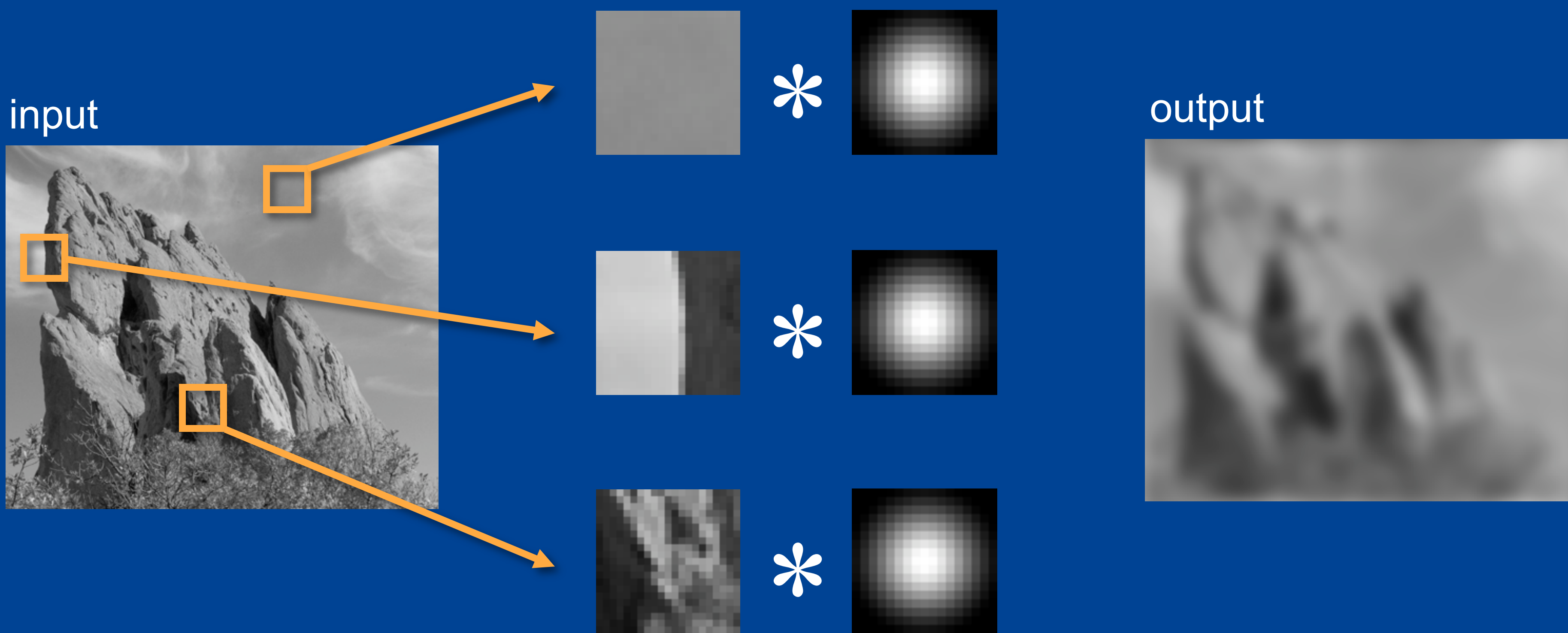
A Gentle Introduction
to Bilateral Filtering
and its Applications

