Wavelet for Graphs and its Deployment to Image Processing

Michael Elad

The Computer Science Department

The Technion – Israel Institute of technology

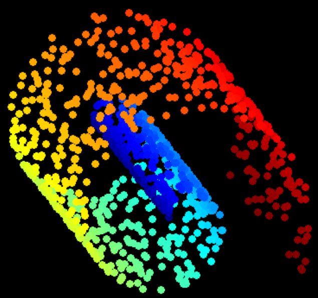
Haifa 32000, Israel

Technion

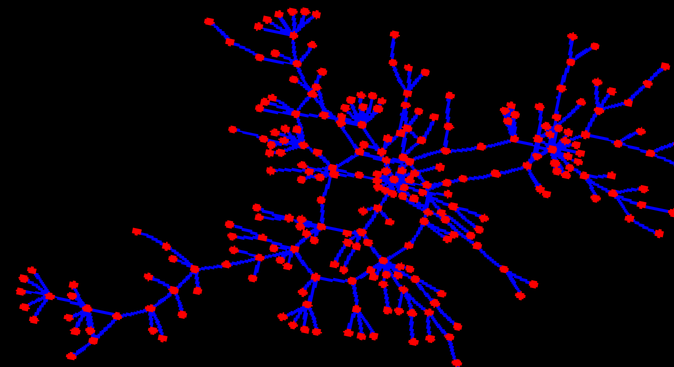
Israel Institute of Technology

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

This Talk is About …



Processing of Non-Conventionally Structured Signals



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

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

This Talk is About …

As you will see, we will use the developed tools to process "regular" signals (e.g., images) , handling them differently and more effectively

This is Joint Work With

Idan Ram   Israel Cohen

The EE department - the Technion

1. I. Ram, M. Elad, and I. Cohen, "Generalized Tree-Based Wavelet Transform", IEEE Trans. Signal Processing, vol. 59, no. 9, pp. 4199–4209, 2011.
2. 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.
3. I. Ram, M. Elad, and I. Cohen, "The RTBWT Frame – Theory and Use for Images", to appear in IEEE Trans. on Image Processing.
4. I. Ram, M. Elad, and I. Cohen, "Image Processing using Smooth Ordering of its Patches", IEEE Transactions on Image Processing, Vol. 22, No. 7, pp. 2764–2774 , July 2013.
5. I. Ram, I. Cohen, and M. Elad, "Facial Image Compression using Patch-Ordering-Based Adaptive Wavelet Transform", Submitted to IEEE Signal Processing Letters.

Part I – GTBWT

Generalized Tree-Based Wavelet Transform – The Basics

This part is taken from the following two papers :

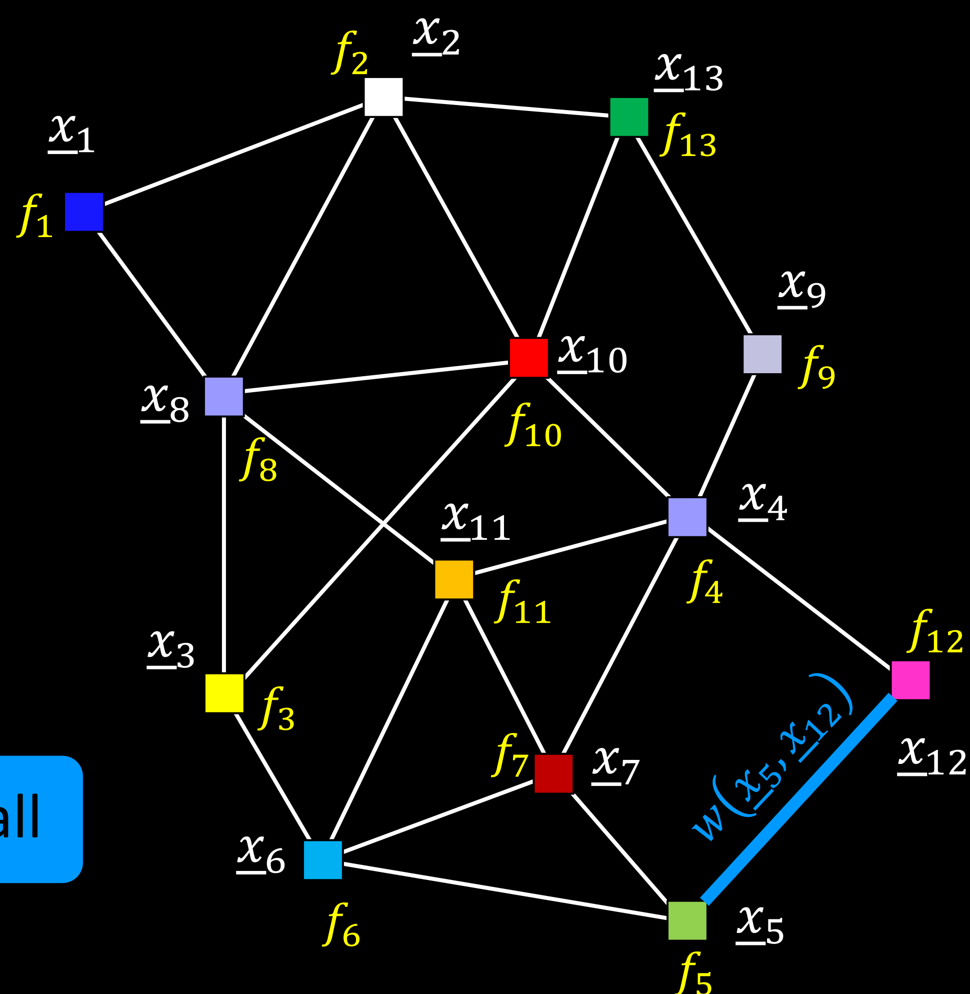
- I. Ram, M. Elad, and I. Cohen, "Generalized Tree-Based Wavelet Transform", IEEE Trans. Signal Processing, vol. 59, no. 9, pp. 4199–4209, 2011.
- I. Ram, M. Elad, and I. Cohen, "Redundant Wavelets on Graphs and High Dimensional Data Clouds", IEEE Signal Processing Letters, Vol. 19, No. 5, pp. 291–294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

Problem Formulation

- We are given a graph:
  - The $i-th$ node is characterized by a d-dimen. feature vector $\underline{x}_i$
  - The $i-th$ node has a value f_i
  - The edge between the $i-th$ and $j-th$ nodes carries the distance $w(\underline{x}_i, \underline{x}_j)$ for an arbitrary distance measure $w(\cdot,\cdot)$.
- Assumption: a “short edge” implies close-by values, i.e.

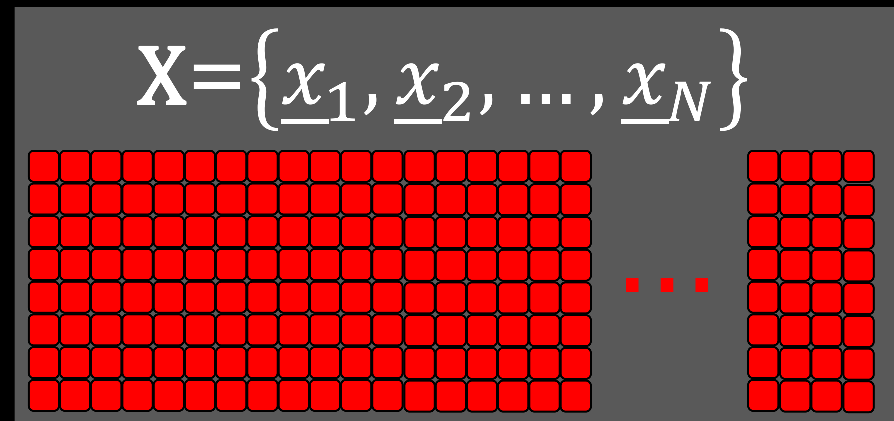
$$w(\underline{x}_i, \underline{x}_j) \text{ small} \rightarrow |f_i - f_j| \text{ small}$$

for almost every pair (i, j).



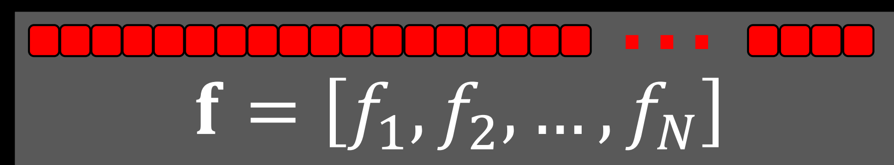
A Different Way to Look at this Data

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



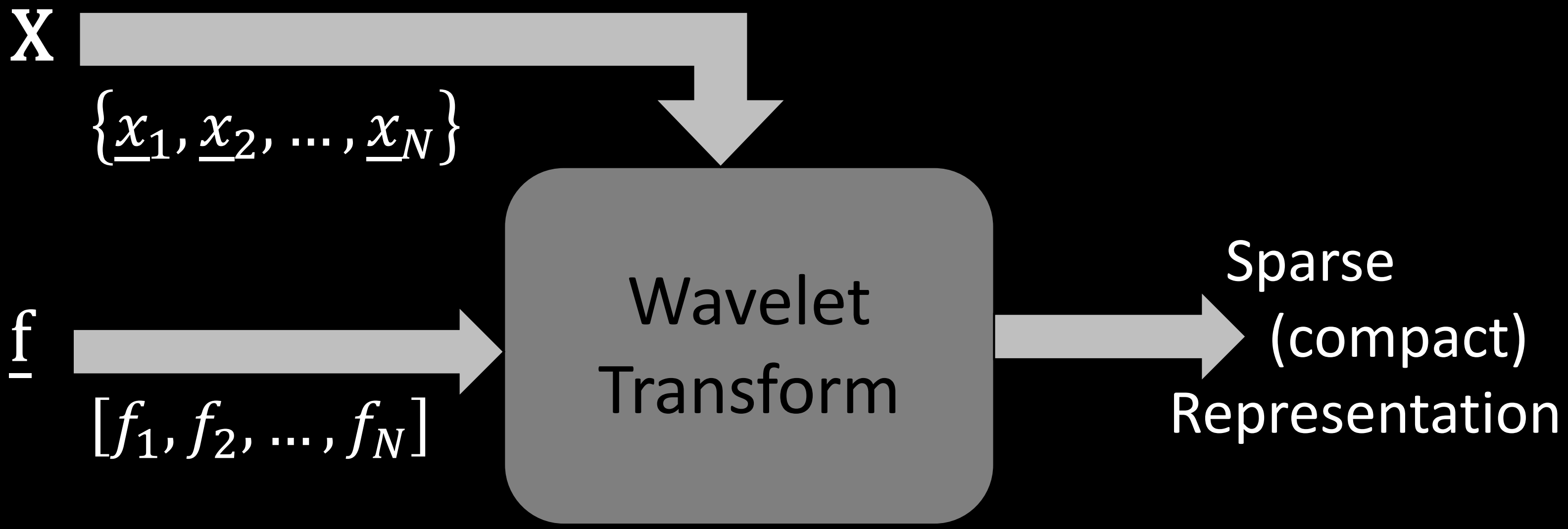
- A scalar function is defined on these coordinates, $f: \mathbf{X} \to \mathbb{R}$, giving $\mathbf{f} = [f_1, f_2, \dots, f_N]$.

- We may regard this dataset as a set of N samples taken from a high dimensional function $f: \mathbb{R}^d \to \mathbb{R}$.



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

Our Goal



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.

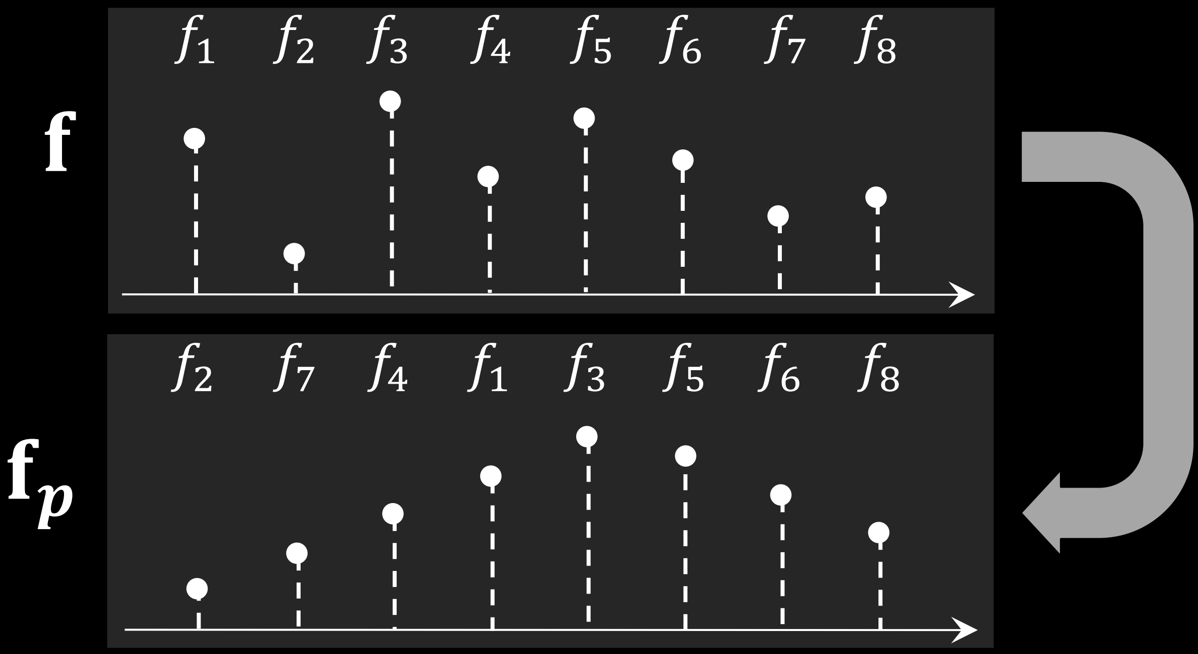
Wavelet for Graphs – A Wonderful Idea

I wish we would have thought of it first …

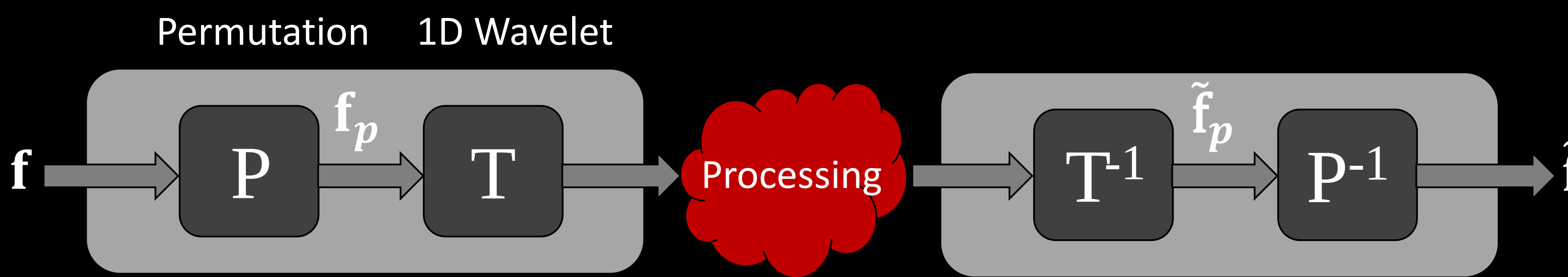
- "Diffusion Wavelets"
  R. R. Coifman, and M. Maggioni, 2006.
- "Multiscale Methods for Data on Graphs and Irregular …. Situations"
  M. Jansen, G. P. Nason, and B. W. Silverman, 2008.
- "Wavelets on Graph via Spectal Graph Theory"
  D. K. Hammond, and P. Vandergheynst, and R. Gribonval, 2010.
- "Multiscale Wavelets on Trees, Graphs and High … 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, 2012.

…

The Main Idea (1) - Permutation

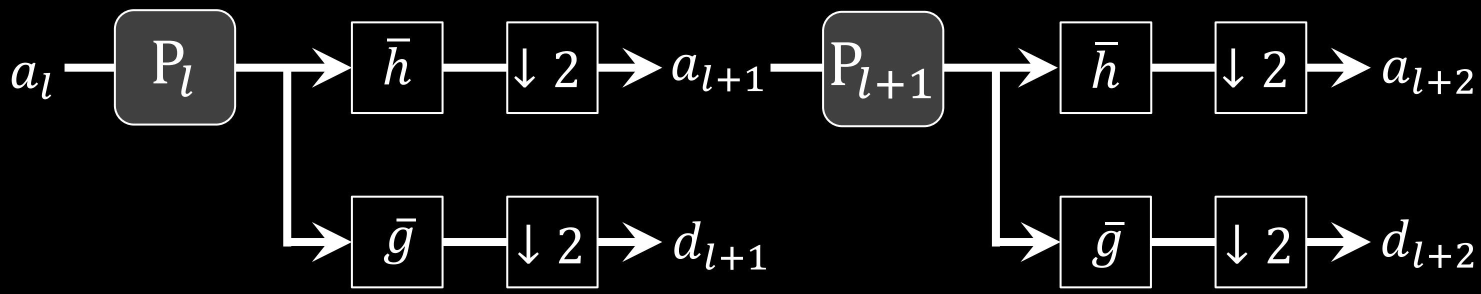


Permutation using $\mathbf{X} = \{x_1, x_2, \dots, x_N\}$



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)

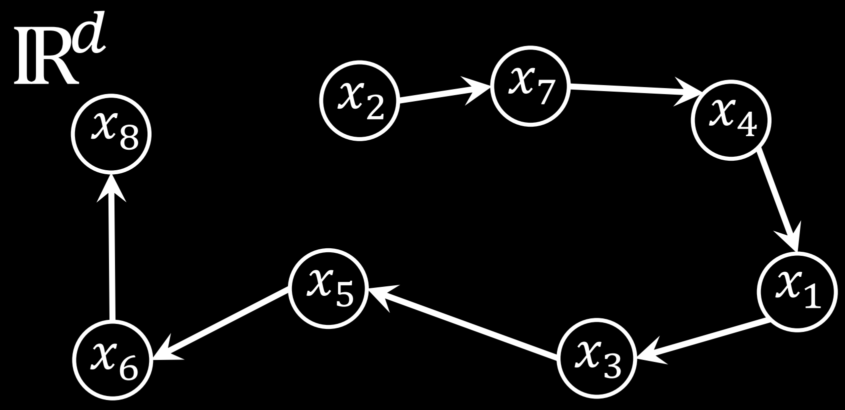
- Lets start with P_0 – the permutation applied on the incoming signal.
- Recall: the wavelet transform is most effective for piecewise regular signals. $\longrightarrow$ thus, P_0 should be chosen such that $\mathrm{P}_0\mathbf{f}$ is most "regular".
- Lets use the feature vectors in $\mathbf{X}$, reflecting the order of the values, f_k. Recall:

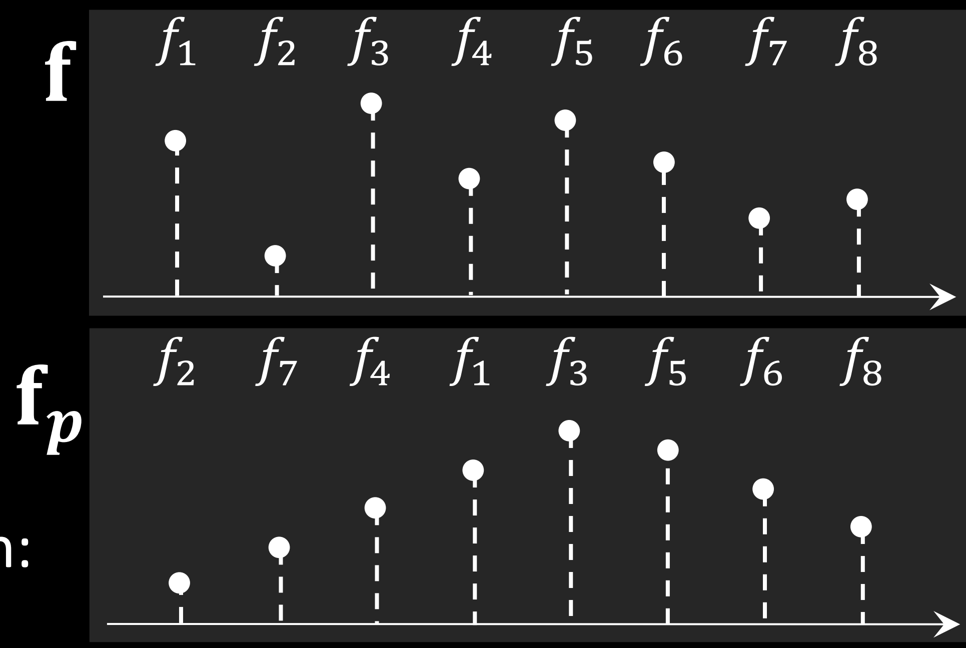
Small $w(x_i, x_j)$ implies small $|f(x_i) - f(x_j)|$ for almost every pair (i, j)

- 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 (2)



