

Segmentation

Segmentation : the problem

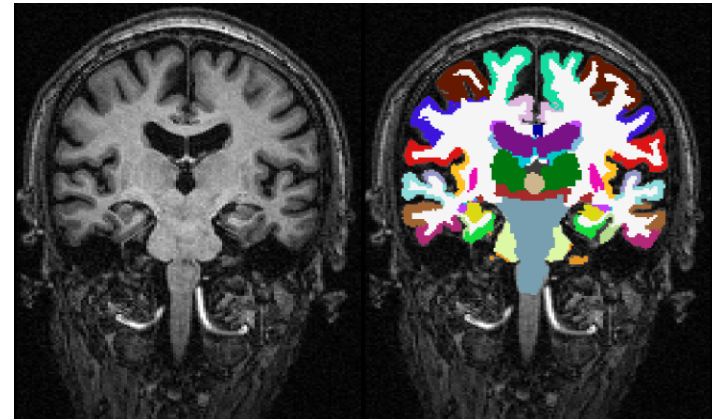
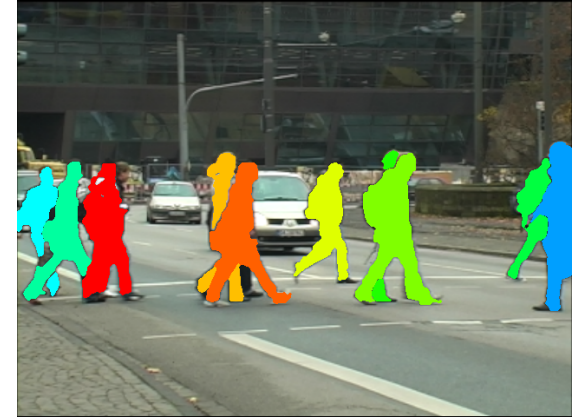
Identifying entities in the image, e.g. objects :

- grouping pixels into segments
- crucial and basically unsolved step



Segmentation : importance

- Understanding scenes
- Industrial inspection
- Disease diagnosis
-



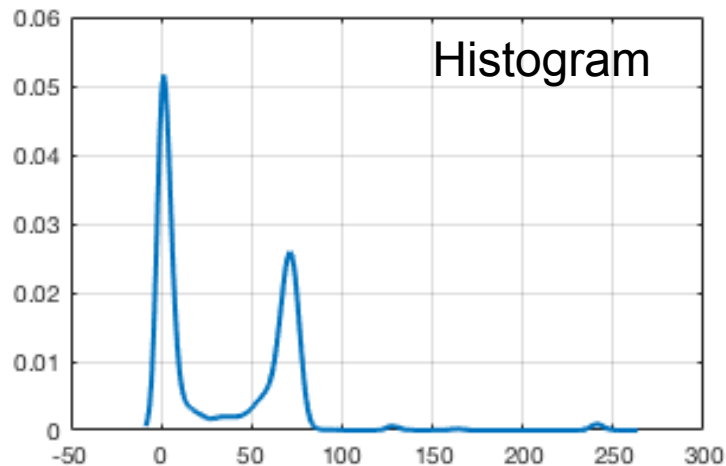
Segmentation : Outline

- Thresholding
- Edge based
- Region based
- Statistical Pattern Recognition based

Thresholding : basic idea

For high contrast between object(s) and background, determine intensity threshold that defines 2 pixel categories : object and background

Example image



Threshold = 5



Threshold = 25



Threshold = 50

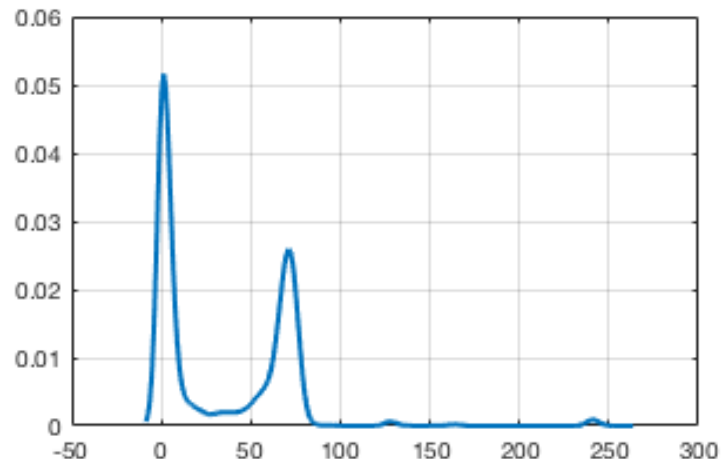


Threshold = 70

Thresholding : threshold selection

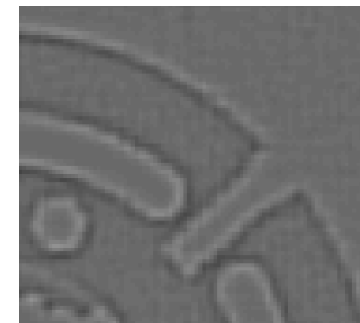


Histogram



Alternatives:

1. If intensities of objects are known → easy
2. From histogram, take the minimum between two peaks
3. For known size (e.g. for industrial application),
increase threshold until reaching a predefined area
4. Maximize sum of gradients at pixels with threshold intensity
5. Low gradient magnitude areas
6. Use regions with high response to
Laplacian filter – points around the edge



Thresholding: Otsu criterion

We want large areas of low variance.
Hence, minimize within-group variance
weighted by group size p



Group 1
 $I > \text{threshold}$
 p_1, σ_1^2



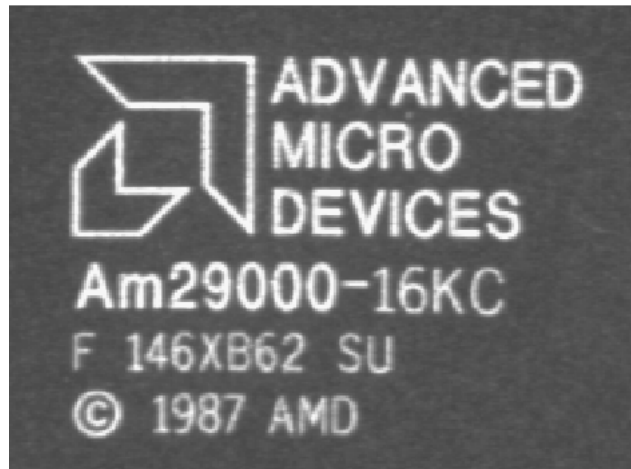
Group 2
 $I \leq \text{threshold}$
 p_2, σ_2^2

determine threshold to minimize $p_1\sigma_1^2 + p_2\sigma_2^2$

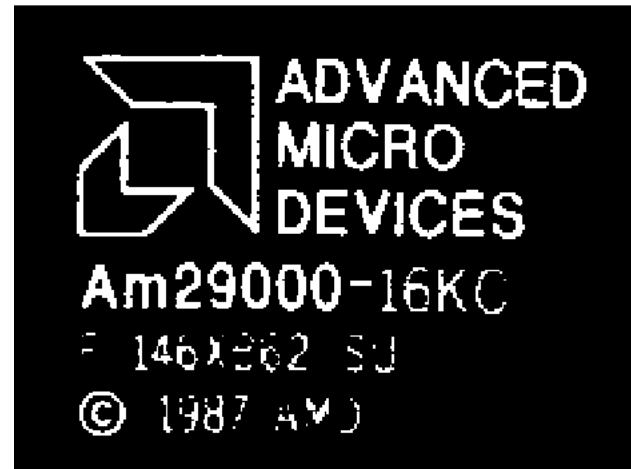
Otsu threshold = 35



Thresholding: Global vs. Local Otsu



Image



Fixed threshold

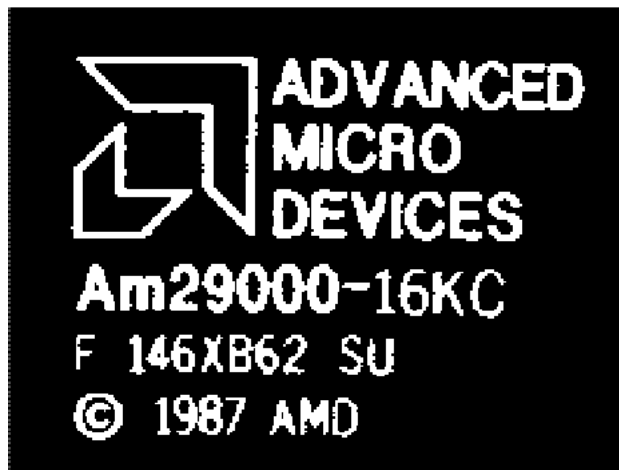
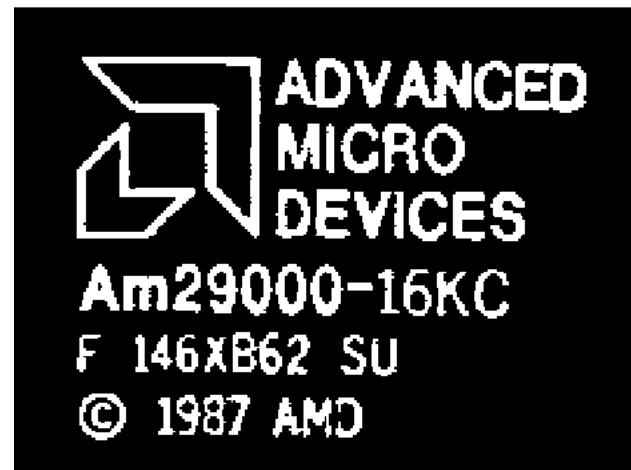


Image specific
Otsu threshold



Local Otsu
threshold

With noise

Pixels may fall on the wrong side of the threshold

Threshold at 35:



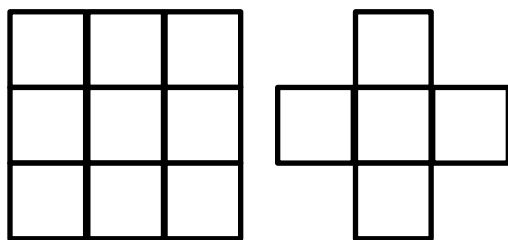
Potential solution: **Mathematical morphology**
to enhance binary images,
e.g., remove isolated islands or holes

Mathematical Morphology: Basics

- Operations on binary images
- Shift-invariant
- Non-linear
- Based on pixel neighborhood defined by structural elements
- View binary image as a set
- Two main operations



Binary Image: A



Binary structural
element: B

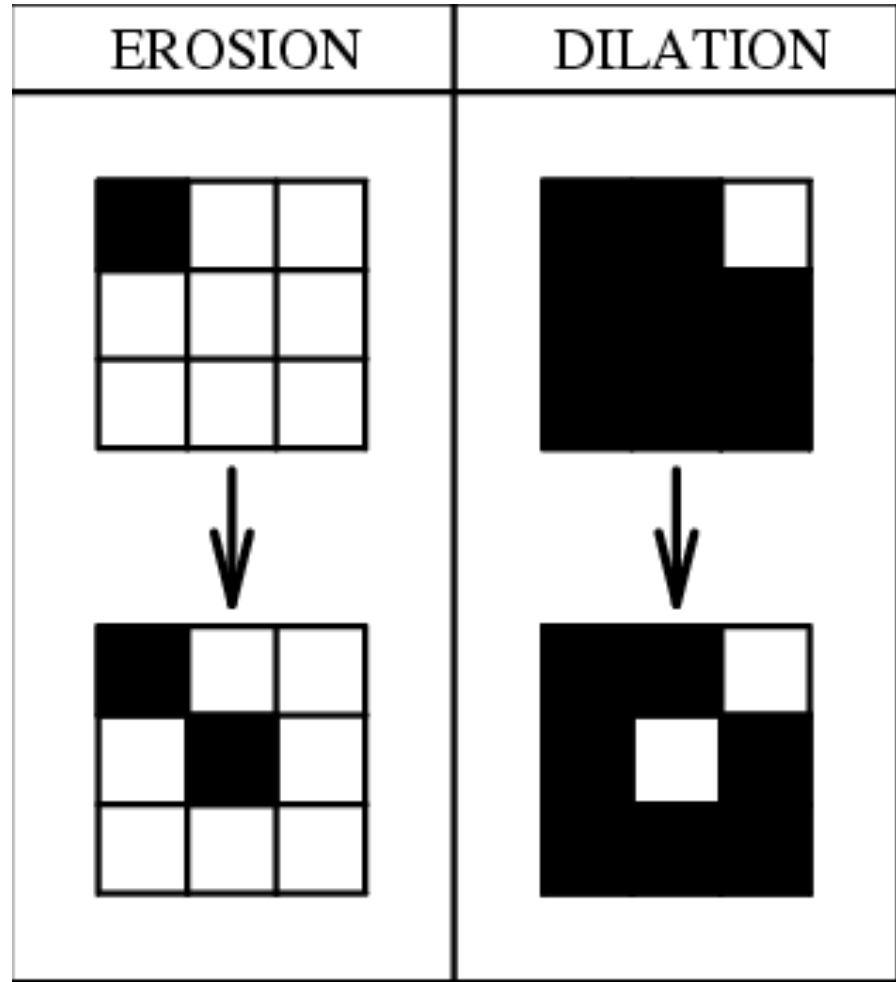
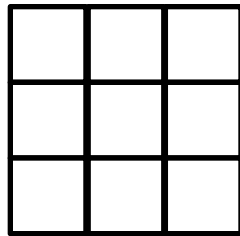
Dilation

$$A \oplus B = \{x | B_x \cap A \neq \emptyset\}$$

Erosion

$$A \ominus B = \{x | B_x \subseteq A\}$$

Simple Illustration

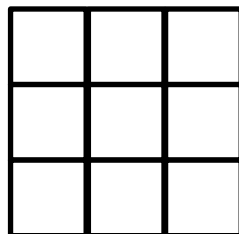


$$A \ominus B = \{x | B_x \subseteq A\} \quad A \oplus B = \{x | B_x \cap A \neq \emptyset\}$$

Erosion and Dilation: Example

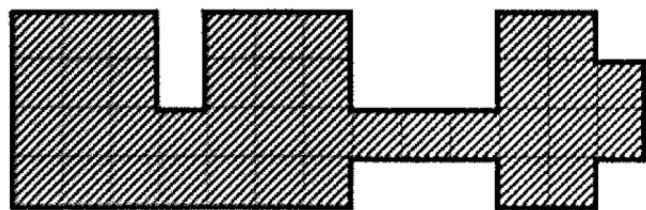


Erosion

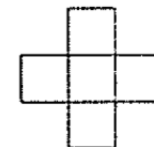


Dilation

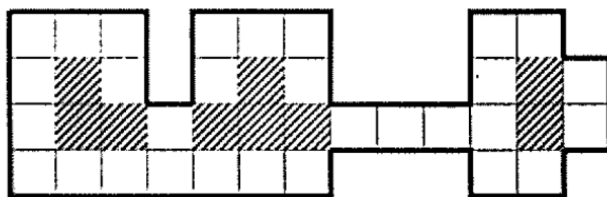
Concatenation of basic operations: Opening and closing



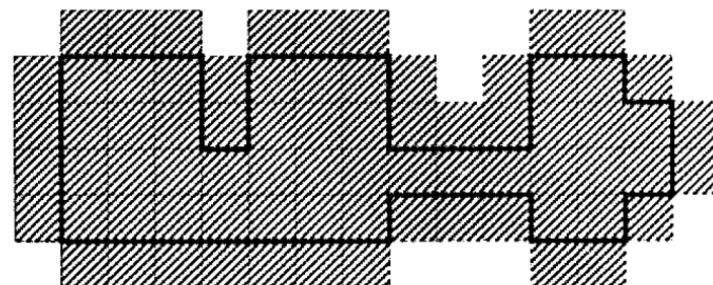
original image structure



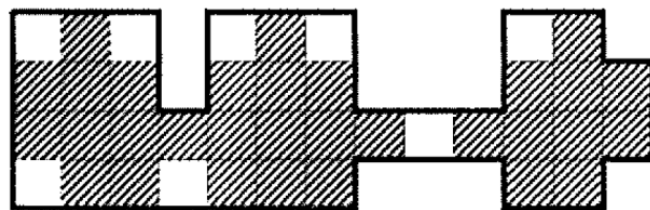
structuring element



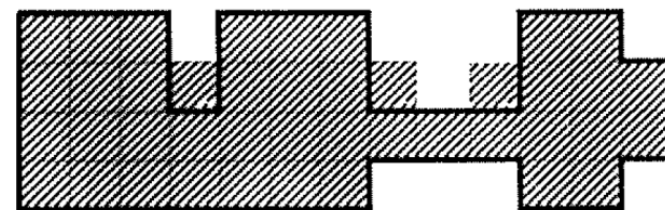
Erosion



Dilation

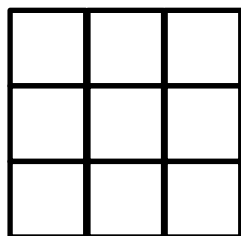


Opening $(A \ominus B) \oplus B$



Closing $(A \oplus B) \ominus B$

Opening and closing: Example



Opening
(mostly removes islands)



Closing
(mostly removes holes)

Binary enhancement as rank order operators

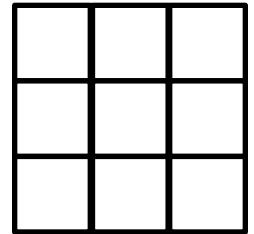
New intensity based on rank-ordered neighborhood values

Enables extension to gray level images



$$i_1 \leq i_2 \leq \dots \leq i_N$$

$$i_t = f_t(i_1, \dots, i_N)$$



Erosion	$i_t = i_1$
Dilation	$i_t = i_N$
Median filtering	$i_t = i_{N/2}$

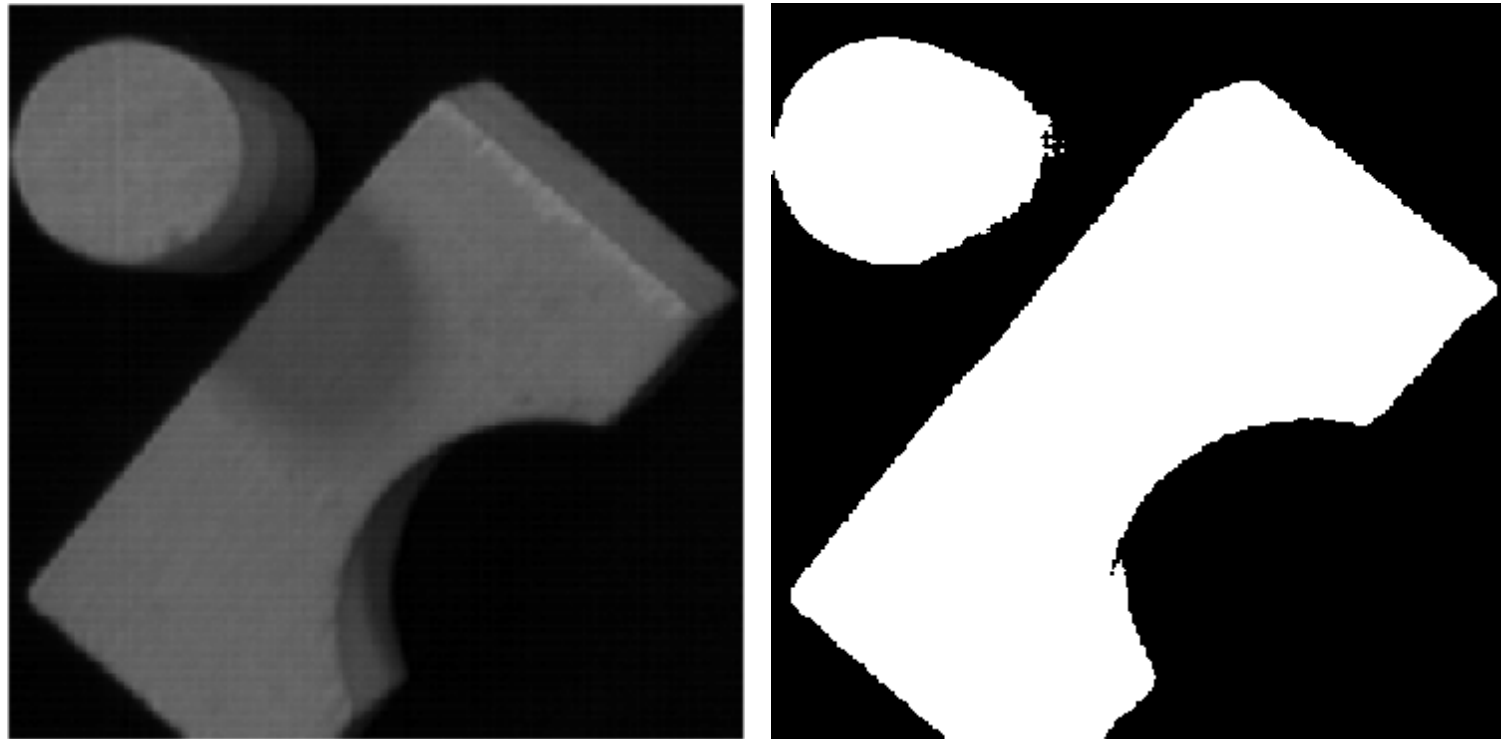
Important difference with convolution :

non-linearity

Binary enhancement : Remarks

- 1. Erosion + dilation (opening)
Dilation + erosion (closing)
- 2. Use the same structural element for both steps
- 3. It is a post-processing approach
(It has many alternatives that enforce neighborhood consistency during segmentation)
- 4. Reminder : median filtering very useful and commonly used as edge preserving smoothing

