UNIVERSITÄT BONN

Juergen Gall

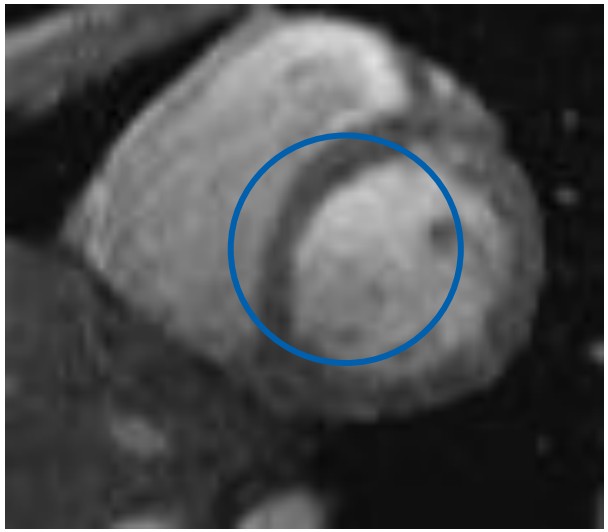
Snakes and Level Sets
MA-INF 2201 - Computer Vision
WS24/25

Deformable contours

UNIVERSITÄT BONN

a.k.a. active contours, snakes

Given: initial contour (model) near desired object



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

Source: Y. Boykov

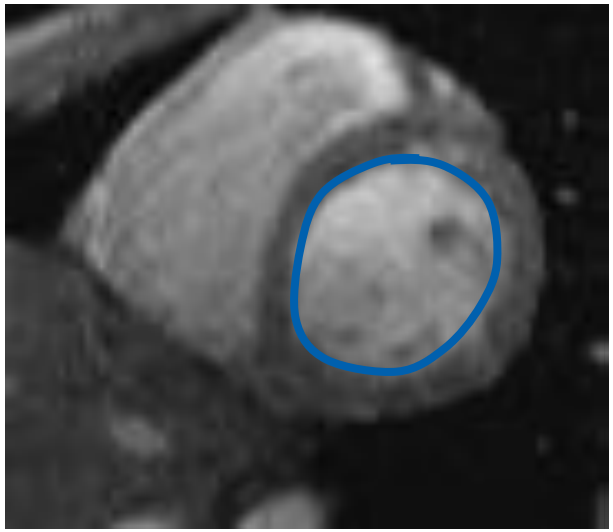
Deformable contours

UNIVERSITÄT BONN

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary



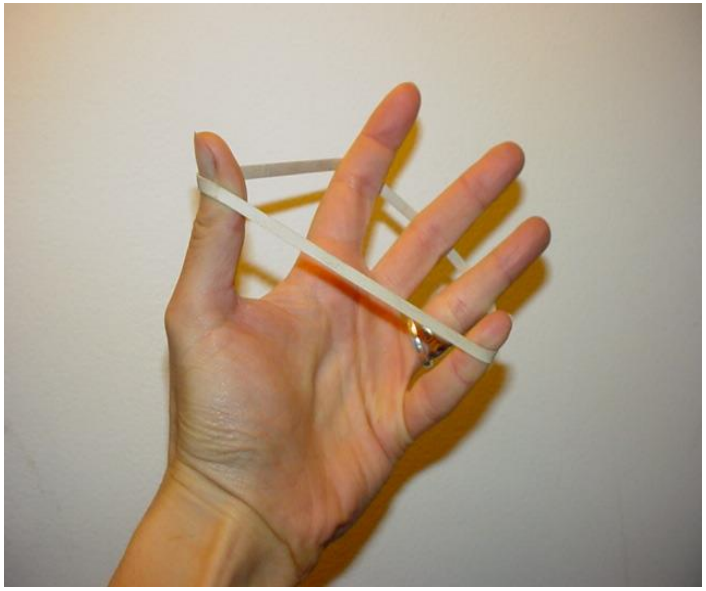
Main idea: elastic band is iteratively adjusted so as to

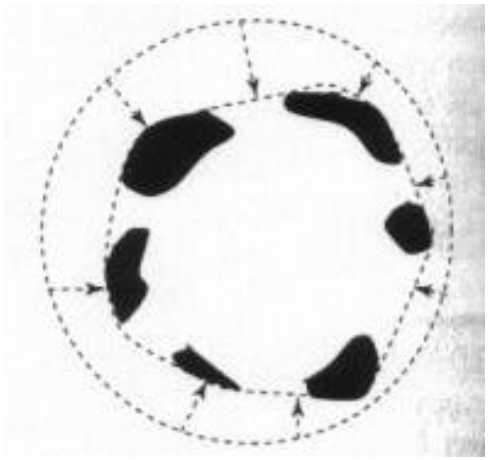
- be near image positions with high gradients, **and**
- satisfy shape “preferences” or contour priors

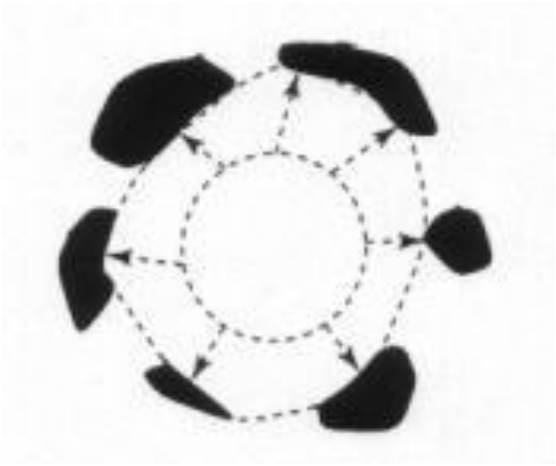
[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Source: Y. Boykov

Deformable contours: intuition

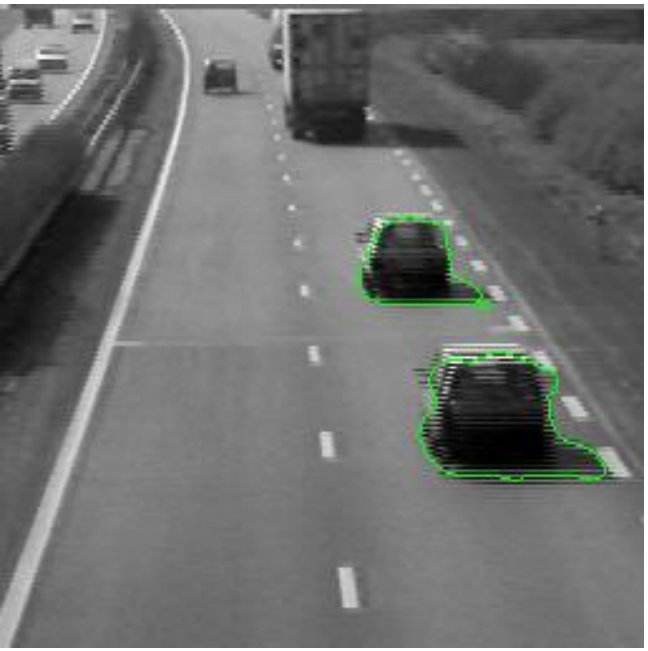
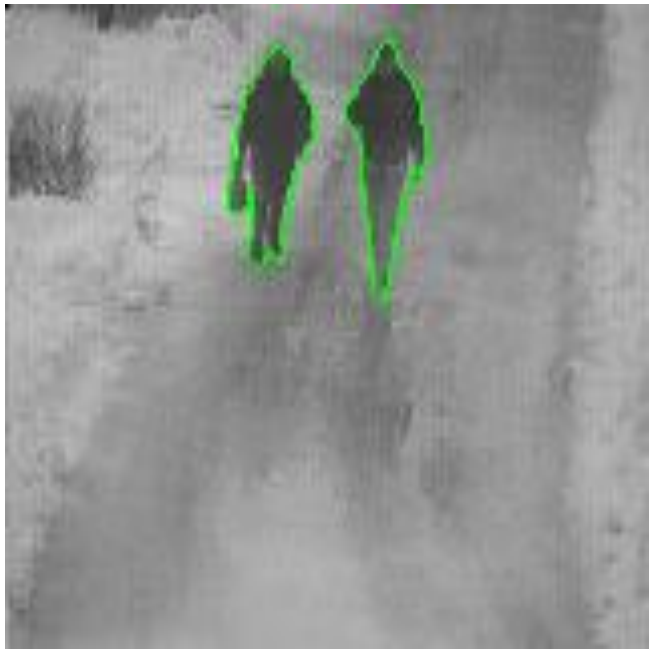
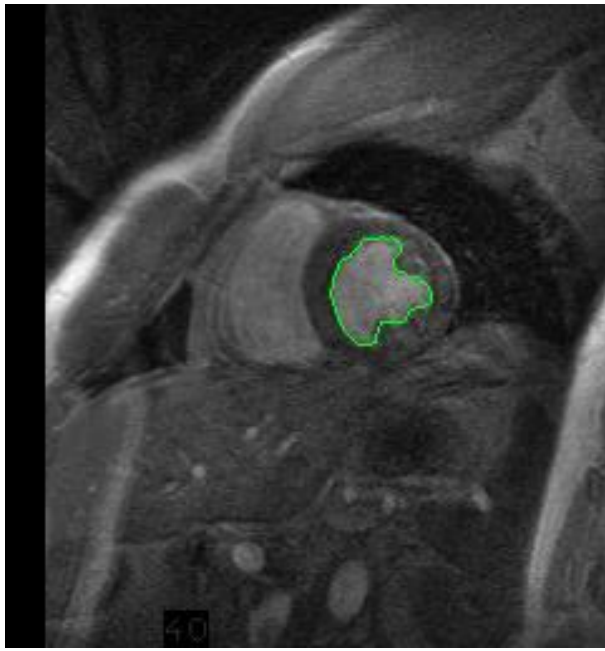






Source: K. Grauman

Why do we want to fit deformable shapes?

- Non-rigid, deformable objects can change their shape over time.

Source: K. Grauman

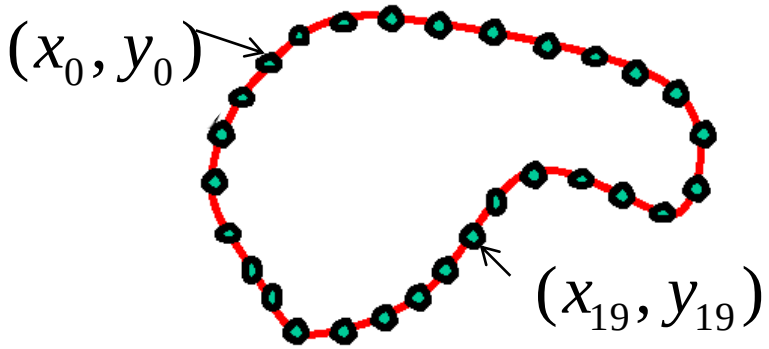
Aspects we need to consider

- Representation of the contours
- Defining the energy functions
- Minimizing the energy function

Source: K. Grauman

Representation

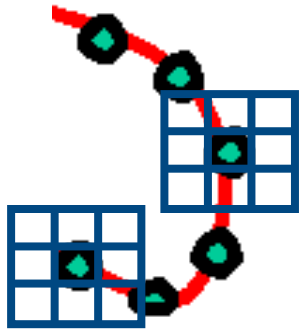
- We'll consider a discrete representation of the contour, consisting of a list of 2d point positions ("vertices").



$$v_i = (x_i, y_i),$$

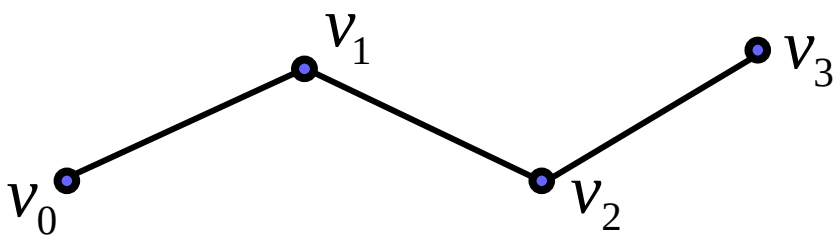
for $i = 0, 1, \ldots, n-1$

- At each iteration, we'll have the option to move each vertex to another nearby location ("state").

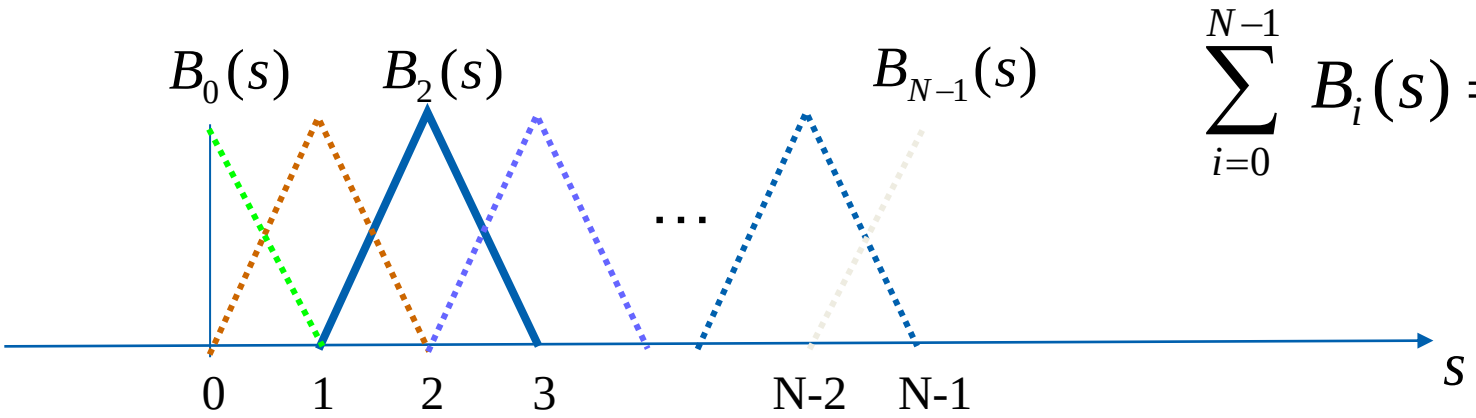


Source: K. Grauman

Polygonal snakes



$$C(s) = \sum_{i=0}^{N-1} v_i \cdot B_i(s) \qquad 0 \le s \le (N-1)$$

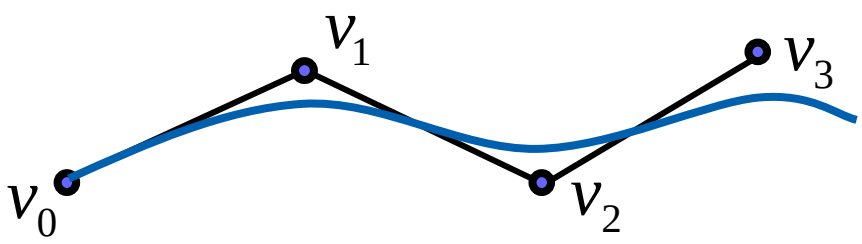


$$\sum_{i=0}^{N-1} B_i(s) = 1 \qquad \forall s$$

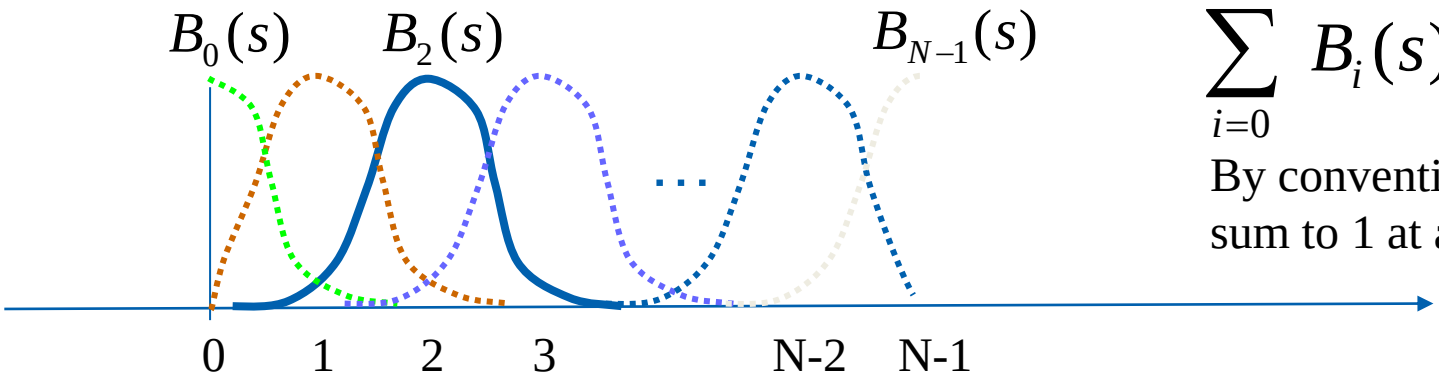
- Contour as a linear combination of N basis functions

