Denoising and Compression Using Wavelets

Juan Pablo Madrigal Cianci
Trevor Gianinni

December 15, 2016

Abstract

An explanation of the theory behind signal and image denoising and compression is presented. Different examples of image and signal denoising and image compression are implemented using `MATLAB`. Some of their characteristics are discussed. The project presents other implementations that were omitted, but we'll be happy to provide them with the code if you contact us at `jpmadrigalc@unm.edu`

1 introduction

1.1 Denoising

It is well known to any scientist and engineer who work with a real world data that signals do not exist without noise, either negligible or not negligible. However, there are many cases in which the noise corrupts the signals in significant manner, and it must be removed from the data in order to proceed with further data analysis. The process of noise removal is generally referred to as signal denoising or simply denoising.

Since the 1990s, wavelets have been found to be a powerful tool for removing noise from a variety of signals (denoising). They allow to analyze the noise level separately at each wavelet scale and to adapt the denoising algorithm accordingly. Wavelet thresholding methods for noise removal, in which the wavelet coefficients are thresholded in order to remove their noisy part, were first introduced by Donoho in 1993.

We consider a Gaussian additive noise model, meaning that if our noisy signal can be modeled as:

$$y(t) = x(t) + \epsilon_{\mu,\sigma}, \tag{1}$$

that is, our noisy signal (or image) is composed by a clean image plus some random noise with mean μ and standard deviation σ. The main idea is then to

decompose the wave using the DWT. Then we use a threshold for the coefficients and as such get rid of some of them. After obtaining the coefficients that we decide to keep, we the reconstruct our image using the IDWT.

1.2 Compression

The goal of compression is to store image data in as little space as possible in a file. Wavelet compression is a form of data compression well suited for image compression. Notable implementations are JPEG 2000 and DjVu.

Using a wavelet transform, the wavelet compression methods are adequate for representing transients, such as percussion sounds in audio, or high-frequency components in two-dimensional images, for example an image of stars on a night sky. This means that the transient elements of a data signal can be represented by a smaller amount of information than would be the case if some other transform, such as the more widespread discrete cosine transform, had been used.

The compression method is as follows. First, we use a wavelet transform. This will produce as many coefficients as there are pixels in the image. These coefficients can then be compressed more easily because the information is statistically concentrated in just a few coefficients. This principle is called transform coding. After that, the coefficients are quantized and the quantized values are entropy encoded and/or run length encoded.

A schematic of how the compression works is given by the following image, taken from MATLAB's wavelet toolbox user manual:

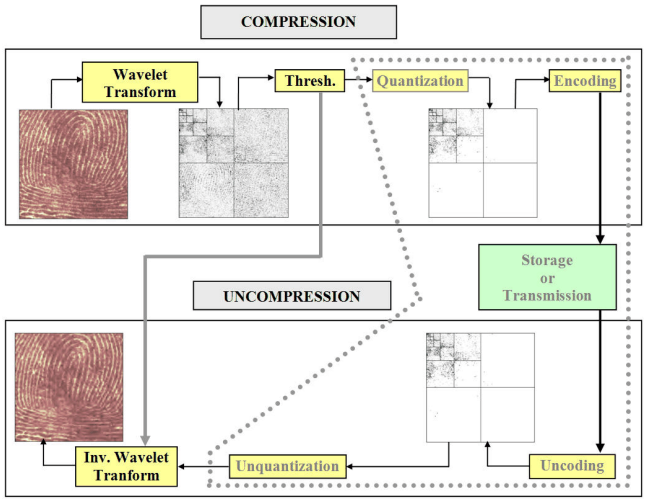


Figure 1: Schematic of the compression process.

2 Mathematical Aspects: Wavelets in 2-D:

Analogous to the one dimensional case, we have two-dimensional wavelets. As in the one dimensional case, we want to show that they form a complete orthonormal basis for $L^2(\mathbb{R}^2)$.

2.1 Proof that $\{\psi_{n,m}\}$ is an orthonormal basis in $L^2(\mathbb{R}^2)$.

Let $\psi_{n,m,j,k}(x)(y) := \psi_{m,k}(x)\psi_{m,j}(y)$, where $n, m, j, k \in \mathbb{Z}$. We know that $\psi_{n,k}(x)$ forms an orthonormal basis in $L^2(\mathbb{R})$. Thus we need to show that $\psi_{n,m,k,j}(x)(y)$ forms an orthonormal basis in $L^2(\mathbb{R}^2)$. Note that we can skip the k, j notation since unless we are on the same support, i,e, for some k, k', j, j' such that $k = k'$ and $j = j'$,

$$\langle \psi_{n,m,k,j}, \psi_{p,q,j',k'} \rangle = 0,$$

always! Thus we want to show that given a $f \in L^2(\mathbb{R}^2)$,

$$f(x,y) = \sum_{n\in\mathbb{Z}} \sum_{m\in\mathbb{Z}} \langle f, \psi_{n,m} \rangle \psi_{n,m}(x,y),$$

where $\psi_{n,m}(x,y) = \psi_n(x)\psi_m(y)$.

