Digital Image Processing

Image Smoothing for Noise Reduction: Bilateral Filtering

Suyash P. Awate

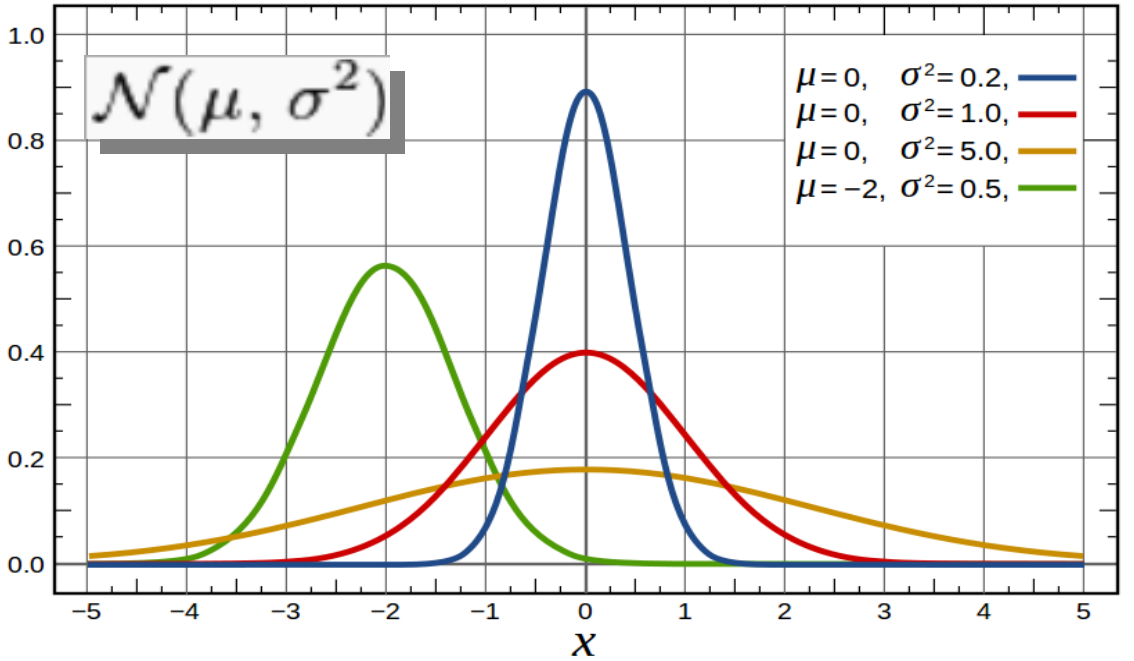
Noise

- A simple model
  - Additive independent-and-identically-distributed Gaussian noise
  - Corrupted image Y = uncorrupted image X + noise Z

$$Z_i \sim \mathcal{N}(0, N)$$
$$Y_i = X_i + Z_i$$

  - Normal PDF = Gaussian PDF

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



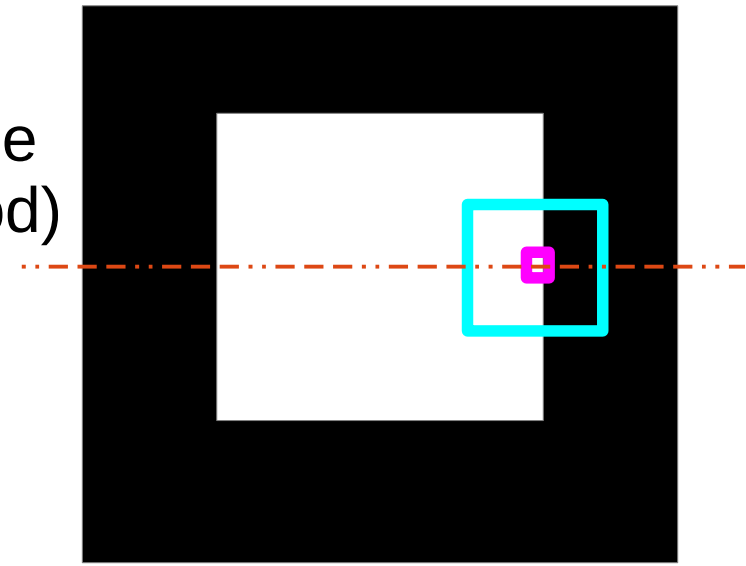
- In the real world, for some applications, noise can be difficult to model

Smoothing

- What is the problem with **weighted-mean** filters ?
  - Fail to preserve edges, while reducing noise
    - Oversmooth

Smoothing

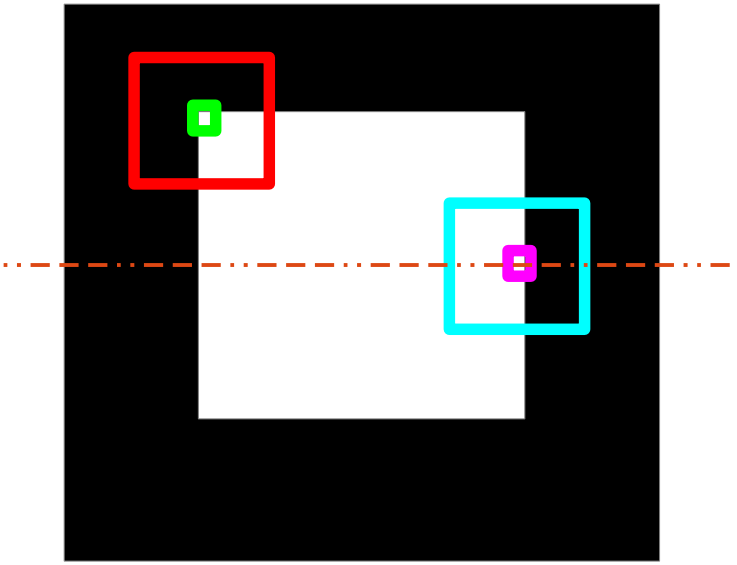
- What is the problem with **median** filter ?
  - Preserves edges in 1D images
    - Consider the scan line (1D) crossing the “magenta” pixel (with blue neighborhood)
      - Color of pixel = white
      - In neighborhood,
        number of white pixels
        > number of black pixels
      - Median = white



- Does it preserve all edges in 2D images ?

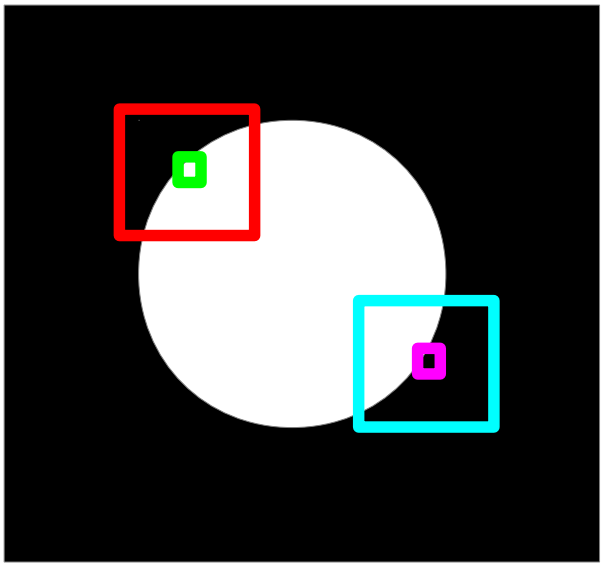
Smoothing

- What is the problem with **median** filter ?
  - Does it preserve all edges in 2D images ?
    - Consider the “green” pixel (with red neighborhood)
      - Color of pixel = white
      - In neighborhood, number of white pixels < number of black pixels
      - Median = black !
      - Edge NOT preserved



Smoothing

- What will happen with **repeated** application of:
  - Gaussian filter on a grayscale image ?
    - What will be the final image ?
  - Median filter on the square image in previous slide ?
    - What about the following image ?



Edge-Preserving Smoothing

- We desire a filter that:
  - 1) Reduces noise
  - 2) Preserves object edges within image
- Price we are willing to pay
  - Nonlinearity of the filtering
    - Filtered value is a nonlinear function of pixel intensities
    - Filter can't be represented as convolution with original image
    - May increase computational load

Edge-Preserving Smoothing

- General idea underlying any kind of smoothing
  - At pixel “p”, smoothed intensity
    = weighted average of all pixel intensities in image
  - Key questions
    - What are the weights ?
    - Are some weights zero ? = what neighborhood to choose ?
  - Examples
    - Mean filter
      - Weights $\propto$ 1 within neighborhood, 0 elsewhere
    - Weighted means filter
      - Weights $\propto$ Gaussian within neighborhood
    - Median filter
      - Weight = 1 for only that neighboring pixel whose value is the median

Edge-Preserving Smoothing

- Bilateral Filter
  - **2 key ideas** used **together**
  - At pixel location “p”,
    weight for another pixel “q” is based on :
  - (1) **Spatial distance** between “q” and “p”
    - Larger distance → lower weight
    - e.g., Gaussian over distance
      - Same as Gaussian smoothing filter
  - (2) **Dissimilarity between intensities** “I(q)” and “I(p)”
    - Larger dissimilarity → lower weight
    - e.g., Gaussian over intensity

Edge-Preserving Smoothing

- Bilateral Filter
  - Updated pixel intensity =

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

where

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

I^{filtered} is the filtered image;

I is the original input image to be filtered;

x are the coordinates of the current pixel to be filtered;

Ω is the window centered in x;

f_r is the range kernel for smoothing differences in intensities.

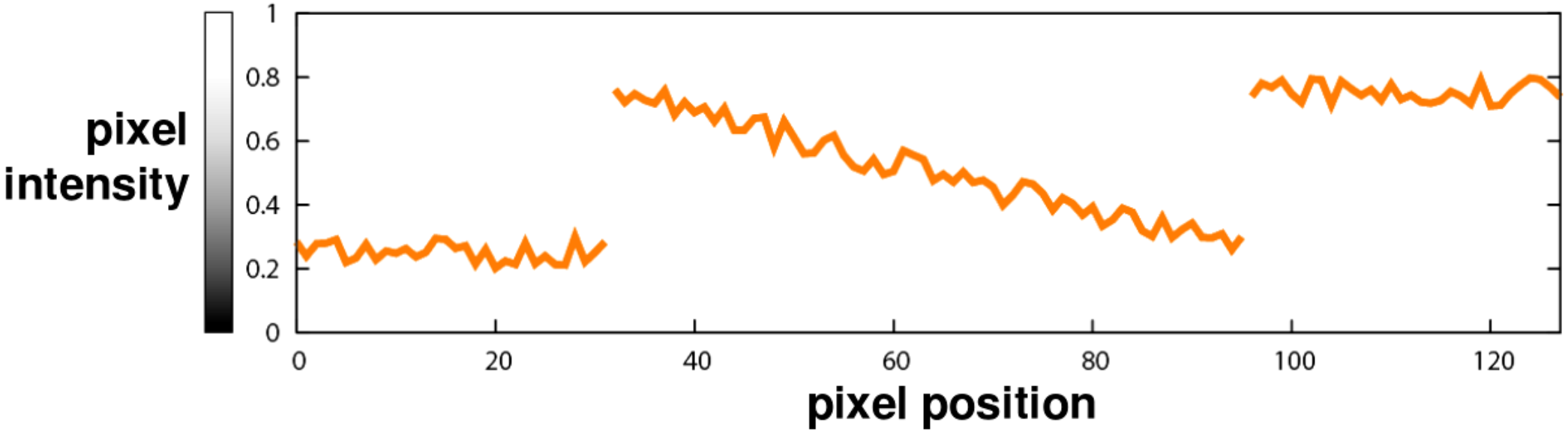
g_s is the spatial kernel for smoothing differences in coordinates

Edge-Preserving Smoothing

- Bilateral Filter
  - Functions f(.) and g(.) are typically Gaussian
  - Why can't this be written as convolution ?
    - Not a linear function of intensities
    - This defines a different weighting mask for every pixel

Edge-Preserving Smoothing

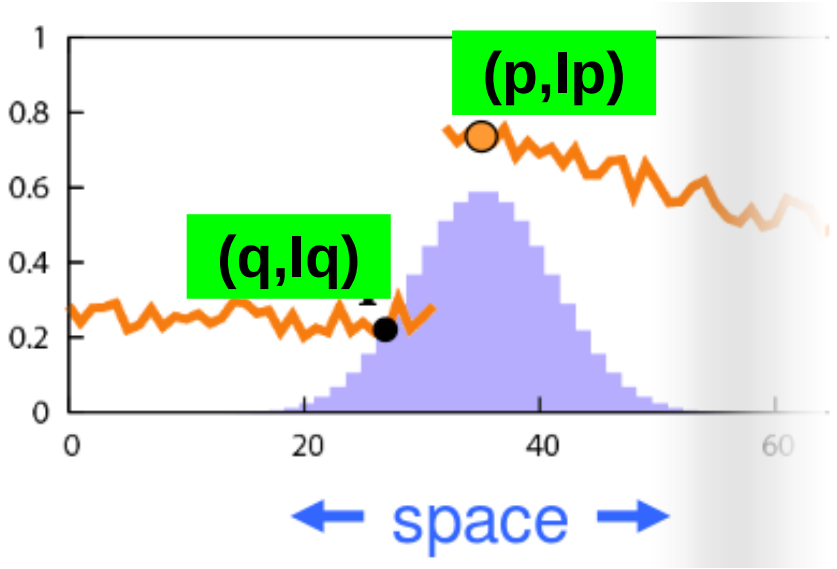
- Bilateral Filter
  - What does Gaussian filter do to a noisy edge ?
    - 1D image



cs.brown.edu/courses/cs129/lectures/bf_course_Brown_Oct2012.pdf

Edge-Preserving Smoothing

- Bilateral Filter
  - Effect of **Gaussian** filter on weights at pixel “p”

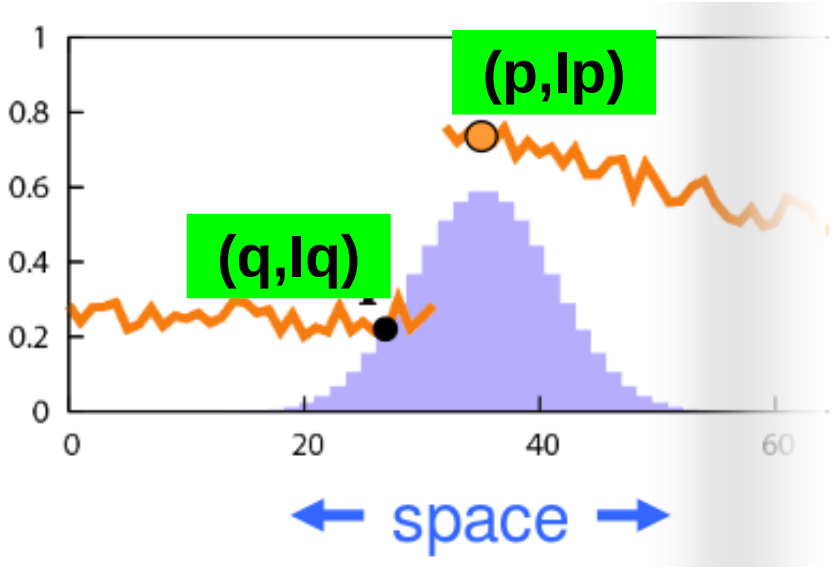


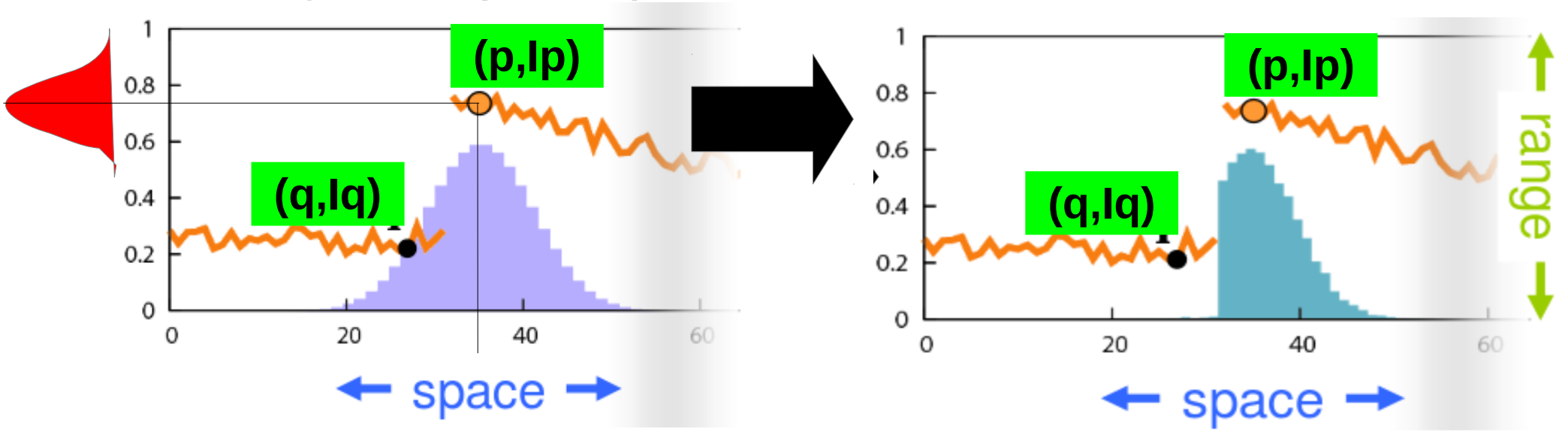
$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Edge-Preserving Smoothing

