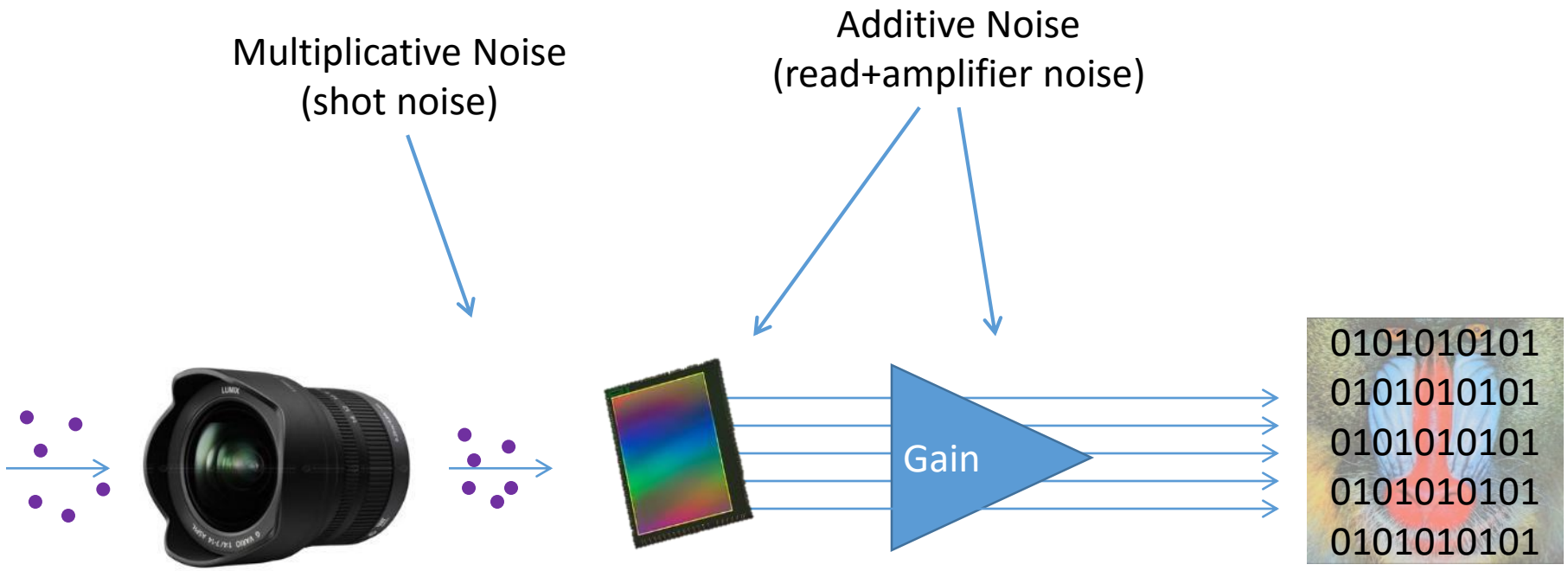
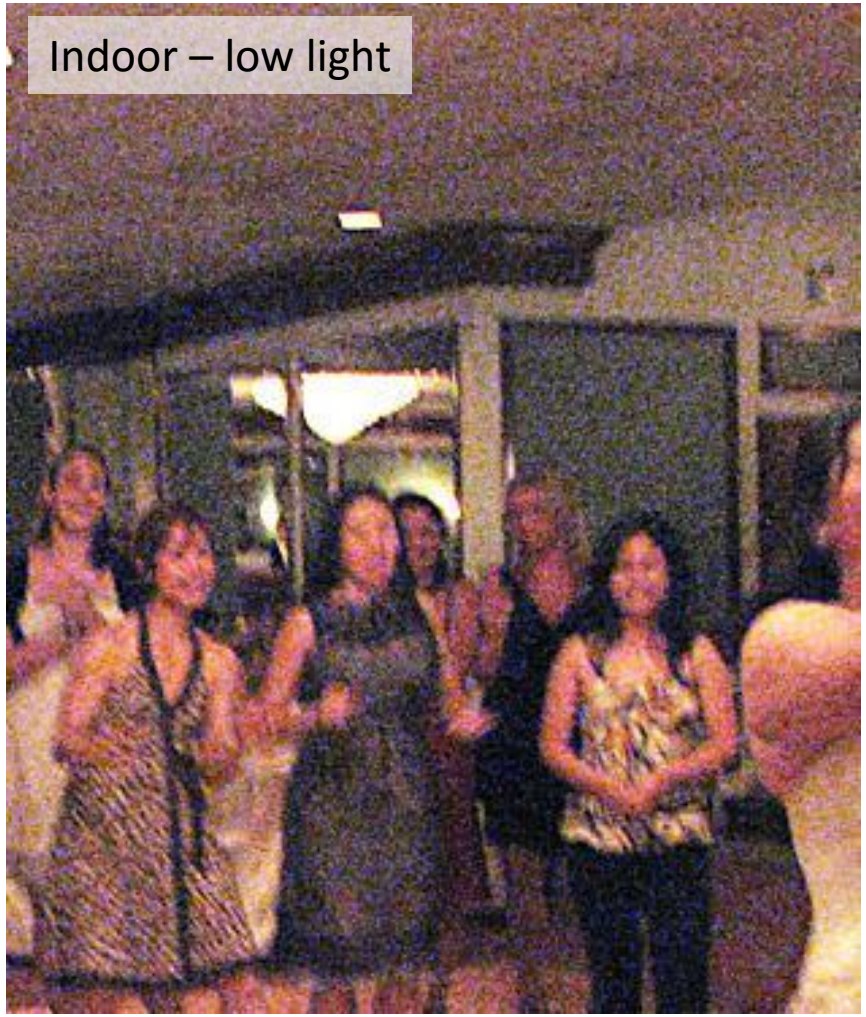
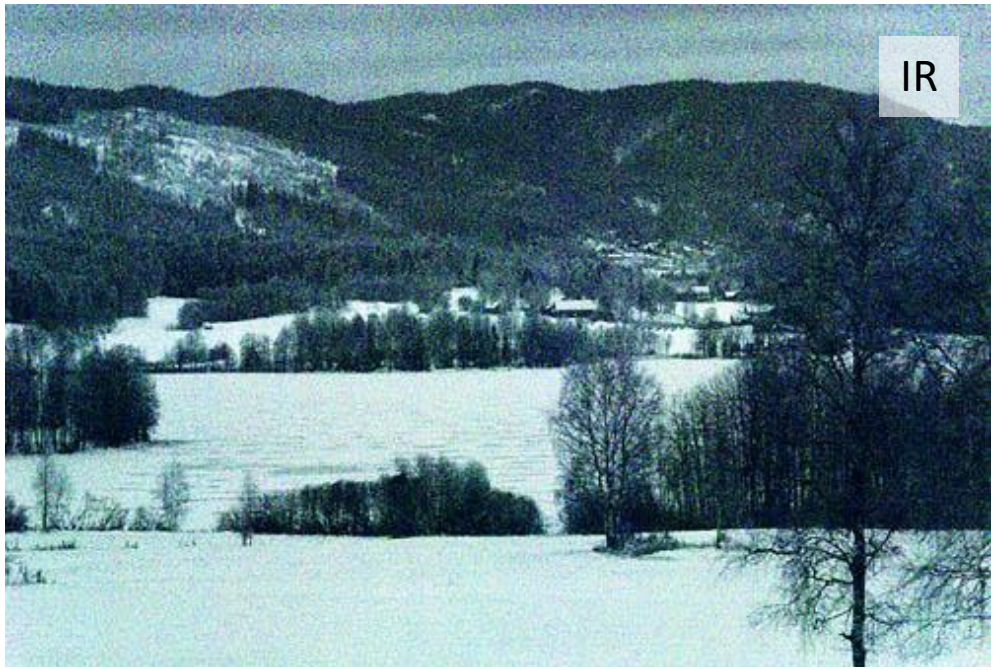
Bilateral Filtering

