Abstract

In this project some applications of wavelets are studied, making focus on the image compression. In order to do so, several computations using MATLAB are conducted.

1 Laboratory 6: Wavelets

Consider the Daubechies wavelets that have compact support and several vanishing moments. Thus in the smooth regions, many coefficients are close to zero. Let us compare it with a Wilson basis (time frequency shifts of a square integrable function) that is used in the standard jpeg compression. In order to do so the following code in MATLAB is implemented as an initial approach:

```
pkg load ltfat
% Use the cameraman image
f = cameraman;
% Ratio to keep
r=0.05;
%% Parameters for the Wilson systems
% Analysis window
ga='itersine';
% synthesis window
gs='itersine';
% No. of channels
M=8;
%% Parameters for the Wavelet system
% Analysis filters
wa='db6';
% Synthesis filters
ws='db6';
% No. of levels
J=5;
%% Show the original and
figure(1);
imagesc(f);
colormap(gray), axis('image');
%Compressed images
figure(2);
subplot(1,2,1);
c_fwt = fwt2(f,wa,J);
[cc_fwt,n]=largestr(c_fwt,r);
r_fwt =ifwt2(cc_fwt,ws,J);
imagesc(r_fwt);
colormap(gray), axis('image');
subplot(1,2,2);
c_wmdct = wmdct2(f,ga,M);
cc_wmdct = largestr(c_wmdct,r);
r_wmdct = iwmdct2(cc_wmdct,gs);
imagesc(r_wmdct);
colormap(gray), axis('image');
```

However, given the incompatibility with this Octave code and the packages loaded on my MATLAB editor (I had some problems trying to load the corresponding package from the direct source[1] and adding the path to the interface. Therefore, the compression and generation of the images was conducted using the alternative code:

```
% Add the LTFAT toolbox path
addpath('ltfat');
% Read the cameraman image
```

[1] https://github.com/ltfat/ltfat/releases/tag/v2.5.0

```matlab
f = imread('cameraman.tif');
% Ratio to keep
r = 0.05;
%% Parameters for the Wavelet system
% Analysis filters
wa = 'db6';
% Synthesis filters
ws = 'db6';
% No. of levels
J = 5;
%% Show the original image
figure(1);
imagesc(f);
colormap(gray);
axis('image');
title('Original Image');
%% Compressed images
figure(2);
% Perform the wavelet transform using 'wavedec2'
[c_fwt, S] = wavedec2(f, J, wa);
% Calculate the number of coefficients to keep based on the ratio
numCoeffsToKeep = round(r * numel(c_fwt));
% Sort the coefficients in descending order of magnitude
[~, sortedIndices] = sort(abs(c_fwt), 'descend');
% Keep the largest coefficients
cc_fwt = zeros(size(c_fwt));
cc_fwt(sortedIndices(1:numCoeffsToKeep)) = c_fwt(sortedIndices(1:
    numCoeffsToKeep));
% Reconstruct the image using the inverse wavelet transform 'waverec2'
r_fwt = waverec2(cc_fwt, S, ws);
subplot(1, 2, 1);
imagesc(r_fwt);
colormap(gray);
axis('image');
title('Wavelet Compression');
% Perform the DCT on the original image
c_dct = dct2(f);
% Keep the largest coefficients based on the ratio
numCoeffsToKeep_dct = round(r * numel(c_dct));
[~, sortedIndices_dct] = sort(abs(c_dct(:)), 'descend');
cc_dct = zeros(size(c_dct));
cc_dct(sortedIndices_dct(1:numCoeffsToKeep_dct)) = c_dct(sortedIndices_dct
    (1:numCoeffsToKeep_dct));
% Reconstruct the image using the inverse DCT 'idct2'
r_dct = idct2(cc_dct);
subplot(1, 2, 2);
imagesc(r_dct);
colormap(gray);
axis('image');
title('DCT Compression');
% Remove the LTFAT toolbox path
rmpath('ltfat');
```

The corresponding code is attached on the file `lab6_v2.m`. In this implementation the function `wavedec2` is used instead of the `fwt2` from the Wavelet Toolbox (which was already installed in a previous Lab).
Moreover, the function `largestr` used to keep the largest coefficients based on the specified ratio `r` is also replaced by an equivalent approach implemented to keep the largest coefficients based on the given ratio. The rest of the coefficients are set to zero in the `cc_fwt` array. Finally the image is reconstructed using the `waverec2` function.
Finally, the function `dct2` is used in order to implement the DCT-based compression

instead of the `wmdct2` function, providing similar results (they both compute the the weighted-modified discrete cosine transform (WMCDT)).

The obtained results are shown in Figures 1 and 2.

