MED PHYS 4D03

Laboratory 0 – Introduction to MATLAB

MATLAB Setup:

Code can only be run in MATLAB if the program can find it. You should add a location on your local computer to MATLAB's path and use this same location for your work in this lab.

Likewise, MATLAB can only find image files if you give it the full path name to the file. One workaround is to navigate into the folder your images are in and run your code from there.

Create a new script in MATLAB (File -> New -> Script). Always write and test code in a script, not on the command line. It is easier to trouble shoot.

Save this script as anything you want. You can then run it by typing the filename on the command line (or by hitting the Run button in the MATLAB console).

I strongly recommend running code line-by-line in the debugging phase. You can highlight a line(s) of code, right click, and press 'Evaluate Selection'.

MATLAB Programming Basics:

Any information you need about a particular MATLAB function can be found in the program Help. You can access the help, by right clicking the function and selecting 'open' and reading the function header (if there is one!). Google can also be your best friend. MATLAB's website has lots of great documentation, examples and forums that likely have the answer to your problem.

MATLAB language is **case sensitive**! If you get warnings about a function or variable not existing, check that you spelled it correctly, with the correct case structure. If you are certain the spelling is correct, then make sure the function is on your path.

A MATLAB script doesn't require header text (but functions do/should).

Any code that follows a % is taken as a comment (i.e. is it ignored, so use this to leave yourself notes)

You can use a semicolon (;) to suppress the output of commands. This is almost always a good idea.

It *can* be a good idea to start a script with the `clear all;` command which will remove all previous variables in memory. But do not start subroutines and functions with it, or you will remove all the variables stored by your calling program.

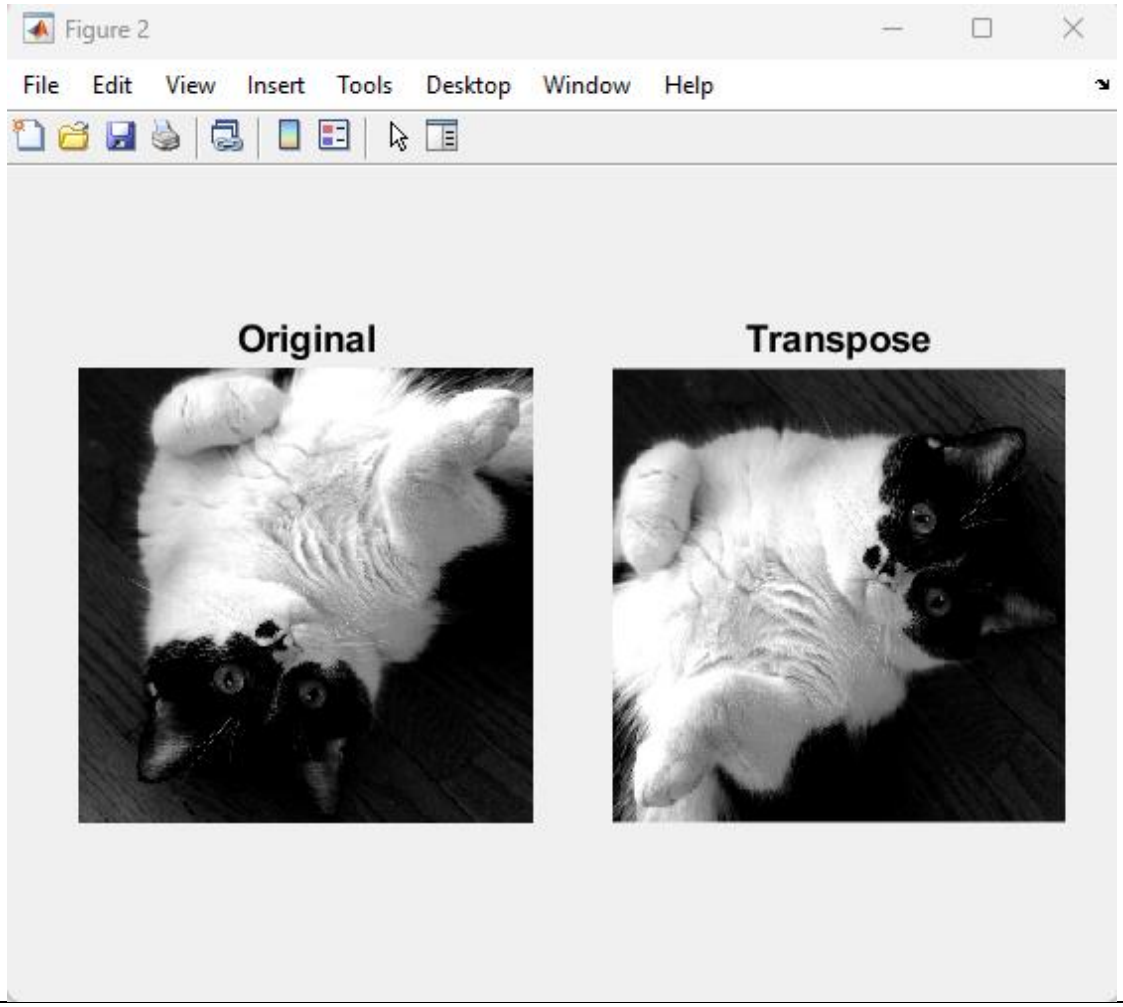
You can also start imaging scripts with `close all;` This will close all open figure windows from your last script and keep your workspace uncluttered.

