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)
Segmentation II

- 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

\[ E = \int_0^1 E_{\text{int}}(\vec{\nu}(s)) + E_{\text{ext}}(\vec{\nu}(s)) \, ds \]

\[ E_{\text{int}}(\vec{\nu}(s)) = \frac{1}{2} \left[ \alpha(s)|\vec{\nu}'(s)|^2 + \beta(s)|\vec{\nu}''(s)|^2 \right] \]

- Gaussian smoothed input image

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

- "membrane" term
- "thin plate" term

(\( \alpha \) and \( \beta \) often constant)
Snakes

• Minimizing this:

\[
E = \int_0^1 \frac{1}{2} \left( \alpha |\vec{V}'(s)|^2 + \beta |\vec{V}''(s)|^2 \right) + E_{\text{ext}}(\vec{V}(s)) \, ds
\]

• through Euler-Lagrange equation yields this:

\[
\alpha \vec{V}''(s) - \beta \vec{V}''''(s) - \nabla E_{\text{ext}}(\vec{V}(s)) = 0
\]

\[
\vec{F}_{\text{int}} \quad \vec{F}_{\text{ext}}
\]

• We solve using gradient descent:

\[
\frac{\partial \vec{V}(s,t)}{\partial t} = \alpha \vec{V}''(s,t) - \beta \vec{V}''''(s,t) - \nabla E_{\text{ext}}(\vec{V}(s,t))
\]
Discretizing the gradient gradient descent equation

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

\[
\frac{\partial X_t}{\partial t} = AX_t + f_x(X_t, Y_t)
\]

\[
\frac{X_t - X_{t-1}}{\gamma} = AX_t + f_x(X_t, Y_t)
\]

\[
X_t = (I - \gamma A)^{-1}\left[ X_{t-1} + \gamma f_x(X_t, Y_t) \right]
\]

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

$X_t$ 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:
  - $\alpha$, $\beta$ and $\gamma$
  - $E_{\text{ext}}$
  - initial snake $\{X_0, Y_0\}$
  - number of iterations

- The curve needs to be sampled densely to be able to compute $\hat{\mathbf{v}}'''(s)$ accurately

- When the snake evolves, it is important to resample the curve regularly
Snakes in action

input image  - gradient magnitude  evolving snake

\( E_{\text{ext}} \)
Snakes in action

- input image
- gradient magnitude
- evolving snake

$E_{\text{ext}}$
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 $E_{\text{ext}}$, the gradient descent equation had 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

\[ \alpha \dddot{v}(s) - \beta \ddddot{v}(s) - \nabla E_{\text{ext}}(v(s)) + \kappa \vec{n}(s) = 0 \]

the unit normal vector
The balloon force

input image  - gradient magnitude  evolving snake

\[ E_{\text{ext}} \]
The balloon force

- high gradient
- low gradient
- high gradient
- high gradient
- high intensity
- high gradient

(play demo)
Static forces

\[ \vec{F}_{\text{ext}} = -\nabla E_{\text{ext}} \]

color = direction
saturation = strength
The distance force

- 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

\[
F_{\text{ext}} = -\nabla D
\]

\[
E = \text{Canny}(I)
\]

\[
D = \text{DT}(\bar{E})
\]
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

\[
E = \int \mu \left( |\nabla U_{\text{gvf}}|^2 + |\nabla V_{\text{gvf}}|^2 \right) + |\nabla E_{\text{ext}}|^2 |\vec{F}_{\text{gvf}} + \nabla E_{\text{ext}}| \, d\vec{x}
\]

smoothness term

\( \vec{F}_{\text{gvf}} = -\nabla E_{\text{ext}} \) close to edges:
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 $\vec{F}_{vfc} = |\nabla I| \otimes \vec{k}$
  - normalize vectors

\[
\vec{k} = (r + \varepsilon)^{-y} \vec{n}
\]

\[
\vec{n} = \left( \frac{-x}{r}, \frac{-y}{r} \right)
\]

\[
r = \sqrt{x^2 + y^2}
\]

