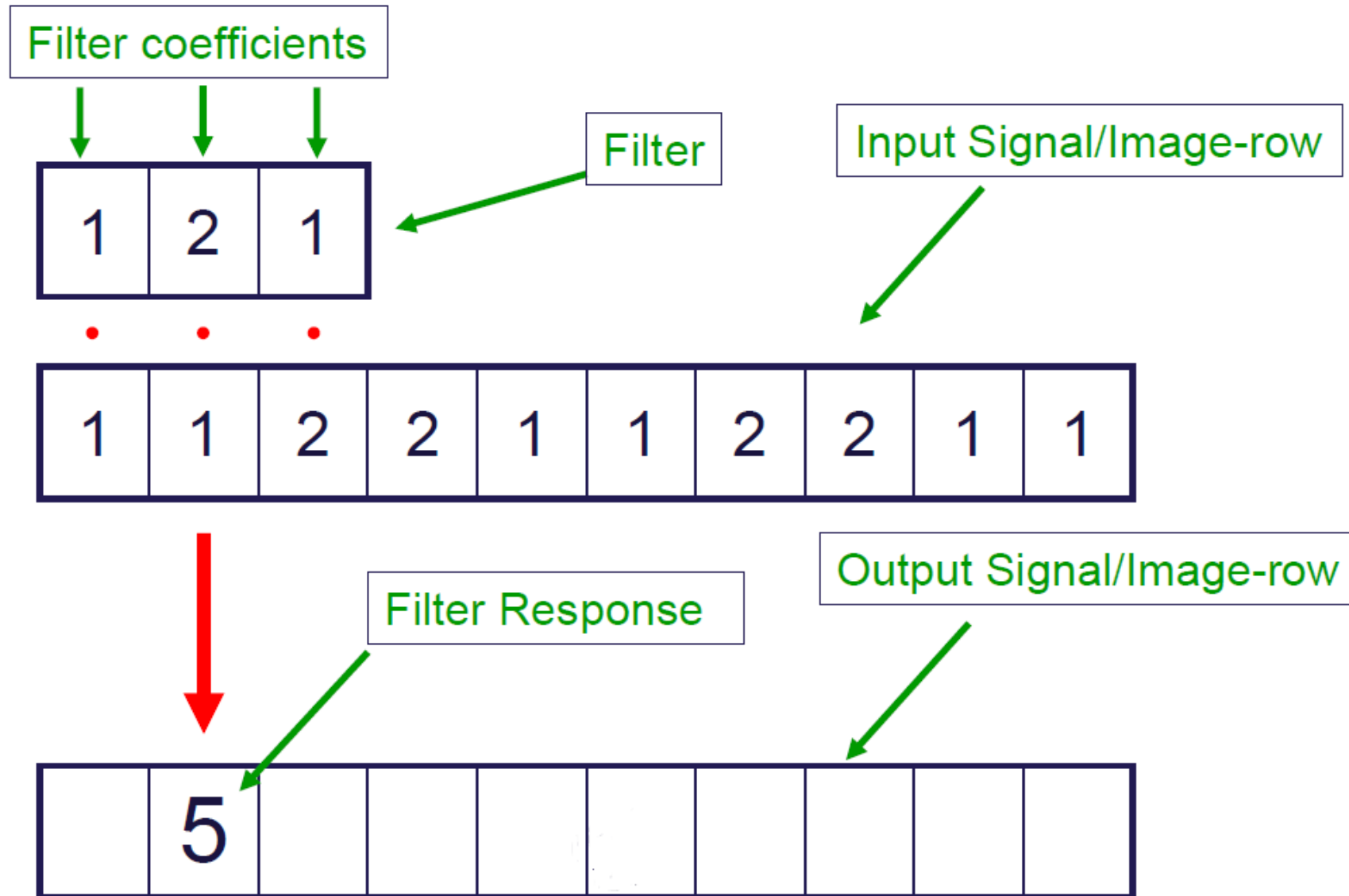


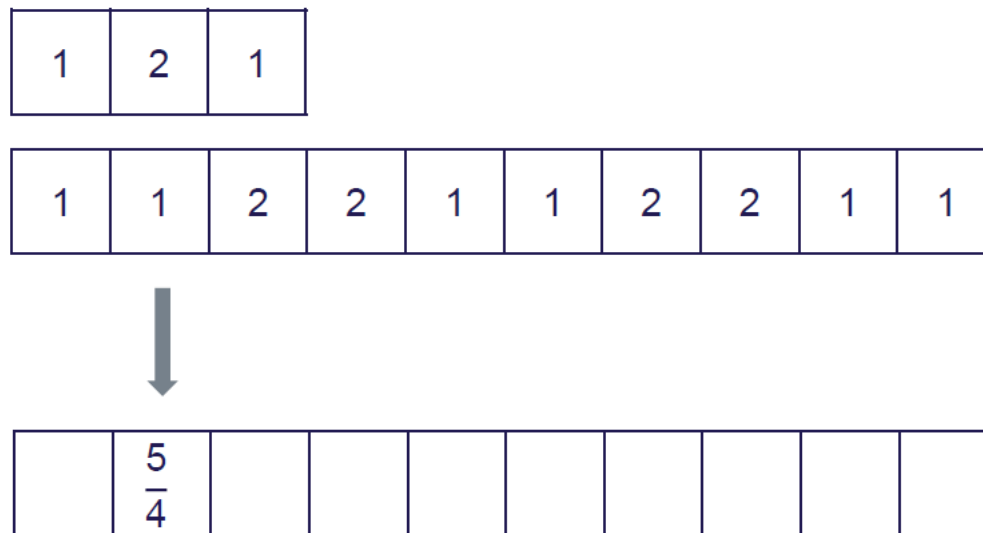
# FBSP: Neighbourhood processing methods

Nick Yao Larsen  
nylarsen@cfin.au.dk

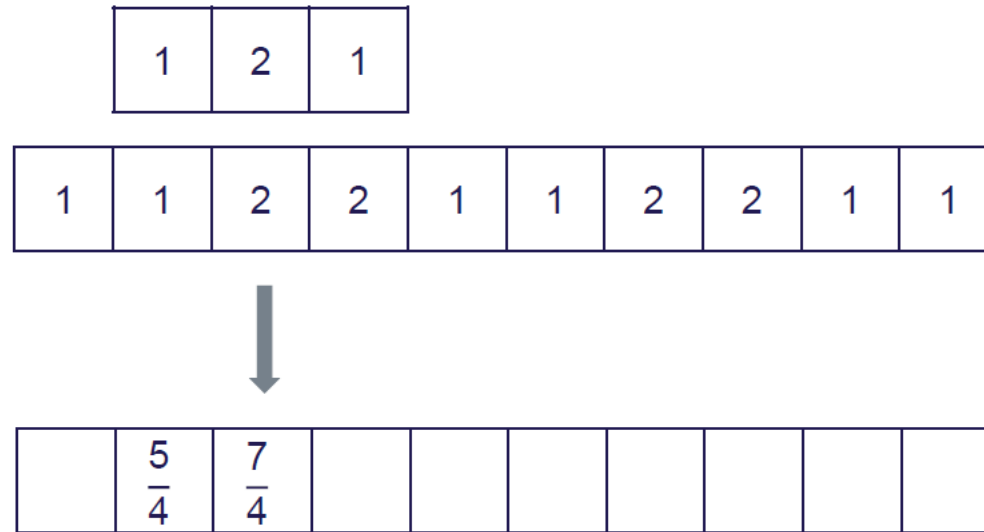
# Correlation (1D)



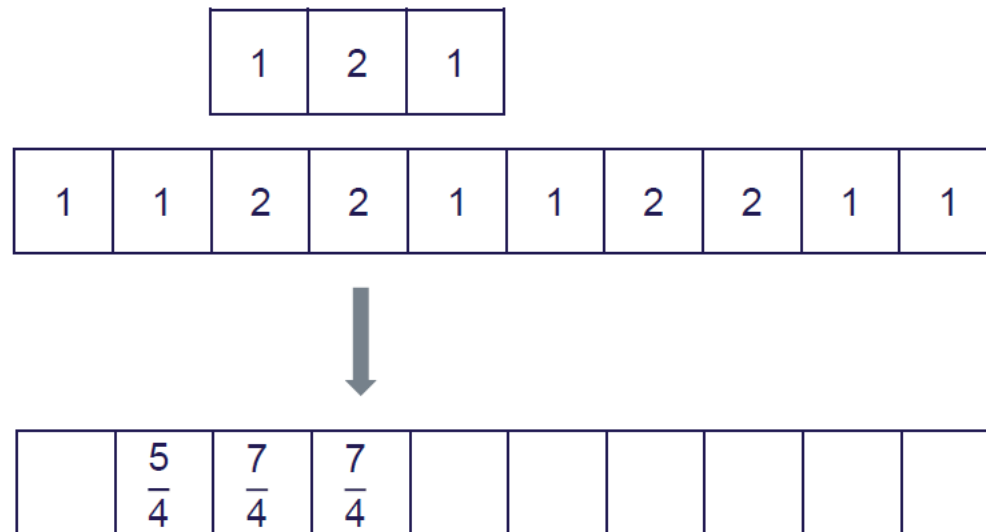
# Correlation (1D)



