Image Restoration

Giacomo Boracchi

CVPR USI, May 22 2020

giacomo.boracchi@polimi.it

https://boracchi.faculty.polimi.it/

Inverse Problems

Direct Problem

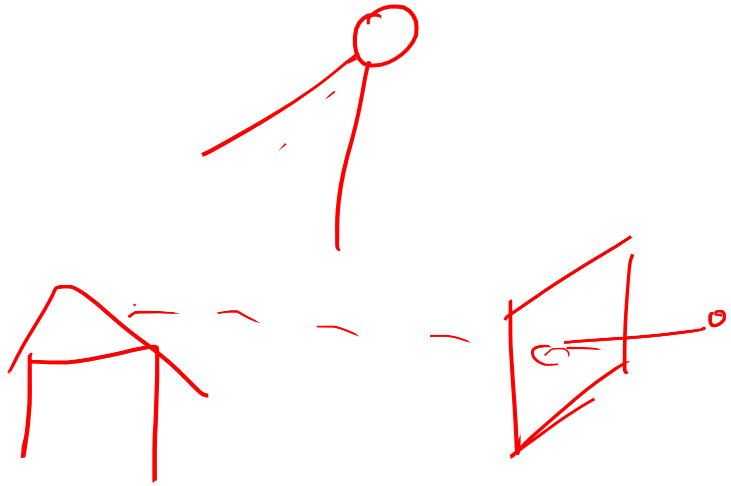
The Inverse Problem

Direct and Inverse problems

Often, direct problems can be addressed through simulation with respect to some know physical (natural) law.

An example related to CV:

- Image rendering



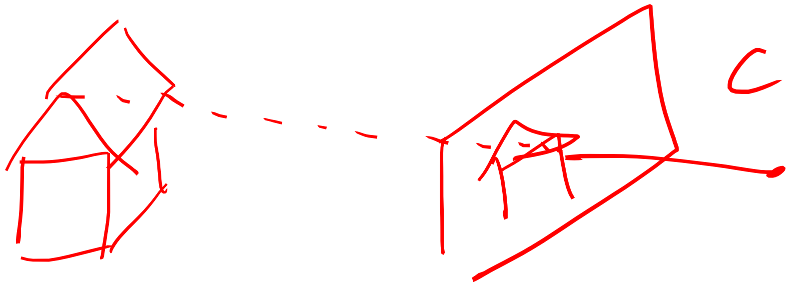
Direct and Inverse problems

Inverse problems goes in the opposite direction and are much more difficult to solve.

They are often ill posed as they admit infinite many solutions

An example:

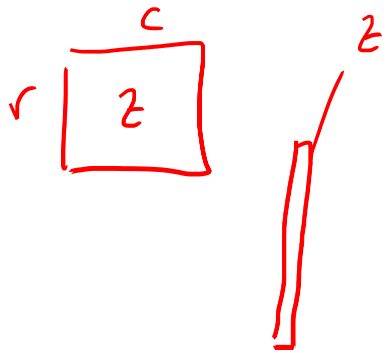
- 3D reconstruction from a single image



Inverse problems in Imaging

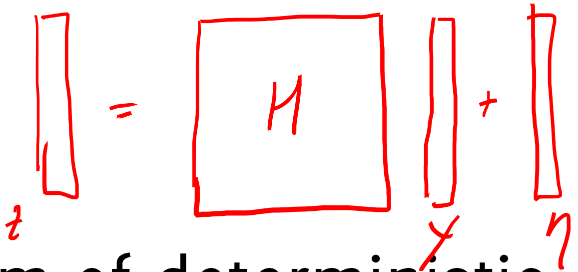
One typically observe

$$z = Hy + \eta$$



Where $y, z, \eta \in \mathbb{R}^d$ are images arranged as vectors that corresponds to

- y : the unknown input image, which is to be recovered
- z : the observation
- η : the noise corrupting image acquisition



And $H \in \mathbb{R}^{d \times d}$ is a linear operator describing some form of deterministic distortion during the image acquisition process

Blur

Blurry images can be modeled

$$z = Hy + \eta$$

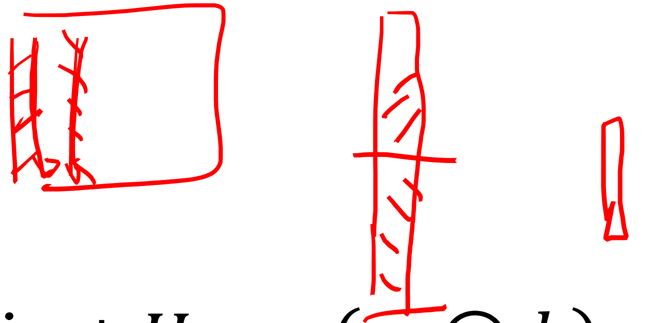
Where Hy in its simplest form is spatially invariant $Hy = (y \circledast h)$

Forward problem

Inverse problem

Blur

$z = Hy \qquad y = H^{-1}z$

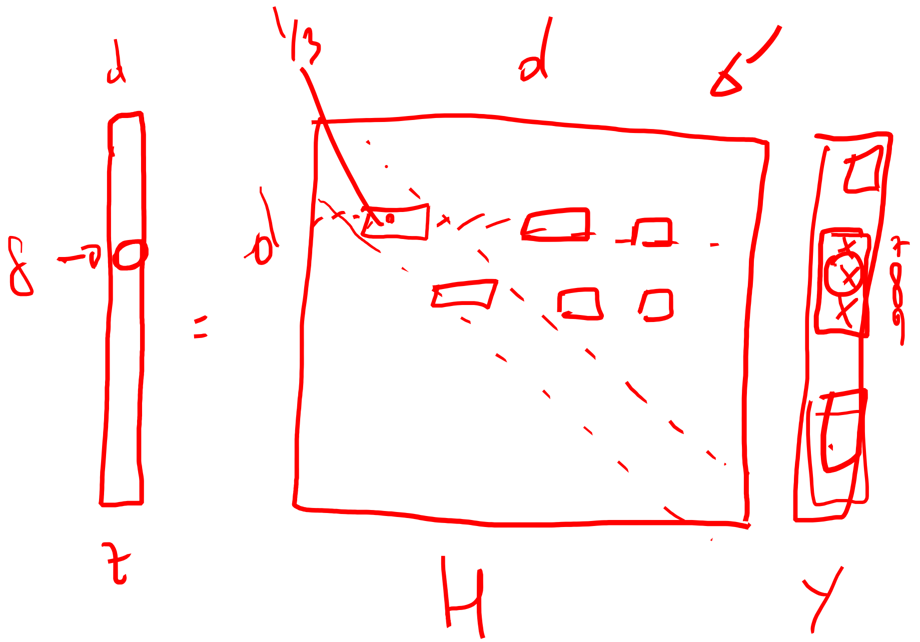


Blurry images can be modeled

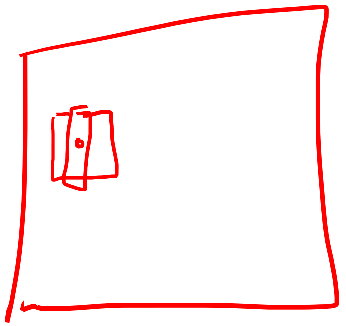
$$z = Hy + \eta$$

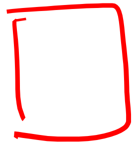
Where Hy in its simplest form is spatially invariant $Hy = (y \circledast h)$

What H corresponds to a spatially invariant blur operator?



$\left[\frac{1}{3}; \frac{1}{3}; \frac{1}{3}\right]$





$z = y * h$

$F(z) = F(y) .* F(h)$

$F(y) = \frac{F(z) + \eta}{F(h)}$

The Deblurring Problem

The convolution theorem states that

$$Fourier((f \circledast g)) = FG$$

- Convolution can be computed by performing
  - Fourier transofrm,
  - Element-wise product
  - Inverse Fourier transform
- (and if FFT is possible, this is less expensive than computing convolutions in space domain)

- Blur can be easily inverted in Fourier domain

Note that the convolution theorem holds when the signal is periodic, and thus the circular convolution has to be computed

The Deblurring Problem

orig image — blurring kernel

Given the image y and the kernel h a blurred observation is given by

$$z = (y \circledast h)$$

then, when the filter h is exactly known, the convolutional blur can be inverted in Fourier Domain

$$Z = YH$$

Y = FFT2(y)

z = IFFT2(Z)

thus,

$$\widehat{y} = Fourier^{-1}\left(\frac{Z}{H}\right)$$

in order to define this ratio in frequencies where $H = 0$

$$\widehat{y} = Fourier^{-1}\left(\frac{Z\bar{H}}{\|H\|^2 + \epsilon}\right)$$

being $\epsilon > 0$ a regularization parameter (difficult to choose in practice).

The Deblurring Problem

but what's up when noise appears?

$$z = (y \circledast h) + \eta$$

Then, when computing the Fourier transform of the observation we have

$$Z = YH + N$$

Thus

$$\widehat{y} = Fourier^{-1}\left(\frac{Y\bar{H}}{\|H\|^2 + \epsilon} + \frac{N\bar{H}}{\|H\|^2 + \epsilon}\right)$$

.. thus even unperceptible amount of noise, may become problematic in the second term of the sum.

Not all blurs are spatially invariant

Inpainting

Blurry images can be modeled

$$z = Hy + \eta$$

H

Forward problem

Inverse problem

Inpainting

Blurry images can be modeled

$$z = Hy + \eta$$

What H corresponds to an inpainting operator?

Denoising

The simplest of Inverse Problems

Denoising: The Issue

A Detail in Camera Raw Image z

Denoising: The Issue

Denoised $\hat{y}$

Denoising: The Issue

A Detail in Camera Raw Image z

Denoising: The Issue

Denoised $\hat{y}$

Image Formation Model

Observation model is

$$z(x) = y(x) + \eta(x), \qquad x \in \mathcal{X}$$

z $\qquad$ y $\qquad$ η

Image Formation Model

Observation model is

$$z(x) = y(x) + \eta(x), \qquad x \in \mathcal{X}$$

Where

- x denotes the pixel coordinates in the domain $\mathcal{X} \subset \mathbb{Z}^2$
- y is the original (noise-free and unknown) image
- z is the noisy observation
- η is the noise realization

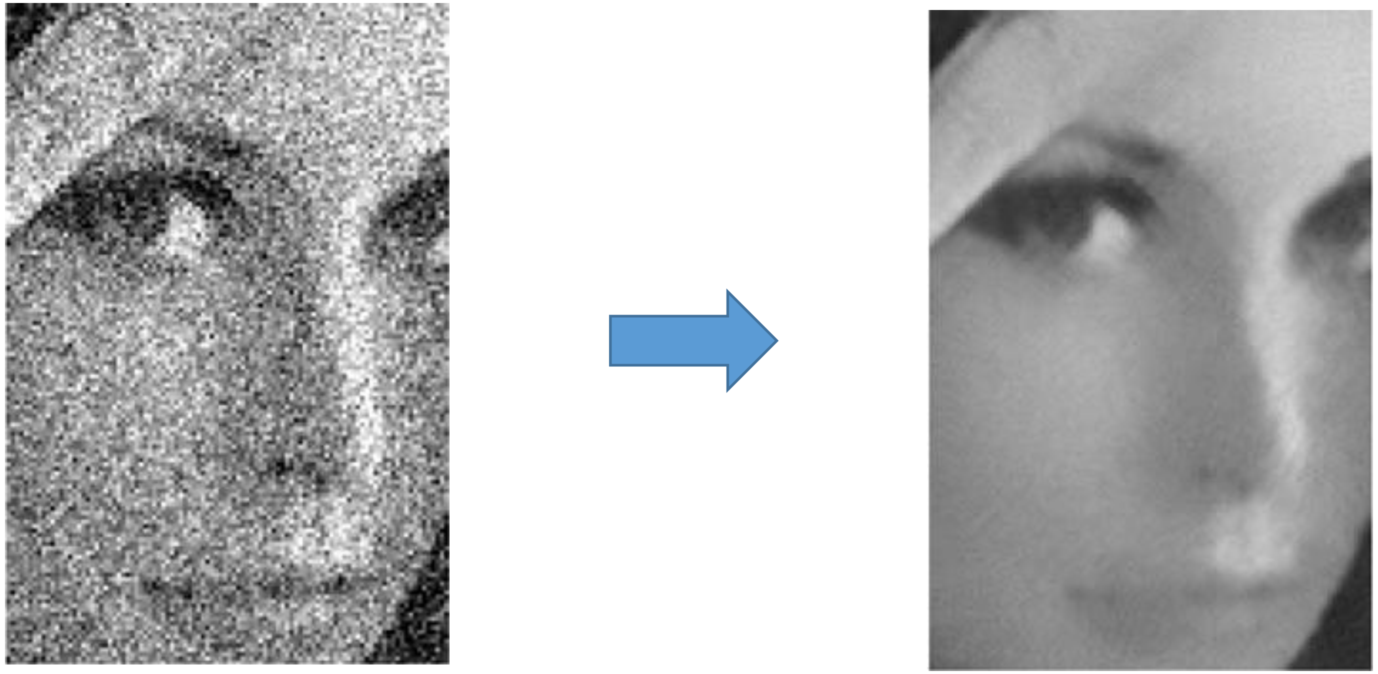
For the sake of simplicity we assume AWG: $\eta \sim N(0, \sigma^2)$ and $\eta(x)$ independent realizations.

The noise standard deviation σ is also assumed as known.

Goal of Image Denoising

The goal of **image denoising** is to compute $\hat{y}$ *realistic* estimate of the original image y, given the noisy observation z

Denoising is an **ill posed problem** and requires some form of **regularization** to promote outputs that are close to natural images



z $\hat{y}$ $\approx y$

$z = y + \eta$

Image Denoising

Deniosing is a fundamental step in image processing pipelines

- Improves the quality of digital images to the standard we are used to
- Eases the following algorithms in imaging pipelines from those solving low-level (e.g., edge detection), till high-level (recognition) problems
- It is also a tool to quantitatively assess the performance of a descriptive model for images.

$$z = y + \eta$$

Denoising as a Regression Problem

Image Denoising

Estimating $y(x)$ from $z(x)$ can be statistically treated as regression of z given x

$$\hat{y}(x) = \mathrm{E}[z \mid x]$$

Denoising and Regression

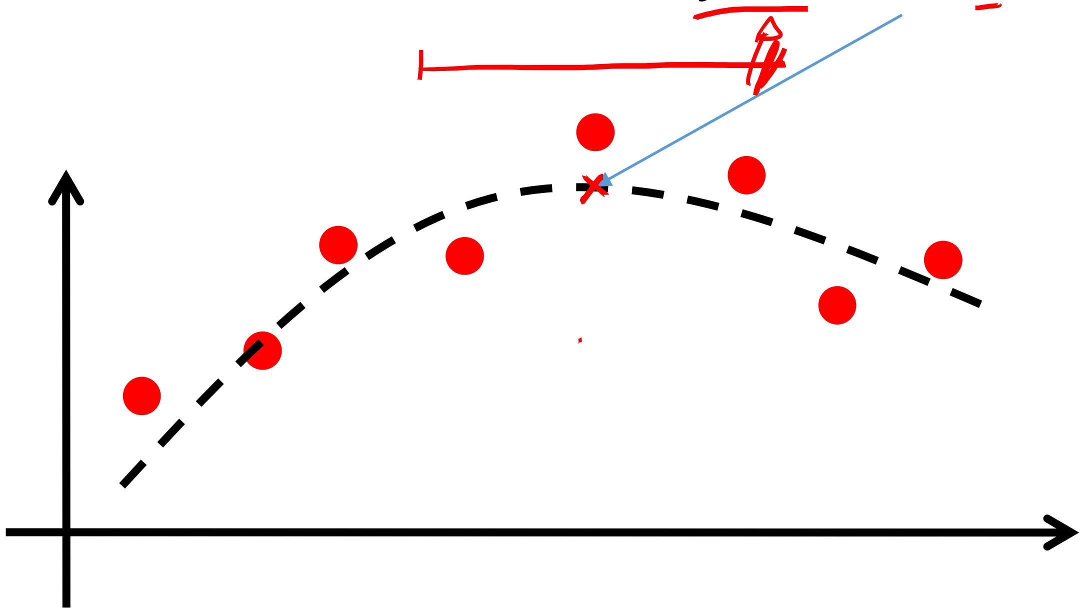
Observation model is

image

$$z(x) = y(x) + \eta(x), \qquad x \in \mathcal{X}$$

Denoising can be formulated as a regression problem

$$\hat{y}(x) = \mathrm{E}[z \mid x]$$



Denoising by Fitting a Constant Value

The problem: Estimate the **constant C that minimizes the distance** w.r.t. noisy observations in U_0, a neighborhood of x_0

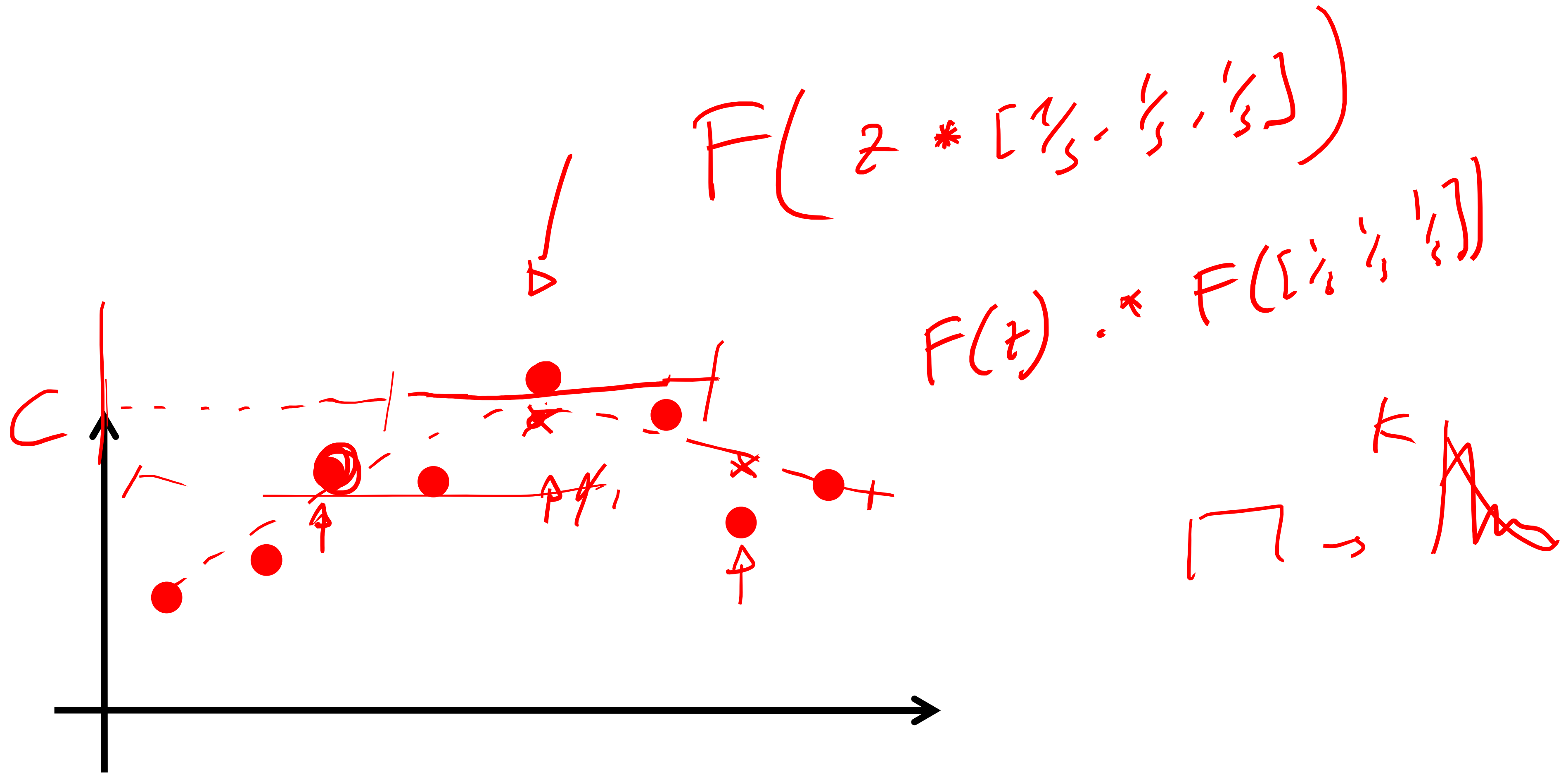
$$\hat{y}(x_0) = \underset{C}{\operatorname{argmin}} \sum_{x \in U_0} (z(x) - C)^2$$

There are many unbiased estimators (namely such that $E_\eta[\hat{y}] = y$), the BLUE (Best Linear Unbiased Estimator) is the averaging noisy samples U_0, namely:

$$\hat{y}(x_0) = \frac{1}{\#U_0} \sum_{x \in U_0} z(x)$$

Denoising by Fitting a Constant Value

The above method replaces each noisy input $z(x)$ with the average over a fixed neighborhood U_x centered in x.



How would you implement this?

Denoising by Weighted Fit of a Constant

The problem: Estimate the **constant C that minimizes a weighted loss** over noisy observations

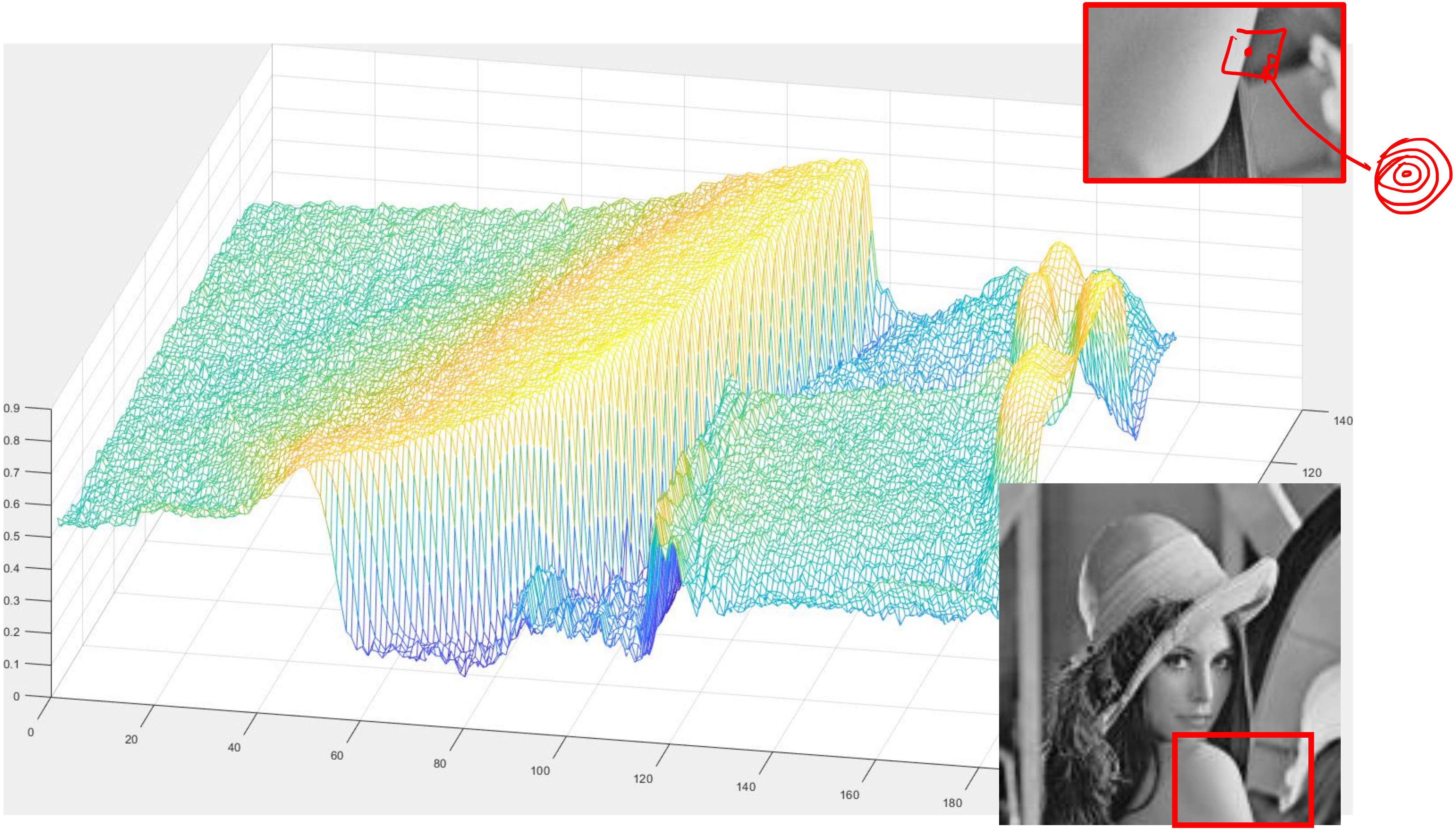
$$\hat{y}_h(x_0) = \underset{C}{\operatorname{argmin}} \sum_{x_s \in \mathcal{X}} w(x_0 - x_s)(z(x_s) - C)^2$$

$$w = \{w(x)\} \text{ s.t. } \sum_{x_s \in \mathcal{X}} w(x_s) = 1$$

This problem can e solved by **computing the convolution** of the image z against a **filter whose coefficients are the error weights**

$$\hat{y}(x_0) = (z \circledast w)(x_0)$$

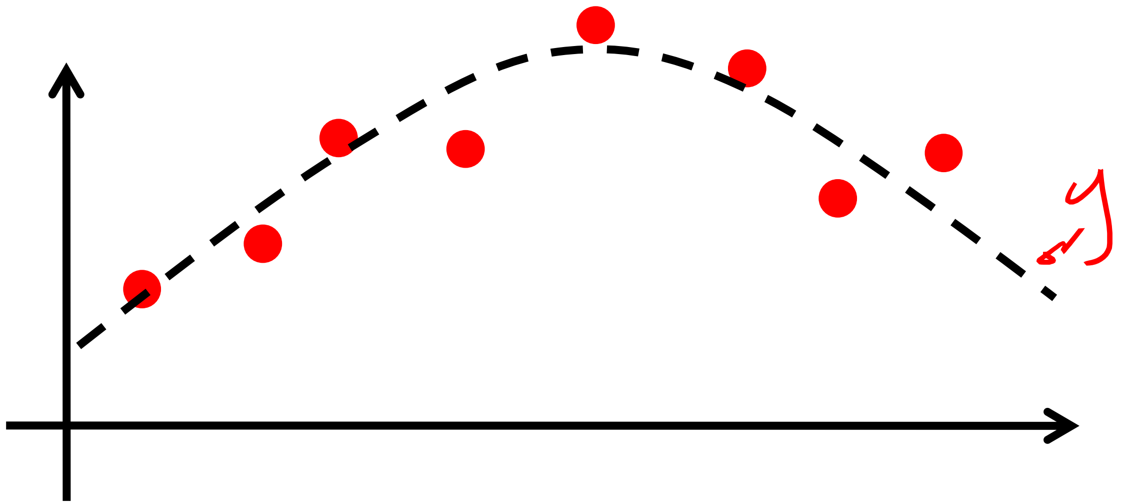
Smoothing is Agnostic to Image Content



Giacomo Boracchi

… of course

To perform regression more effectively, we can leverage a model describing the true signal y



In this illlustration, assuming the noise-free signal follows a polynomial trend might help!