Multiple objects: Connected components



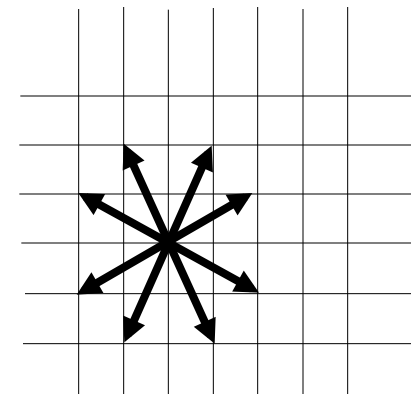
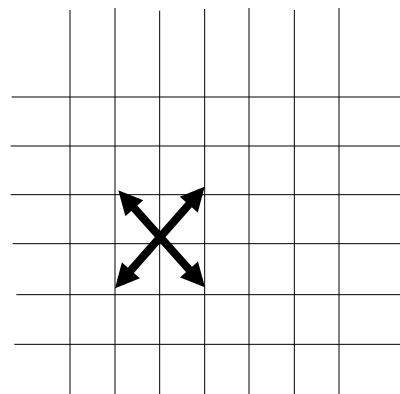
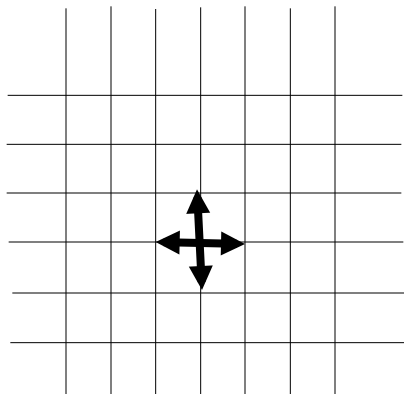
We would like to separate these objects

Connected component analysis

What does it mean to be a connected object?

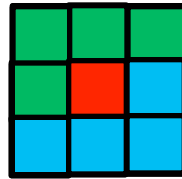
Discrete image space: Neighbourhood on Cartesian image raster

- Pixels connected through neighbourhood chain
- Connected component:
if all its pixel pairs are connected through a chain of pixels all within the same component
- Defined by the pixel neighbourhood structure
- There is no unique definition
- 4- and 8-connectivity the most popular
- There are other possibilities



Single-pass connected-component labeling

- Scanning the image line-by-line (TV scan) enforces an (artificial) causality
- At every pixel (**red**) its neighbourhood is divided into past (**green**) and future (**blue**)



- Label **red**, considering all labels of past pixels. If
 - No label found: start a new label
 - 1 label found: copy it
 - >1 labels found: note their equivalence
- At the end, co-label equivalent components (connected but initially labeled as different)

Computer Vision



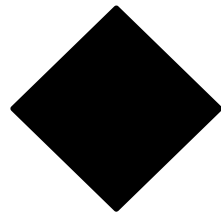
Topology induced (distance) metrics

- Depends on the chosen definition of image topology and neighborhood connectivity
- e.g., D_4 (Manhattan) and D_8 distances
- How to calculate between $P(i,j)$ and $Q(k,l)$?

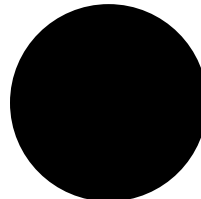
$$D_4(P,Q) = |i - k| + |j - l|$$

$$D_8(P,Q) = \max(|i - k|, |j - l|)$$

- “Discs” (equidistant regions) in D_4 and D_8



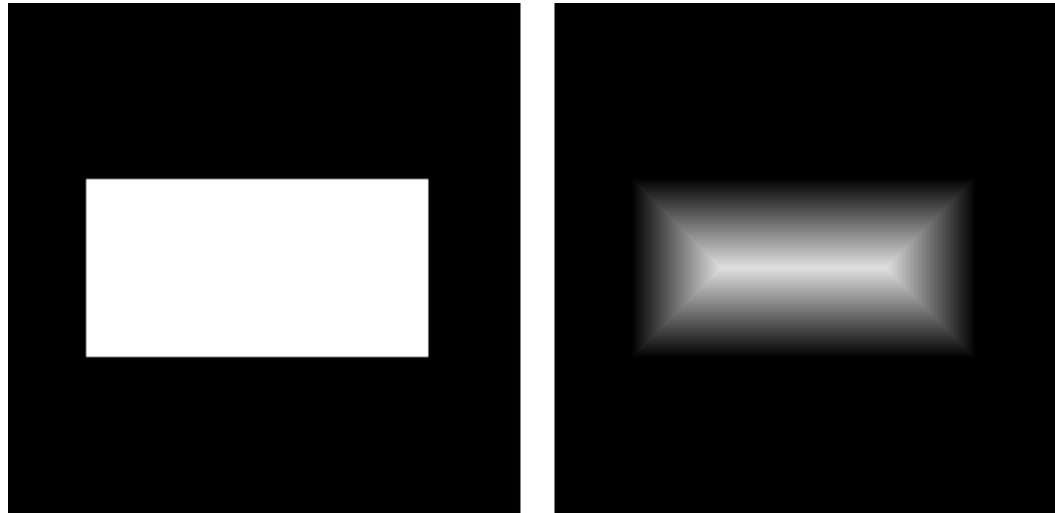
- Euclidean



does not conform with any discrete neighbourhood

Distance calculation

- Distance transformation: “distance map” based on distance propagation along neighbourhoods



- Euclidean distance map
 - True implementation is very cumbersome
 - Approximations are possible

Thresholding : remarks

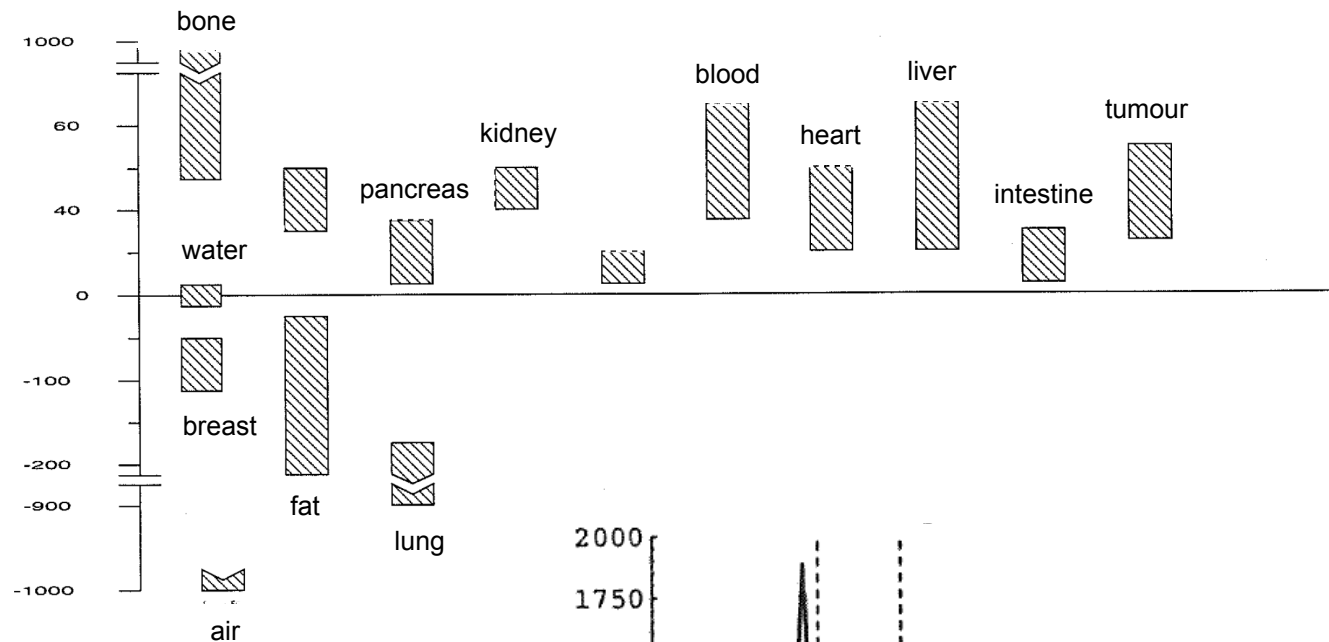
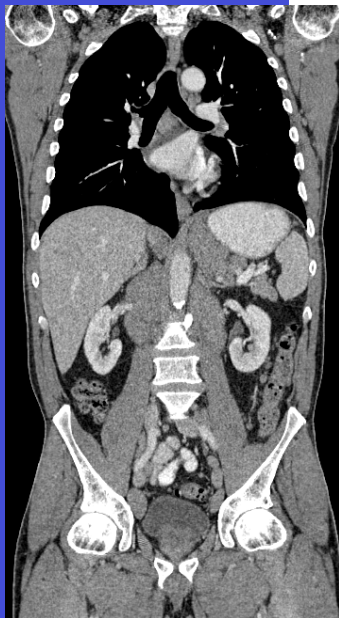
- ❑ Threshold advantages:
 1. Serious bandwidth reduction
 2. Simplification for further processing
 3. Availability of real-time hardware

- ❑ Generally it won't provide a satisfying segmentation

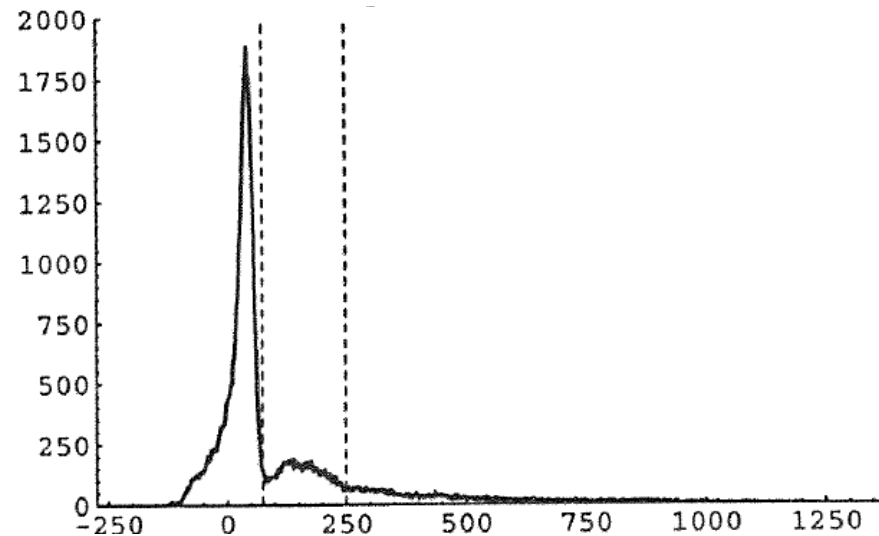
- ❑ Pixel-by-pixel decision
 - ignores neighbouring pixels
 - structural information lost

Thresholding has limitations

X-ray attenuation is tissue dependent



- strong overlap in intensity ranges
- thus, limited separability on histogram



Segmentation : Outline

- Thresholding
- **Edge based**
- Region based
- Statistical Pattern Recognition based

When thresholding is not enough, Edges can help

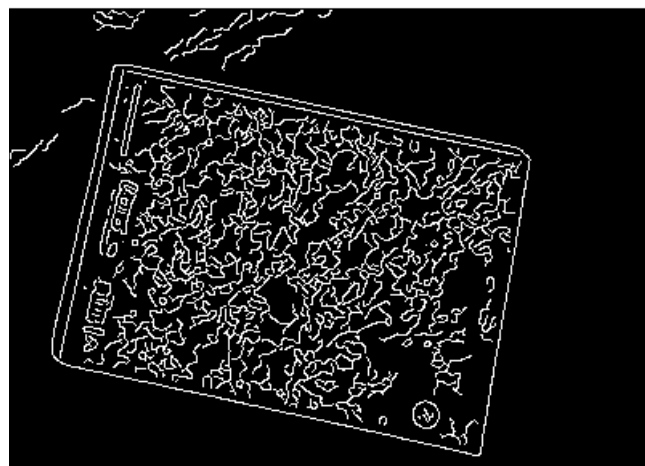
Identifying boundaries between different areas / objects



Image



After thresholding with Otsu criteria



Edge Detection with Canny

Edge linking techniques

- 1. Hough Transform: for predefined shapes
- 2. Elastically deformable contour models
Snakes: generic shape priors
- 3. Many other methods for grouping
combination with user interaction
(dynamic path search – in the script)

Hough transform : principle

Uses parametric shape models to extract objects in lower dimensional spaces

Instead of testing every possible position and orientation for the shape, each pixel is visited while voting for all shapes it may belong to

The simplest example: straight lines

Many further possibilities, like circles, ellipses and generalizations to other shapes



Hough transform : straight lines

Suppose we would like to detect straight lines
e.g. straight object outlines
in all possible positions and orientations.

For general shapes : 3 degrees of freedom

Straight lines, however, remain invariant
under translation along itself

Hence, the image projection of a straight line
is fully characterized by 2 parameters

Hough transform : straight lines

We write the equation of a straight line as

$$y = ax + b$$

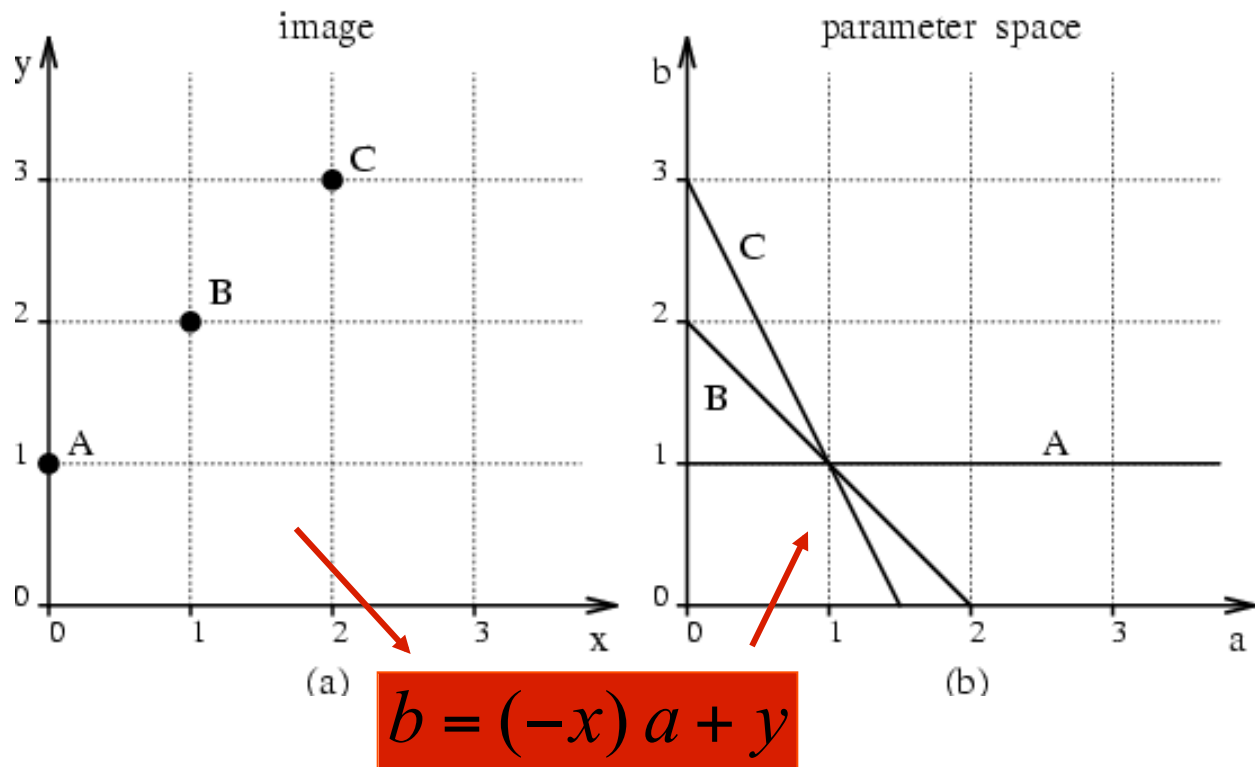
Fixing a point (x,y) , all lines through the point :

$$b = (-x) a + y$$

The Hough transform :

- 1. Inspect all points of interest
- 2. For each point draw the above line in (a,b) - parameter space

Hough transform : straight lines

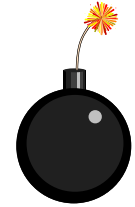


implementation :

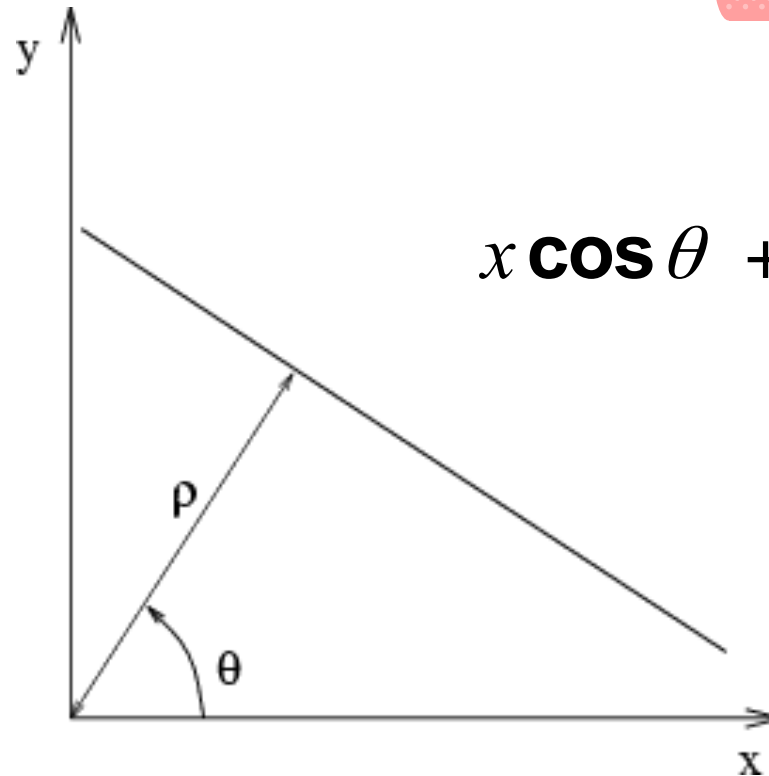
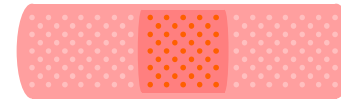
1. the parameter space is discretised
2. a (weighted) counter is incremented at each parameter cell where the lines pass
3. At the end, peaks are detected

