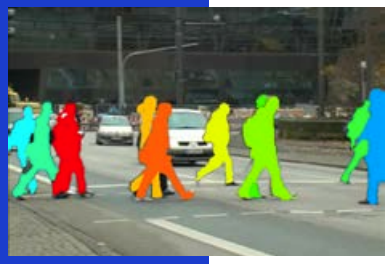
Segmentation

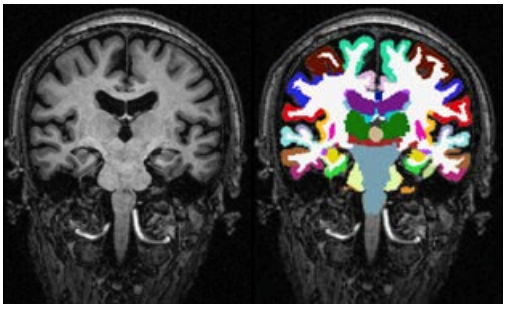
Computer Vision

Segmentation

Grouping pixels into segments, to identify (semantic) entities in the image, e.g. objects

- Industrial inspection
- Disease diagnosis
- Understanding scenes
- ....





Computer Vision

Segmentation : Outline

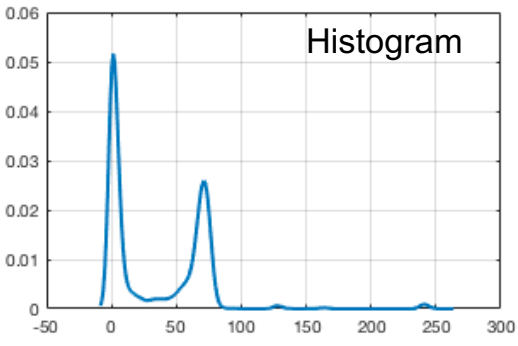
- **Thresholding** (voxel-based)
  - Morphological enhancement
  - Connected-components
- **Edge-based**
  - Hough Transform
  - Deformable contours
- **Region-based**
  - Region-growing, Watershed
- **Pattern Recognition (Feature-)based**
  - Clustering (unsupervised)
  - Generative modeling (supervised)
  - Discriminative modeling (unsupervised)

Computer Vision

Thresholding : basic idea

Assuming contrast between object and background, determine intensity threshold to divide in 2 segments

Example image

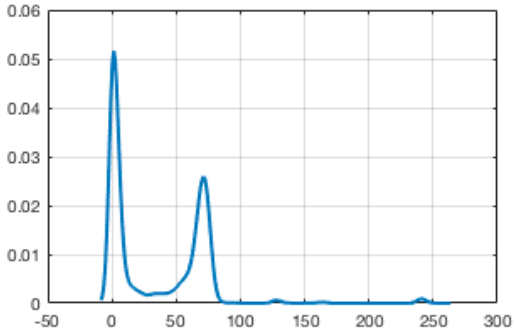


Threshold = 5 Threshold = 25 Threshold = 50 Threshold = 70

Computer Vision

Thresholding : threshold selection

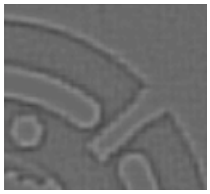
Histogram



If intensities of objects and background are known → easy

Otherwise, make a guess based on, e.g.:

1. From histogram, take the minimum between two peaks
2. For known size (e.g. for industrial applications), increase threshold until reaching a predefined area
3. Maximize sum of gradient magnitudes at pixels with threshold intensity
4. Define based on regions with high response to Laplacian filter, i.e. points around edges



Computer Vision

Thresholding: Otsu criterion

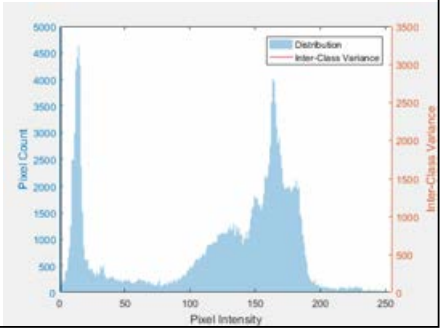
Maximize inter-group variance ==
minimize intra-group variance (weighted by group size *p*)
(i.e., we want large areas of low variance)

Group 1
I > threshold
relative area, variance
p_1, σ_1^2

Group 2
I <= threshold
relative area, variance
p_2, σ_2^2

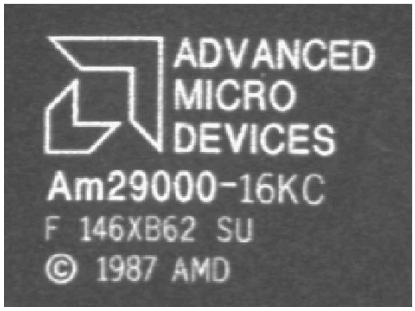
Try for every possible threshold,
to minimize $p_1\sigma_1^2 + p_2\sigma_2^2$

Otsu threshold
= 35

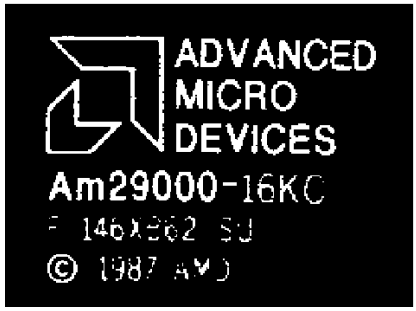


Computer Vision

Thresholding: Global vs. Local Otsu



Image



Fixed threshold

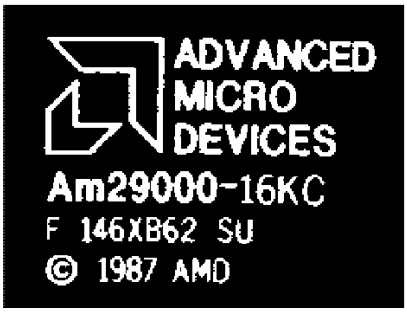
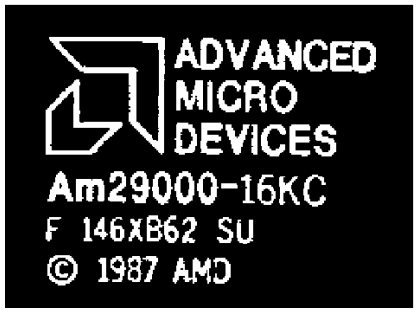


