Sparse Modeling of Graph-Structured Data … and … Images

Michael Elad
The Computer Science Department
The Technion

Workshop on Mathematical Approaches to Large-Dimensional Data Analysis

MEXT-ISM Coop-Math Prgoram and ISM Research Center for Statistical Machine Learning
Dates: March 13 - 15, 2014. Place: The Institute of Statistical Mathematics, Tachikawa, Tokyo

Technion
Israel Institute of Technology

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

erc

SPARSITY:
What is it Good For?

This part relies on the following two papers:

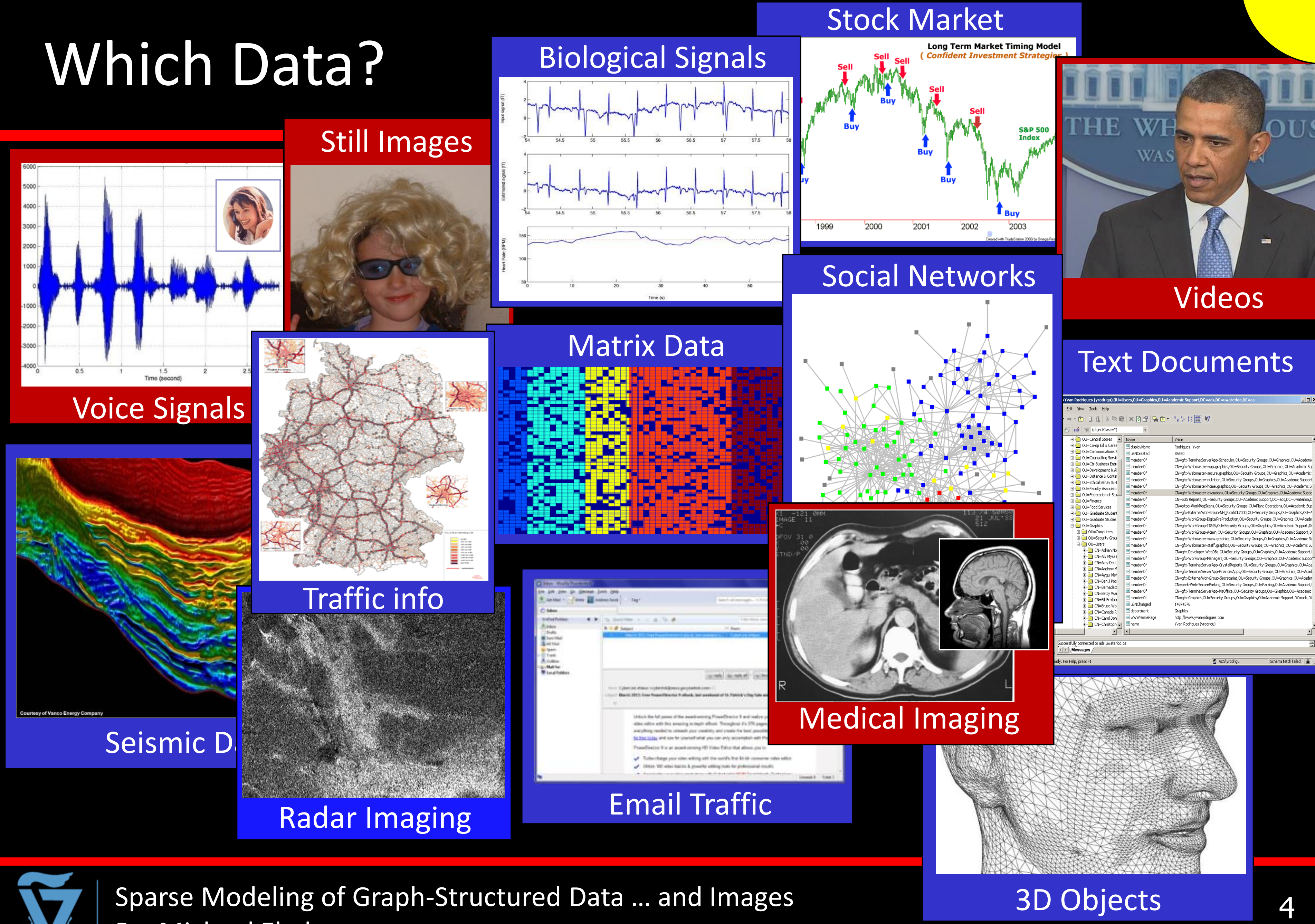
- M. Elad, Sparse and Redundant Representation Modeling — What Next?, IEEE Signal Processing Letters, Vol. 19, No. 12, Pages 922-928, December 2012.
- A.M. Bruckstein, D.L. Donoho, and M. Elad, From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images, SIAM Review, Vol. 51, No. 1, Pages 34-81, February 2009.

Good News

Today, we have the technology and the know-how to effectively process

data

Which Data?


Stock Market
Long Term Market Timing Model
(Confident Investment Strategies)
Sell
Buy
S&P 500 Index
1999
2000
2001
2002
2003
Biological Signals
Still Images
Videos
Voice Signals
Time (second)
Social Networks
Matrix Data
Text Documents
Traffic info
Seismic D
Courtesy of Vanco Energy Company
Medical Imaging
Radar Imaging
Email Traffic
3D Objects

What Processing?

What can we do for such signals?

- Denoising – removal of noise from the data
- Interpolation (inpainting) – recovery of missing values
- Prediction – extrapolating the data beyond the given domain
- Compression – reduction of storage and transmission volumes
- Inference (inverse problems) – recovery from corrupted measurements
- Separation – breaking down a data to its morphological “ingredients”
- Anomaly detection – discovering outliers in the data
- Clustering – gathering subsets of closely related instances within the data
- Summarizing – creating a brief version of the essence of the data

So, Here is a Simple Question

Why all This is Possible?

- Is it obvious that all these processing options should be possible?
- Consider the following data source:

IID Random Number Generator $\mathbb{N}(0,1)$ → $\underline{x} = \{x_1, x_2, x_3, \dots, x_N\}$

Many of the processing tasks mentioned above are impossible for this data

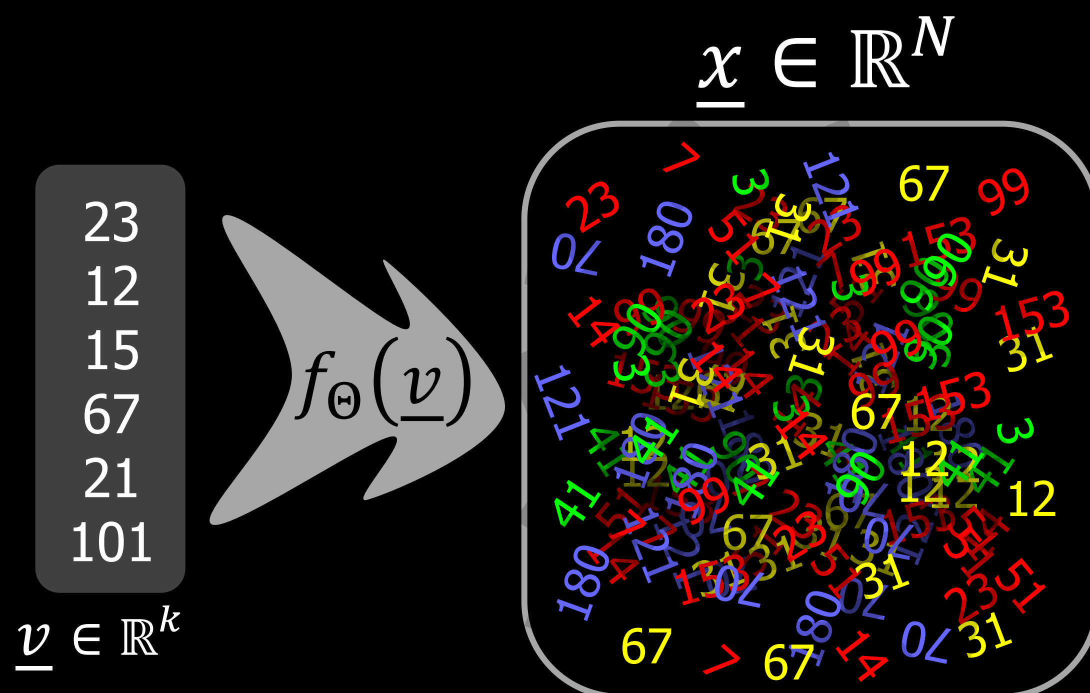
- Is there something common to all the above-mentioned signals, that makes them "processable"?

Why? We Know The Answer(s)

- Low Entropy
- Low Dimensionality
- High Redundancy
- Inner Structure
- Self Dependencies
- Self Similarity
- Manifold Structure
- ….

Our Data is Structured

A signal composed of N scalar numbers has $k \ll N$ true degrees of freedom



Data Models

A "wisely" chosen function

The data we operate on

The low-dimensional representation or "innovation"

$$\underline{x} = f_{\Theta}(\underline{v})$$

Parameters that govern the model (to be learned)

Models are arbitrary beliefs and are **ALWAYS** wrong

Note: This is not the only way to impose structure on data – this approach is known as the "synthesis model"

Processing Data Using Models

Q: Why all This is Possible?

Processing signals
(denoise, interpolate, predict, compress, infer, separate, detect, cluster, summarize, …)

A: Because of the structure!

Processing signals requires knowledge of their structure – we need the model $\underline{x} = f_{\Theta}(\underline{v})$, along with its learned parameters

Processing Data Using Models

Example 1 - Compression

Given a signal $\underline{x}$, its compression is done by computing its representation $\underline{v}$:

$$\underline{x} = f_\Theta(\underline{v})$$

Example 2 - Inference

Given a deteriorated version of a signal, $\underline{y} = \boldsymbol{M}\underline{x} + \underline{z}$, recovering $\underline{x}$ from $\underline{y}$ is done by projecting $\underline{y}$ onto the model:

$$\hat{\underline{x}} = \min_{\underline{v},\underline{x}} \left\| \underline{y} - \boldsymbol{M}\underline{x} \right\|_2^2 \quad s.t.\ \underline{x} = f_\Theta(\underline{v})$$

This covers tasks such as denoising, interpolating, inferring, predicting, …

Example 3 - Separation

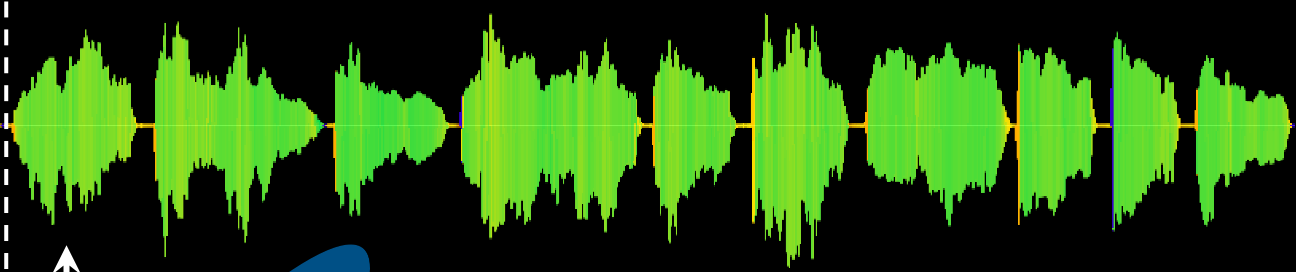
Given a noisy mixture of two signals, $\underline{y} = \underline{x}_1 + \underline{x}_2 + \underline{z}$, each emerging from a different model, separation is done by

