Images filtered by Karacan et al., 2013

BBM444

FUNDAMENTALS OF COMPUTATIONAL PHOTOGRAPHY

Lecture #05 – Image Filtering

Erkut Erdem // Hacettepe University // Spring 2022

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

— Jiaya Jia's slides on Rolling guidance filter

Filtering

- The name “filter” is borrowed from frequency domain processing
- Accept or reject certain frequency components
- <u>Fourier (1807):</u>
  Periodic functions could be represented as a weighted sum of sines and cosines

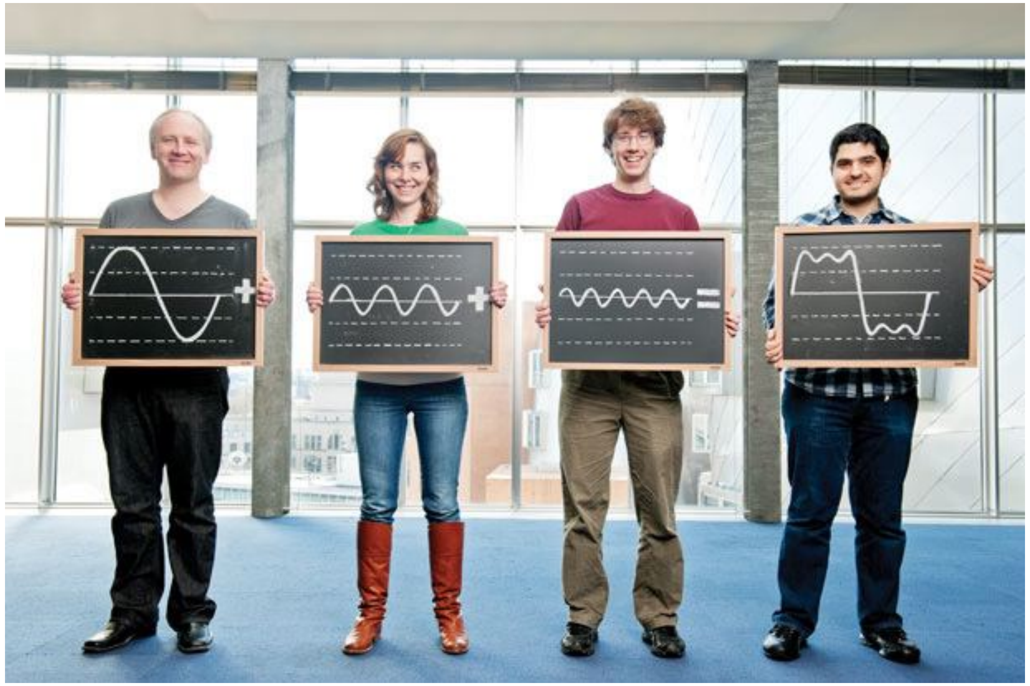
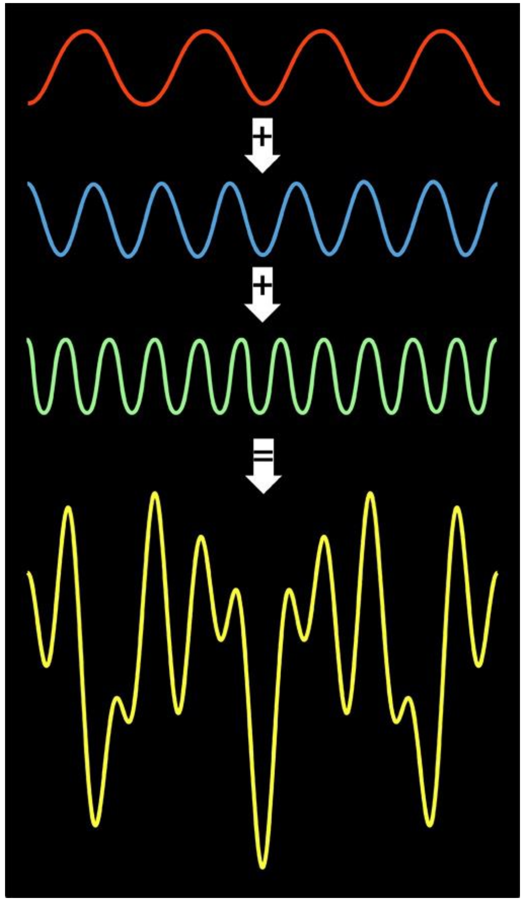


Image courtesy of Technology Review

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

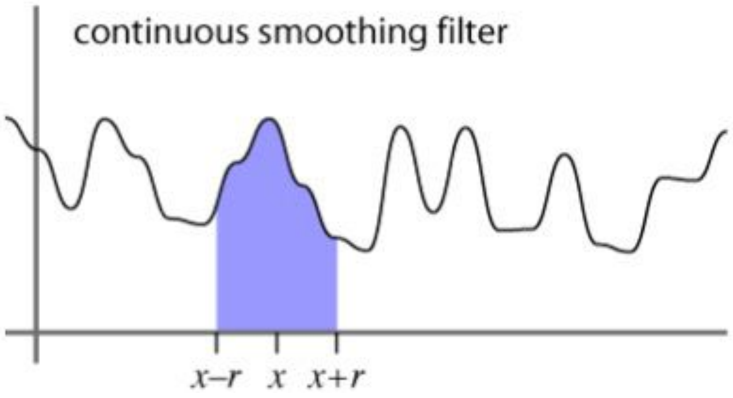
Image Filtering

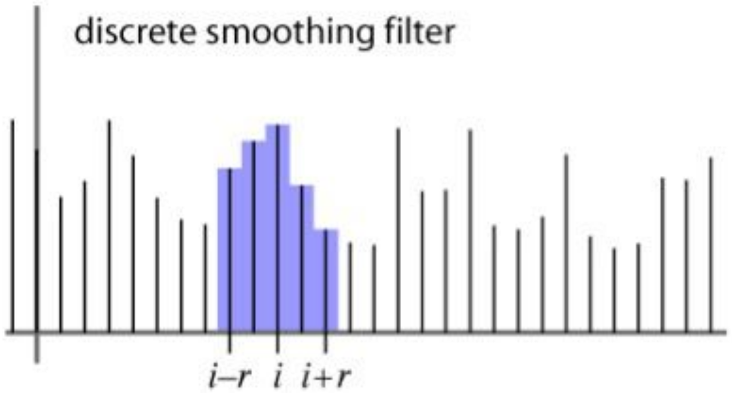
- <u>Idea:</u> Use the information coming from the neighboring pixels for processing
- Design a transformation function of the local neighborhood at each pixel in the image
  - Function specified by a "filter" or mask saying how to combine values from neighbors.
- Various uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Slide adapted from K. Grauman

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>





Discrete convolution

- Simple averaging:

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

- every sample gets the same weight

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

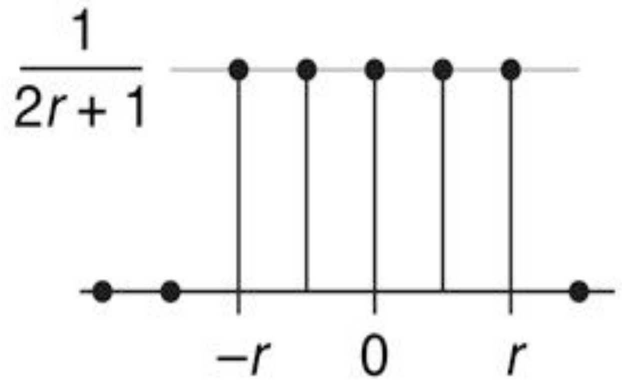
$$(a \star b)[i] = \sum_j a[j] b[i-j]$$

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

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

Filters

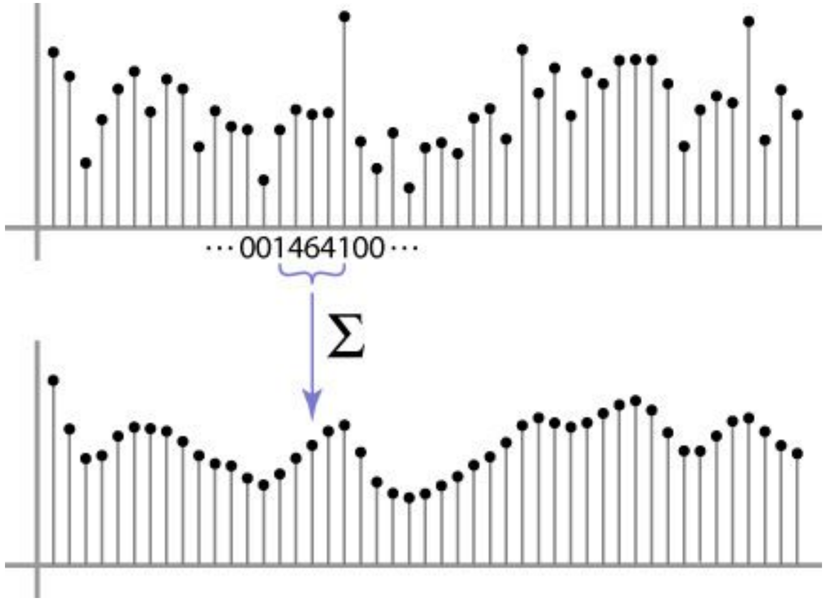
- Sequence of weights a[j] is called a filter
- Filter is nonzero over its region of support
  - usually centered on zero: support radius r
- Filter is normalized so that it sums to 1.0
  - this makes for a weighted average, not just any old weighted sum
- Most filters are symmetric about 0
  - since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



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: $(((\mathbf{a} * \mathbf{b}_1) * \mathbf{b}_2) * \mathbf{b}_3)$
- this is equivalent to applying one filter: $\mathbf{a} * (\mathbf{b}_1 * \mathbf{b}_2 * \mathbf{b}_3)$

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								

Slide credit: S. Seitz

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							

Slide credit: S. Seitz

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	

Slide credit: S. Seitz

Averaging Filter

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

$F[x, y] \otimes H[u, v] \quad G[x, y]$

$F[x, y]$

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

$H[u, v]$

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

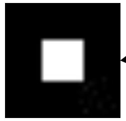
"box filter"

$G[x, y]$

	0	10	20	30	30				

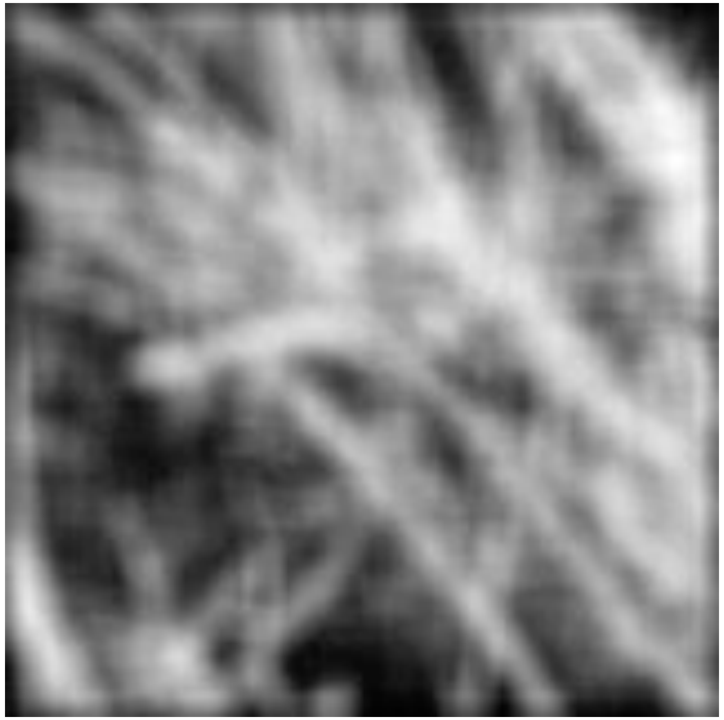
$$G = H \otimes F$$

Smoothing by averaging



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

original



filtered

Slide adapted from K. Grauman

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

$F[x, y]$

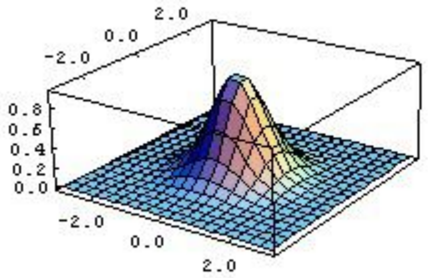
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

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

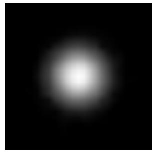
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

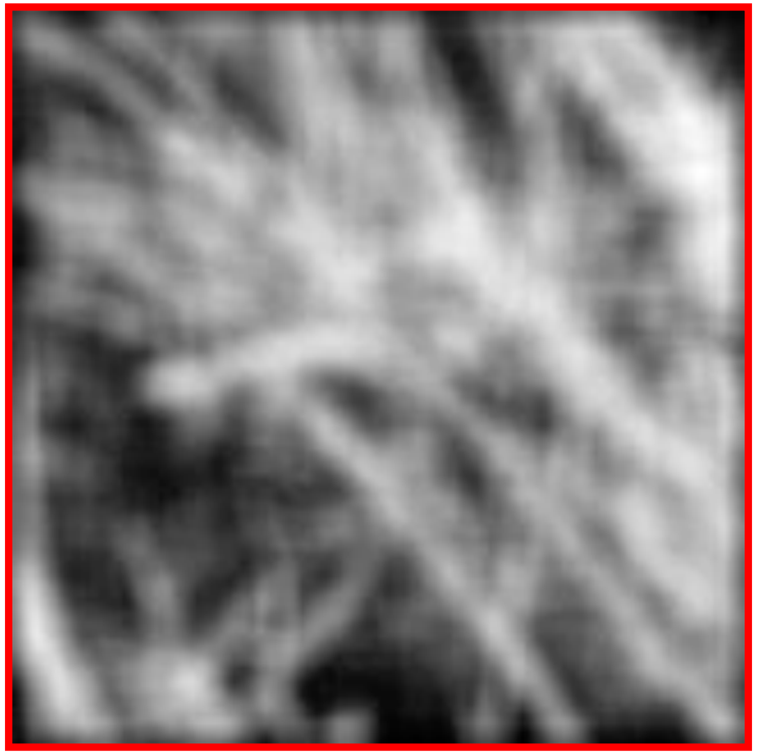


- Removes high-frequency components from the image ("low-pass filter").

Slide credit: S. Seitz

Smoothing with a Gaussian

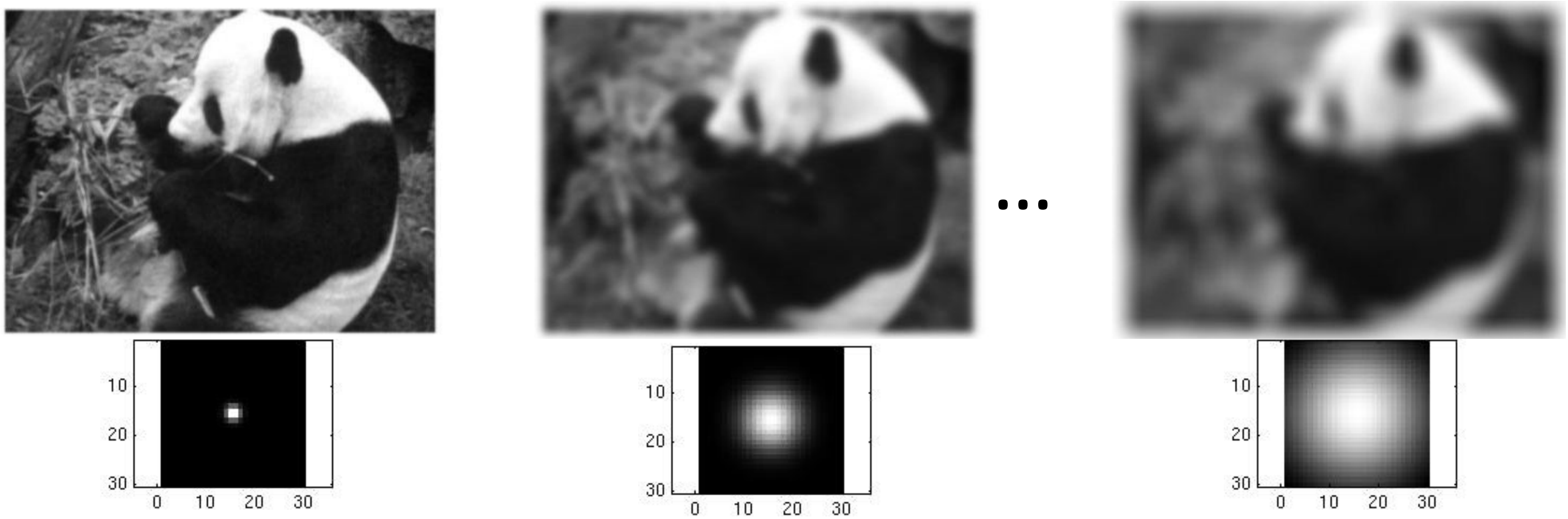




Slide adapted from K. Grauman

Smoothing with a Gaussian

Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



Slide adapted from K. Grauman

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

Slide adapted from F. Durand

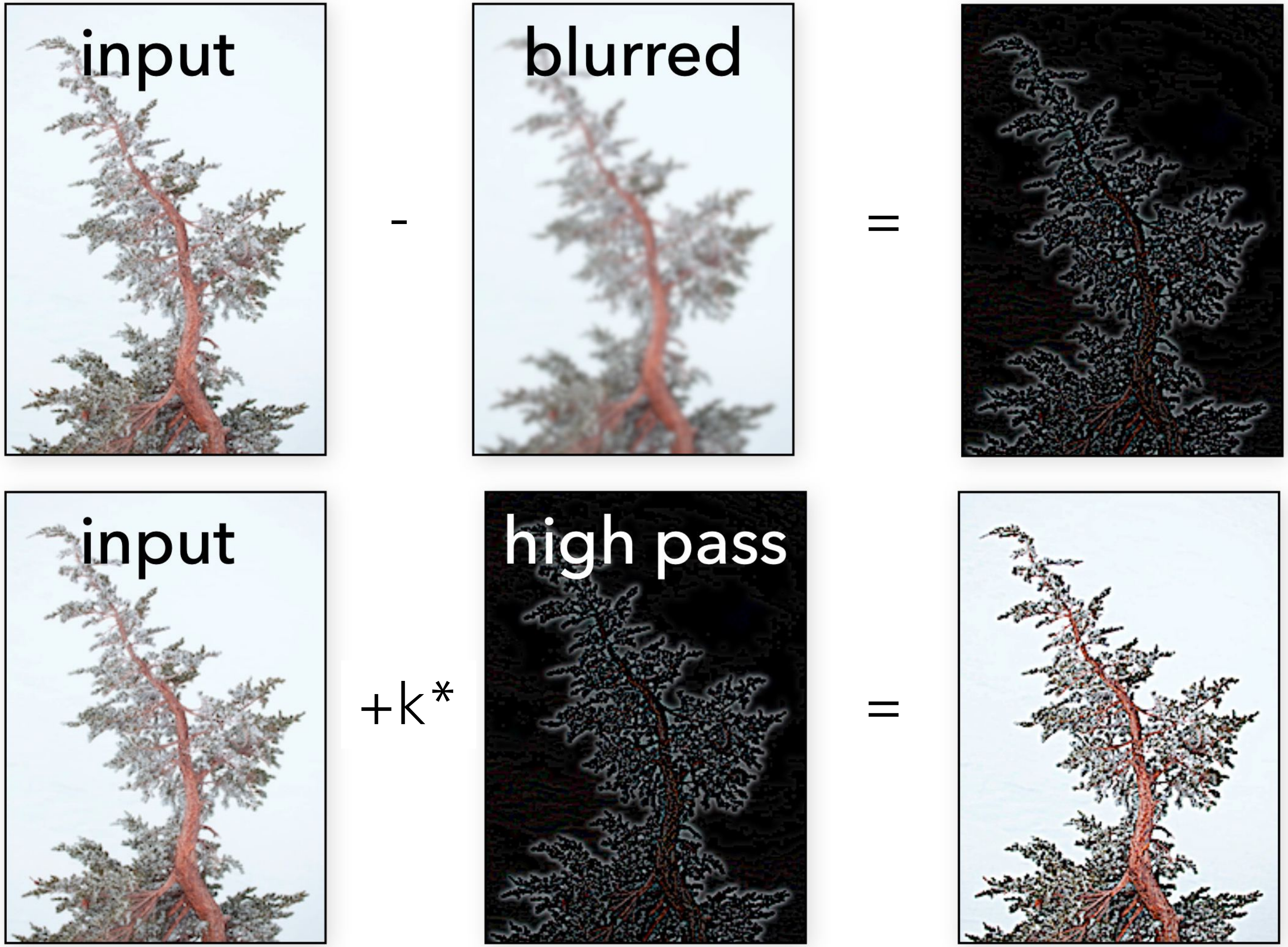
How can we sharpen?

- Blurring was easy
- Sharpening is not as obvious

- Idea: amplify the stuff not in the blurry image
- `output = input + k*(input-blur(input))`

Slide adapted from F. Durand

Sharpening



high pass

sharpened
image

Slide adapted from F. Durand

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

Slide adapted from F. Durand

Sharpening: kernel view

- Recall

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

- Denote δ the Dirac kernel (pure impulse)

$$f = f \otimes \delta$$

Slide adapted from F. Durand

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

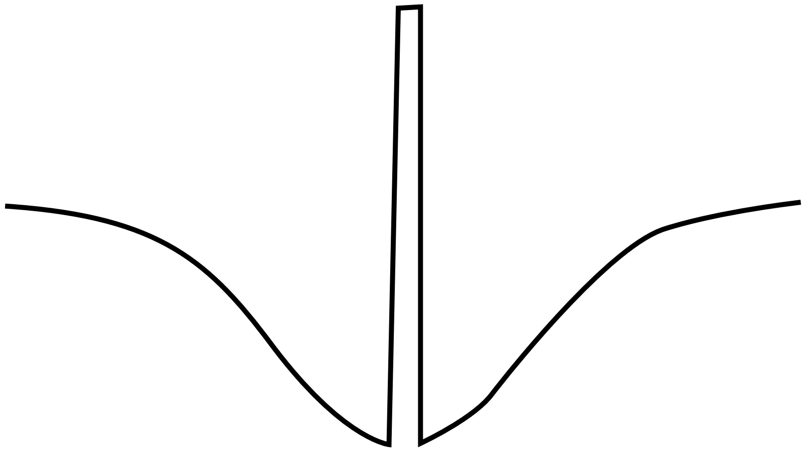
Slide adapted from F. Durand

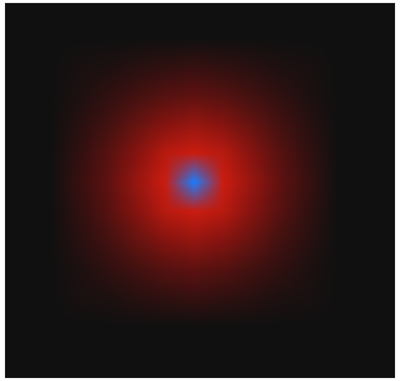
Sharpening kernel

- 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

Slide adapted from F. Durand

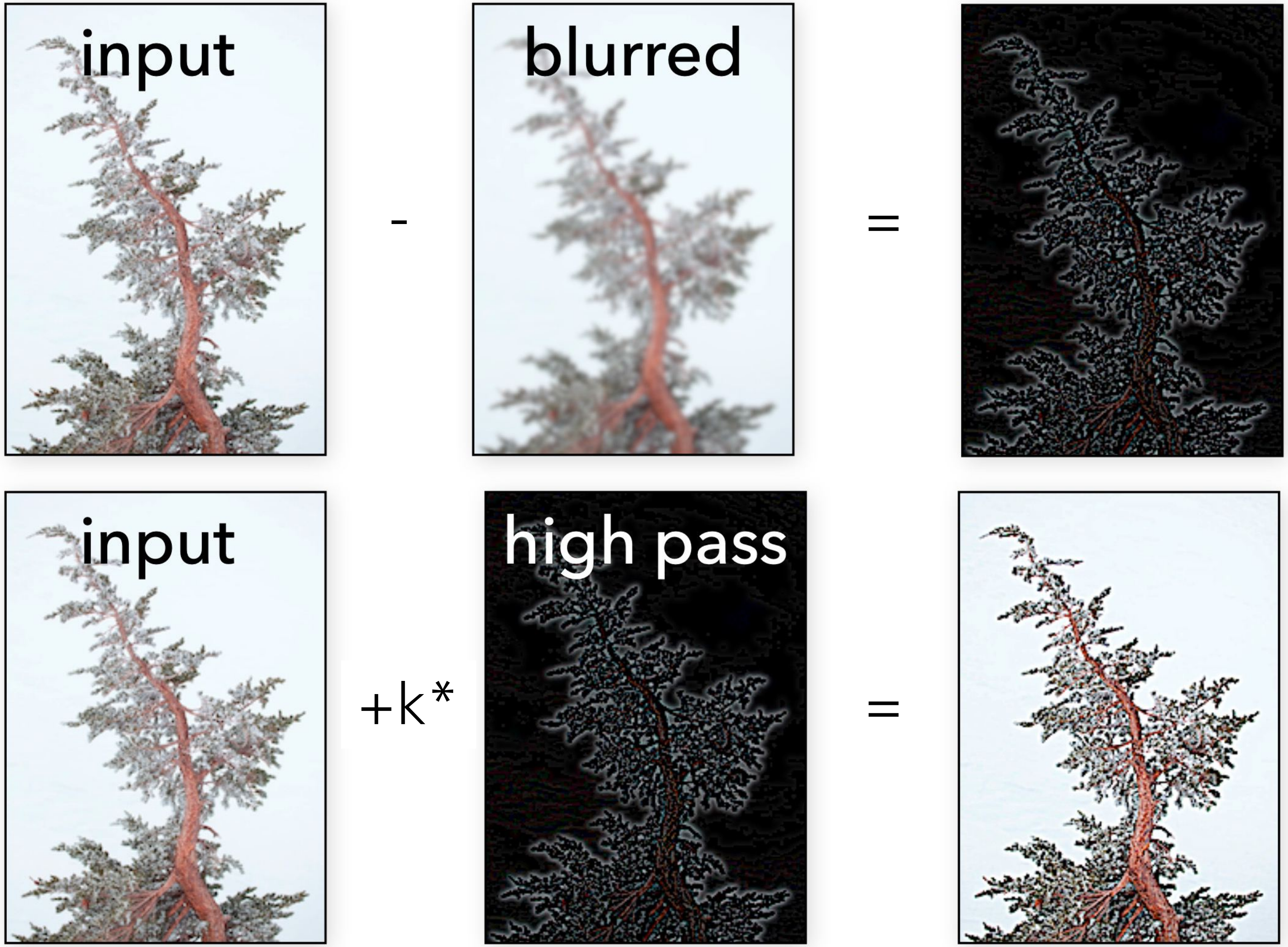
Alternate interpretation

- `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

Slide adapted from F. Durand

Sharpening



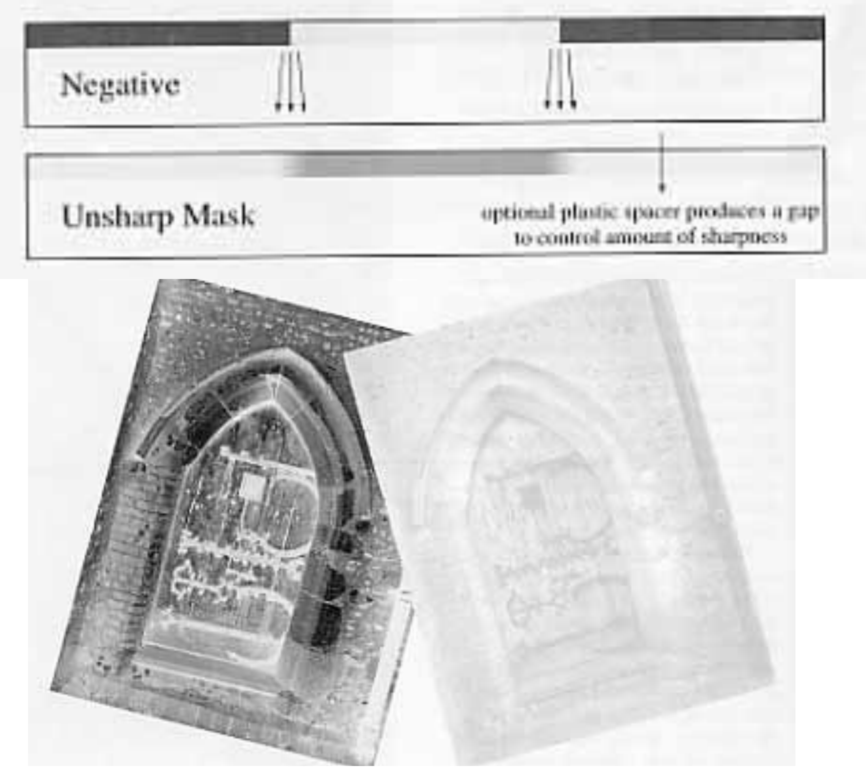
high pass

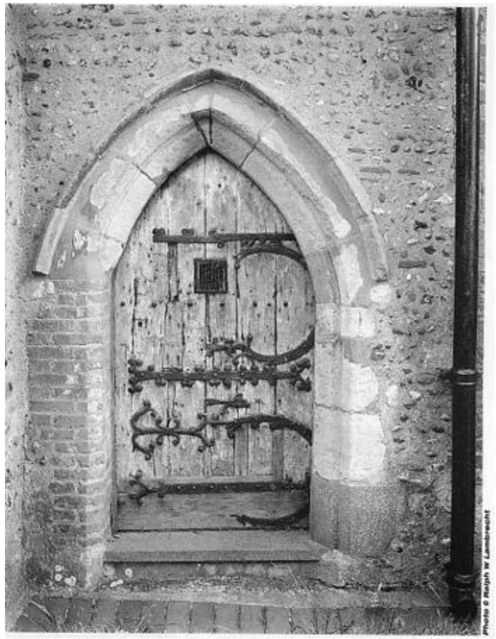
sharpened image

Slide adapted from F. Durand

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

Slide adapted from F. Durand

Unsharp mask

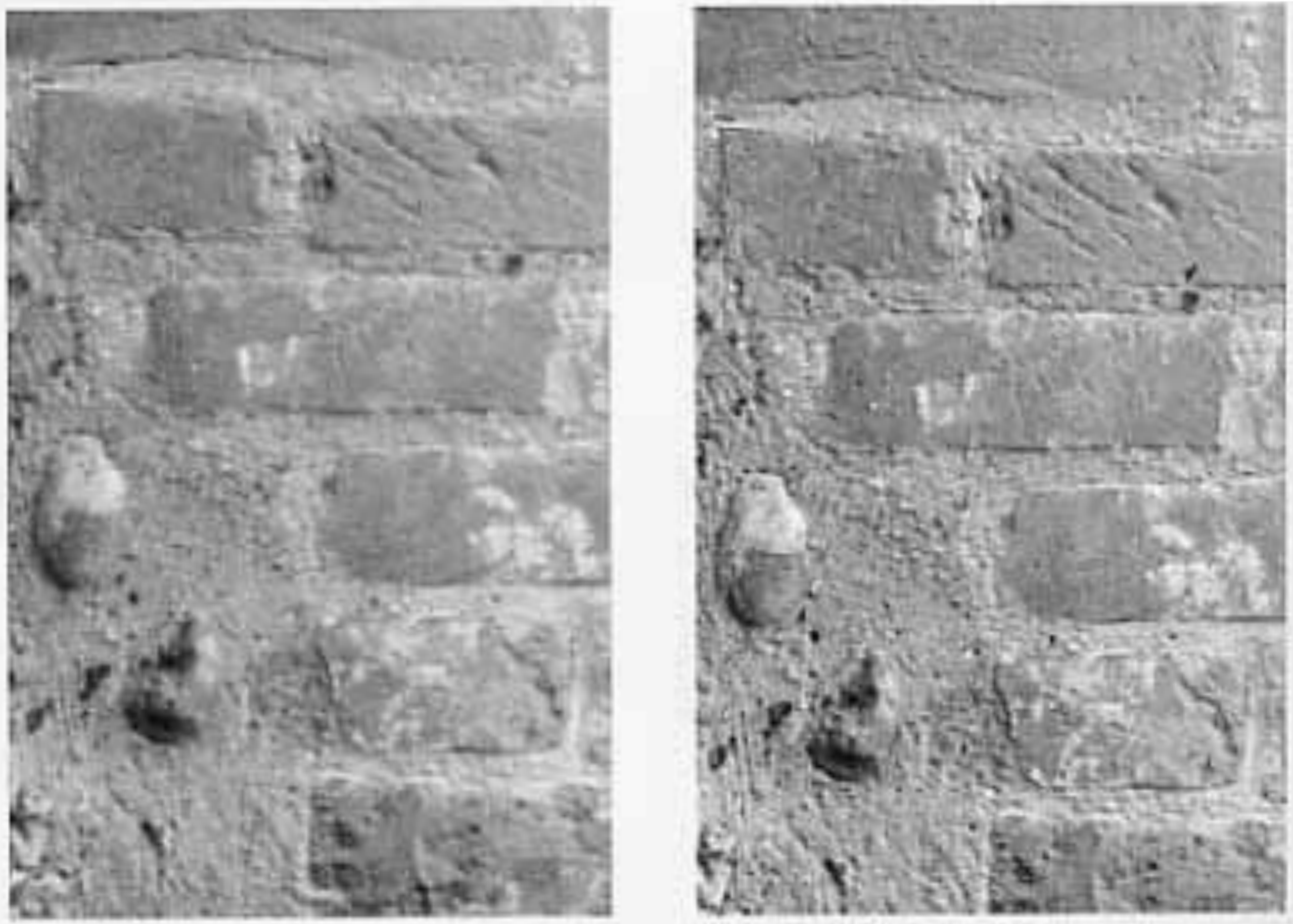


Fig.4: The two examples here show a detail of the brick-work to the left of the church door. The one on the left was printed with the negative alone – the one on the right was printed with both negative and mask as a sandwich. The increase in local contrast and edge sharpness is minute, but clearly visible.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

