

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:

<u>Page</u>	<u>Topic</u>
3	General intro and MATLAB coding guidelines
4-14	Cookbook demonstration of MATLAB
15-16	Some basic MATLAB syntax and control structures.

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.

The screenshot shows the MATLAB R2015a - academic use interface. The top toolbar includes tabs for HOME, PLOTS, and APPS. The main workspace displays the 'Current Folder' pane on the left, showing a list of files and folders in the directory 'I:\Documents\courses_ee455_17fall\ProjectMaterial'. The 'Command Window' on the right shows the execution of MATLAB commands to read and display an image.

Current Folder:

Name	Size	Type
Images		Folder
EE455MatLabIntro.docx	1 MB	Microsoft Word Document
EE455ProjProtocol.docx	19 KB	Microsoft Word Document
EE455ReportModel.docx	41 KB	Microsoft Word Document
f1.gif	143 KB	GIF File
G-W-Matlab-Ch2.pdf	3 MB	Adobe Acrobat Document
Images.zip	6 MB	ZIP File
lake.gif	258 KB	GIF File
main.m	2 KB	Script
MatLab2015.pptx	2 MB	Microsoft PowerPoint
MATLABprimer.pdf	260 KB	Adobe Acrobat Document
mean3x3.m	2 KB	Function
zero.m	1 KB	Function
~\$MatLab2015.pptx	1 KB	Microsoft PowerPoint

Command Window:

```

-----
Your MATLAB license will expire in 40 days.
Please contact your system administrator or
MathWorks to renew this license.
-----

Academic License

>> f = imread('lake.gif');
>> imfinfo lake.gif

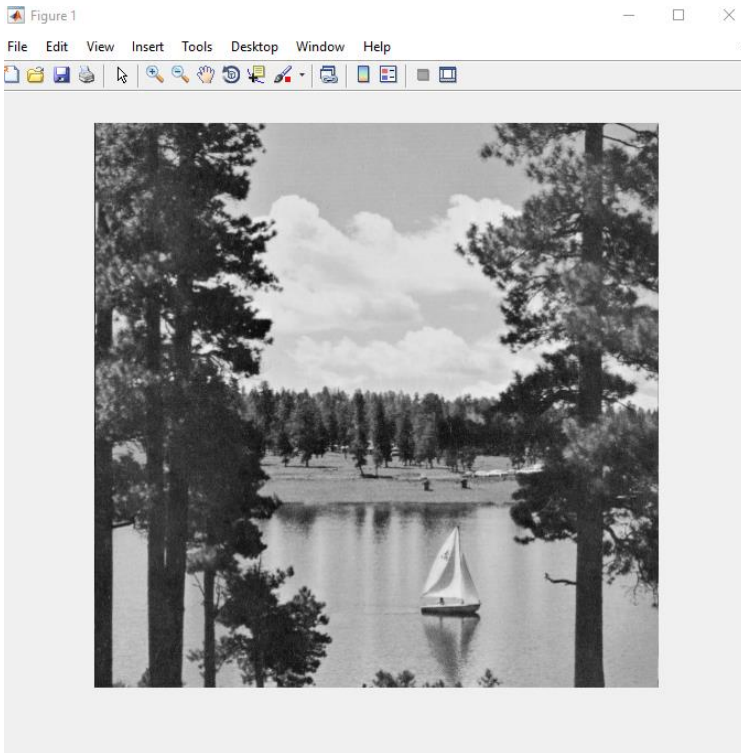
ans =

    Filename: 'I:\Documents\courses\_ee455_17fall\ProjectMaterial\lake.gif'
    FileModDate: '18-Oct-2013 09:59:34'
    FileSize: 265151
    Format: 'GIF'
    FormatVersion: '89a'
    Left: 1
    Top: 1
    Width: 512
    Height: 512
    BitDepth: 8
    ColorType: 'indexed'
    FormatSignature: 'GIF89a'
    BackgroundColor: 70
    AspectRatio: 0
    ColorTable: [256x3 double]
    Interlaced: 'no'
    DelayTime: 10
    TransparentColor: 256
    DisposalMethod: 'DoNotSpecify'

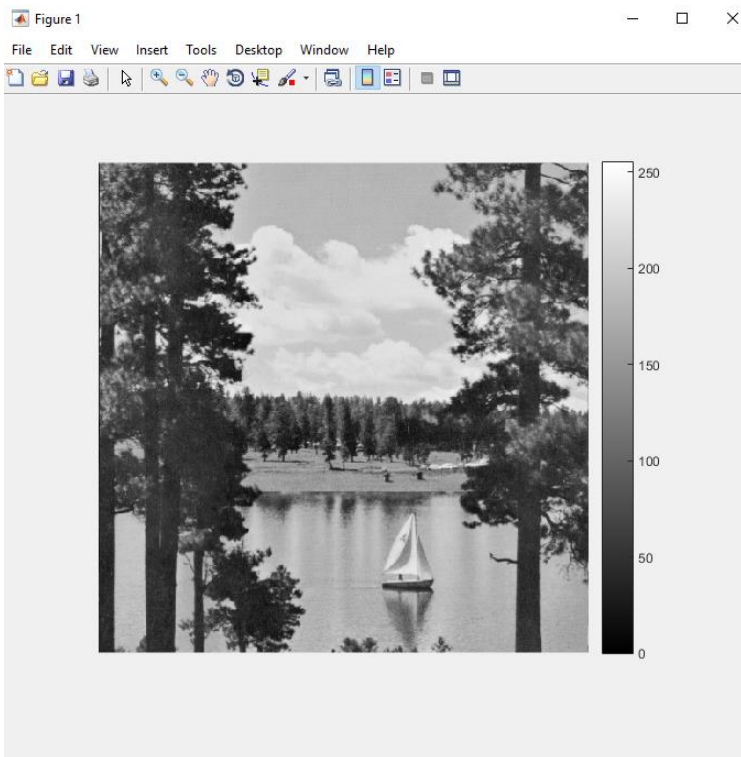
>> imshow(f);
>> imtool(f);
>> plot(f(128,:));
fx >>