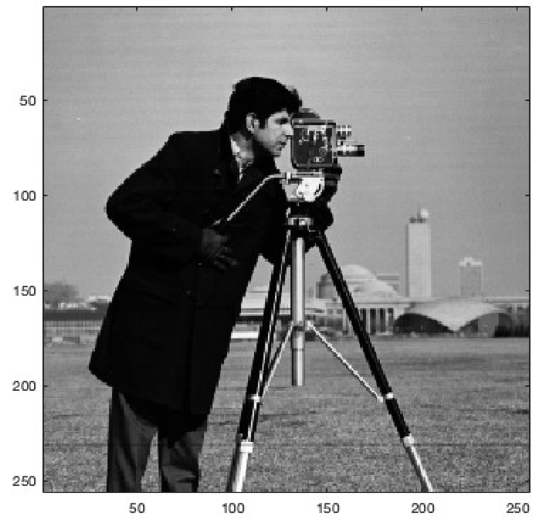


Figure 1: Original `cameraman.jpg` image.

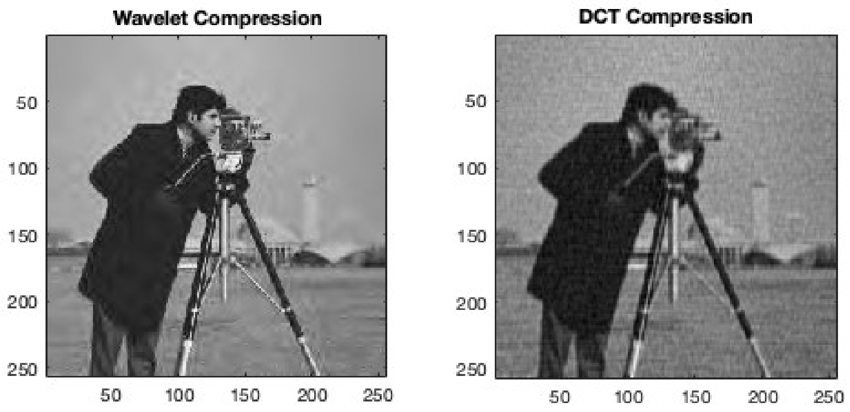


Figure 2: Reconstructed image using the wavelet and DCT compressions.

It is also possible to use the same principle to eliminate noise from an image or sound. Typically the signal is concentrated. In the next experiment, we will use the filter $H(z)$

to remove an undesirable white noise interference from a speech signal. To run the experiment, first read the file `easy.wav` that is provided with the instruction:

```
[x,FS] = audioread('easy.wav');
```

The vector x represents the audio signal and FS is an integer that denotes at which it has been sampled. Recall the sound instruction lines:

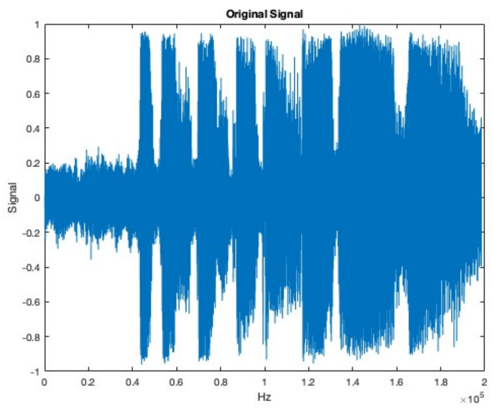
```
player = audioplayer (0.8*x, FS);
play (player);
```

We add some white noise to the signal (we need to load the package statistics):

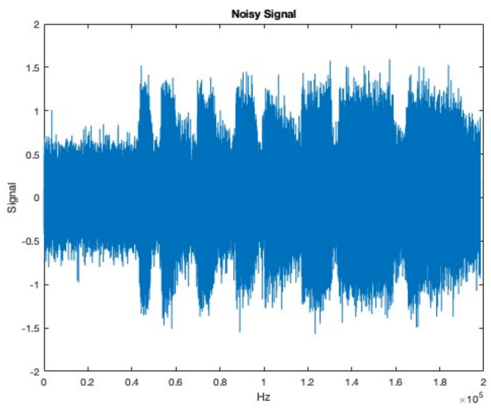
```
y = x + normrnd(0, 0.2, length(x),1);
player = audioplayer (y, FS);
play (player);
```

Exercise 1: *Slightly attenuate the noise, computing the discrete wavelet transform and keeping only the most significant coefficients. You should experiment with different ratios close to one.*

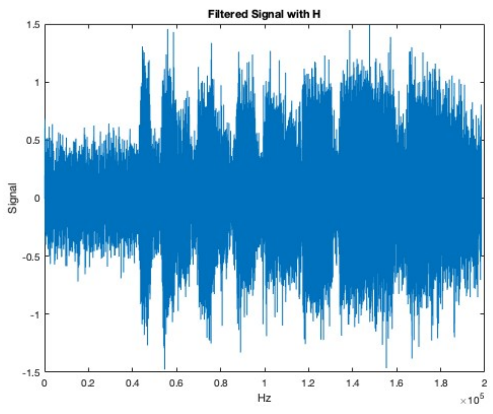
Using the previous code implementation we obtain different reconstructed noises by computing the discrete wavelet transform and keeping only the most significant coefficients. Using different ratios we get the results presented on the following Figures 3 and 4.