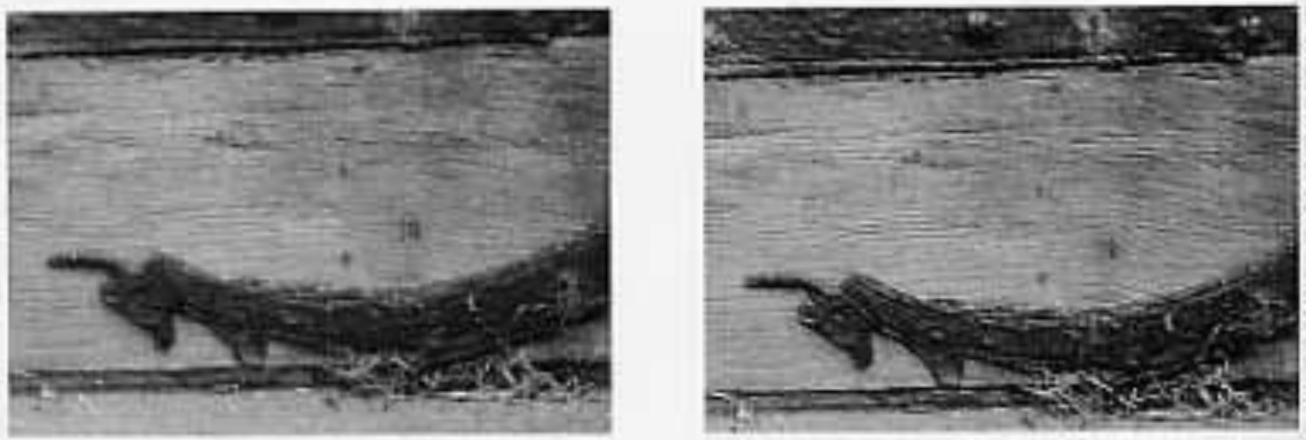


Fig.5: These two examples show a detail of the lower right hand side of the church door. Here the difference in sharpness is clearly visible between the (left) negative and (right) sandwich prints.

All photos © Ralph W Lambrecht

Problem with excess

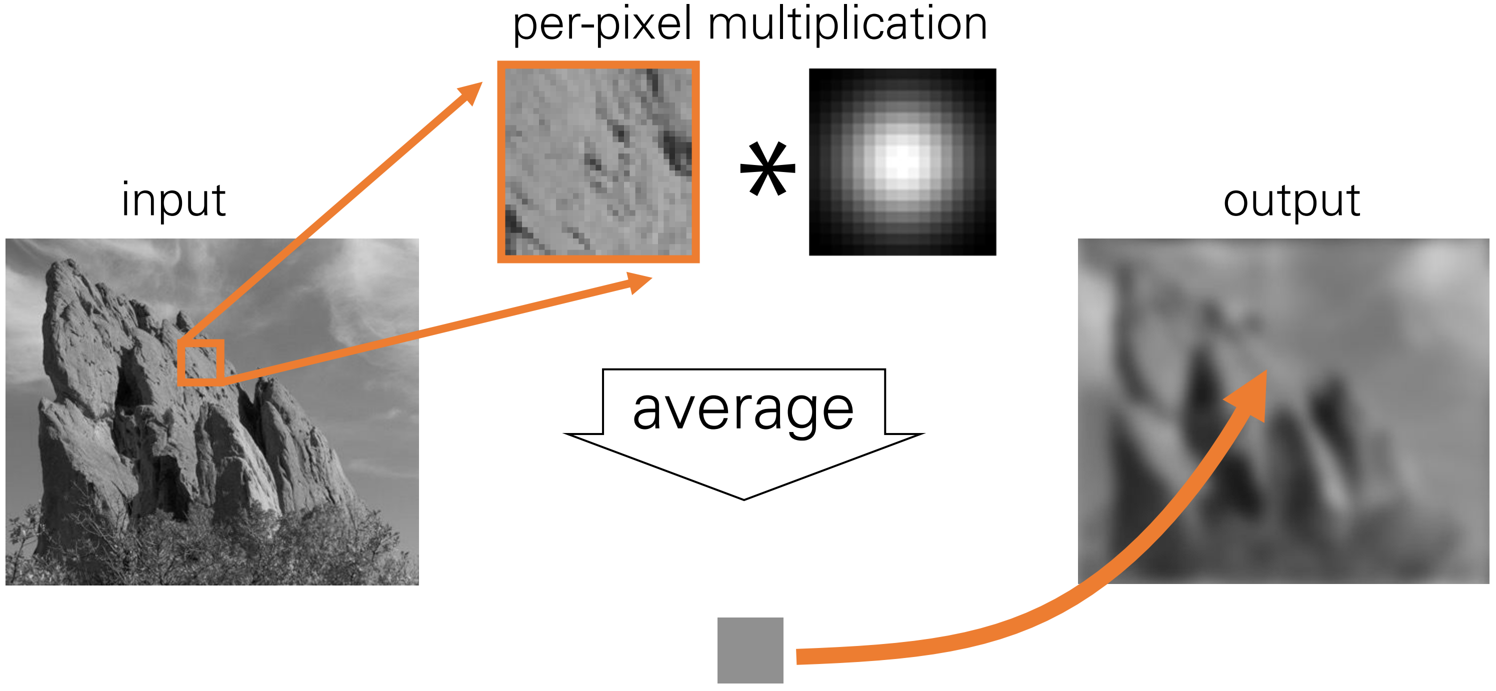
- Haloes around strong edges

Slide adapted from F. Durand

Bilateral Filter

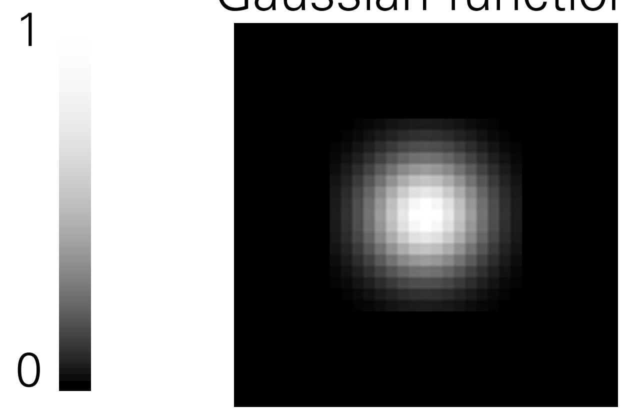
Gaussian Filter

Idea: weighted average of pixels.



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

normalized
Gaussian function

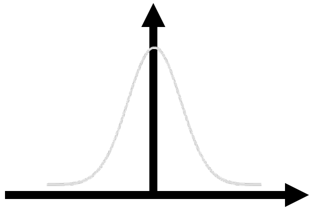


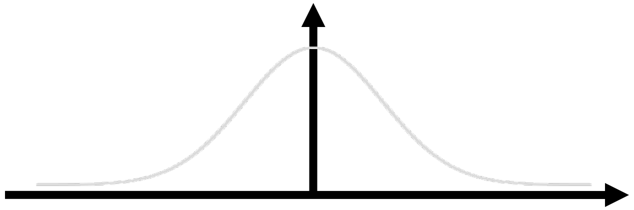
Spatial Parameter


input

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

size of the window


small σ


large σ


limited smoothing


strong smoothing

Properties of Gaussian Blur

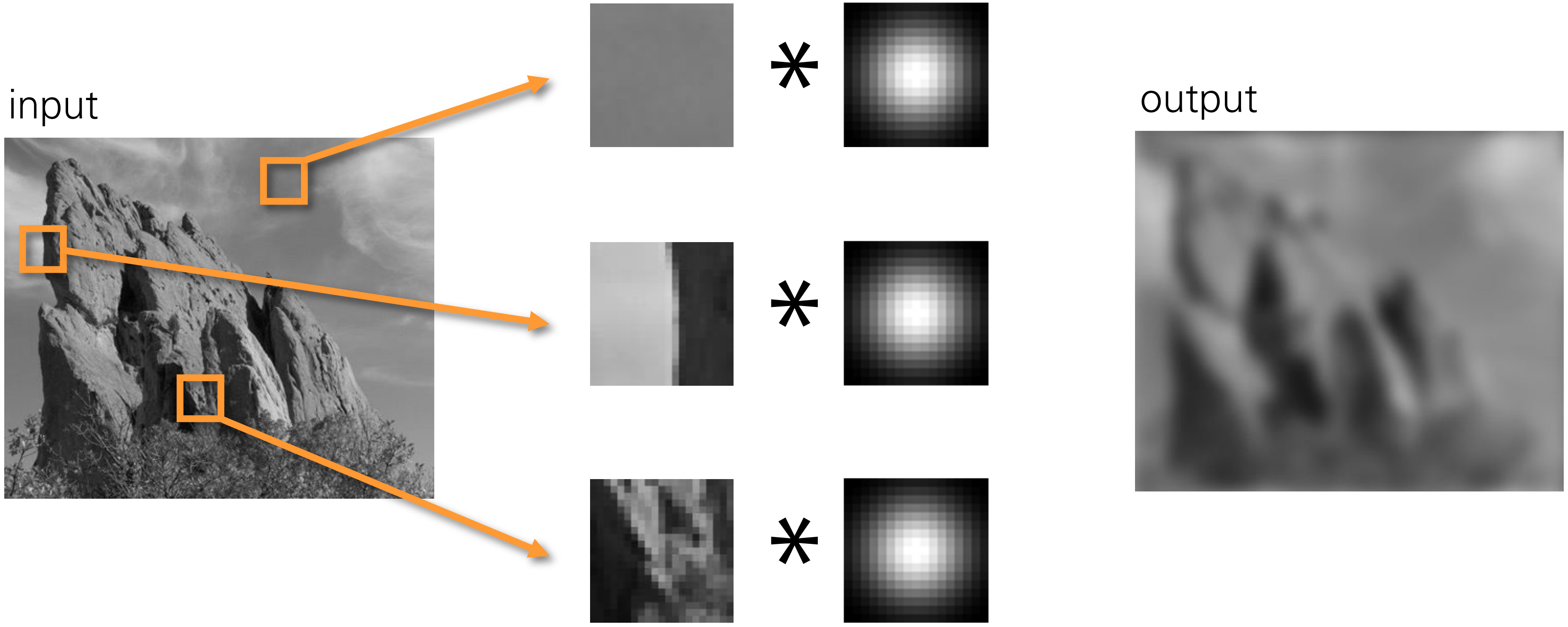
- Weights independent of spatial location
  - linear convolution
  - well-known operation
  - efficient computation (recursive algorithm, FFT…)
- Does smooth images
- But smoothes too much:
  edges are blurred.
  - Only spatial distance matters
  - No edge term

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

input

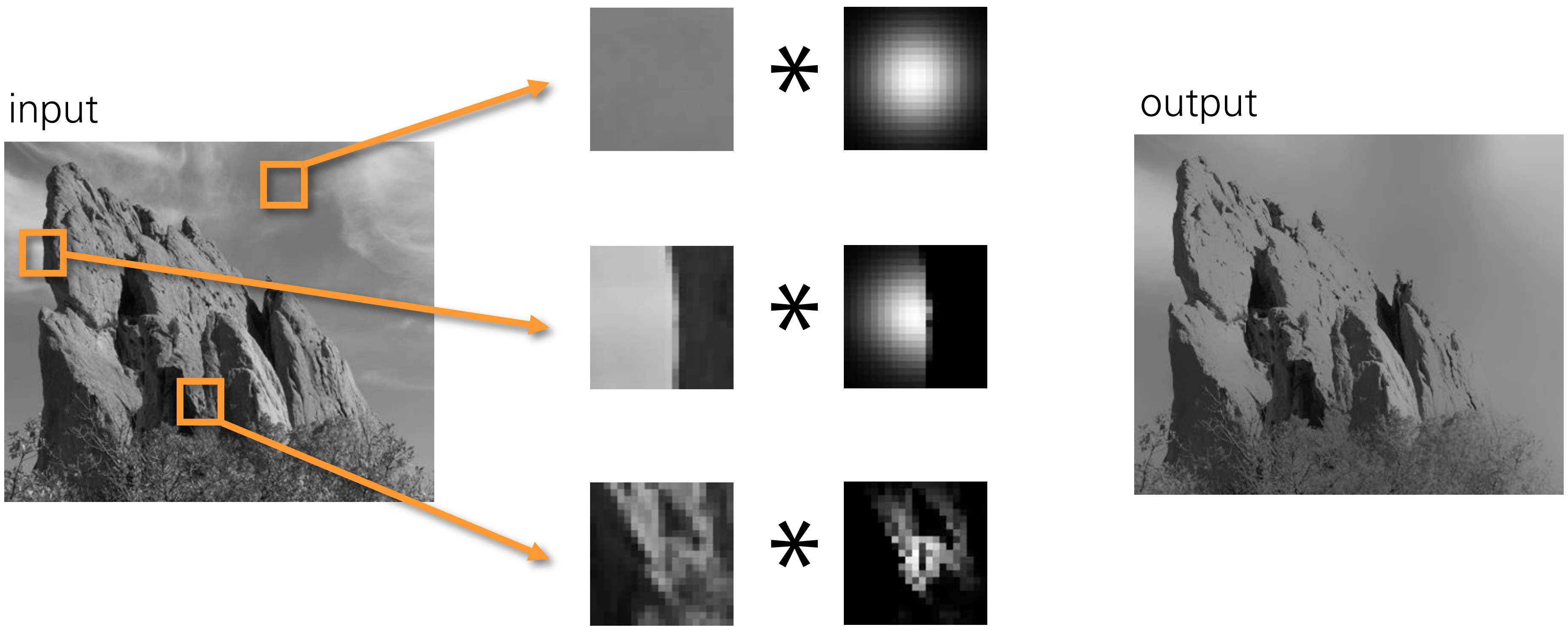
output

Blur Comes from Averaging across Edges



Same Gaussian kernel everywhere.

Bilateral Filter: No Averaging across Edges



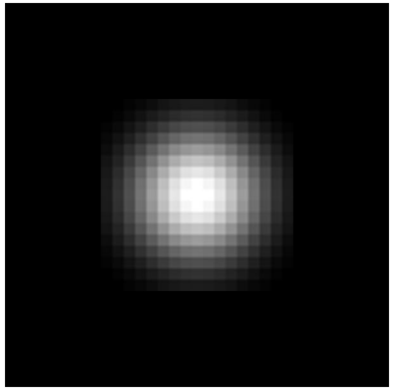
The kernel shape depends on the image content.

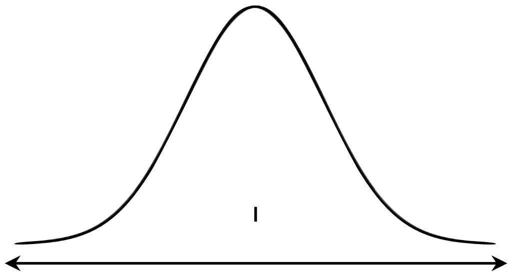
[Aurich 95, Smith 97, Tomasi 98]

Bilateral Filter: An Additional Edge Term

Same idea: weighted average of pixels.

$$BF[I]_{\mathbf{p}} = \underbrace{\overbrace{\frac{1}{W_{\mathbf{p}}}}^{\text{new}}}_{\text{normalization factor}} \sum_{\mathbf{q}\in S} \underbrace{\overbrace{G_{\sigma_s}(\| \mathbf{p}-\mathbf{q} \|)}^{\text{not new}}}_{\text{space weight}} \underbrace{\overbrace{G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)}^{\text{new}}}_{\text{range weight}} I_{\mathbf{q}}$$





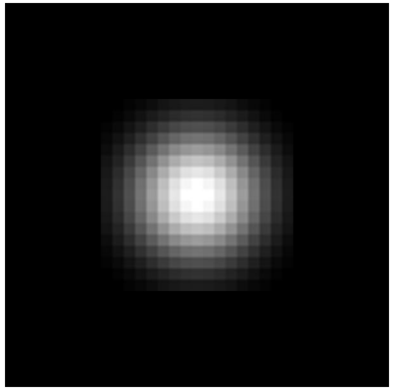
Bilateral Filter: An Additional Edge Term

Same idea: weighted average of pixels.

$$BF[I]_{\mathbf{p}} = \underbrace{\frac{1}{W_{\mathbf{p}}}}_{\text{new}} \sum_{\mathbf{q}\in S} \underbrace{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}_{\text{not new}} \underbrace{G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)}_{\text{new}} I_{\mathbf{q}}$$

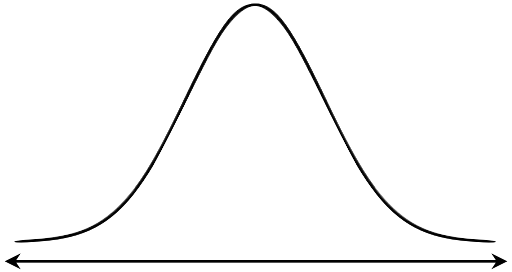
normalization factor

space weight



favor nearby pixels

range weight



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

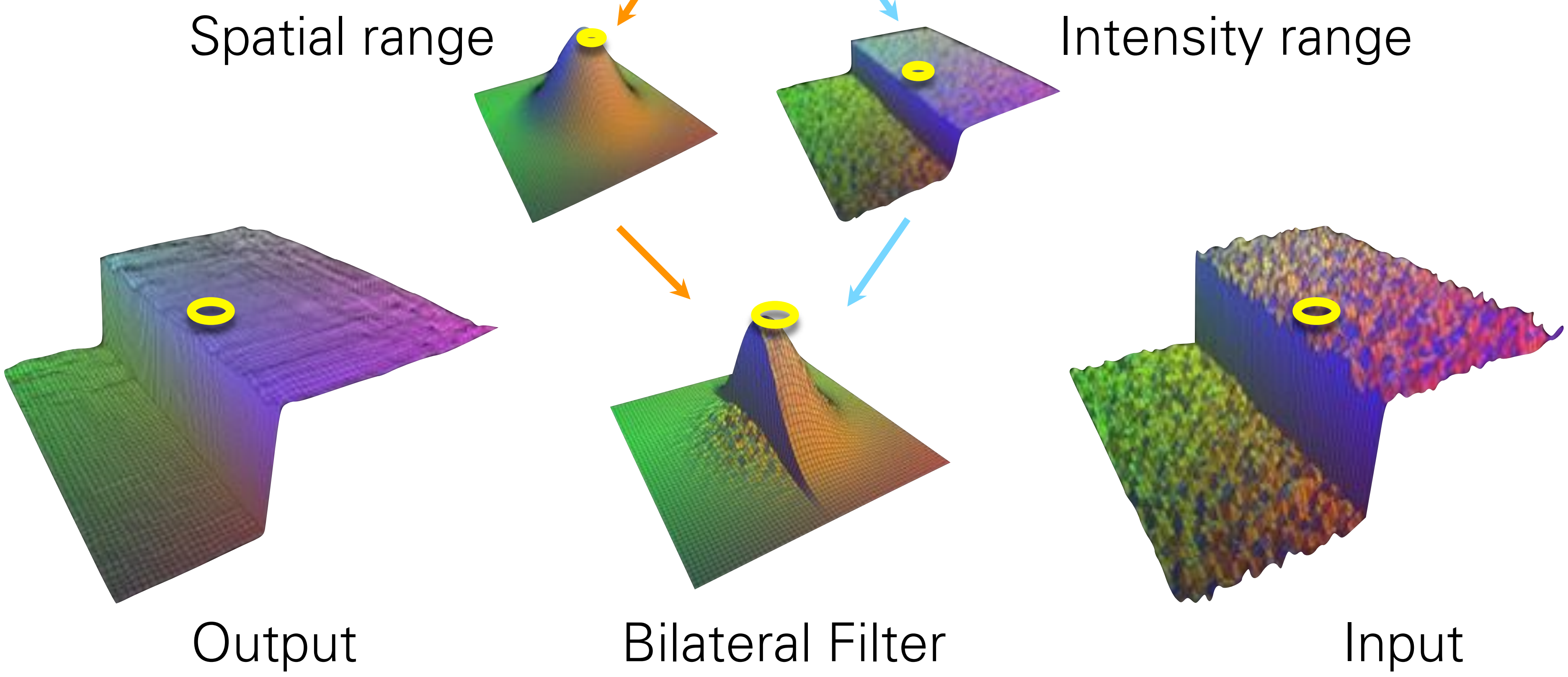
Gaussian filtering visualization

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

Output Gaussian Filter Input

Bilateral filtering visualization

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



Exploring the Parameter Space

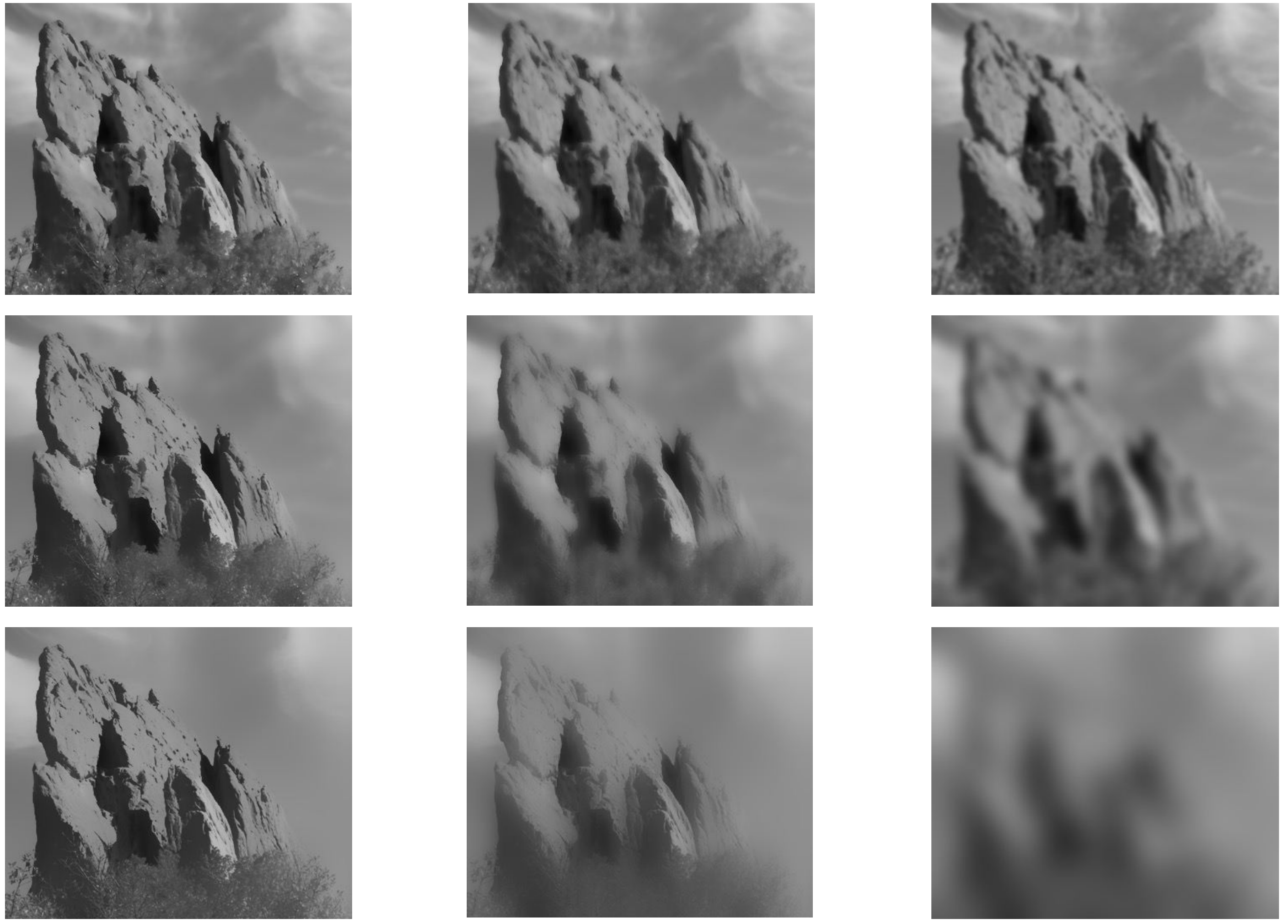
input

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

$\sigma_s = 2$

$\sigma_s = 6$

$\sigma_s = 18$



Bilateral Filtering Color Images

For gray-level images

intensity difference

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

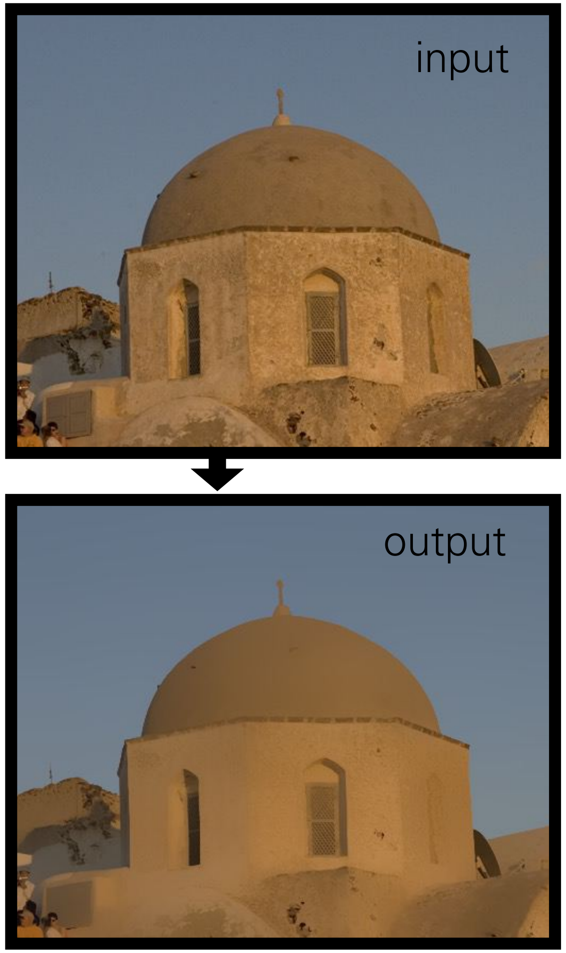
scalar

For color images

color difference

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

3D vector
(RGB, Lab)



Hard to Compute

- Nonlinear

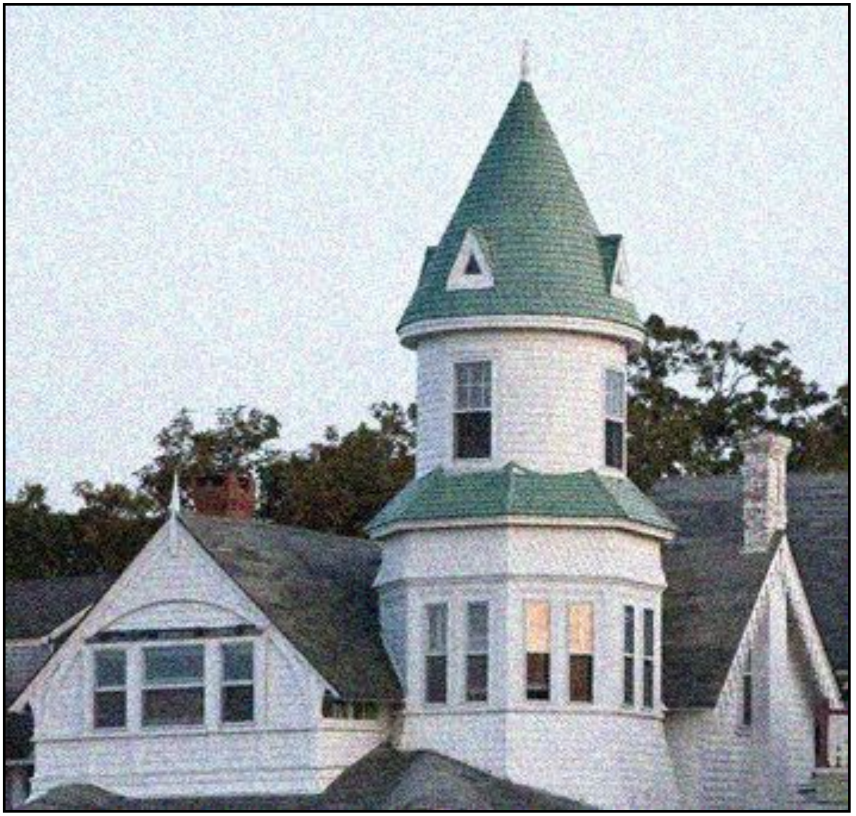
$$BF[I]_{\mathbf{p}} = \boxed{\frac{1}{W_{\mathbf{p}}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|) \boxed{G_{\sigma_r}(| I_{\mathbf{p}} - I_{\mathbf{q}} |)} I_{\mathbf{q}}$$