Hough transform : straight lines

problem : unbounded parameter domain
vertical lines require infinite a



alternative representation:

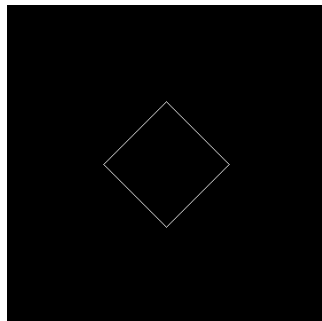


$$x \cos \theta + y \sin \theta = \rho$$

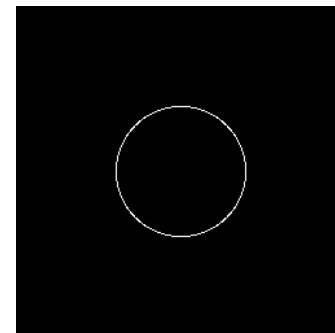
Each point will add a sinusoidal function in the (ρ, θ) parameter space

Hough transform : straight lines

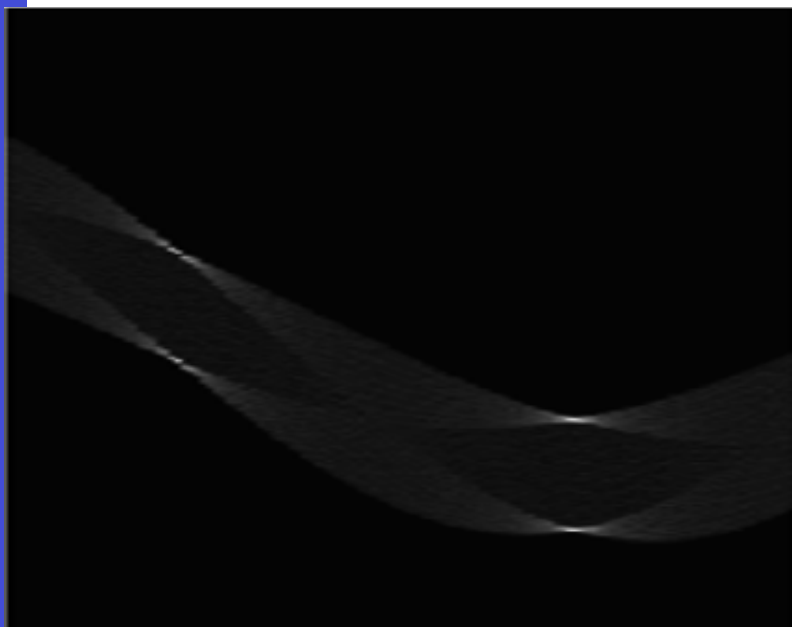
Square :



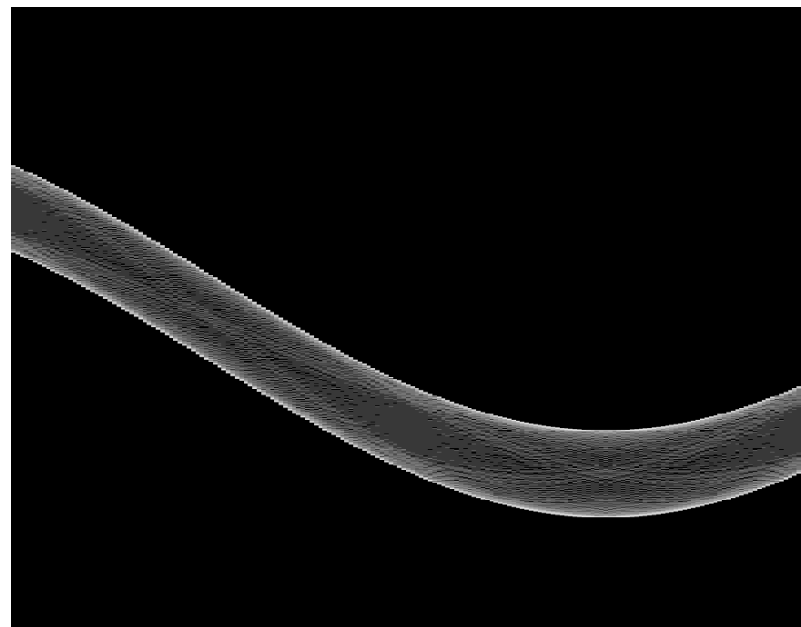
Circle :



ρ

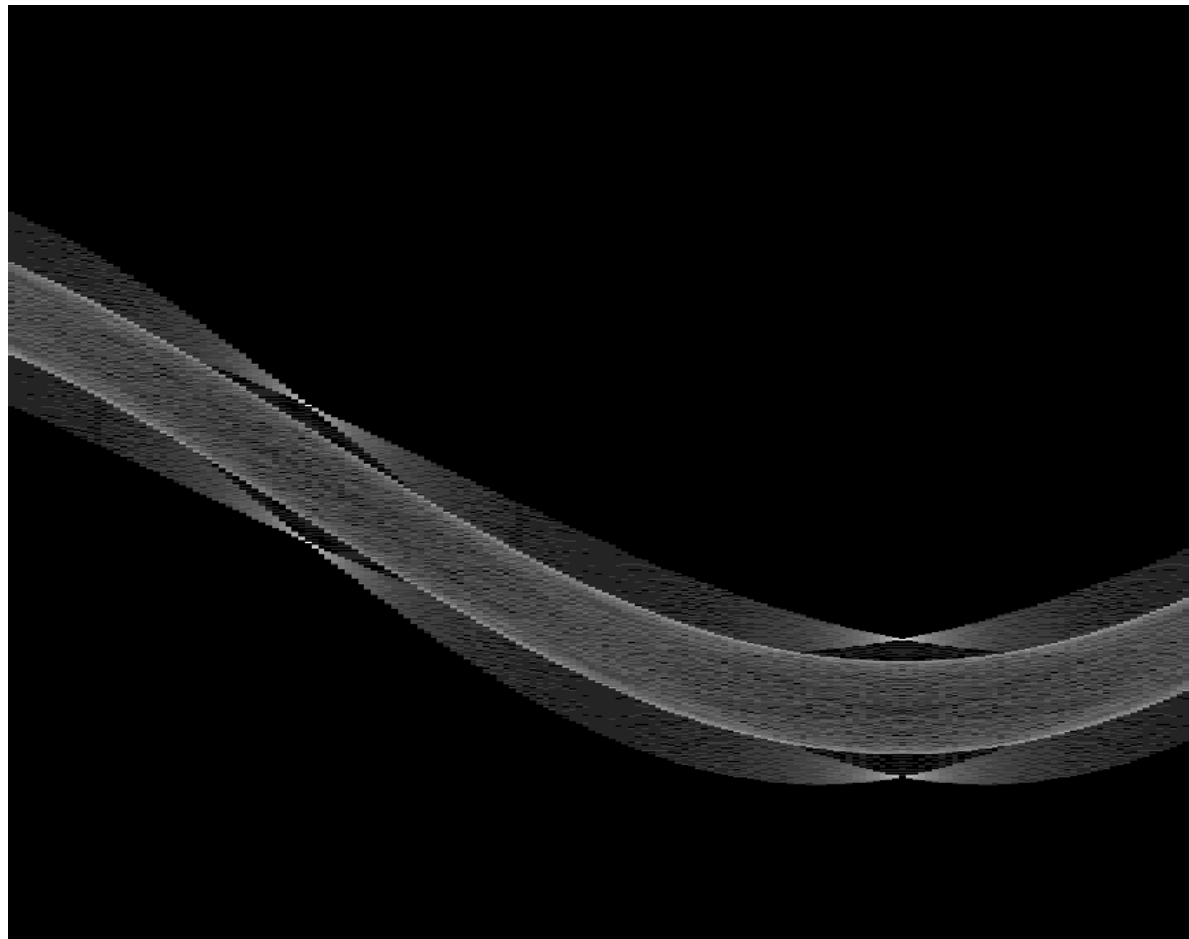
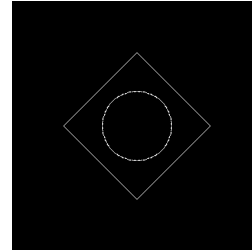


θ

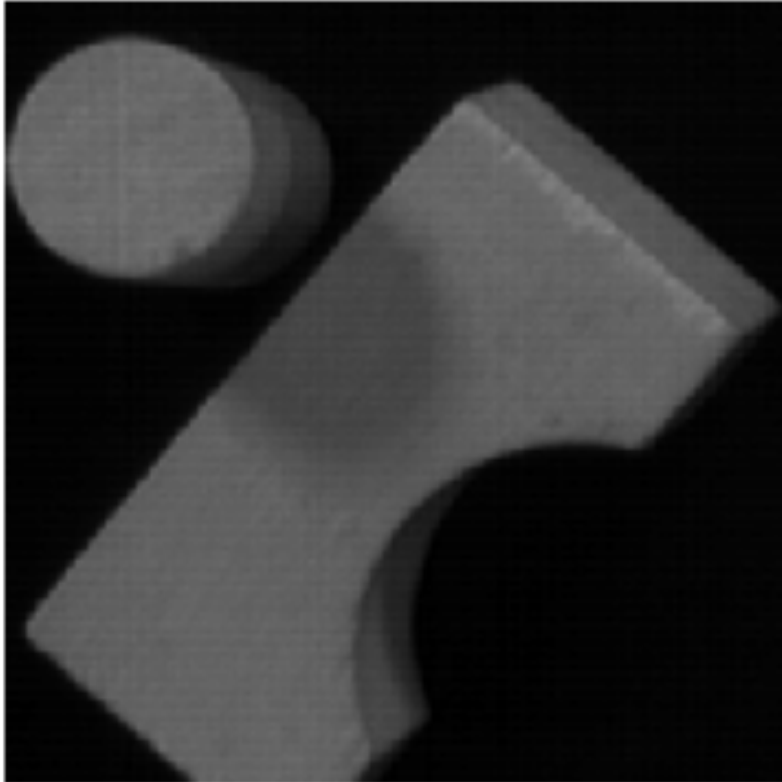


Hough transform : superposition

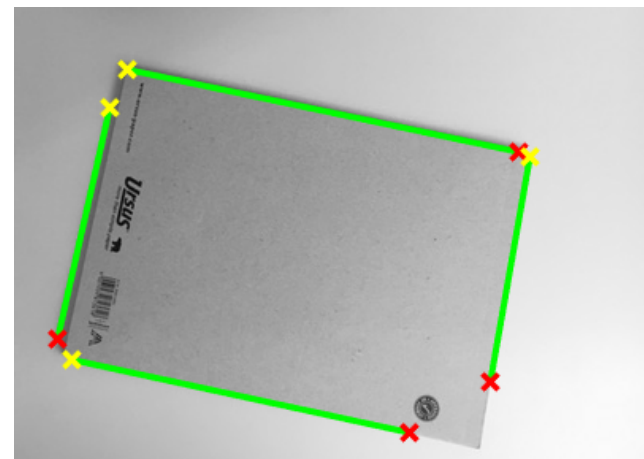
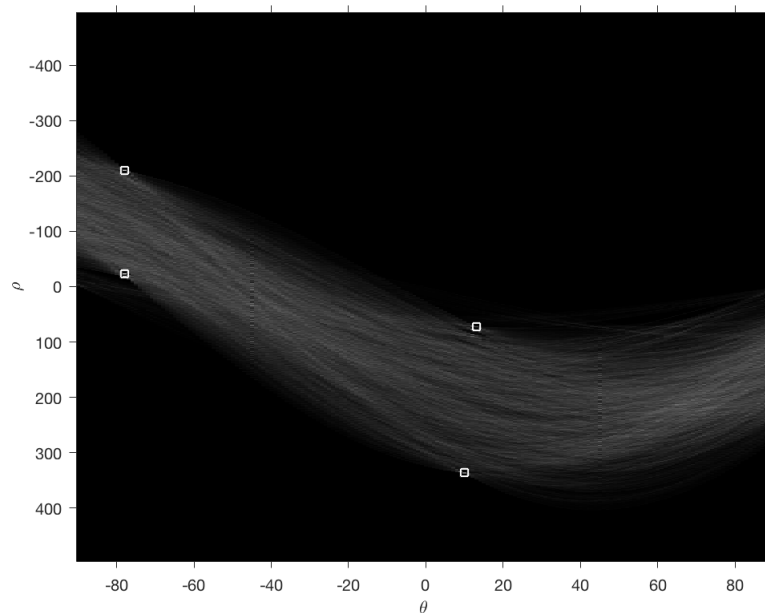
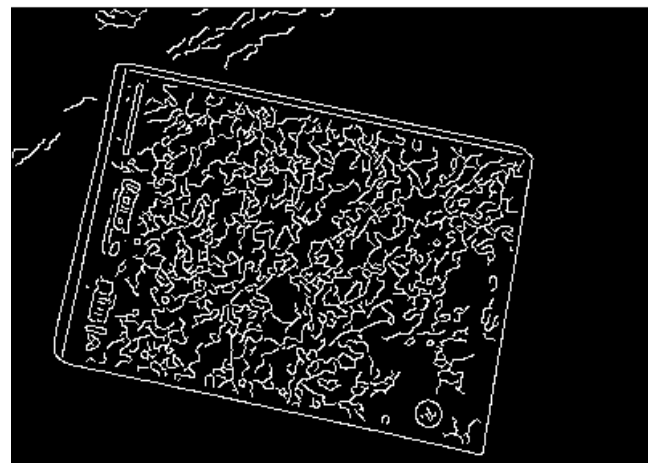
Combined image:



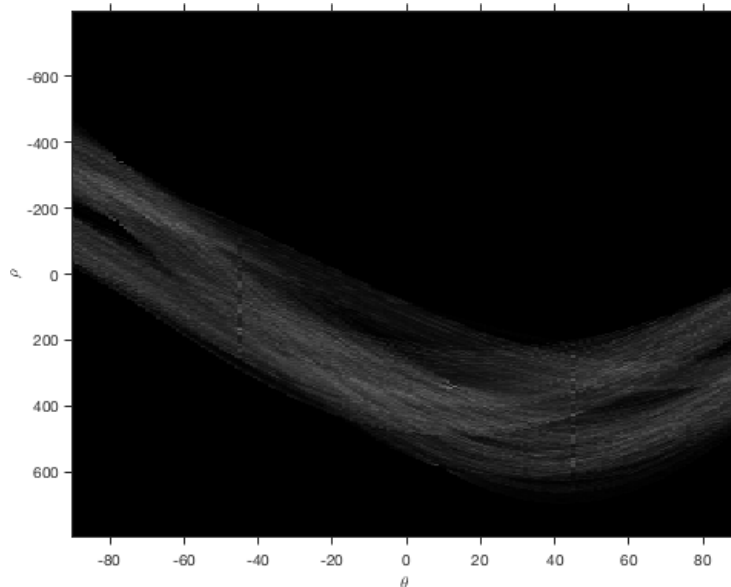
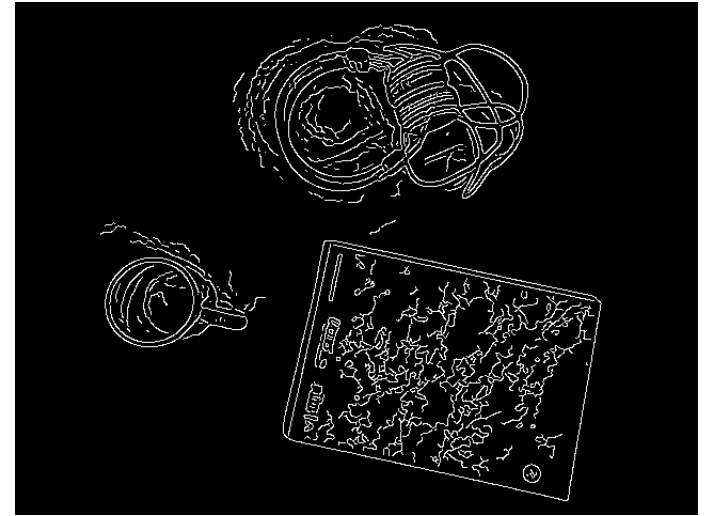
Hough transform : straight lines



Hough transform : detecting the book



Hough transform : works when other objects are present

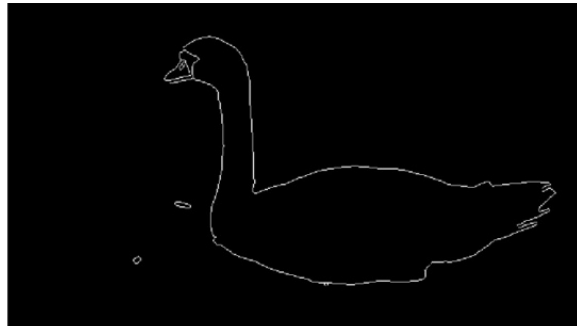


Hough transform : remarks

- ❑ 1. time consuming
- ❑ 2. robust, to noise in the image, ...
- ❑ 3. “good” peak detection is nontrivial
- ❑ 4. Robustness of peak detection is increased by weighting contributions (e.g in the examples weighting with intensity gradient magnitude)
- ❑ 5. Ambiguities possible – if similar objects are close by...

Edges vs. boundaries

Edges are useful to infer shape and occlusion, e.g.



