

Model-based segmentation

- Today we look at methods that search for image features with certain characteristics, e.g.
 - a closed, smooth contour
 - a hand
 - a face
- That is, they segment an object from the image
- These methods all have in common
 - an initial guess (sometimes any random initialization)
 - an iterative process that modifies the initial guess
 - a final, stable shape that the iterative process converges to
 - this stable point is a (local) minimum of an energy functional (implicit or explicitly defined energy functional)

Model-based segmentation

- Active Contour Models (= snakes, Deformable Contours)
 - in 3D: Active/Deformable Surface Models
 - adapts to boundaries in the image
- Active Shape Models (= Point Distribution Models)
 - creates a model using various examples
 - describes the shape and its variability
- Active Appearance Models
 - ASM including grey values within the shape
- Level Set Method
 - curve evolution applied to contour detection
 - active contours that can split apart and merge together
- Book: sections 7.2, 7.3, 10.3 and 10.4

Active contour models

- Kass, Witkin & Terzopoulos (1988)
- In 2D usually called snakes
- A flexible line or closed contour that evolves over time to some minimal energy configuration
 - hopefully matching object boundaries in the image!
- Used for:
 - automatic segmentation of objects
 - interactive delineation of objects
 - tracking objects over time in video

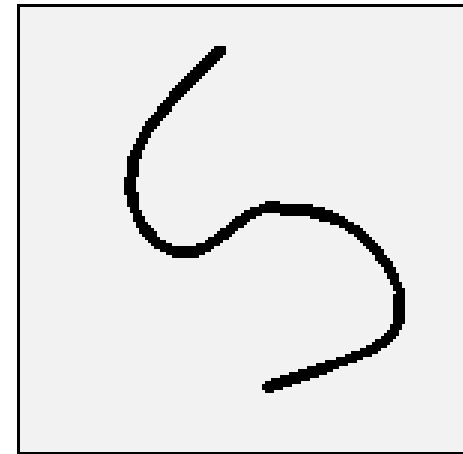
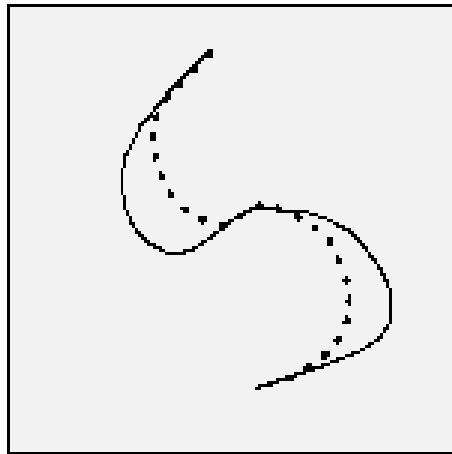
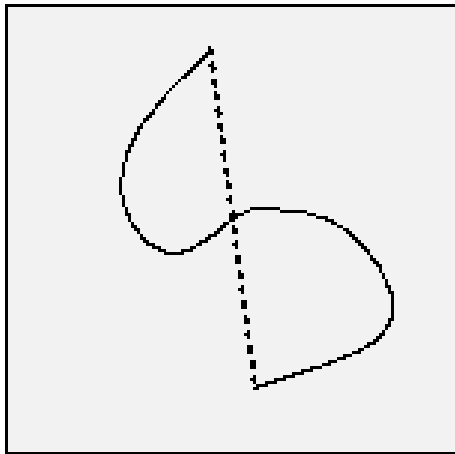
Snakes

- A flexible line:

$$\vec{v}(s) = (x(s), y(s))^T \quad s \in [0, 1]$$

- A closed contour:

$$\vec{v}(0) = \vec{v}(1)$$



Snakes

- Modify the snake to find a local minimum in the energy functional

$$E = \int_0^1 E_{\text{int}}(\vec{v}(s)) + E_{\text{ext}}(\vec{v}(s)) ds$$

$$E_{\text{int}}(\vec{v}(s)) = \frac{1}{2} \left\{ \underbrace{\alpha(s) |\vec{v}'(s)|^2}_{\text{"membrane" term}} + \underbrace{\beta(s) |\vec{v}''(s)|^2}_{\text{"thin plate" term}} \right\} \quad (\alpha \text{ and } \beta \text{ often constant})$$

$$E_{\text{ext}}(\vec{v}(s)) = \begin{cases} G \otimes I & \text{Gaussian smoothed input image} \\ -|\nabla I|^2 & (\nabla I = \nabla G \otimes I) \\ \frac{\partial \theta}{\partial \vec{n}_{\perp}} & \text{derivative of gradient direction perpendicular to gradient} \end{cases}$$

Snakes

- Minimizing this:

$$E = \int_0^1 \frac{1}{2} \{ \alpha |\vec{v}'(s)|^2 + \beta |\vec{v}''(s)|^2 \} + E_{\text{ext}}(\vec{v}(s)) ds$$

- through Euler-Lagrange equation yields this:

$$\underbrace{\alpha \vec{v}''(s) - \beta \vec{v}''''(s)}_{\vec{F}_{\text{int}}} - \underbrace{\nabla E_{\text{ext}}(\vec{v}(s))}_{\vec{F}_{\text{ext}}} = 0$$

- We solve using gradient descent:

$$\frac{\partial \vec{v}(s, t)}{\partial t} = \alpha \vec{v}''(s, t) - \beta \vec{v}''''(s, t) - \nabla E_{\text{ext}}(\vec{v}(s, t))$$

Snakes, discrete

- Discretizing the gradient gradient descent equation

$$\frac{\partial \vec{V}(s, t)}{\partial t} = \alpha \vec{V}''(s, t) - \beta \vec{V}'''(s, t) - \nabla E_{\text{ext}}(\vec{V}(s, t))$$

$$\frac{\partial X_t}{\partial t} = A X_t + f_x(X_t, Y_t)$$

$$\frac{X_t - X_{t-1}}{\gamma} = A X_t + f_x(X_t, Y_t)$$

 step size

$$X_t = (I - \gamma A)^{-1} \{ X_{t-1} + \gamma f_x(X_t, Y_t) \}$$

(we assume $f_x(X_t, Y_t) \approx f_x(X_{t-1}, Y_{t-1})$)

X_t is a vector with x-values of coordinates of points along the curve; Y_t is the y-values

$$f_x = \frac{\partial}{\partial X} E_{\text{ext}}$$

A is a pentadiagonal banded matrix (cyclic if curve is closed)
(invert with Cholesky decomposition)

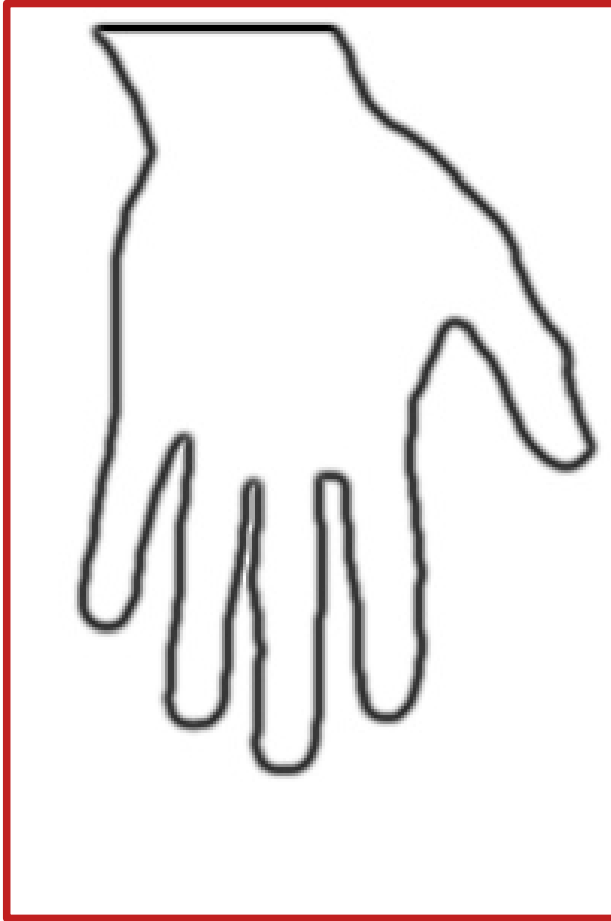
Things to consider

- Important parameters:
 - α , β and γ
 - E_{ext}
 - initial snake $\{X_0, Y_0\}$
 - number of iterations
- The curve needs to be sampled densely to be able to compute $\vec{v}'''(s)$ accurately
- When the snake evolves, it is important to resample the curve regularly

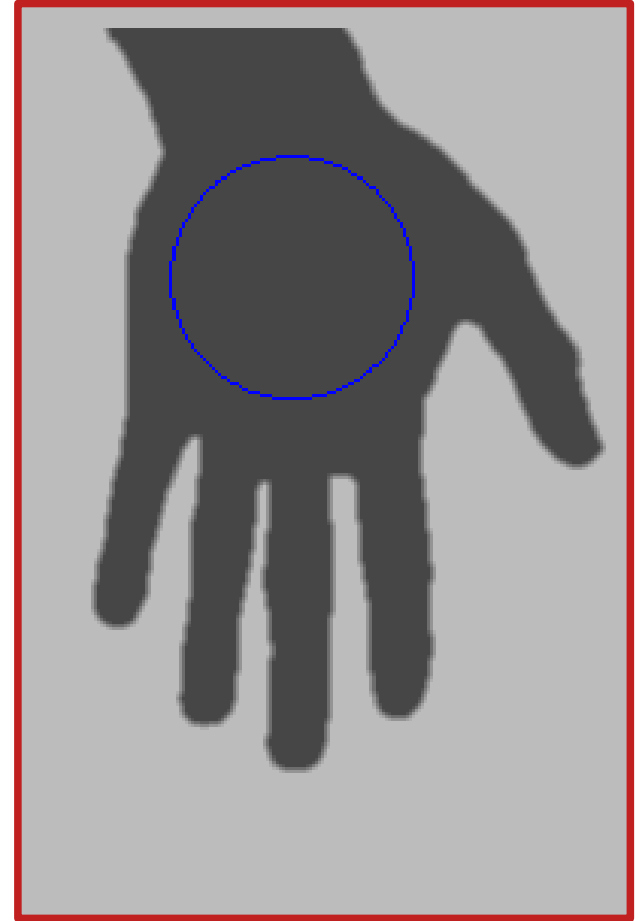
Snakes in action



input image



- gradient magnitude



evolving snake

E_{ext}

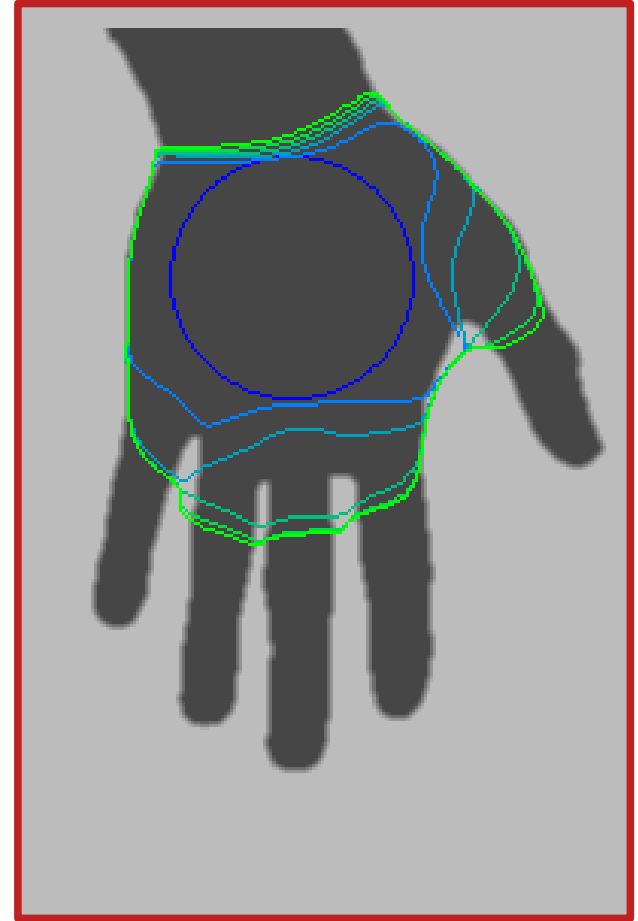
Snakes in action



input image



- gradient magnitude



evolving snake

E_{ext}

Issues with snakes

- Snakes cannot flow into elongated structures
- Snakes cannot move towards edges that are far away
 - when increasing the size of the Gaussian for the gradient magnitude, the edges of the object will move (remember scale-space lecture)
- Snakes do not work well when the initial curve intersects the edges you're looking for
- Solution:
don't use an external energy, but an external force!

When using E_{ext} , the gradient descent equation has an external force $\vec{F}_{\text{ext}} = \nabla E_{\text{ext}}$

Improving the snake

- Add a dynamic force:
(depends on the curve)
 - the balloon force
- Change the static force:
(independent of the curve,
usually depends on the input image)
 - using the distance transform
 - Gradient Vector Flow
 - Vector Field Convolution
 - or anything else you can come up with...

