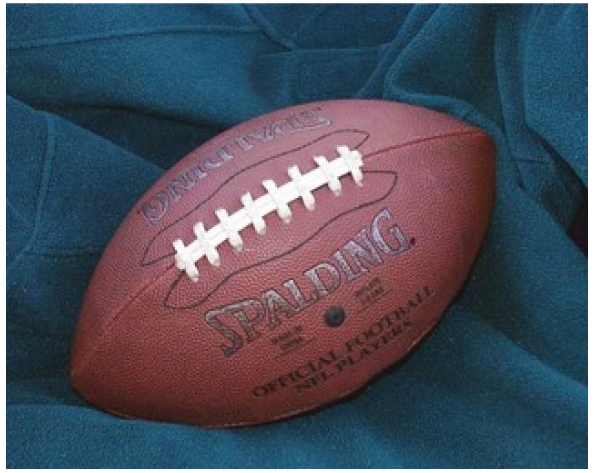
- Also works for RGB image but need to specify low_in, high_in for each channel

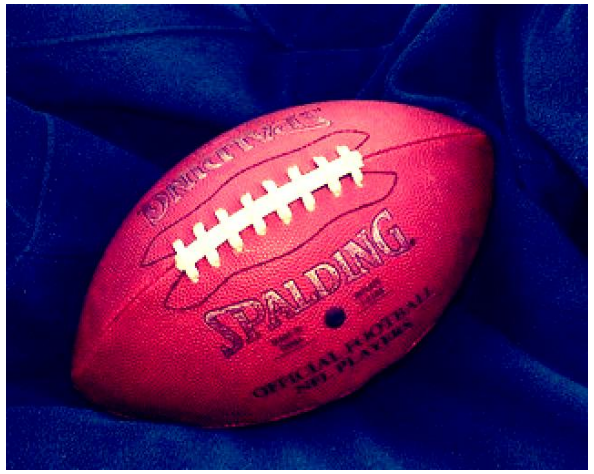
  - J = imadjust(RGB,[low_in_r, low_in_g, low_in_b; high_in_r, high_in_g, high_in_b]):
    - rescale values between low_in_r and high_in_r to [0, 1] linearly
    - Similar for g and b

  - J = imadjust(I,[low_in_r, …, high_in_b],[low_out_r, …, high_out_b]):
    - rescale values between low_in_r and high_in_r to [low_out_r, high_out_r] linearly
    - Similar for g and b

- Example:

```
RGB = imread('football.jpg');
figure; imshow(RGB);

RGB2 = imadjust(RGB,[.2 .3 0; .6 .7 1]);
figure; imshow(RGB2);
```





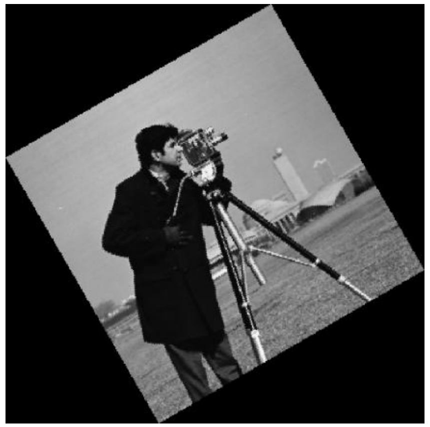
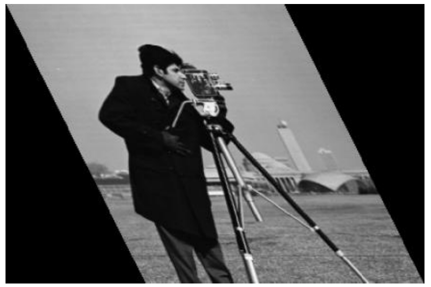
Applying a geometric transformation to an image

- imrotate: rotate an image
  - J = imrotate(I,angle): rotates image I by angle degrees in a counterclockwise direction around its center point.

- imwarp: more general transformations
  - J = imwarp(I,tform): transforms I according to the geometric transformation tform.
  - Here, tform can be created using affine2d(A') (for MATLAB version R2022a or before; note that we need the transpose here for consistency with the newer affinetform2d function, which has a slightly different operation convention) or affinetform2d(A) (for R2022b or newer), where A is a transformation matrix.

