Bilateral Filters

Digital Visual Effects

Yung-Yu Chuang

with slides by Fredo Durand, Ramesh Raskar, Sylvain Paris, Soonmin Bae

Image Denoising

noisy image

naïve denoising
Gaussian blur

better denoising
edge-preserving filter

Smoothing an image without blurring its edges.

A Wide Range of Options

- Diffusion, Bayesian, Wavelets…
  - All have their pros and cons.

- Bilateral filter
  - not always the best result [Buades 05] but often good
  - easy to understand, adapt and set up

Basic denoising

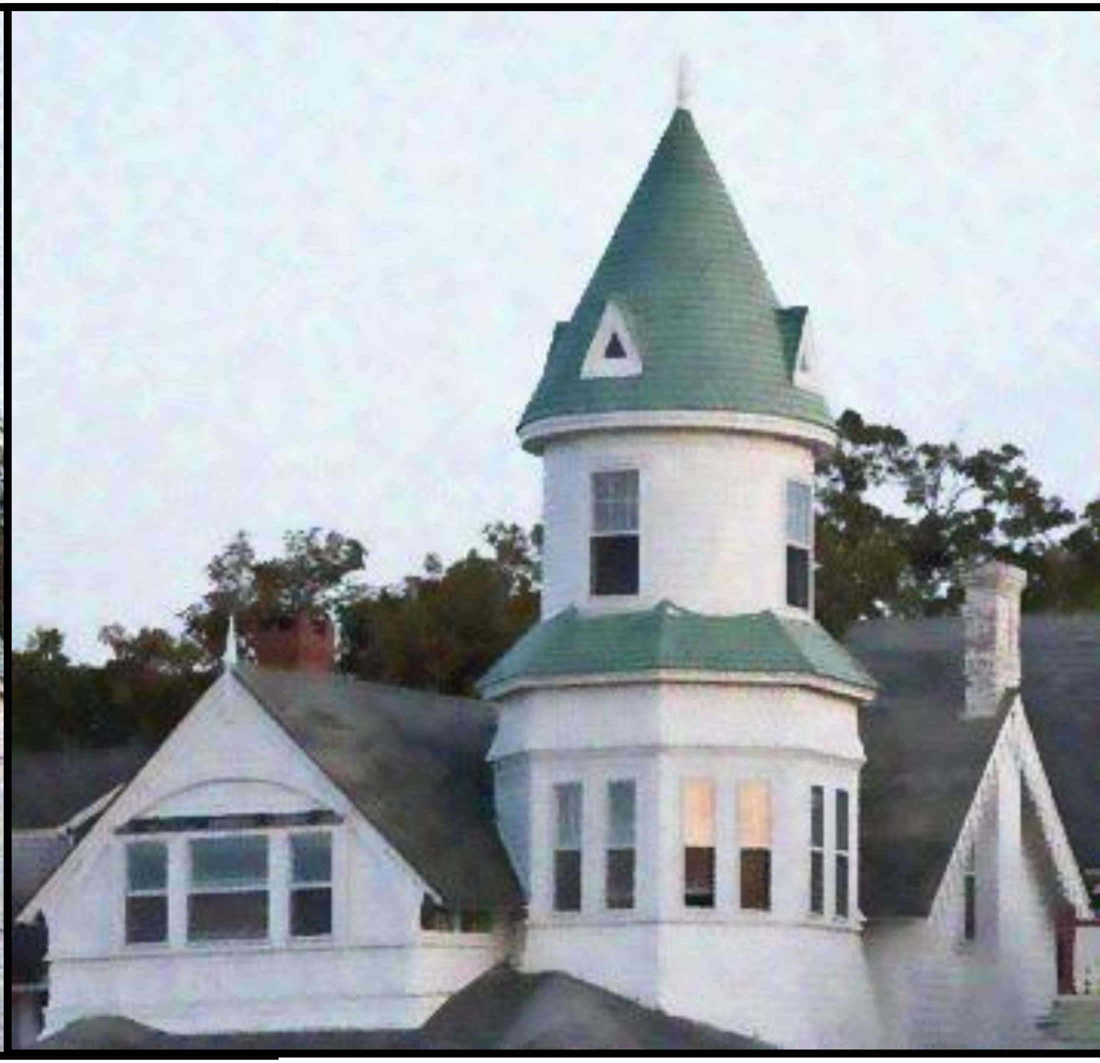
Noisy input

Median 5x5

Basic denoising

Noisy input

Bilateral filter 7x7 window



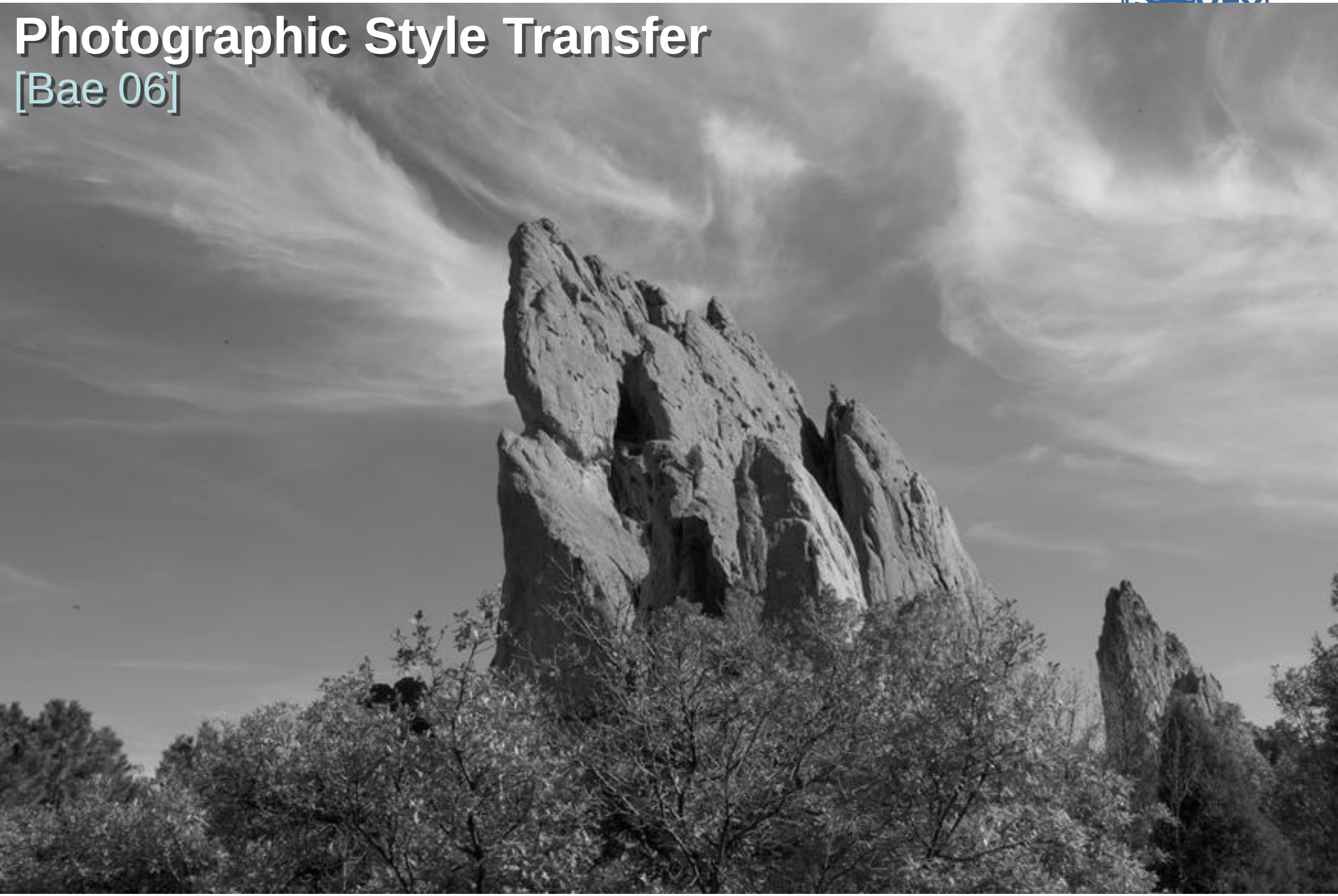
Tone Mapping
[Durand 02]

HDR input

Tone Mapping
[Durand 02]

output

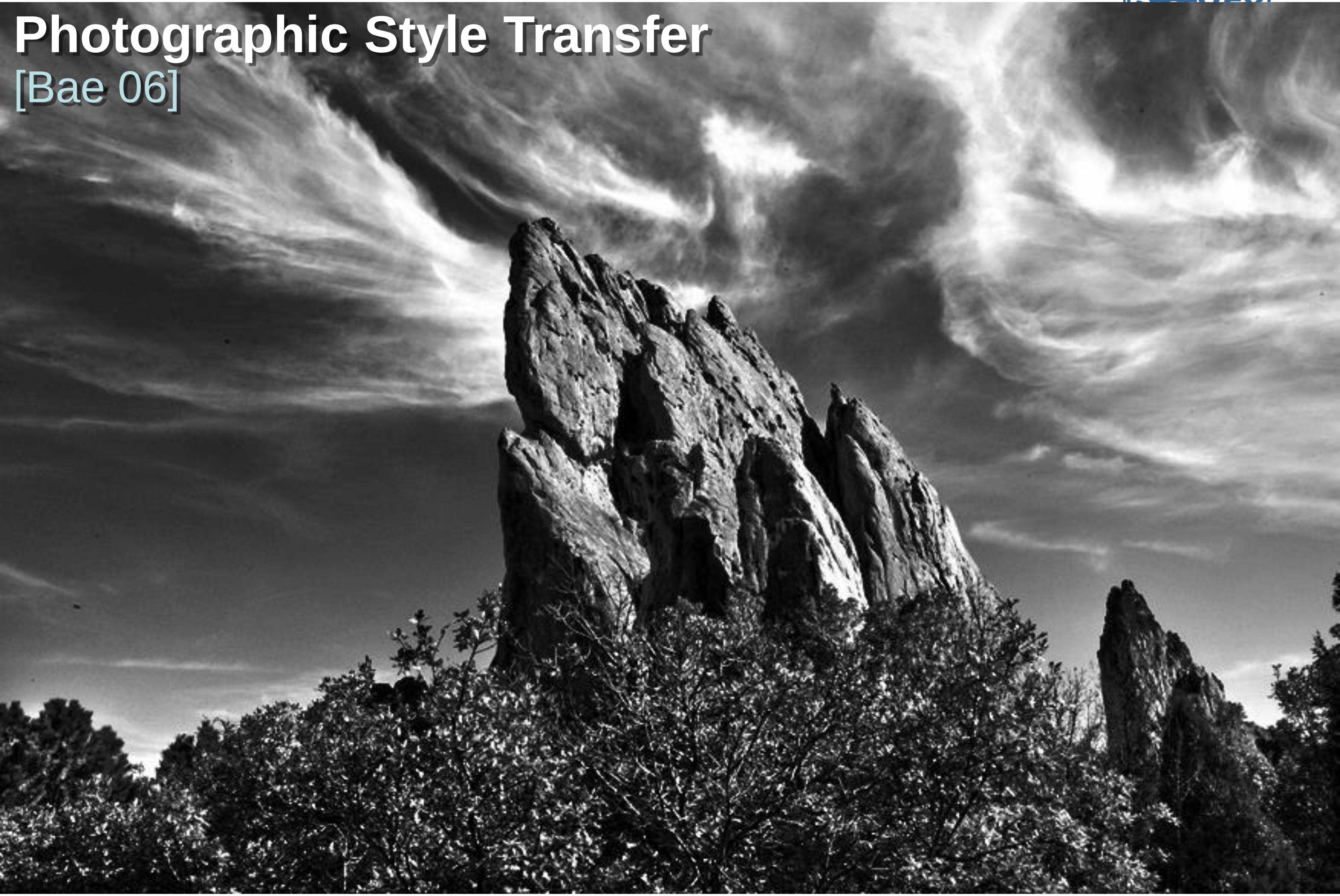
Photographic Style Transfer
[Bae 06]



input

Photographic Style Transfer

[Bae 06]



output

Cartoon Rendition

[Winnemöller 06]

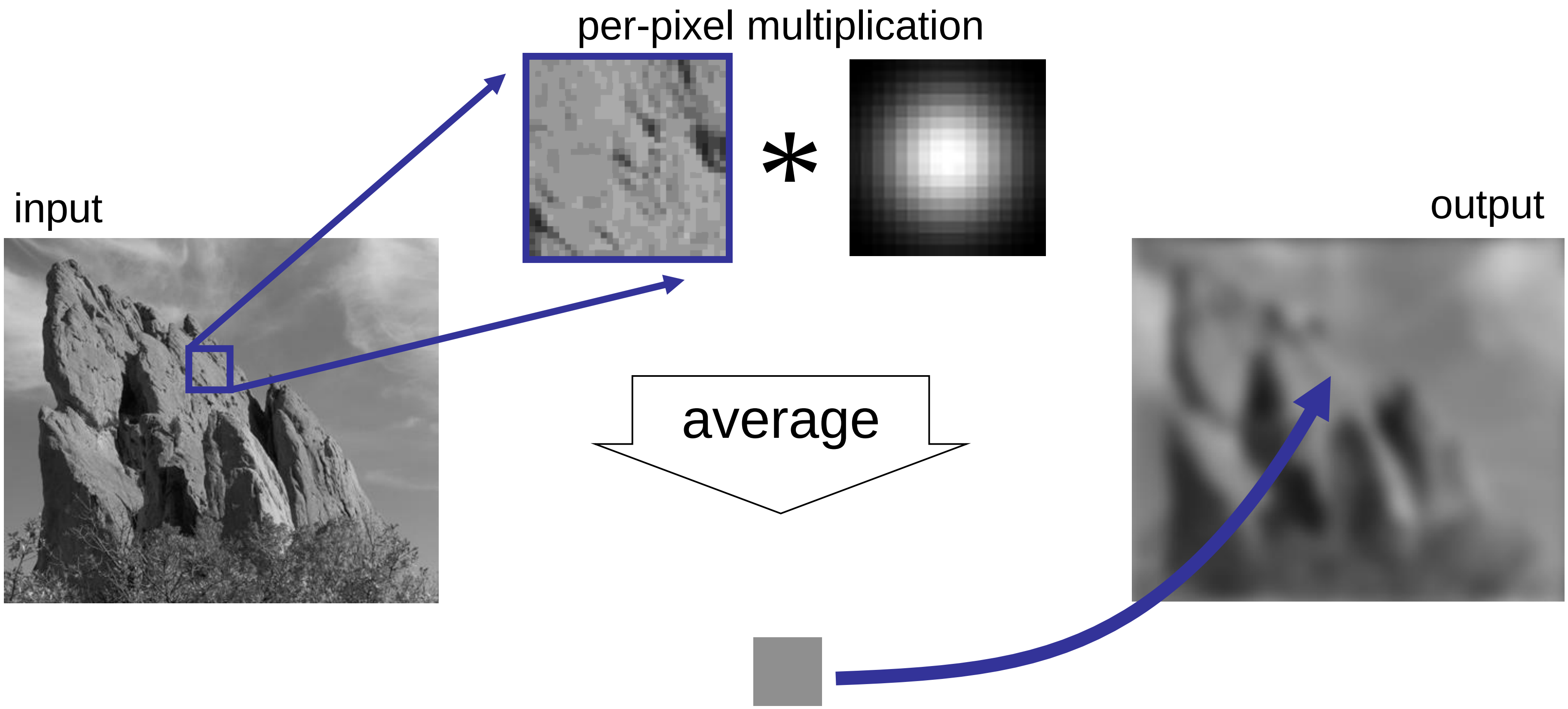
input

Cartoon Rendition

[Winnemöller 06]

output

Gaussian Blur




input


box average

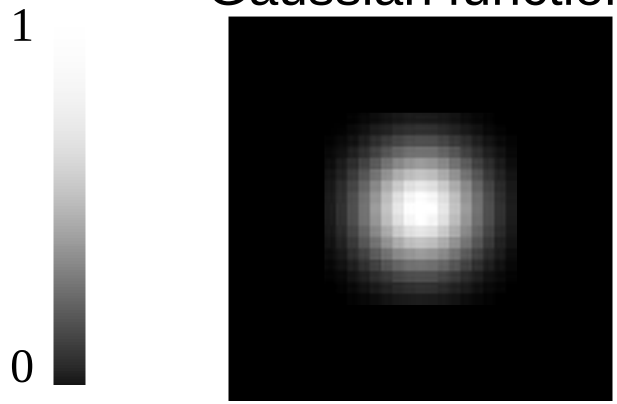
Gaussian blur

Equation of Gaussian Blur

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

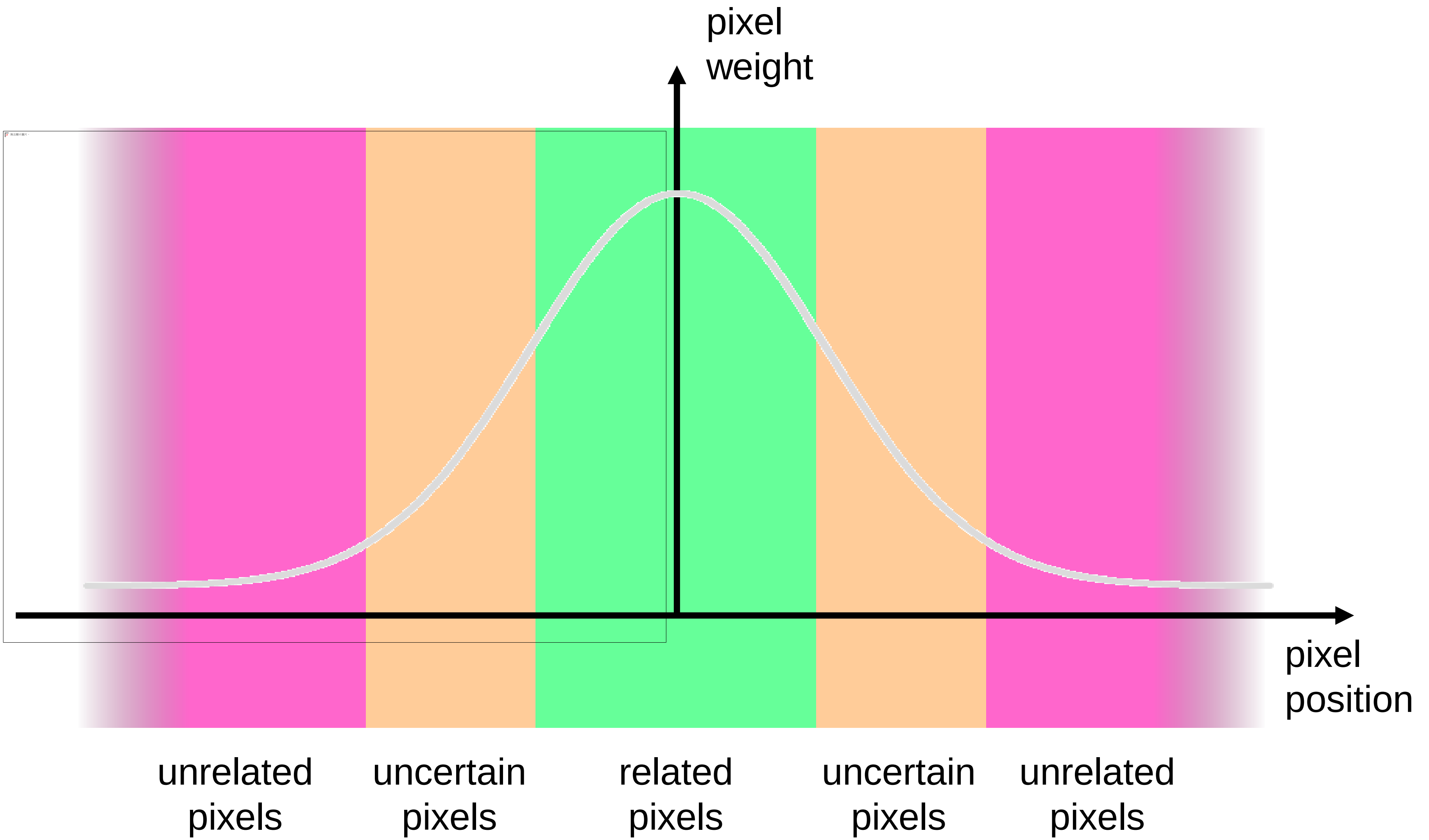
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|) I_{\mathbf{q}}$$

normalized Gaussian function



Gaussian Profile

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

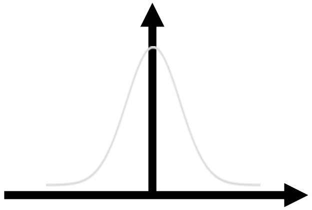


Spatial Parameter

input

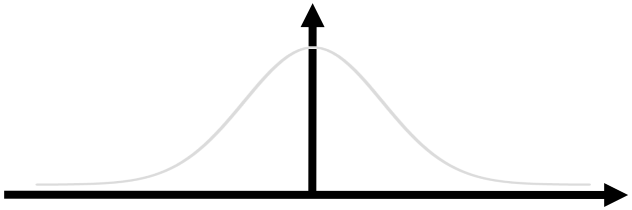
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\| \mathbf{p} - \mathbf{q} \|) \, I_{\mathbf{q}}$$

size of the window



small σ

limited smoothing



large σ

strong smoothing

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

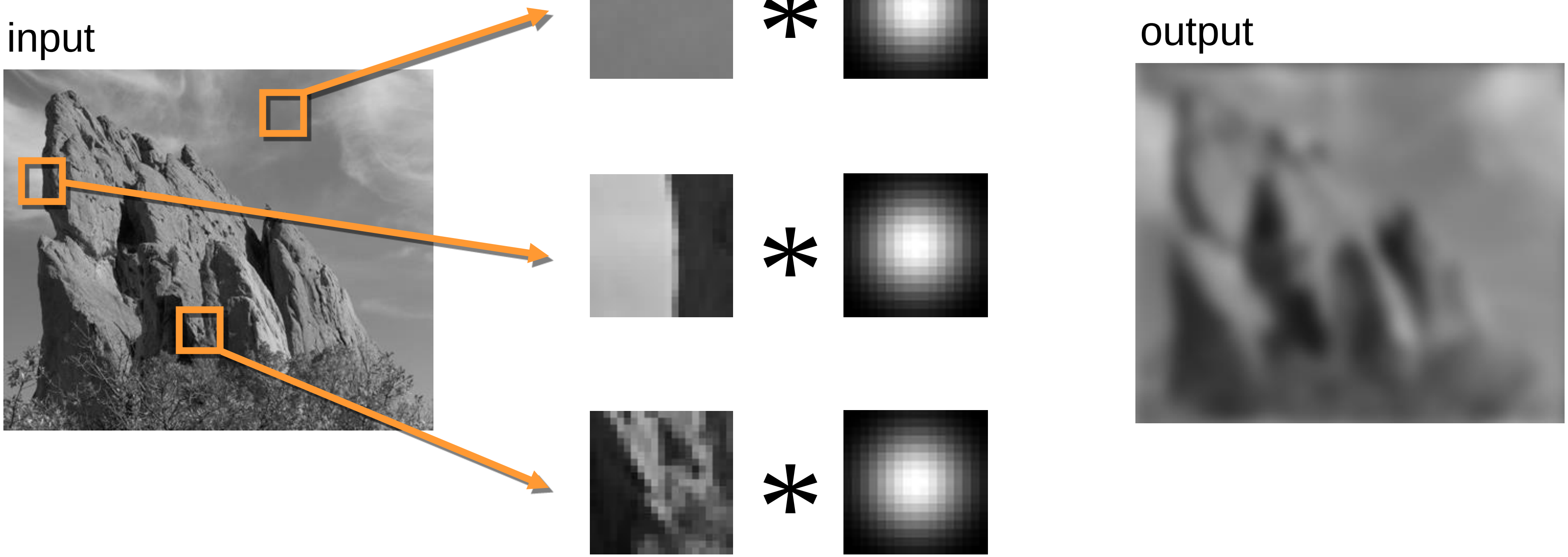
input

output

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

space

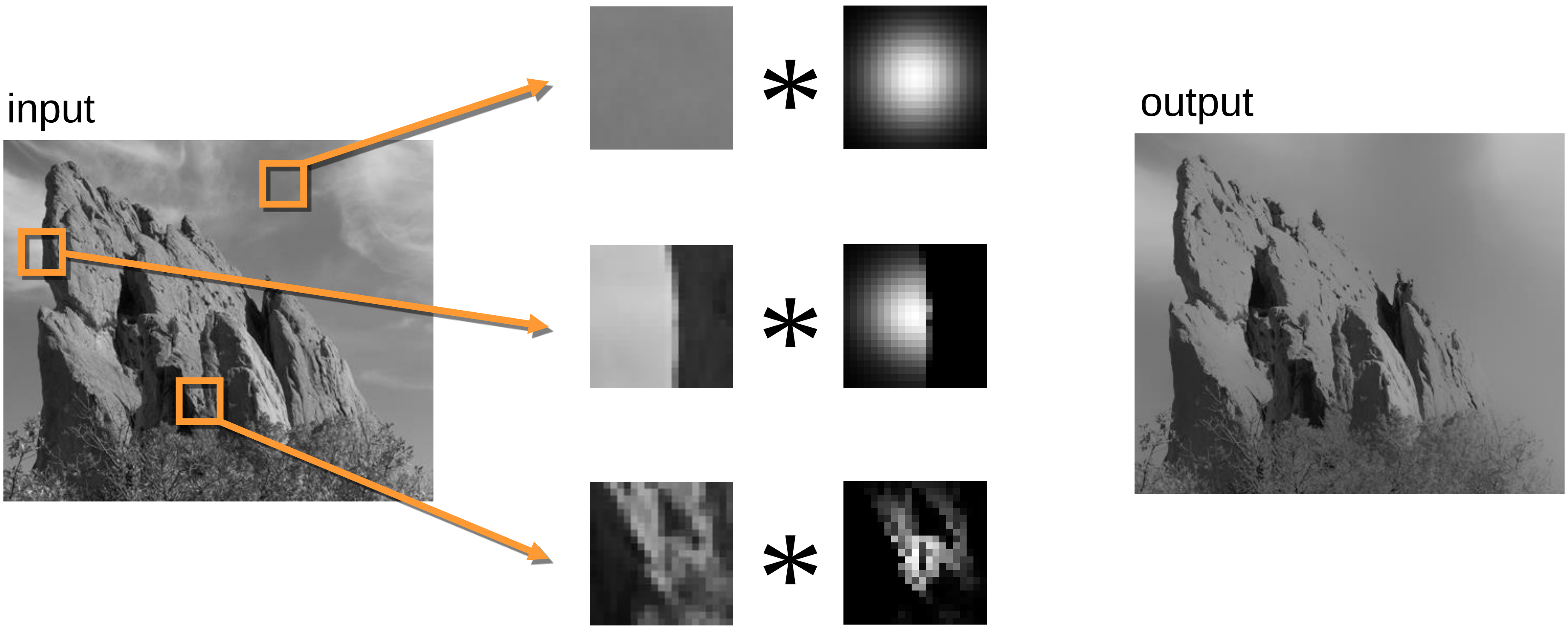
Blur Comes from Averaging across Edges



Same Gaussian kernel everywhere.

Bilateral Filter No Averaging across Edges

[Aurich 95, Smith 97, Tomasi 98]



The kernel shape depends on the image content.

