Computer Graphics - Bilateral & Nonlocal Mean Filtering

Junjie Cao @ DLUT

Spring 2016

http://jjcao.github.io/ComputerGraphics/

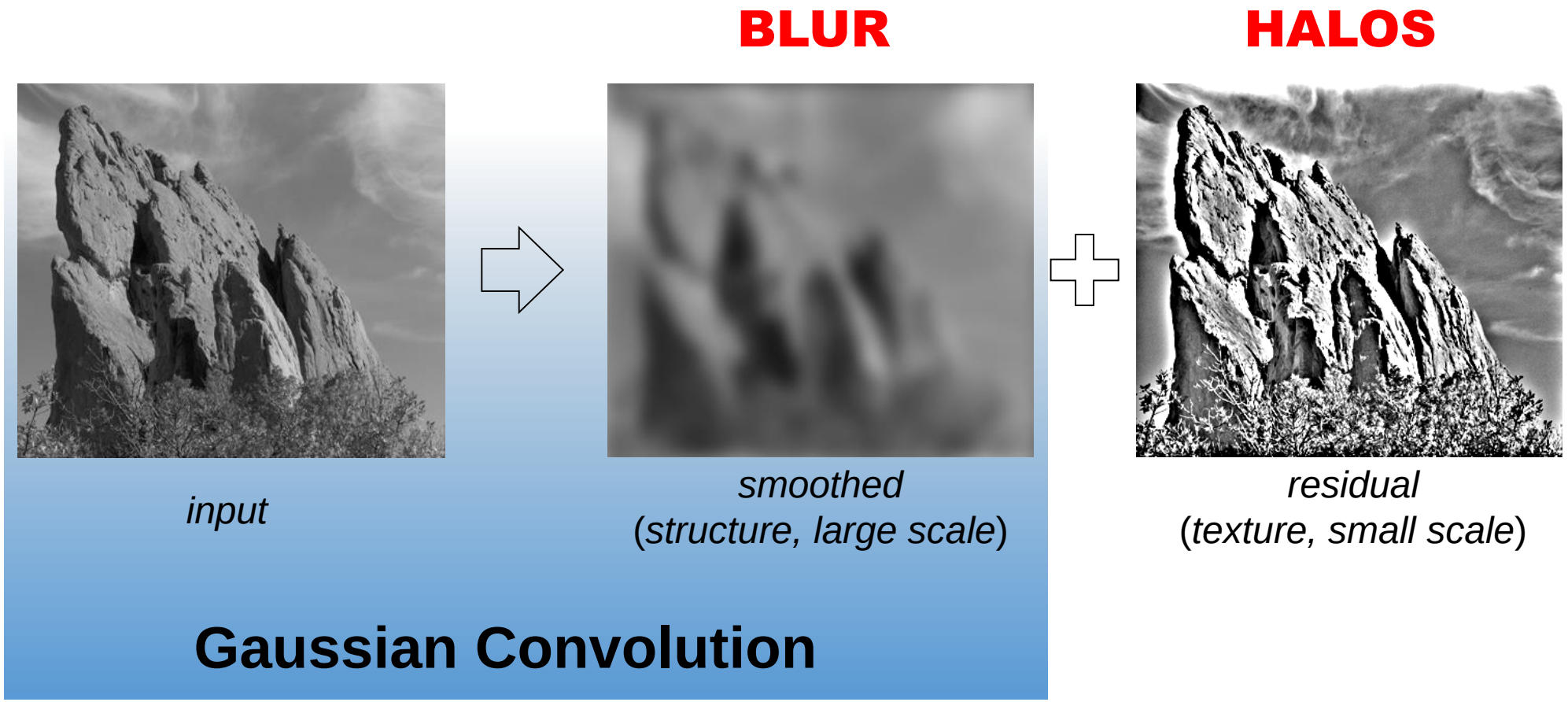
Bilateral filtering

Introduction

Goal: Image Smoothing

Split an image into:

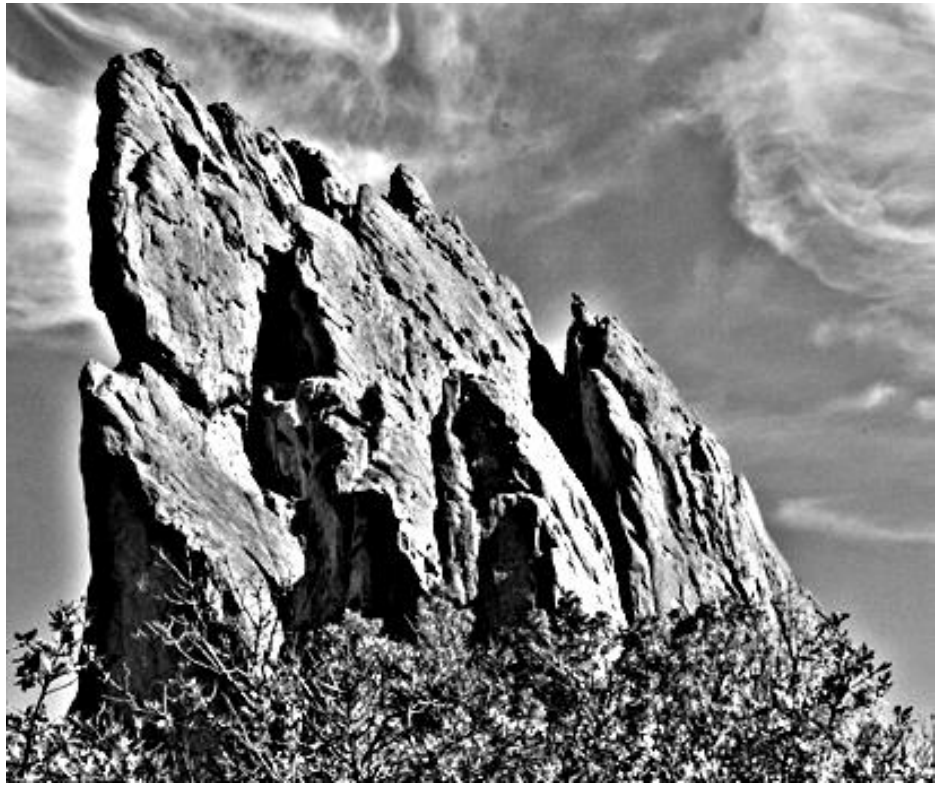
- large-scale features, structure
- small-scale features, texture



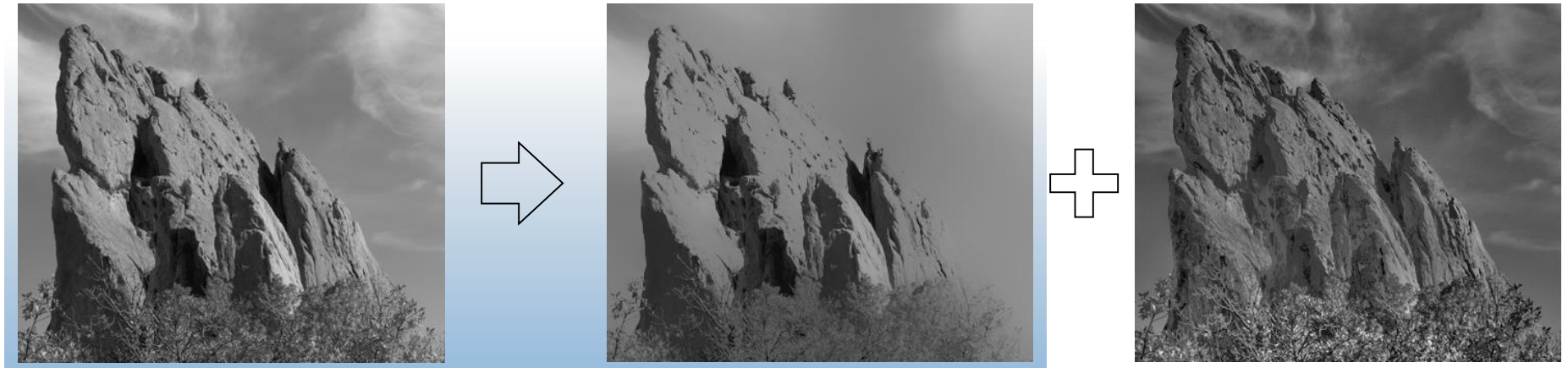
Impact of Blur and Halos

- 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) vs Naïve Approach: Gaussian Blur (with blur & halos)

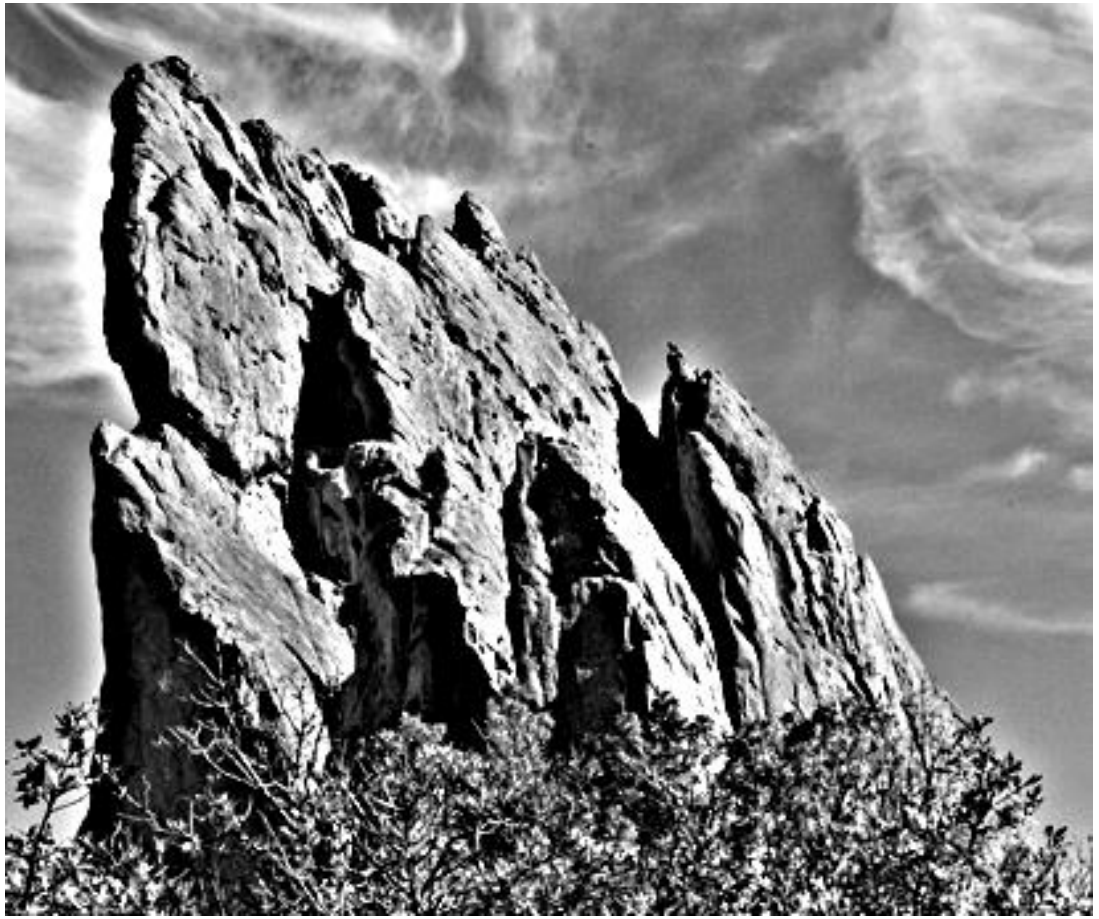


input

smoothed
(*structure, large scale*)

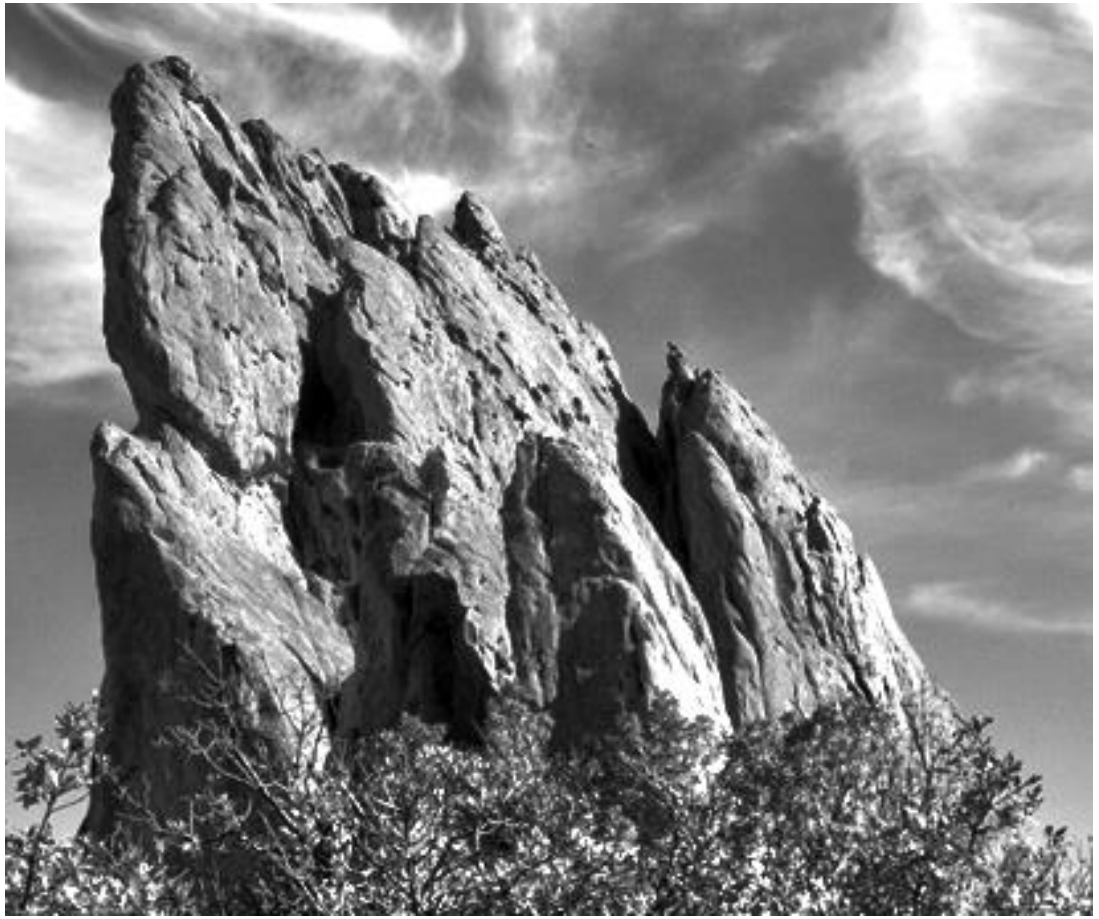
residual
(*texture, small scale*)

edge-preserving: Bilateral Filter



increasing texture
with Gaussian convolution

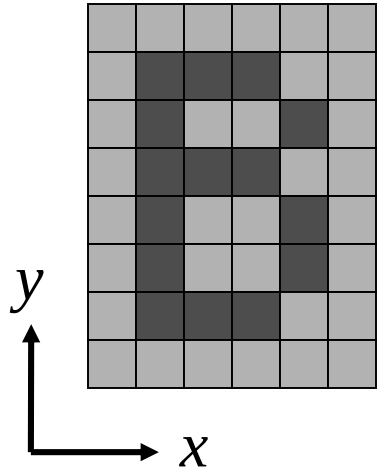
HALOS



increasing texture
with bilateral filter

NO HALOS

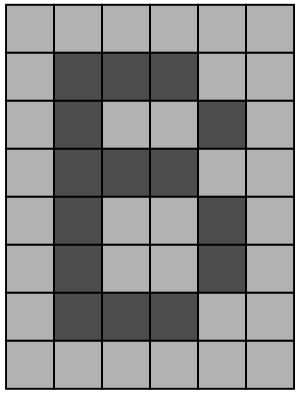
Notation and Definitions



- Image = 2D array of pixels
- Pixel = intensity (scalar) or color (3D vector)
- $I_{\mathbf{p}}$ = value of image I at position: $\mathbf{p} = (p_x, p_y)$
- $F[I]$ = output of filter F applied to image I

What is filtering?

