Week 10 MATLAB Activities

Contents

Time required: 120 minutes

1. Create a MATLAB script named **Wk10Lastname.m**
2. Save all programs and tutorials in this script. Functions can go in separate files or at the top of the file.
3. Include your name and date at the top of the script file as comments.
4. Put a Section Break between each program.

Reading

Matlab A Practical Introduction to Programming and Problem Solving (Stormy Attaway)

Sections 13.2

Welcome to the Matrix!

In MATLAB, images are represented as matrices. Each element of the matrix corresponds to a pixel in the image. The value of each element represents the intensity or color of the corresponding pixel.

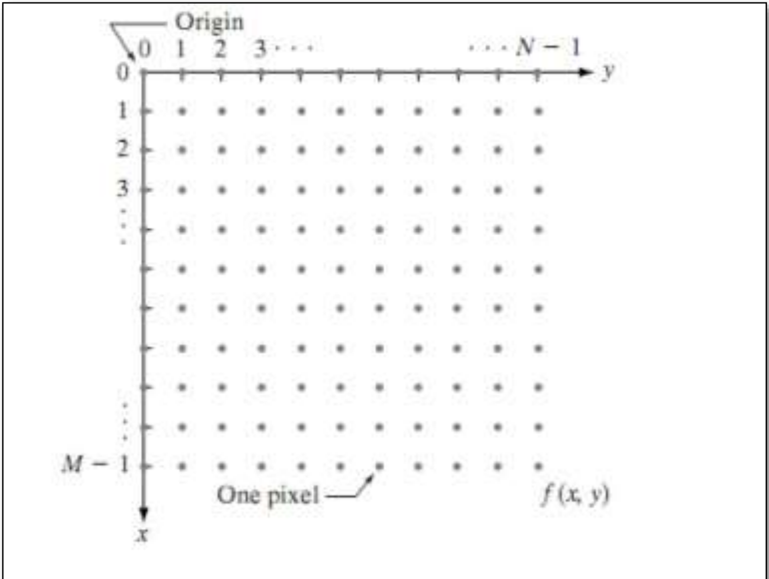
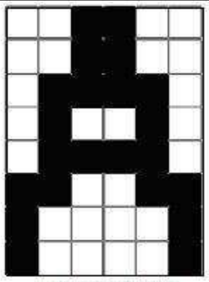


Fig 1 Coordinate convention used to represent digital images

Black and White Images

An image with only black or white pixels could be represented like the following. In this matrix 1 represents black and 0 represents white.



Digitized "A"

$$\begin{array}{cccccc}
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1
\end{array}$$

Matrix Form of "A"

```
% Create a 8x6 matrix representing the letter "A"
a_image = [0, 0, 1, 1, 0, 0;
           0, 0, 1, 1, 0, 0;
           0, 1, 1, 1, 1, 0;
           0, 1, 0, 0, 1, 0;
           0, 1, 1, 1, 1, 0;
           1, 1, 0, 0, 1, 1;
           1, 0, 0, 0, 0, 1;
           1, 0, 0, 0, 0, 1];

% Create a figure with a larger size
figure('Position', [100, 100, 600, 600]);

% Display the black and white image using imagesc
imagesc(a_image);
colormap([1 1 1; 0 0 0]); % Set colormap: [white; black]
title('Letter "A" in Black and White');

% Set the axes to display whole numbers only
ax = gca;
ax.XTick = 1:size(a_image, 2);
ax.YTick = 1:size(a_image, 1);
```

Assignment 1: Experiment with Black and White

1. Take the above B&W program.
2. Each number represents a value which is either 0 - white or 1 - black
3. Replace, add more columns/rows, and modify the grayscale to create your own work of art!

Submission: Attach your modified code and a screenshot of your masterpiece.

