CS413 Coursework Assignment 2017 - MATLAB

Abhir Bhalerao

November, 2017

1 Introduction

The coursework assignment task is to solve **ONE** of the problems set out below using Matlab.

Note that your solution **must** run on the Department of Computer Science linux machines that you have been using in the laboratory practicals and it must be coded solely in Matlab.

The assignment is worth 30% of the module credit and therefore you should look to spend between 30 and 40 hours on this piece of work. You are required to submit the code together with a supporting document (report in PDF format) and these should be submitted using Tabula as a **single ZIP** file (more instructions below).

You need not submit any data provided to you. If you submit your own data, such as video data, please keep these short.

The deadline is Wednesday 10th January, 2018 (Week 1 of Term 2). If you have any questions about the work or need help with it, please ask me or the laboratory tutor in the remaining weeks of *this* term.

2 Problems

Please choose only **ONE** problem to solve.

2.1 Image Compression using DCT and Pyramids

1. Using Matlab, implement a DCT from **first principles**, i.e. without using any built-in Matlab DCT functions. It should take 8×8 blocks from an image and perform forward and inverse transforms. Write your solution as functions which takes an image block and returns a matrix of DCT coefficients for the forward transform, and a function that takes a matrix of coefficients and returns the reconstructed data. Demonstrate that your code works with suitable examples.

2. Demonstrate with examples that your method will work for blocks of arbitrary size by writing a second pair of functions which can take a second block-size

parameter. Demonstrate the new methods work for different block sizes, e.g. 16×16 and 64×64.

3. Use your DCT transform to implement an image compression algorithm.

   The compression should use a simple thresholding of the DCT coefficients and calculate the total number of bits after compression and the resulting distortion (SNR), measured in dBs. Plot a rate-distortion curve to show the performance of your compression on grayscale versions of the Barbara, Lena image and a third image of your choice.

4. Implement a Gaussian and Laplacian pyramid for images of sizes 2^N. You will need to implement the REDUCE and EXPAND operations, again from first principles.

5. Use your Laplacian pyramid implementation to compress images using thresholding of Laplacian coefficients and appropriate quantization. You should quantize coefficients at different levels with different numbers of bits.

6. Compare the performance of your Pyramid Image Coding scheme with your DCT compression by plotting rate-distortion curves of a suitable image or images of your choosing. Give a critical analysis of your methods and results of experiments in your report.

2.2 Image Segmentation

1. Write a colour image segmentation algorithm to allow different coloured objects to be automatically separated and counted. Test out your method on the set of images given in `lego-bricks.zip` and measure the classification error for a number of objects of different colours.

2. Add Gaussian distributed random noise to the images at a number of different levels such that the signal to noise ratio of the images is in the range 1 to 20. Test the performance of your segmentation and optimise it to produce results with the fewest misclassified pixels at different noise levels. Make tables of the performance results.

3. Extend your segmentation method to extract bricks of the same type from images in `lego-bricks.zip` and place them, correctly oriented, into separate images. So for example, all bricks of the same shape and/or colour should be place neatly into rows and columns of a new image (one for each class of objects). Decide for yourself how to categorise objects as being similar, e.g. size, colour, shape.

4. Use the algorithm developed in the previous step and test it on an image of simple, small objects of your choosing against plain and cluttered backgrounds. Explain how well your method performs. In your report, discuss your results and any changes you have made to the method to make it work on your data.

2.3 Object Matching in Videos

1. Using the images from `objects-train.zip` write Matlab code using the Computer Vision Toolbox to build feature descriptors of the grocery labels shown.

2. Design and implement a feature matching framework to locate matching features of known items from video `objects-test-1.mov`. Show features that have been matched and, if a particular object is recognised, locate and label the region which contains the recognised item.

3. Test the performance of your method on the items shown in the cluttered scene (video `objects-test-2.mov`).

4. Capture your own videos of your own grocery labels and perform quantitative testing of your method. You will have to implement a scoring mechanism to estimate how similar are two sets of features.

3 Assessment Criteria

The coursework marks (out of 30) and will be allocated into the following areas:

Functionality	How much of the functionality required is implemented by the solution and how well it has been done.	6 marks
Code	Have Matlab features and functions been used appropriately; is the code readable, well structured and commented.	8 marks
Testing	Evidence of testing to prove that the solution works; including quantitative analysis of results.	6 marks
Documentation	Readability and quality of presentation of the documentation. Contents describe well the features of the solution, and instructions given for running are satisfactory. Clear description of method, presentation of results and evidence of testing.	10 marks

4 Collaboration and Plagiarism

As for all pieces of assessment you undertake, the submission must be entirely your **own** work. Discussion of ideas with your colleagues is OK, but the final piece of work should be your own.

Also, please do not copy without attribution from work you may have found on-line. Using code that is given in the laboratory tutorial examples is allowed.

We will use anti-plagiarism software to look for copying between submissions (past and present) and with external sources. The University's rules on plagiarism and cheating are clearly set out in Section 2, Regulation 11 of the University Calendar:
`go.warwick.ac.uk/calendar/section2/regulations/cheating/`.

Any breaches of the rules will incur a severe penalty or a zero mark.

5 Deadlines and Submission Formats

The deadline for this coursework is **12pm Wednesday, 10th January 2018** (Week 1, Term 2).

Please submit a **single** ZIP file containing a set of source files (`.m`), any extra images/videos you have used (**please keep videos short**), and a PDF of a written report which describes your code and your experiments.

Written Report Your supporting document must be **no more than 1250 words** in length (excluding program examples). Please indicate the word count at the end of the text. It should state **clearly and concisely**:

- an overview of your solution and its main design/algorithmic aspects
- how specific image analysis/video analysis/Matlab feature are used and implemented
- how you have tested and verified the correctness/performance characteristics of your programs
- how to run you code and how to use code
- it should include equations, figures, plots, tables and explanations of algorithms, and code snippets where appropriate

You may like to use the Matlab Code Publishing features, see
`www.mathworks.co.uk/help/matlab/matlab_prog/publishing-matlab-code.html`
to document your experiments and produce your report in PDF.

Abhir Bhalerao, November 2017