Fast Bilateral Filtering

Sylvain Paris, Adobe

sparis@adobe.com

- Manipulating the texture in a photo is a central operation in many tasks.
- Let’s look at a few examples…

Photographic Style Transfer

[Bae 06]

input

Photographic Style Transfer

[Bae 06]

output

Tone Mapping
[Durand 02]

HDR input

Tone Mapping

[Durand 02]

output

Cartoon Rendition
[Winnemöller 06]

input

Cartoon Rendition

[Winnemöller 06]

output

Naïve Approach: Gaussian Blur

BLUR

HALOS

input

smoothed
(*structure, large scale*)

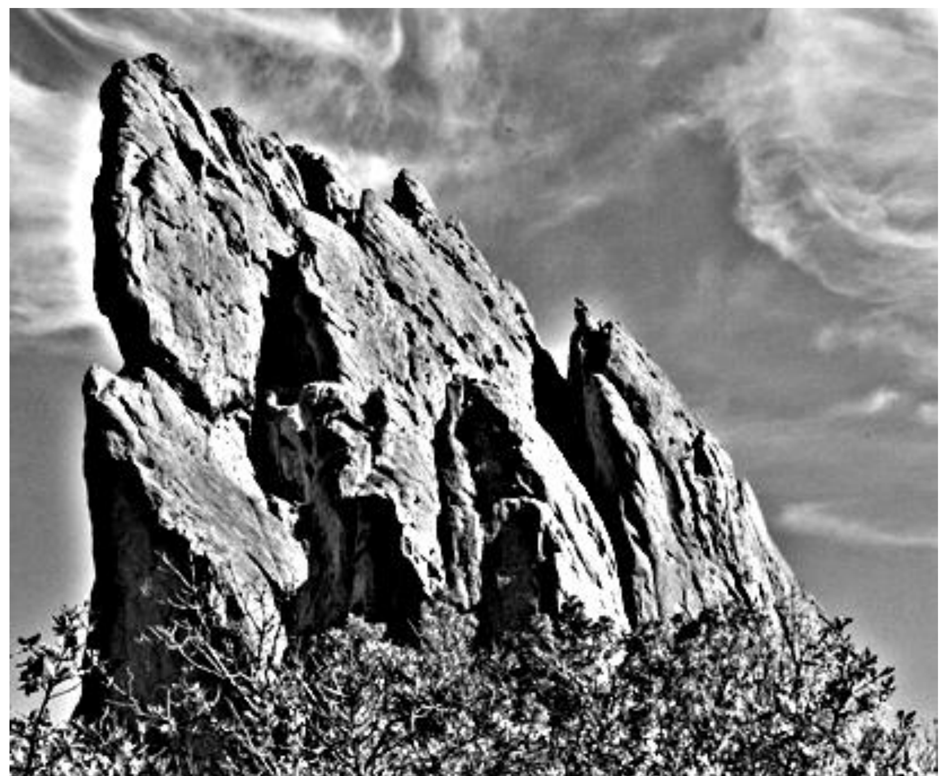
residual
(*texture, small scale*)

Gaussian Convolution

Impact of Blur and Halos

- If the decomposition introduces blur and halos, the final result is corrupted.

Sample manipulation: increasing texture (residual $\times$ 3)



Bilateral Filter: no Blur, no Halos

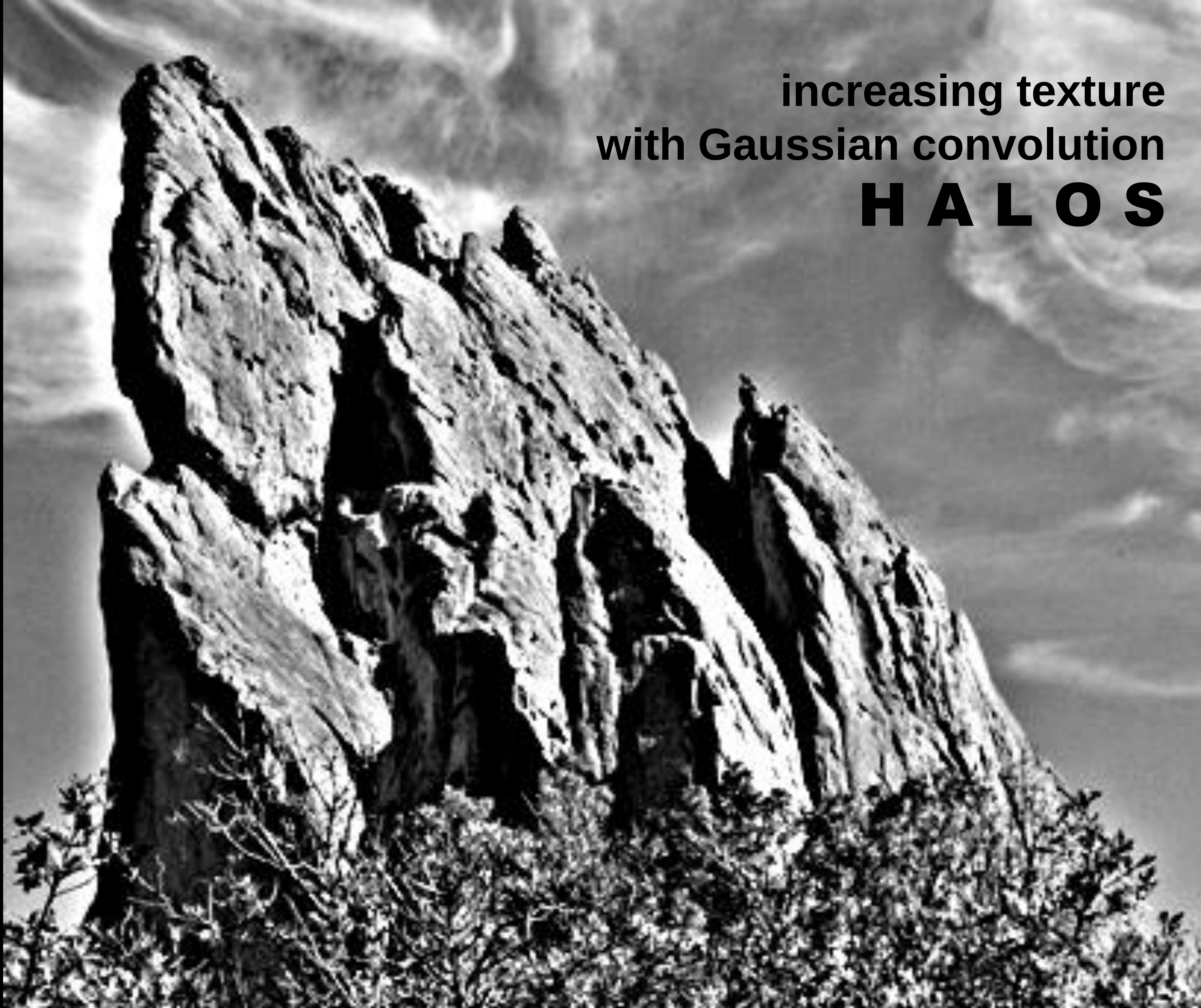
input

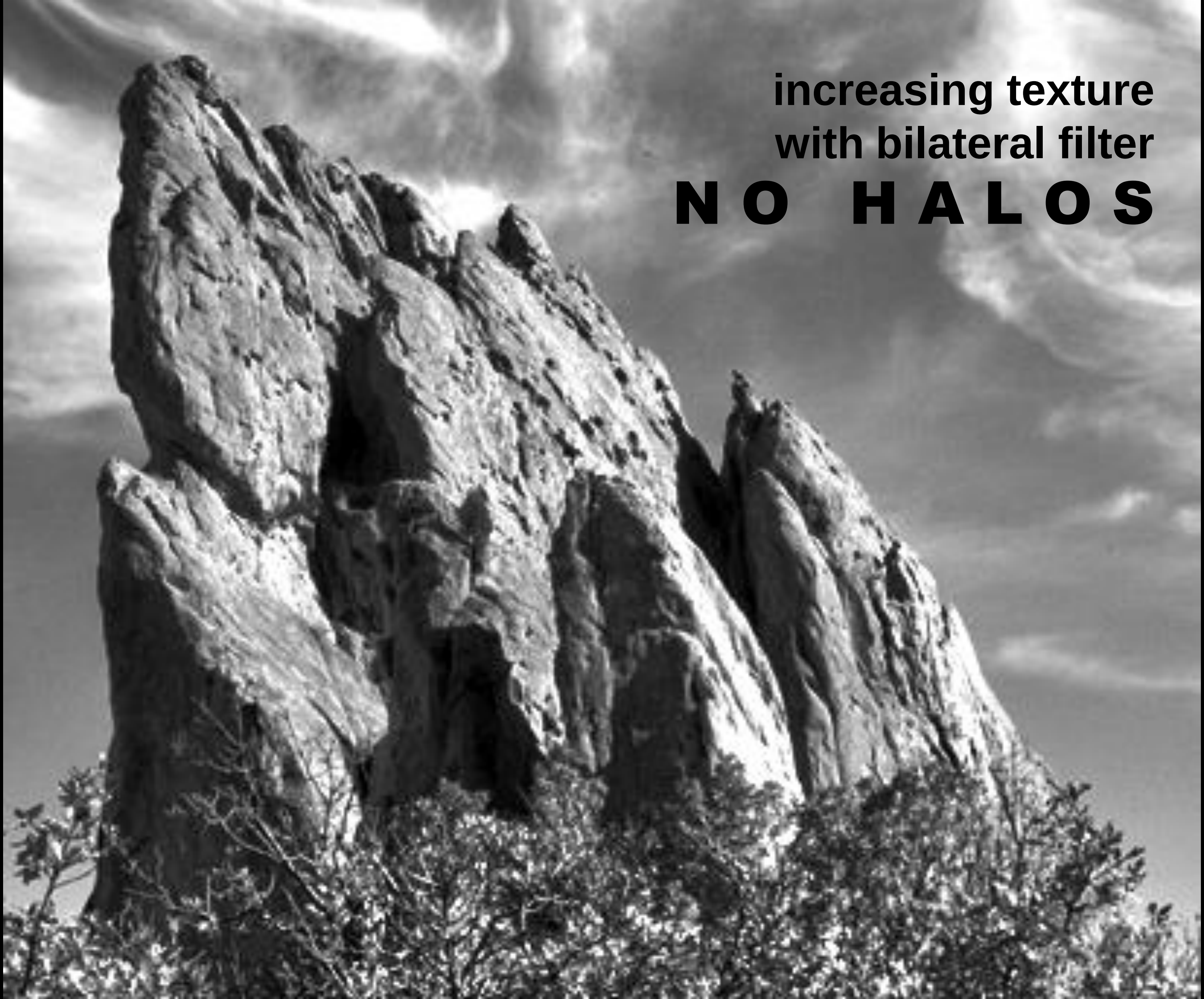
smoothed
(*structure, large scale*)

residual
(*texture, small scale*)

edge-preserving: Bilateral Filter


input


increasing texture
with Gaussian convolution
H A L O S


increasing texture
with bilateral filter
NO HALOS

Traditional Denoising versus Computational Photography

Edge-preserving filtering introduced for denoising.

