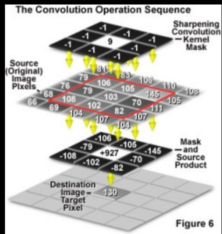


Basic Image Operation (IV)
图像基本操作 (IV)

Mingli Song
College of Computer Science, Zhejiang University
brooksong@zju.edu.cn
http://person.zju.edu.cn/msong/

Spatial filtering

Spatial filtering for smoothing

When you find there are too many artifacts or noises in the image, you can carry out smoothing operation on the image to suppress the noise and reduce the artifacts. However, the smoothing operation will make the image blurring.

Image after low-pass filtering

Spatial filtering

Spatial filtering--smoothing

Smoothing can reduce noises and blurring, which can be used in preprocessing. For instance, remove the subtle details when you just want to extract the big target.

Image after low-pass filtering

Spatial filtering

Spatial filtering--smoothing

- Linear smoothing filter
- Statistical sorting filter

Image after low-pass filtering

Spatial filtering

Linear smoothing filter——Concept

The output of the linear smoothing filter is the mean value of the pixels in the mask. It's also called mean filter.

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Spatial filtering

Linear smoothing filter——application

Mean filter is mainly used for subtle detail removal, namely, eliminating the unwanted region smaller than the mask.

$$\frac{1}{9}\times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Spatial filtering

Linear smoothing filter——example

Simple mean, pixels in the mask window contribute equally to the final result.

$$\frac{1}{9}\times\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Weighted mean, pixels in the mask window contribute unequally to the final result.

$$\frac{1}{16}\times\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Two 3×3 mean filter, each filter's factor equals to the sum of all the coefficients in order to obtain the mean value.

Spatial filtering

Linear smoothing filter——general equation

$$g(x,y)=\frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s,y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

where the size of the filter is $(2a+1) \times (2b+1)$, *w is the filter, f is the input image, g is the output image.*

Spatial filtering

Linear smoothing filter——Explanation

The size of the mask is very important for the final result. When the mask is small, the blurring effect is very subtle, and vice versa.

Spatial filtering

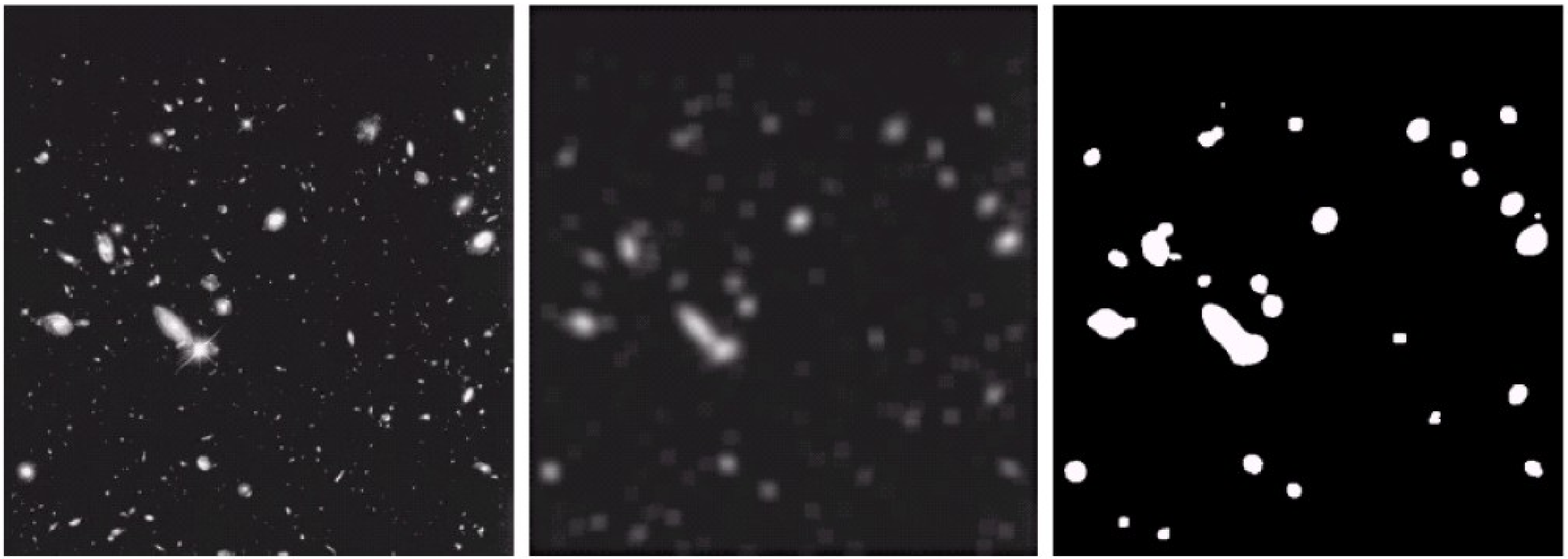
Linear smoothing filter——application

In order to obtain a brief description of the object of interest, linear smoothing filter is used to blur the image to remove the smaller object while keeping the larger object.

Hence, the size of the mask depends on the object to be merged into the background.

Spatial filtering

Linear smoothing filter——application



a b c

FIGURE 3.36 (a) Image from the Hubble Space Telescope. (b) Image processed by a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Spatial filtering

Statistical sorting filter——Concept

Statistical filter is a kind of nonlinear spatial filter, whose response is based on the sorting of pixel value in the mask window. The value of center pixel depends on the sorting result in the window.

The most popular statistical filter is median filter.

Spatial filtering

Statistical sorting filter——Median filter

- Substitute the center pixel with the median value in the neighborhood.
- Provide excellent de-noise ability, which introduce less blurring than the mean filter.
- Be effective to deal with the pulse noise (or pepper noise) because this kind of noise looks like bright or dark point in the image.

Spatial filtering

Statistical sorting filter——median filter

- The median value ξ——in a set of numbers, About half of them are smaller than ξ ， others larger than ξ.

- In order to perform median filtering on a pixel in the image, pixel sorting should be carried out in the mask window to select the median value and change the pixel value to the median value.

Spatial filtering

Statistical sorting filter——media filter, example

In a 3×3 neighborhood, there are a series pixel values ：

（ 21 ， 100 ， 99 ， 22 ， 20 ， 102 ， 97 ， 101 ）

After sorting ：

（ 20 ， 21 ， 22 ， 97 ， 100 ， 101 ， 012 ）

Me

22	17	102	105	106
21	21	100	99	102
19	22	$\boxed{20}$	102	102
24	21	97	101	104
19	18	101	108	101

22	17	102	105	106
21	21	99	102	102
19	21	$\boxed{97}$	102	102
24	21	97	101	104
19	18	101	108	101

Spatial filtering

Statistical sorting filter——median filter

Usually we use $n \times n$ median filter to remove the unwanted brighter or darker pixels in the neighborhood, and their area is less than $n^2/2$ （ half of the mask window ）.

Spatial filtering

Sharpening spatial filter——Purpose

Enhance the detail or sharpen the blurred part in the image

Spatial filtering

Sharpening spatial filter——Tool

Differential operator is a sharpening tool, whose response depends on the variation between neighboring pixel values.

Differential Operator strengthens the edges and other obvious variations (including noise) in the image, and weakens the slight changes.

Spatial filtering

Sharpening operator

- Concept of differential operator
- Second order differential based image enhancement——Laplacian operator
- First order differential based image enhancement——gradient based method

Spatial filtering

Differential——

For a function $f(x)$, we use difference to represent the differential operator:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

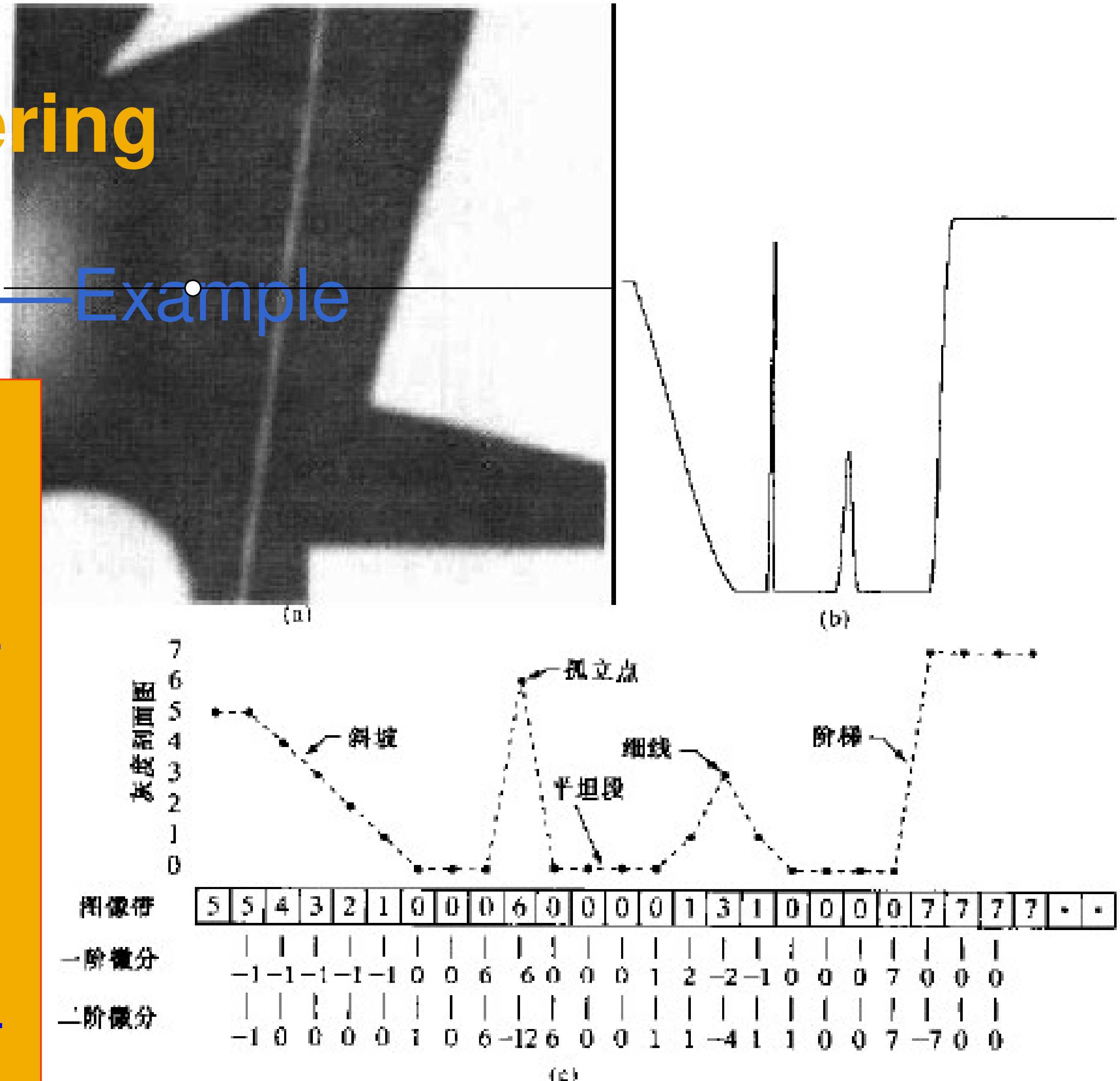
Similarly, the second order differential is :

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Spatial filtering

Differential——Example

(a) A simple image, composed by solid object, line segment and single noisy pixel.
(b) Profile of the image along with the central line, which includes a noisy pixel.
(c) Simplified profile.by dot-line.



$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Spatial filtering

Image enhancement based on first order differential operator——gradient based method

For a function f(x,y)，we first define a 2-D vector：

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

How to compute gradient?

Its magnitude is computed as:

$$\nabla f = \left[G_x^2 + G_y^2 \right]^{\frac{1}{2}} = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{\frac{1}{2}}$$

Spatial filtering

Image enhancement based on first order differential operator——gradient based operator

It's time consuming to compute the gradients for all the pixels in the image. Hence, absolute value is usually used to replace the original gradient magnitude.

$$\nabla f \approx |G_x| + |G_y|$$

Spatial filtering

Gradient based method——Another approach

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-1	0
0	1

0	-1
1	0

Robert cross gradient operator, Gx=(z9-z5) Gy=(z8-z6)

Original image, z_i is the pixel value, z_5 is the center pixel.

$$\boxed{\nabla f = \left[\left(z_9 - z_5\right)^2 + \left(z_8 - z_6\right)^2\right]^{\frac{1}{2}}}$$

Or

$$\boxed{\nabla f \approx \left|z_9 - z_5\right| + \left|z_8 - z_6\right|}$$

Spatial filtering

Image enhancement based on second order differential——Laplacian operator

For a function $f(x,y)$, Laplacian operator is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Spatial filtering

The discrete Laplacian operator:

Along x axis:

$$\frac{\partial^2 f}{\partial x^2} = f\left(x+1, y\right) + f\left(x-1, y\right) - 2f\left(x, y\right)$$

Along y axis:

$$\frac{\partial^2 f}{\partial y^2} = f\left(x, y+1\right) + f\left(x, y-1\right) - 2f\left(x, y\right)$$

Hence, the discrete Laplacian operator is:

$$\nabla^2 f = \left[f\left(x+1, y\right) + f\left(x-1, y\right) + f\left(x, y+1\right) + f\left(x, y-1\right) \right] - 4f\left(x, y\right)$$

Spatial operator

Mask of Laplacian operator

$$\nabla^2 f = \left[f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) \right] - 4f(x, y)$$

0	1	0
1	-4	1
0	1	0

It is rotation invariant.

Spatial filtering

Extending the mask

The elements in the diagonal direction can also be taken into account ：

$$\begin{aligned}\nabla^2 f = [&f(x-1, y-1) + f(x, y-1) + f(x+1, y-1) \\ &+ f(x-1, y) + f(x+1, y) \\ &+ f(x-1, y+1) + f(x, y+1) + f(x+1, y+1)] \\ &-8f(x, y)\end{aligned}$$

Or

$$\nabla^2 f = \sum_{i=-1}^{1}\sum_{j=-1}^{1} f(x+i, y+j) - 9f(x, y)$$

Spatial filering

Extended mask

$$\nabla^2 f = \sum_{i=-1} \sum_{j=-1} f(x+i, y+j) - 9f(x,y)$$

1	1	1
1	-8	1
1	1	1

It's also rotation invariant, i.e., isotropic.

Spatial filtering

Extended mask

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

When you fuse the Laplacian result and the original image together, you have to consider the symbol difference between them.

Spatial filtering

Application of Laplacian operator

Image enhancement by Laplaician:

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y) \\ f(x,y)+\nabla^2 f(x,y) \end{cases}$$

If the center element of the mask is negative

如果拉普拉斯掩模中心系数为负

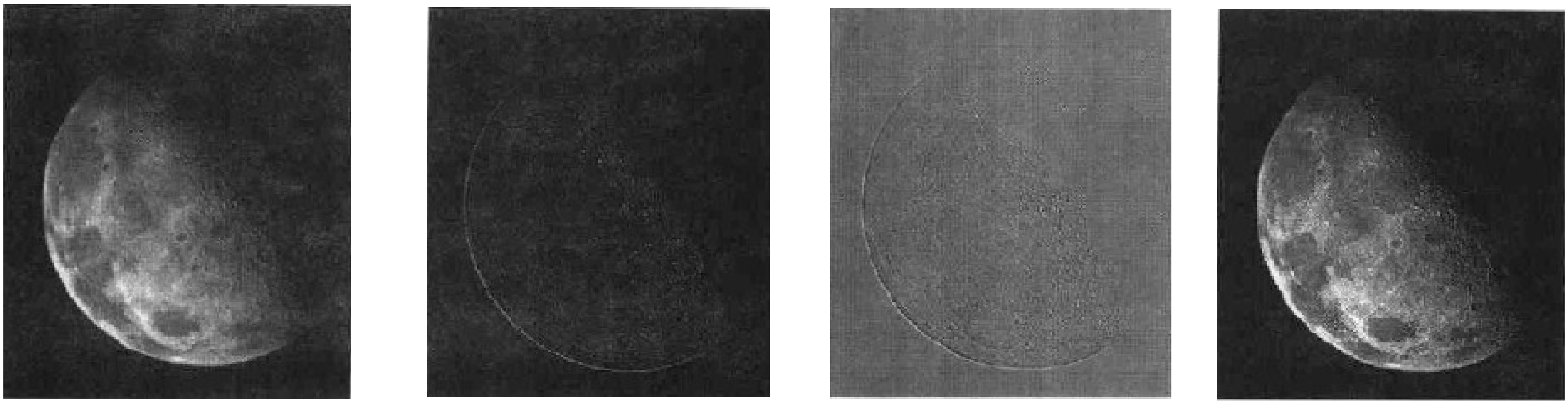
If the center element of the mask is positive

如果拉普拉斯掩模中心系数为正

Fuse the original image and the Laplacian result can preserve the sharpening effect and restore the original visual information.