- Complex, spatially varying kernels
- Cannot be precomputed, no FFT…

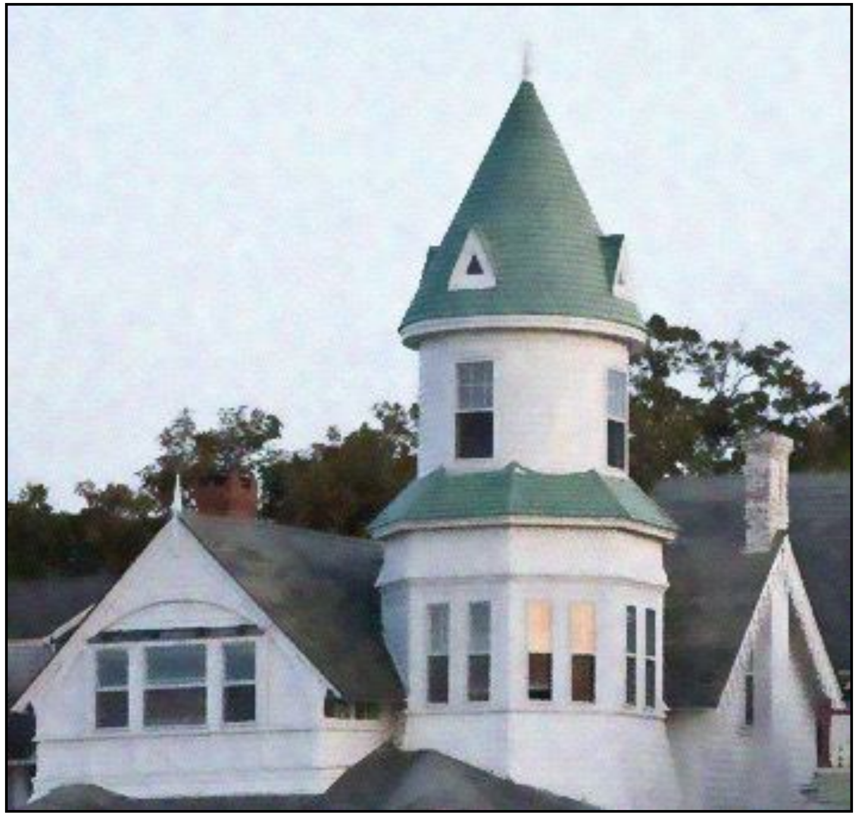
- Brute-force implementation is slow > 10min

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

Denoising



noisy input



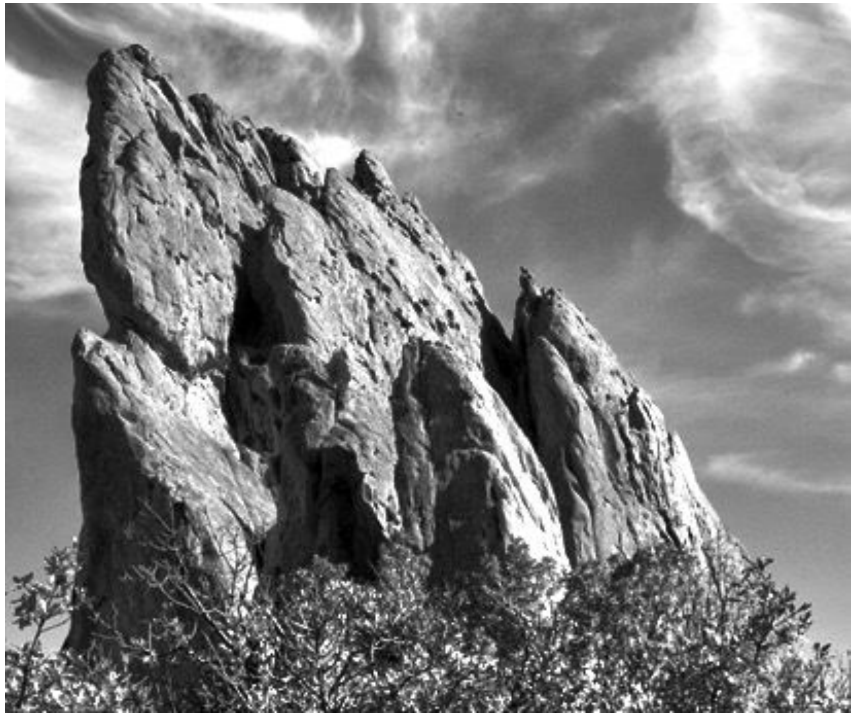
bilateral filtering

median filtering

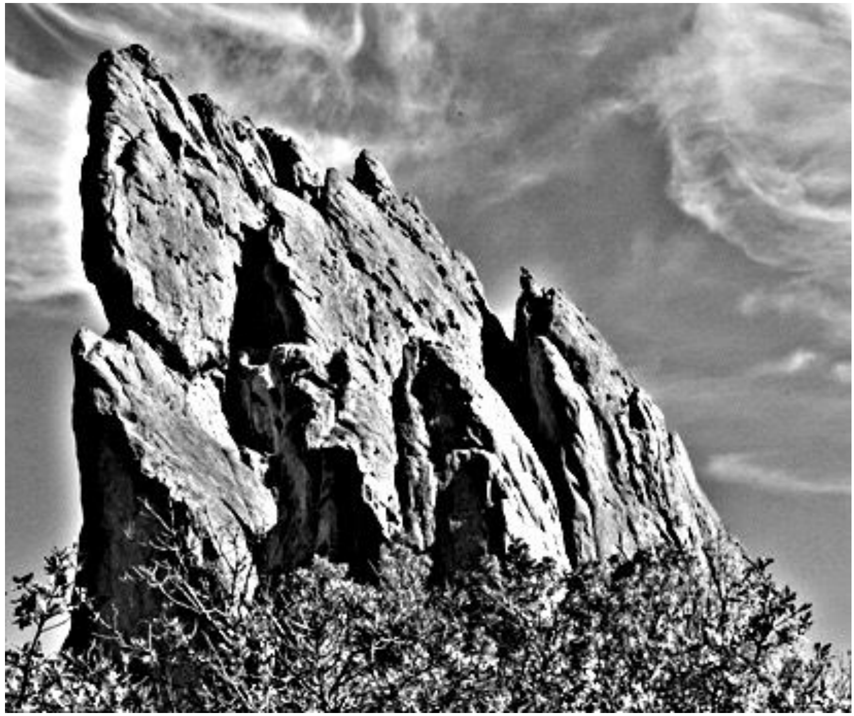
Contrast enhancement

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

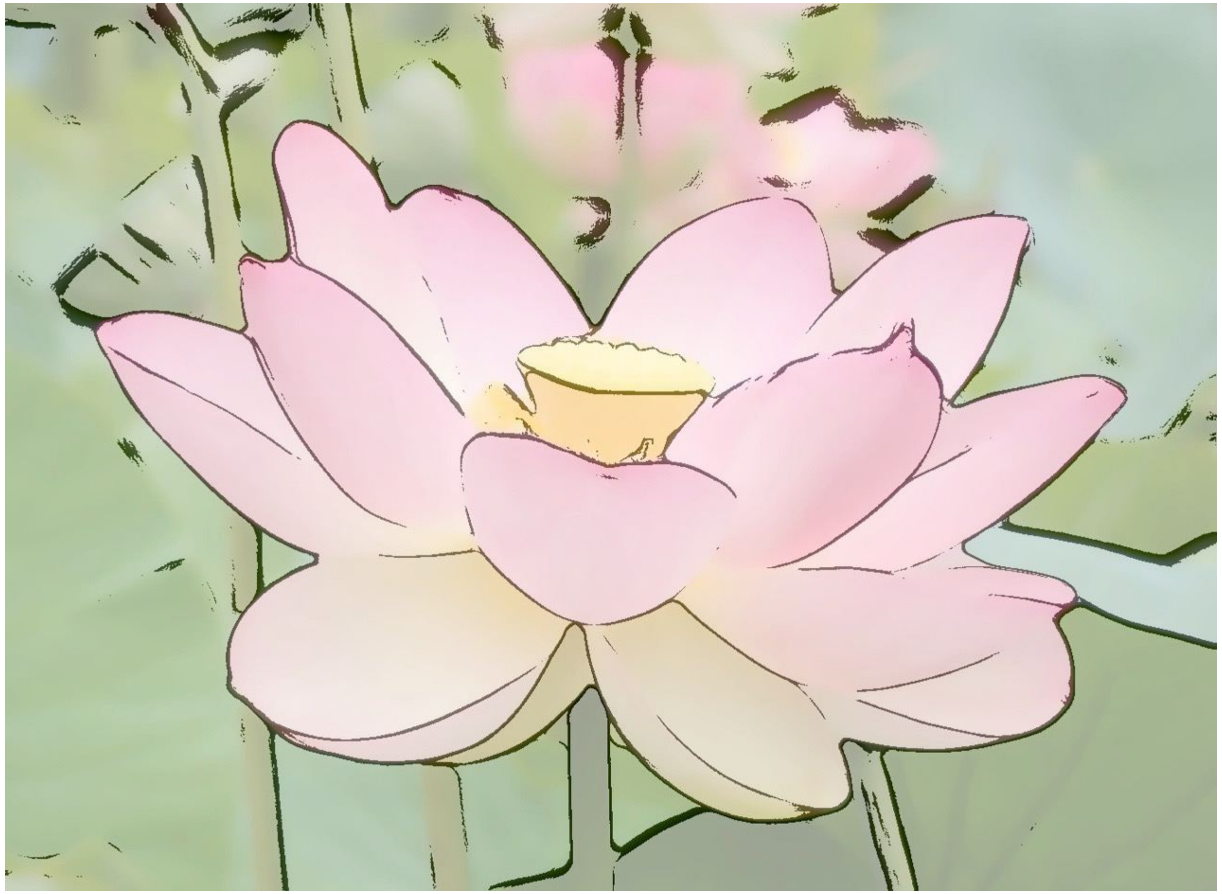
digital pore removal (aka bilateral filtering)

Cartoonization

input

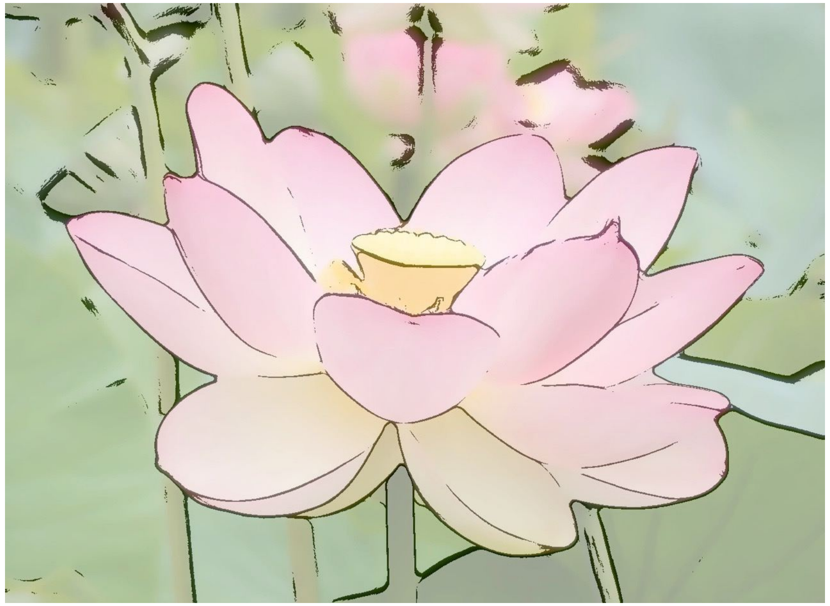
cartoon rendition

Cartoonization



How would you create this effect?

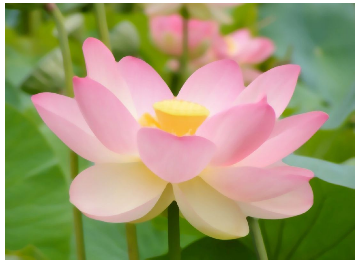
Cartoonization



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.

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*

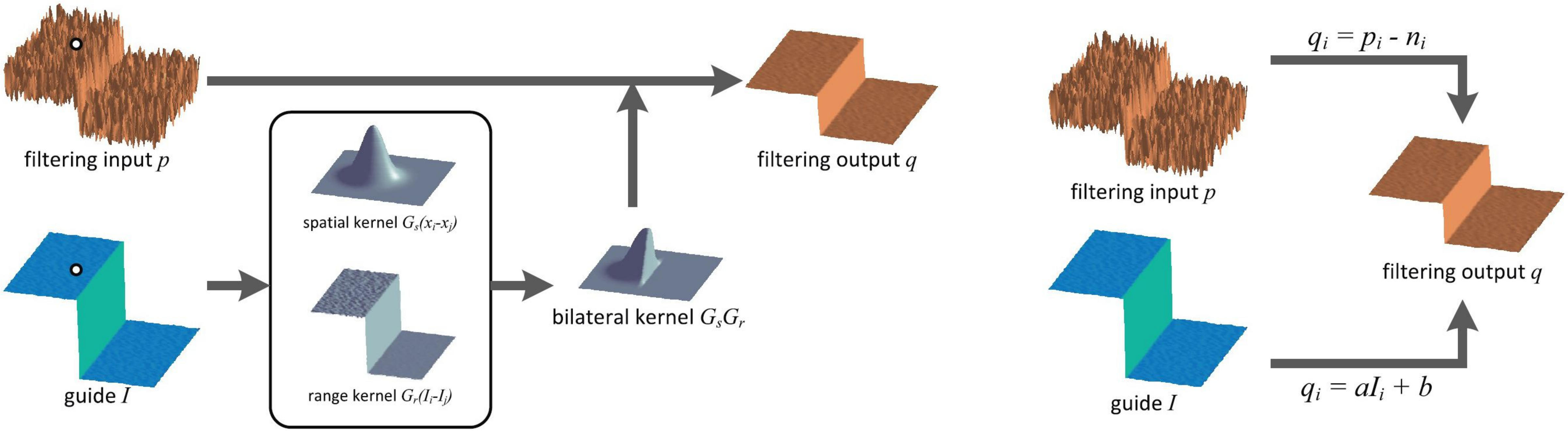
(d) *Stylization* **(e)** *Recoloring* **(f)** *Pencil drawing* **(g)** *Depth-of-field*

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*



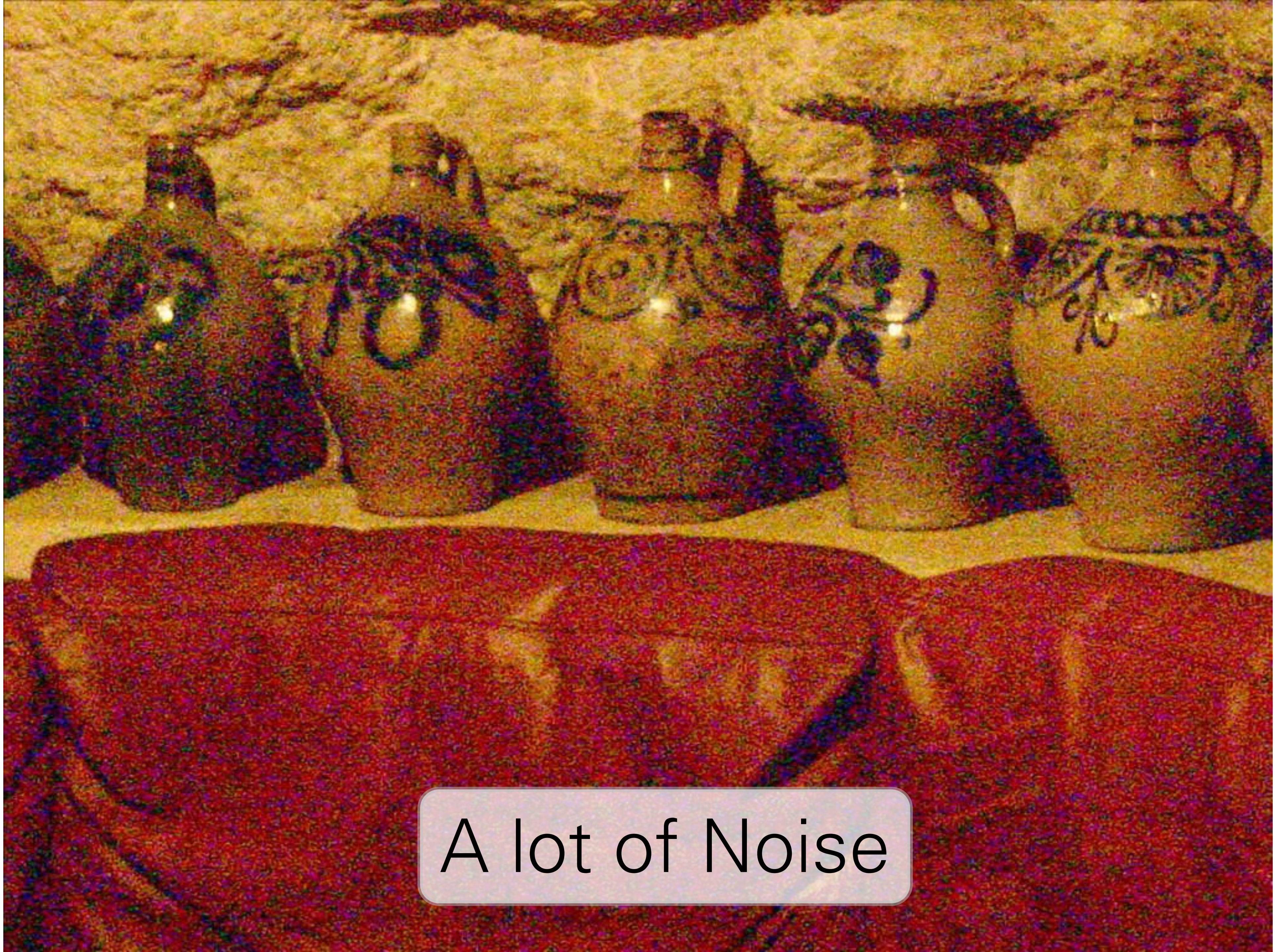
Flash/no-flash photography


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

Image acquisition

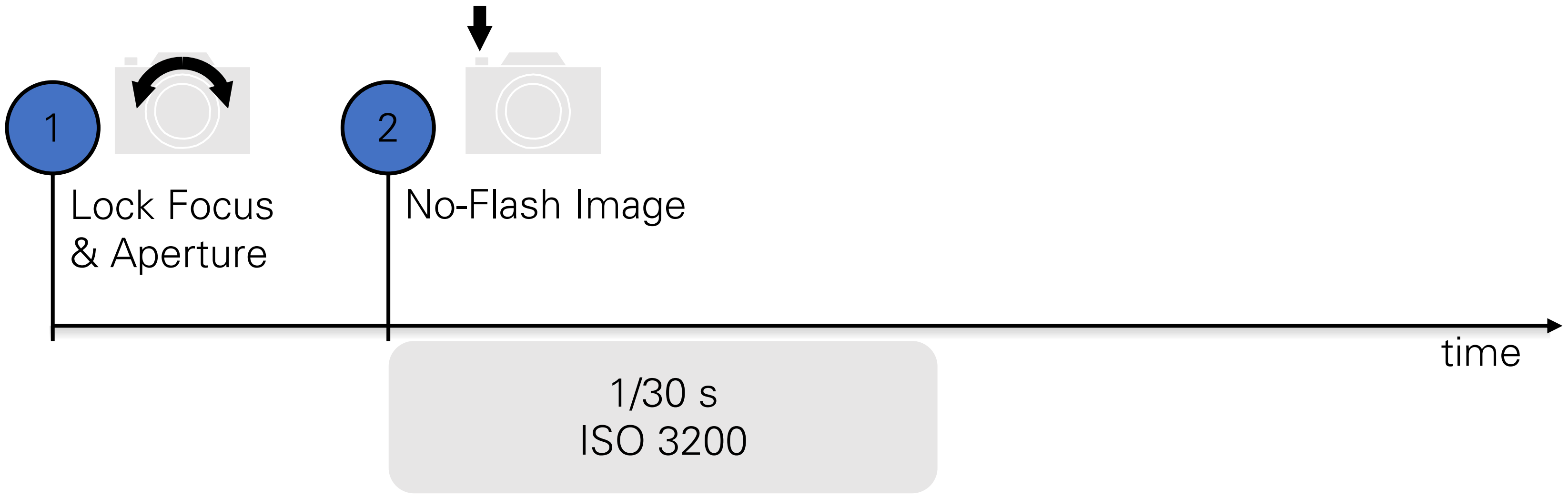
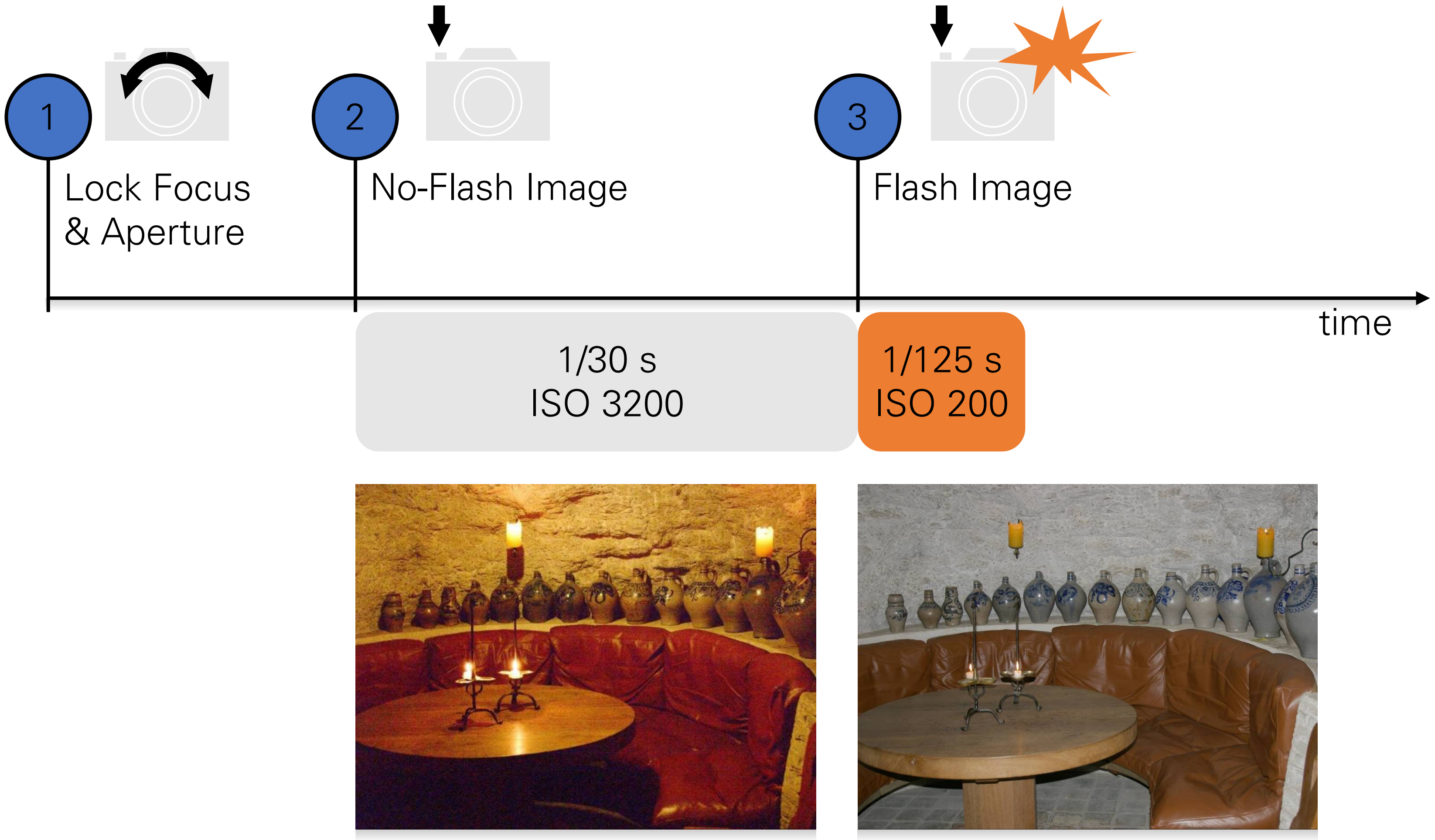


Image acquisition




Denoising Result


No-Flash


Denoising Result

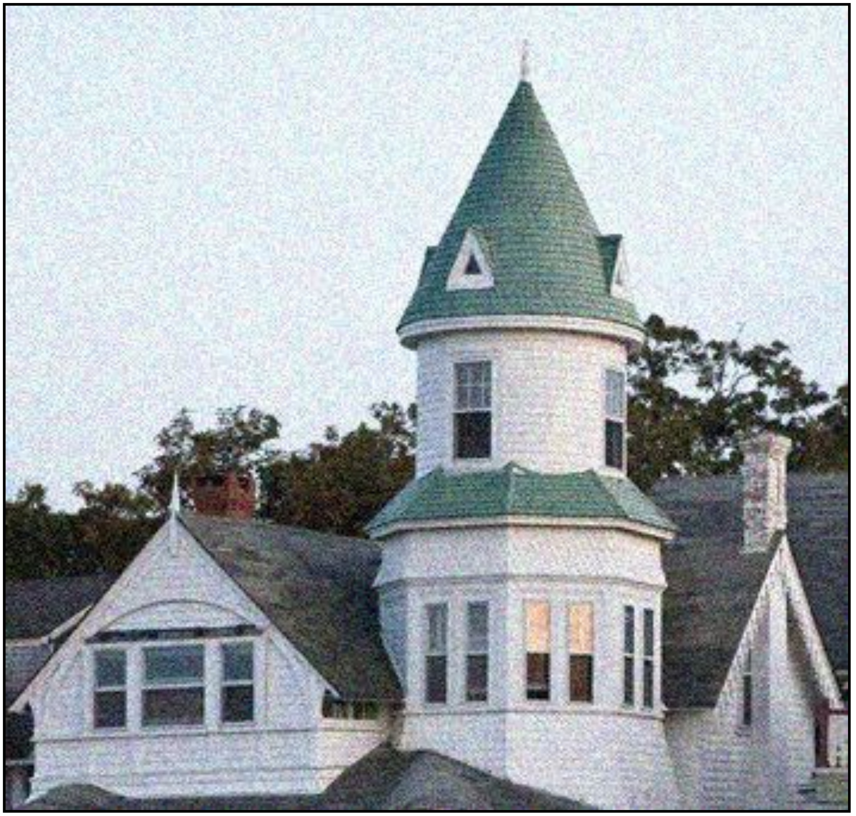
Key idea

Denoise the no-flash image while maintaining the edge structure of the flash image

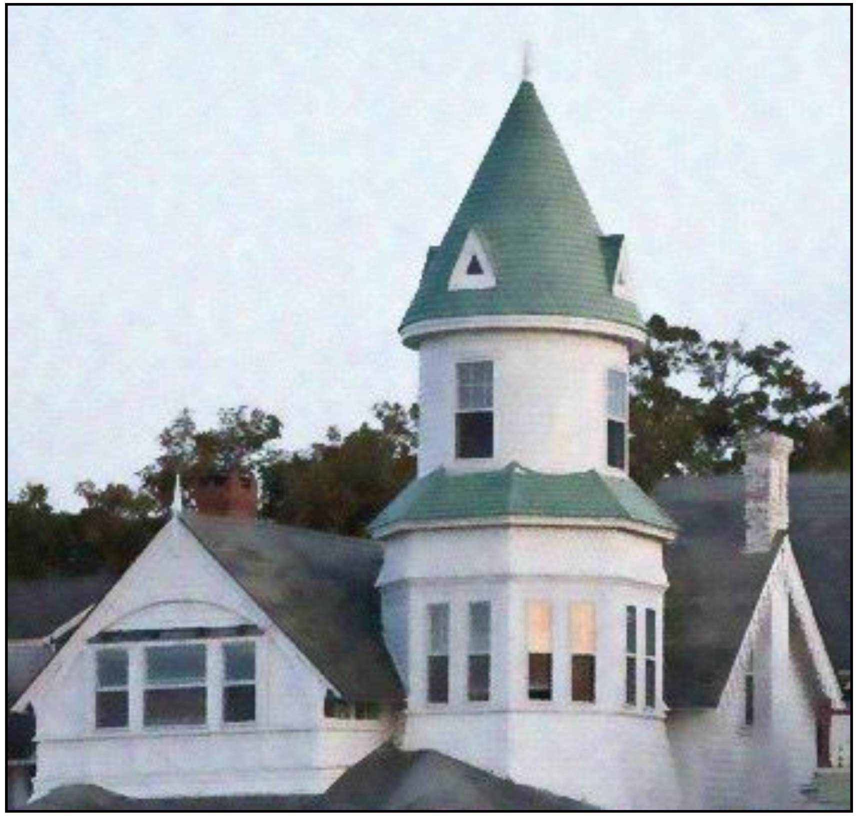
- How would you do this using the image editing techniques we’ve learned about?

