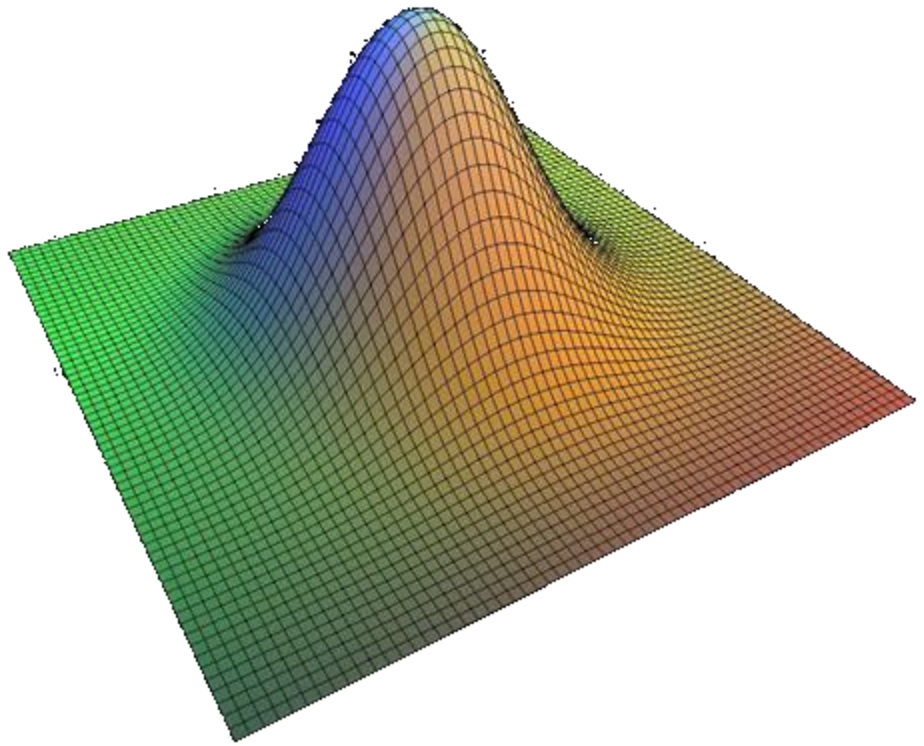
4. Filters



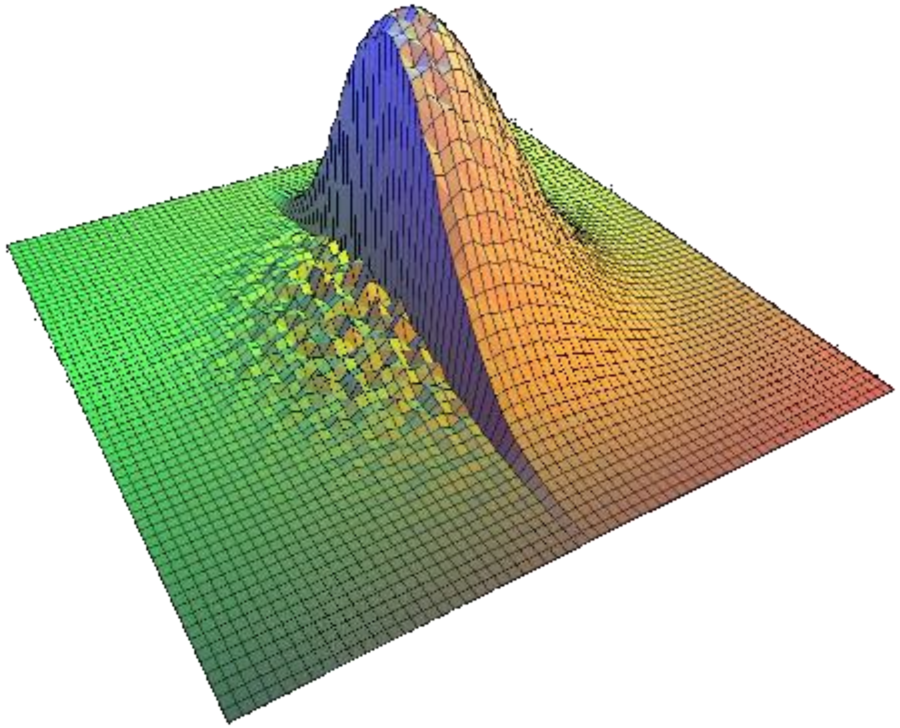
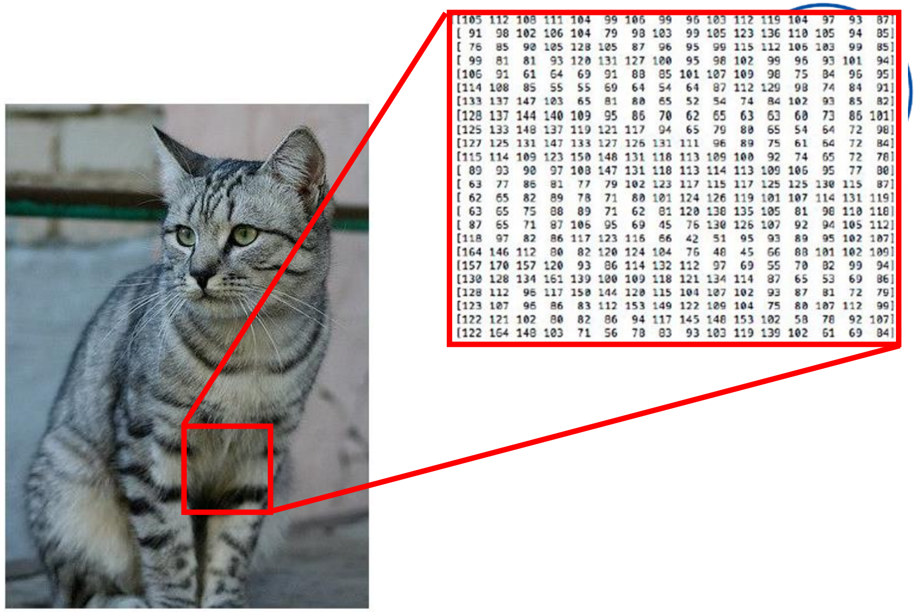


Image Filtering



- An image is a 2D array of pixel values
- Filtering:
  - Replace each pixel by a *linear* combination of its neighbors
  - The combination is determined by the filter's *kernel*
  - Often spatially-invariant, the same kernel is applied to all pixel locations

$$g[\cdot,\cdot] = \frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Gaussian Filter

$$g[\cdot,\cdot] = \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Box Filter

Box filter example

$g[\cdot,\cdot]$

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $f[\cdot,\cdot]$

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	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	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

output $h[\cdot,\cdot]$

	0	10	20						

$$\underbrace{h[m,n]}_{\text{output}} = \sum_{k,l} \underbrace{g[k,l]}_{\text{filter}} \underbrace{f[m+k,n+l]}_{\text{image (signal)}}$$

Box filter example

$g[\cdot,\cdot]$

kernel

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

image $f[\cdot,\cdot]$

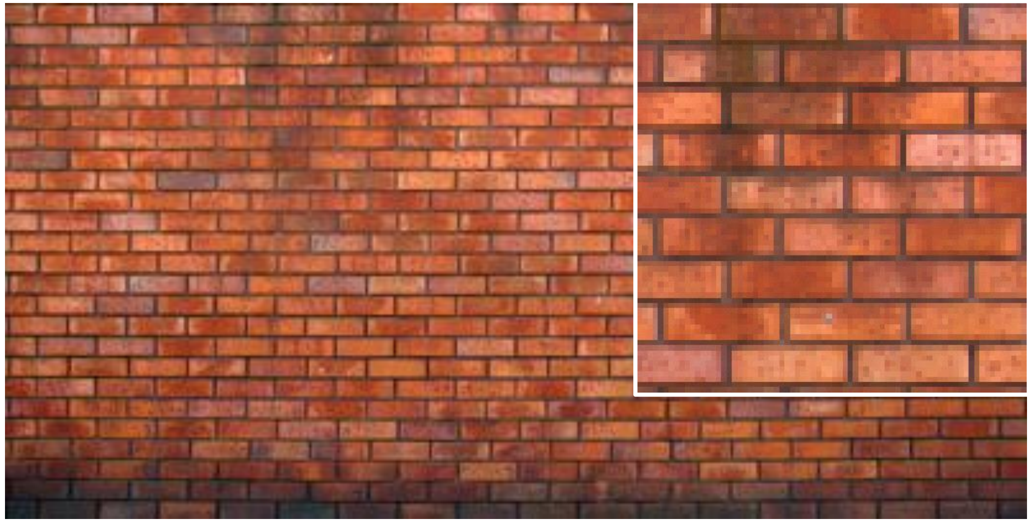
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

output $h[\cdot,\cdot]$

	0	10	20	30					

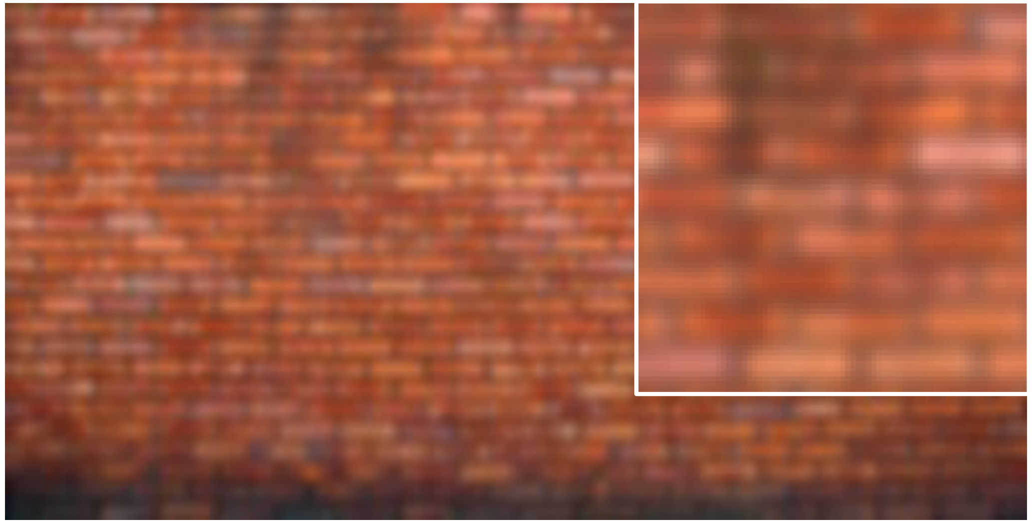
$$\underbrace{h[m,n]}_{\text{output}} = \sum_{k,l} \underbrace{g[k,l]}_{\text{filter}} \underbrace{f[m+k,n+l]}_{\text{image (signal)}}$$

