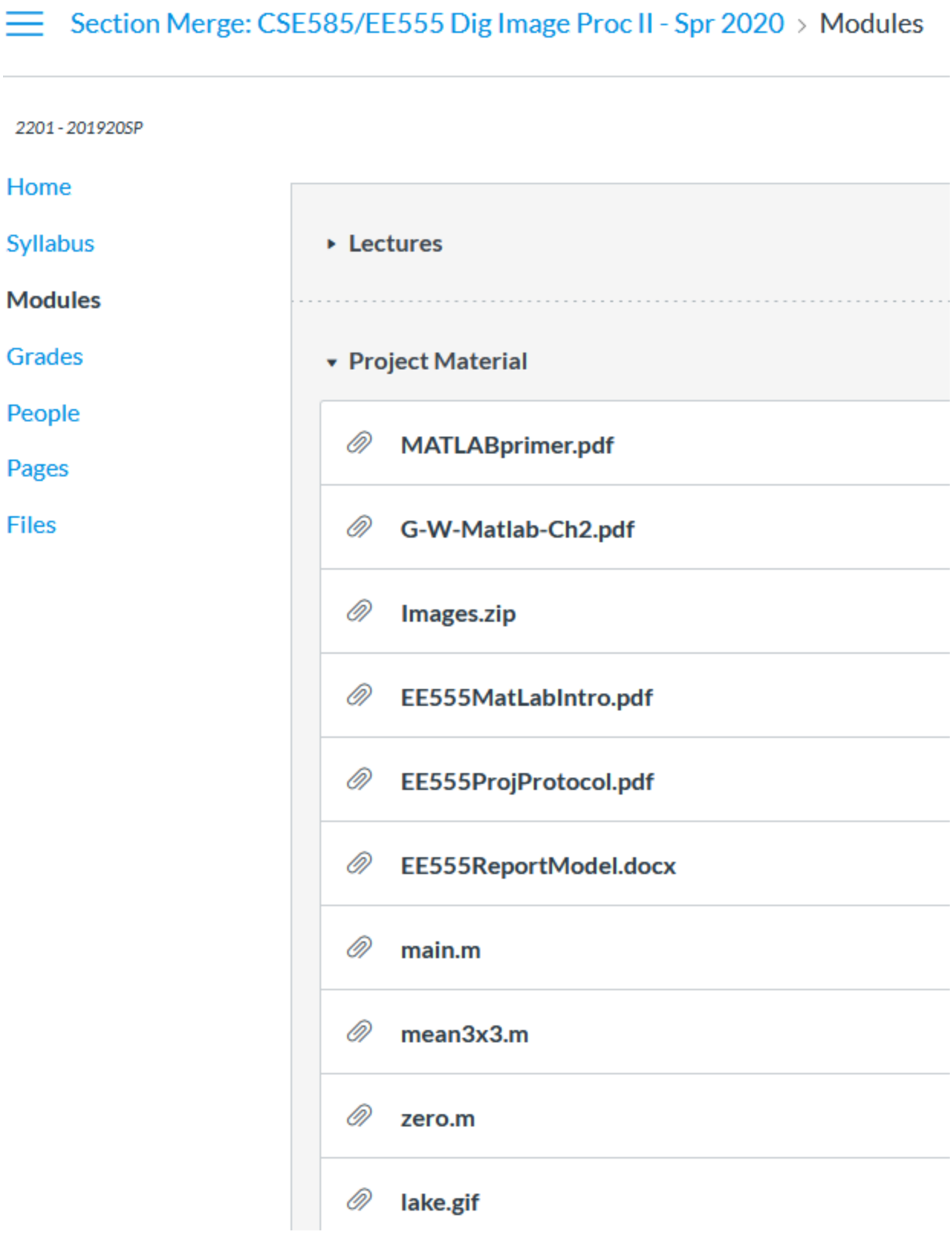

Section Merge: CSE585/EE555 Dig Image Proc II - Spr 2020 › Modules
2201 - 201920SP
Home
Syllabus
Modules
Grades
People
Pages
Files
Lectures
Project Material
MATLABprimer.pdf
G-W-Matlab-Ch2.pdf
Images.zip
EE555MatLabIntro.pdf
EE555ProjProtocol.pdf
EE555ReportModel.docx
main.m
mean3x3.m
zero.m
lake.gif

CSE585/EE555 MATLAB Introduction

This document gives a brief overview of using MATLAB for processing images. The folder called "Project Material" on our class CANVAS site give sample *.m files and sample images used for doing projects. These example files are discussed below.

The folder "Project Material" contains chapter 2 of the following book:

Digital Image Processing using MATLAB
R.C. Gonzalez, R.E.Woods, S.L. Eddins, Upper Saddle River, NJ: Pearson Prentice-Hall, 2004.

This chapter gives a nice overview of MATLAB for image processing. In addition, the folder "Project Material" contains information on how to write proper project reports.

The remainder of this document contains the following:

MATLAB Demo files and General Coding Guidelines

1. The files in folder "Project Material" – main.m, mean3x3.m, zero.m – provide a very simple demo of MATLAB for images. You will need the image "lake.gif" to run this demo.

2. Regarding MATLAB code for CMPEN/EE455 projects:
   a. **You may NOT use highly compressed ARRAY functions for processing images**!

      → You must cycle through an image's 2D pixel array with **For** loops.

