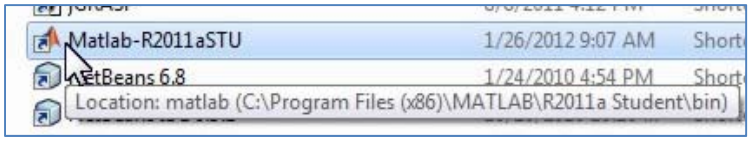
Matlab for CS6320 Beginners

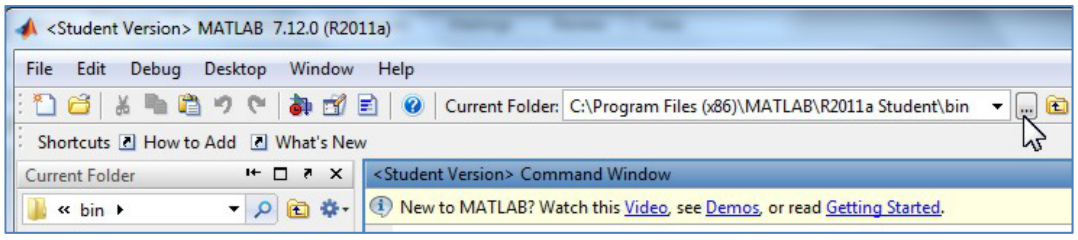
Basics:

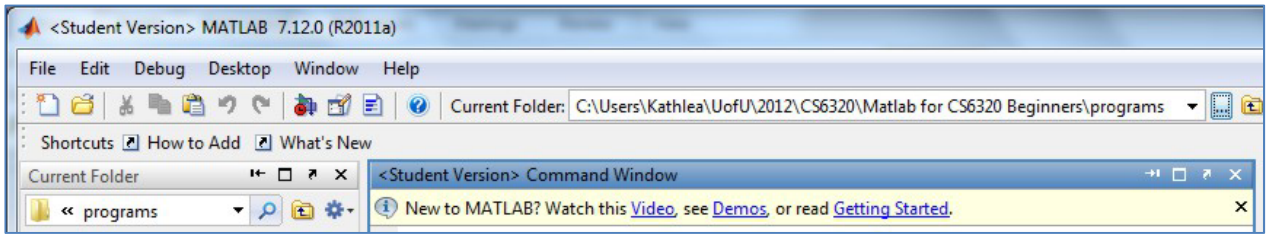
- Starting Matlab
  - CADE Lab remote access

