Generalized Tree-Based Wavelet Transform*
and Applications to Patch-Based Image Processing

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

A Seminar in the Hebrew University of Jerusalem

*Joint work with

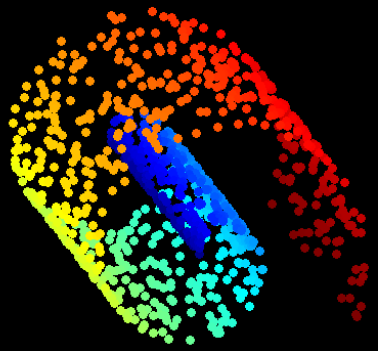
Idan Ram

Israel Cohen

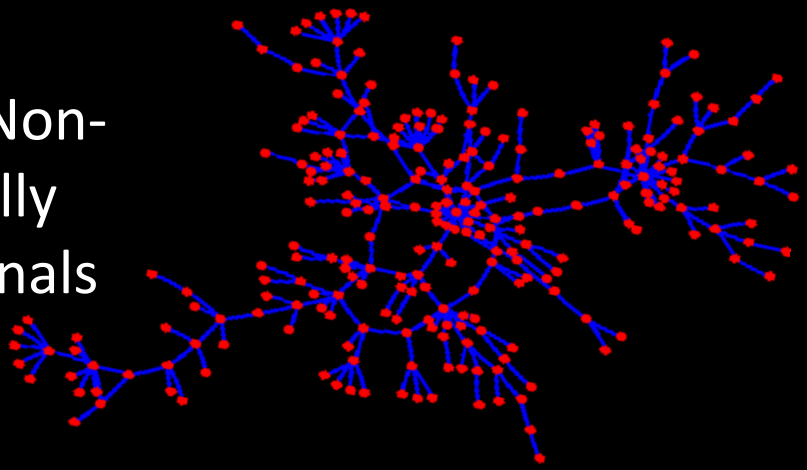
The Electrical Engineering department the Technion – Israel Institute of Technology

Technion
Israel Institute of Technology

This Talk is About …



Processing of Non-Conventionally Structured Signals



Many signal-processing tools (filters, 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

Previous and Related Work

- "Diffusion Wavelets"
  R. R. Coifman, and M. Maggioni, 2006.
- "Multiscale Methods for Data on Graphs and Irregular Multidimensional Situations"
  M. Jansen, G. P. Nason, and B. W. Silverman, 2008.
- "Wavelets on Graph via Spectal Graph Theory"
  D. K. Hammond, and P. Vandergheynst, and R. Gribonval, 2010.
- "Multiscale Wavelets on Trees, Graphs and High Dimensional Data: Theory and Applications to Semi Supervised Learning"
  M . Gavish, and B. Nadler, and R. R. Coifman, 2010.
- "Wavelet Shrinkage on Paths for Denoising of Scattered Data"
  D. Heinen and G. Plonka, to appear

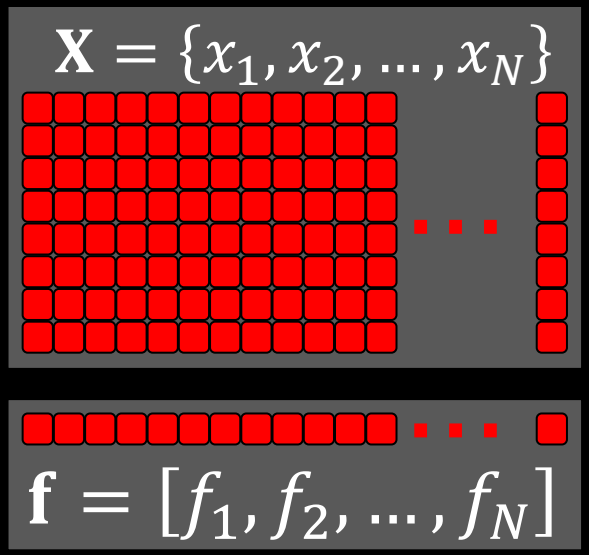
Part I – GTBWT

Generalized Tree-Based Wavelet Transform – The Basics

- I. Ram, M. Elad, and I. Cohen, "Generalized Tree-Based Wavelet Transform", IEEE Trans. Signal Processing, vol. 59, no. 9, pp. 4199–4209, 2011.
- I. Ram, M. Elad, and I. Cohen, "Redundant Wavelets on Graphs and High Dimensional Data Clouds", IEEE Signal Processing Letters, Vol. 19, No. 5, pp. 291–294 , May 2012.

Problem Formulation

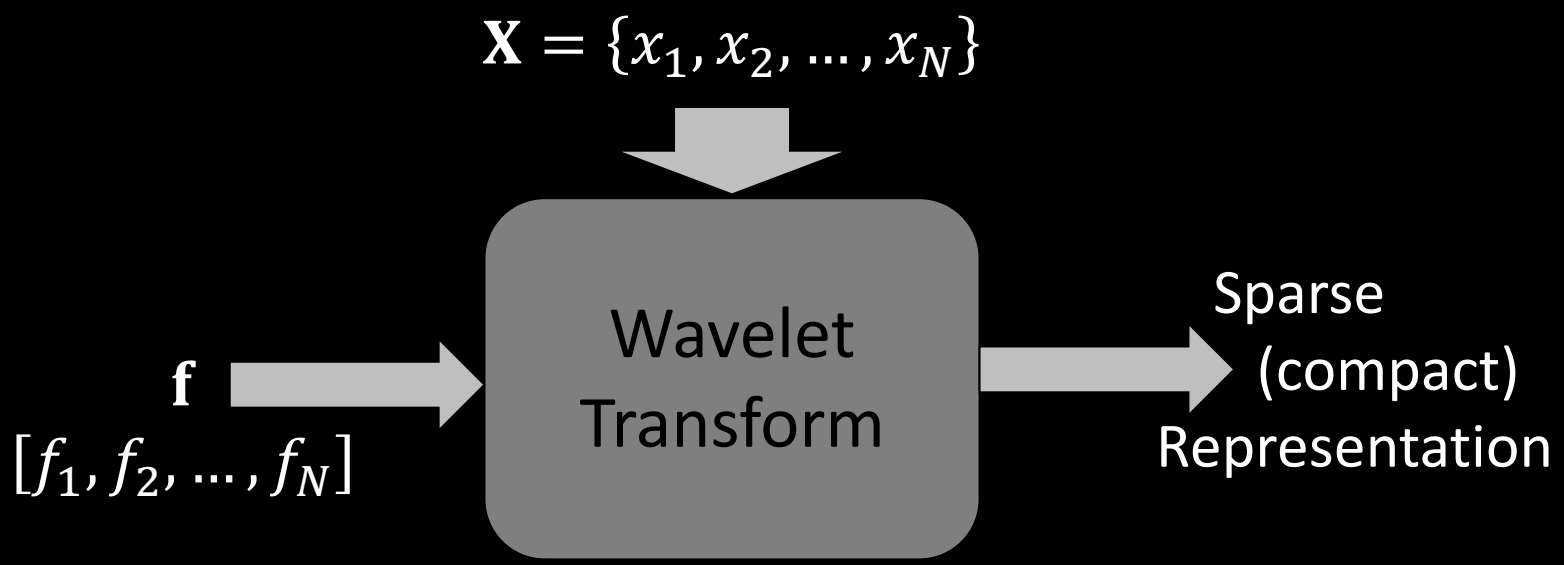
- We start with a set of points $\mathbf{X} = \{x_1, x_2, \dots, x_N\} \in \mathbb{R}^d$. These could be
  - Feature points associated with the nodes of a graph.
  - Set of coordinates for a point-cloud in high-dimensional space.
- A scalar function is defined on these coordinates, $f: \mathbf{X} \to \mathbb{R}$, our signal to process is $\mathbf{f} = [f_1, f_2, \dots, f_N]$.
- We may regard this dataset as a set of samples taken from a high-dimensional function $f(x): \mathbb{R}^d \to \mathbb{R}$.



- Key assumption – A distance-measure $w(x_i, x_j)$ between points in $\mathbb{R}^d$ is available to us. The function behind the scene is "regular":

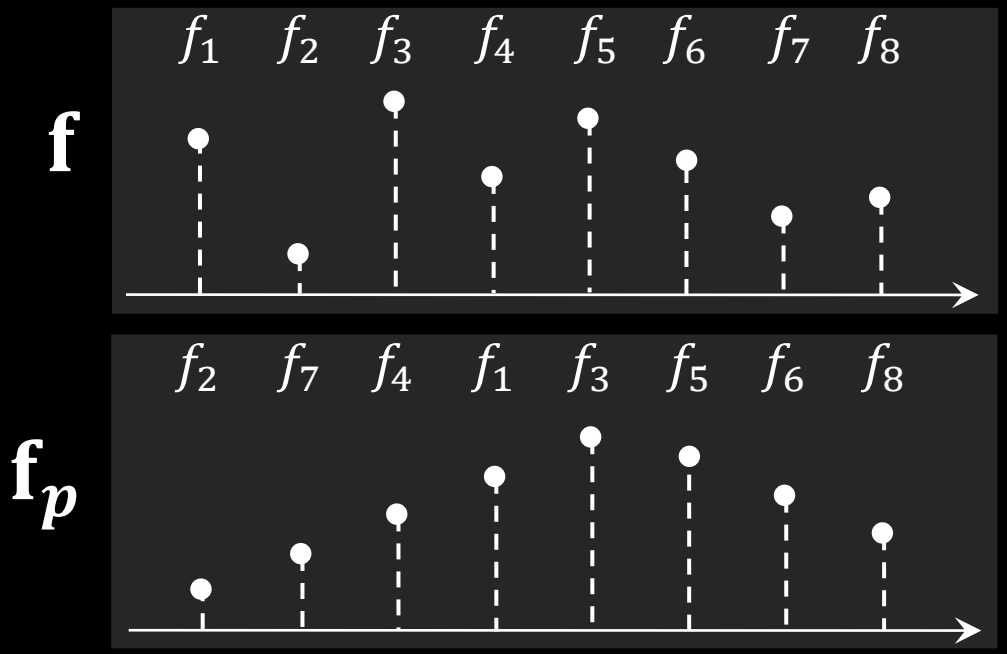
Small $w(x_i, x_j)$ implies small $|f(x_i) - f(x_j)|$ for almost every pair (i, j)