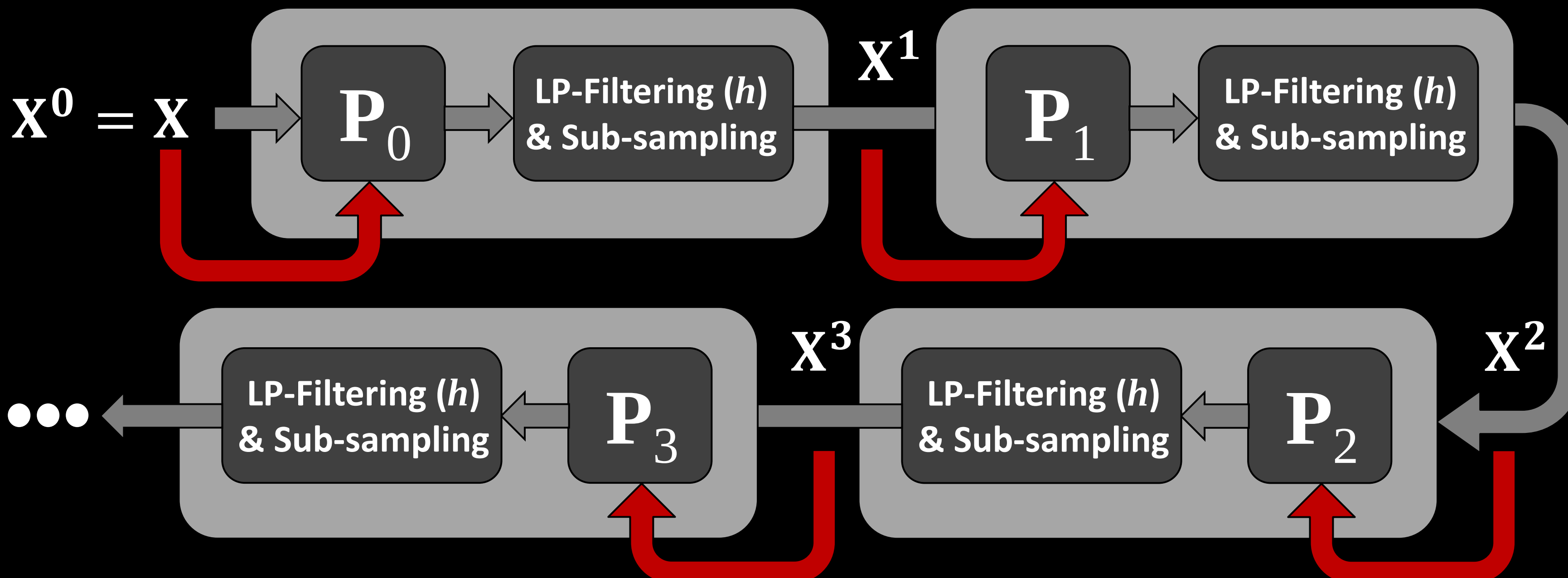


We handle the TSP task by a greedy (and crude) approximation:

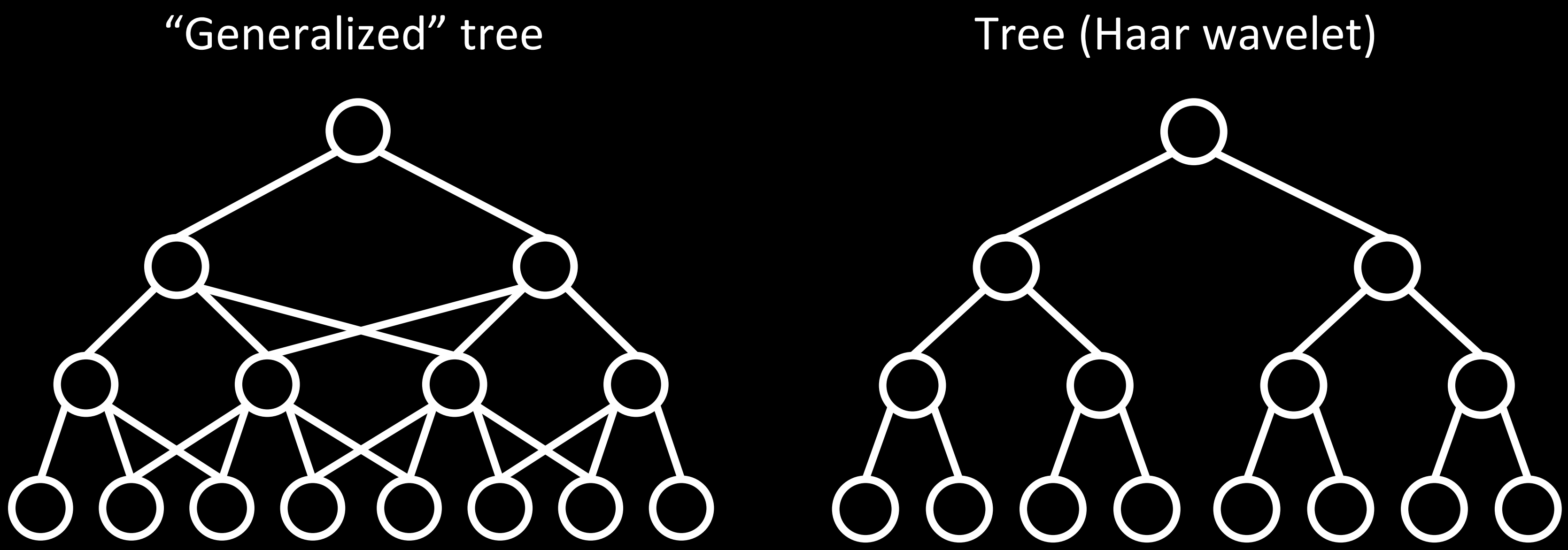
- Initialize with an arbitrary index j;
- Initialize the set of chosen indices to $\Omega(1)=\{j\}$;
- Repeat $k=1:1:N\text{-}1$ times:
  - Find x_i – the nearest neighbor to $x_{\Omega(k)}$ such that $i \notin \Omega$;
  - Set $\Omega(k+1)=\{i\}$;
- Result: the set Ω holds the proposed ordering.

Building the Permutations (3)

- 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:



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).

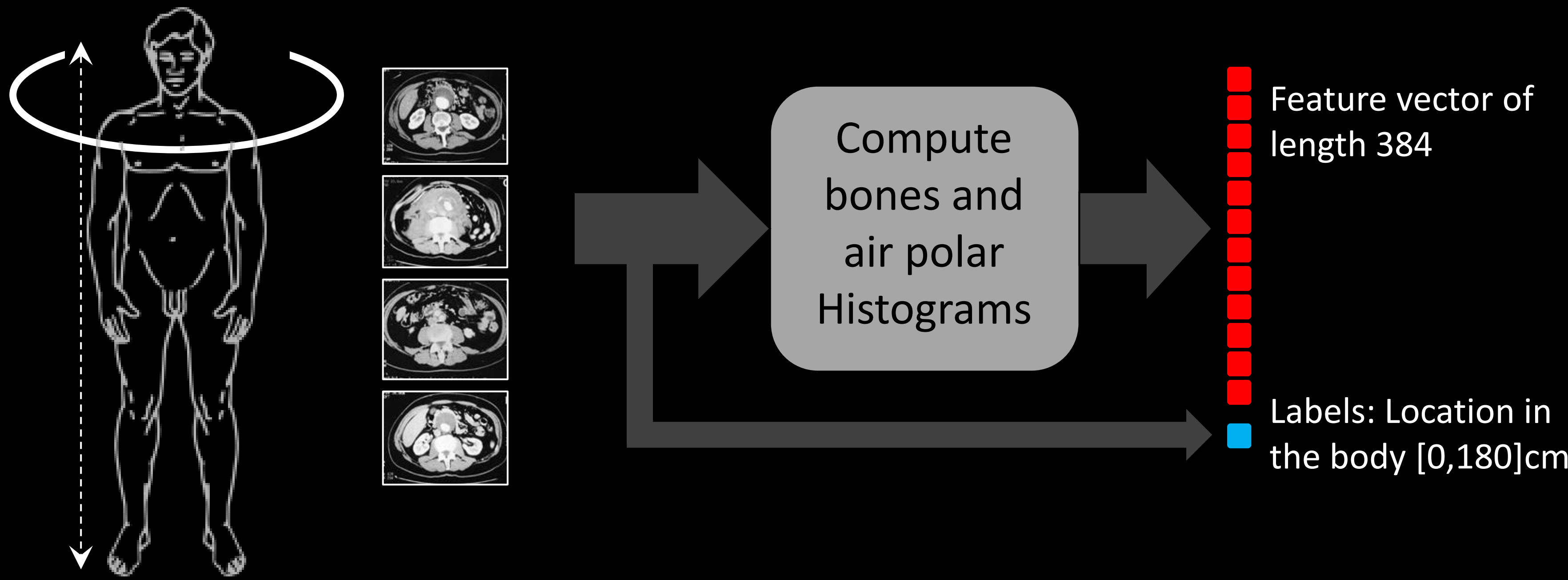
Treating Graph/Cloud-of-points

- 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.

SKIP?

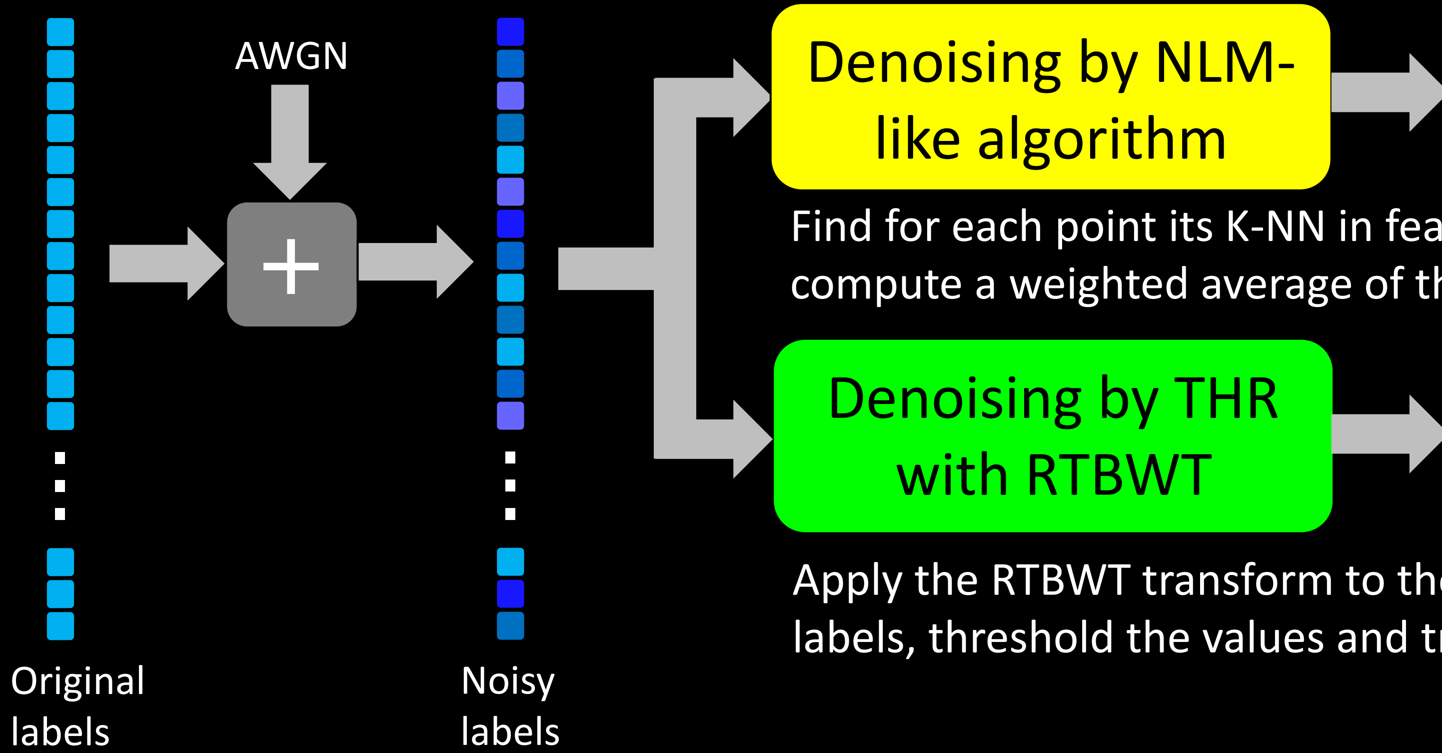
Treating Graphs: The Data

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).

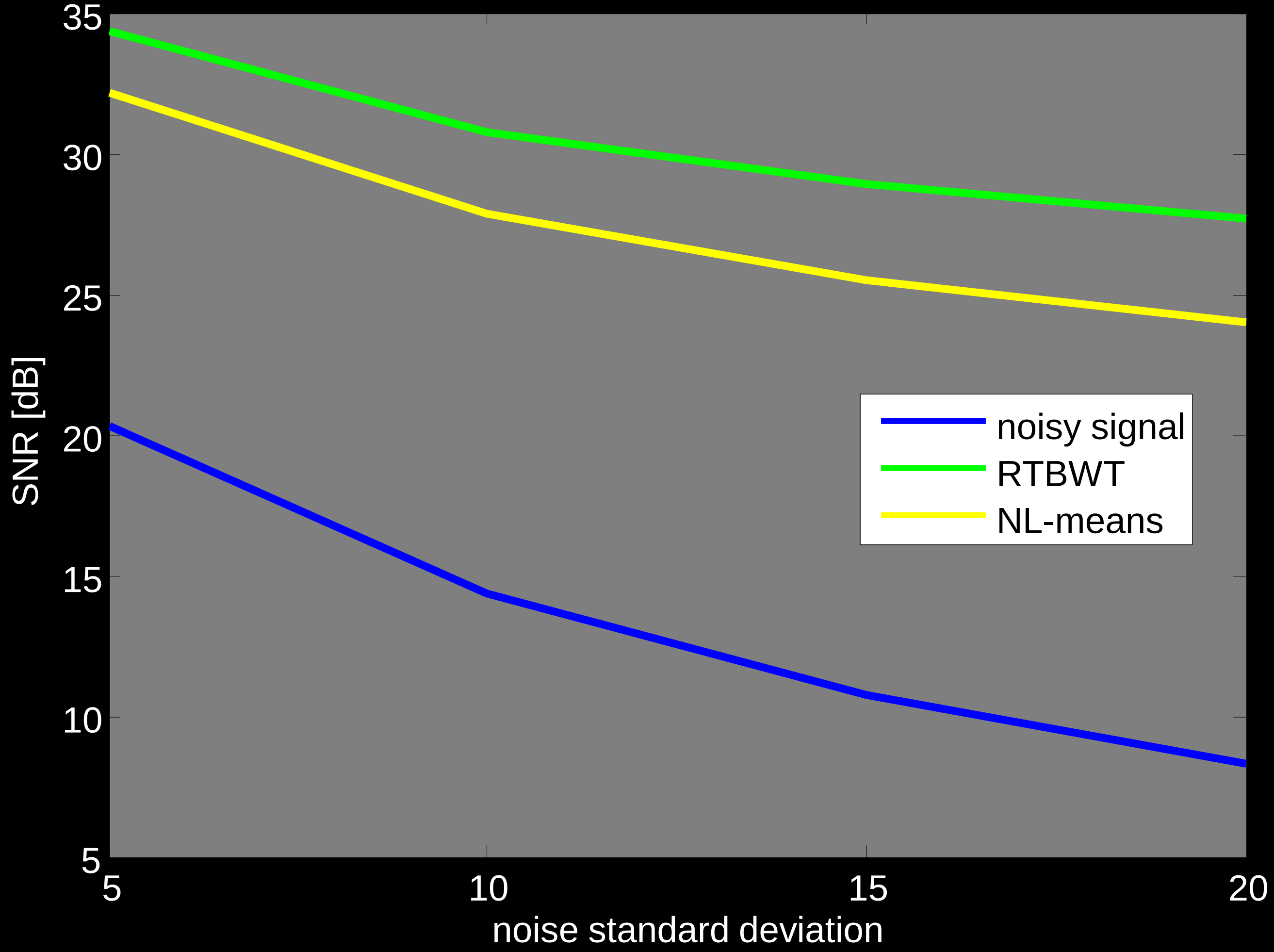
Treating Graphs: Denoising



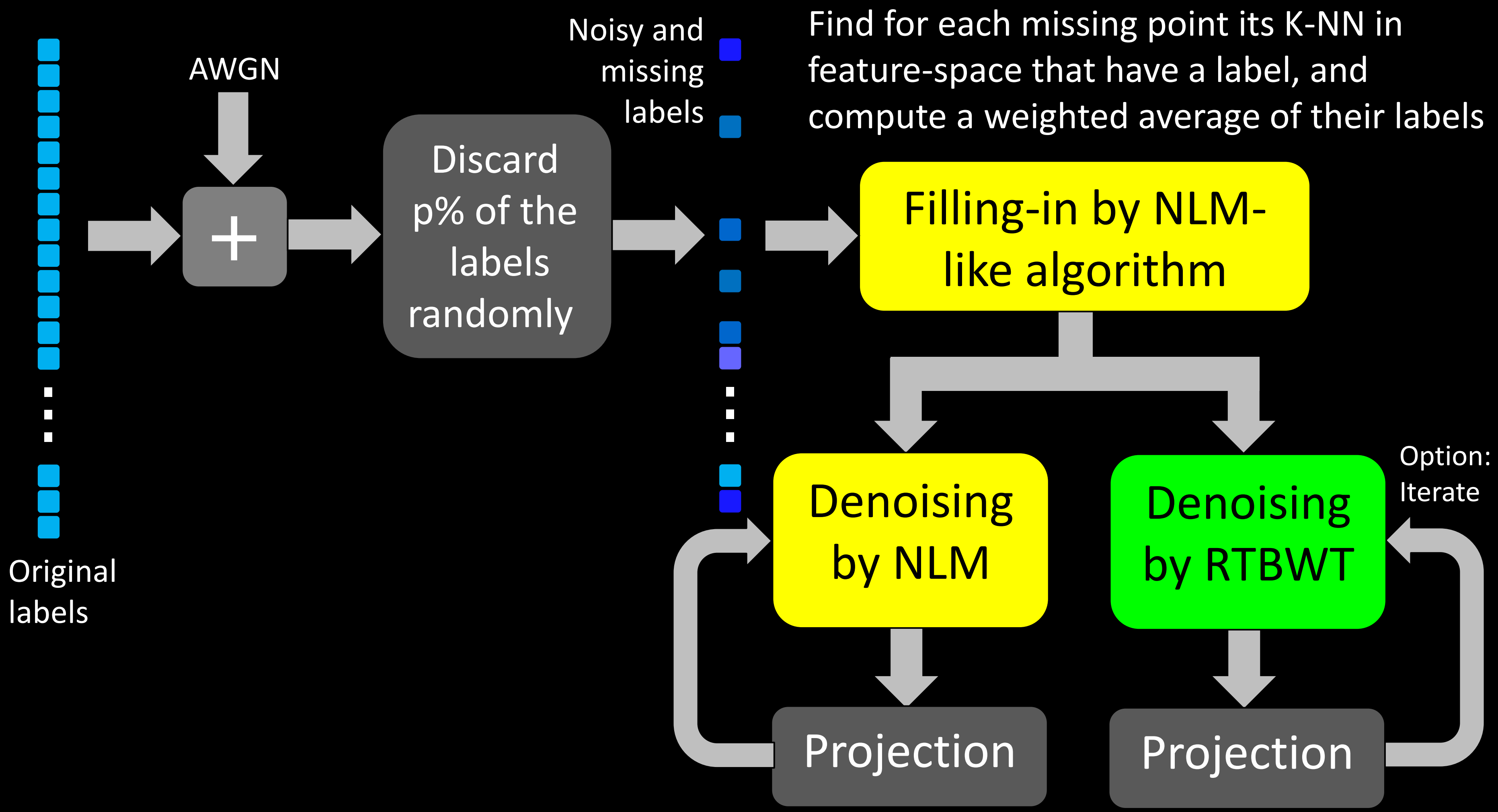
Find for each point its K-NN in feature-space, and compute a weighted average of their labels

Apply the RTBWT transform to the point-cloud labels, threshold the values and transform back

