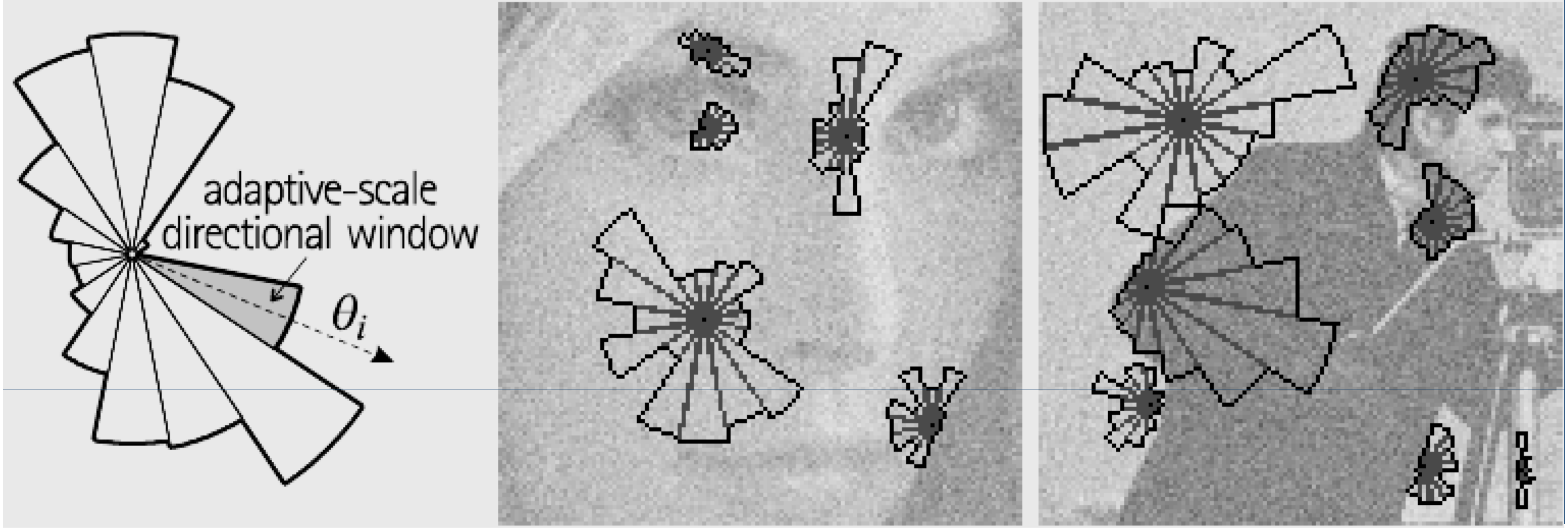


Giacomo Boracchi
Politecnico di Milano
giacomo.boracchi@polimi.it

Joint Research Center, ISPRA

Image Processing case study: LPA-ICI Denoising

27th October 2009

Disclaimer

- Most of the material shown in these slides is taken from *LASIP (Local Approximation in Signal and Image Processing)* laboratory webpage.

- http://www.cs.tut.fi/~lasip/

- There you can download papers and Matlab sources for most of implemented algorithms

Outline

- Image Formation Model
- Denoising approaches
- LPA-ICI Denoising Motivations
- Local Polynomial Approximation
- ICI rule
- Algorithm Details

Image Formation Model

- Observation model is

$$z(x) = y(x) + \eta(x)$$

z	Sensed image
x	Pixel index $x \in X$
y	Original (unknown) image
η	noise.

- For the sake of simplicity we will assume $\eta \sim N(0, \sigma^2)$
- The goal is to obtain $\widehat{y}(x)$, a reliable estimate of $y(x)$, given $z(x)$ and the distribution of η .

Denoising Approaches

- Parametric Approaches
  - Transform Domain Filtering, they assume the noisy-free signal is somehow sparse in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition)

Denoising Approaches

- Parametric Approaches
  - Transform Domain Filtering, they assume the noisy-free signal is somehow sparse in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition)

- Non Parametric Approaches
  - Local Smoothing / Local Approximation
  - Non Local Methods

Denoising Approaches

- Parametric Approaches
  - Transform Domain Filtering, they assume the noisy-free signal is somehow sparse in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition)

- Non Parametric Approaches
  - Local Smoothing / Local Approximation
  - Non Local Methods

Denoising Approaches

- Parametric Approaches
  - Transform Domain Filtering, they assume the noisy-free signal is somehow sparse in a suitable domain (e.g Fourier, DCT, Wavelet) or w.r.t. some dictionary based decomposition)

- Non Parametric Approaches
  - Local Smoothing / Local Approximation
  - Non Local Methods

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

$$\widehat{y}(x) = E[z|x]$$

Non parametric approaches

- **Local / Non Local**

  - *Local methods: weights used for the algorithm depends on the distance between the estimation point* x_0 *and the other observation points* $x_s, \; \|x_0 - x_s\|$

Non parametric approaches

- **Local / Non Local**

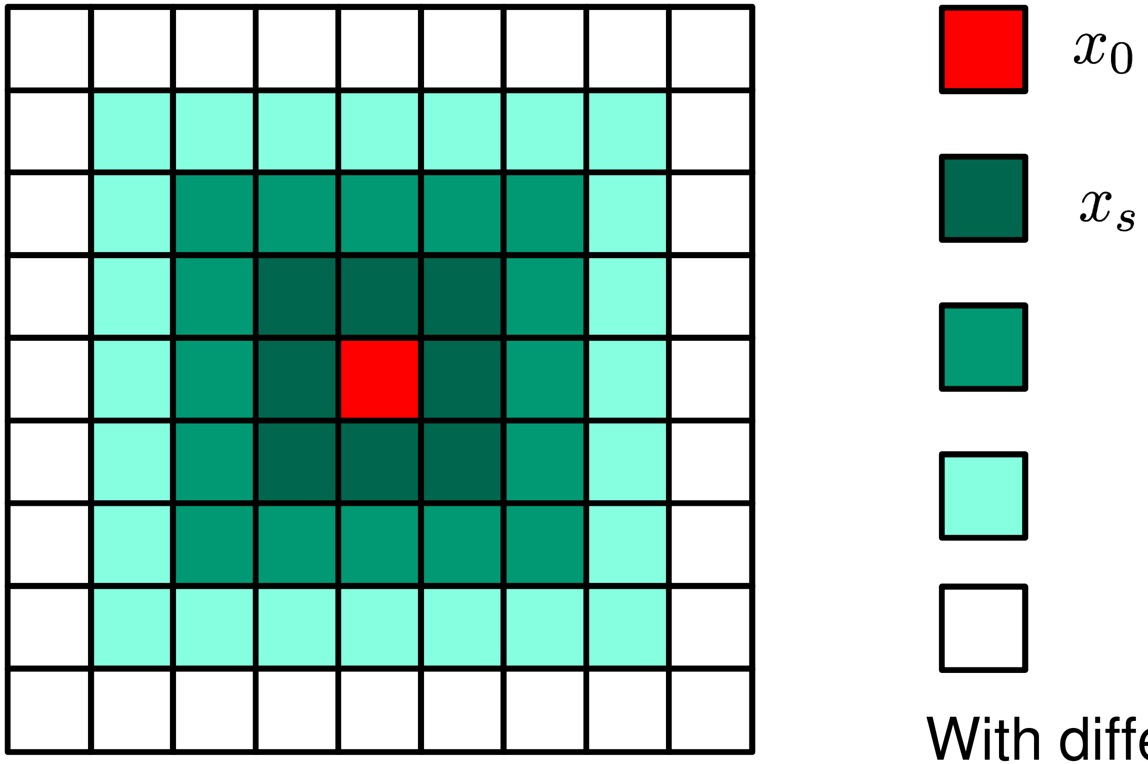
  - *Local methods: weights used for the algorithm depends on the distance between the estimation point* x_0 *and the other observation points* $x_s, \; \|x_0 - x_s\|$

  - *Non Local Methods: the weights are function of the differences of the corresponding signals. The weight used to estimate* y_0 *depends on* y_s *. Typically on something related to* $\|y_0 - y_s\|_2$ *.*

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

Local vs Non-Local

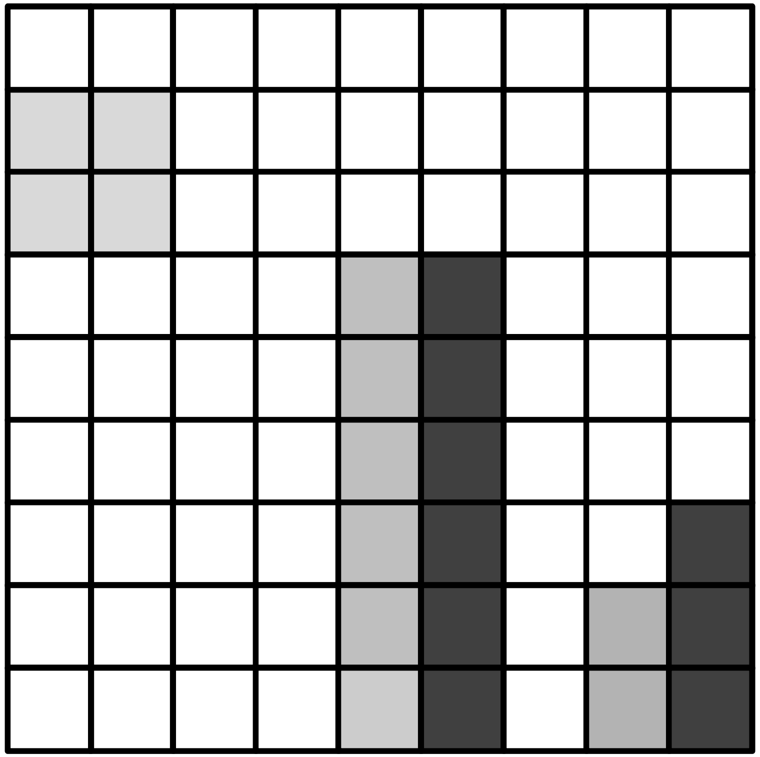
- Local, weights are determined by the pixel distance (regardless of the image content)



With different weights

Local vs Non-Local

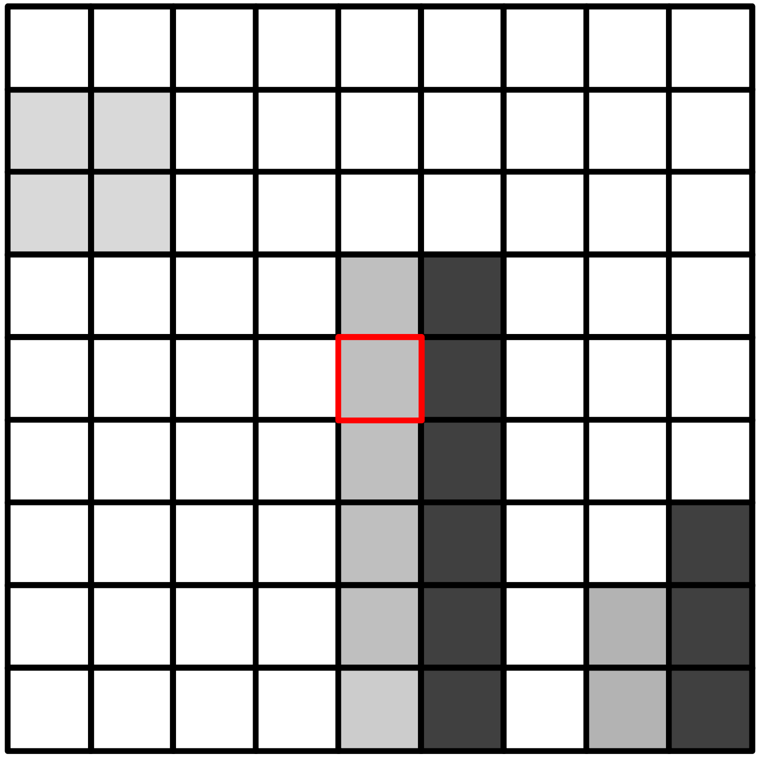
- Non Local, weights are determined by the image similarity



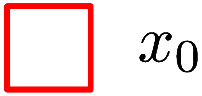
- Example of observation

Local vs Non-Local

- Non Local, weights are determined by the image similarity

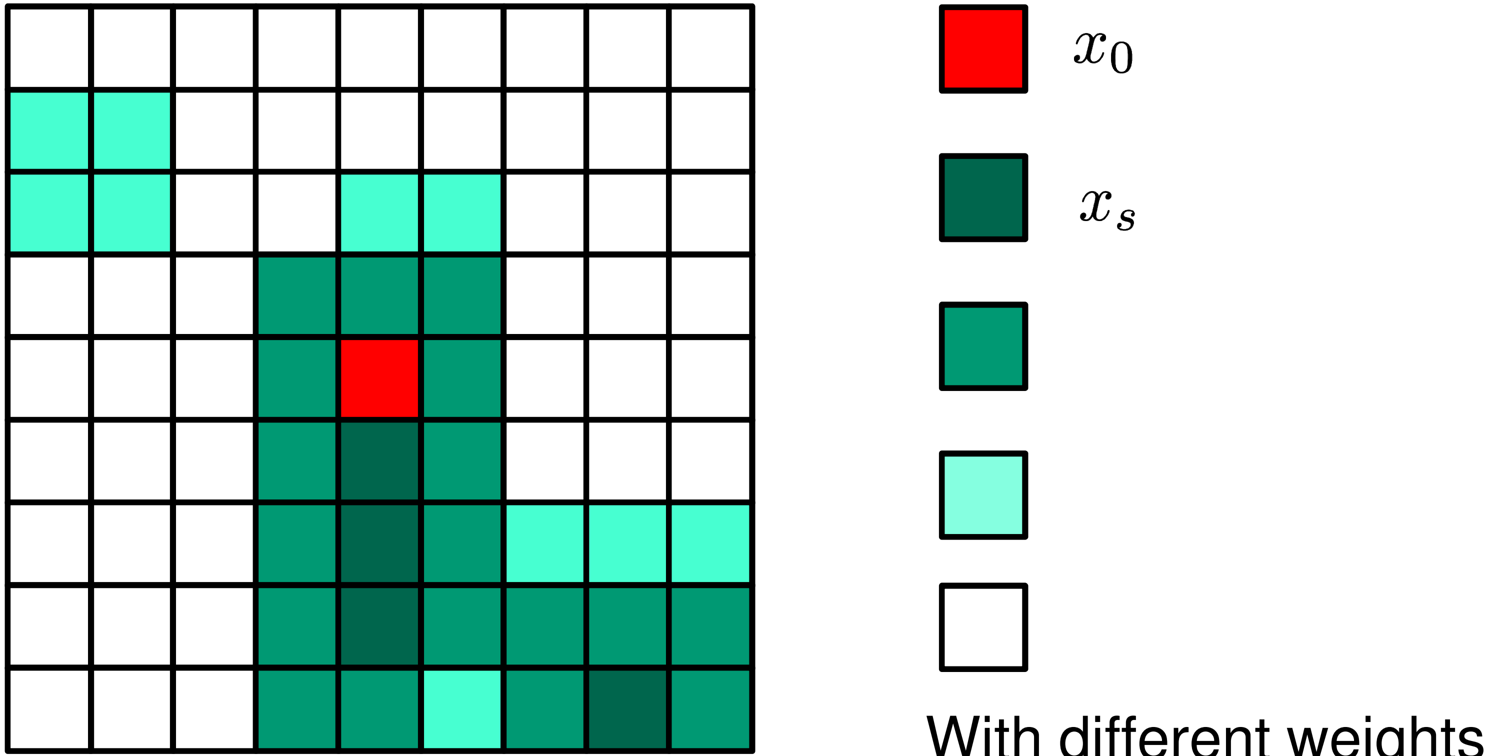


- Example of observation



Local vs Non-Local

- Non Local, weights are determined by the image similarity



Non parametric 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*

Non parametric approaches

- **Pointwise / Multipoint**

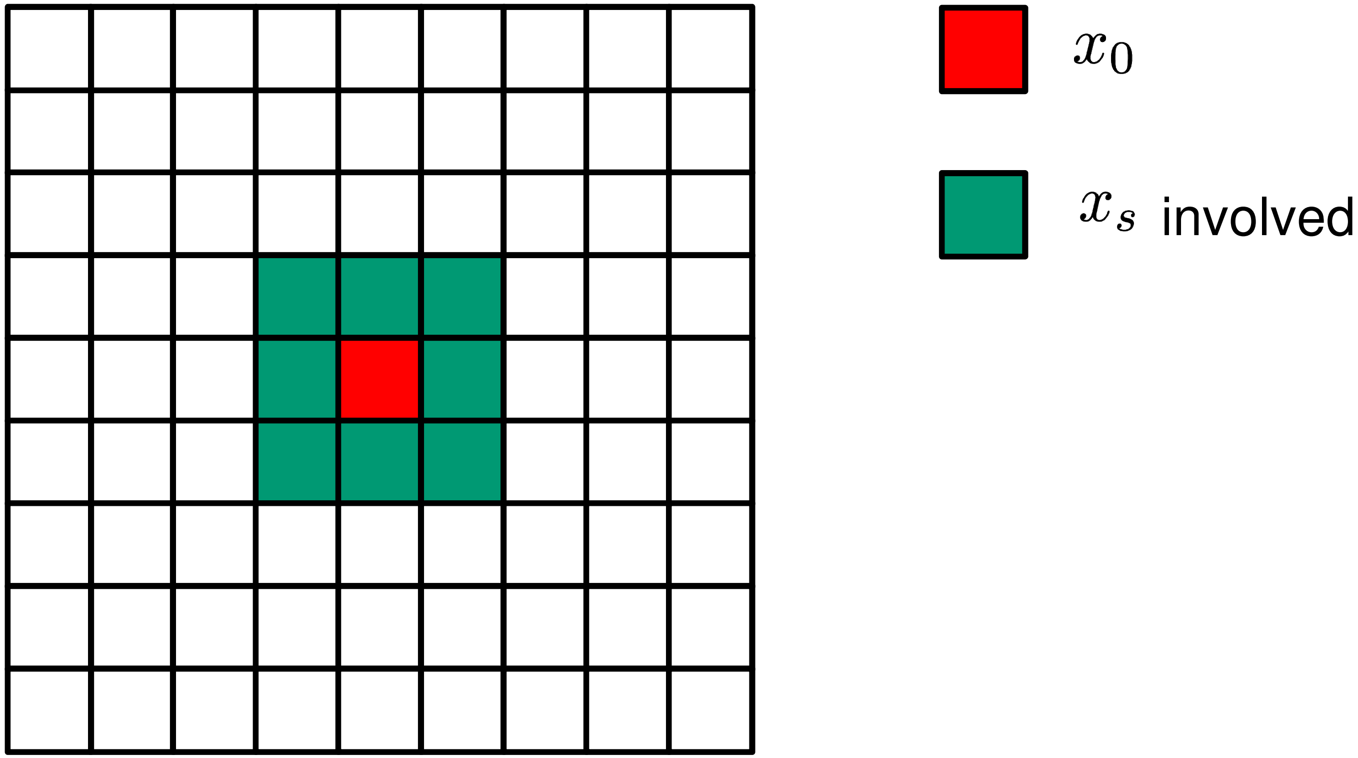
  - *Pointwise: the estimation of noise-free signal is computed for the central point only,* y_0 *and not for all the other points considered*

  - *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", preprint (July 2009), to appear Int. J. Computer Vision.

