University of Burgundy
Image Analysis , MSCV 2018

Sparse Visual Tracking

Ioannis Tsagkatakis
Hassan Saeed

December 23, 2018

1 Introduction

Mathematical transformations are applied to signals to obtain further information from that signal that is not readily available in the raw signal. There are a number of transformations that can be applied, among which the Fourier transforms are probably by far the most popular.

Matlab Code

The code this report as well as the results can be found in github at https://github.com/jtsagata/sparce_waveltes. Please look there for last minuted fixes and additions. Almost all the images of this report have been created by the authors using `Matlab`. To create the images is as simple as running the appropriate script.

1.1 Why Wavelet Transform

The Fourier transform gives the frequency components (spectral) exists in the signal. But, when the time-localization of spectral components is

needed, a transform giving the TIME-FREQUENCY representation of signal is needed. This is in short, if we take the Fourier transform over the whole time axis, we cannot tell at what instant a particular frequency rises. The wavelet transform is a transform which gives this sort of information. There are other transforms which give this information too, like Short-time Fourier transform (STFT) uses a sliding window to find spectrogram, which gives the information of both time and frequency. But still another problem exists: The length of window limits the resolution in frequency. Wavelet transform seems to be a solution to the problem above.

1.2 Discrete Wavelet Transform

The discrete wavelet transform (DWT), provides sufficient information both for analysis and synthesis of the original signal, with a significant reduction in the computation time. The DWT is considerably easier to implement when compared to the Continuous Wavelet Transform (CWT). The main idea is the same as it is in the Continuous Wavelet Transform. A time-scale representation of a digital signal is obtained using digital filtering techniques. The CWT is a correlation between a wavelet at different scales and the signal with the scale (or the frequency) being used as a measure of similarity. The continuous wavelet transform is computed by changing the scale of the analysis window, shifting the window in time, multiplying by the signal, and integrating over all times. In the discrete case, filters of different cutoff frequencies are used to analyze the signal at different scales. The signal is passed through a series of high pass filters to analyze the high frequencies, and it is passed through a series of low pass filters to analyze the low frequencies. The resolution of the signal, which is a measure of the amount of detail information in the signal, is changed by the filtering operations, and the scale is changed by up sampling and down sampling operations. Down sampling a signal corresponds to reducing the sampling rate, or removing some of the samples of the signal. Up sampling a signal corresponds to increasing the sampling rate of a signal by adding new samples to the signal.

1.3 Wavelet Analysis (Decomposition

The procedure starts with passing the signal through a half band digital low pass filter with impulse response. A half band low pass filter removes all frequencies that are above half of the highest frequency in the signal. For example, if a signal has a maximum of 500 Hz component, then half band low pass filtering removes all the frequencies above 250 Hz. After

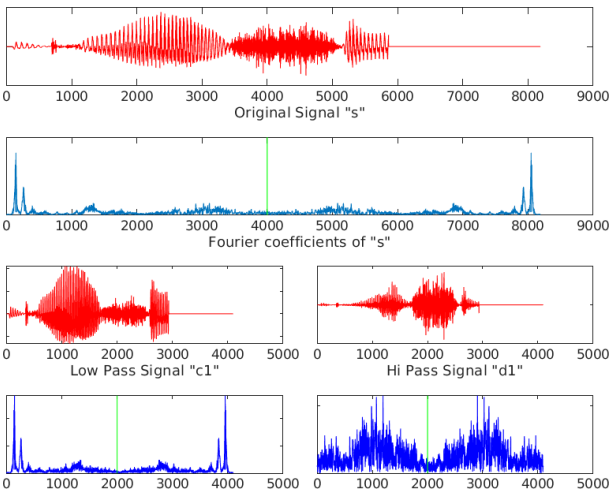

Figure 1: Wavelet Composition of 1D signal

passing the signal through a half band low pass filter, half of the samples can be eliminated according to the Nyquist's rule, since the signal now has a highest frequency of p/2 radians instead of p radians (if p is original highest frequency of signal). Simply discarding every other sample will down sample the signal by two, and the signal will then have half the number of points. The scale of the signal is now doubled. Here, the low pass filtering removes the high frequency information, but leaves the scale unchanged. Only the down sampling process changes the scale. The DWT analyzes the signal at different frequency bands with different resolutions by decomposing the signal into a coarse approximation and detail information. DWT employs two sets of functions, called scaling functions and wavelet functions, which are associated with low pass and high pass filters, respectively.

