Wavelets for Graphs and their Deployment to Image Processing

Idan Ram

Electrical Engineering Department
Technion – Israel Institute of technology

Supervised by Prof. Israel Cohen and Prof. Michael Elad
This Talk is About …

Processing of Non-Conventionally Structured Signals

Many signal-processing tools (filters, alg., transforms, …) are designed for uniformly sampled signals.

Development of comparable methods capable of handling signals defined on graphs and point clouds is important and very much needed.

Our goal: Generalize the wavelet transform to handle this broad family of signals.

The true objective: Find how to bring sparse representation to processing of such signals.

We explore the use of our proposed methods for image processing.
Outline

- Tree-Based Wavelet Transforms
- Image Processing using Tree-Based Wavelets
- Image Processing using Smooth Patch Ordering
- Conclusions
Tree-Based Wavelet Transforms

This part is taken from the following papers:

Problem Formulation

- $\mathbf{X} = \{x_1, x_2, ..., x_N\}$ - the data set, such that $x_i \in \mathbb{R}^n$ may be
  - points in high dimension.
  - feature points associated with the nodes of a graph.

- $f: \mathbf{X} \rightarrow \mathbb{R}$ - a scalar function defined on the above coordinates, $f_i = f(x_i)$

- Our key assumption: under a distance measure $w(x_i, x_j)$

$$w(x_i, x_j) \text{ small } \rightarrow \left| f_i - f_j \right| \text{ small}$$

for almost every pair $(i, j) \Rightarrow f \text{ is regular}
Our Goal

Why Wavelet?

- Wavelet is a highly effective “sparsifying transform” for regular piece-wise smooth signals.
- We would like to imitate this for our data structure.
- However, the signal (vector) $\mathbf{f}$ is not necessarily smooth in general.
Sparsity: A different way to describe a signal's structure

\[ \mathbf{D} \alpha = f \]

Model assumption: \( \alpha \) is sparse, i.e. \( f \) is a linear combination of FEW (\( k \ll N \)) columns from \( \mathbf{D} \)
Wavelet for Graphs – Previous Works

“Diffusion Wavelets”

“Multiscale Methods for Data on Graphs and Irregular Multidimensional Situations”

“Wavelets on Graph via Spectral Graph Theory”

“Multiscale Wavelets on Trees, Graphs and High Dimensional Data: Theory and Applications to Semi Supervised Learning”
The Main Idea - Permutation

Permutation using $X = \{x_1, x_2, \ldots, x_N\}$
In fact, we propose to perform a different permutation in each resolution level of the multi-scale pyramid:

Naturally, these permutations will be applied reversely in the inverse transform.

The additional permutations make the difference between this and the plain 1D wavelet transform applied on $f$.

The transform adapts to the input signal while its linearity and unitarity are preserved.
We wish to design a permutation which produces a smooth signal when it is applied to the signal $f$. 
So, ... for example, we can simply permute by sorting the signal $f$. 

![Diagram showing the permutation of the signal $f$ to $f^p$]

**Building the Permutation $P_0$**
However: we are interested in the case where \( f \) is corrupted (noisy, missing values, ...) and thus such a sort operation is impossible.

To our help come the feature vectors in \( X \), which reflect on the order of the signal values \( f_k \). Recall:

Small \( w(x_i, x_j) \) implies small \( |f(x_i) - f(x_j)| \) for almost every pair \((i, j)\)

Thus, instead of solving for the optimal permutation that “simplifies” \( f \), we order the features in \( X \) to the shortest path that visits each point once.

\[
\min_P \sum_{i=2}^{N} |f^p(i) - f^p(i - 1)|
\]

Total variation of \( f \)

\[
\min_P \sum_{i=2}^{N} w(x_i^p, x_{i-1}^p)
\]

An instance of the Traveling-Salesman-Problem (TSP)
Building the Permutation $P_0$ (cont.)

We handle the TSP task by a greedy (and crude) approximation:

- Initialize with an arbitrary index $j$;
- Initialize the set of chosen indices to $\Omega(1) = \{j\}$;
- Repeat $k = 1:1:N-1$ times:
  - Find $x_i$ – the nearest neighbor to $x_{\Omega(k)}$ such that $i \not\in \Omega$;
  - Set $\Omega(k+1) = \{i\}$;
- Result: the set $\Omega$ holds the proposed ordering.
In order to construct $P_1$, $P_2$, ..., $P_{L-1}$, the permutations at the other pyramid’s levels, we use the same method, applied on propagated (reordered, filtered and sub-sampled) feature-vectors through the same wavelet pyramid:
The relation between the feature points in a full decomposition can be described using a tree-like structure.