We can pursue a "regression-approach", but on images it is difficult to define a parametric expression for y on X

Difficult to Find a Good Model for Lena..

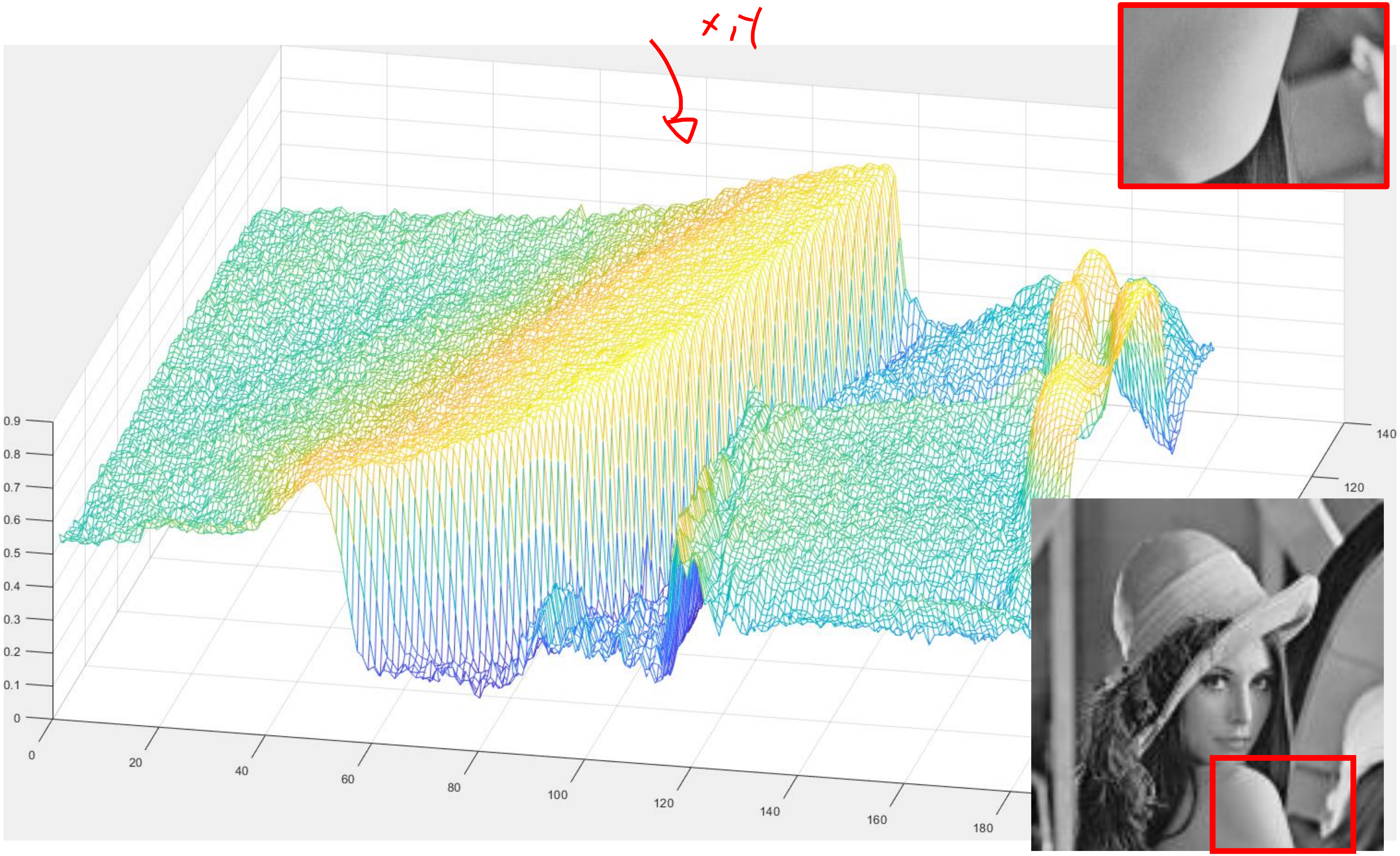


Image Restoration

Giacomo Boracchi

CVPR USI, May 26 2020

giacomo.boracchi@polimi.it

https://boracchi.faculty.polimi.it/

An Overview on Denoising Approaches

Pointwise vs Multipoint Methods

Another view on Denoising Approaches

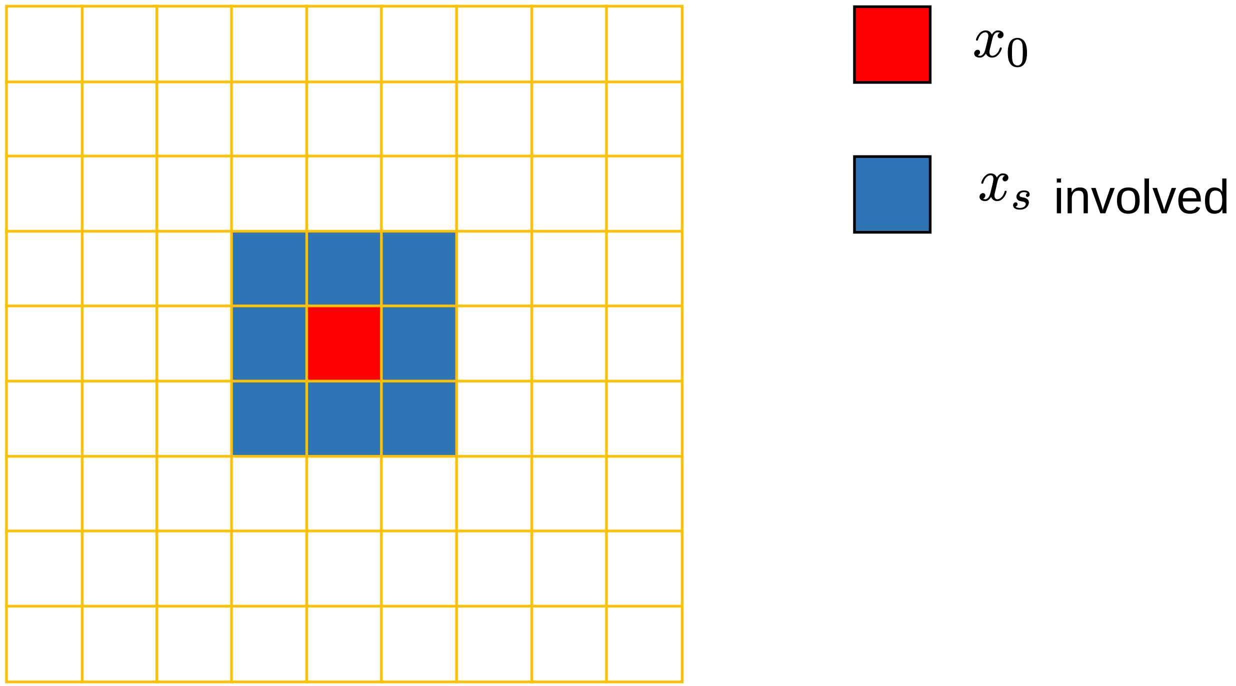
Pointwise / Multipoint

- **Pointwise:** the estimation of noise-free signal is computed for the central point only y_0, and not for all the other points considered
- **Multipoint:** the estimation of the noise-free signal is computed for all the points y_s used by the estimator to estimate y_0.

Katkovnik, V., A. Foi, K. Egiazarian, and J. Astola, "From local kernel to nonlocal multiple-model image denoising",Int. J. Computer Vision 2009

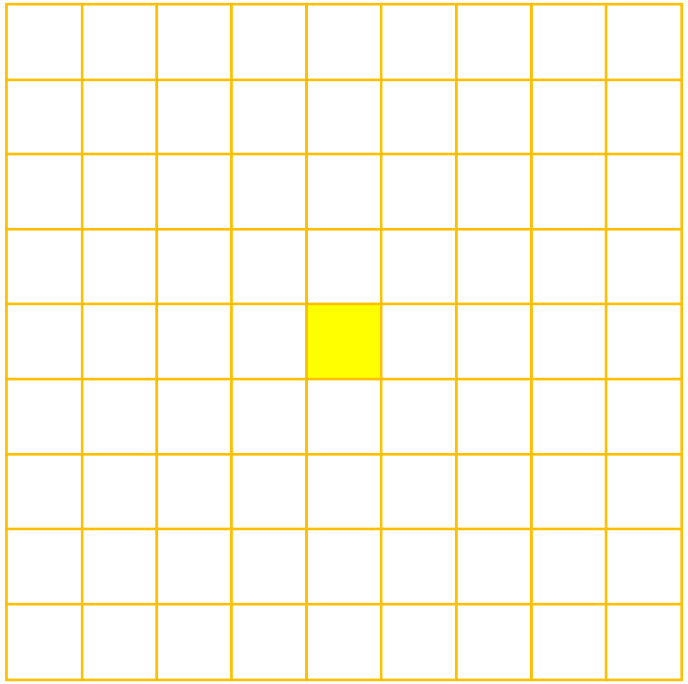
Pointwise vs Multipoint

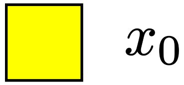
Pointwise: the estimate is given for the central point only



Pointwise vs Multipoint

Pointwise, the estimate is given for the central point only

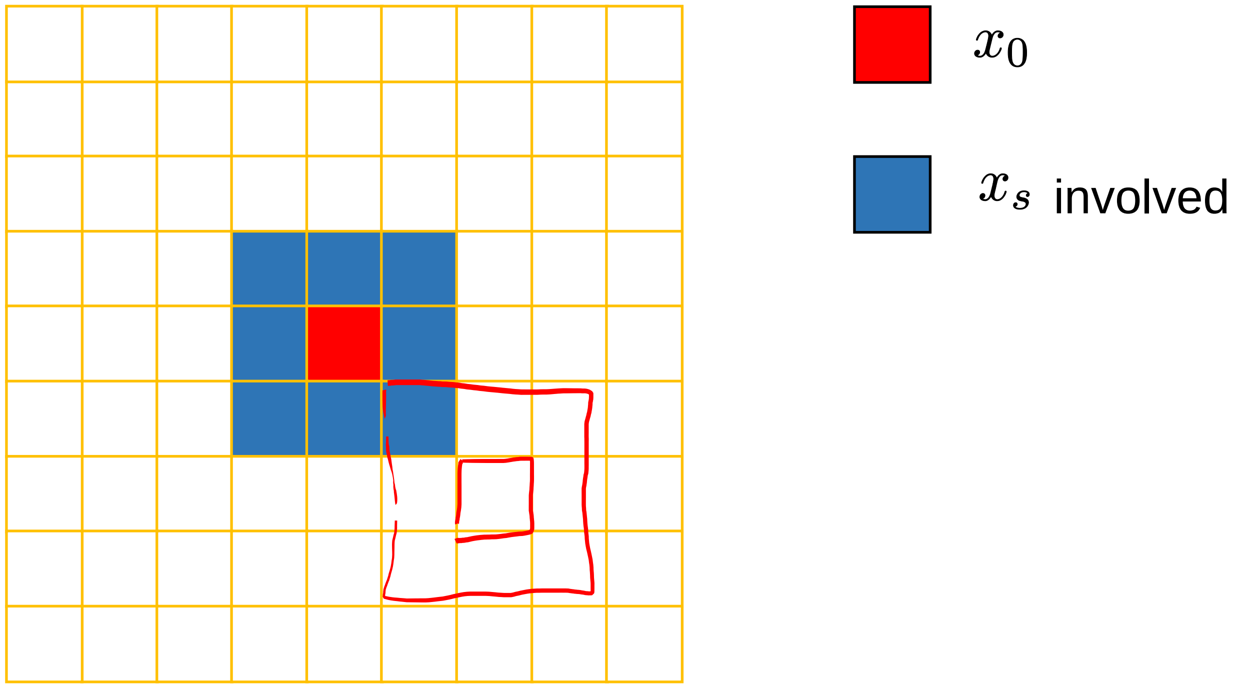




Pixels where
the true signal
is estimated

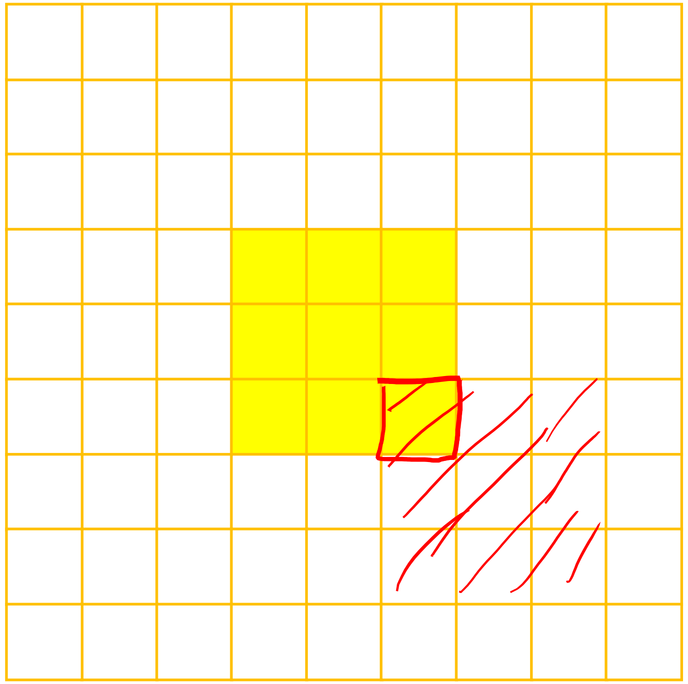
Pointwise vs Multipoint

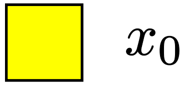
Multipoint, the original image is estimated in all the pixels considered in the filtering



Pointwise vs Multipoint

Multipoint, the original image is estimated in all the pixels considered in the filtering





Pixels where the true signal is estimated

Parametric vs Non-Parametric

Denoising Approaches

Parametric Approaches

- Assume the noisy-free signal y features some **sparsity property** in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition. This implies that your image admits a **global parametric representation**

Non Parametric Approaches

- Local Smoothing / Local Approximation   Bilateral filter / LPA-ICI
- Non Local Methods   NLMs

Deep Learning

- Entirely data driven, these methods do not rely on an explicit prior for handling images

Denoising Approaches

Non Parametric Approaches

- Local Smoothing / Local Approximation
- Non Local Methods

Estimating $y(x)$ from $z(x)$ is treated as regression of z given x

$$\hat{y}(x) = \mathrm{E}[z \mid x]$$

Bilateral Filter

Bilateral Filter

weights depend on x_1 and x_2

The denoised image $\hat{y}$ is a weighted average of (in principle) all the pixels

$$\hat{y}(x_1) = \frac{1}{W_1} \sum_{x_2 \in \mathcal{X}} w\,(x_1, x_2) z(x_2), \qquad \forall x_1 \in \mathcal{X}$$

$x_2 \in U_{x_1}$

where weights $\{w(x_1, x_2)\}$ are adaptively defined depending on the image content and distance from pixel position

And W_1 is a normalization factor for $\hat{y}(x_1)$

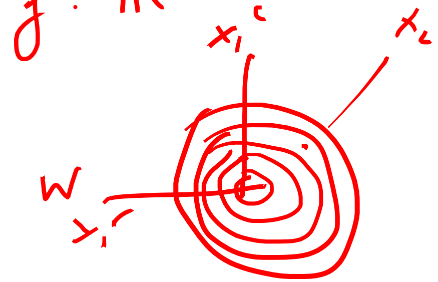
$$W_1 = \sum_{x_2 \in \mathcal{X}} w\,(x_1, x_2)$$

Tomasi, C., & Manduchi, R. *Bilateral filtering for gray and color images.* In ICCV 1998

Weights in Bilateral Filters

In particular, weights are function of photometric and spatial distance between x_1 and x_2

$$w(x_1, x_2) = f_r\big(z(x_1) - z(x_2)\big)\, g_s(\|x_1 - x_2\|)$$



Often functions f_r and g_s are Gaussians

$$f_r(x_1, x_2) = e^{-\frac{||z(x_1) - z(x_2)||^2}{2\sigma_r^2}}, \qquad g_s(x_1, x_2) = e^{-\frac{||x_1 - x_2||^2}{2\sigma_s^2}}$$

Where the parameters σ_r and σ_s regulate the weight decay factors and compensate for different ranges (image intensity / space)

Tomasi, C., & Manduchi, R. *Bilateral filtering for gray and color images.* In ICCV 1998

Weights in Bilateral Filters

Weights take into account two different priors:

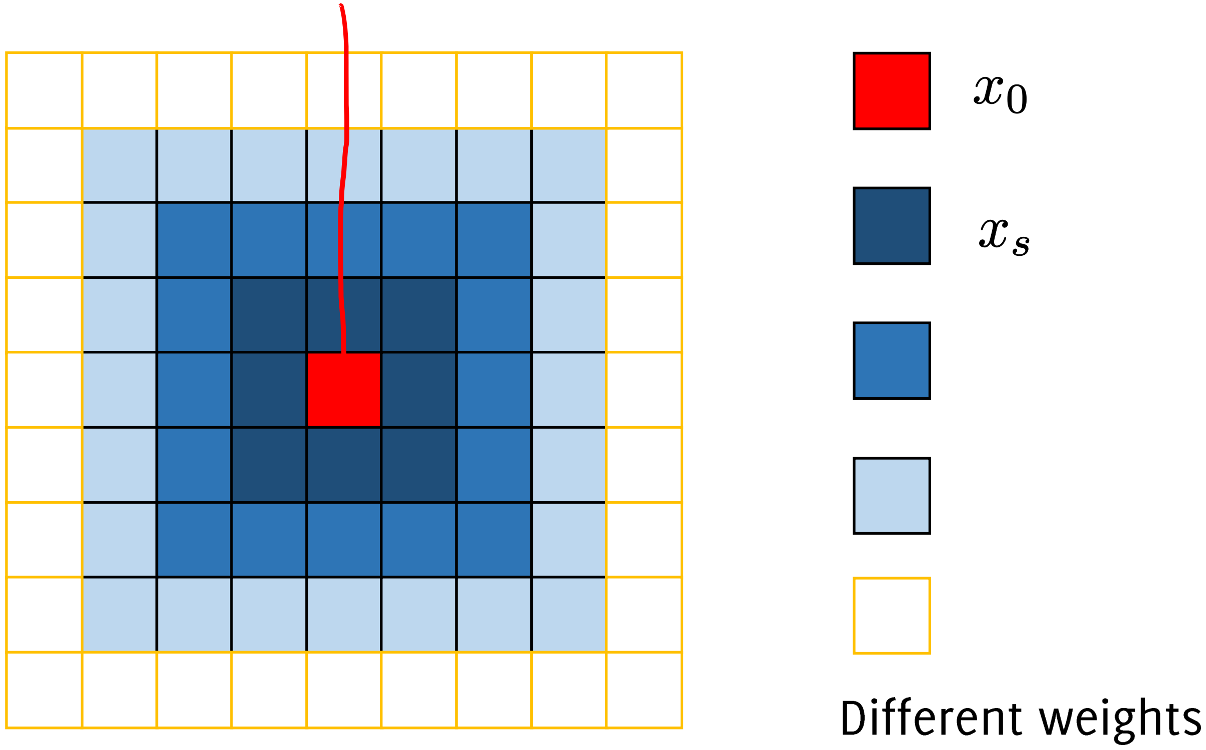
- Local similarity
- Non-local similarity

What do we obtain if we ignore the photometric contribution in the weight definition, namely we set $f_r = 1$?