- Images are not smooth because adjacent pixels are different.
- Smoothing = making adjacent pixels look more similar.
- Smoothing strategy
  - pixel → average of its neighbors



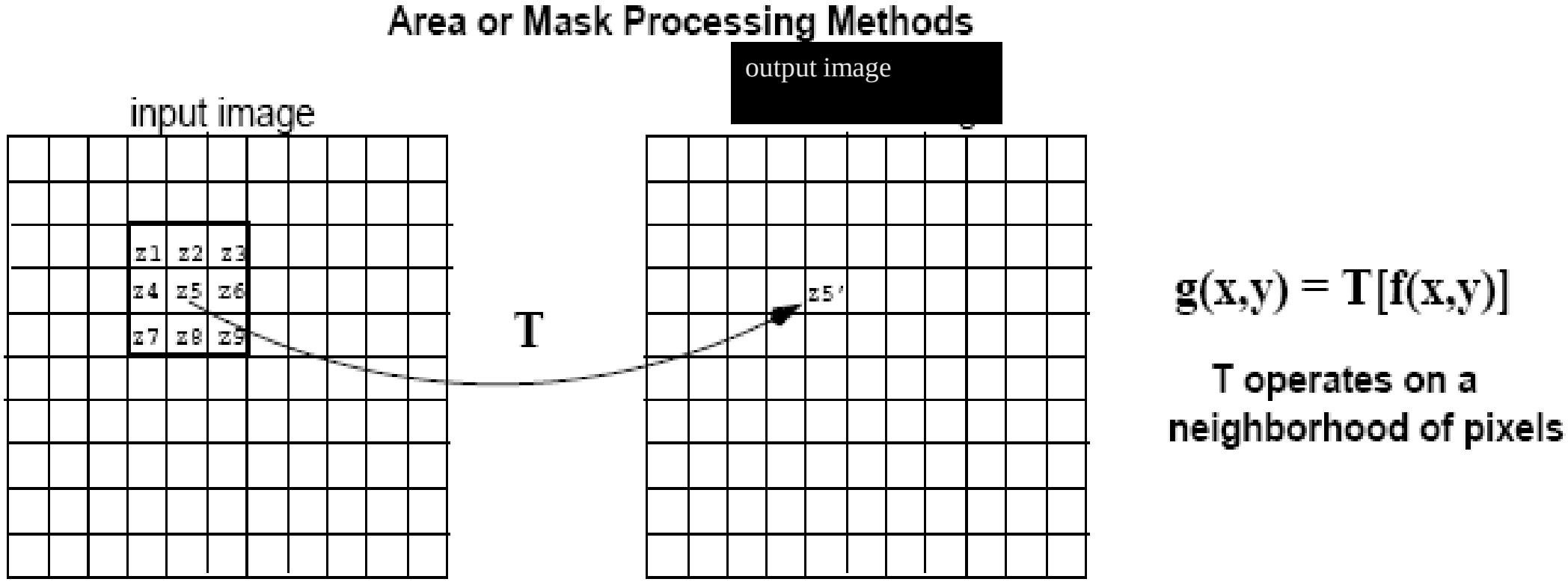
Spatial filtering/Mask processing - Operation

Example: weighted sum of input pixels.

$$z5' = R = w1z1 + w2z2 + \ldots + z9w9$$

mask weights:

w1	w2	w3
w4	w5	w6
w7	w8	w9



$$g(x,y) = T[f(x,y)]$$

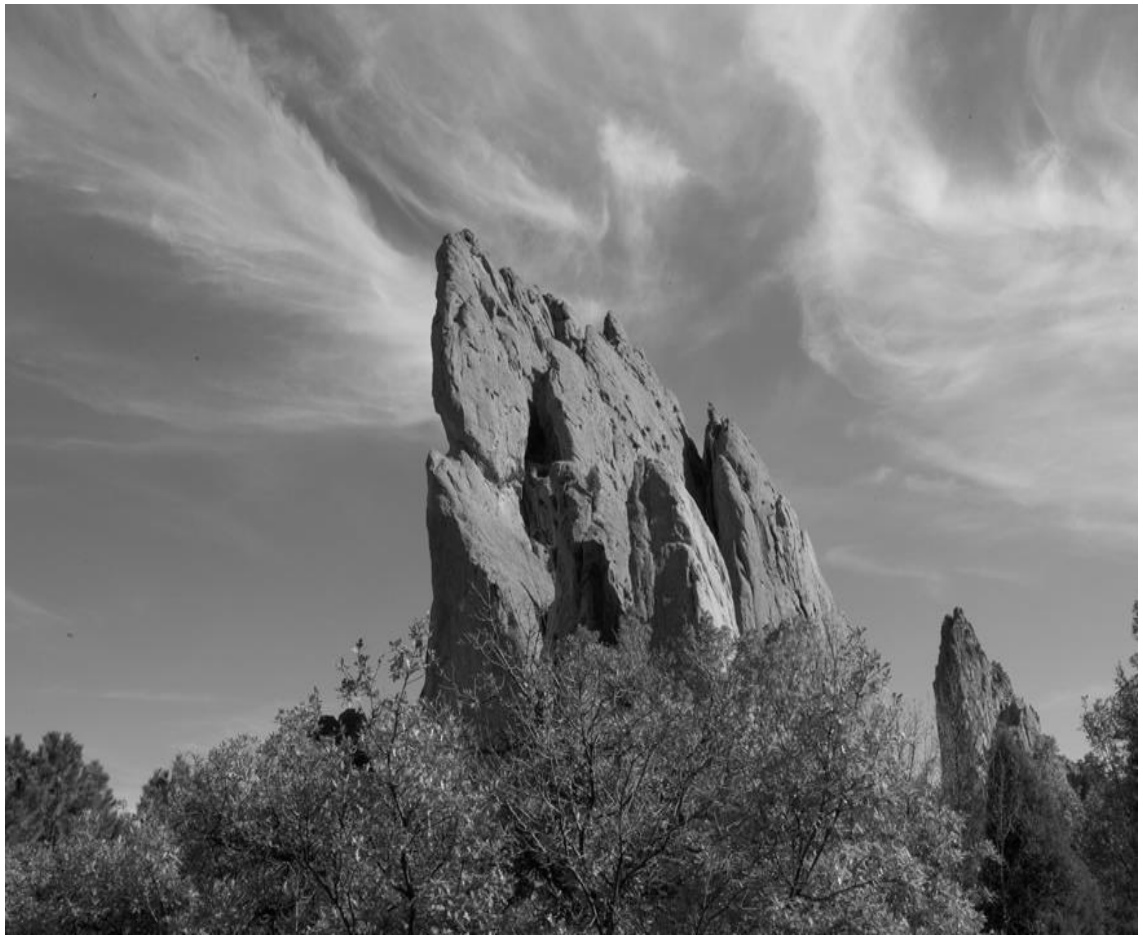
T operates on a neighborhood of pixels

A filtered image is generated as the center of the mask moves to every pixel in the input image.

Why is it important?

- Many applications with high quality results.
  - Computational photography
  - Computer graphics
- Papers in top conferences
  - Siggraph 15: An L1 Image Transform for Edge-Preserving Smoothing and Scene-Level Intrinsic Decomposition
  - Siggraph asia 15: Rolling Guidance Normal Filter for Geometric Processing
  - Siggraph 14: Bilateral texture filtering
  - Siggraph 14: Fast Local Laplacian Filters: Theory and Applications
  - Siggraph asia 14: Depth of field rendering via adaptive recursive filtering
  - …

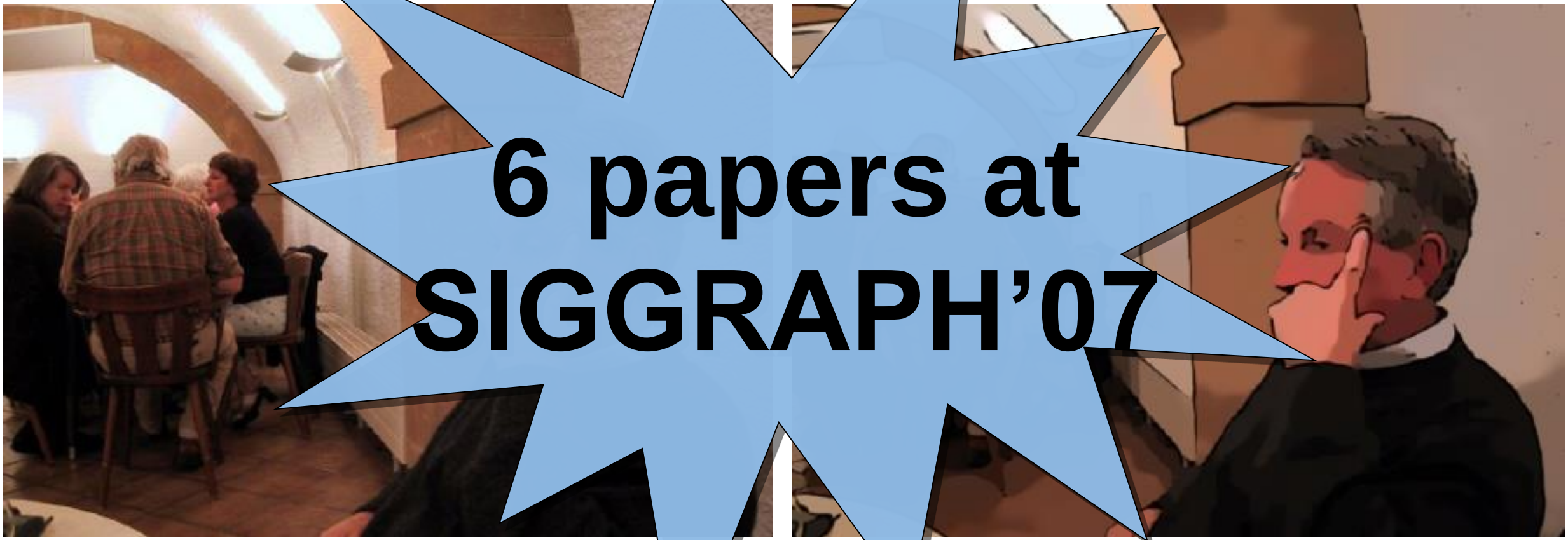
Photographic Style Transfer [Bae 06]



Tone Mapping [Durand 02]

HDR input

Cartoon Rendition [Winnemöller 06]



Intrinsic image decomposition

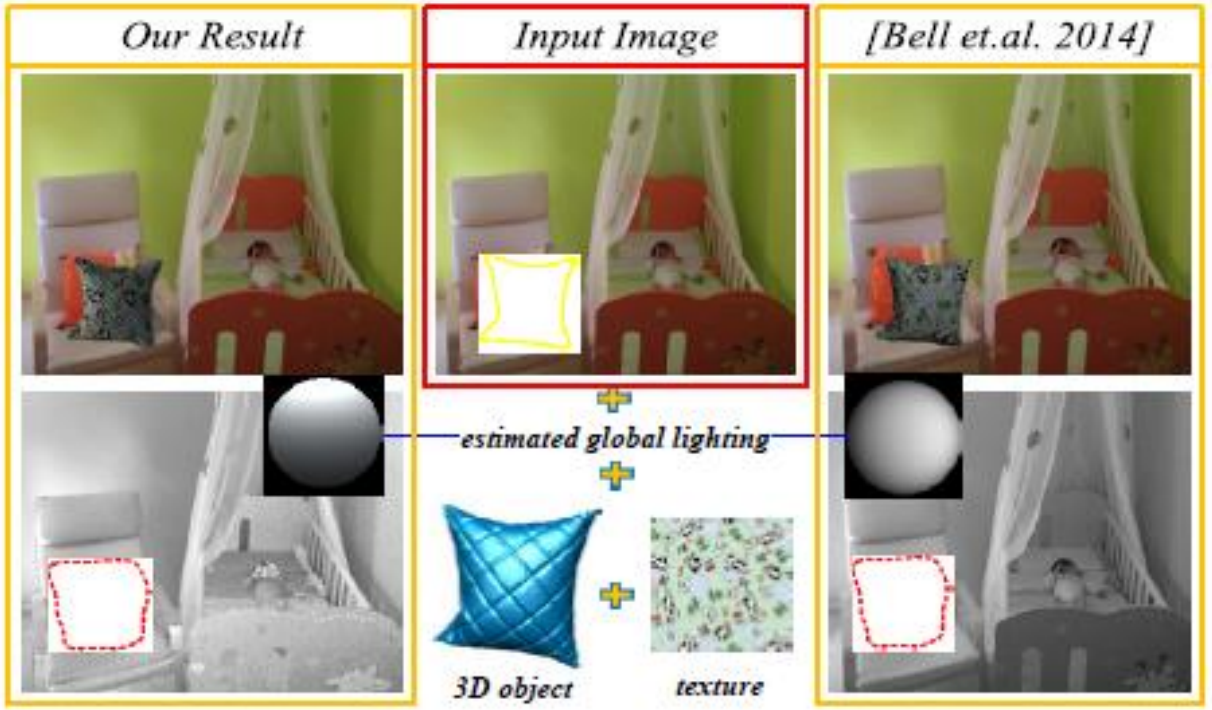
(e) *Original Image*

(f) *Image Flattening*

(g) *Reflectance*

(h) *Shading*

(i) *Re-Texturing*



sig15_An L1 Image Transform for Edge-Preserving Smoothing and Scene-Level Intrinsic Decomposition

Keywords & terms

- Smoothing, denoising, blur, decomposition

- Feature, Structure, texture
- Blur, halo

- Convolution, kernel

- Spatial Filtering,

Many Other Options

- Bilateral filtering is not the only image smoothing filter
  - Diffusion, wavelets, Bayesian…
- We focus on bilateral filtering
  - Suitable for strong smoothing used in computational photography
  - Conceptually simple

