Chapter 3 Exercises

The following exercises are designed to reinforce and develop the concepts and Matlab examples introduced in this chapter. Additional information on all of the Matlab functions represented in this chapter and throughout these exercises is available in Matlab from the function help browser (use *doc <function name>* at the Matlab command prompt, where *<function name>* is the function required)

Matlab functions: **imresize , size , whos , imadd , imsubtract , immultiply , imagesc , imcomplement , imabsdiff , implay , uint8 , horzcat , tic , toc , rgb2ycbcr .**

Exercise 3.1 Using the examples presented on arithmetic operations (Examples 3.1–3.4) we can investigate the use of these operators in a variety of ways. First, load the Matlab example images ' rice.png' and ' cameraman.tif' . Investigate the combined use of the Matlab *imresize*() and size() functions to resize one of the images to be the same size as the other. The Matlab command *whos* used in the syntax 'whos v' will display information about the size and type of a given image or other variable v .
Try adding both of these images together using the standard Matlab addition operator '+' and displaying the result. What happens? Now add them together by using the *imadd*() function but adding a third parameter to the function of ' uint16' or ' double' with quotes that forces an output in a 16-bit or floating-point double-precision data type. You will need to display the output using the *imagesc*() function. How do these two addition operator results differ and why do they differ?
Repeat this exercise using the standard Matlab subtraction operator '-' and also by using the *imsubtract*() function.

Exercise 3.2 Building upon the logical inversion example presented in Example 3.6, investigate the application of the *imcomplement*() function to different image types and different applications. First load the example image ' peppers.png' and apply this operator. What effect do you see and to what aspect of traditional photography does the resulting image colouring relate?
Image inversion is also of particular use in medical imaging for highlighting different aspects of a given image. Apply the operator to the example images ' mir.tif' , ' spine.tif' and cell image ' AT3_1m4_09.tif' . What resulting effects do you see in the transformed images which may be beneficial to a human viewer?
Exercise 3.3 Use the sequence of cell images (' AT3_1m4_01.tif' , ' AT3_1m4_02.tif, . . . ' AT3_1m4_09.tif' , ' AT3_1m4_10.tif') provided in combination with the Matlab *imabsdiff*() function and a Matlab for loop construct to display an animation of the differences between images in the sequence.
You may wish to use an additional enhancement approach to improve the dynamic range of difference images that result from the *imabsdiff*() function. What is result of this differencing operation? How could this be useful?
Hint . You may wish to set up an array containing each of the image file names 01 to 10. The animation effect can be achieved by updating the same figure for each set of differences (e.g. between the kth and (k-1)th images in the sequence) or by investigating the Matlab *implay*() function to play back an array of images.

Exercise 3.4 Using a combination of the Matlab *immultiply*() and *imadd*() functions implement a Matlab function (or sequence of commands) for blending two images A and B into a single image with corresponding blending weights wA and wB such that output image C is

$$\mathrm{C} = w_{\mathrm{A}}\mathrm{A} + w_{\mathrm{B}}\mathrm{B}$$

Experiment with different example images and also with blending information from a sequence of images (e.g. cell images from Exercise 3.3). How could such a technique be used in a real application?

Exercise 3.5 Using the thresholding technique demonstrated in Example 3.7 and with reference to the histogram display functions of Example 3.12, manually select and apply a threshold to the example image ' pillsetc.png' . Compare your result with the adaptive threshold approaches shown in Examples 3.15 and 3.16. Which is better? (Also, which is easier?) Repeat this procedure to isolate the foreground items in

example images ‘ tape.png’ , ‘ coins.png’ and ‘ eight.tif’ . Note that images require to be transformed to grey scale (see Section 1.4.1.1) prior to thresholding.

Exercise 3.6 Looking back to Examples 3.15 and 3.16, investigate the effects of varying the constant offset parameter C when applying it to example images ‘ cameraman.tif’ and ‘ coins.png’ . How do the results for these two images differ? Implement the third method of adaptive thresholding from Section 3.4.2 using the threshold t = floor((max-min) / 2) + C method. Compare this approach against the examples already provided for thresholding the previous two images and other available example images.

Exercise 3.7 Read in the example images ‘ cameraman.tif’ and ‘ circles.png’ and convert the variable resulting from the latter into the unsigned 8-bit type of the former using the Matlab casting function uint8() . Concatenate these two images into a single image (using the Matlab function *horzcat*()) and display it. Why can you not see the circles in the resulting image? (Try also using the Matlab *imagesc* function).
Using the logarithmic transform (Example 3.8), adjust the dynamic range of this concatenated image so that both the outline of the cameraman’ s jacket and the outline of the circles are just visible (parameter C > 10 will be required). By contrast, investigate the use of both histogram equalization and adaptive histogram equalization on this concatenated image. Which approach gives the best results for overall image clarity and why is this approach better for this task?

Exercise 3.8 Consider the Matlab example image ‘ mandi.tif’ , where we can see varying lighting across both the background and foreground. Where is information not clearly visible within this image? What are the properties of these regions in terms of pixel values? Consider the corresponding properties of the logarithmic and exponential transforms (Sections 3.3.1 and 3.3.2) and associated Examples 3.8 and 3.9. Experiment with both of these transforms and determine a suitable choice (with parameters) for enhancing this image. Note that this is large image example and processing may take a few seconds (use the Matlab functions tic and toc to time the operation). Contrast the results obtained using these two transforms to applying histogram equalization contrast stretch (Section 3.4.4) or adaptive histogram equalization (Section 3.4.6) to this image.

Exercise 3.9 Based on the grey-scale gamma correction presented in Example 3.11, apply gamma correction to a colour image using the example ‘ autumn.tif’ . Why can we (and why would we) apply this technique directly to the RGB image representation and not an alternative HSV representation as per the colour image histogram processing of Example 3.22?

Exercise 3.10 Based on Example 3.22, where we apply histogram equalization to a colour image, apply contrast stretching (Section 3.4.3, Example 3.17) to the colour example image ‘ westconcordaerial.png’ using the same approach. Experiment with different parameter values to find an optimum for the visualization of this image. Compare the application of this approach with histogram equalization and adaptive histogram equalization on the same image. Which produces the best result? Which approach is more ‘ tunable’ and which is more automatic in nature? Repeat this for the Matlab image example ‘ peppers.png’ . Do the results differ significantly for this example?

Exercise 3.11 Using the various contrast enhancement and dynamic range manipulation approaches presented throughout this chapter, investigate and make a selection of transforms to improve the contrast in the example image ‘ AT3_1m4_01.tif’ . Once you have achieved a suitable contrast for this example image, extract the corresponding histogram distribution of the image (Example 3.13) and apply it to the other images in the sequence (‘ AT3_1m4_02.tif’ , . . . ‘ AT3_1m4_09.tif’ , ‘ AT3_1m4_10.tif’) using histogram matching
(Example 3.20). Are the contrast settings determined for the initial example image suitable for all of the others? What if the images where not all captured under the same conditions? Here, we see the use of histogram matching as a method for automatically setting the dynamic range adjustment on a series of images based on the initial determination of suitable settings for one example. Where else could this be applied?

Exercise 3.12 In contrast to the approach shown for colour histogram equalization shown in Example 3.22, perform histogram equalization on each of the (R,G,B) channels of an example colour image. Compare the results with that obtained using the approach of Example 3.22. What is the difference in the resulting images? Repeat this for the operations of contrast stretching and adaptive histogram equalization. Investigate also using the YCbCr colour space (Matlab function *rgb2ybcr*()) for performing colour histogram operations on images. Which channel do you need to use? What else do you need to consider?