Image Processing Via Pixel Permutations

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

Brief Introduction

In this talk we revisit some of the

MOST BASIC IDEAS IN IMAGE PROCESSING

I will try to convince you that even there,

core concepts that seem fixed and settled

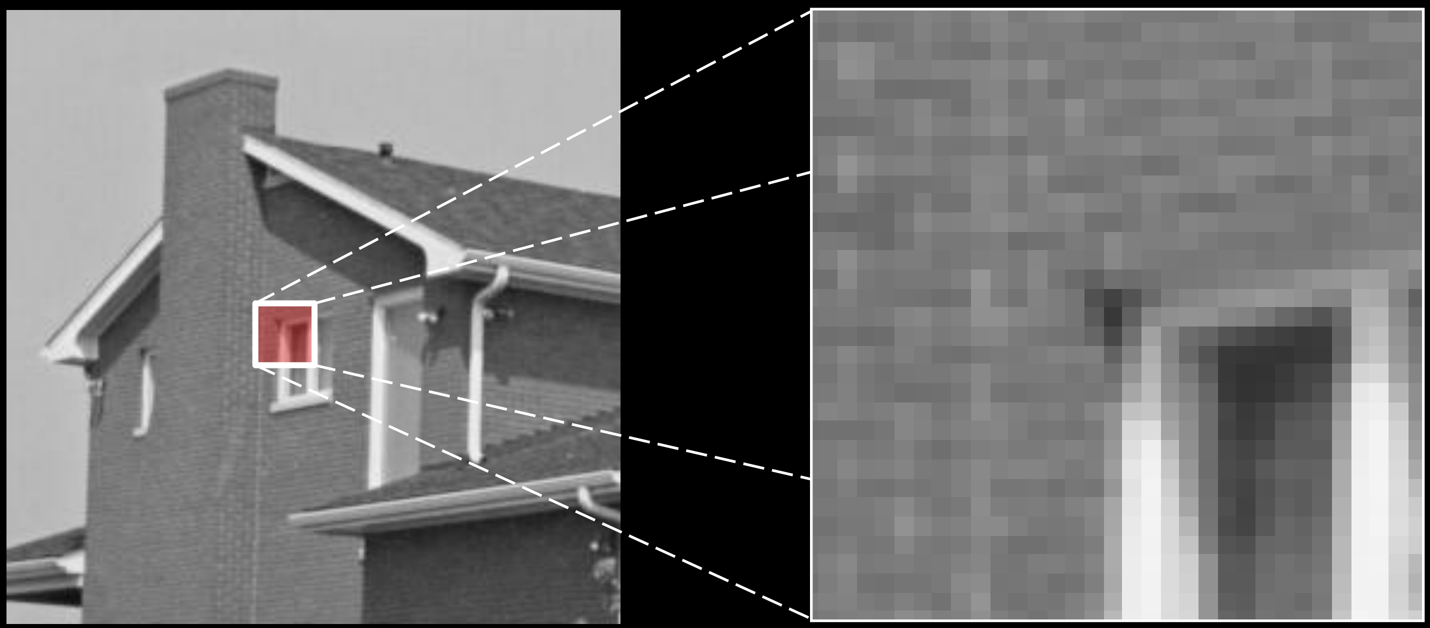
MIGHT BE QUESTIONED

AND APPROACHED IN A NEW WAY

Part I

2D $\rightarrow$ 1D Conversion

Here is an Image …



- An image is a 2D signal.
- As such, there is no sense of causality between the pixels.

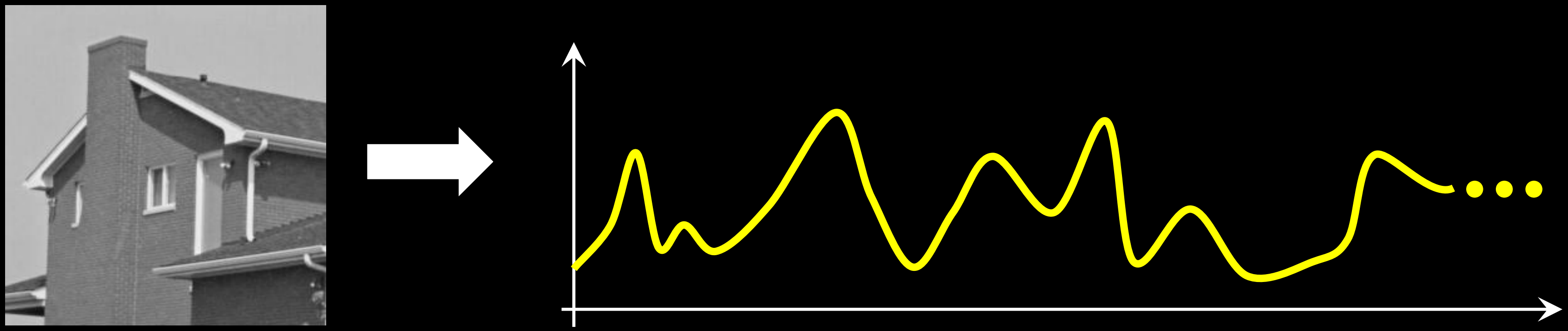
We are Interested in Image Processing

In this talk, we focus on the need to process such images, performing tasks such as :

- ❑ Noise removal.
- ❑ Filling-in missing values.
- ❑ Restoration (deblurring, super-resolution).
- ❑ Reconstruction (e.g. Tomography).
- ❑ Dithering.
- ❑ Compression.
- ❑ …

2D $\longrightarrow$ 1D Conversion ?

Often times, a proposed solution to any of the above tasks starts by 2D to 1D conversion :

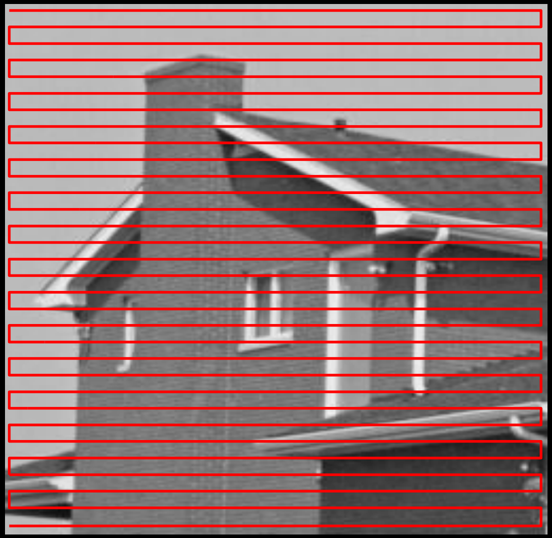


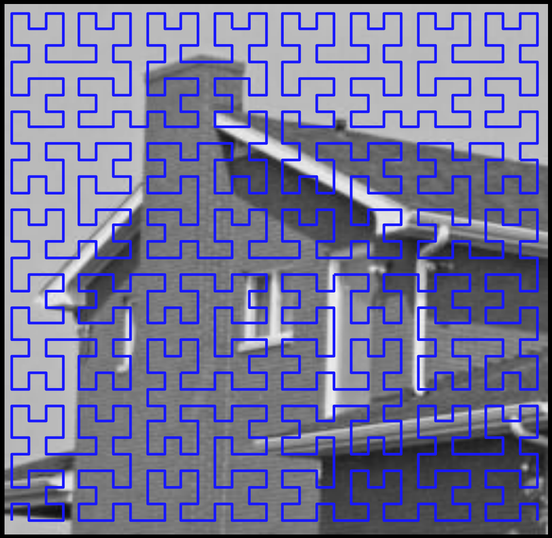
After such a conversion, the image is treated as a regular 1D signal, with implied sampled order and causality.

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 ⟶ 1D : Why Convert ?

