Worksheet 2

This worksheet is concerned with some basic image processing techniques. Review the first worksheet and make sure you are familiar with the techniques it introduced before you start this one. Make sure you understand, for example, what the colon operator does.

Start by setting the Matlab path and reading in the chessboard image as in the last lab class.

1. **Check out the operations from the last lecture**. Try out some of the examples from the last lectures (Lecture 4, 5, and 6), and check that you can get the same results, and that you understand what is happening.

2. **Understand the histogram equalization**. Check Recap of Lecture 5 (page 3 and 4), can you translate the pseudocode in page 4 into a Matlab code?

3. **Develop your edge detector**. Write a Matlab script that reads in an image and does the following operations:

   a. horizontal differencing on the original image

   b. thresholding the difference array at a level that picks out the main bright edges

   c. thresholding the difference array to pick out dark edges

   d. combining b and c into a single array with all the vertical edges

   e. operations a-d but for vertical differencing

   f. combining d and e into a single array with all the edges

   In operation d you might find the logical operator | (or) useful. In operation f, you will have to trim the images to be the same size.

4. **Convert your edge detector from a script to a function**. Look up "function" in Matlab help and make your script into a function. The function should not read an image from disk, but should take it, as well as the threshold value, as an argument. Its first line should be this:

   ```
   function newimage = edge1(image, threshold)
   ```

5. **Image smoothing**. Try the effect of replacing the subtraction in the image differencing operation with an addition. The effect is subtle and you might need to enlarge the output image on the screen to see it. Do this both for the horizontal and vertical directions, and then add the two results. To make the effect more obvious, repeat the operation starting from the previous output. This operation is an example of smoothing or blurring the image.

6. **Introduction to the convolution operation**. Lectures 5 and 6 introduced an important operation known as convolution. Try out a convolution function in Matlab so that you can see its effect. Try this:

```
mask = [-1 1];
newimage = convolve2(image, mask, 'valid');
imshow(newimage, []);
```

   Here, the matrix mask acts as a kind of template for the operation. The -1 and +1 next to each other specify horizontal differencing.

   The function `convolve2` is part of the local library of Matlab functions. It does the same job as the built-in function `conv2` mentioned in the toolbox documentation but it is often faster. Its arguments are an image matrix and a mask matrix, plus a string saying what to do at the edges. In the example, 'valid' means to make the result newimage one column smaller than the input image, so as to avoid having to guess any results. Now guess how to do vertical differencing using the same function and check your result. If you don't know how, review the lecture slide on how to build Matlab matrices. Try this

```
mask = [1 1; 1 1];
newimage = convolve2(image, mask, 'valid');
imshow(newimage, []);
```

   and compare the result with your smoothing operation.

7. **Experiment with convolution**. Try typing in some simple matrices and using them as convolution masks. Try, for example, masks that are all ones, but of different sizes, or different patterns of zero, +1 and -1. If you want ideas, ask the lecturer or discuss it with someone else in the class. Note that you can get an array of ones with the ones function, and you can switch rows and columns using m = m'; (the transpose operation).

8. Try the build-in edge detectors. One of the most common edge detectors is called the Canny edge detector. You can try this with

```
e = edge(image, 'canny');
```

   This uses smoothing, differencing, and local ridge detection. There will be more details in the lectures. Look up the documentation on the edge function, and try varying the arguments to the Canny method, and also try at least one of the other methods on offer, and compare the results.

   Note that you can get Matlab to put graphics in a new window with a call to `figure`.