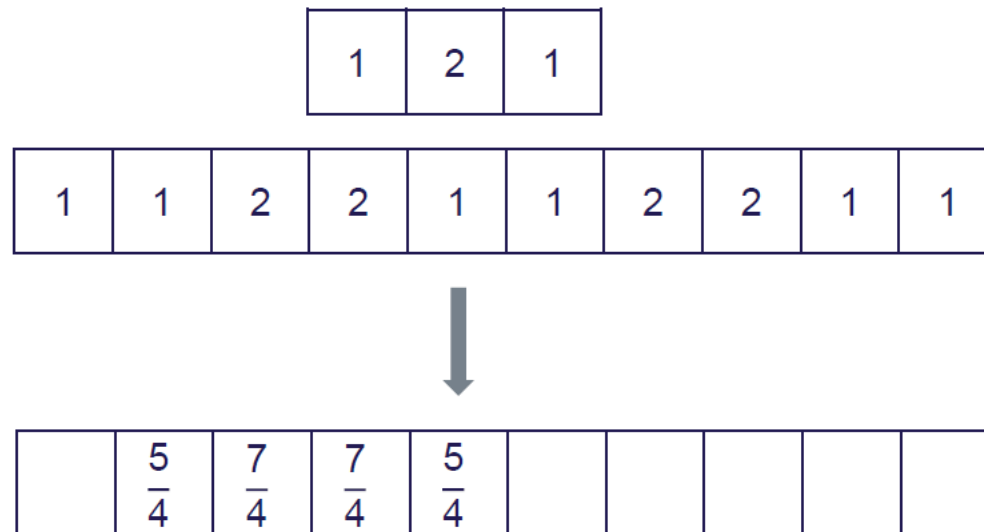
# Correlation (1D)



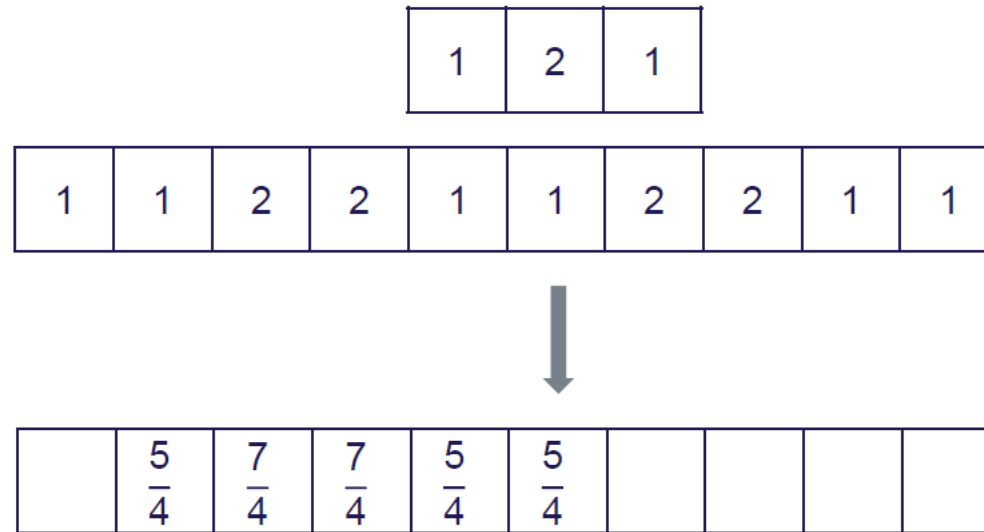
# Correlation (1D)



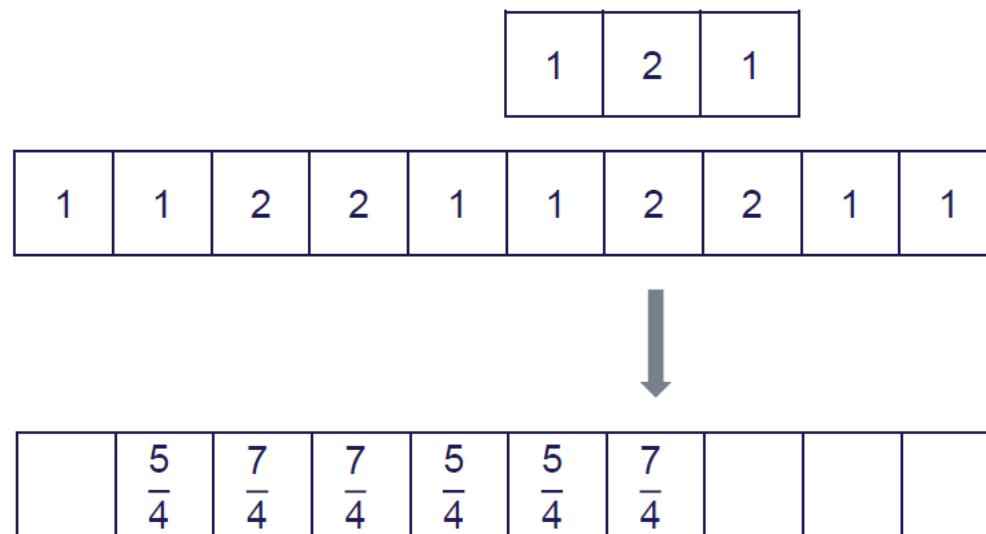
# Correlation (1D)