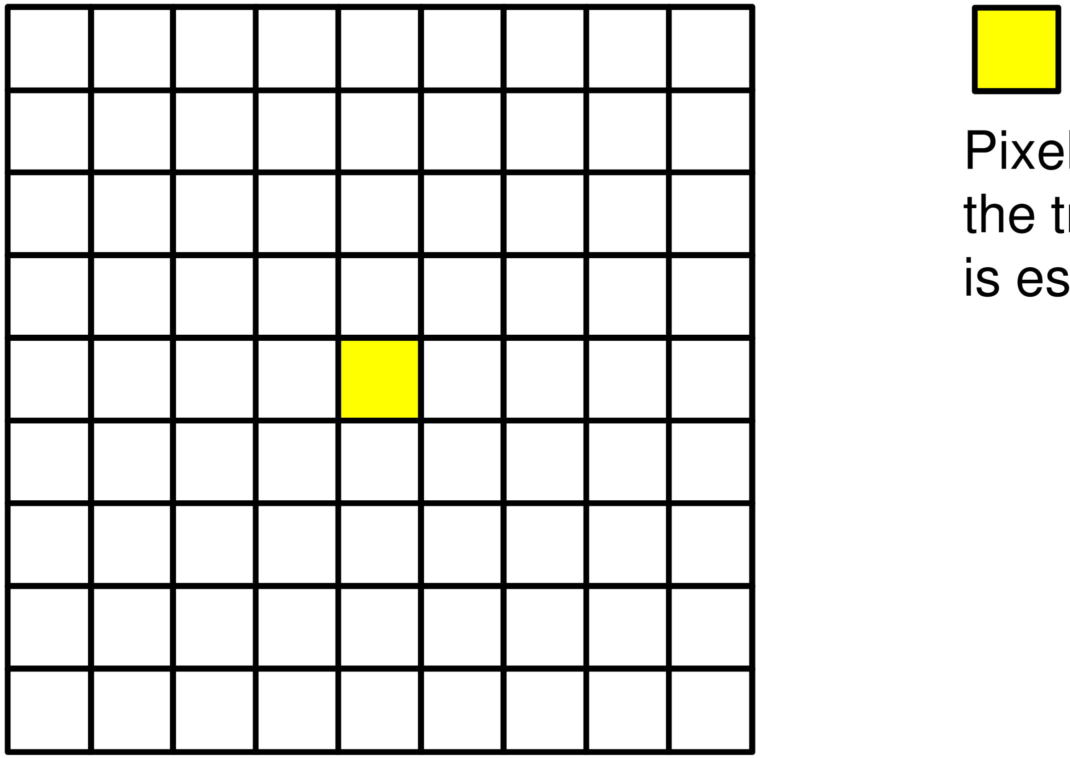
Pointwise vs Multipoint

- Pointwise, the estimate is given for the central point only



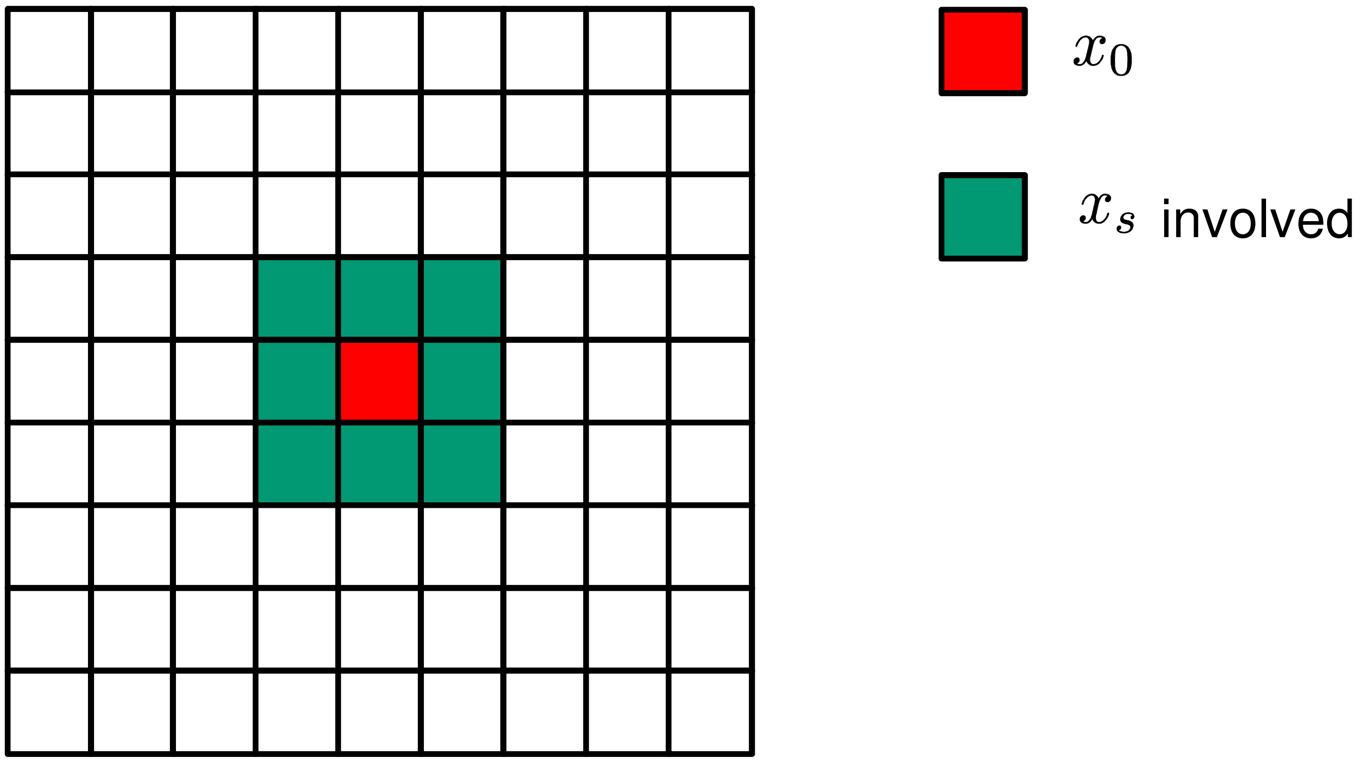
Pointwise vs Multipoint

- Pointwise, the estimate is given for the central point only



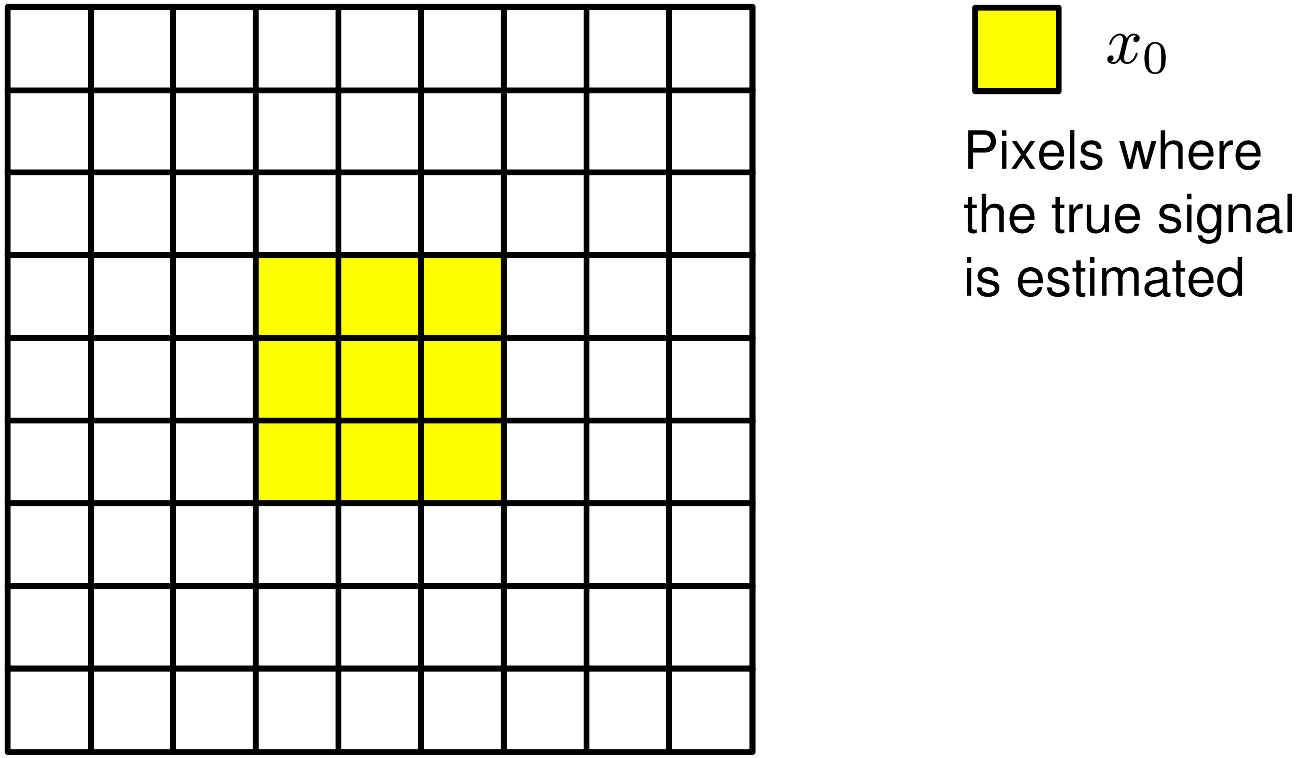
Pointwise vs Multipoint

- 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



Disclaimer

- This classification holds for the principles that determine the algorithm design.

- Most of the algorithm are implemented combining methods from different approaches.

- Furthermore, some of the most effective denoising algoritms enforce parametric assumptions on image areas adatptively selected
  - SA-DCT
  - BM3D

What are we going to see then,

- A Local, pixel-wise denoising algorithm

LPA-ICI : Local Polynomial Approximation using Intersection of Confidence Interval rule.

- We will consider the **denoising** as a **basic problem**, although this method has been successfully applyied to several other problems such as
  - Deblurring
  - Interpolation
  - Enhancement
  - Demosaicing
  - Deblocking
  - Inverse Halftoning

Why?

- Why to use such an algorithm as a case study for Cuda programming?

  - It is **Embarrassingly parallel** (it is a local pixelwise method).
  - It is simple to implement.
  - It is motivated by few clear and easy-to-show assumption.
  - It has been successfully applied to **several image processing challenges**.

LPA-ICI basic ideas

- It combines two independent ideas:

  - **Local Polynomial Approximation** (LPA) for designing linear filters that performs *pixelwise polynomial fit* on a certain neighborhood.

LPA-ICI basic ideas

- It combines two independent ideas:
  - **Local Polynomial Approximation** (LPA) for designing linear filters that performs *pixelwise polynomial fit* on a certain neighborhood.
  - **Intersection of Confidence Interval** rule (ICI) is an *adaptation algorithm,* used to define the most suited neighborhood where the polynomial assumpions fit better the observations.

Local Pointwise Techniques

- **Local Pointwise weighted averages**: the estimate at x_0 is

$$\widehat{y_h}(x_0) = \sum_{x_s \in X} w_h(x_0 - x_s) z(x_s)$$

$$w_h = \{w_h(x)\} \quad s.t. \quad \sum_{x \in X} w_h(x) = 1$$

Local Pointwise Techniques

- **Local Pointwise weighted averages**: the estimate at x_0 is

$$\widehat{y_h}(x_0) = \sum_{x_s \in X} w_h(x_0 - x_s) z(x_s)$$

$$w_h = \{w_h(x)\} \quad s.t. \quad \sum_{x \in X} w_h(x) = 1$$

- Can be interpreted as the 0-th order polynomial that performs **least square fit**

$$\widehat{y_h}(x_0) = \mathrm{argmin}_C \sum_{x_s \in X} w_h(x_0 - x_s) \left(z(x_s) - C\ \right)^2$$

Local Pointwise Techniques

- **Local Pointwise weighted averages**: the estimate at x_0 is

$$\widehat{y_h}(x_0) = \sum_{x_s \in X} w_h(x_0 - x_s) z(x_s)$$

$$w_h = \{w_h(x)\} \quad s.t. \quad \sum_{x \in X} w_h(x) = 1$$

- Can be interpreted as the 0-th order polynomial that performs **least square fit**

$$\widehat{y_h}(x_0) = \mathrm{argmin}_C \sum_{x_s \in X} w_h(x_0 - x_s) \left(z(x_s) - C\ \right)^2$$

- The weights in the MSE are determined by the averaging window w_h and the **parameter** h **scales the window w.r.t. a basic window** w

$$w_h(x) = w(x/h)$$

Local Pointwise Techniques

- **Local Pointwise weighted averages**: the estimate at x_0 is

$$\widehat{y_h}(x_0) = \sum_{x_s \in X} w_h(x_0 - x_s) z(x_s)$$

$$w_h = \{w_h(x)\} \quad s.t. \quad \sum_{x \in X} w_h(x) = 1$$

- Can be interpreted as the 0-th order polynomial that performs least square fit

$$\widehat{y_h}(x_0) = \mathrm{argmin}_C \sum_{x_s \in X} w_h(x_0 - x_s) \left(z(x_s) - C \ \right)^2$$

- The weights in the MSE are determined by the averaging window w_h and the parameter h scales the window w.r.t. a basic window w

$$w_h(x) = w(x/h)$$

Local Pointwise Techniques

- Local Polynomial Approximation
  - **Determine** $p_{h,m}$ the **polynomial expression** (of a fixed order *m*) that better fits the observation on a fixed pixel neighborhood w_h

$$p_{h,m} = \text{argmin}_{p \in \mathcal{P}_m} \sum_{x_s \in X} w_h(x_0 - x_s) \left(z(x_s) - p(x_s) \right)^2$$

Local Pointwise Techniques

- Local Polynomial Approximation
  - **Determine** $p_{h,m}$ the **polynomial expression** (of a fixed order *m*) that better fits the observation on a fixed pixel neighborhood w_h

$$p_{h,m} = \text{argmin}_{p \in \mathcal{P}_m} \sum_{x_s \in X} w_h(x_0 - x_s) \left(z(x_s) - p(x_s)\ \right)^2$$

  - The **signal estimate** is given by $p_{h,m}(x_0)$, the **value** of this **polynomial** in x_0

$$\widehat{y_h}(x_0) = p_{h,m}(x_0)$$

the **weights** w_h determines the **localization of this fit**

LPA Kernel

- The LPA estimate can be obtained via a **convolution** with **discrete LPA kernels** g_h

$$\widehat{y_h}(x_0) = (z \circledast g_h)(x_0)$$

LPA Kernel

- The LPA estimate can be obtained via a **convolution** with **discrete LPA kernels** g_h

$$\widehat{y_h}(x_0) = (z \circledast g_h)(x_0)$$

LPA Kernel

- The LPA estimate can be obtained via a **convolution** with **discrete LPA kernels** g_h

$$\widehat{y_h}(x_0) = (z \circledast g_h)(x_0)$$

- Summarizing, LPA estimates can be obtained via a convolution with **discrete kernels** which are determined by:
  - The **order** of polynomial fit.
  - The **support** of the polynomial fit.
  - The **weight** of the minimization of the polynomial least square fit.

LPA Kernel

- The LPA estimate can be obtained via a **convolution** with **discrete LPA kernels** g_h

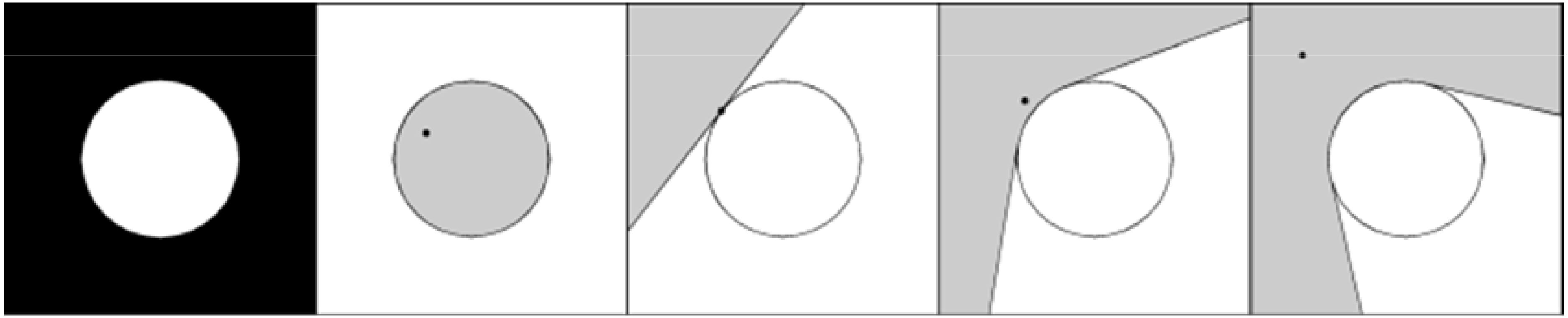
$$\widehat{y_h}(x_0) = (z \circledast g_h)(x_0)$$

- Summarizing, LPA estimates can be obtained via a convolution with **discrete kernels** which are determined by:
  - The **order** of polynomial fit.
  - The **support** of the polynomial fit.
  - The **weight** of the minimization of the polynomial least square fit.

- This makes LPA a perfect tool for **designing adaptive filters**.

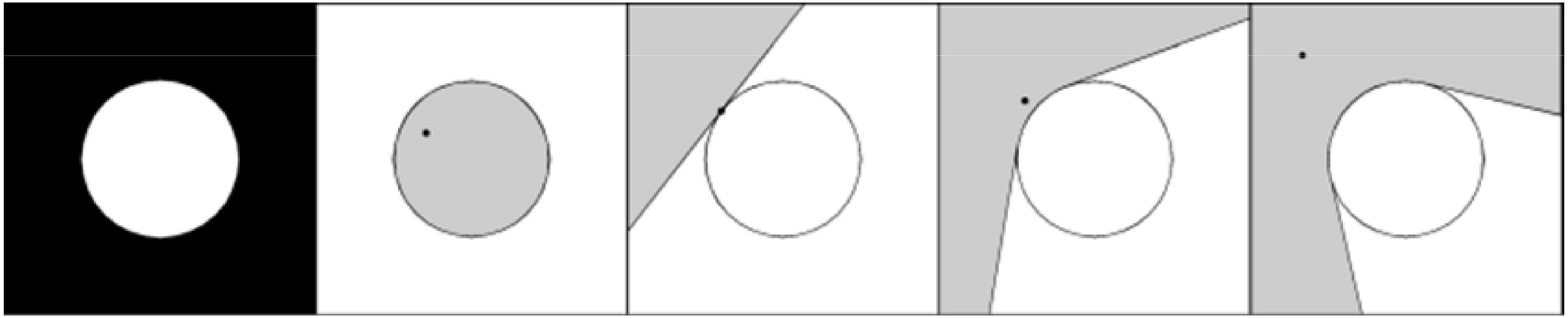
LPA-ICI algorithm: ideal neighborhood

- ***Ideal*** in the sense that it defines the support of **pointwise Least Square kernel estimators**.