- Denoising: decompose into signal + noise
  - Throw away noise
  - Small kernels
- Computational photography: decompose into base + detail
  - Detail is valuable
  - Large kernels

  ⇒ Bilateral filter [Aurich 95, Smith 97, Tomasi 98]

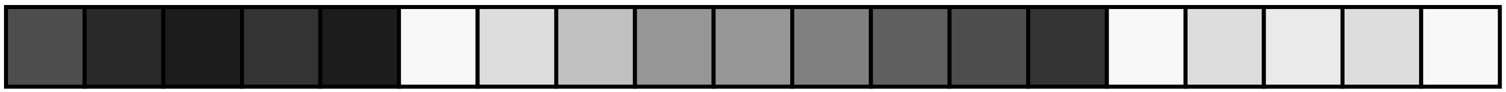
Objective of bilateral filtering

Smooth texture

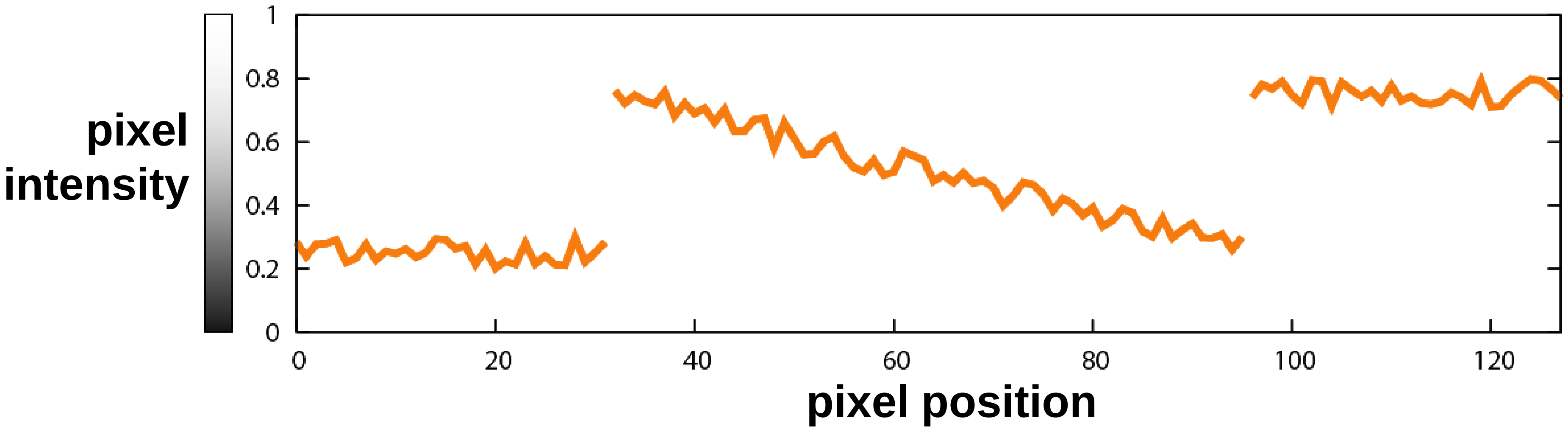
Preserve edges

Illustration a 1D Image

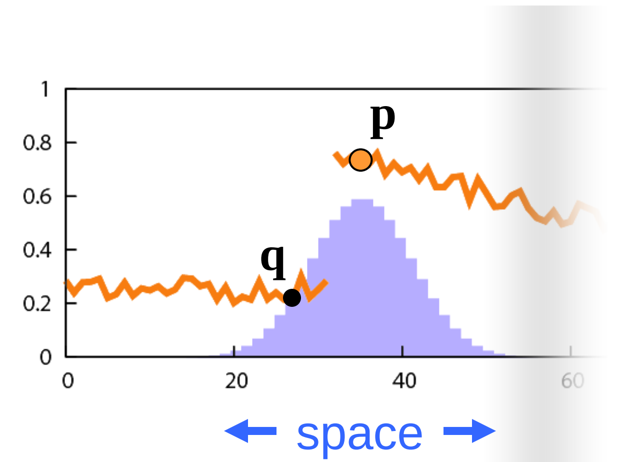
- 1D image = line of pixels



- Better visualized as a plot



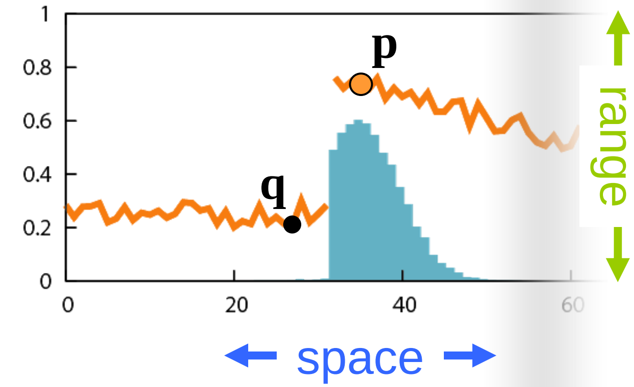
Definition



Gaussian blur

$$I^{\mathrm{b}}_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{\boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}}_{\text{space}} I_{\mathbf{q}}$$

- only spatial distance, intensity ignored



Bilateral filter

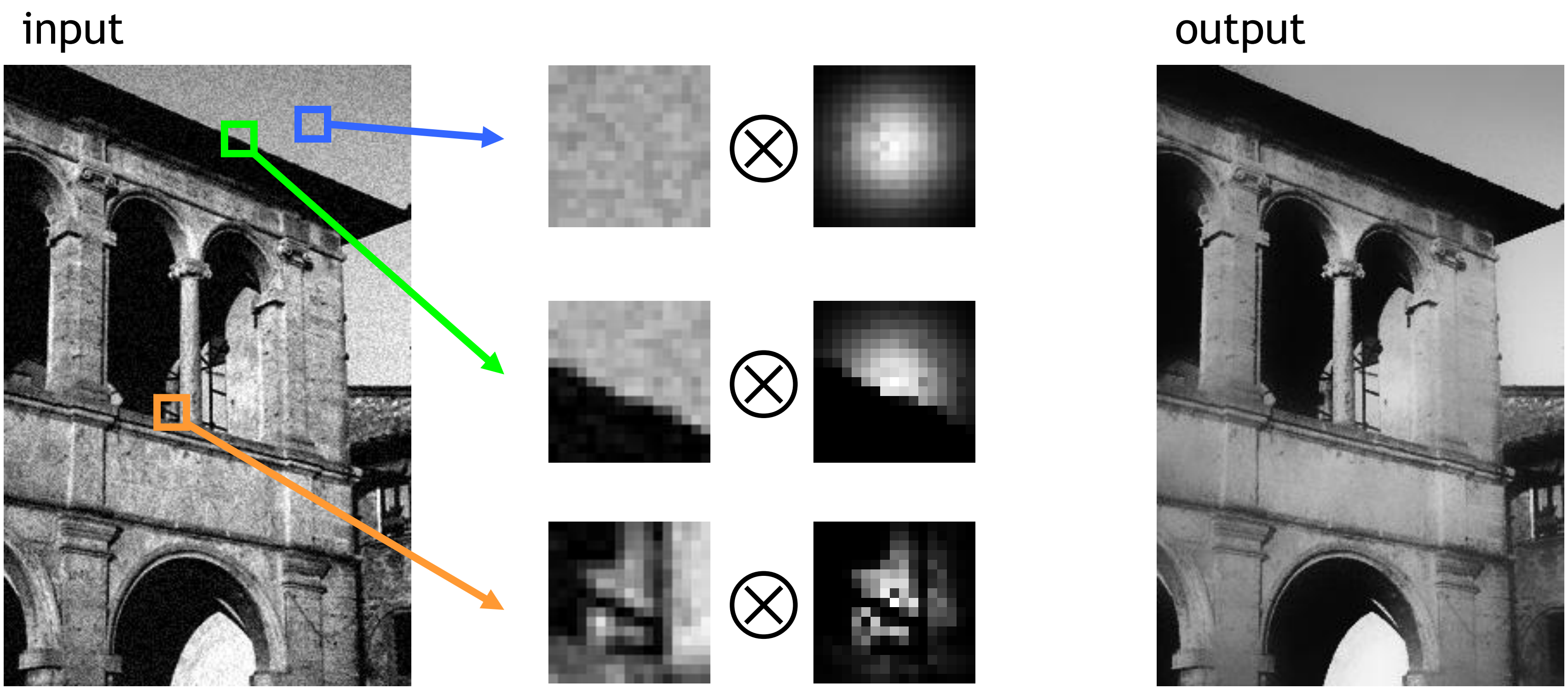
[Aurich 95, Smith 97, Tomasi 98]

$$I^{\mathrm{bf}}_{\mathbf{p}} = \underbrace{\boxed{\frac{1}{W^{\mathrm{bf}}_{\mathbf{p}}}}}_{\text{normalization}} \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{\boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}}_{\text{space}} \underbrace{\boxed{G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}}_{\text{range}} I_{\mathbf{q}}$$

- spatial and range distances
- weights sum to 1

Example on a Real Image

- Kernels can have complex, spatially varying shapes.



Bilateral Filter is Expensive

- Brute-force computation is slow (several minutes)
  - Two nested for loops:
    for each pixel, look at all pixels
  - Non-linear, depends on image content
    ⇨ no FFT, no pre-computation…

- Fast approximations exist [Durand 02, Weiss 06]
  - Significant **loss of accuracy**
  - **No formal understanding** of accuracy versus speed

Today

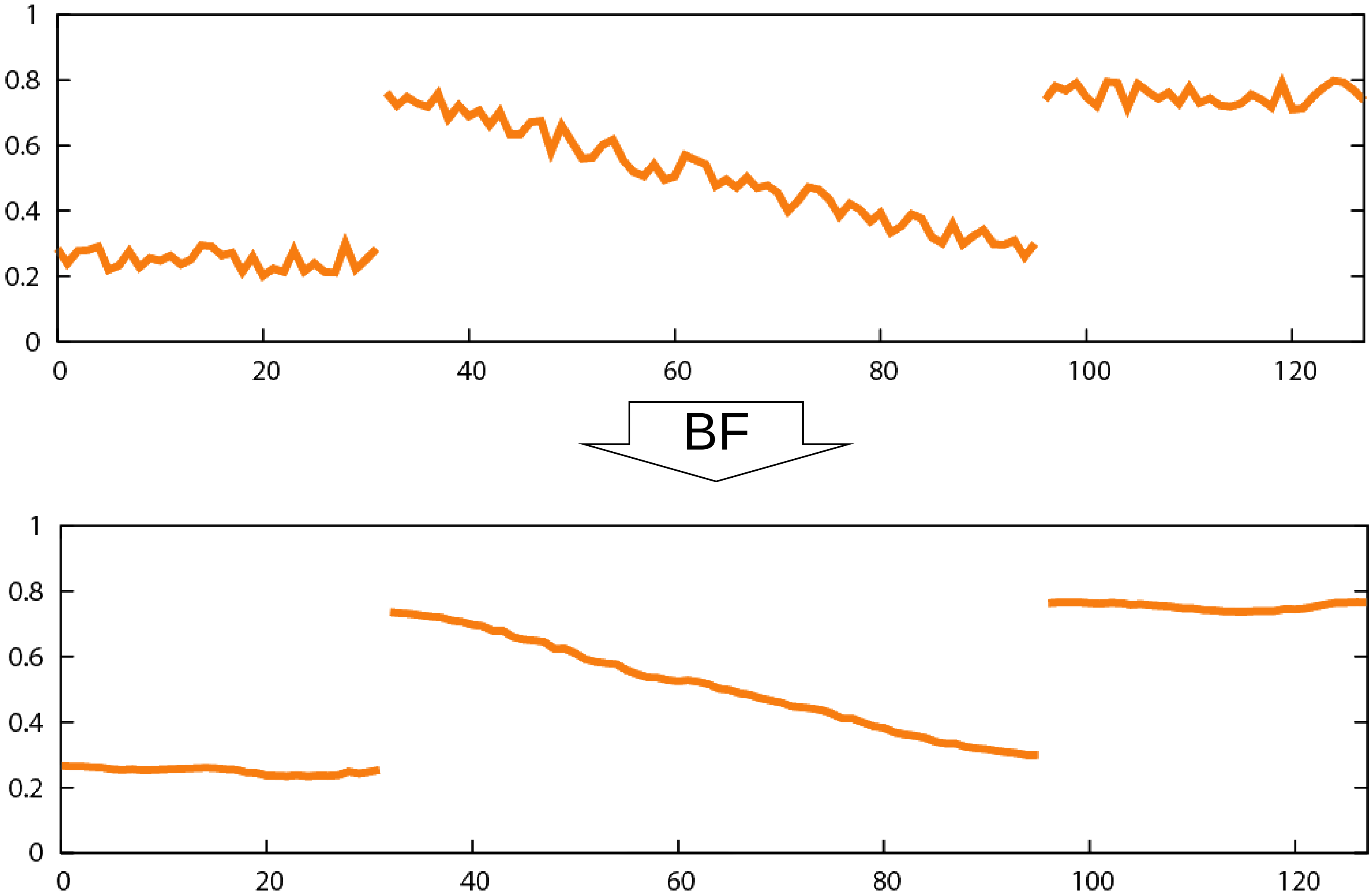
- We will reformulate the bilateral filter
  - Link with **linear filtering**
  - **Fast** and **accurate** algorithm

Questions ?