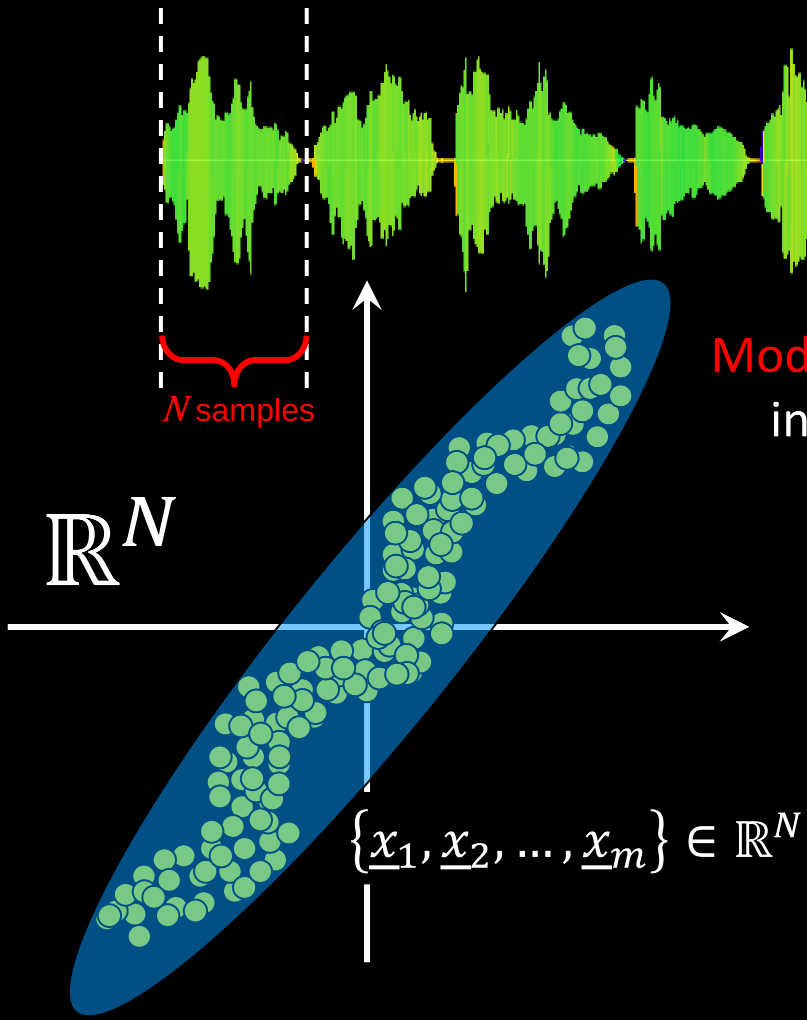
$$\hat{\underline{x}}_1, \hat{\underline{x}}_2 = \min_{\underline{v}_1,\underline{v}_2,\underline{x}_1,\underline{x}_2} \left\| \underline{y} - \underline{x}_1 - \underline{x}_2 \right\|_2^2$$

$$s.t.\ \ \underline{x}_1 = f_\Theta^A(\underline{v}_1)$$
$$\underline{x}_2 = f_\Phi^B(\underline{v}_2)$$

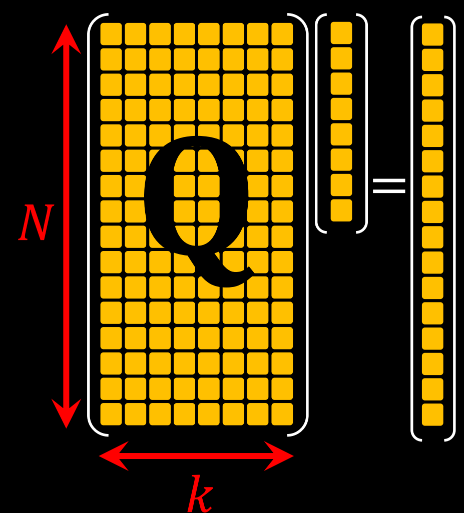
The goodness of the separation is dictated by the overlap between the two models

An Example: PCA-KLT-Hotelling





Model assumption: All these data vectors reside in a single subspace S of dimension $k \ll N$



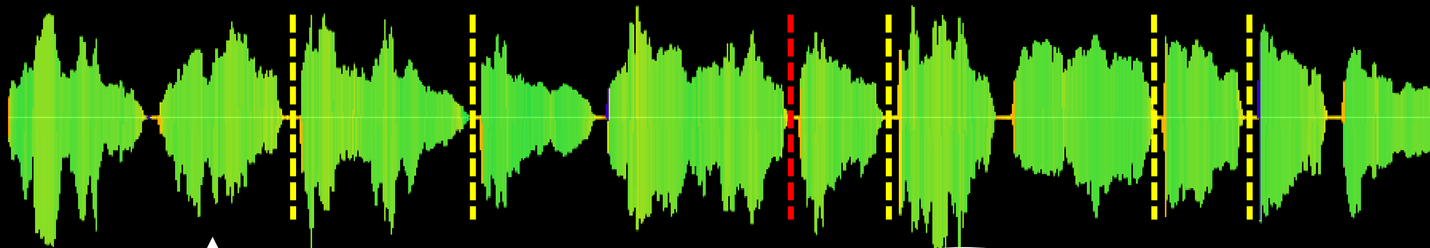
$\underline{x} = f_{\Theta}(\underline{v})$?

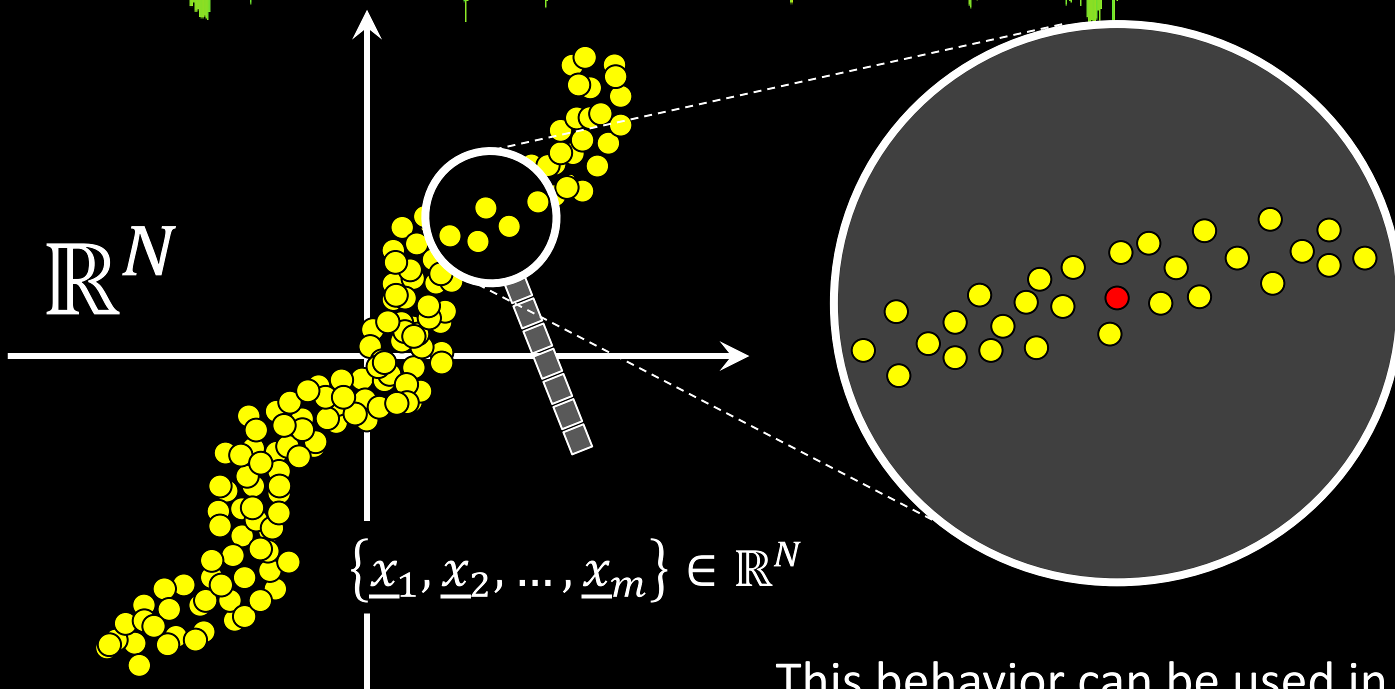
$$\mathbf{Q}\underline{v}_i = \underline{x}_i$$

for $i = 1, 2, \ldots, m$

Learning Θ: finding the best $\mathbf{Q}$

Improving the Model – Local Linearity

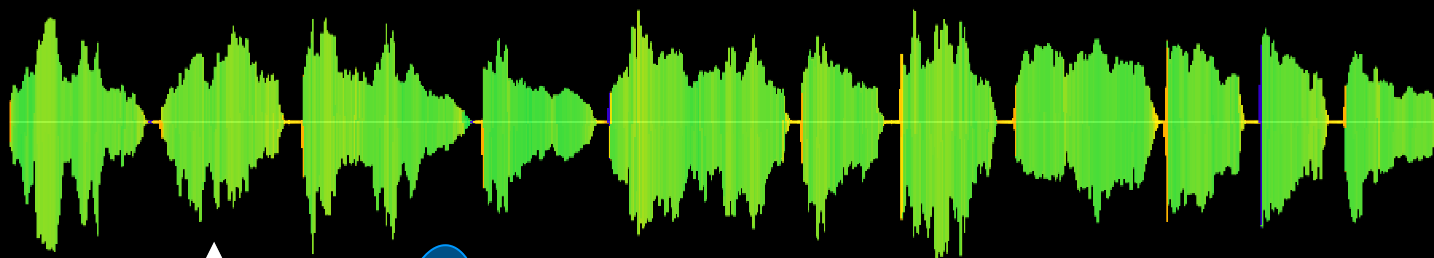


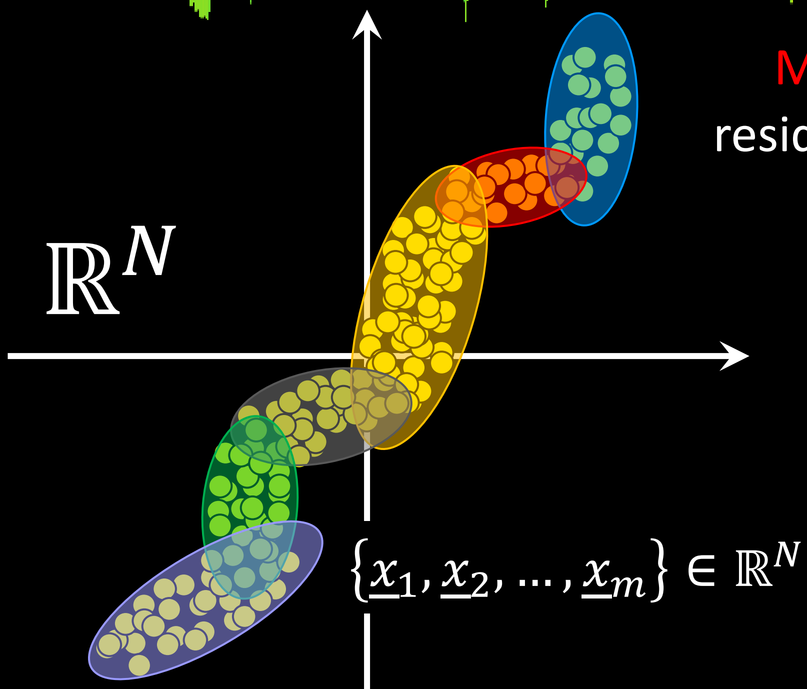


We may assume that around every point $\underline{x}_i$, its nearest neighbors (in $\mathbb{R}^N$) form a very low-dimensional subspace

This behavior can be used in various ways …

Union of (Affine) Subspaces





Model assumption: All these data vectors reside in a union of L affine subspaces, $\bigcup_{l=1}^{L} S_l$, (UoS) of low dimensions $k_l \ll N$

$$\underline{x} = f_{\Theta}(\underline{v})\ ?$$

Identify the subspace $\underline{x}_i$ belongs to, $\underline{x}_i \in S_l$ (Mahalanobis Distance) and $\rightarrow \underline{x}_i = \mathbf{Q}_l \underline{v}_i + \underline{c}_l$

Fitting the model: finding $\{\mathbf{Q}_l, \underline{c}_l\}_{l=1}^{L}$

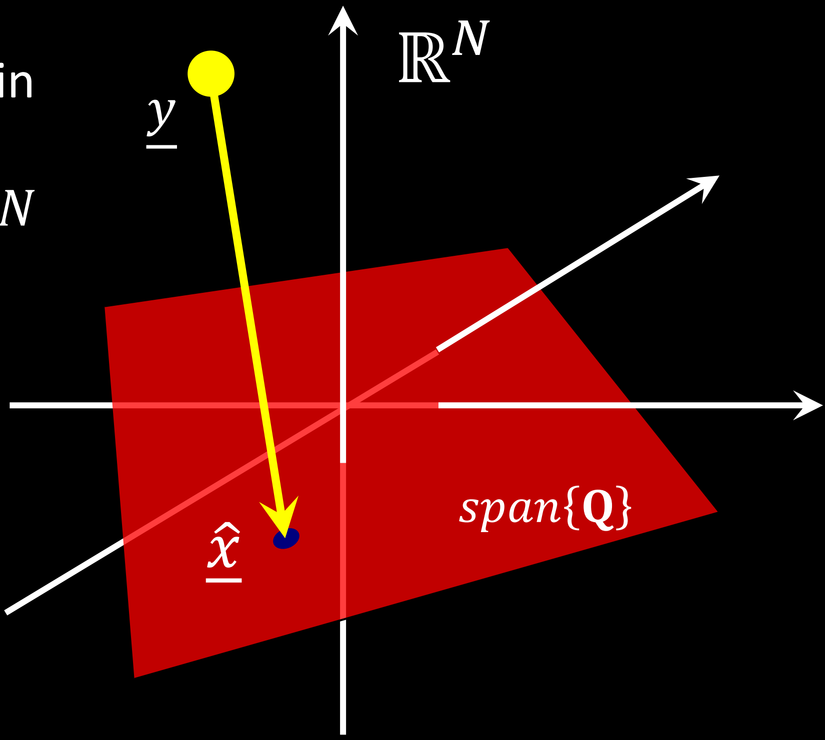
Example: PCA Denoising ($\underline{y} = \underline{x} + \underline{z}$)