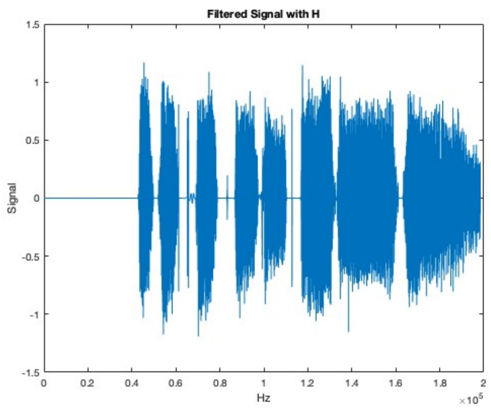
(a) Original signal.



(b) Noisy signal.

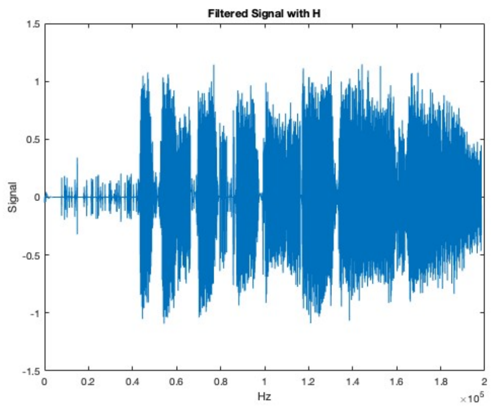


(c) $s = 0.05, J = 5$.

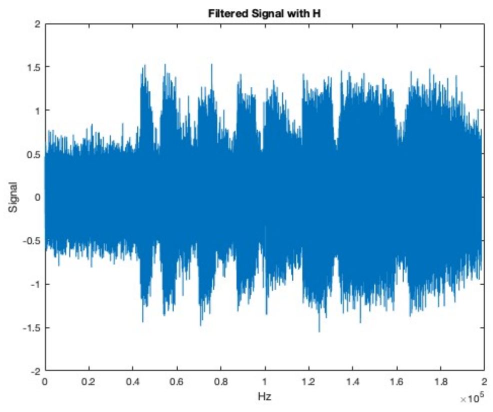


(d) $s = 0.01, J = 5$.

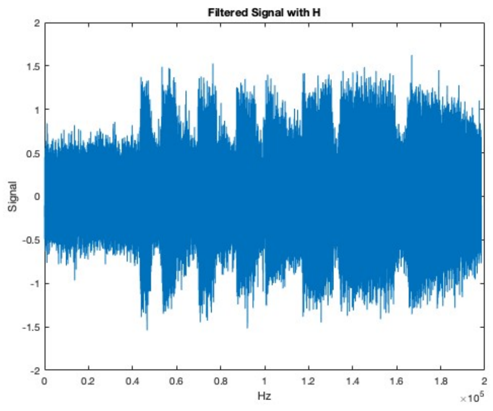
Figure 3: Representation of the sound signals for different parameters s.



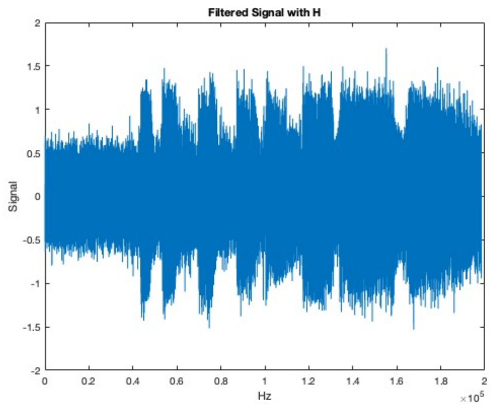
(a) $s = 0.015, J = 10$.



(b) $s = 0.5, J = 5$.



(c) $s = 0.75, J = 5$.



(d) $s = 0.9, J = 5$.

Figure 4: Representation of the sound signals for different parameters s.

Observe that for values of s close to 1 we get a closer signal to the noisy one, meanwhile when reaching s values close to 0 we get more filtered sound, that in the case for example that $s = 0.015$ approaches the original signal. The corresponding code is presented on the file `Lab6_v1.m`.