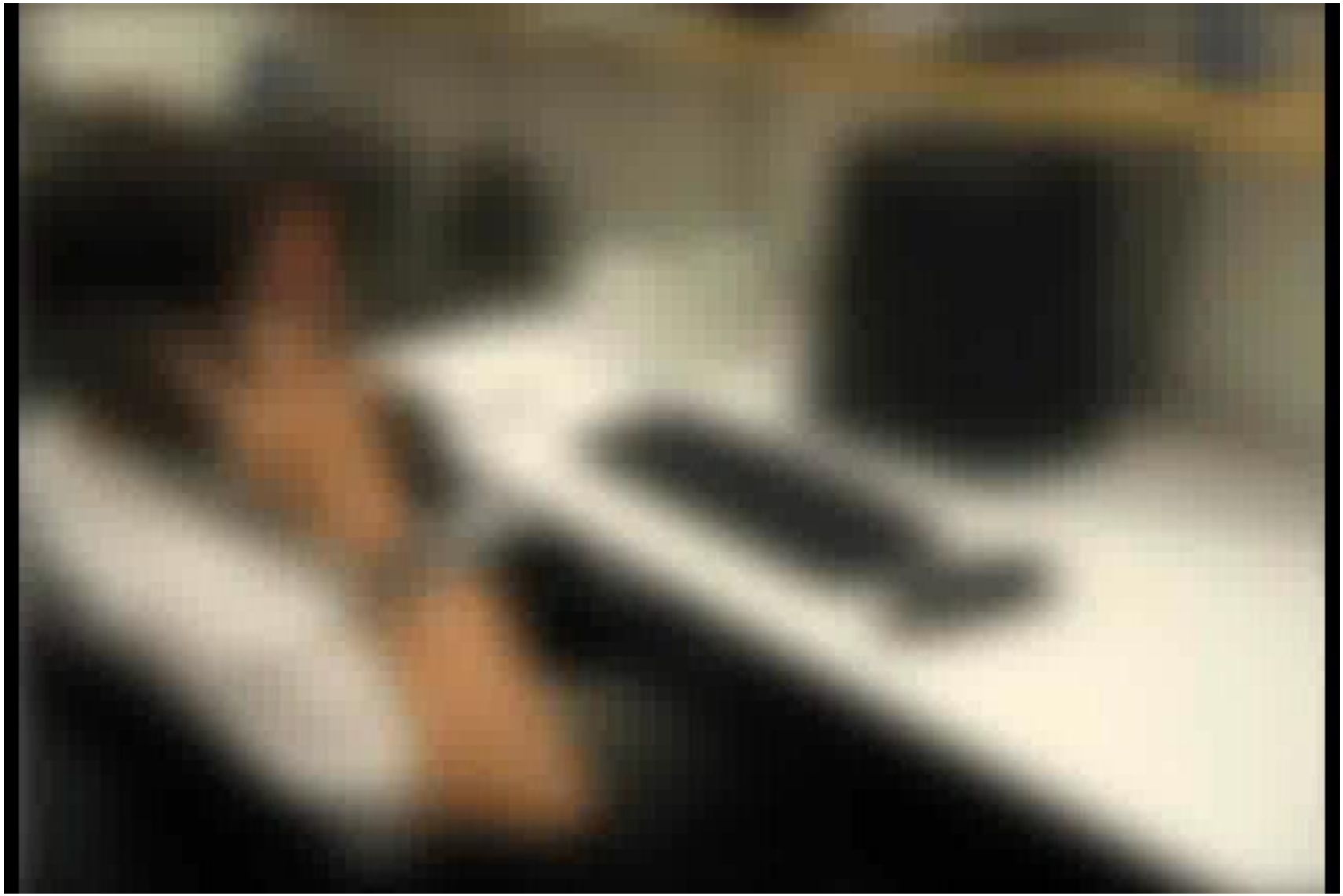
UNIVERSITÄT BONN

Juergen Gall

Recapitulation 1
MA-INF 2201 - Computer Vision
WS24/25

Context



Source: A. Torralba

Humans work with strong priors

Source: A. Torralba

Conclusion

Human vision relies not **only** on data term (pixel information)

Priors are important

They explain the **most likely** scene if data is ambiguous:

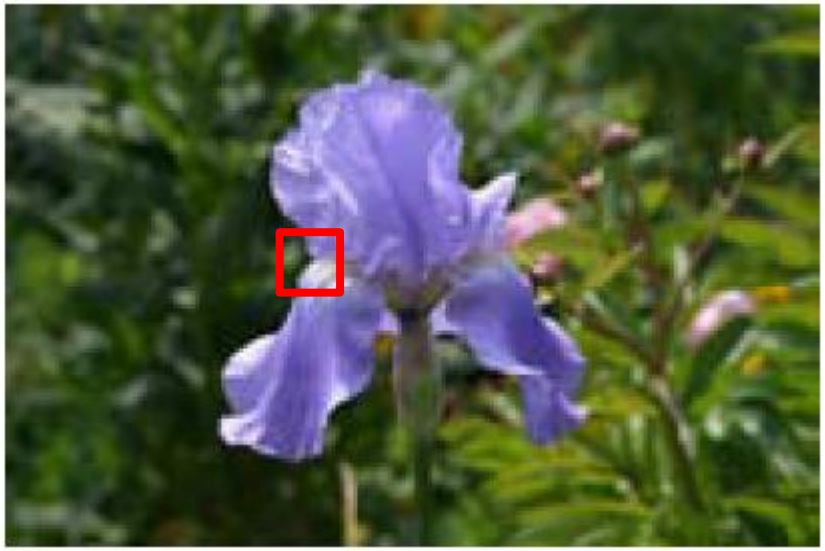
$$\underset{y}{\operatorname{argmax}}\, P(y|I) = \underset{y}{\operatorname{argmax}}\, P(I|y)P(y)$$

Bayes formula:

$$P(y|I) = \frac{P(I|y)P(y)}{P(I)}$$

Posterior = Likelihood x Prior

Image as function



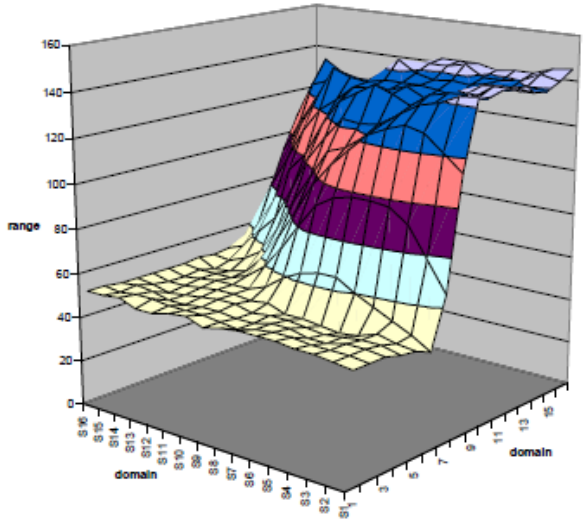
45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

Mapping from image domain (pixels) to values

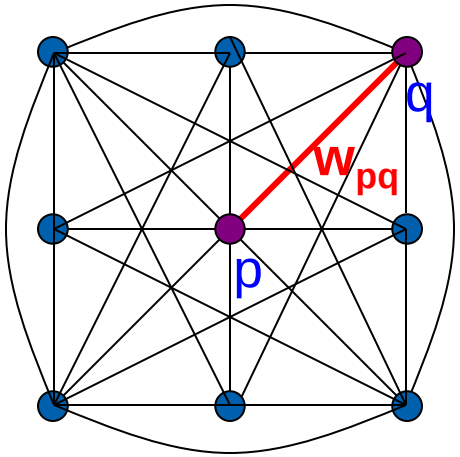
Gray image: $f : \Omega \to \mathrm{R}$

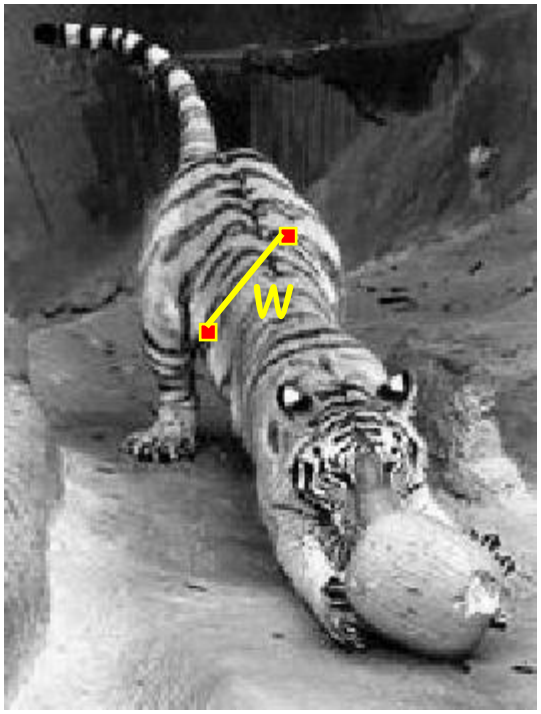
Color images: $f : \Omega \to \mathrm{R}^3$

Image represented as matrix or tensor



Images as graphs





- *Fully-connected* graph
  - node (vertex) for every pixel
  - link between *every* pair of pixels, **p**,**q**
  - affinity weight $\mathbf{w_{pq}}$ for each link (edge)
    - $\mathbf{w_{pq}}$ measures *similarity*
      - similarity is *inversely proportional* to difference (in color and position…)

Source: S. Seitz

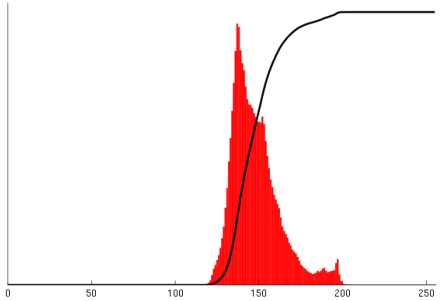
Histogram equalization

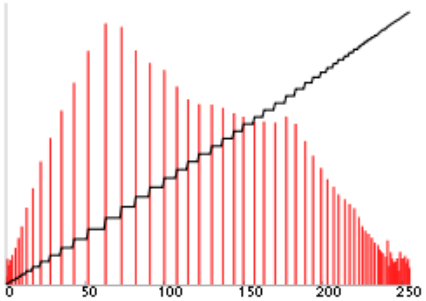
UNIVERSITÄT BONN

Pixels as distribution:

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

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





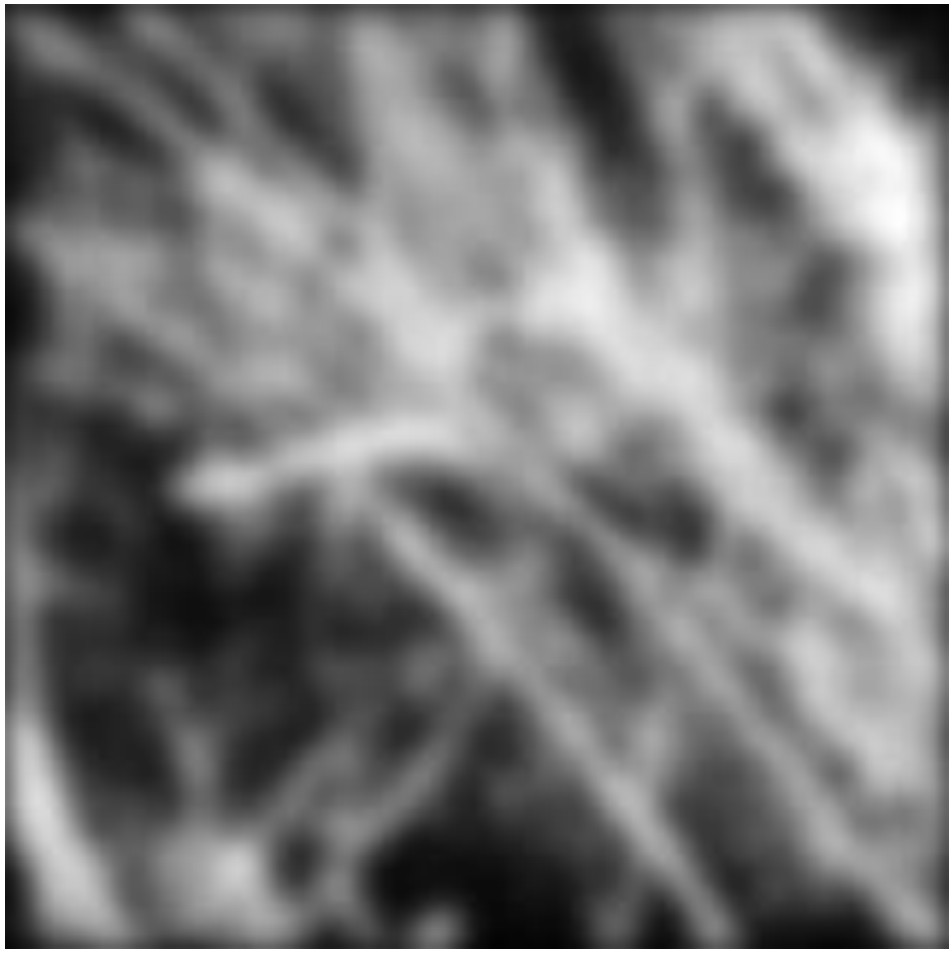
Cumulative distribution function

Make CDF linear:

$$h(f(x)) = F(f(x)) \cdot \max_{x \in \Omega} f(x)$$

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

Linear filtering



Source: S. Lazebnik

Defining convolution

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

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

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

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

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

$g(x,y)$

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

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: $ka * b = a * kb = k (a * b)$

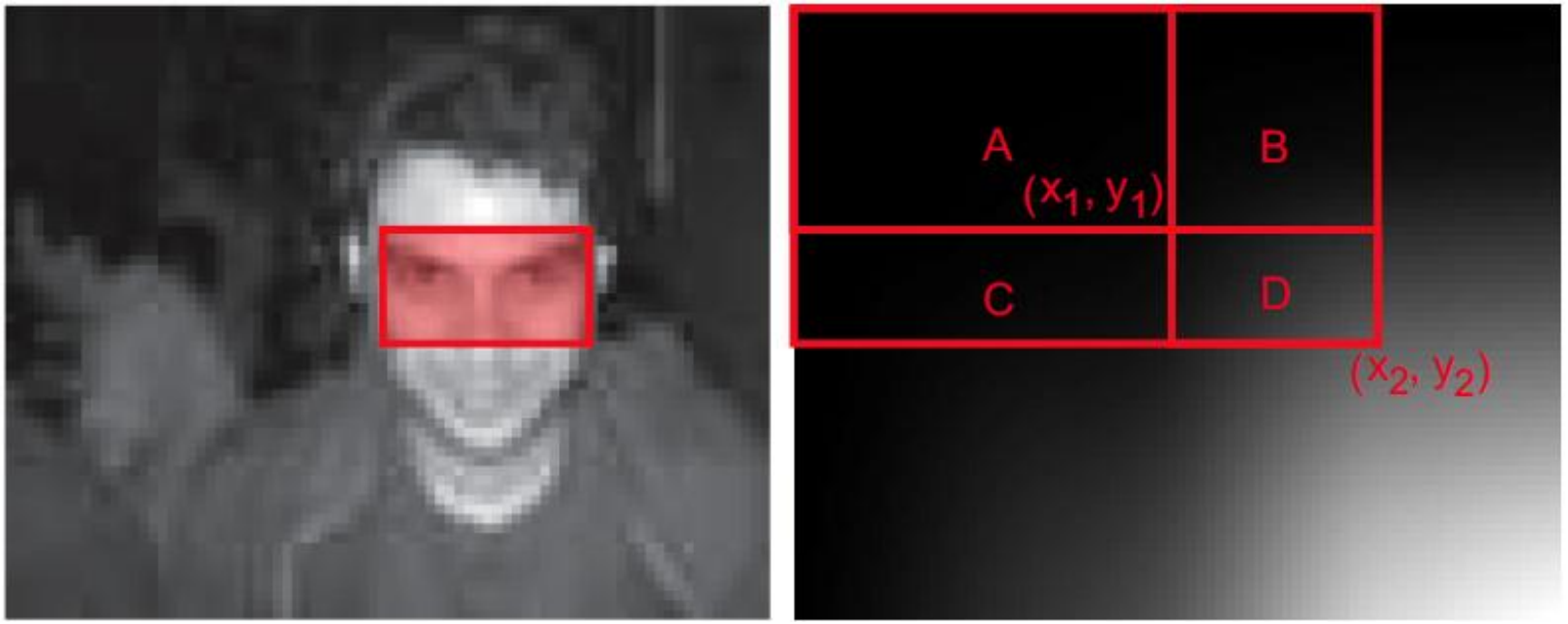
Identity: unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Source: S. Lazebnik

Integral image

UNIVERSITÄT BONN

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

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

(Note that filter sums to 1)

Source: D. Lowe

Practice with linear filters

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

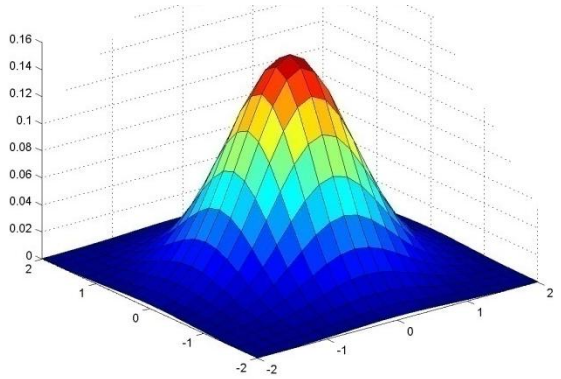
- Accentuates differences with local average

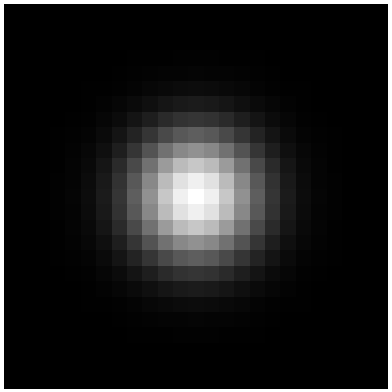
Source: D. Lowe

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, $\sigma = 1$

Source: C. Rasmussen

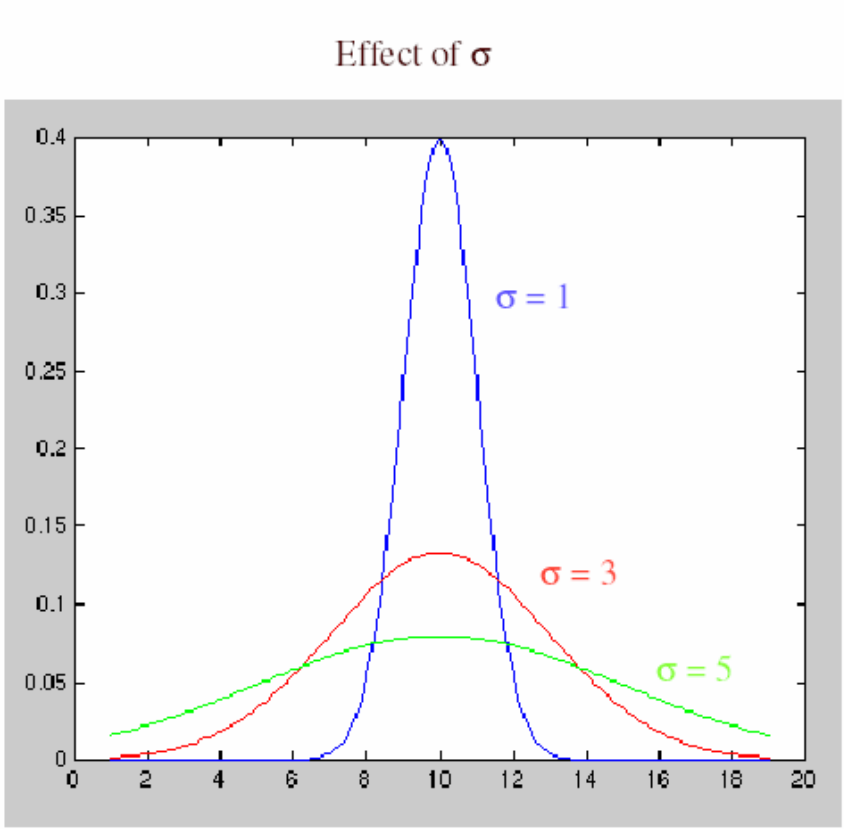
Choosing kernel width

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

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

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

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



Gaussian filters

Remove “high-frequency” components from the image (low-pass filter)

Convolution with self 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

Why is separability useful?

UNIVERSITÄT BONN

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:

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

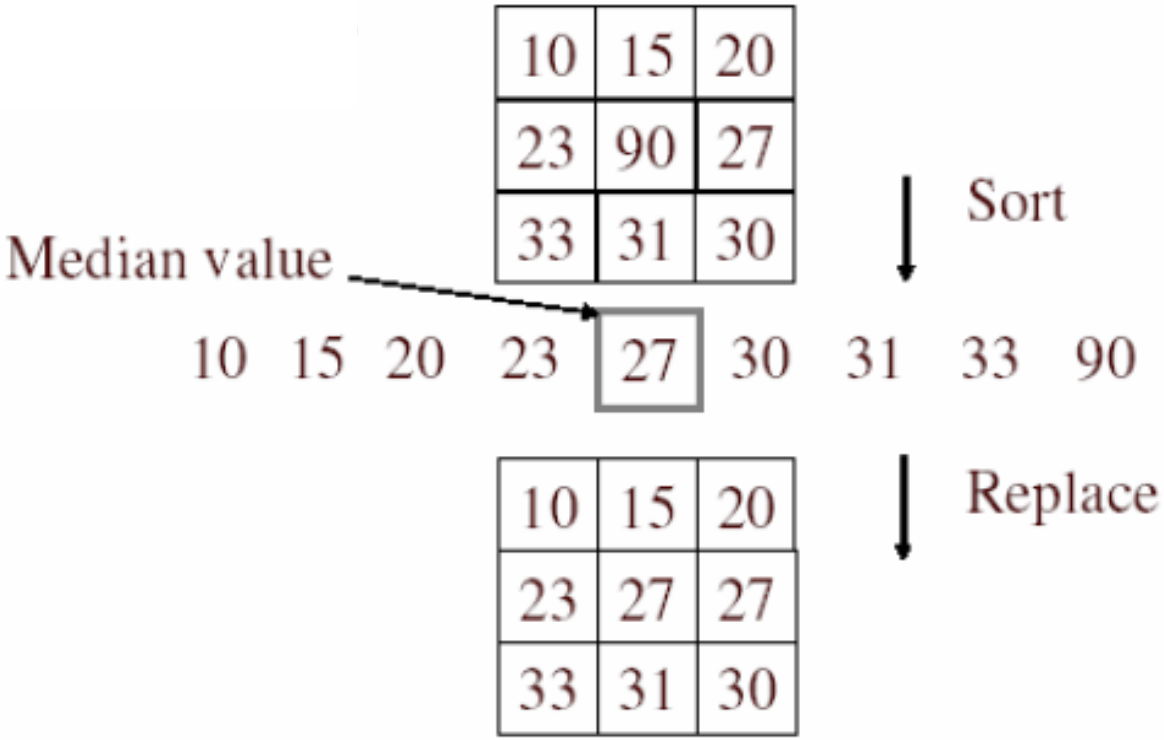
- Singular Value Decomposition (SVD)
- If only the first singular value σ_0 is non-zero, kernel is separabel
- Vertical and horizontal kernels:

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

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

Alternative idea: Median filtering

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



- Is median filtering linear?

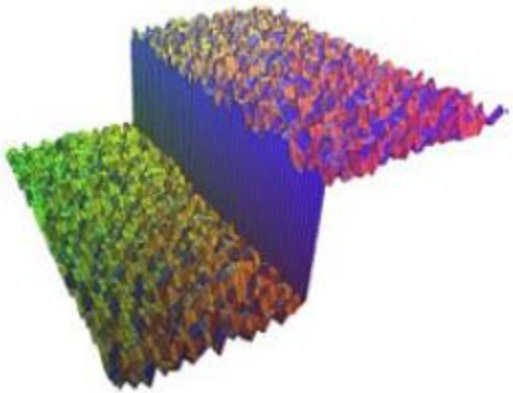
Source: K. Grauman

Bilateral filter

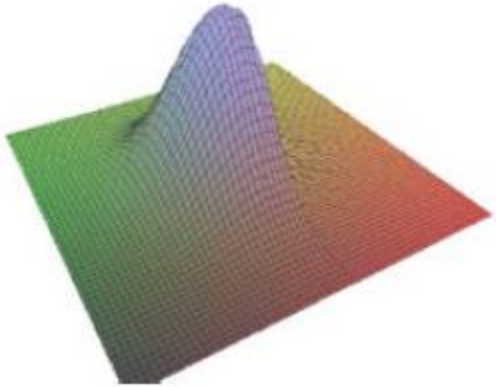
UNIVERSITÄT BONN

Filter is data dependent

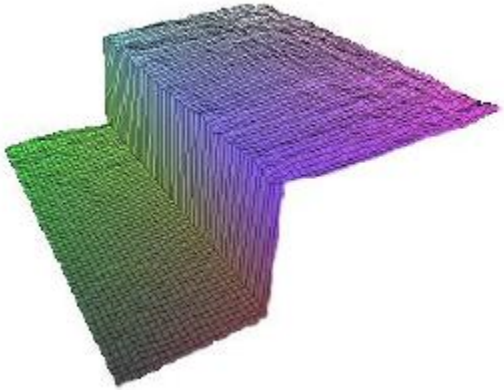
$$g(i,j) = \frac{\sum_{k,l} f(k,l) w(i,j,k,l)}{\sum_{k,l} w(i,j,k,l)}$$



Input



Bilaterial filter



Output

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

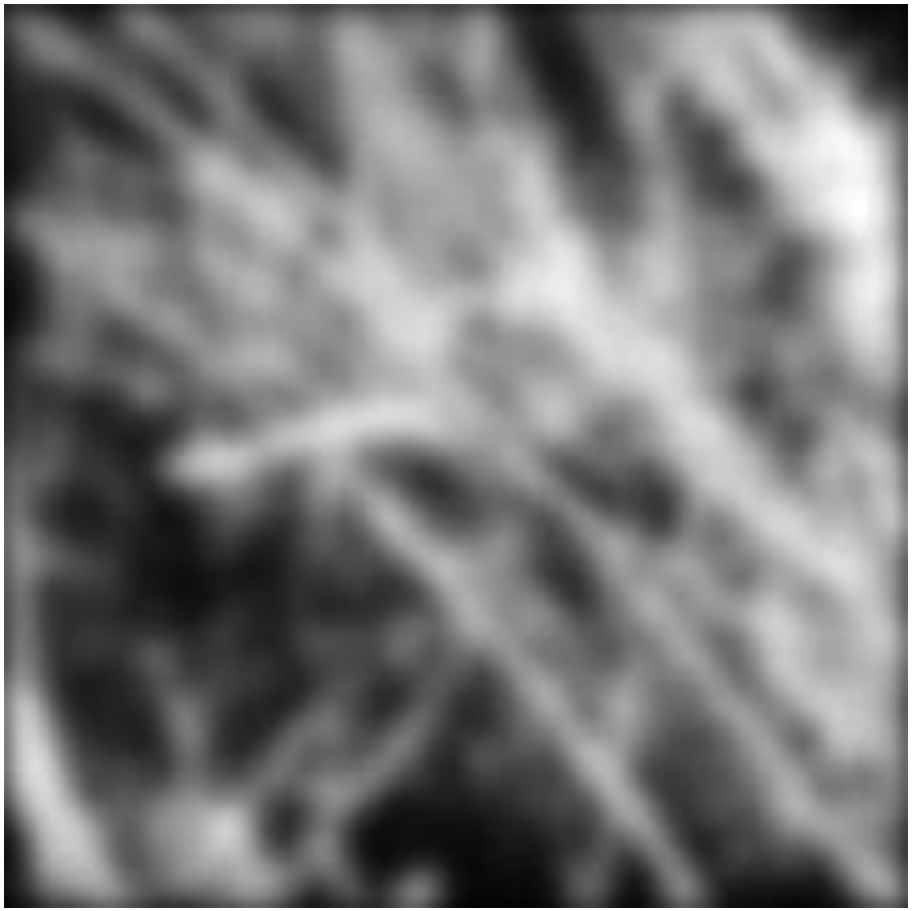
Slow, but methods for approximation exist

Linear filtering

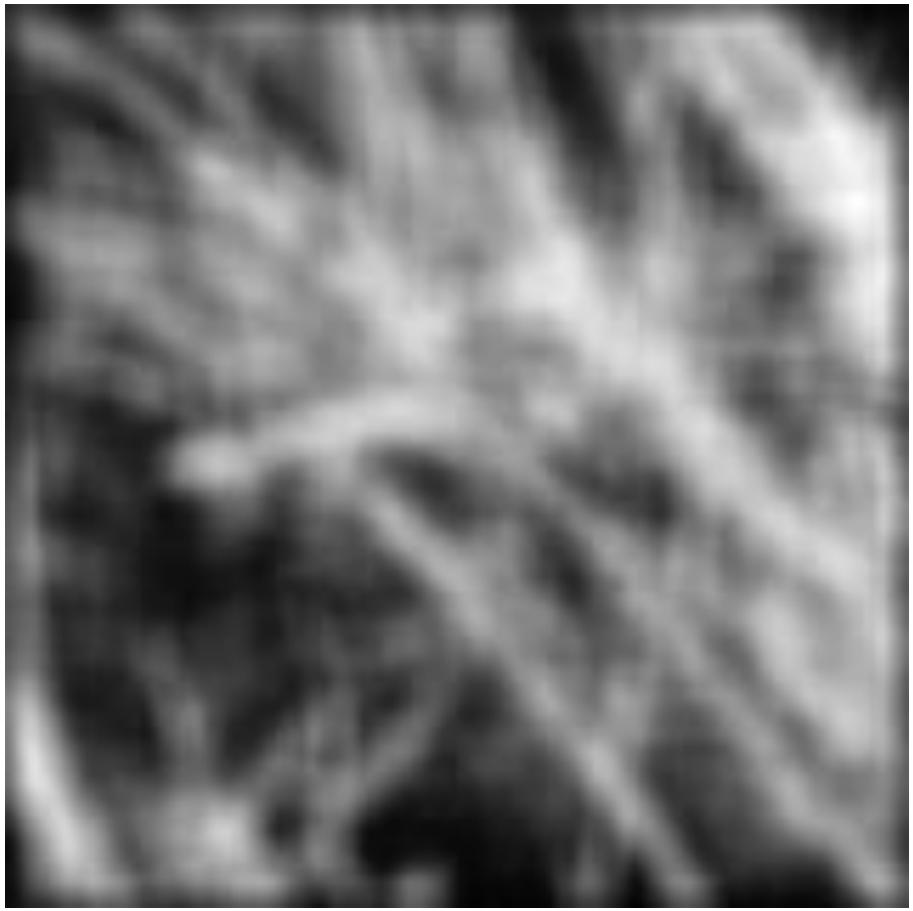
UNIVERSITÄT BONN

Gauss vs. box filter

Gaussian 



Box filter 



A sum of sines

Our building block:

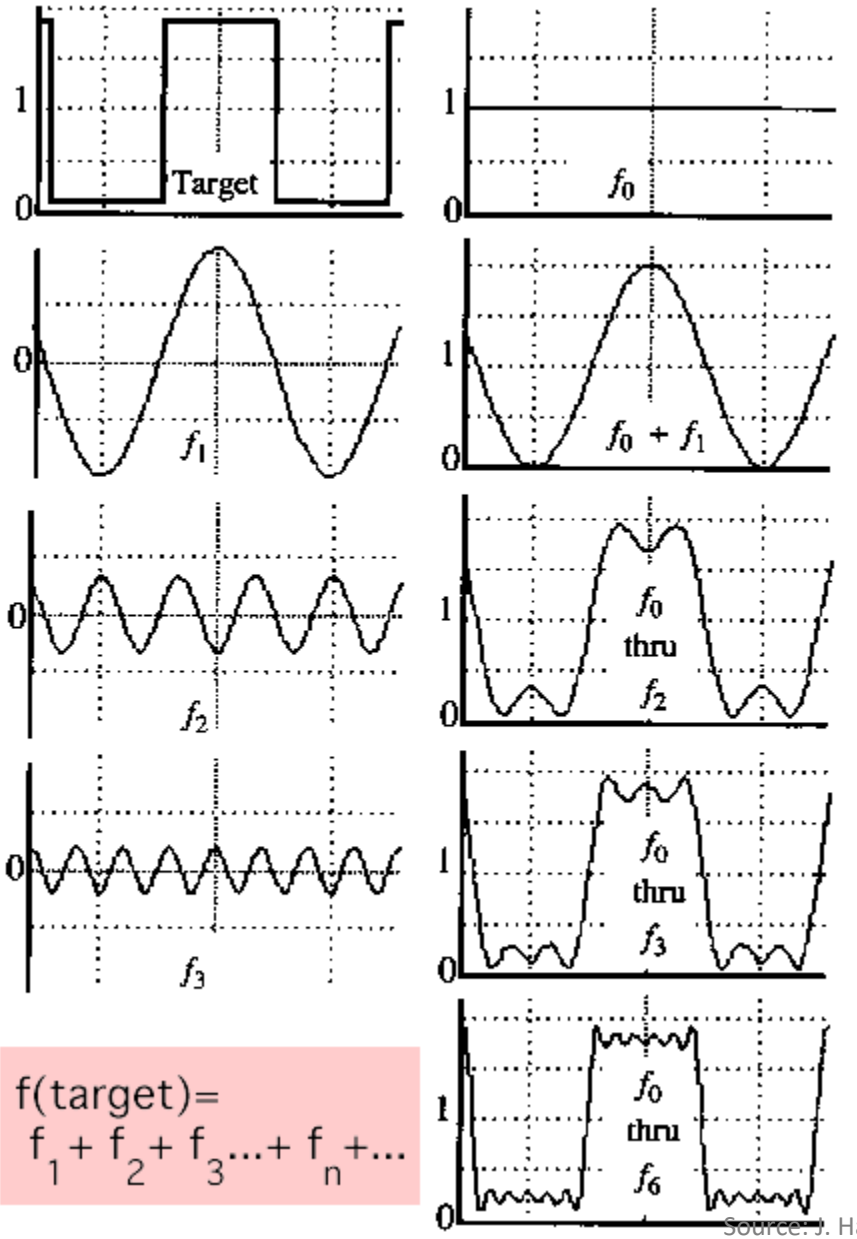
$$A\sin(\omega x + \phi)$$

A: Amplitude

ω: angular frequency

ϕ: phase

Add enough of them to get any signal *f(x)* you want!



Source: J. Hays

Fourier Transform

Fourier transform stores the magnitude and phase at each frequency

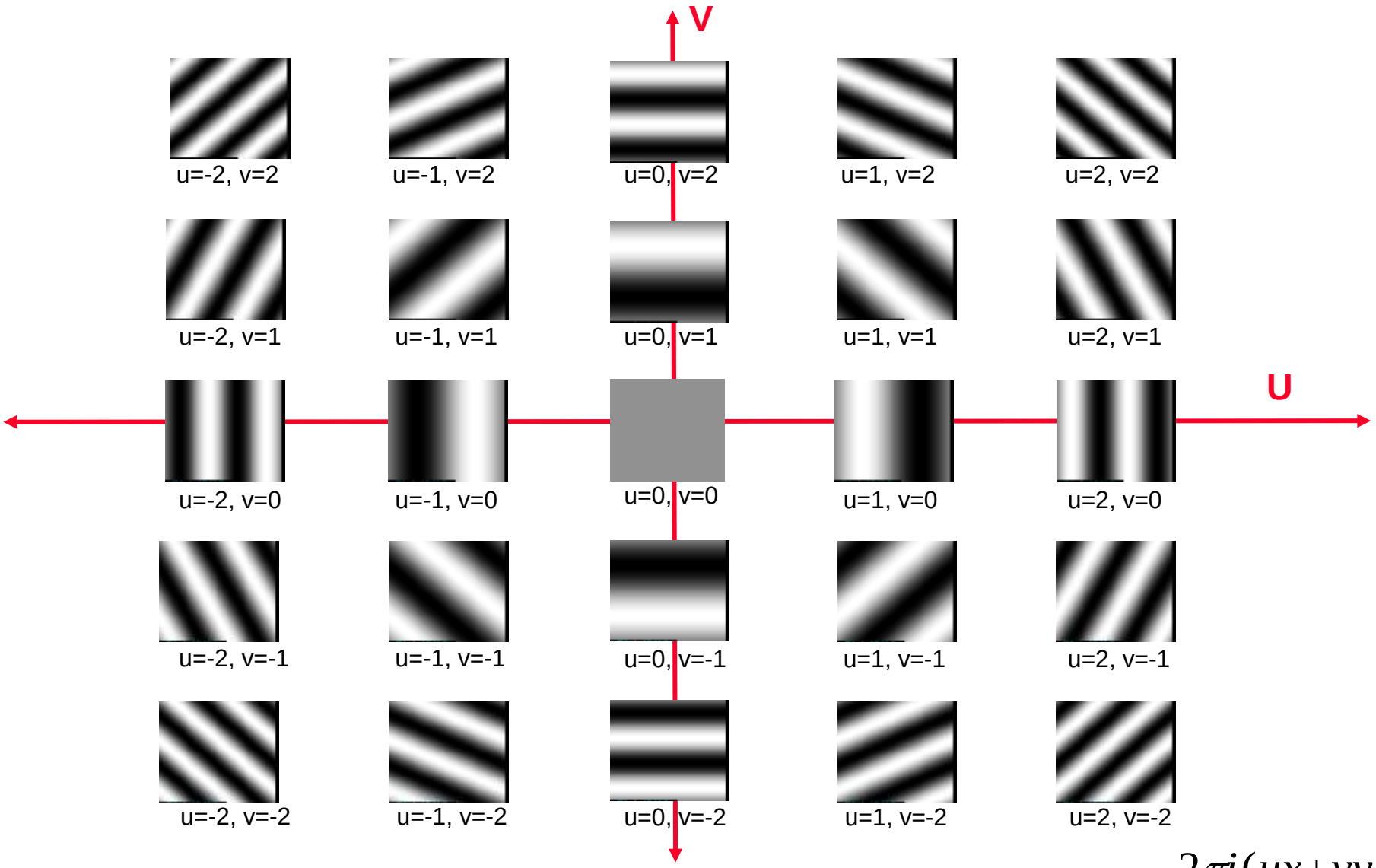
- Magnitude encodes how much signal there is at a particular frequency
- Phase encodes spatial information (indirectly)
- For mathematical convenience, this is often notated in terms of real and complex numbers

Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$

Phase: $\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$

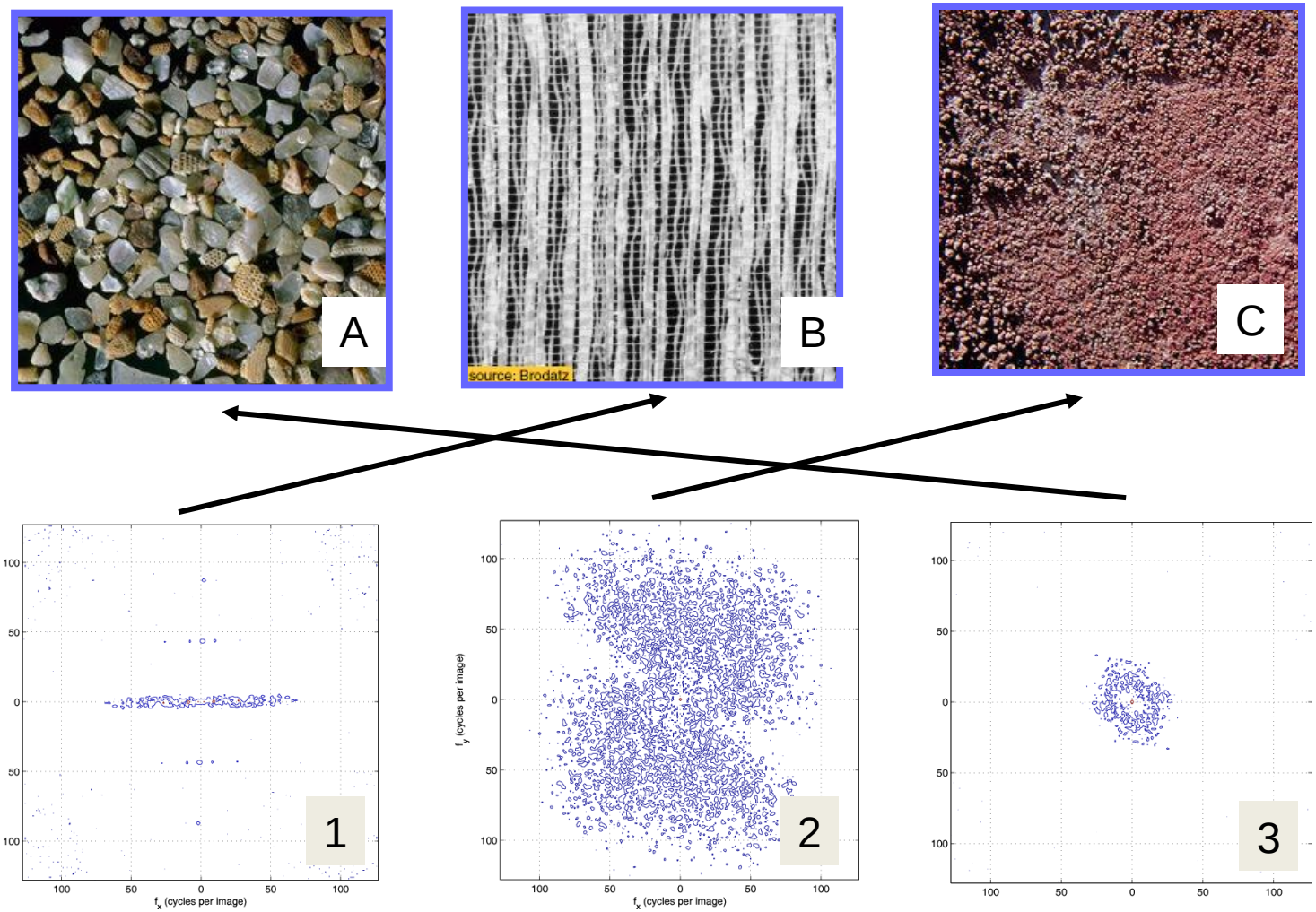
Source: J. Hays

The 2D Basis Functions



$$e^{2\pi i(ux+vy)}$$

Fourier Amplitude Spectrum



fx(cycles/image pixel size)

fx(cycles/image pixel size)

fx(cycles/image pixel size)

