BBM444

FUNDAMENTALS OF COMPUTATIONAL PHOTOGRAPHY

Lecture #05 – Edge-Aware Filtering

Erkut Erdem // Hacettepe University // Spring 2025

HACETTEPE UNIVERSITY COMPUTER VISION LAB

Today's Lecture

- Gaussian filtering

- Sharpening

- Bilateral filter

- Non-local means filter

- RegCov smoothing

- Rolling guidance filter

Disclaimer: The material and slides for this lecture were borrowed from

— Ioannis Gkioulekas' 15-463/15-663/15-862 "Computational Photography" class

— Wojciech Jarosz's CS 89.15/189.5 "Computational Aspects of Digital Photography" class

— Steve Marschner's CS6640 "Computational Photography" class

— Kaiming He's slides on Guided Image Filtering

Filtering

- The name "filter" is borrowed from frequency domain processing

- Accept or reject certain frequency components

- Fourier (1807):
  Periodic functions
  could be represented
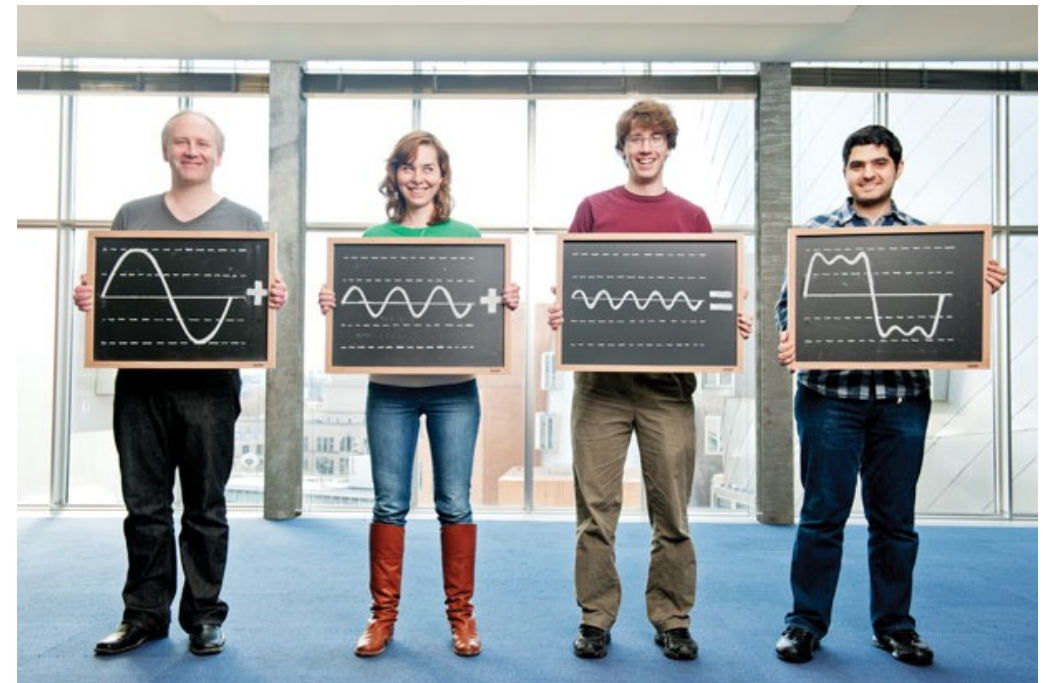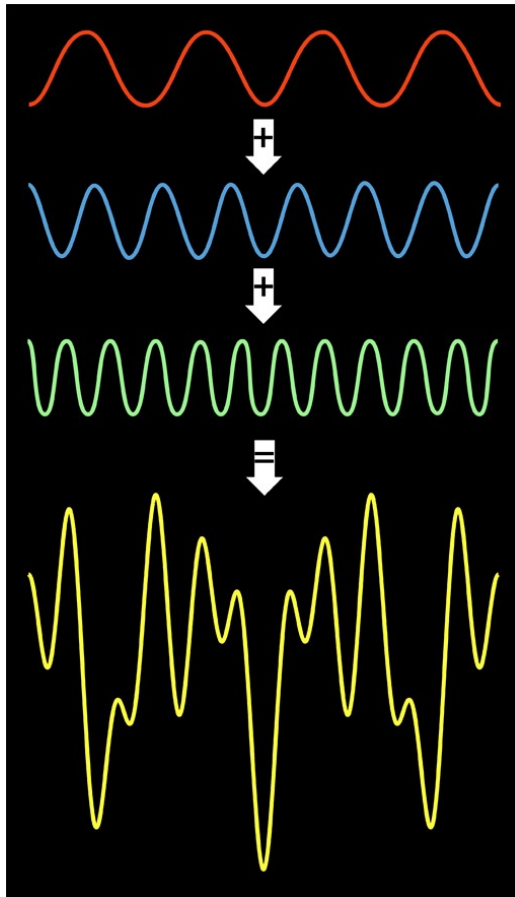 as a weighted sum of
  sines and cosines

Image courtesy of Technology Review

Signals

- A signal is composed of low and high frequency components

low frequency components: smooth / piecewise smooth

Neighboring pixels have similar brightness values
You're within a region

high frequency components: oscillatory

Neighboring pixels have different brightness values
You're either at the edges or noise points

Image Filtering

- <u>Idea:</u> Use the information coming from the neighboring pixels for processing
- Design a transformation function of the local neighborhood at each pixel in the image
    - Function specified by a "filter" or mask saying how to combine values from neighbors.

- Various uses of filtering:
    - Enhance an image (denoise, resize, etc)
    - Extract information (texture, edges, etc)
    - Detect patterns (template matching)

Filtering

- Processing done on a function
  - can be executed in continuous form (e.g. analog circuit)
  - but can also be executed using sampled representation
- Simple example: smoothing by averaging
- Can be modeled mathematically by <u>convolution</u>

continuous smoothing filter

$x-r$ x $x+r$

discrete smoothing filter

$i-r$ i $i+r$
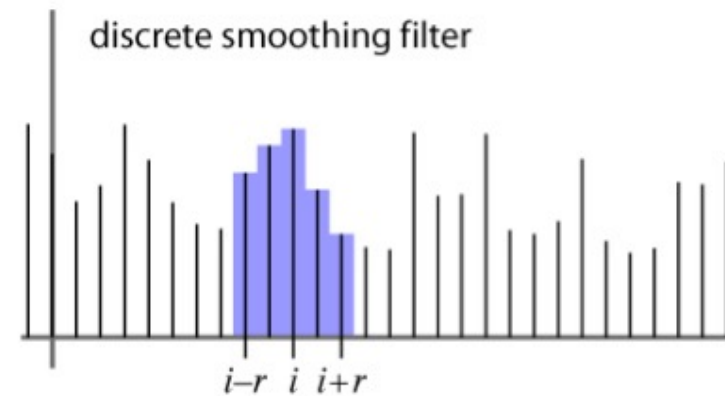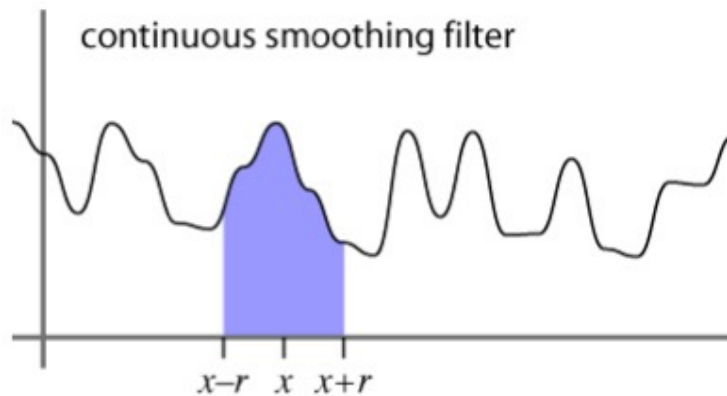
Discrete convolution

- Simple averaging:

$$b_{\mathrm{smooth}}[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} b[j]$$

- every sample gets the same weight

- Convolution: same idea but with <u>weighted</u> average

$$(a \star b)[i] = \sum_{j} a[j]b[i-j]$$

- each sample gets its own weight (normally zero far away)

- This is all convolution is: it is a <u>moving weighted average</u>
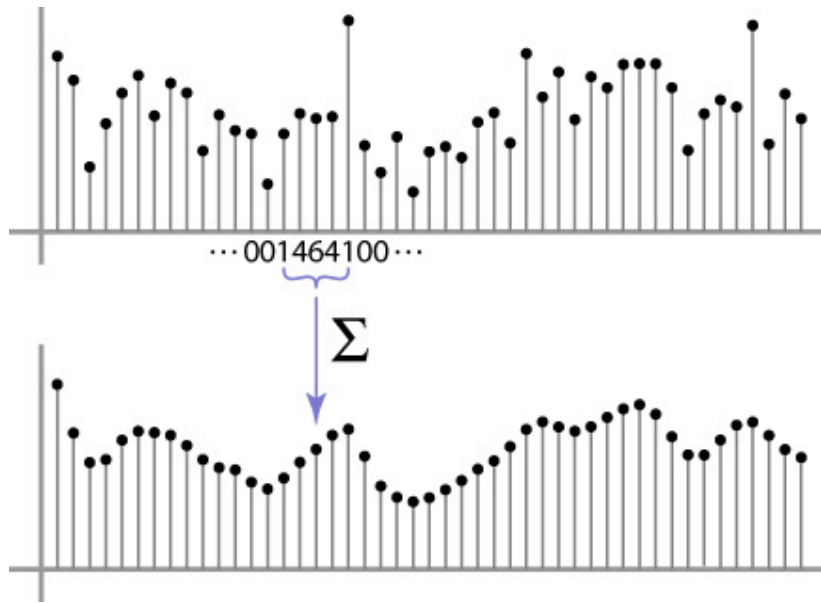
Filters

- Sequence of weights a[j] is called a filter
- Filter is nonzero over its region of support
  - usually centered on zero: support radius r
- Filter is normalized so that it sums to 1.0
  - this makes for a weighted average, not just any old weighted sum
- Most filters are symmetric about 0
  - since for images we usually want to treat left and right the same

$$\frac{1}{2r+1}$$
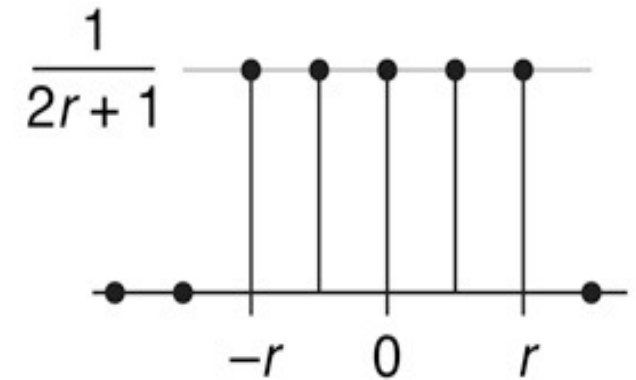
a box filter

Convolution and filtering

- Convolution applies with any sequence of weights
- Example: bell curve (gaussian-like) […, 1, 4, 6, 4, 1, …]/16

Discrete filtering in 2D

- Same equation, one more index

$$(a \star b)[i,j] = \sum_{i',j'} a[i',j']b[i-i',j-j']$$

- now the filter is a rectangle you slide around over a grid of numbers
- Usefulness of associativity
- often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
- this is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

Moving Average In 2D

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0								

Moving Average In 2D

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10							

Moving Average In 2D

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Averaging Filter

- What values belong in the kernel H for the moving average example?

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$\otimes$

$$H[u, v]$$

$$\frac{1}{9}$$

1	1	1
1	?	1
1	1	1

"box filter"

$$G[x, y]$$

0	10	20	30	30				
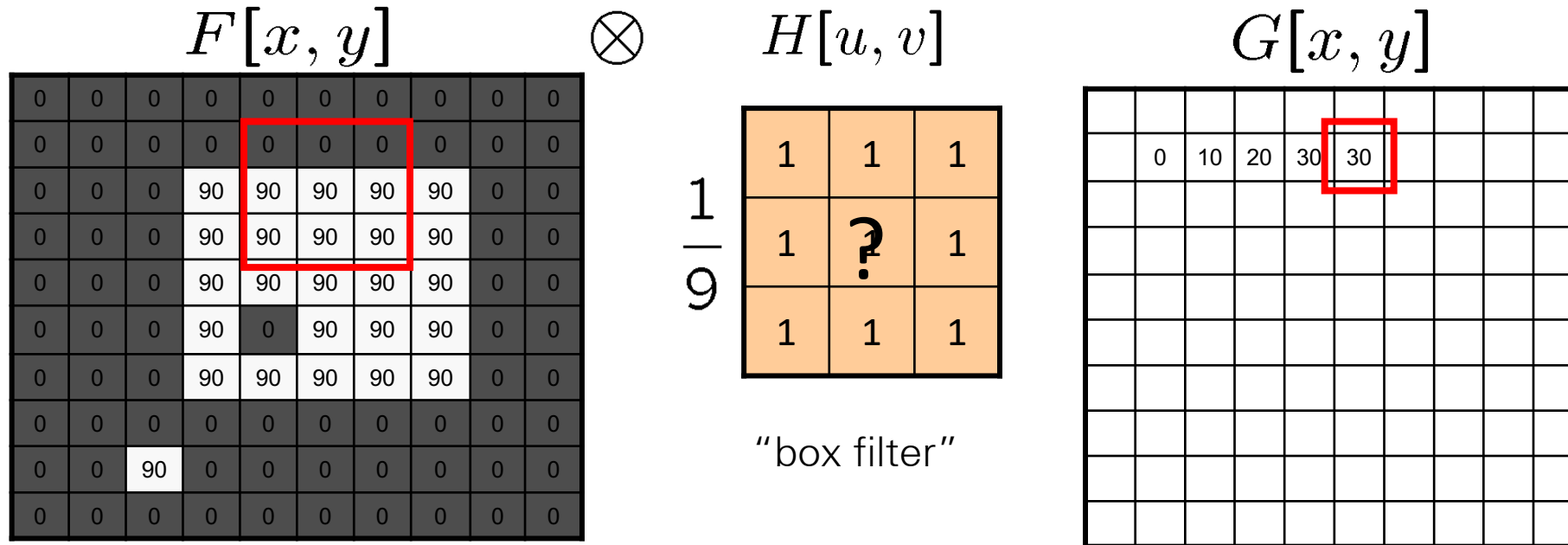
$$G = H \otimes F$$

Smoothing by averaging

depicts box filter:
white = high value, black = low value

original

filtered

Gaussian Filtering

Gaussian Filter

- What if we want nearest neighboring pixels to have the most influence on the output?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$F[x, y]$$

$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$$H[u, v]$$

This kernel is an approximation of a 2d Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$
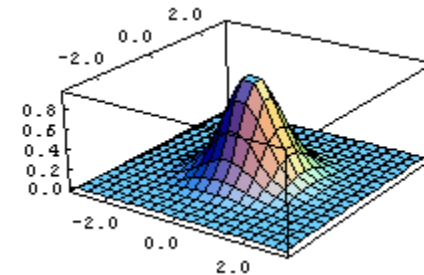
- Removes high-frequency components from the image ("low-pass filter").

Smoothing with a Gaussian
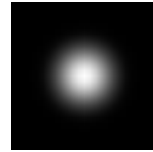
Smoothing with a Gaussian

Parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.
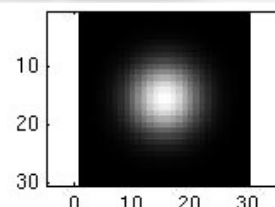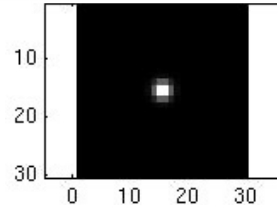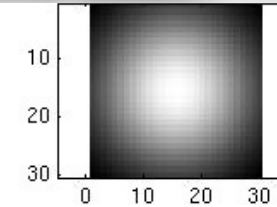
Strategy for Smoothing Images

- Images are not smooth because adjacent pixels are different.

- Smoothing = making adjacent pixels look more similar.

- Smoothing strategy
  pixel ~ average of its neighbors

Sharpening

How can we sharpen?

- Blurring was easy

- Sharpening is not as obvious

How can we sharpen?

- Blurring was easy

- Sharpening is not as obvious

- Idea: amplify the stuff not in the blurry image

- `output = input + k*(input-blur(input))`

Sharpening

input   -   blurred   =   high pass
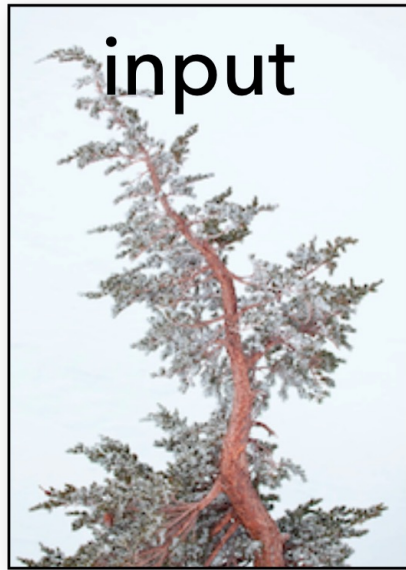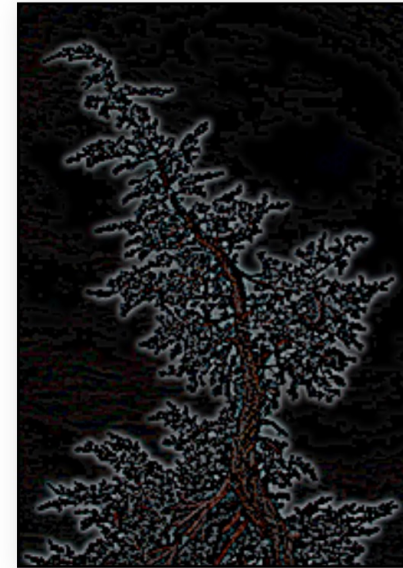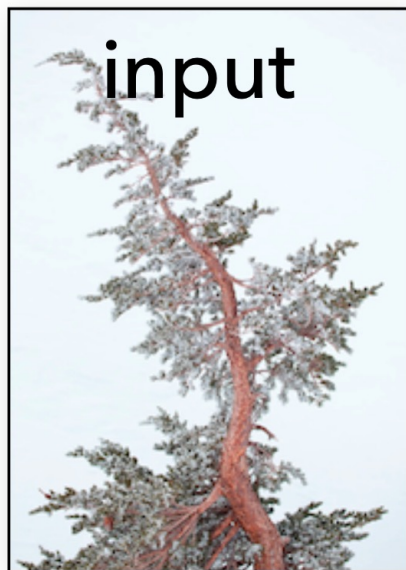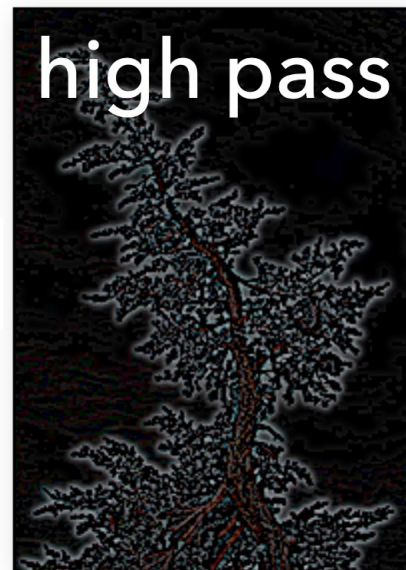
input   +k*   high pass   =   sharpened image

Sharpening: kernel view

- Recall

$$f' = f + k * (f - f \otimes g)$$

f is the input
f' is a sharpened image
g is a blurring kernel
k is a scalar controlling the strength of sharpening

Sharpening: kernel view

- Recall

$$f' = f + k * (f - f \otimes g)$$

- Denote δ the Dirac kernel (pure impulse)

$$f = f \otimes \delta$$

Sharpening: kernel view

- Recall

$$f' = f + k * (f - f \otimes g)$$

$$f' = f \otimes \delta + k * (f \otimes \delta - f \otimes g)$$

$$f' = f \otimes ((k+1)\delta - g)$$

- Sharpening is also a convolution

Sharpening kernel

- Note: many other sharpening kernels exist
  (just like we saw multiple blurring kernels)

- Amplify the difference between a pixel and its neighbors

$$f' = f \otimes ((k+1)\delta - g)$$
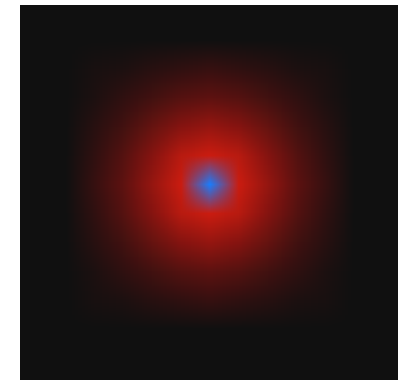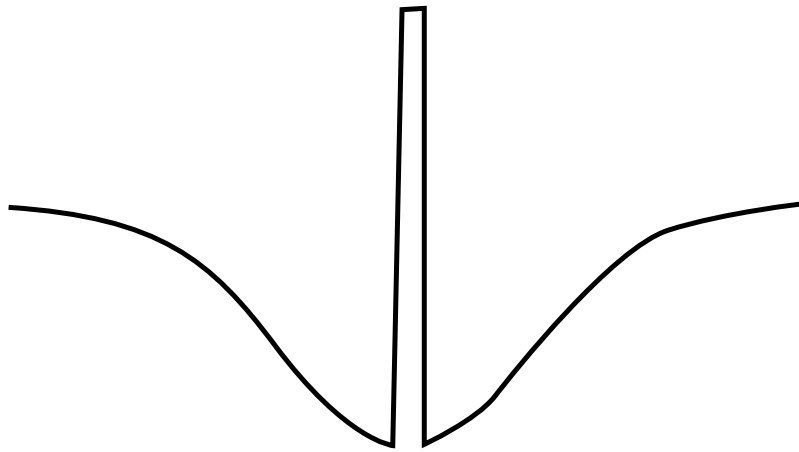
blue: positive
red: negative

Alternate interpretation

- `out = input + k*(input-blur(input))`
- `out = (1 + k)*input - k*blur(input)`
- `out = lerp(blur(input), input, 1+k)`

linearly extrapolate from the blurred image "past" the original input image

Sharpening

input − blurred = high pass

input +k* high pass = sharpened image
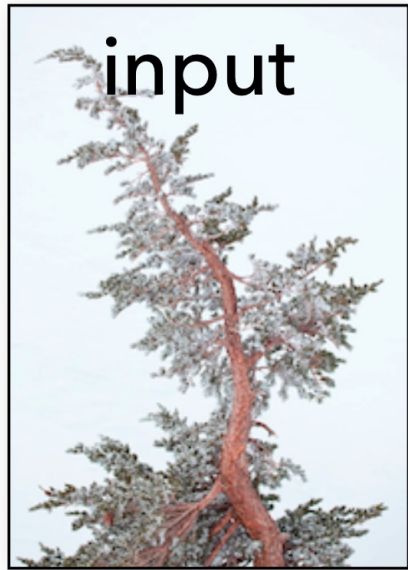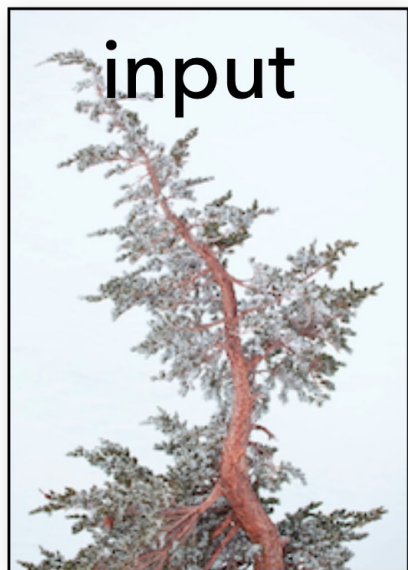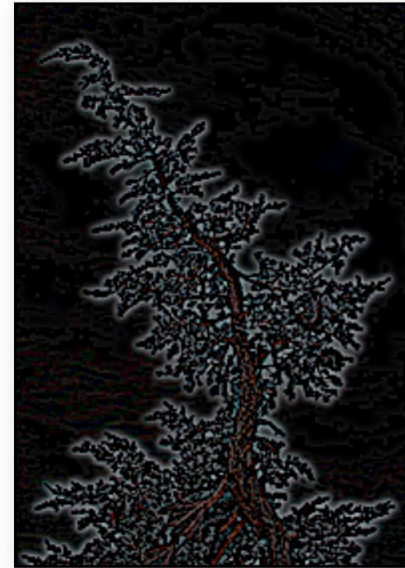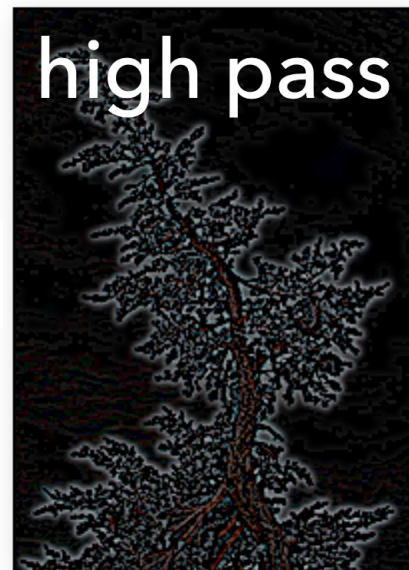
Unsharp mask

- Sharpening is often called "unsharp mask" because photographers used to sandwich a negative with a blurry positive film in order to sharpen

http://www.tech-diy.com/UnsharpMasks.htm
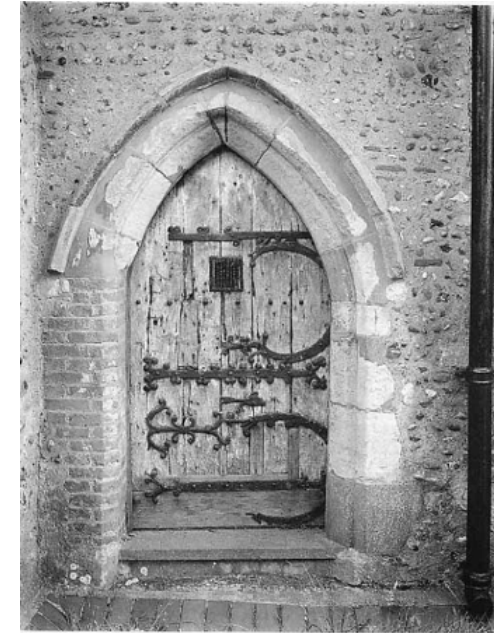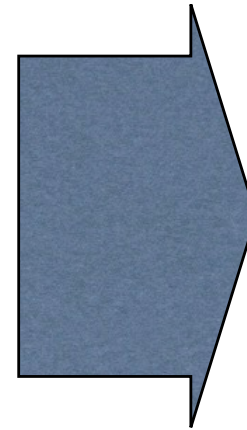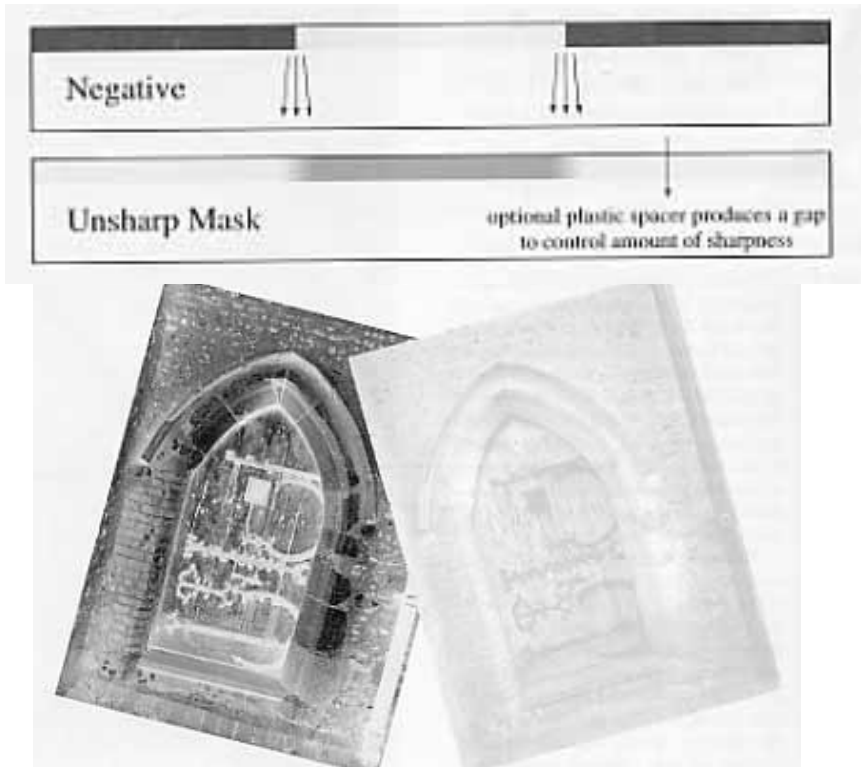
Unsharp mask

Fig.4: The two examples here show a detail of the brickwork to the left of the church door. The one on the left was printed with the negative alone – the one on the right was printed with both negative and mask as a sandwich. The increase in local contrast and edge sharpness is minute, but clearly visible. Grade 2.5 was used for the straight print but increased to 4.5 for the sandwiched image to compensate for the reduced contrast.

Fig.5: These two examples show a detail of the lower right hand side of the church door. Here the difference in sharpness is clearly visible between the (left) negative and (right) sandwich prints.
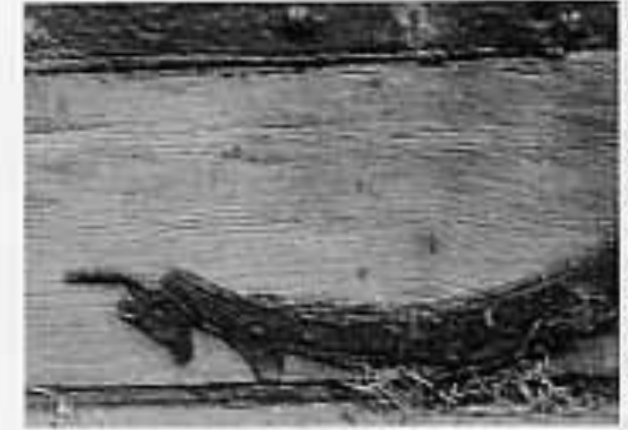
Problem with excess

- Haloes around strong edges

Bilateral Filter

Gaussian Filter

Idea: weighted average of pixels.

per-pixel multiplication

input

$*$

output

average

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

normalized
Gaussian function

1

0
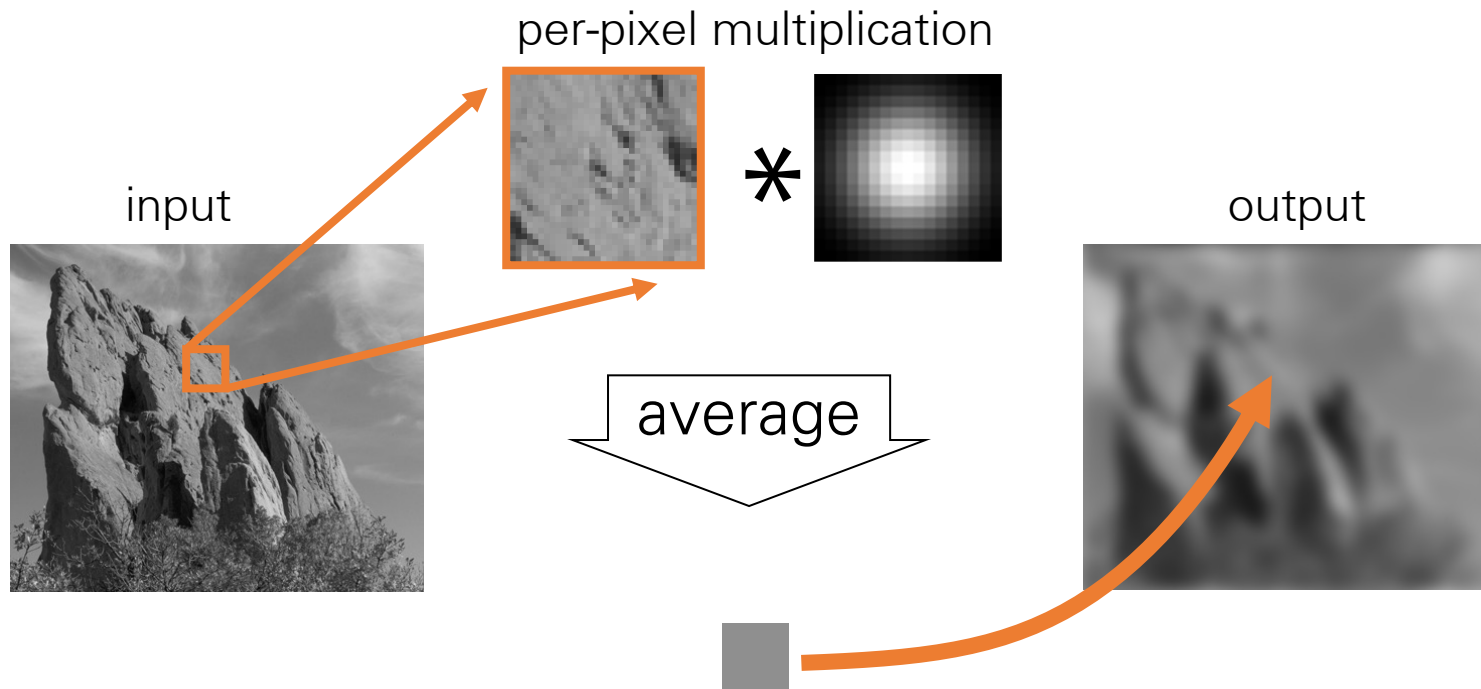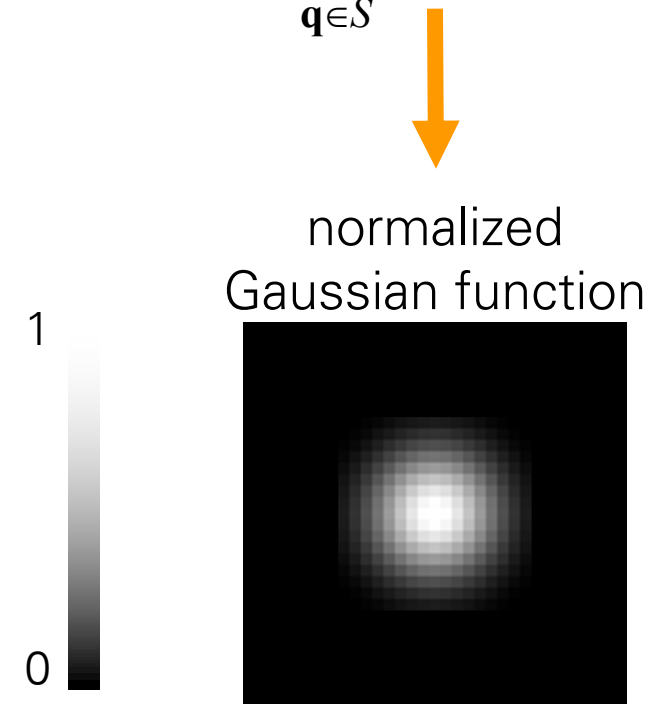
Spatial Parameter

input

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

size of the window
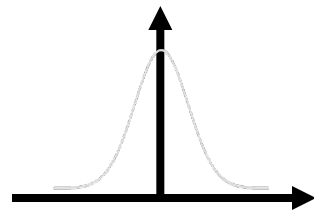
small **σ**

large **σ**

limited smoothing
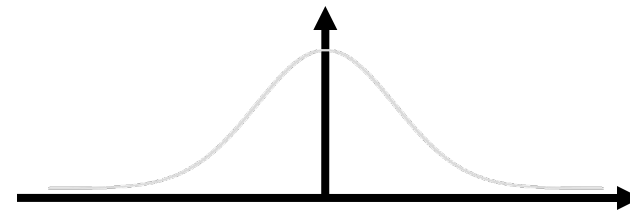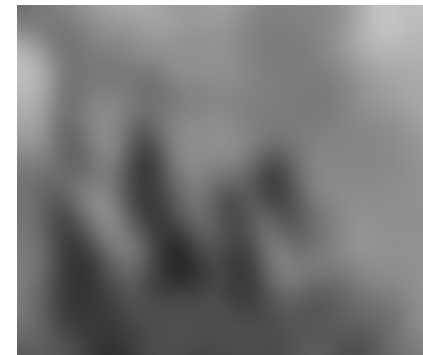
strong smoothing

Properties of Gaussian Blur

- Weights independent of spatial location
  - linear convolution
  - well-known operation
  - efficient computation (recursive algorithm, FFT…)

- Does smooth images

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

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

input
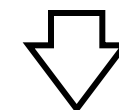
output

Blur Comes from Averaging across Edges

input

output
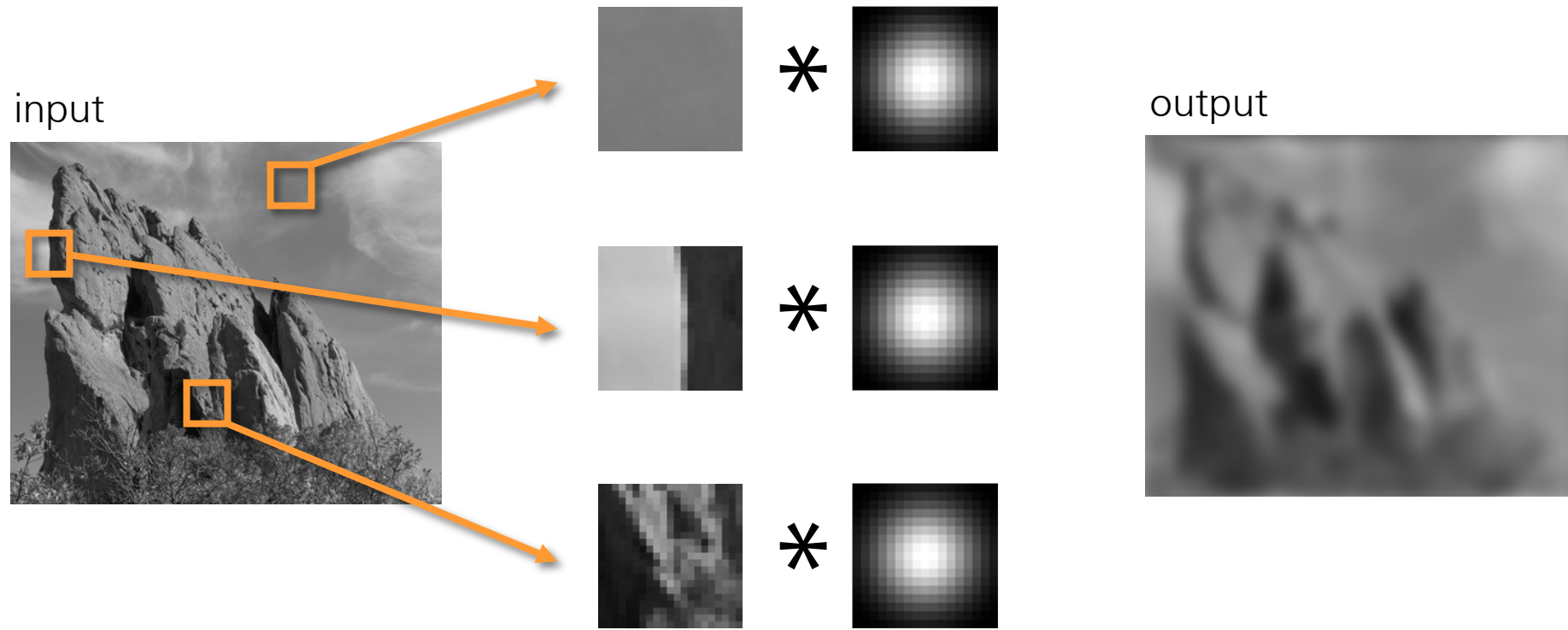
Same Gaussian kernel everywhere.

