University of Zurich[UZH] | ETH zürich
Institute of Informatics – Institute of Neuroinformatics
ROBOTICS & PERCEPTION GROUP

Lecture 04
Image Filtering

Davide Scaramuzza
http://rpg.ifi.uzh.ch

No exercise this afternoon

19.09.2019	**Lecture 01 - Introduction to Computer Vision and Visual Odometry**	Davide Scaramuzza
26.09.2019	**Lecture 02 - Image Formation 1: perspective projection and camera models** **Exercise 01 - Augmented reality wireframe cube**	Davide Scaramuzza Daniel & Mathias Gehrig
03.10.2019	**Lecture 03 - Image Formation 2: camera calibration algorithms** **Exercise 02 - PnP problem**	Davide Scaramuzza Daniel & Mathias Gehrig
10.10.2019	**Lecture 04 - Filtering & Edge detection**	Davide Scaramuzza
17.10.2019	**Lecture 05 - Point Feature Detectors, Part 1** **Exercise 03 - Harris detector + descriptor + matching**	Davide Scaramuzza Daniel & Mathias Gehrig
24.10.2019	**Lecture 06 - Point Feature Detectors, Part 2** **Exercise 04 - SIFT detector + descriptor + matching**	Davide Scaramuzza Daniel & Mathias Gehrig
31.10.2019	**Lecture 07 - Multiple-view geometry** **Exercise 05 - Stereo vision: rectification, epipolar matching, disparity, triangulation**	Davide Scaramuzza Daniel & Mathias Gehrig
07.11.2019	**Lecture 08 - Multiple-view geometry 2** **Exercise 06 - Eight-Point Algorithm**	Antonio Loquercio Daniel & Mathias Gehrig
14.11.2019	**Lecture 09 - Multiple-view geometry 3 (Part 1)**	Antonio Loquercio
21.11.2019	**Lecture 10 - Multiple-view geometry 3 (Part 2)** **Exercise session: Intermediate VO Integration**	Davide Scaramuzza Daniel & Mathias Gehrig
28.11.2019	**Lecture 11 - Optical Flow and Tracking (Lucas-Kanade)** **Exercise 08 - Lucas-Kanade tracker**	Davide Scaramuzza Daniel & Mathias Gehrig
05.12.2019	**Lecture 12 - Place recognition and 3D Reconstruction** **Exercise session: Deep Learning Tutorial**	Davide Scaramuzza Daniel & Mathias Gehrig
12.12.2019	**Lecture 13 - Visual inertial fusion** **Exercise 09 - Bundle Adjustment**	Davide Scaramuzza Daniel & Mathias Gehrig
19.12.2019	**Lecture 14 - Event based vision** After the lecture, we will Scaramuzza's lab. Departure from lecture room at 12:00 via tram 10. **Exercise session: Final VO Integration**	Davide Scaramuzza Daniel & Mathias Gehrig

Image filtering

- The word *filter* comes from frequency-domain processing, where “filtering” refers to the process of accepting or rejecting certain frequency components
- We distinguish between low-pass and high-pass filtering
  - A **low-pass filter** smooths an image (retains low-frequency components)
  - A **high-pass filter** retains the contours (also called edges) of an image (high frequency)

Low-pass filtered image

High-pass filtered image

Low-pass filtering

Low-pass filtering
Motivation: noise reduction

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian distribution

Original

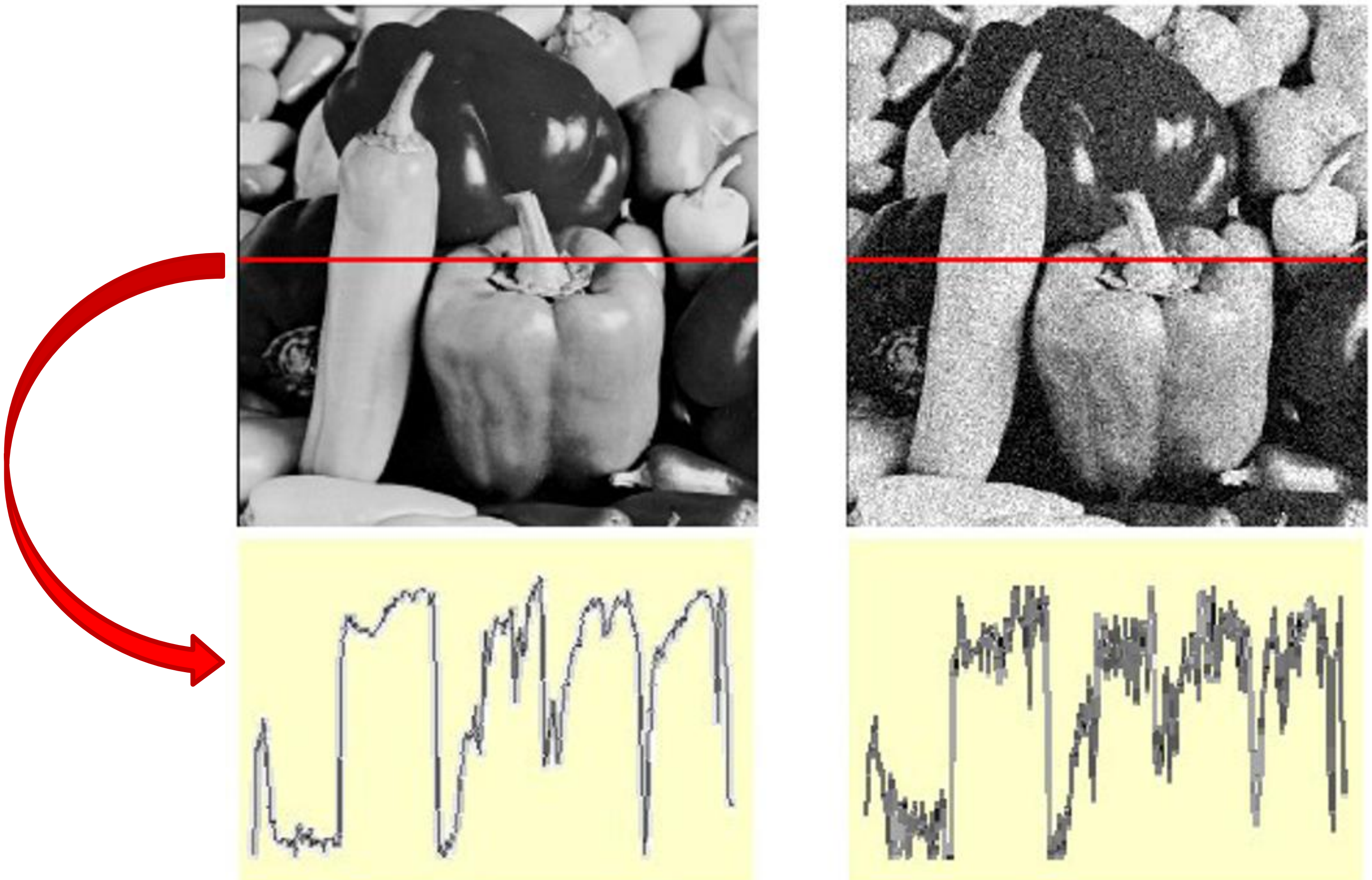
Salt and pepper noise

Impulse noise

Gaussian noise

Source: S. Seitz

Gaussian noise



	Ideal image		Noise process	Gaussian i.i.d. (white) noise
$I(x,y) =$	$\hat{I}(x,y)$	$+$	$\eta(x,y)$	$\eta(x,y) \sim \mathrm{N}(\mu,\sigma)$

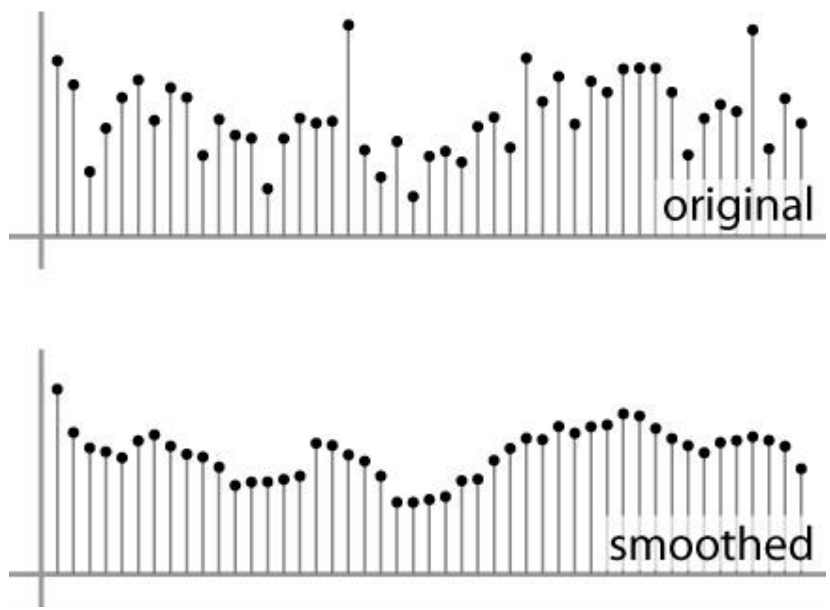
How could we reduce the noise to try to recover the “ideal image”?

Moving average

- Replaces each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise process to be independent from pixel to pixel

