Chapter 1 Exercises

The following exercises are designed to reinforce and develop the concepts and Matlab examples introduced in this chapter
Matlab functions: ***imabsdiff, rgb2hsv.***

Exercise 1.1 Using the examples presented for displaying an image in Matlab together with those for accessing pixel locations, investigate adding and subtracting a scalar value from an individual location, i.e. I(i,j) = I(i,j) + 25 or I(i,j) = I(i,j) - 25 . Start by using the grey-scale 'cell.tif' example image and pixel location (100, 20).What is the effect on the grey-scale colour of adding and subtracting?
Expand your technique to RGB colour images by adding and subtracting to all three of the colour channels in a suitable example image. Also try just adding to one of the individual colour channels whilst leaving the others unchanged. What is the effect on the pixel colour of each of these operations?

Exercise 1.2 Based on your answer to Exercise 1.1, use the for construct in Matlab (see *help for* at the Matlab command prompt) to loop over all the pixels in the image and brighten or darken the image.
You will need to ensure that your program does not try to create a pixel value that is larger or smaller than the pixel can hold. For instance, an 8-bit image can only hold the values 0–255 at each pixel location and similarly for each colour channel for a 24-bit RGB colour image.

Exercise 1.3 Using the grey-scale 'cell.tif' example image, investigate using different false colour maps to display the image. The Matlab function *colormap* can take a range of values to specify different false colour maps: enter help graph3d and look under the Color maps heading to get a full list. What different aspects and details of the image can be seen using these false colourings in place of the conventional grey-scale display?
False colour maps can also be specified numerically as parameters to the *colormap* command: enter *help colormap f*or further details.

Exercise 1.4 Load an example image into Matlab and using the functions introduced in Example 1.1 save it once as a JPEG format file (e.g. sample.jpg) and once as a PNG format image (e.g. sample.png). Next, reload the images from both of these saved files as new images in Matlab, 'Ijpg' and 'Ipng'.
We may expect these two images to be exactly the same, as they started out as the same image and were just saved in different image file formats. If we compare them by subtracting one from the other and taking the absolute difference at each pixel location we can check whether this assumption is correct.
Use the *imabsdiff* Matlab command to create a difference image between 'Ijpg' and 'Ipng'. Display the resulting image using *imagesc*. The difference between these two images is not all zeros as one may expect, but a noise pattern related to the difference in the images introduced by saving in a lossy compression format (i.e. JPEG) and a lossless compression format (i.e. PNG). The difference we see is due to the image information removed in the JPEG version of the file which is not apparent to us when we look at the image. Interestingly, if we view the difference image with *imshow* all we see is a black image because the differences are so small they have very low (i.e. dark) pixel values. The automatic scaling and false colour mapping of *imagesc* allows us to visualize these low pixel values.

Exercise 1.5 Implement a program to perform the bit-slicing technique described in Section 1.2.1 and extract/display the resulting plane images (Figure 1.3) as separate Matlab images.
You may wish to consider displaying a mosaic of several different bit-planes from an image using the subplot function.

Exercise 1.6 Using the Matlab *rgb2hsv* function, write a program to display the individual hue, saturation and value channels of a given RGB colour image. You may wish to refer to Example 1.6 on the display of individual red, green and blue channels.