The balloon force

- Cohen (1989)
- Also known as pressure force
- Adds an **outward, expanding force** to the curve (or an **inward, contracting force**)
- Requires the curve to be initialised **inside** the object (or **outside**)
- Used in addition to the traditional static force
- Adds a new parameter to the method

$$\alpha \vec{V}''(s) - \beta \vec{V}''''(s) - \nabla E_{\text{ext}}(\vec{V}(s)) + \kappa \vec{n}(s) = 0$$

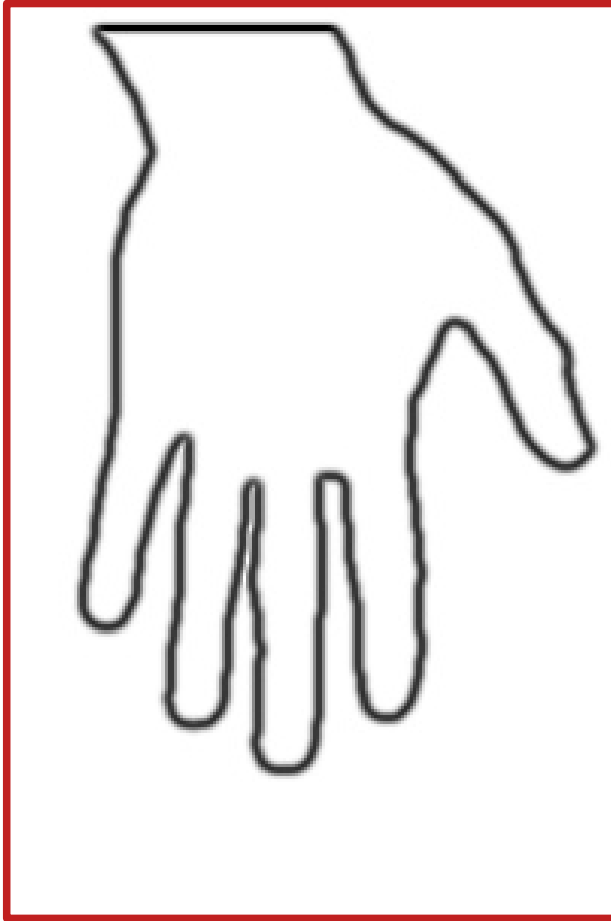


the unit normal vector

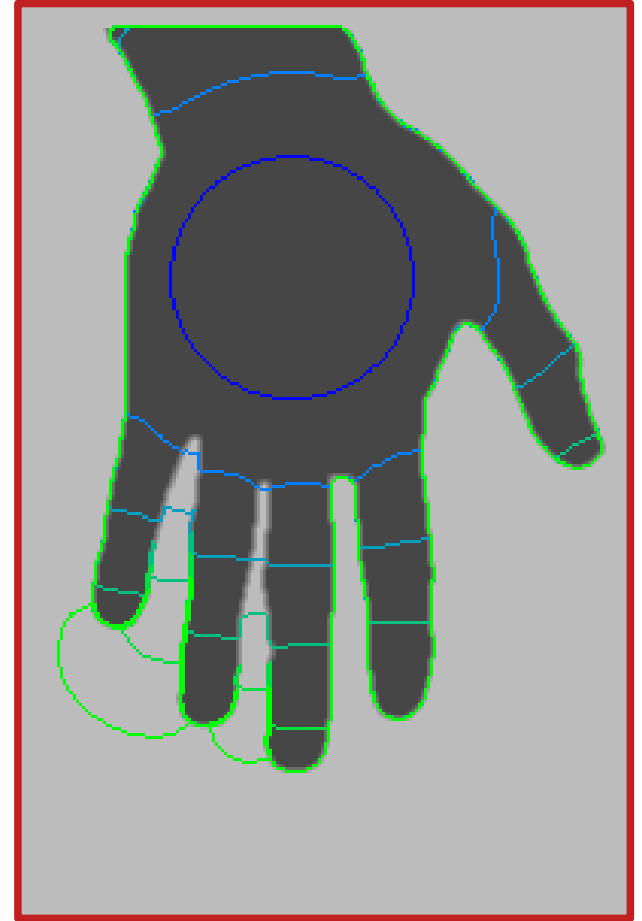
The balloon force



input image



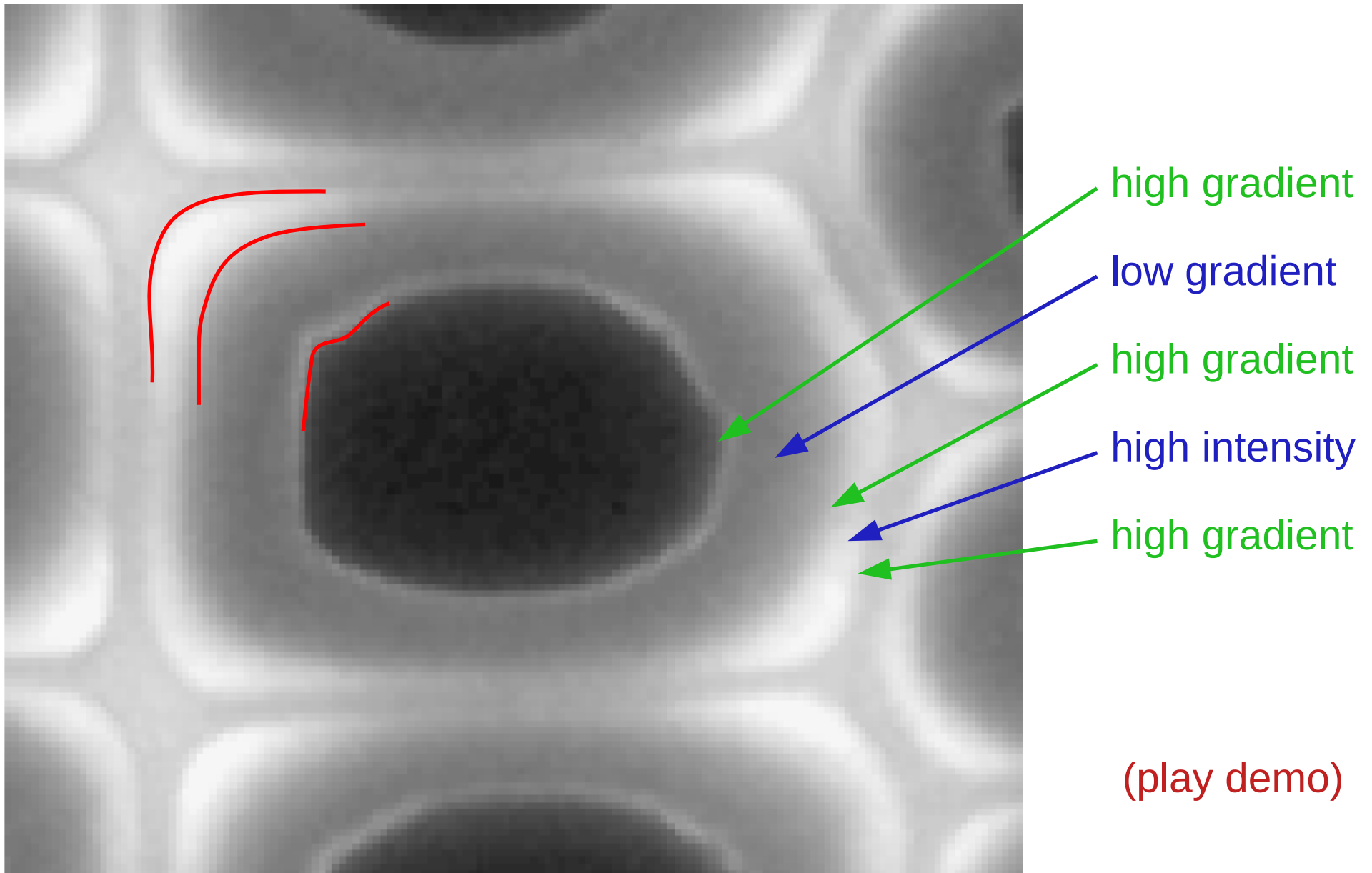
- gradient magnitude



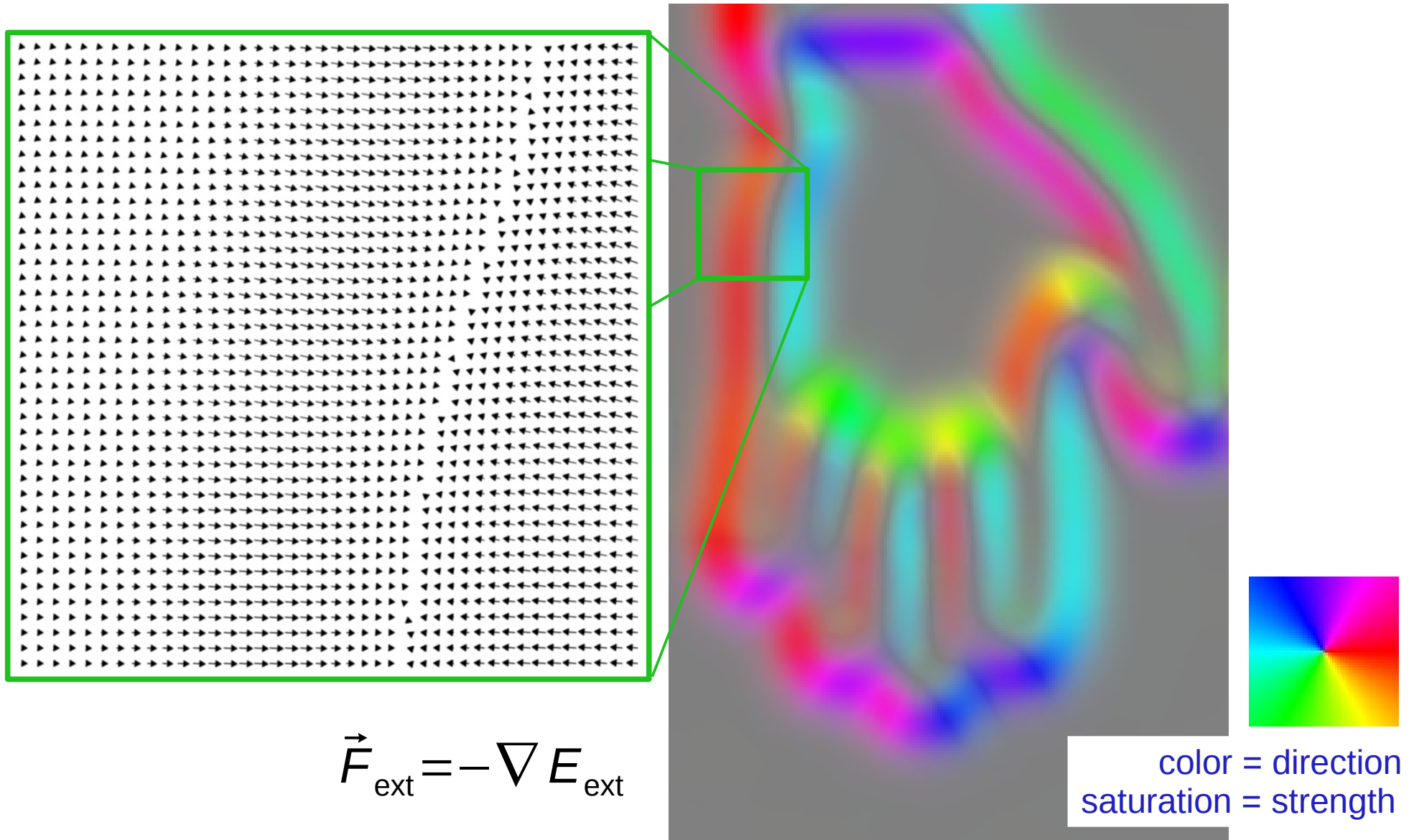
evolving snake

$$E_{\text{ext}}$$

The balloon force

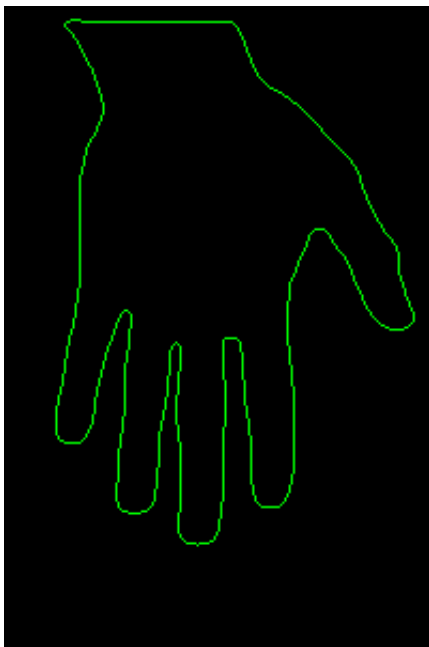


Static forces



The distance force

- Cohen & Cohen (1993)
- Force based on the distance transform of the edges
 - the force fields pulls the snake towards the closest edge, no matter how far away the snake starts