Bilateral Filter: No Averaging across Edges

input

*

*

*

output
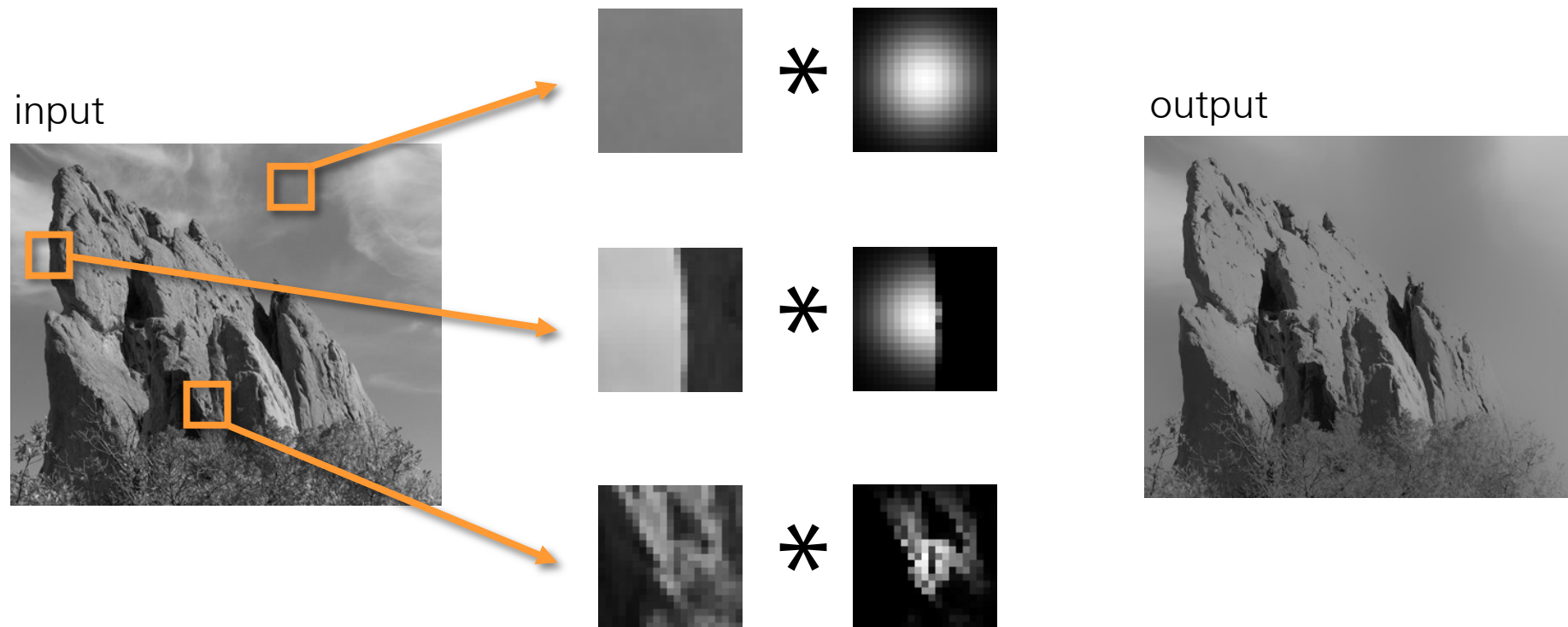
The kernel shape depends on the image content.

[Aurich 95, Smith 97, Tomasi 98]

Bilateral Filter: An Additional Edge Term

Same idea: weighted average of pixels.

new

not new

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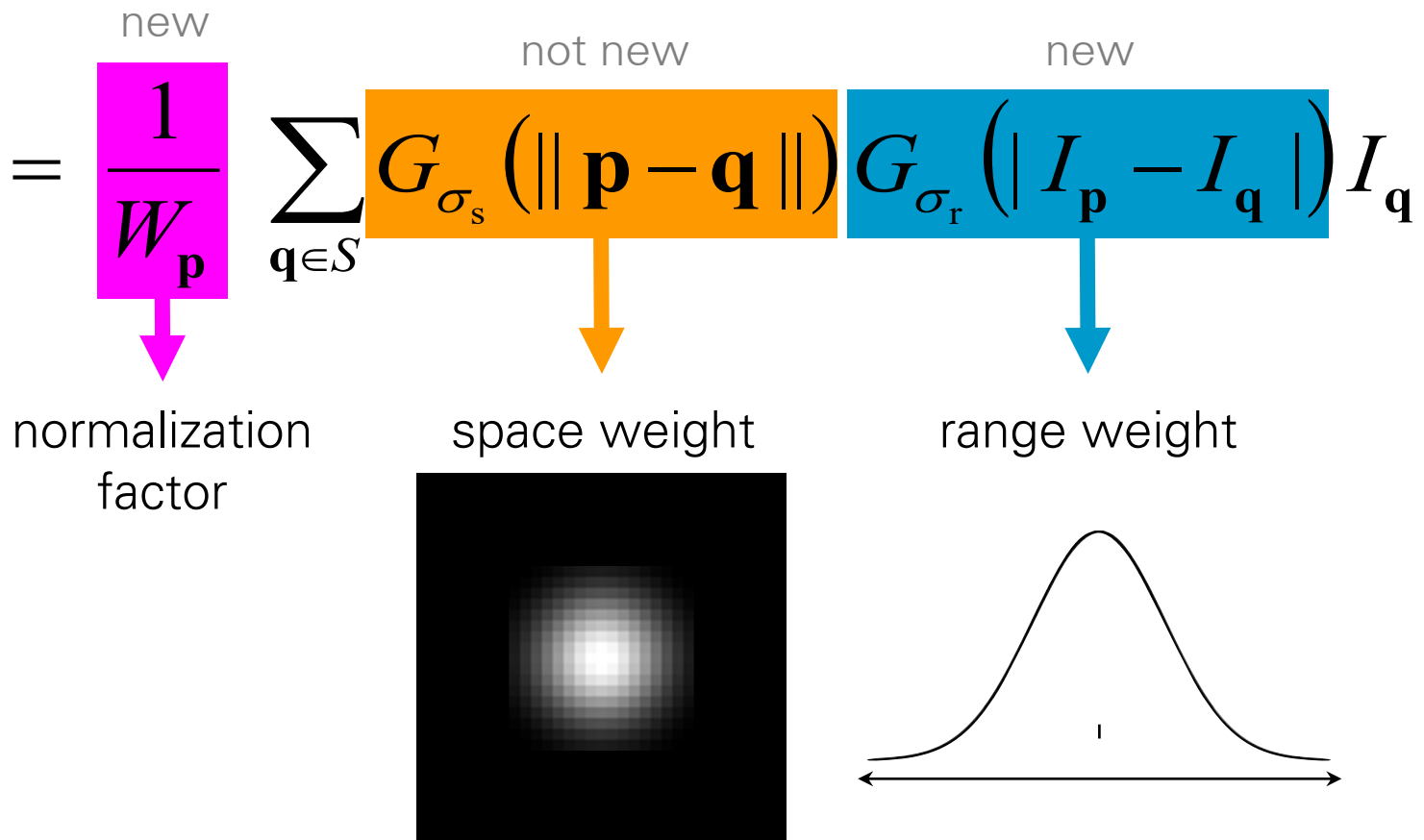
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}\left(\|\,\mathbf{p} - \mathbf{q}\,\|\right) G_{\sigma_r}\left(|\,I_{\mathbf{p}} - I_{\mathbf{q}}\,|\right) I_{\mathbf{q}}$$

normalization
factor

space weight

range weight

Bilateral Filter: An Additional Edge Term

Same idea: weighted average of pixels.

new

not new

new

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

normalization
factor

space weight

range weight

favor nearby pixels

favor similar pixels

Space and Range Parameters

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

- space σ_{s} : spatial extent of the kernel, size of the considered neighborhood.
- range σ_{r} : "minimum" amplitude of an edge

Gaussian filtering visualization

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$
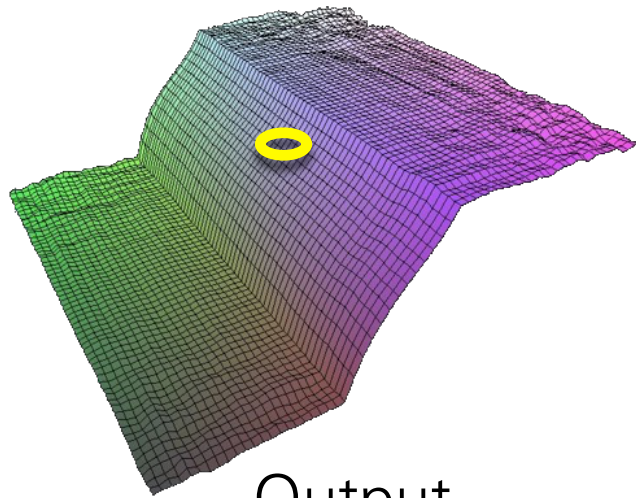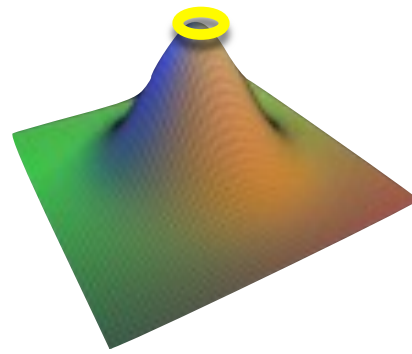
Output   Gaussian Filter   Input
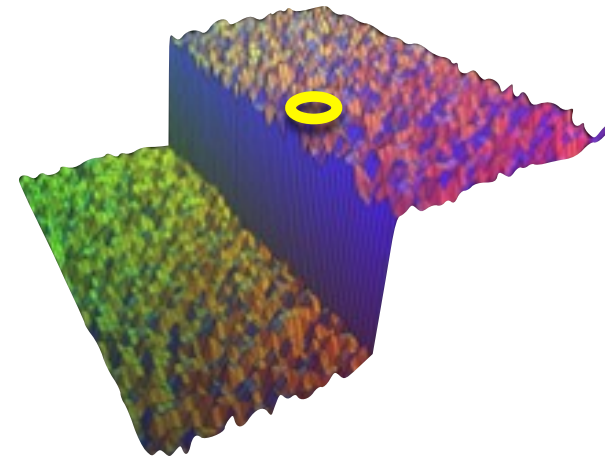
Bilateral filtering visualization

$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k,n+l]$$

Spatial range

Intensity range

Output

Bilateral Filter
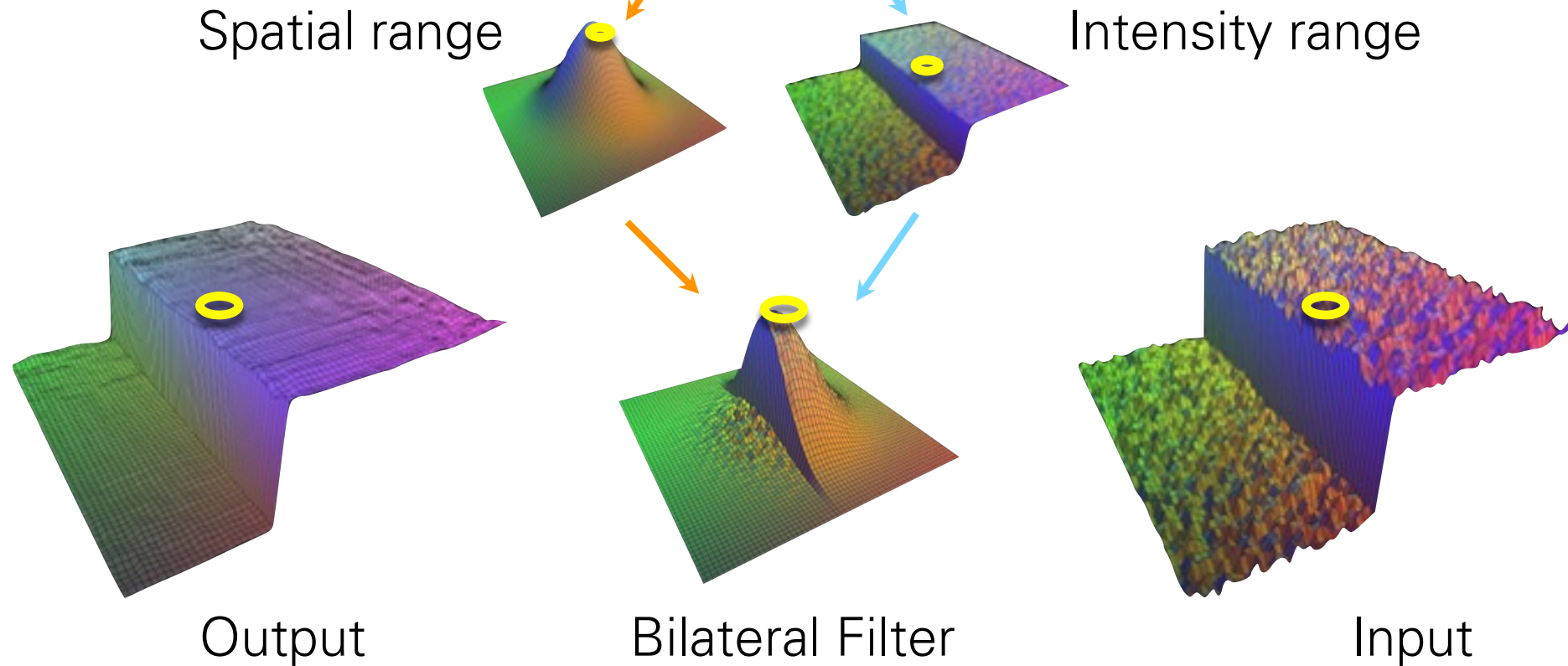
Input

Exploring the Parameter Space

σ_r = 0.1 | σ_r = 0.25 | σ_r = ∞ (Gaussian blur)

input

σ_s = 2

σ_s = 6
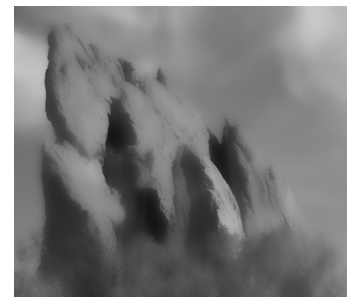
σ_s = 18
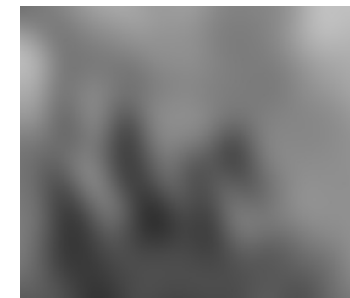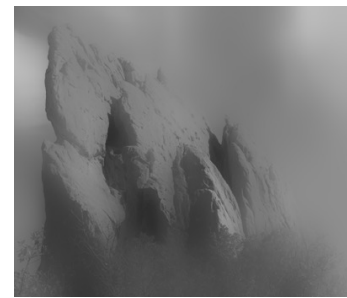
Bilateral Filtering Color Images

For gray-level images

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

intensity difference

scalar

For color images

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

color difference

3D vector
(RGB, Lab)

input

output
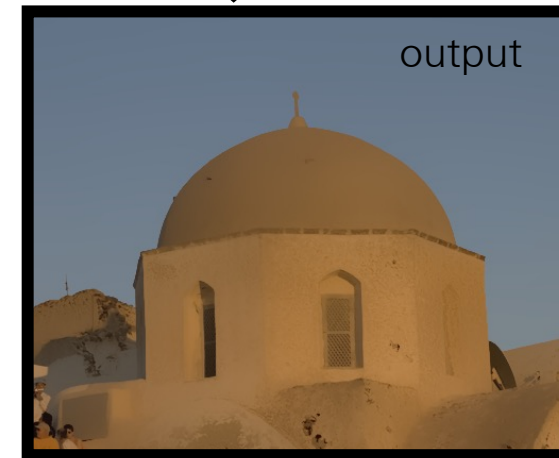
Hard to Compute

- Nonlinear

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

- Complex, spatially varying kernels

- Cannot be precomputed, no FFT…
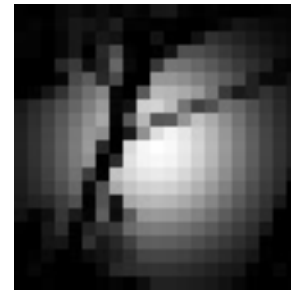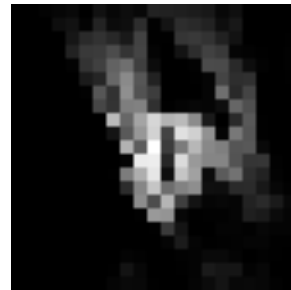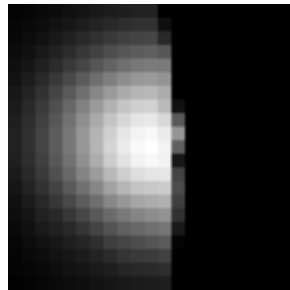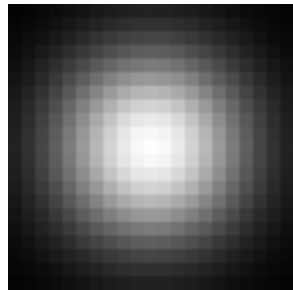
- Brute-force implementation is slow > 10min

Additional Reading: S. Paris and F. Durand, A Fast Approximation of the Bilateral Filter using a Signal Processing Approach, In Proc. ECCV, 2006

Denoising

noisy input   bilateral filtering   median filtering

Contrast enhancement

How would you use bilateral filtering for sharpening?
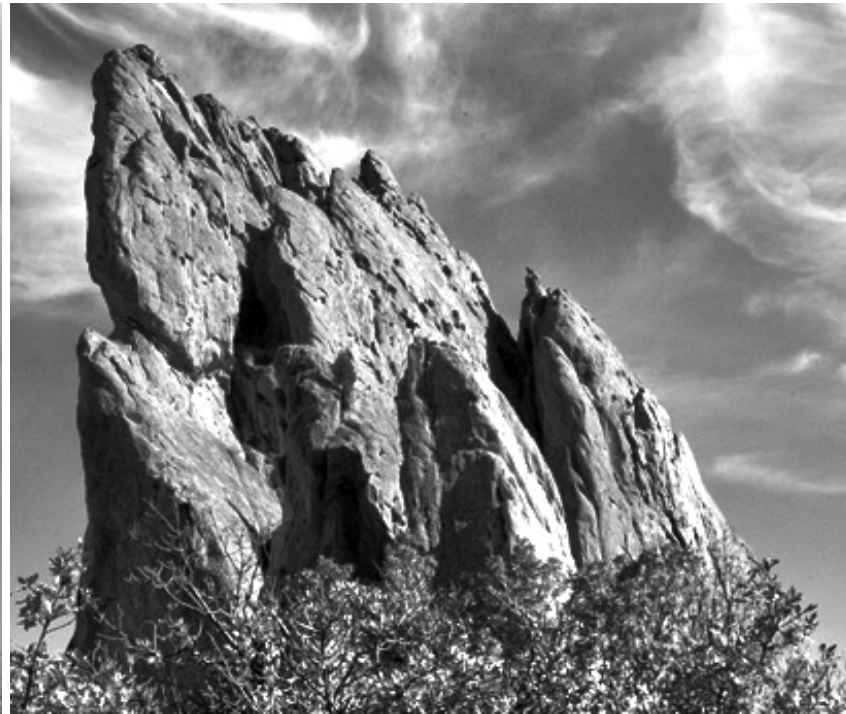
| input | sharpening based on bilateral filtering | sharpening based on Gaussian filtering |
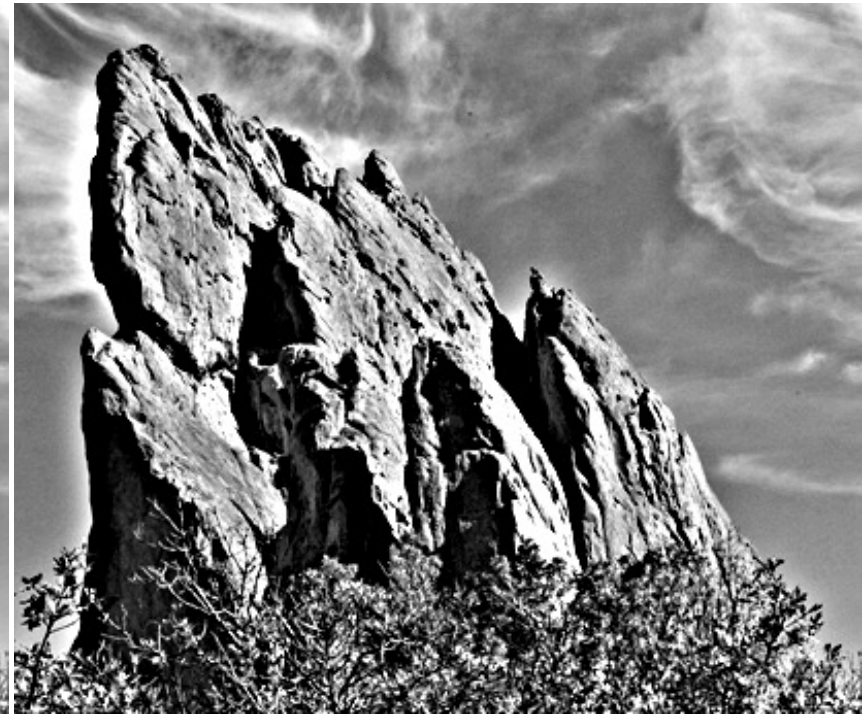
Photo retouching

Photo retouching

original

digital pore removal (aka bilateral filtering)

Before

After

Close-up comparison

original

digital pore removal
(aka bilateral filtering)

Cartoonization

input

cartoon rendition

Cartoonization
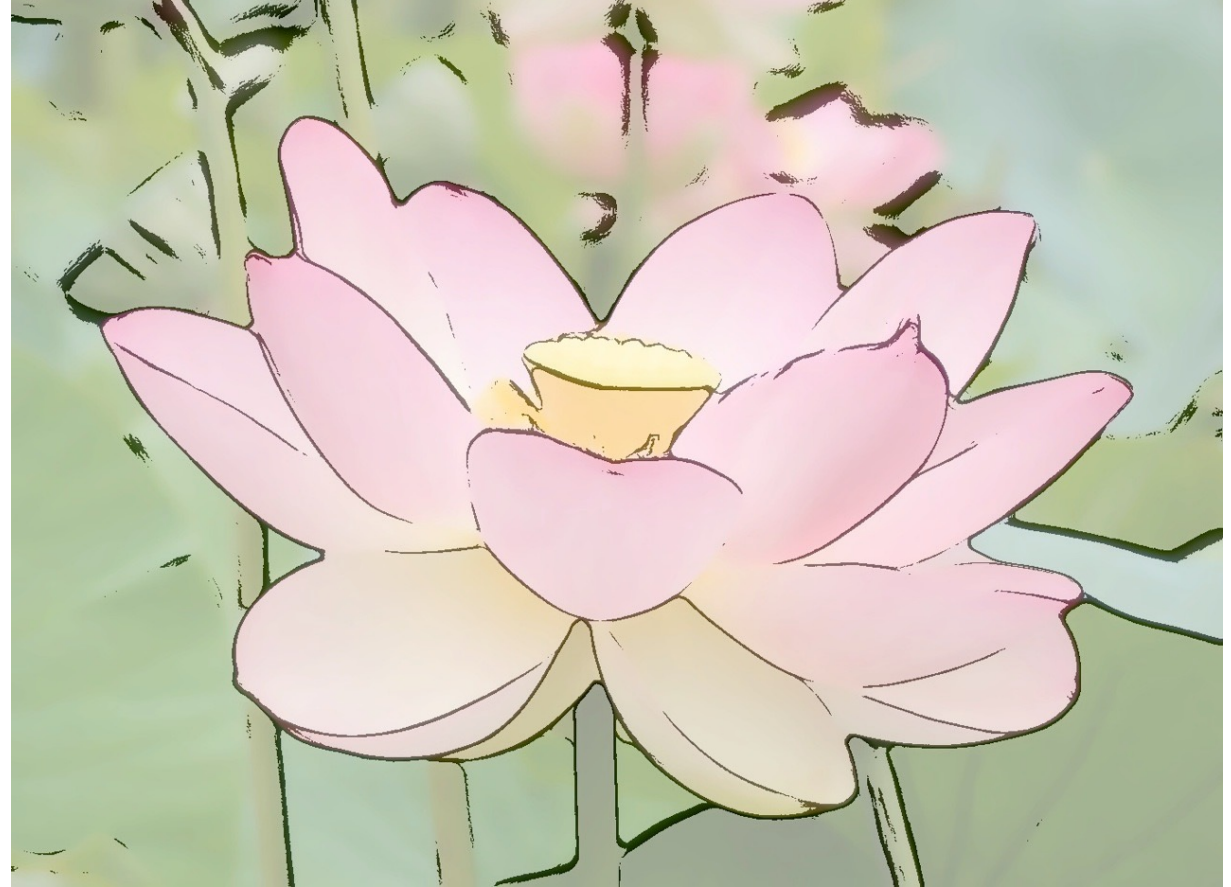
How would you create this effect?

Cartoonization
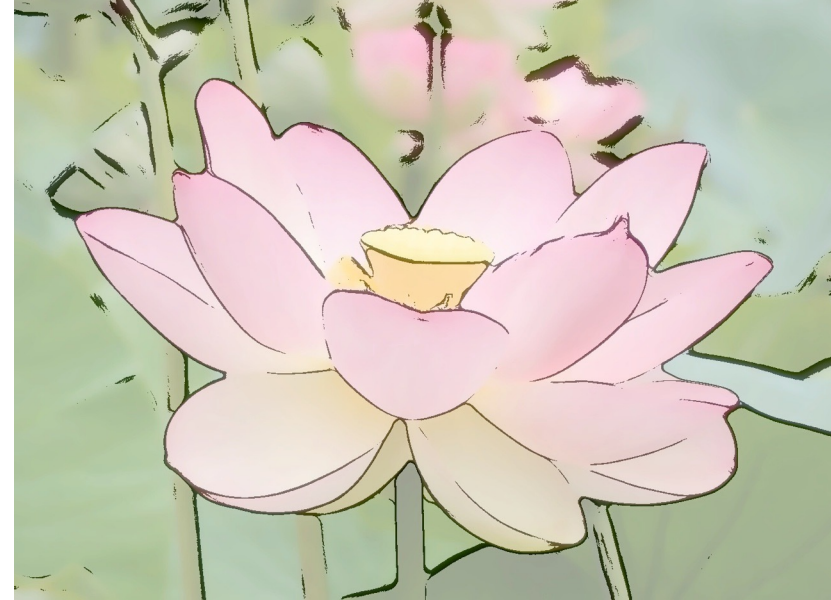
edges from bilaterally filtered image   bilaterally filtered image   cartoon rendition

 + =
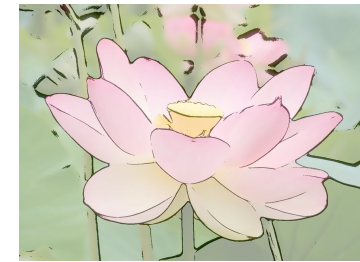
Note: image cartoonization and abstraction are very active research areas.

Is the bilateral filter:

Linear?

Shift-invariant?

Is the bilateral filter:

Linear?

- No.

Shift-invariant?

- No.

Does this have any bad implications?

The bilateral grid

Real-time Edge-Aware Image Processing with the Bilateral Grid

Jiawen Chen   Sylvain Paris   Frédo Durand

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology

Figure 1: *The bilateral grid enables edge-aware image manipulations such as local tone mapping on high resolution images in real time. This 15 megapixel HDR panorama was tone mapped and locally refined using an edge-aware brush at 50 Hz. The inset shows the original input. The process used about 1 MB of texture memory.*

Data structure for fast edge-aware image processing.