The case of PCA/KLT:

$$\hat{\underline{x}} = \min_{\underline{v},\underline{x}} \left\| \underline{y} - \underline{x} \right\|_2^2 \quad s.t.\ \underline{x} = \mathbf{Q}\underline{v}$$

$$\hat{\underline{x}} = \mathbf{Q}(\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\underline{y} = \mathbf{Q}\mathbf{Q}^{\dagger}\underline{y}$$

- The data vector $\underline{y}$ is projected onto the k-dimensional space spanned by $\mathbf{Q}$
- As the noise is spread evenly in the N-dim. space, only k/N of it remains $\rightarrow$ effective denoising



The Case of UoS:

project to all the L subspaces, and choose the outcome that is closest to $\underline{y}$ (complexity is $\times$L)

Lets Talk About Sparsity

Sparsity: A different way to describe a signal's structure

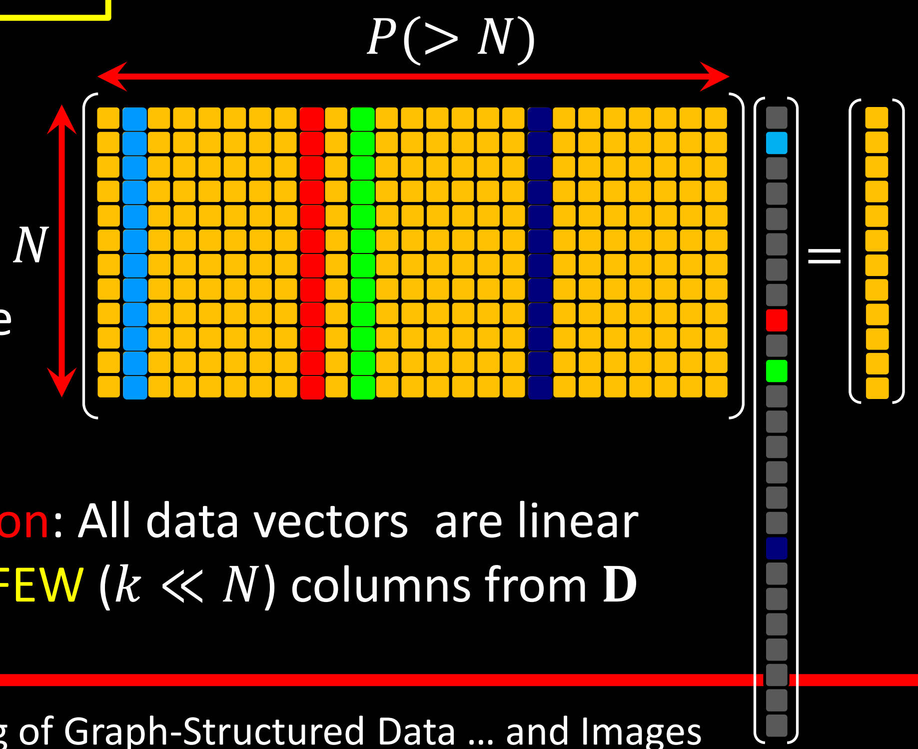
$$\{\underline{x}_1, \underline{x}_2, \dots, \underline{x}_m\} \in \mathbb{R}^N$$

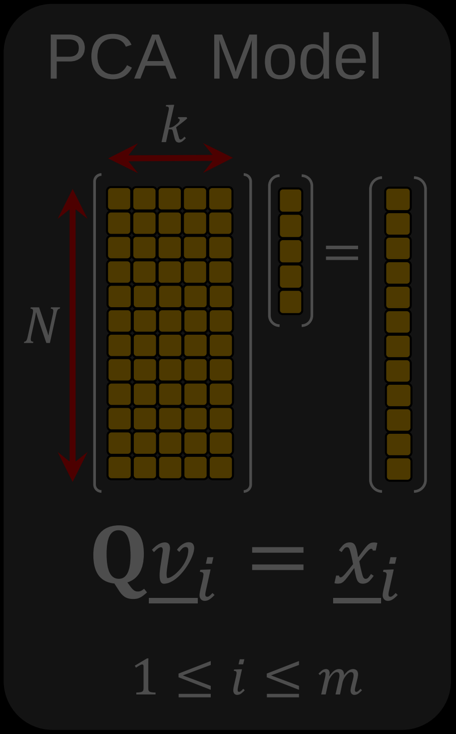
D: *Dictionary*
Its columns: *Atoms*

$$\mathbf{D}\underline{a}_i = \underline{x}_i$$

$1 \le i \le m$

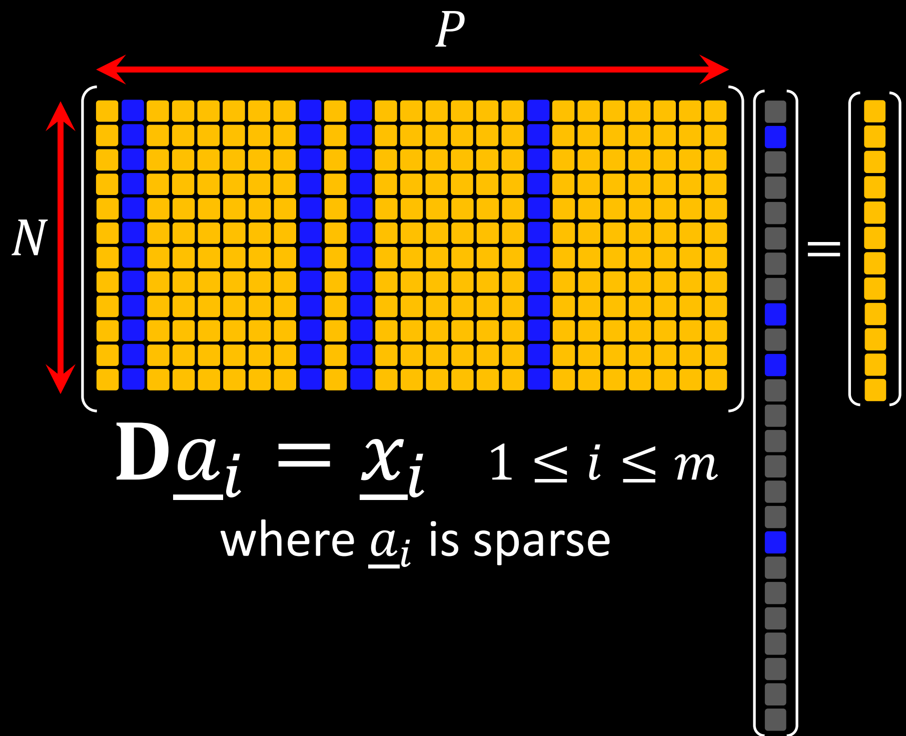
where $\underline{a}_i$ is sparse





Model assumption: All data vectors are linear combination of FEW ($k \ll N$) columns from $\mathbf{D}$

Sparsity – A Closer Look



$$\mathbf{D}\underline{a}_i = \underline{x}_i \quad 1 \le i \le m$$

where $\underline{a}_i$ is sparse

Dimensionality Reduction

If $\|\underline{a}_i\|_0 = k \ll N$, this means that the information carried by $\underline{a}_i$ is $2k \ll N$, thus giving effective compression

Geometric Form

Example: $N = 200, P = 400, k = 10$

- Dim. reduction factor: $\frac{N}{2k} = 10$
- # of subspaces: $\binom{400}{10} \approx 2.6e + 19$

This model leads to a much richer UoS structure, with (exponentially) many more subspaces and yet all are defined through the concise matrix $\mathbf{D}$

Sparsity in Practice: Back to Denoising

Sparsity-Based Model:

$$\underline{\hat{a}} = \min_{\|\underline{a}\|_0 = k} \left\| \underline{y} - \mathbf{D}\underline{a} \right\|_2^2 \rightarrow \underline{\hat{x}} = \mathbf{D}\underline{\hat{a}}$$

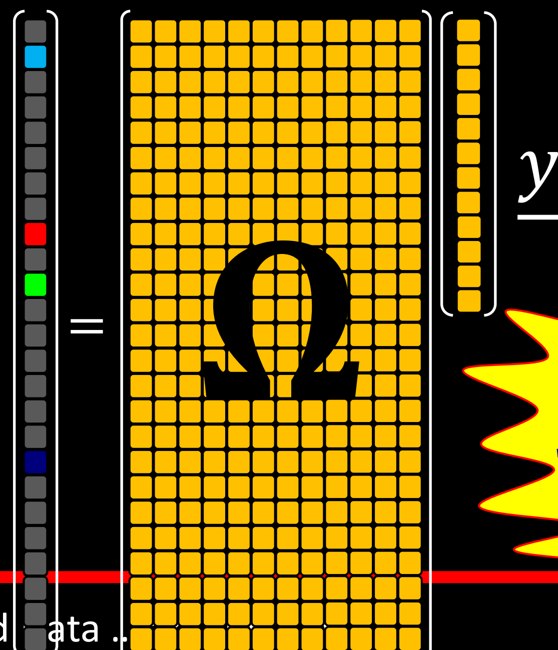
Find the support (the subspace the signal belongs to) and project

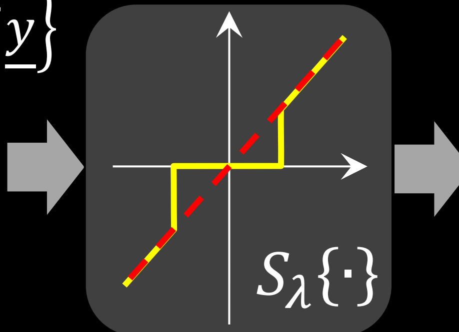
This is known as the Pursuit problem known to be NP-Hard

$$\underline{y} = \underline{x} + \underline{z} \quad \Rightarrow \quad \underline{\hat{x}} = \min_{\underline{v},\underline{x}} \left\| \underline{y} - \underline{x} \right\|_2^2 \quad s.t. \ \underline{x} = f_\Theta(\underline{v})$$

Approximation by the THR algorithm:

$$\underline{\hat{a}} = S_\lambda\left\{\mathbf{\Omega}\underline{y}\right\} = S_\lambda\left\{\mathbf{D}^\dagger\underline{y}\right\}$$





What if $\underline{y} = \underline{x}$?
$\mathbf{\Omega}$ should "sparsify" the signals of interest

To Summarize So Far

Processing data is enabled by an appropriate modeling that exposes its inner structure

Broadly speaking, an effective way to model data is via sparse representations

This leads to a rich and highly effective and popular Union-of-Subspaces model

We shall now turn to adopt this concept for non-conventional data structure - graph

Note: Our motivation is “image processing”

Processing GRAPH Structured Data

Joint work with

Idan Ram   Israel Cohen
The Electrical Engineering department
Technion – Israel Institute of Technology

This part relies on the following two papers:

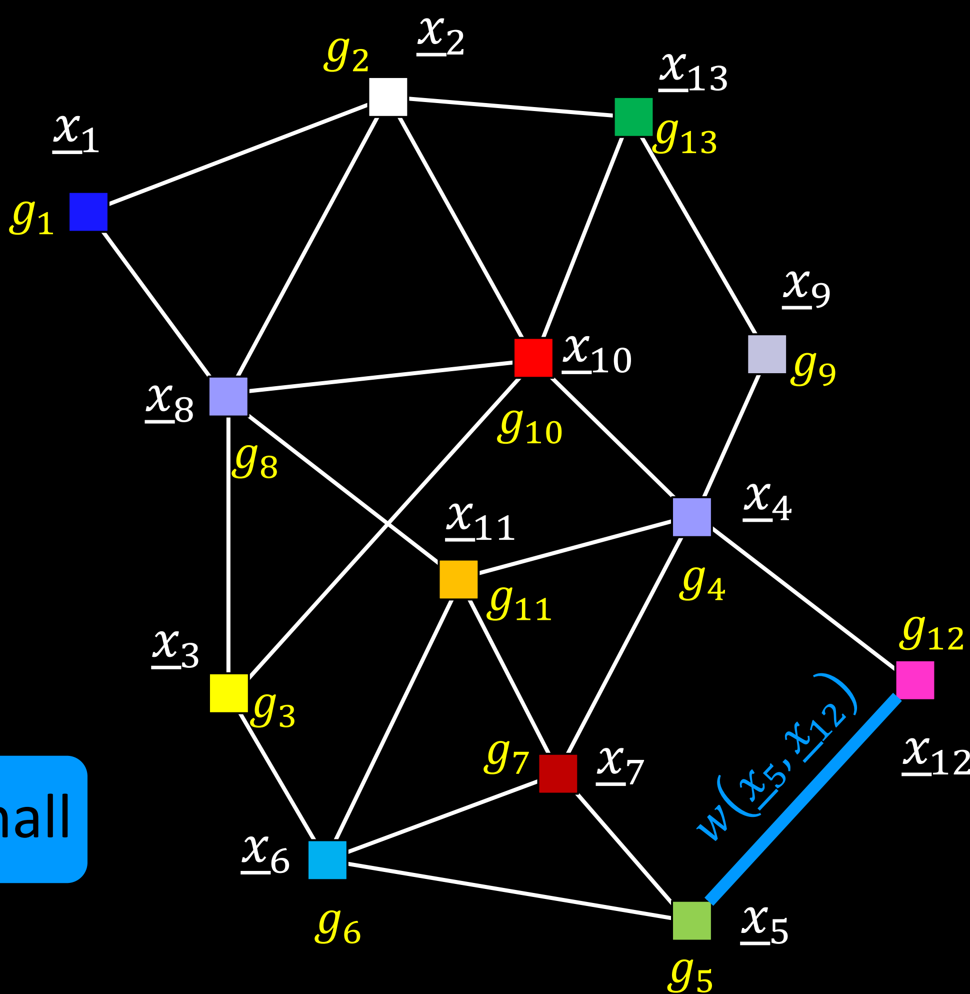
- I. Ram, M. Elad, and I. Cohen, "Generalized Tree-Based Wavelet Transform", IEEE Trans. Signal Processing, vol. 59, no. 9, pp. 4199–4209, 2011.
- I. Ram, M. Elad, and I. Cohen, "Redundant Wavelets on Graphs and High Dimensional Data Clouds", IEEE Signal Processing Letters, Vol. 19, No. 5, pp. 291–294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

Problem Formulation

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

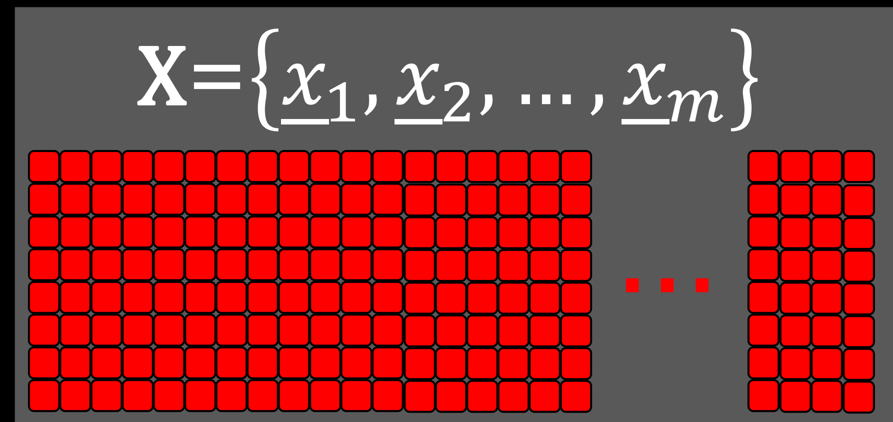
$$w\left(\underline{x}_i, \underline{x}_j\right) \text{ small} \rightarrow \left|g_i - g_j\right| \text{ small}$$

for almost every pair (i, j)



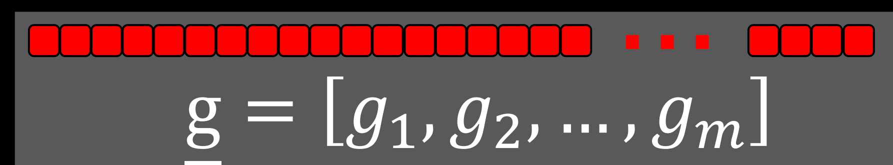
Different Ways to Look at This Data

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



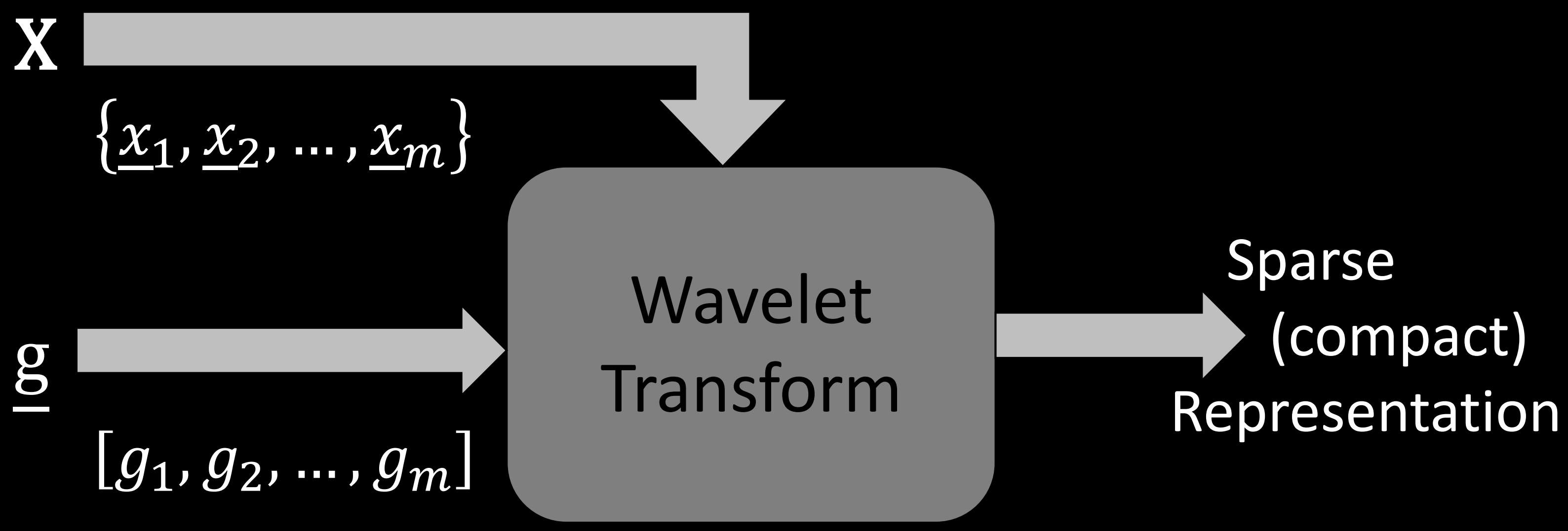
- A scalar function is defined on these coordinates, $g: \mathbf{X} \rightarrow \mathbb{R}$, giving $\underline{\mathrm{g}} = [g_1, g_2, \dots, g_m]$

- We may regard this dataset as a set of m samples taken from a high dimensional function $g: \mathbb{R}^N \rightarrow \mathbb{R}$



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

Our Goal



Why Wavelet?

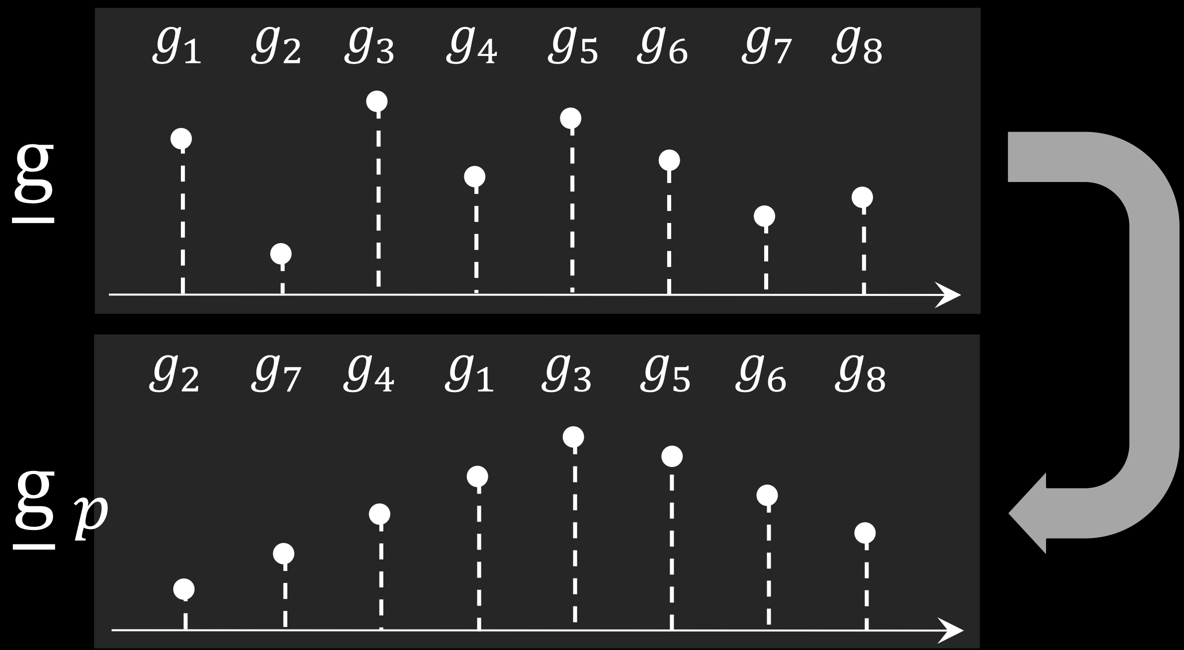
- Wavelet for regular piece-wise smooth signals is a highly effective "sparsifying transform"
- We would like to imitate this for our data structure

Wavelet for Graphs – A Wonderful Idea

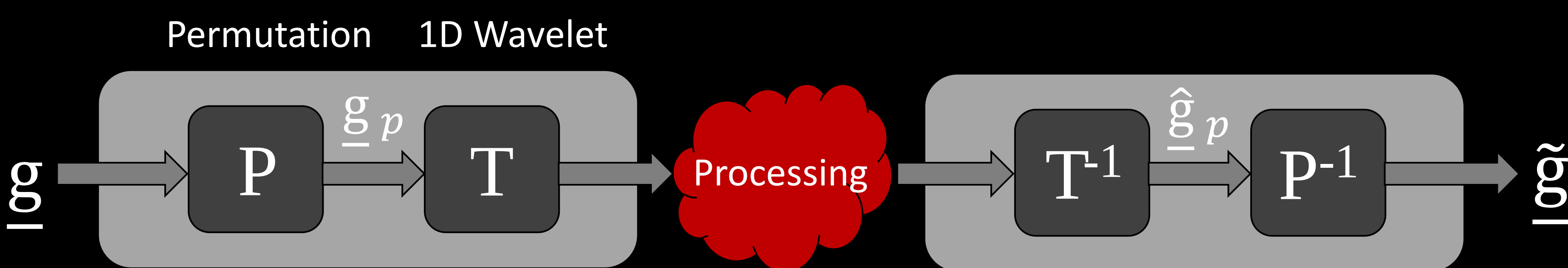
I wish we would have thought of it first …

- "Diffusion Wavelets"
  R. R. Coifman, and M. Maggioni, 2006.
- "Multiscale Methods for Data on Graphs and Irregular 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, 2012

The Main Idea – Permutation

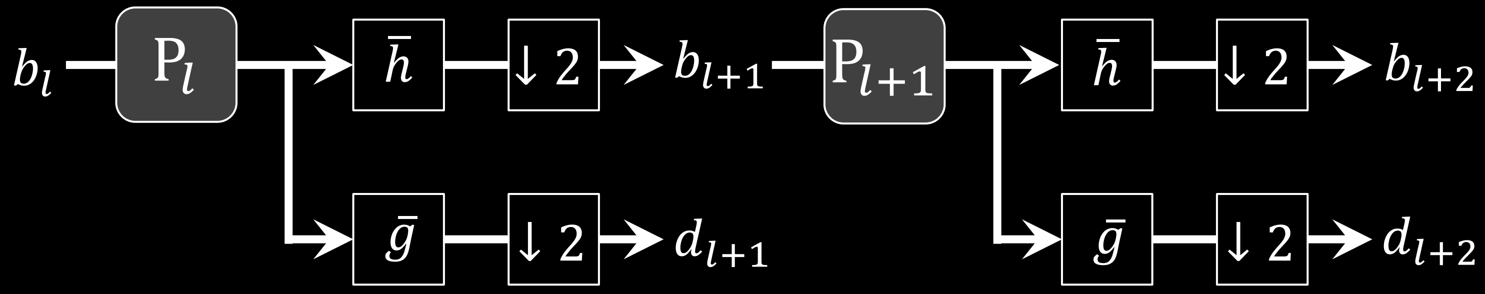


Permutation using $\mathbf{X}=\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_m\}$



Permutation Within the Pyramid

- 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 $\underline{g}$ are the additional permutations, thus preserving the transform's linearity and unitarity, while also adapting to the input signal