Here the raw edge
output is not so bad



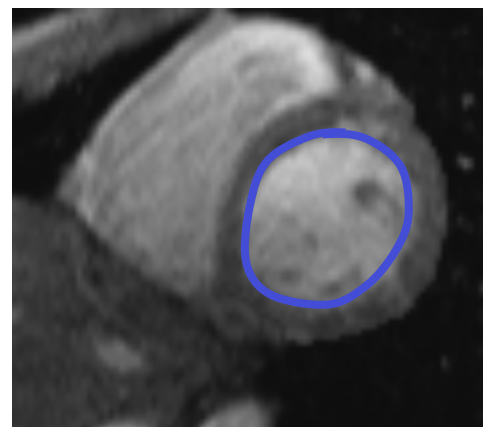
But, quite often the boundaries of
interest are fragmented, and
we have a set of “cluttered” edges

Active contour models: **Snakes**

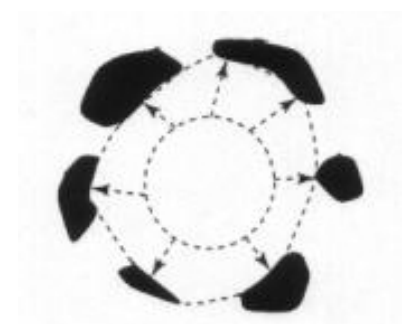
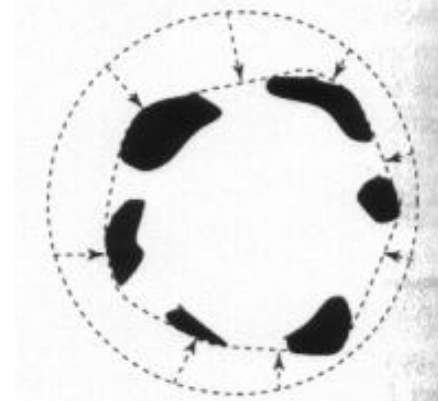
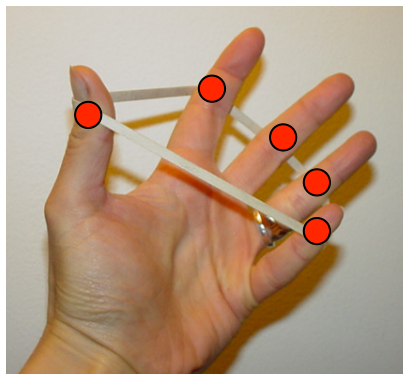
[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Given: initial contour (model) near desired object

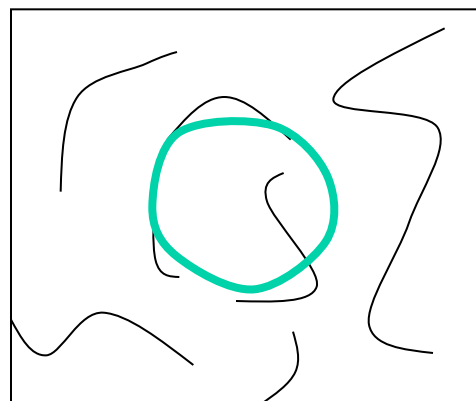
Goal: evolve the contour to fit the object boundary



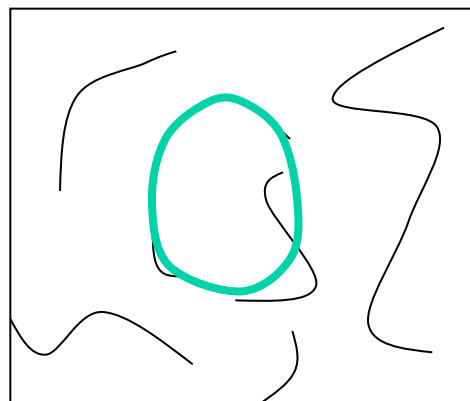
Intuition: an elastic (rubber) band wrapping around structures to cover / fill-in missing parts



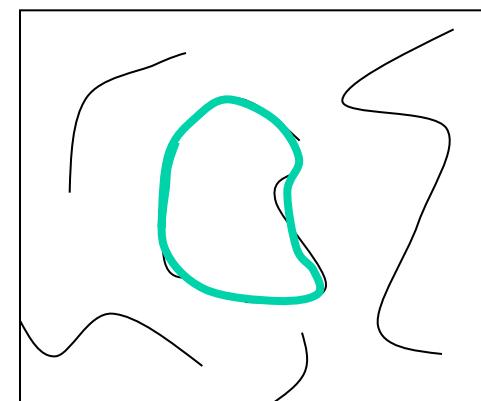
Snakes: Idea



initial



intermediate



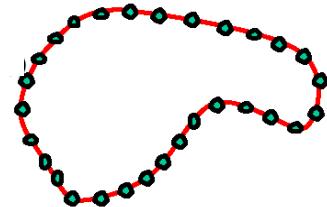
final

How to adjust (iteratively refine) an elastic band for a particular purpose, e.g., to fit high image gradients

- Define an energy function that says how good any configuration is
- Seek next configuration that minimizes energy

Snakes energy function

The total energy of the current snake is:



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

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

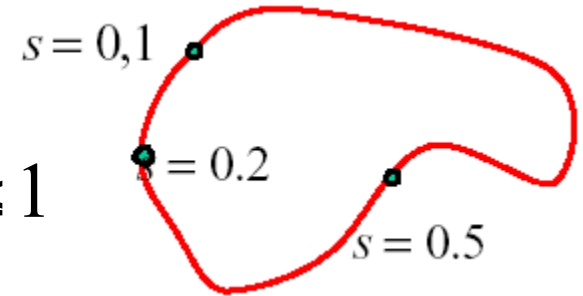
External energy (image energy): encourage contour to fit interesting image structures, e.g. edges

A **good** fit between the current snake and the target shape in the image will yield a **low** energy

Parametric curve representation

Continuous case:

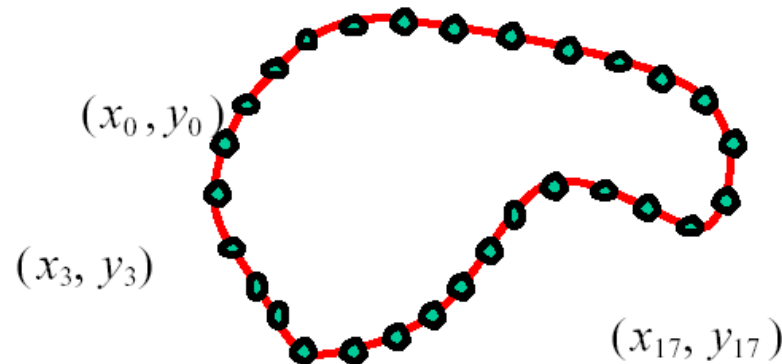
$$\mathbf{v}(s) = (x(s), y(s)) \quad 0 \leq s \leq 1$$



For numerical computation on the image

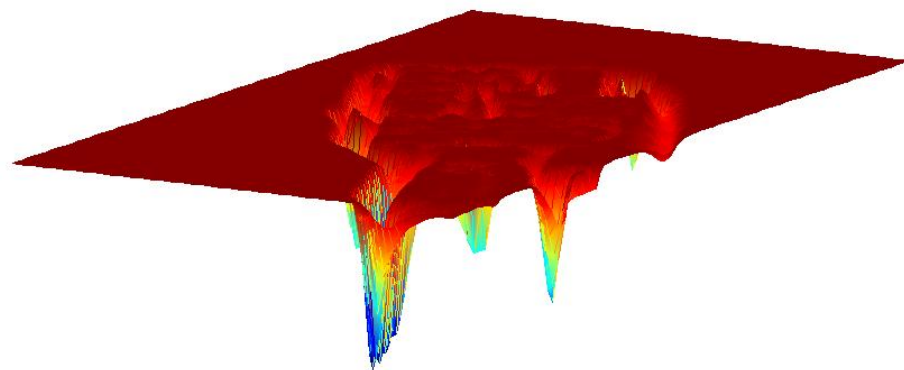
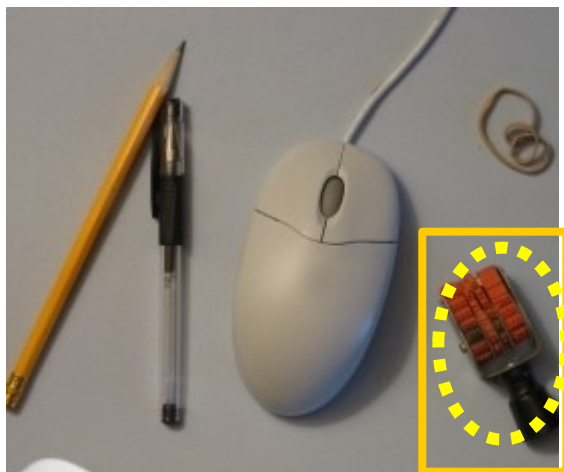
Discretization by a set of n points:

$$\mathbf{v}_i = (x_i, y_i) \quad i = 0 \dots n - 1$$



External (image) energy

- Measure how well the curve matches the image data
- Attracts the curve toward interesting image features
 - Edges, lines, etc.



- (Magnitude of gradient)

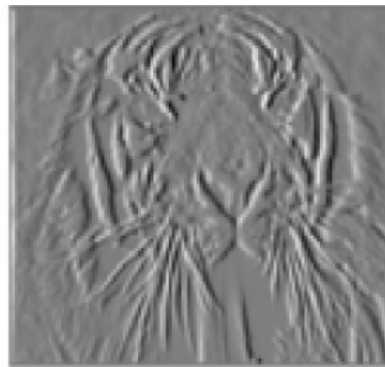
$$- \left(G_x(I)^2 + G_y(I)^2 \right)$$

Defines how the image (edges) affect rubber band

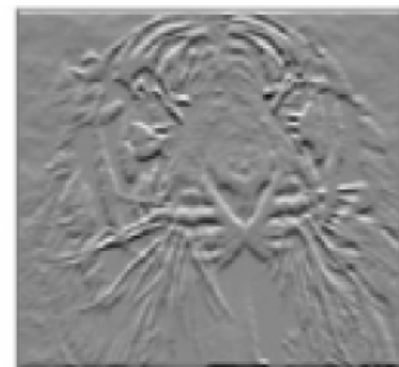
Think of it as gravitational pull towards regions of,
e.g., high contrast

External (image) energy

- Image $I(x,y)$
- Directional derivatives



$G_x(x, y)$



$G_y(x, y)$

- External energy at a point $\mathbf{v}(s)$ on the curve:

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

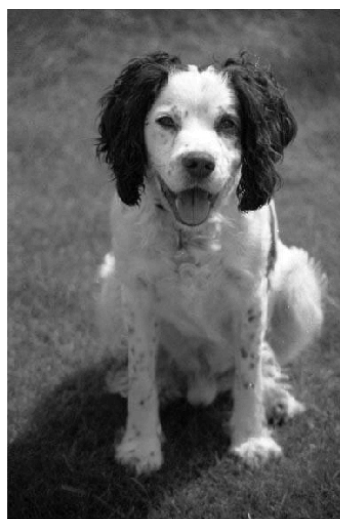
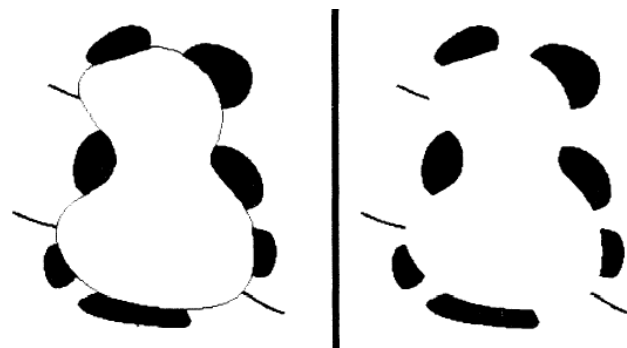
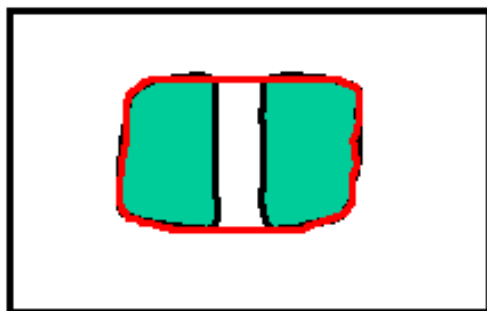
$$E_{external} = \int_0^1 E_{external}(\mathbf{v}(s)) ds$$

- External energy for the curve on discrete image

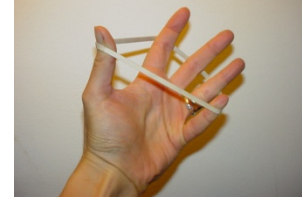
$$E_{external} = - \sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

Internal energy: intuition

Given that typical shapes smoothly continuous boundaries, we thus wish to favor smooth contours with low curvature (to fill-in missing info)



Internal energy



A common choice: Deformation Energy

- The more stretch and bend, the larger this energy value is
- Weights α and β adjust influence of each component

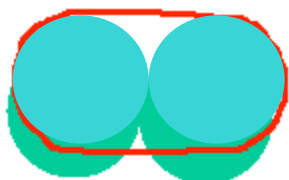
$$E_{internal} = \int_0^1 E_{internal}(v(s)) ds$$

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



Models elasticity
Penalizes tension
(inhibits stretch)

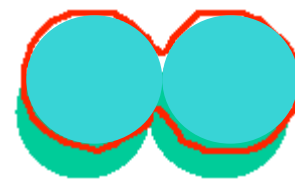
Models stiffness
Penalizes curvature
(inhibits bending)



β : large



medium



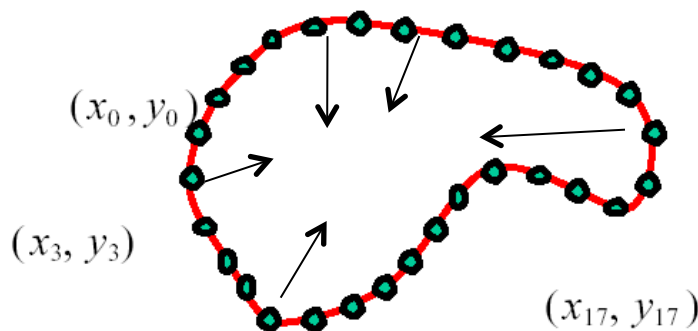
small

Internal energy: Discretization

$$\frac{d\mathbf{v}}{ds} \approx \mathbf{v}_{i+1} - \mathbf{v}_i \quad \frac{d^2\mathbf{v}}{ds^2} \approx (\mathbf{v}_{i+1} - \mathbf{v}_i) - (\mathbf{v}_i - \mathbf{v}_{i-1}) = \mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}$$

$$E_{internal} = \sum_{i=0}^{n-1} \alpha \|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2 + \beta \|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$$

Elasticity, Tension
Stiffness, Curvature



First-term prefers shorter curves.
Problem: This encourages a closed curve to shrink, eventually, to a cluster of coincident points

Possible remedy: adjusting energy term

$$E_{internal} = \sum_{i=0}^{n-1} \alpha (\|\mathbf{v}_{i+1} - \mathbf{v}_i\| - 1)^2 + \beta \|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$$

Encourages equal spacing but makes optimization harder

Energy minimization

Several methods proposed to fit snakes, including methods based on :

- Partial Differential Equations (PDEs)
- Greedy search
- Dynamic programming (for 2D snakes)

Energy minimization through PDEs

Energy to minimize in the continuous case:

$$\frac{1}{2} \int_0^1 \left(\alpha(s) \left\| \frac{\partial v}{\partial s} \right\|^2 + \beta(s) \left\| \frac{\partial^2 v}{\partial s^2} \right\|^2 \right) ds - \int_0^1 P(v) ds$$

Deformation energy

Image energy
(gradients)

Variational calculus \rightarrow Euler-Lagrange differential equation

$$-\frac{\partial}{\partial s} \left(\alpha(s) \frac{\partial v}{\partial s} \right) + \frac{\partial^2}{\partial s^2} \left(\beta(s) \frac{\partial^2 v}{\partial s^2} \right) = -\vec{\nabla} P(v)$$

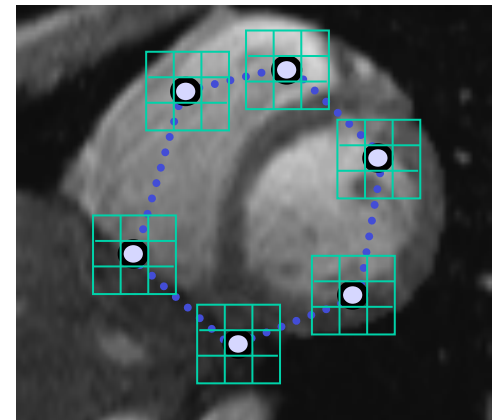
$$-\alpha v_{ss} + \beta v_{ssss} = -P_v$$

In an iterative scheme, use image gradients at curve positions in the previous iteration $t-1$, and hence solve:

$$-\alpha v_{ss}^t + \beta v_{ssss}^t = -P_v \Big|_{v=v^{t-1}} \quad \longrightarrow \quad \mathbf{K}v^t = -P_v \Big|_{v=v^{t-1}}$$

Greedy energy minimization

- For each point, place a search grid (e.g., 5x5) of discrete positions around it and pick the location where the energy function is minimal
- Keep applying this and circling around the curve
- Stop when a predefined number of points have not changed in the last iteration, or after max number of iterations
- Remarks:
 - Individual decisions are not globally optimal
 - Convergence not guaranteed



Energy minimization: dynamic programming

Snake energy can be rewritten as a sum of

- * unary potentials for individual points and
- * interaction potentials between pairs of points

$$E_{total}(\mathbf{v}_0, \dots, \mathbf{v}_{n-1}) = \sum_{i=0}^{n-1} U_i(\mathbf{v}_i) + \sum_{i=0}^{n-2} P_i(\mathbf{v}_i, \mathbf{v}_{i+1})$$

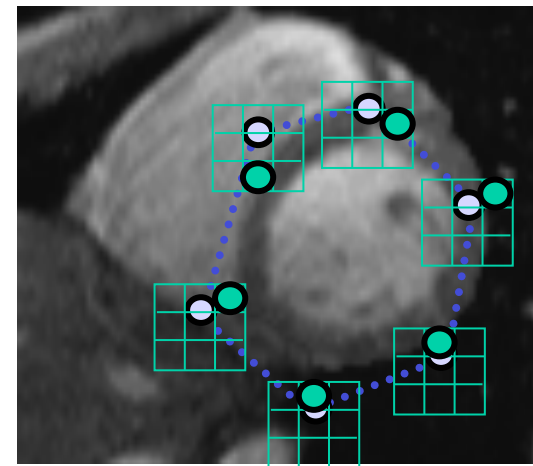
Image energy
(gradients)

Deformation energy
(good for tension,
but for bending?)

We can minimize this form of energy function, using **dynamic programming**,

with the *Viterbi* algorithm:

1. Forward-pass over nodes X positions to find accumulated optimal energies
2. Backtrack to find best positions

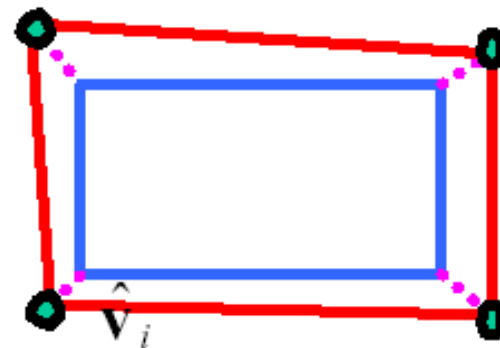


Complexity: $O(nh^2)$ vs brute force search _____?