Bilateral Filter Definition

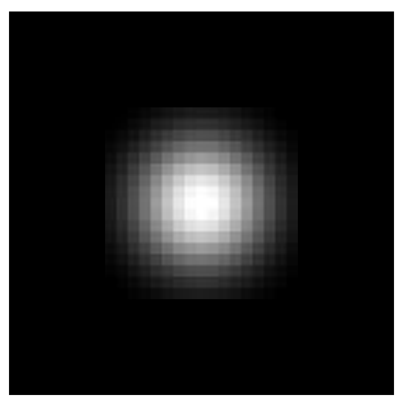
Same idea: **weighted average of pixels.**

new | not new | new

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

normalization factor

space weight



range weight

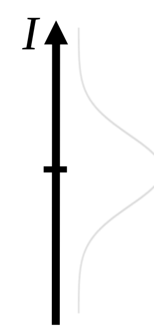
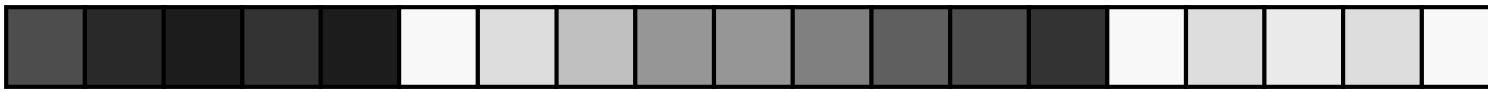
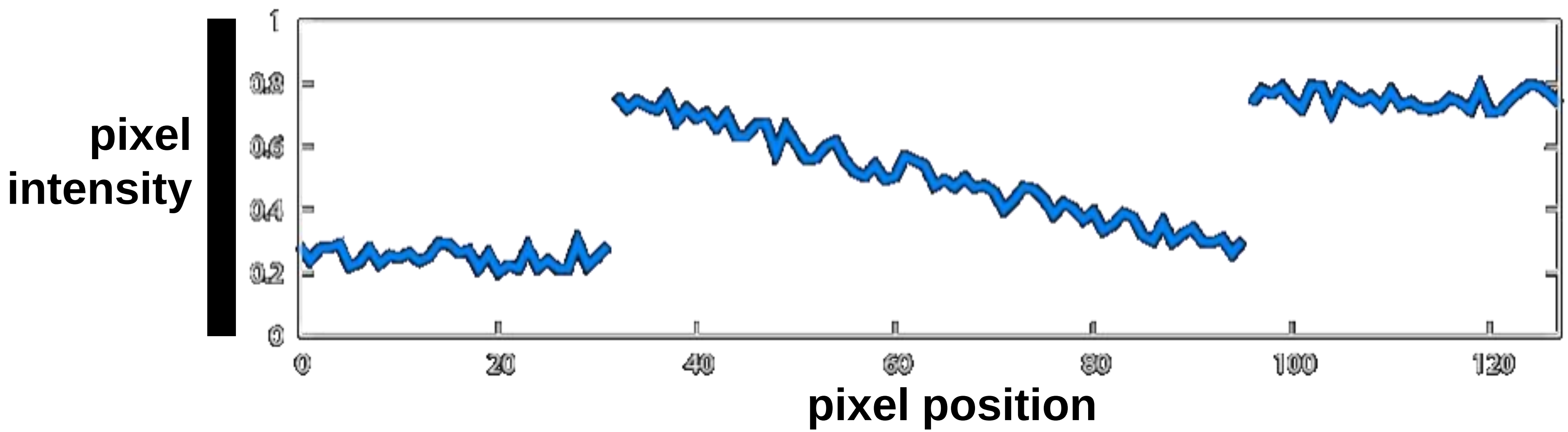


Illustration a 1D Image

- 1D image = line of pixels

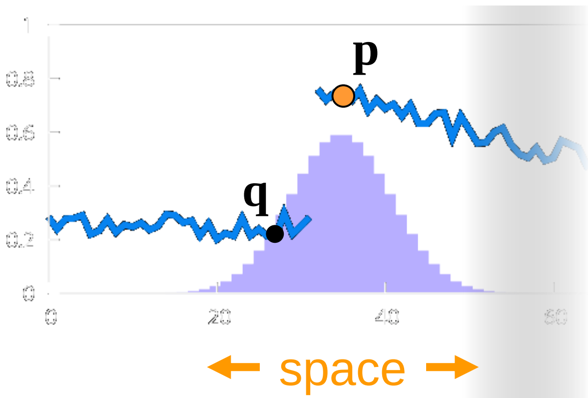


- Better visualized as a plot



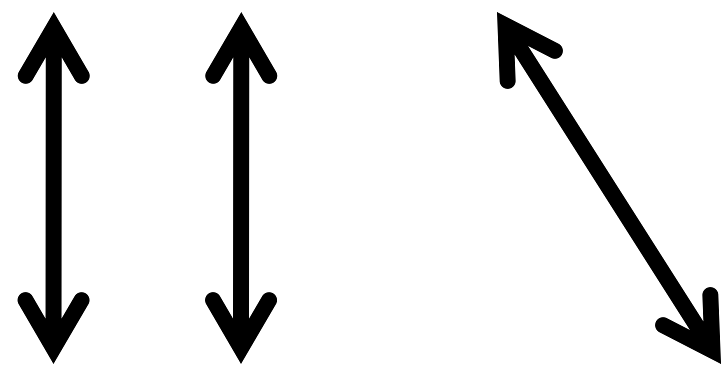
Gaussian Blur and Bilateral Filter

Gaussian blur



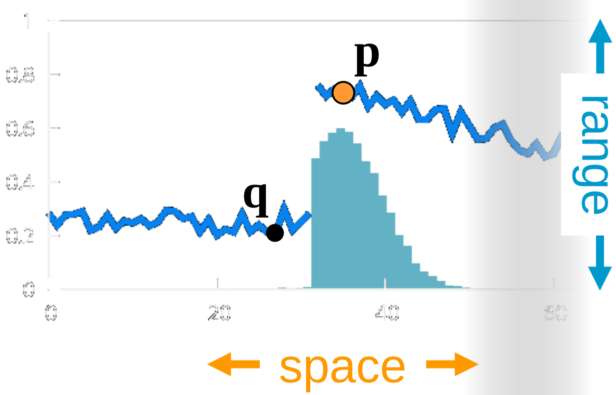
$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\|\mathbf{p}-\mathbf{q}\|)\, I_{\mathbf{q}}$$

space



Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



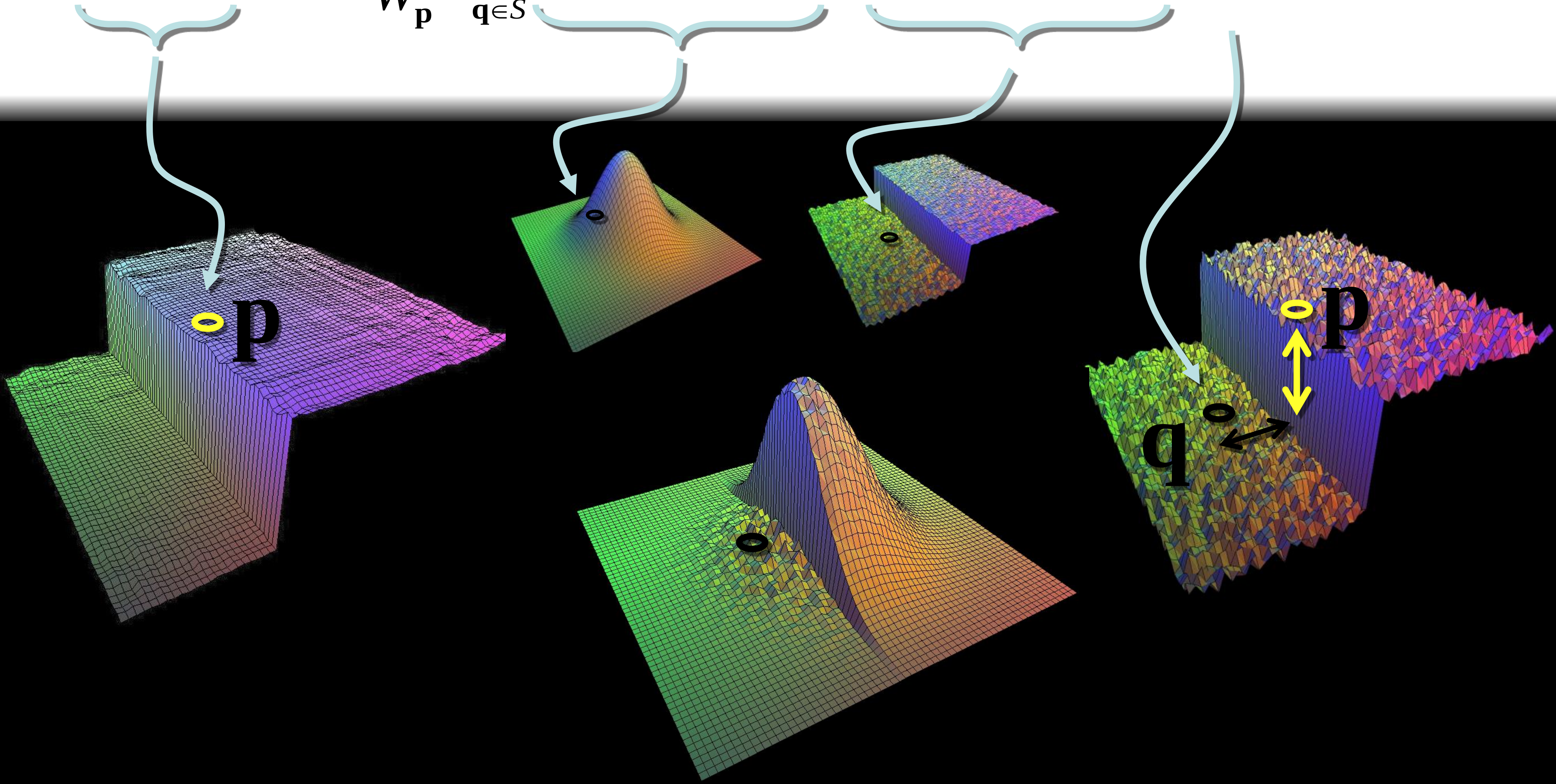
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

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]

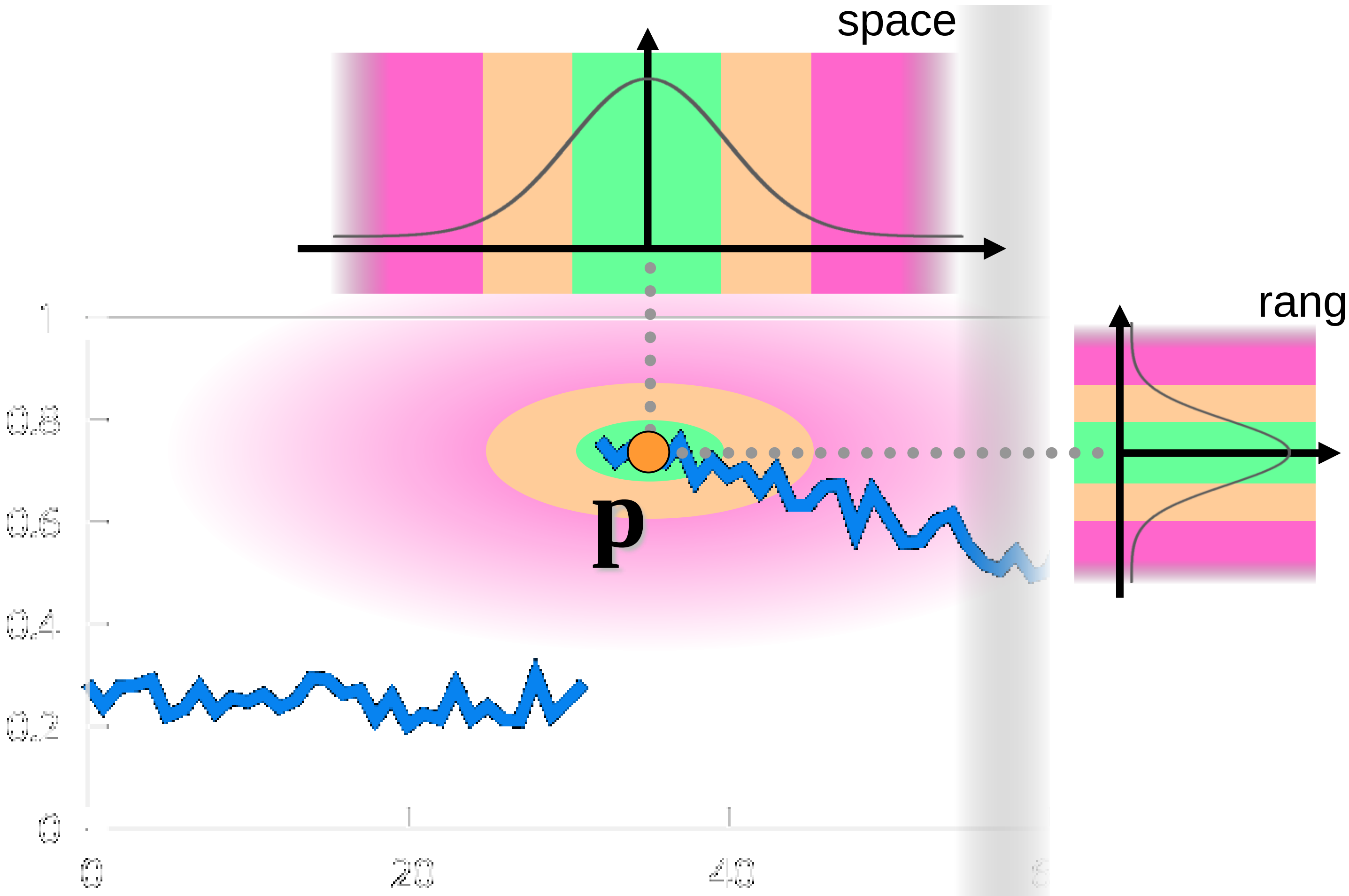
Space and Range Parameters

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

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

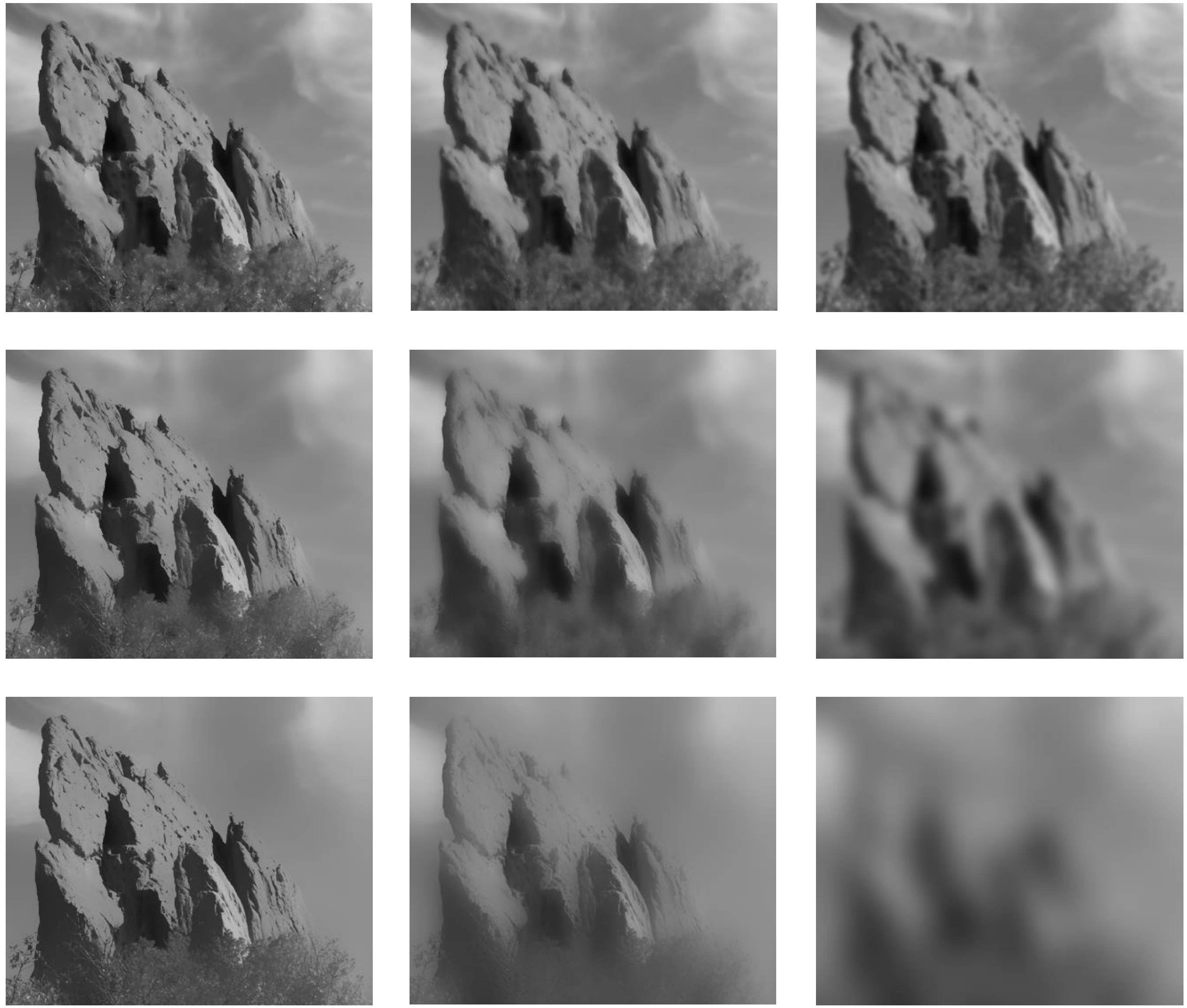
Influence of Pixels

Only pixels close in space and in range are considered.



Exploring the Parameter Space

input



Iterating the Bilateral Filter

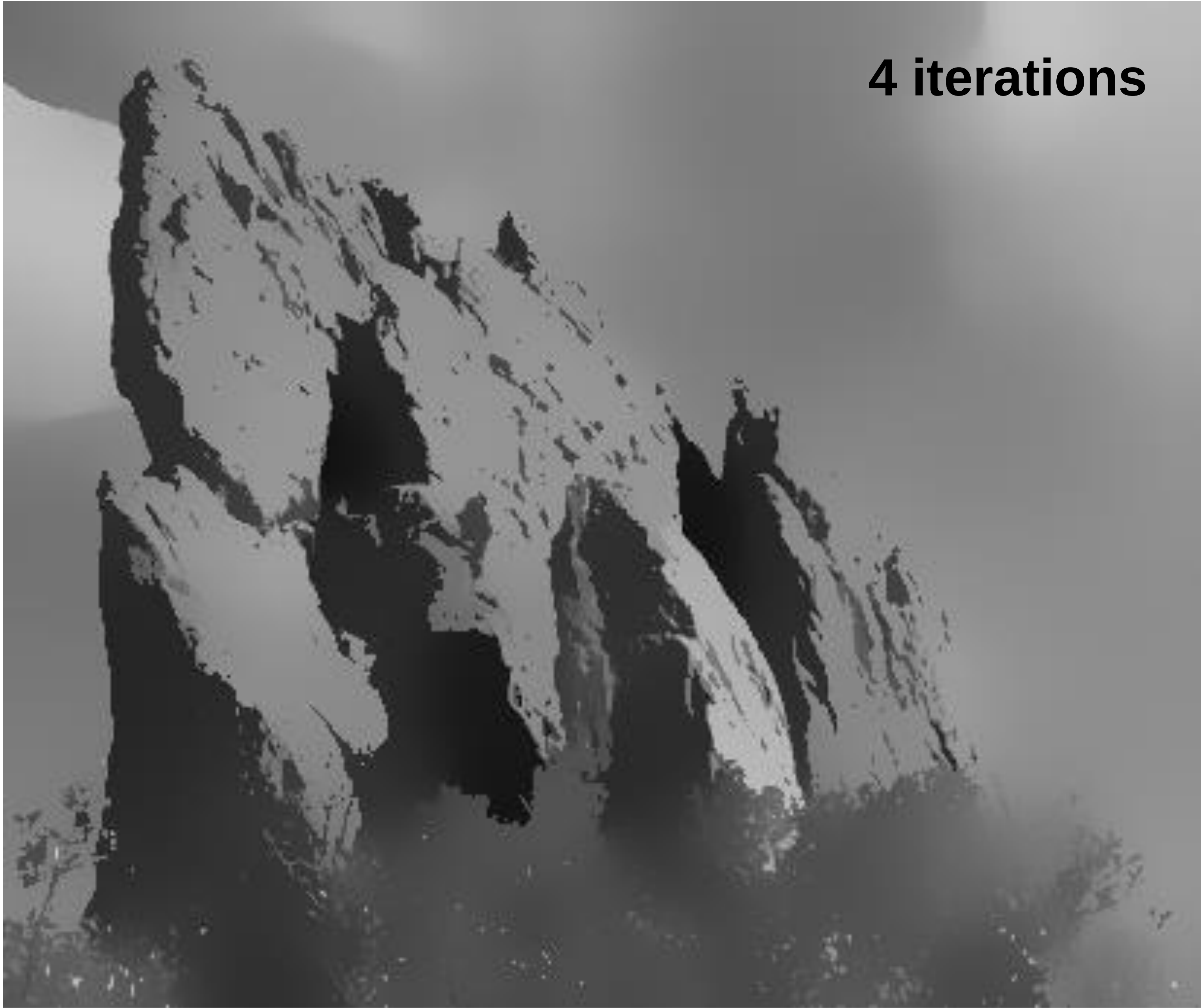
$$I_{(n+1)} = BF\left[I_{(n)}\right]$$

- Generate more piecewise-flat images
- Often not needed in computational photo, but could be useful for applications such as NPR.


input


1 iteration


2 iterations


4 iterations

Advantages of Bilateral Filter

- Easy to understand
  - Weighted mean of nearby pixels
- Easy to adapt
  - Distance between pixel values
- Easy to set up
  - Non-iterative

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

But Bilateral Filter is Nonlinear

- Slow but some accelerations exist:
  - [Elad 02]: Gauss-Seidel iterations
    - **Only for many iterations**
  - [Durand 02, Weiss 06]: fast approximation
    - **No formal understanding** of accuracy versus speed
    - [Weiss 06]: Only **box function** as spatial kernel

A Fast Approximation of the Bilateral Filter using a Signal Processing Approach

Sylvain Paris and Frédo Durand

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

DigiVFX

Definition of Bilateral Filter

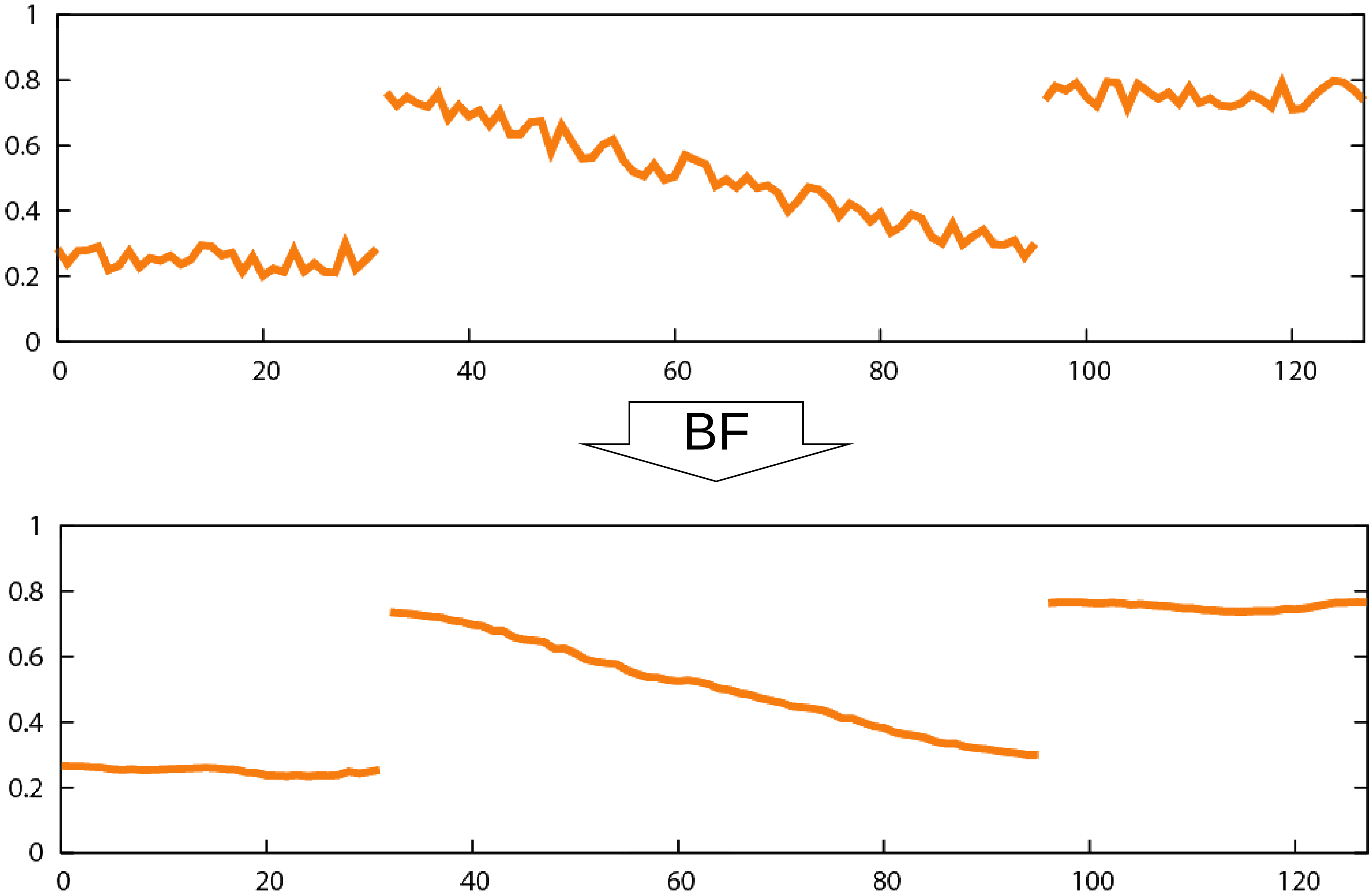
- [Smith 97, Tomasi 98]
- Smoothes an image and preserves edges
- Weighted average of neighbors
- Weights
  - Gaussian on *space* distance
  - Gaussian on *range* distance
  - sum to 1

$$I_{\mathbf{p}}^{\mathrm{bf}} = \frac{1}{W_{\mathbf{p}}^{\mathrm{bf}}} \sum_{\mathbf{q} \in \mathcal{S}} \underbrace{G_{\sigma_{\mathrm{s}}}(\|\mathbf{p} - \mathbf{q}\|)}_{\text{space}} \underbrace{G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)}_{\text{range}} I_{\mathbf{q}}$$

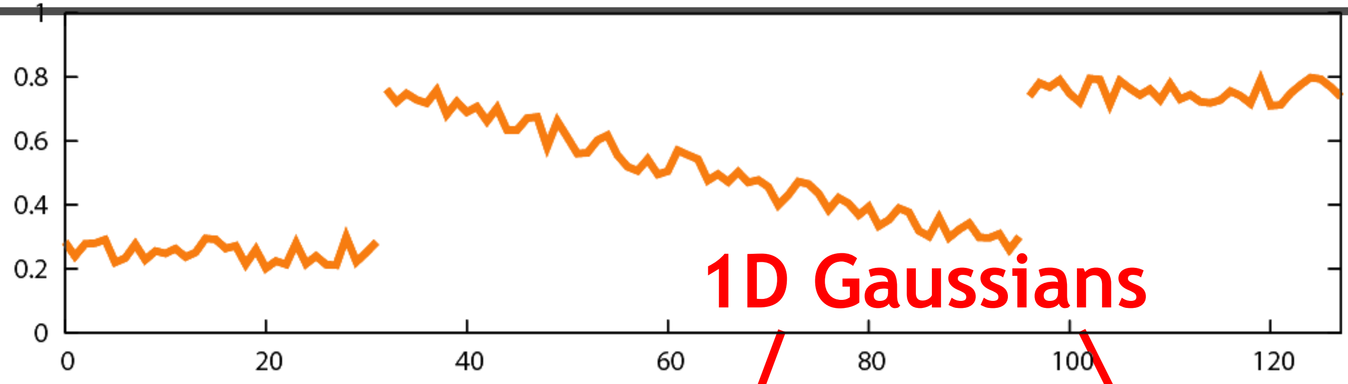
Contributions

- Link with **linear filtering**
- **Fast** and **accurate** approximation

Intuition on 1D Signal

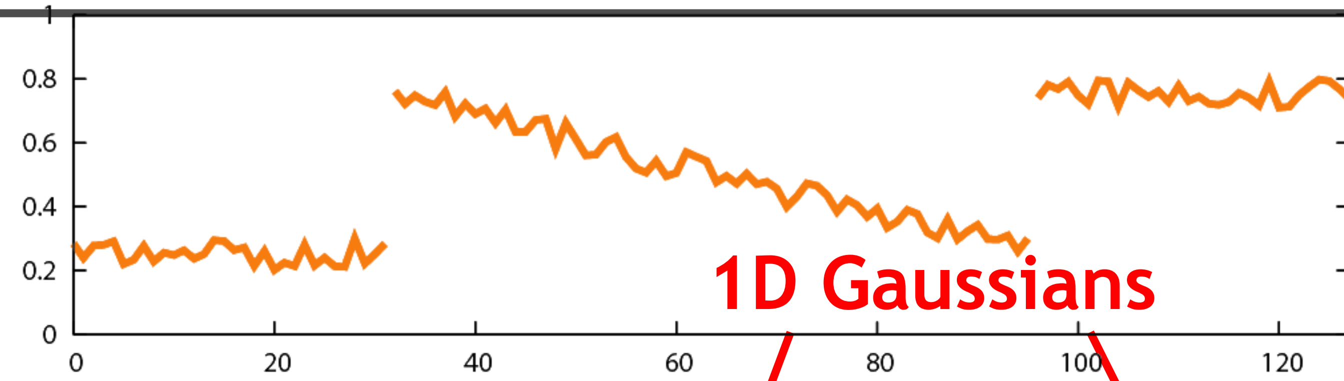


Basic idea



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

Basic idea



1D Gaussians

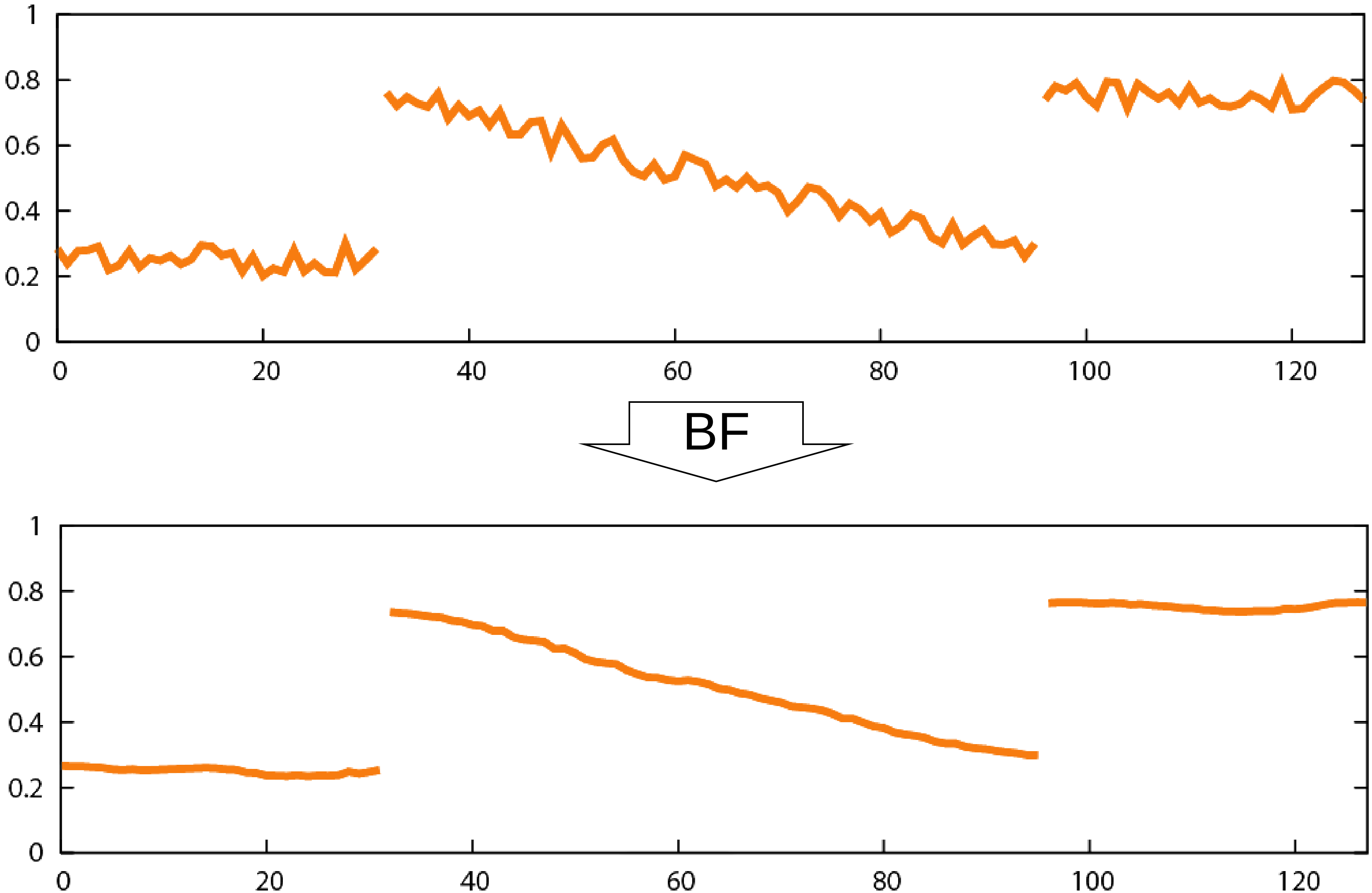
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G(\mathbf{q};\mathbf{p},\sigma_s)\, G(I_{\mathbf{q}}; I_{\mathbf{p}}, \sigma_r)\, I_{\mathbf{q}}$$

