UNIVERSITÄT BONN

Juergen Gall

Linear Filtering

MA-INF 2201 - Computer Vision

WS24/25

Organization

- Lecturer:
  - Prof. Dr. Juergen Gall
  - http://gall.cv-uni-bonn.de/
- Teaching:

Prof. Dr. Juergen Gall

Research Interests

Job Offers/Theses

Publications

Software

Data

Projects

Conferences

Teaching

Talks

Awards

Teaching

PhD Seminar - Graphics, Vision, Audio for Intelligent Systems, SS13, WS13/14, SS14, WS14/15, SS15, WS15/16, SS16, WS16/17, SS17, WS17/18, SS18, WS18/19, SS19, WS19/20

PhD Machine Learning Seminar, Dates

BA-INF 062 - Begleitseminar zur Bachelorarbeit: Computer Vision, SS14, WS14/15, SS15, WS15/16, SS16, WS16/17, SS17, WS17/18, SS18, WS18/19, SS19, WS19/20, SS20, WS20/21, SS21, WS21/22, SS22, WS22/23, SS23, WS23/24

MA-INF 2201 - Computer Vision, WS13/14, WS14/15, WS15/16, WS16/17, WS17/18, WS18/19, WS19/20, WS21/22, WS22/23, WS23/24

MA-INF 2213 - Advanced Computer Vision (Computer Vision II), SS14, SS15, SS16, SS17, SS18, SS19, SS20, SS21, SS22, SS23

MA-INF 2218 - Video Analytics, SS17, SS18, SS19, SS21, SS22, SS23

Organization

- Slides:
  http://gall.cv-uni-bonn.de/teaching/Lectures/cv23.html

MA-INF 2201 - Computer Vision

4L + 2E, WS23/24

Lecturer

Juergen Gall

Content

Tentative: Linear filters, Edges, Derivatives, Hough Transform, Segmentation, Graph Cuts, Mean Shift
Background Subtraction, Temporal Filtering, Active Appearance Models, Shapes, Optical Flow, 2D Tra
Estimation, Articulated Pose Estimation, Deformable Meshes, RGBD Vision

Material

The slides and recordings are available at Slides/Recordings.

Prerequisites

Basic knowledge of linear algebra, analysis, probability theory, Python programming

Organization

UNIVERSITÄT BONN

- Slides via sciebo (https://hochschulcloud.nrw/en/):

- Password: MA-INF2201

Organization

- Structure: 4+2 SWS
- Lecture:
  - Tuesday, 10:15-11:45, HSZ / HS 3
  - Friday, 10:15-11:45, HSZ / HS 3
  - Lectures will be recorded
- Exercise:
  - Start: 16.10.
  - Wednesday, 12:15-13:45, 2.025, Informatikzentrum
  - Wednesday, 14:15-15:45, 2.025, Informatikzentrum

Overview – Part 1

- Image
- Image processing
  - Filtering
  - Edges

Overview – Part 1

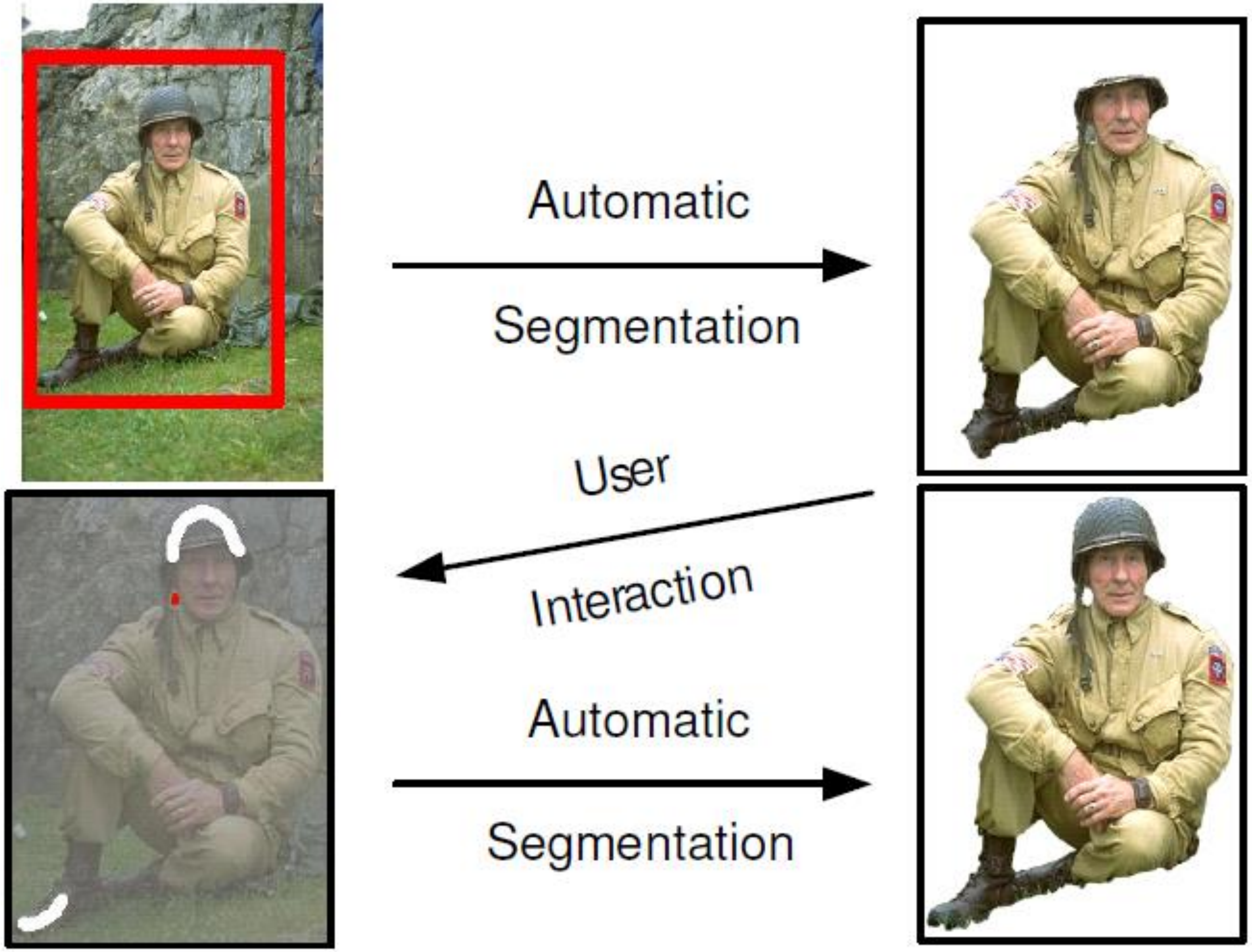
- Detecting objects with single template