CPS592 – Visual Computing and Mixed Reality

Where the noise come from?




Indoor – low light


IR


US

Can we (humans) denoise?

Denoising in the Spatial Domain

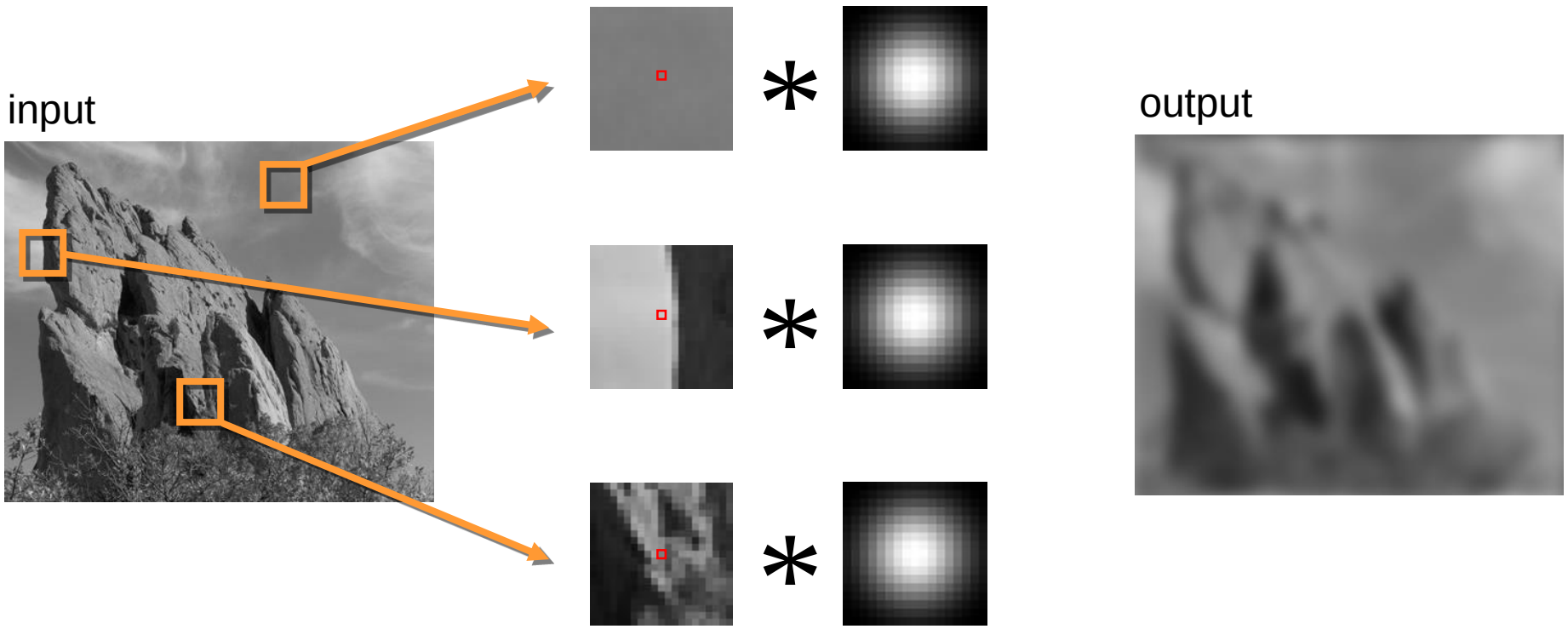
- The “classical” assumption:
  Images are piecewise constant