Image specific Otsu threshold



Local Otsu threshold

Since lighting, noise, etc. may change across the image, one can find different (local) thresholds

Computer Vision

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., to remove isolated islands or holes

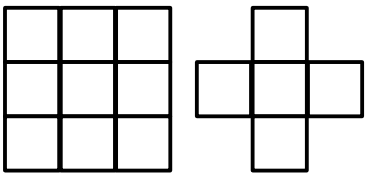
Computer Vision

Mathematical Morphology: Basics

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

Binary Image: A

- Two main operations:



Binary structural element: B

Dilation

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

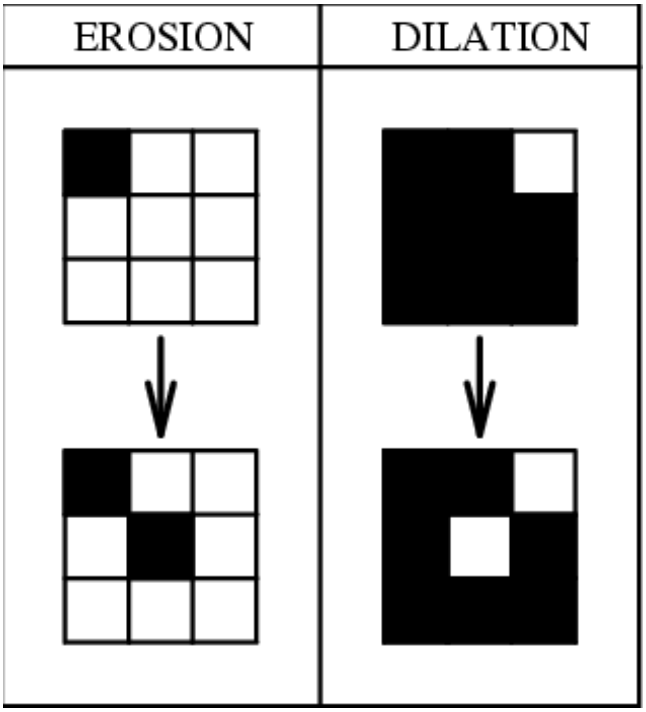
Erosion

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

Computer Vision

Example

Given structural element: 



Erosion

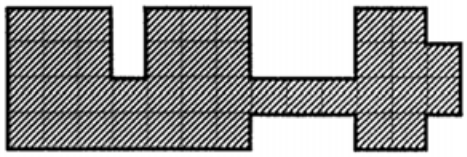
Dilation

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

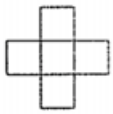
Computer Vision

Opening and Closing

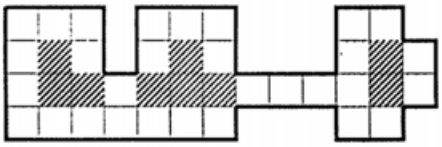
Not to change object size (by much),
concatenate (same number of) opposite basic operations, e.g.



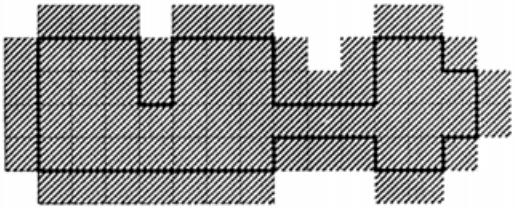
original image structure



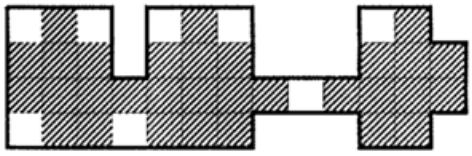
structuring element

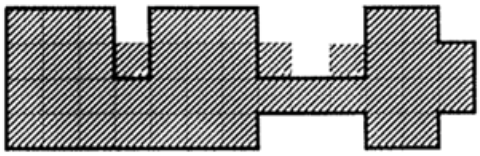


Erosion



Dilation

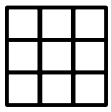




Opening $(A \ominus B) \oplus B$ Closing $(A \oplus B) \ominus B$

Computer Vision

Opening and closing: Example



Opening
(mostly removes islands)

Closing
(mostly removes holes)

Computer Vision

Interpretation of morphological as rank order operators

New intensity based on rank-ordered neighborhood values:

$$i_1 \leq i_2 \leq \cdots \leq i_N$$

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

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

Enables extension to gray level images

Important difference with convolution : non-linearity

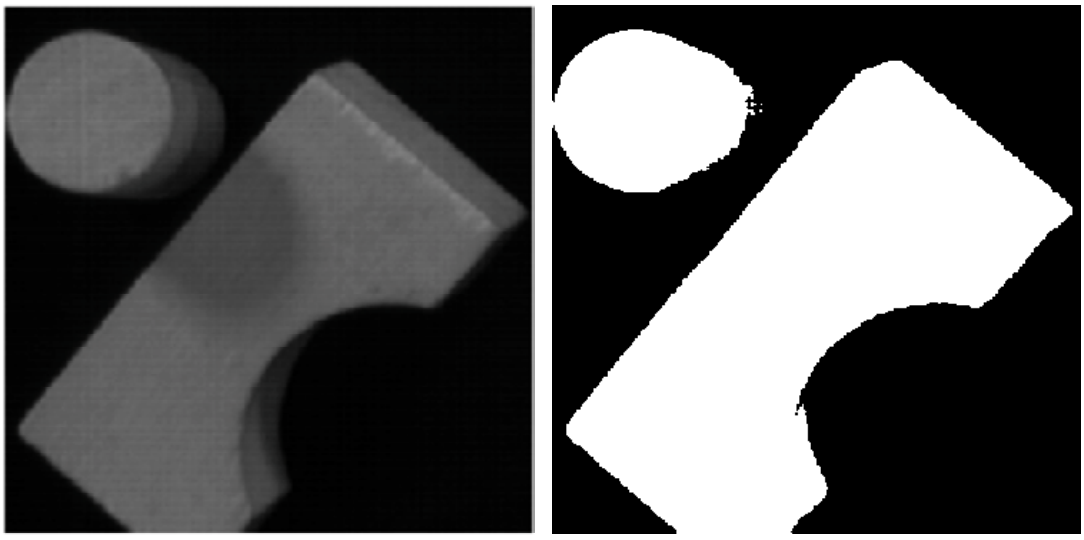
Computer Vision

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

Computer Vision

Multiple objects: Connected components



We would like to separate these objects

Connected component analysis

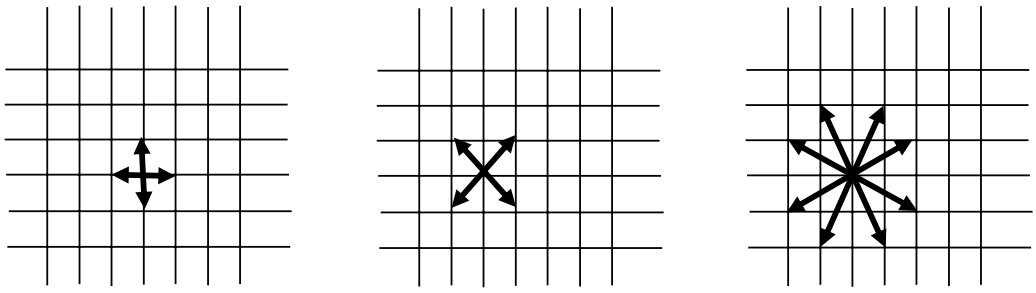
What does it mean to be a connected object?

Computer Vision

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



Computer Vision

Topology induced (distance) metrics

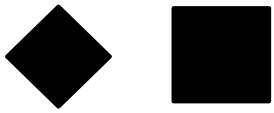
Depends on the chosen definition of image topology and neighborhood connectivity

- e.g., D_4 (Manhattan) and D_8 distances
- Between points 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



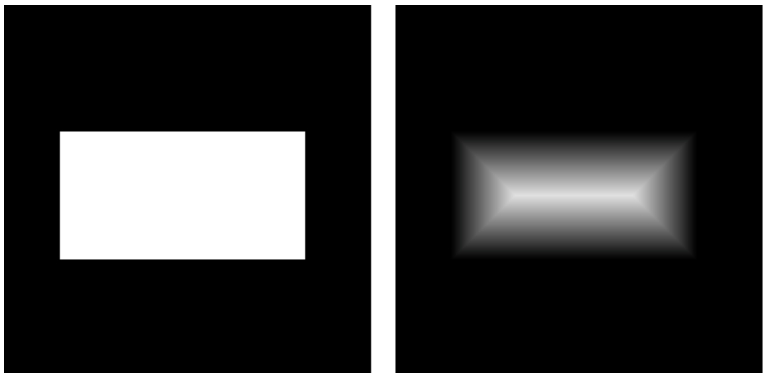
- Euclidean would be 

does not conform with any discrete neighbourhood

Computer Vision

Distance calculation

- Distance transformation: "distance map" based on distance propagation along neighbourhoods

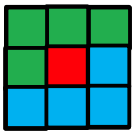


- Euclidean distance map
  - True implementation is very cumbersome
  - Approximations are possible

Computer Vision

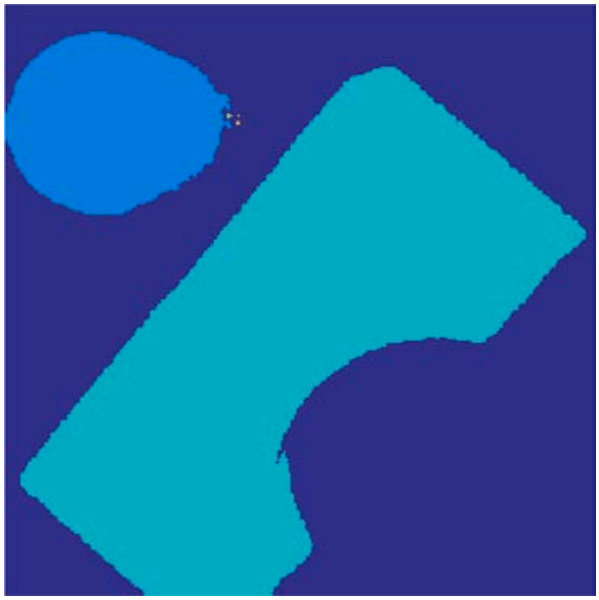
Single-pass 8 connected-component labeling