Gaussian vs box filtering

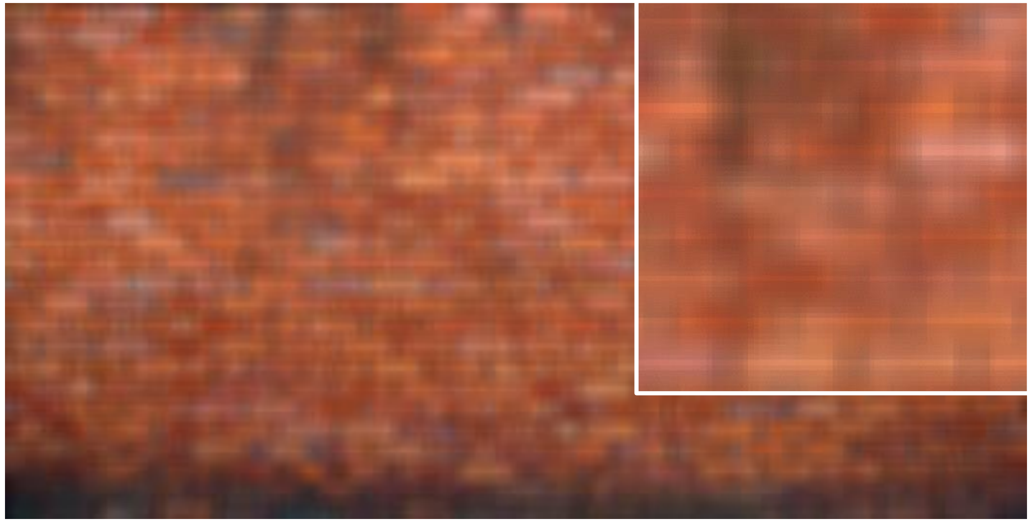


original

Which blur do you like better?



7x7 Gaussian



7x7 box

How to create a soft shadow effect?

shift + overlay

Gaussian blur

Fun with filters

input

filter

0	0	0
0	1	0
0	0	0

output

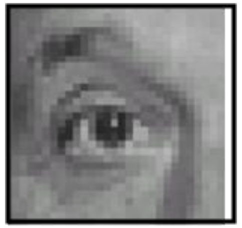
unchanged

input

filter

0	0	0
0	0	1
0	0	0

output



shift to left
by one

Detecting Edges

- How would you detect edges in an image (i.e., discontinuities in a function)?
  - Take derivatives: derivatives are large at discontinuities

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

- How do you differentiate a discrete image (or any other discrete signal)?
  - Use finite differences

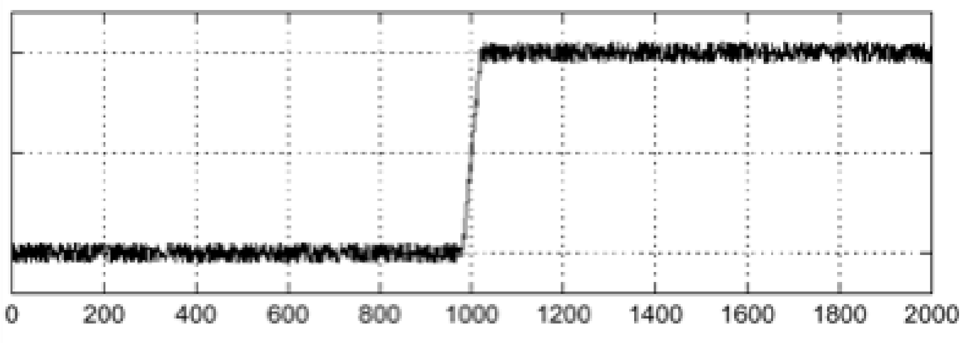
-1	0	1

1	0	-1

Remove limit and set $h = 2$

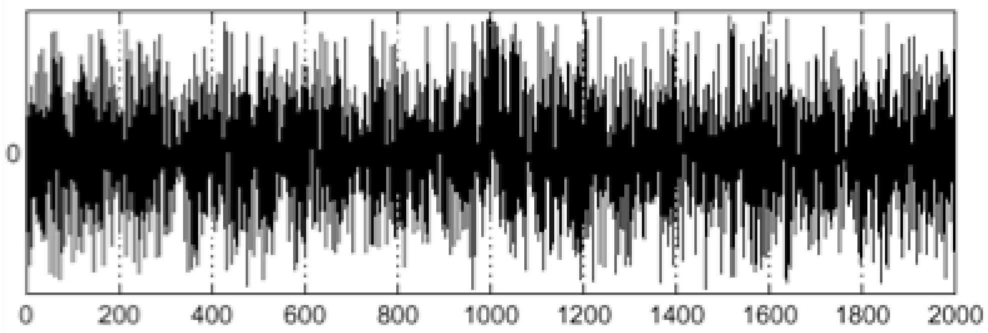
But Images are Noisy

intensity plot



Using a derivative filter:

derivative plot



Blur Before Taking Derivatives

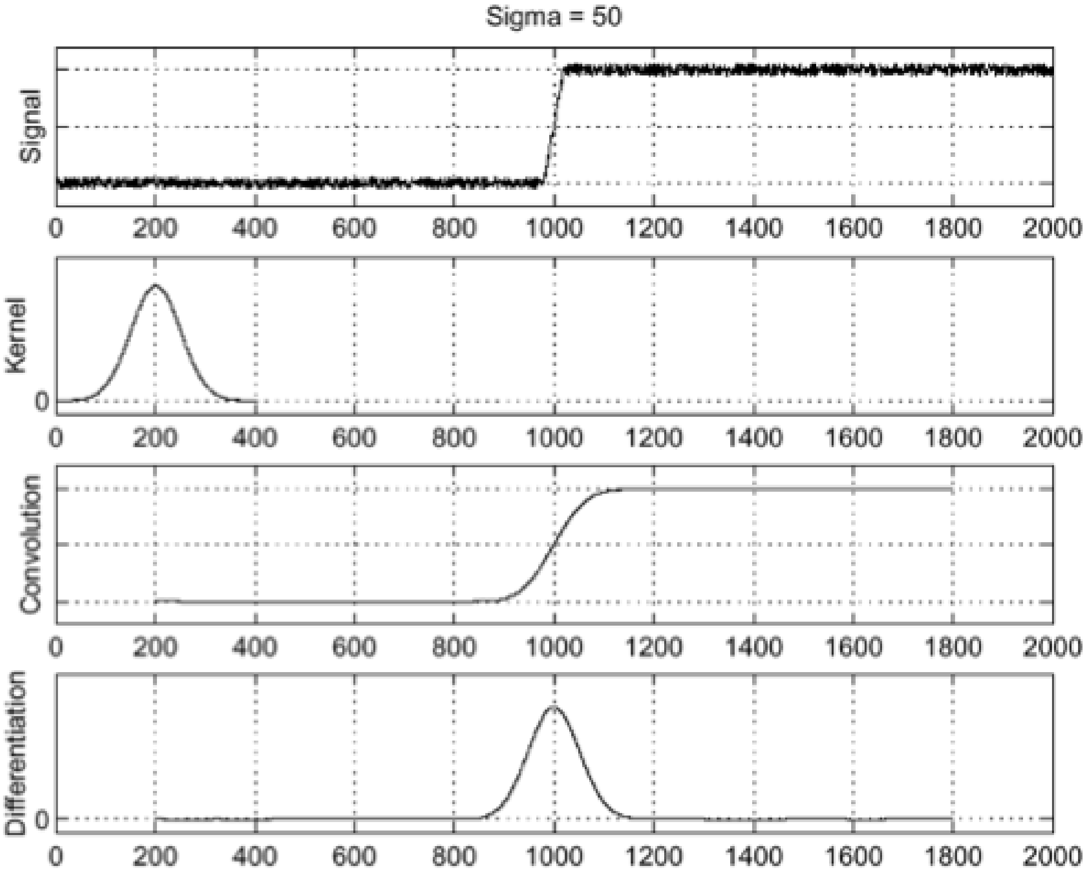
- When using derivative filters, it is critical to blur first!