Permute to Obtain Maximal Regularity

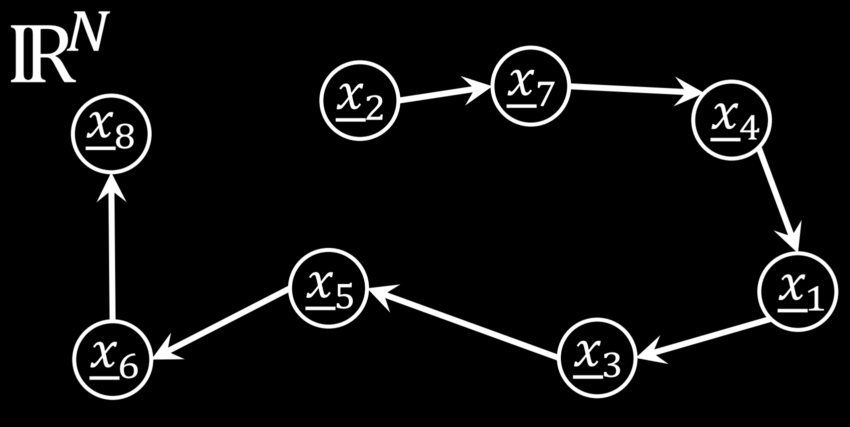
- Lets start with P_0 – the permutation applied on the incoming data
- Recall: for wavelet to be effective, $\mathrm{P}_0\underline{\mathrm{g}}$ should be most "regular"
- **However**: we may be dealing with corrupted signals $\underline{\mathrm{g}}$ (noisy, …)
- To our help comes the feature vectors in $\mathbf{X}$, which reflect on the order of the signal values, g_k. Recall:

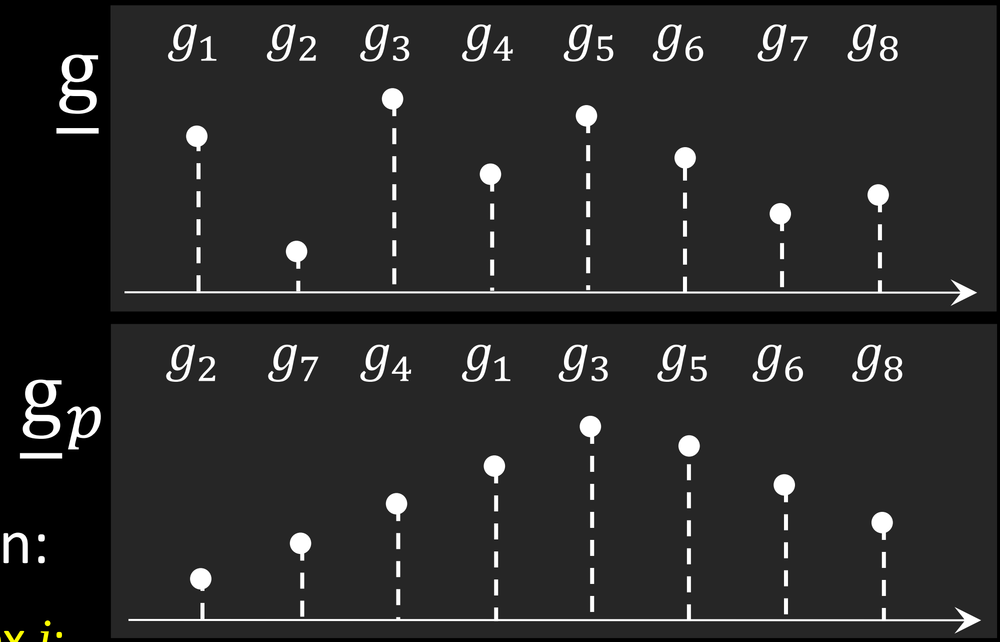
Small $w(x_i, x_j)$ implies small $|g(x_i) - g(x_j)|$ for almost every pair (i, j)

- "Simplifying" $\underline{\mathrm{g}}$ can be done finding the shortest path that visits in each point in X once: the Traveling-Salesman-Problem (TSP):

$$\min_{\mathrm{P}} \sum_{i=2}^{m} |g^p(i) - g^p(i-1)| \;\Rightarrow\; \min_{\mathrm{P}} \sum_{i=2}^{m} w(x_i^p, x_{i-1}^p)$$

Traveling Salesman Problem (TSP)



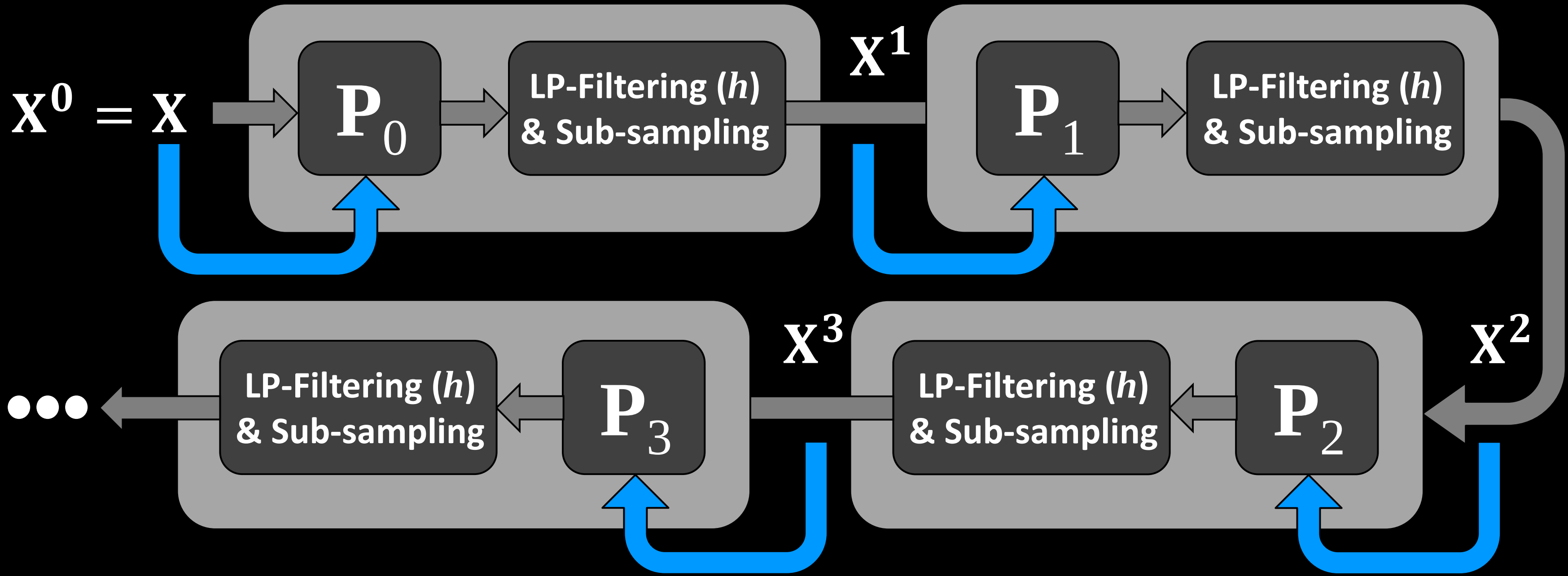


We handle the TSP task by a greedy (and crude) approximation:

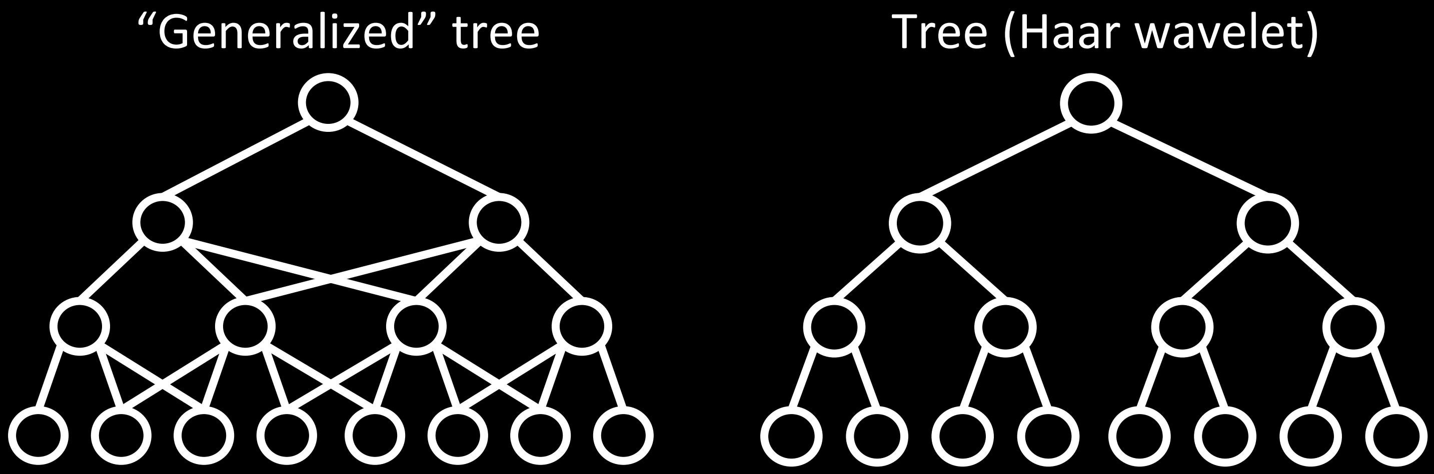
- Initialize with a randomly chosen index j;
- Initialize the set of already chosen indices to $\Omega(1)=\{j\}$;
- Repeat $k=1:1:m\text{-}1$ times:
  - Find $\underline{x}_i$ – the nearest neighbor to $\underline{x}_{\Omega(k)}$ such that $i \notin \Omega$;
  - Set $\Omega(k+1)=\{i\}$;
- Result: the set Ω holds the proposed ordering

What About the Rest of the Permutations?

- 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:



Generalized Tree-Based Wavelet Transform



- 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 should (and could) show how this works on point clouds/graphs, but we will take a different route and discuss implications to image processing

To Summarize So Far

Given a graph or a cloud of points, we can model it in order to process it (denoise, infer, …)

The approach we take is to extend the existing 1D wavelet transform to the graph structure

Our method: Permutation followed by filtering and decimation in each of the pyramid levels

We tested this for graph data with successful results (NOT SHOWN HERE)

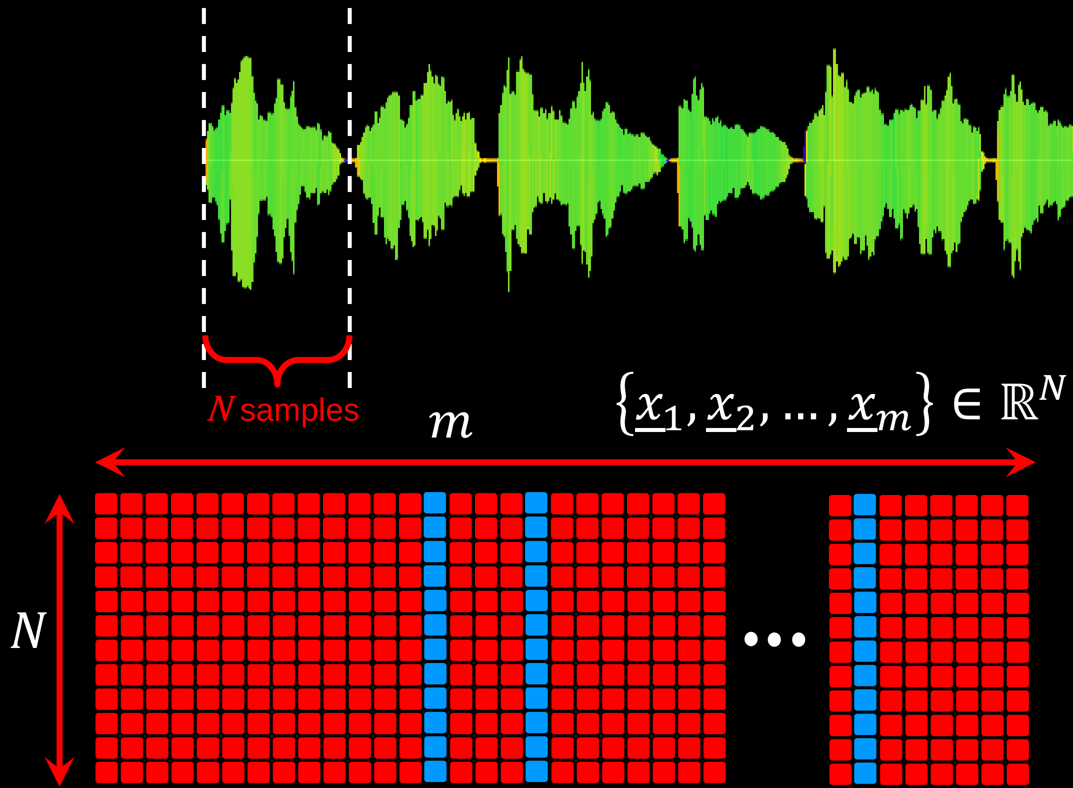
We shall present the applicability of this transform to … images

Turning to
IMAGE PROCESSING

This part relies on 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.