LPA-ICI algorithm: ideal neighborhood

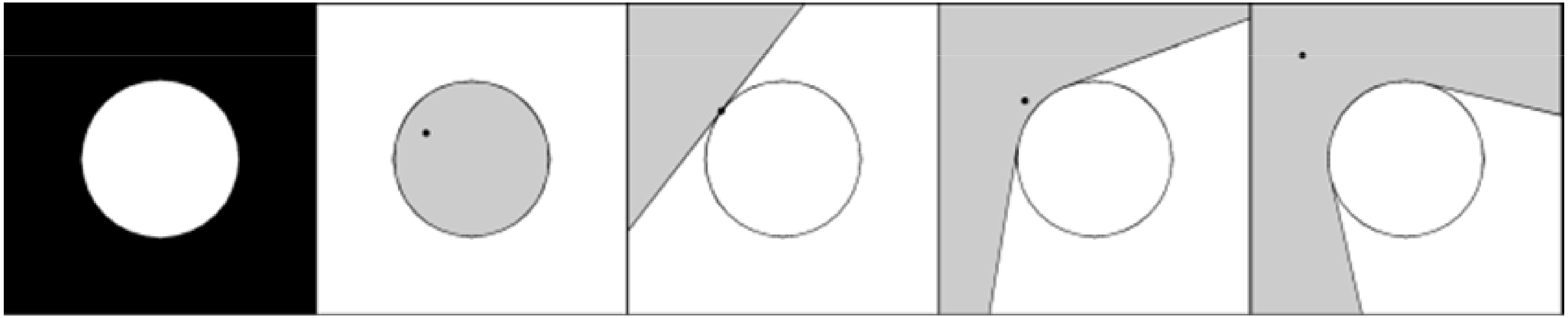
- *Ideal* in the sense that it defines the support of **pointwise Least Square kernel estimators**.



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

LPA-ICI algorithm: ideal neighborhood

- ***Ideal*** in the sense that it defines the support of **pointwise Least Square kernel estimators**.

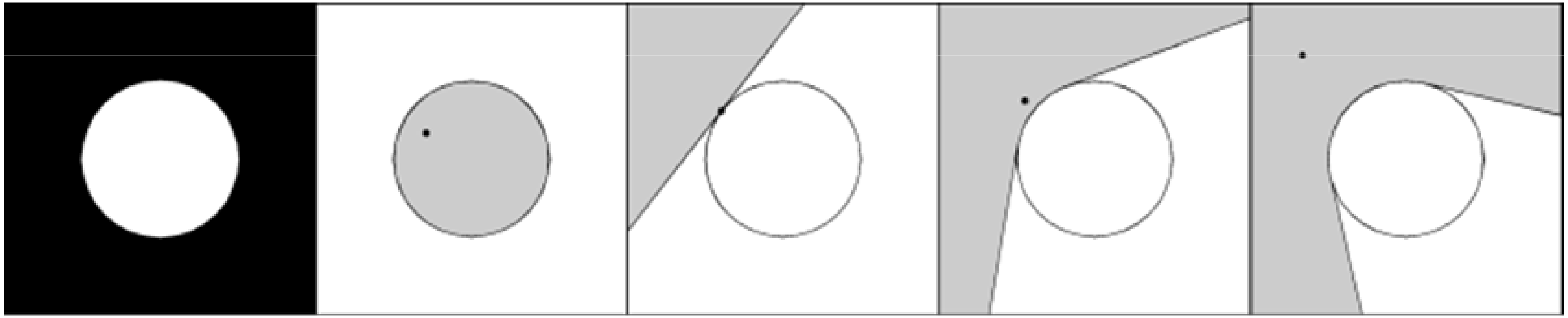


- Typically, even in simple images, **every point has its own different ideal neighborhood**.

- For practical reasons, the ideal **neighborhood** is assumed ***starshaped***

LPA-ICI algorithm: ideal neighborhood

- ***Ideal*** in the sense that it defines the support of **pointwise Least Square kernel estimators**.

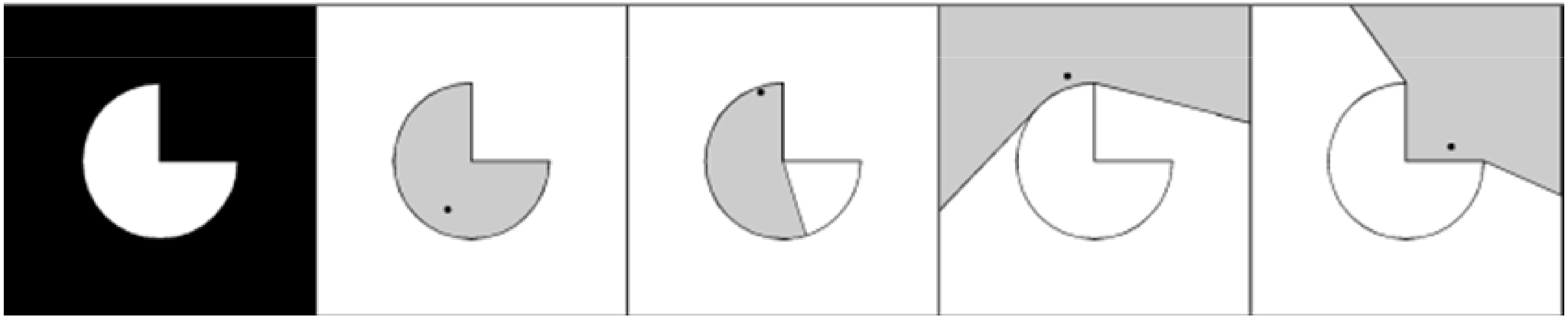


- Typically, even in simple images, **every point has its own different ideal neighborhood**.

- For practical reasons, the ideal **neighborhood** is assumed ***starshaped***

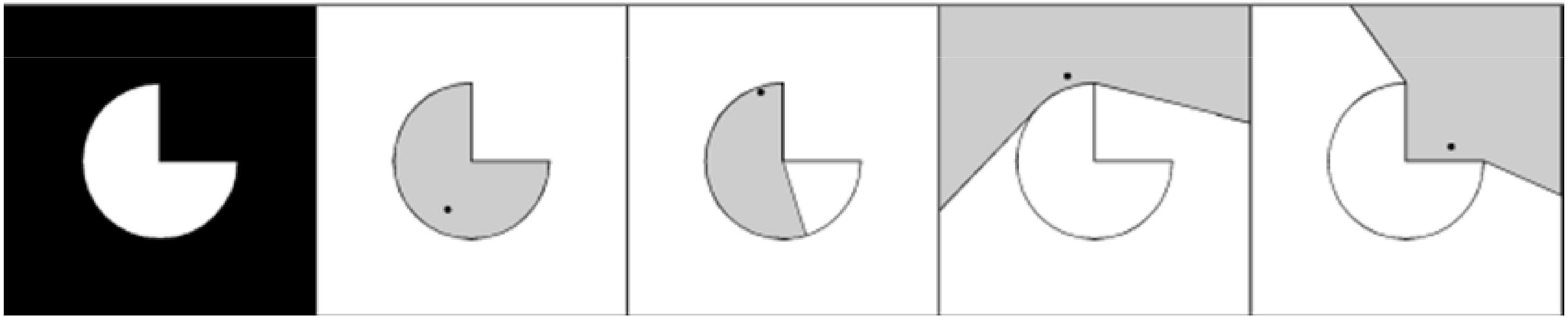
LPA-ICI algorithm: ideal neighborhood

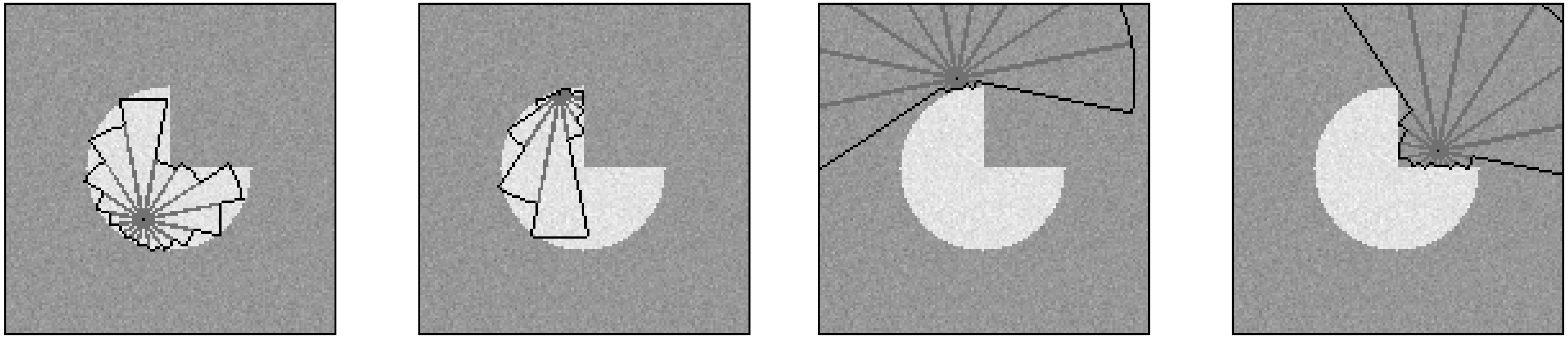
- The ideal neighborhood is built in the discrete image domain using LPA filters having directional supports



LPA-ICI algorithm: ideal neighborhood

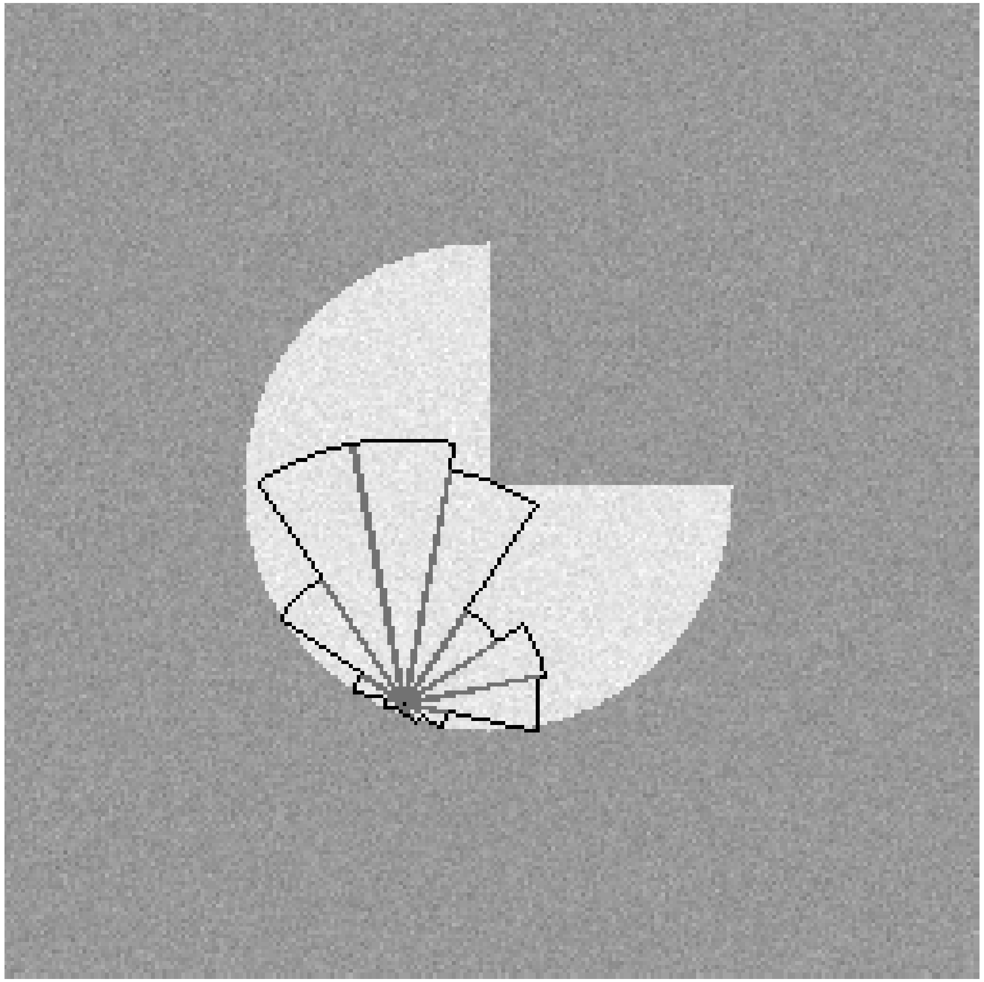
- The ideal neighborhood is built in the discrete image domain using LPA filters having directional supports





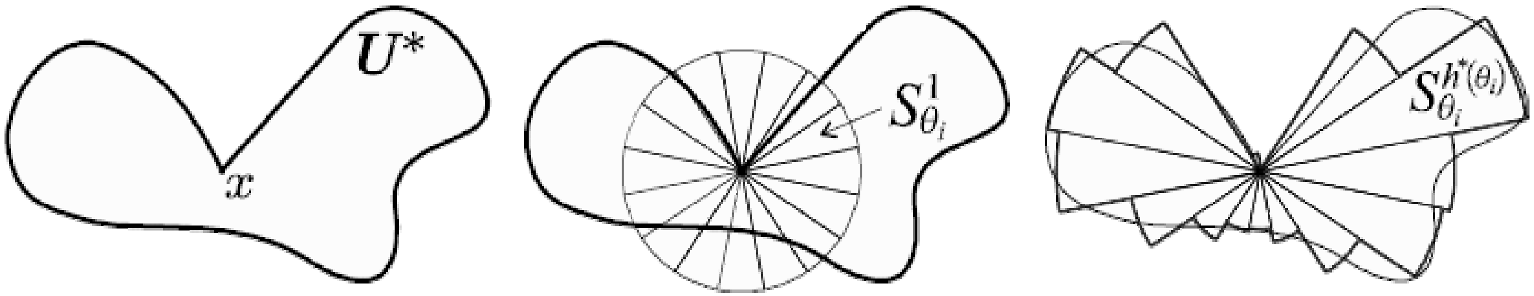
Examples of adaptively selected neighborhoods

- Adaptively selected neighborhoods selected using the LPA-ICI rule



Neighborhood discretization

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



Ideal Neighborhood

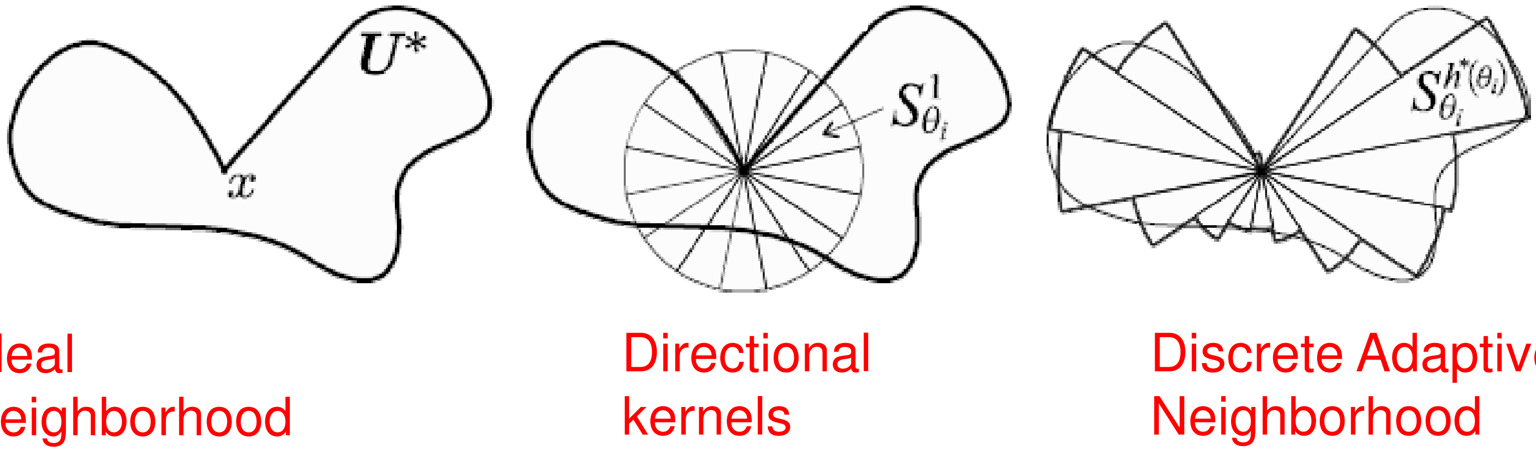
Directional kernels

Discrete Adaptive Neighborhood

where θ determines the orientation of the kernel support, and where h controls by the scale of kernel support.