2D Gaussians

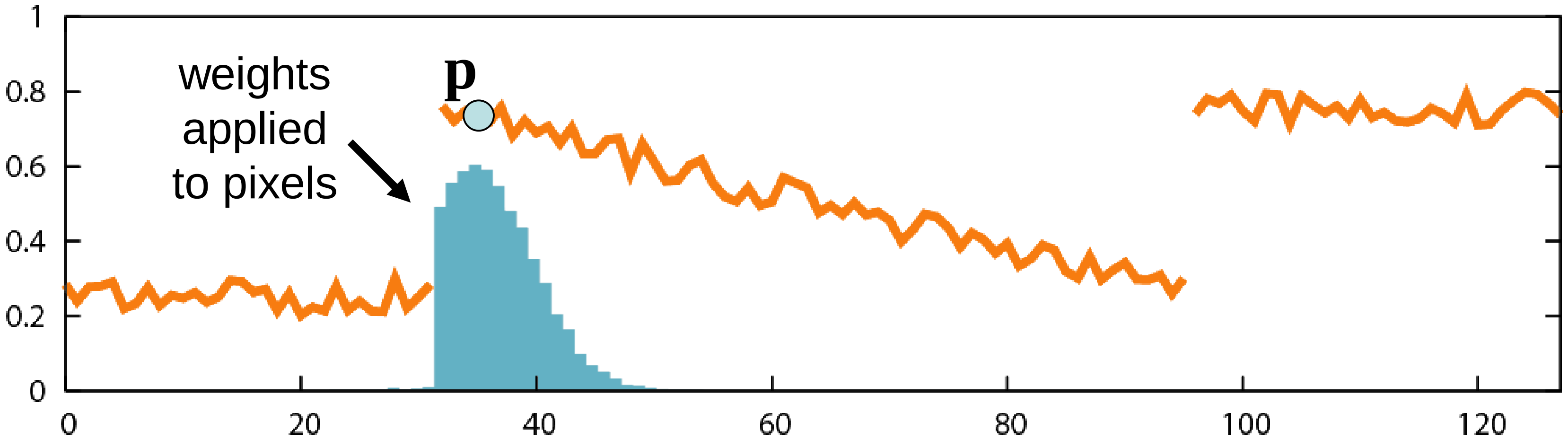
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{<\mathbf{q}, I'_{\mathbf{q}}> \in S'} G(\mathbf{q}, I_{\mathbf{q}}; \mathbf{p}, I_{\mathbf{p}}, \sigma_s, \sigma_r)\, I_{<\mathbf{q}, I'_{\mathbf{q}}>}$$

a special 2D image

Intuition on 1D Signal



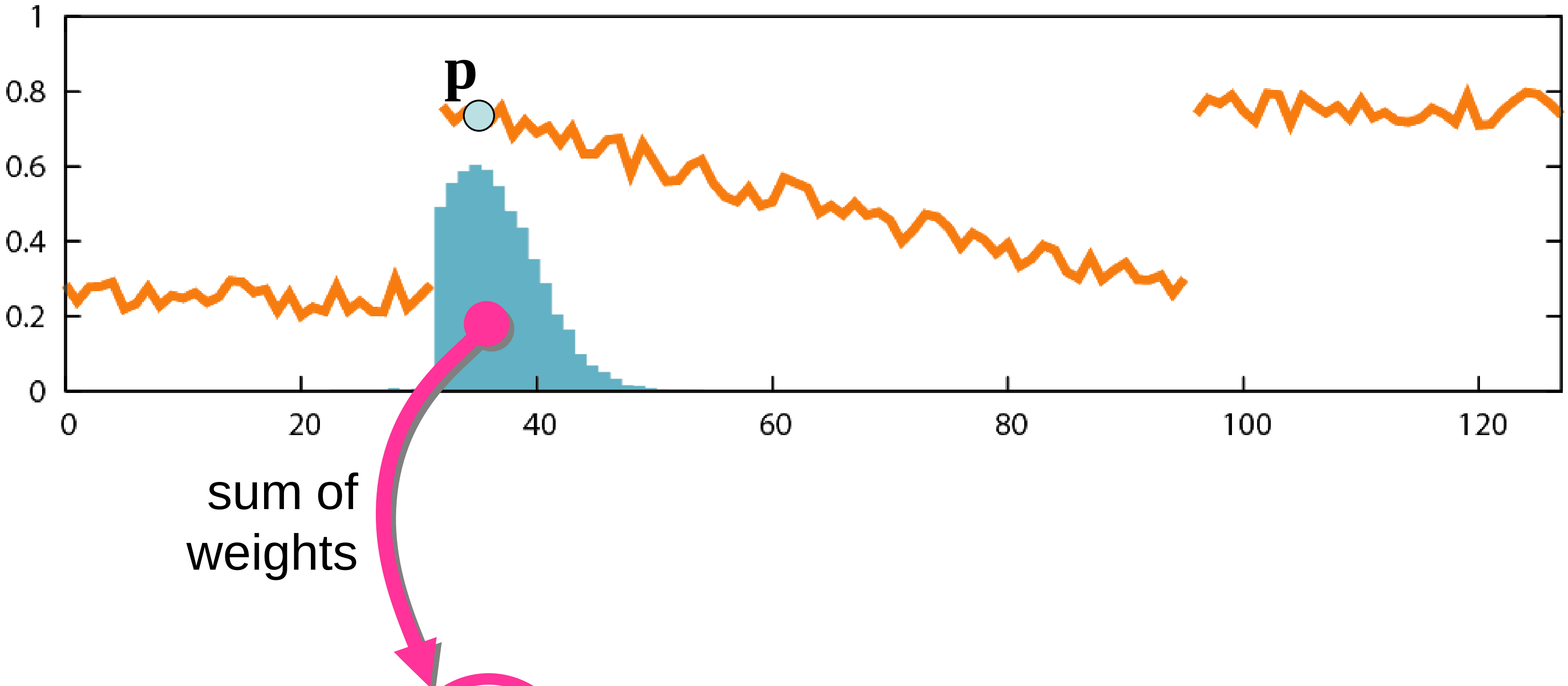
Intuition on 1D Signal
Weighted Average of Neighbors



- Near and similar pixels have influence.
- Far pixels have no influence.
- Pixels with different value have no influence.

Link with Linear Filtering

1. Handling the Division



$$I_{\mathbf{p}}^{\mathrm{bf}} = \frac{1}{W_{\mathbf{p}}^{\mathrm{bf}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

Handling the division with a **projective space**.

Formalization: Handling the Division

$$I_{\mathbf{p}}^{\mathrm{bf}} = \frac{1}{W_{\mathbf{p}}^{\mathrm{bf}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\; G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\; I_{\mathbf{q}}$$

$$W_{\mathbf{p}}^{\mathrm{bf}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\; G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)$$

- Normalizing factor as homogeneous coordinate
  - Multiply both sides by $W_{\mathbf{p}}^{\mathrm{bf}}$

$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}}\, I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\; G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

Formalization: Handling the Division

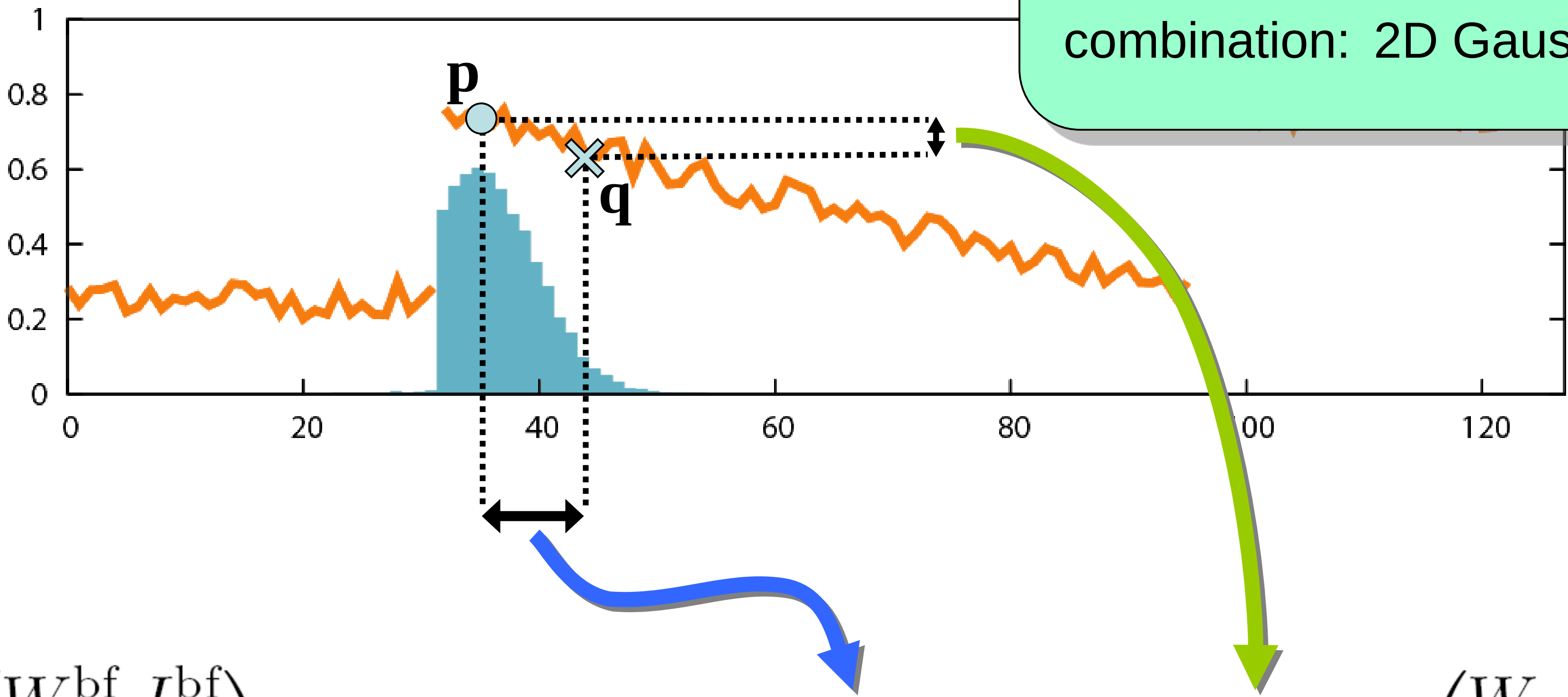
$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}}\, I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix} \quad \text{with } W_{\mathbf{q}}=1$$

- Similar to homogeneous coordinates in projective space
- Division delayed until the end
- Next step: Adding a dimension to make a convolution appear

Link with Linear Filtering
2. Introducing a Convolution

space: 1D Gaussian
× range: 1D Gaussian

combination: 2D Gaussian

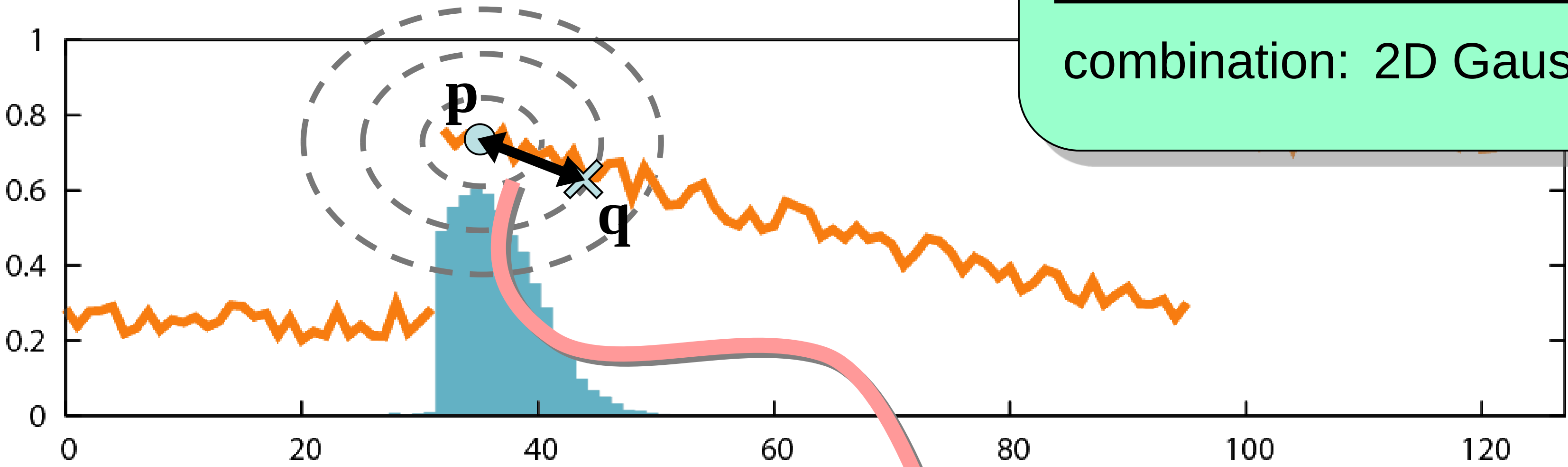


$$\begin{pmatrix} W^{\mathrm{bf}}_{\mathbf{p}} \, I^{\mathrm{bf}}_{\mathbf{p}} \\ W^{\mathrm{bf}}_{\mathbf{p}} \end{pmatrix} = \sum_{\mathbf{q} \in \mathcal{S}} \underbrace{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}_{\text{space}} \underbrace{G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)}_{\text{range}} \begin{pmatrix} W_{\mathbf{q}} \, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

Link with Linear Filtering
2. Introducing a Convolution

space: 1D Gaussian
× range: 1D Gaussian

combination: 2D Gaussian

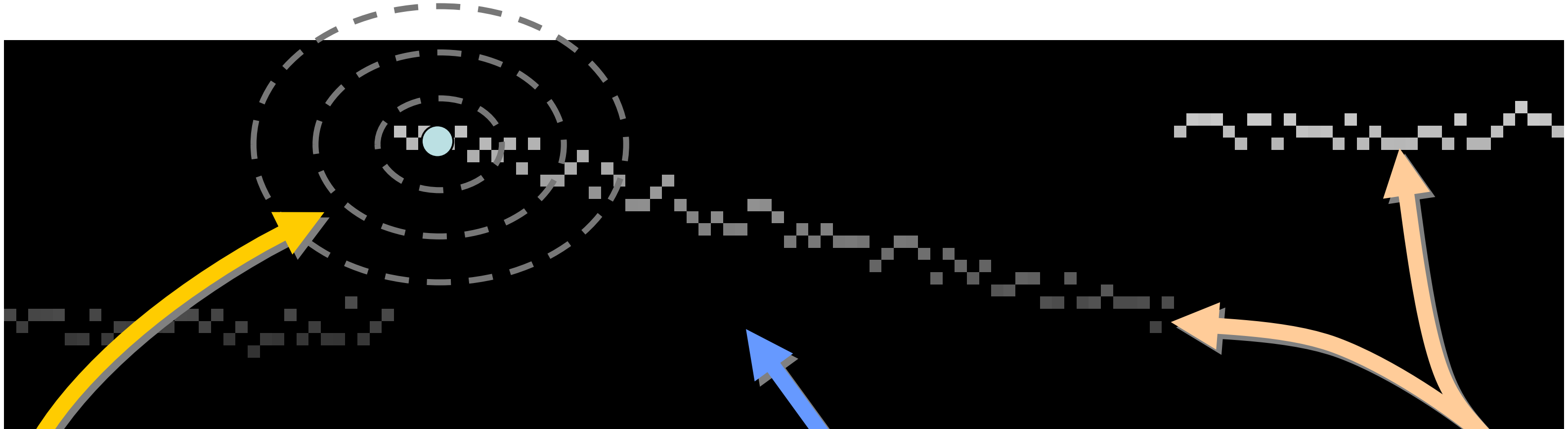


$$\begin{pmatrix} W^{\text{bf}}_{\mathbf{p}}\, I^{\text{bf}}_{\mathbf{p}} \\ W^{\text{bf}}_{\mathbf{p}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\text{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\text{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

space x range

Corresponds to a 3D Gaussian on a 2D image.

Link with Linear Filtering
2. Introducing a Convolution



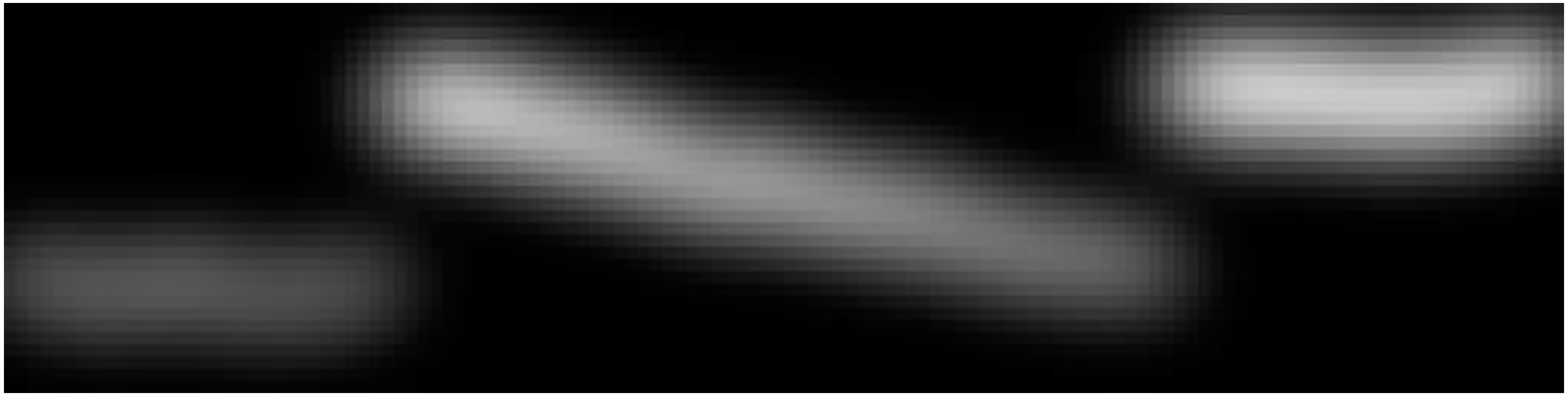
sum all values

black = zero

$$\begin{pmatrix} W^{\mathrm{bf}}_{\mathbf{p}}\, I^{\mathrm{bf}}_{\mathbf{p}} \\ W^{\mathrm{bf}}_{\mathbf{p}} \end{pmatrix} = \sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

sum all values multiplied by kernel ⇨ convolution

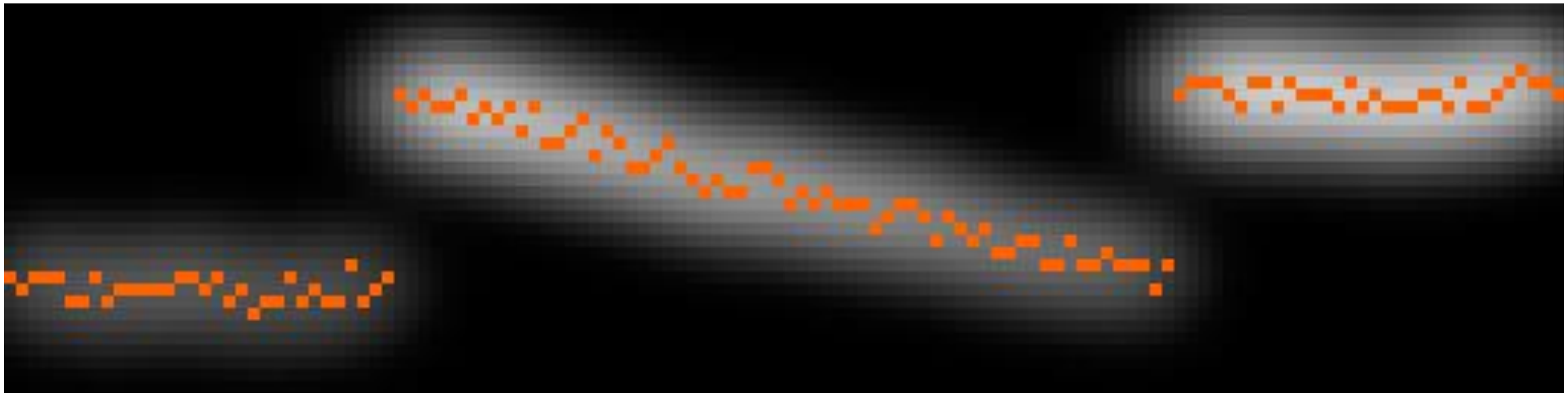
Link with Linear Filtering
2. Introducing a Convolution



result of the convolution

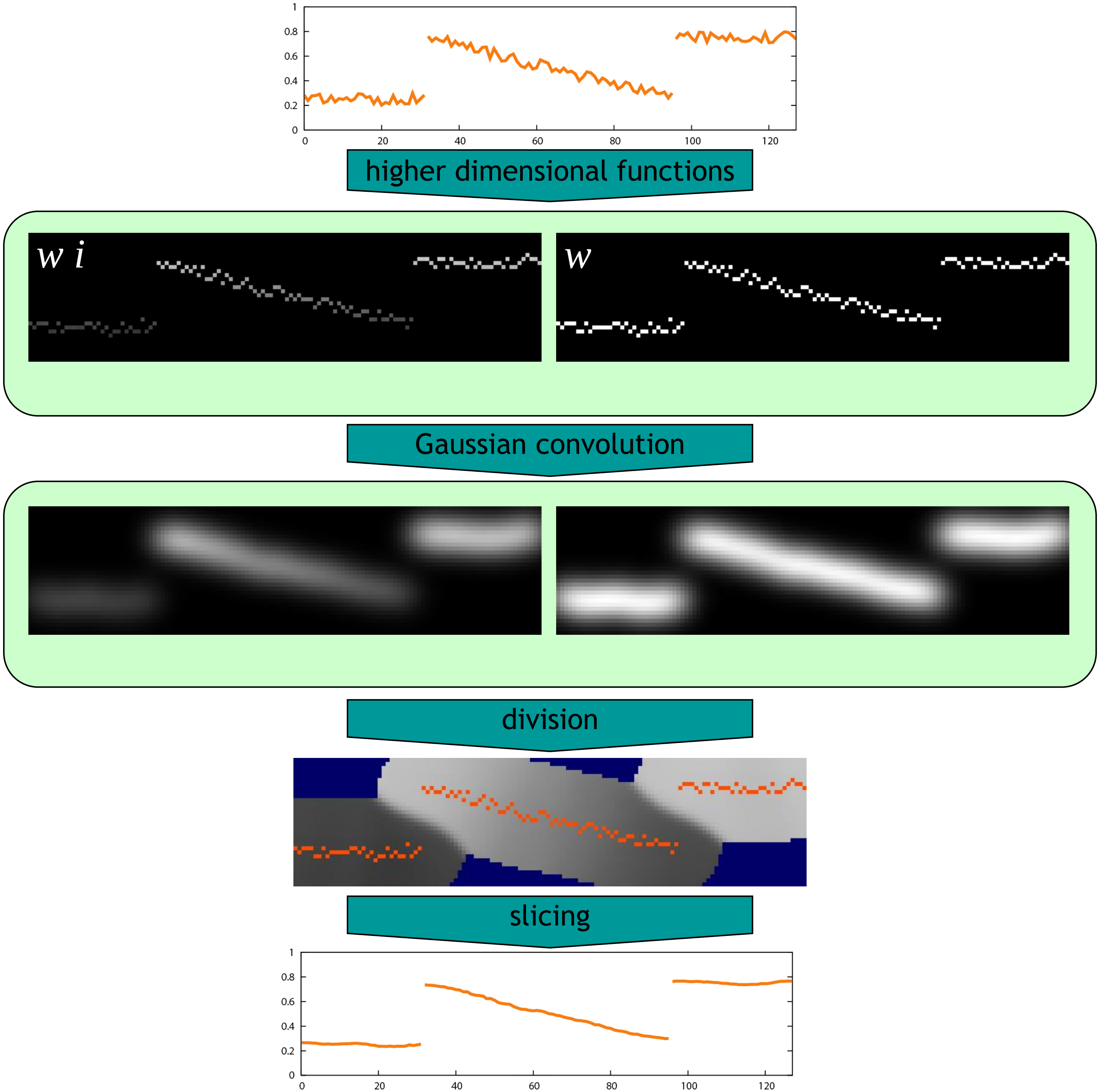
$$\begin{pmatrix} W^{\text{bf}}_{\mathbf{p}} I^{\text{bf}}_{\mathbf{p}} \\ W^{\text{bf}}_{\mathbf{p}} \end{pmatrix} = \sum_{(\mathbf{q},\zeta) \in \mathcal{S} \times \mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} W_{\mathbf{q}} I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$

Link with Linear Filtering
2. Introducing a Convolution



result of the convolution

$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}}\, I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} W_{\mathbf{q}}\, I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix}$$



higher dimensional functions
w i
w
Gaussian convolution
division
slicing

Reformulation: Summary

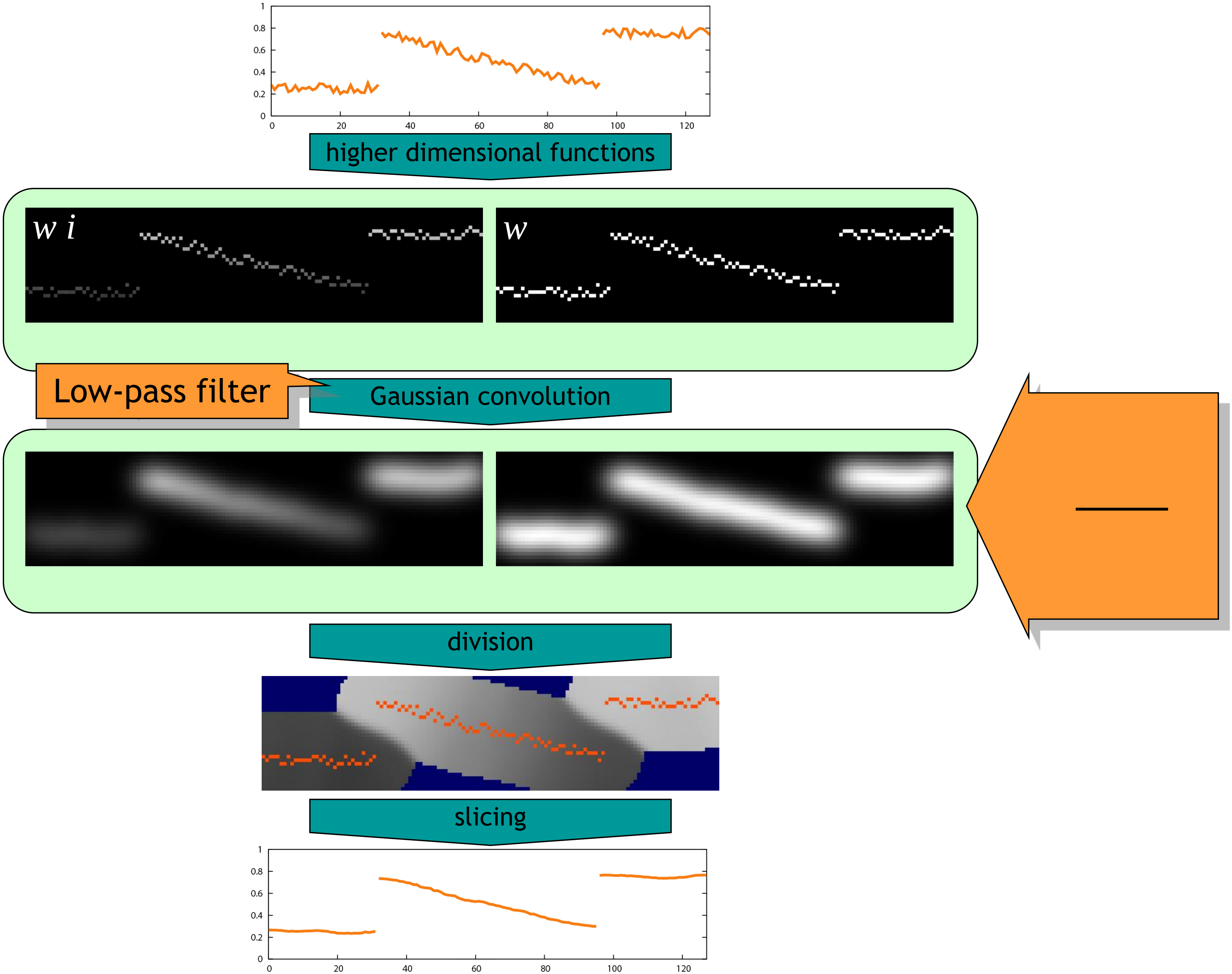
DigiVFX

linear: $$(w^{\mathrm{bf}}\, i^{\mathrm{bf}}, w^{\mathrm{bf}}) = g_{\sigma_{\mathrm{s}},\sigma_{\mathrm{r}}} \otimes (wi, w)$$