Spatial filtering

Laplacian Example



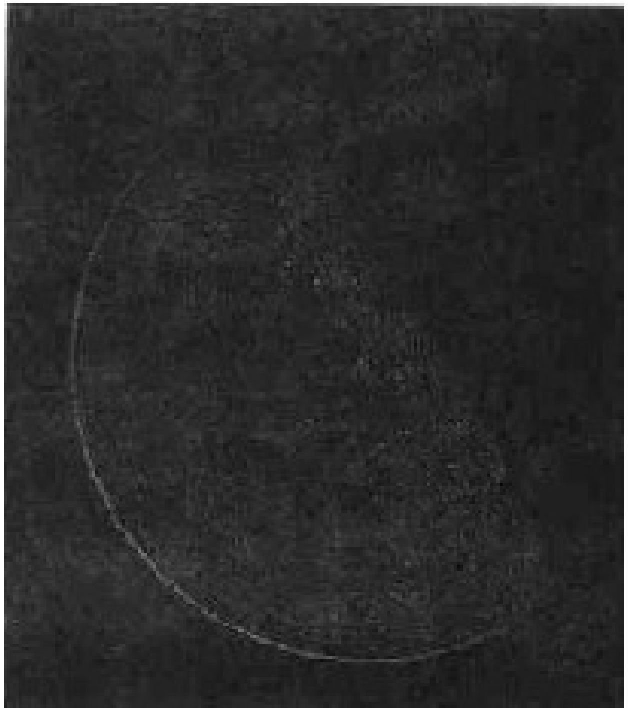
(a)Original image (b)Laplacian result (c)Rearranged Laplacian result (d) After fusion

Spatial filtering

Laplacian: example

(a) Original image

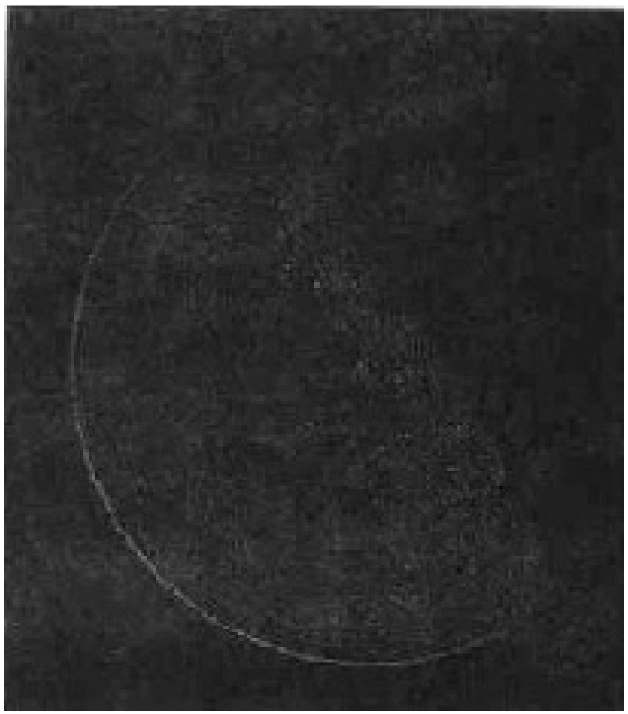
1	1	1
1	-8	1
1	1	1



(b) Laplacian result

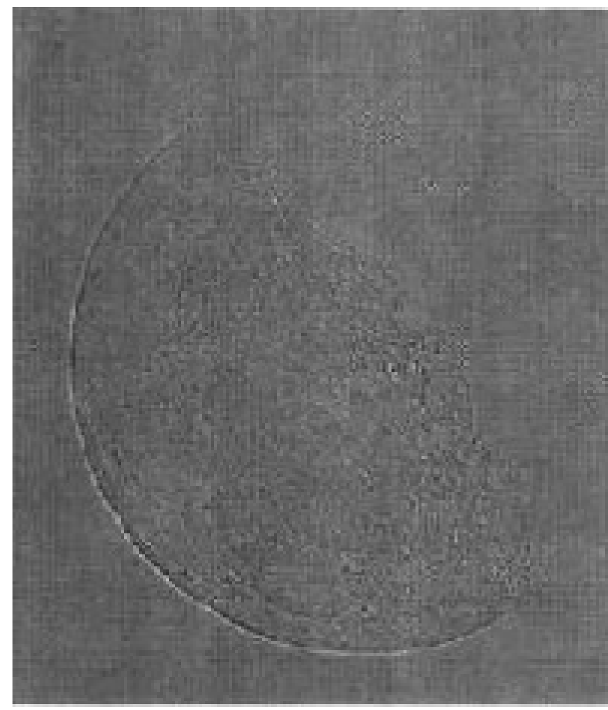
Spatial filtering

Laplacian: example



(b) Laplacian result

Rearrange the Laplacian result in terms of enhancing level

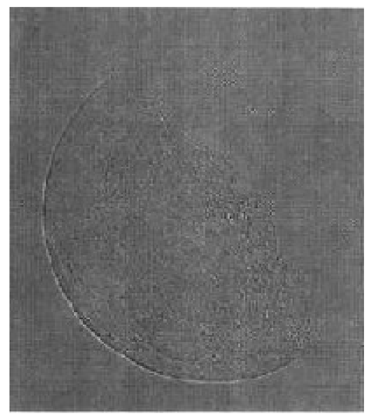


(c) Rearranged Laplacian resu

Spatial filtering

Laplacian-example

(a) Original image



Fuse

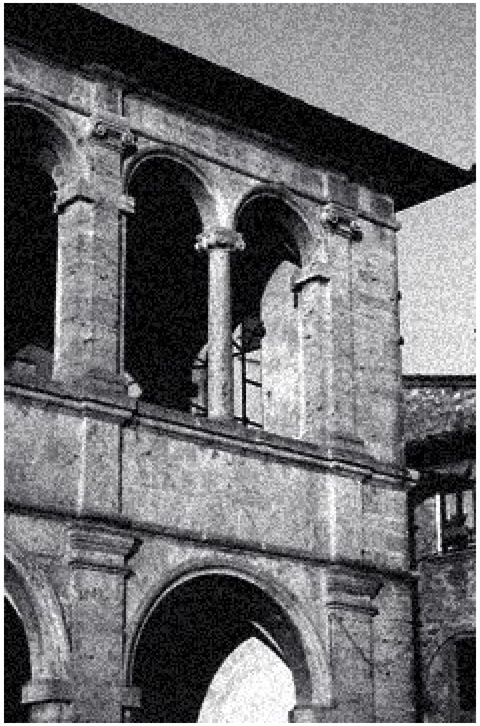
(d) After fusion

- Bilateral filtering

Input

Gaussian smoothing

Bilateral filtering



Input (Noisy image)

Gaussian filtering

Bilateral filtering

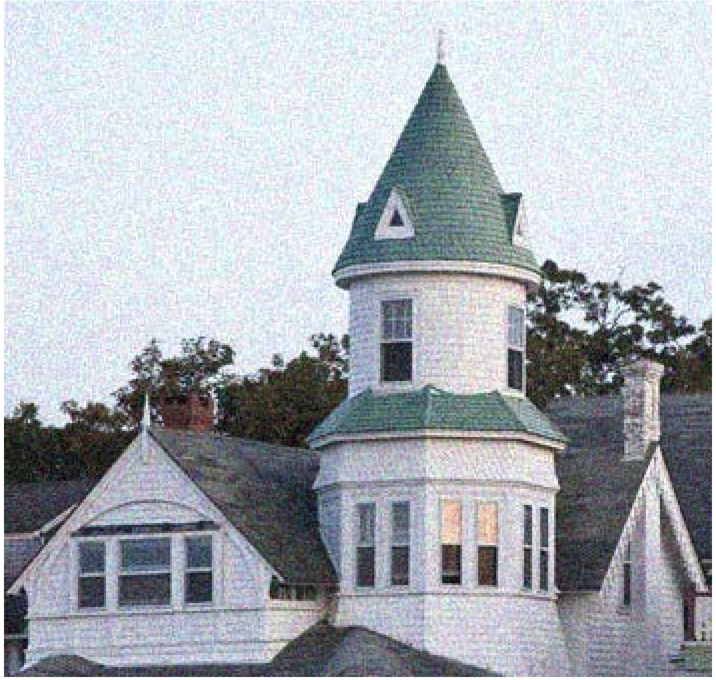
- The bilateral filter is becoming a basic tool in computational photography.

- Many applications with high quality results.
  - Not always the best result but often good
  - Easy to understand, adapt and setup

- Denoise

Noisy image

Median filter



Noisy image

Bilateral filtering

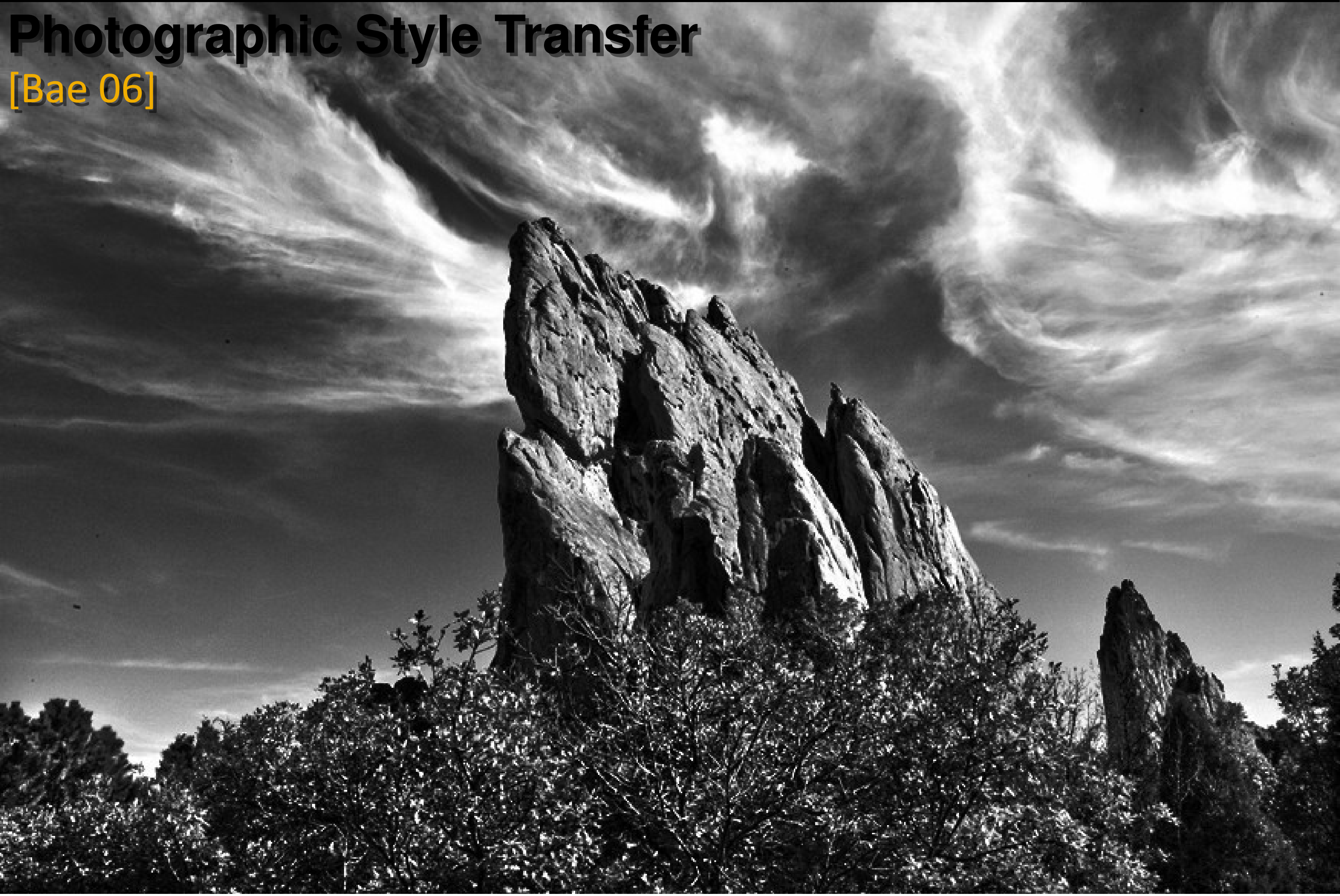
Photographic Style Transfer

[Bae 06]

input

Photographic Style Transfer

[Bae 06]



output

Tone Mapping
[Durand 02]

HDR input

Tone Mapping

[Durand 02]

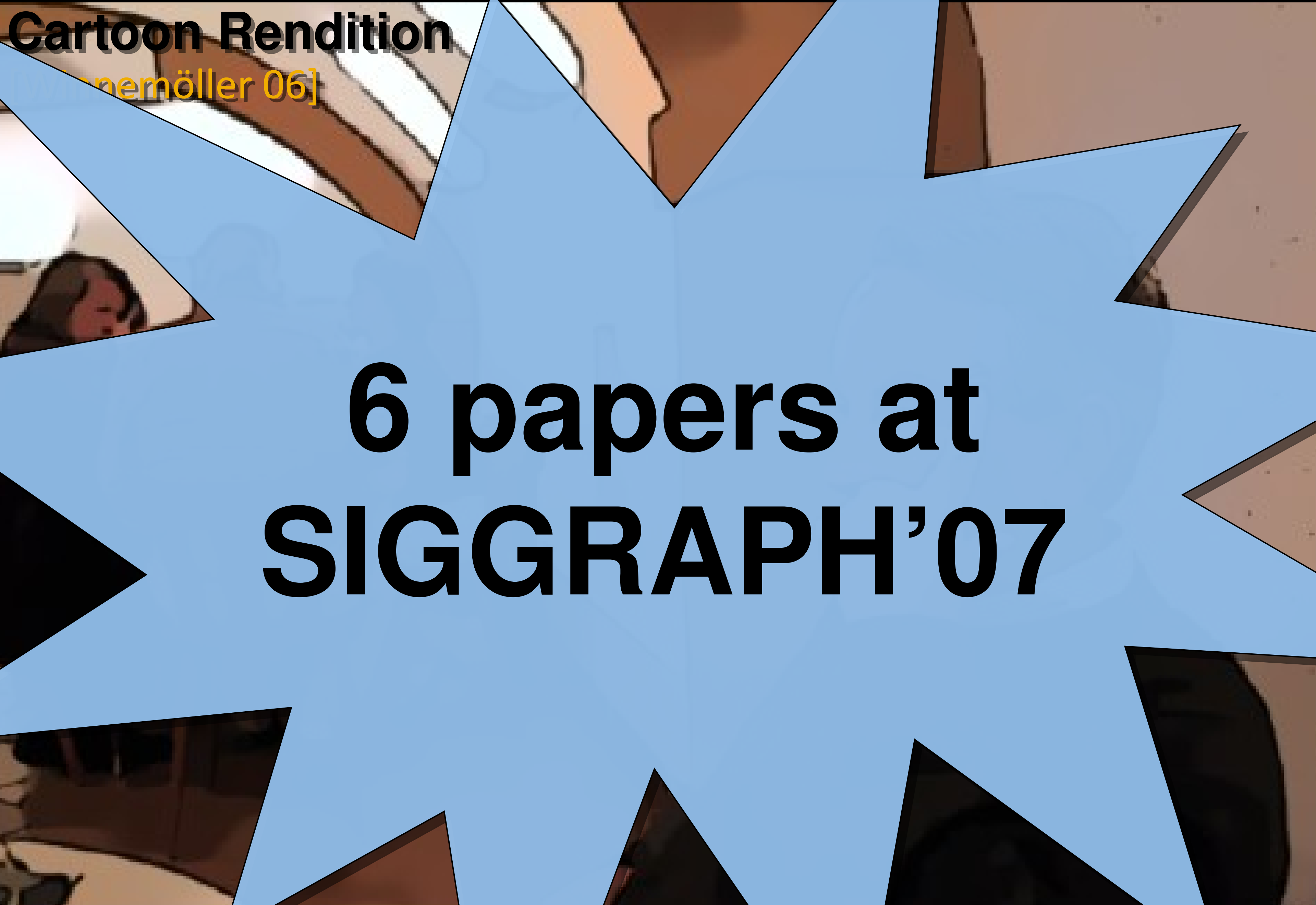
output

Cartoon Rendition
[Winnemöller 06]

input

Cartoon Rendition

nemöller 06]



6 papers at SIGGRAPH’07

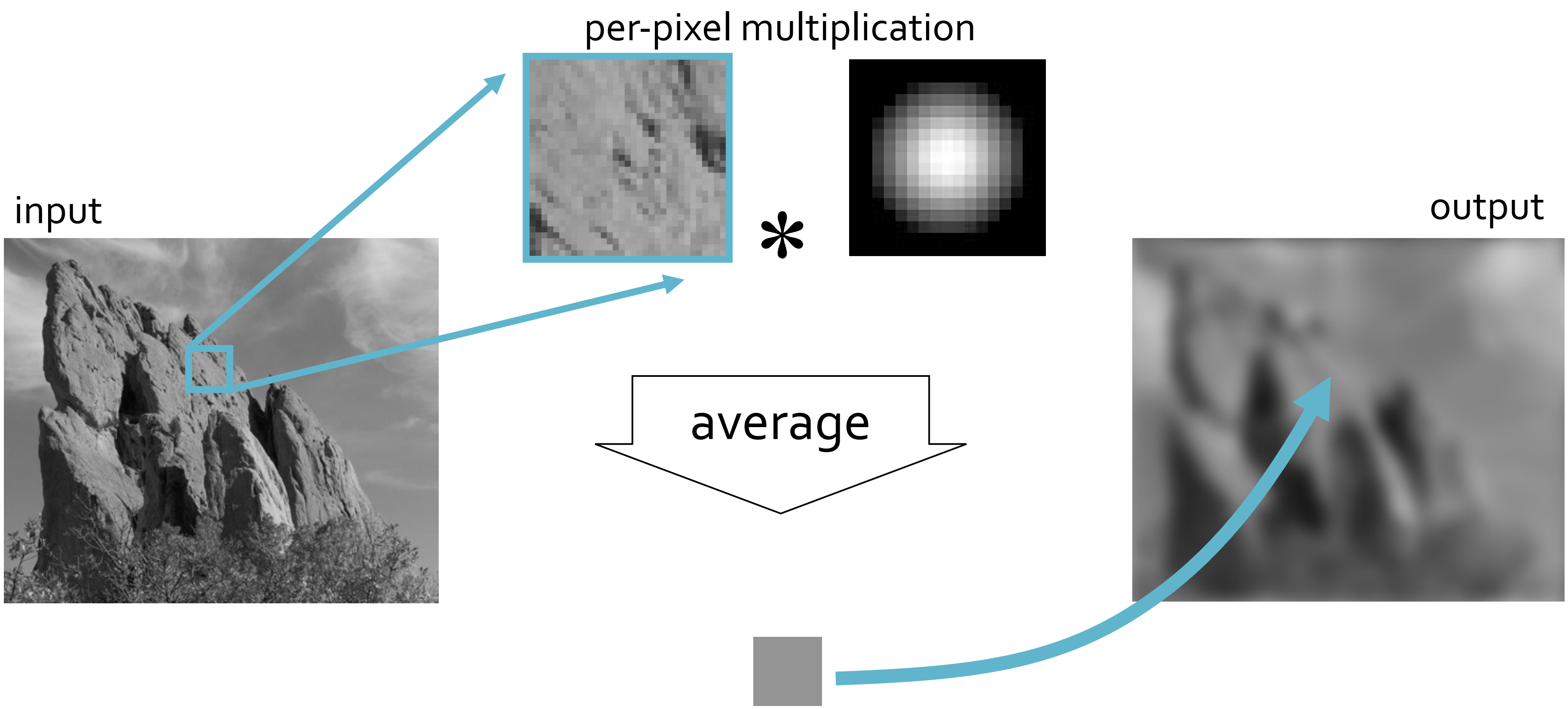
output

Goal: Image Smoothing

Split an image into:

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

Gaussian Blur



Gaussian filtering

BLUR

HALOS

input

smoothed
(*structure, large scale*)

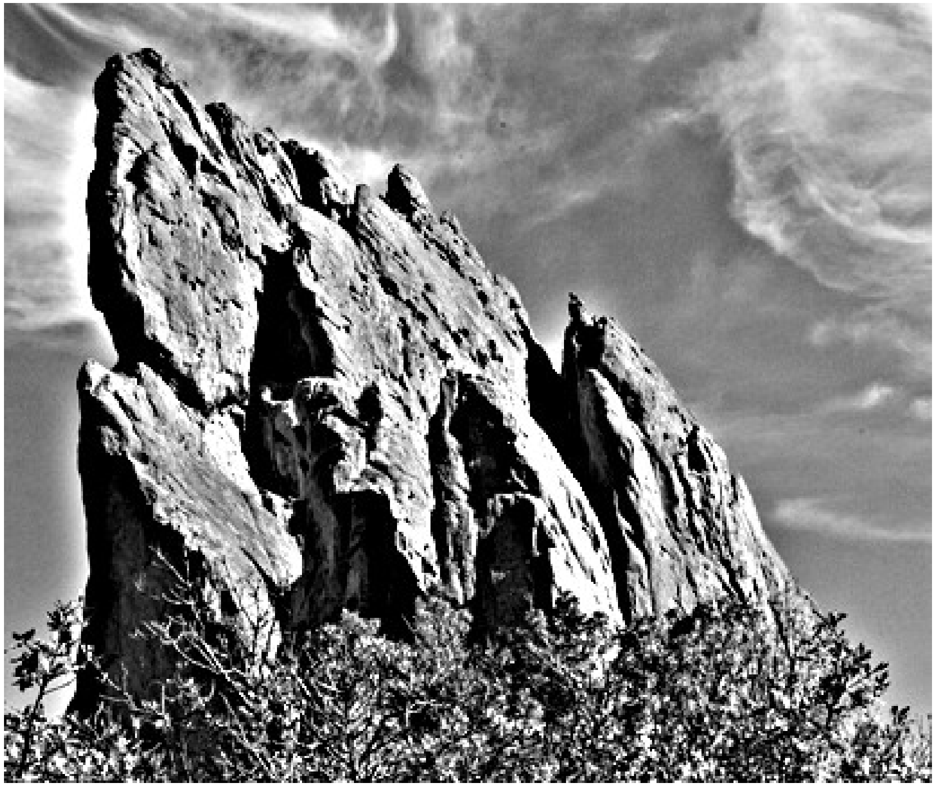
residual
(*texture, small scale*)

Gaussian Convolution

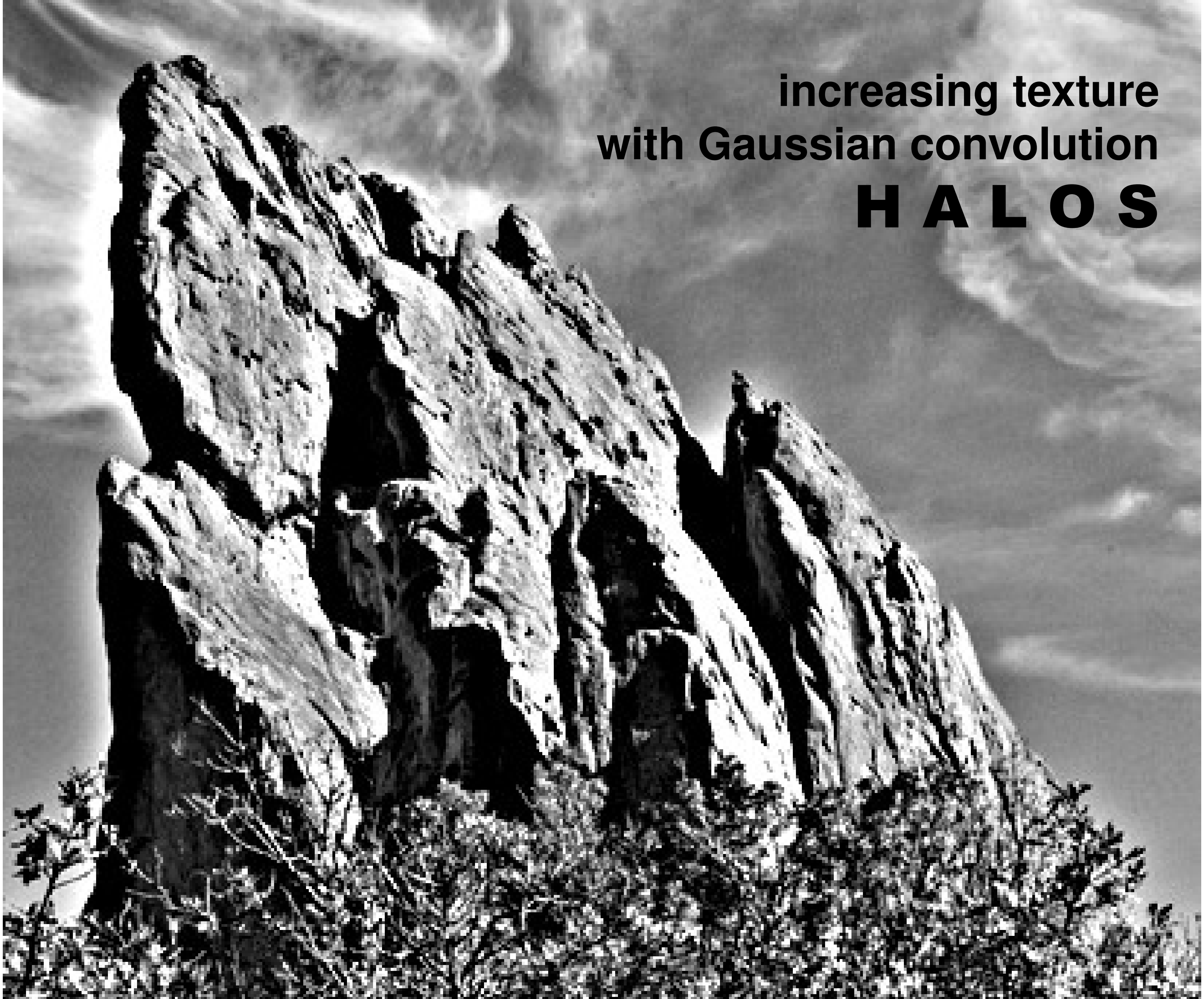
Impact of Blur and Halos

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

Sample manipulation: increasing texture (residual × 3)

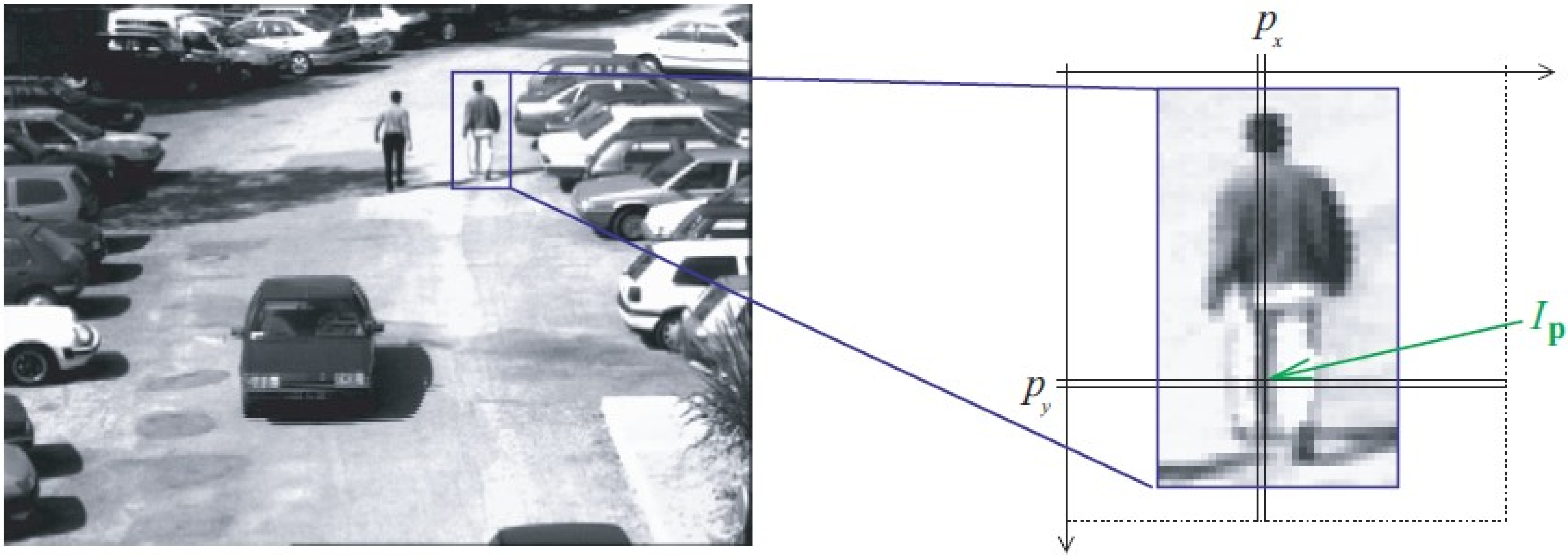



input


increasing texture
with Gaussian convolution
H A L O S

Bilateral filter: General Idea

- An image has two main characteristics
  - The space domain S, which is the set of possible positions in an image. This is related to the resolution, i.e., the number of rows and columns in the image.
  - The intensity domain R, which is the set of possible pixel values. The number of bits used to represent the pixel value may vary. Common pixel representations are unsigned bytes (0 to 255) and floating point.


p_x
p_y
$I_{\mathbf{p}}$

Bilateral filter: General Idea

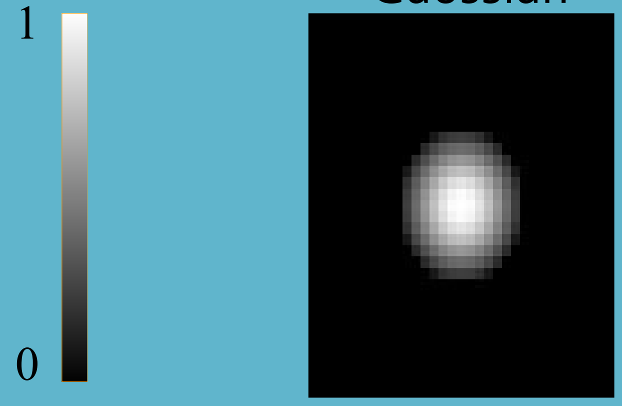
- Every sample is replaced by a weighted average of its neighbors,
- These weights reflect two forces
  - How close are the neighbor and the center sample, so that larger weight to closer samples,
  - How similar are the neighbor and the center sample – larger weight to similar samples.
- All the weights should be normalized to preserve the local mean.

Revisit Gaussian Blur

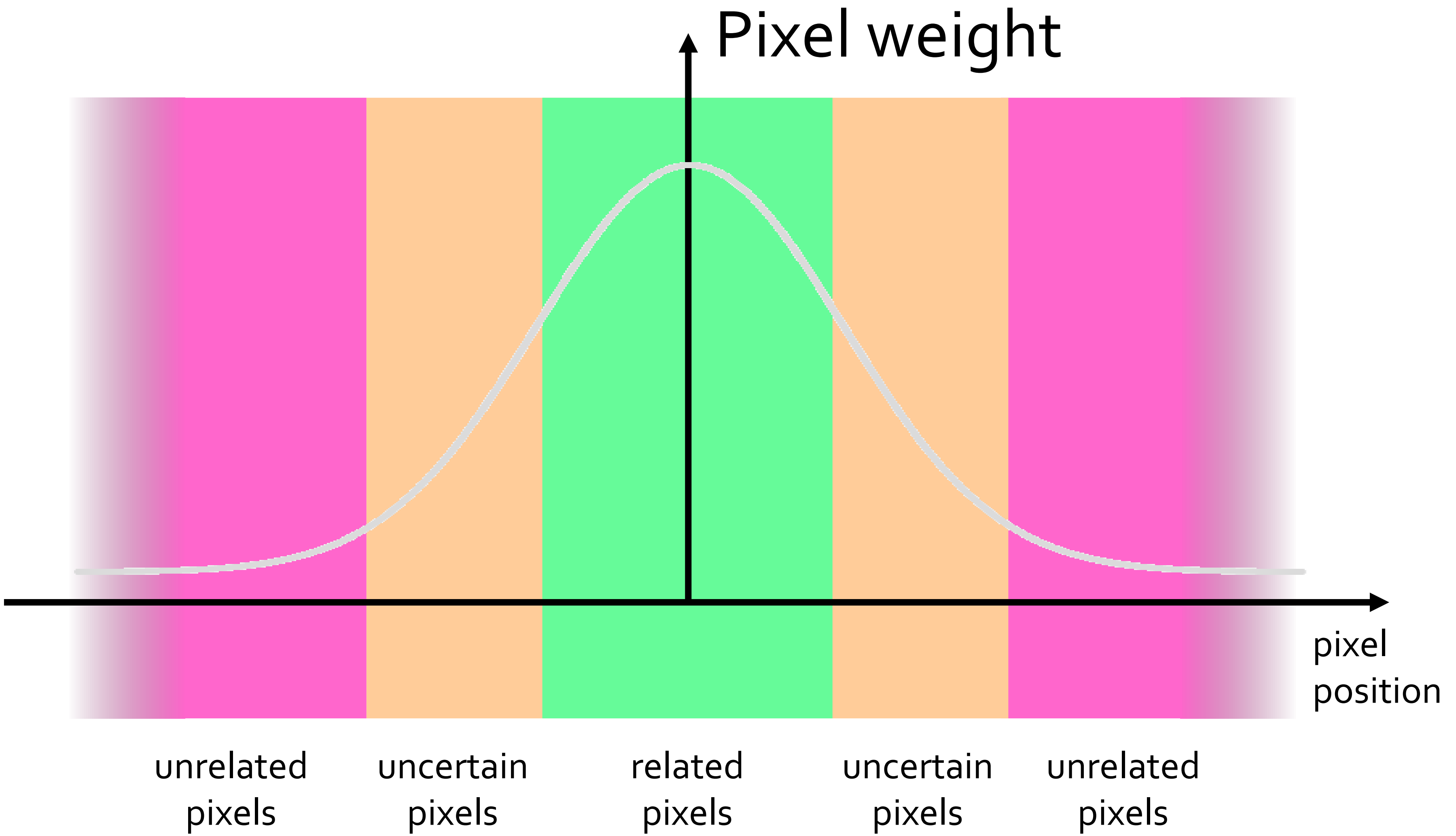
Weighted average of pixels.

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

normalized Gaussian



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

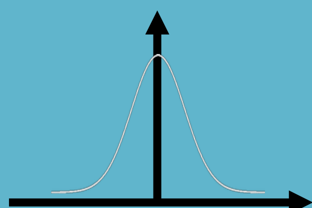


Spatial Parameter

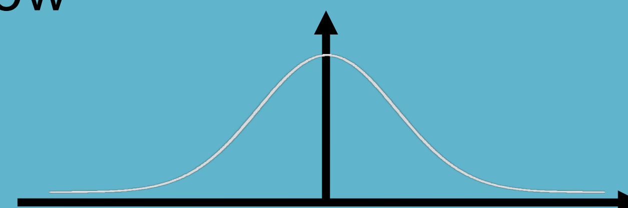
input

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

size of the window



small σ



large σ

limited smoothing

strong smoothing

How to set σ

- Depends on the application.

