Wavelet for Graphs and its Deployment to Image Processing*

Michael Elad
The Computer Science Department
The Technion – Israel Institute of technology
Haifa 32000, Israel

The research leading to these results has been received funding from the European Union's Seventh Framework Programme (FP/2007-2013) ERC grant Agreement ERC-SPARSE-320649

*Joint work with
Idan Ram
Israel Cohen
The Electrical Engineering department
Technion – Israel Institute of Technology

SPARS 2013
Signal Processing with Adaptive Sparse Structured Representations.
July 8-11, 2013. EPFL, Lausanne.
Patch-Based Processing of Images

In the past decade we see more and more researchers suggesting to process a signal or an image with a paradigm of the form:

1. Break the given image into overlapping (small) patches
2. Operate on the patches separately or by exploiting inter-relation between them
3. Put back the resulting patches into a result canvas
Patch-Based Processing of Images

In the past decade we see more and more researchers suggesting to process a signal or an image with a paradigm of the form:

Surprisingly, these methods are very effective, actually leading to today’s state-of-the-art in many applications.

Common theme: The image patches are believed to exhibit a highly-structured geometrical form in the embedding space they reside in.
Who are the researchers promoting this line of work?
Many leading scientists from various places

Various Ideas:
- Non-local-means
- Kernel regression
- Sparse representations
- Locally-learned dictionaries
- BM3D
- Structured sparsity
- Structural clustering
- Subspace clustering
- Gaussian-mixture-models
- Non-local sparse rep.
- Self-similarity
- Manifold learning

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places

Who are the researchers promoting this line of work?
Many leading scientists from various places
This Talk is About ...

A different way to treat an image using its overlapped patches

Process the Patches

This ordering induces a very interesting permutation on the image pixels

Order to form the shortest possible path
Surprisingly, This Talk is Also About ...

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

Now we encounter different types of signals: e.g., point-clouds and graphs. Can we extend classical tools to these signals?

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

In the process, we will find ourselves returning to “regular” signals, handling them differently.

In fact, this is how this work started in the first place.
Part I – GTBWT

Generalized Tree-Based Wavelet Transform – The Basics

This part is taken from the following two papers:

Problem Formulation

- We are given a graph:
  - The $i$-th node is characterized by a $d$-dimensional feature vector $x_i$
  - The $i$-th node has a value $f_i$
  - The edge between the $i$-th and $j$-th nodes carries the distance $w(x_i, x_j)$ for an arbitrary distance measure $w(\cdot, \cdot)$.

- Assumption: a “short edge” implies close-by values, i.e.

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

for almost every pair $(i, j)$. 
Different Ways to Look at This Data

- We start with a set of \( d \)-dimensional vectors \( \mathbf{X} = \{ \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \} \in \mathbb{R}^d \)
  These could be:
  - Feature points for a graph’s nodes,
  - Set of coordinates for a point-cloud.

- A scalar function is defined on these coordinates, \( f : \mathbf{X} \rightarrow \mathbb{R} \),
  giving \( \mathbf{f} = [f_1, f_2, \ldots, f_N] \).

- We may regard this dataset as a set of \( m \) samples taken from a high dimensional function \( f : \mathbb{R}^d \rightarrow \mathbb{R} \).

- The assumption that small \( w(\mathbf{x}_i, \mathbf{x}_j) \) implies small \( |f_i - f_j| \) for almost every pair \((i, j)\) implies that the function behind the scene, \( f \), is “regular”.
Our Goal

Wavelet Transform

Sparse (compact) Representation

Why Wavelet?

- Wavelet for regular piece-wise smooth signals is a highly effective "sparsifying transform". However, the signal (vector) $\mathbf{f}$ is not necessarily smooth in general.

- We would like to imitate this for our data structure.
Wavelet for Graphs – A Wonderful Idea

I wish we would have thought of it first ...

“Diffusion Wavelets”

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

“Wavelets on Graph via Spectral Graph Theory”

“Multiscale Wavelets on Trees, Graphs and High ... Supervised Learning”

“Wavelet Shrinkage on Paths for Denoising of Scattered Data”

...
The Main Idea (1) - Permutation

Permutation using \( X = \{x_1, x_2, \ldots, x_N\} \)
The Main Idea (2) - Permutation

- 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.
- Thus, the difference between this and the plain 1D wavelet transform applied on $f$ are the additional permutations, thus preserving the transform’s linearity and unitarity, while also adapting to the input signal.
Building the Permutations (1)

- Let's start with $P_0$ – the permutation applied on the incoming signal.
- Recall: the wavelet transform is most effective for piecewise regular signals. Thus, $P_0$ should be chosen such that $P_0 f$ is most “regular”.
- So, ... for example, we can simply permute by sorting the signal $f$ ...

