EQ2330 – Image and Video Processing

Project #2: Image Compression

due Monday, December 5, 2016, 11:59 pm

Students should submit their reports electronically. Please email your document in PDF format to the course assistant assigned to this project with the subject line "EQ2330 - Project". Your email should also include relevant Matlab source code. You can find detailed instructions for submitting your report at EQ2330 webpage: `https://www.kth.se/social/course/EQ2330/`
There is a template for the report availible at EQ2330 webpage. It also contains some instructions you are expected to follow.

1 Introduction

The aim of this project is to evaluate achievable performance of two transform based image compression algorithms. The transforms are the discrete cosine transform (DCT) and the fast wavelet transform (FWT). You are to implement the transforms, quantize the transform coefficients, evaluate the quality degradation due to quantization and estimate the rate needed for coding of the quantized transform coefficients. The algorithms that you implement in this project will be used in Project 3, which is why it is recommended to invest the time necessary to get the implementations right.

2 DCT-based Image Compression

This problem investigates the rate-distortion performance of a transform image coder using the DCT for 8×8 blocks of an image.

2.1 Blockwise 8×8 DCT

There are four types of DCT, the most prevalent one being DCT-II, which is the one used in this project. It is a separable orthonormal transform, so the DCT transform of a signal block of size $M \times M$ can be expressed by $y = AxA^T$, and the inverse DCT transform is

$x = A^T y A$, where x is the $M \times M$ signal block, and A is the $M \times M$ transform matrix containing elements

$$a_{ik} = \alpha_i \cos\left(\frac{(2k+1)i\pi}{2M}\right) \quad \text{for} \quad i,k = 0,1,...M-1$$

with

$$\begin{aligned} \alpha_0 &= \sqrt{\frac{1}{M}} \\ \alpha_i &= \sqrt{\frac{2}{M}} \quad \forall i > 0. \end{aligned}$$

Implement a DCT of block size 8×8 for still images and the inverse DCT of block size 8×8. Print out the values of the matrix A. Hint: We do not care about fast algorithms here, simply use a matrix multiplication in MATLAB.

2.2 Uniform Quantizer

Implement quantization of the coefficients by a uniform mid-tread quantizer without threshold characteristic. Use the same step-size for all coefficients. Plot a graph of the quantizer function.

2.3 Distortion and Bit-Rate Estimation

The Peak Signal to Noise Ratio (PSNR) will be used to measure the quality of the reconstructed images. It is defined for 8-bit images as follows:

$$\text{PSNR} = 10 \log_{10}\left(\frac{255^2}{d}\right) \quad \text{[dB]}.$$

The average distortion d will be the mean squared error between the original and the reconstructed images. Compare d with the mean squared error between the original and the quantized DCT coefficients. How do they compare and why?

To estimate the bit-rate required to encode the coefficients, assume that we use the ideal code word length of a variable length code (VLC) that encodes each coefficient individually, but uses a special code for each of the 64 coefficients in a block. In other words: we assume that we use different VLCs for coefficients 1, 2, etc. within each block, but coefficient i uses the same VLC in each block.

Carry out all measurements using the three images *boats, harbour,* and *peppers*, with the same set of VLCs applied across all three images. Vary the quantizer step-size over the range $2^0, 2^1, 2^2, ..., 2^9$ to measure a rate-PSNR curve that shows what PSNR we expect over a range of bit-rates. Just one rate-PSNR curve for the total data set is sufficient.

Hint:

- Different VLCs should be used for different quantizer step-sizes.
- Ideal code word length can be fractional.
- To speed up your measurements you can use the function `dct2`.

3 FWT-based Image Compression

This problem investigates the rate-distortion performance of a transform image coder using the FWT.