```

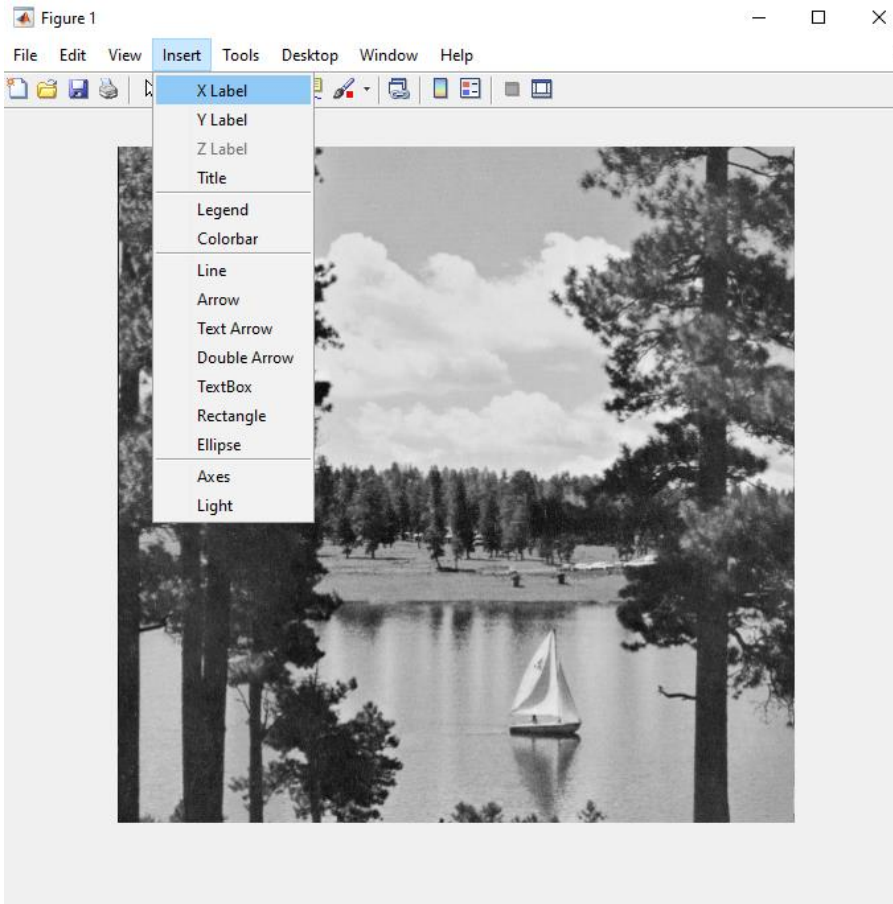
Running commands in the Command Window: **imread**, **imfinfo**, etc.