Flash/no-flash photography
via bilateral filtering

Red Eye

Unflattering Lighting

Motion Blur

Noise
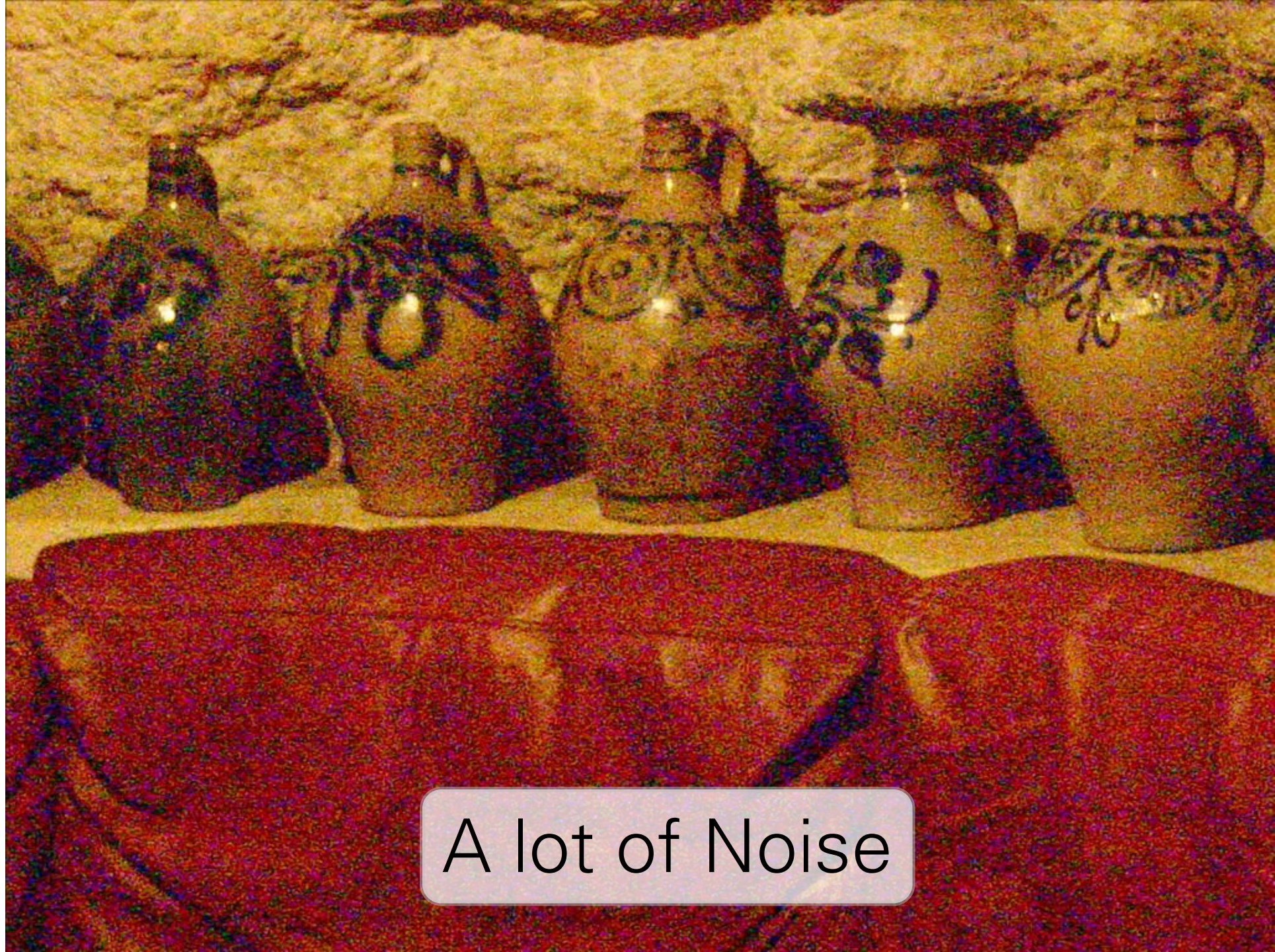
A lot of Noise

Ruined Ambiance

Flash
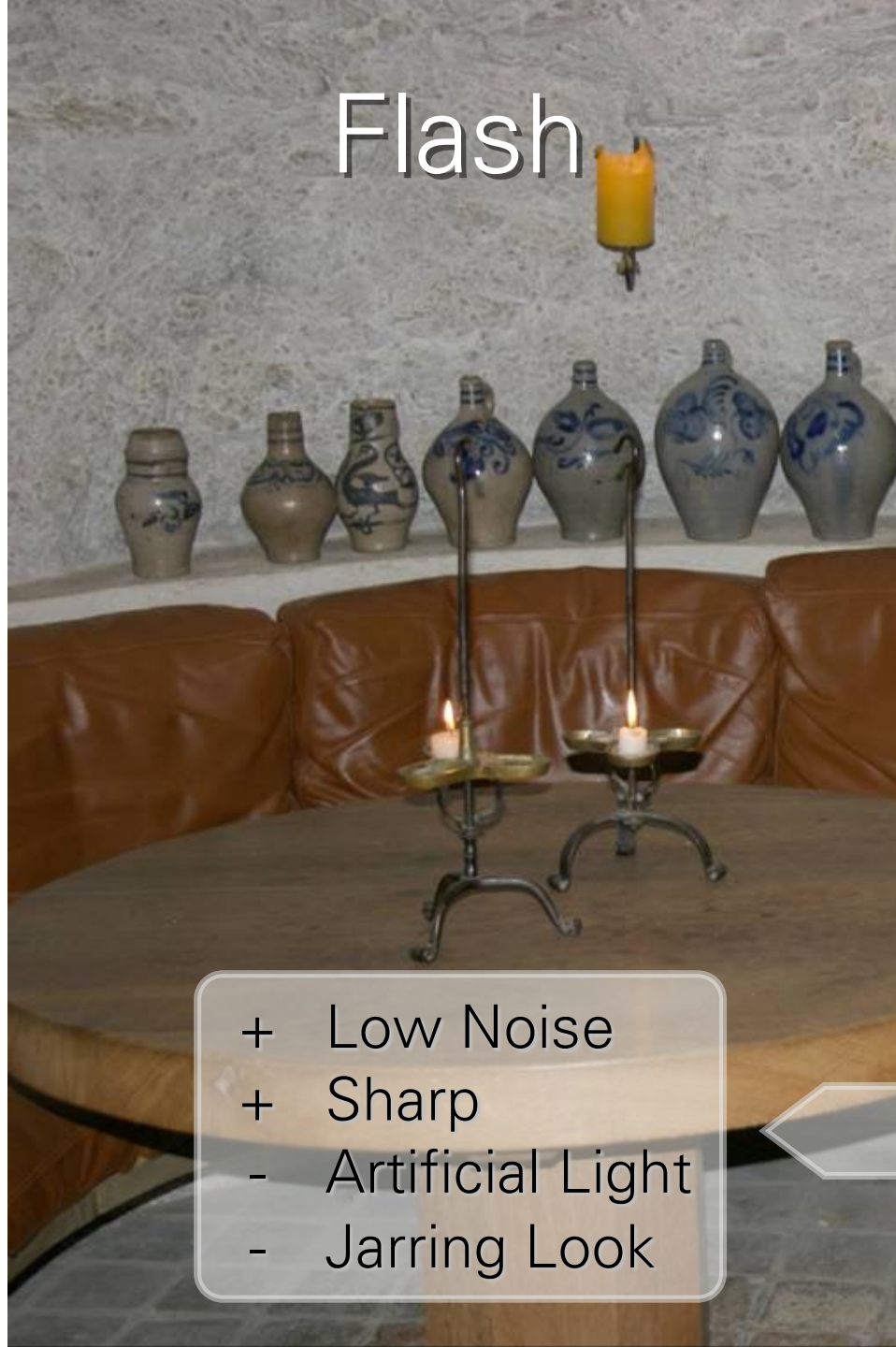
No-Flash

+ Low Noise
+ Sharp
- Artificial Light
- Jarring Look
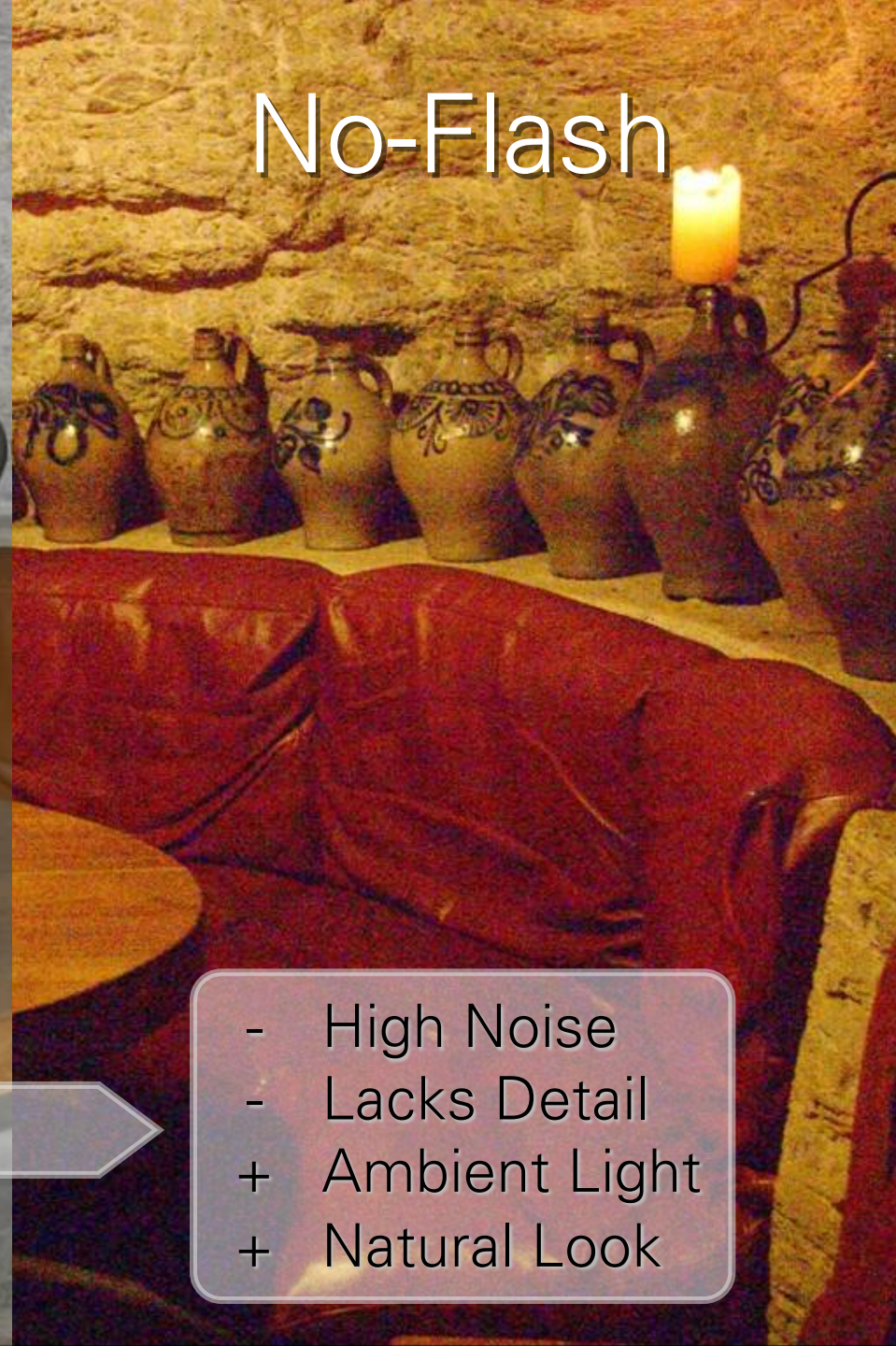
- High Noise
- Lacks Detail
+ Ambient Light
+ Natural Look

Image acquisition

1 — Lock Focus & Aperture

time

Image acquisition

1 — Lock Focus & Aperture

2 — No-Flash Image

time

1/30 s
ISO 3200

Image acquisition

1. Lock Focus & Aperture

2. No-Flash Image

3. Flash Image

time

1/30 s
ISO 3200

1/125 s
ISO 200
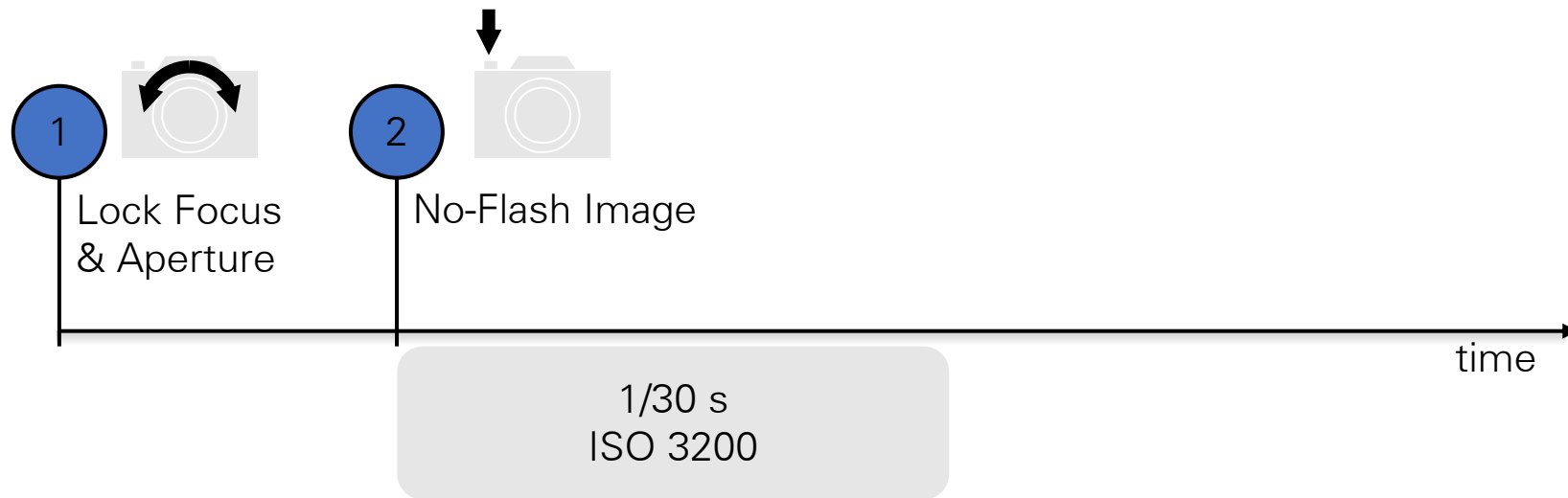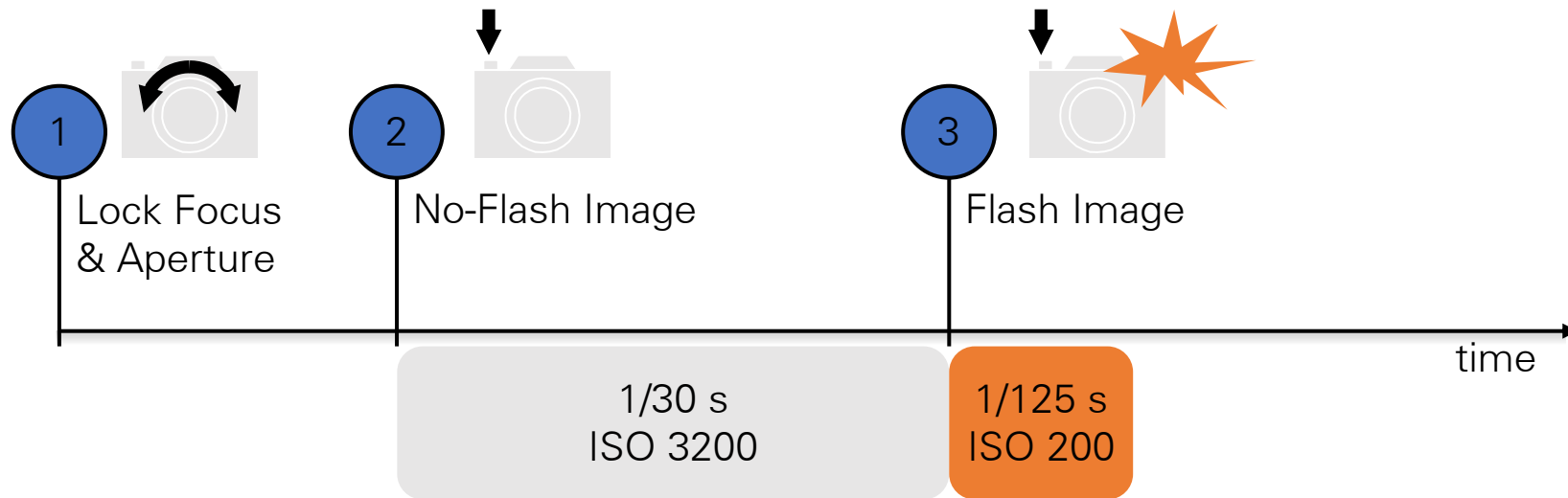
Denoising Result

No-Flash

Denoising Result

Key idea

Denoise the no-flash image while maintaining the edge structure of the flash image
- How would you do this using the image editing techniques we've learned about?

Joint bilateral filtering

Denoising with bilateral filtering

noisy input   bilateral filtering   median filtering

Denoising with bilateral filtering

spatial kernel

$$A_{p(col)}^{Base} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|) \; g_r\big(A_{p(col)} - A_{p'(col)}\big) A_{p'(col)}$$

intensity kernel

- However, results still have noise or blur (or both)

ambient

flash

Bilateral filter

Denoising with joint bilateral filtering

$$A_{p(col)}^{NR} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|) \, g_r(F_{p(col)} - F_{p'(col)}) A_{p'(col)}$$

- In the flash image there are many more details
- Use the flash image F to find edges

Denoising with joint bilateral filtering

$$A_{p(col)}^{NR} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|) \, g_r\left(F_{p(col)} - F_{p'(col)}\right) A_{p'(col)}$$

Bilateral filter
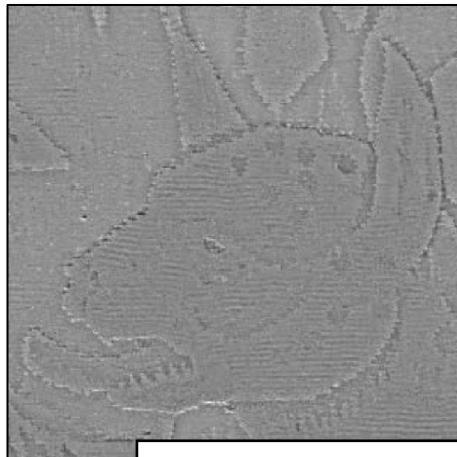
The difference

Joint Bilateral filter

Not all edges in the flash image are real

Can you think of any types of edges that may exist in the flash image but not the ambient one?

Not all edges in the flash image are real

shadows
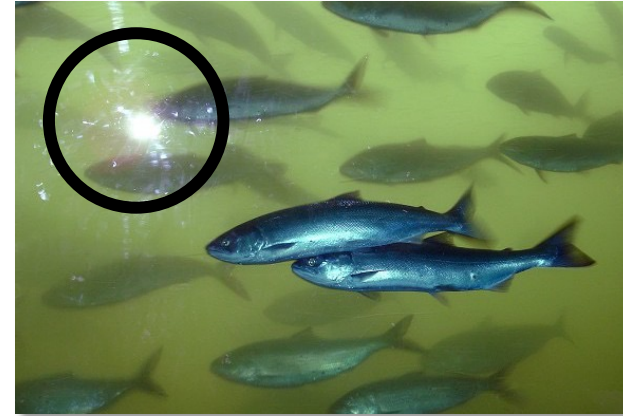
specularities

- May cause over- or under-blur in joint bilateral filter
- We need to eliminate their effect

Detecting shadows

- Observation: the pixels in the flash shadow should be similar to the ambient image.

- Not identical:

  1. Noise.

  2. Inter-reflected flash.

- Compute a shadow mask. $\quad M^{Shad} = \begin{cases} 1 \text{ when } F^{Lin} - A^{Lin} \leq \tau_{Shad} \\ 0 \text{ otherwise.} \end{cases}$

- Take pixel p if $\quad F^{Lin}_{p(col)} - A^{Lin}_{p(col)} \leq \tau_{Shadow}$

- τ_{Shadow} is manually adjusted

- Mask is smoothed and dilated

Detecting specularities

- Take pixels where sensor input is close to maximum (very bright).

    - Over fixed threshold τ_{Spec}

- Create a specularity mask.

- Also smoothed.

- M – the combination of shadow and specularity masks:

    Where $M_p=1$, we use A^{Base}. For other pixels we use A^{NR}.

Detail transfer

- Denoising cannot add details missing in the ambient image
- Exist in flash image because of high SNR
- We use a quotient image:

$$F_{p(col)}^{Detail} = \frac{F_{p(col)} + \varepsilon}{F_{p(col)}^{Base} + \varepsilon}$$

Reduces the effect of noise in F

Bilateral filtered

- Multiply with A^{NR} to add the details
- Masked in the same way

Why does this quotient image make sense for detail?

Detail transfer

- Denoising cannot add details missing in the ambient image
- Exist in flash image because of high SNR
- We use a quotient image:

$$F_{p(col)}^{Detail} = \frac{F_{p(col)} + \varepsilon}{F_{p(col)}^{Base} + \varepsilon}$$

Reduces the effect of noise in F

Detail transfer

Orig. (top)   Detail Transfer (bottom)   Detail Transfer w/o Mask   Shadow and Specularity Mask   Detail Transfer w/ Mask.
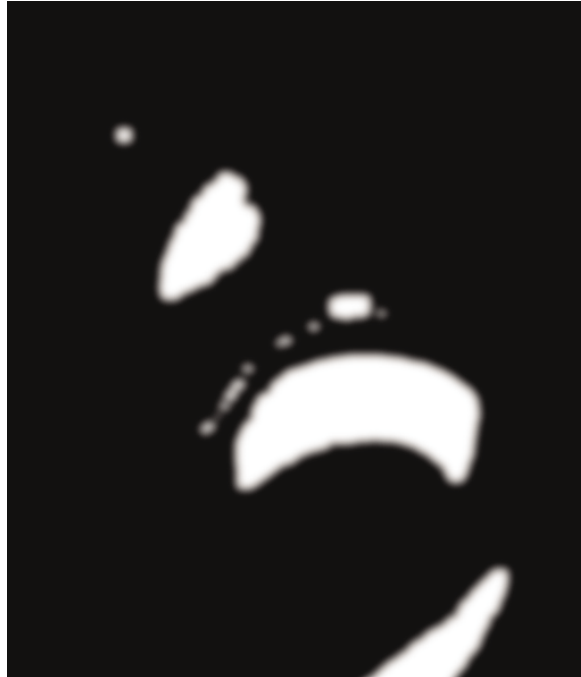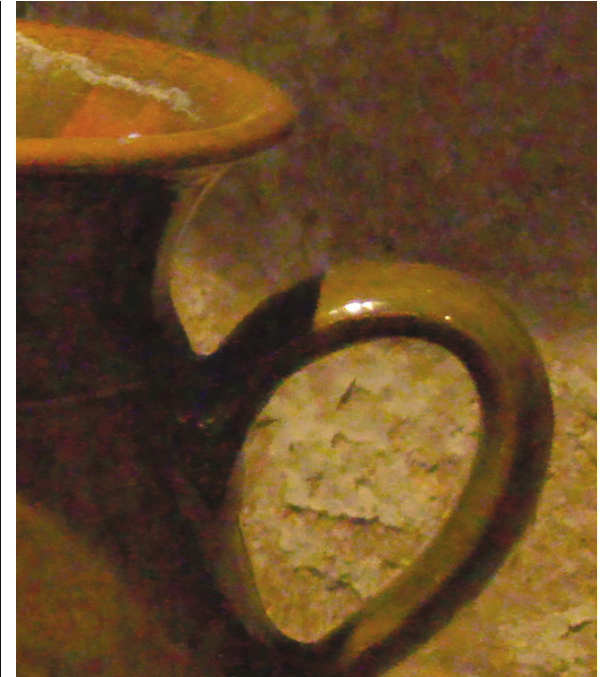
Flash   No-Flash   Detail Transfer w/ Denoising

Figure 8: (top row) The flash image does not contain true detail information in shadows and specular regions. When we naively apply our denoising and

Full pipeline
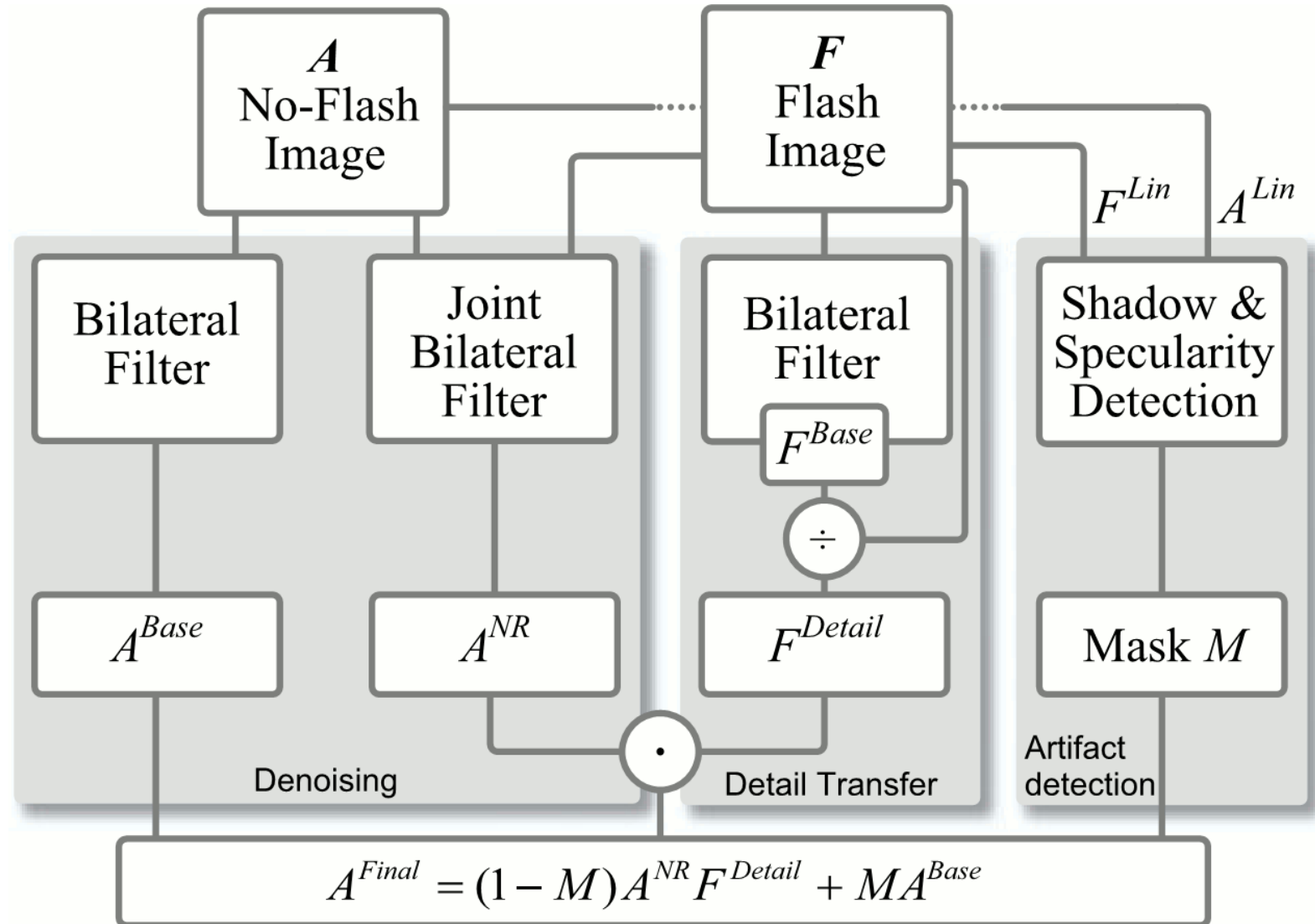
Demonstration

ambient-only

joint bilateral and detail transfer

Flash

No-Flash

No-Flash

Result

Flash

No-Flash

No-Flash

Result

Flash

No-Flash

Flash

No-Flash

Result

Joint bilateral filtering
other applications

Edge-aware depth denoising

$$A_{p(col)} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|)$$

$$g_r\big(F_{p(col)} - F_{p'(col)}\big) A_{p'(col)}$$

Use joint bilateral filtering, with the input image as guide.

One of two input images
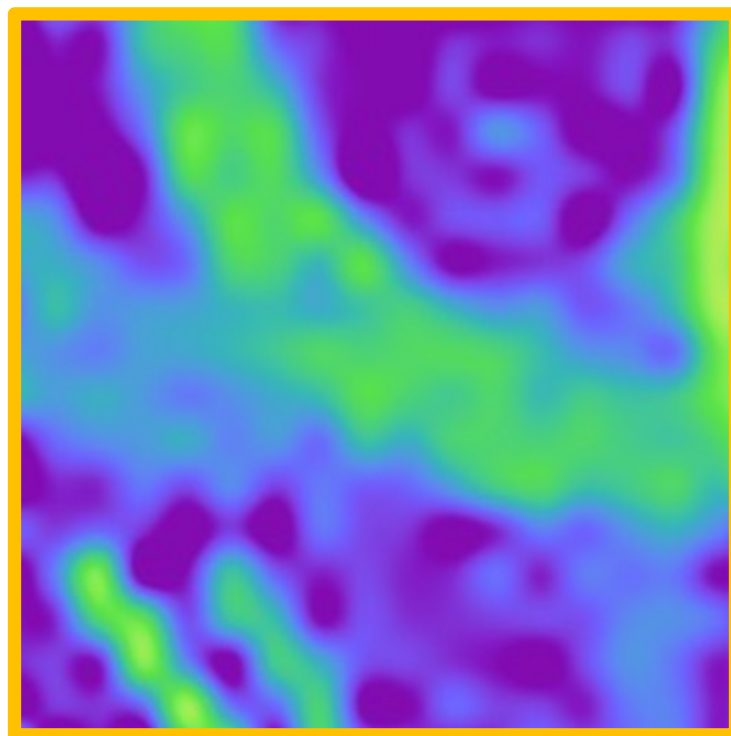
Depth from disparity
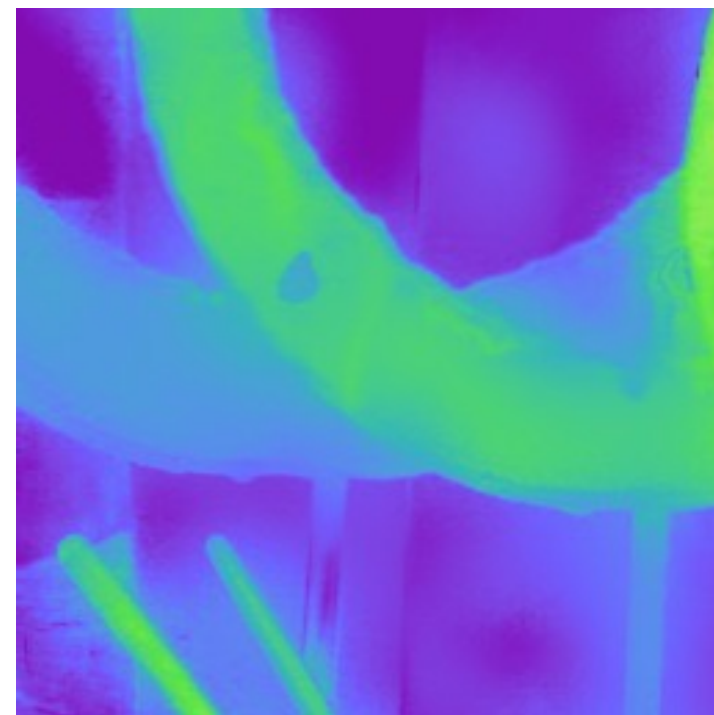
Guided filtering

Other applications of joint bilateral filtering

Deep Bilateral Learning for Real-Time Image Enhancement

MICHAËL GHARBI, MIT CSAIL
JIAWEN CHEN, Google Research
JONATHAN T. BARRON, Google Research
SAMUEL W. HASINOFF, Google Research
FRÉDO DURAND, MIT CSAIL / Inria, Université Côte d'Azur
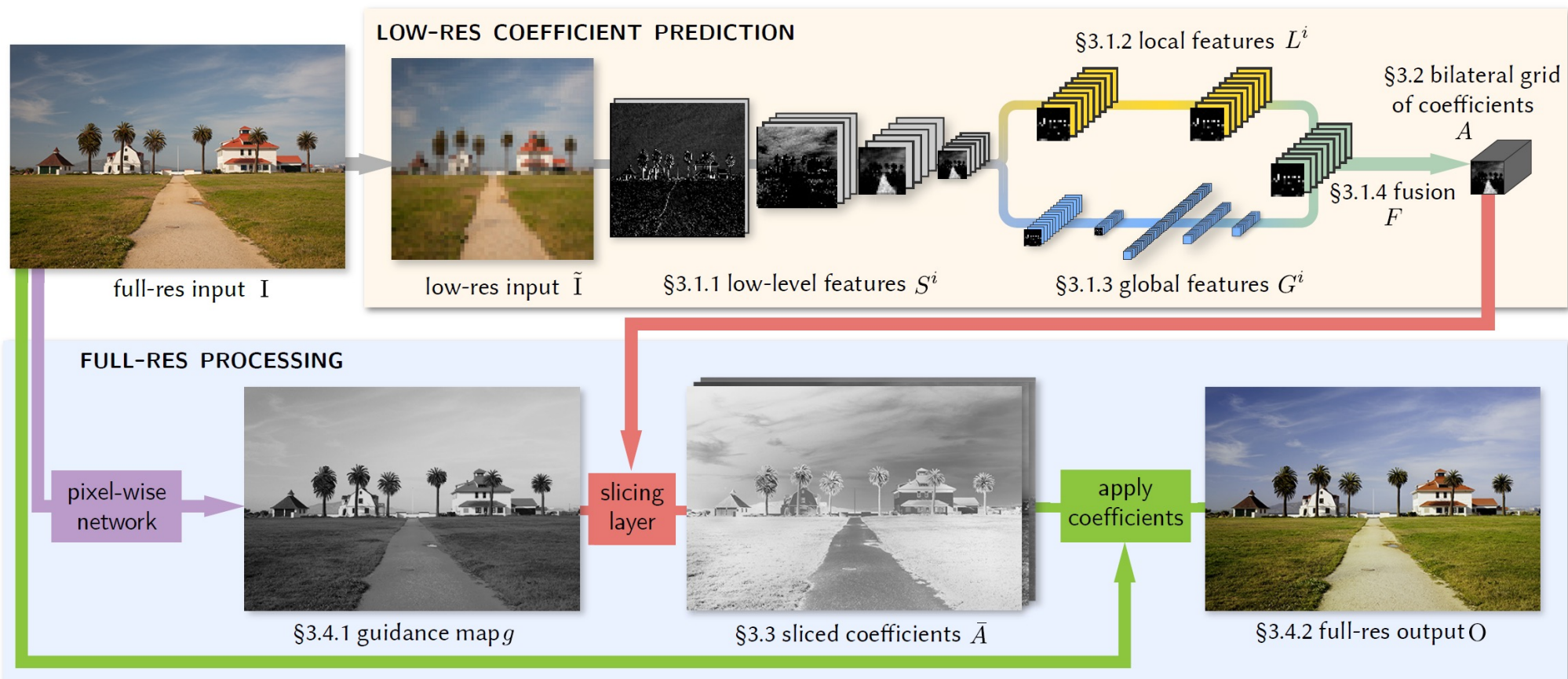
Tonemapping
via bilateral filtering

Display the information

- Recall our HDR class
- Match limited contrast of the medium while preserving details

low contrast

10^{-6} 10^6

Real world:

display

Photo/display:

10^{-6} 10^6

HDR image

high dynamic range

Display the information

- Recall our HDR class
- Match limited contrast of the medium while preserving details

low contrast

10^{-6} 10^6

Real world:

display

10^{-6} 10^6

Photo/display:

HDR image

high dynamic range
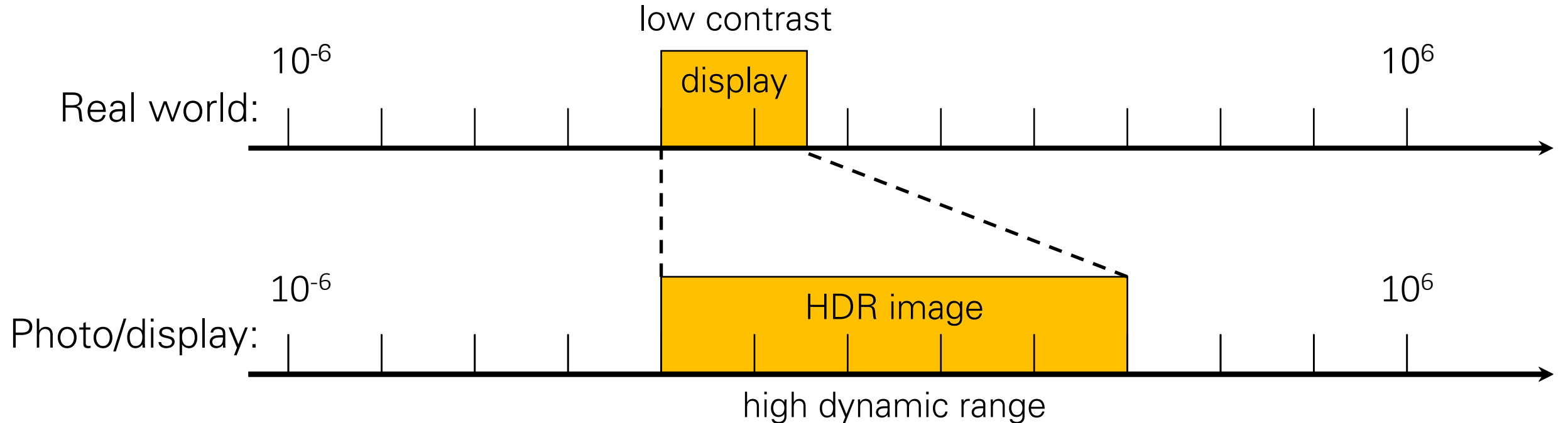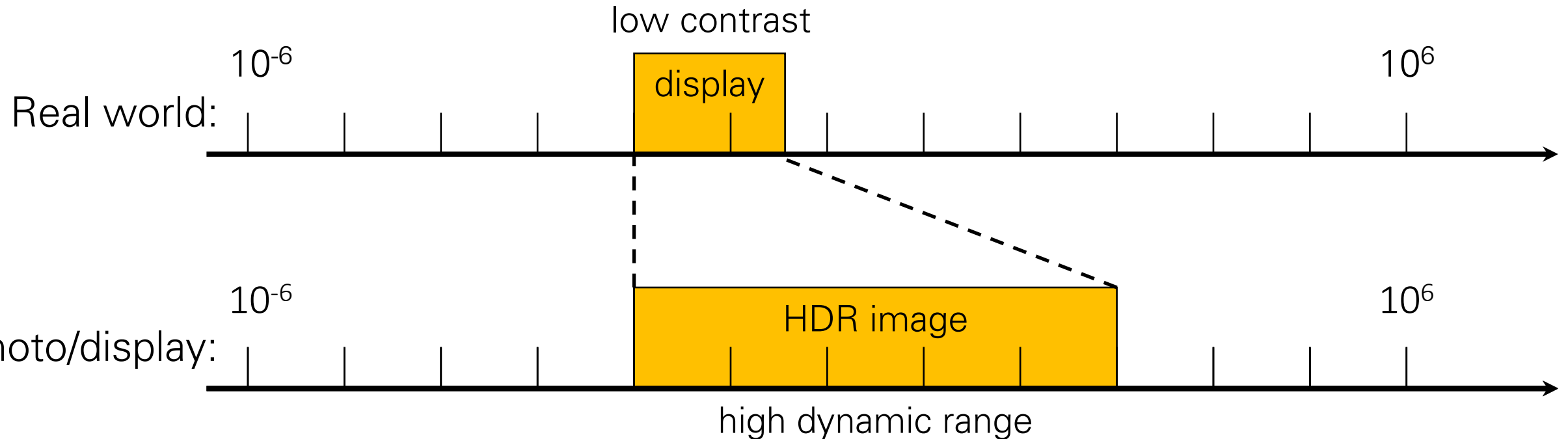
Tonemapping

Tonemapping

- Called tone mapping operators

- Two general categories:

  - Global (spatially invariant)

  - Local (spatially varying)

Tone mapping for very HDR scenes

- sun overexposed
- foreground too dark

Tone mapping for very HDR scenes

- Scene has >100,000:1 dynamic range, JPEG has 255:1

- How can we compress the scene's dynamic range?

