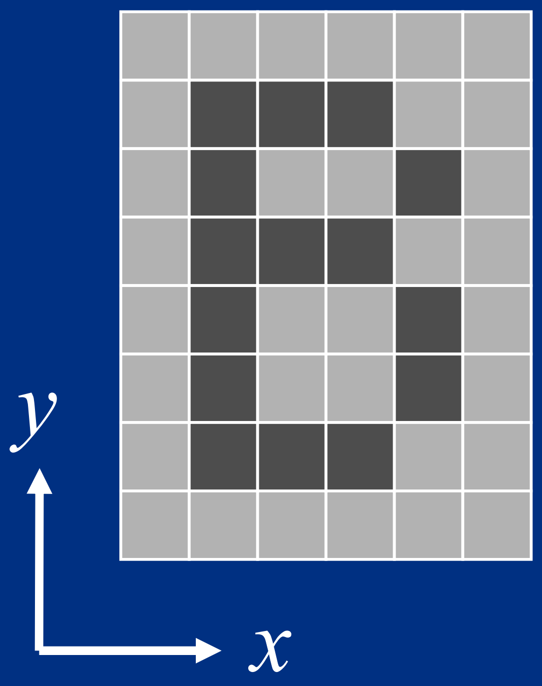
A Gentle Introduction to Bilateral Filtering and its Applications

SIGGRAPH2007

Naïve Image Smoothing: Gaussian Blur

Sylvain Paris – MIT CSAIL

Notation and Definitions

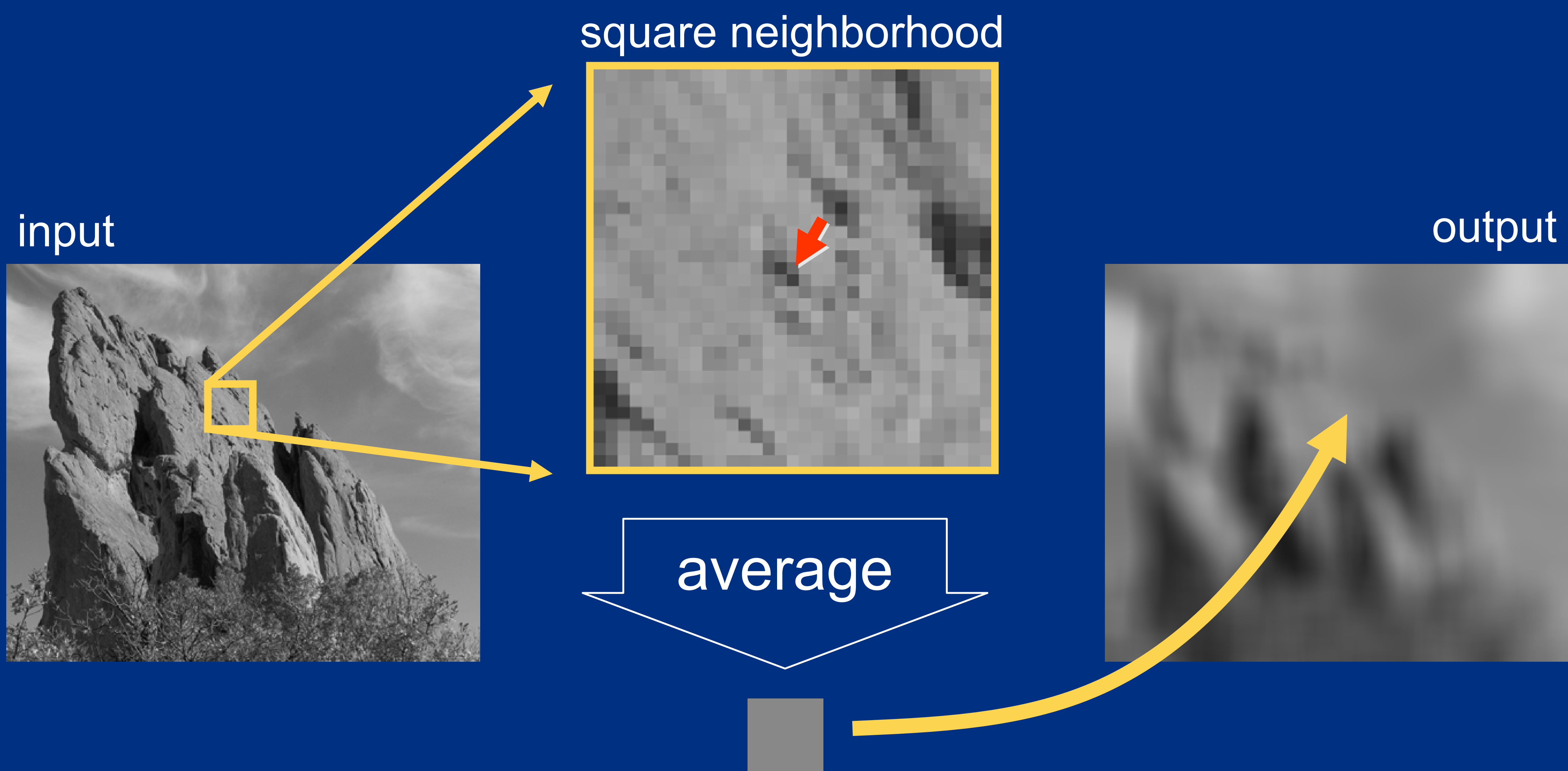


- Image = 2D array of pixels
- Pixel = intensity (scalar) or color (3D vector)
- $I_{\mathbf{p}}$ = value of image I at position: $\mathbf{p} = (p_x , p_y)$
- $F [I]$ = output of filter F applied to image I

Strategy for Smoothing Images

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

Box Average



Equation of Box Average

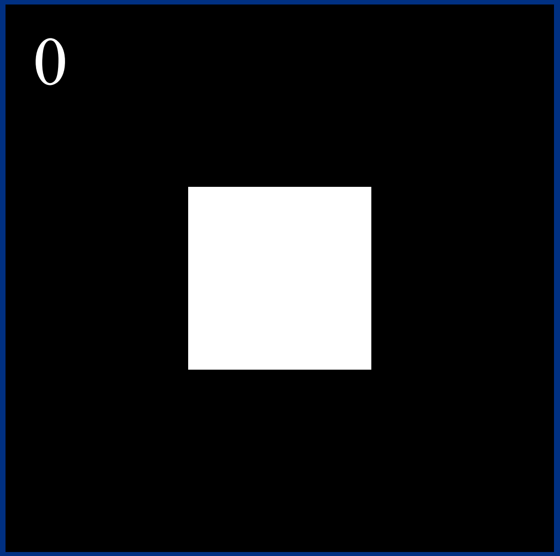
$$BA[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} B_\sigma(\mathbf{p} - \mathbf{q})\, I_{\mathbf{q}}$$