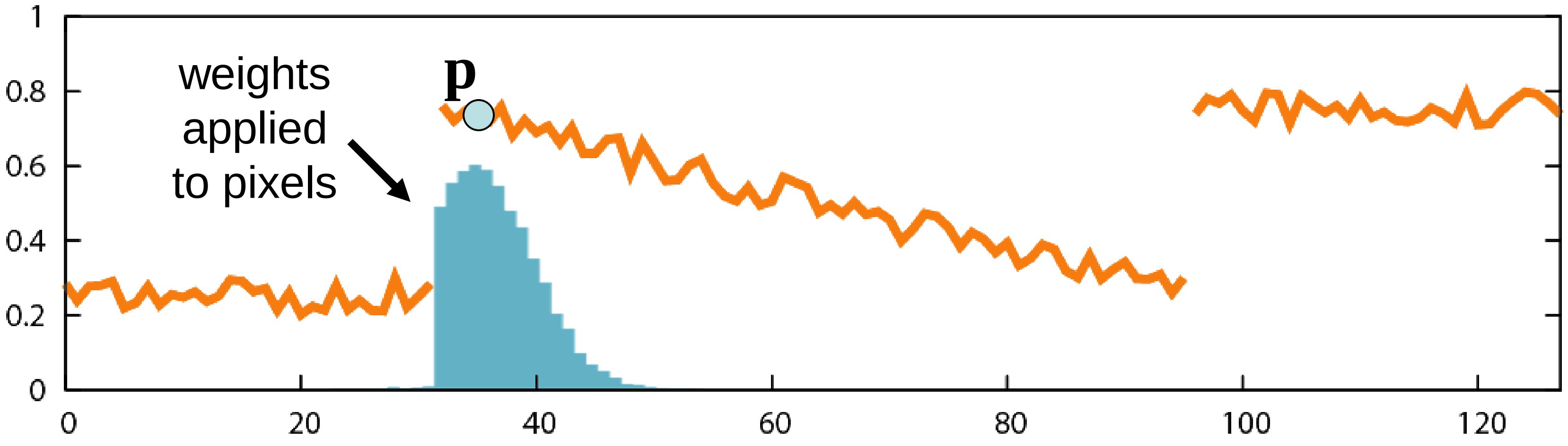
Outline

- Reformulation of the BF
- Fast algorithm to compute the BF
- Practical implementation
- Application and extension
  - Photographic style transfer
  - Bilateral grid

Bilateral Filter on 1D Signal



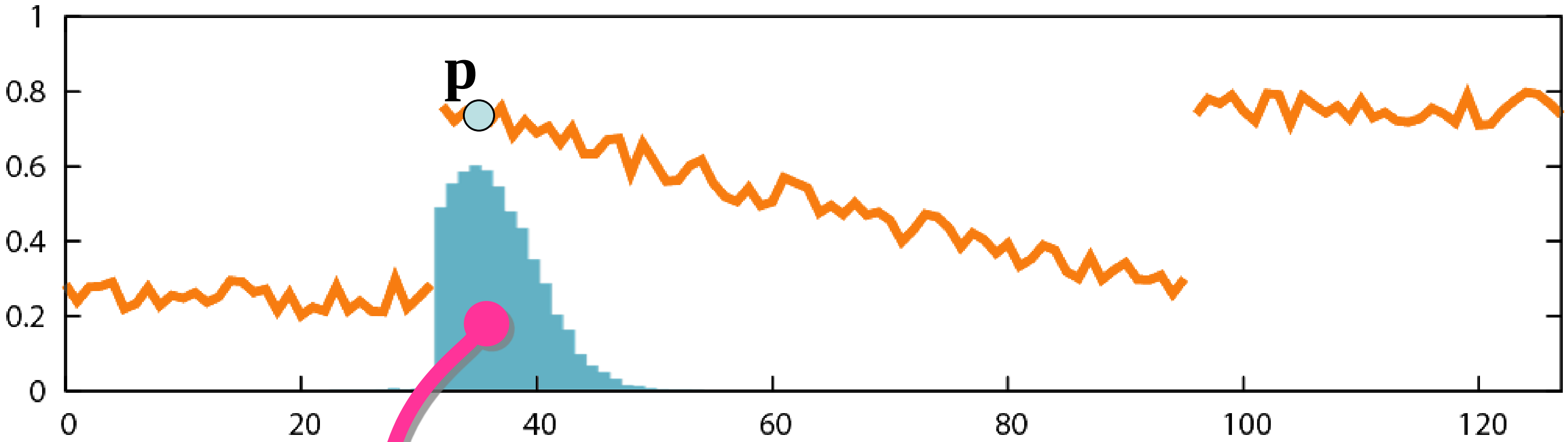
Our Strategy



Reformulate the bilateral filter

- More complex space:
  - Homogeneous intensity
  - Higher-dimensional space
- Simpler expression: mainly a convolution
  - ↳ Leads to a fast algorithm

Link with Linear Filtering
1. Handling the Division



sum of weights

$$I_{\mathbf{p}}^{\mathrm{bf}} = \frac{1}{W_{\mathbf{p}}^{\mathrm{bf}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

Handling the division with a **projective space**.

Formalization: Handling the Division

$$I_{\mathbf{p}}^{\text{bf}} = \frac{1}{W_{\mathbf{p}}^{\text{bf}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\text{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\text{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

$$W_{\mathbf{p}}^{\text{bf}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\text{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\text{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)$$

- Normalizing factor as homogeneous coordinate
  - Multiply both sides by $W_{\mathbf{p}}^{\text{bf}}$

$$\begin{pmatrix} W_{\mathbf{p}}^{\text{bf}}\, I_{\mathbf{p}}^{\text{bf}} \\ W_{\mathbf{p}}^{\text{bf}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\text{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\text{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

Formalization: Handling the Division

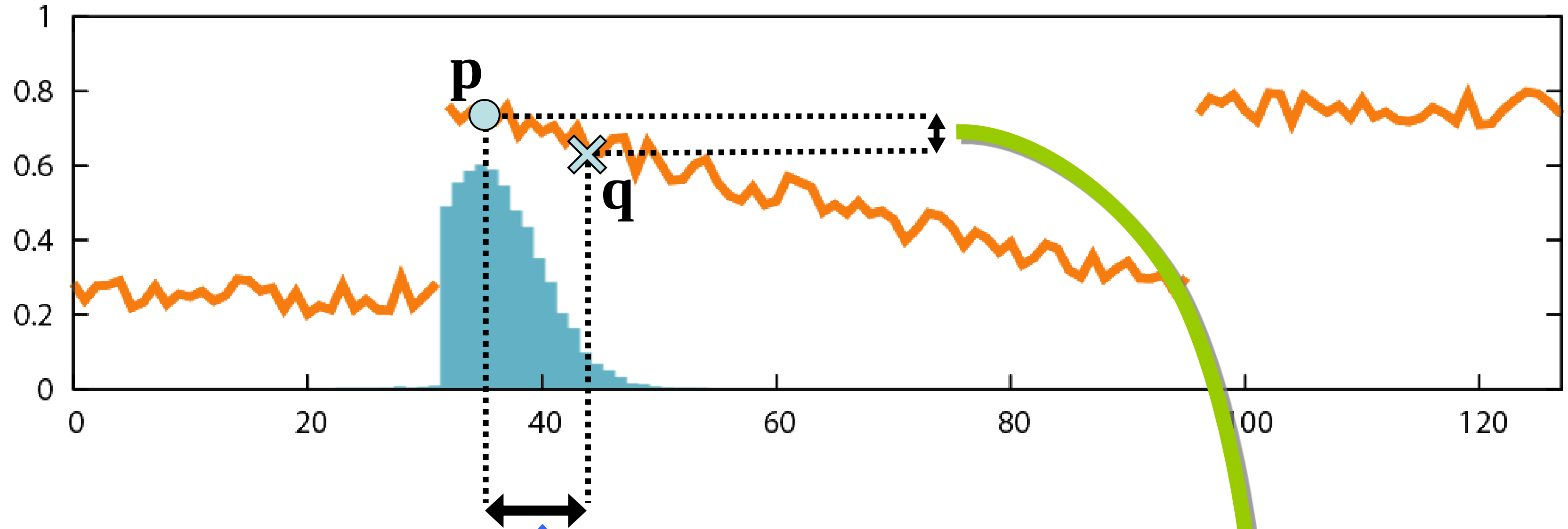
$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}}\ I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) \begin{pmatrix} W_{\mathbf{q}}\ I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix} \text{ with } W_{\mathbf{q}}=1$$

- Similar to homogeneous coordinates in projective space

- Division delayed until the end

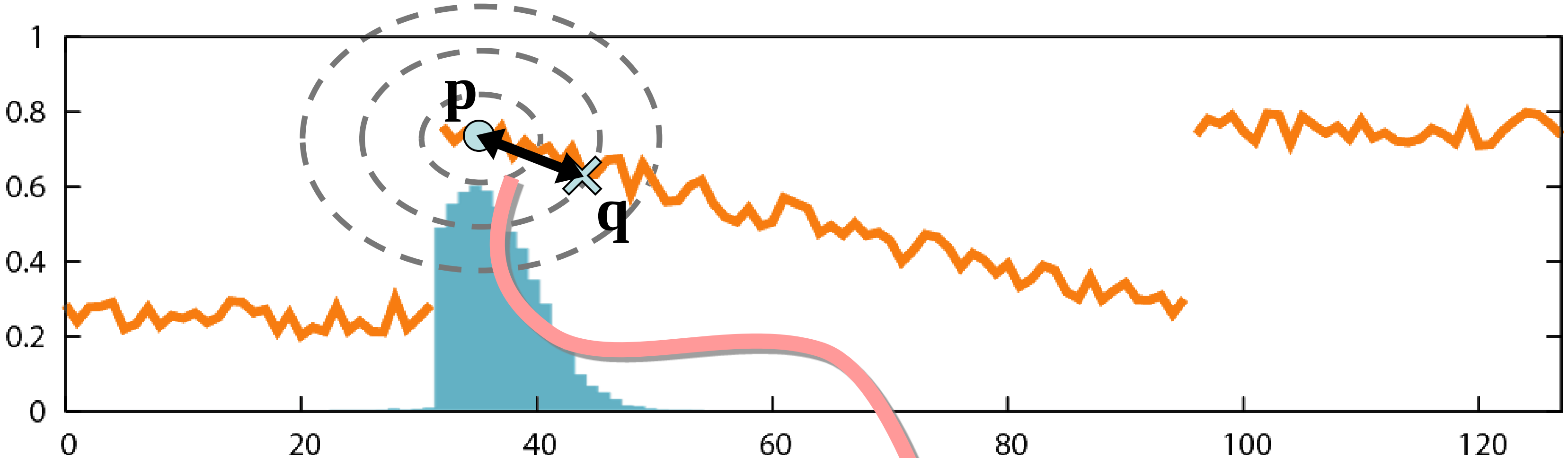
Questions ?