The decomposition of the signal into different frequency bands is simply obtained by successive high pass and low pass filtering of the time domain signal. The original signal x[n] is first passed through a half band high pass filter g[n] and a low pass filter h[n]. After the filtering, half of the samples can be eliminated. This constitutes one level of decomposition. This decomposition can be continued to multi level using the coarse approximation as input to next level.

The decomposition process can be seen in a block diagram in Figure 4.

In figure 1 created by matlab script `demo_1D_composition.m` the ...

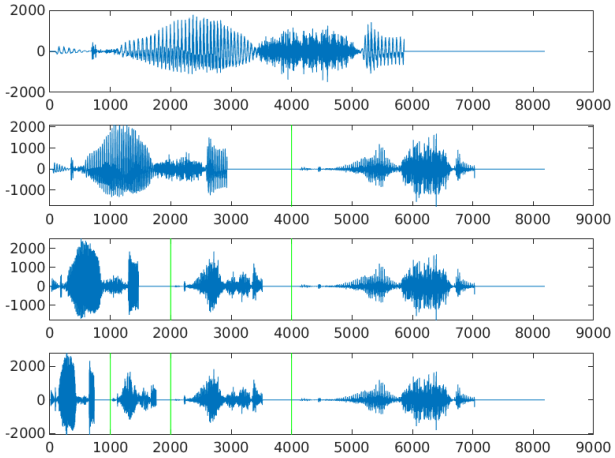

Figure 2: Wavelet decomposition of 1D signal

2 Wavelet Synthesis (Reconstruction)

The reconstruction of original signal from the wavelet coefficients is the reverse process of the decomposition. As we get the wavelet coefficients, we need to up sample the coefficients which halves the scale of the signal. After the signal is up sampled we apply low pass and high pass filters to the up sampled signal to get the perfect reconstructed signal.

3 2D DWT

For images, an algorithm similar to the one-dimensional case is possible for two-dimensional wavelets. This kind of two-dimensional DWT leads to a decomposition of approximation coefficients at level j in four components: the approximation at level $j+1$, and the details in three orientations (horizontal, vertical, and diagonal). The schematic diagram of 2D wavelet analysis (decomposition) is as shown in Figure 5.
The algorithm is simple

1. Apply $1D$ wavelet transform to each row
2. Apply $1D$ wavelet transform to each column of the result

The components of a wavelets transform ... to directional edges of the image. This is demonstrated on Figure 6 on page 7.
Figure 5 outlines the process.

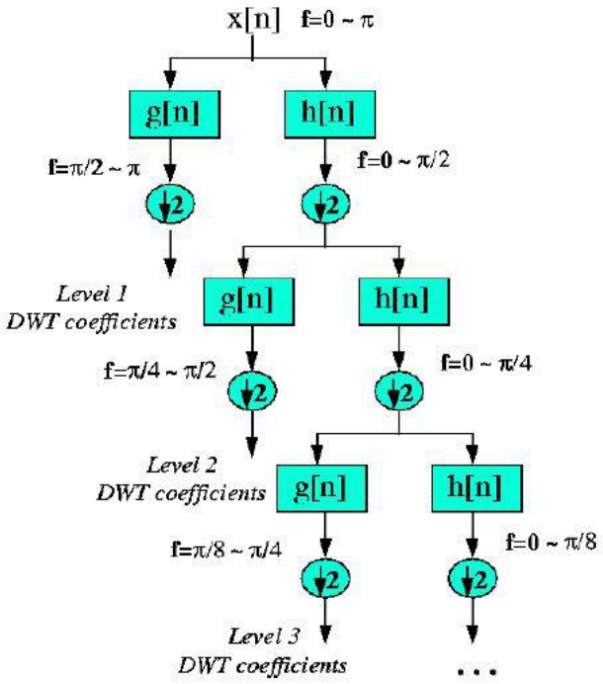

Figure 3: Wavelet Decomposition of 1D signal

4 Denoising

The data sets of many scientific experiments are corrupted with noise, either because of the data acquisition process, or because of environmental effects. A first pre-processing step in analyzing such datasets is denoising, that is, estimating the unknown signal of interest from the available noisy data. There are several different approaches to denoise signals and images. Generally smoothing removes high frequency and retains low frequency (with blurring). De-blurring increases the sharpness signal features by boosting the high frequencies, whereas denoising tries to remove whatever noise is present regardless of the spectral content of a noisy signal. Wavelet transforms enable us to represent signals with a high degree of sparsity. This is the principle behind a non-linear wavelet based signal estimation technique known as wavelet denoising. Wavelet denoising attempts to remove the noise present in the signal while preserving the signal characteristics, regardless of its frequency content.