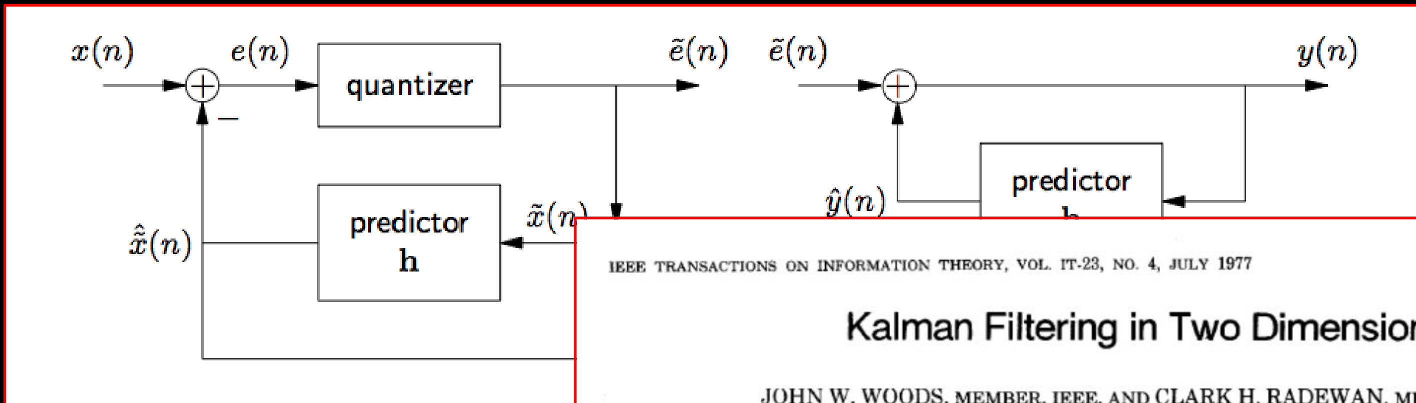
The scientific literature on image processing is loaded with such conversions, and the reasons are many:

- ❑ 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 considerations.
- ❑ Because of run-time considerations.

Note that it is never claimed that 2D → 1D would lead to improved performance in terms of output quality.

2D $\longrightarrow$ 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 $\longrightarrow$ 1D trend is an "old-fashion" trick, it is still very much active and popular in industry and academic work.

2D → 1D : Is It a Good Idea ?

- If anyone proposes a solution to an image processing problem that starts by a 2D → 1D conversion, we should immediately think:

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

- The reasons for this belief are obvious:
  - Loss of neighborhood relations.
  - Unnatural causality.
- So, we conclude that this approach is not a good idea !!

ARE WE SURE ?

Part II

Our Core Idea

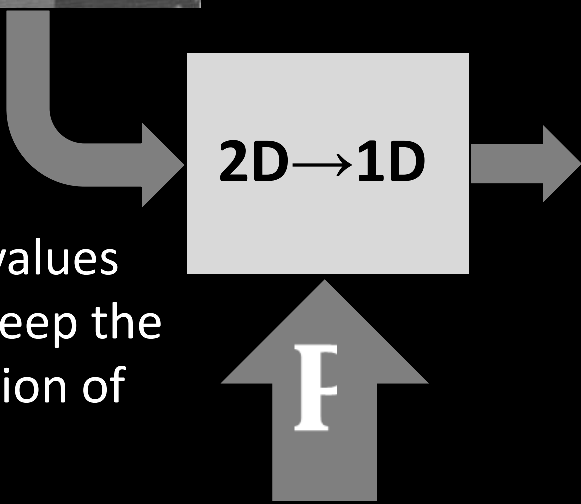
Joint work with

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

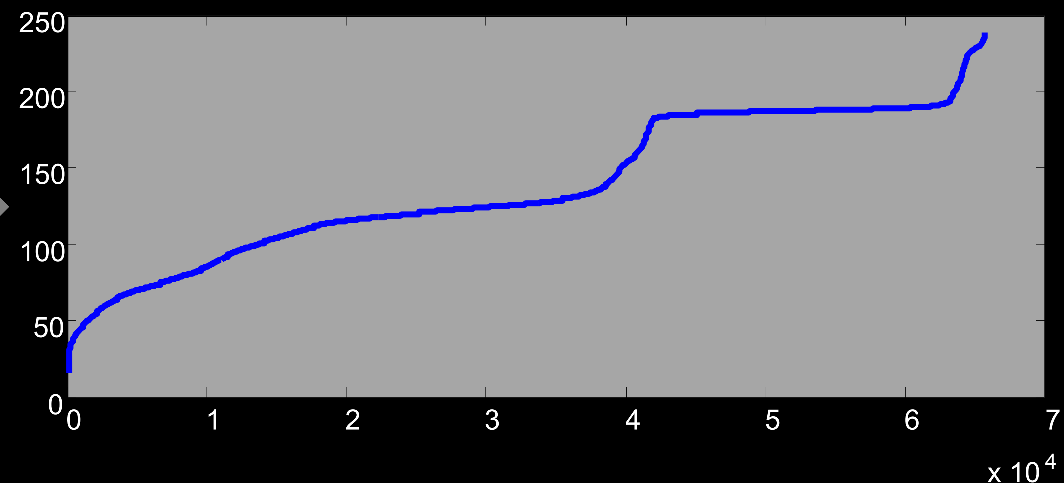
Lets Propose a New 2D → 1D Conversion

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

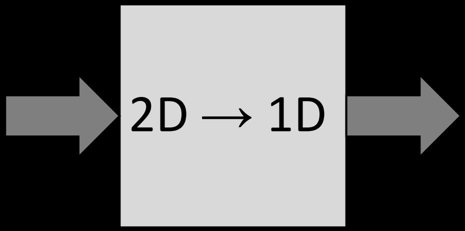
SORT OPERATION ?