Tomasi, C., & Manduchi, R. *Bilateral filtering for gray and color images.* In ICCV 1998

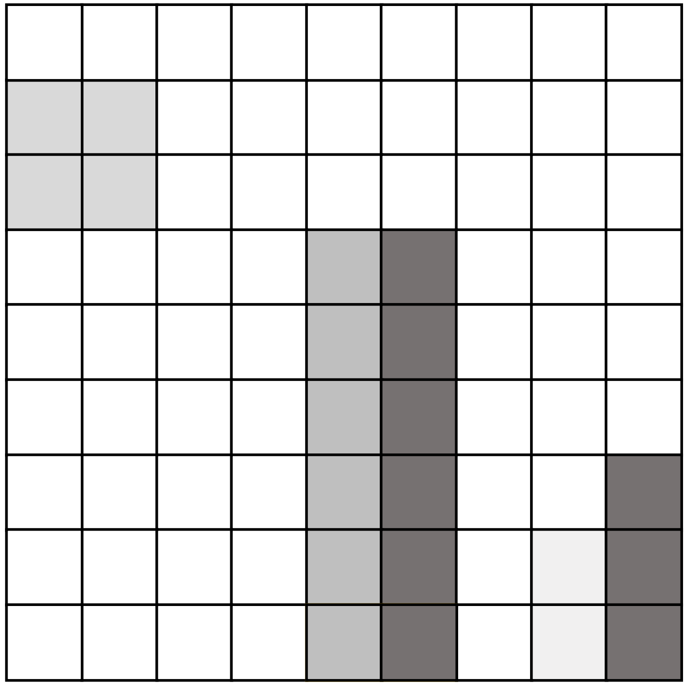
Local vs Non-Local

Local: weights are determined by the pixel distance (regardless of the image content)



Local vs Non-Local

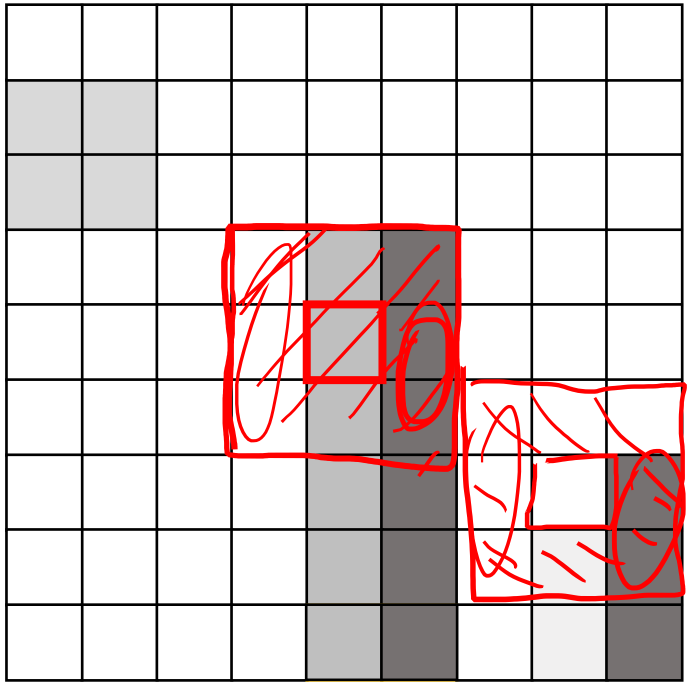
Non Local: weights are determined by the image similarity



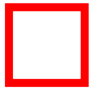
Example of observation

Local vs Non-Local

Non Local: weights are determined by the image similarity

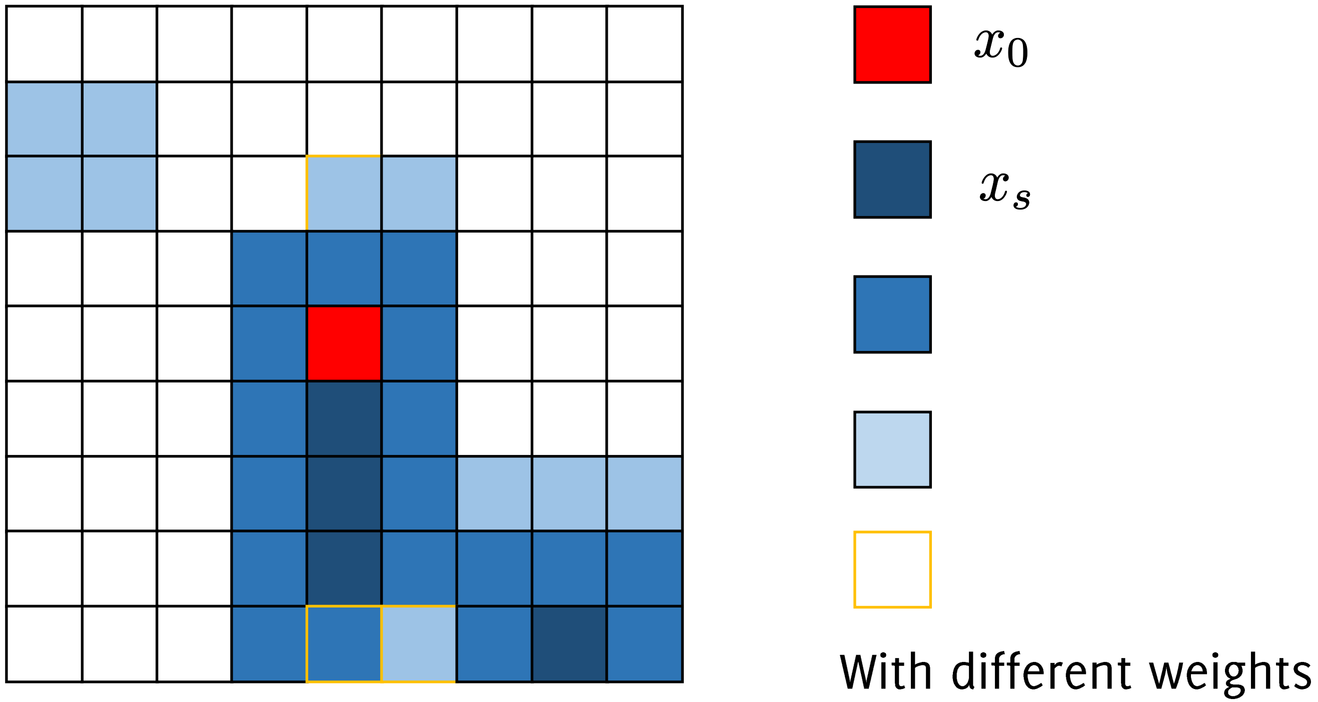


Example of observation

 x_0

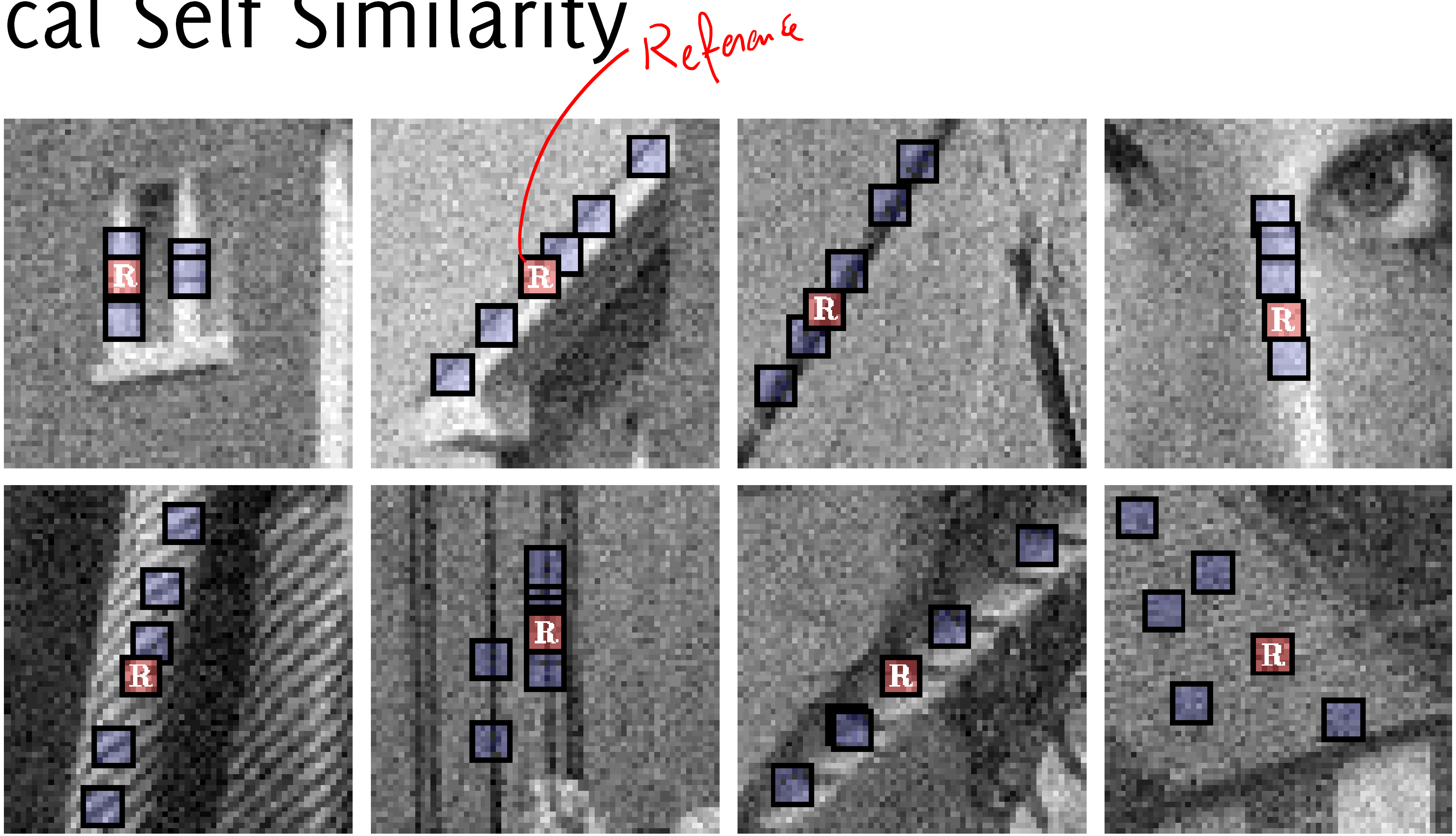
Local vs Non-Local

Non Local, weights are determined by the image similarity



Nonlocal Self Similarity denoising

NonLocal Self Similarity



In a natural image, for any given patch there exist ***many*** *other* ***similar*** *looking* ***patches*** *at* ***different spatial locations.***

Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. *Image denoising by sparse 3-D transform-domain collaborative filtering. IEEE TIP 2007, 16*(8)

Nonlocal Self-similarity

NonLocal Self Similarity in Image Processing

Traced back to **fractal models** of natural images (Barnsley, 1993) and fractal block coding (Jacquin, 1992)
.. *self-transformability on a blockwise basis…*

Texture synthesis and **completion** (Efros and Leung, 1999; Wei and Levoy, 2000).

Predicting the **central pixel** of a patch by exploiting the *long-range correlation* of natural images (Zhang and Wang, 2002)

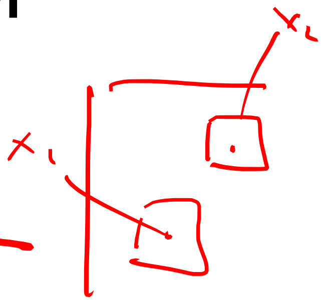
Nonlocal self-similarity as **an effective regularity assumption** at the heart of many successful image **denoising algorithms** (NL-means, BM3D, etc.).

Nonlocal self-similarity was successfully used for **several image/video processing tasks**.

Denoising methods based on self-similarity

These methods leverage **self similarity** of image **patches as a form of regularization** for distinguishing natural images from noise

Self-similarity is assessed at patch-level and in a non-local manner.



Similar patches have to be correctly identified on the basis of a suitable patch **distance measure.**

$\|z_{x_1} - z_{x_2}\|$

$y(x_1) - y(x_2)$

- **This is different from Bilateral Filter which considers only the photometric distance at pixel-wise level**

Such a distance implies the assumption of a **specific descriptive model for natural images** and their self-similarity.

The denoising effectiveness actually depends on the validity of such underlying model.

Image denoising (NL-Means)

Patches

$g_s(x_1 - x_2) = 1$

Let $U \subset \mathbb{Z}^2$ be a spatial neighborhood centered at the origin $(0,0) \in \mathbb{Z}^2$,

- we define a patch centered at a pixel $x \in X$ in the observation z

$$\boldsymbol{z}_{\boldsymbol{x}}(u) = z(x+u), \qquad u \in U$$

- a patch centered at a pixel $x \in X$ in the original image y

$$\boldsymbol{y}_{\boldsymbol{x}}(u) = y(x+u), \qquad u \in U$$

Giacomo Boracchi

Patch Distance

The idea:

The patch-wise distance correlates well with the distance of the two central pixels in the noisy-free patches

The idea is to assess the similarity between pixels $y(x_1)$ and $y(x_2)$ (not available), via the similarity of the corresponding noisy patches $\boldsymbol{z}_{x_1}$ and $\boldsymbol{z}_{x_2}$.

Thus, weights based on photometric differences should be computed from $||\boldsymbol{z}_{x_1} - \boldsymbol{z}_{x_2}||_2$ rather than $z(x_1) - z(x_2)$.

Windowed Patch Distance in NL-means

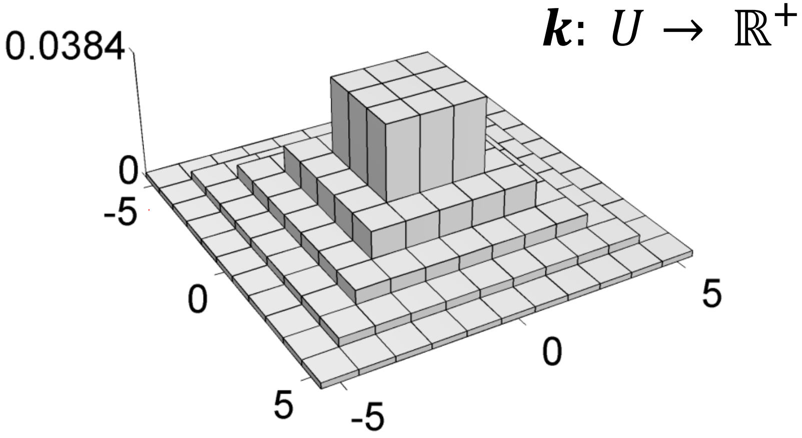
The distance operator is defined as a *windowed quadratic distance* between patches

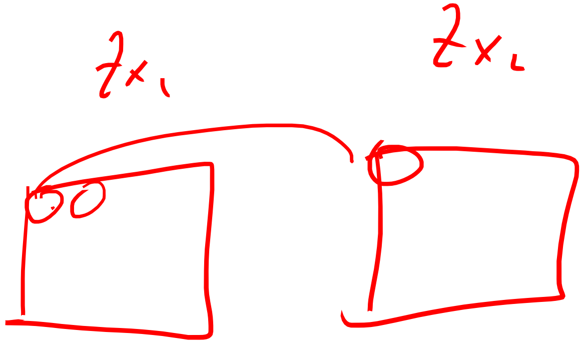
$$d(x_1, x_2) = \sum_{u \in U} \big(z(x_1 + u) - z(x_2 + u)\big)^2 \boldsymbol{k}(u)$$

Where $\boldsymbol{k}: U \to \mathbb{R}^+$ is a windowing kernel (we use $\boldsymbol{k}$ since w is being used for the aggregation weights)

Windowed patch distance in NL-means (cnt.)

The windowing kernel $\boldsymbol{k}\colon U \to \mathbb{R}^+$ adjusts the contribution of each difference term depending on the position of u with respect to the patch center.





$\| z_{x_1} - z_{x_2} \|_2^2$

$\| (z_{x_1} - z_{x_2}) \sqrt{k} \|_2^2$

d performs a **pixel-wise** comparison of the patches

the decay of $\boldsymbol{k}$ reflects how much similarity between $y(x_1)$ and $y(x_2)$ may be implied from the similarity between $y(x_1 + u)$ and $y(x_2 + u)$ when $u \neq 0$.

Non Local Means Filter (NL-means)

The denoised image $\hat{y}$ is a weighted average of all image pixels

$$\hat{y}(x_1) = \sum_{x_2 \in X} w\,(x_1, x_2) z(x_2), \qquad \forall x_1 \in X$$

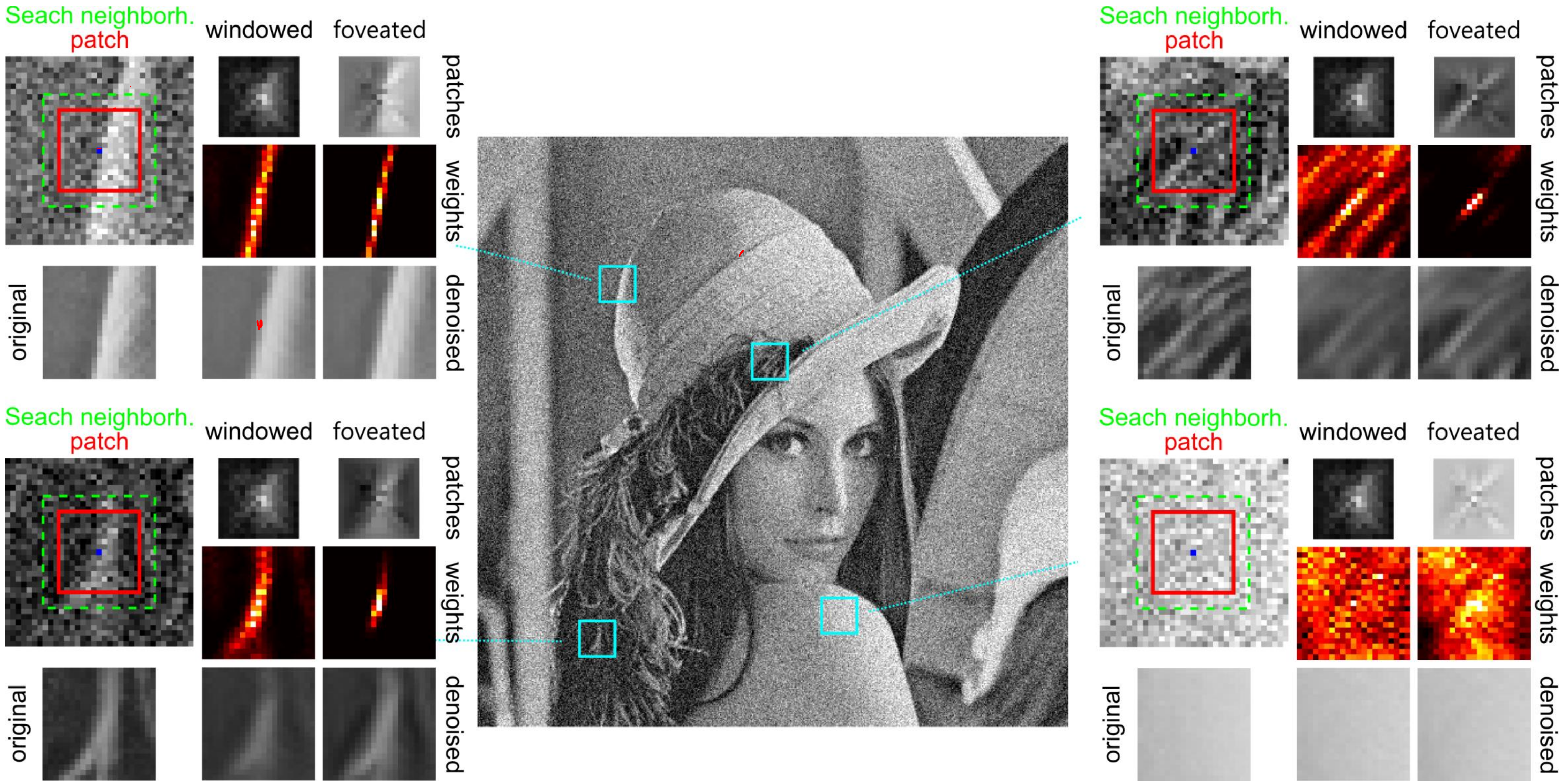
where weights $\{w(x_1, x_2)\}$ are adaptively defined depending on the similarity between two noisy patches $\boldsymbol{z}_{x_1}$ and $\boldsymbol{z}_{x_2}$

$$w(x_1, x_2) = \frac{e^{\left(-\frac{d(x_1,x_2)}{h^2}\right)}}{\sum_X e^{\left(-\frac{d(x_1,x_2)}{h^2}\right)}}$$

- $d(x_1, x_2)$: distance measure between patches in x_1 and x_2,
- $h > 0$ is a smoothing parameter ($h = \sigma$).
- $\boldsymbol{z}_{x_1}$ similar to $\boldsymbol{z}_{x_2} \Rightarrow d(x_1, x_2)$ is small $\Rightarrow$ $w(x_1, x_2)$ large
- NL-means operates pixel-wise

A. Buades, B. Coll, and J. M. Morel, A review of image denoising algorithms, with a new one, Multisc. Model. Simulat., vol. 4, no. 2, 2005

Weights from Patch-distances



Foi, A., & Boracchi, G. Foveated nonlocal self-similarity. *International Journal of Computer Vision*, 2016 *120*(1), 78-110.

Spatially Adaptive Methods

Denoising Approaches

Spatially adaptive methods, The basic principle:

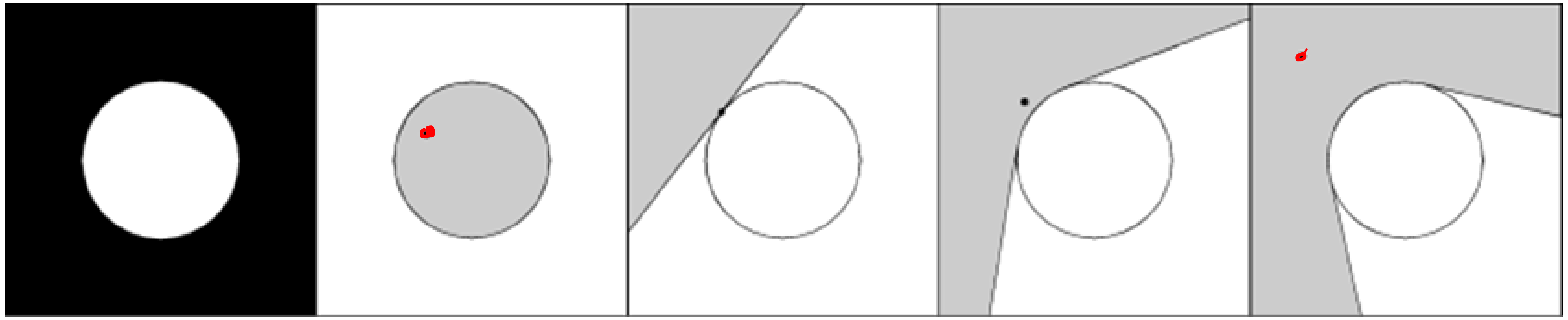
- there are no simple models able to describe the whole image y, thus perform the regression $\hat{y}(x) = \mathrm{E}[z \mid x]$
- Adopt a simple model in small image regions. For instance

$$\forall x \in X, \qquad \exists\, \widetilde{U}_x \ \text{ s.t. } \ y_{|\widetilde{U}_x} \text{ is a polynomial}$$

- Define, in each image pixel, the "**best neighborhood**" where a simple parametric model can be enforced to perform regression.
- For instance, assume that on a suitable pixel-dependent neighborhood, where the image can be described by a polynomial

Ideal neighborhood – an illustrative example

Ideal in the sense that it defines the support of a pointwise Least Square Estimator of the reference point.



Typically, even in simple images, every point has its own different ideal neighborhood.