Tone mapping for very HDR scenes

- Scene has >100,000:1 dynamic range, JPEG has 255:1

- How can we compress the scene's dynamic range?

- Scale linearly?
  - If we scaled linearly from 100,000:1 to 255:1, everything but the sun would be black!

input

Gamma

- gamma correction, applied independently on R, G, B:

$$I = I^\gamma$$

- global tonemapping

- colors are washed out

Tonemapping w/ Simple Gamma

intensity

- gamma in intensity only!

- Intensity details lost

color

Gamma on intensity
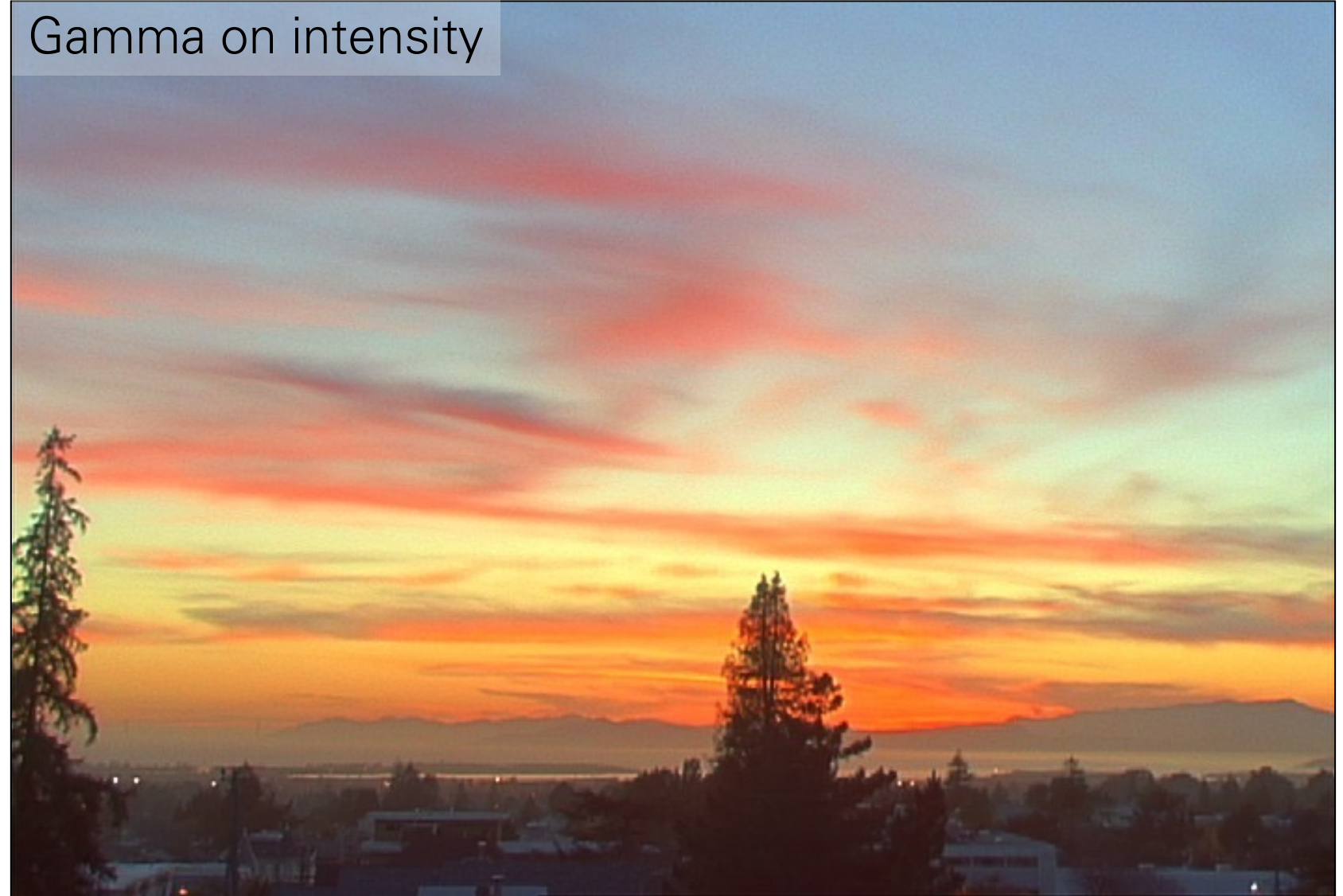
Oppenheim 1968, Chiu et al. 1993

- Reduce contrast of low-frequencies, preserve high frequencies

low-freq.

high-freq.

color
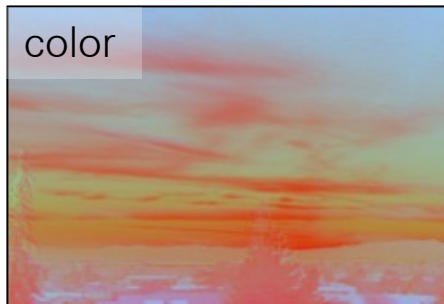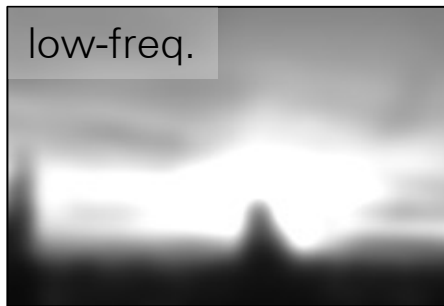
Reduce low frequency

The halo nightmare

- For strong edges; because they contain high frequency

low-freq.

high-freq.

color
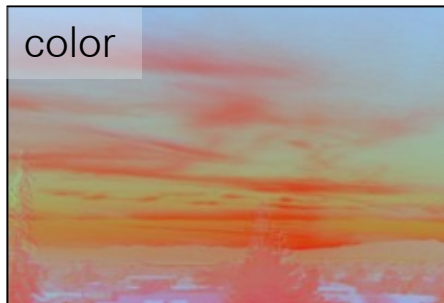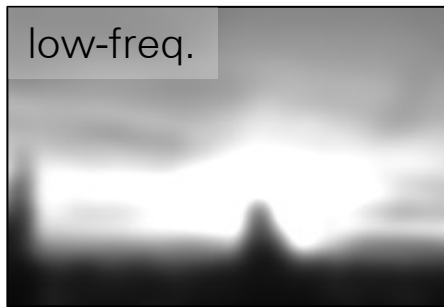
Reduce low frequency

The halo nightmare

- Similar to unsharp mask of luminance in log domain
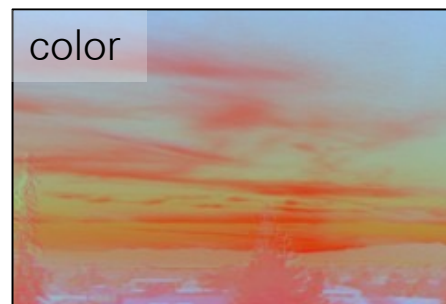

low-freq.

high-freq.

color
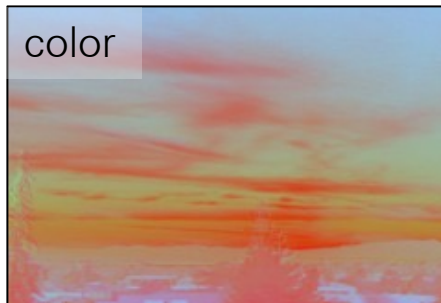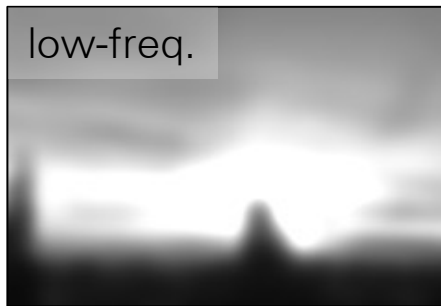
Reduce low frequency

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

- Don't blur across edges, decompose using bilateral filter
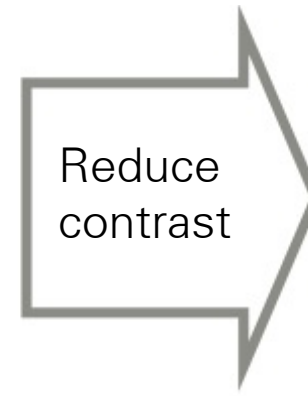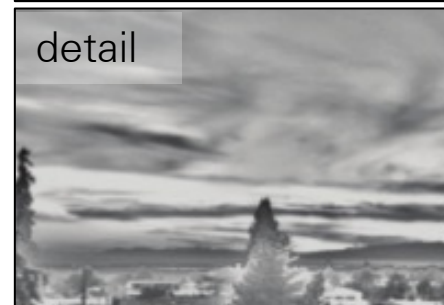
Large scale

large scale

Detail

detail

Color

color

Reduce large scale

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

Input HDR
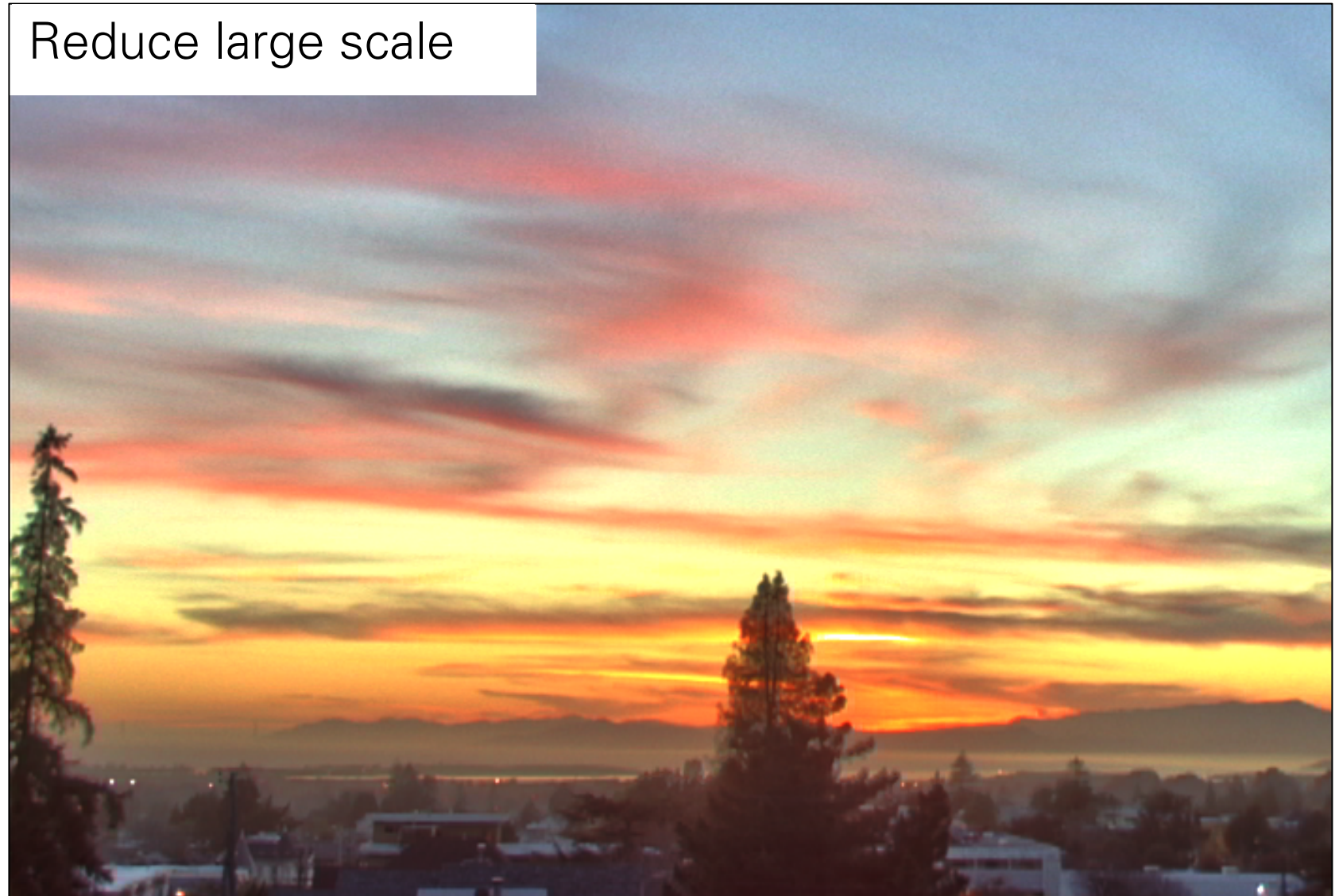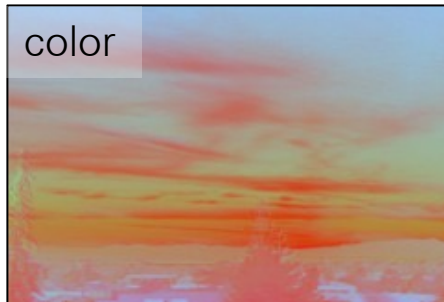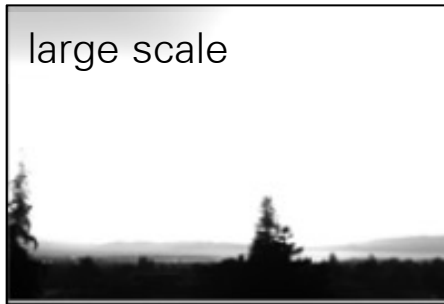
Contrast too high!

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

Input HDR
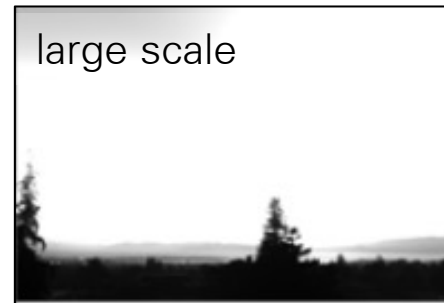
intensity

Intensity
= 0.3R+0.6G+0.1B

color
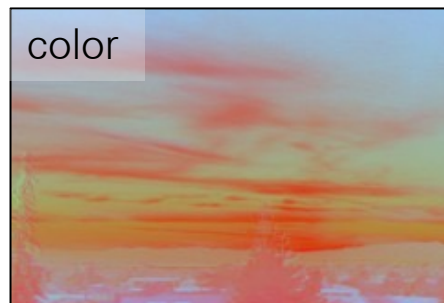
R'=R/intensity
G'=G/intensity
B'=B/intensity

Important to use ratios
(makes it luminance
invariant)

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

Input HDR

intensity

Bilateral
Filter

in log

large scale

color
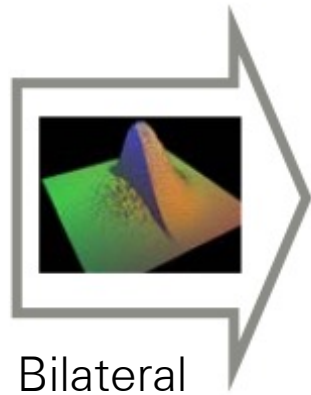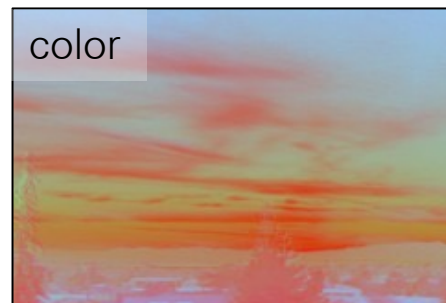
Spatial sigma: 2-5% image size
Range sigma: 0.4 (in log 10)

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

Input HDR

intensity
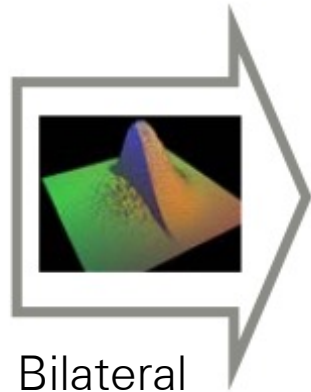
Bilateral
Filter

in log
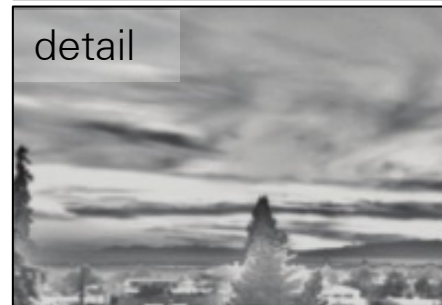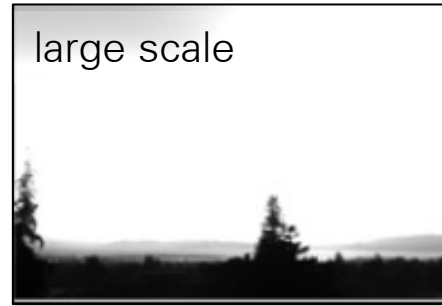
large scale

detail

detail = log intensity - large scale
(residual)
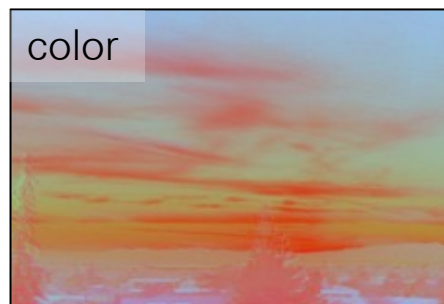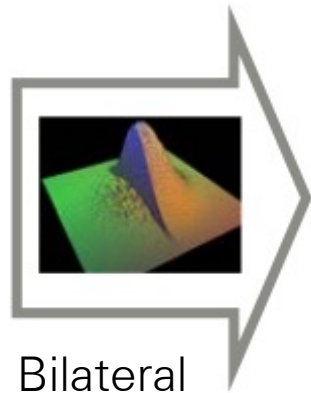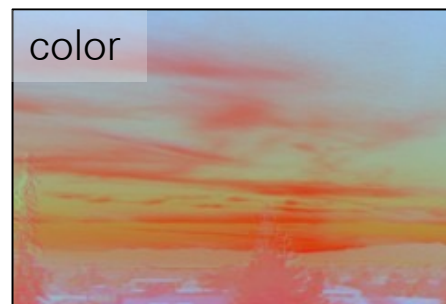
color

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

Input HDR

intensity

Bilateral
Filter

in log

large scale

detail

Reduce
contrast

large scale

color

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

Tonemapping w/ Bilateral Filter (Durand and Dorsey, 2002)

Input HDR

intensity

Bilateral Filter

in log

large scale

detail

Reduce contrast

Preserve!

Large-scale

Detail

Color

output

large scale

detail

color

color
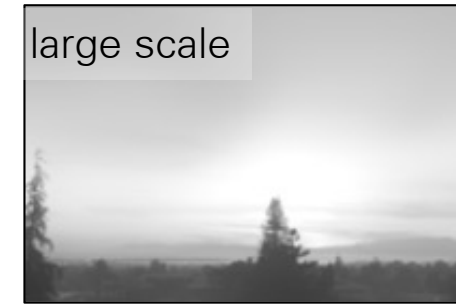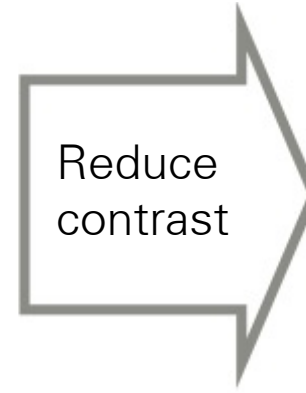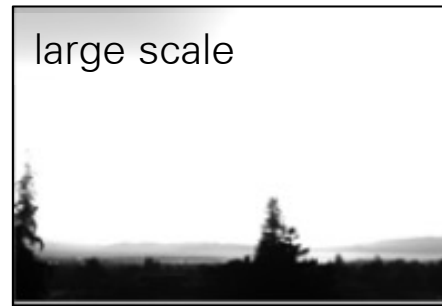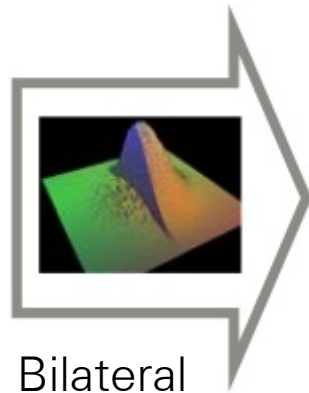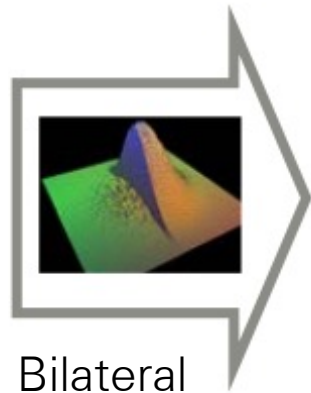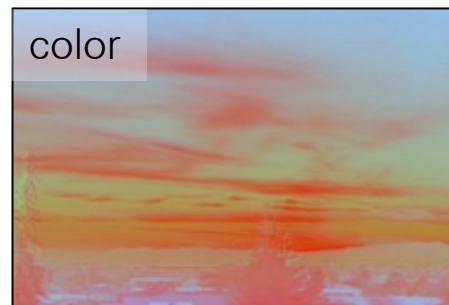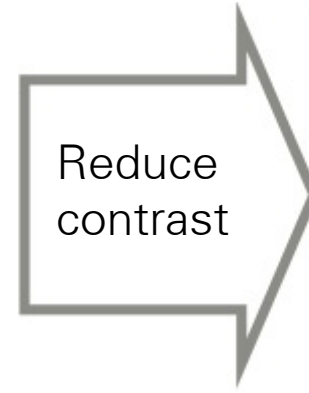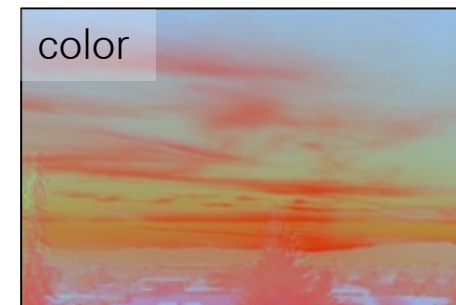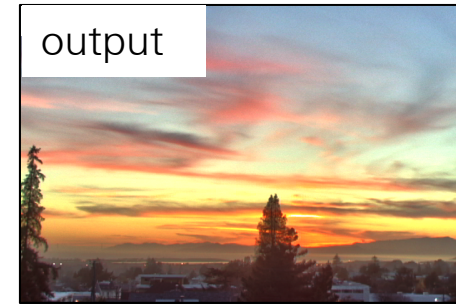
Log domain

- Very important to work in the log domain

- Recall: humans are sensitive to multiplicative contrast

- With log domain, our notion of "strong edge" always corresponds to the same contrast

Scale decomposition in log domain

- inLog = $\log_{10}$(intensity)

- inLogLarge = bilateralFilter(inLog)

- inLogDetail = inLog - inLogLarge

- hence:

  - inLog = inLogDetail + inLogLarge, or

  - intensity = $10^{inLogDetail} * 10^{inLogLarge}$

- Now manipulate large-scale and detail separately

Contrast reduction in log domain

outLog = inLogDetail + k * (inLogLarge – max(inLogLarge))

- Normalize so that the biggest value is 0 in log

- Set target large-scale contrast (e.g. targetRange = $\log_{10}(100)$)

  - i.e. in **linear** output, we want 1:100 contrast for large scale

- Compute range of input's large-scale layer:

  - largeRange = max(inLogLarge) – min(inLogLarge)

- Scale factor k = targetRange / largeRange

multiplication in log
=
γ exponent in linear

Contrast reduction in log domain

$$\text{outLog} = \textcolor{red}{\text{detailAmp}}*\text{inLogDetail} + k*(\text{inLogLarge} - \max(\text{inLogLarge}))$$

- Normalize so that the biggest value is 0 in log

- Set target large-scale contrast (e.g. targetRange = $\log_{10}(100)$)
  - i.e. in **linear** output, we want 1:100 contrast for large scale

- Compute range of input's large-scale layer:
  - largeRange = max(inLogLarge) − min(inLogLarge)

- Scale factor k = targetRange / largeRange

- **Optional:** amplify detail by detailAmp

Contrast reduction in log domain

$$\text{outLog} = \textcolor{red}{\text{detailAmp}}*\text{inLogDetail} + k*(\text{inLogLarge} - \max(\text{inLogLarge}))$$

- outIntensity = 10^{outLog}

- Recall that R′,G′,B′ is the intensity-normalized RGB color

  - outR=outIntensity * R′

  - outG=outIntensity * G′

  - outB=outIntensity * B′

What matters

- Spatial sigma: not very important

- Range sigma: quite important

- Use of the log domain for range: **critical**

  - Because HDR and because perception sensitive to multiplicative contrast

Modern edge-aware filtering: domain transform

Domain Transform for Edge-Aware Image and Video Processing

Eduardo S. L. Gastal*   Manuel M. Oliveira[†]
Instituto de Informática – UFRGS
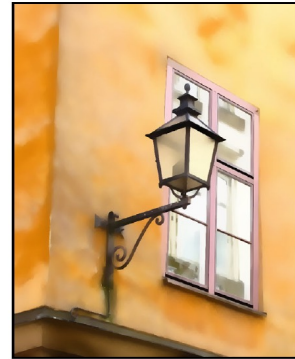
(a) *Photograph*   **(b)** *Edge-aware smoothing*   **(c)** *Detail enhancement*

(d) *Stylization*   **(e)** *Recoloring*   **(f)** *Pencil drawing*   **(g)** *Depth-of-field*
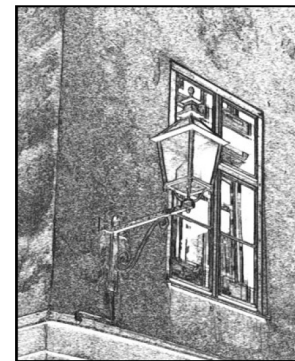
Lots of great examples at: https://www.inf.ufrgs.br/~eslgastal/DomainTransform/

Modern edge-aware filtering: guided filter

Guided Image Filtering

Kaiming He, *Member*, *IEEE*, Jian Sun, *Member*, *IEEE*, and Xiaoou Tang, *Fellow*, *IEEE*

filtering input p

spatial kernel $G_s(x_i\text{-}x_j)$

range kernel $G_r(I_i\text{-}I_j)$

bilateral kernel G_sG_r

filtering output q

guide I

$q_i = p_i - n_i$

filtering input p

filtering output q

guide I

$q_i = aI_i + b$

Guided Image Filtering

filtering input p

$q_i = p_i - n_i$

n_i : noise / texture

filtering output q
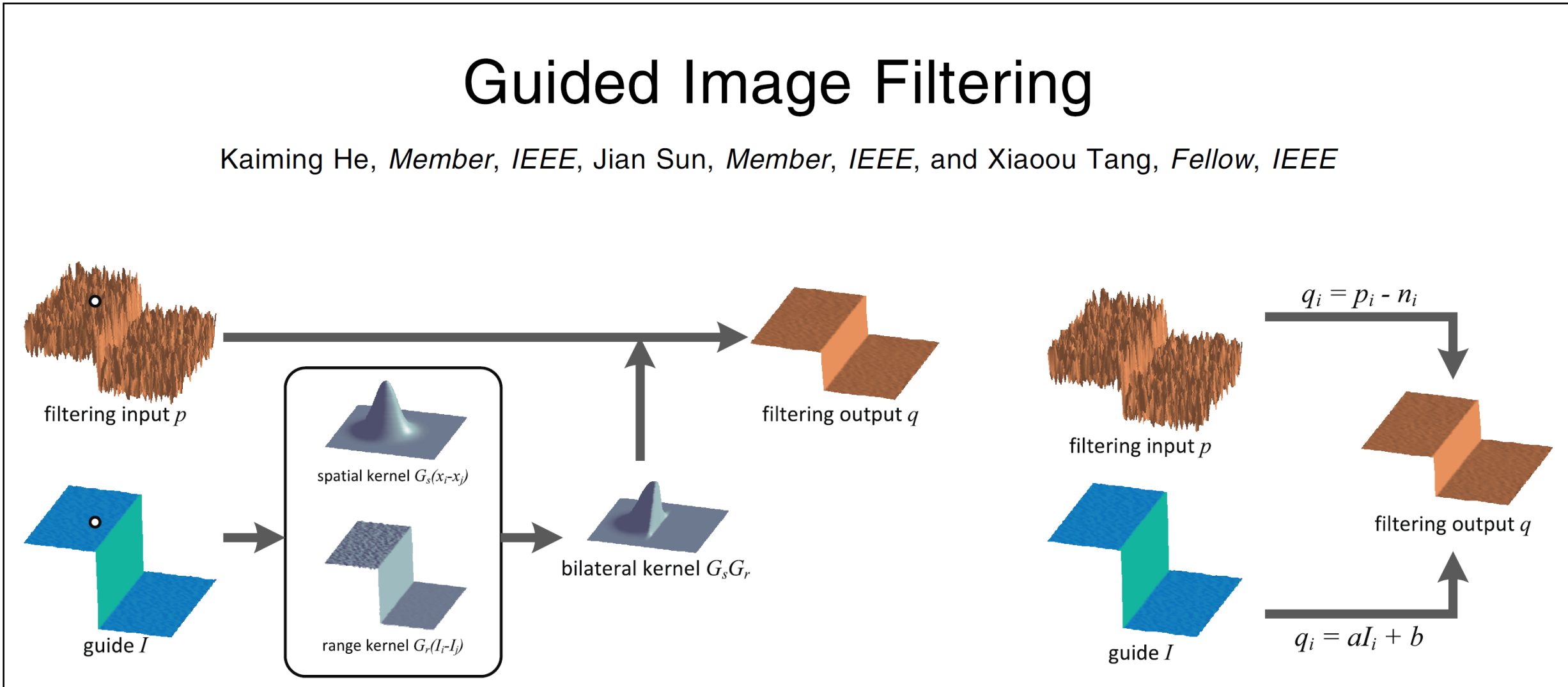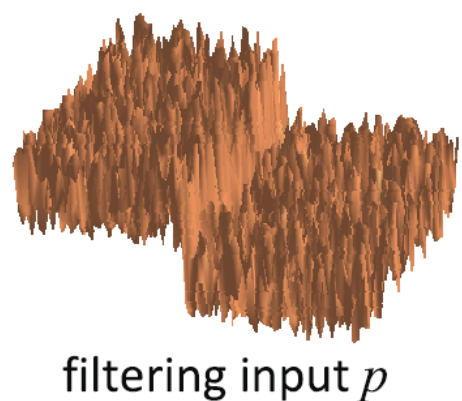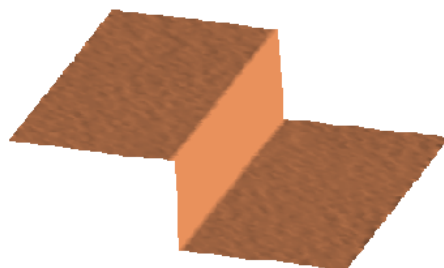
guide I

$\nabla q_i = a \nabla I_i$

$q_i = aI_i + b$
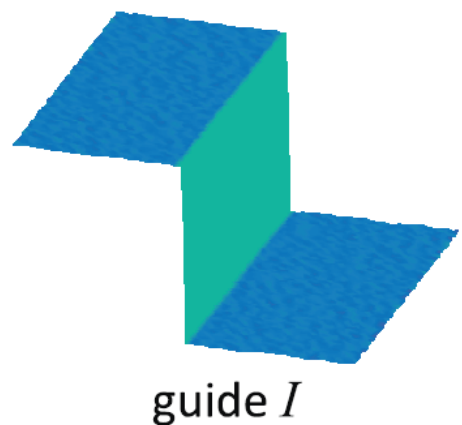
Bilateral/joint bilateral filter does not have this linear model

minimize $\quad E(a_k, b_k) = \sum_{i \in \omega_k} \left(\left(a_k I_i + b_k - p_i \right)^2 + \epsilon a_k^2 \right)$

Linear regression

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon},$$

$$b_k = \bar{p}_k - a_k \mu_k.$$

Guided Image Filtering

- Extend to the entire image
  - In all local windows ω_k, compute the linear coefficients
  - Compute the average of $a_k I_i + b_k$ in all ω_k that covers pixel q_i