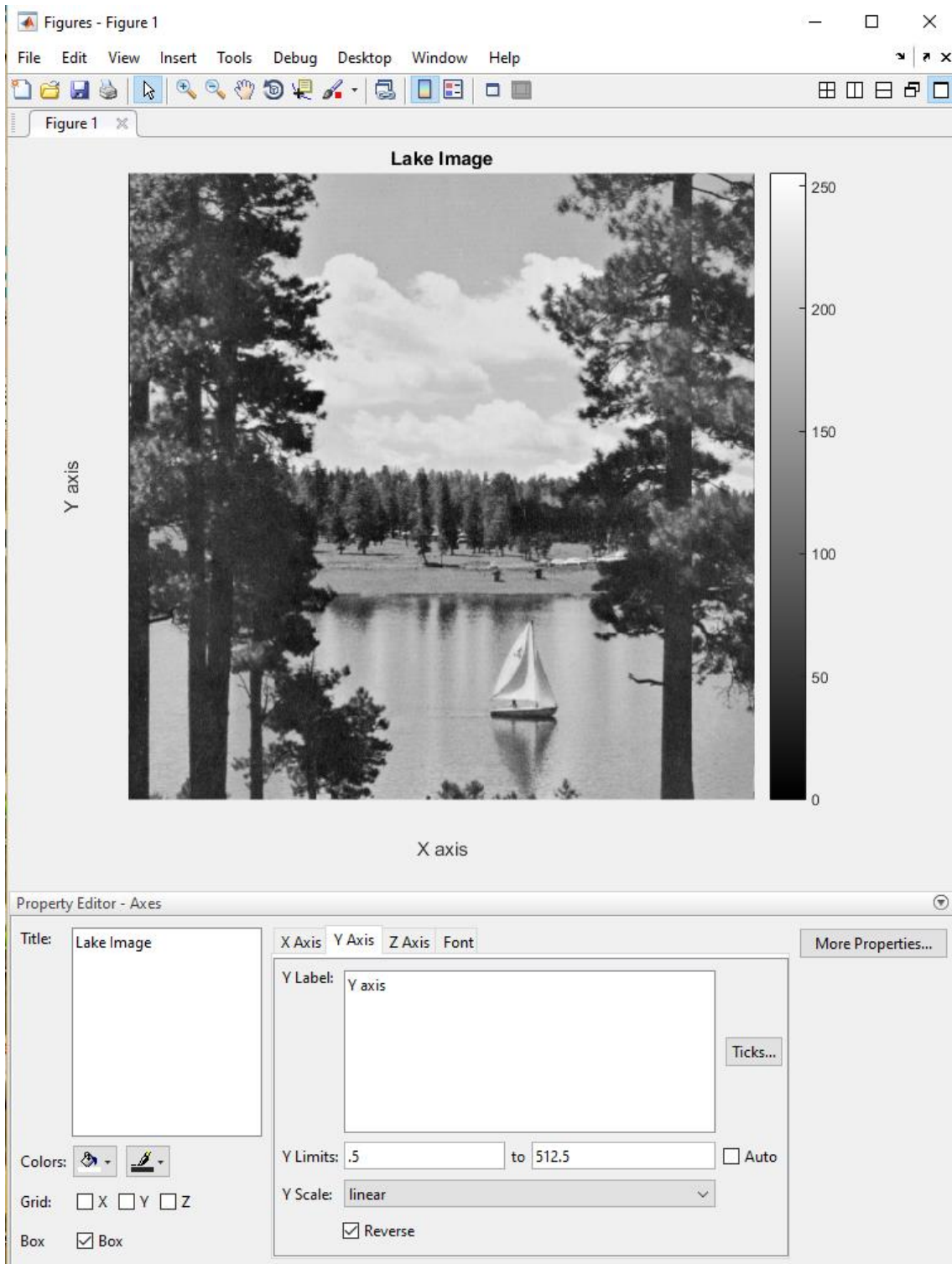
Raw output of **imshow**



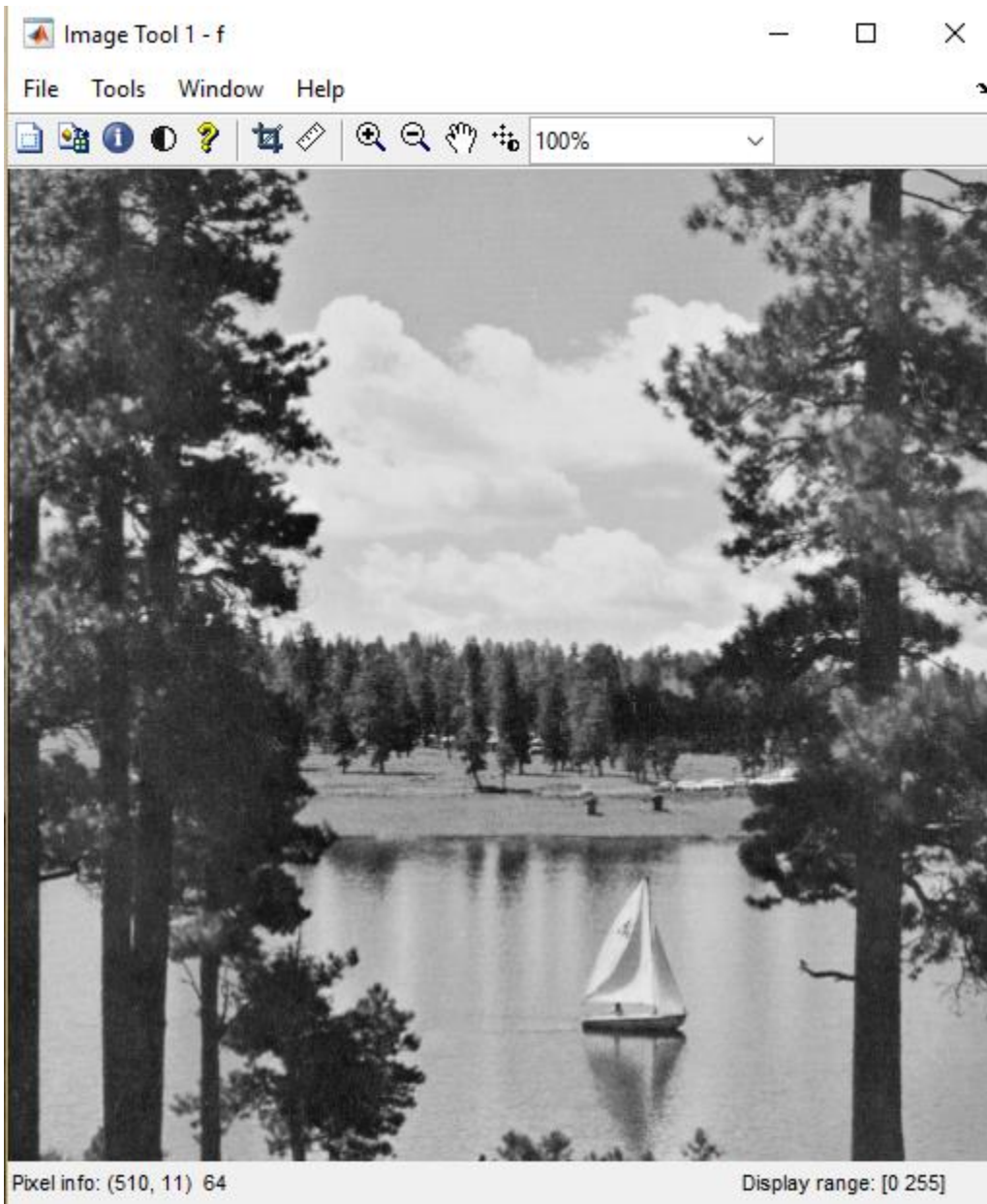
imshow showing gray-scale bar