- 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



Computer Vision

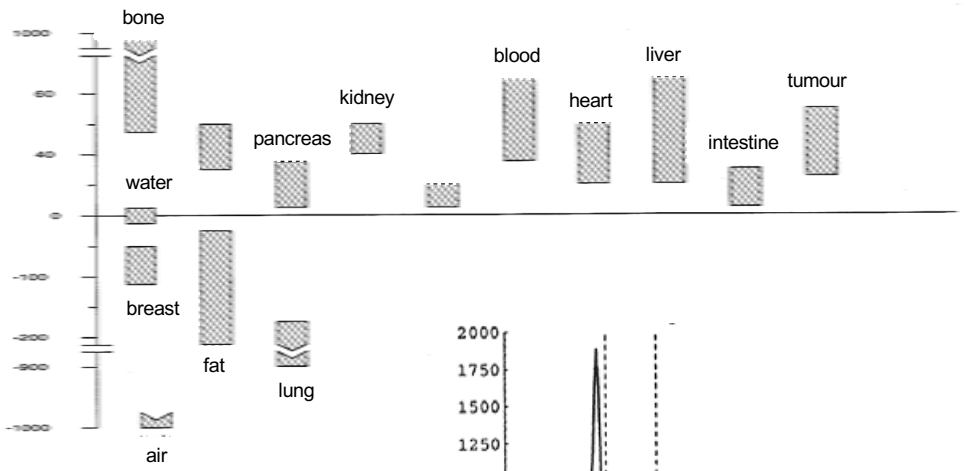
Thresholding : remarks

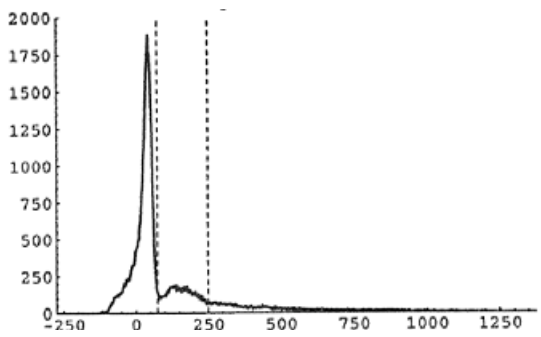
- Threshold advantages:
  1. Serious bandwidth reduction
  2. Simplification for further processing
  3. Availability of real-time hardware
- Generally it won't provide a satisfying segmentation
- Pixel-by-pixel decision
  - ignores neighbouring pixels
  - structural information lost

Computer Vision

Thresholding has limitations

X-ray attenuation is tissue dependent





Overlap in intensity ranges hinder (pixelwise) thresholding decisions

•Spatial continuity, edges, etc. can help separate objects

Computer Vision

Segmentation : Outline

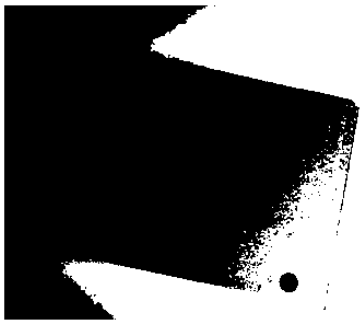
- Thresholding
- **Edge based**
- Region based
- Statistical Pattern Recognition based

Computer Vision

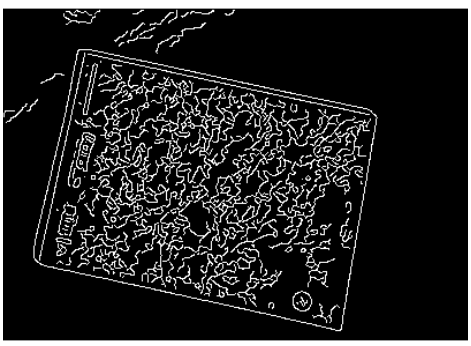
Edges as a strong cue that delineate object boundaries

Identifying boundaries between different areas / objects

Image



After thresholding with Otsu criteria



Edge Detection with Canny

Computer Vision

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)

Computer Vision

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

Computer Vision

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

Computer Vision

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 :

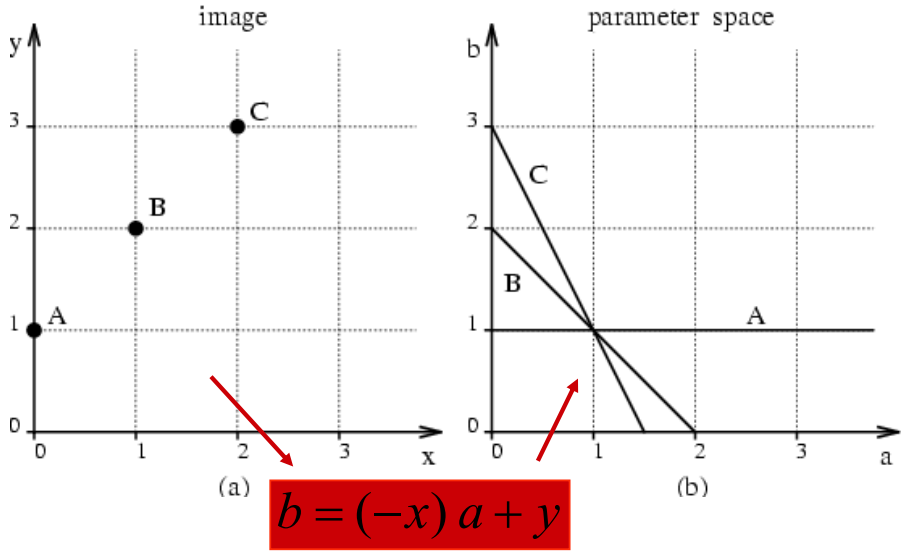
$$b = (-x)\,a + y$$

The Hough transform :

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

Computer Vision

Hough transform : straight lines



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

implementation :

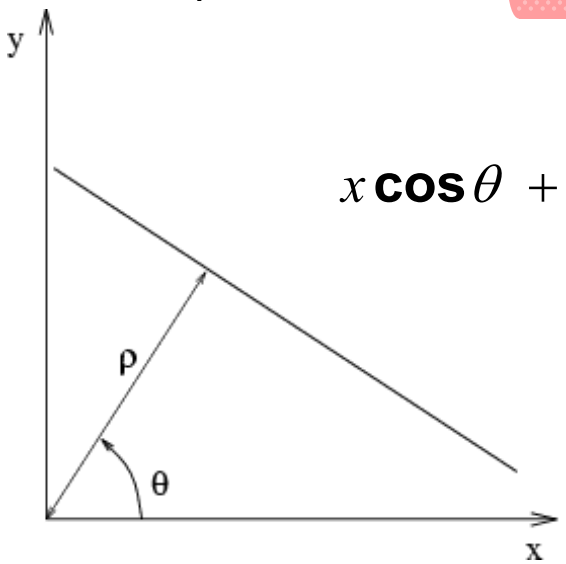
1. the parameter space is discretised
2. a (weighted) counter is incremented at each parameter cell where the lines pass
3. At the end, peaks are detected