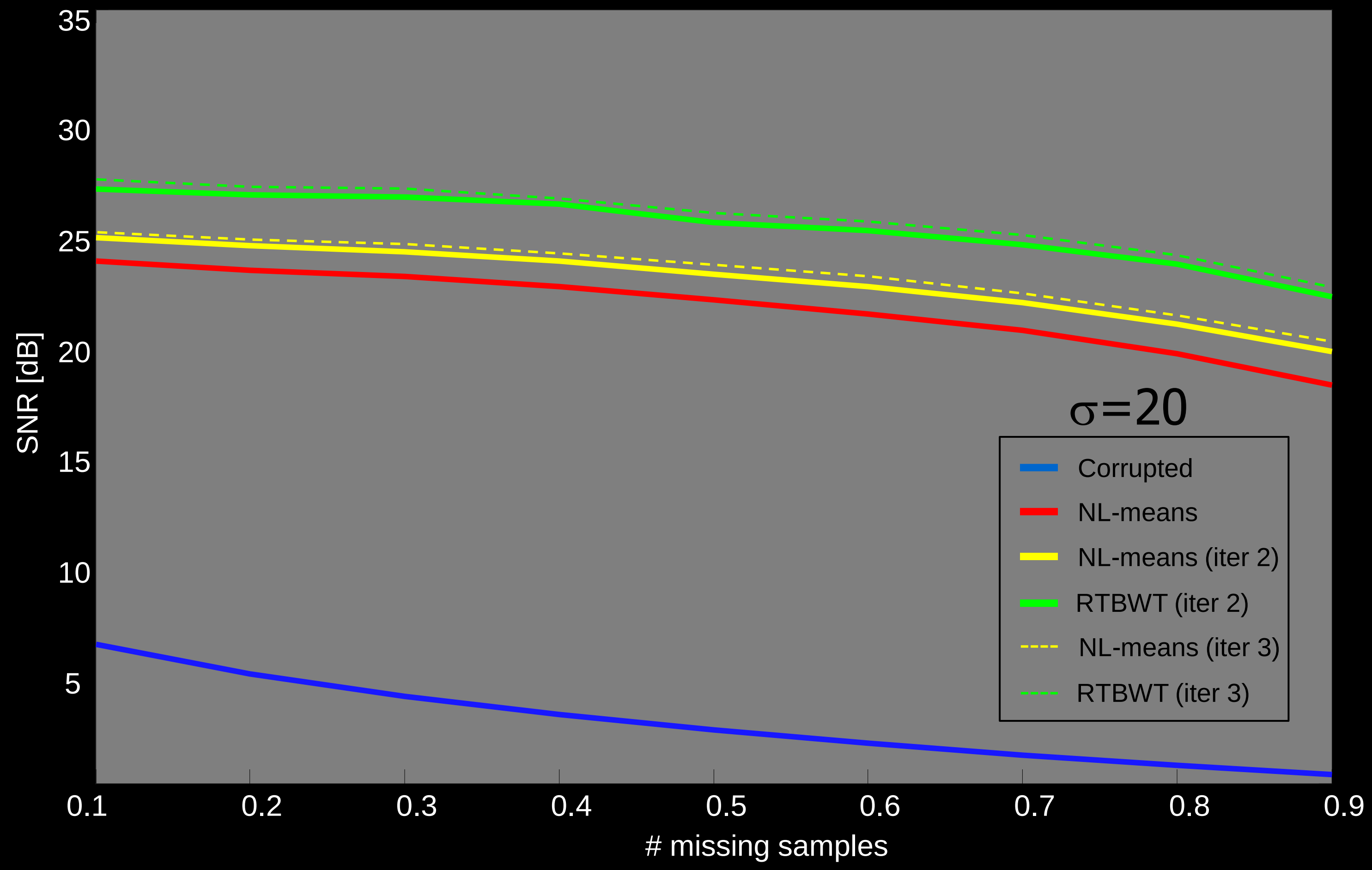
Treating Graphs: Denoising



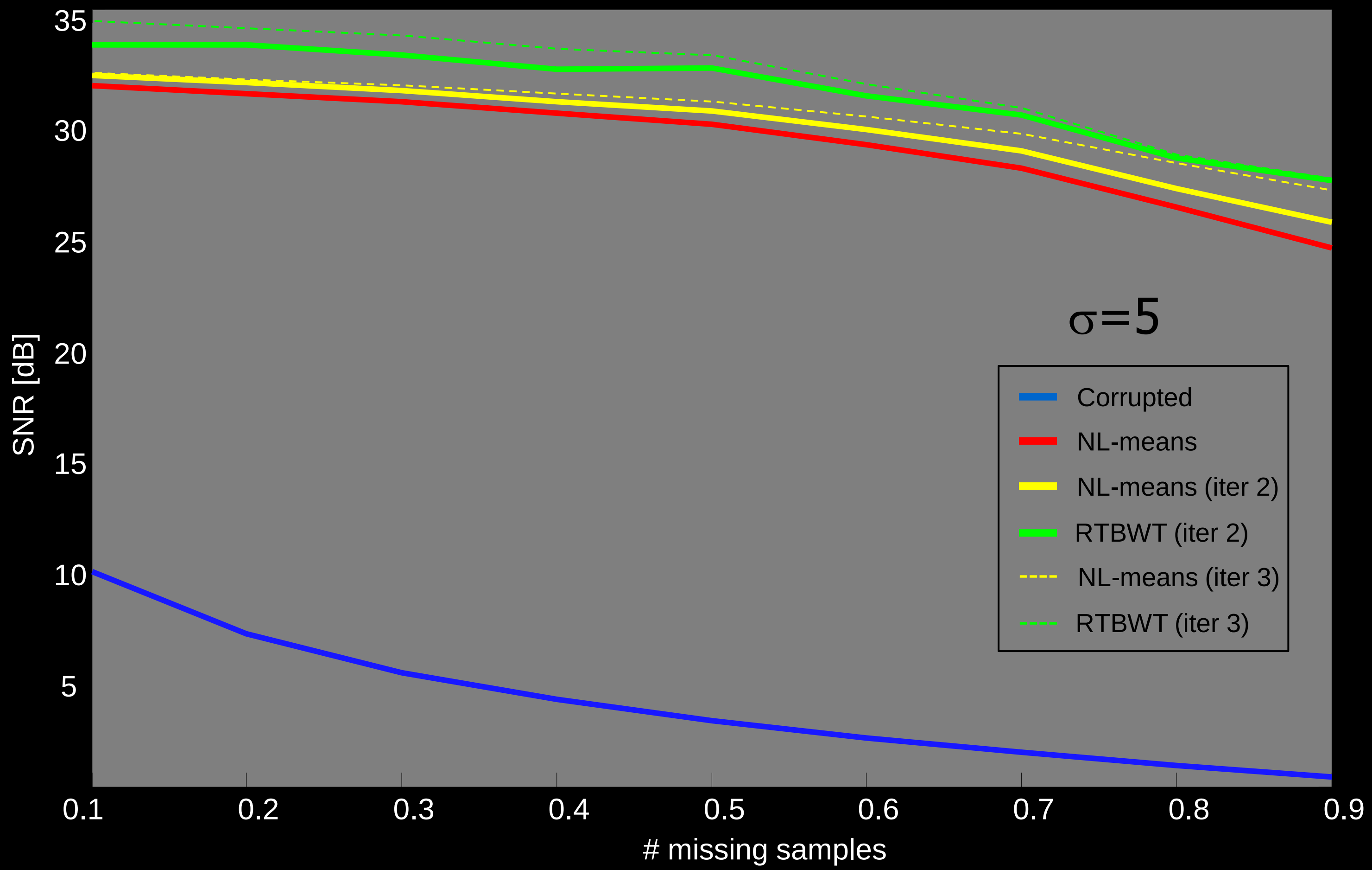
Treating Graphs: Semi-Supervised Learning



Treating Graphs: Semi-Supervised Learning



Treating Graphs: Semi-Supervised Learning



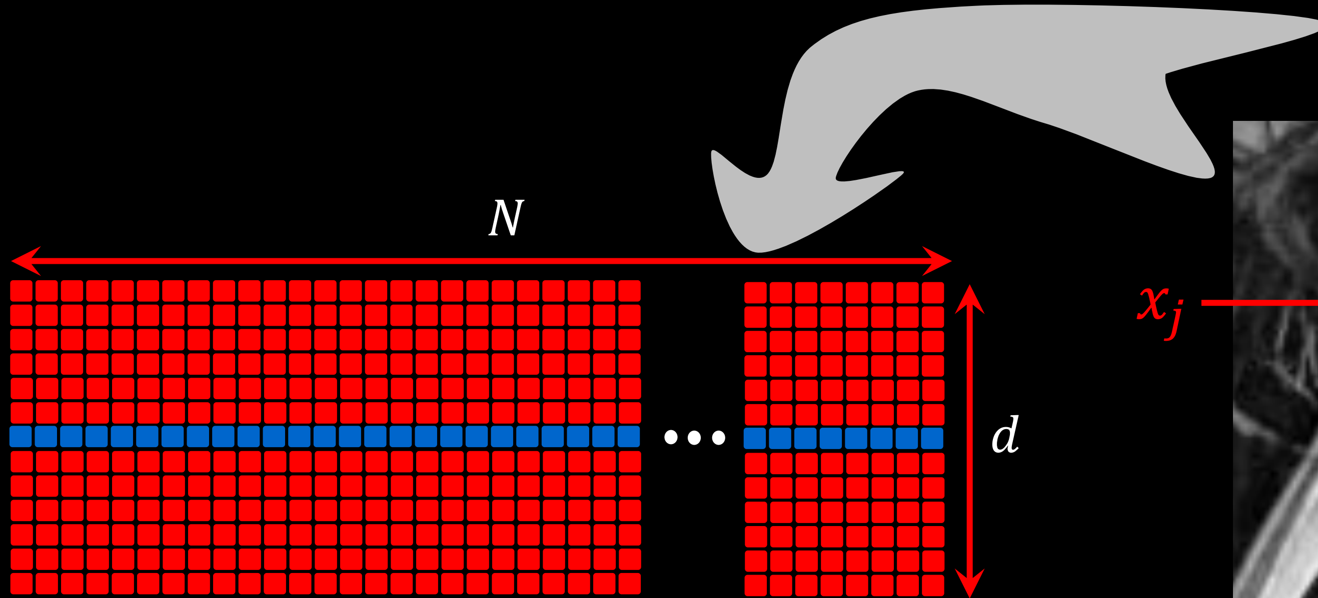
Part II – Handling Images

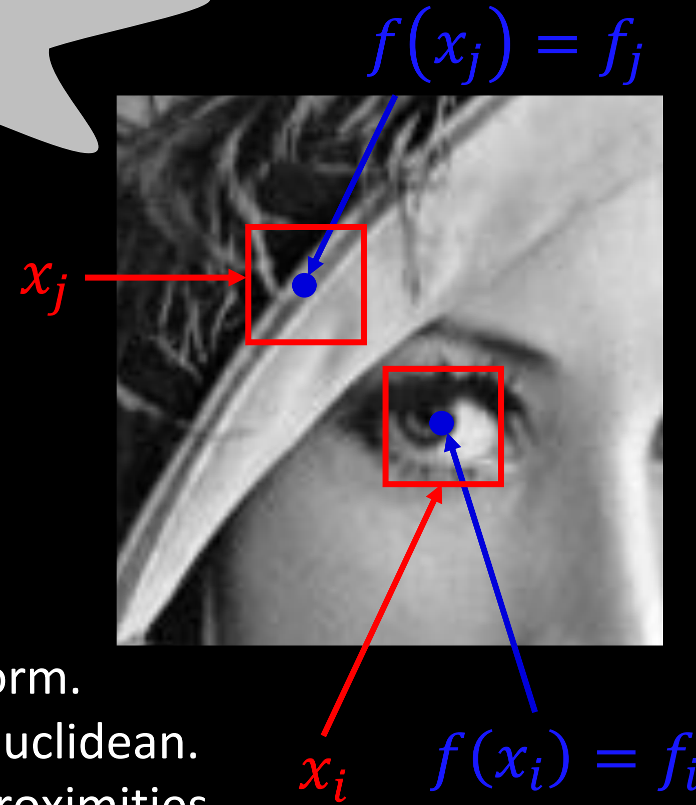
Using GTBWT for Handling Images

This part is taken from the same papers mentioned before …

- 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.

Turning an Image into a Graph?





- Now, that the image is organized as a graph (or point- cloud), we can apply the developed transform.
- The distance measure w(•, •) we will be using is Euclidean.
- After this "conversion", we forget about spatial proximities.
- The overall scheme becomes "yet another" patch-based image processing algorithm …

Patches … Patches … Patches …

In the past decade we see more and more researchers suggesting to process a signal or an image by operating on its patches.

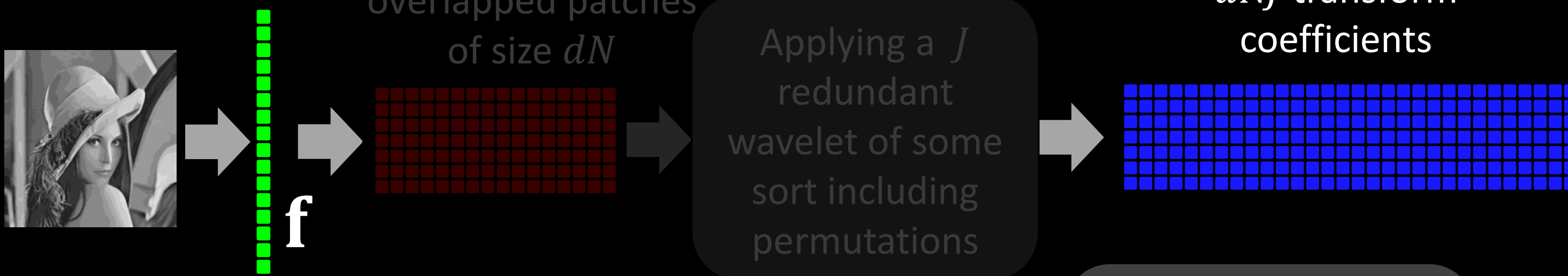
Various Ideas:

- 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
- …

Our Transform

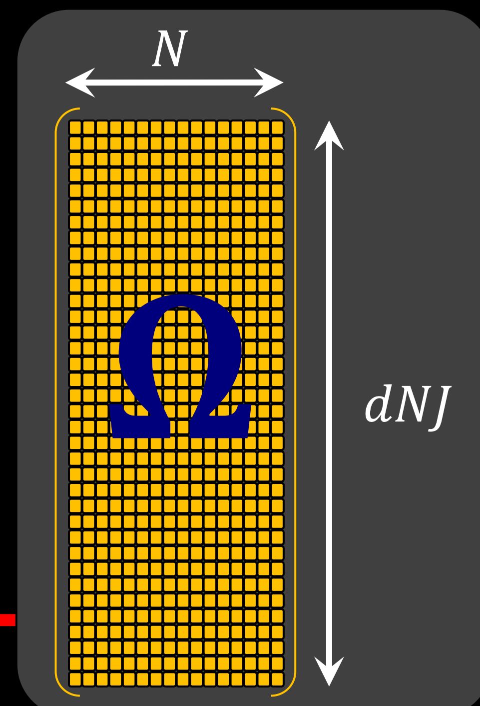
$\mathbf{X}$: Array of overlapped patches of size dN

We obtain an array of dNJ transform coefficients

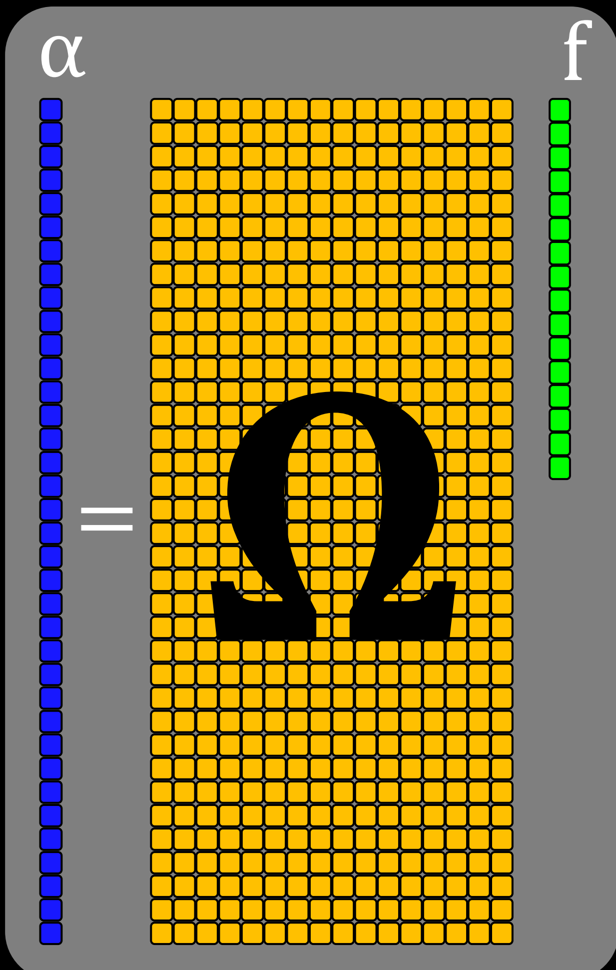


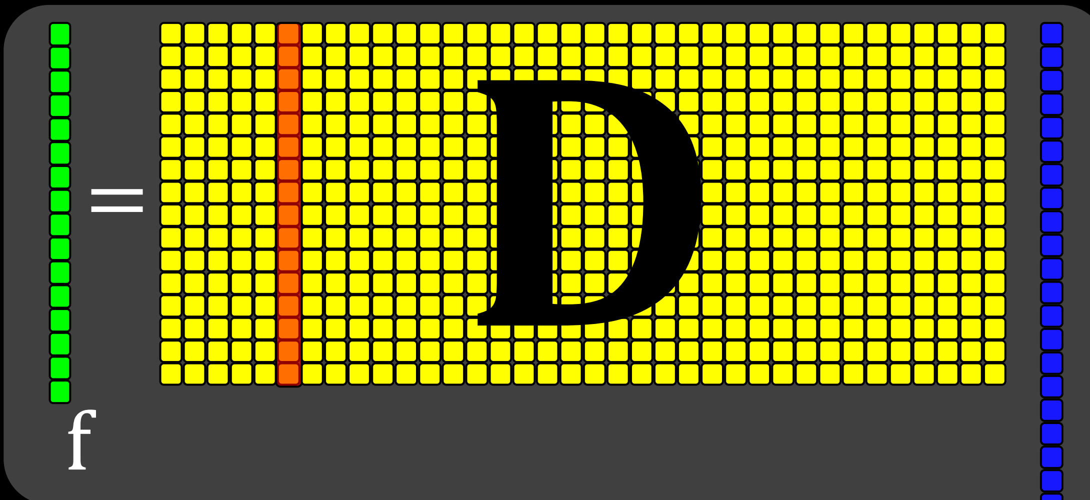
Lexicographic ordering of the N pixels

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



The Representation's Atoms





$$\mathbf{I} = \mathbf{D}\mathbf{\Omega}$$

(Not a Moore-Penrose pair)

Every column in **D** is an

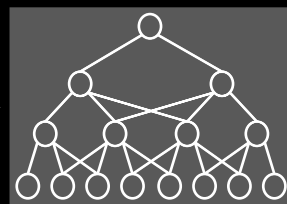
atom

Lets Test It: M-Term Approximation

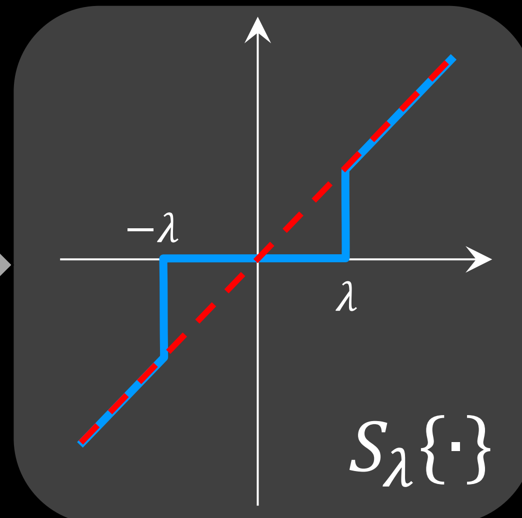
Original Image

$\mathbf{f}$

Multiply by $\boldsymbol{\Omega}$:
Forward GTBWT



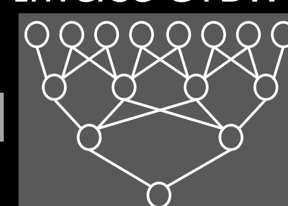
$\boldsymbol{\Omega}\mathbf{f}$



$S_\lambda\{\boldsymbol{\Omega}\mathbf{f}\}$

M non-zeros

Multiply by $\mathbf{D}$:
Inverse GTBWT



Output image

$\hat{\mathbf{f}}$

Show

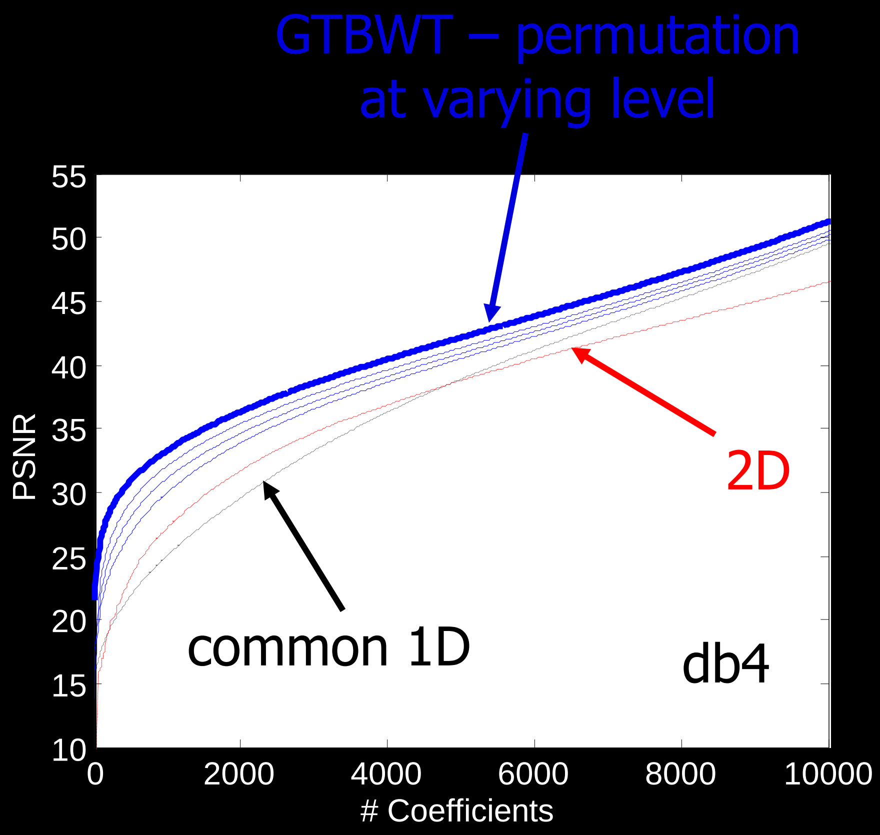
$$\|\mathbf{f}-\hat{\mathbf{f}}\|^2 = \|\mathbf{f}-\mathbf{D}S_\lambda\{\boldsymbol{\Omega}\mathbf{f}\}\|^2$$

as a function of M

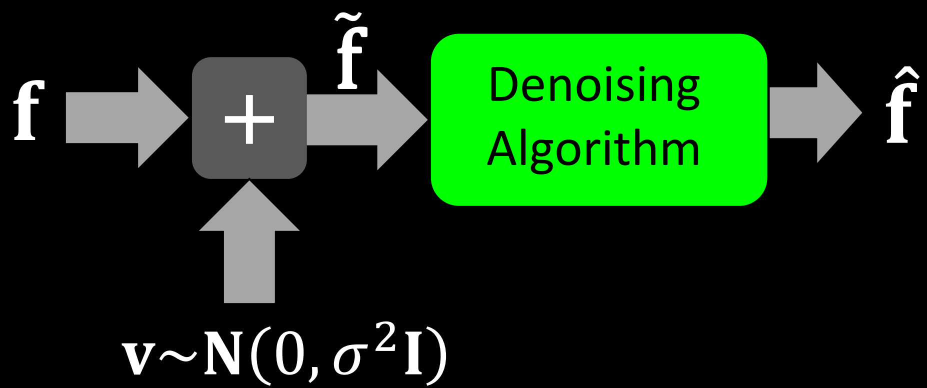
Lets Test It: M-Term Approximation

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



Lets Test It: Image Denoising



Approximation by the THR algorithm:

$$\hat{\mathbf{f}} = \mathbf{D} S_\lambda \{ \mathbf{\Omega} \tilde{\mathbf{f}} \}$$