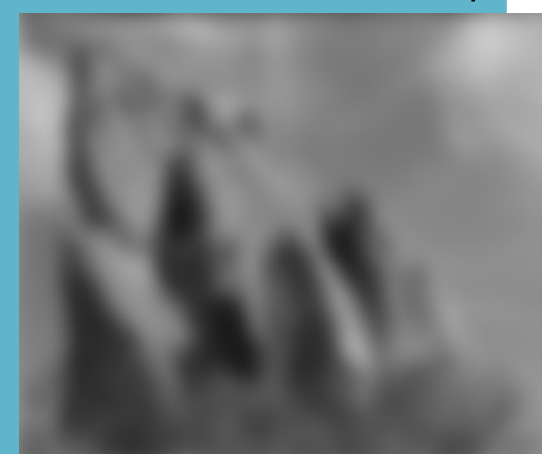
- Common strategy: proportional to image size
  - e.g. 2% of the image diagonal
  - property: independent of image resolution

Properties of Gaussian Blur

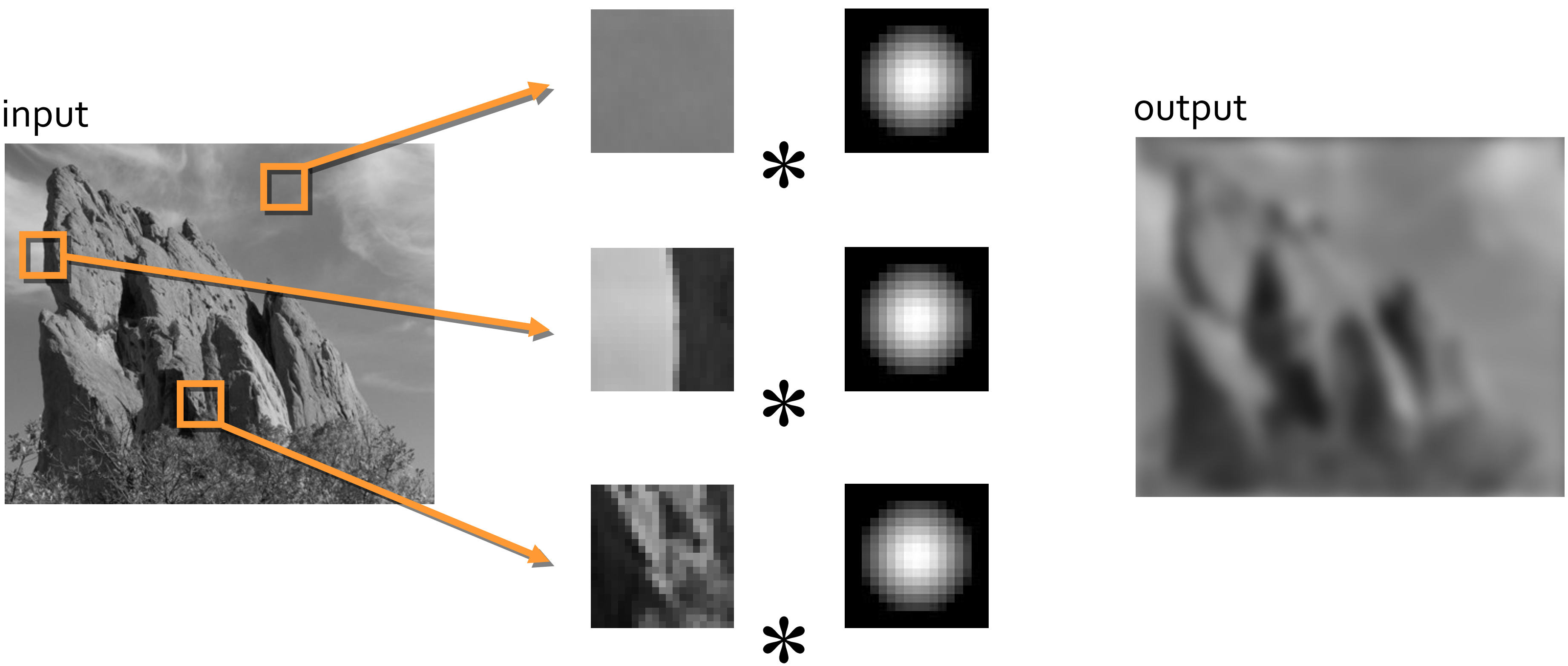
- Does smooth images
- But smoothes too much: **edges are blurred**.
  - Only spatial distance matters
  - No edge term

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

output

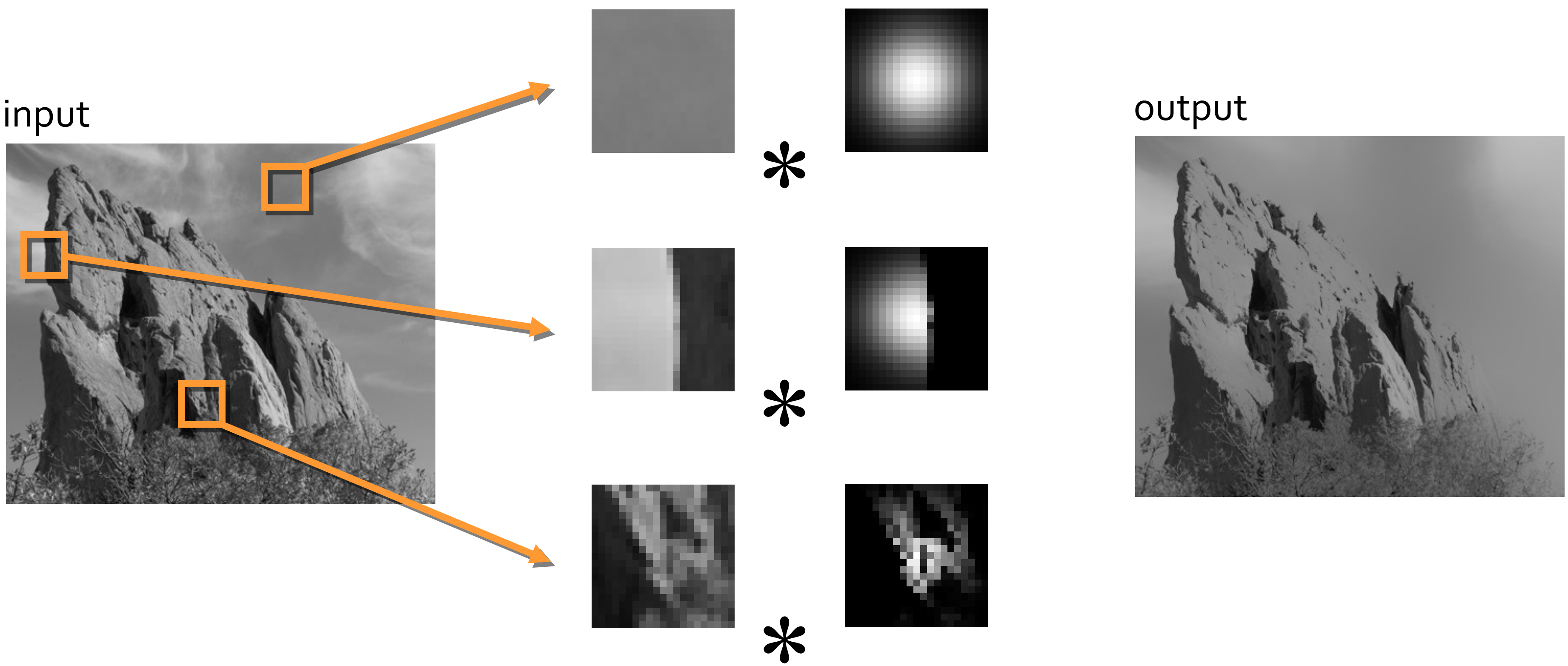


Blur Comes from Averaging across Edges



Same Gaussian kernel everywhere.

Bilateral Filter
No Averaging across Edges



The kernel shape depends on the image content.

Bilateral Filter Definition: an Additional Edge Term

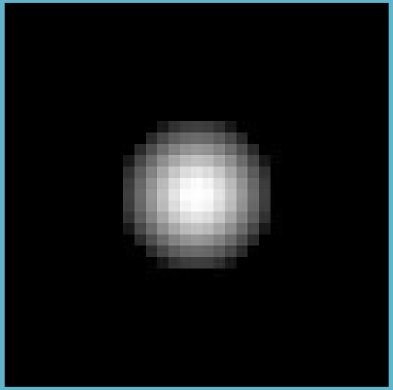
Same idea: **weighted average of pixels**.

new | not new | new

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

normalization factor

space weight



Intensity weight

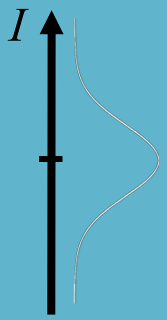
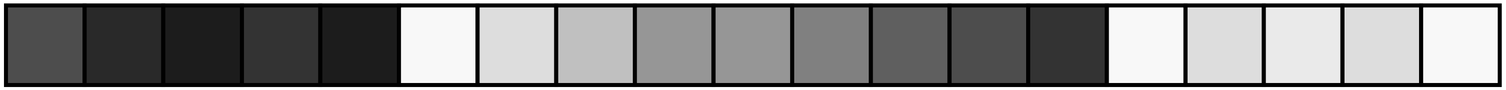
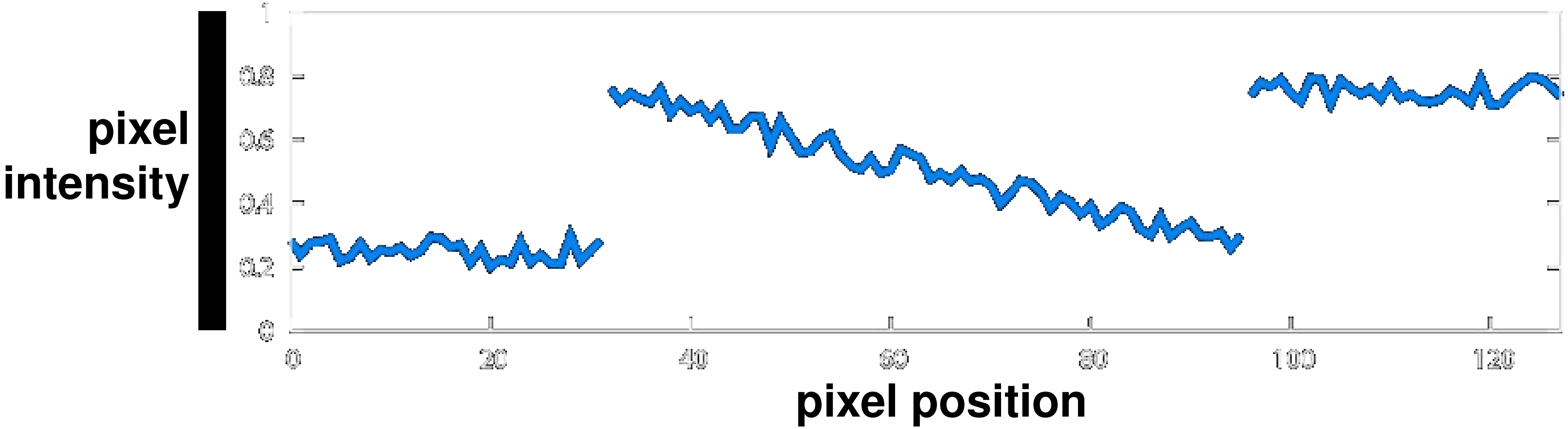


Illustration a 1D Image

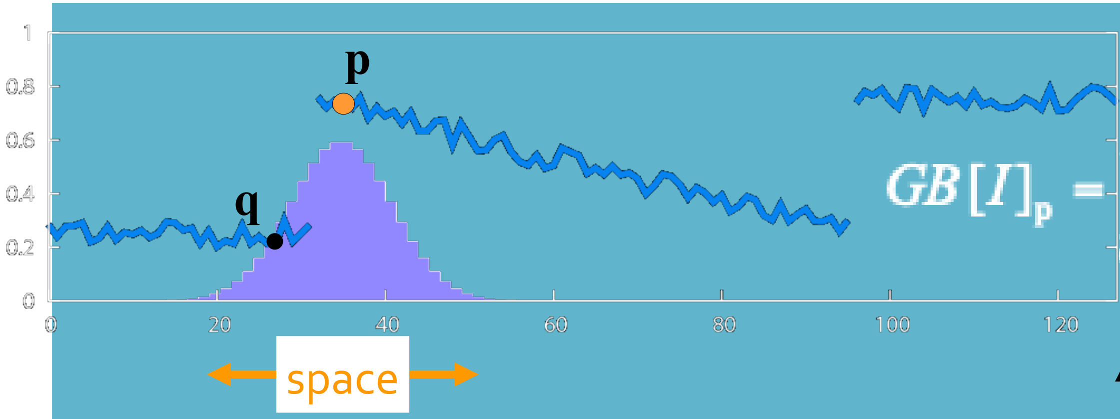
- 1D image = line of pixels



- Better visualized as a plot



Gaussian Blur and Bilateral Filter

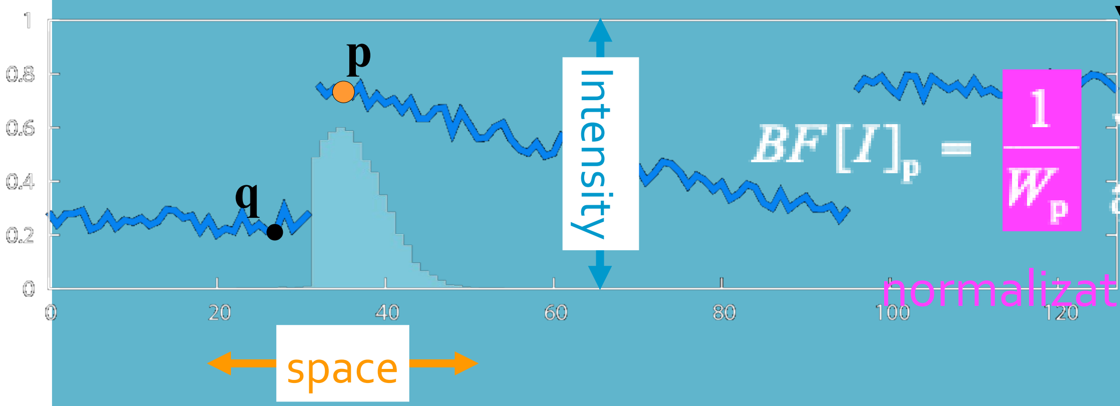


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\|\mathbf{p}-\mathbf{q}\|) I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



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

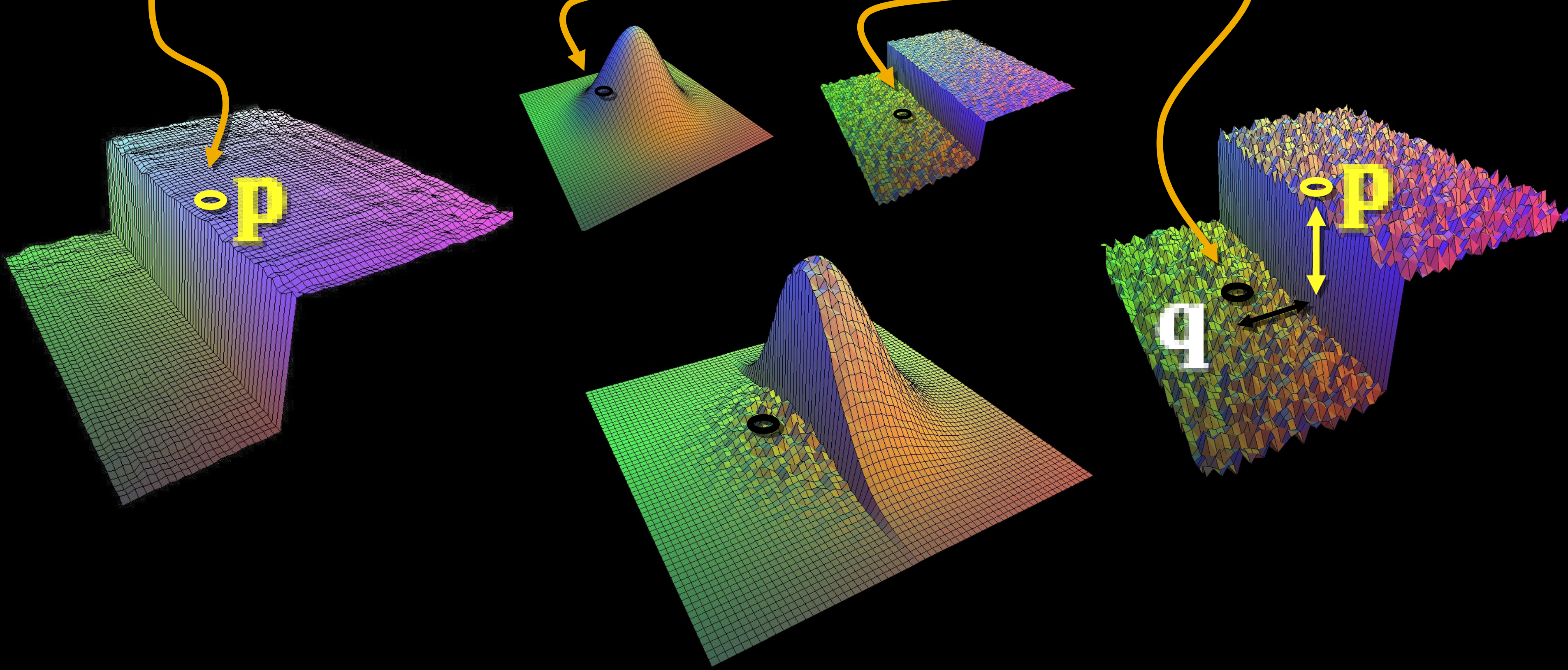
normalization

space

Intensity

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 Intensity Parameters

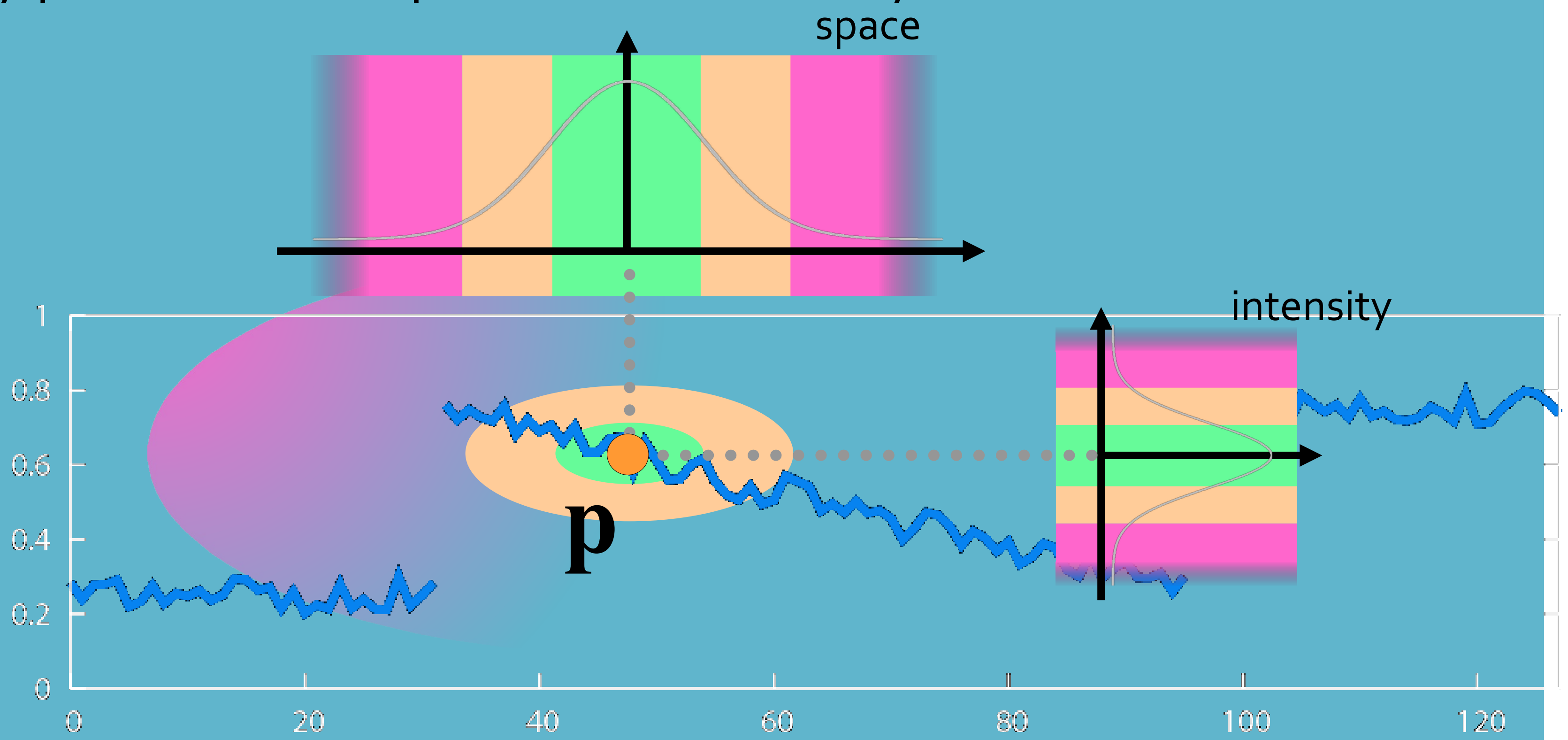
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}}$$

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

- intensity σ_r : amplitude extent of an edge

Influence of Pixels

Only pixels close in space and in intensity are considered.



