Applied Signal Processing 2017

Exercise 2

a) Signal filtering and compression with wavelets.

(i) The file `www.abo.fi/~htoivone/aspdata/nmr.dat` contains a noisy data sequence representing a nuclear magnetic resonance (nmr) spectrum.

Calculate wavelet transforms of the sequence using both the Haar discrete wavelet transform and Daubechies wavelets. Use hard and soft thresholds to remove noise and to compress the transformed signal. Determine the achieved compression when using the threshold value $d = 3$. Calculate the inverse Haar wavelet transform of the compressed signal. Plot the original signal and the signal which has been reconstructed from the compressed transform and evaluate the approximation accuracy of the compressed signal.

(ii) The signal in (i) contains peaks which make compression using Fourier or cosine transform inefficient, since a large number of frequency components are needed to represent the peaks correctly. Determine how well the signal can be approximated using its dominant Fourier transform components if the same compression as in (i) is required.

(iii) Use the Matlab command `'load handel'` to load the audio signal `y` and the associated sampling frequency `Fs`. Use the first 2^{16} samples and compute wavelet transforms with various resolution levels $J \leq 16$ and thresholds to achieve compression (suitable threshold values are in the range $d = 0.02 \cdots 0.07$). Reconstruct the signal from the compressed transforms and plot the samples for $n = 2001 \cdots 2100$ of both the original signal and the signals which have been reconstructed from the compressed transforms. Play the signals to determine whether the compressed signal can be considered acceptable.

b) Image processing using wavelets.

Apply 2-dimensional wavelet transforms with thresholds for image filtering and compression. Use both Haar and Daubechies wavelets, and find thresholds which achieve a suitable compromise between compression level and image quality. The following test cases can be used.

- Apply wavelet filtering to enhance quality of the image `supermies.jpg`, which is taken with a crappy digital camera, and is somewhat noisy.

- Apply wavelet filtering to compress the image `lenna.jpg`.

Matlab programs

`xdaub = dwtdaub(x, J, Hord)`
Computes Daubechies wavelet transform with `J` stages and filter order `2*Hord-1`.

`x = idwtdaub(xdaub, J, Hord)`
Computes inverse Daubechies wavelet transform.

`xdaub2 = dwtdaub2(x2, J, Hord)`
Computes 2-dimensional Daubechies wavelet transform.

`x2 = idwtdaub2(xdaub2, J, Hord)`
Computes inverse 2-dimensional Daubechies wavelet transform.

`[xL, xH] = daubstp(x, H)`
Computes one stage of Daubechies wavelet transform.

`xr = idaubstp(xL, xH, H)`
Computes one stage of inverse Daubechies wavelet transform.

`[xf, nround] = hardthreshold(x, epsilon)`
Applies hard threshold `epsilon` to signal `x`.

`[xf, nround] = softthreshold(x, epsilon)`
Applies soft threshold `epsilon` to signal `x`.

`[xf, nround] = softthreshold2(x, epsilon)`
Applies soft threshold to signal `x` with threshold components in vector `epsilon` applied to different subbands.

`a = hdaub(m)`
Generates the 2*m Daubechies wavelet filter coefficients corresponding to m= $1, 2, \ldots$.

Auxiliary programs

`xp = expandperiodic(x, M, dir)`
Periodic expansion of signal `x`.

`data = imgtodouble(imgname)`
Creates a floating point representation of an image.

`datauint = doubletoimg(data)`
Transforms a floating point image to uint8 representation of image

Data files

`nmr.dat`	Nuclear magnetic resonance data sequence.
`supermies.jpg, lenna.jpg`	Images.

Matlab Tips

Image data is loaded and viewed in 8-bit unsigned integer format, whereas filtering should be done in double precision floating point format. You can use the functions `x = imgtodouble('filename')` to load an image in floating point format and `img = doubletoimg(x)` to transform a double precision array to 8-bit unsigned integer array.

Images can be viewed with the command `image(img)`. However, you have to specify window size in order to view the image using correct aspect ratio. The following code creates an 8-bit unsigned integer array, displays the image and sets the image size and position on desktop:

```
% Display figure
img = doubletoimg(x); % create 8-bit unsigned integer array
image(img); % display figure
% Set figure window size
[n,m] = size(x);
left = 200; % window left side position on desktop
bottom = 200; % window bottom position on desktop
set(gcf, 'position', [left, bottom, 2*m, 2*n]);
```