Image Compression

Fourier and Wavelet Analysis CTH & GU

October 31, 2006

1 Preparation

The programmes and data needed in this assigment are copied to your home directory by typing (at the Unix prompt):

```
cd
cp␣~bergh/fwa/Lab3/*.*␣~/
```

Once these files have been copied, start MatLab. To import and display the Lena image, type:

```
X=imread('lena.tif');
I=im2double(X);
imshow(I)
truesize
```

Both `X` and `I` are 512×512 matrices and contain the grey scale values of the image. The values in `X` are 8-bit numbers ranging from 0 to 255, where 0 corresponds to black and 255 to white. The values in `I` are floating point numbers ranging from 0.0 to 1.0, where 0.0 corresponds to black and 1.0 to white. To compute the 2D-wavelet transform of the image using the Haar (= db1) wavelet type:

```
[wc,s]=wavedec2(I,3,'db1');
```

Here 3 means that we stop the wavelet transform after 3 steps, when the coarsest image has size $2^6 \times 2^6$, $6 = 9 - 3$. (Using `wrcoef2` and `detcoef2`, you could try to see in which blocks the vertical and horizontal edges appear most clearly. Note that little energy is in the high-pass bands.)

2 Compression Programme

We are going to compare the performance between different wavelet bases and the Fourier basis for the compression of images, without recourse to the more sofisticated utilities in the toolbox. More specifically, we will use the Haar, Daubechies-4, and a biorthogonal wavelet basis as well as a discrete time cosine basis.

The file `compress.m` contains the following incomplete programme:

```
function CI=compress(I,L,t)

[N,J]=size(I);

% Forward Wavelet Transform
[wc,s]=...

% Threshold wavelet coefficients
temp=wc(1:2^{N-L},1:2^{N-L});
wc=wthresh(wc,'h',t);
wc(1:2^{N-L},1:2^{N-L})=temp;

% Count number of non-zero coefficients
nz=nnz(wc);

% Inverse Wavelet Transform
CI=...

% Print out the compression ratio
...
```

2.1 Exercise 1

Your task is to complete this programme by replacing the dots ... with appropriate commands. The input is the image `I`, the dyadic size `N-L` ($2^{N-L} \times 2^{N-L}$) of the coarsest image, and the threshold `t` > 0. The programme first computes the forward wavelet transform of the image. Then all wavelet coefficients with modulus smaller than `t` are set to zero and finally the inverse wavelet transform of the thresholded coefficients is computed. The output `CI` is the compressed image. The programme should also print the compression ratio, i.e. the number of pixels in the original image divided by the number of non-zero wavelet coefficients. Set `L` $= 3$ since further splittings in the wavelet transform do not give better compression; most of the compression is gained in the first three splittings of the transform.

Hand in a print out of the complete programme.

3 Wavelet Compression

Modify the threshold until you achieve a compression ratio approximately equal to 30. Then modify the programme so that the wavelet is the Daubechies-4 wavelet, 'db2' (*sic!*). Again find a threshold so that the compression ratio approximately equals 30.

We are also going to use a biorthogonal symmetric basis. The synthesis scaling function is a hat function and the analysis wavelet has two vanishing moments. This has the name `bior2.2` . We now have different filters in the analysis and synthesis parts. Change the programme so that it performs a biorthogonal wavelet transform. Modify the threshold so the compression ratio is the same as above.

3.1 Exercise 2

Compare the three compressed images; the `figure` command might be useful for displaying several images at once. Rank the image qualities from best to worst and write a table with the chosen thresholds and corresponding compression ratios.

What is the difference between and characteristics of each of the three compressed images?

4 Fourier Based Compression

Finally, we shall compare the wavelet methods above with the discrete cosine transform. This transform is the basis of the most popular compression system today: JPEG. The image is first divided into 8×8 blocks and then a 2D discrete cosine transform is applied to each block. The objective being the same as with the wavelet transform – to obtain a space-frequency decomposition of the image. (Using a wavelet transform we really obtain a space-scale decomposition). If the image contains correlations in space and frequency the transform will have most of its energy packed into a few number of large coefficients.