Content of the Course

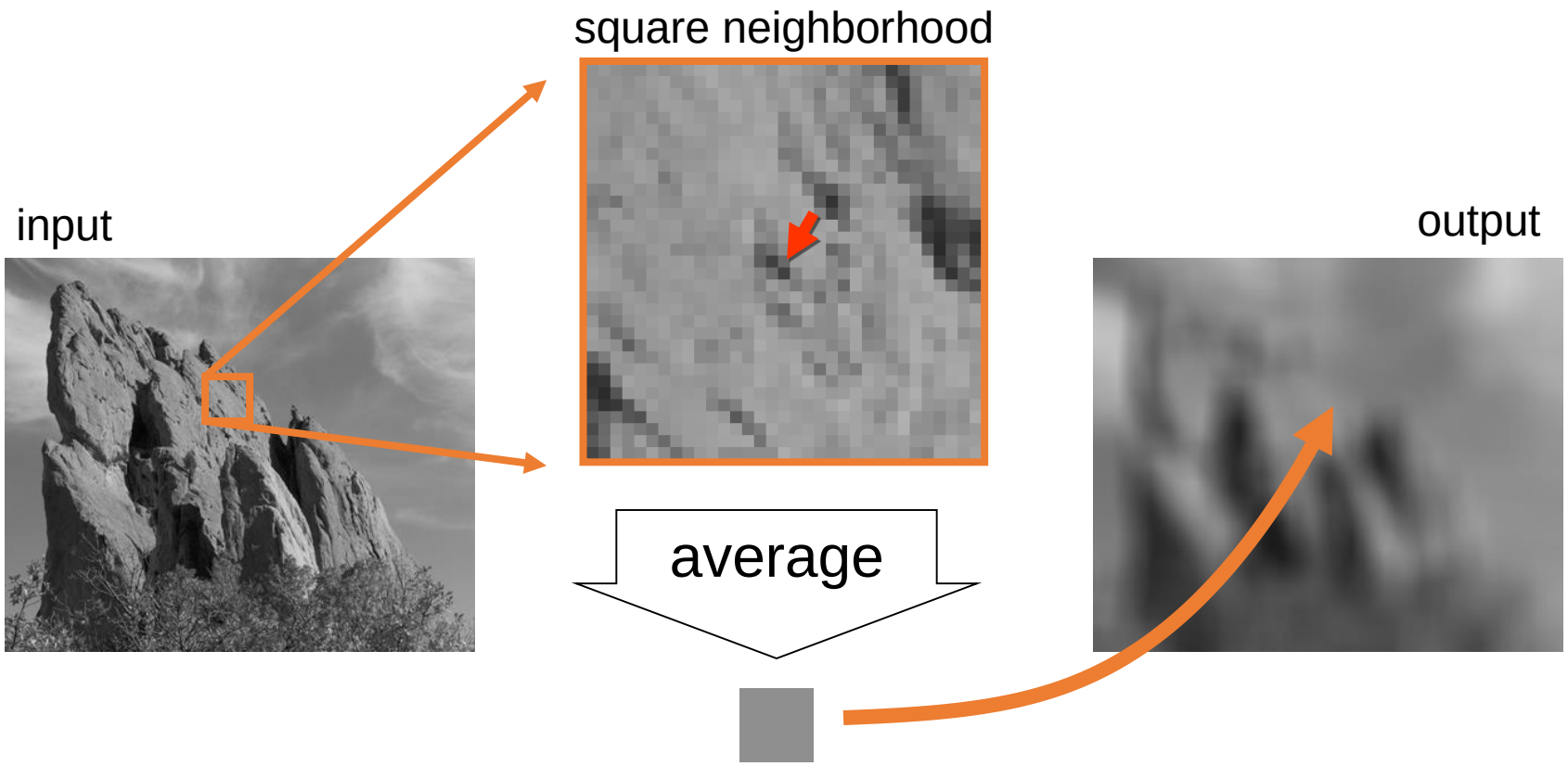
All you need to know about bilateral filtering:

- Definition of the bilateral filter
- Parameter influence and settings
- Applications
- Relationship to other filters
- Theoretical properties
- Efficient implementation

Course webpage (google “bilateral filter course”):
http://people.csail.mit.edu/sparis/bf_course/)
Detailed course notes
C++ and Matlab code

Naïve Image Smoothing: Gaussian Blur

Box Average



Equation of Box Average

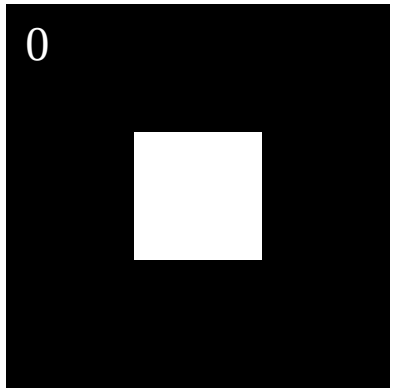
$$BA[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} B_{\sigma}(\mathbf{p} - \mathbf{q}) I_{\mathbf{q}}$$

result at pixel $\mathbf{p}$

sum over all pixels $\mathbf{q}$

normalized box function

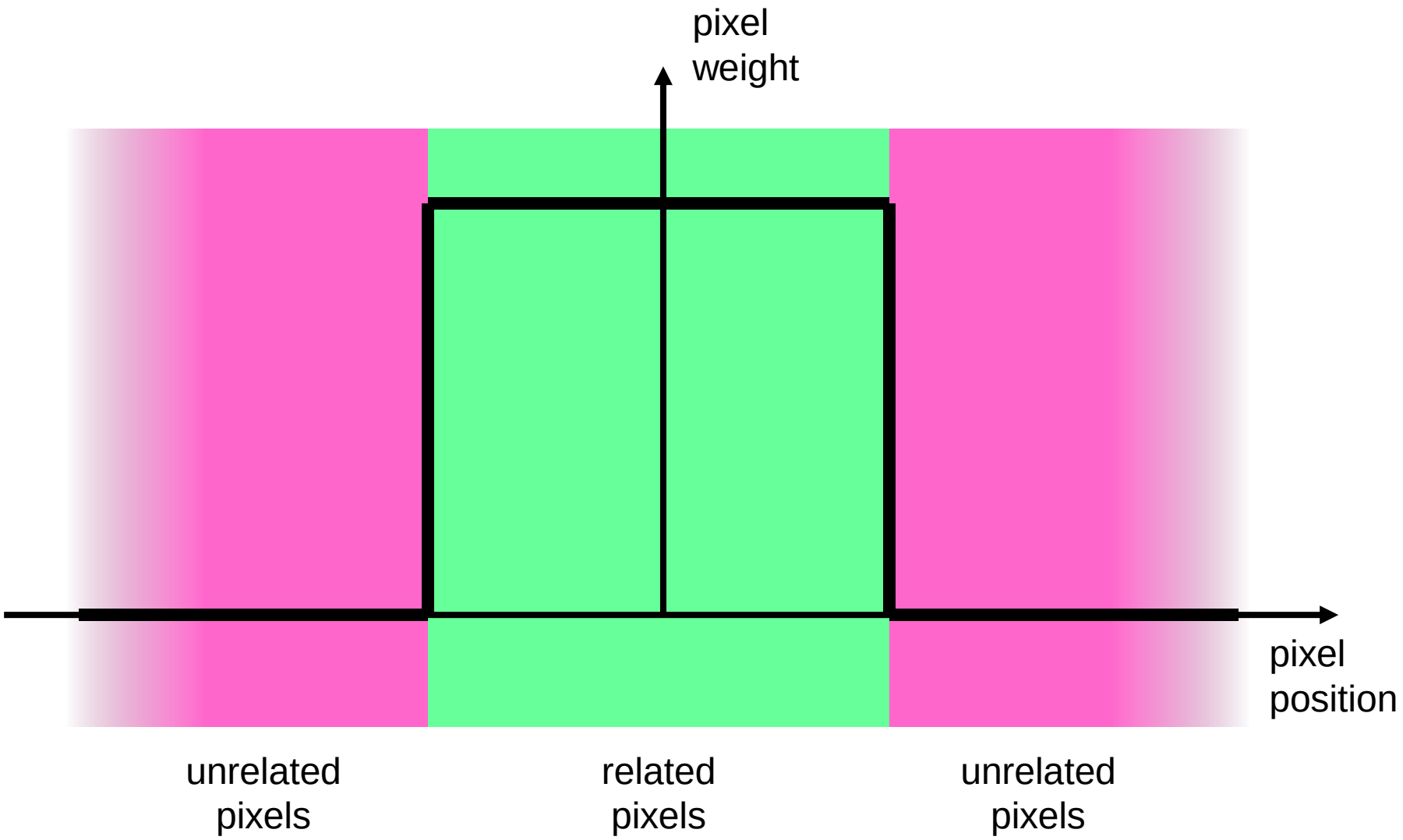
intensity at pixel $\mathbf{q}$



Square Box Generates Defects

- Axis-aligned streaks
- Blocky results

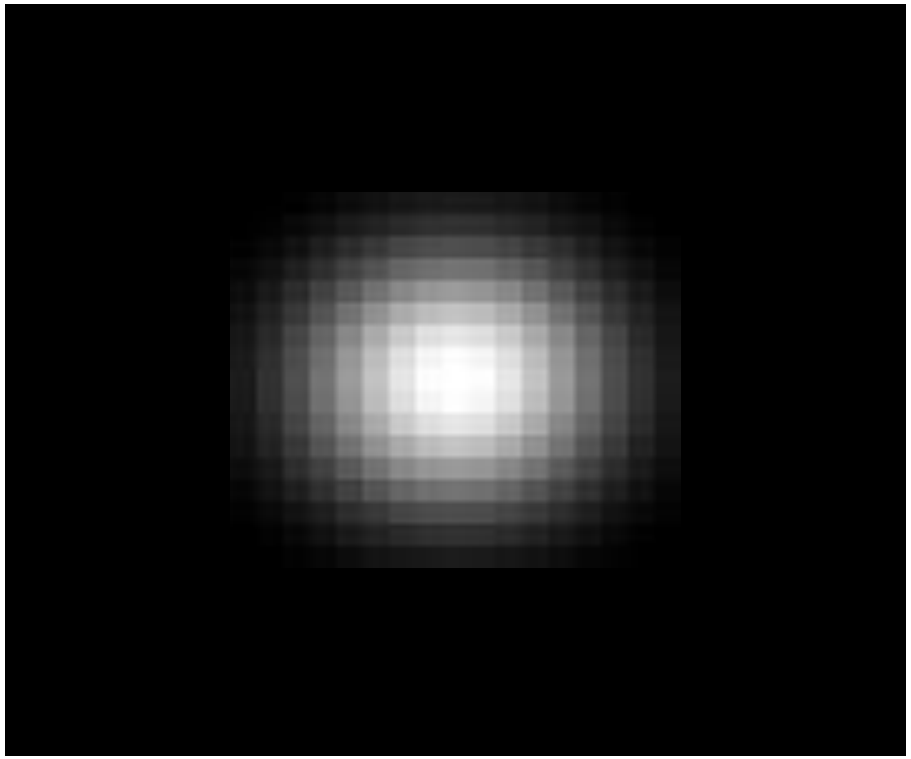
Box Profile


pixel
weight
pixel
position
unrelated
pixels
related
pixels
unrelated
pixels

Strategy to Solve these Problems

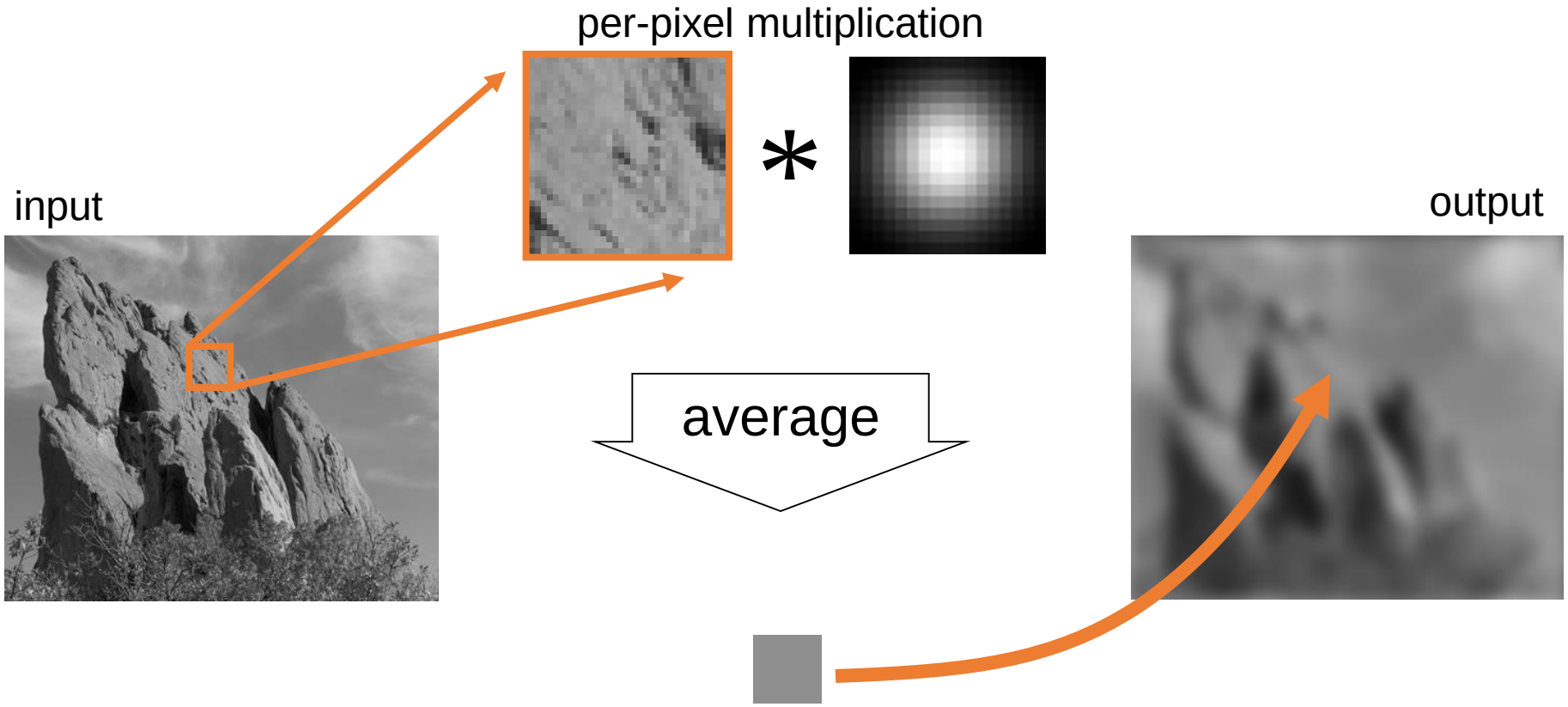
- Use an isotropic (*i.e.* circular) window.
- Use a window with a smooth falloff.

box window



Gaussian window

Gaussian Blur

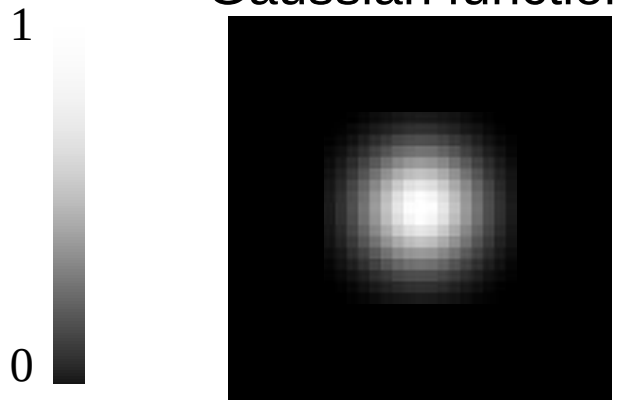


Equation of Gaussian Blur

Same idea: **weighted average of pixels**.