"Fixing the Gaussian Blur": the Bilateral Filter

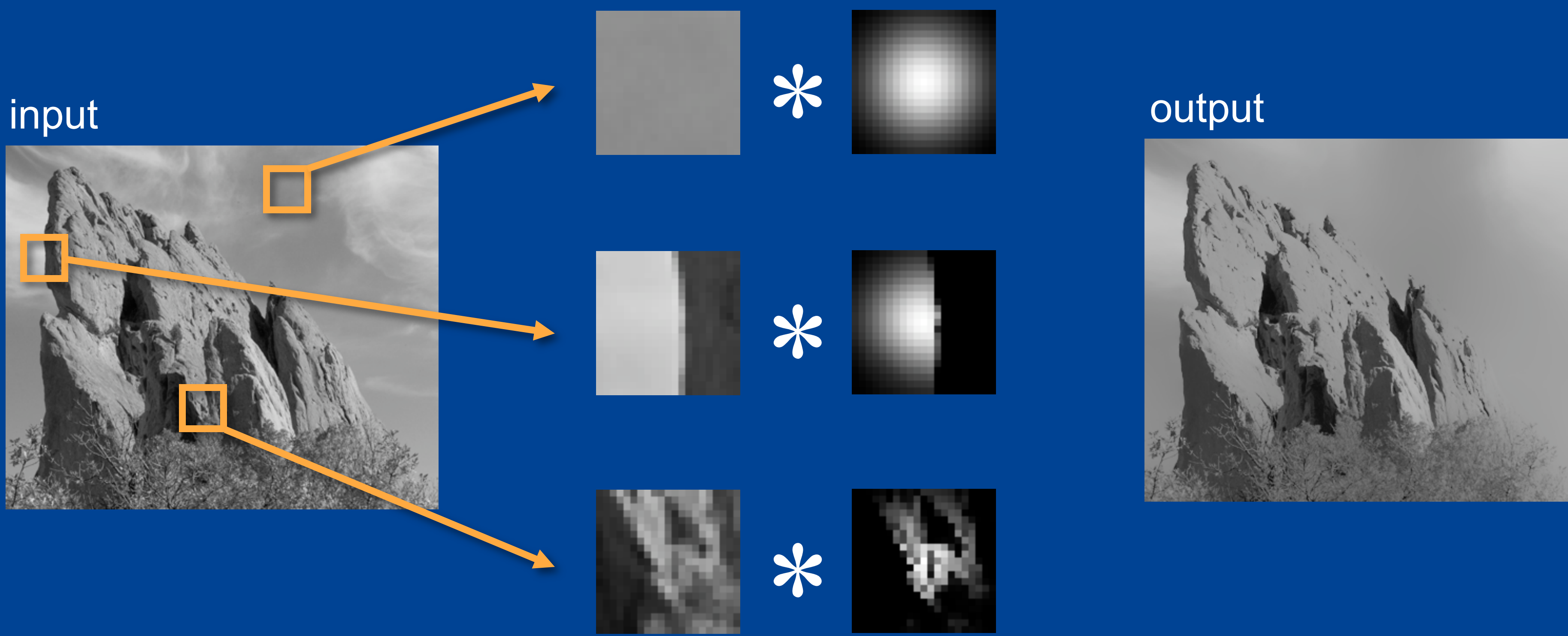
Sylvain Paris – Adobe

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

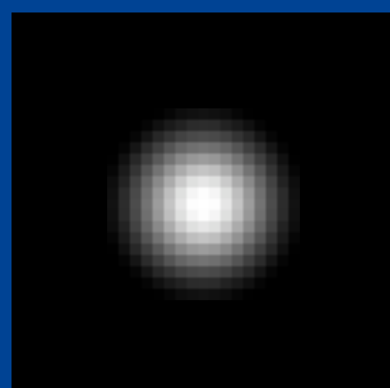
Same idea: **weighted average of pixels**.

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

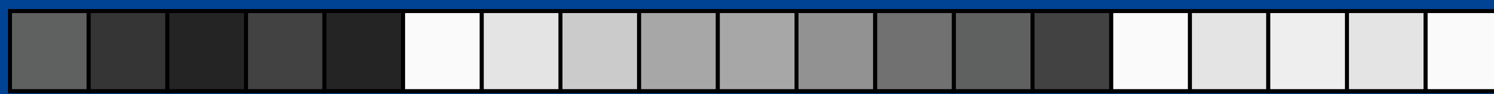
space weight



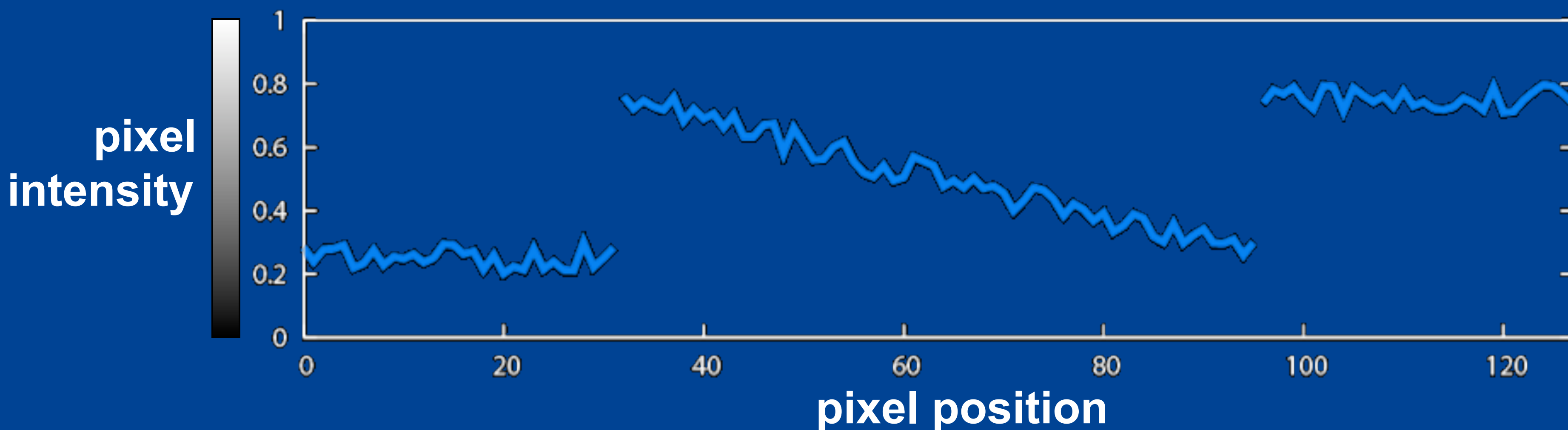
range weight

Illustration a 1D Image

- 1D image = line of pixels

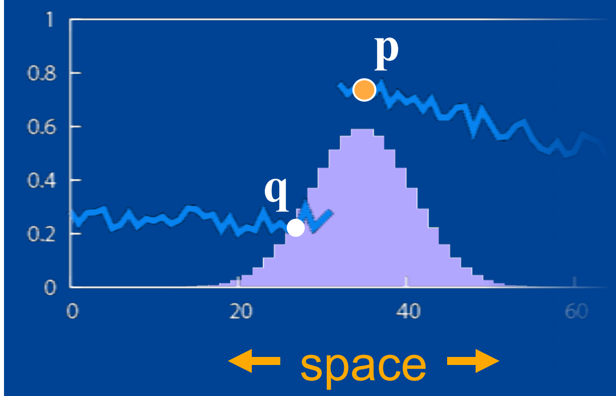


- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

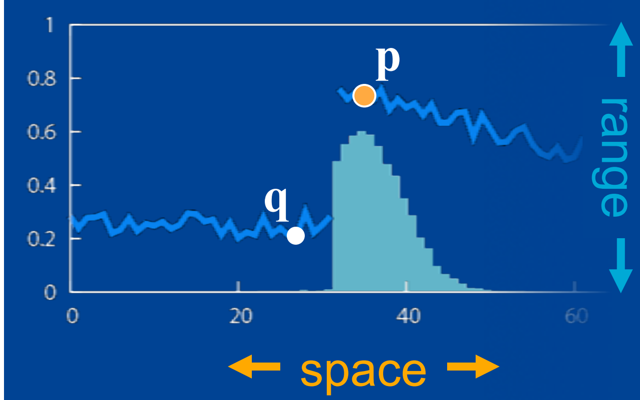


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|) I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