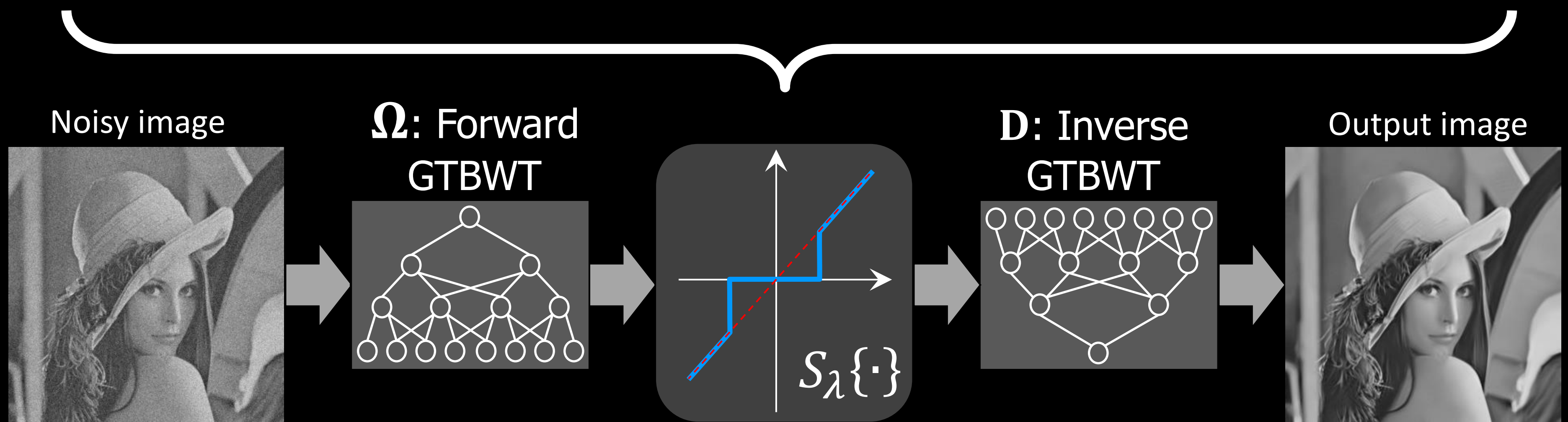


Image Denoising – Improvements

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

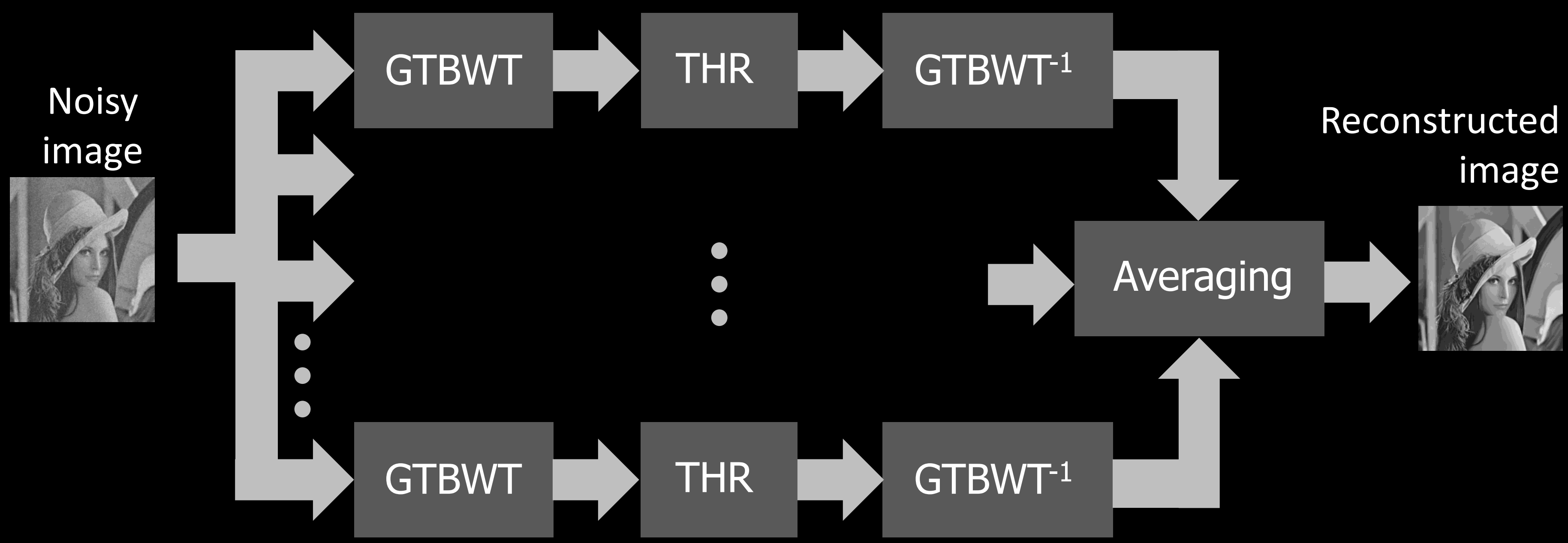


Image Denoising – Improvements

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.

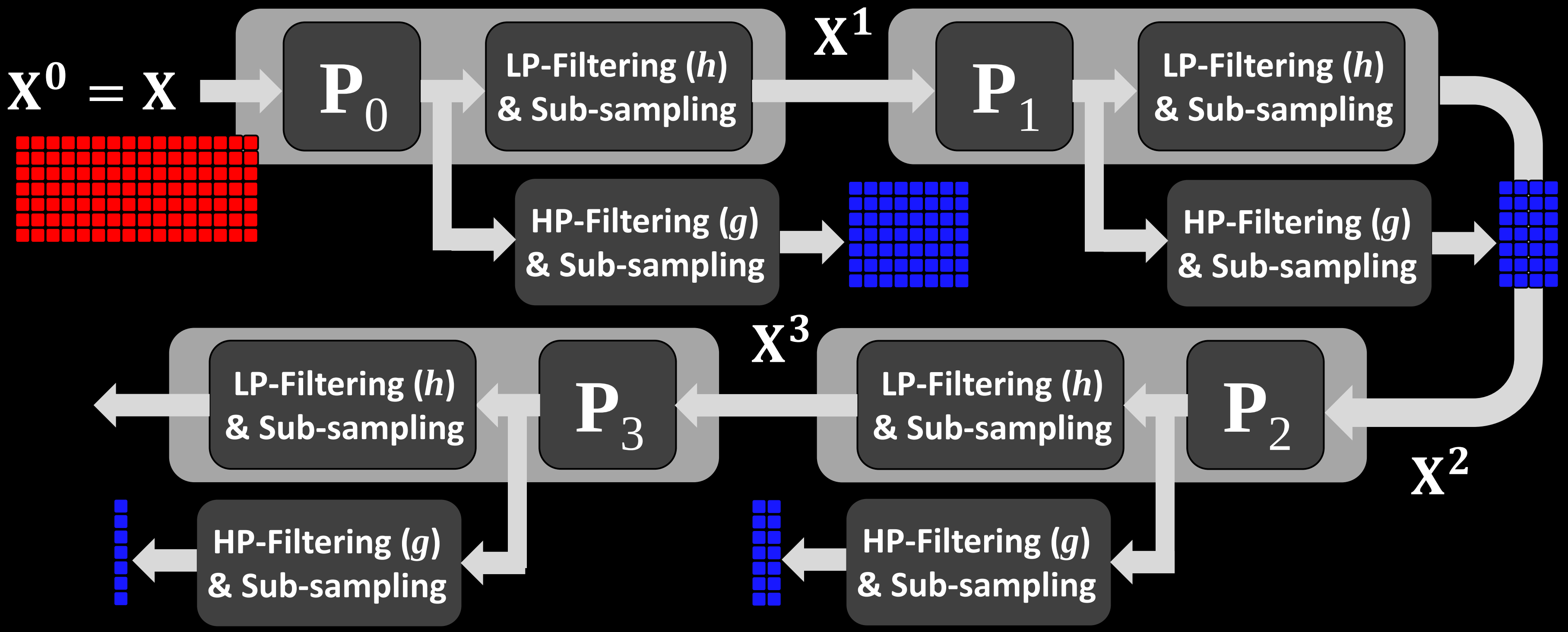
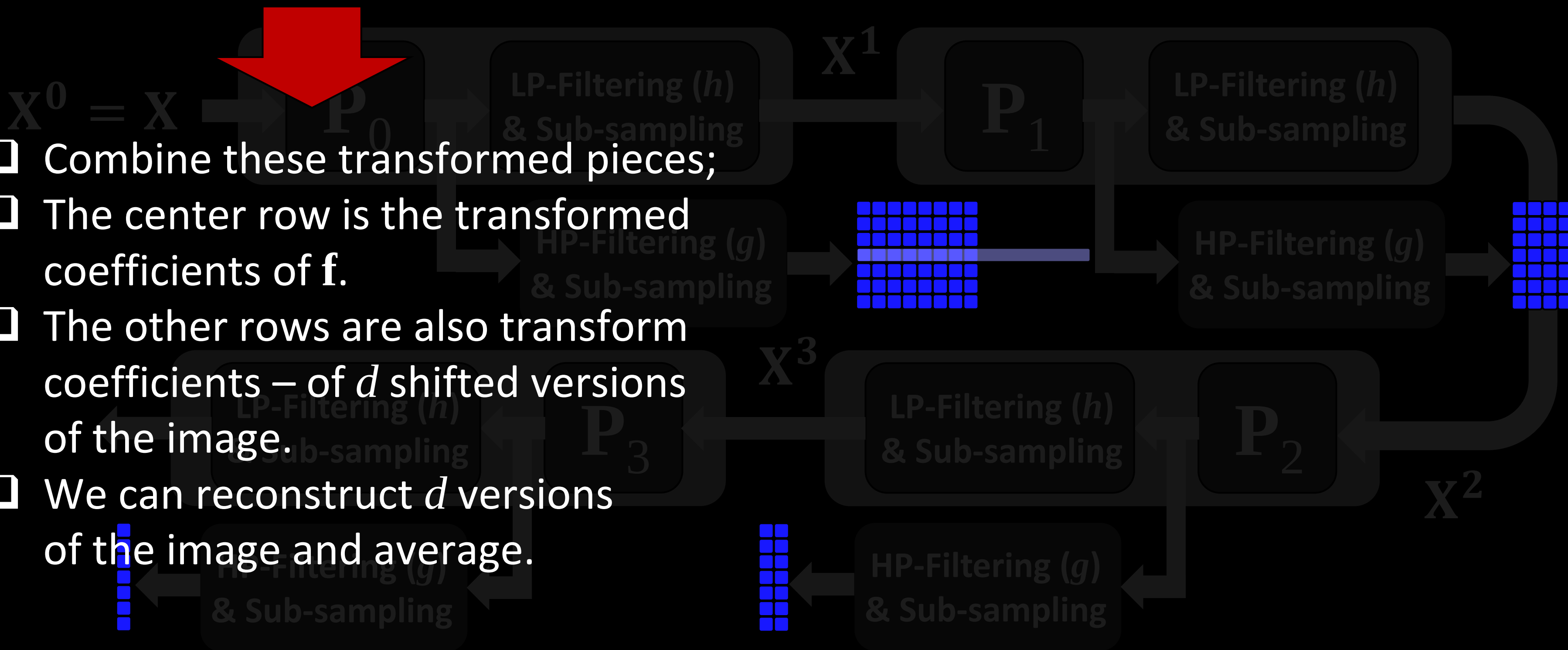


Image Denoising – Improvements

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 d shifted versions of the image.
- We can reconstruct d 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.

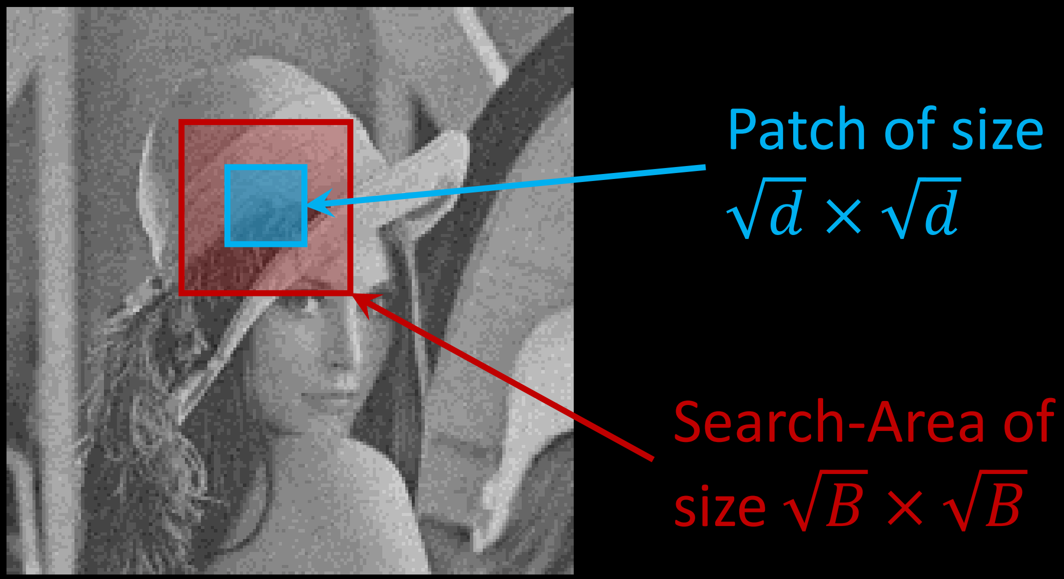
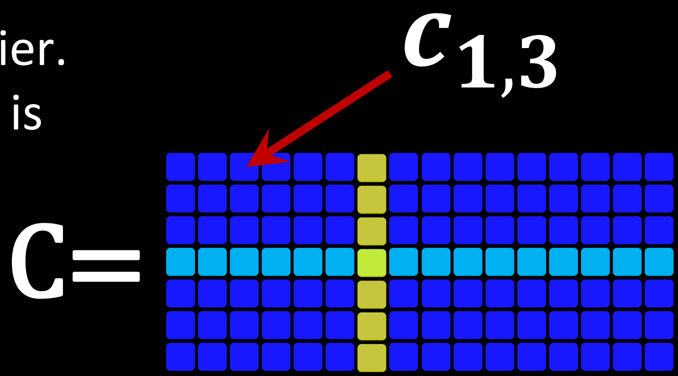


Image Denoising – Improvements

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:

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



- 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.

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 Next?

SKIP?

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

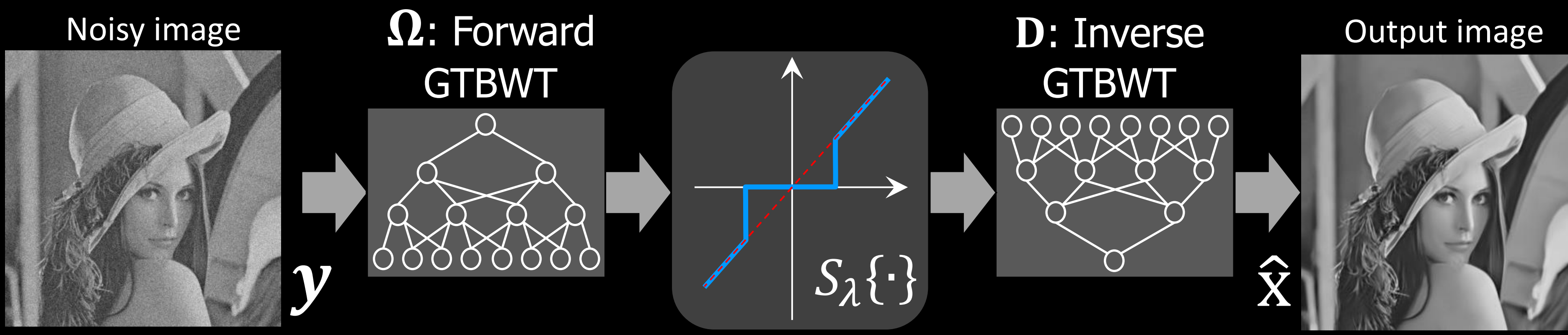
This part is documented in the following draft :

- I. Ram, M. Elad, and I. Cohen, "The RTBWT Frame – Theory and Use for Images", to appear in IEEE Trans. on Image Processing.

We rely heavily on :

- Danielyan, Katkovnik, and Eigiazarian, "BM3D frames and Variational Image Deblurring", IEEE Trans. on Image Processing, Vol. 21, No. 4, pp. 1715-1728, April 2012.

Recall Our Core Scheme



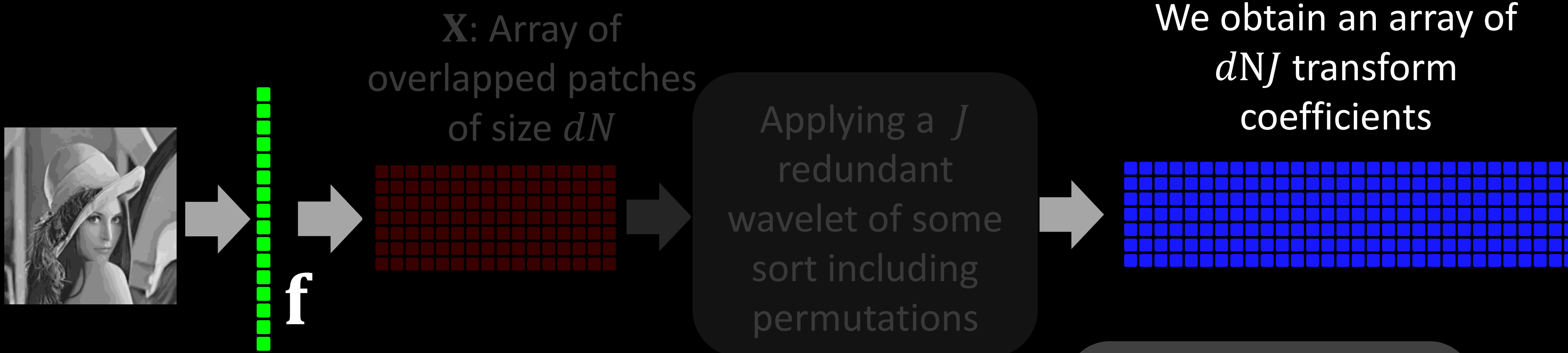
Or, put differently, $\hat{x} = \mathbf{D} \cdot T\{\mathbf{\Omega} y\}$: We refer to GTBWT as a redundant frame, and use a "heuristic" shrinkage method with it, which aims to approximate the solution of

Synthesis: $$\hat{x} = \mathbf{D} \cdot \underset{\alpha}{\text{Argmin}} \|\mathbf{D}\alpha - y\|_2^2 + \lambda\|\alpha\|_p^p$$

or

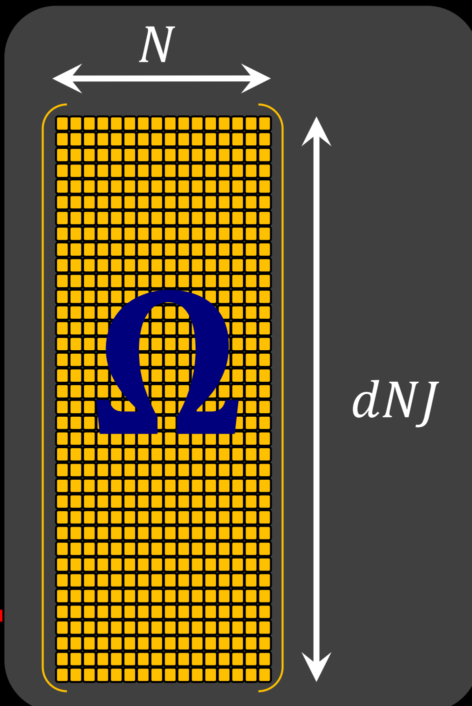
Analysis: $$\hat{x} = \underset{f}{\text{Argmin}} \|x - y\|_2^2 + \lambda\|\mathbf{\Omega} x\|_p^p$$

Recall: Our Transform (Frame)



Lexicographic ordering of the N pixels

- All these operations could be described as one linear operation: multiplication of $\underline{f}$ by a huge matrix $\mathbf{\Omega}$
- This transform is adaptive to the specific image



What Can We Do With This Frame?

We could solve various inverse problems of the form:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{v}$$

where: x is the original image
v is an AWGN, and
$\mathbf{A}$ is a degradation operator of any sort

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

$$\{\hat{\mathrm{x}}, \hat{\alpha}\} = \underset{\alpha,\mathrm{x}}{\mathrm{Argmin}} \begin{array}{c} \|\mathrm{y} - \mathbf{A}\mathrm{x}\|_2^2 + \frac{1}{\beta}\|\mathbf{D}\alpha - \mathrm{x}\|_2^2 + \\ +\lambda\|\alpha\|_p^p + \frac{1}{\mu}\|\boldsymbol{\Omega}\mathrm{x} - \alpha\|_2^2 \end{array}$$

$\beta = 0,\ \mu = \infty \rightarrow$ Synthesis

$\beta = \infty,\ \mu = 0 \rightarrow$ Analysis

Generalized Nash Equilibrium*

Instead of minimizing the joint analysis/synthesis problem:

$$\{\hat{\mathrm{x}}, \hat{\alpha}\} = \underset{\alpha,\mathrm{x}}{\mathrm{Argmin}} \|\mathrm{y} - \mathbf{A}\mathrm{x}\|_2^2 + \frac{1}{\beta}\|\mathbf{D}\alpha - \mathrm{x}\|_2^2 + +\lambda\|\alpha\|_p^p + \frac{1}{\mu}\|\mathbf{\Omega}\mathrm{x} - \alpha\|_2^2$$

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

and solve iteratively

$$\mathrm{x}_{k+1} = \underset{\mathrm{x}}{\mathrm{Argmin}} \ \|\mathrm{y} - \mathbf{A}\mathrm{x}\|_2^2 + \frac{1}{\beta}\|\mathbf{D}\alpha_k - \mathrm{x}\|_2^2$$

$$\alpha_{k+1} = \underset{\alpha}{\mathrm{Argmin}} \ \ \lambda\|\alpha\|_p^p + \frac{1}{\mu}\|\mathbf{\Omega}\mathrm{x}_{k+1} - \alpha\|_2^2$$

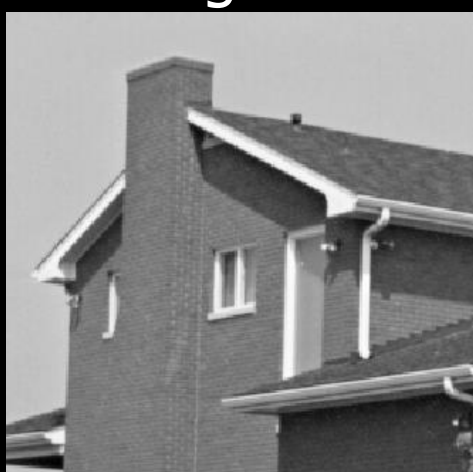
* Danielyan, Katkovnik, and Eigiazarian, “BM3D frames and Variational Image Deblurring”, IEEE Trans. on Image Processing, Vol. 21, No. 4, pp. 1715-1728, April 2012.

