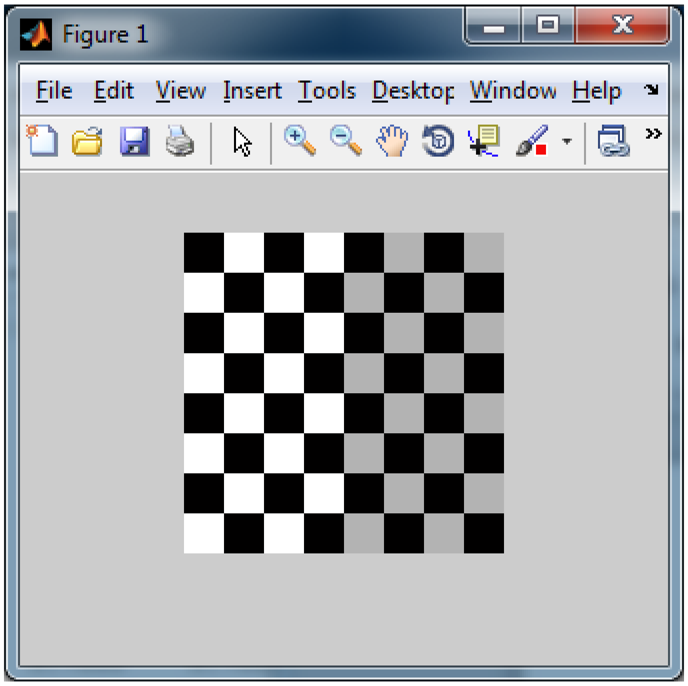
Computer Vision Lab: Edge Detection

CSCI380 Computer Vision

Part A

1. First thing we're going to do is learn how to create a sample image that will be useful for this lab. We will do this by using the Matlab command 'checkerboard'.
2. To give you an idea on how this command works, from the matlab command prompt, type the following:
    a. imshow(checkerboard(20));
3. It should create the image shown below:



4. The 20 in the command is the number of pixels that each box on the checkerboard will represent.
5. Let's create an image from this matrix that we can use later. In order to do that, we need to use the 'imwrite' command.
    a. imwrite(checkerboard(20), 'checkerboard.jpg', 'JPEG');

6. To verify the above worked correctly, let’s read in the newly made image and display it:
    a. myImage = imread(‘checkerboard.jpg’);
    b. imshow(myImage);

7. Now that we have a nice image to use for our edge detection algorithm, let’s begin to implement it. First thing we want to do is create our x and y gradient filters:

    Sx = [1 0 -1; 2 0 -2; 1 0 -1];

    Sy = [1 2 1; 0 0 0; -1 -2 -1];

8. Convolve each of the above filters with the checkerboard image. Before we are able to use the convolve function we must convert our image into a double. We do that by using the ‘double’ command and passing it image variable name, in this case ‘myImage’.

```
sobelXImage = conv2(double(myImage), Sx, 'same');

sobelYImage = conv2(double(myImage), Sy, 'same');
```

9. Let’s go ahead and take a peek at what our images currently look like:

```
figure('Name', 'X gradient');
imshow(uint8(sobelXImage));


figure('Name', 'Y gradient');
imshow(uint8(sobelYImage));
```

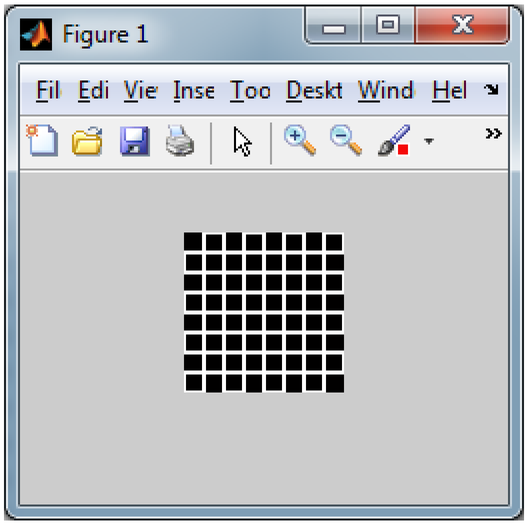
10. Do the gradient images look like you expect them too? Why or why not?