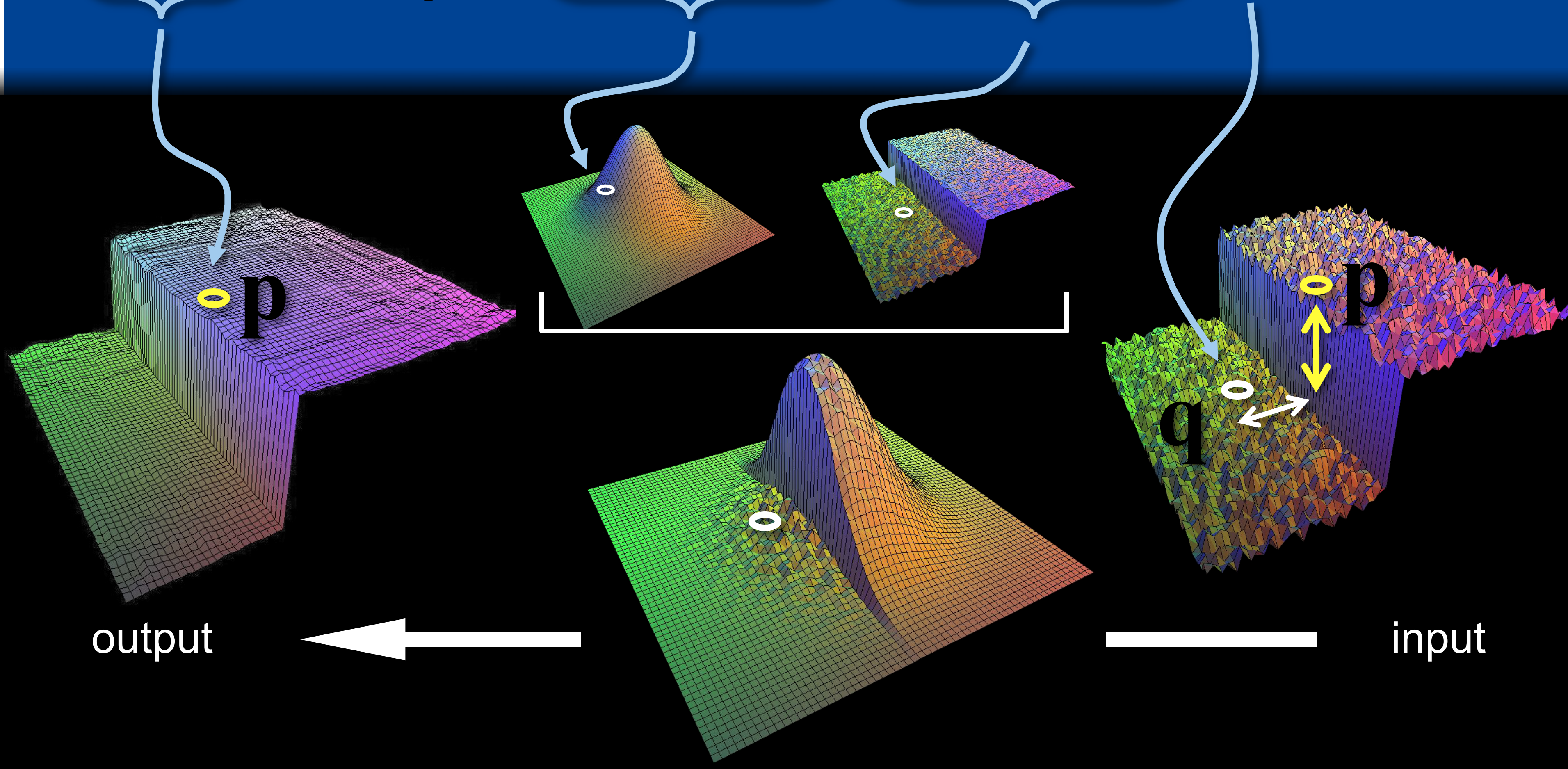
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |) I_{\mathbf{q}}$$

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]

