TSBK02/TSBK06 Computer Lesson: Transform Coding

May 2010

1 Introduction

The problems for this computer lesson deals with transform coding. We use Matlab to solve the problems. In order for Matlab to find some of the functions, you must add the course directory to the search path:

```
>> addpath /site/edu/icg/tsbk02/matlab/
```

2 Test Data

As a test signal, you can use filtered white noise. For instance:

```
>> x=filter(1, [1 -0.9 0.1], randn(256000,1));
```

Another option is to use a real onedimensional signal, such as one of the channels of the music data from lab 2.

3 Discrete Walsh Hadamard Transform

A DWHT transform matrix (in this case of size 4) can be created by the function `dwhtmtx`:

```
>> bs = 4;
>> A = dwhtmtx(bs)
```

The basis functions will be sorted in frequency order.

To transform the signal, divide it into vectors of length 4 and multiply the vectors with the transform matrix. In Matlab we can just as easily transform all the vectors at once like this:

```
>> y = A*reshape(x, bs, []);
```

The signal vector is reshaped into a matrix where each column is a 4-vector and then all columns are multiplied by the transform matrix. In the transformed matrix `y`, we have each transform component in one row.

To see how the signal energy is distributed among the transform components, just calculate the variances of them:

```
>> vars = var(y')
```
Compare this to the signal variance of the original signal:
```
>> varx = var(x)
```
In theory, the average value of the transform variances should be the same as the signal variance.

4 Zone Coding

For a given data rate, allocate bits to the different transform components so that the expected distortion is minimized. Design the quantizers, either by using `lbg` or `trainvq`, or simply take the Lloyd-Max quantizers from the formula collection (remember to scale the values appropriately).
Quantize the transformed signal, using the function `nearest` and measure the distorsion in the transform plane.
Do inverse transformation of the reconstructed transform and measure the distortion in the signal plane.
Also quantize the signal directly, without first transforming it, to the same rate as the transform coder, and measure the distortion. How much do you gain from the transform?
Try varying the size of the transform and see how the transform gain varies with transform size.

5 Discrete Cosine Transform

Also try DCT instead of Hadamard transform. In Matlab the function `dctmtx` can be used to create a DCT matrix:
```
>> A = dctmtx(bs)
>> y = A*reshape(x, bs, []);
```
Another option is to use the function `dct` which is a fast DCT implementation (similar to a FFT):
```
>> y = dct(reshape(x, bs, []));
```
Repeat the quantization experiments and compare the performance of the DCT coder to the Hadamard transform coder.

6 KLT

In order to do a KLT, you need to first estimate the auto correlation function (acf) for the signal. This will estimate the first 64 values of the acf:
```
>> M = 64;
>> Rxx = zeros(M,1);
>> for k=1:M
Rxx(k)=mean(x(1:end-k+1).*x(k:end));
end
```

Note that Matlab starts indexing vectors at 1, so the first value will actually be the acf in 0.
The correlation matrix is then given by:

```
>> Rx = toeplitz(Rxx(1:bs));
```

The transform matrix is then given by the eigenvectors to the correlation matrix:

```
>> [A dummy] = eig(Rx);
>> A = flipud(A');
```

The transposing is needed, since `eig` returns the eigenvectors as columns.