- Example:

```
I = imread('cameraman.tif');
figure; imshow(I);
J = imrotate(I, 30);
figure; imshow(J);
A = [1 0.5 0; 0 1 0; 0 0 1];
tform = affine2d(A');
K = imwarp(I,tform);
figure; imshow(K);
```

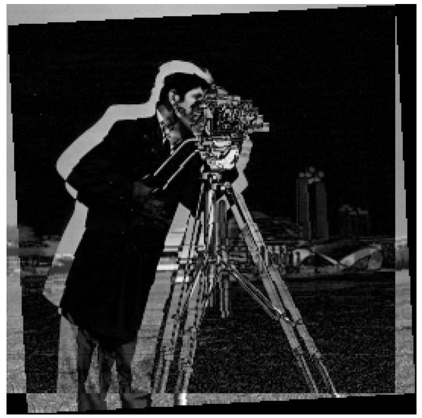
See also: https://www.mathworks.com/help/images/image-registration-and-geometric-transformation.html

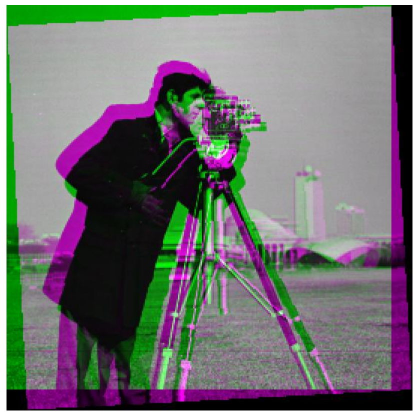
Visualization of image pair

- imshowpair: Comparing two images
  - C = imshowpair(A,B,method): creates a composite RGB image C using image A and B for visualization
  - Possible visualization methods include:
    - "diff": Display a difference image of A and B (computed in grayscale).
    - "falsecolor": Overlay A and B so that the gray regions indicate where the images have the same intensity, while colored regions show where the intensities differ.
    - "montage": Place A and B next to each other in the same figure.
    - and other options

- Example:

```
I = imread('cameraman.tif');
J = imrotate(I, 3);
figure; imshowpair(I,J,"diff");
figure; imshowpair(I,J,"falsecolor");
figure; imshowpair(I,J,"montage");
```





Video processing using MATLAB

- Loading a video file into MATLAB:
  - v = VideoReader(filename);
  - The object v contains many properties of the video, e.g.
    - v.Width
    - v.Height
    - v.NumFrames
    - v.FrameRate
    - v.Duration

- Reading a specific frame of the video as an image:
  - framei = read(v,i);

- Example:

```
% xylophone.mp4 is a built-in sample video
v = VideoReader("xylophone.mp4");
frame = read(v,100); % read frame 100
figure; imshow(frame);
```

Video processing using MATLAB

- Alternatively, we can use hasFrame to check whether there are any remaining video frame, and use readFrame to read the next the next available video frame
  - hasFrame(v) returns logical 1 (true) if there is a video frame available to read from the file associated with v. Otherwise, it returns logical 0 (false).
  - readFrame(v) reads the next available video frame.

- Once the image is loaded, we can apply the image processing functions

- Example:

```
v = VideoReader("xylophone.mp4");
while hasFrame(v)
    frame = readFrame(v);
    … % your operations
end
```

Writing a video from images

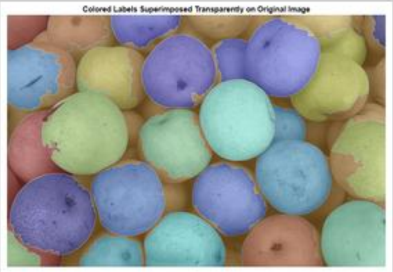
- First create a video file:
  - v = VideoWriter(filename,profile)
  - profile: the specific format (e.g. 'MPEG-4', 'Uncompressed AVI' etc.)

- Then use writeVideo to write different images to the video.