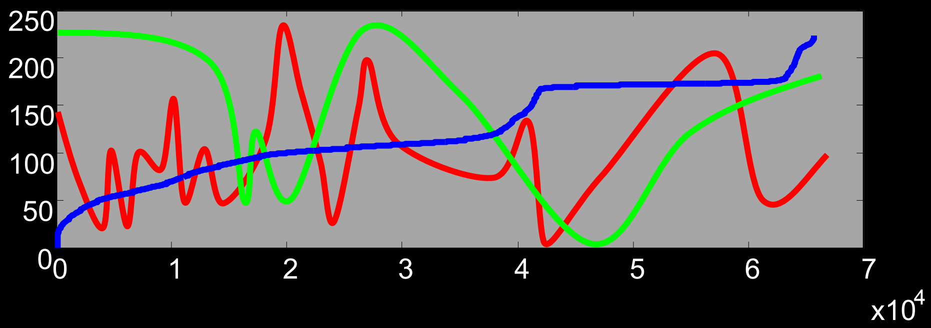


We sort the gray-values but also keep the [x,y] location of each



New 2D → 1D Conversion : Smoothness



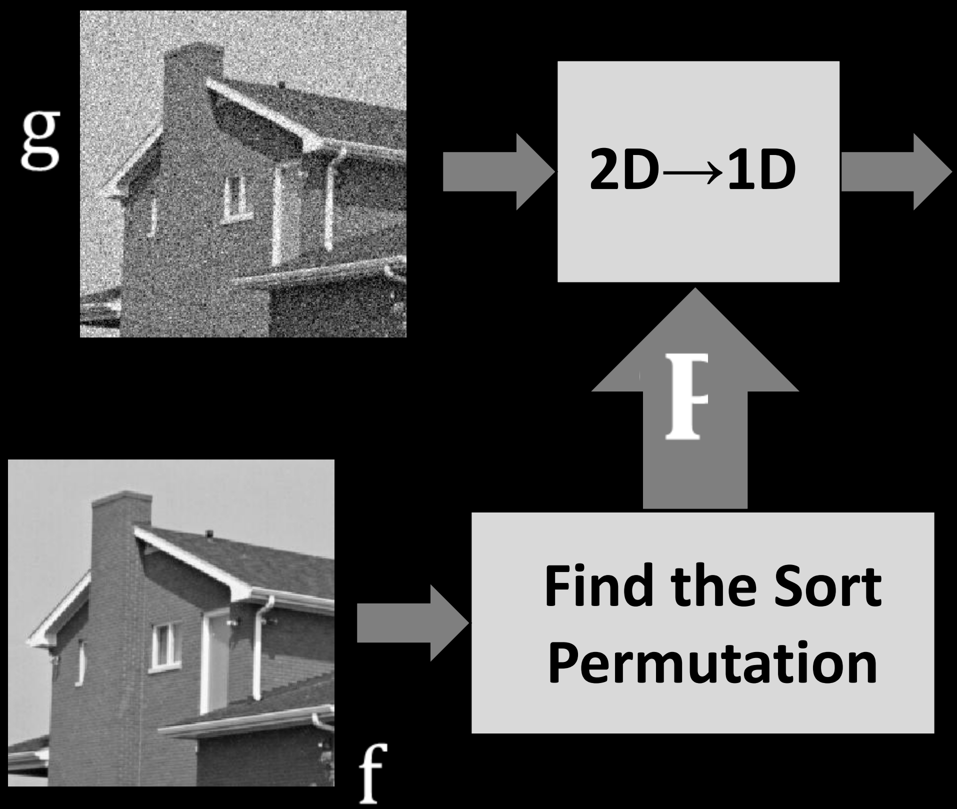


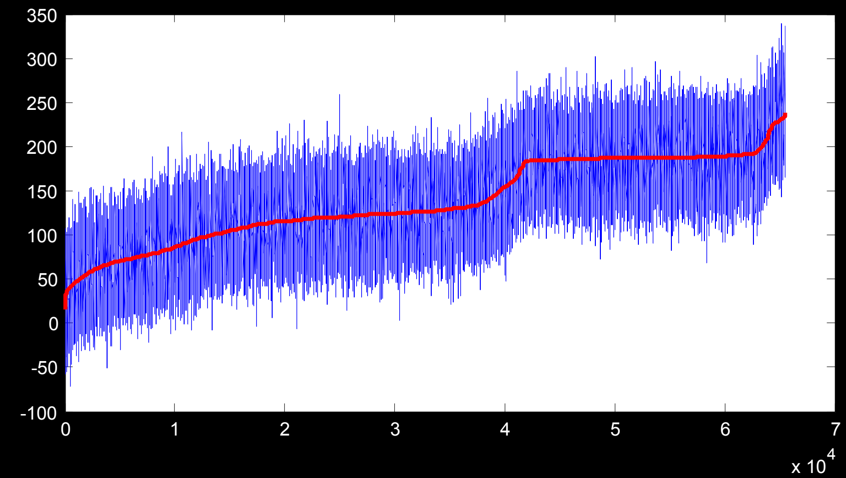
- Given any 2D → 1D conversion based on a permutation $\mathbf{P}$, we may ask how smooth is the resulting 1D signal obtained :

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

- 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.

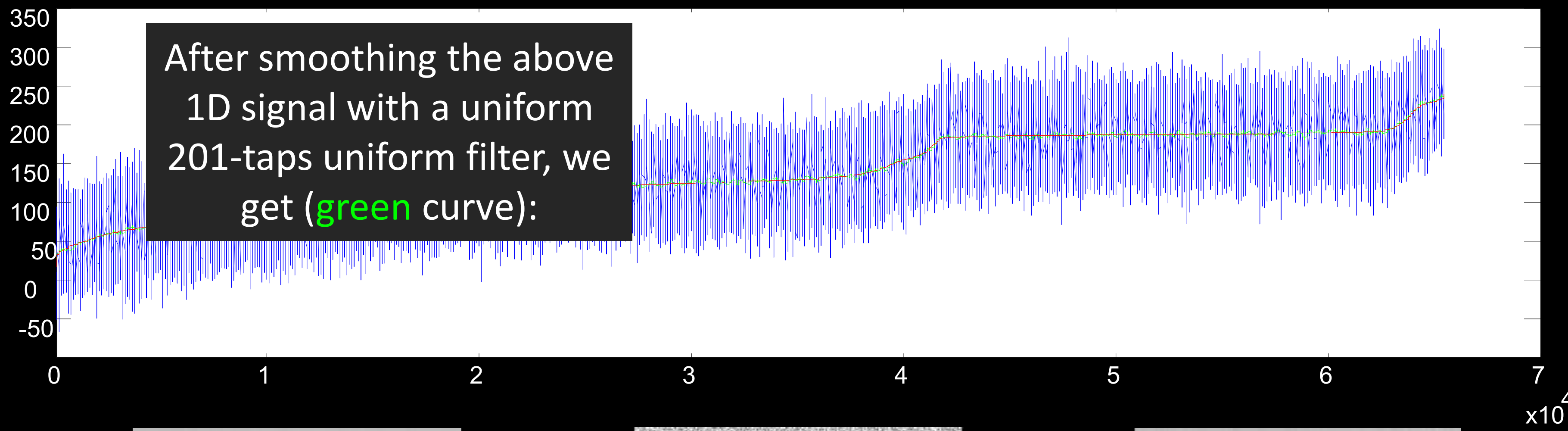
New 2D ⟶ 1D Conversion : An Example





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 

g 

$\hat{f}$ 

Original

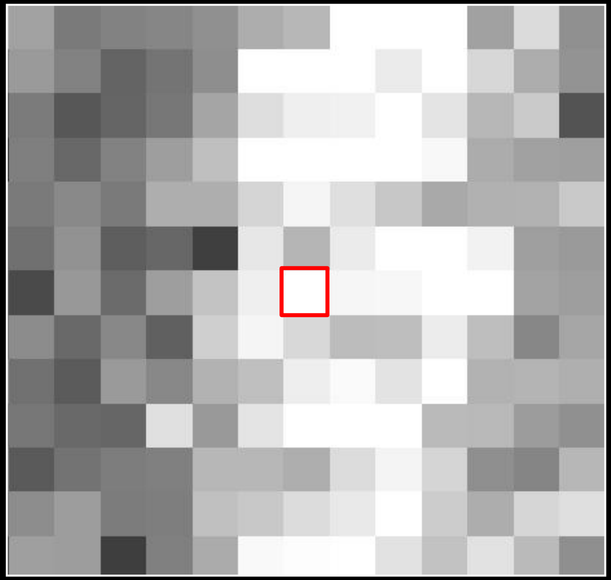
Noisy σ=30 (18.58dB)

Denoised (41.7dB)

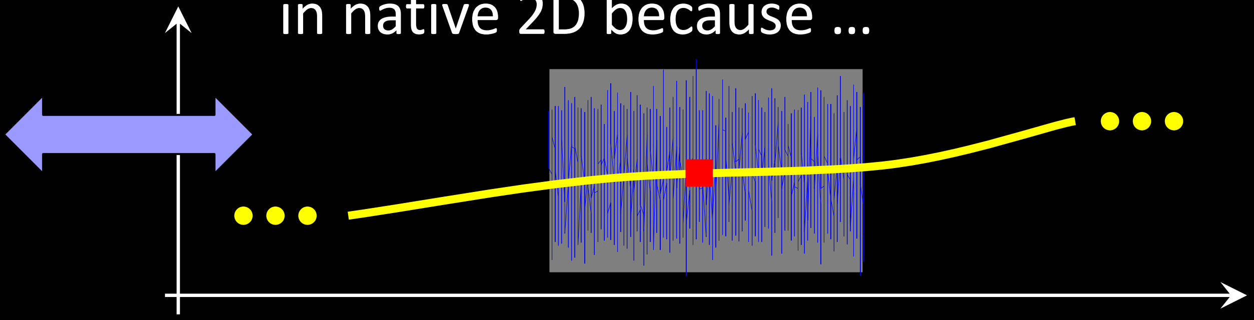
This is Just Great! Isn't It?

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 …



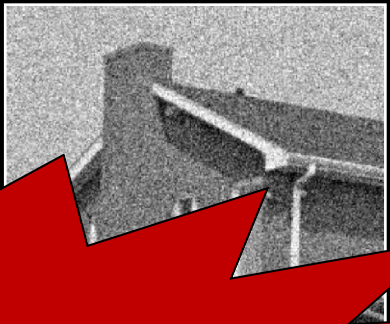
This is Just Great! Isn't It?

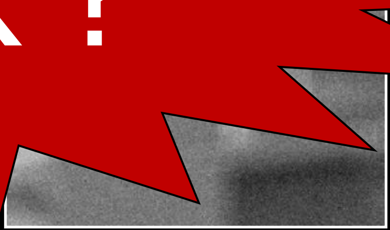
All this is wonderful … but …

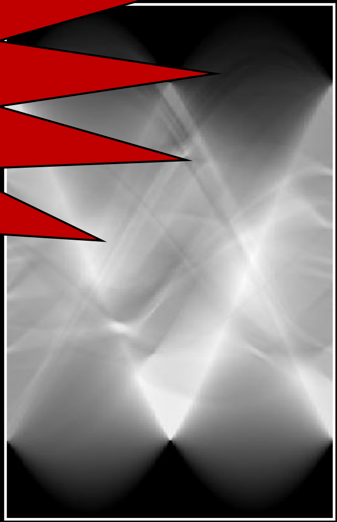
Given corr

SORTI

P







So, Are We Stuck ?

So the above result is impractical.