ω_2 q_i ω_1 ω_3 2r

Definition

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon},$$

$$b_k = \bar{p}_k - a_k \mu_k.$$

$$q_i = \frac{1}{|\omega|}\sum_{k|i\in\omega_k}(a_k I_i + b_k)$$
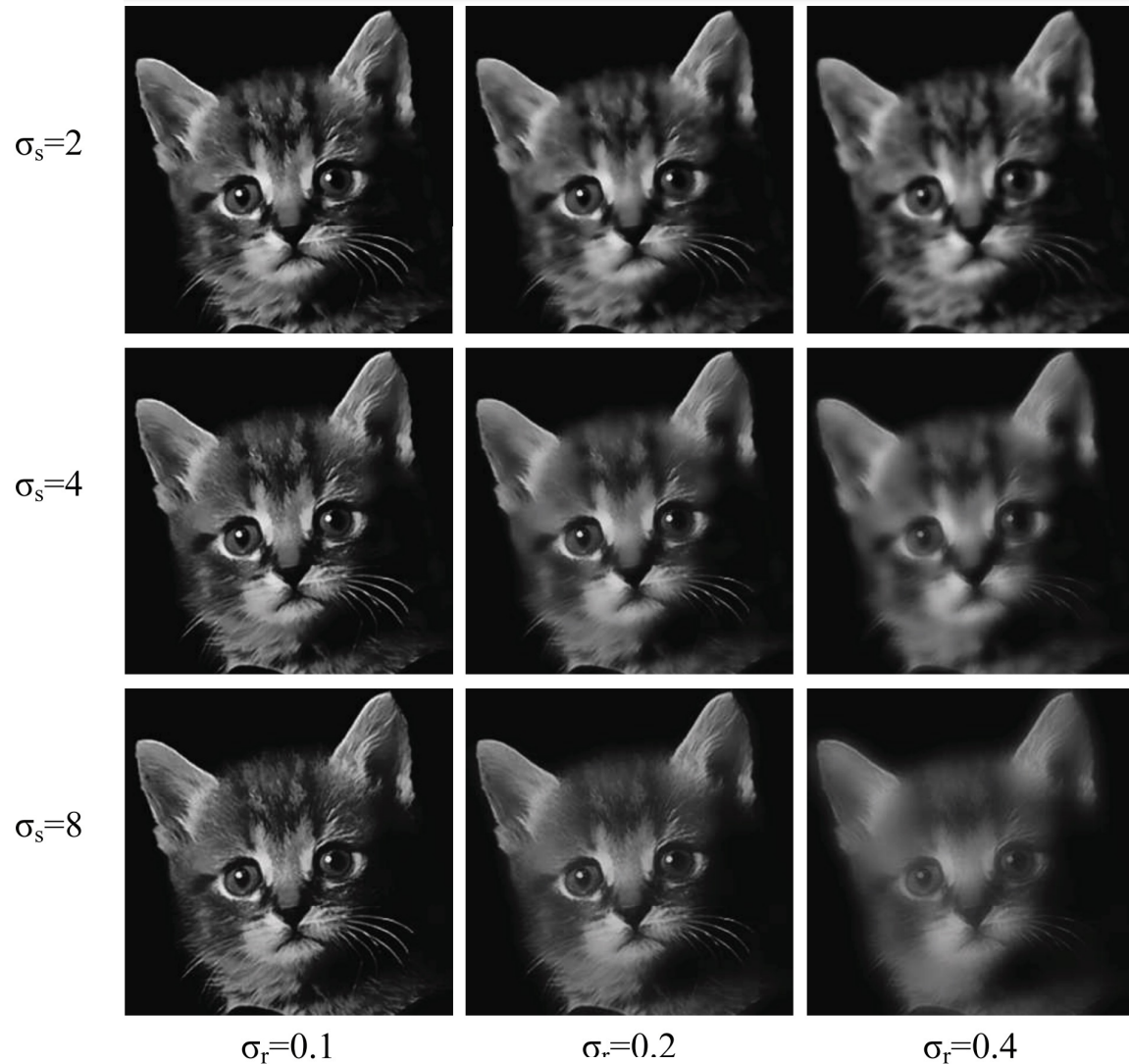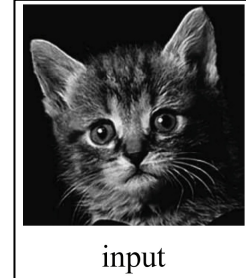
Algorithm 1. Guided Filter.
Input: filtering input image p, guidance image I, radius r, regularization ϵ
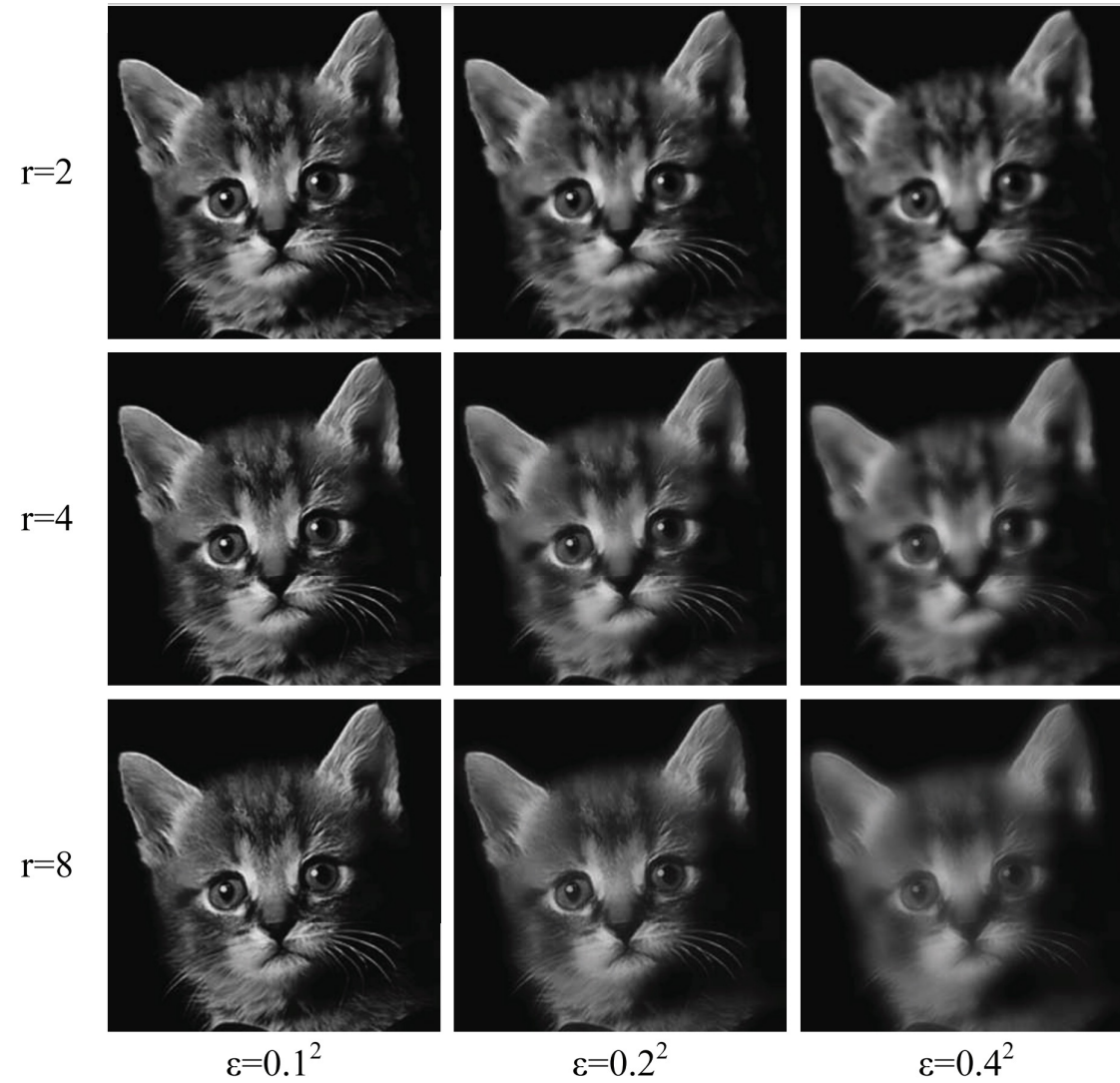**Output:** filtering output q.
1: $\text{mean}_I = f_{\text{mean}}(I)$
    $\text{mean}_p = f_{\text{mean}}(p)$
    $\text{corr}_I = f_{\text{mean}}(I.*I)$
    $\text{corr}_{Ip} = f_{\text{mean}}(I.*p)$
2: $\text{var}_I = \text{corr}_I - \text{mean}_I.*\text{mean}_I$
    $\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I.*\text{mean}_p$
3: $a = \text{cov}_{Ip}./(\text{var}_I + \epsilon)$
    $b = \text{mean}_p - a.*\text{mean}_I$
4: $\text{mean}_a = f_{\text{mean}}(a)$
    $\text{mean}_b = f_{\text{mean}}(b)$
5: $q = \text{mean}_a.*I + \text{mean}_b$
/* f_{mean} is a mean filter with a wide variety of O(N) time methods. */

Guided Image Filtering

- Edge-preserving filter
- O(1) time, fast, accurate
- Gradient preserving
- Parameters
  - Window radius r
  - Regularization ε

ω_2 q_i ω_1 ω_3 $2r$

Definition

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i\in\omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon},$$

$$b_k = \bar{p}_k - a_k \mu_k.$$

$$q_i = \frac{1}{|\omega|}\sum_{k|i\in\omega_k}(a_k I_i + b_k)$$

Algorithm 1. Guided Filter.
Input: filtering input image p, guidance image I, radius r, regularization ϵ
Output: filtering output q.
1: $\text{mean}_I = f_{\text{mean}}(I)$
$\text{mean}_p = f_{\text{mean}}(p)$
$\text{corr}_I = f_{\text{mean}}(I.*I)$
$\text{corr}_{Ip} = f_{\text{mean}}(I.*p)$
2: $\text{var}_I = \text{corr}_I - \text{mean}_I.*\text{mean}_I$
$\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I.*\text{mean}_p$
3: $a = \text{cov}_{Ip}./(\text{var}_I + \epsilon)$
$b = \text{mean}_p - a.*\text{mean}_I$
4: $\text{mean}_a = f_{\text{mean}}(a)$
$\text{mean}_b = f_{\text{mean}}(b)$
5: $q = \text{mean}_a.*I + \text{mean}_b$
$/^*\ f_{\text{mean}}$ is a mean filter with a wide variety of O(N) time methods. $^*/$

Example: Edge-Preserving Smoothing

input

$\sigma_s=2$

$\sigma_s=4$

$\sigma_s=8$

$\sigma_r=0.1$ $\sigma_r=0.2$ $\sigma_r=0.4$

Bilateral Filter

r=2

r=4

r=8

$\varepsilon=0.1^2$ $\varepsilon=0.2^2$ $\varepsilon=0.4^2$

Guided Filter

Example: Detail Enhancement

Original   Guided Filter   Bilateral Filter
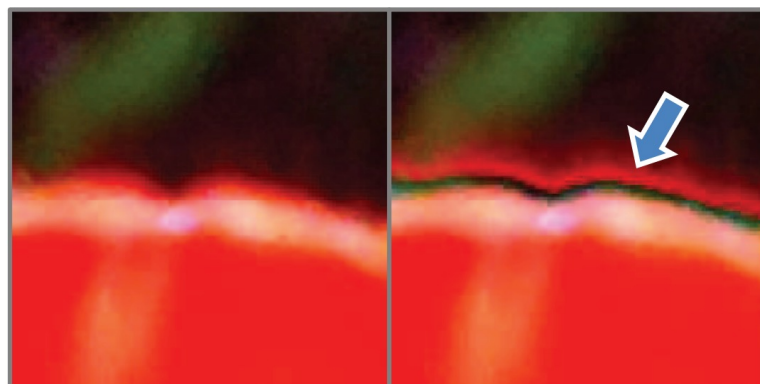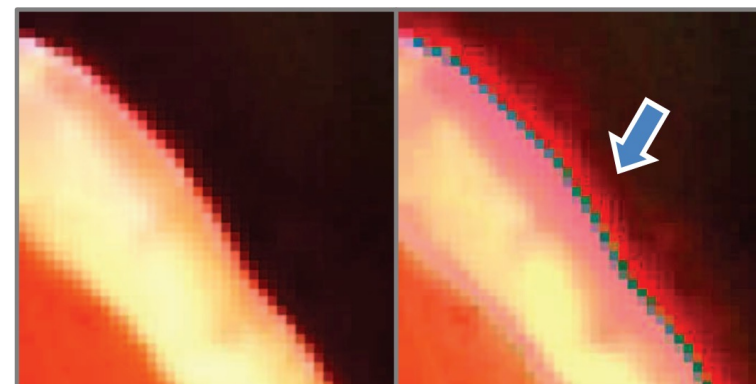
GF   BF   GF   BF

Example: Tonemapping
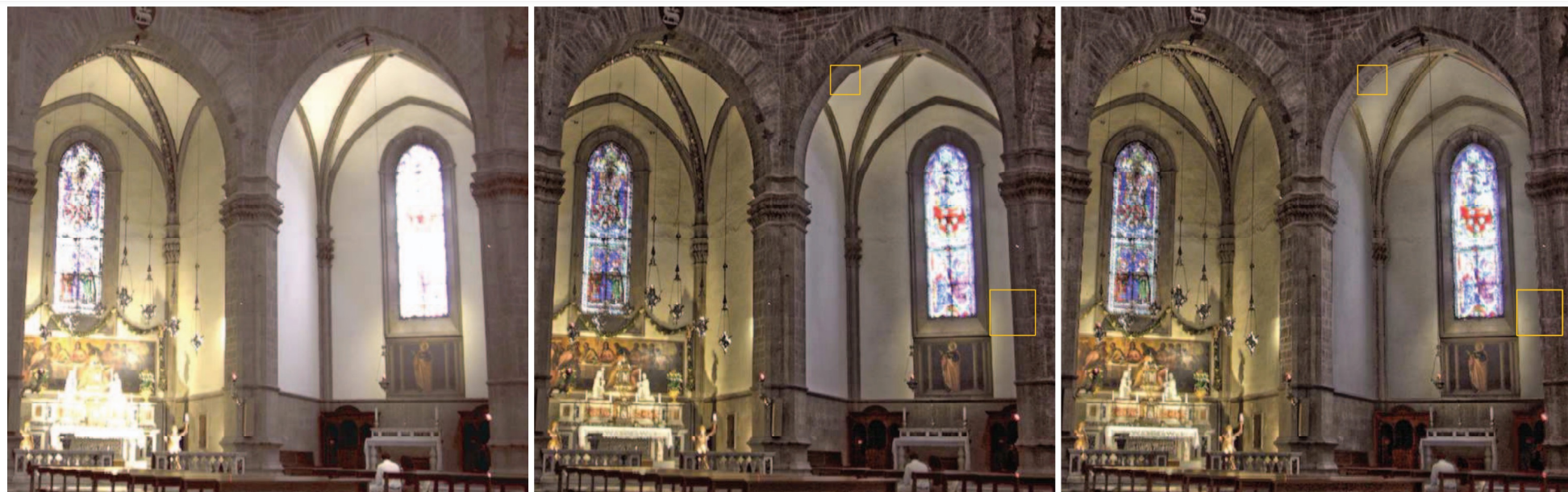
Original HDR   Guided Filter   Bilateral Filter
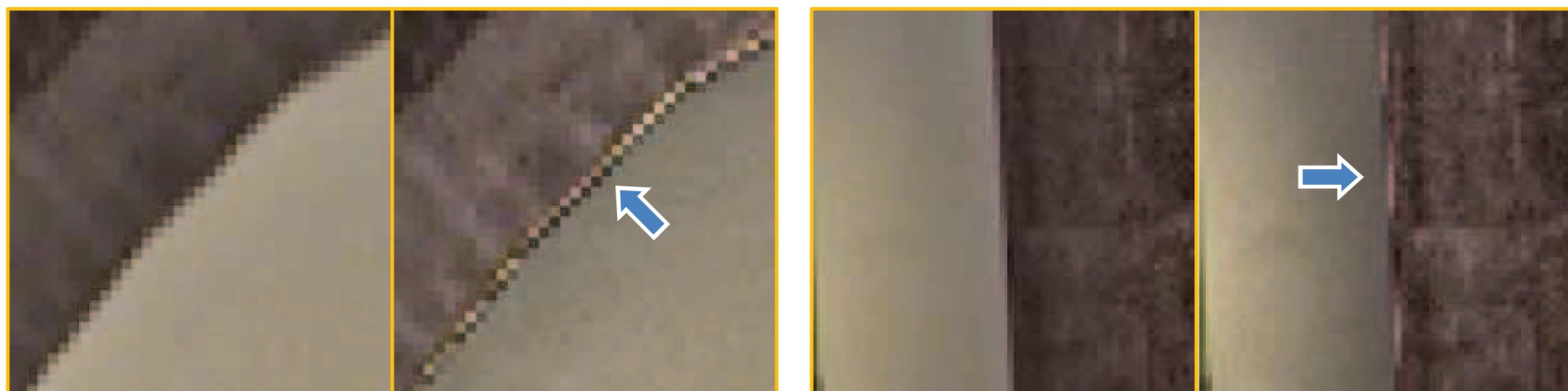
GF   BF   GF   BF

Example: Flash/No-Flash Photography

Guidance I

Guided Filter

Filter Input p

Joint Bilateral Filter

Non-Local Means Filter

Redundancy in natural images
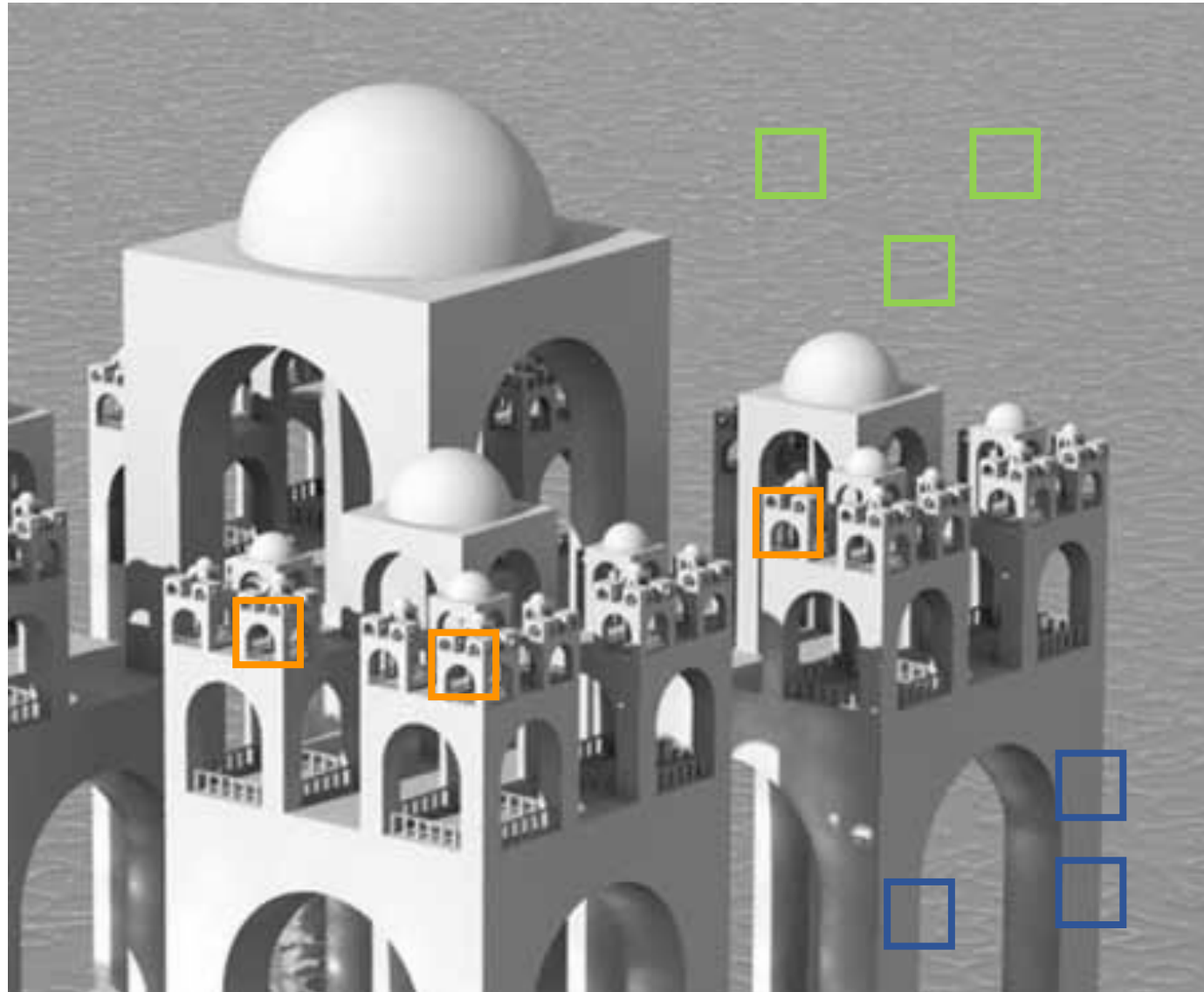
NL-Means Filter (Buades 2005)

- Same goals: 'Smooth within Similar Regions'

- <u>KEY INSIGHT</u>: Generalize, extend 'Similarity'
  - Bilateral:
    Averages neighbors with <u>similar intensities</u>;
  - NL-Means:
    Averages neighbors with <u>similar neighborhoods</u>!

NL-Means Method

- For each and
  every pixel **p:**
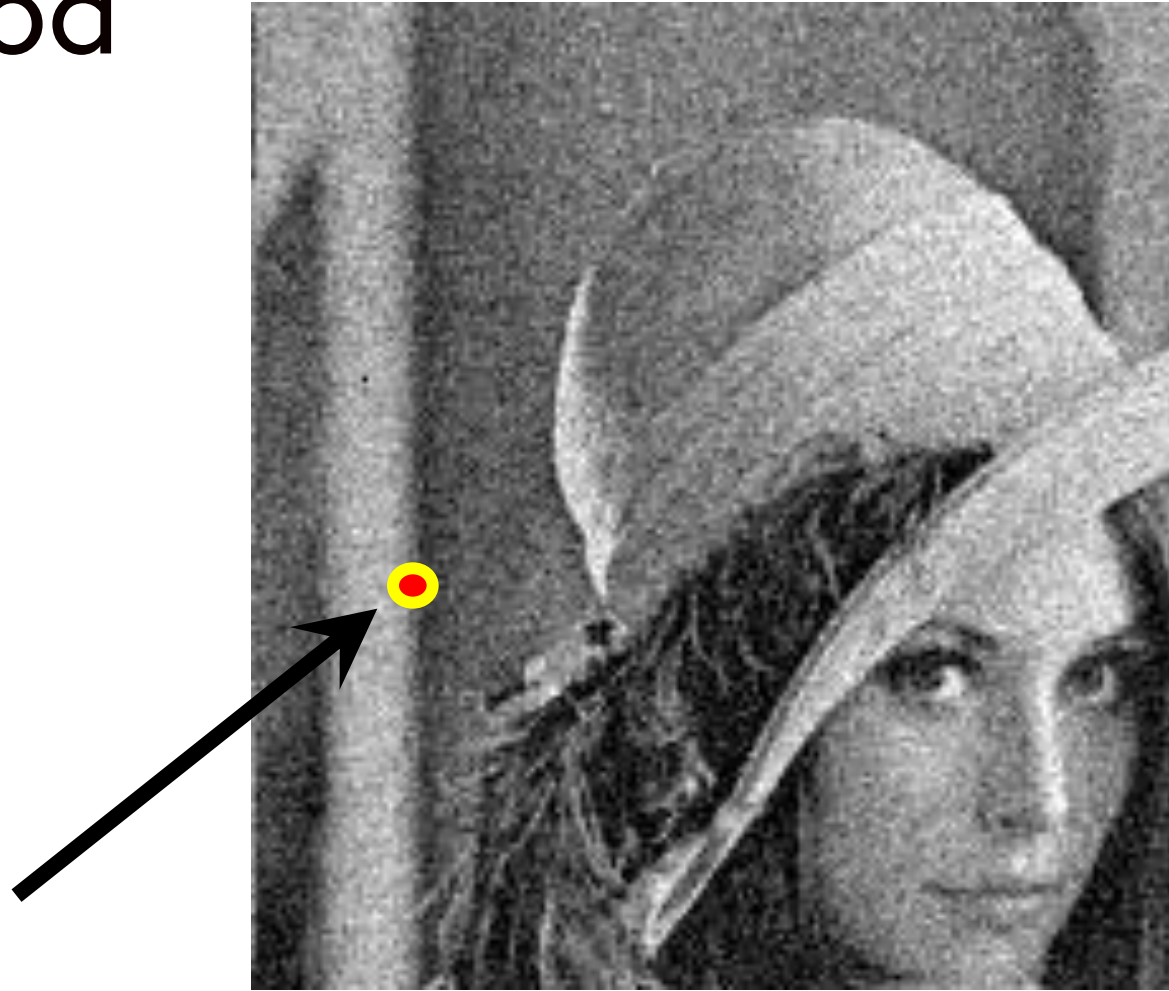
NL-Means Method

- For each and

  every pixel **p:**

  - Define a small, simple
    fixed size neighborhood;

NL-Means Method

$$V_p = \begin{bmatrix} 0.74 \\ 0.32 \\ 0.41 \\ 0.55 \\ \dots \\ \dots \\ \dots \end{bmatrix}$$

- For each and every pixel **p:**

  - Define a small, simple fixed size neighborhood;
  - Define vector Vp: a list of neighboring pixel values.

NL-Means Method

'Similar' pixels **p**, **q**

→ SMALL
  vector distance;

$$\| V_p - V_q \|^2$$
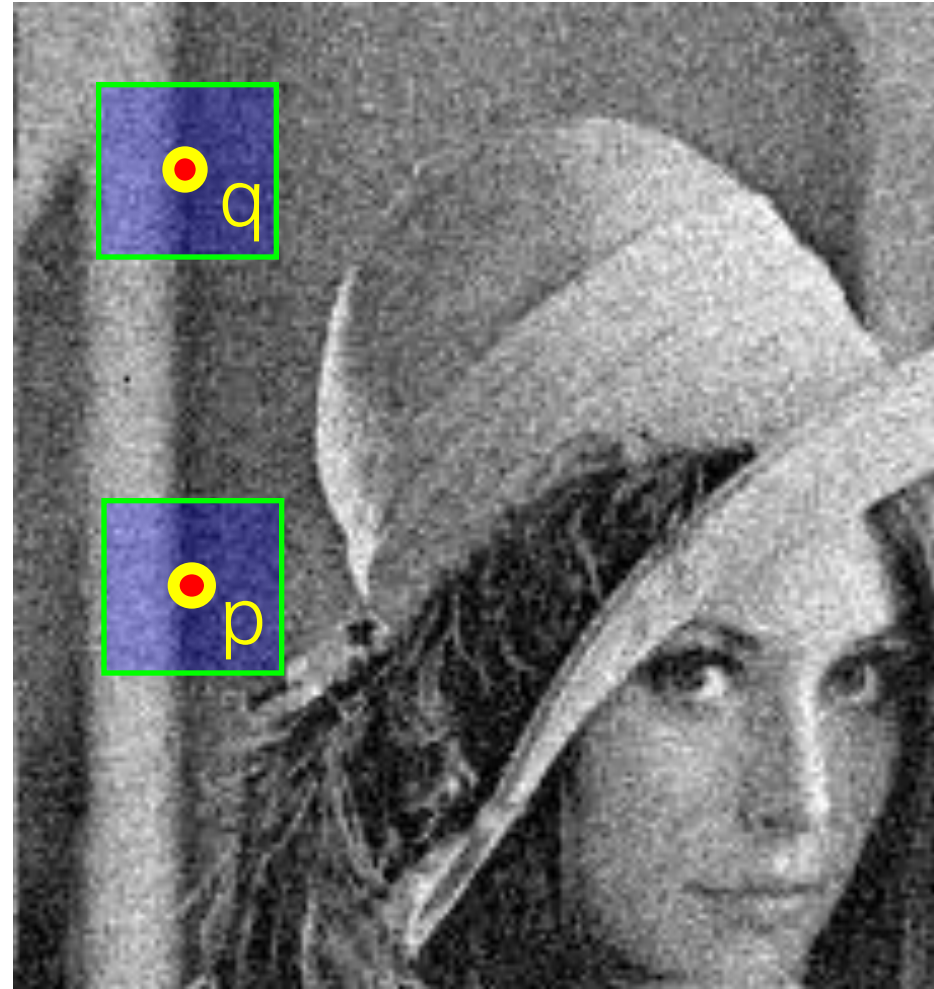
NL-Means Method

'Dissimilar' pixels p, q

→ LARGE
    vector distance;
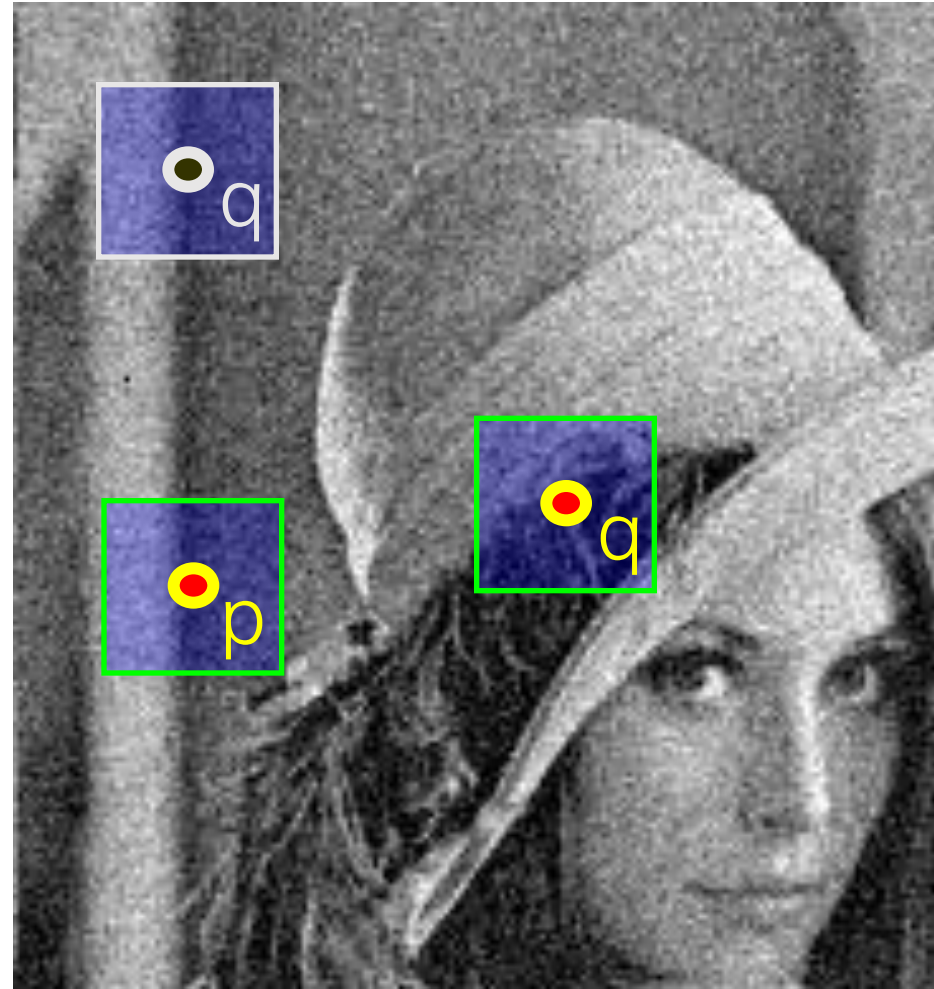
$$\| V_p - V_q \|^2$$

NL-Means Method

'Dissimilar' pixels p, q

→ LARGE
   vector distance;

$$\| V_p - V_q \|^2$$

Filter with this!

NL-Means Method

p, q neighbors define

a vector distance;

$$\| V_p - V_q \|^2$$

Filter with this:
No spatial term!

$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_{\mathrm{s}}}\left(\| \mathbf{p} - \mathbf{q} \|\right) G_{\sigma_{\mathrm{r}}}\left(\| \vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}} \|^2\right) I_{\mathbf{q}}$$

NL-Means Method

pixels p, q neighbors
   Set a vector distance;

$$\| V_p - V_q \|^2$$

Vector Distance to p sets
   weight for each pixel q

$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_{\mathrm{r}}} \left(\| \vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}} \|^2 \right) I_{\mathbf{q}}$$

NL-Means Method

(a) (b) (c)

(d) (e) (f)
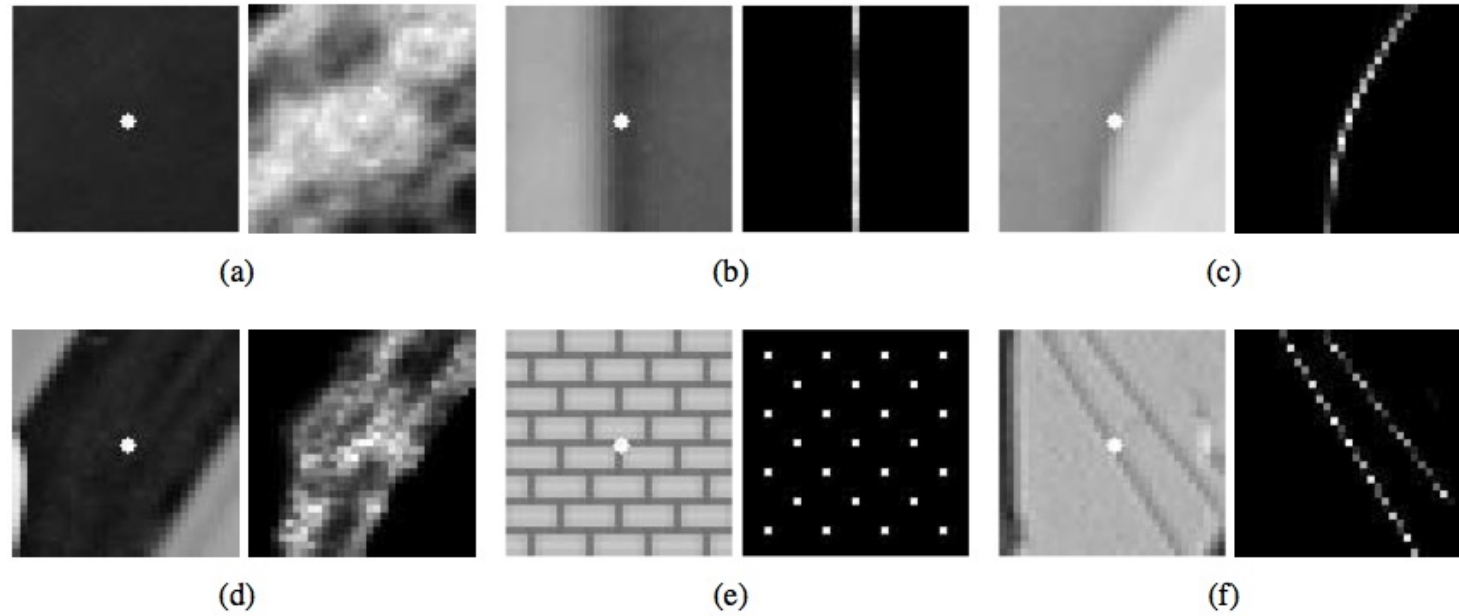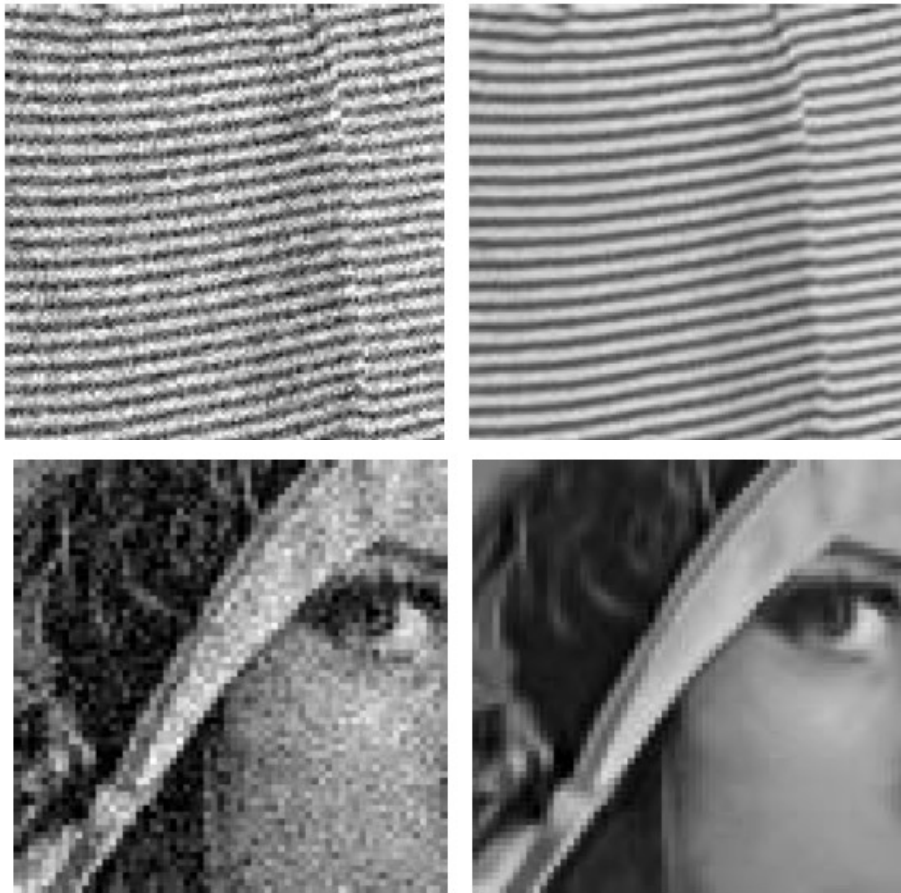
NL-Means Method

- Noisy source image:

NL-Means Method

- Gaussian Filter

Low noise,

Low detail

NL-Means Method

- Anisotropic Diffusion

Note 'stairsteps':
~ piecewise constant

NL-Means Method

- Bilateral Filter

Better, but similar 'stairsteps':

NL-Means Method

- NL-Means:

Sharp,

Low noise,

Few artifacts.