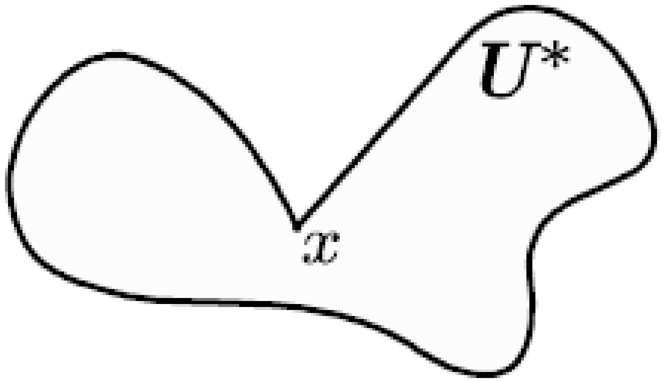
For practical reasons, the ideal neighborhood is assumed starshaped

Further details at LASIP c/o Tampere University of Technology http://www.cs.tut.fi/~lasip/

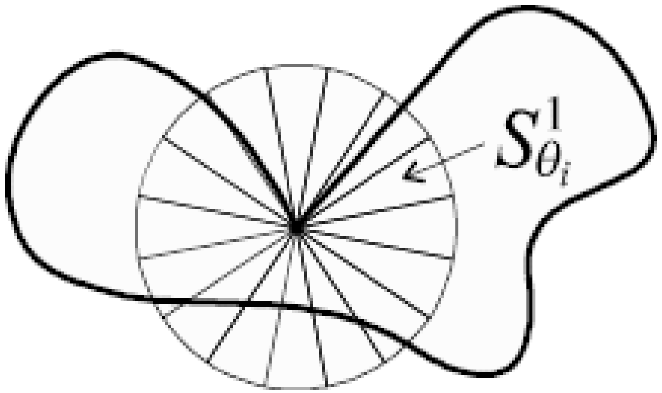
Neighborhood Discretization

A suitable discretization of this neighborhood is obtained by using a set of directional LPA kernels $\{g_{\theta,h}\}_{\theta,h}$, where

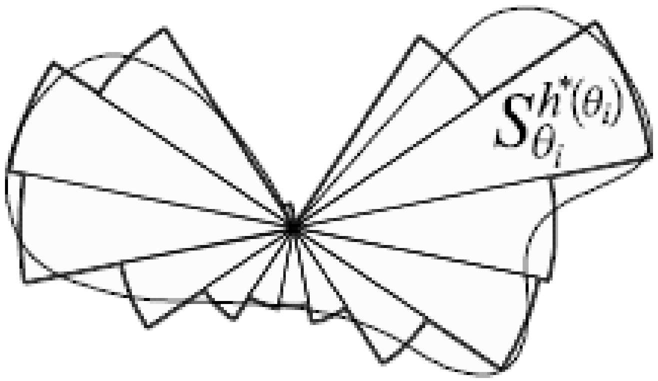
- θ determines the orientation of the kernel support,
- h controls the scale of kernel support



Ideal Neighborhood



Directional kernels

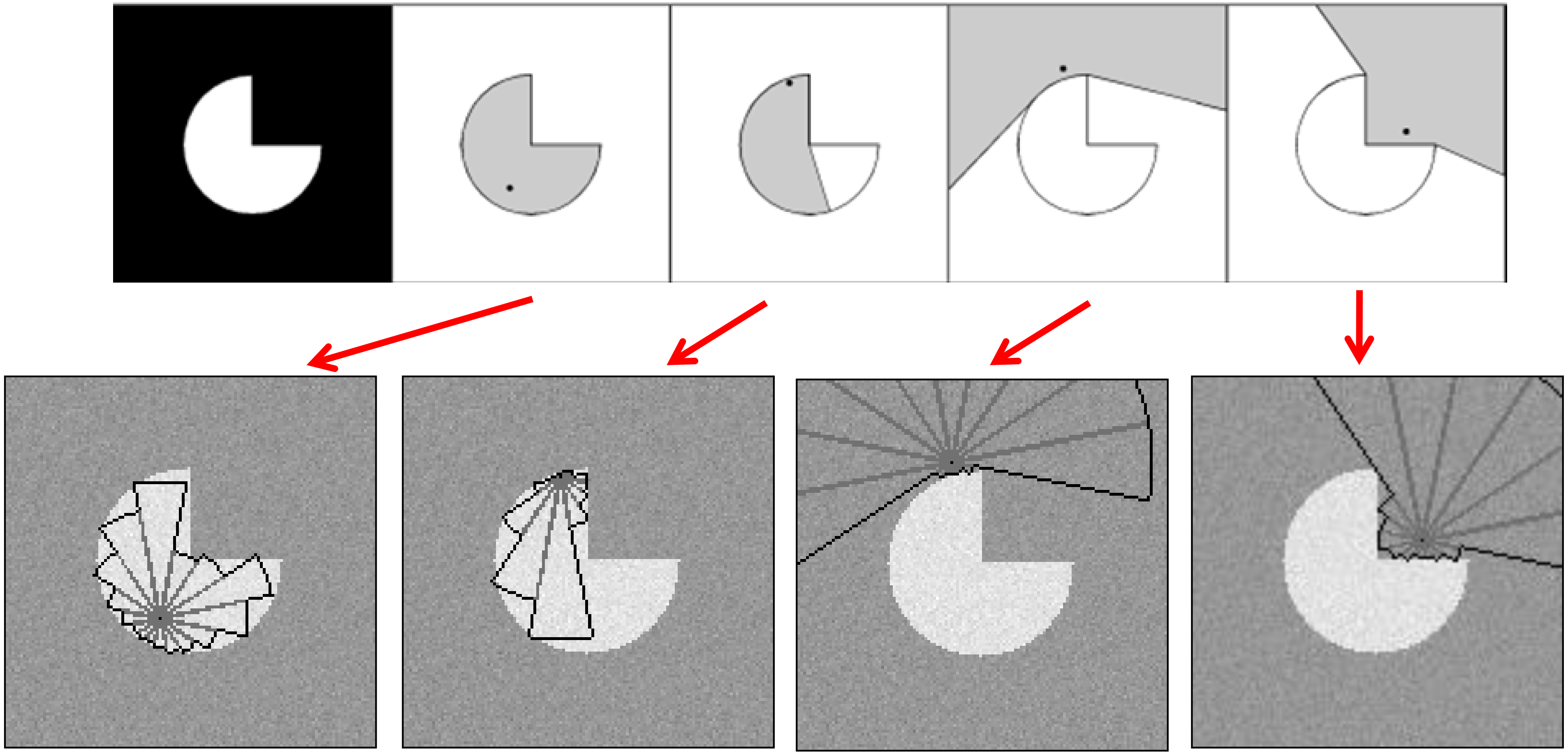


Discrete Adaptive Neighborhood

Local Approximations in Signal and Image Processing http://www.cs.tut.fi/~lasip/

Ideal neighborhood – an illustrative example

Each point has an ideal neighborhood that defines the support of pointwise Least Square Estimator for the reference point.



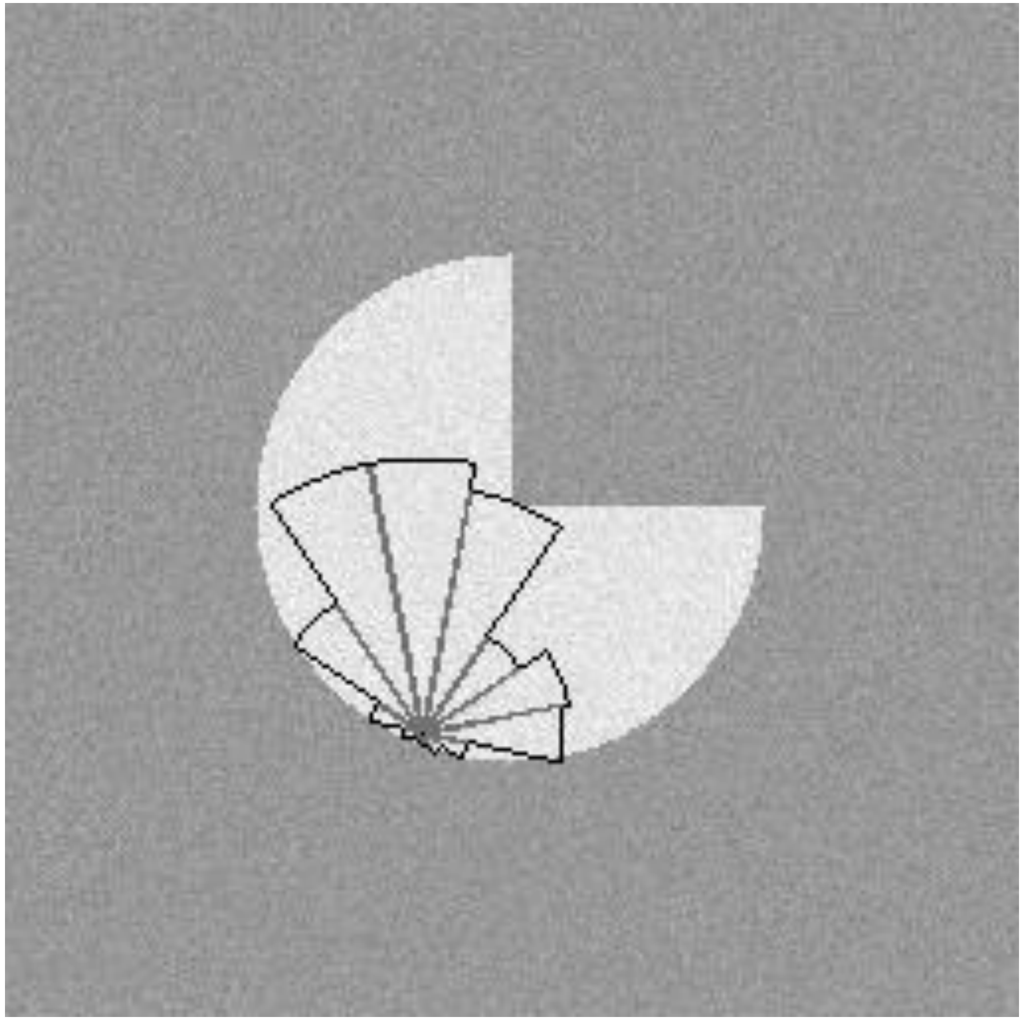
Local Approximations in Signal and Image Processing http://www.cs.tut.fi/~lasip/

Examples of Adaptively Selected Neighorhoods

Define, $\forall x \in X$, the "ideal" neighborhood $\widetilde{U}_x$ where to perform regression by fitting a given polynomial

Compute the denoised estimate at x by "using" only pixels in $\widetilde{U}_x$

$$\hat{y}(x) = \mathrm{E}[z \,|x, \widetilde{U}_x]$$



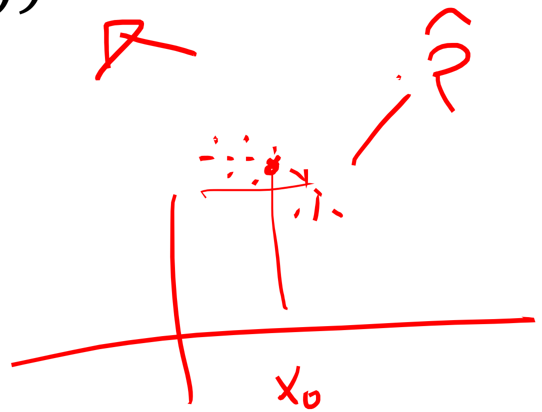
Local Polynomial Approximation

The problem: Estimate the **polynomial** $\boldsymbol{p}$ of degree m **that minimizes a weighted loss** over noisy observations in a neighborhood of x_0

$$\hat{p} = \underset{p \in \mathcal{P}_m}{\operatorname{argmin}} \sum_{x_s \in \mathcal{X}} w(x_0 - x_s)(z(x_s) - p(x_s))^2$$

$$\text{where } w = \{w(x)\} \text{ s.t.} \sum_{x_s \in \mathcal{X}} w(x_s) = 1$$

$$\hat{y}(x_0) = \hat{p}(x_0)$$



This is directly solved by **convolution** of z against a **filter** g **whose coefficients depend on the polynomial degree** m and the **weights** w

$$\hat{y}(x_0) = (z \circledast g)(x_0)$$

Katkovnik, V., K. Egiazarian, J. Astola, "Adaptive window size image de-noising based on intersection of confidence intervals (ICI) rule", J. of Math. Imaging and Vision, 2002.

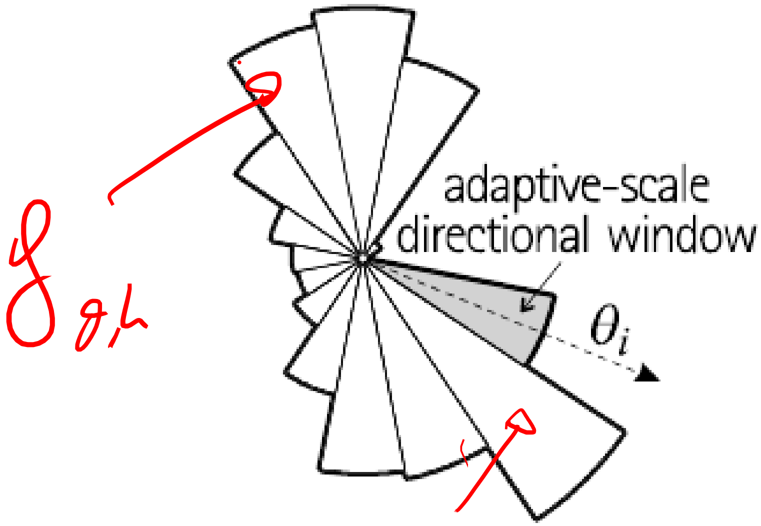
Local Polynomial Approximation

To achieve spatial adaptation we can consider multiple windows which are obtained by rotation and scaling of a basic window w

Let $w_{h,\theta} = w_\theta\left(\frac{\cdot}{h}\right)$ the scaled window over the direction θ

Consider multiple directions $\theta \in \Theta$ and scales $h \in H$ to define a collection of filters $g_{\theta,h}$ yielding a batch of estiamtes

$$\hat{y}_{\theta,h}(x_0) = \left(z \circledast g_{\theta,h}\right)(x_0)$$



Adaptation is performed by selecting, along each direction θ, the best scale h^*

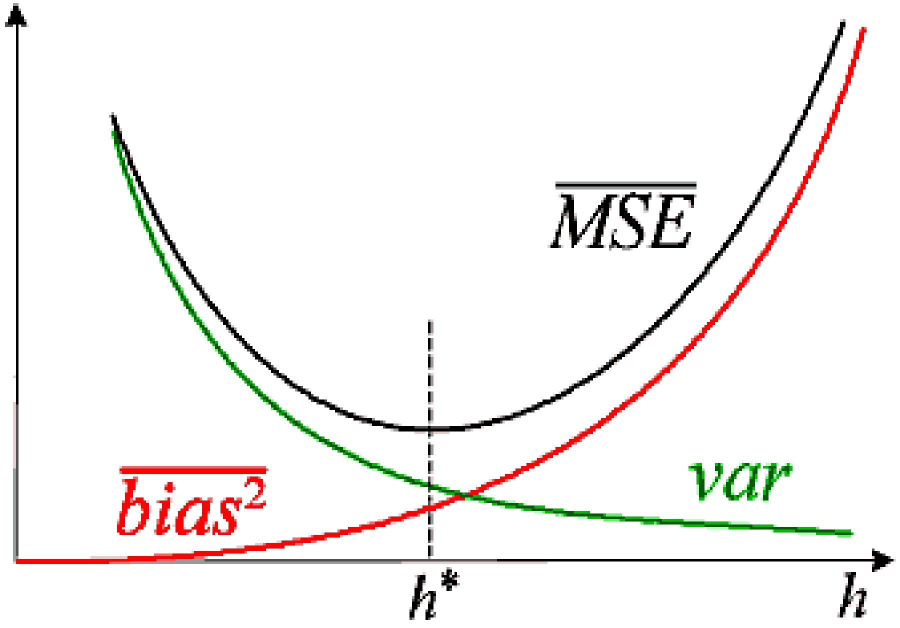
Katkovnik, V., K. Egiazarian, J. Astola, "Adaptive window size image de-noising based on intersection of confidence intervals (ICI) rule", J. of Math. Imaging and Vision, 2002.

Inersection of Confidence Inverval Rule

Inersection of Confidence Inverval (ICI) Rule is an adaptive scale selection criteria from statistical literature.

The scale parameter h controls the trade-off between bias and variance in the LPA estimates.

- Large h corresponds to a large window and smooth estimates, with lower variance and typically increased estimation bias.
- A small h corresponds to noisier estimates, less biased, and with higher variance.



Katkovnik, V., K. Egiazarian, J. Astola, "Adaptive window size image de-noising based on intersection of confidence intervals (ICI) rule", J. of Math. Imaging and Vision, 2002.

ICI Rule

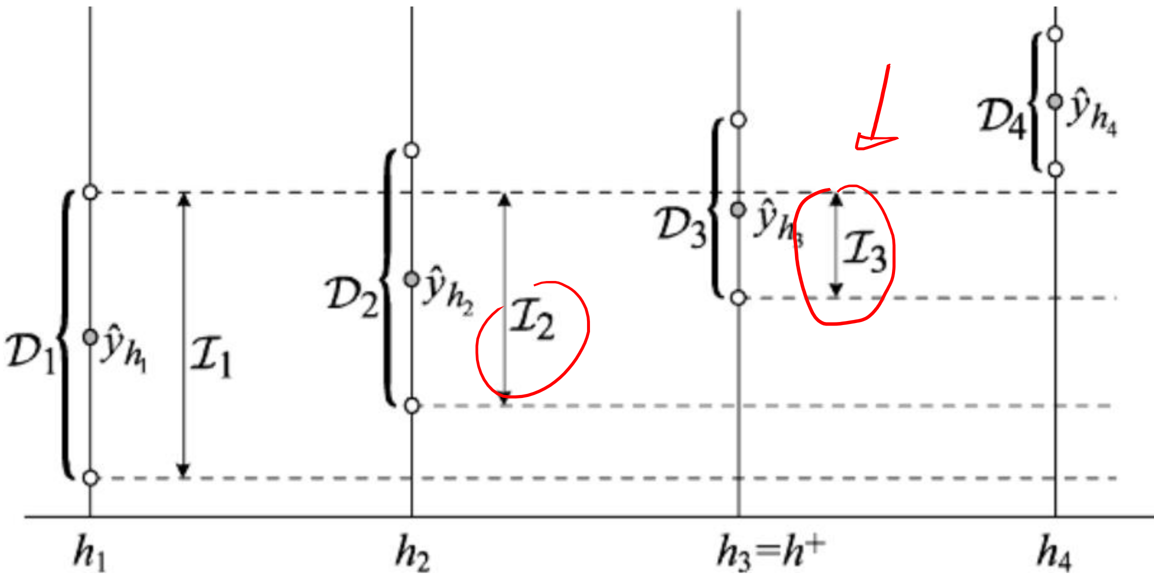
σ is fixed

Consider the intersection $\mathcal{I}_j$ of all the confidence intervals D_j

$$\mathcal{I}_j = \bigcap_{i \leq j} D_j$$

Where $D_i = [\hat{y}_i(x_0) - \Gamma\sigma_i,\ \hat{y}_i(x_0) + \Gamma\sigma_i]$ and σ_i is the standard deviation of the estimator associated to $\hat{y}_i$, and $\Gamma > 0$ is a tuning parameter.

Then, the ICI rule selects the largest scale j^+ such that $\mathcal{I}_{j^+}$ is not empty.

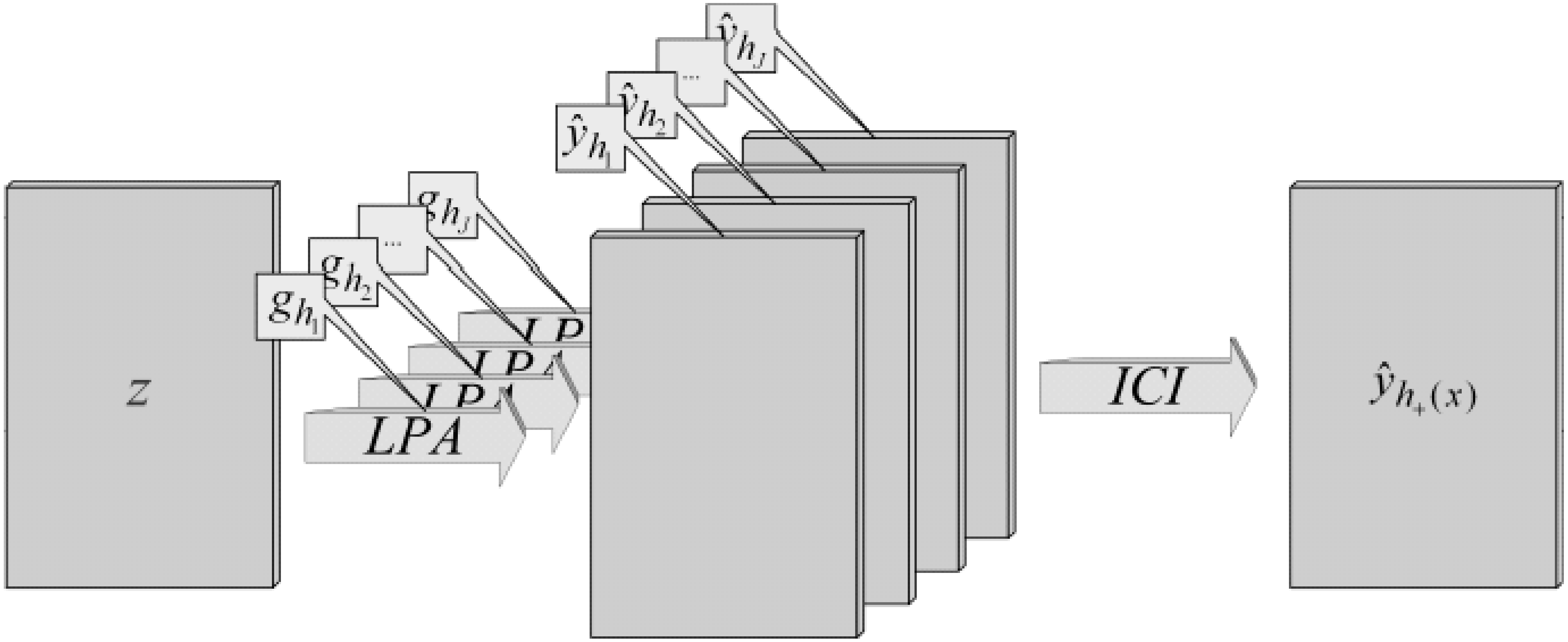


Katkovnik, V., K. Egiazarian, J. Astola, “Adaptive window size image de-noising based on intersection of confidence intervals (ICI) rule”, J. of Math. Imaging and Vision, 2002.

LPA – ICI Rule

ICI is applied independently for each direction, yielding the adaptive scale (length) of the corresponding sector.