Source: Y. Boykov

B-spline snakes



$$C(s) = \sum_{i=0}^{N-1} v_i \cdot B_i(s) \qquad 0 \le s \le (N-1)$$



$$\sum_{i=0}^{N-1} B_i(s) = 1 \qquad \forall s$$

By convention, B-splines sum to 1 at all points s

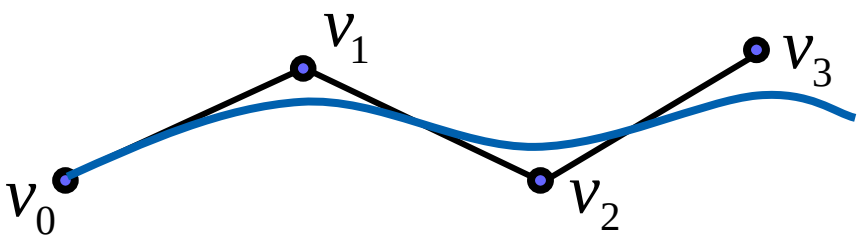
- Each $B_i(s)$ is typically a simple combination of polynomials

A. Blake and M. Isard. **Active Contours**. Springer 1998

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:

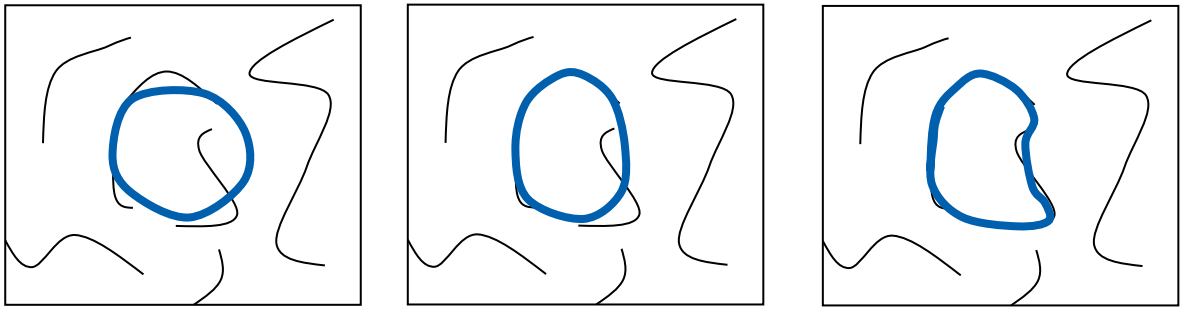
$$\{v_0, v_1, v_2, \ldots, v_{N-1}\}$$

Source: Y. Boykov

Fitting deformable contours

How should we adjust the current contour to form the new contour at each iteration?

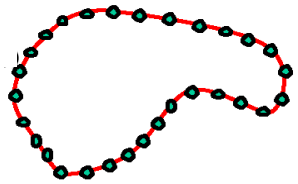
- Define a cost function (“energy” function) that says how good a candidate configuration is.
- Seek next configuration that minimizes that cost function.



Source: K. Grauman

Energy function

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



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

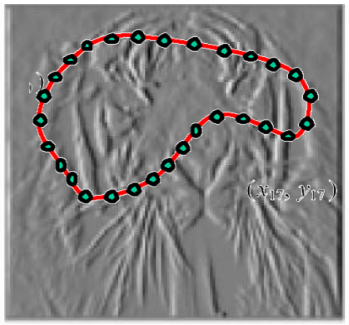
Internal energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

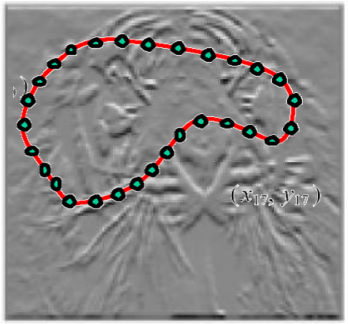
External energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.

Source: K. Grauman

External image energy

- Gradient images $G_x(x, y)$ and $G_y(x, y)$





- External energy at a point on the curve is:

$$E_{external}(v) = -(|G_x(v)|^2 + |G_y(v)|^2)$$

- External energy for the whole curve:

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

Source: K. Grauman

Internal energy

A priori, we want to favor **smooth** shapes, contours with **low curvature**, contours similar to a **known shape**, etc. to balance what is actually observed (i.e., in the gradient image).

Source: K. Grauman

Internal energy

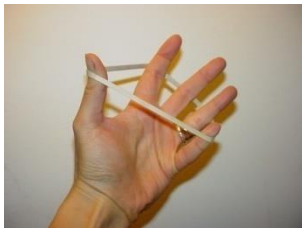
For a *continuous* curve, a common internal energy term is the “bending energy”.

At some point *v(s)* on the curve, this is:

$$E_{internal}(v(s)) = \alpha \left| \frac{dv}{ds} \right|^2 + \beta \left| \frac{d^2 v}{d^2 s} \right|^2$$

Tension, Elasticity

Stiffness, Curvature

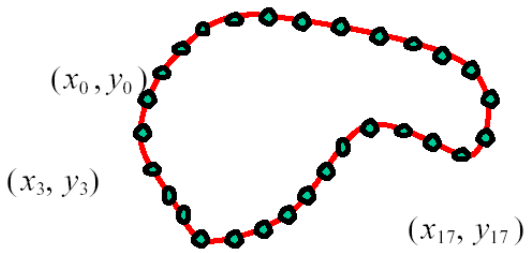


Source: K. Grauman

Internal energy

- For our discrete representation,

$$v_i = (x_i, y_i) \qquad i = 0 \ldots n-1$$



$$\frac{dv}{ds} \approx v_{i+1} - v_i \qquad \frac{d^2v}{ds^2} \approx (v_{i+1} - v_i) - (v_i - v_{i-1}) = v_{i+1} - 2v_i + v_{i-1}$$

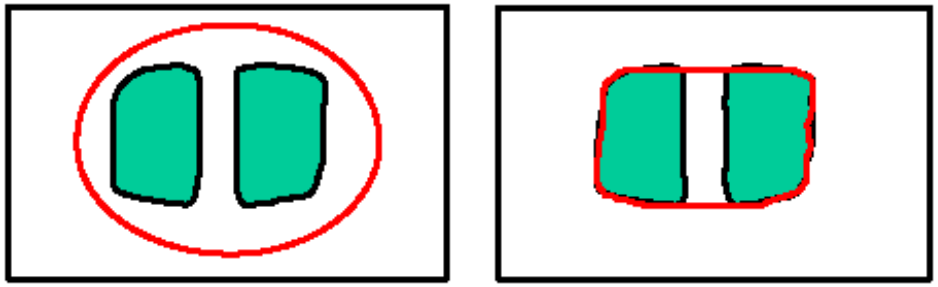
- Note these are derivatives relative to position---not spatial image gradients
- Internal energy for the whole curve:

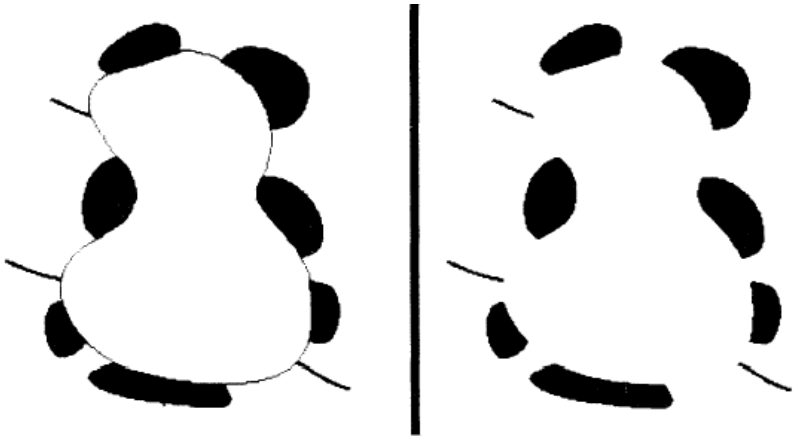
$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left\| v_{i+1} - v_i \right\|^2 + \beta \left\| v_{i+1} - 2v_i + v_{i-1} \right\|^2$$

Source: K. Grauman

Dealing with missing data

- The preferences for low-curvature, smoothness help deal with missing data:





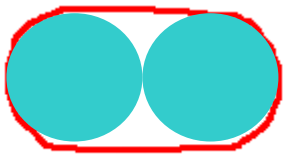
Illusory contours found!

[Figure from Kass et al. 1987]

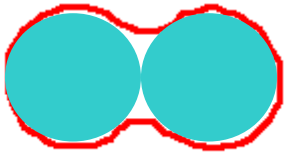
Source: K. Grauman

Total energy: function of the weights

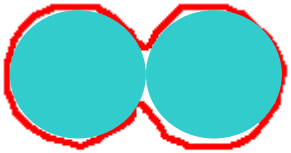
- e.g., α weight controls the penalty for internal elasticity



large α



medium α



small α

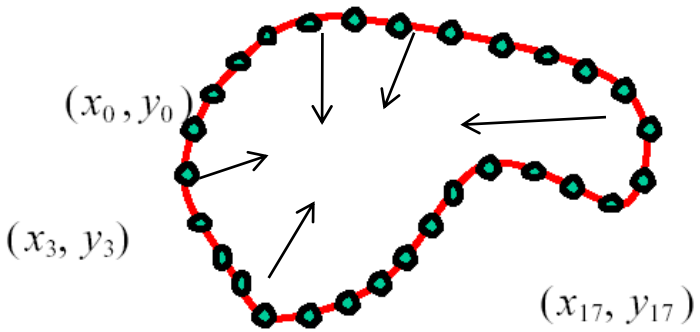
Source: Y. Boykov

Penalizing elasticity

- Current elastic energy definition uses a discrete estimate of the derivative:

$$E_{elastic} = \sum_{i=0}^{n-1} \alpha \left\| v_{i+1} - v_i \right\|^2$$

$$= \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$



What is the possible problem with this definition?

Source: K. Grauman

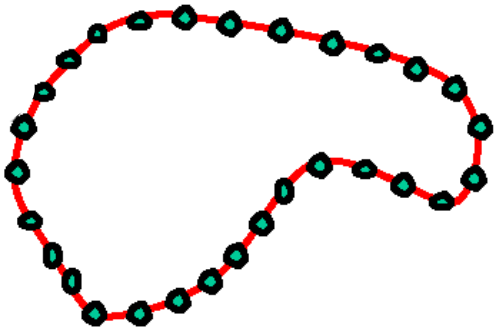
Penalizing elasticity

- Current elastic energy definition uses a discrete estimate of the derivative:

$$E_{elastic} = \sum_{i=0}^{n-1} \alpha \left\| v_{i+1} - v_i \right\|^2$$

Instead:

$$= \alpha \sum_{i=0}^{n-1} \left(\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} - \bar{d} \right)^2$$



where *d* is the average distance between pairs of points – updated at each iteration.

Source: K. Grauman

Total energy: function of the weights

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

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

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

or

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

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.

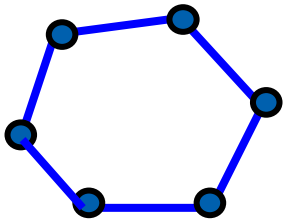
[M. Kass, A. Witkin, D. Terzopoulos. Snakes: Active contour models. IJCV 1987]

Source: T. Brox

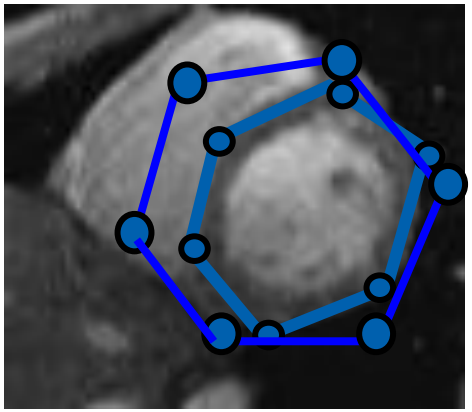
Recap: deformable contour

UNIVERSITÄT BONN

- A simple elastic snake is defined by:
  - A set of *n* points,
  - An internal energy term (tension, bending, plus optional shape prior)
  - An external energy term (gradient-based)



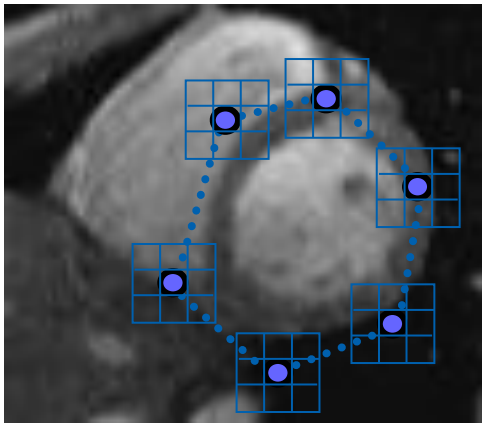
- To use to segment an object:
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy



Source: K. Grauman

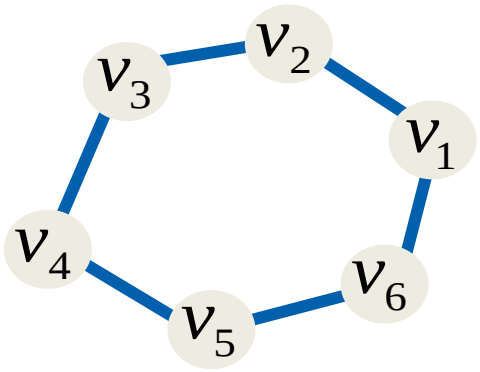
Energy minimization: greedy

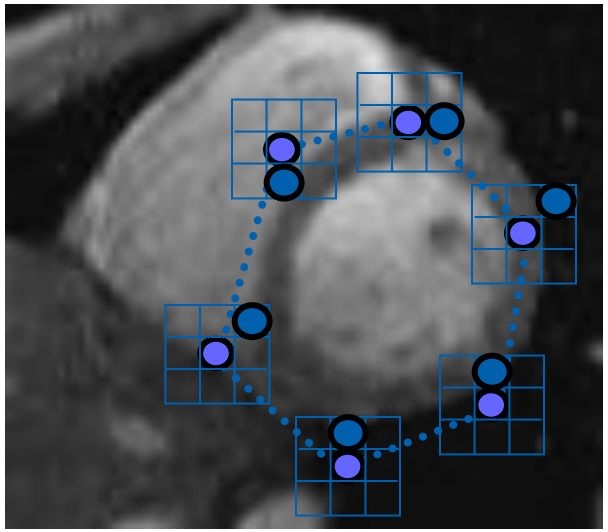
- For each point, search window around it and move to where energy function is minimal
  - Typical window size, e.g., 5 x 5 pixels
- Stop when predefined number of points have not changed in last iteration, or after max number of iterations
- Note:
  - Convergence not guaranteed
  - Need decent initialization



Source: K. Grauman

Dynamic programming





With this form of the energy function, we can minimize using dynamic programming, with the *Viterbi* algorithm.