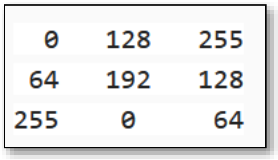
Grayscale Images

Grayscale images are represented in MATLAB as 2-dimensional matrices. Each element of the matrix corresponds to a pixel in the image, and the value of each element represents the intensity of that pixel. The intensity values typically range from 0 to 255, where:

- **0** represents black.
- **255** represents white.

- Values in between represent varying shades of gray.

Let’s look at this simple 3x3 grayscale image.

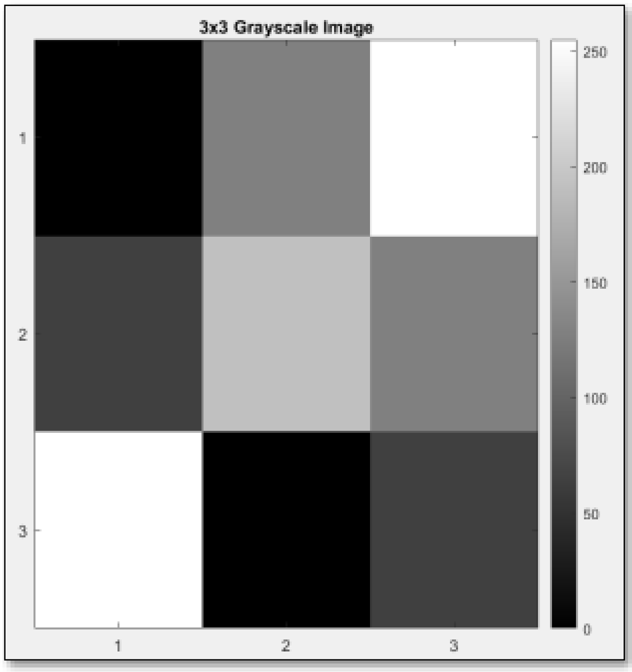


0	128	255
64	192	128
255	0	64

Let’s create and display this image in MATLAB.

```
grayImage = [0, 128, 255; 64, 192, 128; 255, 0, 64];

% Create a figure with a larger size
figure('Position', [100, 100, 600, 600]);

% Display the image using imagesc to scale the colors
imagesc(grayImage);
colormap('gray'); % Apply grayscale colormap
colorbar; % Add a colorbar to show the intensity scale
title('3x3 Grayscale Image');

% Set the axes to display whole numbers only
ax = gca;
ax.XTick = 1:size(grayImage, 2);
ax.YTick = 1:size(grayImage, 1);
```

Example run:



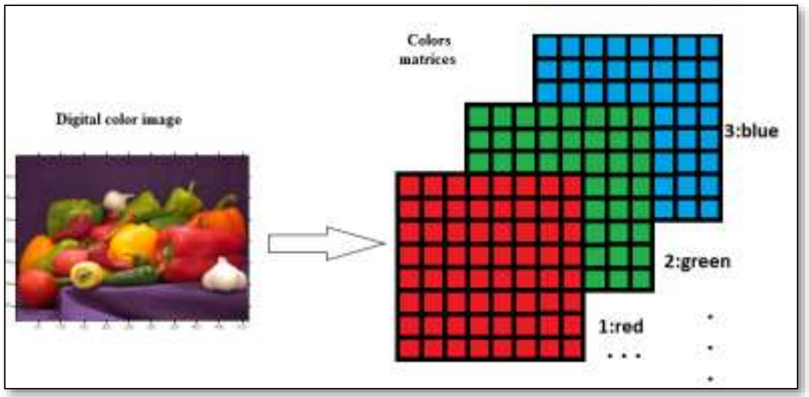
Assignment 2: Experiment with GrayScale

4. Take the above GrayScale program.
5. Each number represents a grayscale value from 0-255
6. Replace, add more columns/rows, and modify the grayscale to create your own work of art!

Submission: Attach your modified code and a screenshot of your masterpiece.