  - Student version on your own computer

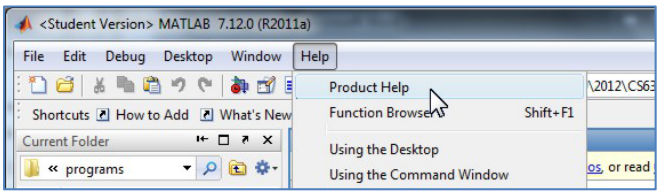


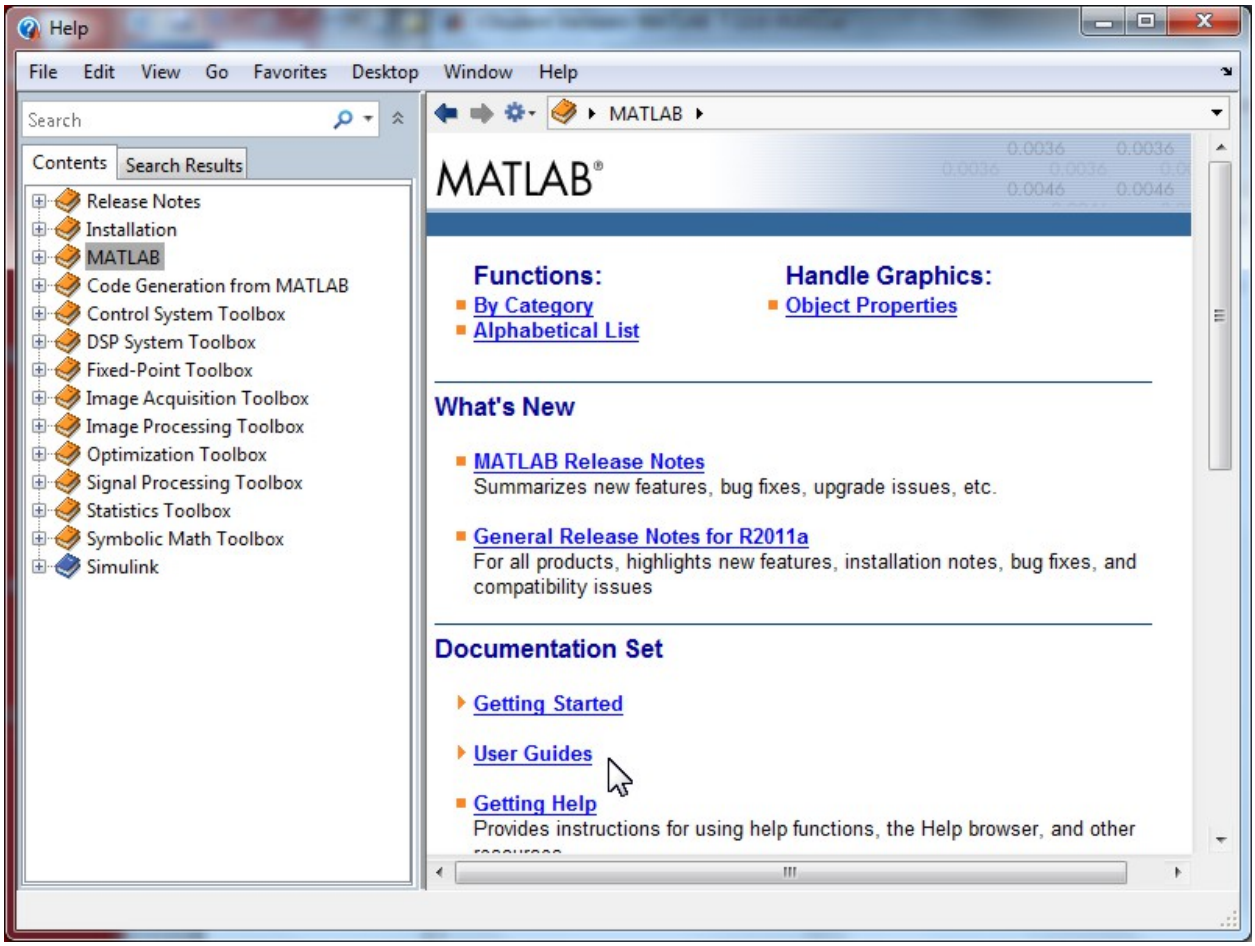
- Change the **Current Folder** to the directory where your programs, images, etc. will be stored





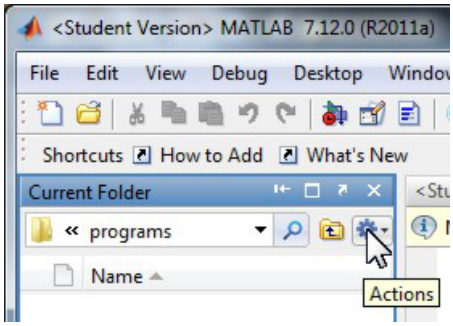
- Getting help from Matlab(primers, tutorials, online help)