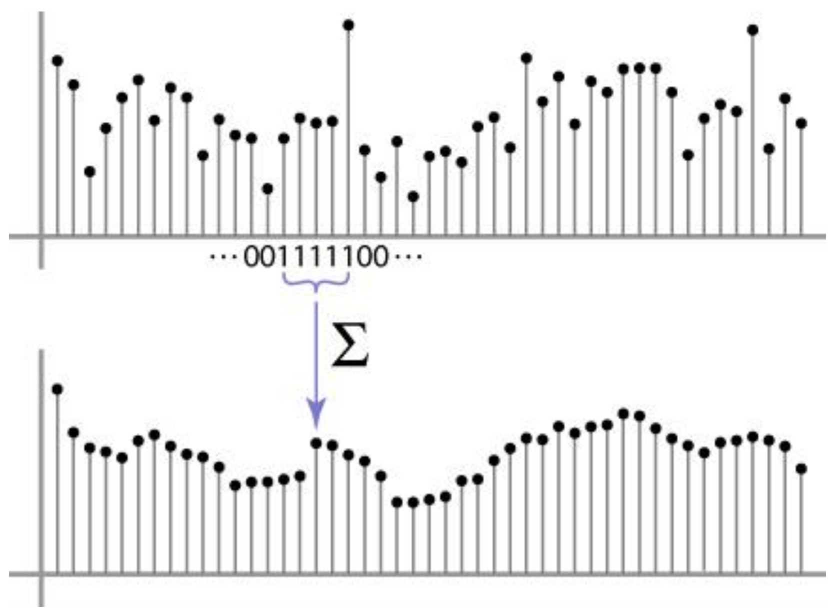
Moving average

- Replaces each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



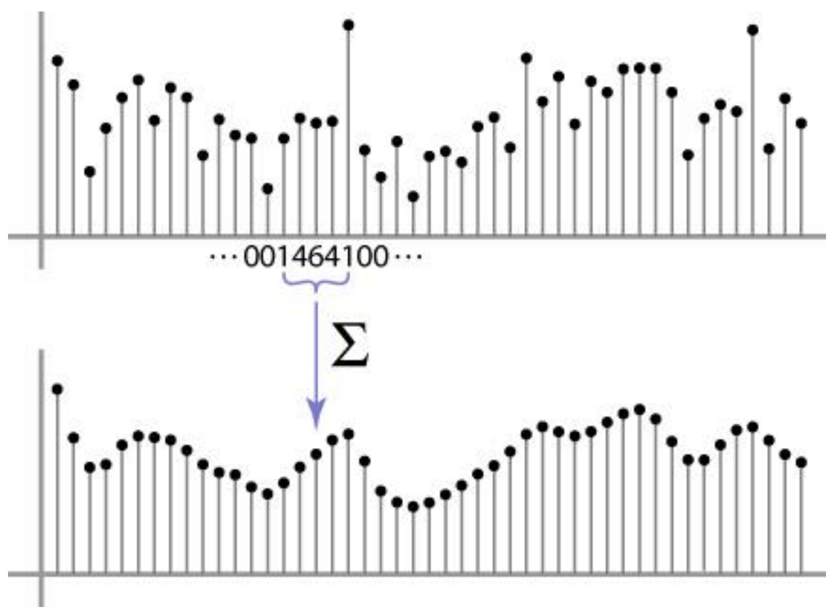
Weighted Moving Average

- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Weighted Moving Average

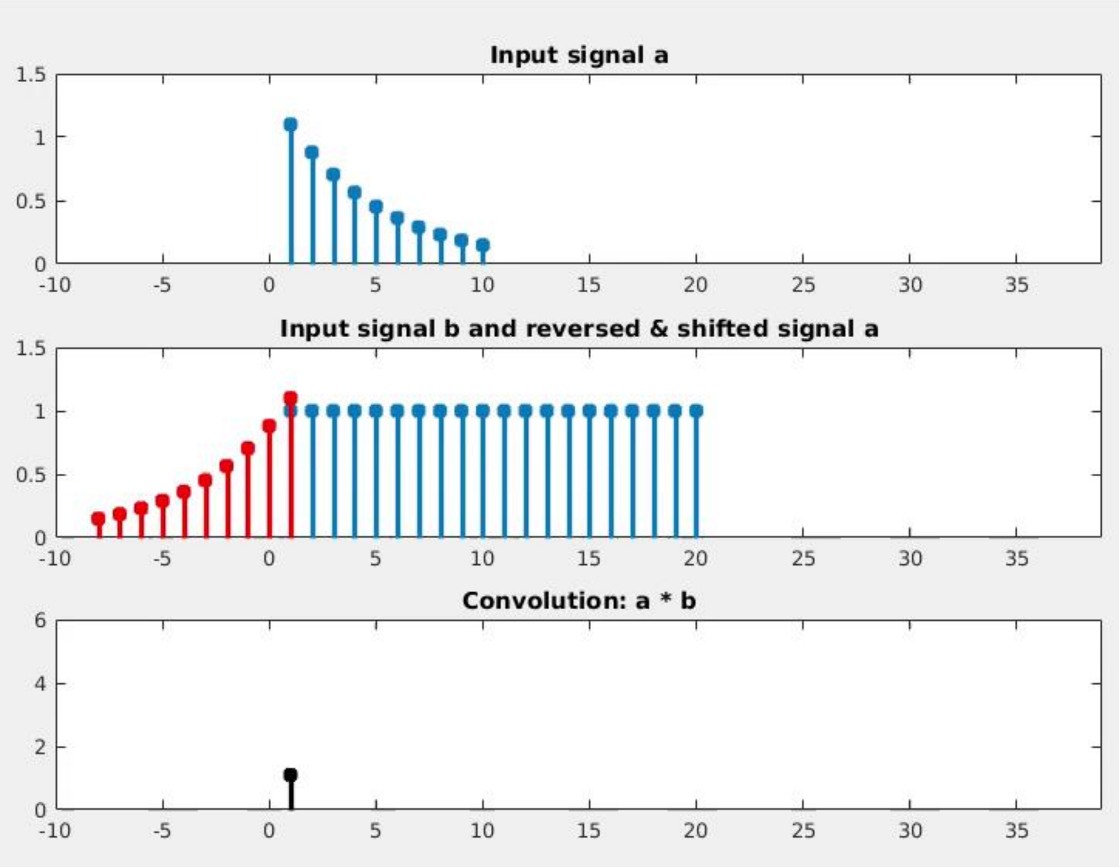
- Non-uniform weights [1, 4, 6, 4, 1] / 16



This operation is called *convolution*

Example of convolution of two sequences (or "signals")

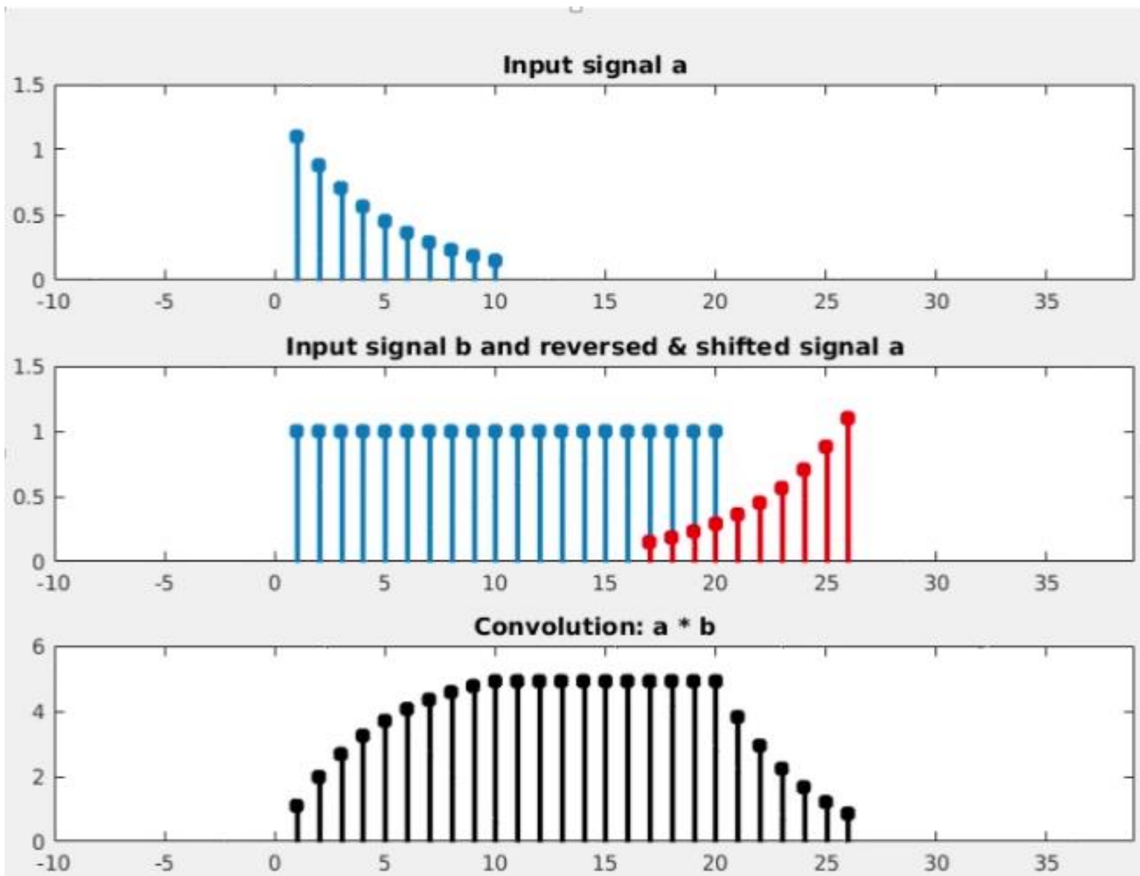
- One of the sequences is flipped (right to left) before sliding over the other
- Notation: $a * b$
- Nice properties: linearity, associativity, commutativity, etc.



This operation is called *convolution*

Example of convolution of two sequences (or "signals")

- One of the sequences is flipped (right to left) before sliding over the other
- Notation: $a * b$
- Nice properties: linearity, associativity, commutativity, etc.

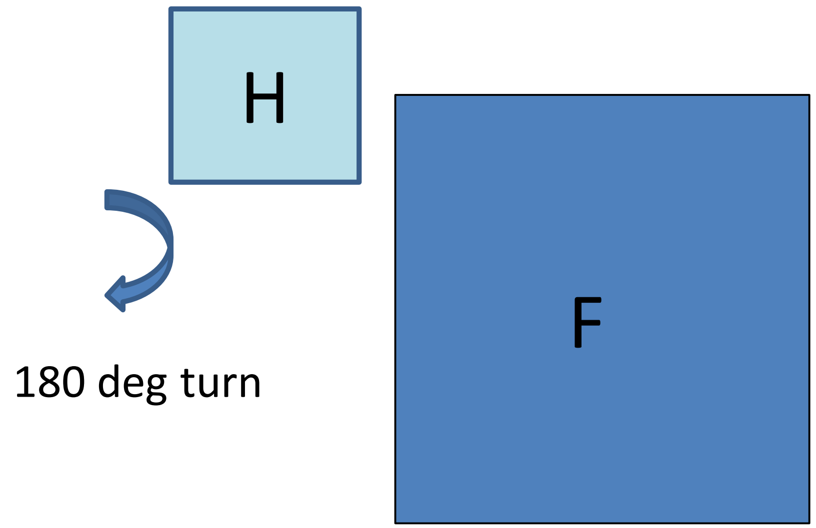


2D Filtering

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left) (=180 deg turn)
  - Then slide the filter over the image and compute sum of products

$$G[x,y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[x-u, y-v]H[u,v]$$

$$G = F * H$$



Filtering an image: replace each pixel with a linear combination of its neighbors.

The **filter $\boldsymbol{H}$** is also called "**kernel**" or "**mask**".

Review: Convolution vs. Cross-correlation

Convolution $G = F * H$

$$G[x,y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[x-u, y-v]H[u,v]$$

Properties: linearity, associativity, commutativity

Cross-correlation $G = F \otimes H$

$$G[x,y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[x+u, y+v]H[u,v]$$

Properties: linearity, but not associativity and commutativity

For a Gaussian or box filter, will the output of convolution and correlation be different?

Example: Moving Average in 2D

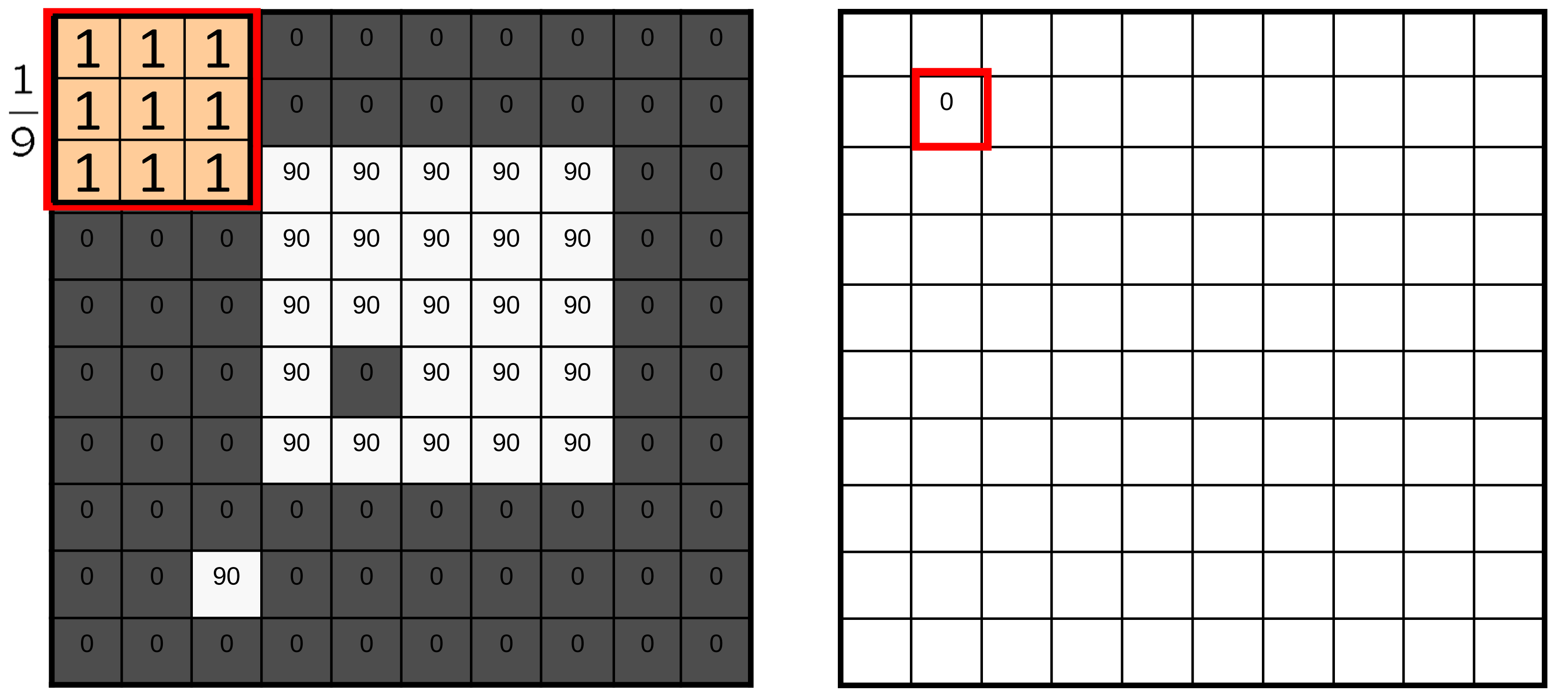
Input image

$F[x, y]$

Filtered image

$G[x, y]$

“box filter”



Example: Moving Average in 2D

Input image

$F[x, y]$

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

Filtered image

$G[x, y]$

	0	10							

Example: Moving Average in 2D

Input image

$$F[x, y]$$

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

Filtered image

$$G[x, y]$$

	0	10	20						

Example: Moving Average in 2D

Input image

$$F[x, y]$$

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

Filtered image

$$G[x, y]$$

	0	10	20	30					

Example: Moving Average in 2D

Input image

$F[x, y]$

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

Filtered image

$G[x, y]$

	0	10	20	30	30				

Example: Moving Average in 2D

Input image

$F[x, y]$

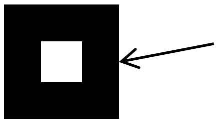
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Filtered image