Joint bilateral filtering

Denoising with bilateral filtering



noisy input



bilateral filtering

median filtering

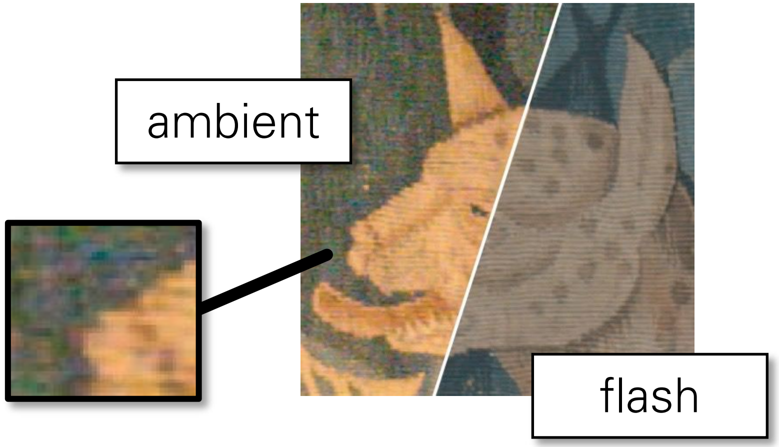
Denoising with bilateral filtering

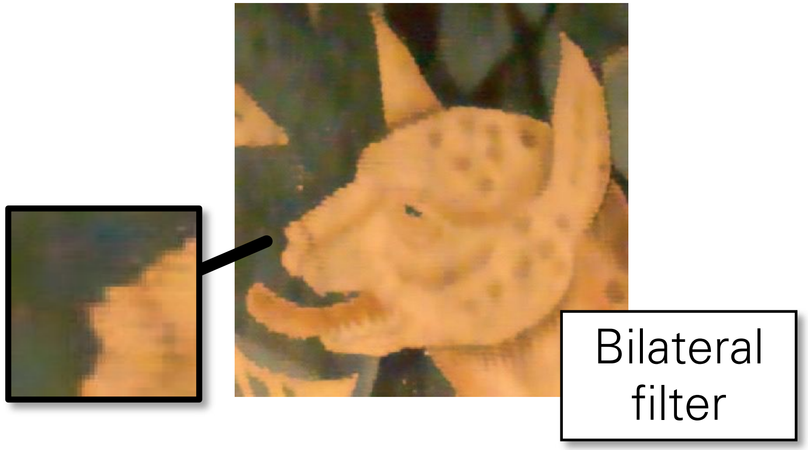
spatial kernel

$$A^{Base}_{p(col)} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} \boxed{g_d(|p - p'|)} \boxed{g_r\left(A_{p(col)} - A_{p'(col)}\right)} A_{p'(col)}$$

intensity kernel

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





Denoising with joint bilateral filtering

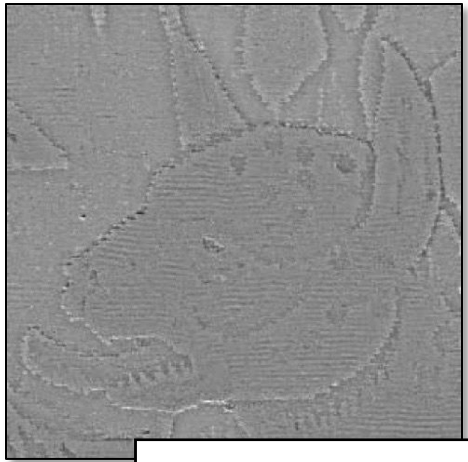
$$A^{NR}_{p(col)} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|) \, g_r(\boldsymbol{F}_{p(col)} - \boldsymbol{F}_{p'(col)}) A_{p'(col)}$$

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

Denoising with joint bilateral filtering

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

Bilateral filter



The difference

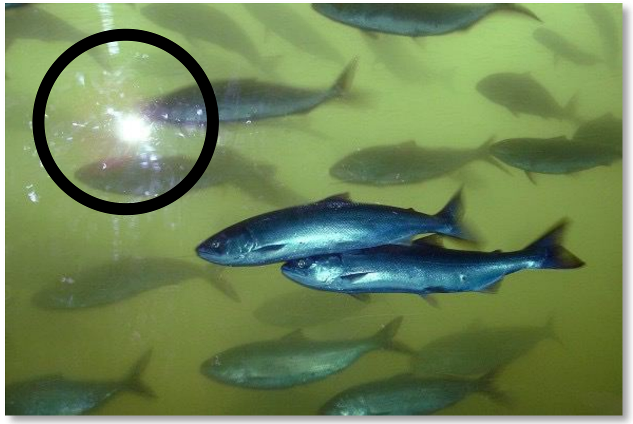
Joint Bilateral filter

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.
- Take pixel p if $F_{p(col)}^{Lin} - A_{p(col)}^{Lin} \leq \tau_{Shadow}$
- 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:

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

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?

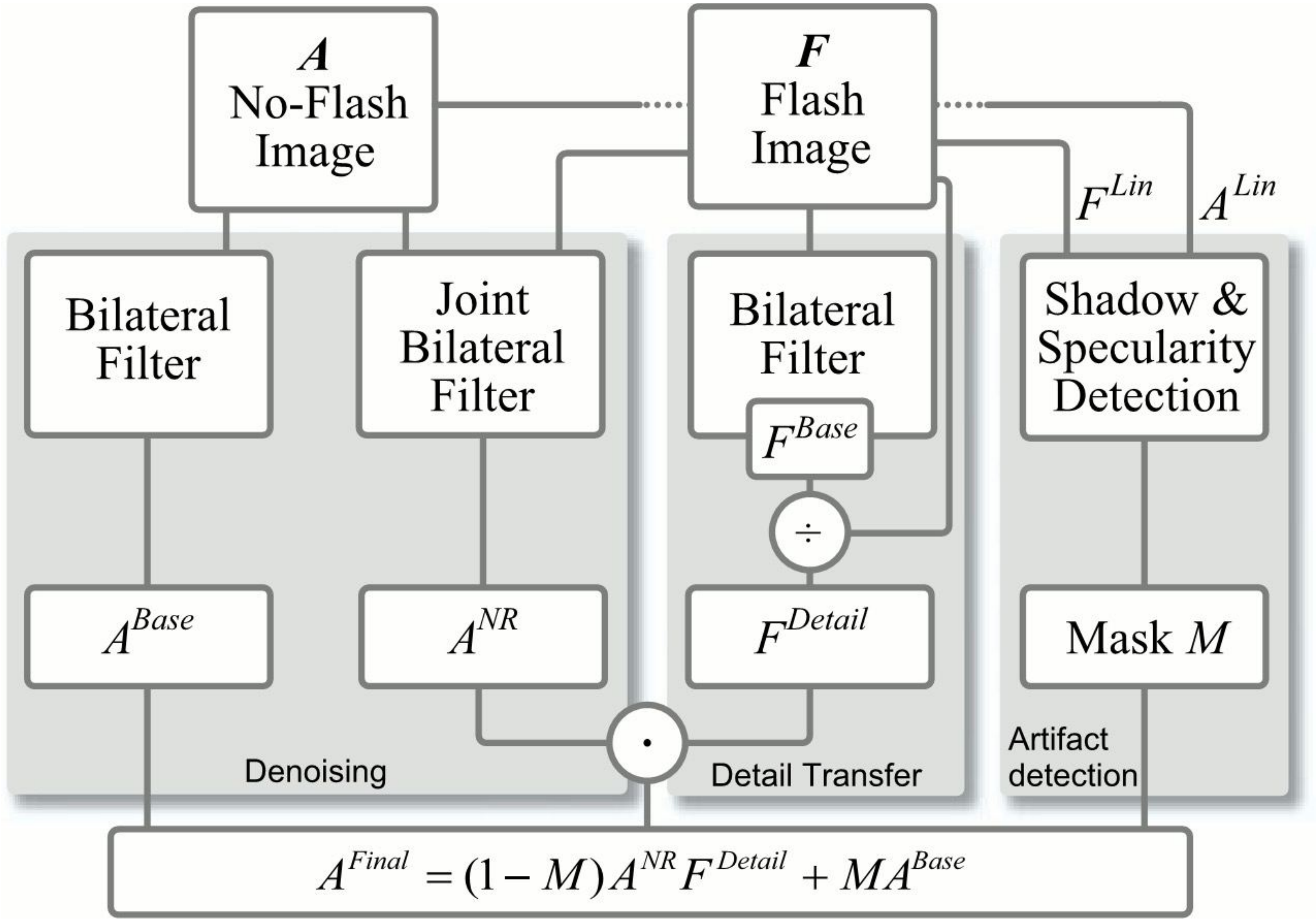
Detail transfer

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

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

Reduces the effect of noise in F

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

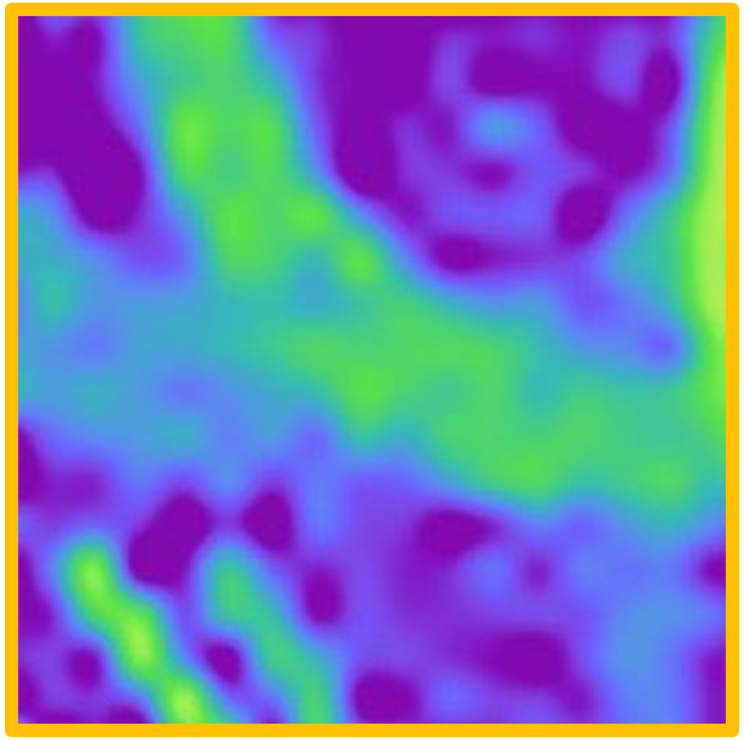

Result

Edge-aware depth denoising

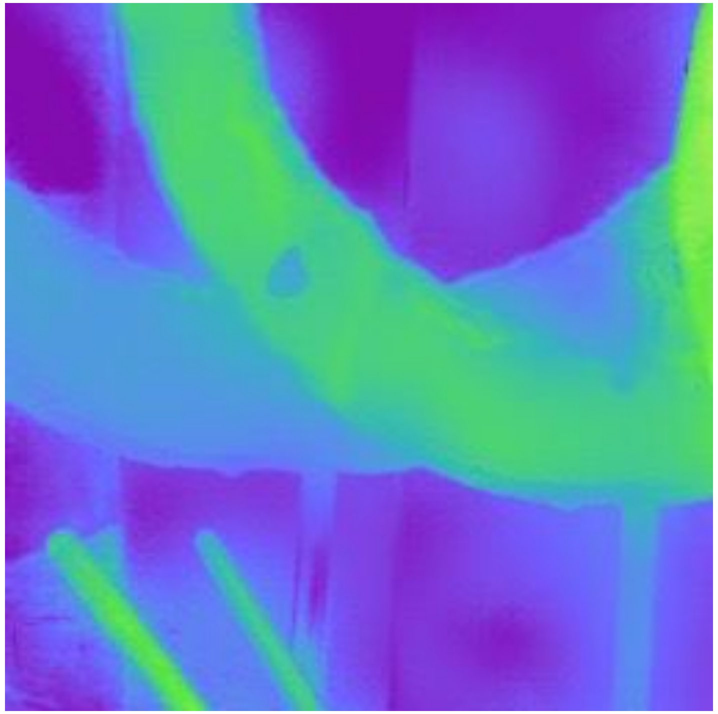
$$A_{p(col)} = \frac{1}{k(p(col))} \sum_{p' \in \Omega} g_d(|p - p'|)\, g_r\left(\boldsymbol{F}_{p(col)} - \boldsymbol{F}_{p'(col)}\right) A_{p'(col)}$$

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

One of two input images



Depth from disparity

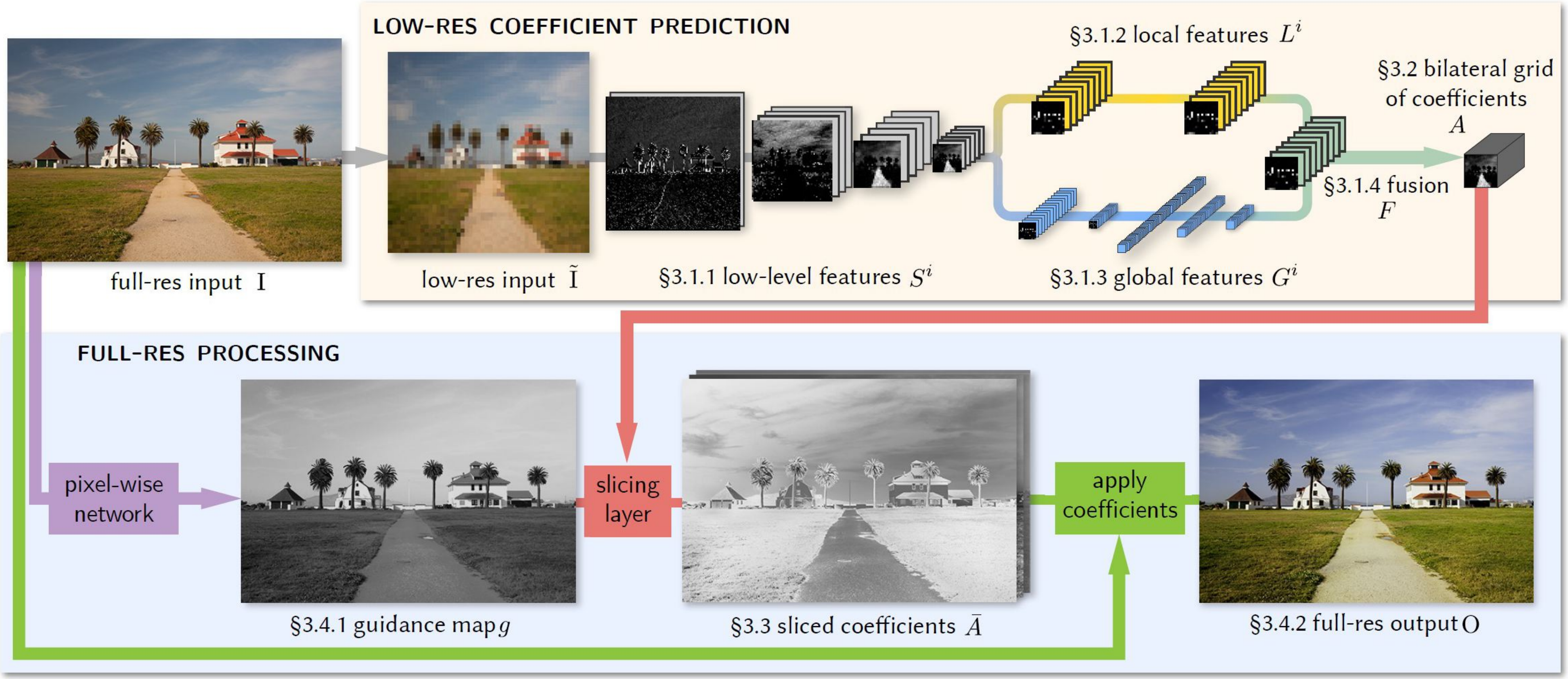


Guided filtering

Other applications of joint bilateral filtering

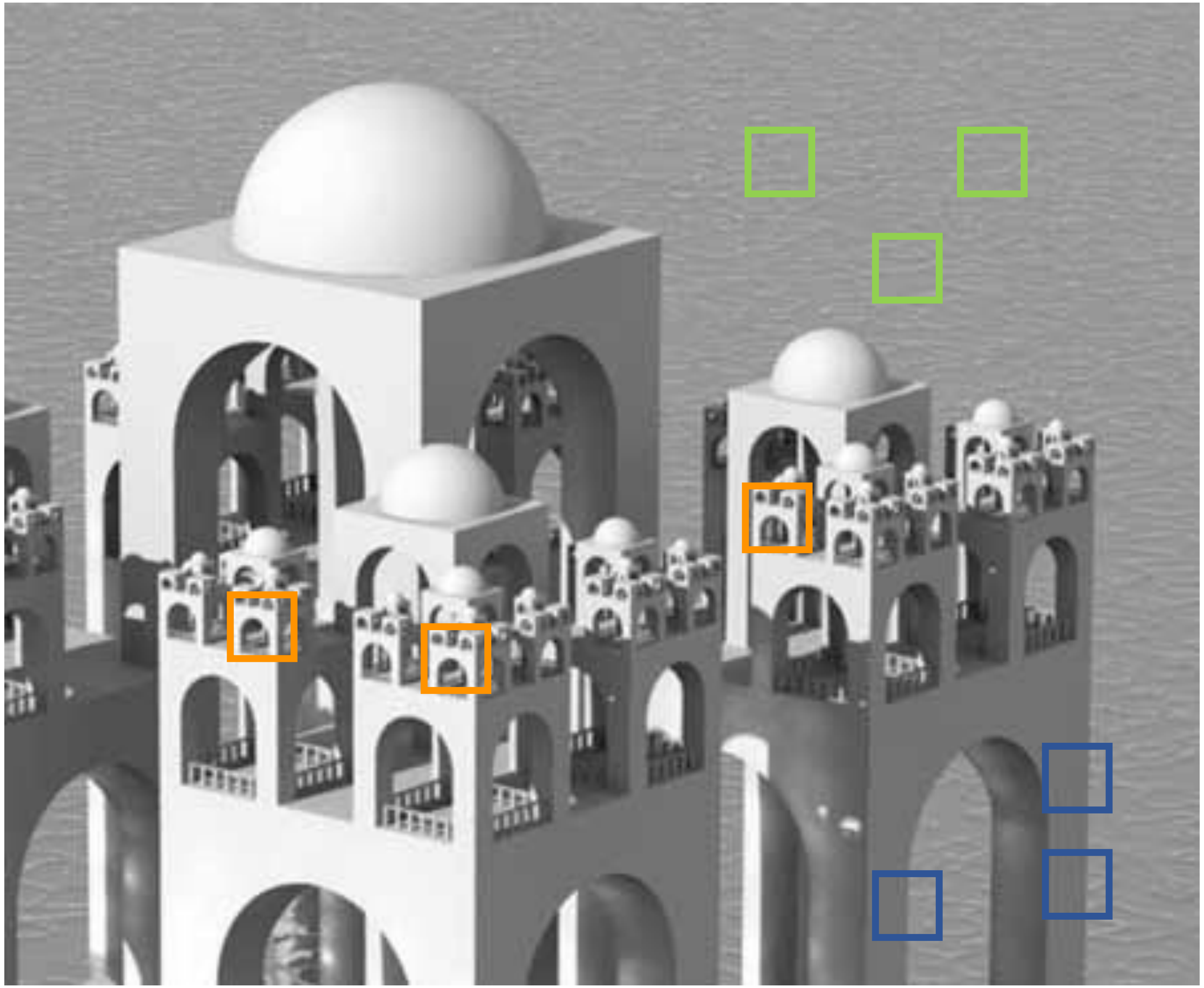
Deep Bilateral Learning for Real-Time Image Enhancement

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



Non-Local Means Filter

Redundancy in natural images

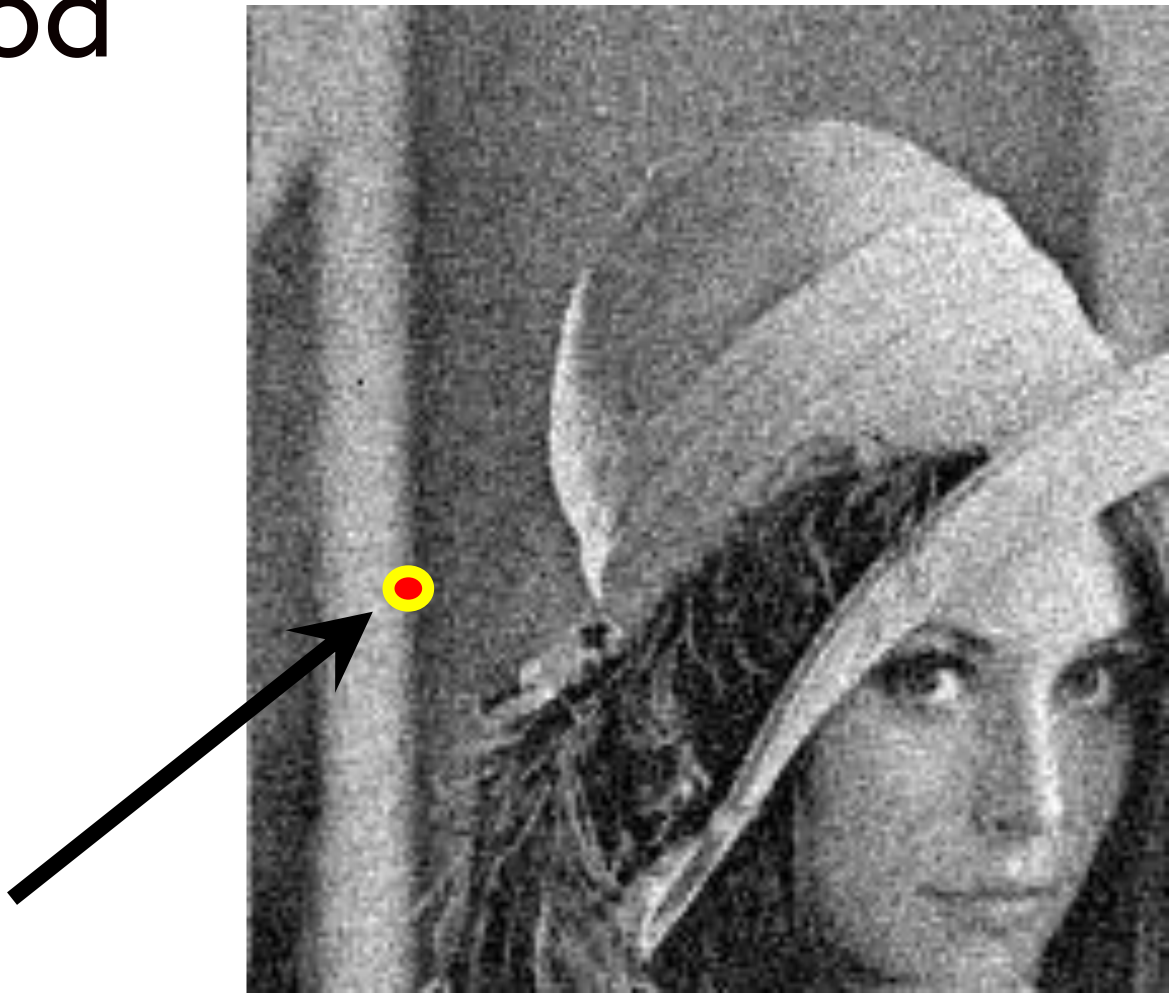


NL-Means Filter (Buades 2005)

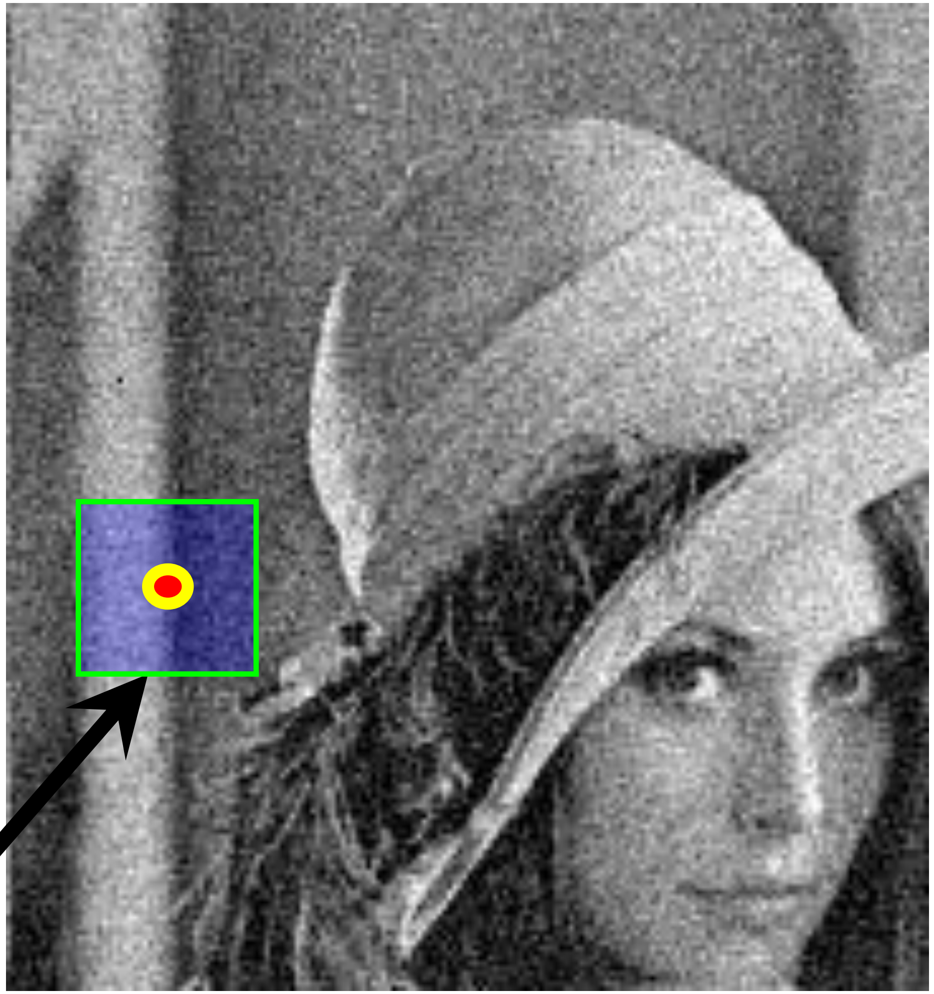
- Same goals: ‘Smooth within Similar Regions’
- <u>KEY INSIGHT</u>: Generalize, extend ‘Similarity’
  - Bilateral:
    Averages neighbors with <u>similar intensities</u>;
  - NL-Means:
    Averages neighbors with <u>similar neighborhoods</u>!

NL-Means Method

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



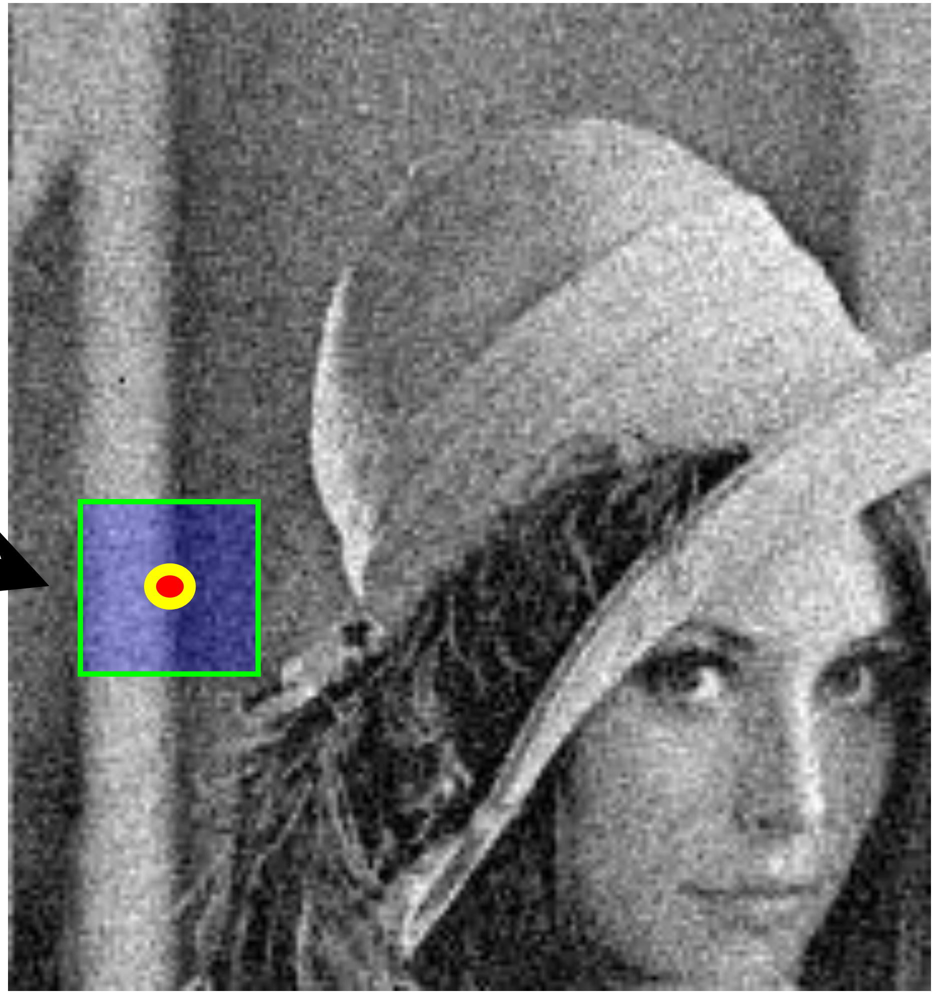
NL-Means Method



- For each and every pixel **p**:
  - Define a small, simple fixed size neighborhood;

NL-Means Method

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



- 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;

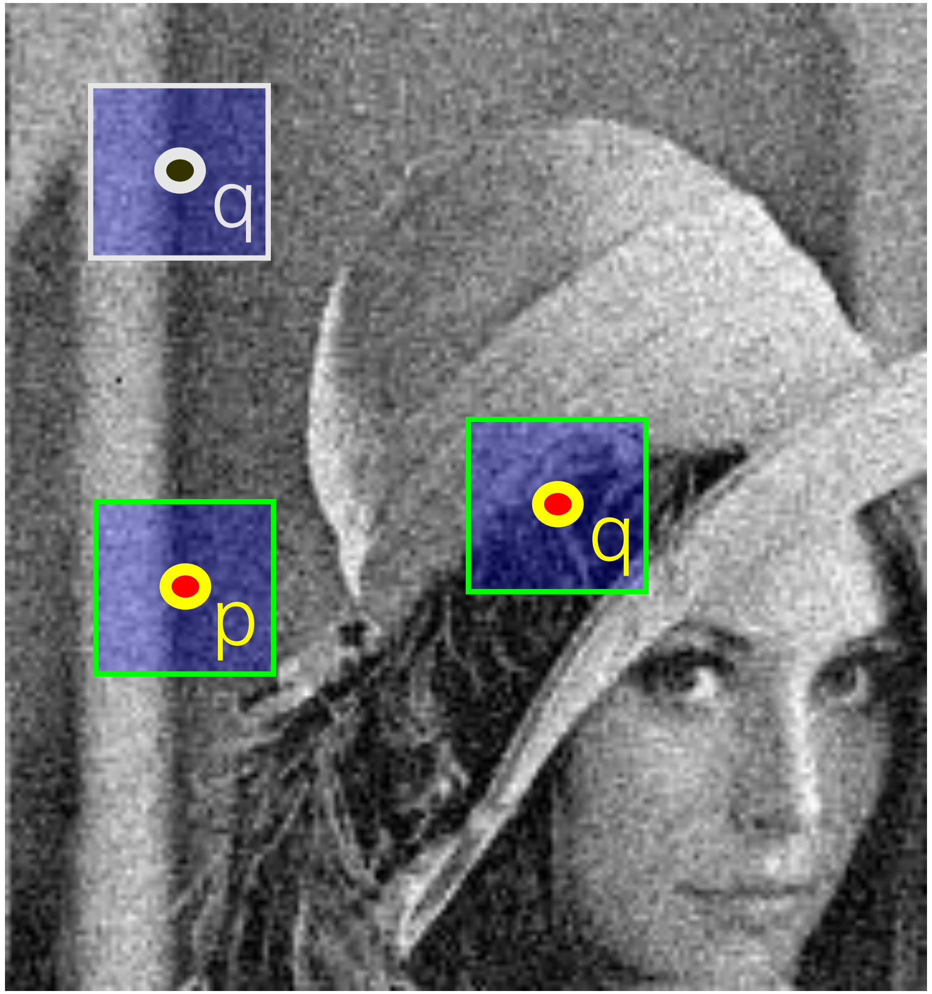
$$|| V_p - V_q ||^2$$

NL-Means Method

'Dissimilar' pixels **p, q**

→ LARGE
vector distance;

$$|| V_p – V_q ||^2$$



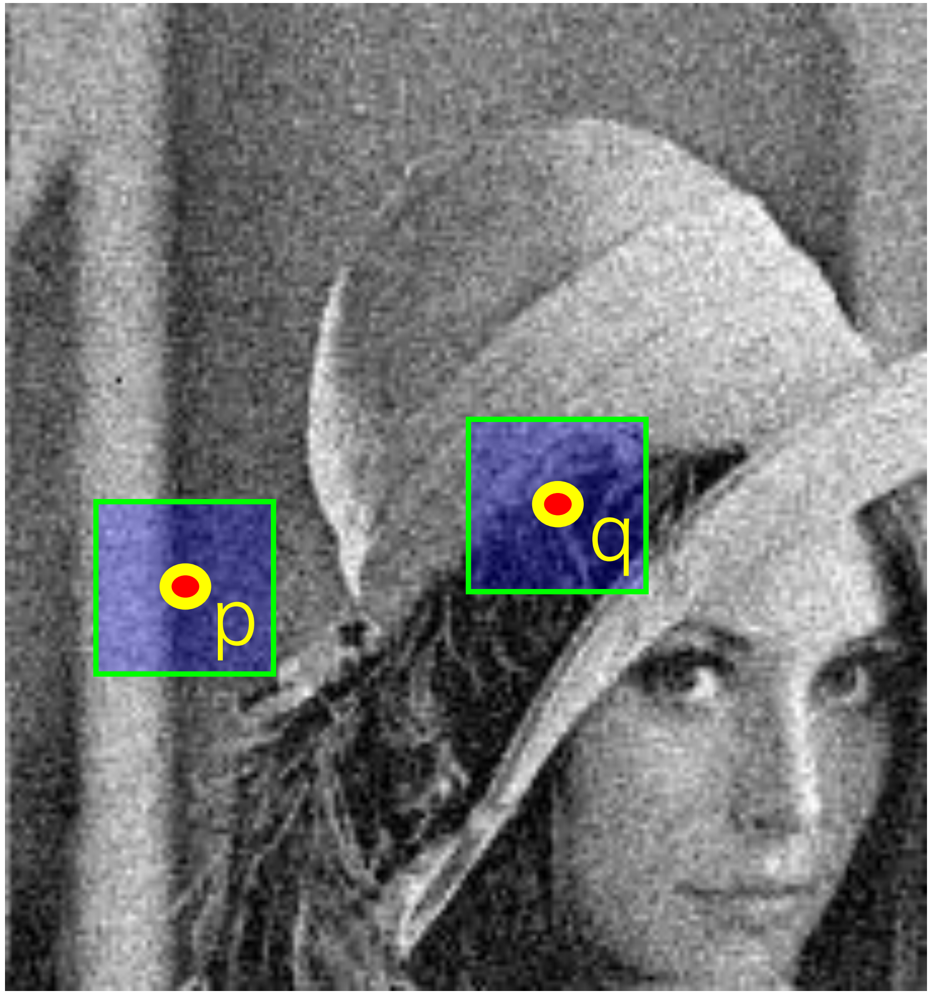
NL-Means Method

'Dissimilar' pixels p, q

→ LARGE
vector distance;

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

Filter with this!