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

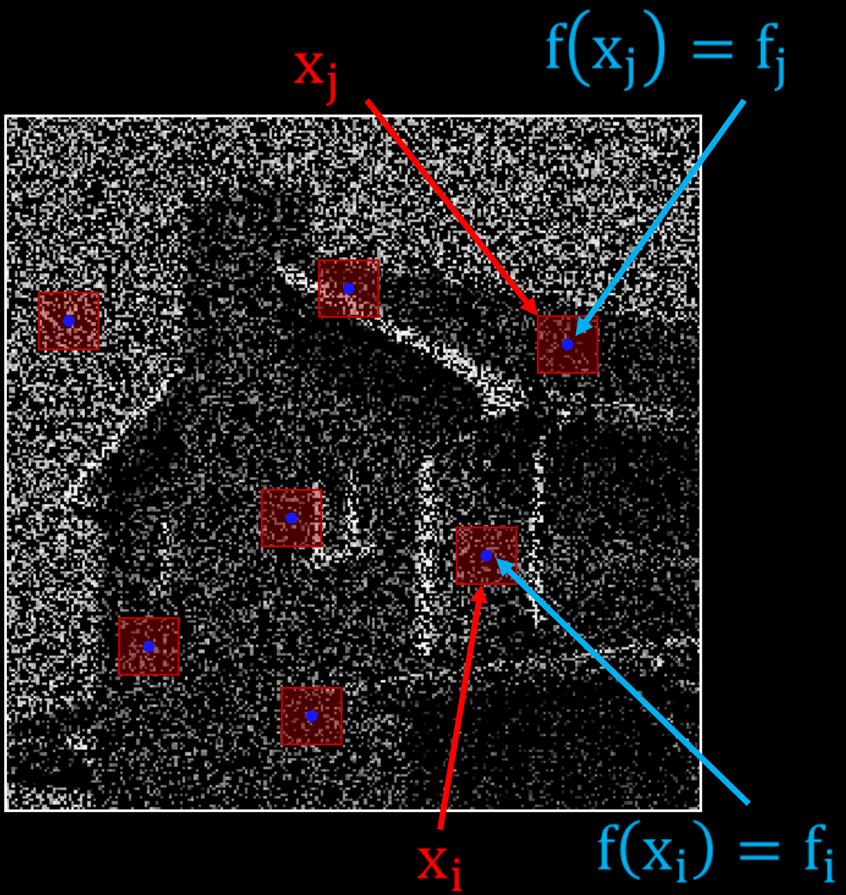
Our Goal – Sorting the pixels based on their TRUE gray value

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

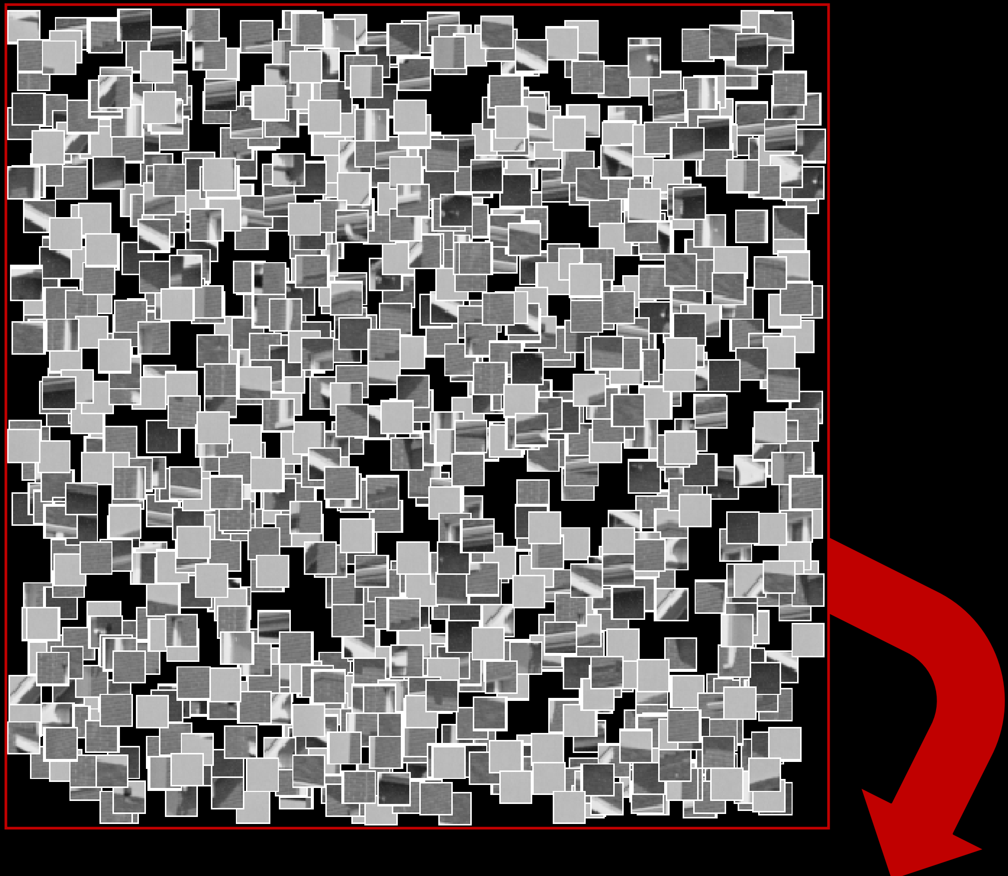
We will design $\mathbf{P}$ based on these feature vectors

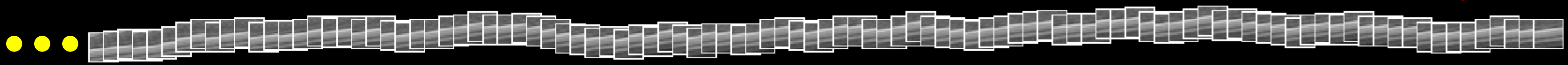


An Alternative for Constructing $\mathbf{P}$

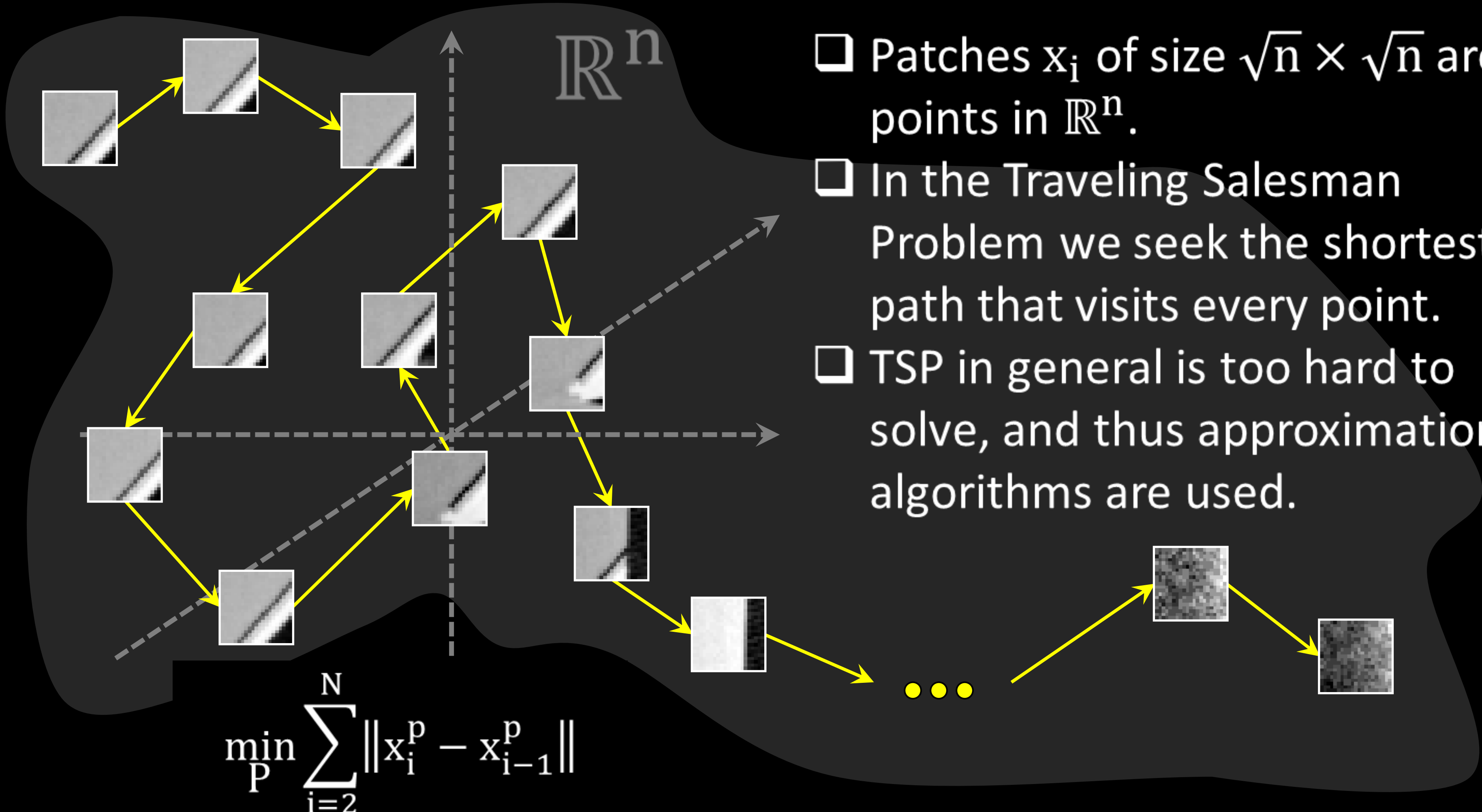
We will construct $\mathbf{P}$ by the following stages:

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)



- 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 $\mathbf{P}$?