$$E = \text{Canny}(I)$$



$$D = \text{DT}(\bar{E})$$



$$\vec{F}_{\text{ext}} = -\nabla D$$

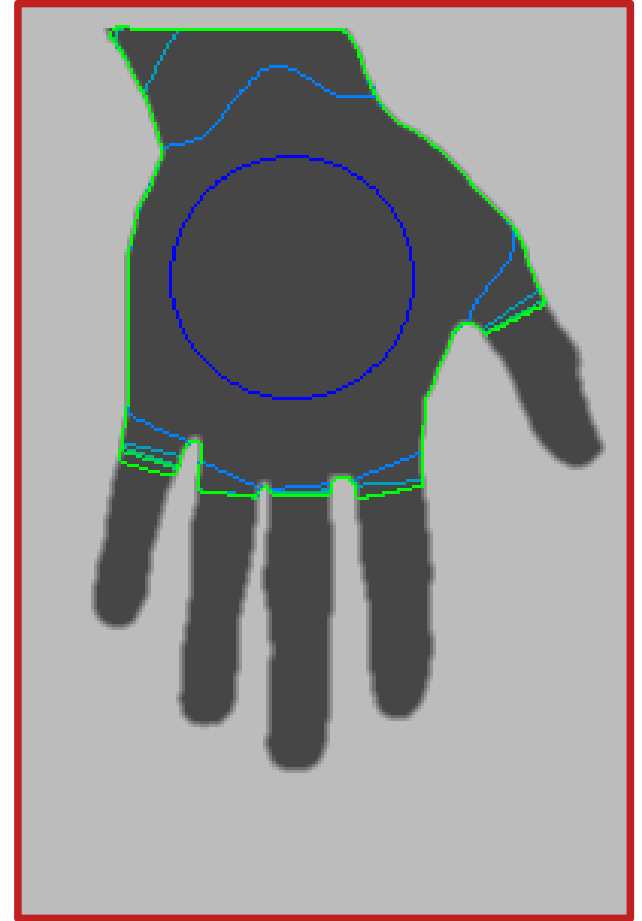
The distance force



input image



gradient of
distance transform



evolving snake

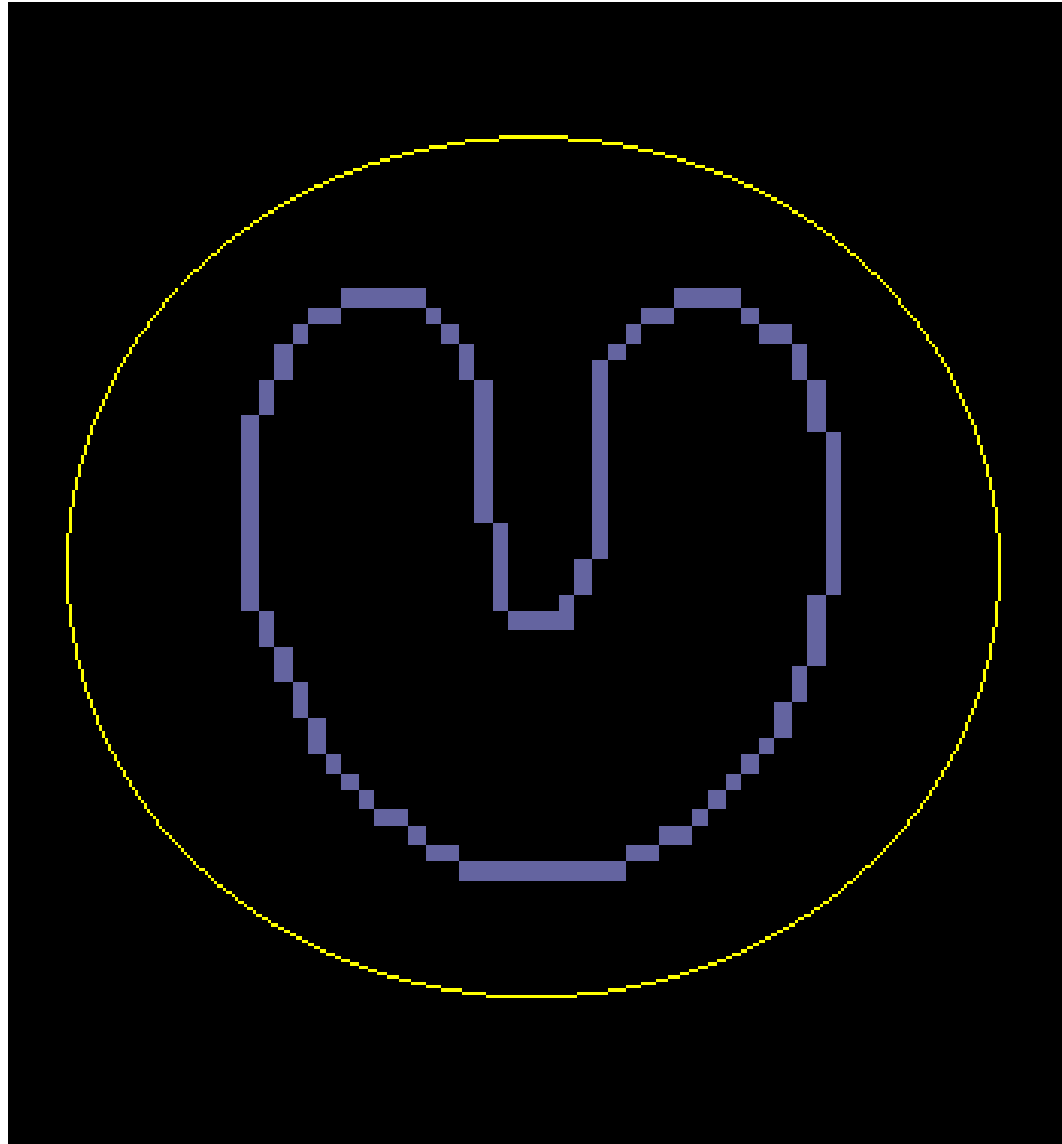
Gradient vector flow

- Xu & Prince (1998)
- More “refined” way of creating a force field across the whole image
- Propagate gradient information
 - stronger edges carry more weight
 - force can pull snake into narrow structures
 - *normalize vectors*
- Computed by finding $\vec{F}_{\text{gvf}} = (U_{\text{gvf}}, V_{\text{gvf}})^T$ that minimizes

$$E = \int \mu \underbrace{\left(|\nabla U_{\text{gvf}}|^2 + |\nabla V_{\text{gvf}}|^2 \right)}_{\text{smoothness term}} + \underbrace{|\nabla E_{\text{ext}}|^2 \left| \vec{F}_{\text{gvf}} + \nabla E_{\text{ext}} \right|}_{\text{close to edges:}} d\vec{x}$$

$$\vec{F}_{\text{gvf}} = -\nabla E_{\text{ext}}$$

Gradient vector flow



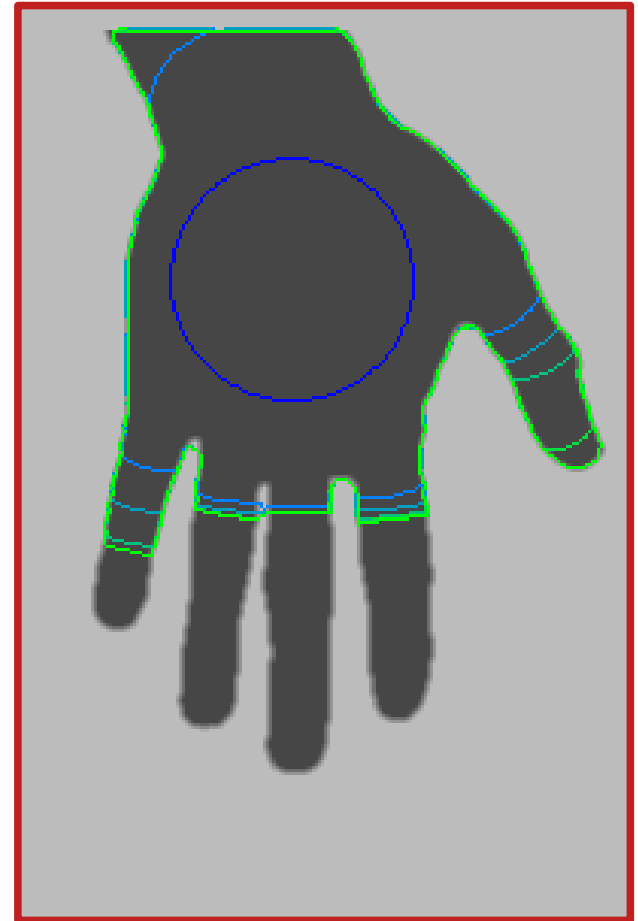
Gradient vector flow



input image



GVF force



evolving snake

this is how far the forces were propagated,
keep iterating to fill the image!

Vector field convolution

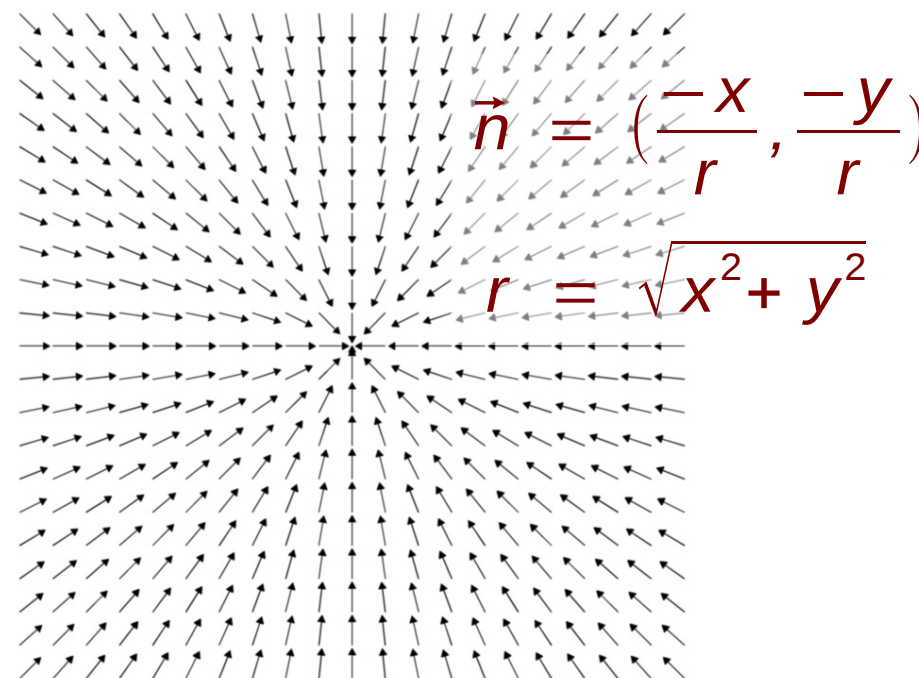
- Li & Acton (2007)
- Very similar to GVF, but much easier & faster to compute:
 - create a vector field kernel \vec{k}
 - convolve edge map with kernel $\vec{F}_{\text{vfc}} = |\nabla I| \otimes \vec{k}$
 - *normalize vectors*

$$\vec{k} = (r + \varepsilon)^{-\gamma} \vec{n}$$

$$\gamma \in [1, 3]$$

$$\gamma = 2 \Rightarrow \text{gravitation!}$$

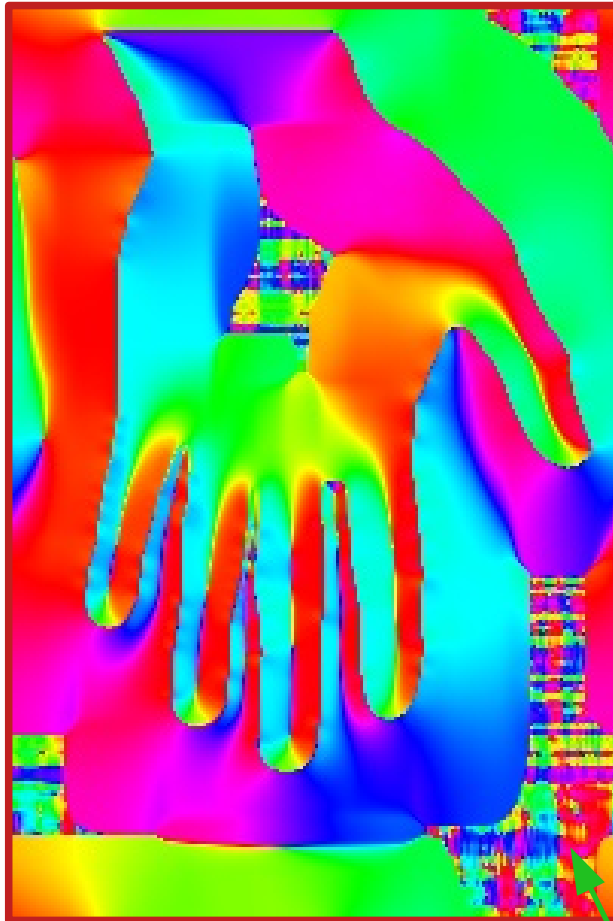
ε is some small
value to avoid
division by 0



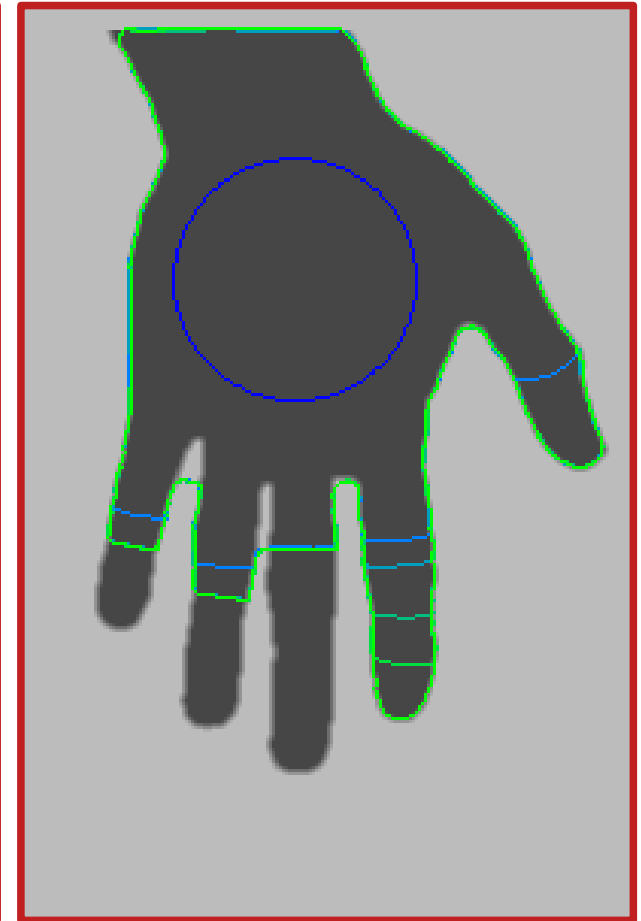
Vector field convolution



input image



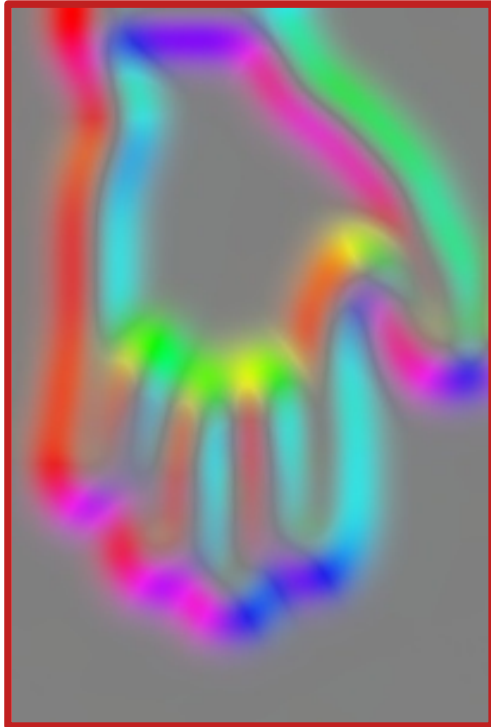
VFC force



evolving snake

this is numerical inaccuracy
increase kernel size to avoid this!