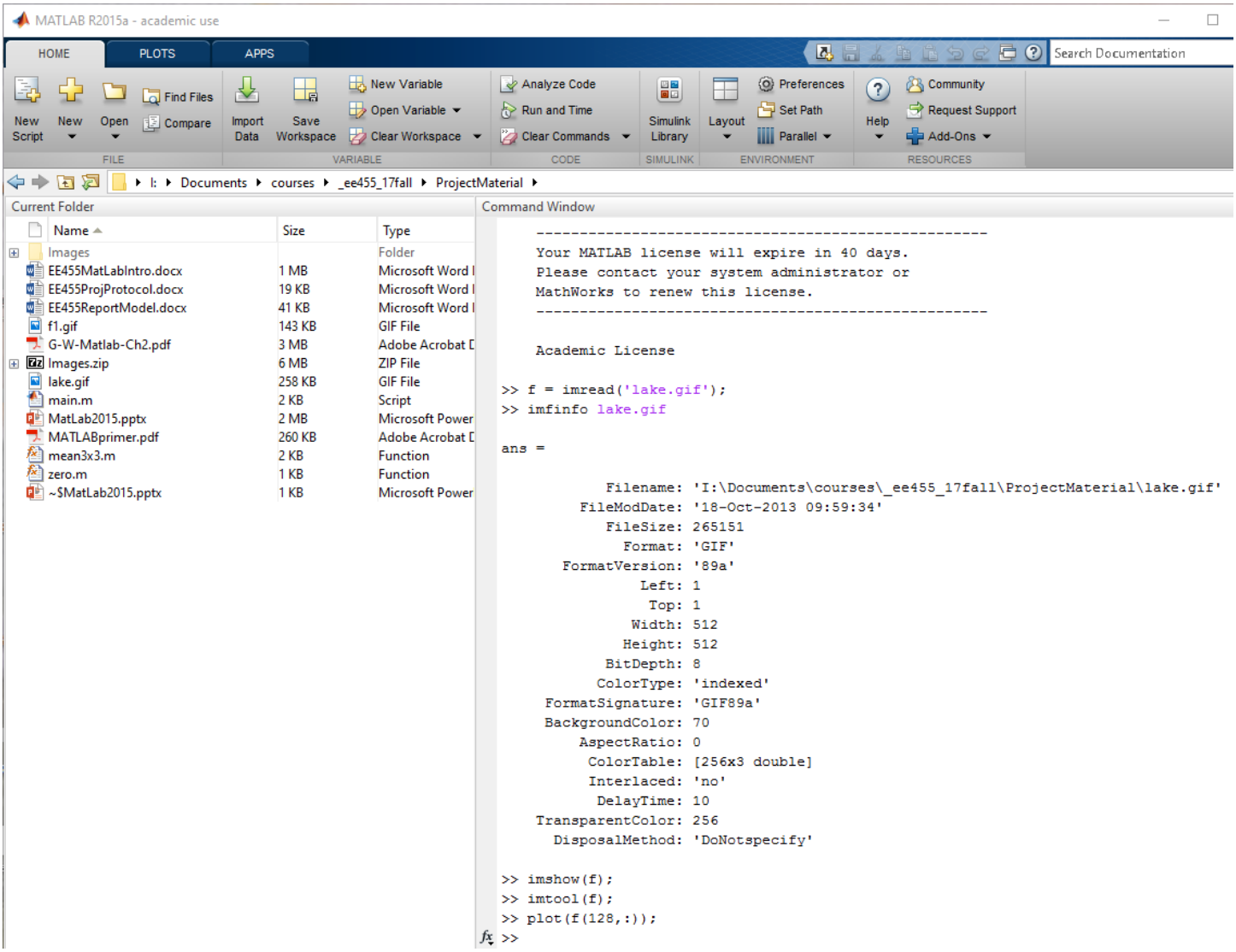
   b. Use **For** loops for the "x" and "y" indices for processing pixel data, as I do in the example *.m files.

   c. Be sure to handle border artifacts properly – either zero them out or pad the original data with zeroes.

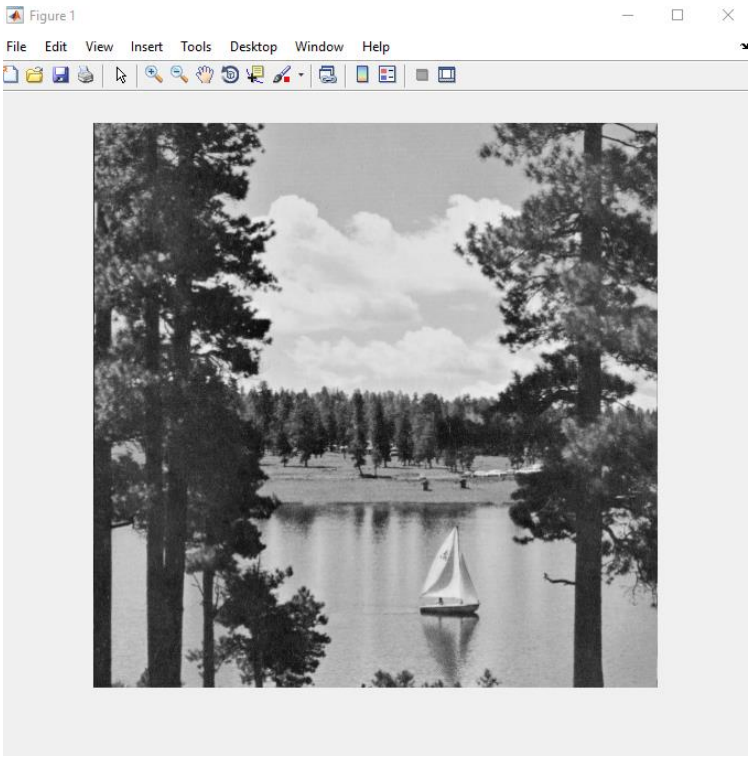
   d. Beware that our images are 8-BIT images! You will typically have to convert them to some higher precision data type BEFORE processing.

   e. Images need to be converted back AFTER processing for writing out or display.

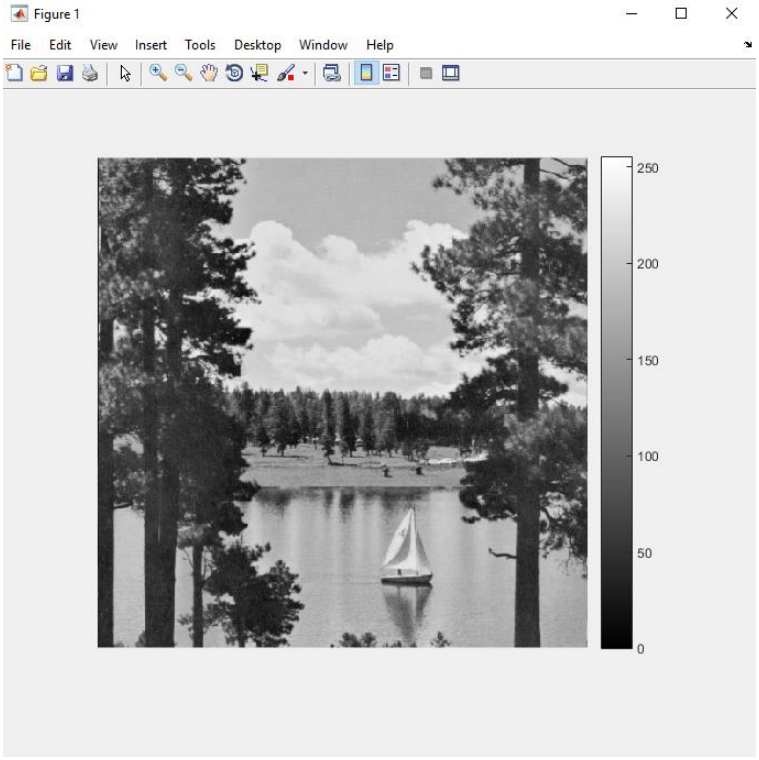
   f. See the document "EE555ProjProtocol" for more guidelines.