![Graphs showing permutation effects](image)
Building the Permutations (2)

- **However**: we will be dealing with corrupted signals \( f \) (noisy, missing values, ...) and thus such a sort operation is impossible.

- To our help comes 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 in each point once, in what will be an instance of the Traveling-Salesman-Problem (TSP):

  \[
  \min_p \sum_{i=2}^{N} |f^p(i) - f^p(i - 1)| \quad \text{versus} \quad \min_p \sum_{i=2}^{N} w(x^p_i, x^p_{i-1})
  \]
Building the Permutations (3)

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 \( i \notin \Omega \);
  - Set \( \Omega(k+1) = \{ i \} \);
- Result: the set \( \Omega \) holds the proposed ordering.
So far we concentrated on $P_0$ at the finest level of the multi-scale pyramid. In order to construct $P_1, P_2, \ldots, P_{L-1}$, 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:

$$X^0 = X \rightarrow P_0 \rightarrow \text{LP-Filtering (h) \& Sub-sampling} \rightarrow X^1 \rightarrow P_1 \rightarrow \text{LP-Filtering (h) \& Sub-sampling} \rightarrow X^2 \rightarrow P_2 \rightarrow \text{LP-Filtering (h) \& Sub-sampling} \rightarrow X^3 \rightarrow P_3 \rightarrow \text{LP-Filtering (h) \& Sub-sampling} \rightarrow \ldots$$
Why “Generalized Tree ...”?

- Our proposed transform: Generalized Tree-Based Wavelet Transform (GTBWT).
- We also developed a Redundant version of this transform based on the stationary wavelet transform [Shensa, 1992] [Beylkin, 1992] – also related to the “A-Trous Wavelet” (will not be presented here).
Treating Graph/Cloud-of-points

- Just to complete the picture, we should demonstrate the (R)GTBWT capabilities on graphs/cloud of points.
- We took several classical machine learning train + test data for several regression problems, and tested the proposed transform in
  - Cleaning (denoising) the data from additive noise;
  - Filling in missing values (semi-supervised learning); and
  - Detecting anomalies (outliers) in the data.
- The results are encouraging. We shall present herein one such experiment briefly.
Treating Graphs: The Data

Data Set: Relative Location of CT axial axis slices

More details: Overall 53500 such pairs of feature and value, extracted from 74 different patients (43 male and 31 female).
Treating Graphs: **Denoising**

Original labels + AWGN → Noisy labels

**Denoising by NLM-like algorithm**
Find for each point its K-NN in feature-space, and compute a weighted average of their labels.

**Denoising by THR with RTBWT**
Apply the RTBWT transform to the point-cloud labels, threshold the values and transform back.
Treating Graphs: Denoising

- Noisy signal
- RTBWT
- NL-means

Graph showing SNR [dB] vs. noise standard deviation.
Treating Graphs: **Semi-Supervised Learning**

- Add AWGN noise to original labels.
- Discard p% of the labels randomly.
- Find for each missing point its K-NN in feature-space that have a label, and compute a weighted average of their labels.

**Filling-in by NLM-like algorithm**

- Denoising by NLM
- Denoising by RTBWT

**Option:** Iterate

**Projection**
Treating Graphs: Semi-Supervised Learning

![Graph Plot](image)

- **SNR [dB]**
- **# missing samples**
- **$\sigma=20$**

Legend:
- Corrupted
- NL-means
- NL-means (iter 2)
- RTBWT (iter 2)
- NL-means (iter 3)
- RTBWT (iter 3)
Treating Graphs: Semi-Supervised Learning

![Graph](image_url)

- **SNR [dB]**
  - **σ=5**
  - Corrupted
  - NL-means
  - NL-means (iter 2)
  - RTBWT (iter 2)
  - NL-means (iter 3)
  - RTBWT (iter 3)

**# missing samples**

- 0.1
- 0.2
- 0.3
- 0.4
- 0.5
- 0.6
- 0.7
- 0.8
- 0.9

Wavelets for Graphs and Its Deployment to Image Processing
By: Michael Elad
Part II – Handling Images

Using GTBWT by Handling Image Patches

This part is taken from the same papers mentioned before ...