The external forces



gradient



distance



GVF



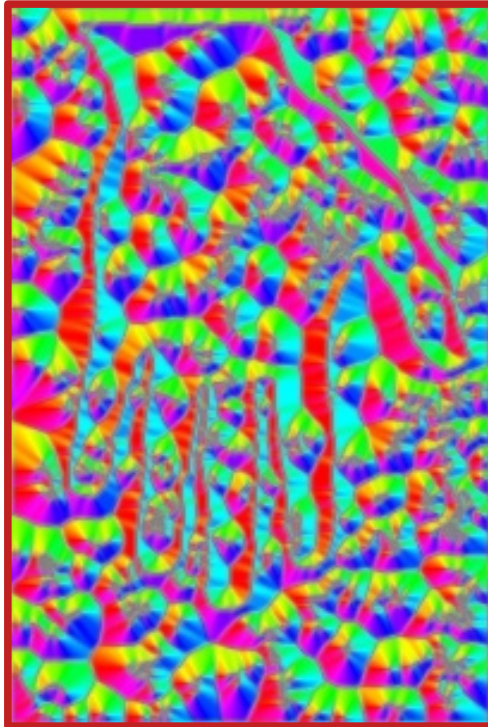
VFC

What do the improved external forces have in common?

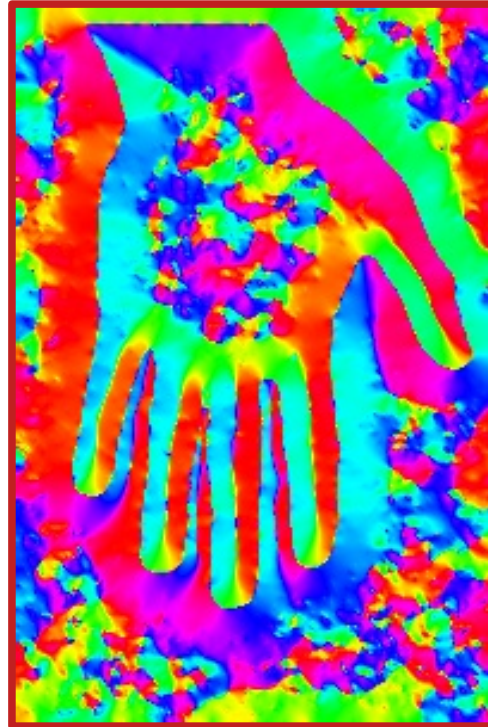
What about noise?



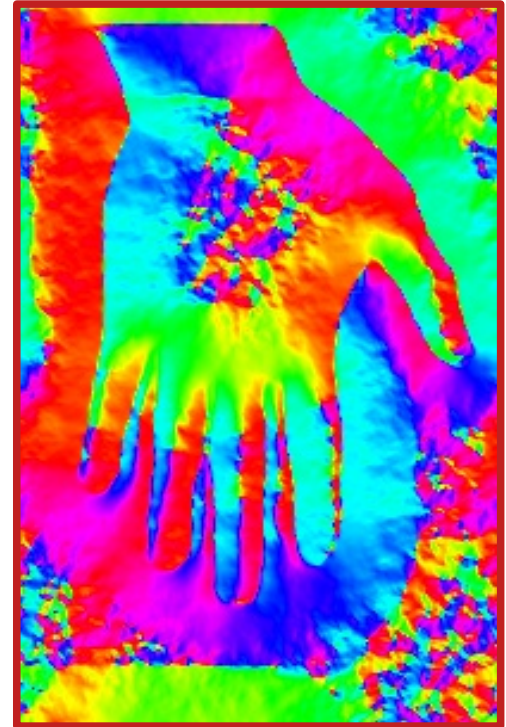
gradient



distance

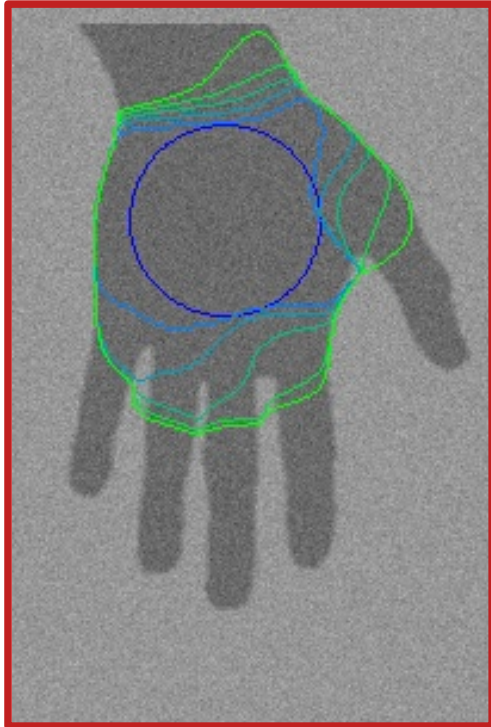


GVF

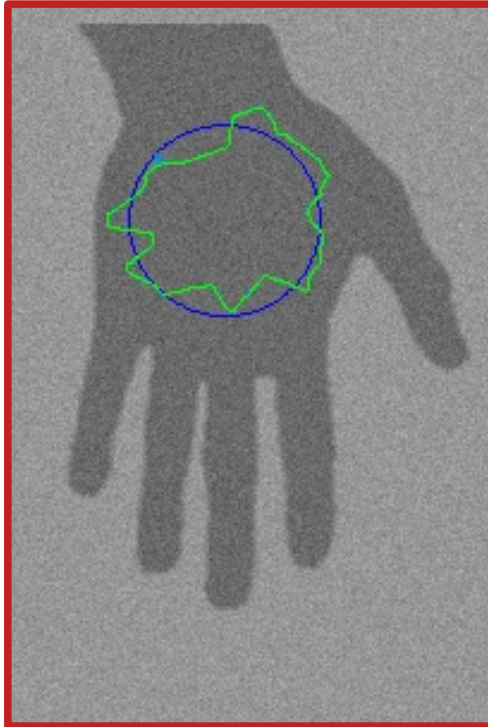


VFC

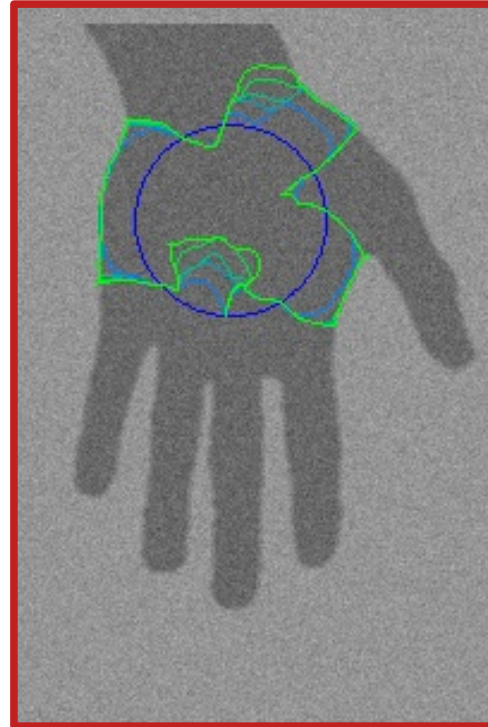
What about noise?



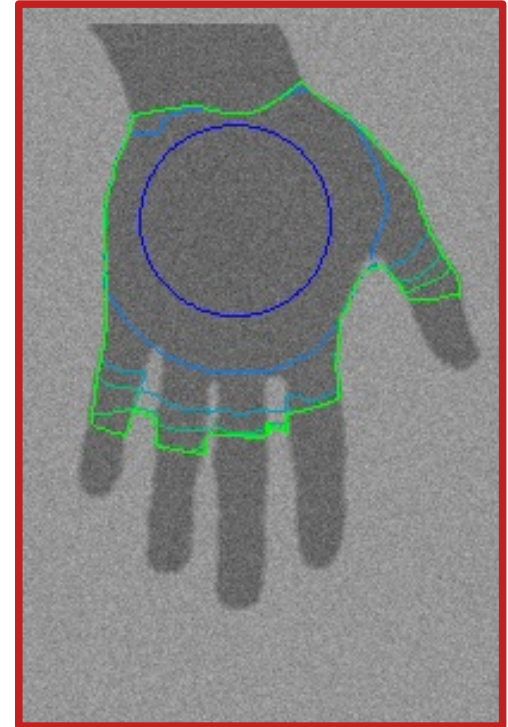
gradient



distance

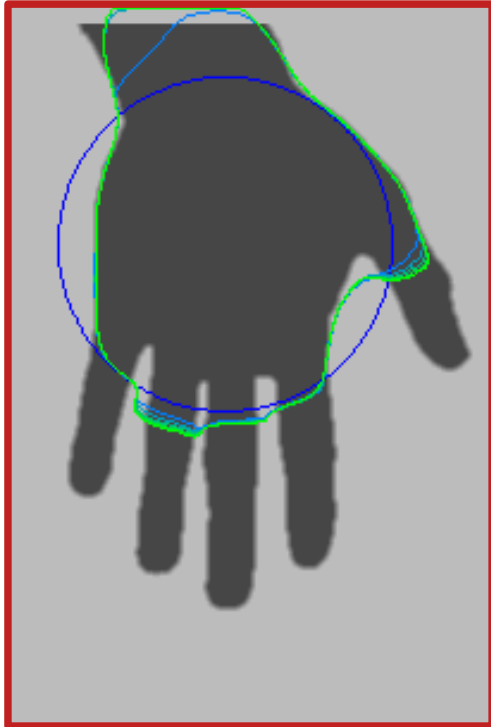


GVF

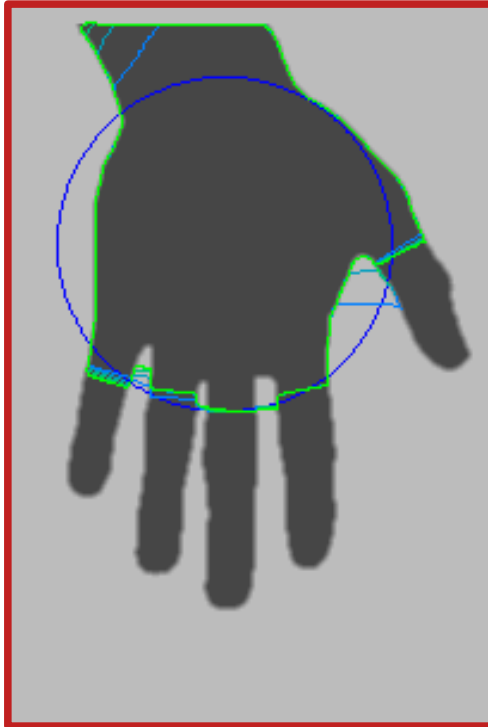


VFC

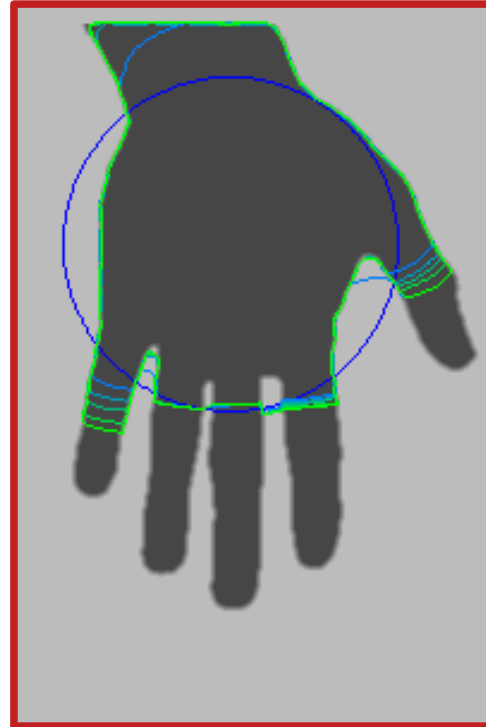
What about initialization?



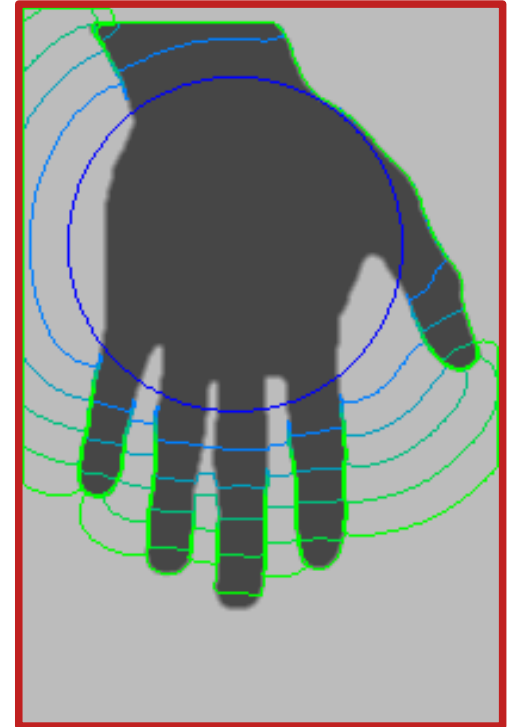
gradient



distance



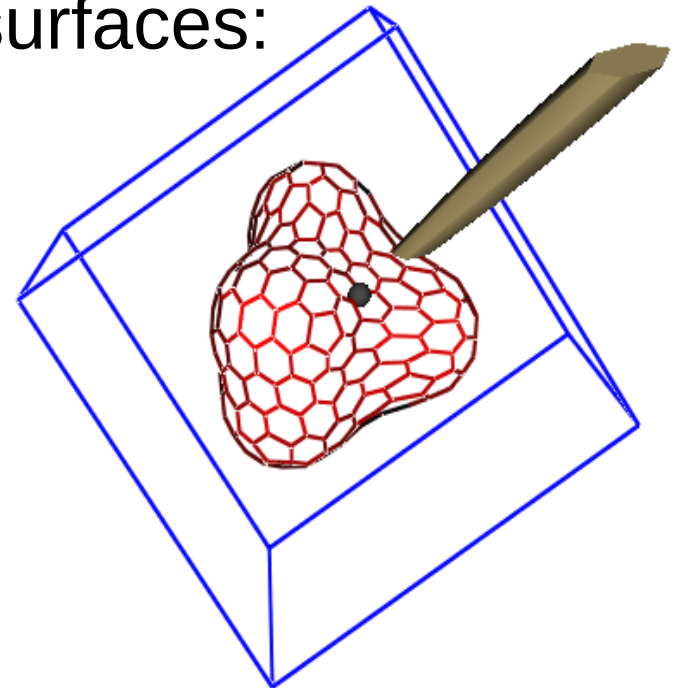
GVF



balloon

Snakes in 3D

- Snakes are explicitly 2D constructs, but 3D generalization is not impossible
- First “try” involved 2D snakes on successive slices of the volume, initializing each with the final result of the previous slice
- The better approach is with active surfaces:
 - surface represented as a mesh
 - internal forces (derivatives) calculated using mesh neighbours
 - all external force formulations generalize easily to 3D