Overview – Part 1

- Track objects

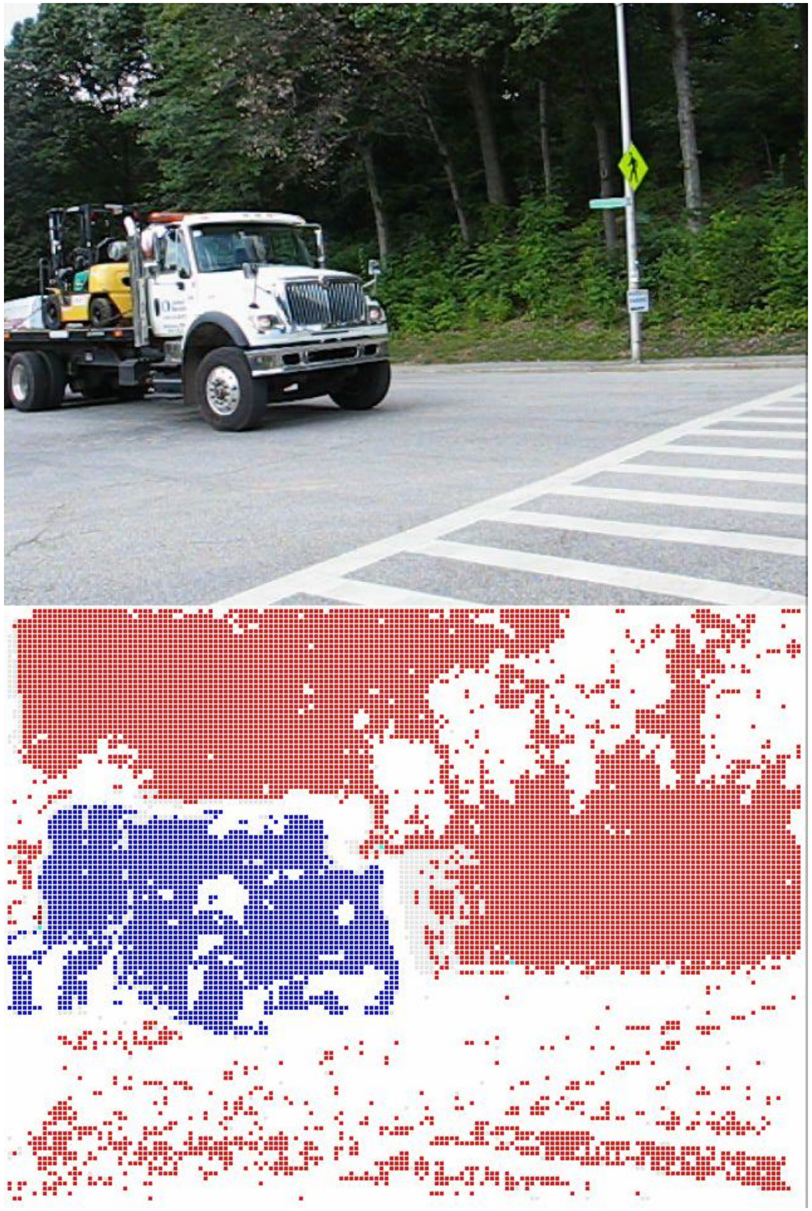
Overview – Part 1

- Segment objects



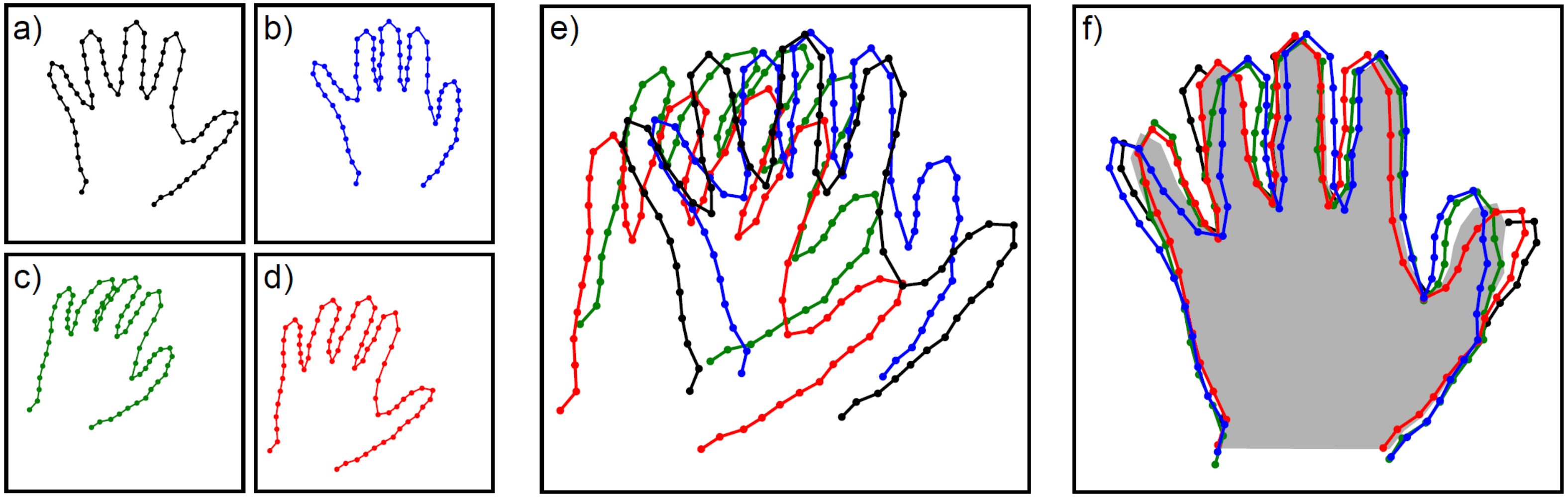
Overview – Part 1

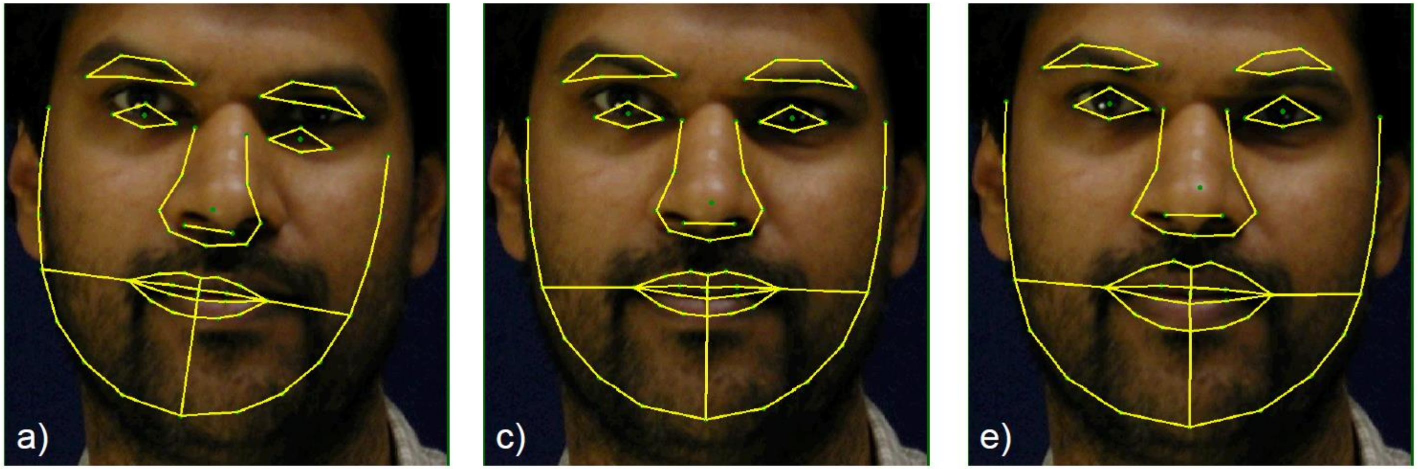
- Motion segmentation
- Optical Flow



Overview – Part 1

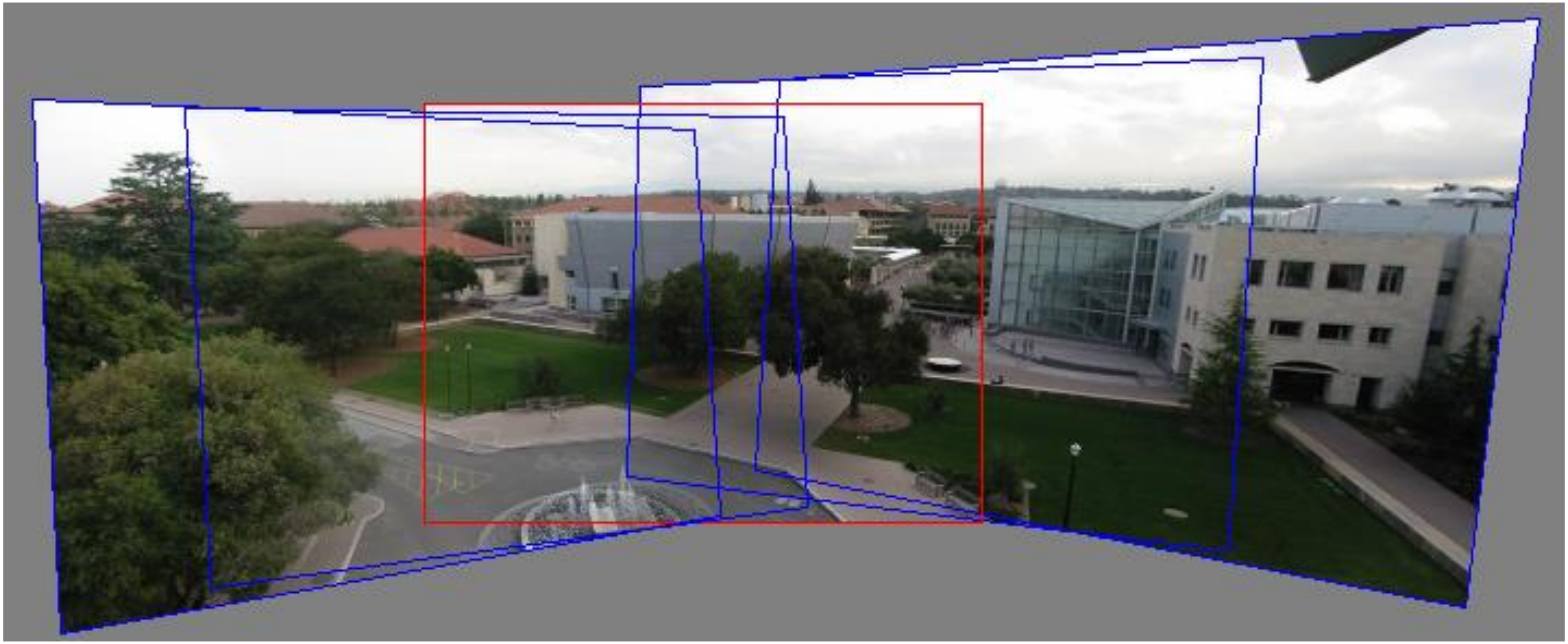
- Statistical shape models





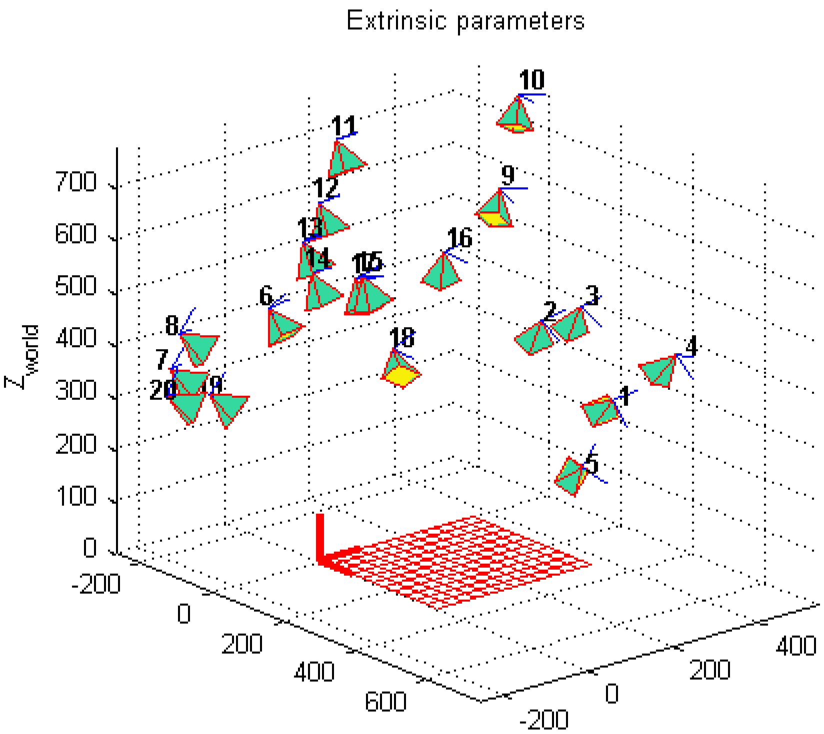
Overview – Part 1

- Align images / Panorama views



Overview – Part 2

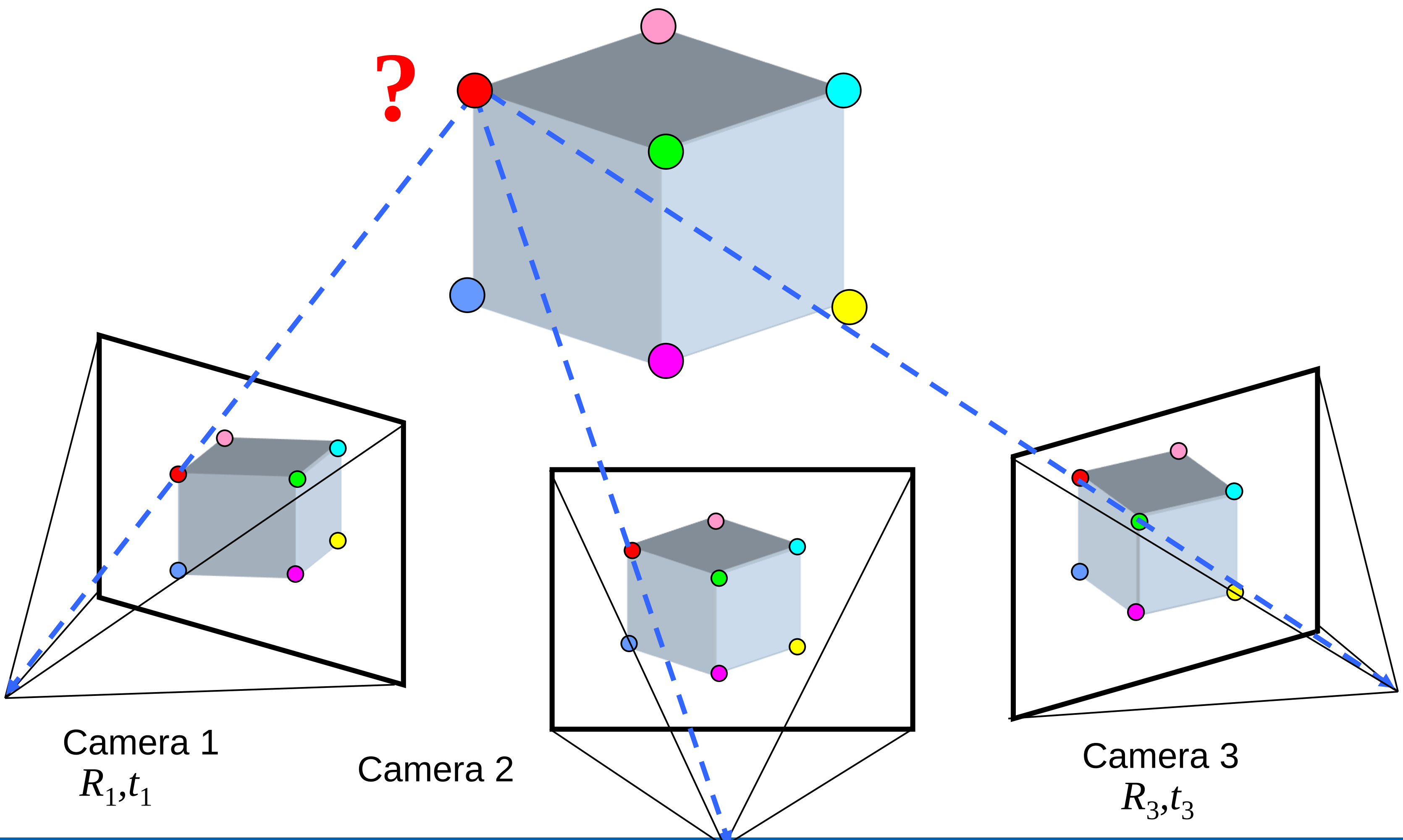
- Cameras
- Camera calibration



Overview – Part 2

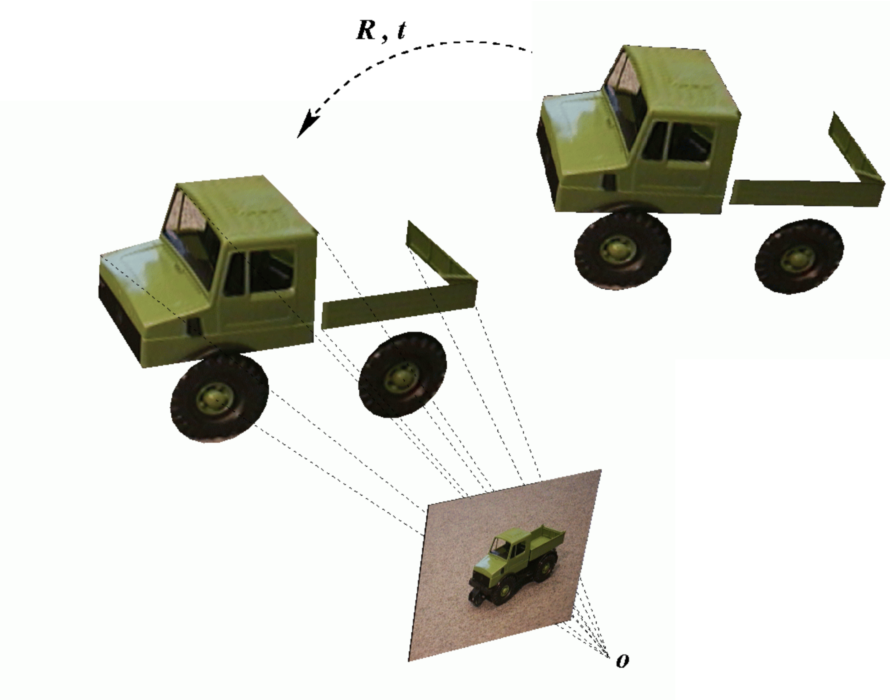
UNIVERSITÄT BONN

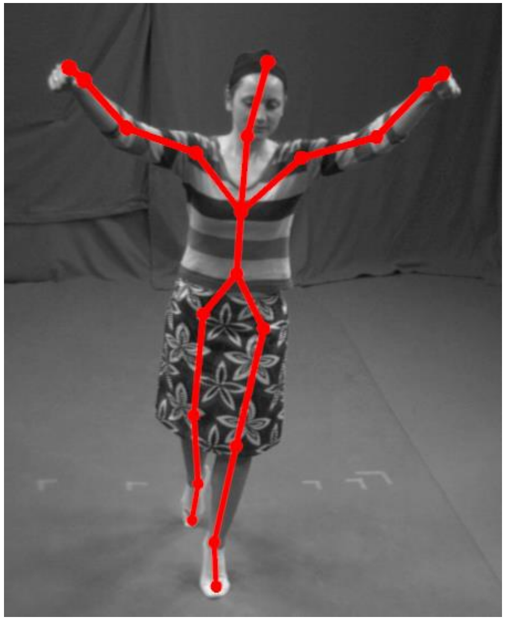
- 3D Reconstruction / Stereo / Structure from motion