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 $\mathbf{P}$**, 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.

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

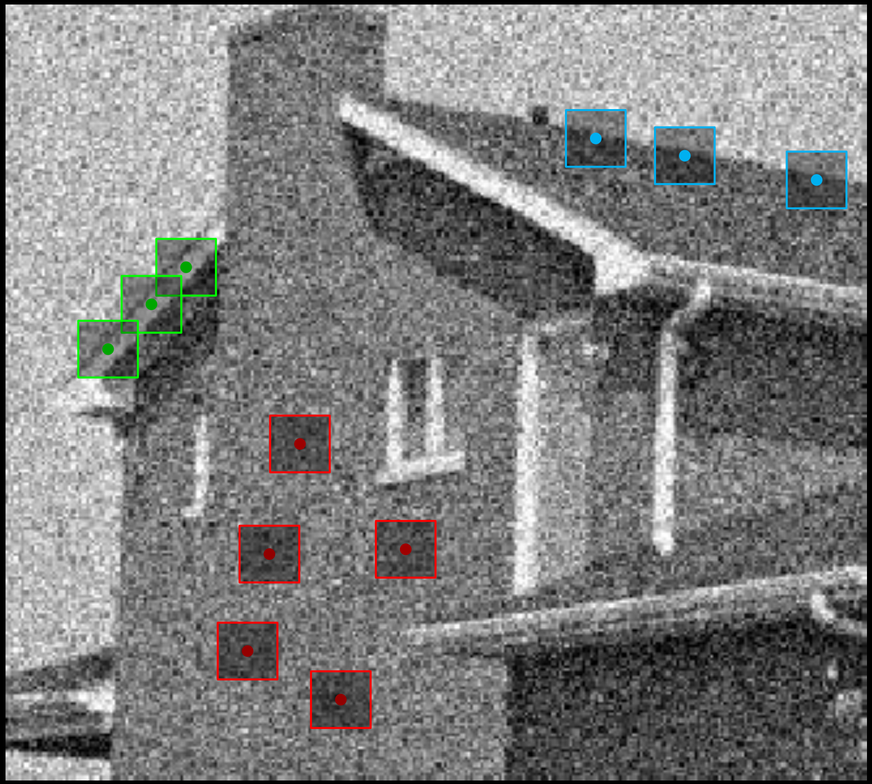
The Proposed Alternative : A Closer Look

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.