Our Goal

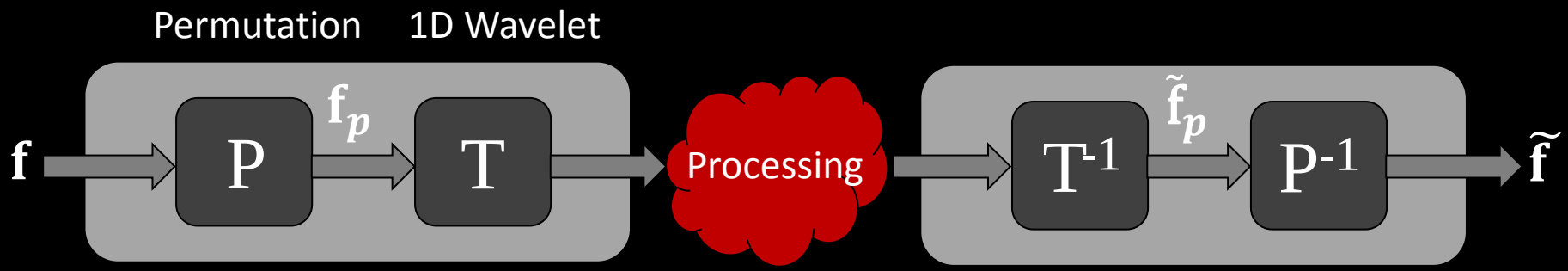


- We develop both an orthogonal wavelet transform and a redundant alternative, both efficiently representing the input signal $\mathbf{f}$.
- Our problem: The regular wavelet transform produces a small number of large coefficients when it is applied to piecewise regular signals. But, the signal (vector) $\mathbf{f}$ is not necessarily smooth in general.

The Main Idea (1) - Permutation

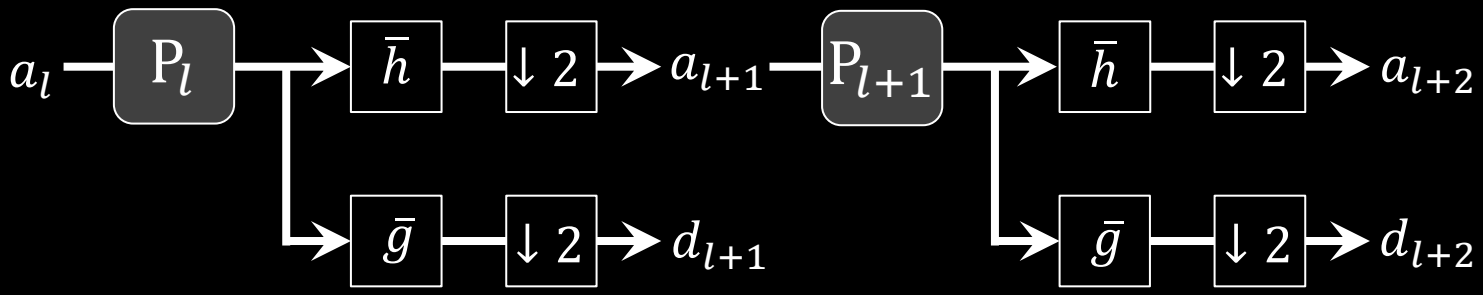


Permutation using
$\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$



The Main Idea (2) - Permutation

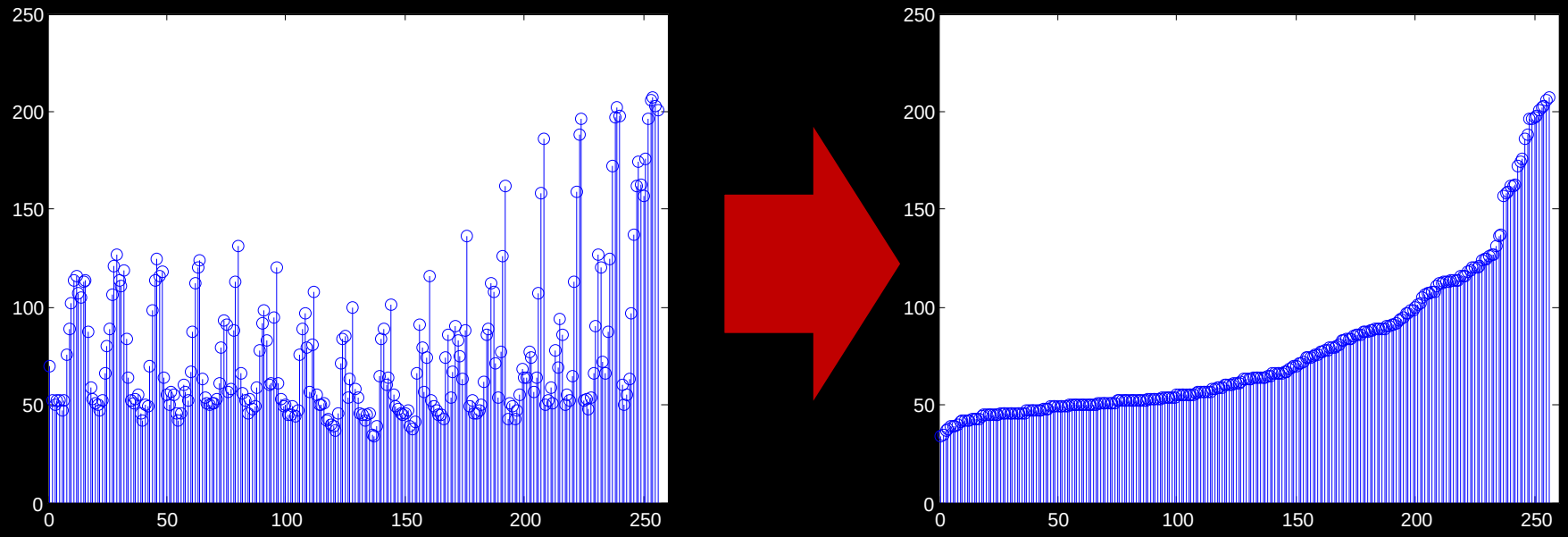
- In fact, we propose to perform a **different** permutation in each resolution level of the multi-scale pyramid:



- Naturally, these permutations will be applied reversely in the inverse transform.
- Thus, the difference between this and the plain 1D wavelet transform applied on $\mathbf{f}$ are the additional permutations, thus preserving the transform's **linearity** and unitarity, while also adapting to the input signal.

Building the Permutations (1)

- Lets start with $\mathbf{P}_0$ – the permutation applied on the incoming signal.
- Recall: the wavelet transform is most effective for piecewise regular signals.
  $\rightarrow$ thus, $\mathbf{P}_0$ should be chosen such that $\mathbf{P}_0\mathbf{f}$ is most "regular".
- So, … for example, we can simply permute by sorting the signal $\mathbf{f}$ …



Building the Permutations (2)

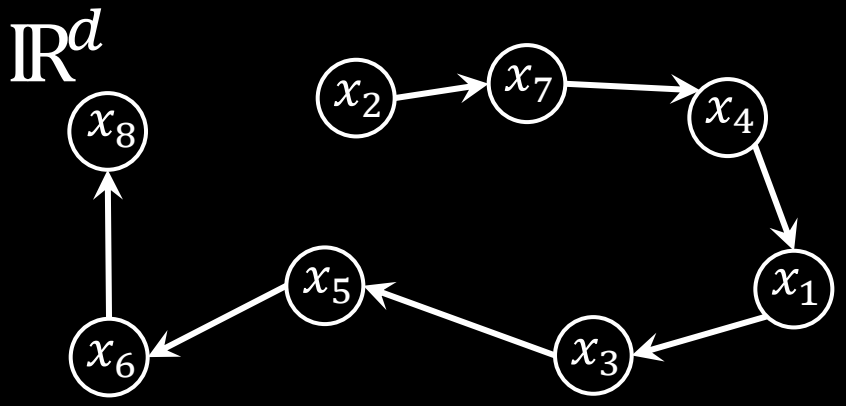
- **However**: we will be dealing with corrupted signals $\mathbf{f}$ (noisy, missing values, …) and thus such a sort operation is impossible.
- To our help comes the feature vectors in $\mathbf{X}$, which reflect on the order of the signal values, f_k. Recall:

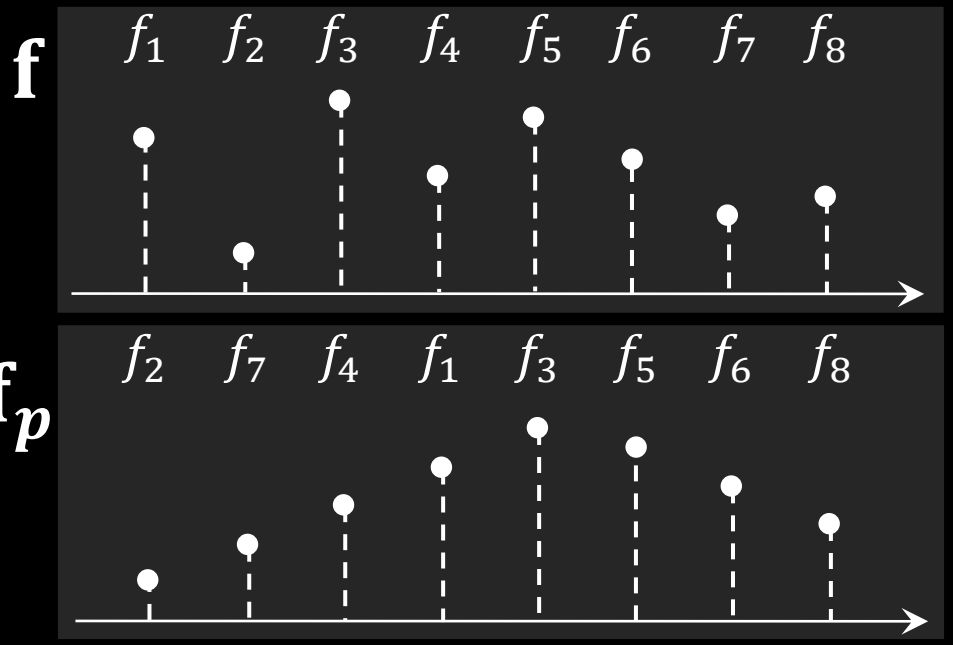
Small $w(x_i, x_j)$ implies small $|f(x_i) - f(x_j)|$ for almost every pair (i, j)