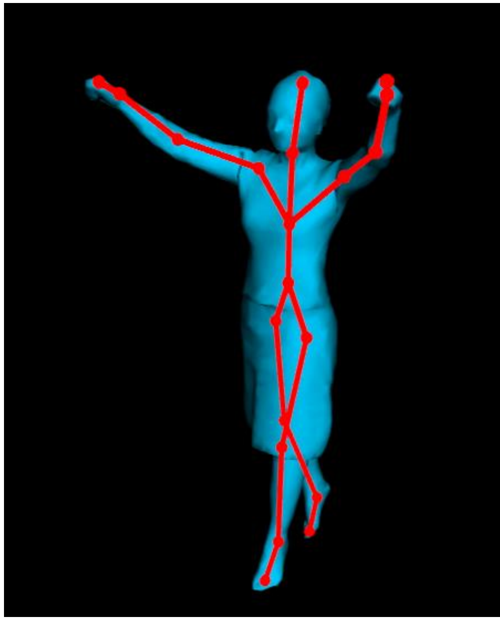


Overview – Part 2

- Pose estimation

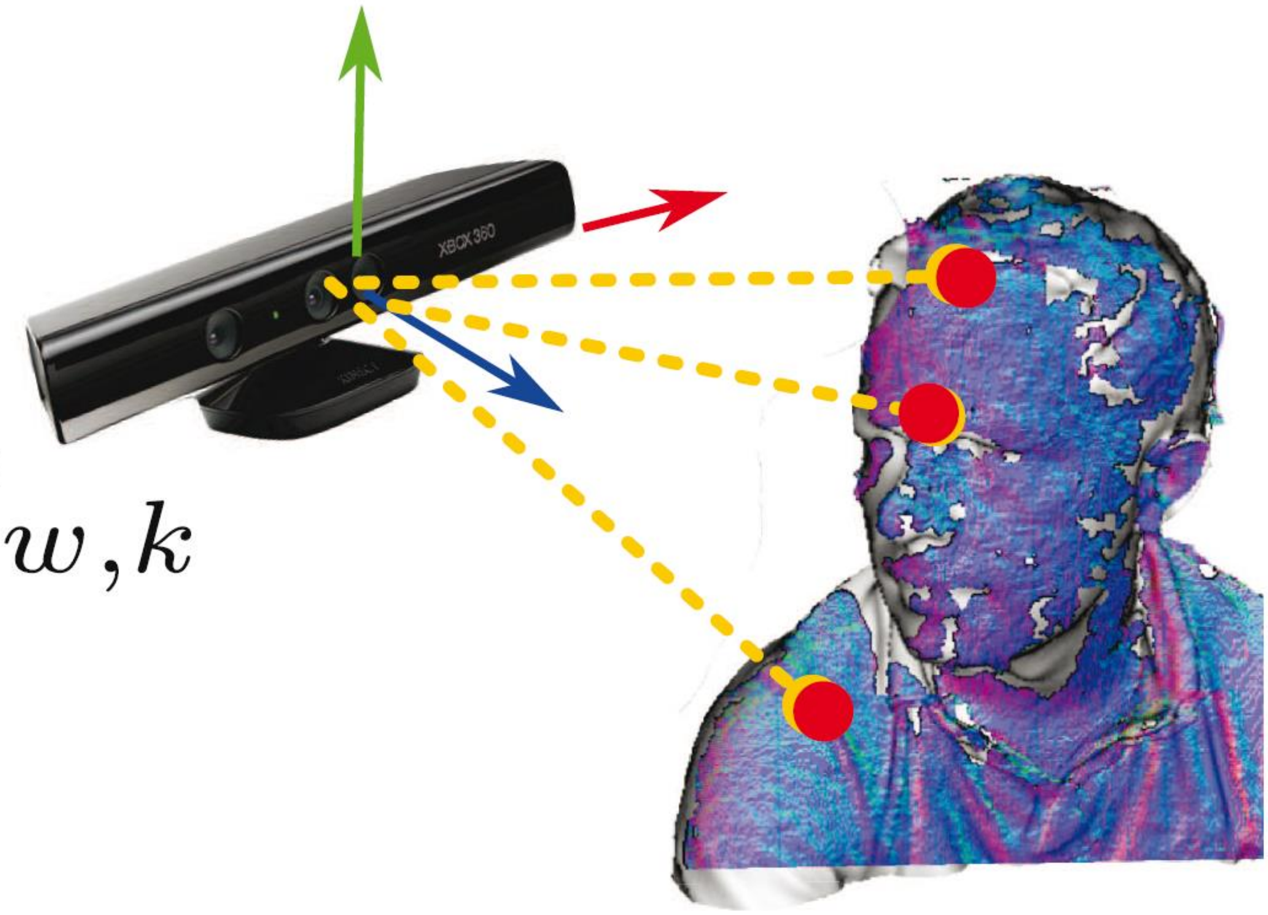






Overview – Part 2

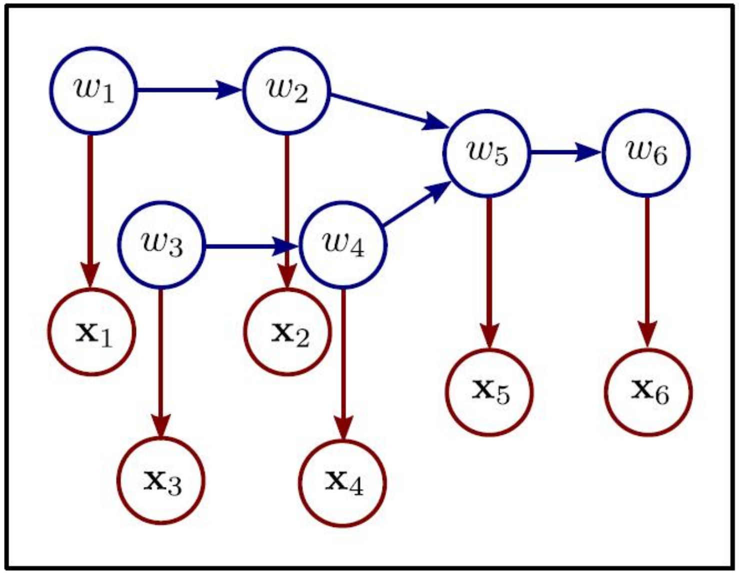
- Depth sensors

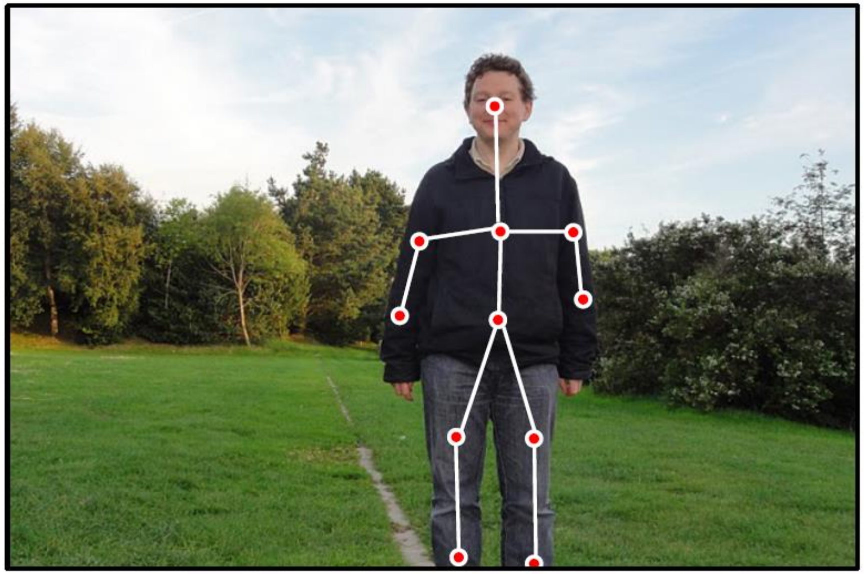


$$\mathbf{T}_{w,k}$$

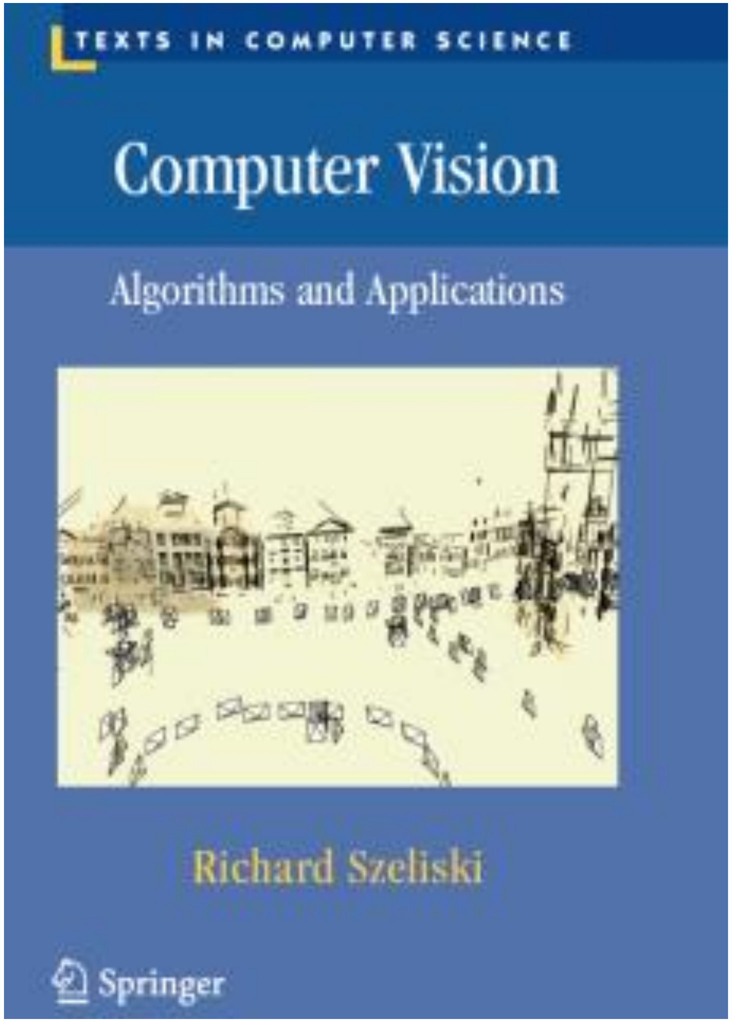
Overview – Bonus

- Graphical models



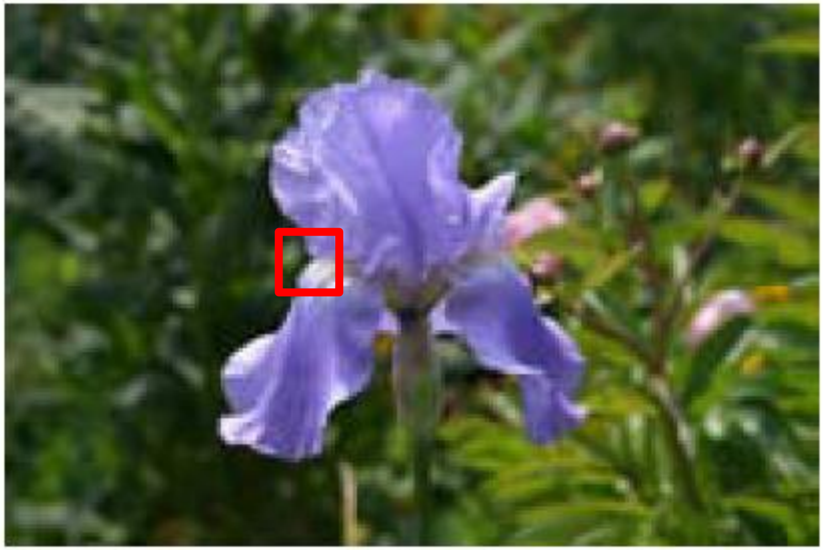


Literature



Chapter 2 and 3. Computer Vision - Algorithms and Applications, Szeliski, Richard, Springer, 2011

Image as function



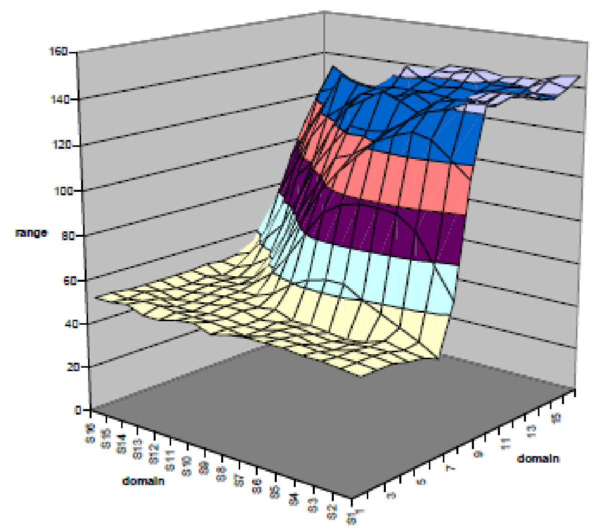
45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

Mapping from image domain (pixels) to values

Gray image: $f : \Omega \to \mathrm{R}$

Color images: $f : \Omega \to \mathrm{R}^3$

Image represented as matrix or tensor

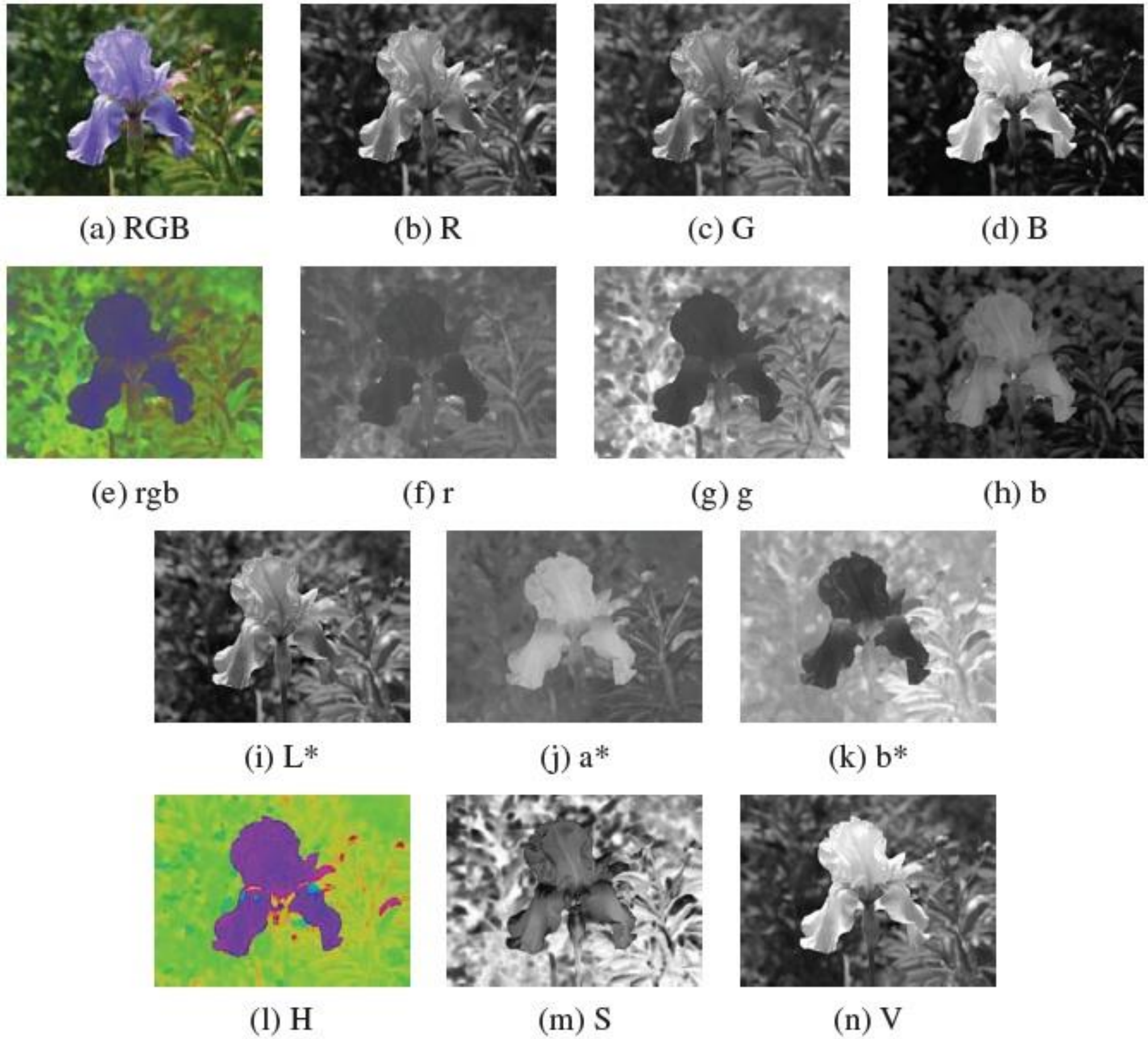


Color Spaces

UNIVERSITÄT BONN

There are many color spaces

Popular: Intensity, RGB, Lab, HSV



Pixel transforms

Image operator h:

$$g(\boldsymbol{x}) = h(f(\boldsymbol{x}))$$

$$g(i,j) = h(f(i,j)) \quad \boldsymbol{x} = (i,j)$$

Blend two images linearly

$$g(\boldsymbol{x}) = (1-\alpha)f_0(\boldsymbol{x}) + \alpha f_1(\boldsymbol{x})$$