The Convolution Theorem

UNIVERSITÄT BONN

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

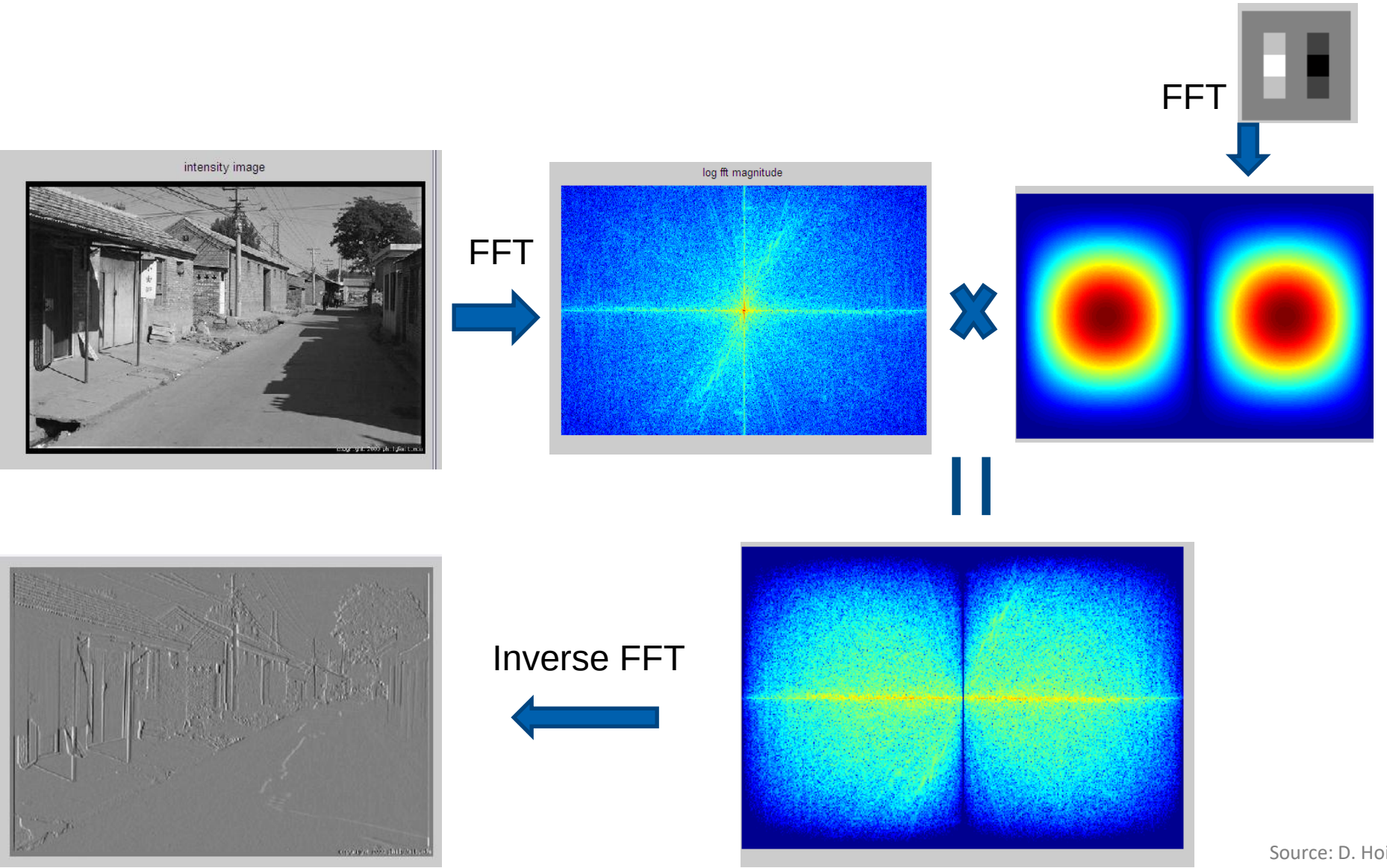
- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$f\ [gh] = f\ [g] * f\ [h]$$

- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

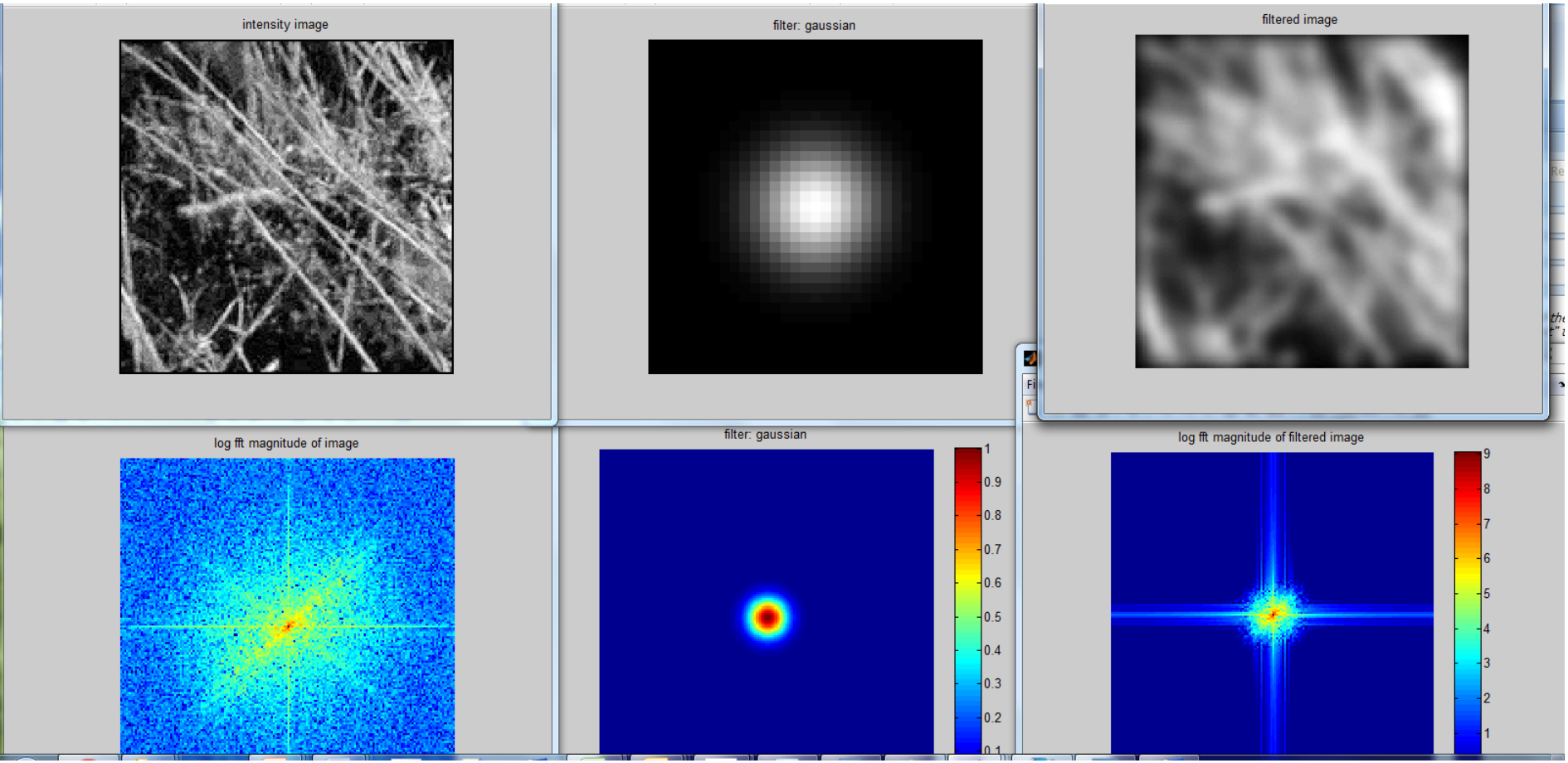
Source: J. Hays

Filtering in frequency domain



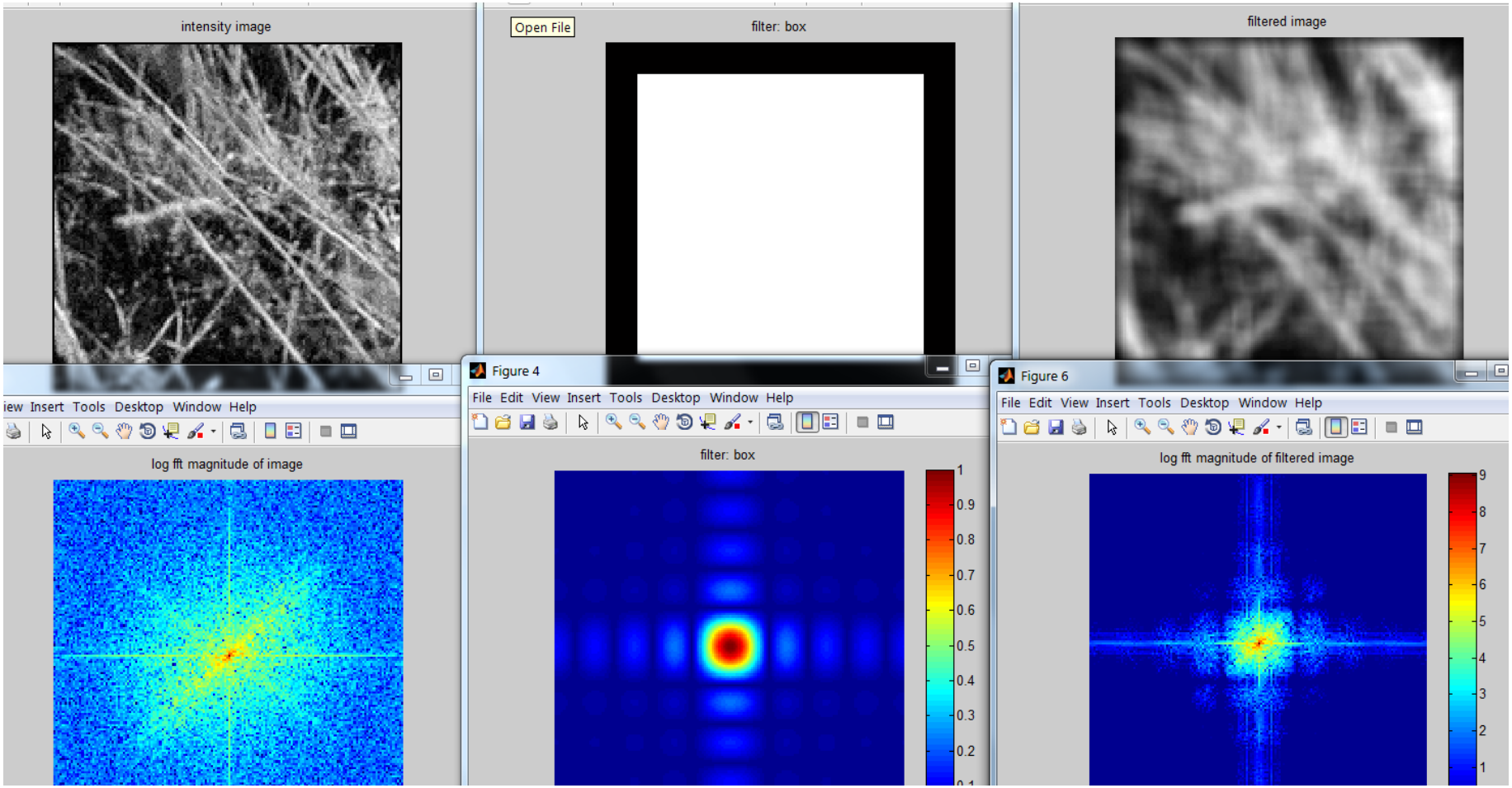
Source: D. Hoiem

Gaussian



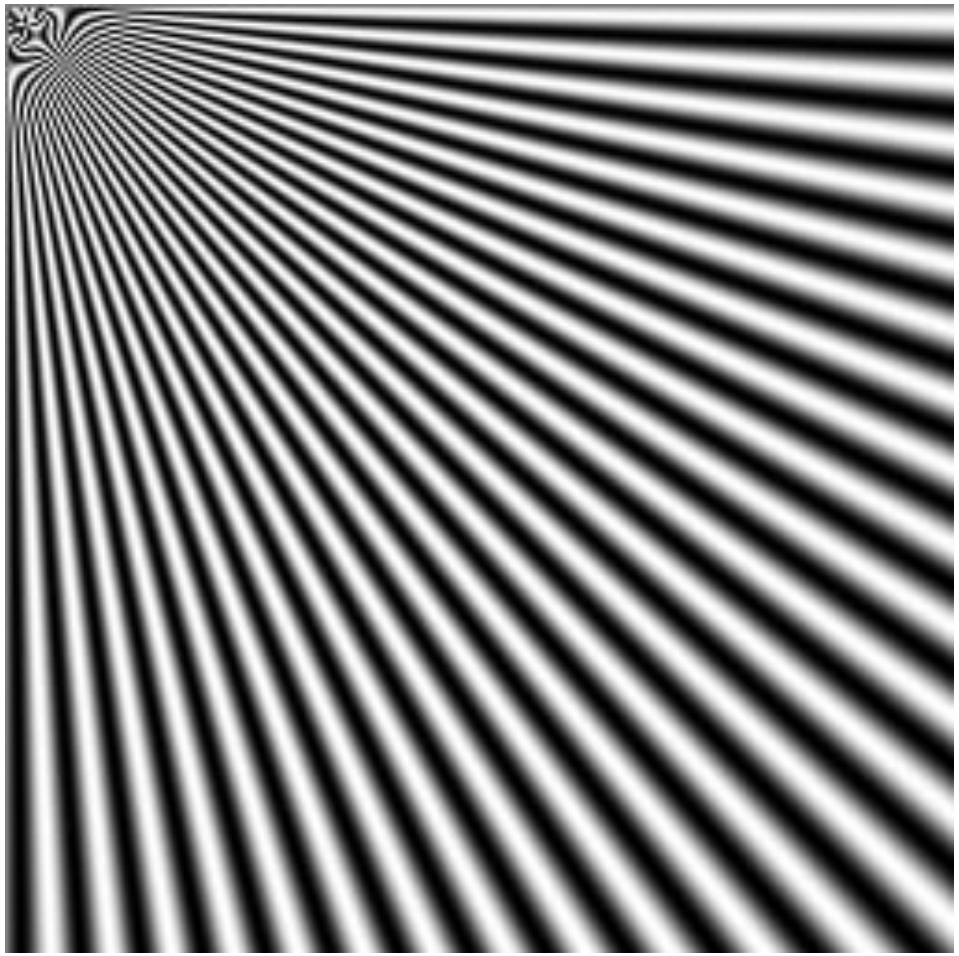
Source: J. Hays

Box filter



Source: J. Hays

Aliasing



Source: S. Narasimhan

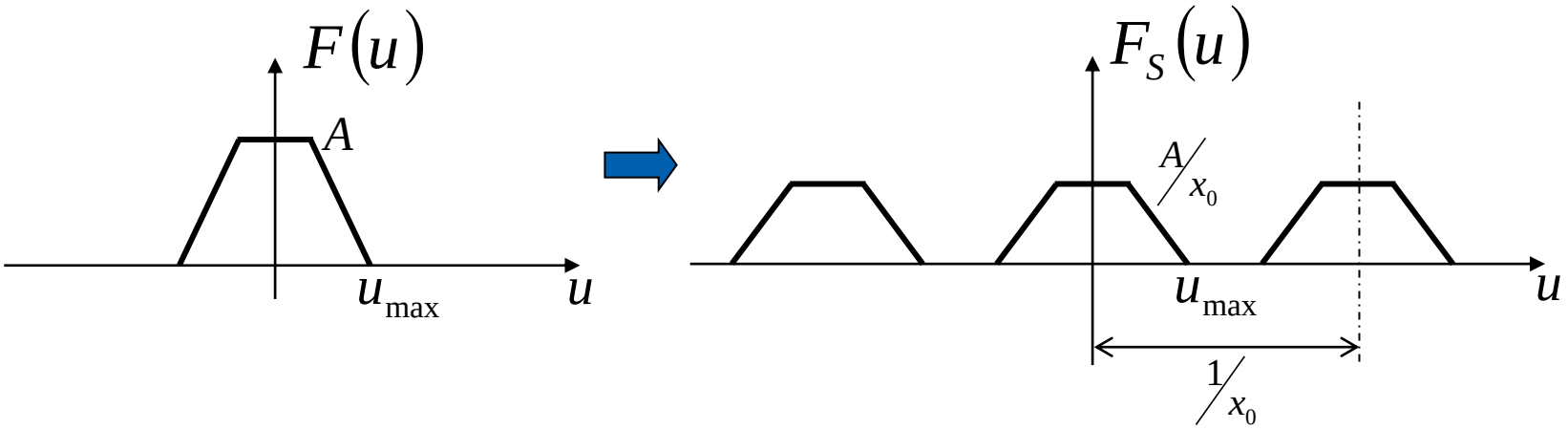
Sampling Theorem

Sampled function:

$$f_s(x) = f(x)s(x) = f(x)\sum_{n=-\infty}^{\infty}\delta(x - nx_0)$$

Sampling frequency $\frac{1}{x_0}$

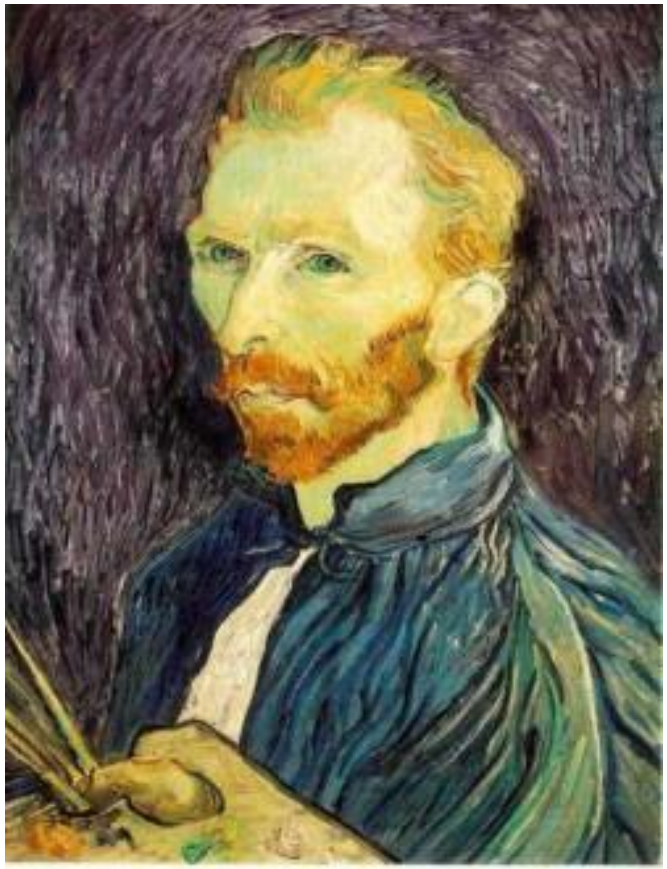
$$F_S(u) = F(u) * S(u) = F(u) * \frac{1}{x_0}\sum_{n=-\infty}^{\infty}\delta\left(u - \frac{n}{x_0}\right)$$



Only if $u_{\max} \leq \frac{1}{2x_0}$

Source: S. Narasimhan

Sub-Sampling with Gaussian Pre-Filtering



Gaussian 1/2

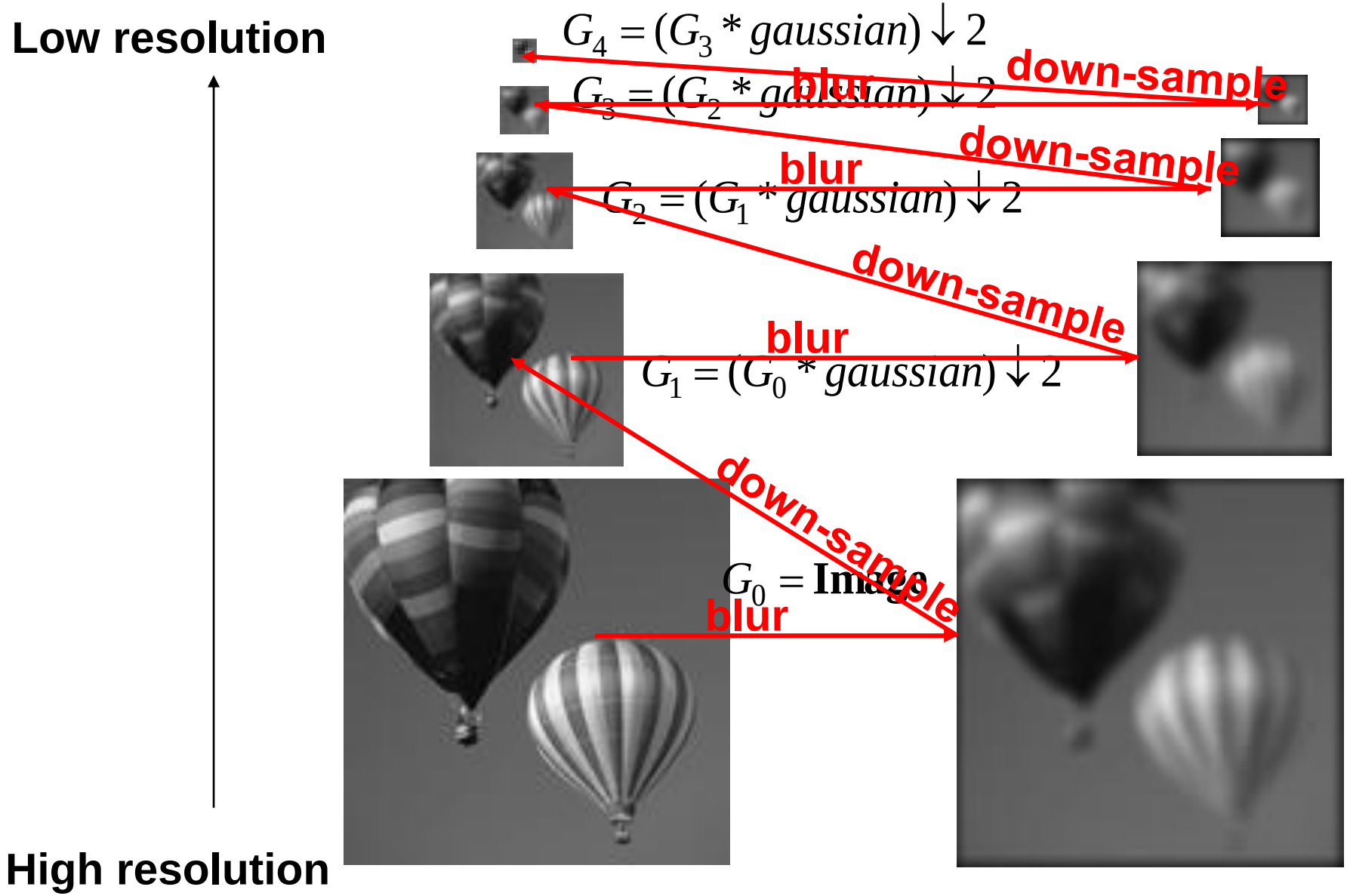
G 1/4

G 1/8

- Solution: filter the image, *then* subsample
  - Filter size should double for each ½ size reduction.

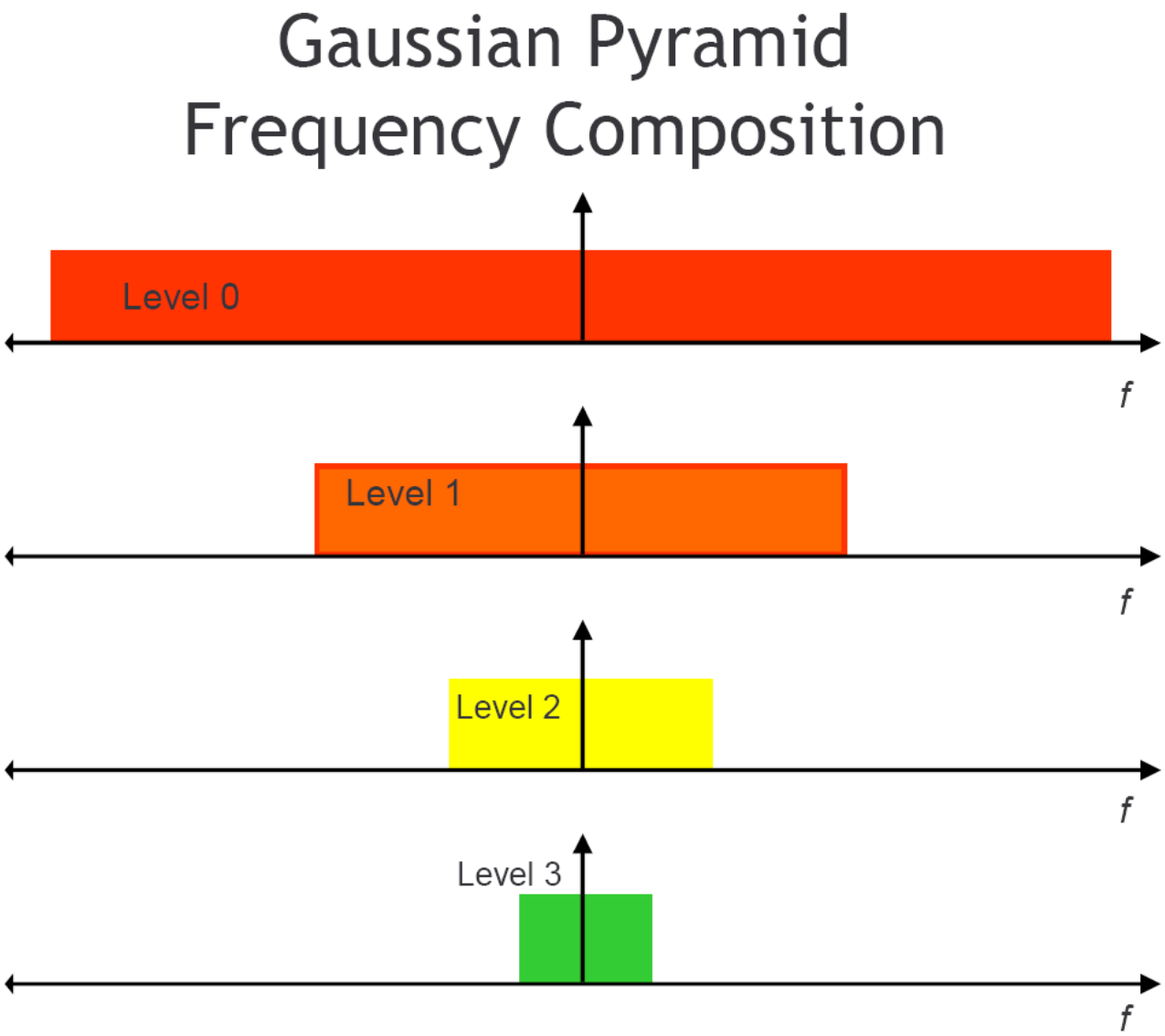
Source: S. Narasimhan

Gaussian pyramid



Source: S. Narasimhan

Frequencies



Source: S. Narasimhan

Sharpening

What does blurring take away?

–

=

Let’s add it back:

+

=

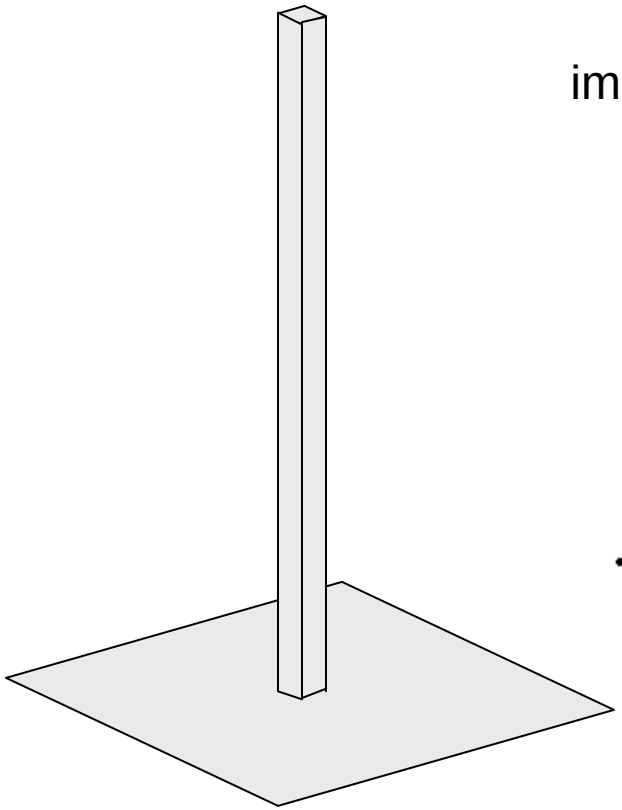
Laplacian of Gaussian

$$f + (f - f * g) = 2f - f * g = f * (2e - g)$$

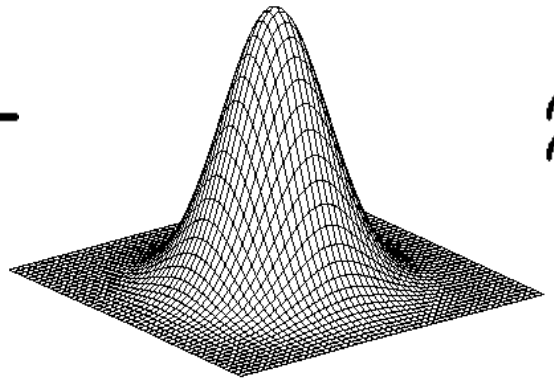
image

blurred image

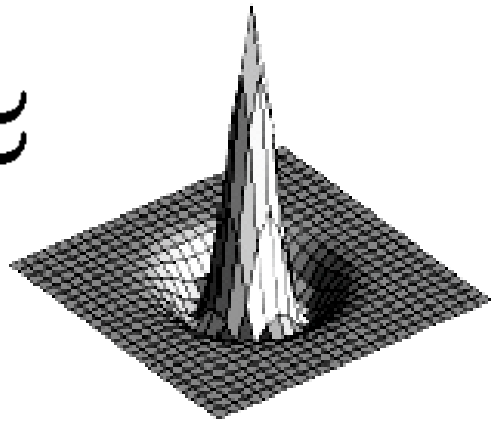
unit impulse (identity)



unit impulse



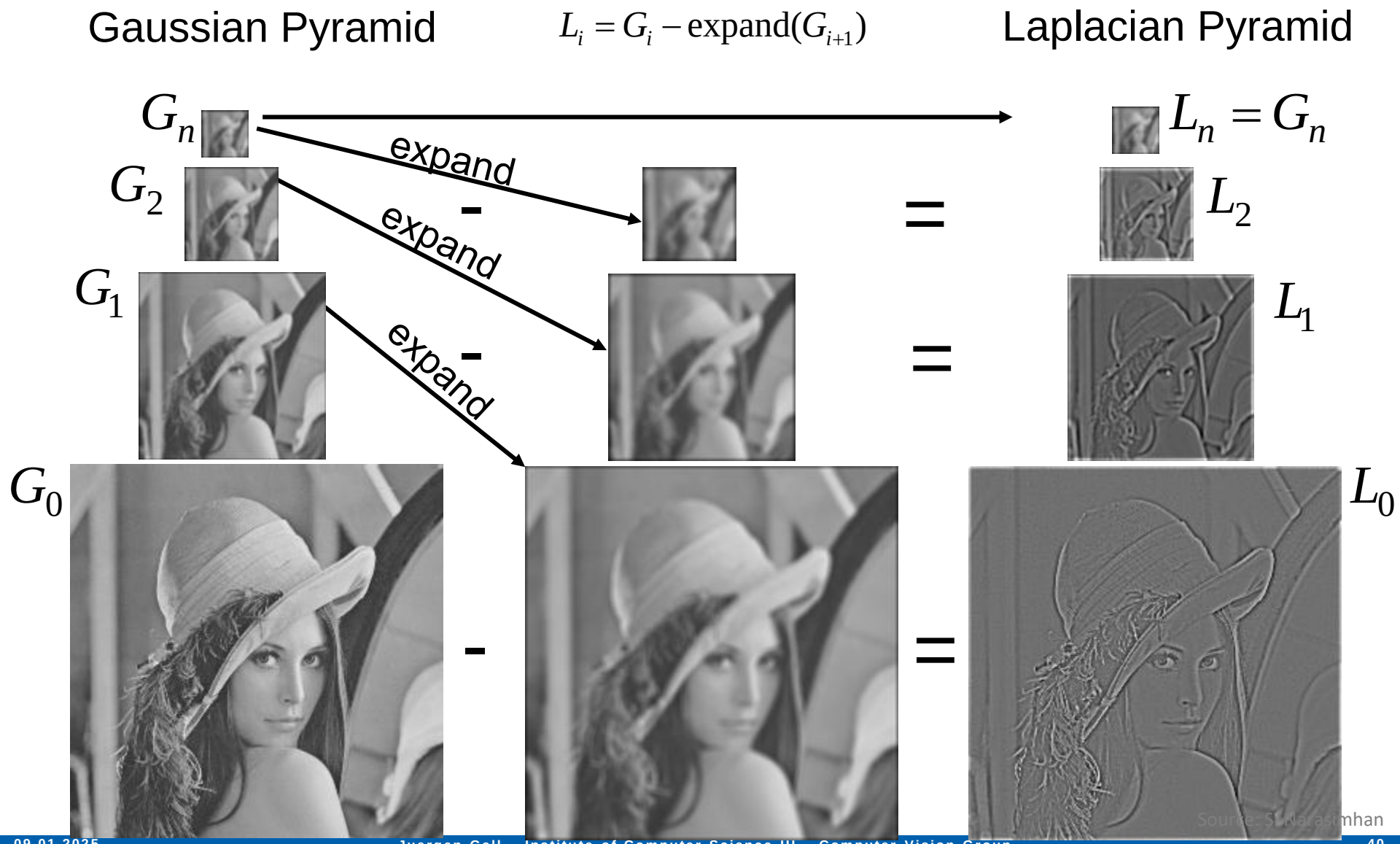
Gaussian



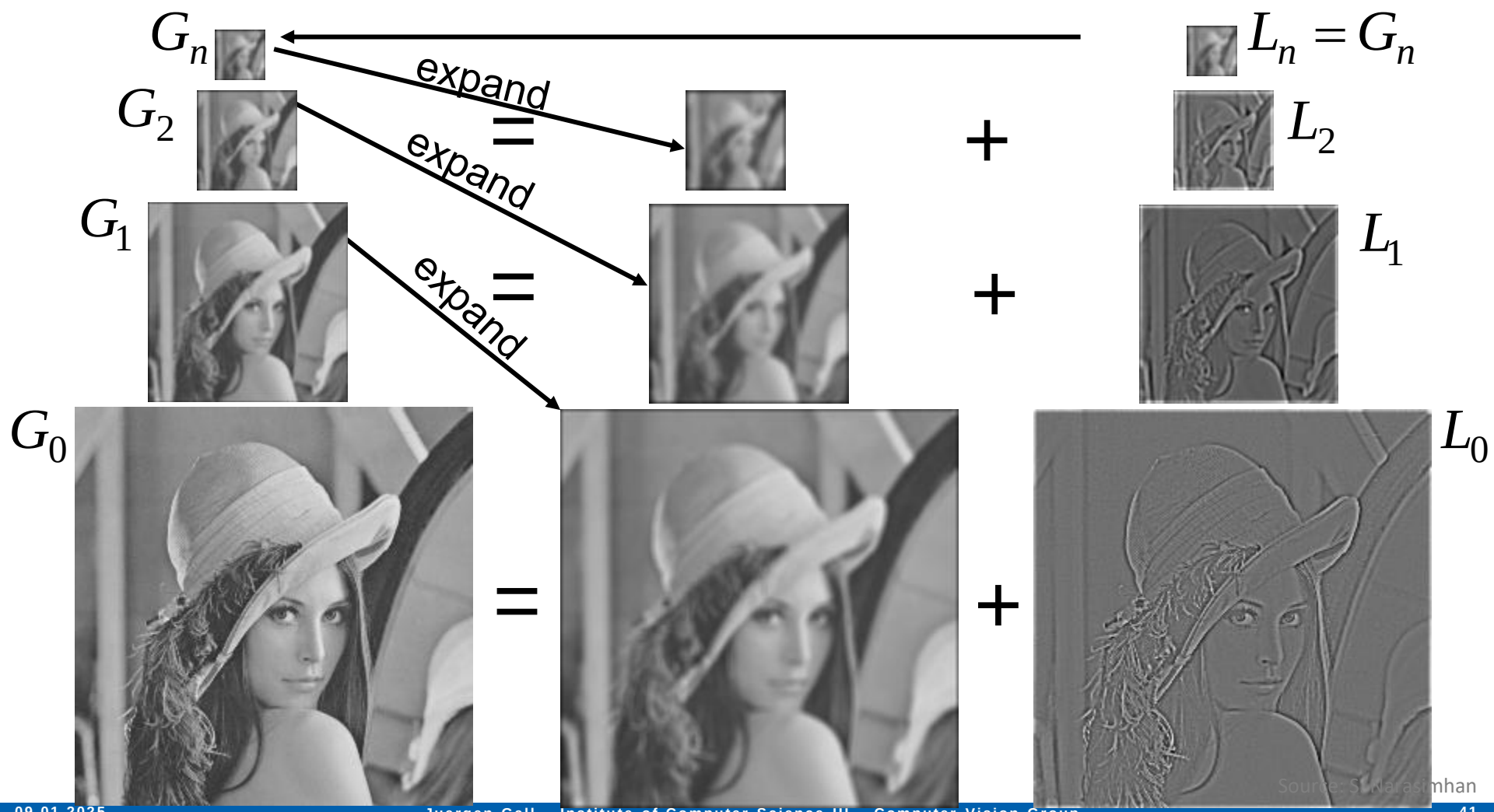
Laplacian of Gaussian

Source: S. Lazebnik

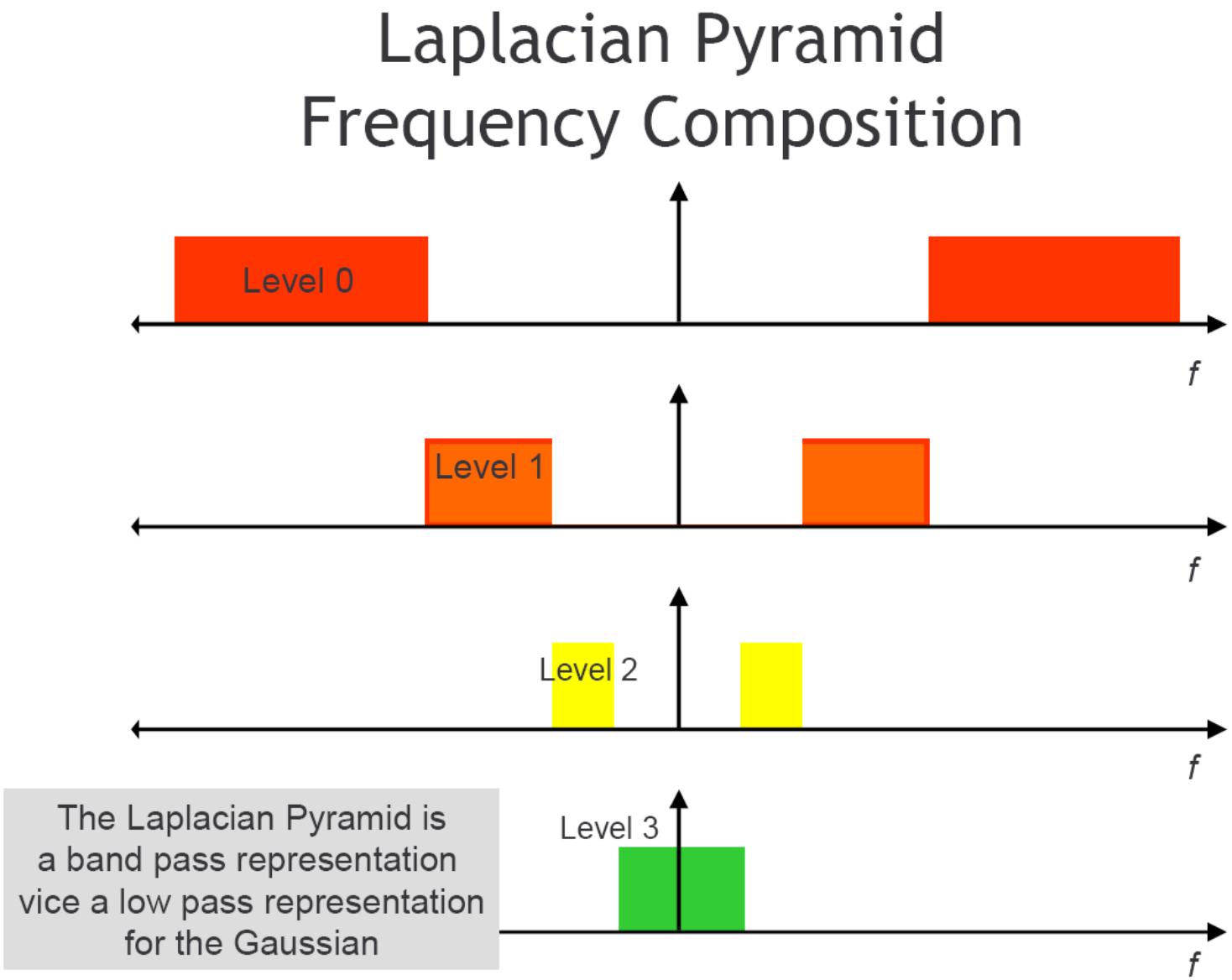
Laplacian pyramid



Laplacian pyramid

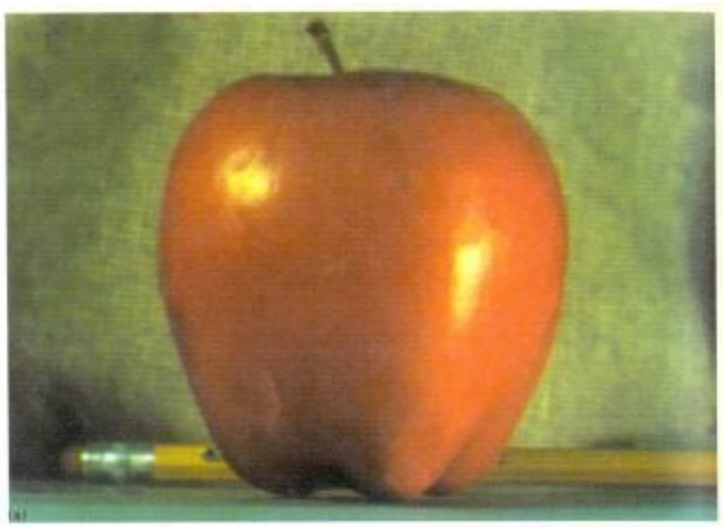


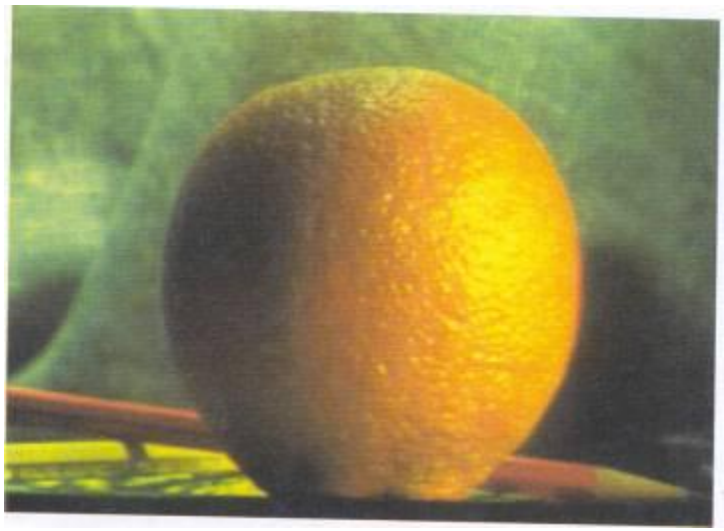
Frequencies

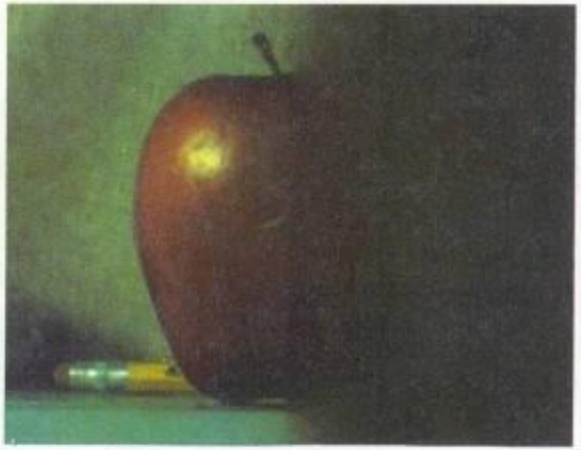


Source: S. Narasimhan

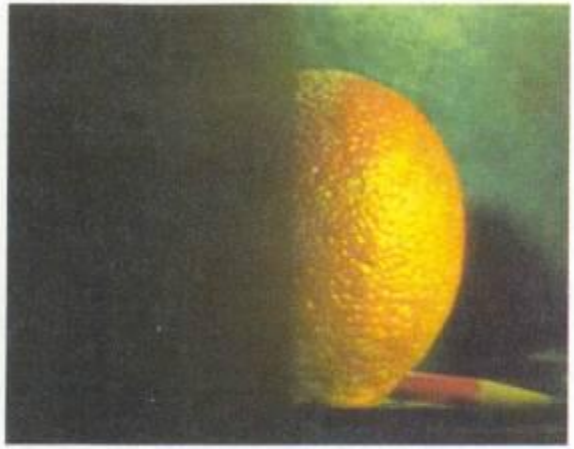
Pyramid Blending







(d)