NL-Means Method

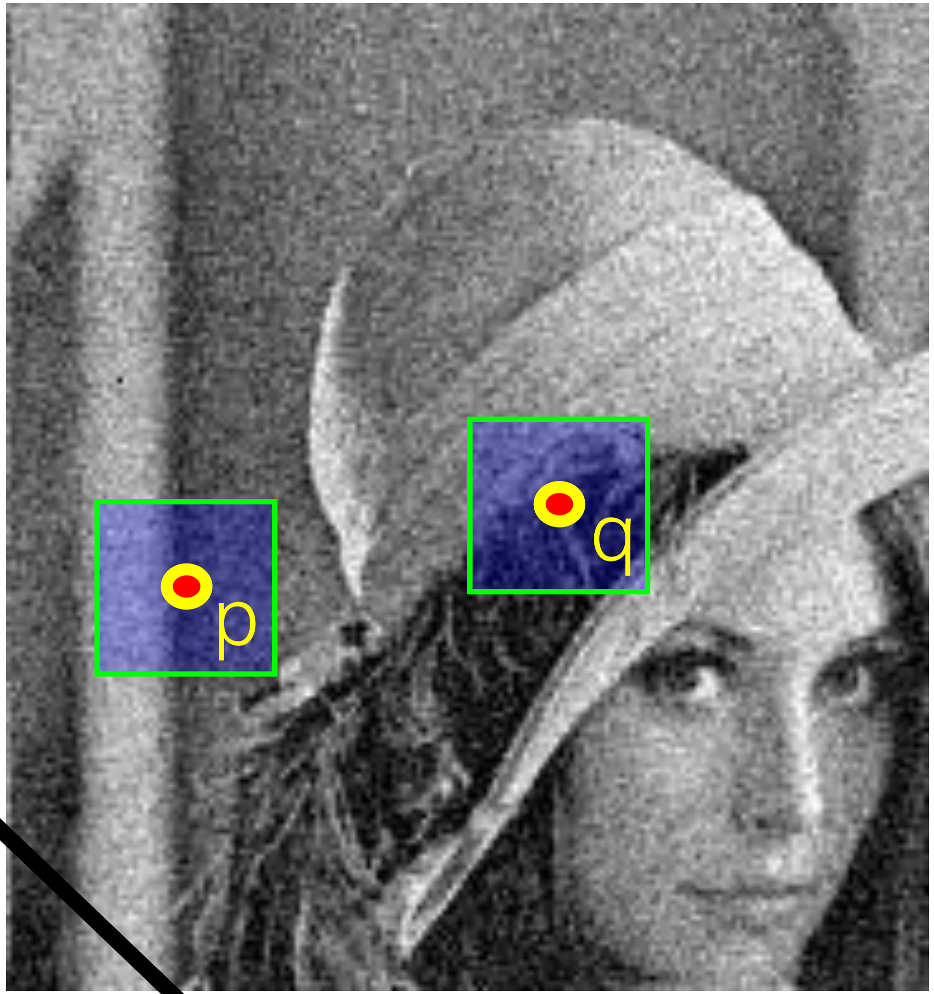
p, q neighbors define

a vector distance;

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

Filter with this:

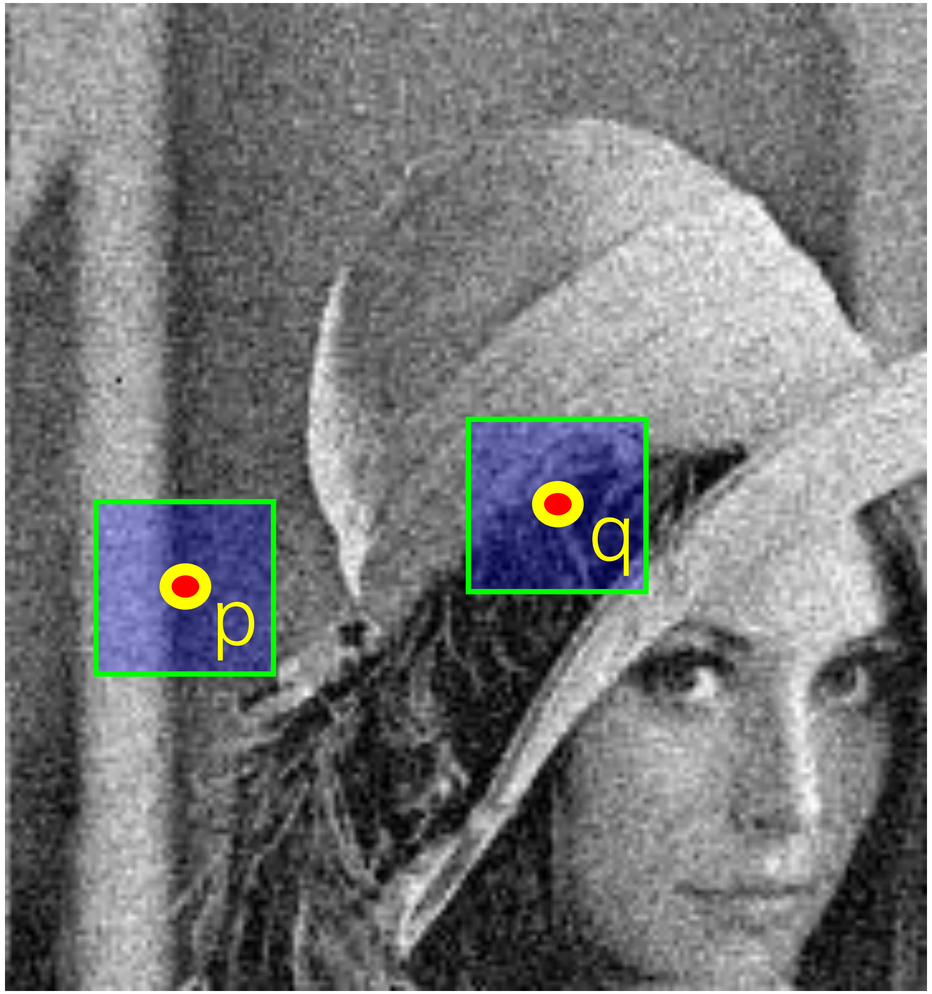
No spatial term!



$$NLMF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} \cancel{G_{\sigma_s}(\| \mathbf{p} - \mathbf{q} \|)} \, G_{\sigma_r}\left(\boxed{\| \vec{V}_{\mathbf{p}} - \vec{V}_{\mathbf{q}} \|^2} \right) I_{\mathbf{q}}$$

NL-Means Method

pixels p, q neighbors
Set a vector distance;

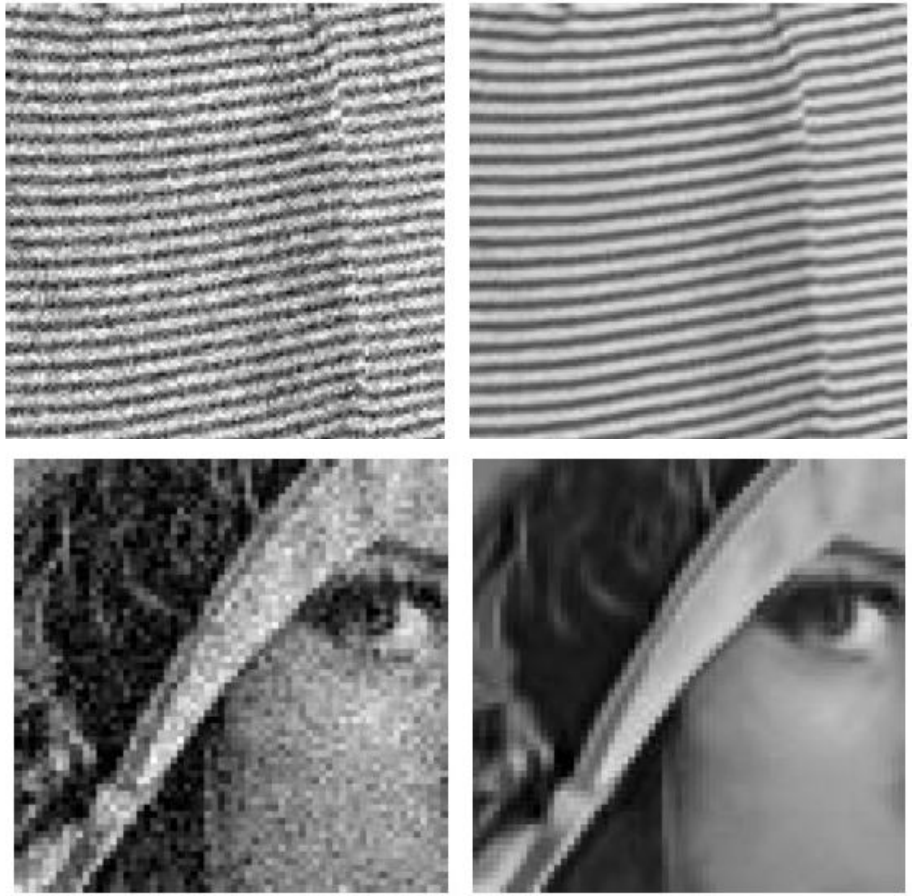


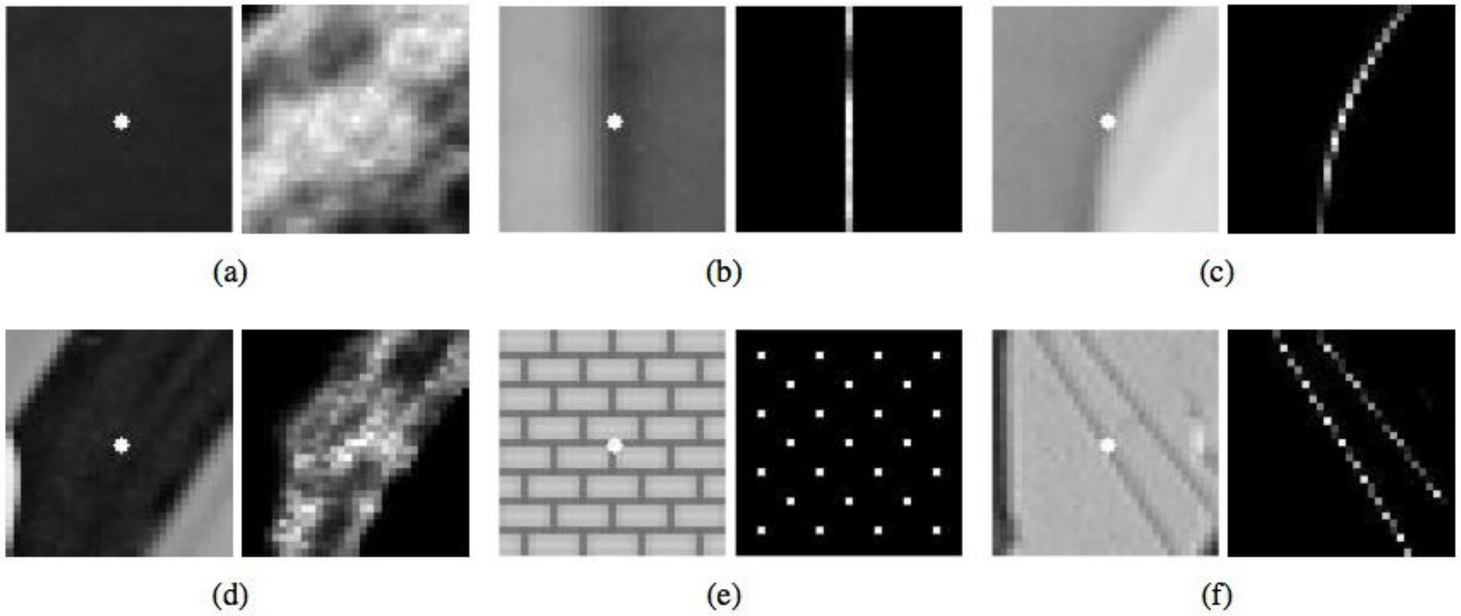
$$\boxed{\| V_p - V_q \|^2}$$

Vector Distance to p sets
weight for each pixel q

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

NL-Means Method





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

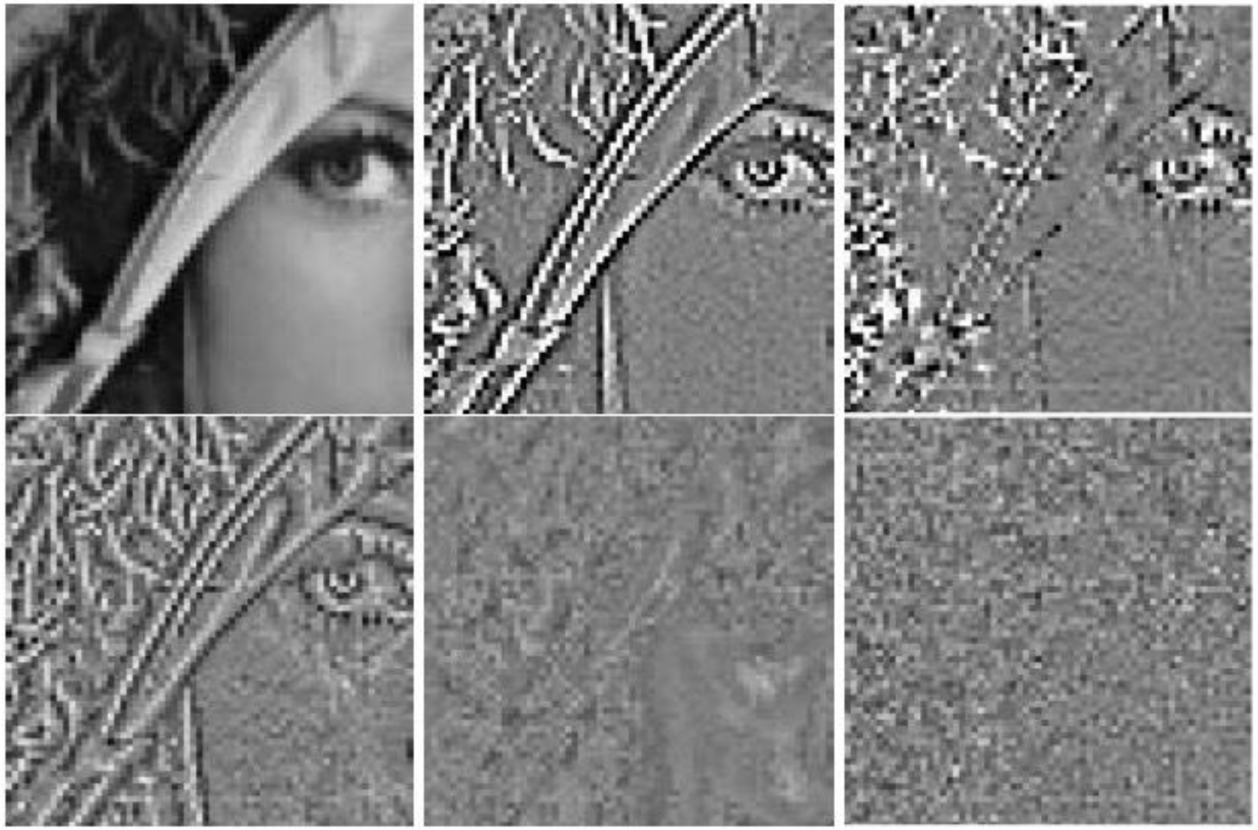
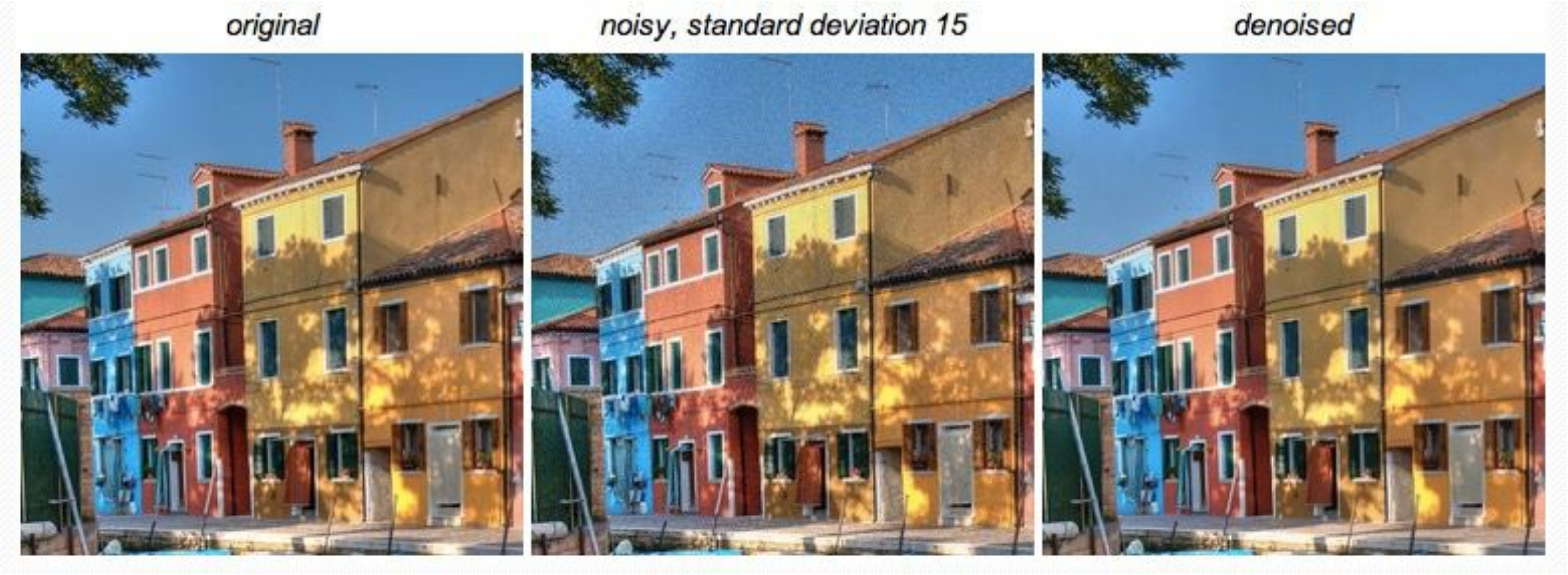


Figure 4. Method noise experience on a natural image. Displaying of the image difference $u - D_h(u)$. From left to right and from top to bottom: original image, Gauss filtering, anisotropic filtering, Total variation minimization, Neighborhood filtering and NL-means algorithm. The visual experiments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

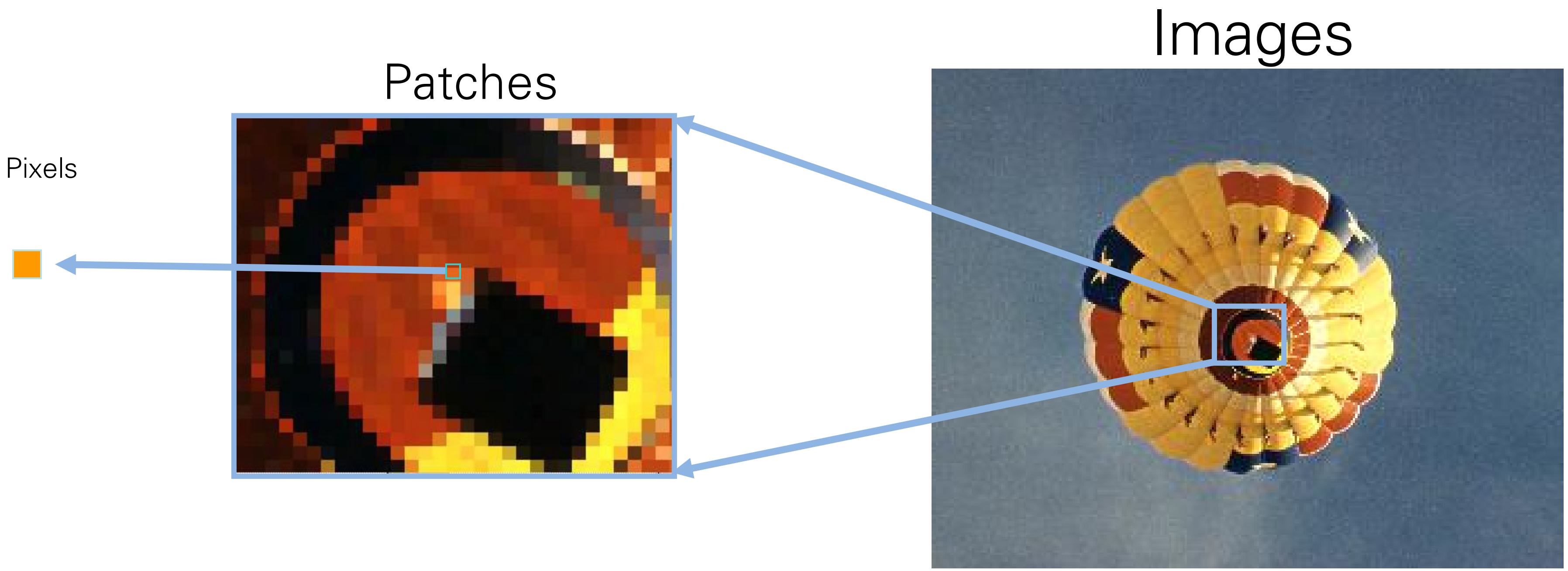
NL-Means Method



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

RegCov Smoothing

From pixels to patches and to images

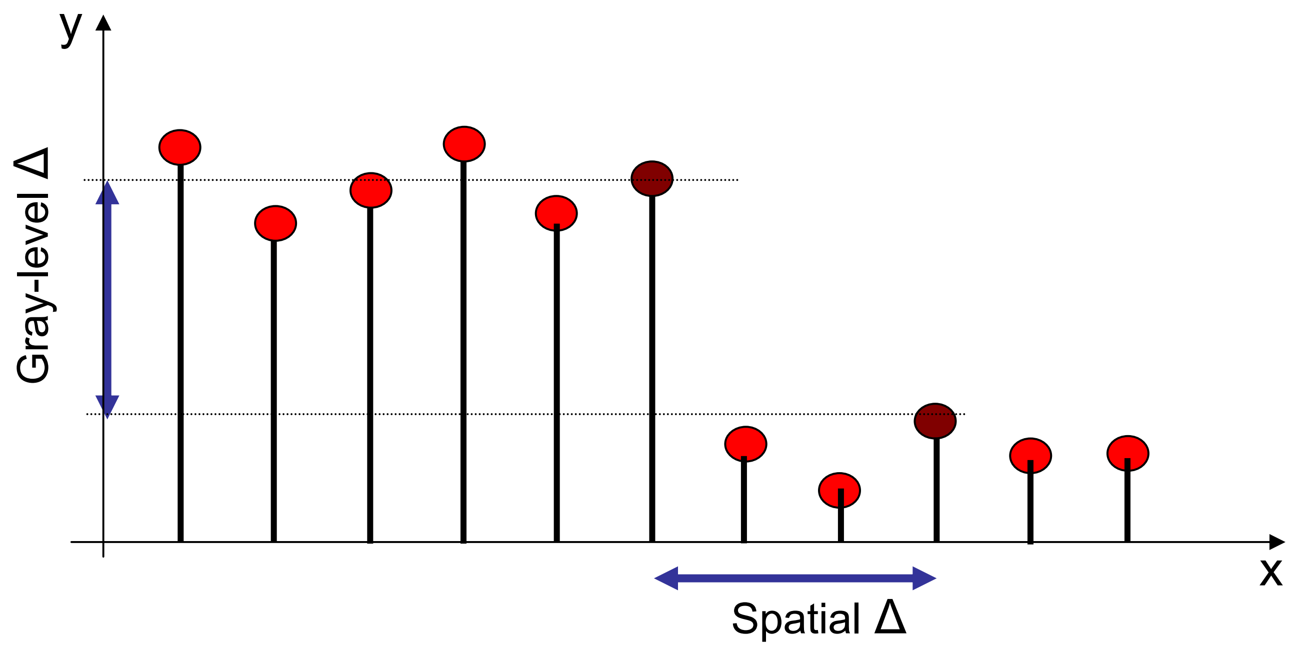


Similarities can be defined at different scales..