Define $f(x,y) := f_x(y)$. We start by fixing an $x \in \mathbb{R}$. Note that since $f \in L^2(\mathbb{R}^2)$ then $f(x,y) = f_x(y) \in L^2(\mathbb{R})$. for almost every $x \in \mathbb{R}$. Let

$$f_x(y) = \sum_{n\in\mathbb{Z}} \langle f_x, \psi_m \rangle \psi_m(y). \tag{2}$$

$$\implies f(x,y) = \sum_{n\in\mathbb{Z}} \langle f_x, \psi_m \rangle \psi_m(y), \tag{3}$$

$$\tag{4}$$

where the last equality is made in the $L^2(\mathbb{R})$ sense. Moreover, let

$$F_m(x) = \langle f_x, \psi_m \rangle = \int_{-\infty}^{\infty} f(x,z)\overline{\psi_m(z)}dz \in L^2(\mathbb{R}), \text{ since } f \in L^2(\mathbb{R}^2). \tag{5}$$

Moreover, since $\psi_n(x)$ is a basis in $L^2(\mathbb{R})$ and $f_m(x) \in L^2(\mathbb{R})$, we have that

$$F_m(x) = \sum_{n\in\mathbb{Z}} \langle F_m, \psi_n \rangle \psi_n(x). \tag{6}$$

We need to show that $F_m(x) \in L^2(\mathbb{R})$. To do so we compute

$$||F_m||^2 = \int_{-\infty}^{\infty} |F_m(x)|^2 dx = \int_{-\infty}^{\infty} |\int_{-\infty}^{\infty} f(x,z)\psi_m(z)dz|^2 dx \tag{7}$$

$$\leq \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} |f(x,z)|^2 dz \int_{-\infty}^{\infty} |\psi_m(z)|^2 dz \right) dx \tag{8}$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |f(x,z)|^2 dz dx = ||f(x,z)||^2 < \infty, \tag{9}$$

$$\implies F_m(x) \in L^2(\mathbb{R}), \tag{10}$$

where we used Cauchy-Schwarz from equation (7) to equation (8) and equation (8) is true because $||\psi_m|| = 1$.

Also, note that

$$\langle F_m, \psi_n \rangle_{L^2(\mathbb{R})} = \int_{-\infty}^{\infty} F_m(x)\overline{\psi_n(x)}dx = \tag{11}$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,z)\overline{\psi_m(z)\psi_n(x)}dzdx, \tag{12}$$

where we used Foubini's theorem between (11) and (12) Thus

$$f(x,y) = \sum_{m\in\mathbb{Z}} \left(\sum_{n\in\mathbb{Z}} \langle F_m, \psi_n \rangle_{L^2(\mathbb{R})} \psi_n(x) \right) \psi_m(y) \quad \text{(By eq. (6))} \tag{13}$$

$$= \sum_{m\in\mathbb{Z}} \sum_{n\in\mathbb{Z}} \langle f, \psi_m\psi_n \rangle_{L^2(\mathbb{R}^2)} \psi_n(x)\psi_m(y) \tag{14}$$

$$= \sum_{m\in\mathbb{Z}} \sum_{n\in\mathbb{Z}} \langle f, \psi_{n,m} \rangle_{L^2(\mathbb{R}^2)} \psi_{m,n}(x,y) \tag{15}$$

$$\tag{16}$$

$\implies \psi_{m,n}(x,y)$ is a complete system in $L^2(\mathbb{R}^2)$.

Thus having show that it is indeed a basis, we want to show that it's also orthonormal, i.e,

$$\langle \psi_{n,m}, \psi_{p,q} \rangle = \delta_{(p,q),(n,m)}. \tag{17}$$

To do so consider

$$\langle \psi_{n,m}, \psi_{p,q} \rangle = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \psi_p(x)\psi_q(y)\overline{\psi_n(x)\psi_m(y)}dxdy \tag{18}$$

$$= \int_{-\infty}^{\infty} \psi_q(y)\overline{\psi_m(y)} \left(\int_{-\infty}^{\infty} \psi_p(x)\overline{\psi_n(x)}dx \right) dy \tag{19}$$

$$= \int_{-\infty}^{\infty} \psi_q(y)\overline{\psi_m(y)}\delta_{p,n}dy = \delta_{p,n} \int_{-\infty}^{\infty} \psi_q(y)\overline{\psi_m(y)}dy \tag{20}$$

$$= \delta_{p,n}\delta_{q,m} = \delta_{(p,q),(n,m)} \tag{21}$$

Therefore, we have that $\{\psi_{n,m}\}$ is an orthonormal basis in $L^2(\mathbb{R}^2)$.

2.2 Proof that they form an Orthogonal MRA

Let $V_j = v_j \otimes v_j$. Naturally, this means that $V_{j+1} = v_{j+1} \otimes v_{j+1}$. We want to show $V_j \subseteq V_{j+1}$. We assume that we are in an FIR. Thus it is sufficient to show that the basis elements of V_j are in V_{j+1}.