Computer Vision

Hough transform : straight lines

problem : unbounded parameter domain
vertical lines require infinite *a*

alternative representation:

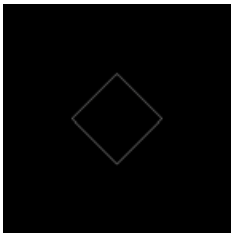


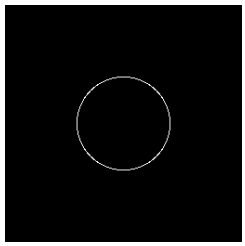
$$x\cos\theta + y\sin\theta = \rho$$

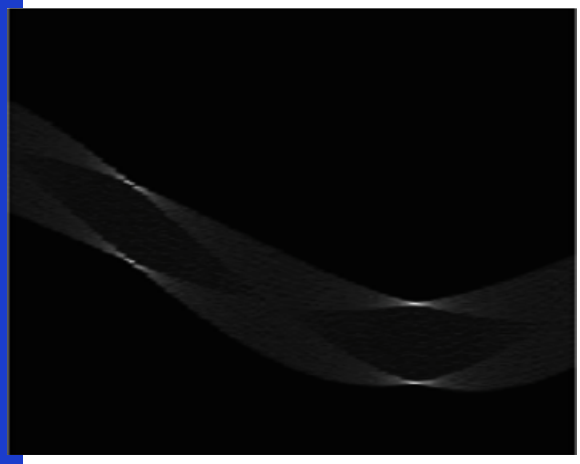
Each point will add a sinusoidal function in the (ρ,θ) parameter space

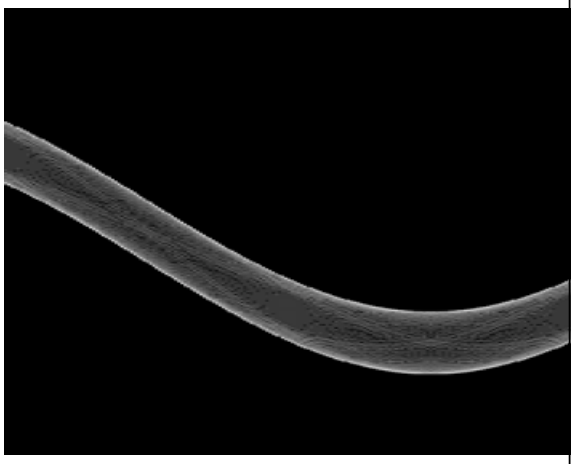
Computer Vision

Hough transform : straight lines

Square : 

Circle : 





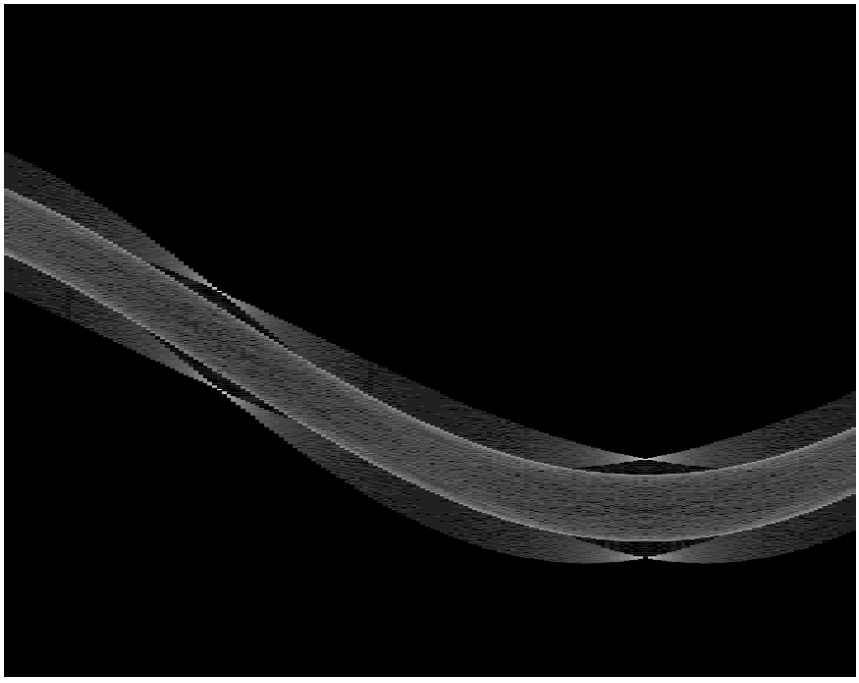
ρ

θ

Computer Vision

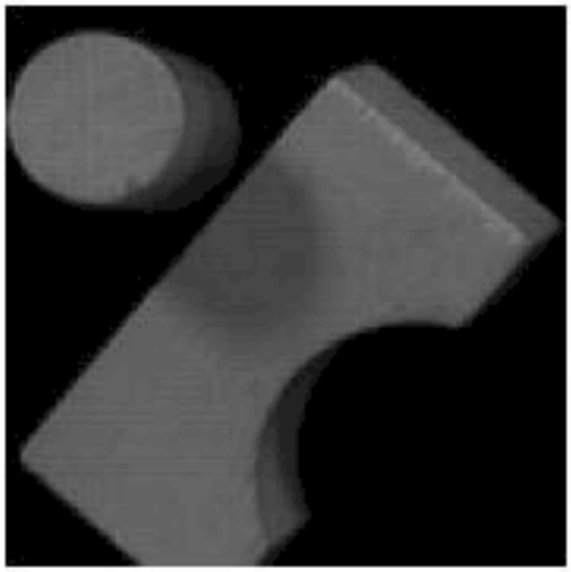
Hough transform : superposition

Combined image: 