Source: Y. Boykov

Dynamic programming

UNIVERSITÄT BONN

- Possible because snake energy can be rewritten as a sum of pair-wise interaction potentials:

$$E_{total}(v_1,\ldots,v_n) = \sum_{i=1}^{n-1} E_i(v_i, v_{i+1}) + E_n(v_n)$$

- Or sum of triple-interaction potentials

$$E_{total}(v_1,\ldots,v_n) = \sum_{i=1}^{n-2} E_i(v_i, v_{i+1}, v_{i+2}) + E_{n-1}(v_{n-1}, v_n) + E_n(v_n)$$

Dynamic programming

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

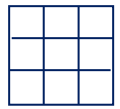
$$\text{argmin}_{w_{1\ldots N}} \left[\sum_{n=1}^{N} U_n(w_n) + \sum_{n=1}^{N-1} P_n(w_n, w_{n+1}) \right]$$

- 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:

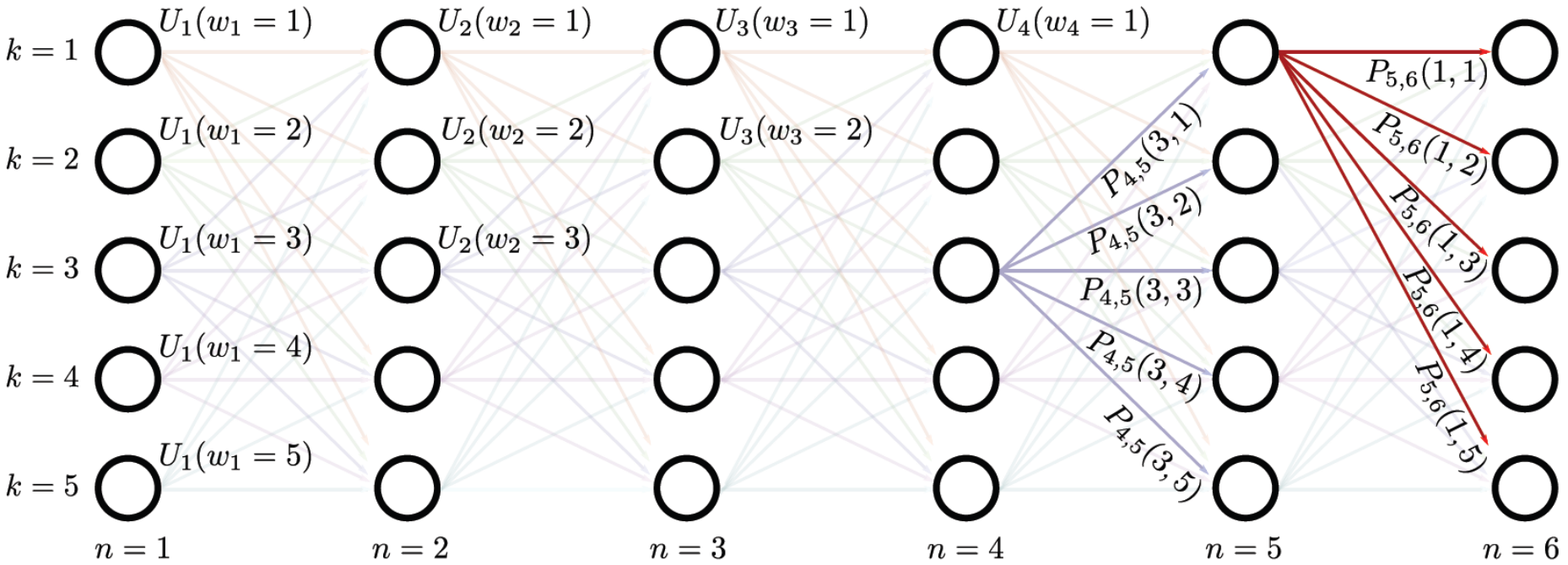


Dynamic programming

Minimizes functions of the form:

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

Set up as cost for traversing graph – each path from left to right is one possible configuration of world states

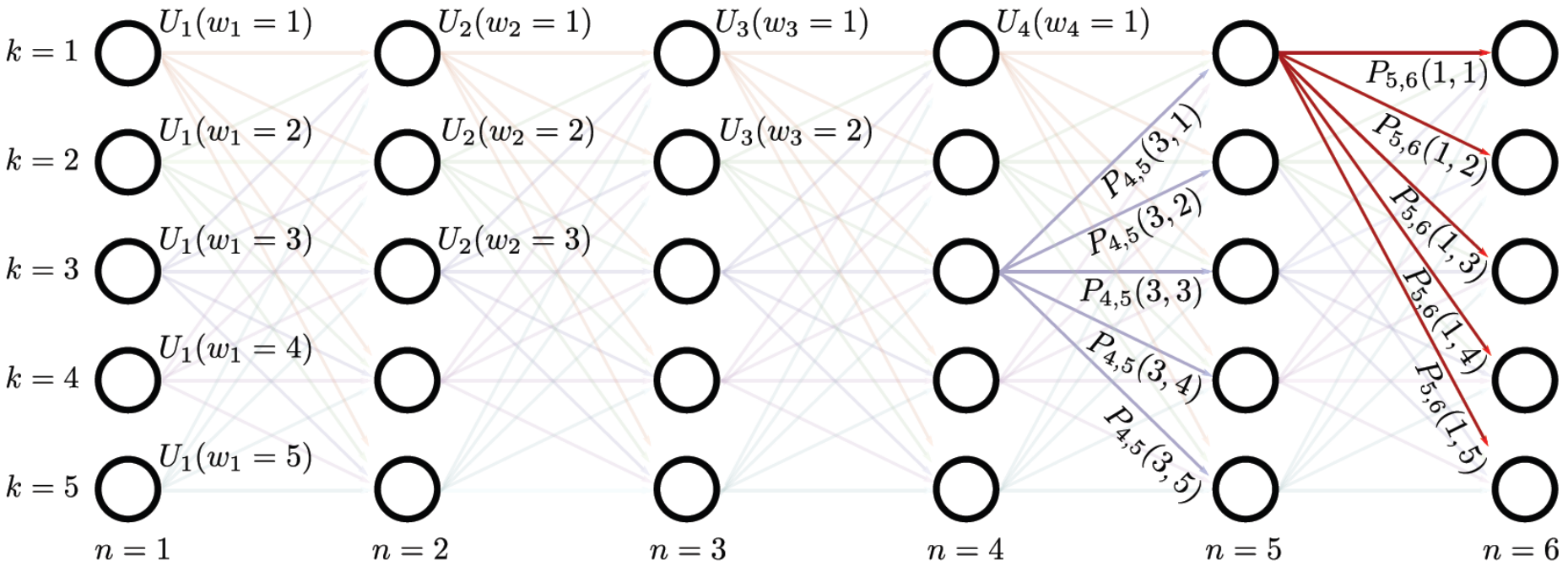


Simon J.D. Prince

Dynamic programming

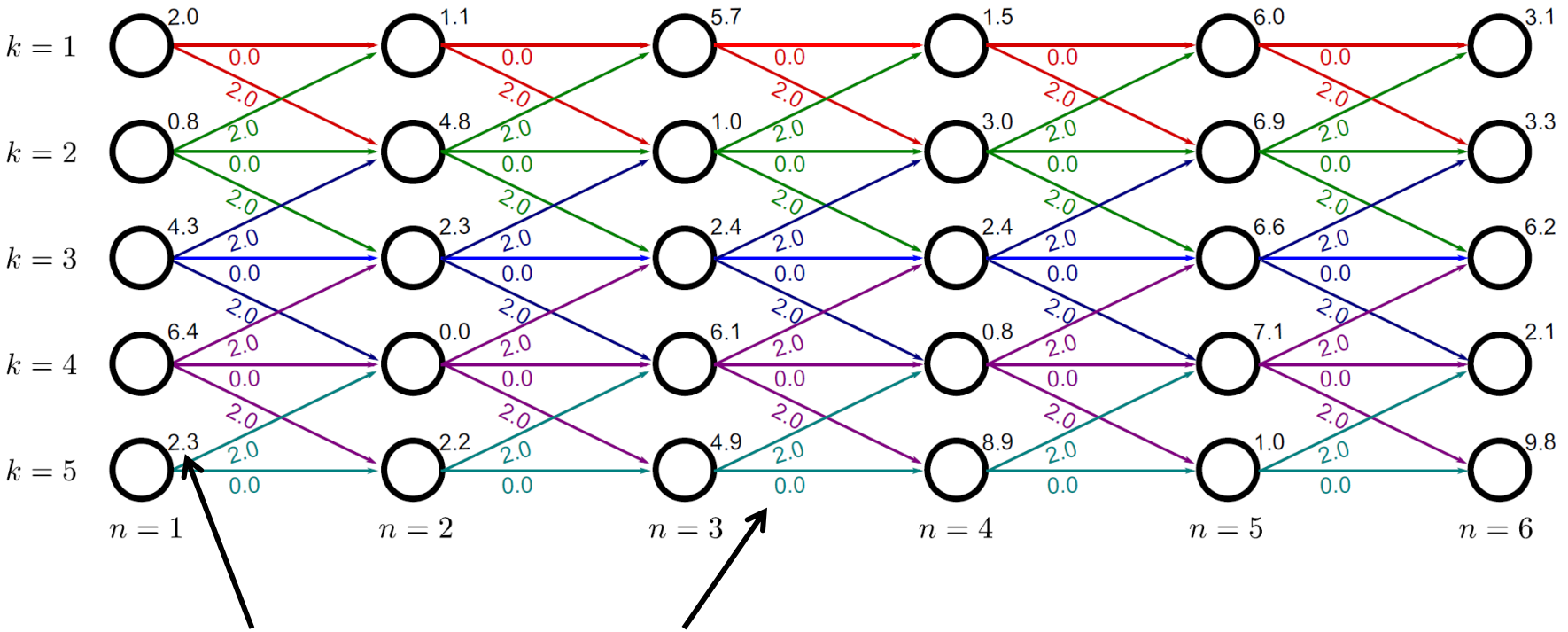
Algorithm:

1. Work through graph computing minimum possible cost $S_{n,k}$ to reach each node
2. When we get to last column, find minimum
3. Trace back to see how we got there



Simon J.D. Prince

Worked example



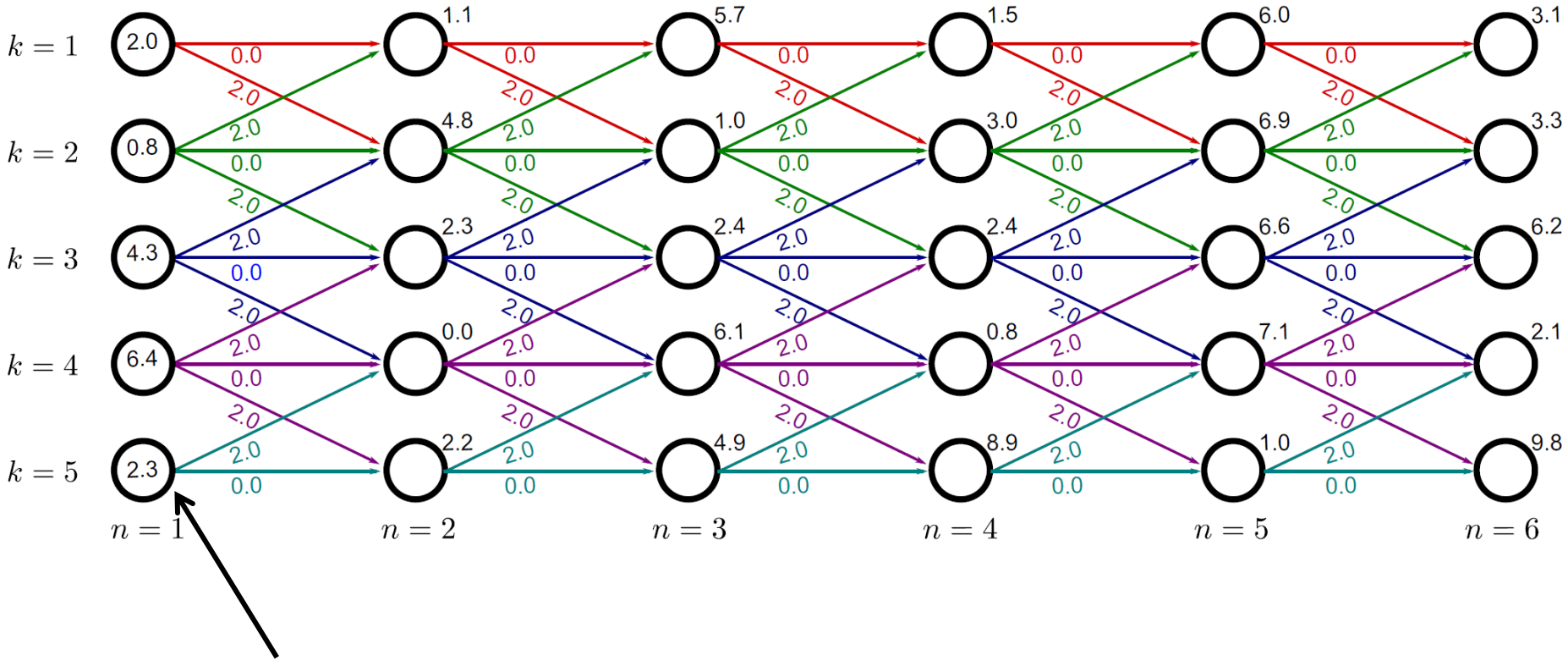
Unary cost

Pairwise costs:

- Zero cost to stay at same label
- Cost of 2 to change label by 1
- Infinite cost for changing by more than one (not shown)

Simon J.D. Prince

Worked example

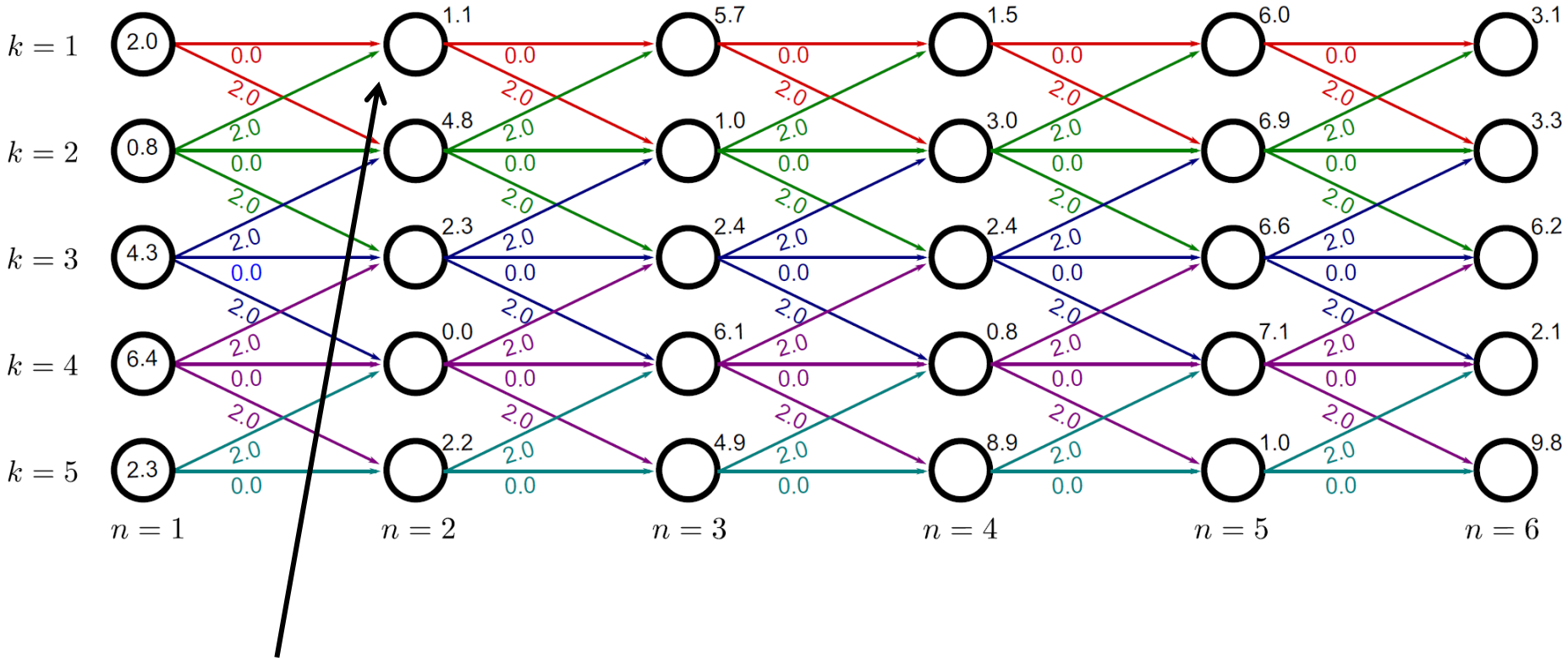


Minimum cost $S_{1,1} \ldots S_{1,5}$
to reach first node is just unary cost

$$S_{1,k} = U_1(w_1 = k)$$

Simon J.D. Prince

Worked example

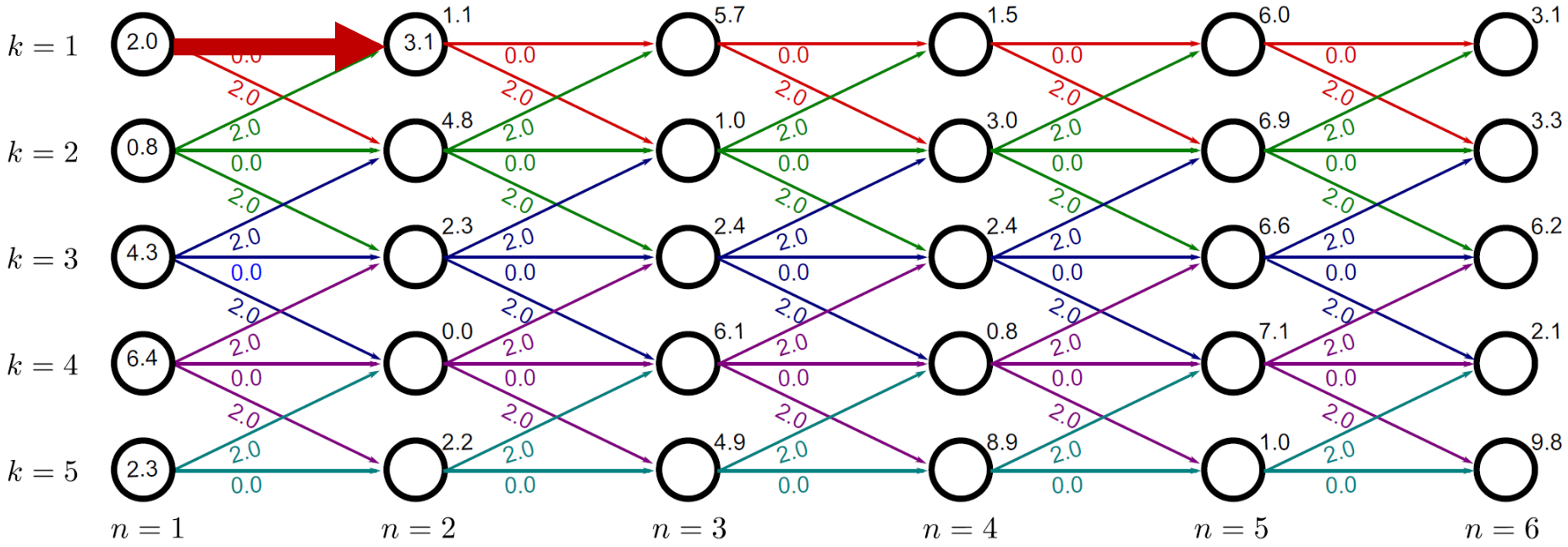


Minimum cost $S_{2,1}$ is minimum of two possible routes to get here

Route 1: 2.0+0.0+1.1 = 3.1
Route 2: 0.8+2.0+1.1 = 3.9

Simon J.D. Prince

Worked example

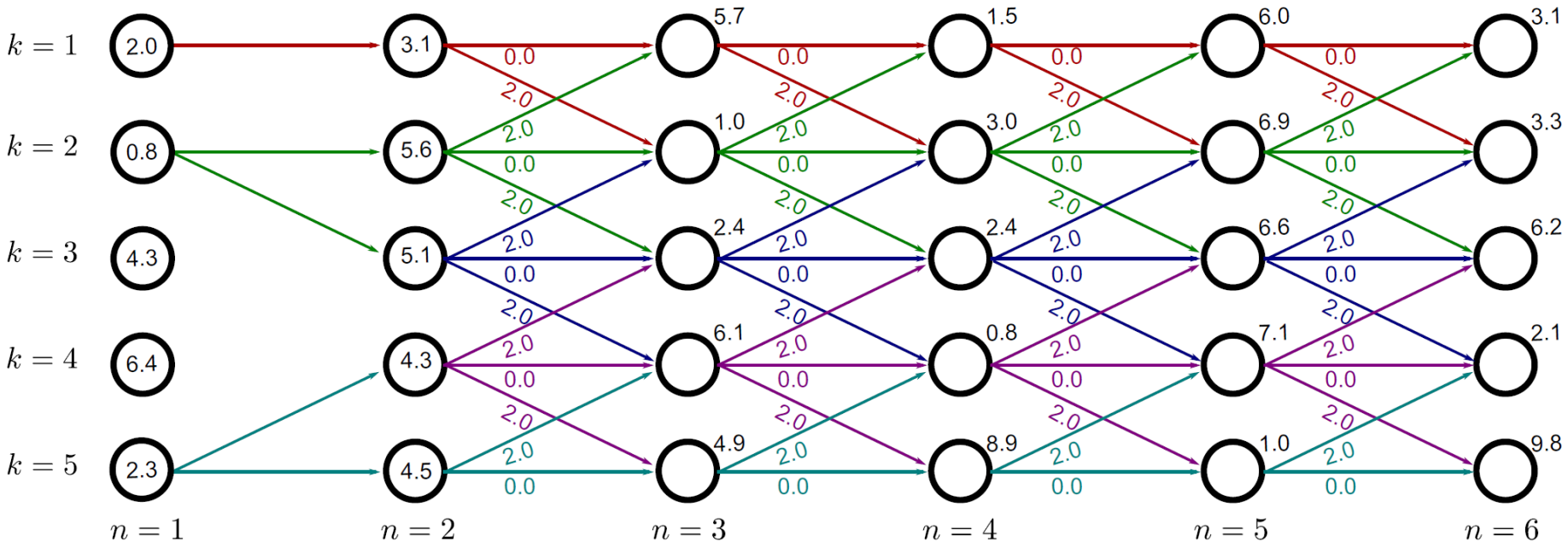


Minimum cost $S_{2,1}$ is minimum of two possible routes to get here

Route 1: 2.0+0.0+1.1 = 3.1
Route 2: 0.8+2.0+1.1 = 3.9

Simon J.D. Prince

Worked example

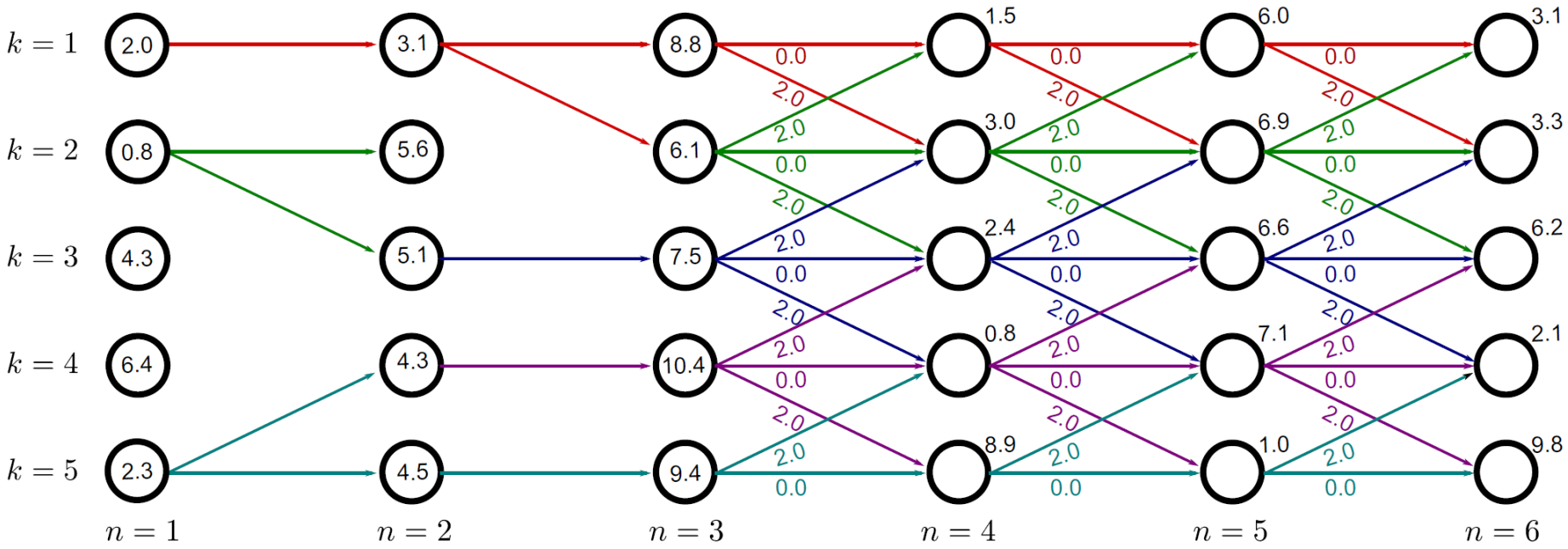


General rule:

$$S_{n,k} = U_n(w_n = k) + \min_l \left[S_{n-1,l} + P_n(w_n = k, w_{n-1} = l) \right]$$