result at pixel $\mathbf{p}$

sum over all pixels $\mathbf{q}$

intensity at pixel $\mathbf{q}$

normalized box function



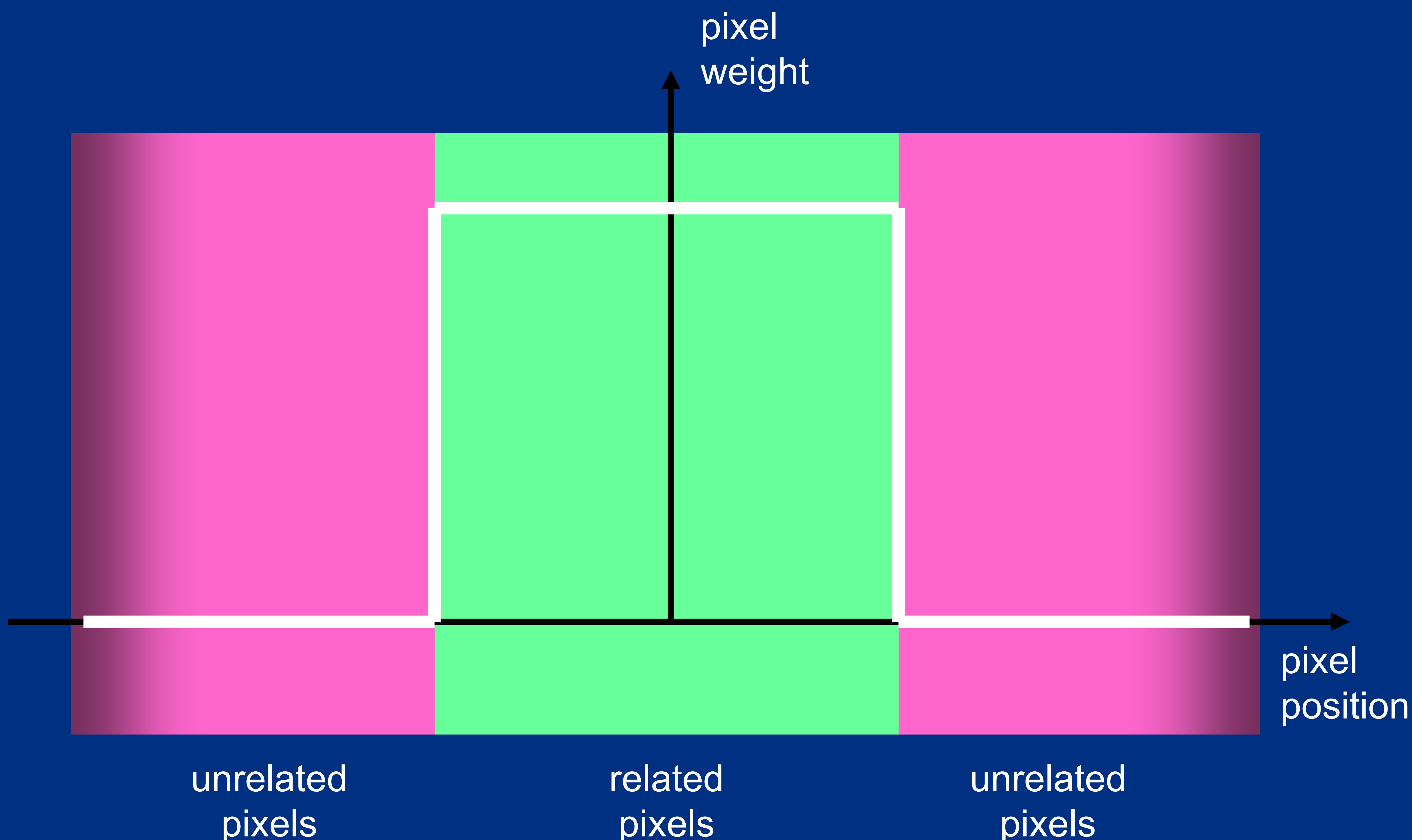
Square Box Generates Defects

- Axis-aligned streaks
- Blocky results

output

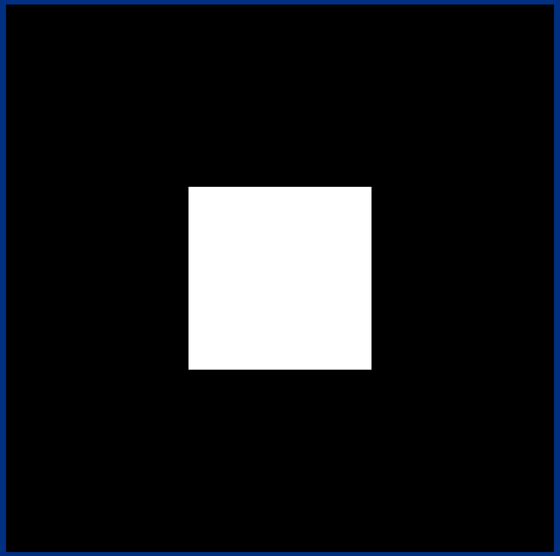
input

Box Profile

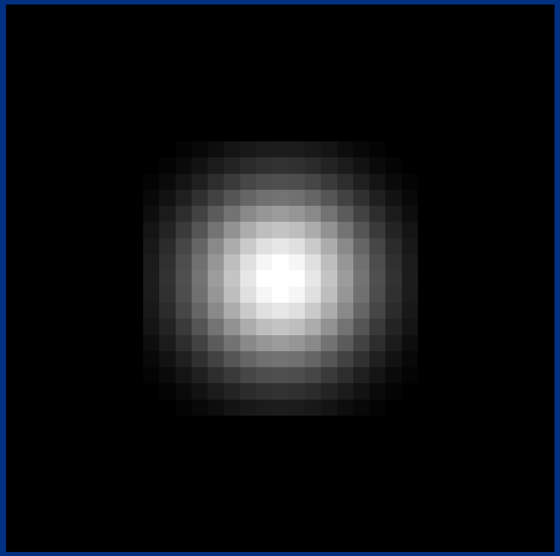

pixel
weight
pixel
position
unrelated
pixels
related
pixels
unrelated
pixels

Strategy to Solve these Problems

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

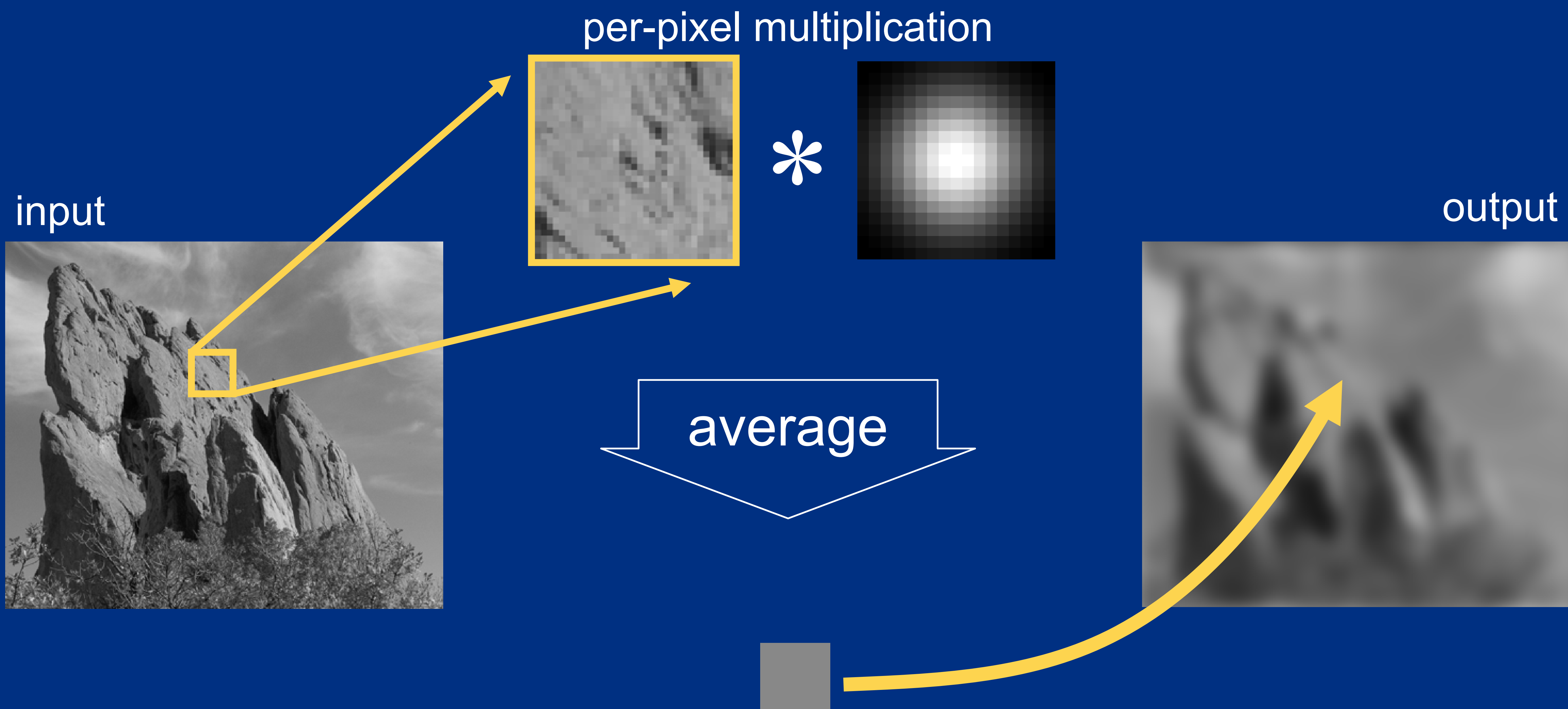


box window



Gaussian window

Gaussian Blur




input


box average