- Thus, instead of solving for the optimal permutation that "simplifies" $\mathbf{f}$, we order the features in $\mathbf{X}$ to the shortest path that visits in each point once, in what will be an instance of the Traveling-Salesman-Problem (TSP):

$$\min_{\mathrm{P}} \sum_{i=2}^{N} |f^p(i) - f^p(i-1)| \quad \Rightarrow \quad \min_{\mathrm{P}} \sum_{i=2}^{N} w(x_i^p, x_{i-1}^p)$$

Building the Permutations (3)



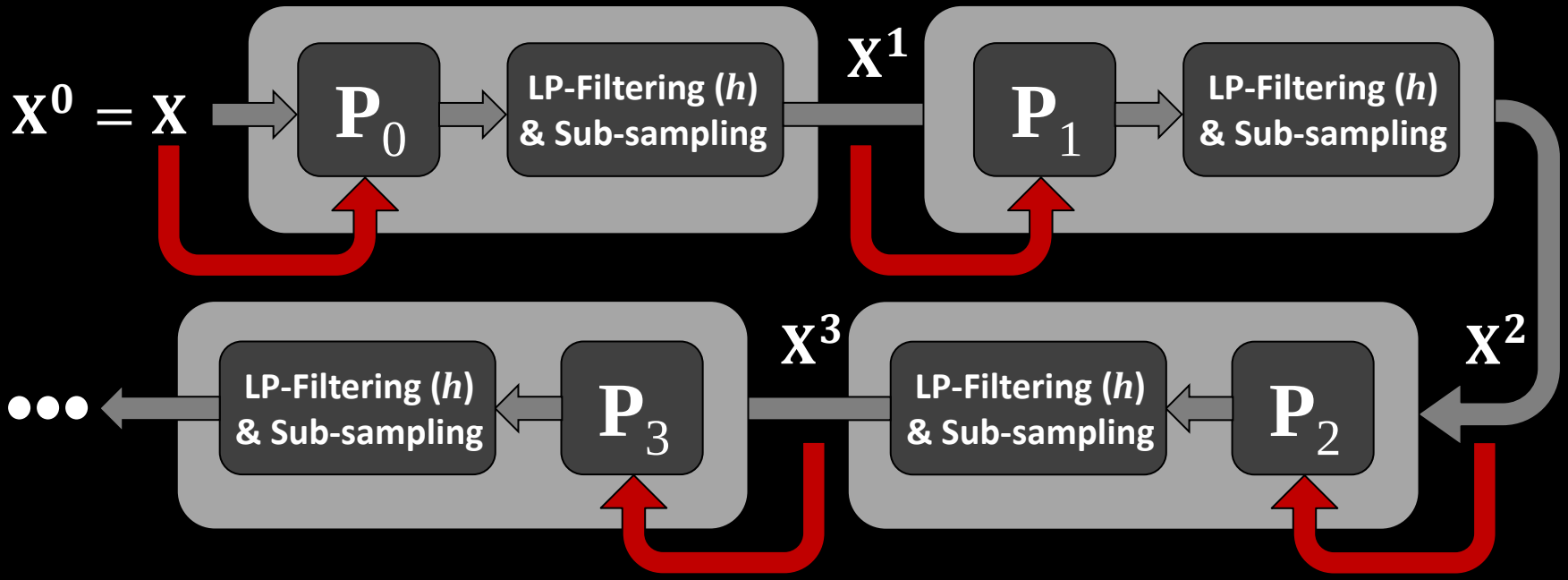


We handle the TSP task by a simple (and crude) approximation:

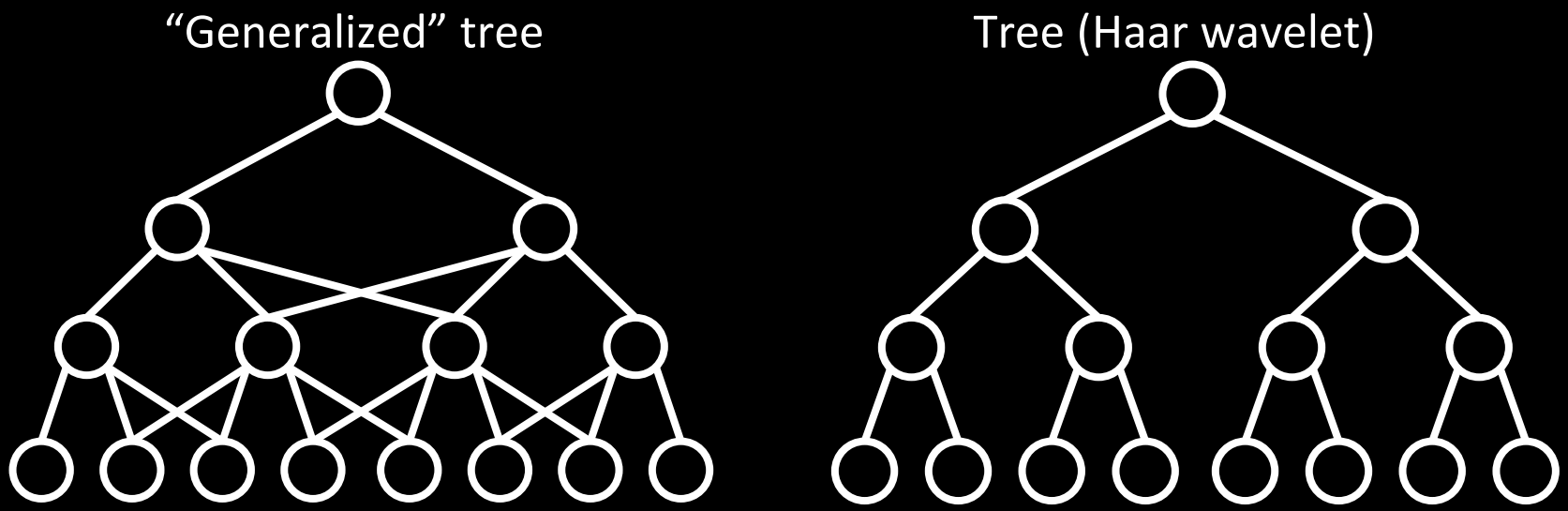
- Initialize with an arbitrary index j;
- Initialize the set of chosen indices to $\Omega(1)=\{j\}$;
- Repeat $k=1:1:N\text{-}1$ times:
  - Find x_i – the nearest neighbor to $x_{\Omega(k)}$ such that $i \notin \Omega$;
  - Set $\Omega(k+1)=\{i\}$;
- Result: the set Ω holds the proposed ordering.

Building the Permutations (4)

- So far we concentrated on P_0 at the finest level of the multi-scale pyramid.
- 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:



Why “Generalized Tree …”?



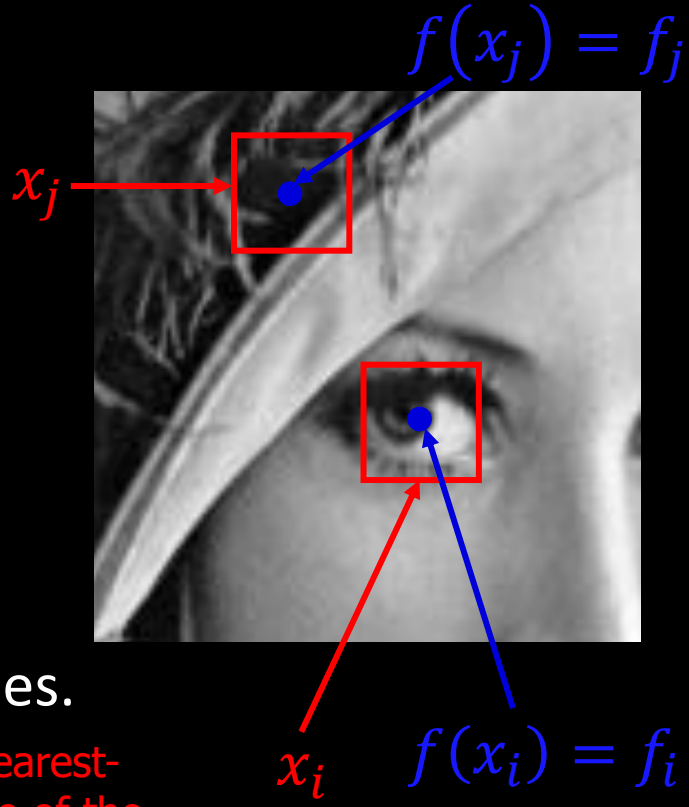
- Our proposed transform: **Generalized Tree-Based Wavelet Transform (GTBWT)**.
- We also developed a redundant version of this transform based on the stationary wavelet transform [Shensa, 1992] [Beylkin, 1992] – also related to the “A-Trous Wavelet” (will not be presented here).
- At this stage we could (or should) show how this works on point clouds/graphs, but we will take a different route and demonstrate these tools for images.

Part II – Handling Images

Using GTBWT for Image Denoising and Beyond

Could Images Fit This Data-Structure?

- Yes. Starting with an image of size $\sqrt{N} \times \sqrt{N}$, do the following:
  - Extract all possible patches of size $\sqrt{d} \times \sqrt{d}$ with complete overlaps – these will serve as the set of features (or coordinates) matrix $\mathbf{X}$.
  - The values $f(x_i) = f_i$ will be the center pixel in these patches.



- Once constructed this way, we forget all about spatial proximities in the image*, and start thinking in terms of (Euclidean) proximities between patches.

