Worksheet 4

If you have not completed the first three worksheets, please ensure that you go through them before going further.

- Start by setting the Matlab path and reading in the chessboard image as before. If you are tired of looking at the chessboard image you can experiment with other images try replacing 'chess1' with 'maze_left', 'edin_lib', 'moffat' or 'head' in the call to `teachimage`.

1. **Check out findpeaks**. One common operation that is useful is finding local maxima in an array. A local maximum is a place where the value stored is greater than all its neighbouring values. A function for doing this, `findpeaks`, is provided in the local library. First, try it out on a simple example like this (which you can cut and paste from here into Matlab):

   ```
   im = [0 0 0 1 1 0 0; 0 0 1 1 2 1 0;...
   0 4 1 1 1 1 0; 0 1 1 1 0 0 0]
   [rows,cols,vals] = findpeaks(im)
   ```

   The local maxima of `im` are the elements with values 2 and 4. In the result, `rows` and `cols` contain the row and column coordinates of these elements. The third result `vals` gives the values themselves. Check that you can tie all this in and that you understand what is happening here.

   Now try the maximum detector on a grey-level image that you have read into memory. It will be clearest if you smooth it first with:

   ```
    g = fspecial('gauss', 25, 5);
   s = convolve2(image, g, 'reflect');
   ```

   Then you can get the local maxima with

   ```
   [rows, cols, vals] = findpeaks(s);
   ```

   To see where the peaks are, you can plot them on top of the image. The Matlab `hold` operation allows you to superimpose graphics. The function `plot` will plot a set of points. It goes together like this:

   ```
   imshow(s);
   hold on;
   plot(cols, rows, 'g*');
   hold off;
   ```

   Note that the order of `cols` and `rows` has changed in the call to plot. This is because in ordinary Matlab array access, row comes before column, but the `plot` function is specified in terms of x and y coordinates, where x comes before y. This confusing change of convention is so entrenched in mathematics that it can't really be avoided – you just have to look out for it.

If you want to look at the code for `findpeaks`, you can (as with all the local code, and some of the Matlab toolbox code). Just type `edit findpeaks` and it will appear in an edit window. You will find that `findpeaks` calls another function, `getgreymaxima`, which does most of the work – you can look at that too.

2. **Apply a threshold to the peaks**. Suppose you want to use only the strongest peaks. You can pick them out by thresholding the `vals` result of `findpeaks`. For example, to find peaks that are at least 70% as big as the biggest, you could do this:

   ```
   thresh = 0.7 * max(vals);
   bigpeakindex = vals > thresh;
   ```

   The result of this, `bigpeakindex`, is a binary index array. Make sure you understand what it represents by printing it out and tying it in with the other arrays. Now you can pick out just the features that have these large values, like this:

   ```
   bigpeakrows = rows(bigpeakindex);
   bigpeakcols = cols(bigpeakindex);
   ```

   Now plot these peaks as in question 1, and check that they are at the centres of the brightest parts of the original image.

3. **Complete your feature detector**. Incorporate a call to `findpeaks` in the answer to Question 3 of Worksheet 3. You now have a feature detector that finds the row and column of an instance of a pattern in an image.

4. **Investigate some morphological operations**. Generate a binary image, either by thresholding a grey-level image, or using an edge detector (see previous worksheets). Try out the effect of various morphological operations on it, by looking up the documentation for `bwmorph` and using some of the options (with erosion `imerode` and dilation `imdilate` as a minimum). Try changing the structuring element (`strel`) to produce greater levels of shape smoothing.