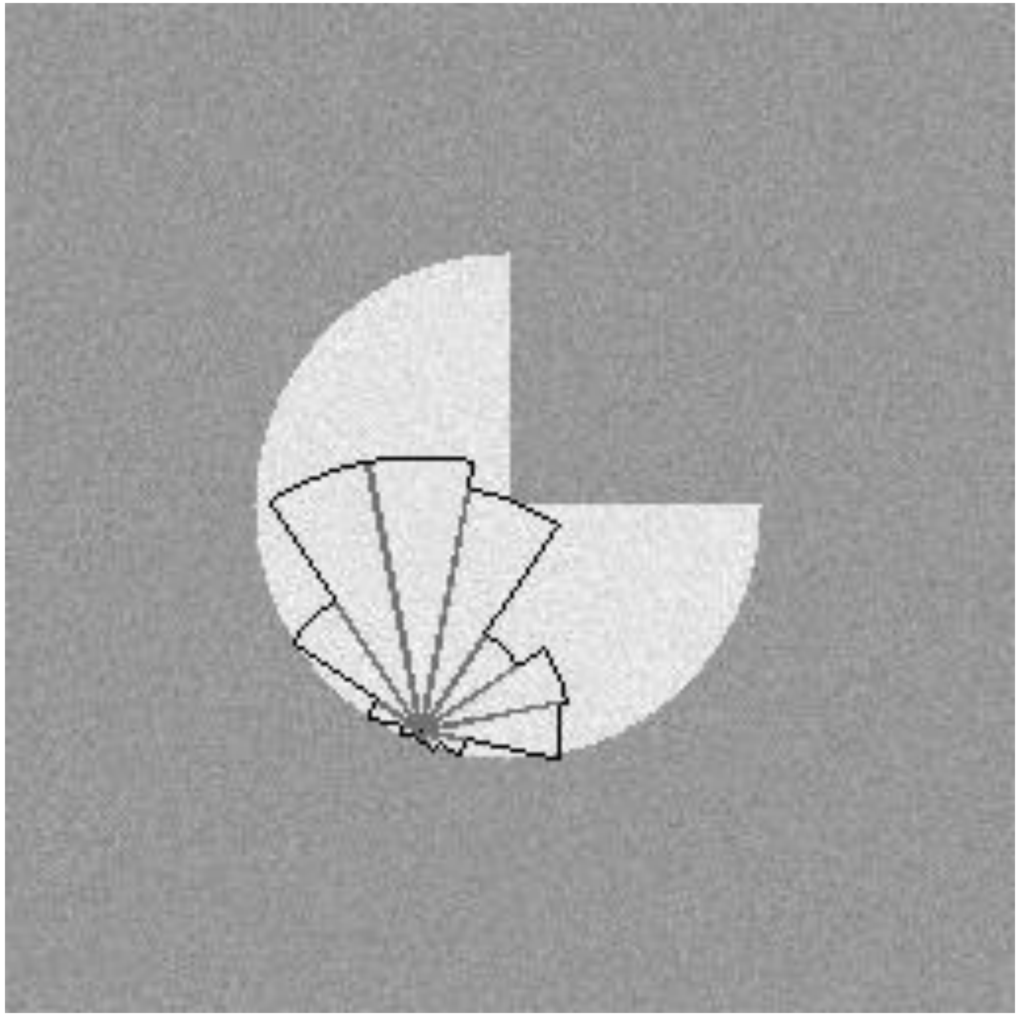
∀θ



Local Approximations in Signal and Image Processing http://www.cs.tut.fi/~lasip/

Examples of adaptively selected neighorhoods

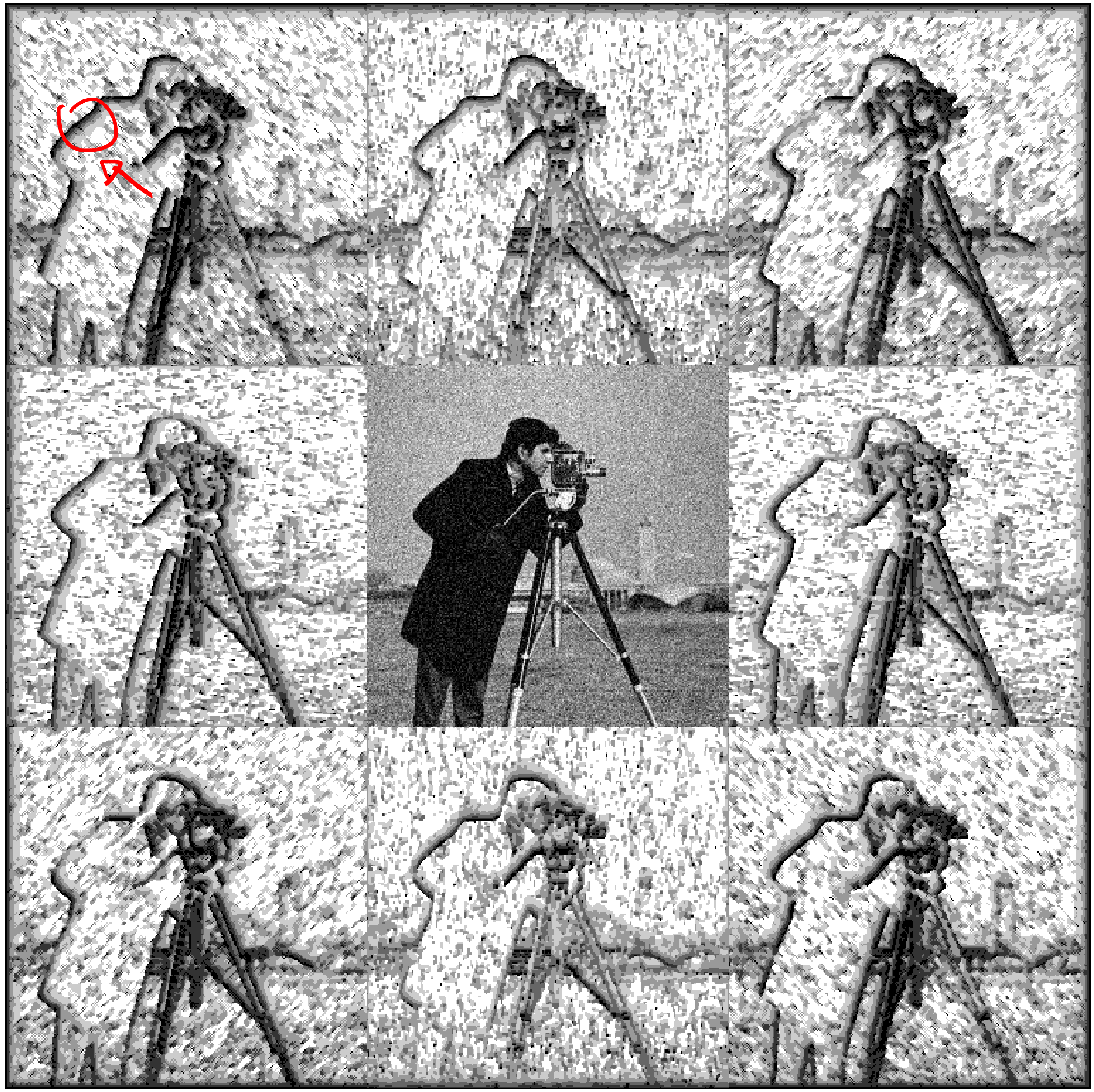
Neighborhoods adaptively selected using the LPA-ICI rule



Local Approximations in Signal and Image Processing http://www.cs.tut.fi/~lasip/

Adaptive Scales

The adaptive scales reveal the distribution of features (such as edges) across the corresponding direction.



Local Approximations in Signal and Image Processing http://www.cs.tut.fi/~lasip/

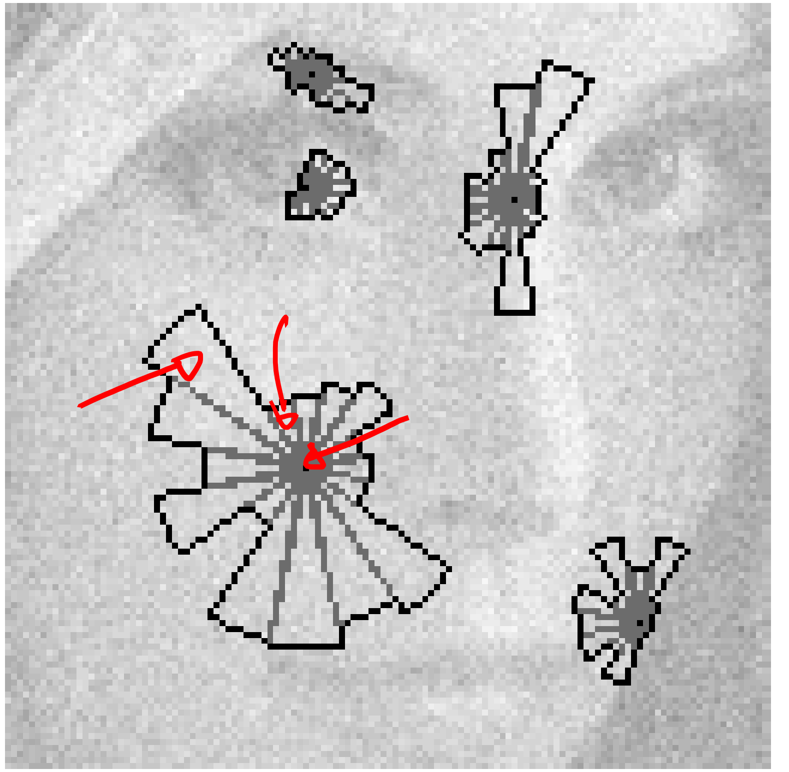
Final Estimate

Directional adaptive-scale estimates are then *fused* together into the anisotropic estimate

$$\hat{y}(x_0) = \sum_{\theta \in \Theta} \lambda(x,\theta)\, \hat{y}_{h^+,\theta}(x)$$

And aggregation weights $\lambda(x,\theta)$ are inversely proportional to the variance of the estimators

$$\lambda(x,\theta) = \frac{\sigma^{-2}(h^+,\theta,x)}{\sum_\theta \sigma^{-2}(h^+,\theta,x)}$$



Local Approximations in Signal and Image Processing http://www.cs.tut.fi/~lasip/

Final Estimate

The directional adaptive-scale estimates are fused together to obtain the final *anisotropic* estimate (shown in the center).

Example of Performance

Original, noisy, denoised using polynomial regression on adaptively defined neighborhoods (LPA-ICI)

original

noisy

Parametric Approaches

Denoising Approaches

Parametric Approaches

- Assume the noisy-free signal y features some **sparsity property** in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition. This implies that your image admits a **global parametric representation**

$$y = \sum_{i=1}^{L} x_i \underline{e}_i$$

Sparsity as a Denoising Prior

Sparsity and Parsimony

The principle of sparsity or “parsimony” consists in *representing some phenomenon with as few variable as possible*

Stretch back to philosopher William Ockham in 14th Century

Wrinch and Jeffreys [1921] relate simplicity to parsimony:

> *The **existence of simple laws** is, then, apparently, to be regarded as **a quality of nature**; and accordingly we may infer that it is justifiable to **prefer a simple law to a more complex one that fits our observations slightly better**.*

Simplicity is the number of learning parameters

Sparsity in Statistics

Statistics: simple models are preferred.

Sparsity is used to **prevent overfitting and improve interpretability of learned models.** LASSO

In model fitting, the number of parameters is typically used as a criterion to perform model selection.

See Bayes Information Criterion (BIC), Akaike Information Criterion (AIC), …., Lasso.

Sparsity in Signal Processing

Signal Processing: similar concepts but different terminology.

Vectors corresponds to signals and **data modeling is crucial** for **solving inverse problems.**

A very successful **prior** is to assume that **images are approximated by sparse linear combinations** of **prototypes** (basis elements / atoms of a dictionary), resulting in simpler and compact model.

Noise, having no structure, **is not expected to be sparse.**

$z \longmapsto z \in \mathbb{R}^d$ vector space

How to promote sparsity?

Sparisity does not make much sense in space domain → move to a different domain

256

256

cameraman

Sparse cameraman (only 50% of pixels are preserved)

Basic Notions for Transformed-Domain Image Processing

Giacomo Boracchi

Preliminary Notation: Linear Independence

A set of vectors $\{\boldsymbol{e_i}\}_{i=1,\dots,N}$ living in a vector space V is said to be linearly independent if

$$\sum_i x_i \boldsymbol{e}_i = \mathbf{0} \Leftrightarrow x_i = 0 \ \forall i$$

Preliminary Notation: Basis

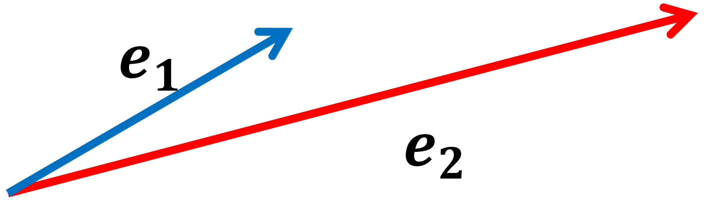
A basis $\{\boldsymbol{e_i}\}_{i=1,\dots,d}$ of a vector space V is a set of linearly independent vectors that spans the whole vector space V, namely:

$$\forall\, \boldsymbol{v} \in V, \exists \{x_i\}_{i=1,\dots,d} \;\; \text{s.t. } \boldsymbol{v} = \sum_i x_i \boldsymbol{e}_i$$

d is said the dimension of V (potentially it could not be finite)

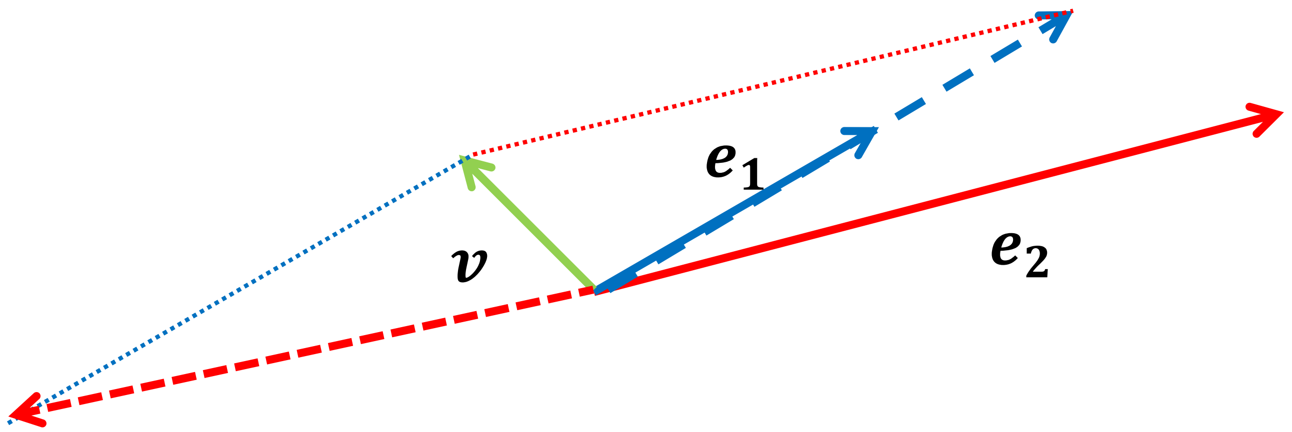
Example

Th two vectors $\{\boldsymbol{e_1}, \boldsymbol{e_2}\}$ represent a basis for the plane



Example

Th two vectors $\{\boldsymbol{e_1}, \boldsymbol{e_2}\}$ represent a basis for the plane

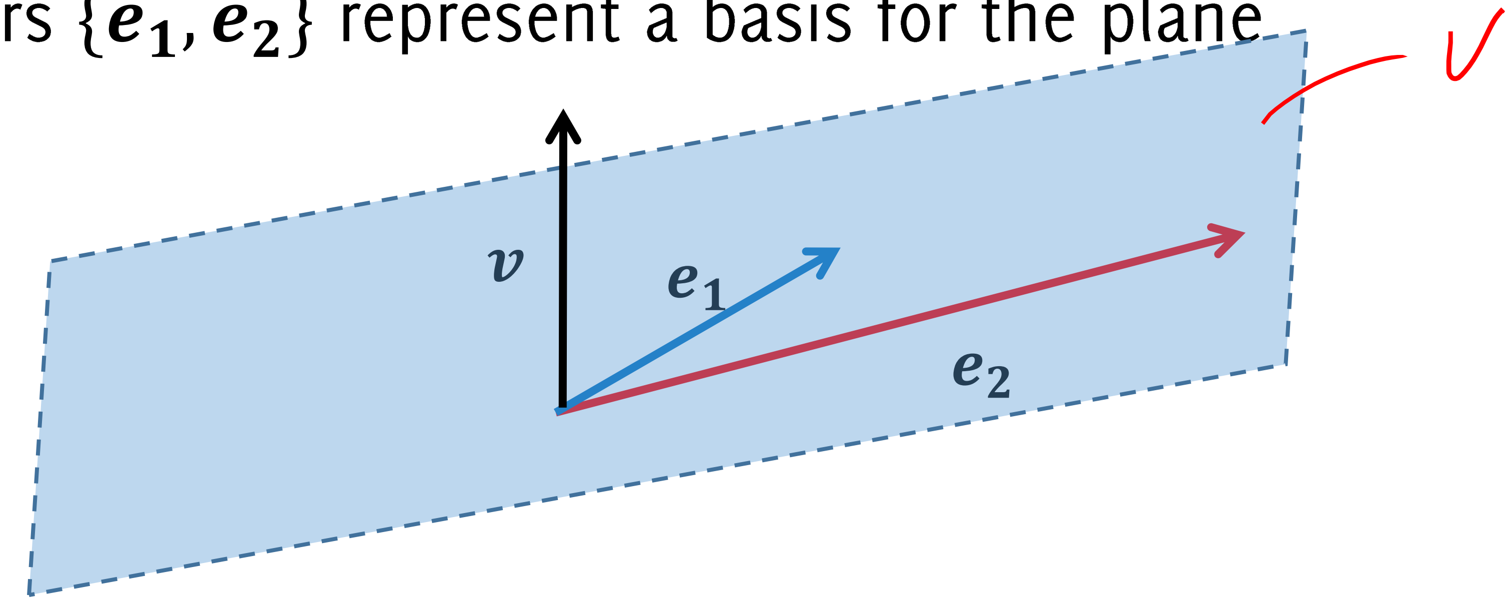


Since any vector in this plane can be written as a linear combination of these two

$$\boldsymbol{v} = 1.5 * \boldsymbol{e_1} + (-1) * \boldsymbol{e_2}$$

Example

Th two vectors $\{\boldsymbol{e_1}, \boldsymbol{e_2}\}$ represent a basis for the plane



This holds for any vector except the orthogonal ones

A very important property

Let $\{\boldsymbol{e_i}\}_{i=1,\dots,d}$ be a set of linearly independent generators of V then:

$$\forall\, \boldsymbol{v} \in V, \exists \{x_i\}_{i=1,\dots,d} \;\; \text{s.t.}\;\; \boldsymbol{v} = \sum_i x_i \boldsymbol{e}_i$$

And such representation **is unique**

$$\boldsymbol{v} \rightarrow \{x_i\}_{i=1,\dots,d}$$

signal ↦ coefficients

Proof: follows trivially from the definition of linear independence

A very important property

Let $\{\boldsymbol{e_i}\}_{i=1,\dots,d}$ be a set of linearly independent generators of V then:

$$\forall\, \boldsymbol{v} \in V, \exists \{x_i\}_{i=1,\dots,d} \ \text{ s.t. } \boldsymbol{v} = \sum_i x_i \boldsymbol{e}_i$$

And such representation **is unique**

$$\boldsymbol{v} \to \{x_i\}_{i=1,\dots,d}$$

Proof: follo

Ok, we know that this represenation exists and is unique

$$\forall\, \boldsymbol{v} \in V, \exists \{x_i\}_{i=1,\dots,d} \ \text{ s.t. } \boldsymbol{v} = \sum_i x_i \boldsymbol{e}_i$$

.. But given an input $\boldsymbol{v}$, how to compute $\boldsymbol{v} \to \{x_i\}_{i=1,\dots,d}$

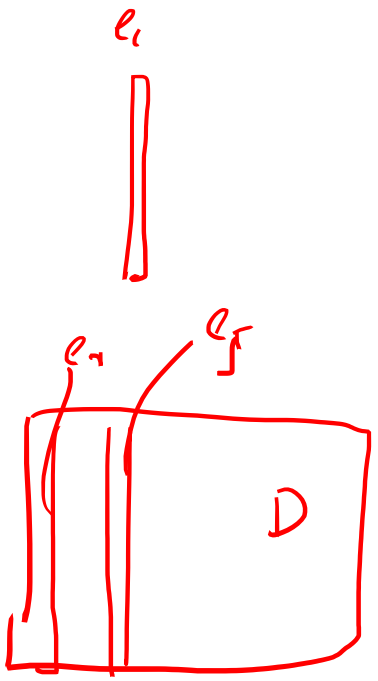
Orthonormal Basis in Euclidean Space

An orthonormal basis is basis $\{\boldsymbol{e_i}\}_{i=1,\ldots,d}$ is such that

$$\boldsymbol{e_i}^\top \boldsymbol{e_j} = \delta_{i,j}$$

The advantage of using orthonormal basis is that

$$\forall\, \boldsymbol{v} \in V, \exists \{x_i\}_{i=1,\ldots,d} \;\; \text{s.t. } \boldsymbol{v} = \sum_i x_i \boldsymbol{e}_i$$



and

$$x_i = \boldsymbol{e_i}^\top \boldsymbol{v} = \boldsymbol{v}^\top \boldsymbol{e_i}$$

If we arrange the basis $\{\boldsymbol{e_i}\}_{i=1,\ldots,d}$ in the columns of a $d \times d$ matrix D

$$\boldsymbol{x} = \boldsymbol{D}^\top \boldsymbol{v}$$

Ok, let's go back to images

Which is the basis we used for representing digital images so far?

$v = \sum x_i e_i$?

$\boldsymbol{v} =$

116	23	33
16	3	73
5	4	30

Ok, let's go back to images

Which is the basis we used for representing digital images so far?

$\boldsymbol{v} =$

116	23	33
16	3	73
5	4	30

$\boldsymbol{e_1} =$

1	0	0
0	0	0
0	0	0

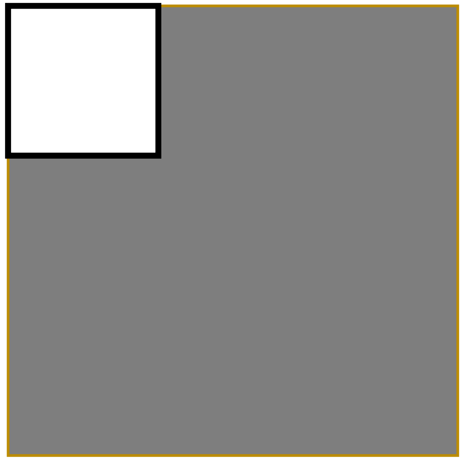
The canonical basis of $\mathbb{R}^d$

Which is the basis we used for representing digital images so far?

$\boldsymbol{v} =$

116	23	33
16	3	73
5	4	30

$\boldsymbol{e_1} =$



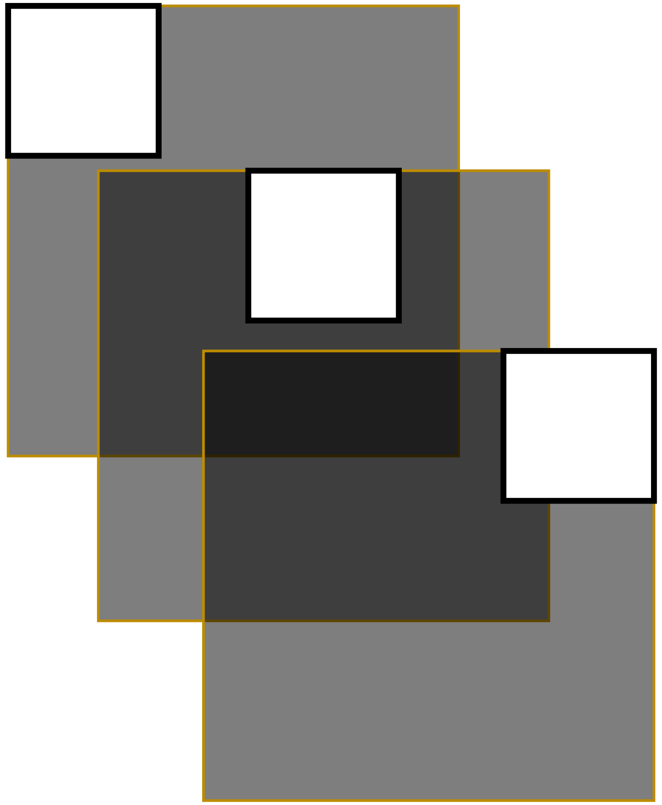
The canonical basis of $\mathbb{R}^d$

Which is the basis we used for representing digital images so far?

$\boldsymbol{v} =$

116	23	33
16	3	73
5	4	30

$\boldsymbol{e_1} =$

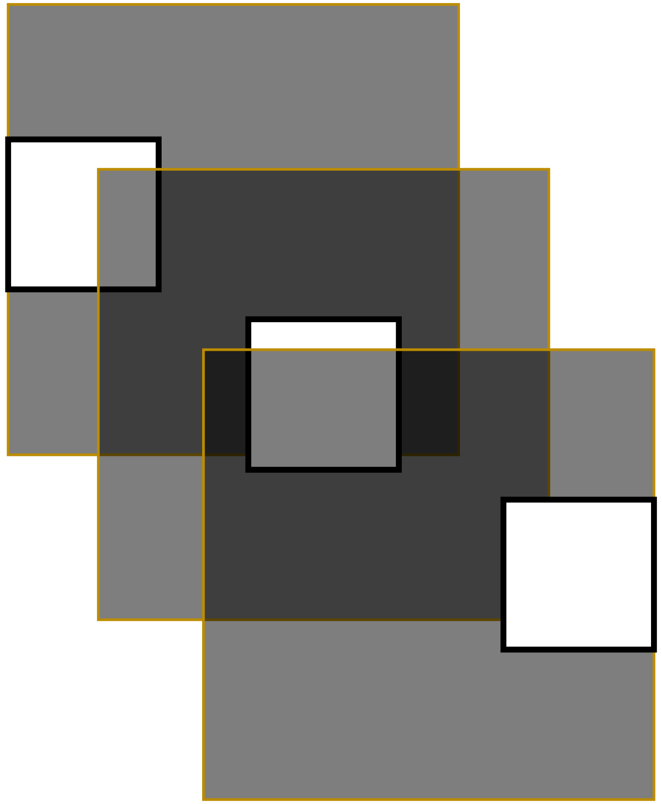


The canonical basis of $\mathbb{R}^d$

Which is the basis we used for representing digital images so far?

$$\boldsymbol{v} = \begin{bmatrix} 116 & 23 & 33 \\ 16 & 3 & 73 \\ 5 & 4 & 30 \end{bmatrix}$$

$\boldsymbol{e_1} =$



The canonical basis of $\mathbb{R}^d$

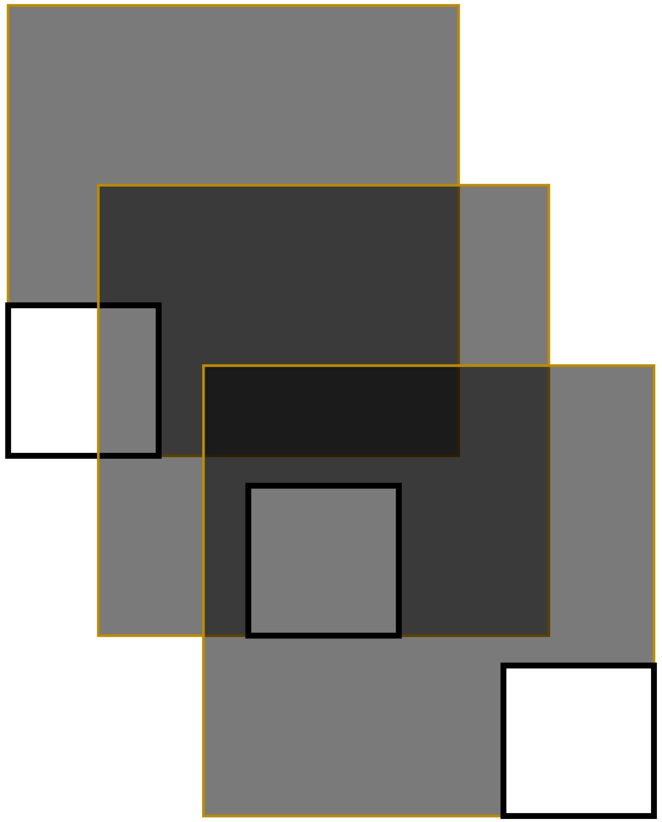
Which is the basis we used for representing digital images so far?