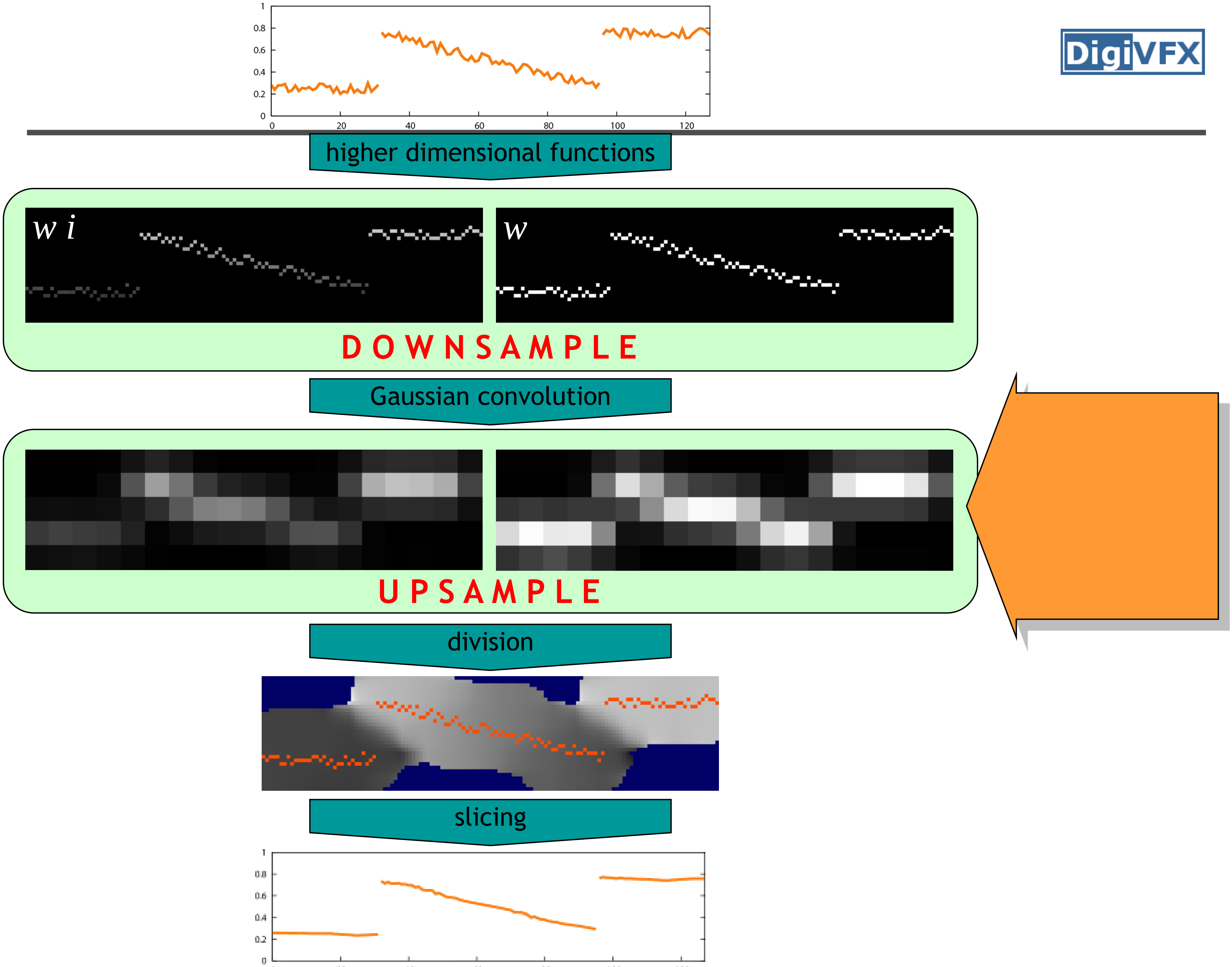
nonlinear: $$I^{\mathrm{bf}}_{\mathbf{p}} = \frac{w^{\mathrm{bf}}(\mathbf{p}, I_{\mathbf{p}})\; i^{\mathrm{bf}}(\mathbf{p}, I_{\mathbf{p}})}{w^{\mathrm{bf}}(\mathbf{p}, I_{\mathbf{p}})}$$

1. Convolution in higher dimension
   - expensive but well understood (linear, FFT, etc)
2. Division and slicing
   - nonlinear but simple and pixel-wise

Exact reformulation



higher dimensional functions
w i
w
Low-pass filter
Gaussian convolution
division
slicing

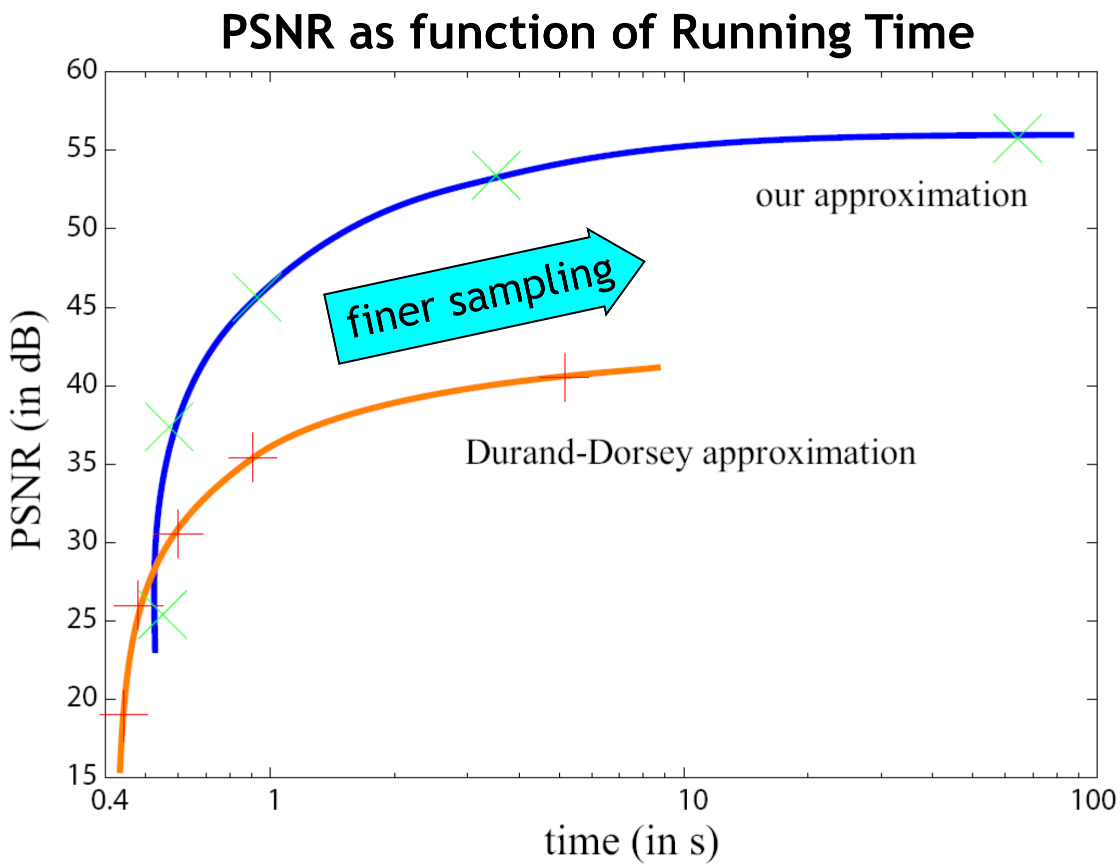

higher dimensional functions
w i
w
DOWNSAMPLE
Gaussian convolution
UPSAMPLE
division
slicing

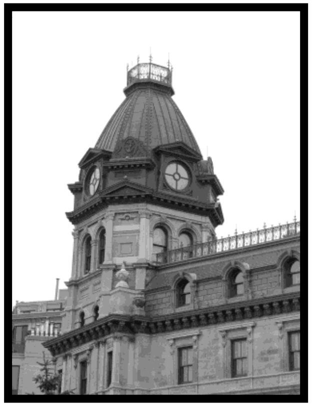
Fast Convolution by Downsampling

- Downsampling cuts frequencies above Nyquist limit
  - Less data to process
  - But induces error
- Evaluation of the approximation
  - Precision versus running time
  - Visual accuracy

Accuracy versus Running Time

- Finer sampling increases accuracy.
- More precise than previous work.





Digital photograph 1200 × 1600

Straightforward implementation is over 10 minutes.

Visual Results

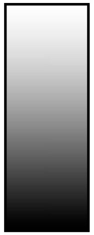
- Comparison with previous work [Durand 02]
  - running time = 1s for both techniques

1200 × 1600

input | exact BF | our result | prev. work

difference with exact computation (intensities in [0:1])

0.1



0

Two-scale Tone Management for Photographic Look

Soonmin Bae, Sylvain Paris, and Frédo Durand

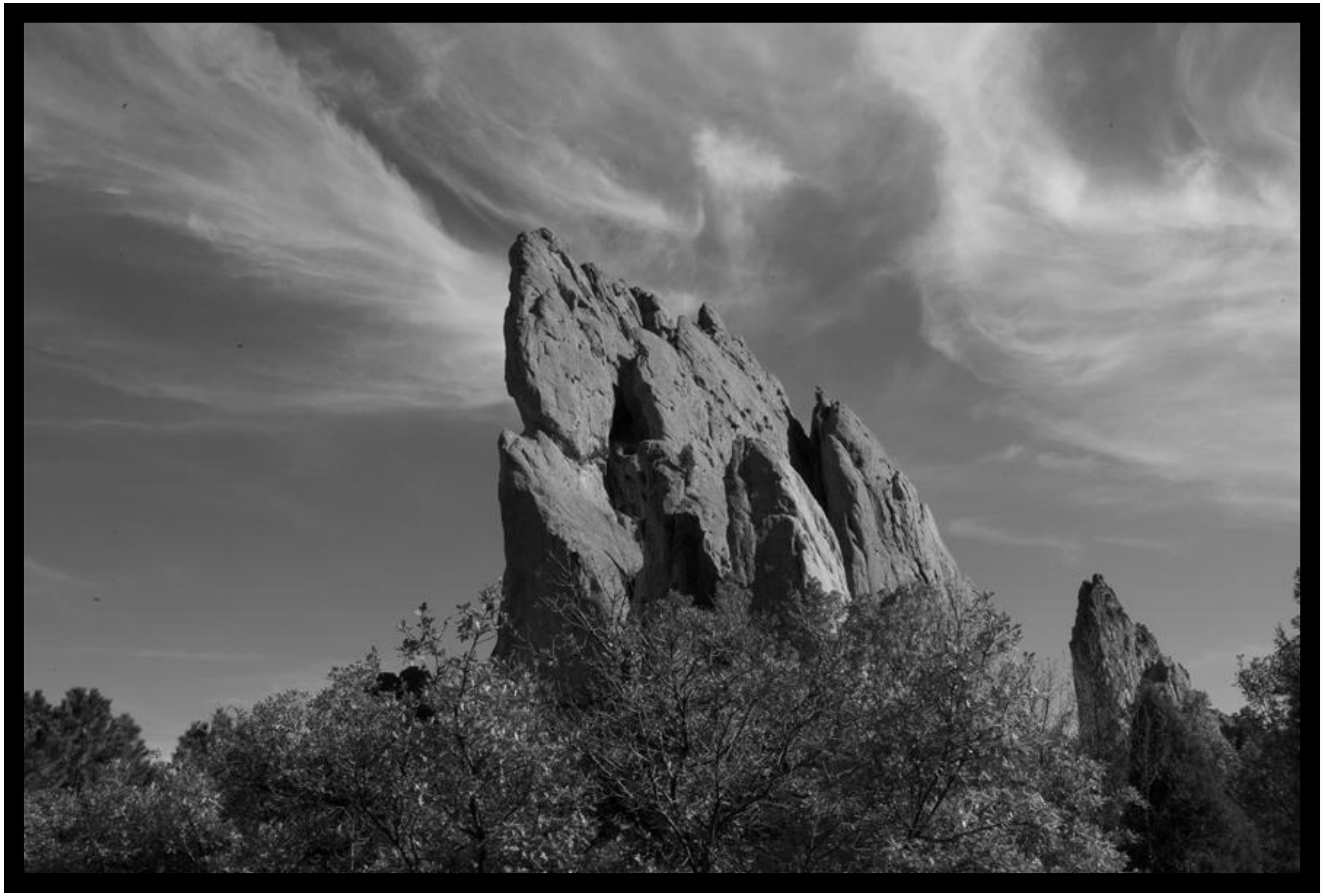
MIT CSAIL

SIGGRAPH2006

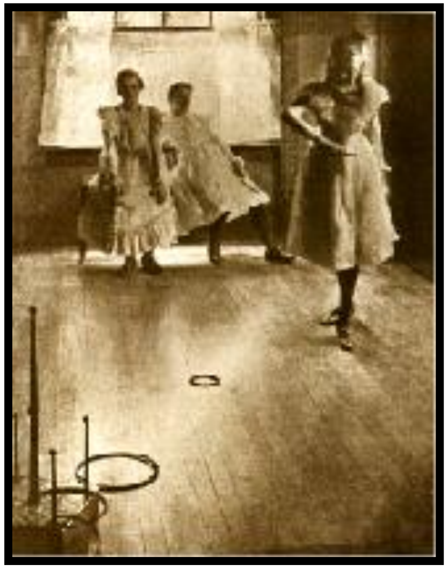
Ansel Adams

Ansel Adams, *Clearing Winter Storm*

An Amateur Photographer



A Variety of Looks



Goals

- Control over photographic look
- Transfer "look" from a model photo

For example,

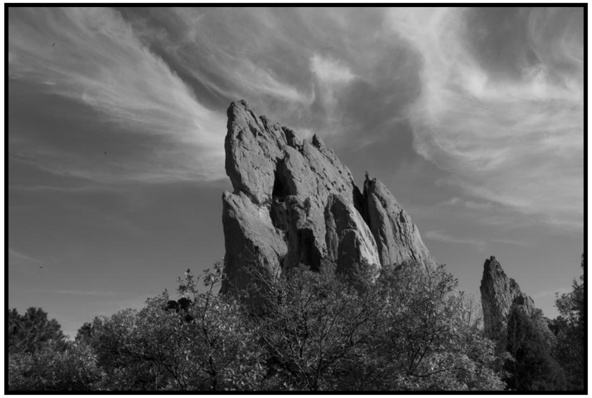
we want

with the look of

Aspects of Photographic Look

- Subject choice
- Framing and composition

➔ Specified by input photos



Input

- Tone distribution and contrast

➔Modified based on model photos

Model

Tonal Aspects of Look

Ansel Adams

Kenro Izu

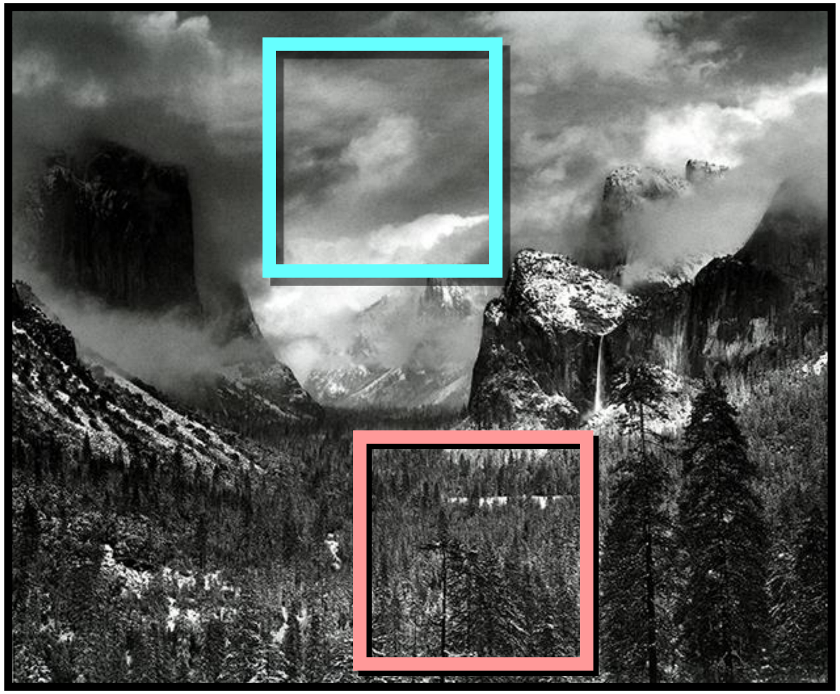
Tonal aspects of Look - Global Contrast

Ansel Adams

Kenro Izu

High Global Contrast **Low Global Contrast**

Tonal aspects of Look - Local Contrast



Ansel Adams

Kenro Izu

Variable amount of texture **Texture everywhere**

Overview

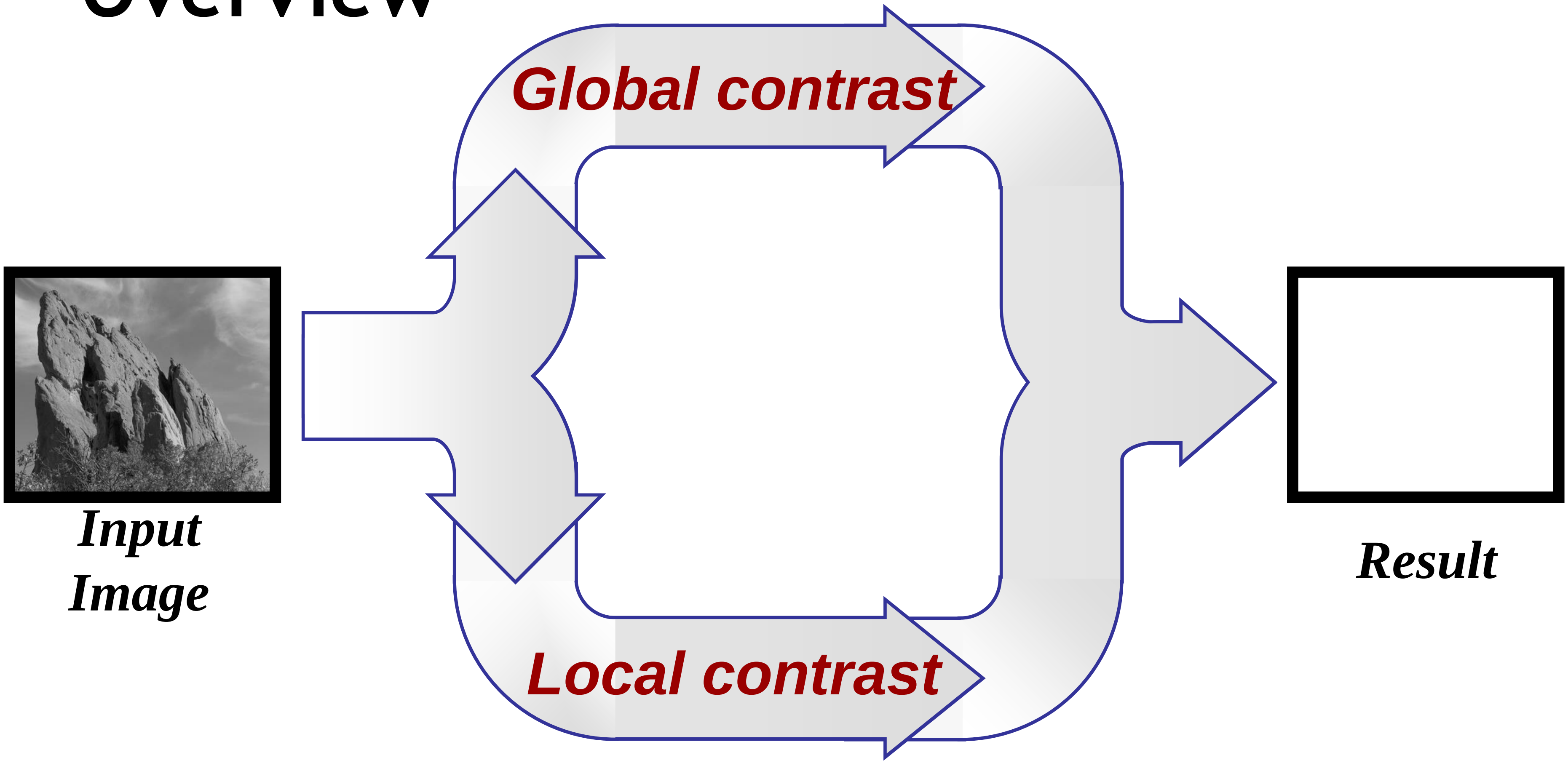
Model

Input Image

Result

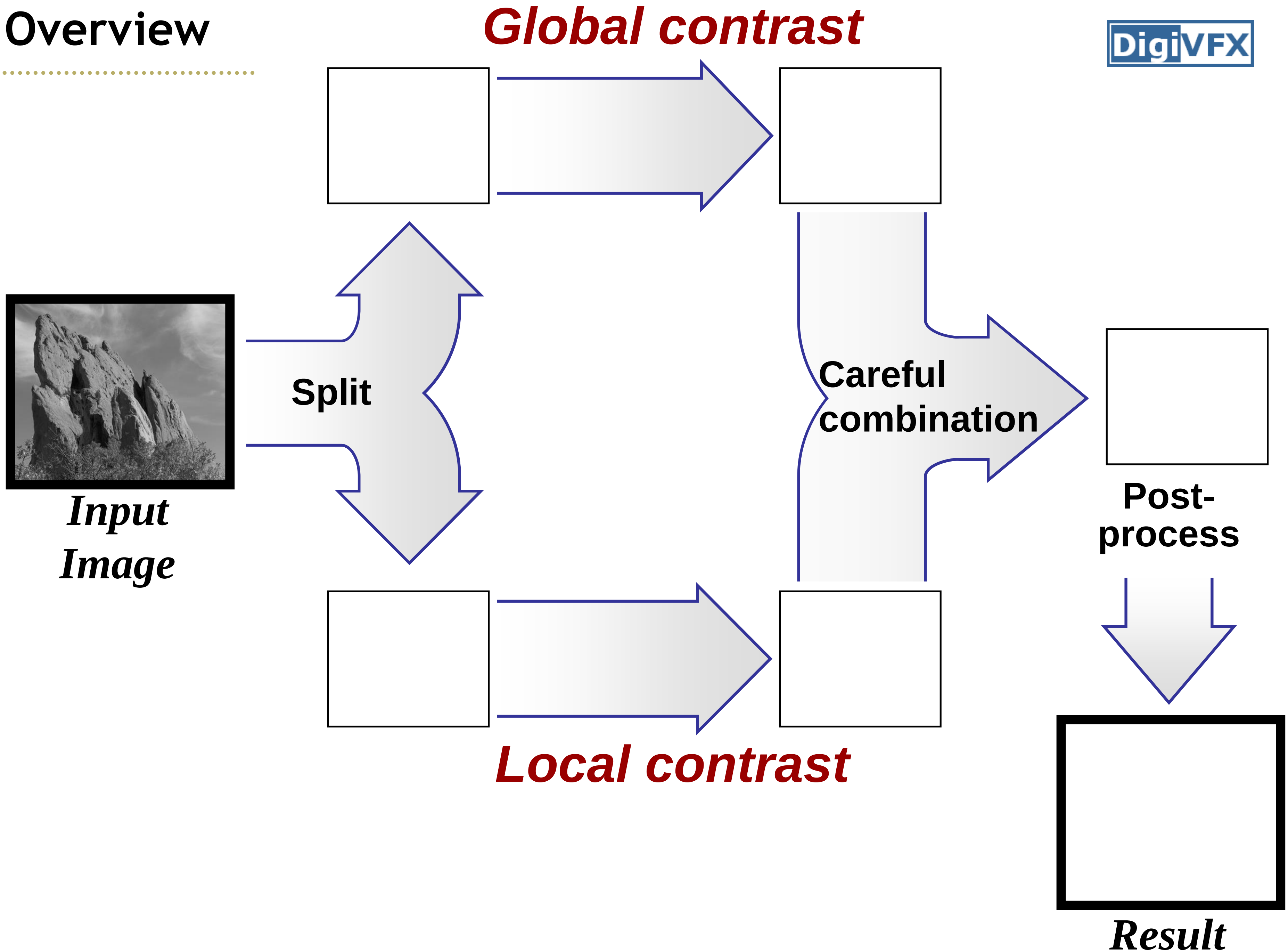
- Transfer look between photographs
  - Tonal aspects

Overview

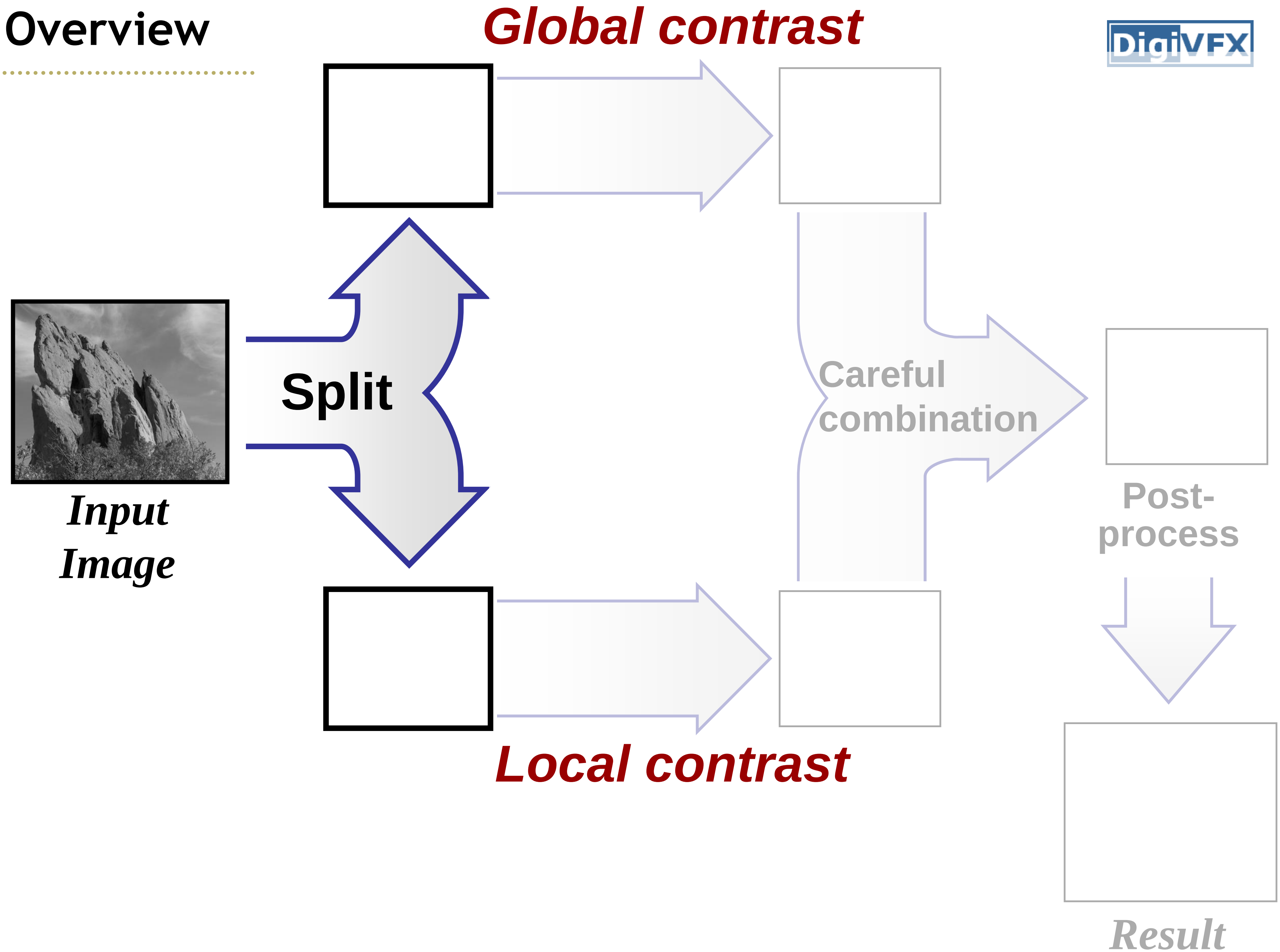


- Separate global and local contrast

Overview

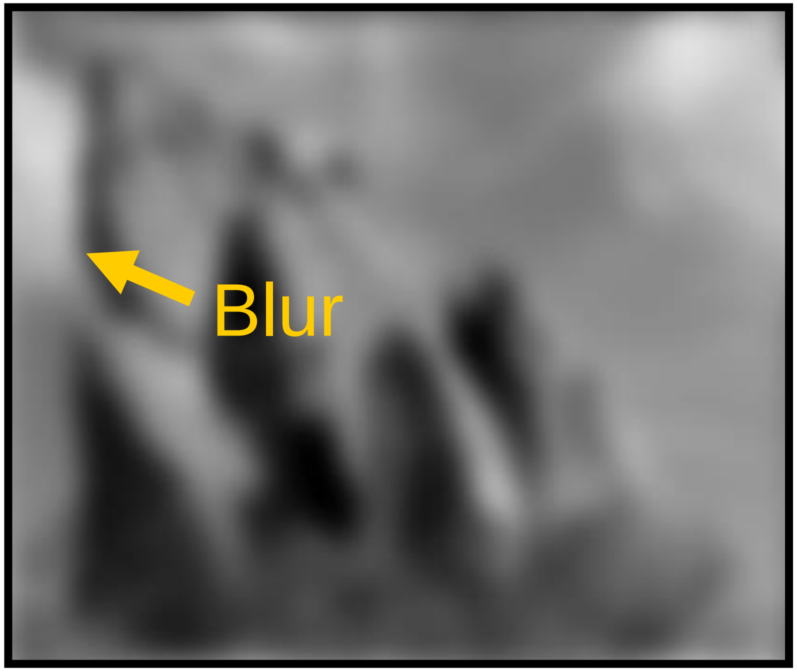

Global contrast
Input Image
Split
Careful combination
Post-process
Local contrast
Result

Overview


Global contrast
Split
Input Image
Careful combination
Post-process
Local contrast
Result

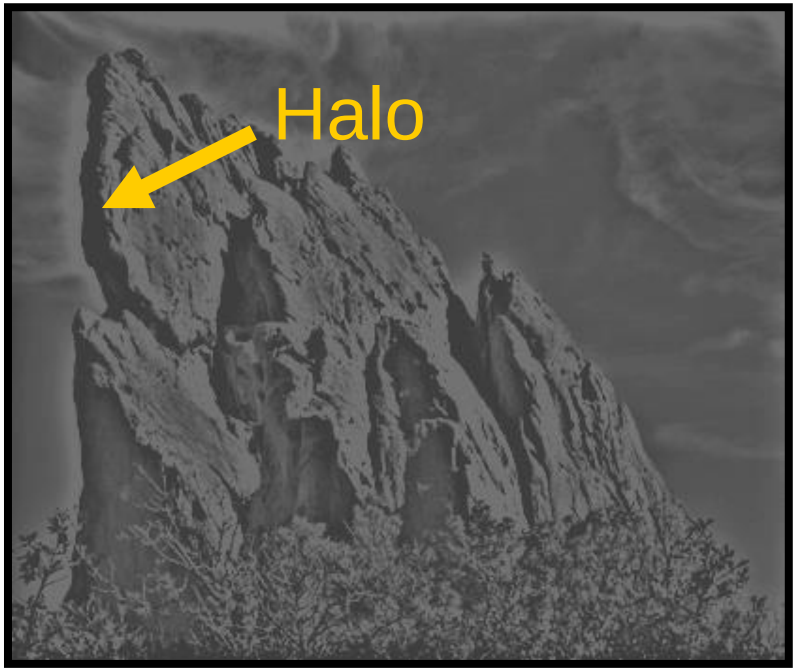
Split Global vs. Local Contrast

- Naïve decomposition: low vs. high frequency
  - Problem: introduce blur & halos



Low frequency

Global contrast



High frequency

Local contrast

Bilateral Filter

- Edge-preserving smoothing [Tomasi 98]
- We build upon tone mapping [Durand 02]