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

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

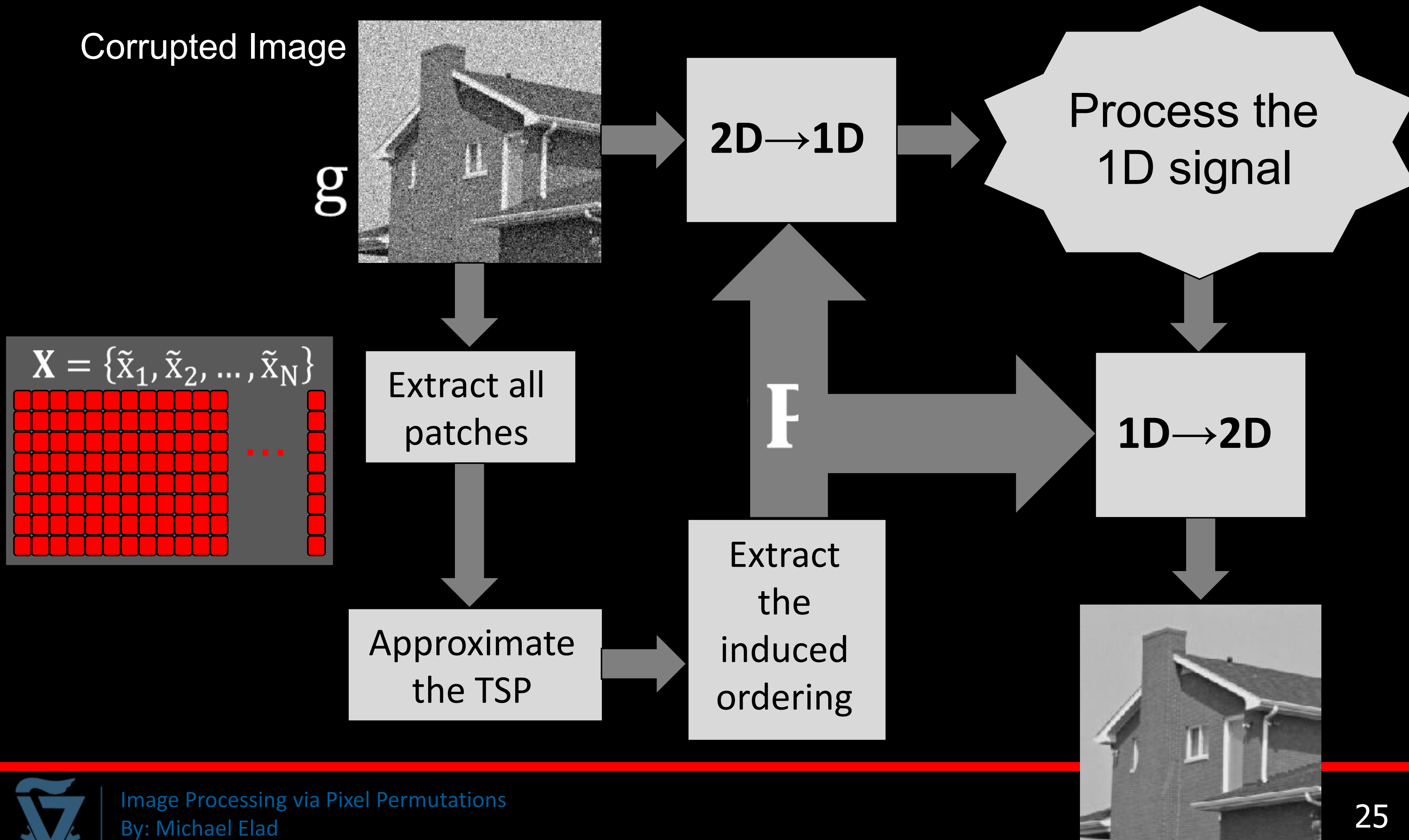


The order is similar, not necessarily the distances themselves

Part III

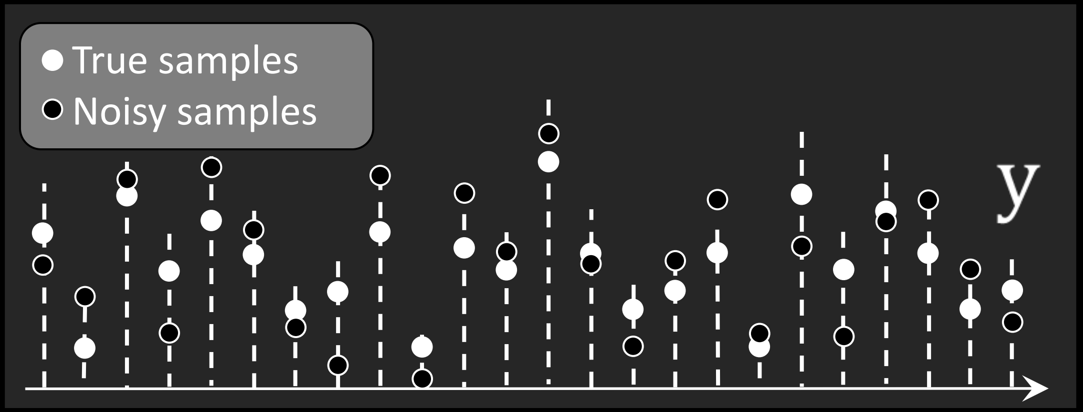
Image Denoising & Inpainting

The Core Scheme

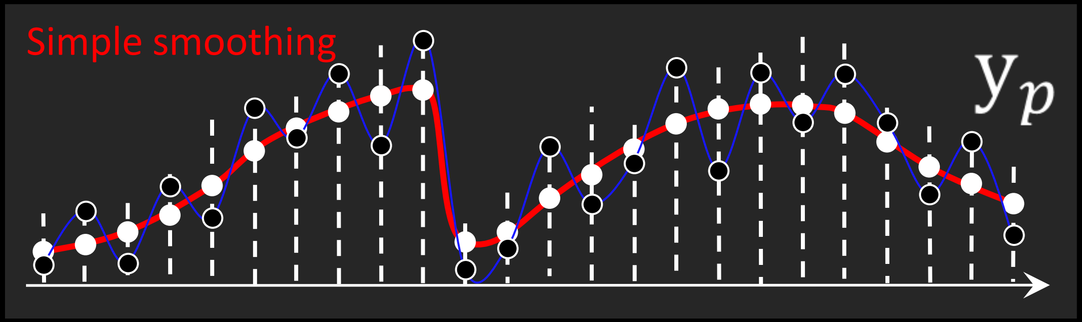


Intuition: Why Should This Work?

Noisy with σ=25 (20.18dB)



Ordering based on the noisy pixels



Reconstruction: 32.65dB

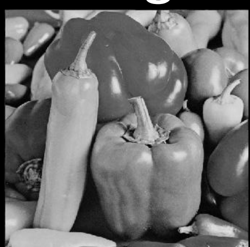
The “Simple Smoothing” We Do

Simple smoothing works fine

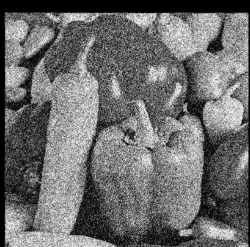
but

We can do better by a training phase

Training image



Original image



Noisy image

Compute the TSP permutation

Apply the permutation on the pixels

Apply the permutation on the pixels

Apply a 1D filter h

+

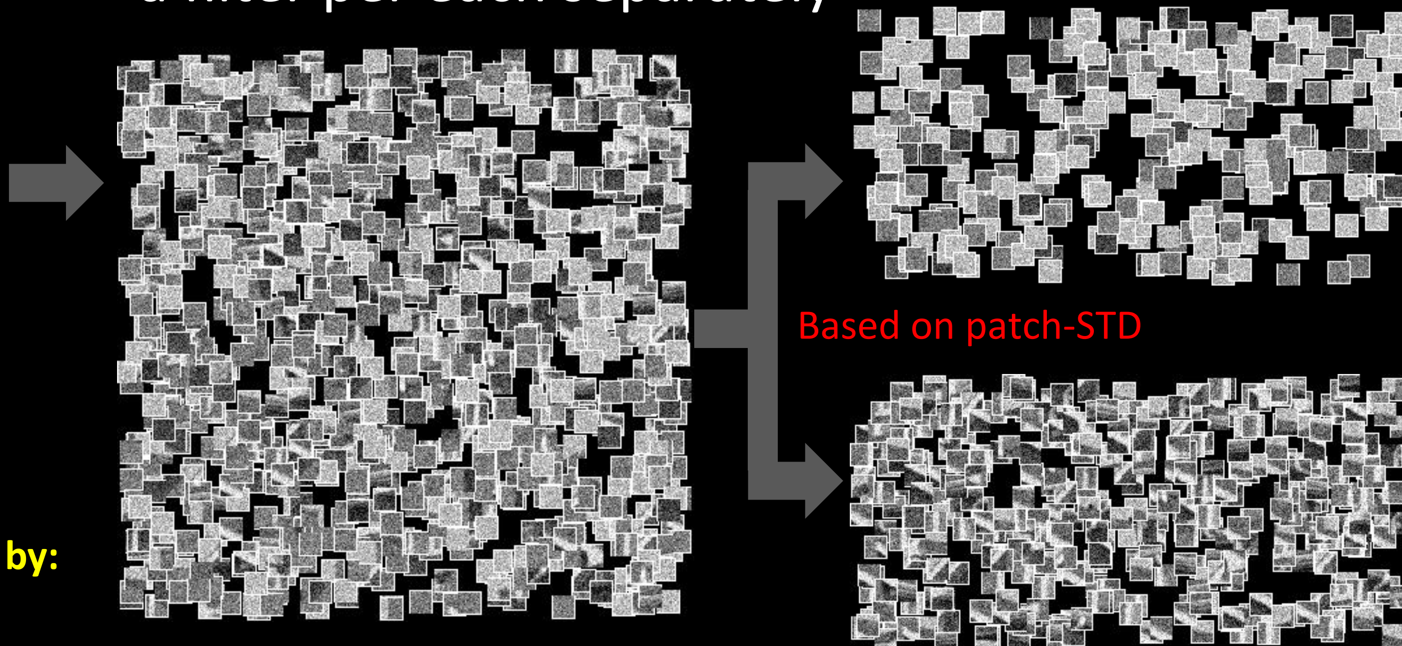
-

optimize h to minimize the reconstruction MSE

Naturally, this is done off-line and on other images

Filtering – A Further Improvement

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



The results we show hereafter were obtained by:

(i) Cycle-spinning
(ii) Sub-image averaging
(iii) Two iterations
(iv) Learning the filter, and
(v) Switched smoothing.

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
	1^{st} iteration	35.33	31.58	28.54
	2^{nd} iteration	35.41	**31.81**	**29.00**
Barbara	K-SVD	34.41	29.53	25.40
	1^{st} iteration	**34.48**	30.46	27.17
	2^{nd} iteration	34.46	**30.54**	**27.45**
House	K-SVD	**36.00**	32.12	28.15
	1^{st} iteration	35.58	32.48	29.37
	2^{nd} iteration	35.94	**32.65**	**29.93**

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

What About Inpainting?

0.8 of the pixels are missing

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

Order these patches as before distance uses EXISTING pixels only

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

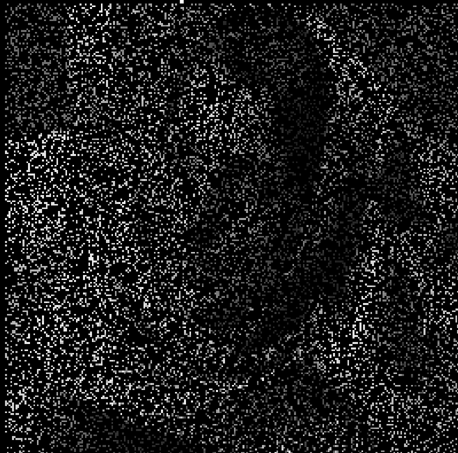
Fill the missing values in a simple (cubic interpolation) way

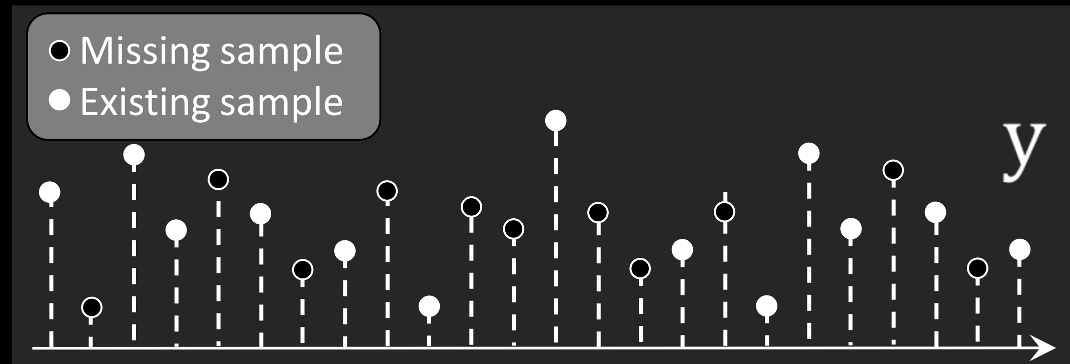
Reconstruction: 29.71dB*

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

The Rationale

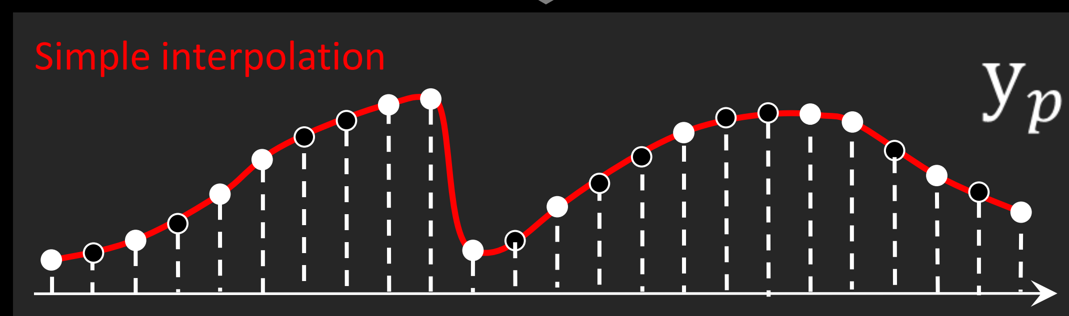
0.8 of the pixels are missing





Ordering

Reconstruction: 27.15dB



Inpainting Results – Examples

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

Inpainting Results

Reconstruction results from 80% missing pixels using various methods:

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

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.