- Neighboring pixels are highly correlated

⇒ Denoise = “Average nearby pixels” (filtering)

Slides taken from Sylvain Paris, Siggraph 2007

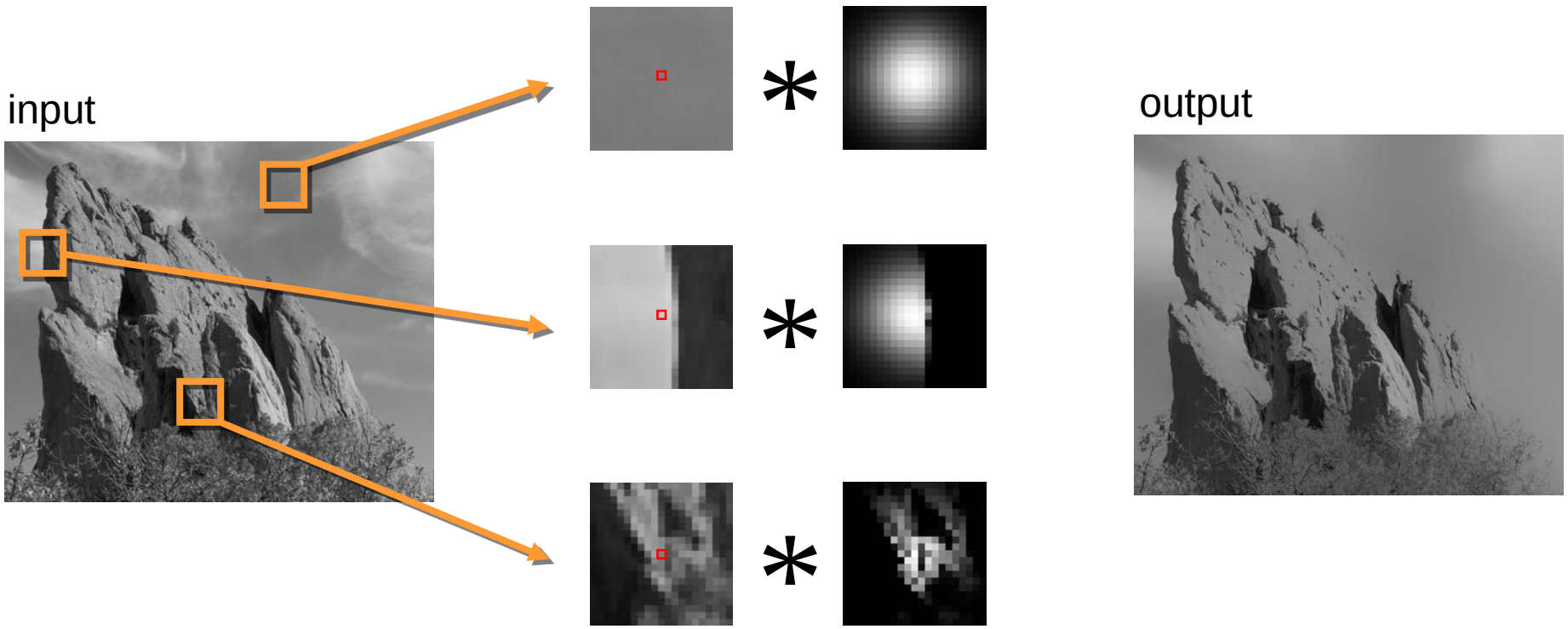
Gaussian Smoothing



Same Gaussian kernel everywhere
Averages across edges ⇒ blur

Slides taken from Sylvain Paris, Siggraph 2007

Bilateral Filtering



Kernel shape depends on image content
Avoids averaging across edges

Bilateral Filter Definition: an Additional Edge Term

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

new

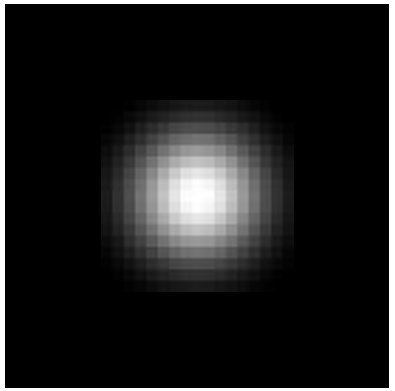
not new

new

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

normalization factor

space weight



range weight

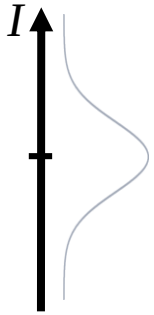
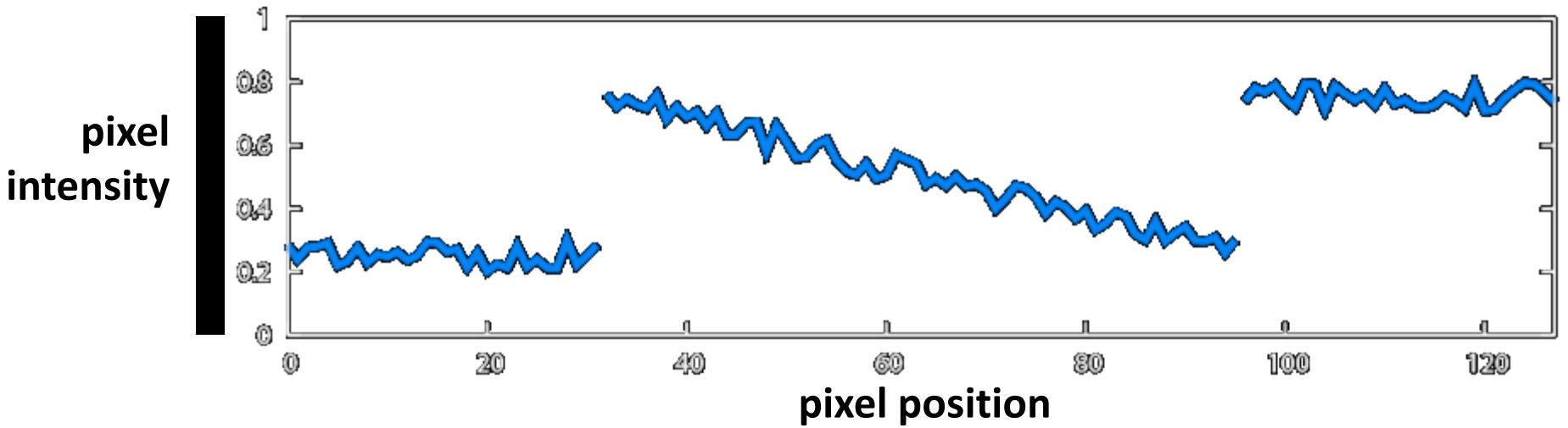