$116 \cdot e_1 + 23\, e_2 \ldots + 33\, e_3 + \ldots$

$v =$

116	23	33
16	3	73
5	4	30

$e_1 =$



Thus the canonical basis

Canonical basis

$$\{\boldsymbol{e_i}\}_{i=1,\dots,d}$$

Being $\boldsymbol{e_j} = \text{zeros}(1, d)$; $\boldsymbol{e_j}(j) = 1$

Uses each coefficient to represent a pixel:

- all coefficients are equally meaningful
- thus, it is not usefull at all for compression

$\boldsymbol{e_1} =$

1	0	0
0	0	0
0	0	0

Are there basis that ease image processing tasks?

Fourier Transform

2D Fourier Transform

The (u,v)-element of the 2D Fourier basis is defined as

$$e^{-i2\pi(ux+vy)} = \cos\big(2\pi(ux+vy)\big) + i\sin\big(2\pi(ux+vy)\big)$$

Each Fourier coefficient is computed with an inner product with the correspondinf function.

The Fourier basis functions are constant where $y = -\frac{ux}{c} + c$

The Fourier basis functions have unlimited support.

The Fourier Transform is invertible (it is an orthonormal transform)

Fourier domain is also called frequency domain.

2D Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is defined as

$$F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\pi i\left(\frac{km}{M}+\frac{ln}{N}\right)}$$

The Fourier Transform admit a fast implementation (FFT) when the signal/image sizes are powers of 2.

2D Fourier Basis Elements

Frequency and Orientation of the 2D Fourier Basis Elements

$$b_{u,v} \quad u = 1, v = 10$$

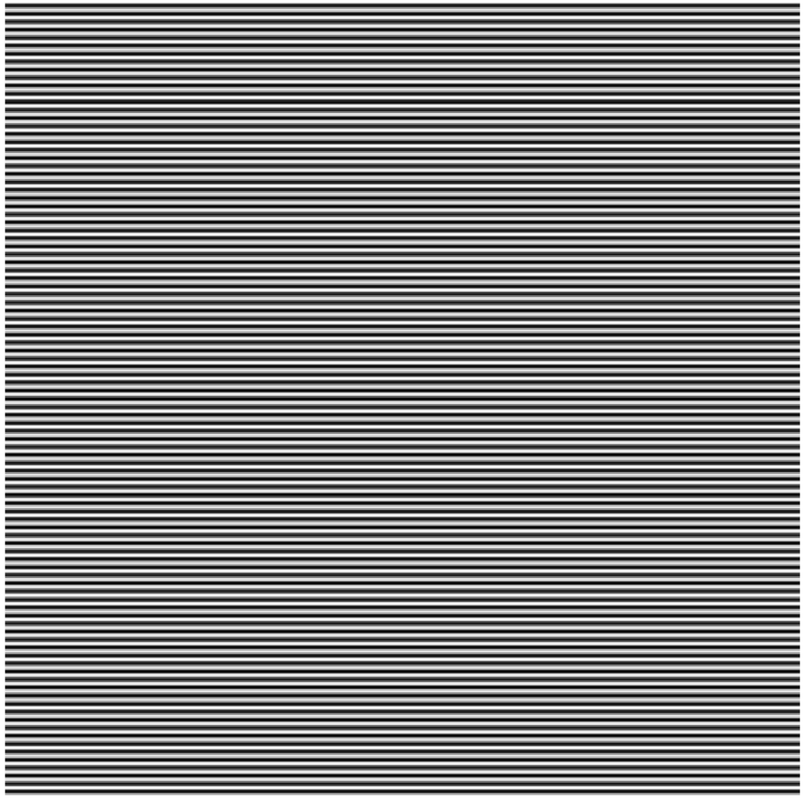
the u=1v=10Fourier domain basis element

2D Fourier Basis Elements

Frequency and Orientation of the 2D Fourier Basis Elements

$$b_{u,v} \quad u = 100, v = 1$$

the u=100v=1Fourier domain basis element



2D Fourier Basis Elements

Frequency and Orientation of the 2D Fourier Basis Elements

$$b_{u,v} \quad u = 10, v = 10$$

the u=10v=10Fourier domain basis element

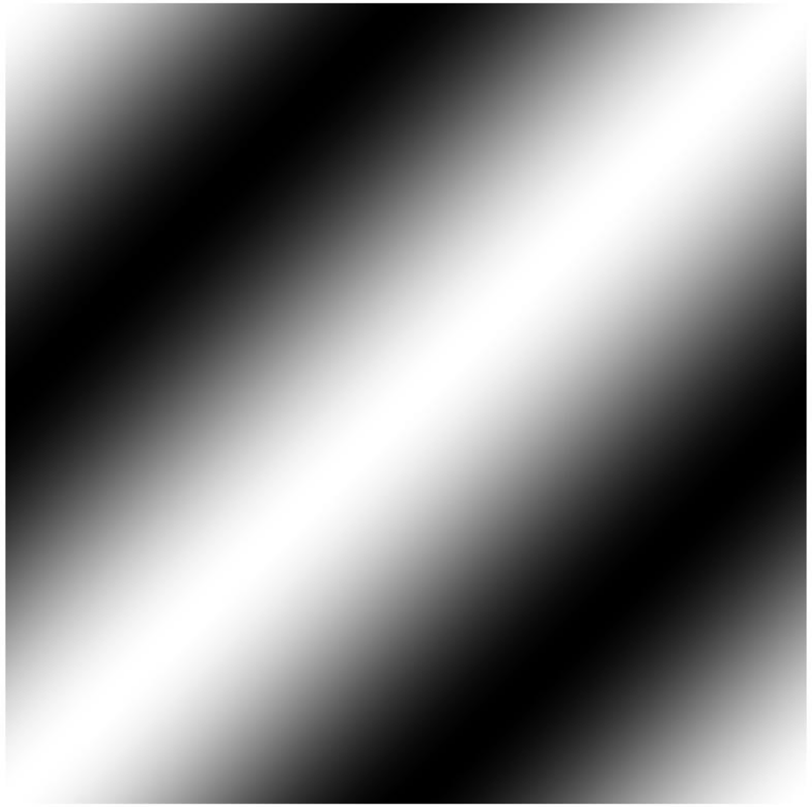
2D Fourier Basis Elements

Frequency and Orientation of the 2D Fourier Basis Elements

$$b_{u,v} \quad u = 2, v = 2$$

note that in atlab, Fourier coefficients are indexed starting from 1

the u=2v=2Fourier domain basis element



Try it in Matlab

```
%Examples of Basis Elements
Y=zeros(512);
u=10
v=10
Y(u,v)=1;
% Y is the image in Fourier Domain having only the (u,v)-coefficient set to 1 and the other set to 0
figure(1),imshow(Y,[]),title(['the u=',num2str(u), 'v=',num2str(v), 'space domain basis element'] )
 % by inverting the Fourier transform one get the (u,v)-element of the 2D discrete Fourier basis
y=ifft2(Y);
figure(2),imshow(real(y),[]),title(['the u=',num2str(u), 'v=',num2str(v), 'Fourier domain basis element'] )
```

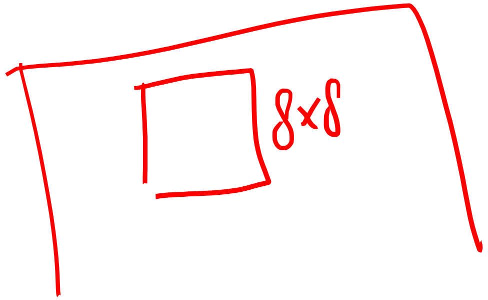
Limitations of Fourier Transform

- Fourier Transform yield complex coefficients,
- Fourier Trasformation it’s a global trasforamation (each and every pixel affects in principle each and every coefficient)

DCT

Difficult to find a parametric model describing the whole image and enabling separation between noise and the image content

- Better operate with a real-valued basis
- Better operate patch-wise, such that the parametric model applies to small portions of the image



Denoising by Thresholding in Transform Domain

Denoising by Thresholding

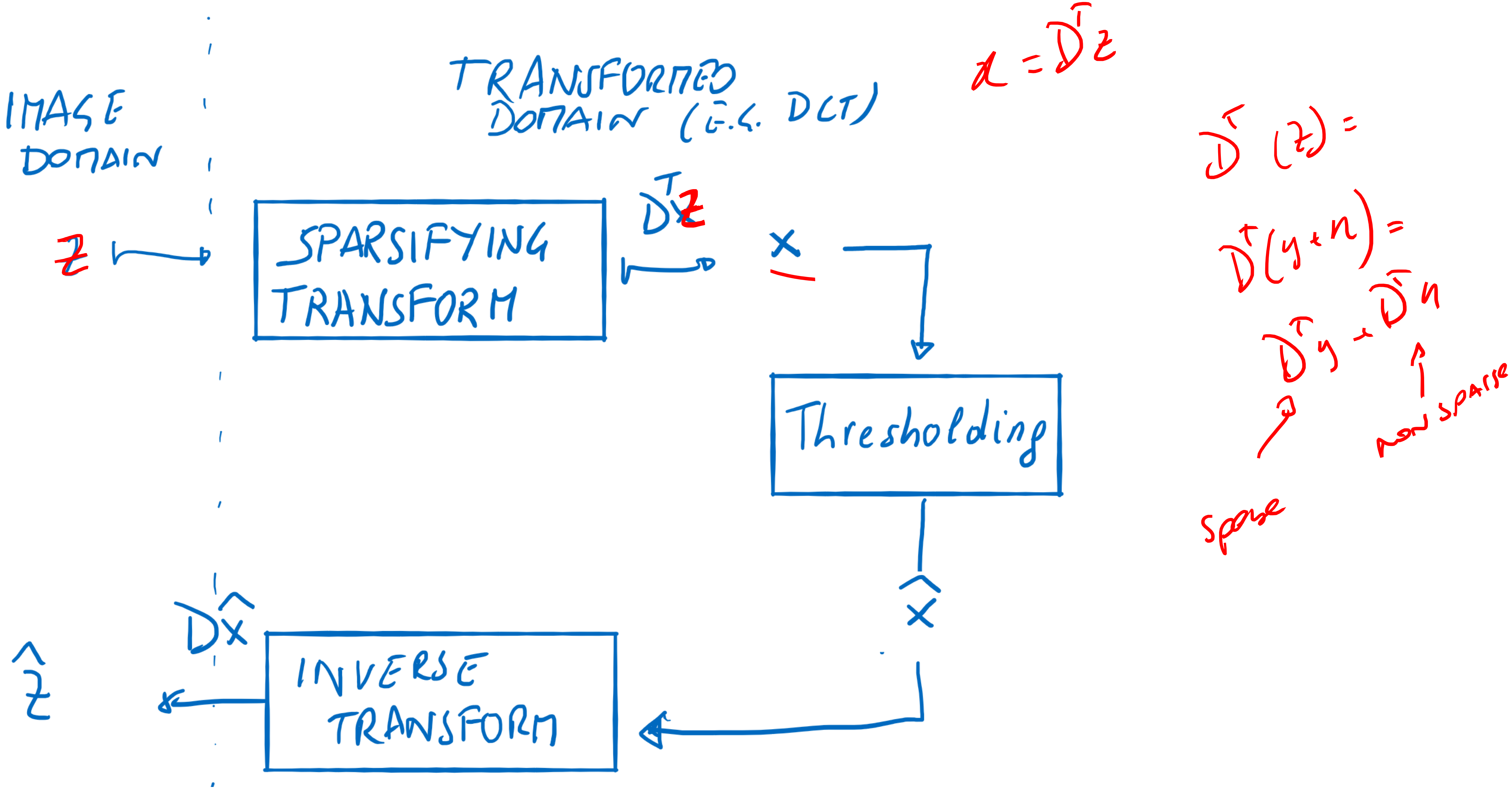
The underlying assumption is that a clean signal admits **a sparse representation w.r.t.** a suitable basis $\{\boldsymbol{e_i}\}_{i=1,\dots,d}$ (or set of generators)

Noise, being unstructred, spreads in all the coefficients

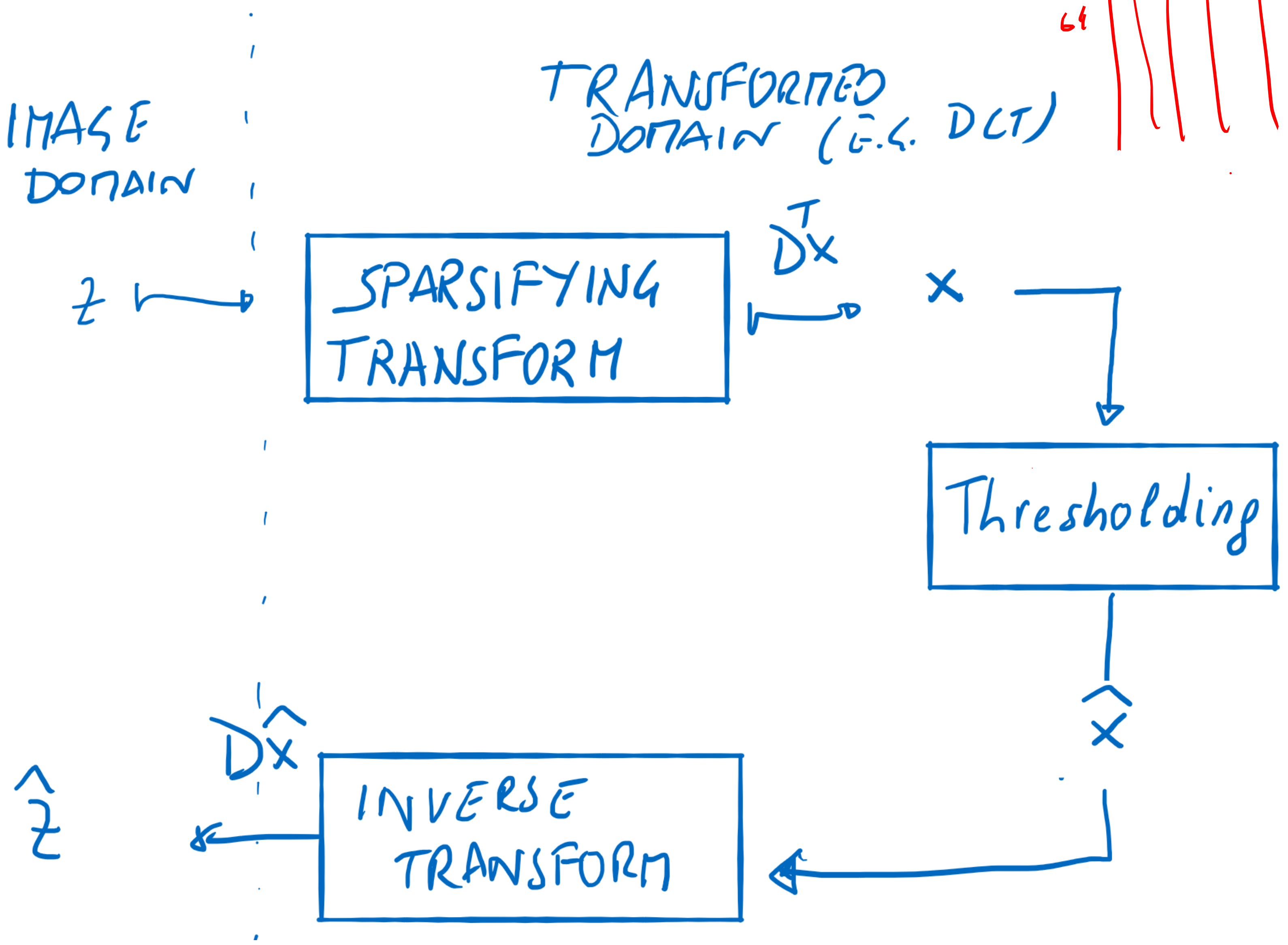
A general denoising appraoch consists in

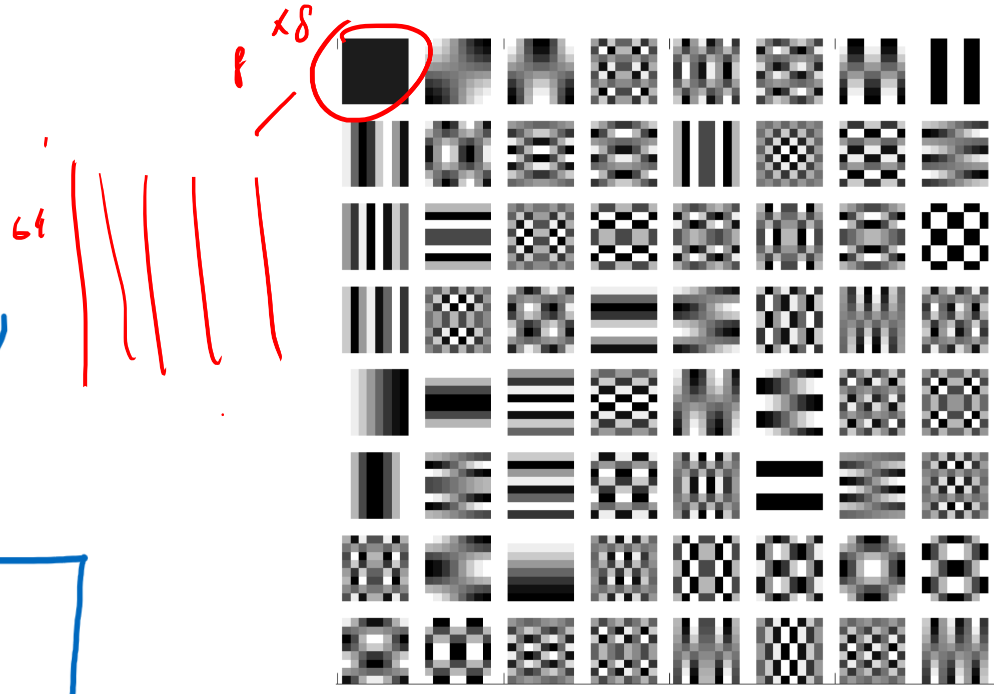
- Cropping the image patch-wise
- Transform each patch according to a sparsitying transformation
- Perform Thresholding
- Invert Transformation

Sparsity Promoting Denoising



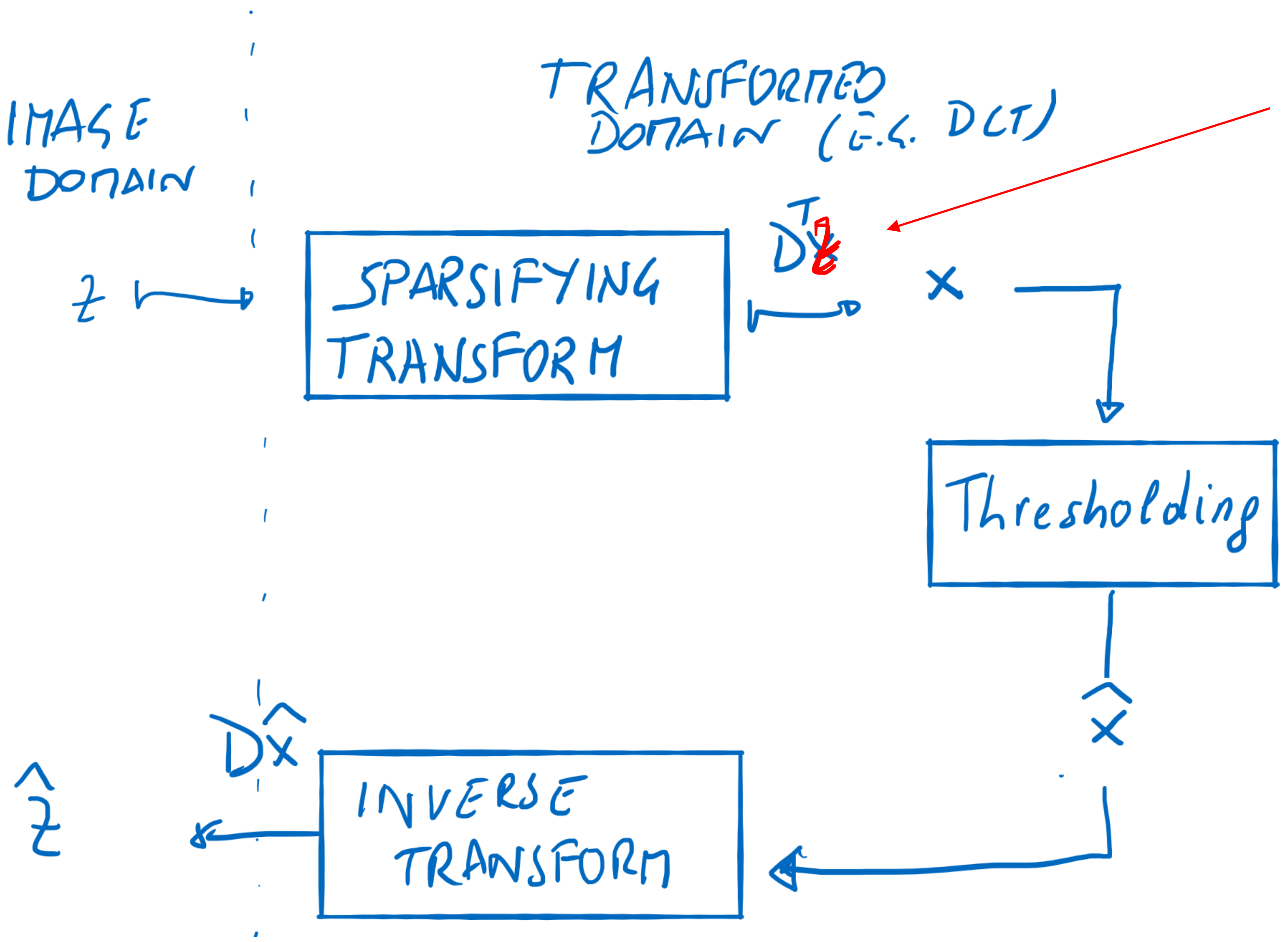
Sparsity Promoting Denoising





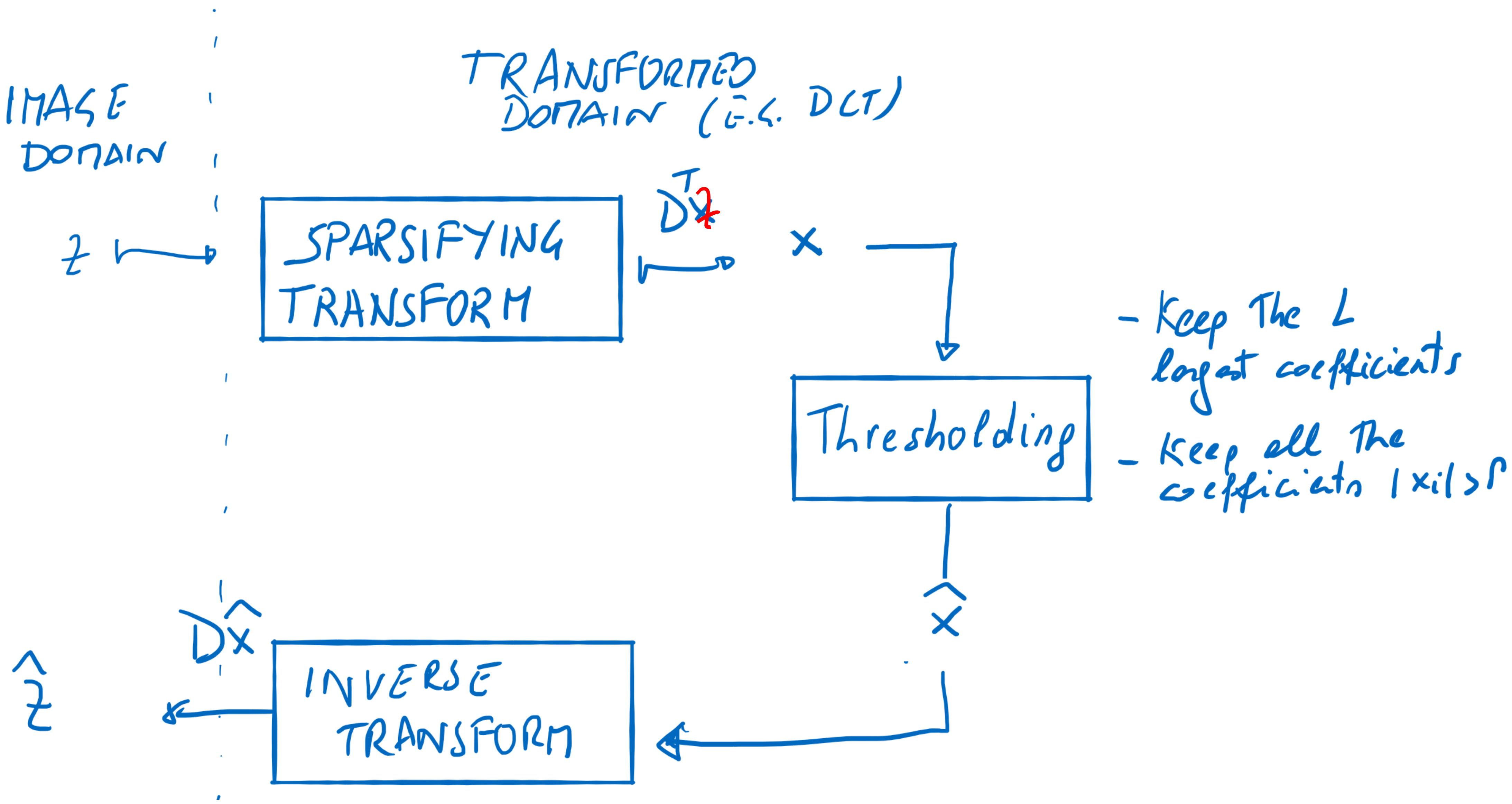
DCT: Discrete Cosine Transform is often used at patch-level. These are the elements of the basis, which are then arranged column-wise in the matrix D