After bilateral filtering

Global contrast

Residual after filtering

Local contrast

Bilateral Filter

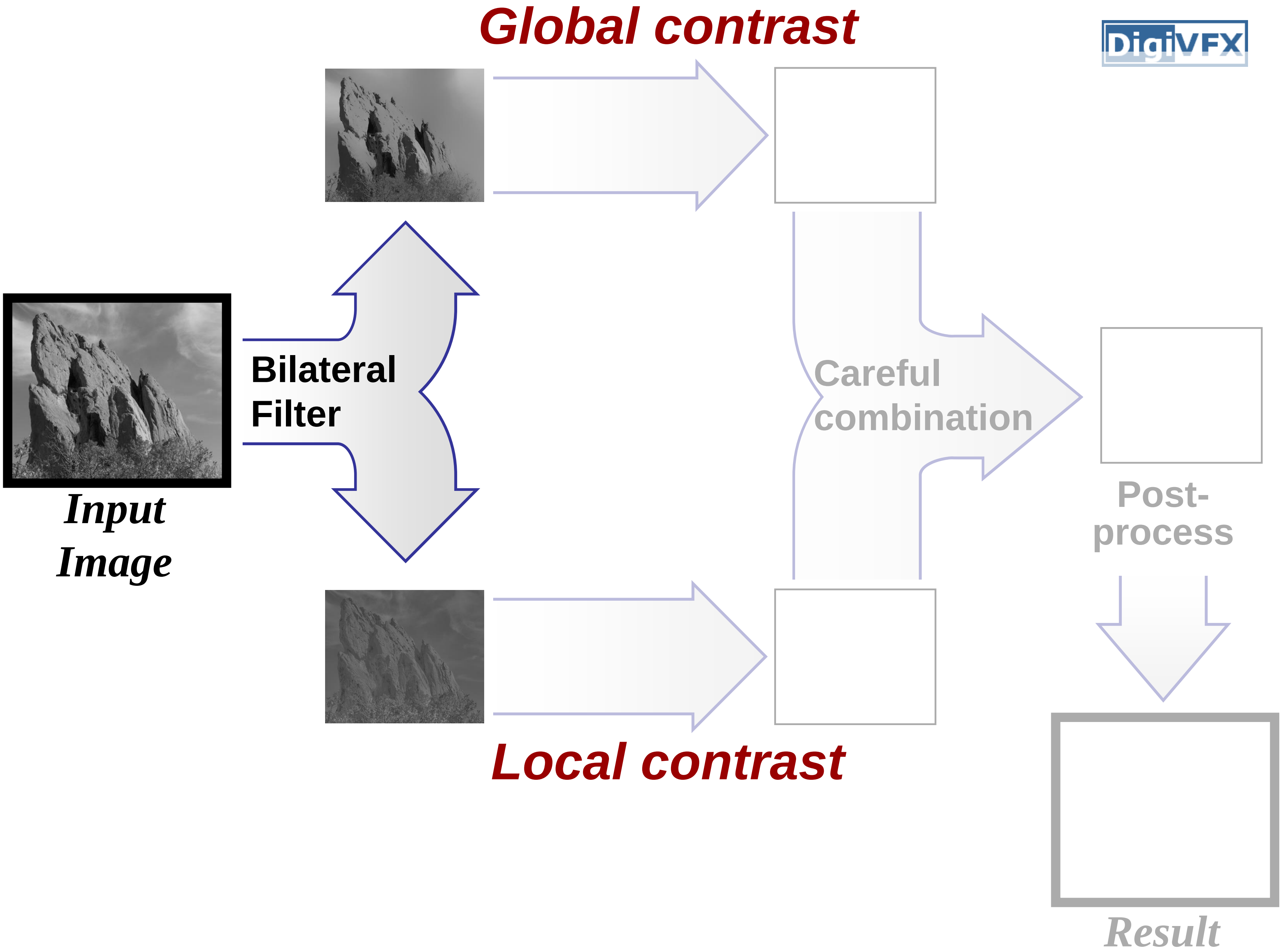
- Edge-preserving smoothing [Tomasi 98]
- We build upon tone mapping [Durand 02]

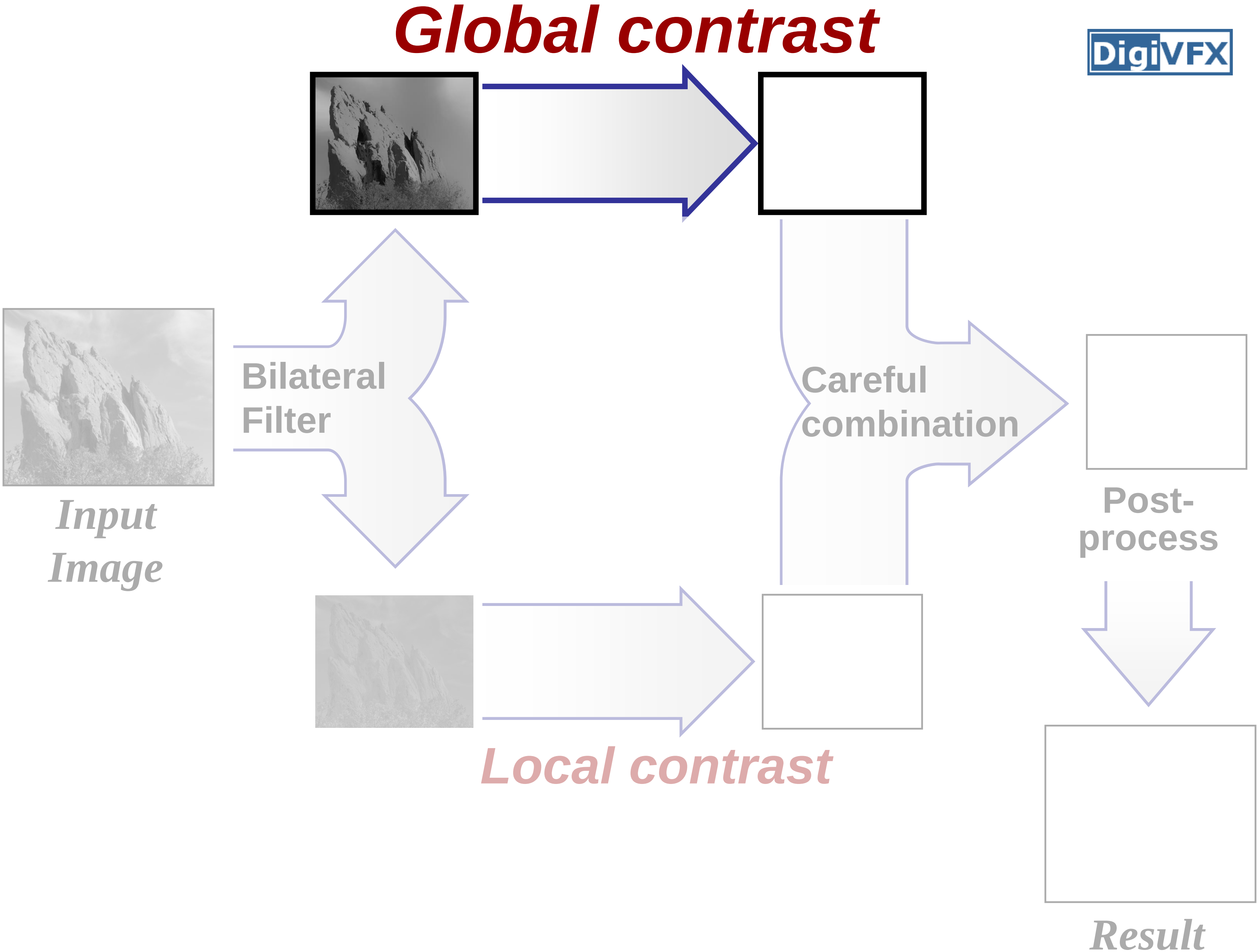
After bilateral filtering

Global contrast

Residual after filtering

Local contrast

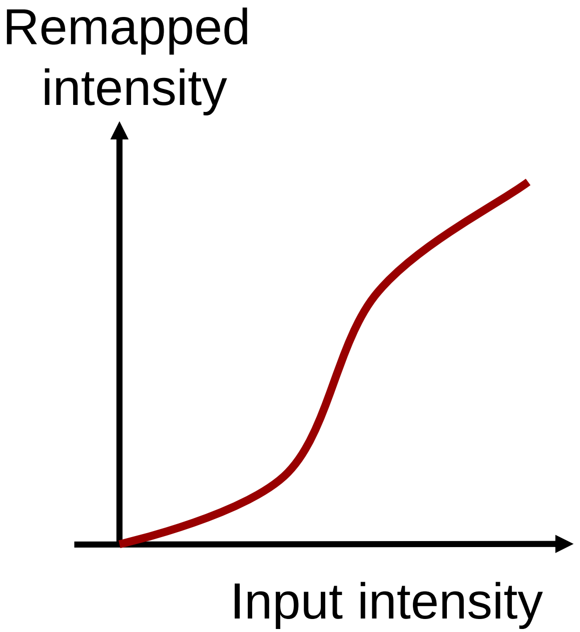

Global contrast
Bilateral
Filter
Careful
combination
Input
Image
Post-
process
Local contrast
Result


Global contrast
Input Image
Bilateral Filter
Careful combination
Post-process
Local contrast
Result

Global Contrast

- Intensity remapping of base layer

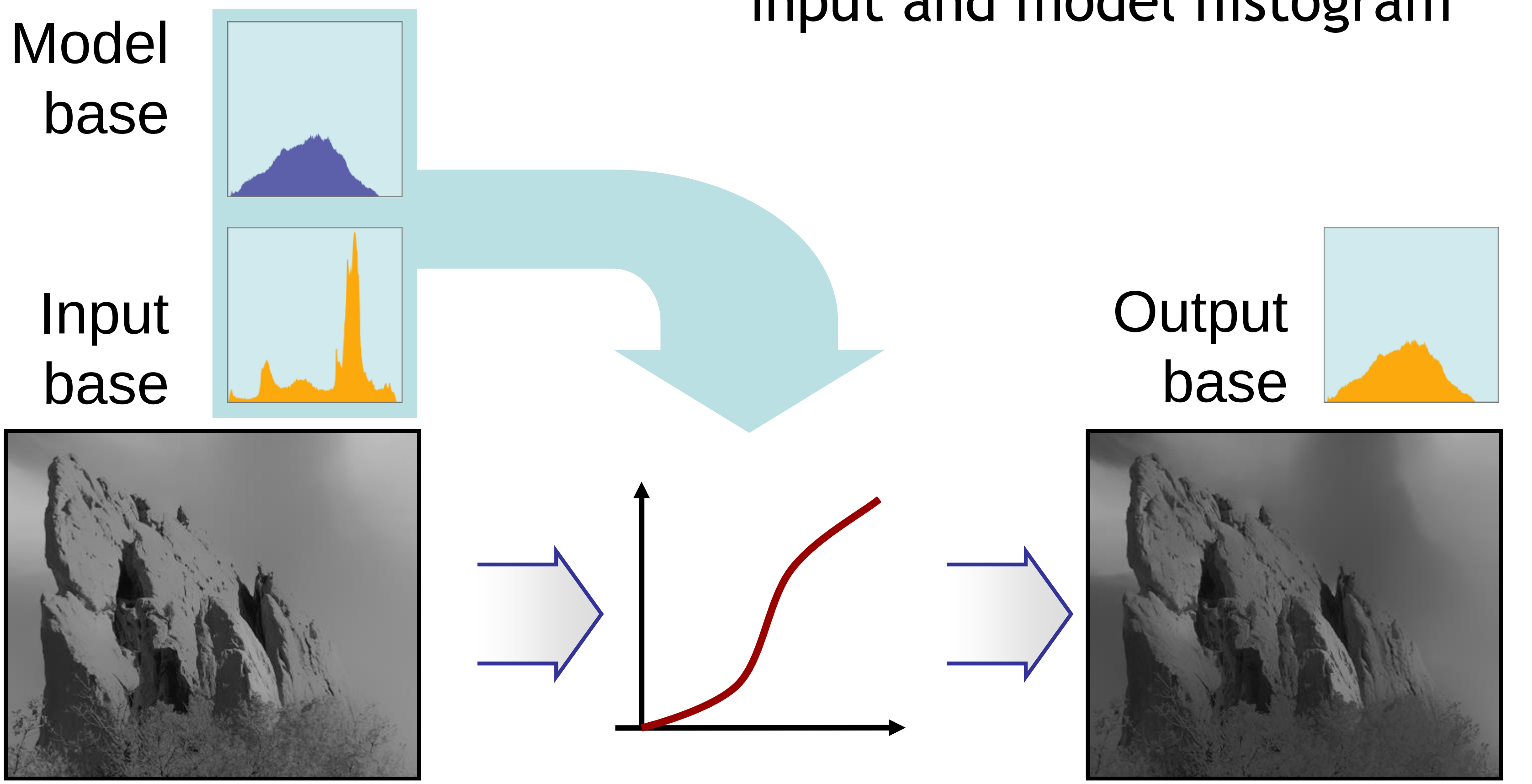
Input base

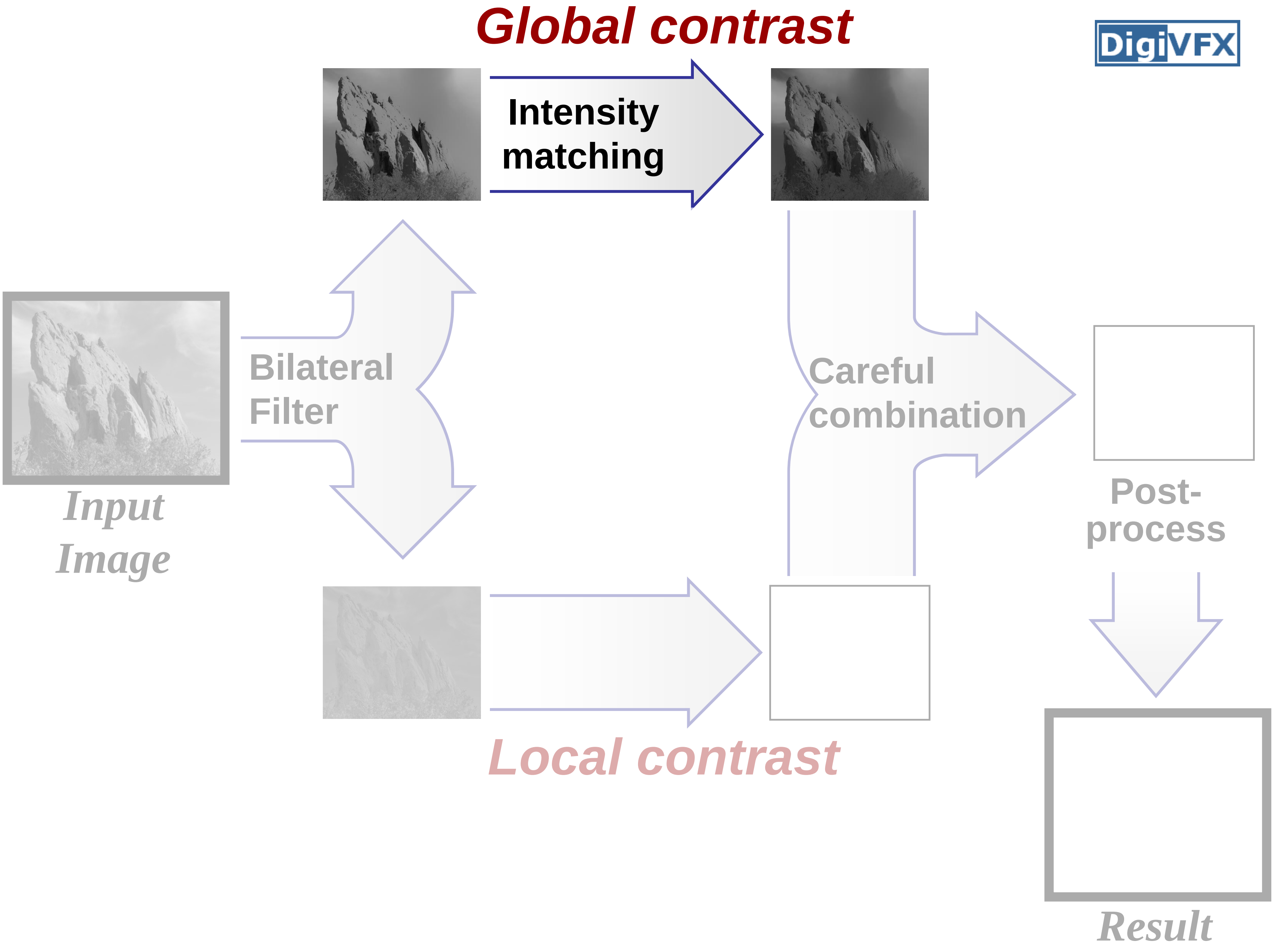


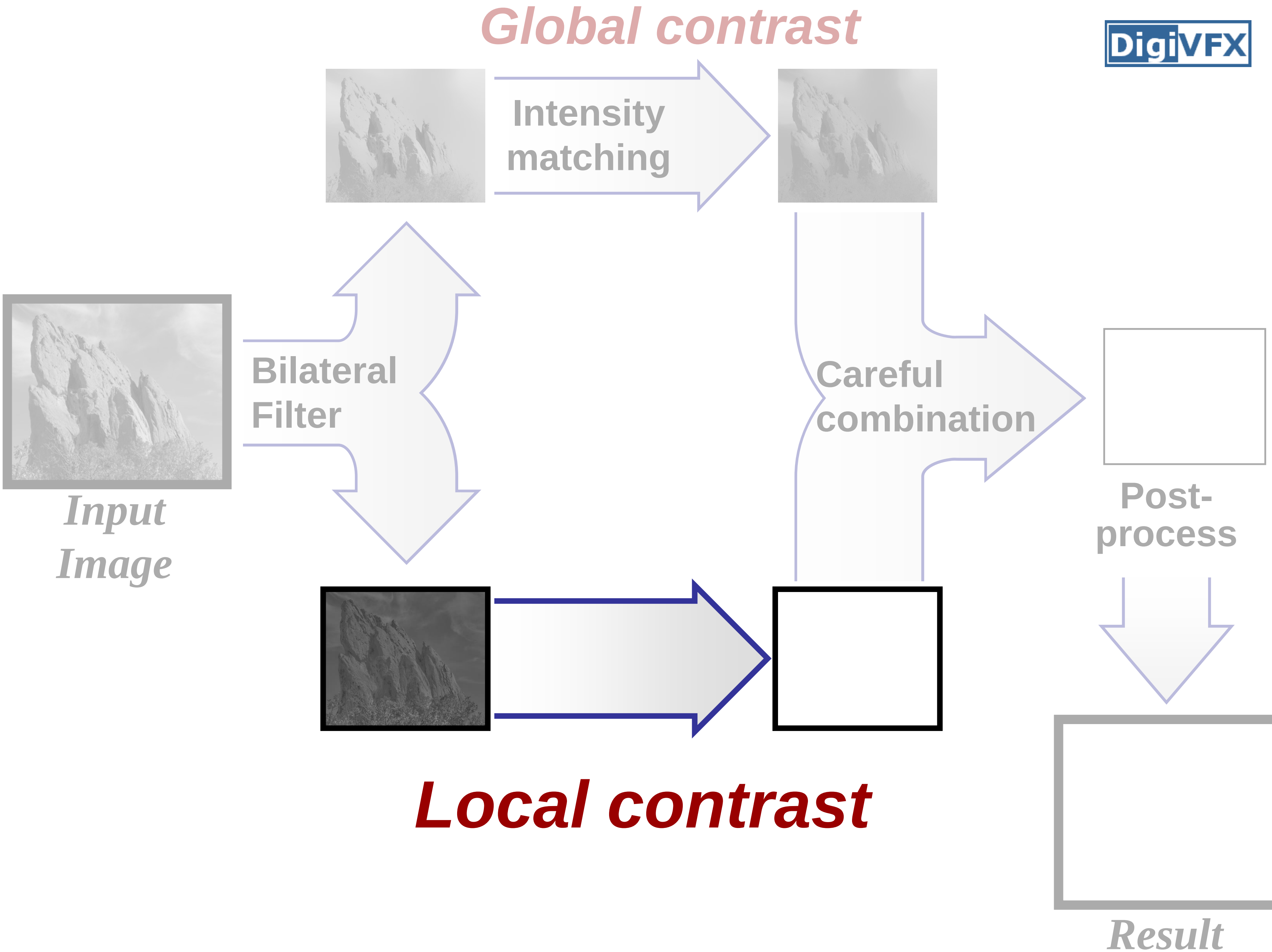
After remapping

Global Contrast (Model Transfer)

- Histogram matching
  - Remapping function given input and model histogram




Global contrast
Intensity matching
Bilateral Filter
Careful combination
Input Image
Post-process
Result
Local contrast

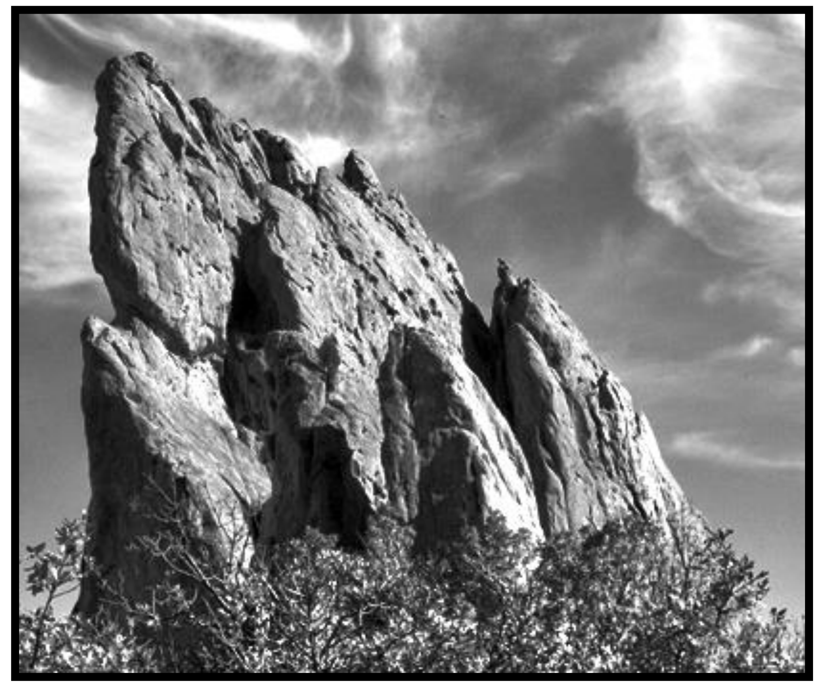


Global contrast
Intensity matching
Input Image
Bilateral Filter
Careful combination
Post-process
Result
Local contrast

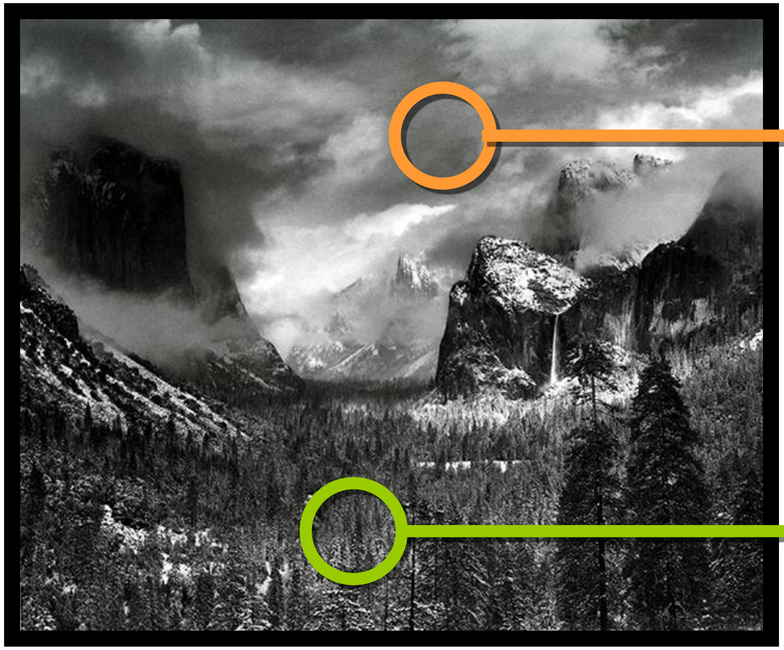
Local Contrast: Detail Layer

- Uniform control:
  - Multiply all values in the detail layer


Input


Base + 3 × Detail

The amount of local contrast is not uniform

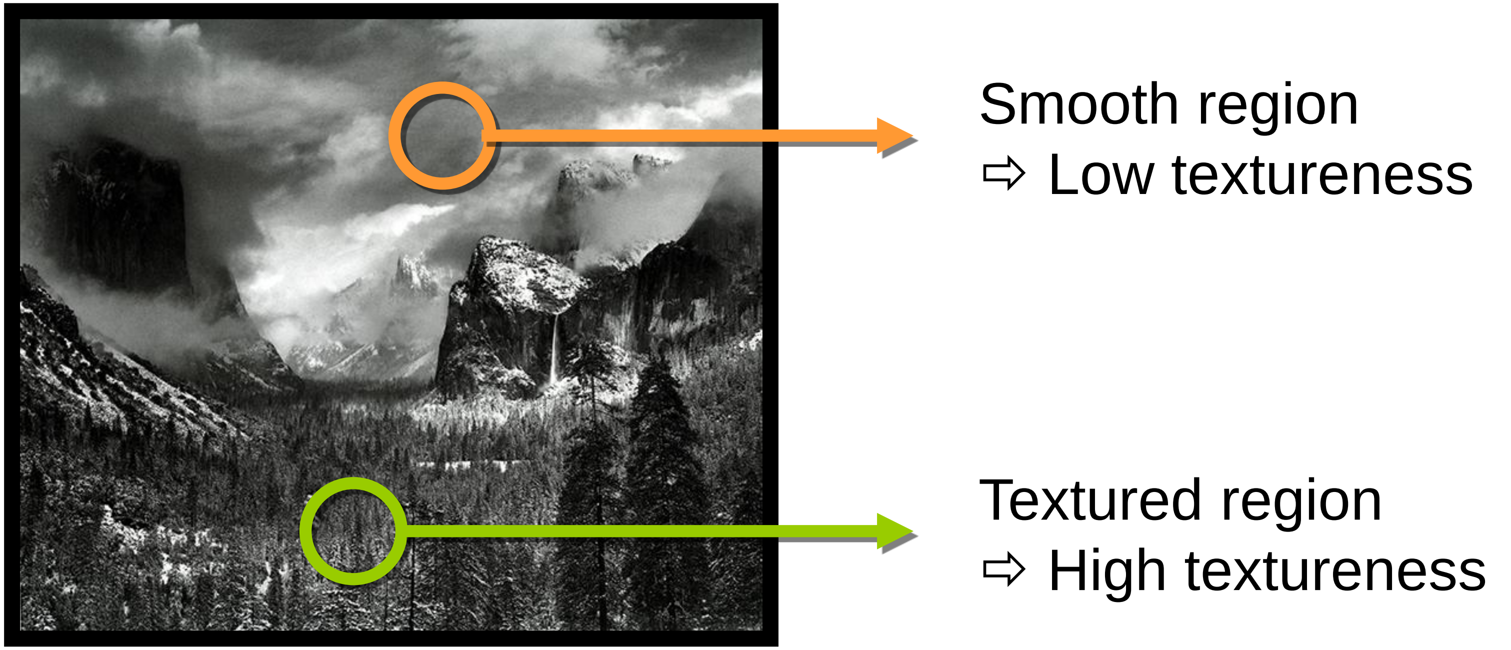


Smooth region

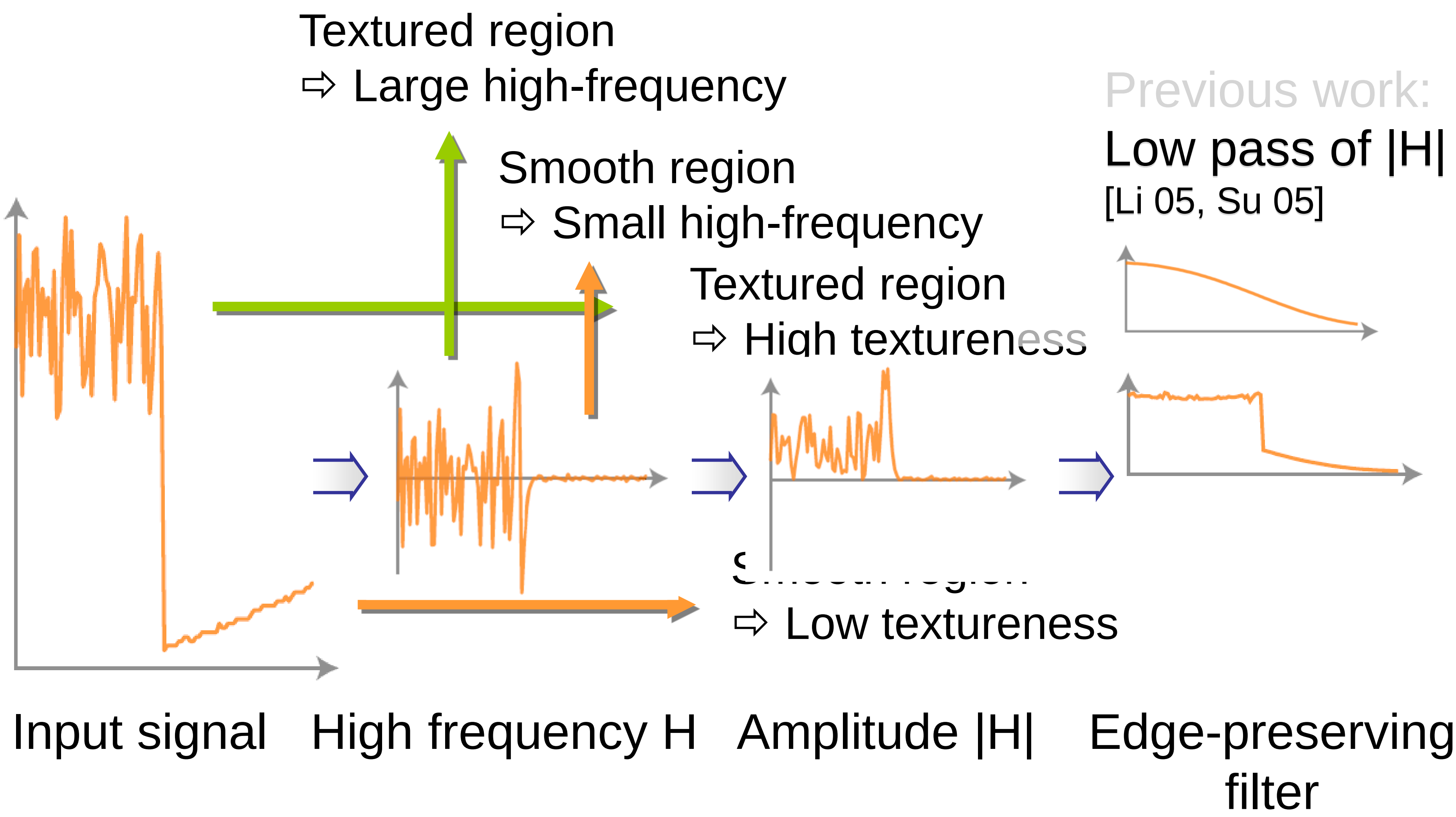
Textured region

Local Contrast Variation

- We define "textureness": amount of local contrast
  - at each pixel based on surrounding region