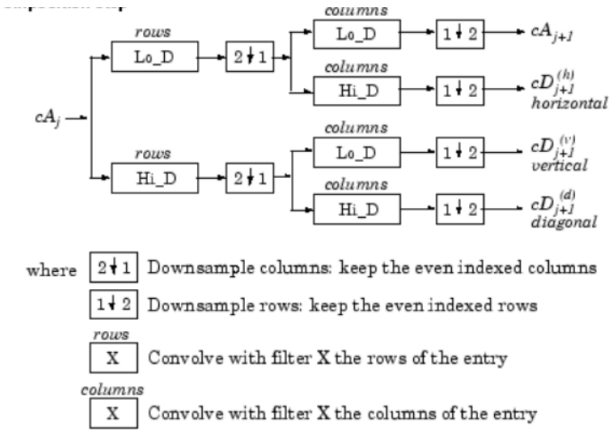

Figure 4: Schematic diagram of 2D DWT decomposition

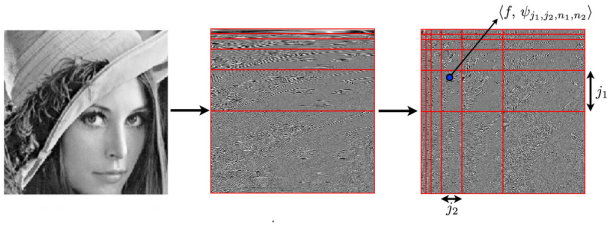

Figure 5: 2D Decompositionusing wavelets

5 Study of a Harr Wavelet Filter

The code this report as well as the results can be found in github at https://github.com/jtsagata/sparce_waveltes.