Sparsity Promoting Denoising

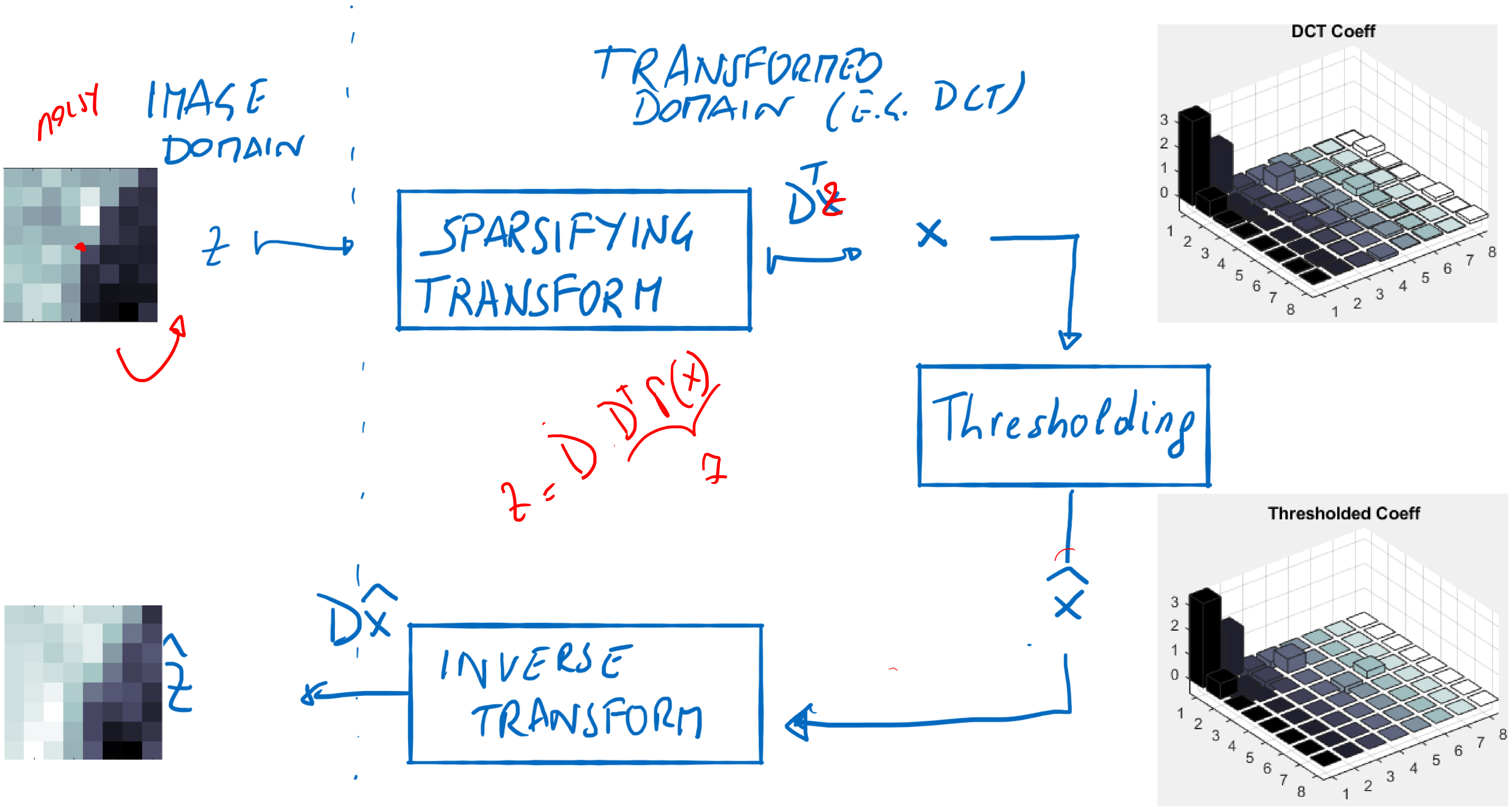


When the transformation is not w.r.t. an orthonormal basis, decomposition equation is not as simple. In case or redundant set of generators the representation is not unique and it has to «be pursued»

Sparsity Promoting Denoising

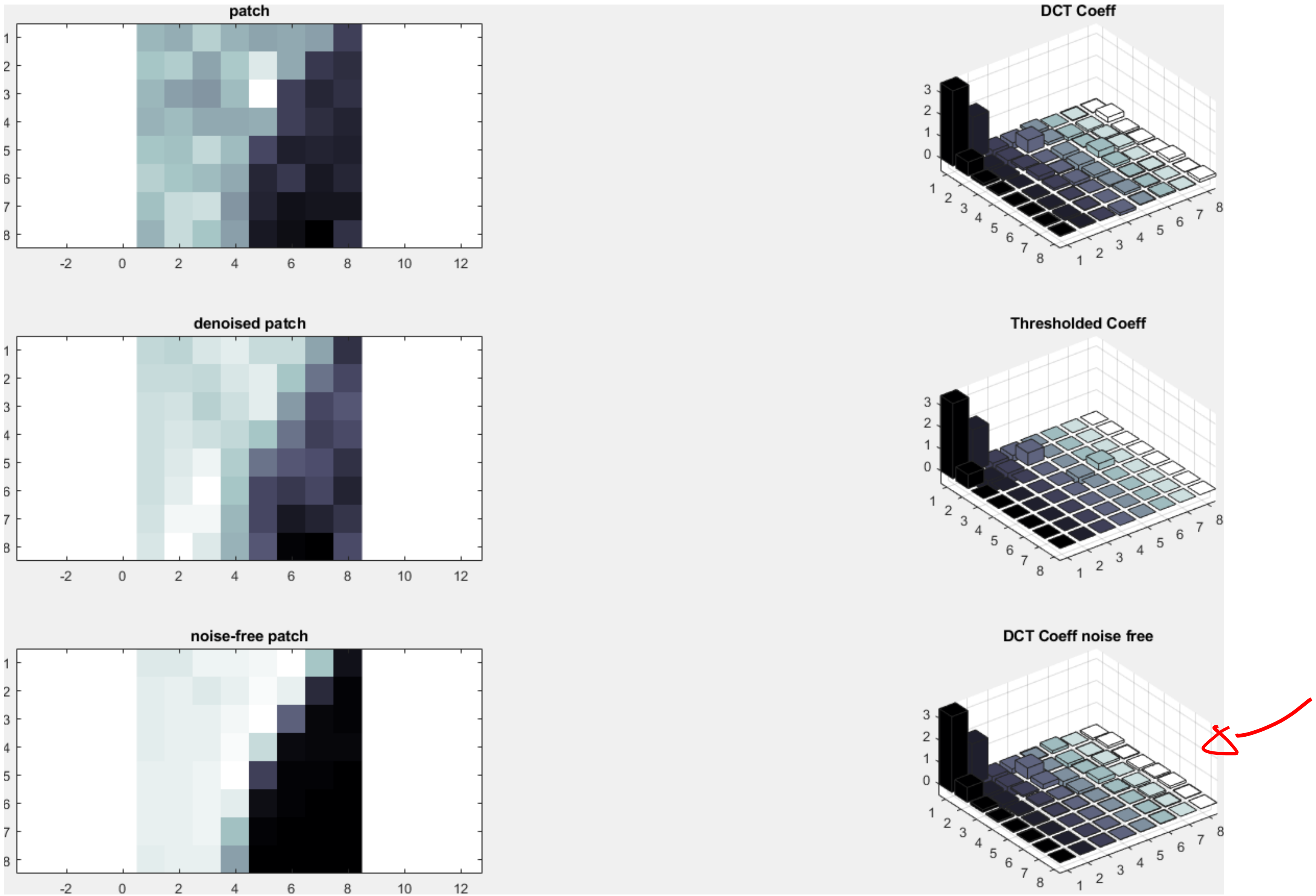


Sparsity Promoting Denoising



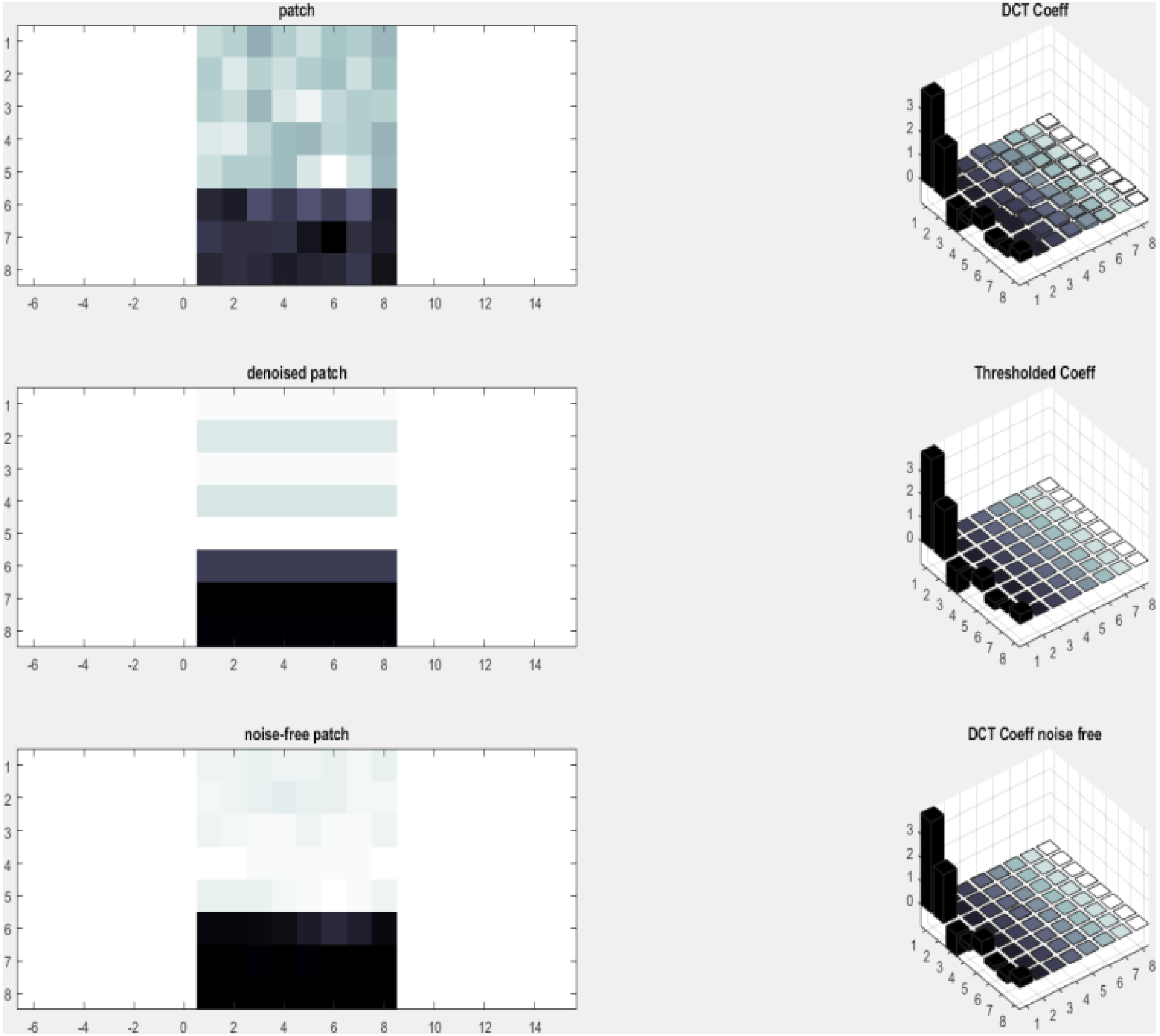
Giacomo Boracchi

Sparsity Promoting Denoising

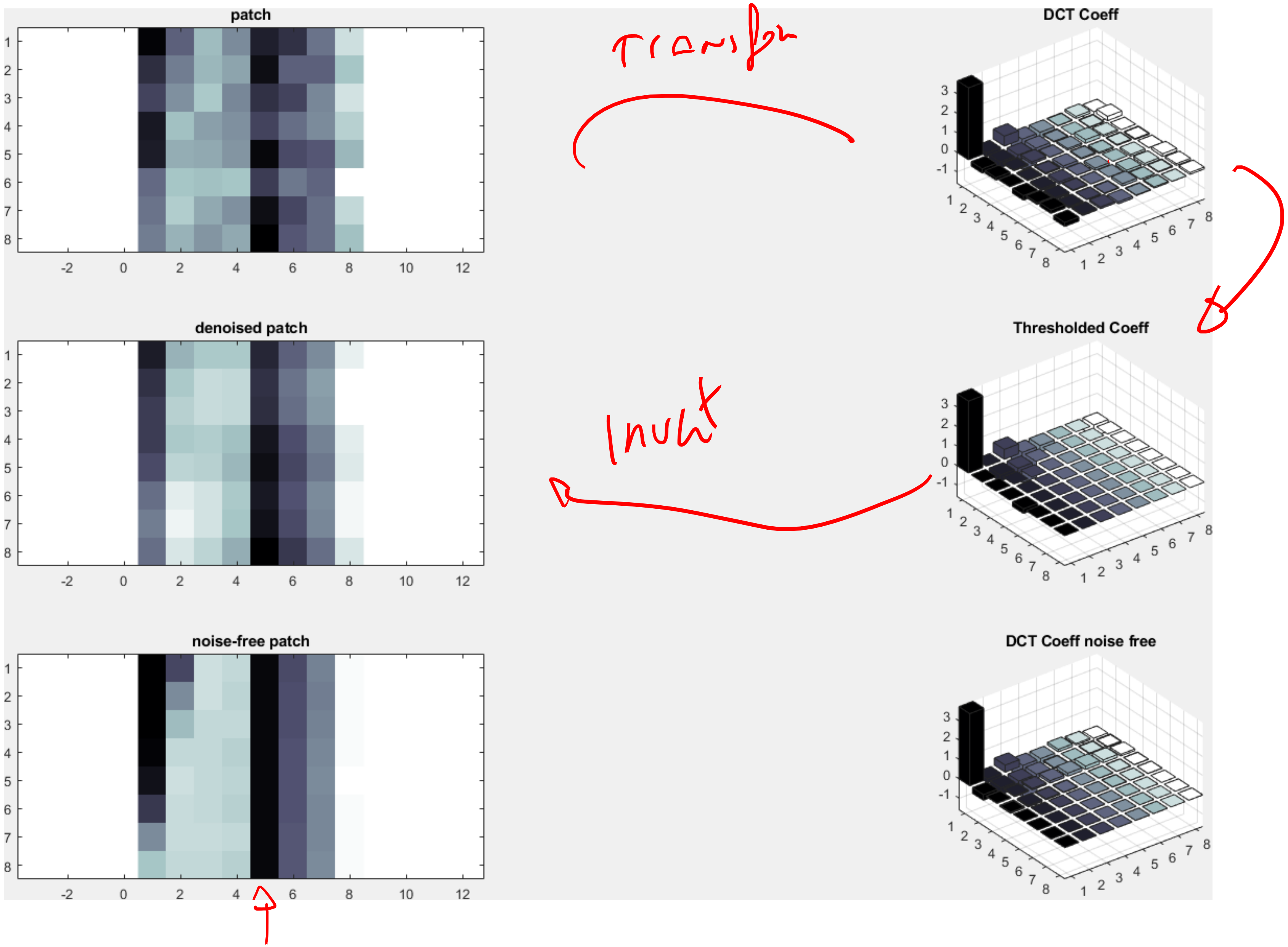


Giacomo Boracchi

Sparsity Promoting Denoising

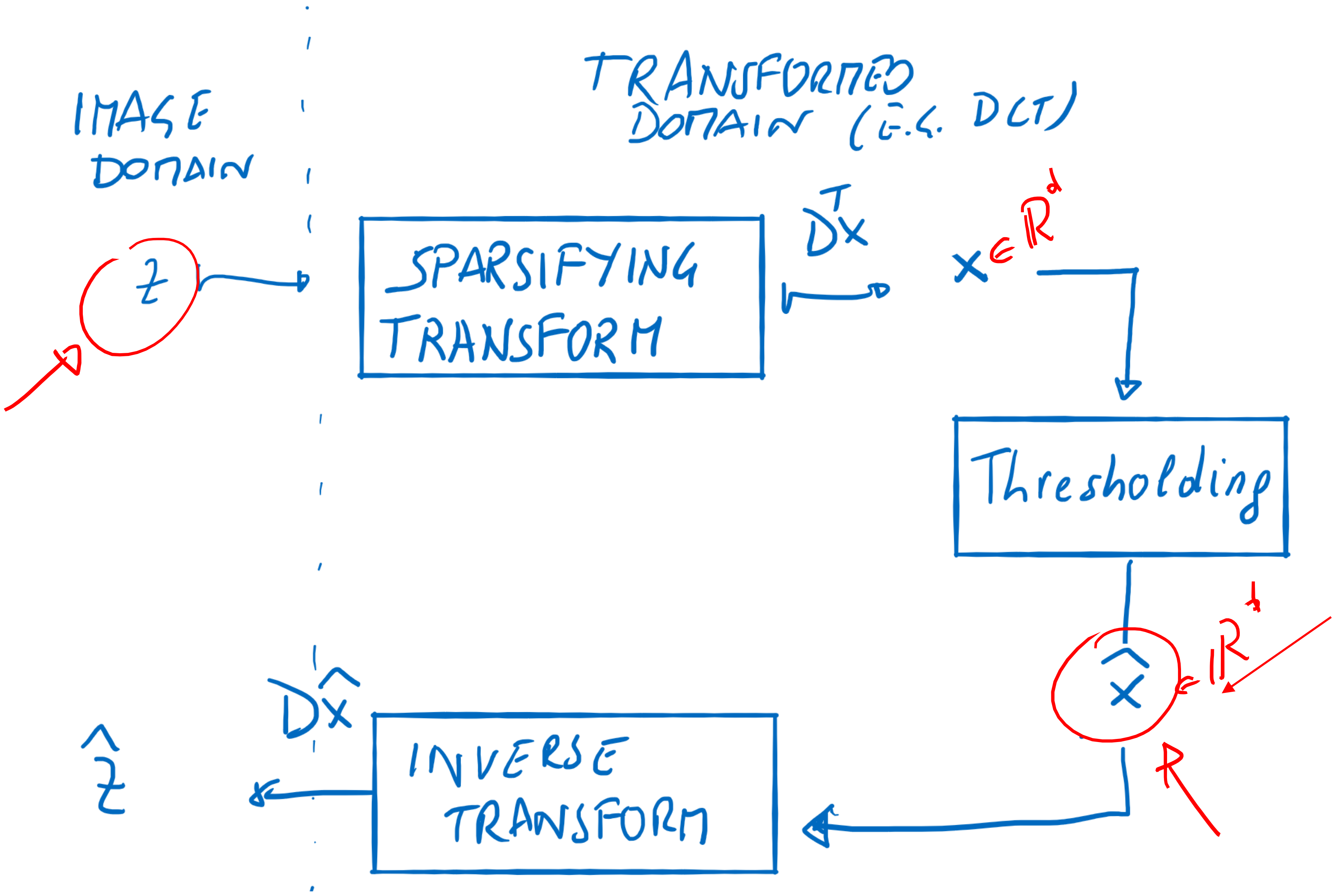


Sparsity Promoting Denoising



Giacomo Boracchi

Sparsity Promoting Denoising

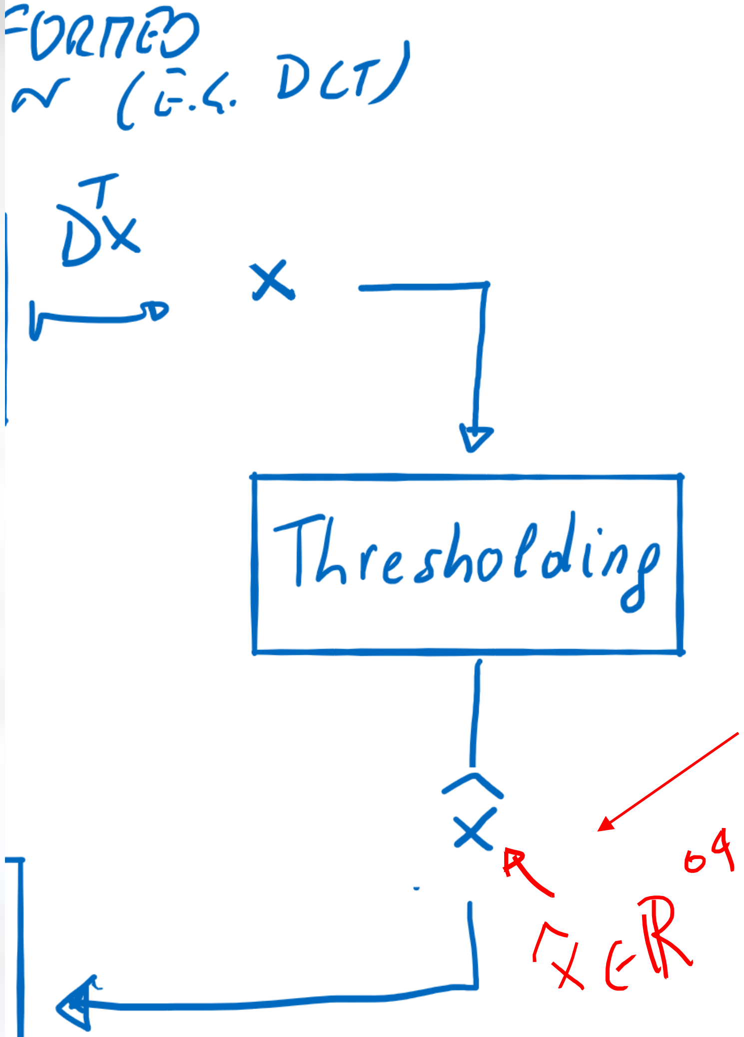


This is a much more compressed representation than z.