Neighborhood discretization

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

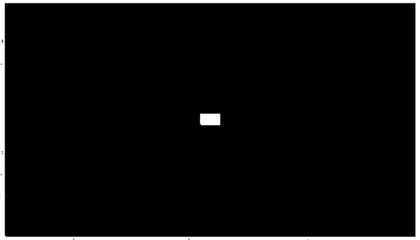


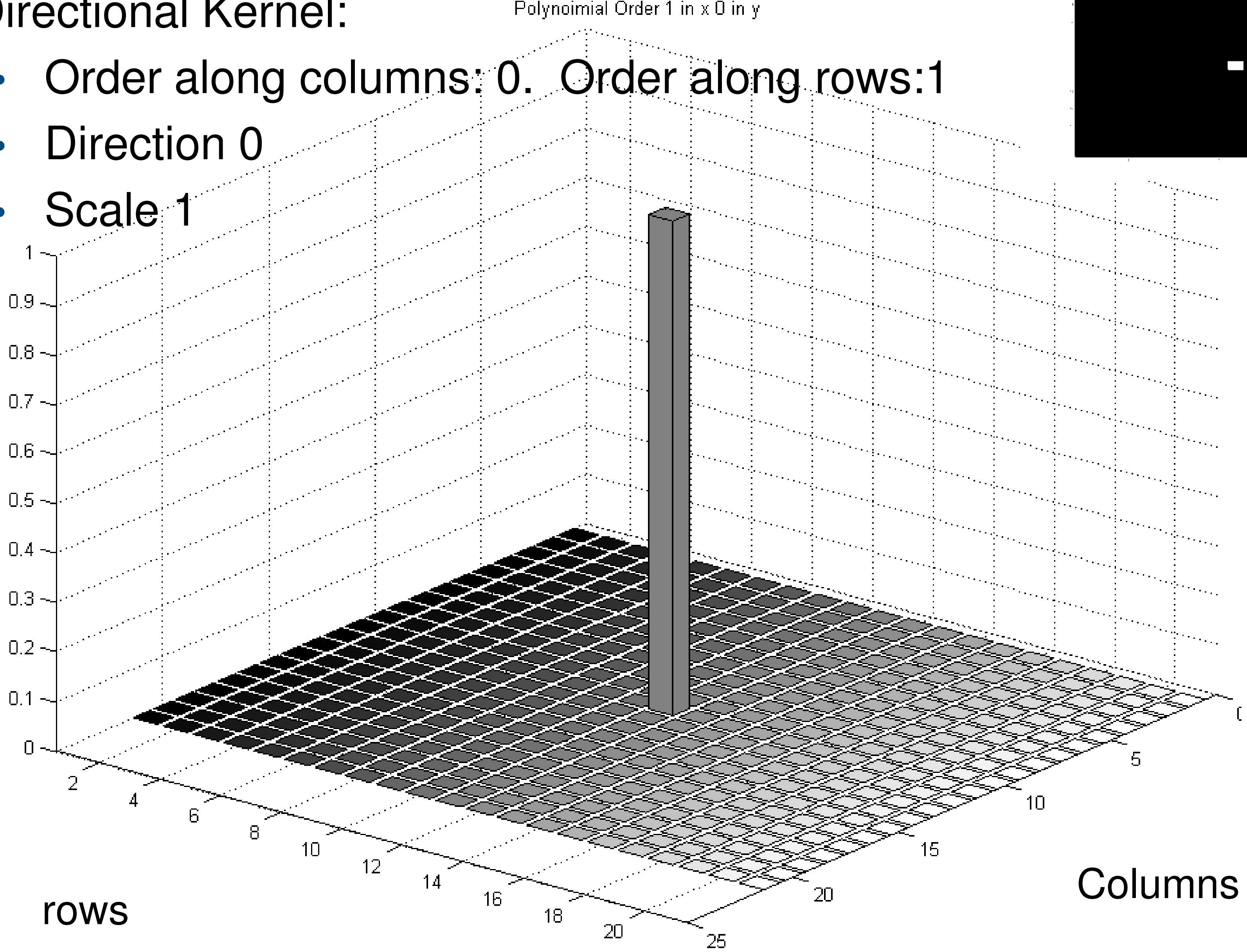
where θ determines the orientation of the kernel support, and where h controls by the scale of kernel support.

- The initial **shape optimization problem** can be solved by using standard easy-to-implement **varying-scale kernel techniques**, such as **the *ICI* rule**.

Example of LPA Anisotropic Kernels

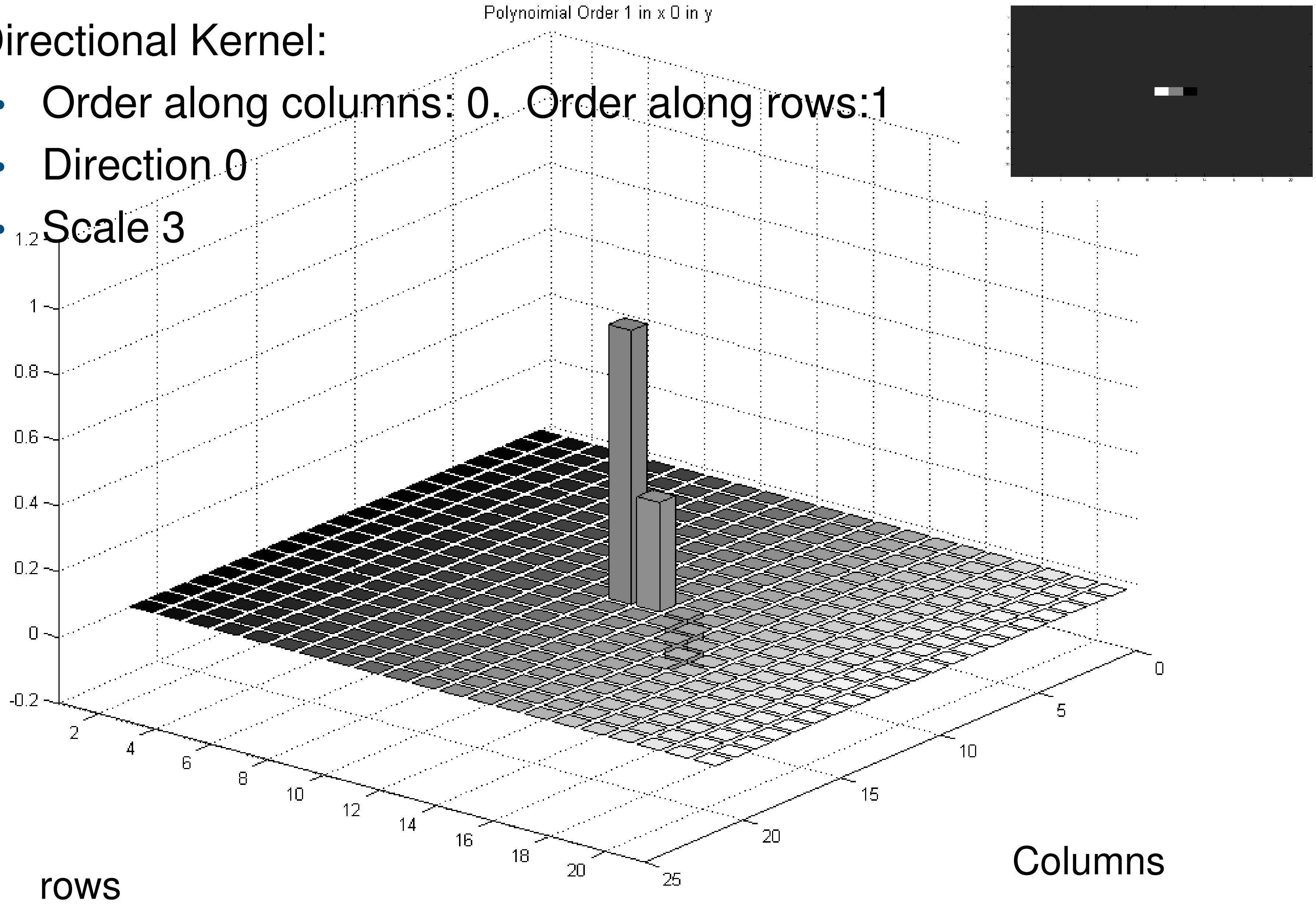
- Directional Kernel:
  - Order along columns: 0. Order along rows:1
  - Direction 0
  - Scale 1





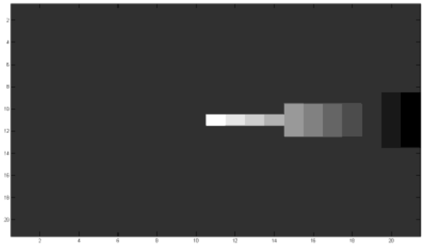
Example of LPA Anisotropic Kernels

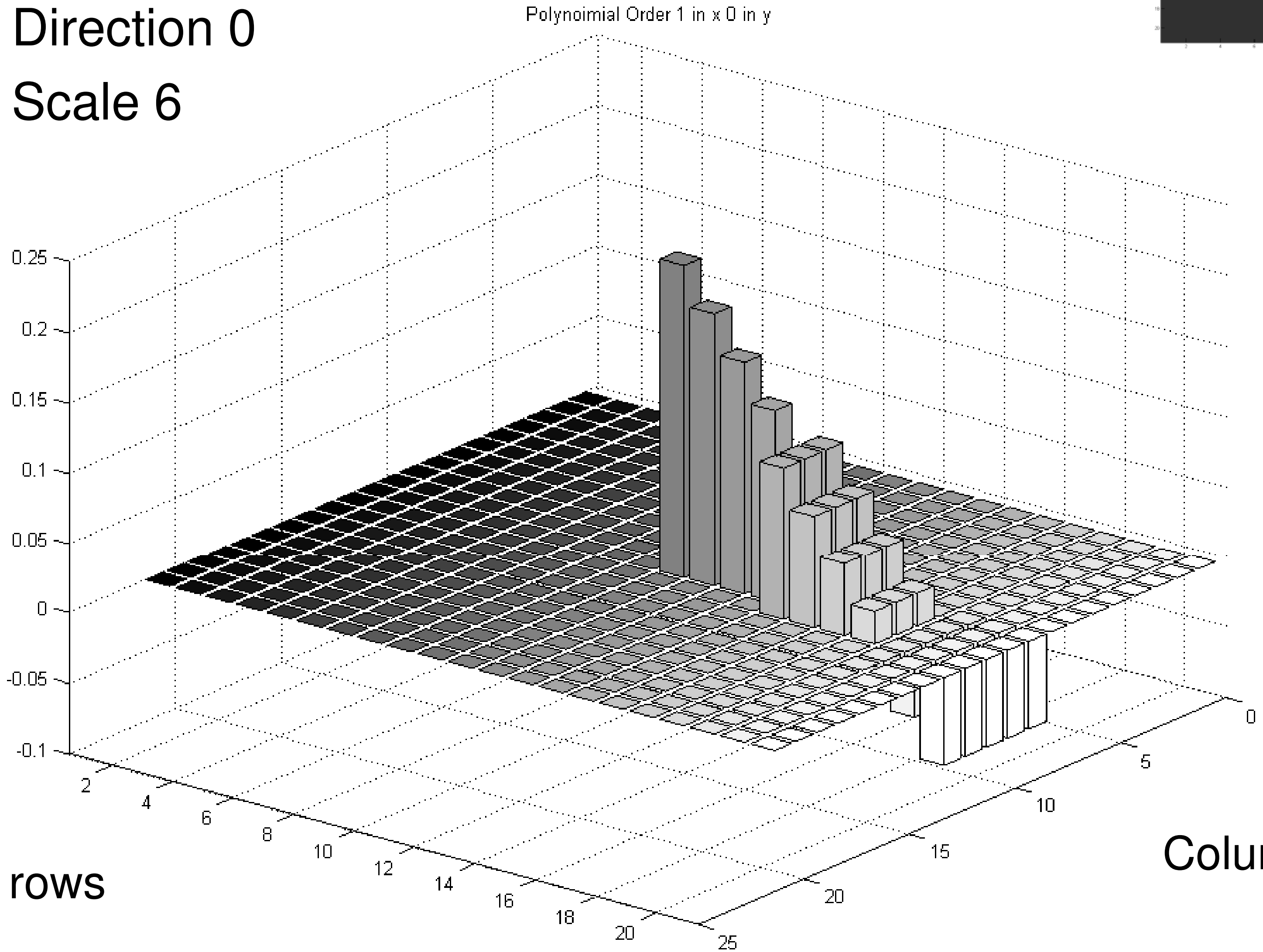
- Directional Kernel:
  - Order along columns: 0. Order along rows:1
  - Direction 0
  - Scale 3



Example of LPA Anisotropic Kernels

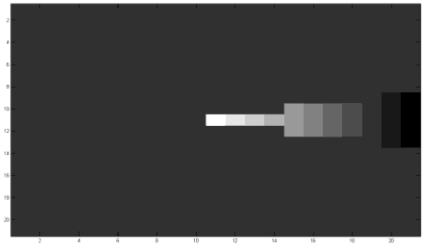
- Directional Kernel:
  - Order along columns: 0. Order along rows:1
  - Direction 0
  - Scale 6

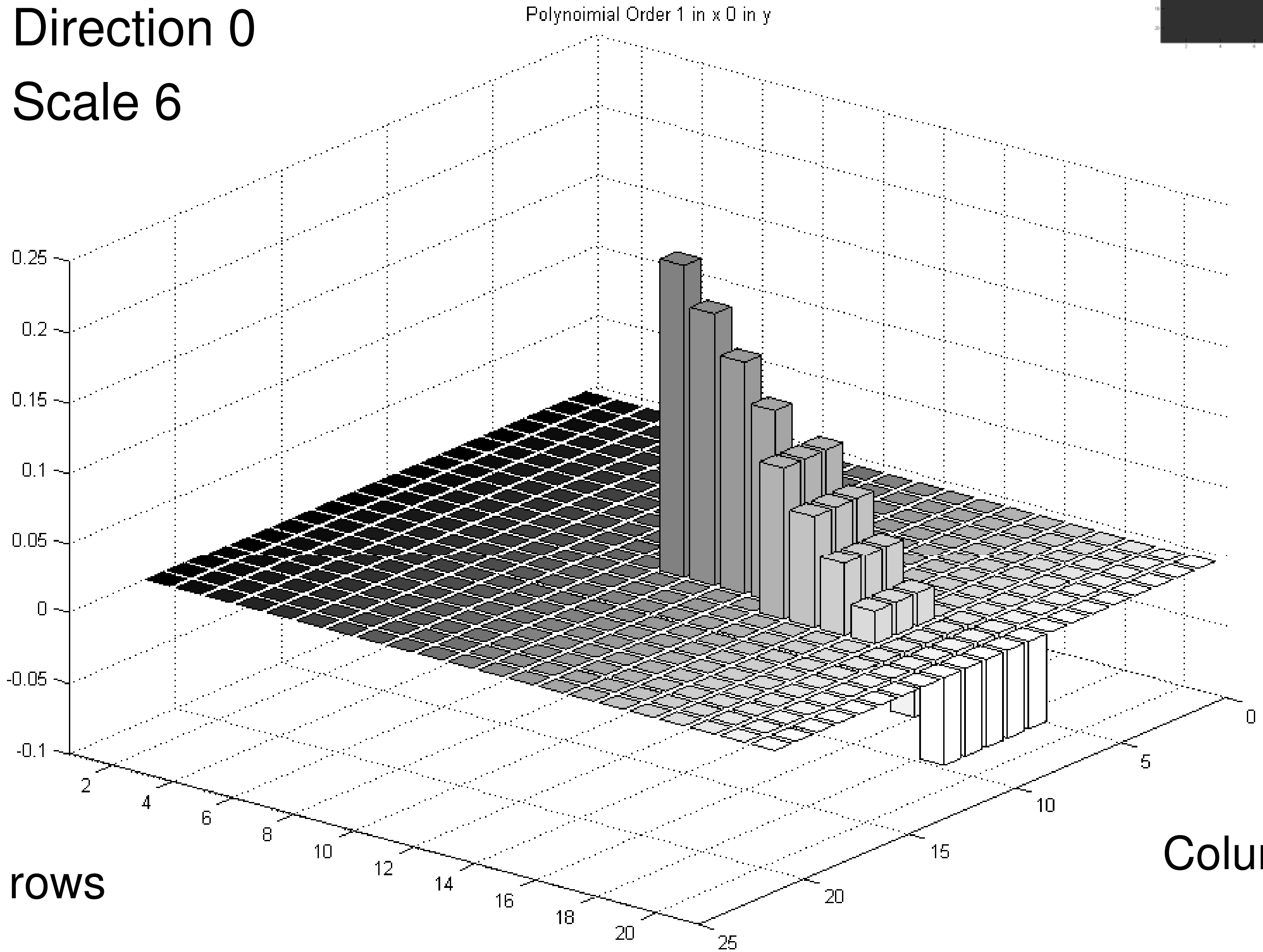




Example of LPA Anisotropic Kernels

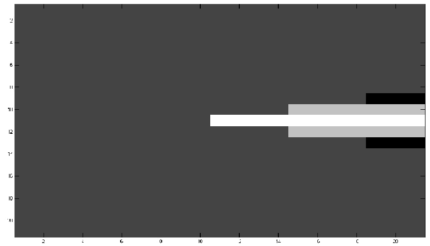
- Directional Kernel:
  - Order along columns: 0. Order along rows:1
  - Direction 0
  - Scale 6

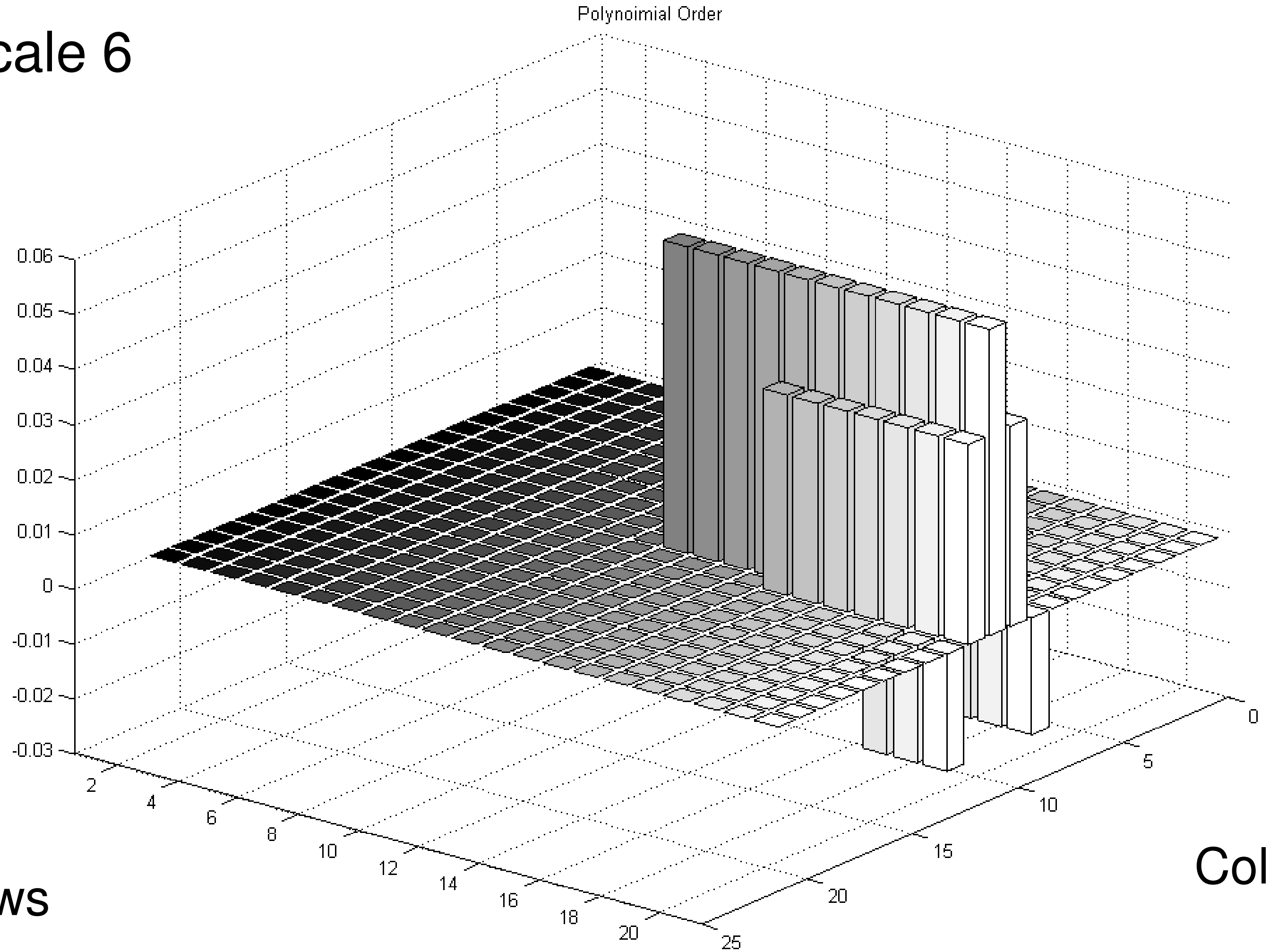




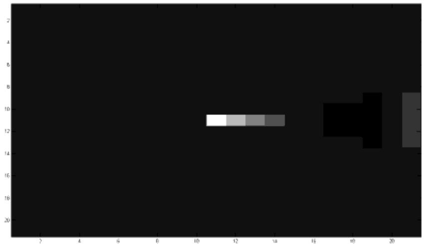
Example of LPA Anisotropic Kernels

- Directional Kernel:
  - Order along columns: 2. Order along rows: 0
  - Direction 0
  - Scale 6