Gamma correction:

$$g(\boldsymbol{x}) = [f(\boldsymbol{x})]^{1/\gamma}$$

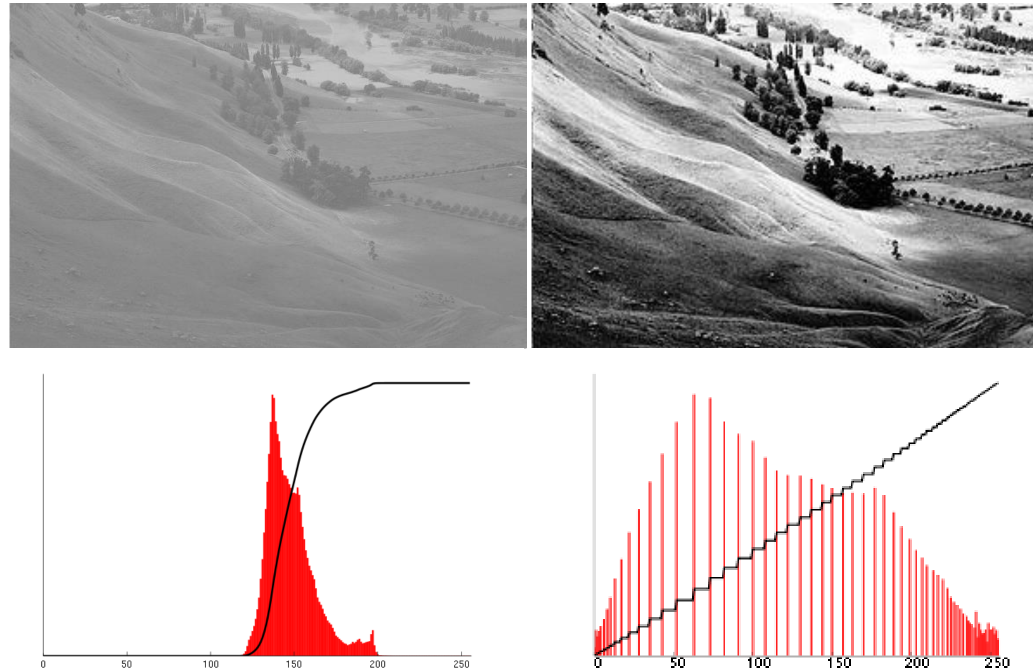
Histogram equalization

Pixels as distribution:

$$p_f(y) = \frac{|\{x \in \Omega : f(x) = y\}|}{|\Omega|}$$

$$F_f(y) = \int_0^y p_f(y')dy' = \sum_{y'=0}^{y} p_f(y')$$

Cumulative distribution function



Make CDF linear:

$$h(f(x)) = F(f(x)) \cdot 255$$

$$p_h(z) = \frac{|\{x \in \Omega : h(f(x)) = z\}|}{|\Omega|} = \frac{1}{255}$$

Proof

Probability theory:

$$F_h(z) = \int_0^z p_h(z')dz' = \int_0^{h^{-1}(z)} p_f(y')dy' = F_f(h^{-1}(z))$$

$$y = h^{-1}(z)$$

We get:

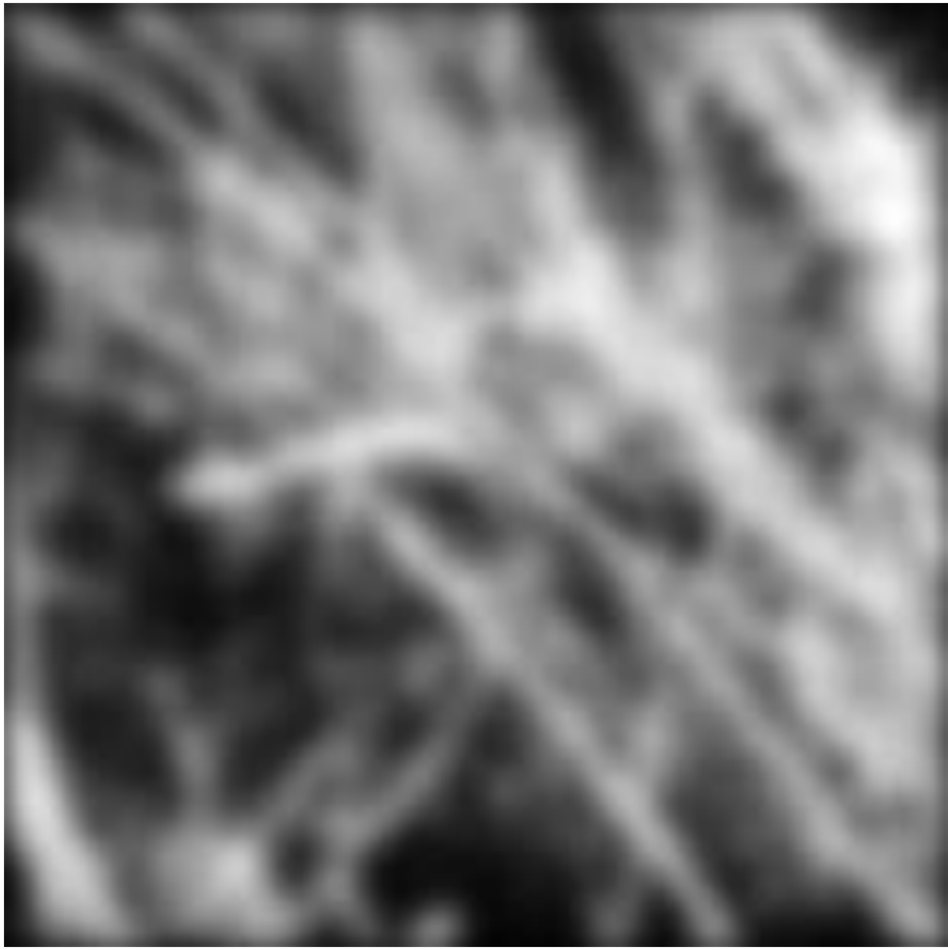
$$p_h(z) = \frac{d}{dz}F_h(z) = \frac{d}{dz}F_f(h^{-1}(z)) = p_f(h^{-1}(z))\frac{d}{dz}h^{-1}(z)$$

Since $h(f(x)) = F(f(x)) \cdot 255$ and $\frac{d}{dy}h(y) = 255 \cdot p_f(y)$:

$$p_f(h^{-1}(z))\frac{d}{dz}h^{-1}(z) = \frac{1}{255} \cdot \underbrace{\frac{d}{dy}h(y)\Big|_{y=h^{-1}(z)} \cdot \frac{d}{dz}h^{-1}(z)}_{\frac{d}{dz}h(h^{-1}(z)) = 1}$$

$$p_h(z) = \frac{1}{255}$$

Linear filtering



Source: S. Lazebnik

Motivation: Image denoising

How can we reduce noise in a photograph?

Source: S. Lazebnik

Moving average

Let's replace each pixel with a weighted average of its neighborhood

The weights are called the filter kernel

What are the weights for the average of a 3x3 neighborhood?

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

"box filter"

Source: D. Lowe

Defining convolution

Let f be the image and h be the kernel. The output of convolving f with h is denoted: $g = f * h$

$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l) = \sum_{k,l} f(k,l)h(i-k, j-l)$$

$f(x,y)$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$*$

$h(x,y)$

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$=$

$g(x,y)$

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

Convolution

Continuous convolution:

$$g(\boldsymbol{x}) = \int f(\boldsymbol{x} - \boldsymbol{u}) h(\boldsymbol{u}) d\boldsymbol{u}$$

h is called impulse response function:

$$h * \delta = h \qquad \delta(x) = \begin{cases} \infty & \text{if} \quad x{=}0 \\ 0 & \text{otherwise} \end{cases}$$

Proof:

$$g(x) = \int \delta(x - u) h(u) du = h(x)$$

Key properties

Linearity: filter(f1 + f2) = filter(f1) + filter(f2)

Shift invariance: same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))

Theoretical result: any linear shift-invariant operator can be represented as a convolution

Source: S. Lazebnik

Proof

UNIVERSITÄT BONN

Linear shift-invariant operator T as a convolution

Using $f(i,j) = \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} f(p,q)\delta(i-p, j-q)$ with $\delta(p,q) = \begin{cases} 1 & \text{if} \quad p=0 \text{ and } q=0 \\ 0 & \text{otherwise} \end{cases}$
we get:

$$(T \circ f)(i,j) = T\left\{ \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} f(p,q)\delta(i-p, j-q) \right\}$$

$$= \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} f(p,q)(T \circ \delta)(i-p, j-q)$$ linear

$$= \sum_{p=-\infty}^{\infty} \sum_{q=-\infty}^{\infty} f(i-p, j-q)(T \circ \delta)(p,q)$$ shift-invariant

$$= (f * (T \circ \delta))(i,j)$$

Properties in more detail

Commutative: a * b = b * a

- Conceptually no difference between filter and signal

Associative: a * (b * c) = (a * b) * c

- Often apply several filters one after another: (((a * b1) * b2) * b3)
- This is equivalent to applying one filter: a * (b1 * b2 * b3)

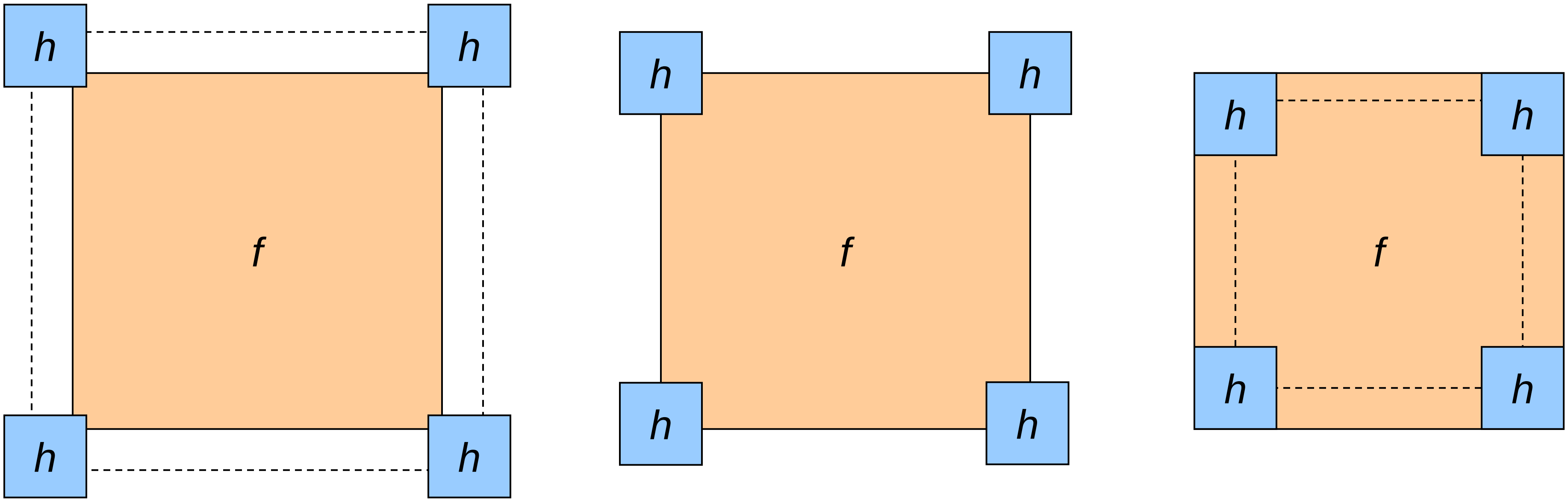
Distributes over addition: a * (b + c) = (a * b) + (a * c)

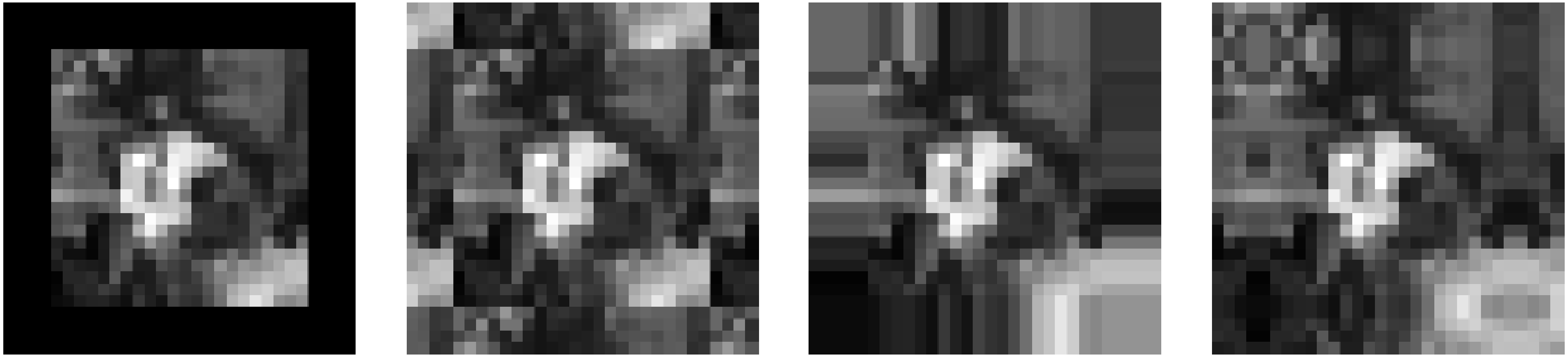
Scalars factor out: ka * b = a * kb = k (a * b)