- Bilateral Filter
  - Effect of **Gaussian** filter on weights at pixel “p”
  - Effect of **bilateral** filter on weights at pixel “p”

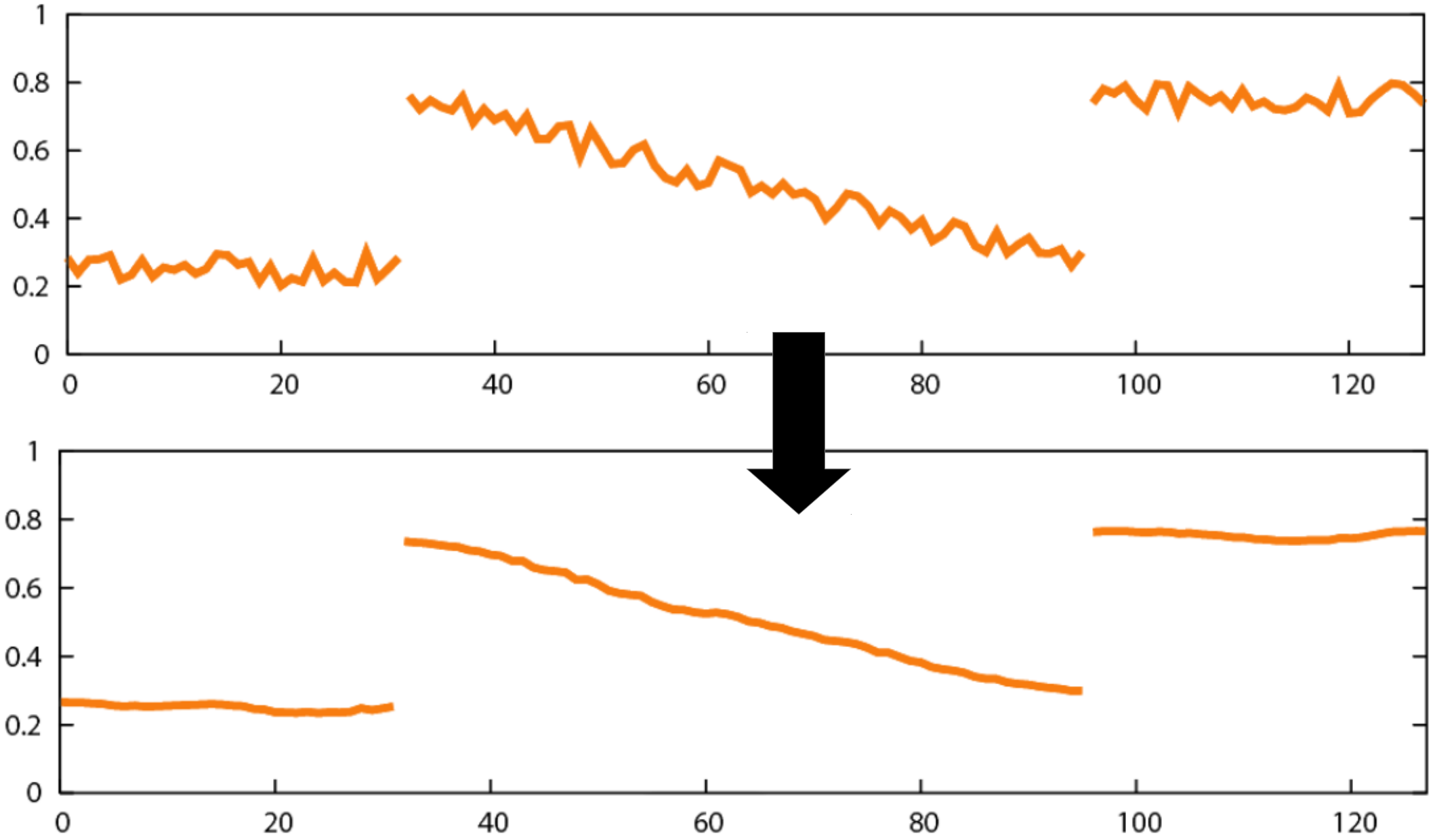




$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

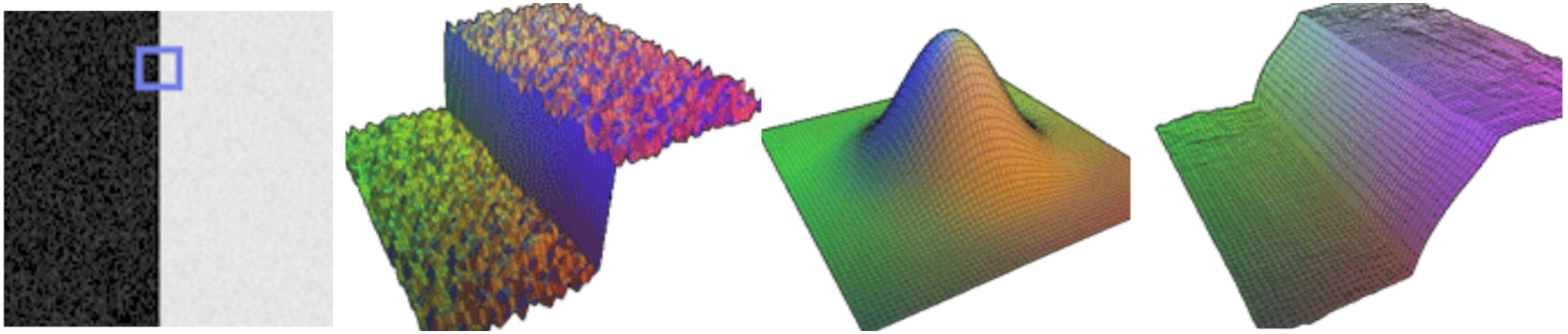
Edge-Preserving Smoothing

- Bilateral Filter

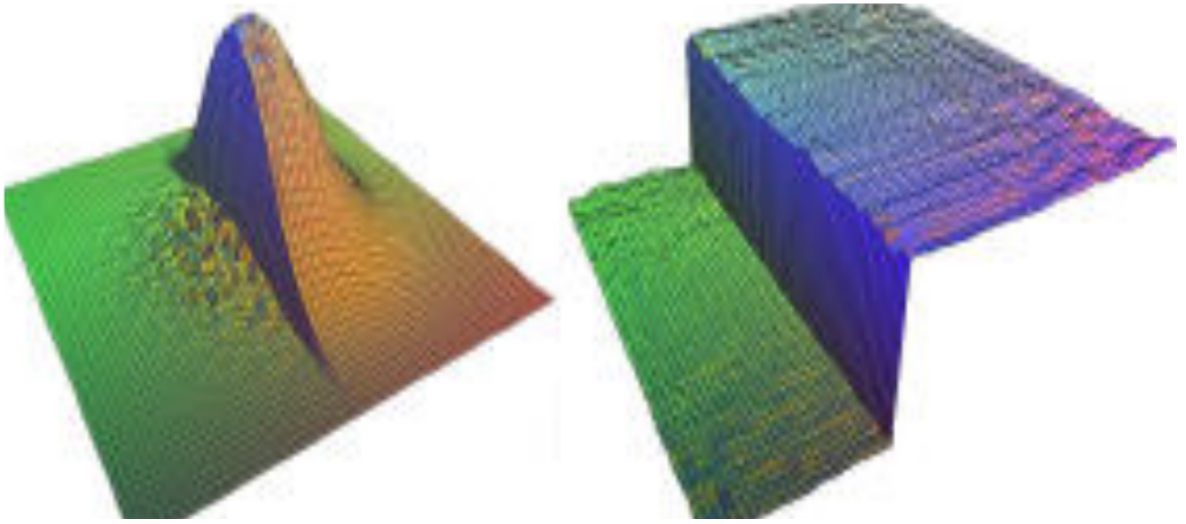


Edge-Preserving Smoothing

- Bilateral Filter
  - What does Gaussian filter do to a noisy edge in 2D ?

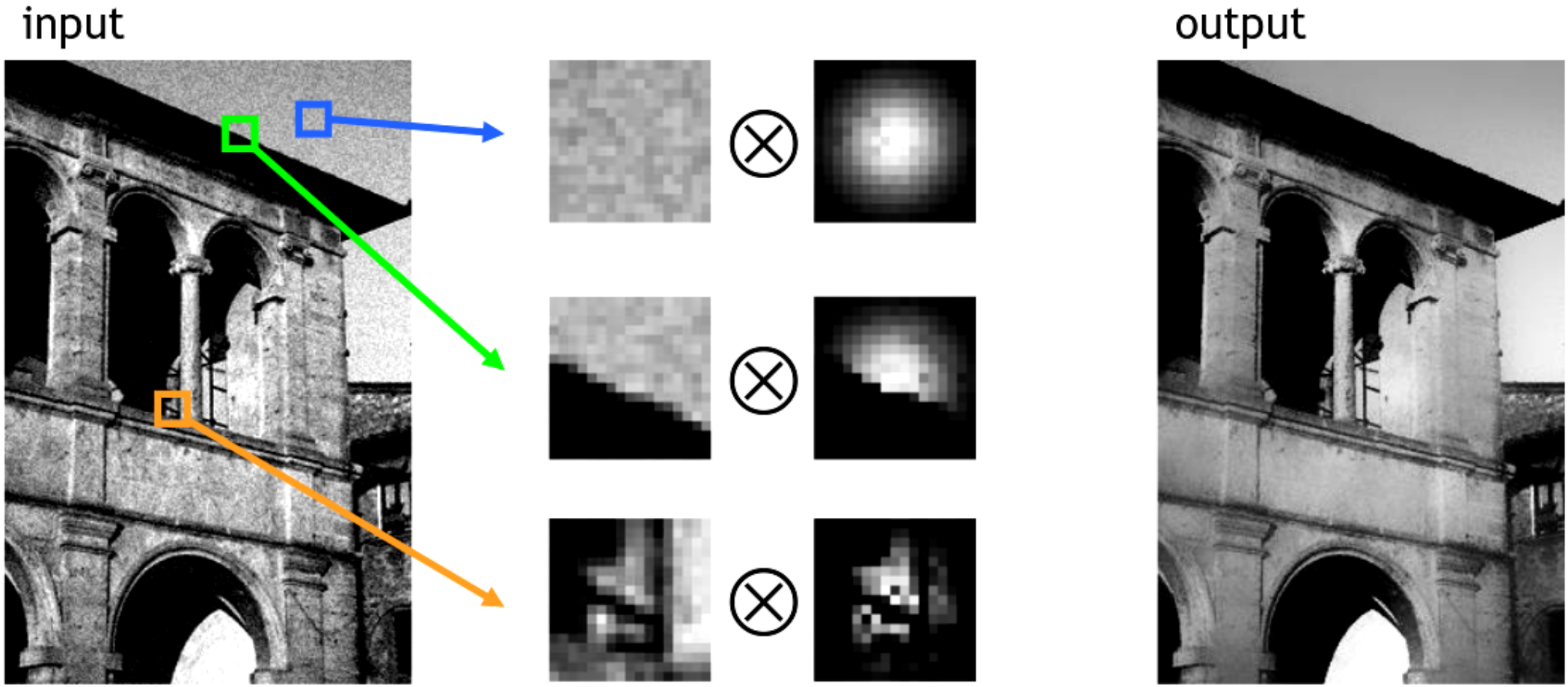


  - What does bilateral filter do to a noisy edge in 2D ?
    - Mask for bright pixel =