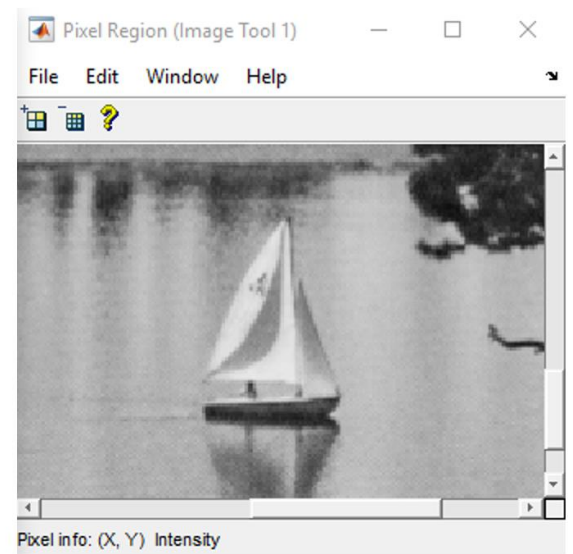
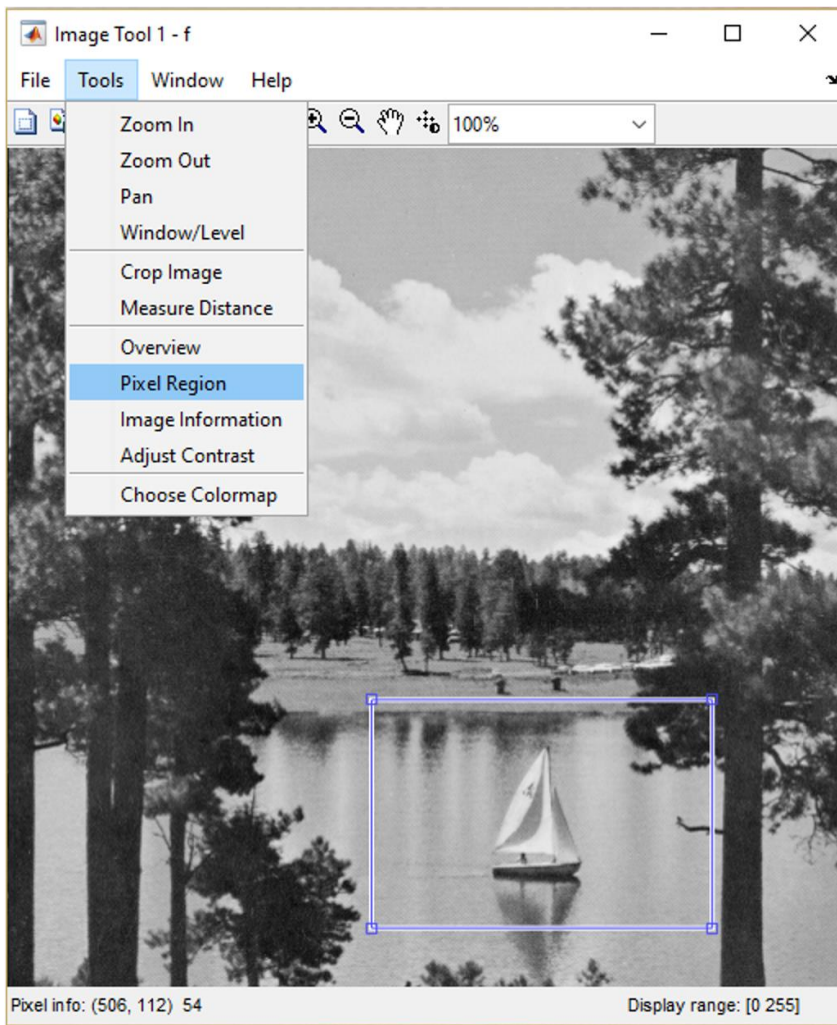
Various tools for annotating image in **imshow**