Let's examine the basis elements of V_j.

$$V_j = \overline{\text{span}\{\phi_{j,k}(x)\phi_{j,n}(y)\}} = \text{span}\{\phi_{j,k}(x)\phi_{j,n}(y)\} \quad (22)$$

$$= \text{span}\{\phi_{j,k,n}(x,y)\} = \{\sum_{n\in\mathbb{Z}}\sum_{k\in\mathbb{Z}} a_{j,k,n}\phi_{j,k,n}(x,y) | \sum_{n\in\mathbb{Z}}\sum_{k\in\mathbb{Z}} |a_{j,k,n}|^2 < \infty\}. \quad (23)$$

Where the right hand side of (22) is closed since we have all possible products here, including the limit of sequences. Moreover, we have that $\phi_{j,k,b}(x,y)$ are the basis elements.

Without loss of generality, we assume that $j = 0$, otherwise the arguement is identical but the notation might vary slightly. As we know, the scaling functions in 1-D were $\phi(x)$ and $\phi(y)$. We define our scaling function in 2-D as $\phi(x,y) := \phi(x)\phi(y)$. Clearly, since $\phi(x), \phi(y) \in V_0$, then $\phi(x,y) \in V_0 \otimes V_0$.

Recall that $\phi_{j,n}(x) = 2^{J/2}\phi(2^J x - n)$. This in turn implies that

$$\phi_{j,n,k}(x,y) = 2^J\phi(2^J x - n)\phi(2^J y - k) = 2^J\phi(2^j x - m, 2^J y - k) \quad (24)$$

$$\implies \phi_{0,n,k}(x,y) = \phi(x-n, y-k), \quad (25)$$

for $j = 0$. We thus need to show that $\phi_{0,n,k}(x,y) \in V_1 \otimes V_1$. Recall that $\{\phi_{0,n}(x)\}$ is an orthonormal basis for V_1 and that the set of scaled-translates of $\phi(2x)$, $\{2^{1/2}\phi(2x-n)\}$ form an orthonormal basis for V_1. Thus a basis for $V_0 \otimes V_0$ is given by

$$\{\phi_{1,k}(x)\phi_{1,n}(y)\} = \{\phi_{1,n,k}(x,y)\} = \{2\phi(2x-n, 2y-k)\} \quad (26)$$

Thus, we need to show that

$$\phi(x-n, y-k) = 2\sum_{p\in\mathbb{Z}}\sum_{q\in\mathbb{Z}} h_{p,q}\phi(2x-p, 2y-p)$$

, which is a finite sum since we assumed that we had an FIR. Thus, $\phi(x) = \phi_{0,0}(x) = \sum_{p\in\mathbb{Z}} h_p \phi_{1,p}(x)$ is a finite sequence. This in turn implies that

$$\phi_{0,n}(x) = \phi(x-n) = \sum_{p\in\mathbb{Z}} h_p \phi_{1,p}(x-n) = \sum_{p\in\mathbb{Z}} h_p 2^{1/2} \phi(2(x-n)-p) \tag{27}$$

$$= \sum_{p\in\mathbb{Z}} h_p 2^{1/2} \phi(2x-(2n+p)). \text{ Let } r = 2n+p \implies p = r-2n \tag{28}$$

$$\sum_{r\in\mathbb{Z}} h_{r-2n} \phi(2x-r) = \sum_{r\in\mathbb{Z}} h_{r-2n} \phi_{1,r}(x) \tag{29}$$

$$\implies \phi_{0,n}(x) = \sum_{r\in\mathbb{Z}} h_{r-2n} \phi_{1,r}(x). \tag{30}$$

Similarly, $\phi_{0,k}(x) = \sum_{s\in\mathbb{Z}} h_{s-2k} \phi_{1,s}(y)$, which implies that

$$\phi_{0,n,k}(x,y) = \phi_{0,n}(x)\phi_{0,k}(y) = \sum_{r\in\mathbb{Z}} \sum_{r\in\mathbb{Z}} h_{r-2n} h_{s-2k} \phi_{1,r}(x) \phi_{1,s}(y) \tag{31}$$

$$= \sum_{r\in\mathbb{Z}} \sum_{r\in\mathbb{Z}} h_{r,s,n,k} \phi_{1,r,s}(x,y) \tag{32}$$

$$\implies \phi_{0,n,k} \in V_1 \otimes V_1. \tag{33}$$

Now let's show that the orthogonal complement of V_j is W_j. Note that the rules of arithmetic hold for direct sums and tensor products.

$$V_{j+1} = v_{j+1} \otimes v_{j+1} = (v_j \oplus w_j) \otimes (v_j \oplus w_j) \tag{34}$$

$$= (v_j \otimes v_j) \oplus ((v_j \otimes v_j) \oplus (v_j \otimes w_j) \oplus (w_j \otimes v_j) \oplus (w_j \otimes w_j)) \tag{35}$$

$$= (v_j \otimes v_j) ((v_j \otimes w_j) \oplus (w_j \otimes v_j) \oplus (w_j \otimes w_j)) \tag{36}$$

$$V_j \oplus W_j. \tag{37}$$

Since W_j is the direct sum of three tensor products we need three wavelets to span the detail space. We need one to span V_j, thus, we need four in total:

$$\text{for } (v_j \otimes v_j) \implies \varphi(x)\varphi(y) \tag{38}$$

$$\text{for } (v_j \otimes W_j) \implies \varphi(x)\phi(y) \tag{39}$$

$$\text{for } (W_j \otimes v_j) \implies \phi(x)\varphi(y) \tag{40}$$

$$\text{for } (W_j \otimes W_j) \implies \phi(x)\phi(y). \tag{41}$$

Define the Haar basis in 2D, $\varphi(x,y) = \chi_{[0,1]^2}(x,y) = \chi_{[0,1]}(x)\chi_{[0,1]}(y)$. The two dimensional MRA process is given by:

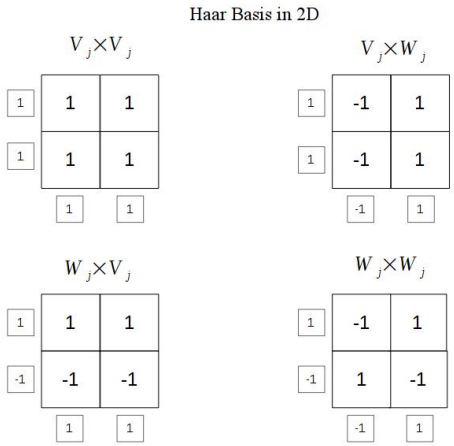


Figure 2: 2D Haar

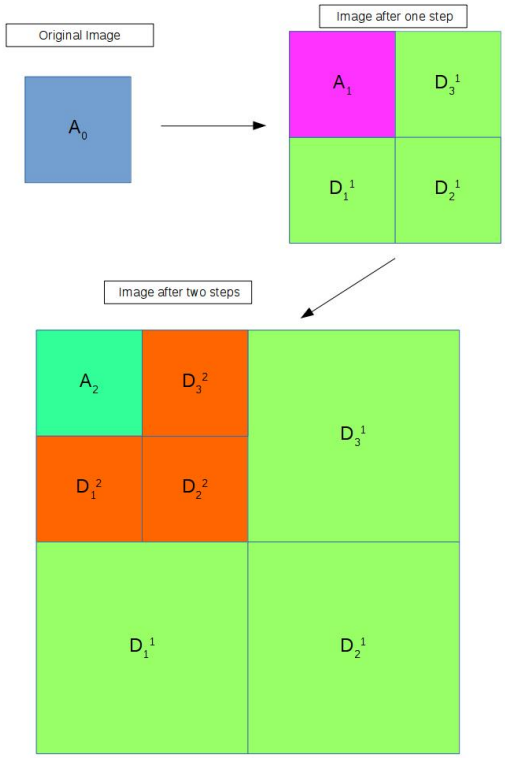


Figure 3:

2.3 Fast Wavelet Transform

The Fast Wavelet Transform is a mathematical algorithm designed to turn a waveform or signal in the time domain int o a sequence of coefficients based on an orthogonal basis of small finite waves, or wavelets. The transform can be easily extended to multidimensional signals, such as images, where the time domain is replaced with the space domain.

It has as theoretical foundation the device of a finitely generated, orthogonal multiresolution analysis (MRA). In the terms given there, one selects a sampling scale J with sampling rate of 2^J per unit interval, and projects the given signal f onto the space V_J; in theory by computing the scalar products

$$s_n^{(J)} = 2^J \langle f(t), \phi(2^J t - n) \rangle. \tag{42}$$

where ϕ is the scaling function of the chosen wavelet transform; in practice by any suitable sampling procedure under the condition that the signal is highly oversampled, so

$$P_J[f](x) = \sum_{n\in\mathbb{Z}} s_n^{(J)} \phi(2^J x - n)$$

is the orthogonal projection or at least some good approximation of the original signal in V_J. The MRA is characterized by its scaling sequence $a = (a_{-N}, \ldots, a_0, \ldots, a_N)$ and its wavelet sequence $b = (b_{-N}, \ldots, b_0, \ldots, b_N)$ (some coefficients might be zero). Those allow to compute the wavelet coefficients $d_n^{(k)}$, at least some range $k = M, ..., J-1$, without having to approximate the integrals in the corresponding scalar products. Instead, one can directly, with the help of convolution and decimation operators, compute those coefficients from the first approximation $s^{(J)}$.

2.3.1 Mallat's Algorithm

In 1988, Mallat produced a fast wavelet decomposition and reconstruction algorithm. The Mallat algorithm for discrete wavelet transform (DWT) is, in fact, a classical scheme in the signal processing community, known as a two-channel subband coder using conjugate quadrature filters or quadrature mirror filters (QMFs). Denote A_i and D_i as the i^{th} approximate and detailed coefficients respectively.

1. he decomposition algorithm starts with signal s, next calculates the coordinates of A_1 and D_1, and then those of A_2 and D_2, and so on.