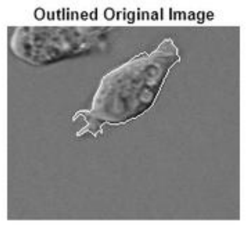
```
open(v); % open the object for writing

% write the images one by one
writeVideo(v,A1);
writeVideo(v,A2);
...
writeVideo(v,Ak);

close(v); % close the object
```

More image/video processing functions in MATLAB

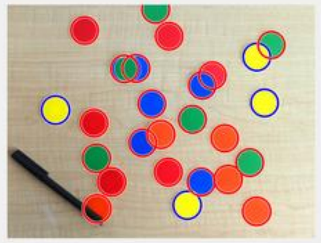
- Check out the image processing toolbox and computer vision toolbox!



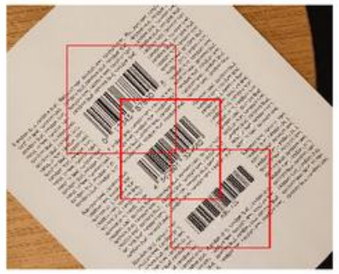
Marker-Controlled Watershed Segmentation
Separate touching objects in an image by using watershed segmentation.



Detect Cell Using Edge Detection and Morphology
Detect an object against the background using edge detection and basic morphology.



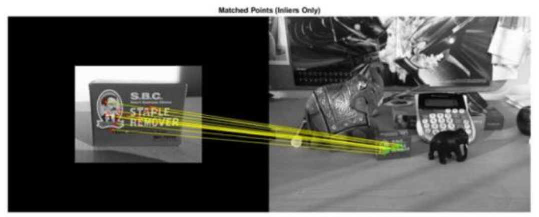
Detect and Measure Circular Objects in an Image
Automatically detect circular objects in an image and visualize the detected circles.



Read Barcodes In Image
Detect, decode, and localize 1-D and 2-D barcodes in an image. (Computer Vision Toolbox)

Segment and Read Text in Image
Automatically detect and recognize text in images using MSER and OCR. (Computer Vision Toolbox)

Find Object in Cluttered Scene Using Image Point Features
Detect a particular object in a cluttered scene, given a reference... (Computer Vision Toolbox)

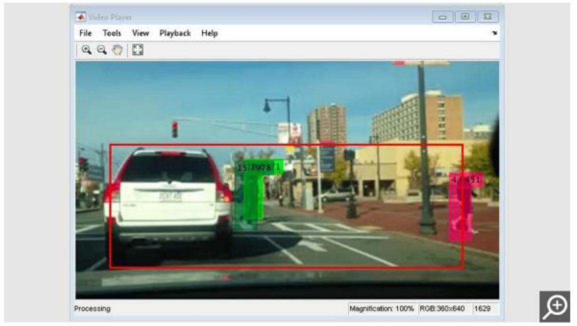


Feature Detection, Extraction, and Matching

Detect, extract, and match features such as blobs, edges, and corners, across multiple images. Use the matched features for registration, object classification, or in complex workflows such as SLAM.

Object Detection in a Cluttered Scene Using Point Feature Matching

Documentation | Examples



Multi-Object Tracking and Motion Estimation

Estimate motion and track multiple objects in video and image sequences.

Multi-Object Tracking with DeepSORT

Documentation | Examples

Reminder: Lab 8 this week

January

Sun	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	🟫 7	8	9	10	11
12	13	🟫 14	15	🟨 16	17	18
19	20	🟫 21	22	🟨 23	24	25
26	27	[28]	[29]	[30]	[31]	

February

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						[1]
[2]	[3]	🟫 4	5	🟨 6	7	8
9	10	🟫 11	12	🟨 13	14	15
16	17	🟫 18	19	🟨 20	21	22
23	24	🟫 25	26	🟩 27	28	1

March

Sun	Mon	Tue	Wed	Thu	Fri	Sat
2	[3]	[4]	[5]	[6]	[7]	[8]
9	10	🟫 11	12	🟨 13	14	15
16	17	🟫 18	19	🟨 20	21	22
23	24	🟫 25	26	🟨 27	28	29
30	31					

April

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		🟫 1	2	🟨 3	4	5
6	7	🟫 8	9	🟨 10	11	12
13	14	🟫 15	16	🟩 17		

🟫 **Lecture 1- Lecture 13**

🟨 **Lab 1 - Lab 10 (40%)**

🟩 **Test 1 (30%)**
Test 2 (30%)

Thank you!

Next time:

- Calculus and optimization using MATLAB