Our proposed transform: **Generalized Tree-Based Wavelet Transform (GTBWT)**.
A redundant representation may be obtained using several random variants of the GTBWT. A more elegant solution is to modify the redundant wavelet transform.

Redundant Tree-Based Wavelet Transform (RTBWT)

à trous algorithm

[Shensa, 1992] [Beylkin, 1992]
This part is taken from the following papers:

Turning an Image into a Graph

- $Y$ – an image containing $N$ pixels.
- $y$ – column stacked version of $Y$.
- $y_i$ – $i$-th sample in $y$.

We choose:

- $x_i$ – column stacked version of an $\sqrt{n} \times \sqrt{n}$ patch around the location of $y_i$ in $Y$.
- $w(x_i, x_j)$ – the Euclidean distance $||x_i - x_j||$.

- Now, that the image is organized as a graph (or point-cloud), we can apply the developed transforms.
- After this “conversion”, we forget about spatial proximities.
- The overall scheme becomes “yet another” patch-based image processing algorithm.
M-Term Approximation

Multiply by $T$: Forward GTBWT

$S_{\lambda} \{Ty\}$

Multiply by $T^{-1}$: Inverse GTBWT

$M$ non-zeros

Show

$$\|y - \hat{y}\|^2 = \|y - T^{-1} S_{\lambda} \{Ty\}\|^2$$

as a function of $M$
For a 128×128 center portion of the image Lena

we compare the image representation efficiency of the

- GTBWT
- A common 1D wavelet transform
- 2D wavelet transform
The Representation Basis Functions

<table>
<thead>
<tr>
<th>Scaling functions</th>
<th>wavelets $l = 14$</th>
<th>wavelets $l = 13$</th>
<th>wavelets $l = 12$</th>
<th>wavelets $l = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original image</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- wavelets $l = 10$
- wavelets $l = 9$
- wavelets $l = 8$
- wavelets $l = 7$
- wavelets $l = 6$
- wavelets $l = 5$
Image Denoising – The Basic Scheme

\[ \hat{y} = T^{-1} S_\lambda \{ Tz \} \]

- **y**: Noisy image
- **v ~ N(0, \sigma^2 I)**: Gaussian noise
- **z**: Output image
- **T**: Forward GTBWT
- **T^{-1}**: Inverse GTBWT
**Image Denoising - Improvements**

**Cycle-spinning:** Apply the above scheme several (10) times, with different random variations of the GTBWT, and average.

Equivalent to using a redundant transform
Alternatively, we can simply apply our proposed Redundant Tree-Based Wavelet Transform.

Preliminary tests showed that the GTBWT and RTBWT perform similarly.
Sub-image averaging: Apply the above scheme to \( n \) sub-images and reconstruct the image from the obtained results.
Improved thresholding: thresholding the wavelet coefficients based on the norm of the (transformed) vector they belong to:

- Classical thresholding: every coefficient within $\mathbf{C}$ is passed through the function:

$$c_{i,j} = \begin{cases} c_{i,j} & \left| c_{i,j} \right| \geq T \\ 0 & \left| c_{i,j} \right| < T \end{cases}$$

- Instead we propose to force “joint-sparsity” on the above array of coefficients, forcing all rows to share the same support:

$$c_{i,j} = \begin{cases} c_{i,j} & \left\| c_{*,j} \right\|_2 \geq T \\ 0 & \left\| c_{*,j} \right\|_2 < T \end{cases}$$
Restricting the NN: When searching the nearest-neighbor for the ordering, restriction to near-by area is helpful, both computationally and in terms of the output quality.

Patch of size $\sqrt{n} \times \sqrt{n}$

Search-Area of size $B \times B$
Image Denoising – Results

- We explore the results obtained with two iterations of our scheme.
- The RTBWT PSNR results are good and competitive, especially for $\sigma \geq 25$.

