CS413 Image Compression using DCT and Pyramids

Callum Marvell
u1606521

January 10, 2020

1 Introduction & Overview

This report further details and outlines the methodology and specific techniques behind the accompanying submission as part of the CS413 coursework task.

The aim of this task, broadly speaking, was to develop both a DCT-based compression suite (from first principles) as well as a similarly functional laplacian-pyramid-based compression suite. Hence, the task can be easily split into two fairly distinct portions; one focusing on the DCT compression and the other on the laplacian-pyramid method. It is explored as such, with specific techniques used in the implementations submitted in the remainder of this report.

2 Specific Techniques

2.1 DCT and Inverse DCT

Implementation of DCT from first principles has two primary techniques:

- Direct implementation of the DCT formula, through use of nested for-loops.
- Use of pre-calculated basis vectors (based on the block size used) and associating values to each of the basis vectors representing their relative weight within a block being converted

To simplify somewhat, after starting with a direct implementation of the DCT formula that was very slow (taking up to 7 minutes to fully convert an image for slightly larger block sizes), the decision was made to move to a basis vector implementation. This very closely mirrors the approach used in the labs - so the codebase was migrated from there and edited to fit the purpose.

Inverse DCT was developed for both strategies, but given the final decision to use basis vectors, is an approach specifically designed to undo that particular approach.

2.2 Block Thresholding and Quantization

When thresholding the DCT matrix for a block, any threshold setting can be used - the function developed allows for this through parameter options. Whatever value given (between 0 and 1) acts as a multiplier for the largest coefficient, ultimately producing the minimum acceptable absolute value which is not removed. Anything below this value is set to 0, anything above remains as it is. This drastically increases the potential compression factor of the image, as it produces a large contiguous block of 0-values at the end of all DCT blocks (when traversing along them in the diagonal pattern typical to Huffman encoding and other such encoding schemes). Compression values demonstrating this can be found in the notebook.

Thresholded blocks can optionally also be quantised to even further increase the maximum possible compression that the image can reach. This works by more heavily quantising the bigger values in the top-right corner (since those further away often end up being removed through quantisation anyway). This is achieved, as detailed in the coursework specification, by building a quantisation matrix and dividing values (and multiplying the result) in the DCT block by their corresponding quantisation matrix value. This allows for the values that remain in the DCT matrix to be represented with generally fewer bits than usual. This

is not directly accounted for in the compression calculations within the notebook and the functions available within (in other words, it is assumed to have negligible affect over just thresholding) - however assuming the compression of the largest value to the nearest 8 (as specified in the coursework specification) this should have only so much bearing on the efficacy of the compression itself. Quantisation does more at lower threshold values (since at higher ones it often only quantises one single value per DCT block, saving only 3 bits) - at which point it has less of a compression enhancing effect and more of a visual-distortion effect than simply increasing the threshold on the DCT block.

2.3 Laplacian Pyramids

Gaussian pyramids are constructed for images by repeated subsampling and convolution of a gaussian filter. More specifically, both the EXPAND and REDUCE operations were implemented (using laboratory work already done as a base). A pyramid generation function was built off the back of this, capable of generating an appropriately deep pyramid for any given square image of side length $2\hat{n}$. Laplacian pyramids are computed (approximately) by finding the difference of guassians (DoGs) of the level in the corresponding gaussian pyramid.

2.4 Thresholding and Quantization on Pyramids

The laplacian levels are the only ones all stored in the final solution (the lowest acceptable gaussian level is stored to maximally reduce required space in compression), so only these are thresholded/quantised. Thresholding is on absolute value (since 90% of the values in a laplacian level are very small) and instantly massively reduces required storage space (by optimising run-length encoding). Quantization affects higher laplacian levels (those with smaller index) more as they carry less important information than lower levels. Each level is quantised to a number of bits per pixel proportional to it's level - as detailed in cs413 notes.

3 Testing

While specific testing functions and tools weren't specifically added, new functions were added incrementally with thorough testing facilitated by jupyter notebooks - and simple testing of most functions is easy, simply by bypassing the undesired parts of the process.

3.1 Performance Graphs

Performance graphs are included below, showing the performance of DCT with thresholding versus thresholding and quantisation, as well as performance for laplacian pyramid compression.