2. The reconstruction algorithm called the inverse discrete wavelet transform (IDWT) starts from the coordinates of A_J and D_J, next calculates the coordinates of A_{J-1}, and then using the coordinates of A_{J1} and D_{J1} calculates those of A_{J2}, and so on. More precisely, the algorithm is given by

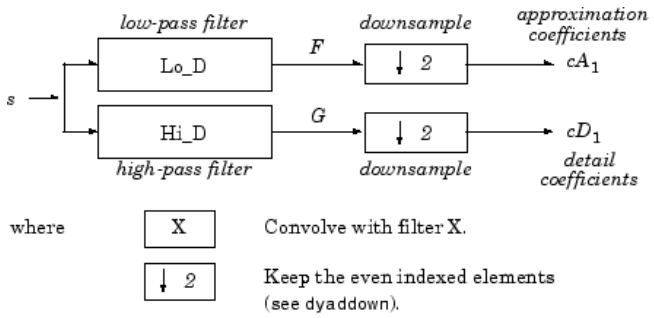


Figure 4: Schematic of Mallat's algorithm.

3 Introduction to MATLAB's Wavelet Toolbox

MATLABhas its own in-built functions to apply wavelets to signals, images, etc. Moreover, they also include an app that implements these functions, but we

found out that it is usually more efficient and easier to generalize to just use the inbuilt functions and write the script. A good source of information for the Wavelet toolbox app can be found at https://www.mathworks.com/help/wavelet/index.html. The wavelet toolbox includes different wavelets to work with. Among them are Haar, Doubechies, symlets (almost symmetric wavelet), and many more. Some of these wavelets and their scaling functions look like the following picture:

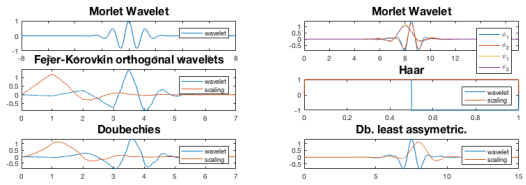


Figure 5: Graph of different wavelets used throughout the project

In order to find more information on these wavelets, we just need to type `waveinfo` from the `MATLAB`command bar. We now show the codes that we used for the project.

3.1 Codes for the 1D Case

The code for the one dimensional denoising is fairly simple. It just consist of the `MATLAB`function `XD=wden(X,TPTR,SORH,SCAL,N,WNAME)` , which returns a denoised version `XD` of the input signal `X` obtained by thresholding the wavelet coefficients. `TPTR` is the threshold selection rule specified as a string, `SORH` specifies soft or hard thresholding with `'s'` or `'h'`. `SCAL` defines the type of threshold rescaling `N` is the level of the wavelet transform and `WNAME` is the wavelet (i.e, Haar,...). An implementation of this code is as follows:

Listing 1: Implementation of Signal Denoising

```
1          %let yn be a noisy signal; audio, electric,
               etc.
2          %wden performs the denoising
3          yden = wden(yn,'rigrsure','s','mln',4,'sym4'
               );
4          %plots the results
5      figure(121)
6      plot(yn); title('noisy signal');
7      figure(122)
8      plot(yn); title('denoised signal');
```

3.2 Codes for the 2D Case

The code for two dimensional case is a little more complicated. Initially, we want to convert our image into a matrix, which is done by the MATLABcommand `I=imread('name of file.ext')`. After doing this, we convert our image into gray scale by using `I=rgb2gray(I)`. After loading and processing our image, we obtain the decomposition coefficients by `[C,S] = wavedec2(J,level,wname)`, which corresponds to a wavelet packet decomposition of the matrix X, at level N, with a particular wavelet. We then use `thr = wthrmngr('dw2ddenoLVL',...'penalhi',C,S,1)` in order the obtain the threshold, and, finally, we use `[XDEN,cfsDEN,dimCFS] = wdencmp('lvd',C,S,wname,level,thr,sorh)` in order to perform the denoising. A script for the implementation is given by:

Listing 2: Implementation of Image Denoising

```
%loads a noisy picture
J=imread('noisy.png');
level = 10;
%converts it to gray scale
J=rgb2gray(I);
%Let's make it noisy (Gaussian white noise)
%obtain the wavelet transform
%of a noisy
%image down to level 5 using a
%biorthogonal spline wavelet.
wname = 'haar';
level = 10;
[C,S] = wavedec2(J,level,wname);

% to see other wavelets that can be called uncomment
    "
% help waveinfo

%Obtain denoising (wavelet shrinkage) thresholds.
    Use the Birge-Massart strategy with a tuning
    parameter of 3.
thr = wthrmngr('dw2ddenoLVL','penalhi',C,S,1);
sorh = 's';
%Performs the denosing.
[XDEN,cfsDEN,dimCFS] = wdencmp('lvd',C,S,wname,level
    ,thr,sorh);
```

3.3 Compression

The compression algorithm is fairly similar to the image denoising algorithm.