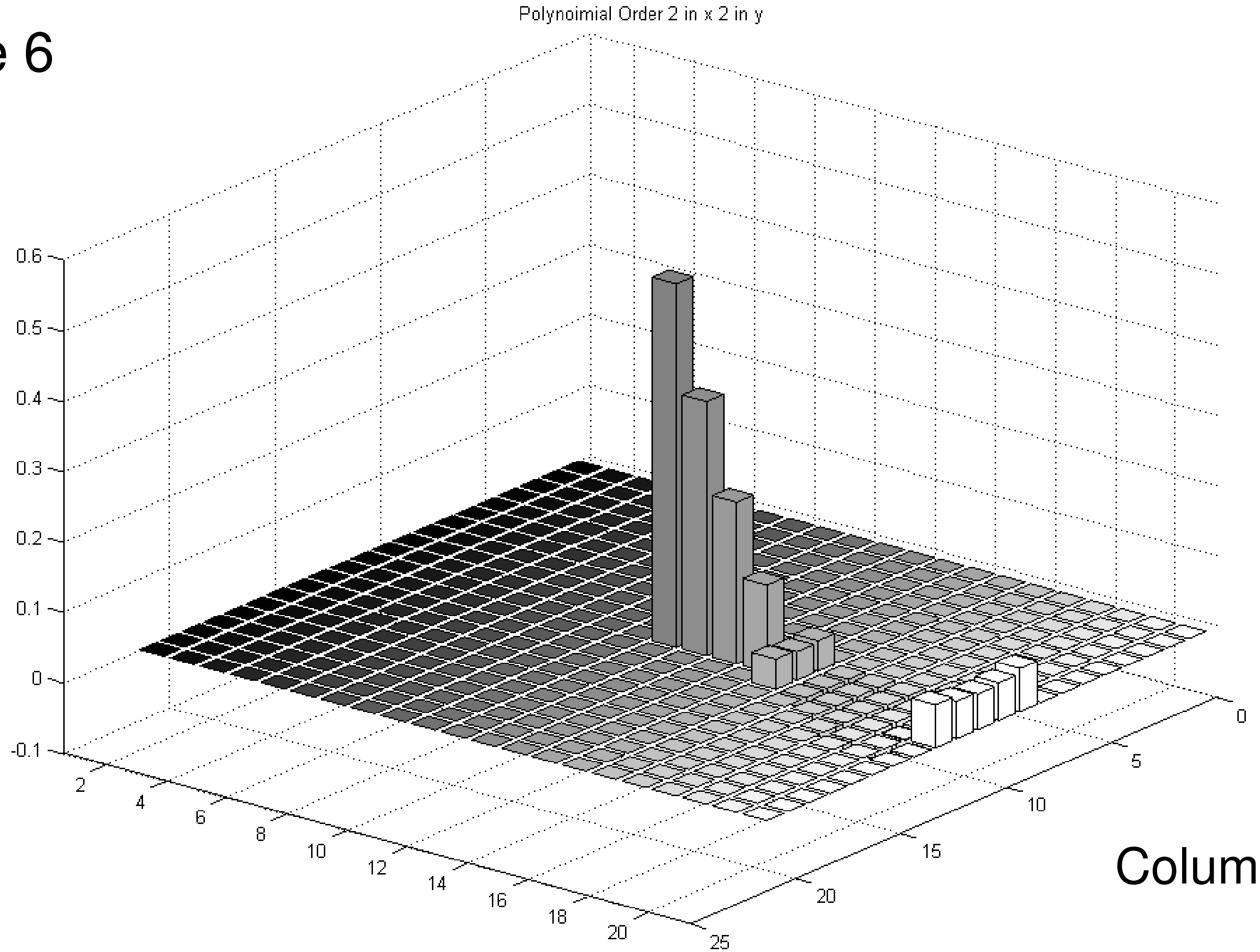




Example of LPA Anisotropic Kernels

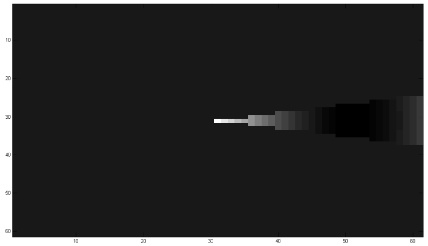


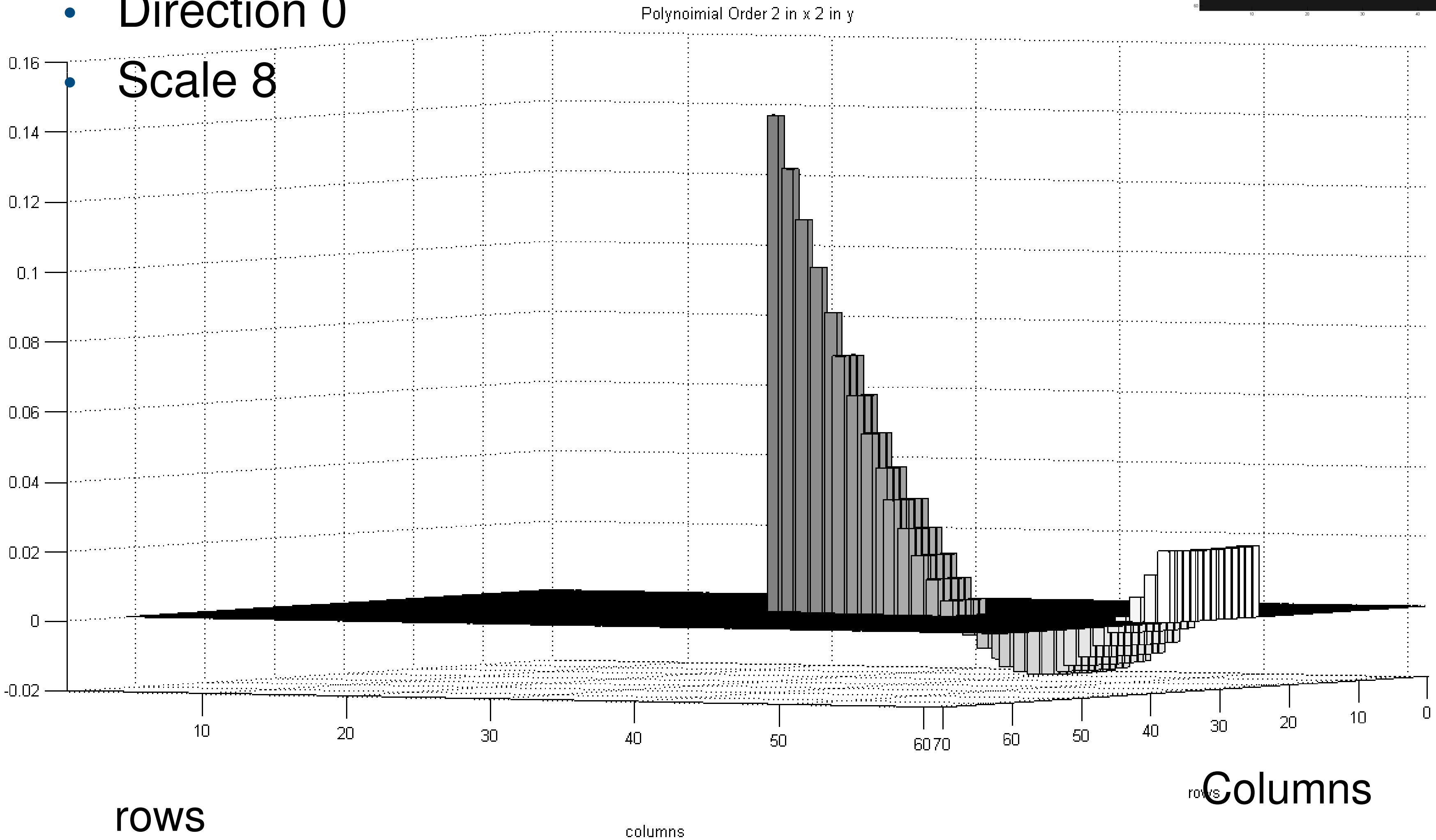
- Directional Kernel:
  - Order along columns: 2. Order along rows: 2
  - Direction 0
  - Scale 6



Example of LPA Anisotropic Kernels

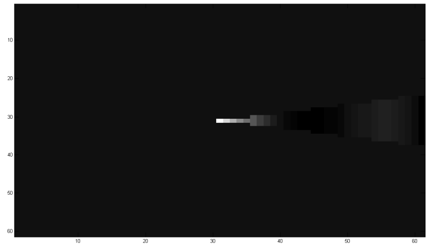
- Directional Kernel:
  - Order along columns: 2. Order along rows: 1
  - Direction 0
  - Scale 8

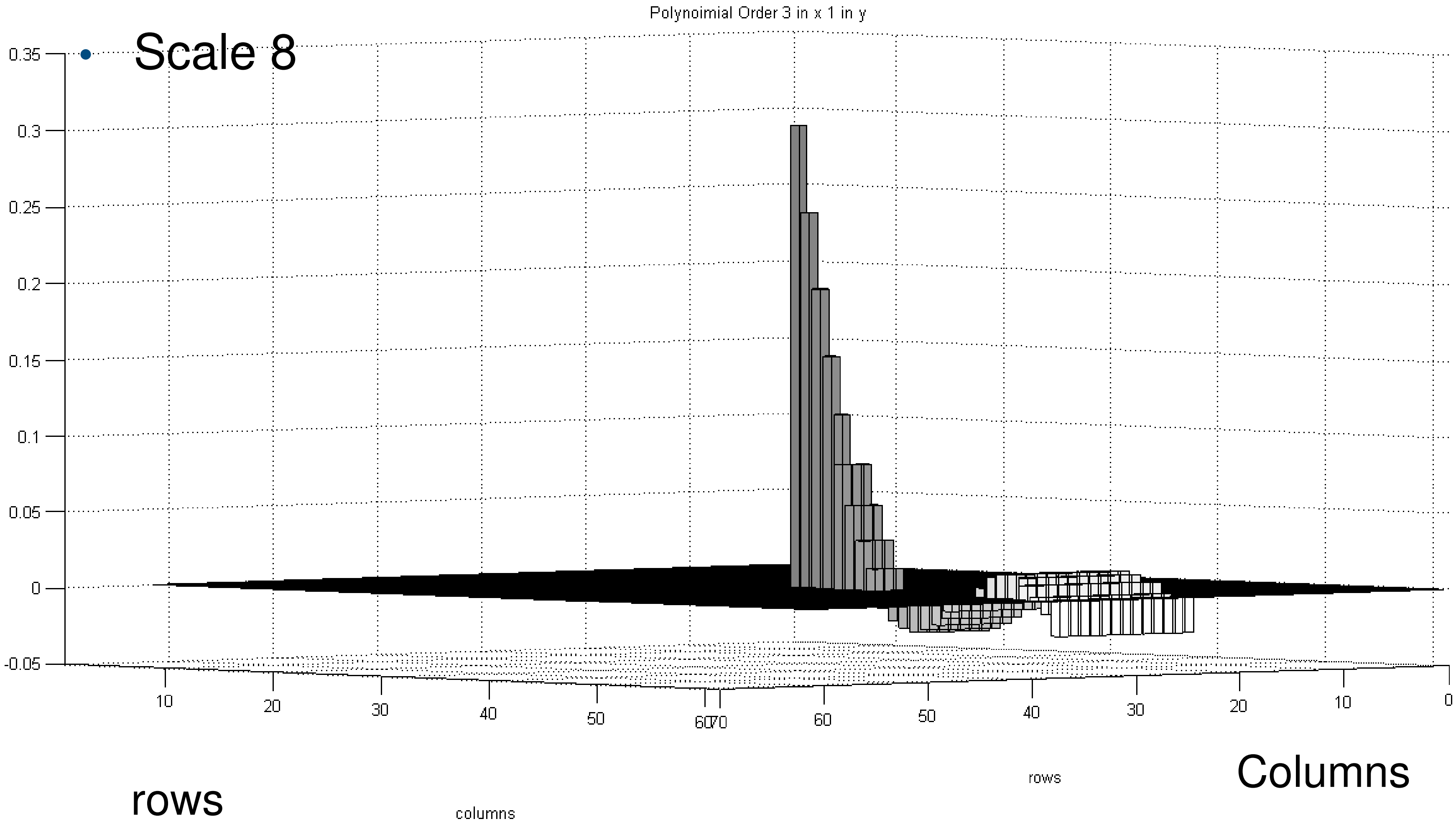




Example of LPA Anisotropic Kernels

- Directional Kernel:
  - Order along columns: 1. Order along rows: 3
  - Direction 0
  - Scale 8





LPA Kernels for derivatives estimation

- Kernels can be defined also for polynomial derivatives:

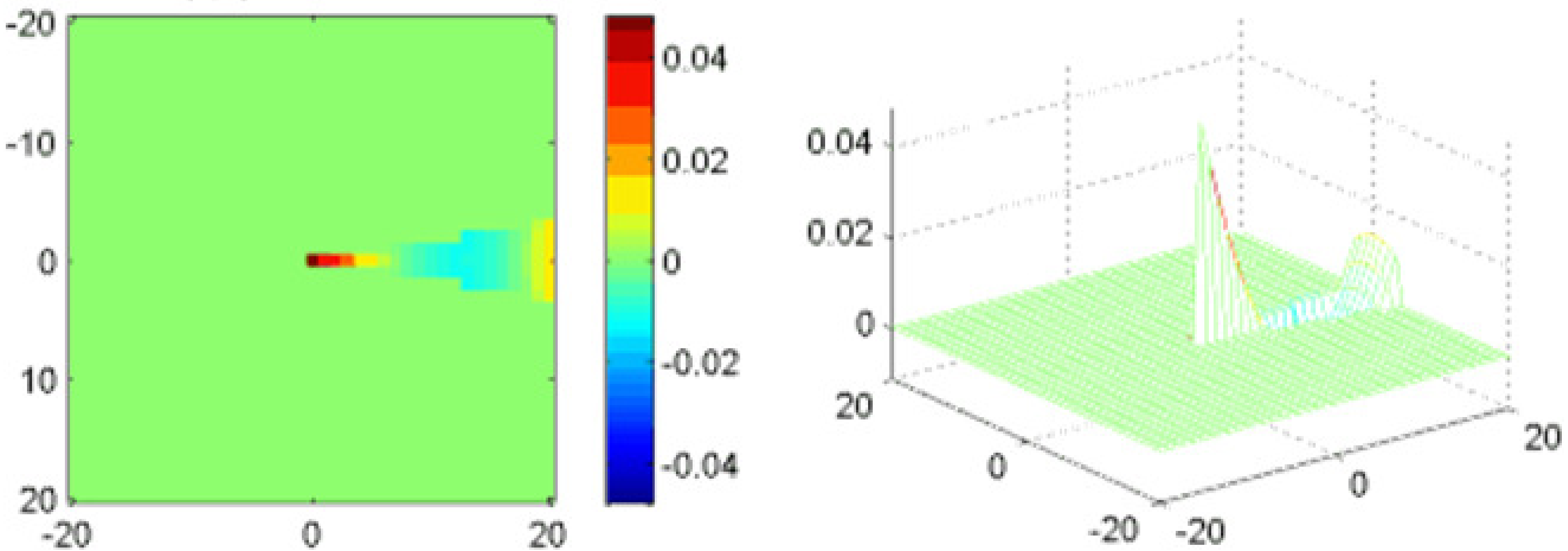
$$p_{h,m} = \operatorname{argmin}_{p \in \mathcal{P}_m} \sum_{x_s \in X} w_h(x_0 - x_s)\left(z(x_s) - p(x_s)\ \right)^2$$

$$\widehat{y_h}(x_0) = \frac{\partial^{r+c} p_{h,m}}{\partial x_1^r \partial x_2^c}(x_0)$$