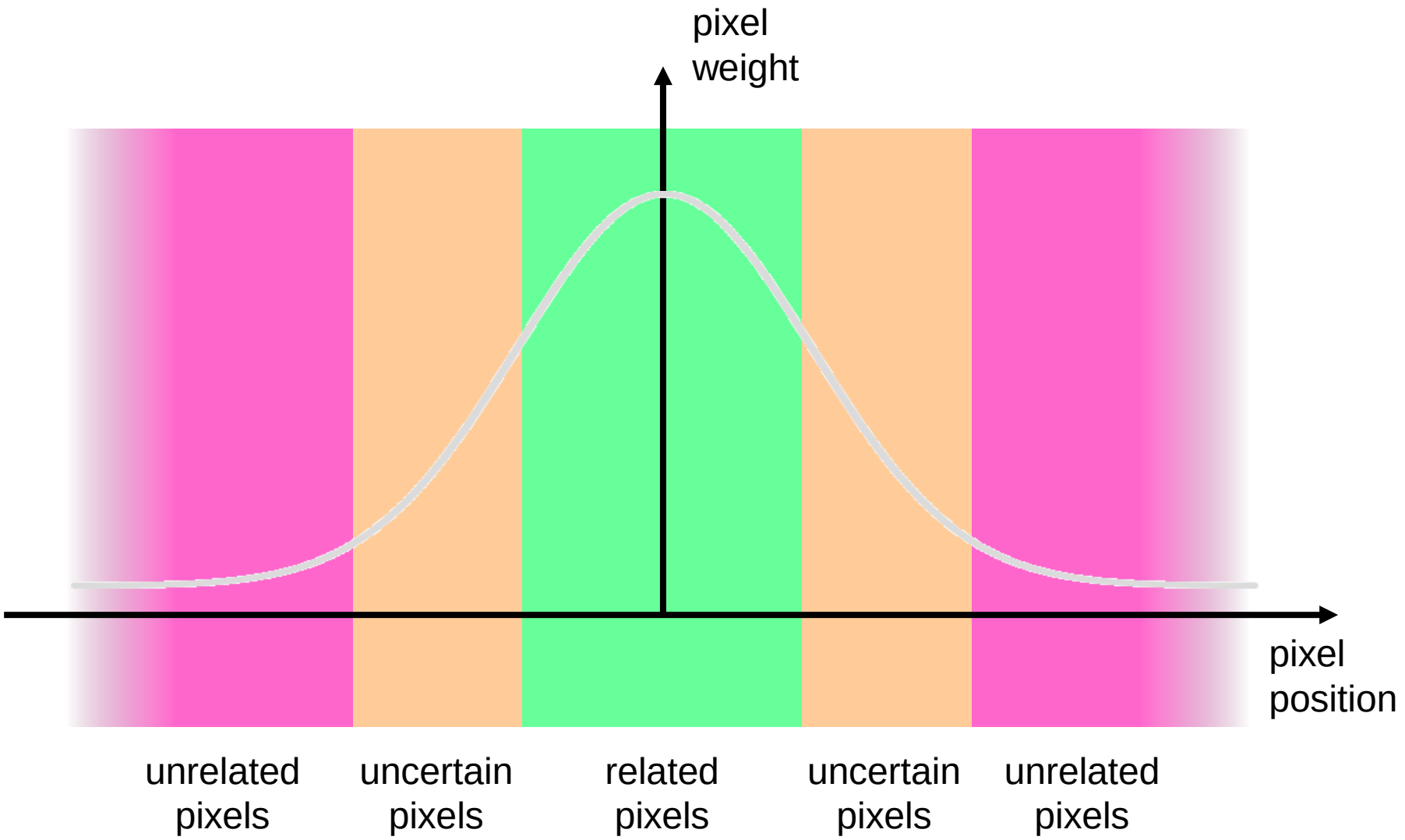
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\|\mathbf{p} - \mathbf{q}\|) I_{\mathbf{q}}$$

normalized
Gaussian function



Gaussian Profile

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

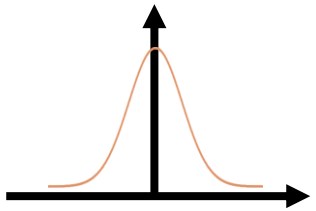


Spatial Parameter

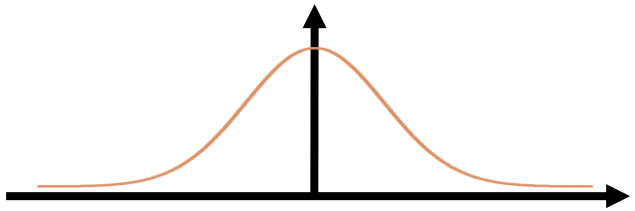
input

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(||\mathbf{p} - \mathbf{q}||) I_{\mathbf{q}}$$

size of the window



small σ



large σ

limited smoothing

strong smoothing

How to set σ

- Depends on the application.

- Common strategy: proportional to image size
  - e.g. 2% of the image diagonal
  - property: independent of image resolution

Properties of Gaussian Blur

- Weights independent of spatial location
  - linear convolution
  - well-known operation
  - efficient computation (recursive algorithm, FFT…)

Properties of Gaussian Blur

- Does smooth images
- But smoothes too much: **edges are blurred**.
  - Only spatial distance matters
  - No edge term

input

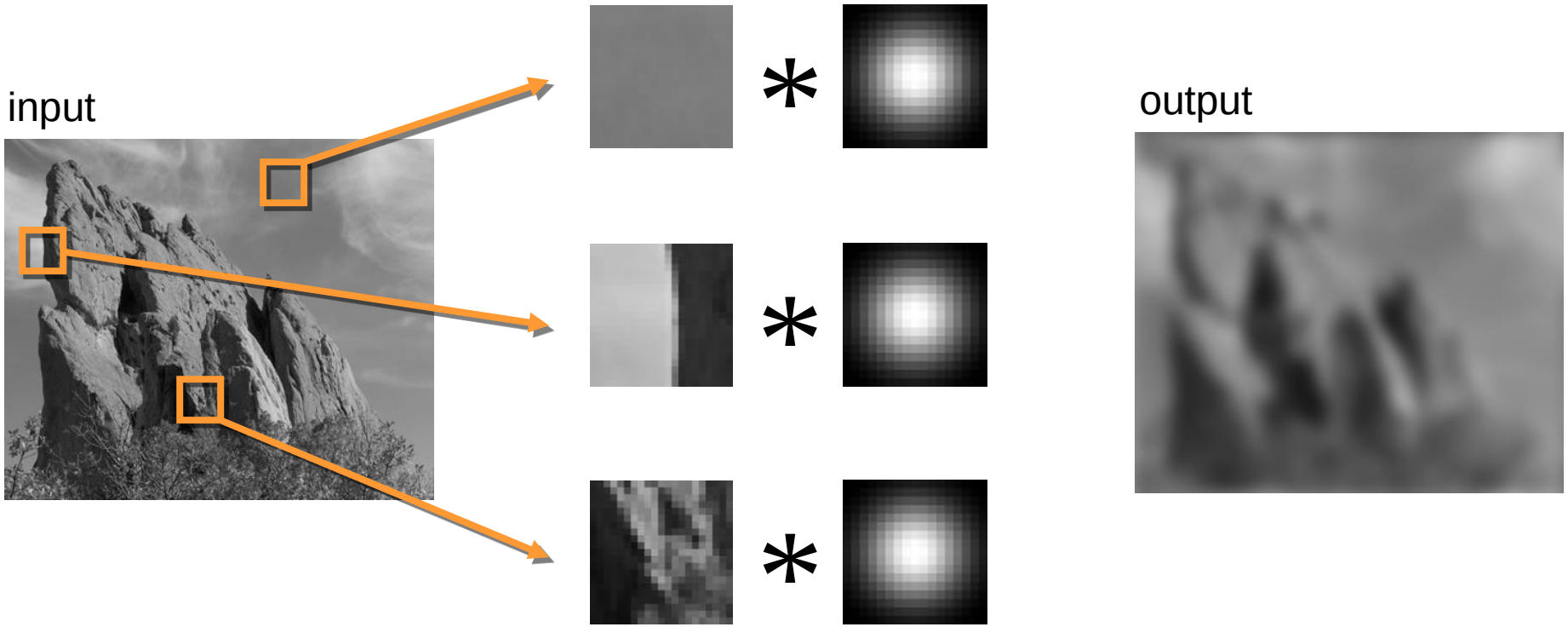
output

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}\left(||\mathbf{p} - \mathbf{q}||\right) I_{\mathbf{q}}$$

space

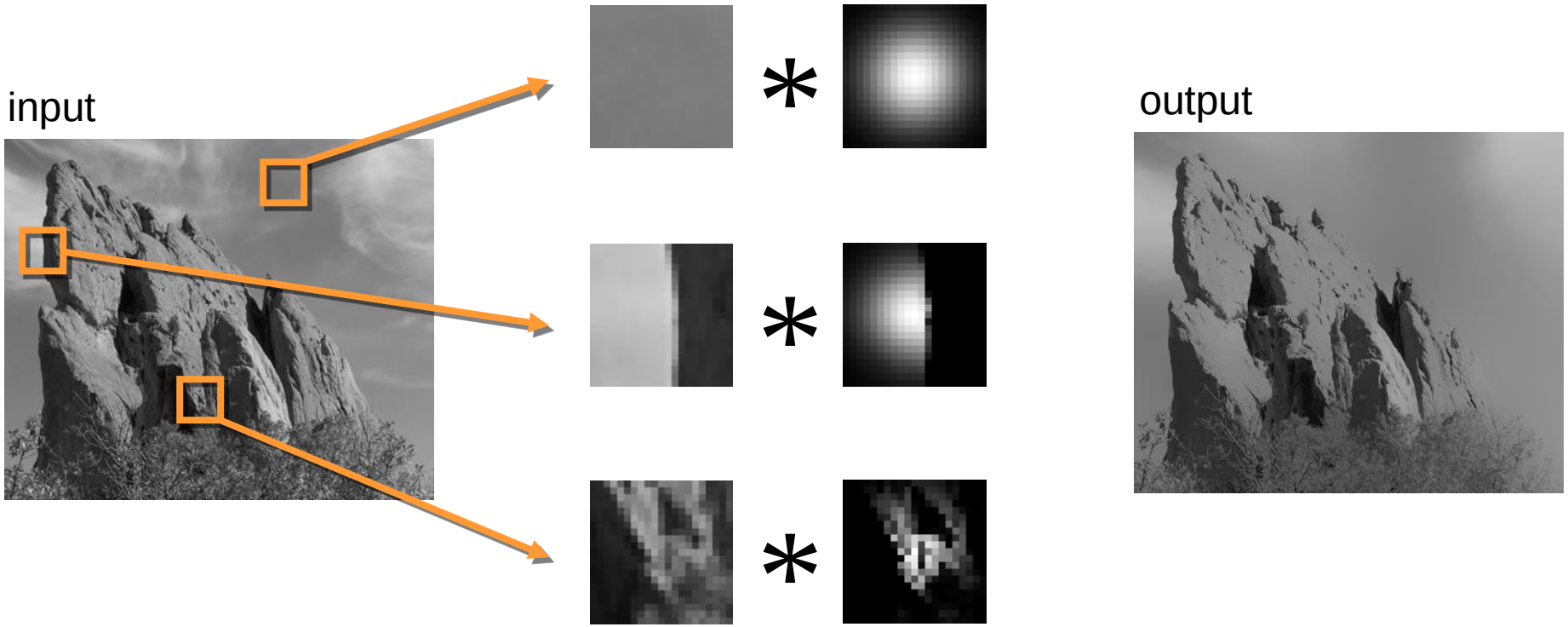
“Fixing the Gaussian Blur”: the Bilateral Filter

Blur Comes from Averaging across Edges



Same Gaussian kernel everywhere.

Bilateral Filter
No Averaging across Edges



The kernel shape depends on the image content.

[Aurich 95, Smith 97, Tomasi 98]

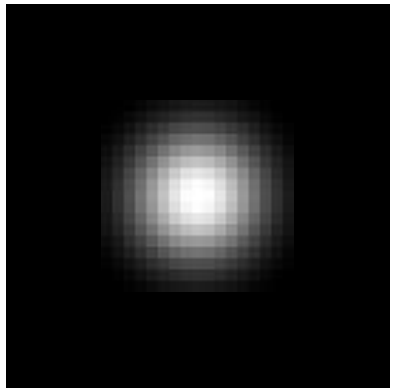
Bilateral Filter Definition: an Additional Edge Term

Same idea: **weighted average of pixels**.

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) \, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) \, I_{\mathbf{q}}$$

new — normalization factor

not new — ***space*** weight



new — ***range*** weight

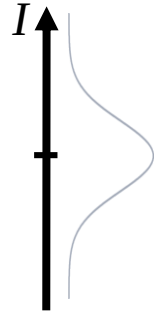
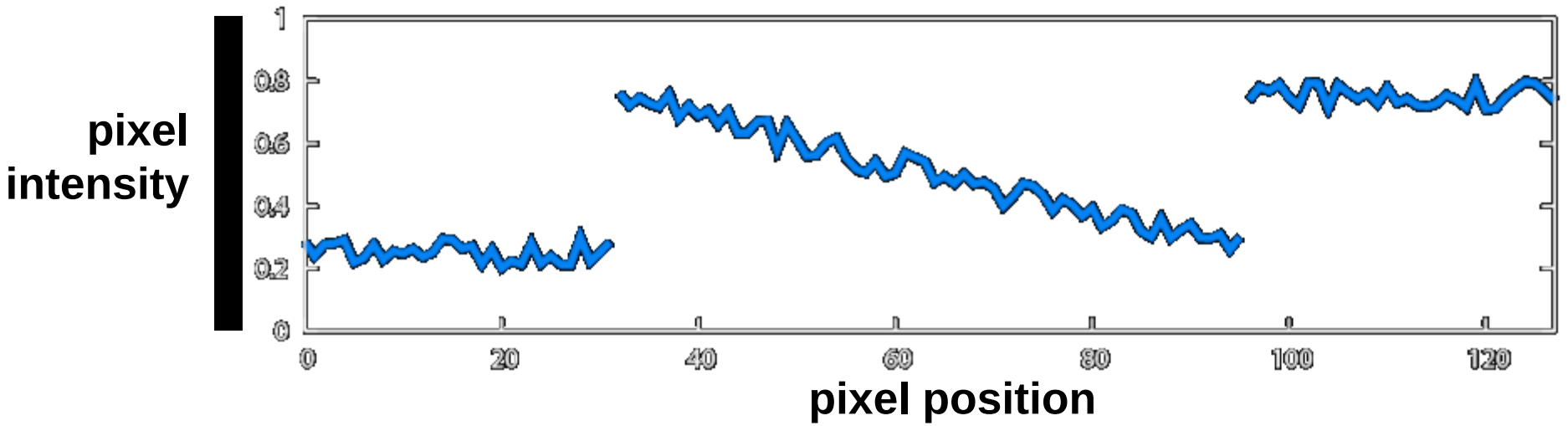


Illustration a 1D Image

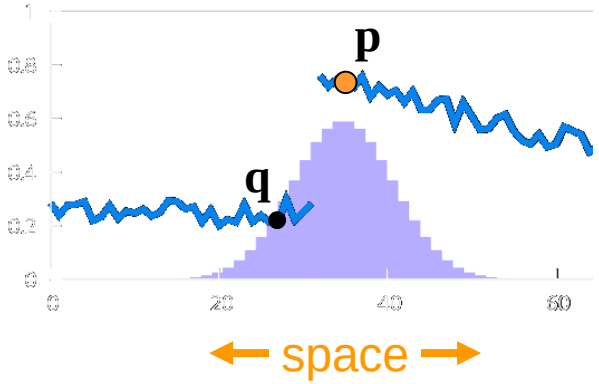
- 1D image = line of pixels

- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

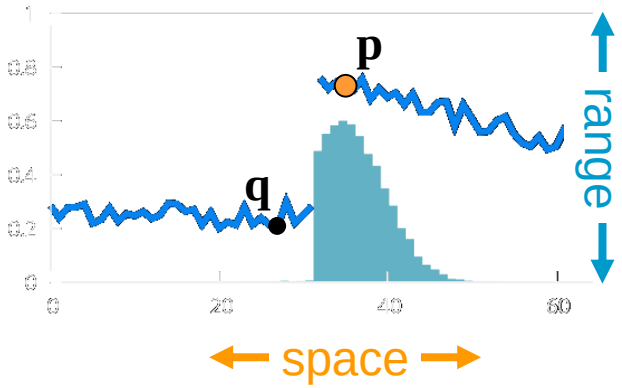


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}\left(\left|\left|\mathbf{p}-\mathbf{q}\right|\right|\right) I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}\left(\left|\left|\mathbf{p}-\mathbf{q}\right|\right|\right) G_{\sigma_r}\left(\left|I_{\mathbf{p}}-I_{\mathbf{q}}\right|\right) I_{\mathbf{q}}$$