Edge-Preserving Smoothing

- Bilateral Filter
  - On real images, weight masks can be quite complex

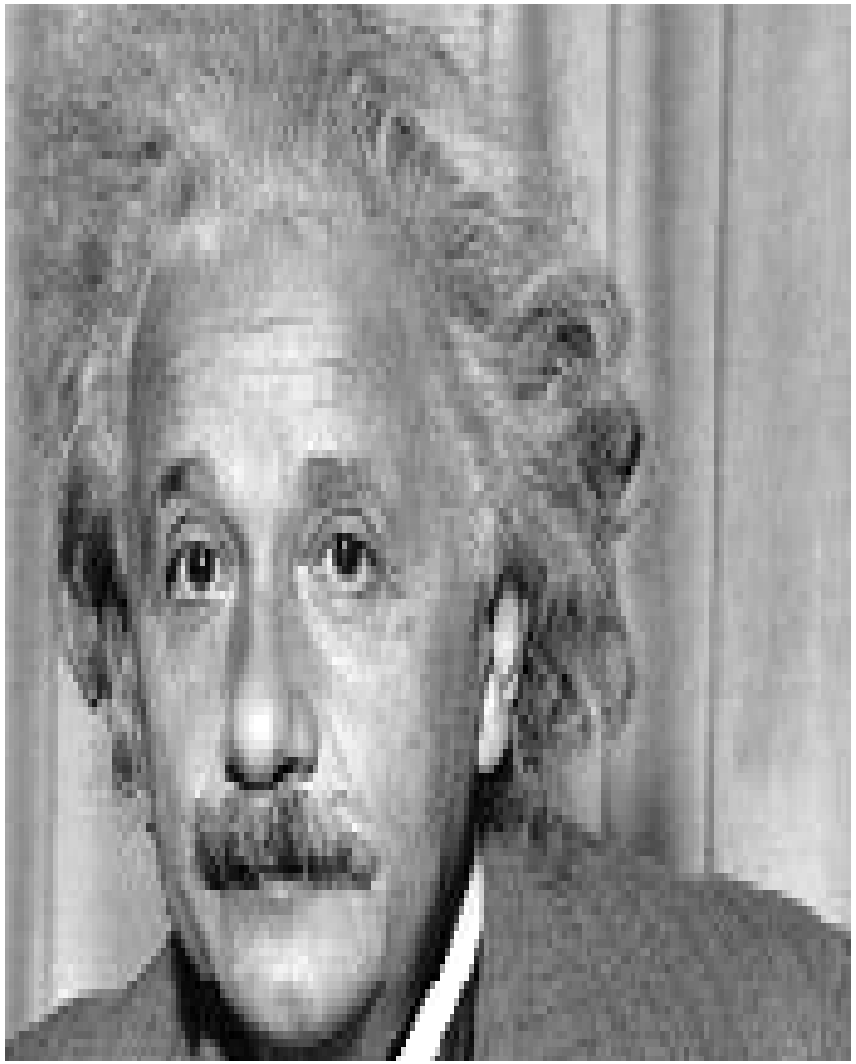


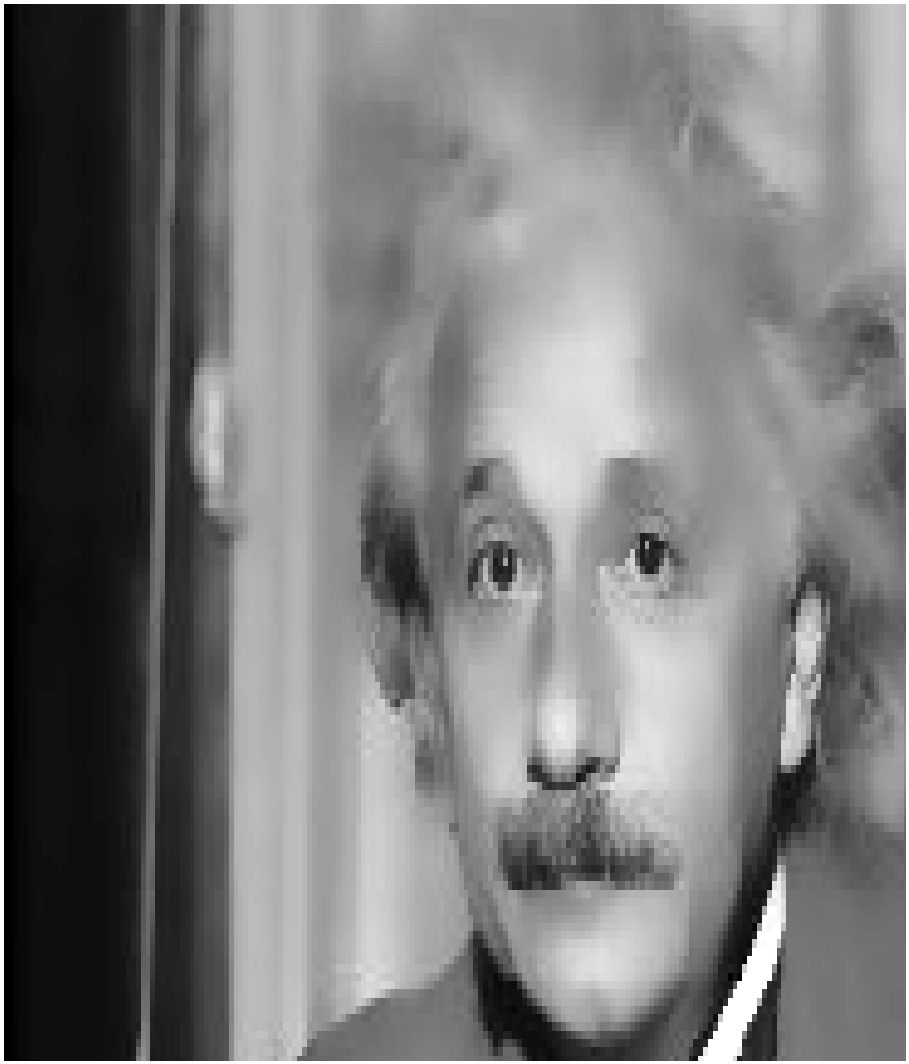
Edge-Preserving Smoothing

- Bilateral Filter

Edge-Preserving Smoothing

- Bilateral Filter





Edge-Preserving Smoothing

- Bilateral Filter

Edge-Preserving Smoothing

- Bilateral Filter

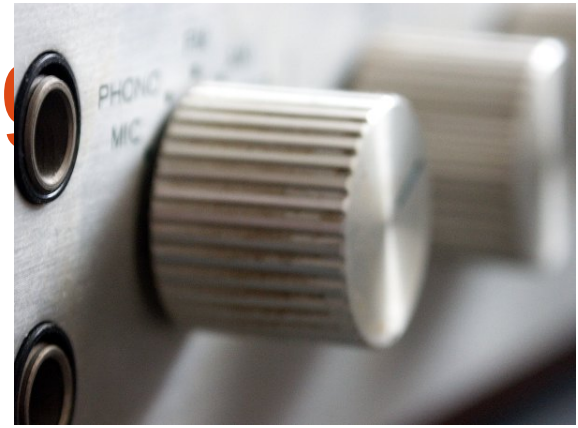
$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

  - What happens when,
    for Gaussian f(.) on **intensities**, variance → ∞ ?
  - What happens when,
    for Gaussian f(.) on intensities, variance → 0 ?
  - What happens when,
    for Gaussian g(.) on **space**, variance → ∞ ?
  - What happens when,
    for Gaussian g(.) on space, variance → 0 ?

Edge-Preserving Smoothing

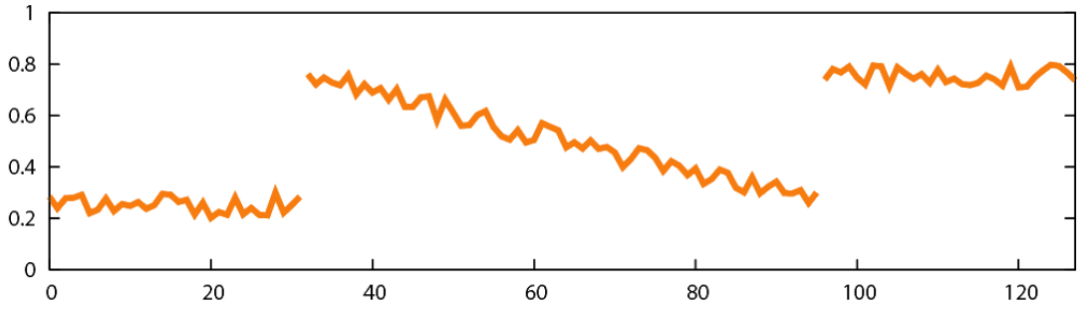
- Bilateral Filter
  - What happens when,
    for Gaussian f(.) on **intensities**, variance $\rightarrow \infty$?
    - Bilateral filter $\rightarrow$ Gaussian smoothing
  - What happens when,
    for Gaussian f(.) on intensities, variance $\rightarrow 0$?
    - Filter doesn't modify image
  - What happens when,
    for Gaussian g(.) on **space**, variance $\rightarrow \infty$?
    - Filter averages over all pixels. Neighborhood = entire image.
  - What happens when,
    for Gaussian g(.) on space, variance $\rightarrow 0$?
    - Filter doesn't modify image

Edge-Preserving Smoothing



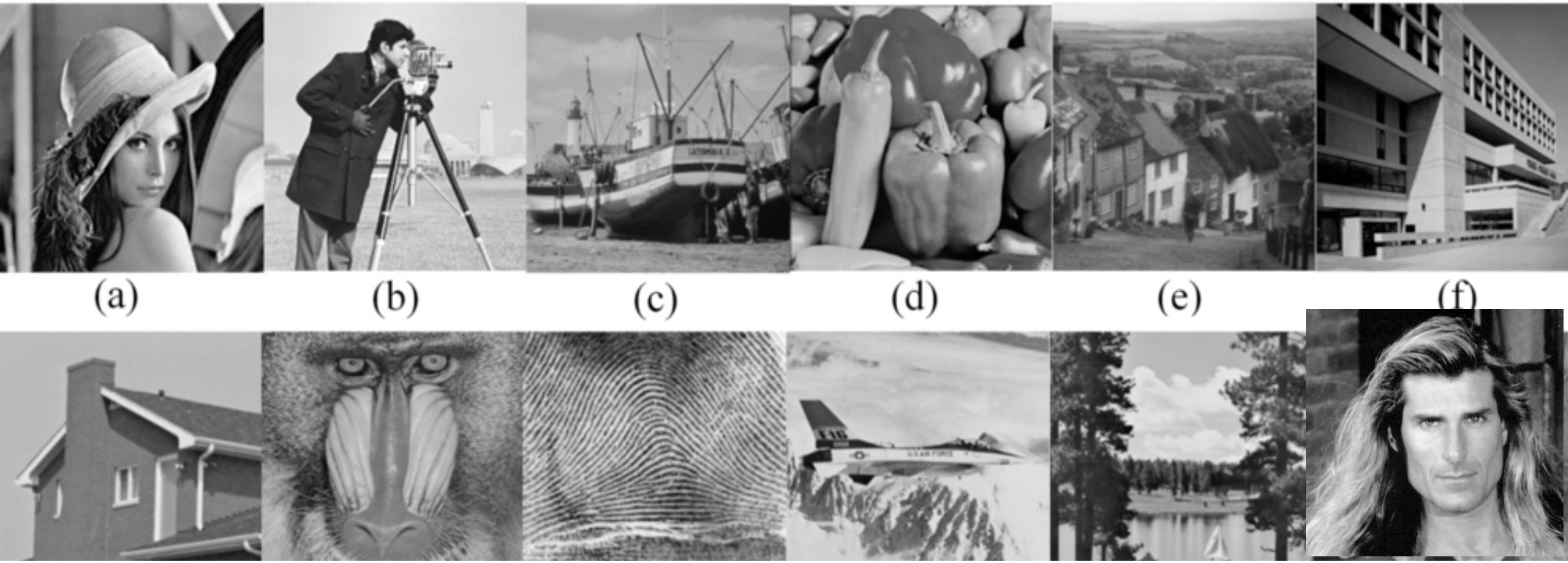
- Bilateral Filter
  - Parameter tuning ?
    - Standard deviation (SD) of Gaussian f(.) on intensities
      - Edges with contrast <= SD are blurred.
        We want these to correspond to noise !
      - Edges with contrast >> SD are better preserved.
        We want these to correspond to signal !
    - SD of Gaussian g(.) on space
      - Larger SD can allow more averaging → more noise reduction
      - Larger SD can allow averaging intensities from different objects
        - May be undesirable

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$



Standard Test Images

- http://sipi.usc.edu/database
- http://www.imageprocessingplace.com/root_files_V3/image_databases.htm
- https://en.wikipedia.org/wiki/Standard_test_image
- And more …



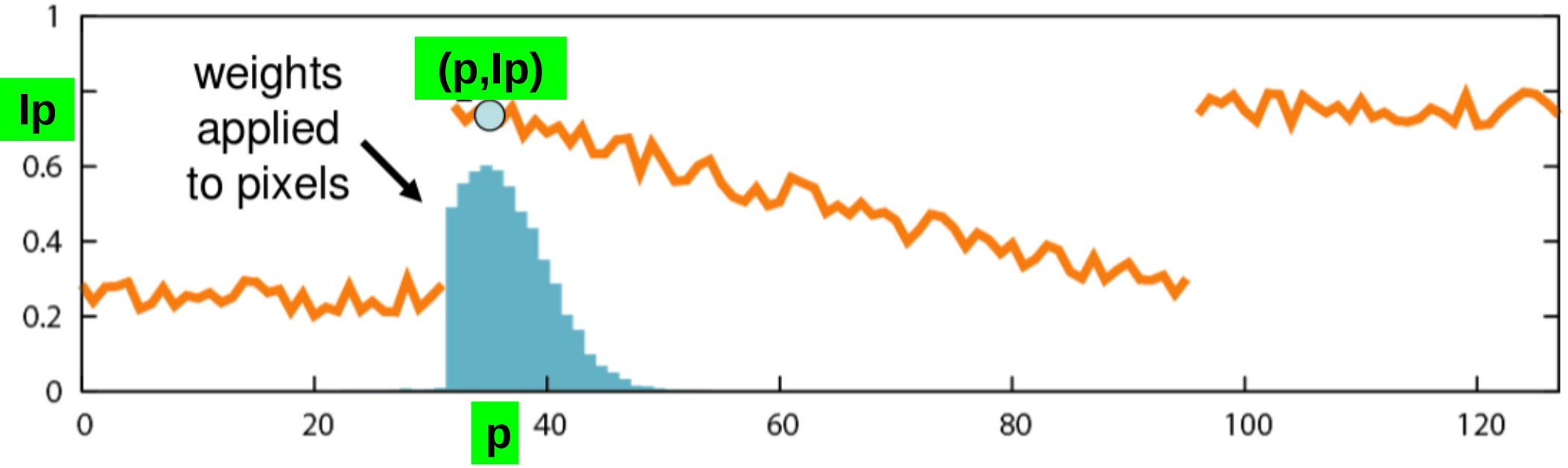
Edge-Preserving Smoothing

- Bilateral Filter
  - Is nonlinear → NO convolution → no separability → slow
  - But can we still compute it fast ?
  - Yes !
  - How ?
    - Rewrite it using linear filtering in a different space (NOT image-coordinate space)
    - Fast and (sufficiently) accurate algorithm

      [cs.brown.edu/courses/cs129/lectures/bf_course_Brown_Oct2012.pdf]

Edge-Preserving Smoothing

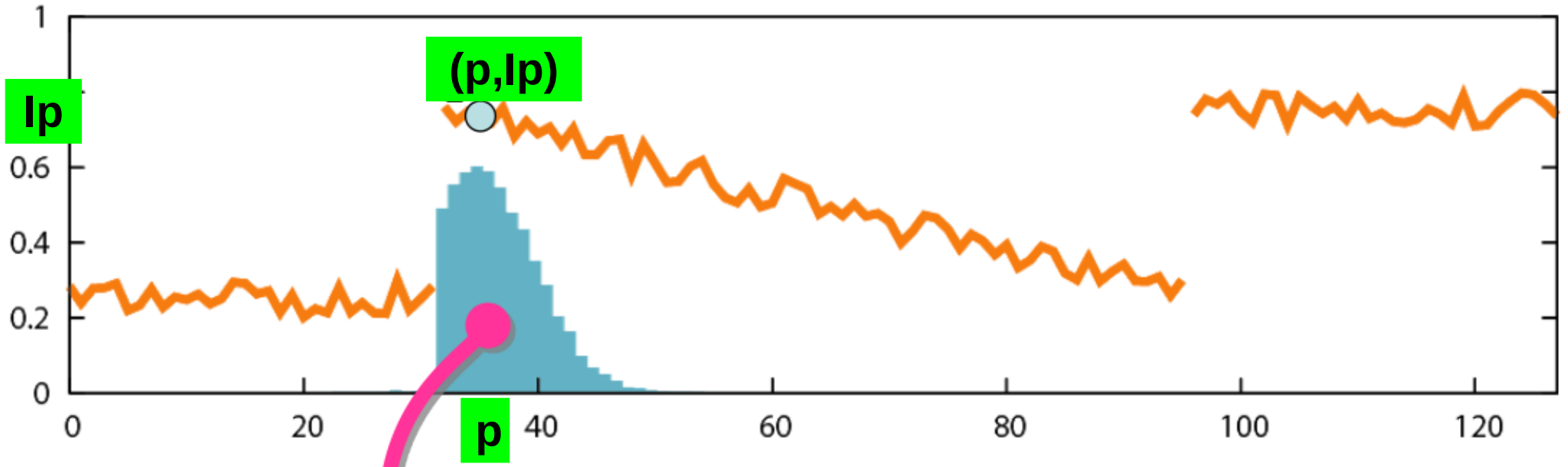
- Bilateral Filter
  - Recap
  - Processing at pixel “p”



cs.brown.edu/courses/cs129/lectures/bf_course_Brown_Oct2012.pdf

Edge-Preserving Smoothing

- Bilateral Filter



sum of weights

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

Edge-Preserving Smoothing

- Bilateral Filter
  - Two key equations for processing pixel “p”

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

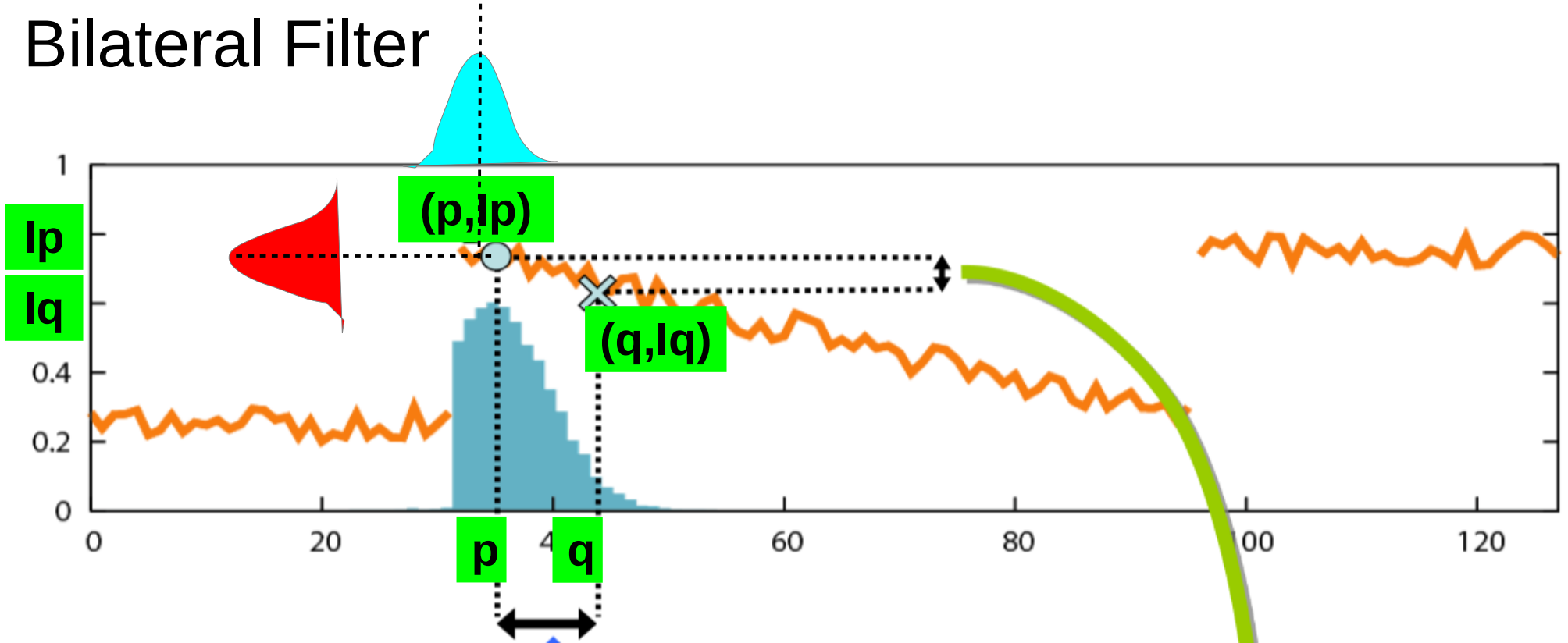
$$W_{\mathbf{p}}^{\mathrm{bf}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)$$