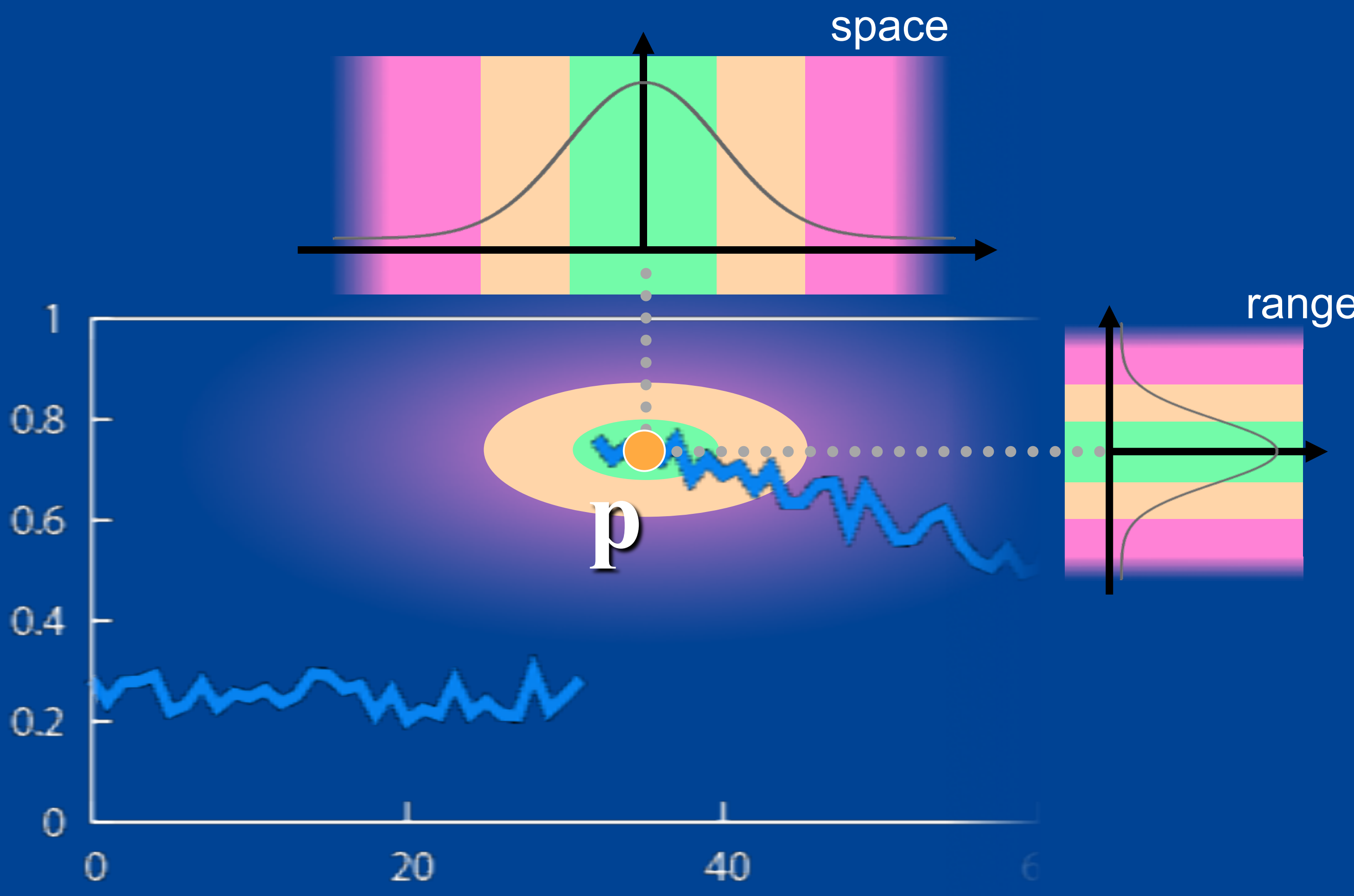
Space and Range Parameters

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

- 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