Identity: unit impulse e = […, 0, 0, 1, 0, 0, …],
a * e = a

Source: S. Lazebnik

Border padding and output size



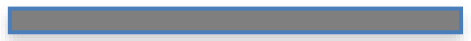


zero　　wrap　　clamp　　mirror

Rectangular filter

Source: A. Torralba

Rectangular filter



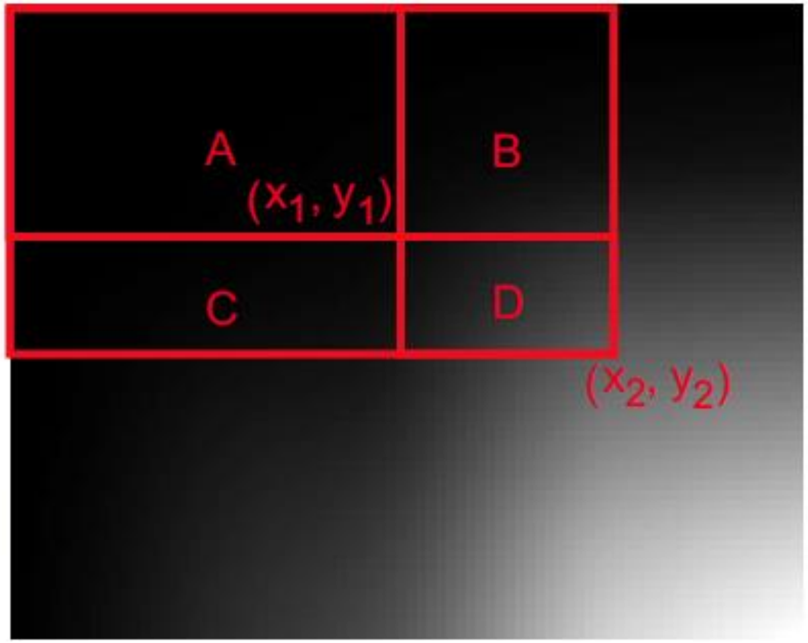
Source: A. Torralba

Rectangular filter

Source: A. Torralba

Integral image

Precompute average



Get average values of any size by reading only 4 values!

Source: A. Torralba

Integral image

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

Original image

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

Integral image

$$s(i,j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k,l)$$

Recursive:

$$s(i,j) = s(i-1,j) + s(i,j-1) - s(i-1,j-1) + f(i,j)$$

17+19-11+3 = 28

Integral image

3	2	7	2	3
1	5	1	3	4
5	1	**3**	5	1
4	3	2	1	6
2	4	1	4	8

Original image

3	5	12	14	17
4	*11*	**19**	24	31
9	**17**	28	38	46
13	24	37	48	62
15	30	44	59	81

Integral image

3	5	12	*14*	17
4	11	19	24	31
9	17	28	38	46
13	24	37	**48**	62
15	30	44	59	81

Integral image

4 values independent of size:

$$s(i_1, j_1) - s(i_1, j_0 - 1) - s(i_0 - 1, j_1) + s(i_0 - 1, j_0 - 1)$$

48-13-14+3 = 24 = 5+1+3+1+3+5+3+2+1

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
1	0	0
0	0	0

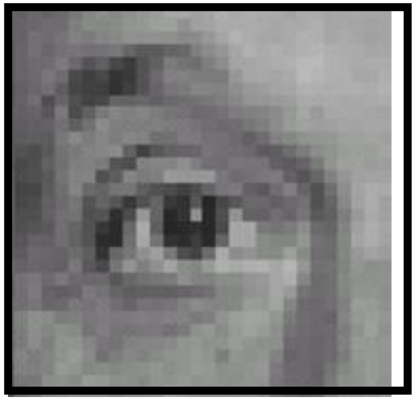
?

Source: D. Lowe

Practice with linear filters

Original

0	0	0
1	0	0
0	0	0



Shifted *left*
By 1 pixel

Source: D. Lowe

Practice with linear filters

Original

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

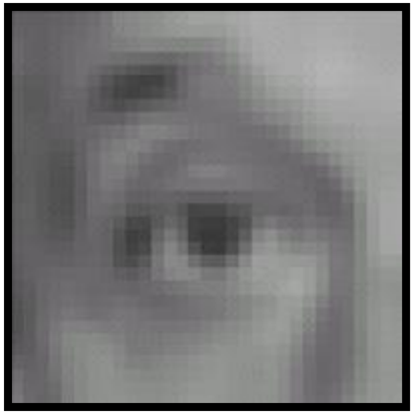
?

Source: D. Lowe

Practice with linear filters

Original

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



Blur (with a
box filter)

Source: D. Lowe

Practice with linear filters

Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

UNIVERSITÄT BONN

Original

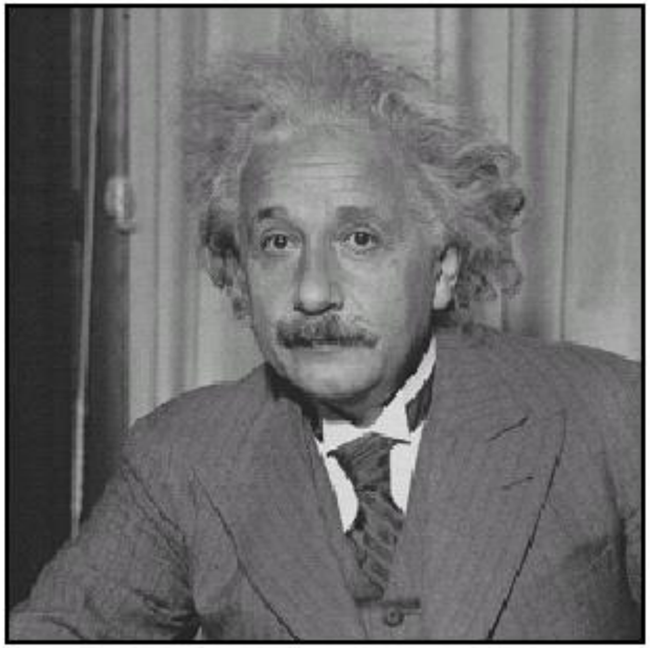
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

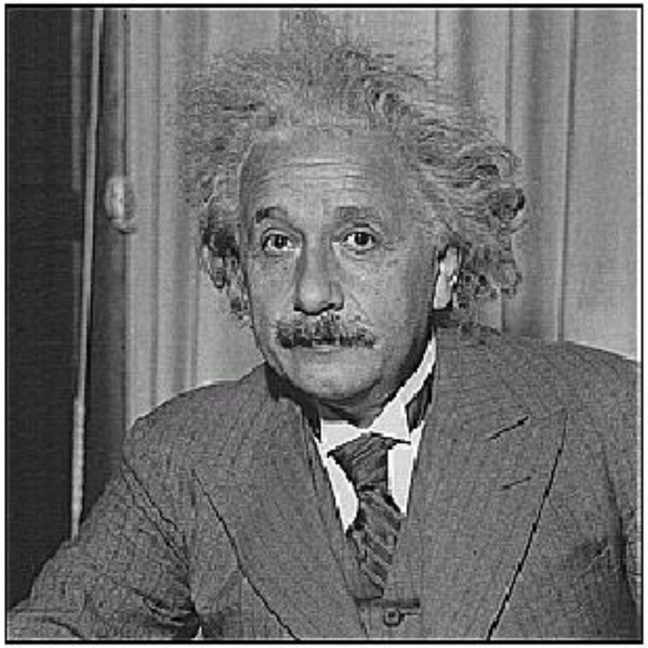
- Accentuates differences with local average

Source: D. Lowe

Sharpening



before



after

Source: D. Lowe

Sharpening

What does blurring take away?

−

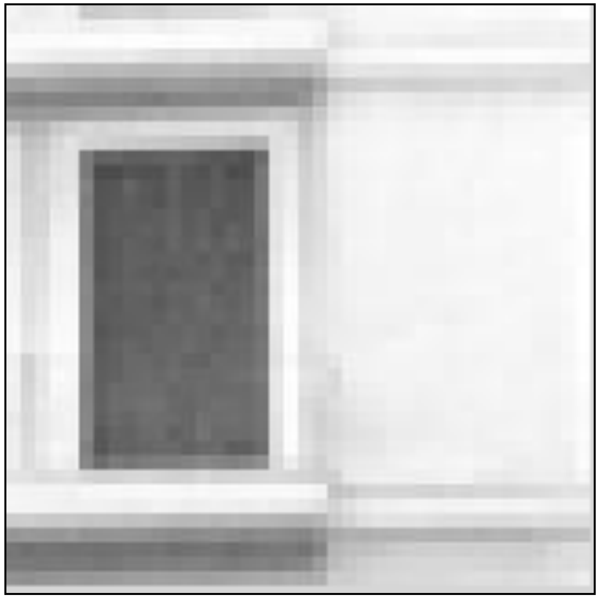
=

Let’s add it back:

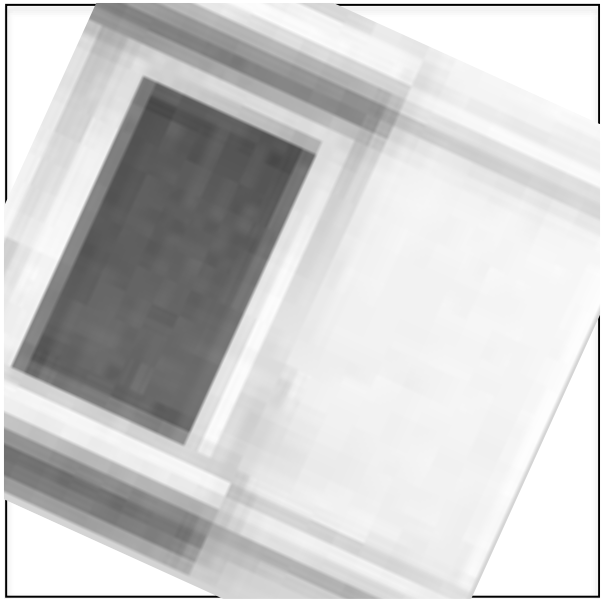
+

=

Image rotation



?

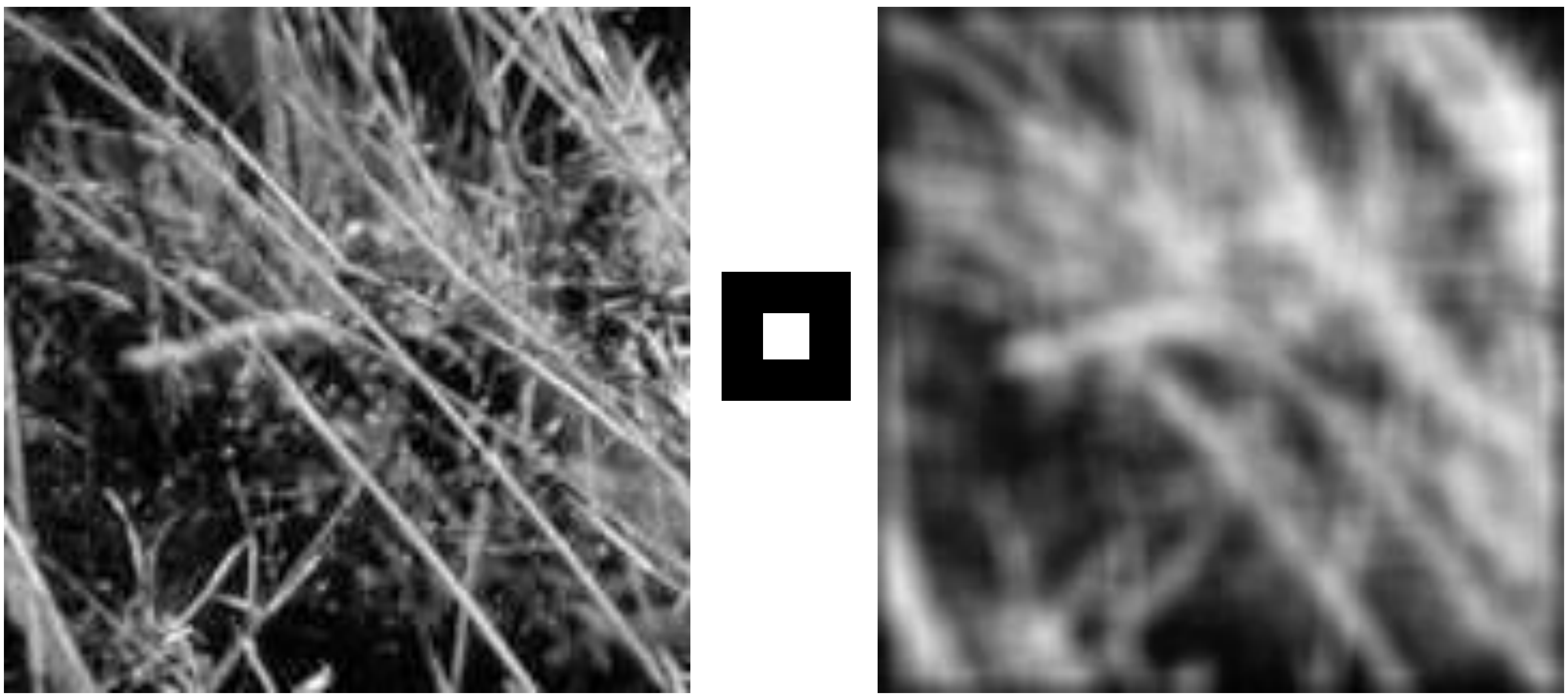


It is linear, but not a spatially invariant operation. There is no convolution.

Source: A. Torralba

Smoothing with box filter revisited

What's wrong with this picture?