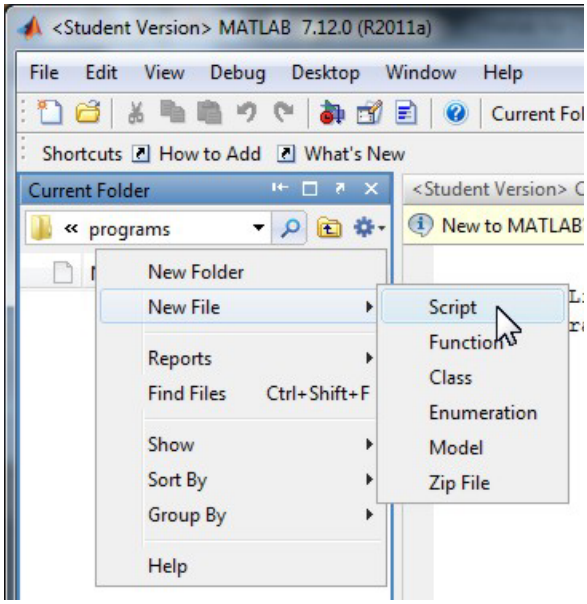




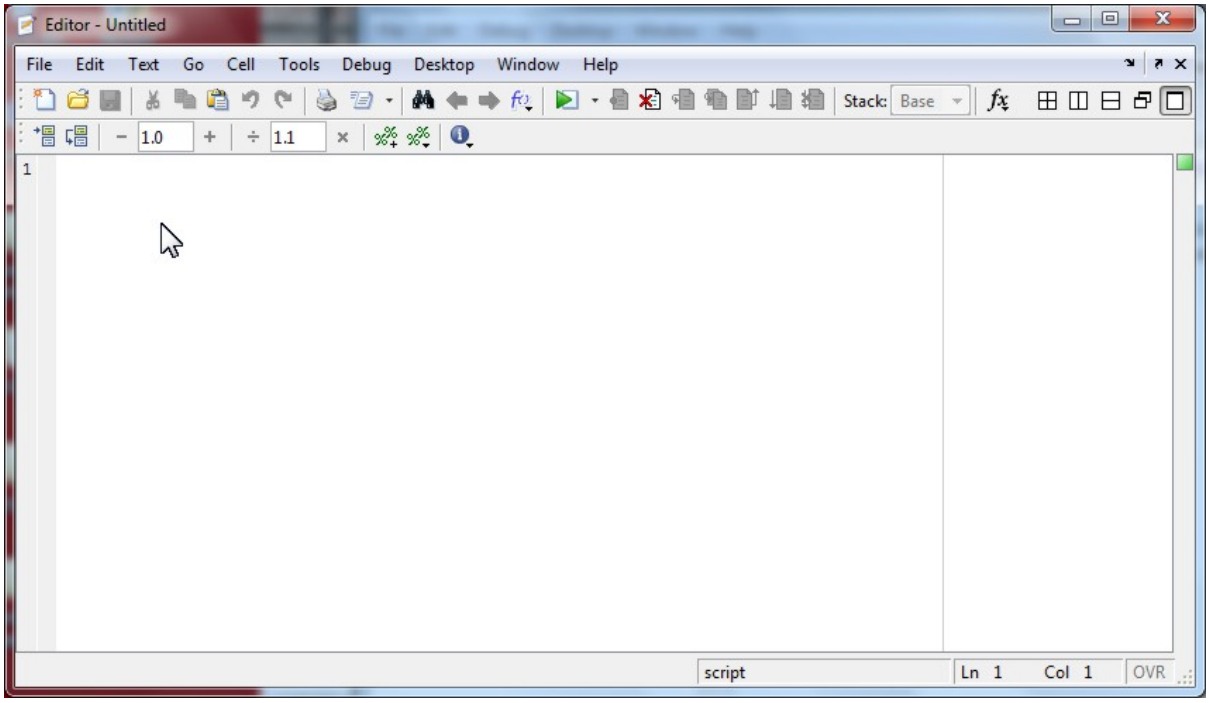
  - If you are a Matlab beginner you can click on **Getting Started** for general help with how Matlab works or click on **User's Guide | Programming fundamentals | Syntax Basics** for basic information such as how variables are created and initialized in Matlab.
  - An excellent description of Matlab expressions can be found in **Getting Started | Matrices and Arrays | Expressions**. It points out the fact that Matlab stands for "Matrix Laboratory". **Whenever possible use a matrix expression** instead of a `for` loop to make matrix calculations. These expressions will execute much faster than nested `for` loops, because Matlab is optimized for manipulating matrices.

- Creating, writing and running programs (m-files)
  - You can run commands in the **Command Window** to try out how they work
  - For automatically running commands, create an m-file. This is the "source code" for MatLab programs. m-files are interpreted programs, called scripts, and are not compiled before running. Remember: Matlab is a "Computational Program", not your usual programming language.
  - m-files can be created in two ways: type `edit` in the **Command window** or click on the Actions icon in the **Current Folder** window and select **New File | Script**:

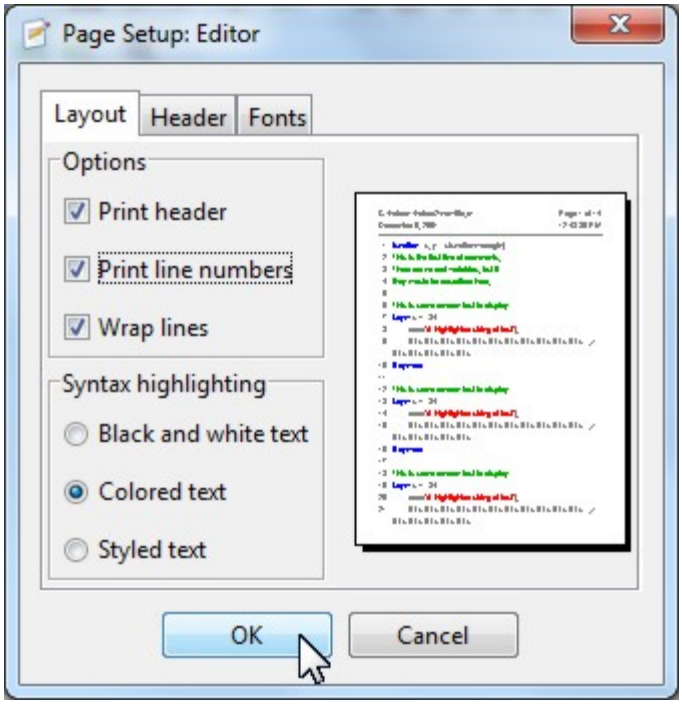




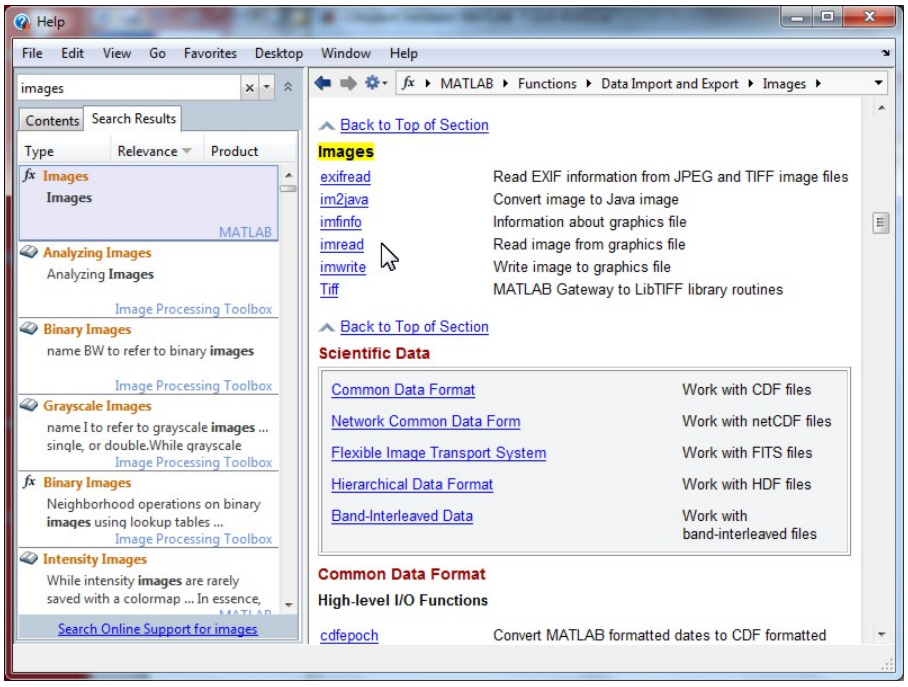
  - The **Editor - Untitled** window will pop up (when you save it you can give it a name):



- If you want to print your script, you can set the page layout to print line numbers for readability

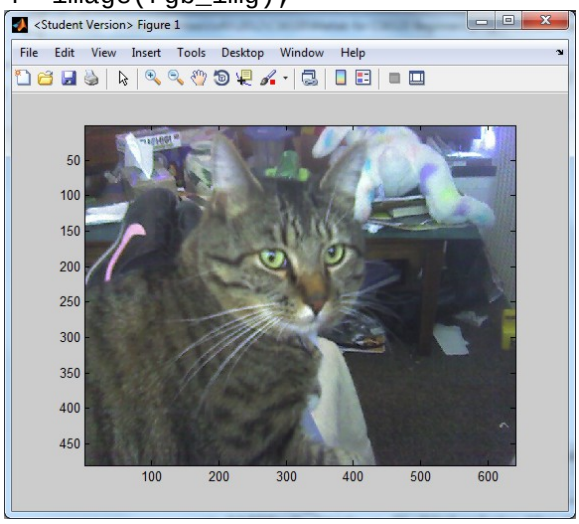


- Reading and writing images (matrices of pixel data)
  - type images in Search box and click on first result labeled as a basic MATLAB feature available without needing the Image Processing Toolbox