# Correlation (1D)

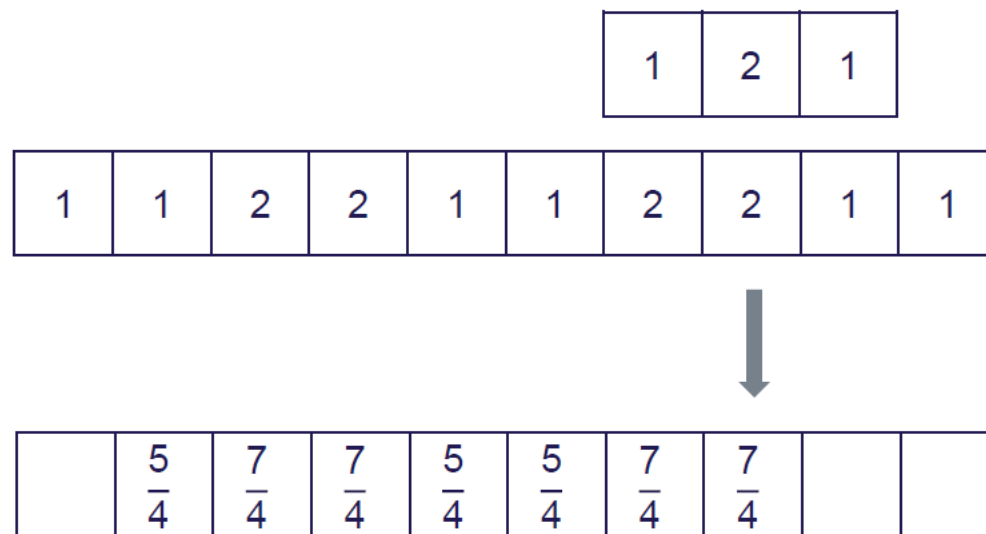


# Correlation (1D)

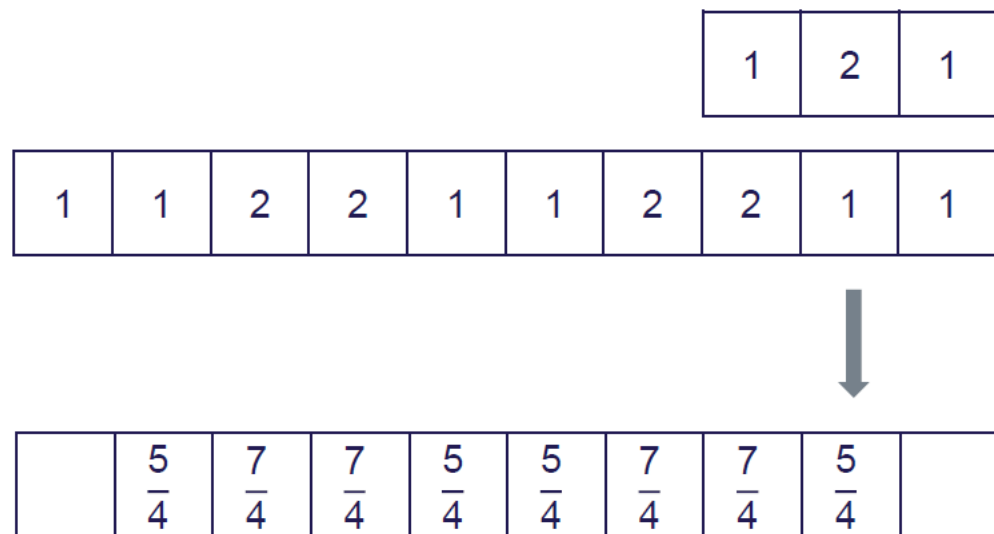




# Correlation (1D)



# Correlation (1D)

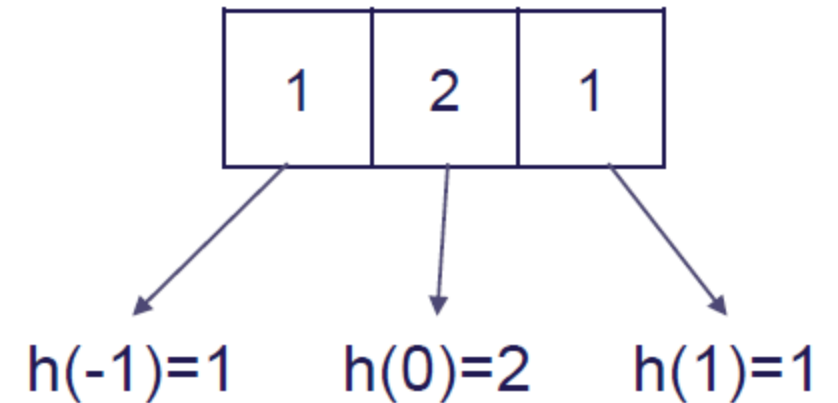


# The math of correlation

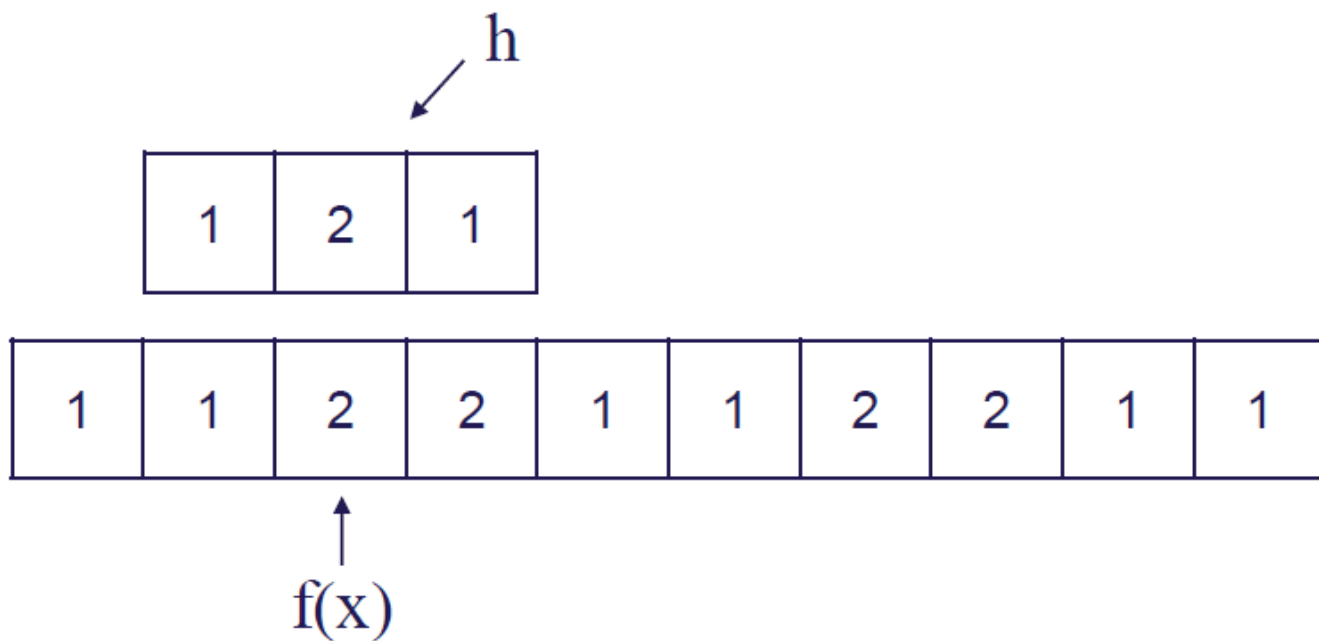
$$g(x) = h \circ f(x) = \sum_{i=-n}^n h(i) f(x+i)$$

- $g(x)$ : output
- $h$ : filter
- $\circ$  means correlation
- $f(x)$ : input
- $n$  = radius of filter

Example filter:



# The math of correlation



$$g(x) = h \circ f(x) = \sum_{i=-n}^n h(i) f(x+i)$$

$$i = -1: h(-1) \cdot f(x-1) = 1 \cdot 1 = 1$$

$$i = 0: h(0) \cdot f(x) = 2 \cdot 2 = 4$$

$$i = 1: h(1) \cdot f(x+1) = 1 \cdot 2 = 2$$

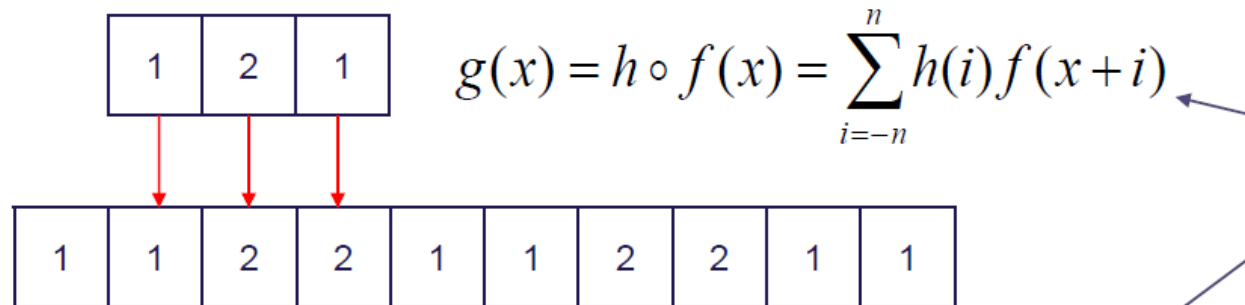
$$g(x) = 1 + 4 + 2 = 7$$

$$\text{Normalize: } g(x) = 7 / 4$$

# Correlation vs. Convolution

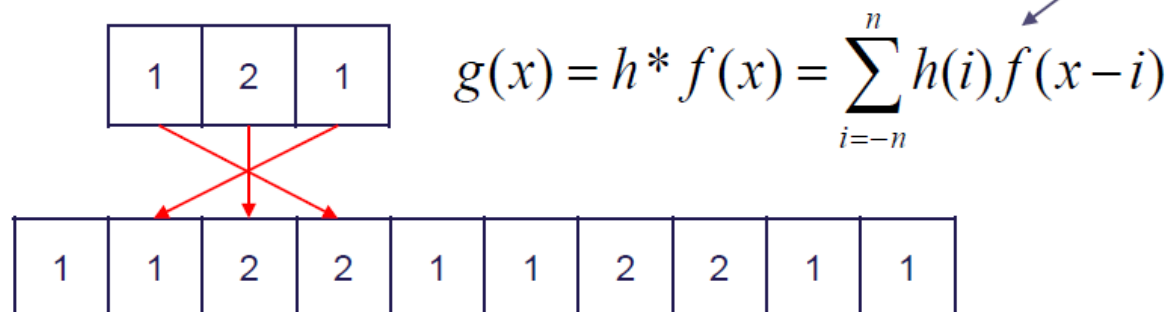
The key difference lies in the orientation of the filter during the operation.

Essentially, when applying **convolution** the filter is flipped both horizontally and vertically before being applied to the input image.



• Correlation

• Convolution



• In image processing we use correlation, but (nearly) always call it convolution! We're silly like that.

• Note: When the filter is symmetric: correlation = convolution!

# Correlation vs. Convolution Example

- Suppose we have a 3x3 kernel:
- $h = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$  Original kernel (used in correlation)
- In **correlation**, this kernel is used as is.
- In **convolution**, the kernel **is flipped** both horizontally and vertically before applying it to the image.
- $h = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$  Flipped kernel (used in convolution):

The flipping of the kernel in convolution is a mathematical convention that facilitates the analysis and manipulation of signals and images in both the spatial and frequency domains.

Inspiration: <https://glassboxmedicine.com/2019/07/26/convolution-vs-cross-correlation/>


# Summary – correlation vs convolution

- In image processing, correlation and convolution are two closely related operations that involve the interaction of an image with a filter or kernel. While they are similar in their mathematical formulation, there is a key difference in the way the filter is applied to the image.
- **Convolution:** The filter (also known as a kernel or mask) **is flipped** both horizontally and vertically before the element-wise multiplication with the image. The filter is moved over the image, and at each position, the element-wise product of the filter and the overlapped region of the image is calculated.
- **Correlation:** The filter is **not flipped** before being applied. It is simply moved over the image, and at each position, the element-wise product of the filter and the overlapped region of the image is calculated. The sum of these products gives the value of the corresponding pixel in the output image.

# Convolution on images

- The filter is now in 2D
- This is called a kernel, and the coefficients are kernel coefficients.
- They are often square, and always odd-sized: 3x3, 5x5, 7x7, etc.

Normalization


$$\frac{1}{10}$$

1	1	1
1	2	1
1	1	1

1	2	0	1	3
2	1	4	2	2
1	0	1	0	1
1	2	1	0	2
2	5	3	1	2


	$\frac{13}{10}$			



# Convolution on images

- The filter is now in 2D
- This is called a kernel, and the coefficients are kernel coefficients.
- They are often square, and always odd-sized: 3x3, 5x5, 7x7, etc.

Normalization


$$\frac{1}{10}$$

1	1	1
1	2	1
1	1	1


1	2	0	1	3
2	1	4	2	2
1	0	1	0	1
1	2	1	0	2
2	5	3	1	2

	$\frac{13}{10}$	$\frac{15}{10}$		

# Convolution on images

- The filter is now in 2D
- This is called a kernel, and the coefficients are kernel coefficients.
- They are often square, and always odd-sized: 3x3, 5x5, 7x7, etc.