1. Do the 2-dimensional wavelet composition using `[c l] = wavedec2(x,n,w)`
2. Perform the compression using
   `[xd,cxd,lxd,perf0,perfl2] = wdencmp(opt,c,l,w,n,thr,sorh,keepapp);`

When implemented, this code looks like

Listing 3: Implementation of Image Compression

```
[xx,map] = imread('audrey.jpg') ; %reads Image
xx=rgb2gray(xx);% Makes Image in grayscale
colormap(map)
n = 5;                    % Decomposition Level
w = 'sym8';               % Near symmetric wavelet
[c l] = wavedec2(xx,n,w); % Multilevel 2-D wavelet
    decomposition.
%he WDENCMP function performs a compression process
    from the wavelet decomposition structure [c,l]
    of the image.
%Does Various compressions with different thresholds
for i=1:5
opt = 'gbl'; % Global threshold
thr = 200*i;    % Threshold
sorh = 'h';  % Hard thresholding
keepapp = 1; % Approximation coefficients cannot be
    thresholded
[xd,cxd,lxd,perf0,perfl2] = wdencmp(opt,c,l,w,n,thr,
    sorh,keepapp);
colormap(map)
%plot results in same figure
subplot(3,2,i),imshow(xd,map)
title(['Compressed. Threshold = ' num2str(200*i)])
colormap(map)
end
subplot(3,2,6), imshow(xx, map)
title('Original Image')
```

4 Implementations

4.1 One Dimensional Signals

We know put our knowledge into use. Let's start by considering a simple example: a signal with some noise associated to it. We use a symlet as well as

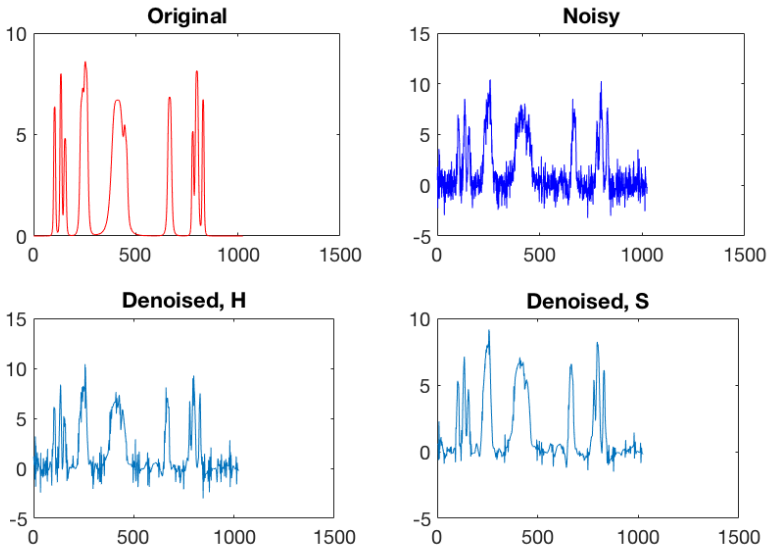


Figure 6: First example: a bumpy signal denoised by a symlet, using both a soft threshold (bottom right) and a hard threshold (bottom left)

As we can see from Figure 6, the soft threshold seems to work better than a hard threshold. Moreover, note that not all the noise was able to be eliminated from the signal, which is something that we should expect. Note that this is the case for a symlet, which is way smoother than, say, the Haar wavelet. Had we used this wavelet the denoised version would not have been very smooth, as we can see in Figure 7 bellow

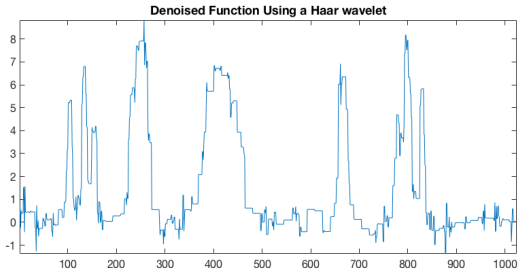


Figure 7: Note how this figure is far less smooth than the bottom two picfure on Figure 7

We continue our application of signal denoising and now consider a real world example: an electrocardiogram (ECG). In practice, physicians are interested in

finding 4 peaks denoted by PQRS, which denote the phases of the heartbeat[1]. We obtained a raw signal for the ECG. Given their electronical characteristics, these devices tend to be somewhat sensitive to noise. Performing a denoising on the raw signal using a symlet and a soft thresholding we get that the constrat between the raw and denoised signal is given by Figure 8:

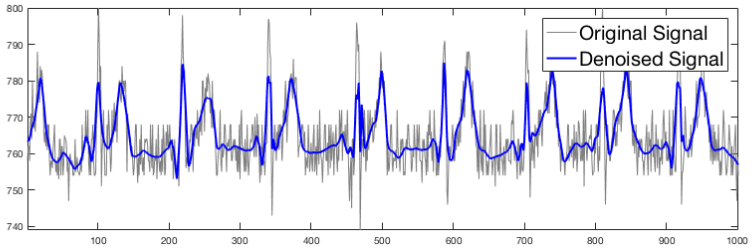


Figure 8: We can see the noisy signal (gray) and the denoised signal. As we can see, there's no much information that can be obtained from the noisy signal!

Moreover, doing the same denoising but with other wavelets yields Figure 9. From this figure we can see how different wavelets behave for the same task. Note again that the Haar wavelet is much less smooth than Daubechis and the Biorthogonal.