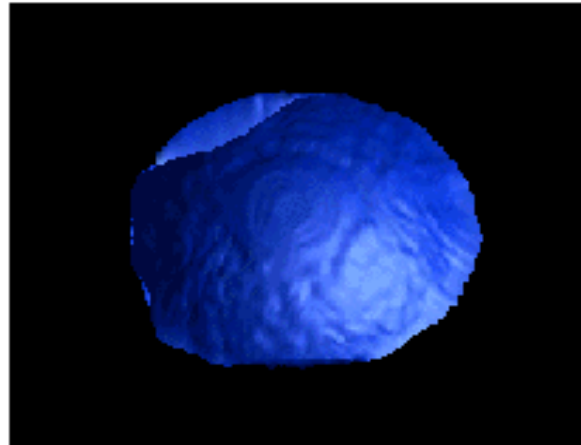


Snakes in 3D

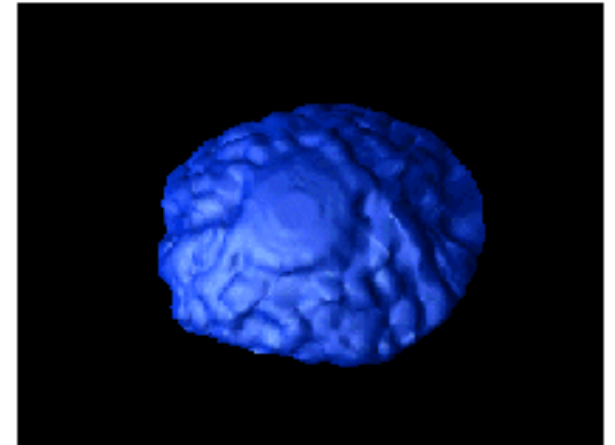
iter = 1



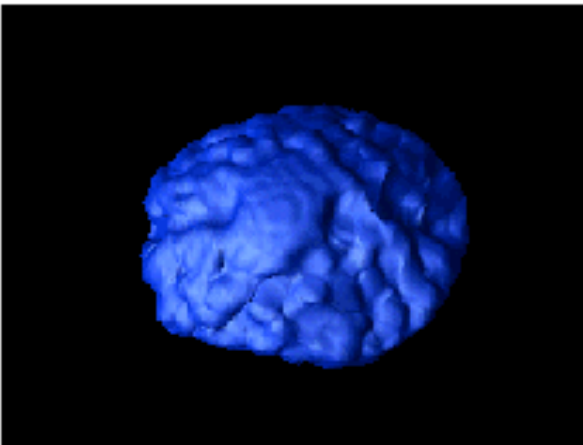
off bottom view



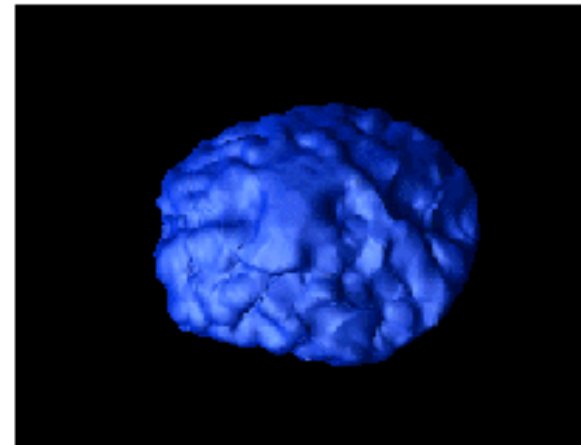
iter = 10



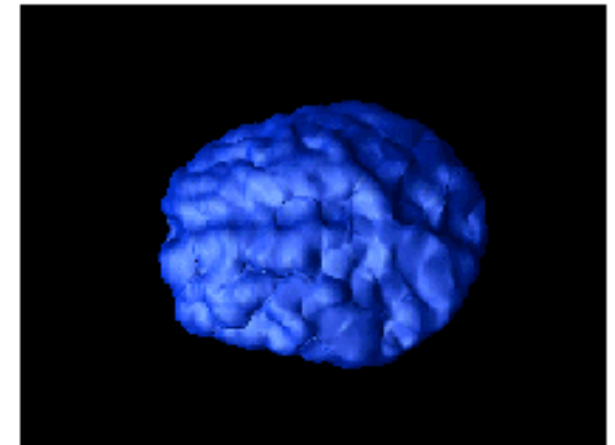
iter = 30



iter = 60



iter = 200



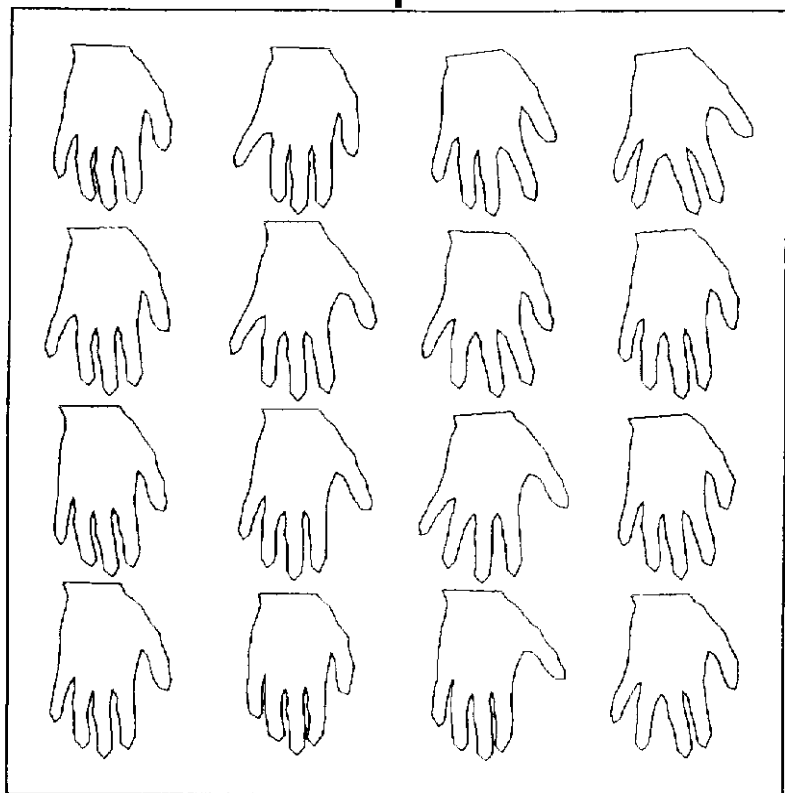
Active shape models

- A snake uses application-specific knowledge by:
 - the initialization (the initial shape to be deformed)
 - the choice of parameters (α , β , γ , κ , ...)
 - the choice of external force
- Once the snake starts to deform no further knowledge is used (it just looks for a stable point)
- Sometimes we know in advance the shape of the object we are looking for
- How can we incorporate this information into the snake deformation algorithm?

Adding shape information allows detection of partially occluded shapes, for example.

Active shape models

- Cootes, Taylor, Cooper & Graham (1994)
- Like a snake, detects an object in an image
- Control the possible ways that the snake deforms
- Learn the possible shapes from examples



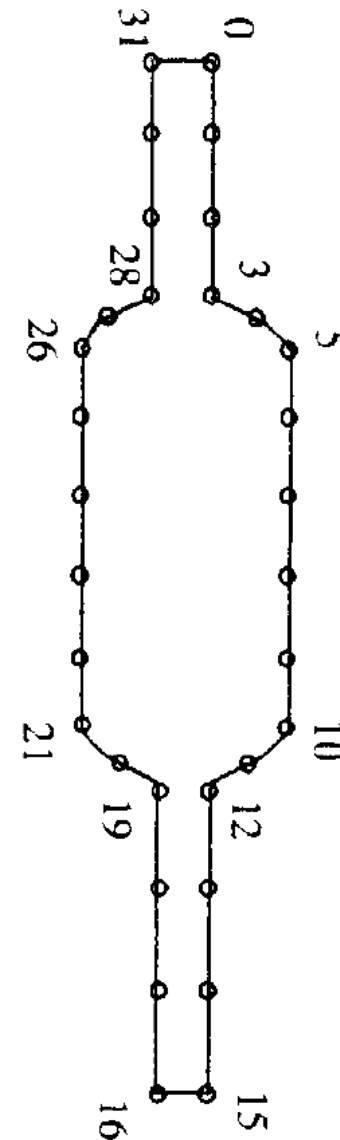
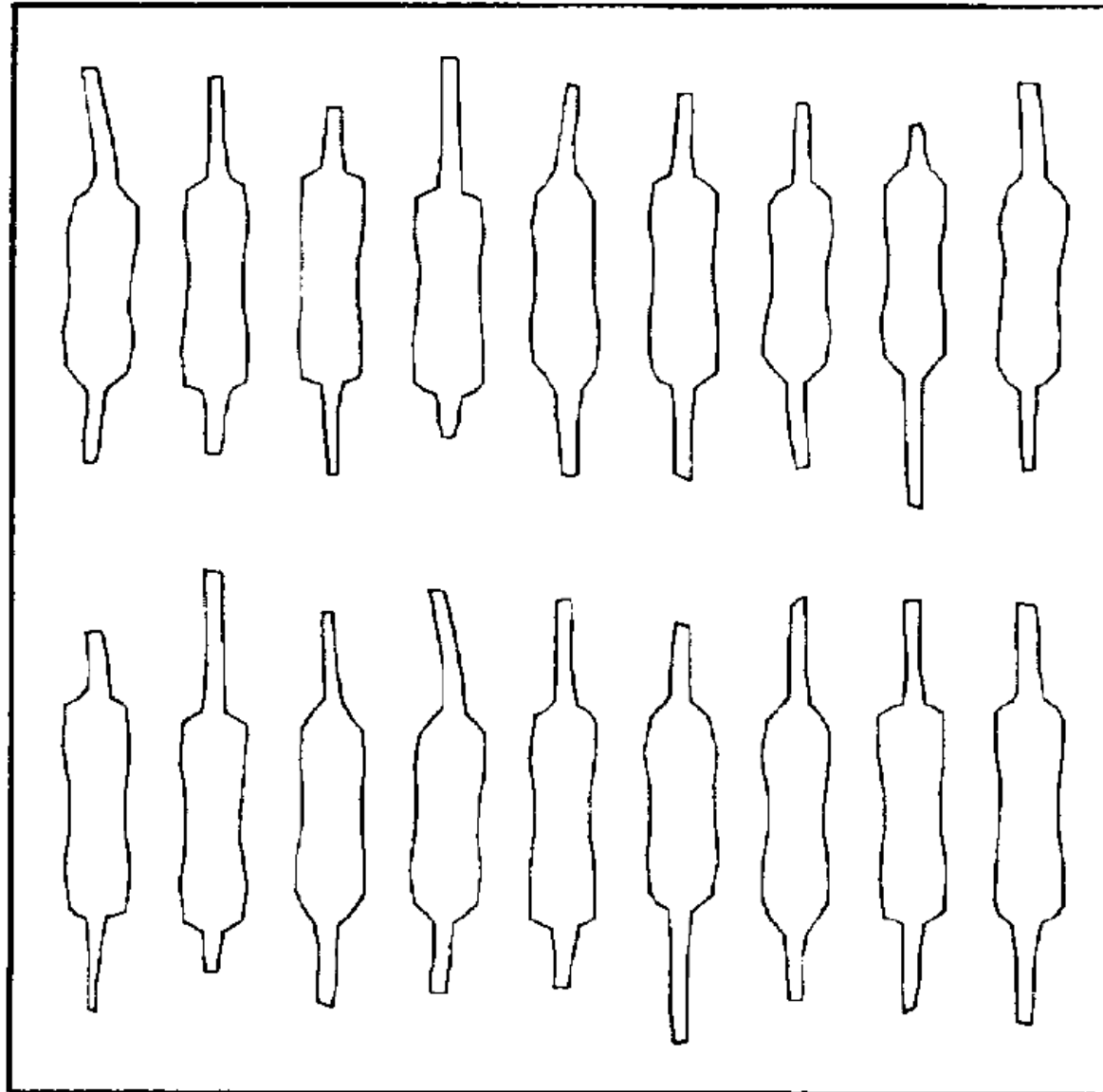
For example:
when looking for a hand, we know that the fingers can have many different positions with respect to the hand, but we also know that there are exactly 5 of them, that they all have a fixed length, and that they cannot bend at e.g. 90°

(All the examples I'm showing here are from the Cootes et al. paper)

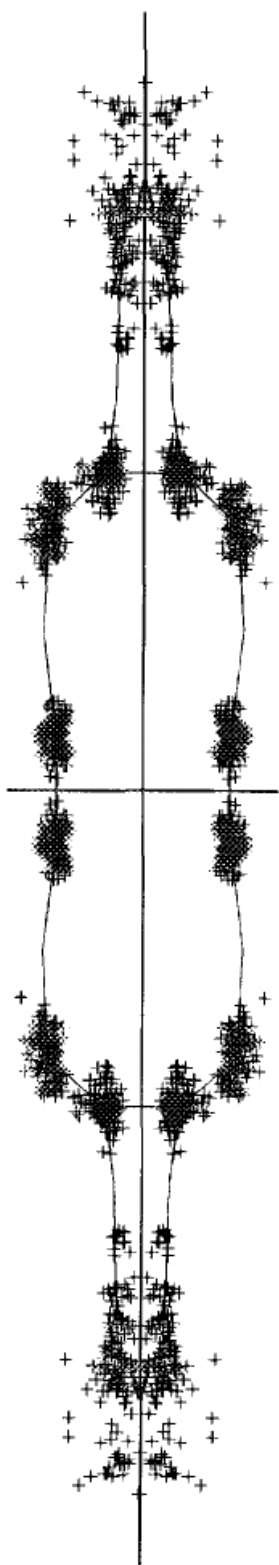
Active shape models

- The Active Shape Model (ASM) consists of a set of points describing a boundary, and for each point, a distance along the x and y axes it is allowed to move
- The ASM is trained by (manually) selecting points along the object's contour in a set of training images
 - in each image, each point must match the same location along the contour of the object
- It is required that the training images contain all possible shape variations – no shape changes will be allowed unless they are represented in the test set!
- The ASM is fitted to a new image much like a snake, but instead of internal forces there is a constraint on the relative position of the points representing the boundary