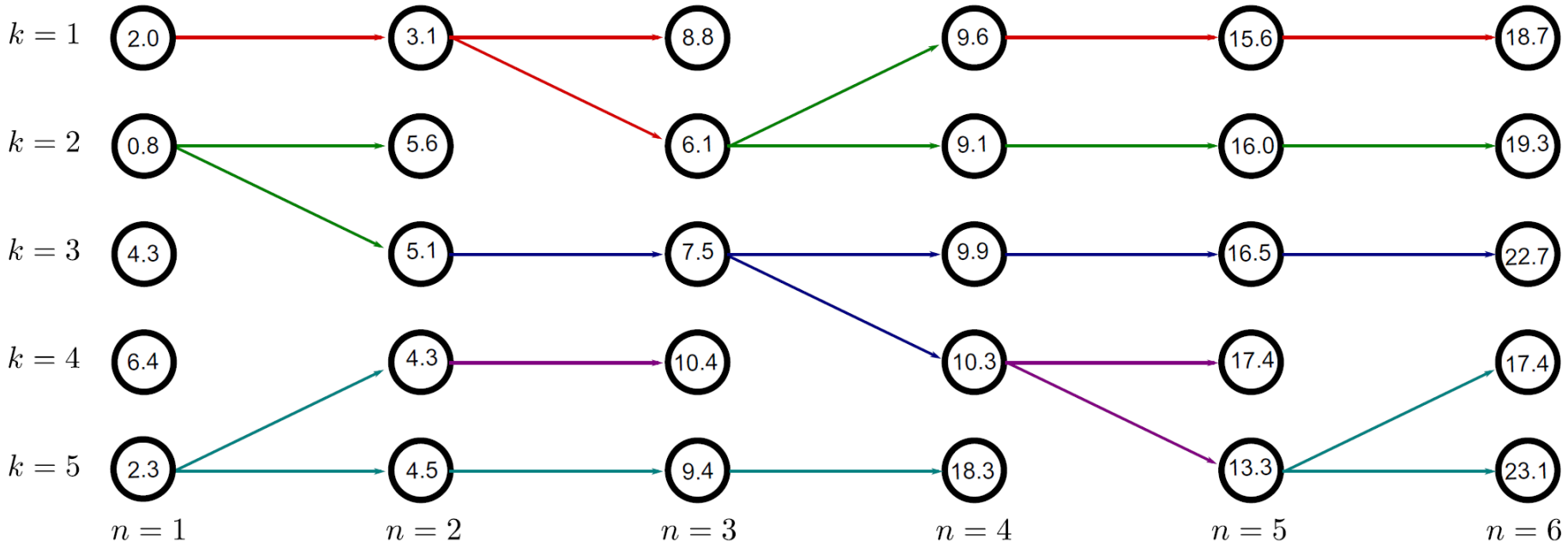
Simon J.D. Prince

Worked example



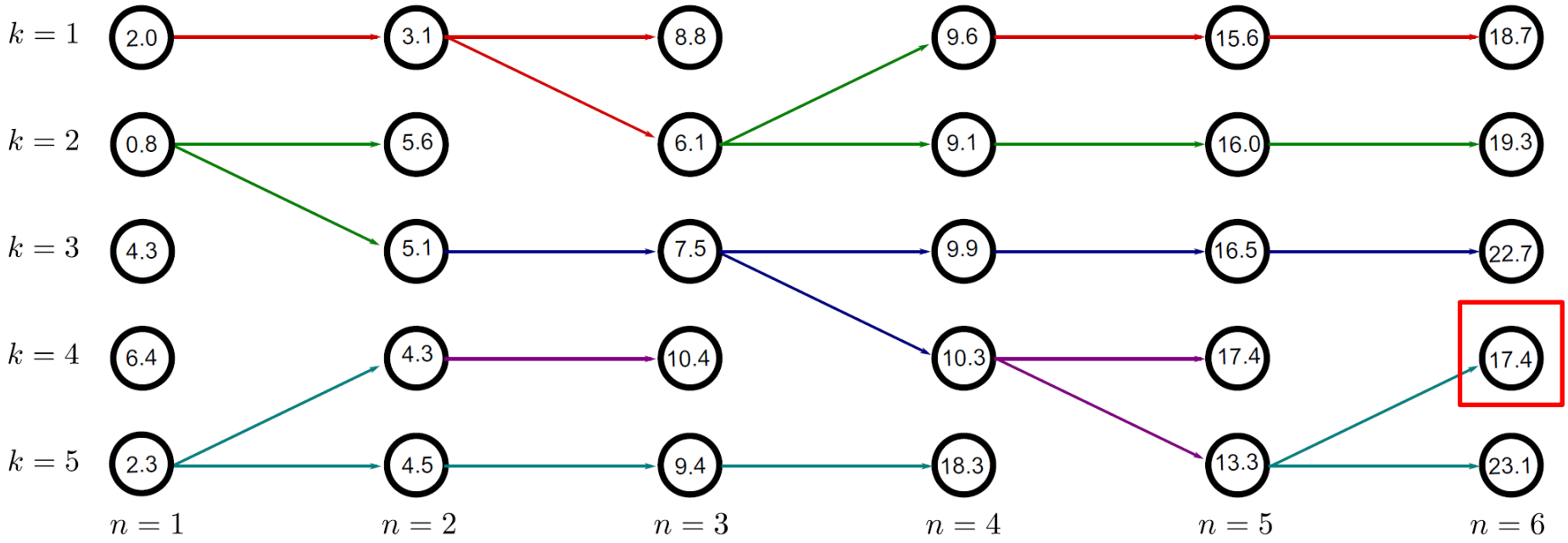
Work through the graph, computing the minimum cost to reach each node

Worked example



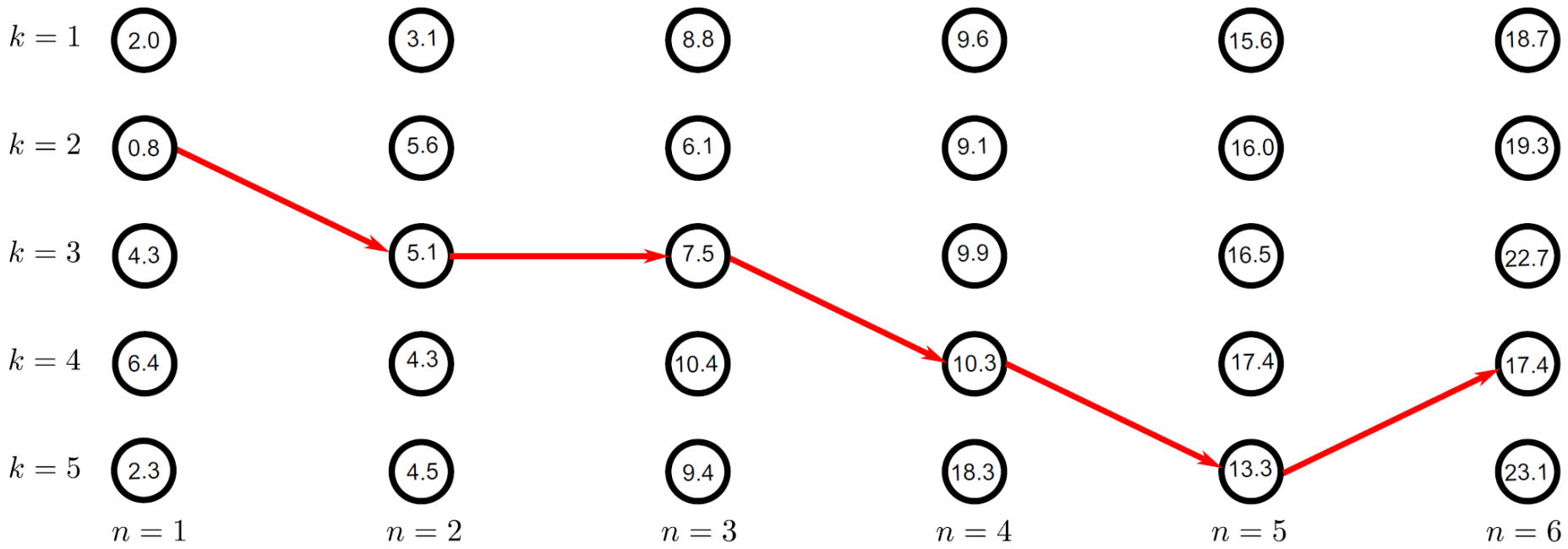
Keep going until we reach the end of the graph

Worked example



Find the minimum possible cost to reach the final column

Worked example



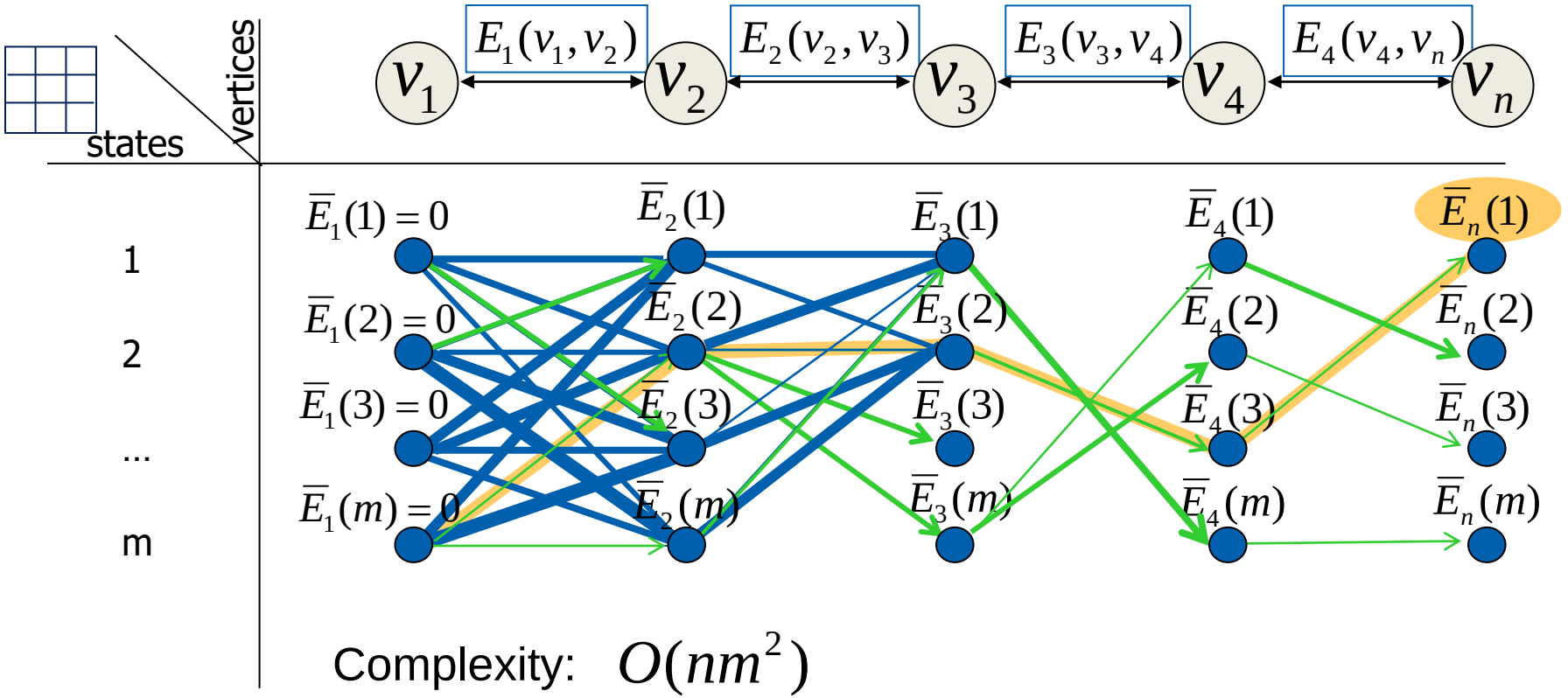
Trace back the route that we arrived here by – this is the minimum configuration

Simon J.D. Prince

Example: 2D Snakes

Main idea: determine optimal position (state) of predecessor, for each possible position of self. Then backtrack from best state for last vertex.

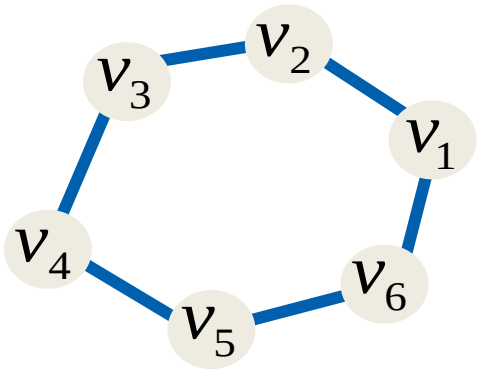
$$E_{total} = E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n) + E_n(v_n)$$

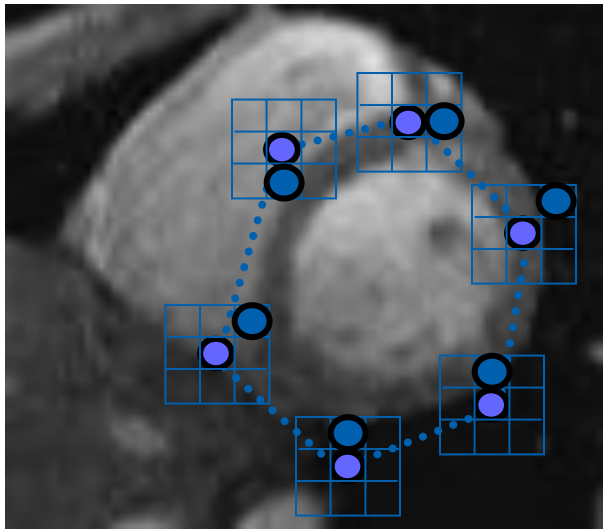


Complexity: $O(nm^2)$

Source: Y. Boykov

Energy minimization: dynamic programming





With this form of the energy function, we can minimize using dynamic programming, with the *Viterbi* algorithm.

Iterate until optimal position for each point is the center of the box, i.e., the snake is optimal in the local search space constrained by boxes.

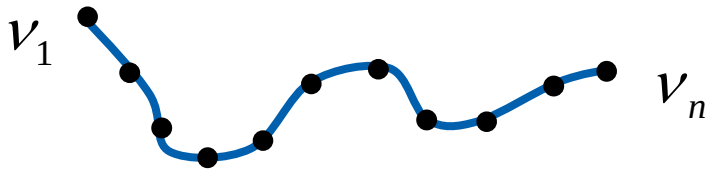
Source: Y. Boykov

Dynamic programming

UNIVERSITÄT BONN

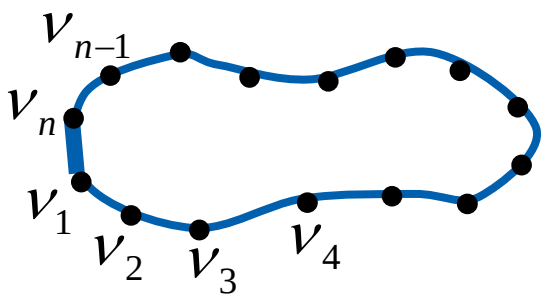
DP can be applied to optimize an open ended snake

$$E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n)$$



For a closed snake, a “loop” is introduced into the total energy.

$$E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n) + E_n(v_n, v_1)$$



Work around:

1) Fix v_1 and solve for rest.
2) Fix an intermediate node at its position found in (1), solve for rest.

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.

Source: T. Brox

Local optimization with gradient descent

UNIVERSITÄT BONN

To minimize the energy functional we can employ the calculus of variation and gradient descent.

The energy functional is of the form:

$$E(C) = \int \mathcal{L}(C, C_s, C_{ss}) ds$$

The corresponding Euler-Lagrange equation is:

$$\frac{dE}{dC} = \frac{d\mathcal{L}}{dC} - \frac{d}{ds}\frac{d\mathcal{L}}{dC_s} + \frac{d^2}{(ds)^2}\frac{d\mathcal{L}}{dC_{ss}}$$

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

$$\frac{dE}{dC} = \frac{d\mathcal{L}}{dC} - \frac{d}{ds}\frac{d\mathcal{L}}{dC_s} + \frac{d^2}{(ds)^2}\frac{d\mathcal{L}}{dC_{ss}} = -\nabla |\nabla I(C)|^2 - \alpha C_{ss} + \beta C_{ssss} = 0$$

Source: T. Brox

Local optimization with gradient descent

UNIVERSITÄT BONN

To minimize the energy functional we can employ the calculus of variation and gradient descent.