$G[x, y]$

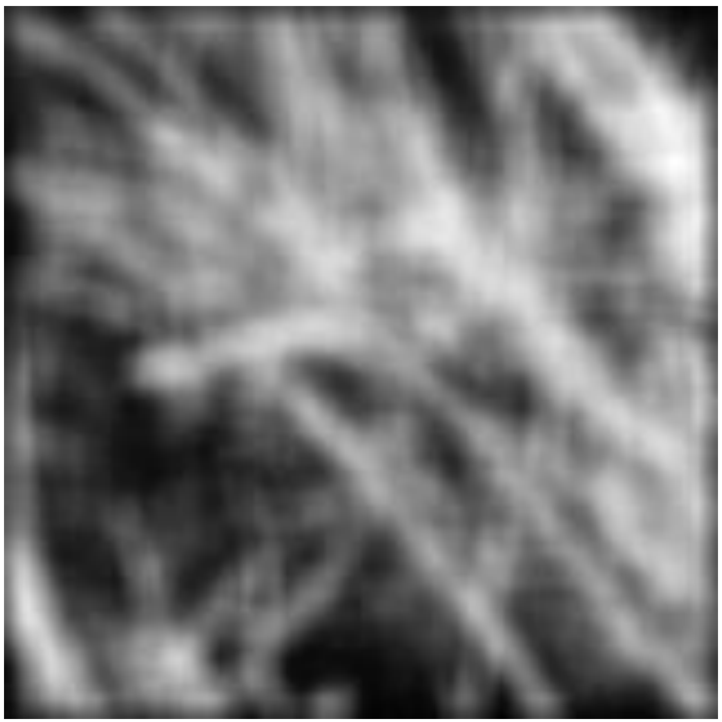
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Example: Moving Average in 2D



Box filter:
white = high value, black = zero value

original



filtered

Gaussian filter

- What if we want the closest pixels to have higher 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

$F[x, y]$

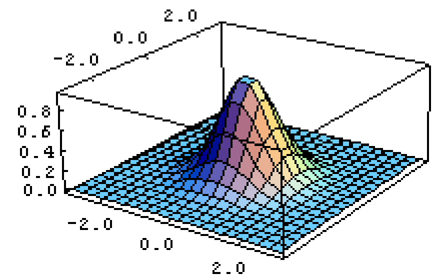
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

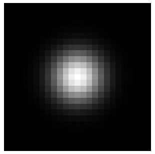
$H[u, v]$

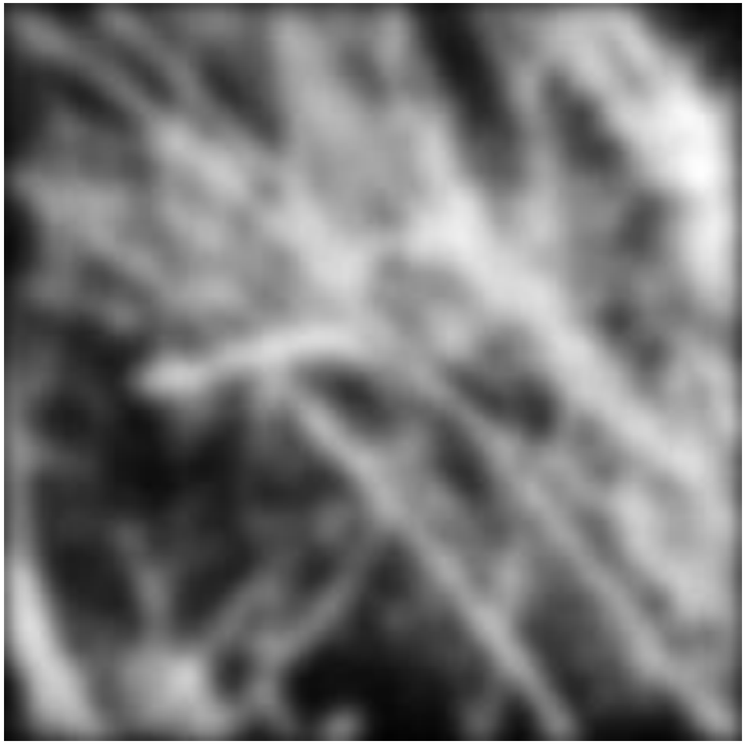
This kernel is the approximation of a Gaussian function:

$$H[u, v] \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

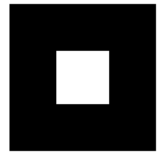


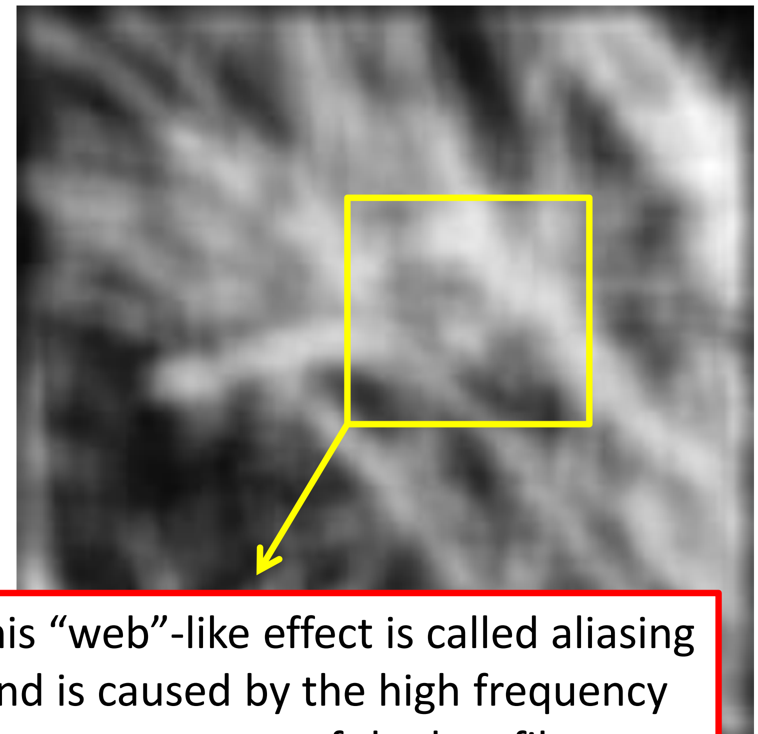
Smoothing with a Gaussian





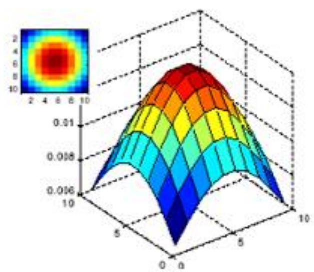
Compare the result with a box filter



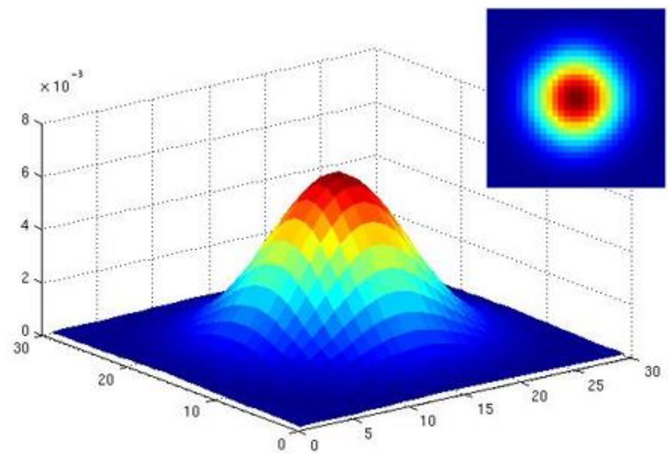


Gaussian filters

- What parameters matter?
- **Size** of the kernel
  - NB: a Gaussian function has **infinite support**, but discrete filters use finite kernels



σ = 5 pixels
with 10 × 10 pixel kernel

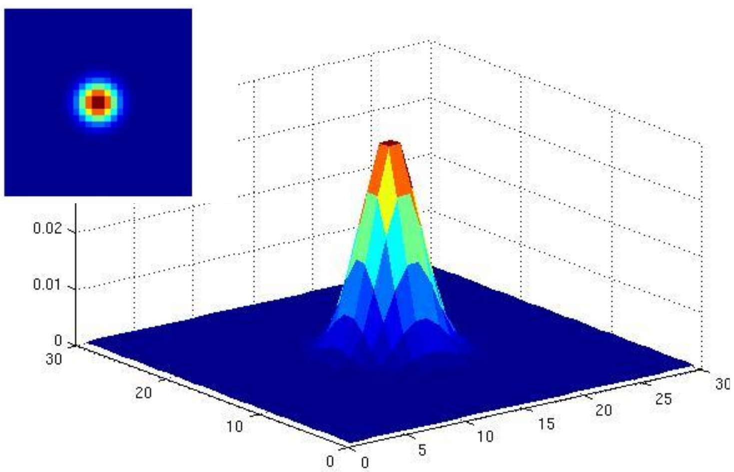


σ = 5 pixels
with 30 × 30 pixel kernel

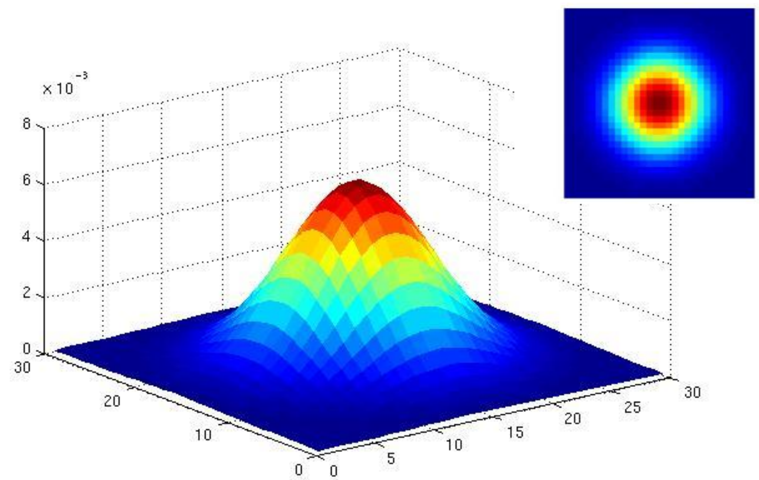
Which one approximates better the ideal Gaussian filter, the left or the right one?

Gaussian filters

- What parameters matter?
- **Variance** of Gaussian: control the amount of smoothing



σ = 2 pixels
with 30×30 pixel kernel

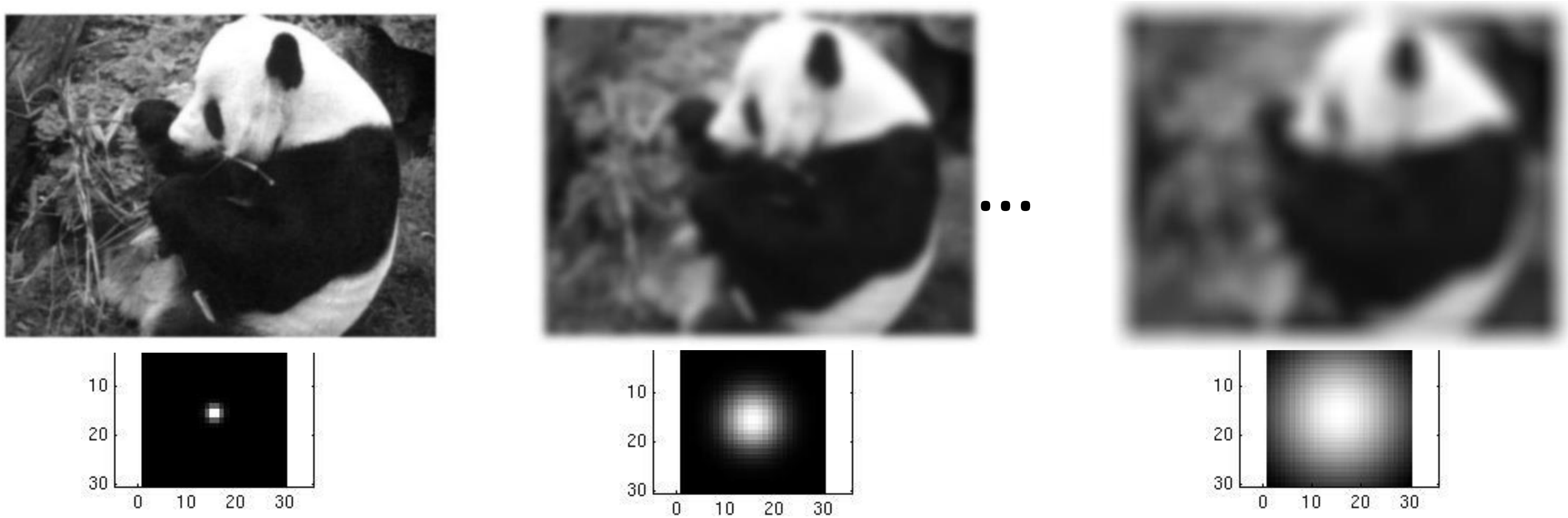


σ = 5 pixels
with 30×30 pixel kernel

Recall: standard deviation = σ [pixels], variance = σ^2 [pixels^2]

Smoothing with a Gaussian

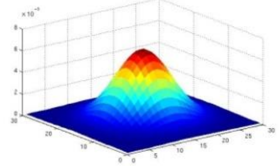
σ is called “scale” of the Gaussian kernel, and controls the amount of smoothing.