Optional: specify shape prior

If object is some smooth variation of a known shape, we can add a term to penalize deviations from that shape:

$$\delta \sum_{i=0}^{n-1} (v_i - \hat{v}_i)^2$$

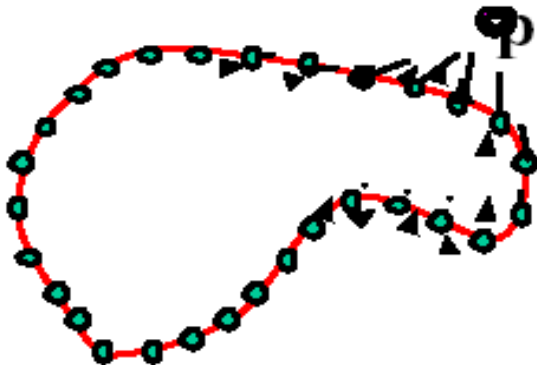
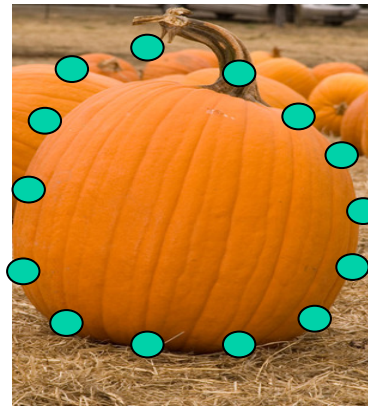
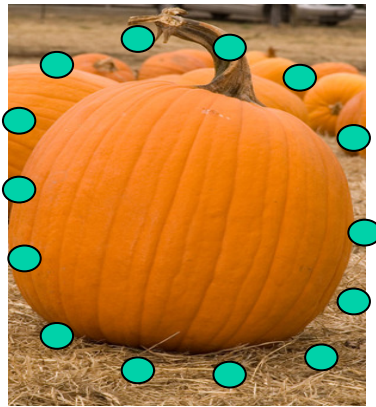


where $\{\hat{v}_i\}$ are the points of the known shape

Pressure or Interactive forces

Quite easy to define additional / alternative forces in the energy:

- Constant pressure can push the curve outside; aka. balloons
- Energy function can be altered online based on user input; e.g. **push or pull the snake points with the mouse pointer**
- Some heuristic force
e.g. avoiding image borders, utilizing output of another algorithm



E_{push}

=

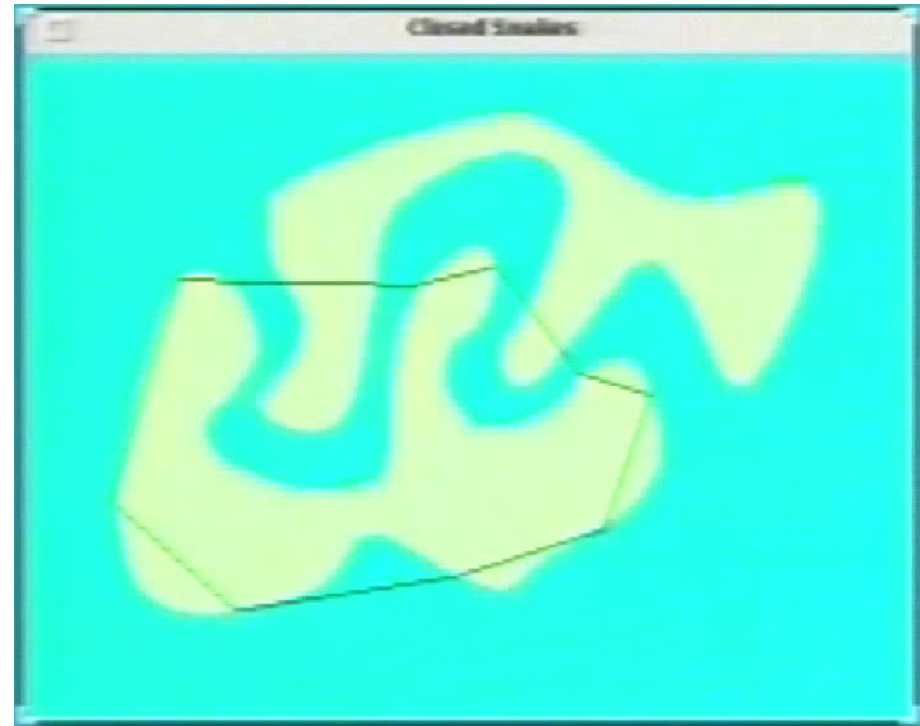
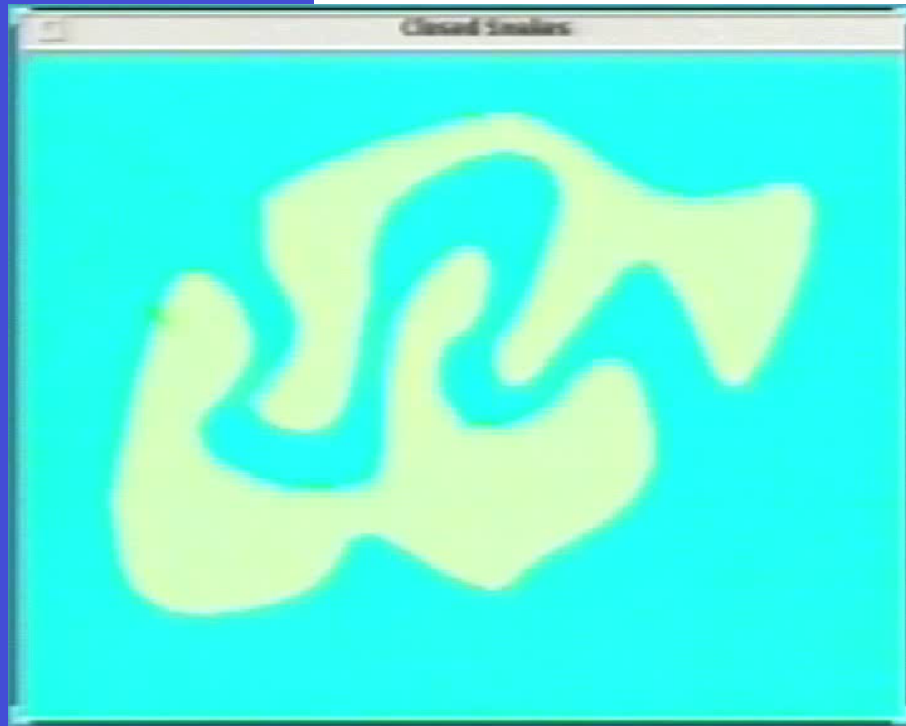
$$+ \sum_{i=0}^{n-1} \frac{r^2}{|v_i - p|^2}$$

distance of snake points
to pointer p

“Elastic springs” attached
to snakes points push them
away from the pointer (p)
with nearby points
pushed the hardest

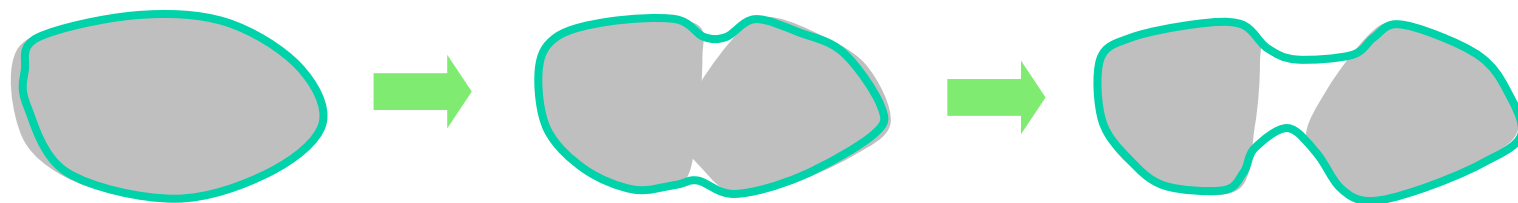
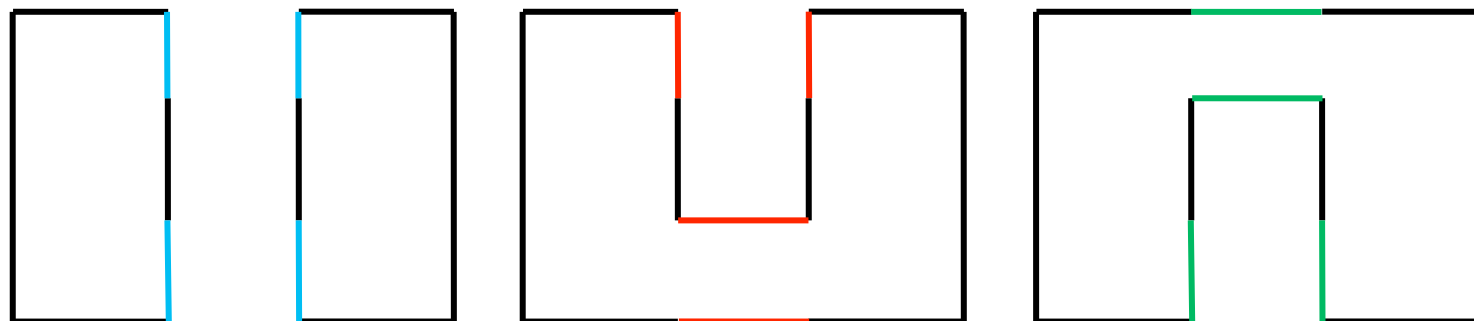
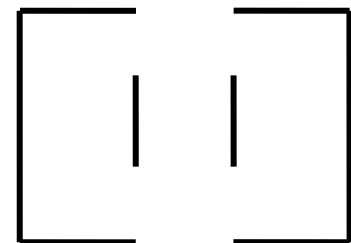
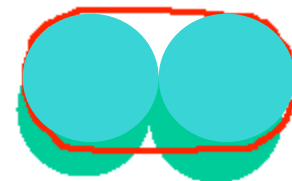
Snakes with Interactive Forces

Example video



Limitations

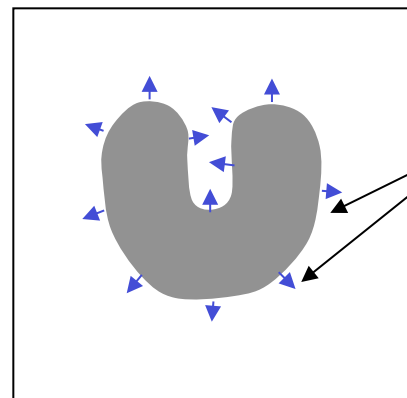
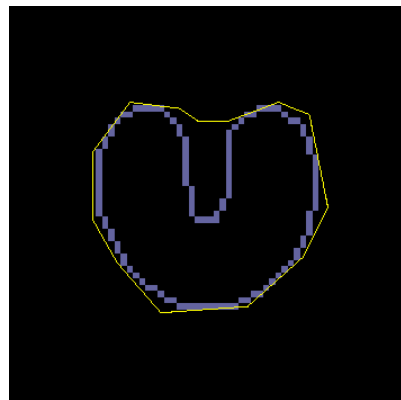
- Smoothing choice is critical, not to **over-smooth the boundary**
- **Not robust to topological differences changes**, e.g., “what is a gap to fill in?”, un/connected components



Limitations: Only locally optimal

Snakes only “see” nearby object boundaries

i.e. the external energy does not consider the edges far away from the curve (determined by gradient kernel, DP search box, etc)



∇I
image gradients
large only near
the boundary

Potential remedy: External energy based on the **distance** from edges (makes snakes “farsighted”)

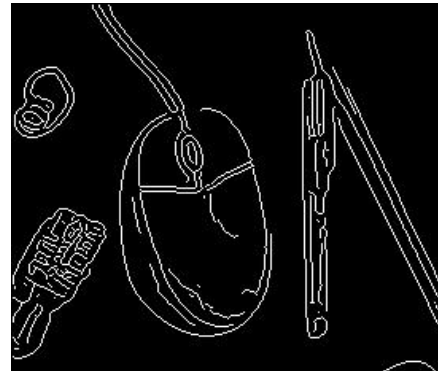
Values tell how far each location is from nearest edge



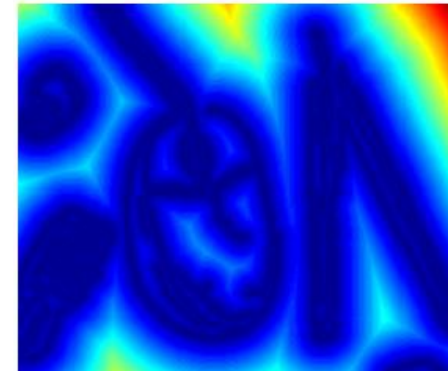
original



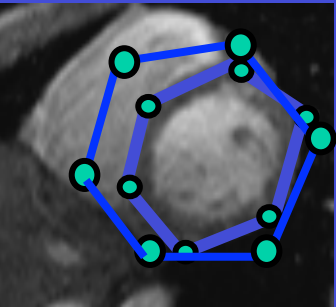
gradient



edges



distance map



Snakes: Summary

- Framework to fit deformable contours via optimization
- Define a curve as a set of n points, an internal deformation and an external image-based energy
- Initialize “near” object boundary, and iteratively optimize the curve points to minimize the total energy

Pros:

- Useful to fit non-rigid arbitrary prior shapes in images
- **Contour remains connected, i.e. topology is fixed**
- Possible to connect / fill in invisible contours
- **Flexibility in energy function definition,**
i.e., allows other forces and interactive input

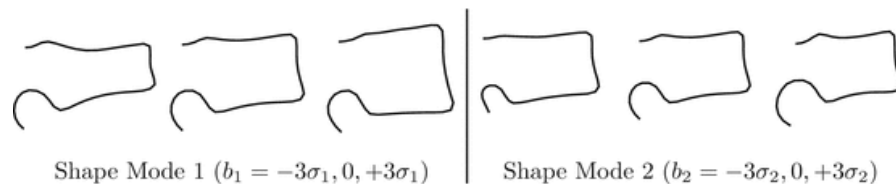
Cons:

- **Local optimization:** may get stuck in local minimum
Thus, needs good initialization near true boundary
- **Susceptible to parameterization** of energy function,
must be set based on prior information, experience, etc.

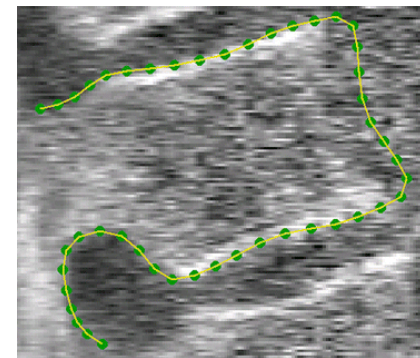
Active Shape and Appearance Models

Shape model: Implicit energy via fitness to a statistical model

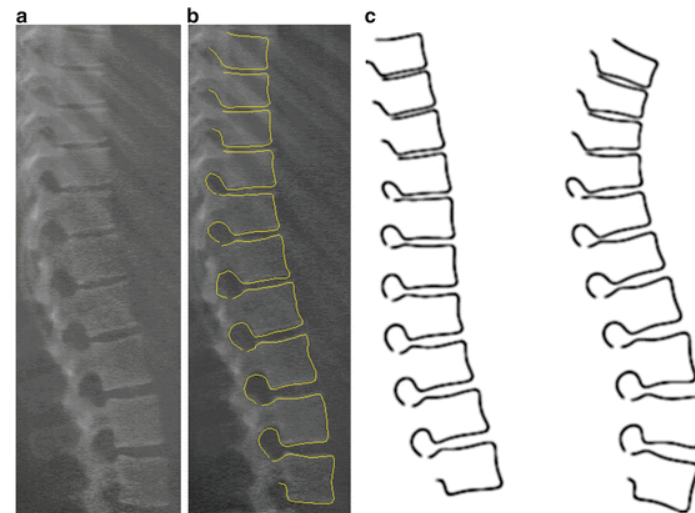
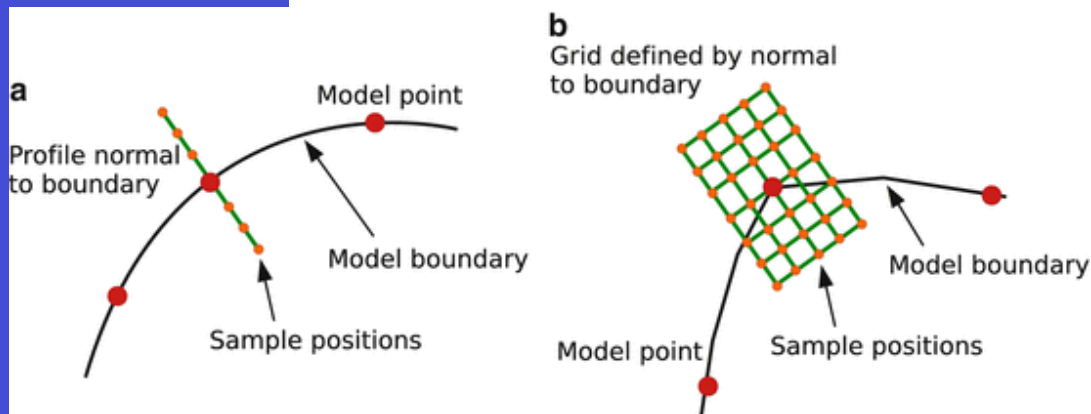
(remember PCA)



While updating the curve, project on model space to find closest shape model

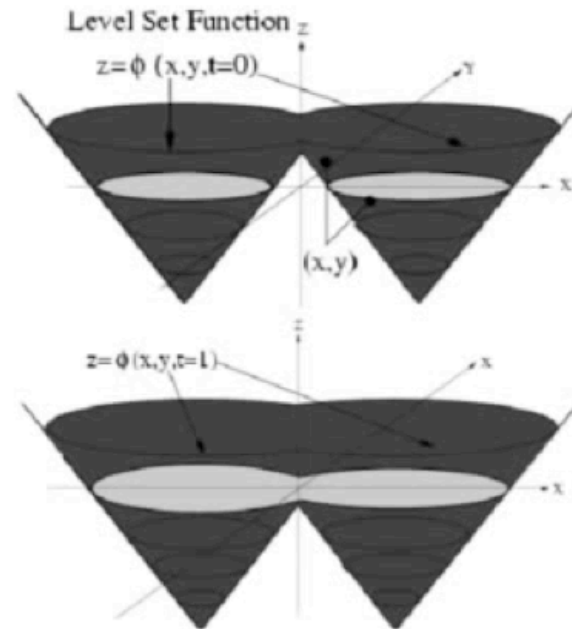


Appearance model: Make a local intensity model of edge at point v_i



Implicit curve definitions: Level-sets

- Instead of representing the contour explicitly as a set of points
- Implicit models - the contour is the *level set* of a higher dimensional function



The level set function:

$$z = \Phi(x, y, t)$$

Contour at time t :

$$0 = \Phi(x, y, t)$$

This allows the topology of the curve to change

Segmentation : Outline

- Thresholding
- Edge based
- **Region based**
- Statistical Pattern Recognition based

Region growing : principle

On the basis of segment homogeneity rather than inhomogeneity around edges

start with detection of “homogeneous” regions (e.g. low intensity variance) as the “seeds”

These are grown as long as homogeneity criterion is satisfied

Choice of appropriate homogeneity criteria is not straightforward

Region growing : example



Seeds of low intensity variance are grown, keeping intensity between two slowly floating thresholds and merging overlapping segments

Region growing : ex. cont'd



- * Often does not perform well
- * Also, risk of leakage through low contrast edges

Region growing : remarks

region growing pure is a one-way process :
if seeds are wrong, errors cannot be corrected

solution : **split-and-merge** procedures

merges of similar regions

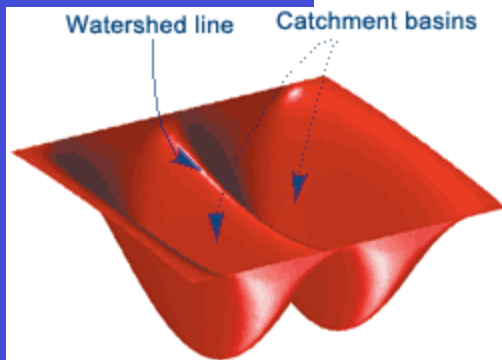
splitting more difficult : many ways to do it

Splitting calls for special decompositions
of segments, e.g. “**quadtrees**”

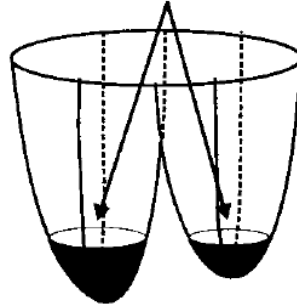
Region and edge based methods can be
combined: **hybrid approaches**