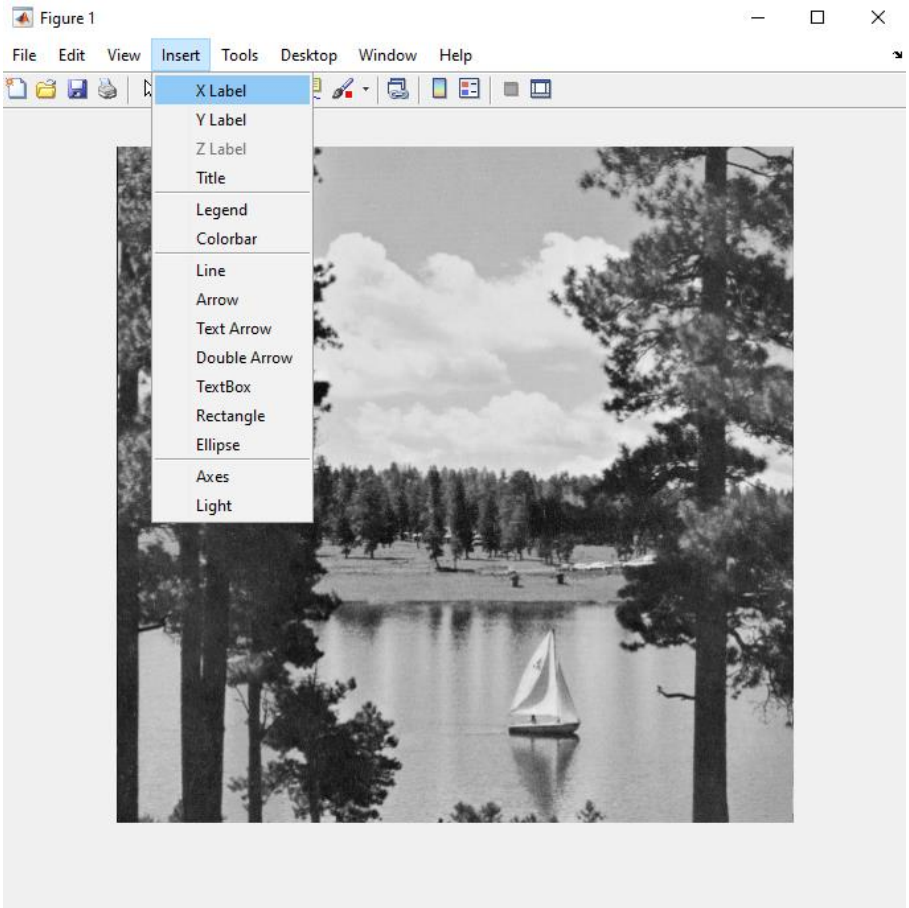
Running commands in the Command Window: **imread**, **imfinfo**, etc.