  - In 1st equation, multiply both sides by normalization factor

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

Edge-Preserving Smoothing

- Bilateral Filter

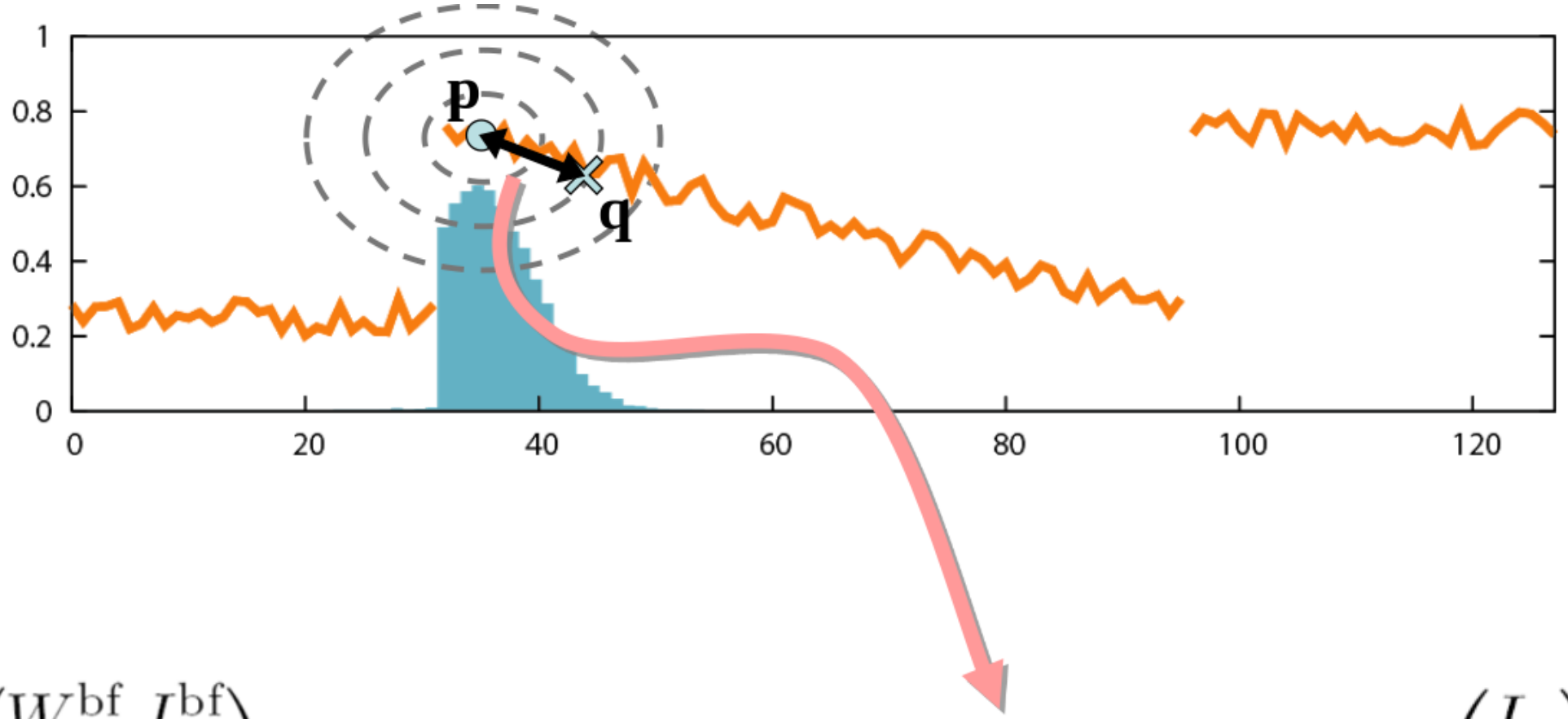


$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}} I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{\mathbf{q}\in\mathcal{S}} \underbrace{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}_{\text{space}} \underbrace{G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|}_{\text{range}} \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

2D Gaussian

Edge-Preserving Smoothing

- Bilateral Filter

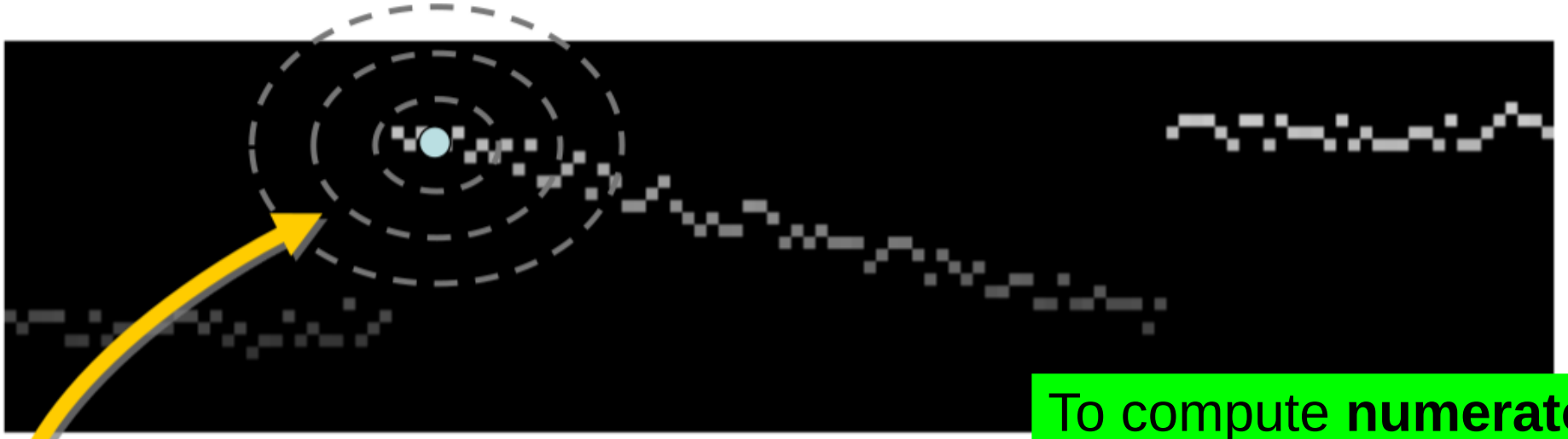


$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}} I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|) \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

space x range

Edge-Preserving Smoothing

- Bilateral Filter : **Convolution in <location,intensity>**



sum all values

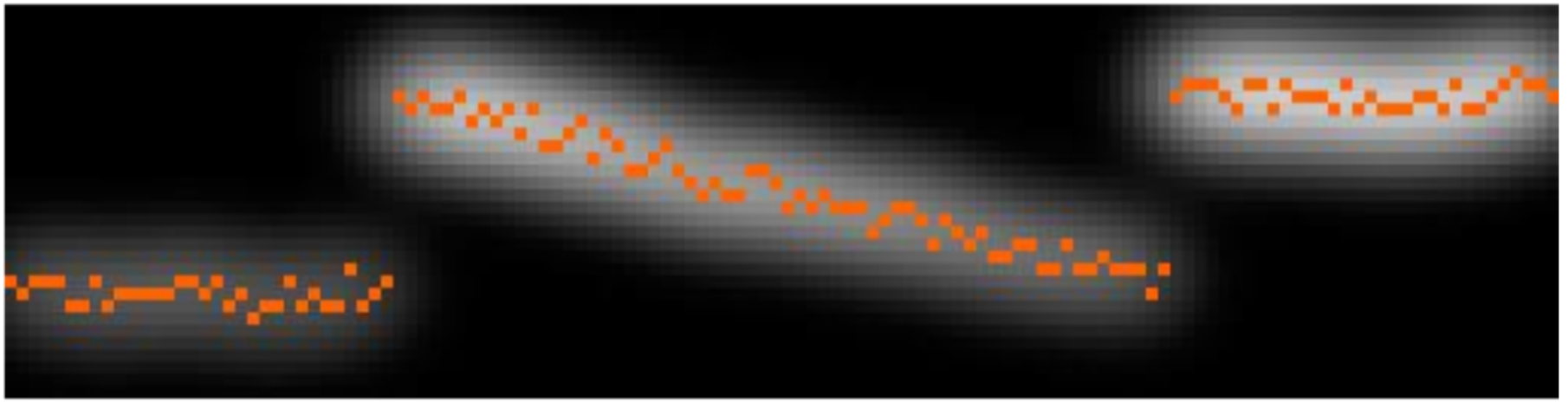
To compute **numerator**.
Construct a **joint space**.
1) At “location” (p,Ip),
assign intensity Ip
2) Assign 0 elsewhere

$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}} I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} \end{pmatrix} = \sum\sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

Edge-Preserving Smoothing

- Bilateral Filter

Get numerator values by picking them from "locations" (q,Iq) in convolved image



result of the convolution

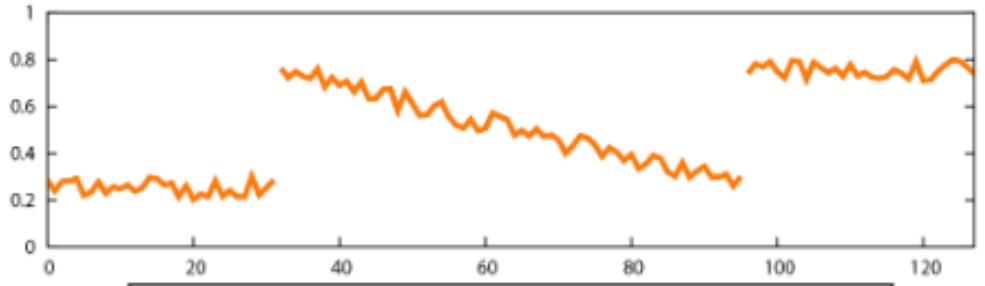
$$\begin{pmatrix} W_{\mathbf{p}}^{\mathrm{bf}} & I_{\mathbf{p}}^{\mathrm{bf}} \\ W_{\mathbf{p}}^{\mathrm{bf}} & \end{pmatrix} = \sum\sum_{(\mathbf{q},\zeta)\in\mathcal{S}\times\mathcal{R}} \text{space-range Gaussian} \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix}$$

Edge-Preserving Smoothing

- Bilateral Filter
  - How do we compute values in the denominator ?

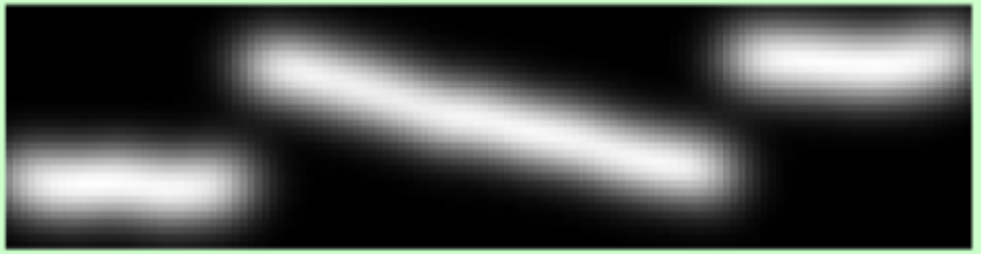
Edge-Preserving Smoothing

- Bilateral Filter: **Get numerator, denominator values**



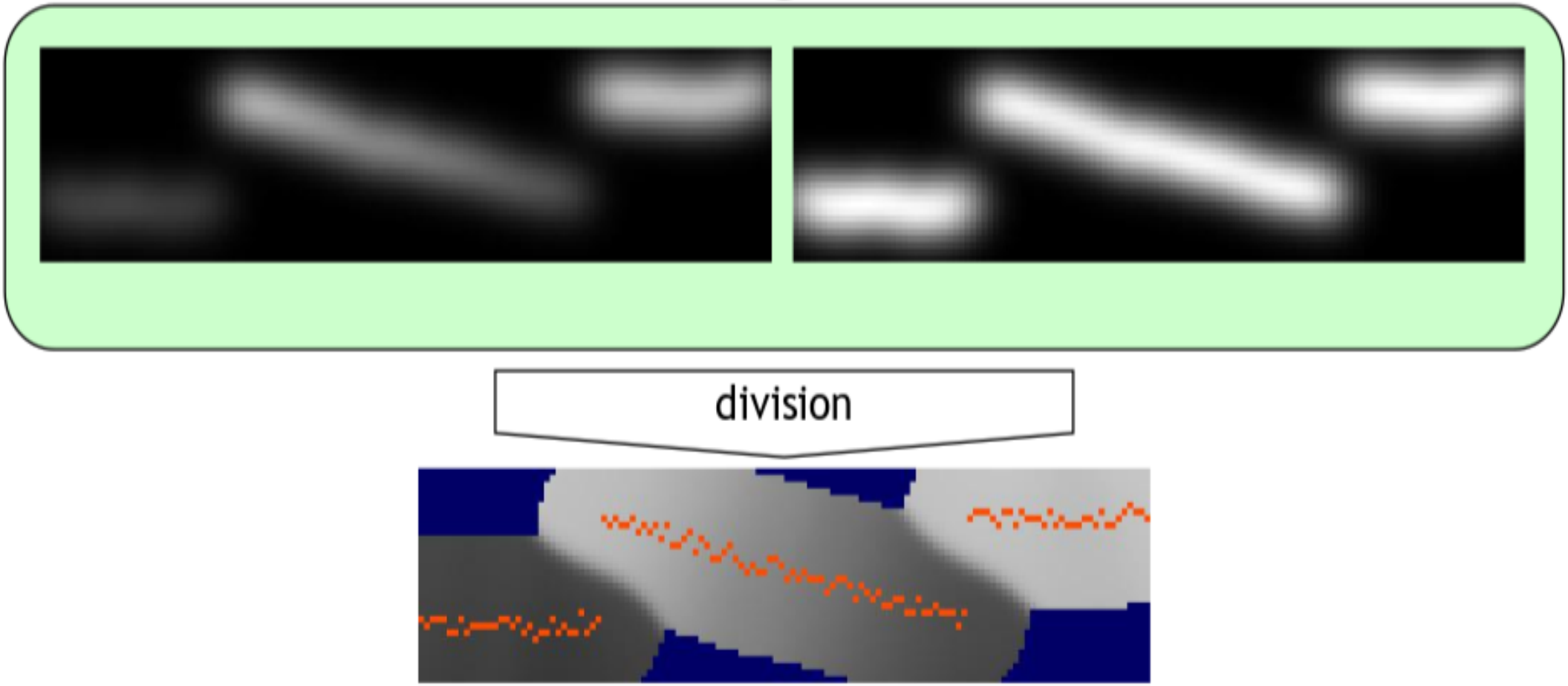
higher dimensional functions

Gaussian convolution



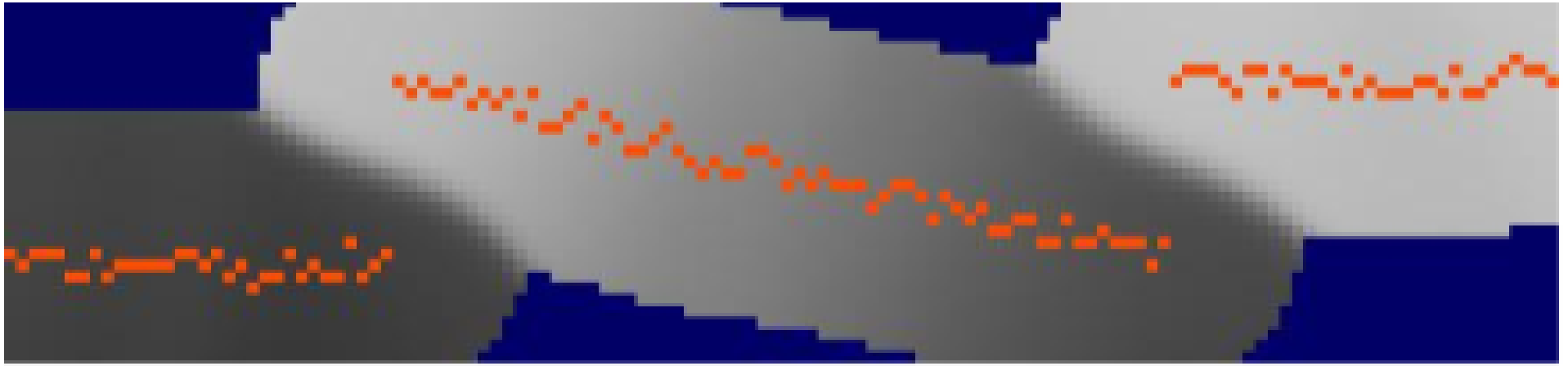
Edge-Preserving Smoothing

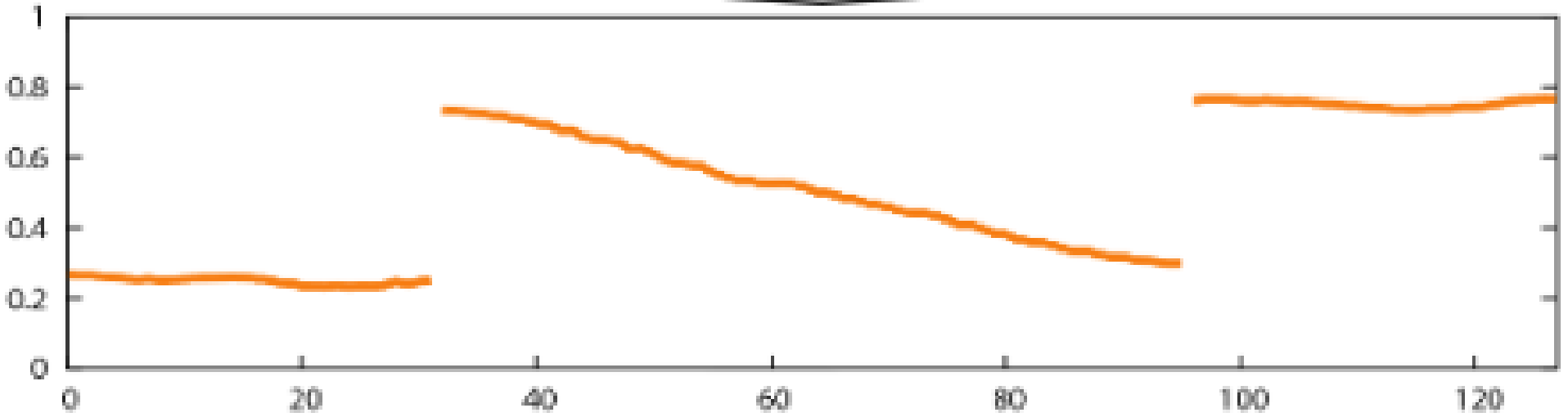
- Bilateral Filter: **Divide numerator by denominator**
  - Consider values only at “locations” (p,Ip) in numerator image and denominator image