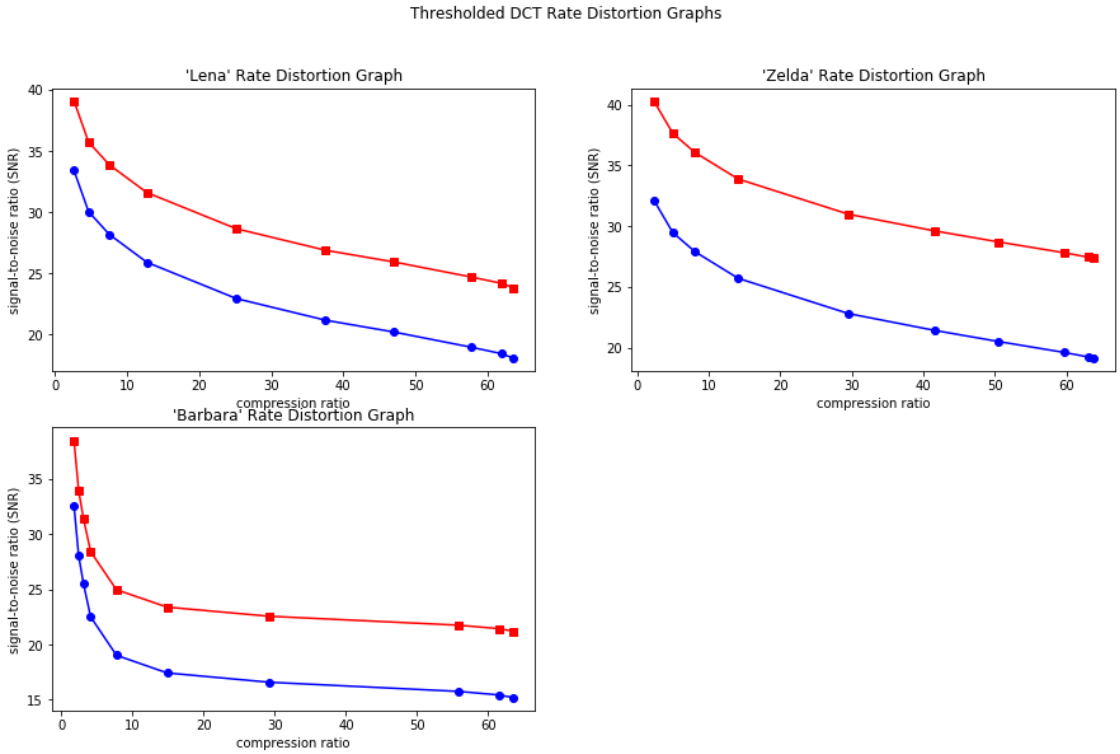

Figure 1: Thresholded DCT Rate-Distortion Graphs

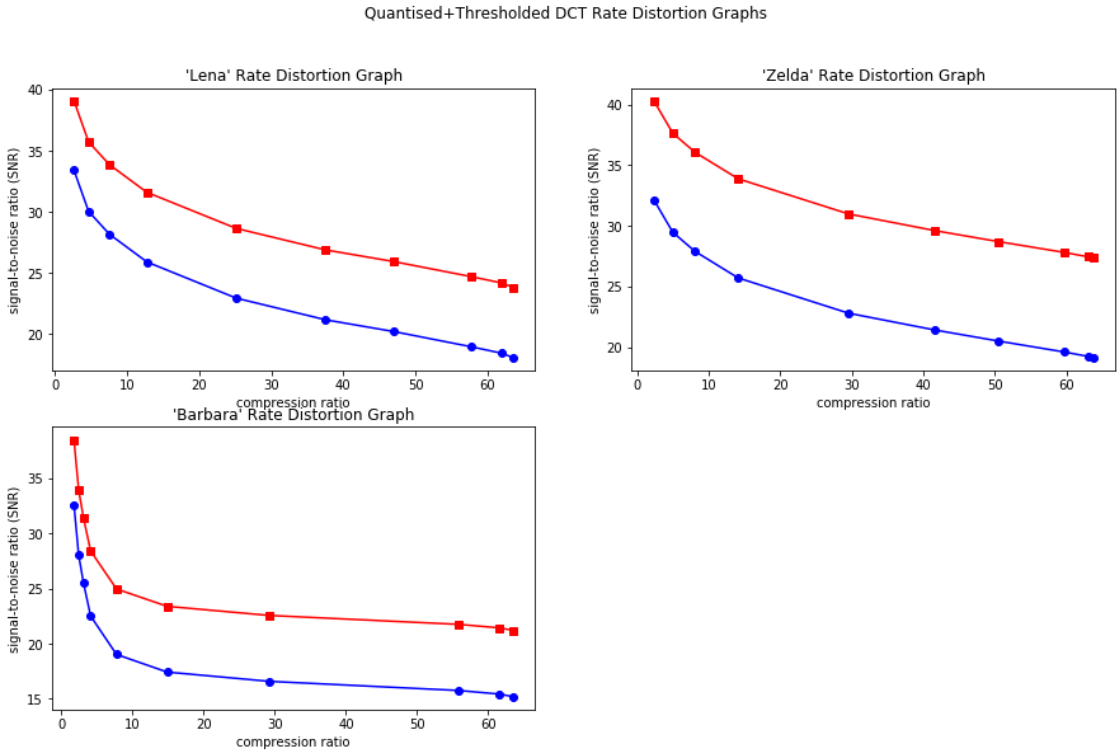

Figure 2: Thresholded & Quantised DCT Rate-Distortion Graphs

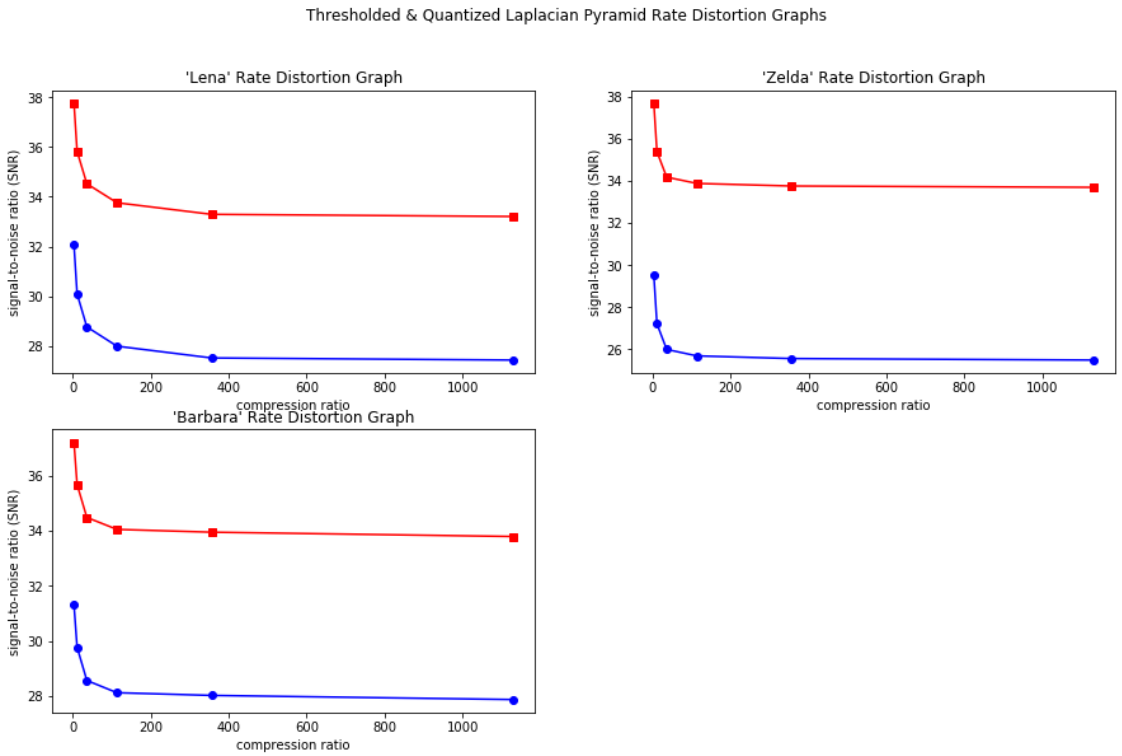

Figure 3: Thresholded & Quantised Laplacian Pyramid Rate-Distortion Graphs

As is plain to see, the compression factor of laplacian compression is regularly significantly higher. However, the resultant image quality is significantly lower (or at least not representative of the original image - with many fine-grain details being lost) - particularly at the levels of compression that give the particularly impressive compression ratios in the 100s/1000s. As such, it seems that thresholding DCT components is a much safer and more reasonable compression method for most situations (where relatively visually lossless compression is desirable) - however for certain situations, where final visual quality is less important or noticeable (such as thumbnails or similar) the increased compression ratio of laplacian compression might well win out.

4 Conclusion

All tasks were completed as laid out in the specification for task 1 of the CS413 coursework. This ultimately required a relatively thorough investigation and significant understanding of DCT and Laplacian pyramid compression methods. Finally, a brief comparison between the two methods was conducted, via use of rate-distortion graphs and analysis of the difference between them, as well as speculation about potential use cases for each.

——Currently there are roughly 980 words in this document——