(h)

(l)

Source: A. Efros

Finite difference filters

UNIVERSITÄT BONN

Other approximations of derivative filters exist:

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Source: K. Grauman

Derivative theorem of convolution

Differentiation is convolution, and convolution is associative:

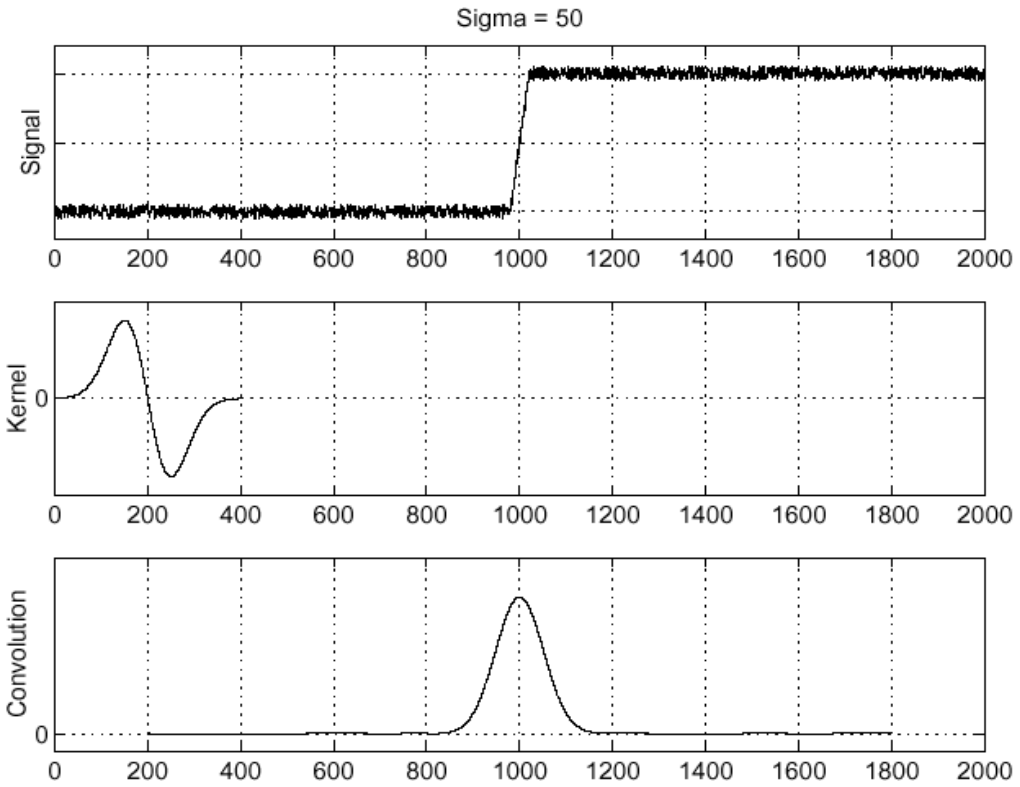
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

This saves us one operation:

f

$\frac{d}{dx} g$

$f * \frac{d}{dx} g$



Source: S. Seitz

The Canny edge detector

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei

Canny edge detector

UNIVERSITÄT BONN

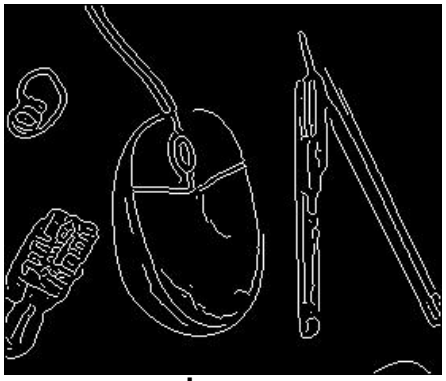
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   - Thin wide “ridges” down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

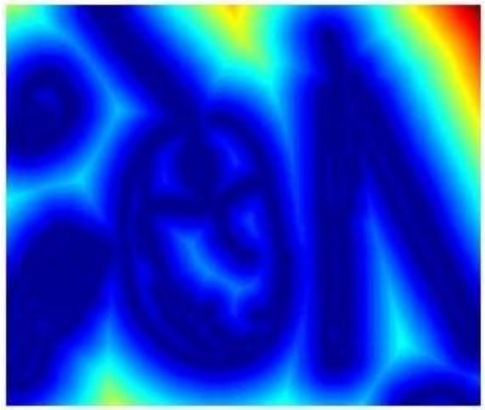
Source: S. Lazebnik

Distance transform

original



edges



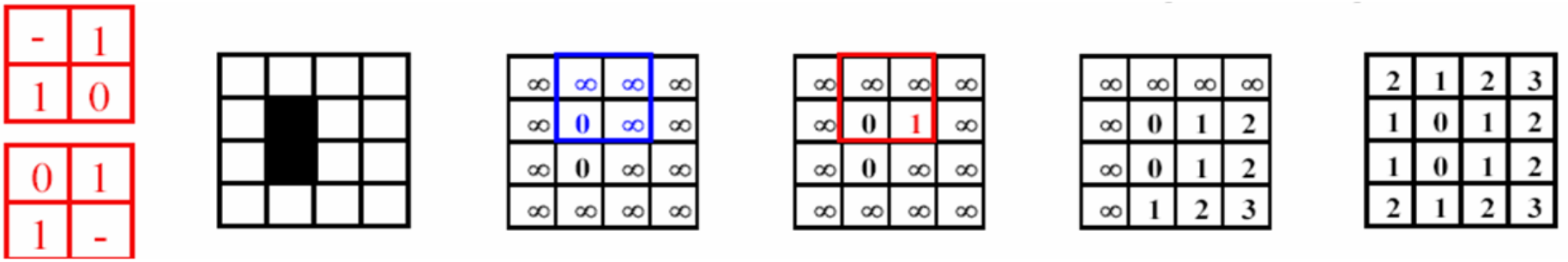
distance transform

Value at (x,y) tells how far that position is from the nearest edge point (or other binary mage structure)

Source: K. Grauman

Distance Transform (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right



Source: D. Huttenlocher

Chamfer distance

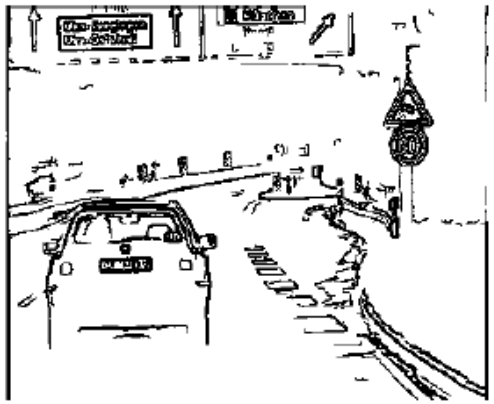
- Average distance to nearest feature

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

- T: template shape→ a set of points
- I: image to search→ a set of points
- $d_I(t)$: min distance for point t to some point in I

Source: K. Grauman

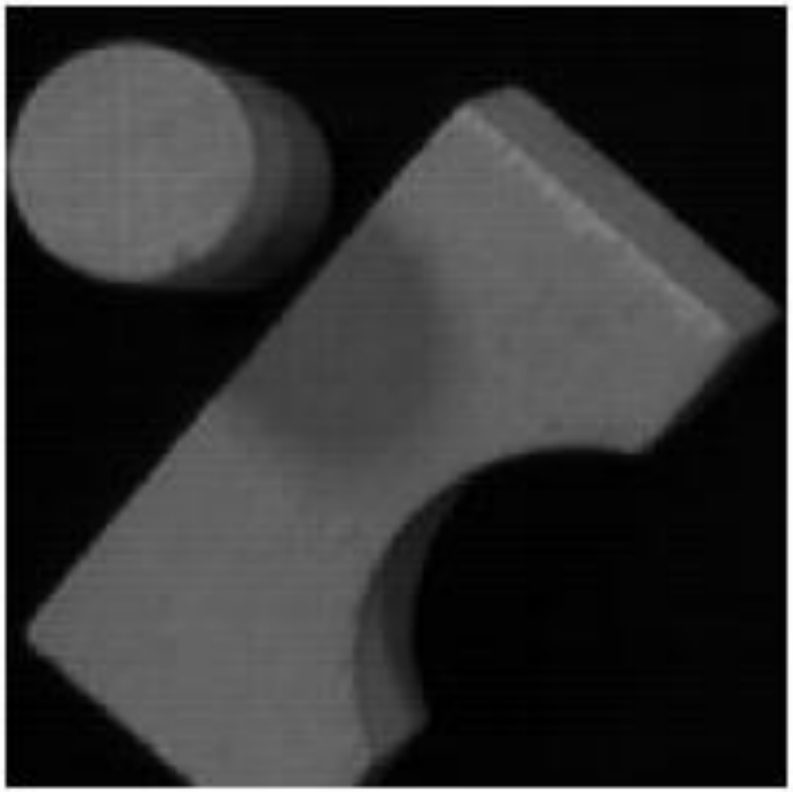
Chamfer distance

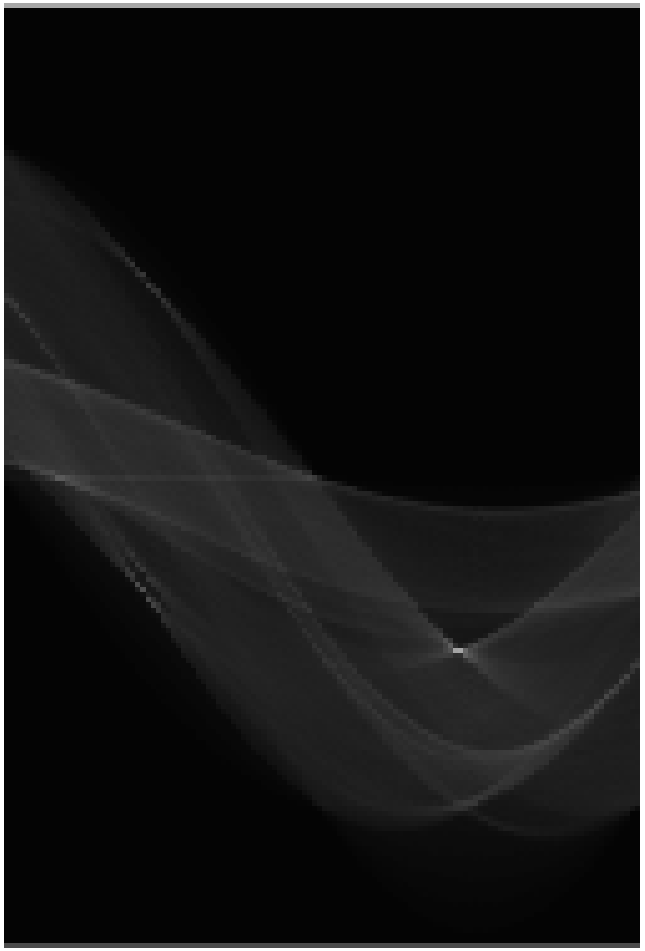


Edge image **Distance transform image**

Source: D. Gavrila

Hough transform





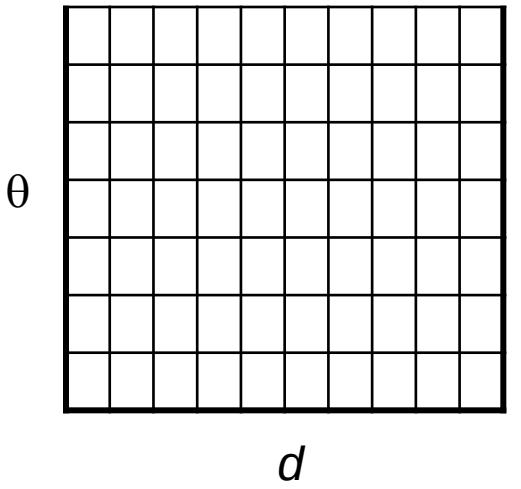
Source: K. Grauman

Hough transform algorithm

Using the polar parameterization:

$$x\cos\theta - y\sin\theta = d$$

H: accumulator array (votes)



Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. for each edge point I[x,y] in the image

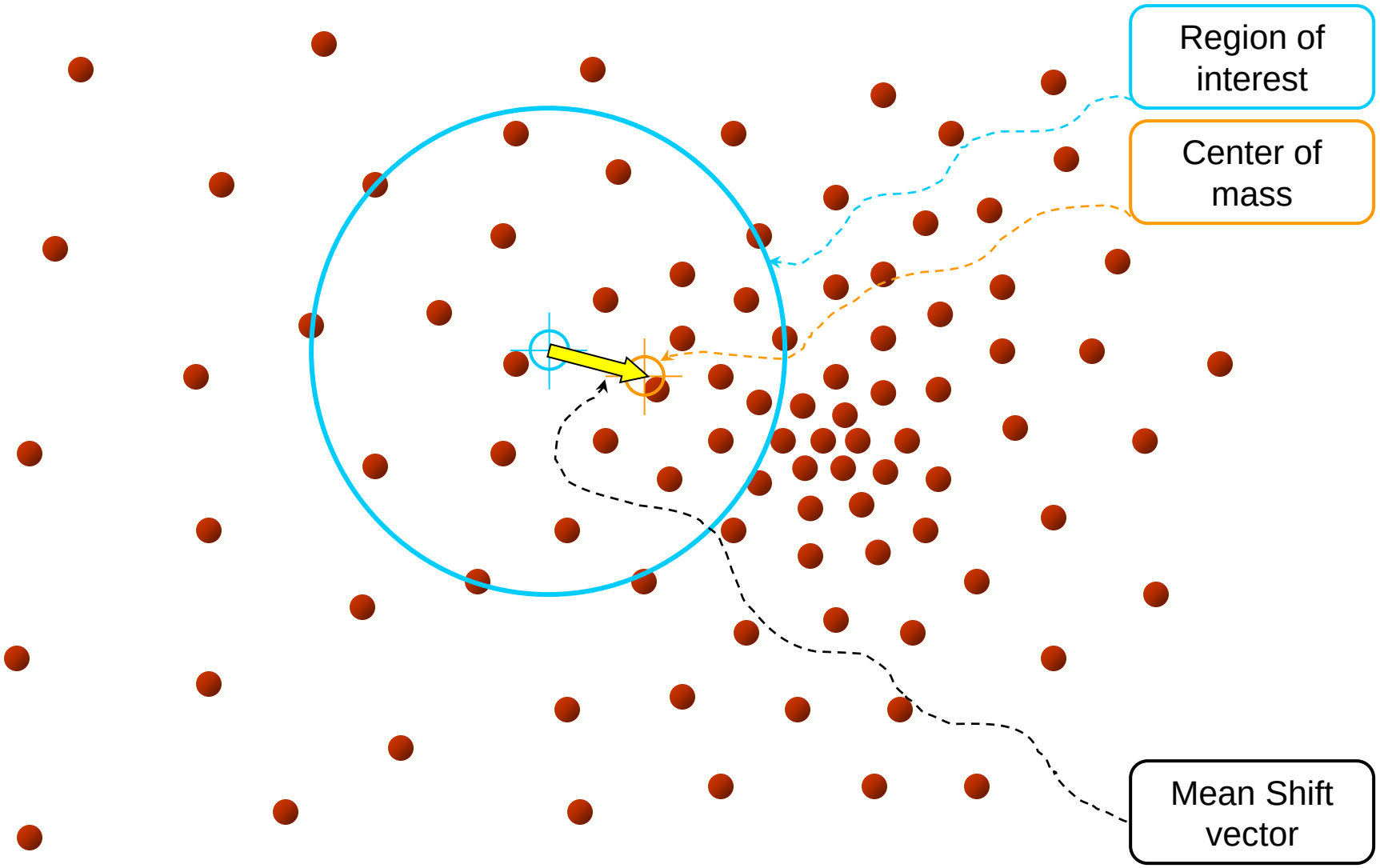
   for θ = 0 to 180 // some quantization

   $d = x\cos\theta - y\sin\theta$

   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by $d = x\cos\theta - y\sin\theta$

Source: S. Seitz

Mean Shift



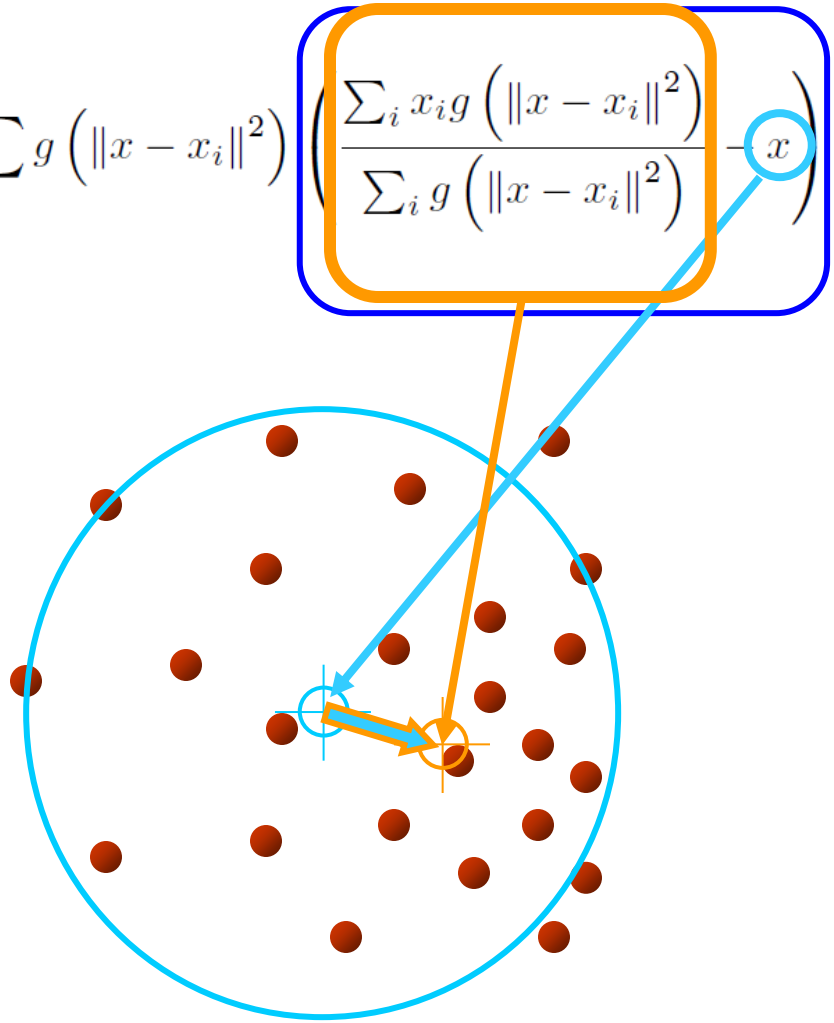
Objective : Find the densest region

Distribution of identical billiard balls

Source: Y. Ukrainitz and B. Sarel

Mean shift

$$\nabla P(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\nabla k_i = \frac{2c}{N}\sum_i g\left(\|x - x_i\|^2\right)\left(\frac{\sum_i x_i g\left(\|x - x_i\|^2\right)}{\sum_i g\left(\|x - x_i\|^2\right)} - x\right)$$



Simple Mean Shift procedure:

- Compute mean shift vector

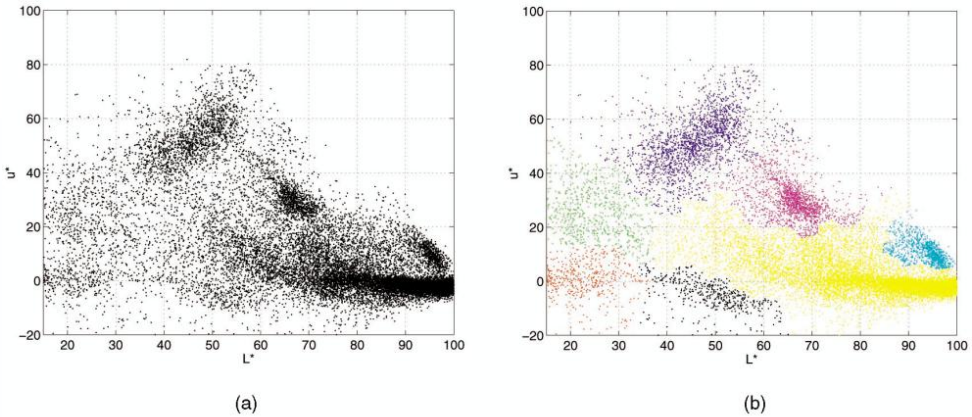
$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{h}\right)} - \mathbf{x}\right]$$

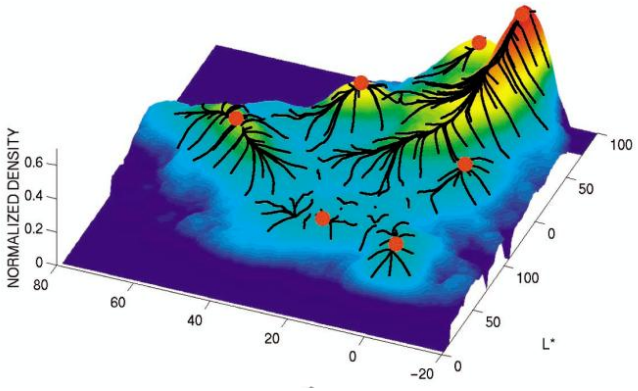
- Translate the Kernel window by **m(x)**

Source: Y. Ukrainitz and B. Sarel

Mean shift clustering/segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode





Source: K. Grauman

Mean Shift Tracking

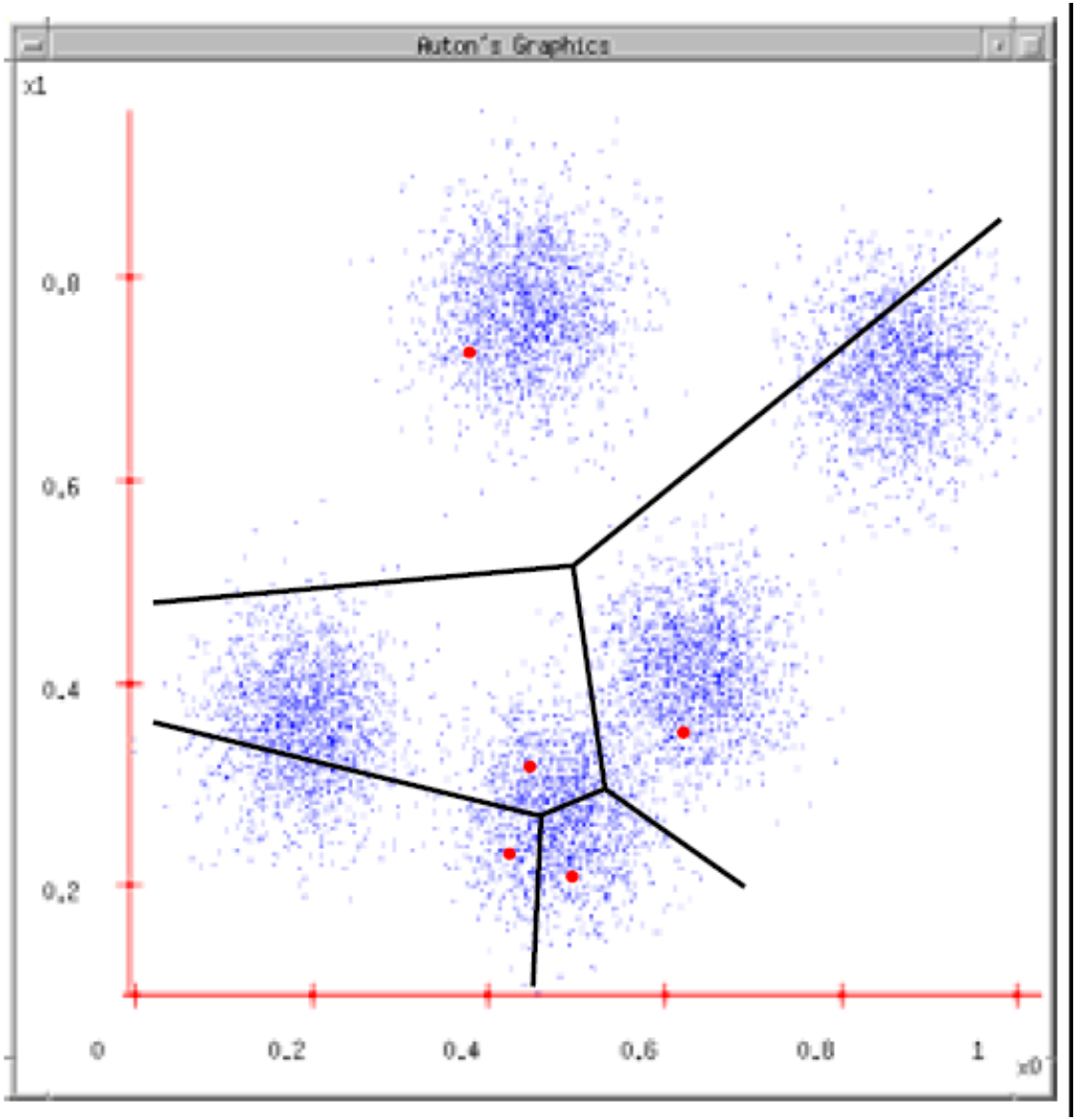
UNIVERSITÄT BONN

Goal: Mark object in first frame and locate it in all frames

D. Comaniciu. **Real-Time Tracking of Non-Rigid Objects using Mean Shift.** CVPR 2000

K-means

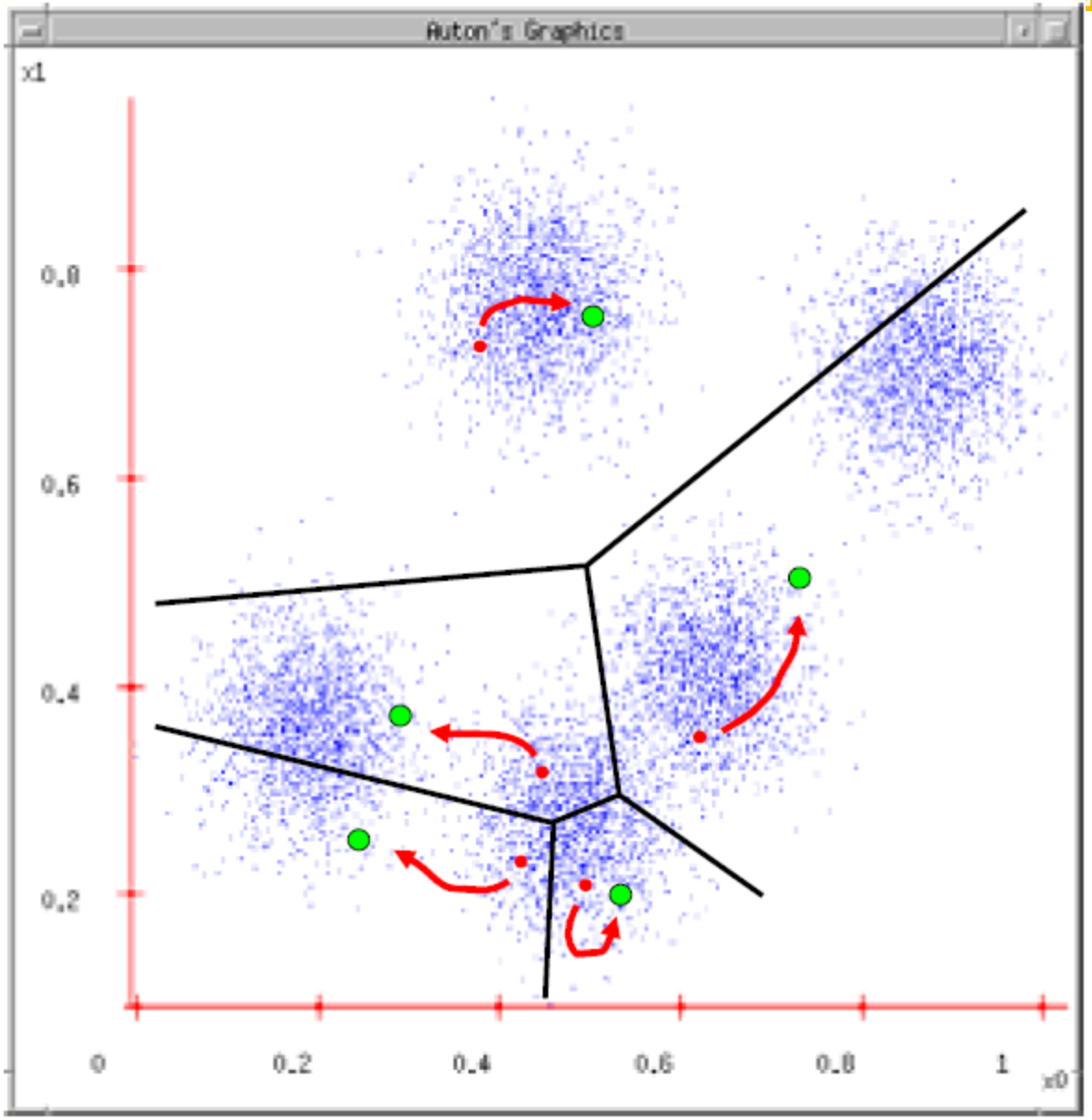
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Source: A. Moore

K-means

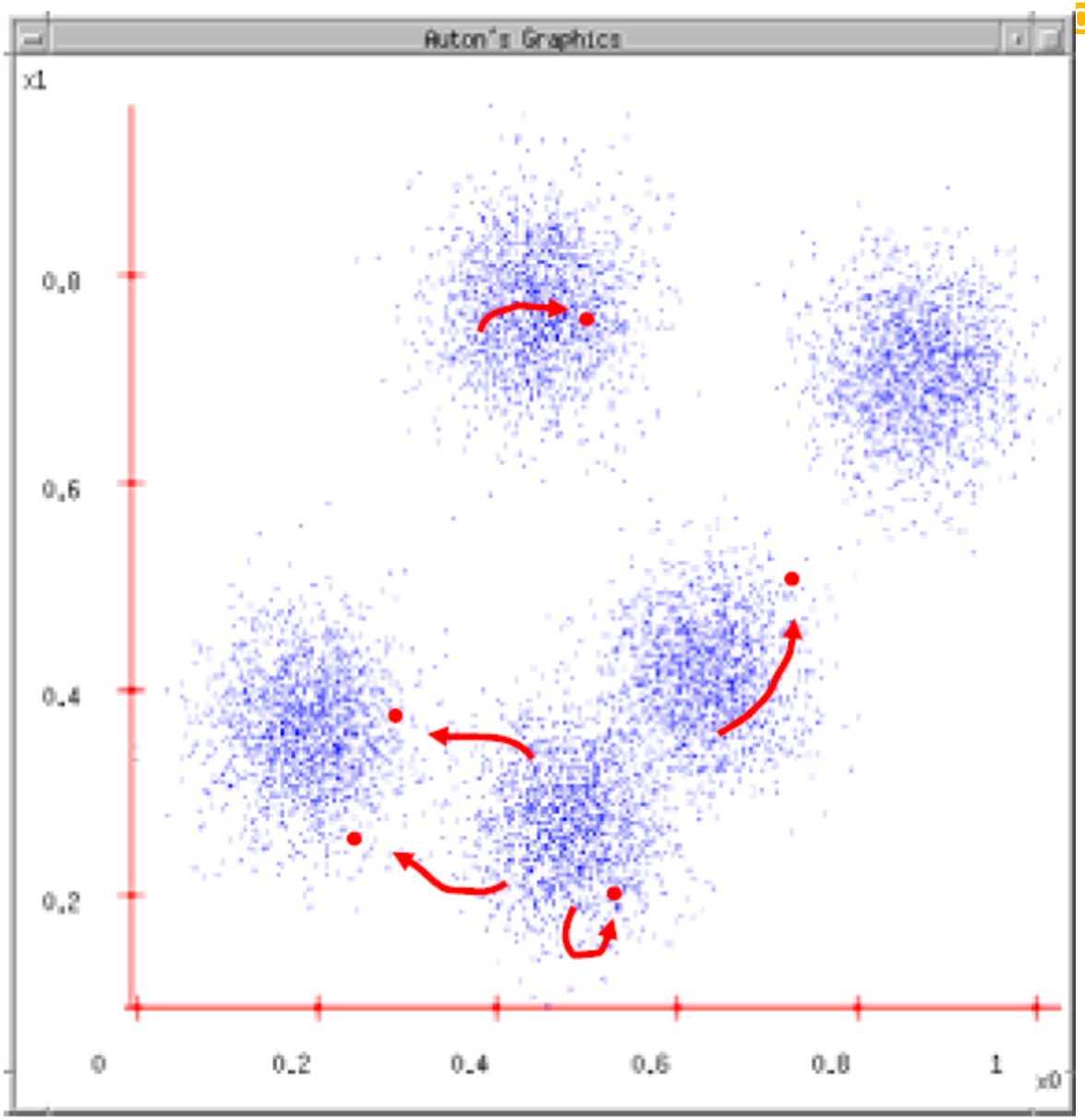
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



Source: A. Moore

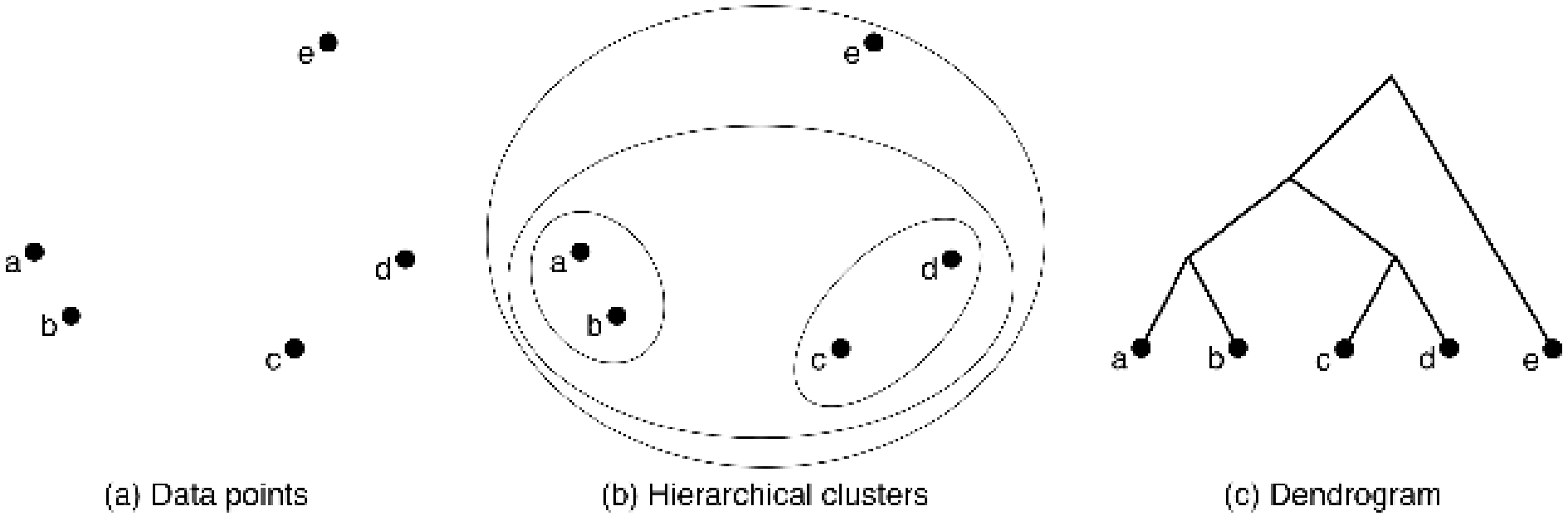
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Source: A. Moore

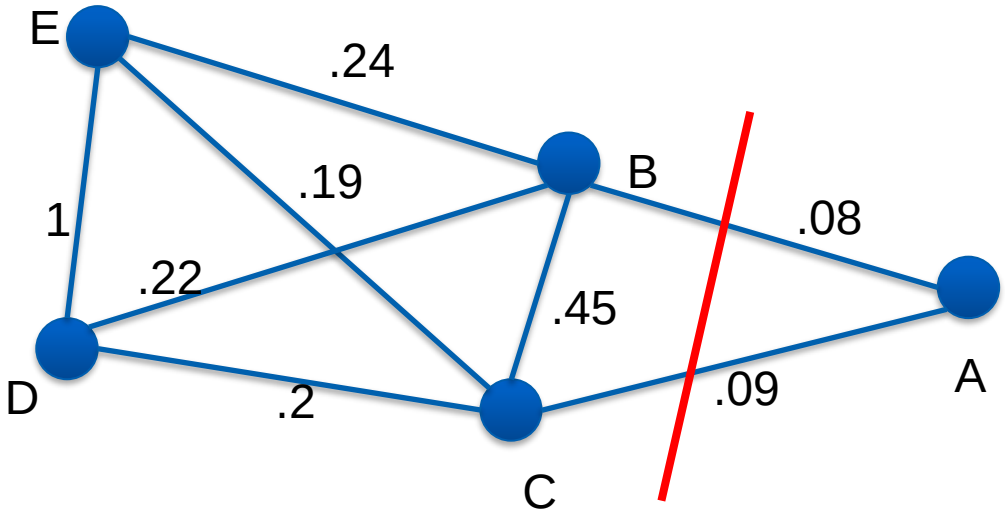
Agglomerative Clustering



Source: R. Akella

Spectral Clustering Example

- Minimum Cut

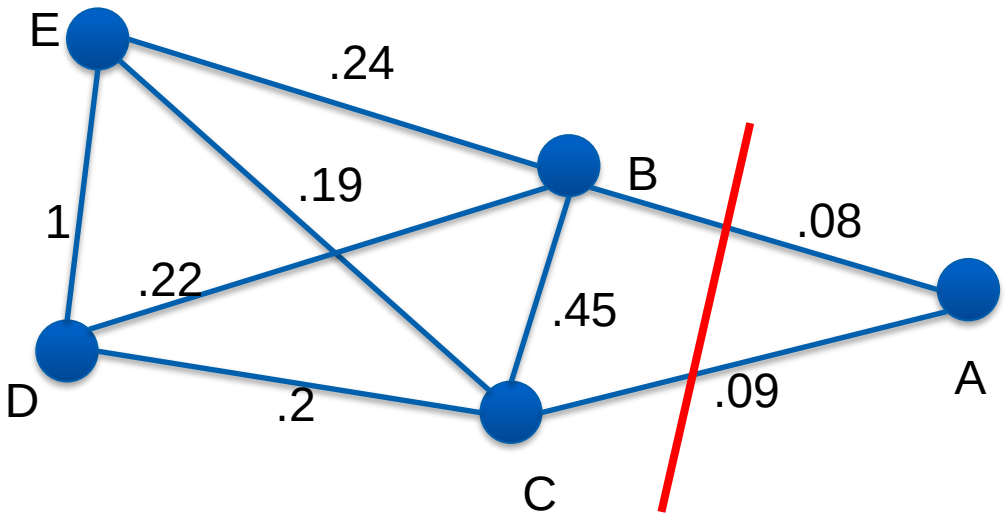


Cut(BCDE,A)=0.08+0.09=0.17

Spectral Clustering Example

- Normalized Minimum Cut

$$NormCut(C_1, C_2) = \frac{Cut(C_1, C_2)}{Vol(C_1)} + \frac{Cut(C_1, C_2)}{Vol(C_2)}$$

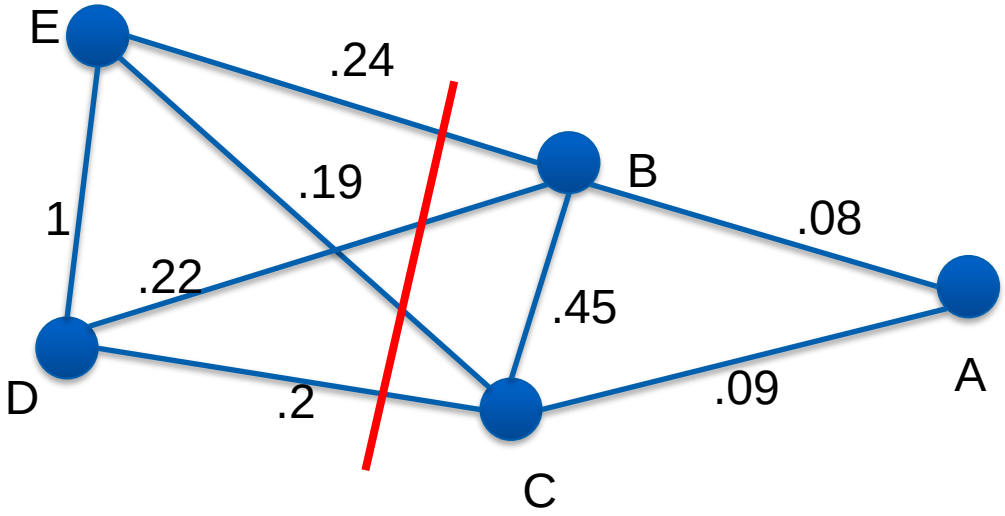


Cut(BCDE,A)=0.17/(0.99+0.93+1.42+1.43) + 0.17/0.17 = 1.036

Spectral Clustering Example

- Normalized Minimum Cut

$$NormCut(C_1, C_2) = \frac{Cut(C_1, C_2)}{Vol(C_1)} + \frac{Cut(C_1, C_2)}{Vol(C_2)}$$