The energy functional is of the form:

$$E(C) = \int \mathcal{L}(C, C_s, C_{ss}) ds$$

The corresponding Euler-Lagrange equation is:

$$\frac{dE}{dC} = \frac{d\mathcal{L}}{dC} - \frac{d}{ds}\frac{d\mathcal{L}}{dC_s} + \frac{d^2}{(ds)^2}\frac{d\mathcal{L}}{dC_{ss}} = -\nabla|\nabla I(C)|^2 - \alpha C_{ss} + \beta C_{ssss} = 0$$

The gradient descent is then given by:

$$\frac{\partial C(s,t)}{\partial t} = -\frac{dE}{dC} = \nabla|\nabla I(C)|^2 + \alpha C_{ss} - \beta C_{ssss}$$

Source: T. Brox

Local optimization with gradient descent

UNIVERSITÄT BONN

Discretize curve (u is x- or y coordinate):

$$\frac{\partial C}{\partial t} = \nabla |\nabla I(C)|^2 + \alpha C_{ss} - \beta C_{ssss}$$

$$C_{ss}(s_i) = \frac{u(s_{i-1}) - 2u(s_i) + u(s_{i+1})}{\delta s^2}$$

$$C_{ssss}(s_i) = \frac{u(s_{i-2}) - 4u(s_{i-1}) + 6u(s_i) - 4u(s_{i+1}) + u(s_{i+2})}{\delta s^4}$$

J. Invins and J. Porril. **Everything you always wanted to know about snakes.** AIVRU Technical Memo rev. 2000

Local optimization with gradient descent

UNIVERSITÄT BONN

$$\frac{\partial C}{\partial t} = \nabla |\nabla I(C)|^2 + \alpha C_{ss} - \beta C_{ssss}$$

$$C_{ss}(s_i) = \frac{u(s_{i-1}) - 2u(s_i) + u(s_{i+1})}{\delta s^2}$$

$$C_{ssss}(s_i) = \frac{u(s_{i-2}) - 4u(s_{i-1}) + 6u(s_i) - 4u(s_{i+1}) + u(s_{i+2})}{\delta s^4}$$

Discretize curve (u is x- or y coordinate):

$$\begin{aligned} & u(s_i^{t+1}) - u(s_i^t) \\ & = \frac{\alpha \delta t}{\delta s^2} \left(u(s_{i-1}^{t+1}) - 2u(s_i^{t+1}) + u(s_{i+1}^{t+1}) \right) \\ & - \frac{\beta \delta t}{\delta s^4} \left(u(s_{i-2}^{t+1}) - 4u(s_{i-1}^{t+1}) + 6u(s_i^{t+1}) - 4u(s_{i+1}^{t+1}) + u(s_{i+2}^{t+1}) \right) \\ & + \delta t \nabla |\nabla I(u(s_i^t))|^2 \end{aligned}$$

Local optimization with gradient descent

UNIVERSITÄT BONN

$$
\begin{aligned}
& u(s_i^{t+1}) - u(s_i^t) \\
& \quad = \frac{\alpha\delta t}{\delta s^2}\left(u(s_{i-1}^{t+1}) - 2u(s_i^{t+1}) + u(s_{i+1}^{t+1})\right) \\
& \quad - \frac{\beta\delta t}{\delta s^4}\left(u(s_{i-2}^{t+1}) - 4u(s_{i-1}^{t+1}) + 6u(s_i^{t+1}) - 4u(s_{i+1}^{t+1}) + u(s_{i+2}^{t+1})\right) \\
& \quad + \delta t \nabla |\nabla I(u(s_i^t))|^2
\end{aligned}
$$

Rearrange:

$$
\begin{aligned}
& \frac{\beta\delta t}{\delta s^4}(u(s_{i-2}^{t+1}) + u(s_{i+2}^{t+1})) \\
& + \left(-\frac{\alpha\delta t}{\delta s^2} - \frac{4\beta\delta t}{\delta s^4}\right)(u(s_{i-1}^{t+1}) + u(s_{i+1}^{t+1})) \\
& + \left(1 + \frac{2\alpha\delta t}{\delta s^2} + \frac{6\beta\delta t}{\delta s^4}\right) u(s_i^{t+1}) \\
& \quad = u(s_i^t) + \delta t \nabla |\nabla I(u(s_i^t))|^2
\end{aligned}
$$

Local optimization with gradient descent

UNIVERSITÄT BONN

Rearrange:

Note cycle!

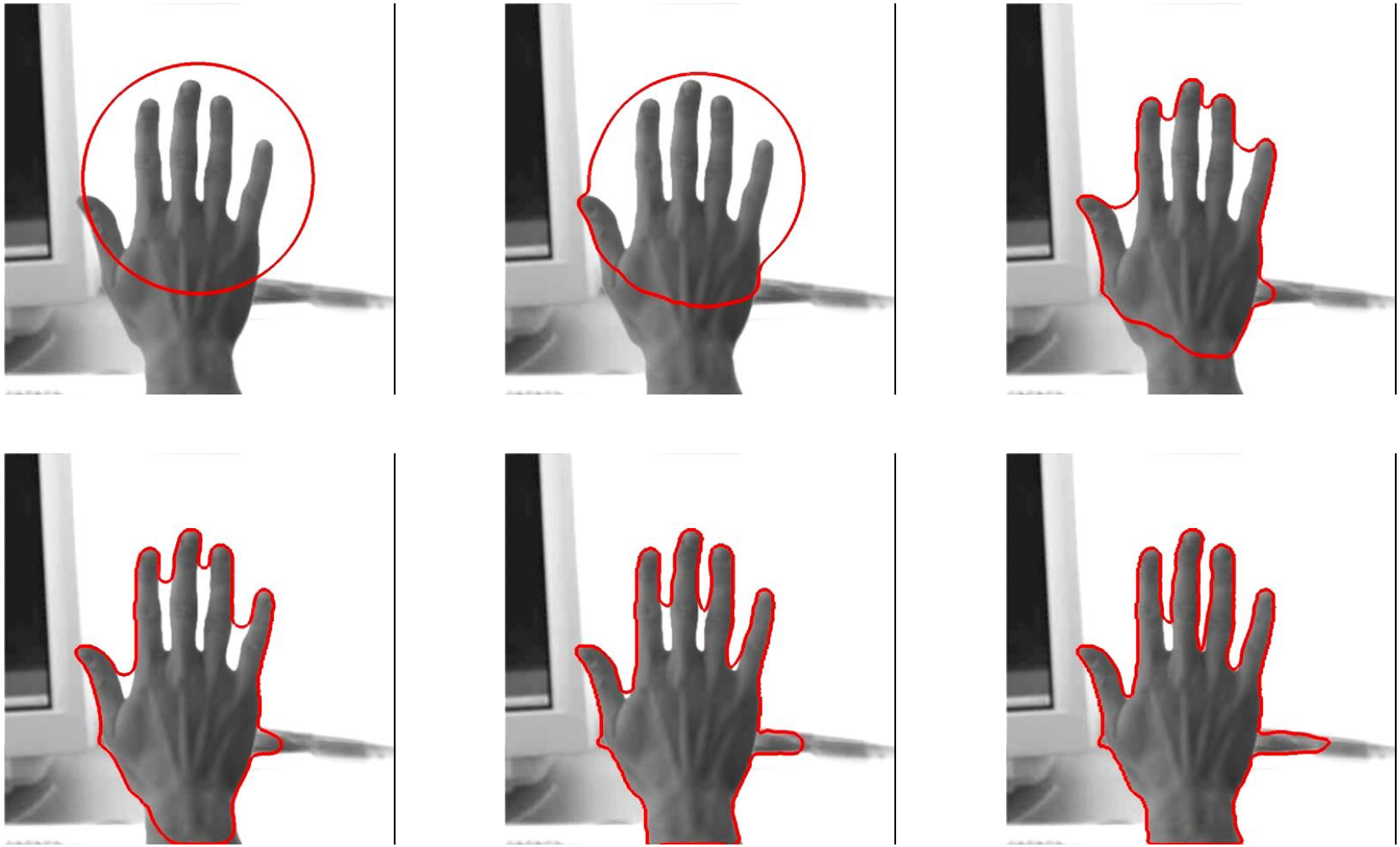
$$\underbrace{\begin{bmatrix} r & q & p & & & p & q \\ q & r & q & p & & & p \\ p & q & r & q & p & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & p & q & r & q & p \\ p & & & p & q & r & q \\ q & p & & & p & q & r \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{pmatrix} \begin{bmatrix} u_0^{t+1} \\ u_1^{t+1} \\ u_2^{t+1} \\ \vdots \\ u_{N-3}^{t+1} \\ u_{N-2}^{t+1} \\ u_{N-1}^{t+1} \end{bmatrix} \end{pmatrix}}_{\mathbf{u}^{t+1}} = \underbrace{\begin{pmatrix} \begin{bmatrix} \tilde{u}_0^{t+1} \\ \tilde{u}_1^{t+1} \\ \tilde{u}_2^{t+1} \\ \vdots \\ \tilde{u}_{N-3}^{t+1} \\ \tilde{u}_{N-2}^{t+1} \\ \tilde{u}_{N-1}^{t+1} \end{bmatrix} \end{pmatrix}}_{\tilde{\mathbf{u}}^{t+1}}$$

$$p \equiv \beta \frac{\delta t}{\delta s^4} \qquad q \equiv -\alpha \frac{\delta t}{\delta s^2} - 4\beta \frac{\delta t}{\delta s^4} \qquad r \equiv 1 + 2\alpha \frac{\delta t}{\delta s^2} + 6\beta \frac{\delta t}{\delta s^4}$$

Compute inverse of M by LU decomposition (only once!)

$$u^{t+1} = M^{-1}(\tilde{u}^{t+1}) = M^{-1}\left(u^t + \delta t \nabla |\nabla I(u^t)|^2\right)$$

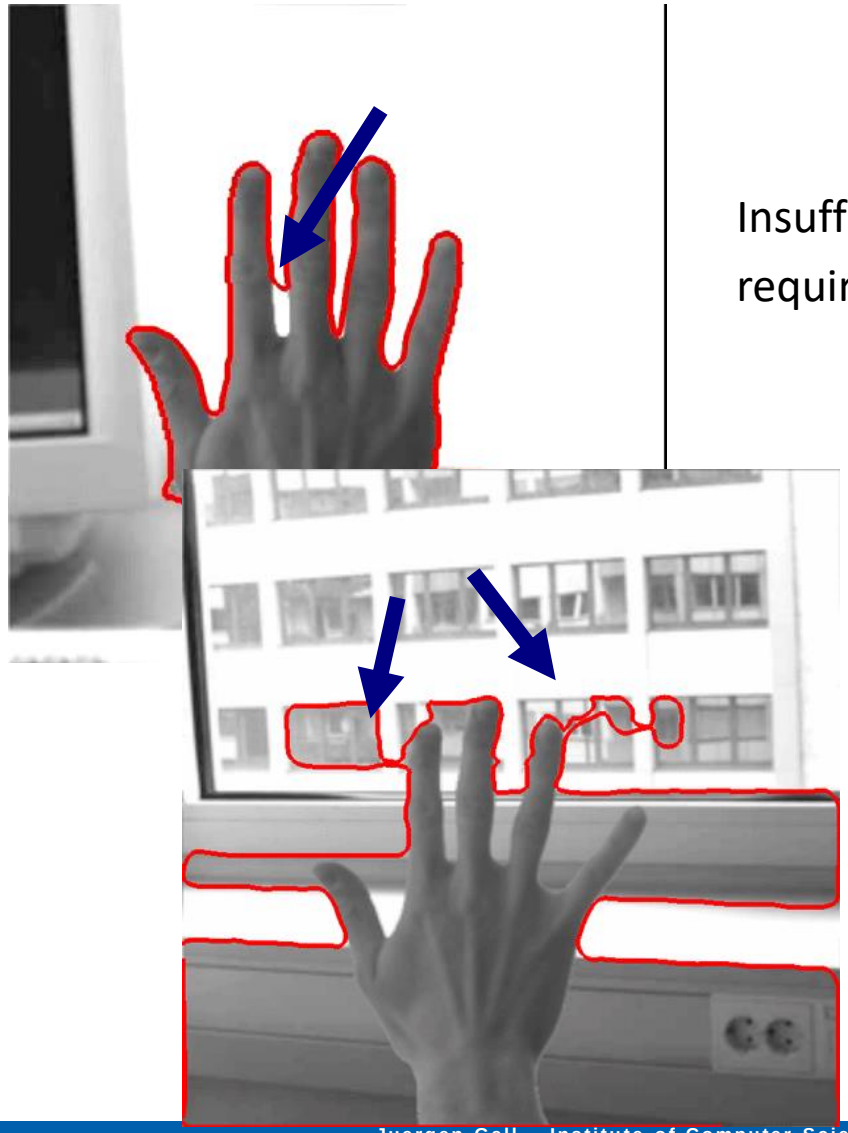
Evolution of Explicit Boundaries



Cremers, Tischhäuser, Weickert, Schnörr, "Diffusion Snakes", IJCV '02

Source: D. Cremers

Evolution of Explicit Boundaries



Insufficient resolution / control point density

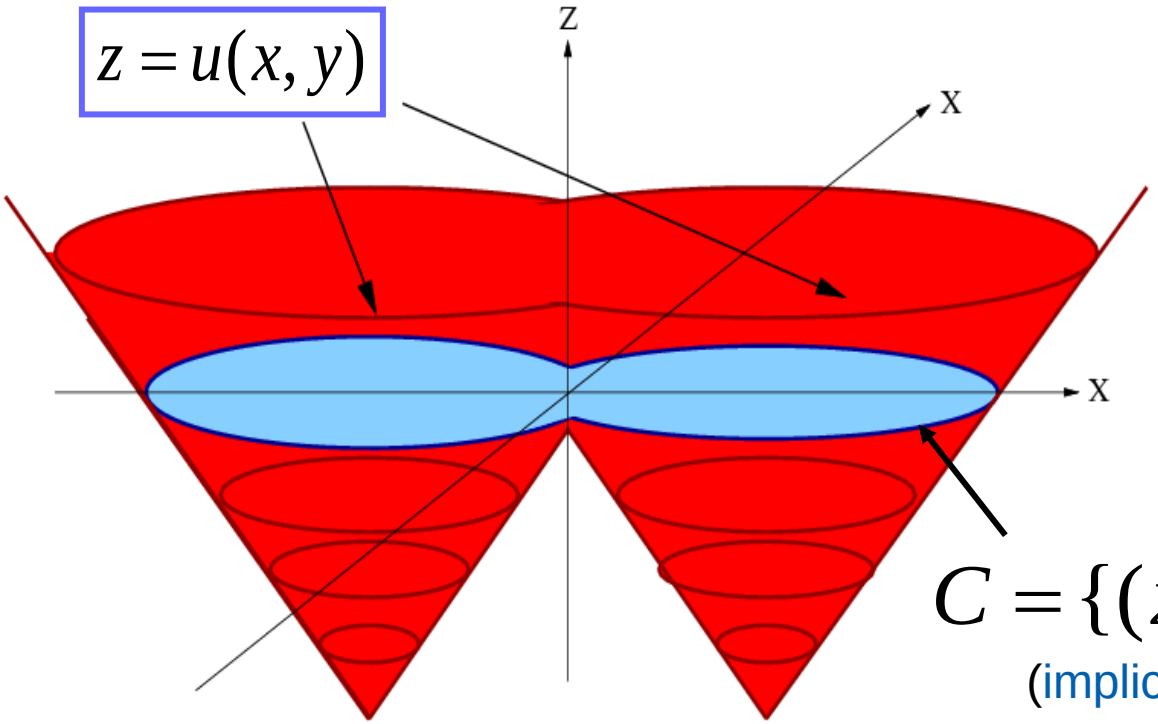
requires control point regridding mechanisms