Normalization


$$\frac{1}{10}$$

1	1	1
1	2	1
1	1	1

1	2	0	1	3
2	1	4	2	2
1	0	1	0	1
1	2	1	0	2
2	5	3	1	2

	$\frac{13}{10}$	$\frac{15}{10}$	$\frac{16}{10}$	

# 2D convolution in practice

output

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

7	6	5	5	6	7
6	4	3	3	4	6
5	3	2	2	3	5
5	3	2	2	3	5
6	4	3	3	4	6
7	6	5	5	6	7

input

# Quiz!

- Fill out the rest of the output image

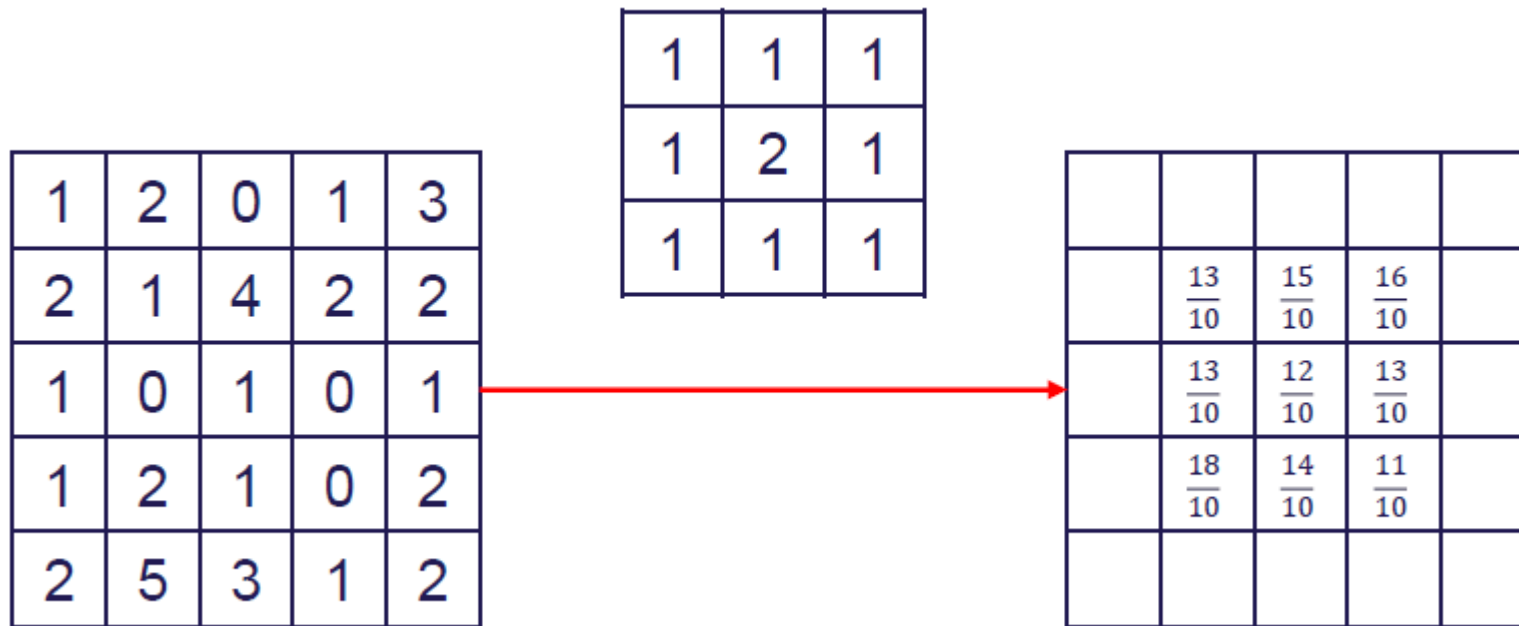
1	2	0	1	3
2	1	4	2	2
1	0	1	0	1
1	2	1	0	2
2	5	3	1	2

1	1	1
1	2	1
1	1	1

	$\frac{13}{10}$	$\frac{15}{10}$	$\frac{16}{10}$	

# Quiz!

- Fill out the rest of the output image



# Math of 2D correlation/convolution

Correlation

$$g(x, y) = h \circ f(x, y) = \sum_{j=-R}^R \sum_{i=-R}^R h(i, j) f(x+i, y+j)$$

Convolution

$$g(x, y) = h * f(x, y) = \sum_{j=-R}^R \sum_{i=-R}^R h(i, j) f(x-i, y-j)$$

R is the radius  
of a kernel

Note: When the filter is symmetric: correlation = convolution!

1	1	1
1	1	1
1	1	1

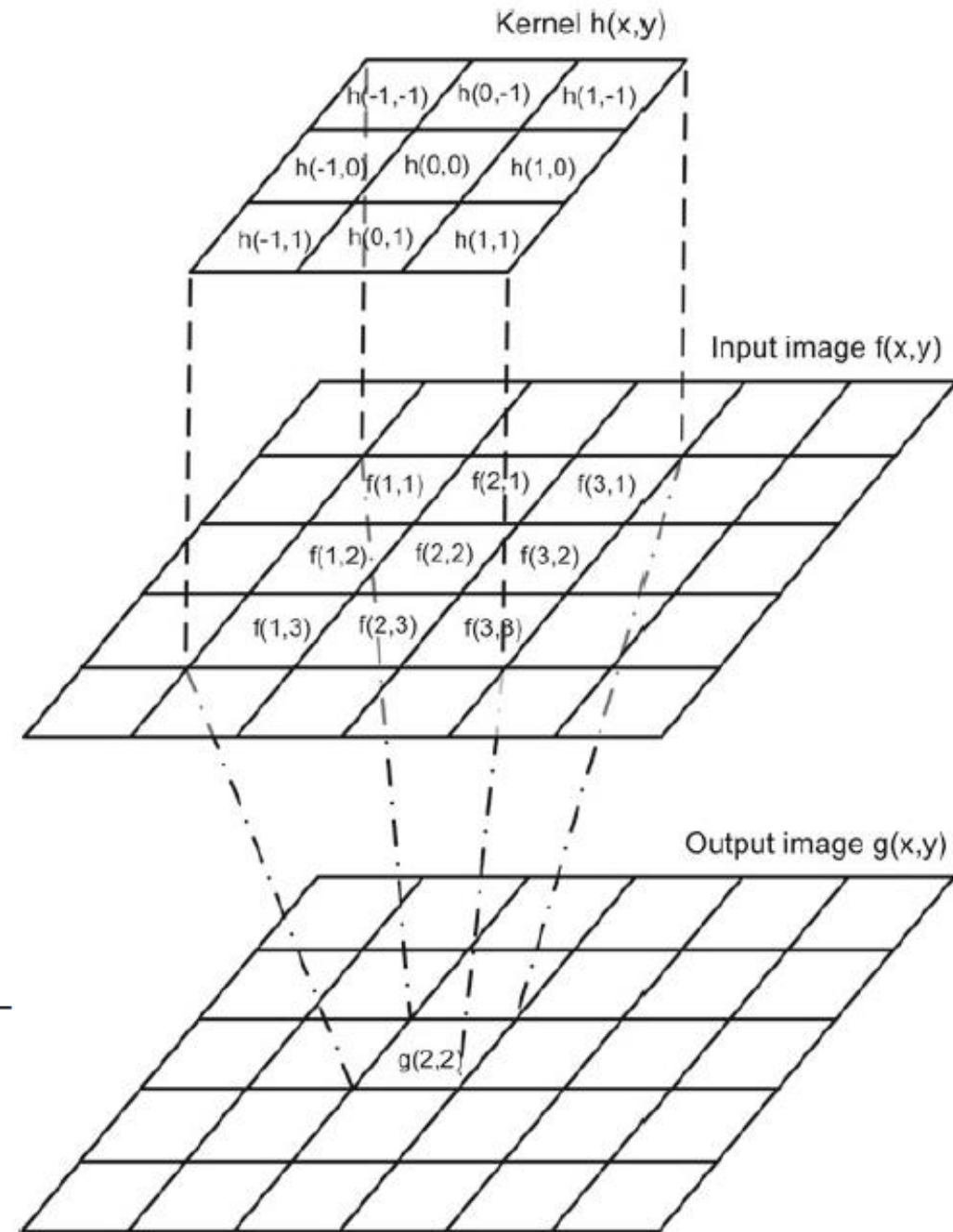
2	3	2
-1	0	-1
2	3	2

# Example

$$g(x, y) = \sum_{j=-R}^R \sum_{i=-R}^R h(i, j) \cdot f(x + i, y + j),$$

$$g(2, 2) =$$

$$\begin{aligned} &h(-1, -1) \cdot f(1, 1) + h(0, -1) \cdot f(2, 1) + h(1, -1) \cdot f(3, 1) + \\ &h(-1, 0) \cdot f(1, 2) + h(0, 0) \cdot f(2, 2) + h(1, 0) \cdot f(3, 2) + \\ &h(-1, 1) \cdot f(1, 3) + h(0, 1) \cdot f(2, 3) + h(1, 1) \cdot f(3, 3) \end{aligned}$$



# Problems on the borders

- Why is the output image smaller than the input?
- The bigger the kernel, the bigger the problems
- Does it matter? Yes, if we are going to combine images afterwards.

1	2	0	1	3
2	1	4	2	2
1	0	1	0	1
1	2	1	0	2
2	5	3	1	2

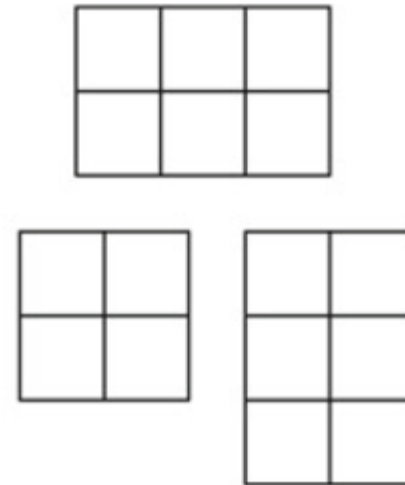
	$\frac{13}{10}$	$\frac{15}{10}$	$\frac{16}{10}$	
	$\frac{13}{10}$	$\frac{12}{10}$	$\frac{13}{10}$	
	$\frac{18}{10}$	$\frac{14}{10}$	$\frac{11}{10}$	



# Problems on the borders “solutions”

- Crop the input image after processing
- Add a fixed value around the edge: 0, 255 (padding)
  - Changes histogram
- Truncate kernel when at the edge: 3x3 => for example 2x3:
  - Complex and not well-defined
- Mirror padding
  - Copy the outer border in the input
  - Copy the outer border in the output

1	2	0	1	3
2	1	4	2	2
1	0	1	0	1
1	2	1	0	2
2	5	3	1	2

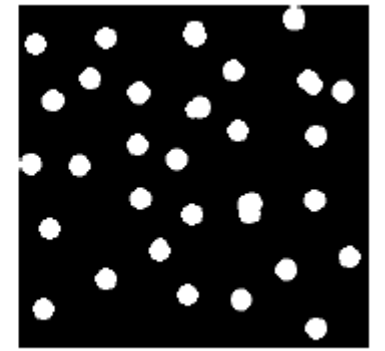
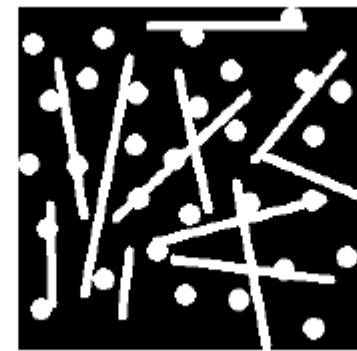