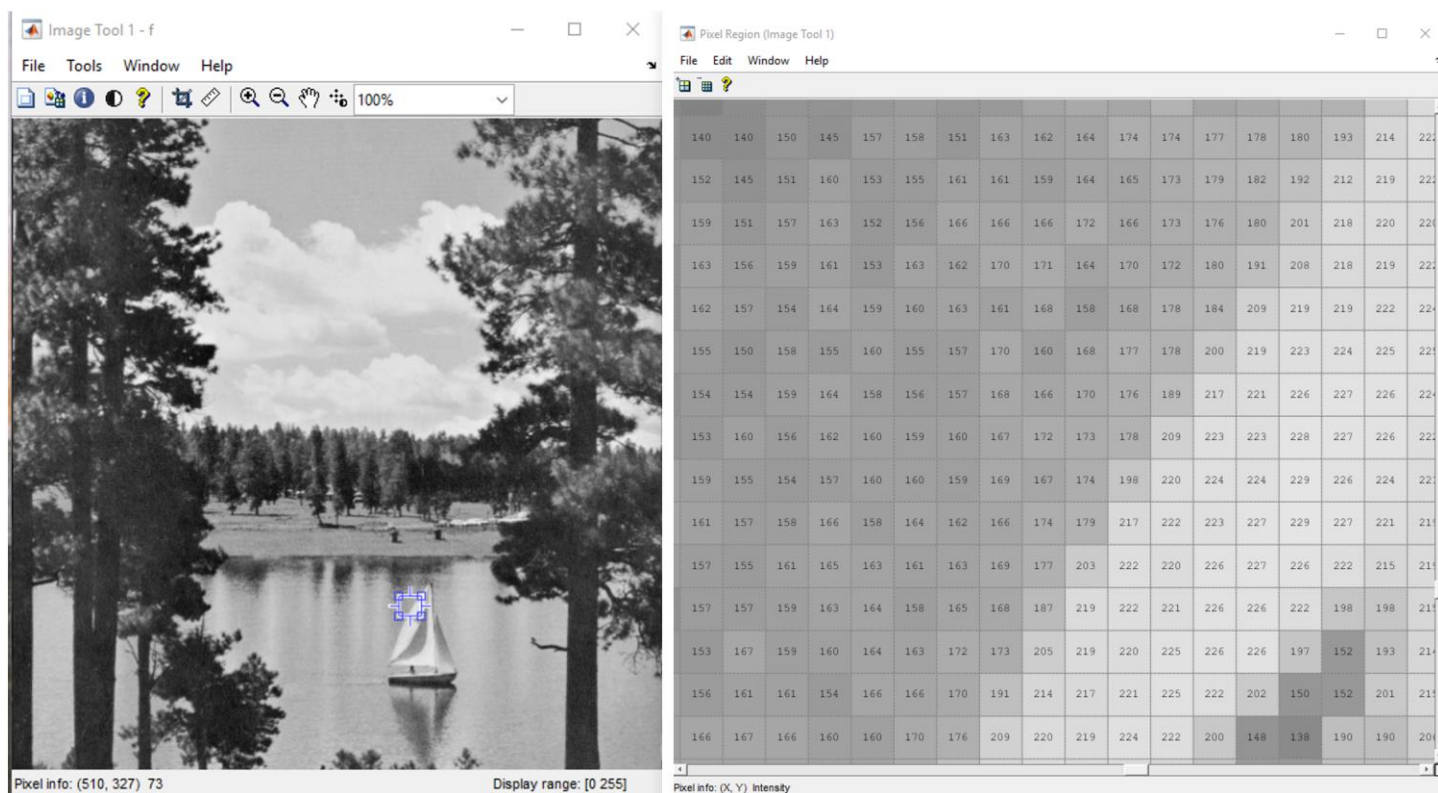
Plot options for **imshow**



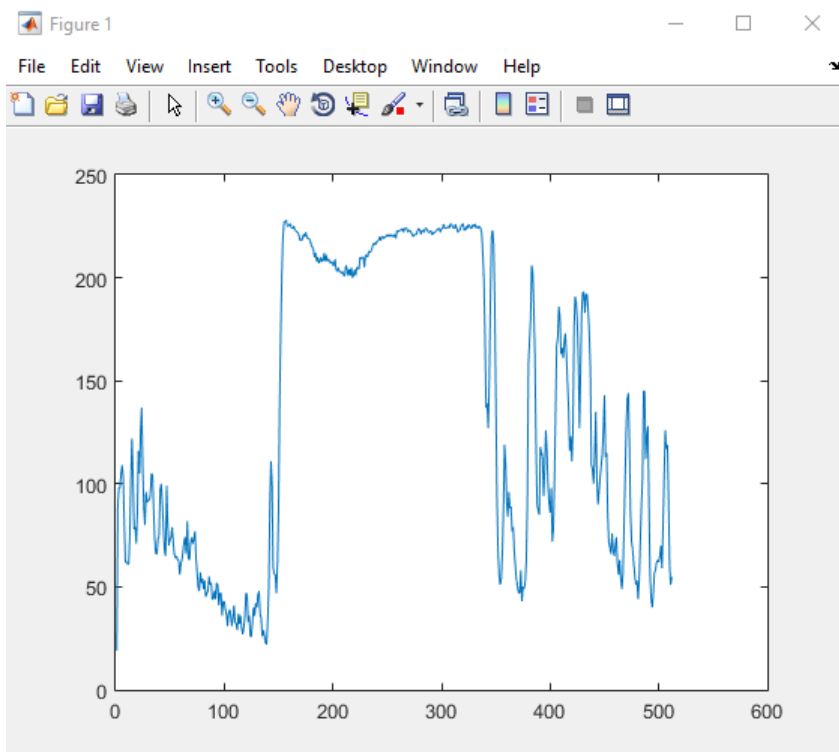