NL-Means Method
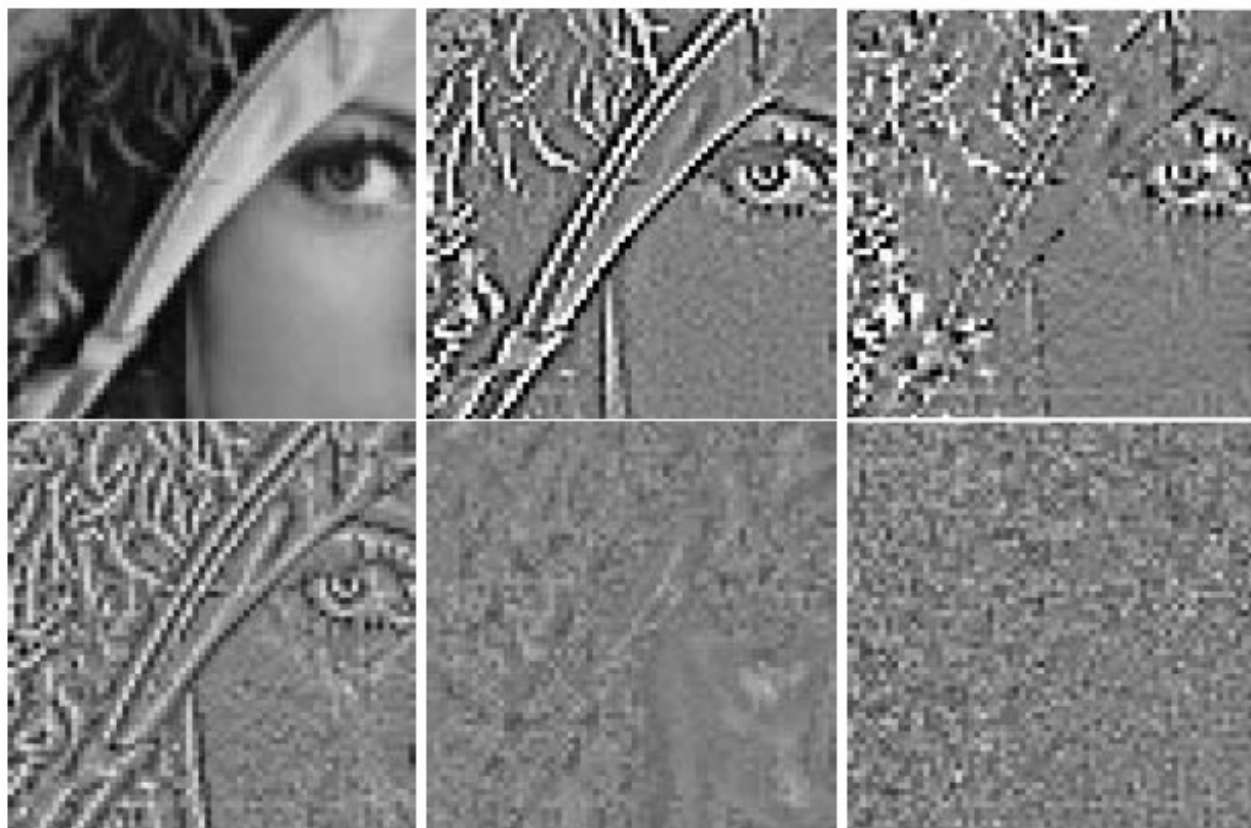
Figure 4. Method noise experience on a natural image. Displaying of the image difference $u - D_h(u)$. From left to right and from top to bottom: original image, Gauss filtering, anisotropic filtering, Total variation minimization, Neighborhood filtering and NL-means algorithm. The visual experiments corroborate the formulas of section 2.

NL-Means Method

original | noisy, standard deviation 15 | denoised

http://www.ipol.im/pub/algo/bcm_non_local_means_denoising/

RegCov Smoothing

From pixels to patches and to images

Images

Patches
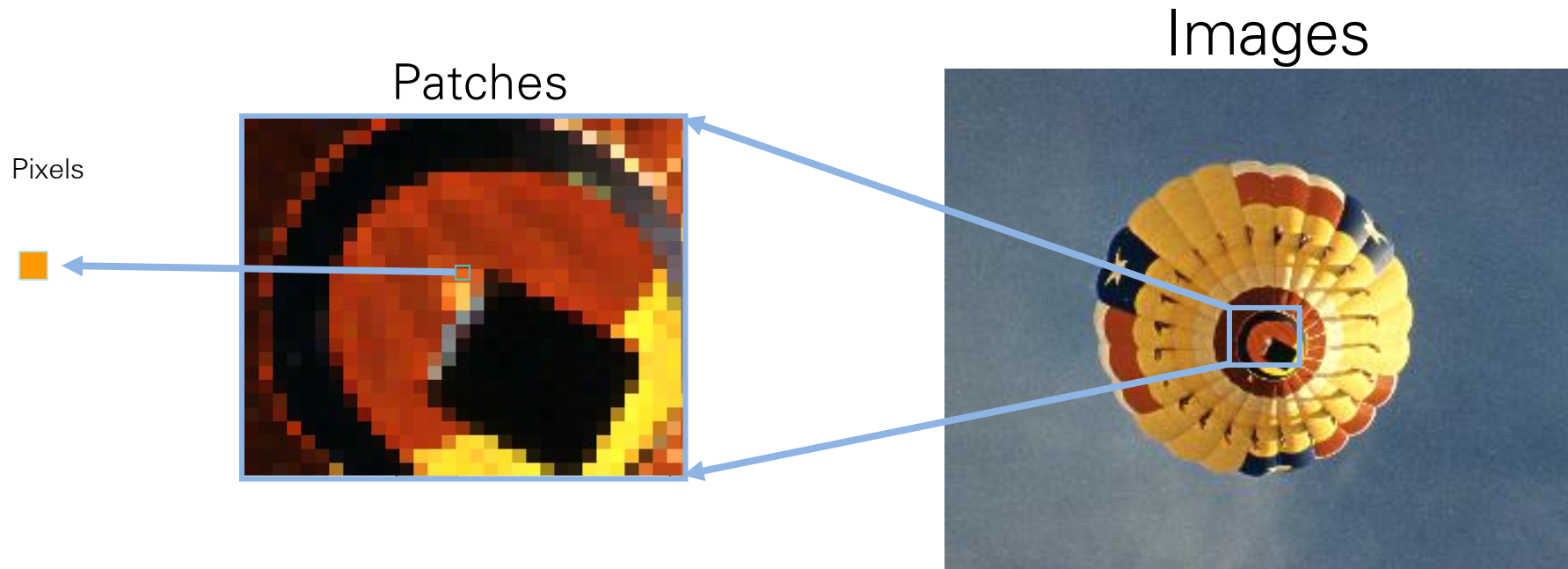
Pixels

Similarities can be defined at different scales..

Pixelwise similarity metrics

- To measure the similarity of two pixels, we can consider
  - Spatial distance
  - Gray-level distance
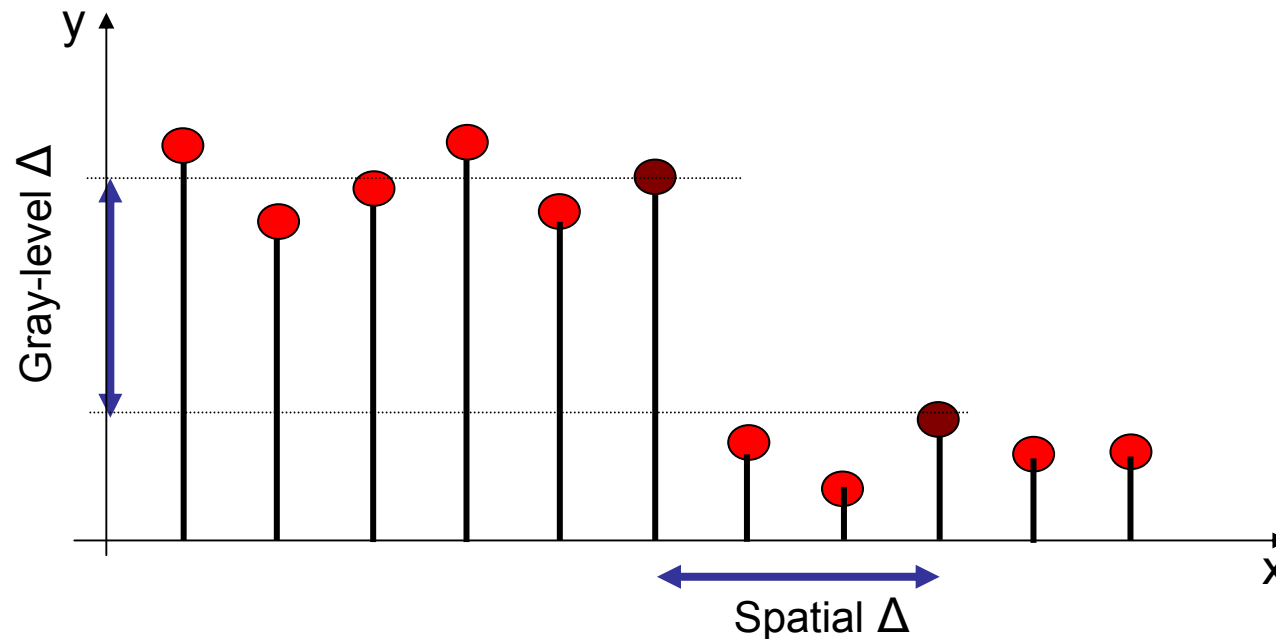
Euclidean metrics

- Natural ways to incorporate the two Δs:
  - Bilateral Kernel [Tomasi, Manduchi, '98] (pixelwise)
  - Non-Local Means Kernel [Buades, et al. '05] (patchwise)
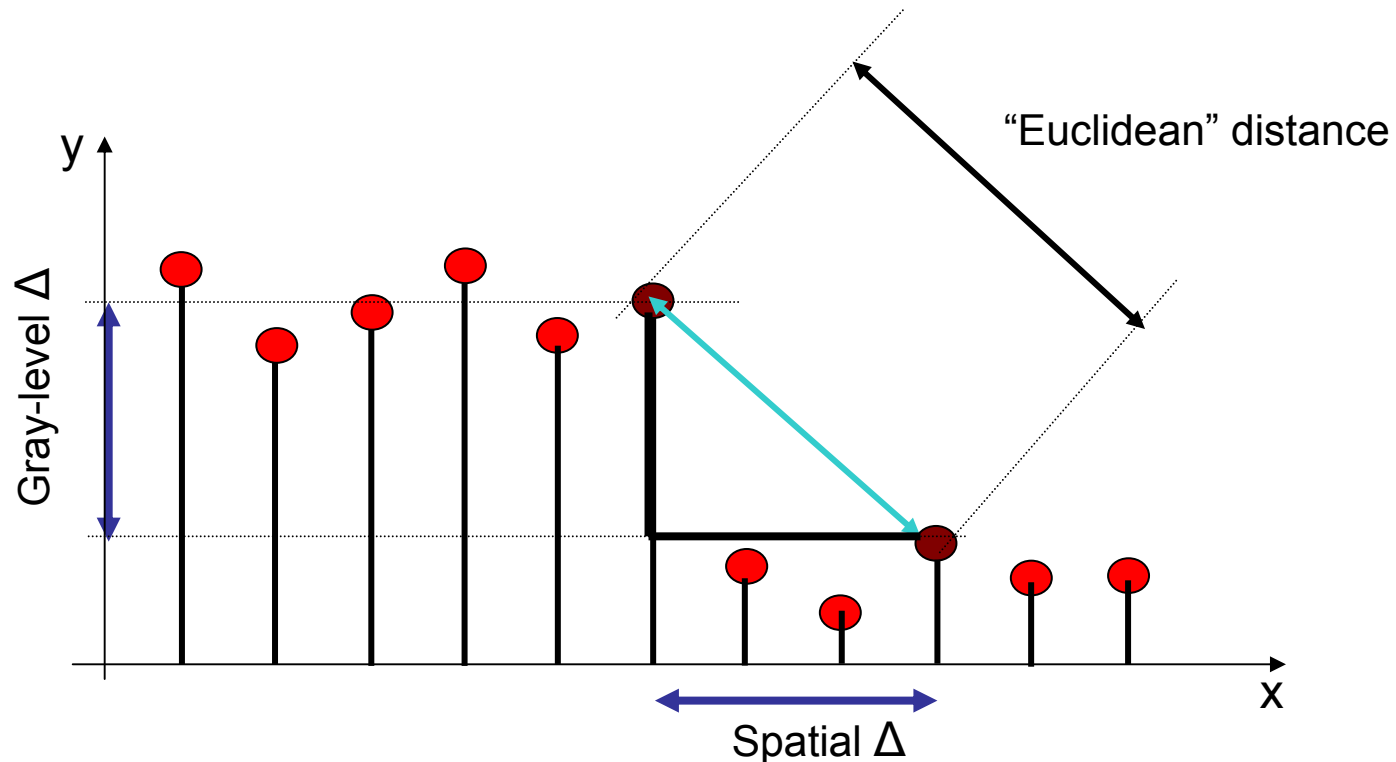
Bilateral Kernel (BL) [Tomasi et al. '98]

Pixels

$$K(\mathbf{x}_l, \mathbf{x}, y_l, y) = \exp\left\{-\frac{\|y_l - y\|^2}{h_r^2} - \frac{\|\mathbf{x}_l - \mathbf{x}\|^2}{h_d^2}\right\}$$

Pixel similarity   **Spatial** similarity
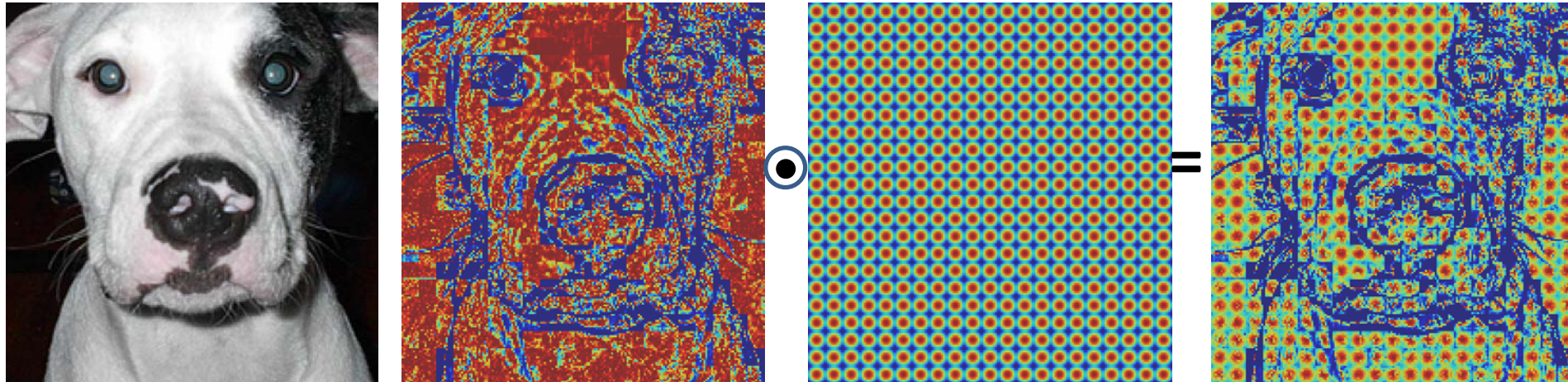
Non-local Means (NLM) [Buades et al. '05]

Patches

$$K(\mathbf{x}_l, \mathbf{x}, \mathbf{y}_l, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{y}_l - \mathbf{y}\|^2}{h_r^2} - \frac{\|\mathbf{x}_l - \mathbf{x}\|^2}{h_d^2}\right\}$$

Patch similarity   **Spatial** similarity

⊙   =

Smoothing effect

Structure-Texture Decomposition

- Decomposing an image into structure and texture components

Input Image

Structure-Texture Decomposition

- Decomposing an image into structure and texture components

Structure Component

Structure-Texture Decomposition

- Decomposing an image into structure and texture components

Texture Component

Structure-Texture Decomposition

- Decomposing an image into structure and texture components

Input Image
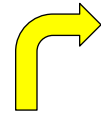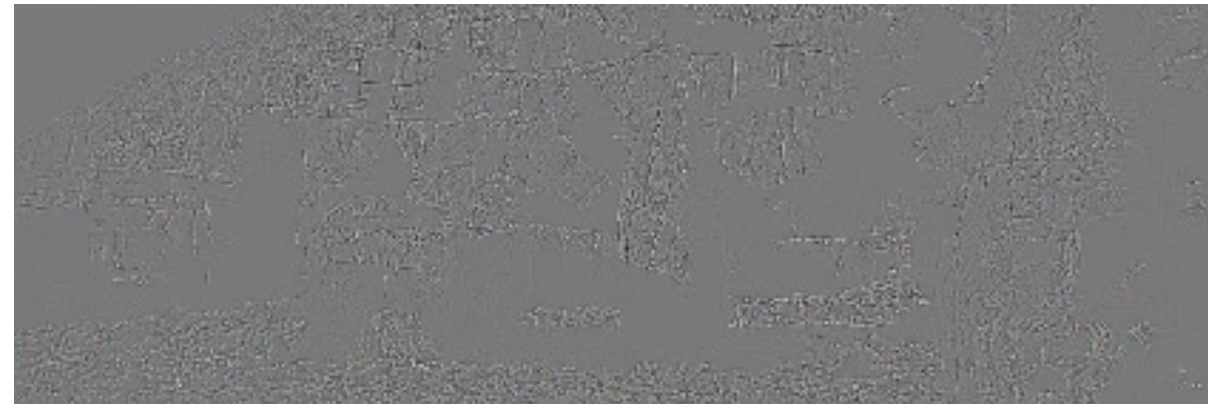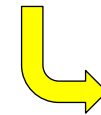
Structure

Texture

Structure-Texture Decomposition

$$F(x, y) = \phi(I, x, y)$$

$$F(x,y) = \left[I(x,y) \quad \left|\frac{\partial I}{\partial x}\right| \quad \left|\frac{\partial I}{\partial y}\right| \quad \left|\frac{\partial^2 I}{\partial x^2}\right| \quad \left|\frac{\partial^2 I}{\partial y^2}\right| \quad x \quad y \right]^T$$
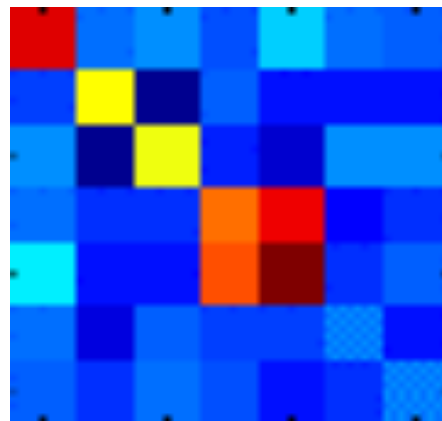
$$\mathbf{C}_R = \frac{1}{n-1} \sum_{i=0}^{n} (\mathbf{z}_k - \mu)(\mathbf{z}_k - \mu)^T$$

Tuzel et al., ECCV 2006

Structure-Texture Decomposition

- Region covariances capture local structure and texture information.

- Similar regions have similar statistics.
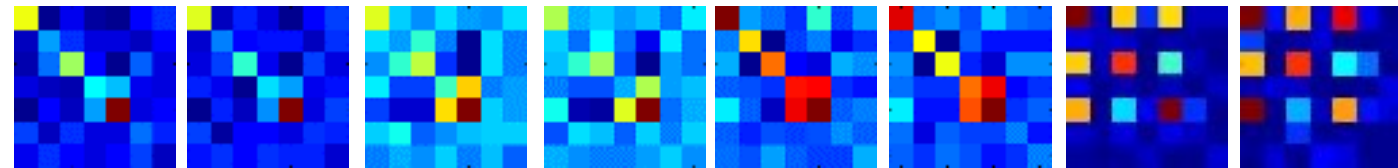
RegCov Smoothing - Formulation

$$I = S + T$$

$$S(\mathbf{p}) = \frac{1}{Z_\mathbf{p}} \sum_{\mathbf{q} \in N(\mathbf{p}, r)} w_\mathbf{pq} I(\mathbf{q})$$
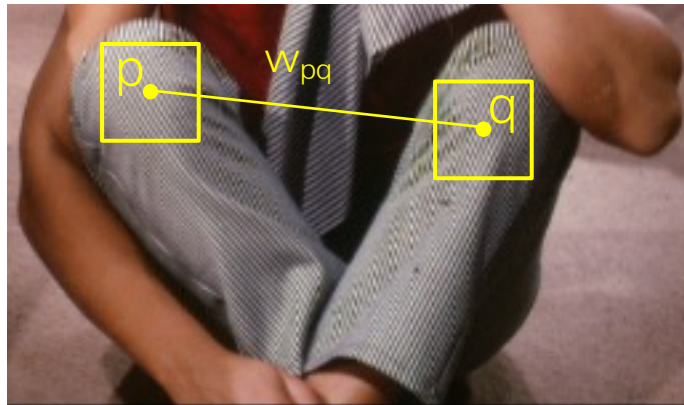
- Structure-texture decomposition via smoothing

- Smoothing as weighted averaging

- Different kernels (w_{pq}) result in different types of filters.

- Three novel patch-based kernels for structure texture decomposition.

- L. Karacan, A. Erdem, E. Erdem, "Structure Preserving Image Smoothing via Region Covariances", ACM TOG 2013 (SIGGRAPH Asia 2013)

RegCov Smoothing – Model 1

- Depends on sigma-points representation of covariance matrices (Hong et al.,CVPR'09)

$$\mathbf{C} = \mathbf{L}\mathbf{L}^T$$ Cholesky Decomposition

$$\mathcal{S} = \{\mathbf{s}_i\}$$ Sigma Points $\quad \mathbf{s}_i = \left\{ \begin{array}{ll} \alpha\sqrt{d}\mathbf{L}_i & \text{if } 1 \leq i \leq d \\ -\alpha\sqrt{d}\mathbf{L}_i & \text{if } d+1 \leq i \leq 2d \end{array} \right.$
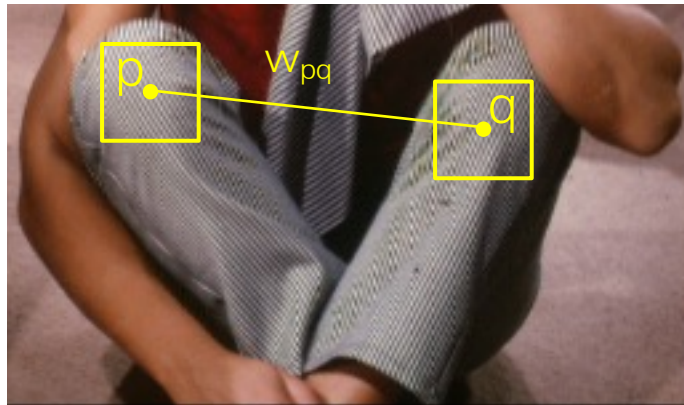
Final representation

$$\Psi(\mathbf{C}) = (\mu, \mathbf{s}_1, \ldots, \mathbf{s}_d, \mathbf{s}_{d+1}, \ldots, \mathbf{s}_{2d})^T$$

Resulting kernel function

$$w_{\mathbf{pq}} \propto \exp\left(-\frac{\|\Psi(\mathbf{C_p}) - \Psi(\mathbf{C_q})\|^2}{2\sigma^2}\right)$$

174

RegCov Smoothing − Model 2

- An alternative way is to use statistical similarity measures.

- A Mahalanobis-like distance measure to compare to image patches.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(\mu_{\mathbf{p}} - \mu_{\mathbf{q}})\mathbf{C}^{-1}(\mu_{\mathbf{p}} - \mu_{\mathbf{q}})^T}$$
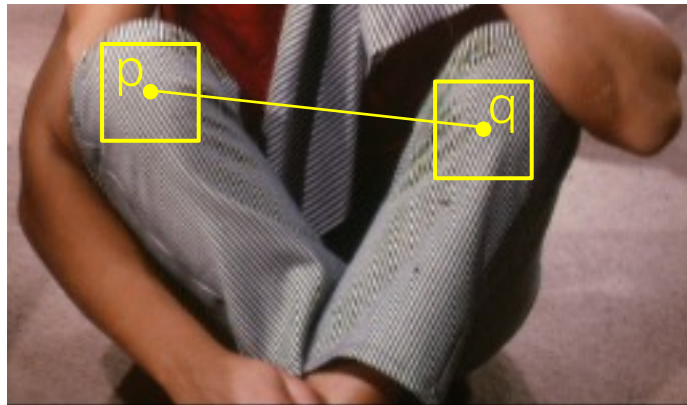
$$\mathbf{C} = \mathbf{C_p} + \mathbf{C_q}$$

Resulting kernel $\qquad w_{\mathbf{pq}} \propto \exp\left(-\frac{d(\mathbf{p}, \mathbf{q})^2}{2\sigma^2}\right)$

RegCov Smoothing – Model 3

- We use Kullback-Leibler(KL)-Divergence measure from probability theory.

- A KL-Divergence form is used to calculate statistical distance between two multivariate normal distribution
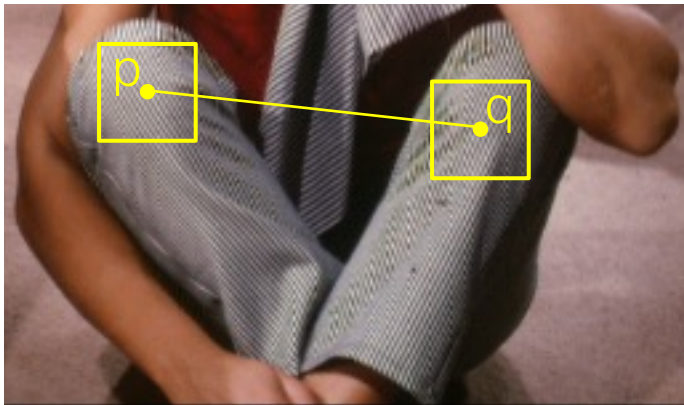
$$d_{KL}(\mathbf{p}, \mathbf{q}) = \frac{1}{2}\left(tr(\mathbf{C_q}^{-1}\mathbf{C_p}) + (\mu_p - \mu_q)^T \mathbf{C_q}^{-1}(\mu_p - \mu_q) - k - ln\left(\frac{\det \mathbf{C_p}}{\det \mathbf{C_q}}\right)\right)$$

Resulting kernel $\qquad w_{pq} \propto \dfrac{d_{KL}(\mathbf{p}, \mathbf{q})}{2\sigma^2}$

resulted from a discussion with Rahul Narain (Berkeley University)

RegCov Smoothing – Smoothing Kernels
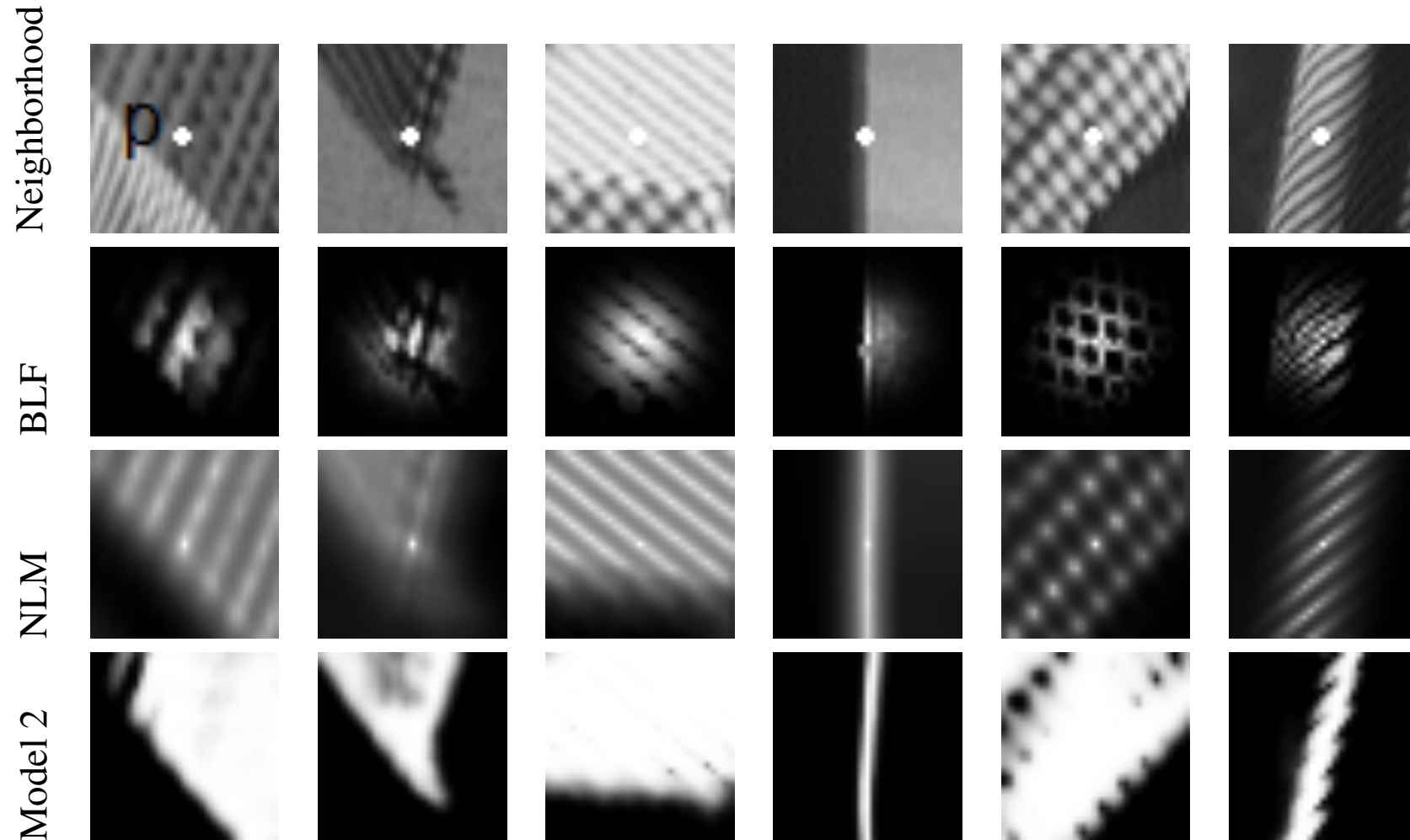
Results

Input

Results

Model1   Model2   Model3
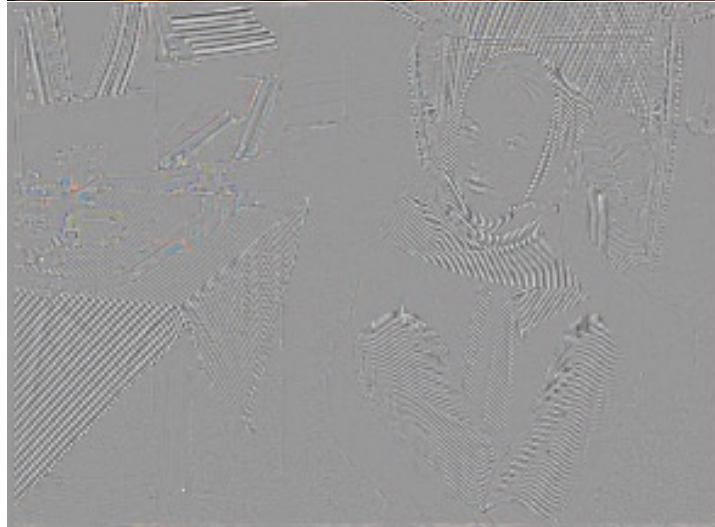
Input
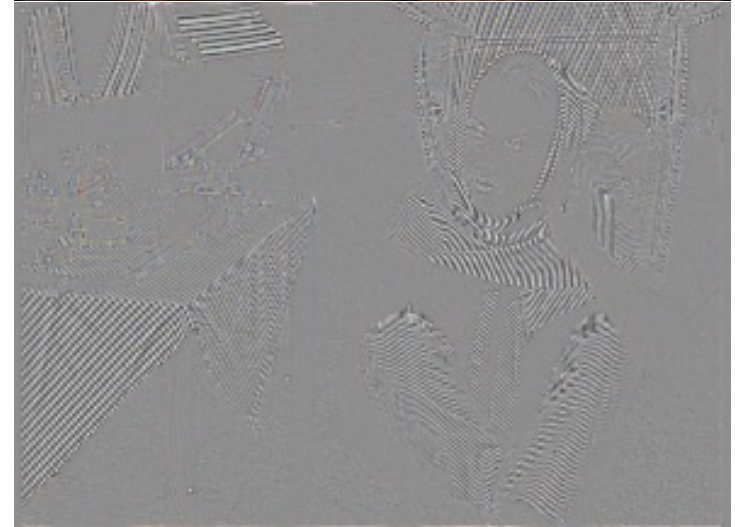
Input
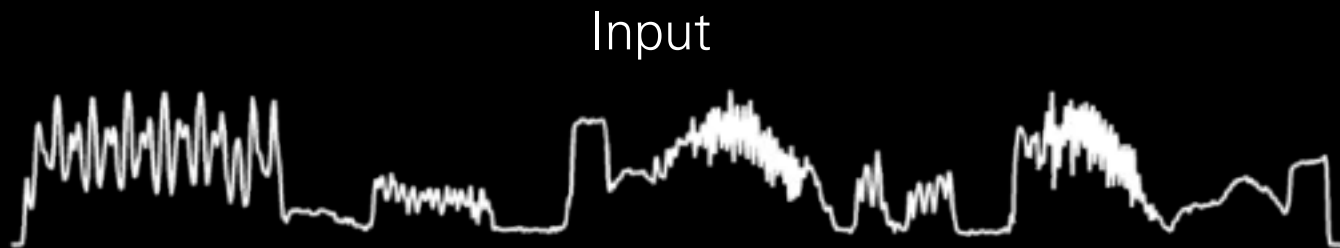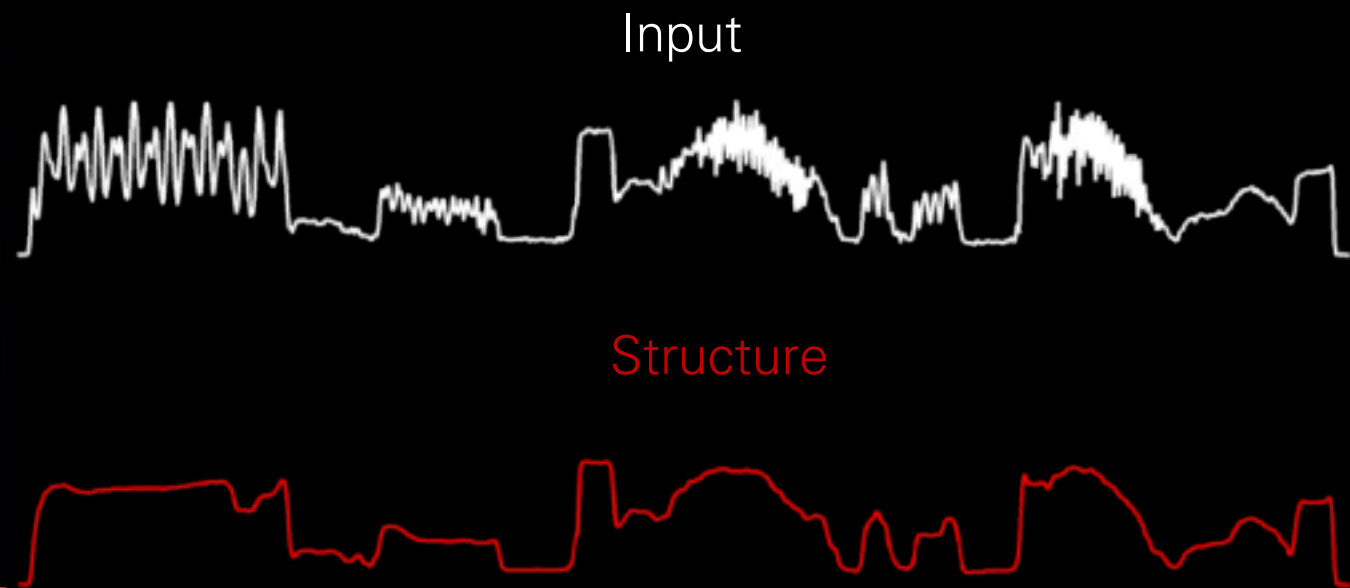
Input