input

Gaussian

blurred

derivative of blurred



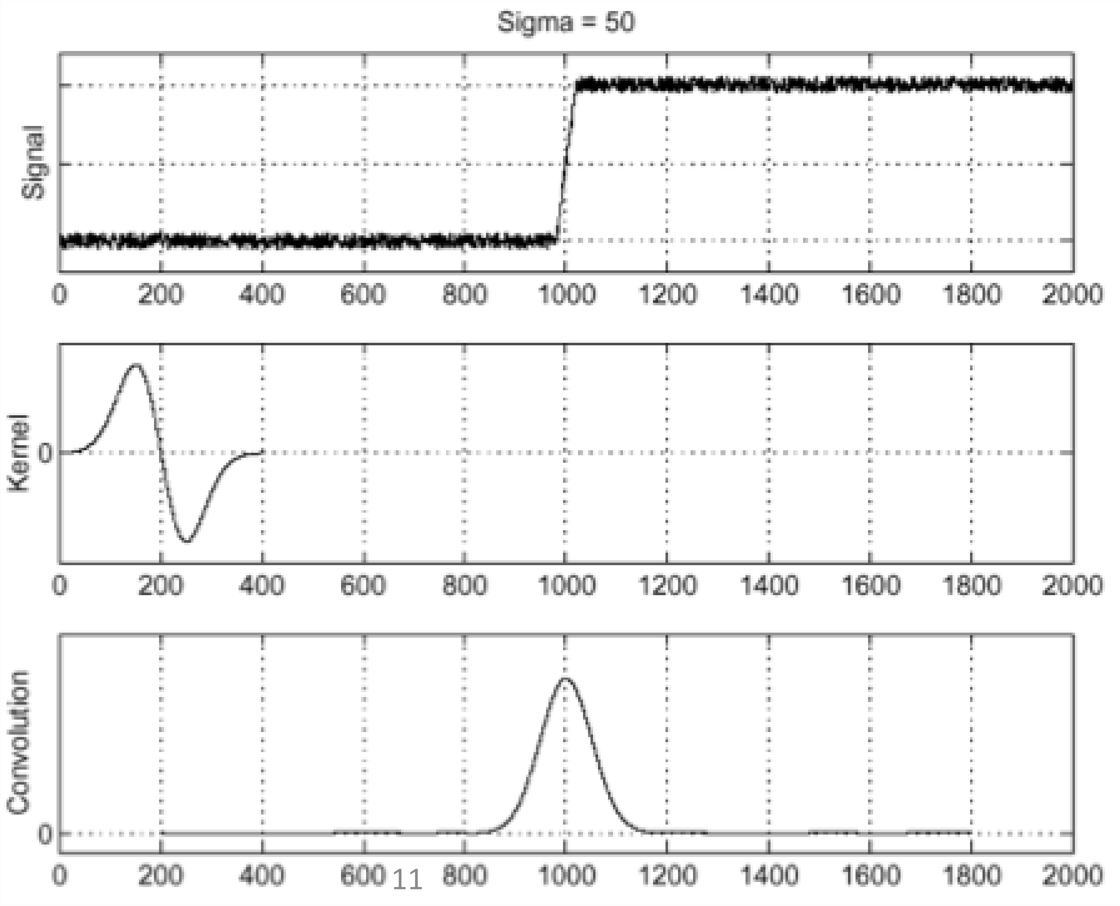
Derivative of Gaussian (DoG) filter

- Derivative theorem of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$
- Applying DoG = Applying blur first and then taking derivative

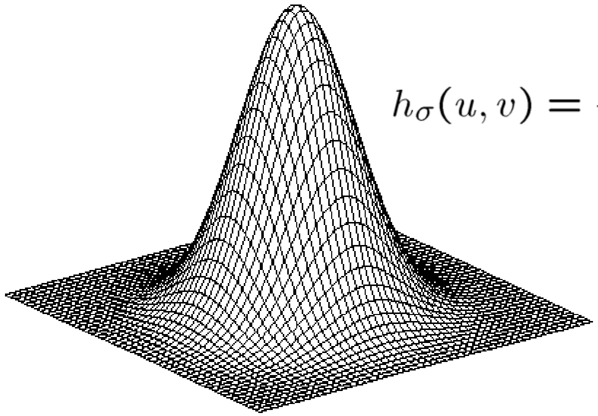
input

derivative of Gaussian

output (same as before)

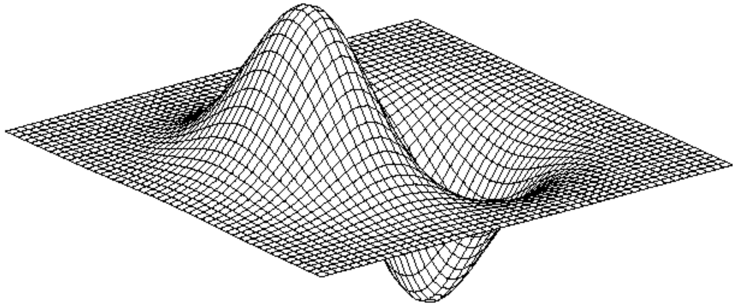


2D Gaussian filters



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

Gaussian



$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Derivative of Gaussian

Derivative of Gaussian filtering

The Sobel filter

Horizontal Sobel filter:

$$\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \end{array}$$

Sobel filter

Blurring

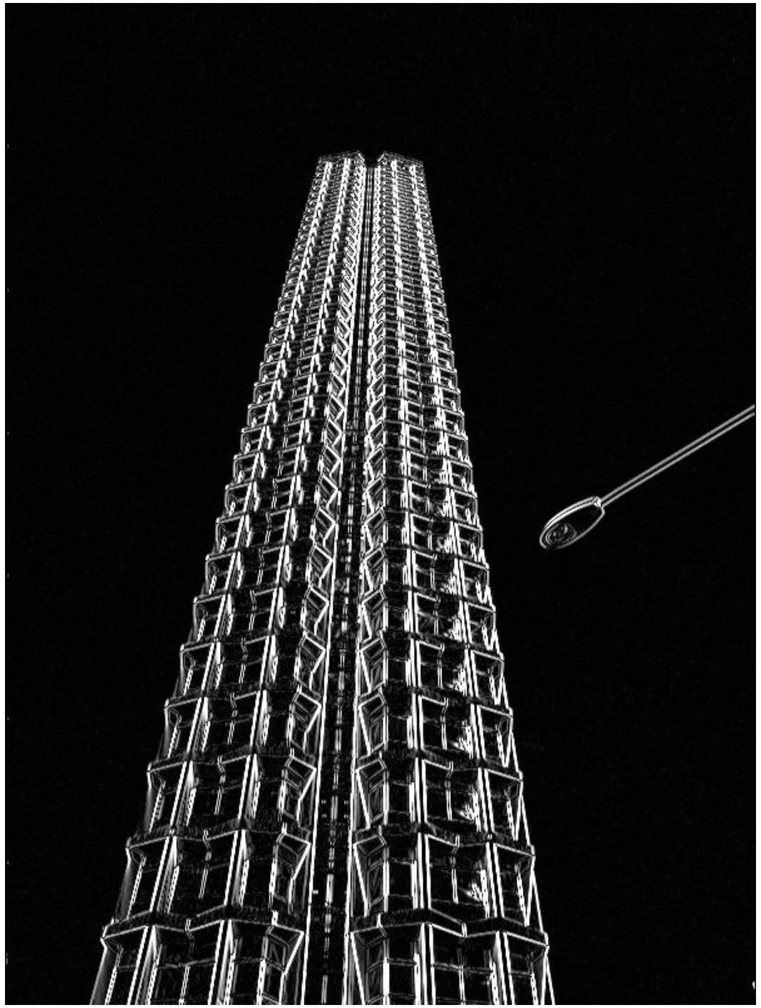
1D derivative filter

Vertical Sobel filter:

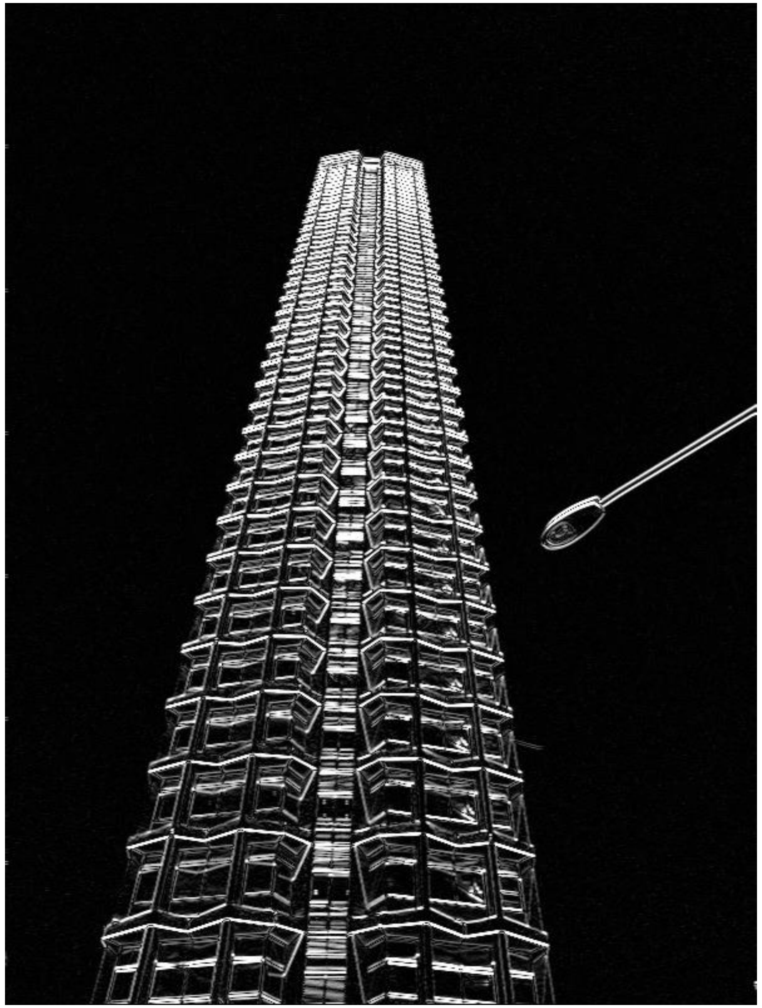
$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