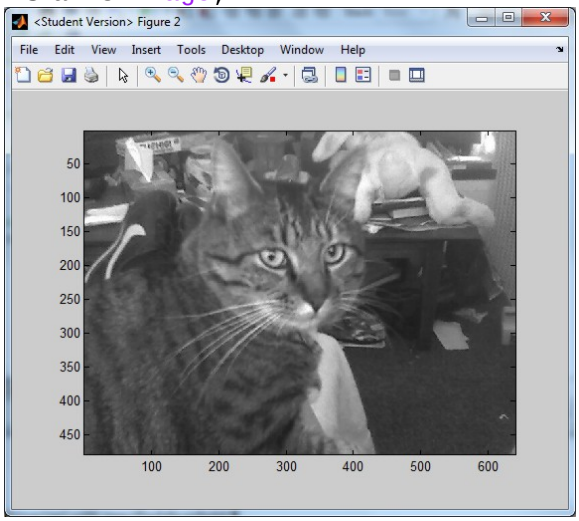
- reading images and displaying them (note the use of the semicolon to suppress command window echo). These are 3D matrices where each pixel is a 3-element vector:

```
clear all; close all; clc;
%load an image and display it
rgb_img = imread('Photo_062011_002.jpg');
image(rgb_img);
```



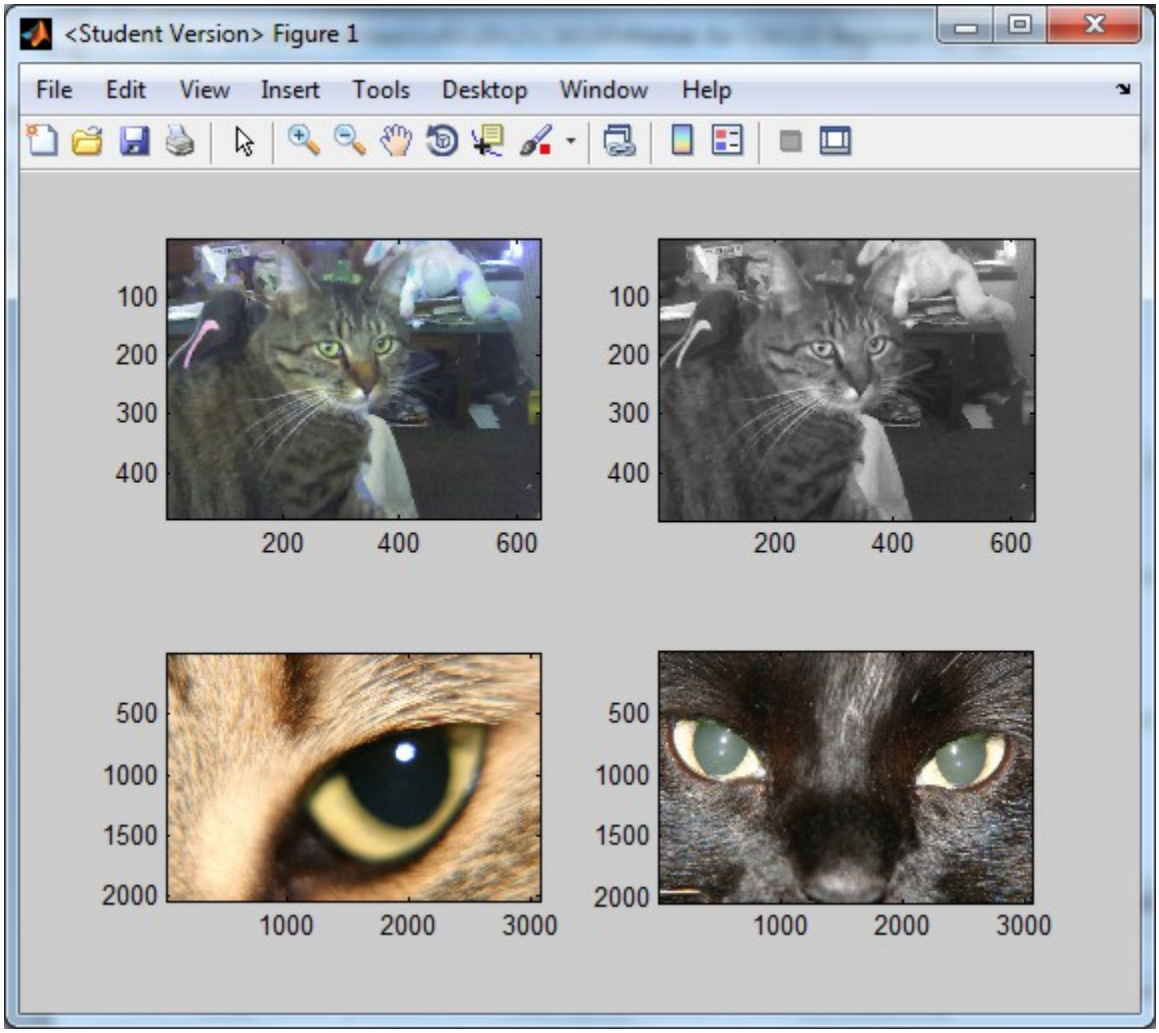
- Convert color to gray scale
  - Find help: User's Guide | Graphics | Displaying Bit-mapped Images | Working with 8-Bit and 16-Bit Images | Converting an 8-Bit RGB Image to Grayscale
  - Example:

```
clear all; close all; clc;
%load an image and display it in figure 1
rgb_img = imread('Photo_062011_002.jpg');
image(rgb_img);
%fit plot box tightly around the image data
axis image;
%Change image to grayscale 2D matrix
I = .2989*rgb_img(:,:,1)...
+.5870*rgb_img(:,:,2)...
+.1140*rgb_img(:,:,3);
%display grayscaled image in figure 2 with gray(256) colormap
figure; colormap(gray(256)); image(I);
axis image;
```