Illustration a 1D Image

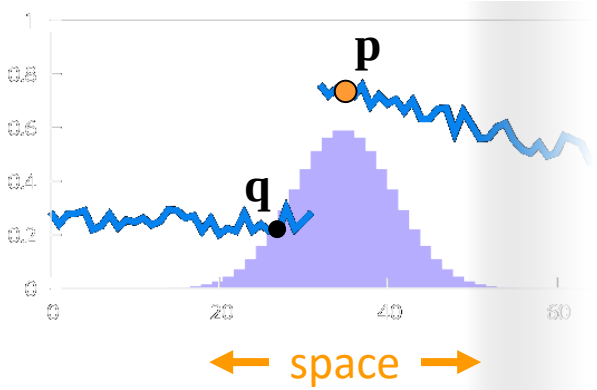
- 1D image = line of pixels

- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

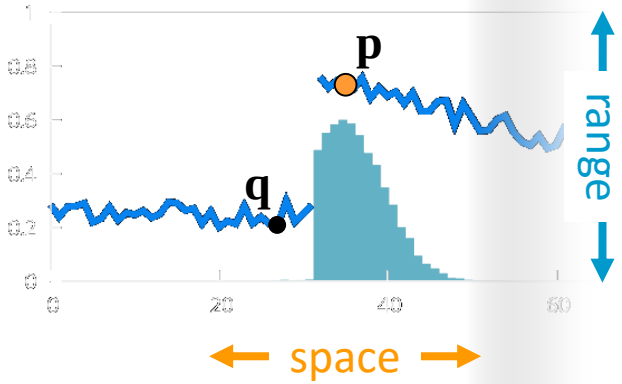


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}(\|\mathbf{p}-\mathbf{q}\|) I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



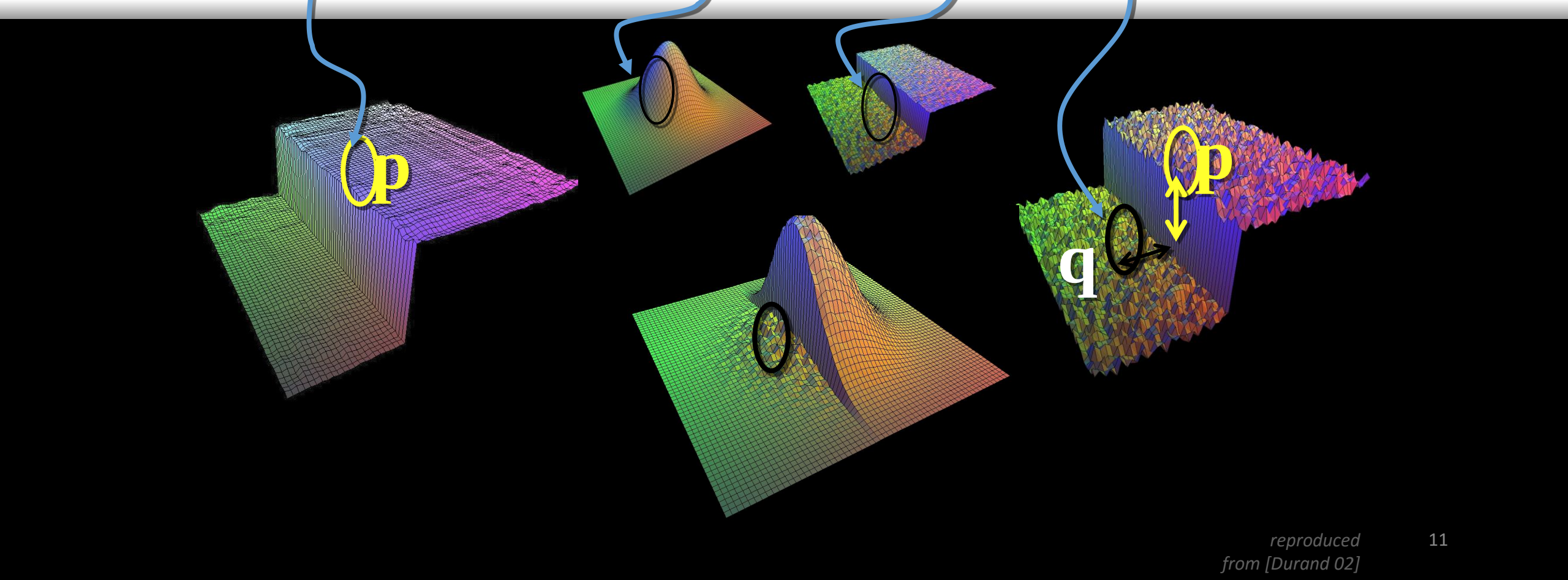
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) I_{\mathbf{q}}$$