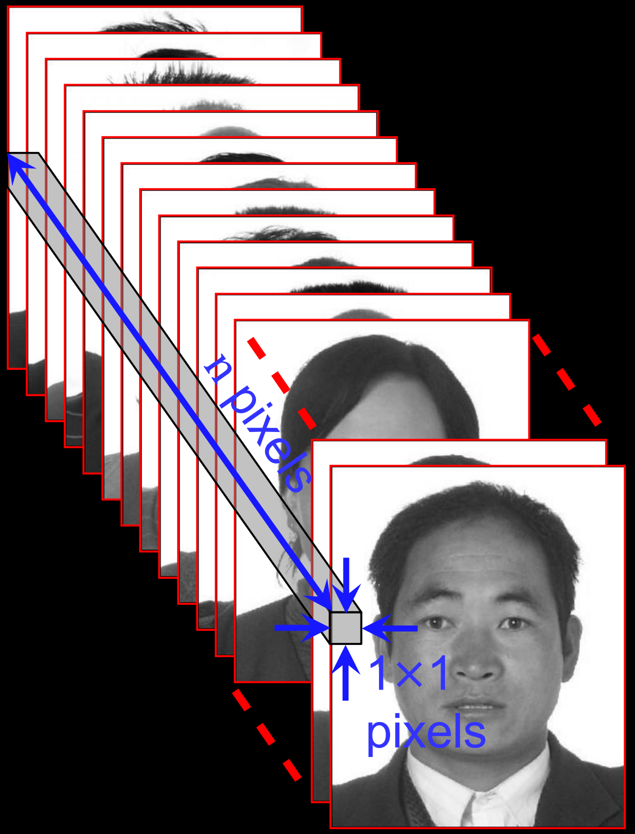
SKIP?

Part IV

Image Compression

Facial Image Compression

- The problem: Compressing photo-ID images.
- General purpose methods (JPEG, JPEG2000) do not take into account the specific family.
- By adapting to the image-content (e.g. 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

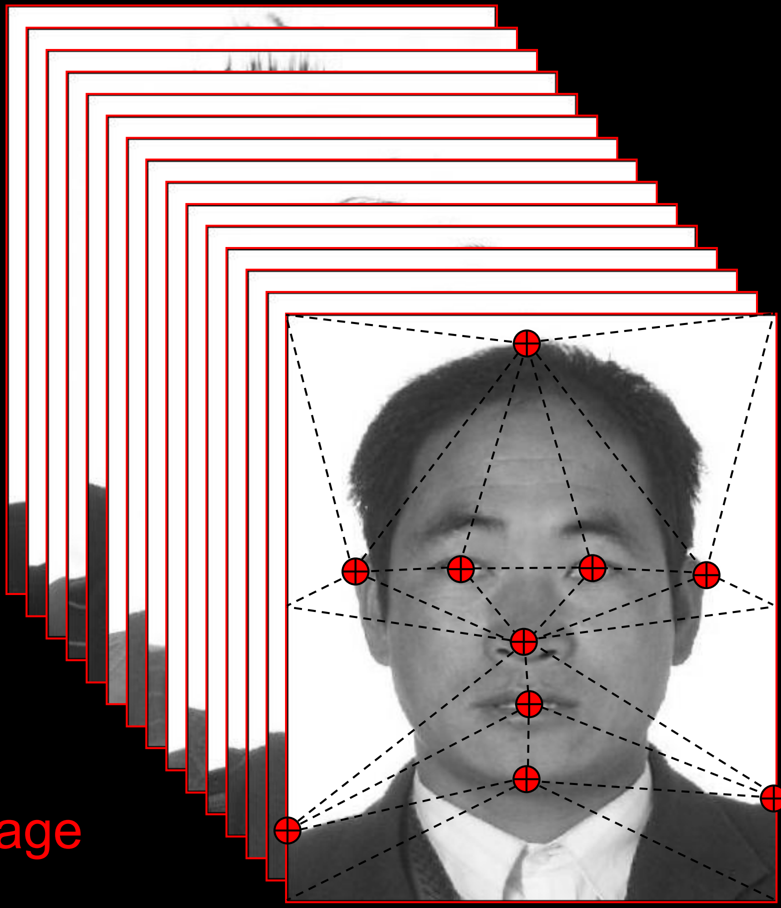
↓

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)