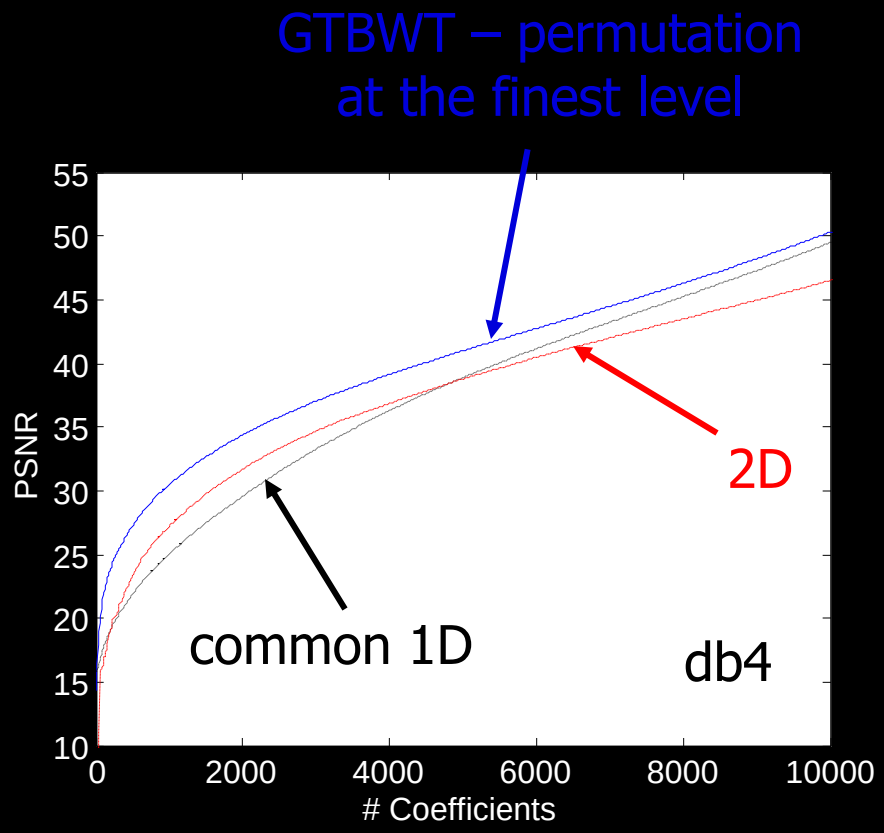
* Not exactly. Actually, if we search the nearest-neighbors within a limited window, some of the spatial proximity remains.

Lets Try (1)

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.

We measure efficiency by the m-term approximation error, i.e. reconstructing the image from m largest coefficients, zeroing the rest.

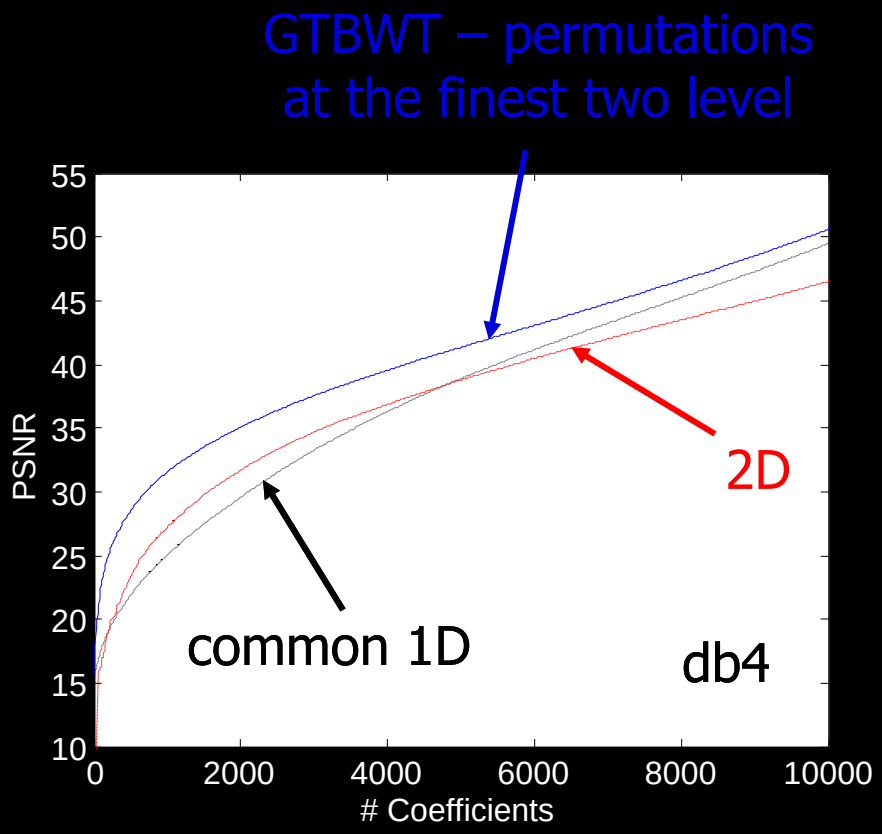


Lets Try (2)

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.

We measure efficiency by the m-term approximation error, i.e. reconstructing the image from m largest coefficients, zeroing the rest.

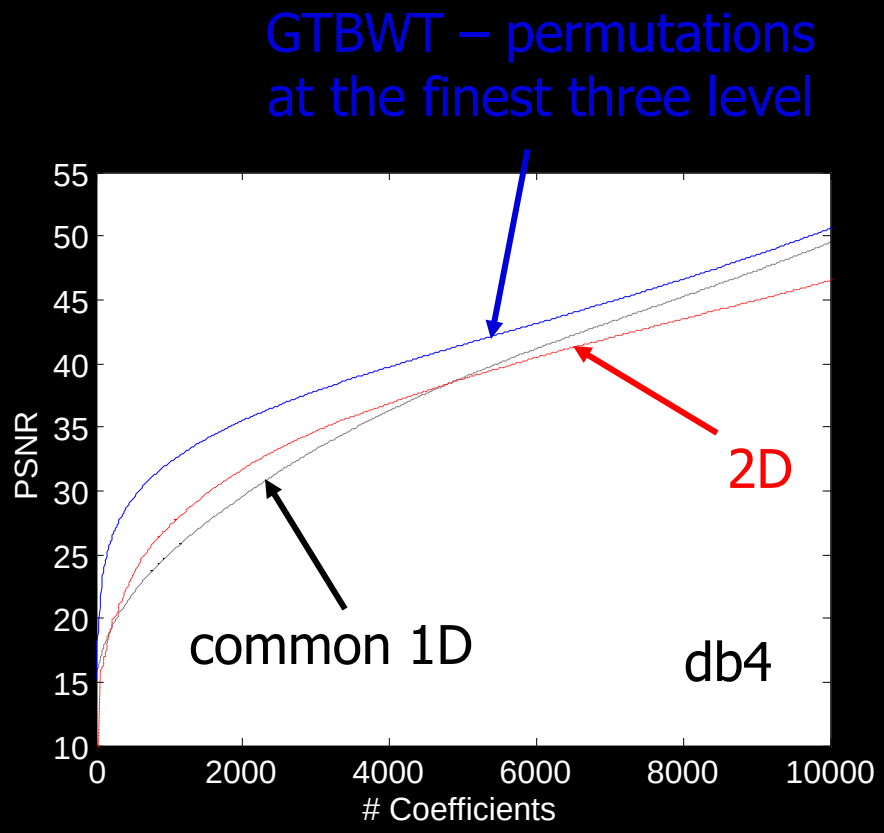


Lets Try (3)

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.

We measure efficiency by the m-term approximation error, i.e. reconstructing the image from m largest coefficients, zeroing the rest.

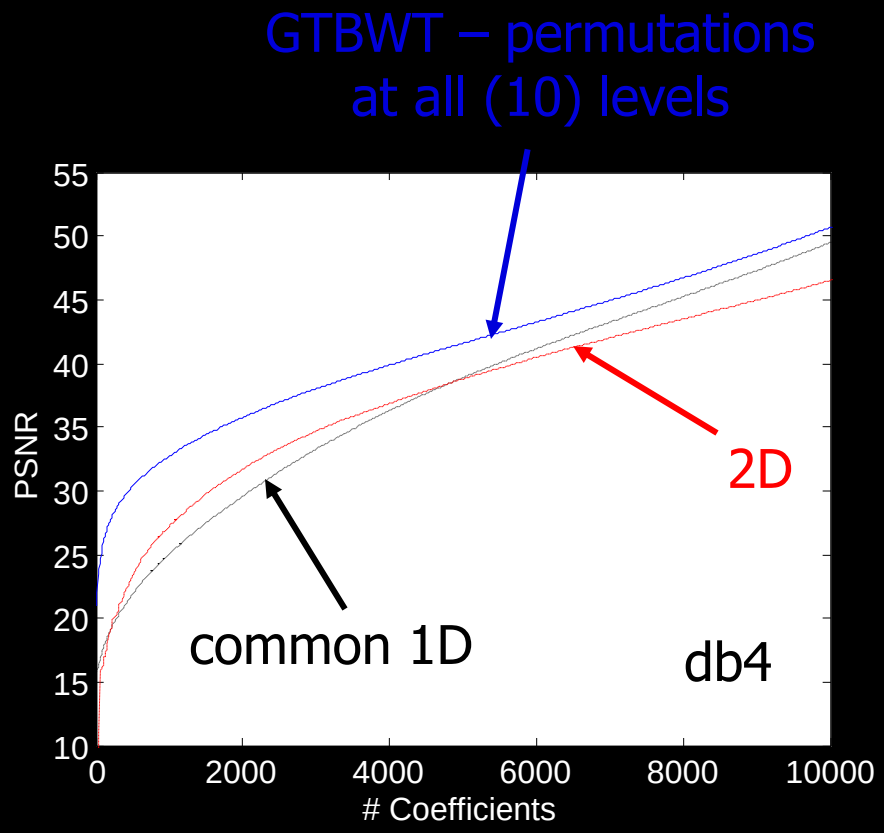


Lets Try (4)