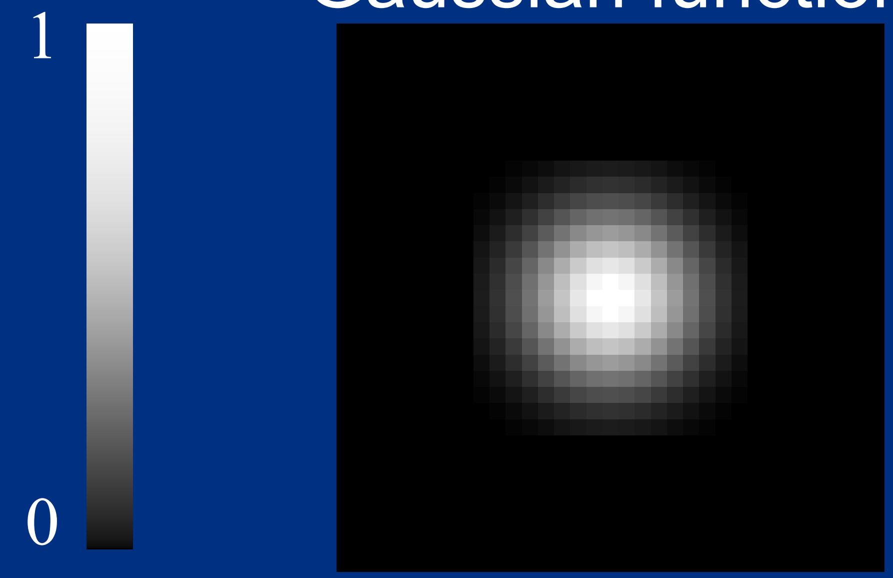

Gaussian blur

Equation of Gaussian Blur

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

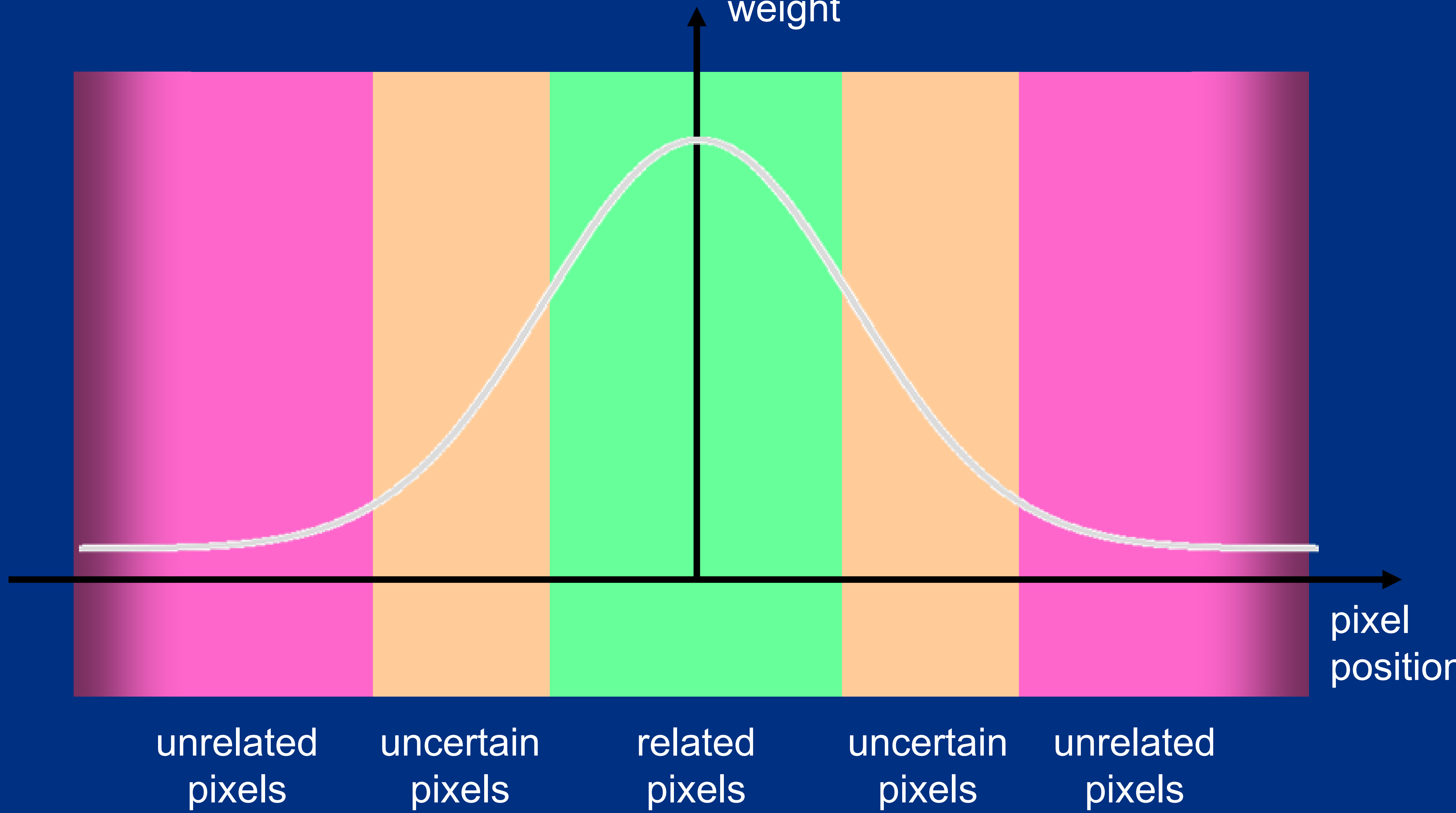
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} \boxed{G_\sigma(\|\mathbf{p}-\mathbf{q}\|)}\, I_{\mathbf{q}}$$

normalized
Gaussian function



Gaussian Profile

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

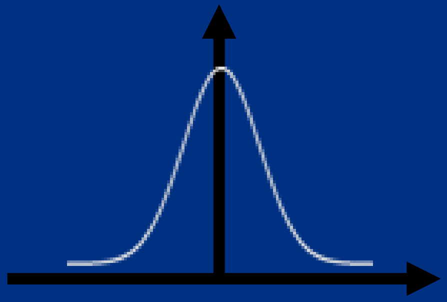


Spatial Parameter

input

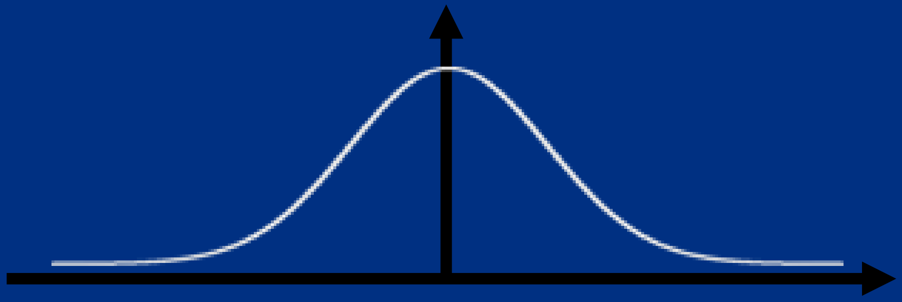
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}\left(\|\mathbf{p}-\mathbf{q}\|\right) I_{\mathbf{q}}$$