Link with Linear Filtering
2. Introducing a Convolution



$$\begin{pmatrix} W^{\mathrm{bf}}_{\mathbf{p}}\, I^{\mathrm{bf}}_{\mathbf{p}} \\ W^{\mathrm{bf}}_{\mathbf{p}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{\underbrace{\boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}}_{\text{space}}\;\underbrace{\boxed{G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}}_{\text{range}}}_{\text{2D Gaussian}} \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

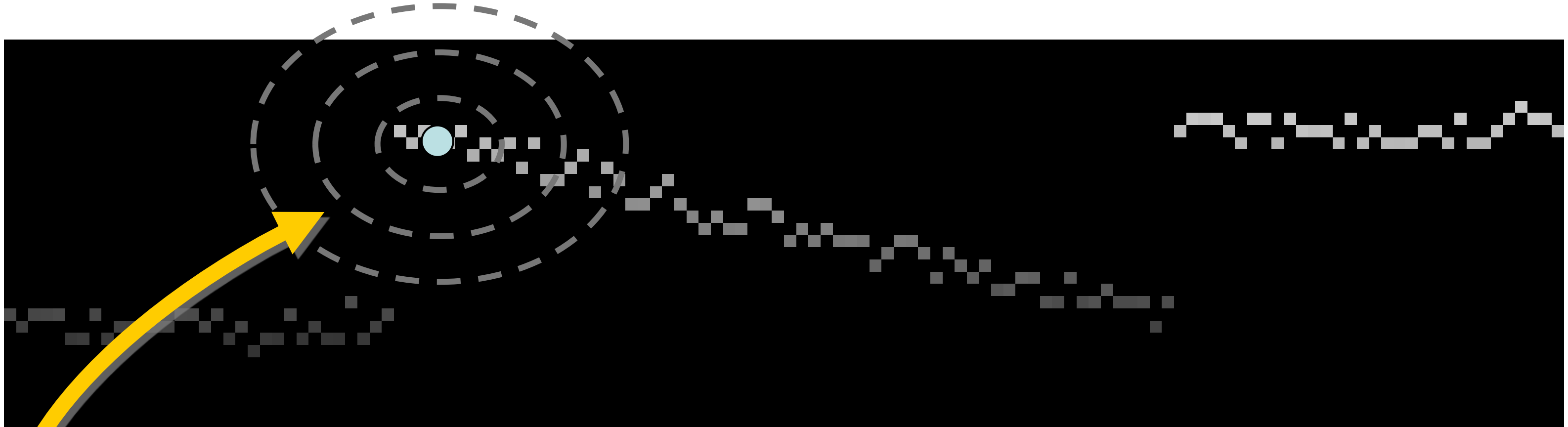
Link with Linear Filtering
2. Introducing a Convolution



$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}}\, I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{\boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}}_{\text{space x range}} \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

Corresponds to a 3D Gaussian on a 2D image.
Result appeared previously in [Barash 02].

Link with Linear Filtering
2. Introducing a Convolution



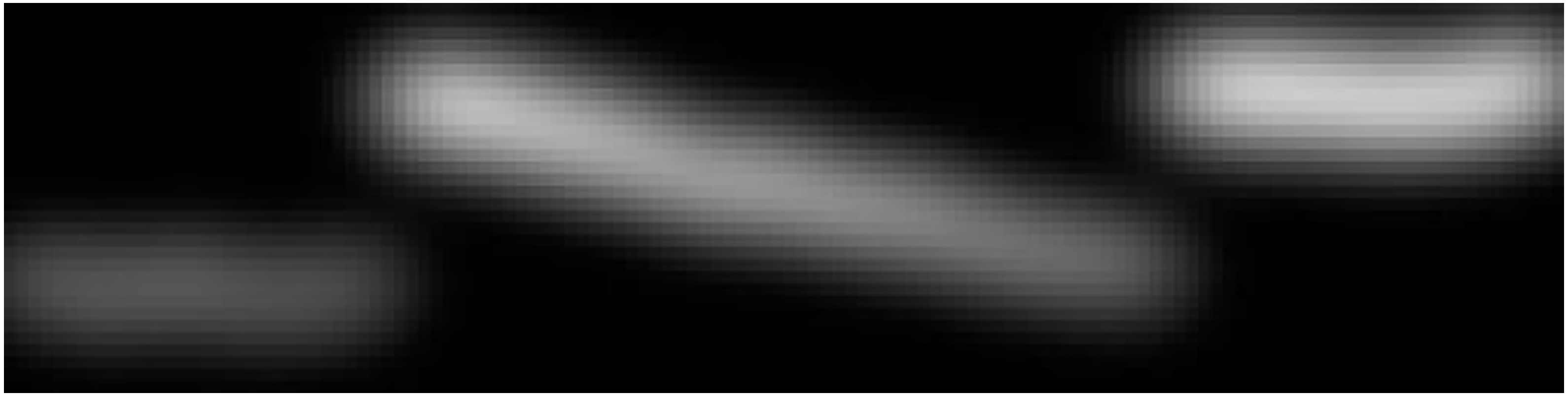
sum all values

$$\begin{pmatrix} W^{\mathrm{bf}}_{\mathbf{p}}\, I^{\mathrm{bf}}_{\mathbf{p}} \\ W^{\mathrm{bf}}_{\mathbf{p}} \end{pmatrix} = \boxed{\sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}}\sum} \; \text{space-range Gaussian} \; \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

sum all values multiplied by kernel ⇨ convolution

Link with Linear Filtering

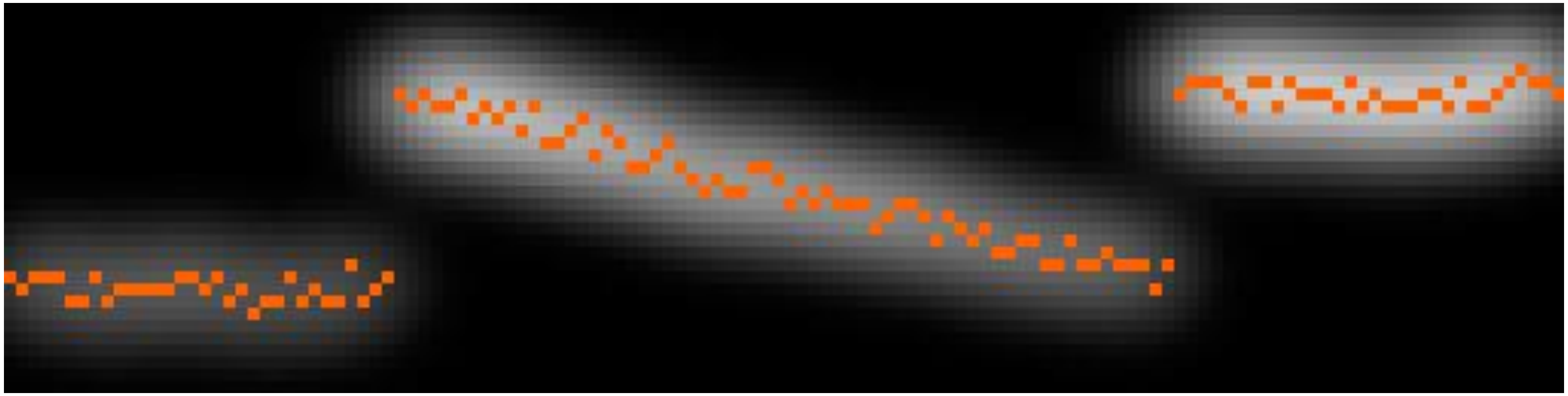
2. Introducing a Convolution



result of the convolution

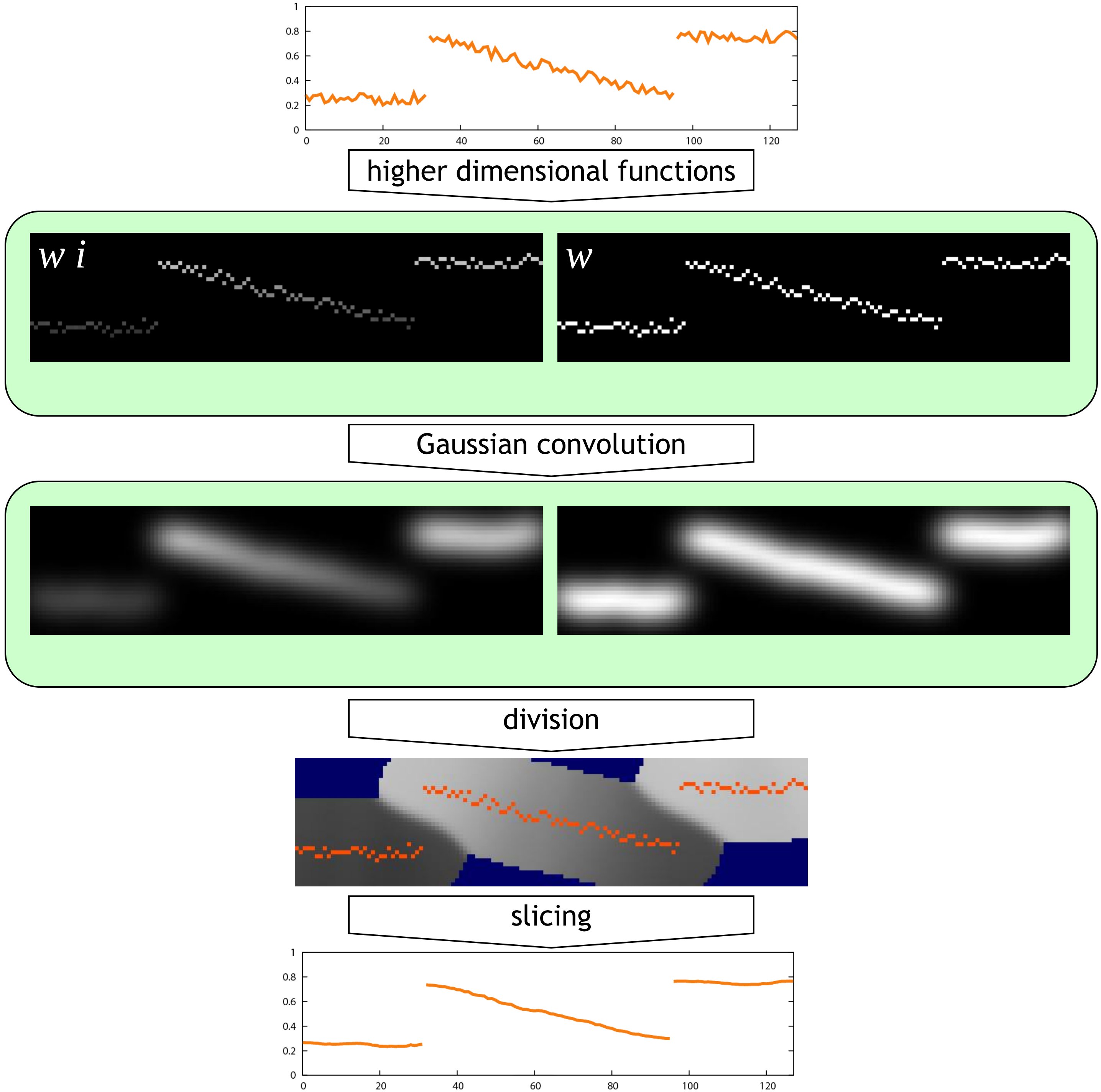
$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}}\, I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \boxed{\text{space-range Gaussian}} \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