Special kernel sizes

	$\frac{13}{10}$	$\frac{15}{10}$	$\frac{16}{10}$	
	$\frac{13}{10}$	$\frac{12}{10}$	$\frac{13}{10}$	
	$\frac{18}{10}$	$\frac{14}{10}$	$\frac{11}{10}$	

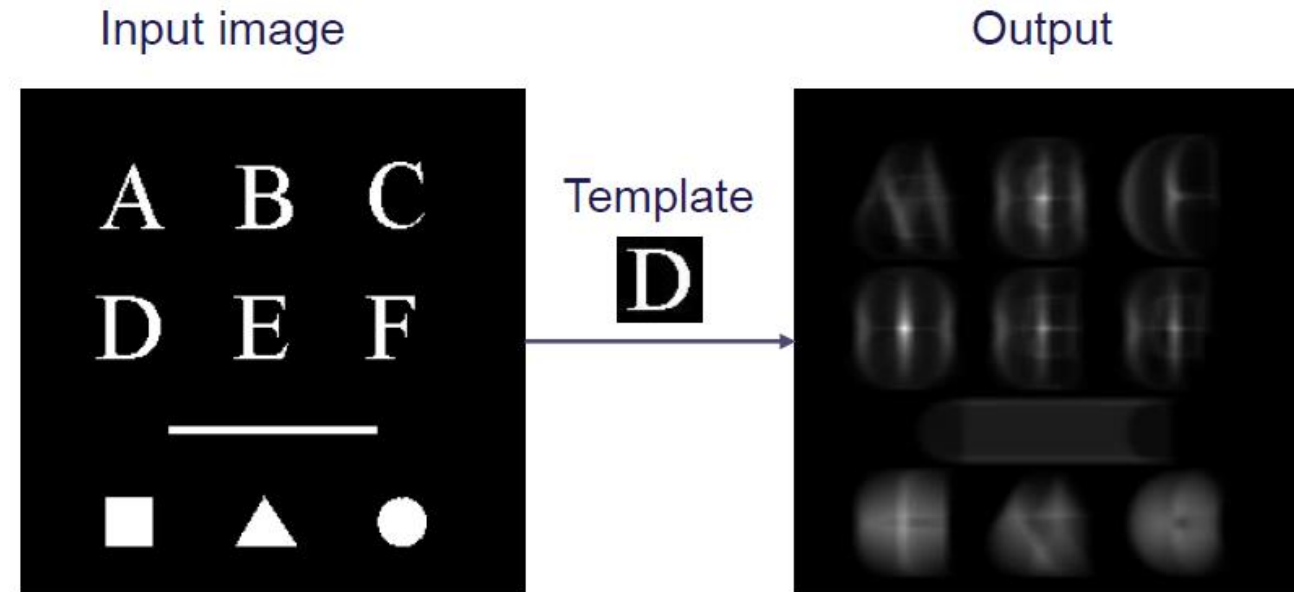
# Application of correlation

- Many things defined by the programmer and some standard operations are:
  - Image blur
  - Remove noise
  - Object detection
  - Edge detection
  - Morphology (Next lecture)



# Template matching

- The filter is called a template or a mask
- The template is just an ordinary picture, which is correlated with the input image
- The brighter the value in the output, the better the match
- Problem: Bright pixels will produce high values!
- Solution: Use normalized cross-correlation

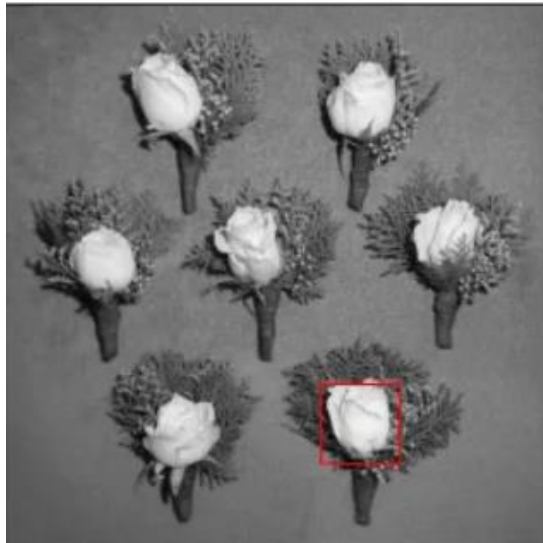


# Template matching

Grayscale Image



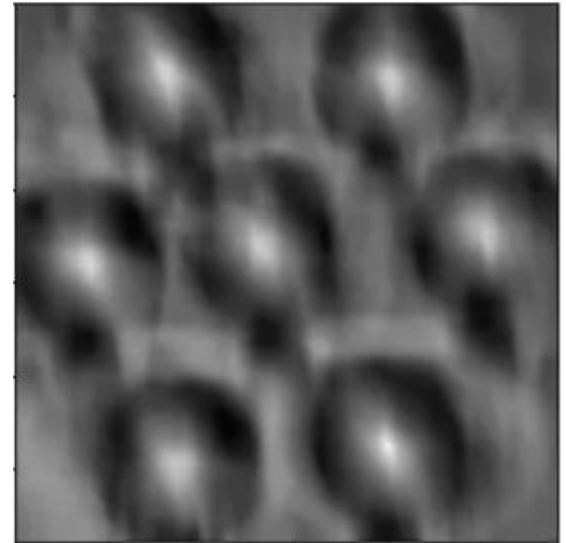
Patch Location



Patch



Template Matching



Through the examples shown:  
We were able to successfully detect all the small flower bouquets by only using one of the flower bouquets as a template

# Image blur



# Image blur

- Also known as: **Smoothing kernel**, low pass filter
- The simplest filter: Mean filter (spatial low pass filter)
- Another option: **Gaussian filter**