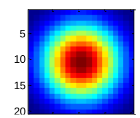
Sample Matlab code

```
>> hsize = 20;
>> sigma = 5;
>> h = fspecial('gaussian', hsize, sigma);
```

```
>> mesh(h);
```



```
>> imagesc(h);
```




```
>> im = imread('panda.jpg');
>> outim = imfilter(im, h);
>> imshow(outim);
```

im

outim

Boundary issues

- What about near the image edges?
  - the filter window falls off the edges of the image
  - need to pad the image borders
  - methods:
    - zero padding (black)
    - wrap around
    - copy edge
    - reflect across edge

Summary on (linear) smoothing filters

- Smoothing filter
  - has positive values (also called coefficients)
  - **sums to 1** → preserve brightness of constant regions
  - removes "high-frequency" components; "low-pass" filter

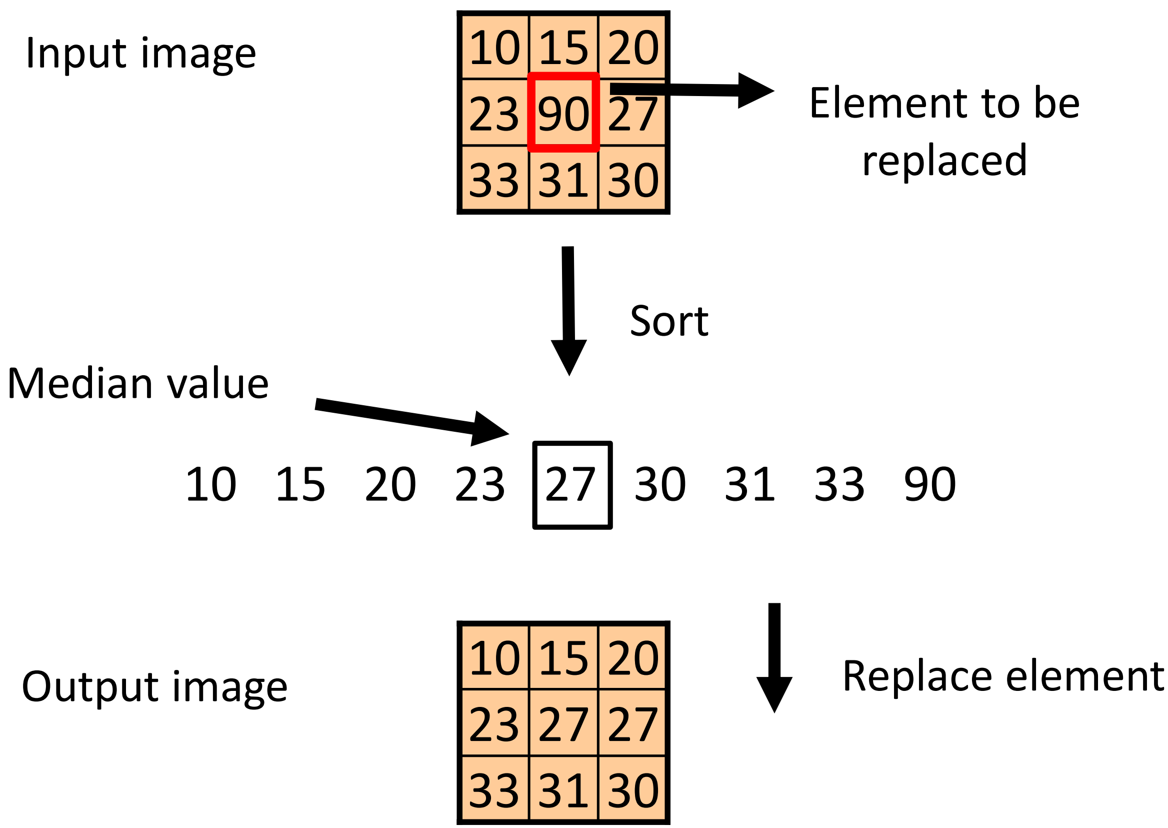
Non-linear filtering

Effect of smoothing filters

Linear smoothing filters do not alleviate salt and pepper noise!

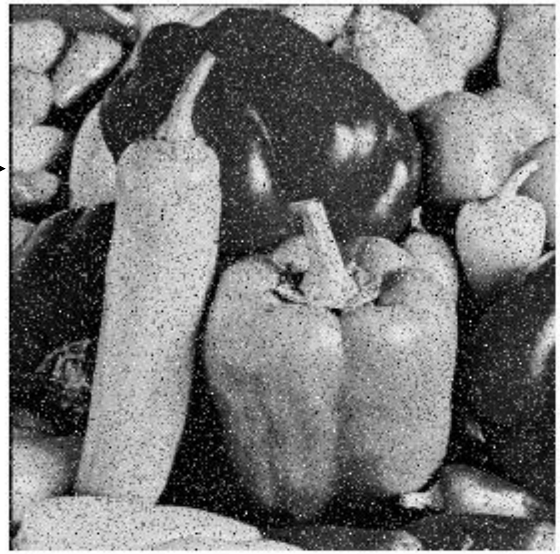
Median filter

- It is a non-linear filter
- Removes spikes: good for “*impulse noise*” and “*salt & pepper noise*”

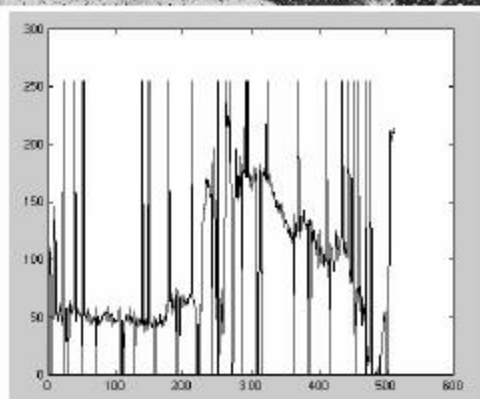


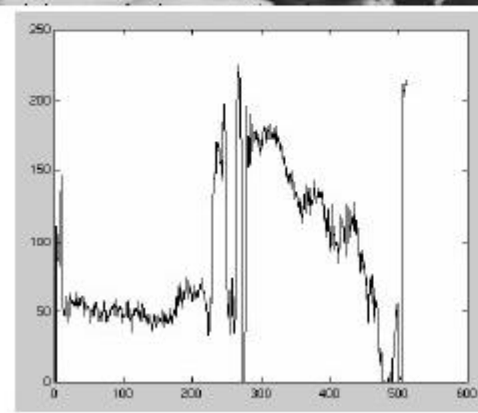
Median filter

Salt and pepper noise



Median filtered

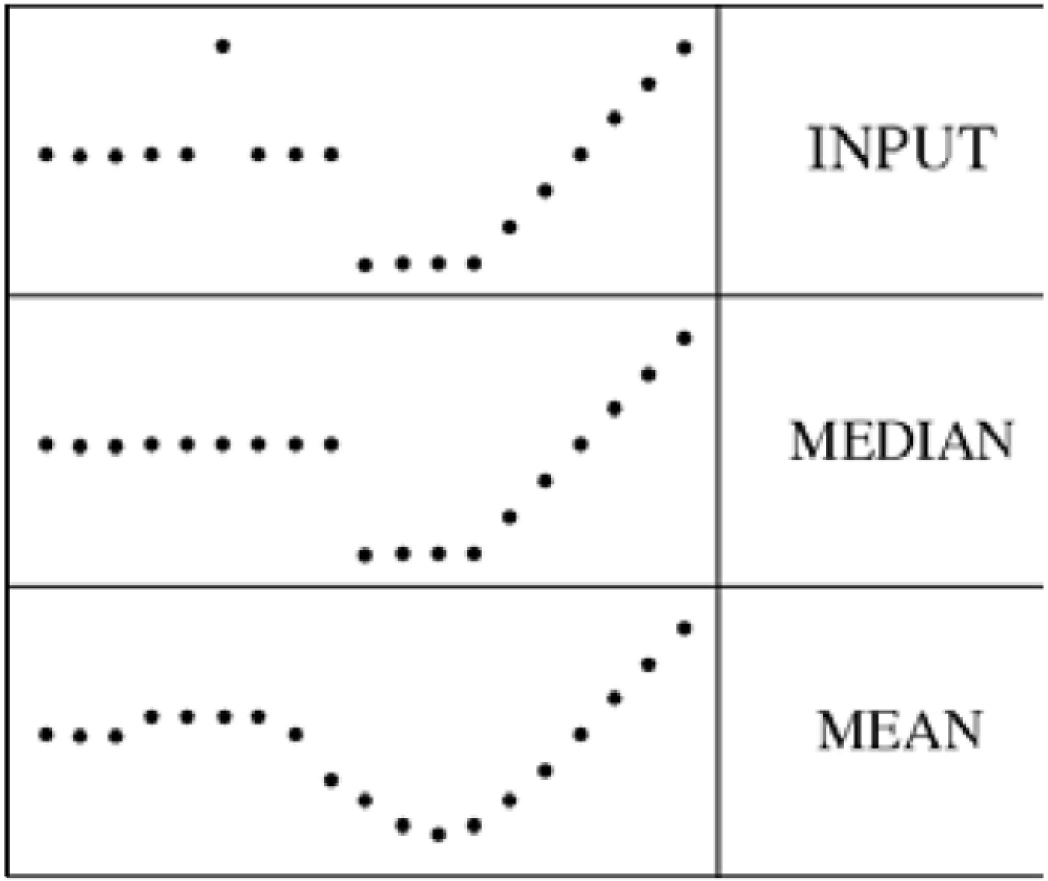




Plots of a row of the image

Median filter

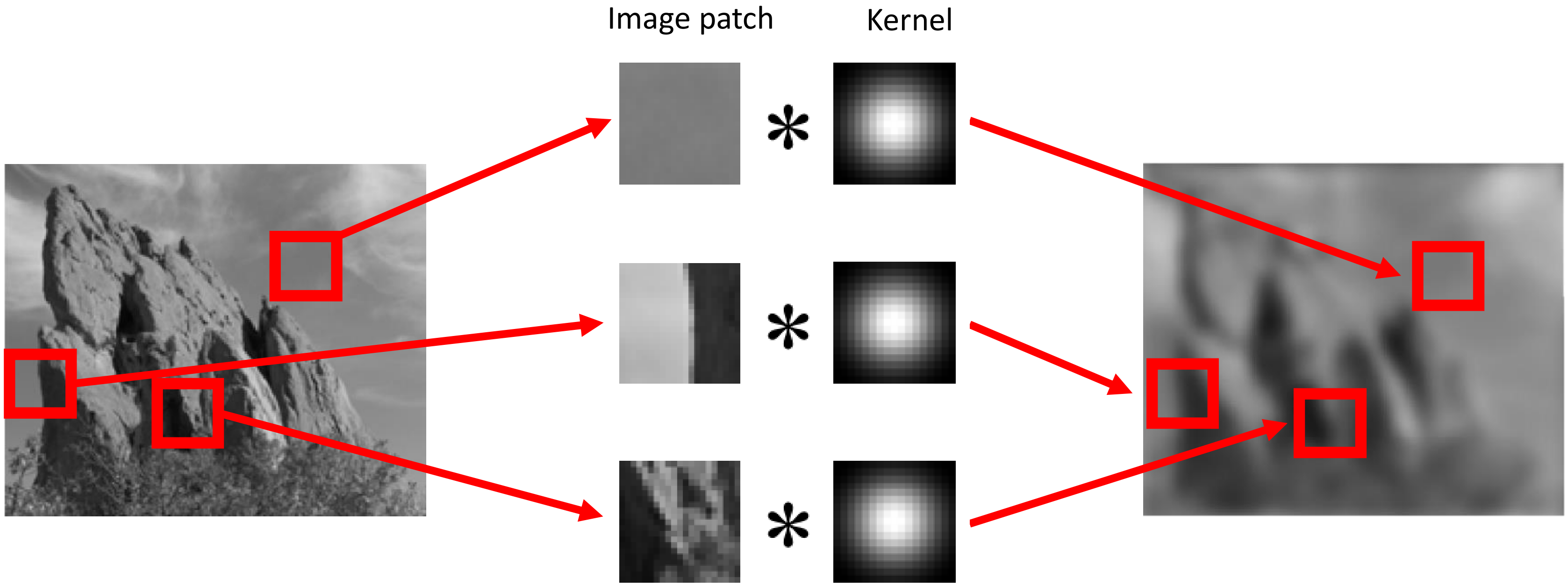
- Median filter preserves strong edges,



… but it removes small edges.