- Display images(multiple)
  - figure, image, and subplot

```
clear all; close all; clc;
%load an image and display it in first row, 1st column, figure 1
im1 = imread('Photo_062011_002.jpg');
subplot(2,2,1);image(im1);
%fit plot box tightly around the image data
axis image;
%Change image to grayscale 2D image
I = .2989*im1(:,:,1)...
+.5870*im1(:,:,2)...
+.1140*im1(:,:,3);
%display grayscaled image in 1st row, 2nd column, figure 1
subplot(2,2,2); colormap(gray(256)); image(I);
axis image;
%load another image and display it in second row, 1st column figure 1
im2 = imread('IMG_1766.jpg');
subplot(2,2,3); image(im2);
axis image;
%load 3rd color image and display it in second row, 2nd column figure 1
im3 = imread('IMG_1768.jpg');
subplot(2,2,4); image(im3);
axis image;
```



- writing images:

```
clear all; close all; clc;
%load an image and display it in figure 1
rgb_img = imread('Photo_062011_002.jpg');
image(rgb_img);
%fit plot box tightly around the image data
axis image;
%Change image to grayscale 2D matrix; note elipsis (...)
I = .2989*rgb_img(:,:,1)...
+.5870*rgb_img(:,:,2)...
+.1140*rgb_img(:,:,3);
%display grayscaled image in figure 2 with gray(256) colormap
figure; colormap(gray(256)); image(I);
axis image;
%write grayscaled image to new file
imwrite(I,gray(256),'grayStarbuck.jpg','jpg');
```

File: grayStarbuck.jpg

- Data types(discrete, conversion to float)
  - color images are 3D matrices, each pixel being a 3-element vector with integer data types (uint8 or uint16)
  - images converted to grayscale are 2D matrices, each pixel is an integer which indexes to a grayscale color map when it's displayed
  - conversion of a grayscale image to single-precision, floating-point value (single):