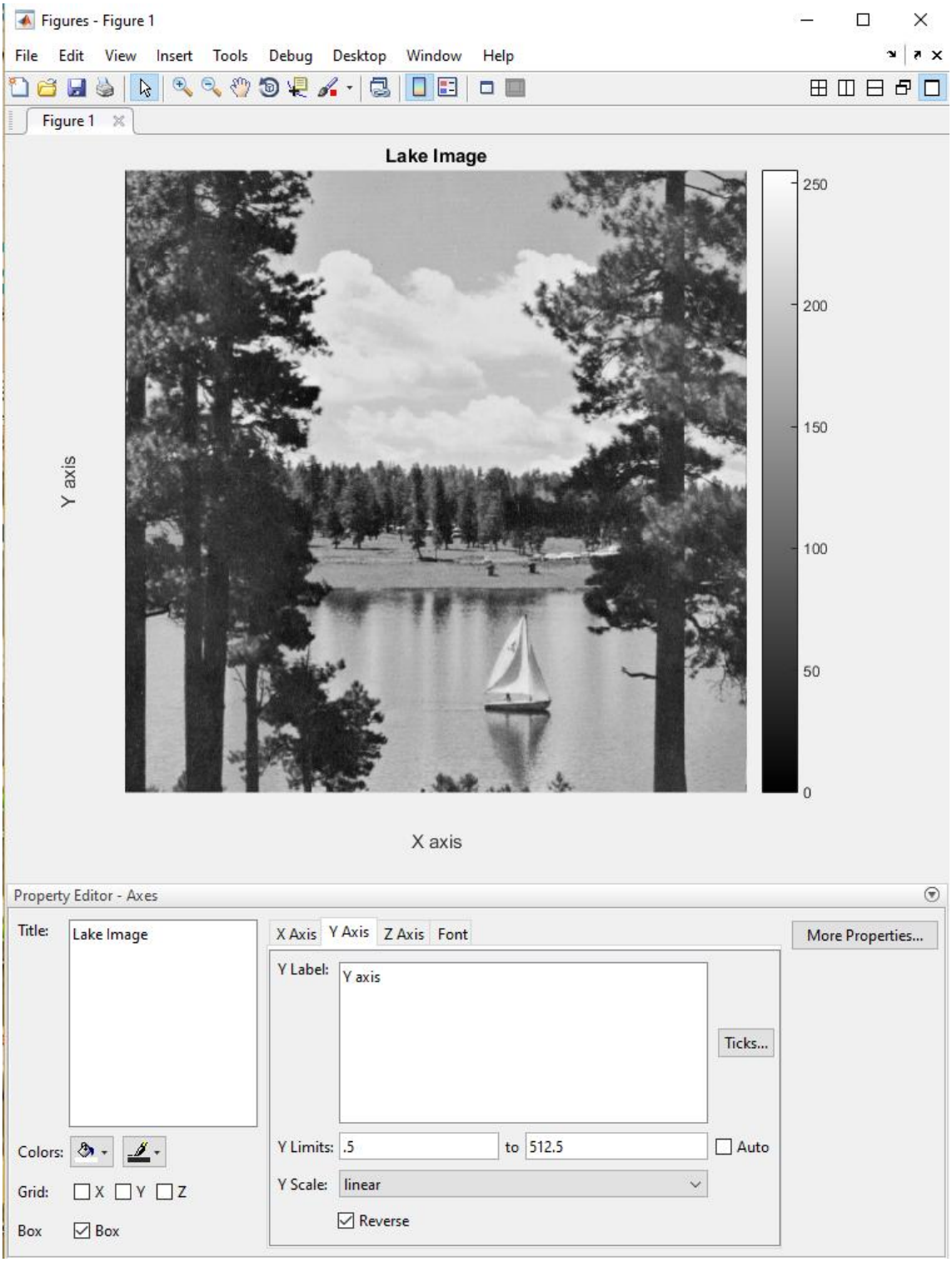
Raw output of **imshow**



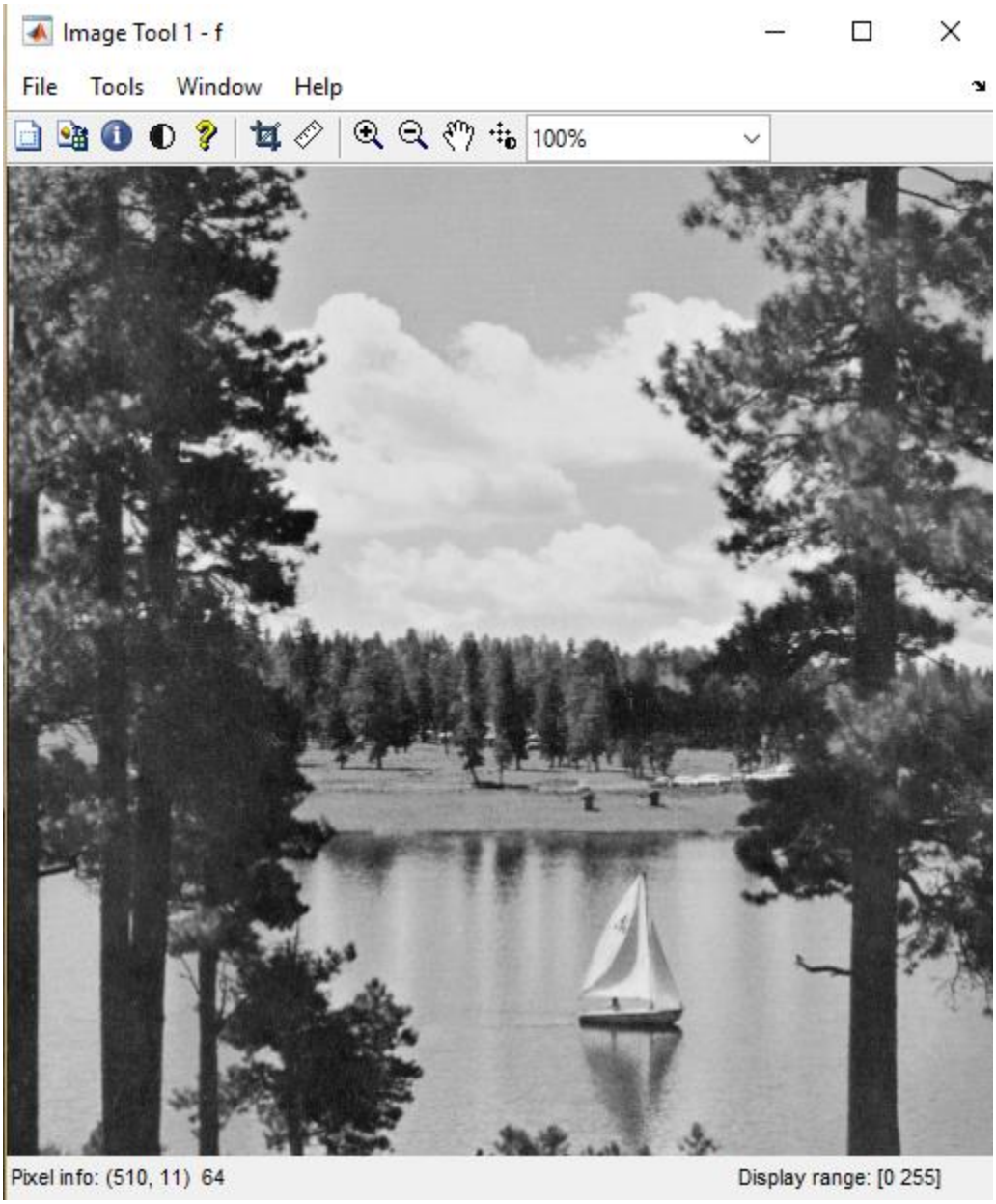
imshow showing gray-scale bar



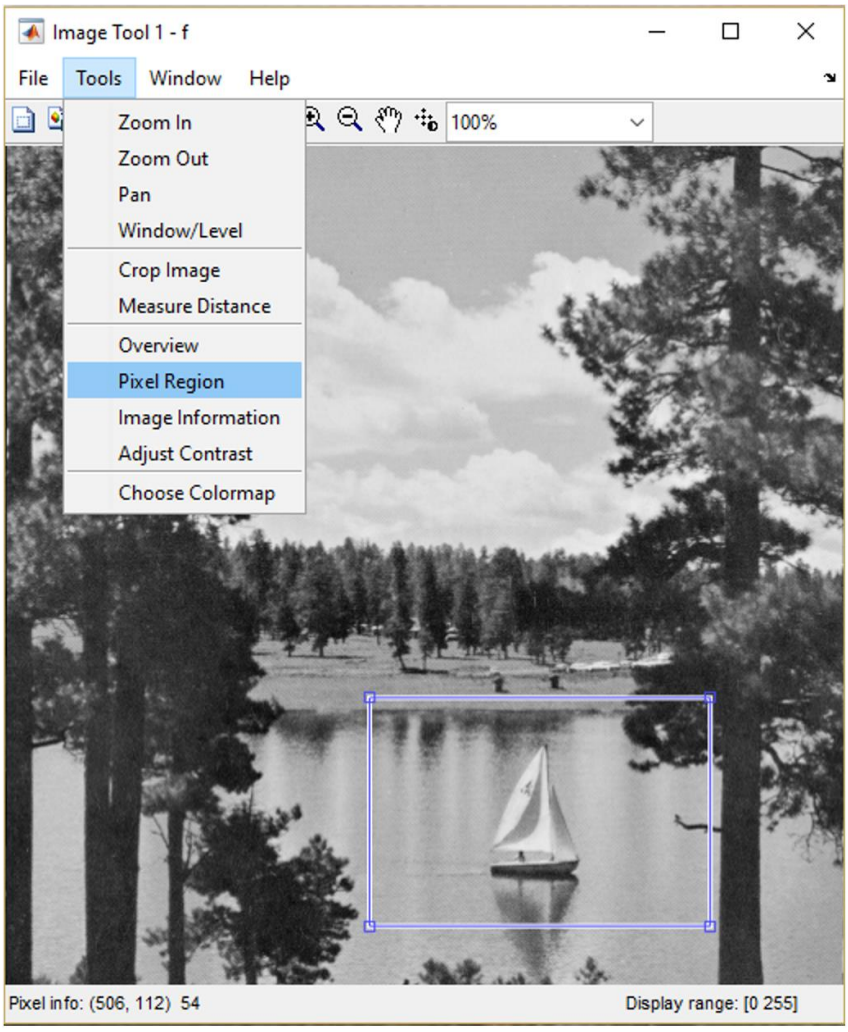
Various tools for annotating image in **imshow**

