Worksheet 1

This worksheet is to get you started with using Matlab for image processing. The goal is to introduce you to the basic operations that will allow you to get going, and exploring some ways to use the system.

You should follow the steps below, but feel free to explore, and ask the lecturer about anything you do not understand.

1. **Start Matlab**. You will find it in the All Programs menu (from the Start button). Click on MATLAB, then R2015a, then MATLAB R2015a.

2. **Check you can do calculations**. After a delay, the Matlab main window will open. Identify the command prompt ">>", click in its window if necessary, and type an arithmetic expression such as

   ```
   2 + 3
   ```

   (followed by "return" but without a semicolon at the end). Matlab should print the answer. You can use Matlab as a (very sophisticated) calculator. Although you will end up putting your programs into files using an editor, this facility for typing in code and having it executed immediately can be very useful for checking out your ideas.

3. **Add the local vision functions to the path**. You will need to be able to access functions written and stored locally. Matlab looks for functions in a list of directories called the path. To add the local material to the path for the current session, type this next to the command prompt:

   ```
   addpath T:\Departments\Informatics\ComputerVision\matlab
   ```

4. **Check out the Help system**. Give the command

   ```
   help teachimage
   ```

   This gives you quick and simple online help about a specific function in this case, teachimage. It works the same for Matlab's own functions and for locally-written functions. Now try

   ```
   helpwin
   ```

   This gets you into Matlab's full online help system. You can look up information about every part of the system. Start by expanding MATLAB in the Contents tab, then click on the "Getting Started". You can explore the basic system in a variety of ways from there. Finally, click on "Image Processing Toolbox" in the Contents tab, then go to the "Getting Started" part of the documentation set. This gives you a good introduction to Matlabs image processing capabilities.

5. **Read in an image**. Now you can get an image from disc into main memory. Matlabhas lots of options for how the image is stored, but we want to use a simple 2-D array of grey-levels, each in the range 0.01.0. The local function teachimage knows where to look for images, and how to read them into arrays in this format. To get the image, type the command

   ```
   im = teachimage('chess1.bmp');
   ```

   Note that the semicolon at the end is needed, otherwise Matlab will print a very large number of grey-level values onto your screen. The variable im has been automatically declared, and refers to the matrix holding the image data.

6. **Look at the image**. Give the command

   ```
   imshow(im);
   ```

   Matlab will make a new window and display the image in it.

7. **Relate the data to the displayed image**. Give the command

   ```
   size(im)
   ```

   (with no semicolon). This will tell you the number of rows and columns in the matrix `im`. Look at which is greater, and relate this to the shape of the image ("landscape" or "portrait") on the screen. In the image window, you will find an icon with a small yellow box and a "+" sign. If you put the cursor over it you will see it is labelled "Data Cursor". Click on it; you can now click on the image and find out the coordinates and grey-level of any point by moving the data cursor around. The data cursor shows the grey-level as "index". Do this for some point, and then check that you can tie this in with the values stored in the matrix in memory. For example

   ```
   im(100, 200)
   ```

   will print out the grey-level of the pixel at row 100, column 200. The data cursor reports X and Y, not row and column, so make sure you understand how these are related.

8. **Try some image manipulation**. First make a copy of the image, to avoid having to read it in again if you make changes:

   ```
   imnew = im;
   ```

   Change some of the values in this matrix, and redisplay it. For example, you can set all the pixels in row 100 to a grey-level of zero like this

   ```
   imnew(100, :)  = 0;
   ```

   (Here, the colon ":" means "all columns".) Think about what effect you expect this to have, then redisplay the image using `imshow` to see if you were right. Try making some other changes to the image. Note that in Matlab arrays are copied on assignment, which means that the original image, `im`, is not changed if you update `imnew`.

9. **Try image thresholding**. Matlab operators like "greater than" will operate on whole arrays, working on each element. That means that if you try typing

   ```
   t = im > 0.5;
   ```

   you will get a new matrix `t` which is has 1 wherever `im` was greater than 0.5, and 0 everywhere else. Try this, display `t`, and think about whether this operation might be useful in any applications.

10. **Try some image processing**. The following code will carry out the operation of "horizontal differencing" on the image.

    ```
    d = im(:, 2:end) - im(:, 1:end-1);
    ```

    To display `d`, you need to tell Matlab that its grey-levels might not be in the normal range, and it should look for the smallest and largest values in d and map them to black and white respectively. This is quite easy: just give the command:

    ```
    imshow(d, []);
    ```

    The extra argument (an empty matrix, shown as []) just tells Matlab to figure out the grey-level range of `d` rather than assuming it's $0.0 - 1.0$. Try to understand the code that calculates `d`. It uses a powerful Matlab operator, the colon, which expresses a range of values. This has been discussed in the Matlab lecture; also look up "colon" in the Matlab Help window.

11. **Exercise**. Figure out how to do "vertical differencing" as opposed to "horizontal differencing" on images, using code similar to that at **step 11** above. Apply it to the chessboard image and compare the results.