Remember the Guitar Signal?



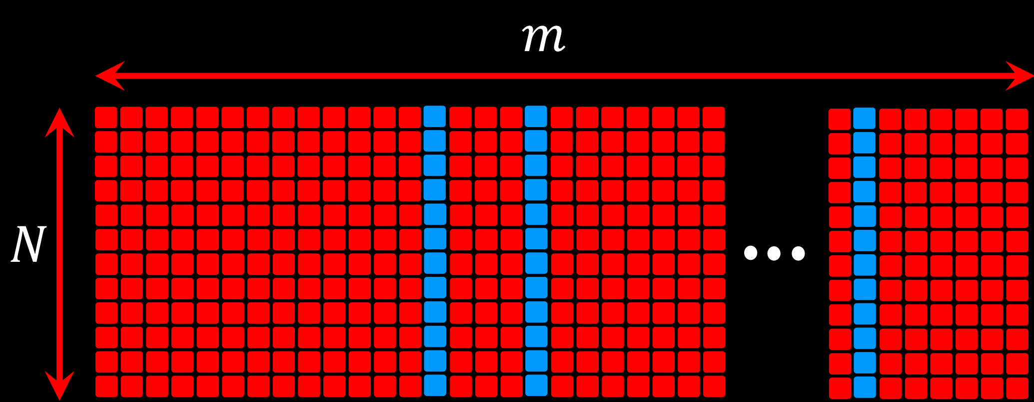
We invested quite an effort to model the columns of this matrix as emerging from a low-dimensional structure

QUESTION:

What about the connection or structure that may exist between these columns?

This brings us to the topic of **GRAPH-STRUCTURED** data modeling

Recall: The Guitar Signal

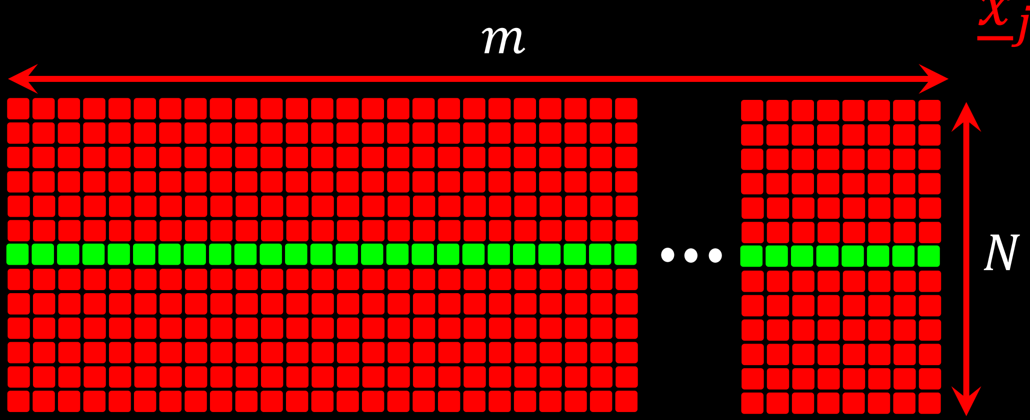


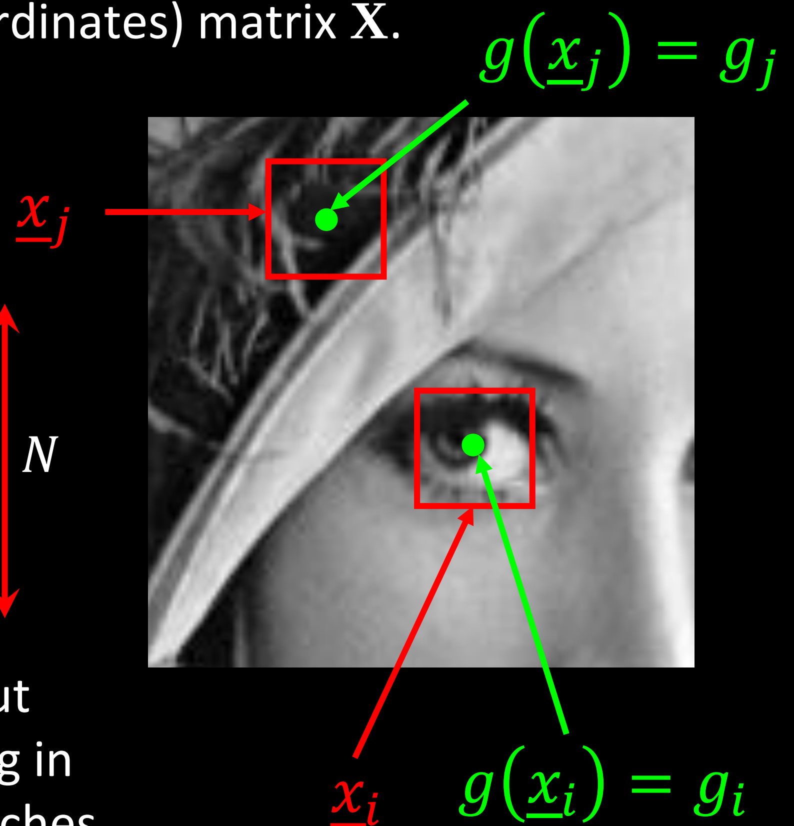
We invested quite an effort to model the columns of this matrix as emerging from a low-dimensional structure

In order to model the inter-block (rows) redundancy, we can consider this matrix as containing the feature vectors of graph nodes, and apply the designed sparsifying wavelet

An Image as a Graph

- Extract all possible patches of size $\sqrt{N} \times \sqrt{N}$ with complete overlaps – these will serve as the set of features (or coordinates) matrix $\mathbf{X}$.
- The values $g(\underline{x}_i) = g_i$ will be the center pixel in these patches.





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

Our Transform

Lexicographic ordering of the m pixels

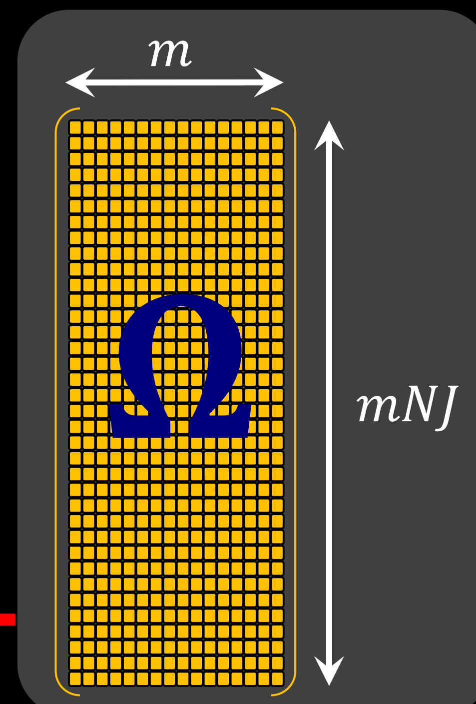
$\underline{g}$

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

Applying a J redundant wavelet of some sort with permutations

We obtain an array of m$\text{N}J$ transform coefficients

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

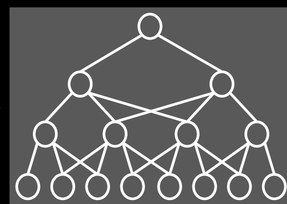


Lets Test It: M-Term Approximation

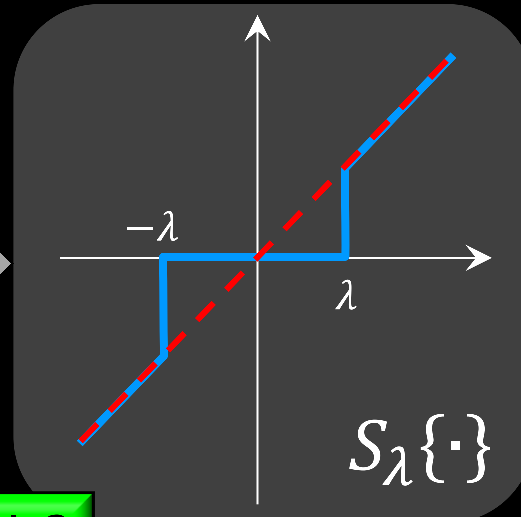
Original Image

$\underline{g}$

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



$\mathbf{\Omega}\underline{g}$

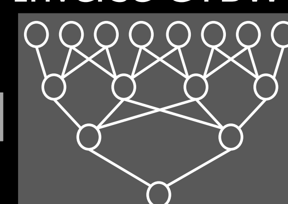


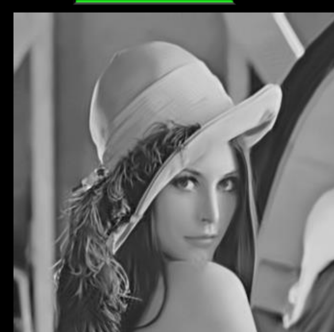
Skip?

$S_\lambda\left\{\mathbf{\Omega}\underline{g}\right\}$

M non-zeros

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





Output image

$\hat{\underline{g}}$

Show

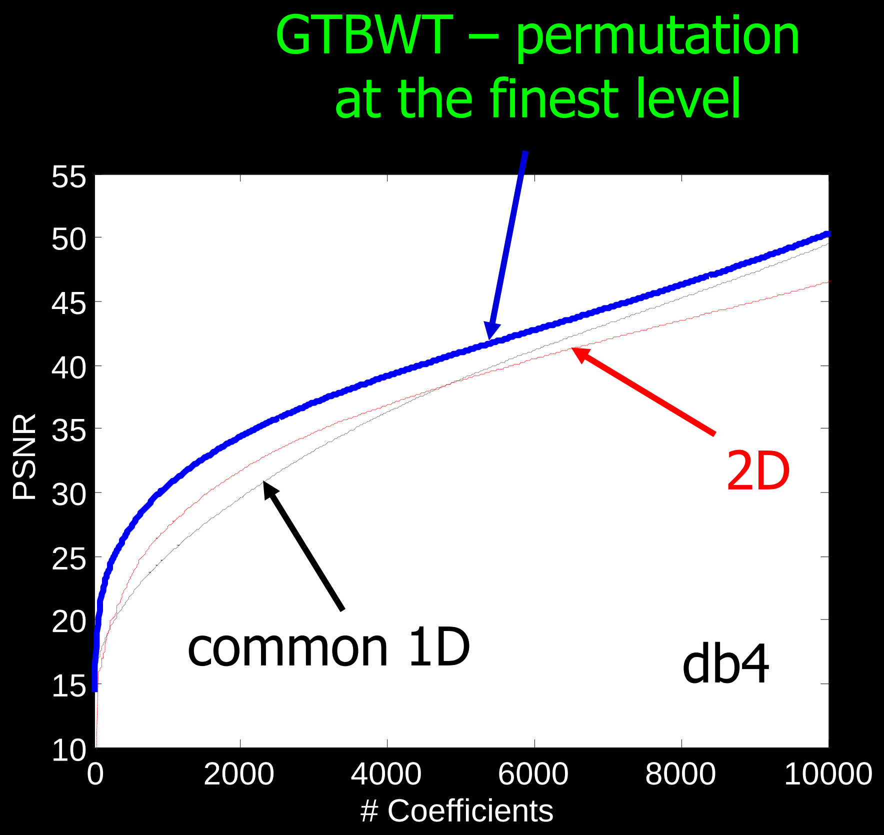
$$\left\|\underline{g}-\hat{\underline{g}}\right\|^2=\left\|\underline{g}-\mathbf{D}S_\lambda\left\{\mathbf{\Omega}\underline{g}\right\}\right\|^2$$