Sobel filter example

original

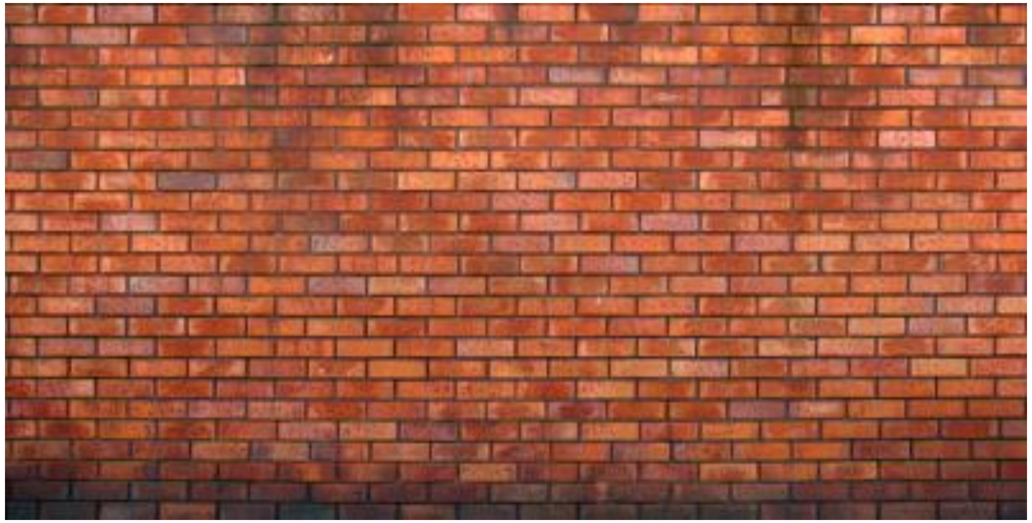


horizontal Sobel filter



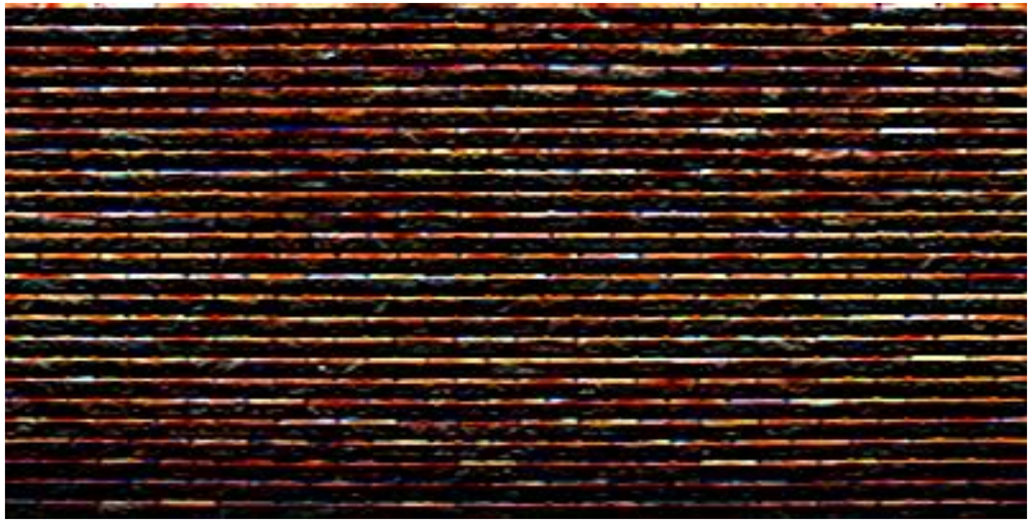
vertical Sobel filter

Sobel filter example



original

horizontal Sobel filter



vertical Sobel filter

Computing Image Gradients

1. Select a derivative filters (there are other similar filters, e.g. Scharr)

$\boldsymbol{S}_x =$

1	0	-1
2	0	-2
1	0	-1

$\boldsymbol{S}_y =$

1	2	1
0	0	0
-1	-2	-1

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right]$$

gradient

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

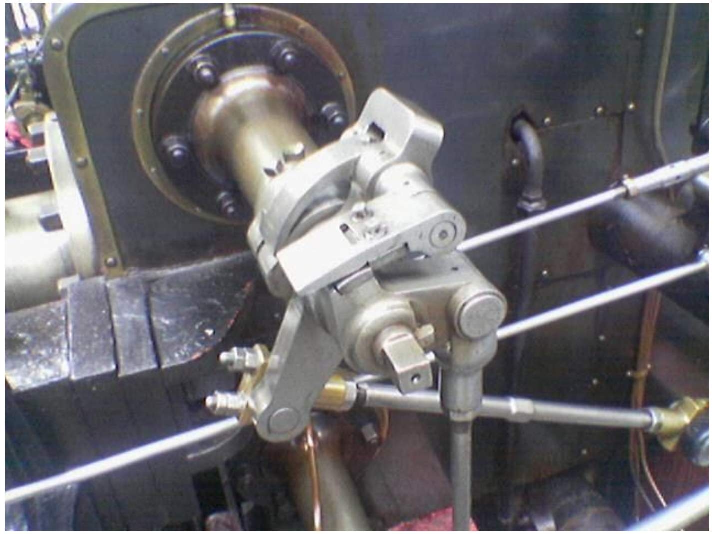
direction

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

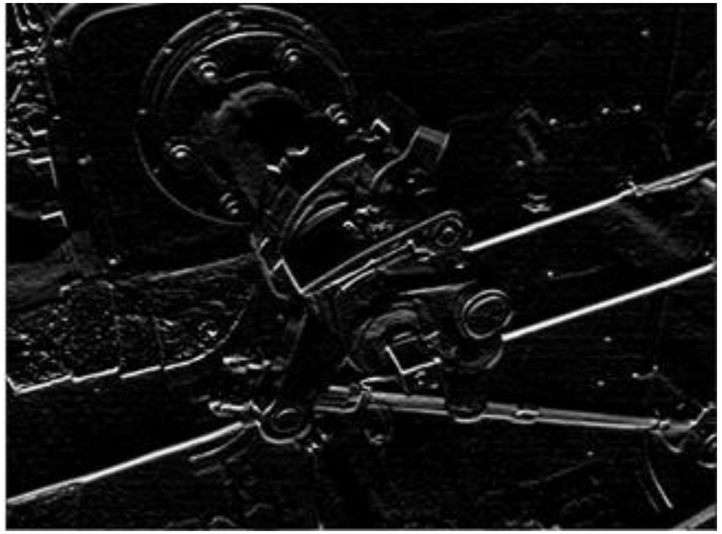
amplitude

Image gradient example

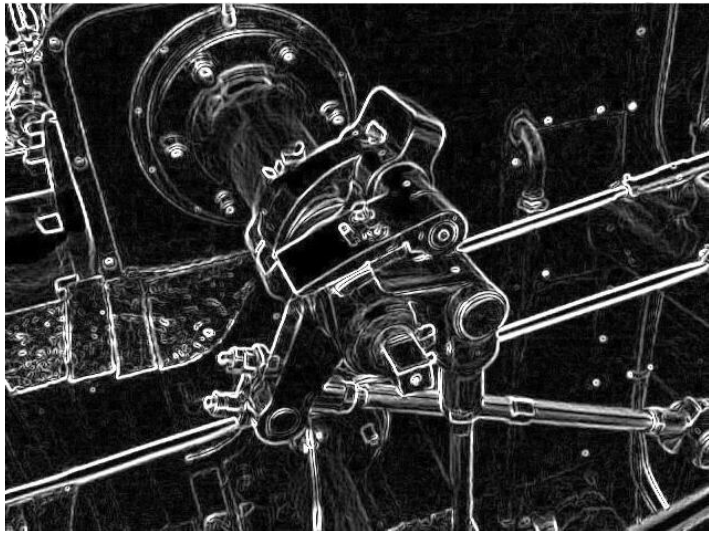
original



vertical derivative



gradient amplitude



horizontal derivative

Questions?