Watershed algorithm

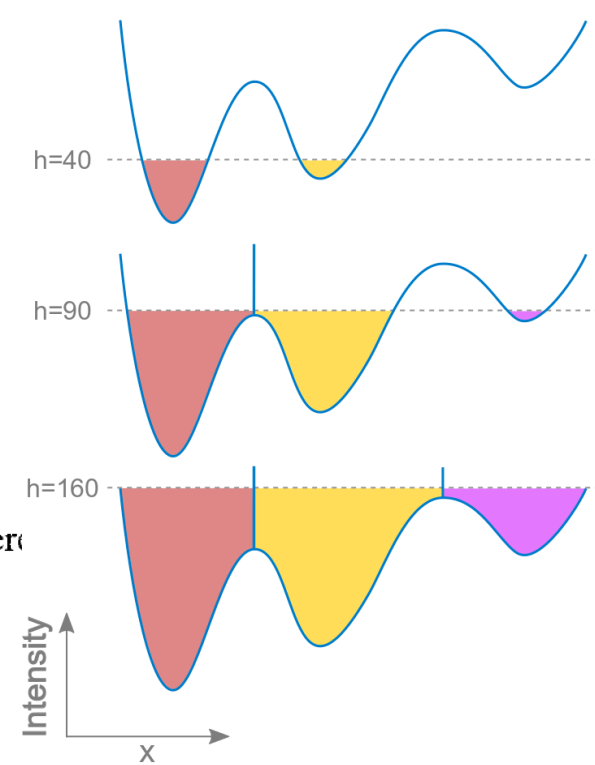
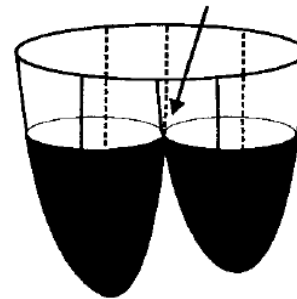
Finding catchment basins
in the image intensity graph



catchment basins begin
filling with water



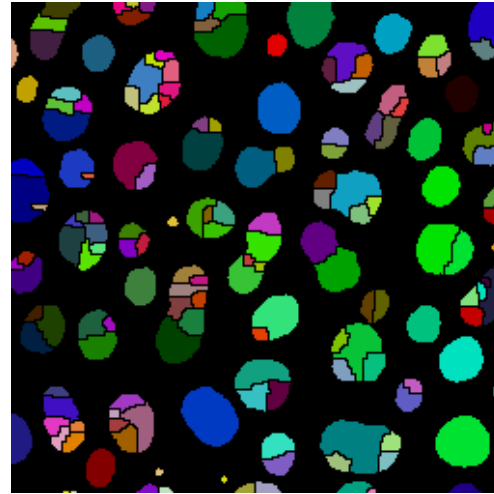
watershed line forms here



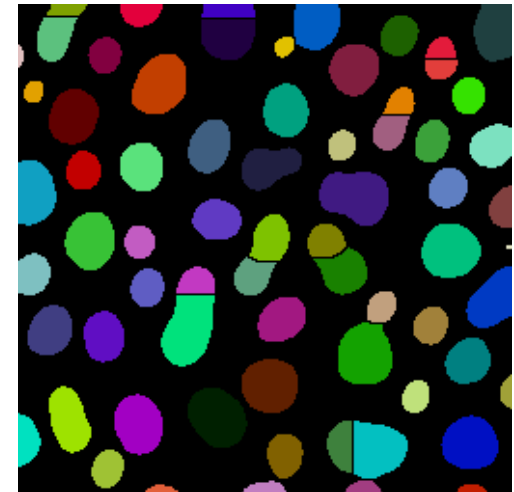
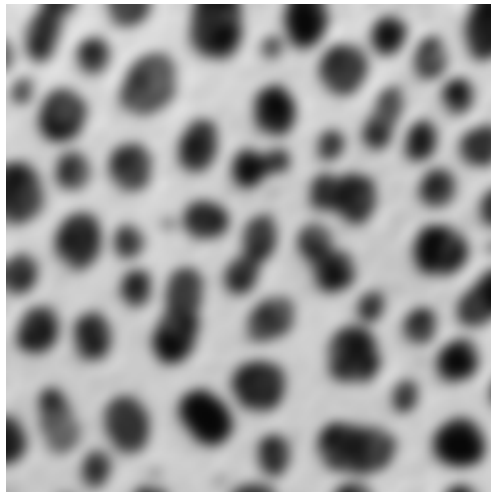
- Starting with local minima
- Different implementations of water filling (usually based on some type of region growing)
 - Stop propagation at detected ridges (lines)
- Usually strong over-segmentation

Watershed segmentation: Example

Watershed on original image



Watershed on Gaussian blurred image

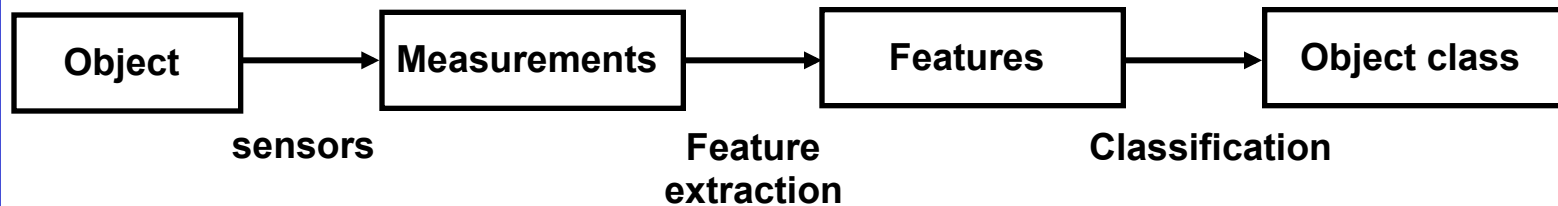


Segmentation : Outline

- Thresholding
- Edge based
- Region based
- **Statistical Pattern Recognition based**

Statistical pattern recognition

- General scheme
- Feature based
- Probabilistic and learning based formulations

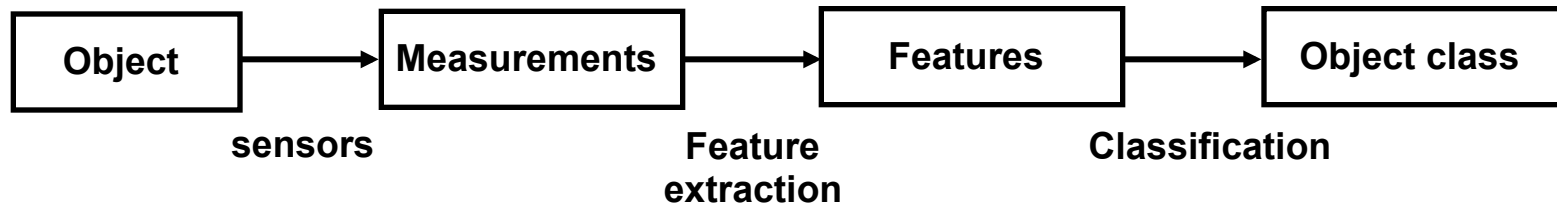


Three alternatives:

1. Unsupervised clustering
2. Supervised generative modeling
3. Supervised discriminative modeling

Additional variations and formulations exist.
Each of these can go very deep

Example application : 1



Objects: Frog

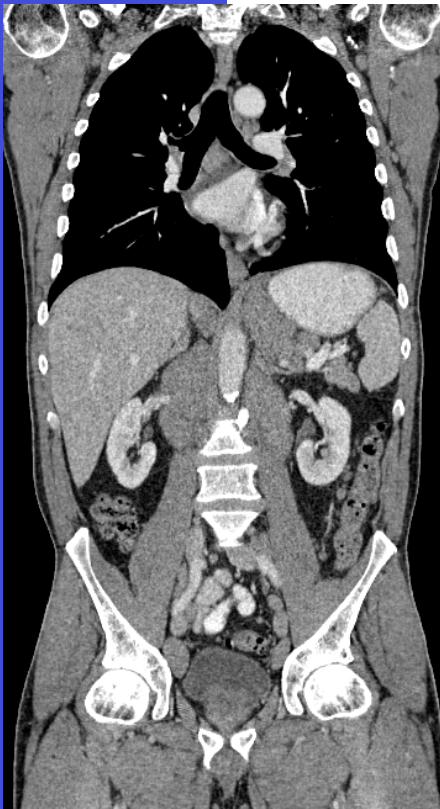
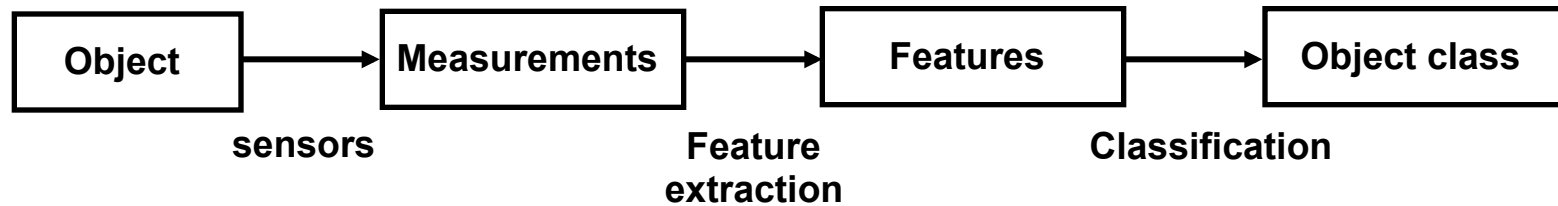
Sensors: Camera

Measurements: Pixels intensity

Features: Color

Object classes: Foreground /
Background

Example application : 2



Objects: Human body

Sensors: X-ray Computed Tomography

Measurements: X-ray attenuation
(Hounsfield unit)

Features: Hounsfield unit

Object classes: Different organs
and outside

In general, the features are essential
and have major influence on the results

Basic Notation

The set of all possible classes:

$$\Omega = \{w_1, w_2, \dots, w_K\}$$

For M measurements, and
 n features extracted for each measurement :

$$\{\vec{v}_j\}_{j=1}^M \quad \vec{v} = \{v_1, v_2, \dots, v_n\} \in \mathbb{R}^n$$

Examples of features:

- Color – 3 dimensional
- X-ray attenuation – 1 dimensional
- Pixel location – 2 dimensional
- Combinations...

Unsupervised clustering: principles

We want to distribute measurements to classes

Goal:

- Homogeneity within classes
- Reducing variance over features
- Can take into account multiple features



Available information:

- We only know the features
- No information on the classes
(to be found as a result of this
“unsupervised” process)

Unsupervised clustering: K-means Algorithm

One of the most popular and widely used algorithm

$$\{\vec{v}_j\}_{j=1}^M \quad \vec{v} = \{v_1, v_2, \dots, v_n\} \in \mathbb{R}^n$$

Choose K centers / means

$$m_i \in \mathbb{R}^n, \quad i = 1, \dots, K$$

Repeat until centers (m's) do not change:

For all measurements j , assign to nearest center i

$$c_j = \arg_i \min \|\vec{v}_j - m_i\|^2$$

For all centers i , update it to center-of-mass

$$m_i = \frac{\sum_j^M \delta_{i=c_j} \vec{v}_j}{\sum_j^M \delta_{i=c_j}}$$

$$\delta_{i=c_j} = \begin{cases} 0, & i \neq c_j \\ 1, & i = c_j \end{cases}$$

K-means analysis

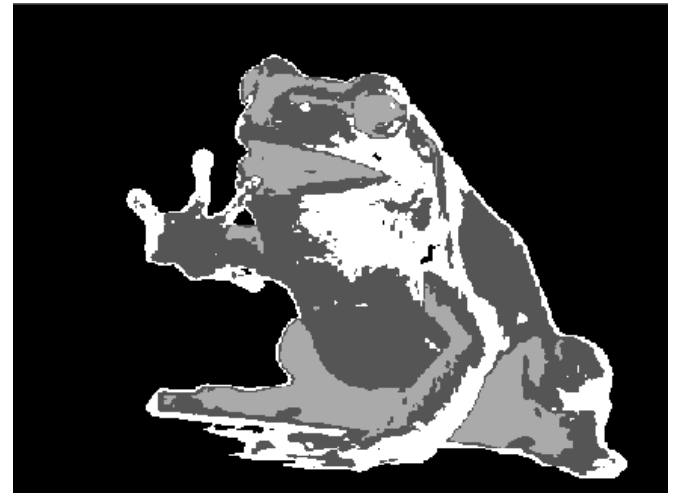
(for K classes)



K=2



K=3



K=4

K-means remarks

Extremely easy to implement

Useful initial analysis

Choice of K has a major influence

Methods exist to choose it automatically:

- Heuristic methods based on variance

- Non-parametric Bayesian

Initialization is important

K-means can get stuck in local minima

- Multiple initializations

- Multi-scale methods

Generic probabilistic formulation

Extracted features are different for each measurement : random / non-deterministic

Joint probability distribution of features and classes

$$p(\vec{v}, w_i)$$

Marginal distributions

- Probability of class occurrence (a priori probability)

$$P(w_i) = \int p(\vec{v}, w_i) d\vec{v}$$

- Probability of observing a specific feature

$$p(\vec{v}) = \sum_{i=1}^s p(\vec{v}, w_i)$$

Generic probabilistic formulation

Conditional distributions

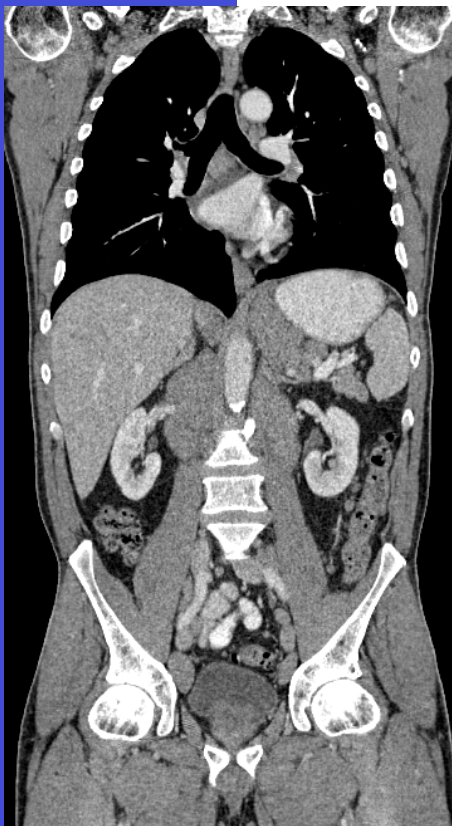
- Given class information
(measurement likelihood)

$$p(\vec{v}|w_i) = \frac{p(\vec{v}, w_i)}{p(w_i)}$$

- Given measurements
(class posterior) \rightarrow Bayes' theorem

$$P(w_i|\vec{v}) = \frac{p(\vec{v}, w_i)}{p(\vec{v})} = \frac{p(\vec{v}|w_i)P(w_i)}{p(\vec{v})}$$

Generic probabilistic formulation: examples



$\Omega = \{\text{bone, nervous tissue, muscle, air}\}$

\vec{v} : X-ray attenuation (HU) at a given pixel

$P(\text{bone})$: Probability that any pixel is bone

$P(\vec{v} = 250)$: Probability that a pixels value is 250 HU

$P(\vec{v} = 250|\text{bone})$: Probability that a bone pixel is 250 HU

$P(\text{bone}|\vec{v} = 250)$: Probability of the pixel being bone given its measurement of 250 HU

Supervised Generative Models : principle

Assumes existing examples where we can learn distributions for measurements and classes:

$$p(\vec{v}|w_i) \quad p(w_i)$$

Make use of this information to segment images, e.g.

\vec{v} : grey level intensity

$$p(\vec{v}|w_{bg}) = \mathcal{N}(250, 10)$$

$$p(\vec{v}|w_{fg}) = \mathcal{N}(150, 50)$$

$$p(w_{fg}) = p(w_{bg}) = 0.5$$



a-posteriori distribution

Compute posterior distributions
via Bayes theorem:

$$P(w_i|\vec{v}) = \frac{p(\vec{v}, w_i)}{p(\vec{v})} = \frac{\overset{\text{Can be learnt}}{p(\vec{v}|w_i)} \overset{\text{Prior}}{P(w_i)}}{\underset{\text{Constant}}{p(\vec{v})}}$$

Final segmentation by maximum a-posterior (MAP)

$$\arg_i \max(p(w_i|\vec{v}))$$



$$p(w_{bg}|\vec{v})$$



$$p(w_{fg}|\vec{v})$$

Importance of distributions

Learning the right distribution is crucial for the accuracy of the segmentation

e.g., what would happen with a different choice:

$$p(\vec{v}|w_{fg}) = \mathcal{N}(150, 50)$$

$$p(\vec{v}|w_{bg}) = \mathcal{N}(200, 50)$$



$$p(w_{bg}|\vec{v})$$



$$p(w_{fg}|\vec{v})$$

Likelihood instead of posterior

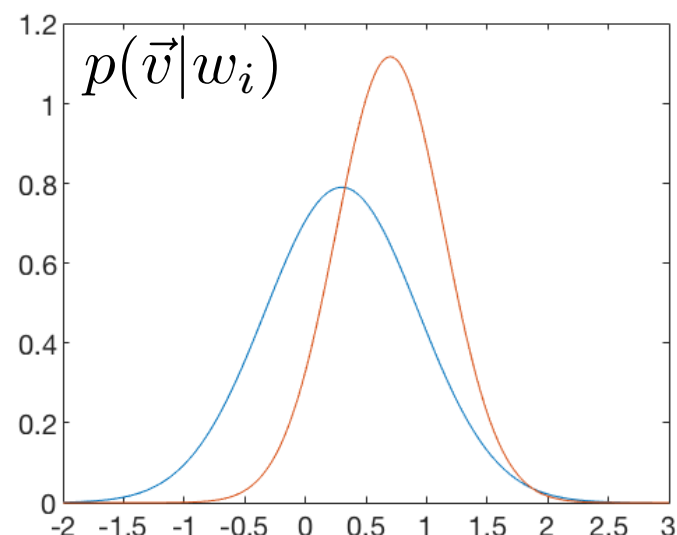
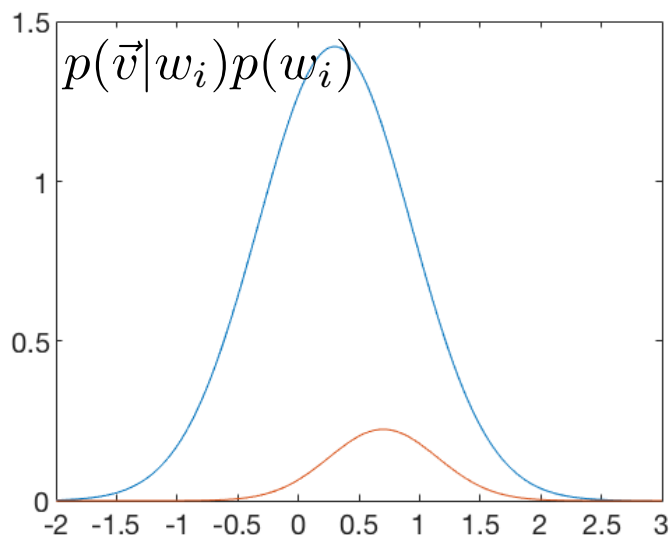
$$P(w_i|\vec{v}) = \frac{p(\vec{v}, w_i)}{p(\vec{v})} = \frac{p(\vec{v}|w_i)P(w_i)}{p(\vec{v})}$$

Posterior distribution can be difficult to compute

- How to get class priors?
in accuracy, class imbalance, etc.

$$p(\vec{v}|w_{bg}) = \mathcal{N}(0.3, 0.4) \quad p(\vec{v}|w_{fg}) = \mathcal{N}(0.7, 0.2)$$

$$p(w_{bg}) = 0.9, \quad p(w_{fg}) = 0.1$$



Alternatively: Maximize the likelihood function $\arg_i \max p(\vec{v}|w_i)$

Outline

- **Supervised generative learning:**
From individual pixels to combinations
Markov Random Fields
Gibbs sampling
Graph-cuts
- **Supervised discriminative learning
for segmentation**
KNN
Random Forests

From individual pixels to combinations:

- Earlier probabilistic segmentation model considers each pixel independently (similarly to simple thresholding)
- Independence between pixels
- Mathematical morphology for thresholding results
- Is there a way to do this with graphical models?