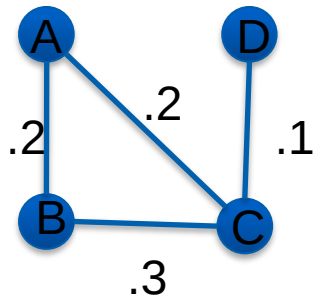


Cut(BCDE,A)=0.17/(0.99+0.93+1.42+1.43) + 0.17/0.17 = 1.036

Cut(DE,ABC)=0.85/(1.42+1.43) + 0.85/(0.17+0.99+0.93) = 0.705

Spectral Clustering

- Construct an **affinity** matrix



W

	A	B	C	D
A	0	.2	.2	0
B	.2	0	.3	0
C	.2	.3	0	.1
D	0	0	.1	0

- Graph Laplacian

$$L = D - W$$

	A	B	C	D
A	.4	-.2	-.2	-0
B	-.2	.5	-.3	-0
C	-.2	-.3	.6	-.1
D	0	0	-.1	.1

Spectral Clustering

UNIVERSITÄT BONN

Normalized Minimum Cut (Relaxed):

$$min_x Ncut(x) = min_y \frac{y^T(\mathbf{D}-\mathbf{W})y}{y^T\mathbf{D}y}$$

Generalized eigenvalue problem:

$$(\mathbf{D}-\mathbf{W})\boldsymbol{y} = \lambda \mathbf{D}y$$

Equal eigenvalue problem:

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}z = \lambda z \qquad z = \mathbf{D}^{\frac{1}{2}}\boldsymbol{y}$$

Eigenvector corresponding to the second smallest eigenvalue

More than two clusters, Laplacian eigenmaps

The eigenvector to the third smallest eigenvalue indicates an alternative partitioning of the graph

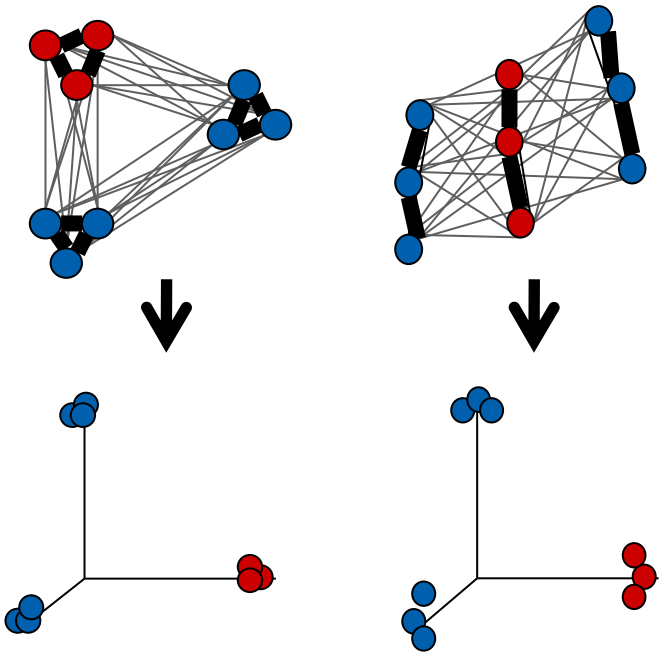
In the general case we compute the k smallest eigenvalues

The corresponding eigenvectors span a subspace

- $Y = (y_1, y_2, \dots, y_k) \in \mathbb{R}^{Nxk}$ matrix
- Each data point $1, 2, \dots, N$ maps to a k-dimensional vector (i-th row of Y)
- Mapping is called **Laplacian eigenmap**

Standard clustering techniques (k-means) to convert the real-valued vectors into integer labels

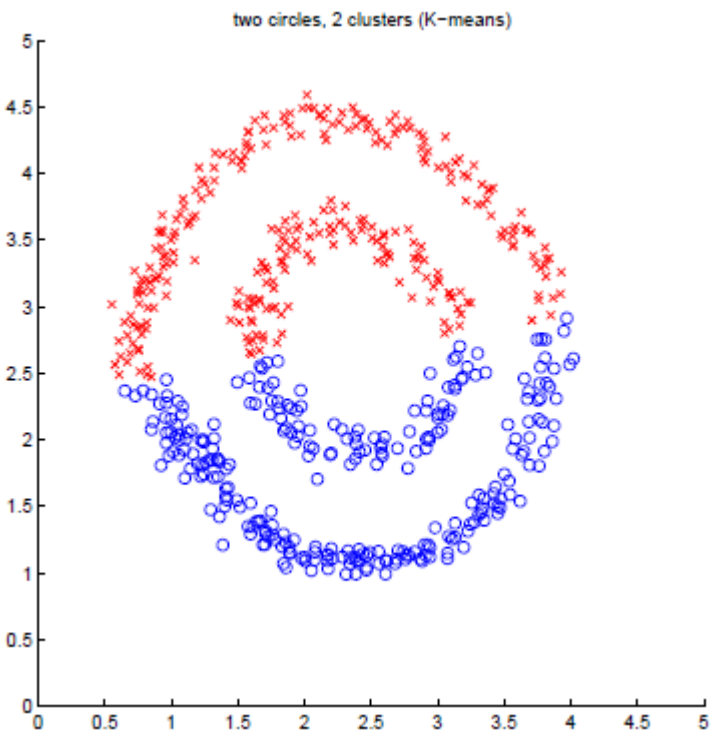
→ **spectral clustering**



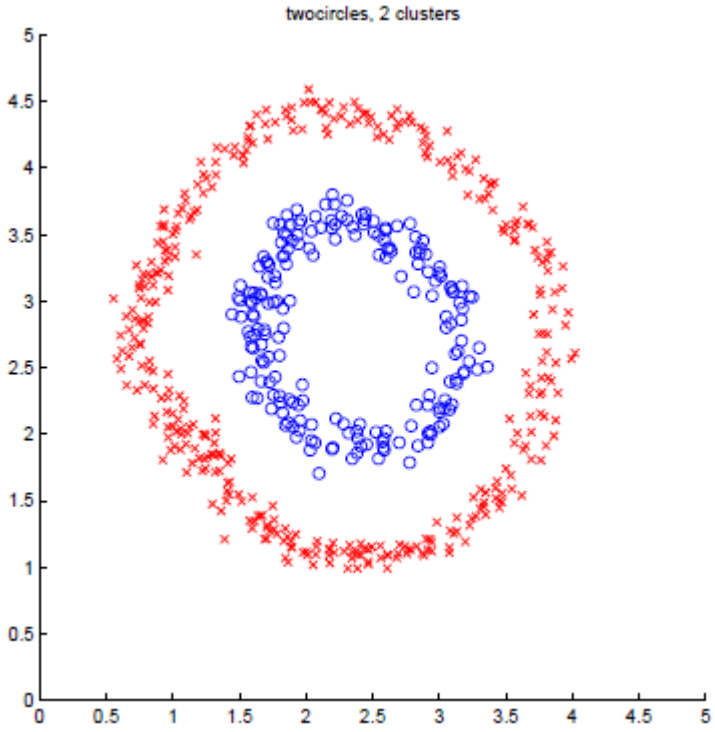
Mappings from a graph to its eigenspace, clustering in the eigenspace

[Ulrike von Luxburg. **A Tutorial on Spectral Clustering.** Statistics and Computing 2007]

K means vs. spectral clustering



K means



Spectral clustering

A. Ng et al. **On spectral clustering: Analysis and an algorithm.** NIPS 2001

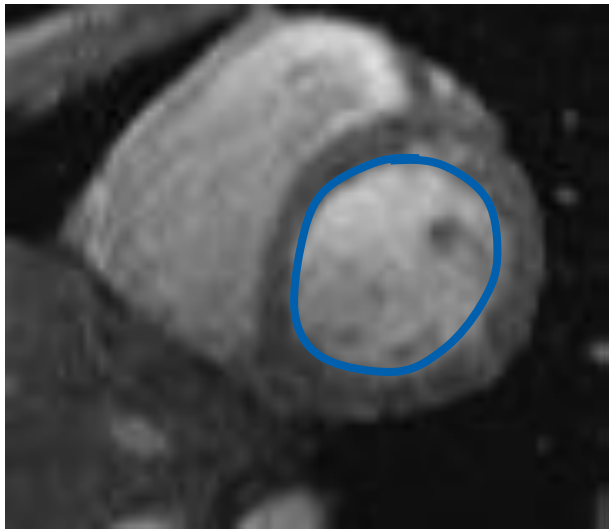
Deformable contours

UNIVERSITÄT BONN

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary



Main idea: elastic band is iteratively adjusted so as to

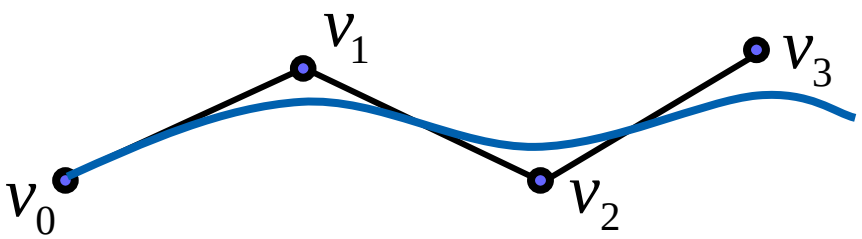
- be near image positions with high gradients, **and**
- satisfy shape “preferences” or contour priors

[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Source: Y. Boykov

B-spline snakes

Explicit representation of contour: $C(s) = \sum_{i=0}^{N-1} v_i \cdot B_i(s)$



Smooth (differentiable) curve passes close to the poly-line joining the knot (control) points

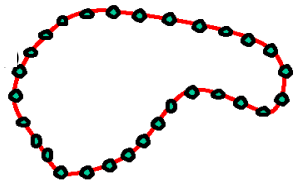
B-spline snakes - parametric representation of continuous contours via finite number of parameters:

$$\{v_0, v_1, v_2, \ldots, v_{N-1}\}$$

Source: Y. Boykov

Energy function

The total energy (cost) of the current snake is defined as:



$$E_{total} = E_{internal} + E_{external}$$

Internal energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

External energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour and the target shape in the image will yield a **low** value for this cost function.

Source: K. Grauman

Total energy: function of the weights

UNIVERSITÄT BONN

$$E_{total} = E_{internal} + \gamma E_{external}$$

$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left(\bar{d} - \|v_{i+1} - v_i\|\right)^2 + \beta \|v_{i+1} - 2v_i + v_{i-1}\|^2$$

Source: K. Grauman

The Snakes model (continuous)

Kass, Witkin, and Terzopoulos proposed the following energy functional:

$$E(C) = -\int_0^1 |\nabla I(C(s))|^2 ds + \alpha \int_0^1 |C_s(s)|^2 ds + \beta \int_0^1 |C_{ss}(s)|^2 ds$$

External energy (data) — Internal energy (prior)

$C : [0,1] \to \Omega$ is a parametric contour. C_s and C_{ss} denote the first and second derivative of this contour.

The first term is called **external energy**, since it depends on the (external) input image. The second two terms are called **internal energy**, since they are inherent to the model and independent of the data.

Minimizing the external energy drives the contour to go along maxima of the gradient.

[M. Kass, A. Witkin, D. Terzopoulos. Snakes: Active contour models. IJCV 1987]

Source: T. Brox

Energy minimization

UNIVERSITÄT BONN

- Several algorithms have been proposed to fit deformable contours.
- We'll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)
  - Variational optimization

Source: K. Grauman

Dynamic programming

- General minimization problem with unary U_n and pairwise terms P_n:

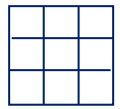
$$\text{argmin}_{w_{1\ldots N}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{n=1}^{N-1} P_n(w_n, w_{n+1}) \right]$$

- Snakes with control points w_n:

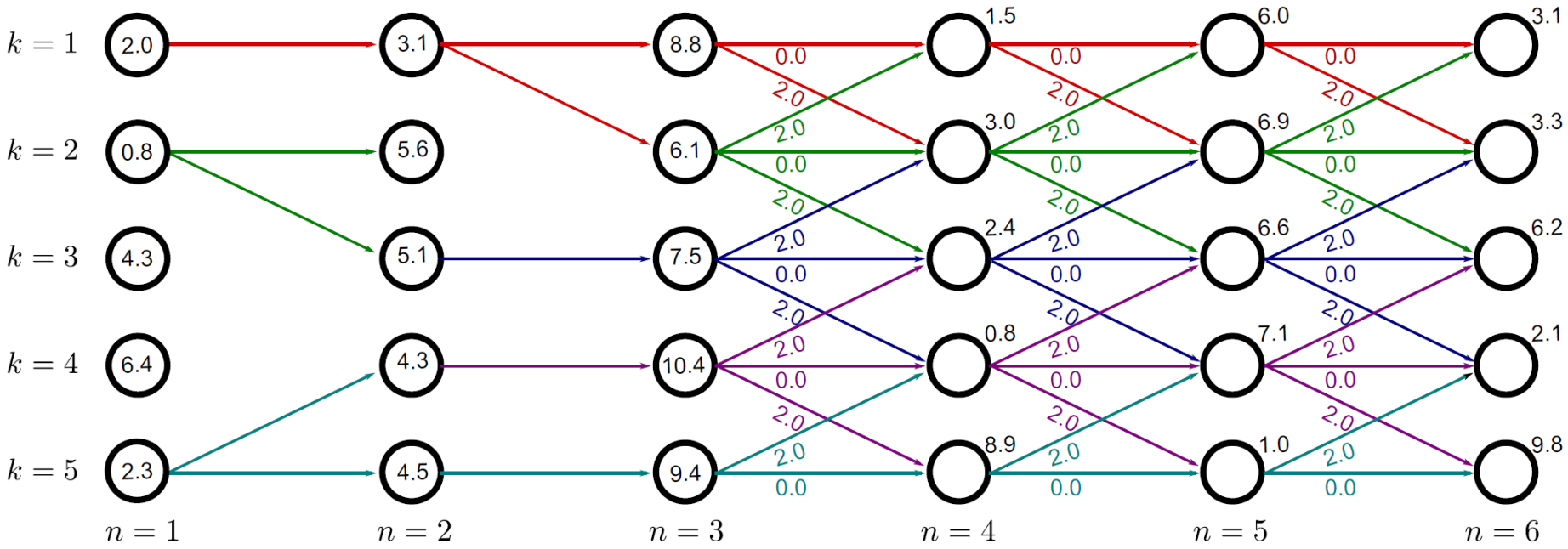
$$U_n(w_n) = -\|G(w_n)\|^2$$

$$P_n(w_n, w_{n+1}) = \alpha \|w_{n+1} - w_n\|^2$$

- States of w_n=k are pixels in neighborhood:

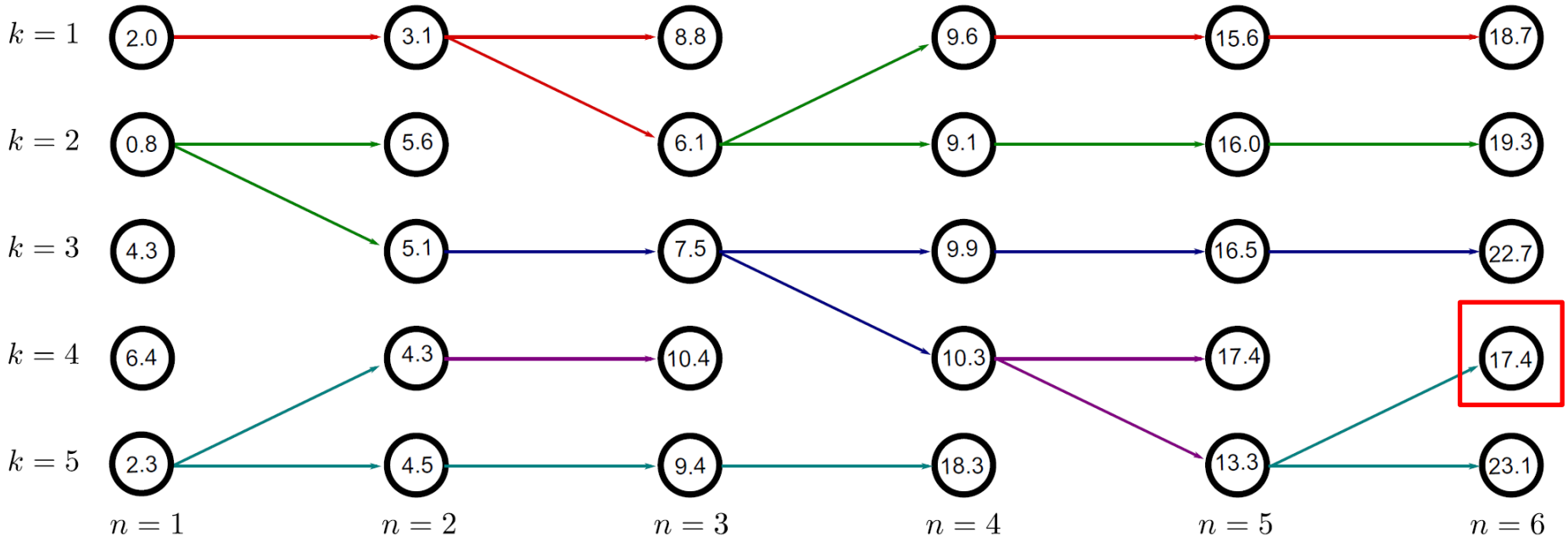


Worked example



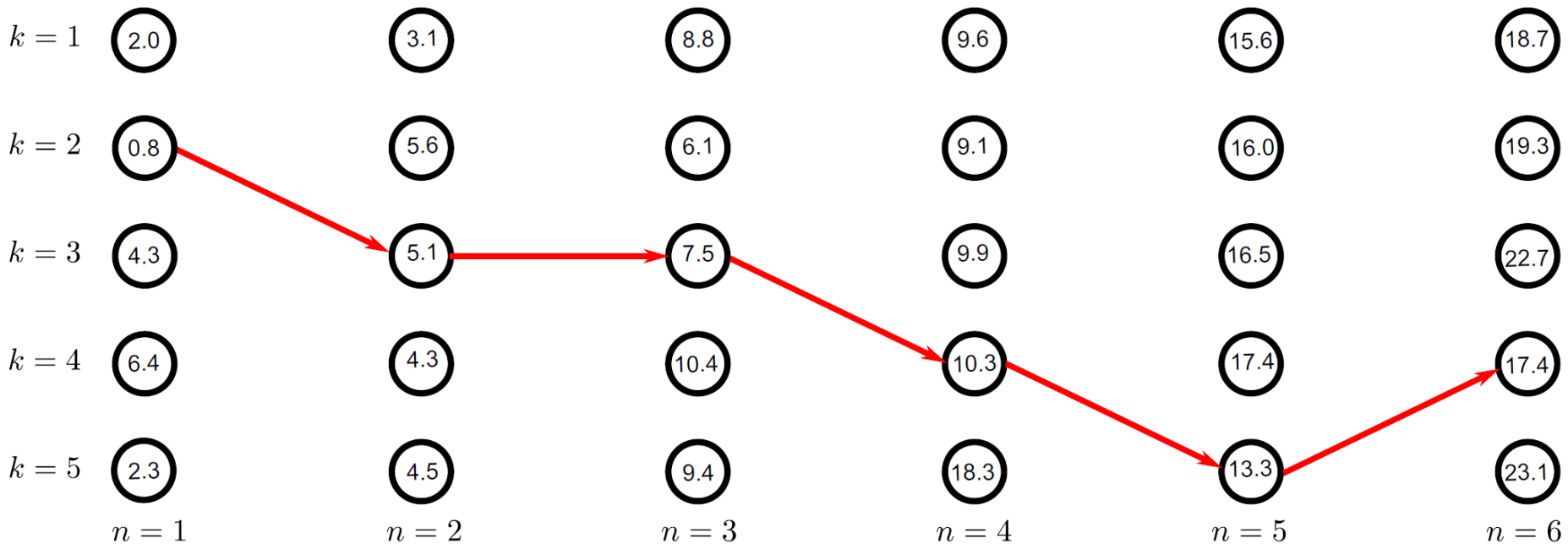
Work through the graph, computing the minimum cost to reach each node

Worked example



Find the minimum possible cost to reach the final column

Worked example



Trace back the route that we arrived here by – this is the minimum configuration

Simon J.D. Prince

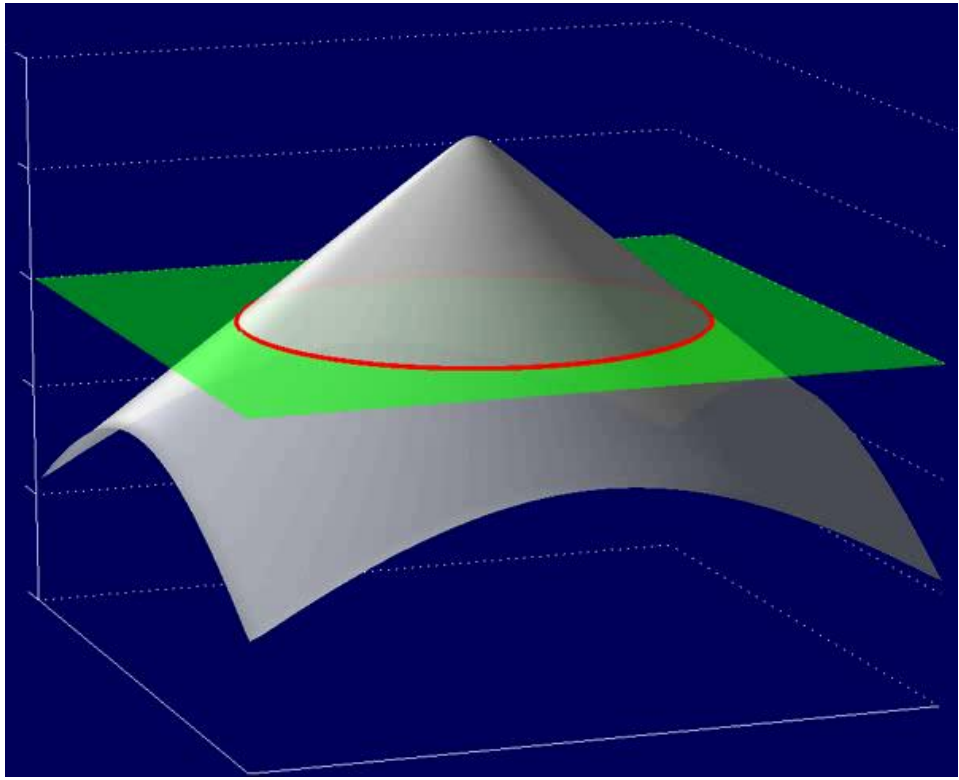
Level set representation of contours

UNIVERSITÄT BONN

- Introduce an **embedding function** $\phi : \Omega \rightarrow \mathbb{R}$
- The zero-level line represents the contour (can be any other level)

$$C = \{\mathbf{x} \in \Omega | \ \phi(\mathbf{x}) = 0\}$$

- For evolving C evolve ϕ
- No regridding necessary
- Allows for topological changes
- Can be applied in any dimension
- Represents the contour *and* the enclosed region



Source: D. Cremers

Geodesic active contours

Caselles et al**. Geodesic Active Contours.** IJCV 1997 introduced two improvements of the snake model.

1. A level set representation of the contour

2. A variation of the model, called the **geodesic active contour**:

$$E(C) = \int_0^1 w(|\nabla I(C(s))|^2)|C_s(s)|ds = \int_C w(|\nabla I|^2)ds$$

It consists of a single term that penalizes the **geodesic length** of the contour using a metric induced by the image gradient.

For comparison: measuring the Euclidean length of a contour reads:

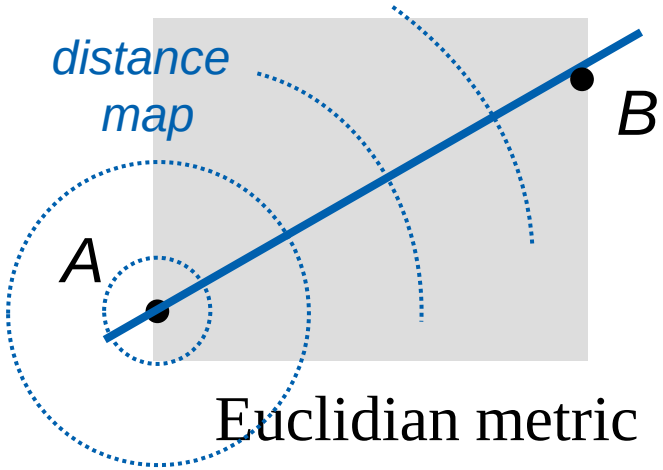
$$E(C) = \int_0^1 |C_s(s)|ds = \int_C ds$$

→ Combines the internal and external energy of the Snake model

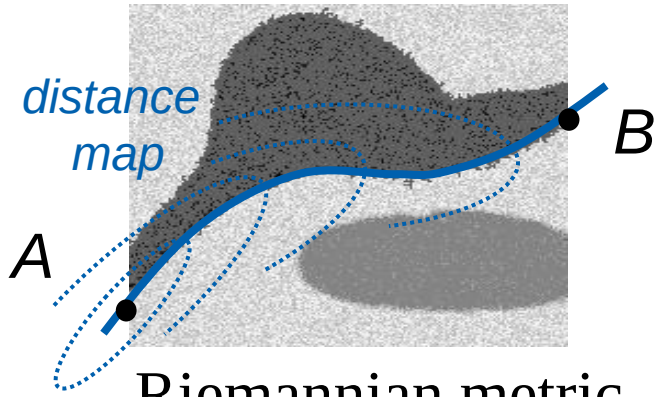
Source: T. Brox

Geodesic active contours

- Energy $E(C) = \int_A^B g(|\nabla I_{C(s)}|)\,ds$ = "weighted length" of C
- *Geodesic:* Shortest curve between two points (wrt. E).



Euclidian metric
(constant $g() = 1$)



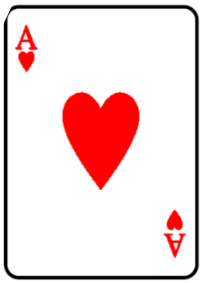
Riemannian metric
(space varying $g(|\nabla I|) \geq 0$)

$E(C)$ = Euclidean length of C

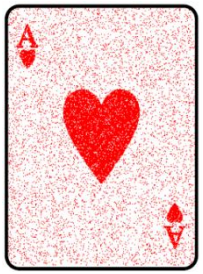
$E(C)$ = image-weighted length of C

Source: Y. Boykov

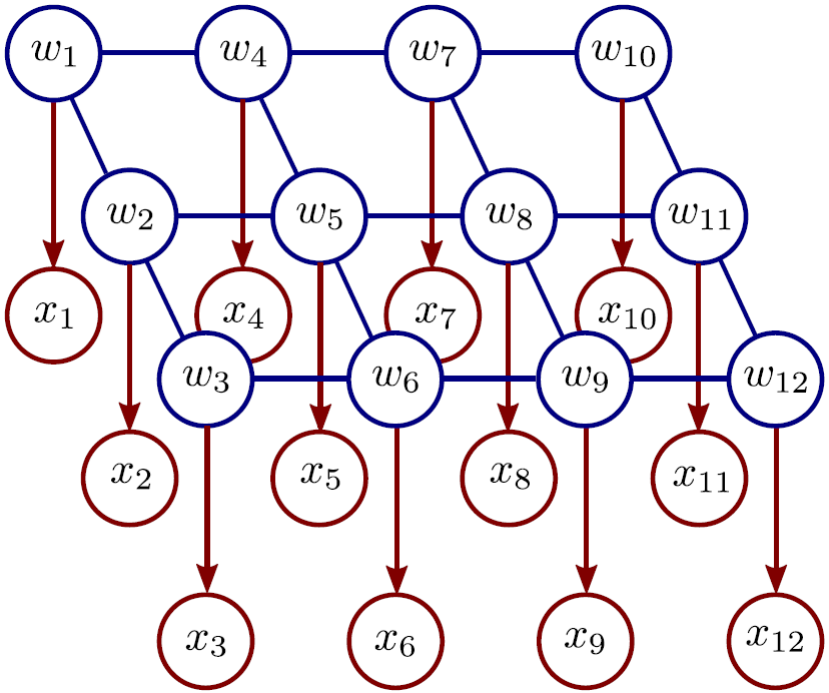
Denoising with MRFs



Original image, **w**



Observed image, **x**



MRF Prior (pairwise cliques)

$$Pr(w_{1\ldots N}) = \frac{1}{Z}\exp\left[-\sum_{(m,n)\in\mathcal{C}}\psi[w_m, w_n, \boldsymbol{\theta}]\right]$$

Likelihoods

$$Pr(x_n|w_n = 0) = \text{Bern}_{x_n}[\rho]$$
$$Pr(x_n|w_n = 1) = \text{Bern}_{x_n}[1-\rho]$$

Inference : $$Pr(w_{1\ldots N}|x_{1\ldots N}) = \frac{\prod_{n=1}^{N} Pr(x_n|w_n)Pr(w_{1\ldots N})}{Pr(x_{1\ldots N})}$$

Graph Cuts Overview

Graph cuts used to optimise this cost function:

$$\underset{w_{1\ldots N}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{(m,n)\in\mathcal{C}} P_{mn}(w_m, w_n) \right]$$

Unary terms
(compatability of data with label y)

Pairwise terms
(compatability of neighboring labels)

Approach:

Convert minimization into the form of a standard CS problem,

MAXIMUM FLOW or MINIMUM CUT ON A GRAPH

Polynomial-time methods for solving this problem are known

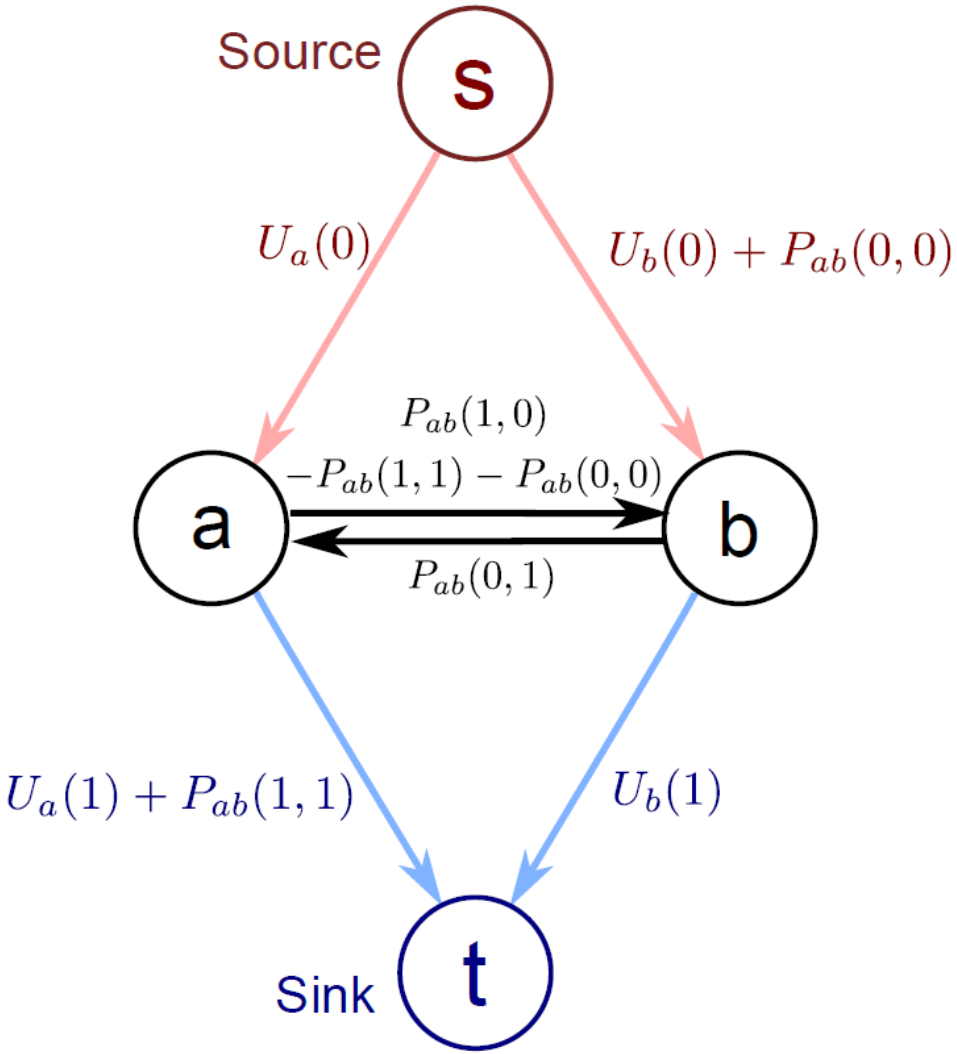
General Pairwise costs

$$P_{m,n}(0,0) = \theta_{00} \quad P_{m,n}(1,0) = \theta_{10}$$
$$P_{m,n}(0,1) = \theta_{01} \quad P_{m,n}(1,1) = \theta_{11}$$

Modify graph to

- Add P(0,0) to edge s-b
  - Implies that solutions 0,0 and 1,0 also pay this cost
- Subtract P(0,0) from edge b-a
  - Solution 1,0 has this cost removed again

Similar approach for P(1,1)



Submodularity

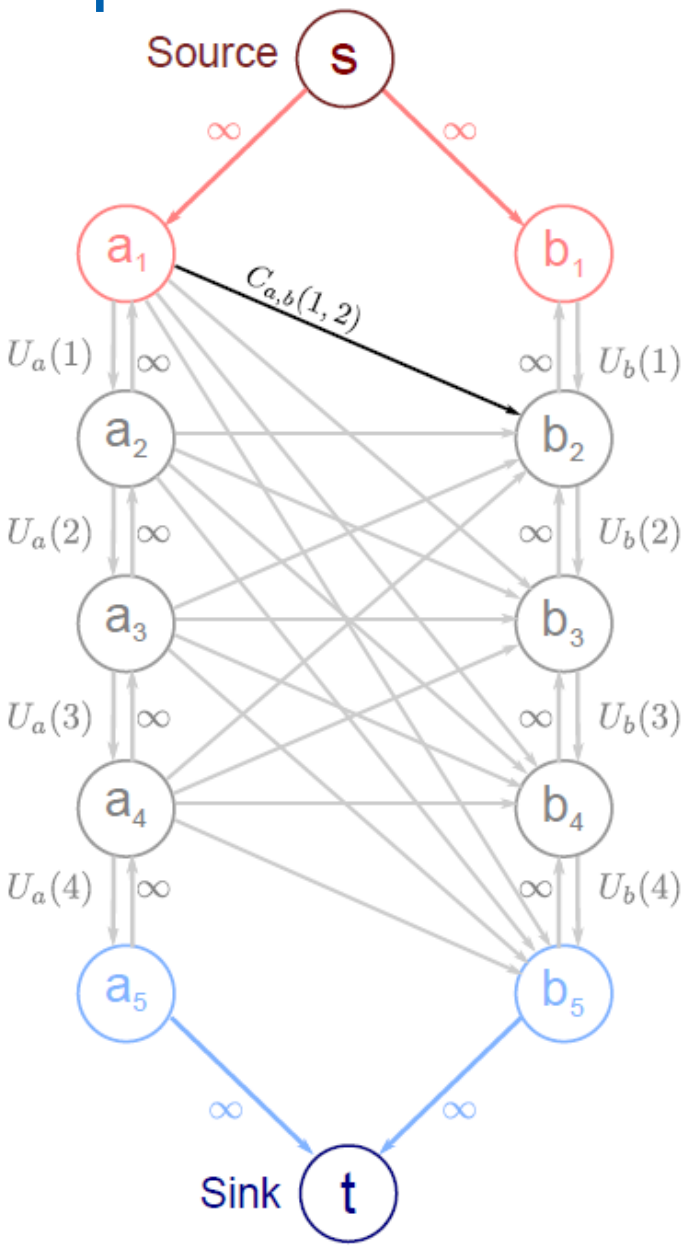
$$\theta_{10} + \theta_{01} - \theta_{11} - \theta_{00} \geq 0$$

If this condition is obeyed, it is said that the problem is “submodular” and it can be solved in polynomial time.

If it is not obeyed then the problem is NP hard.

Usually it is not a problem as we tend to favour smooth solutions.

Multiple Labels



Inter-pixel edges have costs defined as:

$$C_{ab}(i,j) =$$

$$P_{ab}(i,j-1) + P_{ab}(i-1,j)$$

$$- P_{a,b}(i,j) - P_{ab}(i-1,j-1)$$

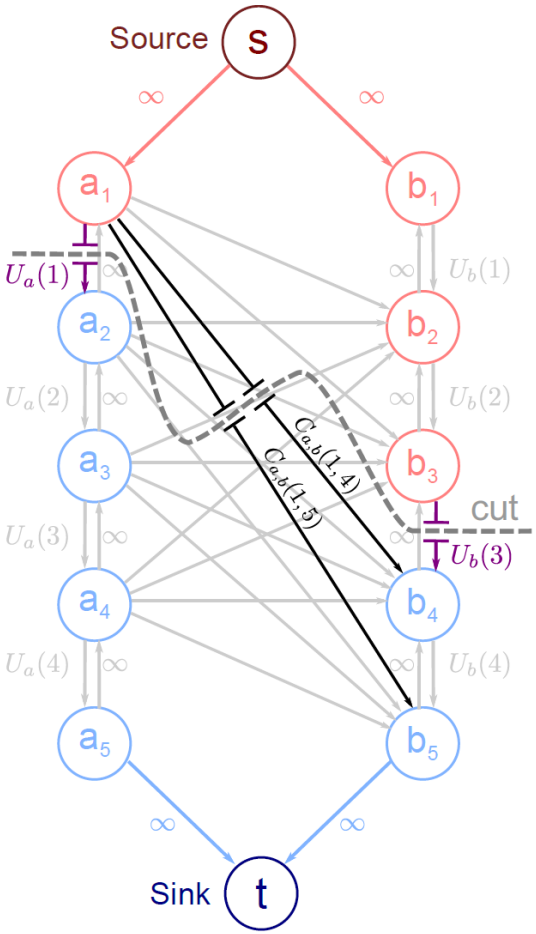
Superfluous terms :

$$P_{ab}(i,0) = 0 \qquad P_{ab}(i,K+1) = 0$$

$$P_{ab}(0,j) = 0 \qquad P_{ab}(K+1,j) = 0$$

For all i,j where K is number of labels

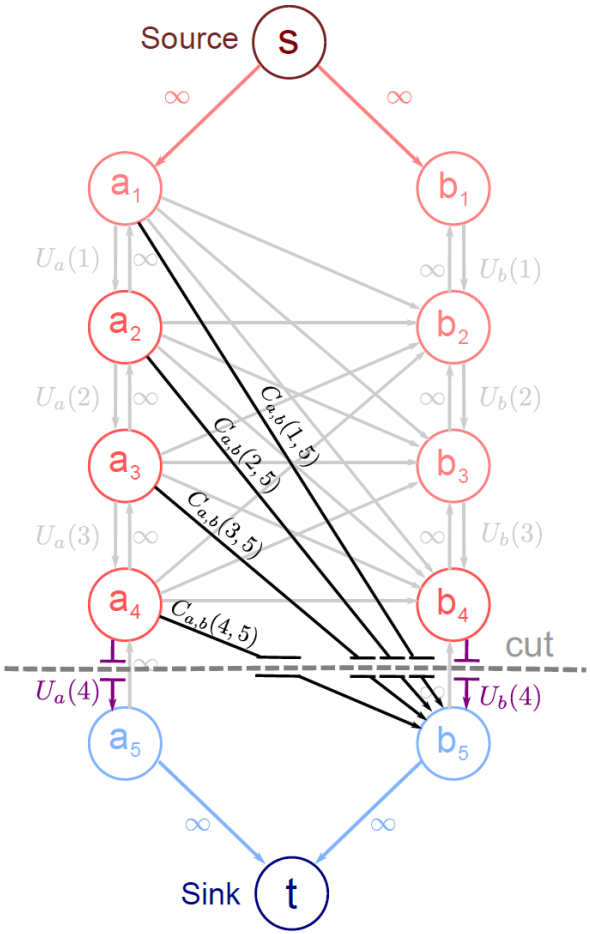
Example Cuts



1	3

Cost

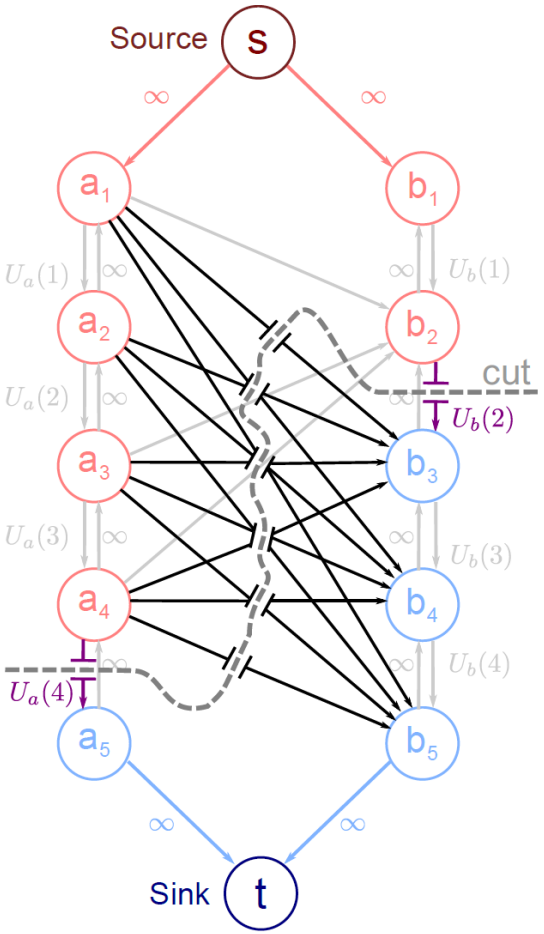
$$U_a(1) + U_b(3) + \sum_{i=1}^{1} \sum_{j=4}^{5} C_{ab}(i,j)$$
$$= U_a(1) + U_b(3) + P_{ab}(1,3)$$



4	4

Cost

$$U_a(4) + U_b(4) + \sum_{i=1}^{4} \sum_{j=5}^{5} C_{ab}(i,j)$$
$$= U_a(4) + U_b(4) + P_{ab}(4,4)$$



4	2

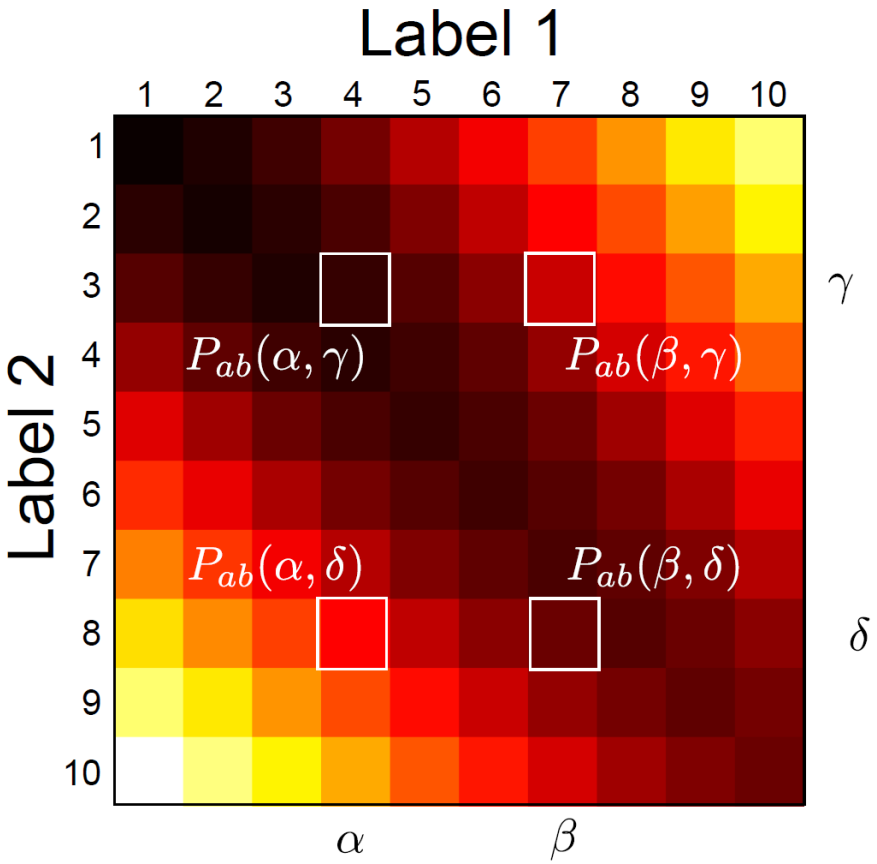
Cost

$$U_a(4) + U_b(2) + \sum_{i=1}^{4} \sum_{j=3}^{5} C_{ab}(i,j)$$
$$= U_a(4) + U_b(2) + P_{ab}(4,2)$$

Must cut links from before cut on pixel a to after cut on pixel b.