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

1	1	1
1	1	1
1	1	1

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

1	2	1
2	4	2
1	2	1



Input image



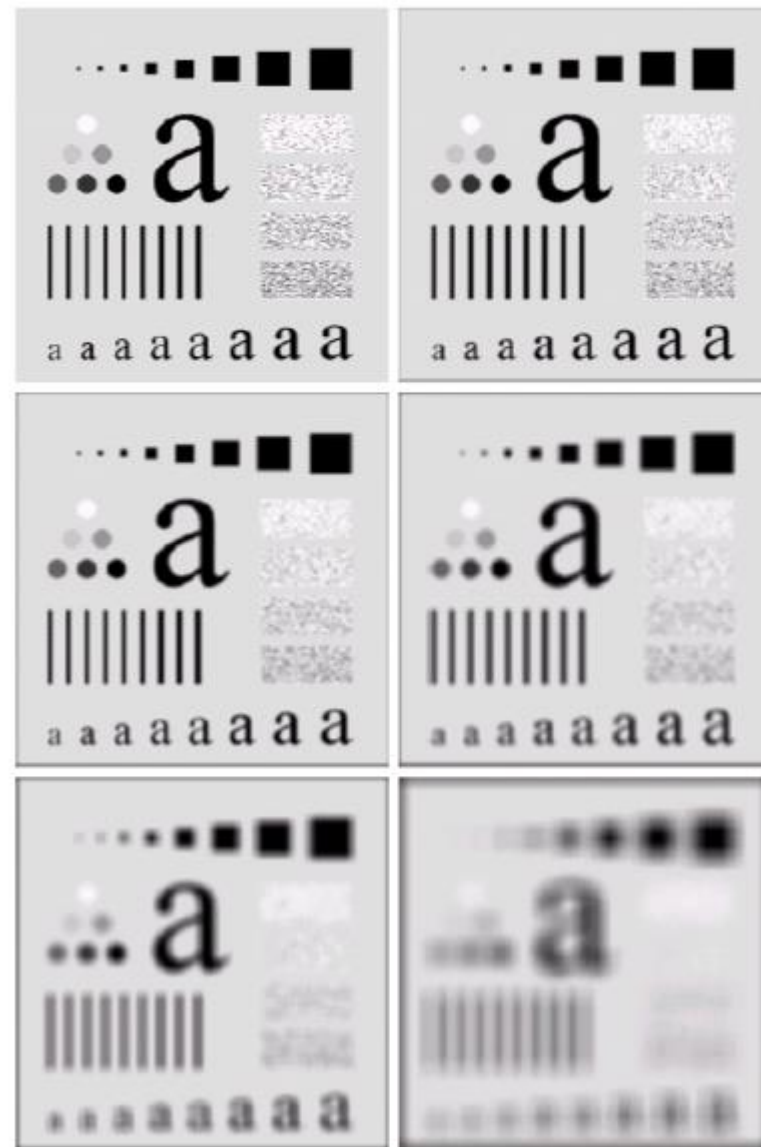
11x11 kernel



29x29 kernel

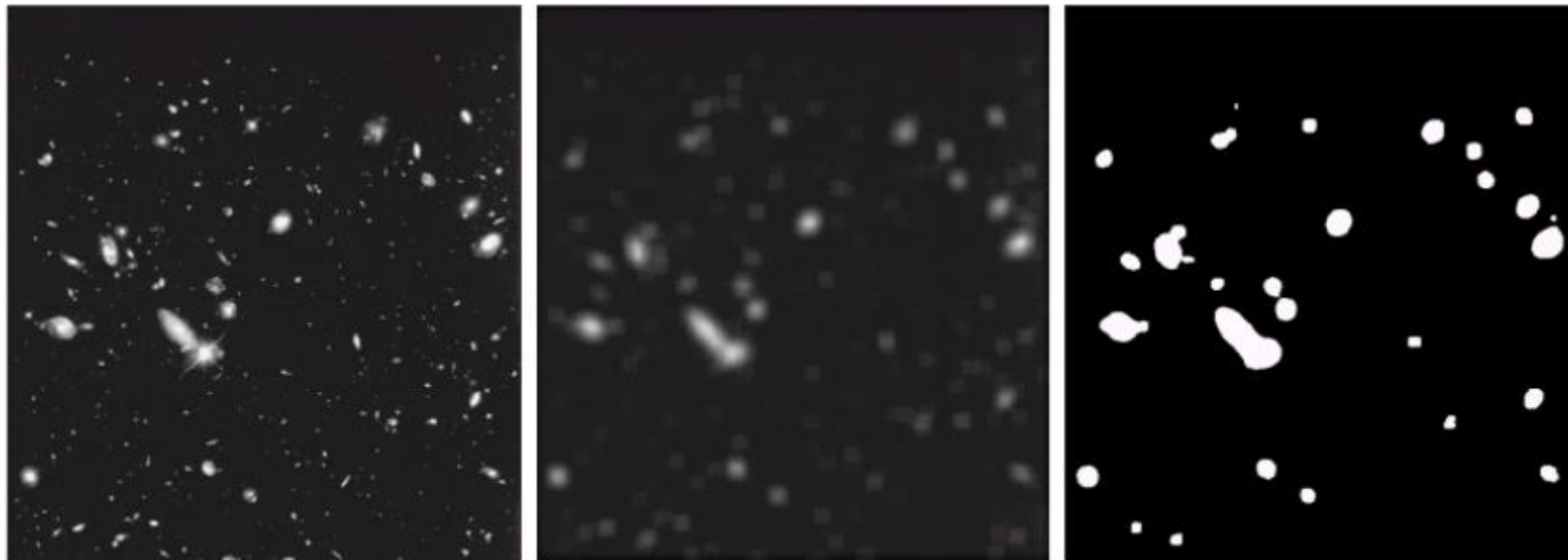
# Application of blurring

- Blurring to remove
- identity or other details
- Degree of blurring = kernel size



# Application of blurring

- Preprocessing: enhance objects
- Blurring + Thresholding



a b c

**FIGURE 3.36** (a) Image from the Hubble Space Telescope. (b) Image processed by a  $15 \times 15$  averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)



# Rank filters

- Aka: order-statistics filters
- Not based on correlation but still neighborhood processing
- Principle:
  - Define a mask/kernel, e.g., 3x3
  - Sort all pixel-values within the mask into ascending order
  - Select a pixel-value according to the filter type: Median, min., max., range,...
- Example to the right: Median filter

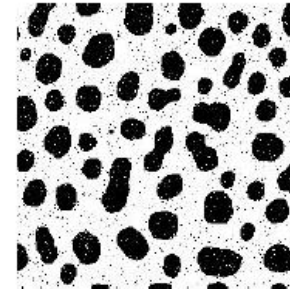
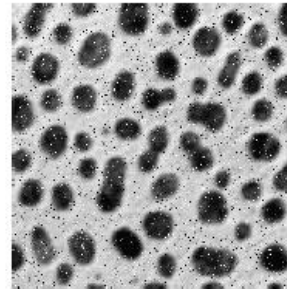
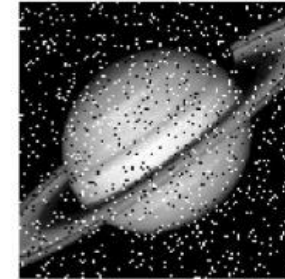
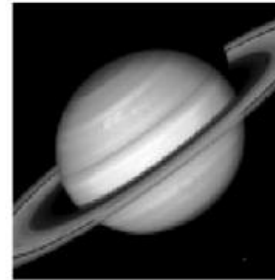
1	2	0	1	3
2	1	4	2	2
1	0	1	0	1
1	2	1	0	2
2	5	3	1	2

Sorted: 0, 0, 1, 1, 1, 1, 2, 2, 4

	1			

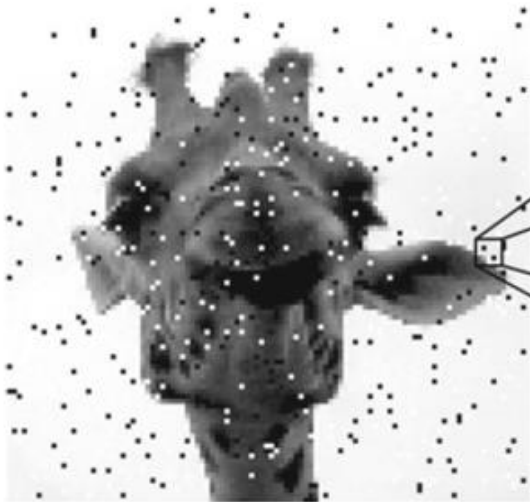
# Median filter

- Good for cleaning salt-and-pepper noise
- Good at removing noise in binary images
- Better than the mean filter as blurring is minimized and edges stay sharp

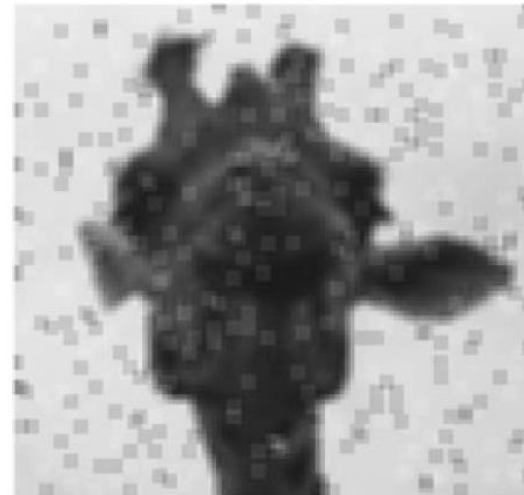


# Difference between mean and median filter

- Apply a Median and a Mean filter (use a 3x3 kernel) on the image below. Think about the differences.



205	204	204	206	255
206	0	208	206	206
201	199	205	206	209
61	128	213	0	205
59	65	255	206	255



Mean filtered



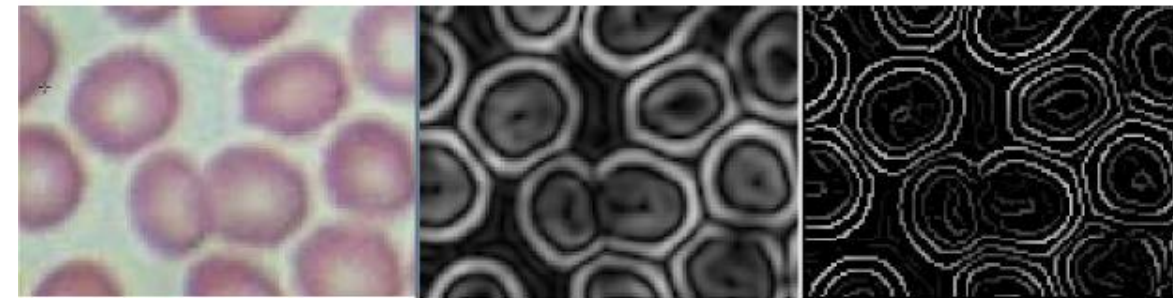
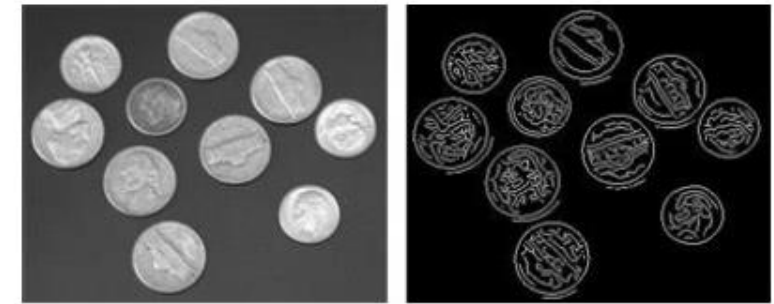
Median filtered

Ordering : [0, 199, 201, 204, 204, 205, 205, 206, 208]