Fixed topology

requires heuristic splitting mechanisms

Source: D. Cremers

Non-parametric implicit representation of contours via Level Sets



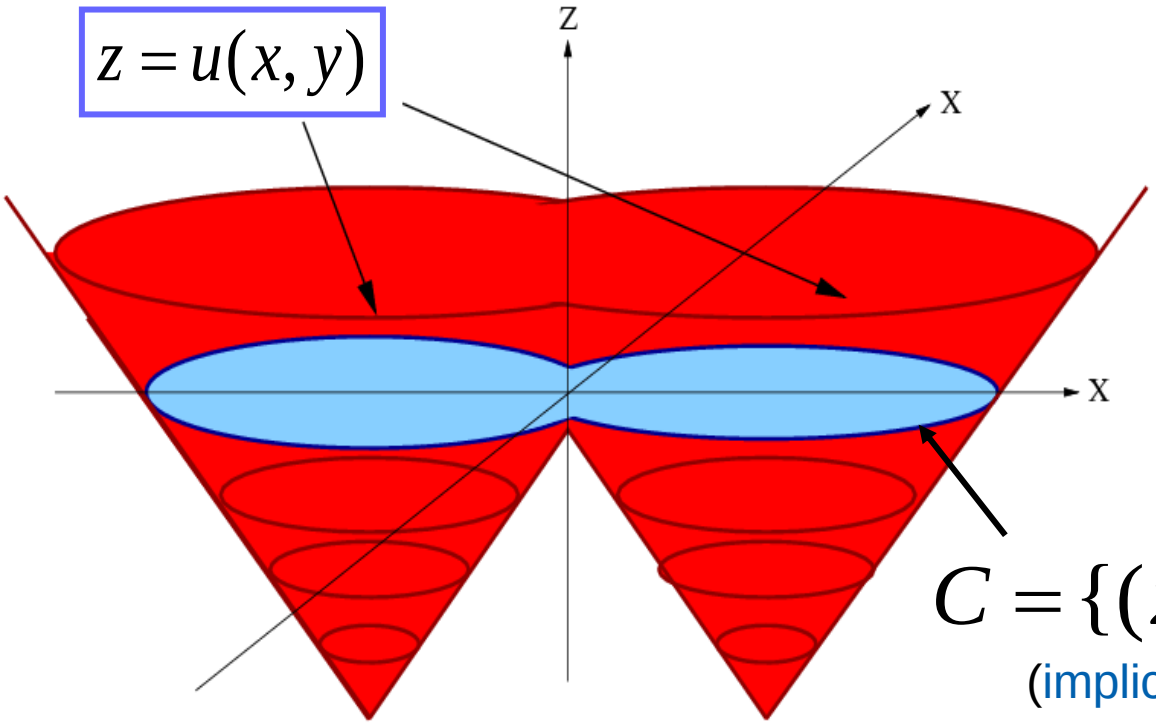
$$C = \{(x, y) : u(x, y) = 0\}$$

(implicit contour representation)

- Let contour *C* be a zero-level set of some function *u(x,y)*
- Function *u(x,y)* is called a *level-set* function

Source: Y. Boykov

Non-parametric implicit representation of contours via Level Sets



$$C = \{(x, y) : u(x, y) = 0\}$$

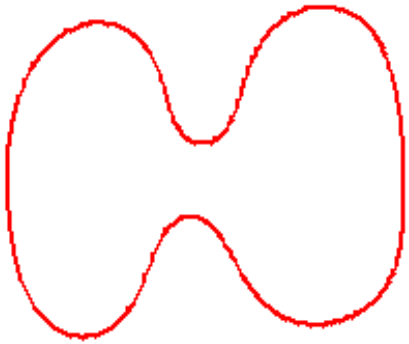
(implicit contour representation)

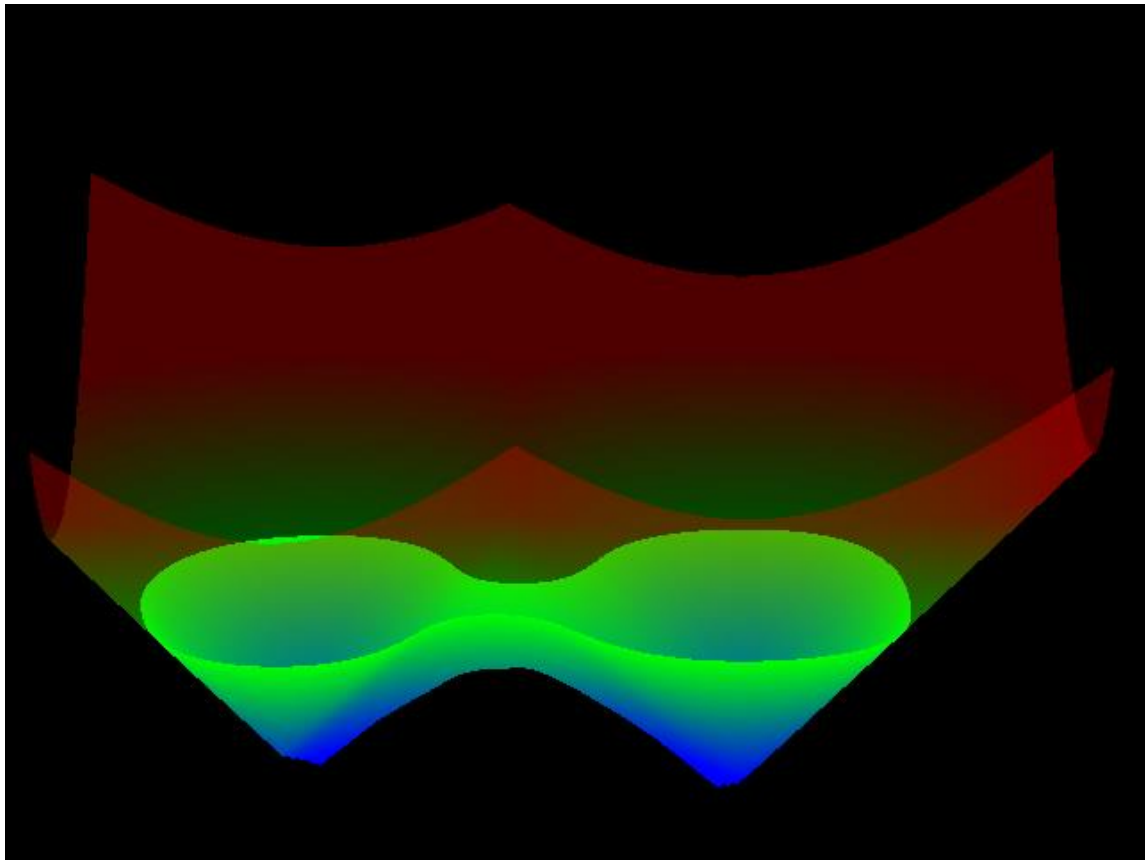
- For a given *u(x,y)* one can get *C* by thresholding
  Contour interior are points with negative values of *u(x,y)*
- How to get some level set function *u(x,y)* for a given *C* ?

  Signed distance map: $u(x, y) = \pm \mathrm{dist}\big((x, y), C\big)$

Source: Y. Boykov

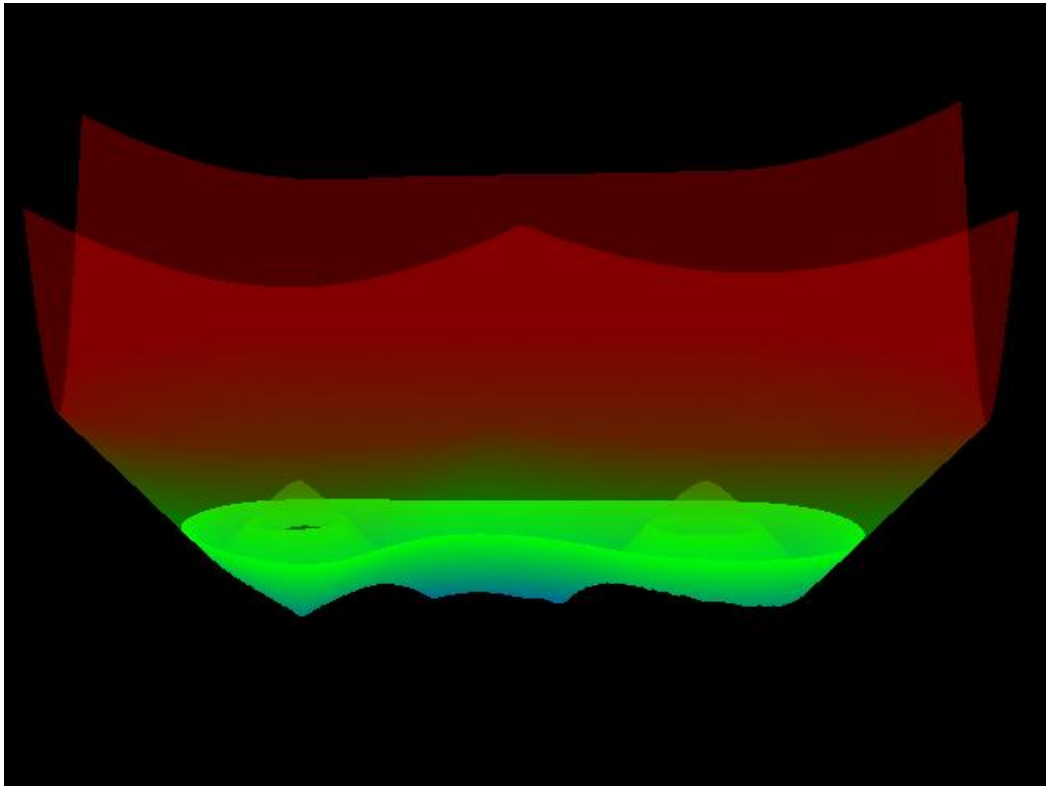
Example of distance map





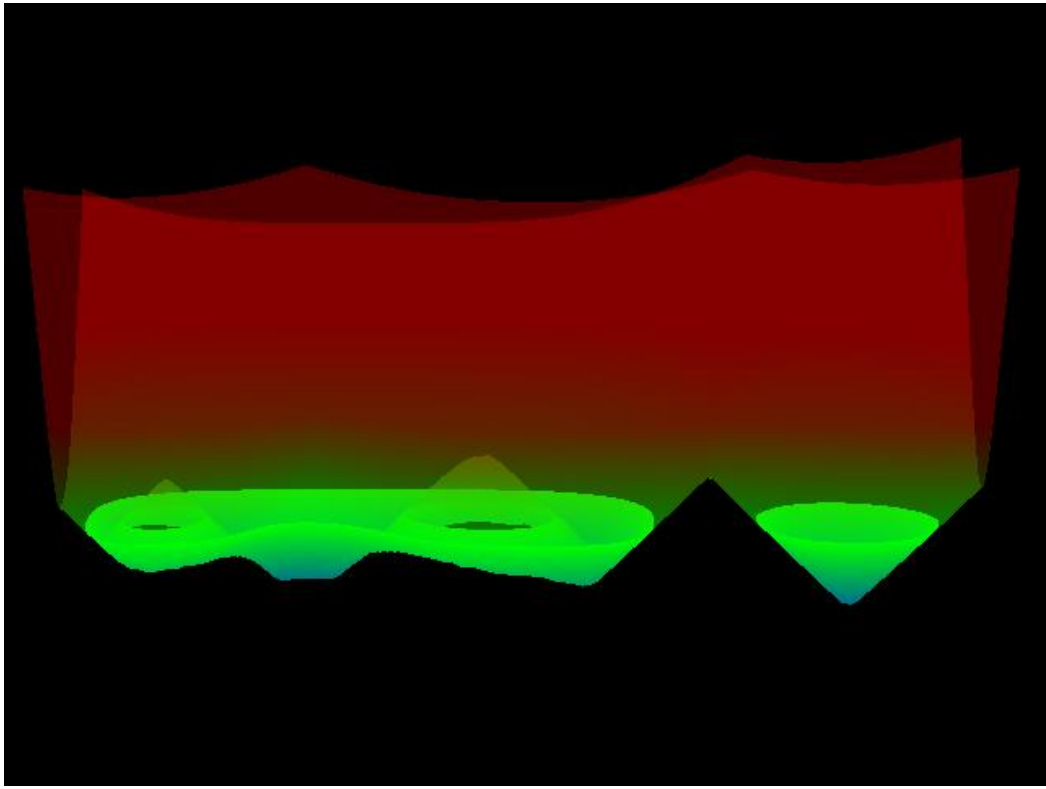
Source: O. Juan

Holes



Source: O. Juan

Isolated parts



Source: O. Juan

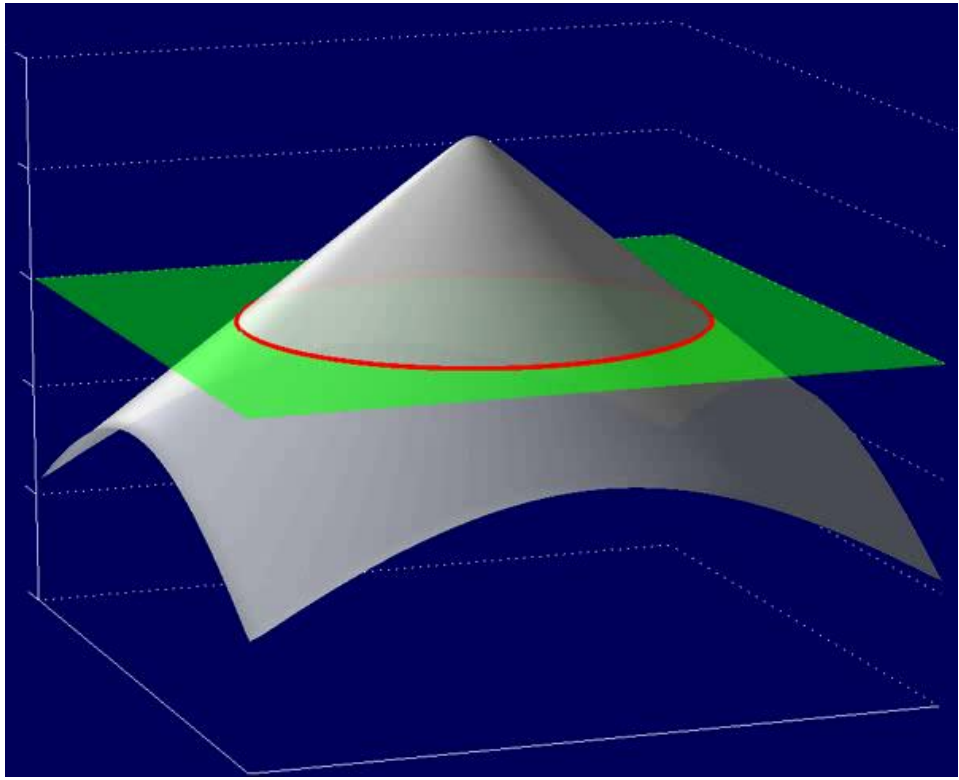
Level set representation of contours

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 re-gridding necessary
- Allows for topological changes
- Can be applied in any dimension
- Represents the contour *and* the enclosed region