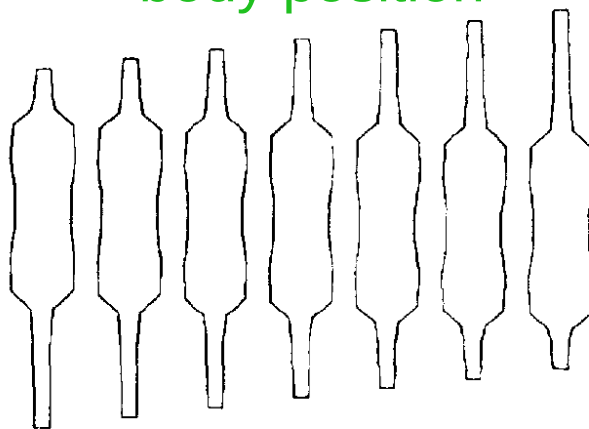
Example: resistors



Example: resistors

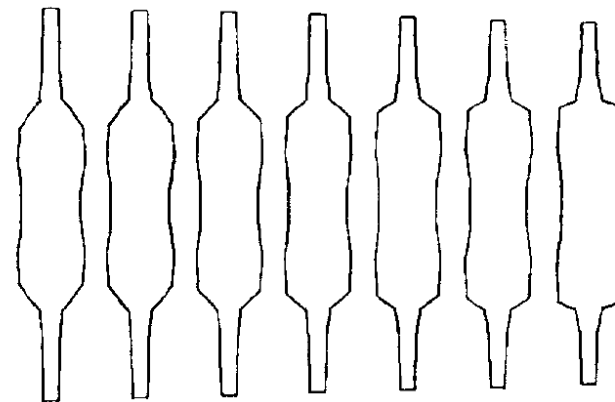


body position



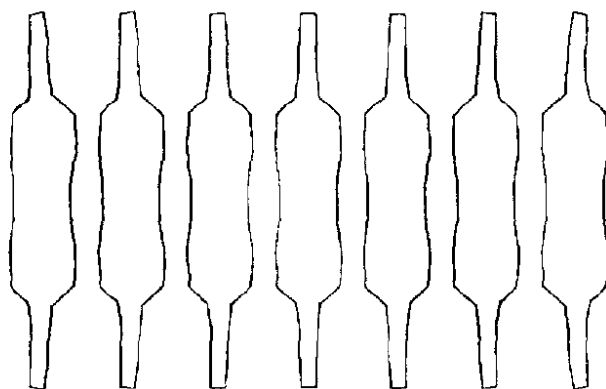
$$-2\sqrt{\lambda_1} \longleftarrow b_1 \longrightarrow 2\sqrt{\lambda_1}$$

“shoulder” shape



$$-2\sqrt{\lambda_2} \longleftarrow b_2 \longrightarrow 2\sqrt{\lambda_3}$$

wire curvature



$$-2\sqrt{\lambda_3} \longleftarrow b_3 \longrightarrow 2\sqrt{\lambda_3}$$

These are the 3 most important modes of variation captured by the model

ASM training

- The training set can be seen as M $(2N)$ -dimensional vectors:

$$\vec{x} = (x_1, y_1, x_2, y_2, \dots, x_N, y_N)^T$$

M images
2 dimensions
 N points

- Aligning training set:

- rotate, scale & translate each shape to align with 1st shape
- calculate mean shape
- normalize orientation, scale and origin of mean
- align all shapes to the mean
- recalculate mean & repeat until convergence

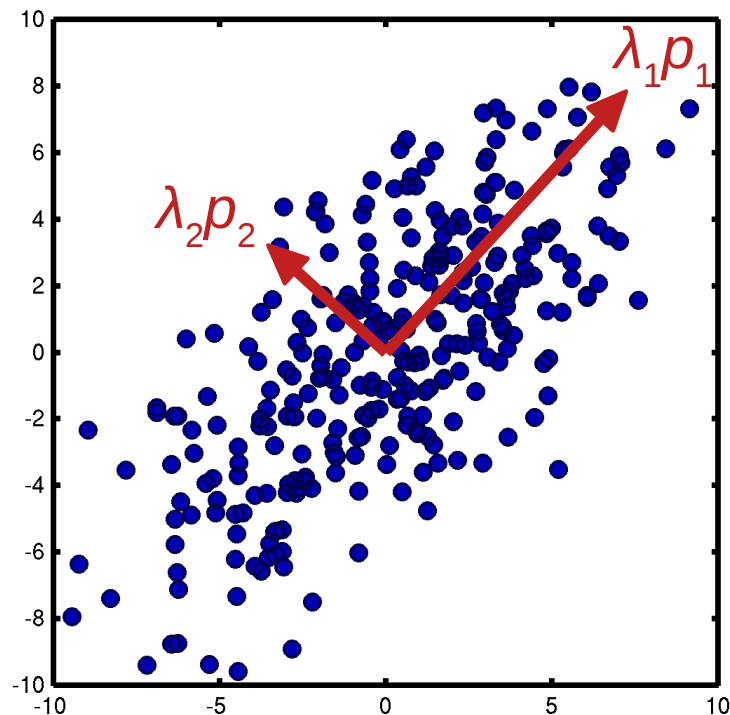
$$\hat{\vec{x}} = \frac{1}{M} \sum_{i=1}^M \vec{x}_i$$

- After alignment, the vectors occupy a subset of \mathbb{R}^{2N}
- Simplify the training set using PCA

Note: this divides up the differences between images into a rigid transformation and a non-rigid deformation of the shape

Principal component analysis

- PCA extracts orthogonal vectors describing the most important axes in the data
- Assumes normal (Gaussian) distribution of points!



$$d\vec{x}_i = \vec{x}_i - \hat{\vec{x}}$$

$$S = \frac{1}{M} \sum_{i=1}^M d\vec{x}_i d\vec{x}_i^T$$

$$S \vec{p}_k = \lambda_k \vec{p}_k$$

S is $2N \times 2N$ covariance matrix
 λ_k is eigenvalue
 \vec{p}_k is unit-length vector

The active shape model

- Any allowed shape is given by:

$$\vec{x} = \hat{\vec{x}} + P \vec{b}$$

$$P = (\vec{p}_1, \vec{p}_2, \dots, \vec{p}_K) \quad (K \leq 2N)$$

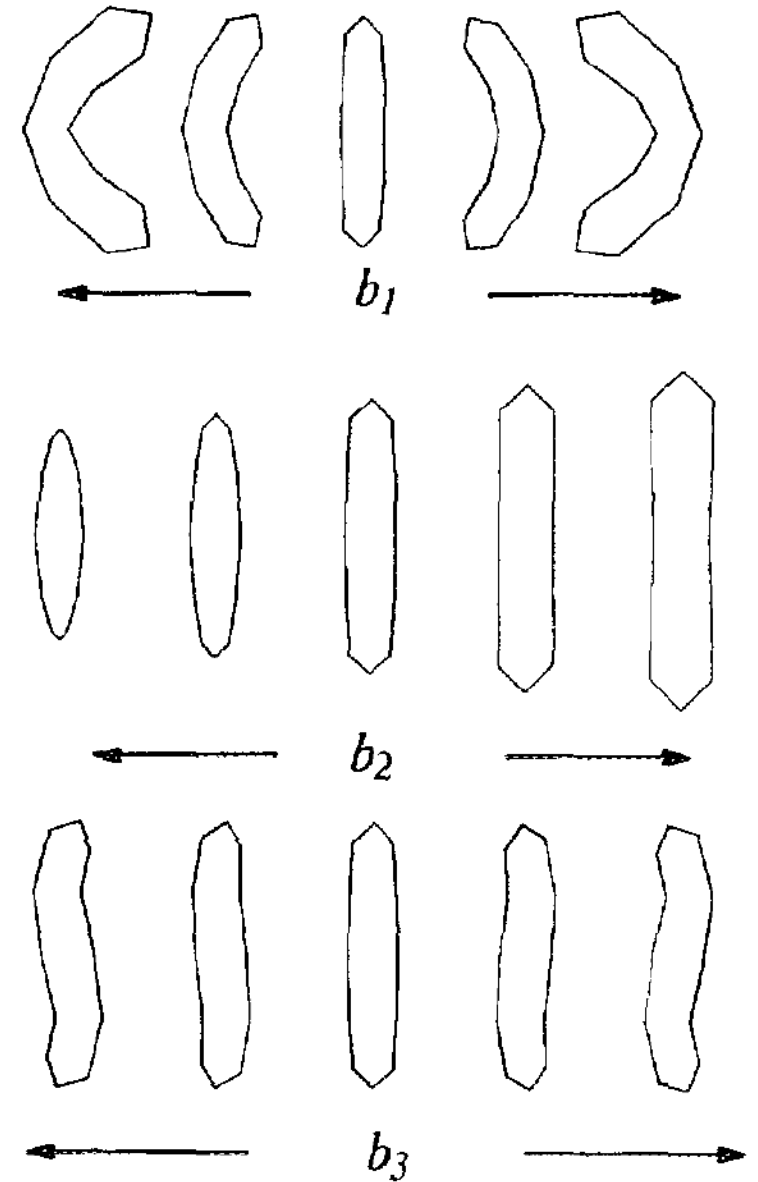
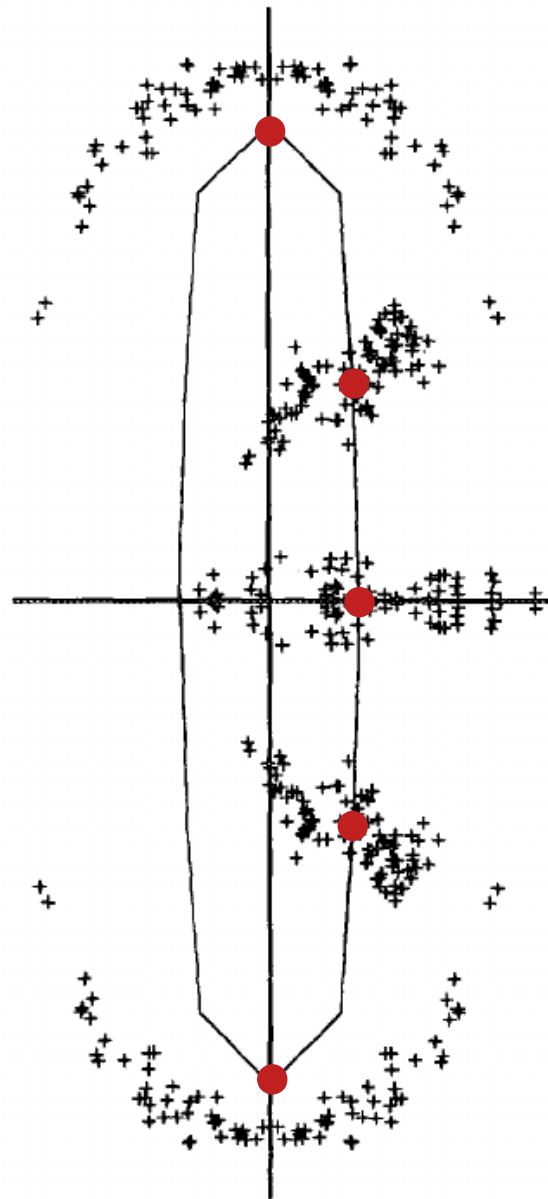
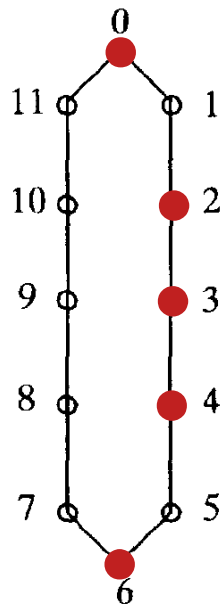
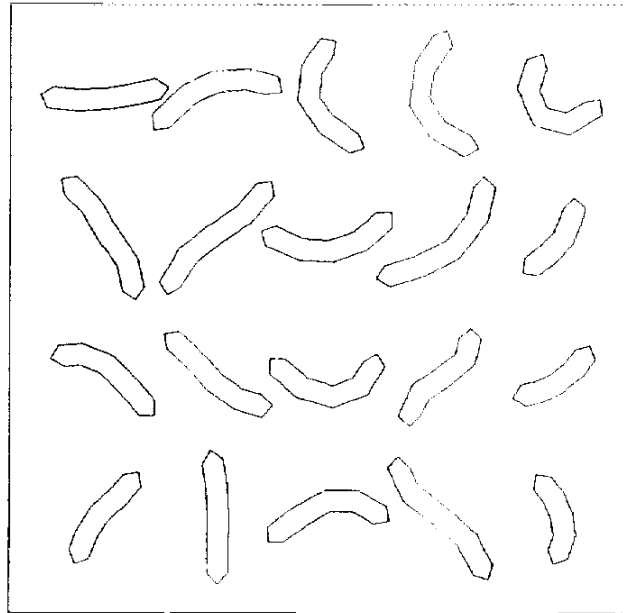
weights to be fitted

(matrix formed by first K eigenvectors)

$$-3\sqrt{\lambda_i} \leq b_i \leq 3\sqrt{\lambda_i}$$

- Any rotation, scaling and translation of an allowed shape is also allowed
- Because of the assumption of normally distributed points, certain shape variations cannot be modeled!