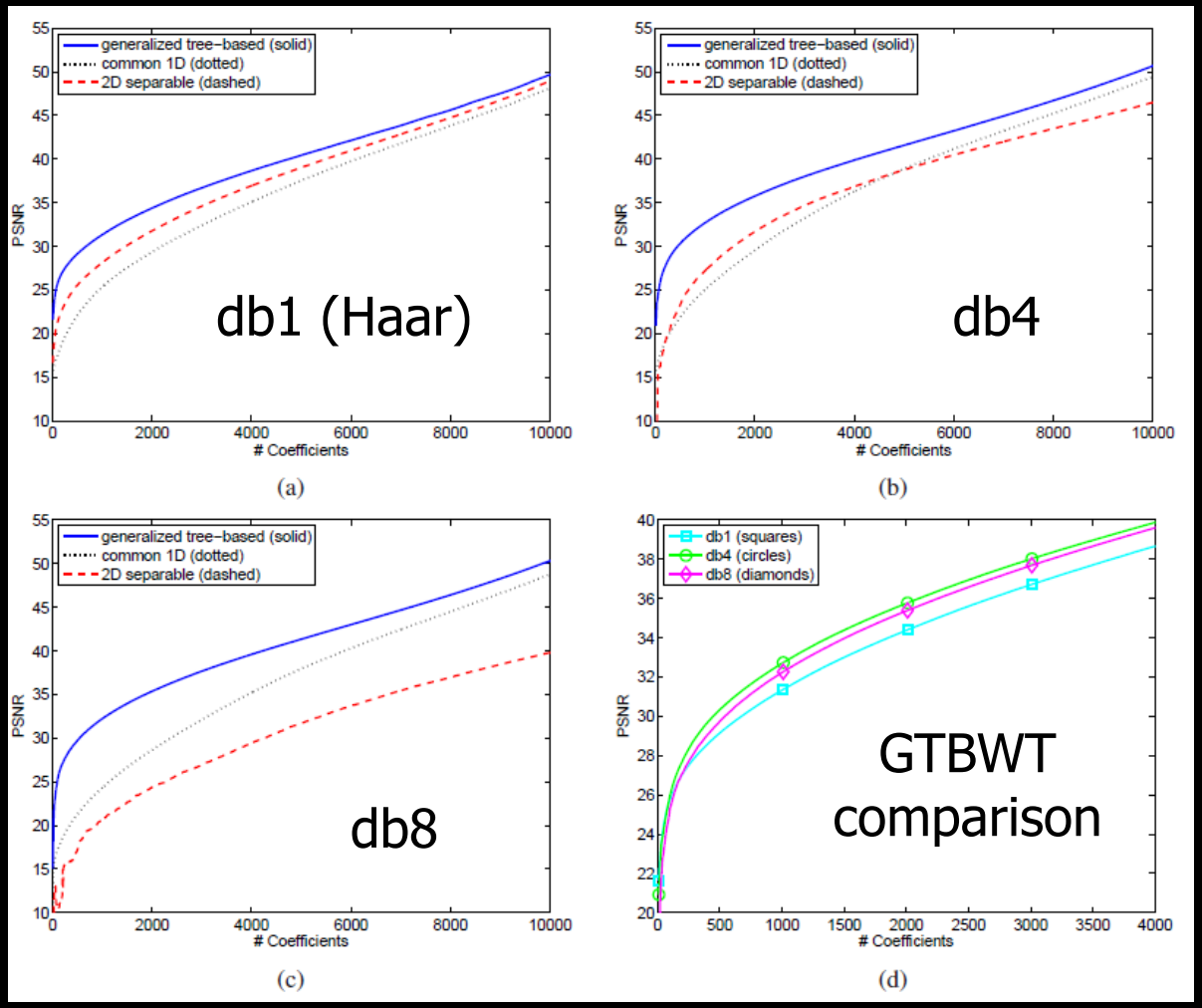
For a 128×128 center portion of the image Lenna, we compare the image representation efficiency of the

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

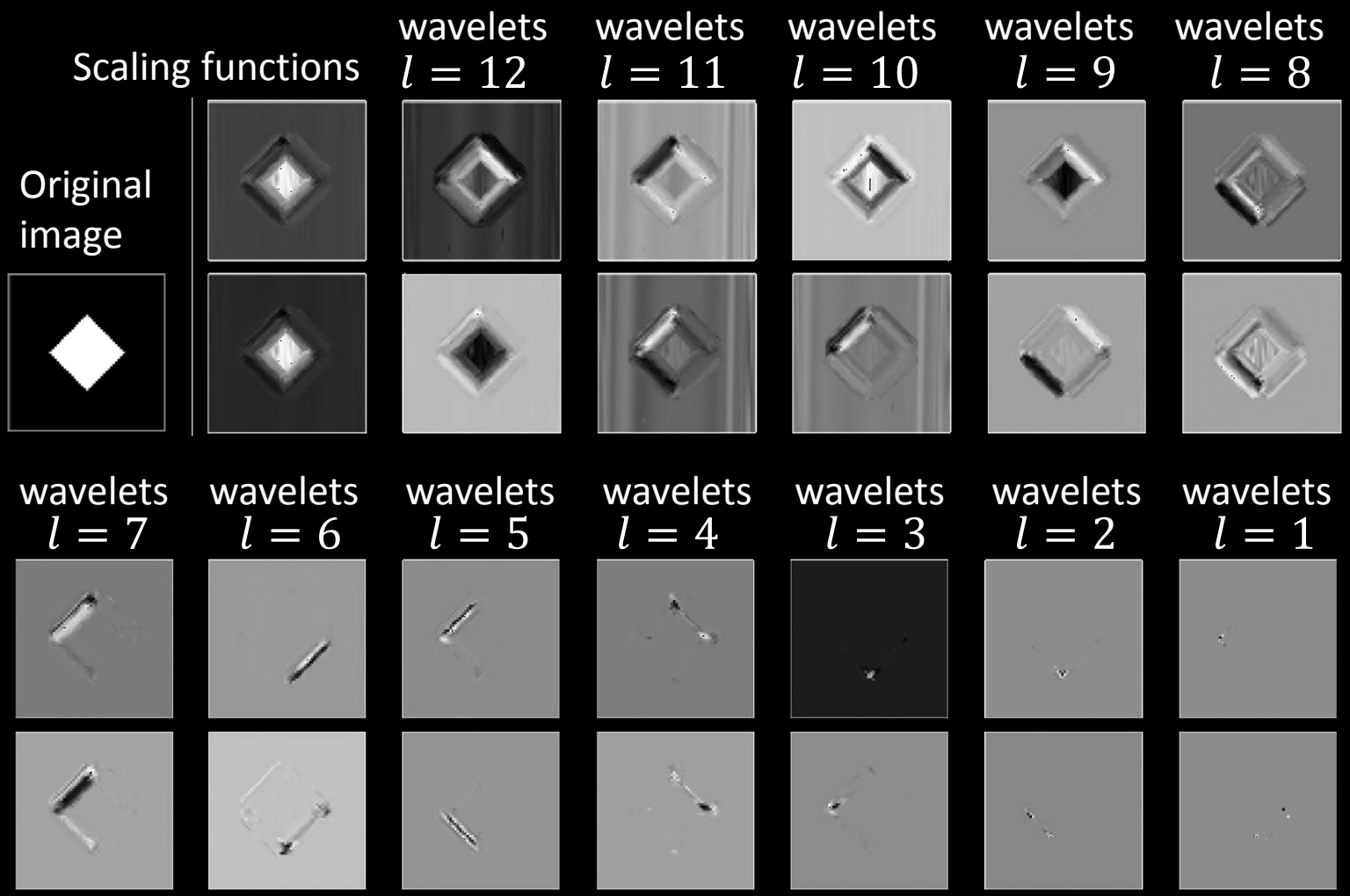
We measure efficiency by the m-term approximation error, i.e. reconstructing the image from m largest coefficients, zeroing the rest.