Edge-Preserving Smoothing

- Bilateral Filter: **Update image to get filtered result**
  - At pixel “p”, filtered value is the division at “location” (p,Ip)



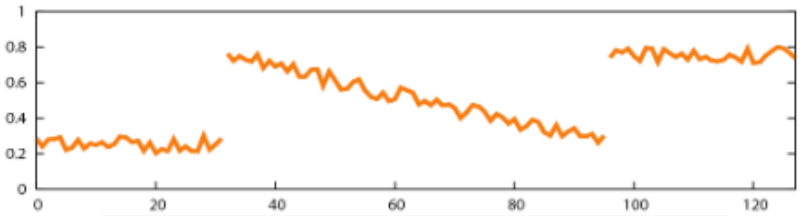


Edge-Preserving Smoothing

- Bilateral Filter
  - What did we do ?

    (1) Two convolutions in a higher-dimensional space <p,Ip>
    - Linear operation
    - Linear time complexity

    (2) Pixel-by-pixel division between 2 images
    - Nonlinear operation
    - Simple, fast, O(# pixels in image)
  - Exact formulation (analytically)
  - Why is this faster than standard algorithm ?
    - Lets see next ...

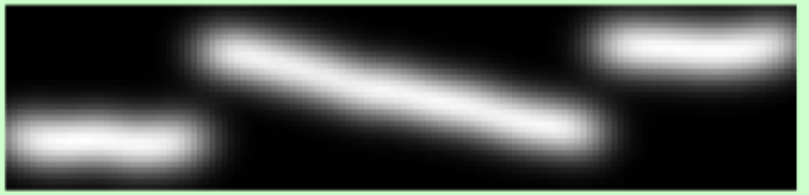


higher dimensional functions

Recap:

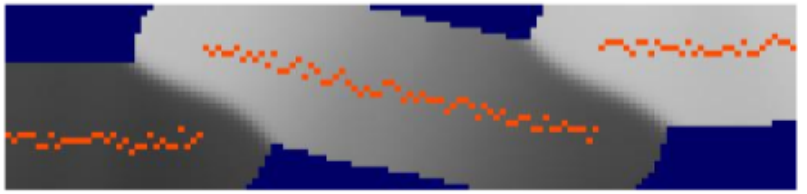
- simple operations
- complex space

Gaussian convolution

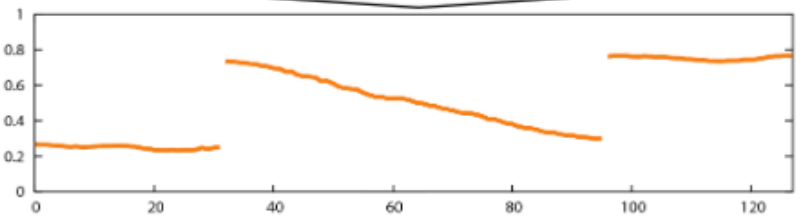


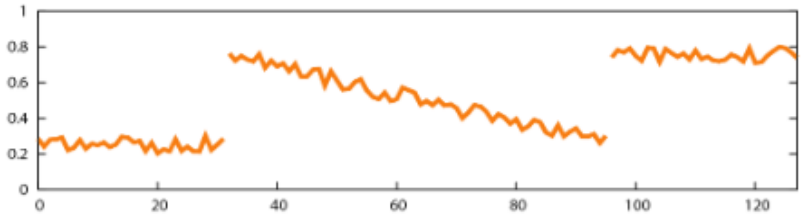
—

division



slicing





higher dimensional functions

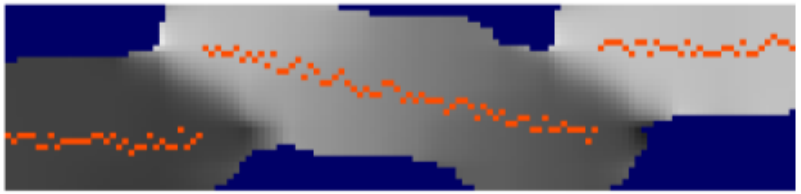
Strategy: downsampled convolution

DOWNSAMPLE **Averaging** in square with one **corner** at **pixel**

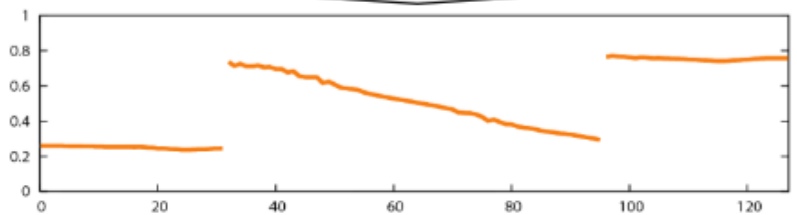
Gaussian convolution

UPSAMPLE

division



slicing

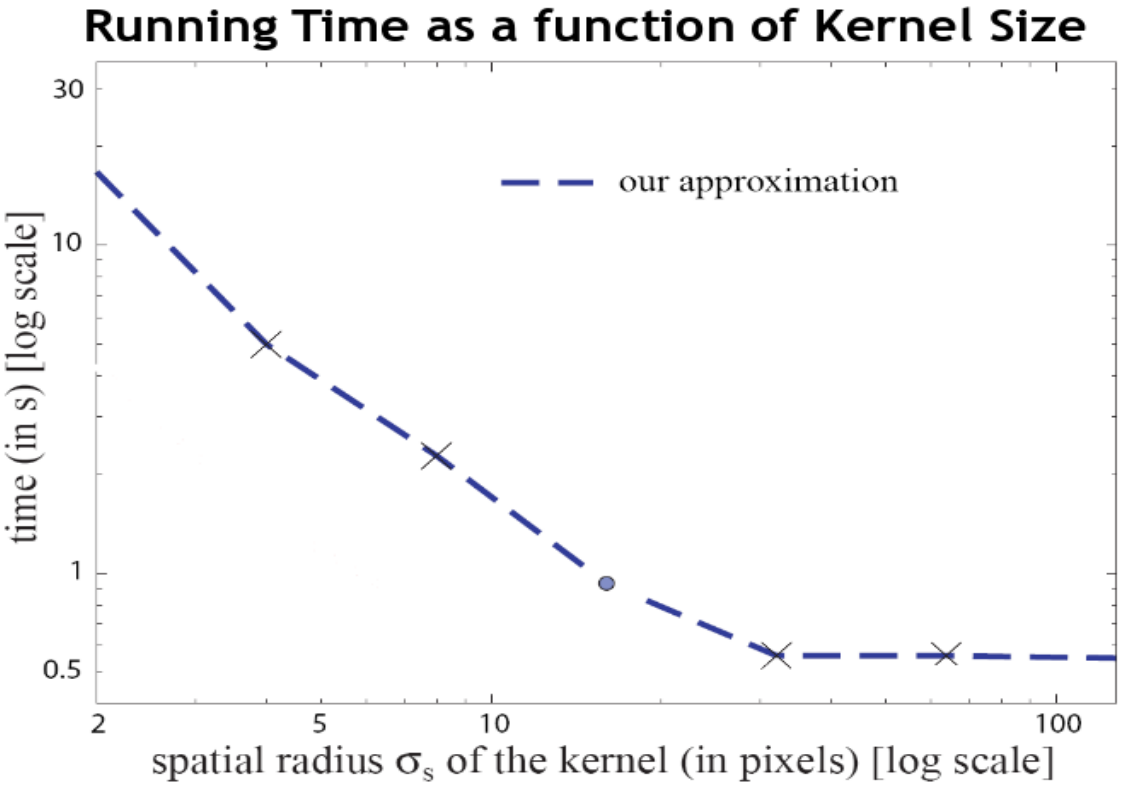


Edge-Preserving Smoothing

- Bilateral Filter
  - Why did downsampling NOT change results much ?
    - Convolution blurs anyway
    - Downsampling must involve some “**smoothing**” / **averaging**
      - NOT selecting every n-th pixel !
    - But that averaging can be done in **MI operations** only
      - Not (MI * box-filter-size) operations; No convolution !
  - Will this downsampling work for a general image ?
    - Yes
    - To understand why, we need concepts from:
      - Fourier analysis
      - Sampling theorem
      - Smoothing as low-pass filtering

Edge-Preserving Smoothing

- Bilateral Filter
  - Effect of downsampling
    - Faster algorithm ← less data to process
    - Error ← approximation
  - Larger kernel → Can afford more downsampling
    - So faster algorithm !
  - Typical trade off
    - Speed versus accuracy
  - Evaluate



Edge-Preserving Smoothing

- Bilateral Filter
  - Evaluate running time
  - 1200 x 600 image with reasonable parameter settings
    - Original algorithm → 10 minutes
    - New algorithm → Few seconds

1200 × 1600

  - Algorithmic complexity (M*N image, P*Q mask)
    - Original algorithm : MN PQ 3 (find exp weights; multiply twice)
    - New algorithm :
      - Downsample image + mask: M N I → M/a N/b I/c, P Q J → P/a Q/b J/c
      - Separable convolutions in joint space: 2 (M/a N/b I/c) (P/a + Q/b + J/c)
        - If a = b = c, then operations = (M N I) (P + Q + J) (2 / a^4)
      - Upsampling: MN 4 (bilinear interpolation at p-lp locations only)
      - Division and update : MN