- Also these estimates can be obtaine via a convolutional

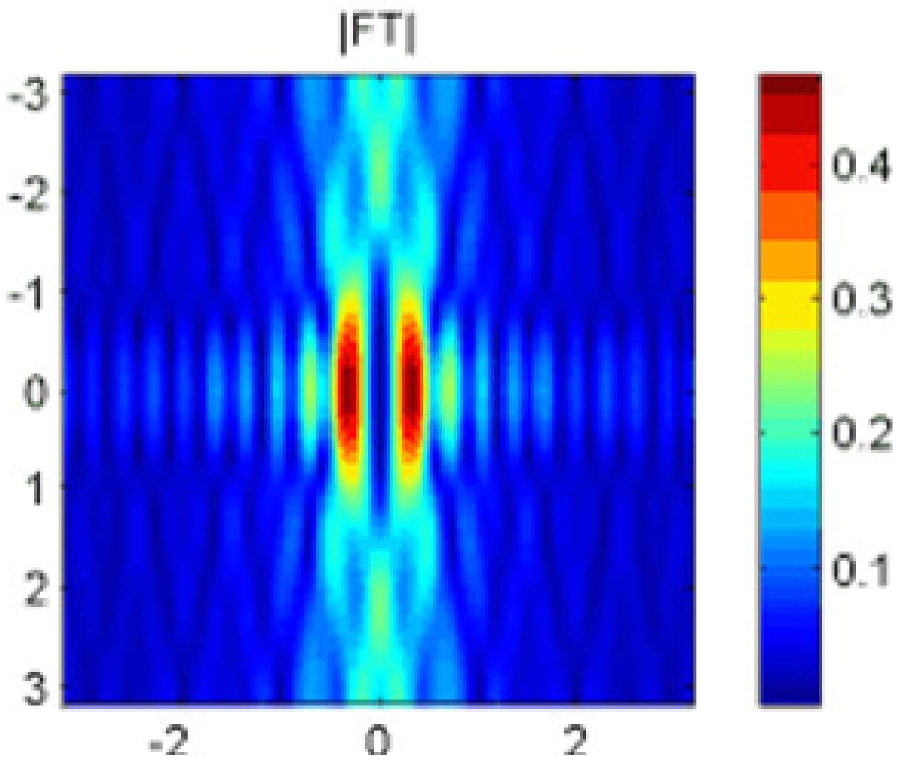
Example of LPA Anisotropic Kernels

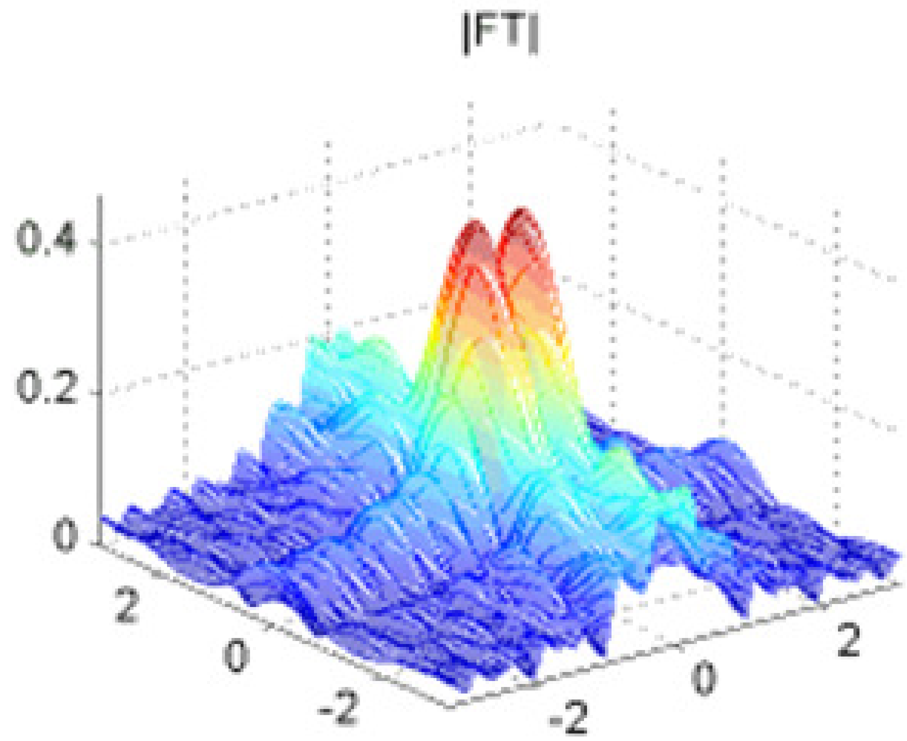
- Directional Kernel: 1° derivative along rows
  - Order along columns: 2. Order along rows: 2
  - Direction 0
  - Scale 6



Example of LPA Anisotropic Kernels

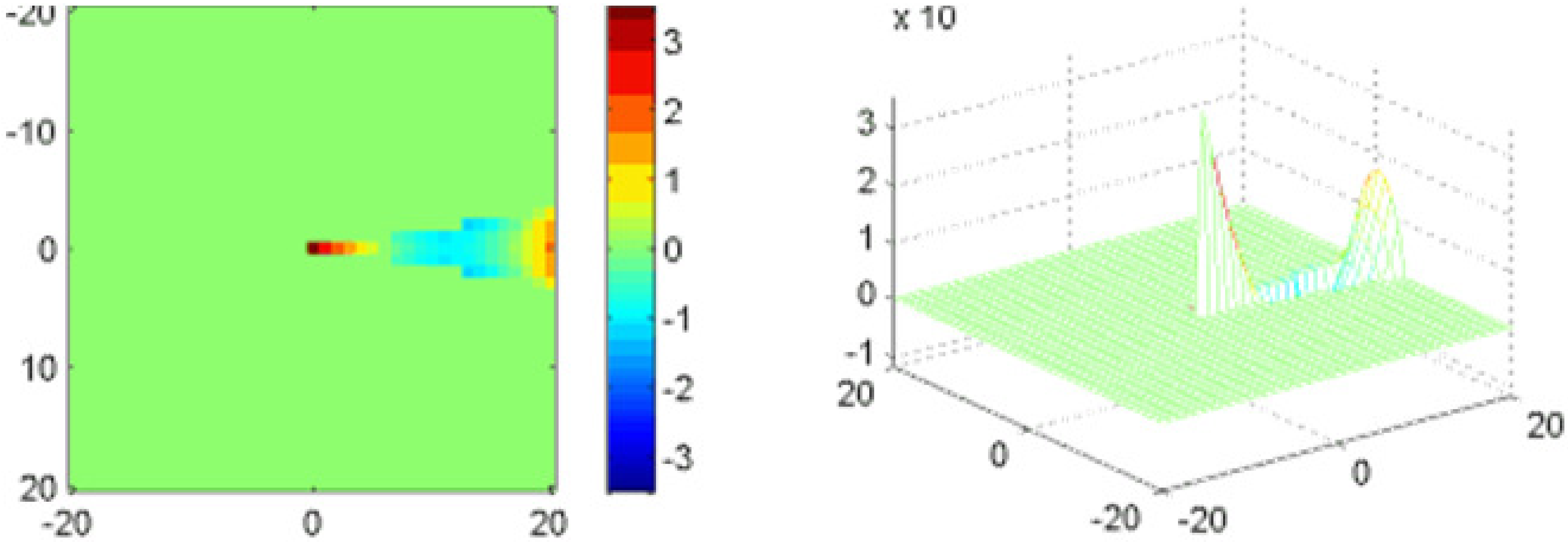
- Directional Kernel: 1° derivative along rows
  - Order along columns: 2. Order along rows: 2
  - Direction 0
  - Scale 6 Fourier Domain





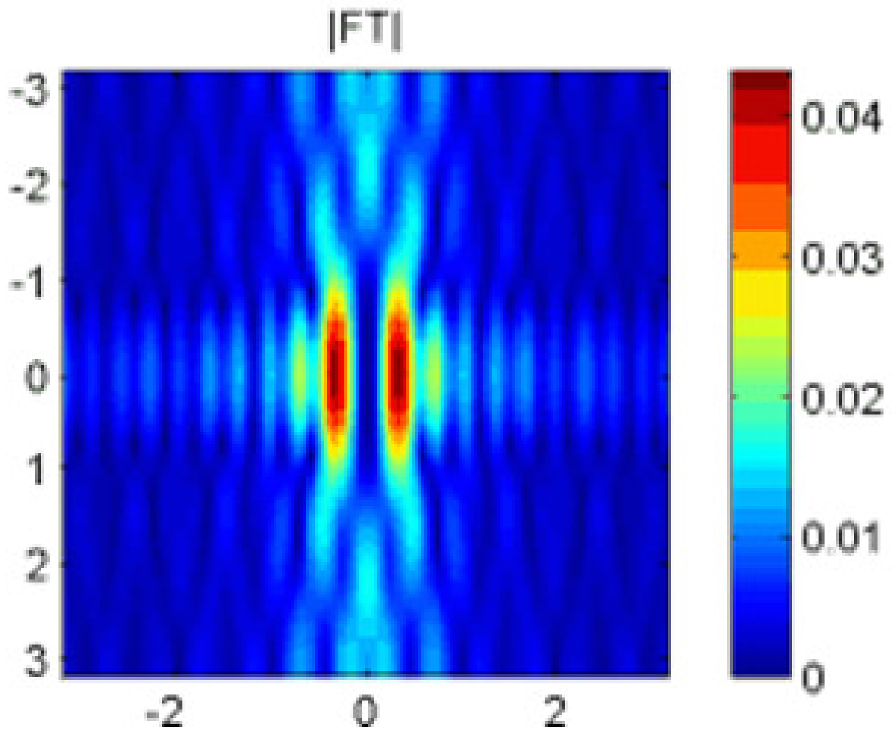
Example of LPA Anisotropic Kernels

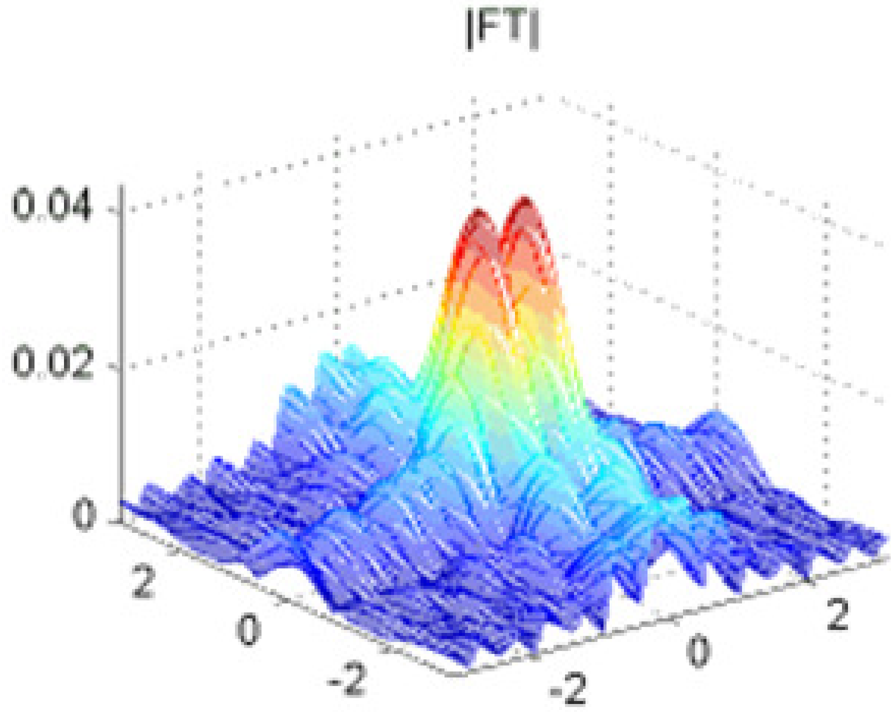
- Directional Kernel: 2° derivative along rows
  - Order along columns: 2. Order along rows: 2
  - Direction 0
  - Scale



Example of LPA Anisotropic Kernels

- Directional Kernel: 2° derivative along rows
  - Order along columns: 2. Order along rows: 2
  - Direction 0
  - Scale 6 Fourier Domain





LPA Benefits

- Unlike many other transforms which start from the continuous domain and then discretized, this technique **works directly** in the **multidimensional discrete domain**;
- The LPA kernels can be designed of **any dimension**, **non-separable and anisotropic** with **arbitrary orientation**, width and length;
- Any desirable **smoothness** of the kernel can be set.
- The kernel support can be flexibly shaped to any **desirable geometry**. In this way a special design can be done for complex form objects and specific applications.
- Both **smoothing** and corresponding **differentiating** directional **kernels** can be designed.

LPA Benefits

- These kernels are by definition **asymmetric**, allowing efficient edge adaptation. Traditional symmetric or nearly-symmetric supports tend to produce either so-called ringing artifacts or oversmoothing in the vicinity of the edges.

- The Directional *LPA* allows to consider **several different problems** within a **unified framework**.

- When using LPA one implicitly assumes that the original signal is well-approximated by a polynomial in a neighborhood of each pixel: this hypothesis suits perfectly pixel-wise parallelization.

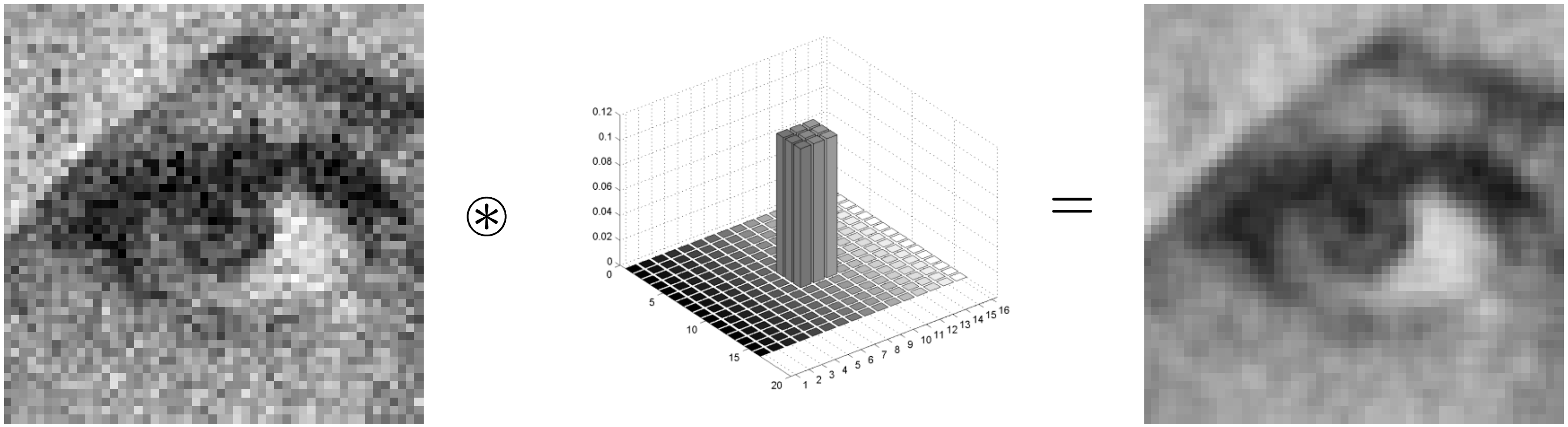
Katkovnik, V., K. Egiazarian, and J. Astola, "*Local Approximation Techniques in Signal and Image Processing*", SPIE Press, Monograph Vol. PM157, September 2006.

Foi, A., "*Anisotropic nonparametric image processing: theory, algorithms and applications,*" Ph.D. Thesis, Dip. di Matematica, Politecnico di Milano, April 2005.

Adaptiveness

- The choice of the scale parameter h is crucial as it controls the amount of smoothing in the estimation.

- Example on lena image

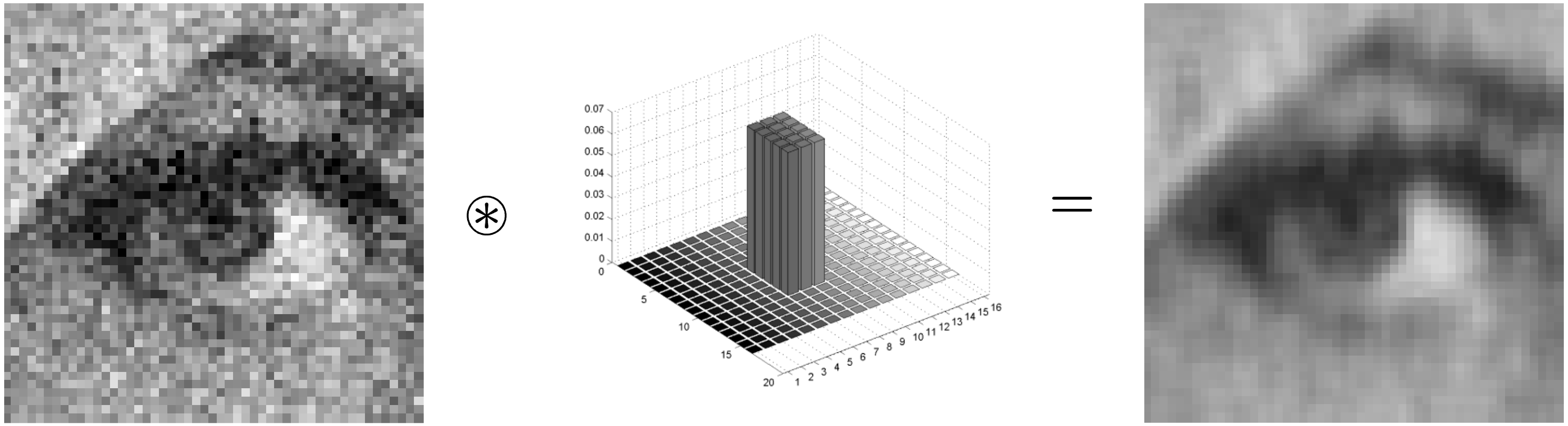


- Large h corresponds to less noisy output (i.e. lower variance) but typically it may result in higher bias
- Smaller h corresponds to noiser estimates (i.e. higher variance), less biased.

Adaptiveness

- The choice of the scale parameter h is crucial as it controls the amount of smoothing in the estimation.

- Example on lena image

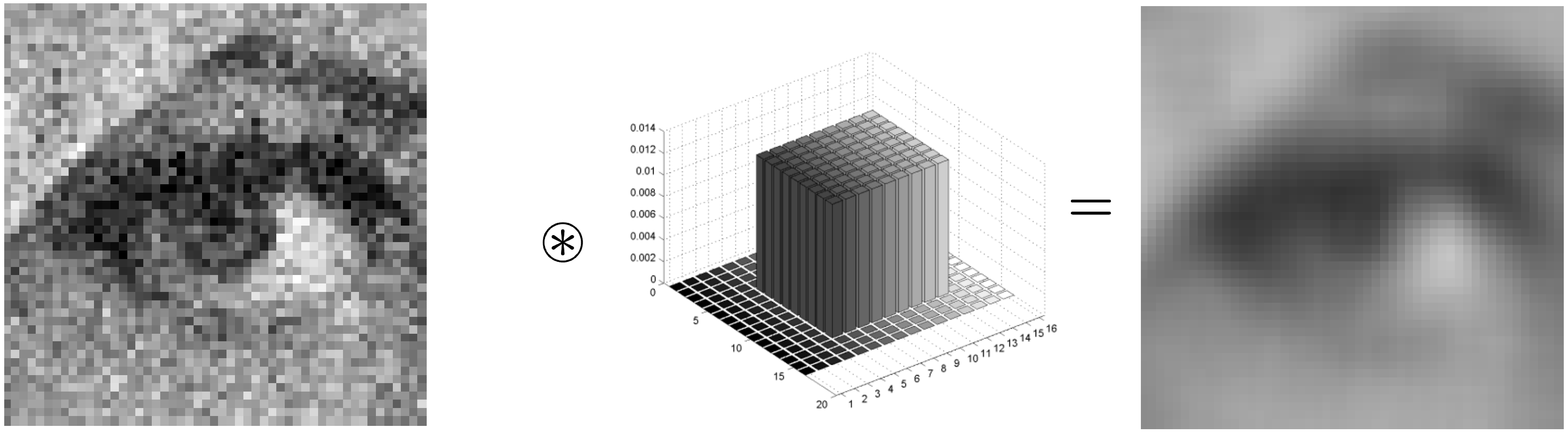


- Large h corresponds to less noisy output (i.e. lower variance) but typically it may result in higher bias
- Smaller h corresponds to noiser estimates (i.e. higher variance), less biased.

Adaptiveness

- The choice of the scale parameter h is crucial as it controls the amount of smoothing in the estimation.

- Example on lena image



- Large h corresponds to less noisy output (i.e. lower variance) but typically it may result in higher bias
- Smaller h corresponds to noiser estimates (i.e. higher variance), less biased.