Source: D. Cremers

The level set method

UNIVERSITÄT BONN

At all times the contour is the zero-level set: $\phi\Big(C(t),t\Big) = 0 \quad \forall\, t.$

Then the total time derivative of $\phi\Big(C(t),t\Big)$ must vanish:

$$\frac{d}{dt}\phi\Big(C(t),t\Big) = \nabla\phi^\top \frac{\partial C}{\partial t} + \frac{\partial \phi}{\partial t} = 0$$

Yields an evolution equation for $\phi\Big(C(t),t\Big)$: $\frac{\partial \phi}{\partial t} = -\nabla\phi^\top \frac{\partial C}{\partial t}$

Note that this equation is only valid for $\phi = 0$, i.e. at C

S. Osher and R. Fedkiw. Level set methods: an overview and some recent results. Journal of Computational Physics 2000.

Source: T. Brox

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:

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

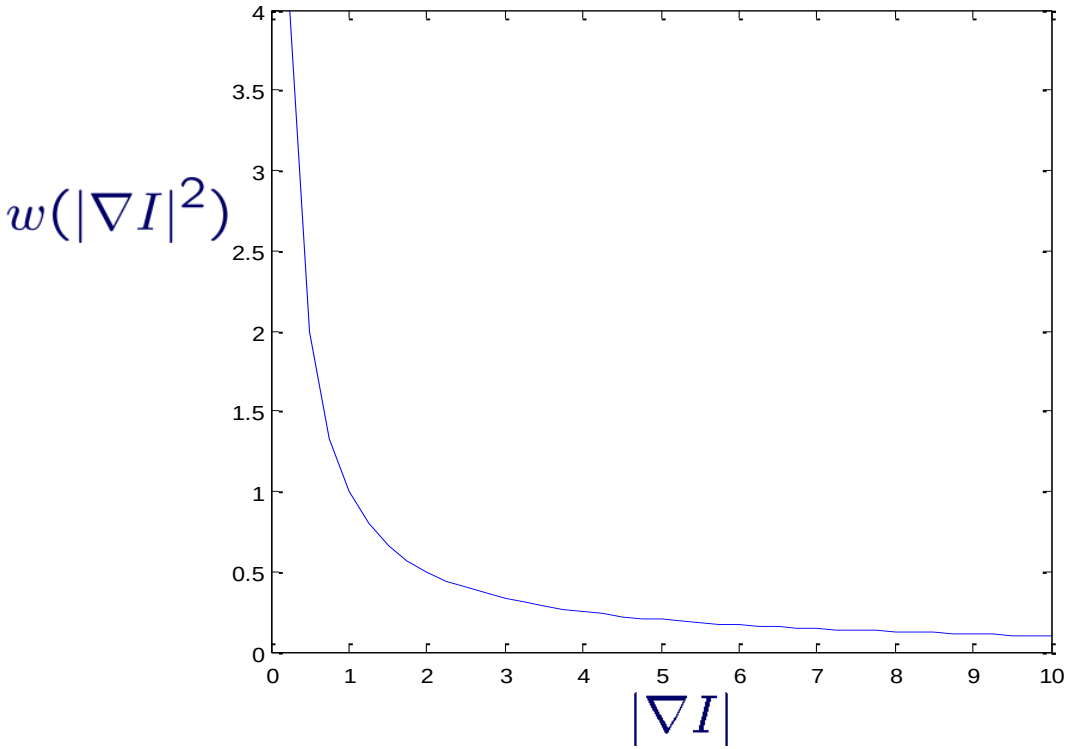
→ Combines the internal and external energy of the Snake model

Source: T. Brox

Geodesic active contours

The metric in geodesic active contours is given by any decreasing function of the image gradient.

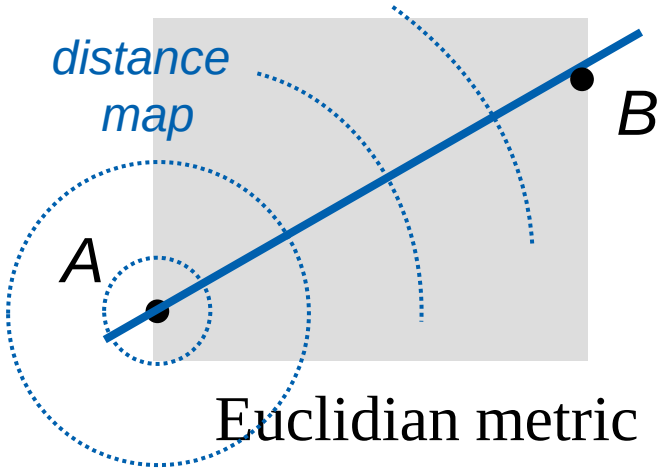
For example: $w(|\nabla I|^2) = \frac{1}{\sqrt{|\nabla I|^2 + \epsilon^2}}, \quad \epsilon = 1$



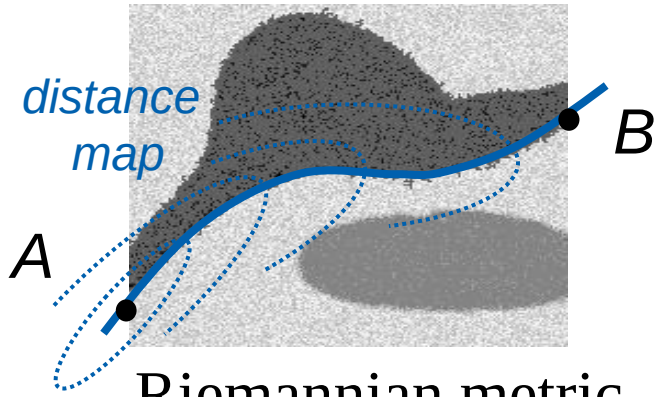
Source: T. Brox

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

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.

Source: T. Brox

Gradient descent

Like the snakes model, geodesic active contours can be minimized via gradient descent.

The Euler-Lagrange equations are:

$$w\kappa\mathbf{n} + (\nabla w^\top \mathbf{n})\mathbf{n} = 0$$

κ curvature
$\mathbf{n}$ the outer normal vector.

The gradient descent starting from an initial contour $C(s, t = 0) = C_0(s)$ is then described by:

$$\partial_t C(s,t) = -\frac{dE(C)}{dC} = -w\kappa\mathbf{n} - (\nabla w^\top \mathbf{n})\mathbf{n}$$

Caselles et al. Geodesic Active Contours. IJCV 1997

Source: T. Brox

Recall: The level set method

UNIVERSITÄT BONN

At all times the contour is the zero-level set: $\phi\big(C(t),t\big) = 0 \quad \forall\, t.$

Then the total time derivative of $\phi\big(C(t),t\big)$ must vanish:

$$\frac{d}{dt}\phi\big(C(t),t\big) = \nabla\phi^\top \frac{\partial C}{\partial t} + \frac{\partial \phi}{\partial t} = 0$$

Yields an evolution equation for $\phi\big(C(t),t\big)$: $\frac{\partial \phi}{\partial t} = -\nabla\phi^\top \frac{\partial C}{\partial t}$

Note that this equation is only valid for $\phi = 0$, i.e. at C

S. Osher and R. Fedkiw. Level set methods: an overview and some recent results. Journal of Computational Physics 2000.

Source: T. Brox

Applying the level set method

- We can use the level set method to implement the curve evolution.

- Remember that the level set method translates the evolution of a curve of the form

$$\partial_t C(s,t) = -\frac{dE(C)}{dC} = -w\kappa\mathbf{n} - (\nabla w^\top \mathbf{n})\mathbf{n}$$

into an evolution of the embedding function:

$$\frac{\partial \phi}{\partial t} = -\nabla \phi^\top \frac{\partial C}{\partial t} \qquad \mathbf{n} = \frac{\nabla \phi}{|\nabla \phi|}$$

- The local curvature of level lines in the embedding function is given as

$$\kappa = \mathrm{div}\mathbf{n} = \mathrm{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right)$$

- So we obtain:

$$\partial_t \phi = w|\nabla \phi|\mathrm{div}\left(\frac{\nabla \phi}{|\nabla \phi|}\right) + (\nabla w^\top \nabla \phi)$$

Source: T. Brox

Implementation

Geodesic active contour:

$$\partial_t \phi = w|\nabla\phi| \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) + \nabla w^\top \nabla\phi$$

Reformulation

$$|\nabla\phi| \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) = |\nabla\phi| \left(\frac{\partial}{\partial x}\frac{\phi_x}{\sqrt{\phi_x^2+\phi_y^2}} + \frac{\partial}{\partial y}\frac{\phi_y}{\sqrt{\phi_x^2+\phi_y^2}}\right)$$

$$= |\nabla\phi| \left(\frac{\phi_{xx}\sqrt{\phi_x^2+\phi_y^2} - \phi_x\frac{2\phi_x\phi_{xx}+2\phi_y\phi_{xy}}{2\sqrt{\phi_x^2+\phi_y^2}}}{\phi_x^2+\phi_y^2} + \frac{\phi_{yy}\sqrt{\phi_x^2+\phi_y^2} - \phi_y\frac{2\phi_y\phi_{yy}+2\phi_x\phi_{xy}}{2\sqrt{\phi_x^2+\phi_y^2}}}{\phi_x^2+\phi_y^2}\right)$$

$$= \frac{\phi_{xx}(\cancel{\phi_x^2}+\phi_y^2) - \cancel{\phi_x^2\phi_{xx}} - \phi_x\phi_y\phi_{xy} + \phi_{yy}(\phi_x^2+\cancel{\phi_y^2}) - \cancel{\phi_y^2\phi_{yy}} - \phi_x\phi_y\phi_{xy}}{\phi_x^2+\phi_y^2}$$

$$= \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{\phi_x^2+\phi_y^2}$$

Source: T. Brox

Implementation

Geodesic active contour:

$$\partial_t \phi = w|\nabla\phi| \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) + \nabla w^\top \nabla\phi$$

Mean curvature:

$$|\nabla\phi| \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) = \frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{\phi_x^2 + \phi_y^2}$$

$$w(|\nabla I|^2) = \frac{1}{\sqrt{|\nabla I|^2 + \epsilon^2}}, \qquad \epsilon = 1$$

Discretization of mean curvature motion

UNIVERSITÄT BONN

- Task: Spatially discretize this equation

$$\frac{\phi_{xx}\phi_y^2 - 2\phi_x\phi_y\phi_{xy} + \phi_{yy}\phi_x^2}{\phi_x^2 + \phi_y^2}$$

i-1,j-1	i,j-1	i+1,j-1
i-1,j	i,j	i+1,j
i-1,j+1	i,j+1	i+1,j+1

- First derivatives: central differences

$$\phi_x \approx \frac{1}{2}(\phi_{i+1,j} - \phi_{i-1,j}) \qquad \phi_y \approx \frac{1}{2}(\phi_{i,j+1} - \phi_{i,j-1})$$

- Second derivatives:

$$\phi_{xx} \approx (\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}) \qquad \phi_{yy} \approx (\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1})$$

$$\phi_{xy} \approx \frac{1}{4}(\phi_{i+1,j+1} - \phi_{i+1,j-1} - \phi_{i-1,j+1} + \phi_{i-1,j-1})$$

Source: T. Brox

Geodesic active contours: explicit scheme

UNIVERSITÄT BONN

Gradient descent according to the first term leads to the following explicit scheme (mean curvature motion weighted by w):

$$\phi^{k+1} = \phi^k + \tau w \frac{\phi^k_{xx}(\phi^k)^2_y - 2\phi^k_x \phi^k_y \phi^k_{xy} + \phi^k_{yy}(\phi^k)^2_x}{(\phi^k)^2_x + (\phi^k)^2_y + \epsilon}$$

It is empirically stable for $\tau \leq \frac{1}{4 \max w}, \epsilon = 10^{-4}$

We need rather small time steps, i.e., many iterations are necessary.

Source: T. Brox

Implementation

Geodesic active contour:

$$\partial_t \phi = w|\nabla\phi| \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) + \nabla w^\top \nabla\phi$$

$$\phi^{k+1} = \phi^k + \tau w \frac{\phi^k_{xx}(\phi^k)^2_y - 2\phi^k_x\phi^k_y\phi^k_{xy} + \phi^k_{yy}(\phi^k)^2_x}{(\phi^k)^2_x + (\phi^k)^2_y + \epsilon} + X$$

Implementation of the front propagation term

UNIVERSITÄT BONN

Aside from the mean curvature term, we have to implement the term that propagates the curve towards edges.

$$\nabla w^\top \nabla \phi$$

Front propagation can be implemented with the upwind scheme.

Approximate ∇w with central differences (standard).

Approximate $\nabla \phi$ with one-sided differences.

Choose the uphill direction:

$$\begin{aligned} &\max(w_x, 0)(\phi_{x+1,y} - \phi_{x,y}) + \min(w_x, 0)(\phi_{x,y} - \phi_{x-1,y}) \\ +\, &\max(w_y, 0)(\phi_{x,y+1} - \phi_{x,y}) + \min(w_y, 0)(\phi_{x,y} - \phi_{x,y-1}) \end{aligned}$$

The corresponding explicit scheme is stable for time step sizes $\tau \leq \frac{1}{2}$.

Source: T. Brox

Implementation

Geodesic active contour:

$$\partial_t \phi = w|\nabla\phi| \operatorname{div}\left(\frac{\nabla\phi}{|\nabla\phi|}\right) + \nabla w^\top \nabla\phi$$

$$\phi^{k+1} = \phi^k + \tau w \frac{\phi^k_{xx}(\phi^k)^2_y - 2\phi^k_x\phi^k_y\phi^k_{xy} + \phi^k_{yy}(\phi^k)^2_x}{(\phi^k)^2_x + (\phi^k)^2_y + \epsilon}$$

$$+\max(w_x,0)(\phi^k_{x+1,y}-\phi^k_{x,y})+\min(w_x,0)(\phi^k_{x,y}-\phi^k_{x-1,y})$$
$$+\max(w_y,0)(\phi^k_{x,y+1}-\phi^k_{x,y})+\min(w_y,0)(\phi^k_{x,y}-\phi^k_{x,y-1})$$

Geodesic active contours: an example

R. Goldenberg et al. **Fast Geodesic Active Contours.** TIP 2001

Summary

- The snake model seeks contours that go through gradient maxima while at the same time penalizing the length and curvature of the contour.

- The geodesic active contour model seeks a curve of minimum geodesic length, where the metric is again given by an edge indicator.

- The models can be locally minimized by gradient descent.

- The contour can be represented either explicitly by control points, or implicitly by an embedding function (level set function).

Source: T. Brox

UNIVERSITÄT BONN