<table>
<thead>
<tr>
<th>Image</th>
<th>$\sigma$/PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 / 28.14</td>
</tr>
<tr>
<td>Lena</td>
<td></td>
</tr>
<tr>
<td>BM3D</td>
<td>35.93</td>
</tr>
<tr>
<td>1 iteration</td>
<td>35.70</td>
</tr>
<tr>
<td>2 iterations</td>
<td>35.45</td>
</tr>
<tr>
<td>1 iteration + Wiener</td>
<td>35.75</td>
</tr>
<tr>
<td>Barbara</td>
<td></td>
</tr>
<tr>
<td>BM3D</td>
<td>34.98</td>
</tr>
<tr>
<td>1 iteration</td>
<td>34.50</td>
</tr>
<tr>
<td>2 iterations</td>
<td>34.55</td>
</tr>
<tr>
<td>1 iteration + Wiener</td>
<td>34.55</td>
</tr>
<tr>
<td>House</td>
<td></td>
</tr>
<tr>
<td>BM3D</td>
<td>36.71</td>
</tr>
<tr>
<td>1 iteration</td>
<td>36.41</td>
</tr>
<tr>
<td>2 iterations</td>
<td>35.68</td>
</tr>
<tr>
<td>1 iteration + Wiener</td>
<td>35.86</td>
</tr>
</tbody>
</table>
Image Deblurring

**Lexicographic ordering of the $N$ pixels**

- All these operations could be described as one **linear** operation: multiplication of $y$ by a huge matrix $\Omega$.
- This transform is **adaptive** to the specific image.

$X$: Array of overlapped patches of size $nN$

Applying a $J$ redundant tree-based wavelet transform

We obtain an array of $nNJ$ transform coefficients
Image Deblurring (cont.)

\[ \Psi = \mathbf{c} \]

\[ \mathbf{I} = \mathbf{D} \Psi \]

Every column in \( \mathbf{D} \) is an atom
We can use these operators to solve various inverse problems of the form:

$$z = Hy + v$$

where:
- $y$ is the original image
- $v$ is an AWGN, and
- $H$ is a degradation operator of any sort

We could consider solving a synthesis or analysis problems, or their combination:

$$\{\hat{y}, \hat{c}\} = \text{Argmin}_{y,c} \|z - Hy\|_2^2 + \eta \|y - Dc\|_2^2 +$$

$$+ \lambda \|c\|_p^p + \mu \|c - \Omega y\|_2^2$$

$\eta \to \infty \Rightarrow$ Synthesis

$\eta = 0 \Rightarrow$ Analysis

$\mu = 0 \Rightarrow$ Synthesis

$\mu \to \infty \Rightarrow$ Analysis
Generalized Nash Equilibrium*

Instead of minimizing the joint analysis/synthesis problem:

$$\{\hat{y}, \hat{c}\} = \text{Argmin}_{y, c} \|z - Hy\|_2^2 + \eta \|y - Dc\|_2^2 + \lambda \|c\|_p^p + \mu \|c - \Omega y\|_2^2$$

we handle it as a generalized Nash equilibrium process, and break it down into two separate and easy to handle parts:

**Inversion:** \(y_{k+1} = \text{Argmin}_y \|z - Hy\|_2^2 + \eta \|y - Dc\|_2^2\)

**Denoising:** \(c_{k+1} = \text{Argmin}_c \lambda \|c\|_p^p + \mu \|c - \Omega y\|_2^2\)

# Image Deblurring - Results

<table>
<thead>
<tr>
<th>Image</th>
<th>BM3D-DEB ISNR</th>
<th>IDD-BM3D ISNR init. with BM3D-DEB</th>
<th>Ours ISNR 1 iteration with simple initialization</th>
<th>Ours ISNR 3 iterations with simple initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>6.55</td>
<td>6.61</td>
<td>6.92</td>
<td>5.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Barbara</td>
<td>4.13</td>
<td>3.96</td>
<td>4.57</td>
<td>2.54</td>
</tr>
<tr>
<td>House</td>
<td>8.22</td>
<td>8.55</td>
<td>8.79</td>
<td>7.58</td>
</tr>
<tr>
<td>Cameraman</td>
<td>6.46</td>
<td>7.12</td>
<td>7.38</td>
<td>6.18</td>
</tr>
</tbody>
</table>

\[
\text{Blur PSF} = \frac{1}{1 + i^2 + j^2} \quad -7 \leq i, j \leq 7
\]

\[\sigma^2 = 8\]
Image Compression

- The problem: Compressing photo-ID images.
- **General purpose** methods (JPEG, JPEG2000) do not take into account the specific family.
- By **adapting** to the image-content (e.g. adaptive transform), better results could be obtained.
- We perform **Geometric alignment** of the images [Goldenberg, Kimmel, & E. ('05)] as it is very helpful.
- For our technique to operate well, we find the best **common Redundant Tree-Based Wavelet Transform** fitting a training set of facial images.
- Our pixel ordering is therefore designed on patches of size $1 \times 1 \times G$ pixels from the training volume.
Compression by Pixel-Ordering

Detect main features and warp the images (20 bytes)