imtool for displaying images



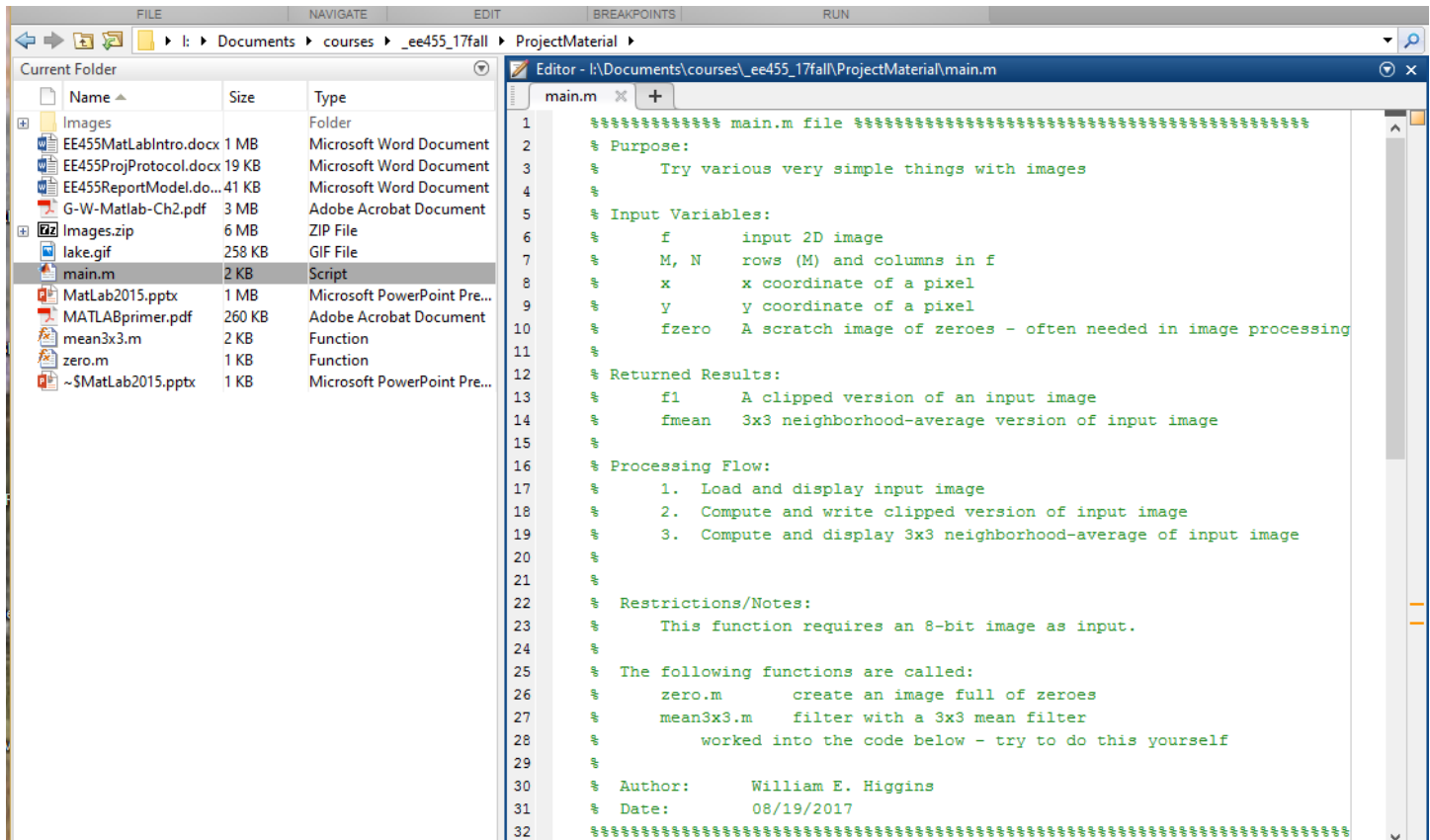
Selecting a pixel region and zooming with **imtool**