Structure

Model2 Structure

Model2 Texture

Input

Structure
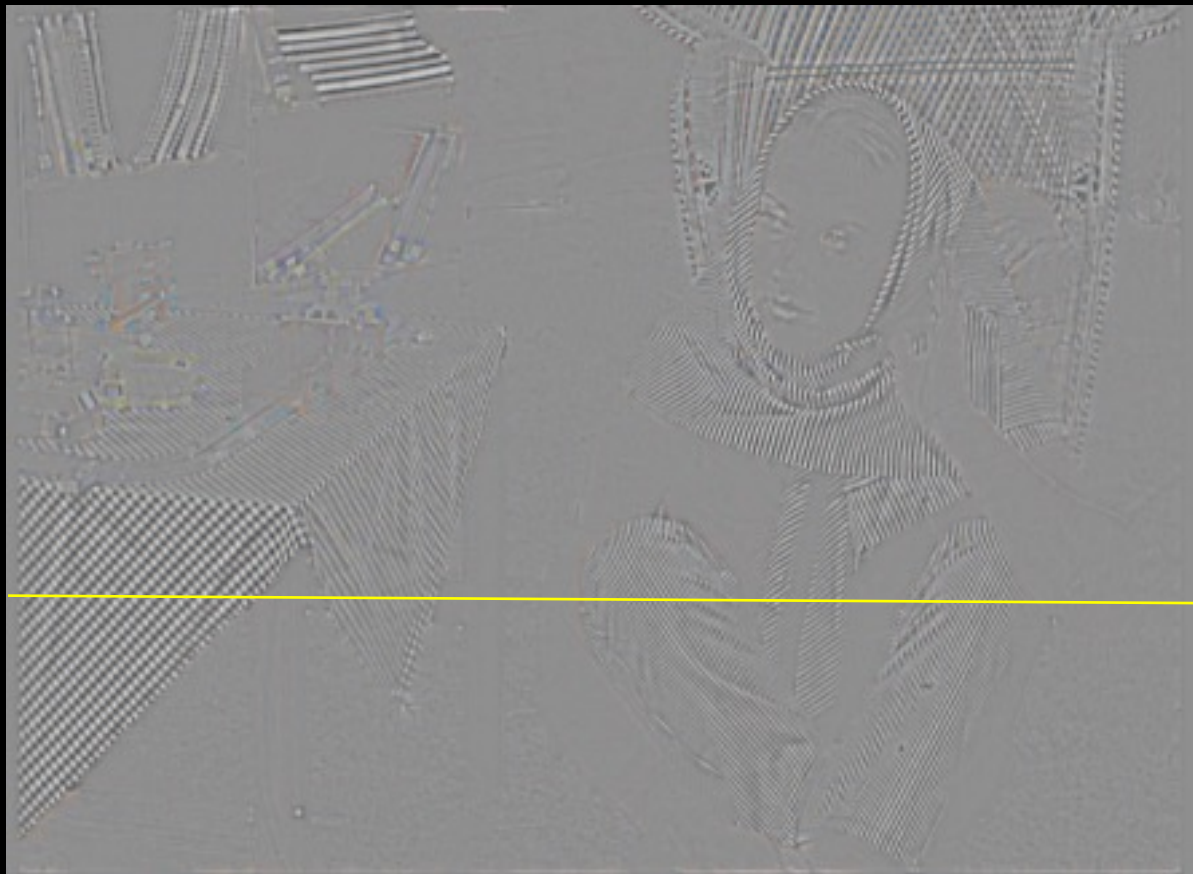
Texture
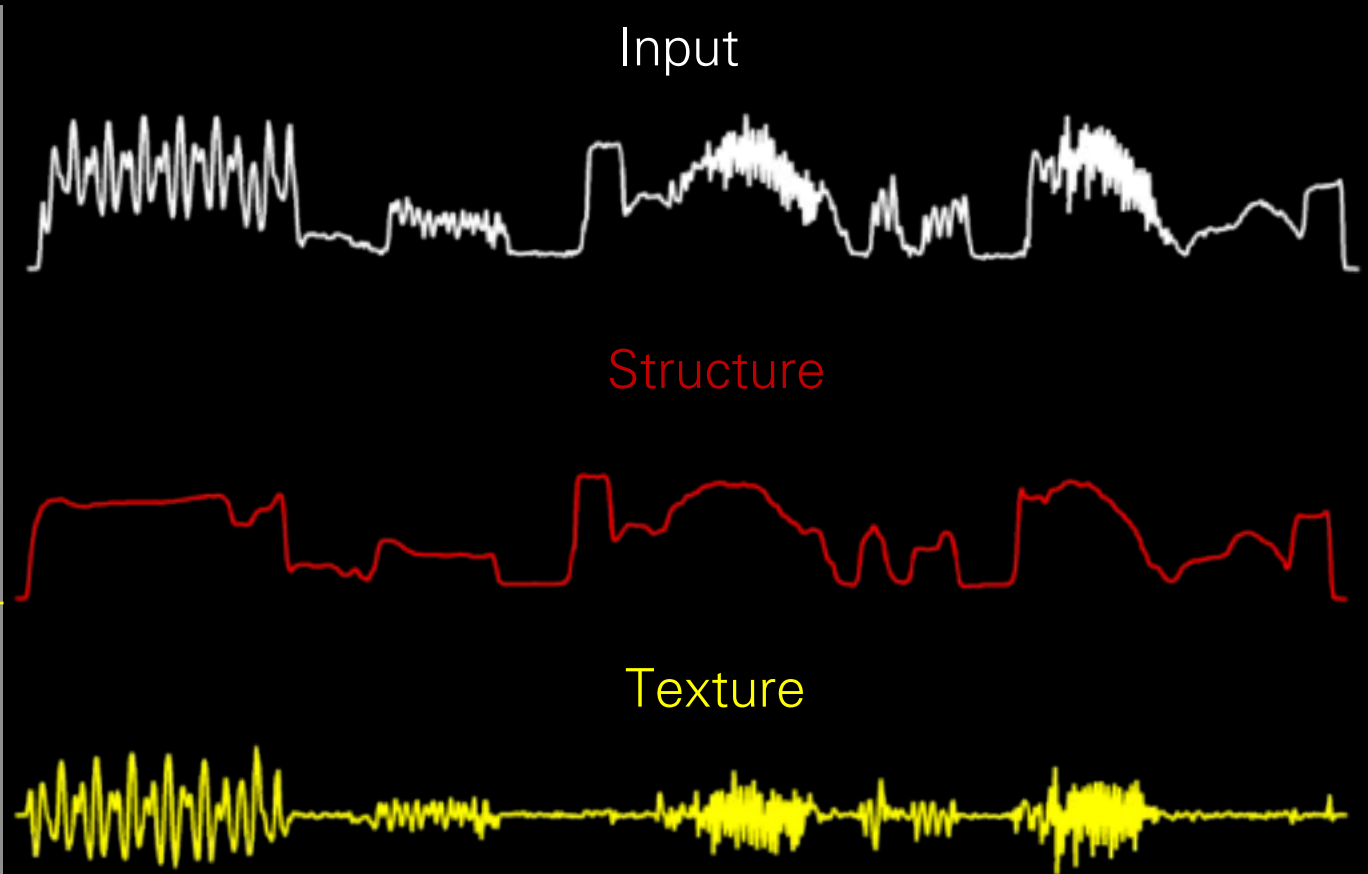
Results

Input　　　　　　　　Model2　　　　　　　　Model3
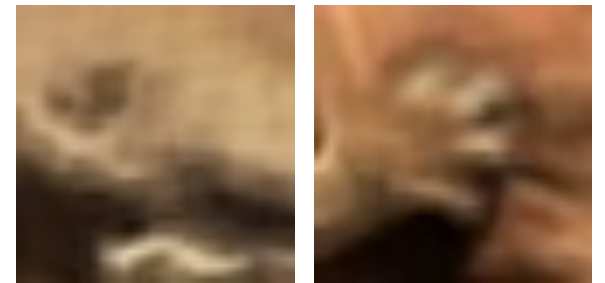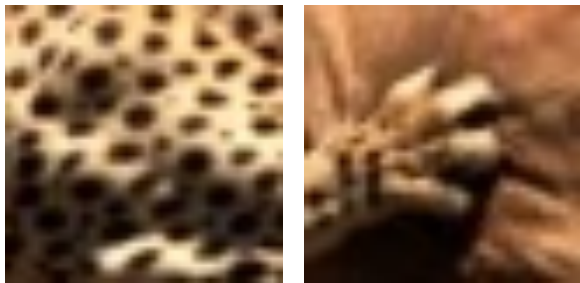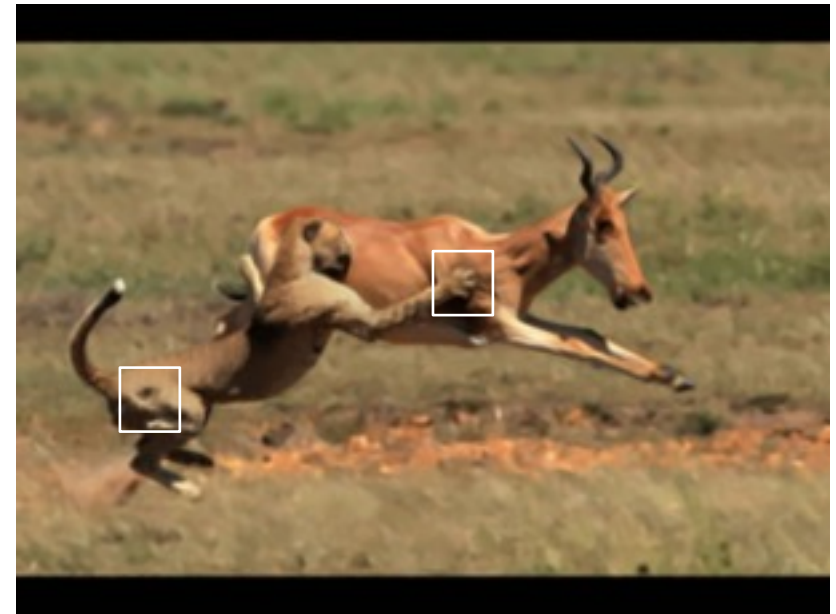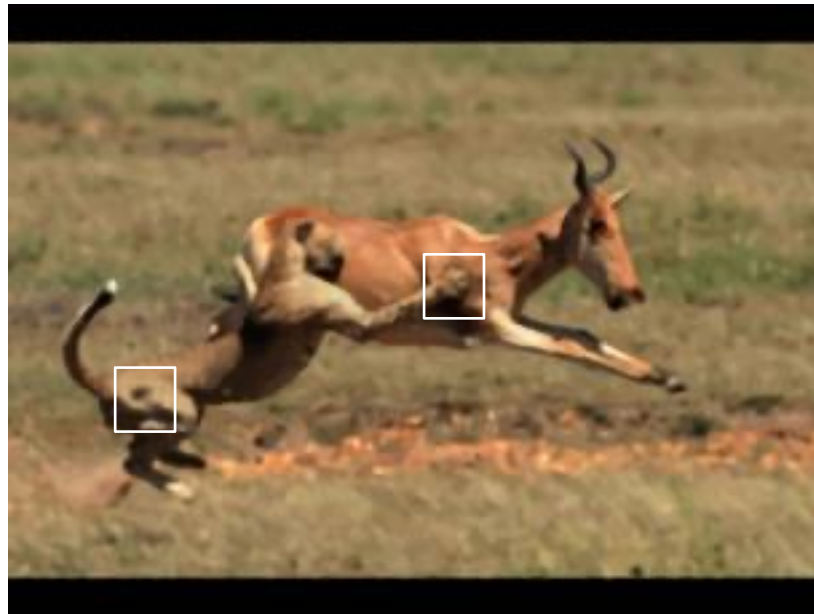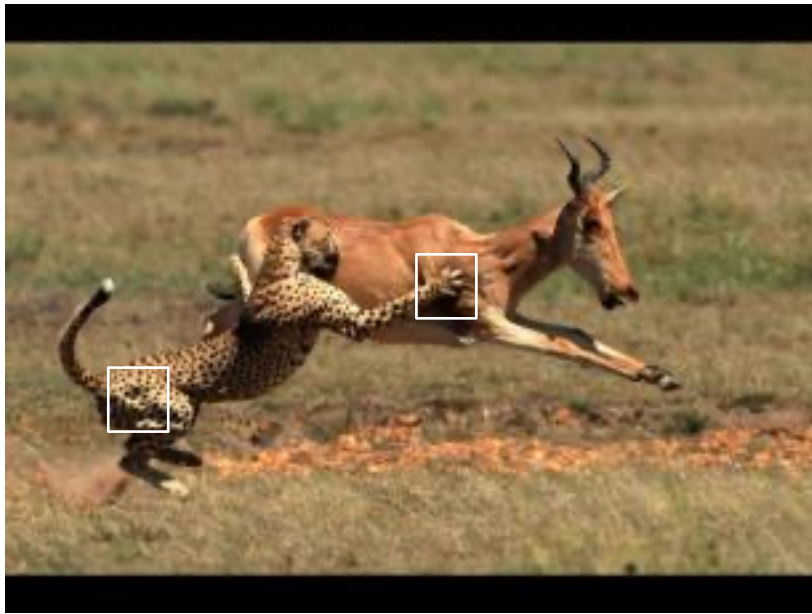
Experimental evaluation

Input

Experimental evaluation

TV
Rudin et al. 1992

Experimental evaluation

Bilateral
Filter

Experimental evaluation

Envelope
Extraction
Subr et al. 2009

Experimental evaluation

RTV
Xu et al. 2012

Experimental evaluation

Model 1

Experimental evaluation

Model 2

Experimental evaluation

Model 3

Experimental evaluation
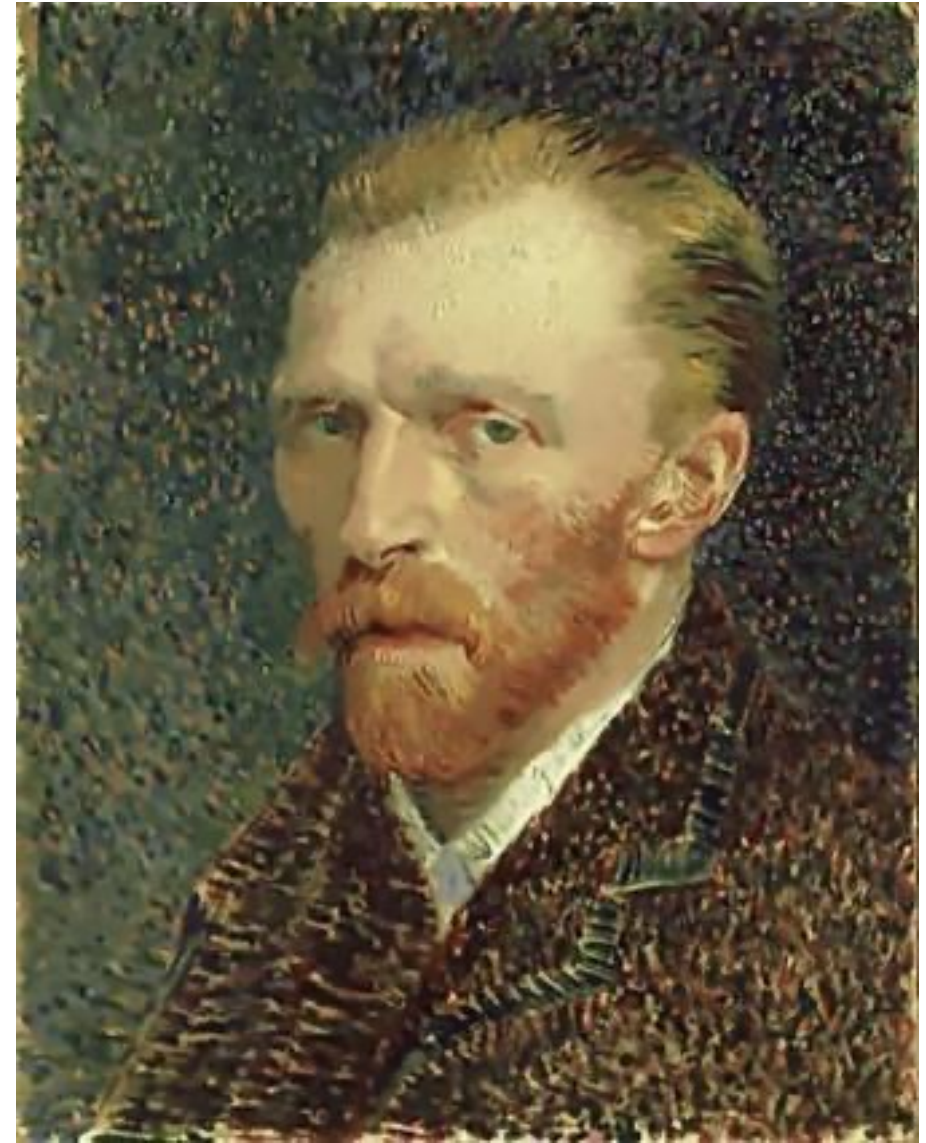
Input
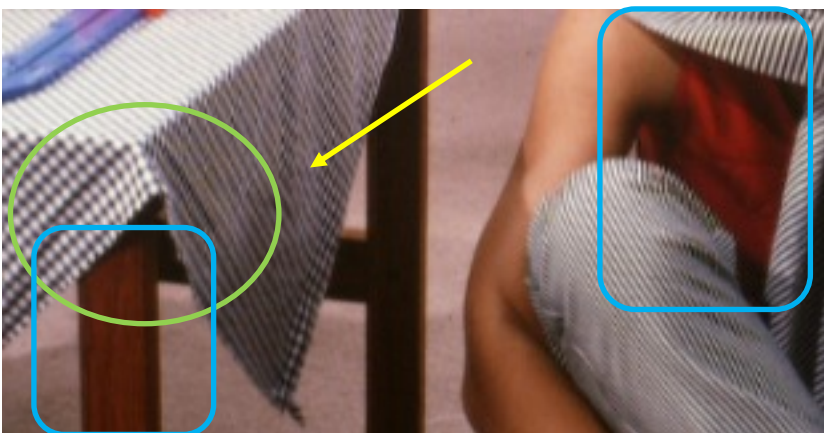
Local Exrema
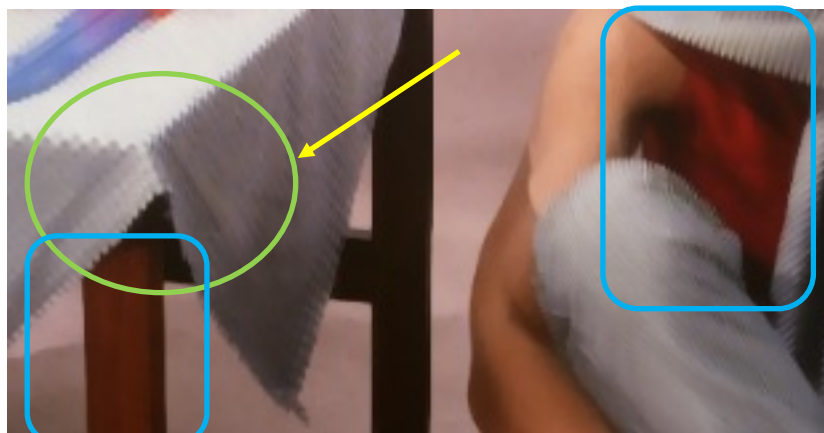
RTV

Model1
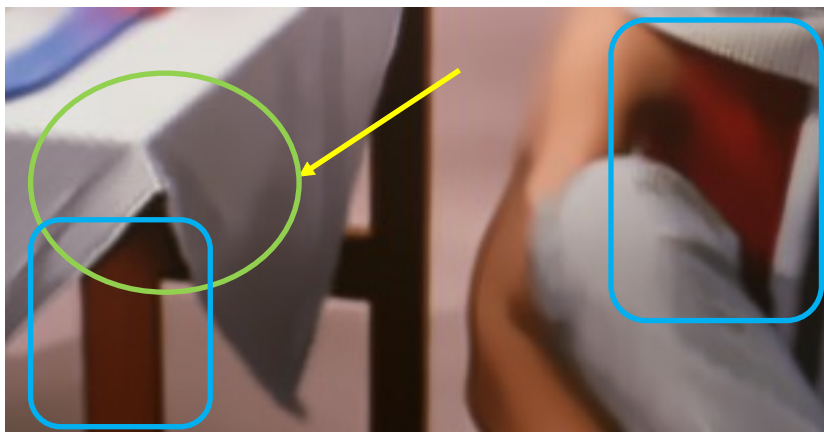
Model2
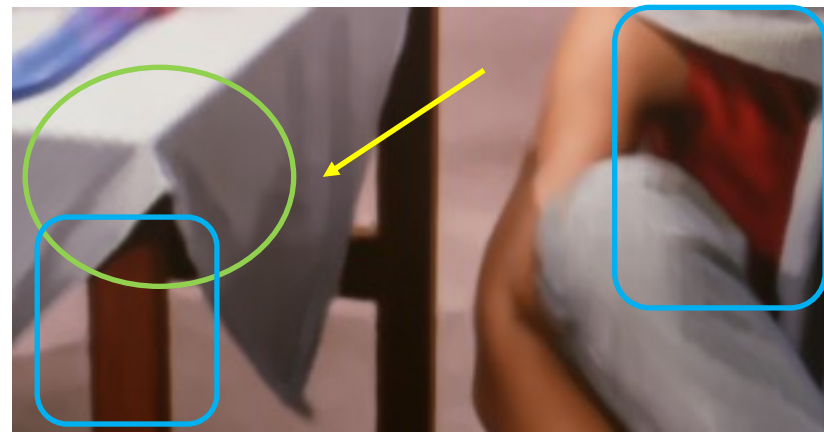
Model3
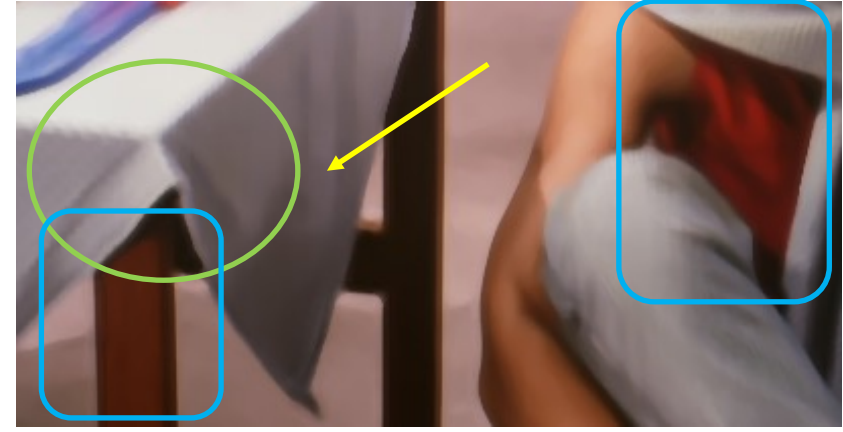
Shading preserved

Structure preserved

No unintuitive edge

Multiscale decomposition
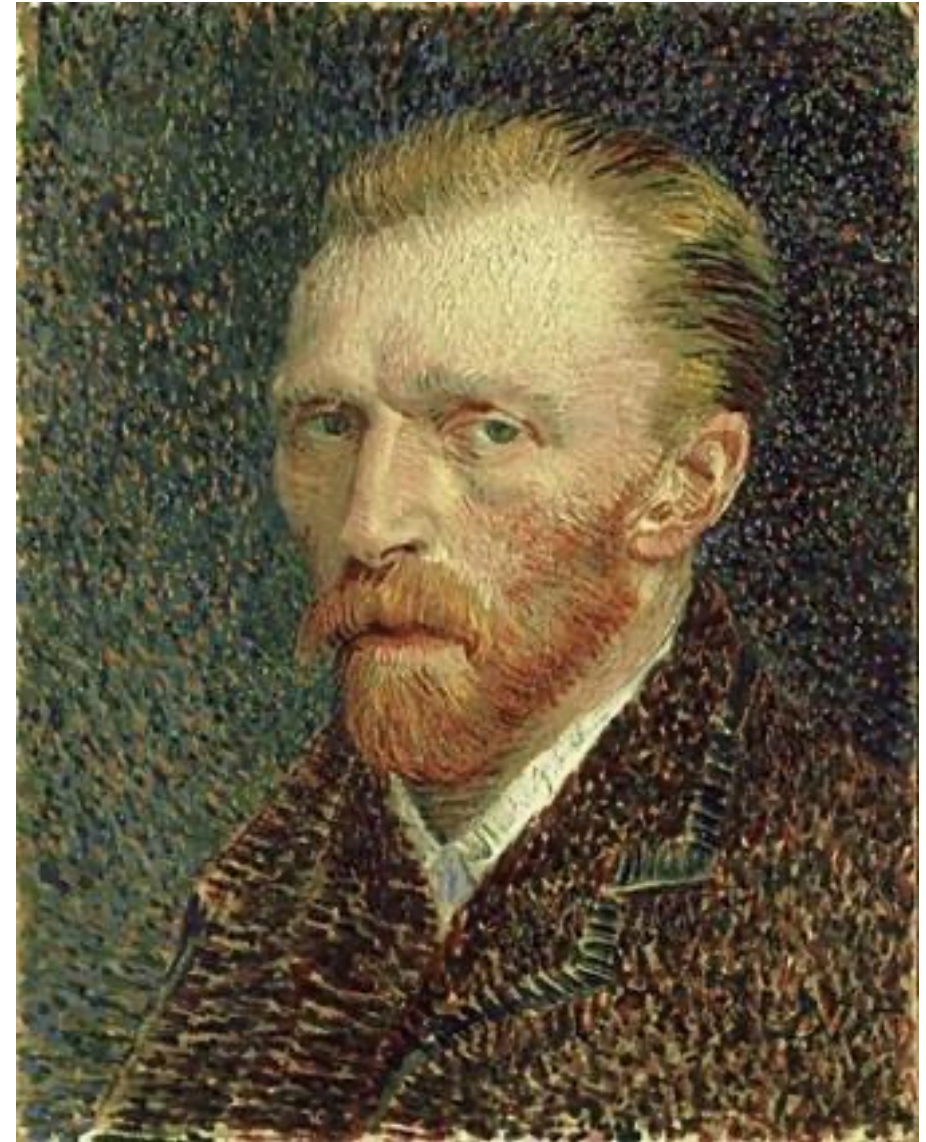
Multiscale decomposition

$S_1(k = 5)$

Multiscale decomposition

$S_2(k = 7)$

Multiscale decomposition

$S_3(k = 9)$

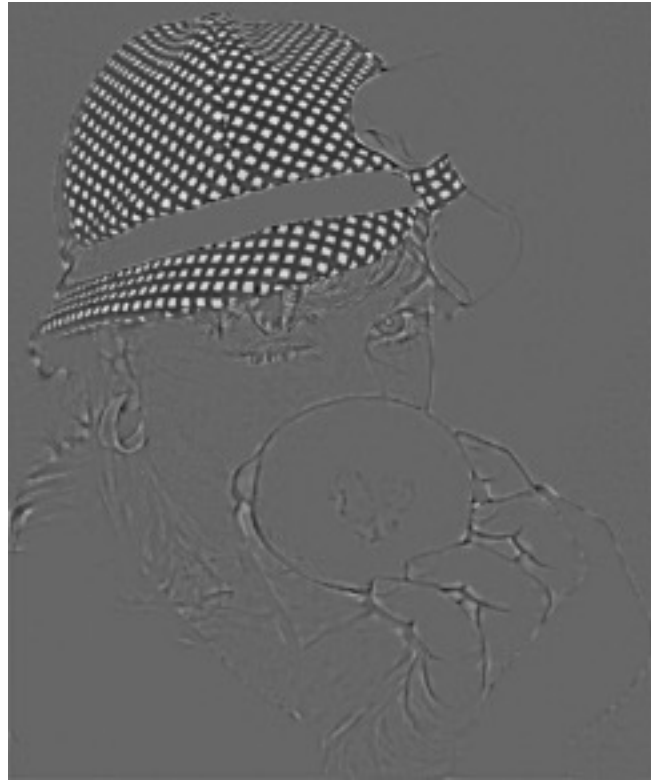# Challenging cases

Input   Model2

Challenging cases

Input   Model2 Texture   Model2+Model1

Edge detection

Edge detection

Edge detection

Canny edges of original image

Canny edges of smoothed image

Image abstraction

Image abstraction

Detail boosting

Image composition

Inverse Halftoning

Input
halftone image

Inverse Halftoning

Smoothed image
(Model 2)

Inverse Halftoning

Smoothed image
(Model 2 + Shock filter)