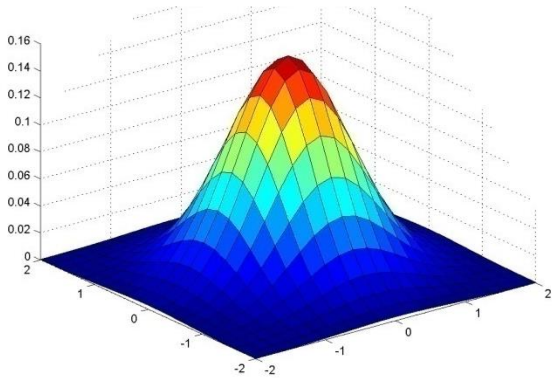


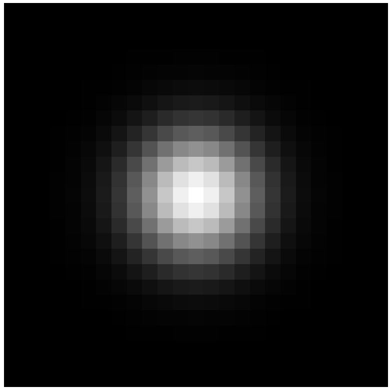
Source: D. Forsyth

Gaussian Kernel

Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

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





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

5 x 5, σ = 1

Source: C. Rasmussen

Choosing kernel width

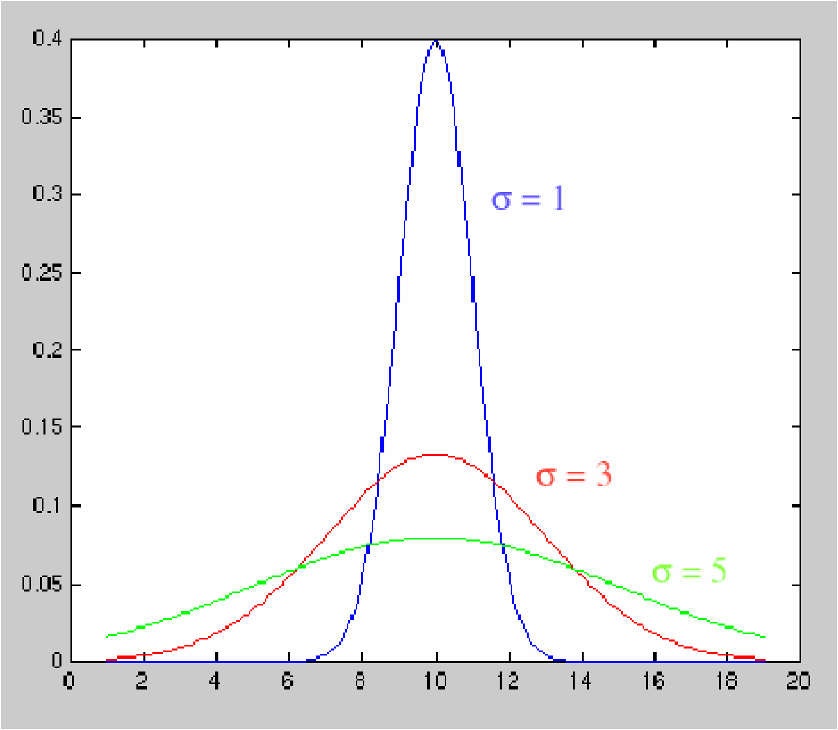
Rule of thumb: set filter half-width to about 3σ

$$P(\mu - \sigma < X \leq \mu + \sigma) \approx 68\%$$

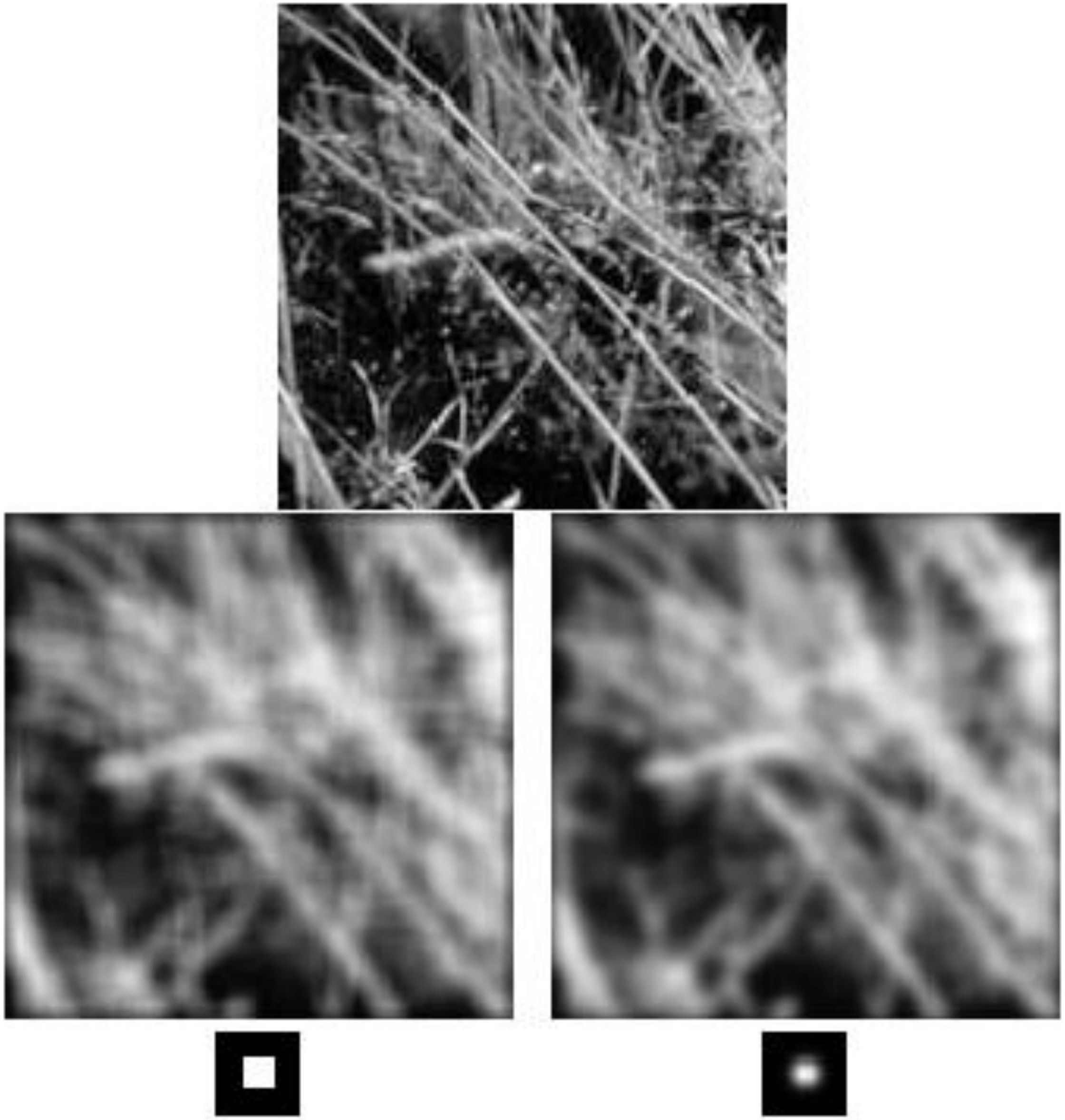
$$P(\mu - 2\sigma < X \leq \mu + 2\sigma) \approx 95.5\%$$

$$P(\mu - 3\sigma < X \leq \mu + 3\sigma) \approx 99.7\%$$

Effect of σ



Gaussian vs. box filtering



Source: S. Lazebnik

Gaussian filters

- Remove "high-frequency" components from the image (low-pass filter)
- Convolution with itself is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. σ is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- Separable kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D Lowe

Separability example

2D convolution
(center location only)

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array}$$

The filter factors
into a product of 1D
filters:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} = \begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array}$$

Perform convolution
along rows:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 5 & 5 \\ \hline 4 & 4 & 6 \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline & 11 & \\ \hline & 18 & \\ \hline & 18 & \\ \hline \end{array}$$

Followed by convolution
along the remaining column:

Source: K. Grauman

Why is separability useful?

What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?

– $O(n^2 m^2)$

What if the kernel is separable?

– $O(n^2 m)$

Source: S. Lazebnik

Is a kernel separable?

UNIVERSITÄT BONN

- Kernel as matrix:

$$K = \sum_i \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$$

- Singular Value Decomposition (SVD)
- If only the first singular value σ_0 is non-zero, kernel is separable
- Vertical and horizontal kernels:

$$\sqrt{\sigma_0}\boldsymbol{u}_0 \text{ and } \sqrt{\sigma_0}\boldsymbol{v}_0^T$$

- Approximation: $K \approx \sigma_0 u_0 (v_0)^T$

SVD - Definition

UNIVERSITÄT BONN

A[m x n] = U[m x r] Σ[r x r] (V[n x r])T

A: Input data matrix
- m x n matrix

U: Left singular vectors
- m x r matrix

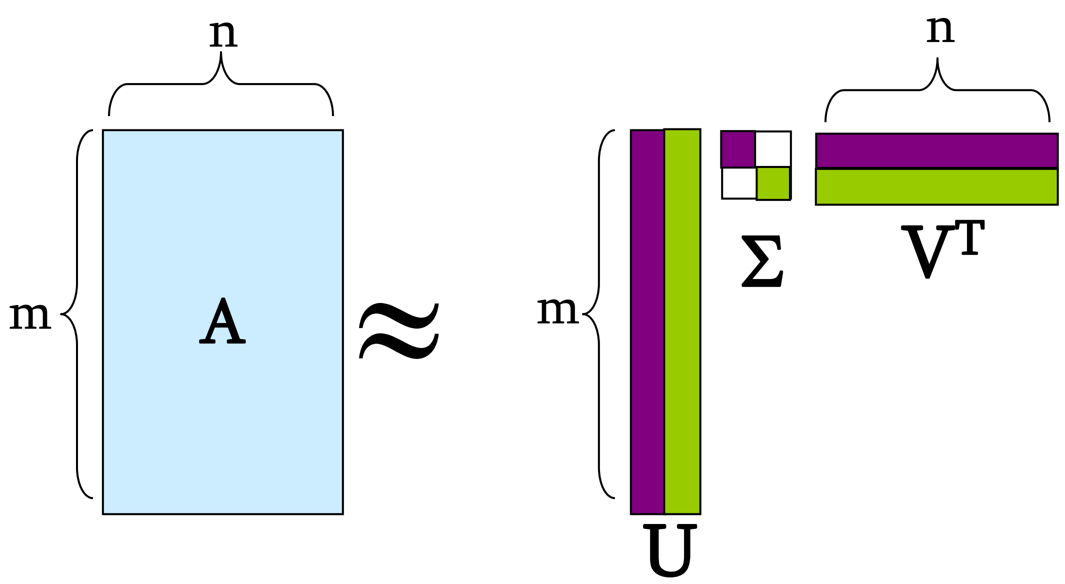
Σ: Singular values
- r x r diagonal matrix (r : rank of the matrix A)

V: Right singular vectors
- n x r matrix

Source: J Leskovec

SVD

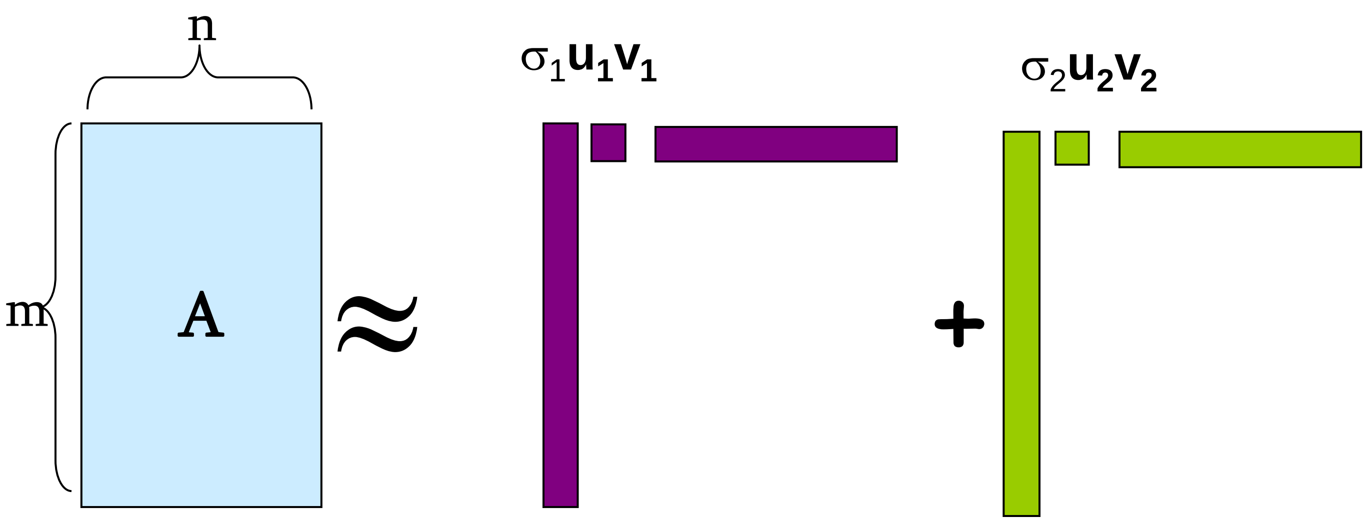
$$\mathbf{A} \approx \mathbf{U\Sigma V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^{\mathsf{T}}$$



Source: J Leskovec

SVD

$$\mathbf{A} \approx \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^{\mathsf{T}}$$



σ_i … scalar
u_i … vector
v_i … vector

Source: J Leskovec

SVD - Properties

It is always possible to decompose a real matrix A into $A = U \Sigma V^T$, where

U, Σ, V: unique

U, V: column orthonormal:

- $U^T U = I; V^T V = I$ (I: identity matrix)
- (Cols. are orthogonal unit vectors)

Σ: diagonal

- Entries (singular values) are positive, and sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq \dots \geq 0$)

