Worksheet 3

1. **Matlab structure**. If you have not completed Worksheet 2, finish it before moving on. In particular, be sure that you can write a Matlab function, and understand argument passing in Matlab.

2. **General image operations**. Review Lecture 4 and try out as many as possible of the operations it introduces, introducing your own variations.

Review Lectures 5 and 6 and then try out the techniques using the following exercises (complete the tasks as a **homework** if you could not manage to finish them at this lab session). Start by setting the Matlab path and reading in the chessboard image as in the last lab class.

1. **Try built-in convolution masks**. Matlab has a toolbox function, `fspecial`, for making some special kinds of masks. Use the Matlab toolbox documentation to learn about these. Try, for example

   ```
   m = fspecial('gauss', 11, 2)
   ```

   Then carry out convolution with this mask and display the result. You can make a big Gaussian mask, and see the profile of weights that has been discussed in the lecture, like this:

   ```
   mbig = fspecial('gauss', 101, 20); profile = mbig(51, :); % a row through the centre plot(profile);
   ```

   Note that % starts a comment in Matlab. Look up `fspecial` in the documentation to see what the arguments are. For a Gaussian mask it's a good idea to always make the second parameter about 5 times as big as the third, as in these examples: can you figure out why? Try some of the other kinds of masks `fspecial` offers, to see what their effects are.

2. **Thresholding and convolution**. You built an edge detector using simple differencing and thresholding in the last class. Now write a function to combine smoothing, differencing and thresholding in a single function that does these operations:

   a. Convolve the input image with a Gaussian smoothing mask of width sigma.

   b. Convolve the smoothed image with a vertical differencing mask.

   c. Convolve the smoothed image with a horizontal differencing mask.

   d. Use the results of b and c to automatically decide a threshold (see below).

   e. Threshold the results of b and c and combine them.

   Your function should start with the following line

   ```
   function newimage = edge2(image, sigma, thresh_const)
   ```

   The argument `sigma` is used to set the size of the smoothing mask. Step d can be done

in various ways, but you might set the threshold as a constant multiplied by the standard deviation of the values in the array, or as a constant multiplied by the maximum value in the array. The argument `thresh_const` is used to pass in the value of the constant.

3. **A simple feature detector**. Use a chunk of the image (**Your face?**) as a convolution mask, to find parts of the image (**Our group photo?**) that resemble it. Develop the code as a script then turn it into a function. You will need to use the following:

   - the Data Cursor in the figure window, to get the coordinates of the part of the image you want to pick out.
   - the colon operator to pick out part of the image (the lecture slides have examples of doing exactly this).
   - (possibly?) the `rot90` or `flip` function to do the rotation of the mask necessary to make a template, as in the lectures.
   - the `mean` function to get the average of the values in the mask, as in the lecture, so that you can subtract this from the mask to give it zero mean.

   Remember that feature matching is actually usually done by **correlation**. Remember that convolution and correlation are not the same thing. The same MATLAB code can be used for both correlation and convolution, but you have to be careful to understand whether you need to flip the mask or not. Try it both ways to see what the difference is! After convolving the image with your mask, you should get peaks of brightness at the original mask position, and at places in the image that have a similar structure.

4. **Demos**. At this point you might like to look at some demos that relate to what you have been doing. The relevant ones are:

   - intro_demo
   - masks_demo
   - convolution_demo

   To use them, give the command (for example)

   ```
   edit intro_demo
   ```

   and then use cell mode to run each section of the demo in turn (click in a cell and type CTRL+ENTER).