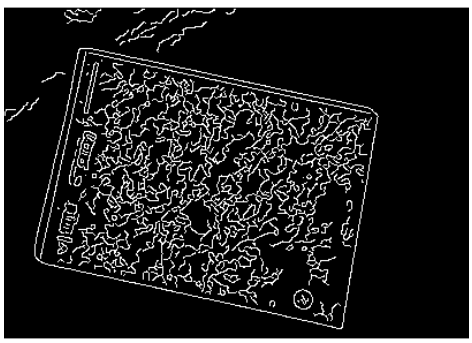
Computer Vision

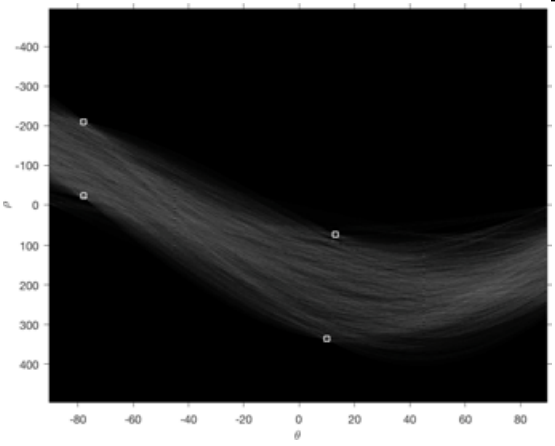
Hough transform : straight lines

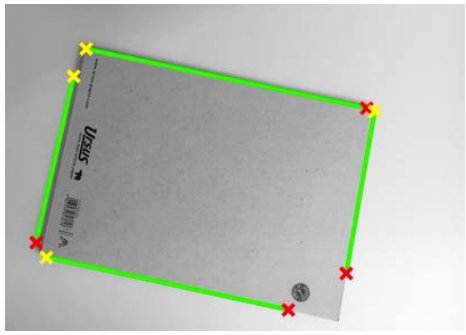


Computer Vision

Hough transform : detecting the book

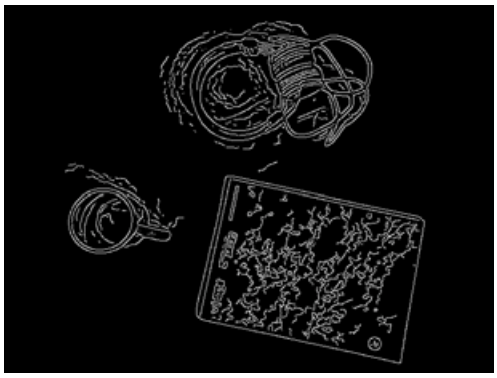


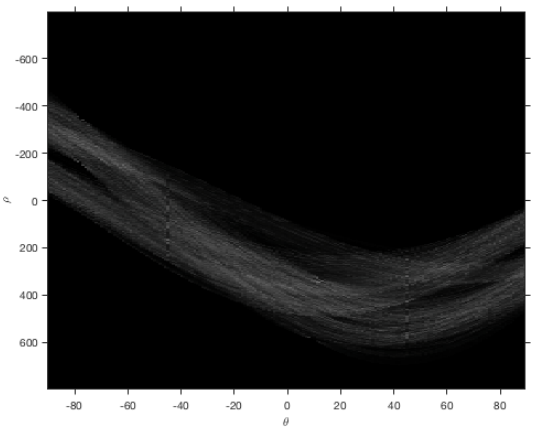


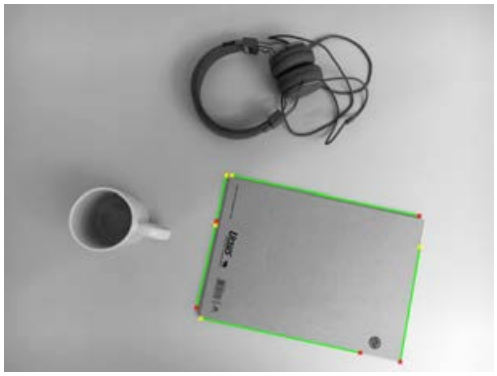


Computer Vision

Hough transform : works when other objects are present







Computer Vision

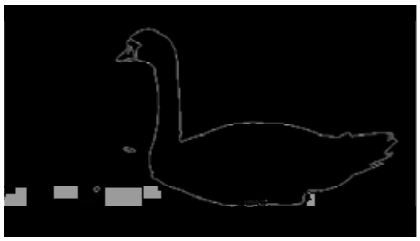
Hough transform : remarks

- 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…

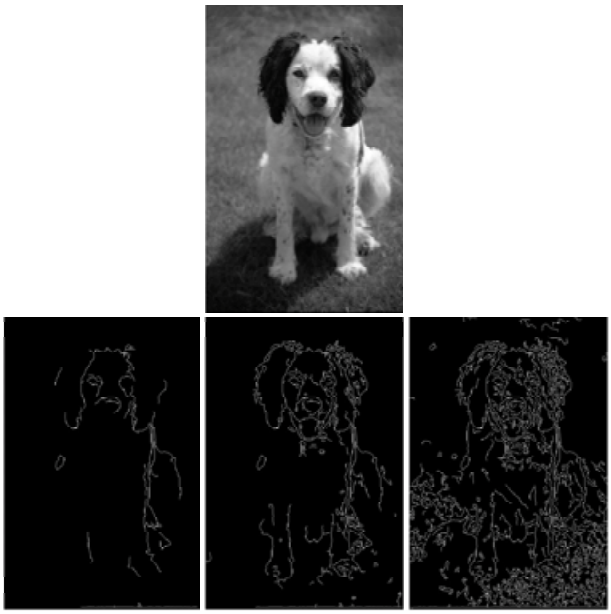
Computer Vision

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

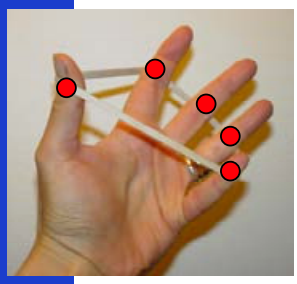
Images from D. Jacobs

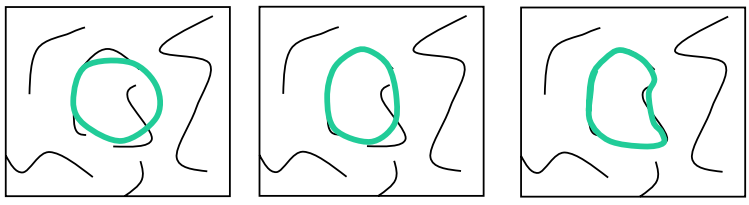
Computer Vision

Active contour models: **Snakes**

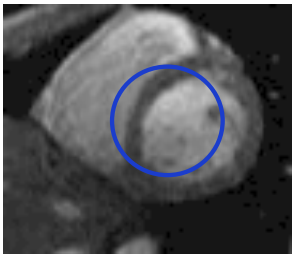
[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

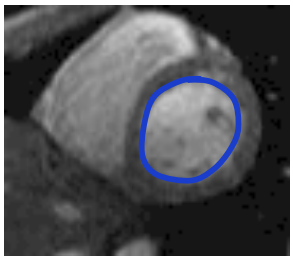
Intuition: an elastic band wrap around given structures while filling-in any missing parts (thanks to its continuous structure)





Idea: Evolve a contour iteratively to fit object boundary,
e.g. high image gradients





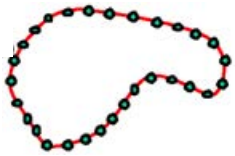
Implementation:

- Define a function for how good any given configuration is
- Iteratively optimize configuration

Computer Vision

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

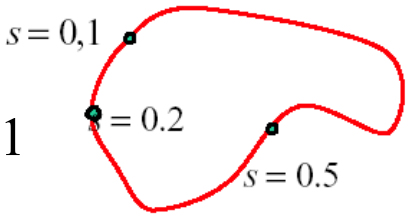