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

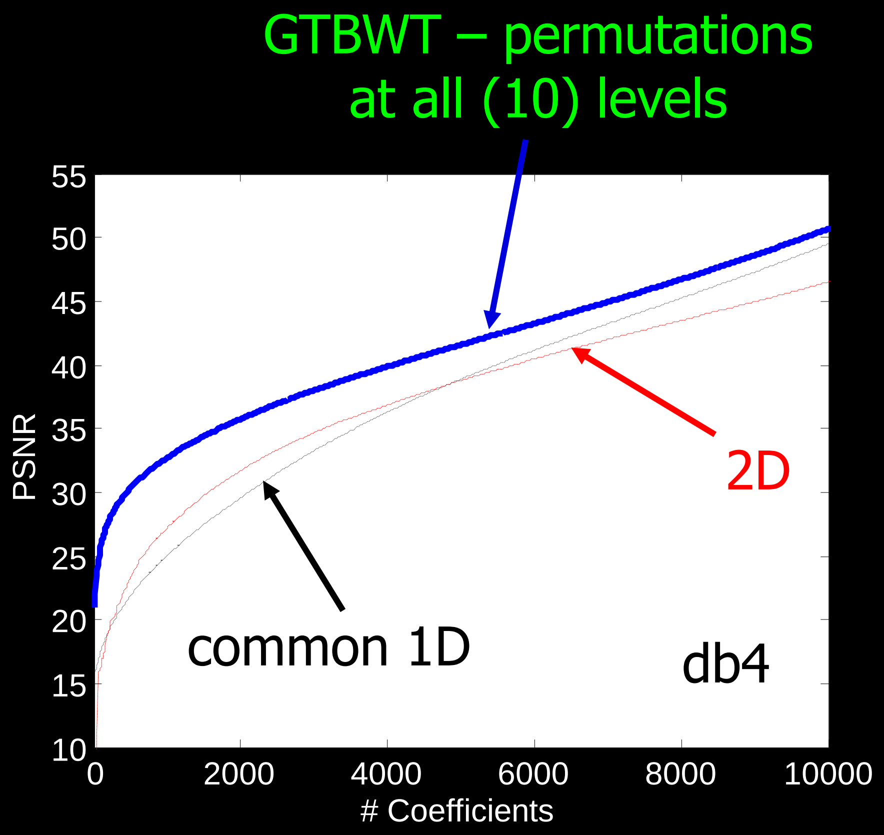
- ❑ GTBWT
- ❑ A common 1D wavelet transform
- ❑ 2D wavelet transform



Lets Test It: 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

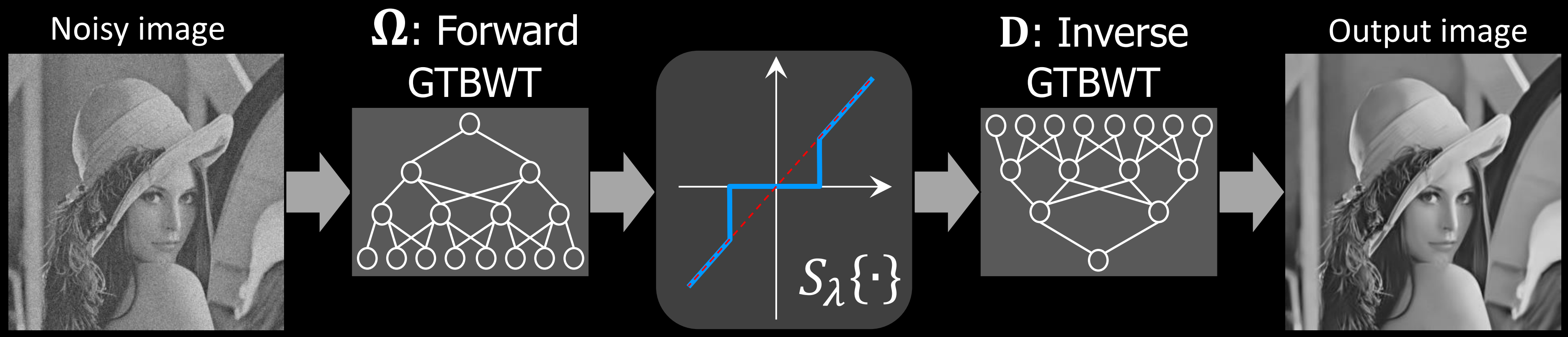


Lets Test It: Denoising Via Sparsity ($\underline{y} = \underline{x} + \underline{z}$)

$\underline{y}$ → **Denoising Algorithm** → $\hat{\underline{x}}$

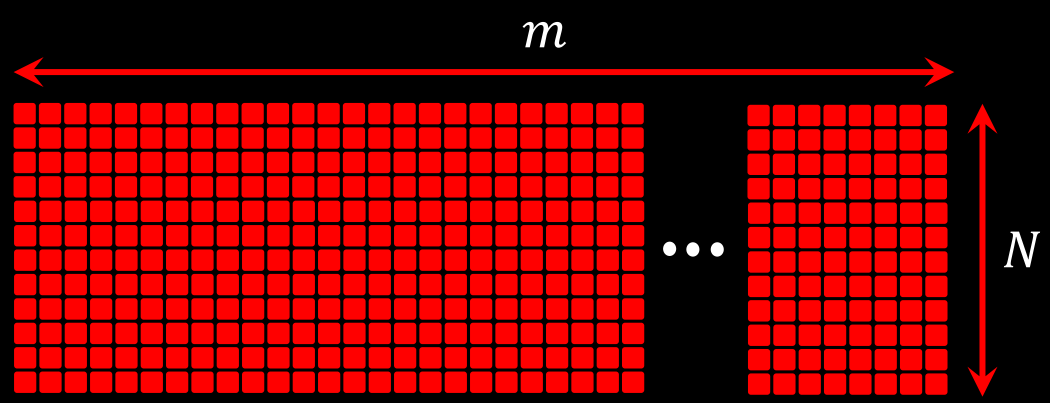
Approximation by the THR algorithm:

$$\hat{\underline{x}} = \mathbf{D}S_\lambda\left\{\mathbf{\Omega}\underline{y}\right\}$$



Wait!

Lets Not Do the Same Mistake Twice



Given this matrix containing all the image patches we agreed that we should exploit both columns and rows' redundancies

Using only the GTBWT will operate on rows, wasting the redundancies within the columns

We apply the GTBWT on the rows of this matrix, and take further steps (sub-image averaging, joint-sparsity) in order to address the within-columns redundancy as well

Image Denoising – Results

We apply the proposed scheme with the Symmlet 8 wavelet to noisy versions of the images Lena and Barbara, and compare to K-SVD & BM3D algorithms.

Original Noisy Denoised

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**

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

This part is based on the following papers:

- I. Ram, M. Elad, and I. Cohen, "The RTBWT Frame – Theory and Use for Images", working draft to be submitted soon.
- I. Ram, M. Elad, and I. Cohen, "Image Processing using Smooth Ordering of its Patches", to appear in IEEE Transactions on Image Processing.

Recall: Our Transform

Lexicographic ordering of the m pixels

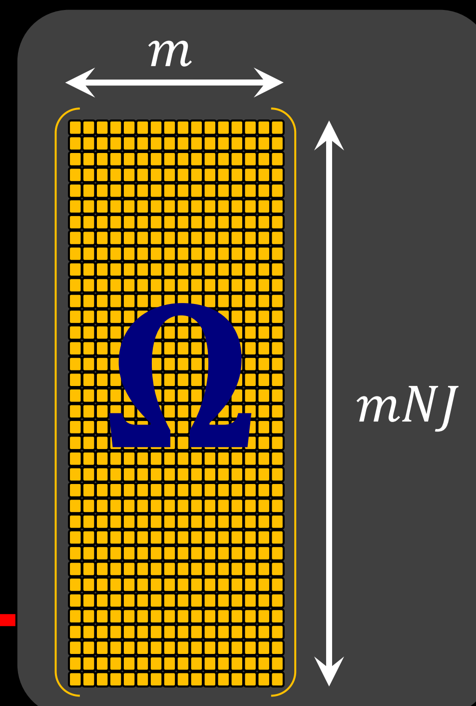
$\underline{g}$

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

Applying a J redundant wavelet of some sort with permutations

We obtain an array of mNJ transform coefficients

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



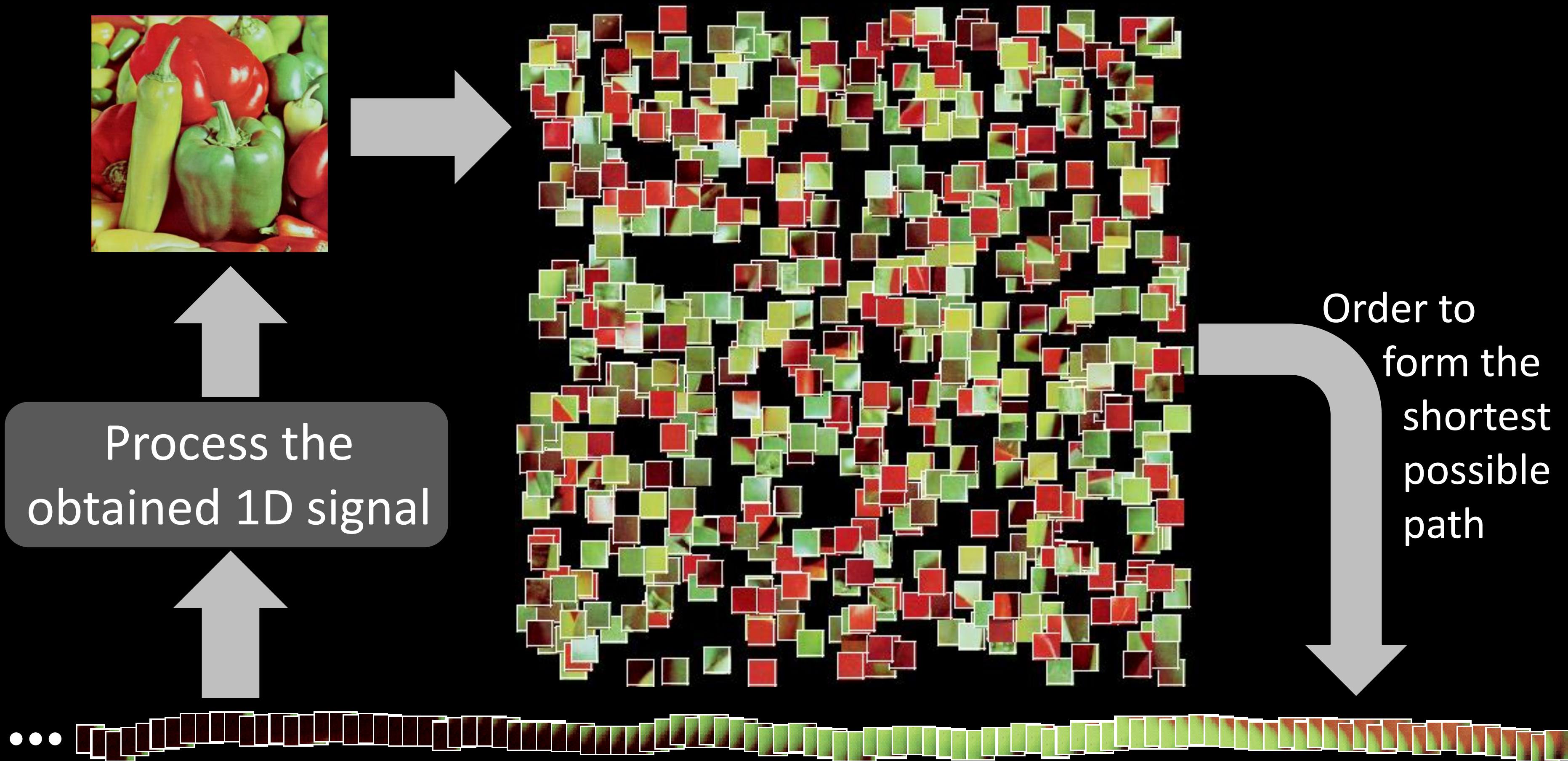
A: What Can We Do With Ω ?