From individual pixels to combinations: general formulation

Class for
each pixel $\mathbf{c} = \{c_j\}_{j=1}^M$, $c_j \in \{w_1, \dots, w_K\}$

Joint distribution for an individual pixel:

$$\begin{aligned} p(c_j = w_i, \vec{v}_j) &= p(\vec{v}_j | w_i) P(w_i) \\ &= p(\vec{v}_j | c_j) P(c_j) \end{aligned}$$

Joint distribution
of all pixels when
independent

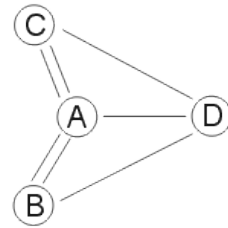
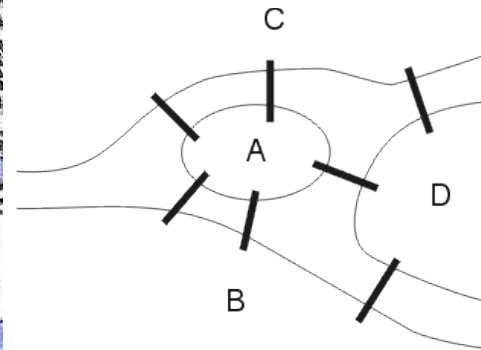
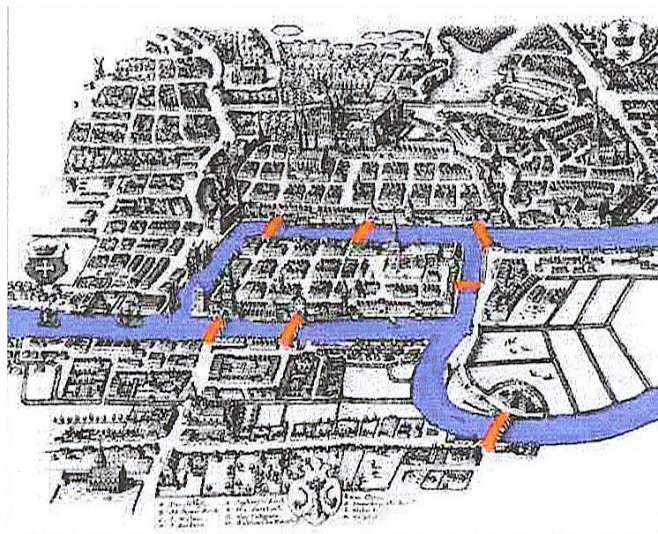
$$p(\mathbf{c}, \vec{\mathbf{v}}) = \prod_{j=1}^M p(\vec{v}_j | c_j) P(c_j)$$

Joint distribution
when pixels are
not independent

$$p(\mathbf{c}, \vec{\mathbf{v}}) = \prod_{j=1}^M p(\vec{v}_j | c_j) P(\mathbf{c})$$

Topology

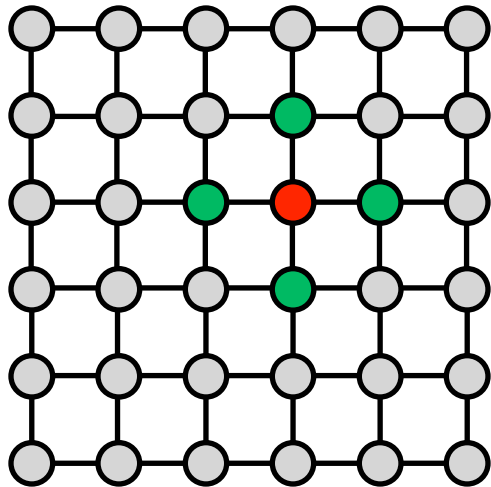
- The problem of the bridges of Königsberg Prusia (Today's Russia - Kaliningrad)
- Solution: Euler, 1736
- The birth of graph theory
- Independent of distances



Markov Random Fields: principle

$$p(\mathbf{c}, \vec{\mathbf{v}}) = \prod_{j=1}^M p(\vec{v}_j | c_j) P(\mathbf{c})$$

- MRF sets up the prior distribution based on the **Markovian property**
- Depends on the neighborhood structure



$$G_j = \text{neighbors of } j$$
$$P(c_j | \mathbf{c}_{/j}) = P(c_j | \mathbf{c}_{G_j})$$

- Probability of c_j only depends on its neighbors if all others are given

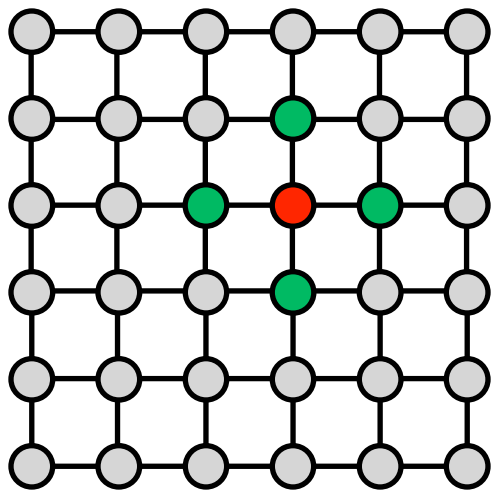
Markov Random Fields: Energies and Gibbs distributions

$$P(c_j | \mathbf{c}_{/j}) = P(c_j | \mathbf{c}_{G_j})$$

Common way to define it is through defining an **energy**

$$E(\mathbf{c}) = \sum_{j=1}^M \sum_{k \in G_j} d(c_j, c_k)$$

$d(c_j, c_k)$: distance between c_j and c_k



From energies we can define a probability distribution

Gibbs Distribution
(Boltzmann Distribution)

$$P(\mathbf{c}) = \frac{1}{Z} \exp(-E(\mathbf{c}))$$

Distance to enforce consistency!

Markov Random Fields: Posterior Maximization

Segmentation through posterior maximization:

$$\begin{aligned}\arg_{\mathbf{c}} \max P(\mathbf{c}|\vec{\mathbf{v}}) &= \arg_{\mathbf{c}} \max p(\vec{\mathbf{v}}|\mathbf{c})P(\mathbf{c}) \\ &= \arg_{\mathbf{c}} \max \prod_{j=1}^M p(\vec{v}_j|c_j) \exp(-E(\mathbf{c}))\end{aligned}$$

If the data model is also exponential of an energy

$$p(\vec{v}_j|c_j) \propto \exp(-f(\vec{v}_j, \theta_{c_j}))$$

Energy model for the prior distribution – **Ising / Potts Model**

$$d(c_j, c_k) = \lambda \delta(c_j \neq c_k) = \begin{cases} 0, & c_j = c_k \\ \lambda, & c_j \neq c_k \end{cases}$$

Optimization

$$\arg_{\mathbf{c}} \max p(\mathbf{c}|\vec{\mathbf{v}}) = \arg_{\mathbf{c}} \max p(\vec{\mathbf{v}}|\mathbf{c})P(\mathbf{c})$$

Equivalently

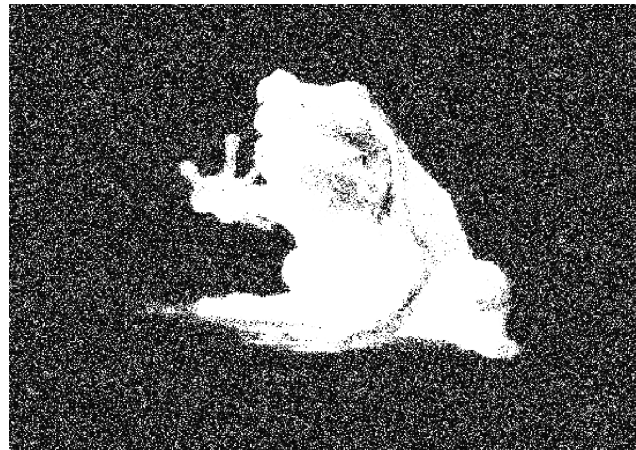
$$\arg_{\mathbf{c}} \min \sum_{j=1}^M (\vec{v}_j - \mu_{c_j})^T \Sigma_{c_j}^{-1} (\vec{v}_j - \mu_{c_j}) + \lambda \sum_{j=1}^M \sum_{k \in G_j} \delta(c_j \neq c_k)$$

- Very difficult to compute the posterior distribution
- Difficult to solve the optimization exactly in 2D and higher
- NP-hard [Boykov, Veksler and Zabih 2001 TPAMI]
- Approximate energy minimization
 - Gibbs sampling [Geman & Geman 1984]
 - Iterated conditional modes (ICM) [Ferrari et al. 1995]
 - via Graph Cuts [Boykov, Veksler & Zabih 2001]
 - ...

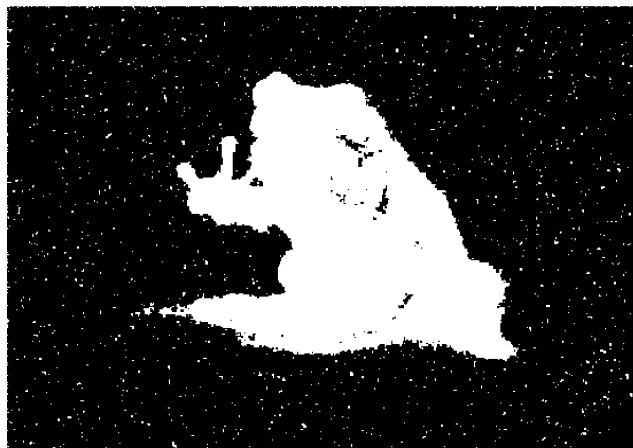
Stochastic Relaxation / Gibbs Sampling: in action



Noisy image



Initial segmentation



After 1 iteration of Gibbs

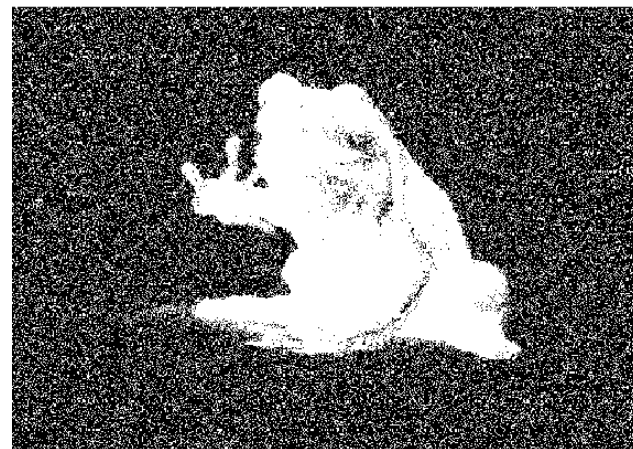


After 25 iterations

Stochastic Relaxation / Gibbs Sampling: analysis with respect to λ



Noisy image



$\lambda = 0$



$\lambda = 5$



$\lambda = 20$

Stochastic Relaxation / Gibbs Sampling: remarks

- Very simple implementation and effective
- Stochastic in nature
- Solution depends on the lambda parameter
- Solution depends on initial condition
- Not very efficient – convergence is slow
- Does not always converge to a pleasing result

Remarks on Generative Supervised Models

- Mathematically sound
 - Flexible and generic
Model specifications can change
 - Can be extended to various features
 - Wealth of research
 - Links to Bayesian methods
 - Features are very important
-
- Optimization and inference can be hard
 - For high dimensional features
estimation of distributions can be problematic
Latent variables / factors

Discriminative learning: principles

For mapping from features to classes

$$\{\vec{v}_j\}_{j=1}^M \longrightarrow \{w_i\}_{i=1}^K$$

A procedure that takes the features as input and predicts the segmentation label / class assignment

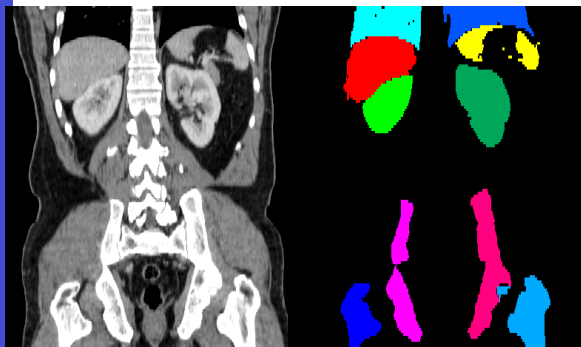
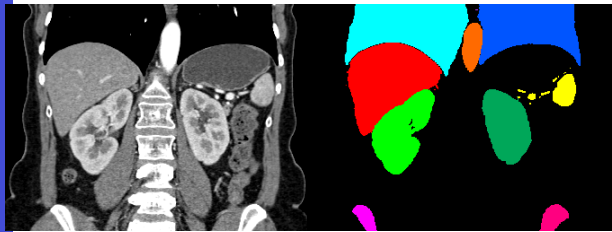
$$f(\vec{v}_j) = c_j$$

In terms of probabilities, this is similar to directly modeling posterior or its maximum

$$f(\vec{v}_j) \sim p(c_j | \vec{v}_j) \qquad f(\vec{v}_j) \sim \arg_{c_j} \max p(c_j | \vec{v}_j)$$

”Learn” the ”mapping” from ”examples”

Discriminative learning: examples - Training Data



- Composed of images and the corresponding segmentations of the objects you are interested in
- Application dependent
- The “ground-truth” segmentations are often annotated manually or with a semi-automatic algorithm
- Can be VERY expensive
- and VERY valuable
- Think of other applications...

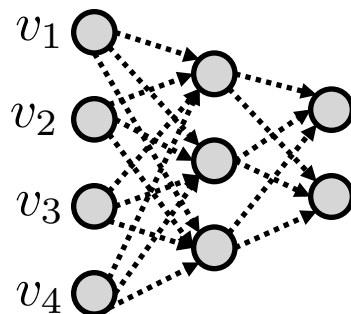
Discriminative learning: Mapping

$$f(\vec{v}_j) = c_j$$

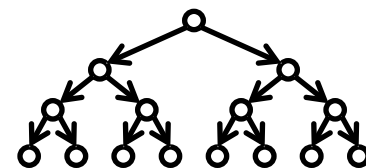
- Mapping is a parametric model
- KNN – K-nearest neighbors
- Logistic regression (of binary class)

$$c_j = \frac{1}{1 + \exp(-\beta_0 - \beta^T \vec{v})}, \quad \beta^T \vec{v} = \sum_{i=1}^d \beta_i v_i$$

- Neural networks: $c_j = \sigma(\beta_2^T \sigma(\beta_1^T \sigma(\beta_0^T \vec{v})))$

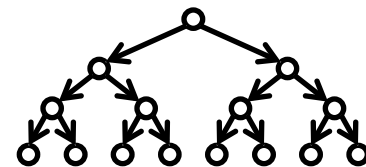
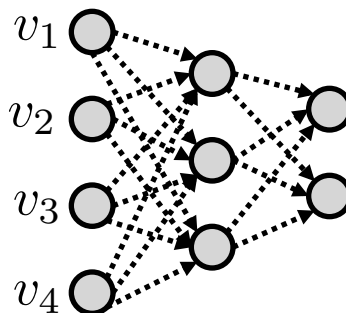


- Decision trees - random forests
- ...



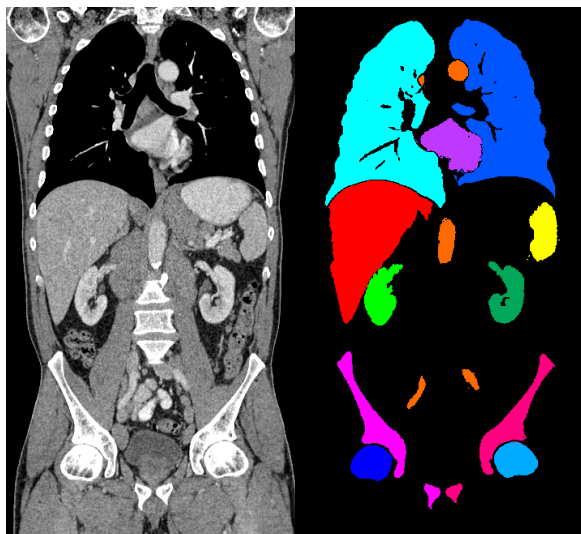
Discriminative learning: Learning / Training Phase

$$c_j = \frac{1}{1 + \exp(-\beta_0 - \beta^T \vec{v})}$$



$$c_j = \sigma(\beta_2^T \sigma(\beta_1^T \sigma(\beta_0^T \vec{v})))$$

- Estimating the parameters of the model, in order to obtain the best possible segmentation in the training examples




Training data: $\{(\vec{v}_n, c_n)\}_{n=1}^N$

- Optimization by minimizing the discrepancy between algorithm segmentation and ground truth

Discriminative learning: Segmentation Testing phase



$$\{\vec{v}_j\}_{j=1}^M$$

$$f(\vec{v}_j) = c_j$$

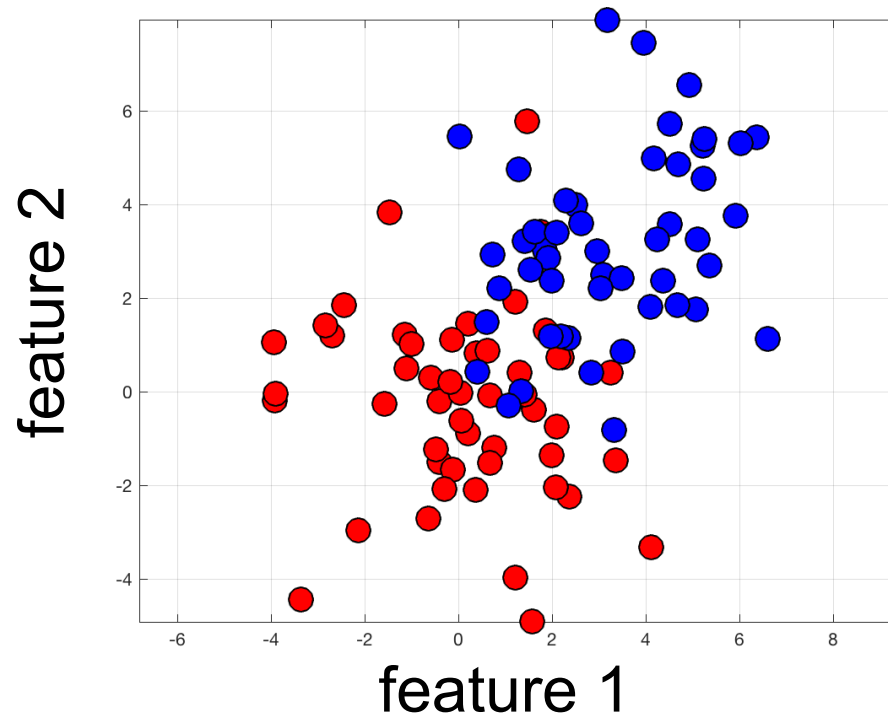


- Extract the features used during training
- Use the mapping learned before

K-nearest neighbors - KNN

- Find the K nearest neighbors within the training dataset.
- For training examples we know the features and the labels