11. What is missing from the above question and how would you fix it?

12. Now that we understand what the gradients are doing and are able to display them properly, let’s continue on with our edge detection. We now need to square each of the X and Y portions, and then take the square root of them to determine the magnitude of our gradients.

$$E(u,v) = \text{sqrt}(\ (D_x(u,v))^2 + (D_y(u,v))^2\ )$$

13. Enter the appropriate matlab command for the above formula and display the image. You should see something like:



14. Each of the edges are shown in white.

Part B (turn-in)

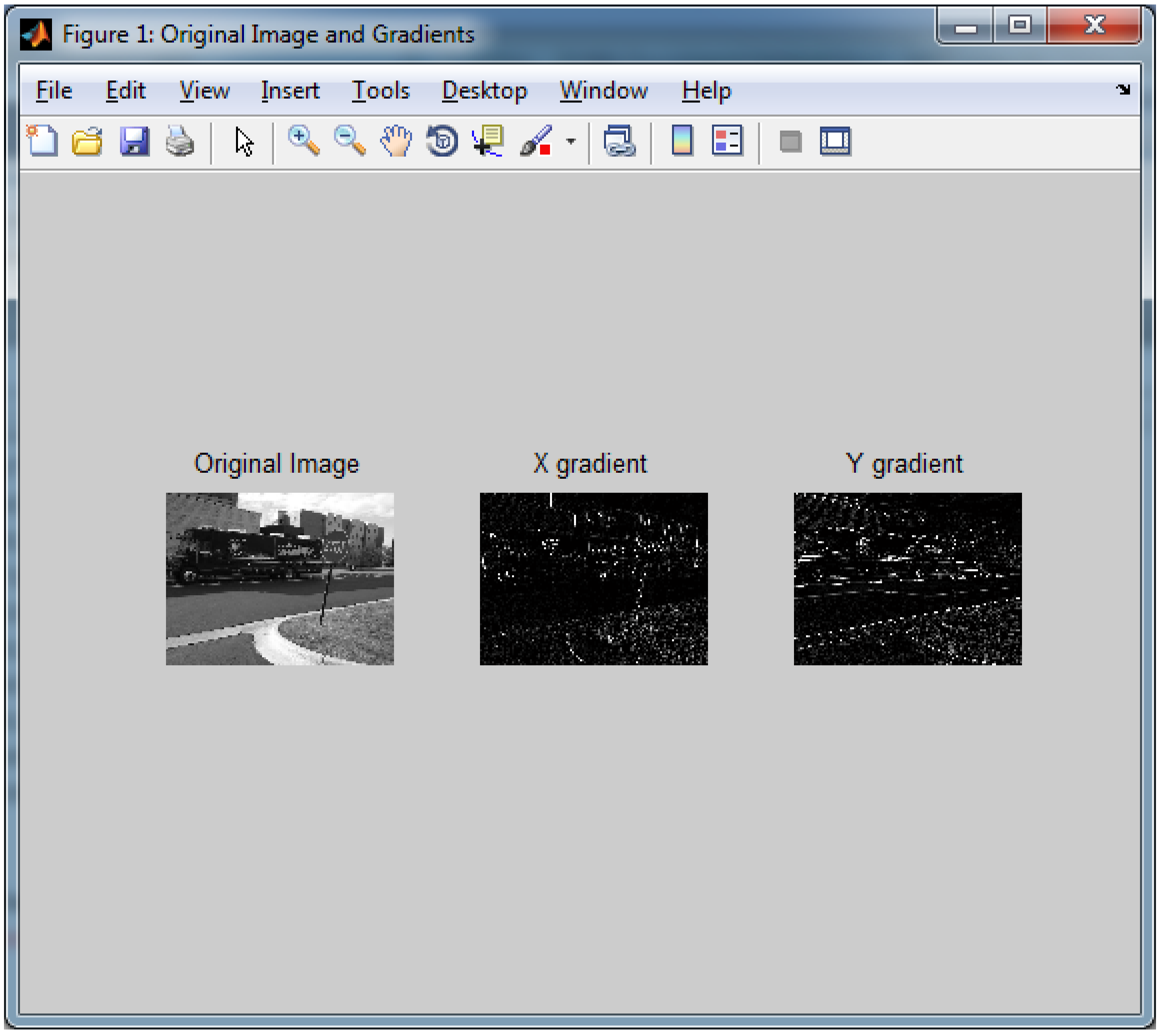
15. Download the sample image that comes with the lab (DU and Kent Mobile.jpg). Repeat the above steps and display the original image, the X gradient, and the Y gradient on one form/figure (Note: Don't forget to convert it to black and white first) .

16. After we sum the squares and take the square root we have the magnitude of our edge strength. We may not want to look at every single edge detected. We can limit which edges we want to see by applying a threshold. We can do this by using the following matlab commands:

```
threshold = 50;
edgeDetectedImage = (uint8(edgeDetectedImage) > threshold) * 255;
```

17. Create 4 additional figures each with their own respective image on it (using threshold {50, 100, 150, 200}). Make sure the titles of the figures reflect their contents.

18. Turn in the final matlab script that you used to accomplish part B. It should display 5 figures, the first of which will look like so:



Additional Exercises for fun if you complete on time.

1. Repeat the above steps using the improved Sobel Edge Detector
2. Research additional edge operators. Implement them and compare/contrast their results