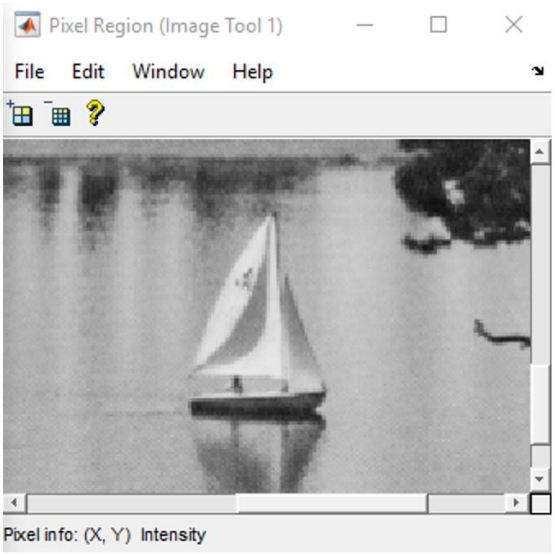


Plot options for **imshow**

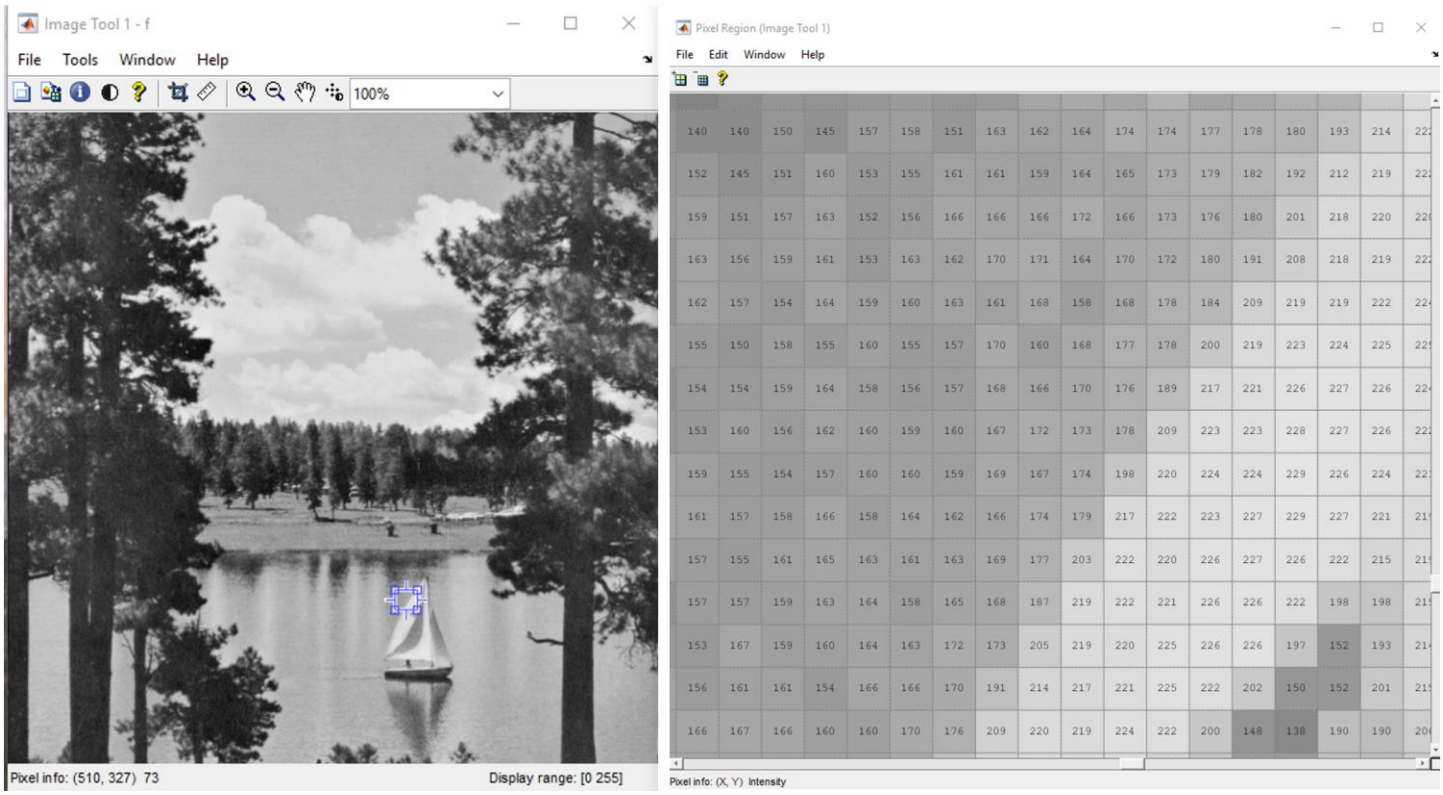


imtool for displaying images