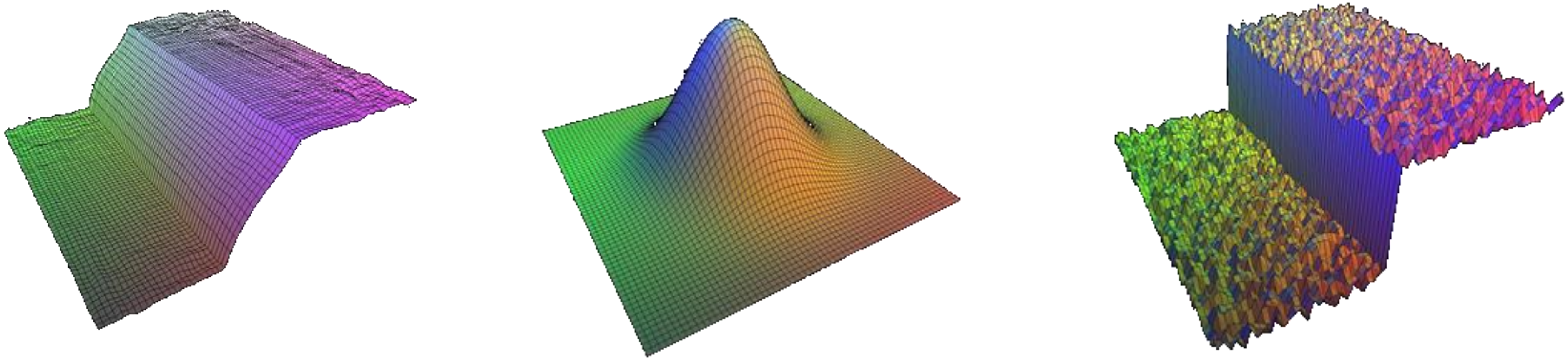
Bilateral filter

- Proposed by Tomasi and Manduci in ICCV 1998
  - "Bilateral Filtering for Gray and Color Images"
- A very good survey by Sylvain Paris et al. 2009
  - Published at Foundations and Trends in Computer Graphics and Vision
  - "Bilateral Filtering: Theory and Applications"
- Related to
  - SUSAN filter
    [Smith and Brady 95] http://citeseer.ist.psu.edu/smith95susan.html
  - Digital-TV [Chan, Osher and Chen 2001]
    http://citeseer.ist.psu.edu/chan01digital.html
  - sigma filter http://www.geogr.ku.dk/CHIPS/Manual/f187.htm

Start with Gaussian Filter

- Here, input is a step function + noise
- Spatial Gaussian filter f
- Output is blurred

$$J = f \otimes I$$

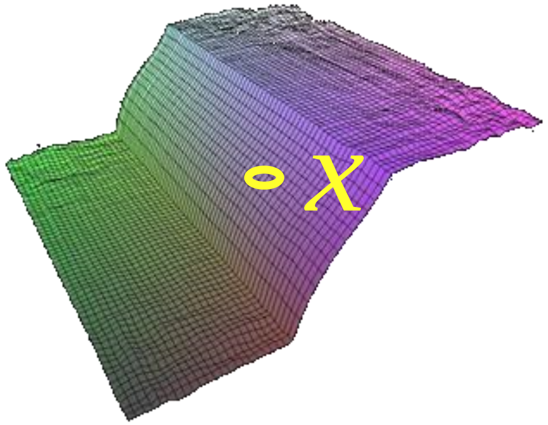


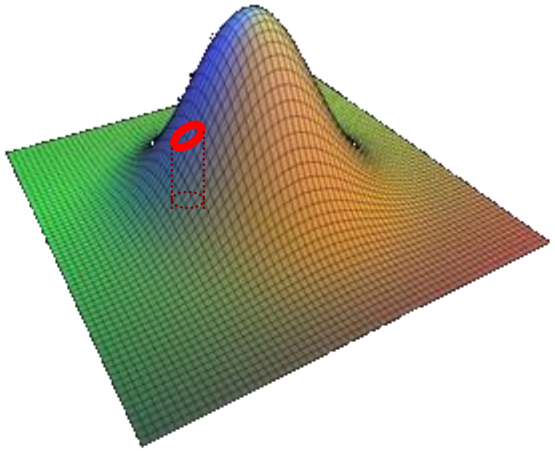
output $\longleftarrow$ input

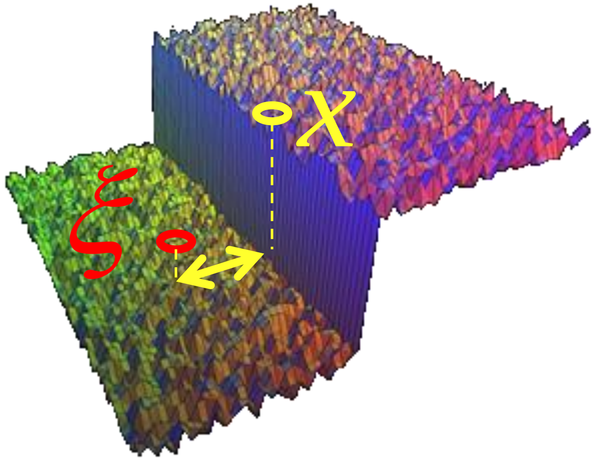
Gaussian Filter as Weighted Average

- Weight of ξ depends on its distance to x

$$J(x) = \sum_{\xi} f(x,\xi)\, I(\xi)$$





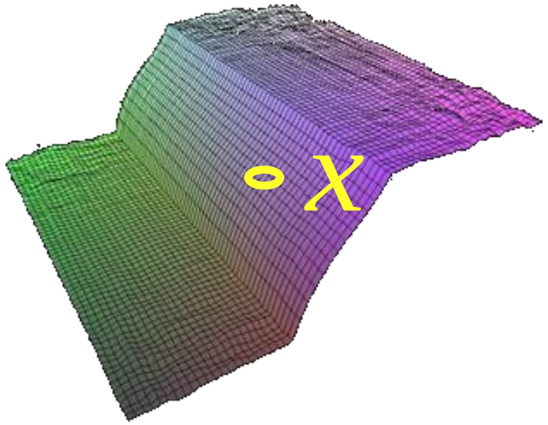


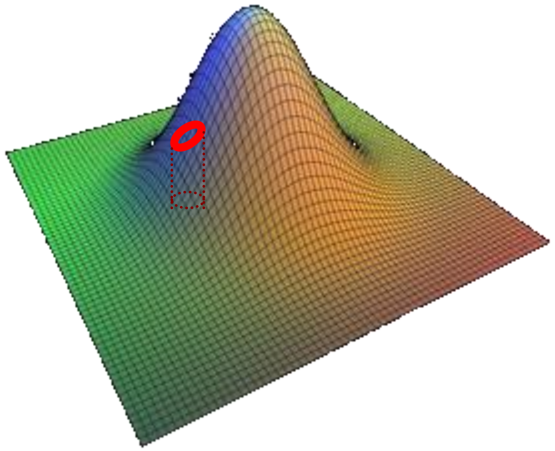
output ⟵ input

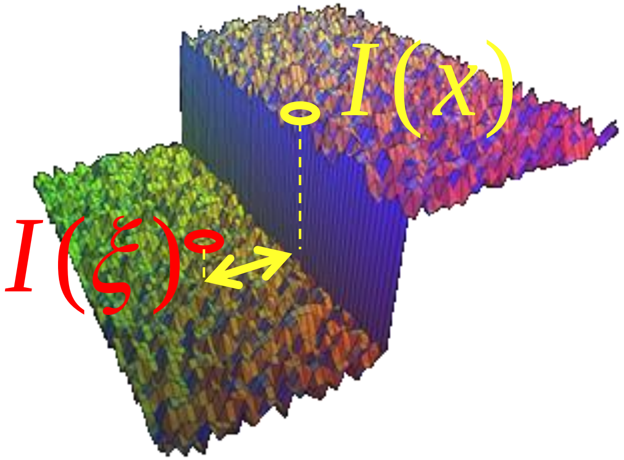
The Problem of Edges

- Here, $I(\xi)$ “pollutes” our estimated $J(x)$
- It is too different to be averaged together

$$J(x) = \sum_{\xi} f(x,\xi)\, I(\xi)$$







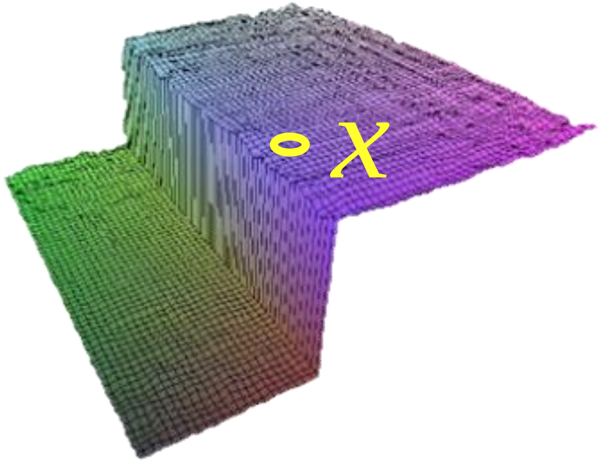
output ⟵ input

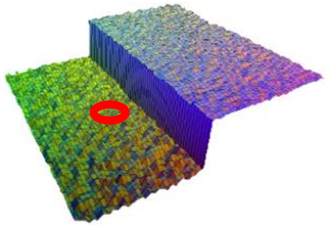
Principle of Bilateral Filter

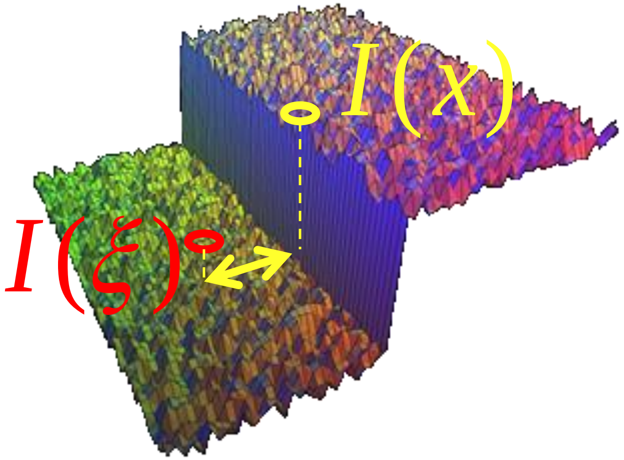
[Tomasi and Manduchi 1998]