Compute the mean image and subtract it

Find the common RTBWT and calculate Huffman Tables for entropy coding

Warp, remove the mean, and apply sparse coding using the RTBWT

Apply quantization and entropy coding
Face Compression - Results

Original images

400 bytes

600 bytes

800 bytes

PSNR

SSIM

<table>
<thead>
<tr>
<th>400 bytes</th>
<th>25.47 / 0.76</th>
</tr>
</thead>
<tbody>
<tr>
<td>600 bytes</td>
<td>28.73 / 0.82</td>
</tr>
<tr>
<td>800 bytes</td>
<td>30.09 / 0.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>400 bytes</th>
<th>29.94 / 0.86</th>
</tr>
</thead>
<tbody>
<tr>
<td>600 bytes</td>
<td>31.83 / 0.87</td>
</tr>
<tr>
<td>800 bytes</td>
<td>32.80 / 0.89</td>
</tr>
</tbody>
</table>
Rate-Distortion Curves

[Bryt & Elad, 2008]
Image Processing using Smooth Patch Ordering

This part is based on the paper:

Returning to the Basics

We extract all patches with overlaps
Then we order these patches to form the shortest path, as before

Suppose we start with a corrupted image

This reordering induces a permutation on the image pixels

What should we expect from this permutation?
Spatial Neighbor ≠ Euclidean Neighbor

What should we expect?

Spatial neighbors are not necessarily expected to remain neighbors in the new ordering.

Noisy images with $\sigma=10$
The Reordered Signal is More Regular

What should we expect?

- The ordering defined by the new path is expected to
  - be robust to noise and degradations.
  - lead to a smooth (or at least, piece-wise smooth) 1D version of the target signal.*

* Measure of smoothness:

\[
\frac{1}{L} \sum_{k=2}^{L} |y[k] - y[k - 1]|
\]

1. Raster scan: 9.57
2. Hilbert curve: 11.77
3. Sorted (ours): 5.63
Processing the Permutated Pixels

- Given a corrupted image of the form:
  \[ z = My + v \]

  where:  
  \( y \) is the original image  
  \( v \) is an AWGN, and  
  \( M \) is a point-wise degradation operator

- We reorder the pixels of the corrupted image using the permutation calculated from its patches.

- We can take advantage of our prior knowledge that the reordered target image should be smooth, and apply the following process:
  1. Re-order the pixels to a 1D signal
  2. Apply the 1D signal a smoothing operator
  3. Re-order the pixels back to their location
Image Denoising

Noisy with $\sigma=25$ (20.18dB)

Reconstruction: 32.65dB

Ordering based on the noisy patches

Simple smoothing

$\hat{y}_p$

$z^p$

* This result is obtained with (i) cycle-spinning, (ii) sub-image averaging, (iii) two iterations, (iv) learning the filter, and (v) switched smoothing.
The “Simple Smoothing” We Do

Simple smoothing works fine

We can do better by a training phase

but

Optimize $h$ to minimize the reconstruction MSE

Naturally, this is done off-line and on other images
Filtering – A Further Improvement

Cluster the patches to smooth and textured sets, and train a filter per each separately.

The results we show hereafter were obtained by:
(i) Cycle-spinning
(ii) Sub-image averaging
(iii) Two iterations
(iv) Learning the filter, and
(v) Switched smoothing.

Based on patch-STD
## Denoising Results Using Patch-Reordering

<table>
<thead>
<tr>
<th>Image</th>
<th></th>
<th>(\sigma/\text{PSNR [dB]})</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10 / 28.14</td>
<td>25 / 20.18</td>
<td>50 / 14.16</td>
</tr>
<tr>
<td>Lena</td>
<td>K-SVD</td>
<td>35.52</td>
<td>31.35</td>
<td>27.85</td>
</tr>
<tr>
<td></td>
<td>BM3D</td>
<td>35.93</td>
<td>32.05</td>
<td>28.96</td>
</tr>
<tr>
<td></td>
<td>1\textsuperscript{st} iteration</td>
<td>35.26</td>
<td>31.55</td>
<td>28.66</td>
</tr>
<tr>
<td></td>
<td>2\textsuperscript{nd} iteration</td>
<td>35.39</td>
<td>31.80</td>
<td>28.96</td>
</tr>
<tr>
<td>Barbara</td>
<td>K-SVD</td>
<td>34.40</td>
<td>29.54</td>
<td>25.43</td>
</tr>
<tr>
<td></td>
<td>BM3D</td>
<td>34.93</td>
<td>30.61</td>
<td>27.16</td>
</tr>
<tr>
<td></td>
<td>1\textsuperscript{st} iteration</td>
<td>34.29</td>
<td>30.36</td>
<td>27.19</td>
</tr>
<tr>
<td></td>
<td>2\textsuperscript{nd} iteration</td>
<td>34.39</td>
<td>30.47</td>
<td>27.35</td>
</tr>
<tr>
<td>House</td>
<td>K-SVD</td>
<td>35.90</td>
<td>31.97</td>
<td>28.01</td>
</tr>
<tr>
<td></td>
<td>BM3D</td>
<td>36.63</td>
<td>32.79</td>
<td>29.54</td>
</tr>
<tr>
<td></td>
<td>1\textsuperscript{st} iteration</td>
<td>35.61</td>
<td>32.34</td>
<td>29.28</td>
</tr>
<tr>
<td></td>
<td>2\textsuperscript{nd} iteration</td>
<td>35.80</td>
<td>32.54</td>
<td>29.64</td>
</tr>
</tbody>
</table>