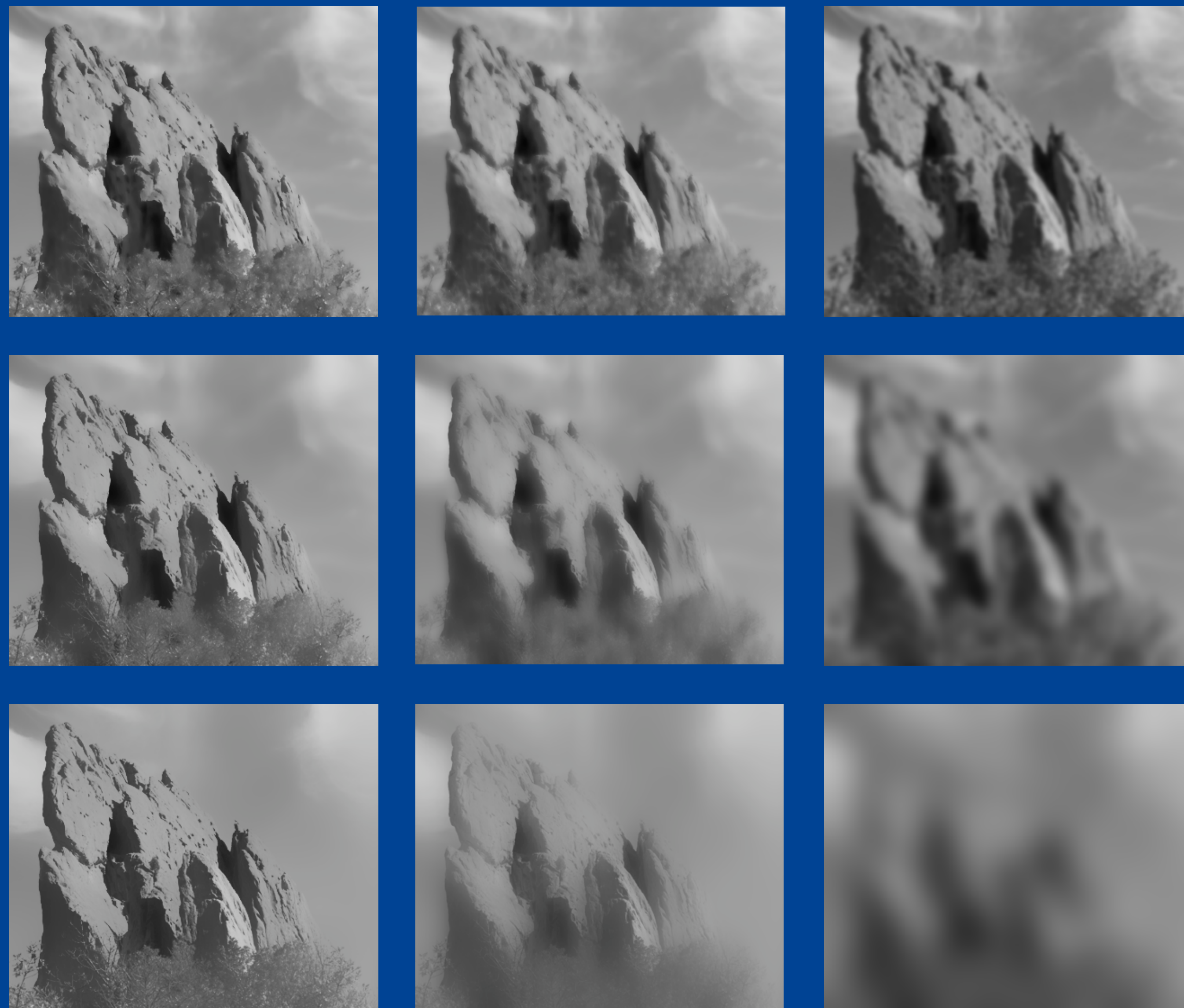
input

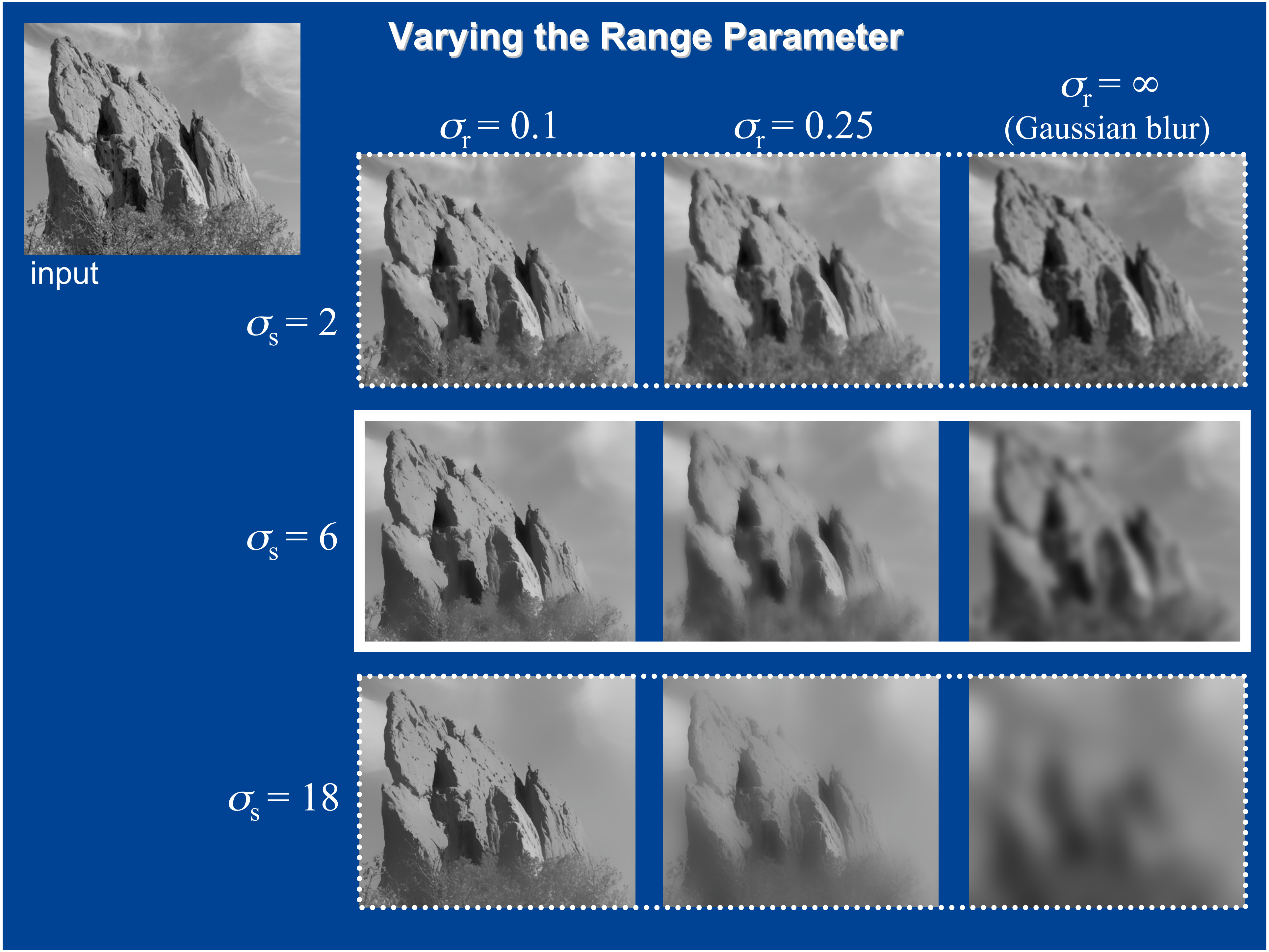
$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (Gaussian blur)