space range

normalization

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

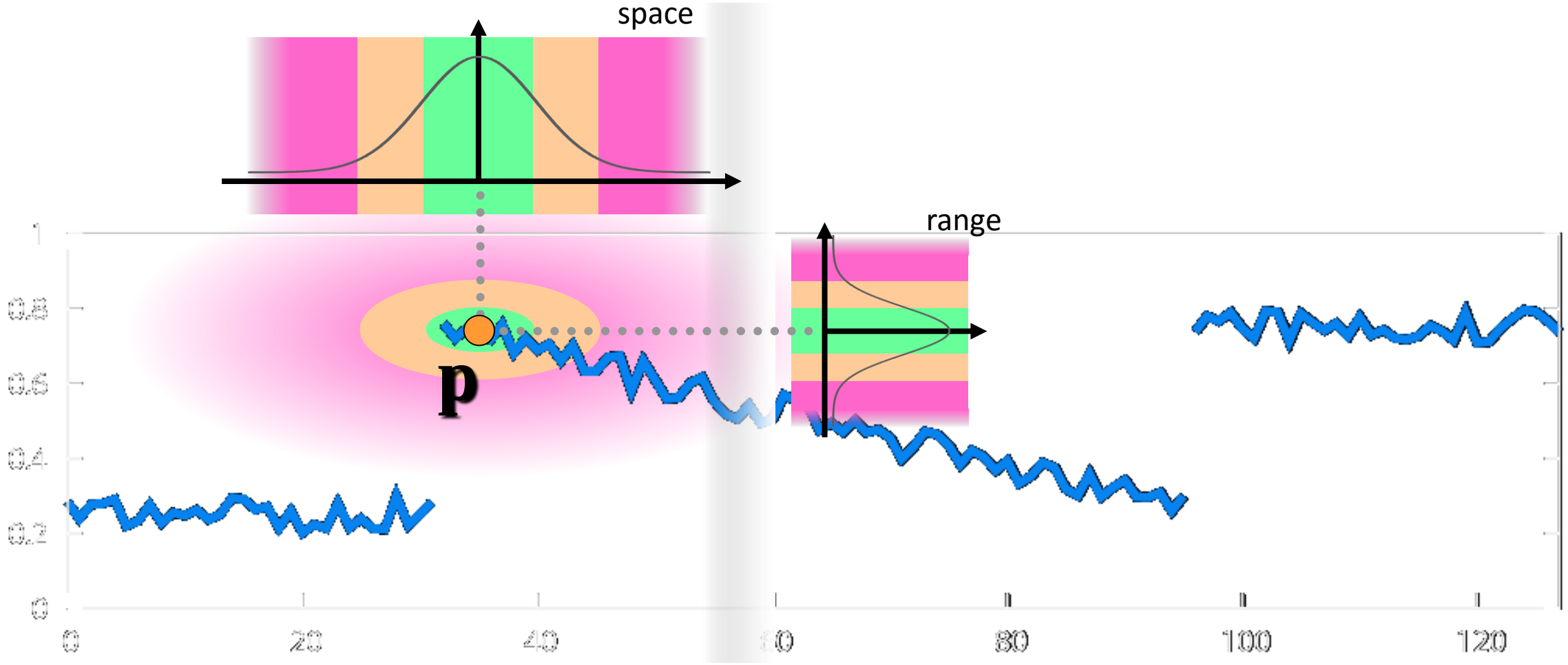
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

- 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.



Exploring the Parameter Space

input

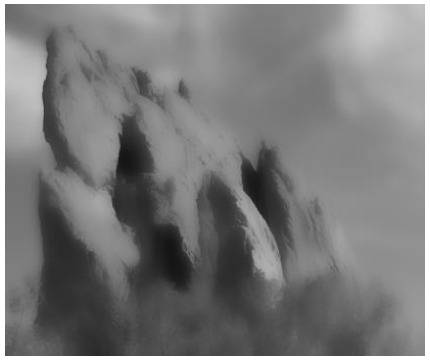
$\sigma_r = 0.1$

$\sigma_r = 0.25$

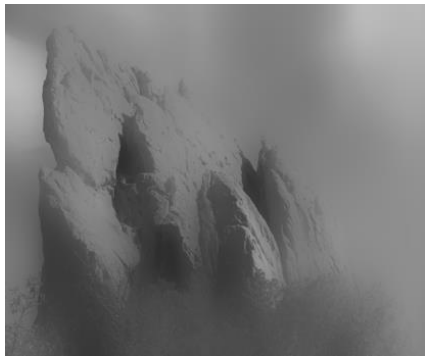
$\sigma_r = \infty$ (Gaussian blur)