size of the window

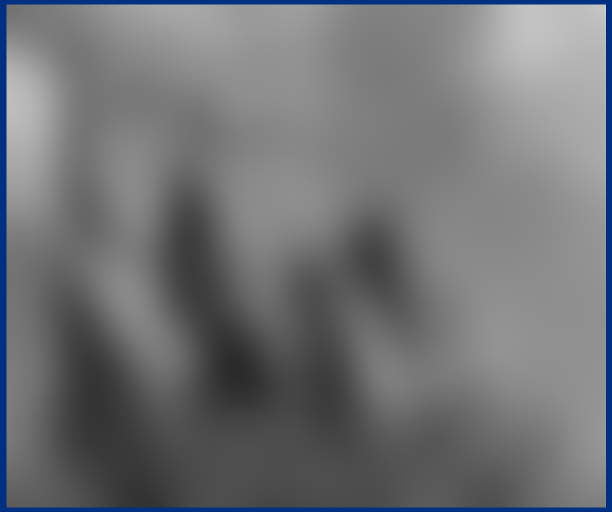


small σ

limited smoothing



large σ



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

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

input

output

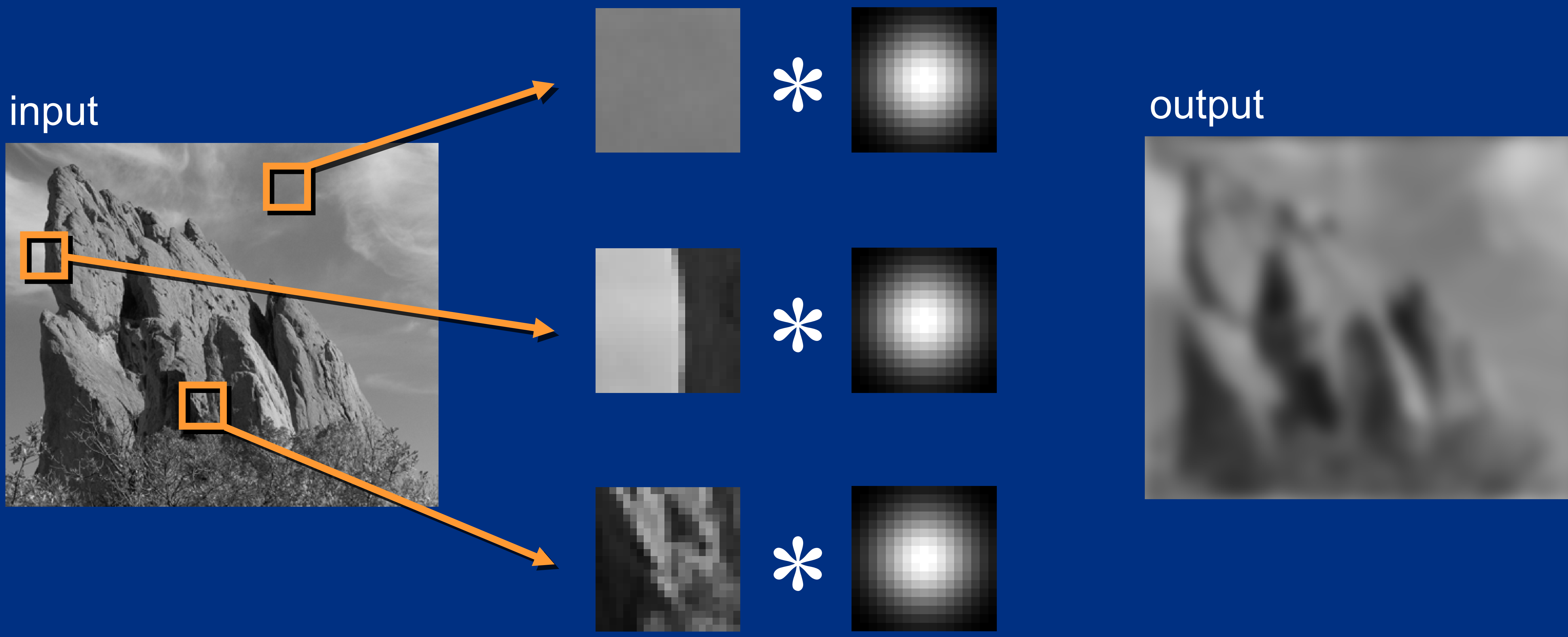
A Gentle Introduction to Bilateral Filtering and its Applications

SIGGRAPH2007

"Fixing the Gaussian Blur": the Bilateral Filter

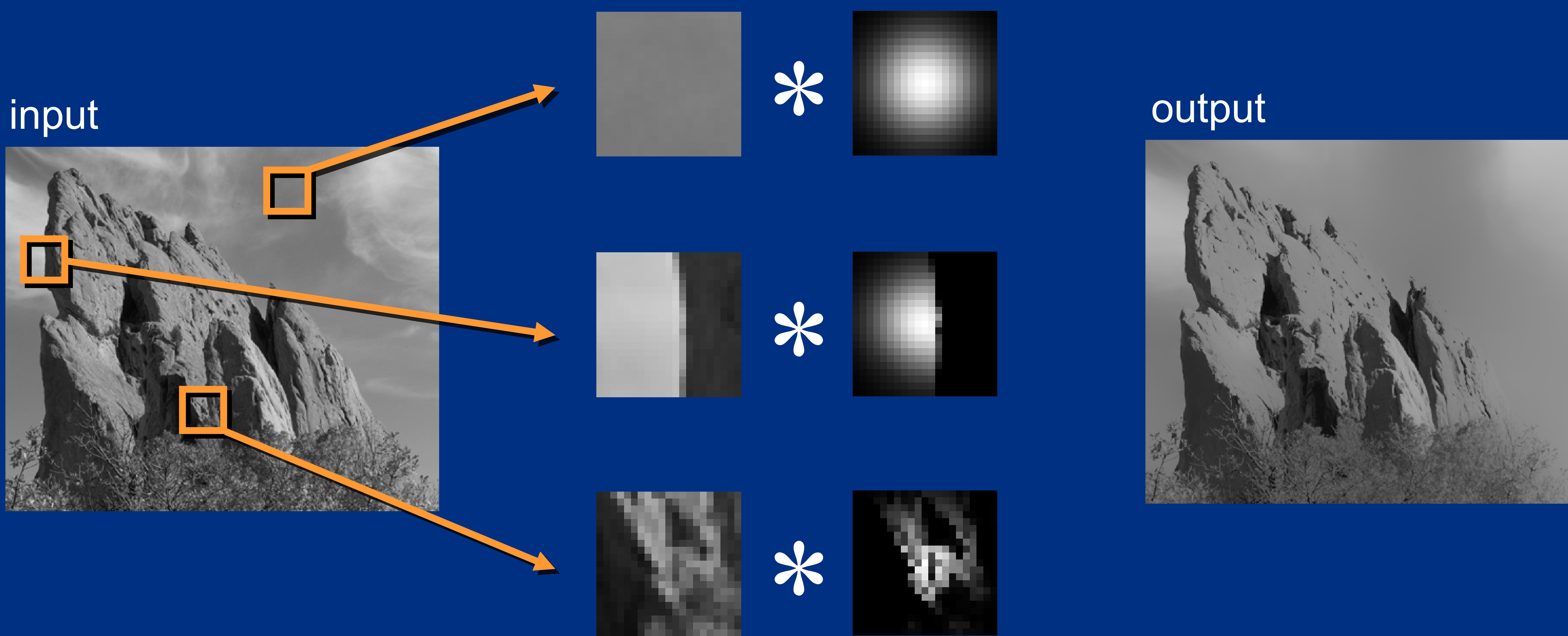
Sylvain Paris – MIT CSAIL

Blur Comes from Averaging across Edges



Same Gaussian kernel everywhere.

Bilateral Filter [Aurich 95, Smith 97, Tomasi 98]
No Averaging across Edges



The kernel shape depends on the image content.

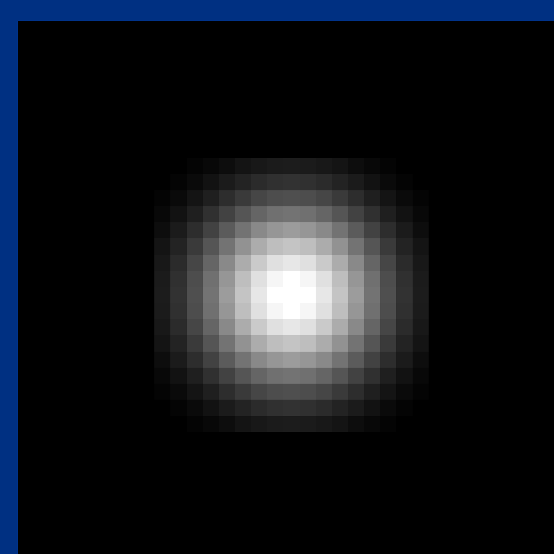
Bilateral Filter Definition: an Additional Edge Term

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

$$BF[I]_{\mathbf{p}} = \overbrace{\boxed{\frac{1}{W_{\mathbf{p}}}}}^{\text{new}}\ \sum_{\mathbf{q}\in S} \overbrace{\boxed{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)}}^{\text{not new}}\ \overbrace{\boxed{G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}}^{\text{new}}\ I_{\mathbf{q}}$$

normalization factor

space weight



range weight

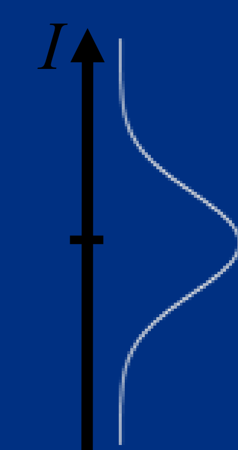
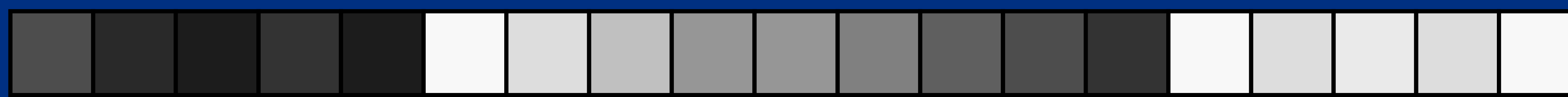
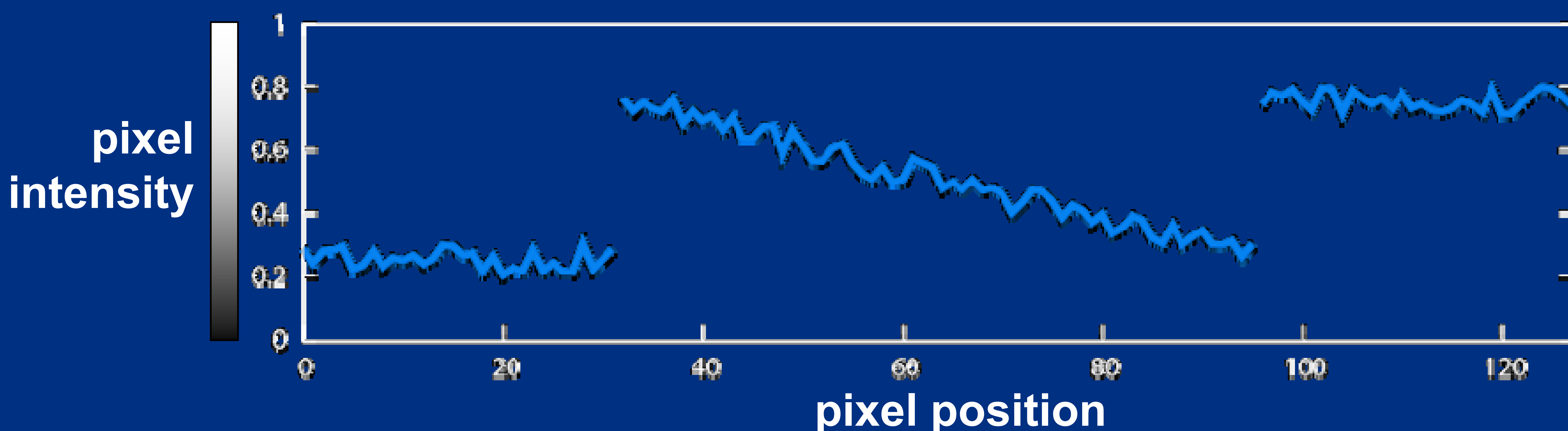