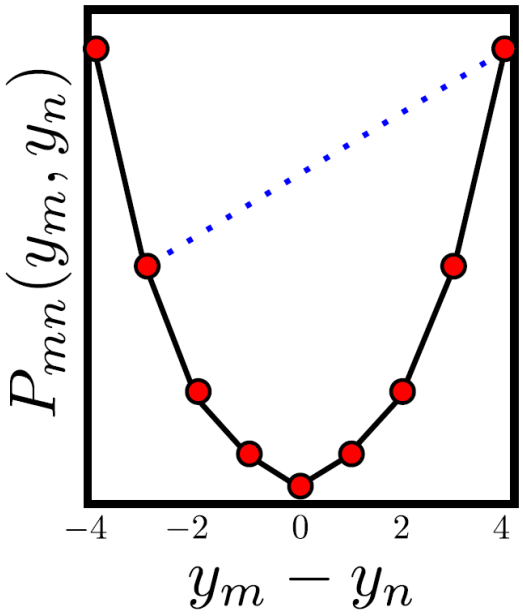
Submodularity

$$P_{ab}(\beta,\gamma) + P_{ab}(\alpha,\delta) - P_{ab}(\beta,\delta) - P_{ab}(\alpha,\gamma) \geq 0 \quad \beta > \alpha \text{ and } \delta > \gamma$$



If not submodular, then the problem is NP hard.

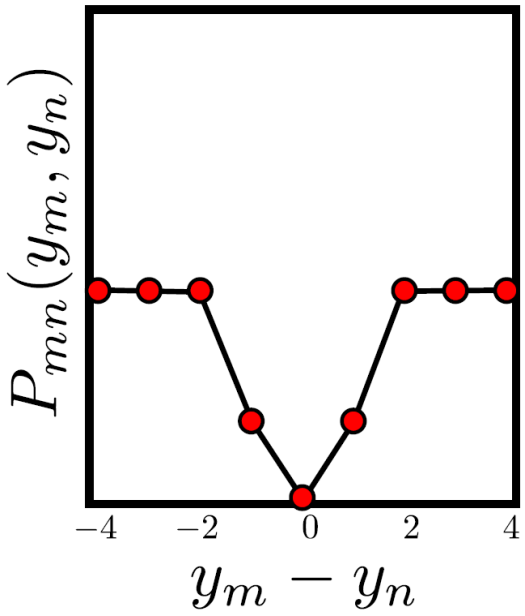
Convex vs. non-convex costs



Quadratic

- Convex
- Submodular

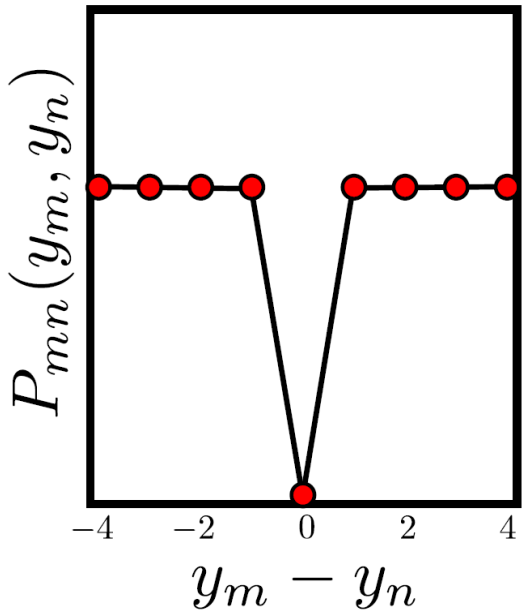
$$\kappa(w_m - w_n)^2$$



Truncated Quadratic

- Not Convex
- Not Submodular

$$\min(\kappa_1, \kappa_2(w_m - w_n)^2)$$



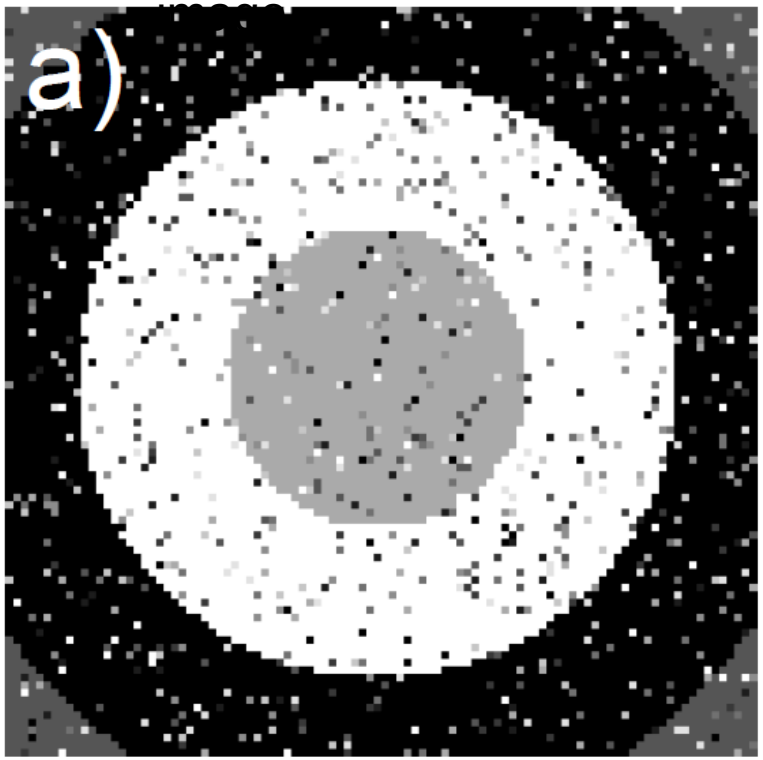
Potts Model

- Not Convex
- Not Submodular

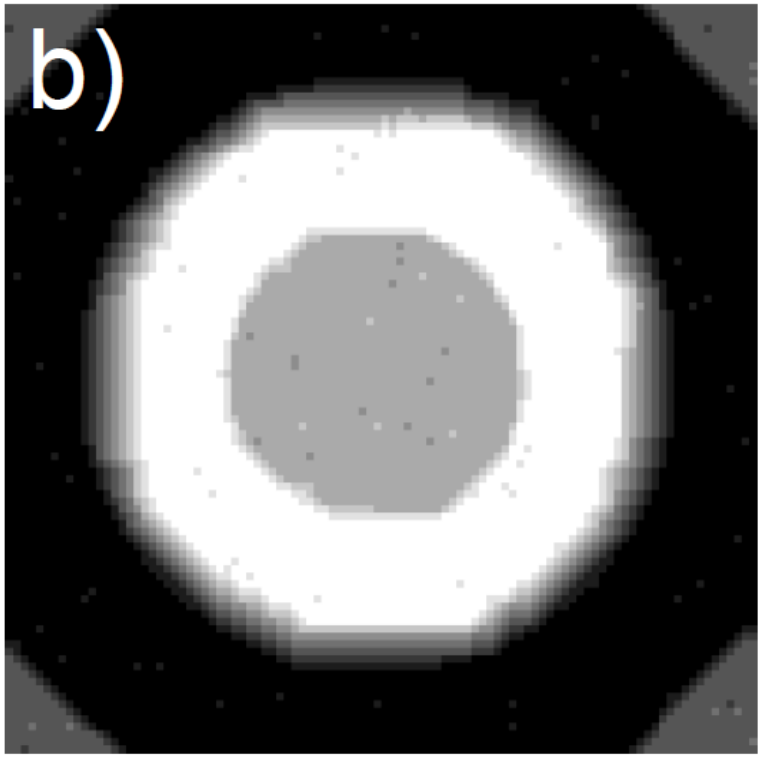
$$\kappa(1 - \delta(w_m - w_n))$$

What is wrong with convex costs?

Observed noisy image



Denoised result



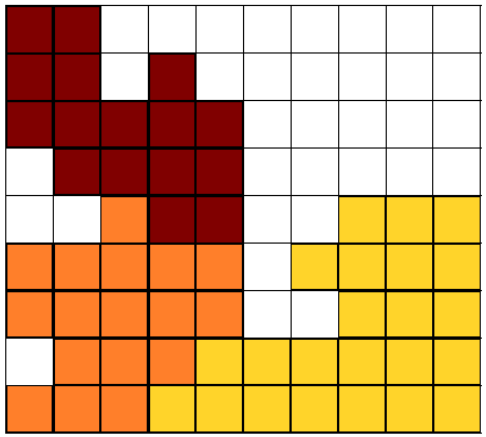
- Pay lower price for many small changes than one large one
- Result: blurring at large changes in intensity

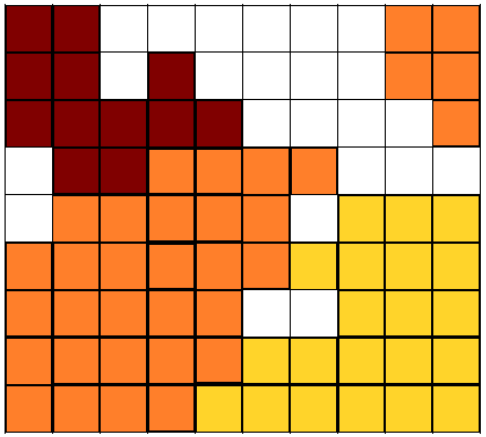
Alpha Expansion Algorithm

UNIVERSITÄT BONN

- break multilabel problem into a series of binary problems
- at each iteration, pick label α and expand (retain original or change to α)

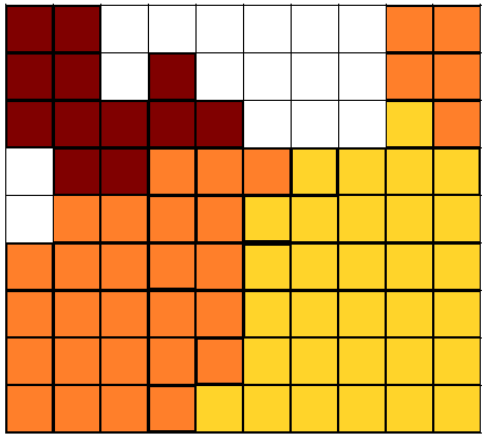
Initial labelling

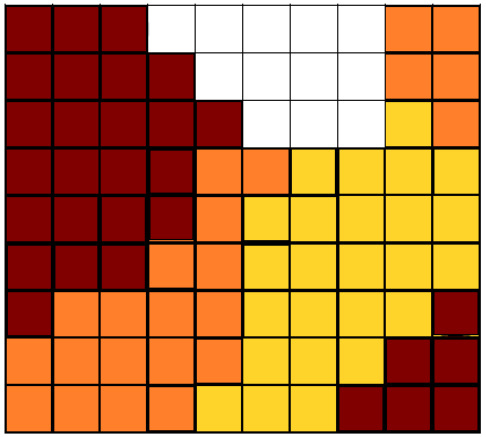




Iteration 1 (orange)

Iteration 2 (yellow)





Iteration 3 (red)

Alpha Expansion Ideas

- For every iteration
  - For every label
  - Expand label using optimal graph cut solution

Co-ordinate descent in label space.

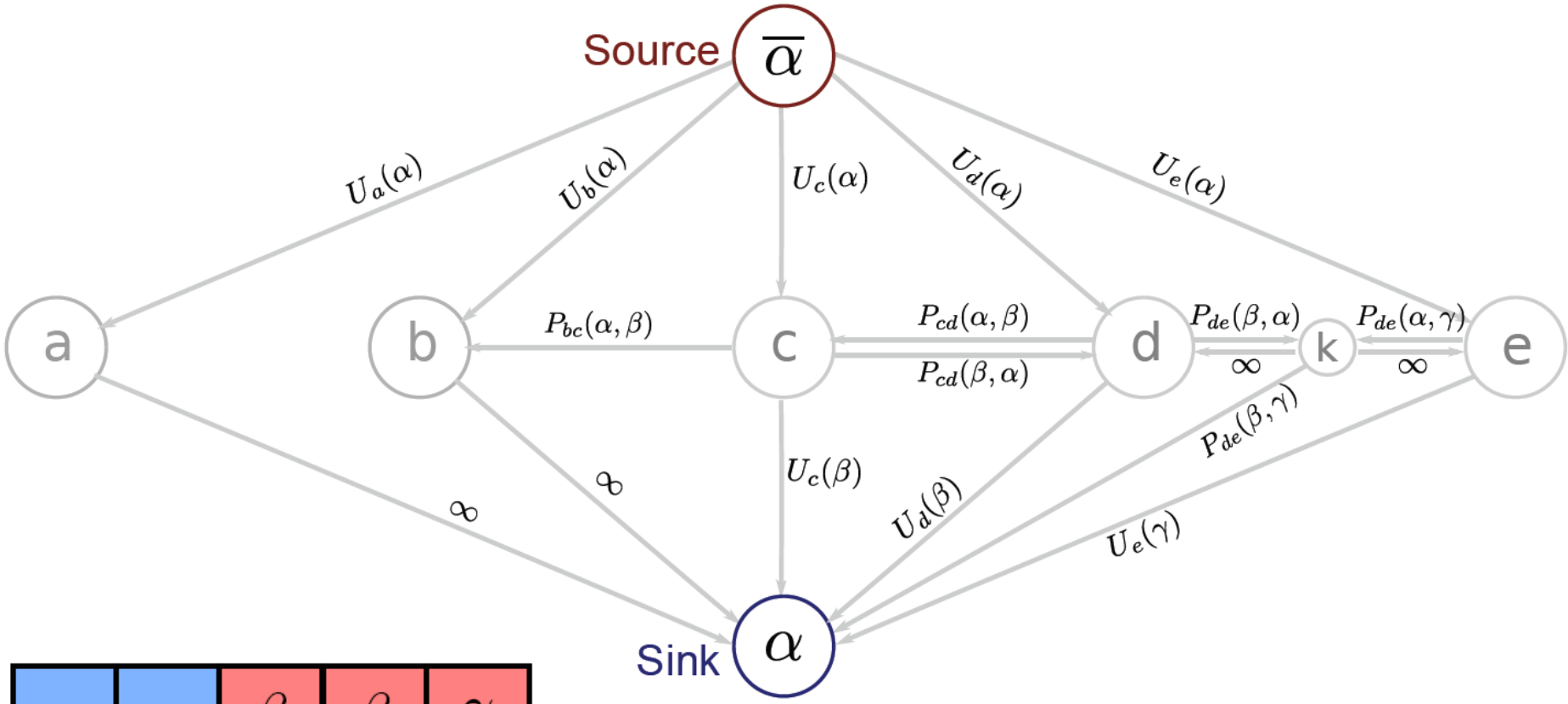
Each step optimal, but overall global maximum not guaranteed

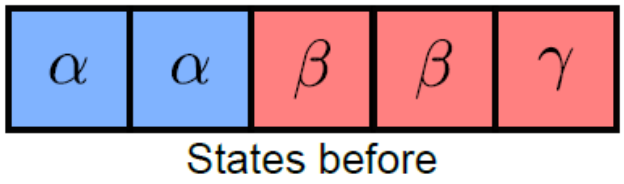
Proved to be within a factor of 2 of global optimum.

Requires that pairwise costs form a metric:

$$\begin{aligned} P(\alpha,\beta) &= 0 \quad \Leftrightarrow \alpha = \beta \\ P(\alpha,\beta) &= P(\beta,\alpha) \geq 0 \\ P(\alpha,\beta) &\leq P(\alpha,\gamma) + P(\gamma,\beta) \end{aligned}$$

Alpha Expansion Construction



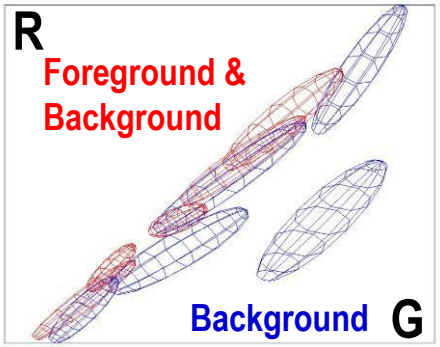


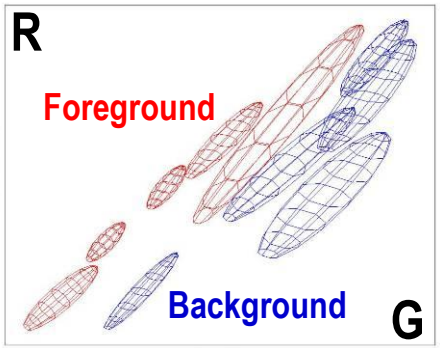
Binary graph cut – either cut link to source (assigned to α) or to sink (retain current label)

Unary costs attached to links between source, sink and pixel nodes appropriately.

Grab Cut

- Loosely specify foreground region
- Iterated graph cut





Gaussian Mixture Model (typically 5-8 components)

Source: K. Grauman

Expectation Maximization

An algorithm specialized to fitting pdfs which are the marginalization of a joint distribution

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log \left[\int Pr(\mathbf{x}_i, \mathbf{h}_i | \boldsymbol{\theta}) \, d\mathbf{h}_i \right] \right]$$

Defines a lower bound on log likelihood and increases bound iteratively

$$\begin{aligned} \mathcal{B}\left[\{q_i(\mathbf{h}_i)\}, \boldsymbol{\theta}\right] &= \sum_{i=1}^{I} \int q_i(\mathbf{h}_i) \log \left[\frac{Pr(\mathbf{x}, \mathbf{h}_i | \boldsymbol{\theta})}{q_i(\mathbf{h}_i)} \right] d\mathbf{h}_{1\ldots I} \\ &\leq \sum_{i=1}^{I} \log \left[\int Pr(\mathbf{x}, \mathbf{h} | \boldsymbol{\theta}) d\mathbf{h}_{1\ldots I} \right]. \end{aligned}$$

Source: S. Prince

E-Step & M-Step

E-Step – Maximize bound w.r.t. distributions $q_i(\mathbf{h}_i)$

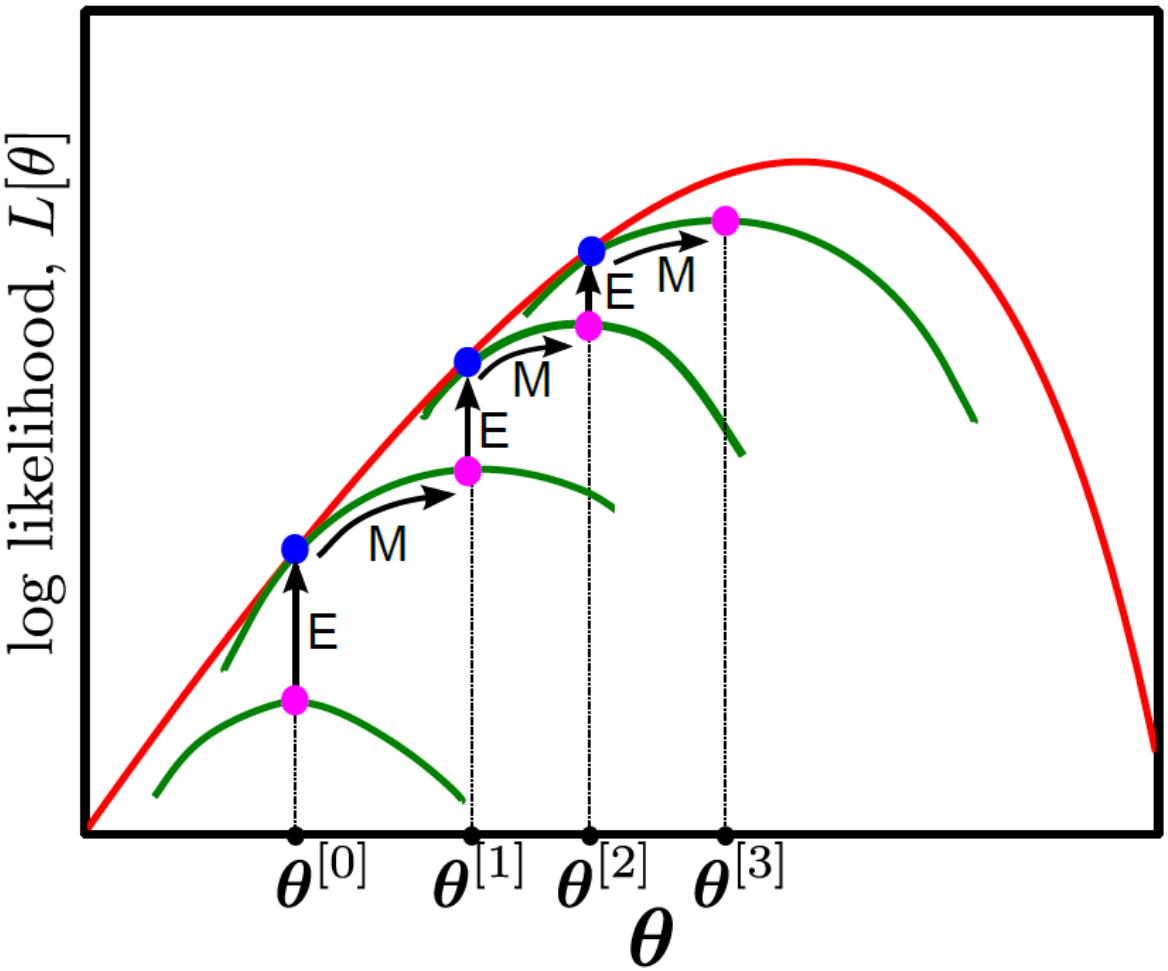
$$\hat{q}_i(\mathbf{h}_i) = Pr(\mathbf{h}_i|\mathbf{x}_i, \boldsymbol{\theta}^{[t]}) = \frac{Pr(\mathbf{x}_i|\mathbf{h}_i, \boldsymbol{\theta}^{[t]})Pr(\mathbf{h}_i|\boldsymbol{\theta}^{[t]})}{Pr(\mathbf{x}_i)}$$

M-Step – Maximize bound w.r.t. parameters θ

$$\hat{\boldsymbol{\theta}}^{[t+1]} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \int \hat{q}_i(\mathbf{h}_i) \log \left[Pr(\mathbf{x}_i, \mathbf{h}_i|\boldsymbol{\theta}) \right] \, d\mathbf{h}_i \right]$$

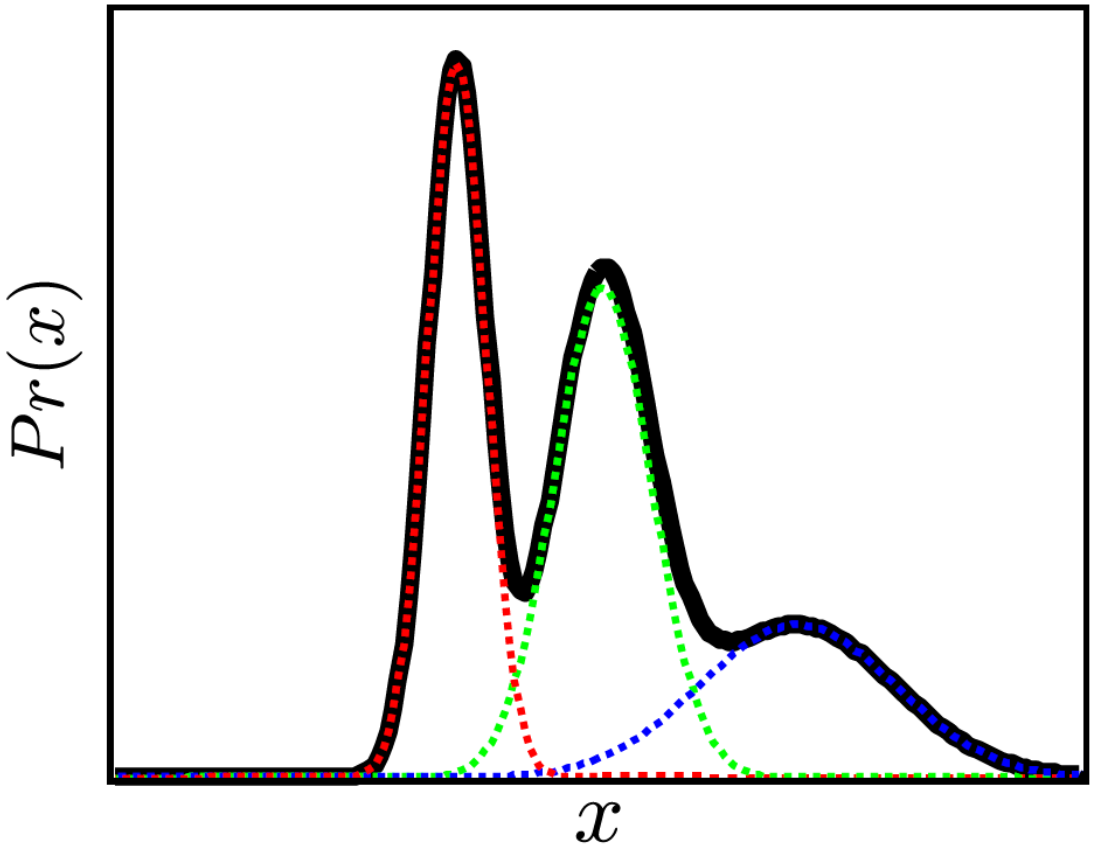
Source: S. Prince

E-Step & M-Step



Source: S. Prince

Mixture of Gaussians (MoG)



$$Pr(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \lambda_k \mathrm{Norm}_{\mathbf{x}}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]$$

Source: S. Prince

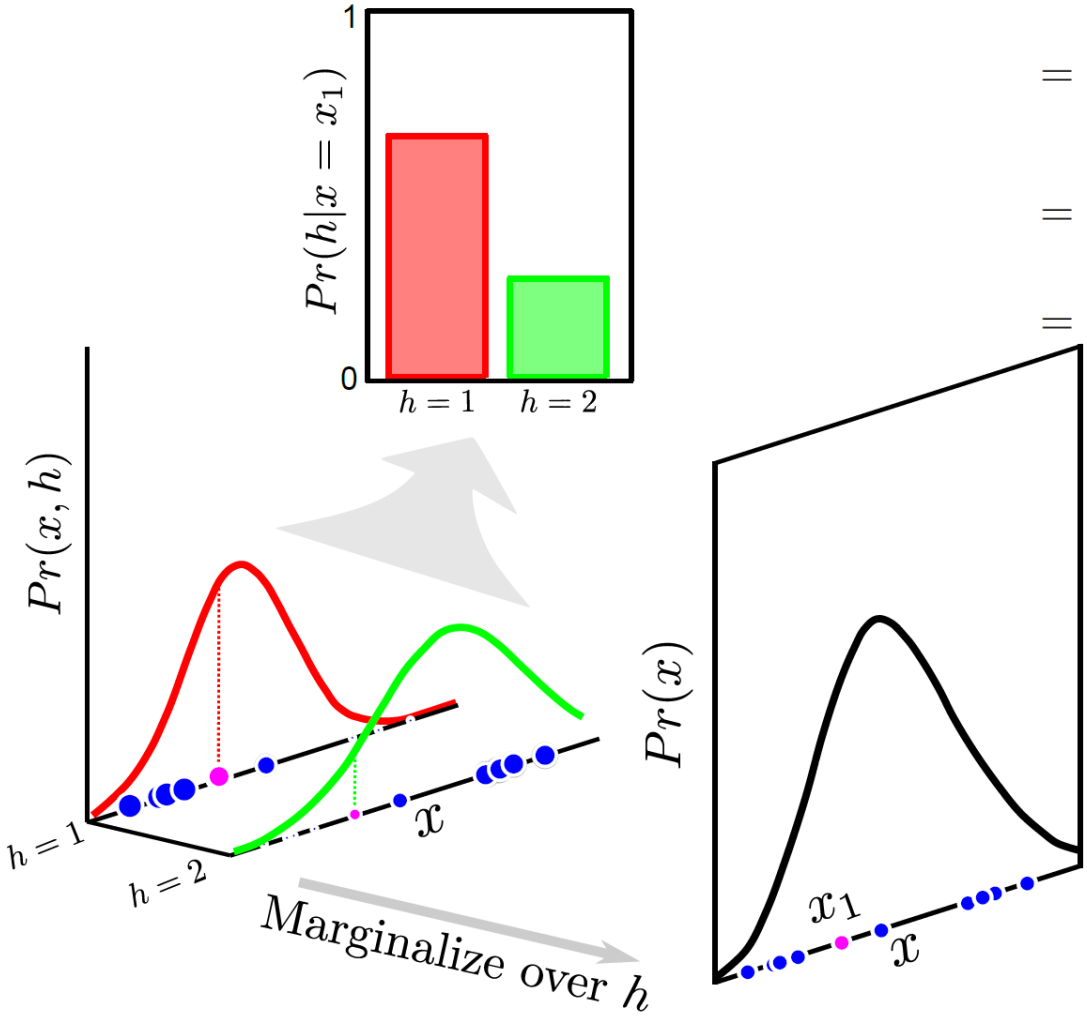
E-Step

$$Pr(h_i = k|\mathbf{x}_i, \boldsymbol{\theta}^{[t]})$$

$$= \frac{Pr(\mathbf{x}_i|h_i = k, \boldsymbol{\theta}^{[t]})Pr(h_i = k, \boldsymbol{\theta}^{[t]})}{\sum_{j=1}^{K} Pr(\mathbf{x}_i|h_i = j, \boldsymbol{\theta}^{[t]})Pr(h_i = j, \boldsymbol{\theta}^{[t]})}$$

$$= \frac{\lambda_k \mathrm{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k]}{\sum_{j=1}^{K} \lambda_j \mathrm{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j]}$$

$$= r_{ik}$$



We'll call this the responsibility of the k^{th} Gaussian for the i^{th} data point

Repeat this procedure for every datapoint!

Source: S. Prince

M-Step

$$\begin{aligned}\hat{\boldsymbol{\theta}}^{[t+1]} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{k=1}^{K} \hat{q}_i(h_i = k) \log \left[Pr(\mathbf{x}_i, h_i = k | \boldsymbol{\theta}) \right] \right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{k=1}^{K} r_{ik} \log \left[\lambda_k \mathrm{Norm}_{\mathbf{x}_i}[\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k] \right] \right].\end{aligned}$$

Take derivative, equate to zero and solve (Lagrange multipliers for λ)

$$\begin{aligned}\lambda_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik}}{\sum_{j=1}^{K} \sum_{i=1}^{I} r_{ij}} \\ \boldsymbol{\mu}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik} \mathbf{x}_i}{\sum_{i=1}^{I} r_{ik}} \\ \boldsymbol{\Sigma}_k^{[t+1]} &= \frac{\sum_{i=1}^{I} r_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})(\mathbf{x}_i - \boldsymbol{\mu}_k^{[t+1]})^T}{\sum_{i=1}^{I} r_{ik}}\end{aligned}$$

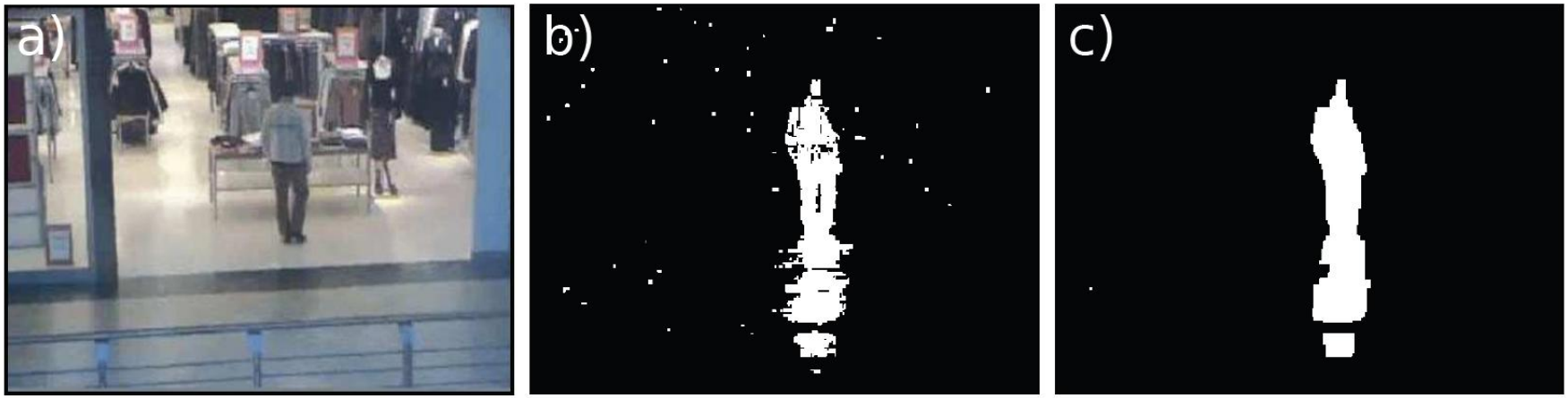
Source: S. Prince

Background subtraction

Model background distribution of each pixel by GMM:

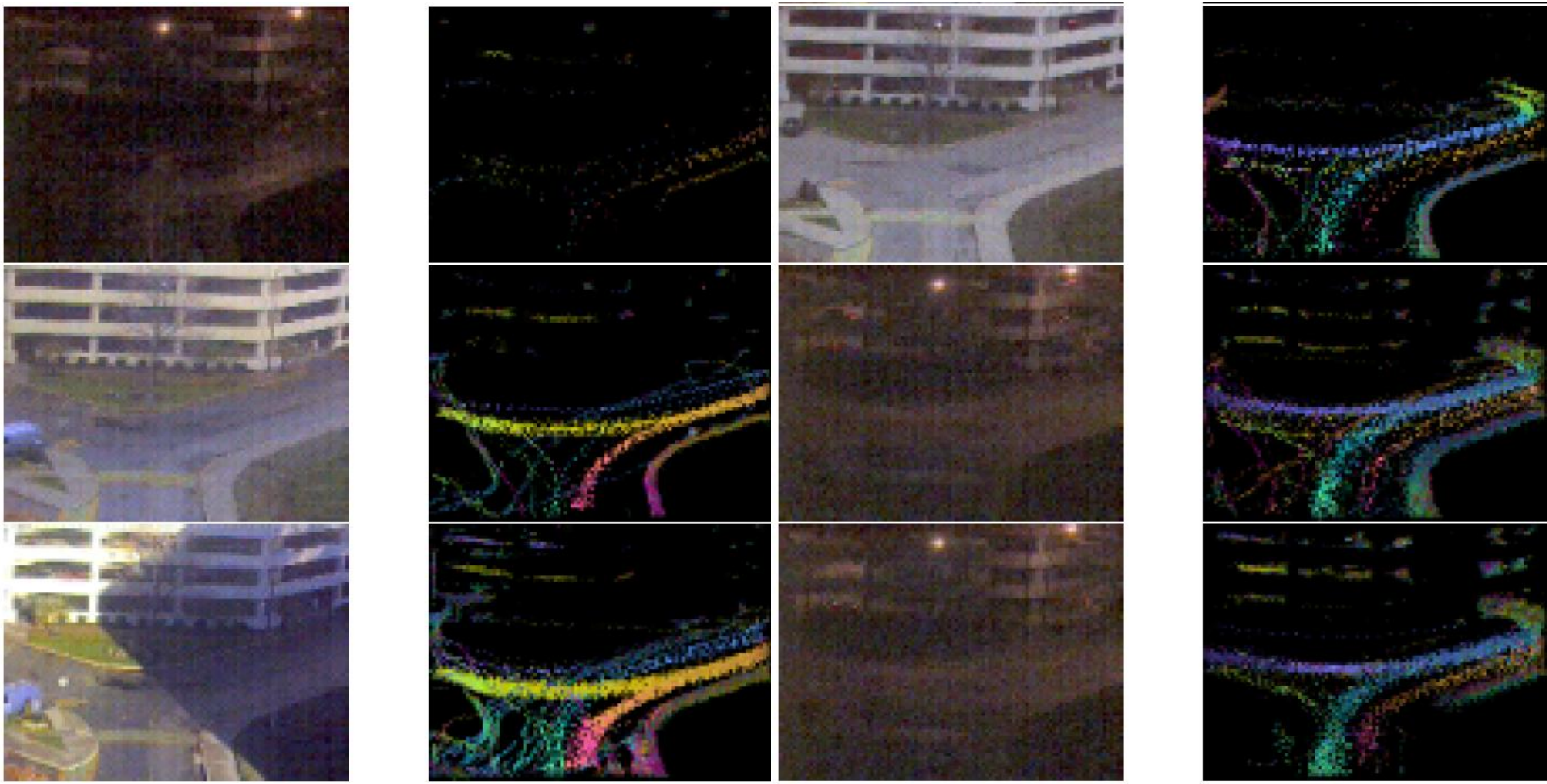
$$Pr(x_n|w=0) = \sum_i \lambda_{n,i} \mathrm{Norm}_{x_n}[\mu_{n,i}, \Sigma_{n,i}]$$

$$Pr(\mathbf{x}_n|w=1) = \kappa,$$



Test image Pixel-wise classification MRF

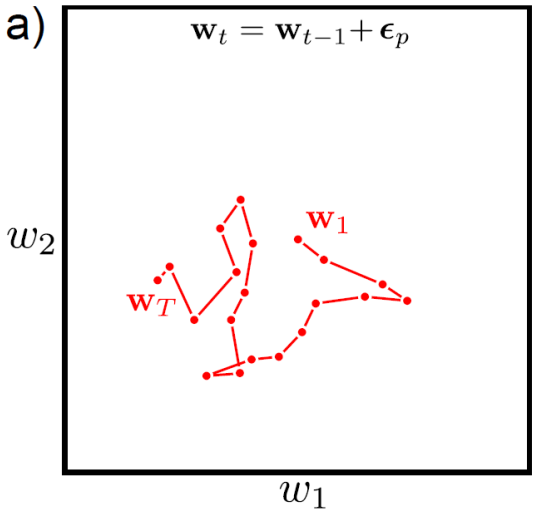
Cars and pedestrians

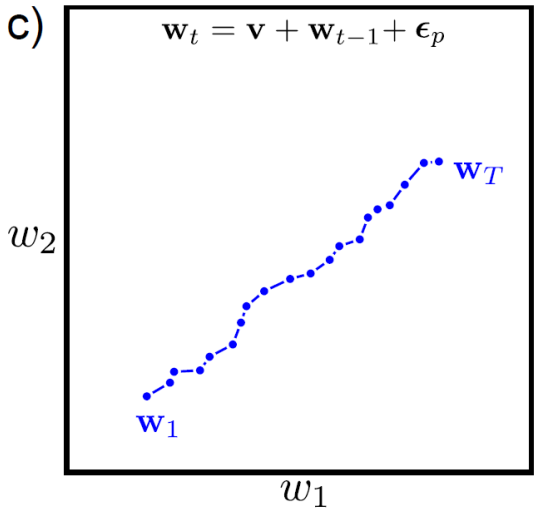


1 day

Temporal Models

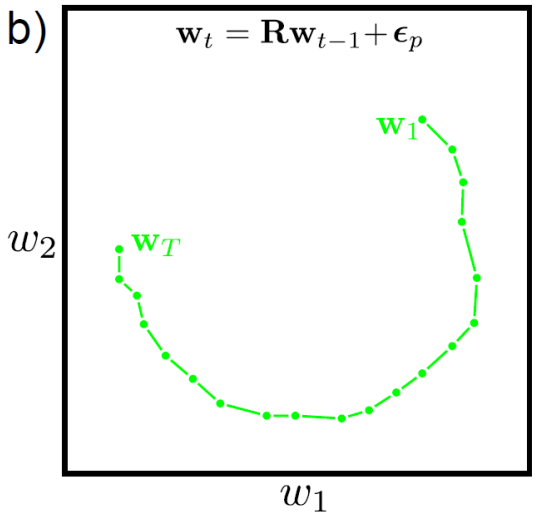
Brownian motion

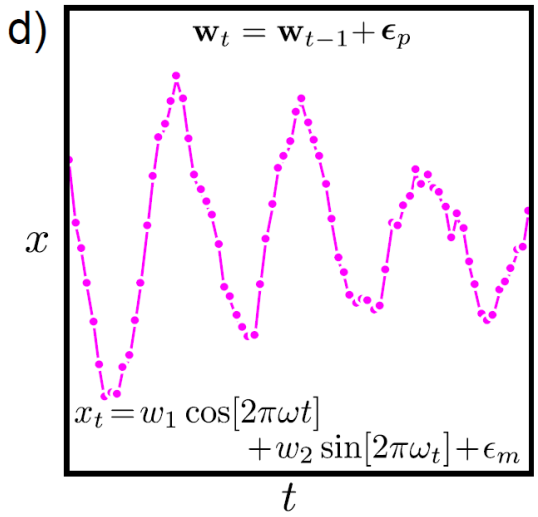




Constant velocity

Constant rotation





Brownian motion with oscillatory measurements

Source: S. Prince

Assumptions

- Only the immediate past matters (Markov)
  - the probability of the state at time t is conditionally independent of states at times 1...t-2 given the state at time t-1, i.e., $P(w_t|w_{t-1}, \ldots, w_1) = P(w_t|w_{t-1})$
- Measurements depend on only the current state
  - the likelihood of the measurements at time t is conditionally independent of all of the other measurements and the states at times 1...t-1 given the state at time t, $P(x_t|w_t, \ldots, w_1, x_{t-1}, \ldots, x_1) = P(x_t|w_t)$