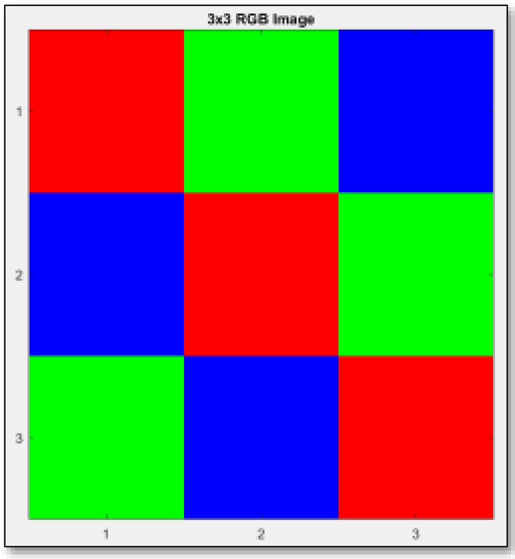
Color Images

For color images, the matrix is 3-dimensional. The first two dimensions represent the pixel coordinates, and the third dimension represents the color channels (Red, Green, and Blue). Each pixel is represented by three values corresponding to the intensity of the red, green, and blue channels.



```
% Create a sample 3x3 RGB image
rgbImage = uint8(cat(3, ...
    [255, 0, 0; 0, 255, 0; 0, 0, 255], ... % Red, Green, Blue
    [0, 255, 0; 0, 0, 255; 255, 0, 0], ... % Green, Blue, Red
    [0, 0, 255; 255, 0, 0; 0, 255, 0]));   % Blue, Red, Green

% Create a figure with a larger size
figure('Position', [100, 100, 600, 600]);

% Display the RGB image
imagesc(rgbImage);
title('3x3 RGB Image');

% Set the axes to display whole numbers only
ax = gca;
ax.XTick = 1:size(rgbImage, 2);
ax.YTick = 1:size(rgbImage, 1);
```

Example run:



Assignment 3: Experiment with RGB Colors

Take the above MATLAB code displaying an RGB image.

1. Go to https://www.rapidtables.com/web/color/RGB_Color.html or another website that shows RGB color combinations.
2. A single RGB color in MATLAB is represented as 255, 0, 0;
3. Replace, add columns/rows, and modify the colors to create your own work of art!

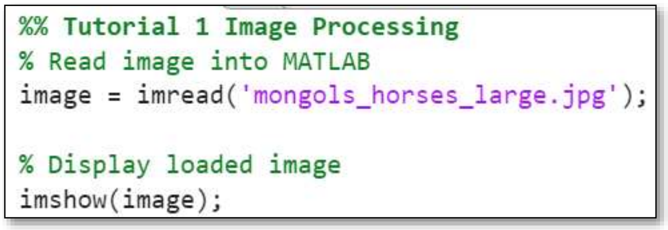
Submission: Attach your modified code and a screenshot of your masterpiece.

Tutorial 1: MATLAB Image Processing

Step 1: Load and Display an Image

To start processing an image, you need to load it into MATLAB. Use the imread function to read an image file.

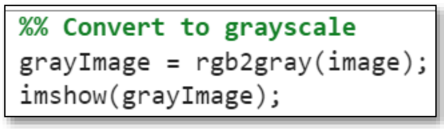
1. Use an image of your own or go to https://unsplash.com/ and download an image file.
2. Upload the image to MATLAB Online.
3. Display the loaded image using the imshow function.



```
%% Tutorial 1 Image Processing
% Read image into MATLAB
image = imread('mongols_horses_large.jpg');

% Display loaded image
imshow(image);
```

Step 2: Convert to Grayscale

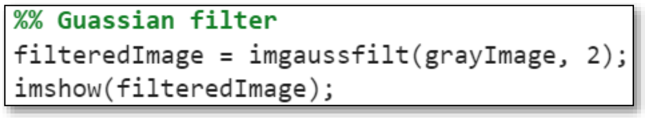
If your image is in color, you might want to convert it to grayscale for certain processing tasks. Use the rgb2gray function.



```
%% Convert to grayscale
grayImage = rgb2gray(image);
imshow(grayImage);
```