Exploring the Parameter Space

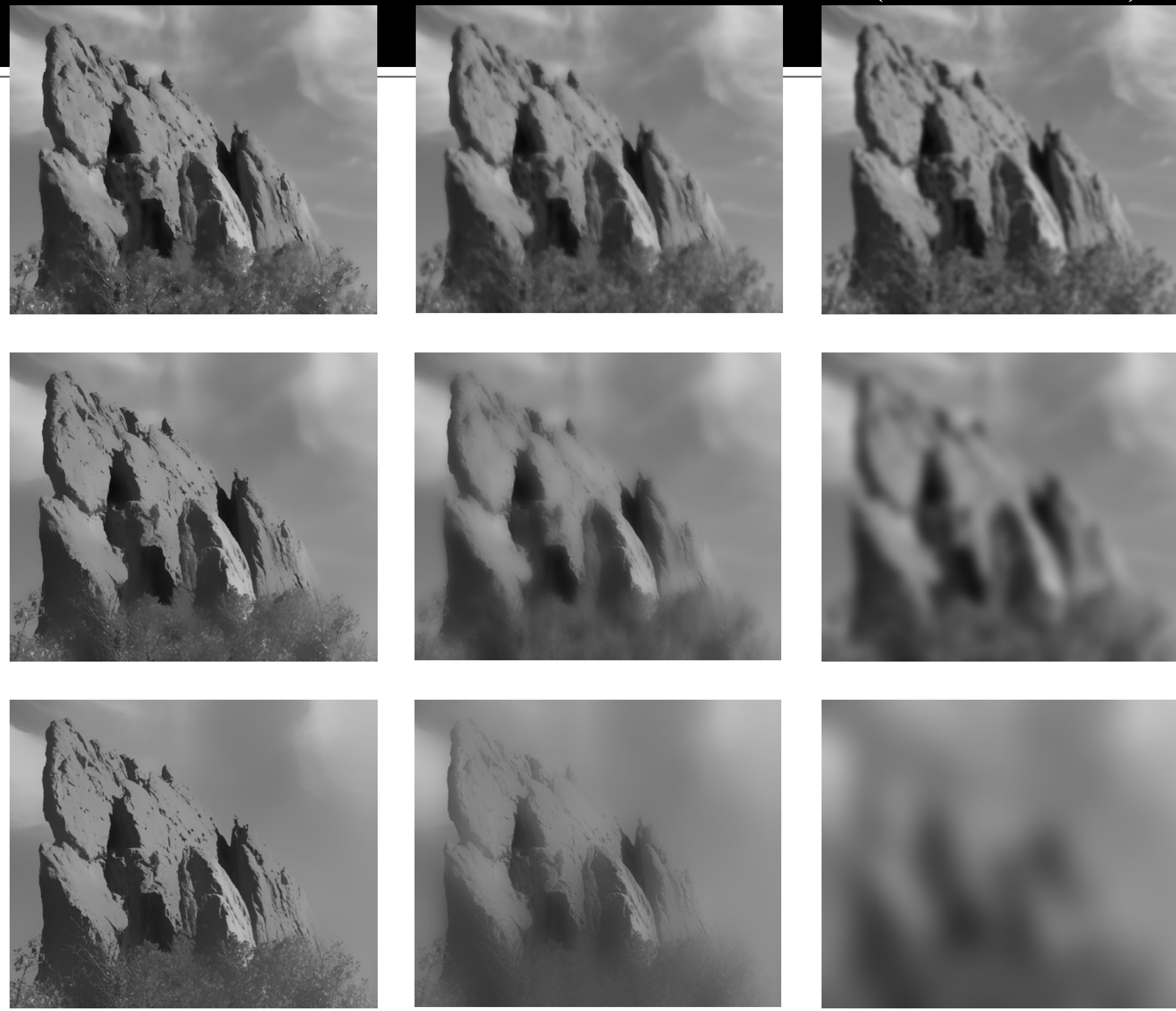
input

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

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$



Varying the Intensity Parameter

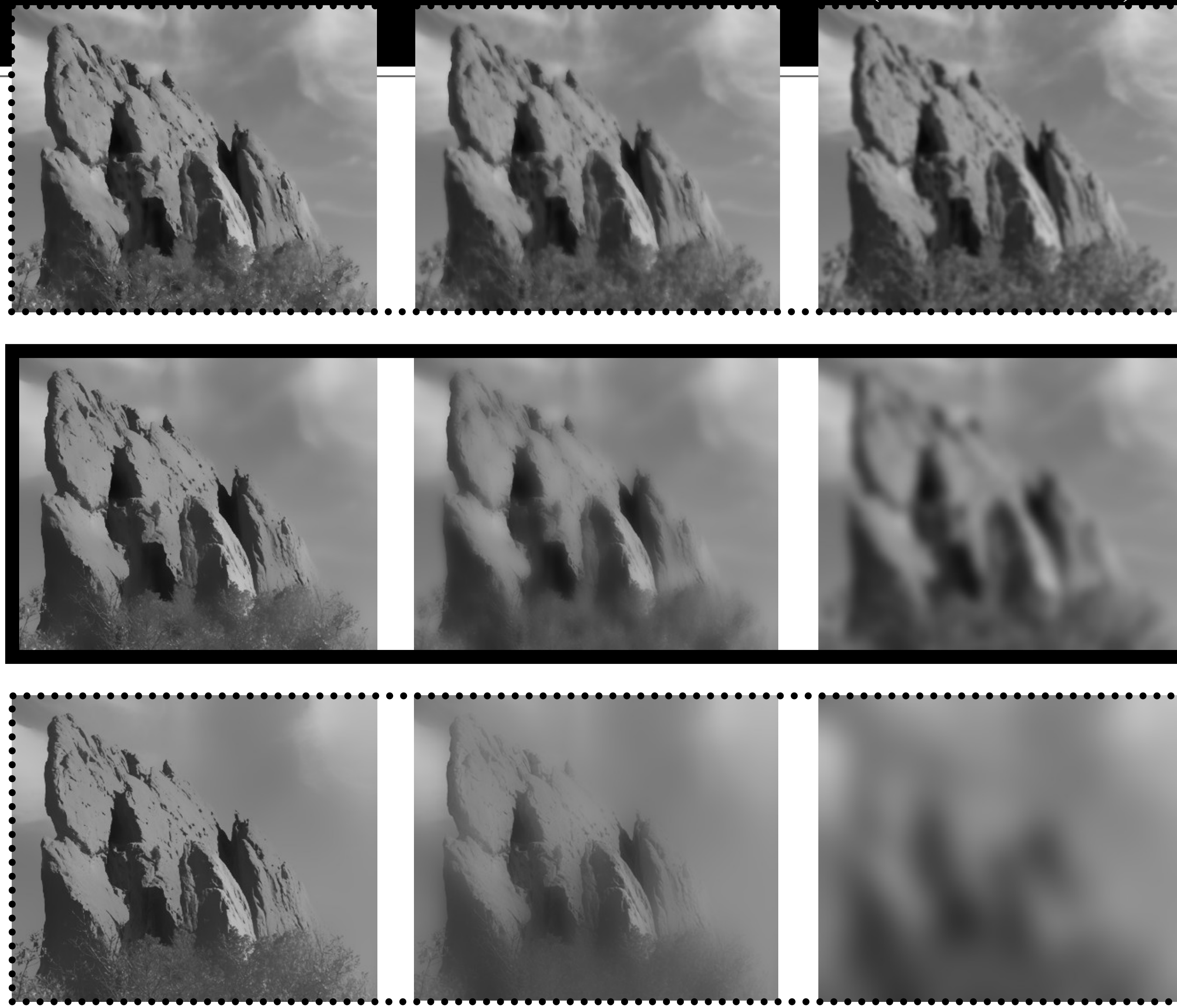
input

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

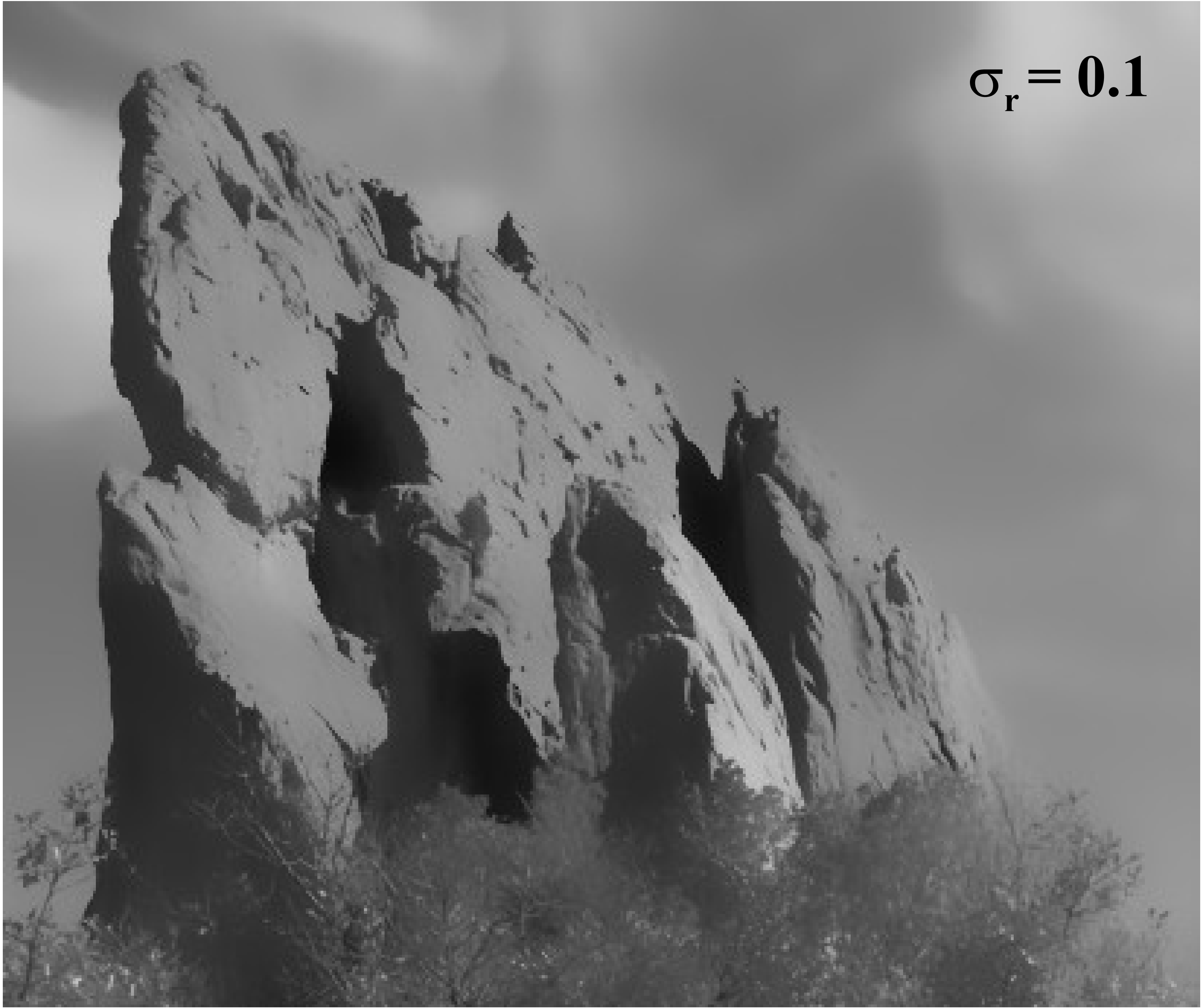
$\sigma_s = 2$

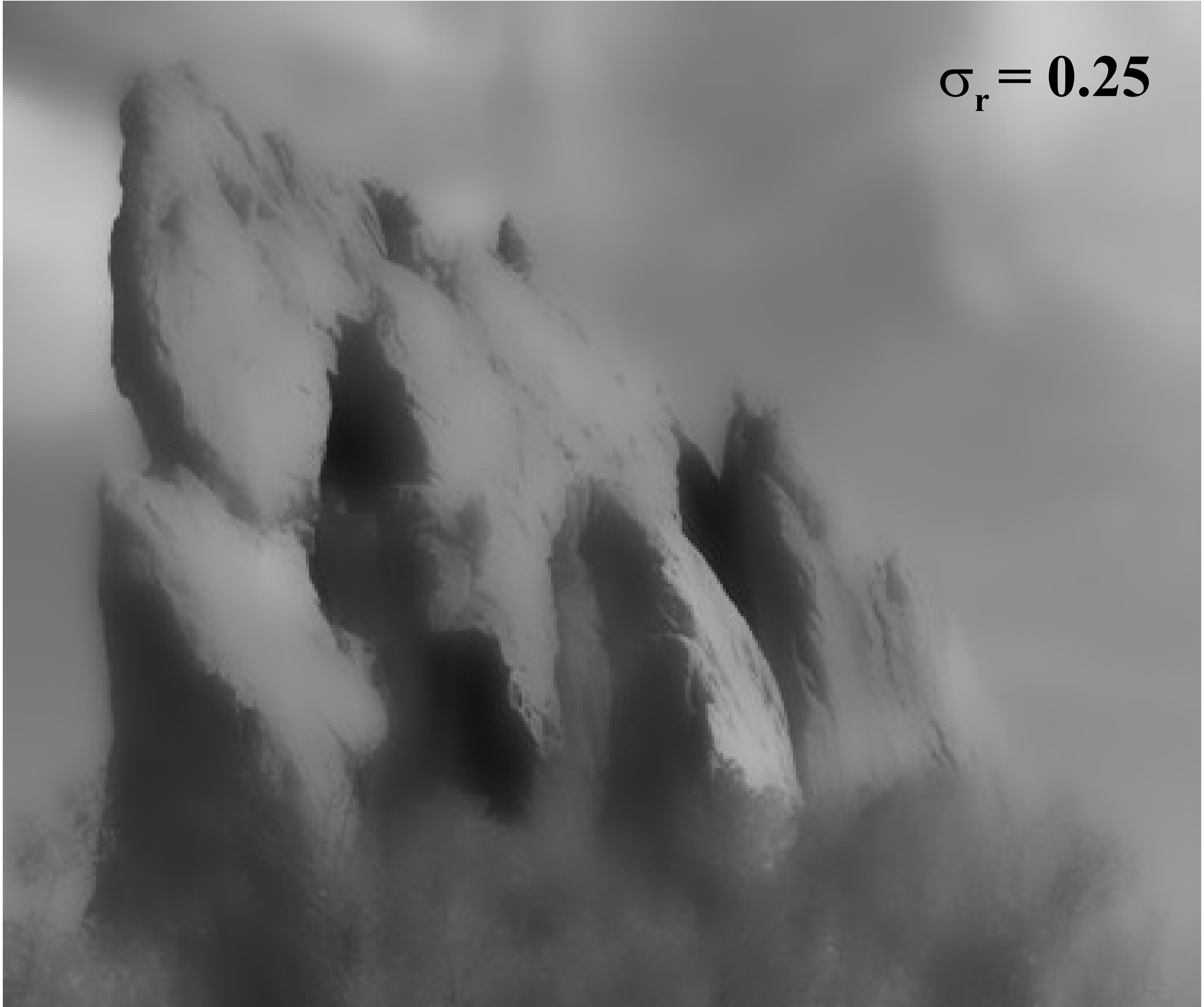
$\sigma_s = 6$

$\sigma_s = 18$




input


$\sigma_r = 0.1$


$\sigma_r = 0.25$


$\sigma_r = \infty$
(Gaussian blur)