3.1 The Two-Band Filter Bank

The most important building blocks of an FWT implementation are the one-dimensional two-band analysis and synthesis filter banks. There are two classical implementations of the filter bank for the FWT: direct implementation and lifting implementation. The direct implementation consists of filters, down-sampling and up-sampling operations. All filtering operations are based on a prototype scaling vector. For more information on the FWT, see Section 7.4 in [1]. The lifting implementation employs a special structure so-called lifting scheme (see an example in Fig. 1). For more information on the lifting implementation see the lecture slides or [2].

Implement a one-dimensional two-band analysis filter bank function in MATLAB, which has the signal and a prototype scaling vector as arguments. Choose one of the two classical implementation methods. The output should be a vector consisting of approximation and detail coefficients, i.e., the wavelet coefficients. The output vector should be equal to the signal size. A few scaling vectors are supplied on the course home page.

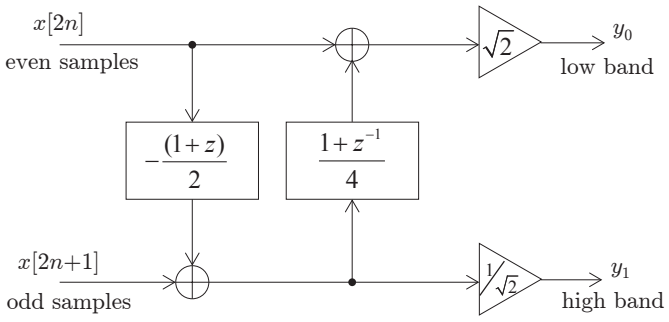

Figure 1: Lifting implementation of the 5/3 filter bank.

Implement a one-dimensional two-band synthesis filter bank function in MATLAB, which has the wavelet coefficients and a prototype scaling vector as arguments. Note that biorthogonal filters allow for perfect reconstruction. Therefore, serial application of the analysis and synthesis filter banks to an input signal should produce an output that is (down to machine precision) equal to the input signal for the given biorthogonal filters.

Hint:

- Use a periodic extension of the signal, as illustrated in Figure 2, to handle border effects that occur when filtering. Due to periodicity, the filtered signal can be truncated such that the correct number of output values is obtained.
- Depending on your implementation you might need to reorder your output signal, e.g., using `circshift()`.

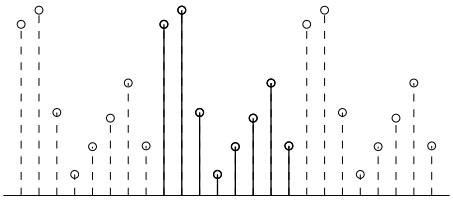

Figure 2: Periodic extension of a signal.

3.2 The FWT

A two-dimensional FWT (inverse FWT) can be implemented by recursive application of the analysis (synthesis) filter bank that you have implemented. Implement the FWT and the inverse FWT of arbitrary scale, using the Daubechies 8-tap filter or 5/3 wavelet, for images of widths and lengths that are a power of 2. Plot the wavelet coefficients for scale 4 of the image *harbour*.

3.3 Uniform Quantizer

Implement quantization of the wavelet coefficients obtained in the previous task, by a uniform mid-tread quantizer without threshold characteristic. Use the same step-size for all coefficients.

3.4 Distortion and Bit-Rate Estimation

Use the PSNR to measure the quality of images that are reconstructed after quantization. Again, the average distortion d will be the mean squared error between the original and the reconstructed images. Compare d with the mean squared error between the original and the quantized wavelet coefficients. How do they compare and why?

To estimate the bit-rate required to encode the wavelet coefficients, assume that we use the ideal code word length of a VLC that encodes each coefficient individually, but uses a special code for each of the subbands.

Carry out all measurements using the three images *boats, harbour,* and *peppers*, with the same set of VLCs applied across all three images. Vary the quantizer step-size over the range $2^0, 2^1, 2^2, ..., 2^9$ to measure a rate-PSNR curve that shows what PSNR we expect over a range of bit-rates. Just one rate-PSNR curve for the total data set is sufficient.

References

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, Upper Saddle River, New Jersey, second edition, 2002.

[2] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, Calif., second edition, 1999.