When the model fails



Fitting an ASM to an image

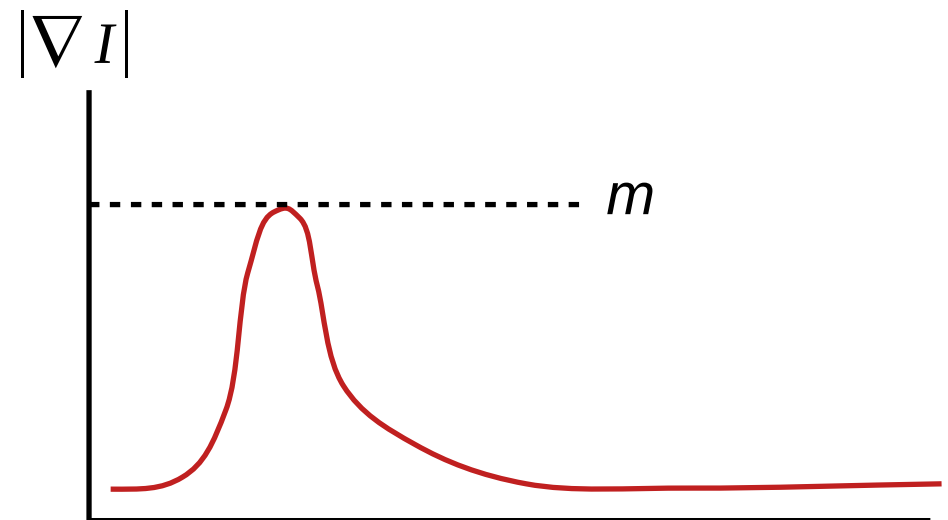
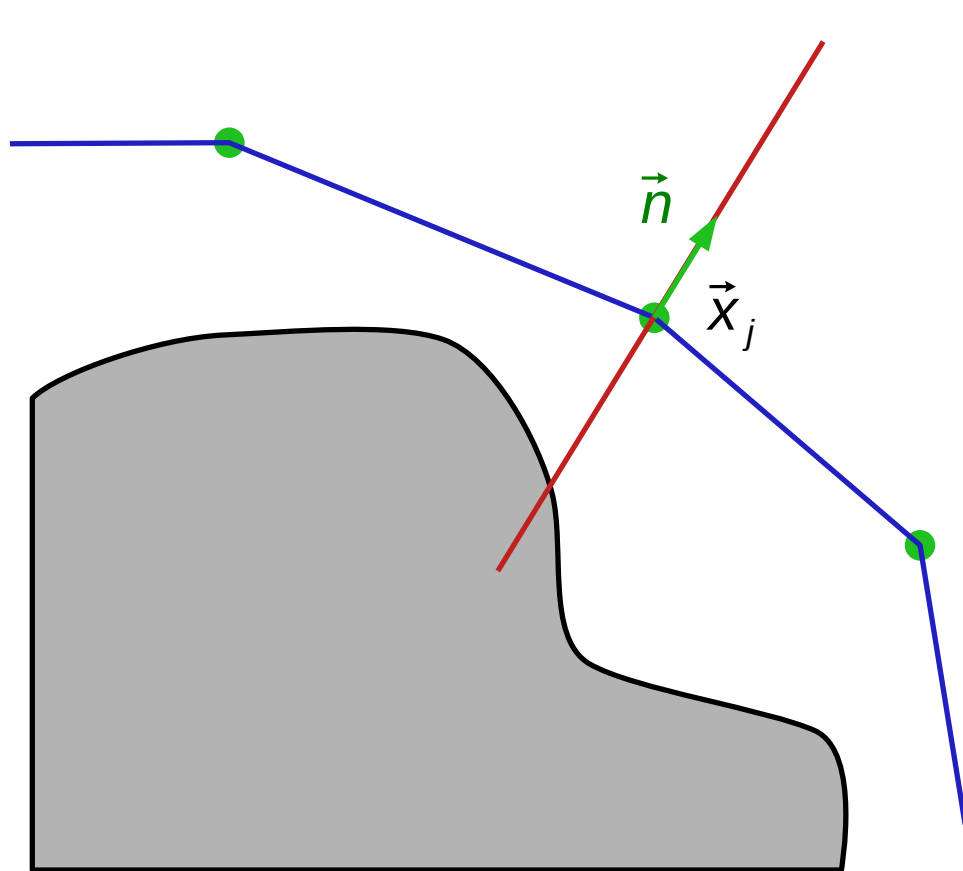
- Choose an initial placement
 - origin (x_o, y_o) , scale s & rotation θ
 - use the mean shape, that is, all $b_i = 0$

$$\vec{X} = M_{s, \theta}(\hat{\vec{X}} + P\vec{b}) + T(x_o, y_o)$$

- Repeat until convergence:
 - look for a better position for each of the points on the shape
 - split the movement into rigid and non-rigid components
 - transform non-rigid components into a shape change
 - constrain shape change to allowed shapes
- “Convergence” can be defined in many ways...

Fitting an ASM to an image

- Step 1: Look, for each point, along normal of curve, for a strong edge
 - this yields an adjustment for each point $d\vec{x}_j = \kappa m \vec{n}$



$d\vec{x}$ composed of all $d\vec{x}_j$ so that $\vec{x} + d\vec{x}$ is the new shape

Fitting an ASM to an image

- Step 2: Find a ds , $d\theta$, dx_o and dy_o that best aligns \vec{x} to $\vec{x} + d\vec{x}$
- Step 3: Find residual adjustments $d\vec{u}$ in local coordinate frame:

$$\vec{x} = M_{s,\theta}(\hat{\vec{x}} + P\vec{b}) + T(x_o, y_o)$$

$$\vec{x} + d\vec{x} = M_{s+ds,\theta+d\theta}(\hat{\vec{x}} + P\vec{b} + d\vec{u}) + T(x_o + dx_o, y_o + dy_o)$$

$$d\vec{u} = M_{(s+ds)^{-1},(\theta+d\theta)^{-1}} \left\{ M_{s,\theta}(\hat{\vec{x}} + P\vec{b}) + d\vec{x} - T(dx_o, dy_o) \right\} - (\vec{x} + P\vec{b})$$

Fitting an ASM to an image

- Step 4a: Map $d\vec{u}$ onto subdomain P:

$$d\vec{b} = P^T d\vec{u} \quad (\text{note that } P^T = P^{-1})$$

(this is a least-squares approximation!)

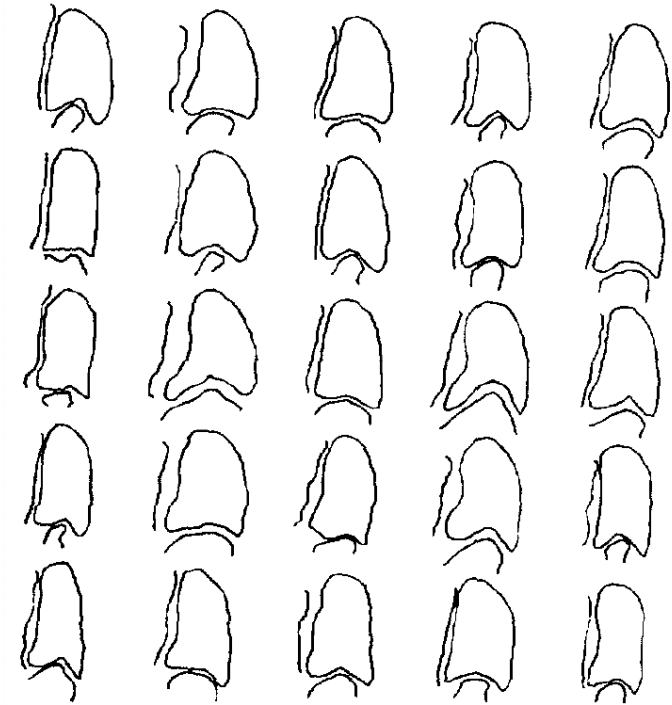
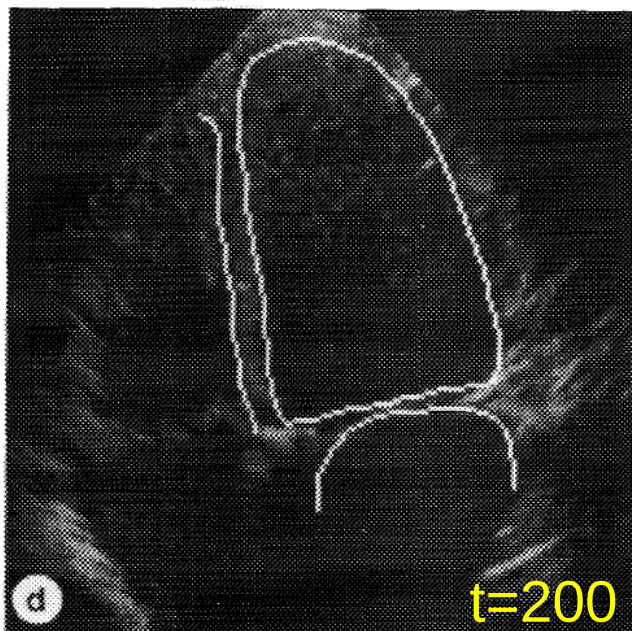
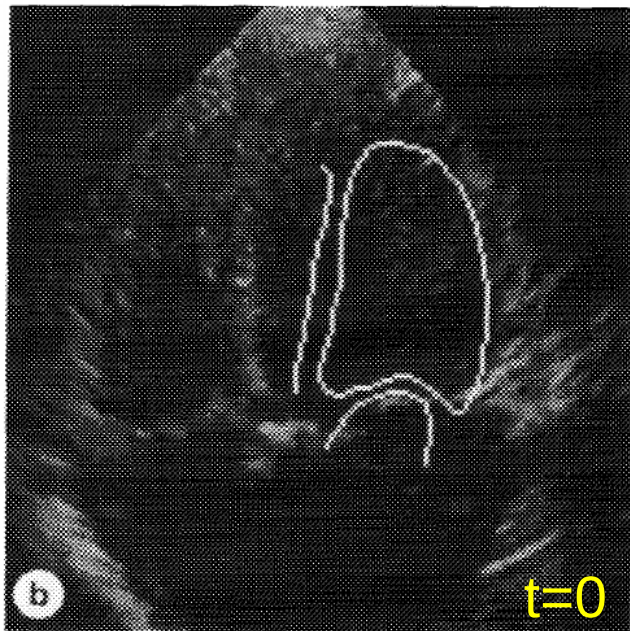
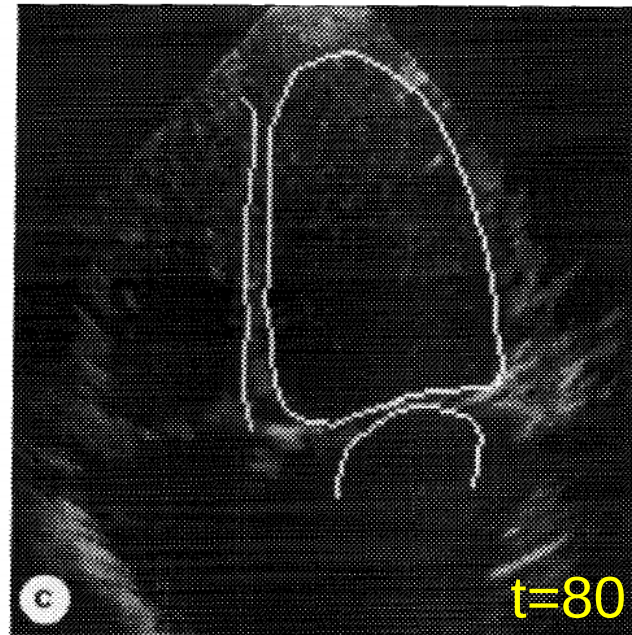
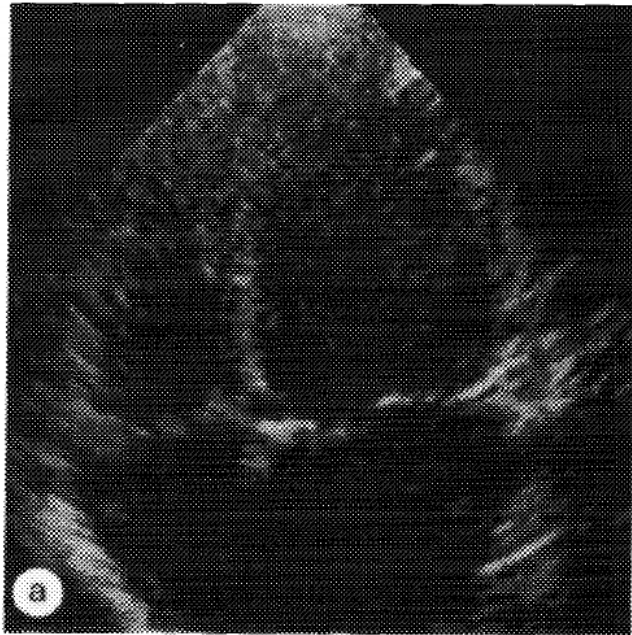
- Step 4b: Update shape parameters & limit \vec{b} to allowed range:

$$\left\{ \begin{array}{lcl} s & \leftarrow & s + \omega_s ds \\ \theta & \leftarrow & \theta + \omega_\theta d\theta \\ x_o & \leftarrow & x_o + \omega_{x_o} dx_o \\ y_o & \leftarrow & y_o + \omega_{y_o} dy_o \\ b_i & \leftarrow & b_i + \omega_{b_i} db_i \end{array} \right.$$