Deblurring Results


Original


Blurred+Noisy


Restored



Deblurring Results

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

$$\text{Blur PSF} = \frac{1}{1+i^2+j^2} \quad -7 \le i,j \le 7$$

$\sigma^2=2$

Part IV – Patch (Re)-Ordering

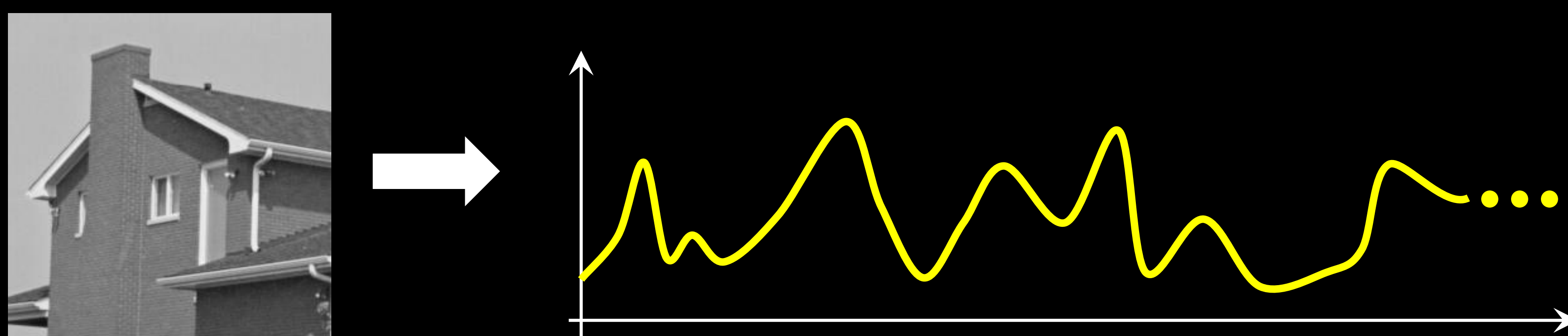
Lets Simplify Things, Shall We?

This part is based on the papers:

- I. Ram, M. Elad, and I. Cohen, "Image Processing using Smooth Ordering of its Patches", IEEE Transactions on Image Processing, Vol. 22, No. 7, pp. 2764–2774 , July 2013.
- I. Ram, I. Cohen, and M. Elad, "Facial Image Compression using Patch-Ordering-Based Adaptive Wavelet Transform", Submitted to IEEE Signal Processing Letters.

2D → 1D Conversion ?

Often times, when facing an image processing task (denoising, compression, …), the proposed solution starts by a 2D to 1D conversion :

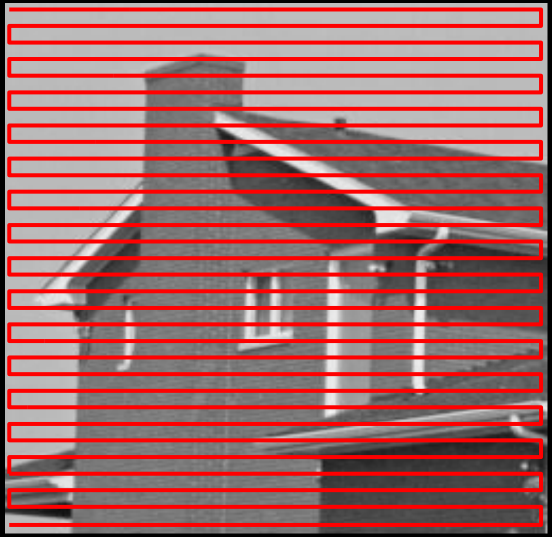


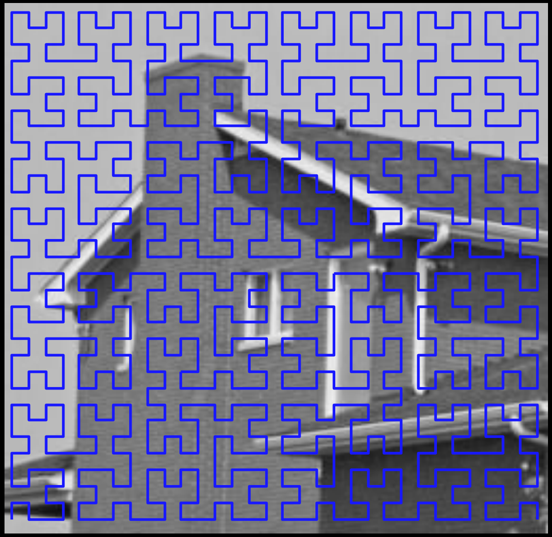
After such a conversion, the image is treated as a regular 1D signal, with implied sampled order and causality.

2D ⟶ 1D : How to Convert ?

- There are many ways to convert an image into a 1D signal. Two very common methods are:

Raster Scan





Hilbert-Peano Scan

- Note that both are “space-filling curves” and image-independent, but we need not restrict ourselves to these types of 2D ⟶1D conversions.

2D $\longrightarrow$ 1D : Why Convert ?

The scientific literature on image processing is loaded with such conversions, and the reasons are many:

- ❑ Because serializing the signal helps later treatment.
- ❑ Because (imposed) causality can simplify things.
- ❑ Because this enables us to borrow ideas from 1D signal processing (e.g. Kalman filter, recursive filters, adaptive filters, prediction, …).
- ❑ Because of memory and run-time considerations.

❑ Common belief: 2D → 1D conversion leads to a

S U B O P T I M A L   S O L U T I O N ! !

because of loss of neighborhood relations and forced causality.

2D ⟶ 1D : Why Convert ?

The scientific literature on image processing is loaded with such conversions, and the reasons are many:

- ❑ Because serializing the signal helps later treatment.
- ❑ Beca
- ❑ Beca ... ıg (e.g. Kalm

ARE WE SURE ?

- ❑ Because of memory and run-time considerations.

❑ Common belief: 2D → 1D conversion leads to a

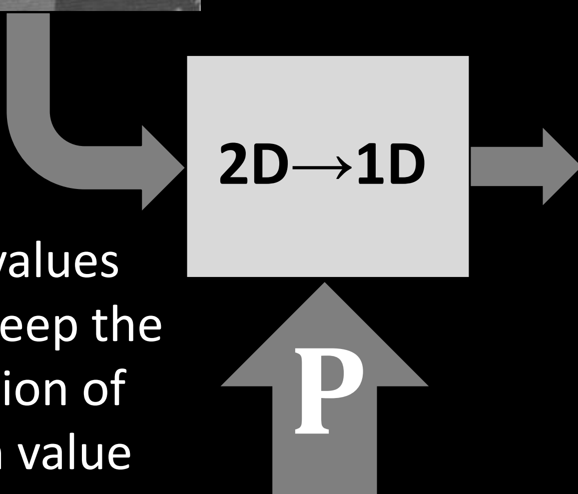
S U B O P T I M A L   S O L U T I O N ! !

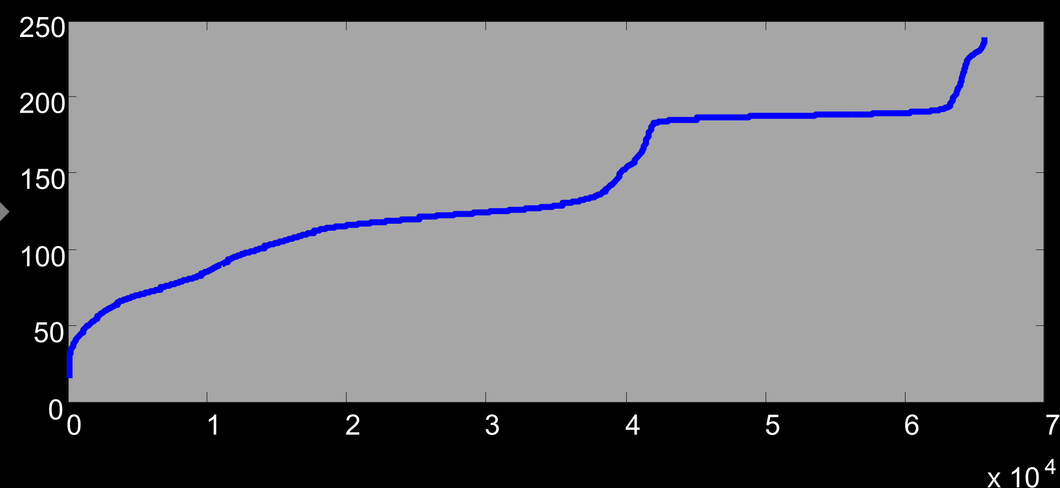
because of loss of neighborhood relations and forced causality.

Lets Propose a New 2D → 1D Conversion

How about permuting the pixels into a 1D signal by a

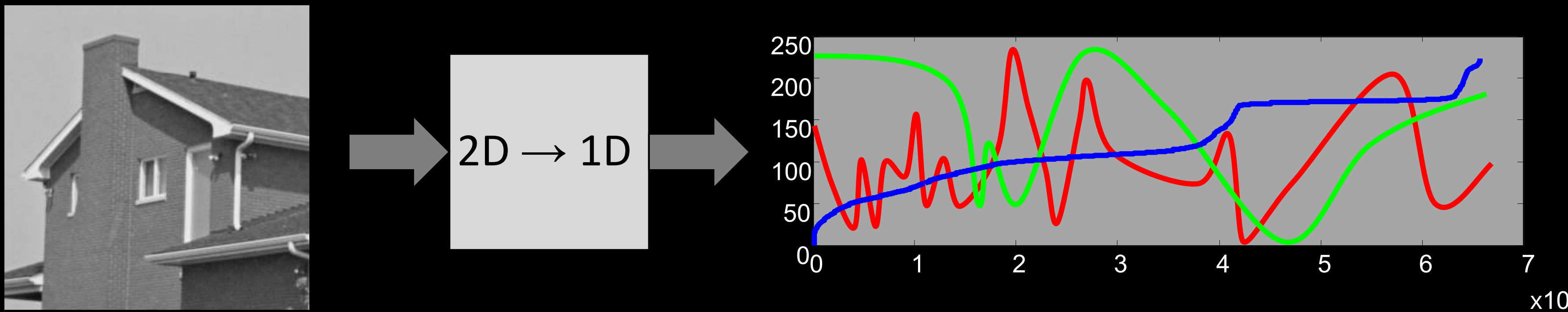
SORT OPERATION ?





We sort the gray-values but also keep the [x,y] location of each such value

New 2D → 1D Conversion : Smoothness



- Given any 2D → 1D conversion based on a permutation **P**, we may ask how smooth is the resulting 1D signal obtained :

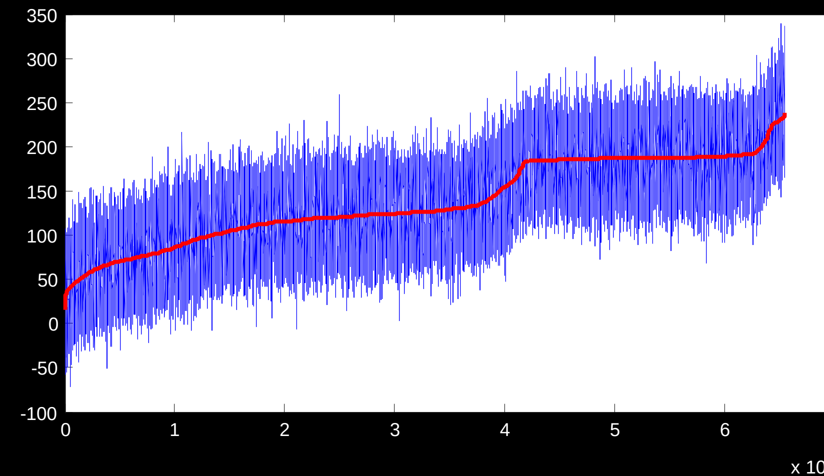
$$\mathrm{TV}\{\mathrm{f},\mathbf{P}\} = \sum_{k=2}^{N} \left| f_P(k) - f_P(k-1) \right|$$

- The sort-ordering leads to the smallest possible TV measure, i.e. it is the smoothest possible.
- Who cares? We all do, as we will see hereafter.

New 2D → 1D Conversion : An Example

g

2D→1D



P

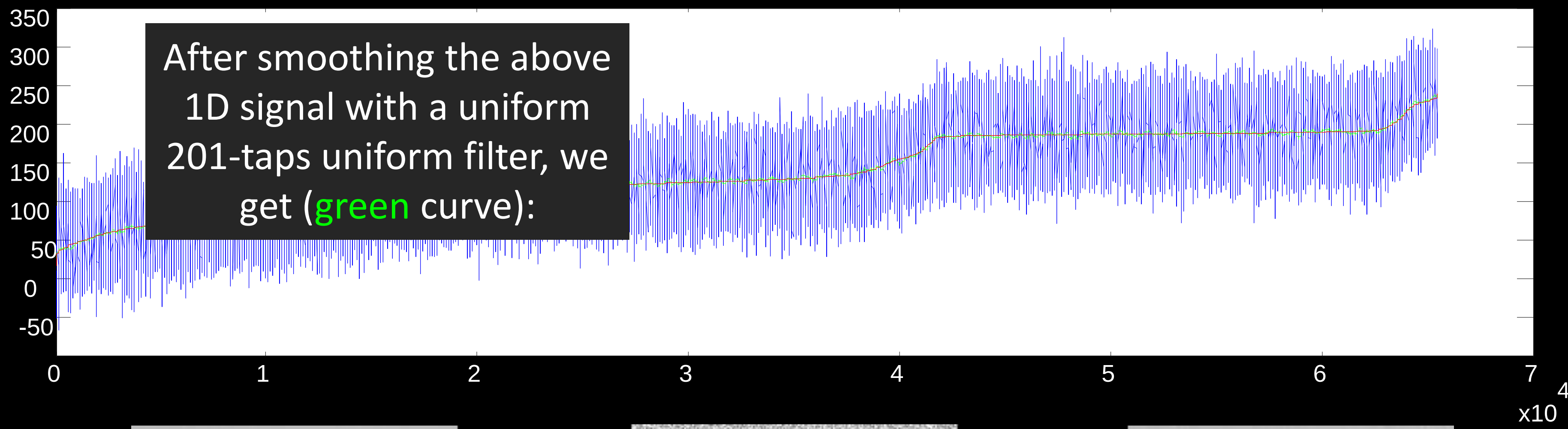
f

Find the Sort Permutation

This means that simple smoothing of the 1D signal is likely to lead to a very good denoising

New 2D → 1D Conversion : An Example

After smoothing the above 1D signal with a uniform 201-taps uniform filter, we get (green curve):



f

Original

g

Noisy σ=30 (18.58dB)

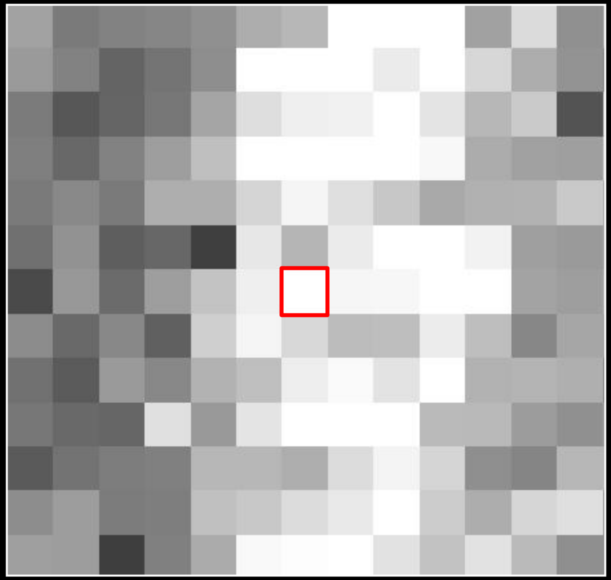
$\hat{f}$

Denoised (41.7dB)

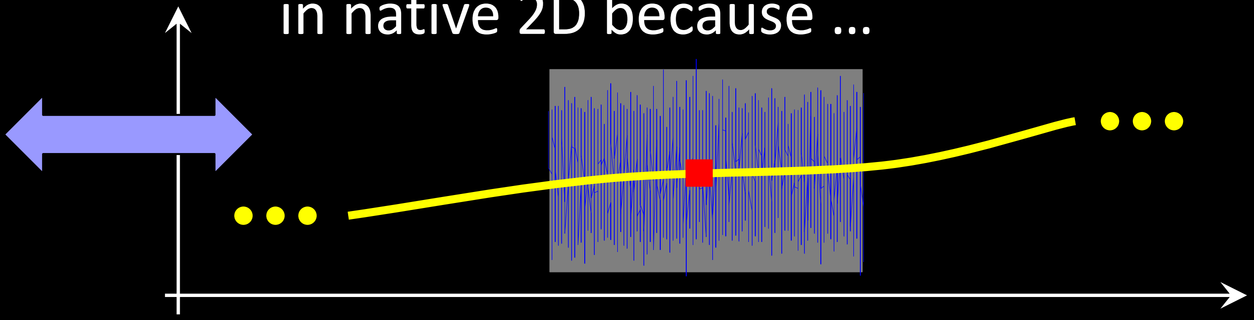
This is Just Great! Isn't It?

This denoising result we just got is nothing short of amazing, and it is far better than any known method

Is it real? Is it fair?



Neighborhood wise, note that this result is even better than treating the image in native 2D because ...



This is Just Great! Isn't It?

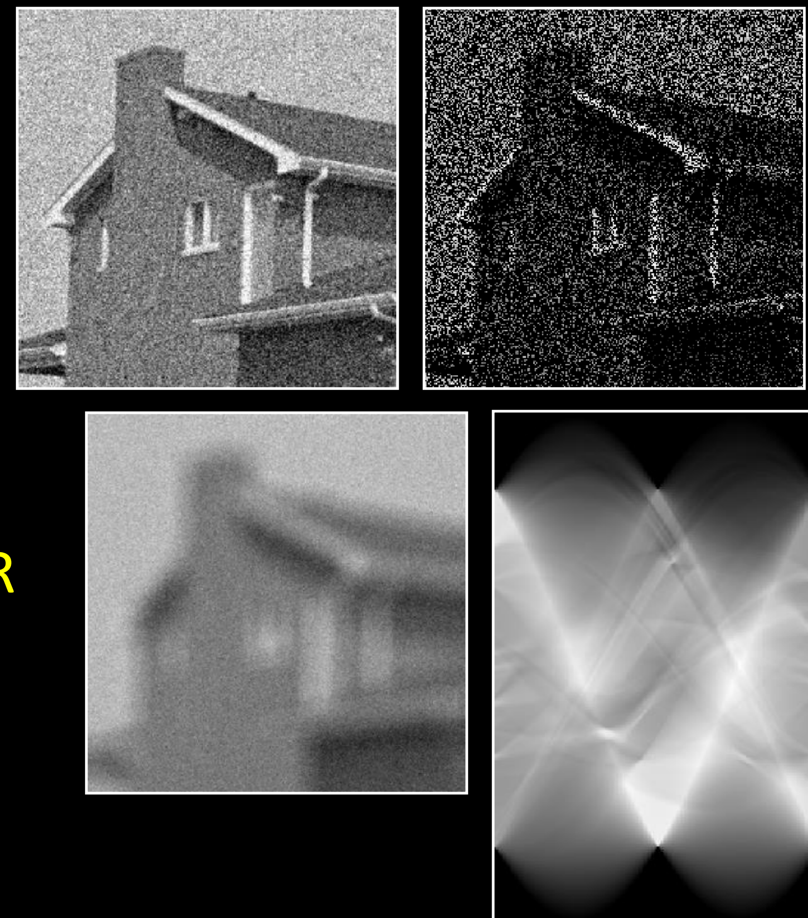
All this is wonderful … but …

Given a corrupted image (noisy, blurred, missing pixels, …)

WE CANNOT KNOW THE SORTING PERMUTATION OPERATOR

P

So the above result is impractical.



This is Just Great! Isn't It?

All this is wonderful … but …

Given a corrupted image (noi y, blurred, mi g r

So, Are We Stuck ?

So the above result is impractical.