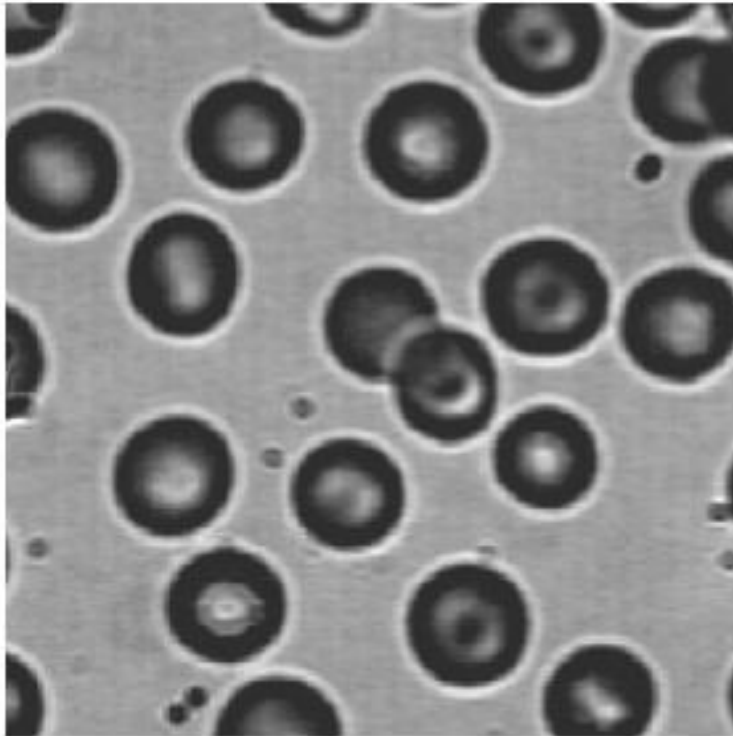
Median = 204

# Why are edges interesting?

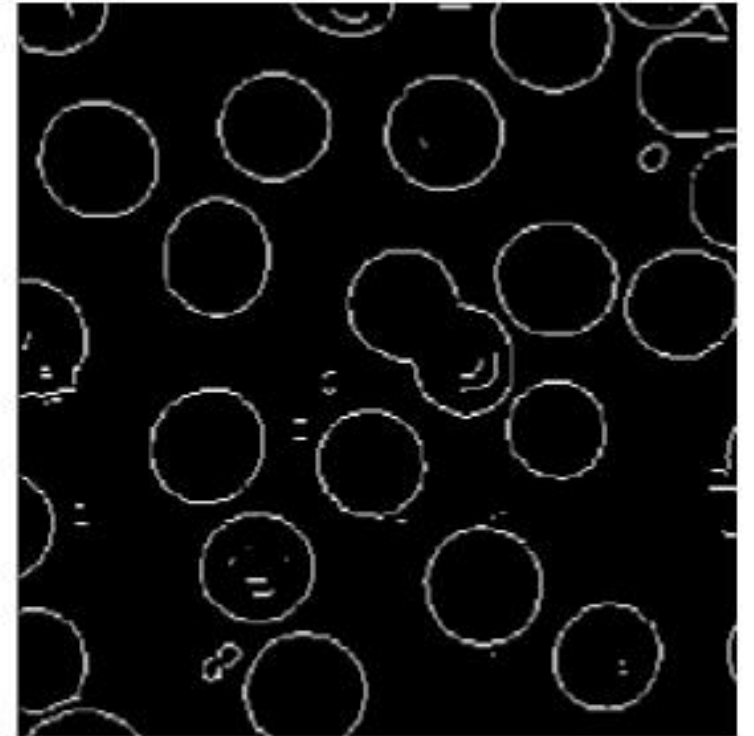
- Edges can (often) represent the information in the image (the objects)
- A higher level of abstraction (less data to process!)
- Edges are features:
  - “Independent” of illumination. As opposed to e.g. color information
- Object recognition and detection (often) use edge features
  - This is true for computer vision as well as for biological vision systems!
- Machine Vision: Excellent for measurements of size



# What are edges?

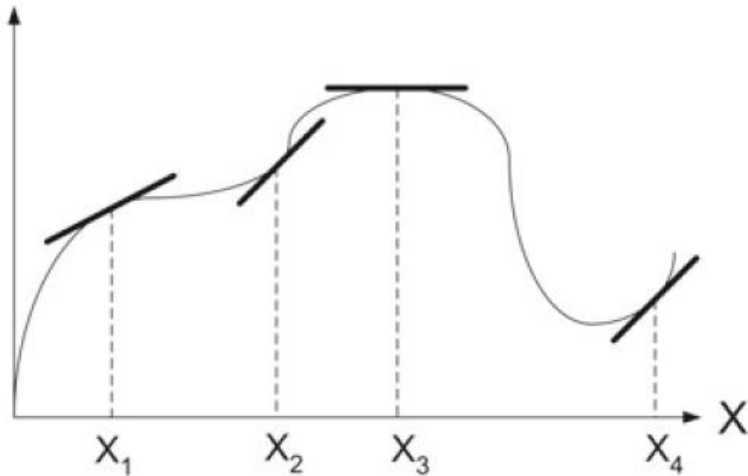


Edge detection



# What are edges?

- To enable edge detection we need to apply the concept of gradients. In the 1D case, we can define the gradient of a point as the slope of the curve at this point. Concretely this corresponds to the slope of the tangent at this point. The tangents to several points are displayed in the figure.

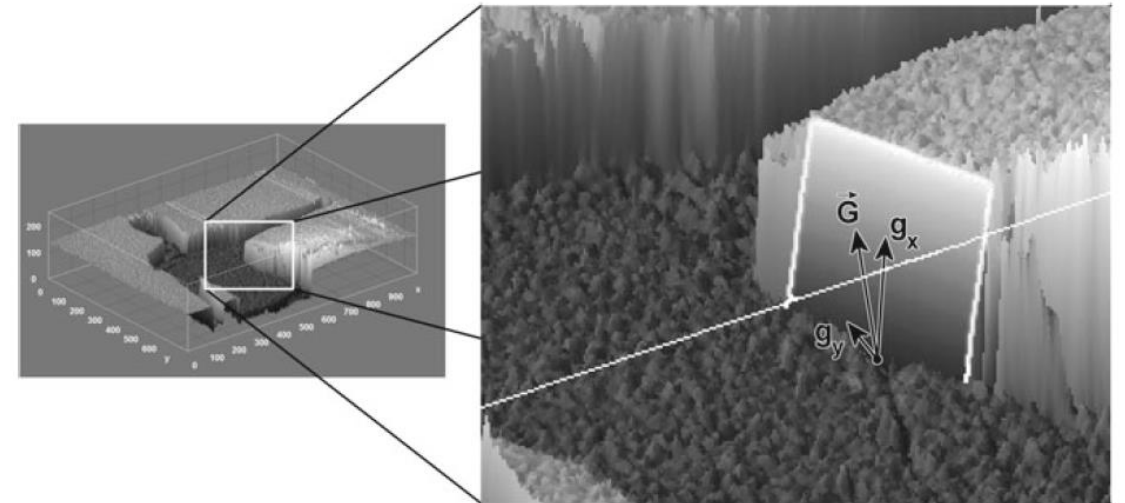
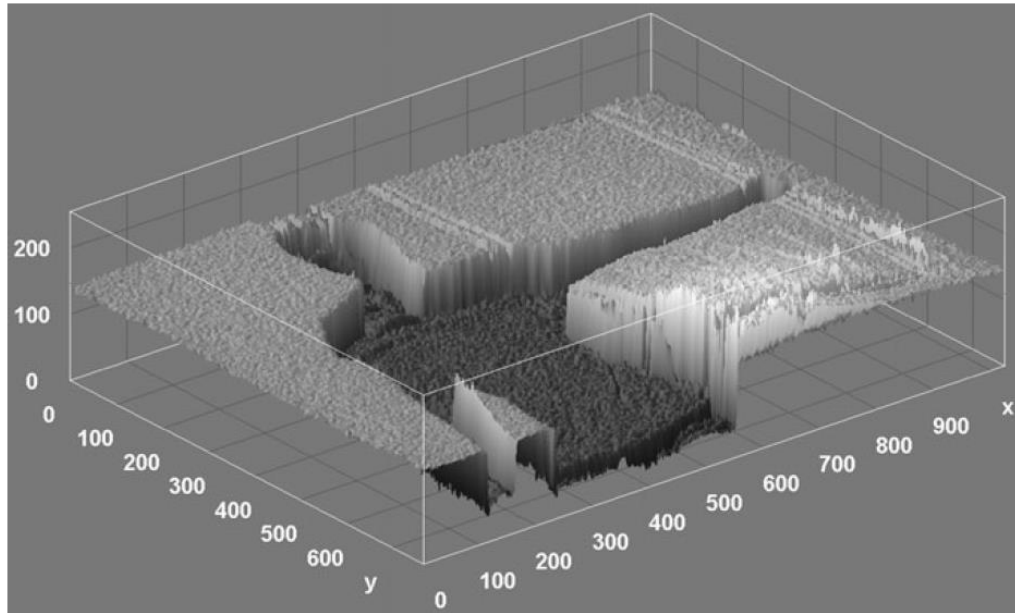


- If we represent an image by height as opposed to intensity, then edges correspond to places where we have steep hills!
- For each point in this image landscape we have two gradients: one in the x-direction and one in the y-direction. Together these two gradients span a plane, known as the tangent plane, which intersects the point



# What are edges?

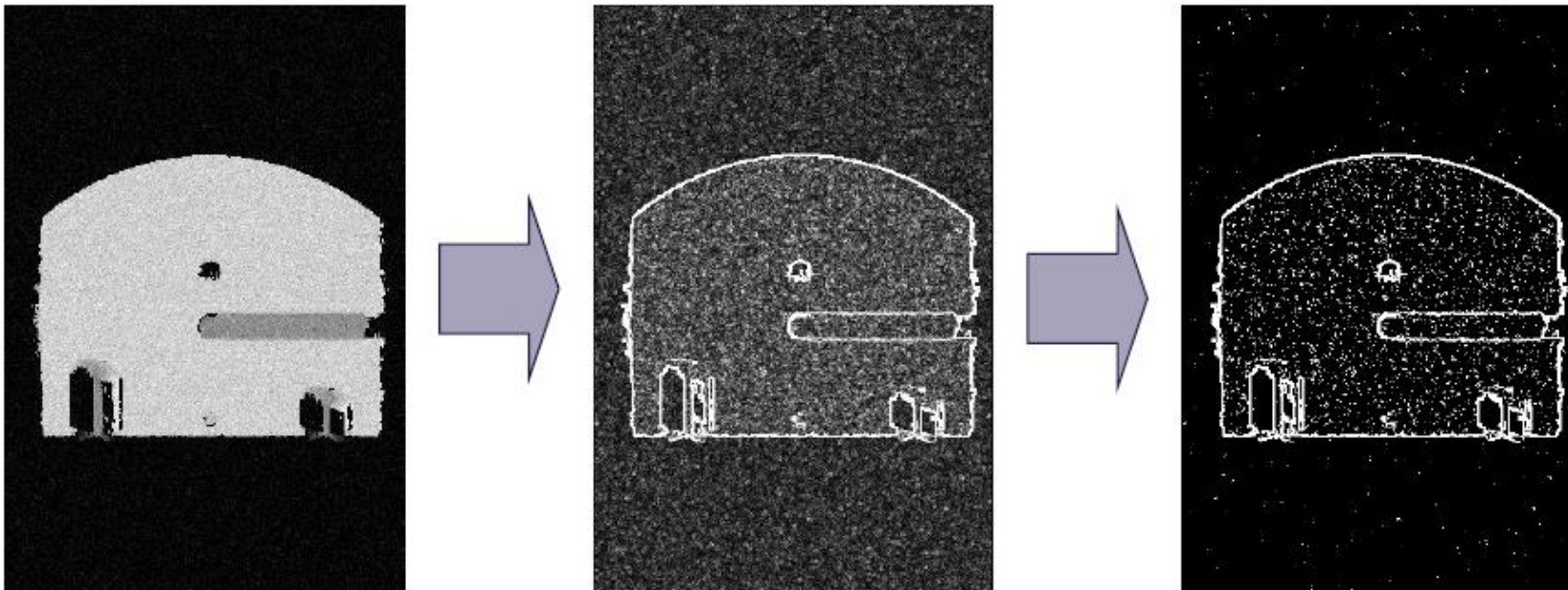
- Local intensity change
- Strong edge = the steep areas in a 3D plot
- Finding steep areas -> Look at the derivative