Varying the Space Parameter

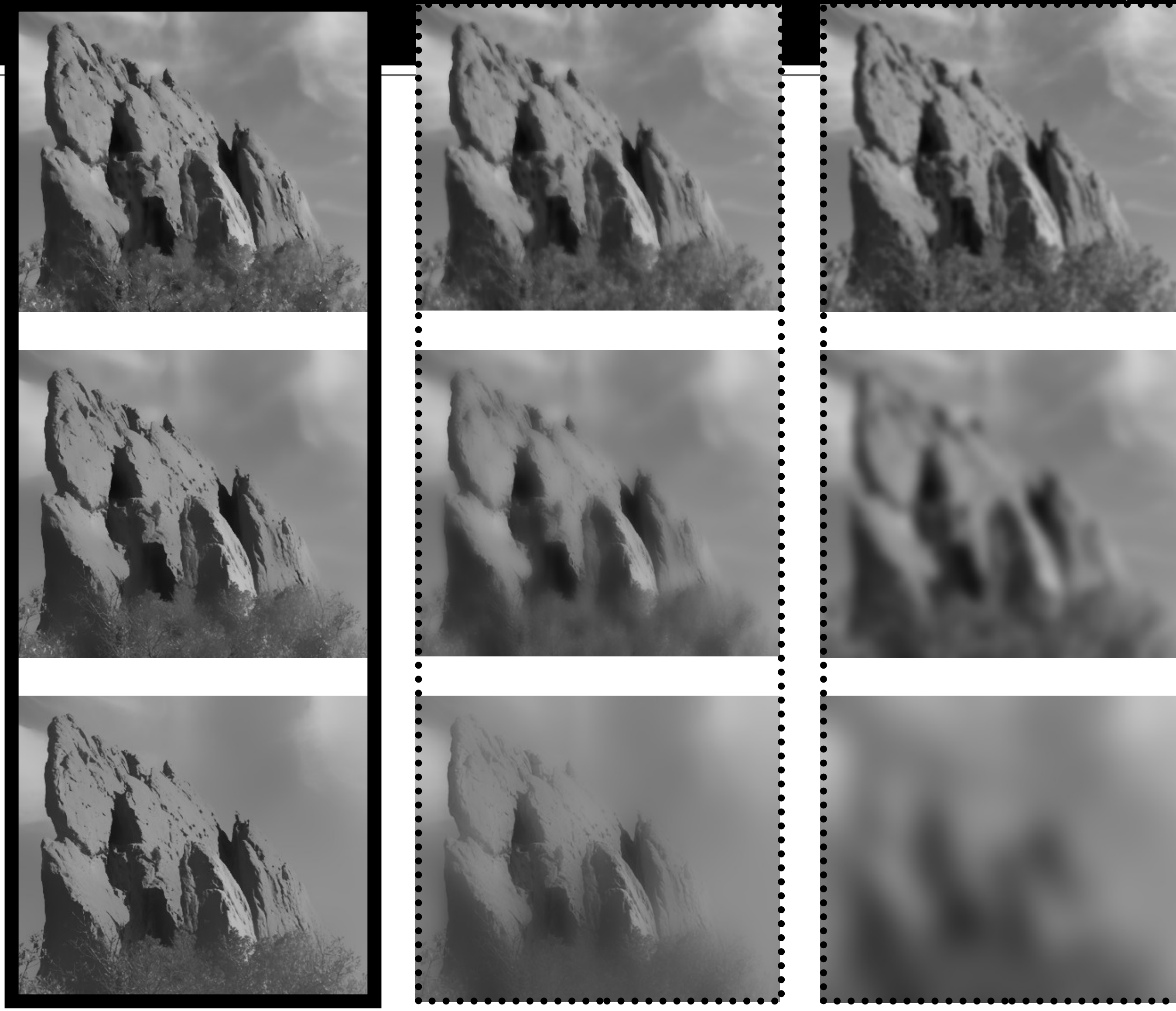
input

$\sigma_r = 0.1$ | $\sigma_r = 0.25$ | $\sigma_r = \infty$ (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
- intensity parameter: proportional to edge amplitude
  - e.g., mean or median of image gradients
- independent of resolution and exposure

Iterating the Bilateral Filter

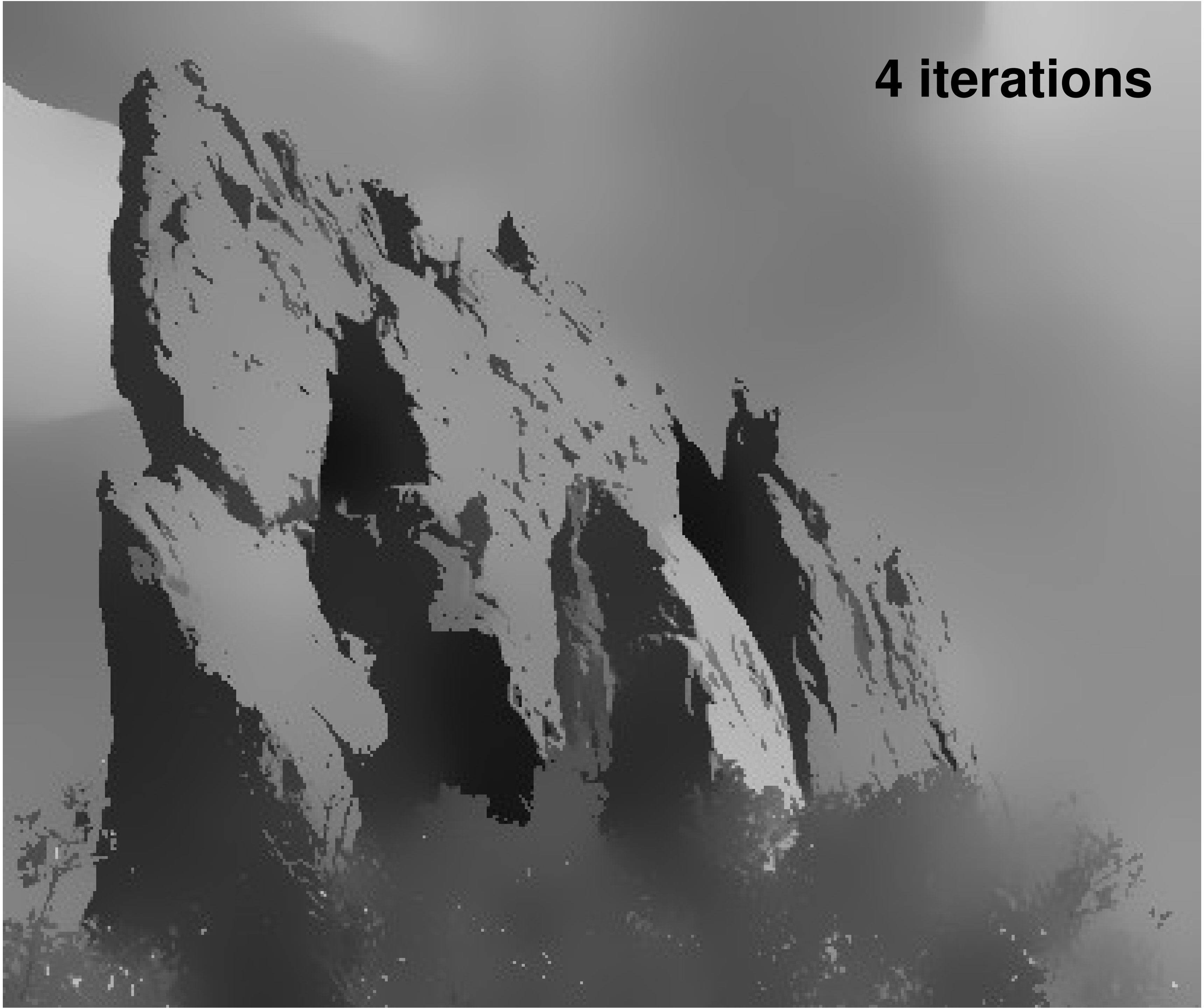
$$I_{(n+1)} = BF\left[I_{(n)}\right]$$

- 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

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

scalar

For color images

color difference

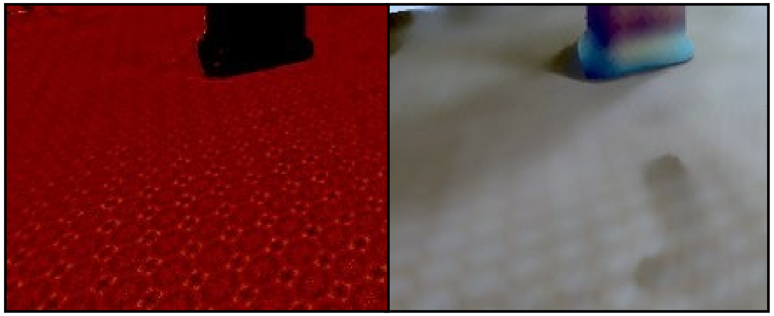
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) G_{\sigma_r}(\| \mathbf{C}_{\mathbf{p}} - \mathbf{C}_{\mathbf{q}} \|) \mathbf{C}_{\mathbf{q}}$$

3D vector
(RGB, Lab)

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

Overview

- Denoising
- Tone mapping
- Relighting & texture editing



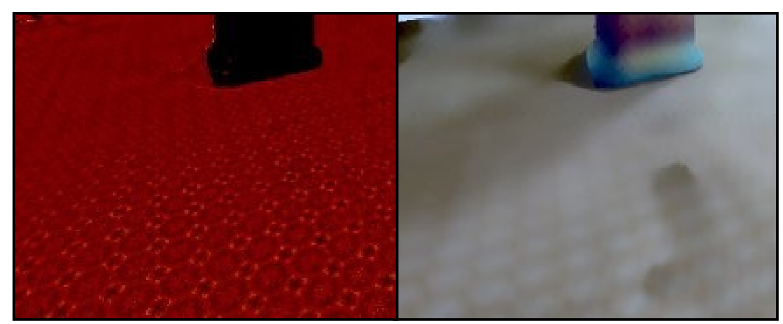
Overview

- **Denoising**

Not most powerful application
Not best denoising, but good & simple

- Tone mapping

- Relighting & texture editing



Basic denoising

Noisy input

Bilateral filter 7x7 window

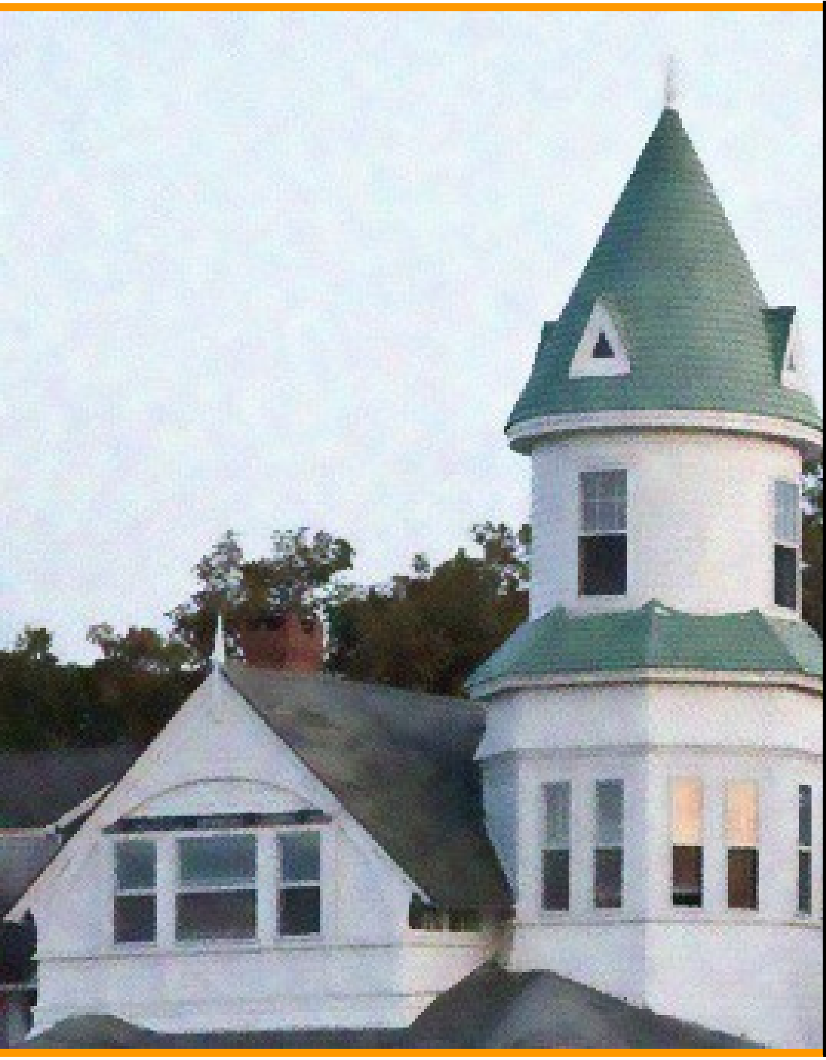
Basic denoising

Bilateral filter

Median 3x3

Basic denoising

Bilateral filter

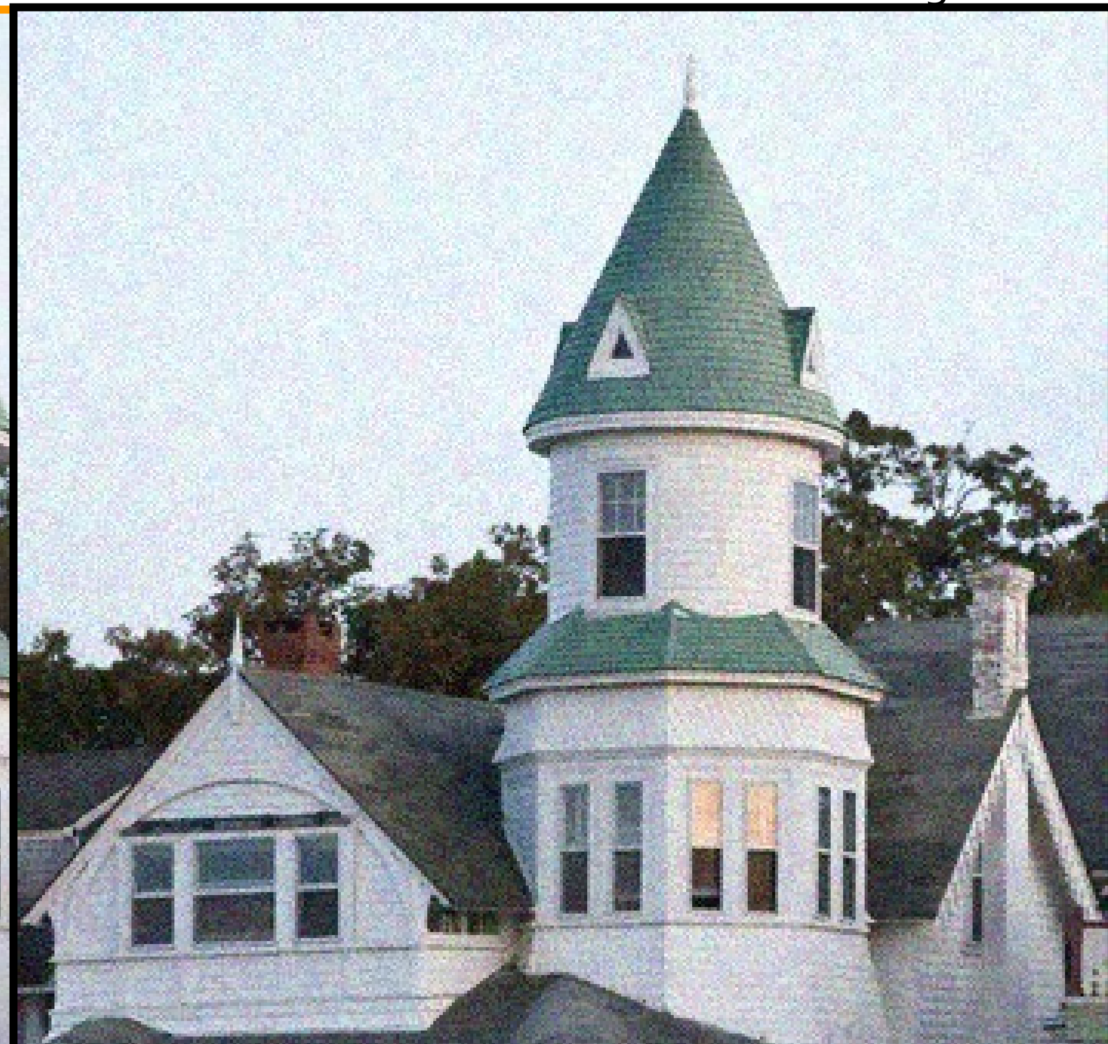


Median 5x5

Basic denoising

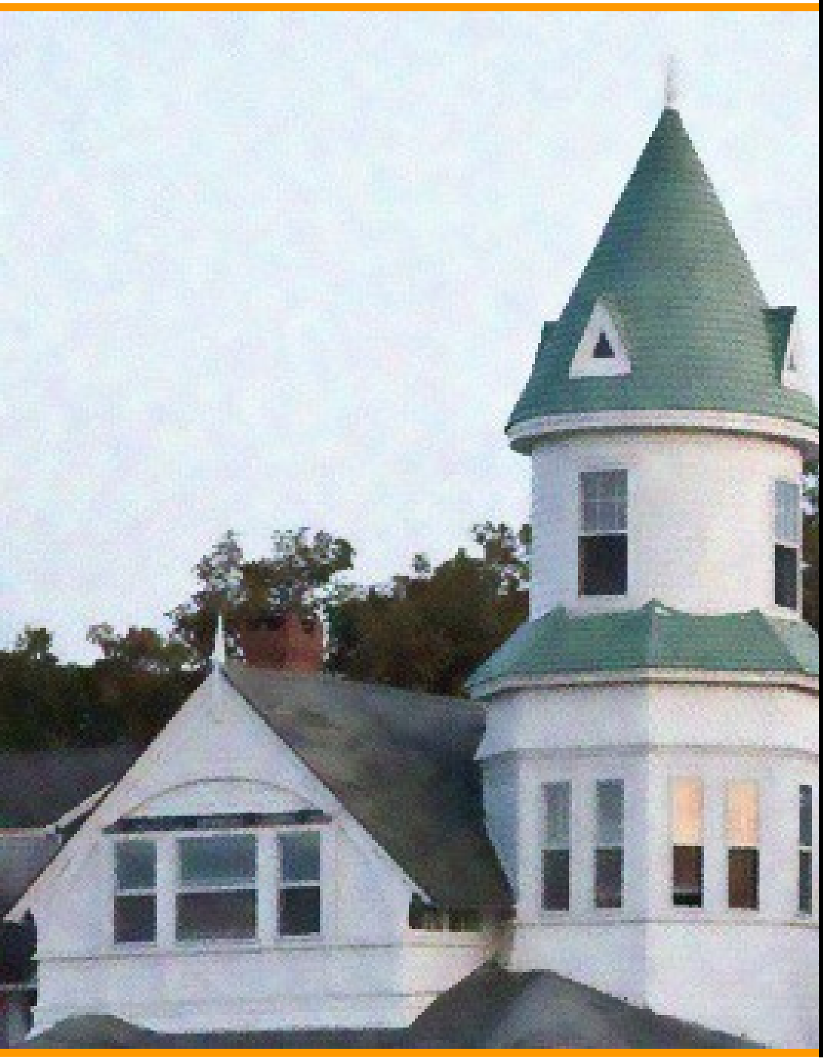
Bilateral filter

Bilateral filter – lower sigma



Basic denoising

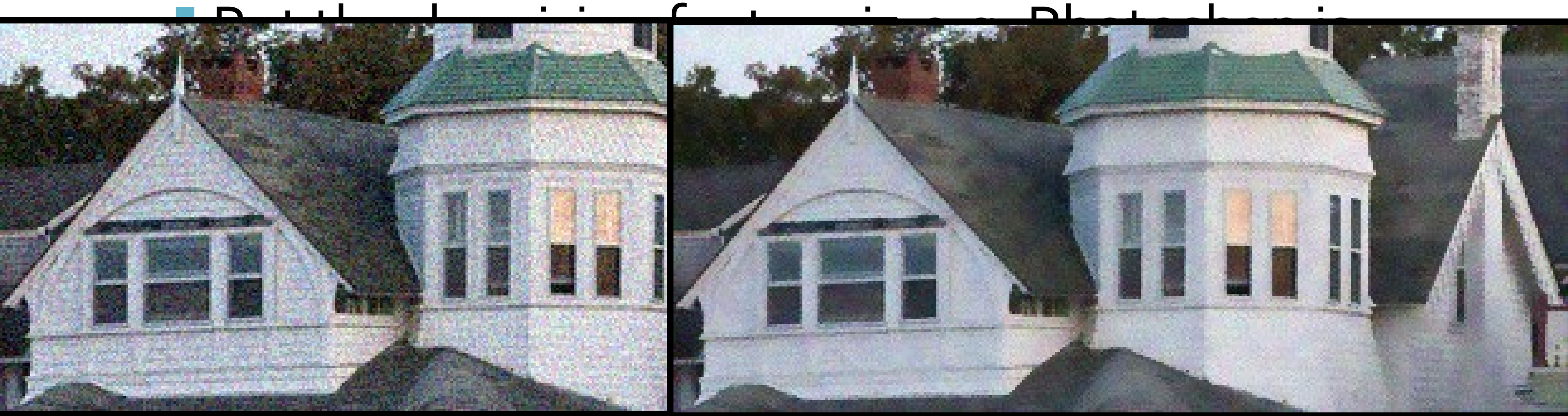
Bilateral filter



Bilateral filter – higher sigma

Denoising

- Small spatial sigma (e.g. 7x7 window)
- Adapt intensity sigma to noise level
- Maybe not best denoising method, but best simplicity/quality tradeoff
  - No need for acceleration (small kernel)