Bottom line: (1) This idea works very well; (2) It is especially competitive for high noise levels; and (3) A second iteration always pays off.
Image Inpainting

0.8 of the pixels are missing

Reconstruction: 29.71dB

* This result is obtained with (i) cycle-spinning, (ii) sub-image averaging, and (iii) three iterations.
Inpainting Results – Examples

Given data 80% missing pixels

Bi-Cubic interpolation

DCT and OMP recovery

1st iteration of the proposed alg.

3rd iteration of the proposed alg.
# Inpainting Results

Reconstruction results from 80% missing pixels using various methods:

<table>
<thead>
<tr>
<th>Image</th>
<th>Method</th>
<th>PSNR [dB]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>Bi-Cubic</td>
<td>30.25</td>
</tr>
<tr>
<td></td>
<td>DCT + OMP</td>
<td>29.97</td>
</tr>
<tr>
<td></td>
<td>Proposed (1\textsuperscript{st} iter.)</td>
<td>30.25</td>
</tr>
<tr>
<td></td>
<td>Proposed (2\textsuperscript{nd} iter.)</td>
<td>31.80</td>
</tr>
<tr>
<td></td>
<td>Proposed (3\textsuperscript{rd} iter.)</td>
<td>31.96</td>
</tr>
<tr>
<td>Barbara</td>
<td>Bi-Cubic</td>
<td>22.88</td>
</tr>
<tr>
<td></td>
<td>DCT + OMP</td>
<td>27.15</td>
</tr>
<tr>
<td></td>
<td>Proposed (1\textsuperscript{st} iter.)</td>
<td>27.56</td>
</tr>
<tr>
<td></td>
<td>Proposed (2\textsuperscript{nd} iter.)</td>
<td>29.34</td>
</tr>
<tr>
<td></td>
<td>Proposed (3\textsuperscript{rd} iter.)</td>
<td>29.71</td>
</tr>
<tr>
<td>House</td>
<td>Bi-Cubic</td>
<td>29.21</td>
</tr>
<tr>
<td></td>
<td>DCT + OMP</td>
<td>29.69</td>
</tr>
<tr>
<td></td>
<td>Proposed (1\textsuperscript{st} iter.)</td>
<td>29.03</td>
</tr>
<tr>
<td></td>
<td>Proposed (2\textsuperscript{nd} iter.)</td>
<td>32.10</td>
</tr>
<tr>
<td></td>
<td>Proposed (3\textsuperscript{rd} iter.)</td>
<td>32.71</td>
</tr>
</tbody>
</table>

**Bottom line:**
1. This idea works very well;
2. It is operating much better than the classic sparse-rep. approach; and
3. Using more iterations always pays off, and substantially so.
We propose new wavelet transforms for scalar functions defined on graphs or high dimensional data clouds.

The proposed transform extends the classical orthonormal and redundant wavelet transforms.

We demonstrate the ability of these transforms to efficiently represent and denoise images.

Finally, we show that using the ordering of the patches only, quite effective processing of images can be obtained.

We also show that the obtained transform can be used as a learned dictionary for the compression of facial images.

We show that the obtained transform can be used as a regularizer in classical image processing Inverse-Problems.
Future Directions

- Finding new image processing applications.
- Finding new applications for the proposed method that involve processing data on graphs and point clouds.
- Improving the ordering scheme, for example by allowing patches/features to be revisited.
- Compressing facial images using more than one dictionary and more advanced entropy coding methods.
- Using pixel permutations as regularizes in image processing problems.
Thank you for your time,

Questions?