Comparison Between Different Wavelets



The Representation's Atoms – Synthetic Image



The Representation's Atoms – Lenna

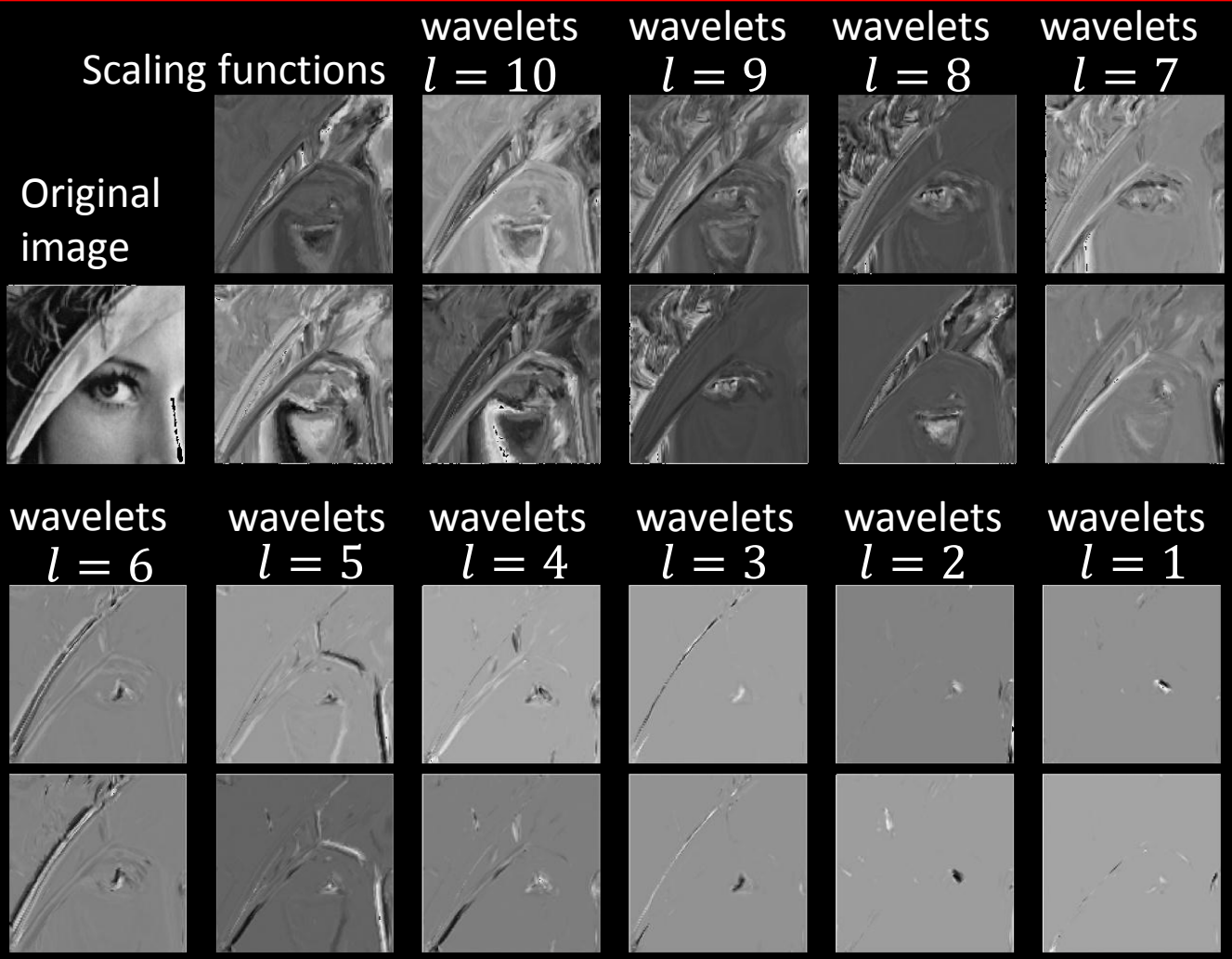


Image Denoising using GTBWT

- We assume that the noisy image, $\tilde{F}$, is a noisy version of a clean image, F, contaminated by white zero-mean Gaussian additive noise with known STD=σ.
- The vectors $\tilde{\mathbf{f}}$ and $\mathbf{f}$ are lexicographic ordering of the noisy and clean images.
- Our goal: recover $\mathbf{f}$ from $\tilde{\mathbf{f}}$, and we will do this using shrinkage over GTBWT:
  - We extract all patches from the noisy image as described above;
  - We apply the GTBWT on this data set;
  - The wavelet coefficients obtained go through a shrinkage operation; and
  - We transform back to get the final outcome.

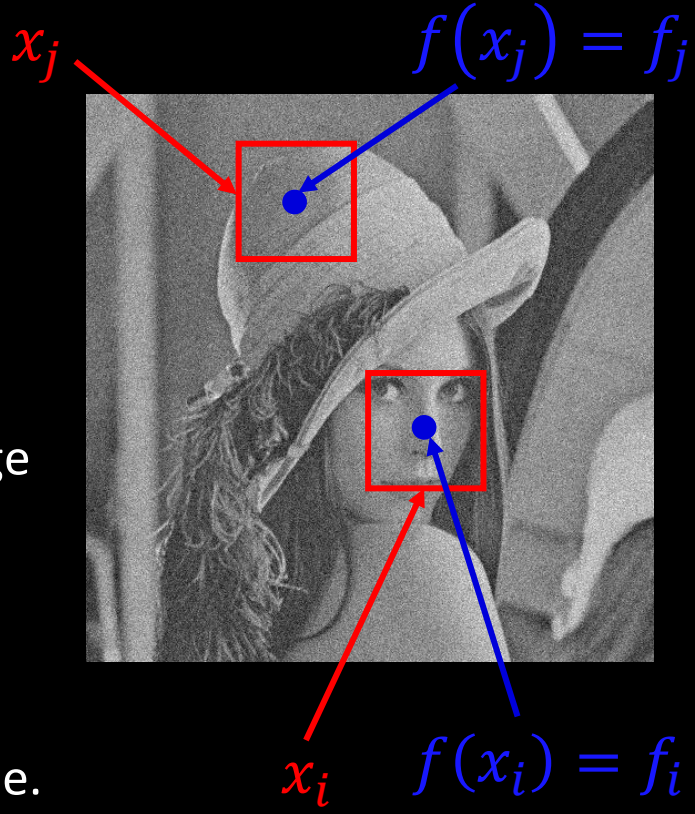


Image Denoising – Block-Diagram

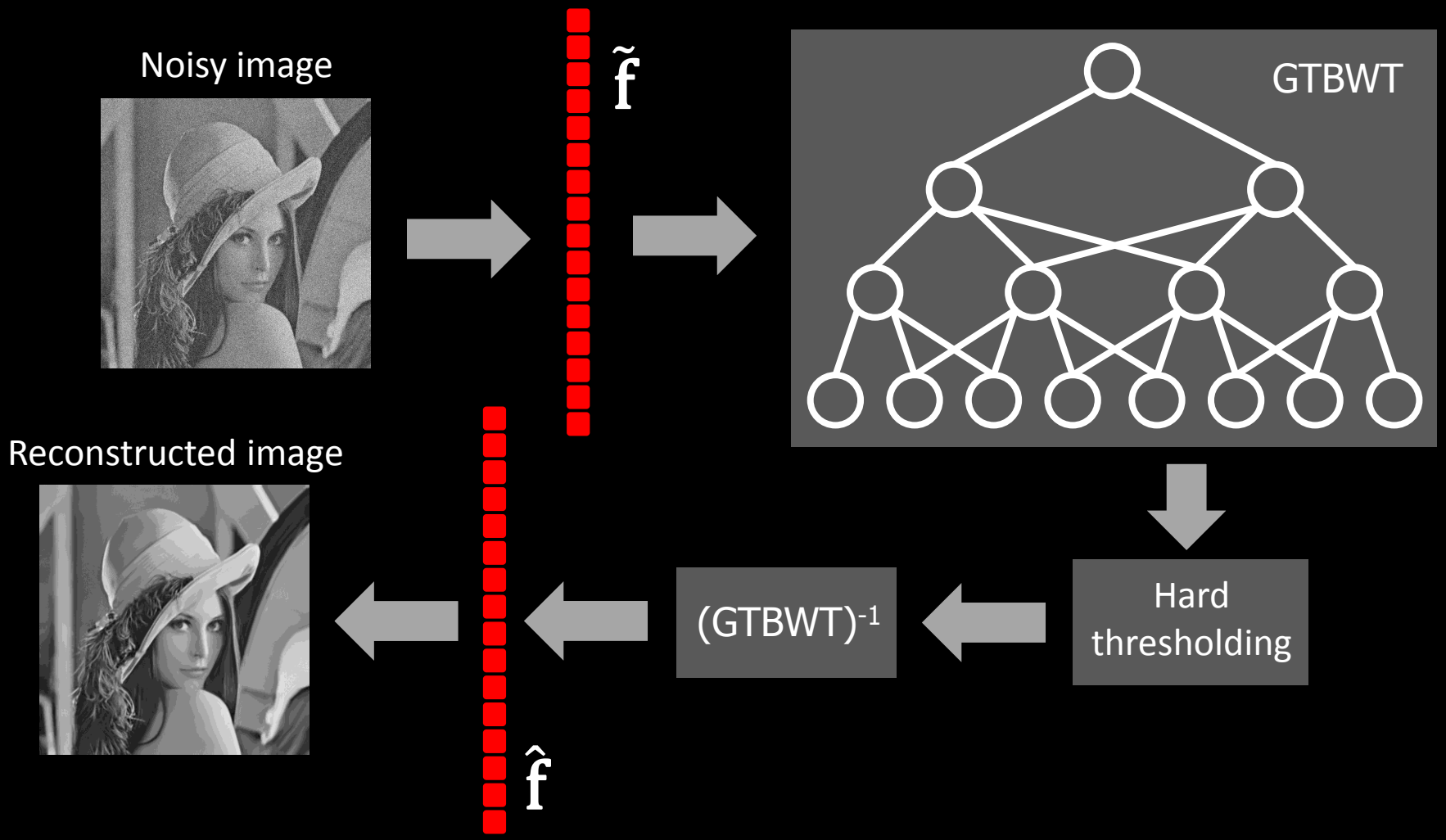


Image Denoising – Improvements

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

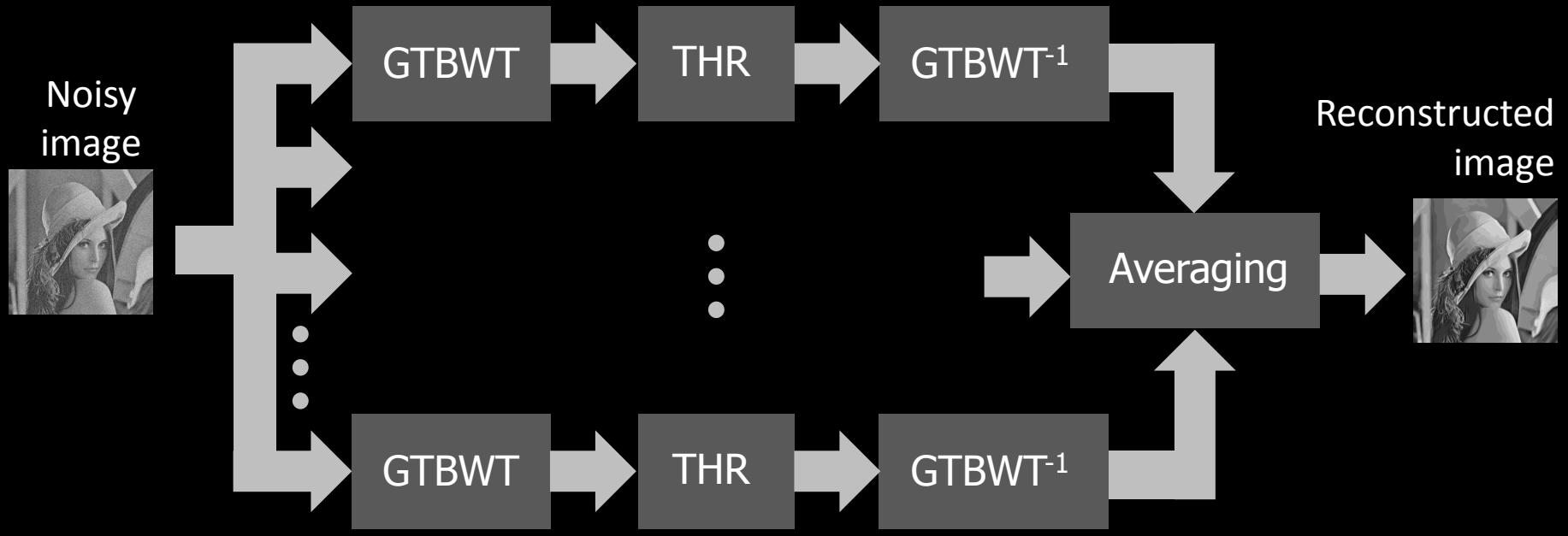


Image Denoising – Improvements

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.