Illustration a 1D Image

- 1D image = line of pixels

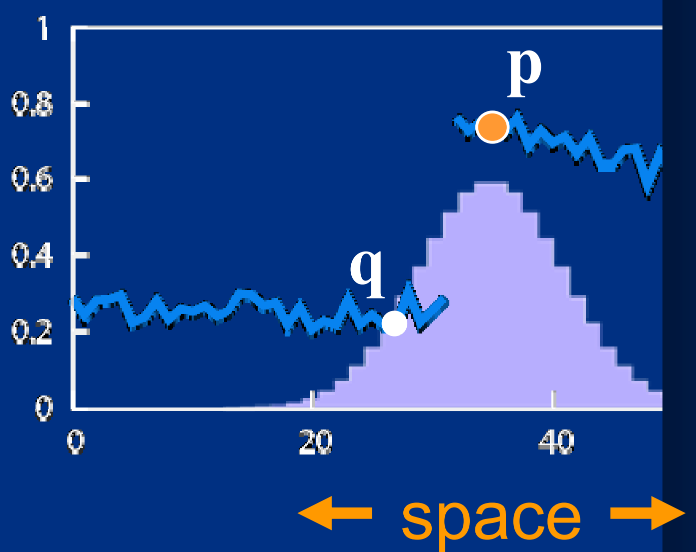


- Better visualized as a plot



Gaussian Blur and Bilateral Filter

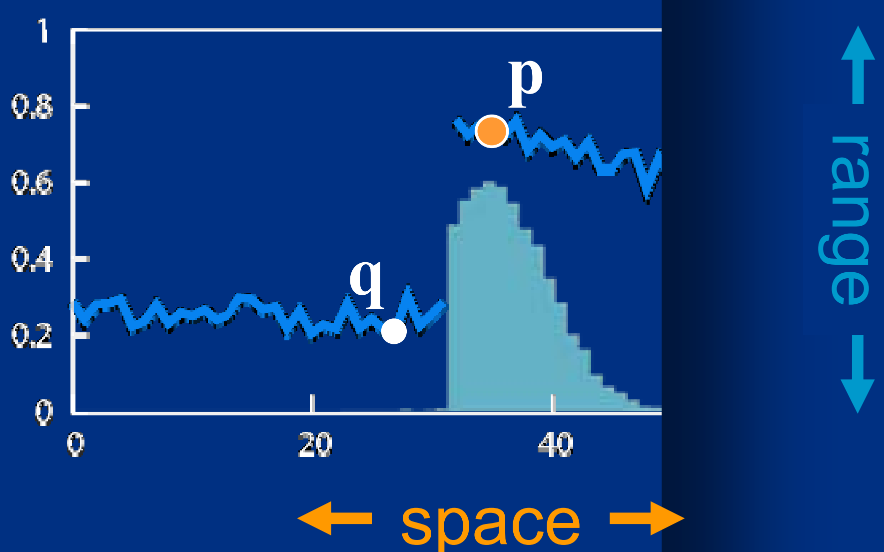
Gaussian blur



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

Bilateral filter

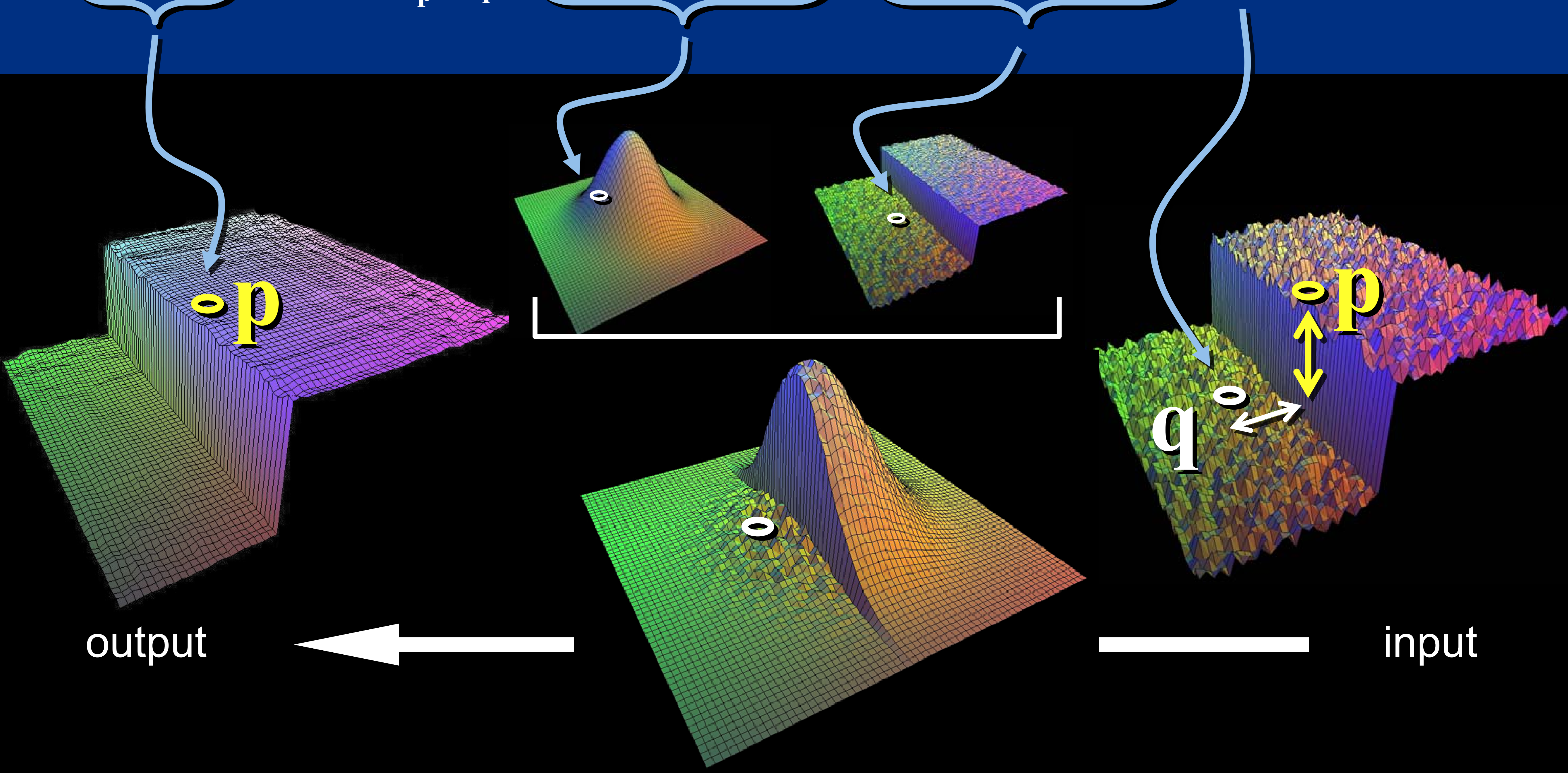
[Aurich 95, Smith 97, Tomasi 98]



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

Bilateral Filter on a Height Field

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



reproduced from [Durand 02]