Training Examples

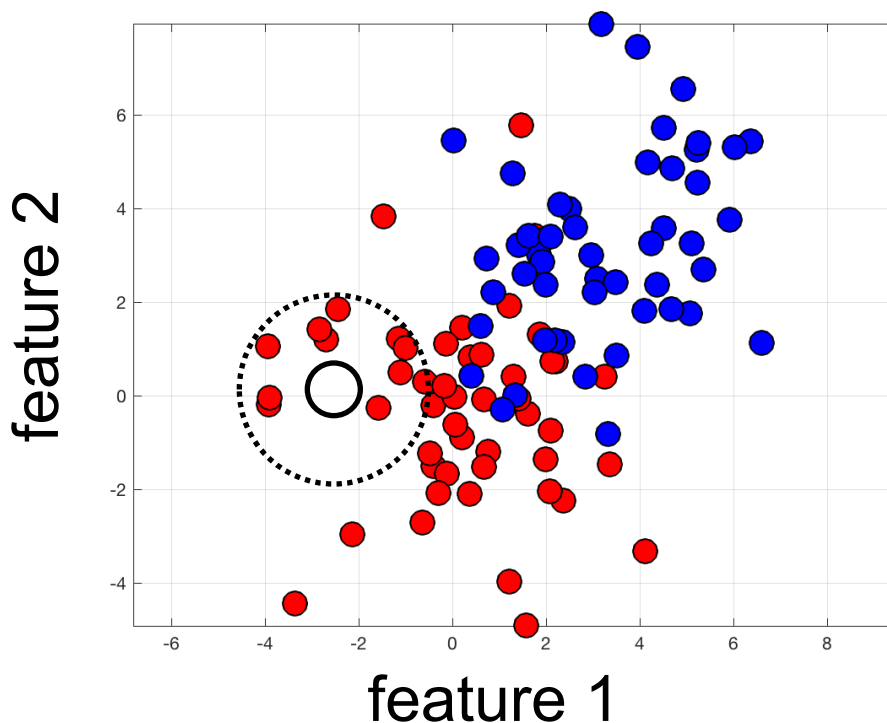


K-nearest neighbors - KNN

- The mapping is defined through the labels of the K-nearest neighbors

$$f(\vec{v}) = c$$

Training Examples



Find the K training examples with minimum distance

$$d(\vec{v}, \vec{v}_n)$$

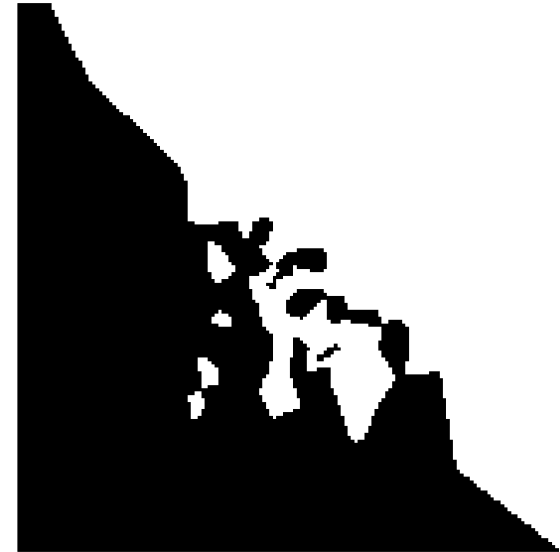
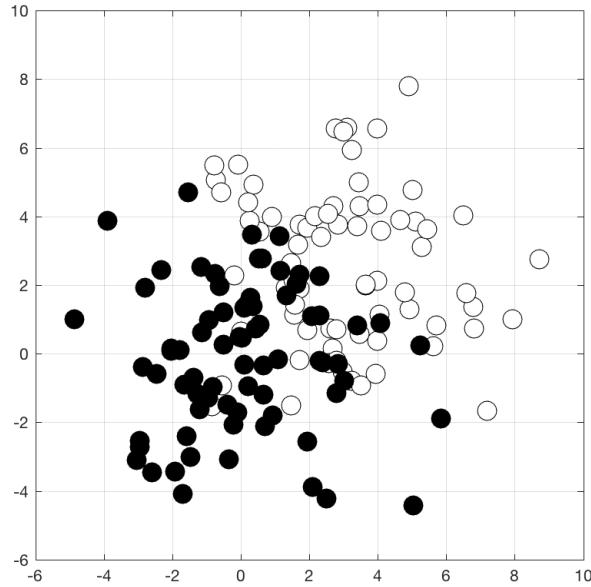
Mapping is the function of the corresponding labels

$$f(\vec{v}) = f(c_1, \dots, c_K)$$

K-nearest neighbors – KNN: parameterization

- Define the term “nearest” \rightarrow distance
- Define the mapping $f(\vec{v}) = f(c_1, \dots, c_K)$
 - Majority voting
 - Weighted majority voting
 - Probabilities with uncertainties
- With an additional training, the parameters of the distance and mapping can also be estimated if there are any...

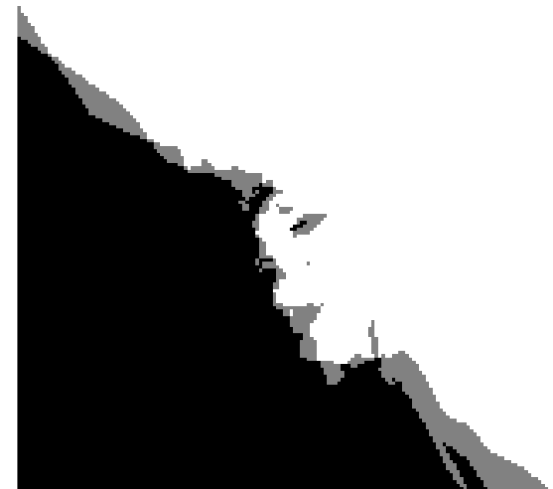
KNN defines a partitioning – majority voting



K=1

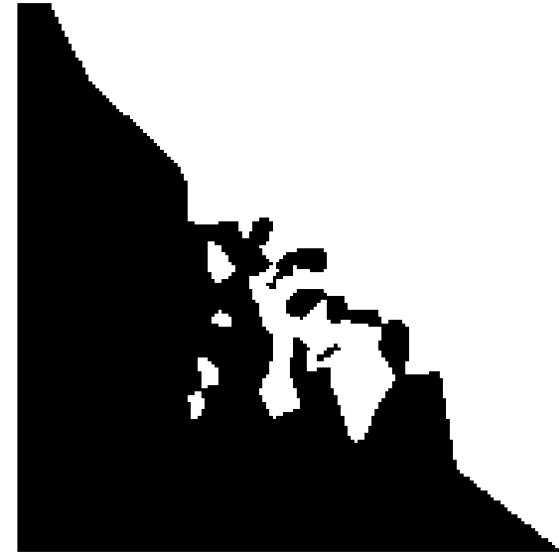
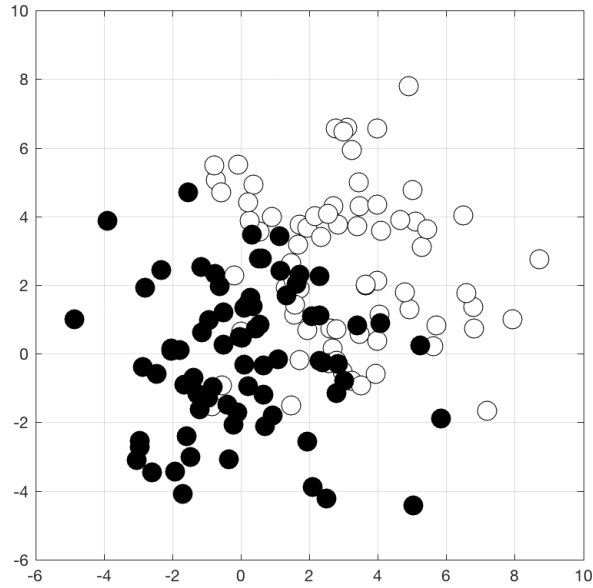


K=5

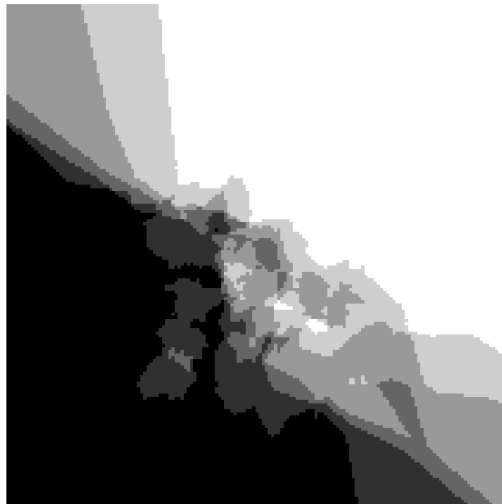


K=10

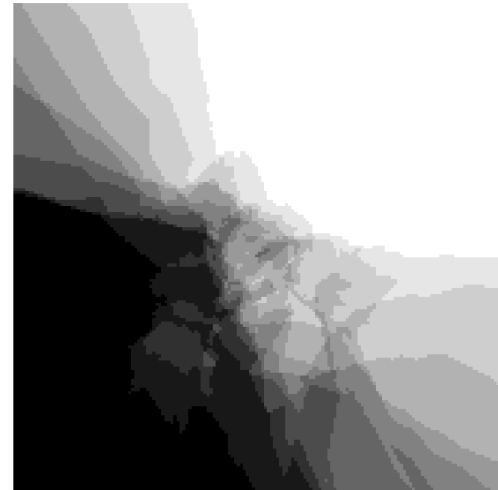
Probabilities via neighborhood proportions



$K=1$



$K=5$



$K=10$

Remarks on KNN

Pros:

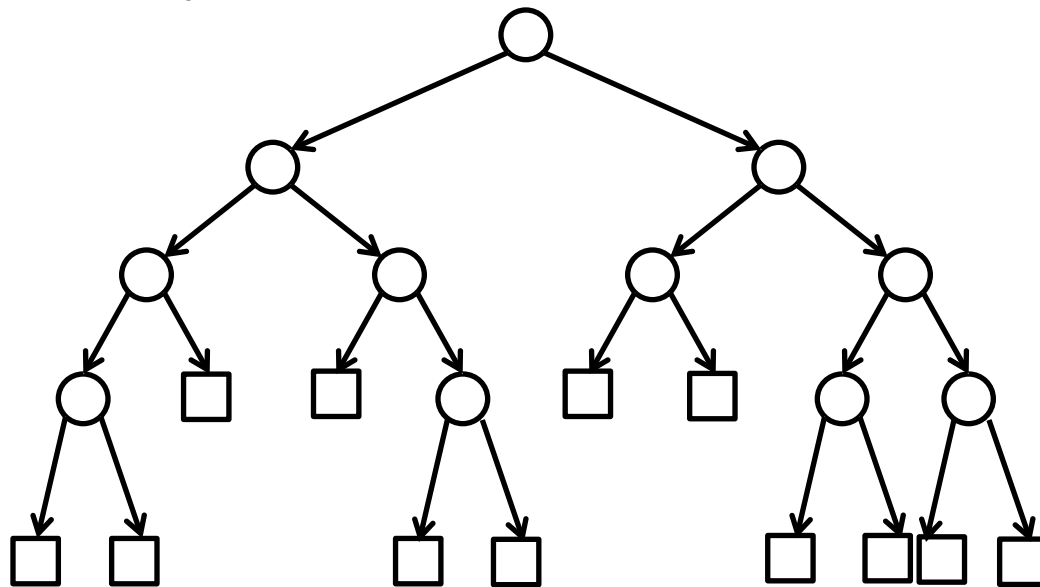
- Very simple to implement
- Very simple to understand
- Efficient implementations possible
approximate nearest neighbors,...
- Distance definition is flexible

Cons:

- Highly depends on the definitions and K
- Need to keep the entire data in memory
for distance computations
- For high dimensional problems (with high d)
need a LOT of training samples for accuracy
(use alternatives instead, e.g. NNs, RFs, ...)

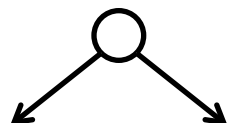
Discriminative learning: Random Forests

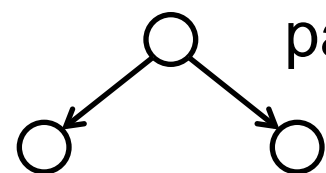
- Binary decision trees



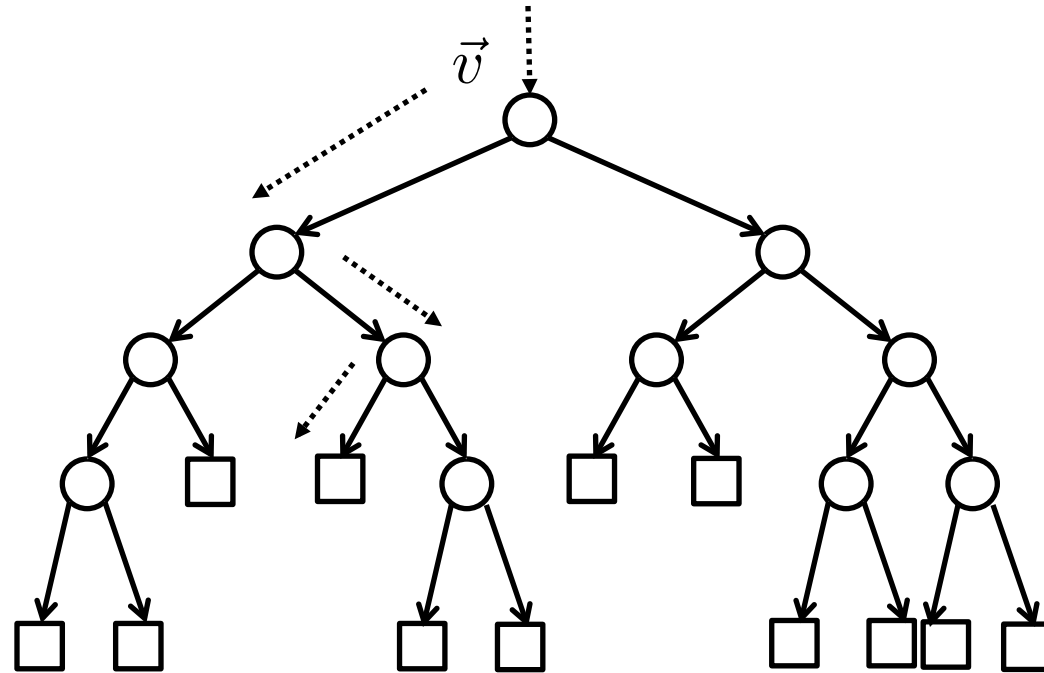
○ : internal nodes

□ : leaf nodes

 : splits

 parent children

Discriminative learning: Decision Trees

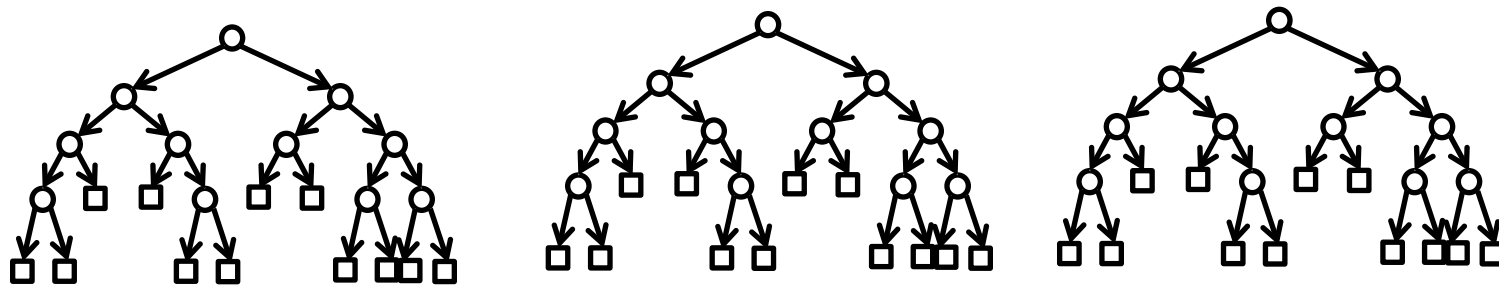


At each internal node n there is a binary question on the features $f_n(\vec{v}) = \begin{cases} 0 \rightarrow & \text{go left} \\ 1 \rightarrow & \text{go right} \end{cases}$

At each leaf node there is a prediction and it changes from leaf node to leaf node $f_l(\vec{v}) = c$

Discriminative learning: Random Forest

- Ensemble of decision trees



- Each tree is different than others
- Focus on a different set of features

$$\vec{v} = \{v_1, \dots, v_d\}$$

$$\vec{v}_i = \{v_{i_1}, v_{i_2}, \dots, v_{i_D}\} \quad d \gg D$$

- An approach to deal with high dimensionality

Random Forests: parameterization

Two levels of parametrization:

- What are the tests?

$$f_n(\vec{v}) = \begin{cases} 0 \rightarrow & \text{go left} \\ 1 \rightarrow & \text{go right} \end{cases}$$

- How are the predictions done?

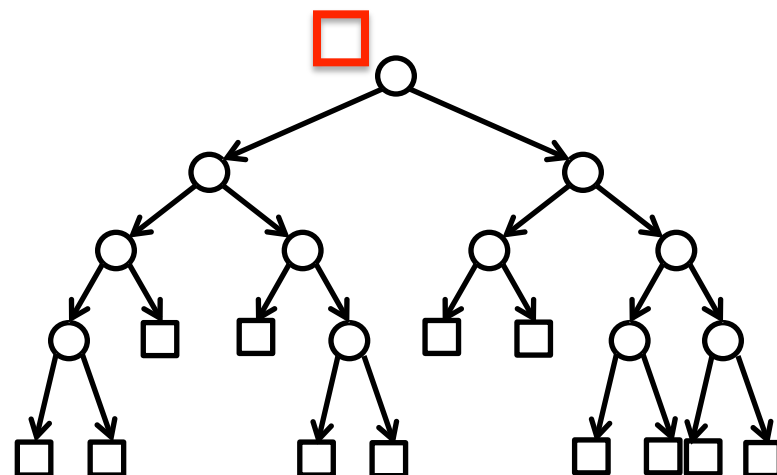
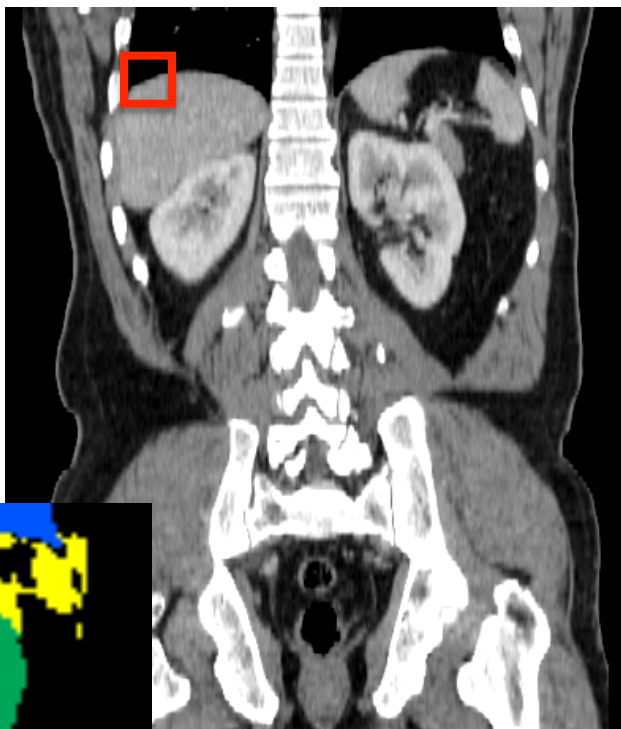
$$f_l(\vec{v}) = c$$

Both of these are learned during training

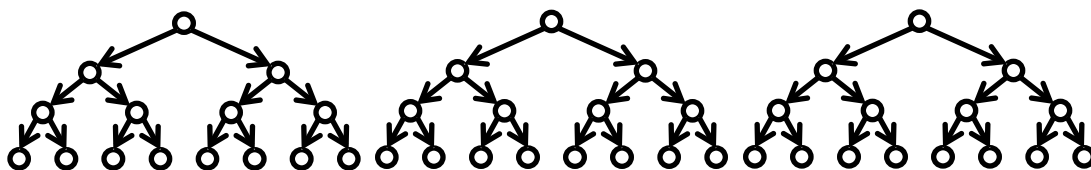
Various alternatives for both

Random forests: During Segmentation

$$\square = \vec{v} = \{v_1, \dots, v_d\}$$



$$f_n(\vec{v}) = \begin{cases} 0 \rightarrow & \text{go left} \\ 1 \rightarrow & \text{go right} \end{cases} \quad f_l(\vec{v}) = c$$



Random Forests: remarks

- Easy to implement
- VERY efficient
- Technology behind Kinect (earlier version)

- Lots of parametric choices
- Needs large number of data
- Training can take time