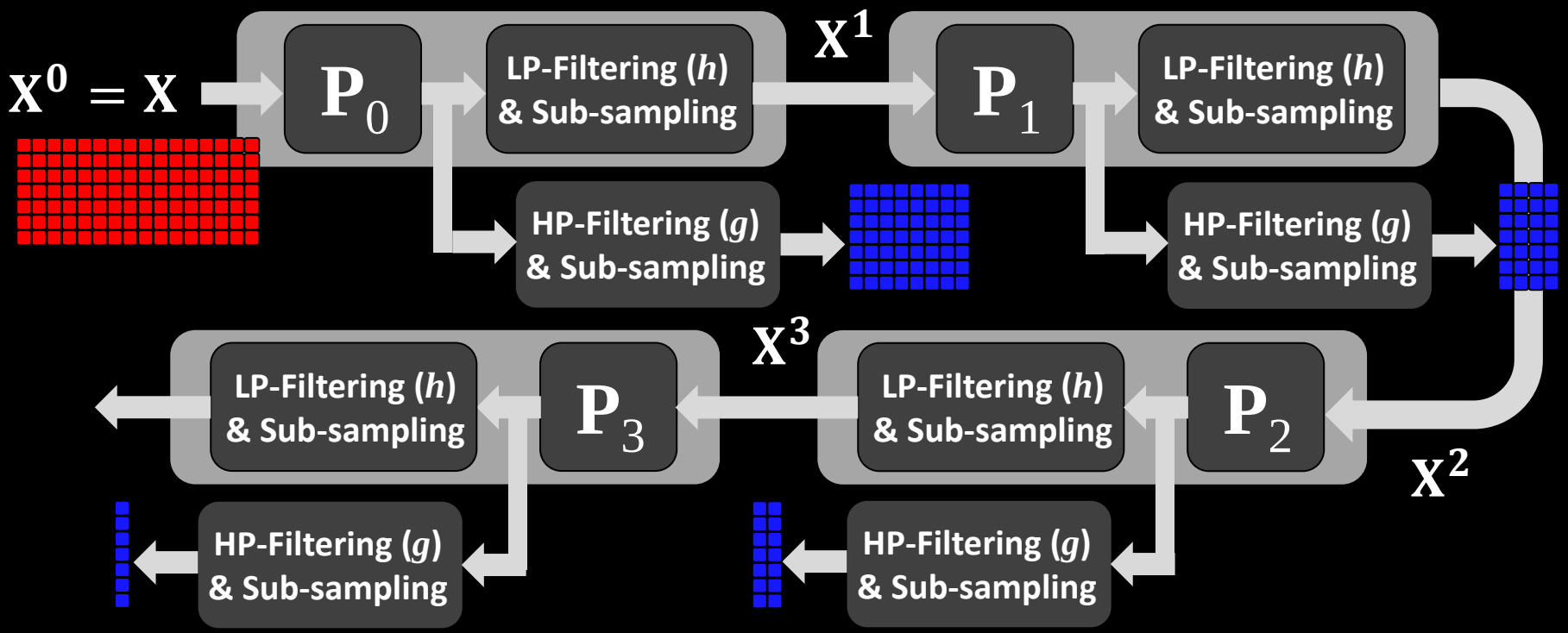
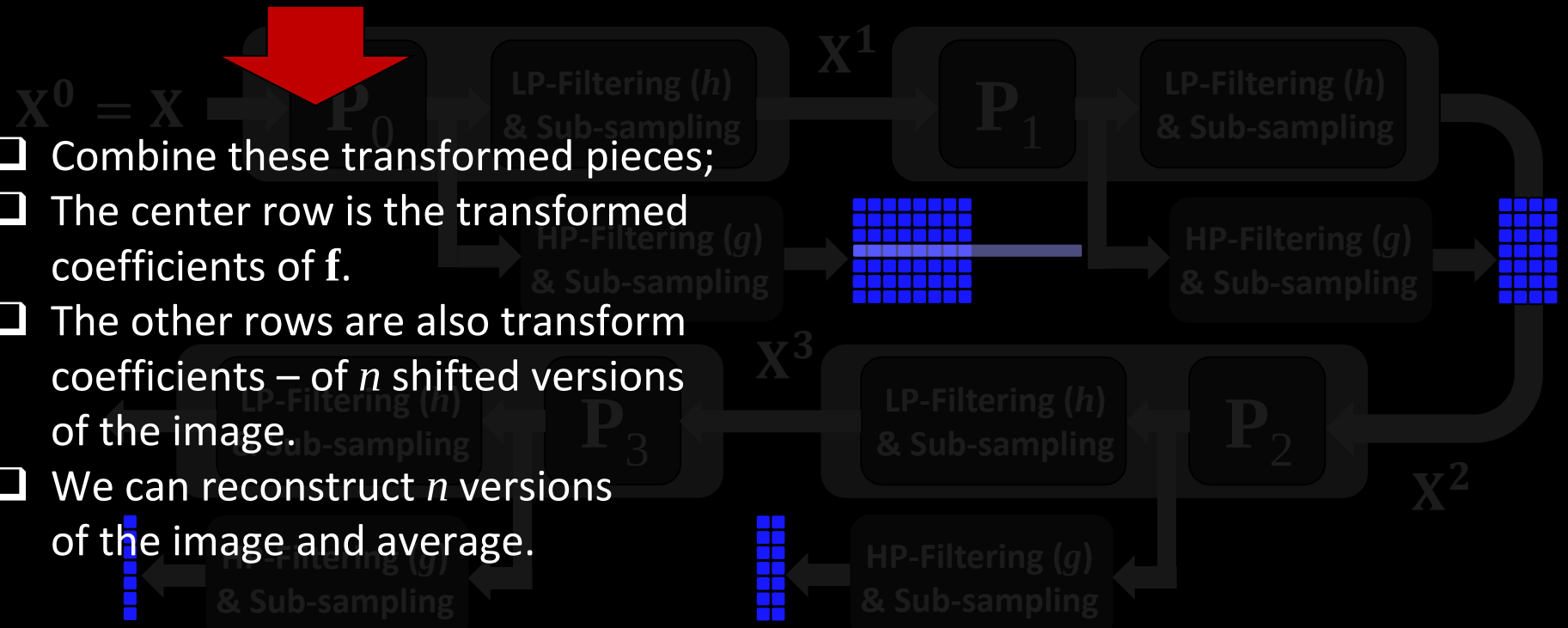


Image Denoising – Improvements

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 $\mathbf{f}$.
- The other rows are also transform coefficients – of n shifted versions of the image.
- We can reconstruct n versions of the image and average.

Image Denoising – Improvements

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.

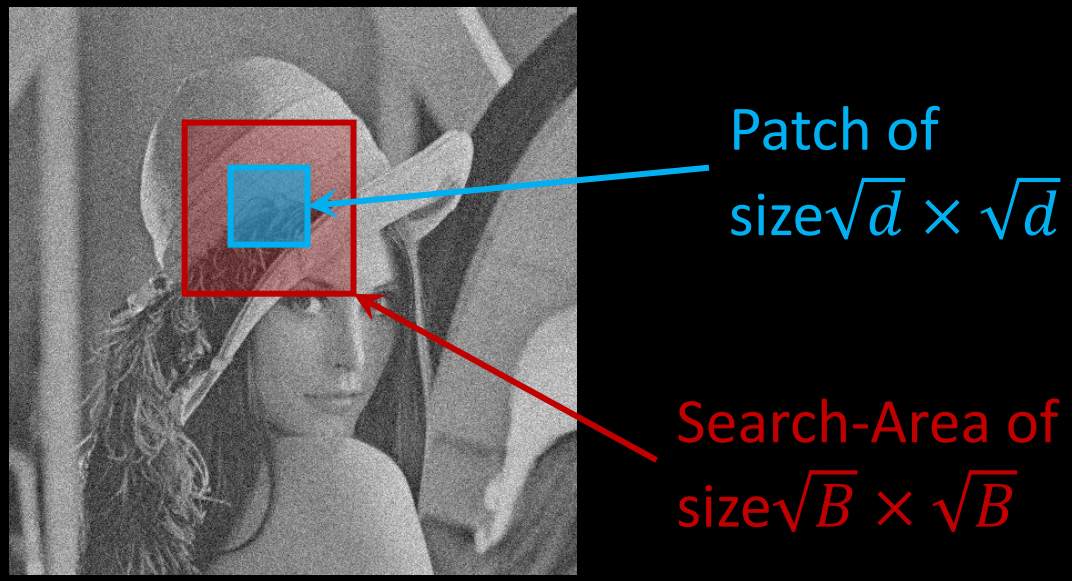
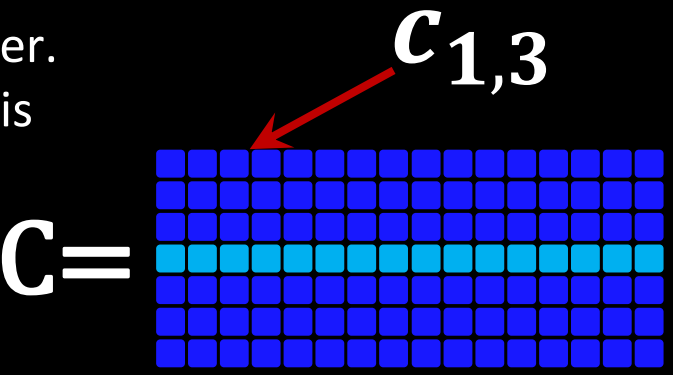