We Need an Alternative for Constructing $\mathbf{P}$

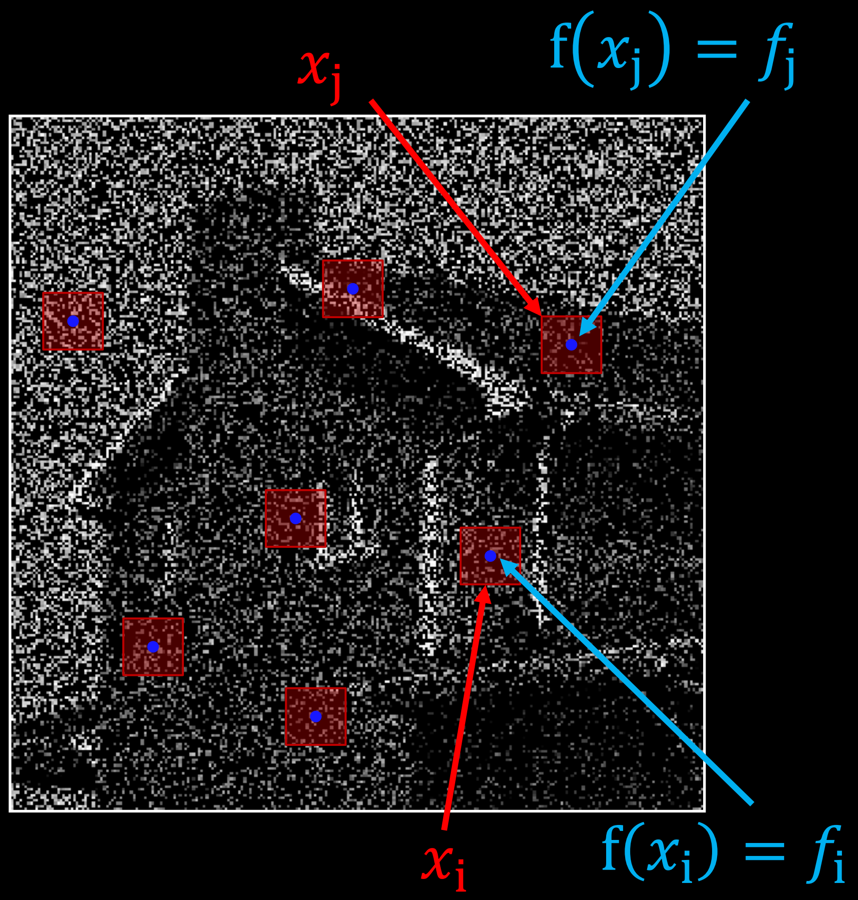
Our Goal – Sorting the pixels based on their TRUE gray value

The problem – the given data is corrupted and thus pixel gray-values are not to be trusted

The idea: Assign a feature vector x to each pixel, to enrich its description

Our approach: Every pixel will be "represented" by the patch around it

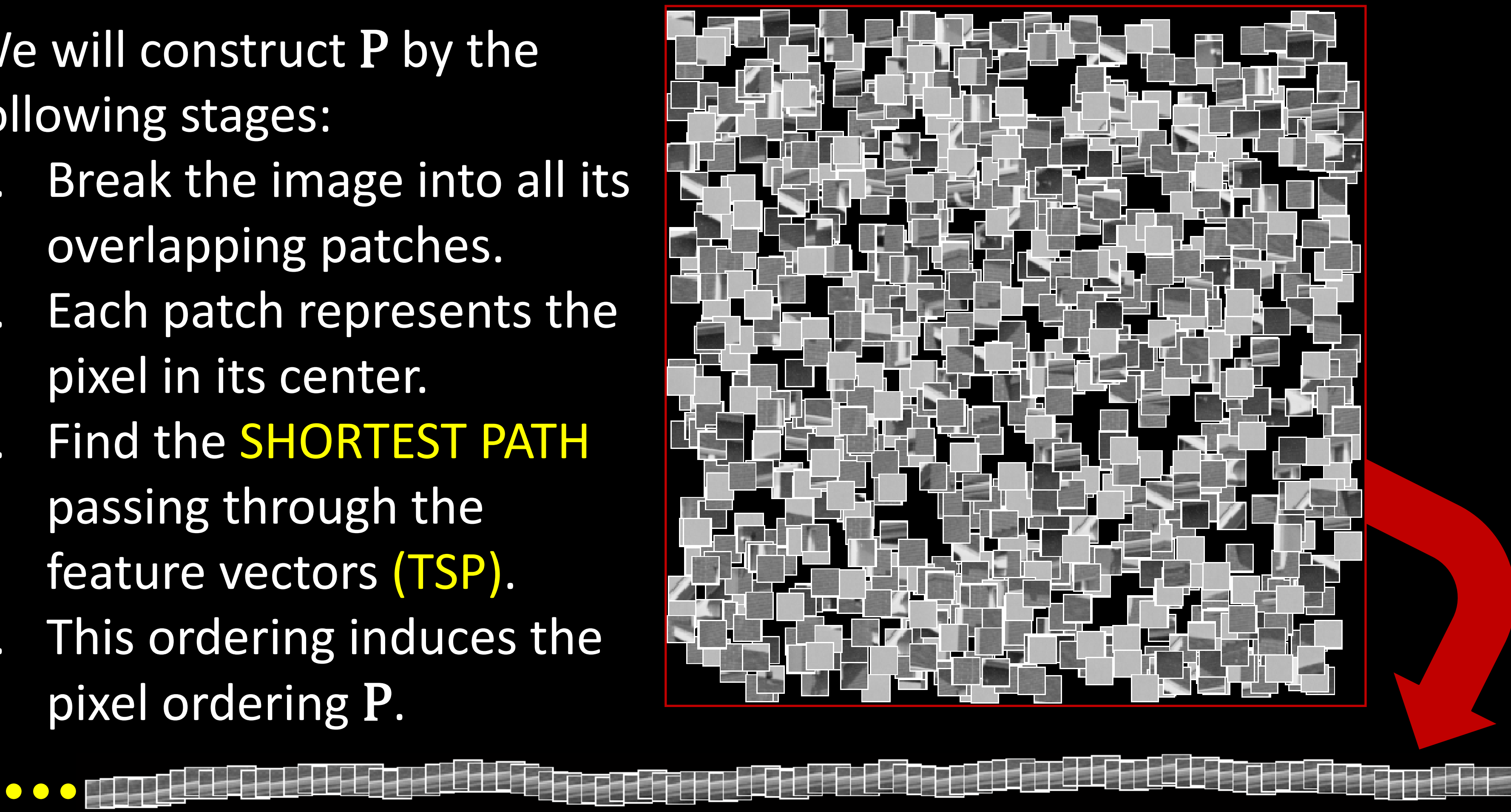
We will design $\mathbf{P}$ based on these feature vectors



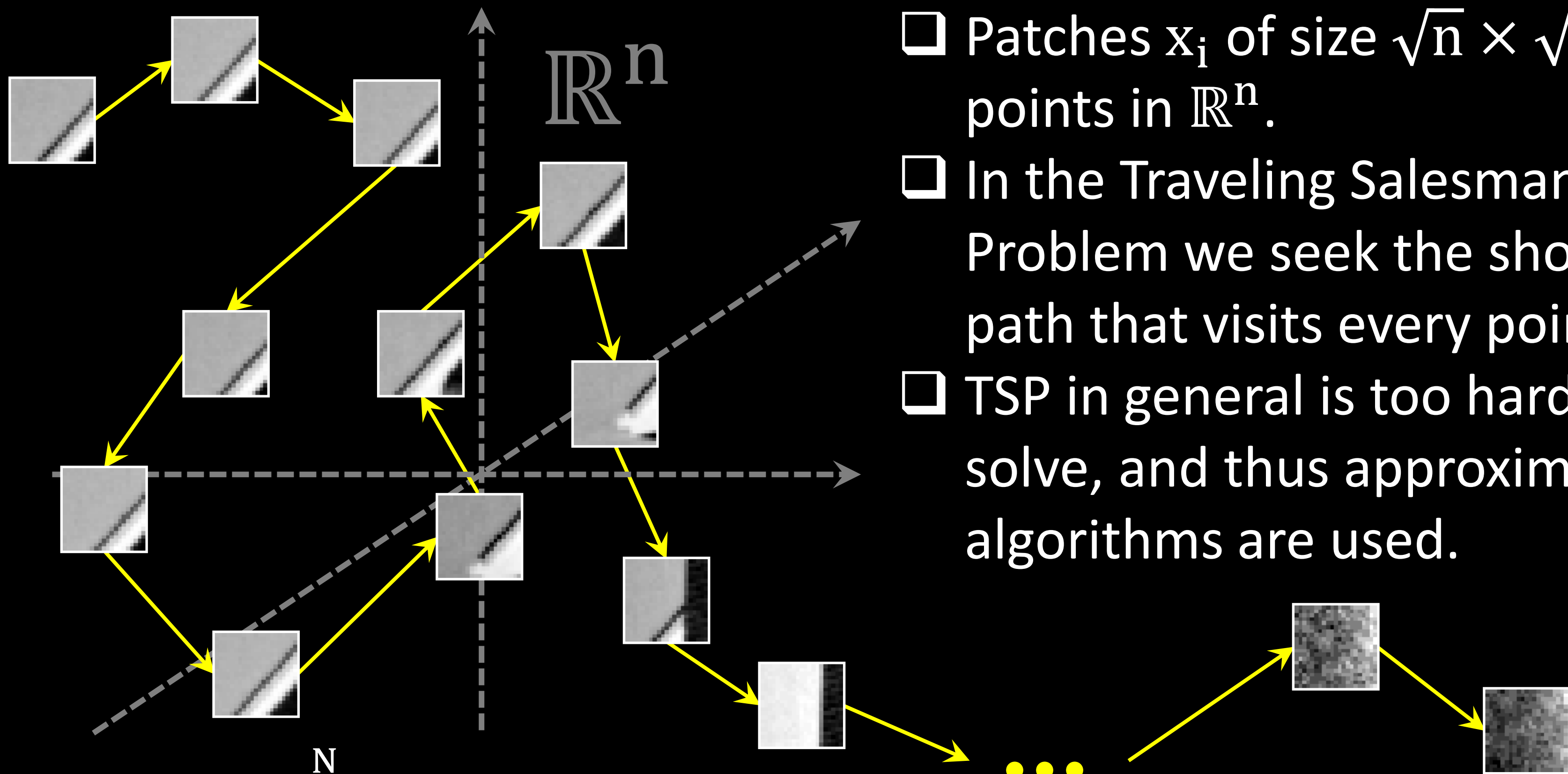
An Alternative for Constructing $\mathbf{P}$

We will construct $\mathbf{P}$ by the following stages:

1. Break the image into all its overlapping patches.
2. Each patch represents the pixel in its center.
3. Find the SHORTEST PATH passing through the feature vectors (TSP).
4. This ordering induces the pixel ordering $\mathbf{P}$.



Traveling Salesman Problem (TSP)



$$\min_{P} \sum_{i=2}^{N} \left\| x_i^p - x_{i-1}^p \right\|$$

- Patches x_i of size $\sqrt{n} \times \sqrt{n}$ are points in $\mathbb{R}^n$.
- In the Traveling Salesman Problem we seek the shortest path that visits every point.
- TSP in general is too hard to solve, and thus approximation algorithms are used.

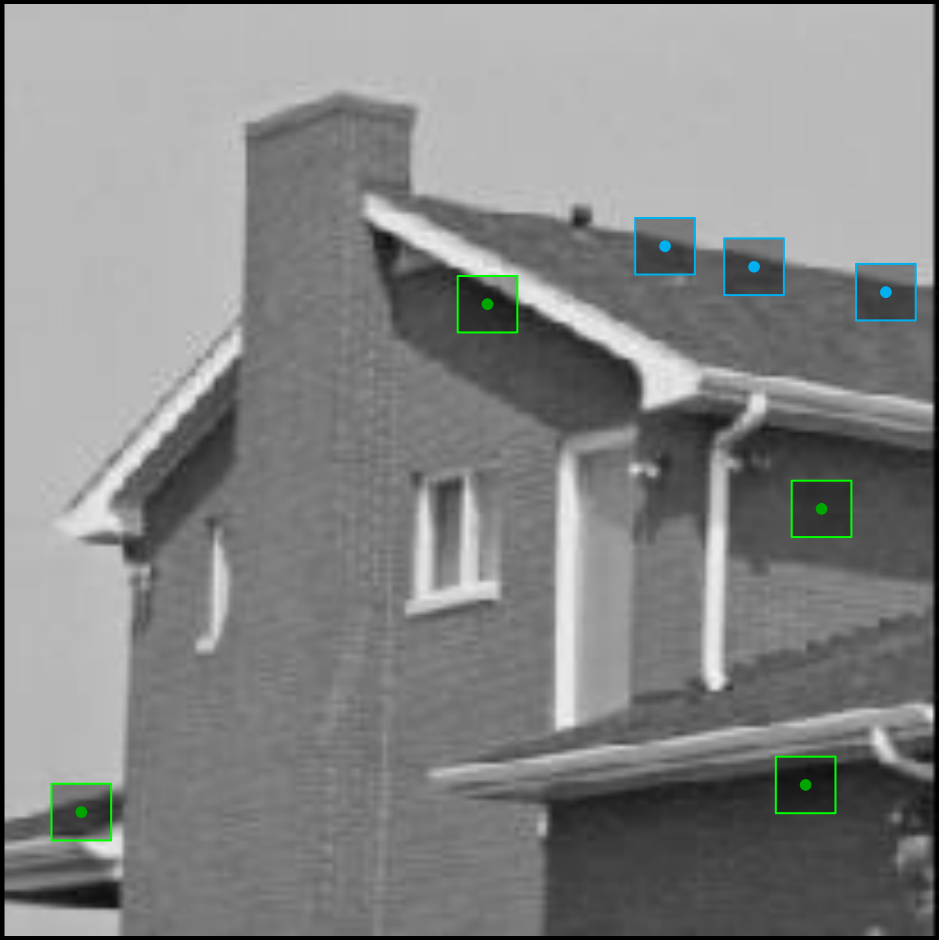
The Proposed Alternative : A Closer Look

Observation 1: Do we Get **P** ?

If two pixels have the same (or close) gray value, this does not mean that their patches are alike.

However …

If several patches are alike, their corresponding centers are likely to be close-by in gray-value



Thus, the proposed ordering will not reproduce the **P**, but at least get close to it, preserving some of the order.

The Proposed Alternative : A Closer Look

Observation 2: "Shortest-Path" ?

- In the shortest-path (and TSP), the path visits every point once, which aligns with our desire to permute the pixels and never replicate them.
- If the patch-size is reduced to 1×1 pixels, and the process is applied on the original (true) image, the obtained ordering is exactly $\mathbf{P}$.

TSP Greedy 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 Ω holds the proposed ordering.

$$\min_{\mathbf{P}} \sum_{k=2}^{N} |f_P(k) - f_P(k-1)| \Longleftrightarrow \min_{P} \sum_{i=2}^{N} \|x_i^p - x_{i-1}^p\|$$

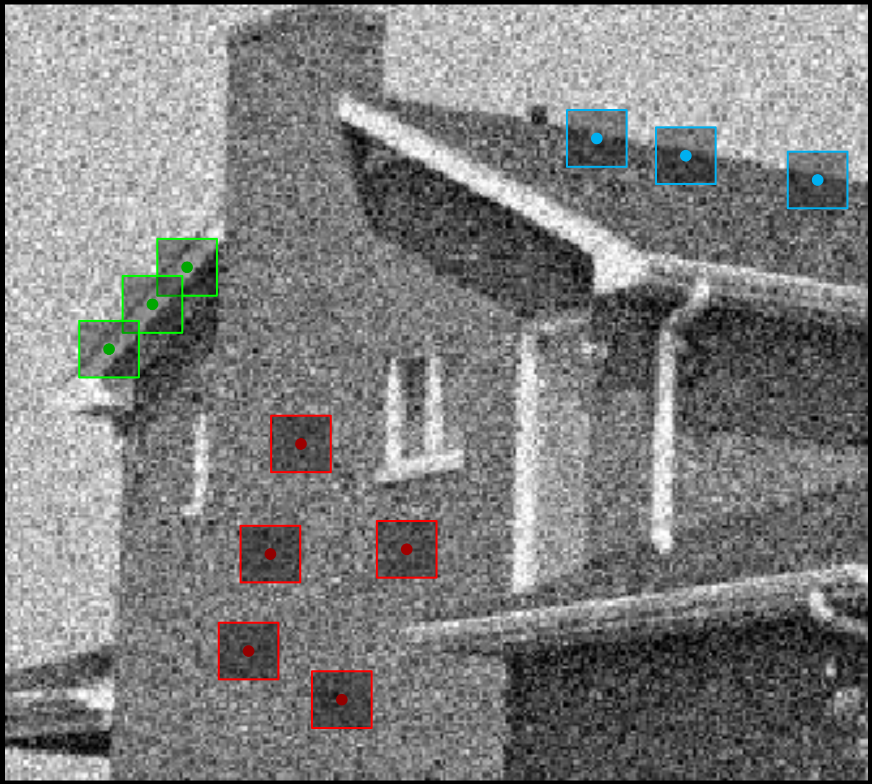
The Proposed Alternative : A Closer Look

Observation 3: Corrupted Data ?

- If we stick to patches of size 1×1 pixels, we will simply sort the pixels in the degraded image – this is not good nor informative for anything.
- The chosen approach has a robustness w.r.t. the degradation, as we rely on patches instead of individual pixels.

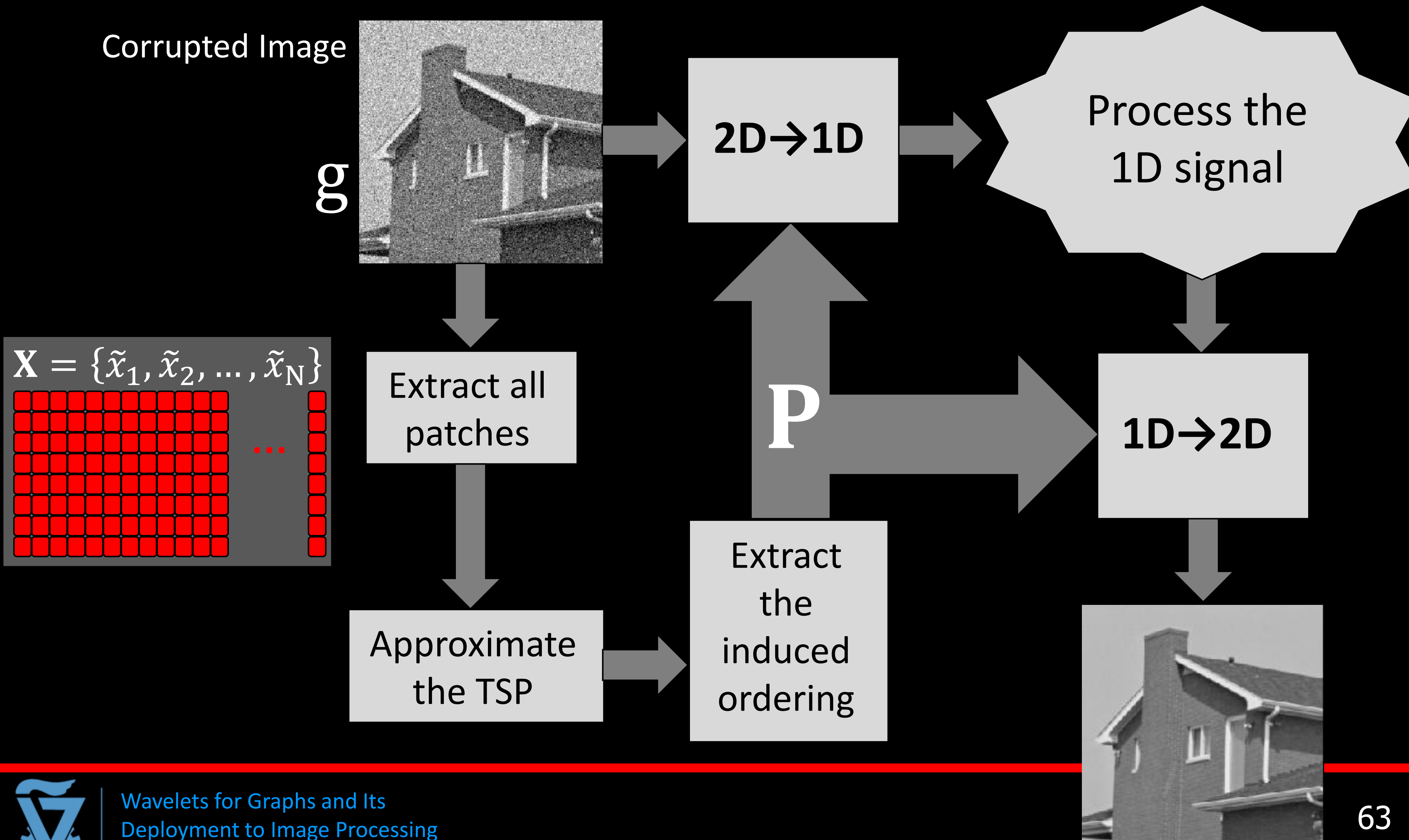
$$\underset{P}{\text{Argmin}} \sum_{i=2}^{N} \left\| x_i^p - x_{i-1}^p \right\|$$

$$\approx \underset{P}{\text{Argmin}} \sum_{i=2}^{N} \left\| \tilde{x}_i^p - \tilde{x}_{i-1}^p \right\|$$



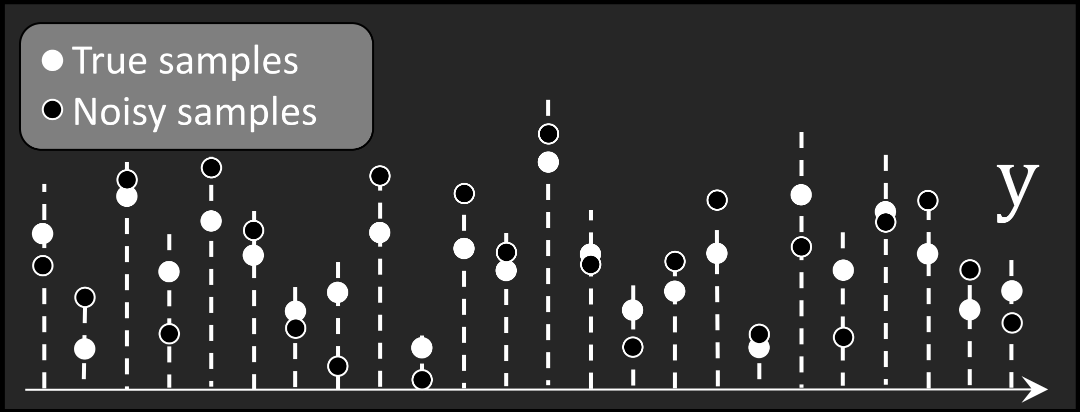
The order is similar, not necessarily the distances themselves

The Core Scheme

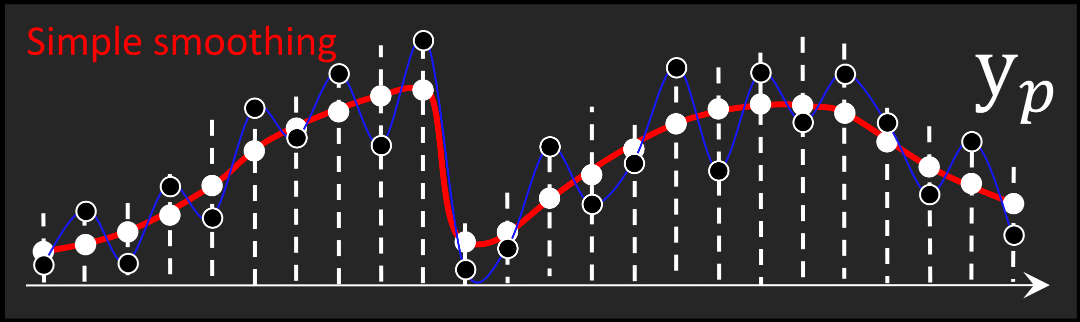


Intuition: Why Should This Work?