A **good** fit between the current snake and the target shape in the image will yield a **low** energy

Computer Vision

Parametric curve representation

Continuous case:

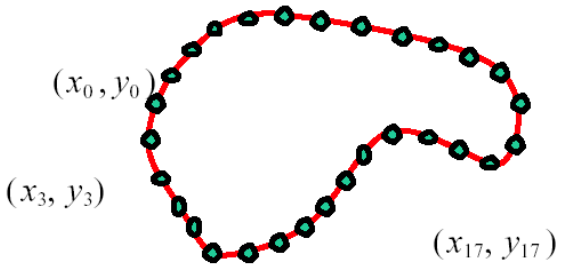
$$v(s) = (x(s), y(s)) \qquad 0 \le s \le 1$$



For numerical computation on the image

Discretization by a set of *n* points:

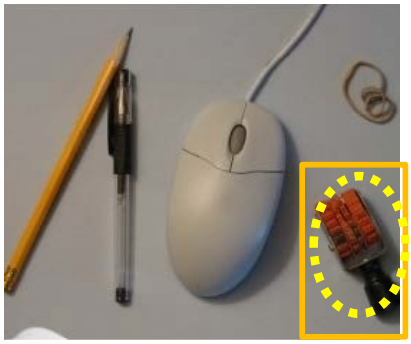
$$v_i = (x_i, y_i) \qquad i = 0 \ldots n-1$$

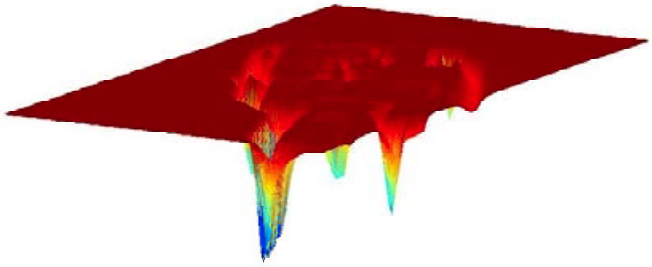


Computer Vision

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)

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

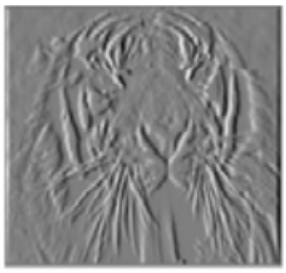
Defines how the image (edges) affect rubber band

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

Computer Vision

External (image) energy

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



$G_x(x,y)$

$G_y(x,y)$

- External energy at a point *v(s)* on the curve:

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

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

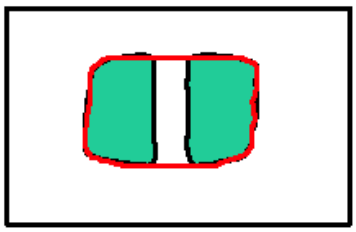
- External energy for the curve on discrete image

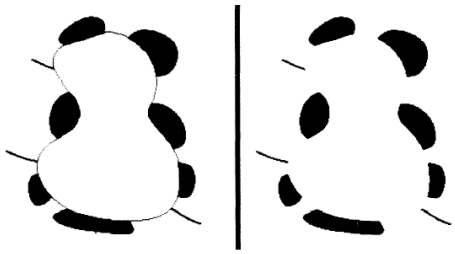
$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

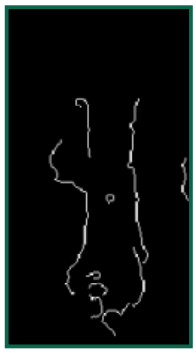
Computer Vision

Internal energy: intuition

Typical objects have continuous and smooth boundaries, i.e. low curvature







Computer Vision

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

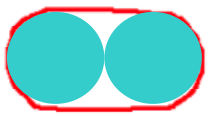
$$E_{internal} = \int_0^1 E_{internal}(v(s))\,ds$$