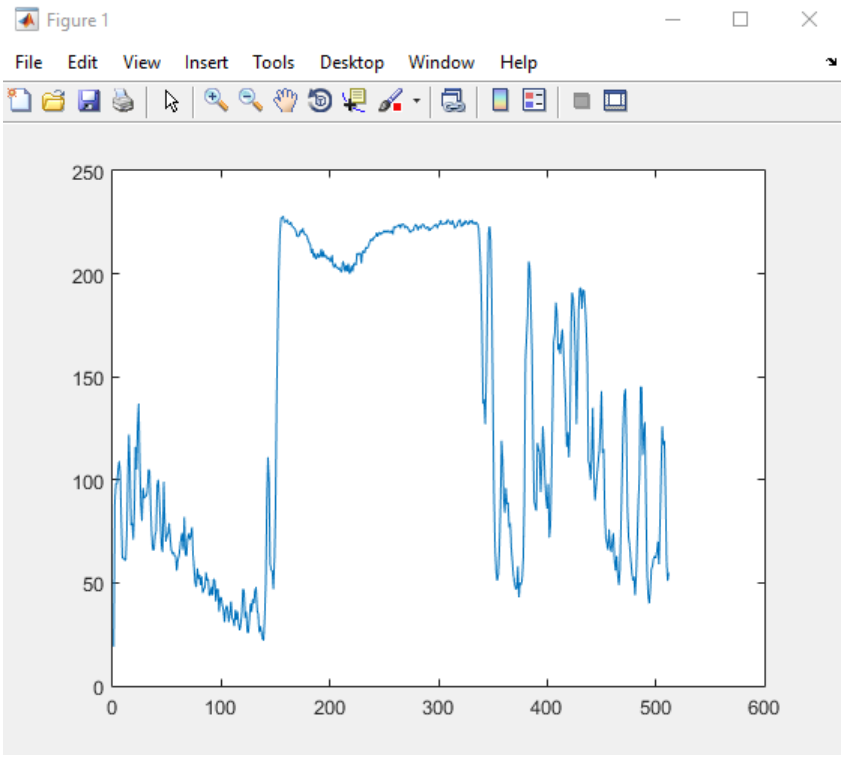




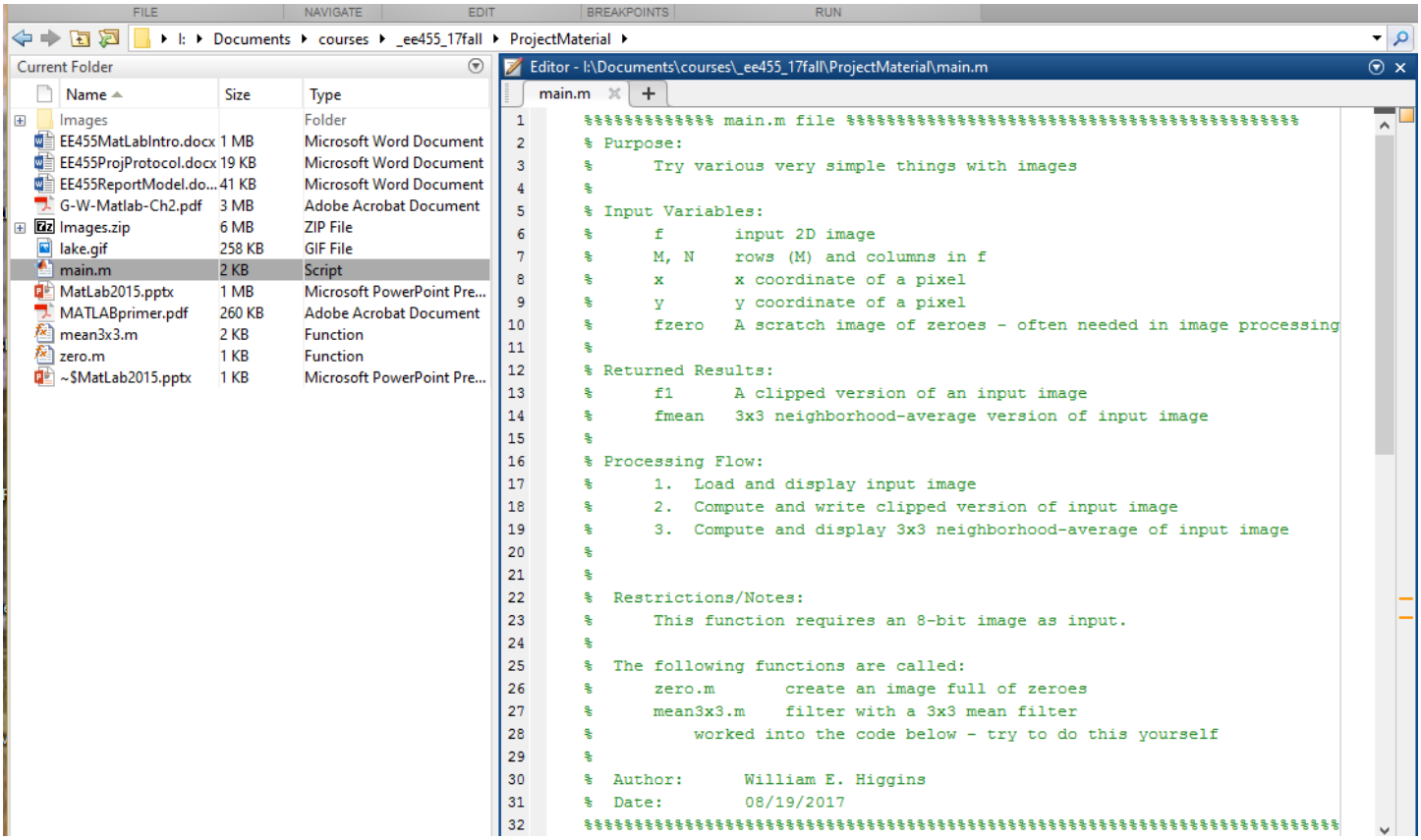
Selecting a pixel region and zooming with **imtool**