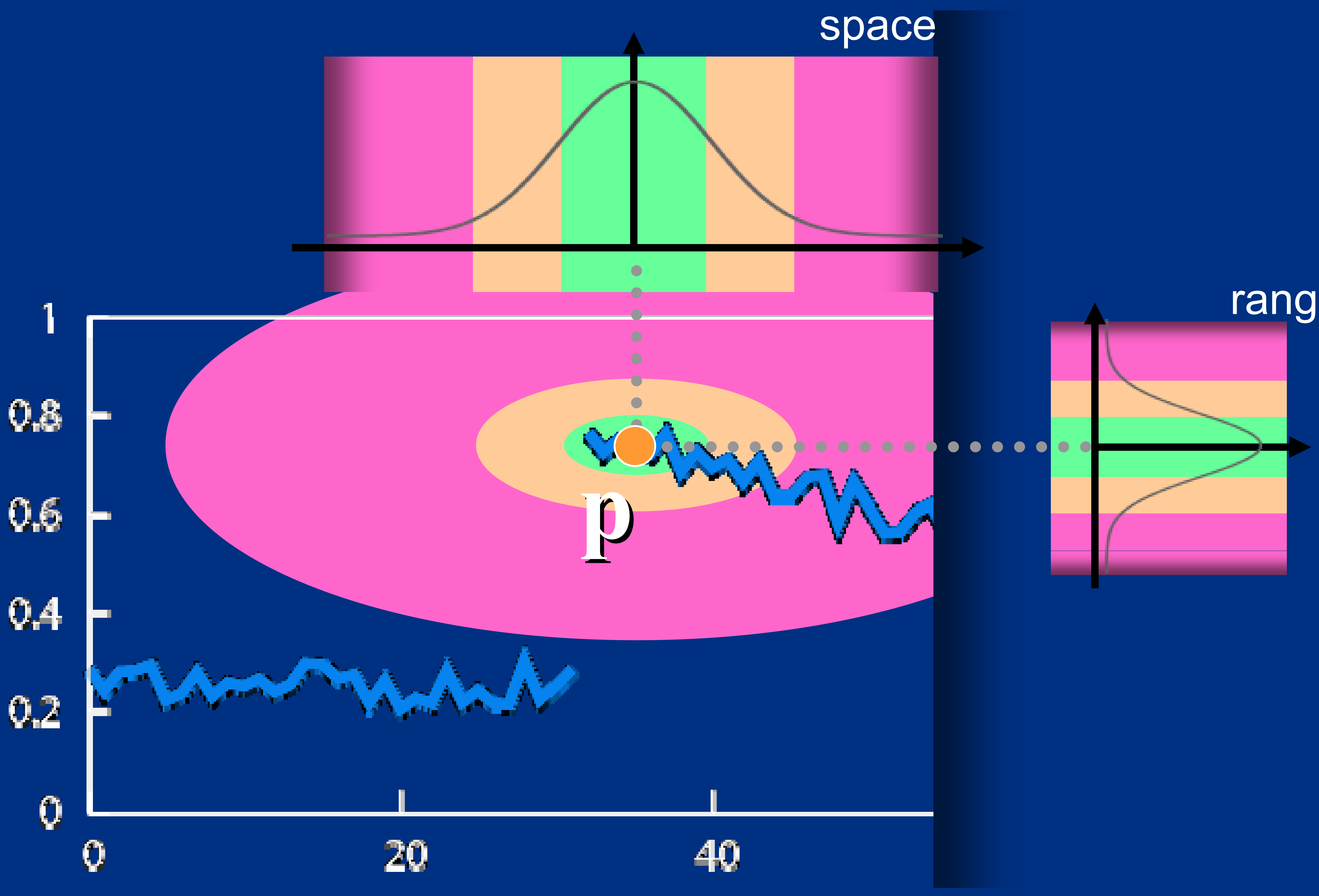
Space and Range Parameters

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

- space σ_s : spatial extent of the kernel, size of the considered neighborhood.
- range σ_r : “minimum” amplitude of an edge

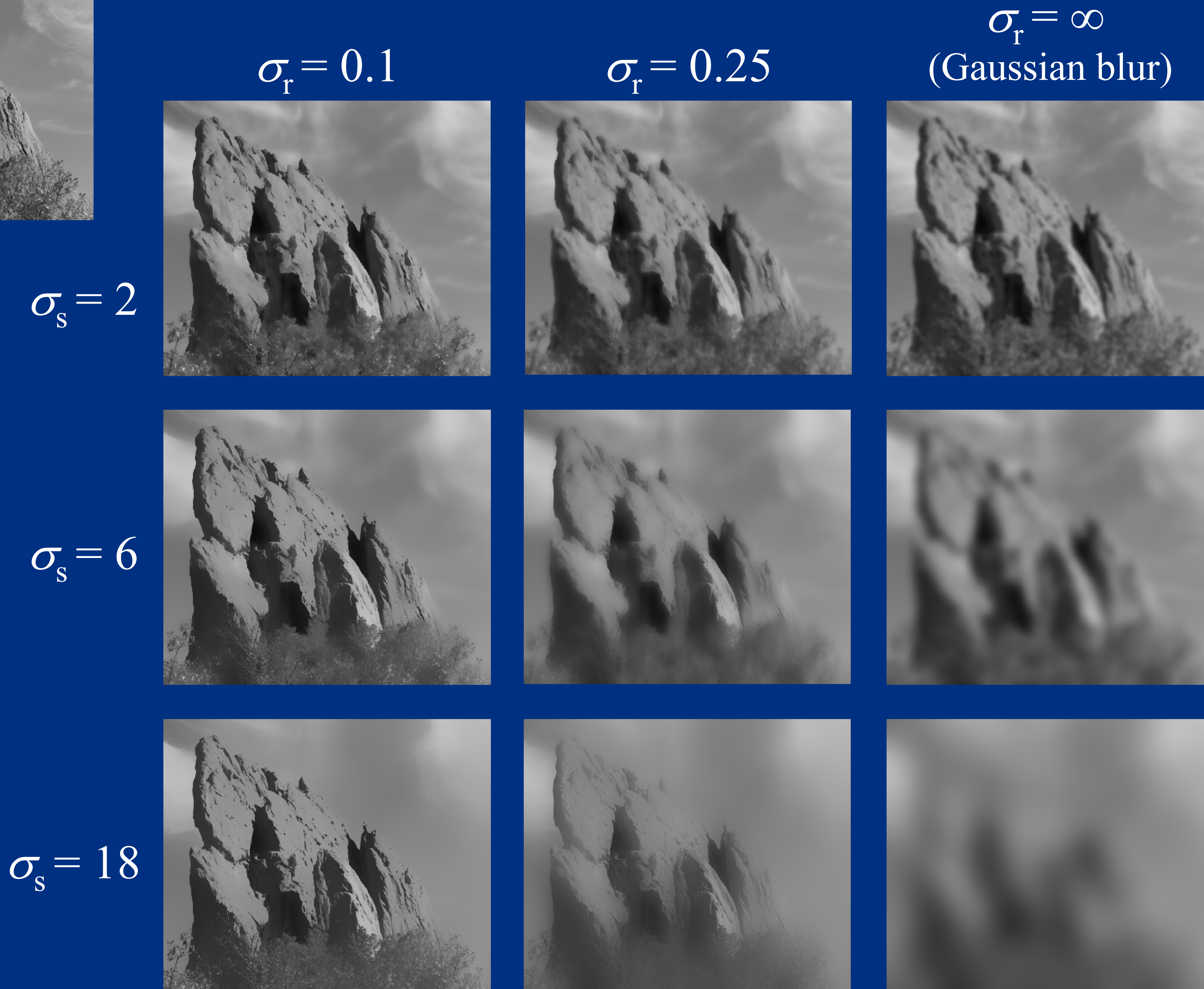
Influence of Pixels

Only pixels close in space and in range are considered.