- Penalty Gaussian g on the intensity difference

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi)\, g(I(\xi) - I(x))\;\; I(\xi)$$







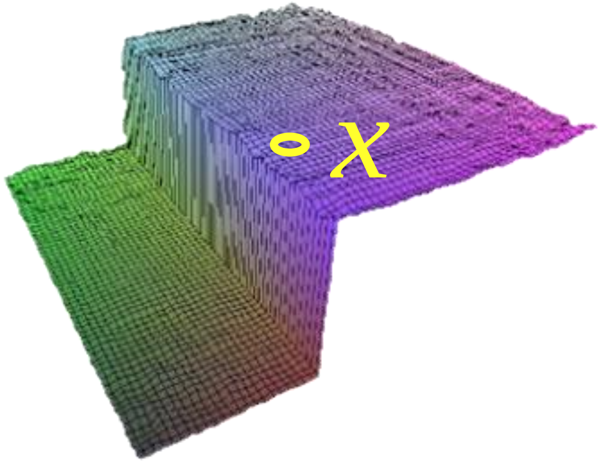
output input

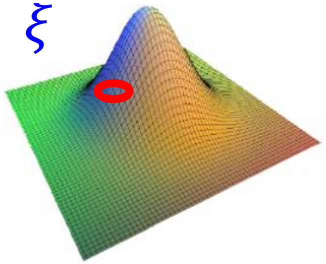
Bilateral Filter

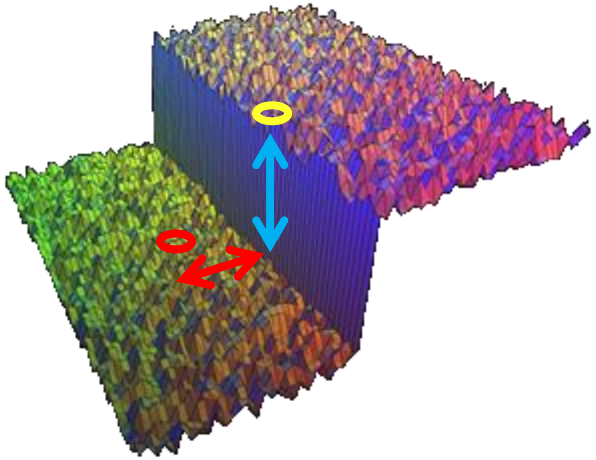
[Tomasi and Manduchi 1998]

- Spatial Gaussian f

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi)\, g(I(\xi) - I(x))\;\; I(\xi)$$







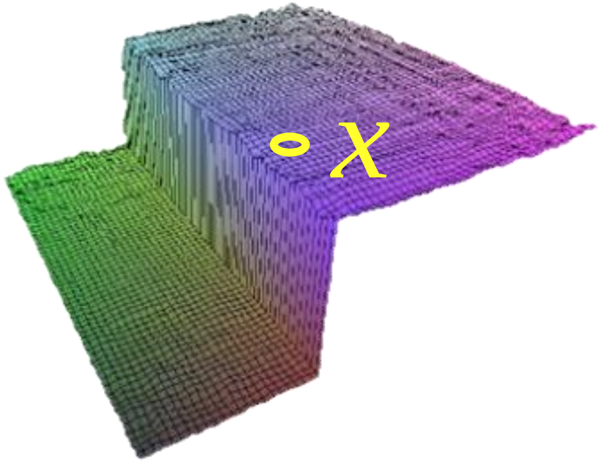
output input

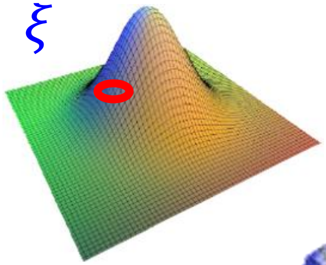
Bilateral Filter

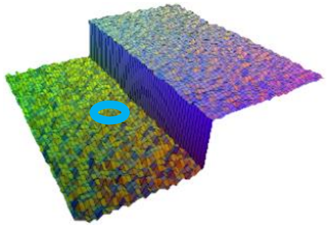
[Tomasi and Manduchi 1998]

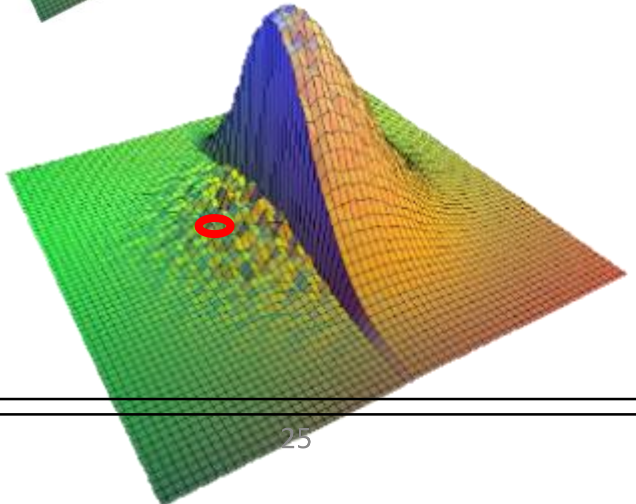
- Combined weight

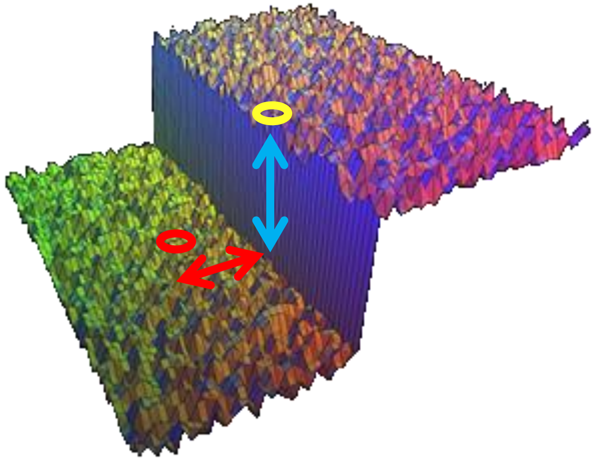
$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi)\, g(I(\xi) - I(x)) \;\; I(\xi)$$











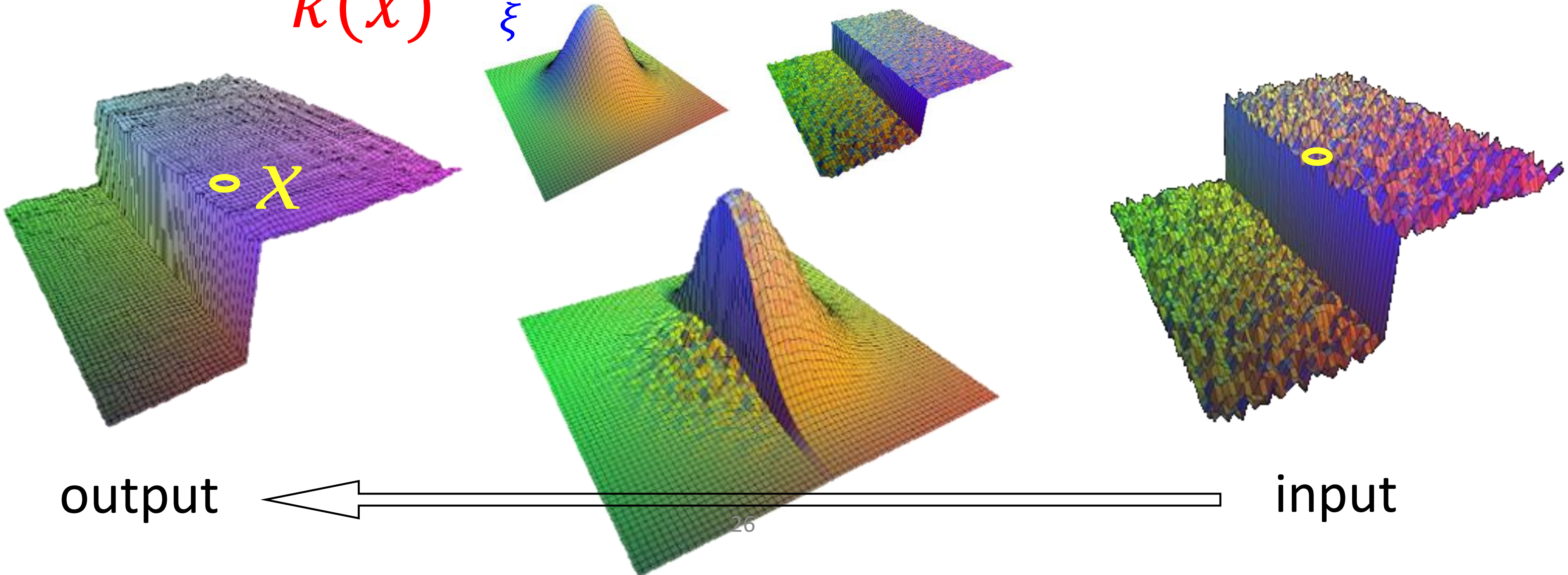
output ⟵ input

Normalization Factor

[Tomasi and Manduchi 1998]

$$k(x) = \sum_{\xi} f(x,\xi)\, g(I(\xi) - I(x))$$

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x,\xi)\, g(I(\xi) - I(x)) \;\; I(\xi)$$