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 - imshow(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 = zeros(M,N);
65 - fmean = mean3x3(f,fzero,M,N);
66 - imshow(fmean);
67
68 %%%%%%%%%%%%% End of the main.m file %%%%%%%%%%%%%

```

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

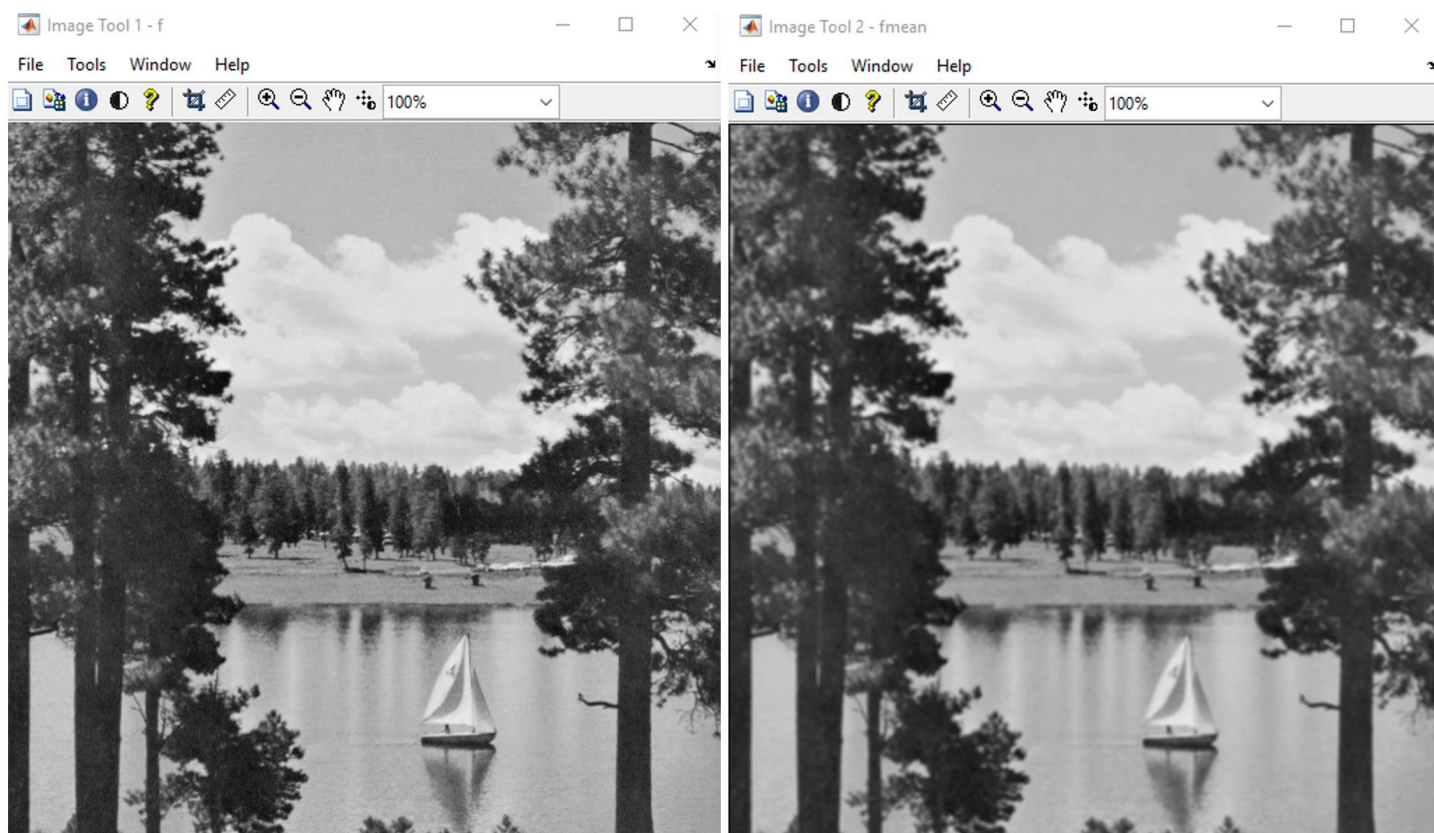
```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Function mean3x3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Purpose:
3  %   Compute a 3X3 mean (neighborhood average) filter at each pixel
4  %   in an image
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
32
33 function [mean] = mean3x3(f,fzero,M,N)
34
35 % Fill the output image with zeroes first
36 % (Step below is admittedly very cumbersome!)
37
38 for x = 1 : M
39     for y = 1 : N
40         mean(x,y) = fzero(x,y);
41     end
42 end
43
44 % Convert f to a 16-bit number, so we can do sums > 255 correctly
45
46 g = uint16(f);
47
48 % Define the coordinate limits for pixels that can be properly
49 %   processed by the 3X3 filter
50
51 xlo = 2; % Can't process first column
52 xhi = M-1; % Can't process last column
53 ylo = 2; % Can't process first row
54 yhi = N-1; % Can't process last row
55
56 % Compute the filtered output image
57
58 for x = xlo : xhi % Don't consider boundary pixels that can't
59     for y = ylo : yhi %   be processed!
60         for i = -1 : 1
61             for j = -1 : 1
62                 mean(x,y) = g(x-i,y-j) + mean(x,y);
63             end
64         end
65         mean(x,y) = mean(x,y) / 9.;
66     end
67 end
68
69 % Convert back to an 8-bit image
70
71 mean = uint8(mean);

