Image Compression

Fourier and Wavelet Analysis CTH & GU

October 29, 2008

1 Preparation

The programmes and data needed in this assigment are available from the web page. There are three images:
`TestbildTV12gray16.tif, TestbildTV12-1024gray16.tif`, which are two versions of the same test image from the Swedish television (from a long time ago; they come from a web page,
`http://www.tv-testbild.com/index.htm`).
There is also a normal photo in the same catalog (`bangalore-gray16.tif`). You are also encouraged to choose a photo of your own choice; to use it as described in this guide, it has to be in the tif-format, however. Athough the images supplied are black and whit, you can try also color images[1].

To carry out all of the exercise, you also need to download a couple of matlab scripts, `compress.m, dctcompress.m` and `wcshow.m`. Once these files have been copied, start MatLab.

We will use the wavelet toolbox in this lab. It is a collection of tools for wavelet analysis, with commands for carrying out elementary wavelet calculations as well as advanced graphical toolboxes. Here only the more elementary instructions will be used. *You are encouraged to read carefully the man-pages for all instructions used in the assignement, and also to read other parts of the wavelet documentation in MatLab.*

To import and display an image, type something like:

[1] However, with color photos you have to do each color layer separately. When using the command `im2double`, write `I=im2double(X(:,:,c))`, where `c` is 1,2 or 3, for different color layers

```
X=imread('TestbildTV12gray16.tif');
I=im2double(X);
imshow(I)
truesize
```

The matrices `X` and `I` contain the grey scale values of the image. The values in `X` are 16-bit numbers ranging from 0 to 65535, where 0 corresponds to black and 65535 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.

The instruction for computing a 2-D single level discrete wavelet transform is `dwt2`, and the basic useage is

```
[cA,cH,cV,cD] = dwt2(I,'db1');

a1=1.0; a2=10.0; a3=10.0; a4=10.0;
imshow([ a1*cA, a2*cH; a3*cV, a4*cD])
```

The argument `'db1'` means that we use the Haar wavelet, and the factors `a1, a2, a3, a4` are used only to make the detail-images more clear.

- *Carry out the decribed procedure on the TV test images, and save the figures.*
- *Make shure that you understand the meaning of* `cA, cH, cV` *and* `cD`*, and explain what you see in the images. What happens if you change from* `'db1'` *to another kind of wavelet?*

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. (Using `wrcoef2` and `detcoef2`, you can try to see in which blocks the vertical and horizontal edges appear most clearly. Note that little energy is in the high-pass bands.) Read the man-pages for wavedec2, to understand the meaning of `wc` and `s`.

You can also display the three levels of the wavelet transform using the function `wcshow`:

```
wcshow(wc,s)
```

- *Try this out on the TV test image, and explain what you see in the image*

2 Wavelet Compression

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)

method = ... ;

% Forward Wavelet Transform
[wc,s] = wavedec2(I,L,method);

% Threshold wavelet coefficients
temp = wc(1:s(1,1)*s(1,2));
                              %this will contain the coarsest
                              %approximation
wc = wthresh(wc,'h',t);
wc(1:1:s(1,1)*s(1,2)) = temp;

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

% Inverse Wavelet Transform
CI = ...

% Print out the compression ratio

 ...
```

- *Complete this programme by replacing the dots ... with appropriate commands. To understand the meaning of the code, read the man pages carefully*
- *The input is the image to compress (*`I`*), the level of wavlet decomposition* `L`*, and the thresholdlevel* `t`*.*
- *The output is the compressed image,* `IC`*.*
- *Carry out experiments with different levels of wavlet decompositions and thresholdlevels. Try to obtain a compression ration of approximately 30. Also try out different wavelets:* `'db1', 'db2', 'bior2.2'`
- *To which level is it useful to carry out the wavelet decomposition? Why is it not useful to continue further? Which wavelets give the best result?*
- *Compare the 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.*
- *In addition to answers to the questions, and some compressed images, the hand in should contain the completed matlab code*

3 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.

- 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?

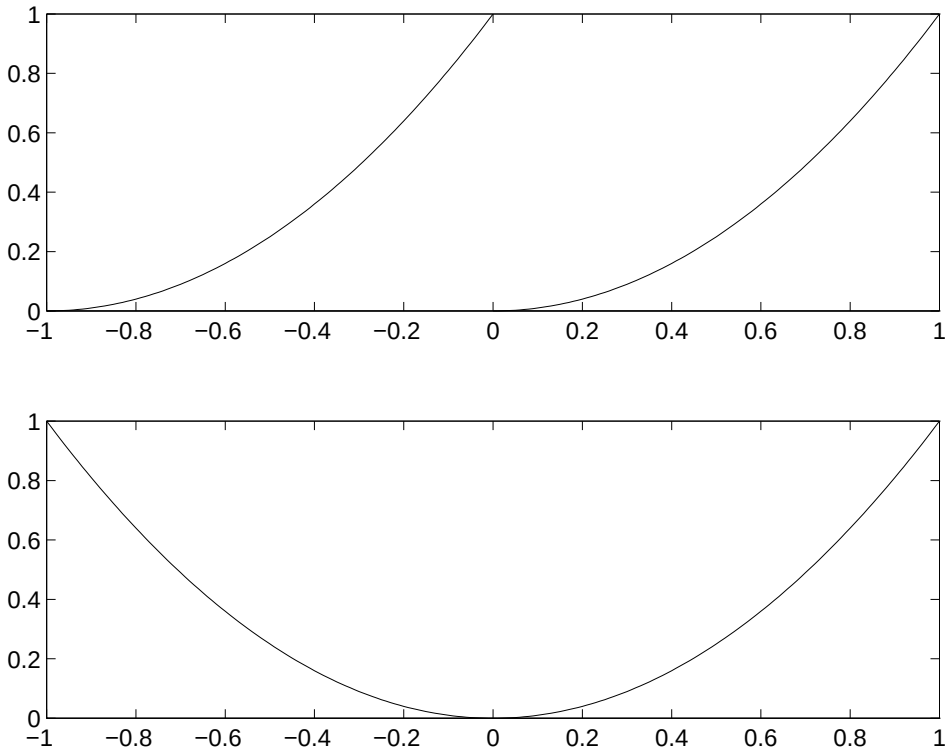

Figure 1: Top: $f(t)$ with period $T = 1$. Bottom: $g(t)$ with period $T = 2$.