General Instructions

Due Date:
Saturday, April 22 at 11:00pm (submit via blackboard)

Assignment Summary Instructions:
This assignment has two problem with several tasks. You will use MATLAB as a tool to solve the problems for the given test cases provided and that it is flexible for any additional test case that might be used to evaluate your code.

- Problem #1: Sobel Edge Detection
- Problem #2: Where's Waldo?

Submission Instructions:
You will submit a **.zip file** that will contain the following files:

1. **.m file – Matlab file (separate sections using %% for each problem)**. Your final script should be in the zipped folder and named as **MA#_USERNAME.m (for example, MA9 _dwburles.m).**
2. .pdf file - scan of your algorithm sheet (use the template) for Problem #1
3. .pdf file - scan of your algorithm sheet (use the template) for Problem #2
4. Any required function files

Submit your **.zip file** to blackboard using the Mastery Assignment 1 link. Your final submission should be a zipped folder named as **MA#_USERNAME.zip (for example, MA9_dwburles.zip).** The file must be completely submit before 11pm on April 22 for credit. **No late work is accepted and will result in a zero on the assignment**. It is your responsibility to make sure the file is completely submitted prior to the deadline. You are provided 2 upload opportunities in case your first upload is incomplete or a mistake.

Academic Honesty Reminder:

The work you submit for this assignment should be your work alone. You are encouraged to support one another through collaboration in brainstorming approaches to the problem and troubleshooting. **However, all support should be only verbal in nature**. Sharing files and showing other students your file is considered physical and visual help and is considered an academic honesty violation.

Some examples of academic misconduct in ENGI 1331 include but are not limited to the following actions:

1. Picking up and using or discarding another student's written or computer output;
2. Using the computer account of another student;
3. Representing as one's own the work of another on assignments, quizzes, and projects;
4. Giving another student a copy of one's work on an assignment before the due date.
5. Copying work from online resources (Chegg, google forums, etc.)
6. Posting work to online resources where other students can view your work

This assignment will be checked for similarity using a MATLAB code. The similarity code will check each submission for likeness between other student submissions, past student submissions, the solution manual, and online resources and postings. If your submission is flagged for a high level of similarity, the ENGI 1331 faculty will review the files, and then the guilty parties will be turned in for an academic honesty violation if deemed appropriate.

NOTE: Since this is an automated system for all sections, if any of your work is not your own, you will be caught. Changing variable names, adding comments, or spacing will not trick the similarity algorithm and will result in a violation.

Problem #1: Sobel Edge Detection

Background:

Many applications in engineering and science require the correct identification of edges. Many tools for automated manufacturing processes are equipped with cameras to detect markers (such as a thick black line) that might designate special reference points or physical locations. An edge-detector can process and extract relevant features in a set of images before they are fed into a pattern recognition algorithm, which can result in superior performance.

For this problem, you will find the edges of cholesterol crystals, which cause gallstones and atherosclerosis in the human body. For drugs to prevent crystallization of these maladies, researchers need to determine how pharmaceuticals affect crystal growth by examining sizes and shapes.

Another application relates to pattern recognition. In robotics, a common feature is for a robot to behave a certain way in response to visual cues or special markers.

The following are two such kernels for detecting horizontal and vertical edges, together called the Sobel operator.

-1	-2	-1
0	0	0
1	2	1

Horizontal

-1	0	1
-2	0	2
-1	0	1

Vertical

We notice that the kernels are vertically and horizontally symmetric with the sum of the kernel elements being zero. This means that as the kernel passes (slides) over the image, the vertically or horizontally aligned pixels that differ in intensity from their neighbors will be multiplied with the non-zero parts of the kernel, which results in a non-zero pixel. Edges which are nonzero should therefore appear white and non-edges close to black.

****Note: The effective luminance of an image is a single component of the YUV color space, which is used for color television transmission signals. The luminance component is the Y, and it represent the black and white signal. The UV components are the chrominance.**

Task 1: Load the image crystals.jpeg and save as ***crystal***.