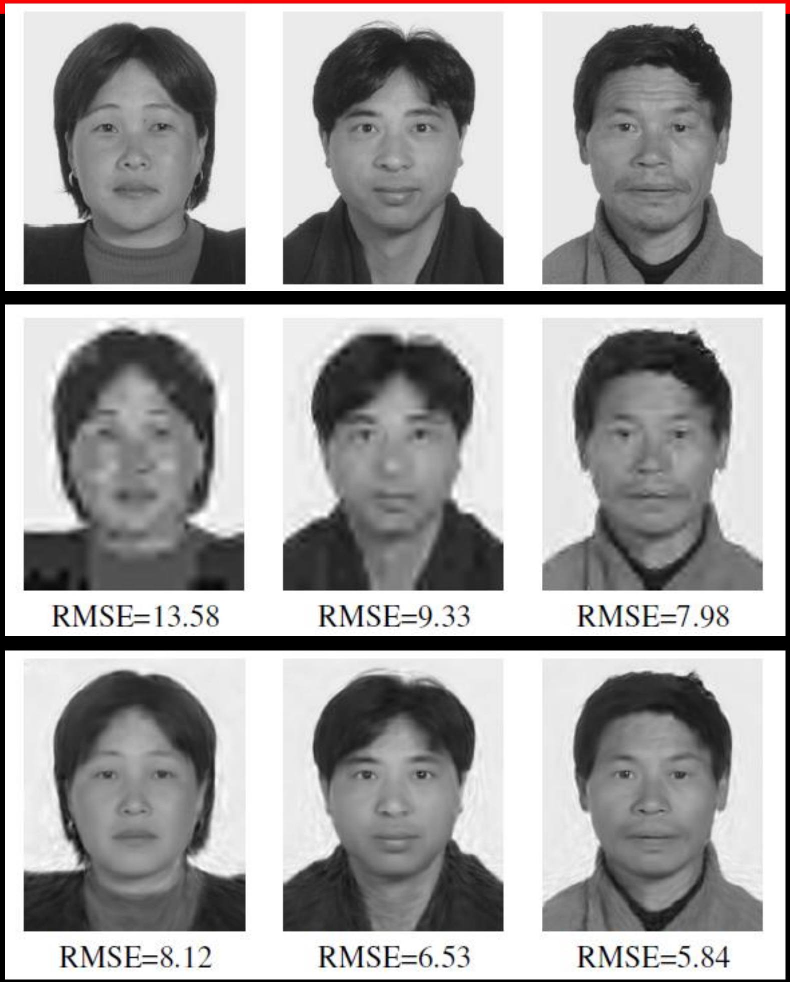


Results

The original images

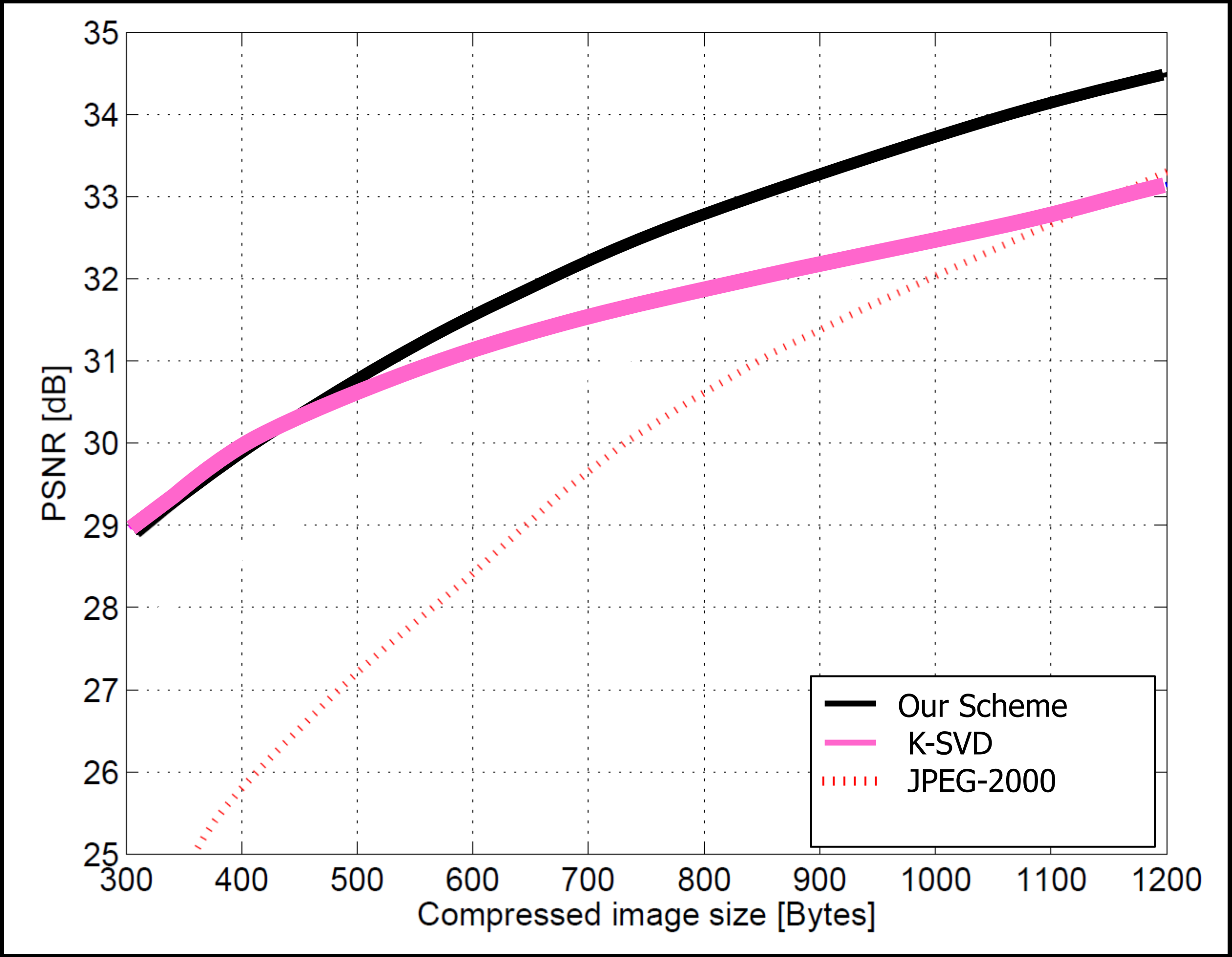
JPEG2000

Our scheme



400 bytes 600 bytes 800 bytes

Rate-Distortion Curves



Part V

Time to Finish

Conclusions

2D to 1D conversion is not necessarily a bad idea, and especially so if done in an image adaptive way

We propose such a 1D ordering based on approximating the shortest path in the patch domain

We demonstrate the effectiveness of this approach to image denoising, inpainting and compression

What next? Many things …

- ❑ Use this paradigm for general inverse problems
- ❑ Why just permutation and not other orderings
- ❑ Merge with statistical modeling of images
- ❑ Improve the TSP approximation
- ❑ …

Thank you to

Chen Sagiv and Jacob Cohen

for a very interesting event and

for inviting me

Questions ?

Image Denoising – Improvements

Cycle-spinning: Apply the above scheme several times, with a different random ordering, and then average the results.

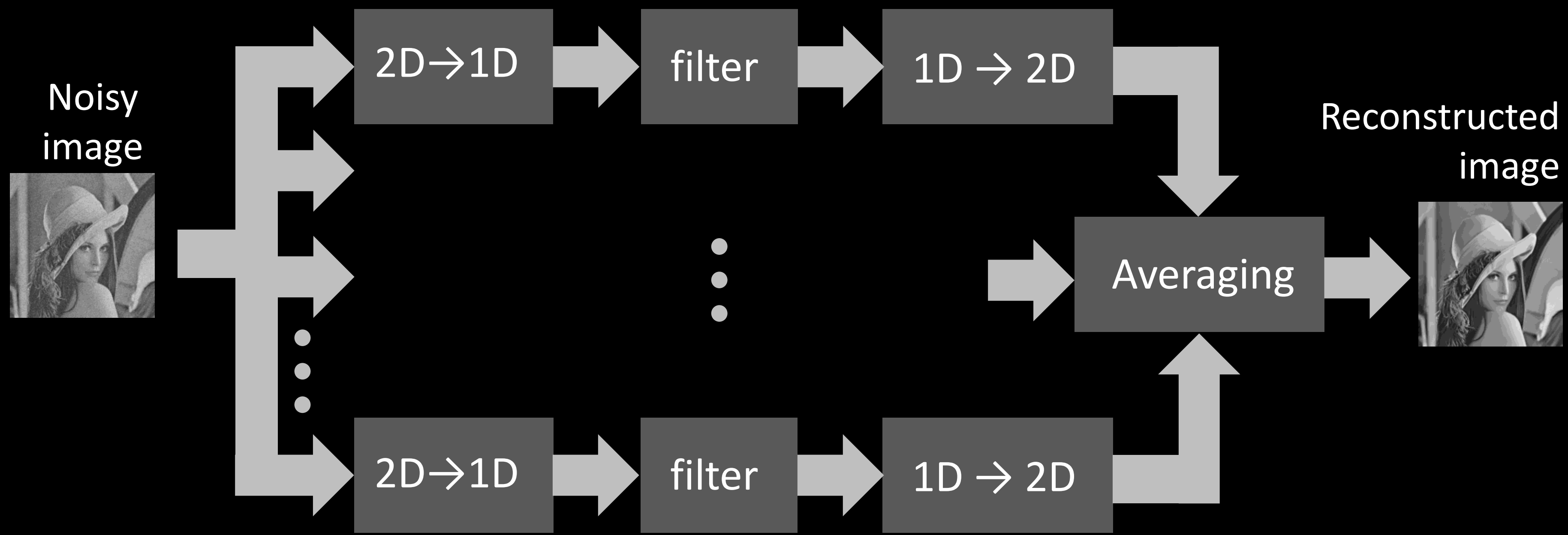


Image Denoising – Improvements

Sub-image averaging: A by-product of our patch-ordering approach is the fact that the whole patches are ordered. Why should we then adopt the found ordering only w.r.t. the middle row (corresponding to the center pixel)?

Instead, we can filter each of the (reordered) rows, and then average back these to a single image.

Note that these rows correspond to various fixed shifts of the image, so they have to be positioned correctly before averaging.

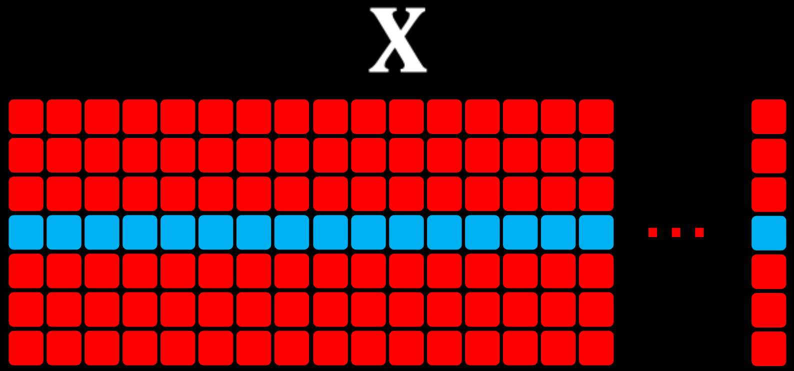


Image Denoising – Improvements

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.