Slide credit: P. Milanfar

Pixelwise similarity metrics

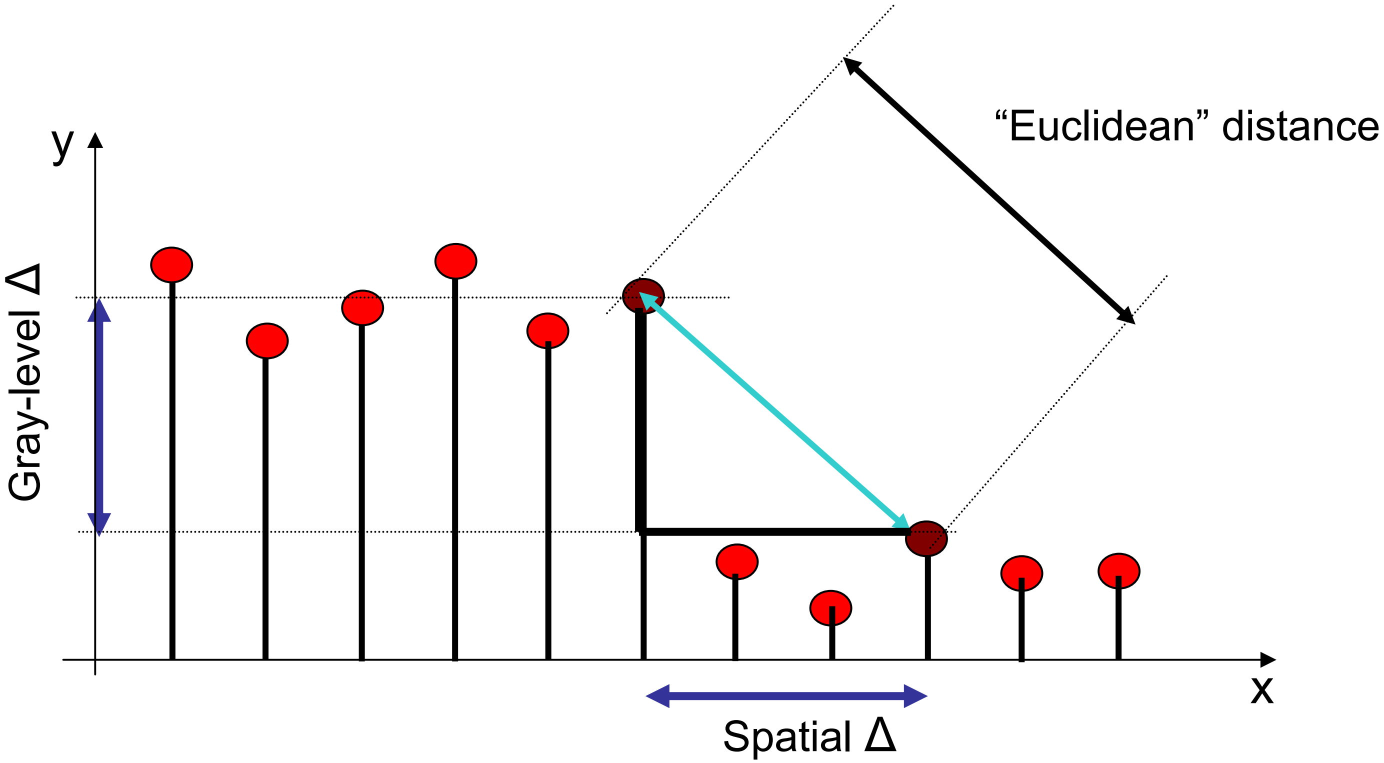
- To measure the similarity of two pixels, we can consider
  - Spatial distance
  - Gray-level distance



Slide credit: P. Milanfar

Euclidean metrics

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



Slide credit: P. Milanfar

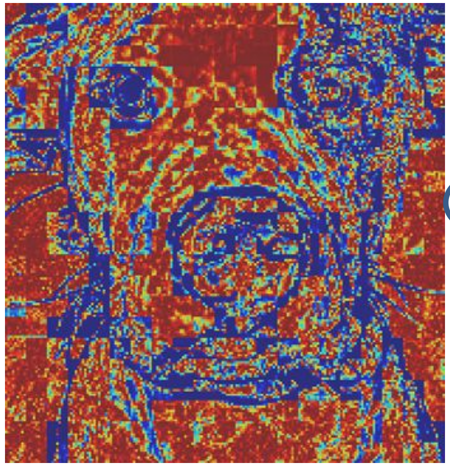
Bilateral Kernel (BL) [Tomasi et al. ‘98]

Pixels

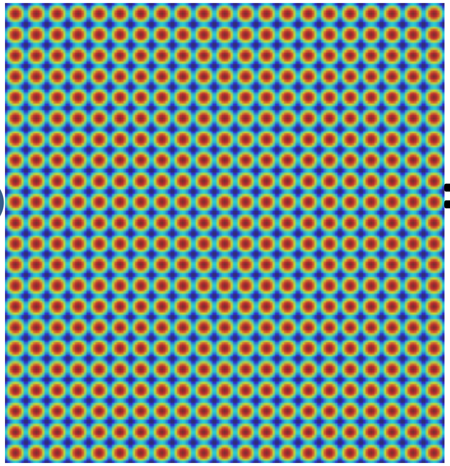
$$K(\mathbf{x}_l, \mathbf{x}, y_l, y) = \exp\left\{-\frac{\|y_l - y\|^2}{h_r^2} - \frac{\|\mathbf{x}_l - \mathbf{x}\|^2}{h_d^2}\right\}$$

Pixel similarity

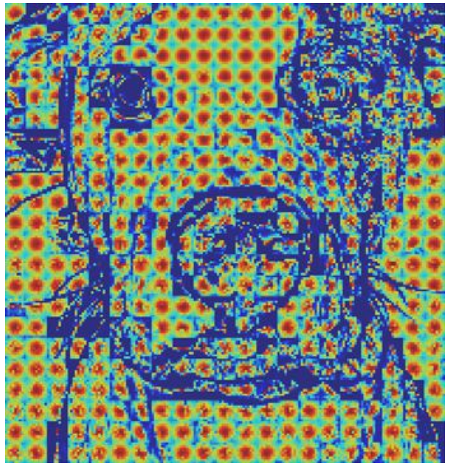
Spatial similarity



$\odot$



$=$



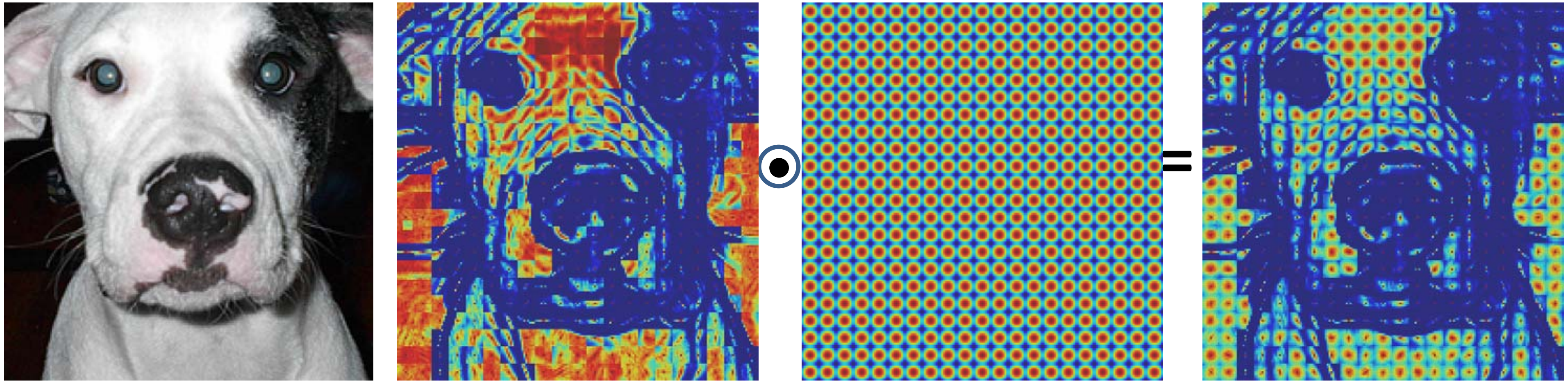
Slide credit: P. Milanfar

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

Slide credit: P. Milanfar

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

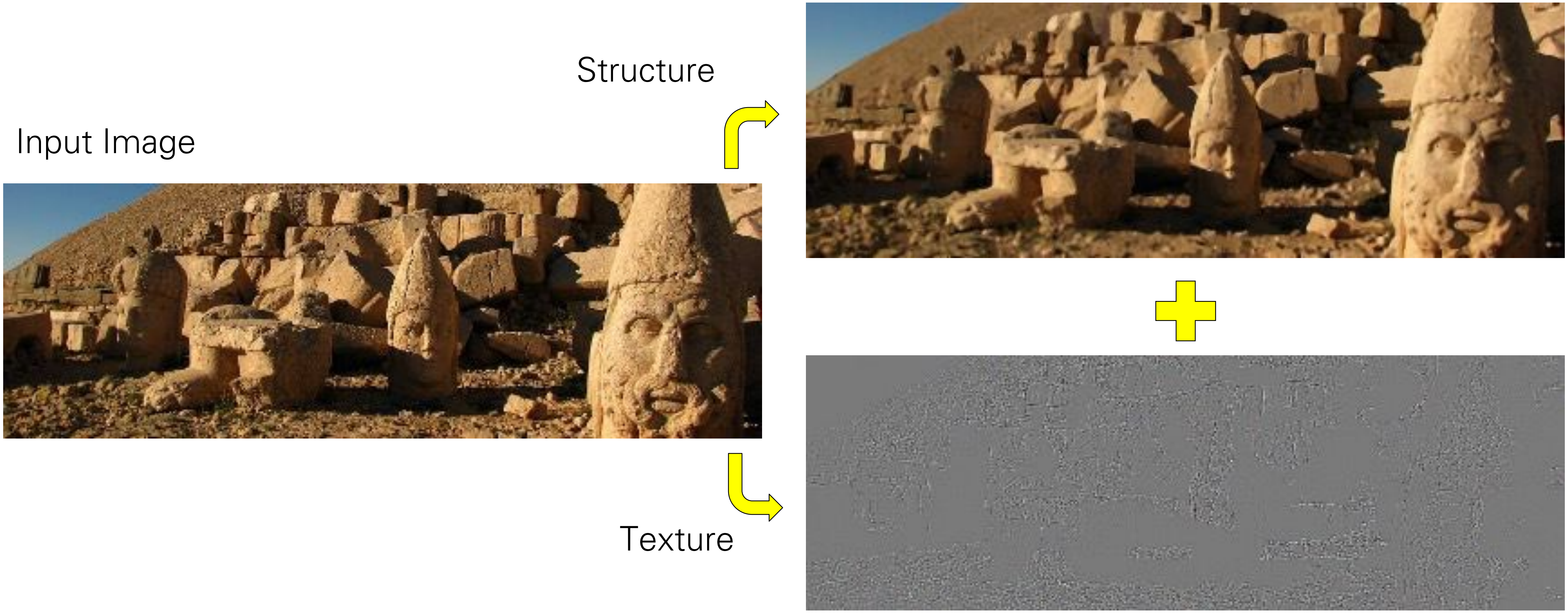
Structure-Texture Decomposition

- Decomposing an image into structure and texture components

Texture Component

Structure-Texture Decomposition

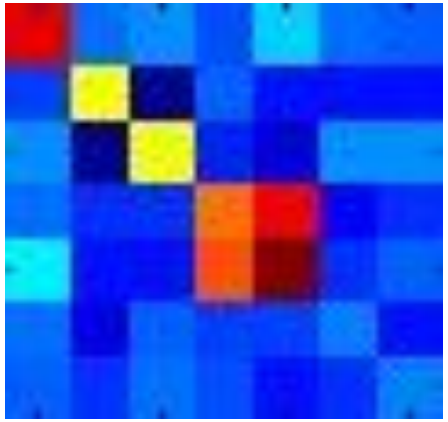
- Decomposing an image into structure and texture components



Structure-Texture Decomposition

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

$$F(x,y) = \left[I(x,y) \quad \left|\frac{\partial I}{\partial x}\right| \quad \left|\frac{\partial I}{\partial y}\right| \quad \left|\frac{\partial^2 I}{\partial x^2}\right| \quad \left|\frac{\partial^2 I}{\partial y^2}\right| \quad x \quad y \right]^T$$

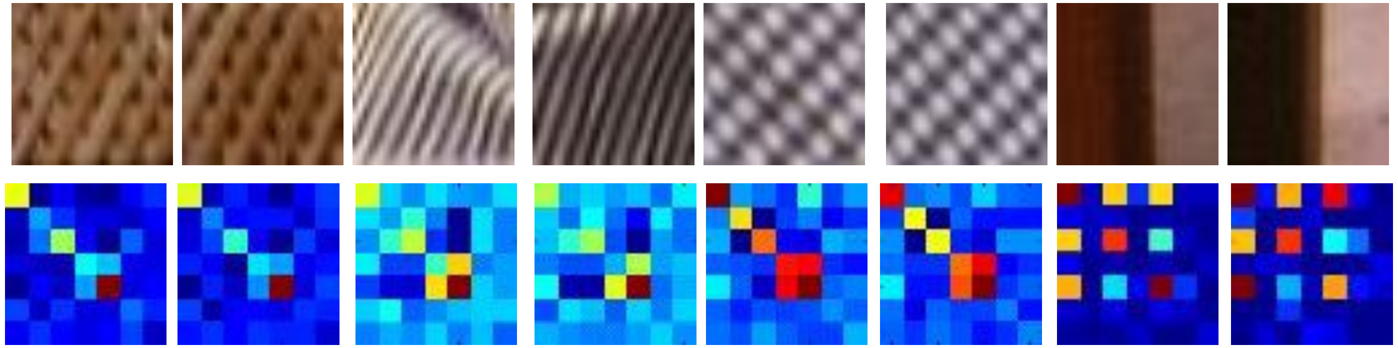


$$\mathbf{C}_R = \frac{1}{n-1} \sum_{i=0}^{n} (\mathbf{z}_k - \mu)(\mathbf{z}_k - \mu)^T$$

Tuzel et al., ECCV 2006

Structure-Texture Decomposition

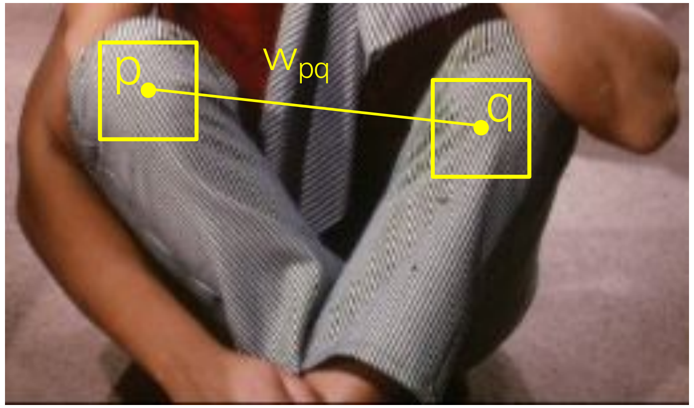
- Region covariances capture local structure and texture information.
- Similar regions have similar statistics.



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)

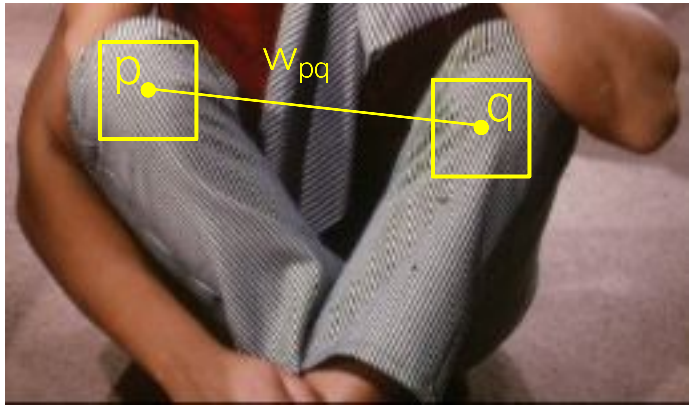
RegCov Smoothing – Model 1

- 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

$$\mathbf{s}_i = \begin{cases} \alpha\sqrt{d}\mathbf{L}_i & \text{if } 1 \le i \le d \\ -\alpha\sqrt{d}\mathbf{L}_i & \text{if } d+1 \le i \le 2d \end{cases}$$



Final representation

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

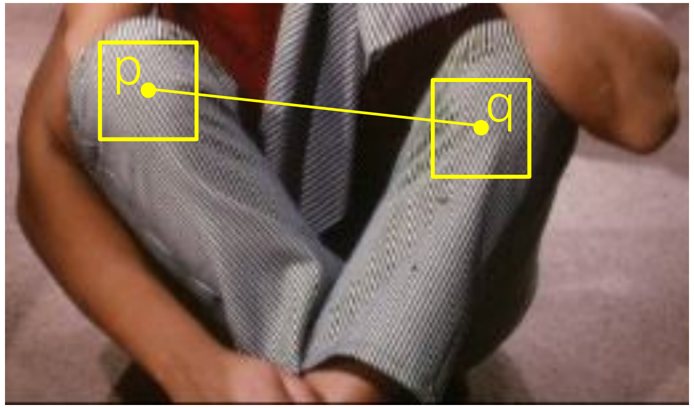
Resulting kernel function

$$w_{\mathbf{pq}} \propto \exp\left(-\frac{\|\Psi(\mathbf{C}_\mathbf{p}) - \Psi(\mathbf{C}_\mathbf{q})\|^2}{2\sigma^2}\right)$$

RegCov Smoothing – Model 2

- An alternative way is to use statistical similarity measures.
- A Mahalanobis-like distance measure to compare to image patches.

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

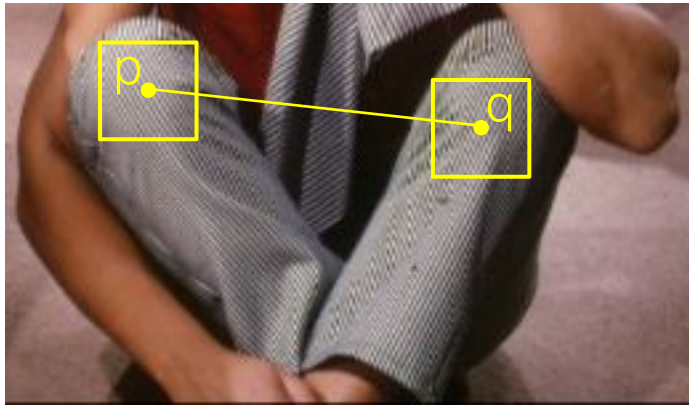


$$\mathbf{C} = \mathbf{C}_{\mathbf{p}} + \mathbf{C}_{\mathbf{q}}$$

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

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



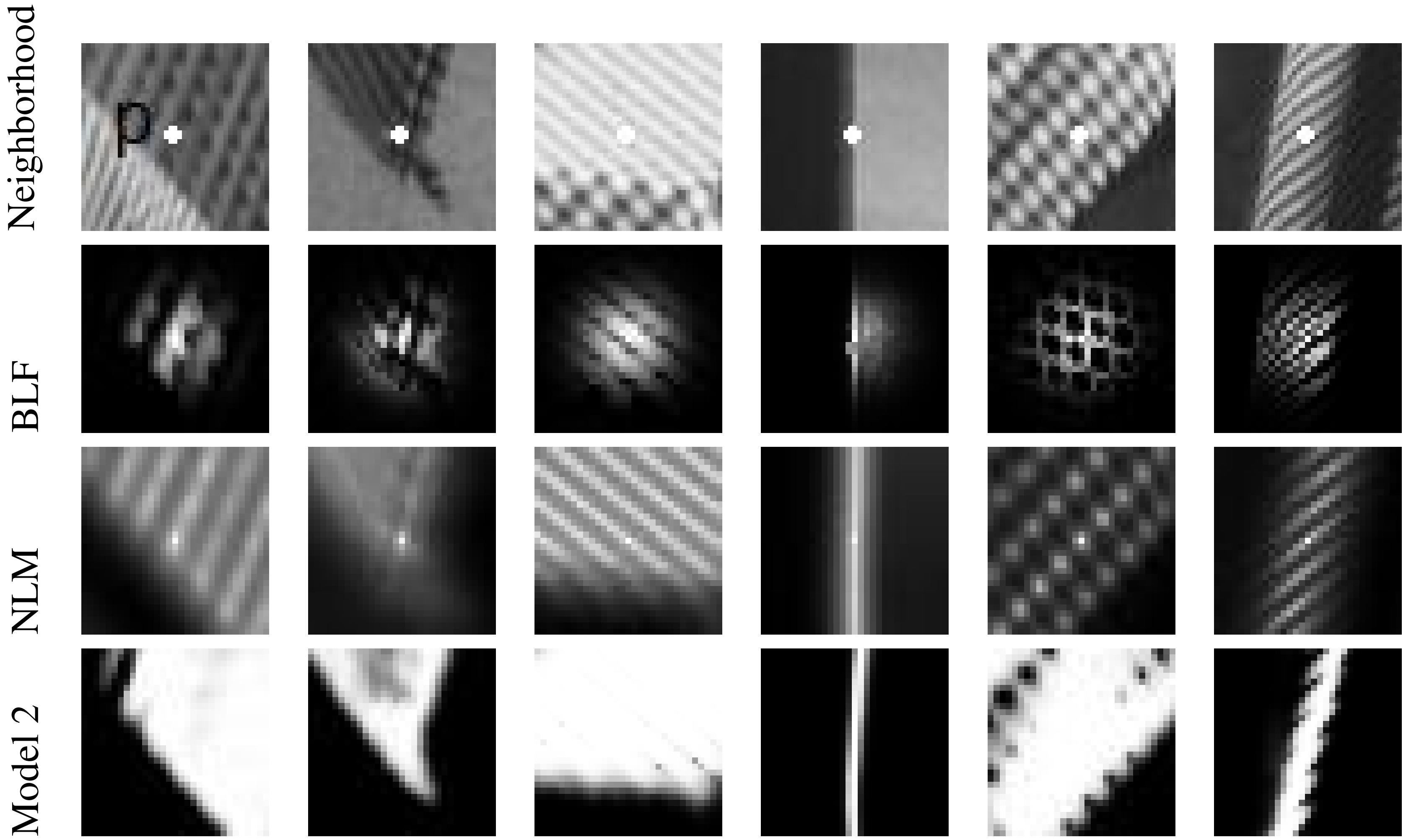
$$d_{KL}(\mathbf{p},\mathbf{q}) = \frac{1}{2}\left(tr(\mathbf{C_q}^{-1}\mathbf{C_p}) + (\mu_p - \mu_q)^T \mathbf{C_q}^{-1}(\mu_p - \mu_q) - k - ln\left(\frac{\det \mathbf{C_p}}{\det \mathbf{C_q}}\right)\right)$$

Resulting kernel

$$w_{pq} \propto \frac{d_{KL}(\mathbf{p},\mathbf{q})}{2\sigma^2}$$

resulted from a discussion with Rahul Narain (Berkeley University)