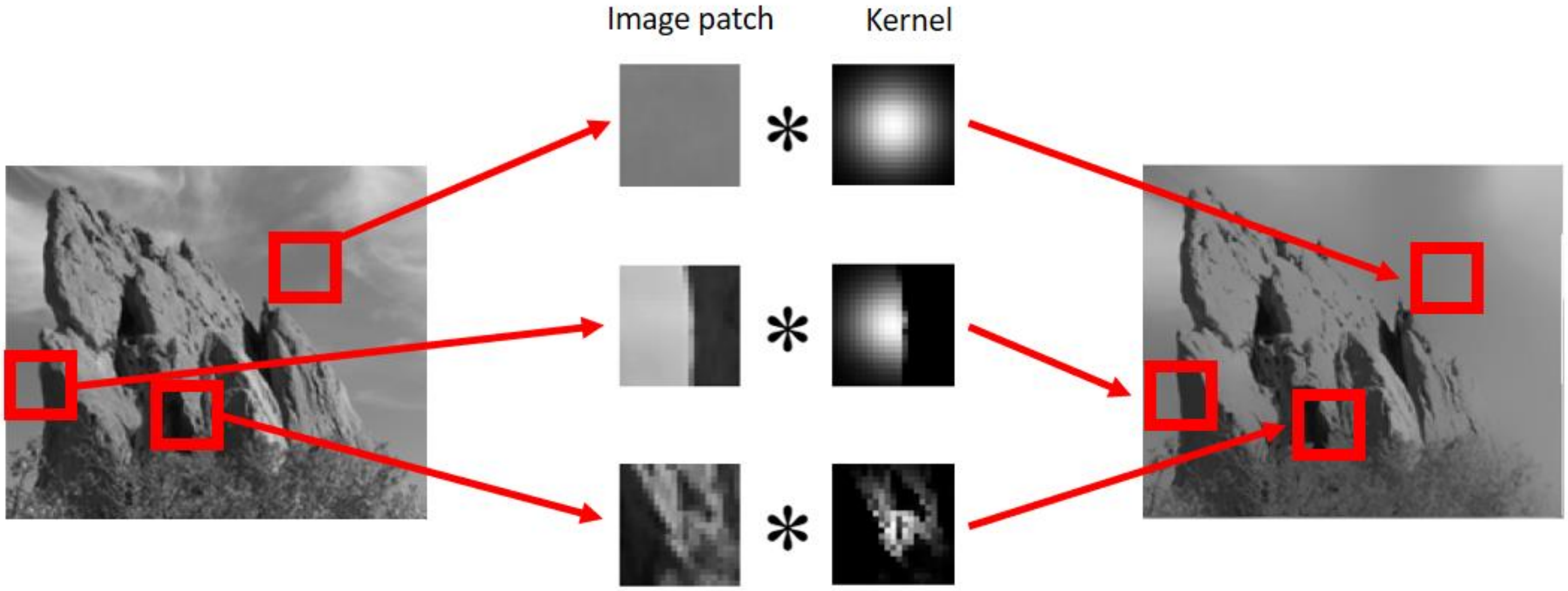
Bilateral filter

- **Gaussian filters do not preserve strong egdes (discontinuites)**. This is because they apply the same kernel everywhere.
- **Median filters** do preserve strong edges but remove small (weak) edges.



Bilateral filter

- **Gaussian filters do not preserve strong egdes (discontinuites)**. This is because they apply the same kernel everywhere.
- **Median filters** do preserve strong edges but remove small (weak) edges.
- **Bilateral filters** solve this by adapting the kernel locally to the intensity profile, so they are patch-content dependent: they only average pixels with similar brightness. Basically, the weighted average discards the influence of pixels with different brightness (across the discontinuity), while all the pixels that are on the same size of the discontinuity are smoothed.

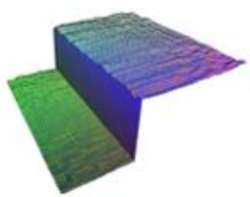


Bilateral filter

1-D Gaussian range component

spatial component

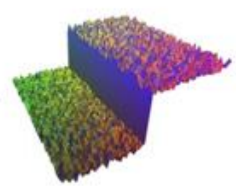
$$I'[x,y] = \frac{1}{W_p[x,y]} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I[x-u, y-v] G_{\sigma_r}(I[x-u,y-v] - I[x,y]) G_{\sigma_s}[u,v]$$



smoothed image

$=$ $\frac{1}{W_p[x,y]}$ $\cdot$

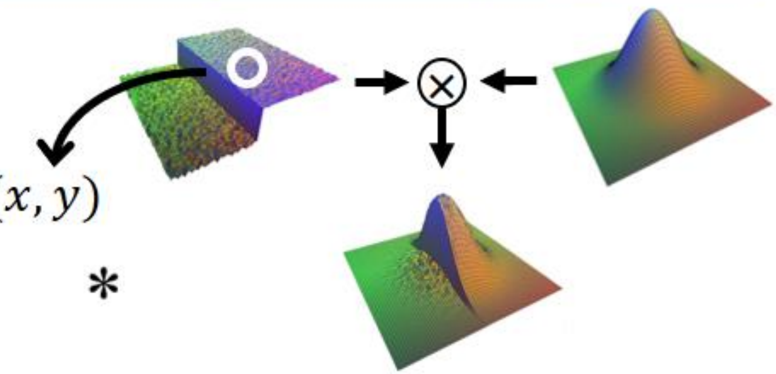
normalization factor



input image

(x, y)

$*$

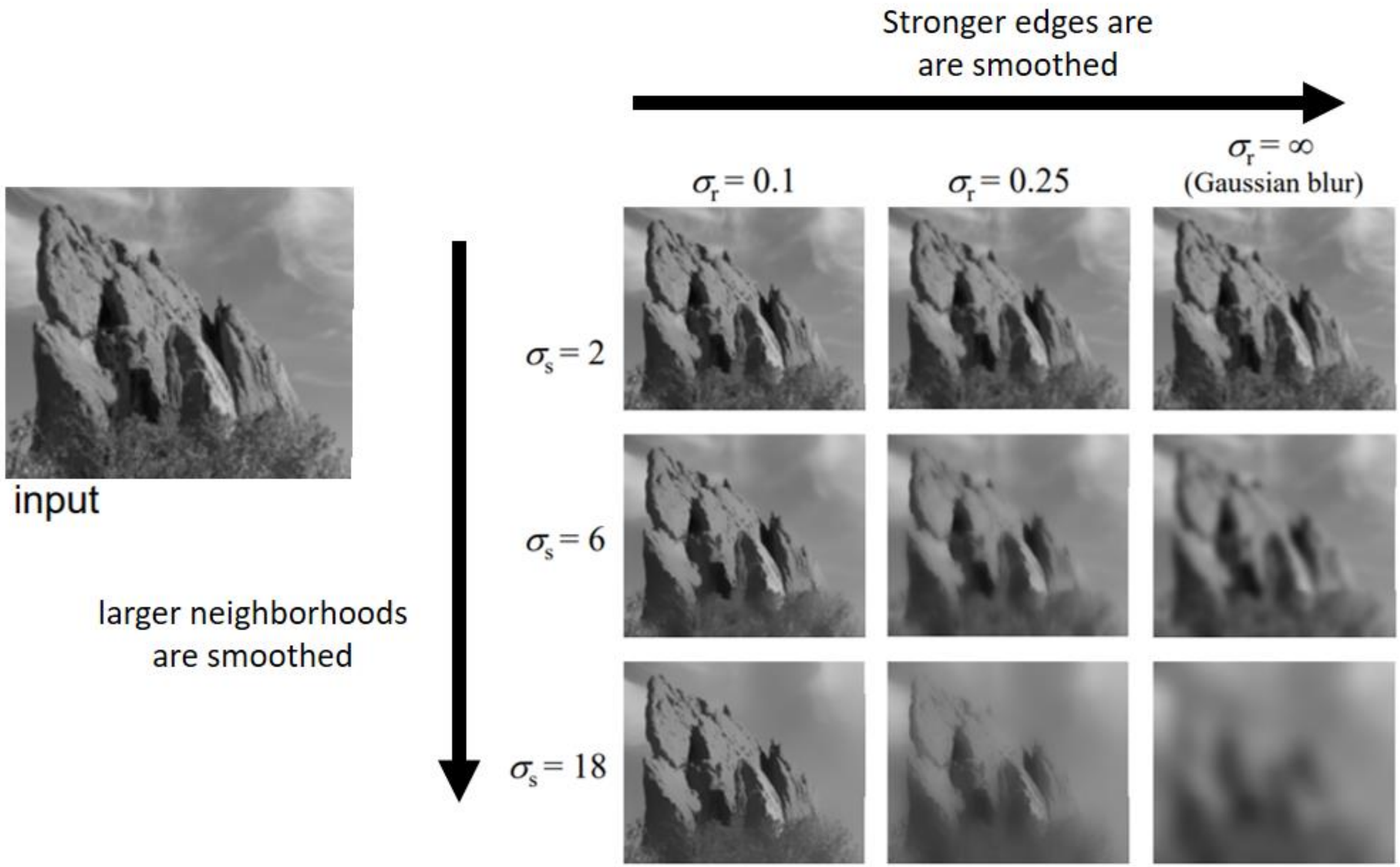


adaptive filter

$$W_p[x,y] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} G_{\sigma_r}(I[x-u,y-v] - I[x,y]) G_{\sigma_s}[u,v]$$

Normalization factor
(so that the filter values sum up to 1)

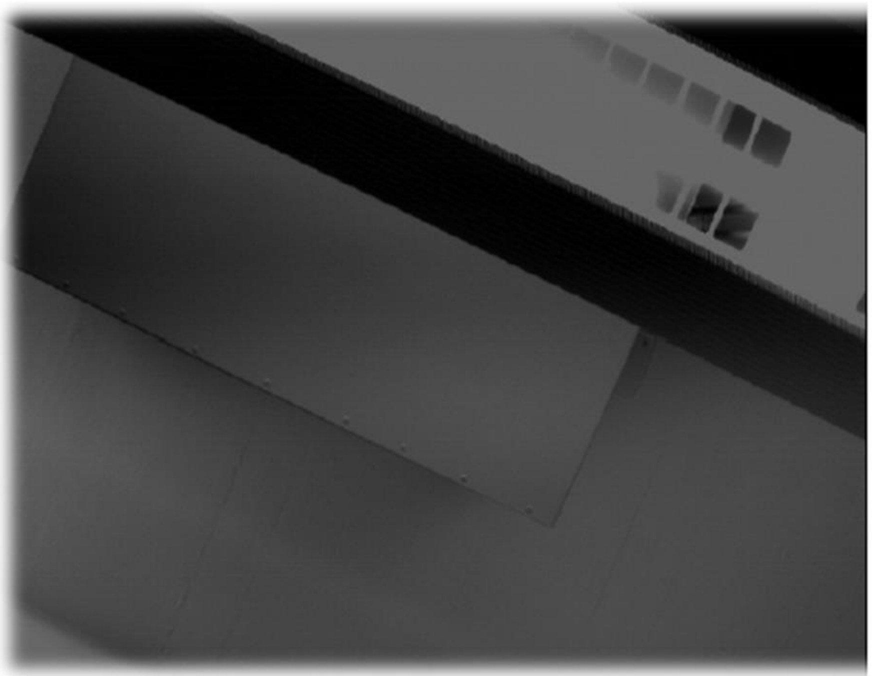
Bilateral filter

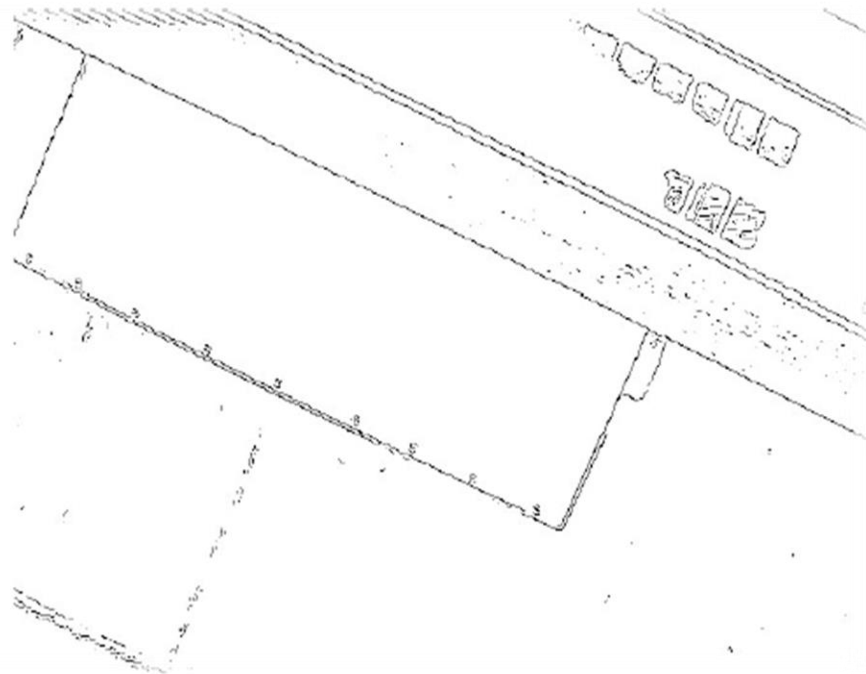


High-pass filtering
(edge detection)