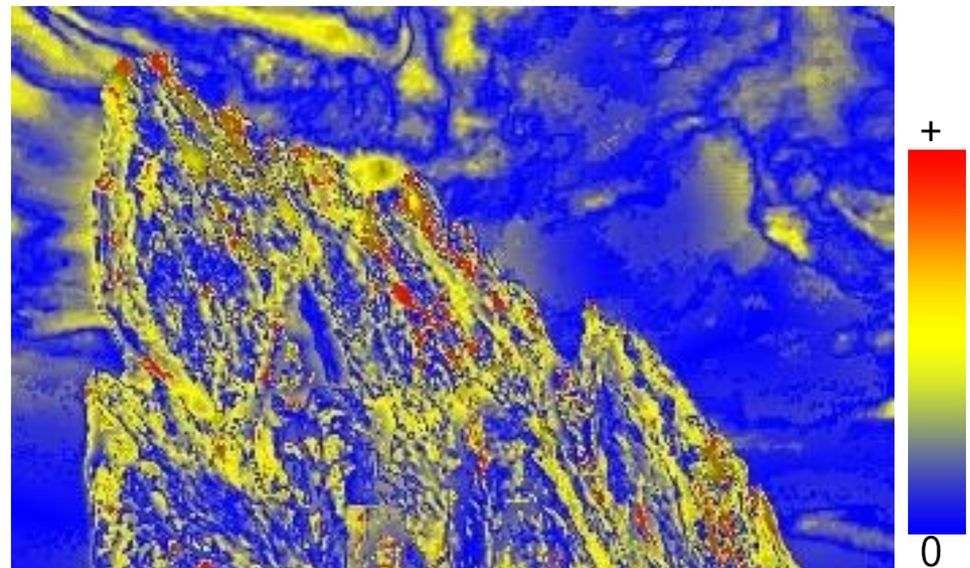


"Textureness": 1D Example



Textureness

Input

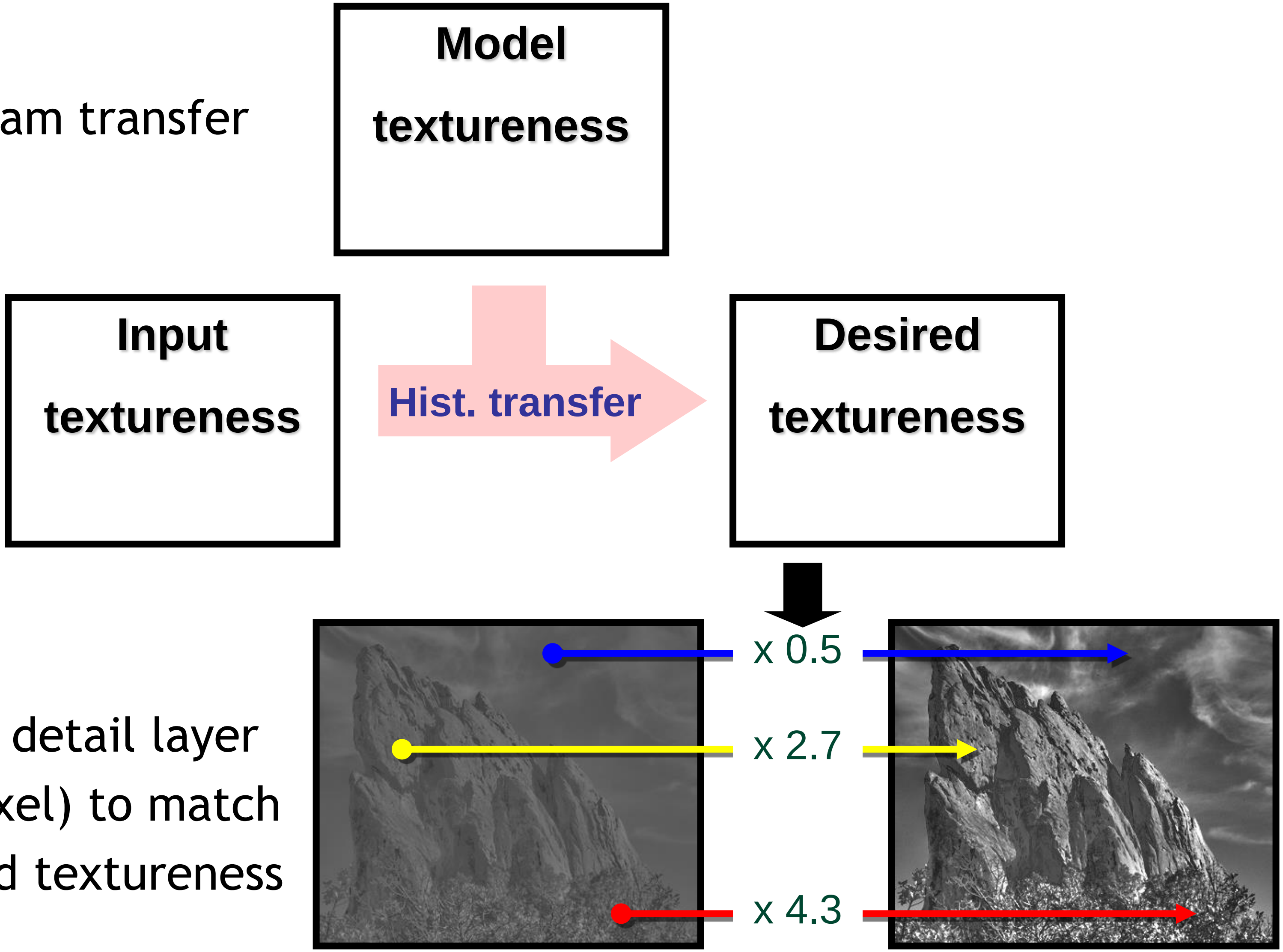


Textureness

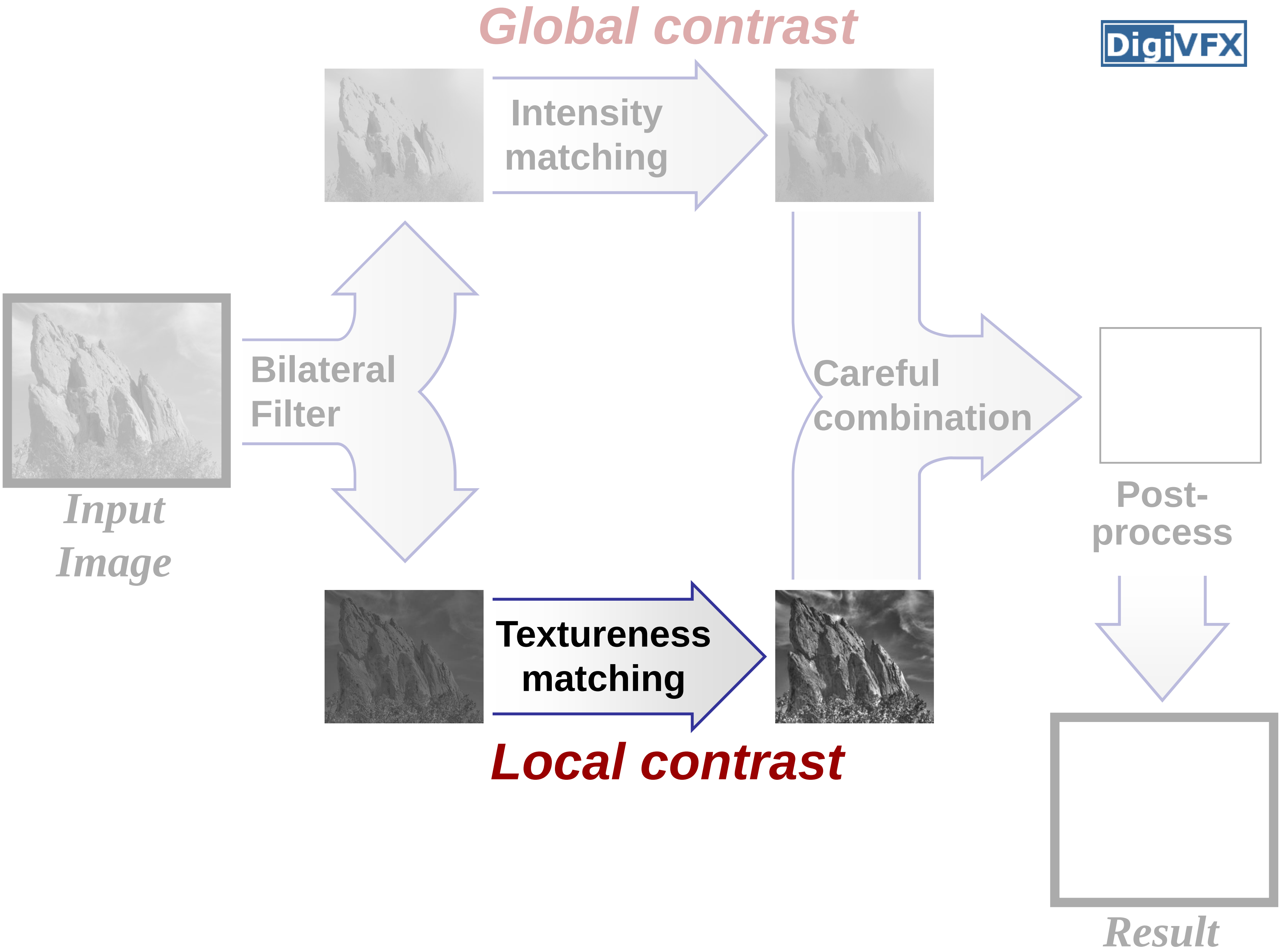
Textureness Transfer

Step 1:
Histogram transfer

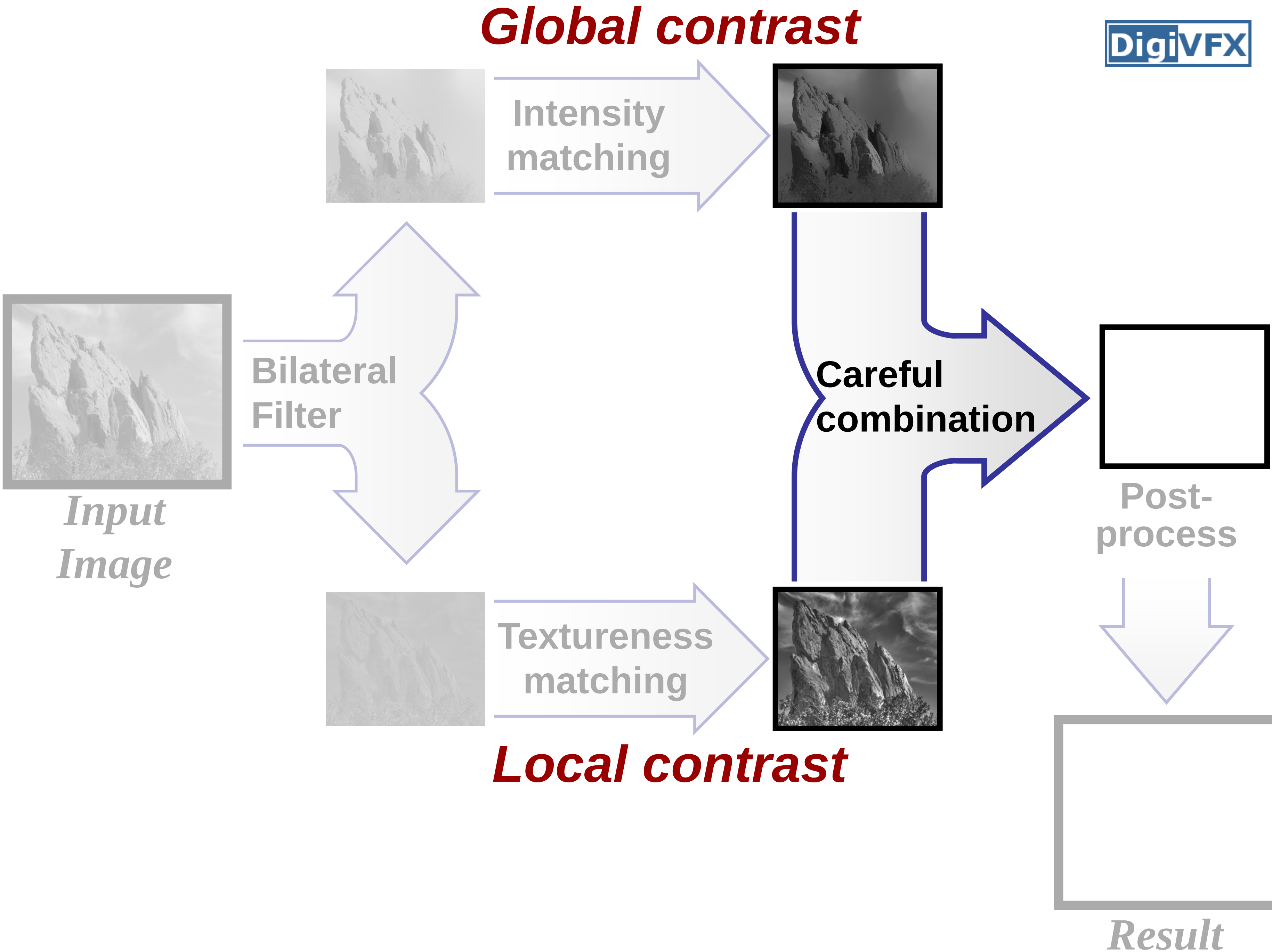
Step 2:
Scaling detail layer
(per pixel) to match
desired textureness

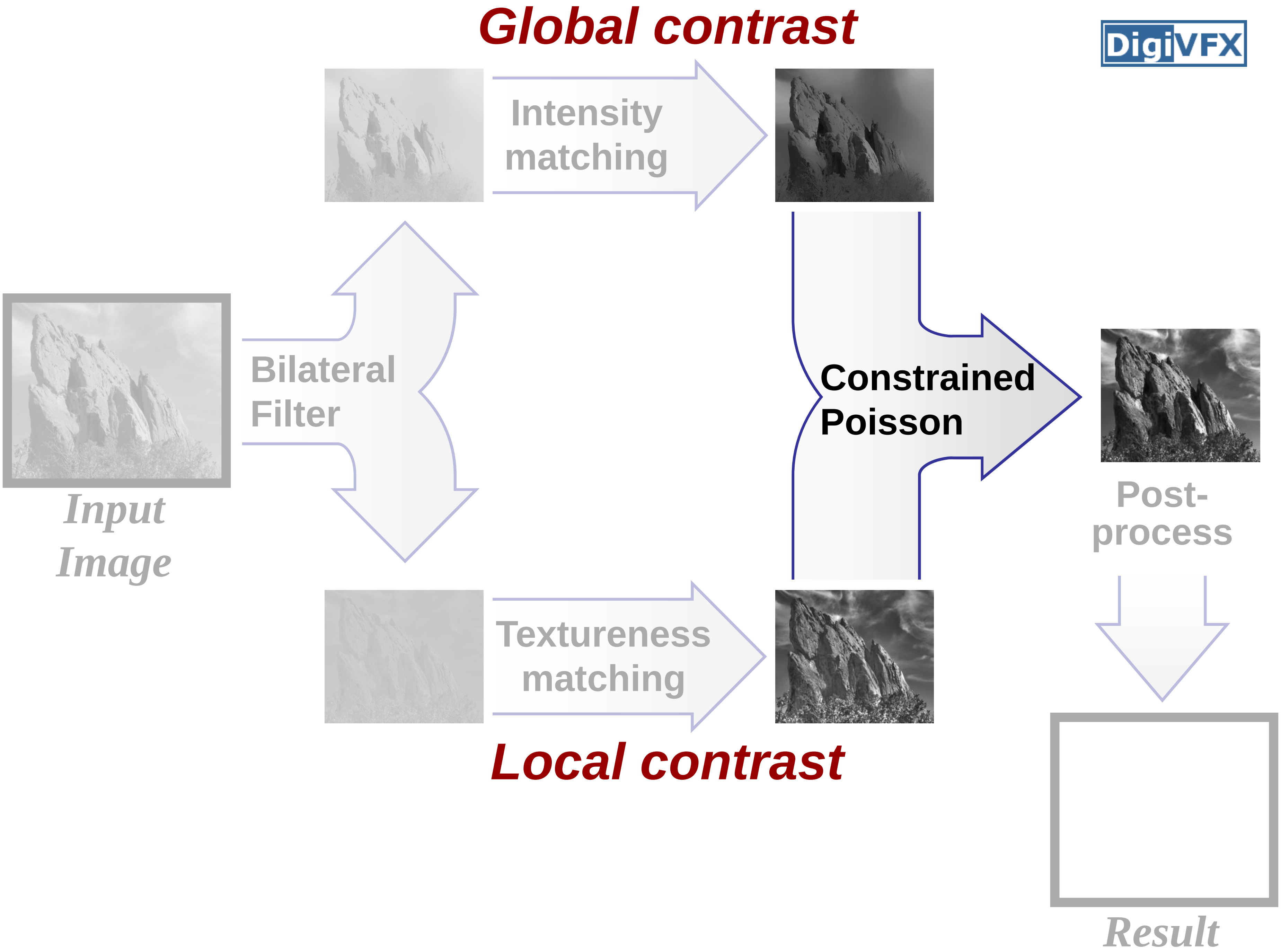


Input detail   Output detail

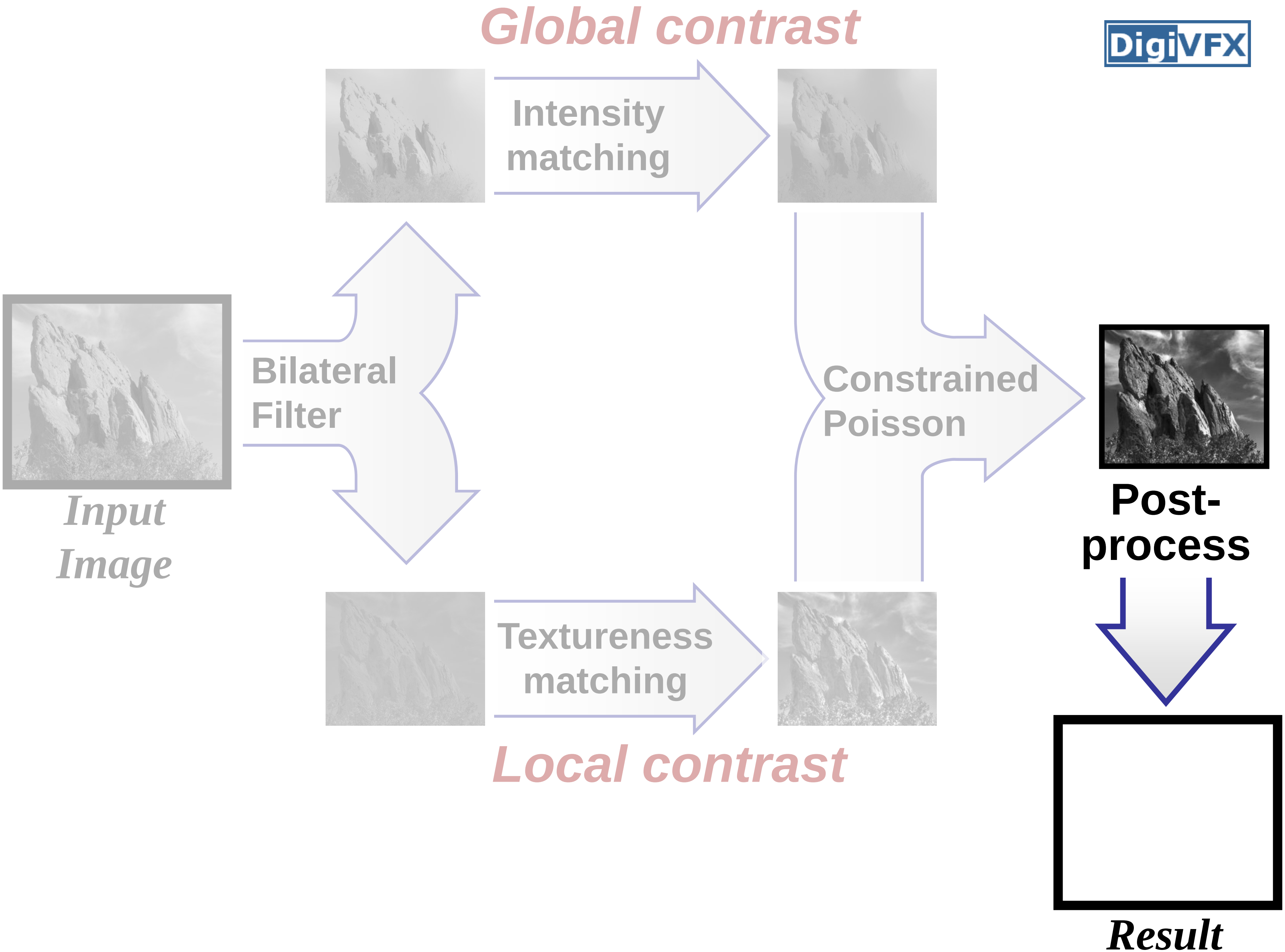


Global contrast
Intensity matching
Bilateral Filter
Careful combination
Input Image
Post-process
Textureness matching
Local contrast
Result


Global contrast
Intensity matching
Bilateral Filter
Input Image
Careful combination
Post-process
Result
Textureness matching
Local contrast



Global contrast
Intensity matching
Bilateral Filter
Input Image
Constrained Poisson
Post-process
Textureness matching
Local contrast
Result


Global contrast
Intensity matching
Input Image
Bilateral Filter
Constrained Poisson
Post-process
Textureness matching
Local contrast
Result

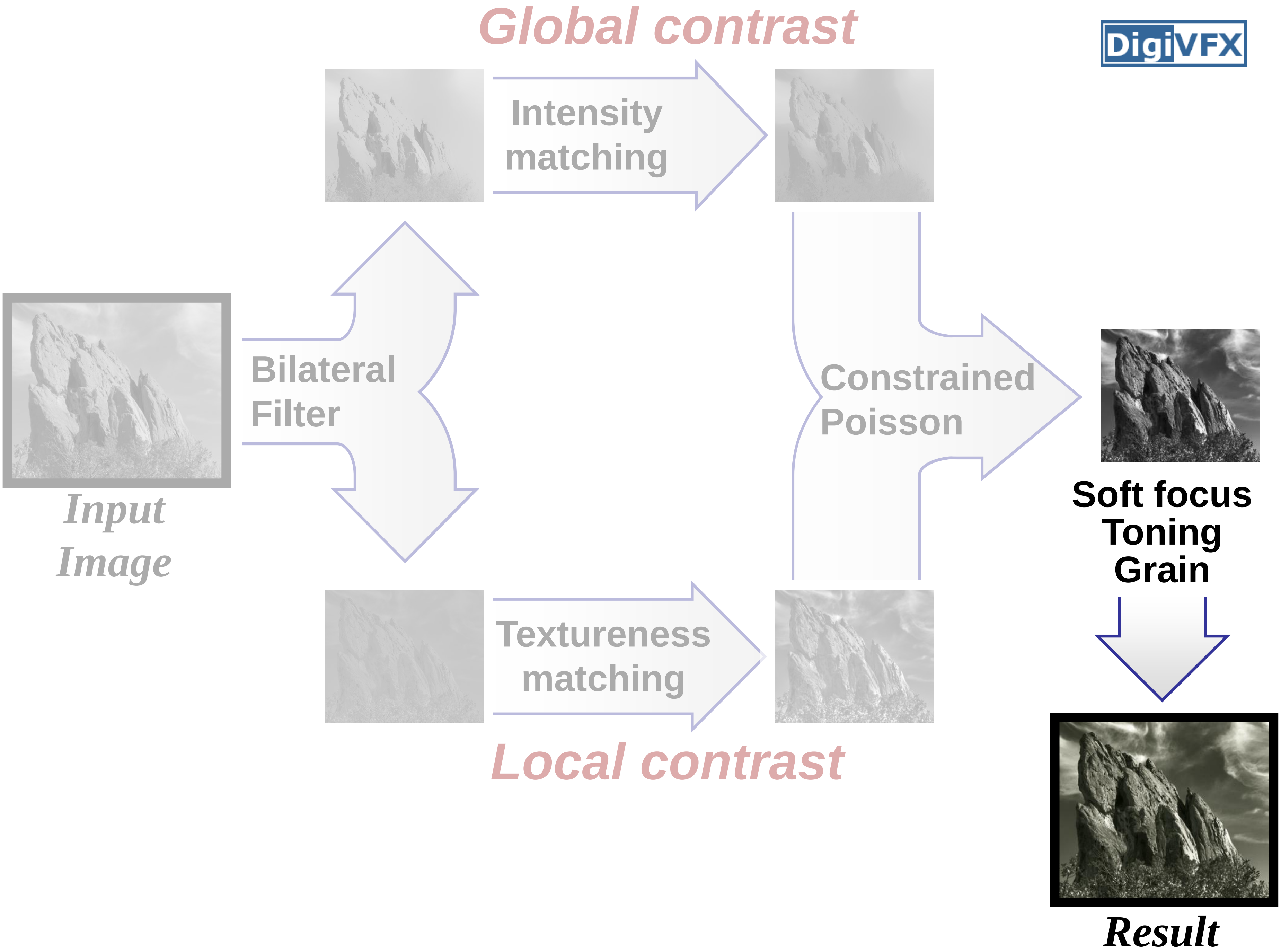
Additional Effects

- **Soft focus** (high frequency manipulation)
- **Film grain** (texture synthesis [Heeger 95])
- **Color toning** (chrominance = f (luminance))

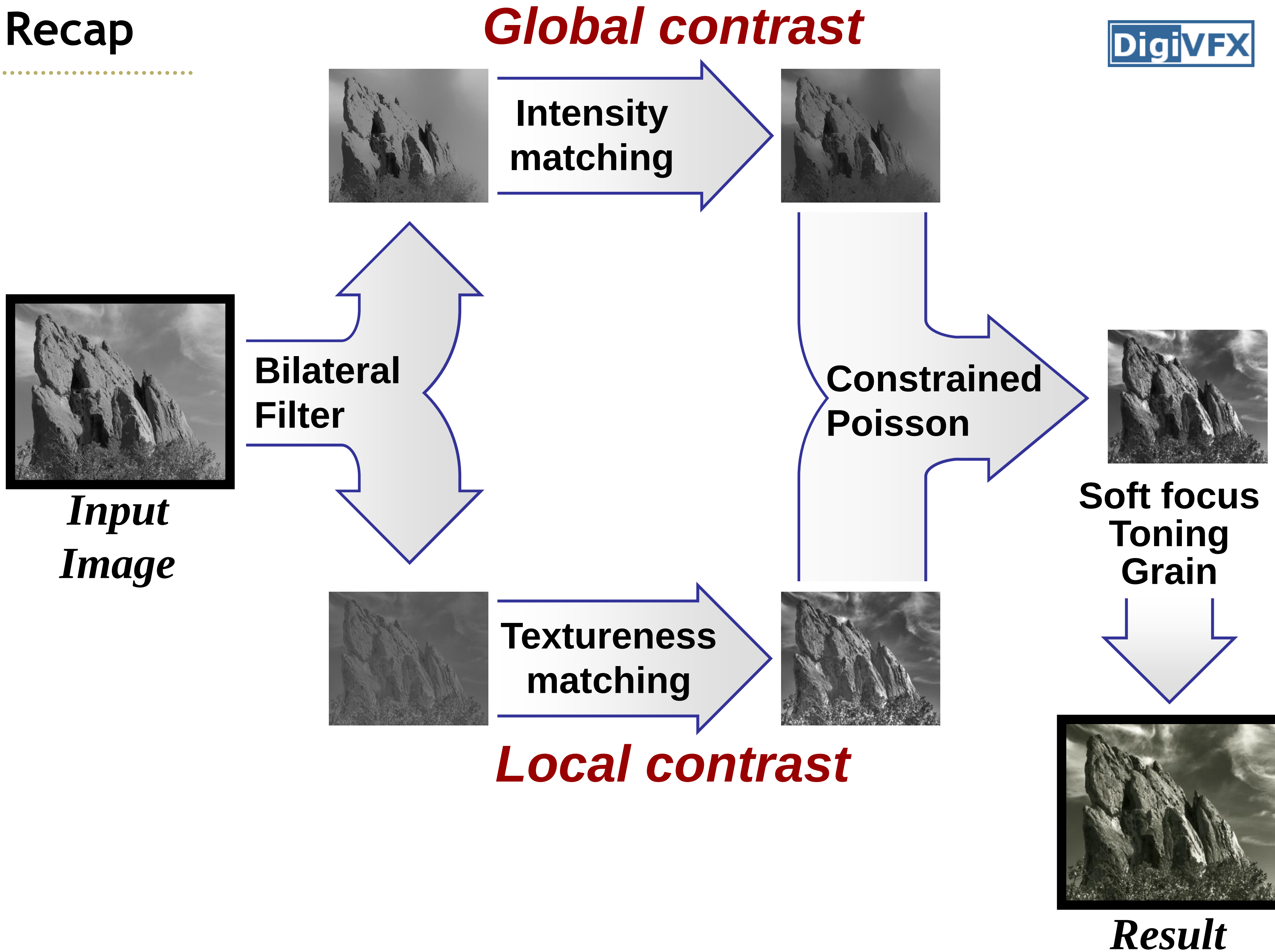
model

before effects

after effects


Global contrast
Intensity matching
Bilateral Filter
Input Image
Constrained Poisson
Soft focus
Toning
Grain
Textureness matching
Local contrast
Result

Recap



Global contrast
Intensity matching
Bilateral Filter
Constrained Poisson
Input Image
Soft focus
Toning
Grain
Textureness matching
Local contrast
Result

Results

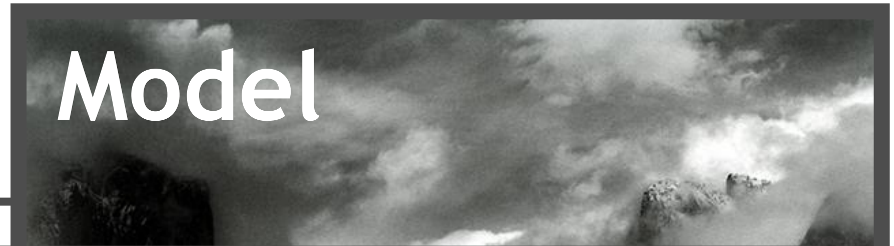
User provides input and model photographs.
→ Our system **automatically** produces the result.