Edge-Preserving Smoothing

- Bilateral Filter
  - Evaluate accuracy

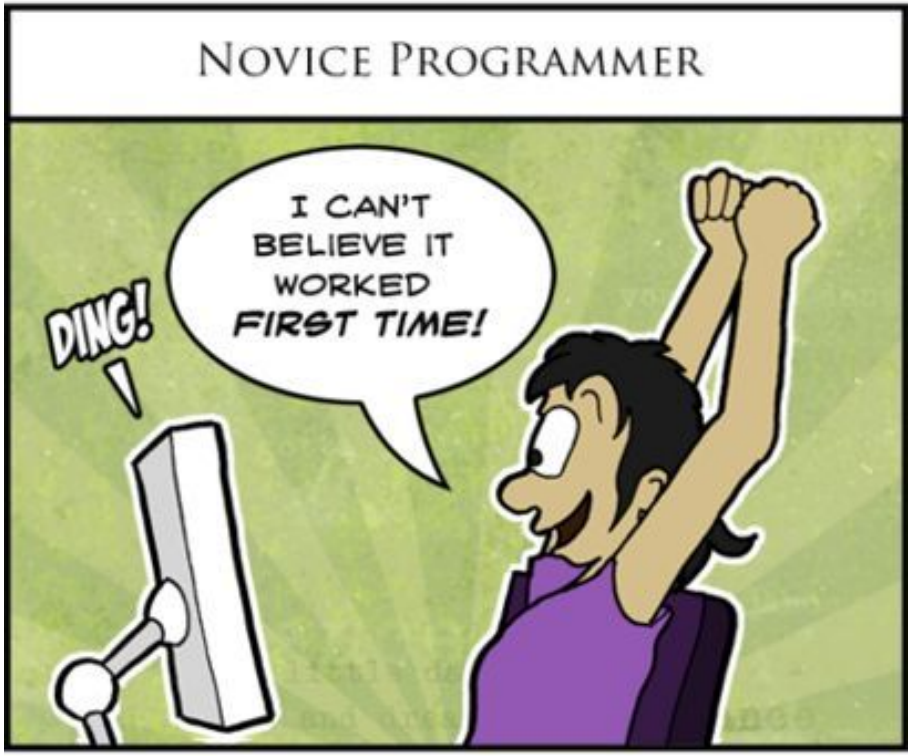
1200 × 1600

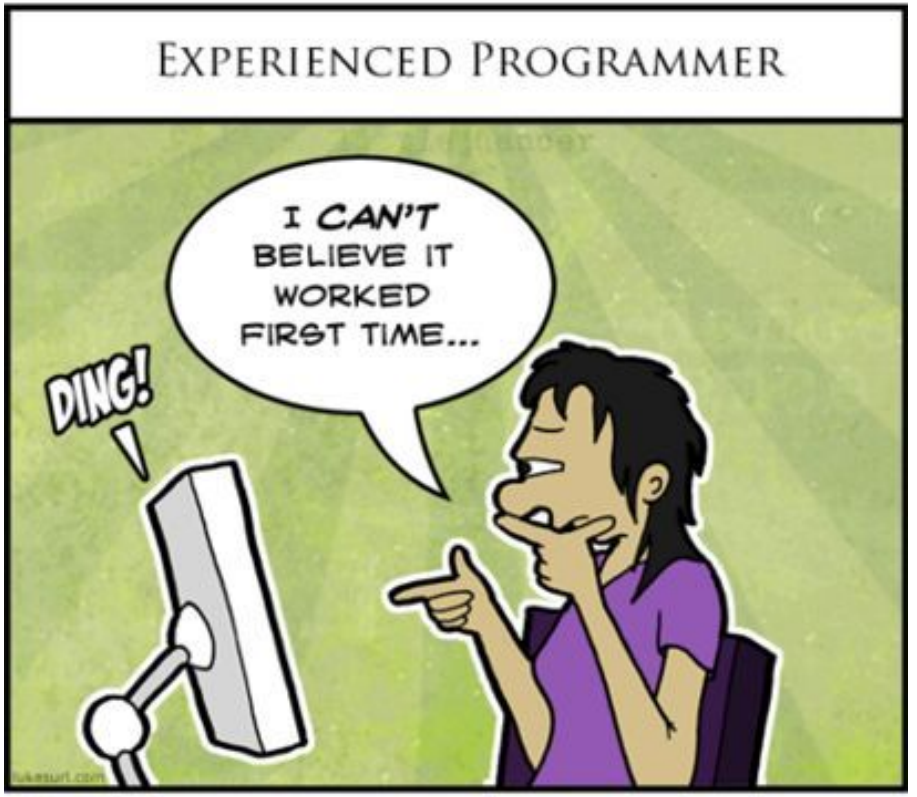
input

exact BF

our result

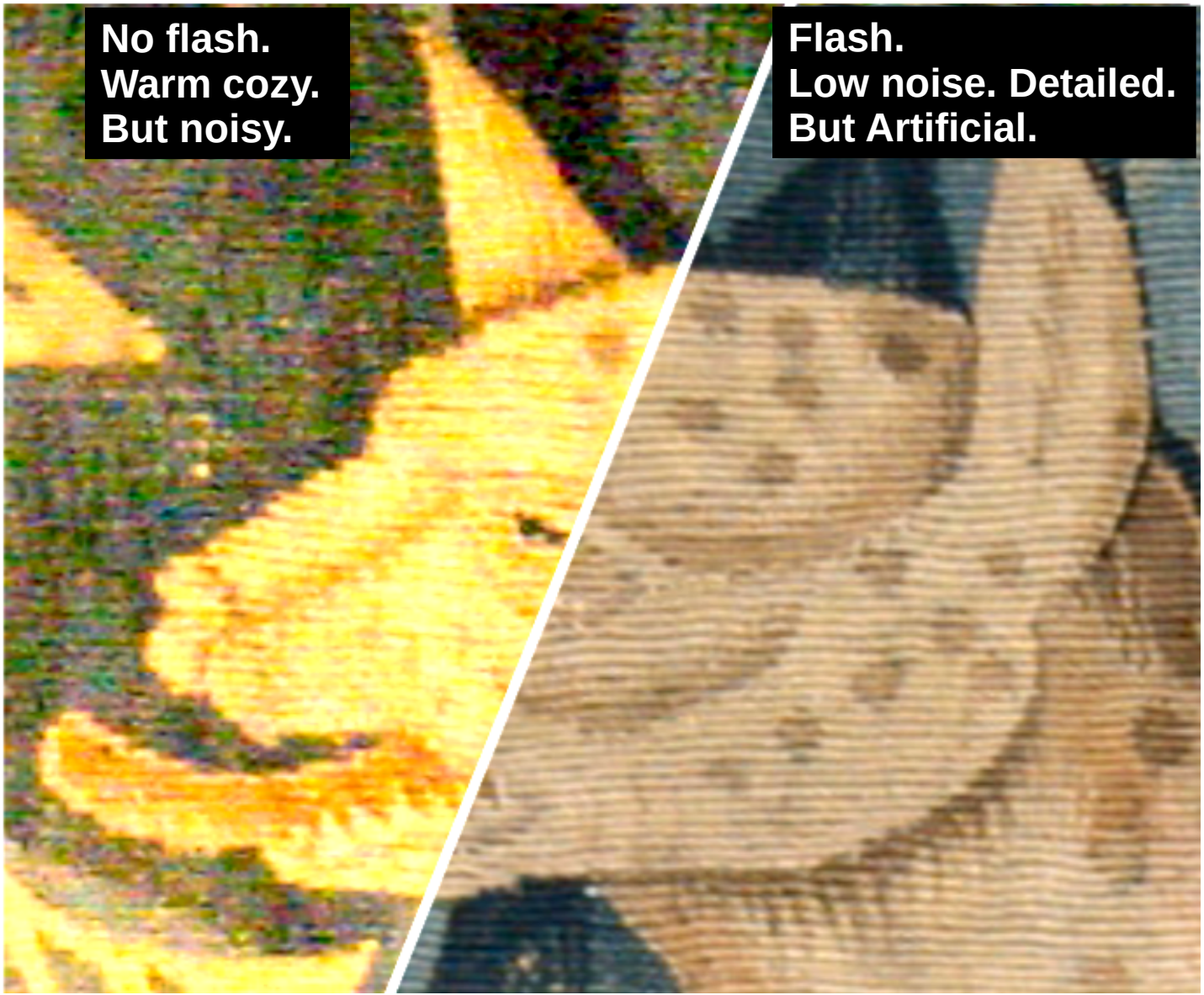
difference with exact computation (intensities in [0:1])


NOVICE PROGRAMMER
I CAN'T BELIEVE IT WORKED FIRST TIME!
DING!


EXPERIENCED PROGRAMMER
I CAN'T BELIEVE IT WORKED FIRST TIME...
DING!

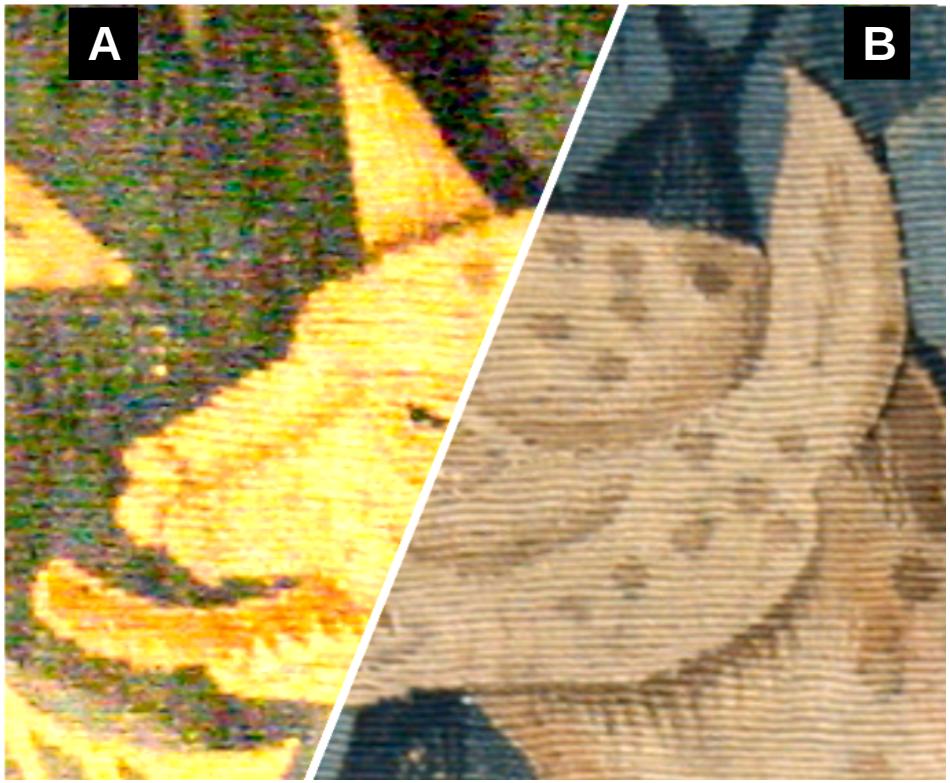
Edge-Preserving Smoothing

- **Cross / Joint** Bilateral Filter
  - Application: Merging camera images with & without flash
    - Images of a Belgian carpet



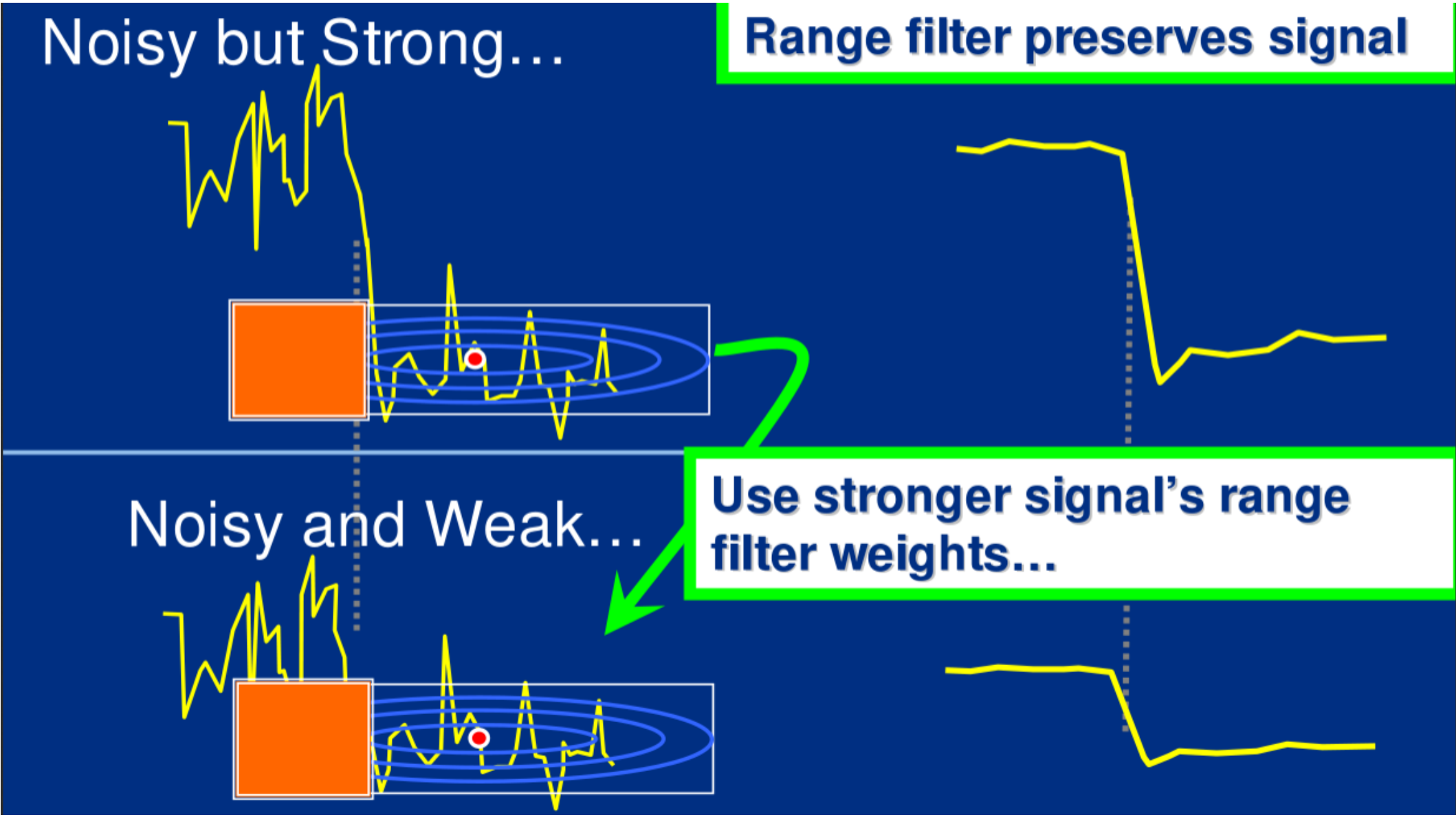
Edge-Preserving Smoothing

- **Cross / Joint** Bilateral Filter
  - 2 kinds of weights
    - Spatial Gaussian and Intensity Gaussian
    - **A** = image **without** flash
    - **B** = image **with** flash
  - Strategy
    - Smooth image A
    - 1) Weights based on spatial location → From A itself
    - 2) Weights based on intensity → From intensities in **B**



Edge-Preserving Smoothing

- Cross / Joint Bilateral Filter



people.csail.mit.edu/sparis/bf_course/slides/08_applications_advanced.pdf

Edge-Preserving Smoothing

- Cross / Joint Bilateral Filter
  - Enhances ability to find weak details in noisy image A (without flash)
    - Intensities in image B (with flash) inform which intensities to average in image A for noise reduction in image A