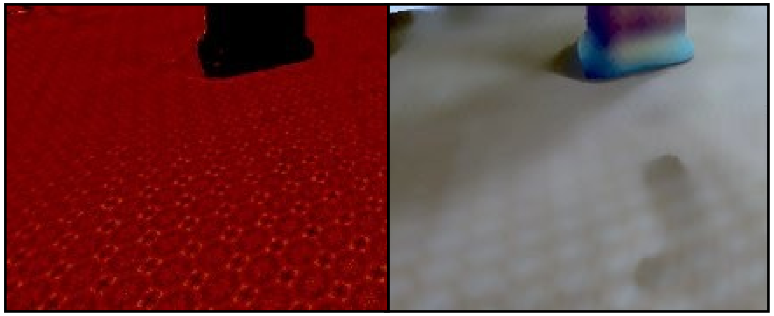


Overview

- Denoising

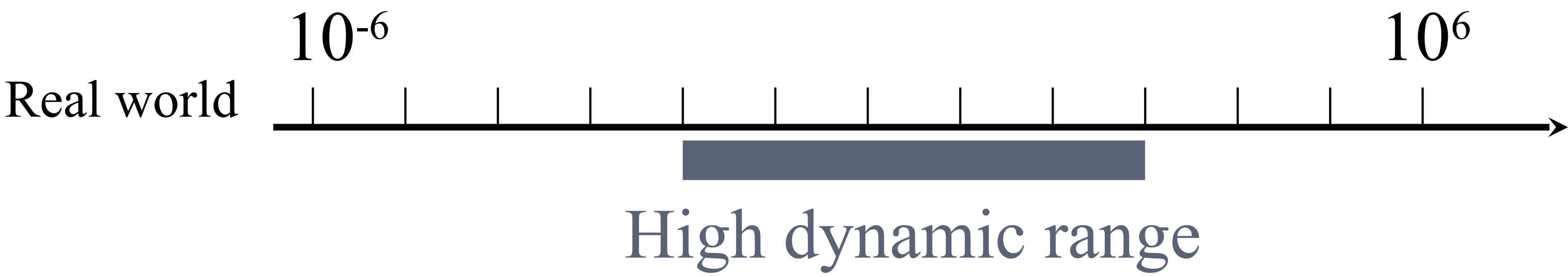
- **Tone mapping**

- Relighting & texture editing



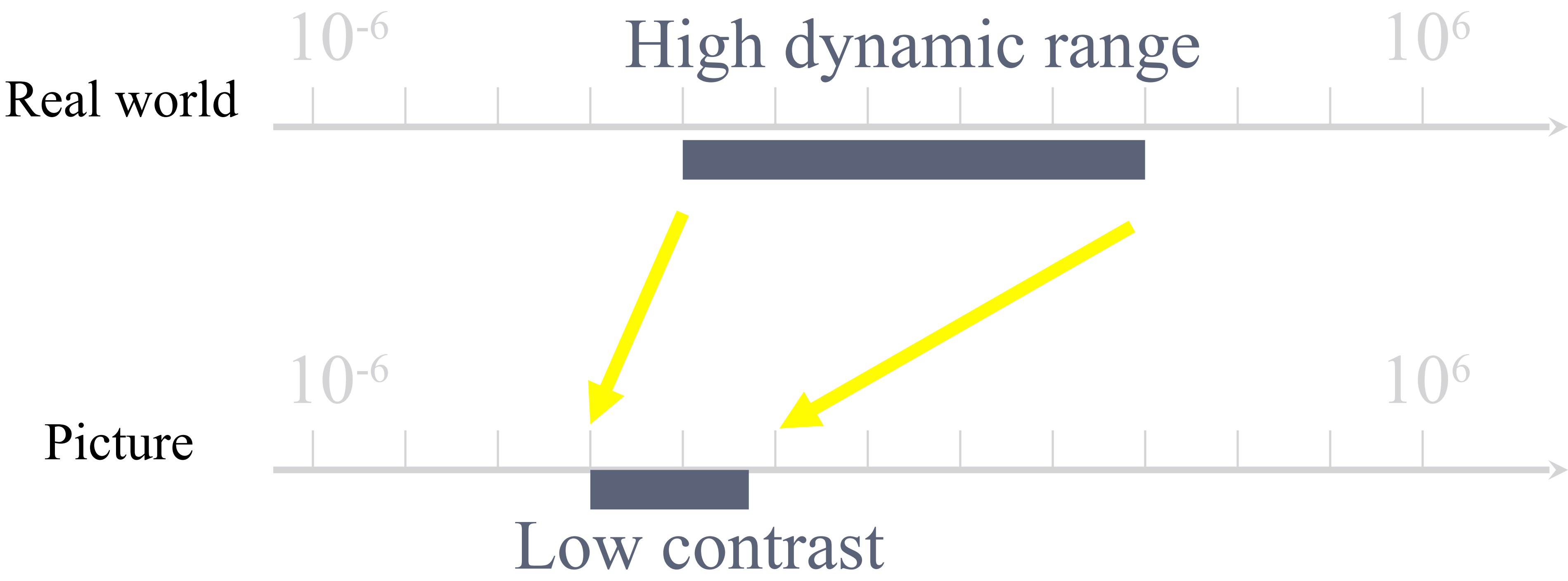
Real world dynamic range

- Eye can adapt from ~ 10^{-6} to 10^{6} cd/m^2
- Often 1 : 10,000 in a scene



Problem: Contrast reduction

- Match limited contrast of the medium
- Preserve details

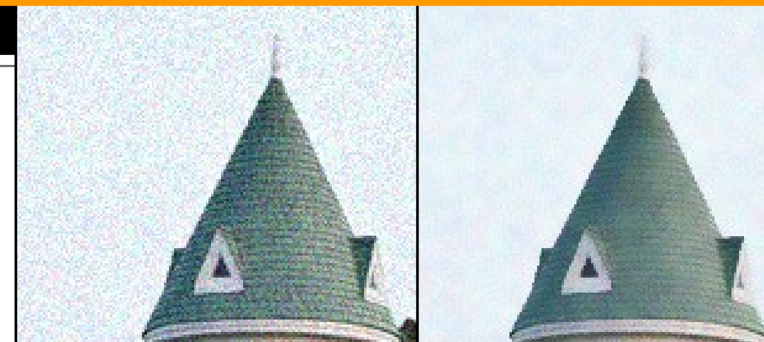


Overview

- Denoising
- Tone mapping
- Relighting & texture editing

Overview

- **Denoising**



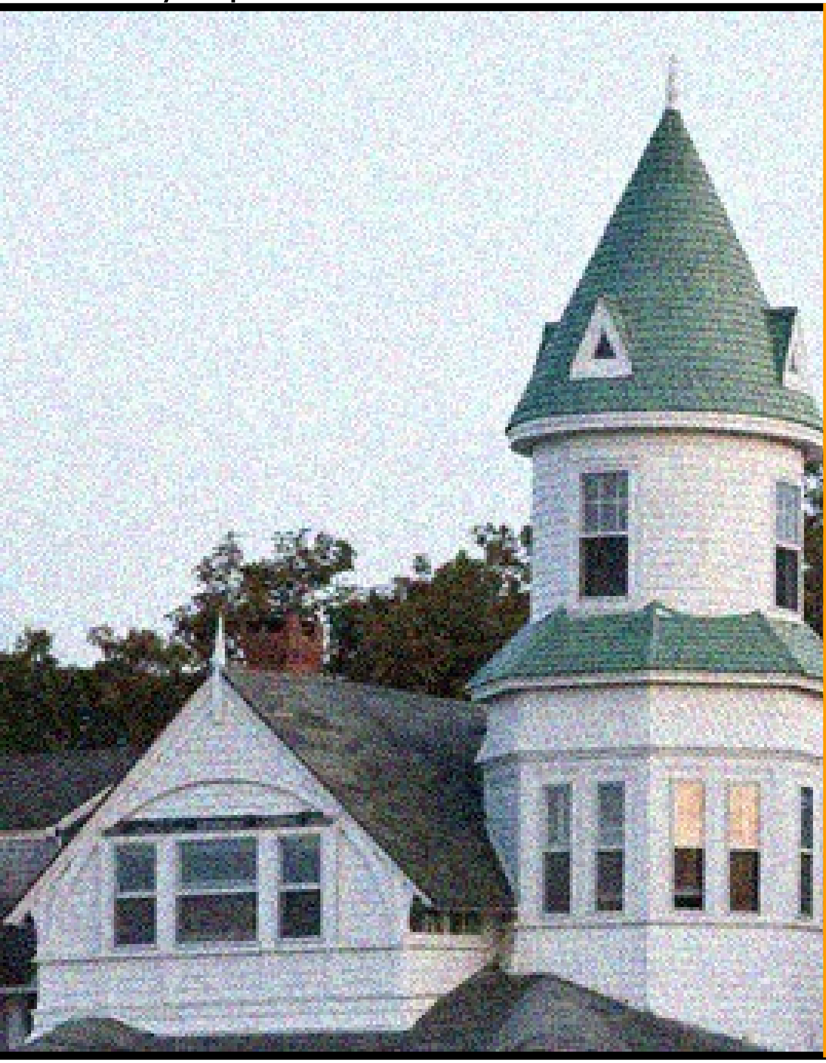
Not most powerful application
Not best denoising, but good & simple