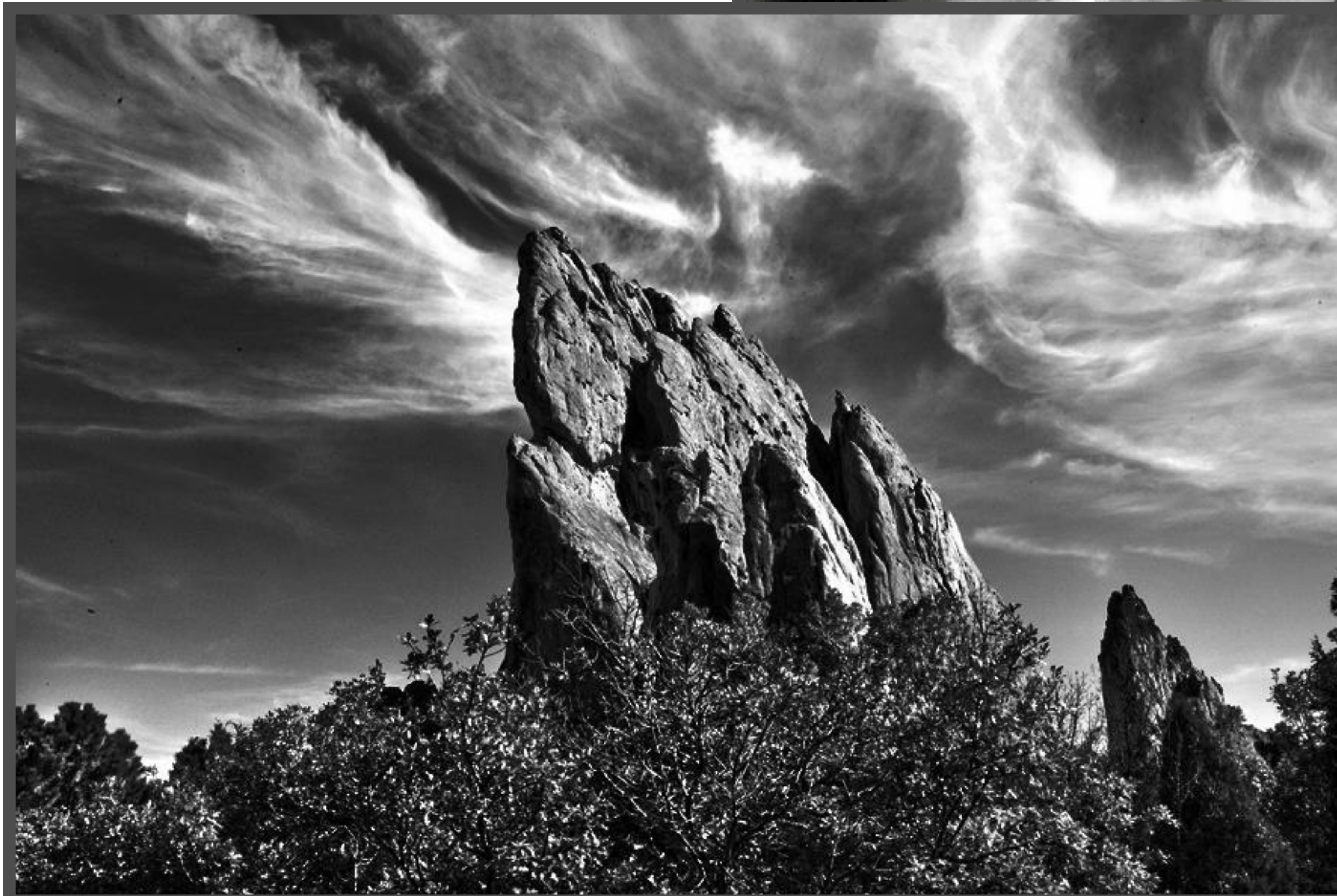
Running times:

- 6 seconds for 1 MPixel or less
- 23 seconds for 4 MPixels
  - multi-grid Poisson solver and fast bilateral filter [Paris 06]

Result

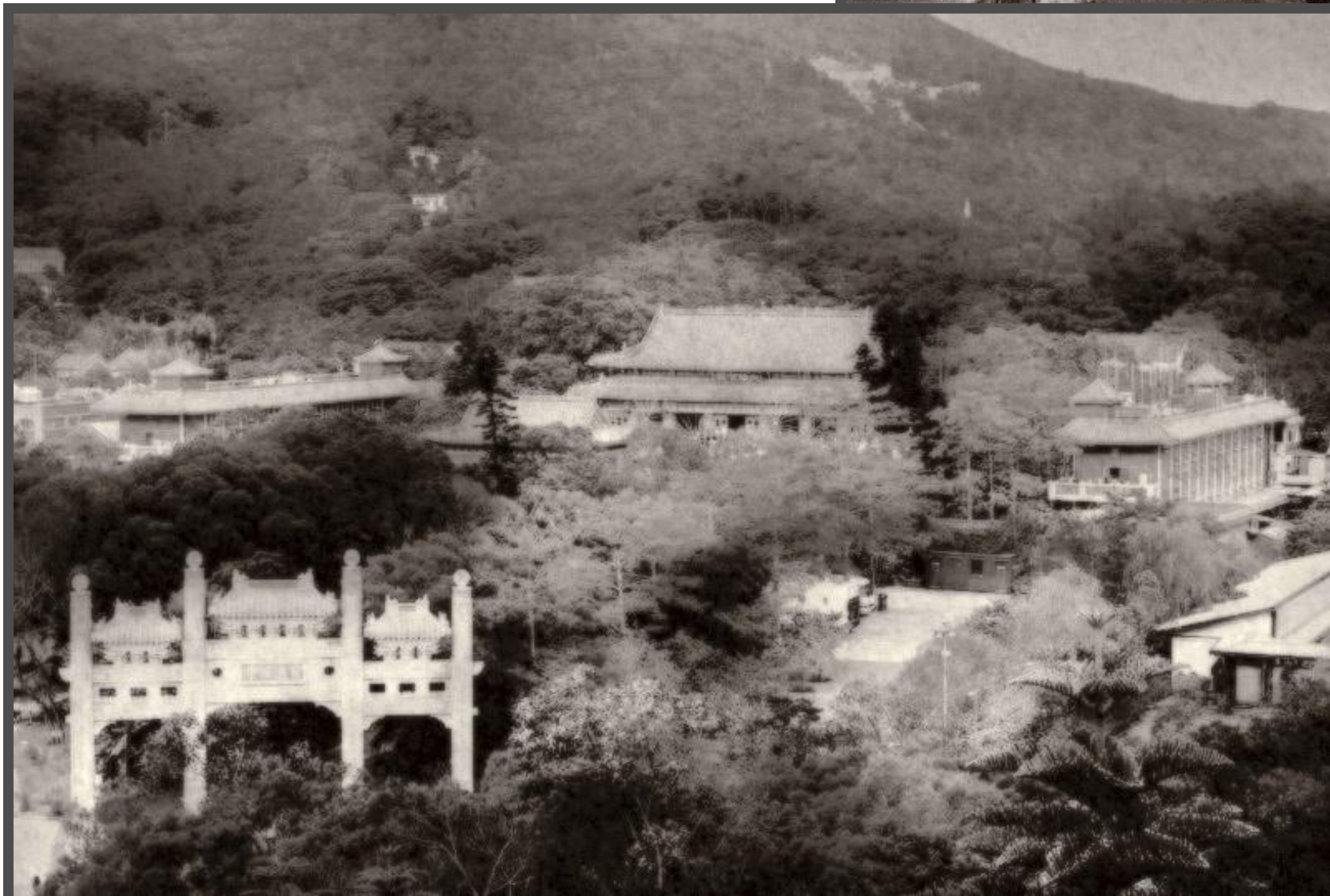
Input



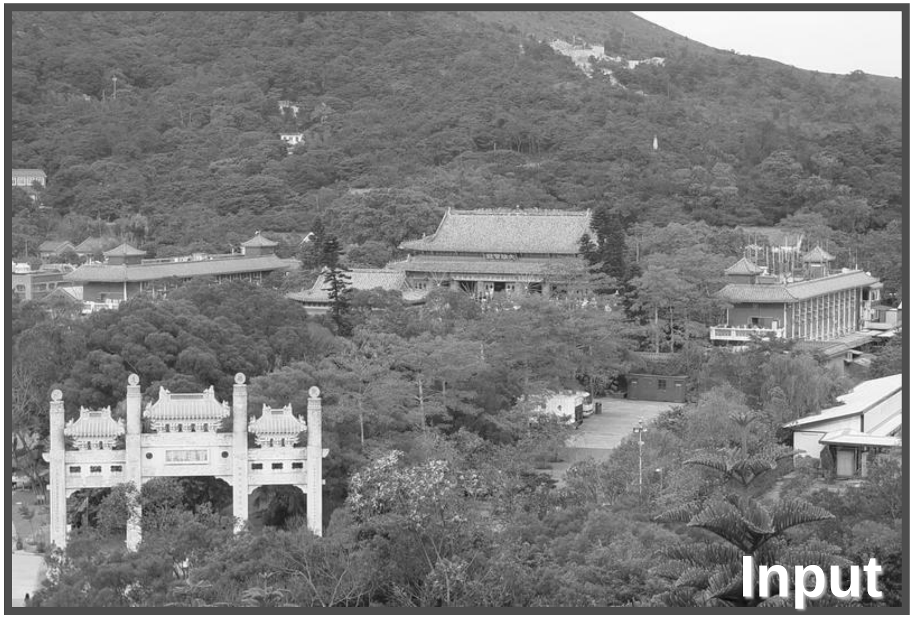


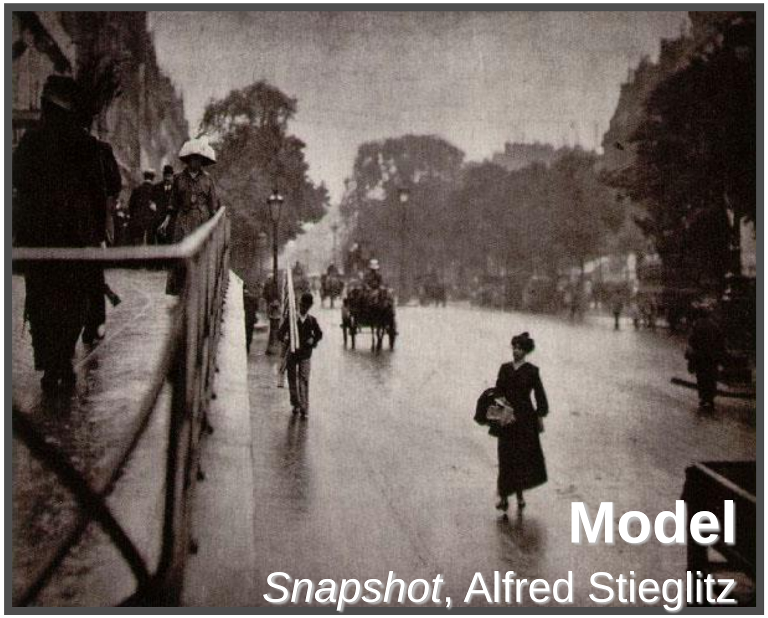
Result

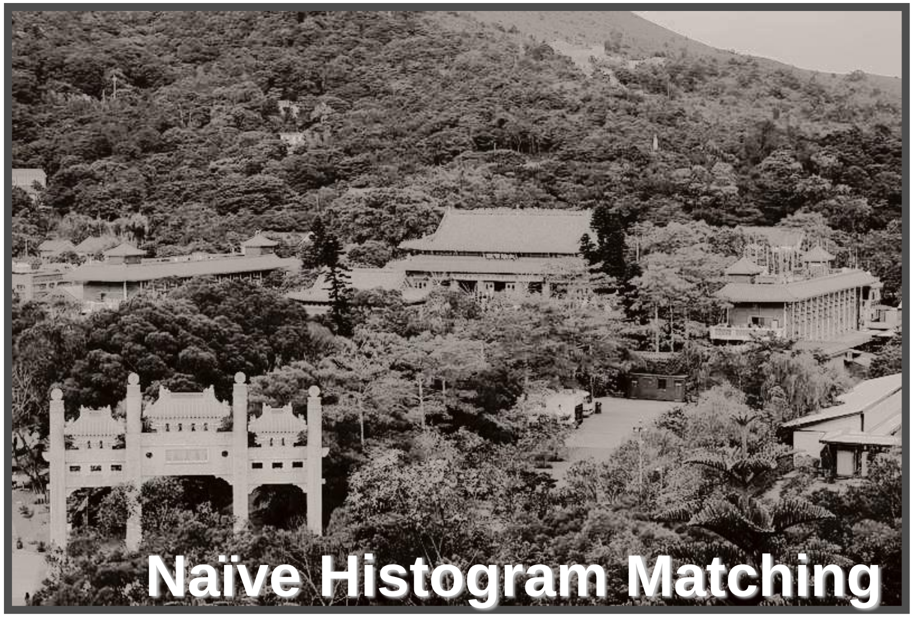
Result

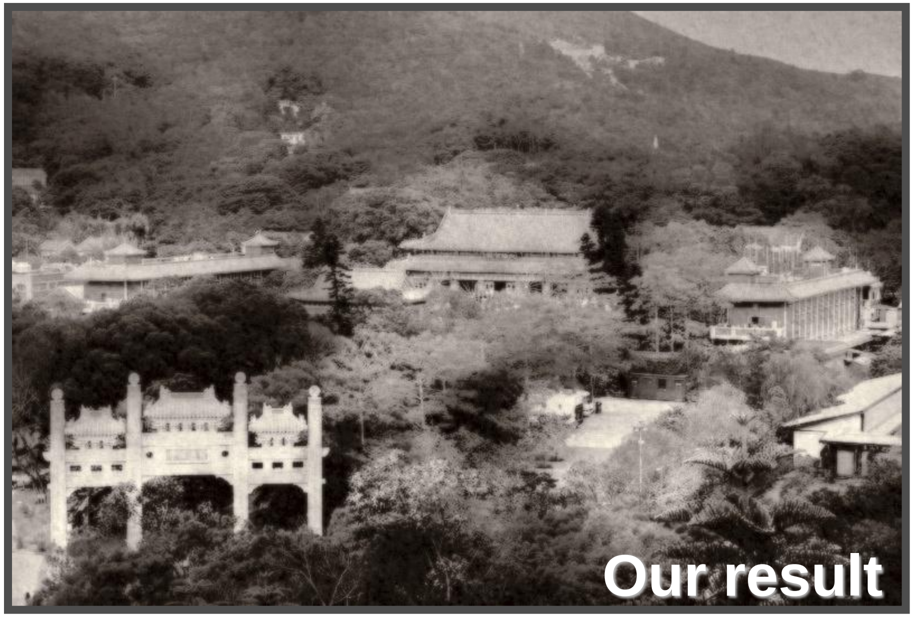


Comparison with Naïve Histogram Matching



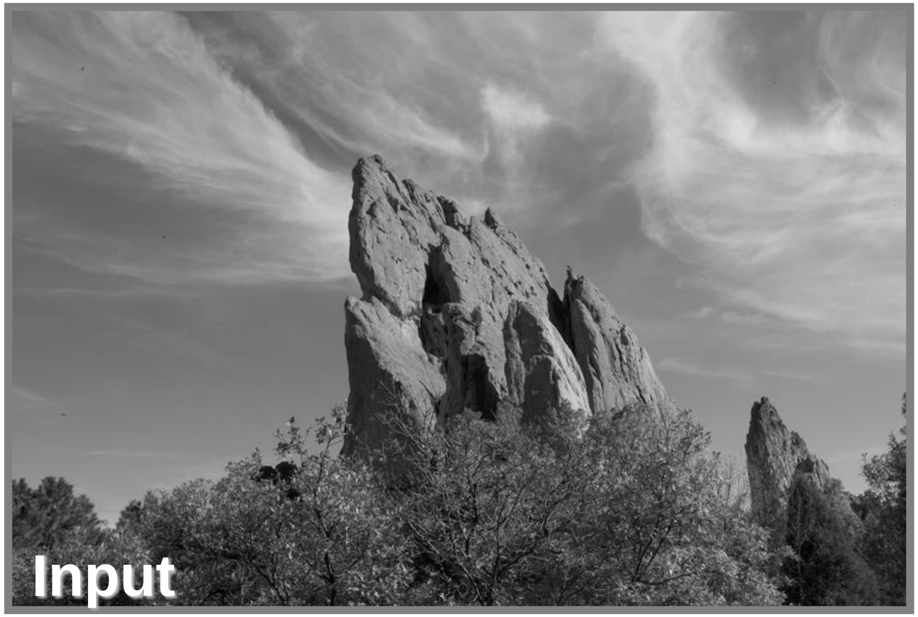






Local contrast, sharpness unfaithful

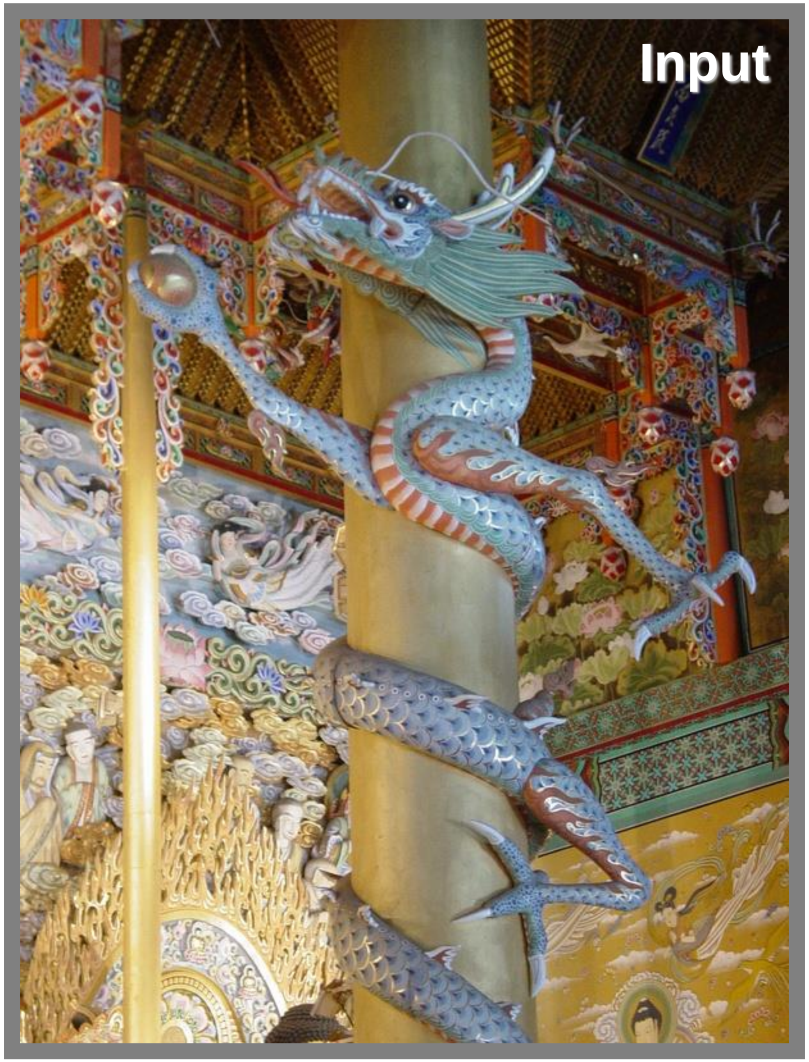
Comparison with Naïve Histogram Matching

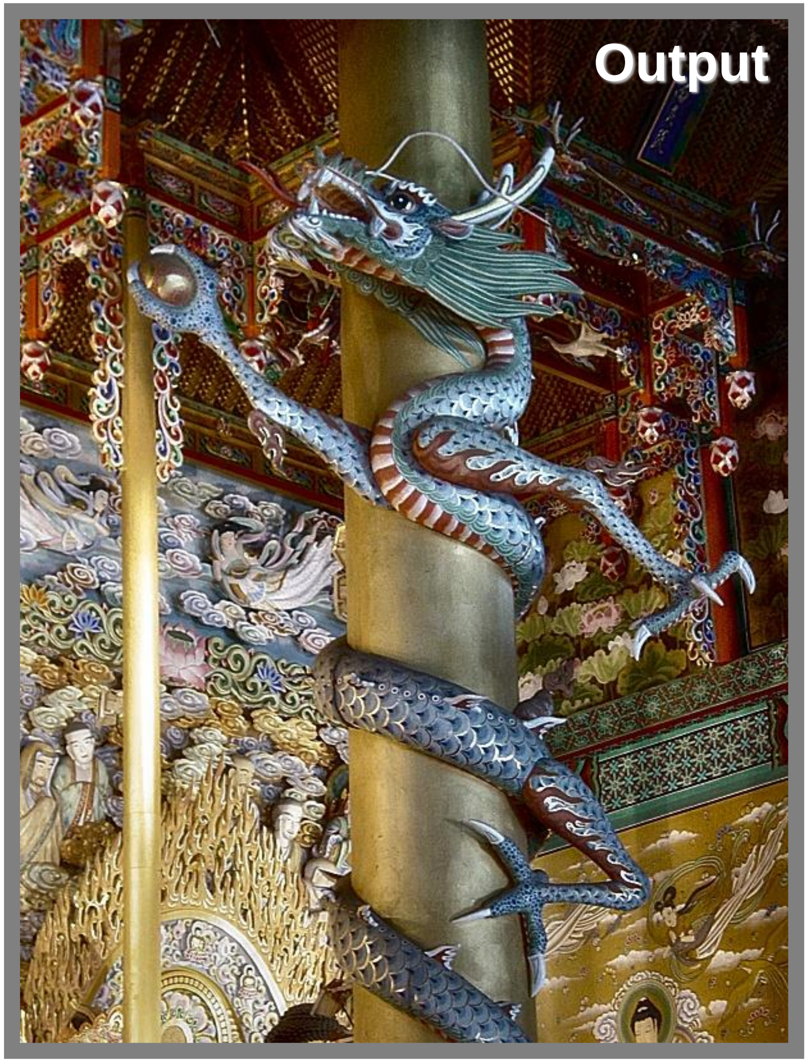


Local contrast too low

Color Images

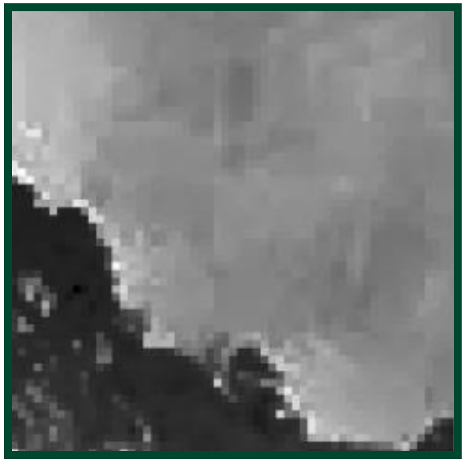
- Lab color space: modify only luminance





Limitations

- Noise and JPEG artifacts
  - amplified defects



- Can lead to unexpected results if the image content is too different from the model
  - Portraits, in particular, can suffer

Conclusions

- Transfer “look” from a model photo
- Two-scale tone management
  - Global and local contrast
  - New edge-preserving textureness
  - Constrained Poisson reconstruction
  - Additional effects

Joint bilateral filtering

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega} I_q \, f(||p-q||) \, g(||I_p - I_q||)$$

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega} I_q \, f(||p-q||) \, g(||\tilde{I}_p - \tilde{I}_q||)$$

Flash / No-Flash Photo Improvement (Petschnigg04) (Eisemann04)

Merge best features: warm, cozy candle light (no-flash)
low-noise, detailed flash image

Overview

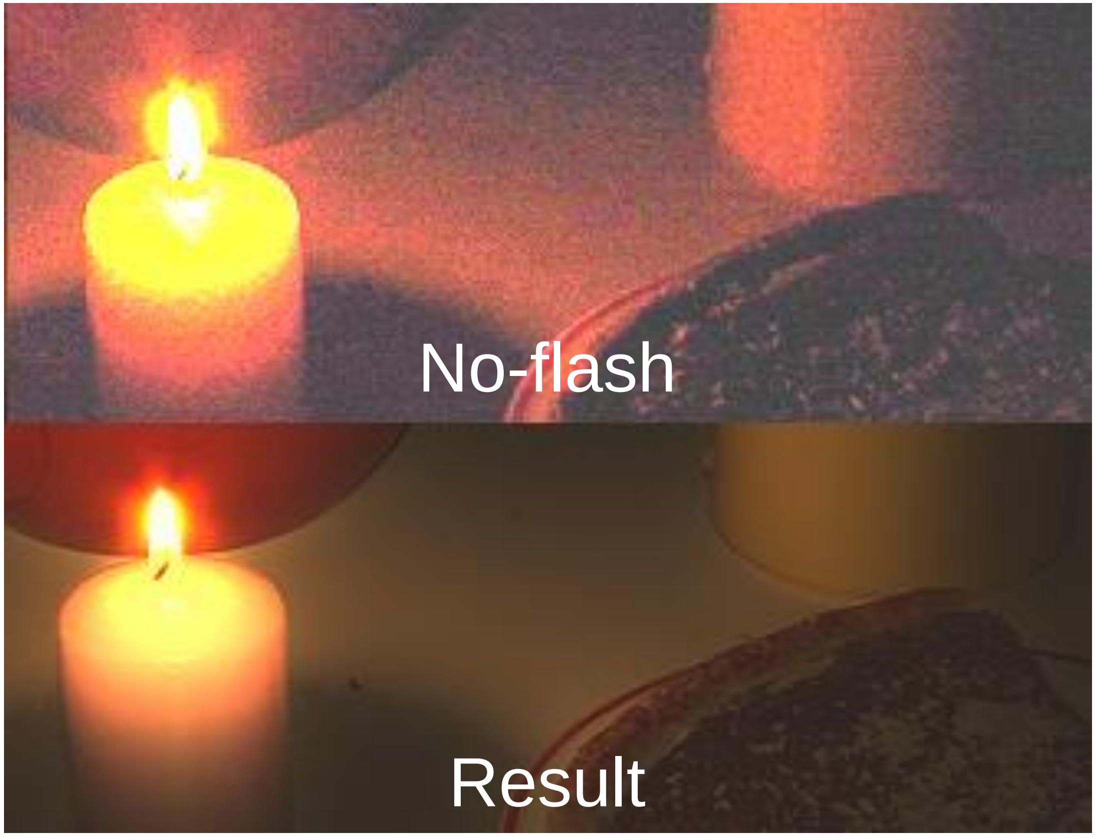
Basic approach of both flash/noflash papers

Remove noise + details
from image A,

Keep as image A Lighting

Obtain noise-free details
from image B,

Discard Image B Lighting



Petschnigg:

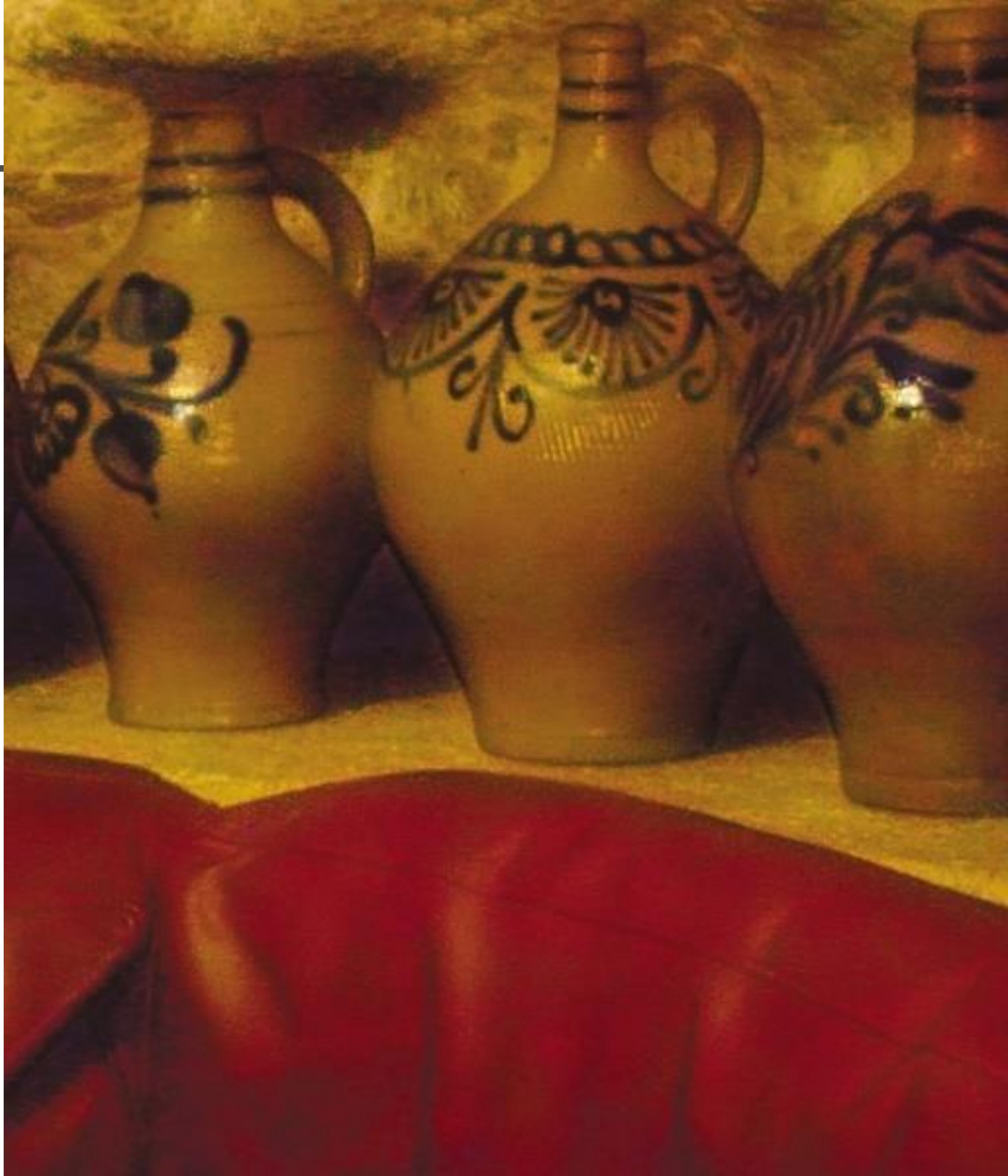
- Flash

Petschnigg:

- No Flash,

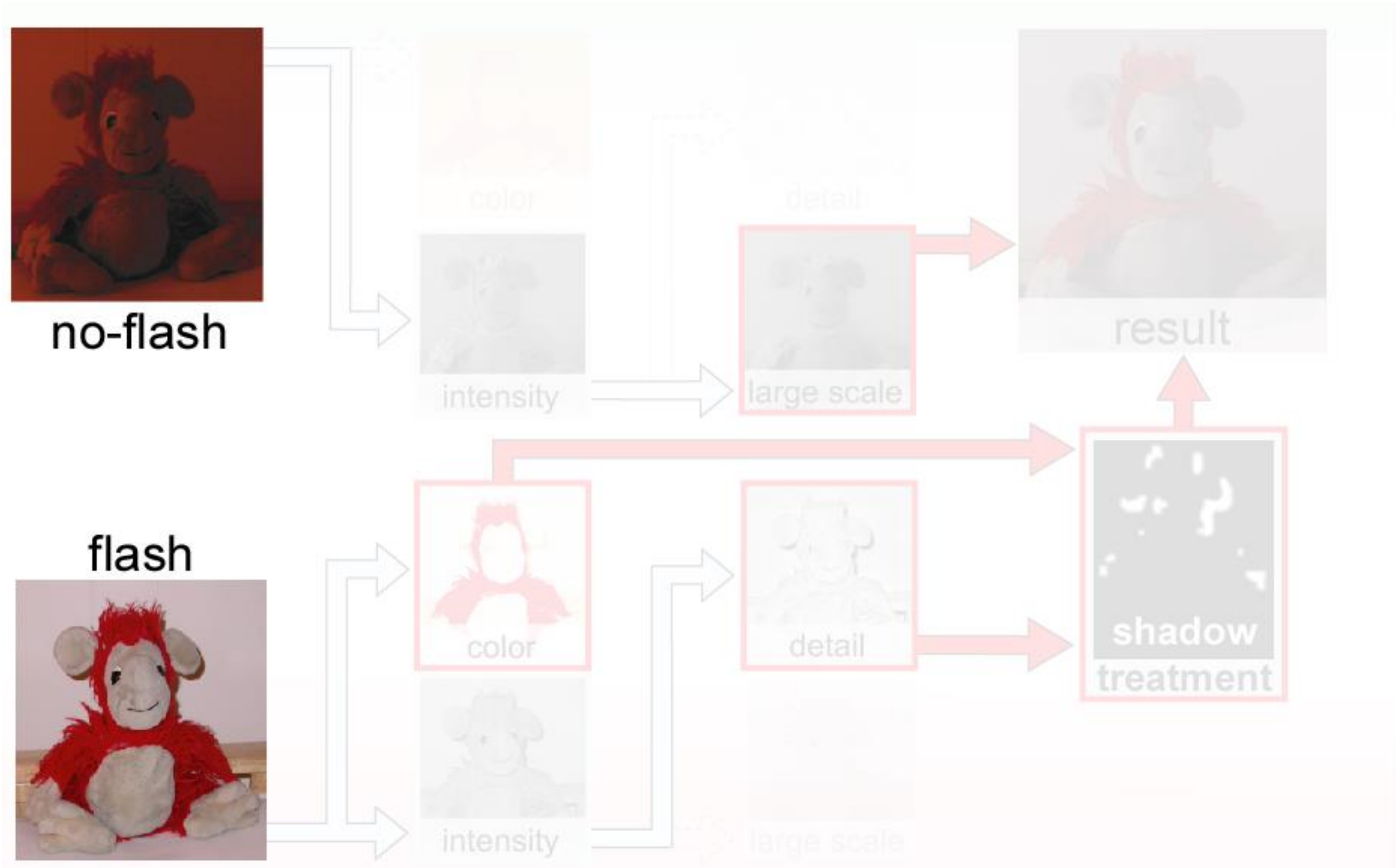
Petschnigg:

- Result



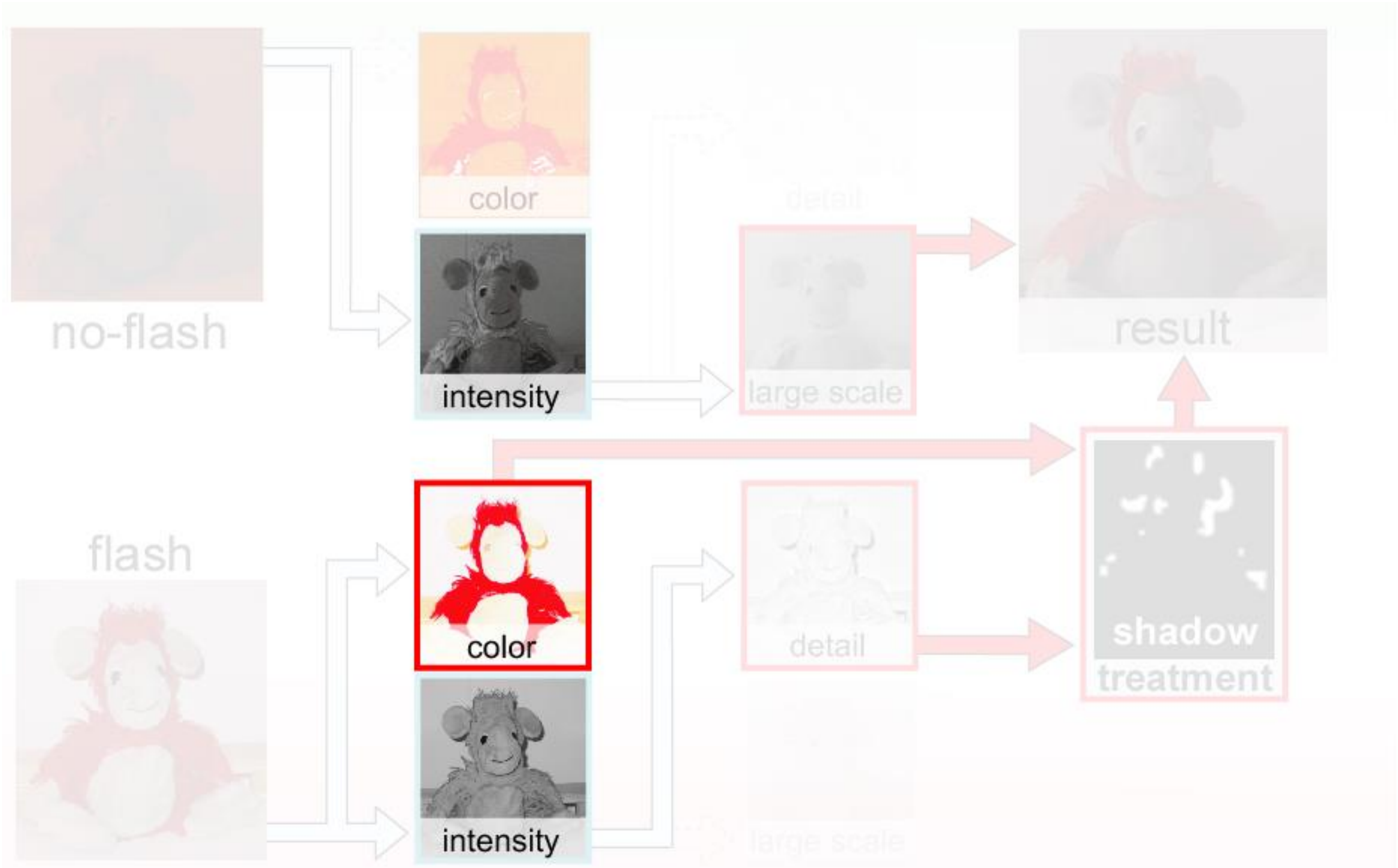
Our Approach

Registration



Our Approach

Decomposition



Decomposition

Color / Intensity:

original

=

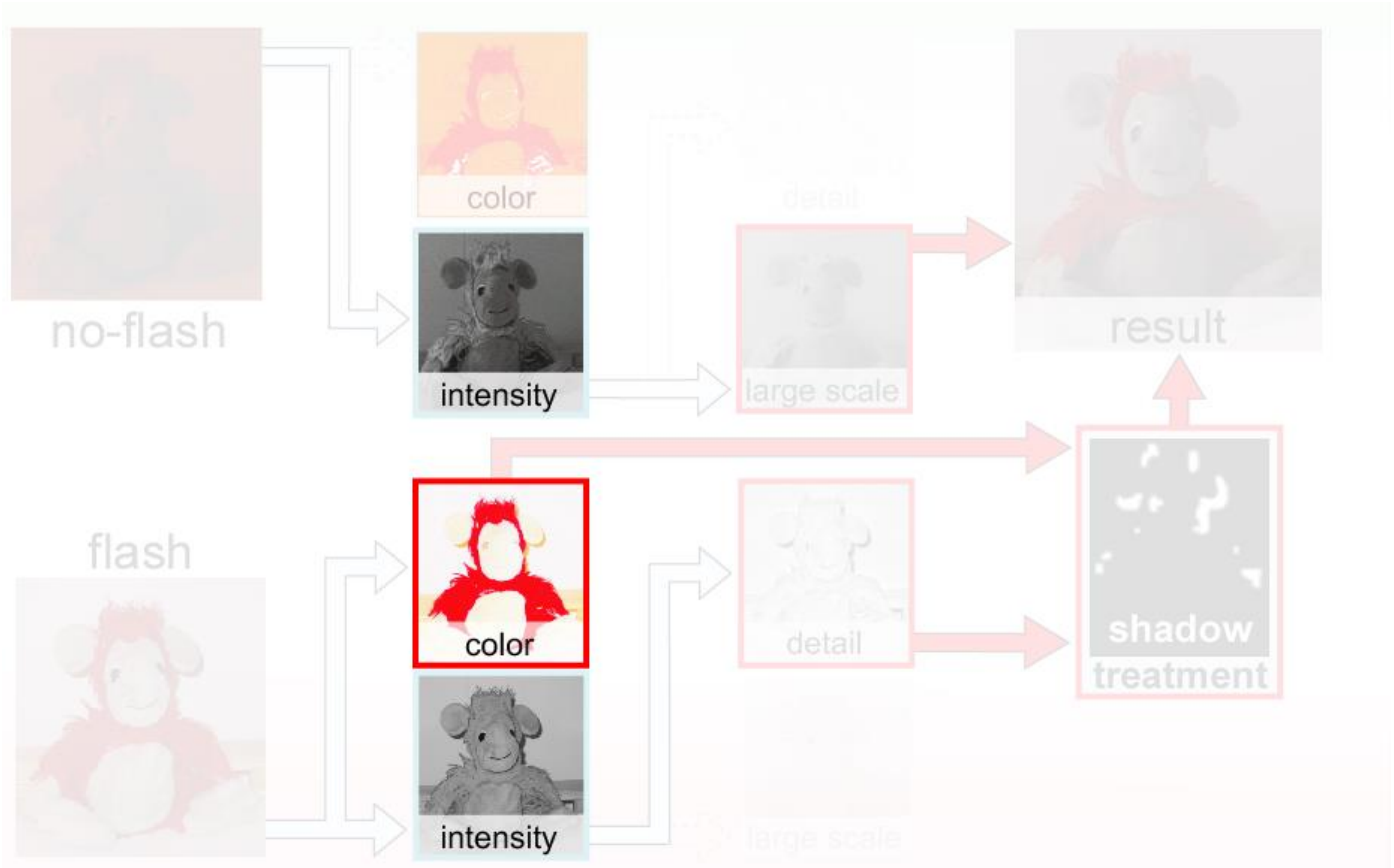
intensity

*

color

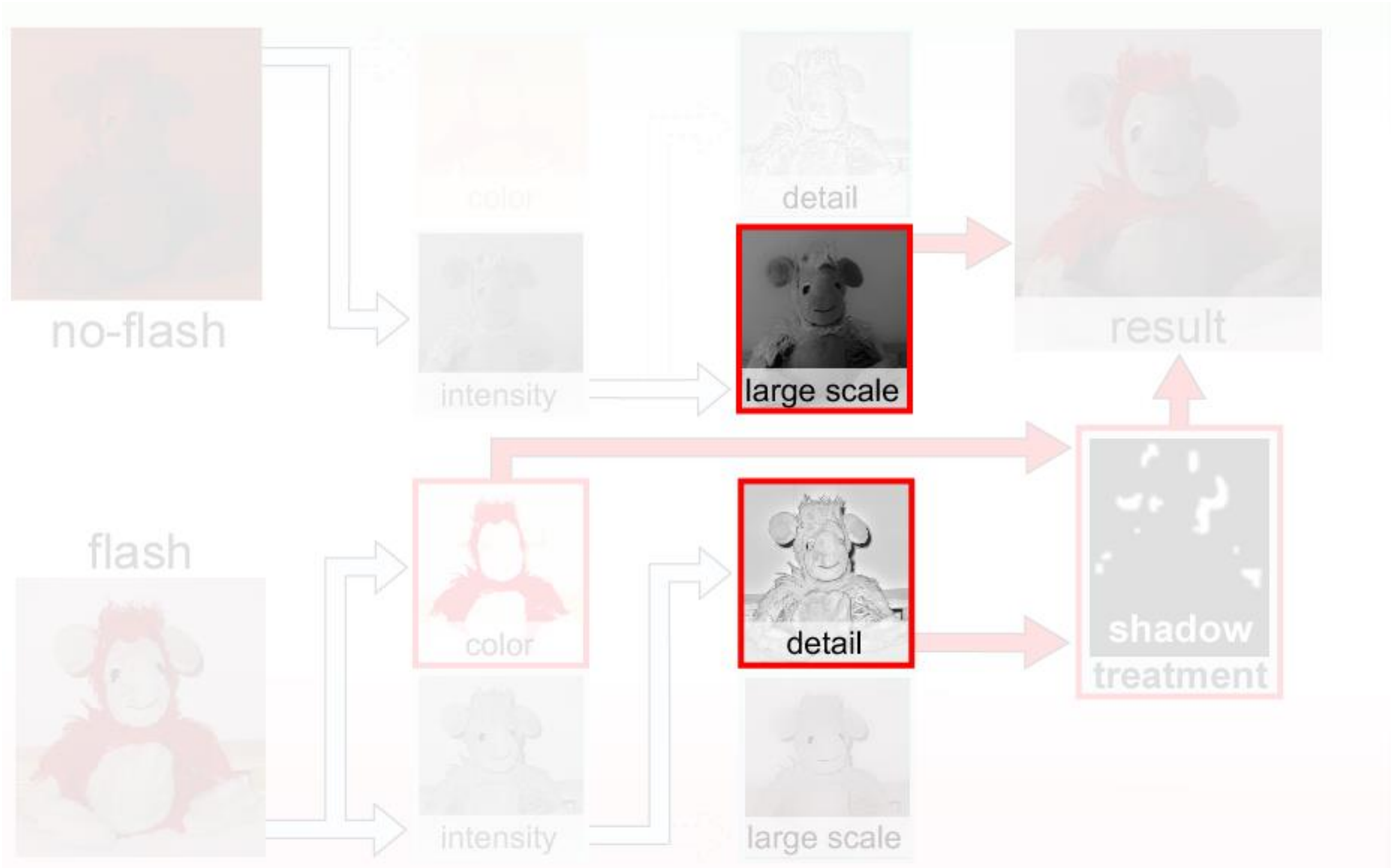
Our Approach

Decomposition



Our Approach

Decoupling



Decoupling

- Lighting : Large-scale variation
- Texture : Small-scale variation

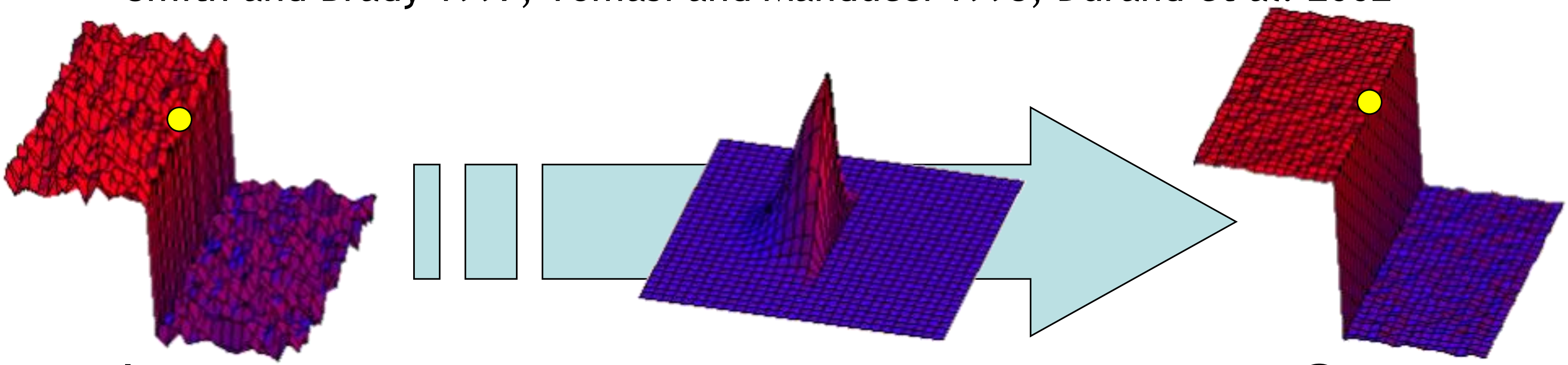
Lighting

Texture

Large-scale Layer

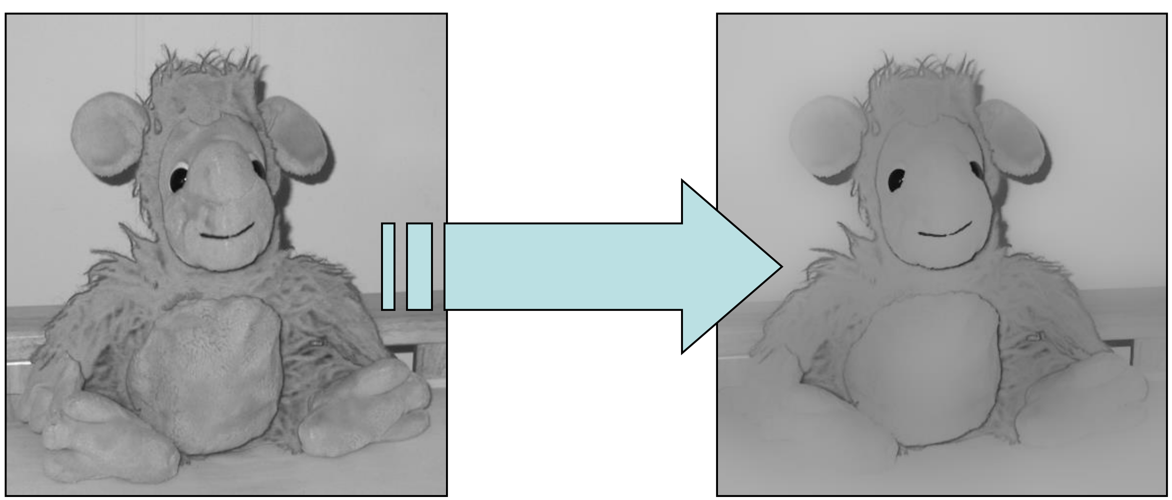
- Bilateral filter – edge preserving filter

Smith and Brady 1997; Tomasi and Manducci 1998; Durand et al. 2002



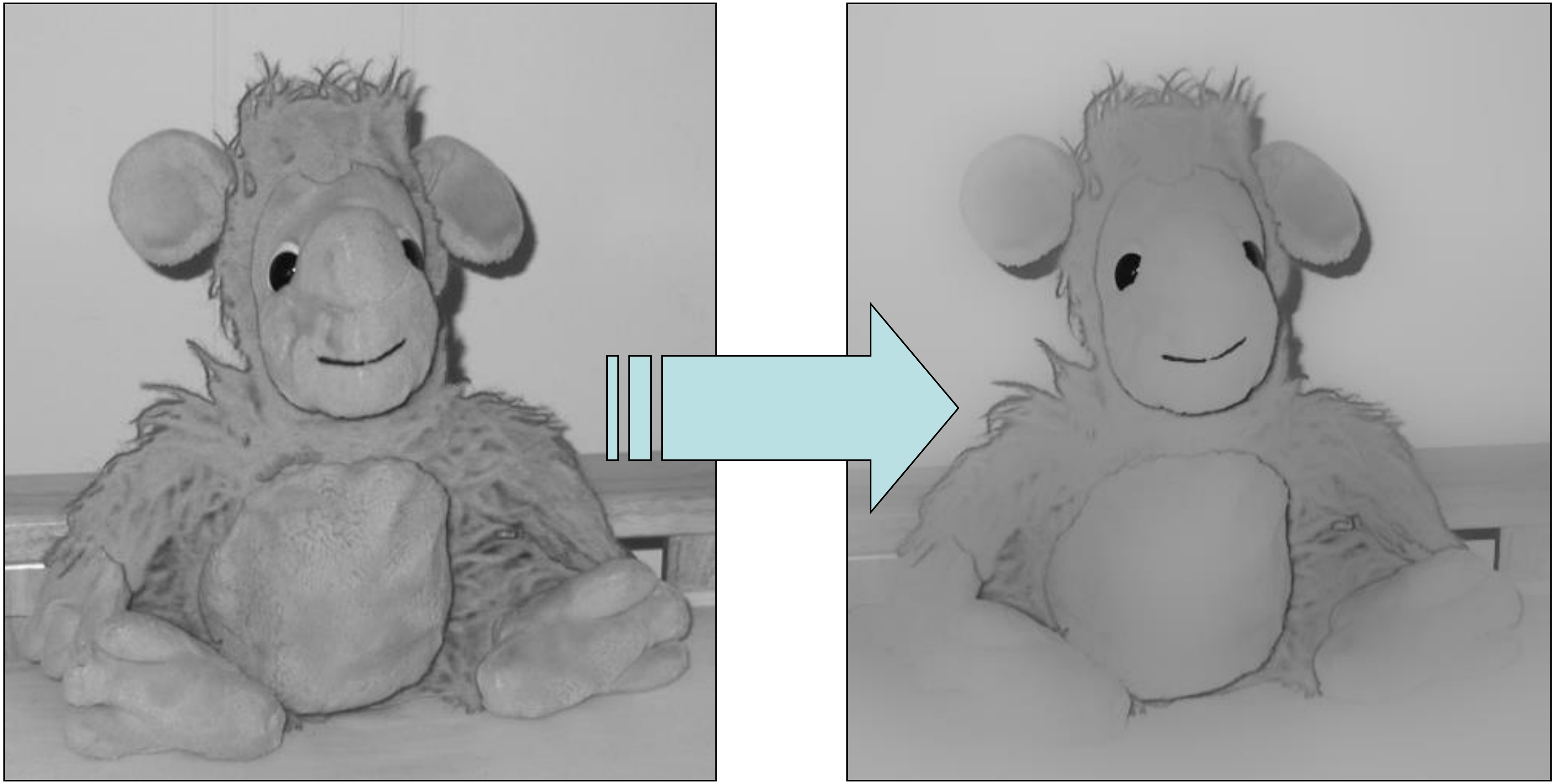
Input

Output



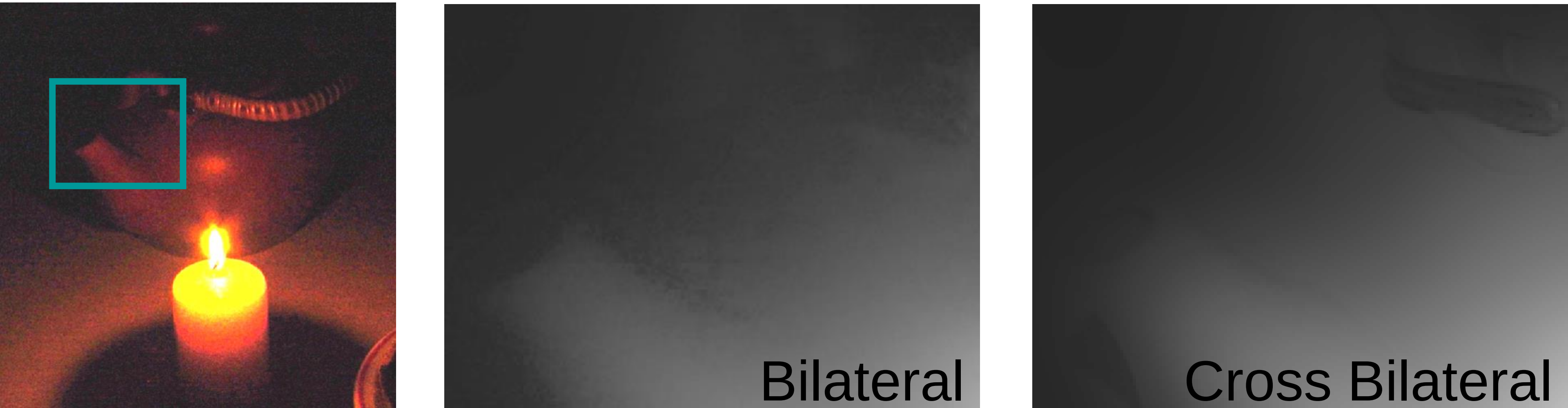
Large-scale Layer

- Bilateral filter

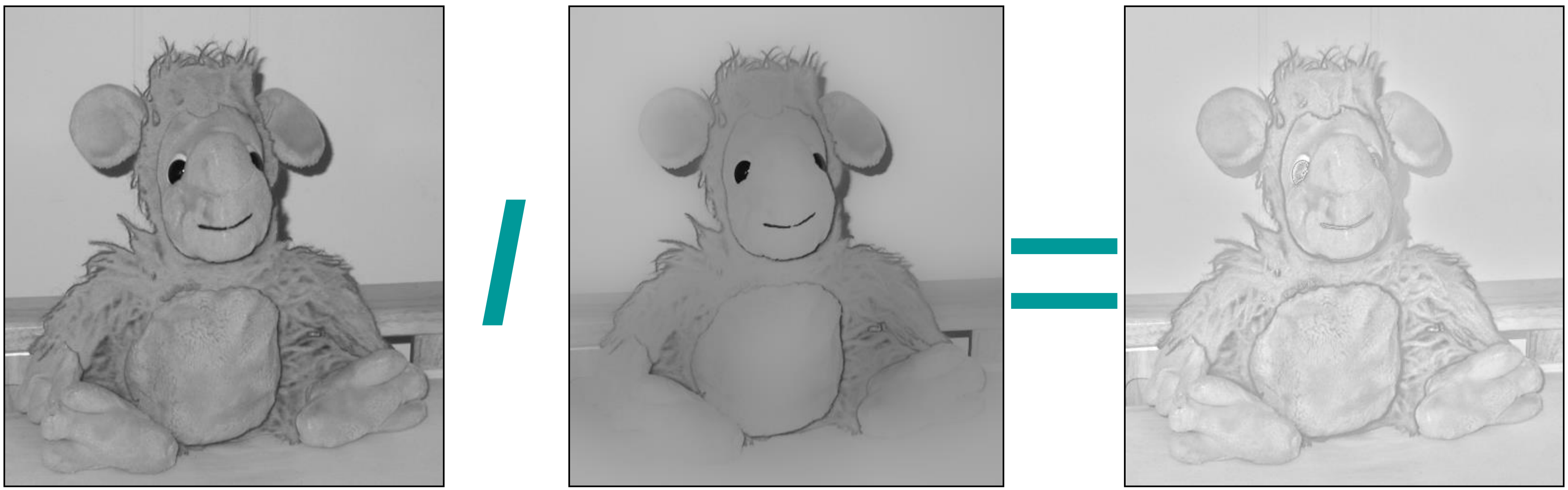


Cross Bilateral Filter

- Similar to joint bilateral filter by Petschnigg et al.
- When no-flash image is too noisy
- Borrow similarity from flash image
  - ➢ edge stopping from flash image



Detail Layer



Intensity / Large-scale = Detail

Recombination: Large scale * Detail = Intensity

Recombination

Large-scale
No-flash

*

Detail
Flash

=

Intensity
Result

Recombination: Large scale * Detail = Intensity

Recombination

shadows

Intensity Result

*

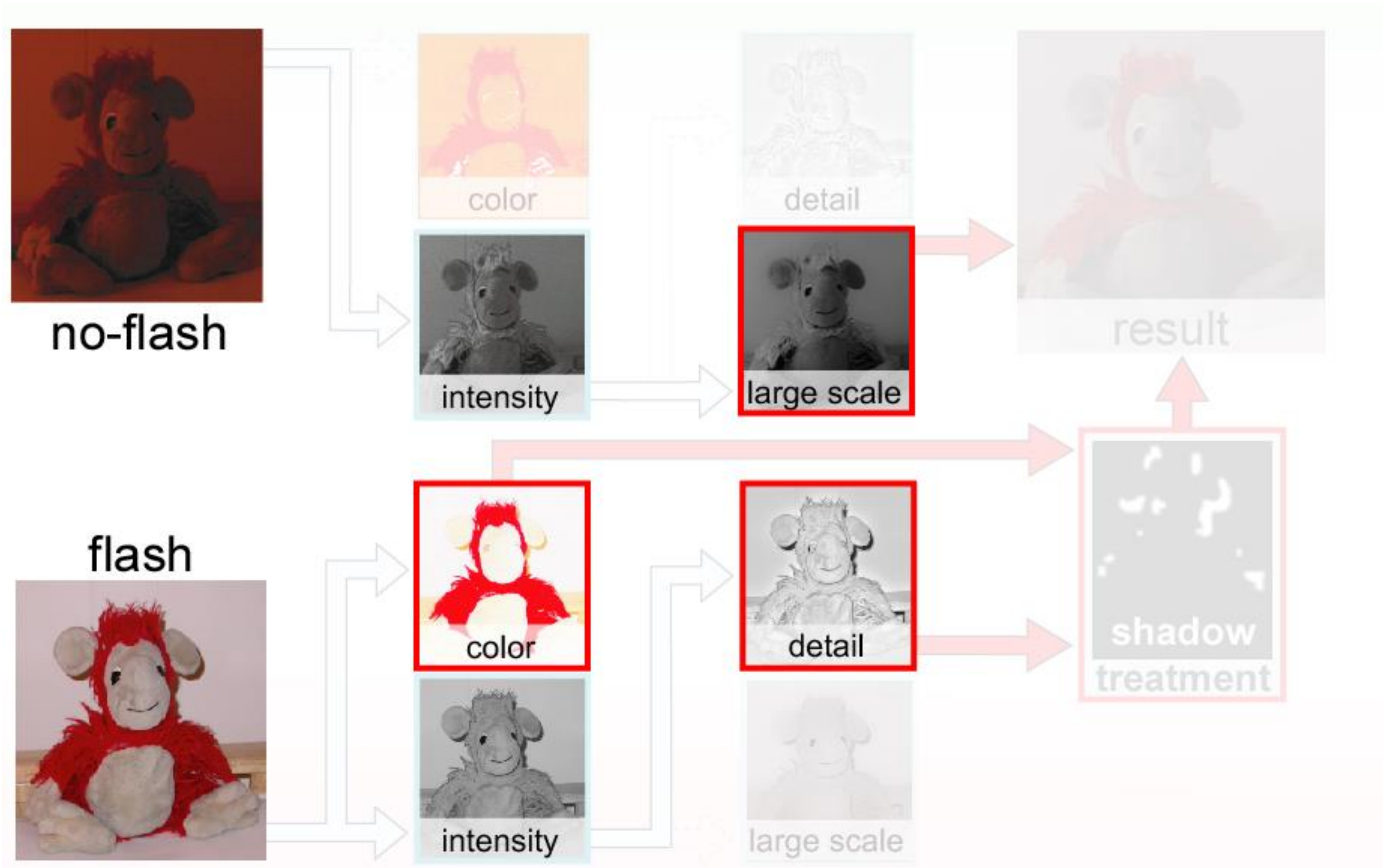
Color Flash

≈

Result

Recombination: Intensity * Color = Original

Our Approach



Results

No-flash

Flash