Step 3: Apply Filters

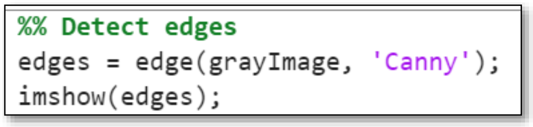
You can apply various filters to the image. For example, to apply a Gaussian filter, use the imgaussfilt function.



```
%% Guassian filter
filteredImage = imgaussfilt(grayImage, 2);
imshow(filteredImage);
```

Step 4: Edge Detection

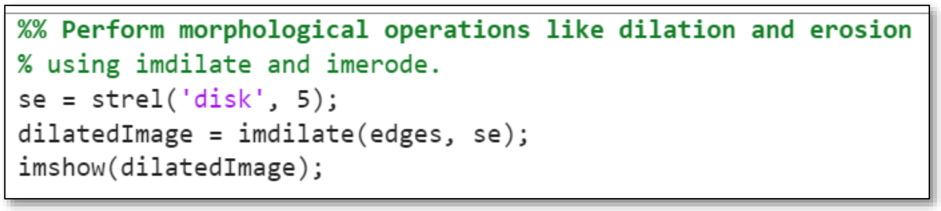
Detect edges in the image using the edge function. The Canny method is commonly used.



```
%% Detect edges
edges = edge(grayImage, 'Canny');
imshow(edges);
```

Step 5: Morphological Operations

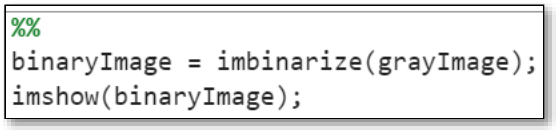
Perform morphological operations like dilation and erosion using imdilate and imerode.



```
%% Perform morphological operations like dilation and erosion
% using imdilate and imerode.
se = strel('disk', 5);
dilatedImage = imdilate(edges, se);
imshow(dilatedImage);
```

Step 6: Segment the Image

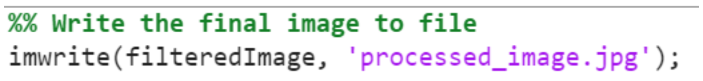
Segment the image to isolate regions of interest. Use the imbinarize function for thresholding.



```
%%
binaryImage = imbinarize(grayImage);
imshow(binaryImage);
```

Step 7: Save the Processed Image

Save the processed image using the imwrite function.

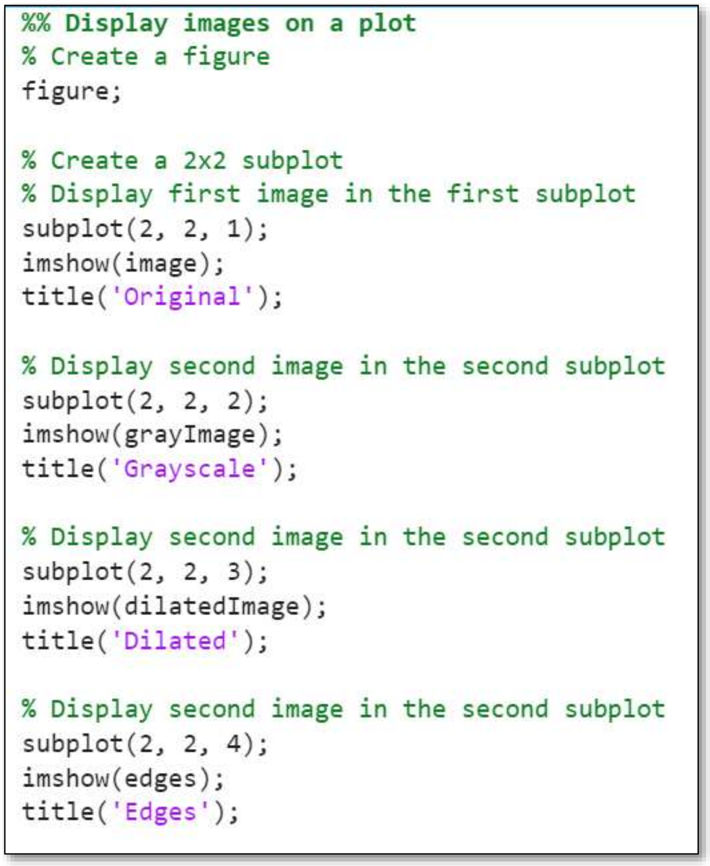


```
%% Write the final image to file
imwrite(filteredImage, 'processed_image.jpg');
```