Noisy with σ=25 (20.18dB)



Ordering based on the noisy pixels



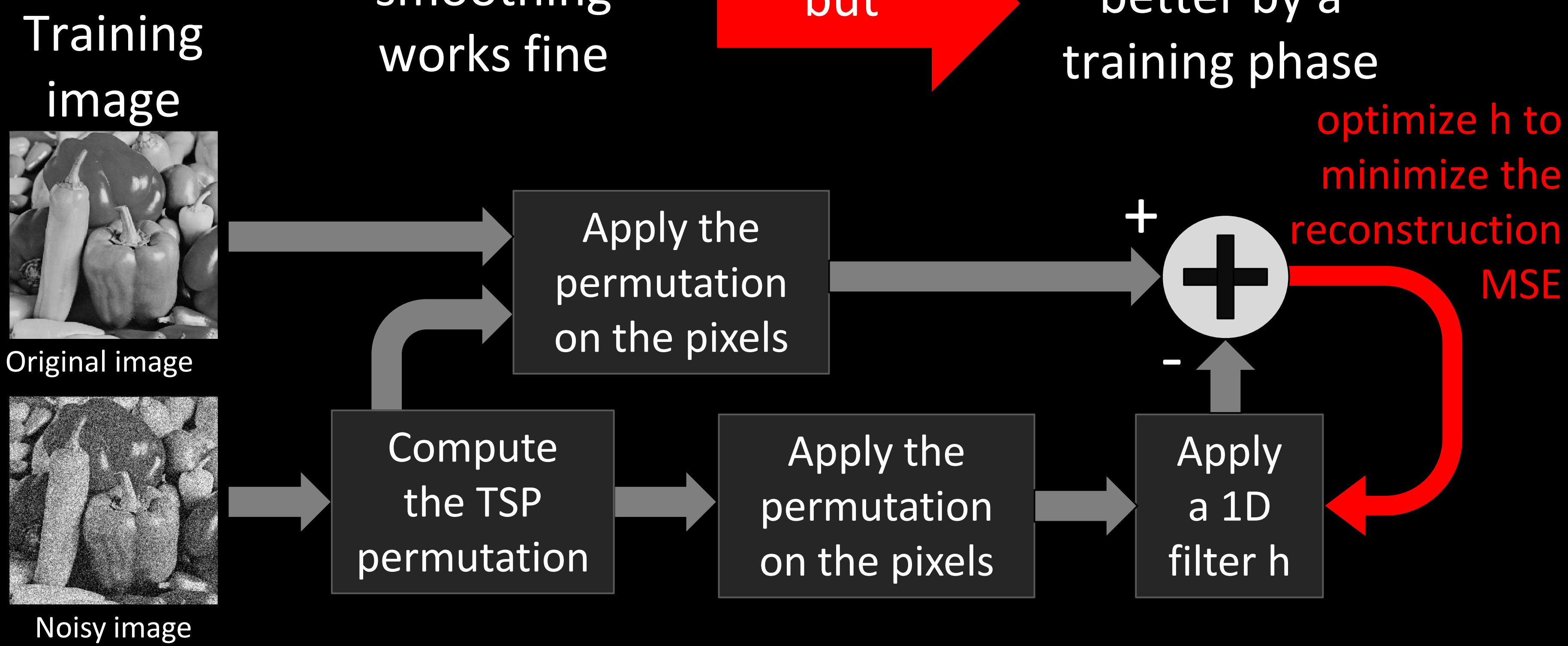
Reconstruction: 32.65dB

The "Simple Smoothing" We Do

Simple smoothing works fine

but

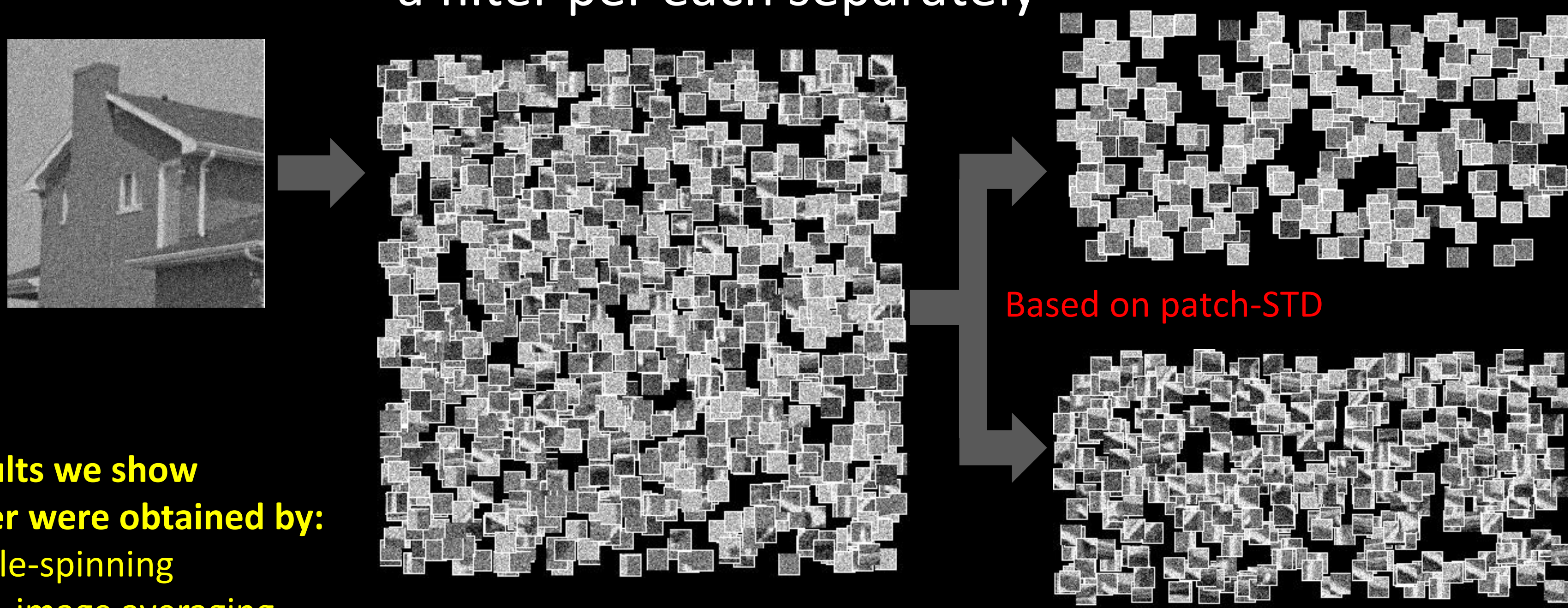
We can do better by a training phase



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.

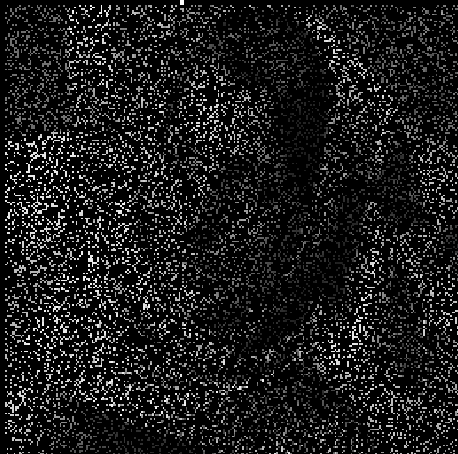
Denoising Results Using Patch-Reordering

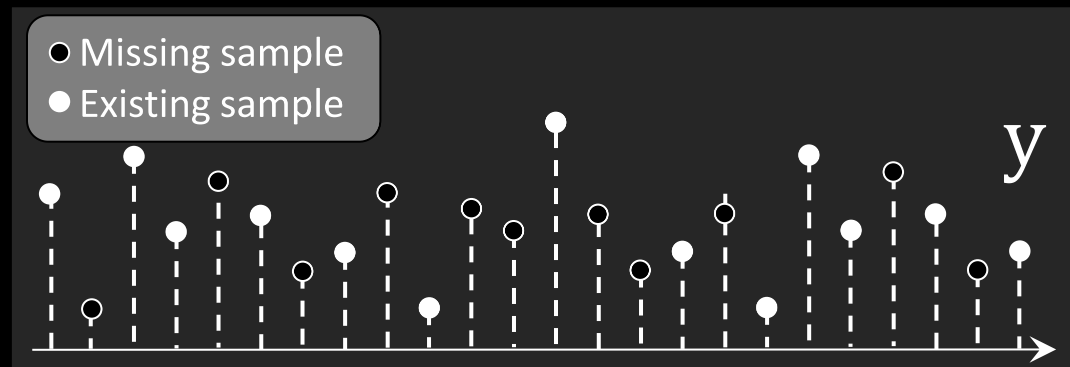
Image		σ/PSNR [dB]		
		10 / 28.14	25 / 20.18	50 / 14.16
Lena	K-SVD	35.49	31.36	27.82
	BM3D	**35.93**	**32.08**	28.86
	1st iteration	35.33	31.58	28.54
	2nd iteration	35.41	31.81	**29.00**
Barbara	K-SVD	34.41	29.53	25.40
	BM3D	**34.98**	**30.72**	27.17
	1st iteration	34.48	30.46	27.17
	2nd iteration	34.46	30.54	**27.45**
House	K-SVD	36.00	32.12	28.15
	BM3D	**36.71**	**32.86**	29.37
	1st iteration	35.58	32.48	29.37
	2nd iteration	35.94	32.65	**29.93**

Bottom line: This idea works very well, it is especially competitive for high noise levels, and a second iteration almost always pays off.

The Rationale

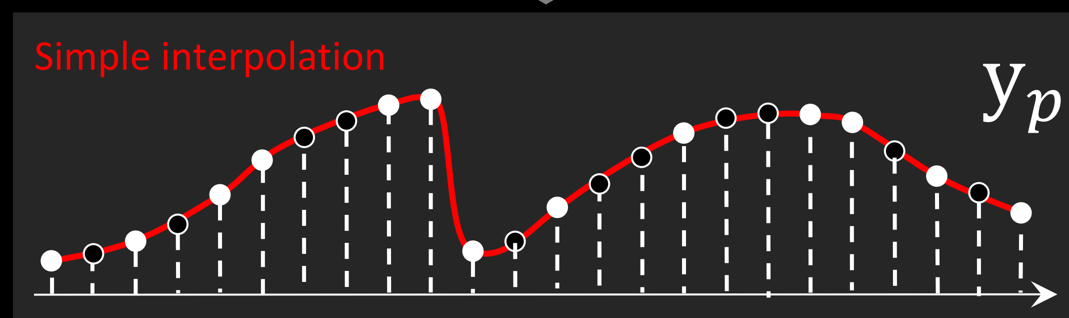
0.8 of the pixels are missing





Ordering*

* distance uses EXISTING pixels only



Reconstruction: 27.15dB

Inpainting Results – Examples

Given data 80% missing pixels | Bi-Cubic interpolation | Sparse representation recovery | 1^{st} iteration of the proposed alg. | 3^{rd} iteration of the proposed alg.

Inpainting Results

Reconstruction results from 80% missing pixels using various methods:

Bottom line:

(1) This idea works very well;

(2) It is operating much better than the classic sparse-rep. approach; and

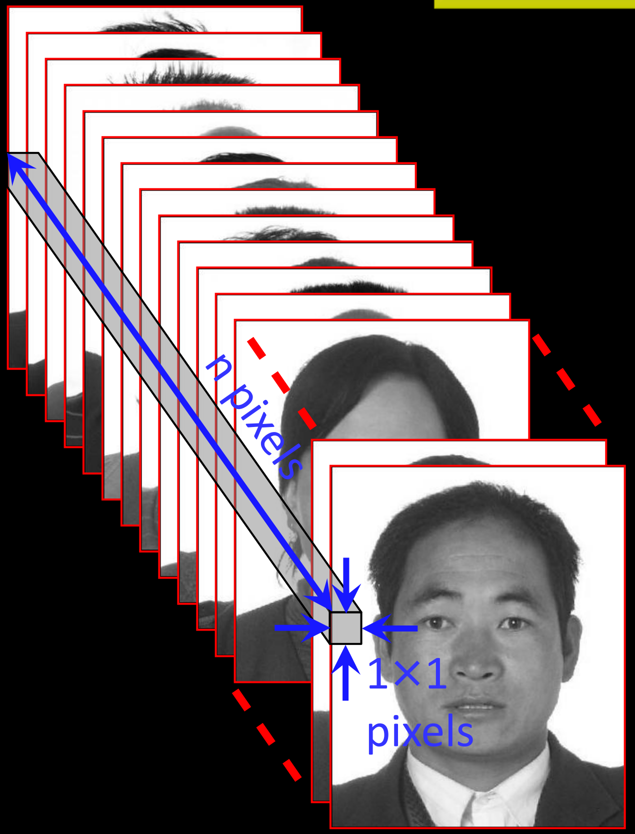
(3) Using more iterations always pays off, and substantially so.

Image	Method	PSNR [dB]
Lena	Bi-Cubic	30.25
	DCT + OMP	29.97
	Proposed (1st iter.)	30.25
	Proposed (2nd iter.)	31.80
	Proposed (3rd iter.)	**31.96**
Barbara	Bi-Cubic	22.88
	DCT + OMP	27.15
	Proposed (1st iter.)	27.56
	Proposed (2nd iter.)	29.34
	Proposed (3rd iter.)	**29.71**
House	Bi-Cubic	29.21
	DCT + OMP	29.69
	Proposed (1st iter.)	29.03
	Proposed (2nd iter.)	32.10
	Proposed (3rd iter.)	**32.71**

What About Image Compression?

SKIP?

- 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. pixel ordering), better results could be obtained.
- For our technique to operate well, we find the best common pixel-ordering fitting a training set of facial images.
- Our pixel ordering is therefore designed on patches of size 1×1×n pixels from the training volume.
- Geometric alignment of the image is very helpful and should be done [Goldenberg, Kimmel, & E. ('05)].



Compression by Pixel-Ordering

Detect main features and warp the images (20 bytes)

↓

Compute the mean image and subtract it

↓

Find the common ordering that creates the smoothest path

↓

2D→1D, apply wavelet and code leading coefficients

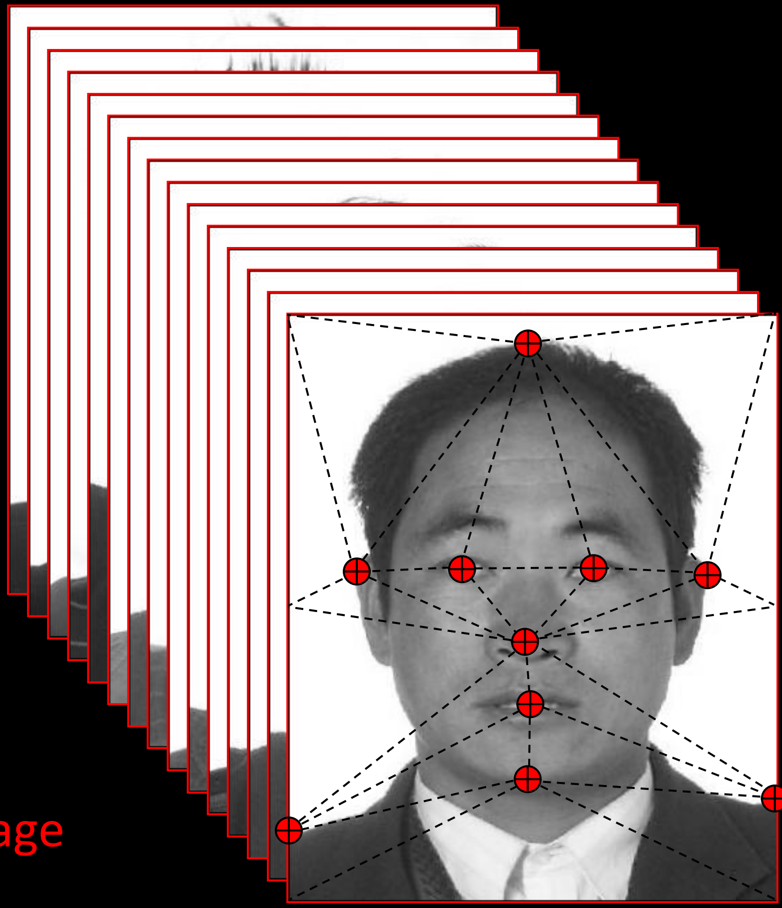
↓

Warp, remove the mean, permute, apply wavelet on the 1D signal and code

On the training set

On the test image

Training set (2500 images)

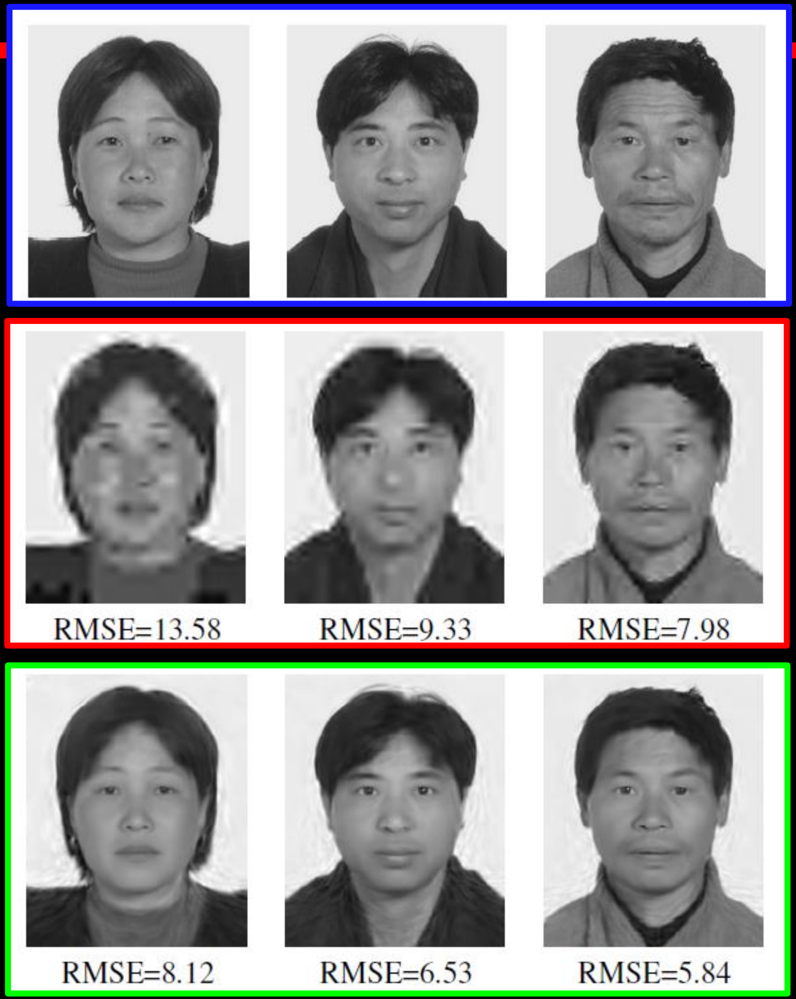


Results

The original images

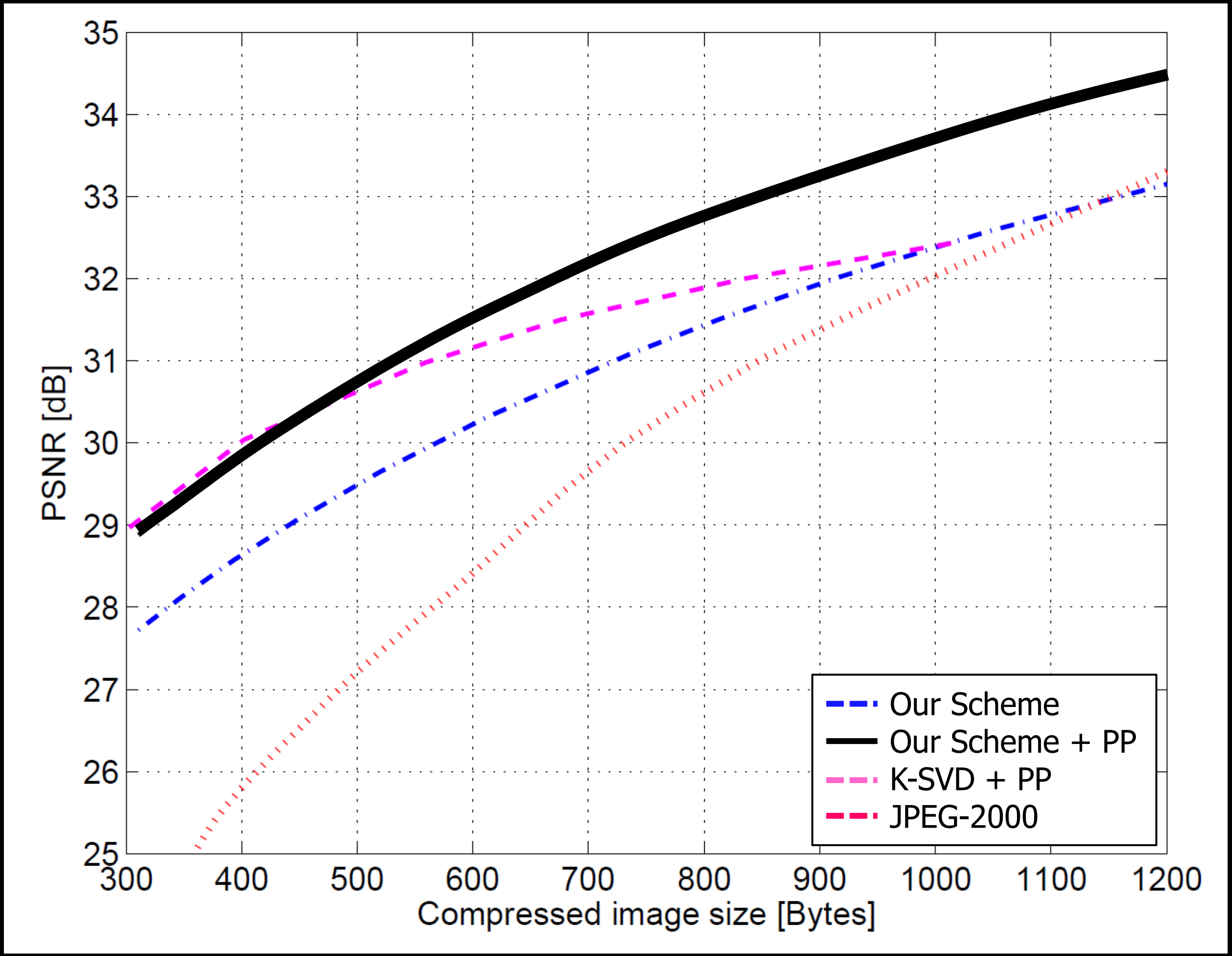
JPEG2000

Our scheme



400 bytes 600 bytes 800 bytes

Rate-Distortion Curves



Part IV – Time to Finish

Conclusions

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

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

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

What Next ?