Encoding $\hat{x}$ **instead of** z can significantly reduce the size of the image

JPEG Compression performs encoding of $\hat{x}$

Sparsity Promoting Denoising



This is a much more compressed representation than z.

Encoding $\hat{x}$ **instead of** z can significantly reduce the size of the image

JPEG Compression performs encoding of $\hat{x}$

Recent Denoising Algorithms

Hand-Crafted Algorithms

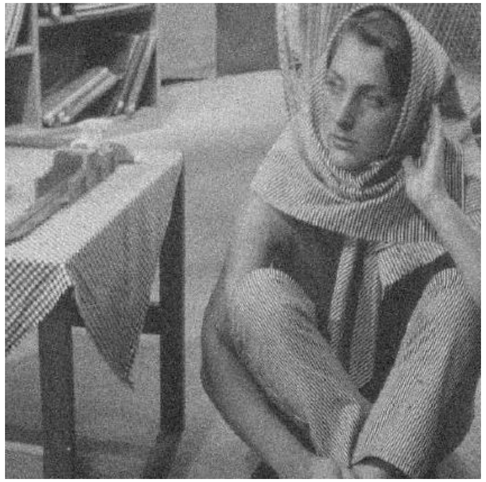
Recent denoisng algorithm combine principles from different approaches:

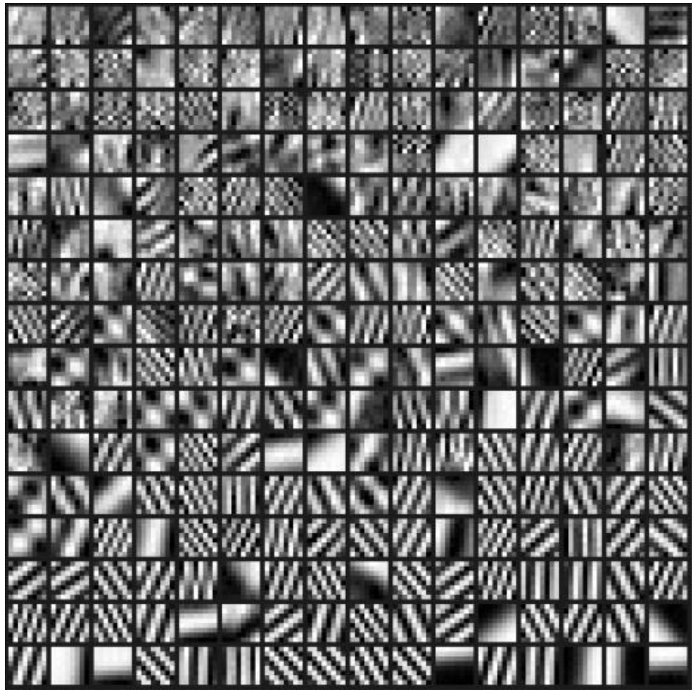
- Spatial Adaptivity
- Nonlocal self-similarity
- Sparsity w.r.t. to a basis / a learned set of generators
- …

K-SVD Image Denoising

original

Noisy image $(\sigma = 20)$





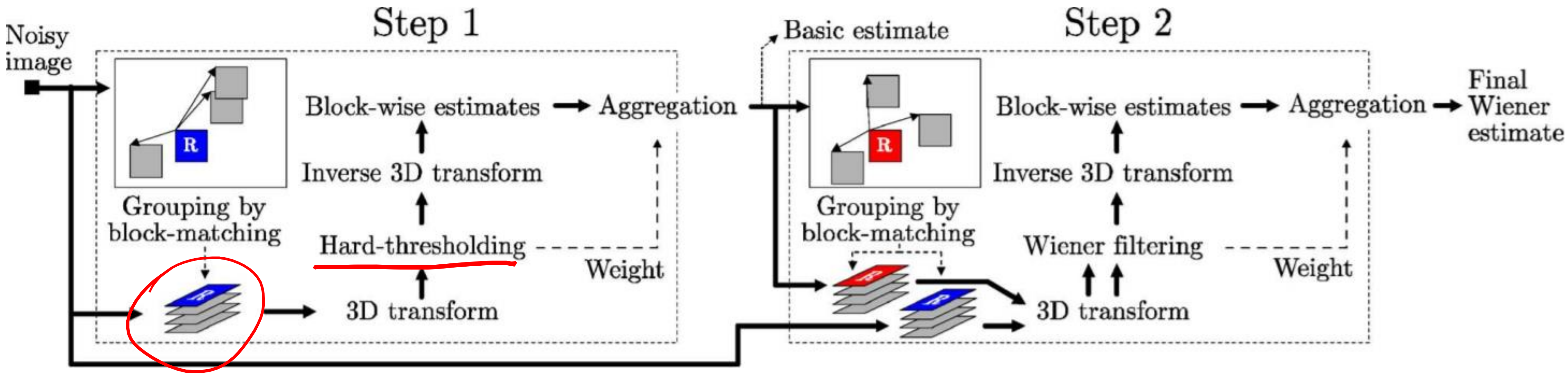
The obtained dictionary after 10 iterations

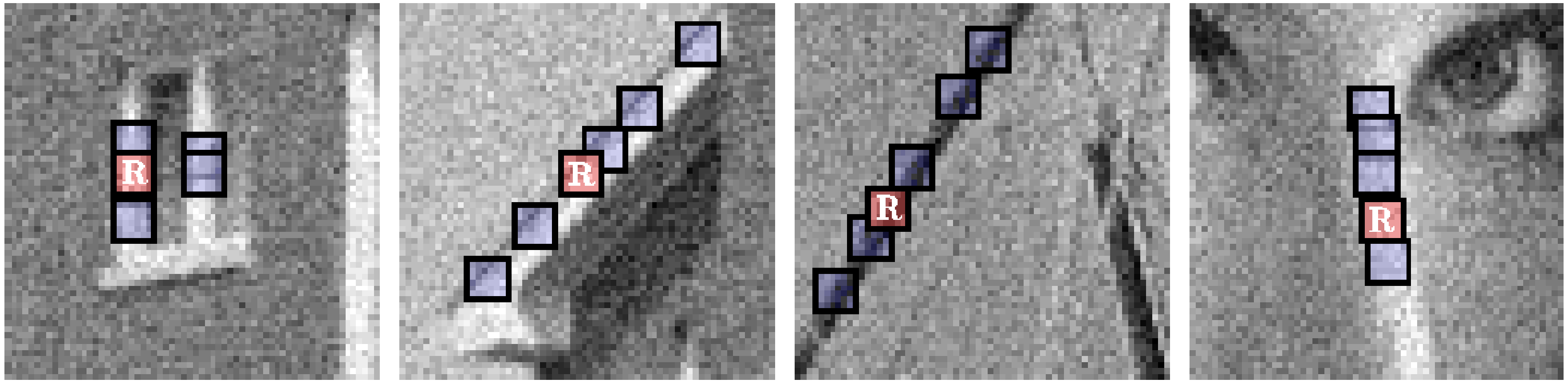
Result 30.829dB

Aharon, M., Elad, M., & Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on signal processing, 54(11), 4311-4322.

Sparse and Redundant Representation Modeling of Signals – Theory and Applications By: Michael Elad

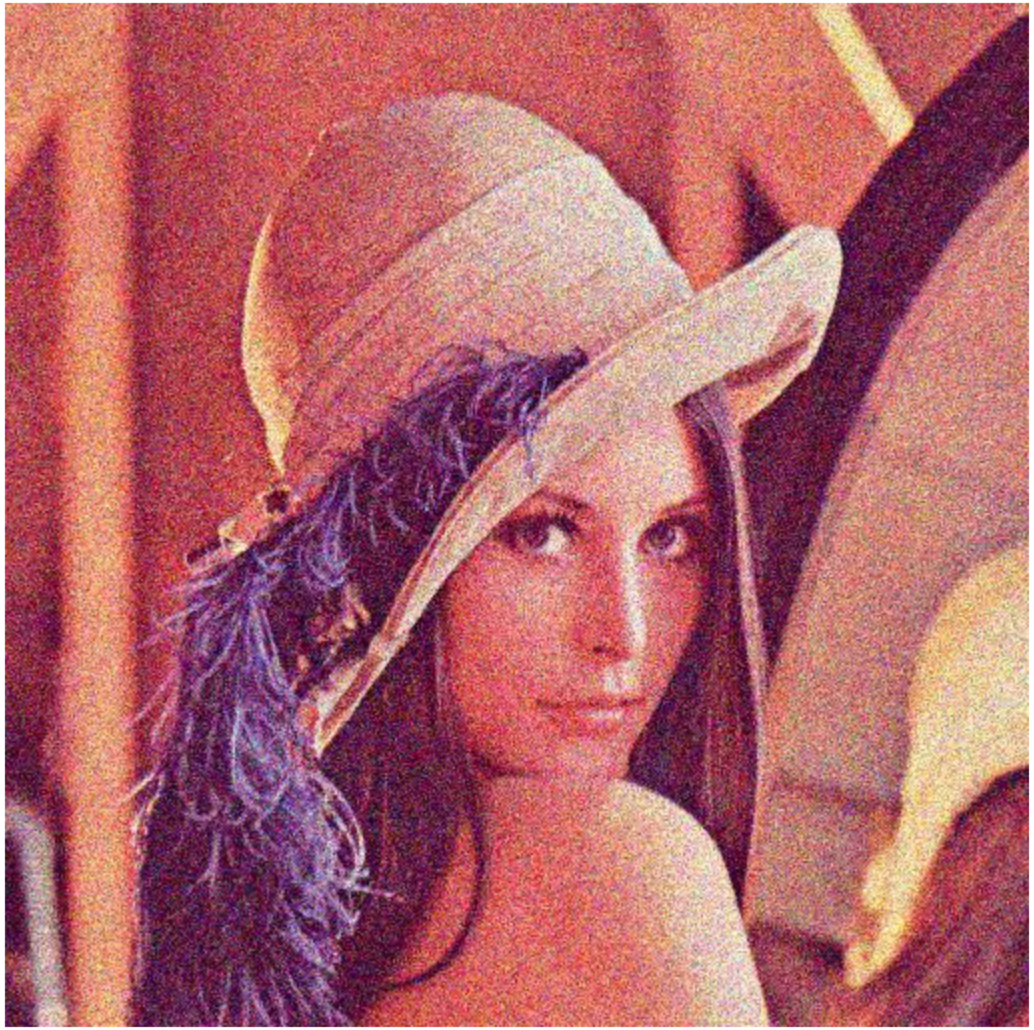
BM3D: Block Matching and 3D collaborative Filtering





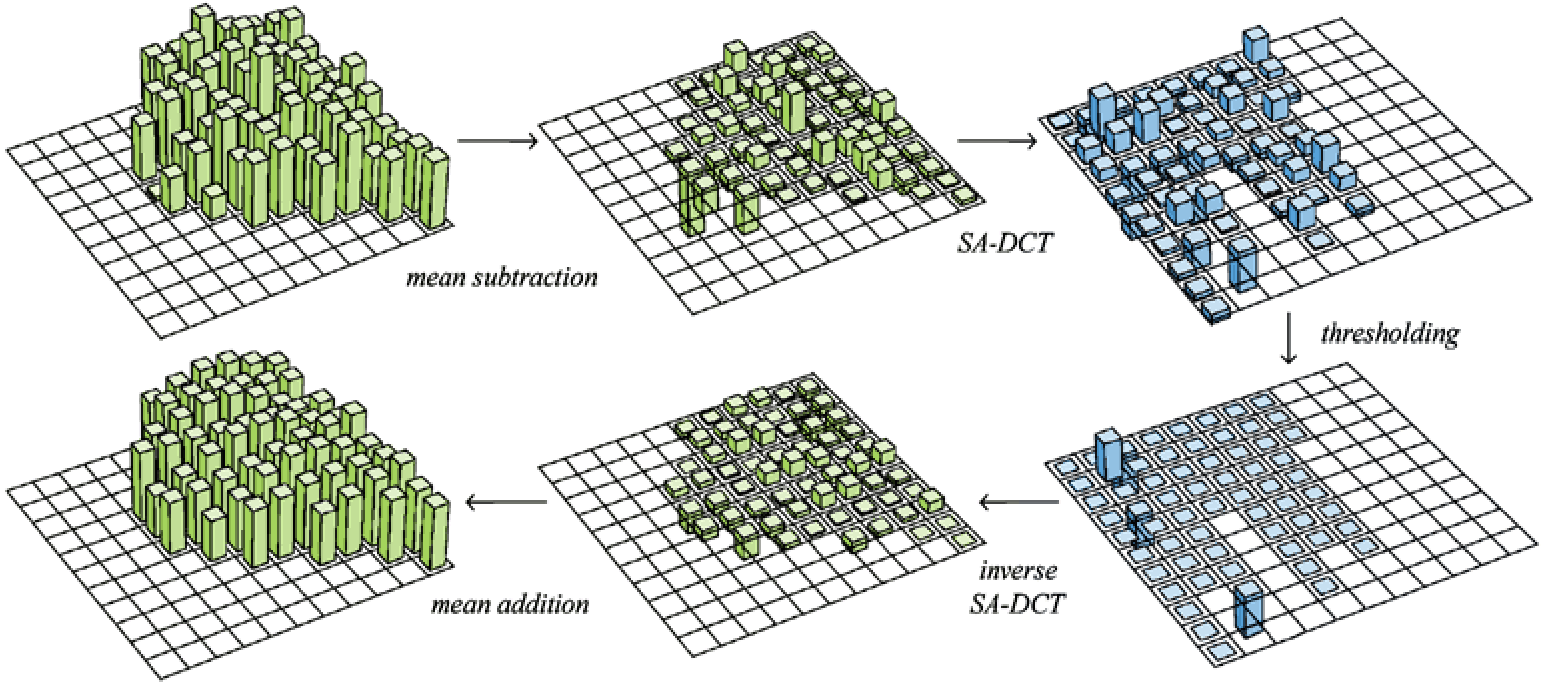
Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE TIP 2007*

BM3D denoising ($\sigma = 35$)



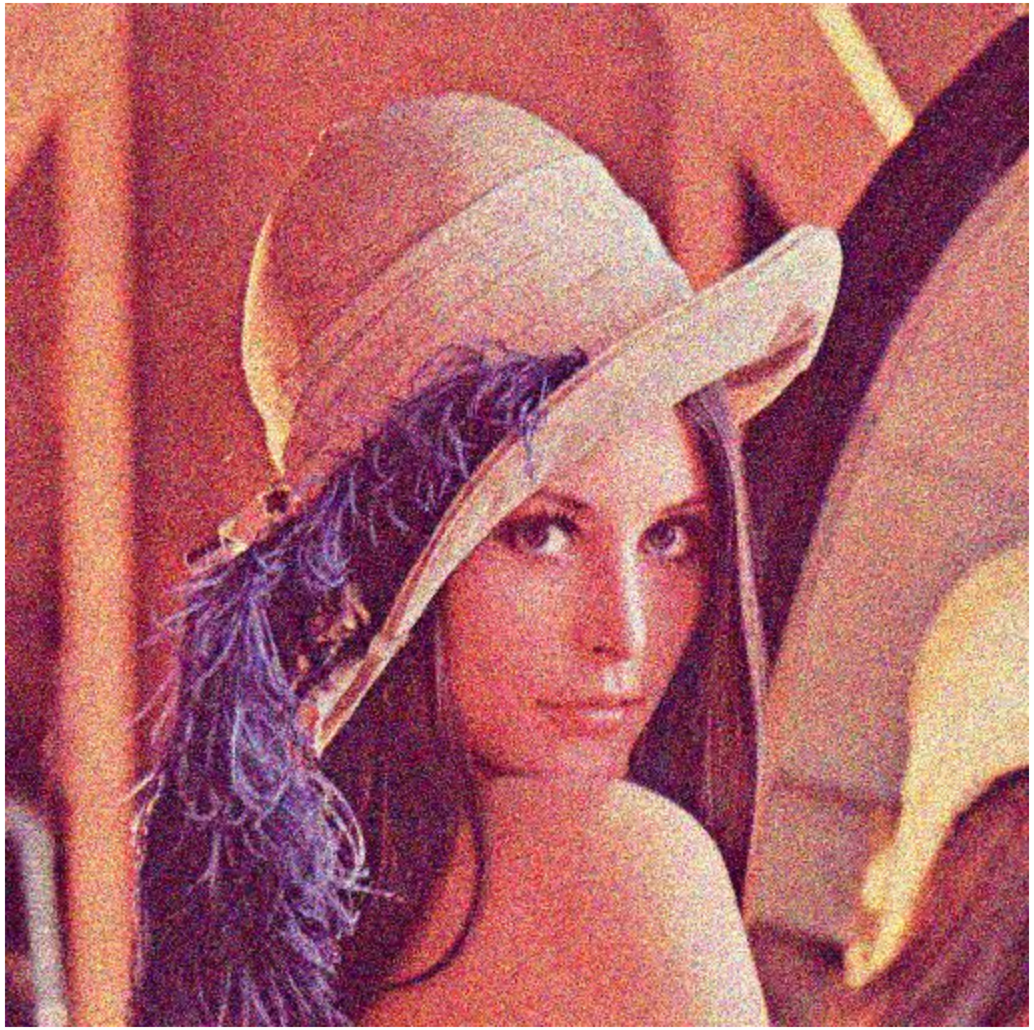
Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE TIP 2007*

Shape Adaptive DCT



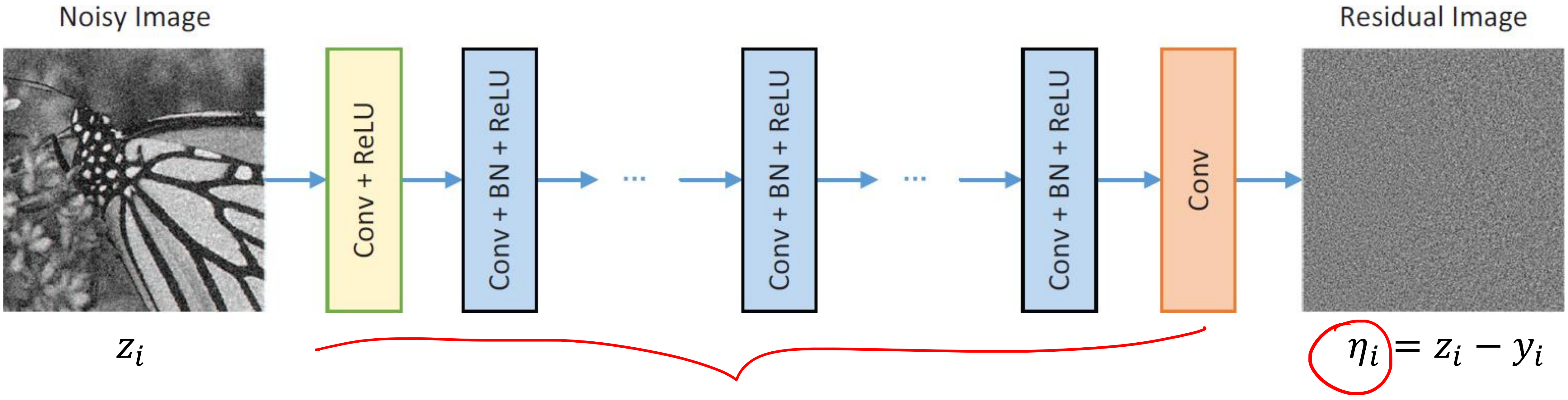
Foi, A., V. Katkovnik, and K. Egiazarian, “Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images”, IEEE TIP 2007

Shape Adaptive DCT ($\sigma = 35$)



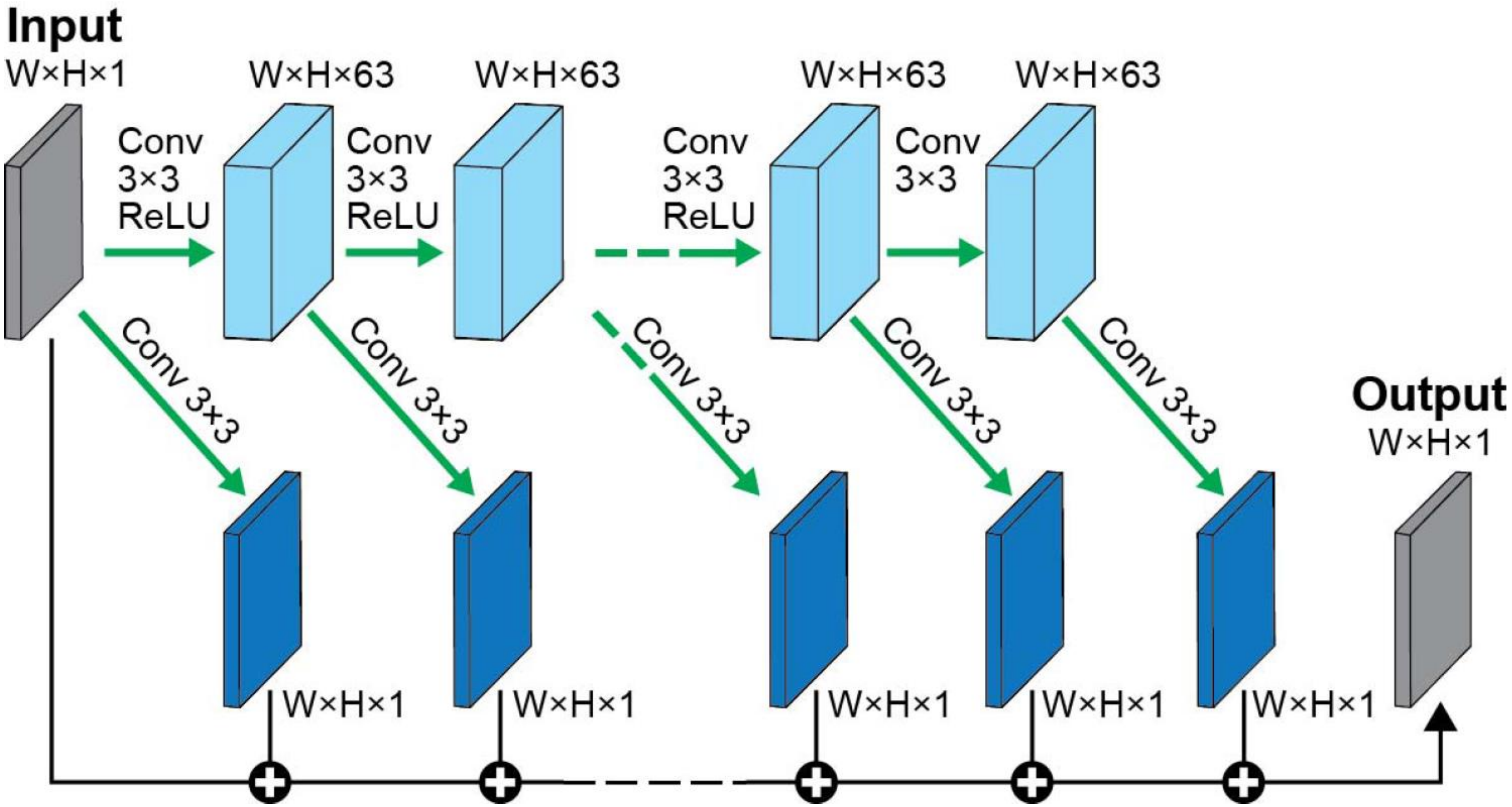
Deep Learning Methods

Dn-CNN



K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," IEEE Transactions on Image Processing, vol. 26, no. 7, pp. 3142–3155, July 2017.

The Network Architecture



Remez, T., Litany, O., Giryes, R., & Bronstein, A. M. (2018). Class-Aware Fully Convolutional Gaussian and Poisson Denoising. IEEE Transactions on Image Processing, 27(11), 5707-5722.

That’s all, folks!