Zooming down to pixel values in **imtool**



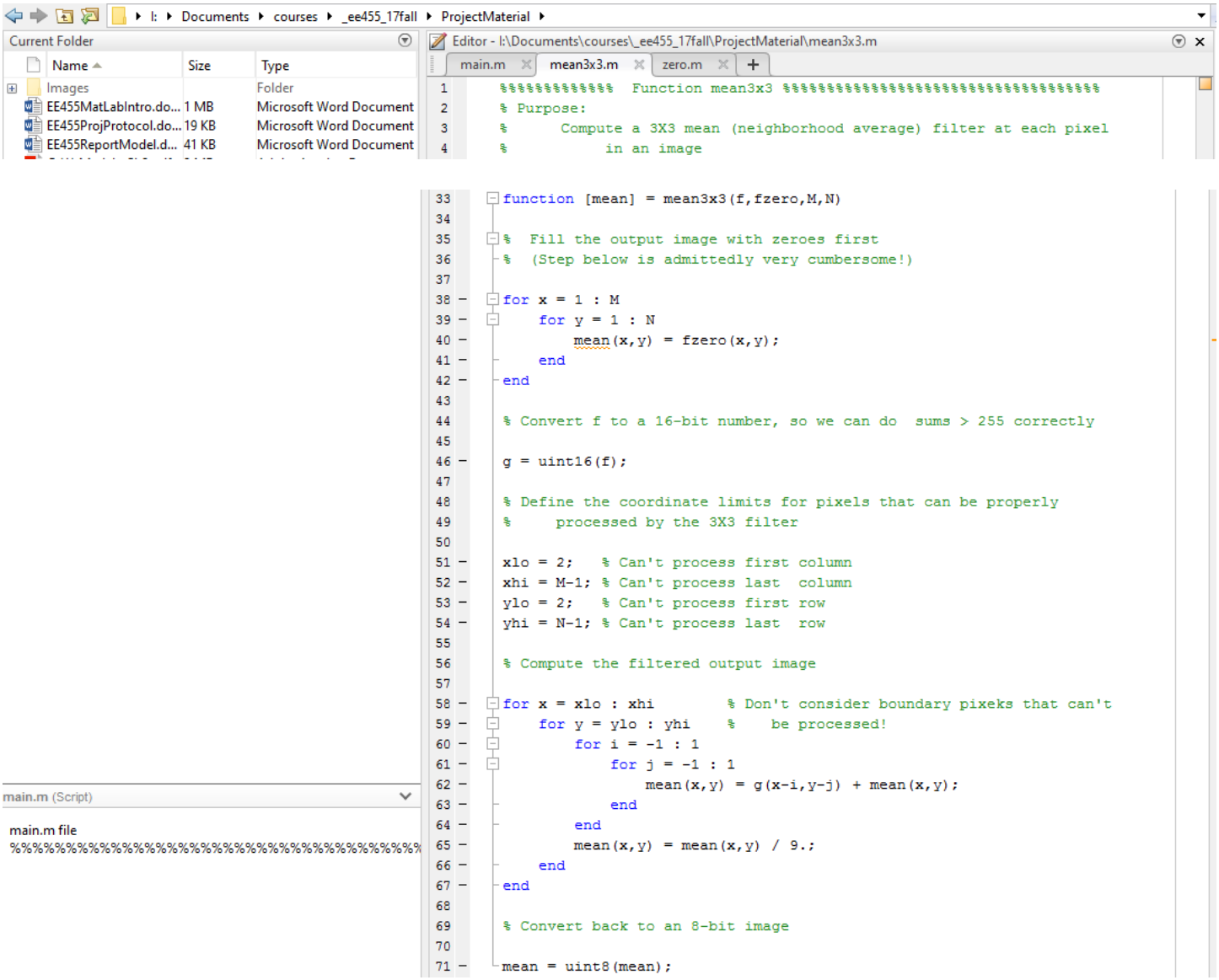
plot function for plotting the line x=128 in the “lake” image