$\sigma_s = 2$

$\sigma_s = 6$



$\sigma_s = 18$



Varying the Range Parameter

input

$\sigma_r = 0.1$ $\sigma_r = 0.25$ $\sigma_r = \infty$

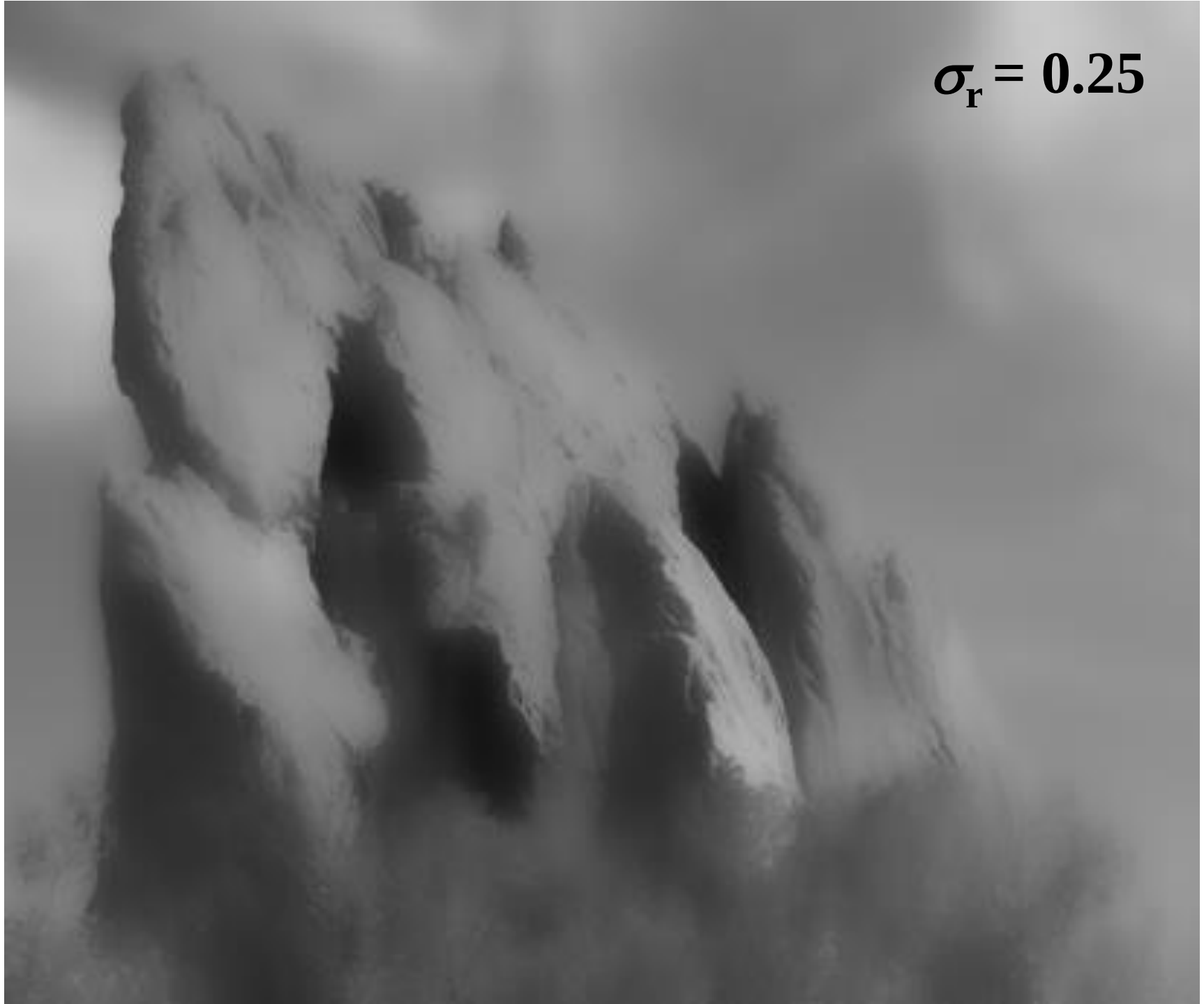
$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$