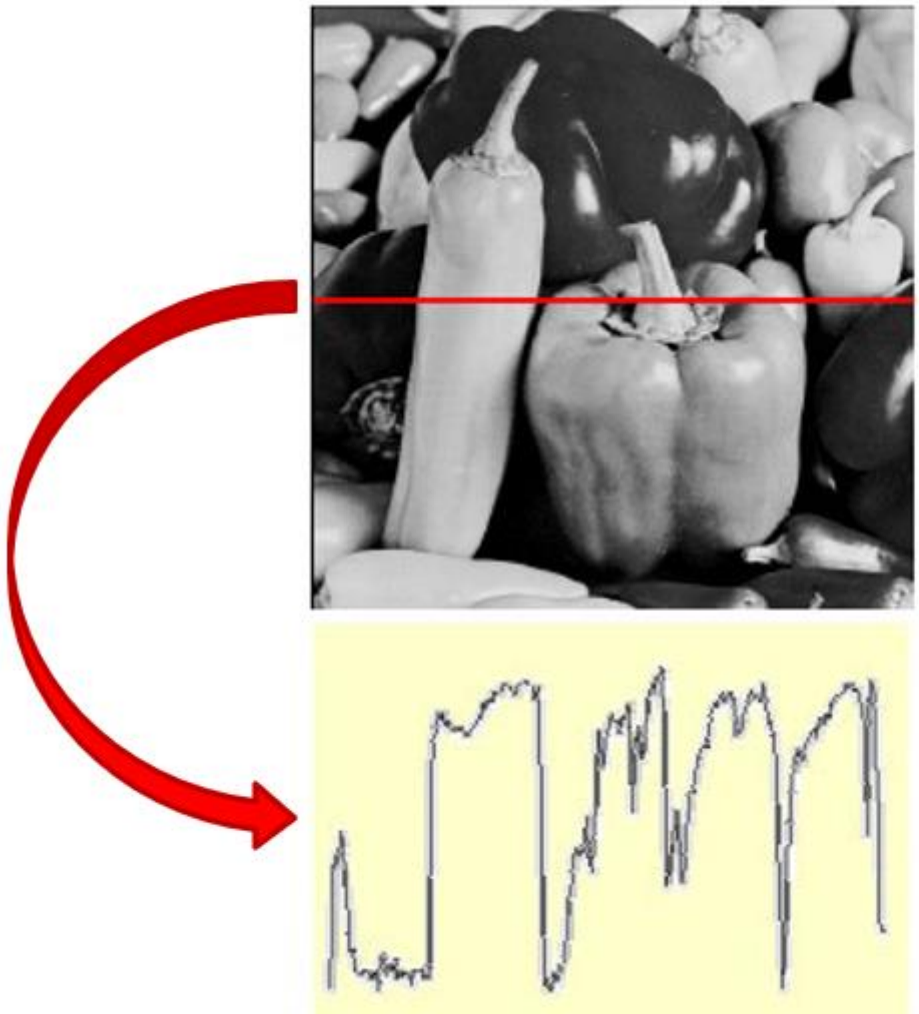
Edge detection

- Ultimate goal of edge detection: an idealized line drawing.
- Edge contours in the image correspond to important scene contours.

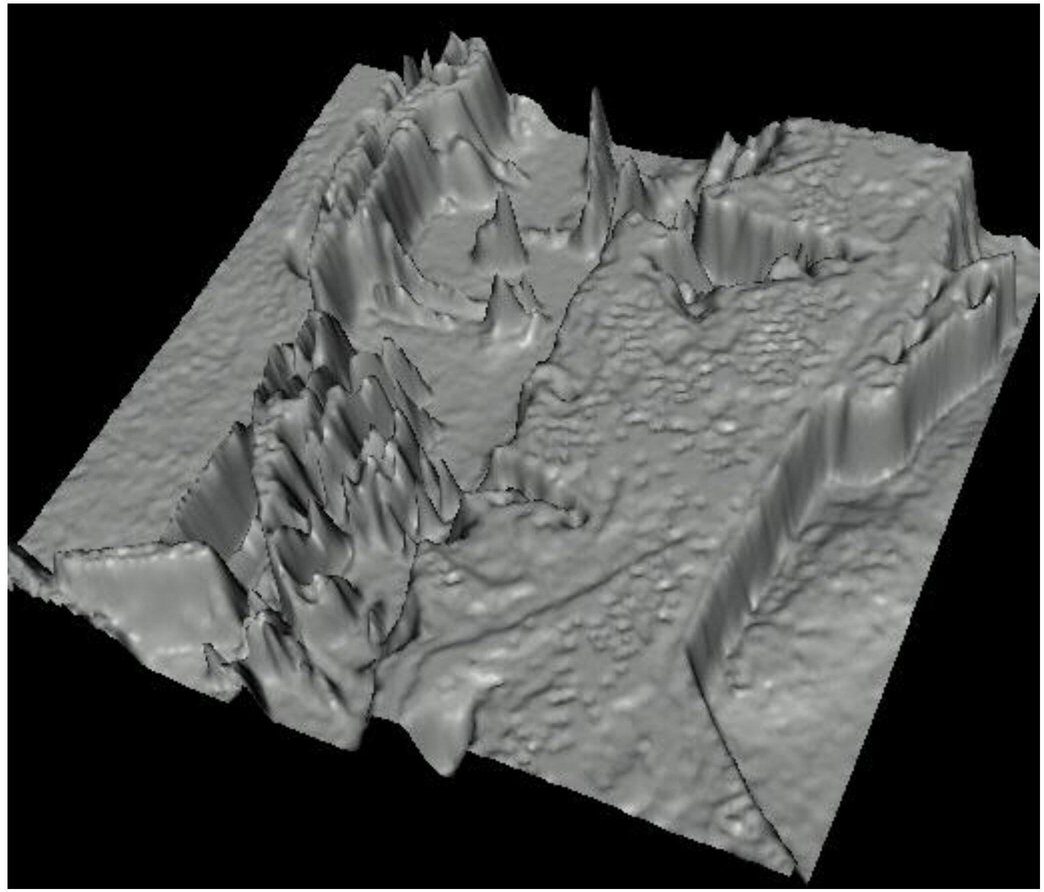




Edges are sharp intensity changes



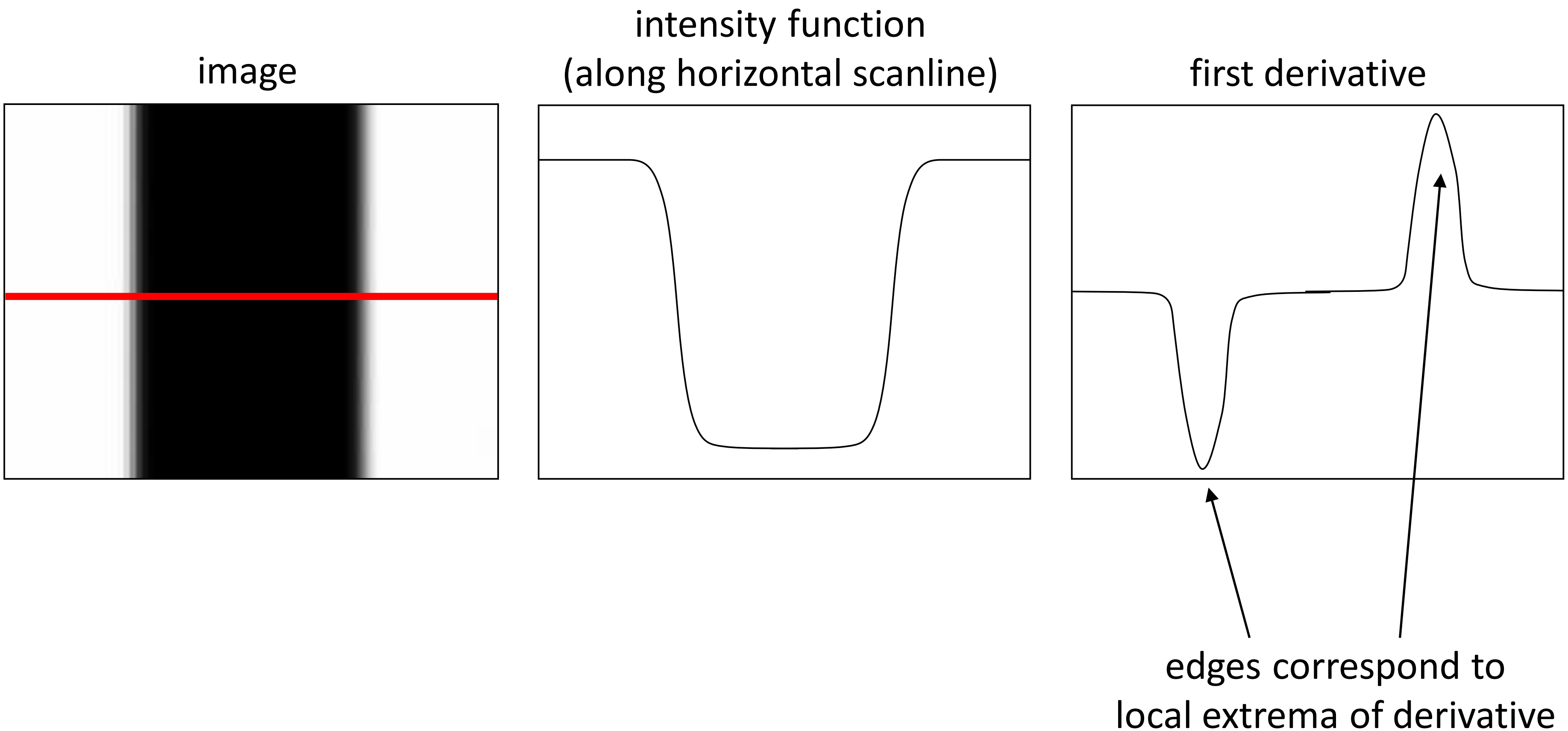
Images as functions $\mathrm{F}(x, y)$



Edges look like steep cliffs

Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Differentiation and convolution

For a 2D function $I(x, y)$ the partial derivative along x is:

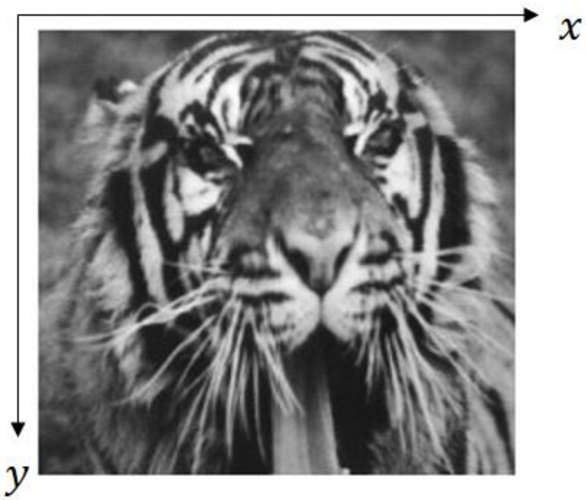
$$\frac{\partial I(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{I(x+\varepsilon, y) - I(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

$$\frac{\partial I(x,y)}{\partial x} \approx \frac{I(x+1, y) - I(x,y)}{1}$$

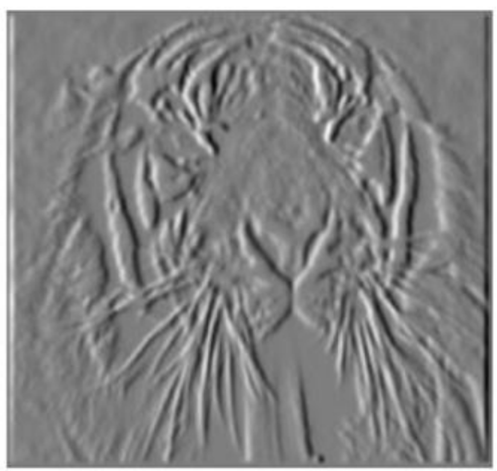
What would be the respective filters along x and y to implement the partial derivatives as a convolution?