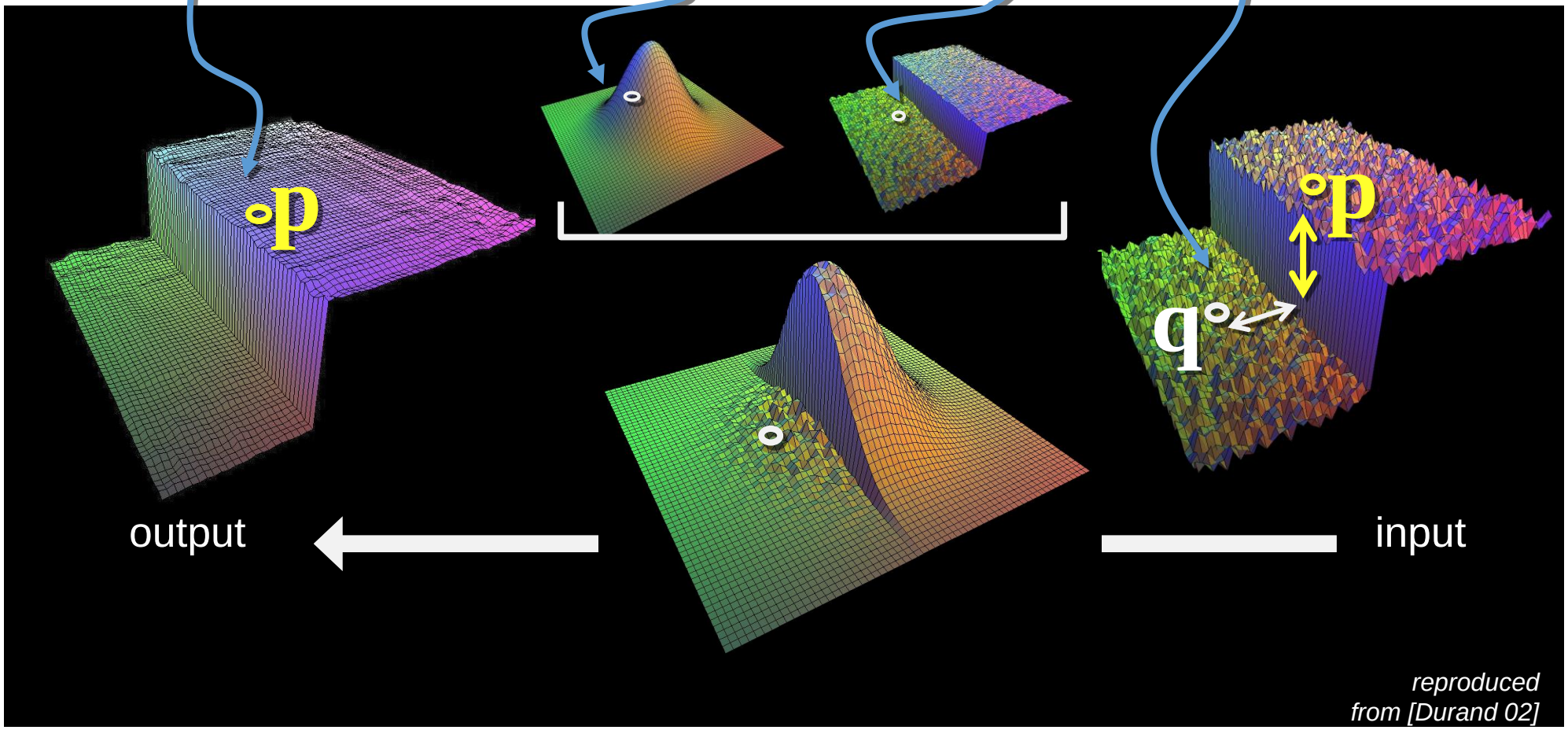
normalization

space

range

Bilateral Filter on a Height Field

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |) I_{\mathbf{q}}$$



reproduced from [Durand 02]

Space and Range Parameters

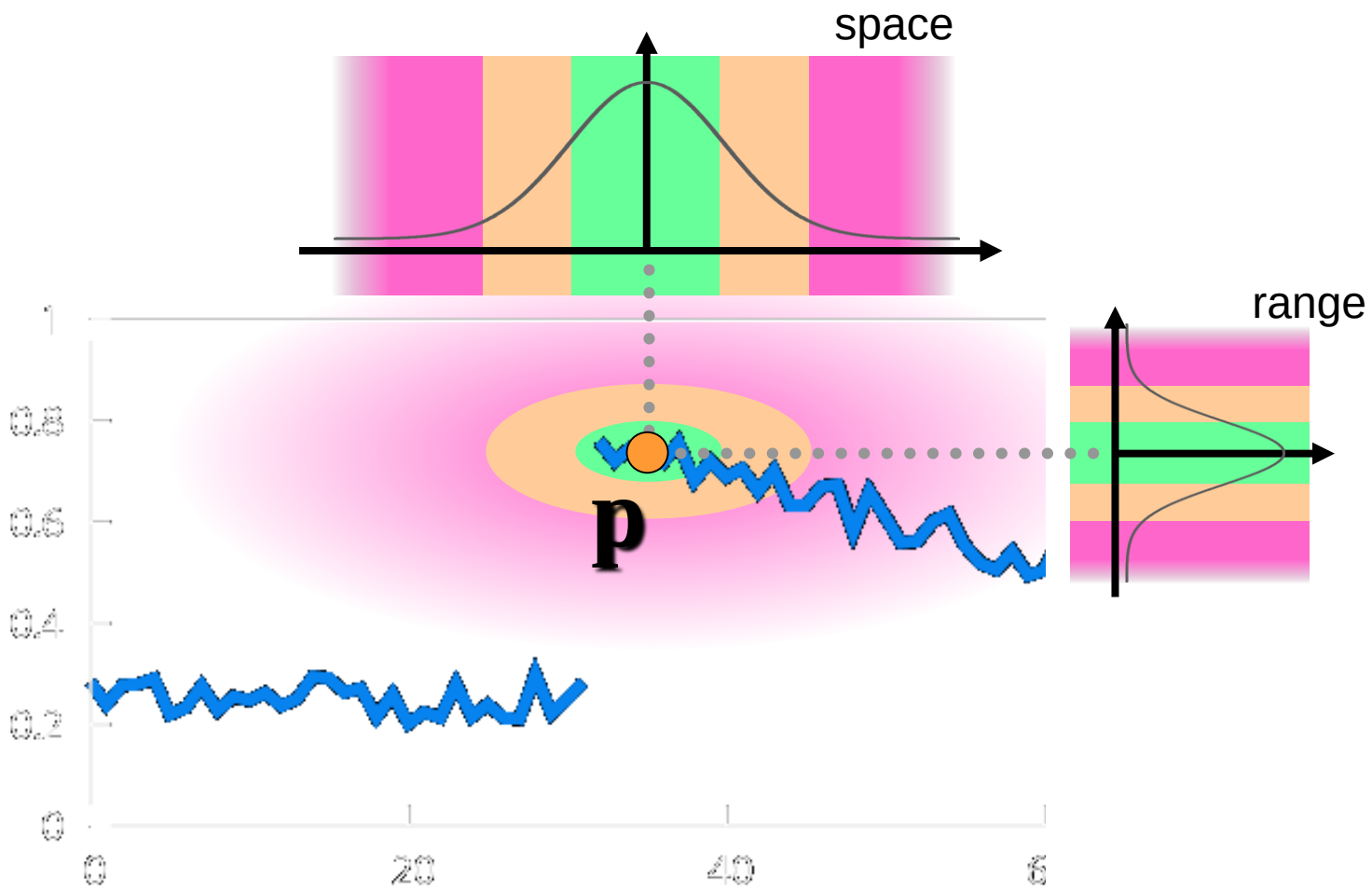
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}\left(\left|\left| \mathbf{p} - \mathbf{q} \right|\right|\right) G_{\sigma_r}\left(\left| I_{\mathbf{p}} - I_{\mathbf{q}} \right|\right) I_{\mathbf{q}}$$

- space σ_s : spatial extent of the kernel, size of the considered neighborhood.

- range σ_r : “minimum” amplitude of an edge

Influence of Pixels

Only pixels close in space and in range are considered.



Varying the Range Parameter

input

$\sigma_r = 0.1$ $\sigma_r = 0.25$ $\sigma_r = \infty$ (Gaussian blur)

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$

Varying the Space Parameter

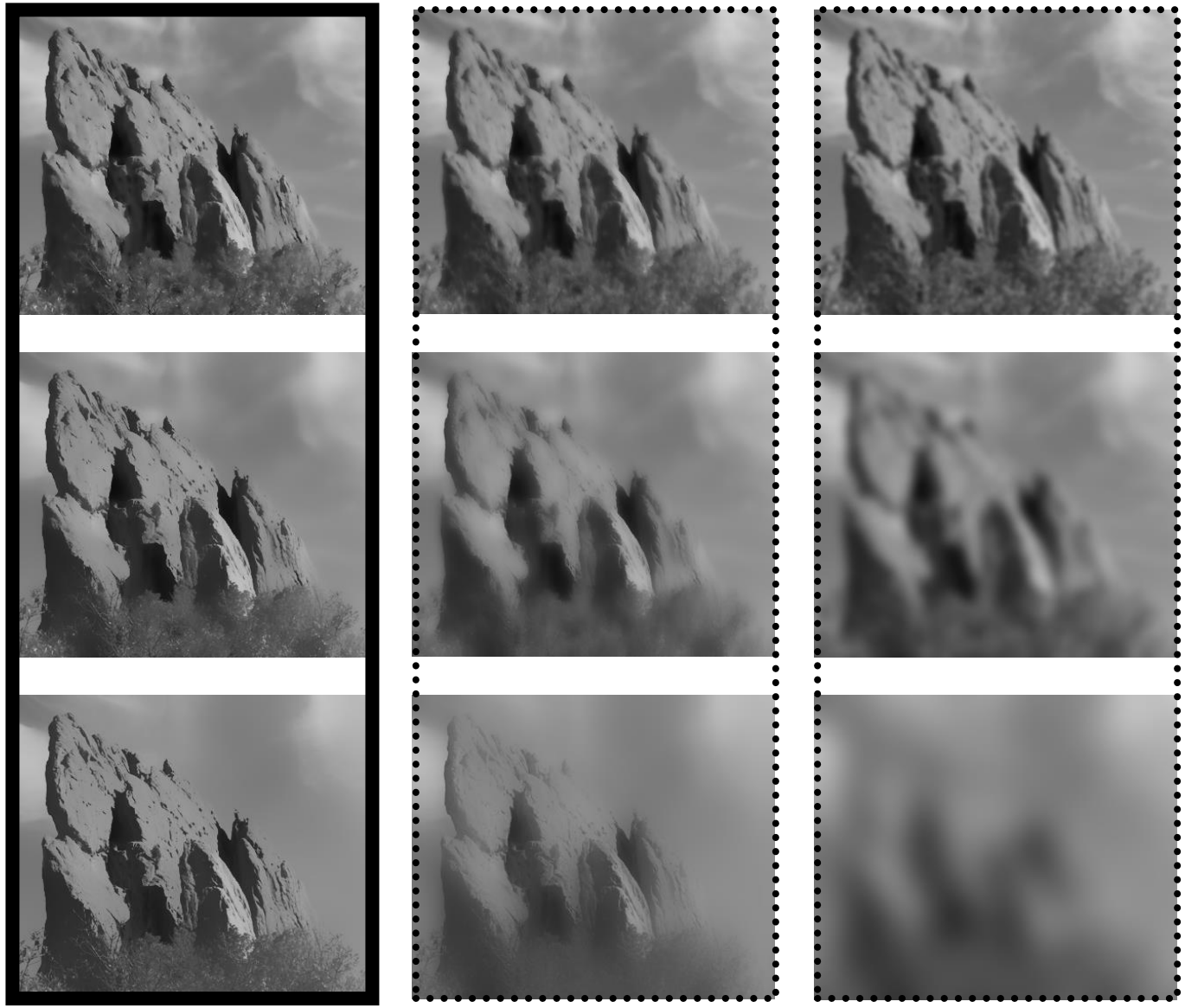
input

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (Gaussian blur)

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$



How to Set the Parameters

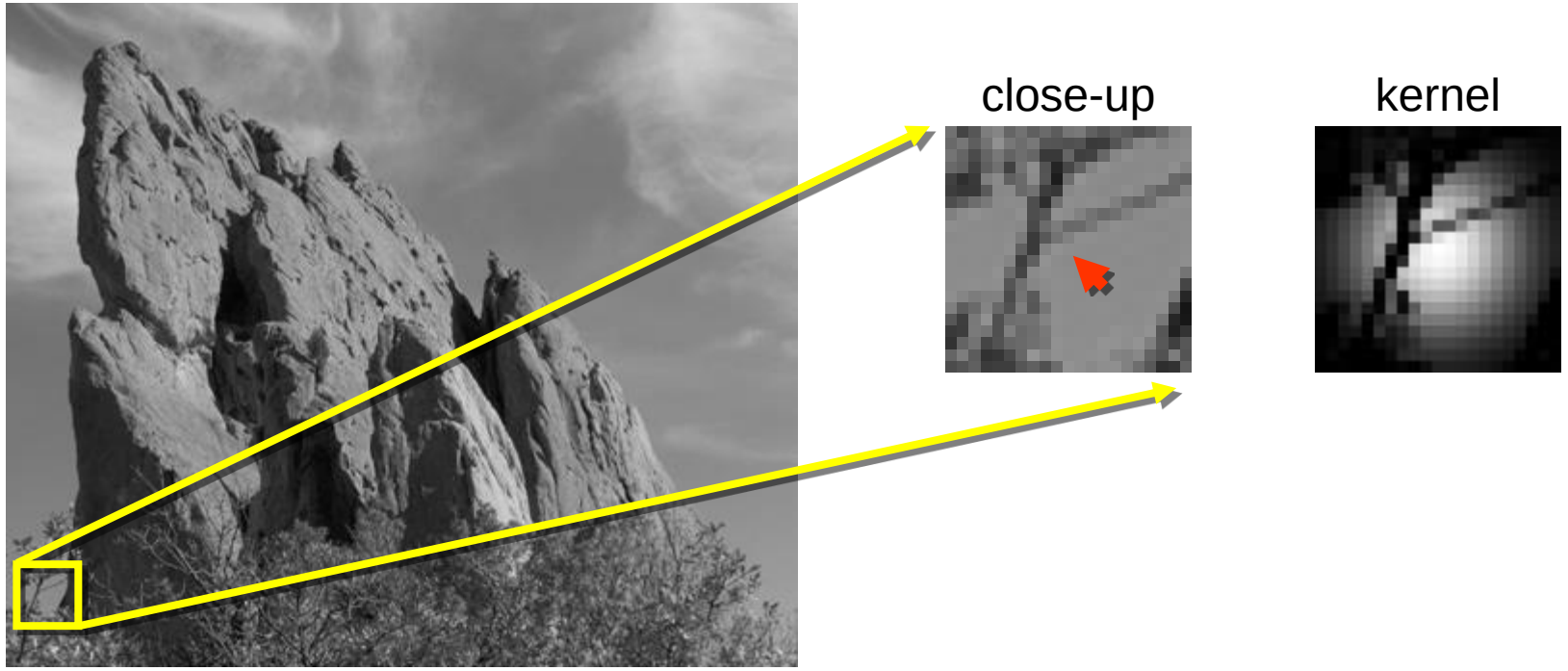
Depends on the application. For instance:

- space parameter: proportional to image size
  - e.g., 2% of image diagonal
- range parameter: proportional to edge amplitude
  - e.g., mean or median of image gradients
- independent of resolution and exposure

A Few
More Advanced
Remarks

Bilateral Filter Crosses Thin Lines

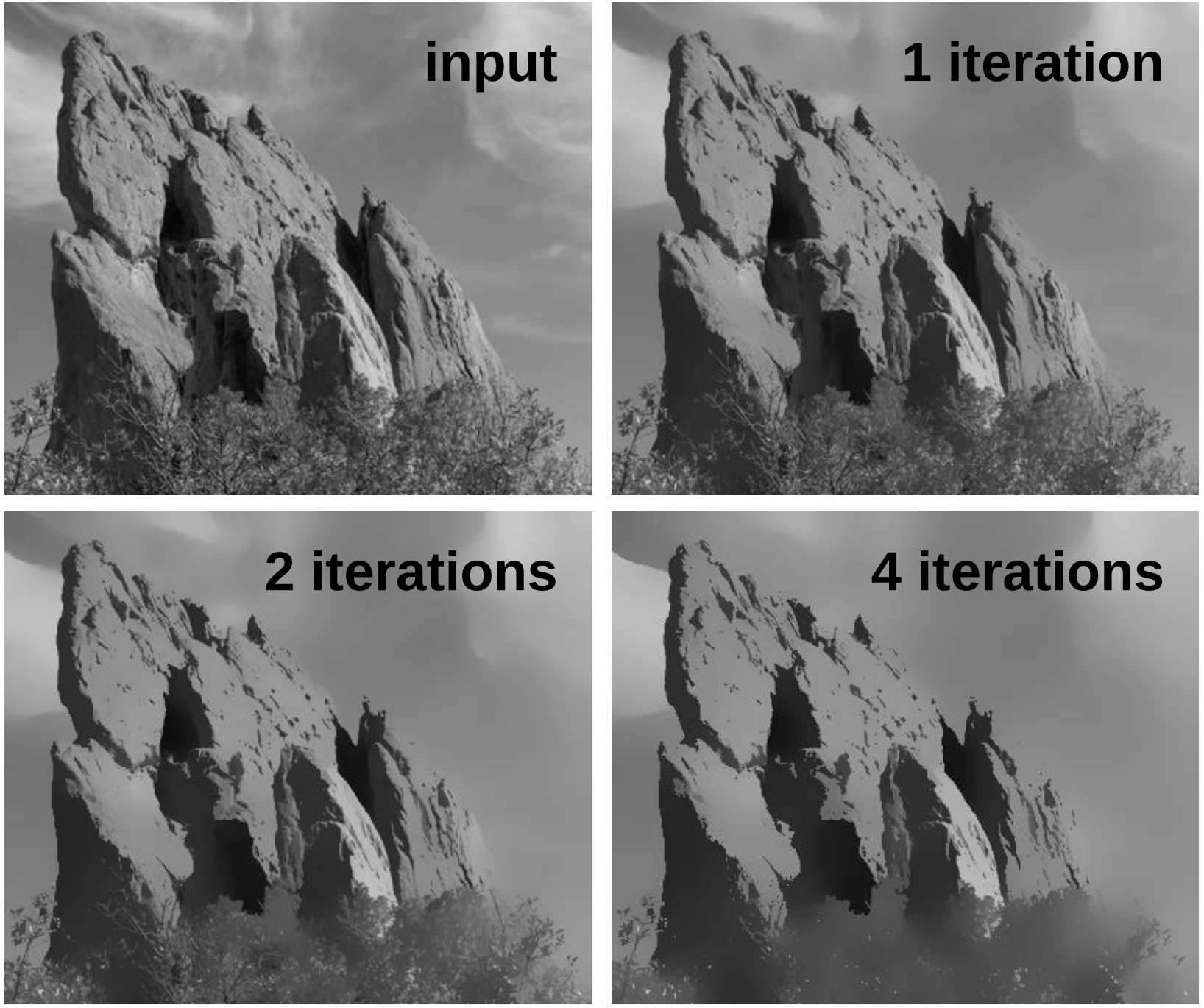
- Desirable for smoothing: more pixels = more robust
- Different from diffusion that stops at thin lines
- Bilateral filter averages across features thinner than $\sim 2\sigma_s$



Iterating the Bilateral Filter

$$I_{(n+1)} = BF\left[I_{(n)}\right]$$

- Generate more piecewise-flat images
- Often not needed in computational photo.


input
1 iteration
2 iterations
4 iterations

Bilateral Filtering Color Images

For gray-level images

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

intensity difference

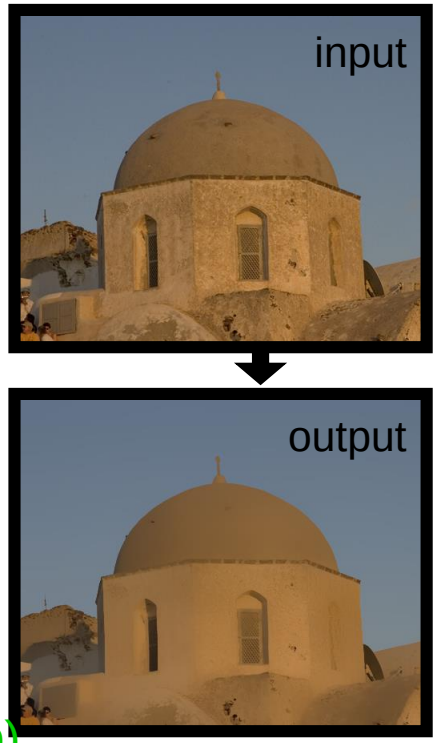
scalar

For color images

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(\|\mathbf{C}_{\mathbf{p}} - \mathbf{C}_{\mathbf{q}}\|)\, \mathbf{C}_{\mathbf{q}}$$

color difference

3D vector
(RGB, Lab)



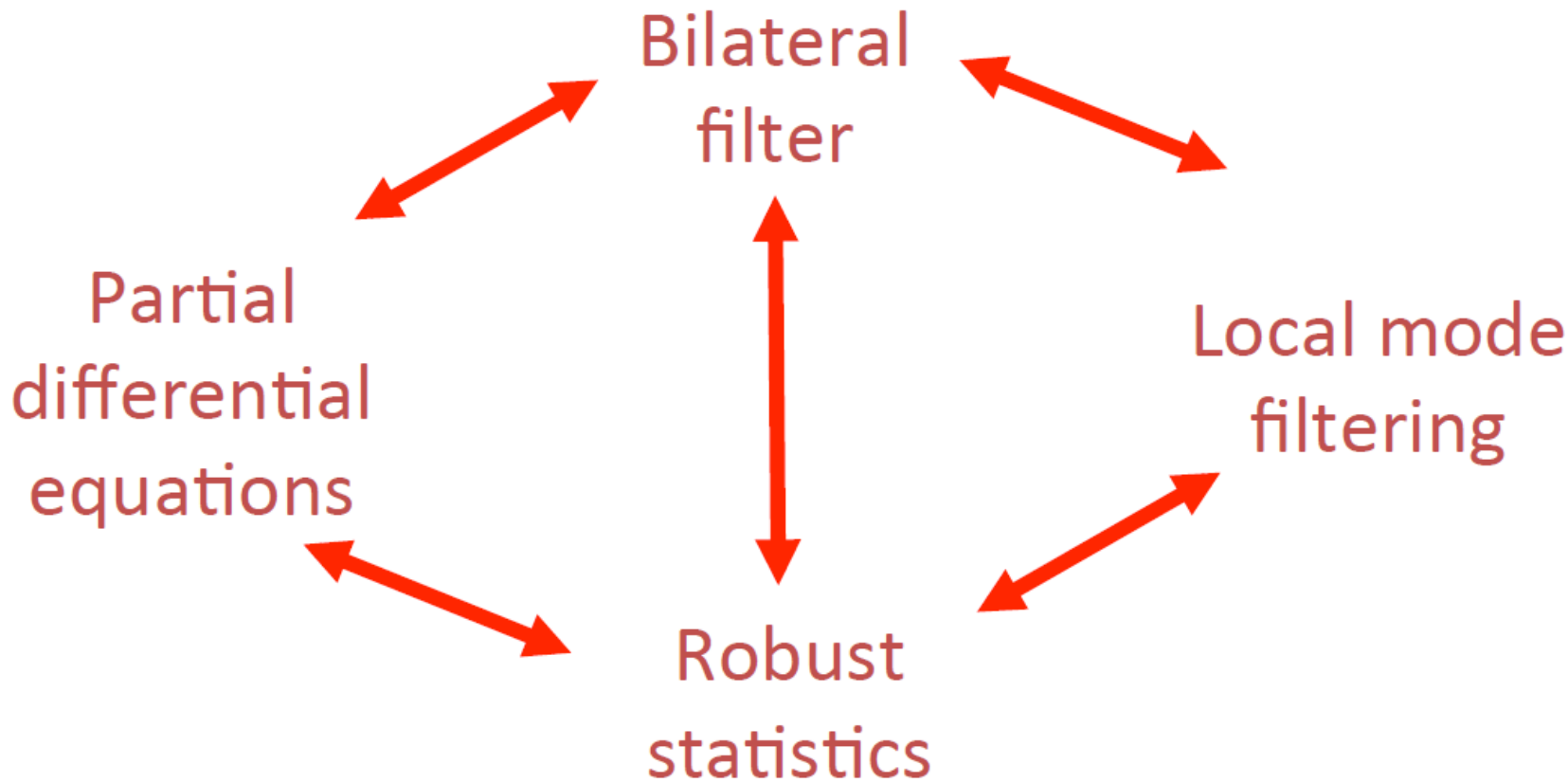
The bilateral filter is extremely easy to adapt to your need.

Hard to Compute

- Nonlinear $$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p} - \mathbf{q}||) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$
- Complex, spatially varying kernels
  - Cannot be precomputed, no FFT…

- Brute-force implementation is slow > 10min

Goal: Understand how does bilateral filter relates with other methods



Additional Reading: *Generalised Nonlocal Image Smoothing*, L. Pizarro, P. Mrazek, S. Didas, S. Grewenig and J. Weickert, IJCV, 2010

Data-driven image processing: “Image manipulation by example”

(main idea: pixel patterns in another part of the image are hints for how to improve image in the current region)

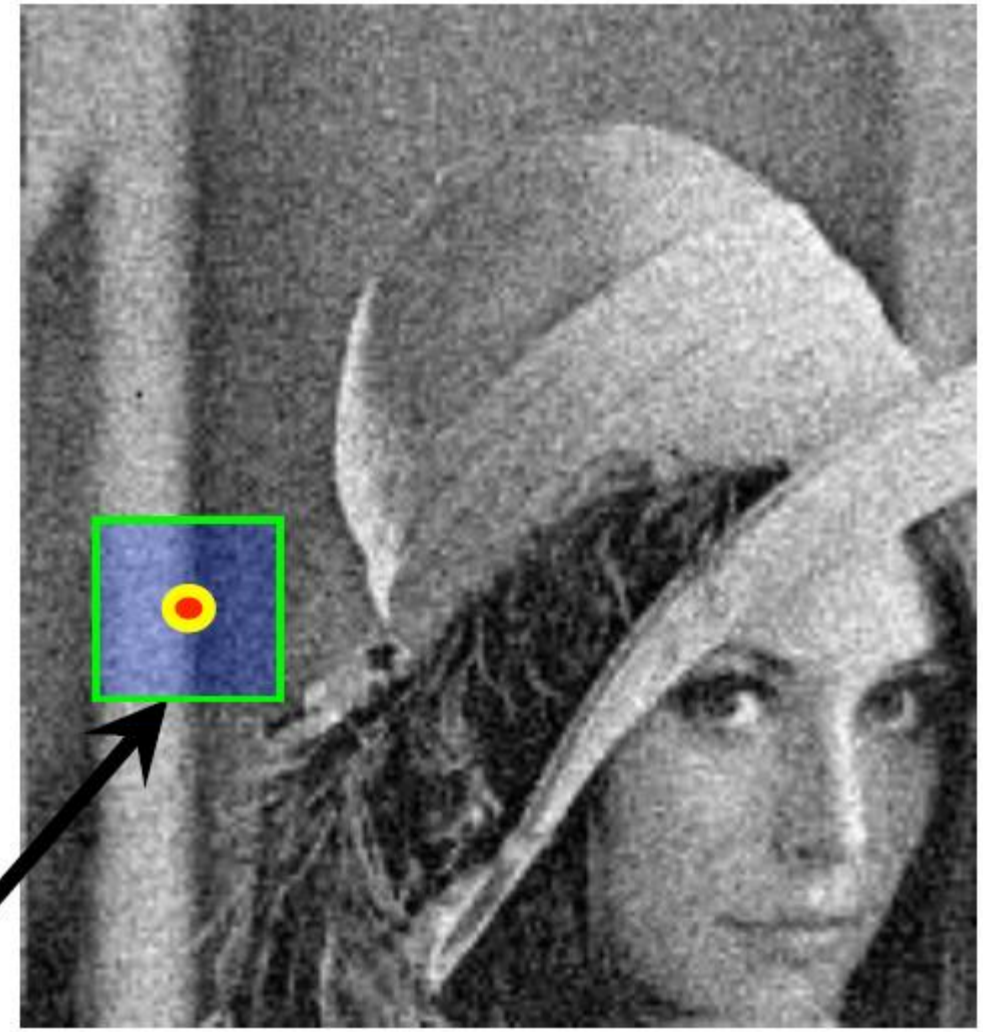
New Idea: NL-Means Filter (Buades 2005)

- Same goals: ‘Smooth within Similar Regions’
- **KEY INSIGHT**: Generalize, extend ‘Similarity’
  - **Bilateral:**
    - Averages local neighbors with **similar intensities**;
  - **NL-Means:**
    - Averages nonlocal neighbors with **similar neighborhoods!**

NL-Means Method: Buades (2005)

- For each and every pixel **p:**

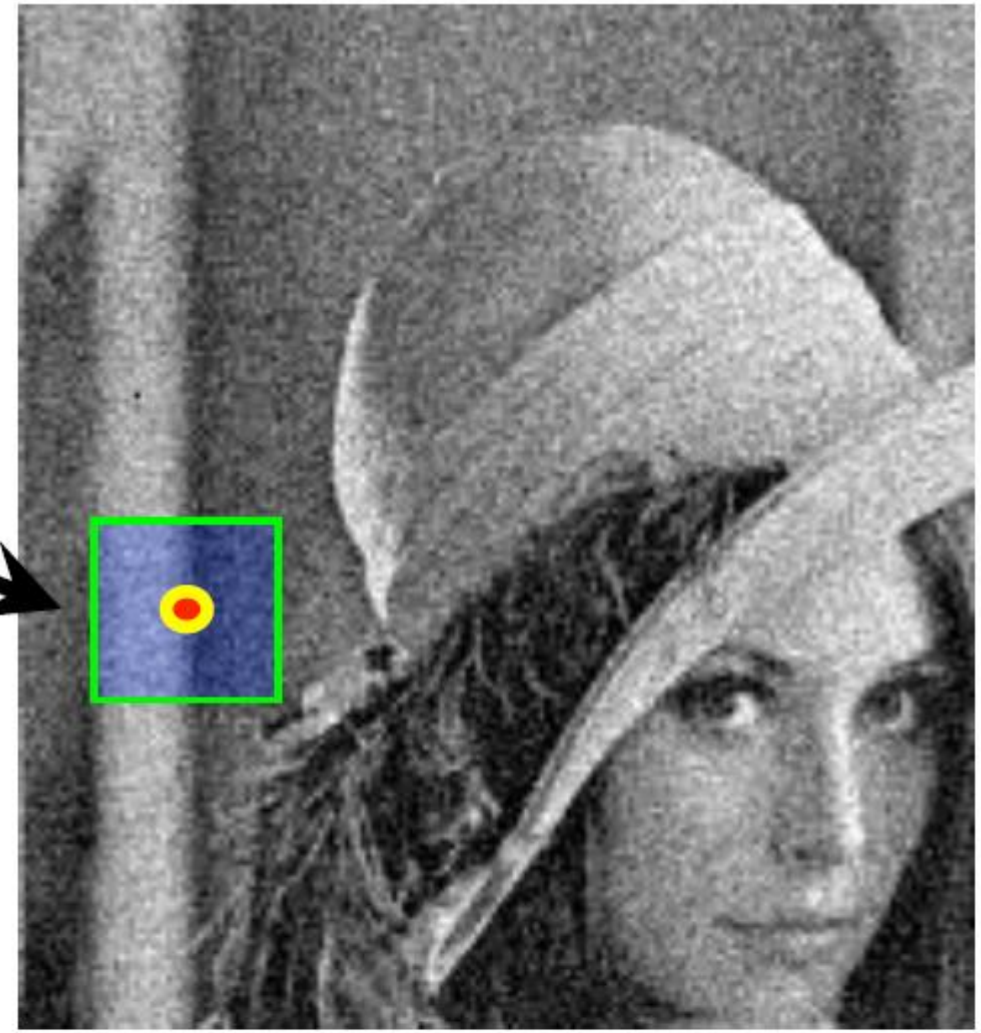
NL-Means Method: Buades (2005)



- For each and every pixel **p:**
  - Define a small, simple fixed size neighborhood;

NL-Means Method: Buades (2005)

$$V_p = \begin{bmatrix} 0.74 \\ 0.32 \\ 0.41 \\ 0.55 \\ \dots \\ \dots \\ \dots \end{bmatrix}$$



- For each and every pixel **p:**
  - Define a small, simple fixed size neighborhood;
  - Define vector $\mathbf{V_p}$: a list of neighboring pixel values.