Top part of **main.m** file in the bin area on our CANVAS site

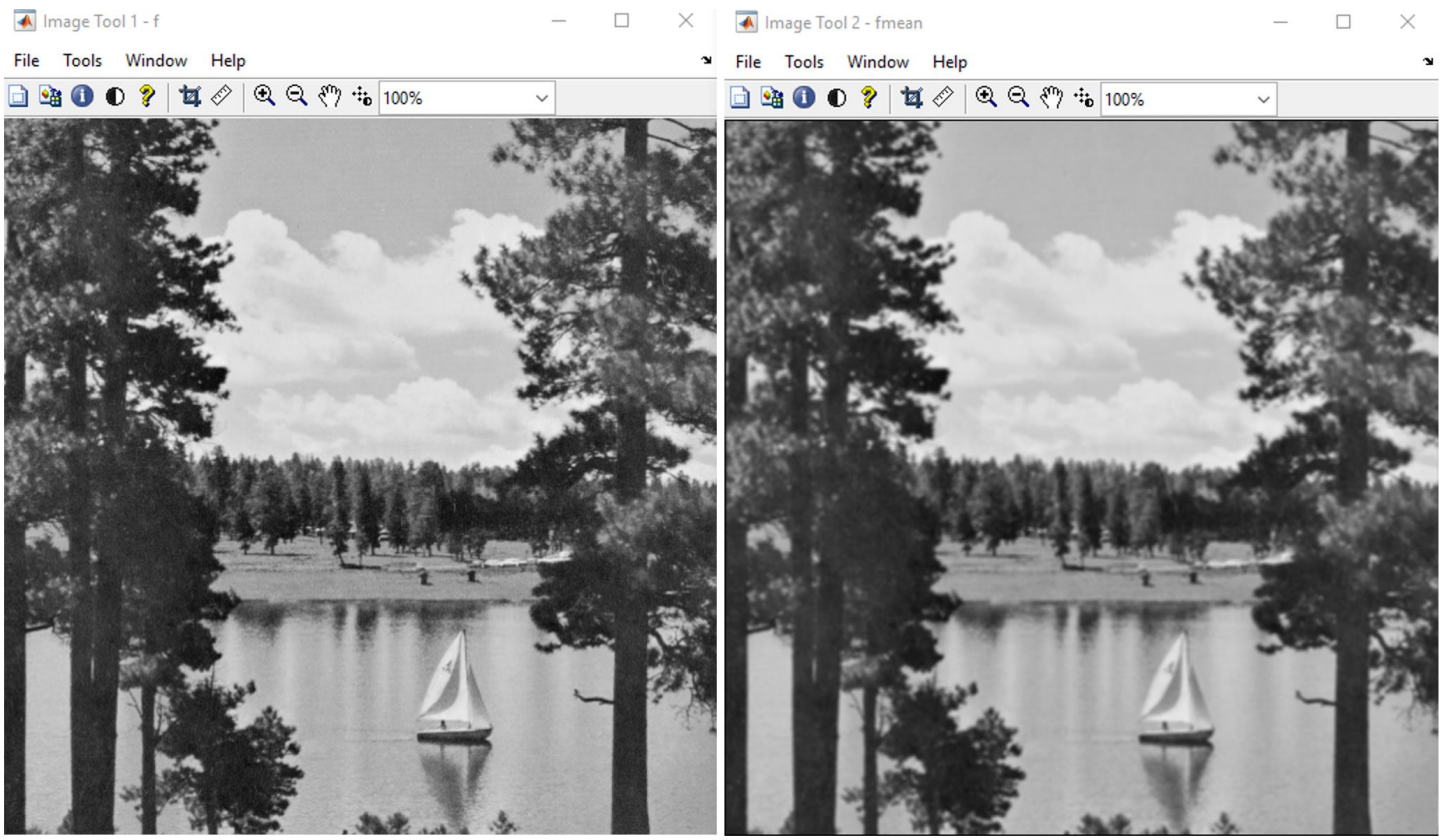
```
34 -    clear;      % Clear out all memory
35
36      %  Read in image "lake.gif" into array "f", get its 2D dimensions,
37      %      and display it on the screen
38      %  Note that "lake.gif" is needed in the directory of this .m file
39
40 -    f=imread('lake.gif');
41 -    [M, N] = size(f);
42 -    imtool(f);
43
44      %  Create and write a new image "f1" that is the same as the original image
45      %     but truncates all gray levels to a max value of 128
46
47 -    for x = 1 : M
48 -        for y = 1 : N
49 -            if f(x,y) > 127
50 -                f1(x,y) = min(f(x,y),128);
51 -            else
52 -                f1(x,y) = f(x,y);
53 -            end
54 -        end
55 -    end
56 -    imwrite(f1,'f1.gif');
57
58      % Do a very simple 3x3 neighborhood average on image "f"
59      %    1.  Create an image "fzero" loaded with zeros
60      %    2.  Create and display an MxN image "mean" which contains
61      %        the 3x3 filtered version of f, with zeroes in the border
62      %        locations that can't be filtered
63
64 -    fzero = zero(M,N);
65 -    fmean = mean3x3(f,fzero,M,N);
66 -    imtool(fmean);
67
68      %%%%%%%%%%%%%% End of the main.m file %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Body of **main.m** – this is run in MATLAB.

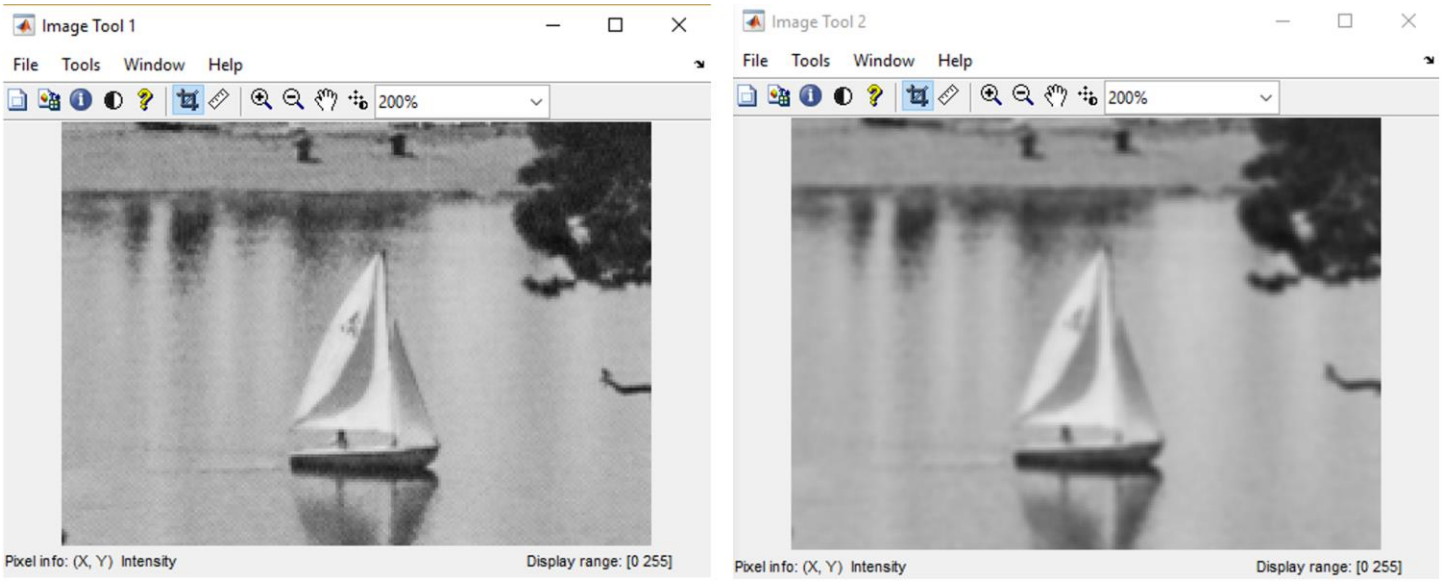


Function **mean3x3** used by the **main.m** file.

Outputs of running **main.m**:



Original “lake” (image “f”) and the 3x3 local-mean filtered version “fmean”



200% zoom on the images “f” and “fmean” to see the differences better

MATLAB INTRODUCTORY TUTORIAL

1. CONTROL STRUCTURES IN MATLAB :

The 'if' statement :

```
if  condition
        if block
else
        else block
end
```

The 'switch' statement :

```
switch variable
        case value1
                statement block 1
        case value 2
                statement block 2
        .
        .
        .
        otherwise
                default statement block
end
```

NOTE : 'otherwise' is the matlab equivalent of the keyword 'default' used in 'switch' statements in C/C++.

The 'for' statement :

```
for variable = m : i : n
        for loop block
end
```

m ← Initial value of the loop variable
i ← Increment to the value of the loop variable after the end of each execution of the loop block
n ← The value of the loop variable above which the loop should be terminated

The 'while' statement :

```
while condition
        while loop block
end
```

2. SCRIPT FILES :

MATLAB provides the option of writing a set of instructions into a single file called a script file (extension '.m'). This helps the user to write them into one file and storing it in memory so that he/she can execute them just by typing the name of the script file without the extension in on the command line. These files are particularly helpful for writing functions which will be described in the next section.

3. WRITING FUNCTIONS IN MATLAB :

The syntax for the definition of a function is:

function [*return argument list*] = *function_name* (*input argument list*)

function body

end

The function definition should be saved in a script file whose filename should be the name of the function i.e in this case, "*function_name.m*". Note that unlike C/C++ and some other HLLs, Matlab allows a function to return more than one object.

An additional feature of functions supported by Matlab is variable number input/output arguments. The keywords 'varargin' and 'varargout' can be used in the function definition to denote variable number of input and output arguments respectively. In such a situation, two other useful pre-defined variables are 'nargin' and 'nargout' which contain the number of input and output arguments provided while calling the function at run time.

NOTE : It is advised to use 'varagin' and 'varagout' always at the end of an argument list.

4. INTRODUCTION TO THE IMAGE PROCESSING TOOLBOX :

MATLAB provides built-in functions that can be used for various image operations, including:

imread ← reads an image from secondary storage and returns a matrix of gray values (for a grayscale image)

imwrite ← writes a matrix of gray values into an image file of specified format

imshow ← displays the image represented by the grayscale matrix

imfinfo ← returns a structure whose fields contain information about an image in a graphics file. Some of these fields are height, width, color type, etc.,

imhist ← displays the histogram plot of an image

*F**or syntax and detailed function descriptions, type 'help' followed by the function name on the Matlab command line**.

Makeup artist Mimi Choi turned her face into a brick wall.