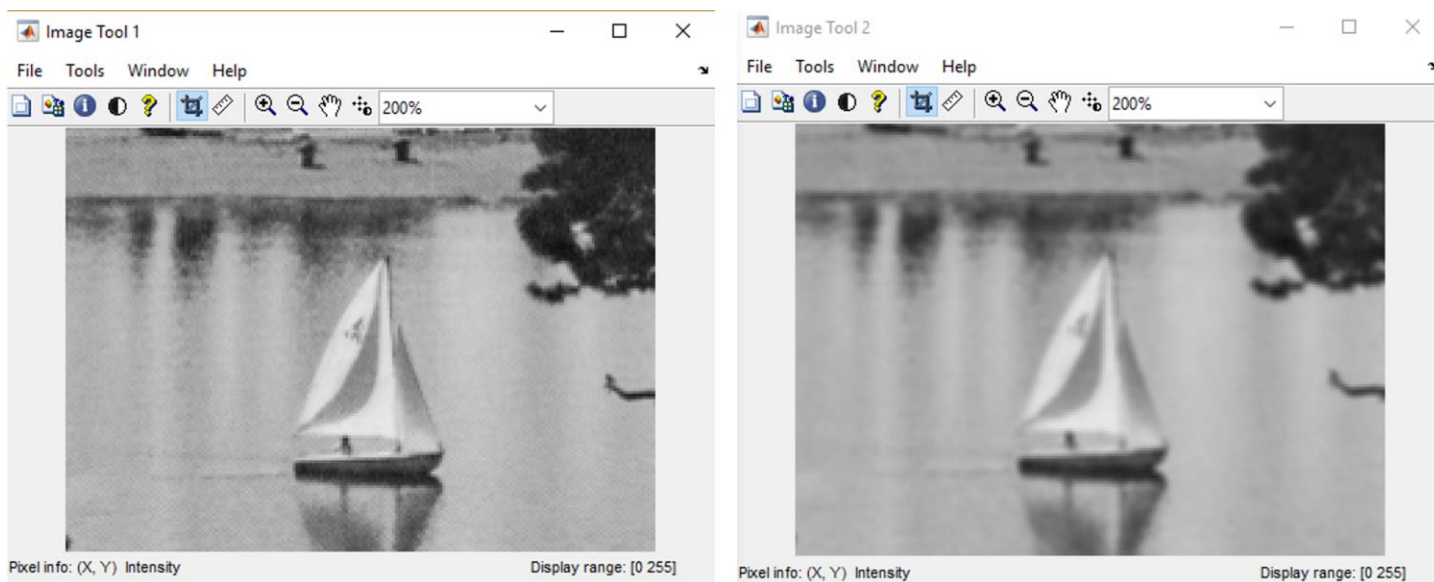
```

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 value 1
        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 ‘varargin’ and ‘varargout’ 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

***For 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.