input

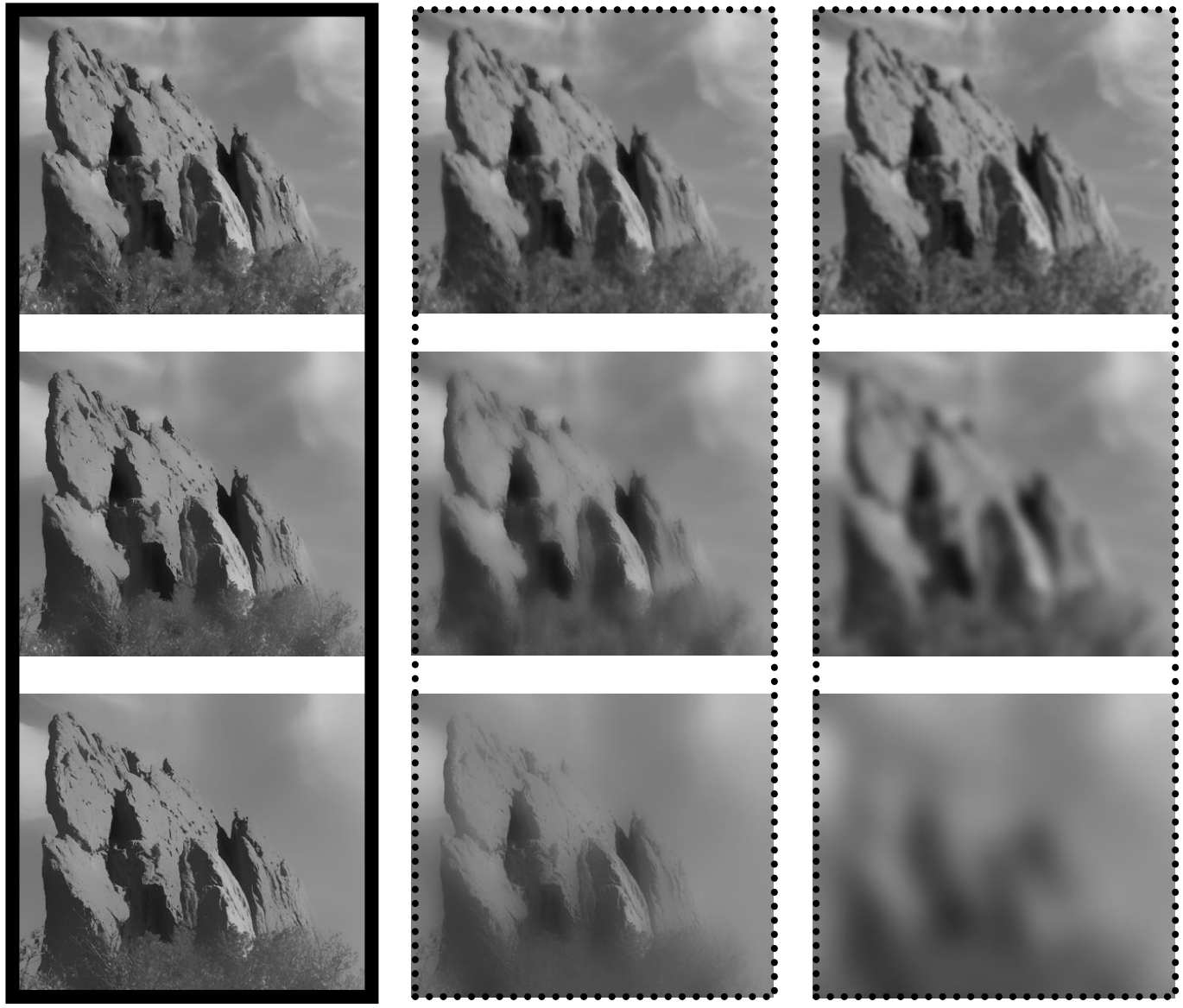

$\sigma_r = 0.1$
(Gaussian blur)


$\sigma_r = 0.25$


$\sigma_r = \infty$

Varying the Space Parameter

input




input


$\sigma_s = 2$
Bilateral Filtering


$\sigma_s = 6$


$\sigma_s = 18$

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

Implementation

- Brute-force Implementation
- Separable Kernel [Pham and Van Vliet 05]
- Box Kernel [Weiss 06]

Brute-force Implementation

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

For each pixel $\mathbf{p}$

For each pixel $\mathbf{q}$

Compute $G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) I_{\mathbf{q}}$

8 megapixel photo: 64,000,000,000,000 iterations!

VERY SLOW!

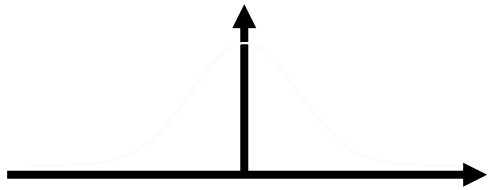
More than 10 minutes per image

Complexity

- Complexity = *"how many operations are needed, how this number varies"*
- S = space domain = set of pixel positions
- $|S|$ = cardinality of S = number of pixels
  - In the order of 1 to 10 millions
- Brute-force implementation: $O(|S|^2)$

Better Brute-force Implementation

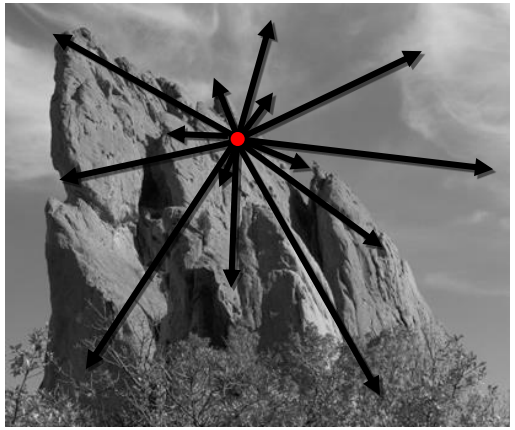
Idea: Far away pixels are negligible



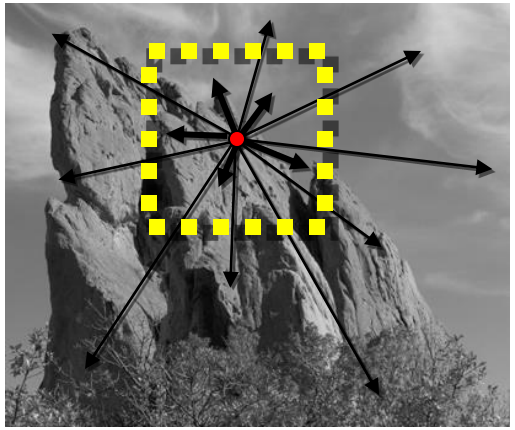
For each pixel $\mathbf{p}$

a. For each pixel $\mathbf{q}$ such that $\| \mathbf{p} - \mathbf{q} \| < cte \times \sigma_s$

looking at all pixels



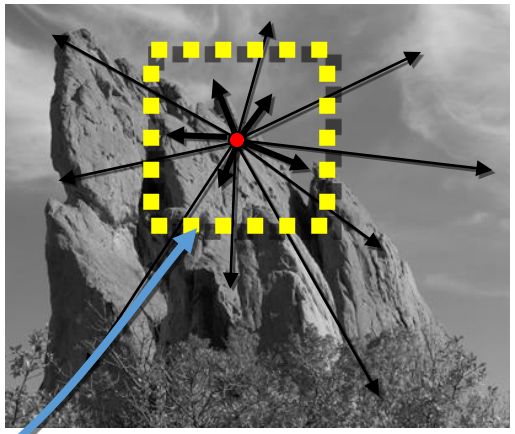
looking at neighbors only



Discussion

- Complexity:

$$O(|S| \times \sigma_s^2)$$



neighborhood area

- Fast for small kernels: σ_s ~ 1 or 2 pixels

- BUT: slow for larger kernels

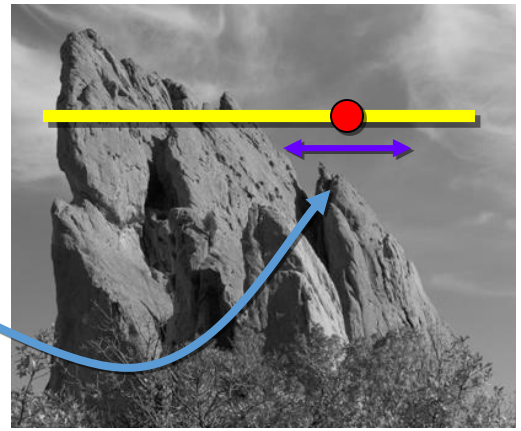
Separable Kernel

[Pham and Van Vliet 05]