Could an Image Become a Graph?

- Now, that the image is organized as a graph (or point-cloud), we can apply the developed transform.
- The distance measure $w(\bullet, \bullet)$ we will be using is Euclidean.
- It seems that after this “conversion”, we forget all about spatial proximities.

$$f(x_j) = f_j$$

$$f(x_i) = f_i$$
Our Transform

\[ f \]  

Lexicographic ordering of the \( N \) pixels

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

\[ \mathbf{X} \]: Array of overlapped patches of size \( dN \)

Applying a \( J \) redundant wavelet of some sort including permutations

\[ \Omega \]

We obtain an array of \( dNJ \) transform coefficients
Let's Test It: M-Term Approximation

Original Image

Multiply by \( \Omega \): Forward GTBWT

Multiply by \( D \): Inverse GTBWT

\[ \| f - \hat{f} \|^2 = \| f - D S_\lambda \{ \Omega f \} \|^2 \]

as a function of \( M \)

Output image
Let's Test It: M-Term Approximation

For a 128×128 center portion of the image Lenna, we compare the image representation efficiency of the

- GTBWT
- A common 1D wavelet transform
- 2D wavelet transform

![Graph showing PSNR vs. number of coefficients for different transforms. The graph includes lines for GTBWT – permutation at varying level, common 1D, and db4 2D transforms.]
Comparison Between Different Wavelets

- **db1 (Haar)**
- **db4**
- **db8**
- **GTBWT comparison**
The Representation’s Atoms – Synthetic Image

Scaling functions

Original image

wavelets $l = 7$

wavelets $l = 6$

wavelets $l = 5$

wavelets $l = 4$

wavelets $l = 3$

wavelets $l = 2$

wavelets $l = 1$
The Representation’s Atoms – Lenna

Scaling functions

Original image

wavelets $l = 10$

wavelets $l = 9$

wavelets $l = 8$

wavelets $l = 7$

wavelets $l = 6$

wavelets $l = 5$

wavelets $l = 4$

wavelets $l = 3$

wavelets $l = 2$

wavelets $l = 1$
Let's Test It: Image Denoising

\[ f \xrightarrow{\text{Denoising Algorithm}} \tilde{f} \xrightarrow{\text{Approximation by the THR algorithm:}} \hat{f} \]

\[ v \sim N(0, \sigma^2 I) \]

Noisy image \rightarrow \Omega: Forward GTBWT \rightarrow \mathcal{S}_\lambda{} \rightarrow D: \text{Inverse GTBWT} \rightarrow Output image

\[ \hat{f} = DS_\lambda\{\Omega \tilde{f}\} \]
**Image Denoising – Improvements**

**Cycle-spinning:** Apply the above scheme several (10) times, with a different GTBWT (different random ordering), and average.
Sub-image averaging: A by-product of GTBWT is the propagation of the whole patches. Thus, we get $n$ transform vectors, each for a shifted version of the image and those can be averaged.

\[ X^0 = X \]

\[ P_0 \] LP-Filtering ($h$) & Sub-sampling

\[ X^1 \]

\[ P_1 \] LP-Filtering ($h$) & Sub-sampling

\[ X^2 \]

\[ P_2 \] LP-Filtering ($h$) & Sub-sampling

\[ X^3 \]

\[ P_3 \] LP-Filtering ($h$) & Sub-sampling

HP-Filtering ($g$) & Sub-sampling

HP-Filtering ($g$) & Sub-sampling

HP-Filtering ($g$) & Sub-sampling

HP-Filtering ($g$) & Sub-sampling
Image Denoising – Improvements

Sub-image averaging: A by-product of GTBWT is the propagation of the whole patches. Thus, we get $n$ transform vectors, each for a shifted version of the image and those can be averaged.

- Combine these transformed pieces;
- The center row is the transformed coefficients of $f$.
- The other rows are also transform coefficients – of $d$ shifted versions of the image.
- We can reconstruct $d$ versions of the image and average.
Restricting the NN: It appears that when searching the nearest-neighbor for the ordering, restriction to near-by area is helpful, both computationally (obviously) and in terms of the output quality.