Source: J Leskovec

SVD – Example

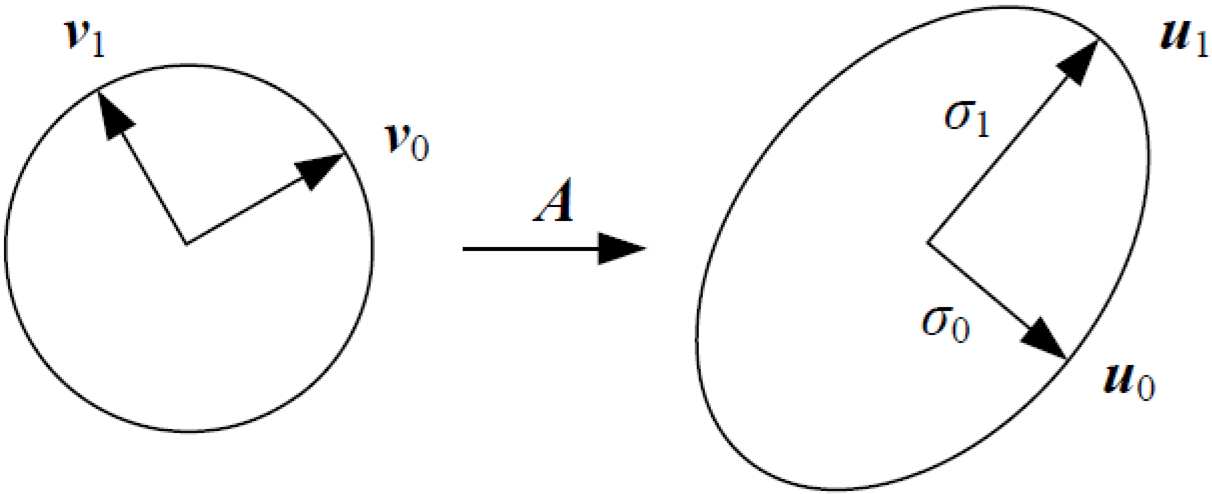
$A = U \Sigma V^T$ - example:

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

Source: J Leskovec

Geometric interpretation

Change of basis vectors:



$$\boldsymbol{A}_{M\times N} \quad = \quad \boldsymbol{U}_{M\times P}\boldsymbol{\Sigma}_{P\times P}\boldsymbol{V}^T_{P\times N}$$

$$\boldsymbol{AV} = \boldsymbol{U\Sigma} \quad \text{or} \quad \boldsymbol{A}\boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j$$

Noise

Salt and pepper noise: contains random occurrences of black and white pixels

Impulse noise: contains random occurrences of white pixels

Gaussian noise: variations in intensity drawn from a Gaussian normal distribution

Original

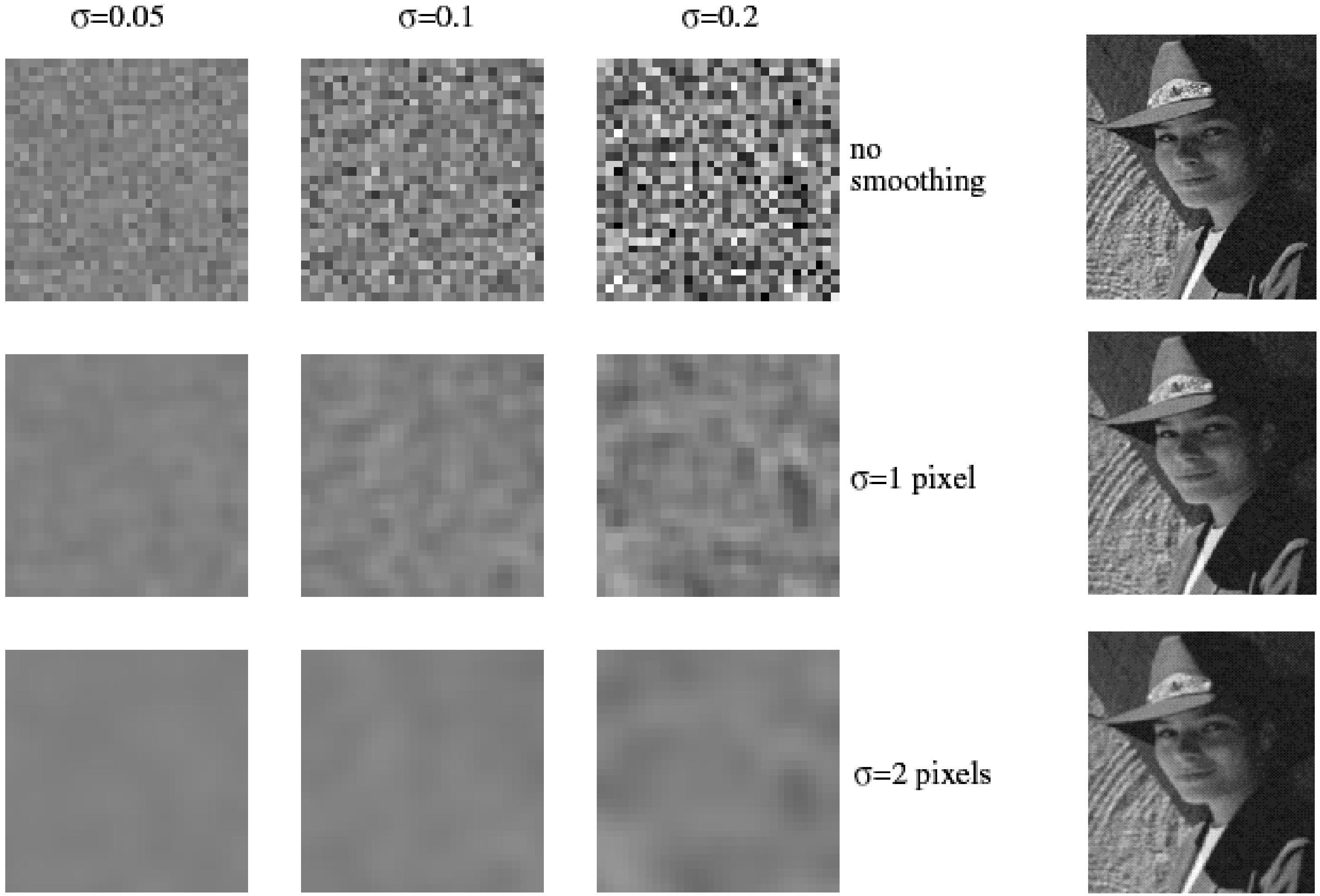
Salt and pepper noise

Impulse noise

Gaussian noise

Source: S Seitz

Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Source: S. Lazebnik

Reducing salt-and-pepper noise

What's wrong with the results?

3x3

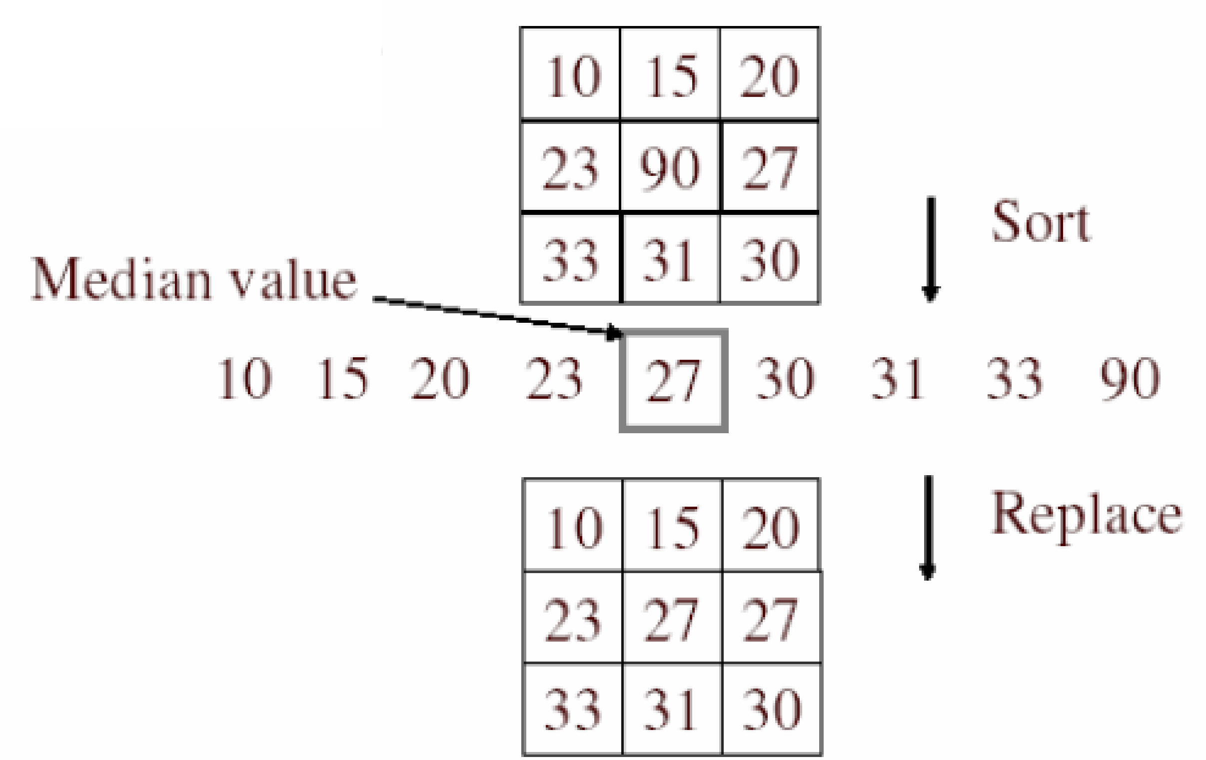
5x5

7x7

Source: S. Lazebnik

Alternative idea: Median filtering

A median filter operates over a window by selecting the median intensity in the window



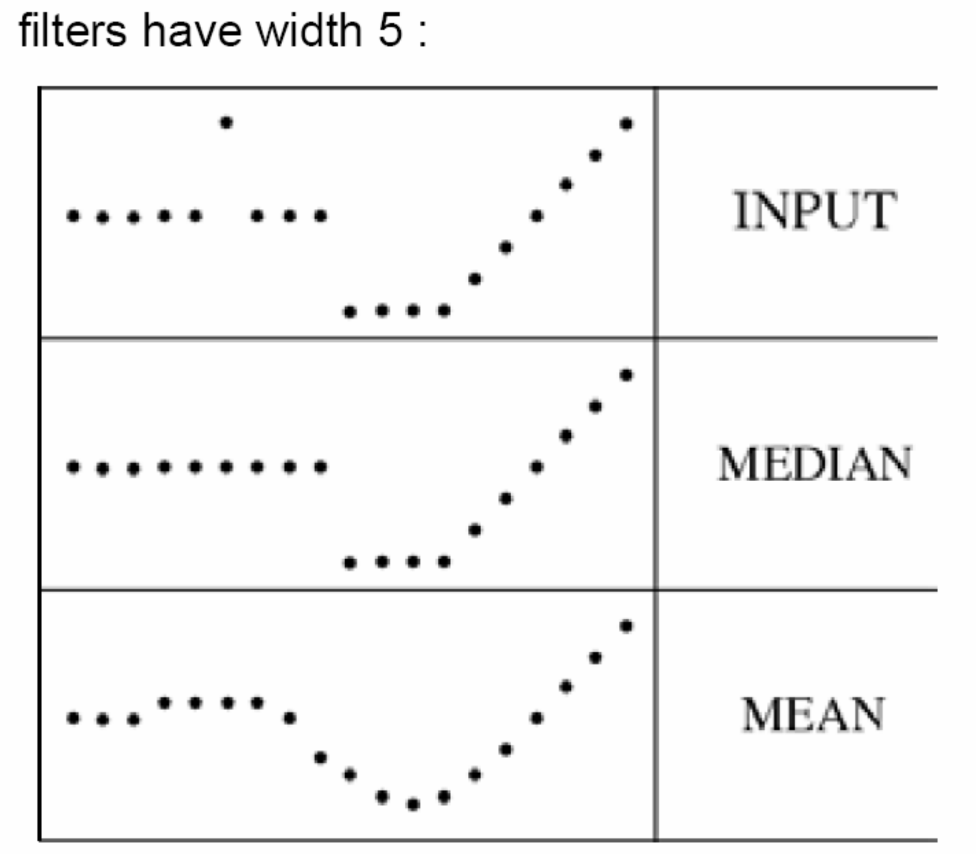
Is median filtering linear?

Source: K. Grauman

Median filter

What advantage does median filtering have over Gaussian filtering?