Link with Linear Filtering
2. Introducing a Convolution



result of the convolution

$$\begin{pmatrix} \boxed{W^{\mathrm{bf}}_{\mathbf{p}}\, I^{\mathrm{bf}}_{\mathbf{p}}} \\ W^{\mathrm{bf}}_{\mathbf{p}} \end{pmatrix} = \sum\sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$


higher dimensional functions
w i
w
Gaussian convolution
division
slicing

Reformulation: Summary

$$\textbf{linear:} \qquad (w^{\mathrm{bf}}\, i^{\mathrm{bf}}, w^{\mathrm{bf}}) = g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}} \otimes (wi, w)$$

$$\textbf{nonlinear:} \qquad I^{\mathrm{bf}}_{\mathbf{p}} = \frac{w^{\mathrm{bf}}(\mathbf{p}, I_{\mathbf{p}})\; i^{\mathrm{bf}}(\mathbf{p}, I_{\mathbf{p}})}{w^{\mathrm{bf}}(\mathbf{p}, I_{\mathbf{p}})}$$

1. Convolution in higher dimension
   - expensive but well understood (linear, FFT, etc)
2. Division and slicing
   - nonlinear but simple and pixel-wise

Exact reformulation

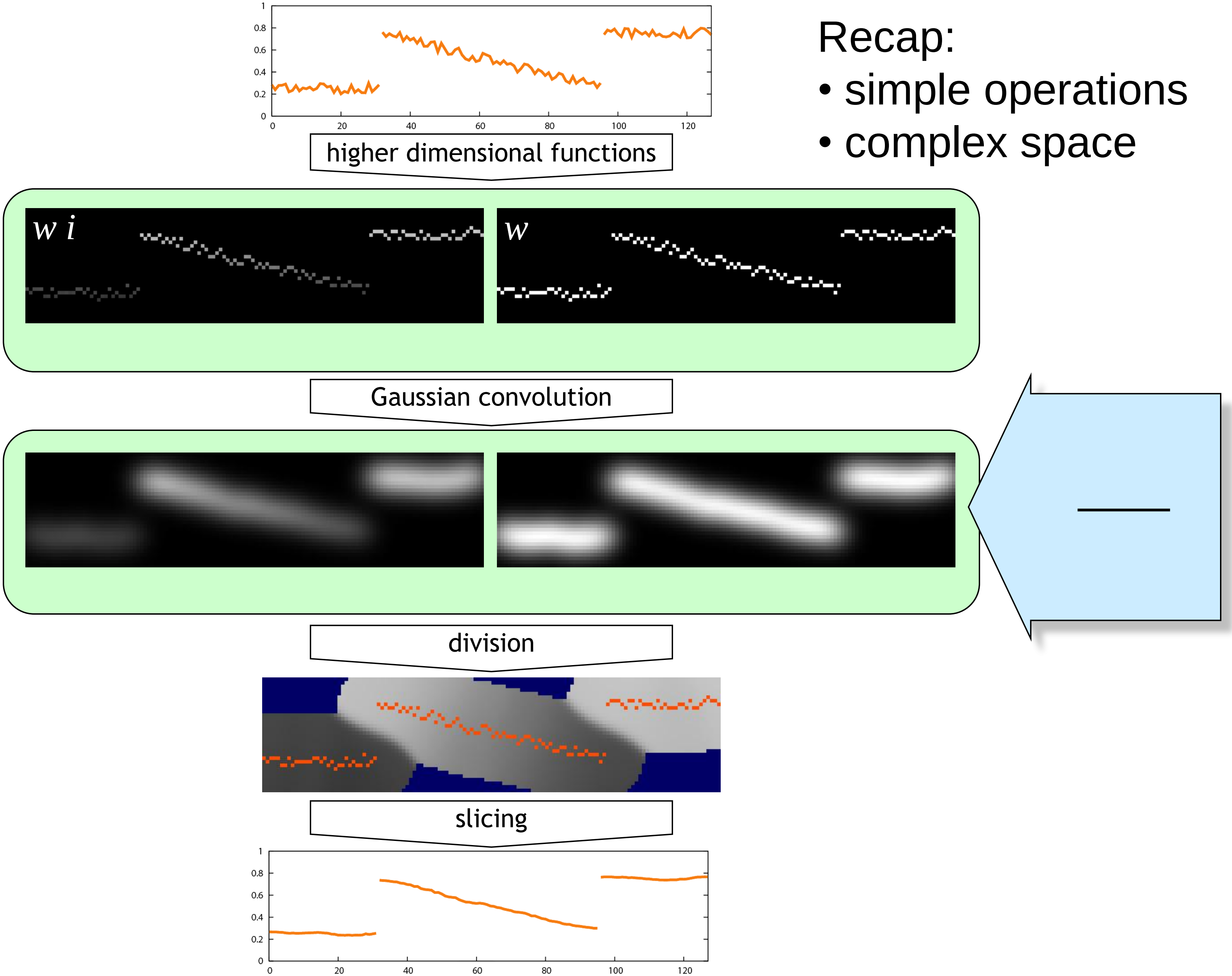
Questions ?

Outline

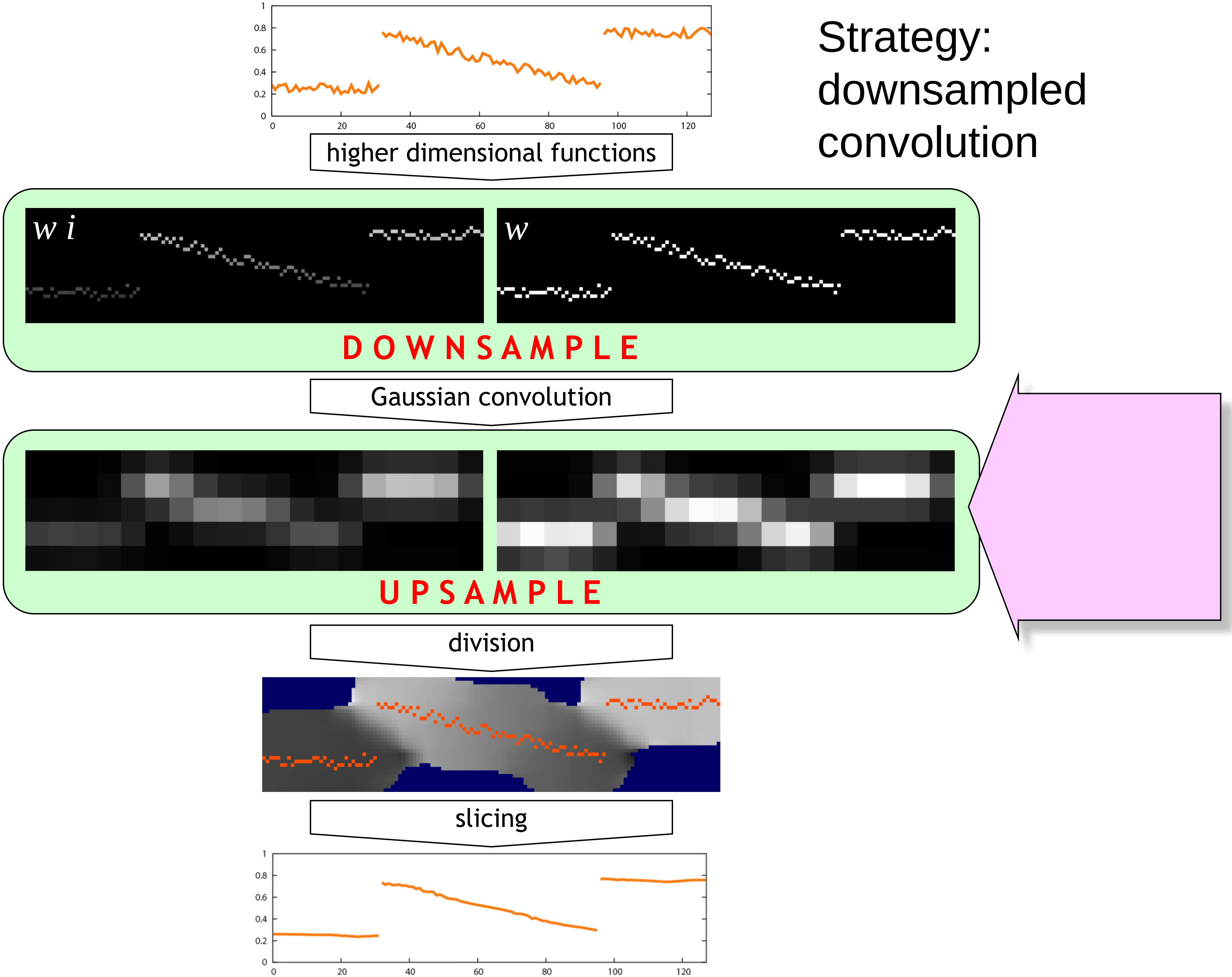
- Reformulation of the BF
- Fast algorithm to compute the BF
- Practical implementation
- Application and extension
  - Photographic style transfer
  - Bilateral grid

Recap:

- simple operations
- complex space

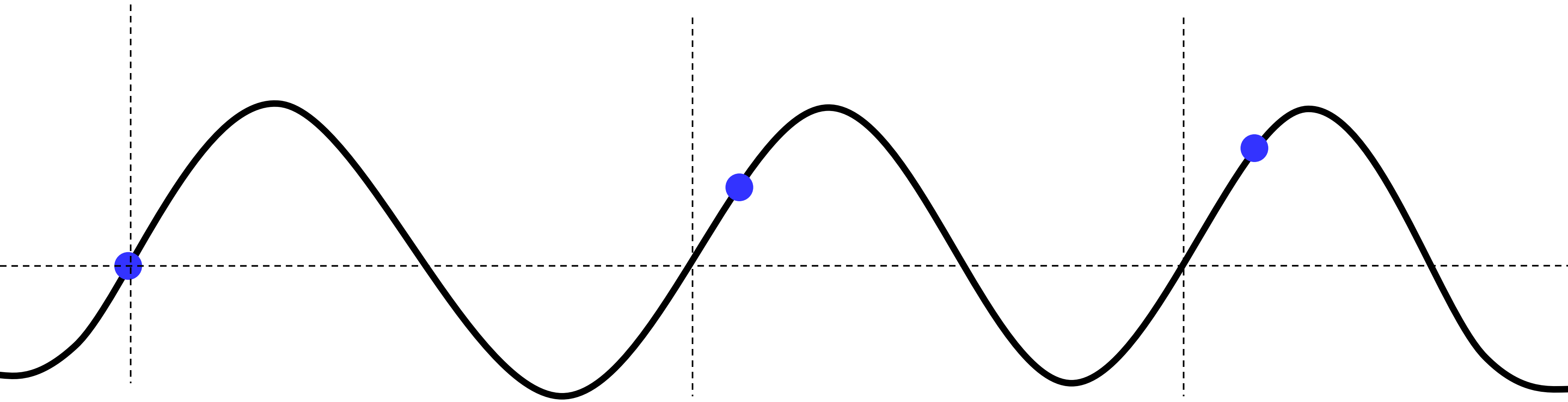


Strategy: downsampled convolution



Sampling Theorem

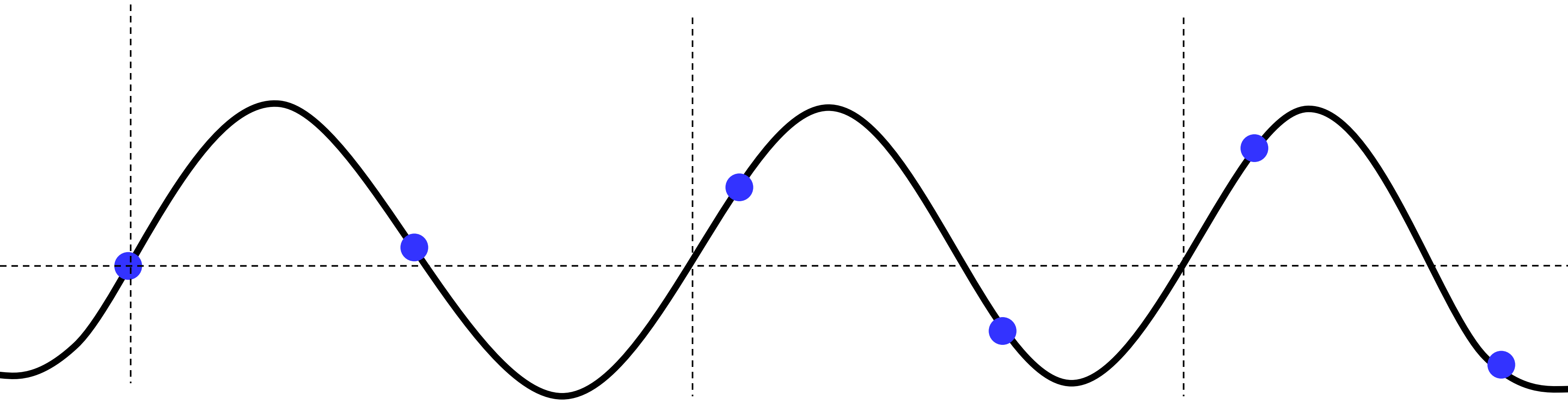
- Sampling a signal at a least twice its smallest wavelength is enough.



Not enough

Sampling Theorem

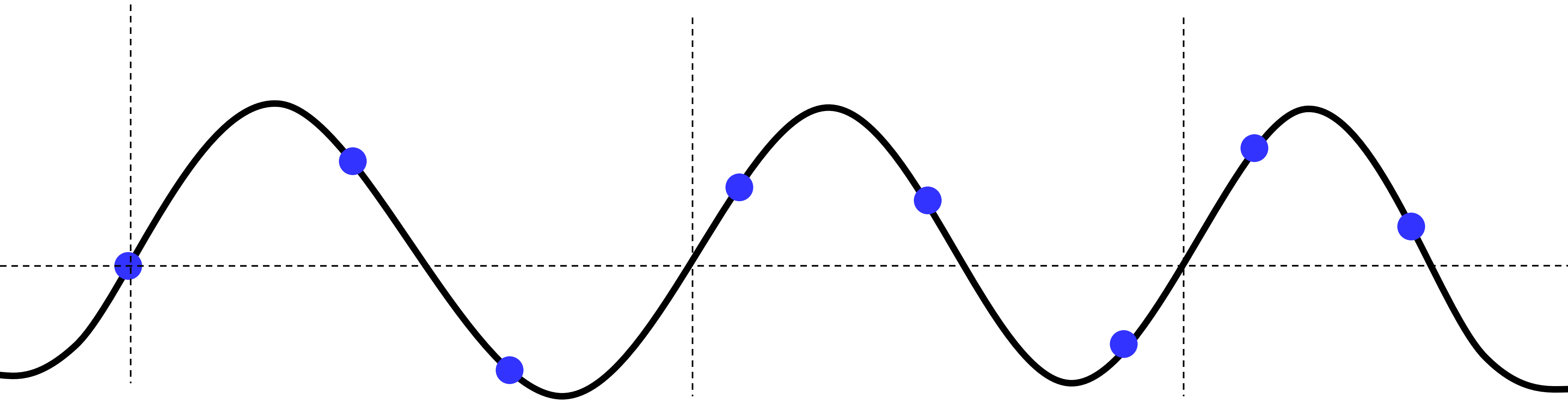
- Sampling a signal at a least twice its smallest wavelength is enough.



Not enough

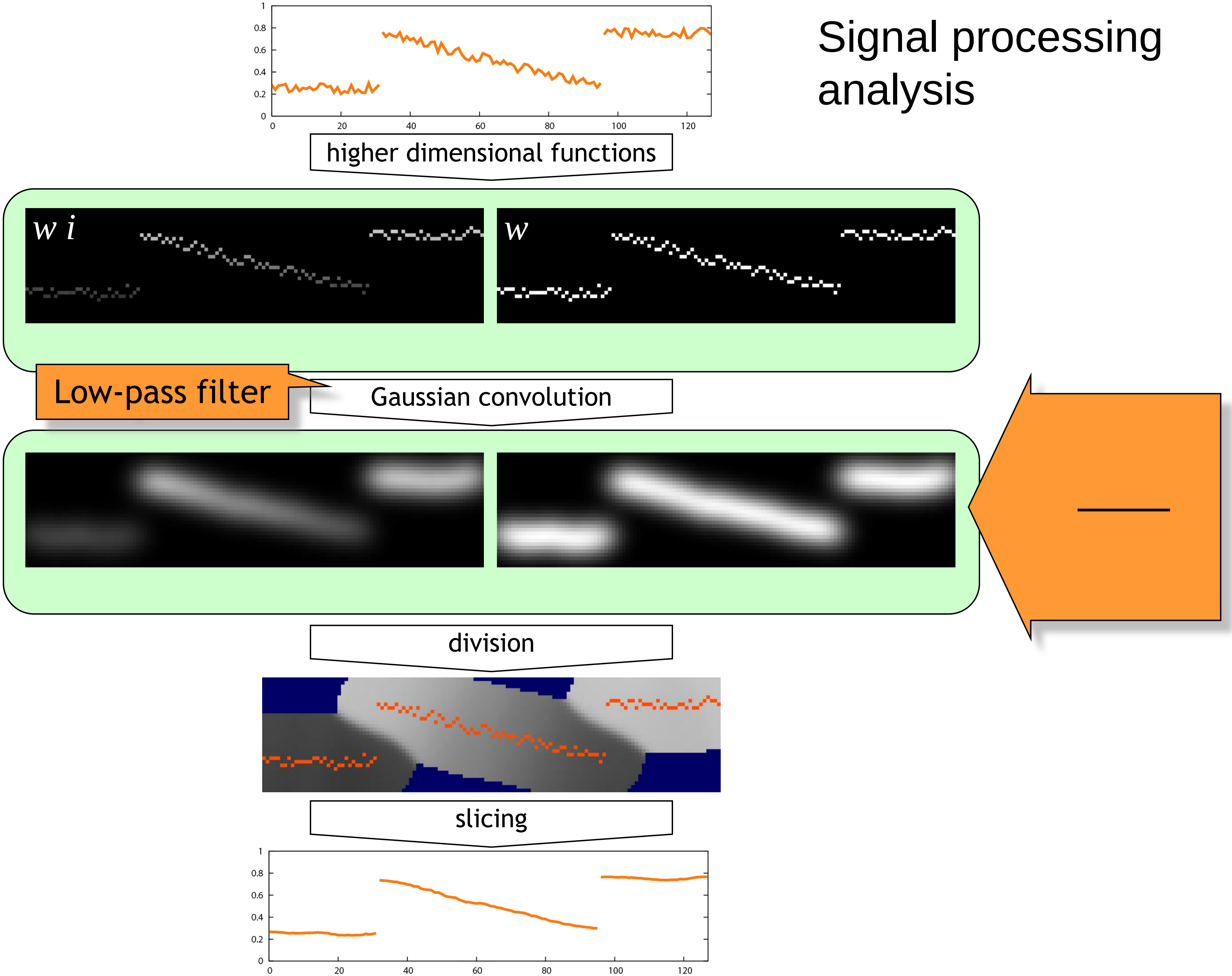
Sampling Theorem

- Sampling a signal at a least twice its smallest wavelength is enough.