Bias – Variance Trade Off

- Intiutively the scale parameters controls the trade off between bias and variance in the LPA estimates:
  - Bias $b_{\widehat{y}_h(x_0)} = y(x_0) - (y \circledast g_h)(x_0)$
  - Variance $\sigma^2_{\widehat{y}_h(x_0)} = (\sigma^2 \circledast g_h^2)(x_0) = \sigma^2 \|g_h\|_2^2$
- The following upper bound holds for the mean squared error

$$MSE(x) = E\left\{(y(x) - \widehat{y}_h(x_0))\right\} = b^2_{\widehat{y}_h(x_0)} + \sigma^2_{\widehat{y}_h(x_0)}$$

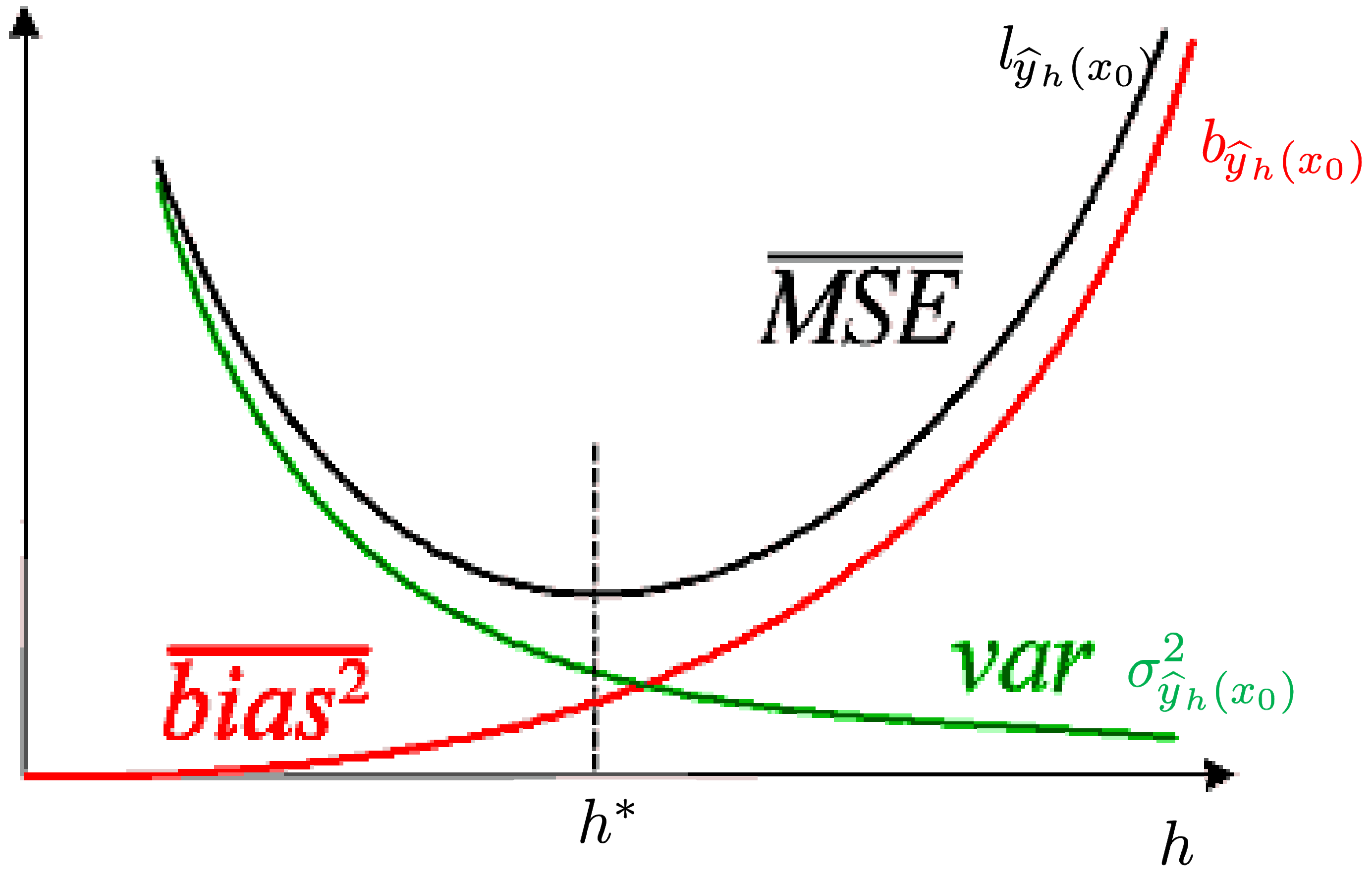
- Furthermore,the following asymptotic expressions hold for bias and the variance

$$b_{\widehat{y}_h(x_0)} \approx ch^a \qquad \sigma^2_{\widehat{y}_h(x_0)} \approx dh^{-b}$$

$$MSE(x) < l_{\widehat{y}_h(x_0)} = ch^a + dh^{-b}$$

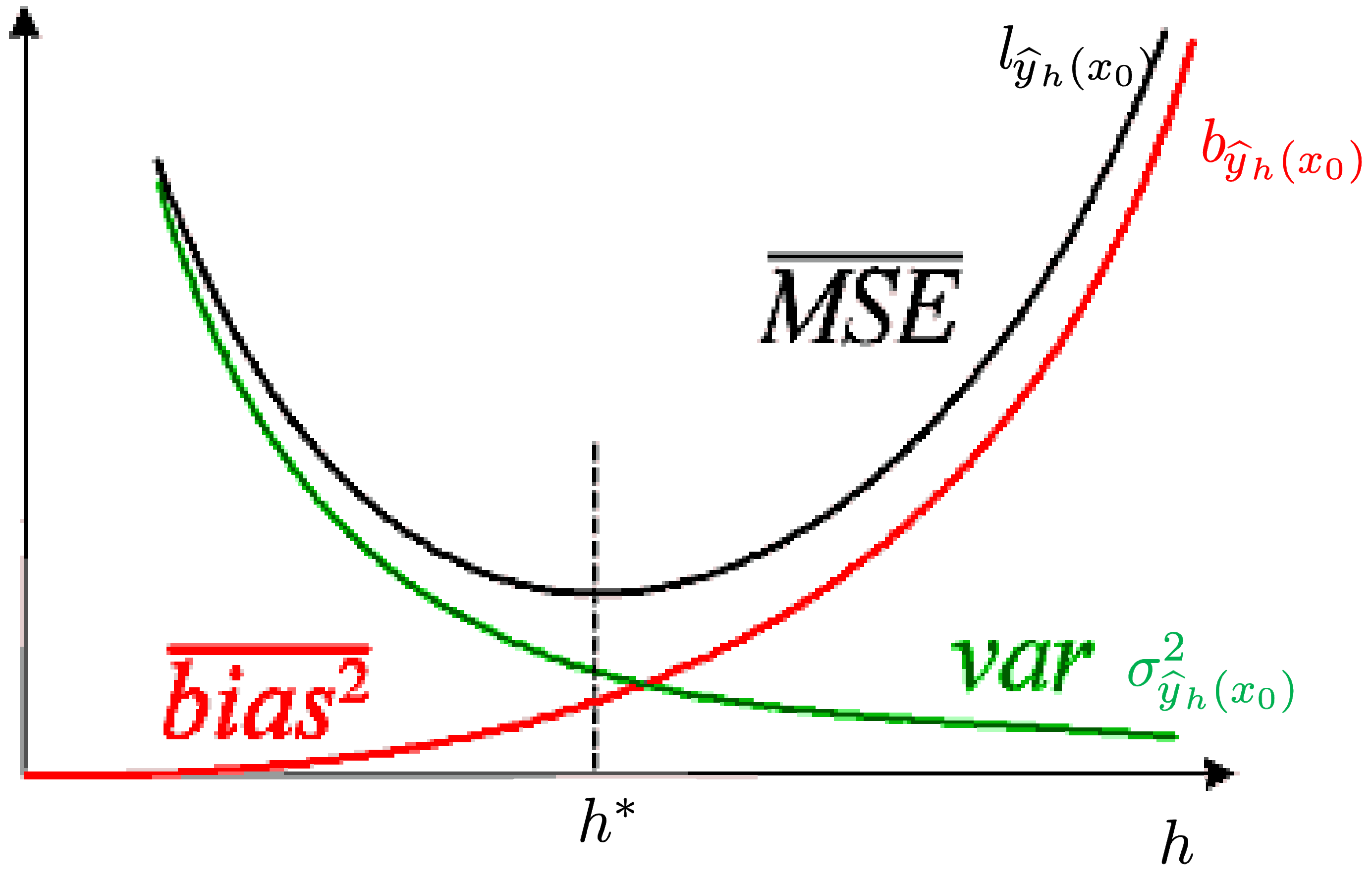
Thus

- In *practice* this hypothesis are enough for applying the ICI rule to determine the optimal scale h^*



Thus

- In *practice* this hypothesis are enough for applying the ICI rule to determine the optimal scale h^*



- The ICI rule is used to determine kernel scale h^+ that approximates h^*

The ICI rule

- Consider a fixed kernel direction d :
- For each pixel x , the estimates $\widehat{y}_{h,d}(x)$ are computed on a set of increasing scales $h \in H = \{h_j\}_{j=1}^{J}$
- For each estimate we can build a confidence interval as follows

$$\mathcal{D}_{h,d}(x) = [\widehat{y}_{h,d}(x) - \Gamma\sigma_{h,d},\ y\hat{_{h,d}}(x) + \Gamma\sigma_{h,d}]$$

where $\Gamma > 0$ is a tuining parameter

- The ICI rule yields a pointwise adaptive estimate $\widehat{y}_{h^+,d}(x)$ such that

$h^+ \approx h^*$ $h \in H$ in a sense that $\widehat{y}_{h^+,d}(x) \approx \widehat{y}_{h^*,d}(x)$

Goldenshluger, A., and A. Nemirovski, *"On spatial adaptive estimation of nonparametric regression"*, Math. Meth. Statistics, vol. 6, pp. 135-170, 1997

The ICI rule

- The ICI rule can be state as follows:

Consider the intersection of confidence intervals

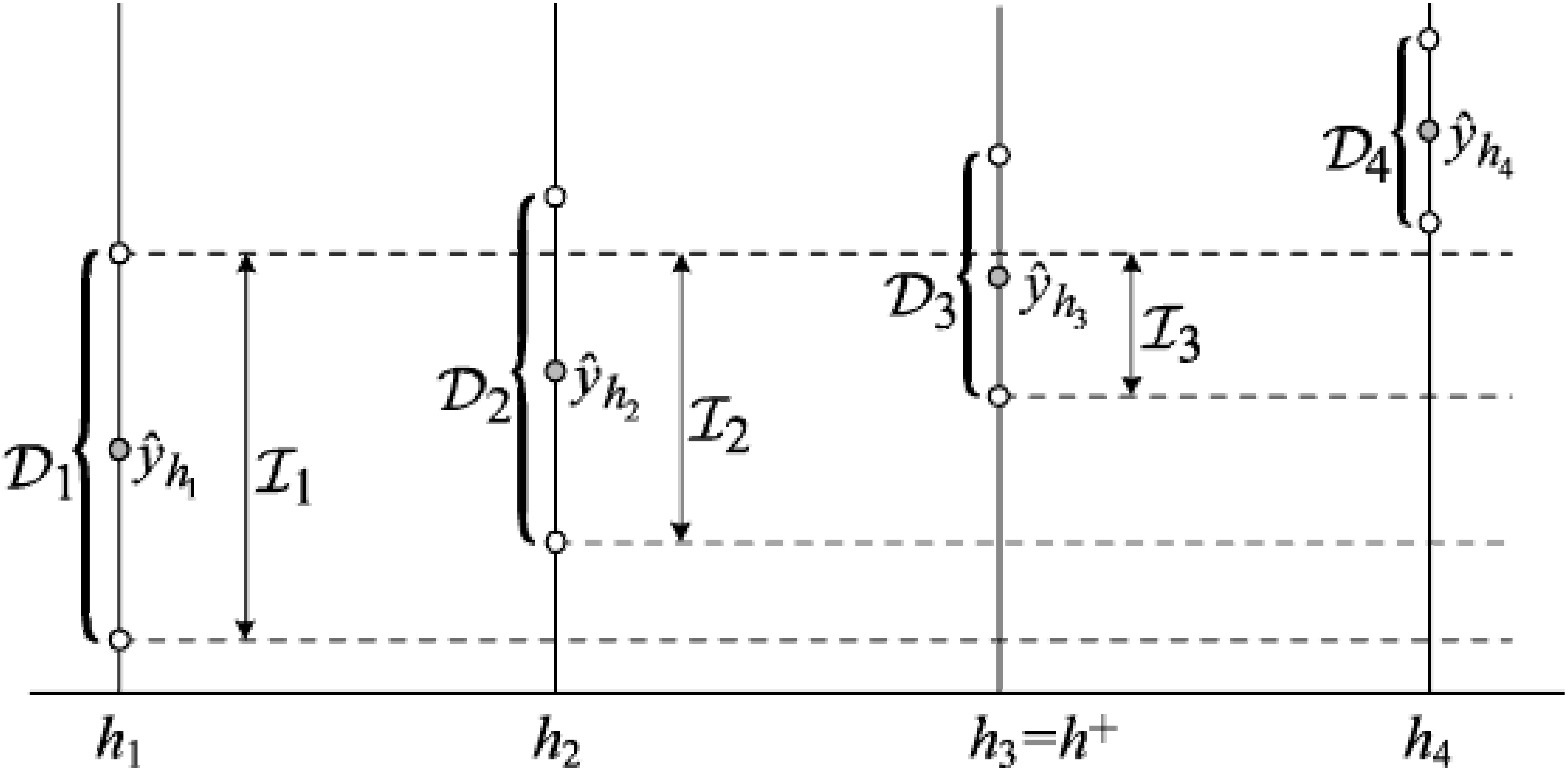
$$\mathcal{I}_{h,d}(x) = \bigcap_{h_j \leq h} \mathcal{D}_{h_j,d}(x)$$

where $\mathcal{D}_{h,d}(x) = [\widehat{y}_{h,d}(x) - \Gamma\sigma_{h,d},\ \hat{y_{h,d}}(x) + \Gamma\sigma_{h,d}],\ \Gamma > 0$.

Then let j^+ *be the largest of the scale indexes for which* $\mathcal{I}_{j^+,d} \neq \emptyset$ *and* $\mathcal{I}_{j^+ +1,d} = \emptyset$ *: then,*h^+ *is defined as* $h^+ = h_{j^+}$ *and the adaptive estimate is* $\widehat{y}_{h^+,d}(x)$ *.*

Goldenshluger, A., and A. Nemirovski, ***"On spatial adaptive estimation of nonparametric regression"***, Math. Meth. Statistics, vol. 6, pp. 135-170, 1997

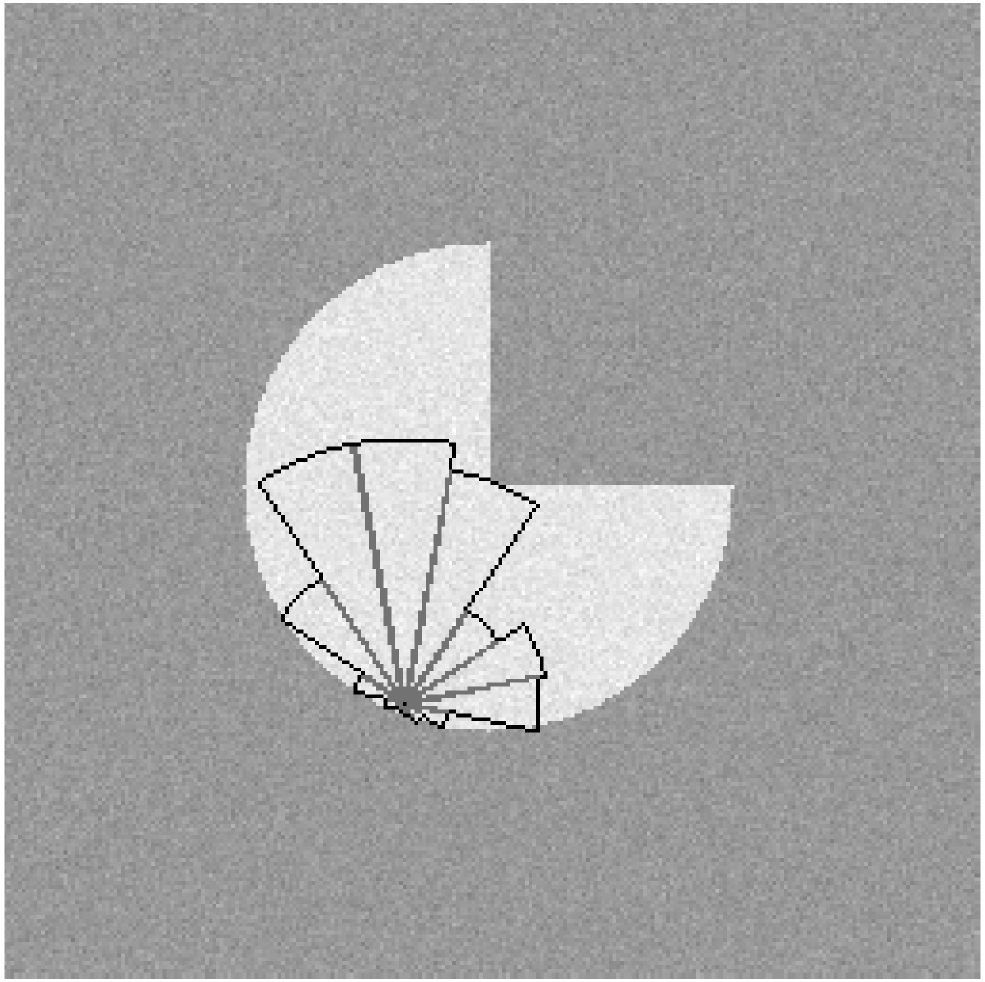
The ICI rule



Goldenshluger, A., and A. Nemirovski, *"On spatial adaptive estimation of nonparametric regression"*, Math. Meth. Statistics, vol. 6, pp. 135-170, 1997