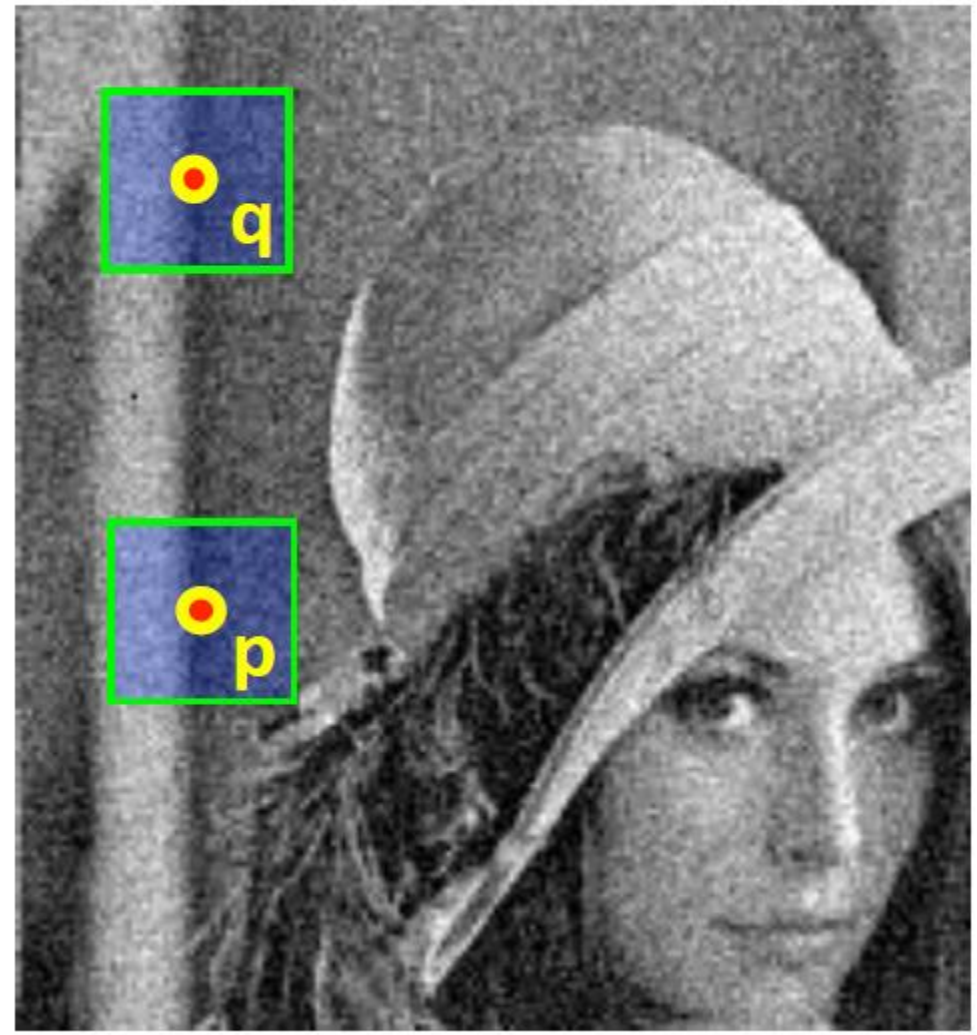
NL-Means Method: Buades (2005)

'Similar' pixels **p, q**

→ **SMALL**

vector distance;

$$|| V_p - V_q ||^2$$

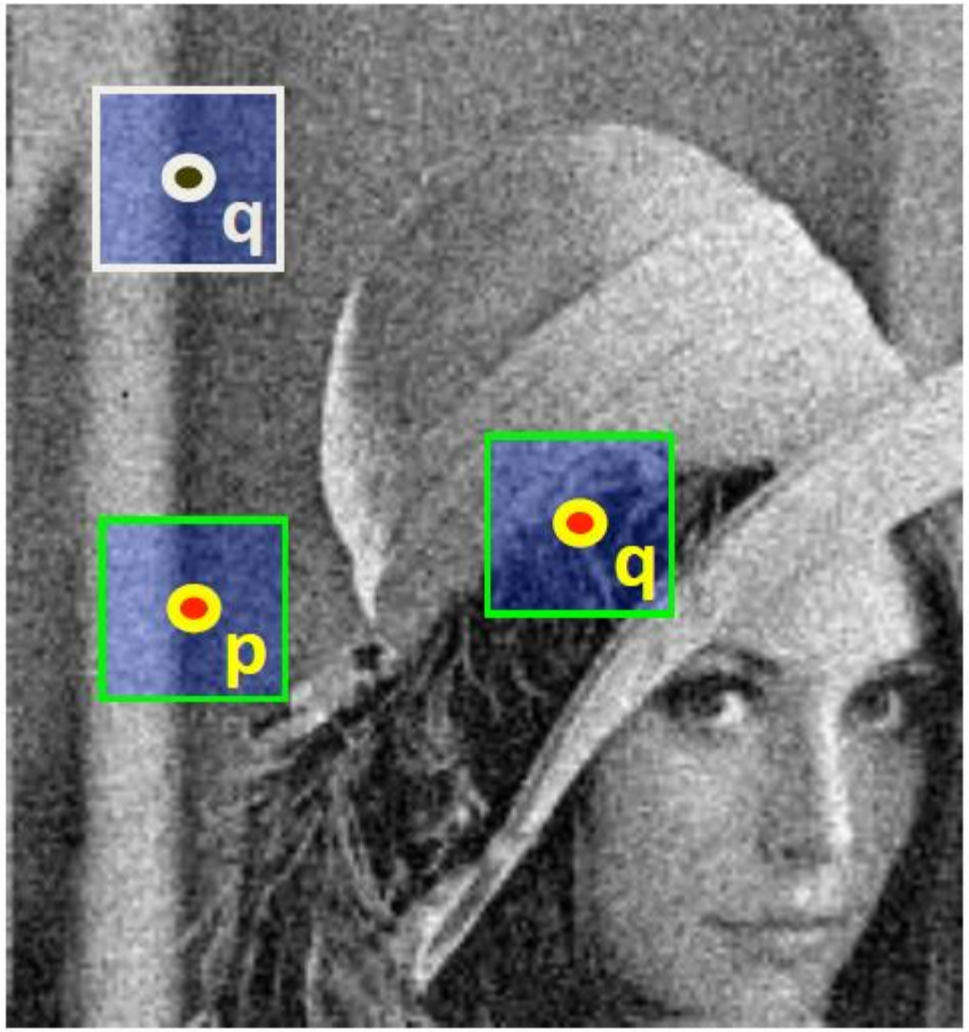


Denoising using non-local means

NL-Means Method:
Buades (2005)

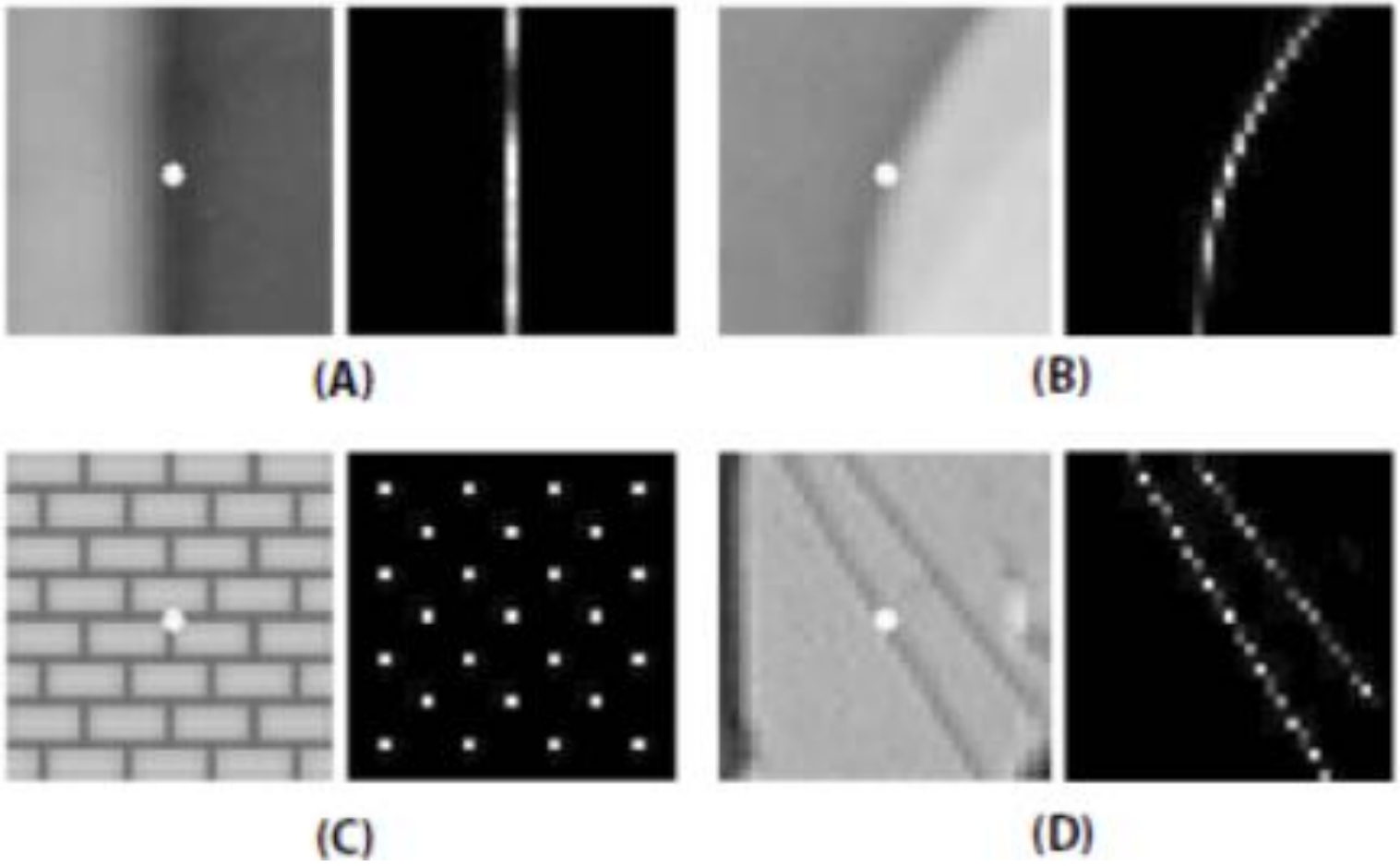
'Dissimilar' pixels **p, q**
→ **LARGE**
vector distance;

$$|| V_p - V_q ||^2$$



Denoising using non-local means

- **In each image pair below:**
  - **Image at left shows the pixel *p* to denoise.**
  - **Image at right shows weights of pixels in 21x21-pixel kernel support window surrounding *p*.**



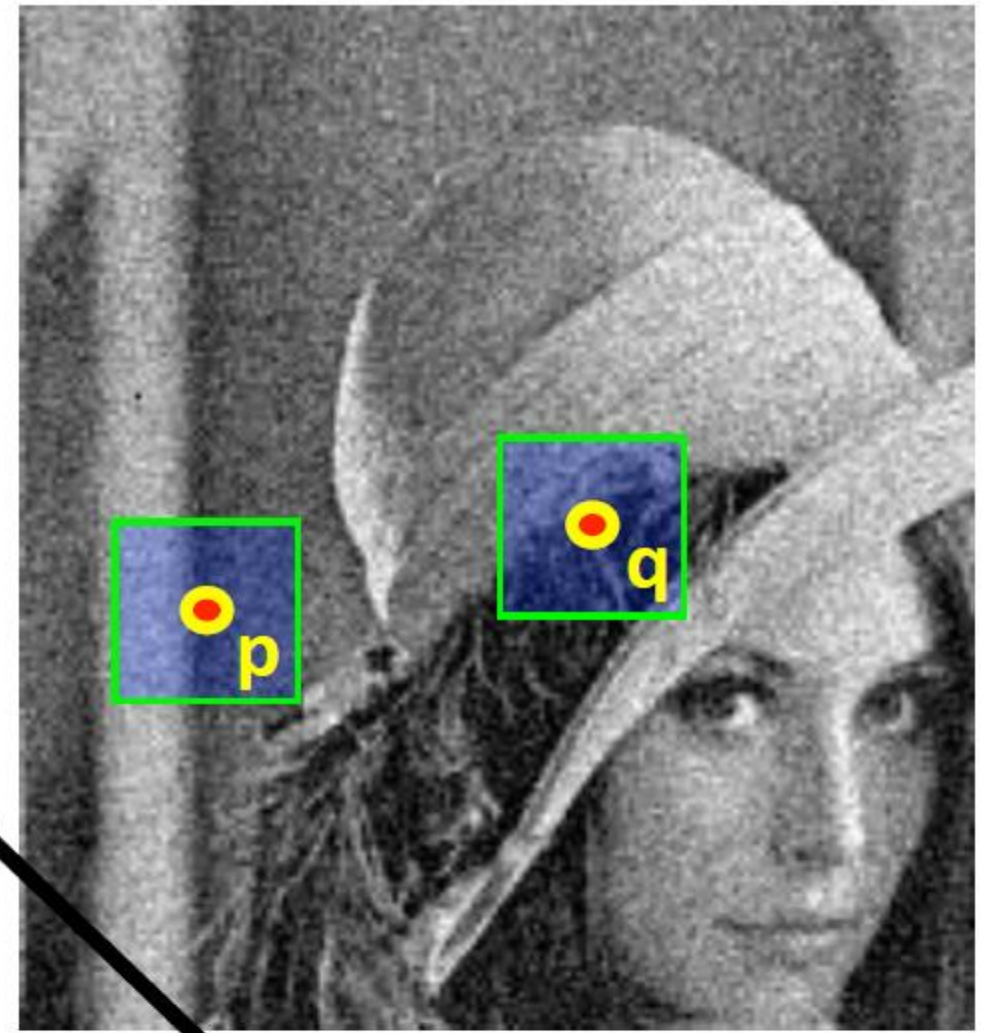
NL-Means Method: Buades (2005)

p, q neighbors define a vector distance;

$$|| V_p - V_q ||^2$$

Filter with this:

No spatial term!