$\sigma_s = 2$

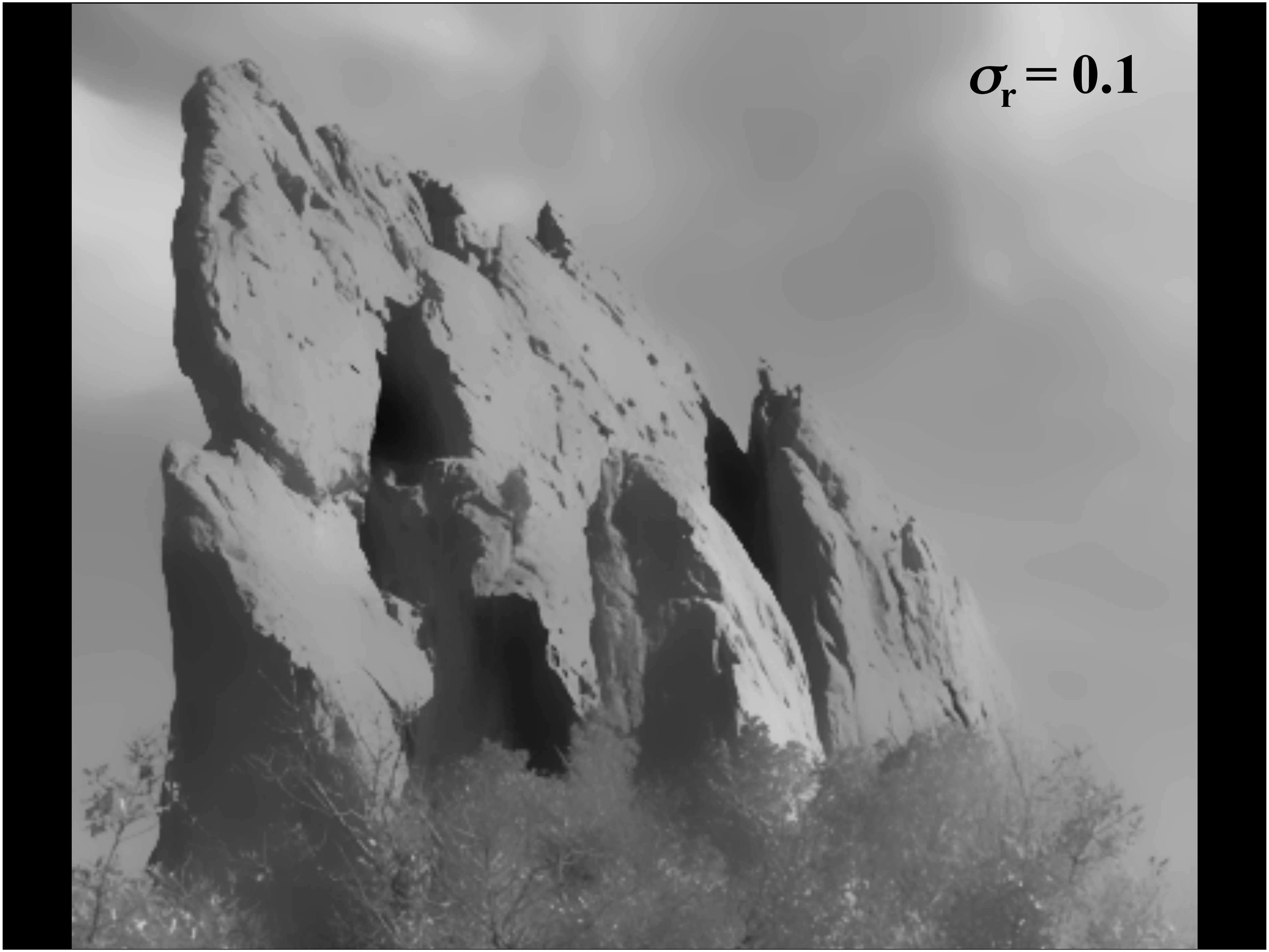
$\sigma_s = 6$

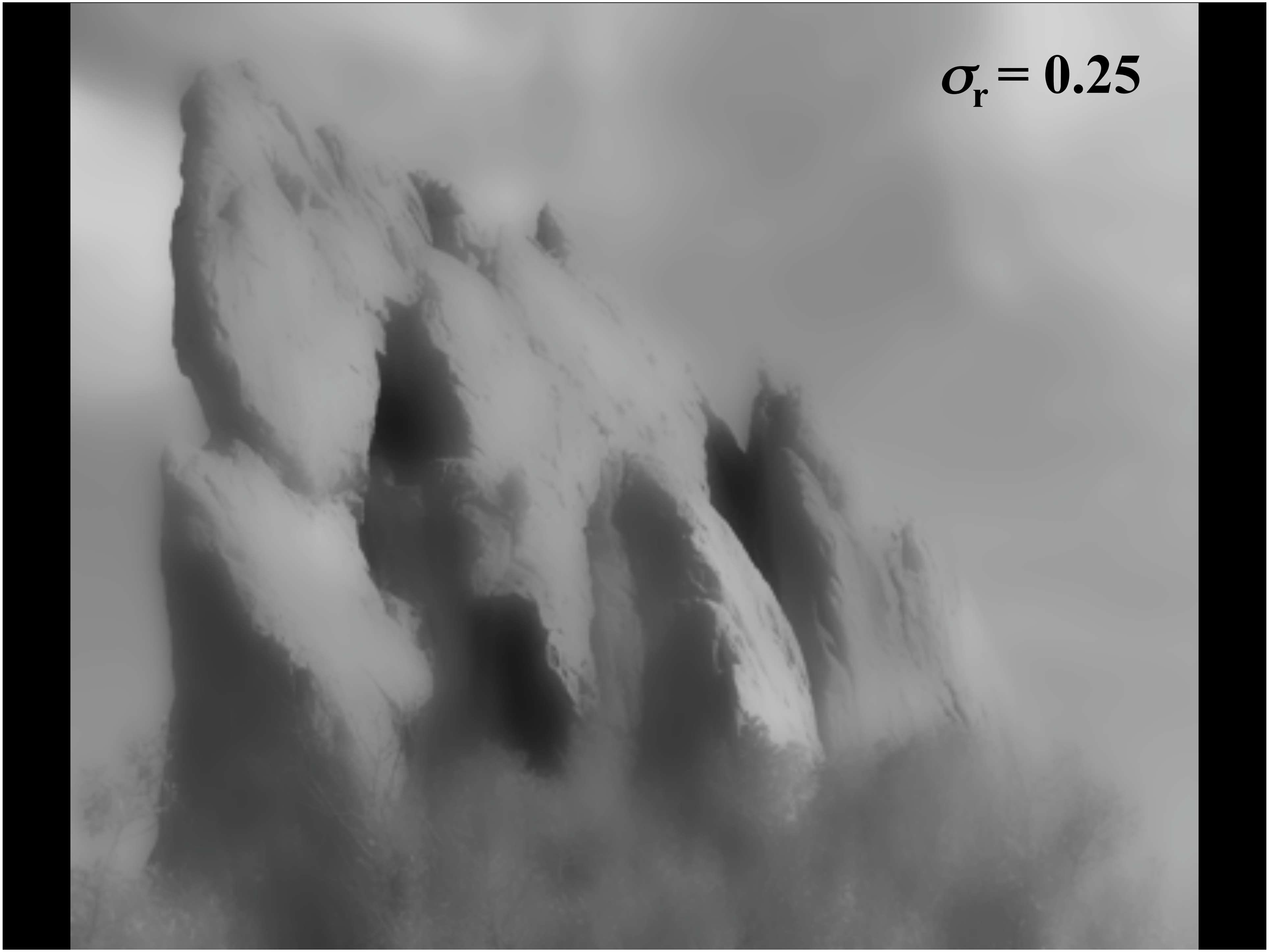
$\sigma_s = 18$




Varying the Range Parameter
input
$\sigma_r = 0.1$
$\sigma_r = 0.25$
$\sigma_r = \infty$
(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