Example:
Below is a sample script that loads the image 'cat.tif', displays it,

```
% The following script will lead you through some basic operations related
% to images, like opening an image and finding its dimensions and data type.

% A two-dimensional image is an n by m matrix of values, consisting of 'n' rows and
% 'm' columns.  The values can be of several different data types.

pic = imread('cat.tif');

% Convert to double so you can have values extending past 255.
pic = double(pic);


% 'imread' will read the image file saved as 'cat.tif' into the variable 'pic'.
% Note the use of quotation marks with the file name. You can use any variable name
% of your choice.
% Note the use of semicolon at the end of the command to suppress the output.
% Without it, MATLAB will display the entire matrix in the command window.

%% Double comment then space creates a section break
% Create a MATLAB figure and display it.

% This line creates a figure graphics object. The parts in the brackets give the
figure a nicely formatted title.
figure('Name','Cat','NumberTitle','off');
imagesc(pic);
colormap('gray');
axis off;

% 'figure' generates a new figure object each time you run the command. Otherwise
% MATLAB will replace the first open figure with the second and so on.
% 'imagesc' intensity scales the data and displays the image.
% 'colormap('gray')' maps the image values to a range of gray level intensities.
[xsize, ysize] = size(pic);

% xsize and ysize are now variables holding the n (or x) dimension and the m (or y)
dimension respectively.  You can use these in your script, for example:
filesize = xsize * ysize;

% Note that you need to convert a number to a string to display it.
disp(['The image has: ', num2str(filesize),' pixels'])
```

Lab exercises:

1) Images in MATLAB are just treated as matrices. Try taking the transpose of the image, and display the result. Use tiledlayout to make a single figure object so you can compare the transposed image to the original. The result should look like:



2) It is often useful to find the minimum and maximum pixel values in an image. Find these in your image using the max() and min() commands. Note: these commands work on 1D arrays only, but you can convert an ND matrix into a 1D vector using 'X(:)'.

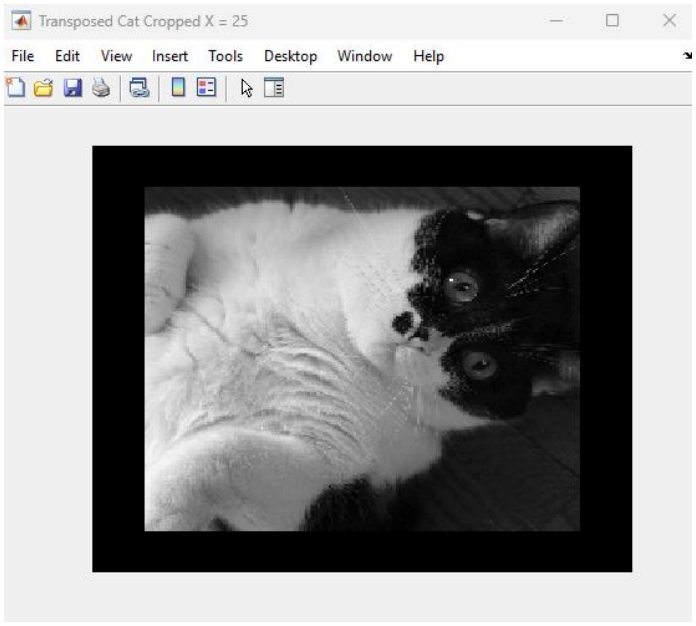
You should find that the max is 255 and the minimum is 0.