RegCov Smoothing – Smoothing Kernels



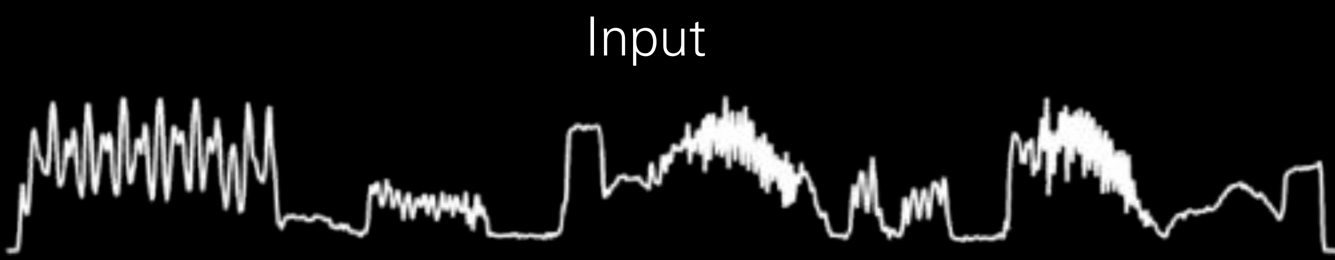
Results

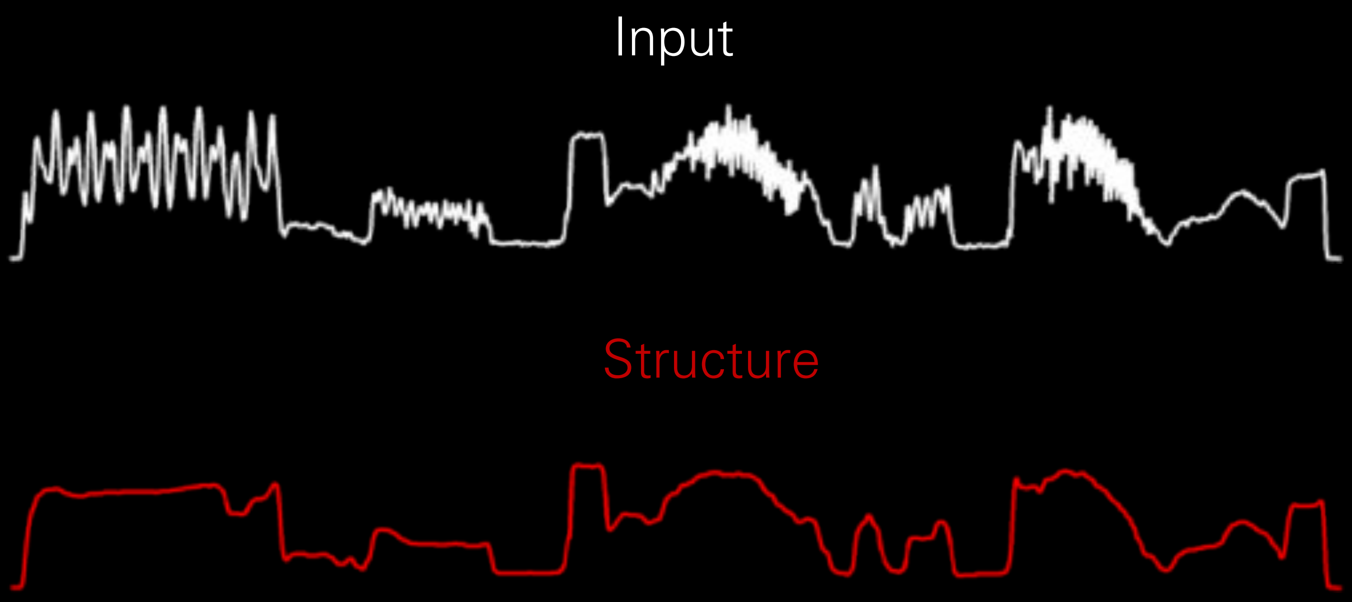
Input

Results

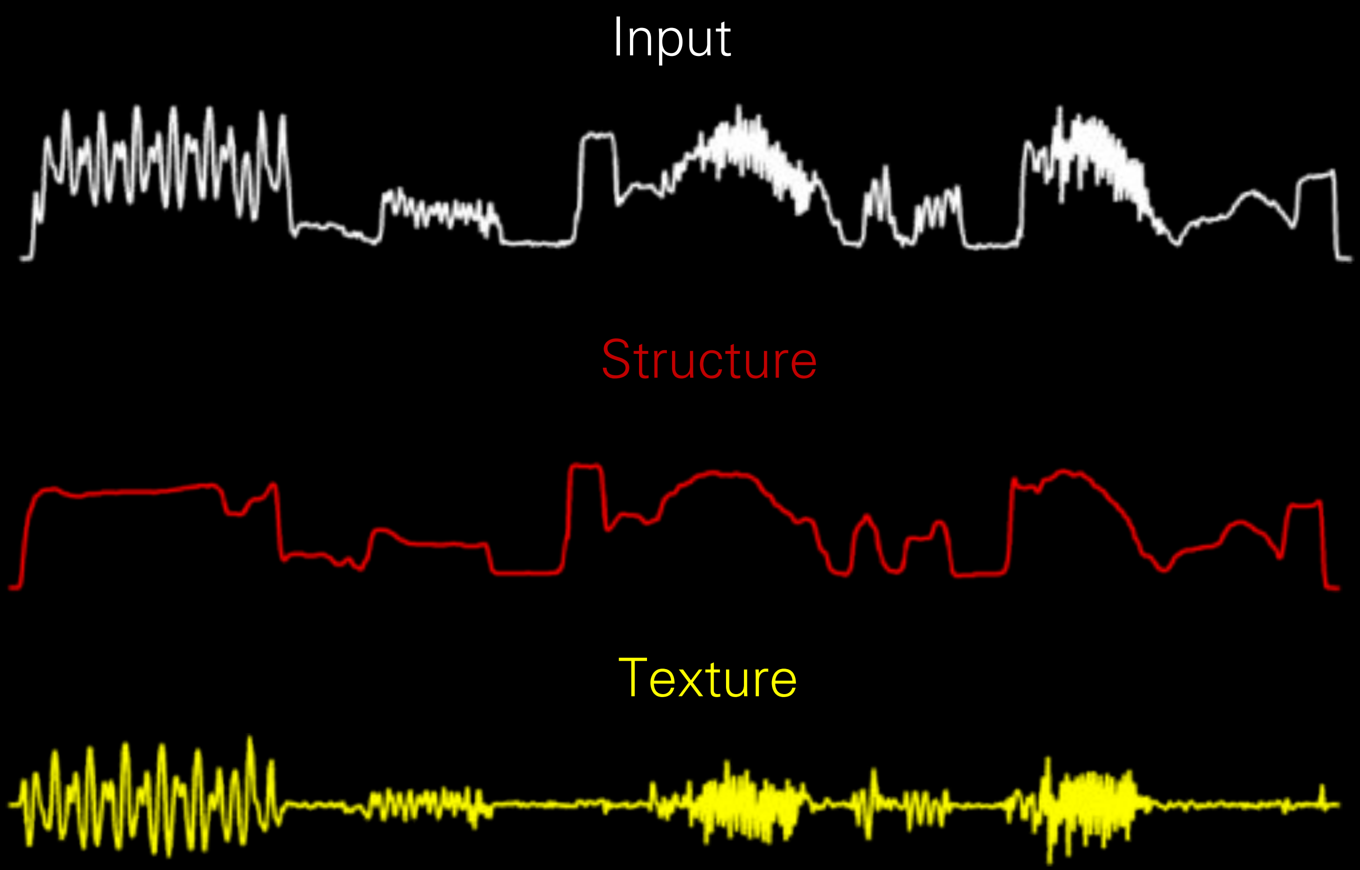
Model1 Model2 Model3

Input



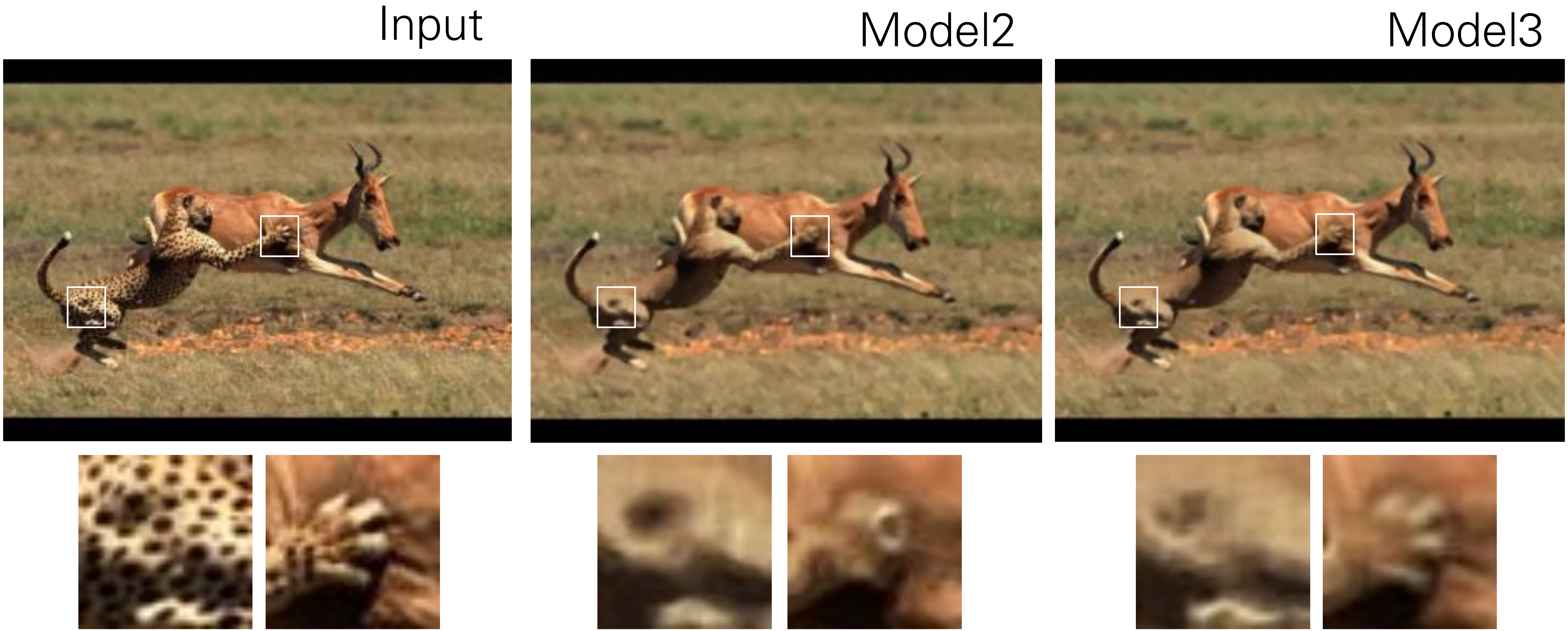


Model2 Structure



Model2 Texture

Results



Experimental evaluation

Input

Experimental evaluation

TV

Rudin et al. 1992

Experimental evaluation

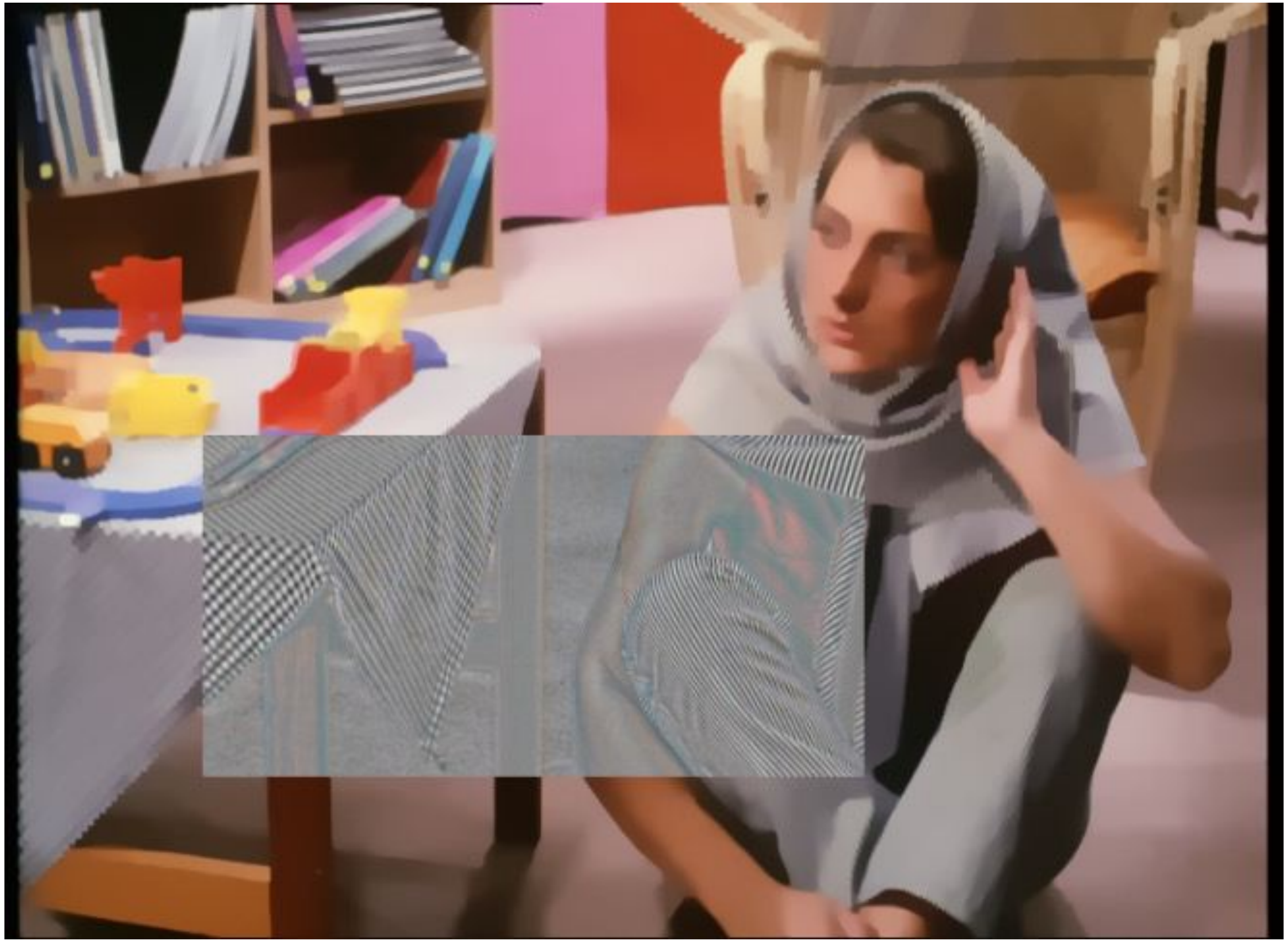
Bilateral Filter

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

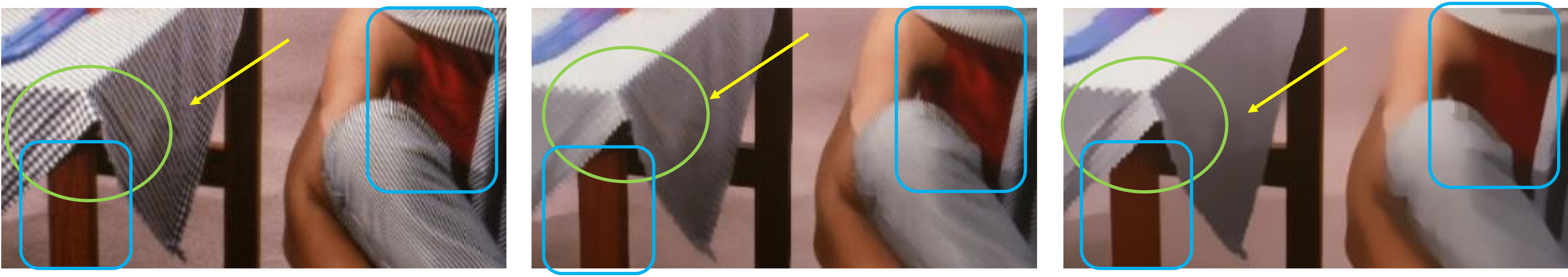
Model 3

Experimental evaluation

Input

Local Exrema

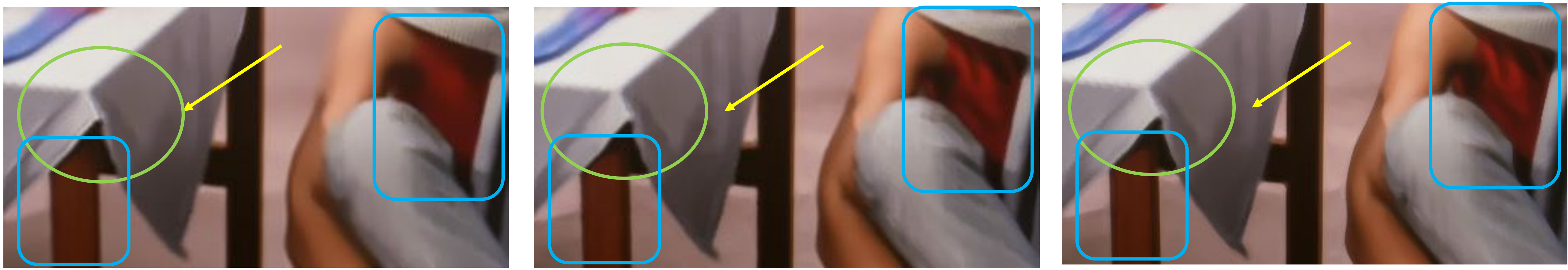
RTV



Model1

Model2

Model3

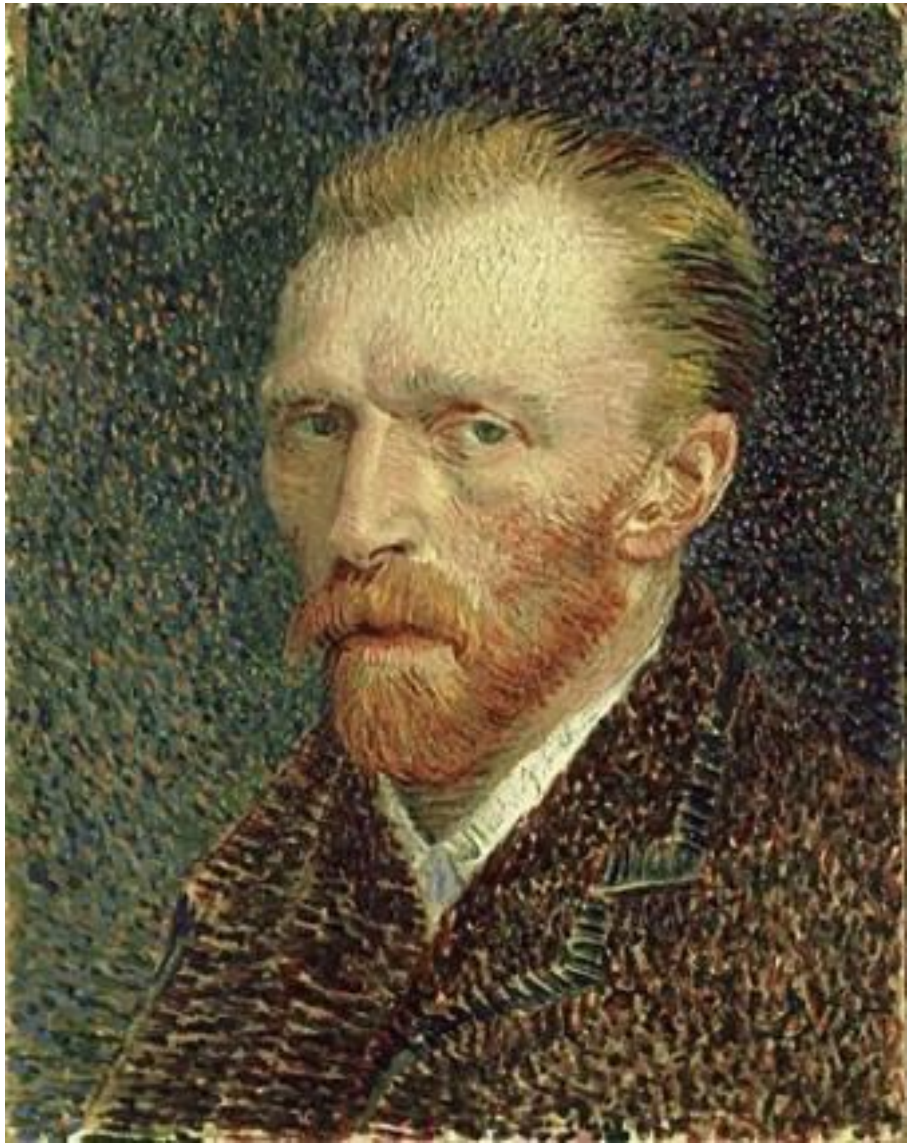


Shading preserved

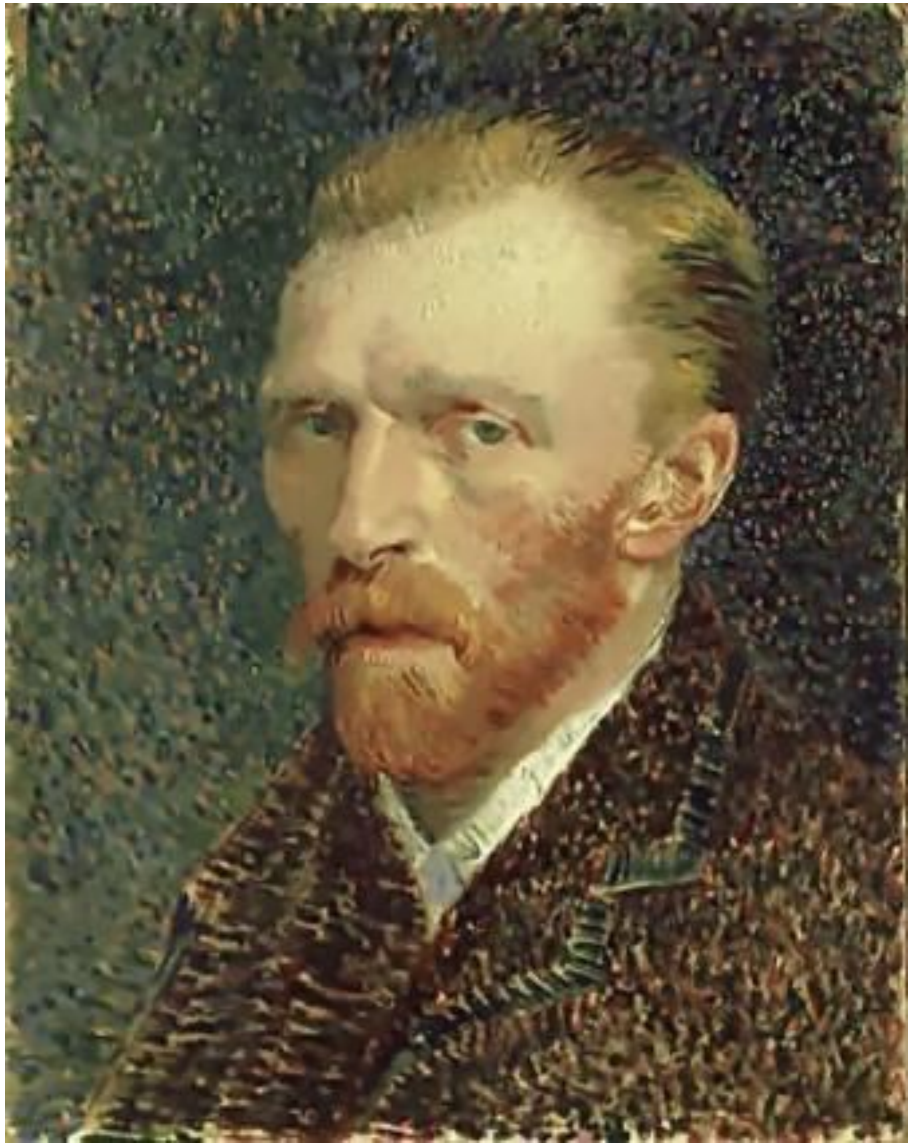
Structure preserved

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

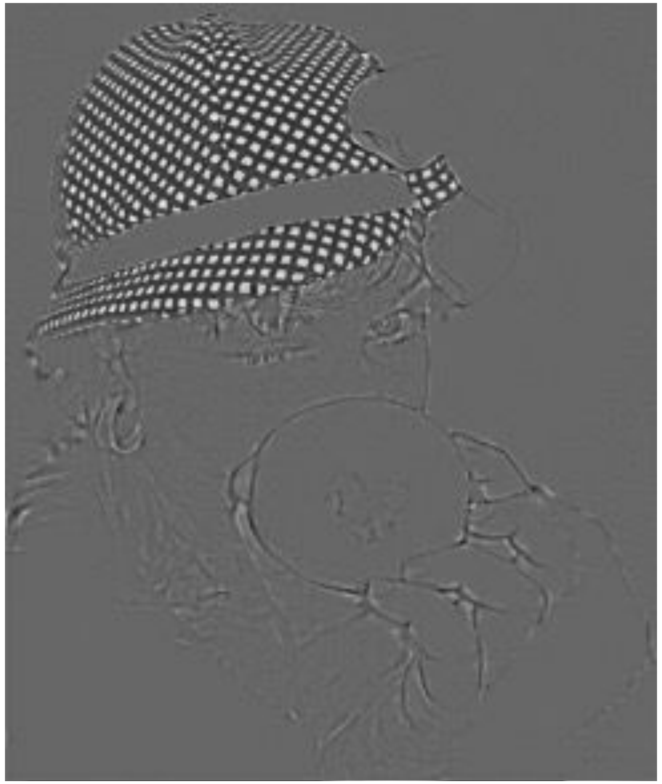
Input

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

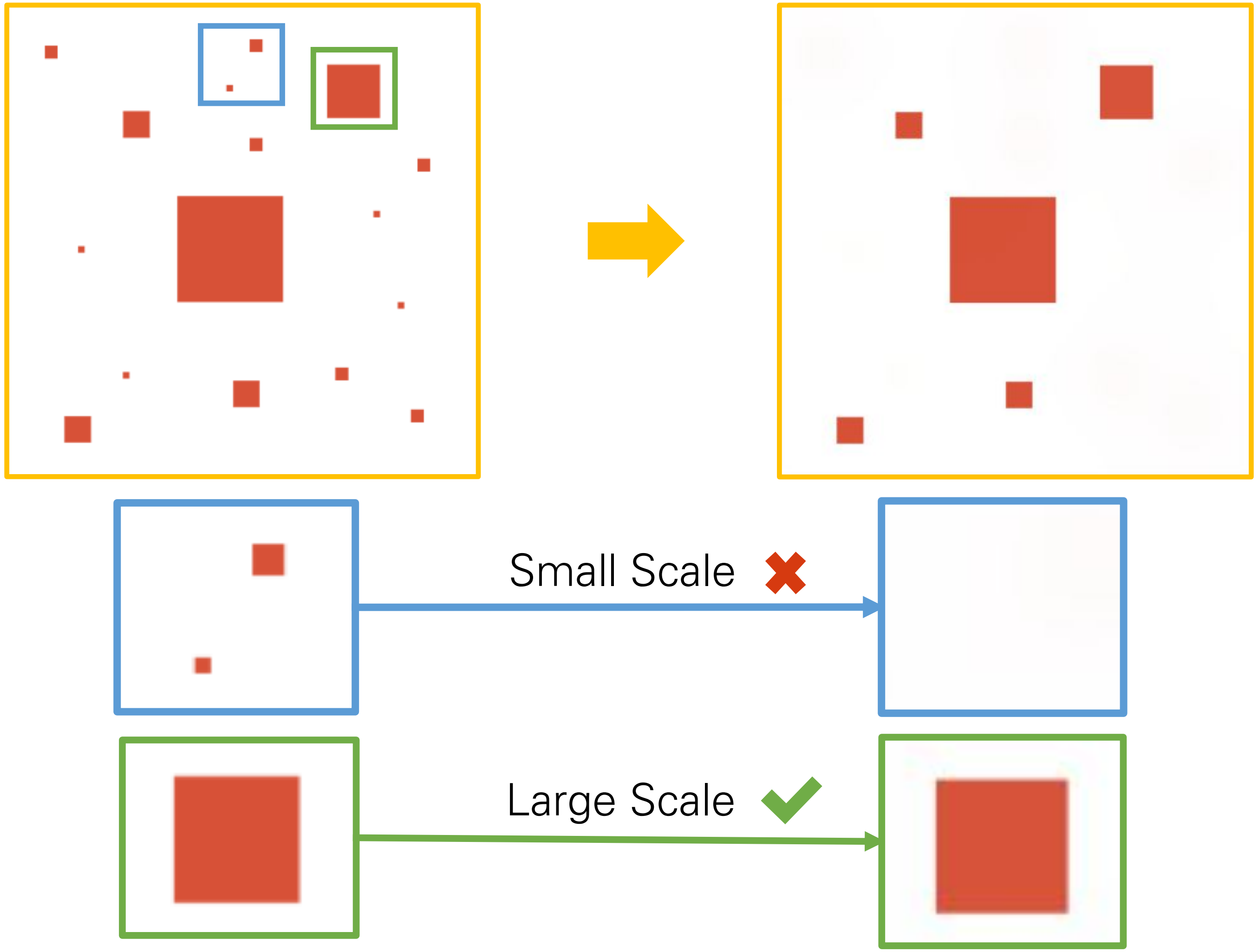
Image abstraction

Detail boosting

Image composition

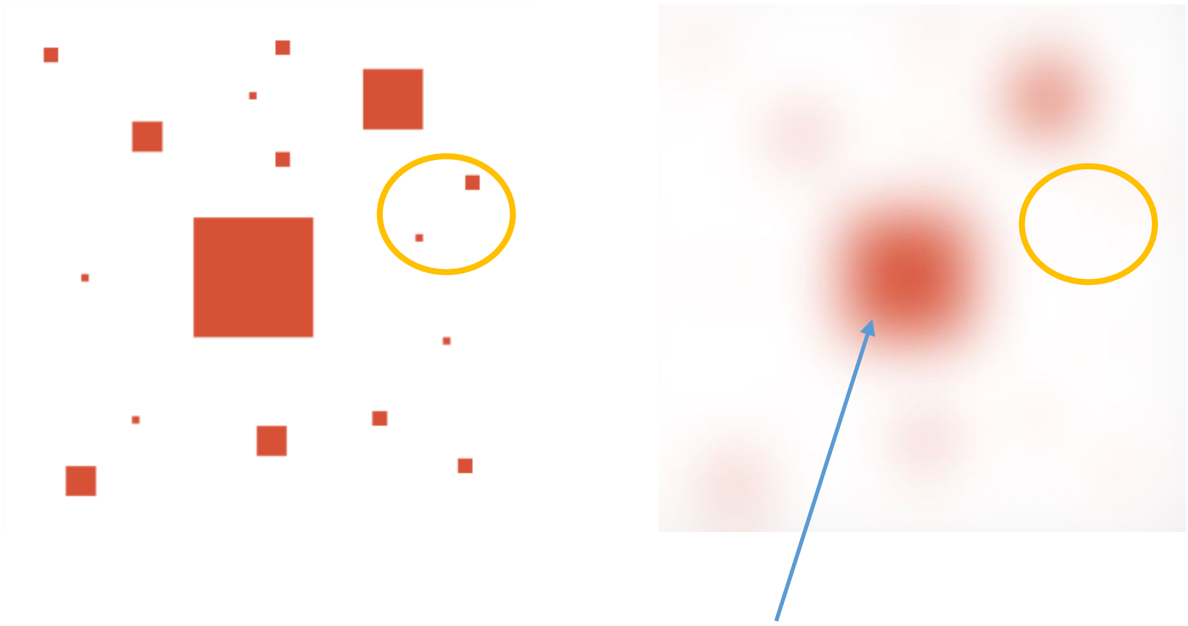
Rolling Guidance Filter

Scale-Aware Filtering

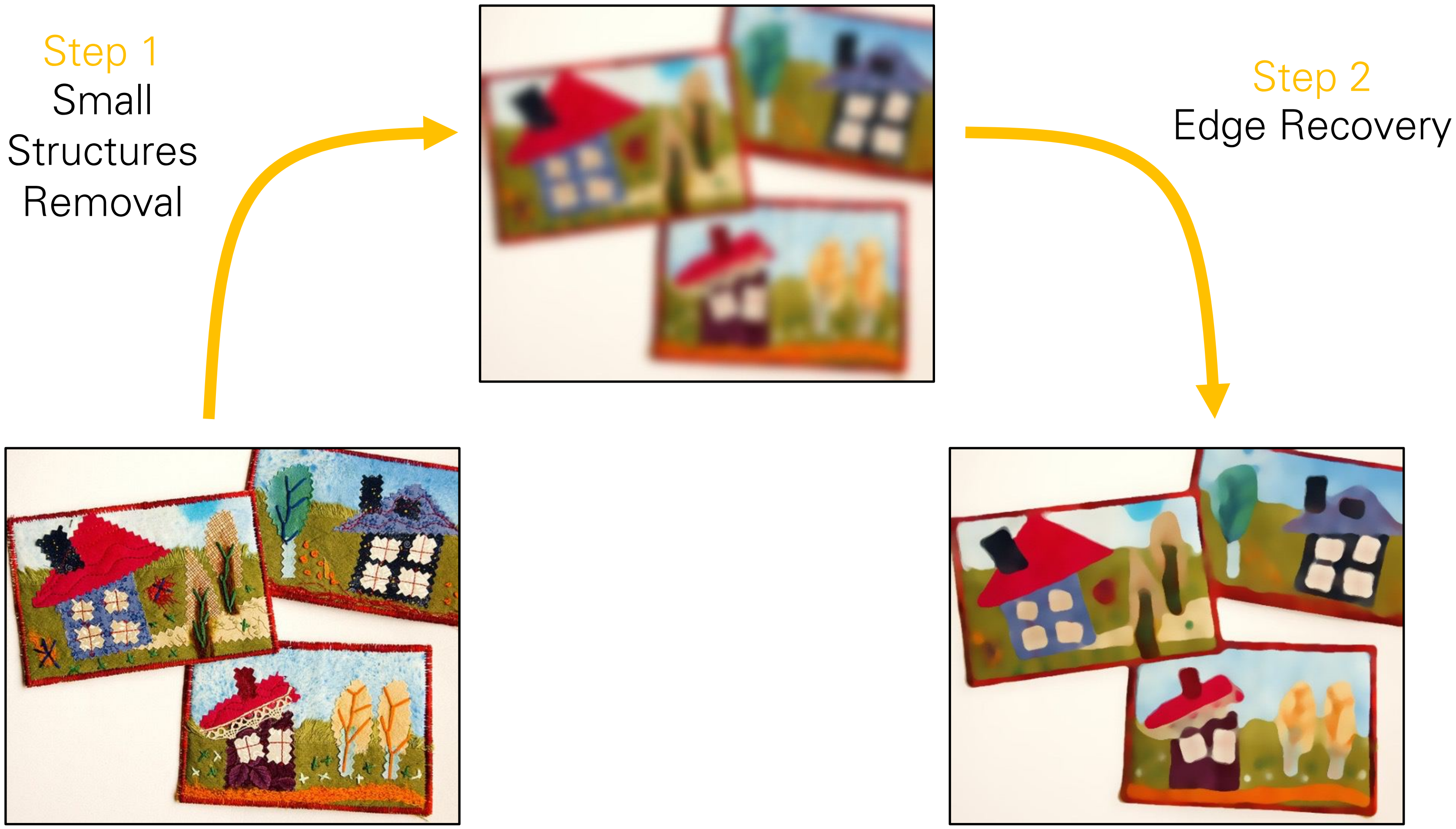


Main Idea

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

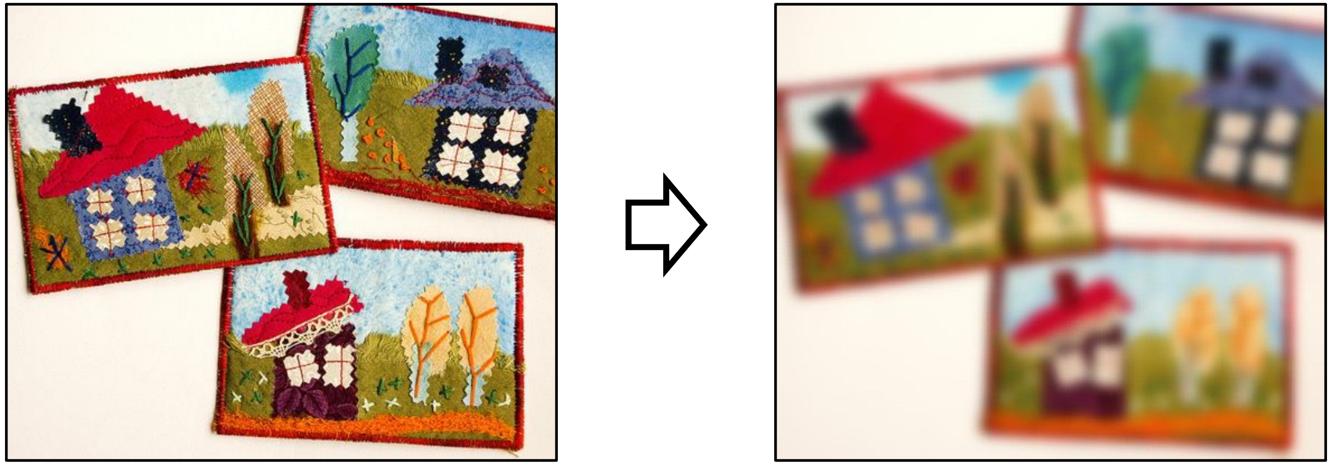


RGF: A scale-aware Filter



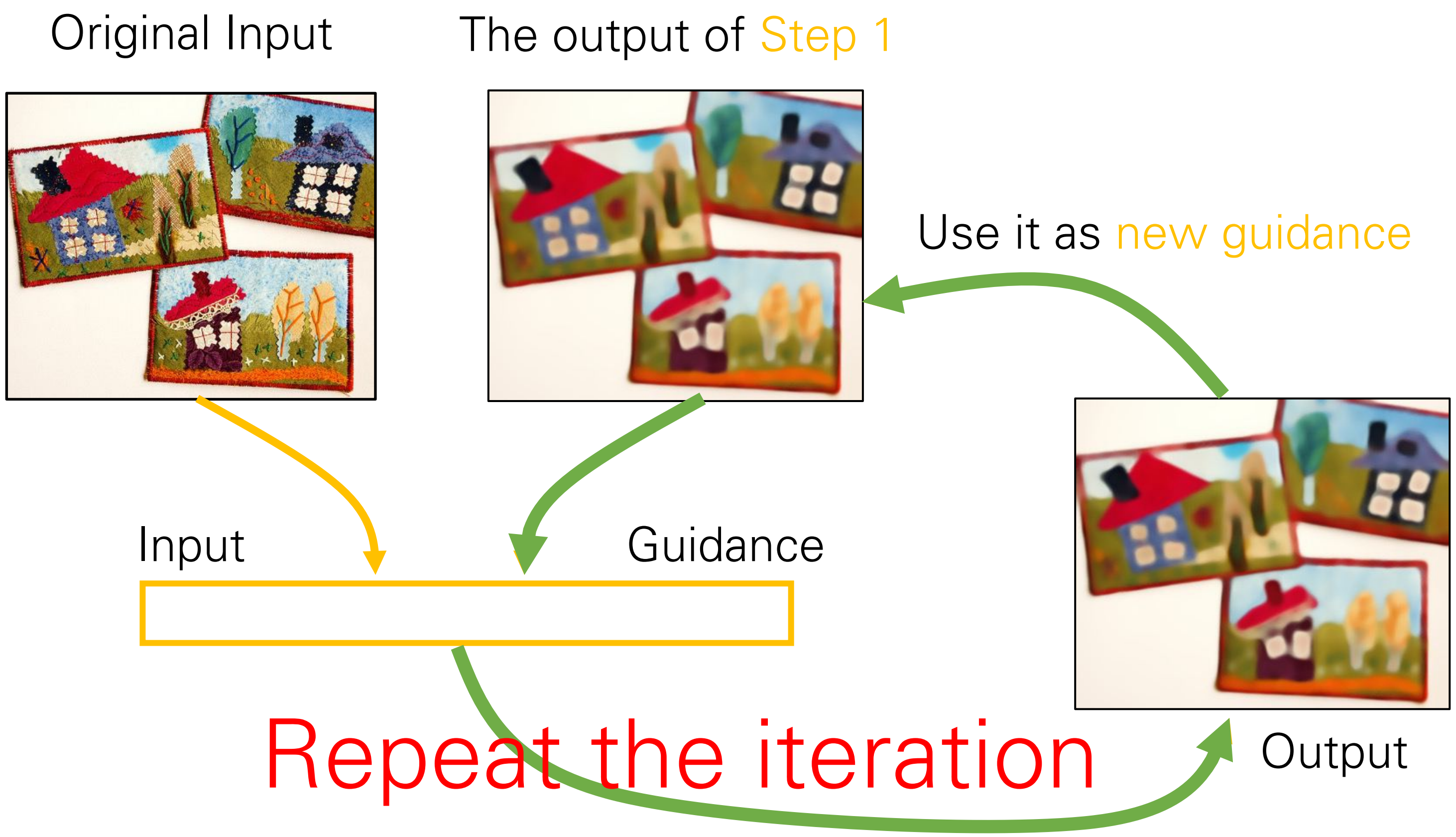
Step 1: Small Structures Removal

Gaussian Filter

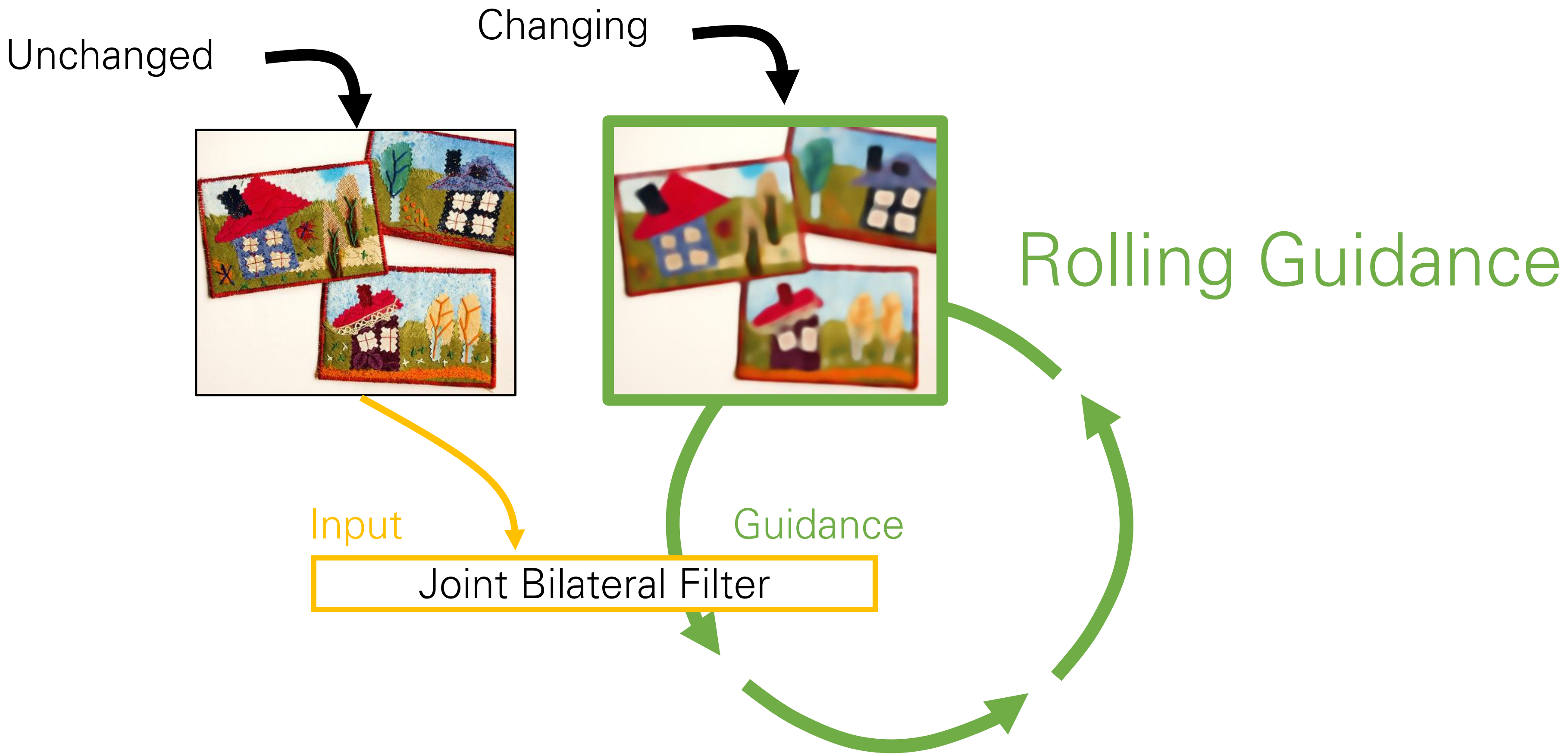


Step 2: Edge Recovery

- A rolling guidance



Rolling Guidance



Rolling Guidance

Guidance for the 1st
iteration

Rolling Guidance

Guidance for the 2nd iteration

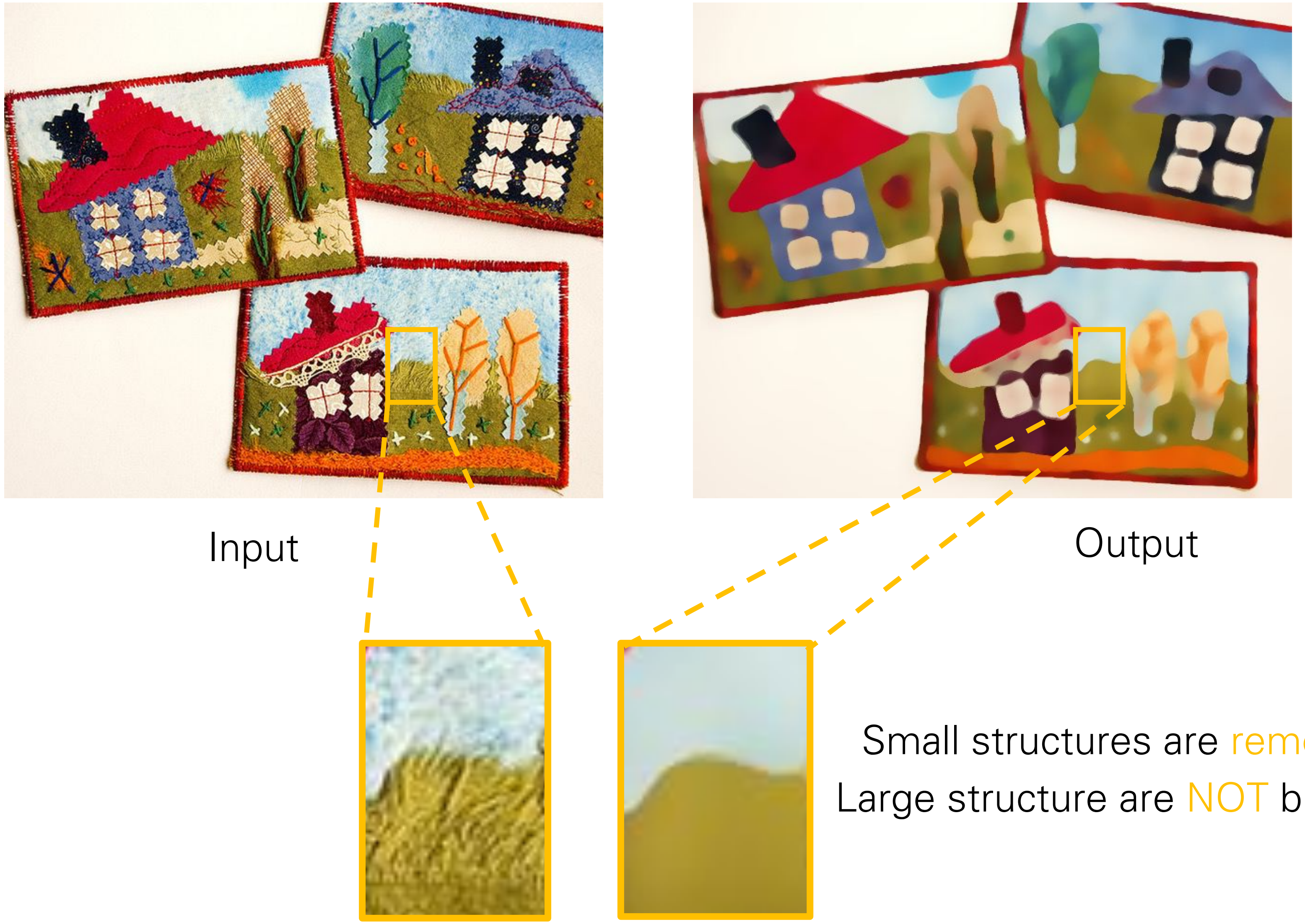
Rolling Guidance

Guidance for the 3rd iteration

Rolling Guidance

Guidance for the 5th
iteration