Source: S. Prince

Recursive Estimation

Time t

$$Pr(w_t|x_t, x_{1\ldots t-1}) = \frac{Pr(x_t|w_t, x_{1\ldots t-1})Pr(w_t|x_{1\ldots t-1})}{Pr(x_t|x_{1\ldots t-1})}$$

$$= \frac{Pr(x_t|w_t)Pr(w_t|x_{1\ldots t-1})}{\int Pr(x_t|w_t, x_{1\ldots t-1})Pr(w_t|x_{1\ldots t-1})dw_t}$$

$$= \frac{Pr(x_t|w_t)Pr(w_t|x_{1\ldots t-1})}{\int Pr(x_t|w_t)Pr(w_t|x_{1\ldots t-1})dw_t}$$

Measurement model

Prediction from temporal model

Summary

Alternate between:

Temporal Evolution

Temporal model

$$Pr(\mathbf{w}_t|\mathbf{x}_{1\ldots t-1}) = \int Pr(\mathbf{w}_t|\mathbf{w}_{t-1})Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\ldots t-1})\, d\mathbf{w}_{t-1}$$

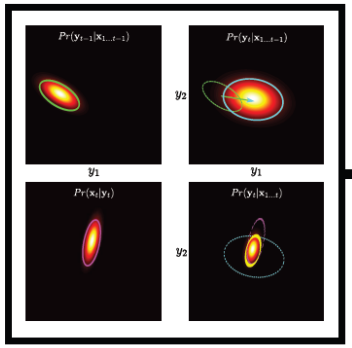
Measurement Update

Measurement model

$$Pr(\mathbf{w}_t|\mathbf{x}_{1\ldots t}) = \frac{Pr(\mathbf{x}_t|\mathbf{w}_t)Pr(\mathbf{w}_t|\mathbf{x}_{1\ldots t-1})}{\int Pr(\mathbf{x}_t|\mathbf{w}_t)Pr(\mathbf{w}_t|\mathbf{x}_{1\ldots t-1})\, d\mathbf{w}_t}$$

Source: S. Prince

Roadmap

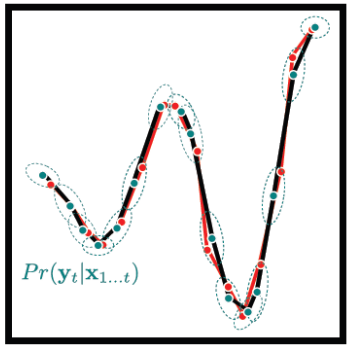


Kalman filter

Problem 1

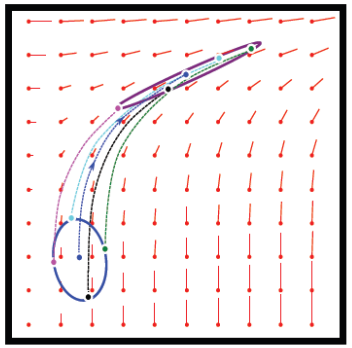
Linear

Linearize with Taylor expansion



Extended Kalman filter

Approximate normal with particles

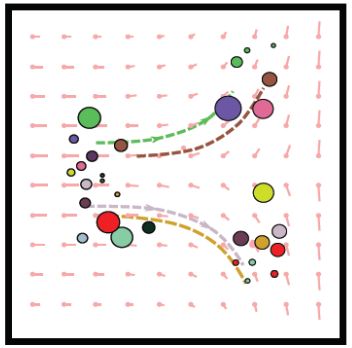


Unscented Kalman filter

Problem 2

Unimodal

Represent distribution with particles



Particle filter

Source: S. Prince

Kalman Filter Definition

Time evolution equation

$$Pr(\mathbf{w}_t|\mathbf{w}_{t-1}) = \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_p + \boldsymbol{\Psi}\mathbf{w}_{t-1}, \boldsymbol{\Sigma}_p]$$

State transition matrix

Additive Gaussian noise

Measurement equation

$$Pr(\mathbf{x}_t|\mathbf{w}_t) = \text{Norm}_{\mathbf{x}_t}[\boldsymbol{\mu}_m + \boldsymbol{\Phi}\mathbf{w}_t, \boldsymbol{\Sigma}_m]$$

Relates state and measurement

Additive Gaussian noise

Source: S. Prince

Kalman Filter Summary

Time evolution equation

$$Pr(\mathbf{w}_t|\mathbf{w}_{t-1}) = \text{Norm}_{\mathbf{w}_t}[\boldsymbol{\mu}_p + \boldsymbol{\Psi}\mathbf{w}_{t-1}, \boldsymbol{\Sigma}_p]$$

Measurement equation

$$Pr(\mathbf{x}_t|\mathbf{w}_t) = \text{Norm}_{\mathbf{x}_t}[\boldsymbol{\mu}_m + \boldsymbol{\Phi}\mathbf{w}_t, \boldsymbol{\Sigma}_m]$$

Inference

$$\begin{aligned}
\text{State Prediction:} \quad & \boldsymbol{\mu}_+ = \boldsymbol{\mu}_p + \boldsymbol{\Psi}\boldsymbol{\mu}_{t-1} \\
\text{Covariance Prediction:} \quad & \boldsymbol{\Sigma}_+ = \boldsymbol{\Sigma}_p + \boldsymbol{\Psi}\boldsymbol{\Sigma}_{t-1}\boldsymbol{\Psi}^T \\
\text{State Update:} \quad & \boldsymbol{\mu}_t = \boldsymbol{\mu}_+ + \mathbf{K}(\mathbf{x}_t - \boldsymbol{\mu}_m - \boldsymbol{\Phi}\boldsymbol{\mu}_+) \\
\text{Covariance Update:} \quad & \boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}\boldsymbol{\Phi})\boldsymbol{\Sigma}_+,
\end{aligned}$$

$$\mathbf{K} = \boldsymbol{\Sigma}_+\boldsymbol{\Phi}^T(\boldsymbol{\Sigma}_m + \boldsymbol{\Phi}\boldsymbol{\Sigma}_+\boldsymbol{\Phi}^T)^{-1}$$

Source: S. Prince

Fixed lag smoother

State evolution equation

$$\begin{bmatrix} \mathbf{w}_t \\ \mathbf{w}_t^{[1]} \\ \mathbf{w}_t^{[2]} \\ \vdots \\ \mathbf{w}_t^{[\tau]} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Psi} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w}_{t-1} \\ \mathbf{w}_{t-1}^{[1]} \\ \mathbf{w}_{t-1}^{[2]} \\ \vdots \\ \mathbf{w}_{t-1}^{[\tau]} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_p \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0}, \end{bmatrix}$$

Estimate delayed by τ

Measurement equation

$$\mathbf{x}_t = \begin{bmatrix} \boldsymbol{\Phi} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w}_t \\ \mathbf{w}_t^{[1]} \\ \mathbf{w}_t^{[2]} \\ \vdots \\ \mathbf{w}_t^{[\tau]}, \end{bmatrix} + \boldsymbol{\epsilon}_m$$

Source: S. Prince

Fixed interval smoothing

Having μ_t, Σ_t, $\mu_{+|t}$, and $\Sigma_{+|t}$ from forward pass where $Pr(w_{t+1}|x_{1...t}) = Norm(\mu_{+|t}, \Sigma_{+|t})$

Backward set of recursions

$$\begin{aligned}\boldsymbol{\mu}_{t|T} &= \boldsymbol{\mu}_t + \mathbf{C}_t(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{+|t}) \\ \boldsymbol{\Sigma}_{t|T} &= \boldsymbol{\Sigma}_t + \mathbf{C}_t(\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{+|t})\mathbf{C}_t^T\end{aligned}$$

where

$$\mathbf{C}_t = \boldsymbol{\Sigma}_t \ \boldsymbol{\Psi}^T \boldsymbol{\Sigma}_{+|t}^{-1}$$

Source: S. Prince

Learning parameters

Parameters of equations:

$$w_t = \Psi w_{t-1} + \epsilon_p$$

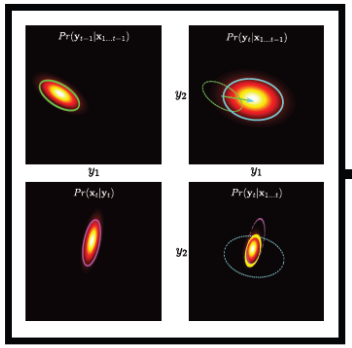
$$x_t = \Phi w_t + \epsilon_m$$

$$\theta = \{\Psi, \Sigma_p, \Phi, \Sigma_m, \mu_0, \Sigma_0\}$$

Expectation-Maximation

- E step:
  - Fixed interval smoothing to get (w_t)
- M step:
  - Maximize log-likelihood to get θ

Roadmap

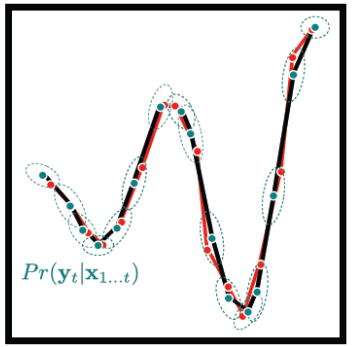


Kalman filter

Problem 1

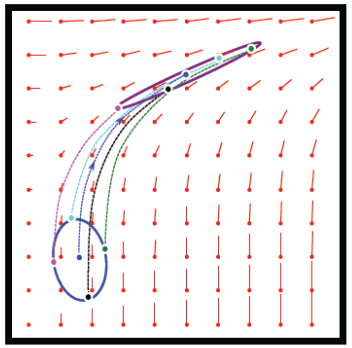
Linear

Linearize with Taylor expansion



Extended Kalman filter

Approximate normal with particles

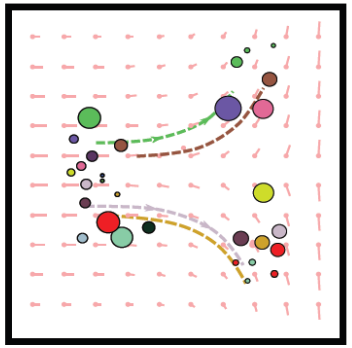


Unscented Kalman filter

Problem 2

Unimodal

Represent distribution with particles



Particle filter

Source: S. Prince

Extended Kalman Filter

Allows non-linear measurement and temporal equations

$$\mathbf{w}_t = \mathbf{f}[\mathbf{w}_{t-1}, \boldsymbol{\epsilon}_p]$$

$$\mathbf{x}_t = \mathbf{g}[\mathbf{w}_t, \boldsymbol{\epsilon}_m]$$

Key idea: take Taylor expansion and treat as locally linear

Source: S. Prince

Unscented Kalman Filter

Approximate with particles where D_w is dimensionality of state:

$$\begin{aligned} Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\ldots t-1}) &= \mathrm{Norm}_{\mathbf{w}_{t-1}}[\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}] \\ &\approx \sum_{j=0}^{2D_\mathbf{w}} a_j \delta[\mathbf{w}_{t-1} - \hat{\mathbf{w}}^{[j]}], \end{aligned}$$

weights — particles or sigma points

Choose so that

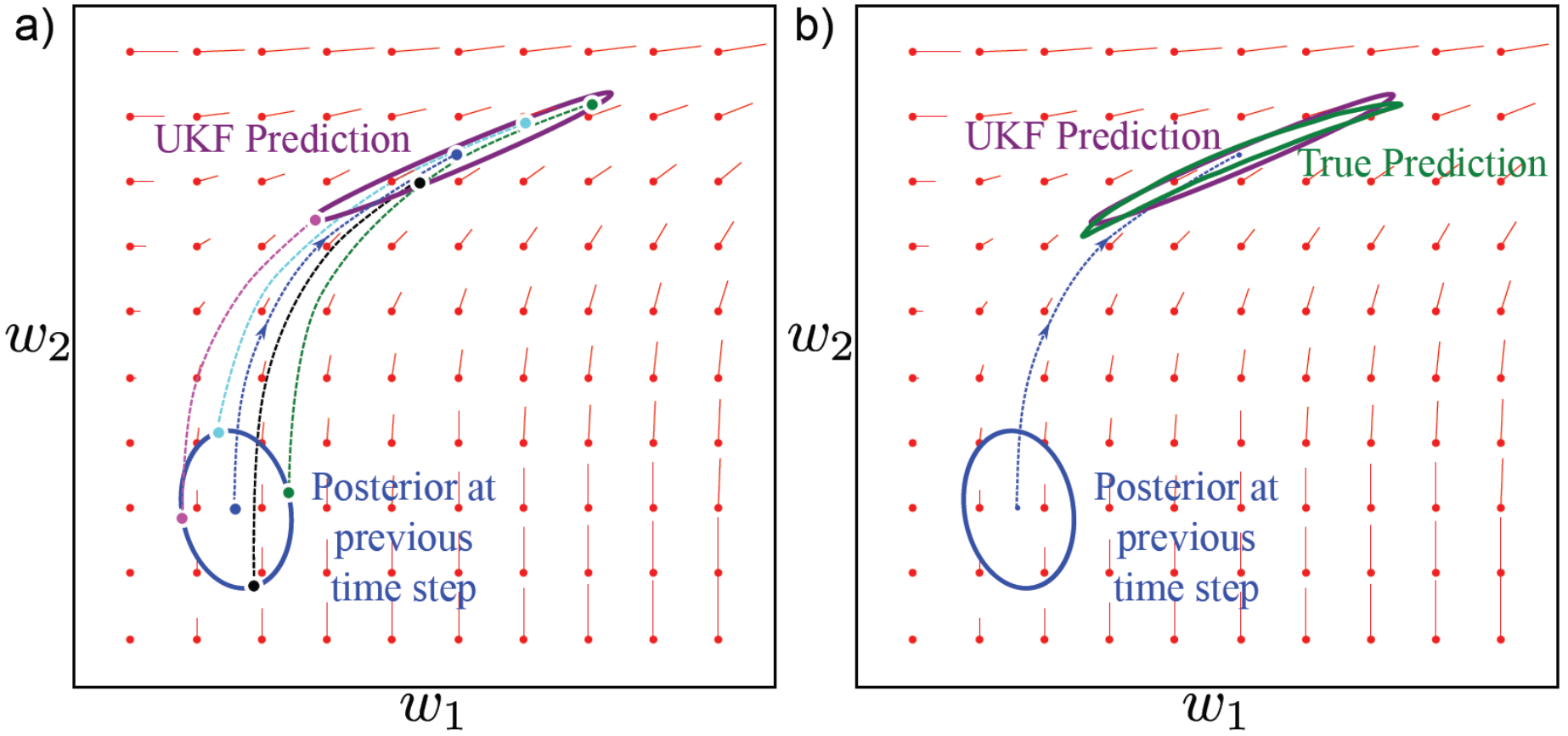
$$\sum_{j=0}^{2D_w} a_j = 1 \qquad \delta(x) = \begin{cases} 1 & x = 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\boldsymbol{\mu}_{t-1} = \sum_{j=0}^{2D_\mathbf{w}} a_j \hat{\mathbf{w}}^{[j]}$$

$$\boldsymbol{\Sigma}_{t-1} = \sum_{j=0}^{2D_\mathbf{w}} a_j (\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_{t-1})(\hat{\mathbf{w}}^{[j]} - \boldsymbol{\mu}_{t-1})^T$$

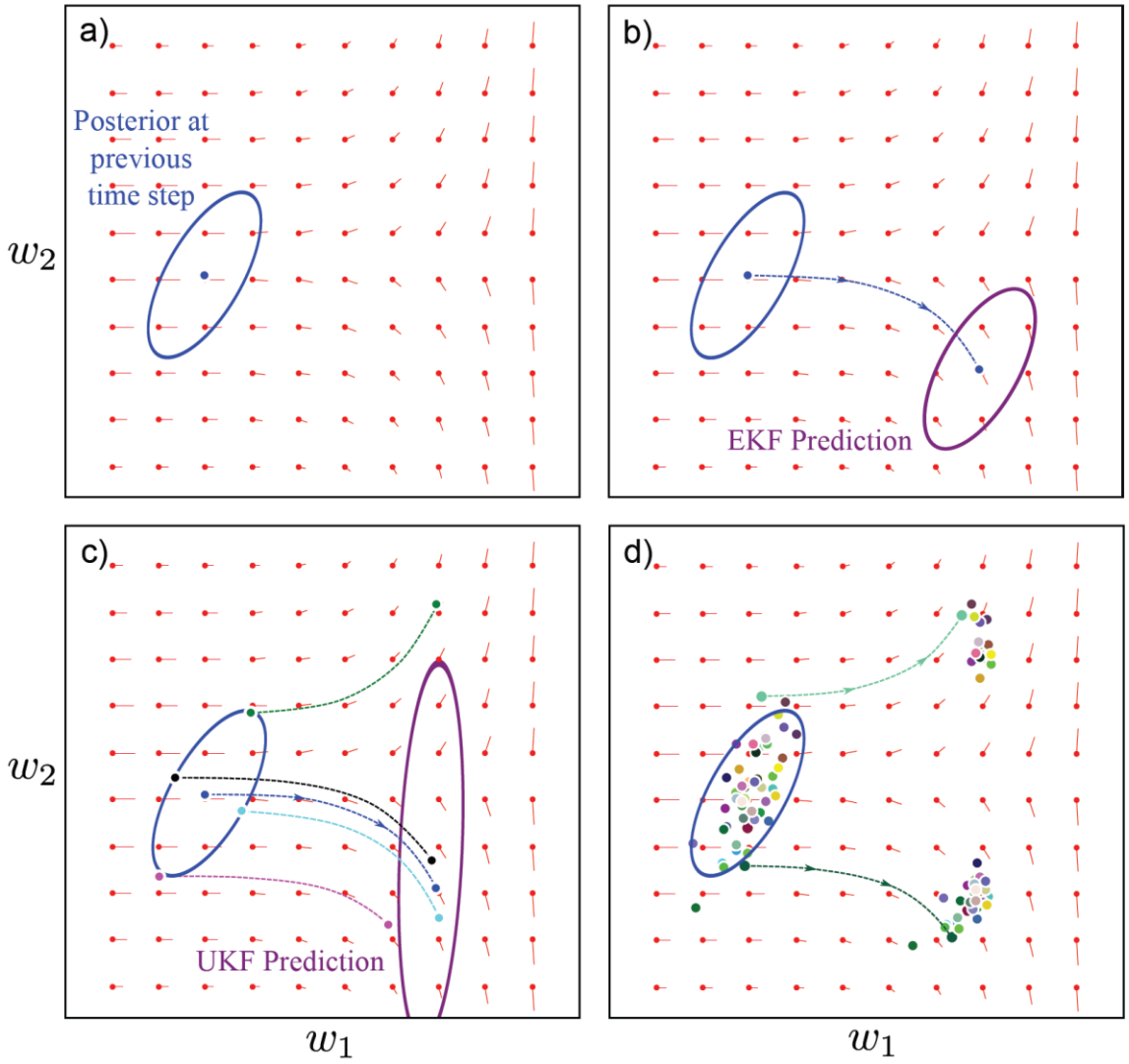
Source: S. Prince

Unscented Kalman Filter



Source: S. Prince

Problems with UKF



Source: S. Prince

Particle filters

Key idea:

- Represent probability distribution as a set of weighted particles

$$Pr(\mathbf{w}_{t-1}|\mathbf{x}_{1\ldots t-1}) = \sum_{j=1}^{J} a_j \delta[\mathbf{w}_{t-1} - \hat{\mathbf{w}}_{t-1}^{[j]}]$$

Advantages and disadvantages:

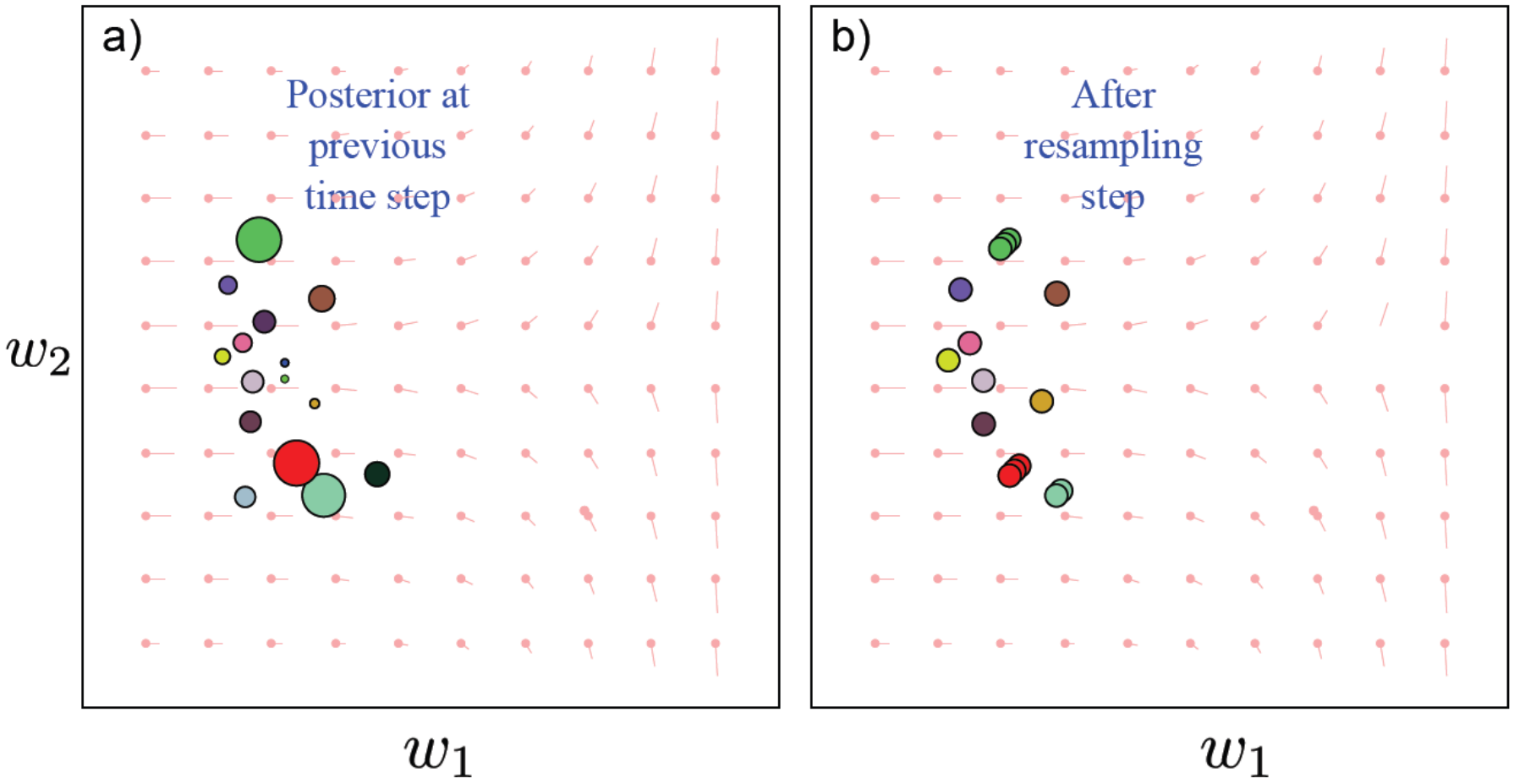
+ Can represent non-Gaussian multimodal densities
+ No need for data association
- Expensive

N. Gordon et al. **Novel approach to nonlinear/non-Gaussian Bayesian state estimation.** 1993
M. Isard and A. Blake. **Contour tracking by stochastic propagation of conditional density.** ECCV 1996

Source: S. Prince

Particle filter

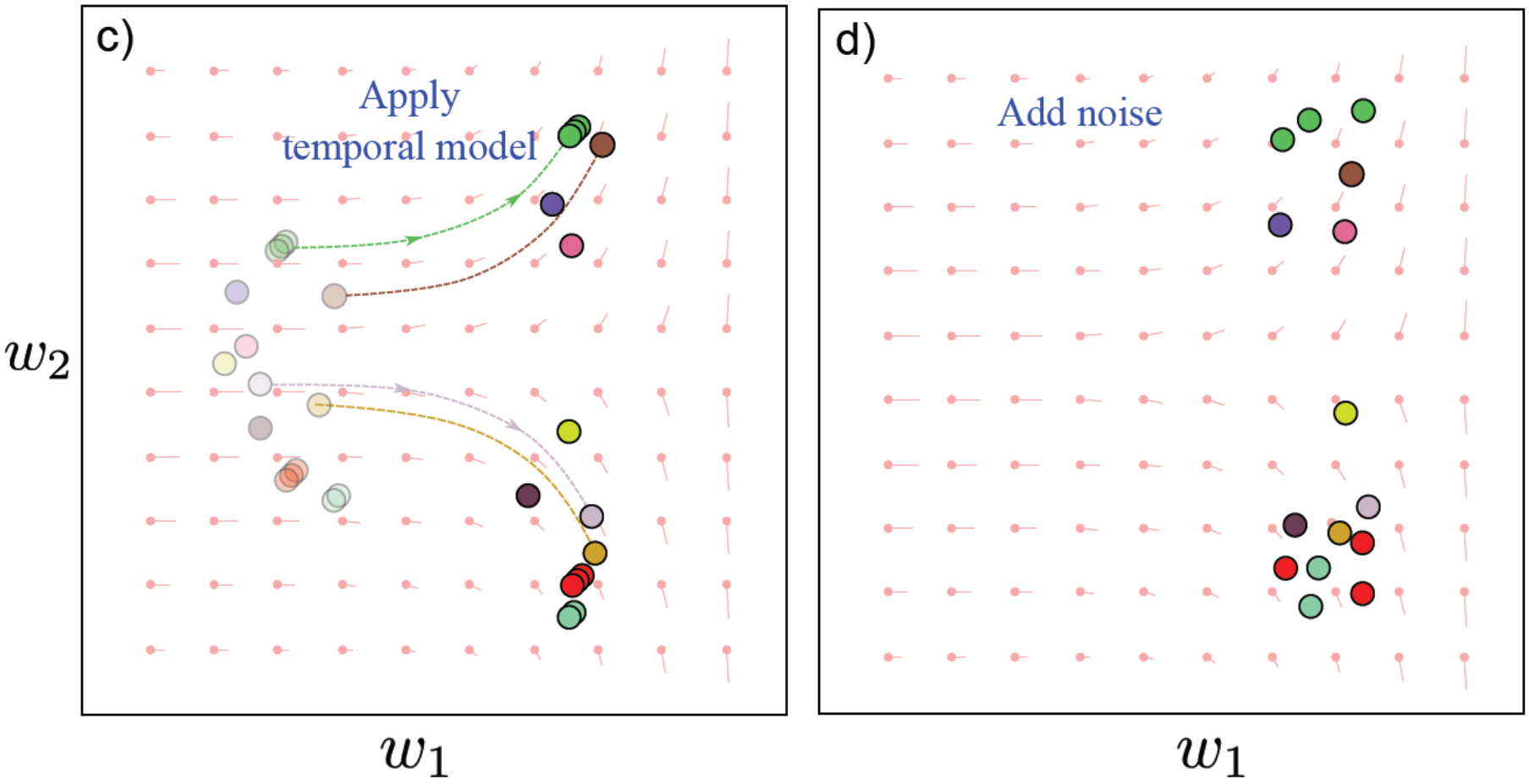
Stage 1: Resample from weighted particles according to their weight to get unweighted particles



Source: S. Prince

Particle filter

Stage 2: Pass unweighted samples through temporal model and add noise

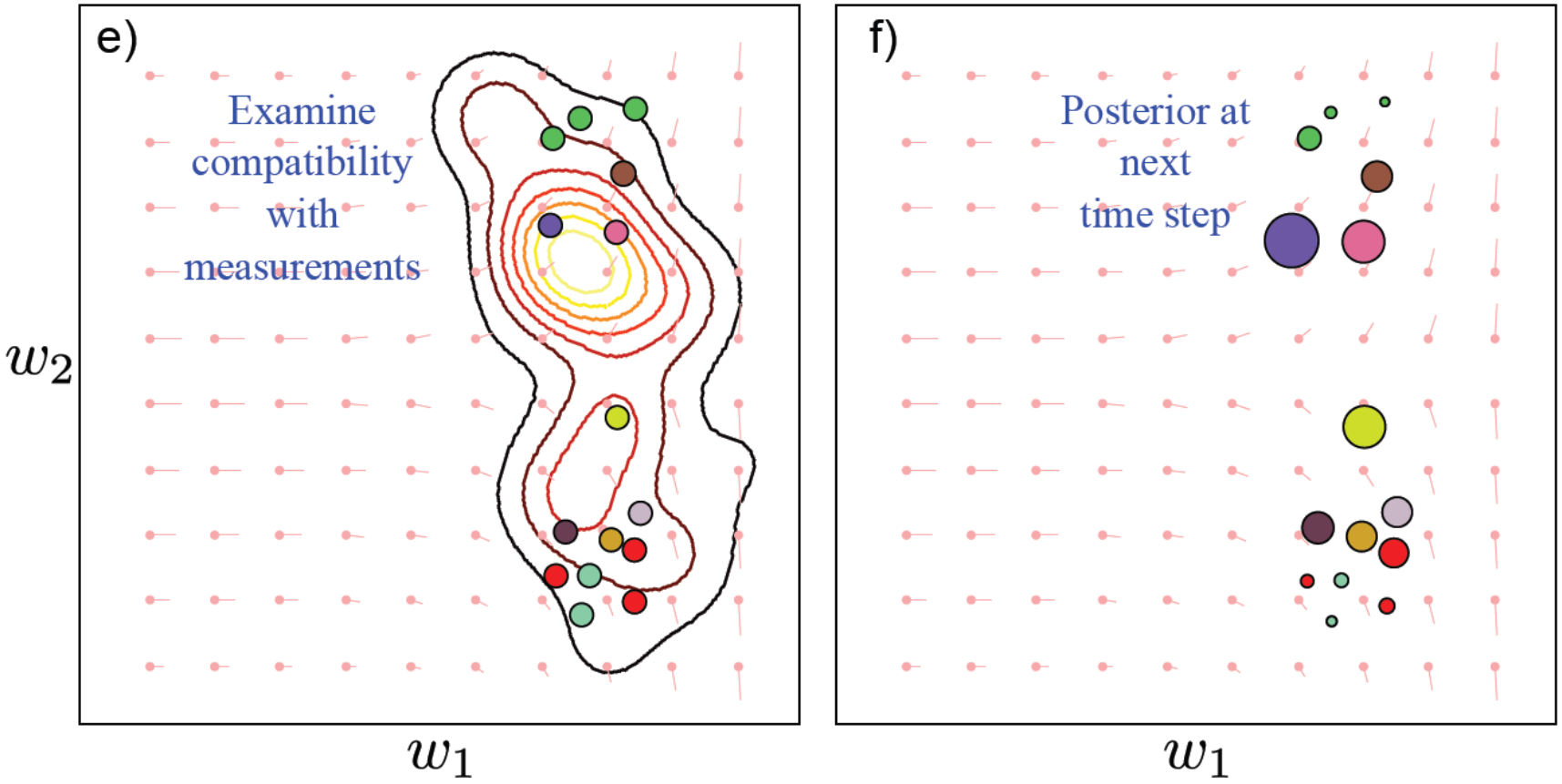


Source: S. Prince

Particle filter

UNIVERSITÄT BONN

Stage 3: Weight samples by measurement density



Source: S. Prince

Generic particle filter

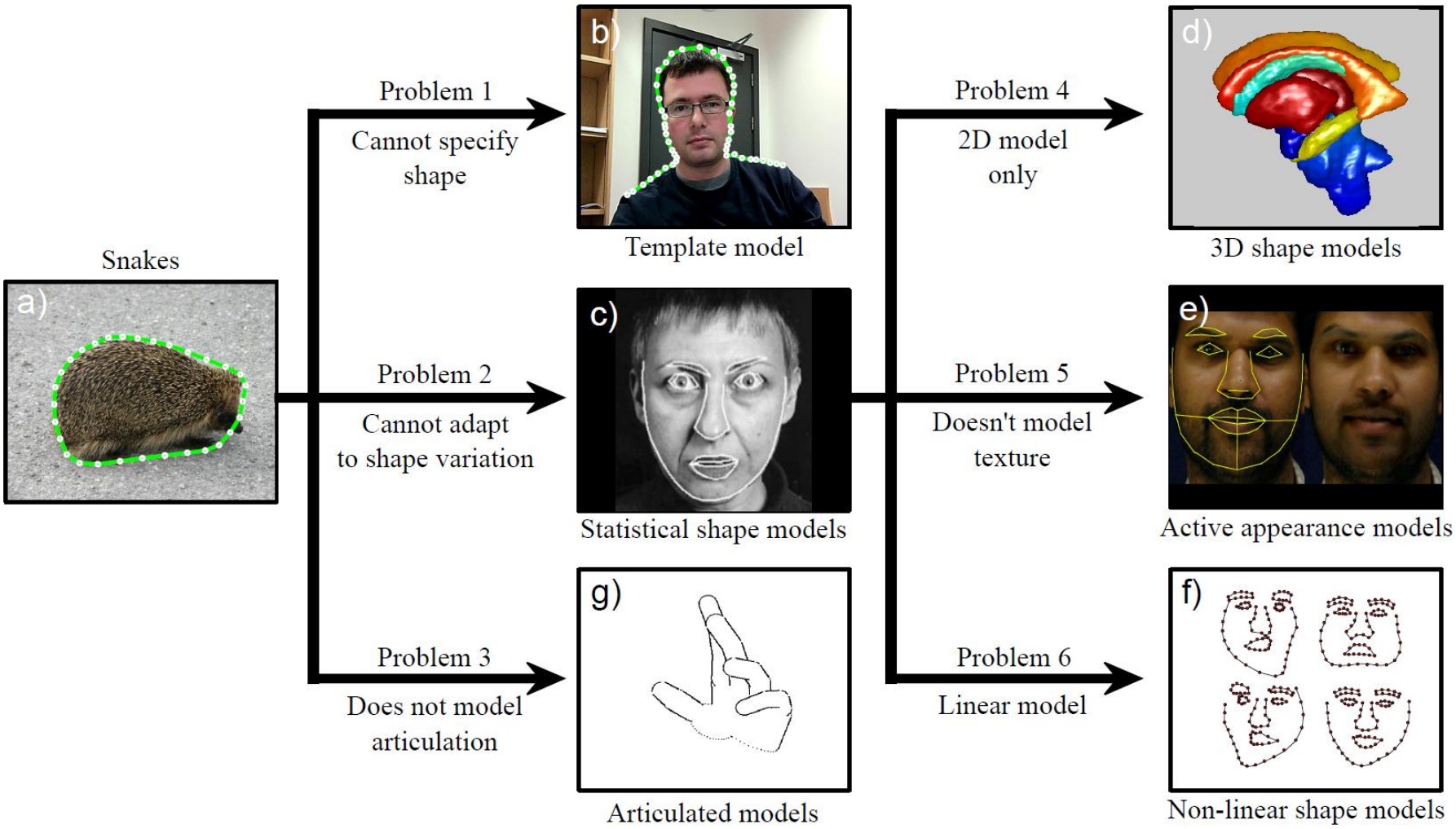
UNIVERSITÄT BONN

1. Initialisation
   - $t \leftarrow 0$
   - For $i = 1, \ldots, n$, sample $x_{0,0}^{(i)}$ from η_0
2. Prediction

   Temporal model
   - For $i = 1, \ldots, n$, sample $\bar{x}_{t+1,0}^{(i)}$ from $K_t(x_{t,0}^{(i)}, \cdot)$
3. Updating

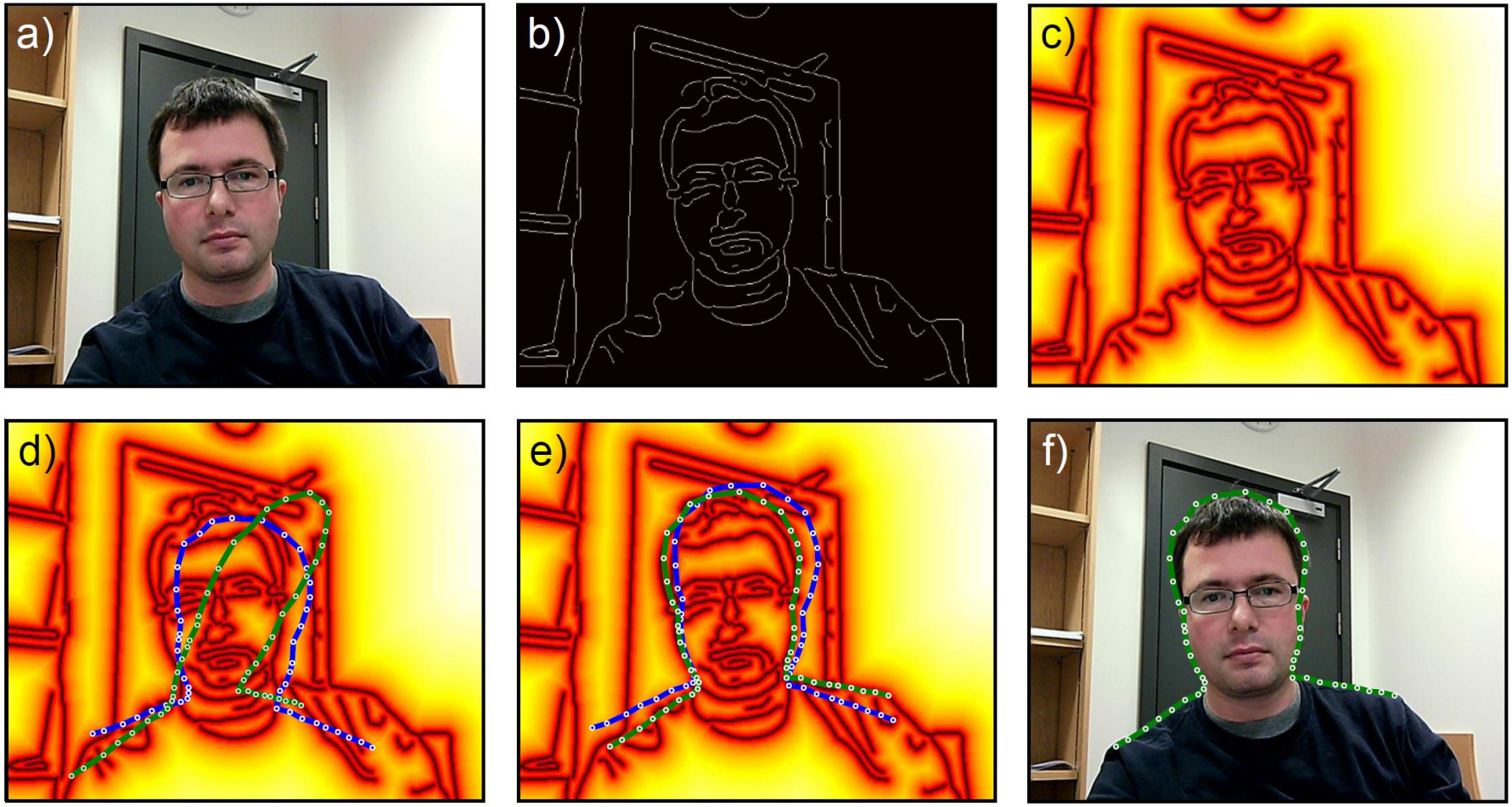
   Likelihood
   - For $i = 1, \ldots, n$, set $\pi_{t+1,0}^{(i)} \leftarrow g_{t+1}(y_{t+1} - h_{t+1}(\bar{x}_{t+1,0}^{(i)}))$
   - For $i = 1, \ldots, n$, set $\pi_{t+1,0}^{(i)} \leftarrow \frac{\pi_{t+1,0}^{(i)}}{\sum_{j=1}^{n} \pi_{t+1,0}^{(j)}}$
4. Resampling
   - For $i = 1, \ldots, n$, set $x_{t+1,0}^{(i)} \leftarrow \bar{x}_{t+1,0}^{(j)}$ with probability $\pi_{t+1,0}^{(j)}$
   - $t \leftarrow t + 1$ and go to step 2

Relationships between models



Source: S. Prince

Shape template model



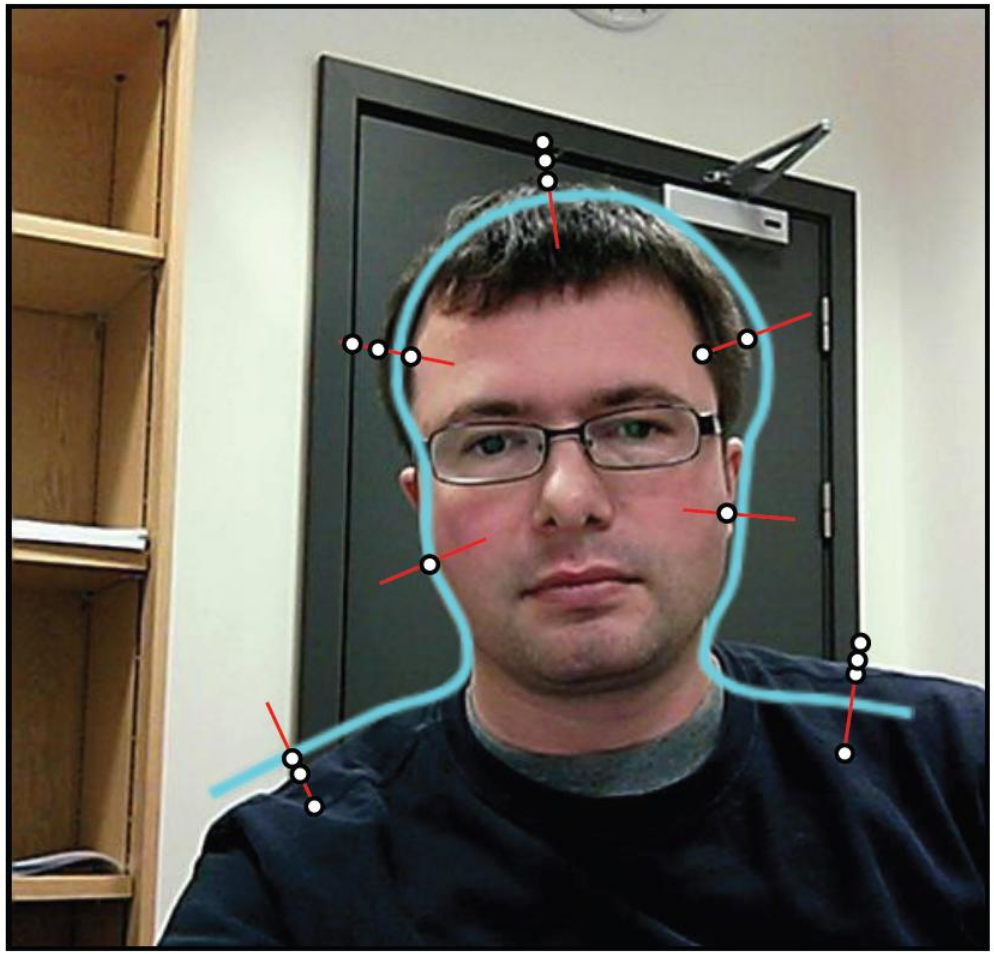
Local optimum

Global optimum

Blue initialization, green estimate

Source: S. Prince

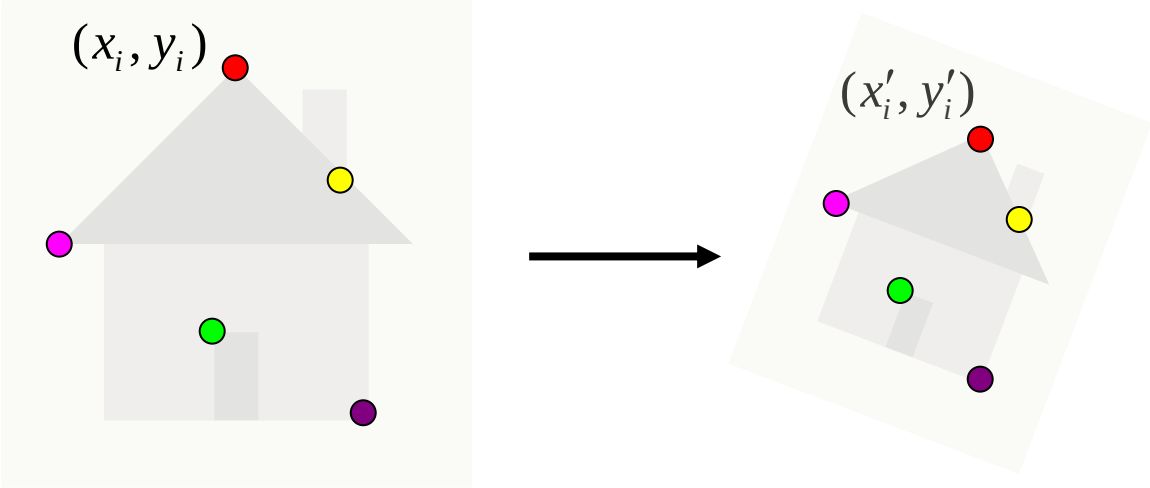
Iterative closest points



- Find nearest edge point to each landmark point
- Compute transformation in closed form
- Repeat