3) Find the matrix indices where the maximum value occurred using the find() command in MATLAB. Use the Data Cursor in your figure window to confirm your finding.

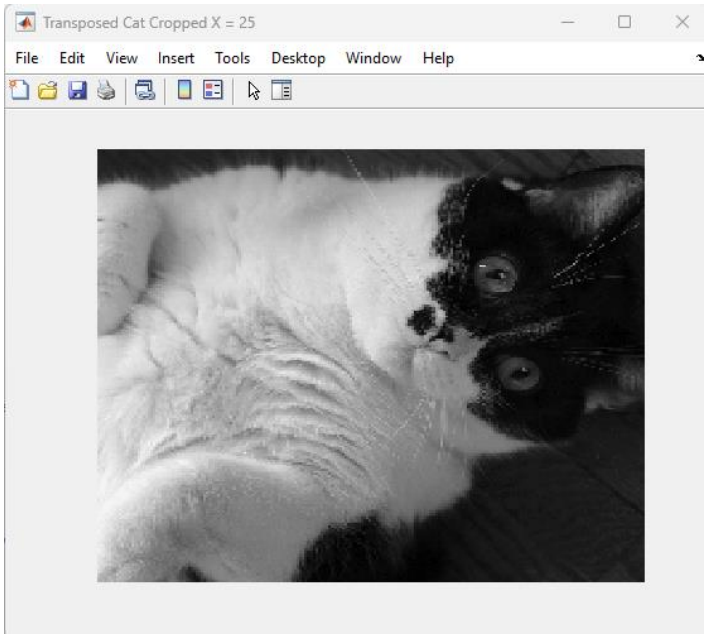
If using the transposed image, this should be at x= 81, y =170.

4) Create a **crop function** in MATLAB, which sets the pixel values X deep from the edge of the image to 0, leaving the center of the image. Create a second crop function that removes the rows and columns, instead of setting them = 0.

For X = 25, the first function should return:

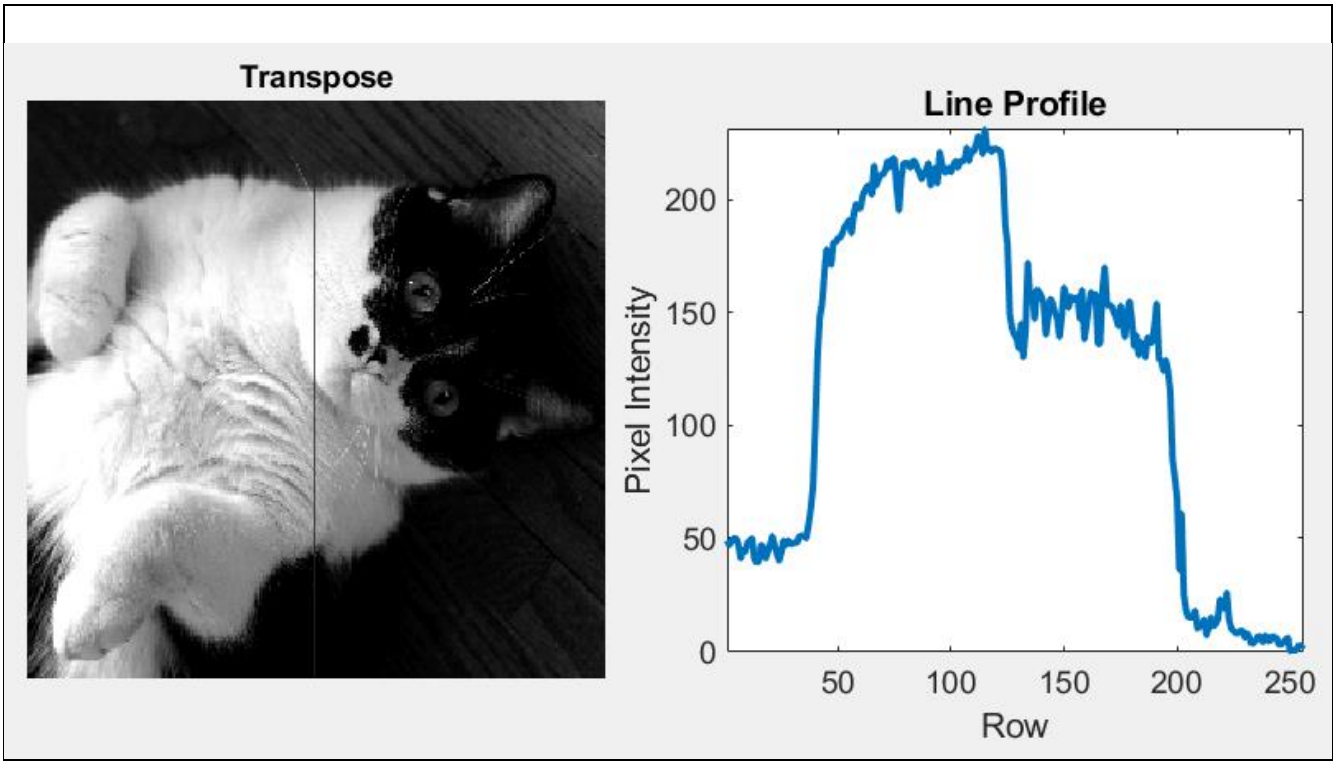


The second function should return:



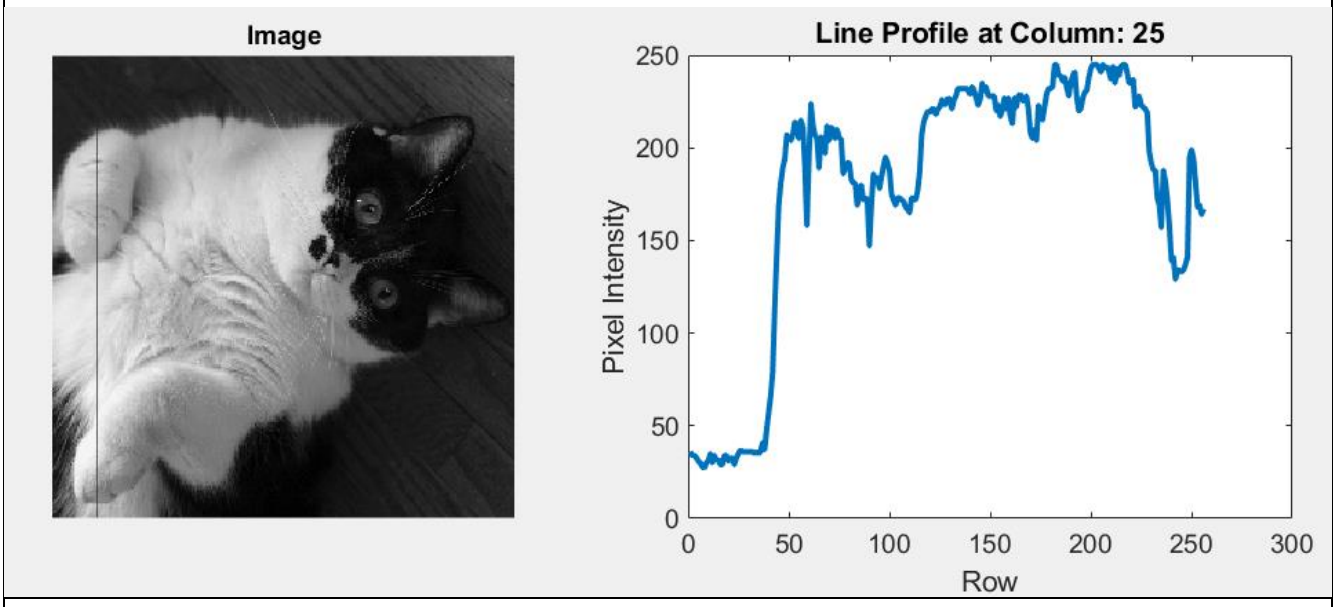
5) Build a two-image figure using ‘tiledlayout’, in tile 1, display the cat image. In tile 2, display a line profile ‘plot’ of the image intensities of the middle column. Set the ‘LineWidth’ to 3 so the line is easier to see. You can add a vertical line in the cat image, but adding a ‘xline’ to the plot.