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

Search-Area of size $\sqrt{B} \times \sqrt{B}$
Improved thresholding: Instead of thresholding the wavelet coefficients based on their value, threshold them based on the norm of the (transformed) vector they belong to:

- Recall the transformed vectors as described earlier.
- Classical thresholding: every coefficient within $\mathbf{C}$ is passed through the function:
  
  $$
  c_{i,j} = \begin{cases} 
  c_{i,j} & |c_{i,j}| \geq T \\
  0 & |c_{i,j}| < T 
  \end{cases}
  $$

- The proposed alternative would be to force “joint-sparsity” on the above array of coefficients, forcing all rows to share the same support:
  
  $$
  c_{i,j} = \begin{cases} 
  c_{i,j} & \|c_{*,j}\|_2 \geq T \\
  0 & \|c_{*,j}\|_2 < T 
  \end{cases}
  $$
Image Denoising – Results

- We apply the proposed scheme with the Symmlet 8 wavelet to noisy versions of the images Lena and Barbara.

- For comparison reasons, we also apply to the two images the K-SVD and BM3D algorithms.

<table>
<thead>
<tr>
<th>σ/PSNR</th>
<th>Image</th>
<th>K-SVD</th>
<th>BM3D</th>
<th>GTBWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/28.14</td>
<td>Lena</td>
<td>35.51</td>
<td>35.93</td>
<td>35.87</td>
</tr>
<tr>
<td></td>
<td>Barbara</td>
<td>34.44</td>
<td>34.98</td>
<td>34.94</td>
</tr>
<tr>
<td>25/20.18</td>
<td>Lena</td>
<td>31.36</td>
<td>32.08</td>
<td>32.16</td>
</tr>
<tr>
<td></td>
<td>Barbara</td>
<td>29.57</td>
<td>30.72</td>
<td>30.75</td>
</tr>
</tbody>
</table>

- The PSNR results are quite good and competitive.

- What about run time?
Relation to BM3D?

BM3D:
- 3D Transform & threshold

Our scheme:
- Reorder, GTBWT, and threshold

Wavelets for Graphs and Its Deployment to Image Processing
By: Michael Elad
Relation to BM3D?

In a nut-shell, while BM3D searches for patch neighbors and process them locally, our approach seeks one path through all the patches (each gets its own neighbors as a consequence), and the eventual processing is done globally.
What Next?

We have a highly effective sparsifying transform for images. It is “linear” and image adaptive.

A: Refer to this transform as an abstract sparsification operator and use it in general image processing tasks.

B: Streep this idea to its bones: keep the patch-reordering, and propose a new way to process images.
Part III – Frame

Interpreting the GTBWT as a Frame and using it as a Regularizer

This part is documented in the following draft:


We rely heavily on

Recall Our Core Scheme

Or, put differently, \( \hat{x} = D \cdot T\{\Omega y}\): We refer to GTBWT as a redundant frame, and use a “heuristic” shrinkage method with it, which aims to approximate the solution of

**Synthesis:**
\[
\hat{x} = D \cdot \text{Argmin}_{\alpha} \|D\alpha - y\|_2^2 + \lambda \|\alpha\|_p^p
\]

or

**Analysis:**
\[
\hat{x} = \text{Argmin}_{f} \|x - y\|_2^2 + \lambda \|\Omega x\|_p^p
\]
Recall: Our Transform (Frame)

- **X**: Array of overlapped patches of size $dN$

  ![X](image)

- **f**: Lexicographic ordering of the $N$ pixels

  ![f](image)

- Applying a $J$ redundant wavelet of some sort including permutations

  ![Wavelet](image)

- We obtain an array of $dNJ$ transform coefficients

  ![Coefficients](image)

- All these operations could be described as one **linear** operation: multiplication of $f$ by a huge matrix $\Omega$

- This transform is **adaptive** to the specific image
Our Notations

\[ \alpha \quad x \quad \Omega \]

\[ D = x \Omega \]

\[ I = D \Omega \]

(Not a Moore-Penrose pair)
What Can We Do With This Frame?

We could solve various inverse problems of the form:

\[ y = Ax + v \]

where:
- \( x \) is the original image
- \( v \) is an AWGN, and
- \( A \) is a degradation operator of any sort

We could consider the synthesis, the analysis, or their combination:

\[
\{ \hat{x}, \hat{\alpha} \} = \text{Argmin}_{\alpha, x} \left( \|y - Ax\|_2^2 + \frac{1}{\beta} \|D\alpha - x\|_2^2 + \lambda \|\alpha\|_p^p + \frac{1}{\mu} \|\Omega x - \alpha\|_2^2 \right)
\]

\[ \beta = 0 \quad \mu = \infty \quad \rightarrow \text{Synthesis} \]
\[ \beta = \infty \quad \mu = 0 \quad \rightarrow \text{Analysis} \]
Generalized Nash Equilibrium*