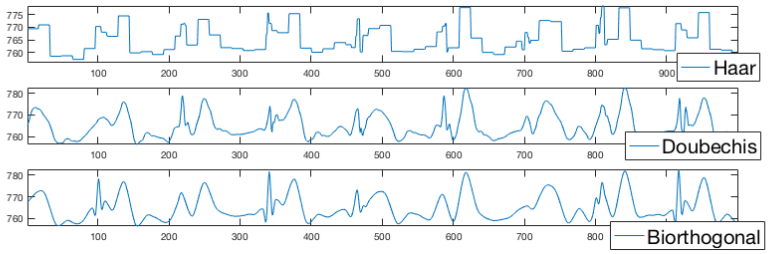


Figure 9:

Let us now consider 2 dimensional denoising on Images.

4.2 Two-Dimensional Denoising

We are now interested in showing the denoising results for 2 dimensional signals, i.e, images. Let's start with a (rather unpleasant) example. Consider the following noise image:

[1] see http://www.practicalclinicalskills.com/reading-ekg

Figure 10: A picture of the president-elect on a rather windy day.

If we denoise the image using a bi-orthogonal wavelet and a soft thresholding, we obtain the following:

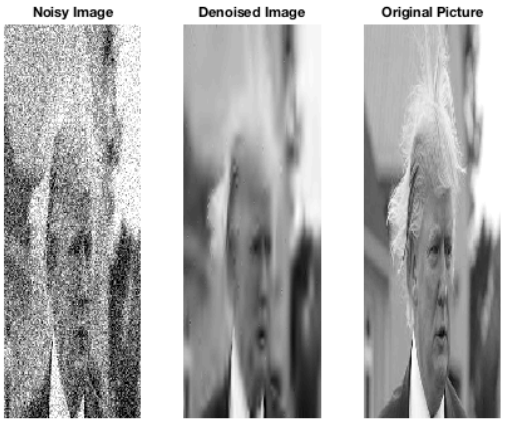


Figure 11: Comparisson between noisy, denoised and original..

Note from Figure 11 even thought the noisy image looks good, some information is lost while denoising it, which makes it seem blurry. Again, comparing against a denosing made with Haar wavelet we get the following:

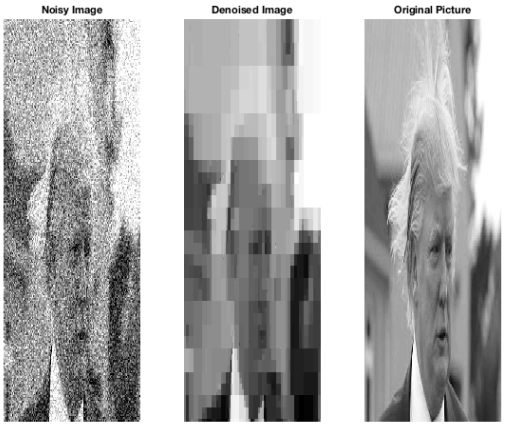


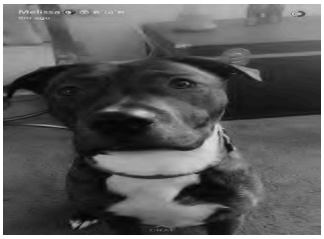
Figure 12: Trump denoised using a Haar transform.

As we can see from Figure 12, it is almost impossible to distinguish the denoised picture, and as such, the denoising procedure creates a more *cryptic* image than the noisy one (although noise-free). Denoising with a variety of wavelets on a picture that has both more resolution and more contrast is shown in the next page.

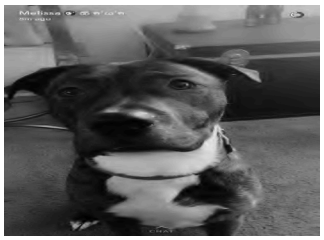
Haar

Daubechis

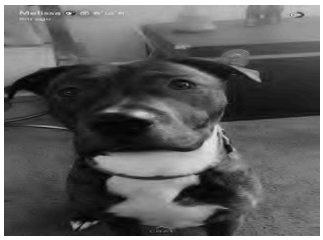
Symlet



Biorthogonal

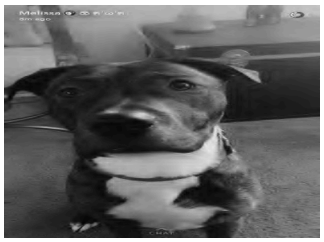


Rev. Biorth

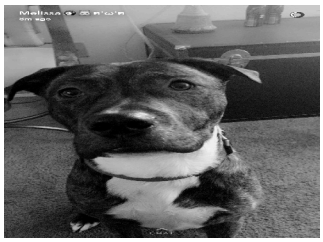


Disc Meyer

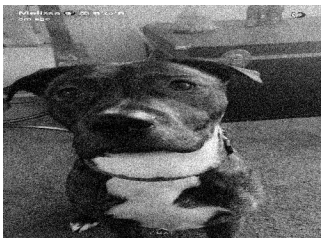
Fejer-Korovkin



Original Image



Noisy Image



Note that the Haar and Daubechis wavelet become somewhat pixelated. Moreover, note that, while the more complicated waves (such as the bi-orthogonal) get really similar to the original image, they loose a little bit of contrast with respect to the original picture. This is more evident in the carpeted part of the picture.