Examples of adaptively selected neighborhoods

- Adaptively selected neighborhoods selected using the LPA-ICI rule

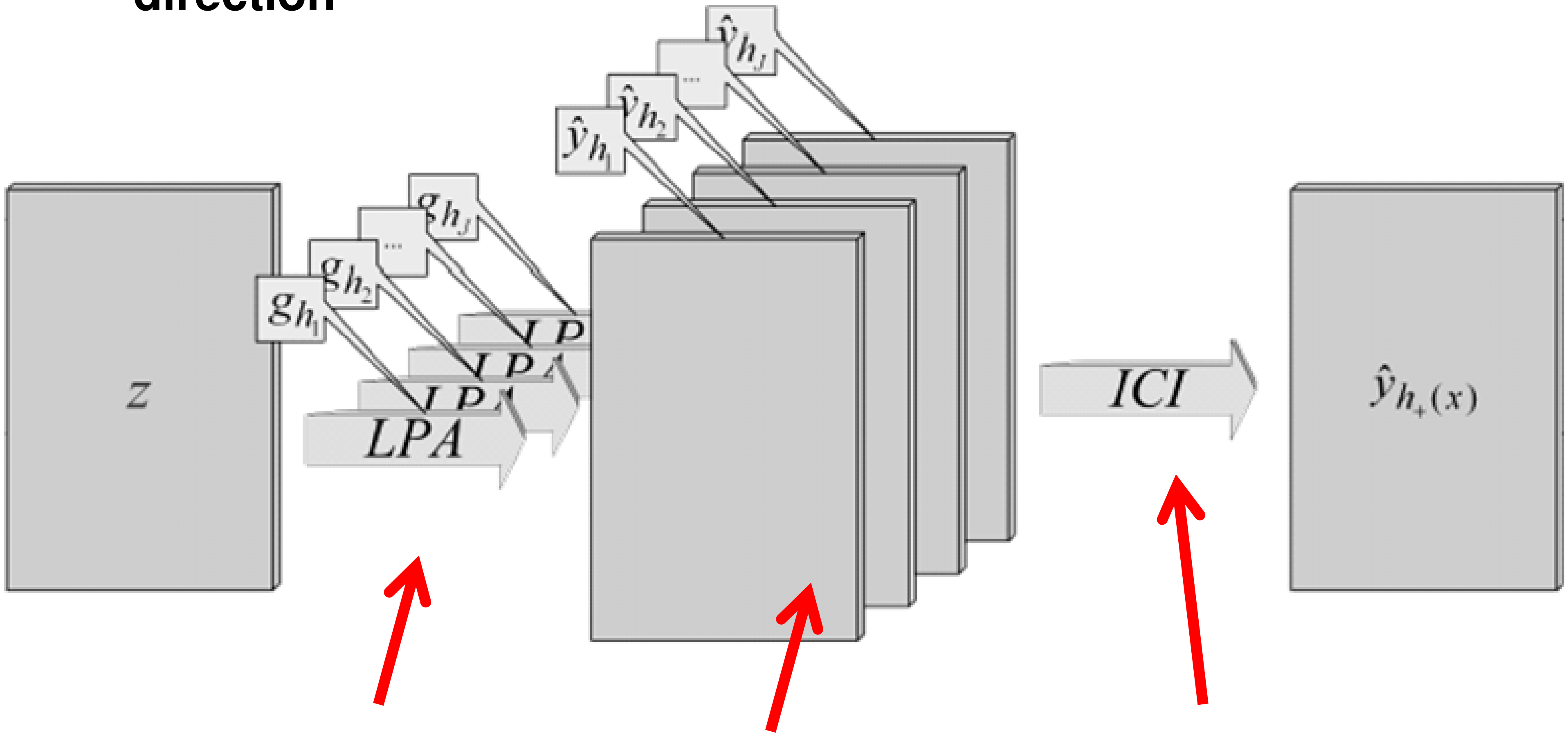


LPA-ICI Denoising Algorithm Details

1. Compute the ICI-selected scales along each direction
2. ICI-selected scales filtering and Update Directional Estimates
3. Fusing of directional estimates

ICI scale selection - Matlab implementation-

- It is typically faster to perform 2D-convolution **- given a kernel direction -**

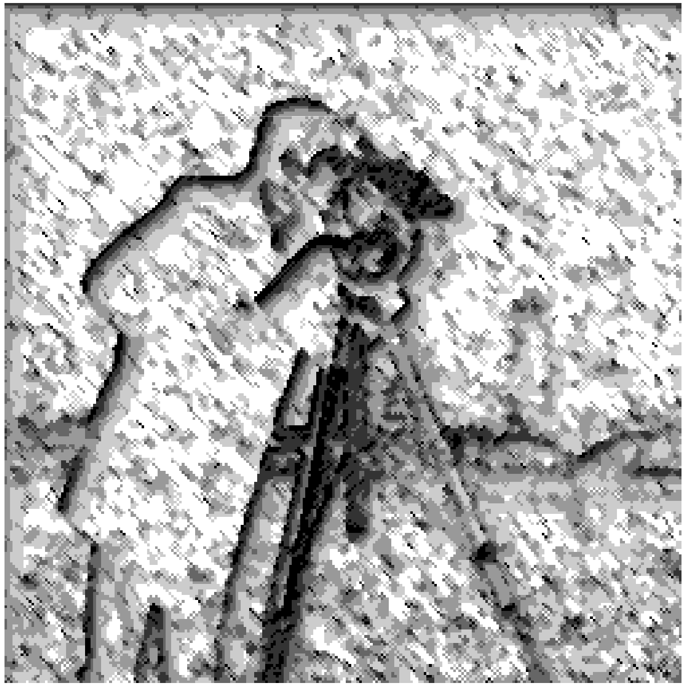


Compute *H* image convolutions, one per each kernel length

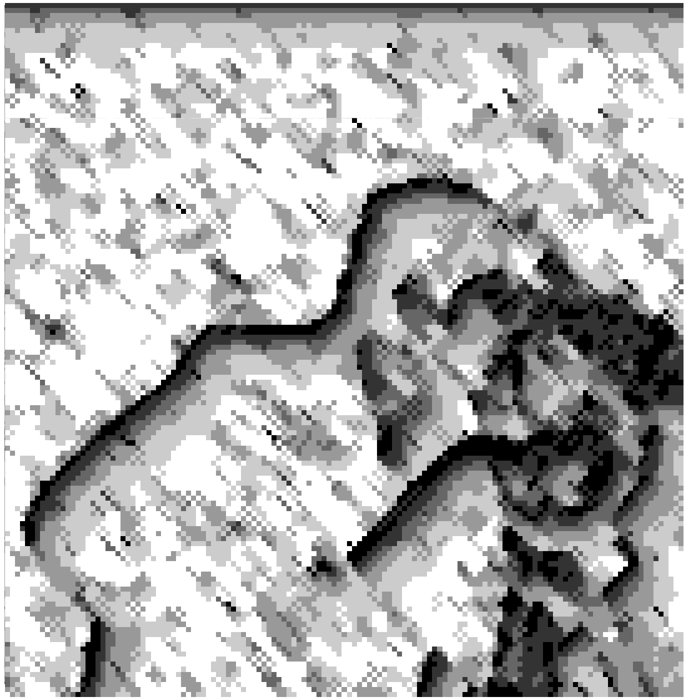
Compute the confidence intervals

For each pixel, compute the Intersection of Confidence Intervals

Example ICI selected Scales and Directional Estimates



Example ICI selected Scales and Directional Estimates


Step 2: ICI-selected scales filtering

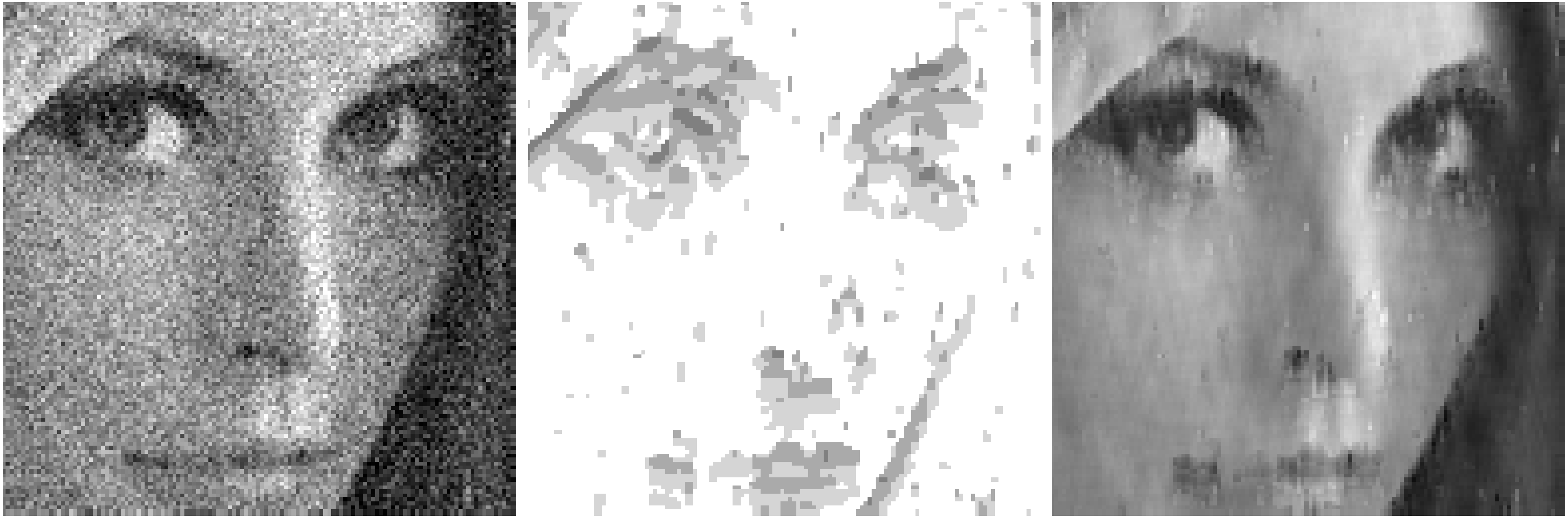
- Use a **median filter** or a **Weight Order Statistics** filter on the **ICI selected scales**
  - This operation is typically performed separately for the scales selected along each direction
  - When using WOS, these are directed “orthogonally” to anisotropic LPA-Kernels
- The “old” directional estimates are replaced by the LPA estimates and the standard deviations corresponding to the scales filtered.
- In such a way we **remove isolated** pixels where the **ICI selected “the wrong scale”**

Example ICI selected scales



Example ICI selected scales

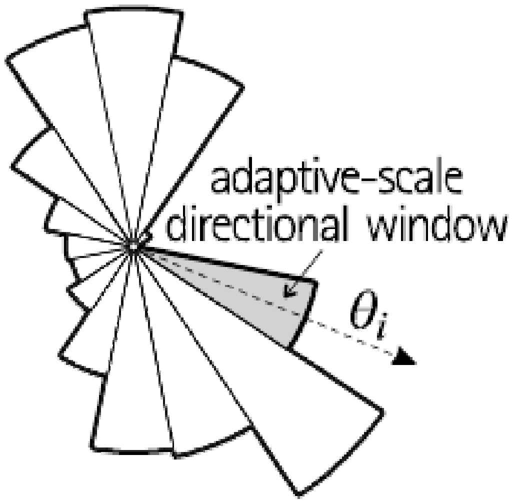
Example ICI selected scales

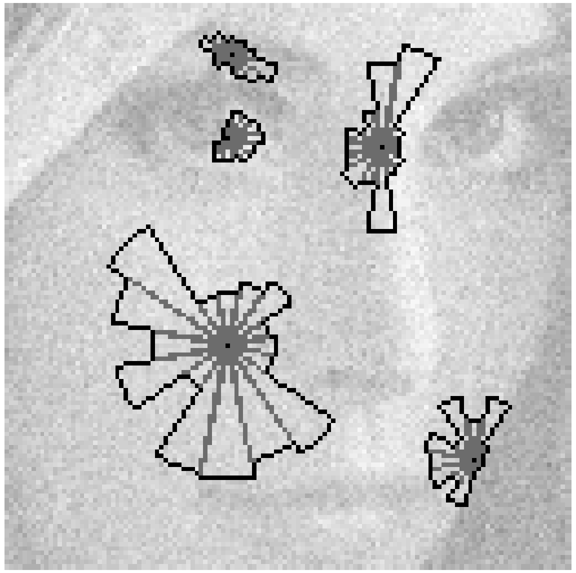


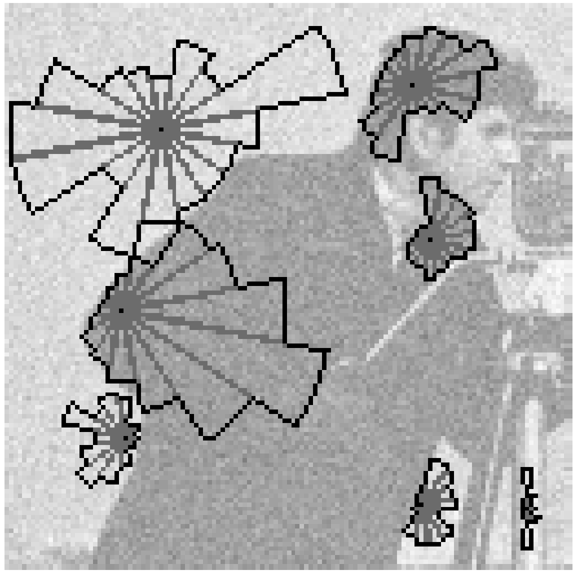
Step 3: Fusing of directional estimates

- For each pixel
  - Fuse the directional estimates with a convex linear combination the weights are determined by the inverse of the directional estimates variance

$$\hat{y}(x) = \sum_d \frac{\widehat{y}_{d,h^+}(x)}{\sigma^2_{d,h^+}(x)} \; / \; \sum_d \sigma^{-2}_{d,h^+}(x)$$







  - In such a way, larger weights are assigned to the less noisy estimates

Example of Final Estimate

- Directional Estimates

Fused Estimate

Denoising Performance

- Details: original image (top left), anisotropic *LPA-ICI*, *ISNR*=8.2dB (top right), TI wavelets (DB4), *ISNR*=7.4dB (bottom left), TI wavelets (Haar), *ISNR*=7.8dB (bottom right).

Implementation Details

1. Compute the ICI-selected scales along each direction
2. ICI-selected scales filtering and Update Directional Estimates
3. Fusing of directional estimates

These instructions will be performed on a Cuda Kernel, and parallelized for each pixel.

Step 1 + 3 : Compute ICI-selected scales

- For each pixel
  - For each LPA kernel direction *d*
  - *// initialize the ICI selected scale* $h_d^+(x) = J$
    - For each scale kernel $h \in H$

      // Compute the LPA estimate at x and its standard deviation

      $\widehat{y}_{h,d}(x) = (z \circledast k_{h,d})(x) \qquad \sigma_{h,d} = \|k_{h,d}\|_2^2 \sigma$
    - *// Determine confidence interval*

      $\mathcal{D}_{h,d}(x) = [\widehat{y}_{h,d}(x) - \Gamma\sigma_{h,d},\ \hat{y_{h,d}}(x) + \Gamma\sigma_{h,d}]$
    - *// Determine Intersection of Confidence Intervals*

      $$\mathcal{I}_{h,d}(x) = \bigcap_{h_j \leq h} \mathcal{D}_{h_j,d}(x)$$
    - *If* $\mathcal{I}_{h,d} = \emptyset$
      - *// The ICI selected scale is h-1*
      - $h_d^+(x) = h - 1$
      - *Break*
  - *// fuse the directional estimates*

Implementation Details

- The Γ parameter determine the width of Confidence Intervals:
  - Larger Γ tend to select larger scales
  - Smaller Γ tend to select smaller scales

- The convolution has to be computed also at image boundaries.
  - An efficient and practical solution in this case consist of padding the original image with a value much smaller than 0 (e.g. -10000)

Implementation Details

- The Γ parameter determine the width of Confidence Intervals:
  - Larger Γ tend to select larger scales
  - Smaller Γ tend to select smaller scales
- The convolution has to be computed also at image boundaries.
  - An efficient and practical solution in this case consist of padding the original image with a value much smaller than 0 (e.g. -10000)

Add PADDING area

Implementation Details

- The Γ parameter determine the width of Confidence Intervals:
  - Larger Γ tend to select larger scales
  - Smaller Γ tend to select smaller scales

- The convolution has to be computed also at image boundaries.
  - An efficient and practical solution in this case consist of padding the original image with a value much smaller than 0 (e.g. -10000)

Add PADDING area

Compute convolution against the padded image

Implementation Details

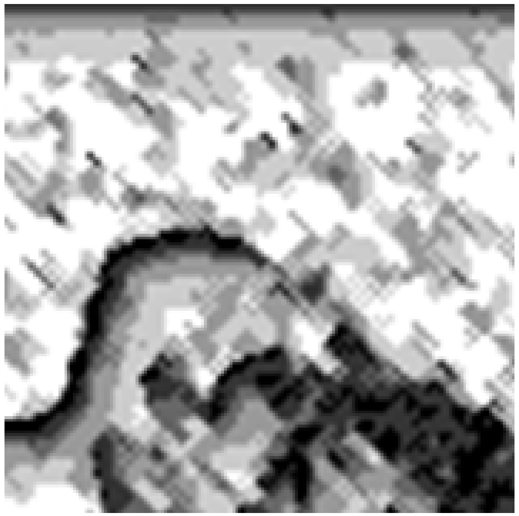
- The Γ parameter determine the width of Confidence Intervals:
  - Larger Γ tend to select larger scales
  - Smaller Γ tend to select smaller scales

- The convolution has to be computed also at image boundaries.
  - An efficient and practical solution in this case consist of padding the original image with a value much smaller than 0 (e.g. -10000)

Add PADDING area

Compute convolution against the padded image

Compute scales on PADDING free image



Implementation Details

- The Γ parameter determine the width of Confidence Intervals:
  - Larger Γ tend to select larger scales
  - Smaller Γ tend to select smaller scales

- The convolution has to be computed also at image boundaries.
  - An efficient and practical solution in this case consist of padding the original image with a value much smaller than 0 (e.g. -10000)

- The value of the noise standard deviation is assumed known (and here it will be provided). Typically it can be estimated using *MAD* estimator

Donoho, D.L., and I.M. Johnstone, *"Ideal spatial adaptation via wavelet shrinkage"*, Biometrika, n. 81, pp. 425-455, 1994

Implementation Details

- The Γ parameter determine the width of Confidence Intervals:
  - Larger Γ tend to select larger scales
  - Smaller Γ tend to select smaller scales

- The convolution has to be computed also at image boundaries.
  - An efficient and practical solution in this case consist of padding the original image with a value much smaller than 0 (e.g. -10000)

- The value of the noise standard deviation is assumed known (and here it will be provided). Typically it can be estimated using *MAD* estimator

- As an error metric you can use RMSE

$$RMSE(\widehat{y}) = \sqrt{\frac{\sum_x \left(\widehat{y}(x) - y(x)\right)^2}{\#X}}$$

Other Applications

- Deblurring

Other Applications

- Deblurring

Method Experiment	1	2	3	4
LPA-ICI directional	8.23	7.78	6.04	3.76
GEM (Dias)	8.10	7.47	5.17	–
EM (Figueiredo and Nowak)	7.59	6.93	4.88	2.94
ForWaRD (Neelamani et al.)	7.30	6.75	5.07	2.98

ISNR

Other Applications

- Inverse Halftoning

Other Applications

- Inverse Halftoning

original grayscale image (top left), binary Floyd-Steinberg halftone (top right), anisotropic *LPA-ICI* estimate, *PSNR*=32.4dB (bottom left) and wavelet-based *WinHD* estimate (Rice Univ.), *PSNR*=32.1dB (bottom right)