DSP Project

Transform coding of images

This document provides an outline for a Matlab project to be completed by the end of the course. You are expected to investigate, in detail, methods related to solving the problem. There is a design element to the project, and a quantitative evaluation of the performance of the proposed methods must be performed and presented. You are to write up a comprehensive report (of no more than 8 pages) describing your method and results. You should work in groups of two, although you may work alone if you really want to.

If you wish to propose a project of your own, then please come and talk to me.

Note that this document is still in preparation, and may be added to during the course of the project.

The task

The aim of this project is to investigate transform-based coding methods for image and video compression. The most common (although slightly outdated) transformation for this purpose is the discrete cosine transform (DCT), which is used in the JPEG still image compression format.

The transform coding aspects of the JPEG format are described in good detail in the document `dct.pdf`, available on the course website. It is important for you to read through this document and understand it in detail. The document `compres.pdf` contains some additional information, and has some nice figures.

JPEG image compression operates on 8x8 blocks of nonoverlapping image data. Assuming a greyscale image, the following steps are applied to each block:

1. Take the data and transform into the DCT domain.
2. Quantise the DCT coefficients according to the level of compression required. This is a lossy process — data is destroyed in JPEG compression. Since the human visual system has different sensitivity to different DCT components, a different quantisation step size is applied to each coefficient. Tests on human subjects have determined standard sensitivity functions.
3. The DCT coefficients are zigzag reordered, basically from low to high frequencies, in the hope that long runs of zeros will be present after the quantisation. These can be efficiently represented using runlength coding, and some compression is achieved in this way.
4. In JPEG compression the resulting data is Huffman coded. A Huffman code is a prefix code which assigns short codewords to symbols which occur frequently, and can losslessly compress data when the probability distribution of the symbols is not uniform.

In this project we don't want to go into the coding aspects of the problem (at least not initially). Instead we will replace the runlength coding and Huffman coding stages with a generic coder — `gzip` or `winzip` (which both use Lempel-Ziv coding, a run-length scheme with some clever use of context). Good compression will still be obtained from the data reorganisation and truncation effected by the quantisation and zigzag reordering.

The starting point for this project, should you need one, is the file `project04_jpeg.m` from

`http://www.dip.ee.uct.ac.za/~nicolls/eee4001f/projects/project04`.

This implements a basic transform coder for a greyscale image. Note

that it uses the Matlab command `gzip`, which is apparently only available in recent versions. If your version is missing this command you may have more luck with the `zip` command. Alternatively, comment out the portion of the code that uses this, and just compress the resulting file using a third-party application like `winzip`.

(Note: I wrote `project04_jpeg.m` quickly, and there may be some small mistakes. Bonus marks if you spot one.)

Approach

Look at the code provided, and understand it. The functionality looks simple, but the brevity of the source can be misleading — Matlab can do things in a few lines that will take thousands of lines in a low-level programming language. Pay particular attention to:

1. The specification of the DCT in terms of matrix multiplication. You will need to relate the definition of the DCT (in terms of the double sum) to the matrix that effects the transformation. It may help to examine the formula and the matrix for the simpler case of $N = 4$. Note also that the DCT matrix `T` in the code has a very special structure — it is orthogonal (or unitary), so $\mathbf{T}'\mathbf{T} = \mathbf{T}\mathbf{T}' = \mathbf{I}$. This means that $\mathbf{T}^{-1} = \mathbf{T}'$. Unitary transformations are exceptionally important in engineering applications (and in linear algebra in general).
2. The process used for raster reordering. This would be painful to program in general, but can be made a one-liner in Matlab with some care.
3. Data types. Bear in mind that Matlab *does* have data types — it often just hides it from you. Specifically, our image data is `uint8` and we want our output data to be `uint8`, but we can't do complicated math on 8-bit numbers. So there are various conversions in the code (which may be wrong).

At the very least, you should analyse the performance of this algorithm. This requires quantifying the error between the reconstructed data and the original data for different compression levels. The standard measure for error (which is admittedly only partially satisfactory) is called the peak signal-to-noise ratio (PSNR). Other avenues of exploration include:

1. Exploring the DCT and quantisation. Look at some blocks of data in detail, and report on what you see.
2. Changing the block size from 8×8. This will involve finding new quantisation matrices (I think they're out there — I've used them before).
3. Implementing (or finding an existing implementation of) the run-length code and/or the Huffman code. Arithmetic coding is an alternative entropy coder which is now available since the patent has expired.

I did an undergraduate thesis on transform coding. Believe it or not, the document is still available at `http://www.dip.ee.uct.ac.za/~nicolls/publish/FNugrad.pdf`.

If you find coding of static images to be a little passé, you could consider looking at the transform coding aspects of video sequences.