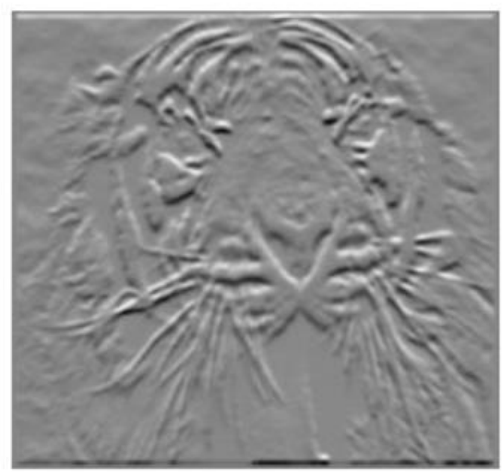
Partial derivatives of an image



$$\frac{\partial I(x,y)}{\partial x}$$

-1	1





$$\frac{\partial I(x,y)}{\partial y}$$

-1
1

Alternative Finite-difference filters

Prewitt filter $G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$ and $G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$

Sobel filter $G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$ and $G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$

```
Sample Matlab code
>> im = imread('lion.jpg');
>> h = fspecial('sobel');
>> outim = imfilter(double(im), h);
>> imagesc(outim);
>> colormap gray;
```

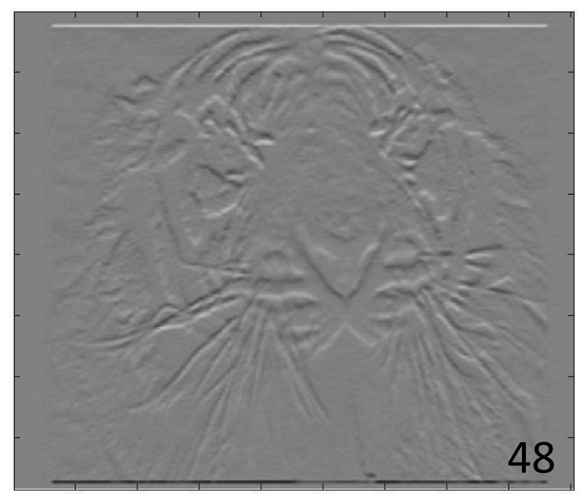
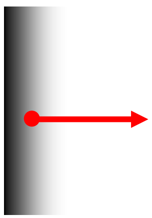


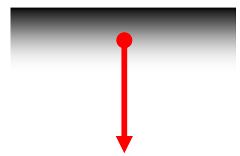
Image gradient

The gradient of an image: $$\nabla \mathrm{I} = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

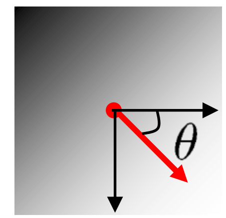
The gradient points in the direction of fastest intensity change



$$\nabla \mathrm{I} = \left[\frac{\partial I}{\partial x}, 0\right]$$



$$\nabla \mathrm{I} = \left[0, \frac{\partial I}{\partial y}\right]$$



$$\nabla \mathrm{I} = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

The gradient direction (orientation of edge normal) is given by:

$$\theta = atan2\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

The *edge strength* is given by the gradient magnitude

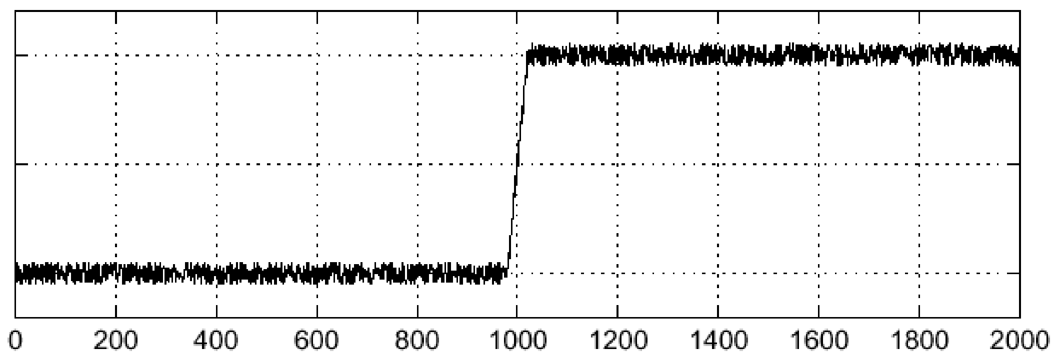
$$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

Effects of noise

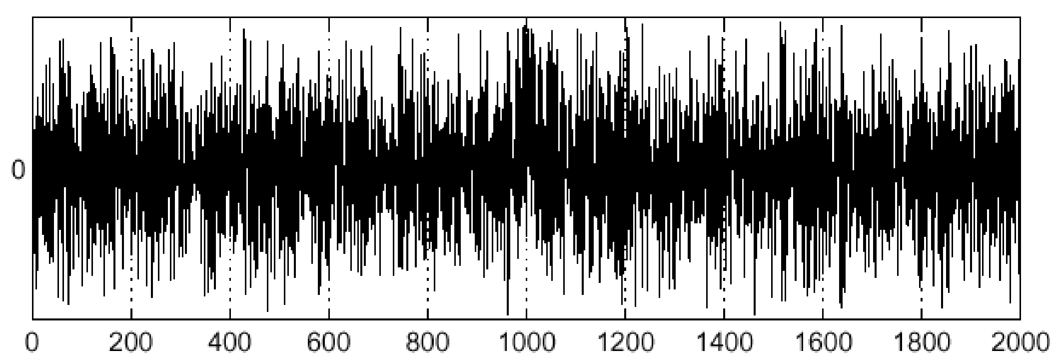
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$I(x)$



$\frac{\partial I(x)}{\partial x}$



Where is the edge?

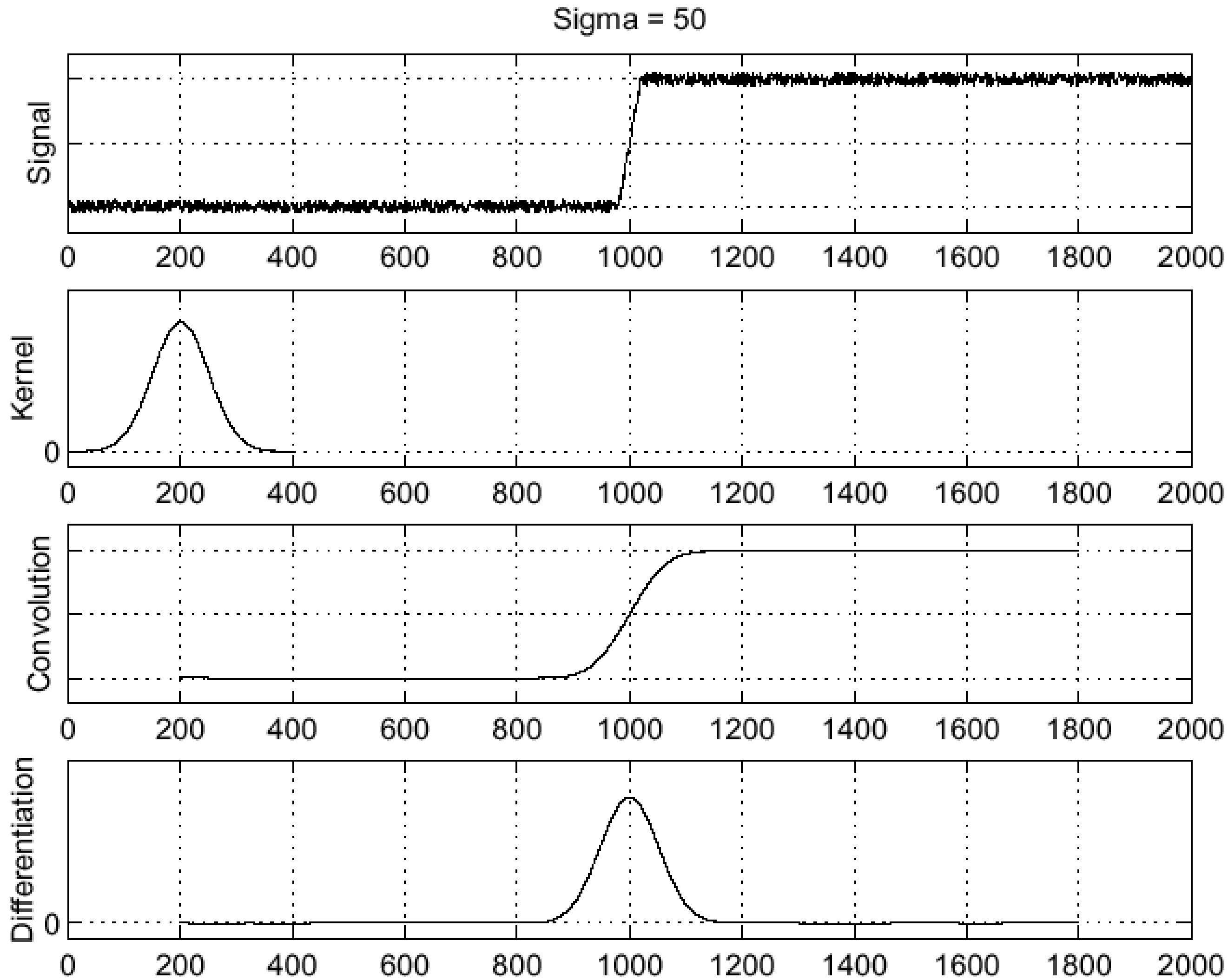
Solution: smooth first

I

H

$I * H$

$\frac{\partial}{\partial x}(I * H)$



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(I * H)$

Alternative: combine derivative and smoothing filter

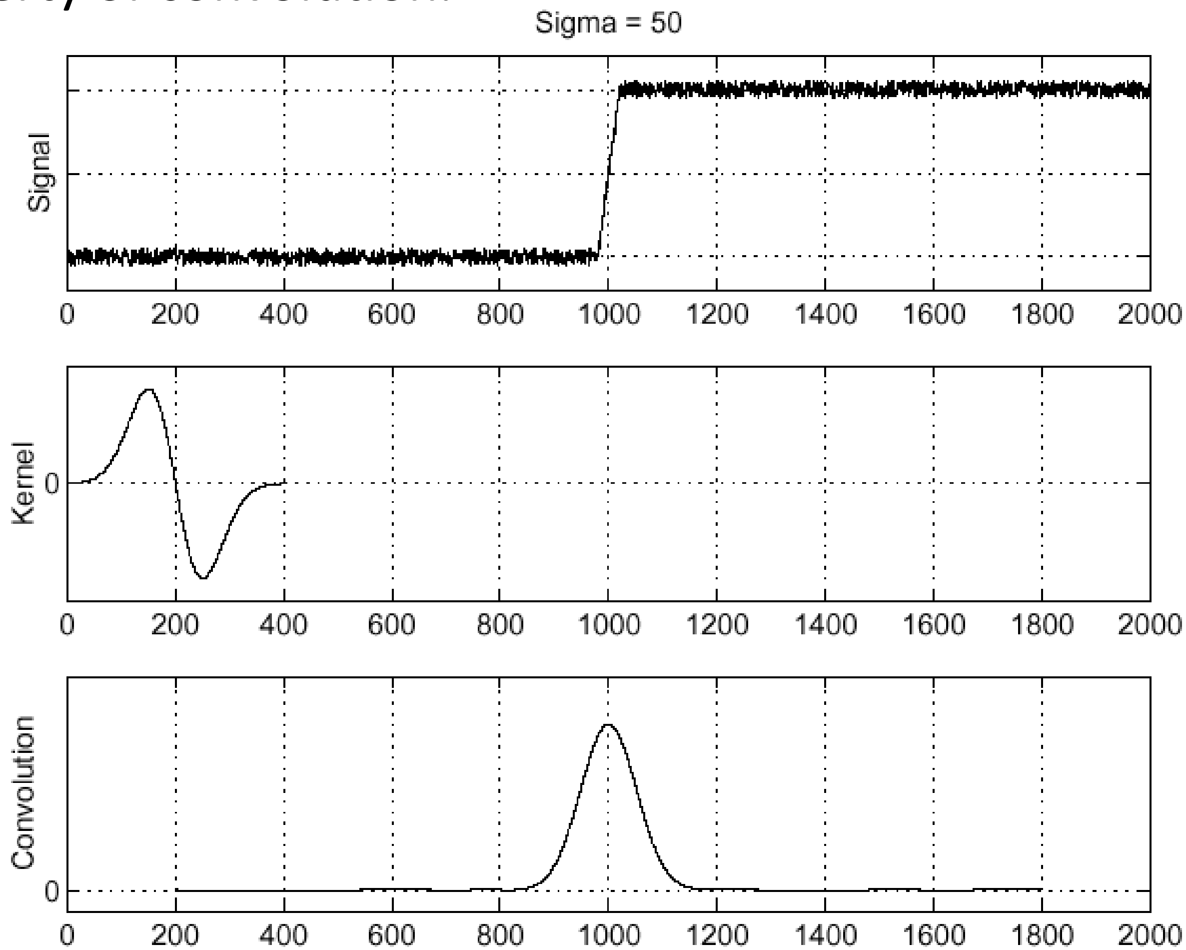
$$\frac{\partial}{\partial x}(I * H) = I * \frac{\partial H}{\partial x}$$

Differentiation property of convolution.