$$H_o(z) = \frac{1}{\sqrt{2}}\left(1 + z^{-1}\right)$$

6 Denoising

7 Inpaining Sparse

8 Inpaining 2

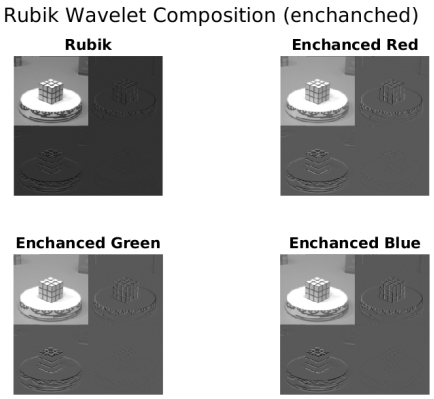

Figure 6: Enchanced 2D to make edges more visible

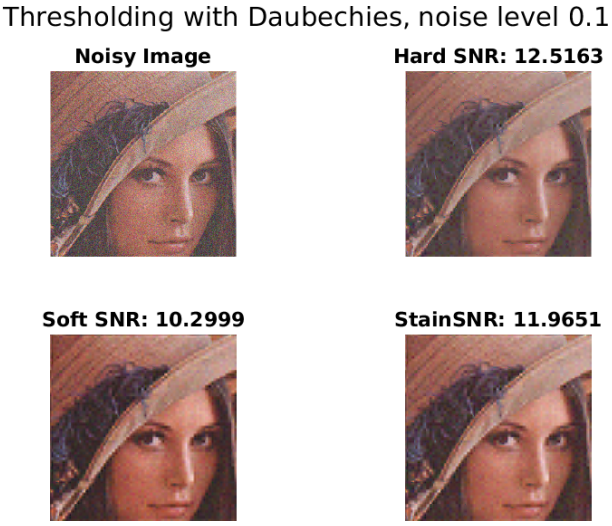

Figure 7: Threshold functions and denoising results

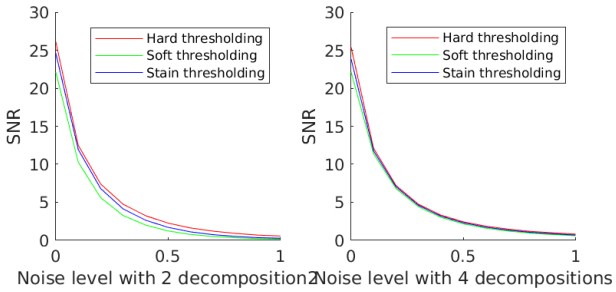

Figure 8: Effect of th

Figure 9: Give me a name


noisy lena
Adaptive SNR 6.7998
Fixed 6.8571

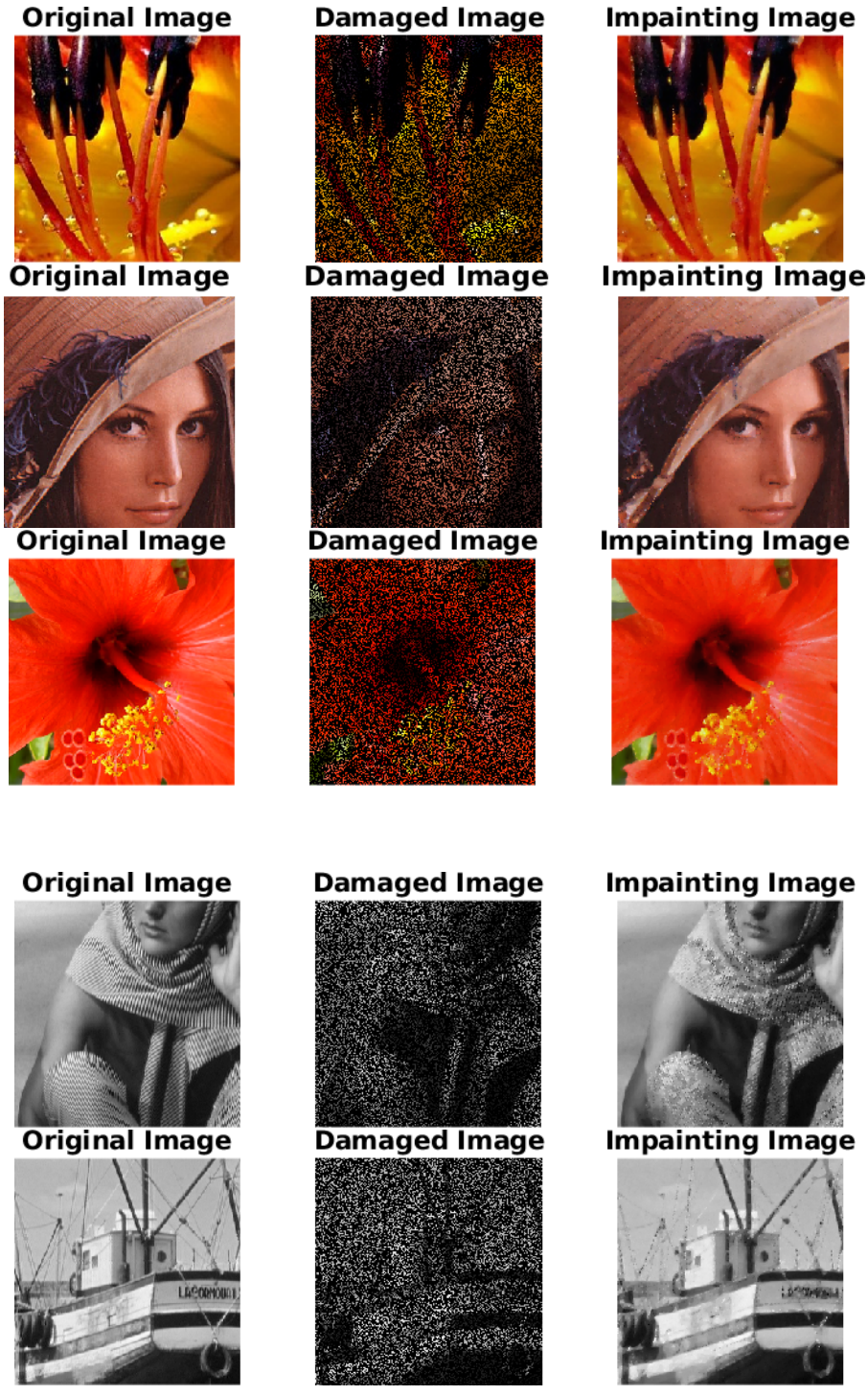

Figure 10: Give me a name

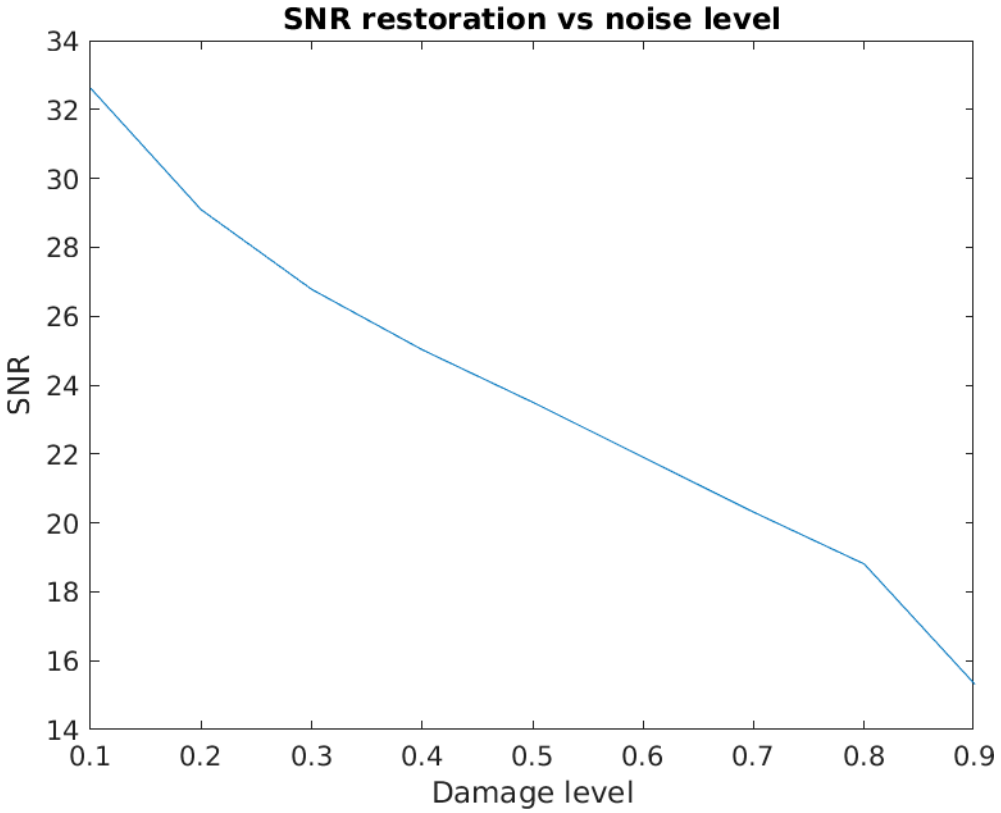

Figure 11: Give me a name

9 Matlab code

The file `getThreshold.m` calculates the theshold value for the threshold operations.

```
function [threshold] = getThreshold(channel, level)
    n = size(channel,1);

    % Mask the image part
    channel(1:n/(2^level), 1:n/(2^level)) = intmax;

    % Covert to vector
    channel=channel(:);

    % Remove the image part from vector
    channel(channel==intmax) = [];

    % Calculate the Threshold
    sigma=median(abs(channel(:)))/0.6745;
    threshold = 3 * sigma;
end
```

The file `Denoising.m` performs denoising operation on an image.

```
function [wave_image, dsnr] = Denoising(signal,level, Ho,
   theshold_type)

    % Do Wavelet composition
    wcR=fwt_or_2d(0, signal(:,:,1), level, Ho);
    wcG=fwt_or_2d(0, signal(:,:,2), level, Ho);
    wcB=fwt_or_2d(0, signal(:,:,3), level, Ho);

     % Apply Thresholding