Exploring the Parameter Space

input



Varying the Range Parameter

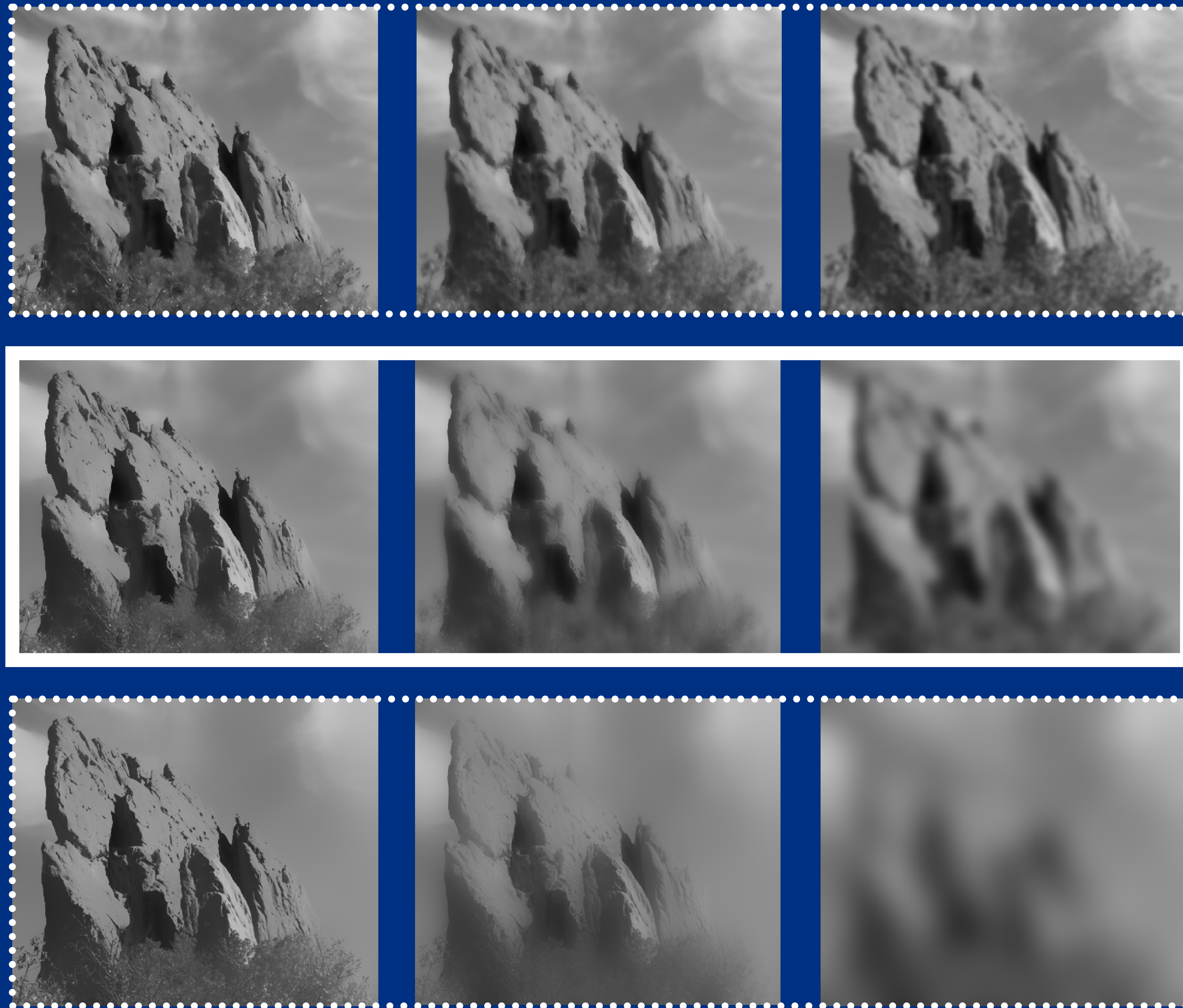
input

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

$\sigma_s = 2$

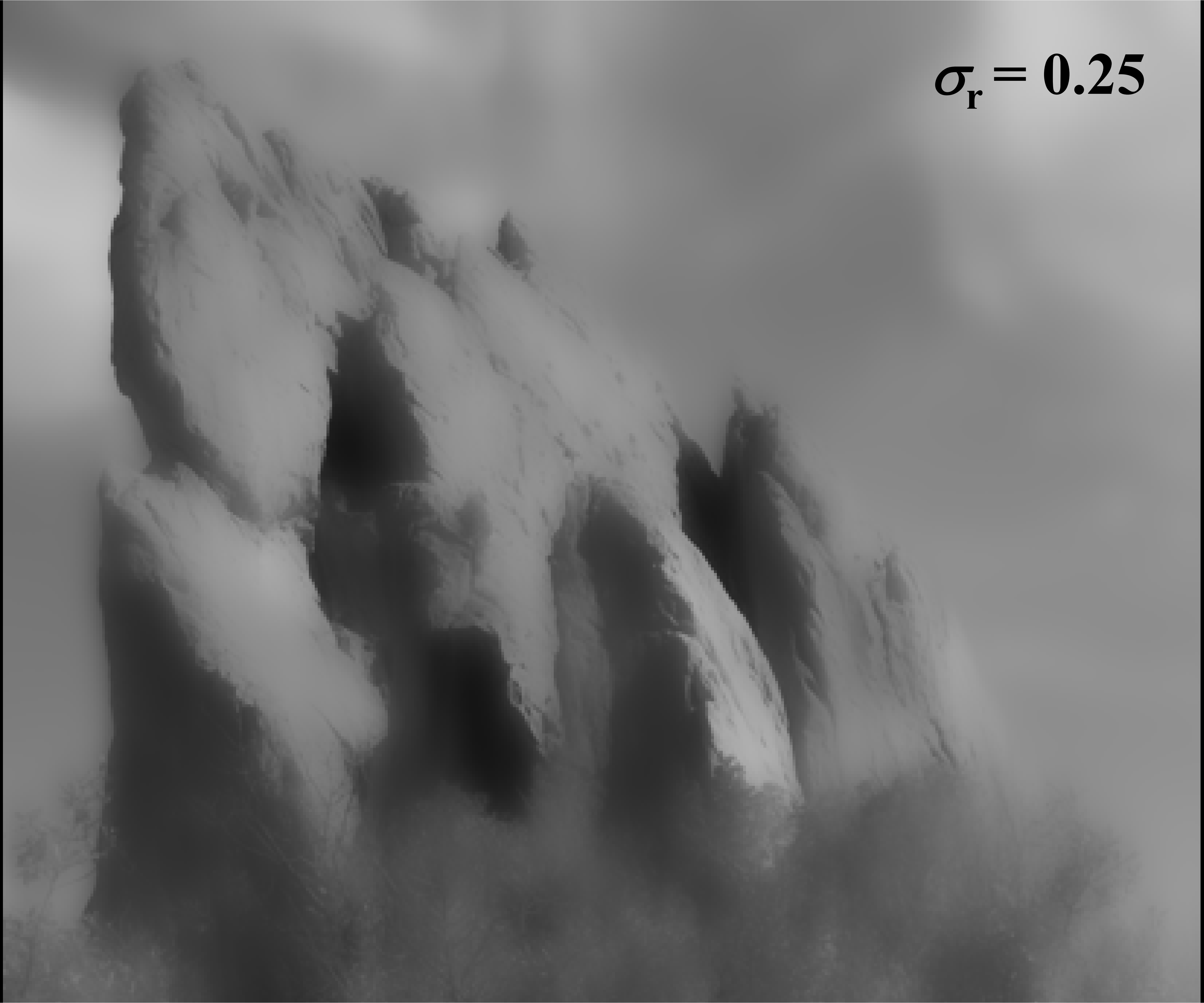
$\sigma_s = 6$

$\sigma_s = 18$




input


$\sigma_r = 0.1$


$\sigma_r = 0.25$


$\sigma_r = \infty$
(Gaussian blur)