Finally, this can also be applied to videos, because videos are nothing else but a collection of images, thus, if we manage to extract the frames of the video (about 30 to 60 pictures per second of video), we can denoise them individually and as such clean a video. moreover, if we extract the sound file of the video, we can also denoise this, By using it as a one dimensional signal. All of this can be done using MATLABand the methods described in section 3!

4.3 Image Compression

Consider the following image to which we will perform compression with different threshold

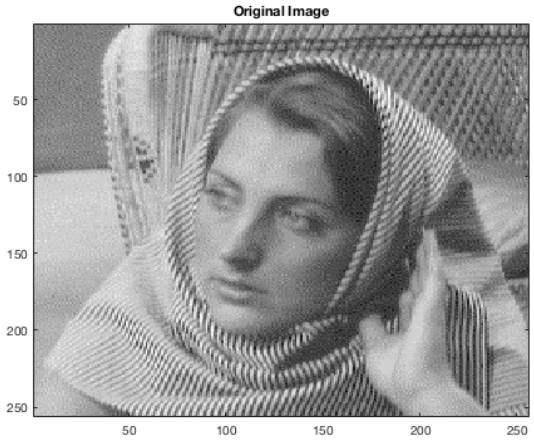


Figure 13: Image from the MATLAB.

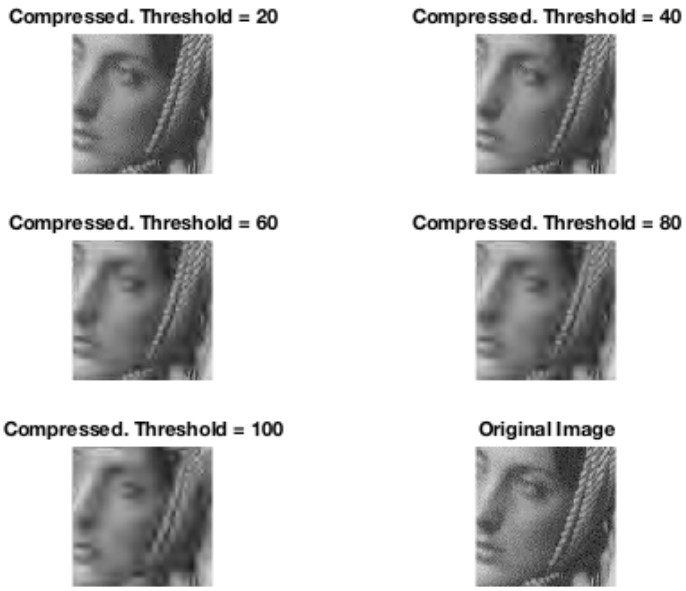


Figure 14: Different thresholding levels

Now, let's consider a different picture with a higher resolution.

Figure 15: A higher resolution picture of Audrey Hepburn.

Compressing this image for higher thresholding values we get that

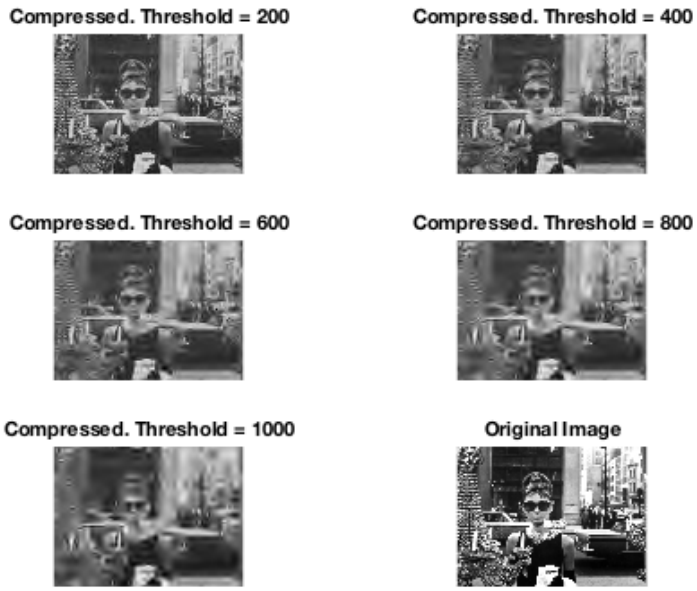


Figure 16: Compression for different threshold levels.

References:

- Pereyra, Mara Cristina, and Lesley A. Ward. Harmonic analysis: from Fourier to wavelets. Vol. 63. American Mathematical Soc., 2012.
- Mallat, Stphane G., and Zhifeng Zhang. "Matching pursuits with time-frequency dictionaries." IEEE Transactions on signal processing 41.12 (1993): 3397-3415.
- Rami Cohen. Signal Denoising Using Wavelets. 2012. Technion, Technical Institute of Israel.
- Misiti, Michel, et al. "Matlab Wavelet Toolbox Userś Guide. Version 3." (2004).
- Donoho, David L. "Wavelet shrinkage and WVD: A 10-minute tour." Presented on the International Conference on Wavelets and Applications, Tolouse, France. 1992.