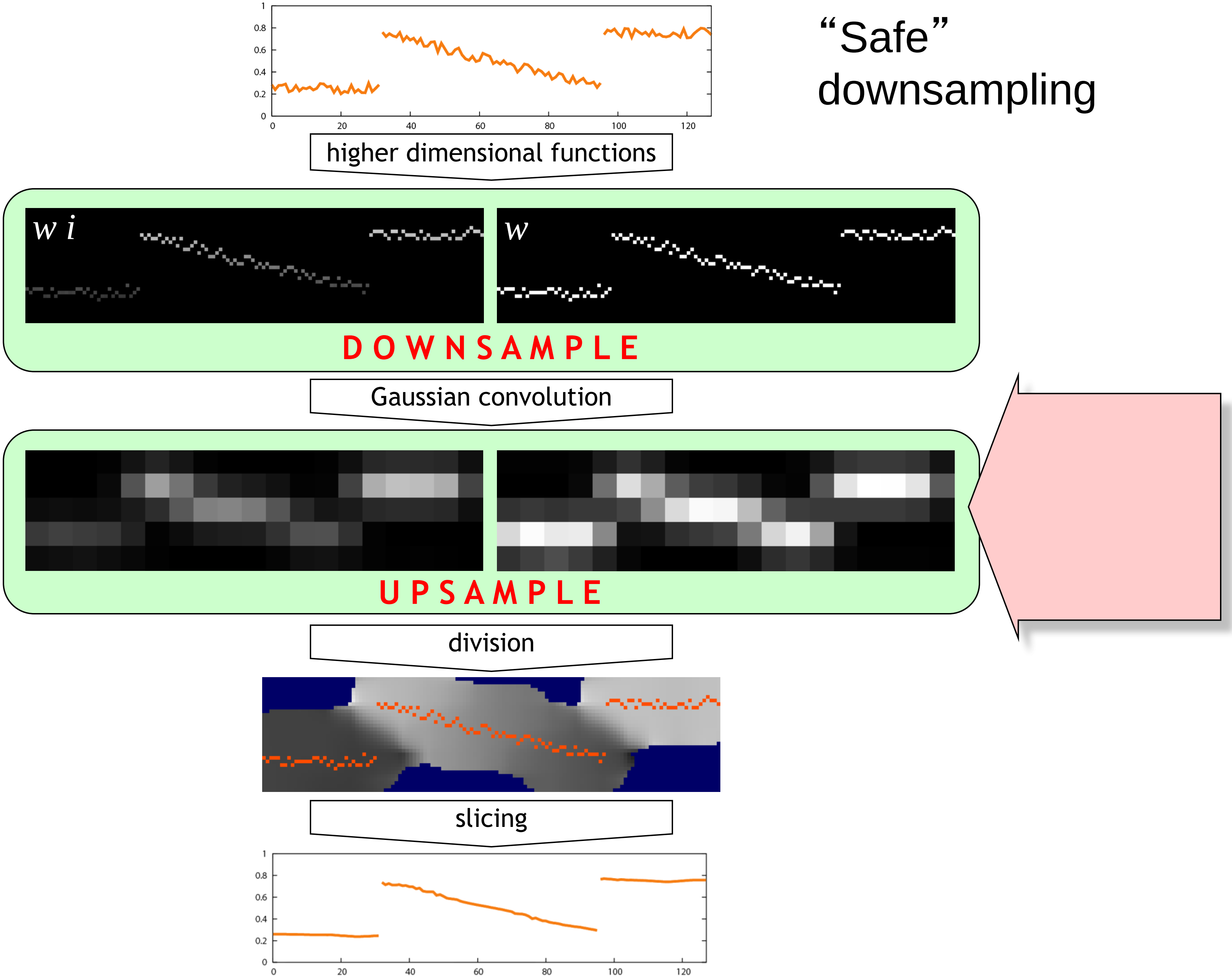


Enough

Signal processing analysis



“Safe” downsampling



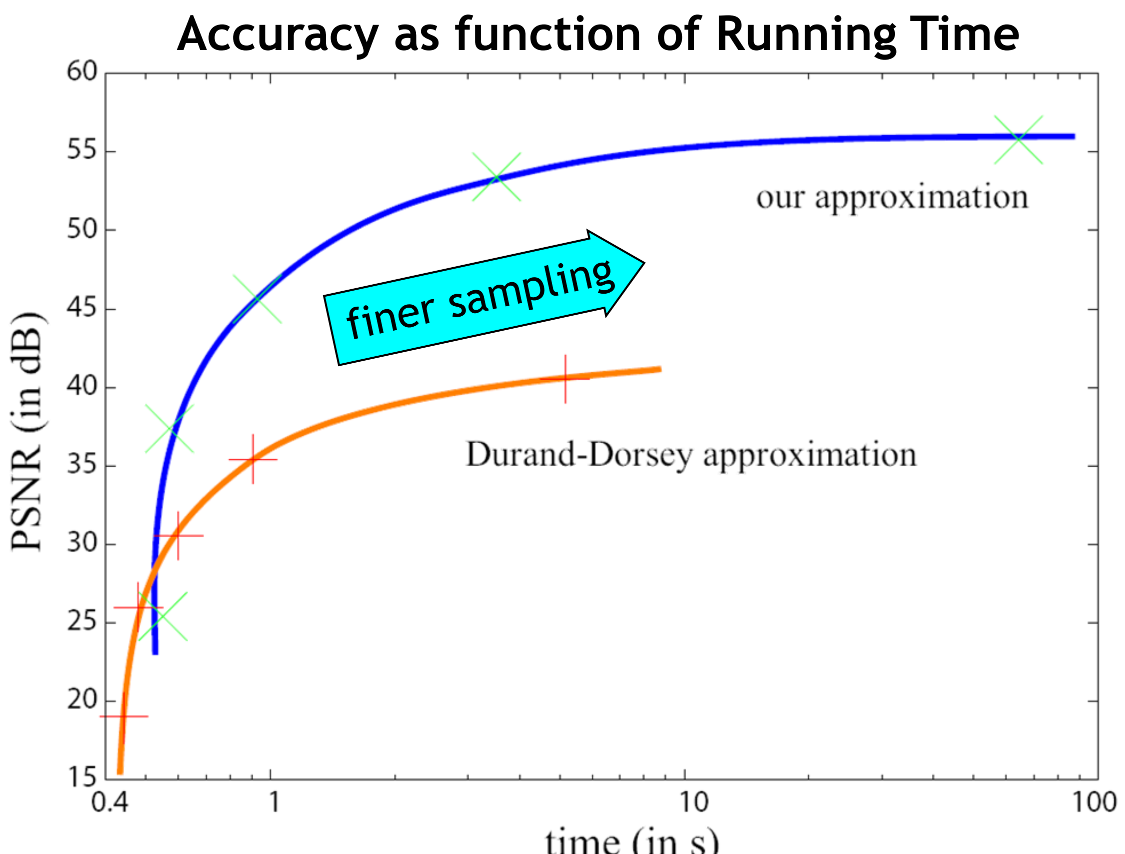
Fast Convolution by Downsampling

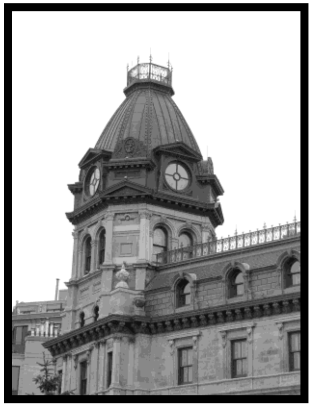
- Downsampling cuts frequencies above Nyquist limit (half the sampling rate)
  - Less data to process
  - But introduces error
- Evaluation of the approximation
- Efficient implementation

Accuracy versus Running Time

≈1 second instead of **several minutes**

- Finer sampling increases accuracy.
- More precise than previous work.





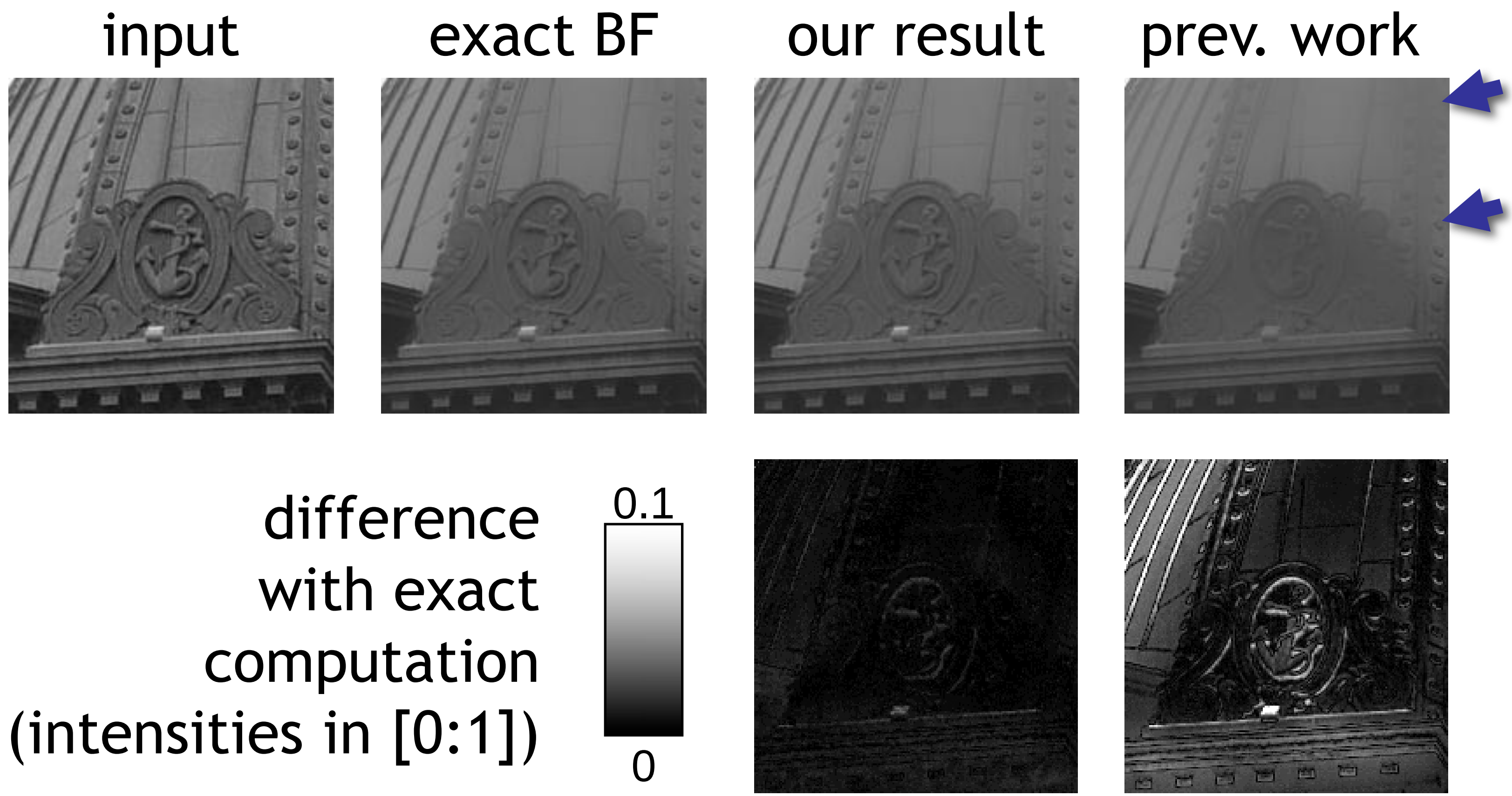
Digital photograph 1200×1600

Brute-force bilateral filter takes over 10 minutes.

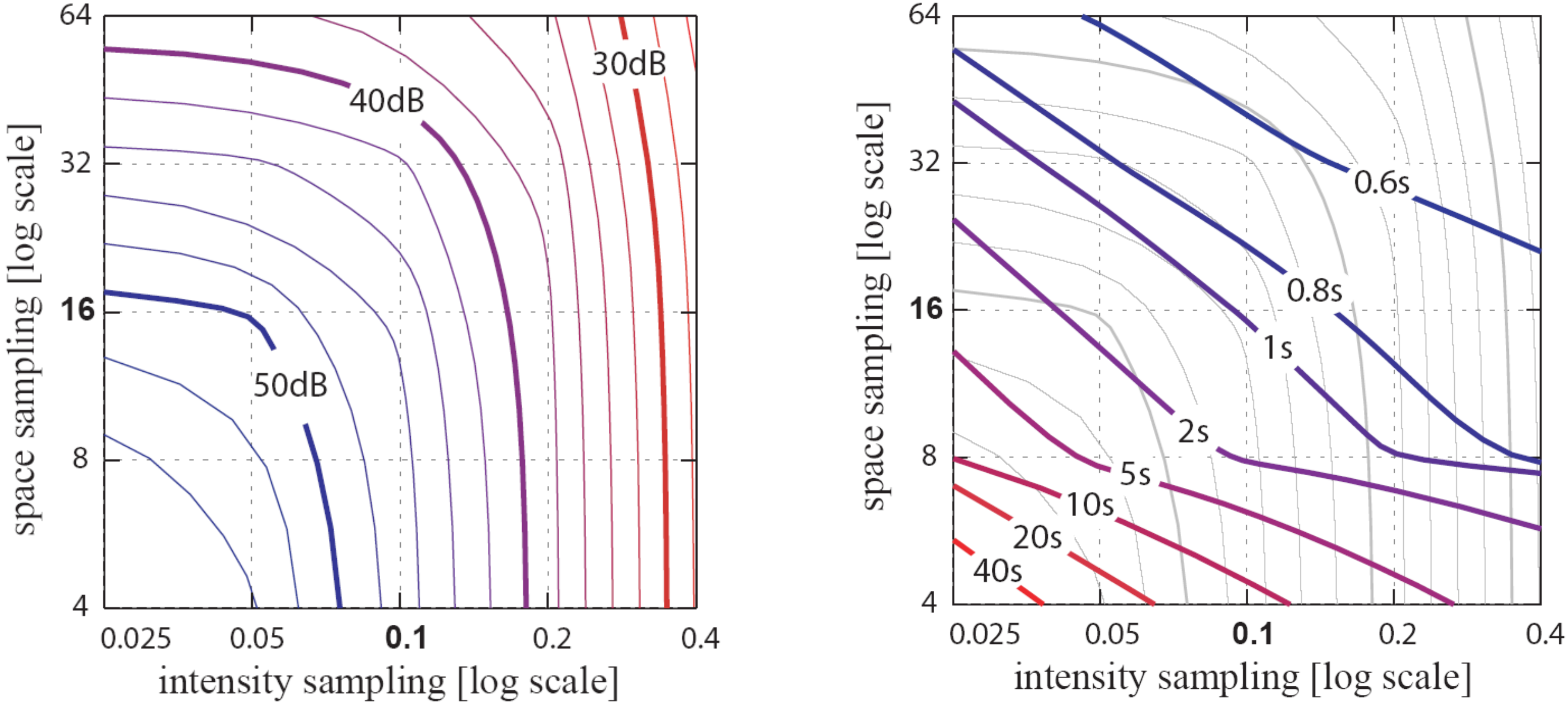
Visual Results

1200 × 1600

- Comparison with previous work [Durand 02]
  - running time = 1s for both techniques