In order to perform the sobel kernel operation, we must first convert the image to grayscale. A grayscale image is a single layer matrix that represents a measure of the effective luminance of the original image. The formula for converting an image to grayscale is as follows:

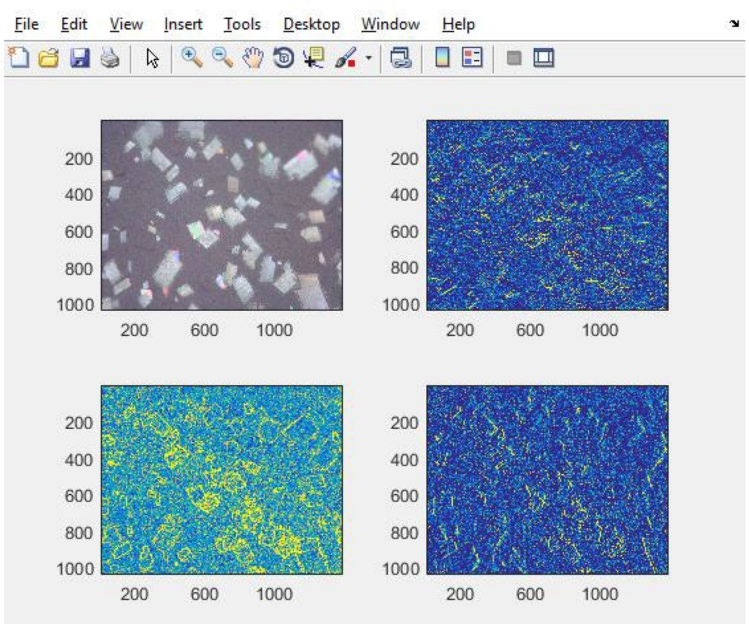
intensity = 0.2989*red + 0.5870*green + 0.1140*blue

Task 2: Create the two new matrices defined as the horizontal and vertical sobel kernels using the exact matrices shown in the background for this problem.

Task 3: "Convolve" the new image with the sobel kernels using the command conv2(double(arg1), double(arg2)) where arg1 is the crystal image and arg2 is either the horizontal or vertical kernel. You will execute this command twice creating two new matrices called ***Horizontal*** and ***Vertical***, once for the horizontal kernel, and once for the vertical kernel respectively. For more information on convolving, please visit: http://matlabtricks.com/post-26/tutorial-on-2d-convolution-of-images.

NOTE: For this function to work properly, you will have to type cast your arguments as class double. For more information of MATLAB Number Classes, see http://www.mathworks.com/help/matlab/matlab_prog/fundamental-matlab-classes.html. Chances are your MATLAB values are already in class double, but perform specified operations just in case they are not.

Task 4: Display the original image, the normalized vertical edges, the normalized horizontal edges, and the combined vertical and horizontal edges as shown below



(Top Left): Original Crystal Image

(Top Right & Bottom Right): The horizontal edges (***Horizontal)*** and the vertical edges (***Vertical***) separately using the image(arg1) command. Arg1 should be the absolute value of the results found using the conv2() command

(Bottom Left): Normalized vertical and horizontal edges combined using the image(arg1) command. This time, your equation for arg1 will be: (X.^2+Y.^2).^0.5
with X being the convolution results for the horizontal component and Y being the

convolution results of the vertical component.

Problem #2: Where's Waldo?

Problem Statement: The goal is to write a function that finds Waldo (image provided) in a larger image. The inputs into the functions should be the Waldo image matrix and the larger image matrix. In your script, prior to sending the image matrices into your function, the program should check to make sure that neither dimension of the Waldo matrix is larger than the dimensions of the background matrix. If this is the case the script should end and xshow an error message. Otherwise, the function should output a final image matrix that includes a red rectangle around Waldo as well as the range of pixels where Waldo was found. Your program should then display the image with the locations of where the Waldo matrices were found and output the pixel range as shown in the sample output. NOTE: The sample output uses P2_sampleWaldo.jpg and P2_sampleBackground. Two other sets are provided for you to test your code. Make sure not to get tricked with similar Waldo characters shown in the waldoLookALikes.jpg .

Coding Requirements:

- Error when the Waldo matrix is larger than the background image.
- You must use explicit loops to solve this problem, **NO IMPLICIT LOOPS ARE ALLOWED. This includes functions such as find and use the colon (:) operator.**
    - o You cannot use any built-in image functions to match Waldo in the larger image. You must manual evaluate the two matrices to find the match using loops.
- Your function should have two inputs: Waldo image matrix & Larger image matrix
- Your function should output the Final image matrix with Waldo boxed in and the range of pixels Waldo can be found. The range can be passed back to the script from the function in any form you choose.

<u>**Sample Output Image and Command Window Below**</u>