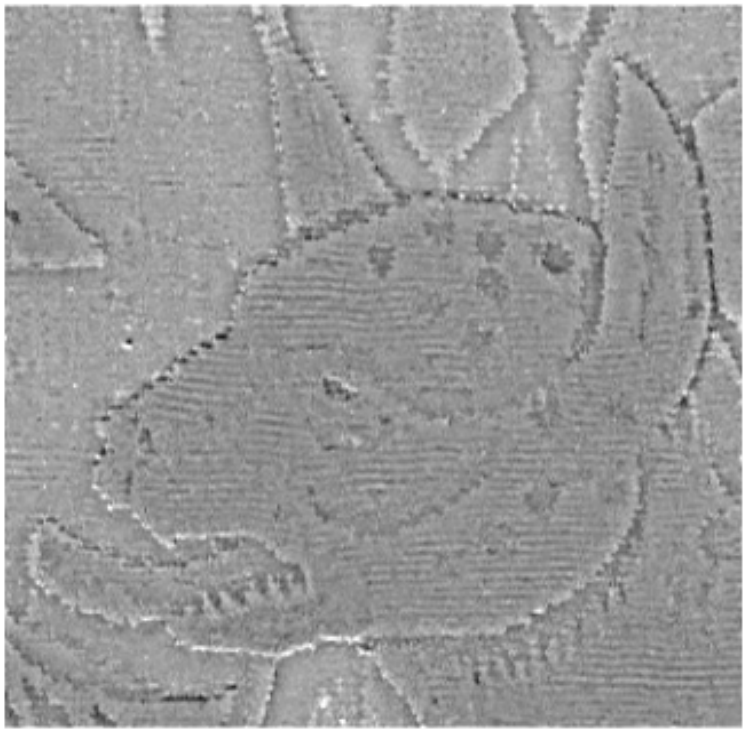
Edge-

- Cross / Joint Bilateral Filter

(a) No-Flash Flash

(b) Denoised via Bilateral Filter

(c) Denoised via Joint Bilateral Filter



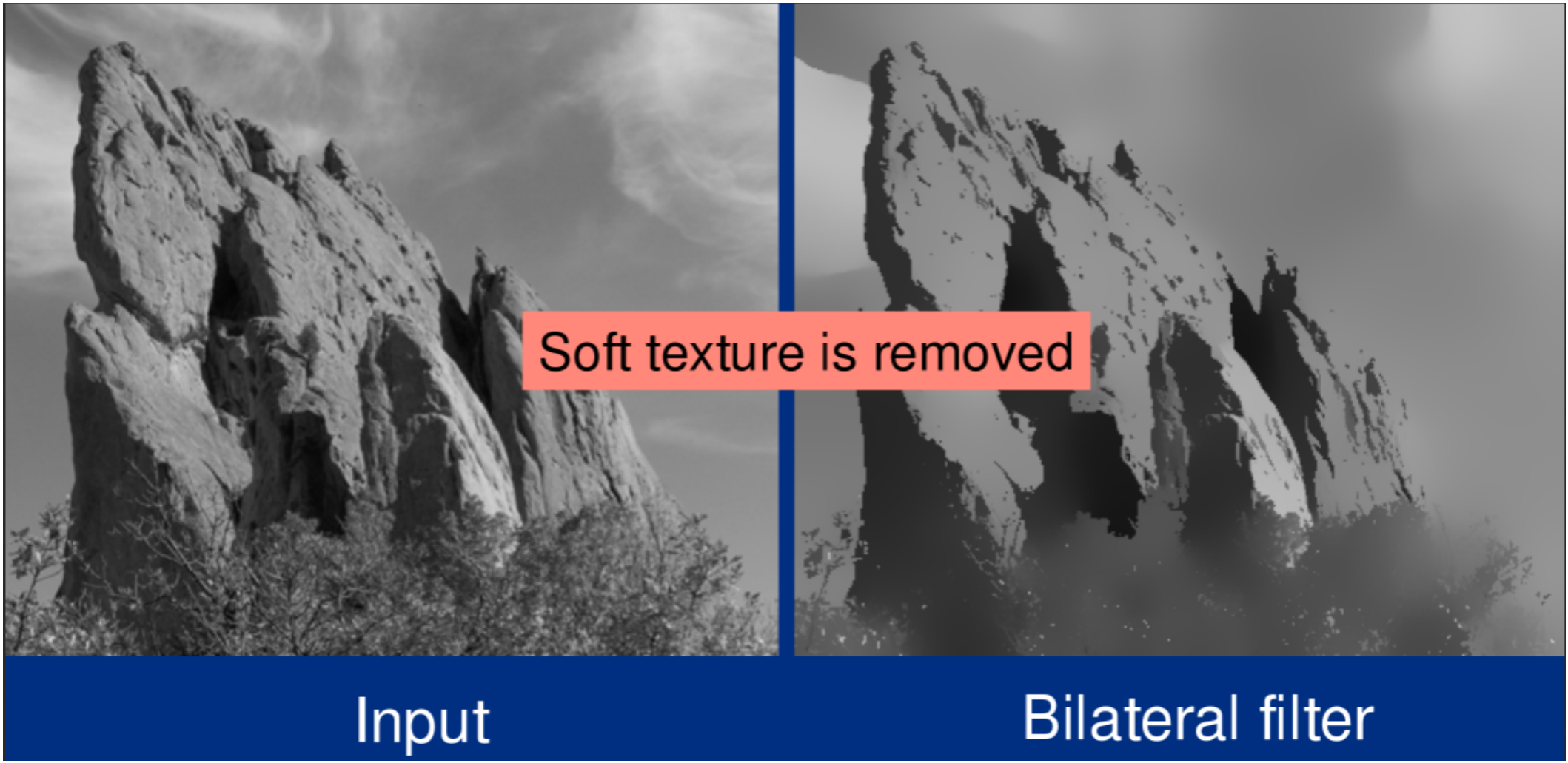
(d) Difference

research.microsoft.com/en-us/um/people/hoppe/flash.pdf

Edge-Preserving Smoothing

- Bilateral Filter **Limitations**

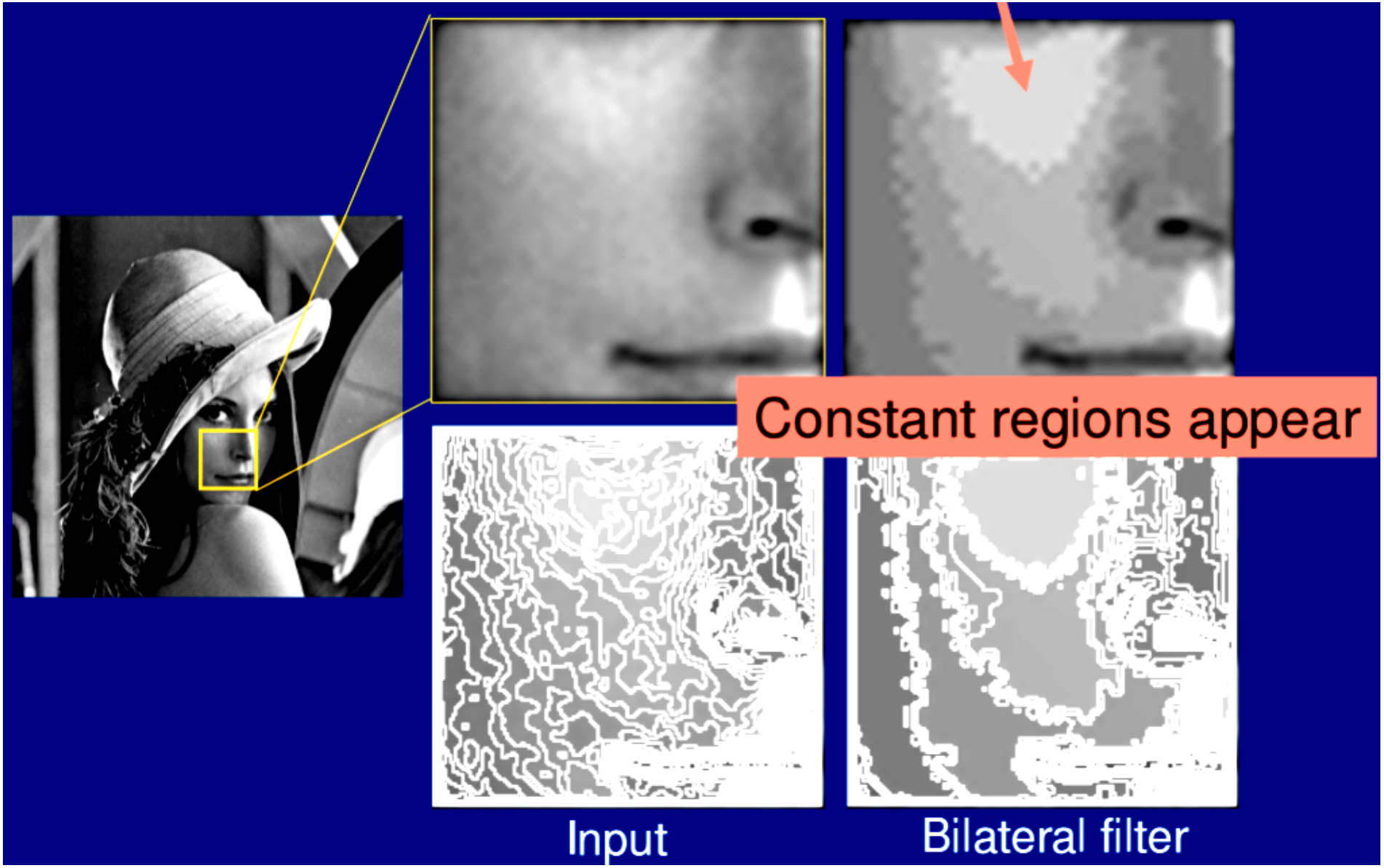
  (1) Texture “softer” than intensity-kernel SD → removed



Edge-Preserving Smoothing

- Bilateral Filter **Limitations**

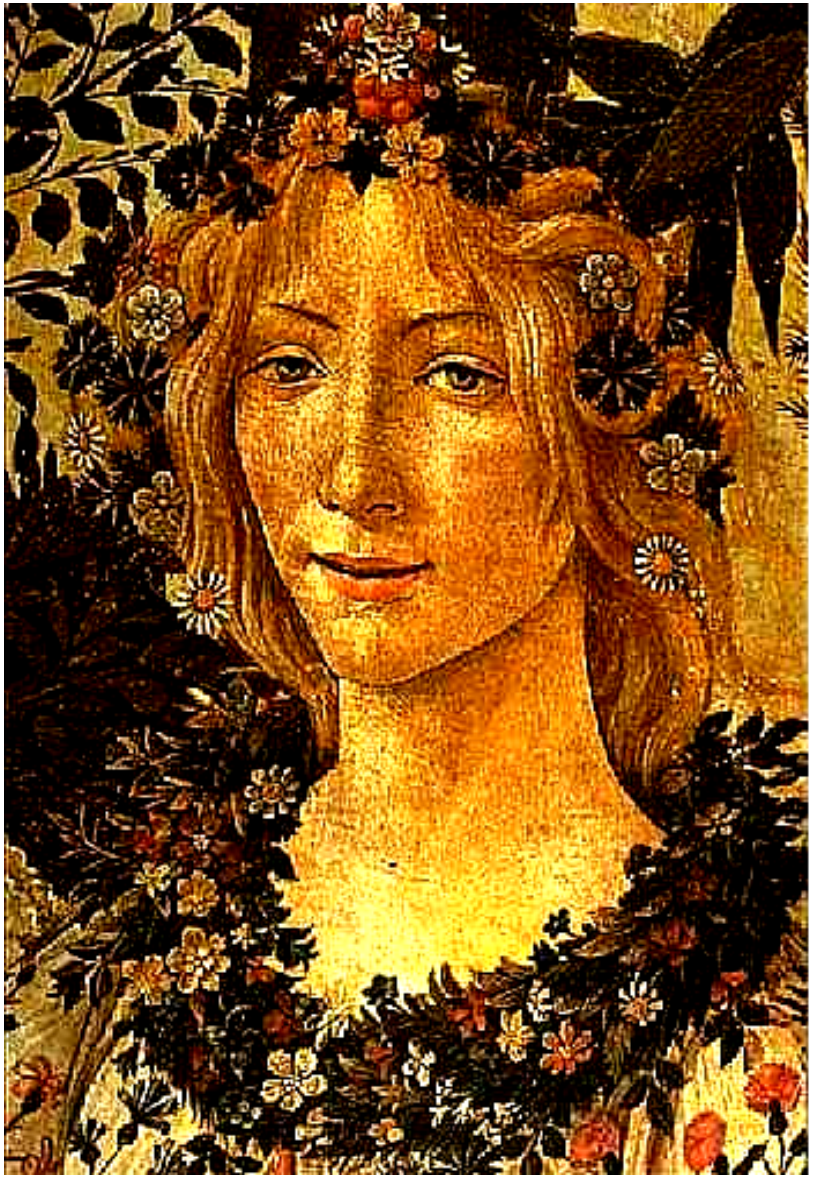
(2) “Staircase” artifacts (constant-intensity pieces)

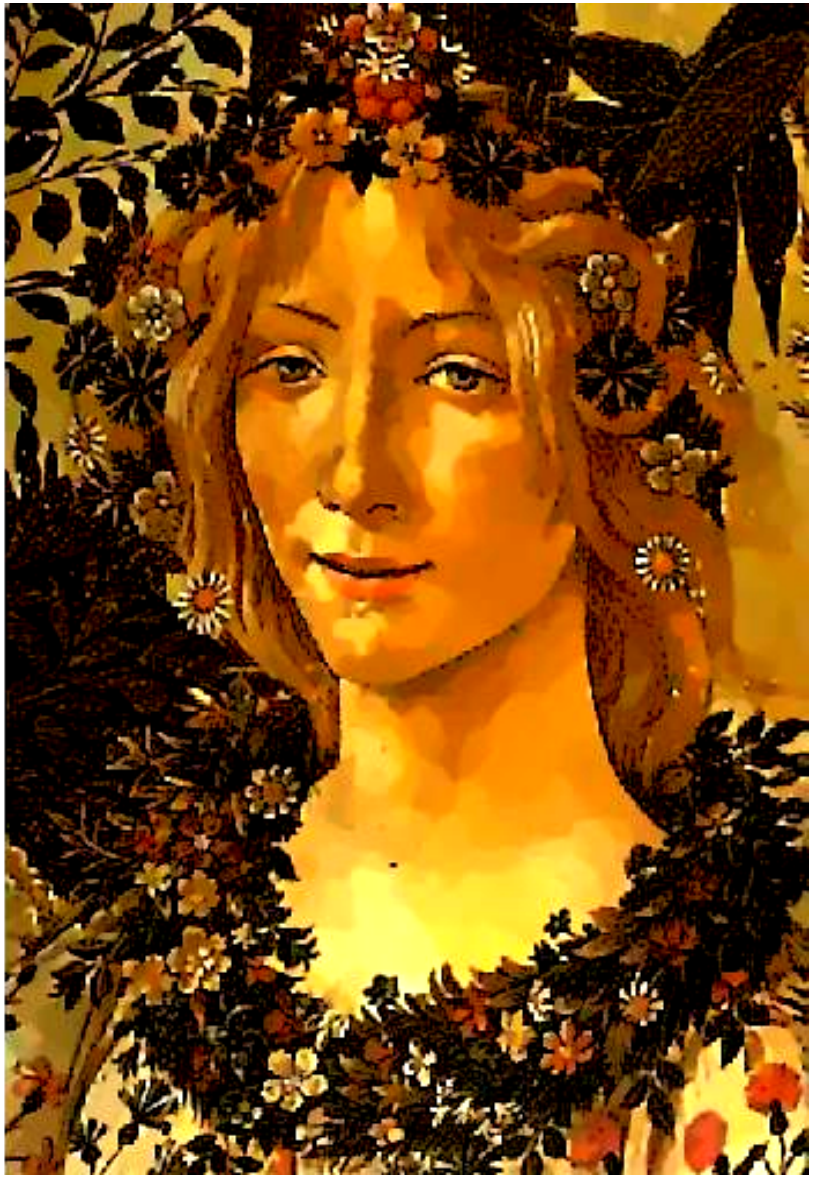


Edge-Preserving Smoothing

- Bilateral Filter Limitations

(2) “Staircase” artifacts





Edge-Preserving Smoothing

- Bilateral Filter Limitations – Staircasing
  - Let intensity function f(x) = (locally) **monotonic + convex**

    (1) Num. of pixels “q” s.t. $I_q \in (I_p - h, I_p)$ **>**
    Num. of pixels “r” s.t. $I_r \in (I_p, I_p + h)$

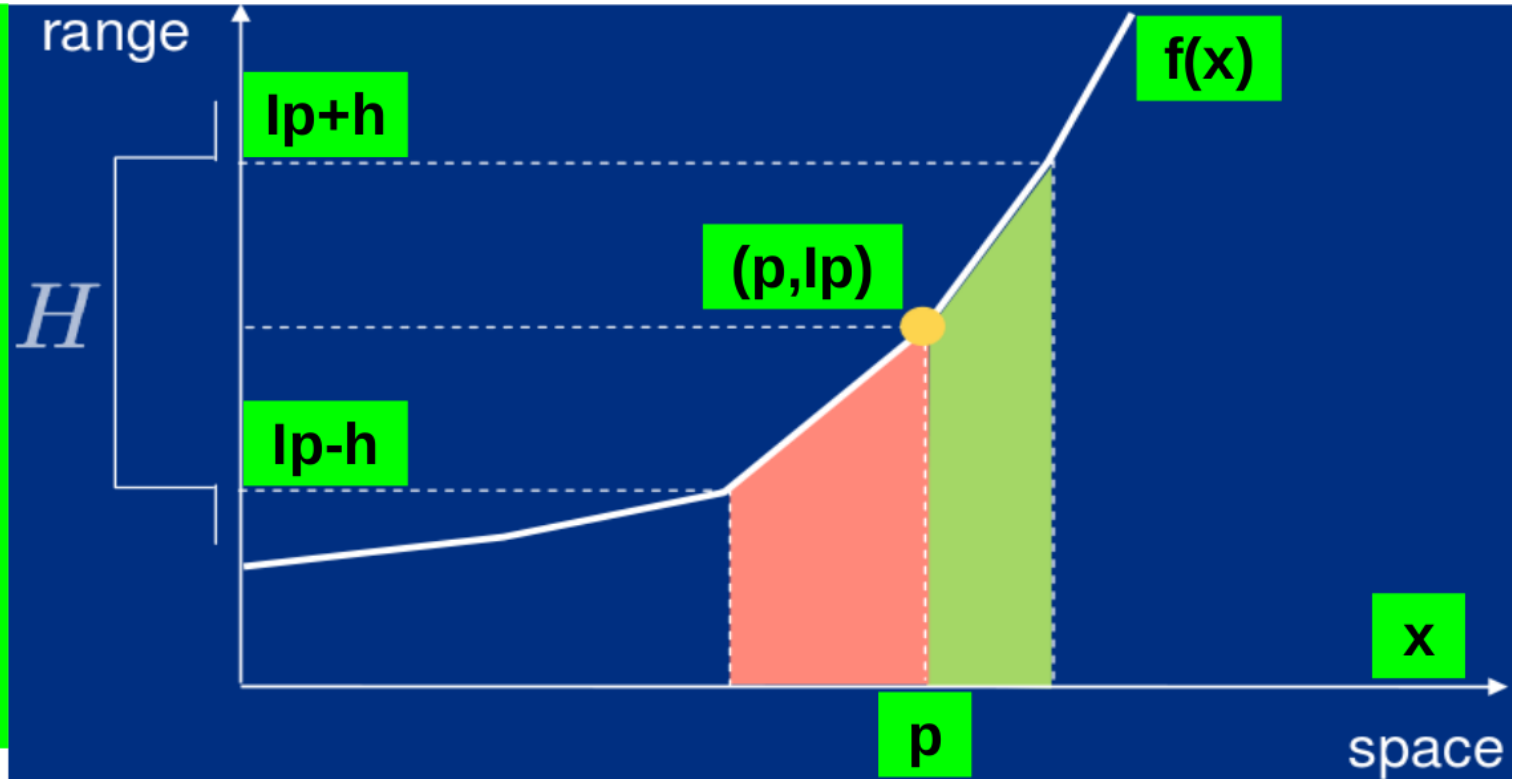
    (2) All points “q” and “r” have equal weight

For simplicity,

Intensity kernel = box function H.

Ignore spatial kernel.

Weighted avg = area under curve / length of x-interval



Edge-Preserving Smoothing

- Bilateral Filter Limitations – Staircasing

  (3) Weighted average

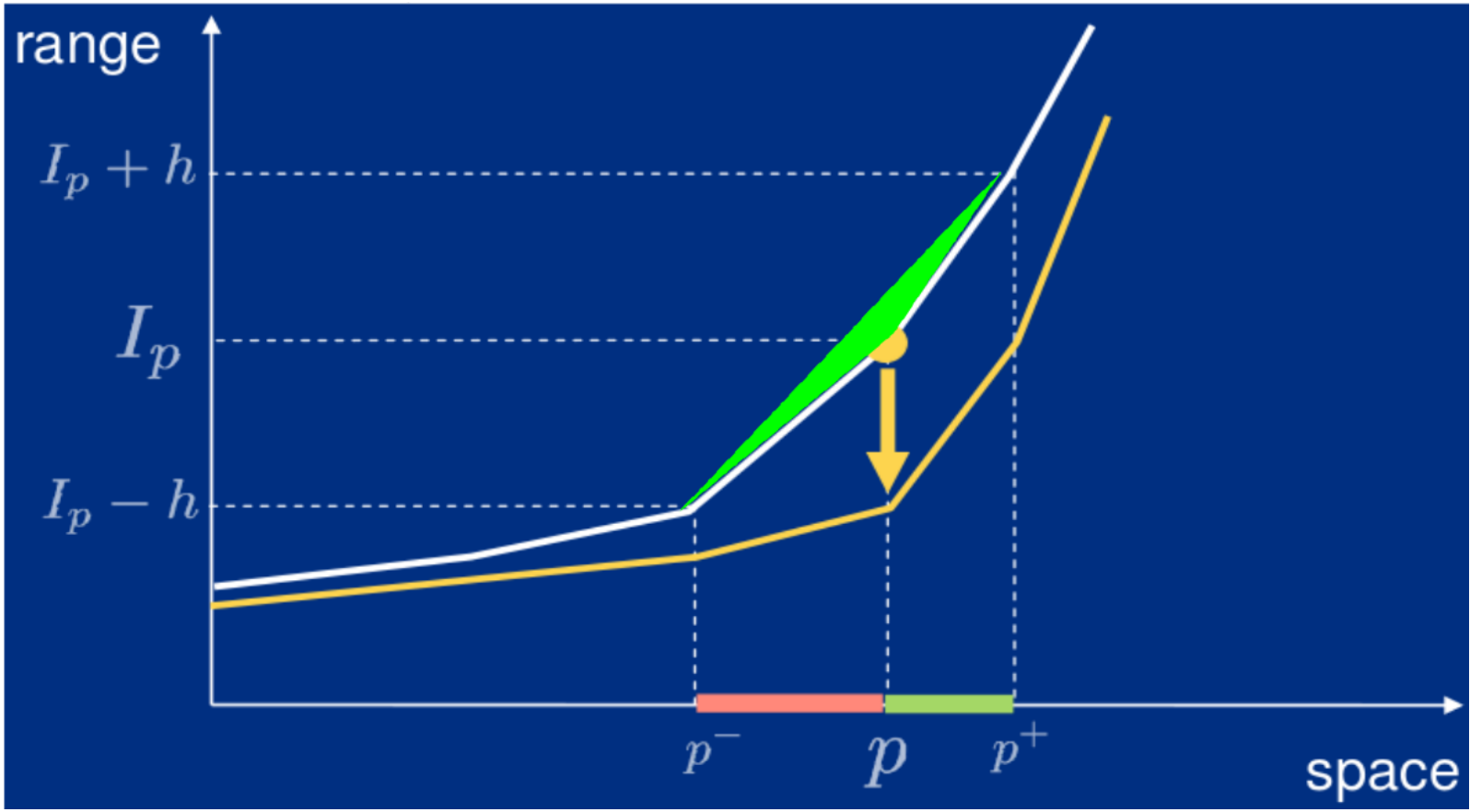
  = (right-trapezoid area – green area) / length of x-interval

  = Ip – green area / length of x-interval < Ip

For simplicity,

Intensity kernel = box function H

Ignore spatial kernel.