Source: S. Prince

Fitting an affine transformation



$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & \cdots & & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & \cdots & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x'_i \\ y'_i \\ \cdots \end{bmatrix}$$

Source: K. Grauman

Least squares problem

A *m x n* matrix of rank *n*:

$$Ax = b$$

Solution with pseudo-inverse:

$$A^T(Ax - b) = 0$$

$$x = (A^T A)^{-1} A^T b$$

Pseudo-inverse $\mathbf{A}^+$

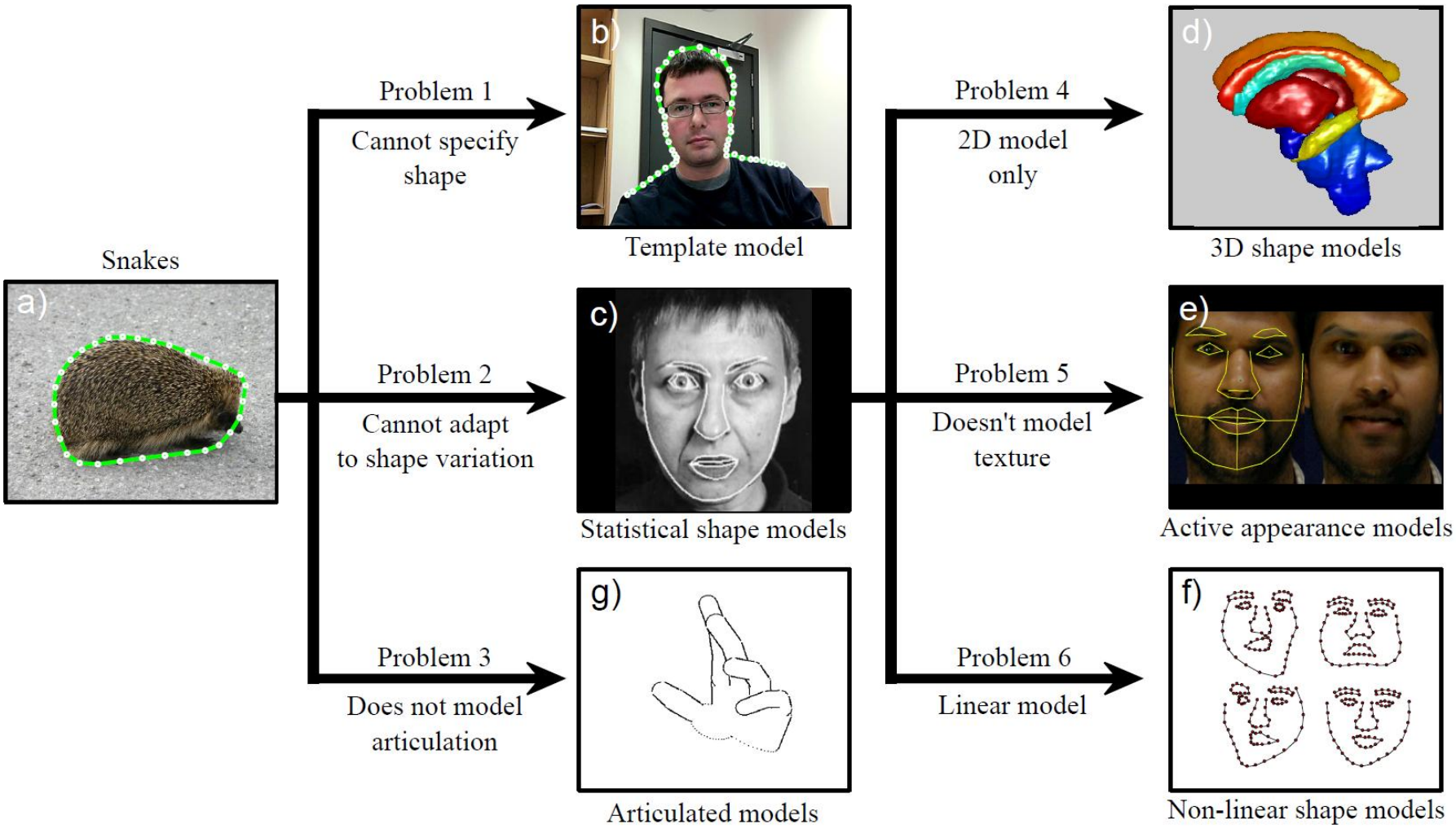
Solution with SVD:

- SVD: $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$; $\mathbf{A}^+ = \mathbf{V}\mathbf{D}^{-1}\mathbf{U}^T$
- $\mathbf{b}' = \mathbf{U}^T\mathbf{b}$
- $y_i = b'_i / D_{ii}$
- $\mathbf{x} = \mathbf{V}\mathbf{y}$

There are different ways to solve the problem. The optimal method depends on size and sparseness of **A**

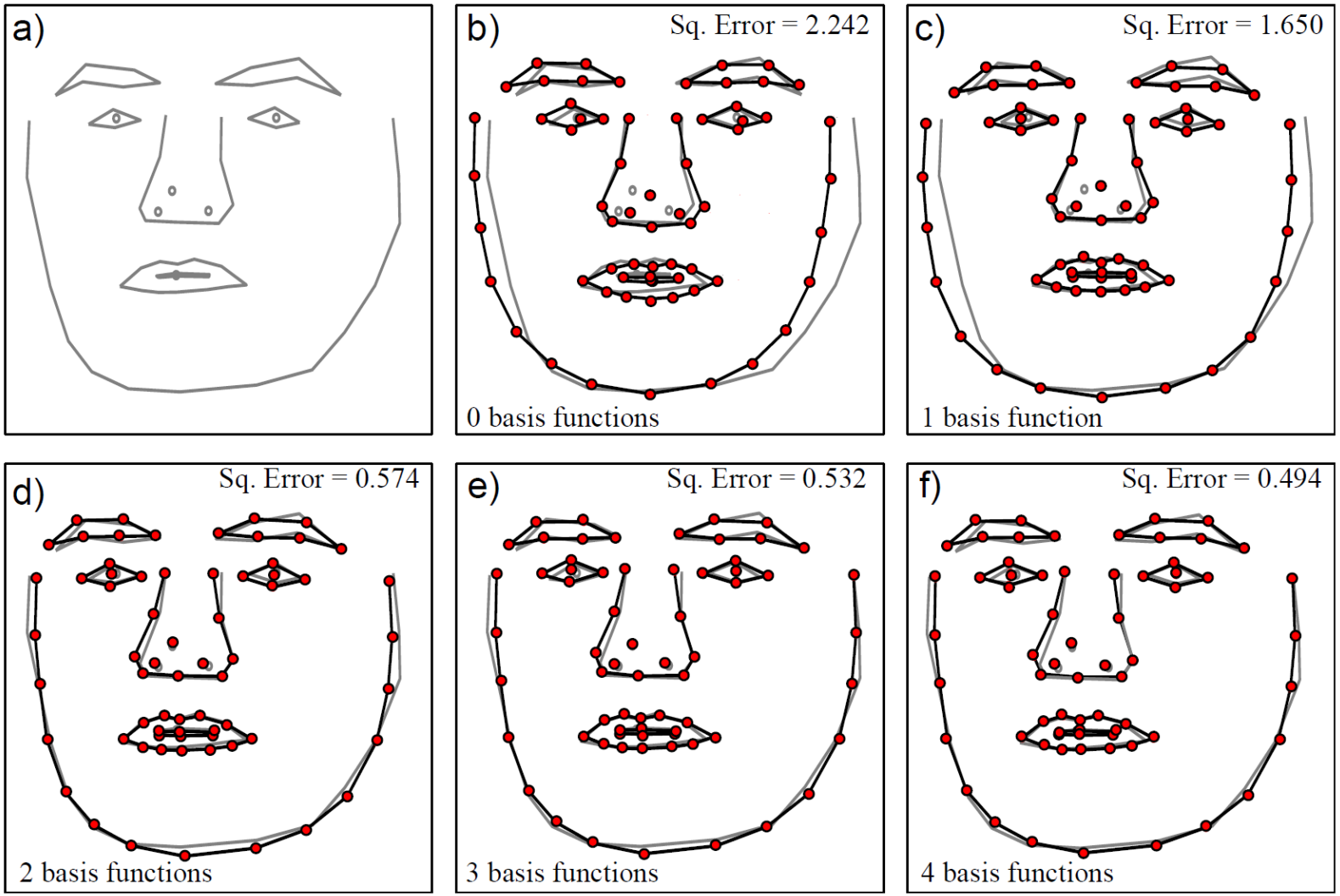
Specific C++ linear algebra libraries exist!

Relationships between models



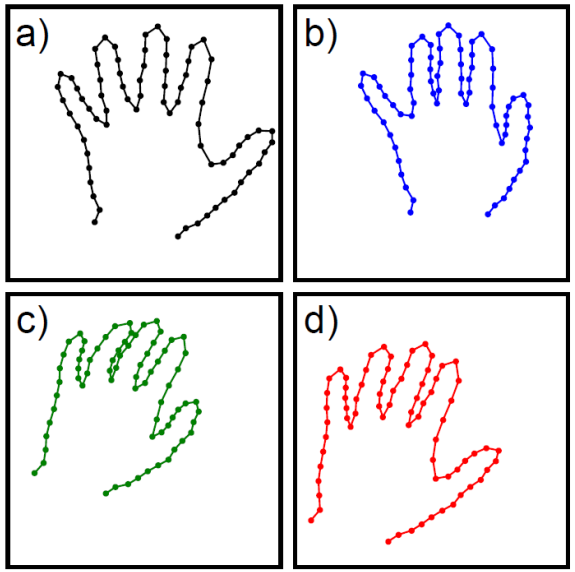
Source: S. Prince

Subspace shape model

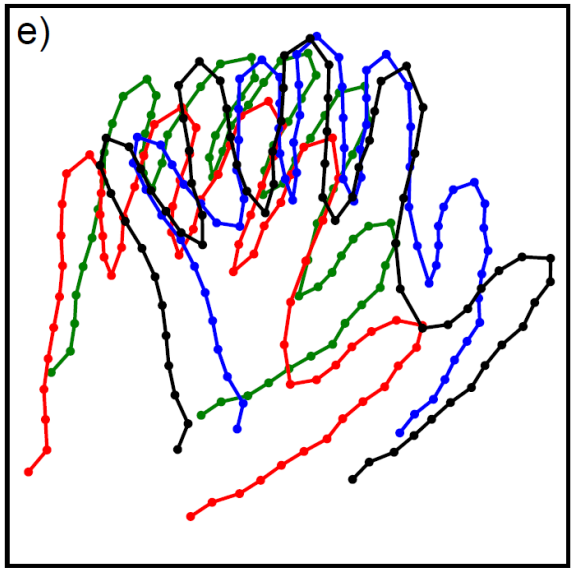


Source: S. Prince

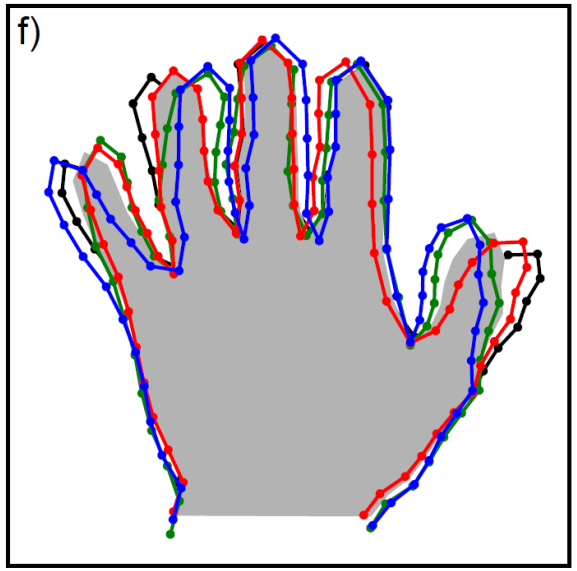
Generalized Procrustes analysis



Training data



Before alignment

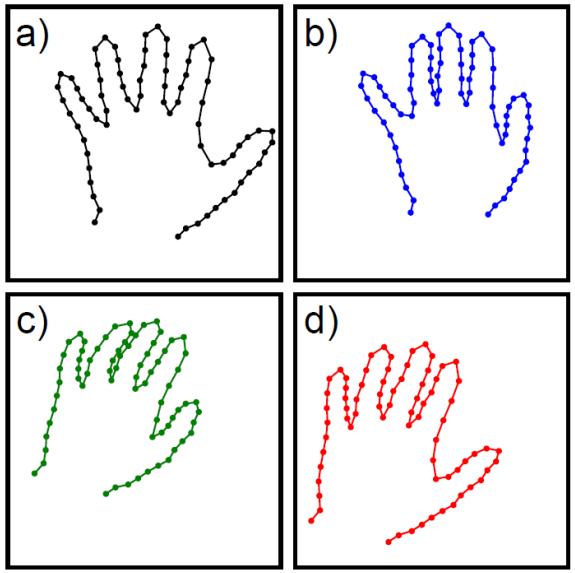


After alignment

Source: S. Prince

Generalized Procrustes analysis

- Given *I* shapes with *N* landmarks
- Correspondences between shapes are known (landmarks)
- Take one shape as mean shape
- Iterate:
  - Solve for affine transformation ψ_i for each shape i to the mean $A_i x = b_i$
  - Apply transformation ψ_i to each shape i
  - Compute the mean μ_n of each landmark (mean shape)



Subspace shape model

- Generate data from model:

$$\mathbf{w}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \epsilon_i$$

  - $\boldsymbol{\mu}$ is the mean shape
  - the matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_K]$ contains K basis functions in it columns
  - ϵ_i is normal noise with covariance $\sigma^2\mathbf{I}$
- Can alternatively write

$$\mathbf{w}_i = \boldsymbol{\mu} + \sum_{k=1}^{K} \boldsymbol{\phi}_k h_{ik} + \epsilon_i$$

Source: S. Prince

Approximating with subspace

Subspace model

$$\mathbf{w}_i = \boldsymbol{\mu} + \sum_{k=1}^{K} \boldsymbol{\phi}_k h_{ik} + \epsilon_i$$

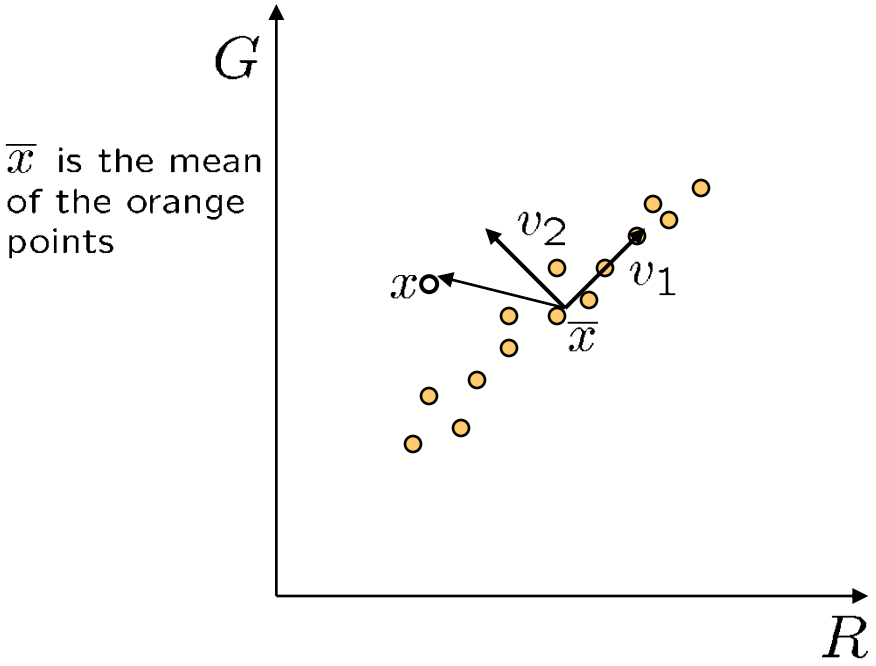
Can approximate an vector w with a weighted sum of the basis functions

$$\mathbf{w}_i \approx \boldsymbol{\mu} + \sum_{k=1}^{K} \boldsymbol{\phi}_k h_{ik}$$

Surprising how well this works even with a small number of basis functions

Source: S. Prince

Linear subspaces



Consider the variation along direction **v** among all of the orange points:

$$var(\mathbf{v}) = \sum_{\text{orange point } \mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2$$

What unit vector **v** minimizes *var*?

$$\mathbf{v}_2 = min_{\mathbf{v}} \{var(\mathbf{v})\}$$

What unit vector **v** maximizes *var*?

$$\mathbf{v}_1 = max_{\mathbf{v}} \{var(\mathbf{v})\}$$

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\|^2 \\ &= \sum_{\mathbf{x}} \mathbf{v}^{\mathrm{T}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \left[\sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \right] \mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \mathbf{A} \mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x} - \overline{\mathbf{x}})(\mathbf{x} - \overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

Solution: $\mathbf{v}_1$ is eigenvector of **A** with *largest* eigenvalue
$\mathbf{v}_2$ is eigenvector of **A** with *smallest* eigenvalue

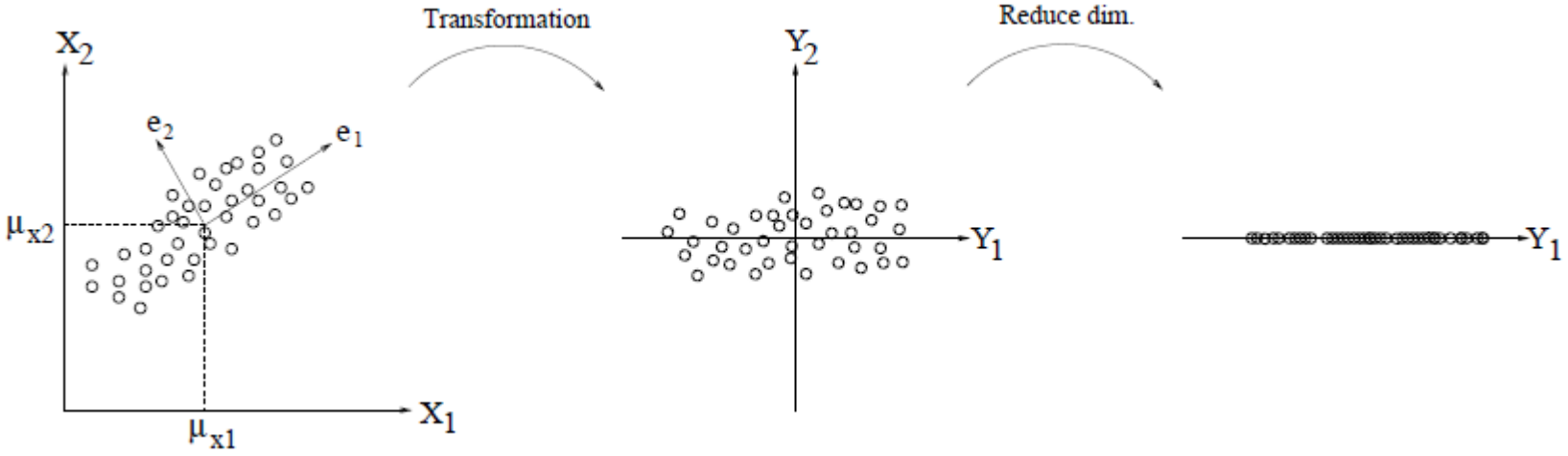
Principal component analysis (PCA)

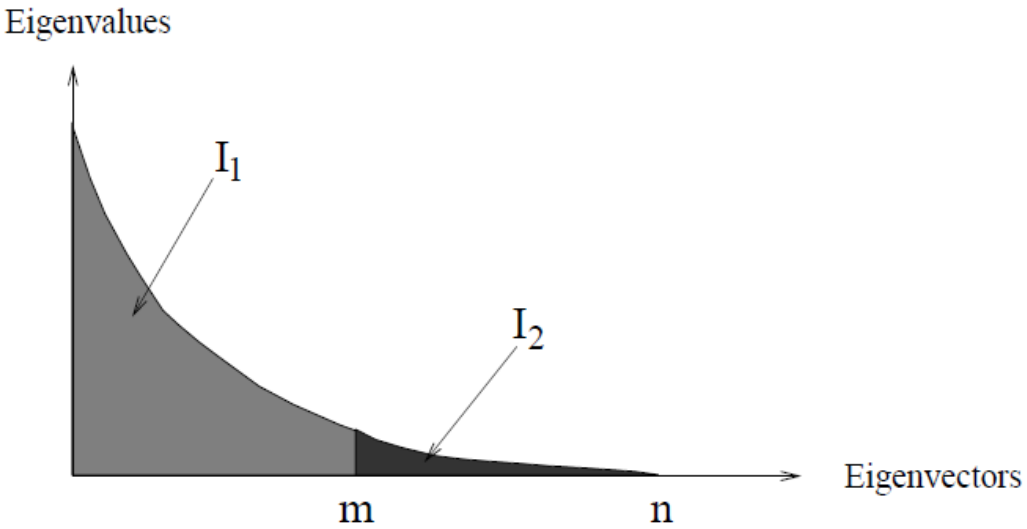
UNIVERSITÄT BONN

- Given data points w_i
- Compute mean μ
- $W = (w_1 - \mu, \ldots, w_n - \mu)$
- Eigenvalue decomposition:
  $WW^T = U\Sigma V^T V\Sigma U^T = UL^2U^T$ (u_i basis vectors)
- SVD: $W = U\Sigma V^T$

Principal component analysis (PCA)

- Dimensionality reduction





$$\frac{\sum_{i=1}^{i=K} L_{ii}^2}{\sum_{i=1}^{i=D} L_{ii}^2} > 1 - \epsilon$$

Source: T. Moeslund

Learning PPCA

Learn parameters $\boldsymbol{\mu}, \boldsymbol{\Phi}$ and σ^2 from data $\{\mathbf{w}_i\}_{i=1}^{I}$ where $\mathbf{w}_i = [\mathbf{w}_{i1}^T, \mathbf{w}_{i2}^T \ldots, \mathbf{w}_{iN}^T]^T$.

Learn mean: $\boldsymbol{\mu} = \dfrac{\sum_{i=1}^{I} \mathbf{w}_i}{I}$

Then set $\mathbf{W} = [\mathbf{w}_1 - \boldsymbol{\mu}, \mathbf{w}_2 - \boldsymbol{\mu}, \ldots, \mathbf{w}_I - \boldsymbol{\mu}]$ and compute eigen-decomposition

$$\mathbf{W}\mathbf{W}^T = \mathbf{U}\mathbf{L}^2\mathbf{U}^T$$

Choose parameters

$$\hat{\sigma}^2 = \frac{1}{D-K} \sum_{j=K+1}^{D} L_{jj}^2$$

$$\hat{\boldsymbol{\Phi}} = \mathbf{U}_K(\mathbf{L}_K^2 - \hat{\sigma}^2\mathbf{I})^{1/2}$$

Source: S. Prince

Simple version with PCA

UNIVERSITÄT BONN

1. Initialise the shape parameters, $\mathbf{b}$, to zero (the mean shape).
2. Generate the model point positions using $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}\mathbf{b}$
3. Find the pose parameters (X_t, Y_t, s, θ) which best align the model points $\mathbf{x}$ to the current found points $\mathbf{Y}$

$$T_{X_t,Y_t,s,\theta}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X_t \\ Y_t \end{pmatrix} + \begin{pmatrix} s\cos\theta & -s\sin\theta \\ s\sin\theta & s\cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

4. Project $\mathbf{Y}$ into the model co-ordinate frame by inverting the transformation T:

$$\mathbf{y} = T^{-1}_{X_t,Y_t,s,\theta}(\mathbf{Y})$$

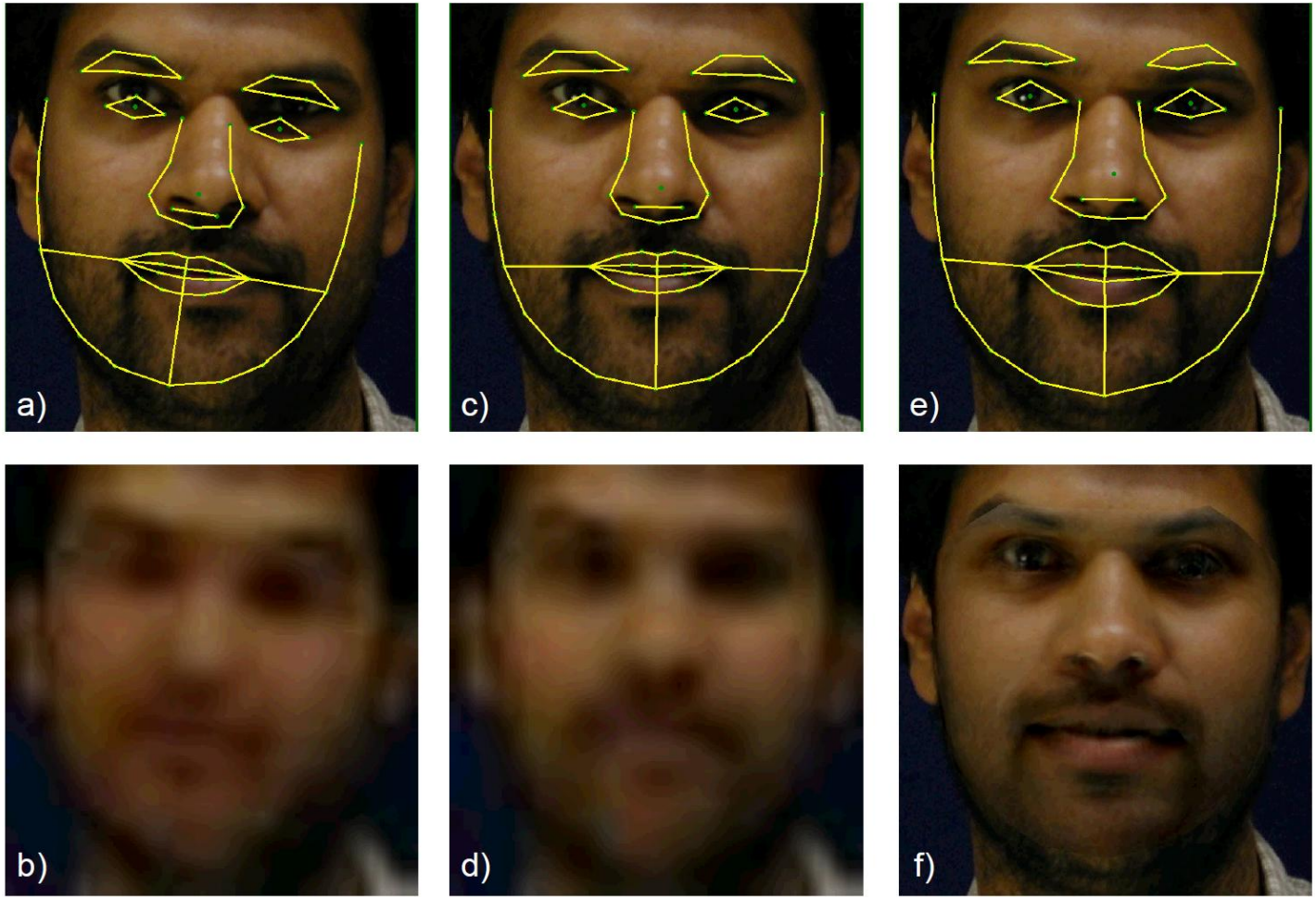
5. Project $\mathbf{y}$ into the tangent plane to $\bar{\mathbf{x}}$ by scaling: $\mathbf{y}' = \mathbf{y}/(\mathbf{y}.\bar{\mathbf{x}})$. (optional)
6. Update the model parameters to match to $\mathbf{y}'$

$$\mathbf{b} = \mathbf{P}^T(\mathbf{y}' - \bar{\mathbf{x}})$$

7. If not converged, return to step 2.

T. Cootes. An Introduction to Active Shape Models.

Statistical models for shape and appearance



Source: T. Cootes

Motivation: mosaics

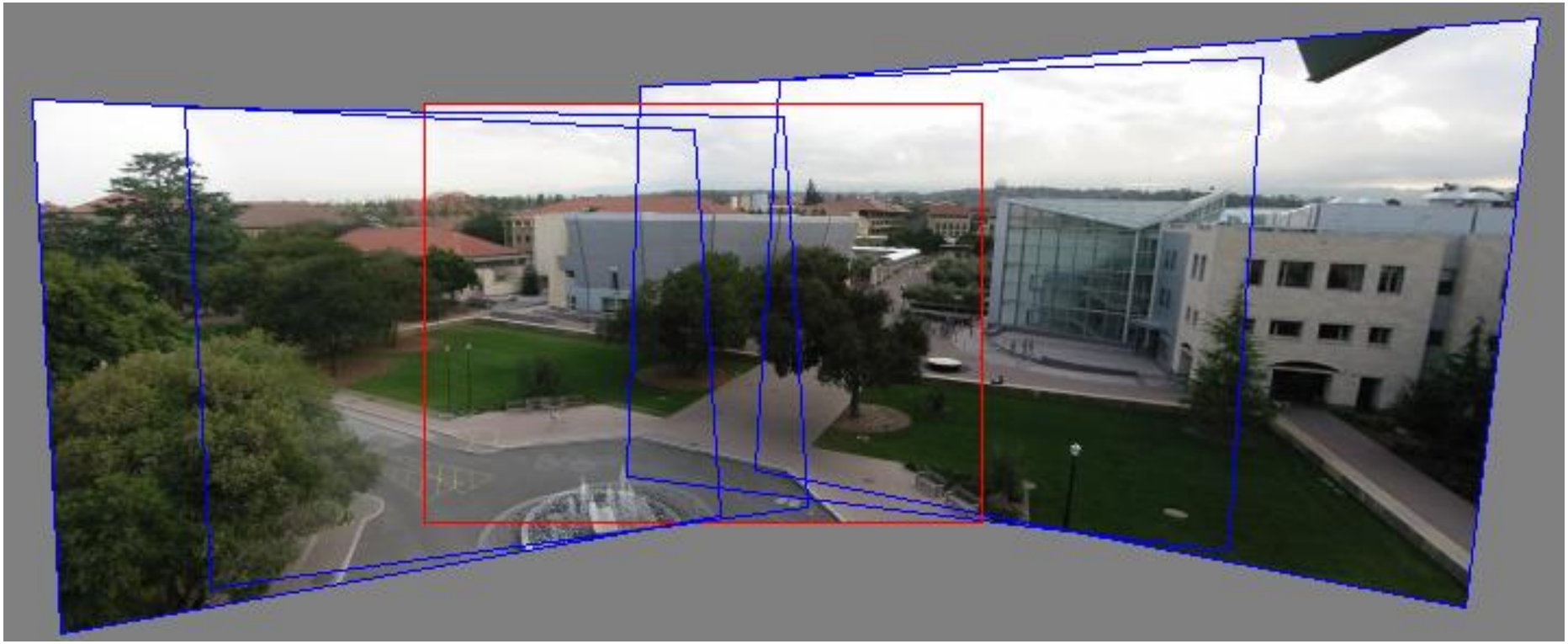
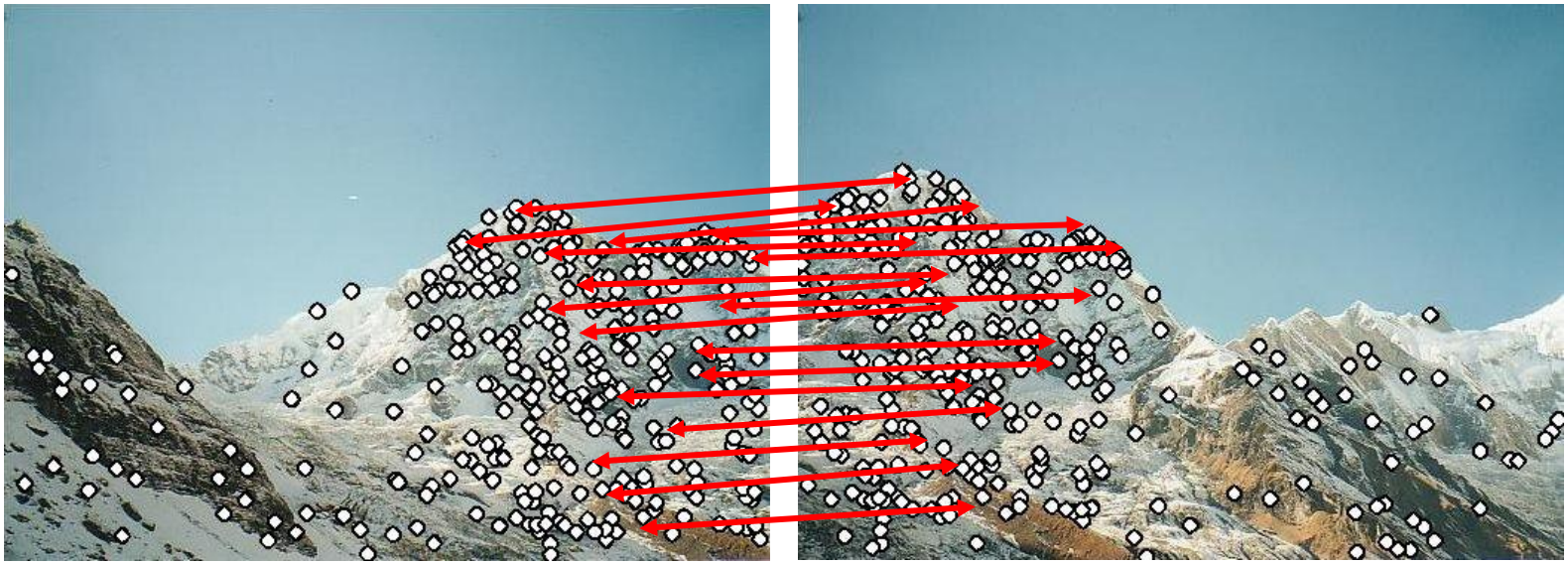


Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fall/

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

Robust feature-based alignment

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

Local features: desired properties

UNIVERSITÄT BONN

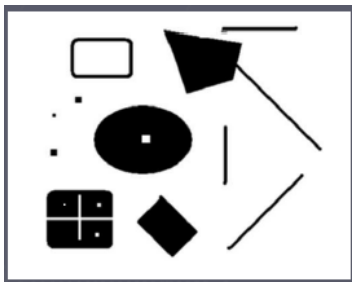
- Repeatability
  - The same feature can be found in several images despite geometric and photometric transformations
- Saliency
  - Each feature has a distinctive description
- Locality
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion
- Sometimes: Compactness and efficiency
  - Low dimensionality

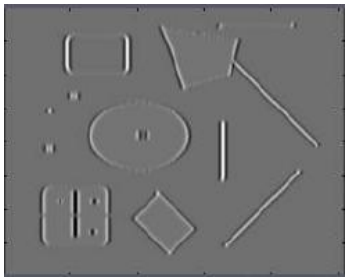
Kristen Grauman

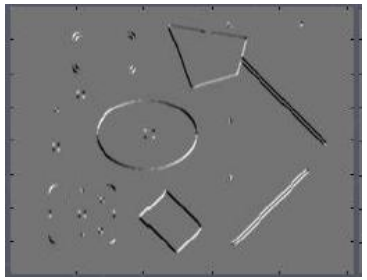
Corners as distinctive interest points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).







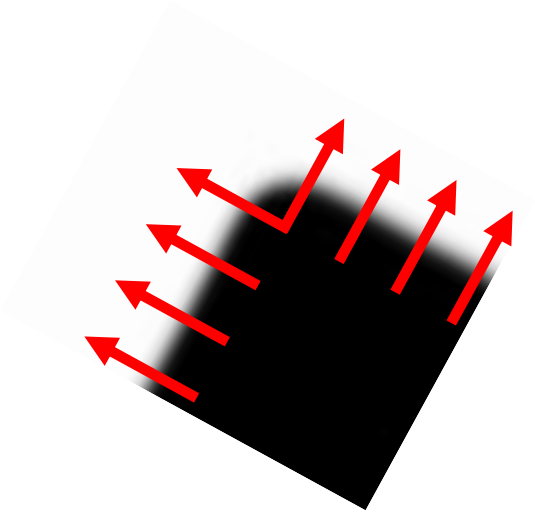
Notation:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

Kristen Grauman

What does this matrix reveal?

Since *M* is symmetric, we have $M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$

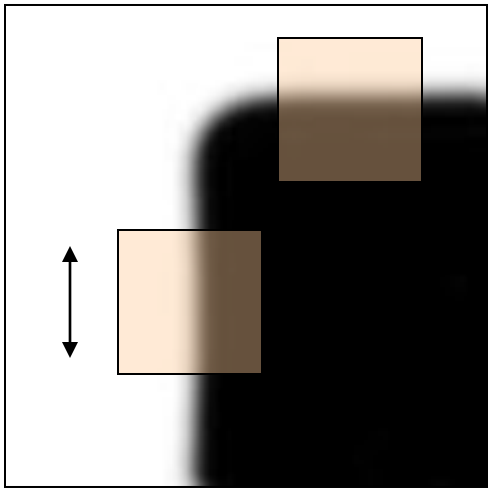


$$Mx_i = \lambda_i x_i$$

The *eigenvalues* of *M* reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

Kristen Grauman

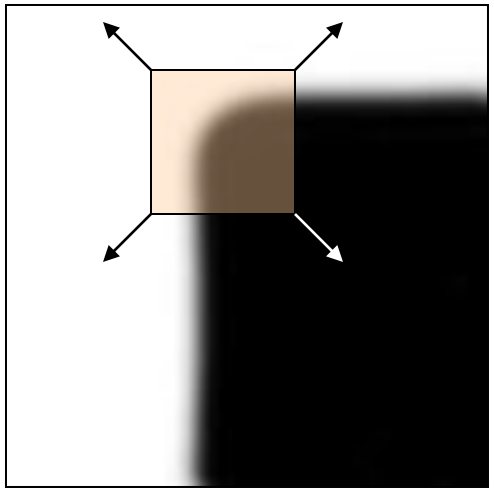
Corner response function



“edge”:

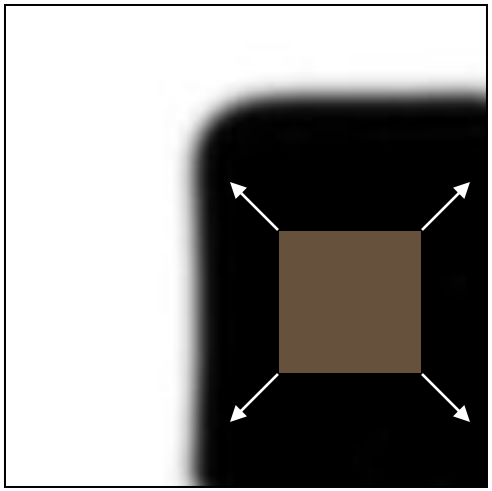
$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$



“corner”:

λ_1 and λ_2 are large, $\lambda_1 \sim \lambda_2$;



“flat” region

λ_1 and λ_2 are small;

$$f = |M| - k\left(\mathrm{Tr}(M)\right)^2 = \lambda_1\lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2$$

Harris corner detector

1) Compute *M* matrix for each image window to get their *cornerness* scores.
2) Find points whose surrounding window gave large corner response (*f*> threshold)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Kristen Grauman

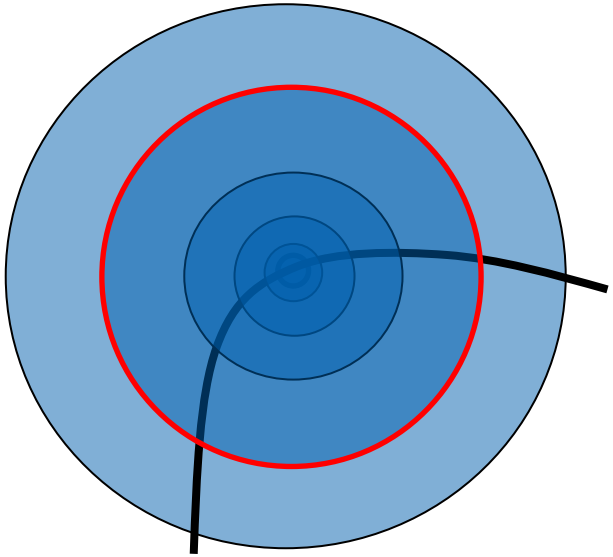
Example of Harris application

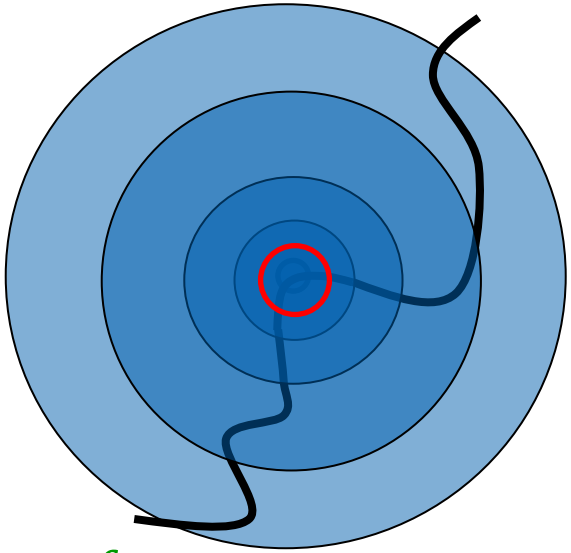
Kristen Grauman

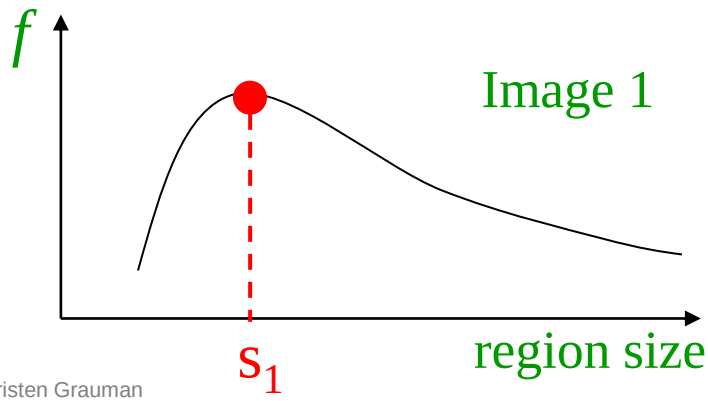
Automatic scale selection

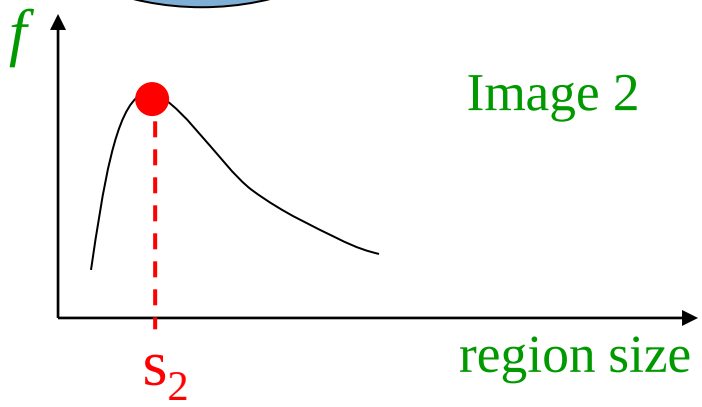
Intuition:

- Find scale that gives local maxima of some function *f* in both position and scale.