$$-3\sqrt{\lambda_i} \leq b_i \leq 3\sqrt{\lambda_i}$$

the weights ω make the movement slower – this avoids overshooting the target

ASM example



heart ventricle model
with 96 points, 12
degrees of freedom

Active appearance models

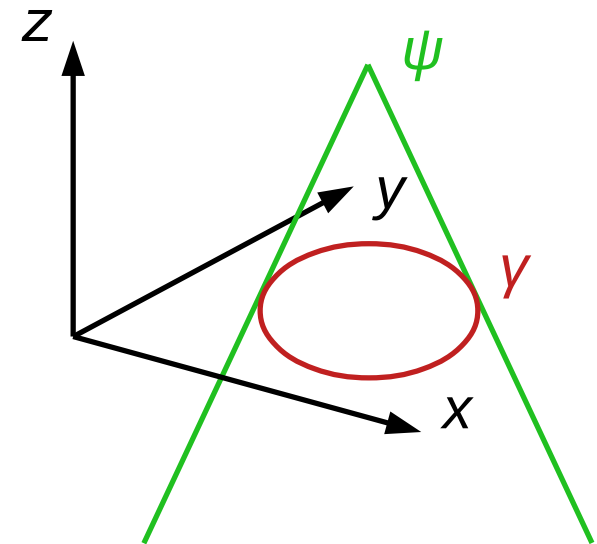
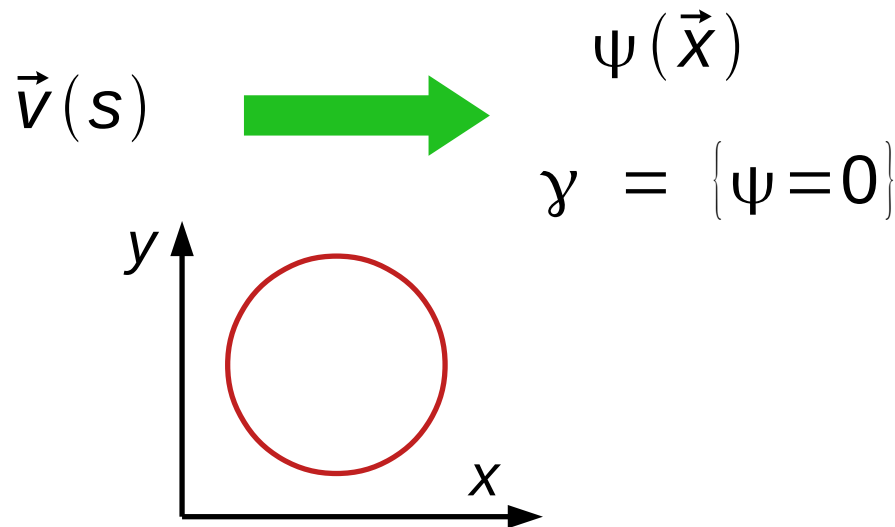
- Just like ASM, but also includes information on grey values inside of shape:
 - $2N$ parameters describing boundary of shape
 - PCA yields n shape vectors
 - M parameters describing grey values
(i.e. the pixels after scaling, rotating and shifting the patch to the mean shape)
 - PCA yields m intensity vectors
 - we now have $n+m$ vectors, which we combine but weigh differently, depending on relative importance
 - perform PCA again on matrix of vectors, further simplifying the model

Level sets

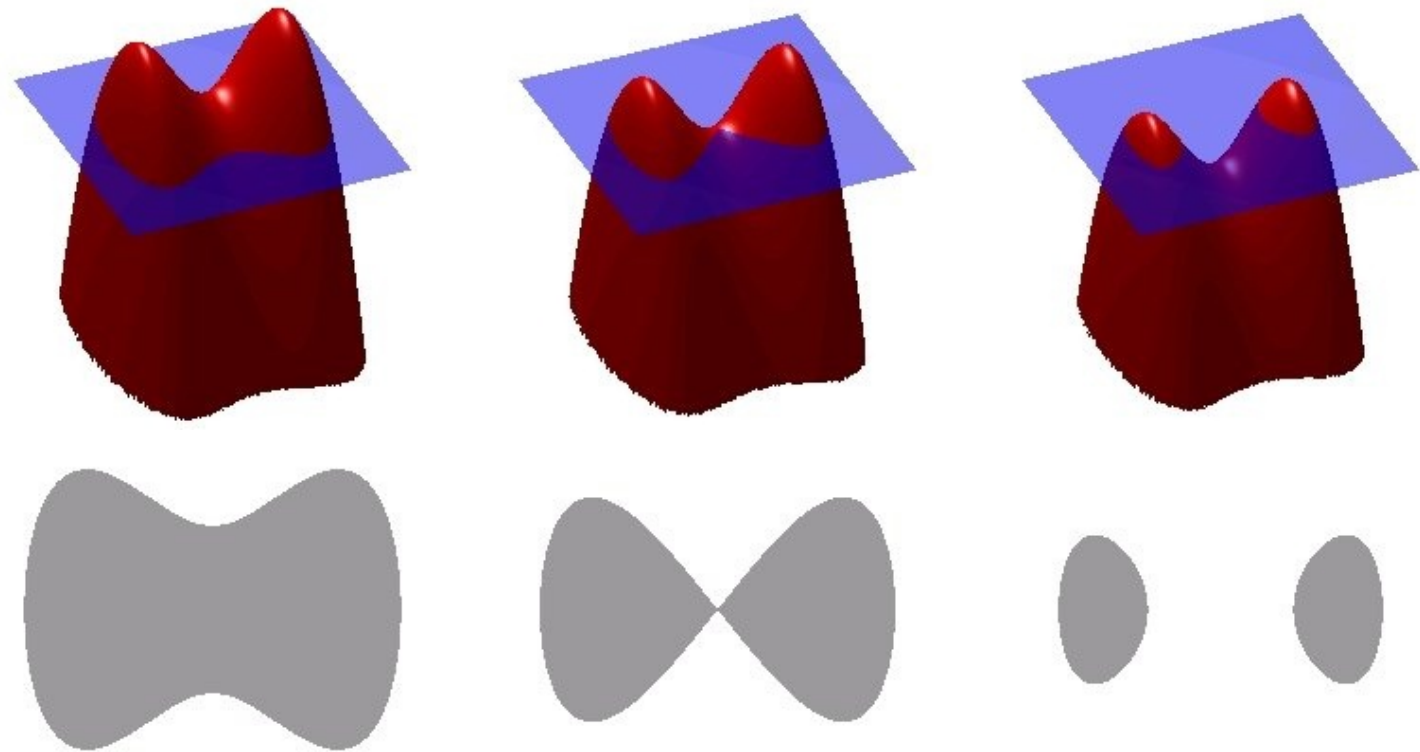
- Developed for physics simulations, to model solid/liquid interfaces that move at curvature-dependent speeds
Osher & Sethian (1988)
- Applied to images as a substitute for snakes
Caselles, Catté, Coll & Dibos (1993)
Malladi, Sethian & Vemuri (1995)
- Addresses problems with snakes:
 - sampling of snake is problematic
 - snake cannot split or merge
(if there's two objects in the image, start with two snakes)
 - snakes in higher dimensions are complex
- Simple to understand, a little harder to implement

Level sets

- Instead of defining a curve through a set of sample points, we embed the curve in a higher-dimensional space
 - for example: instead of a 1D curve in 2D, we have a 2D surface in 3D
- The curve is the set of points for which the surface crosses the 0 level



Level sets



Note:

- nothing special is required for the curve to split into two
- the function ψ is always well-behaved
- this is trivial to generalise to any number of dimensions
- the shape of the function away from the zero level set is not important
- the function is always positive inside the object, negative outside

Level sets


- The surface is modified according to a speed function

$$\gamma(t) = \{ \psi(\vec{X}, t) = 0 \} \quad \frac{\partial}{\partial t} \psi + F |\nabla \psi| = 0$$



- The speed function contains the equivalent of the internal and external forces of the snake

$$F = k(F_A + F_G)$$

advection term,
constant speed
(= balloon force)



geometry term,
curvature dependent
(= internal forces)



$$F_G = \nabla \cdot \frac{\nabla \psi}{|\nabla \psi|}$$

$$k = \frac{1}{1 + |\nabla G_\sigma \otimes I|}$$

(small at edges)

Improving level sets

- If the image gradient is weak, the curve can pass it
- Once passed this point, it cannot go back
- Solution: add a term that pulls curve towards edges
 - seems logical, considering what we learned with snakes!

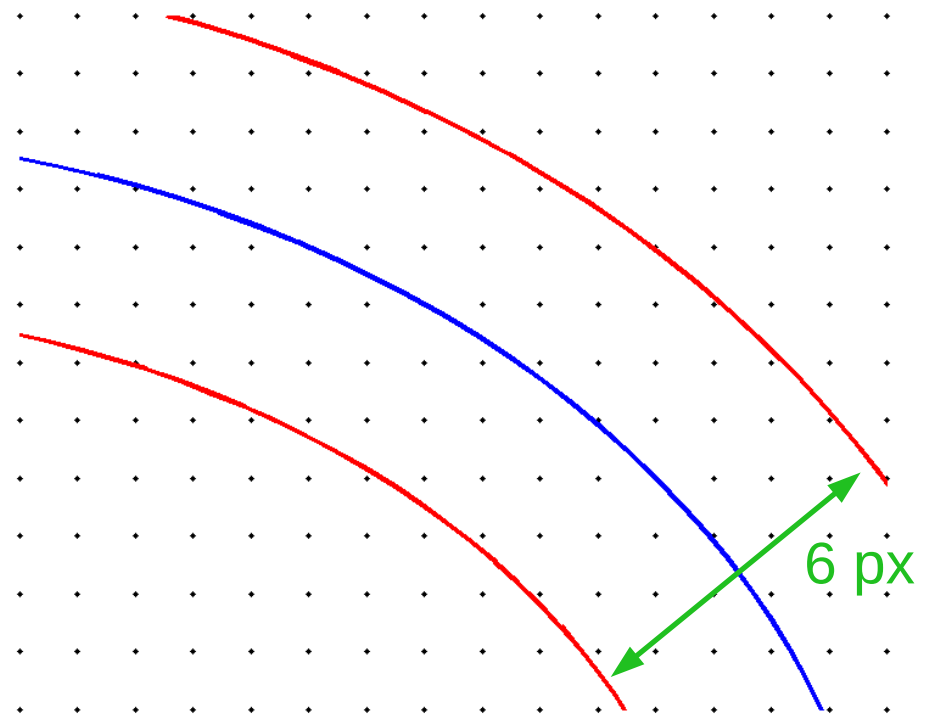
$$\frac{\partial}{\partial t} \psi + \underbrace{k \left(F_A + \nabla \cdot \frac{\nabla \psi}{|\nabla \psi|} \right) |\nabla \psi|}_{\text{old speed function}} + \underbrace{\nabla k \cdot \nabla \psi}_{\text{pulls towards edge}} = 0$$

Implementing level sets

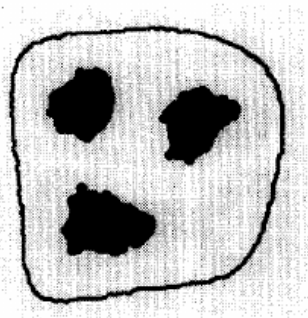
- The initial function ψ is generated from the initial closed contour using a distance function, $\psi(\vec{x}, t=0) = \pm d$ where d is the distance from \vec{x} to the contour
- The closed contour can always be recovered by looking for the zero crossings of ψ
- The function k is an appropriate speed only on the contour
 - level sets other than the zero level set will move at different speeds, which can create very large gradients in ψ
- Two solutions:
 - 1: extend the function values of k for $\psi=0$
 - 2: regularly reinitialise the function ψ from its zero crossings

Narrow band implementation

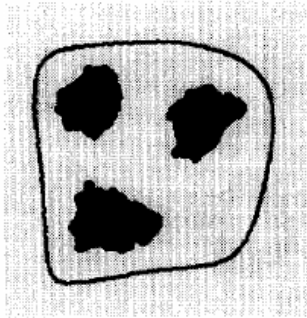
- Level sets usually implemented in a narrow band around $\psi=0$
 - this saves a lot of computation
- When the curve comes too close to the band edge:
 - reinitialise the function ψ
 - this defines a new band



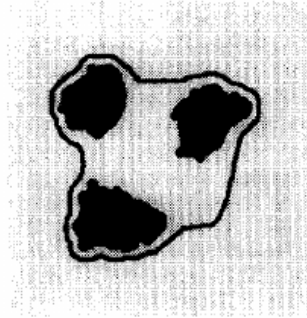
Example



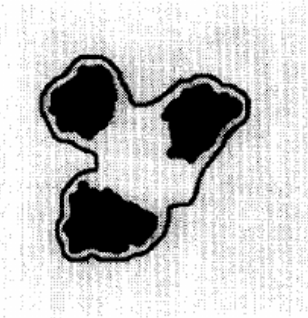
(a) $t = 0.0000$



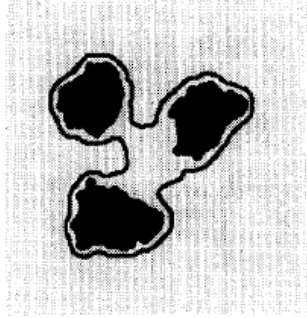
(b) $t = 0.0250$



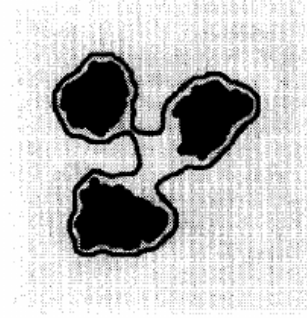
(c) $t = 0.0875$



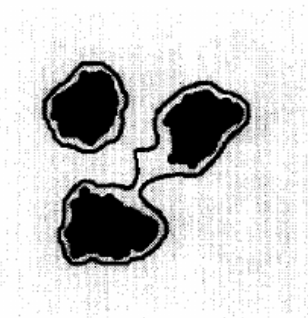
(d) $t = 0.1250$



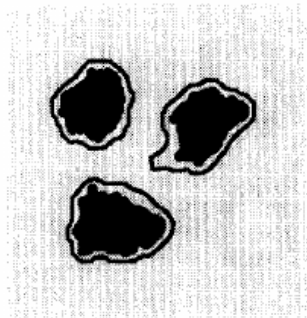
(e) $t = 0.1625$



(f) $t = 0.1750$



(g) $t = 0.1875$



(h) $t = 0.2000$



(i) $t = 0.2500$

(from Malladi et al., 1995)

Summary

- Snake – Active Contour Model
 - simple, versatile
 - lots of parameters to tweak
 - 3D extension not trivial but doable
 - one object, one snake
- ASM – Active Shape Model
 - a snake with knowledge
 - trained with a set of examples
 - robust against partial occlusions
- Level Set
 - “different way of implementing a snake”
 - n D extension trivial
 - adapts to any number of contours