Varying the Space Parameter

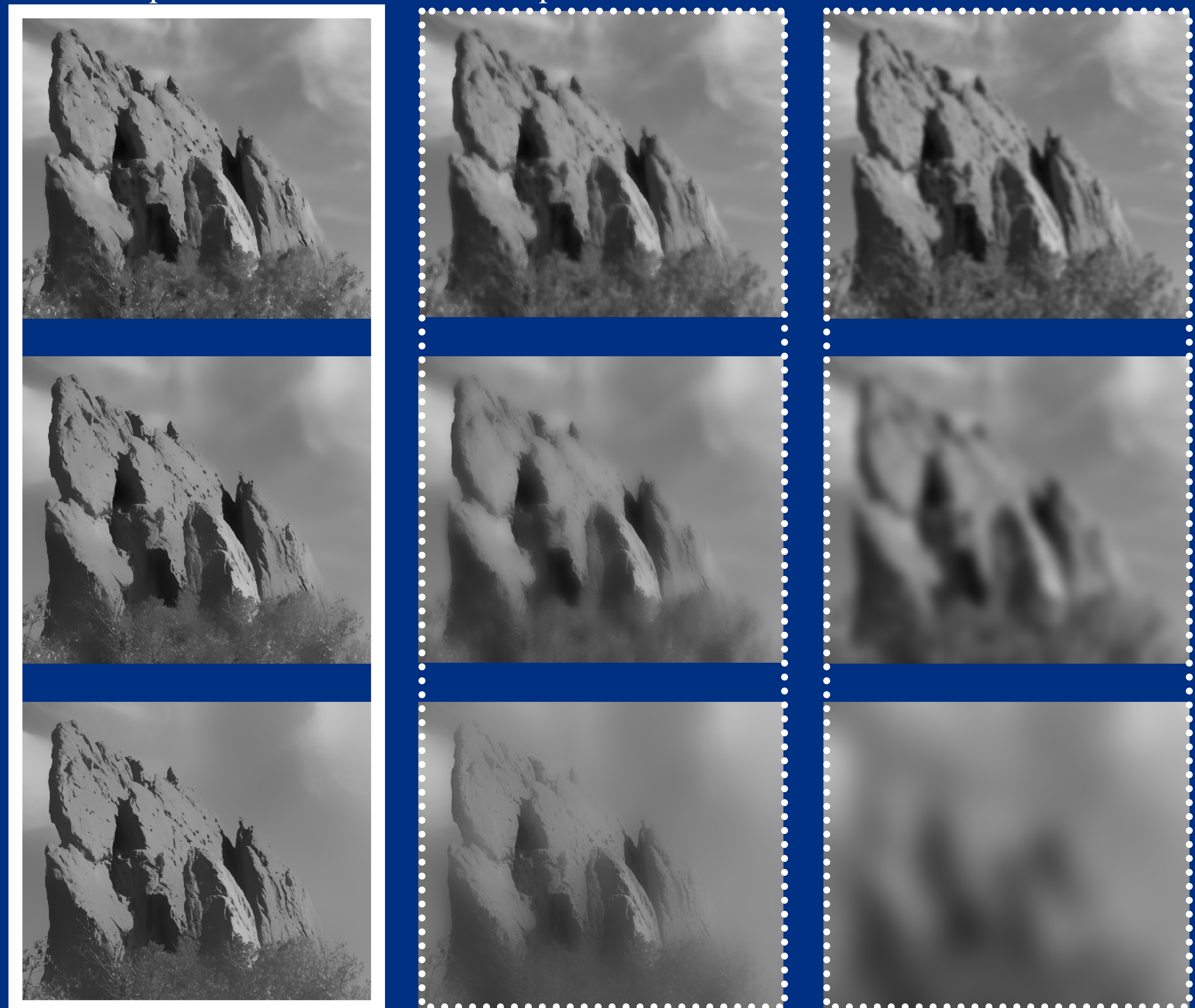
input

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

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$




input


$\sigma_s = 2$


$\sigma_s = 6$


$\sigma_s = 18$

How to Set the Parameters

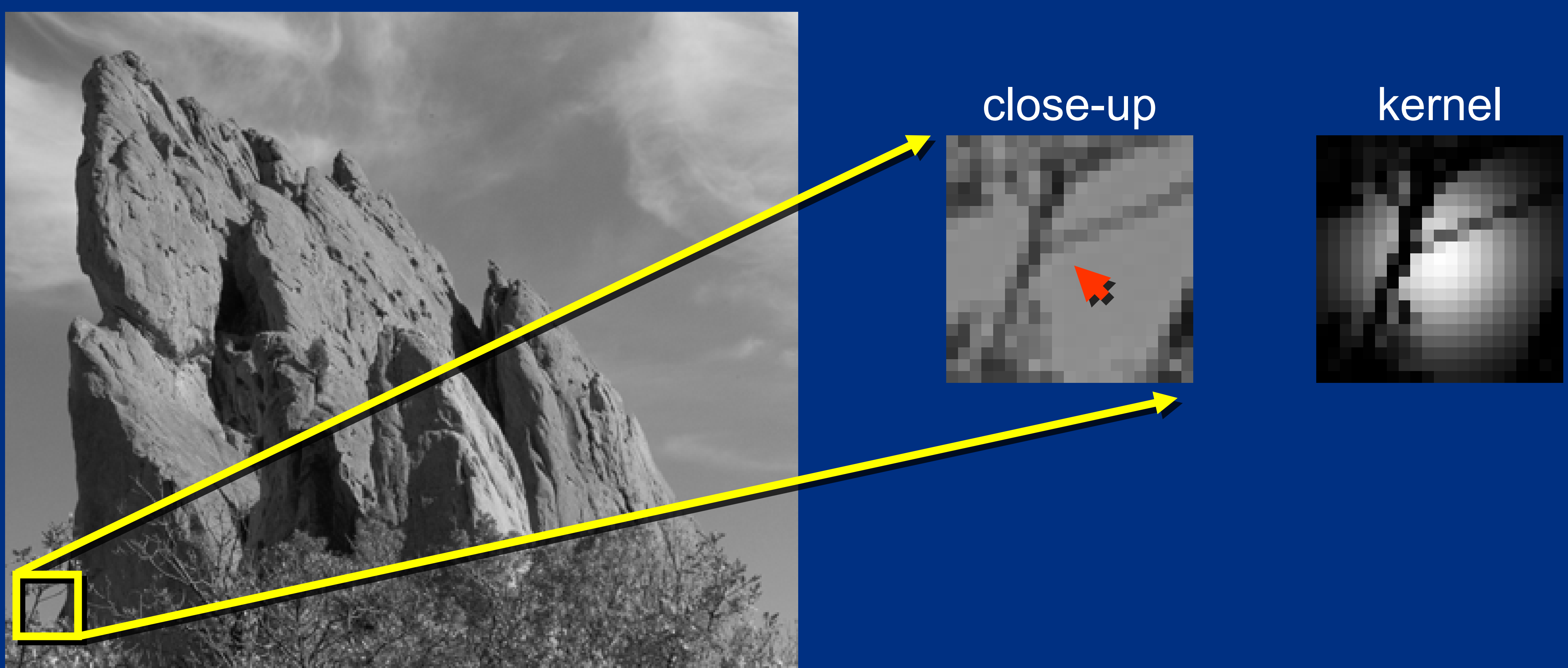
Depends on the application. For instance:

- 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

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



Iterating the Bilateral Filter

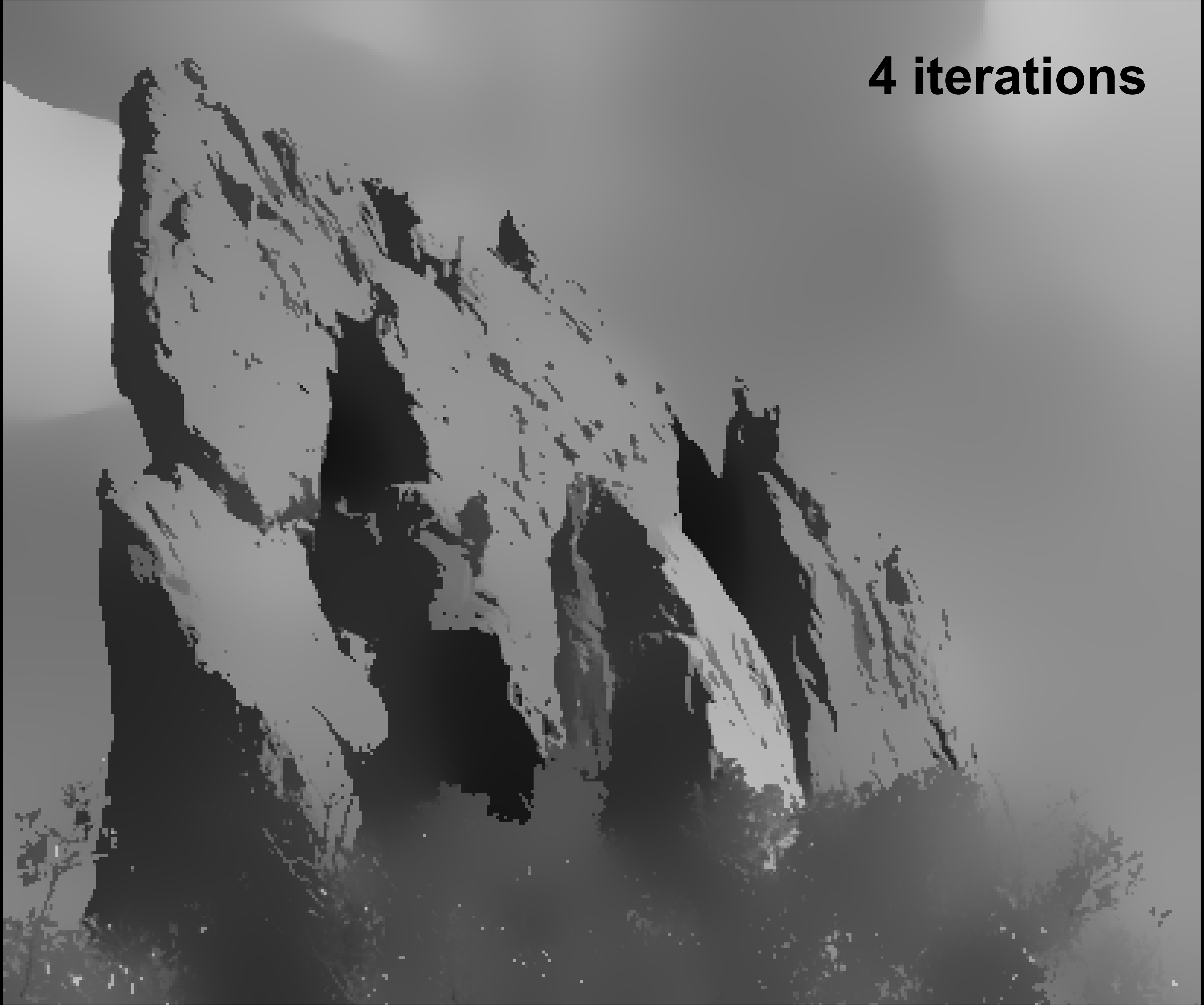
$$I_{(n+1)} = BF\left[I_{(n)}\right]$$

- 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

intensity difference

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

scalar

For color images

color difference

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

3D vector
(RGB, Lab)

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

Hard to Compute

- Nonlinear

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

- Complex, spatially varying kernels
  - Cannot be precomputed, no FFT

- Brute-force implementation is slow > 10min

Questions ?