$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(||\mathbf{p}-\mathbf{q}||) G_{\sigma_r}(||\vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}}||^2) I_{\mathbf{q}}$$

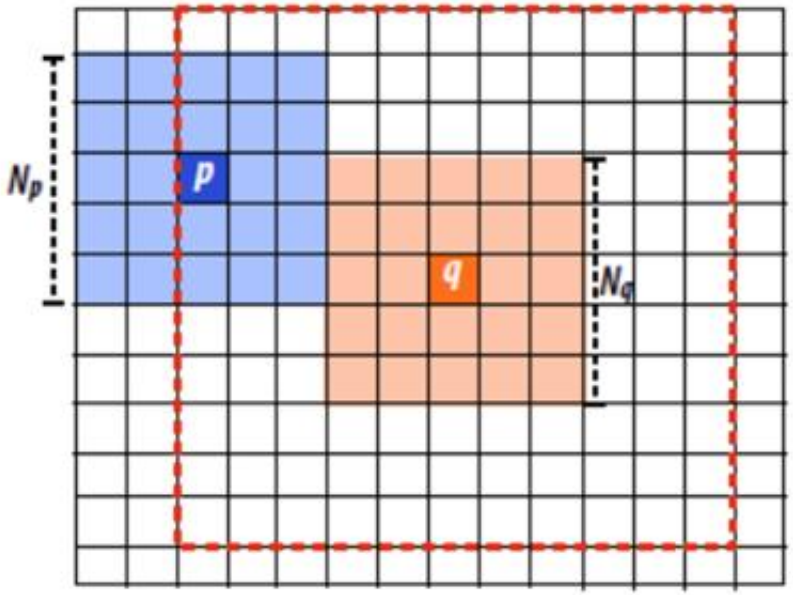
Denoising using non-local means

- **Main idea: replace pixel with average value of nearby pixels that have a similar surrounding region.**
  - **Assumption: images have repeating structure**

$$\mathrm{NL}[I](\mathrm{p}) = \sum_{\mathrm{q} \in \mathrm{S(p)}} w(\mathrm{p},\mathrm{q}) I(q)$$

All points in search region about p

$$w(\mathrm{p},\mathrm{q}) = \frac{1}{C_p} e^{-\frac{\|N_p - N_q\|^2}{h^2}}$$

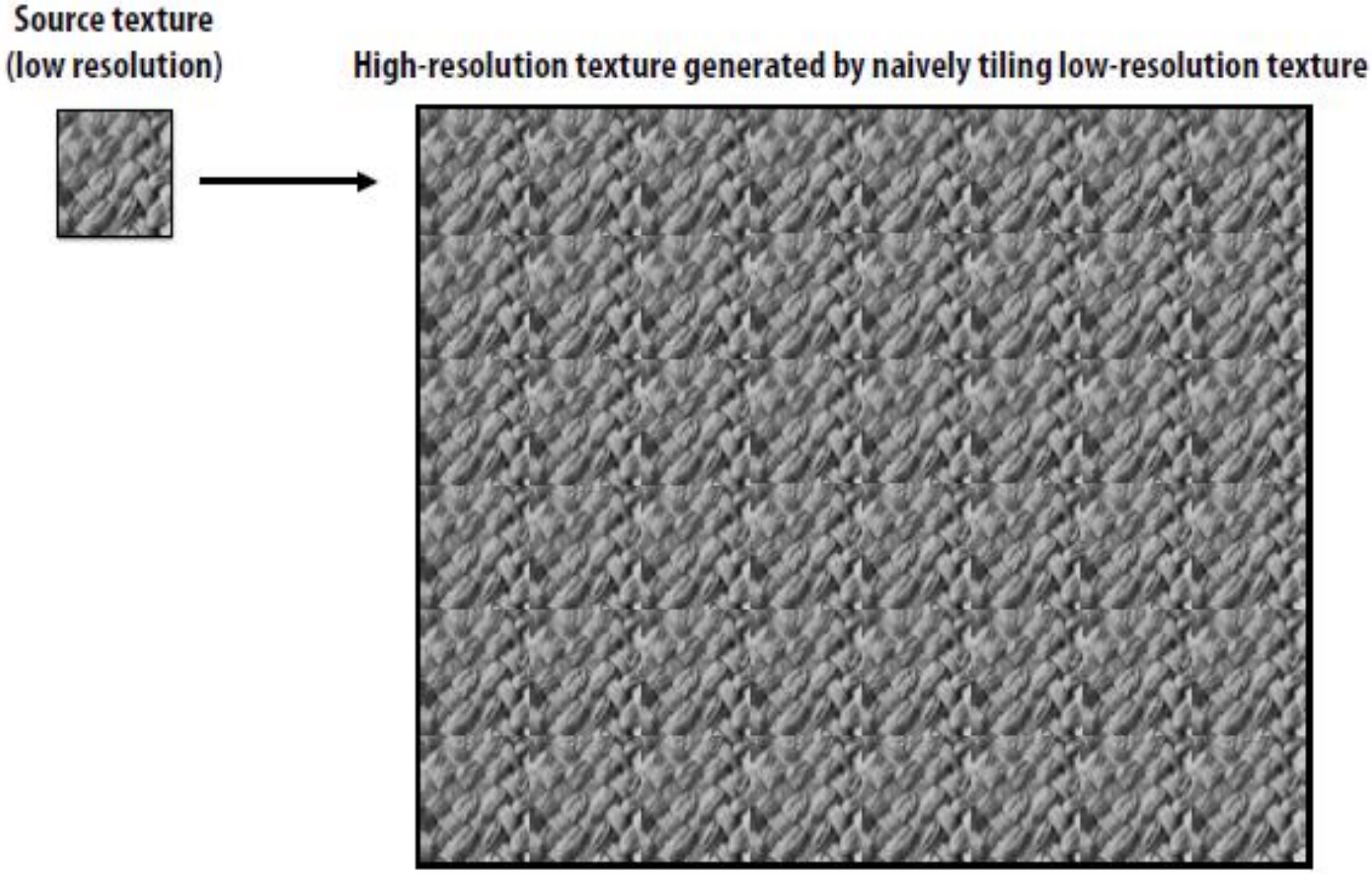


- N_p **and** P_q **are vectors of pixel values in square window around pixels** *p* **and** *q* **(highlighted regions in figure)**
- L_2 **difference between** N_p **and** N_q **= "similarity" of surrounding regions**
- C*p* **is just a normalization constant to ensure weights sum to one for pixel** *p***.**
- *S* **is the search region around** *p* **(given by dotted red line in figure)**

- Input, Gaussian, Anisotropic Diffusion, Biliteral, NLM

Texture synthesis

- Input: low-resolution texture image
- Desired output: high-resolution texture that appears “like” the input

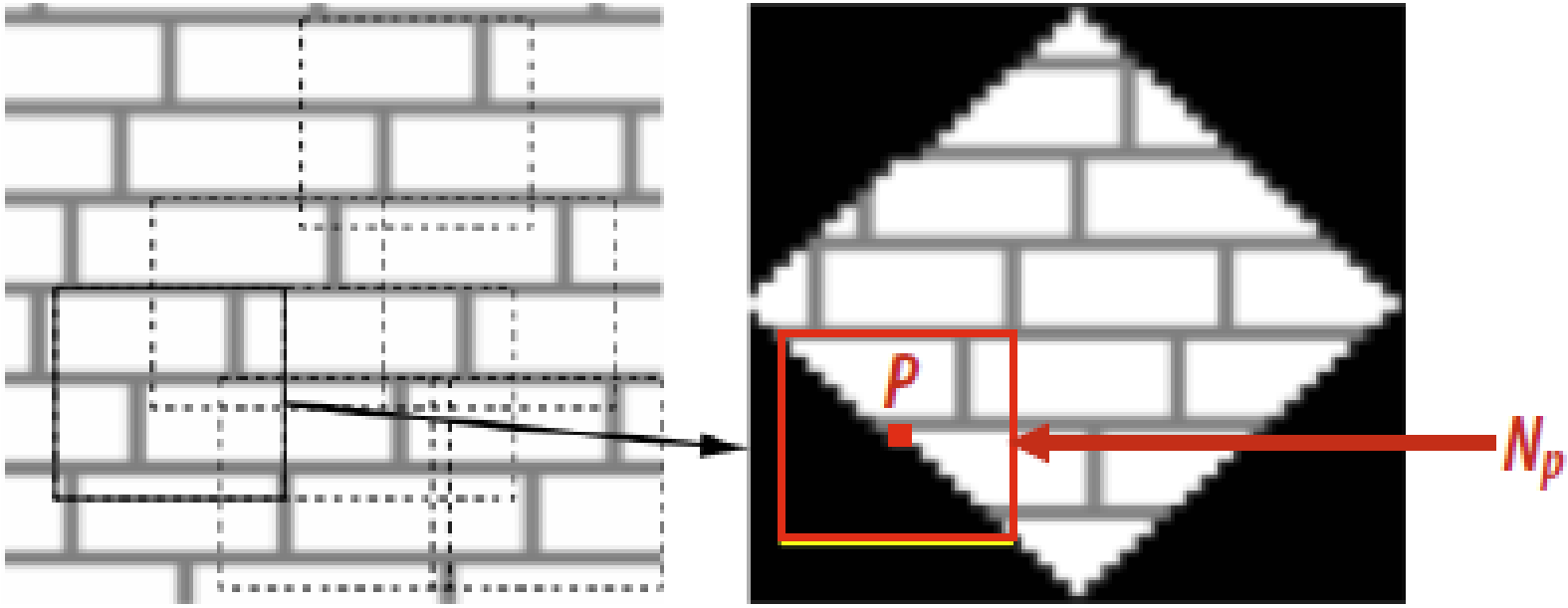


Algorithm: non-parametric texture synthesis

- **Main idea: given the NxN neighborhood *Np* around unknown pixel *p*, want a probability distribution function for possible values of *p*, given *Np*: *P*(p=X | *Np*)**

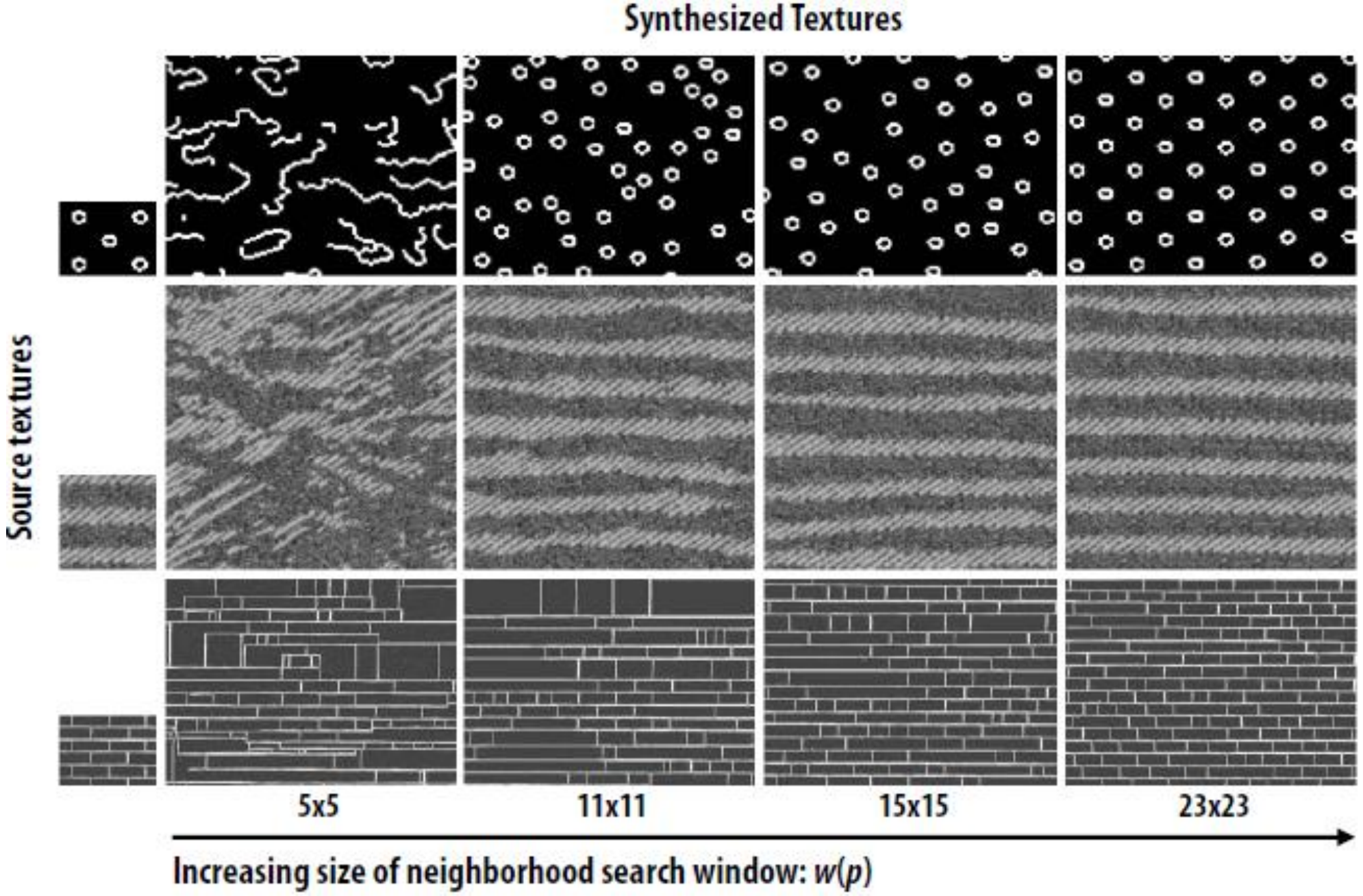
For each pixel *p* to synthesize:

1. Find other NxN patches (Nq) in the image that are most similar to Np
2. Center pixels of the closest patches are candidates for *p*
3. Randomly sample from candidates weighted by distance *d(Np,Nq)*



[Efros and Leung 99]

Non-parametric texture synthesis [Efros and Leung 99]



More texture synthesis examples [Efros and Leung 99]

Source textures

Synthesized Textures

Naive tiling solution

Image completion example

Original Image

Masked Region

Completion Result

Goal: fill in masked region with "plausible" pixel values.

See PatchMatch algorithm [Barnes 2009] for a fast randomized algorithm for finding similar patches

Image credit: [Barnes et al. 2009]

Image processing summary

- **Image processing via convolution**
  - **Different operations specified by changing weights of convolution kernel**
  - **Separable filters lend themselves to efficiency implementation as multiple 1D filters**

- **Data-driven image processing techniques**
  - **Key idea: use examples from other places in the image as priors to determine how to manipulate image**

- **To learn more: consider CMU 15-463/663: “Computational Photography”**


Any questions?

References - courses

- “A Gentle Introduction to Bilateral Filtering and its Applications” given by **Sylvain Paris**, Pierre Kornprobst, Jack Tumblin, and Frédo Durand (http://people.csail.mit.edu/sparis/bf_course/)
- Bilateral Filtering, and Non-local Means Denoising, **Erkut Erdem**

References

- Siggraph 15: An L1 Image Transform for Edge-Preserving Smoothing and Scene-Level Intrinsic Decomposition. [code]
- Siggraph 14: Bilateral texture filtering
- Siggraph 14: Fast Local Laplacian Filters: Theory and Applications. [code]