Instead of minimizing the joint analysis/synthesis problem:

\[
\{\hat{x}, \hat{\alpha}\} = \arg\min_{\alpha, x} \|y - Ax\|_2^2 + \frac{1}{\beta} \|D\alpha - x\|_2^2 + \lambda \|\alpha\|_p^p + \frac{1}{\mu} \|\Omega x - \alpha\|_2^2
\]

break it down into two separate and easy to handle parts:

\[
x_{k+1} = \arg\min_x \|y - Ax\|_2^2 + \frac{1}{\beta} \|D\alpha_k - x\|_2^2
\]

\[
\alpha_{k+1} = \arg\min_{\alpha} \lambda \|\alpha\|_p^p + \frac{1}{\mu} \|\Omega x_{k+1} - \alpha\|_2^2
\]

Deblurring Results

Original                     Blurred                 Restored
Deblurring Results

<table>
<thead>
<tr>
<th>Image</th>
<th>Input PSNR</th>
<th>BM3D-DEB ISNR</th>
<th>IDD-BM3D ISNR init. with BM3D-DEB</th>
<th>Ours ISNR Init. with BM3D-DEB</th>
<th>Ours ISNR 3 iterations with simple initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>27.25</td>
<td>7.95</td>
<td>7.97</td>
<td>8.08</td>
<td>8.20</td>
</tr>
<tr>
<td>Barbara</td>
<td>23.34</td>
<td>7.80</td>
<td>7.64</td>
<td>8.25</td>
<td>6.21</td>
</tr>
<tr>
<td>House</td>
<td>25.61</td>
<td>9.32</td>
<td>9.95</td>
<td>9.80</td>
<td>10.06</td>
</tr>
<tr>
<td>Cameraman</td>
<td>22.23</td>
<td>8.19</td>
<td>8.85</td>
<td>9.19</td>
<td>8.52</td>
</tr>
</tbody>
</table>

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

\[ \sigma^2 = 2 \]
Part IV – Patch (Re)-Ordering

Let's Simplify Things, Shall We?

This part is based on the paper:

Returning to the Basics

Suppose we start with a clean image.

We extract all (with overlaps) patches of size $B \times B$ (e.g. $B=20$).

Then we order these patches to form the shortest path, as before.

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.
The Reordered Signal is More Regular

What should we expect?

- The new path is expected to lead to very smooth* (or at least, piece-wise smooth) 1D signal.
- The ordering is expected to be robust to noise and degradations → the underlying signal should still be smooth.

* Measure of smoothness:

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

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

Assumptions:
- After a shortest-path reordering of the patches form a clean image, we expect a highly regular signal.
- Reordering a corrupted image is likely to lead to a good quality sort as well, due to the robustness brought by the patch-matching.

An Idea: Given a corrupted image of the form:

\[ y = Mx + v \]

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

Apply this process:

1. Re-order the pixels to a 1D signal
2. Process the 1D signal with a simple filter
3. Re-order the pixels back to their location
Use the Reordering for Denoising

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

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

Extract all (with overlaps) patches of size $10 \times 10$

Order these patches as before (TSP)

Reconstruction: 32.65dB *

Smooth/filter the values along this row in a simple way

Take the center-row – it represents a permutation of the image pixels to a regular function
Intuition: Why Should This Work?