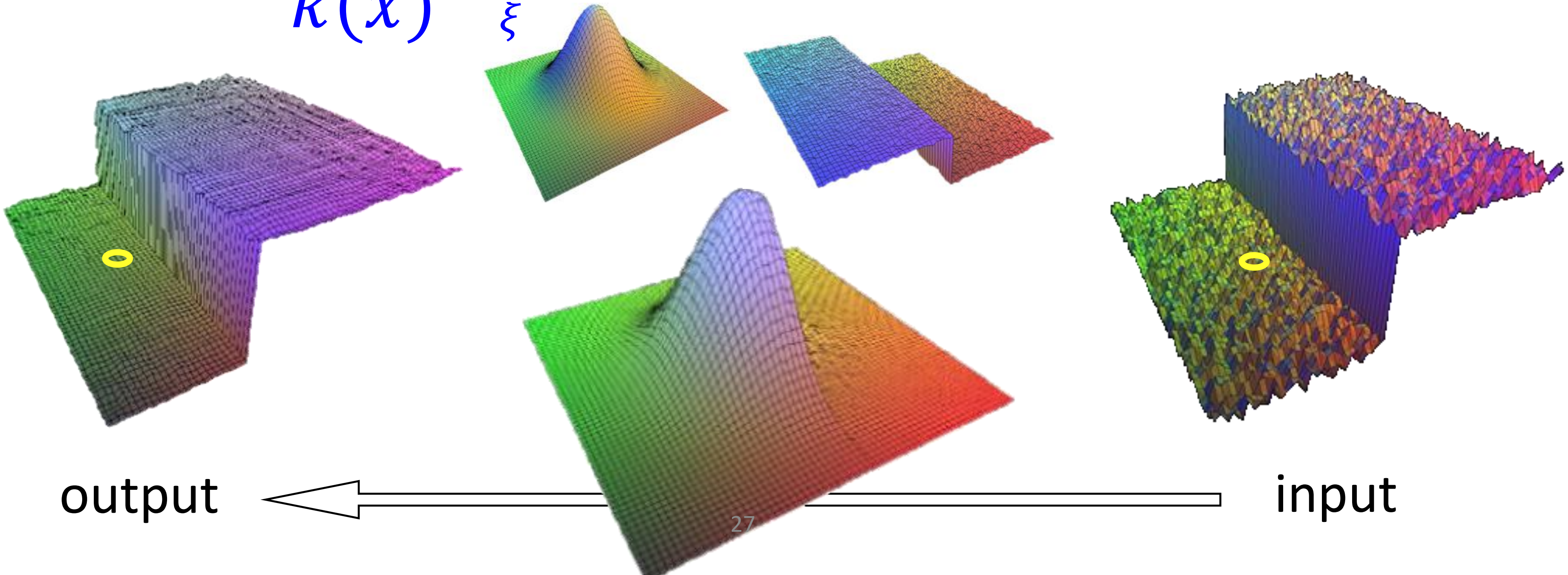


Bilateral Filter is Spatially-Variant

[Tomasi and Manduchi 1998]

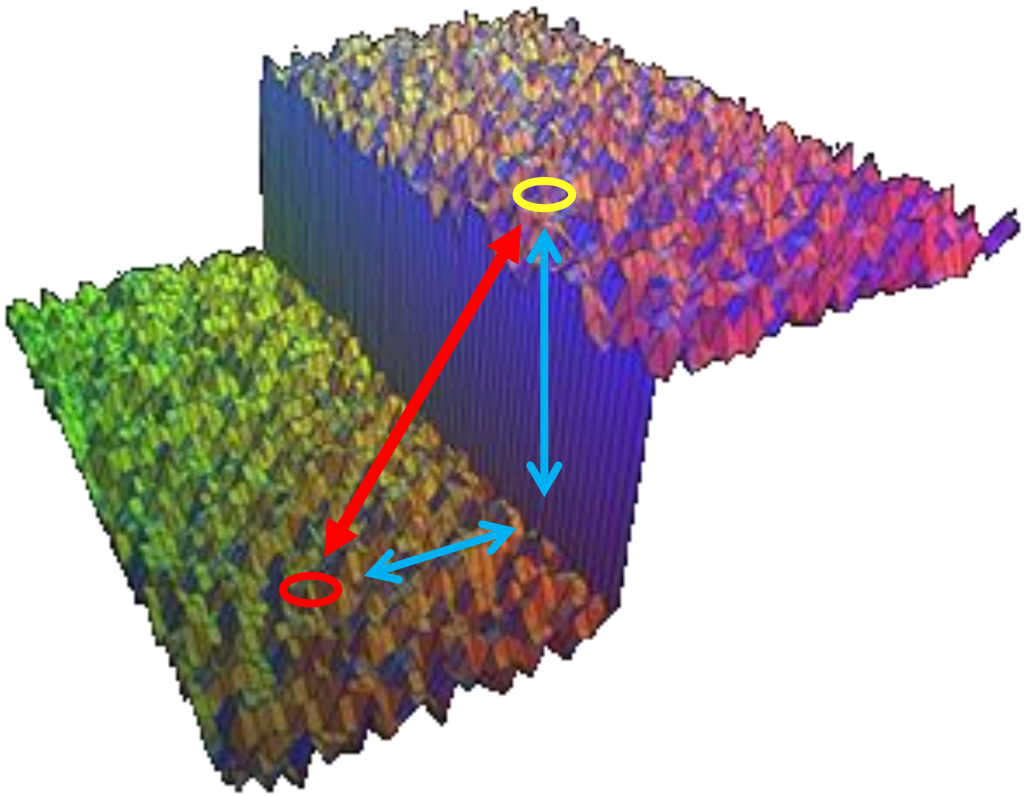
- The weights are different for each output pixel

$$J(x) = \frac{1}{k(x)} \sum_{\xi} f(x, \xi)\, g(I(\xi) - I(x)) \quad I(\xi)$$



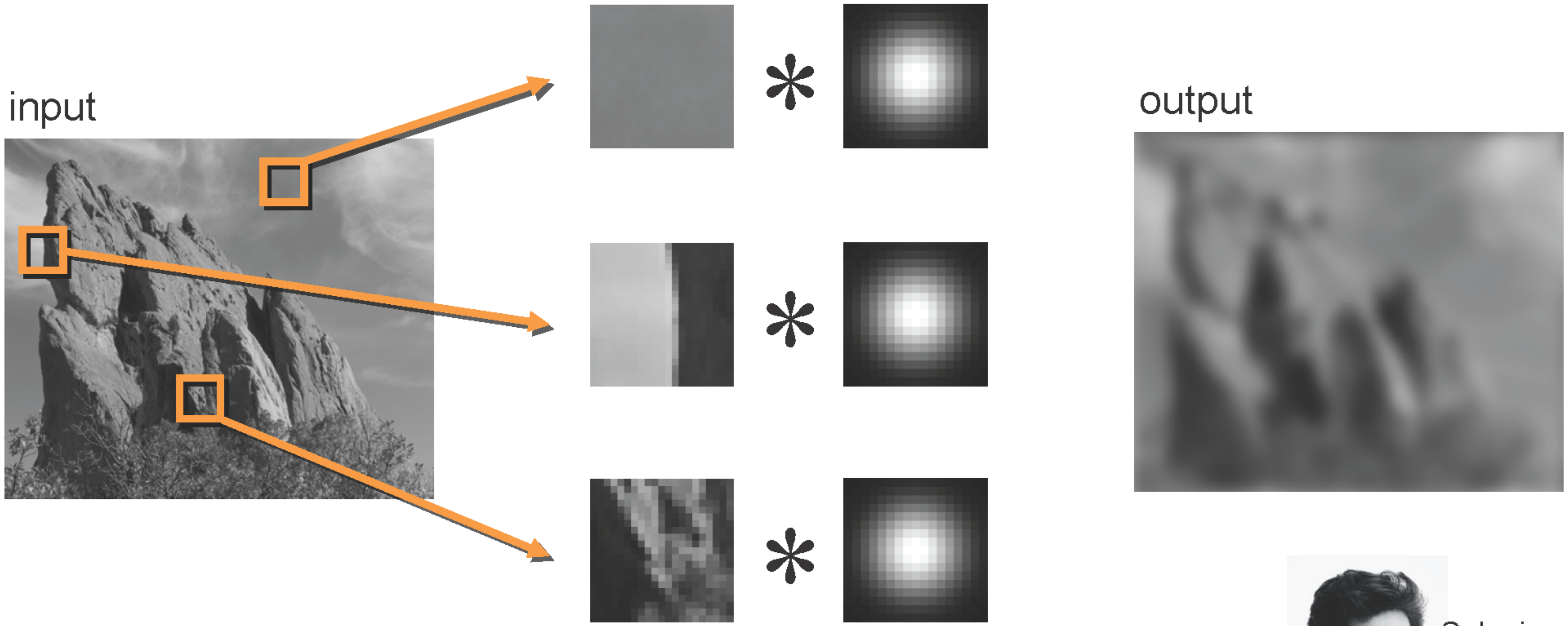
Other Explanation

- The bilateral filter uses the 3D distance to compute the weight



Bilateral Filter vs Gaussian Filter

- Bilateral Filter – fix to the Gaussian filter
- Gaussian filtering applies the same filter everywhere