Ω (and $\mathbf{D}$) is the core of a sparsity-based model for **THE** image

- Deblurring
- Anomaly Detection
- Separation
- Super-Resolution
- Denoising
- Inpainting
- …
- …
- Sampling: Compressed-Sensing
- Compression
- Tomographic Reconstruction

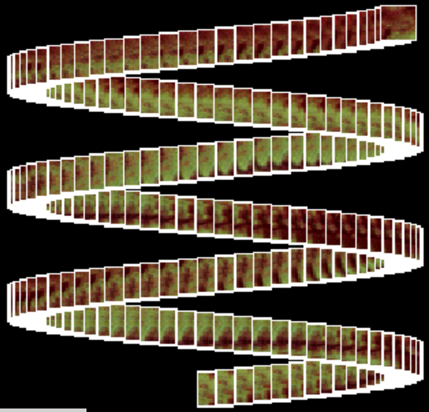
A: E.g. Deblurring Results

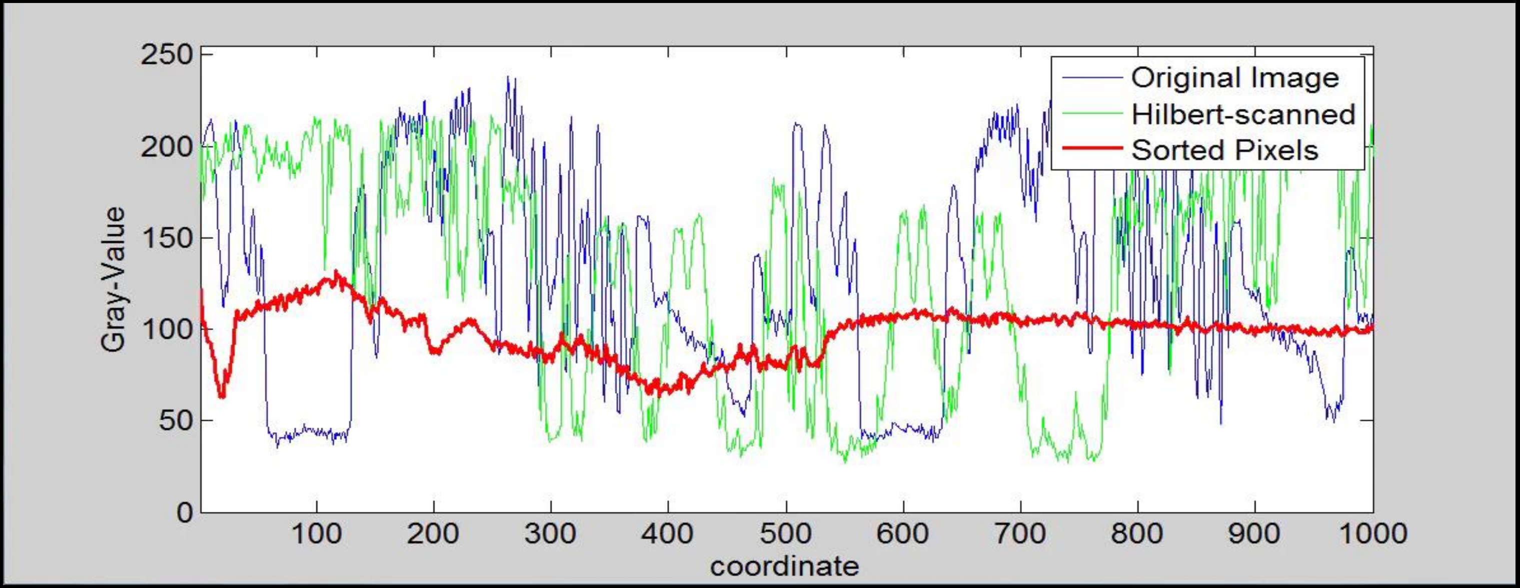
B: Alternative: Ordering the Patches



Key Idea: Regularity Due to Ordering

- Considering the center (or any other) pixel in each patch, 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

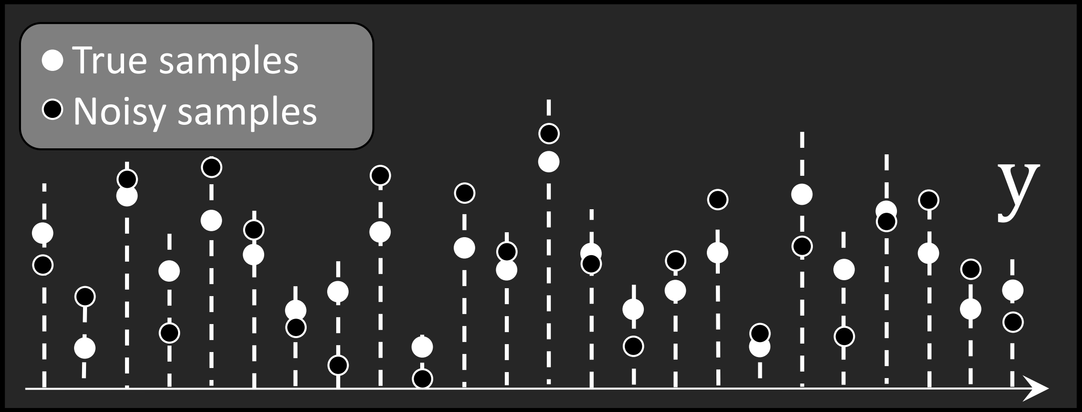




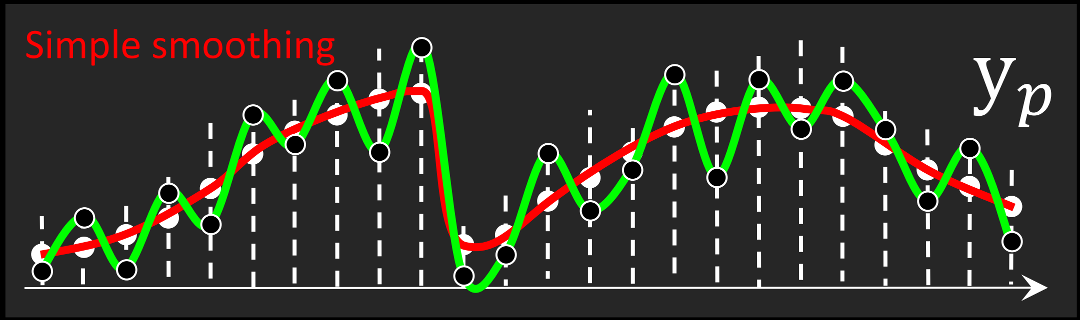
B: Image Denoising with Patch-Reordering

Noisy with σ=25 (20.18dB)

Reconstruction: 32.65dB

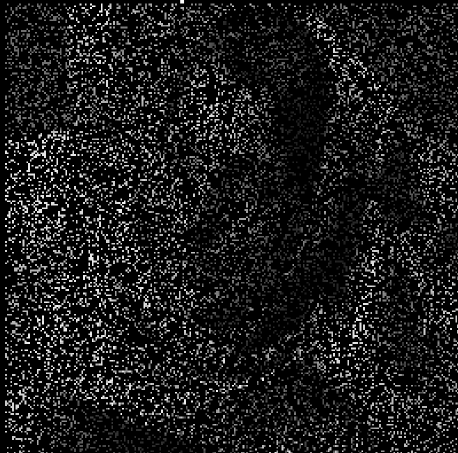


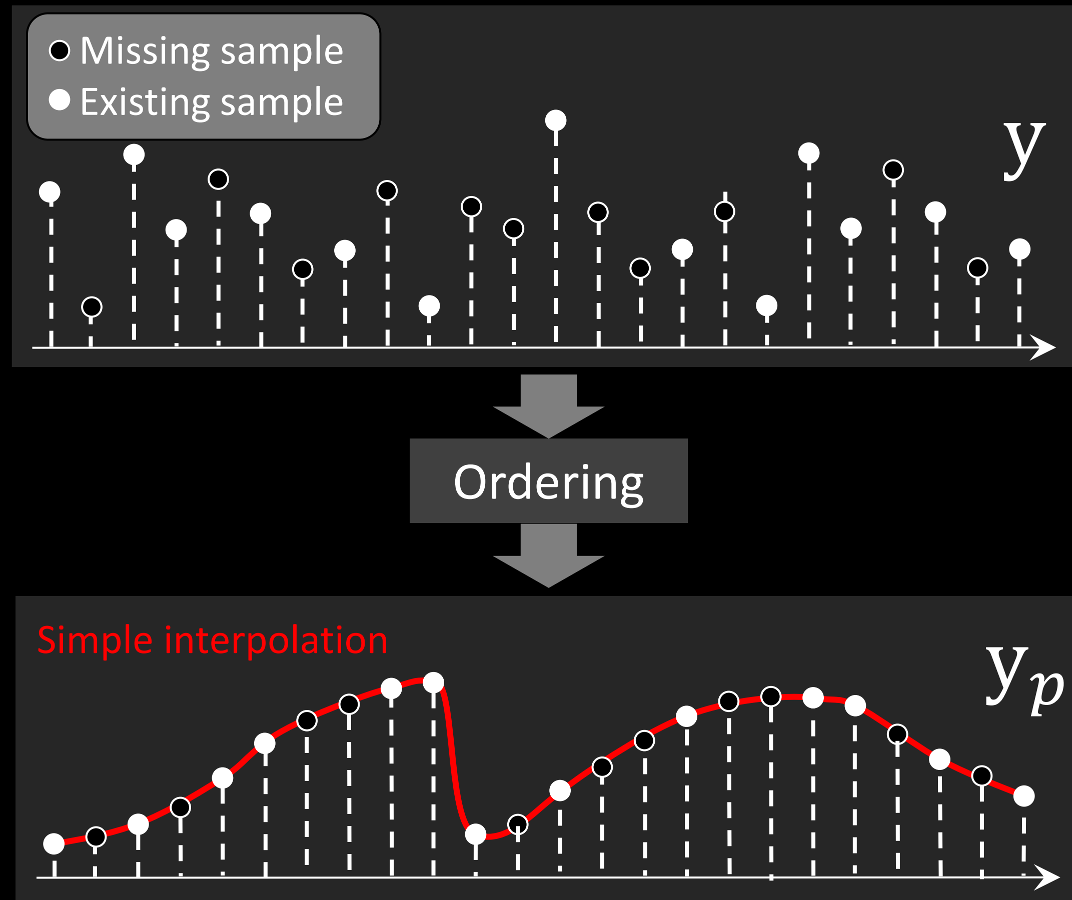
Ordering based on the noisy pixels



B: Image Inpainting with Patch-Reordering

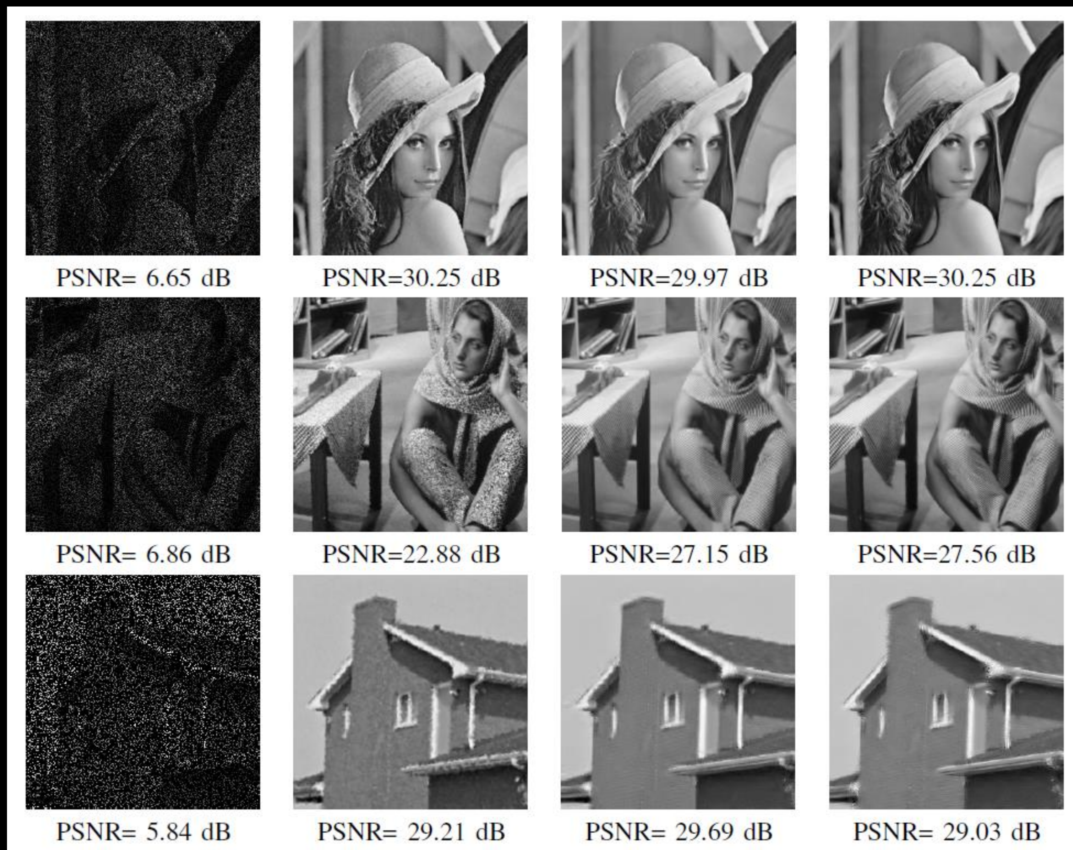
0.8 of the pixels are missing





Reconstruction: 27.15dB

B: Inpainting Results – Examples



Given data 80% missing pixels	Bi-Cubic interpolation	DCT and OMP recovery	1^{st} iteration of the proposed alg.

Time To Finish

Conclusions

Processing data is enabled by an appropriate modeling that can expose its inner structure

Sparsity-based models are highly effective and lead to state-of-the art processing in many disciplines

We have shown how sparsity becomes applicable also for graph structured data

We have shown how classical image processing tasks can benefit from the new construction

What next? Processing graph data, different patch-embedding, learned dictionaries, lifting scheme, ….

These slides can be found in http://www.cs.technion.ac.il/~elad

Thank you for your time

and …

Thank you to the organizers of this event:

Kenji Fukumizu and Kei Kobayashi

Institute of Statistical Mathematics

Yoshiyuki Kabashima and Taiji Suzuki

Tokyo Institute of Technology

Ryota Tomioka

Toyota Technological Institute, Chicago

for inviting me

Questions?