Tutorial 2: Display Images on Plots

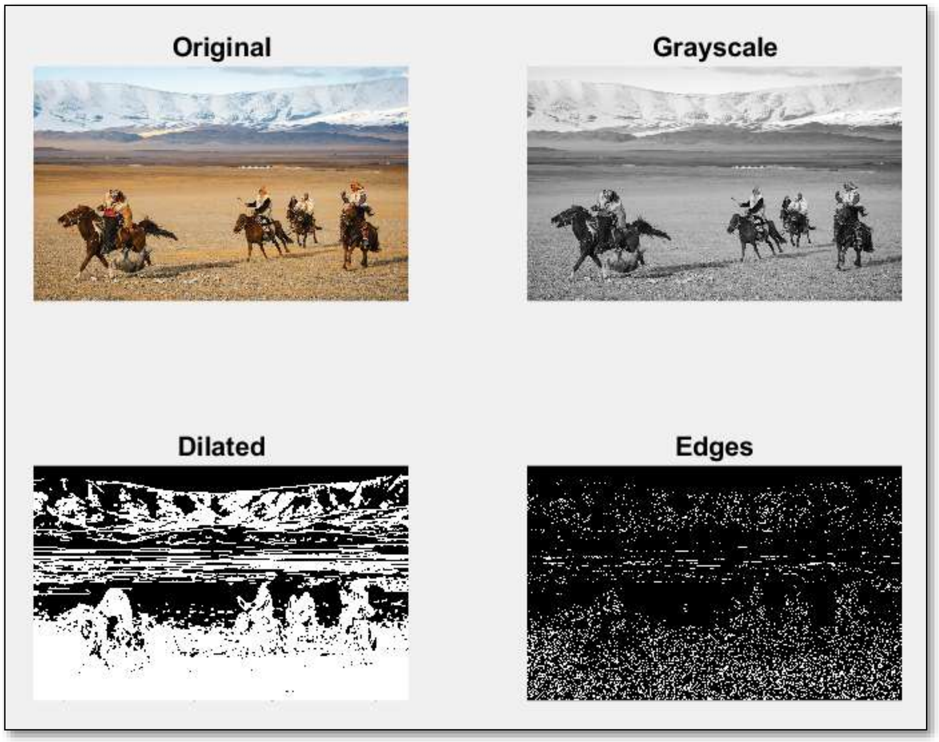
You can use the subplot command in MATLAB to display any number of images in a plot.

Pick 4 of your favorite image manipulations to display.



```
%% Display images on a plot
% Create a figure
figure;

% Create a 2x2 subplot
% Display first image in the first subplot
subplot(2, 2, 1);
imshow(image);
title('Original');

% Display second image in the second subplot
subplot(2, 2, 2);
imshow(grayImage);
title('Grayscale');

% Display second image in the second subplot
subplot(2, 2, 3);
imshow(dilatedImage);
title('Dilated');

% Display second image in the second subplot
subplot(2, 2, 4);
imshow(edges);
title('Edges');
```

Example run:



Assignment Submission

1. Submit properly named and commented script files.
2. Attach your image file.
3. Attach screenshots of your results.
4. Attach all to the assignment in Blackboard.