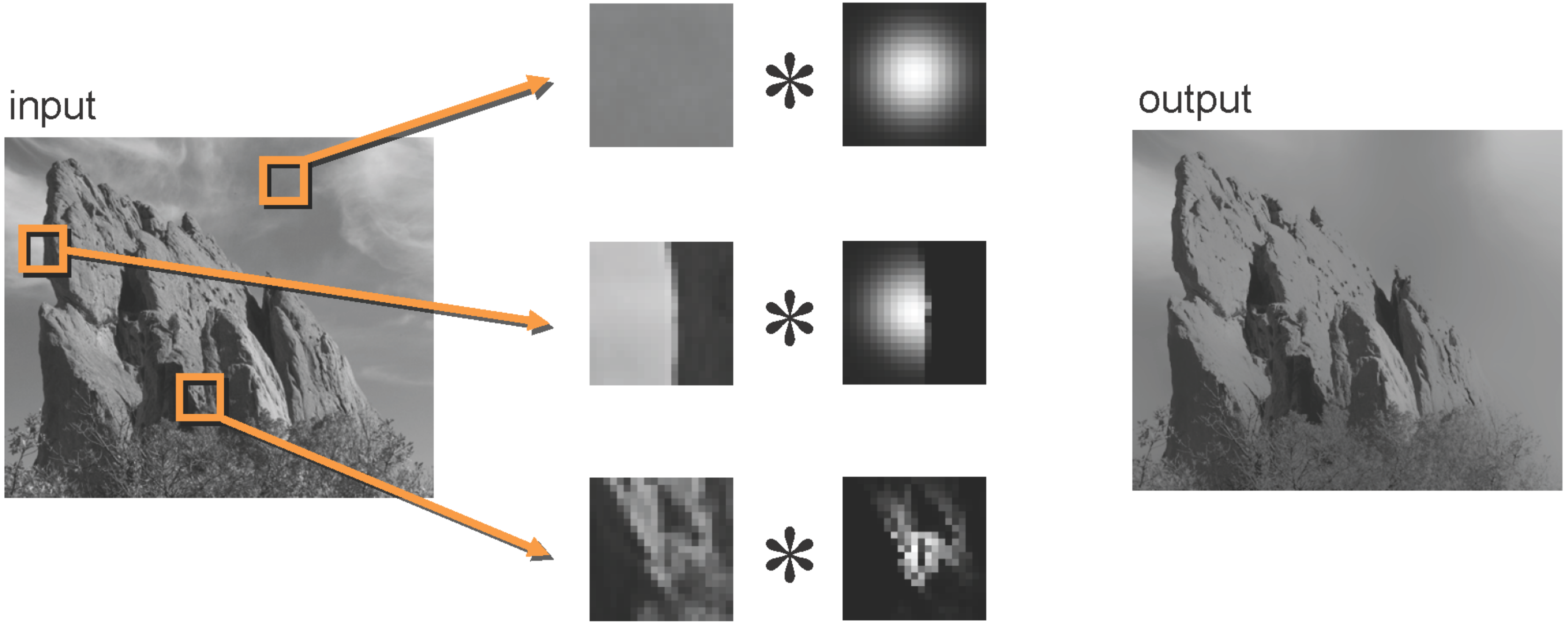


Same Gaussian kernel everywhere.

[notes adapted from Sylvain Paris SIGGRAPH'07 course notes on Bilateral Filter]

Bilateral Filter vs Gaussian Filter

- Adjust kernel based on image content



The kernel shape depends on the image content.

Results: Denoise

noisy image

naïve denoising
Gaussian blur

better denoising
edge-preserving filter

Smoothing an image without blurring its edges.

More Examples

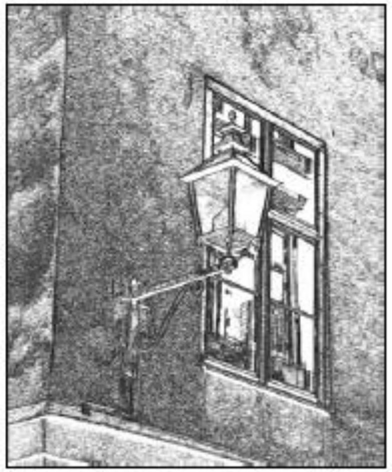
(a) *Photograph*

(b) *Edge-aware smoothing*

(c) *Detail enhancement*

(d) *Stylization*

(e) *Recoloring*



(f) *Pencil drawing*

(g) *Depth-of-field*

[Gastal and Oliveira 2011]

Questions?

Guided Image Filtering

Kaiming He[1], Jian Sun[2], and Xiaoou Tang[1,3]

[1] Department of Information Engineering, The Chinese University of Hong Kong
[2] Microsoft Research Asia
[3] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China

ECCV 2010

Formulation

- Given an input image P, the output image Q is computed as,

$$Q_i = \sum_j W_{ij}(I) P_j$$

- The weights are computed from the guide image I,

$$W_{ij}(I) = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right)$$

the total number of such windows containing both i and j

the k-th local window containing both i and j

μ_k, σ_k are the mean and variance of pixel values in the window ω_k

Spatially-Variant Kernals

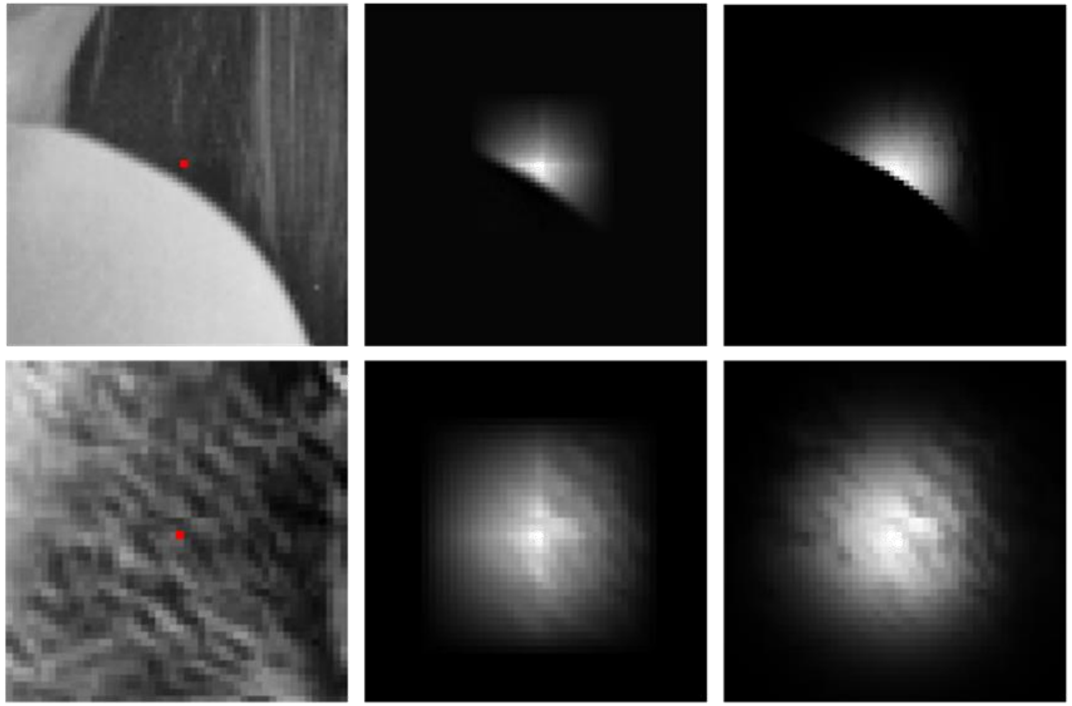


Fig. 3. Filter kernels. Top: a step edge (guided filter: $r = 7, \epsilon = 0.1^2$, bilateral filter: $\sigma_s = 7, \sigma_r = 0.1$). Bottom: a textured patch (guided filter: $r = 8, \epsilon = 0.2^2$, bilateral filter: $\sigma_s = 8, \sigma_r = 0.2$). The kernels are centered at the pixels denote by the red dots.

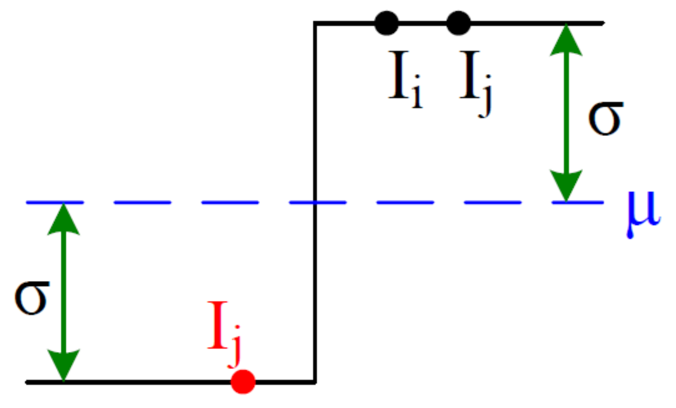
Explanation

- When i and j are from different sides of an edge,

$$\frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2} \sim \ -1$$

- When i and j are from the same side,

$$\frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2} \sim \ 1$$



$$W_{ij}(I) = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right)$$

Better Than Bilateral Filter

- $\mathrm{O}(N)$ time, N is the number of pixels
  - Bilateral filter is $O(Nr^2)$, r is the local window size

$$Q_i = \sum_j W_{ij}(I) P_j$$

$$W_{ij}(I) = \frac{1}{|\omega|^2} \sum_{k:(i,j)\in\omega_k} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right)$$

μ_k, σ_k can be computed with box filter ($\mathrm{O}(N)$).