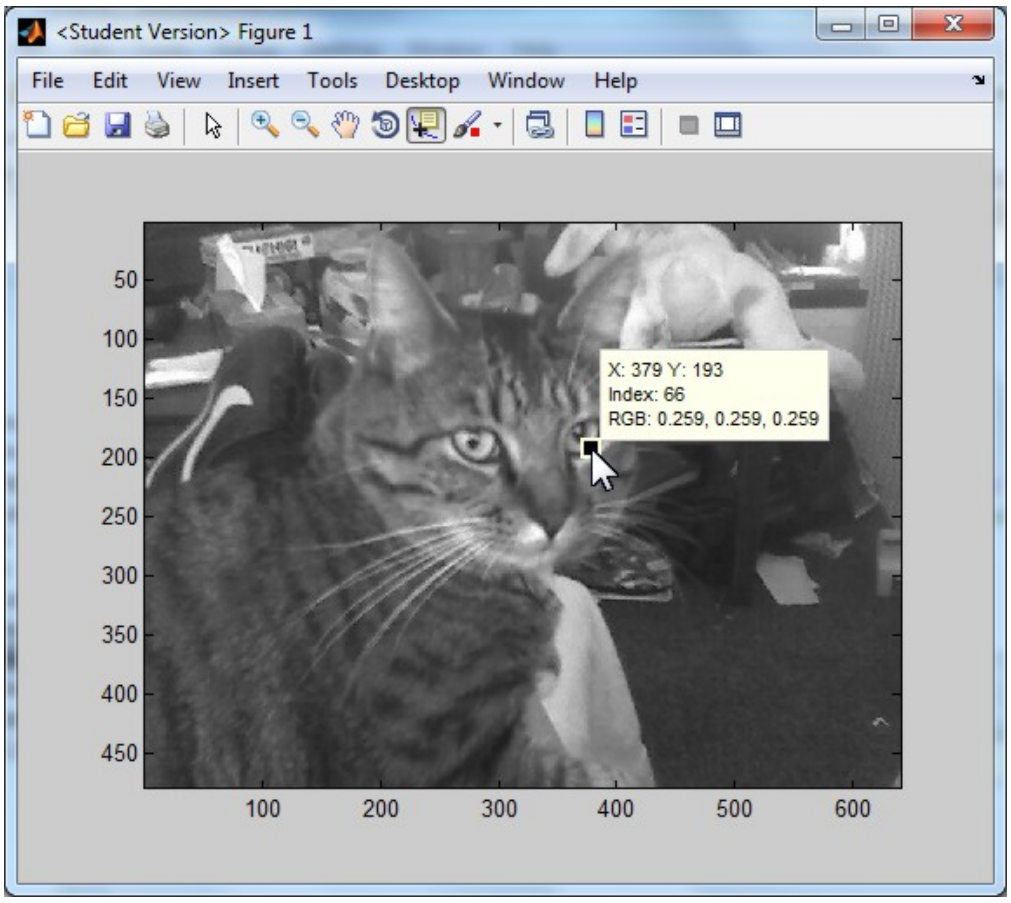
```
clear all; close all; clc;
%load an image and display it in figure 1
rgb_img = imread('Photo_062011_002.jpg');
image(rgb_img);
%fit plot box tightly around the image data
axis image;
%Change image to grayscale 2D matrix; note elipsis (...)
I = .2989*rgb_img(:,:,1)...
 +.5870*rgb_img(:,:,2)...
 +.1140*rgb_img(:,:,3);
%display grayscaled image in figure 2 with gray(256) colormap
figure; colormap(gray(256)); image(I);
axis image;
whos I
%convert grayscale to single in [0,1) range
S = single(I)/255;
whos S

```

Command Window

```
  Name        Size                Bytes  Class    Attributes

  I         480x640             307200  uint8

  Name        Size                Bytes  Class    Attributes

  S         480x640            1228800  single
```

- Read cursor position and values (manual definition of features)

```
clear all; close all; clc;
rgb_img = imread('Photo_062011_002.jpg');
I = .2989*rgb_img(:,:,1)...
+.5870*rgb_img(:,:,2)...
+.1140*rgb_img(:,:,3);
%display grayscaled image in figure 2 with gray(256) colormap
fig = figure; colormap(gray(256)); image(I);
axis image;
%get data object
dcm_obj = datacursormode(fig);
%enable cursormode
set(dcm_obj,'DisplayStyle','datatip',...
'SnapToDataVertex','on','Enable','on')
%prompt user
disp('Click on the image to display a data tip, then press Return.')
pause % Wait while the user does this.
%save cursor position and display in Command Window
c_info = getCursorInfo(dcm_obj);
pos = c_info.Position

```



Command Window

```
Click on the image to display a data tip, then press Return.


pos =

   379   193
```

- Overlay gray scale image with processed binary (logical data type) image information(color overlay)

point operation:

- Thresholding an image (binarize) with specified threshold.

Filtering:

- Neighborhood filter:
    - Averaging (smoothing)
    - Edge detection(x,y derivatives, magnitude -> edge image)
    - Application: overlay binarized edge image with original