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

Reconstruction: 32.65dB

Ordering based on the noisy pixels

Simple smoothing

True samples

Noisy samples
The “Simple Smoothing” We Do

Simple smoothing works fine

but

We can do better by a training phase

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>(\sigma/\text{PSNR [dB]})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 / 28.14</td>
</tr>
<tr>
<td></td>
<td>25 / 20.18</td>
</tr>
<tr>
<td></td>
<td>50 / 14.16</td>
</tr>
<tr>
<td>Lena</td>
<td>K-SVD</td>
</tr>
<tr>
<td></td>
<td>1\text{st iteration}</td>
</tr>
<tr>
<td></td>
<td>2\text{nd iteration}</td>
</tr>
<tr>
<td></td>
<td>35.49</td>
</tr>
<tr>
<td></td>
<td>35.33</td>
</tr>
<tr>
<td></td>
<td>35.41</td>
</tr>
<tr>
<td></td>
<td>31.36</td>
</tr>
<tr>
<td></td>
<td>31.58</td>
</tr>
<tr>
<td></td>
<td>31.81</td>
</tr>
<tr>
<td></td>
<td>27.82</td>
</tr>
<tr>
<td></td>
<td>28.54</td>
</tr>
<tr>
<td></td>
<td>29.00</td>
</tr>
<tr>
<td>Barbara</td>
<td>K-SVD</td>
</tr>
<tr>
<td></td>
<td>1\text{st iteration}</td>
</tr>
<tr>
<td></td>
<td>2\text{nd iteration}</td>
</tr>
<tr>
<td></td>
<td>34.41</td>
</tr>
<tr>
<td></td>
<td>34.48</td>
</tr>
<tr>
<td></td>
<td>34.46</td>
</tr>
<tr>
<td></td>
<td>29.53</td>
</tr>
<tr>
<td></td>
<td>30.46</td>
</tr>
<tr>
<td></td>
<td>30.54</td>
</tr>
<tr>
<td></td>
<td>25.40</td>
</tr>
<tr>
<td></td>
<td>27.17</td>
</tr>
<tr>
<td></td>
<td>27.45</td>
</tr>
<tr>
<td>House</td>
<td>K-SVD</td>
</tr>
<tr>
<td></td>
<td>1\text{st iteration}</td>
</tr>
<tr>
<td></td>
<td>2\text{nd iteration}</td>
</tr>
<tr>
<td></td>
<td>36.00</td>
</tr>
<tr>
<td></td>
<td>35.58</td>
</tr>
<tr>
<td></td>
<td>35.94</td>
</tr>
<tr>
<td></td>
<td>32.12</td>
</tr>
<tr>
<td></td>
<td>32.48</td>
</tr>
<tr>
<td></td>
<td>32.65</td>
</tr>
<tr>
<td></td>
<td>28.15</td>
</tr>
<tr>
<td></td>
<td>29.37</td>
</tr>
<tr>
<td></td>
<td>29.93</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 almost always pays off.
What About Inpainting?

0.8 of the pixels are missing

Extract all (with overlaps) patches of size 9×9

Order these patches as before distance uses EXISTING pixels only

Fill the missing values in a simple (cubic interpolation) way

Take the center-row – it represents a permutation of the image pixels to a regular function

Reconstruction: 29.71dB *

* This result is obtained with (i) cycle-spinning, (ii) sub-image averaging, and (iii) two iterations.
The Rationale

0.8 of the pixels are missing

Reconstruction: 27.15dB

Simple interpolation

Missing sample
Existing sample

Ordering

$\mathbf{y}$

$\mathbf{y_p}$
Inpainting Results – Examples

- **Given data 80% missing pixels**
  - Bi-Cubic interpolation: PSNR= 29.21 dB
  - DCT and OMP recovery: PSNR= 29.69 dB
  - 1st iteration of the proposed alg.: PSNR= 29.03 dB
  - 3rd iteration of the proposed alg.: PSNR= 32.71 dB
## 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 (1st iter.)</td>
<td>30.25</td>
</tr>
<tr>
<td></td>
<td>Proposed (2nd iter.)</td>
<td>31.80</td>
</tr>
<tr>
<td></td>
<td>Proposed (3rd 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 (1st iter.)</td>
<td>27.56</td>
</tr>
<tr>
<td></td>
<td>Proposed (2nd iter.)</td>
<td>29.34</td>
</tr>
<tr>
<td></td>
<td>Proposed (3rd 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 (1st iter.)</td>
<td>29.03</td>
</tr>
<tr>
<td></td>
<td>Proposed (2nd iter.)</td>
<td>32.10</td>
</tr>
<tr>
<td></td>
<td>Proposed (3rd 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.
Part IV – Time to Finish
Conclusions and a Bit More
Conclusions

We propose a new wavelet transform 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 denoising and inpainting can be obtained.

We also show that the obtained transform can be used as a regularizer in classical image processing Inverse-Problems.
What Next?

- Demonstrating the proposed wavelet on more data clouds/graphs?
- Replace the TSP ordering by MDS?
- Why TSP? Who says we cannot revisit patches?
- Replace "sub-image averaging" with a sparsifying transform?
- Exploiting the known distances?
- Improving the TSP approximation solver?
- Sparse Representations and learned dictionaries in the ordered domain?
- Lifting scheme for treating clouds?
- Pixel permutation as regularizer?
Thank you for your time

thanks to the organizers of this event

Volkan Cevher and Matthias Seeger

Questions?

Post-Docs Are Needed