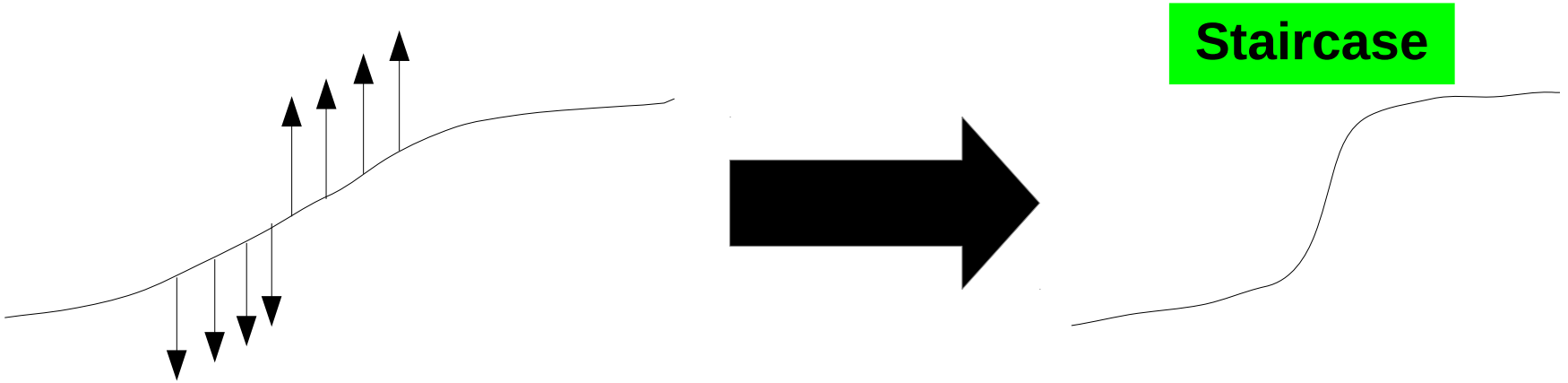
Edge-Preserving Smoothing

- Bilateral Filter Limitations – Staircasing
  - Let intensity function f(x)= (locally) **monotonic + concave**

    (1) Num. of pixels “q” s.t. $I_q \in (I_p\text{-}h, I_p)$ **<**
    Num. of pixels “r” s.t. $I_r \in (I_p, I_p\text{+}h)$

    (2) All points “q” and “r” have equal weight

    (3) **Weighted average** at “p” will be **> Ip**
  - **Reverse smoothing (NOT “sharpening”) occurs !**



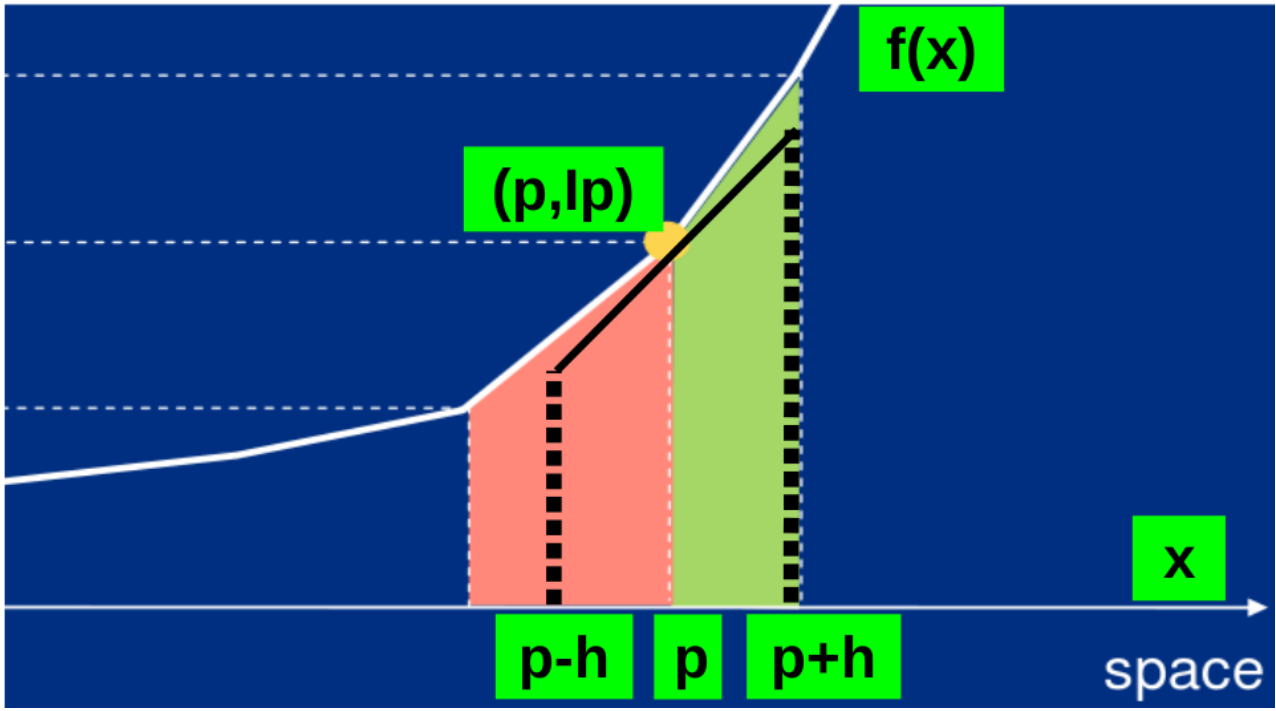
Edge-Preserving Smoothing

- Gaussian convolution doesn't cause staircasing
  - Assume **origin** at **(p , Ip)**
    - **Box, Gaussian spatial-kernel k(x)** → **even** function over “x”
    - **If** intensity function = linear = {**f(x) = mx**} → **odd** function over “x”
      - Weighted avg = Ip + Integral_{ -h → +h } mx k(x) dx = Ip
    - **Convex** f(x) = mx + g(x) s.t. g(x) > 0 → Integral > 0 → w.avg > Ip

Spatial kernel = box function H.

Ignore intensity kernel.

Weighted avg = area under curve / length of x-interval



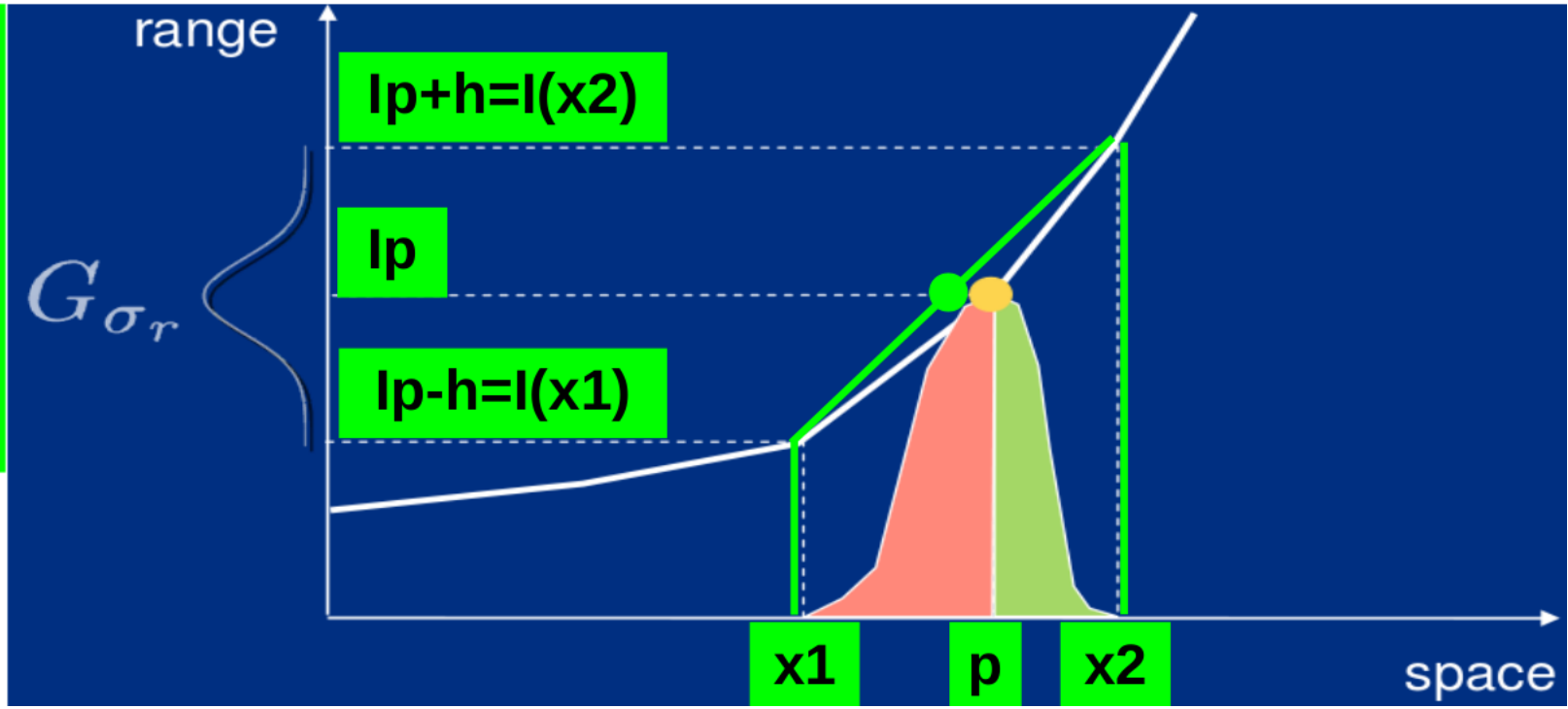
Edge-Preserving Smoothing

- Bilateral Filter Limitations – Staircasing
  - Assume **origin** at **((x1 + x2) / 2 , Ip)**
    - **Gaussian intensity-kernel G(.)** → **even** function over “I” (& “x”)
    - **If** intensity function = linear = {**f(x) = mx**} → **odd** function over “x”
      - Weighted avg = Ip + Integral_{ -(x2-x1)/2 → (x2-x1)/2 } mx G(mx) dx = Ip
    - **Convex** f(x) = mx - g(x) s.t. g(x) > 0 → Integral < 0 → W.Avg < Ip

For simplicity,

Intensity kernel = Gaussian

Ignore spatial kernel.

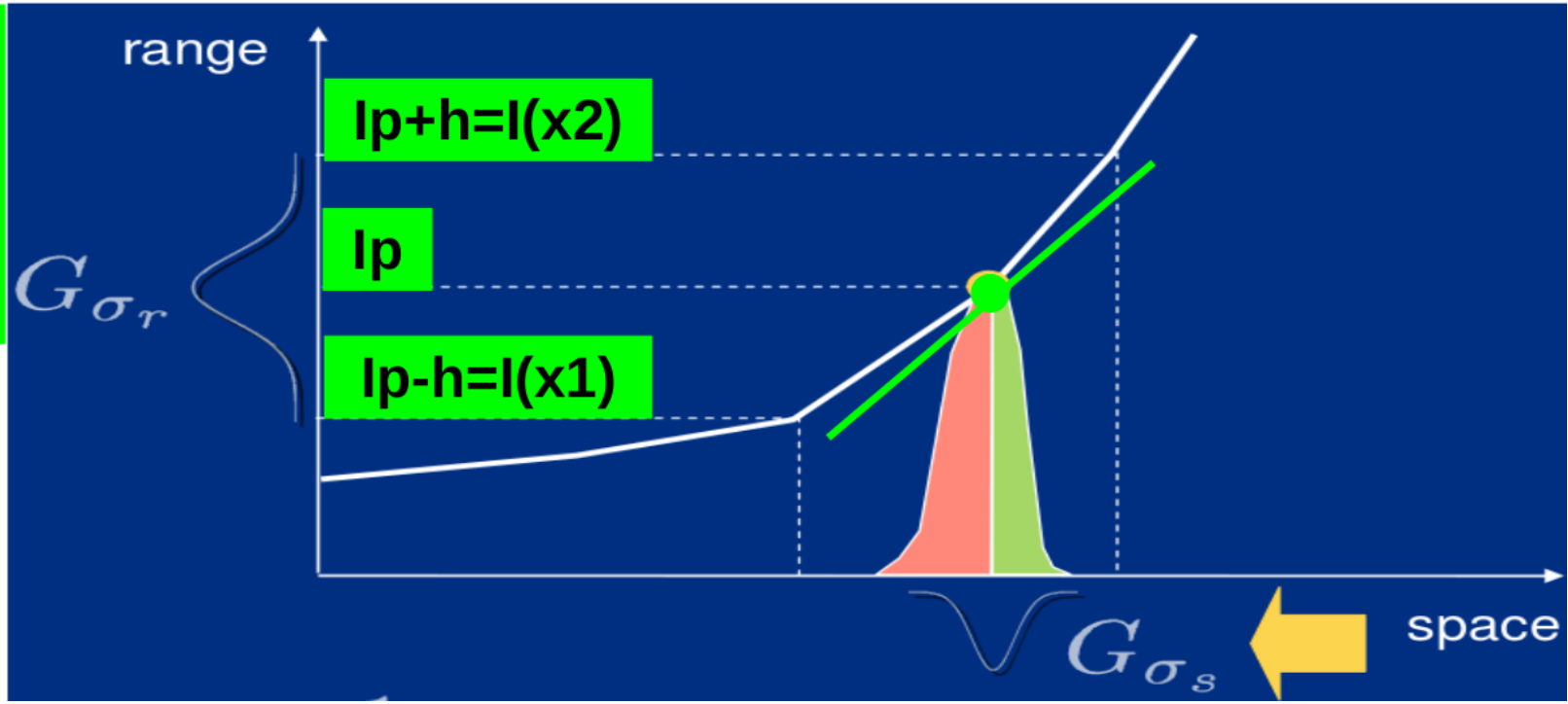


Edge-Preserving Smoothing

- Bilateral Filter Limitations – Staircasing
  - Assume **origin** at **(p , Ip)**
    - Gaussian intensity-kernel + Gaussian spatial-kernel
    - If intensity function = linear → **odd** → Weighted avg = Ip
    - Monotonic Convex f(x) → no guarantees (depends on I, weights)
      - In some cases, reverse smoothing can happen ← empirical evidence

Intensity kernel = Gaussian

Spatial kernel = Gaussian



Edge-Preserving Smoothing

- Bilateral Filter Limitations
  - Can staircasing be useful ?
    - Cartoons !

people.csail.mit.edu/sparis/bf_course/slides/09_limitations.pdf

Edge-Preserving Smoothing

- Bilateral Filter Limitations
  - Can this be useful ? Cartoons !

[Winnemöller, Olsen, Gooch, 2006]