The result should look like:

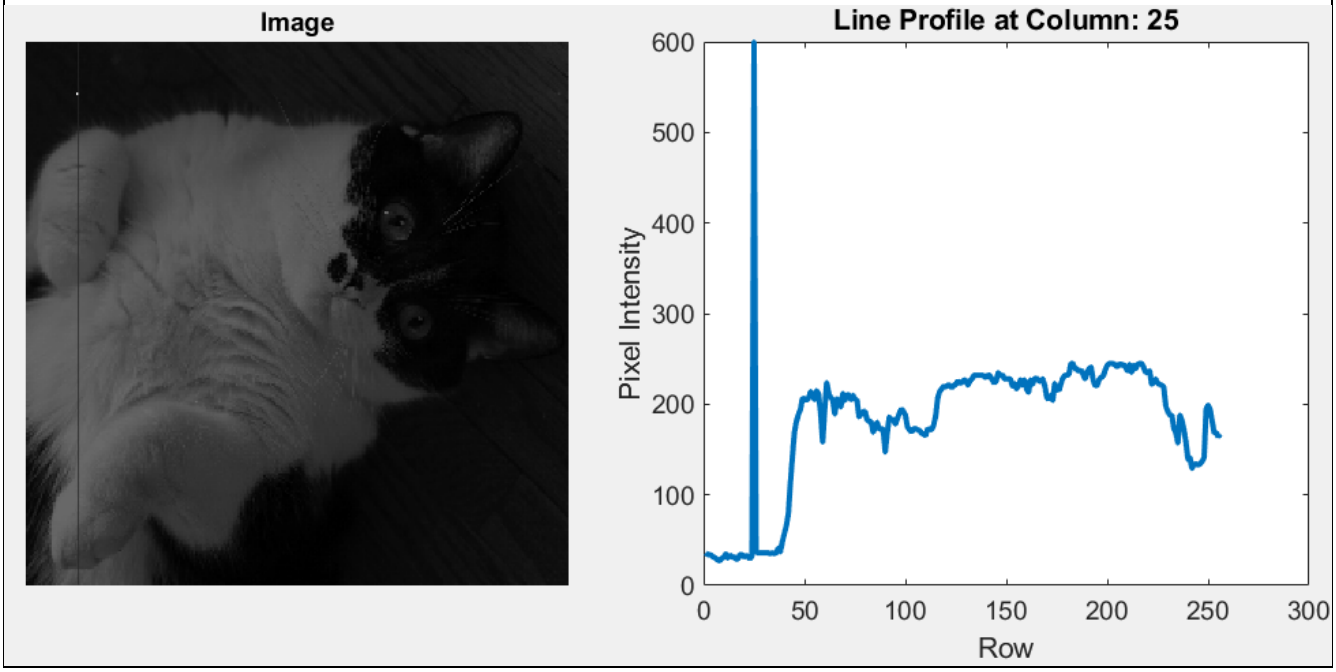


6) Put the code you developed for 5) into a function file called ‘**twoImageTiledDisplay(**img1, column**)**’. Building on 5), set the maximum of the colormap for the cat to the maximum value in the cat image, and the minimum of the colormap to the minimum value. The function should display the picture and return a 2x1 vector that returns [low, high] of the cat image. The line profile should be from the column specified as an input to the function.

You should check the functionality of this by changing a pixel value in the image to something ~ 600 and see how if it still works.



Pixel [25,25] set to 600:



I have provided other sample code in 'SamplePlotCode.m', which is primarily to address minor cosmetic things with MATLAB images, such as presetting the size of the figure window, adjusting plot size, using tiled layout, and playing around with axes. A font size of 16-20 is usually recommended for figures so that it is legible when it is inevitably shrunk for publication.