    wcR_t = perform_thresholding(wcR, getThreshold(wcR,
       level),theshold_type);
    wcG_t = perform_thresholding(wcG, getThreshold(wcG,
       level),theshold_type);
    wcB_t = perform_thresholding(wcB, getThreshold(wcB,
       level),theshold_type);

    % Do Wavelet decomposition
    f_Rc=fwt_or_2d(1,wcR_t,level,Ho);
    f_Gc=fwt_or_2d(1,wcG_t,level,Ho);
    f_Bc=fwt_or_2d(1,wcB_t,level,Ho);
```

```
    wave_image = cat(3, f_Rc , f_Gc , f_Bc);

    dsnr = snr(signal, wave_image);
end
```

The file `ImpaintingPrimalDual.m` performs inpaining using regularization with primal-dual tone varation.

```
function [result,TVEnergy] = ImpaintingPrimalDual(y,
   NumFrames,sigma, tau, theta, Phi)

    % Functors
    K = @(f) grad(f);
    KS = @(u) - div(u);
    Amplitude = @(u) sqrt(sum(u.^2,3));
    F = @(u) sum(sum(Amplitude(u)));

    ProxF  = @(u,lambda) max(0, 1-lambda./repmat(Amplitude(
       u), [1 1 2])).*u;
    ProxFS = @(y,sigma) y - sigma*ProxF(y/sigma,1/sigma);
    ProxG  = @(f,tau) f + Phi(y -Phi(f));


    f=y;
    g=K(y)*0;
    f1=f;


    TVEnergy=zeros(NumFrames,1);
    TVEnergy(1) =  F(K(f1));
    for ii=2:NumFrames
        % Iteration
        fold = f;
        g = ProxFS( g+sigma*K(f1), sigma);
        f = ProxG(f-tau*KS(g), tau);
        f1 = f + theta * (f-fold);

        TVEnergy(ii) = F(K(f1));
        f= f1;
    end

    result = f1;
end
```