- Strategy: filter the rows then the columns

- Two “cheap” 1D filters
  instead of an “expensive” 2D filter

Discussion



- Complexity: $O(|S| \times \sigma_s)$
  - Fast for small kernels (<10 pixels)
- Approximation: BF kernel not separable
  - Satisfying at strong edges and uniform areas
  - Can introduce visible streaks on textured regions


input

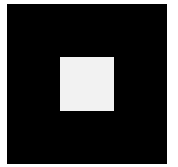

brute-force
implementation


separable kernel
mostly OK,
some visible artifacts
(streaks)

Box Kernel [Weiss 06]

- Bilateral filter with a square box window [Yarovlasky 85]

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



box window

restrict the sum

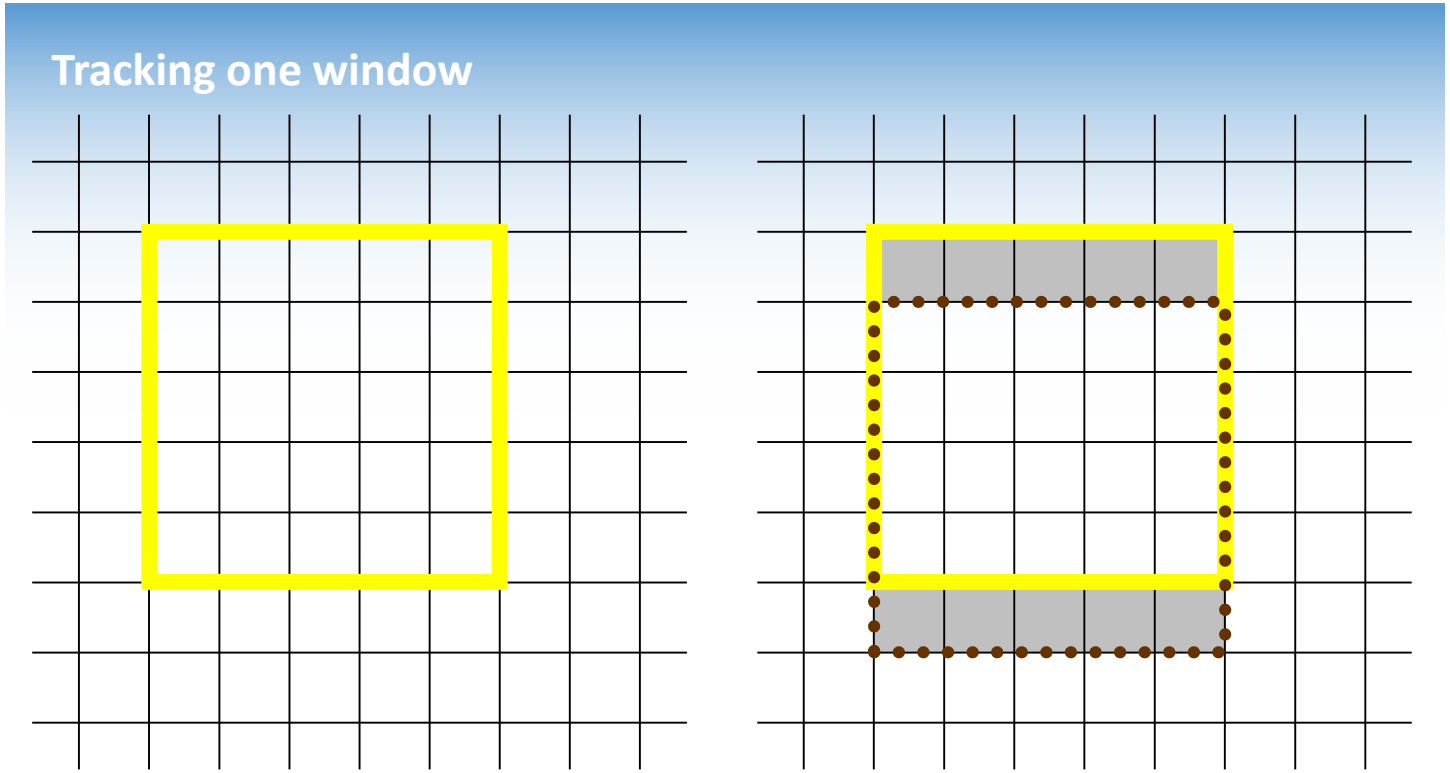
$$Y[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in B_{\sigma_s}} G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

independent of position $\mathbf{q}$

- The bilateral filter can be computed only from the list of pixels in a square neighborhood.

Box Kernel [Weiss 06]

- Idea: fast histograms of square windows



input:
full histogram is known

update:
add one line, remove one line

Discussion

- Complexity: $O(|S| \times \log \sigma_s)$
  - always fast
- Only single-channel images
- Visually satisfying results (no artifacts)
  - 3 passes to remove artifacts due to box windows (Mach bands)

1 iteration

3 iterations


input


brute-force
implementation


box kernel
visually different,
yet no artifacts

Other result

noisy image

naïve denoising
Gaussian blur

Bilateral filtering

Smoothing an image without blurring its edges.

Q&A