Image Denoising – Improvements

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 **C** is passed through the function:

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



- The proposed alternative would be to force "joint-sparsity" on the above array of coefficients, forcing all rows to share the same support:

$$c_{i,j} = \begin{cases} c_{i,j} & \left\|c_{*,j}\right\|_2 \geq T \\ 0 & \left\|c_{*,j}\right\|_2 < T \end{cases}$$

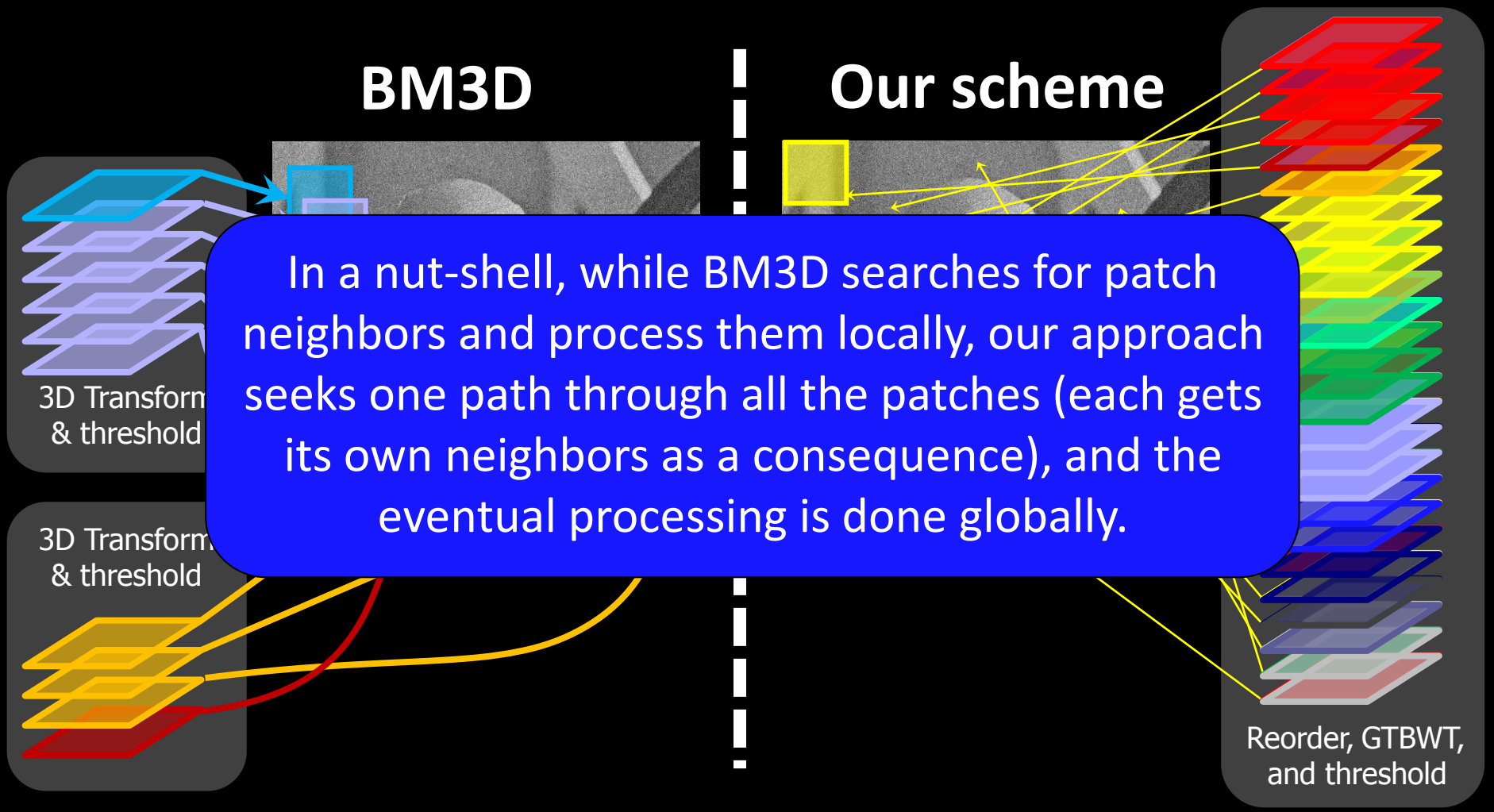
Image Denoising – Results

- 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.

σ/PSNR	Image	K-SVD	BM3D	GTBWT
10/28.14	Lena	35.51	35.93	35.87
	Barbara	34.44	34.98	34.94
25/20.18	Lena	31.36	32.08	32.16
	Barbara	29.57	30.72	30.75

- The PSNR results are quite good and competitive.
- What about run time?

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.

Part III – 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.

What next? The following extremes will appear soon:

- **Complicate things:** Use this transform as a regularizer in inverse problems – we have done that and getting excellent results; or
- **Simplify things:** Keep only the zero-resolution level of the pyramid – see illustration in the next slides.

What if we Keep Only the Reordering?

Noisy with σ=25 (20.18dB)

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

Order these patches as before

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

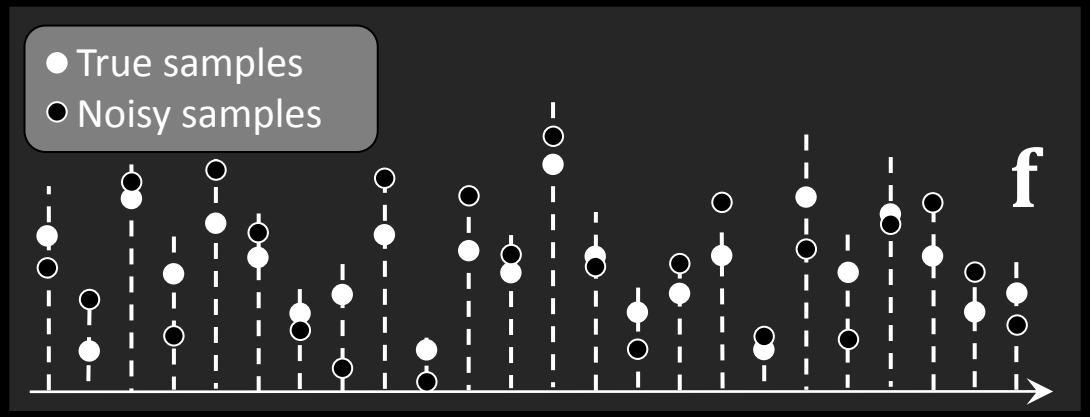
Smooth/filter the values along this row in a simple way

Reconstruction: 32.39dB *

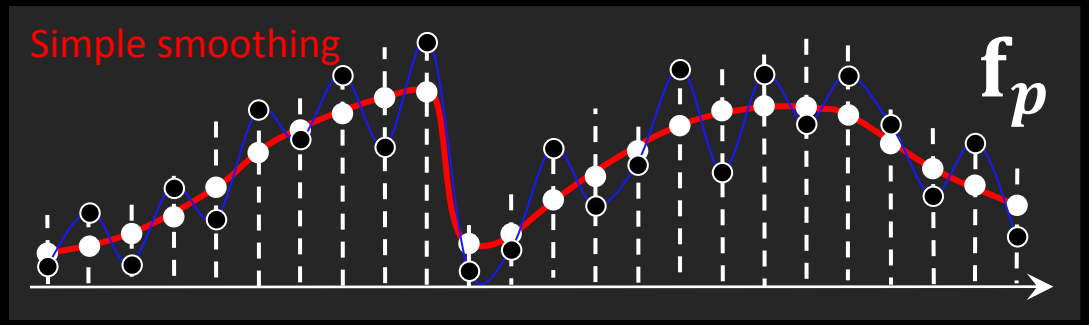
* This result is obtained with (i) cycle-spinning, (ii) sub-image averaging, and (iii) two iterations.

Why Should This Work?

Noisy with σ=25 (20.18dB)



Ordering based on the noisy pixels



Reconstruction: 32.39dB

The “Simple Smoothing” We Do

While simple smoothing works fine, one could do better by:

1. Training the smoothing filter to perform “optimally”:
   - ❑ Take a clean image and create a noisy version of it.
   - ❑ Apply the denoising algorithm as described, but when getting to the stage where a filter is to be used, optimize its taps so that the reconstruction MSE (w.r.t. to the clean image) is minimized.
2. Applying several filters and switching between them base don the local patch content.
3. Turning to non-linear edge-preserving filters (e.g. bilateral filter).

➔ The results above correspond to ideas 1 and 2

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

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

Interpolate missing values along this row in a simple way

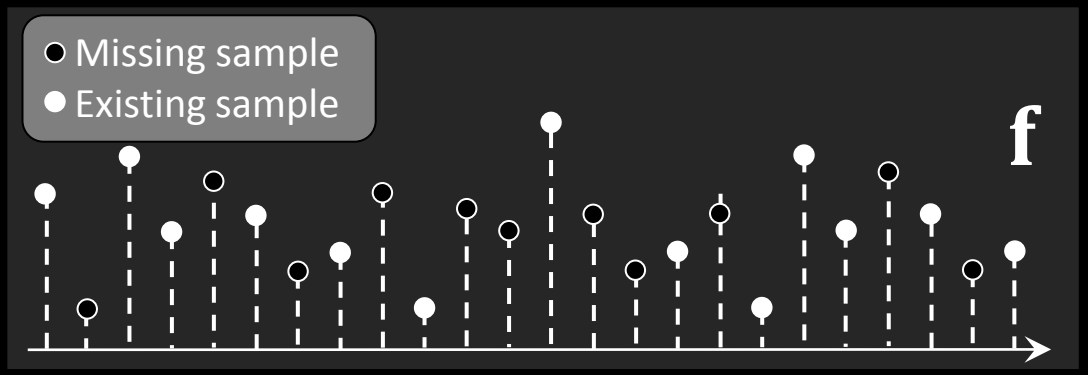
Reconstruction: 27.15dB*

* 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



Ordering