Inverse Halftoning

Model2+Shock Filter

Input halftone image

Kopf and Lischinski 2012

Image Retargeting

Input image

Smoothed image

Image Retargeting

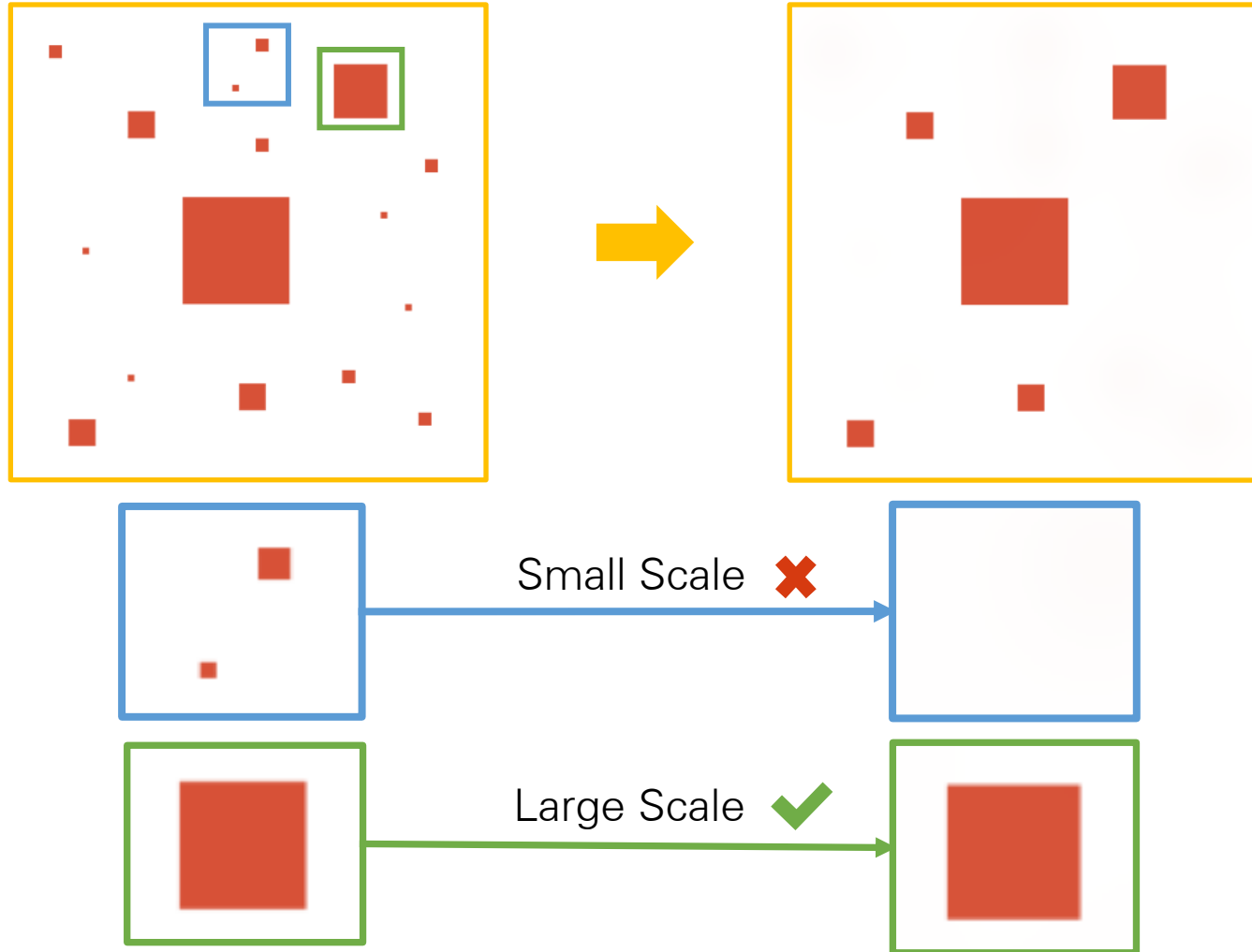Input image
Extracted Seams

Smoothed image
Extracted Seams

Seam Carving, Avidan and Shamir 2007

Image Retargeting

Input image
Retargeting result

Smoothed image
Retargeting result

Seam Carving, Avidan and Shamir 2007

Rolling Guidance Filter

Scale-Aware Filtering

Small Scale ✖

Large Scale ✔

Notion of Scale in Filtering

input image

texture
disappear

white dots
disappear
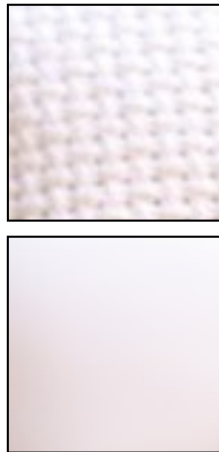
the eye
disappears
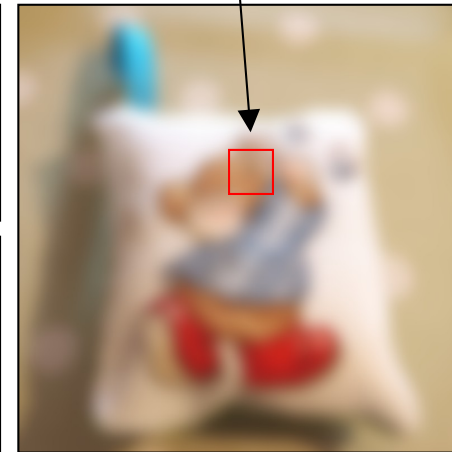
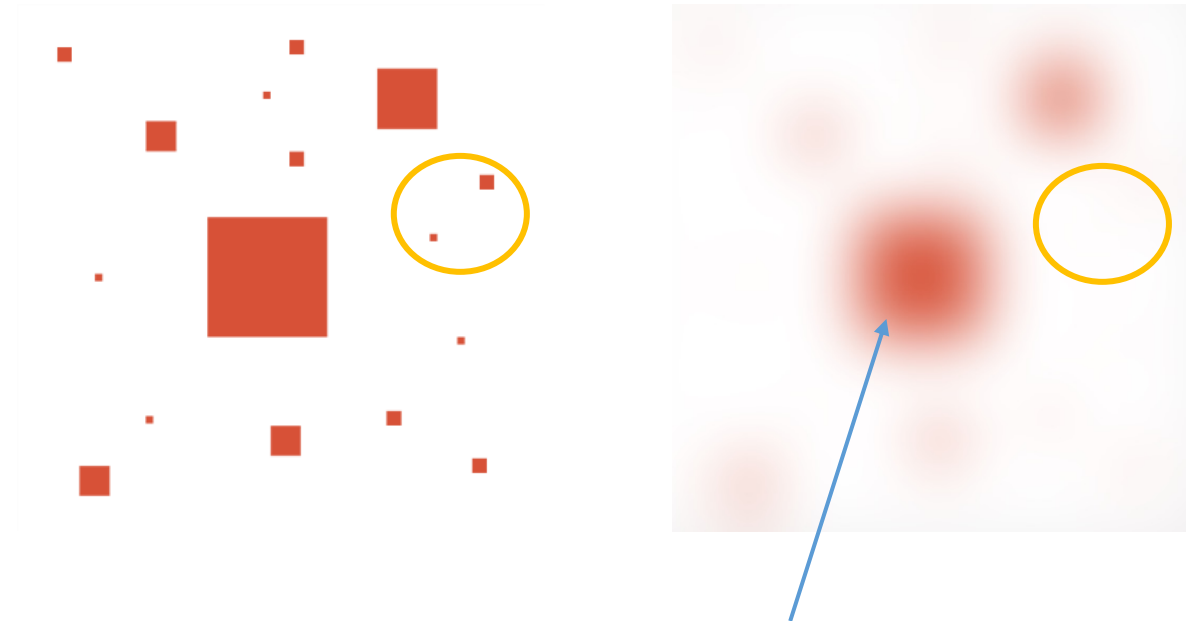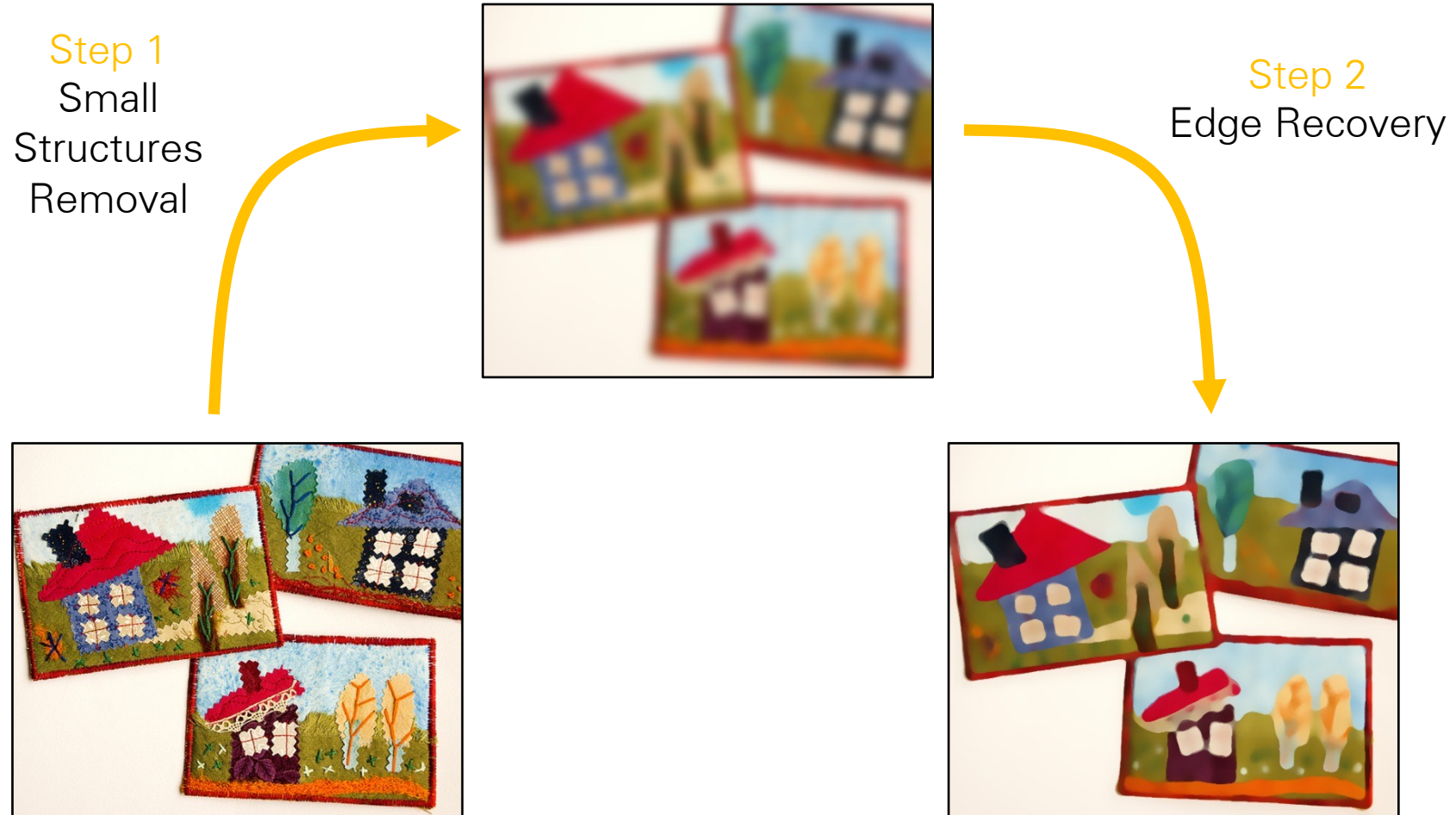s = 0   s = 4   s = 8   s = 14
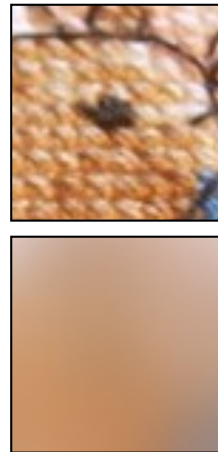
As the Gaussian kernel gets larger, more and more structures disappear.

Main Idea

- Scale Space Theory [Lindeberg, 1994]:
  - An object of size t, will be largely smoothed away with Gaussian filter of variance t^2.
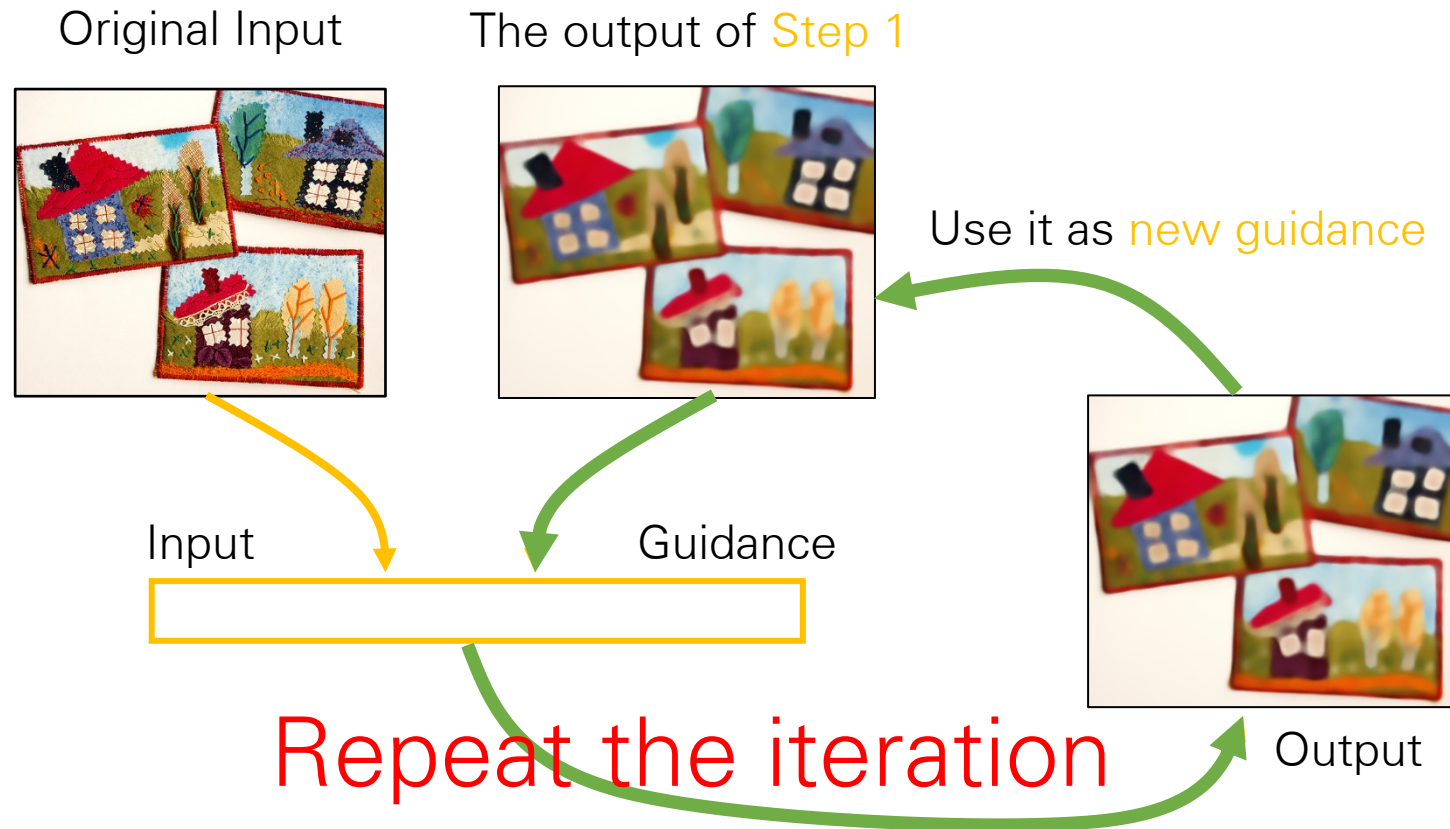
RGF: A scale-aware Filter

Step 1
Small
Structures
Removal

Step 2
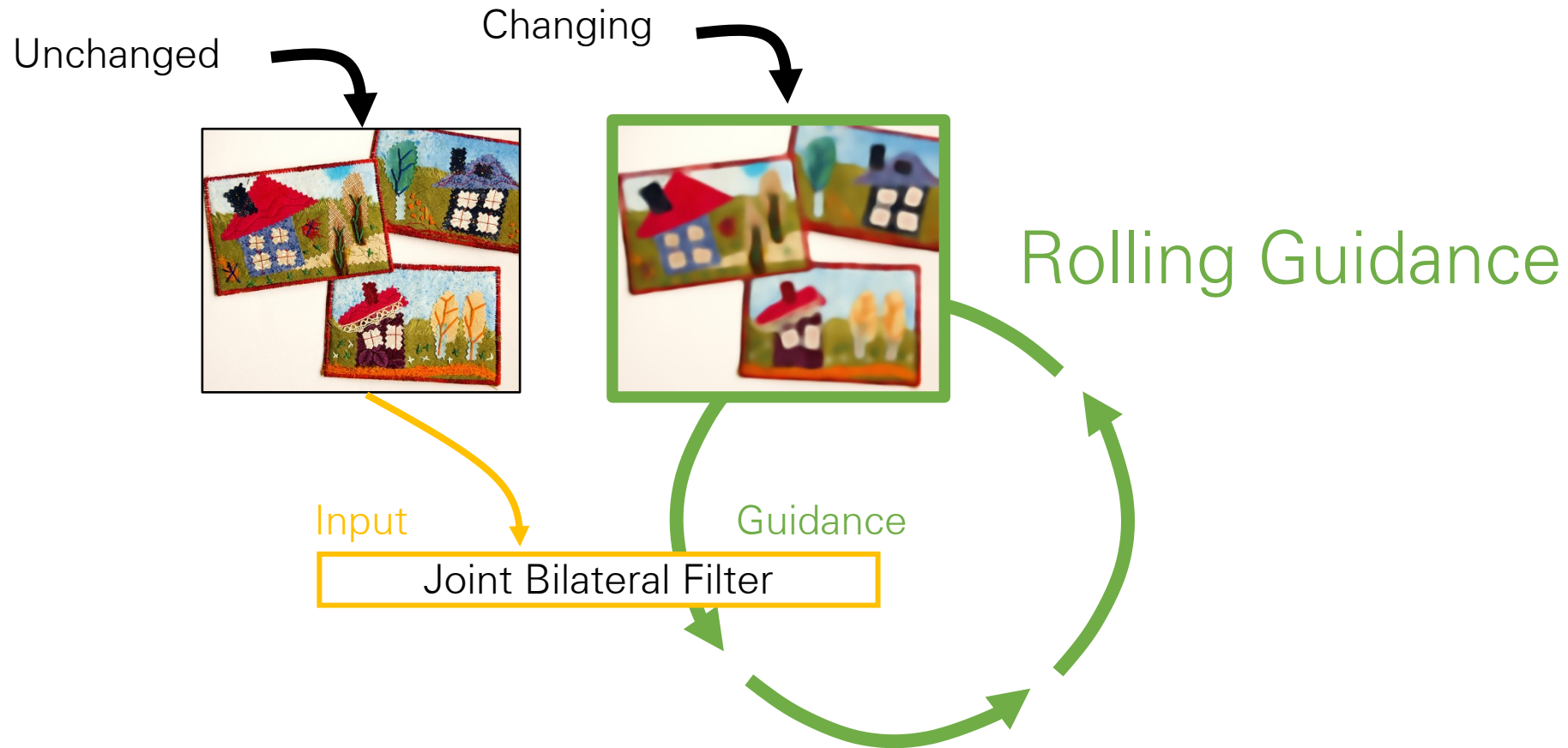Edge Recovery

Step 1: Small Structures Removal

Gaussian Filter

Step 2: Edge Recovery

- A rolling guidance

Original Input

The output of Step 1

Use it as new guidance

Input

Guidance

Output

Repeat the iteration

Rolling Guidance

Unchanged

Changing

Rolling Guidance

Input

Guidance

Joint Bilateral Filter

221

Rolling Guidance

Guidance for the 1st
iteration

Rolling Guidance

Guidance for the 2nd
iteration

Rolling Guidance

Guidance for the 3rd
iteration

Rolling Guidance

Guidance for the 5th
iteration

Rolling Guidance

Input

Output

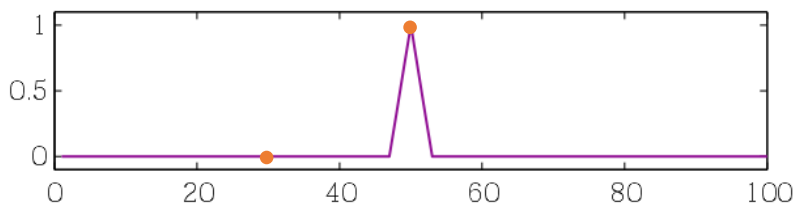Small structures are removed.
Large structure are NOT blurred.

Implementation

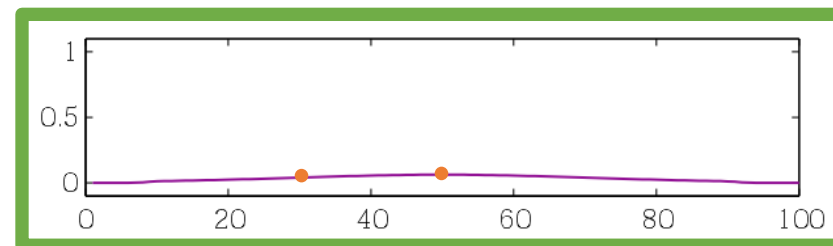Rolling Guidance Filter (RGF) has only 1 line of code

```
1  Mat rollingGuidanceFilter(Mat im, float scale, int iter){
2      Mat res = im.mul(0);
3      while(iter--) res = bilateralFilter(im,res,scale,SIGMA_R);
4      return res;
5  }
```

Small Structure

Input

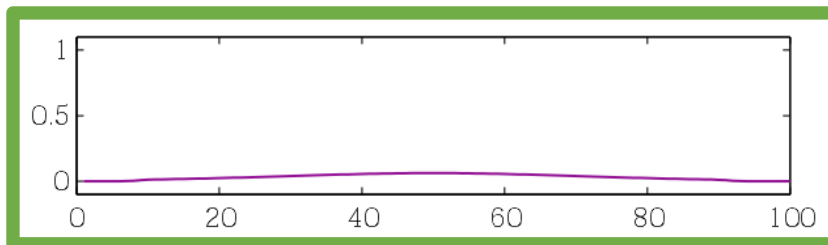

Guidance (output of step 1)

$$J^{t+1}(p) = \frac{1}{K_p} \sum_{q \in N(p)} \exp \left(- \frac{\|p - q\|^2}{2\sigma_s^2} \right.$$

Joint Bilateral Filter

It becomes a Gaussian filter

Same

228

Large Structure

Input Image

Result of Step 1

$$J^{t+1}(p) = \frac{1}{K_p} \sum_{q \in N(p)} \exp\left(-\frac{\|p-q\|^2}{2\sigma_s^2} \boxed{-\frac{\|J^t(p)-J^t(q)\|^2}{2\sigma_r^2}}\right) I(q)$$
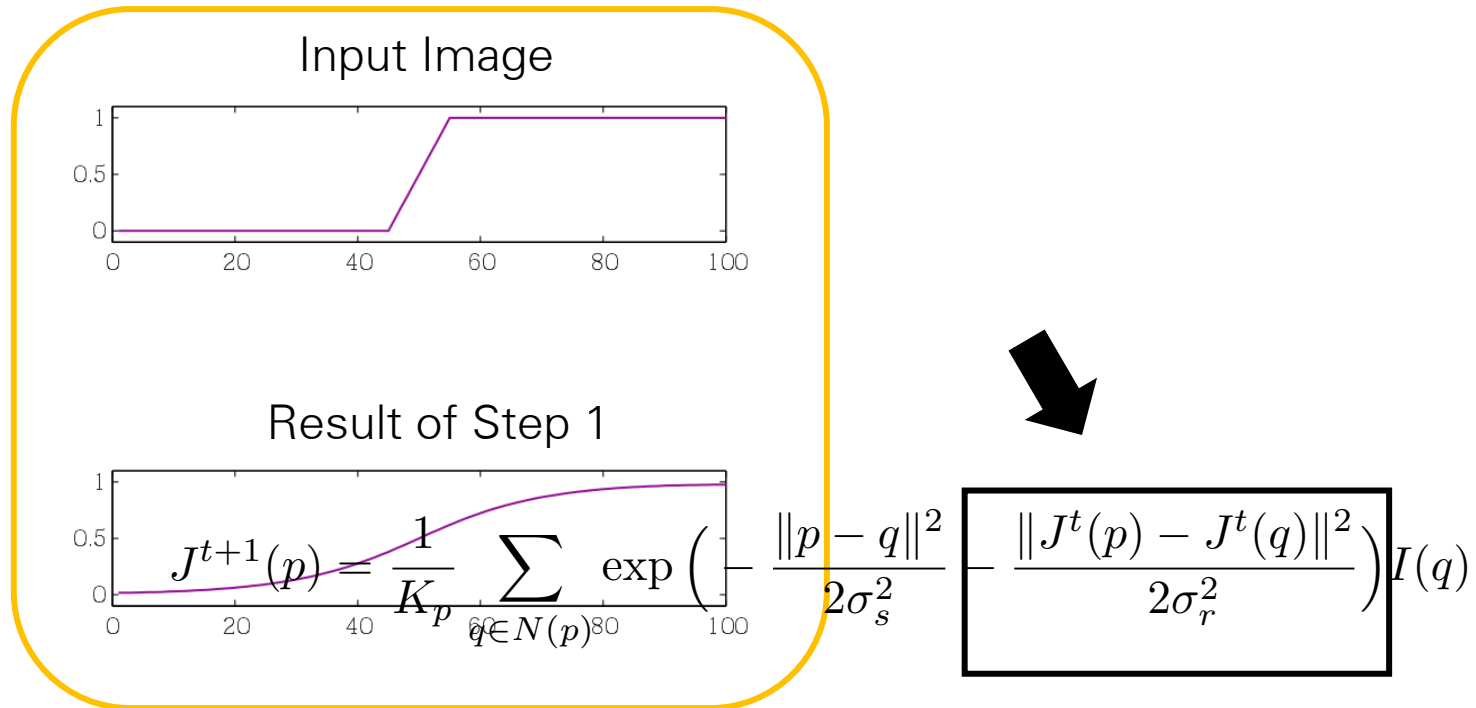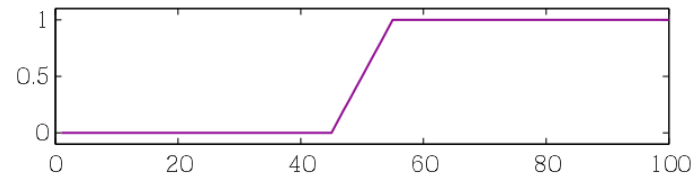
Due to this range weight
It generates sharper results than Gaussian!

Processing

Input Image

Guidance Image

Joint Bilateral Filter

Input

Intermediate iterations
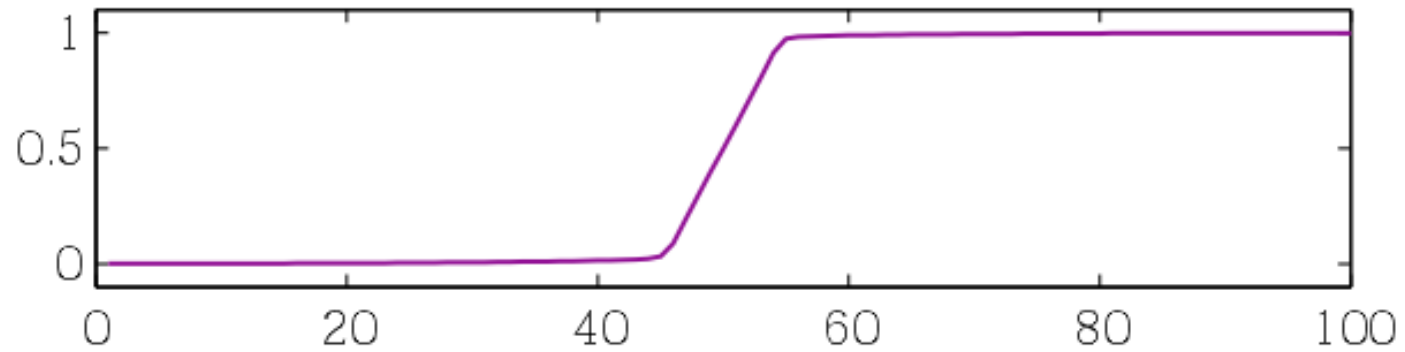
Previous guidance image

Smoothing

Range weight
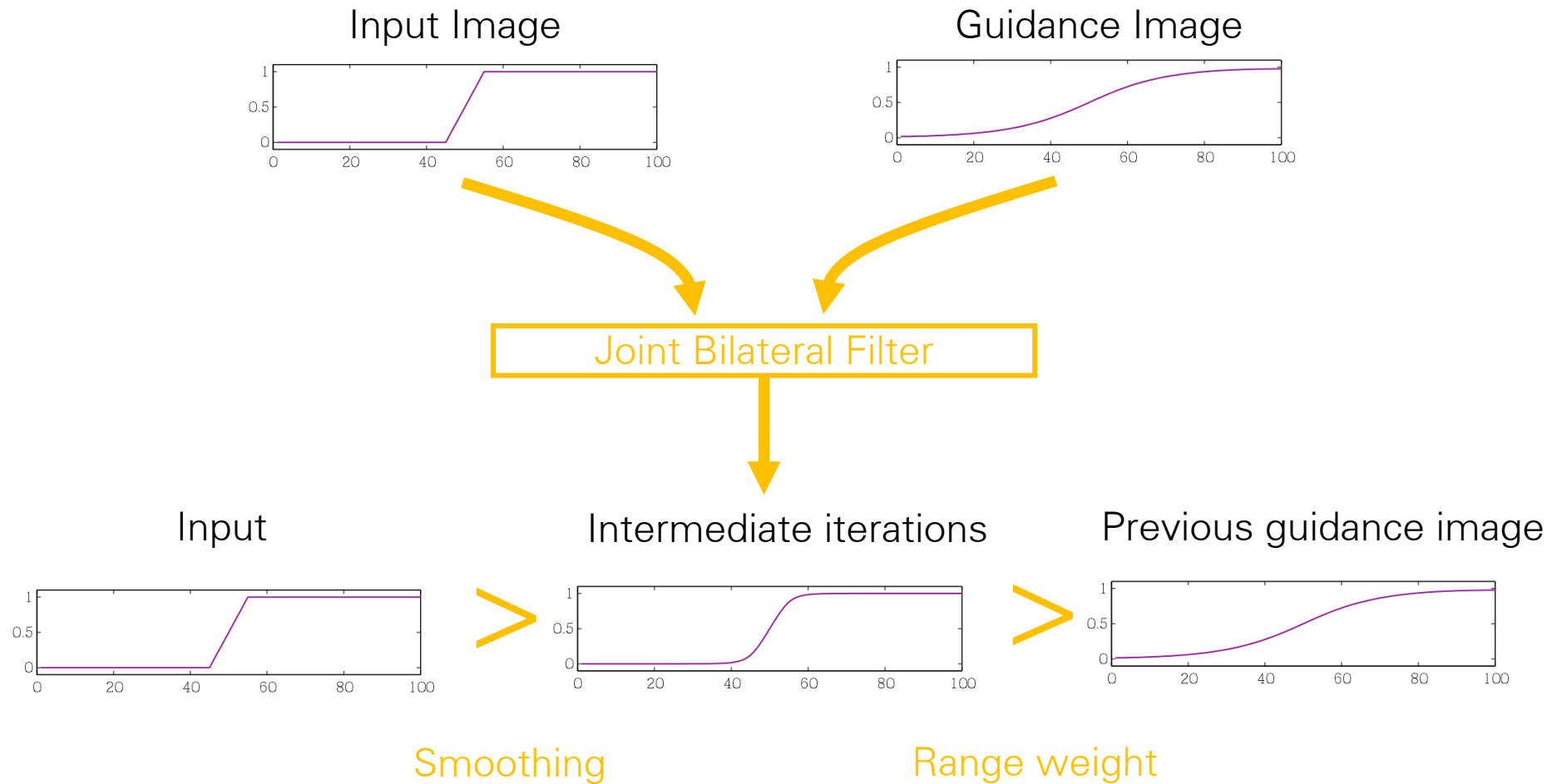
Processing

Input Image

Guidance Image

3ʳᵈ Iteration

Large Structure

Recovered
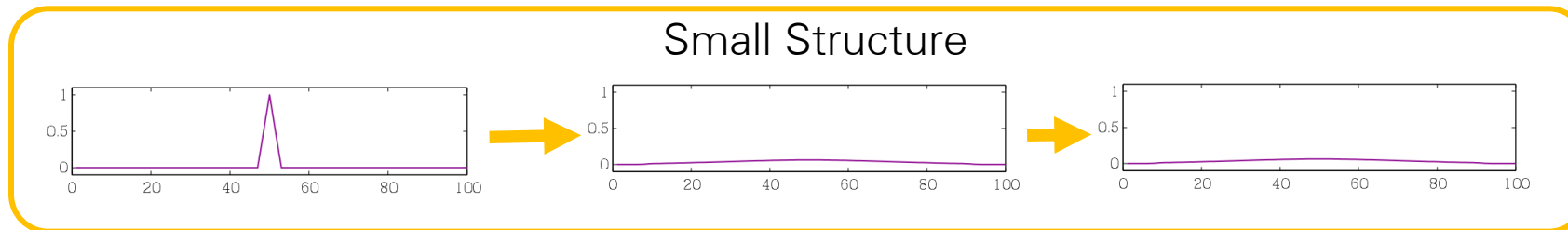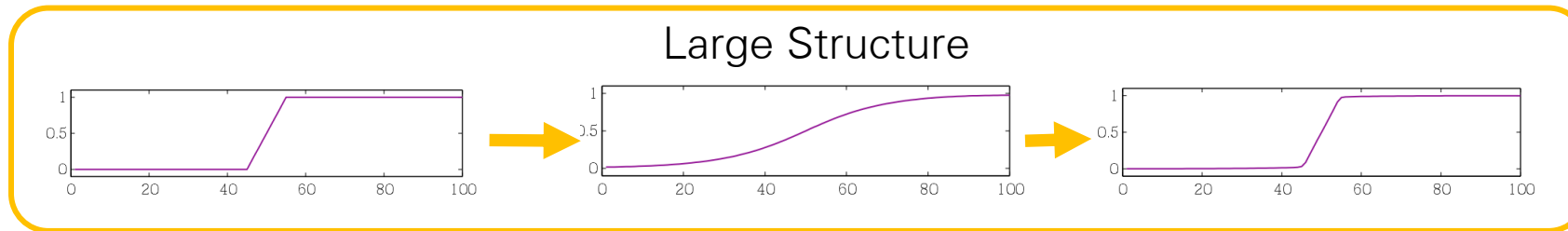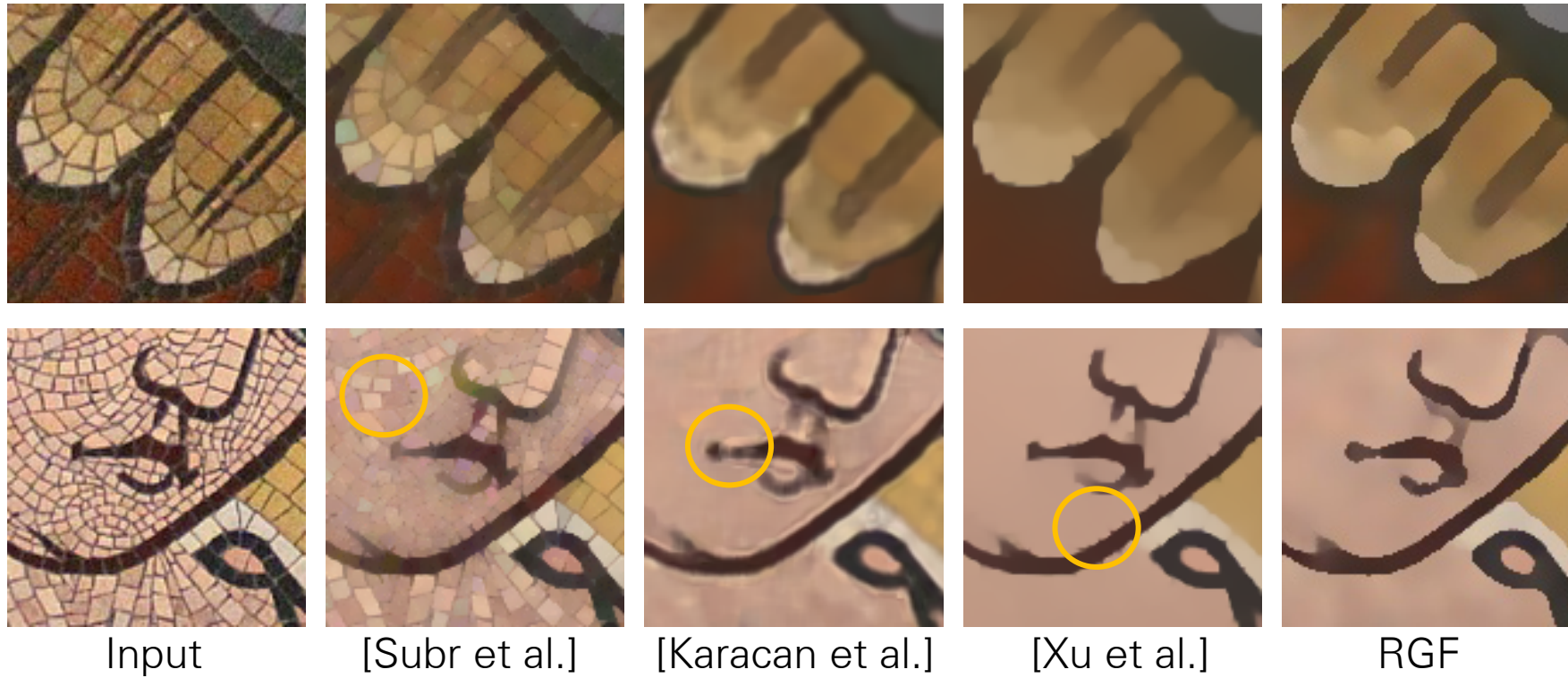
Take-home message

Rolling guidance recovers an edge as long as
it still exists in the blurred image after Gaussian smoothing.

Rolling Guidance Filter

Result Comparison

Input   [Subr et al.]   [Karacan et al.]   [Xu et al.]   RGF

Performance Comparison

Input

RGF
2013]

For 4 Megapixel Image

2

seconds

Performance Comparison

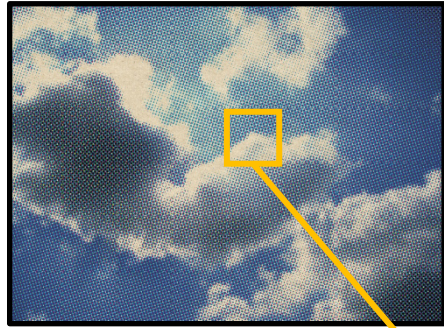| Algorithms	Time (seconds/Megapixel)
Local Extrema [Subr et al., 2009]	95
RTV [Xu et al., 2012]	14
Region Covariance [Karacan et al., 2013]	240
RGF	0.05 (Real-time)

Texture Removal

Texture Removal
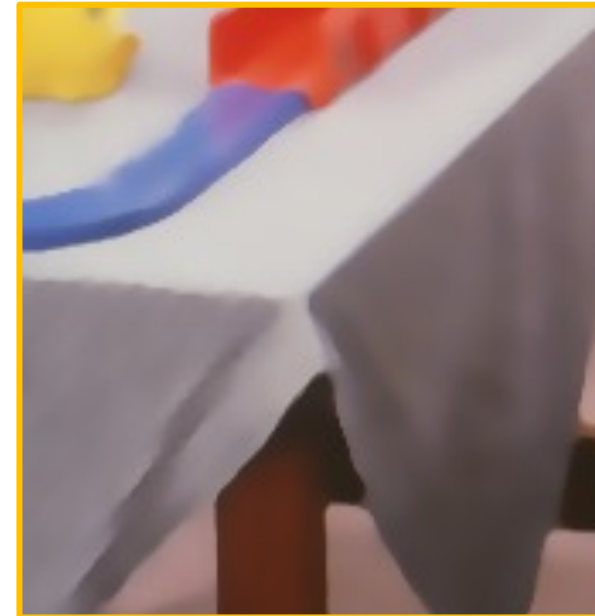
Halftone Image

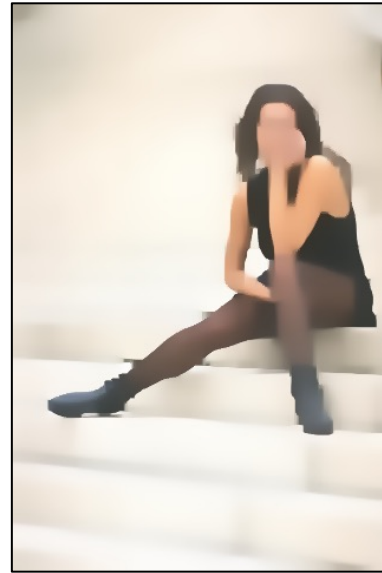# Halftone Image

Boundary detection

Input

Boundary Detection

Filtered Input

Boundary Detection

Multi-Scale Filtering

= 30

determine the scale.

Limitations

- Sharp corners could be rounded
    - It is because sharp corner presents high frequency change.
    - In other words, sharp corners are small-scale structures.

Recap

- Filtering plays a key role for many applications.

- Filtering by taking into account image content generally gives better results.

Next Lecture:
Gradient-domain image processing