Demonstrating the proposed wavelet on more data clouds/graphs

Replace the TSP ordering by MDS?

Exploiting the known distances?

Sparse Representations and learned dictionaries in the ordered domain?

Why TSP? Who says we cannot revisit patches?

Improving the TSP approximation solver

Replace "sub-image averaging" with a sparsifying transform

?

Pixel permutation as regularizer

Lifting scheme for treating clouds?

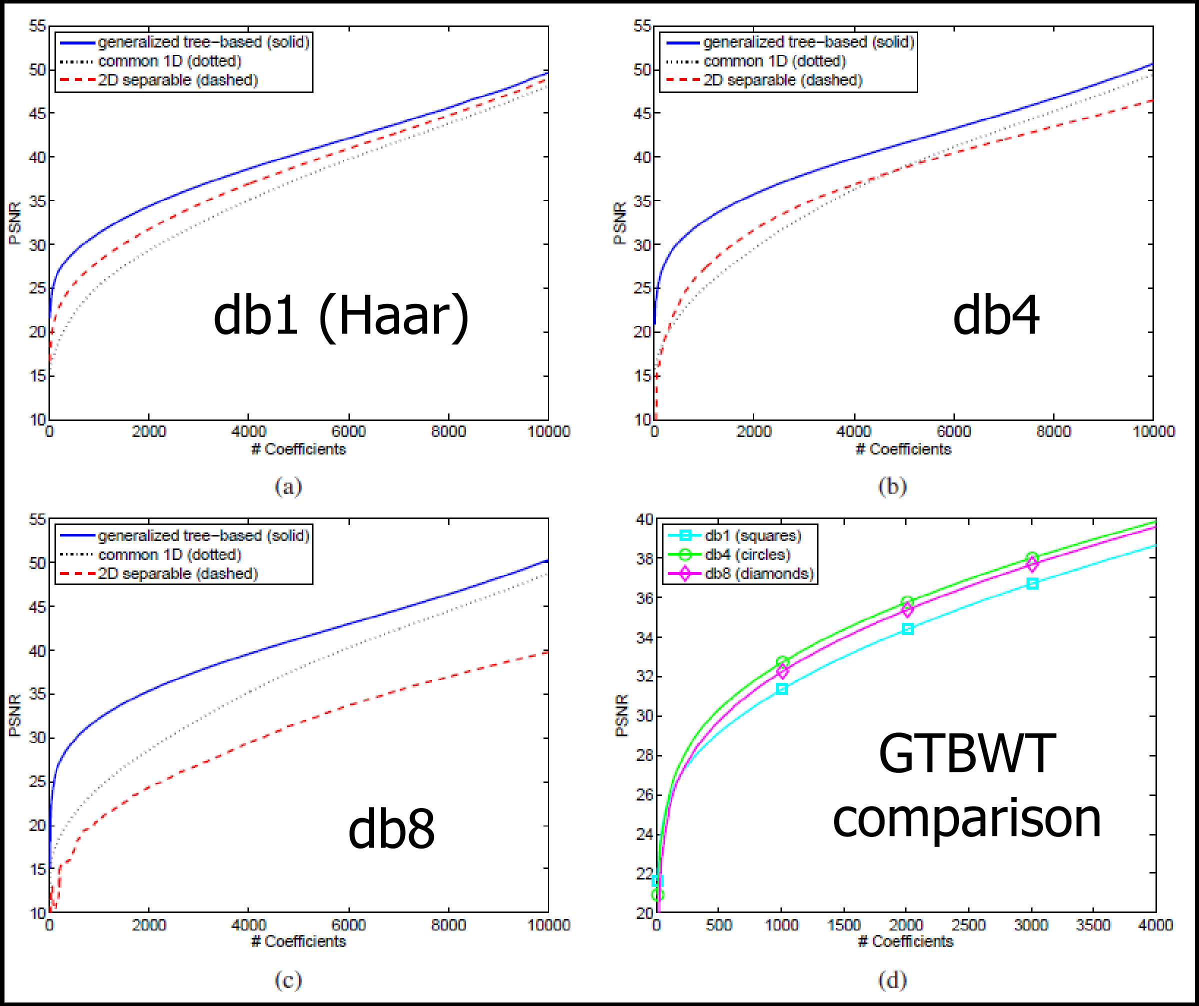
Thank you for your time,

and …

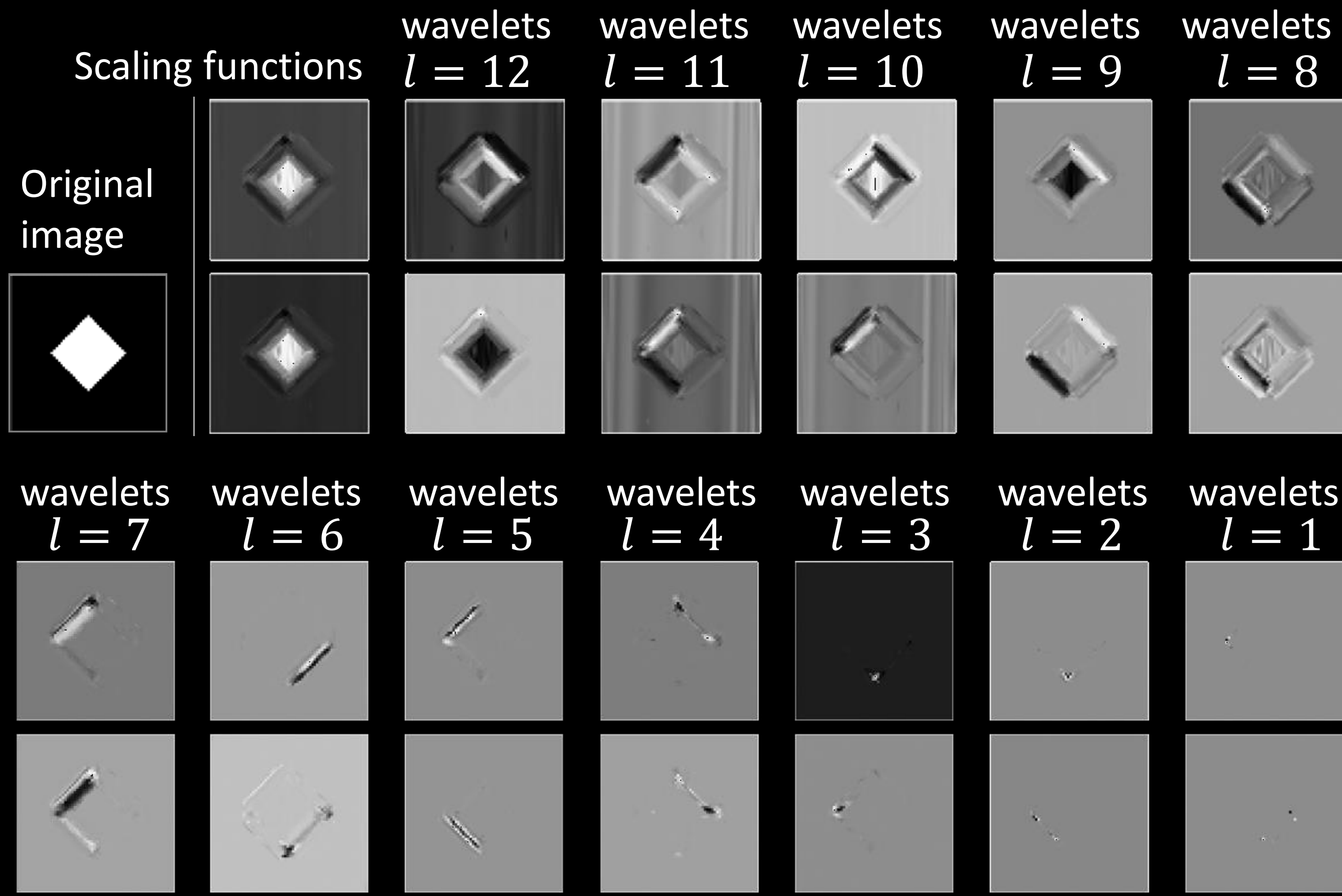
Thanks to the Organizers
and especially
Michael Ng

Questions?

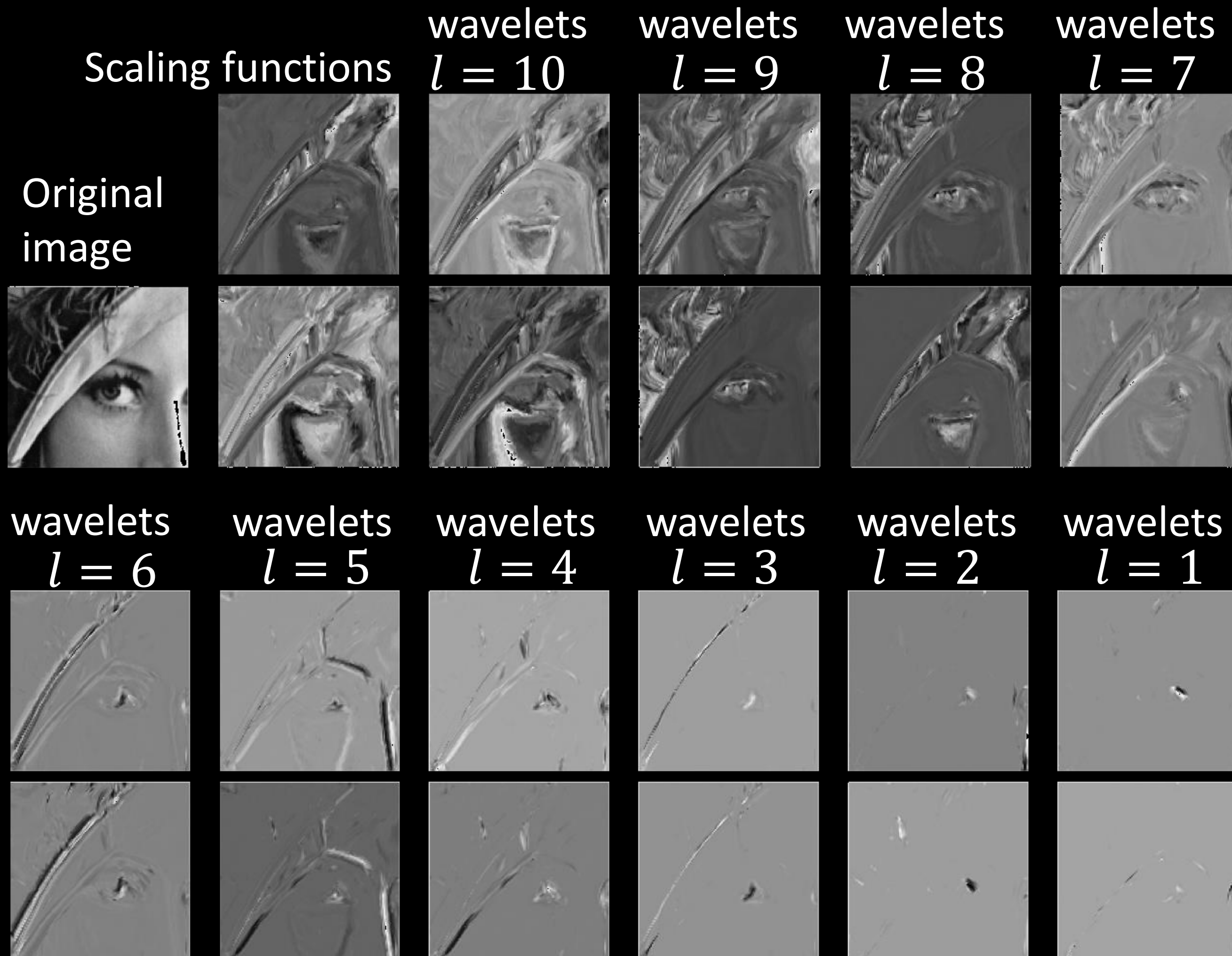
Comparison Between Different Wavelets



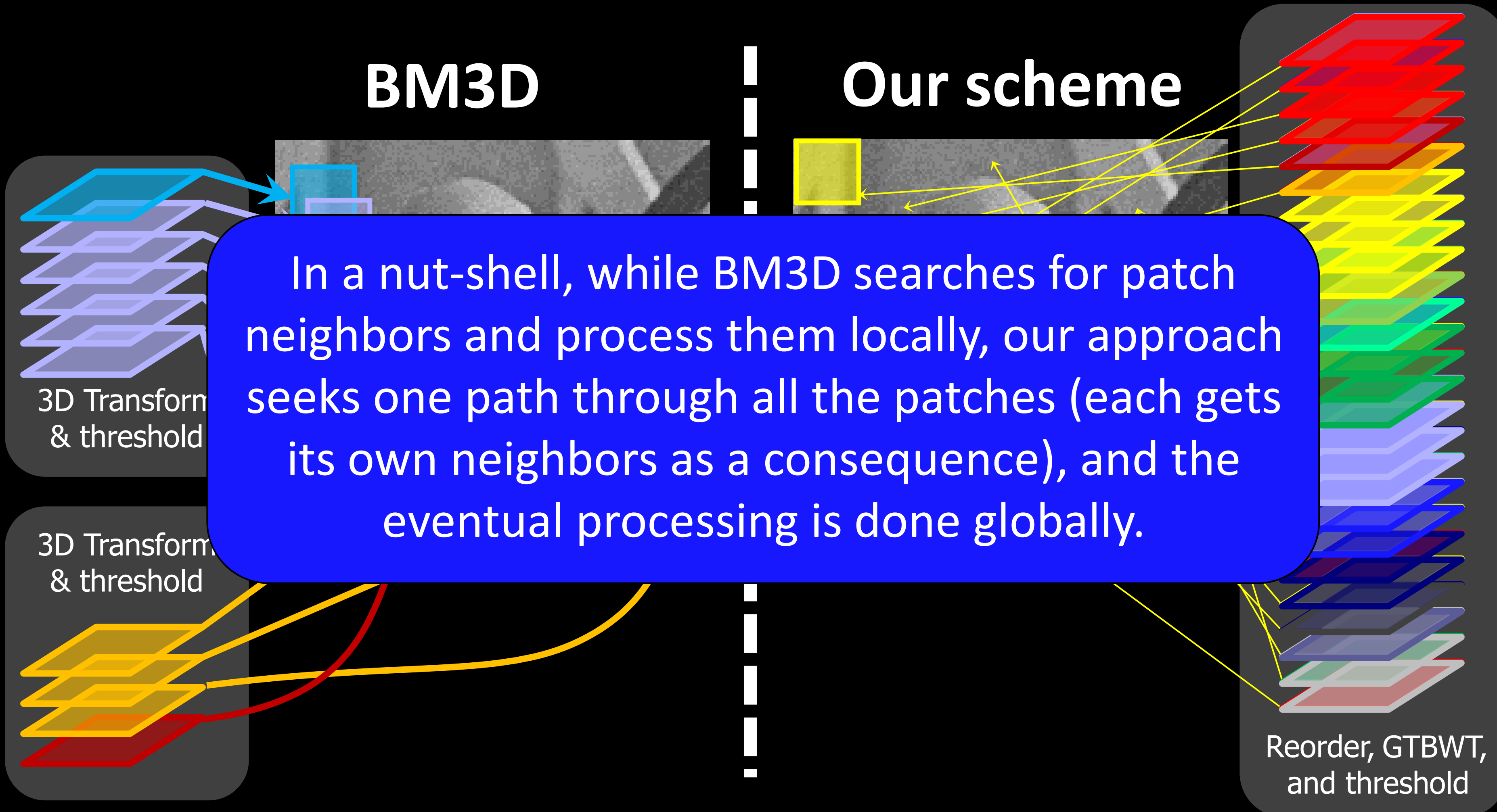
The Representation's Atoms – Synthetic Image



The Representation's Atoms – Lenna



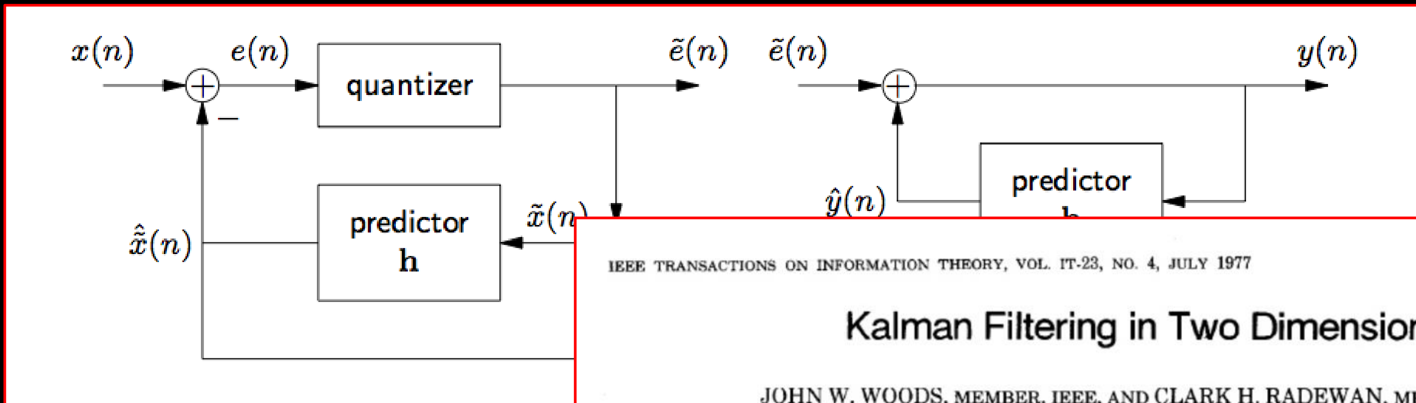
Relation to BM3D?



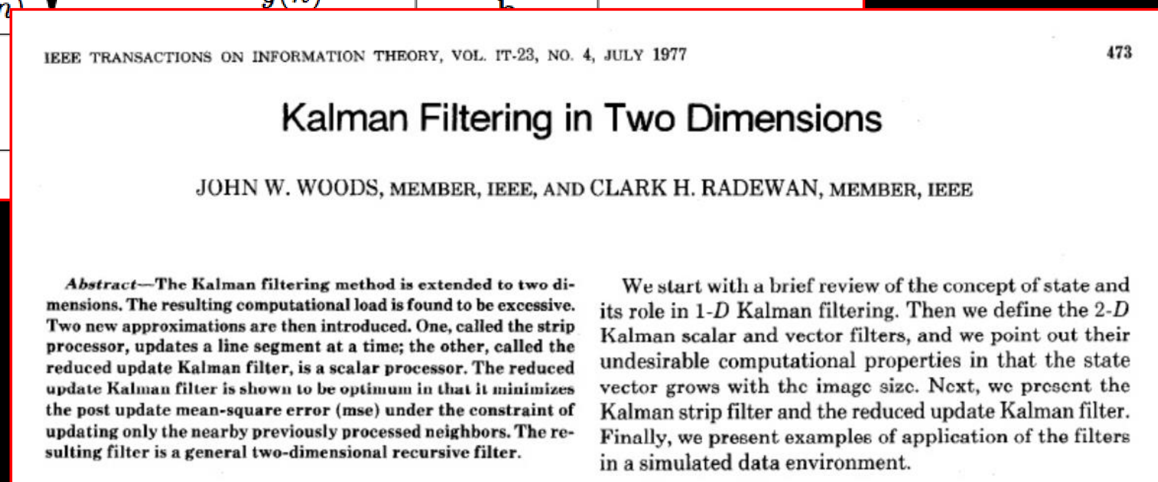
In a nut-shell, while BM3D searches for patch neighbors and process them locally, our approach seeks one path through all the patches (each gets its own neighbors as a consequence), and the eventual processing is done globally.

2D → 1D Processing Examples

DPCM Image Compression



Kalman Filtering for Denoising


IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-23, NO. 4, JULY 1977 473

Kalman Filtering in Two Dimensions

JOHN W. WOODS, MEMBER, IEEE, AND CLARK H. RADEWAN, MEMBER, IEEE

Abstract—The Kalman filtering method is extended to two dimensions. The resulting computational load is found to be excessive. Two new approximations are then introduced. One, called the strip processor, updates a line segment at a time; the other, called the reduced update Kalman filter, is a scalar processor. The reduced update Kalman filter is shown to be optimum in that it minimizes the post update mean-square error (mse) under the constraint of updating only the nearby previously processed neighbors. The resulting filter is a general two-dimensional recursive filter.

We start with a brief review of the concept of state and its role in 1-*D* Kalman filtering. Then we define the 2-*D* Kalman scalar and vector filters, and we point out their undesirable computational properties in that the state vector grows with the image size. Next, we present the Kalman strip filter and the reduced update Kalman filter. Finally, we present examples of application of the filters in a simulated data environment.

While this 2D → 1D trend is an “old-fashion” trick, it is still very much active and popular in industry and academic work.