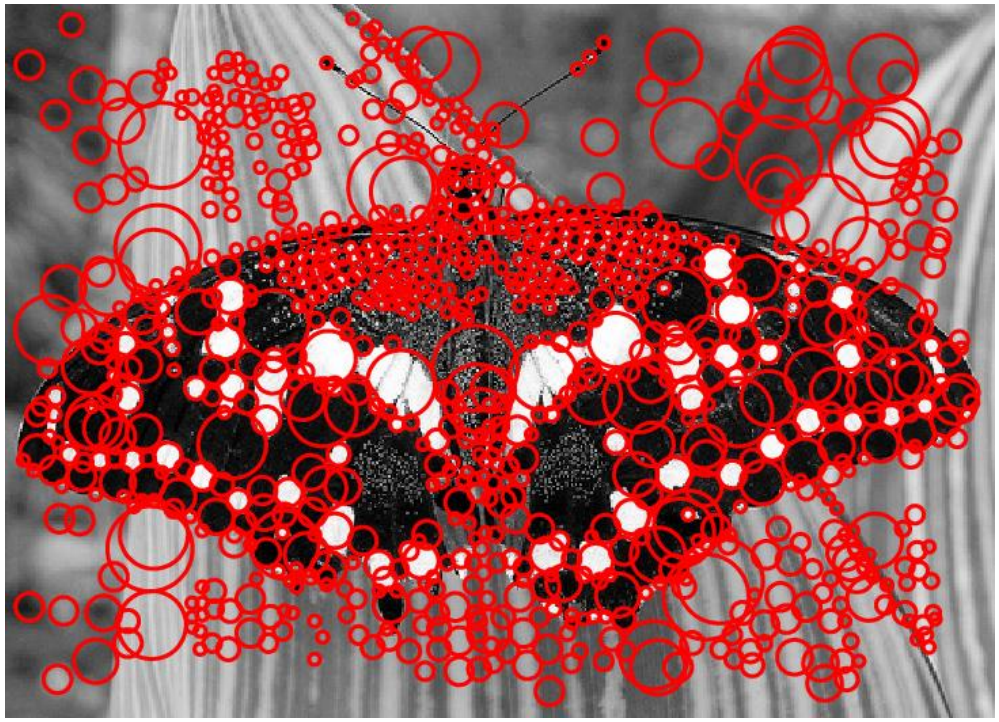






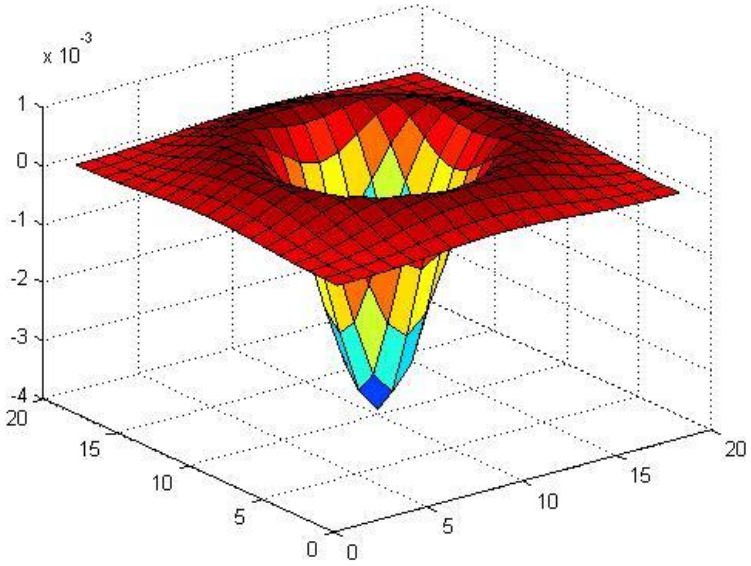
Kristen Grauman

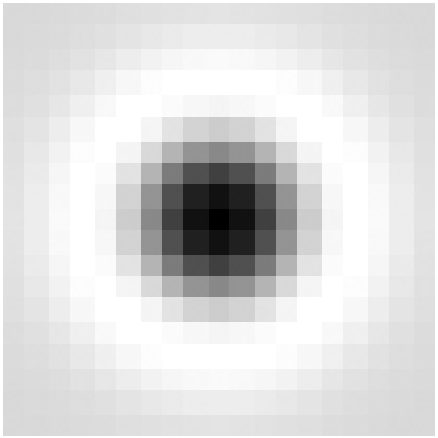
Scale invariant blob detection



Kristen Grauman

Blob detection in 2D





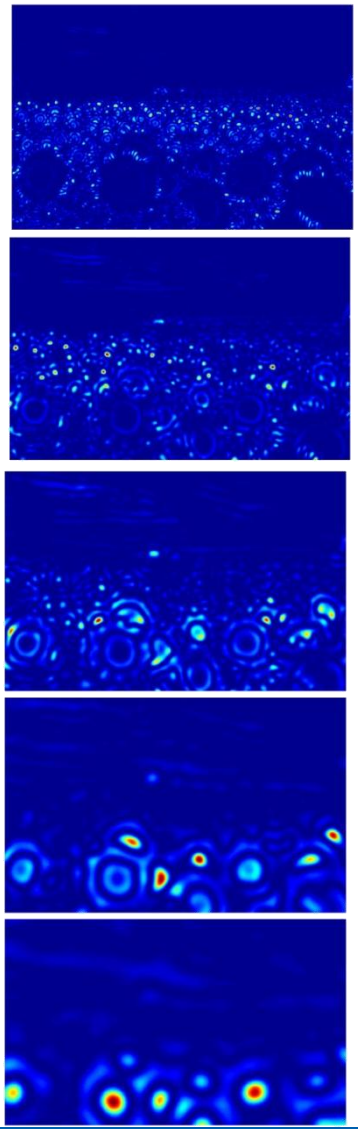
- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

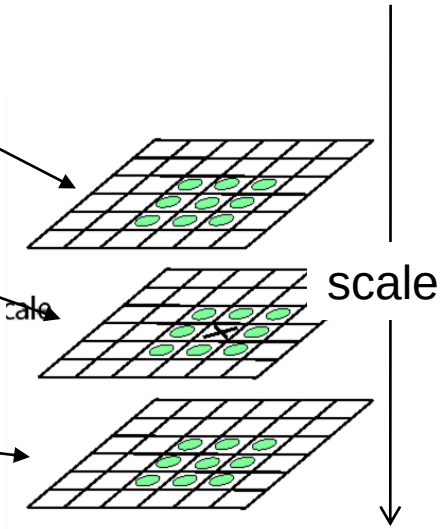
Kristen Grauman

Scale invariant interest points

Interest points are local maxima in both position and scale.

$$L_{xx}(\sigma) + L_{yy}(\sigma)$$





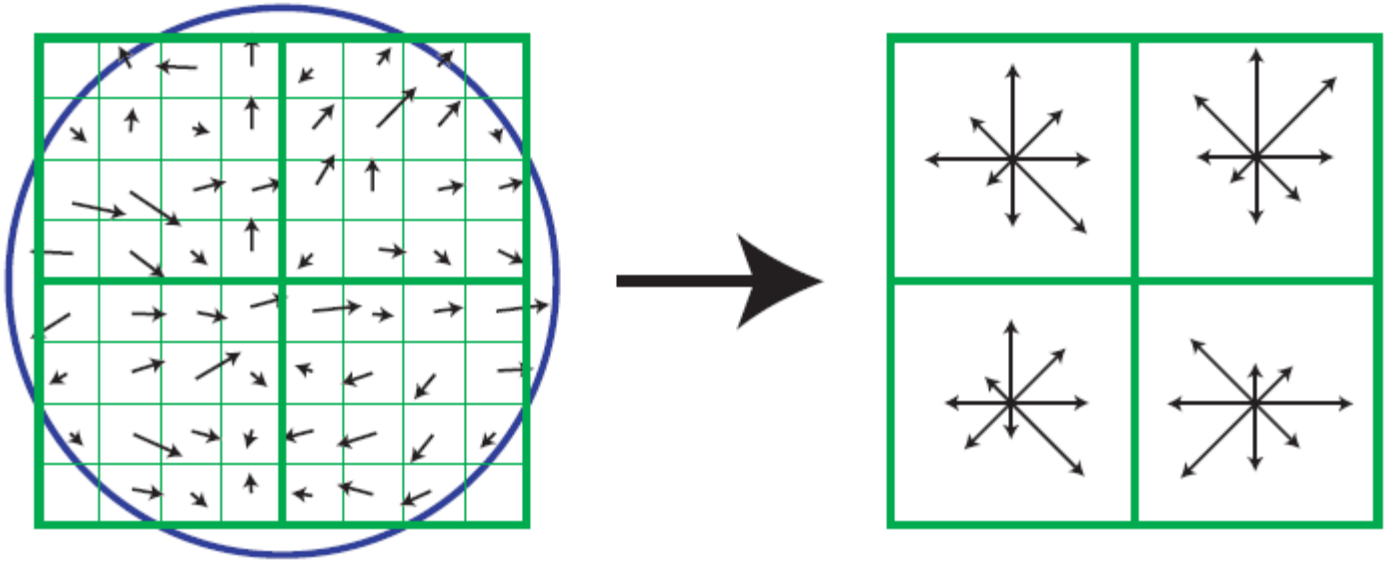
⇒ **List of**
(x, y, σ)

Squared filter response maps

Kristen Grauman

SIFT descriptor [Lowe 2004]

- 128 D vector



- Normalize to 1 (robust to illumination changes)
- Entries above 0.2 are set to 0.2 (reduce impact of very strong gradients e.g. shadow)
- Normalize to 1

Matching local features

Image 1

Image 2

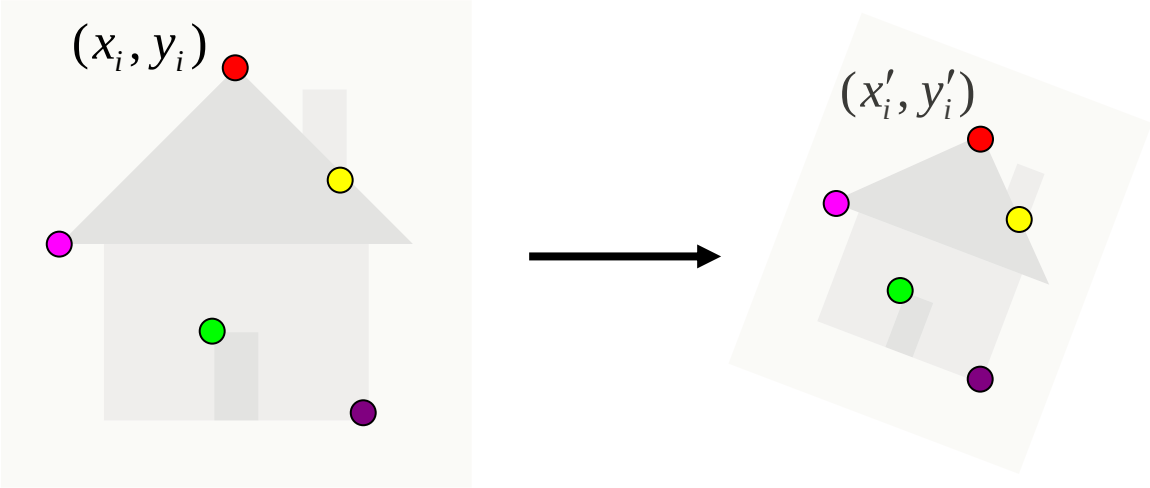
To add robustness to matching, can consider **ratio** : distance to best match / distance to second best match

If low, first match looks good.

If high and close to one, could be ambiguous match.

Kristen Grauman

Fitting an affine transformation

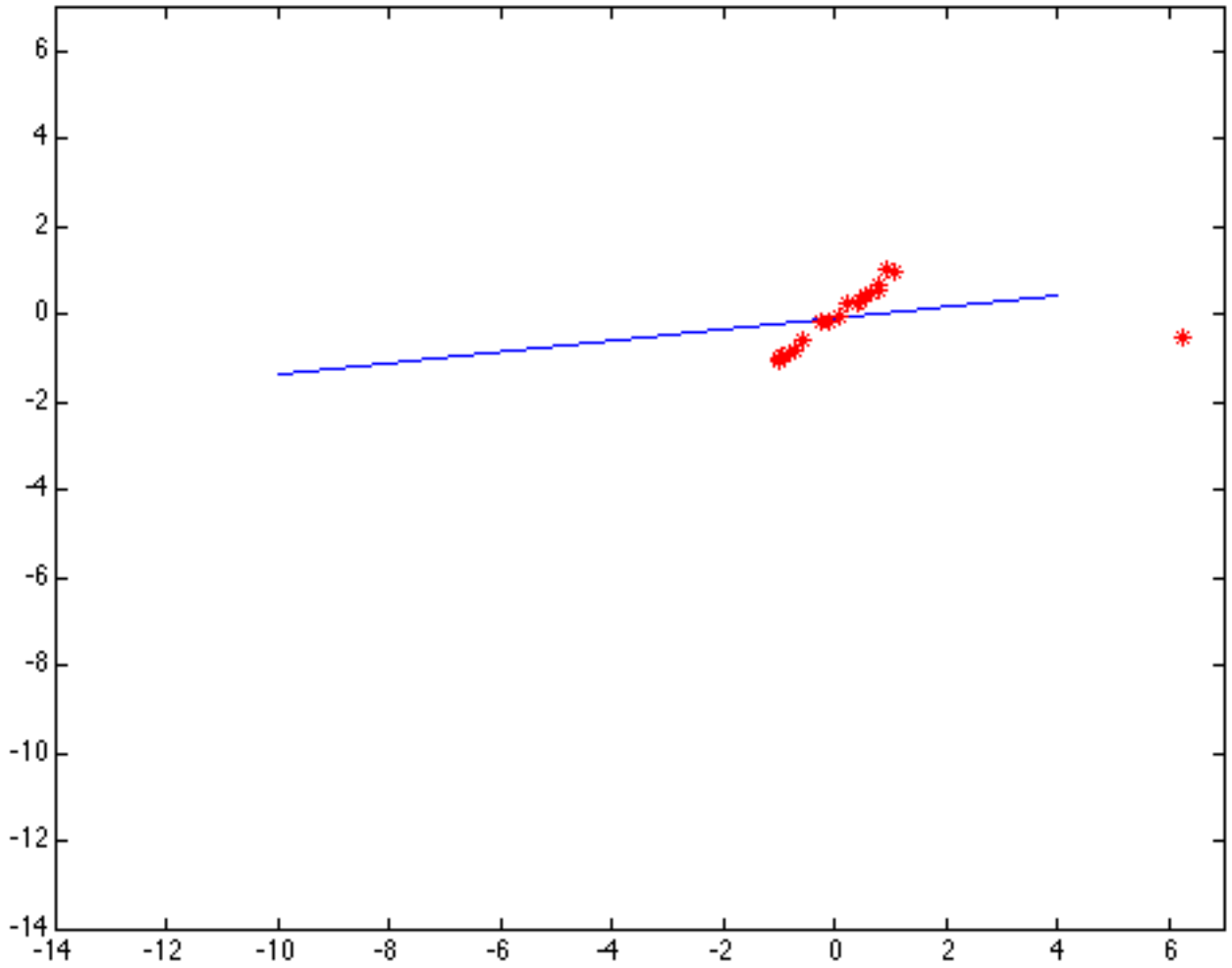


$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & \cdots & & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & \cdots & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x'_i \\ y'_i \\ \cdots \end{bmatrix}$$

Source: K. Grauman

Outliers affect least squares fit



Kristen Grauman

RANSAC: General form

UNIVERSITÄT BONN

RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute estimate of transformation on all of the inliers

Keep the transformation with the largest number of inliers

Kristen Grauman

How Many Samples?

With confidence p

$$\text{P(bad } k \text{ times)} = \big(1 - \text{P(good)}\big)^k \leq 1 - p$$

$$k \log \big(1 - \text{P(good)}\big) \leq \log(1 - p)$$

$$k \geq \log(1 - p) \,/\, \log \big(1 - \text{P(good)}\big)$$

$$\text{P(good)} = \frac{\binom{I}{m}}{\binom{N}{m}} = \prod_{j=0}^{m-1} \frac{I - j}{N - j}$$

Ondřej Chum

Homography

(x, y)

$\left(wx'/w, wy'/w\right) = (x', y')$

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\mathbf{p'} \qquad \mathbf{H} \qquad \mathbf{p}$

Kristen Grauman

Solving for homographies

UNIVERSITÄT BONN

$$\begin{bmatrix} {}_{\mathrm{w}}x_i' \\ {}_{\mathrm{w}}y_i' \\ {}_{\mathrm{w}} \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{A} \qquad \underset{9}{h} \qquad \underset{2n}{0}$$

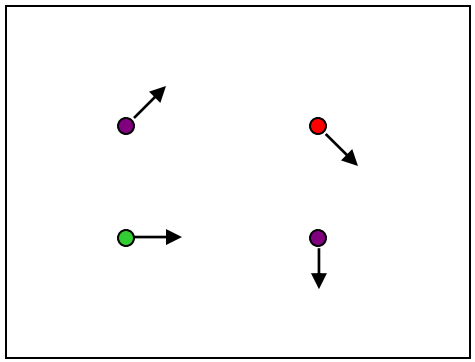
Defines a least squares problem: minimize $\|\mathbf{Ah} - \mathbf{0}\|^2$

- Since h is only defined up to scale, solve for unit vector $\hat{h}$
- Solution: $\hat{h}$ = eigenvector of A^TA with smallest eigenvalue
- Works with 4 or more points

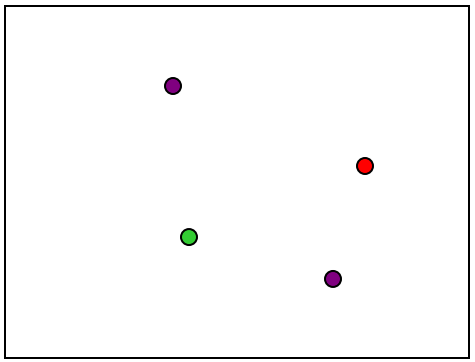
Visual motion

Lana Lazebnik

Estimating optical flow



$I(x,y,t-1)$



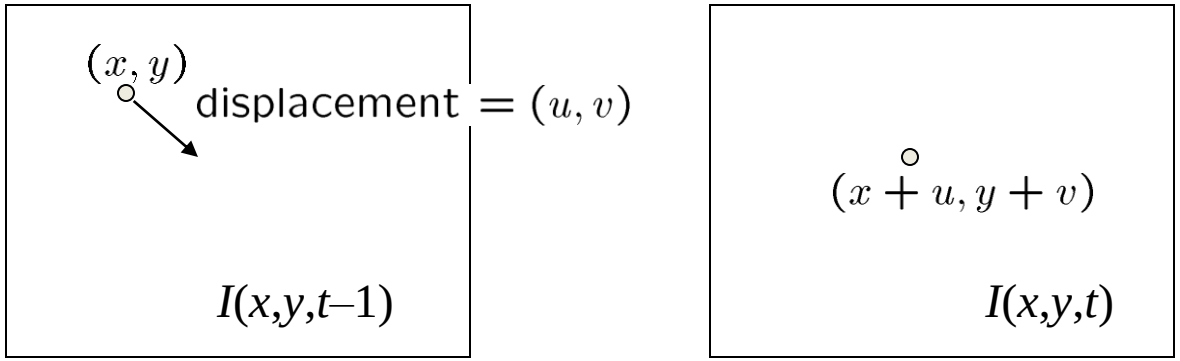
$I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field $u(x,y)$ and $v(x,y)$ between them

- Key assumptions
  - Brightness constancy: projection of the same point looks the same in every frame
  - Small motion: points do not move very far
  - Spatial coherence: points move like their neighbors

Lana Lazebnik

The brightness constancy constraint



- Brightness Constancy Equation:

$$I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t)$$

Linearizing the right side using Taylor expansion:

$$I(x, y, t-1) \approx I(x, y, t) + I_x \cdot u(x, y) + I_y \cdot v(x, y)$$

Hence, $$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

Lana Lazebnik

Lucas-Kanade flow

- Linear least squares problem

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Solution given by $(A^T A)\ d = A^T b$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $A^T b$

The summations are over all pixels in the window

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

Lana Lazebnik

Lucas-Kanade flow

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

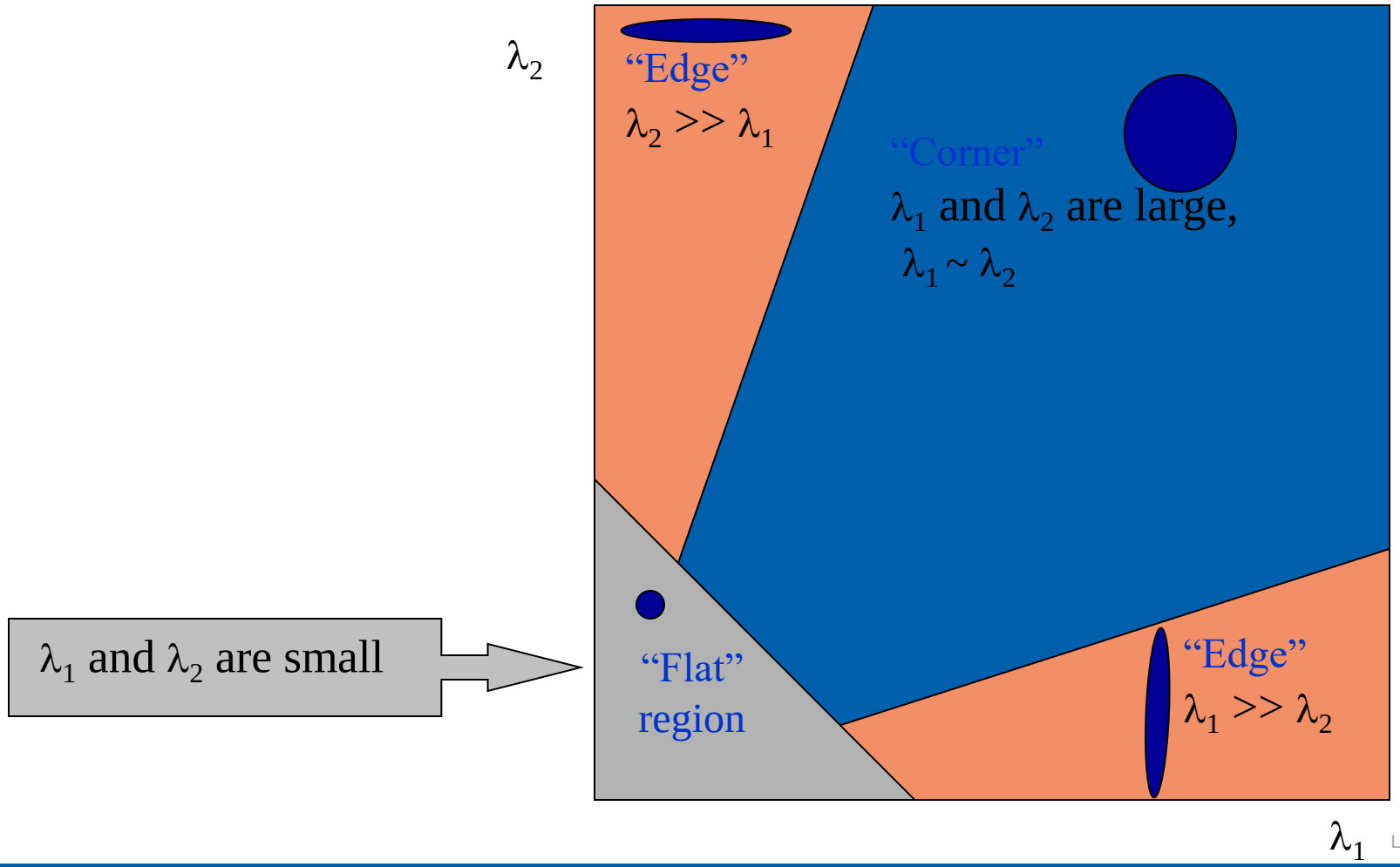
$A^T A$ $\qquad$ $A^T b$

- Recall the Harris corner detector: $M = A^T A$ is the *second moment matrix*
- We can figure out whether the system is solvable by looking at the eigenvalues of the second moment matrix
  - The eigenvectors and eigenvalues of M relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change, and the other eigenvector is orthogonal to it

Lana Lazebnik

Interpreting the eigenvalues

Classification of image points using eigenvalues of the second moment matrix:



Lana Lazebnik

Errors in Lucas-Kanade

- The motion is large (larger than a pixel)
  - Iterative refinement
  - Coarse-to-fine estimation
  - Exhaustive neighborhood search (feature matching)
- A point does not move like its neighbors
  - Motion segmentation
- Brightness constancy does not hold
  - Exhaustive neighborhood search with normalized correlation

Lana Lazebnik

Shi-Tomasi feature tracker

UNIVERSITÄT BONN

- Find good features using eigenvalues of second-moment matrix
  - Key idea: “good” features to track are the ones whose motion can be estimated reliably
- From frame to frame, track with Lucas-Kanade
  - This amounts to assuming a translation model for frame-to-frame feature movement
- Check consistency of tracks by *affine* registration to the first observed instance of the feature
  - Affine model is more accurate for larger displacements
  - Comparing to the first frame helps to minimize drift

J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Lana Lazebnik

Tracking example

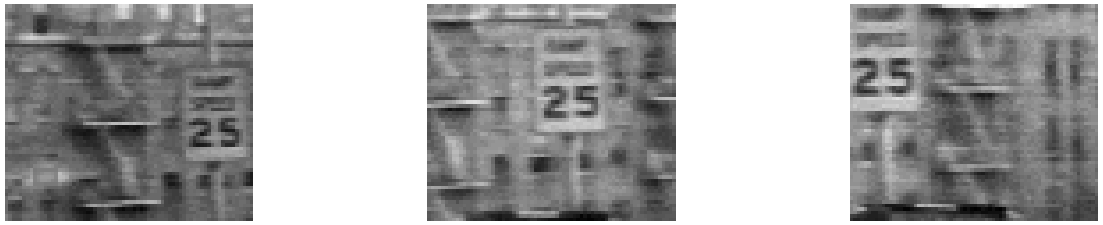


Figure 1: Three frame details from Woody Allen's *Manhattan*. The details are from the 1st, 11th, and 21st frames of a subsequence from the movie.

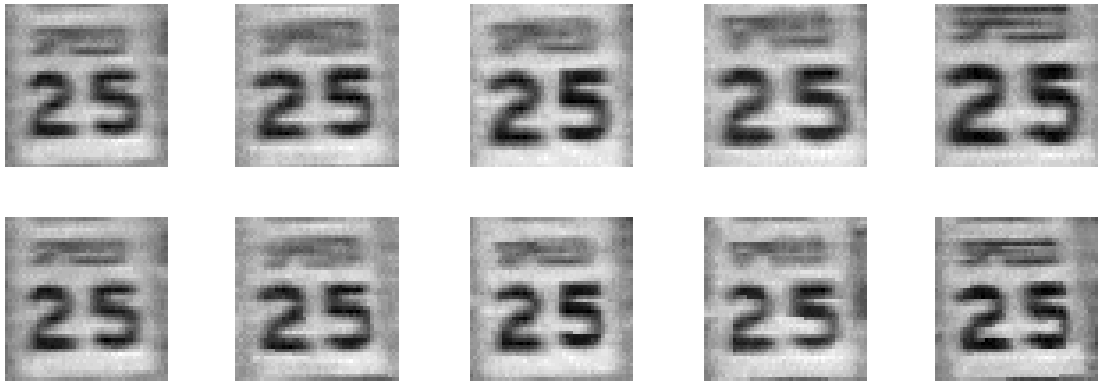


Figure 2: The traffic sign windows from frames 1,6,11,16,21 as tracked (top), and warped by the computed deformation matrices (bottom).

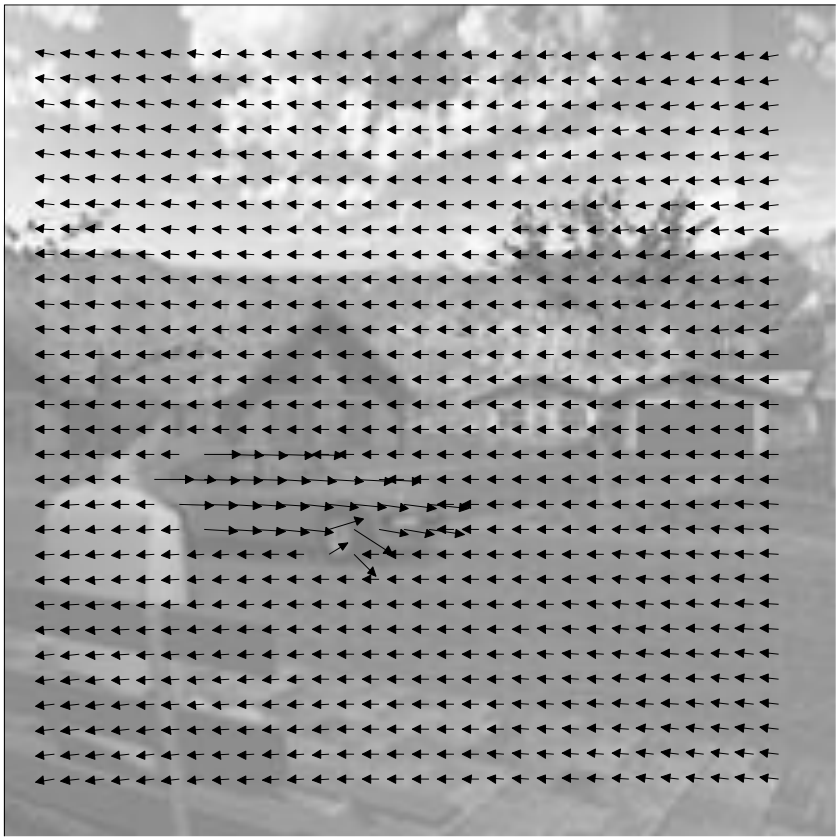
J. Shi and C. Tomasi. Good Features to Track. CVPR 1994.

Lana Lazebnik

Dense motion estimation

- For each pixel in an image, optical flow determines the corresponding pixel in the next image
- Can be cast as a variational problem

Image sequence $I(x, y, t)$



Optical flow field $(u, v)(x, y, t)$.

Thomas Brox

Horn-Schunck model

UNIVERSITÄT BONN

- Most basic model → good to start with

- Assumption I:
  Gray value constancy (optic flow constraint)

$$I_x u + I_y v + I_t = 0$$

reminder: subscripts denote partial derivatives

  Linearized version of

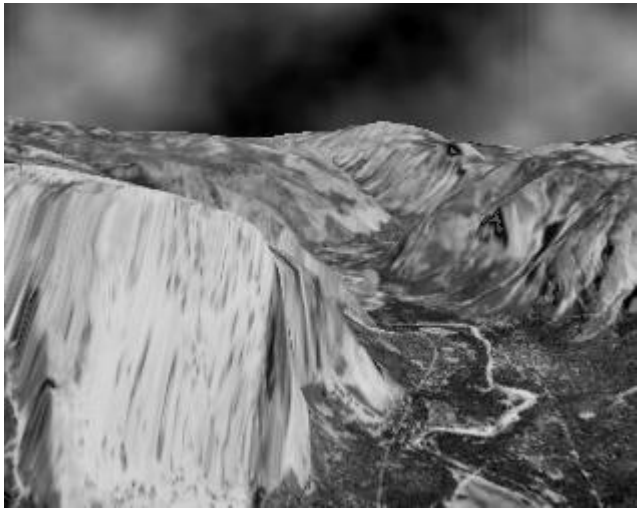
$$I(x+u, y+v, t+1) - I(x, y, t) = 0$$

- Assumption II:
  Smoothness of optic flow field
  $|\nabla u|^2 + |\nabla v|^2 \rightarrow \min$
- Combined in an energy functional:

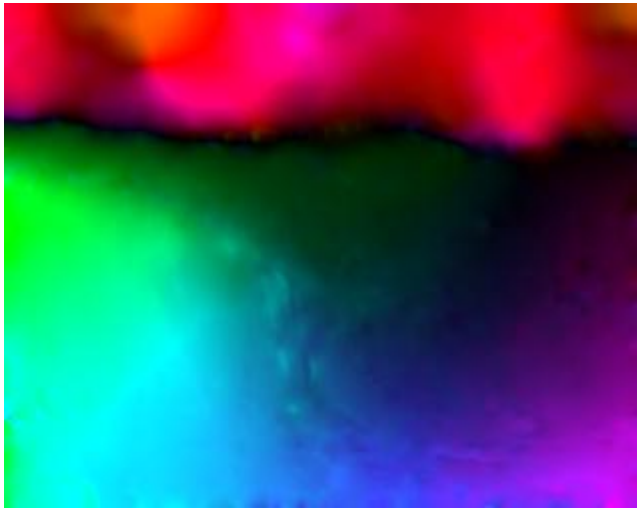
$$E(u,v) = \int_\Omega (I_x u + I_y v + I_t)^2 + \alpha(|\nabla u|^2 + |\nabla v|^2)\, dxdy$$

Thomas Brox

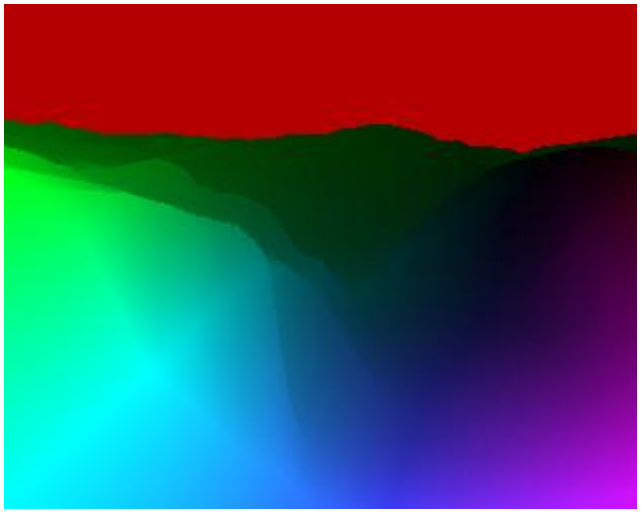
Results with Horn-Schunck method



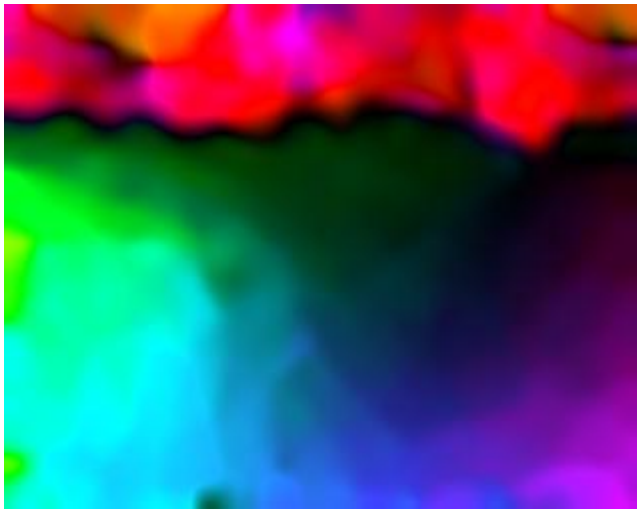
Input sequence



Horn-Schunck (7.17°)



Ground truth



Lucas-Kanade (8.78°)

Thomas Brox

Motion discontinuities

Occlusions

Illumination changes

Large displacements

Really large displacements

Thomas Brox

Incorporating correspondences from descriptor matching

UNIVERSITÄT BONN

$$E(\mathbf{w}(\mathbf{x})) = \int \Psi\left(|I_2(\mathbf{x}+\mathbf{w}(\mathbf{x})) - I_1(\mathbf{x})|^2\right) d\mathbf{x}$$
$$+\gamma \int \Psi\left(|\nabla I_2(\mathbf{x}+\mathbf{w}(\mathbf{x})) - \nabla I_1(\mathbf{x})|^2\right) d\mathbf{x}$$
$$+\alpha \int \Psi\left(|\nabla u(\mathbf{x})|^2 + |\nabla v(\mathbf{x})|^2\right) d\mathbf{x}$$

point correspondences from HOG descriptors

$$+\beta \int \rho(\mathbf{x})\, \Psi((u(\mathbf{x}) - u'(\mathbf{x}))^2 + (v(\mathbf{x}) - v'(\mathbf{x}))^2) d\mathbf{x}$$

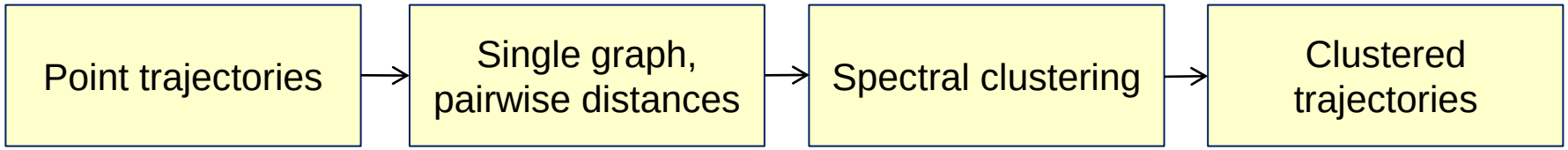
Matching score

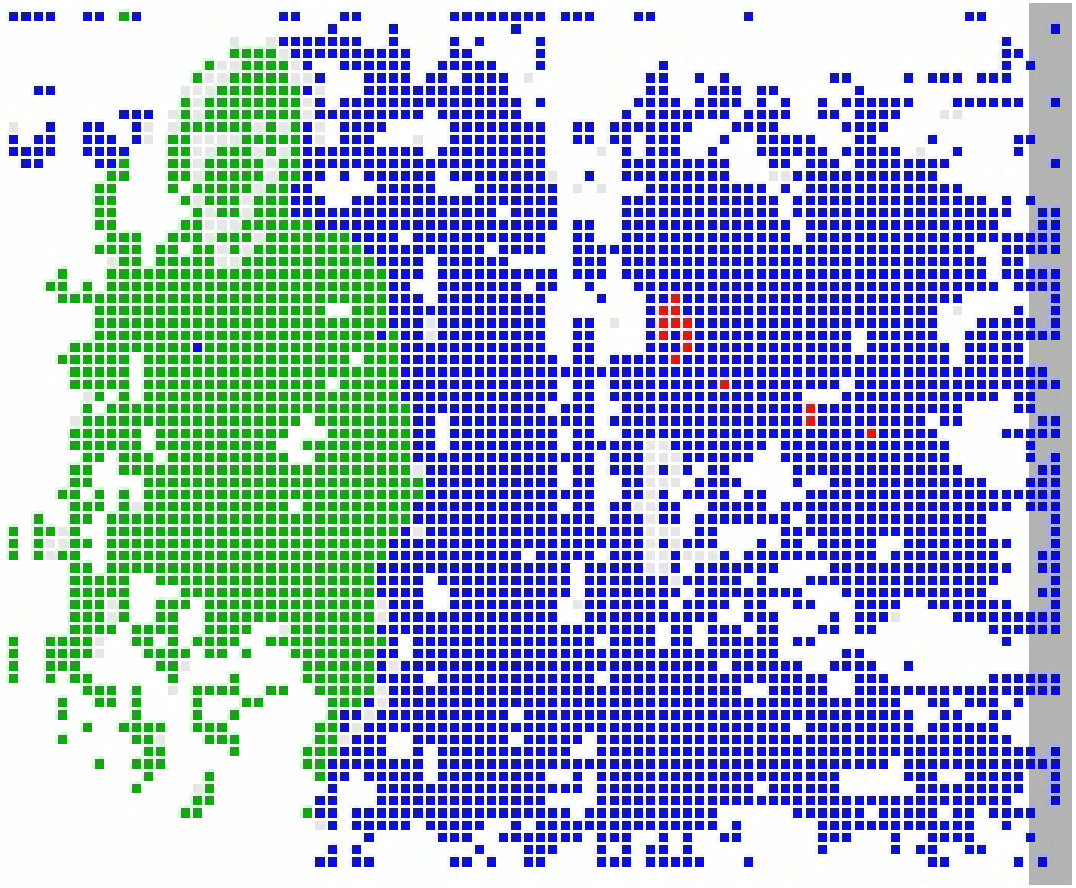
Robust function

Flow = correspondence vector

Thomas Brox

Clustering of point trajectories (Brox-Malik 2010)





Thomas Brox

UNIVERSITÄT BONN