Robustness to outliers



Source: K. Grauman

Gaussian vs. median filtering

UNIVERSITÄT BONN

3x3 5x5 7x7

Gaussian

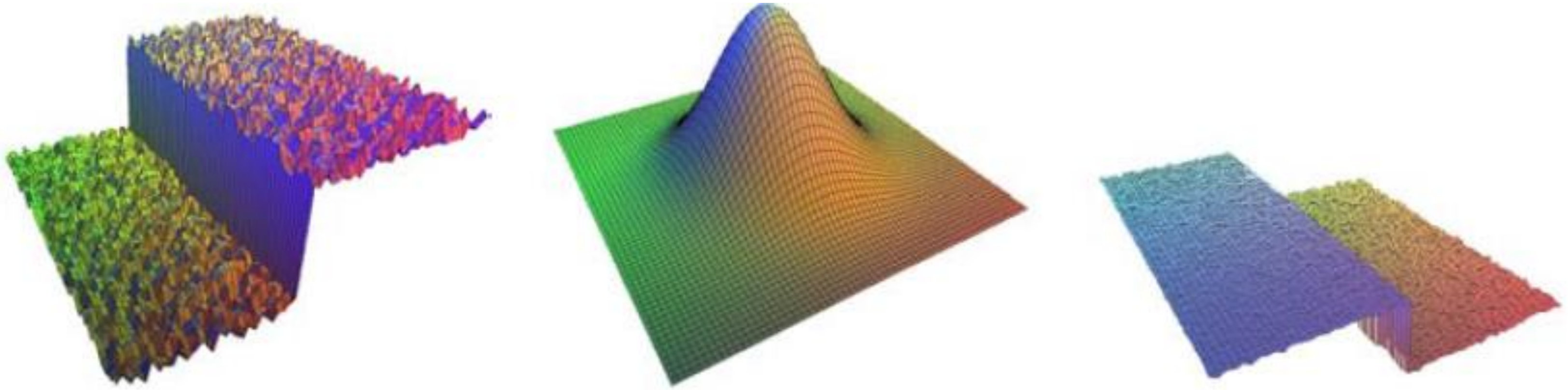
Median

Source: S. Lazebnik

Bilateral filter

Filter is data dependent

$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$



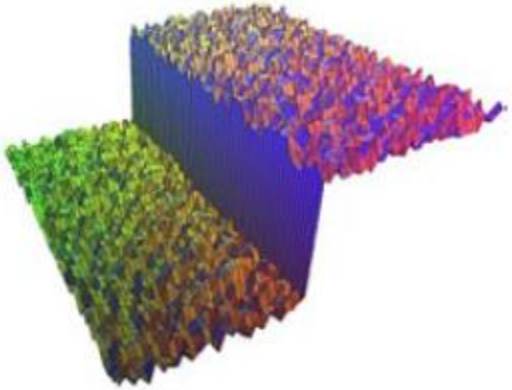
Input Domain kernel Range kernel

$$d(i,j,k,l) = \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_d^2}\right) \qquad r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right)$$

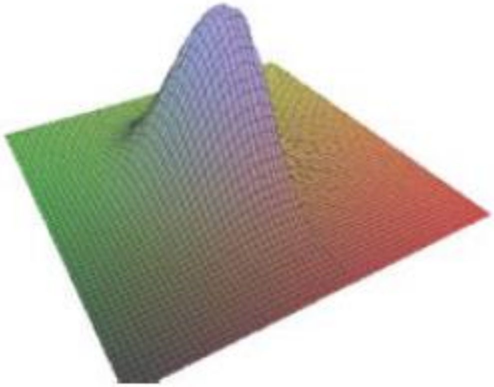
Bilateral filter

Filter is data dependent

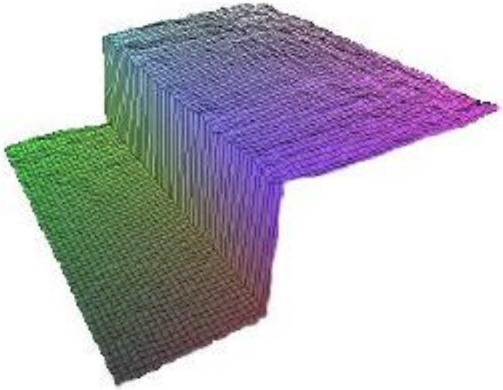
$$g(i,j) = \frac{\sum_{k,l} f(k,l)w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$



Input



Bilaterial filter



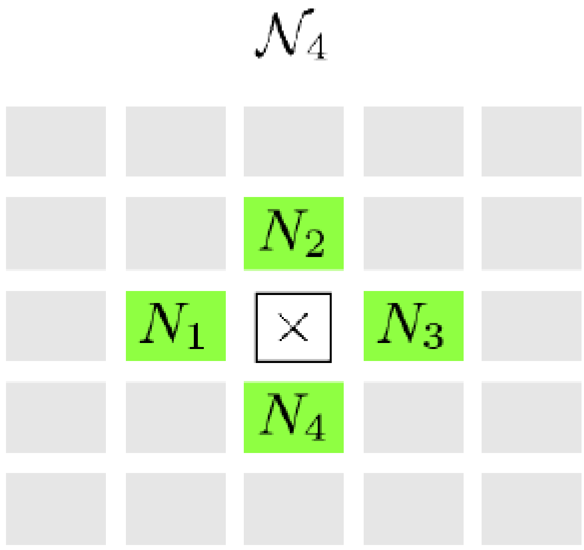
Output

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j)-f(k,l)\|^2}{2\sigma_r^2}\right)$$

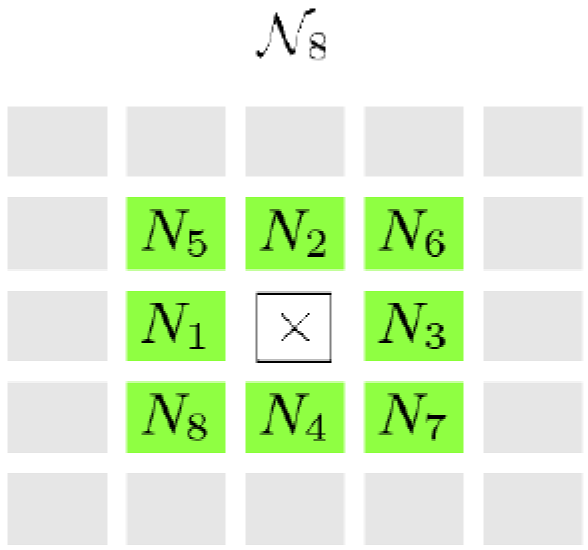
Slow, but methods for approximation exist

Binary images

Neighborhoods

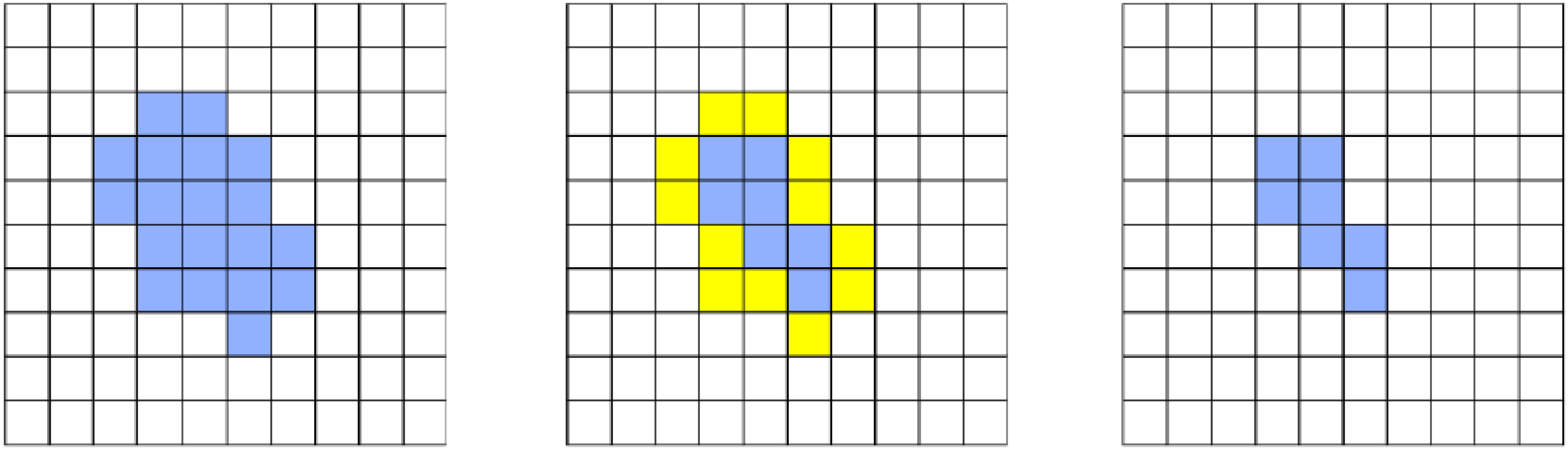


4 Neighborhood



8 Neighborhood

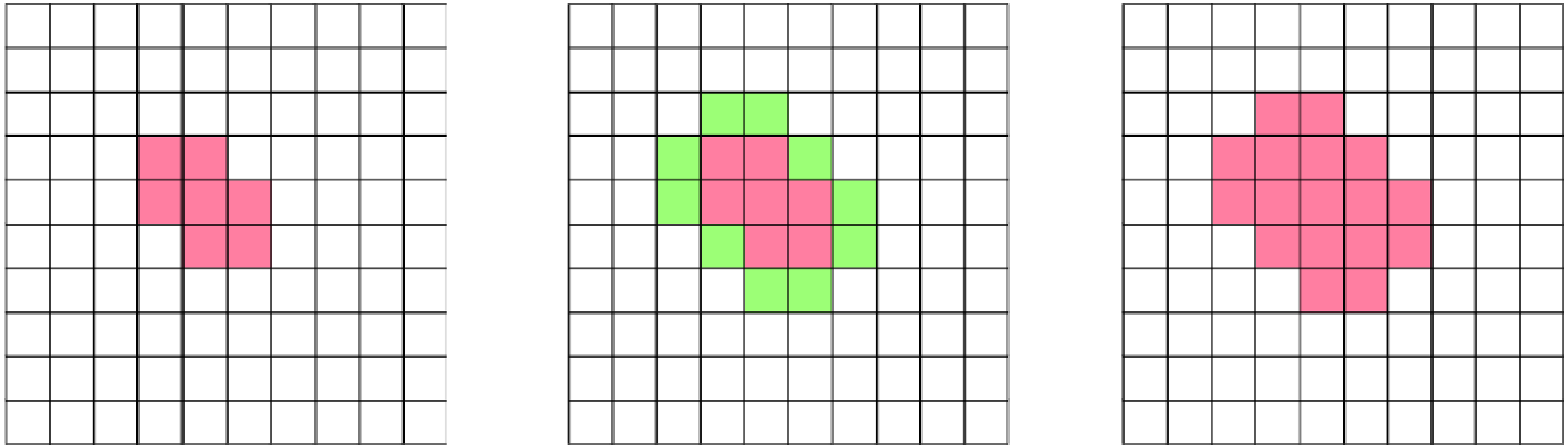
Erosion



(a) (b) (c)

Change a foreground pixel to background if it has a background pixel as a 4-neighbor.

Dilation



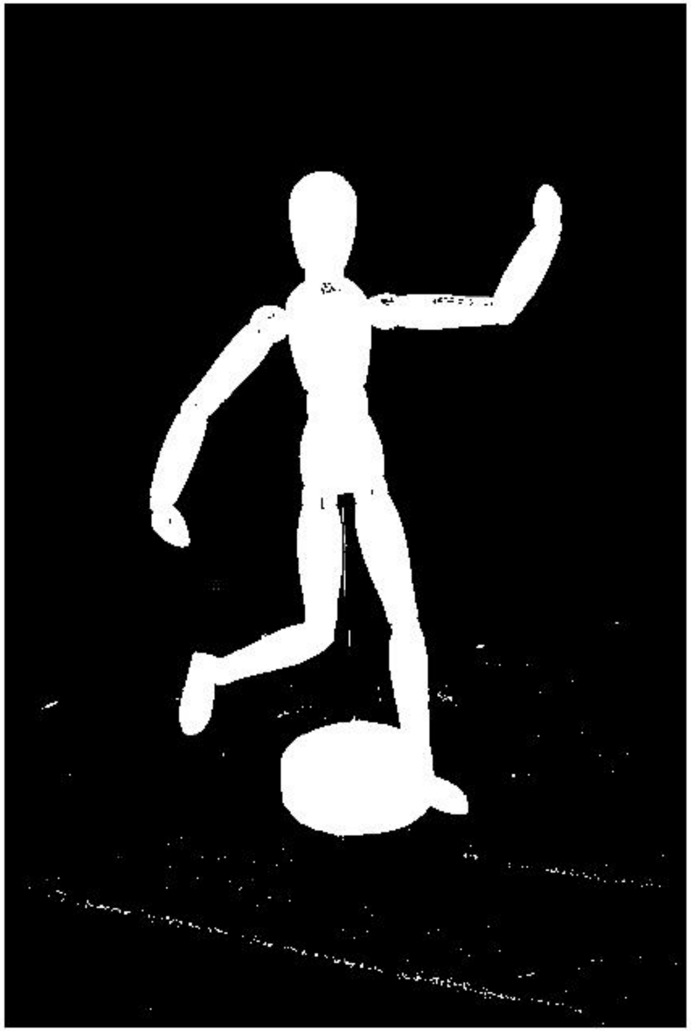
(a) (b) (c)

Change a background pixel to foreground if it has a foreground pixel as a 4-neighbor.

Threshold

UNIVERSITÄT BONN

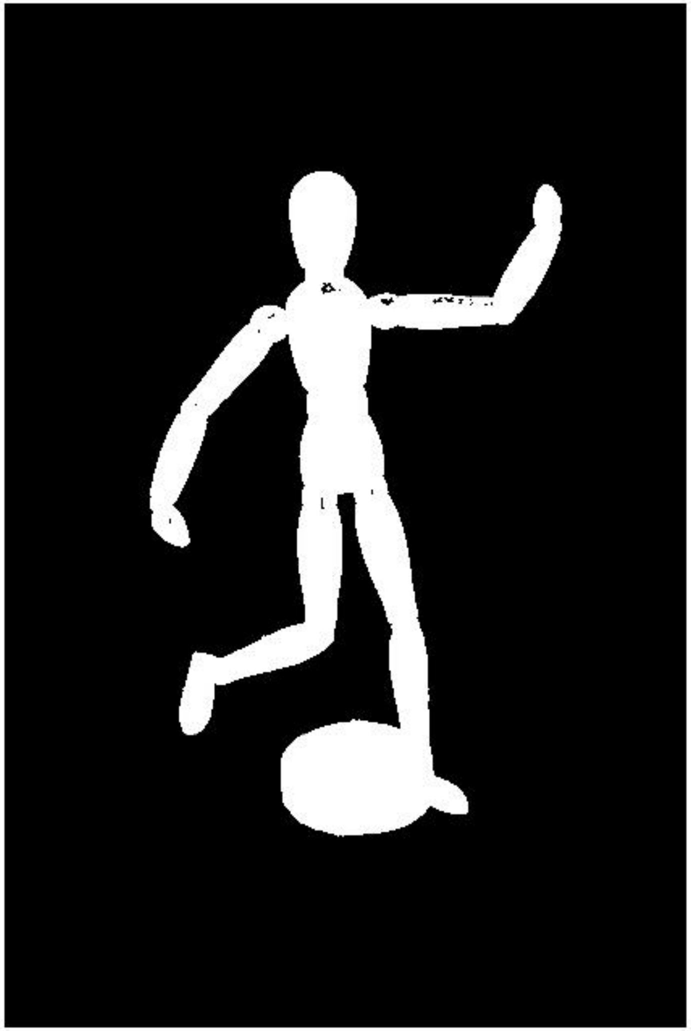
Original image



Initial threshold

Opening = Dilate(Erode)

Original image



After opening

Closing = Erode(Dilate)


Original image


After closing

Thank you for your attention.