**Fig. 5.14** In a 3d representation of an image, a tangent plane is present for each point. Such a plane is defined by two gradient vectors in x- and y-direction, respectively. Here the tangent plane is shown for one pixel

# Edge detection steps

1. Noise reduction
2. Edge enhancement
  - Calculate candidates for the edges
3. Edge localization
  - Decide which edge candidates to keep





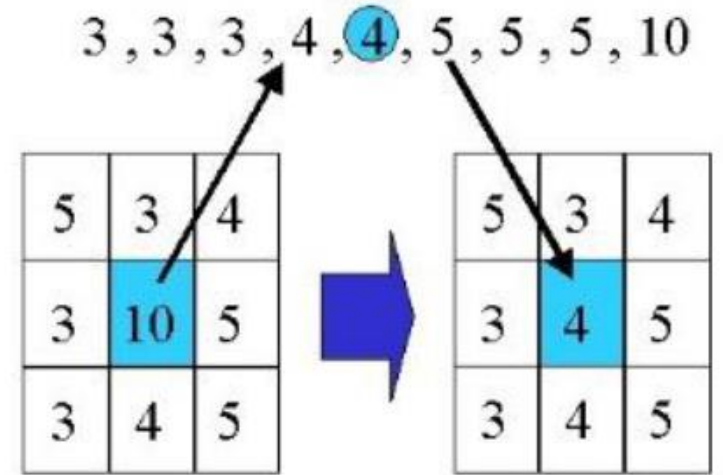
# 1. Noise reduction

## Noise reduction

- Median filter
- Mean filter
- Gaussianblur
- ...

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

1	1	1
1	1	1
1	1	1


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

1	2	1
2	4	2
1	2	1

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

# 1. Noise reduction

## Dilemma:

- Large filter → remove noise 😊
- Large filter → remove edges 😞
- Small filter → keep edges 😊
- Small filter → keep noise 😞

IMAGE WITH SALT AND PEPPER NOISE



IMAGE AFTER MEDIAN FILTERING. WINDOW SIZE 3x3



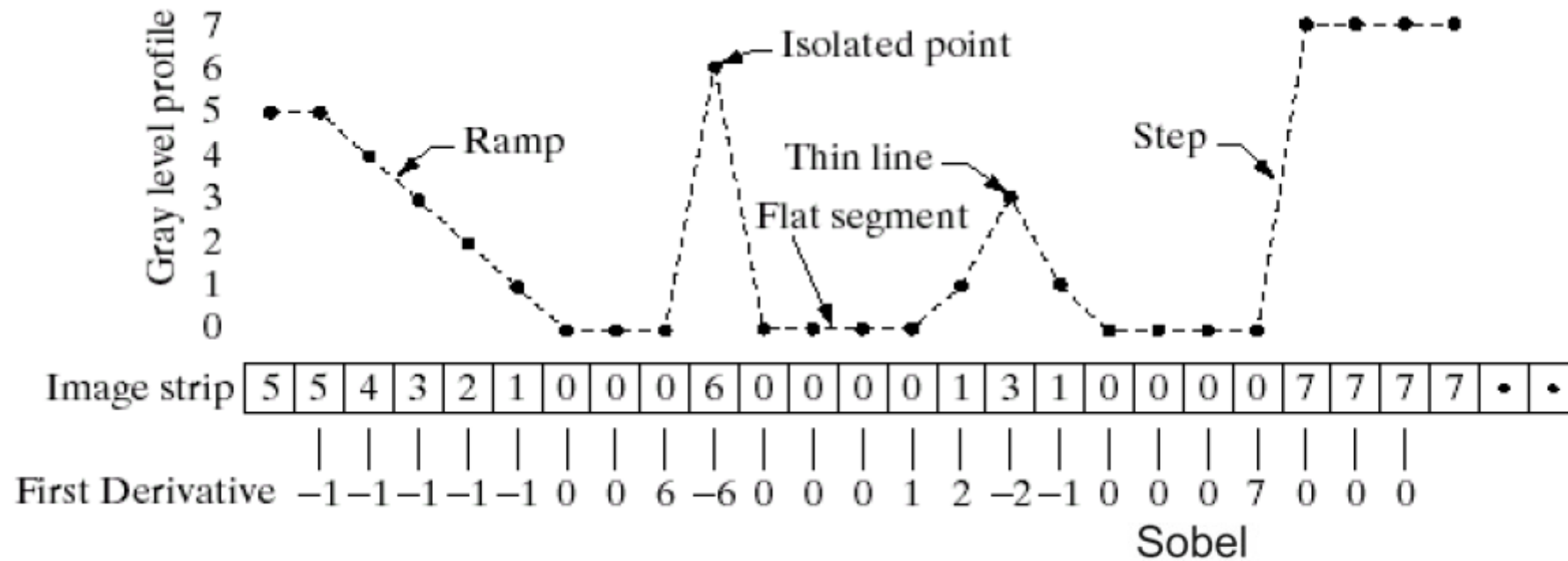
IMAGE AFTER MEDIAN FILTERING. WINDOW SIZE 5x5



IMAGE AFTER MEDIAN FILTERING. WINDOW SIZE 7x7



## 2. Edge enhancement



1D Sobel Kernel

-1	0	+1
----	---	----

Vertical			Horizontal		
-1	0	1	-1	-2	-1
-2	0	2	0	0	0
-1	0	1	1	2	1

## 2. Edge enhancement

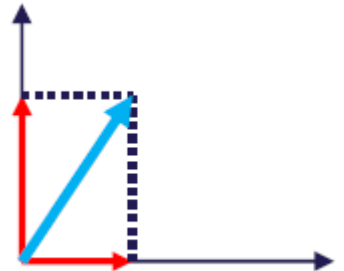
$$g_x(x, y) \approx f(x+1, y) - f(x-1, y)$$

$$g_y(x, y) \approx f(x, y+1) - f(x, y-1)$$

- Gradient vector:  $\vec{g} = [g_x, g_y]$

- Magnitude (how significant is the edge = length of blue arrow):

$$g_m = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$$



# 3. Edge location

## Thresholding the gradient magnitude

- If the magnitude:  $g(x,y) > \text{threshold} \rightarrow \text{edge found: } I(x,y) = 255$
- Else: no edge:  $I(x,y) = 0$



Input image



Horizontal Sobel



Vertical sobel



Combined Sobel



Threshold value 25



Threshold value 60

# Sobel conclusion

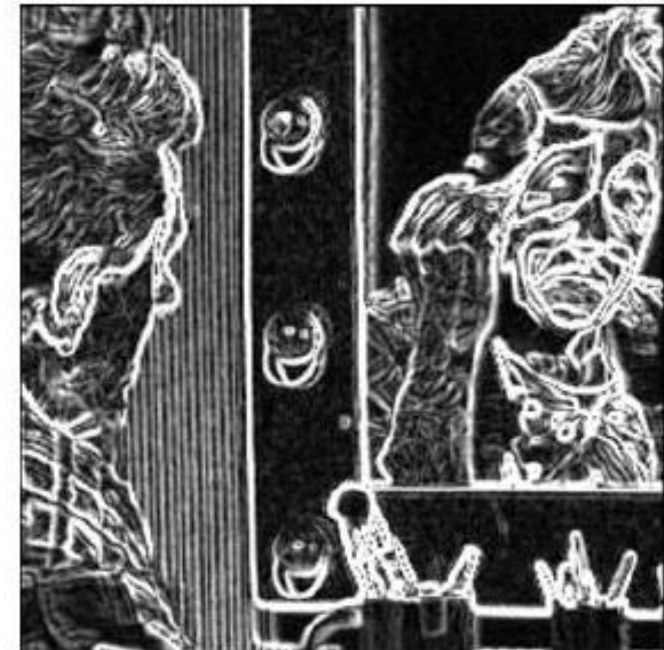
## Pros:

- Simple to understand
- Simple to implement
- Fast

## Cons:

- May produce wide edges due to its sensitivity to abrupt changes in intensity.  
This sensitivity can lead to the detection of wide or thick edges, especially when applied to images with significant intensity variations.

Other edge detectors: Canny, Prewitt Operator, Scharr Operator ....



# Normalize filter response

- To avoid overflow, we must normalize.
  - i.e. Make sure that the max value stays within our memory bounds.

Filter coefficients:

A	B	C
---	---	---

Max pixel values:

...	255	255	255	...
-----	-----	-----	-----	-----

- Max filter response =

$$A \cdot 255 + B \cdot 255 + C \cdot 255 \Leftrightarrow 255 \cdot (A + B + C)$$

- Normalized filter response =
- Filter response / (A + B + C)