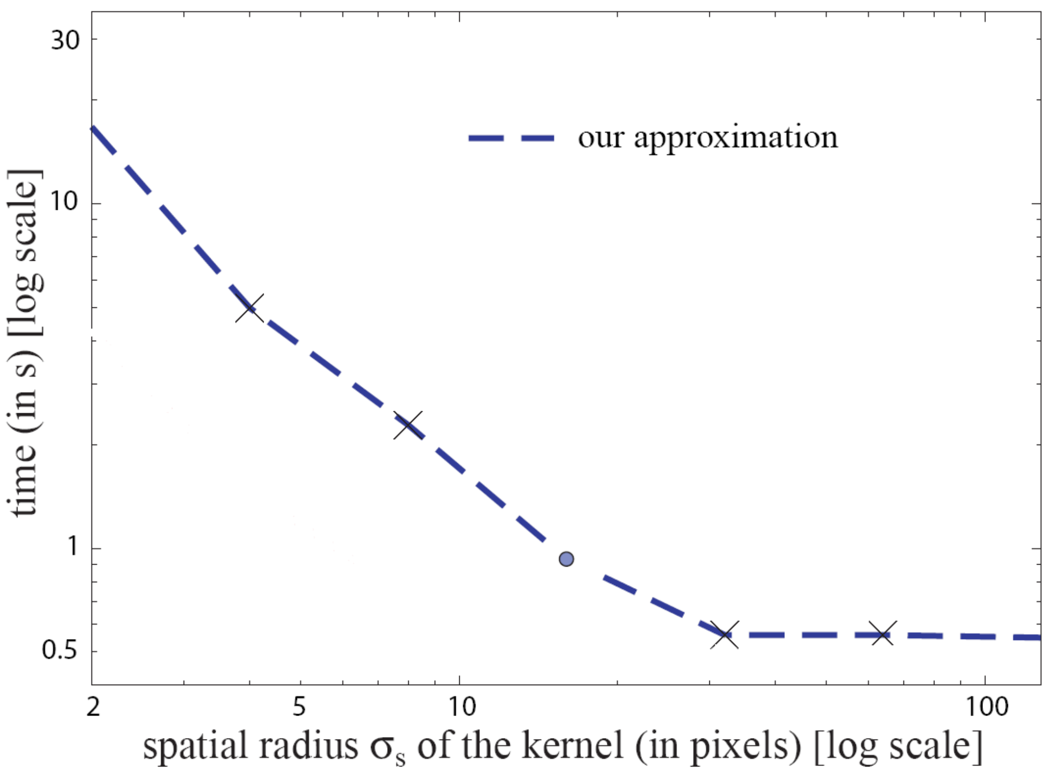
More on Accuracy and Running Times

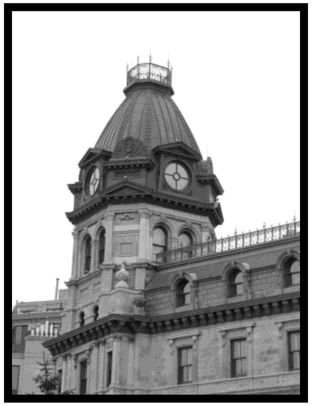


Kernel Size

- **Larger kernels are faster** because we downsample more.
  - ↳ Useful for photography.

Running Time as a function of Kernel Size





Digital photograph
1200 × 1600

Brute-force bilateral filter takes over 10 minutes.

Questions ?

Outline

- Reformulation of the BF
- Fast algorithm to compute the BF
- Practical implementation
- Application and extension
  - Photographic style transfer
  - Bilateral grid

Efficient Implementation

- Never build the full resolution 3D space
  - Bin pixels on the fly
  - Interpolate on the fly

- Separable Gaussian kernel

- 5-tap approximation

Sampling Rate

- 1 sample every sigma
  - Kernel parameters are all equal to 1 sample
  - 5-tap approximation is sufficient

1 – 4 – 6 – 4 – 1

Fast Bilateral Filter

input:	image I
	Gaussian parameters σ_s and σ_r
	sampling rates s_s and s_r
output:	filtered image I^b

1. Initialize all $w_\downarrow i_\downarrow$ and $w_\downarrow$ values to 0.

Initialize the 3D grid to 0.

FAST BILATERAL FILTER

input: image I
Gaussian parameters σ_{s} and σ_{r}
sampling rates s_{s} and s_{r}

output: filtered image I^{b}

1. Initialize all $w_{\downarrow}\, i_{\downarrow}$ and $w_{\downarrow}$ values to 0. **Initialize the 3D grid to 0.**

2. Compute the minimum intensity value:

Useful later to save space. $$I_{\min} \quad \leftarrow \quad \min_{(X,Y)\in\mathcal{S}} I(X,Y)$$

3. For each pixel $(X, Y) \in \mathcal{S}$ with an intensity $I(X, Y) \in \mathcal{R}$ **Look at each pixel.**

(a) Compute the homogeneous vector (wi, w):

Create the data. $$(wi, w) \quad \leftarrow \quad \big(I(X,Y), 1\big)$$

3. For each pixel $(X, Y) \in \mathcal{S}$ with an intensity $I(X, Y) \in \mathcal{R}$ **Look at each pixel.**

   (a) Compute the homogeneous vector (wi, w):

   Create the data.
   $$(wi, w) \quad \leftarrow \quad \big(I(X, Y), 1\big)$$

   (b) Compute the downsampled coordinates (with $[\,\cdot\,]$ the rounding operator)

   Compute the grid location.
   $$(x, y, \zeta) \quad \leftarrow \quad \left(\left[\frac{X}{s_{\mathrm{s}}} \right], \left[\frac{Y}{s_{\mathrm{s}}} \right], \left[\frac{I(X, Y) - I_{\min}}{s_{\mathrm{r}}} \right] \right)$$

3. For each pixel $(X, Y) \in \mathcal{S}$ with an intensity $I(X, Y) \in \mathcal{R}$ **Look at each pixel.**

(a) Compute the homogeneous vector (wi, w):

Create the data. $$(wi, w) \quad \leftarrow \quad (I(X, Y), 1)$$

(b) Compute the downsampled coordinates (with $[\cdot]$ the rounding operator)

Compute the grid location. $$(x, y, \zeta) \quad \leftarrow \quad \left(\left[\frac{X}{s_\mathrm{s}} \right], \left[\frac{Y}{s_\mathrm{s}} \right], \left[\frac{I(X, Y) - I_{\min}}{s_\mathrm{r}} \right] \right)$$

(c) Update the downsampled $\mathcal{S} \times \mathcal{R}$ space

Update the grid. $$\begin{pmatrix} w_\downarrow \, i_\downarrow(x, y, \zeta) \\ w_\downarrow(x, y, \zeta) \end{pmatrix} \quad \leftarrow \quad \begin{pmatrix} w_\downarrow \, i_\downarrow(x, y, \zeta) \\ w_\downarrow(x, y, \zeta) \end{pmatrix} \quad + \quad \begin{pmatrix} wi \\ w \end{pmatrix}$$

4. Convolve $(w_\downarrow\, i_\downarrow, w_\downarrow)$ with a 3D Gaussian g whose parameters are σ_s/s_s and σ_r/s_r

$$(w^b_\downarrow\, i^b_\downarrow, w^b_\downarrow) \quad \leftarrow \quad (w_\downarrow\, i_\downarrow, w_\downarrow) \otimes g$$

Convolve with 1 – 4 – 6 – 4 – 1 along each axis.
3 for loops needed, one for each axis.

In 3D, 15 samples (3 times 5) considered instead of 125 (5^3) for a full convolution.
Same result!

5. For each pixel $(X, Y) \in \mathcal{S}$ with an intensity $I(X, Y) \in \mathcal{R}$ **Look at each pixel.**

(a) Tri-linearly interpolate the functions $w_\downarrow^{\mathrm{b}}\, i_\downarrow^{\mathrm{b}}$ and $w_\downarrow^{\mathrm{b}}$ to obtain $W^{\mathrm{b}}\, I^{\mathrm{b}}$ and W^{b}:

$$\begin{aligned} W^{\mathrm{b}}\, I^{\mathrm{b}}(X, Y) &\leftarrow \operatorname{interpolate}\left(w_\downarrow^{\mathrm{b}}\, i_\downarrow^{\mathrm{b}}, \frac{X}{s_{\mathrm{s}}}, \frac{Y}{s_{\mathrm{s}}}, \frac{I(X, Y)}{s_{\mathrm{r}}}\right) \\ W^{\mathrm{b}}(X, Y) &\leftarrow \operatorname{interpolate}\left(w_\downarrow^{\mathrm{b}}, \frac{X}{s_{\mathrm{s}}}, \frac{Y}{s_{\mathrm{s}}}, \frac{I(X, Y)}{s_{\mathrm{r}}}\right) \end{aligned}$$

5. For each pixel $(X, Y) \in \mathcal{S}$ with an intensity $I(X, Y) \in \mathcal{R}$ **Look at each pixel.**

(a) Tri-linearly interpolate the functions $w_{\downarrow}^{\mathrm{b}}\, i_{\downarrow}^{\mathrm{b}}$ and $w_{\downarrow}^{\mathrm{b}}$ to obtain $W^{\mathrm{b}}\, I^{\mathrm{b}}$ and W^{b}:

$$W^{\mathrm{b}}\, I^{\mathrm{b}}(X, Y) \quad \leftarrow \quad \text{interpolate}\left(w_{\downarrow}^{\mathrm{b}}\, i_{\downarrow}^{\mathrm{b}}, \frac{X}{s_{\mathrm{s}}}, \frac{Y}{s_{\mathrm{s}}}, \frac{I(X, Y)}{s_{\mathrm{r}}}\right)$$

$$W^{\mathrm{b}}(X, Y) \quad \leftarrow \quad \text{interpolate}\left(w_{\downarrow}^{\mathrm{b}}, \frac{X}{s_{\mathrm{s}}}, \frac{Y}{s_{\mathrm{s}}}, \frac{I(X, Y)}{s_{\mathrm{r}}}\right)$$

(b) Normalize the result

$$I^{\mathrm{b}}(X, Y) \quad \leftarrow \quad \frac{W^{\mathrm{b}}\, I^{\mathrm{b}}(X, Y)}{W^{\mathrm{b}}(X, Y)}$$

Comments

- Every sample is processed even if empty
  - The grid is coarse and fairly dense in 3D.
    - e.g. parameters (16,0.1): 256 pixels for 10 bins

      convolution spans 5 bins
      at least 50% occupancy

  - Simple data structure, simple sweep ⇨ fast

  - More sophisticated approaches needed
    for higher dimensional cases
    [Adams et al. 09,10] [Gastal and Oliveira 11,12]

Complexity

- There is no nested loops over the whole set of samples
  - At most: “for each sample, for 5 samples”

$$\mathcal{O}\left(|\mathcal{S}| + \frac{|\mathcal{S}|}{s_{\mathrm{s}}^2} \frac{|\mathcal{R}|}{s_{\mathrm{r}}} \right)$$

Creation + slicing
“for each pixel”

Convolution
“for each grid sample”

References

- **Short version:** A Fast Approximation of the Bilateral Filter using a Signal Processing Approach*. Sylvain Paris and Frédo Durand.* ECCV 2006

- **Long version:** A Fast Approximation of the Bilateral Filter using a Signal Processing Approach. *Sylvain Paris and Frédo Durand.* International Journal of Computer Vision, 2009

- **Different presentation, more applications:** Real-time Edge-Aware Image Processing with the Bilateral Grid. *Jiawen Chen, Sylvain Paris, and Frédo Durand*. SIGGRAPH 2007

- **Survey:** Bilateral Filtering: Theory and Applications. *Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand.* Foundations and Trends in Computer Graphics and Vision, 2009

- **Data, code, etc:** http://people.csail.mit.edu/sparis/bf/