- Tone mapping

- Relighting & texture editing

Basic denoising

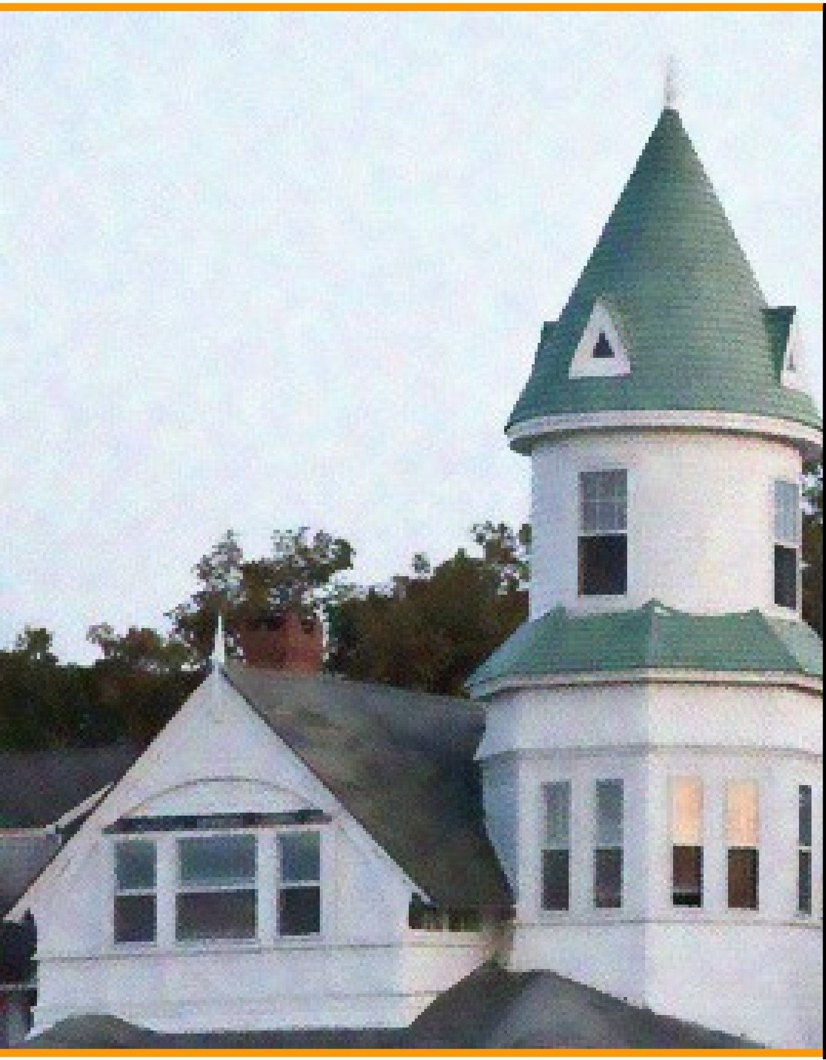
Noisy input



Bilateral filter 7x7 window

Basic denoising

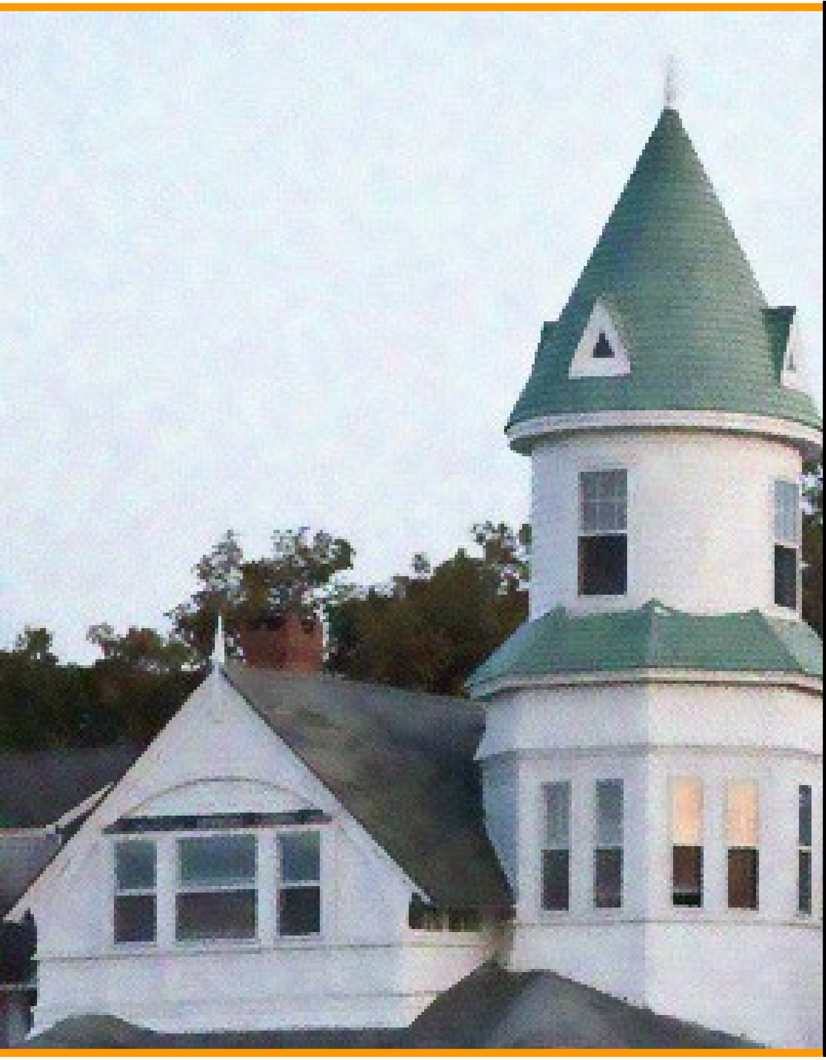
Bilateral filter



Median 3x3

Basic denoising

Bilateral filter

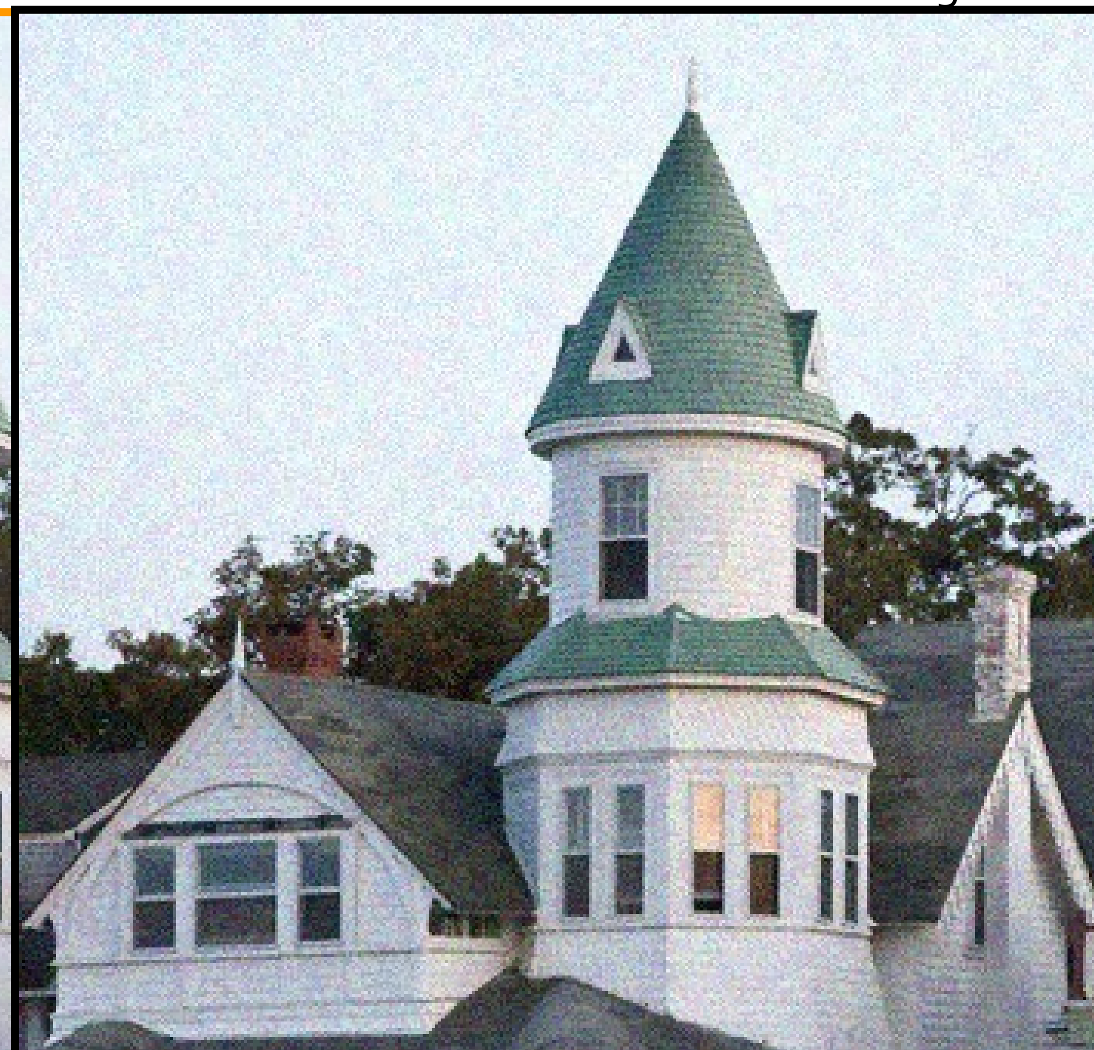


Median 5x5

Basic denoising

Bilateral filter

Bilateral filter – lower sigma



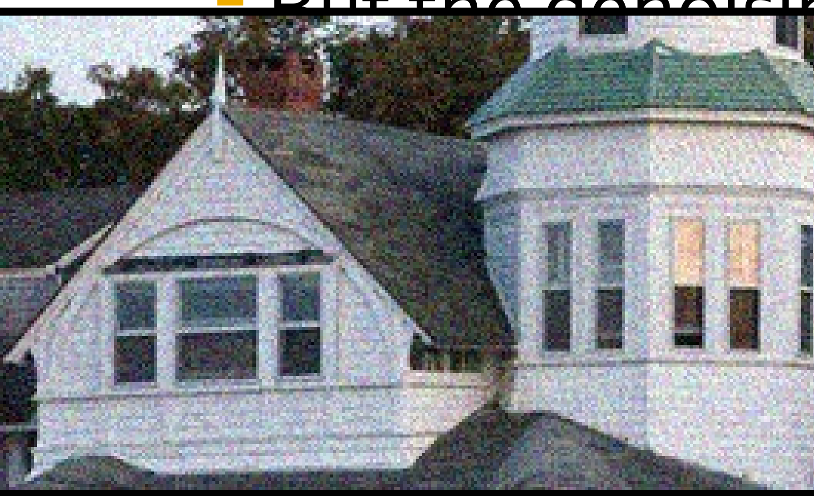
Basic denoising

Bilateral filter

Bilateral filter – higher sigma

Denoising

- Small spatial sigma (e.g. 7x7 window)
- Adapt range sigma to noise level
- Maybe not best denoising method, but best simplicity/quality tradeoff
  - No need for acceleration (small kernel)
  - But the denoising feature in e.g. Photoshop




Overview

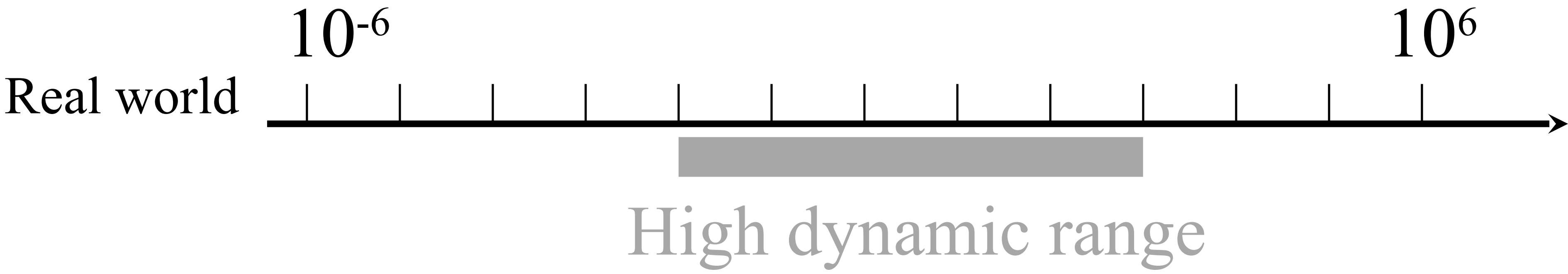
- Denoising

- **Tone mapping**

- Relighting & texture editing

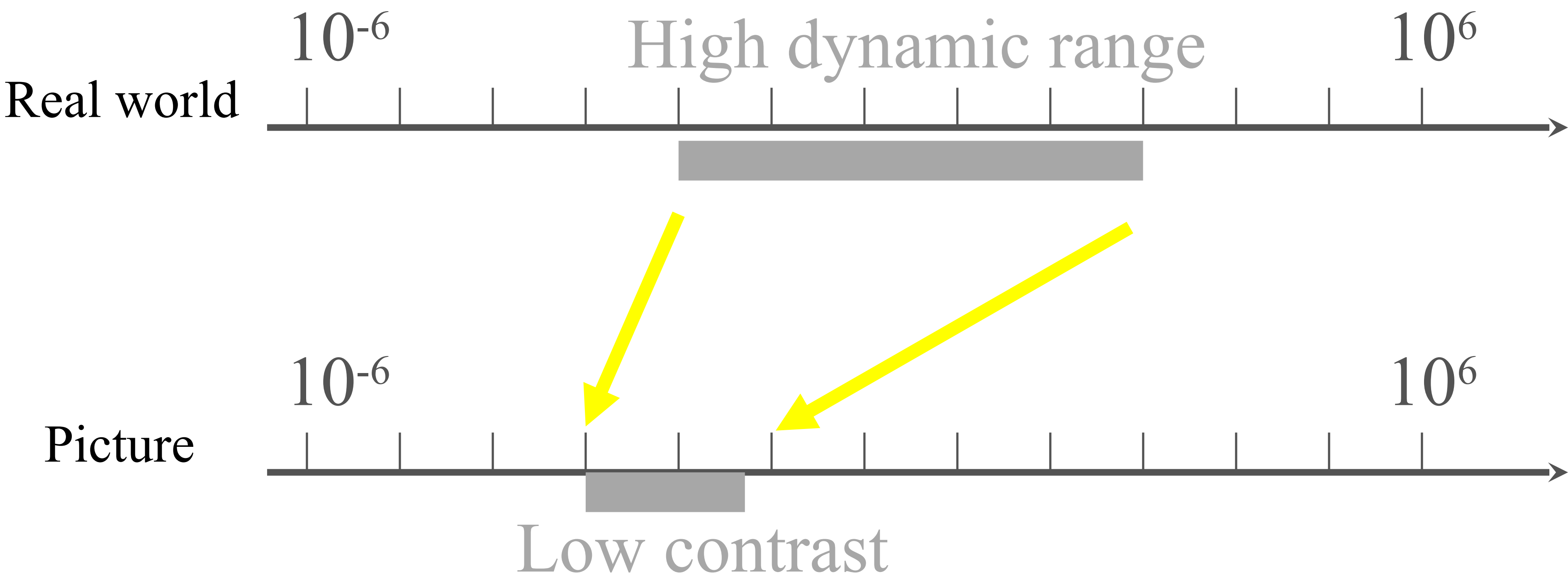
Real world dynamic range

- Eye can adapt from ~ 10^{-6} to 10^{6} cd/m^2
- Often 1 : 10,000 in a scene



Problem: Contrast reduction

- Match limited contrast of the medium
- Preserve details



Tone mapping

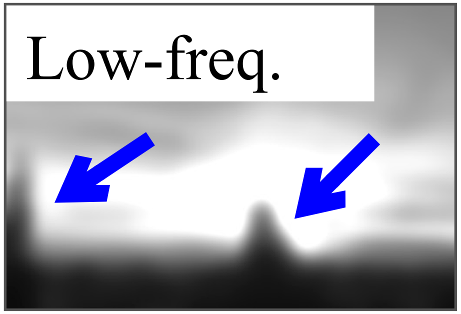
- Input: high-dynamic-range image
  - (floatting point per pixel)

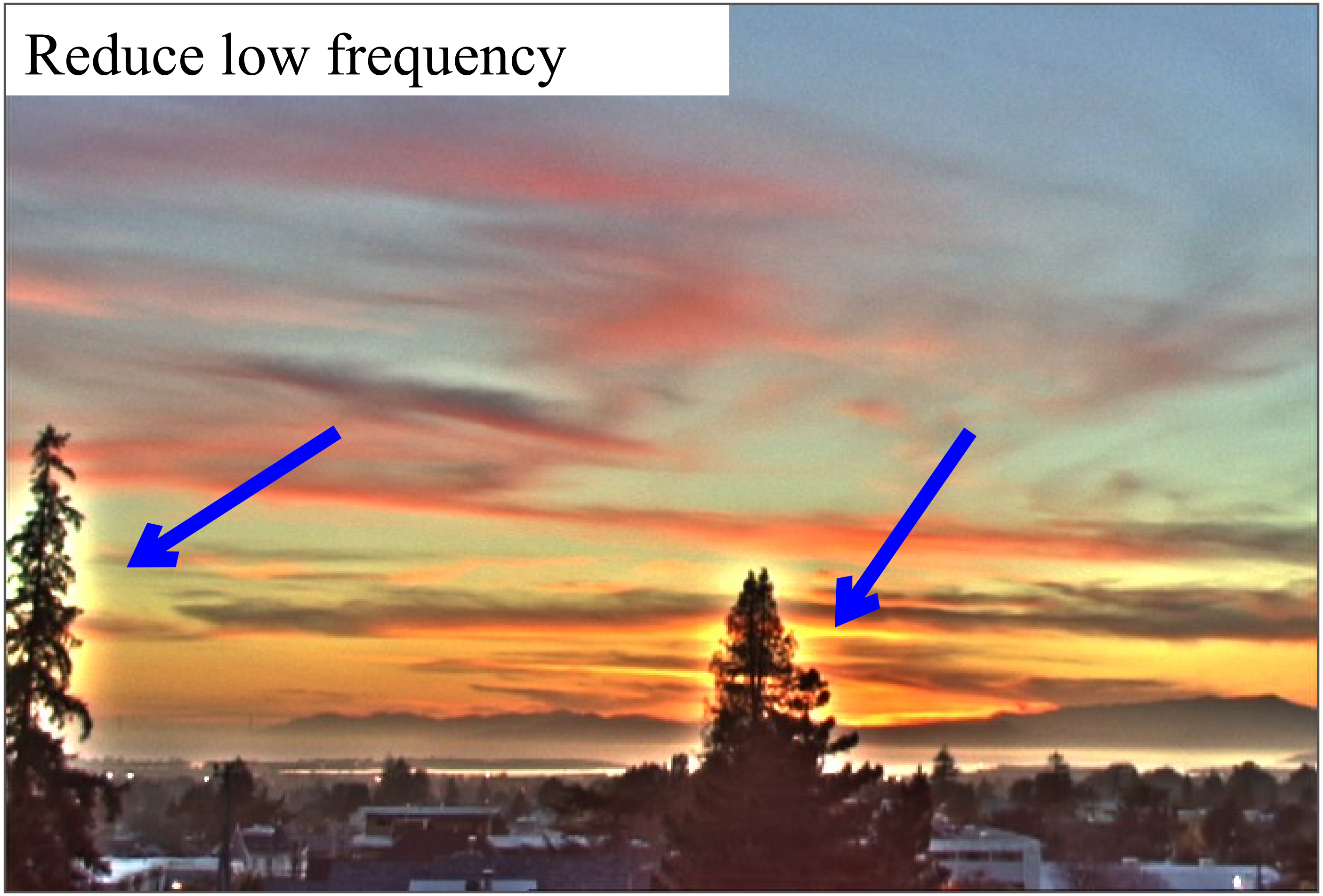
Naïve technique

- Scene has *1:10,000* contrast, display has *1:100*
- Simplest contrast reduction?

The halo nightmare

- For strong edges
- Because they contain high frequency

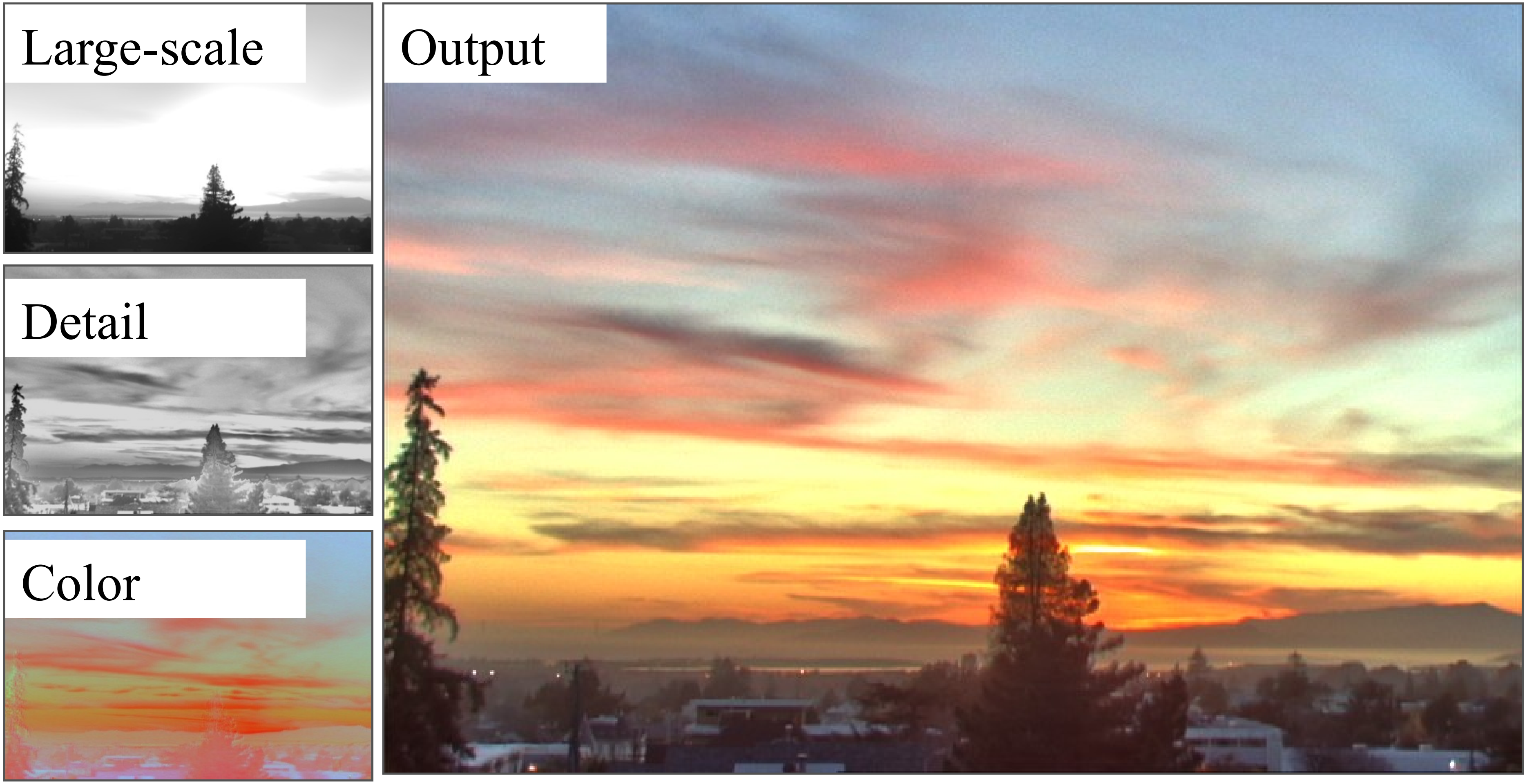




Bilateral filtering to the rescue

- Large scale = bilateral (log intensity)
- Detail = residual

[Durand & Dorsey 2002]



Hard to Compute

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

- Nonlinear
- Complex, spatially varying kernels
  - Cannot be pre-computed

- Brute-force implementation is slow > 10min

Brute-force Implementation

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

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

- Bilateral filtering

Input

Gaussian smoothing

Bilateral filtering