$\varepsilon$ is some small value to avoid division by 0

$y \in [1,3]$

$y = 2 \Rightarrow$ gravitation!
Vector field convolution

- input image
- VFC force
- evolving snake

This is numerical inaccuracy. Increase kernel size to avoid this!
The external forces

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
Active shape models

- A snake uses application-specific knowledge by:
  - the initialization (the initial shape to be deformed)
  - the choice of parameters ($\alpha$, $\beta$, $\gamma$, $\kappa$, …)
  - 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**
- **“shoulder” shape**
- **wire curvature**

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

- The training set can be seen as $M \times (2N)$-dimensional vectors:
  \[ \tilde{x} = (x_1, y_1, x_2, y_2, \ldots, x_N, y_N)^T \]

- 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

- 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!

\[ d \hat{x}_i = \hat{x}_i - \hat{x} \]
\[ S = \frac{1}{M} \sum_{i=1}^{M} d \hat{x}_i \cdot d \hat{x}_i^T \]
\[ S \hat{p}_k = \lambda_k \hat{p}_k \]

- \( S \) is a 2Nx2N covariance matrix
- \( \lambda_k \) is the eigenvalue
- \( \hat{p}_k \) is the unit-length vector
The active shape model

- Any allowed shape is given by:
  \[ \tilde{x} = \hat{x} + P \hat{b} \]
  \[ P = (\tilde{p}_1, \tilde{p}_2, ..., \tilde{p}_K) \quad (K \leq 2N) \]
  weights to be fitted
  \[ -3 \sqrt{\lambda_i} \leq b_i \leq 3 \sqrt{\lambda_i} \]

- 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 \(\theta\)
  - use the mean shape, that is, all \(b_i = 0\)

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

- 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 $d\vec{x}_j = \kappa m\hat{n}$

$d\vec{x}$ 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 $d_s, d\theta, dx_o$ and $dy_o$ that best aligns $\hat{x}$ to $\hat{x} + d\hat{x}$

- Step 3: Find residual adjustments $d\hat{u}$ in local coordinate frame:

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

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

\[
d\hat{u} = M_{(s+ds)^{-1}, (\theta+d\theta)^{-1}}\left[ M_{s, \theta}(\hat{x} + P\hat{b}) + d\hat{x} - T(dx_o, dy_o) \right] - (\hat{x} + P\hat{b})
\]
Fitting an ASM to an image

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

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

(this is a least-squares approximation!)

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

$$\begin{align*}
    s &\leftarrow s + \omega_s d s \\
    \theta &\leftarrow \theta + \omega_\theta d \theta \\
    x_o &\leftarrow x_o + \omega_{x_o} d x_o \\
    y_o &\leftarrow y_o + \omega_{y_o} d y_o \\
    b_i &\leftarrow b_i + \omega_{b_i} d b_i
\end{align*}$$

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

the weights $\omega$ 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

\[ y = \{ \psi = 0 \} \]

\[ \psi(\vec{x}) \]

\[ \vec{v}(s) \]
Level sets

Note:
• nothing special is required for the curve to split into two
• the function $\psi$ 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
The surface is modified according to a speed function

\[ \gamma(t) = \left\{ \psi(\vec{x}, t) = 0 \right\} \]

\[ \frac{\partial}{\partial t} \psi + F |\nabla \psi| = 0 \]

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

\[ F = k (F_A + F_G) \]

\[ F_G = \nabla \cdot \frac{\nabla \psi}{|\nabla \psi|} \]

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

(small at edges)

advection term, constant speed (= balloon force)

geometry term, curvature dependent (= internal forces)
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!

\[
\frac{\partial}{\partial t} \psi + k \left( F_A + \nabla \cdot \frac{\nabla \psi}{|\nabla \psi|} \right) |\nabla \psi| + \nabla k \nabla \psi = 0
\]

- Old speed function.
- Pulls towards edge.
Implementing level sets

- The initial function $\psi$ 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 $\psi$.

- 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 $\psi$.

- Two solutions:
  - 1: extend the function values of $k$ for $\psi=0$.
  - 2: regularly reinitialise the function $\psi$ 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 $\psi$
  - this defines a new band
Example

(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 Surface 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