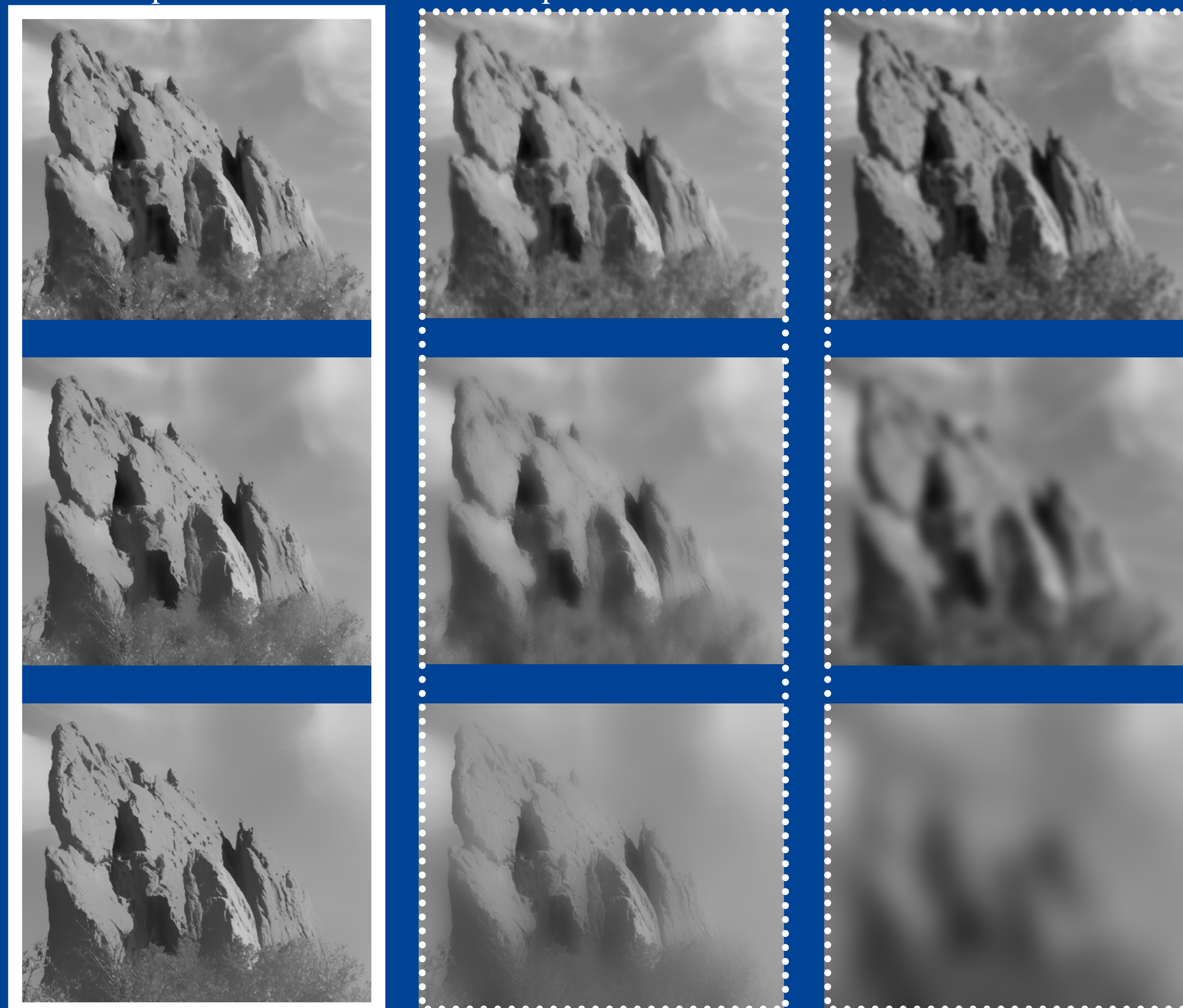
input

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (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

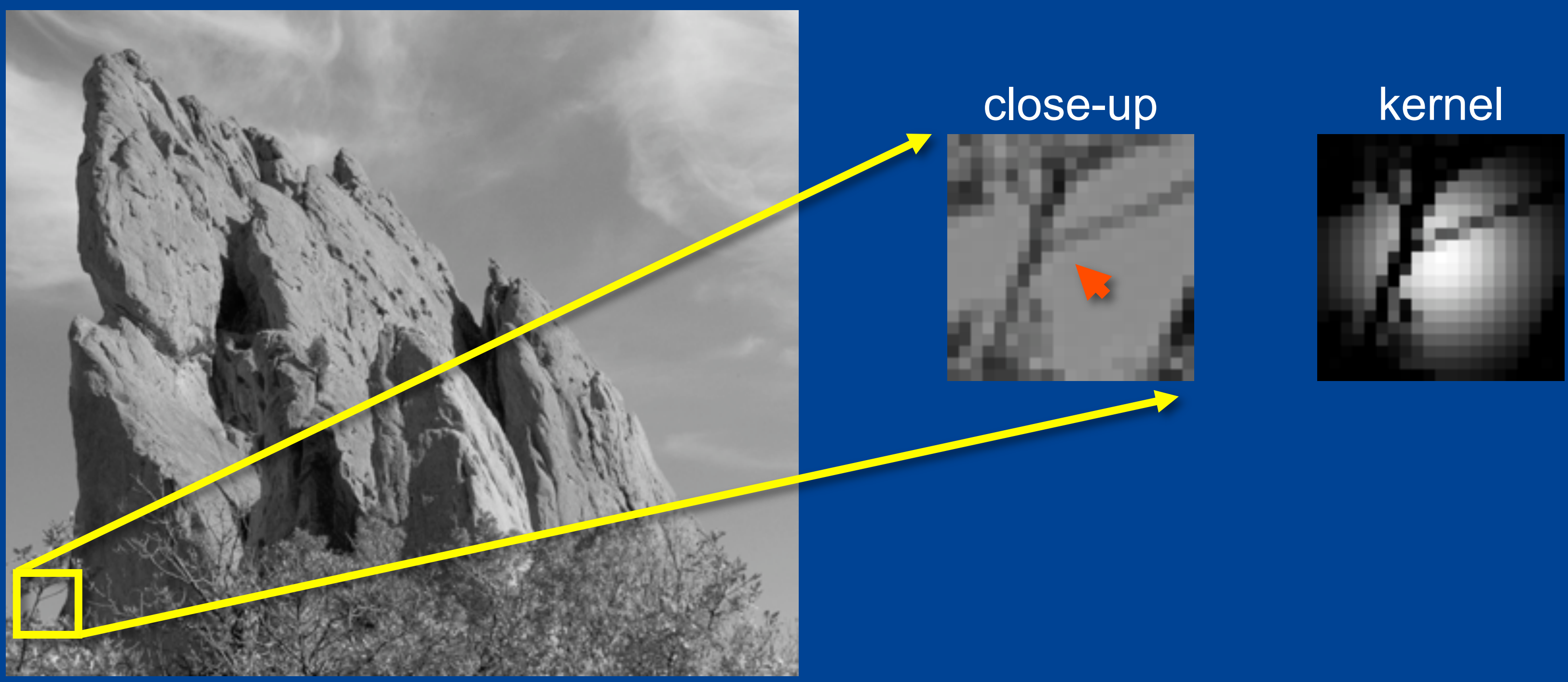
Depends on the application. For instance:

- 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 ~$2\sigma_s$
- Desirable for smoothing: more pixels = more robust
- Different from diffusion that stops at thin lines



Iterating the Bilateral Filter

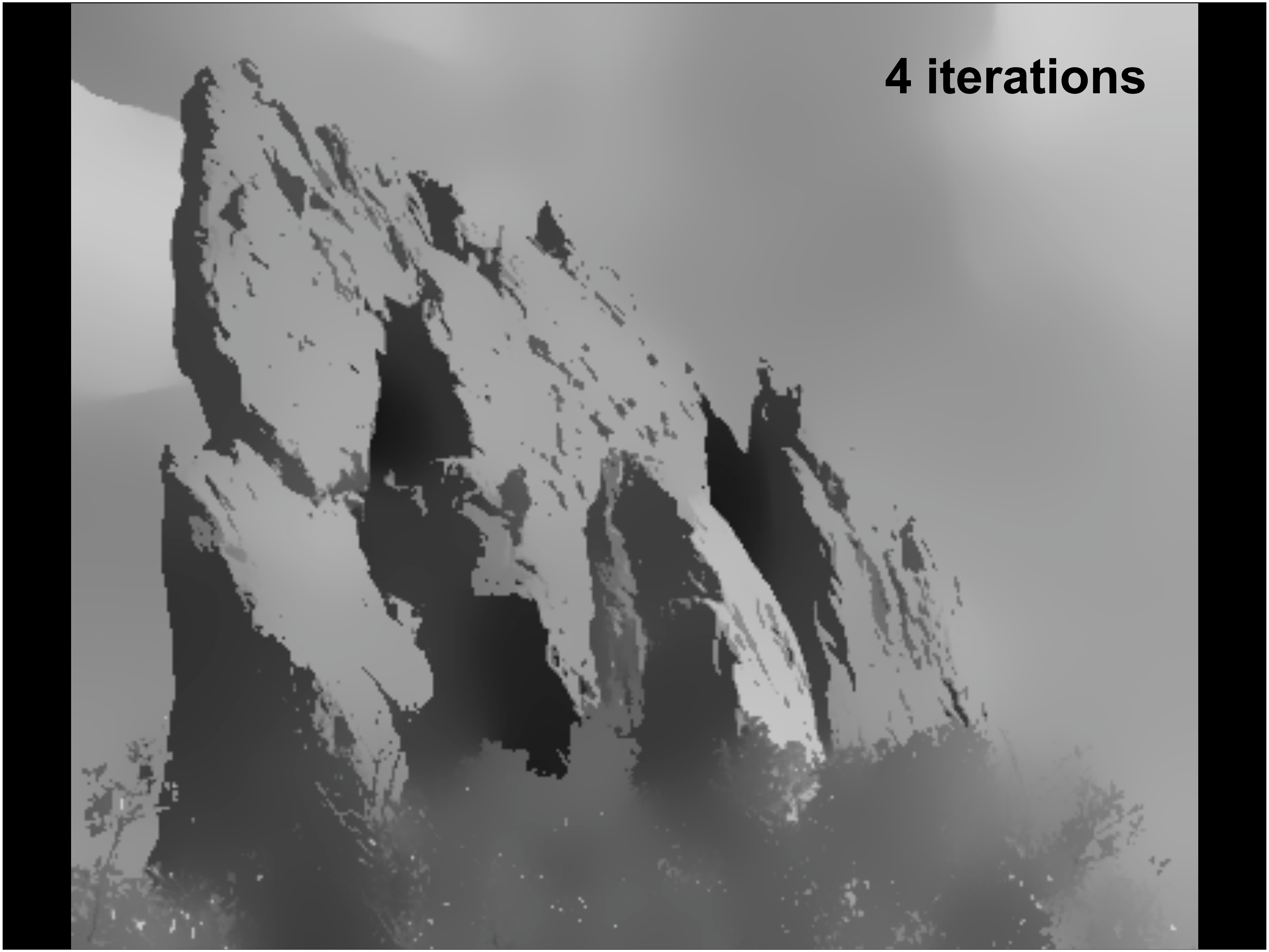
$$I_{(n+1)} = BF\left[I_{(n)}\right]$$

- 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

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

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

Questions ?