I

$\frac{\partial H}{\partial x}$

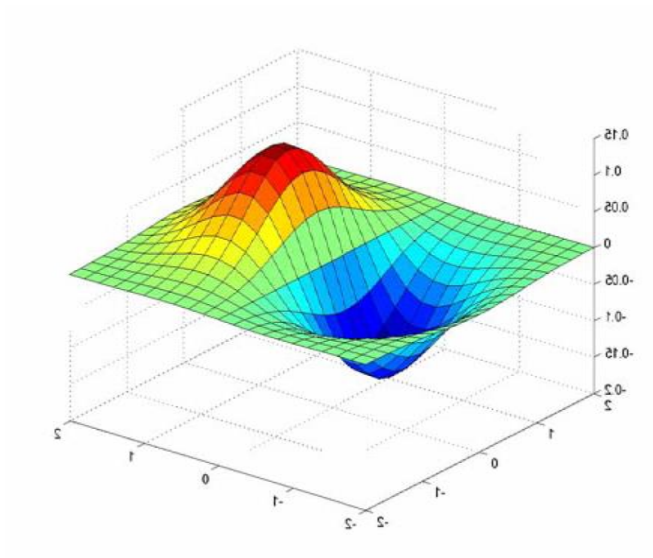
$I * \frac{\partial H}{\partial x}$



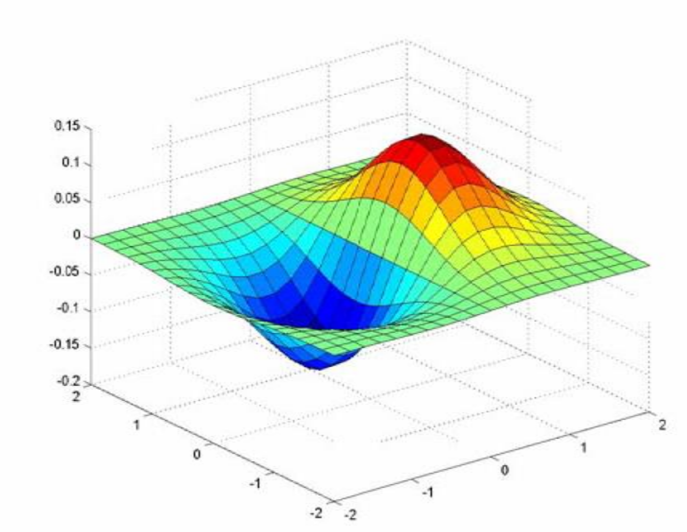
Derivative of Gaussian filter (along x)

$$(I * G) * H = I * (G * H)$$

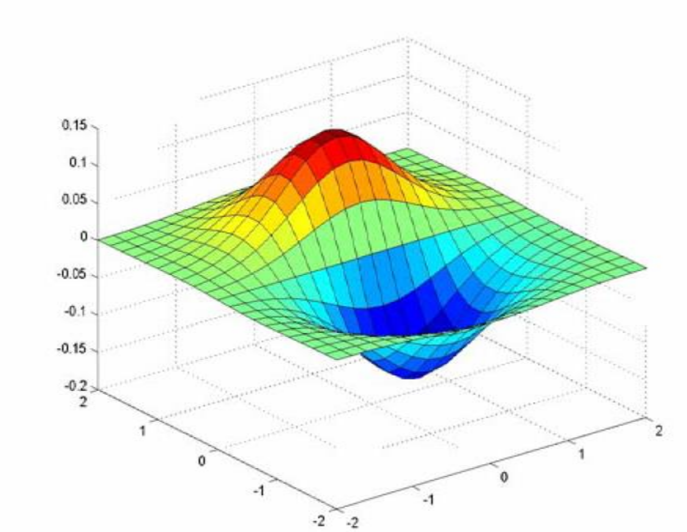
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} * \begin{bmatrix} 1 & -1 \end{bmatrix}$$



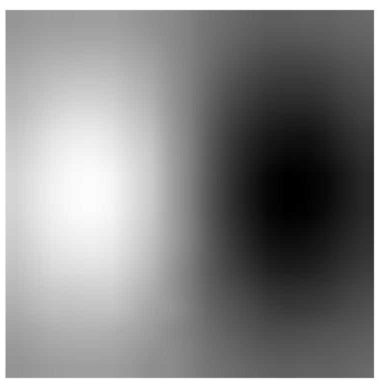
Derivative of Gaussian filters

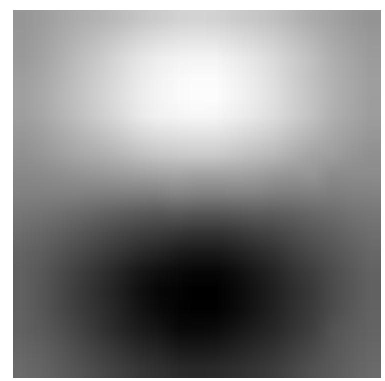


x-direction



y-direction





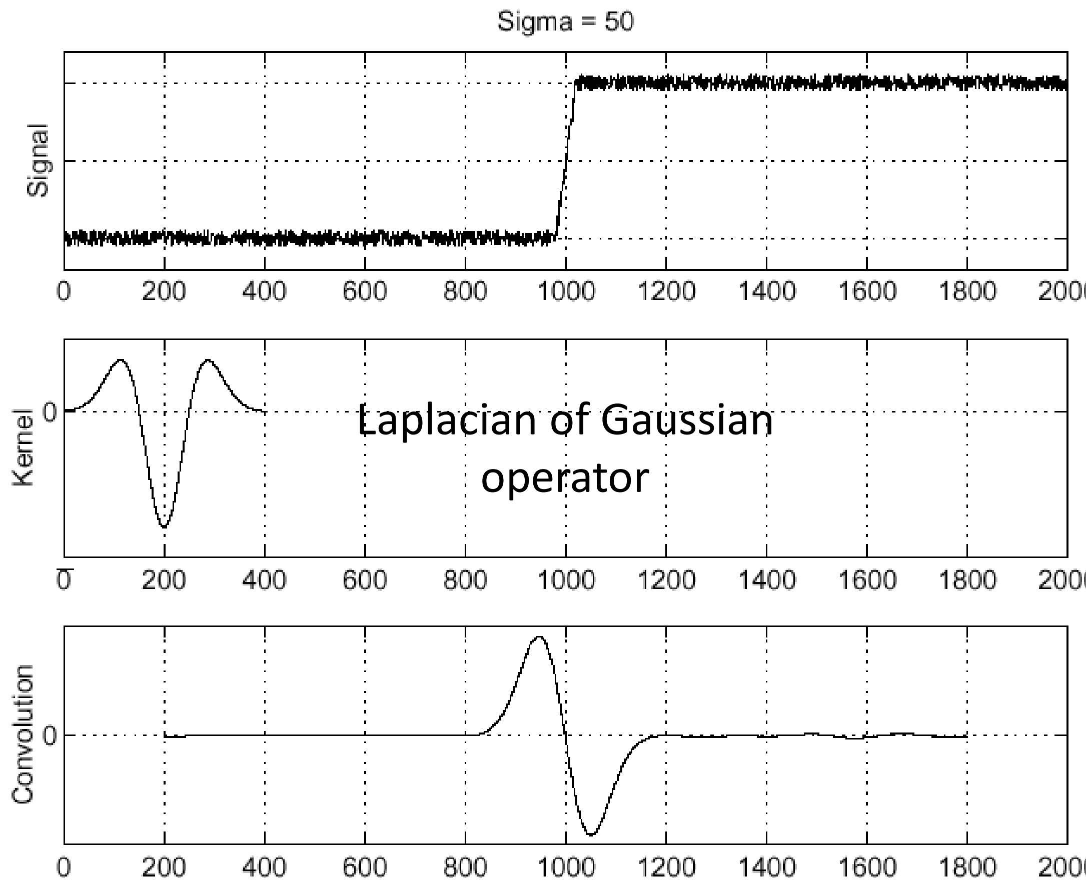
Laplacian of Gaussian

Consider $$\frac{\partial^2}{\partial x^2}(I * H) = I * \frac{\partial^2 H}{\partial x^2}$$

$I(x)$

$$\frac{\partial^2 H}{\partial x^2}$$

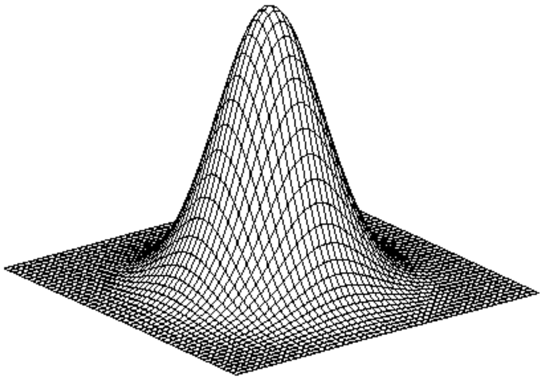
$$I * \frac{\partial^2 H}{\partial x^2}$$



Where is the edge?

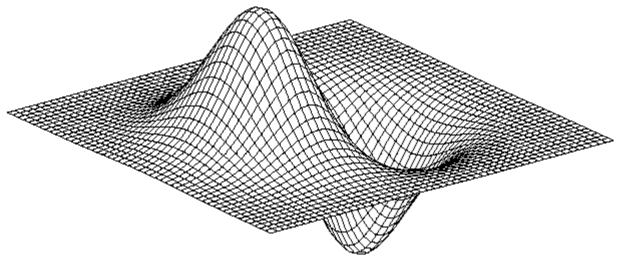
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

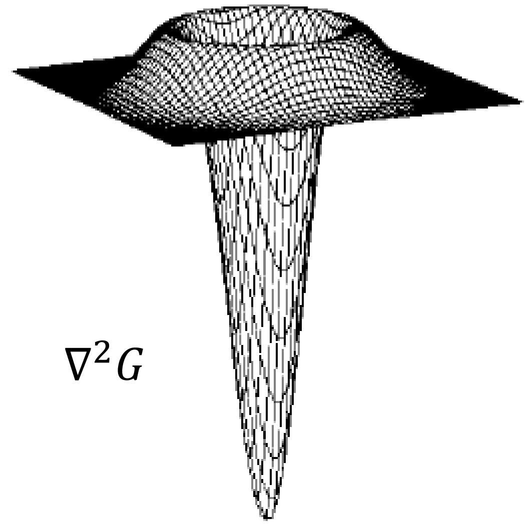
$$G = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial G}{\partial x}$$

Laplacian of Gaussian



$$\nabla^2 G$$

- ∇^2 is the Laplacian operator: $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$

Summary on (linear) filters

- Smoothing filter:
  - **has positive values**
  - **sums to 1** → preserve brightness of constant regions
  - **removes "high-frequency"** components: "low-pass" filter

- Derivative filter:
  - **has opposite signs** used to get high response in regions of high contrast
  - **sums to 0** → no response in constant regions
  - **highlights "high-frequency"** components: "high-pass" filter

The Canny edge-detection algorithm (1986)

Original image (Lenna image: https://en.wikipedia.org/wiki/Lenna)

Take a grayscale image. If not grayscale (i.g., RGB), convert it into a grayscale by replacing each pixel by the mean value of its R, G, B components.

The Canny edge-detection algorithm (1986)

Take a grayscale image. If not grayscale (i.g., RGB), convert it into a grayscale by replacing each pixel by the mean value of its R, G, B components.

Original image (Lenna image: https://en.wikipedia.org/wiki/Lenna)

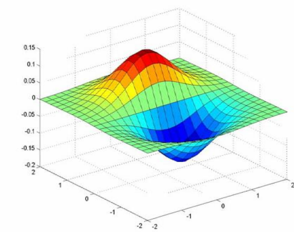
The Canny edge-detection algorithm (1986)

Convolve the image I with x and y derivatives of Gaussian filter

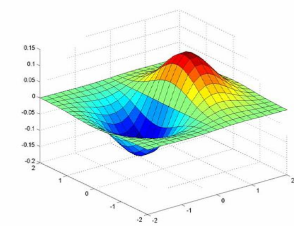
$$\frac{\partial I}{\partial x} = I * \frac{\partial G}{\partial x}$$

$$\frac{\partial I}{\partial y} = I * \frac{\partial G}{\partial y}$$

$\frac{\partial G}{\partial x}$



$\frac{\partial G}{\partial y}$



$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$: Edge strength

The Canny edge-detection algorithm (1986)

Thresholded $||\nabla I||$

Threshold it (i.e., set to 0 all pixels whose value is below a given threshold)

The Canny edge-detection algorithm (1986)

Take local maximum along gradient direction

Thinning: non-maxima suppression (local-maxima detection) along edge direction

Summary (things to remember)

- Image filtering (definition, motivation, applications)
- Moving average
- Linear filters and formulation: box filter, Gaussian filter
- Boundary issues
- Non-linear filters
  - Median & bilateral filters
- Edge detection
  - Derivating filters (Prewitt, Sobel)
  - Combined derivative and smoothing filters (deriv. of Gaussian)
  - Laplacian of Gaussian
  - Canny edge detector
- Readings: Ch. 3.2, 4.2.1 of Szeliski book

Understanding Check

Are you able to:

- Explain the differences between convolution and correlation?
- Explain the differences between a box filter and a Gaussian filter?
- Explain why should one increase the size of the kernel of a Gaussian filter if is large (i.e. close to the size of the filter kernel?
- Explain when would we need a median & bilateral filter?
- Explain how to handle boundary issues?
- Explain the working principle of edge detection with a $1D$ signal?
- Explain how noise does affect this procedure?
- Explain the differential property of convolution?
- Show how to compute the first derivative of an image intensity function along x and y?
- Explain why the Laplacian of Gaussian operator is useful?
- List the properties of smoothing and derivative filters?
- Illustrate the Canny edge detection algorithm?
- Explain what non-maxima suppression is and how it is implemented?