So what is the discrete cosine transform? It is really just a discrete Fourier transform where we have taken care of the boundary coefficients in a special way. Let us say we have a periodic signal $f(t)$ with period T. If $f(0) \neq f(T_-)$ we will have a discontinuity at $t = 0$ (or $t = T$). This will cause the Fourier coefficients to decay slower towards large frequencies and the packing of the coefficients is decreased. If we instead define the $2T$-periodic signal $g(t)$ as

$$g(t) = \begin{cases} f(t) & 0 \leq t < T, \\ f(-t) & -T < t \leq 0. \end{cases}$$

it will be continuous at time $t = -T, 0,$ and T, see Figure 1. The symmetry of g will cause all the sine terms in the Fourier series to be zero and what is left is called the cosine transform. In the discrete case we just have a signal $x = (x_0, x_1, ..., x_{N-1})$ with period N that is periodized in the same way to get period $2N$; we then get the discrete cosine transform. The programme `dctcompress.m` will compress an image using the discrete cosine transform. Modify the threshold so that you can compare the result with the previous ones.

4.1 Exercise 3

The so called blocking effect should now be clearly visible. Why does this blocking occur? Is it the same type of blocking as in the Haar case? How well does it perform compared to the three wavelet bases?

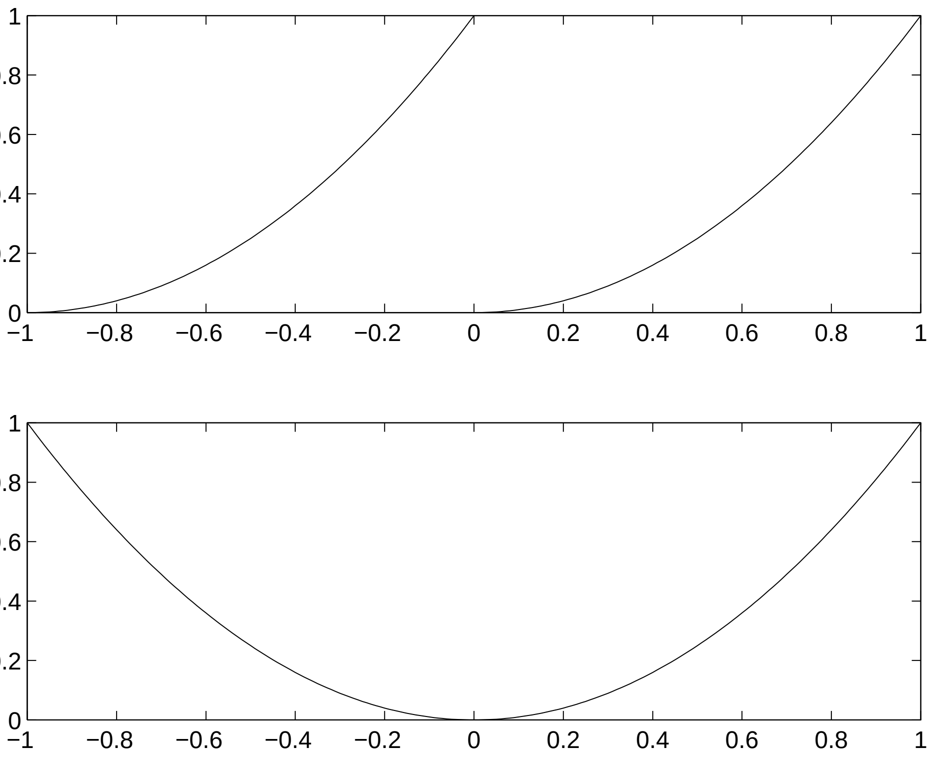

Figure 1: Top: $f(t)$ with period $T = 1$. Bottom: $g(t)$ with period $T = 2$.