Rolling Guidance



Small structures are removed.
Large structure are NOT blurred.

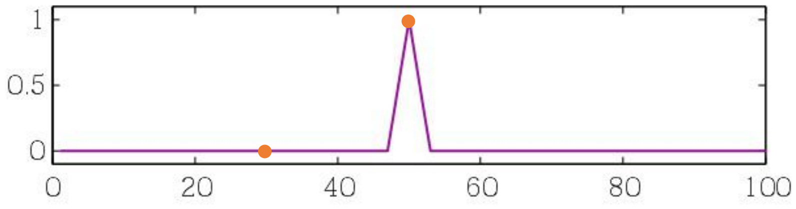
Implementation

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

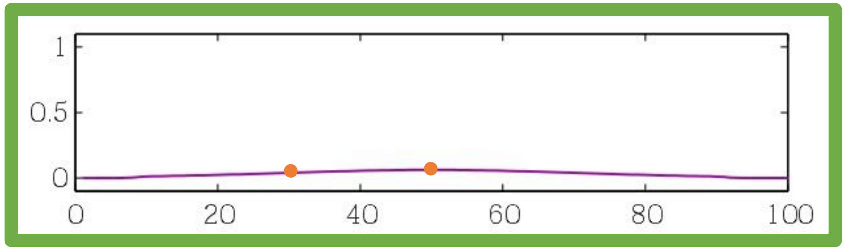
```
Mat rollingGuidanceFilter(Mat im, float scale, int iter){
    Mat res = im.mul(0);
    while(iter--) res = bilateralFilter(im,res,scale,SIGMA_R);
    return res;
}
```

Small Structure

Input



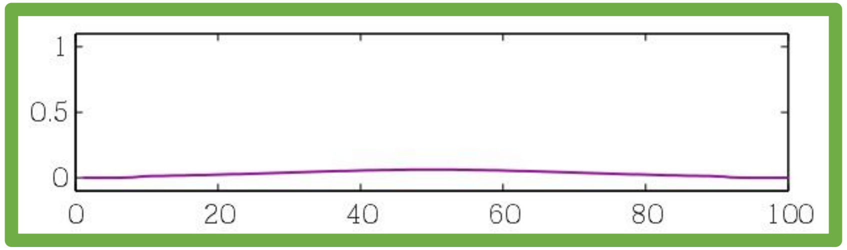
Guidance (output of step 1)



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

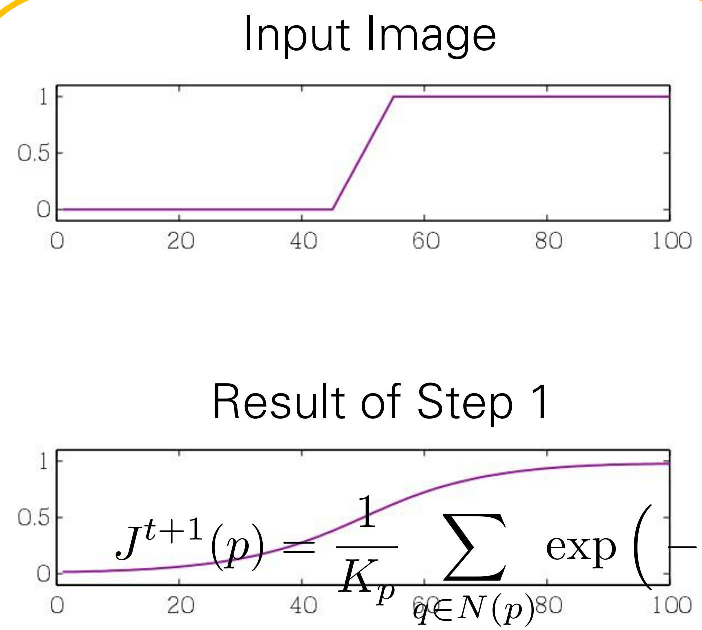
It becomes a Gaussian filter

Joint Bilateral Filter



Same

Large Structure

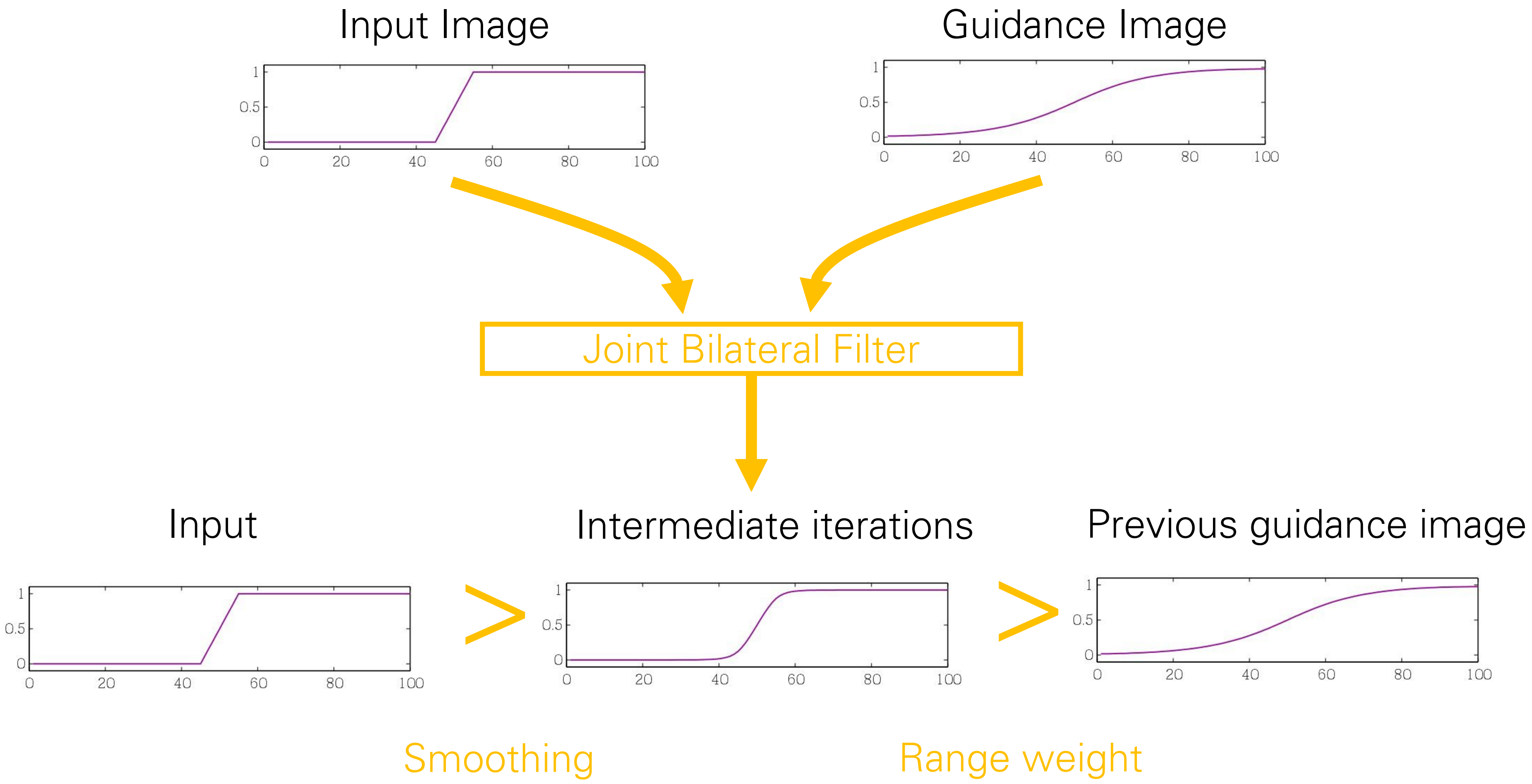


$$J^{t+1}(p) = \frac{1}{K_p} \sum_{q \in N(p)} \exp\left(-\frac{\|p-q\|^2}{2\sigma_s^2} - \boxed{\frac{\|J^t(p) - J^t(q)\|^2}{2\sigma_r^2}} \right) I(q)$$

Due to this range weight

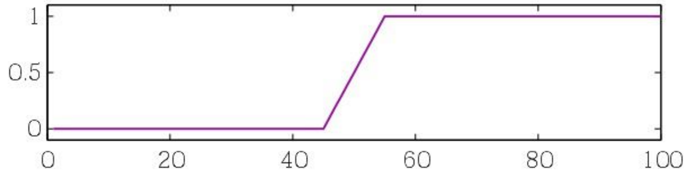
It generates sharper results than Gaussian!

Processing

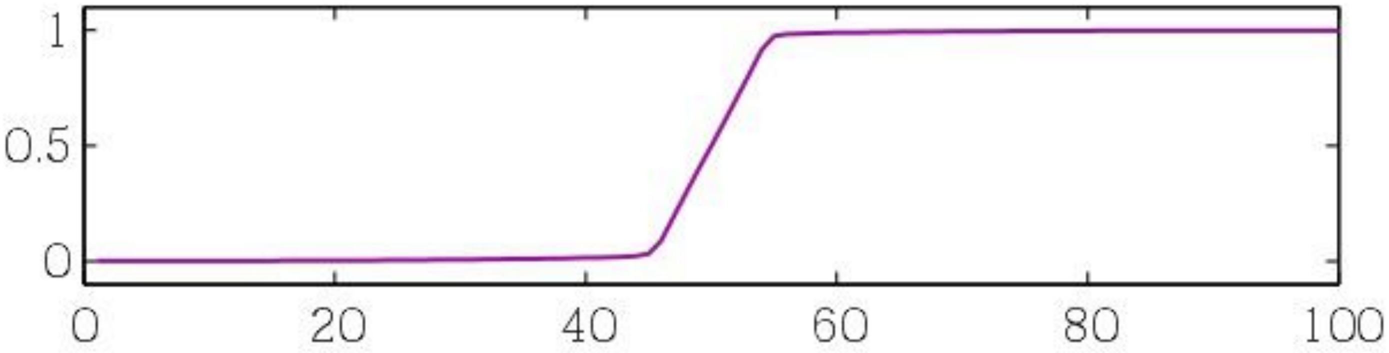

Input Image
Guidance Image
Joint Bilateral Filter
Input
Intermediate iterations
Previous guidance image
>
>
Smoothing
Range weight

Processing

Input Image



Guidance Image



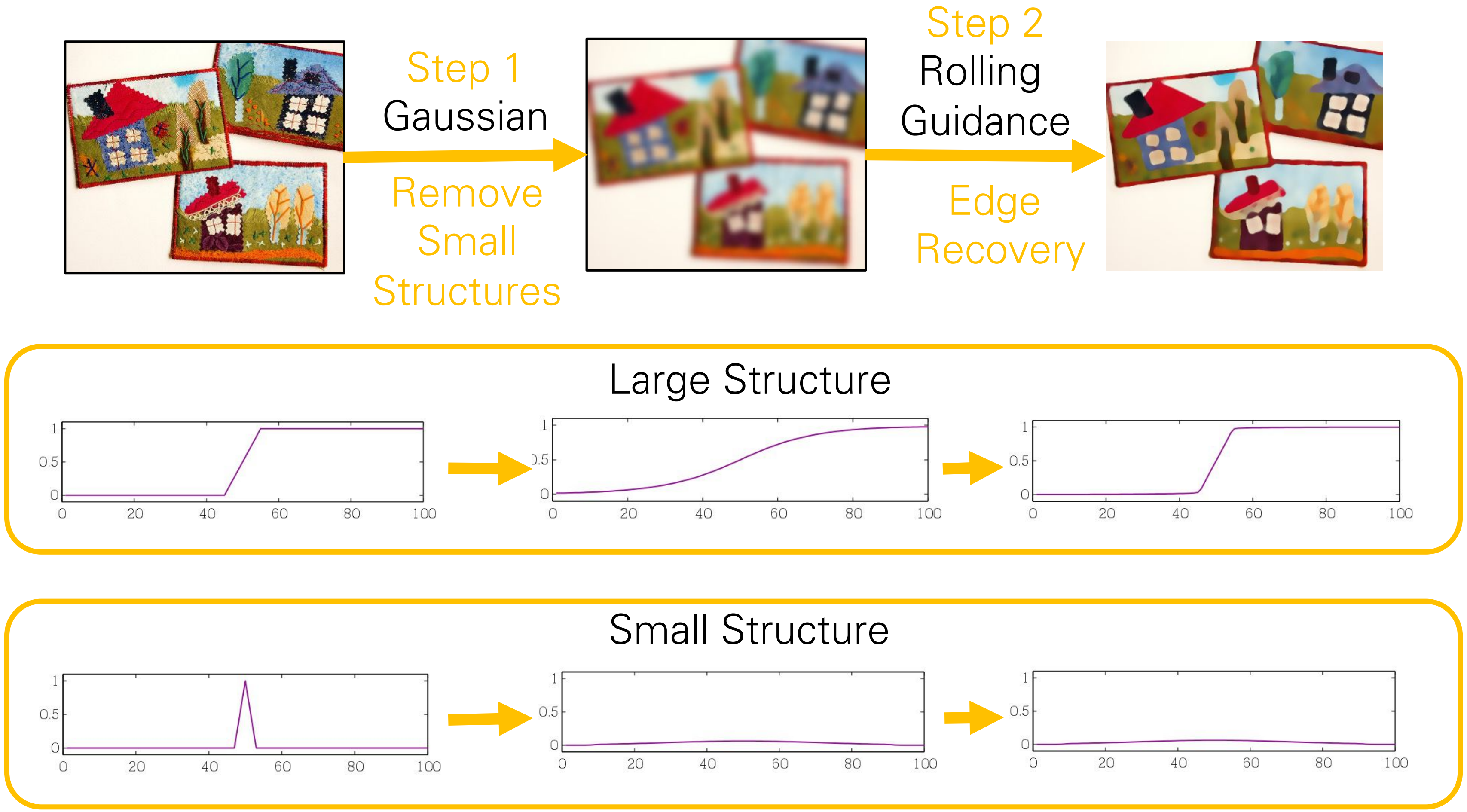
3rd Iteration

Large Structure

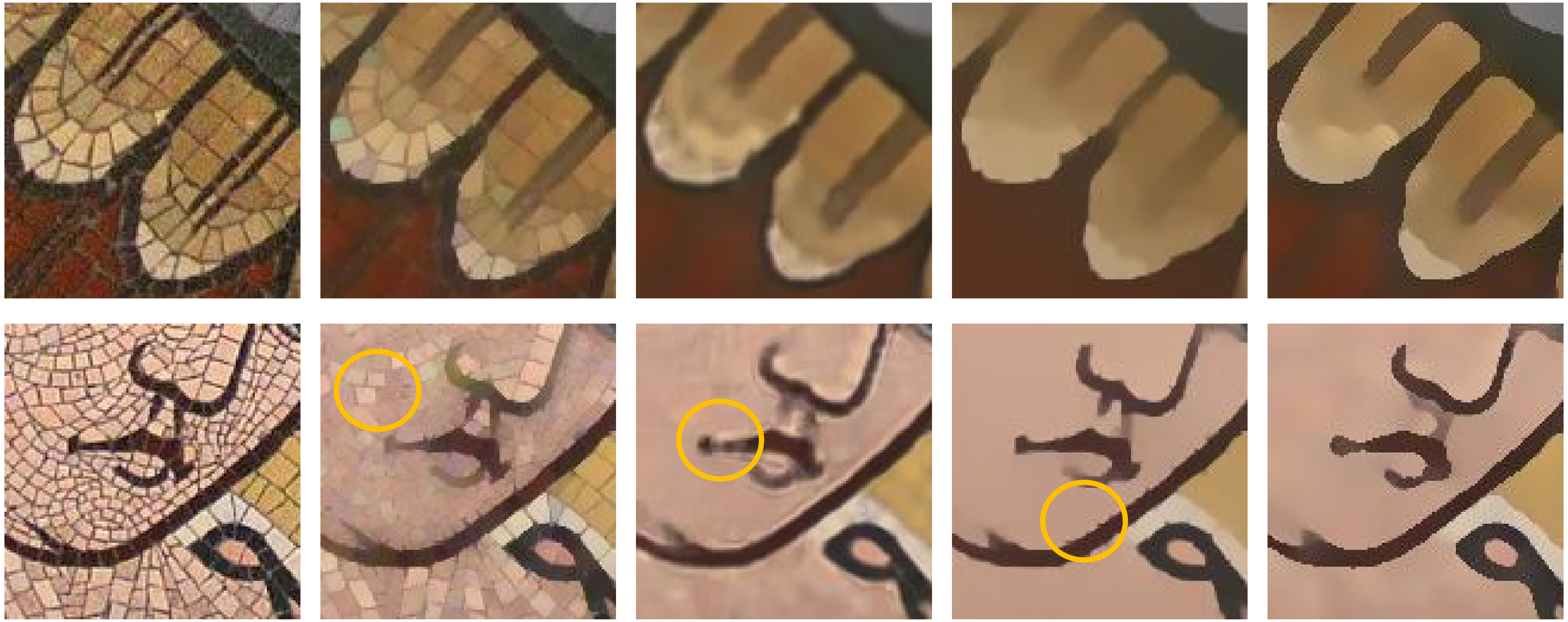
Take-home message

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

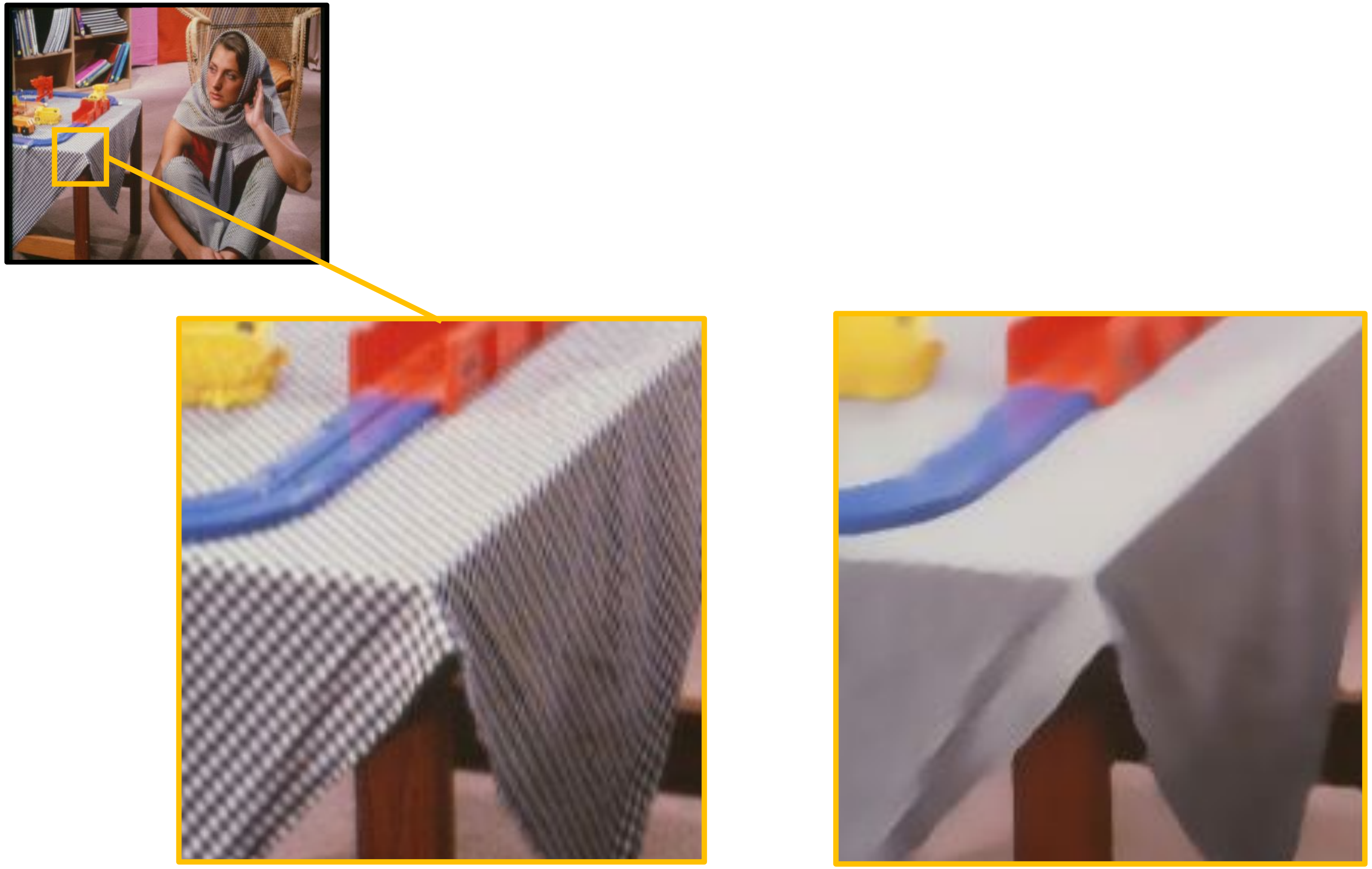
seconds

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)

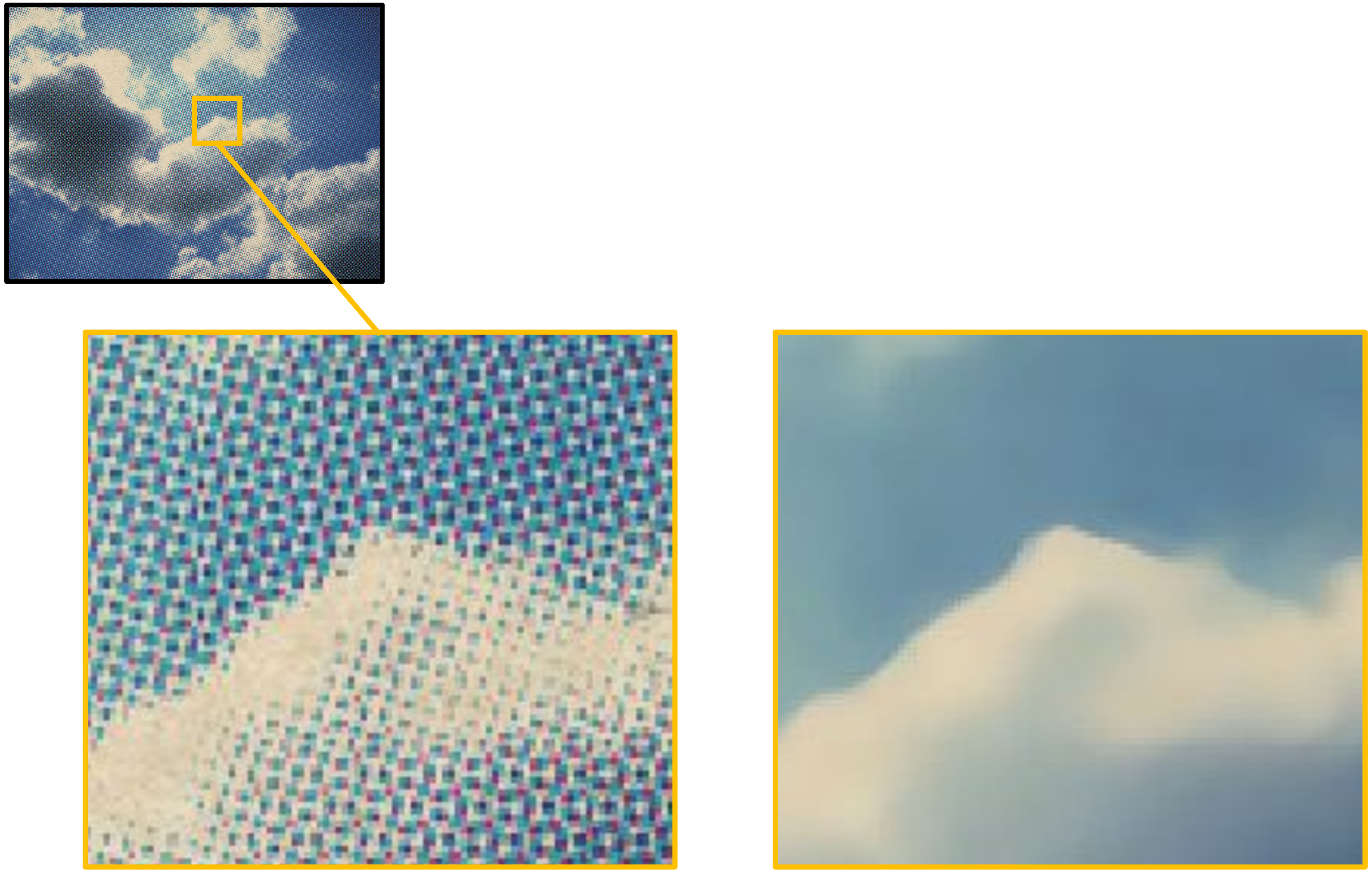
Texture Removal

Texture Removal



Halftone Image

Halftone Image

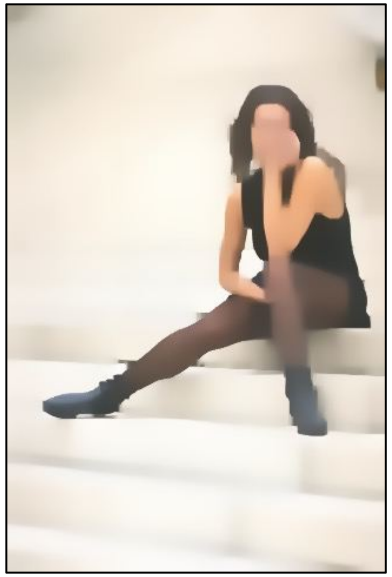


Boundary detection

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:
Edge-aware filtering,
Gradient-domain image
processing