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

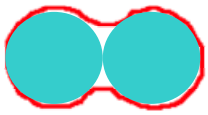
Models elasticity
Penalizes tension
(inhibits stretch)

Models stiffness
Penalizes curvature
(inhibits bending)

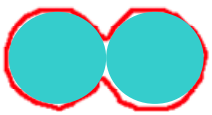
Examples after fitting:



β: large



medium



small

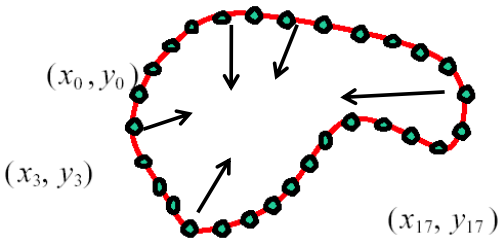
Computer Vision

Internal energy: Discretization

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

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

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

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

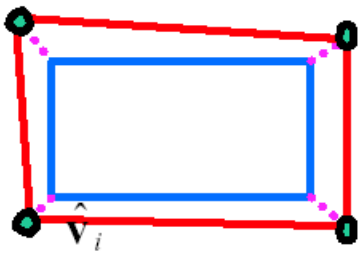
Encourages equal spacing but makes optimization harder

Computer Vision

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

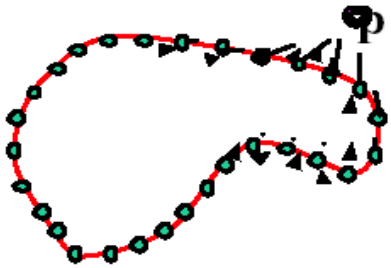
Computer Vision

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

Computer Vision

Energy minimization

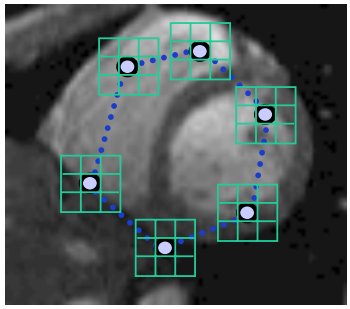
Several methods proposed to fit snakes, including methods based on :

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

Computer Vision

Greedy energy minimization

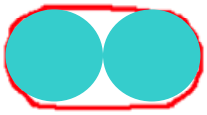
- For each point, place a search grid (e.g., 5x5) of discrete positions around it and pick the location where the energy function is minimal
- Keep applying this and circling around the curve
- Stop when a predefined number of points have not changed in the last iteration,
  or after max number of iterations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



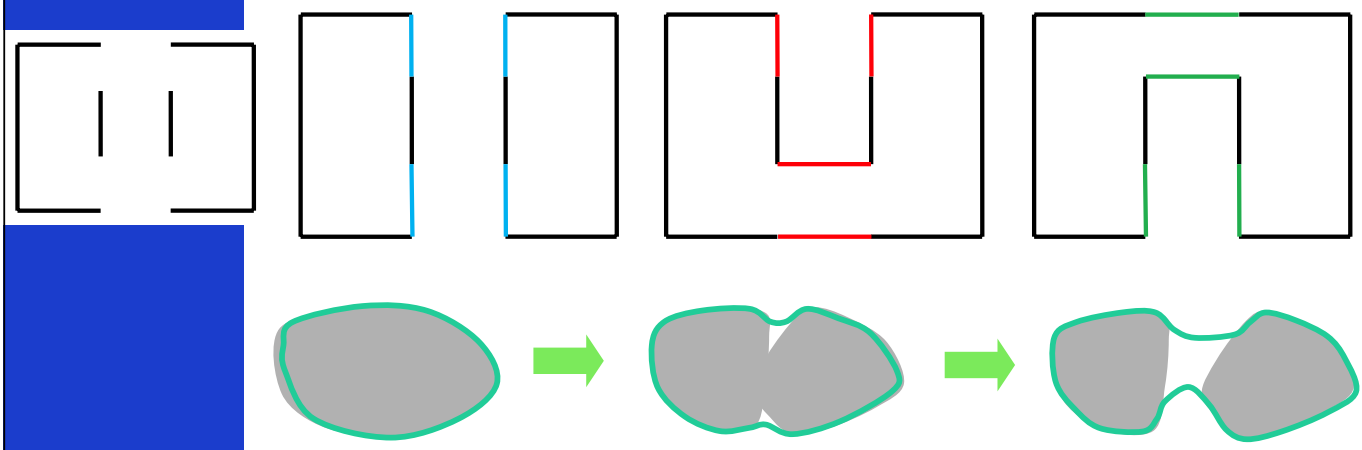
Computer Vision

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

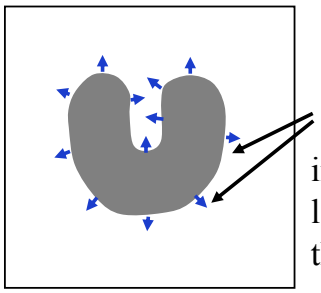


Computer Vision

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)

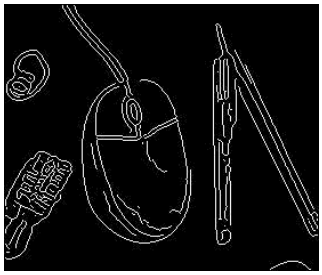


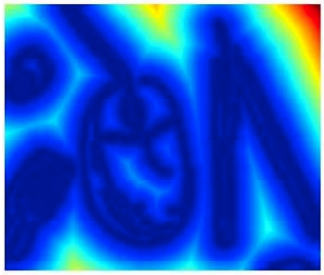
Potential remedy: External energy based on the **distance** from edges (makes snakes "farsighted")

Values tell how far each location is from nearest edge

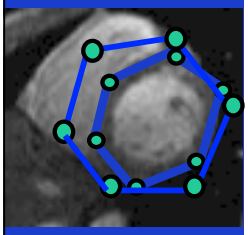

original


gradient


edges


distance map

Computer Vision



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:

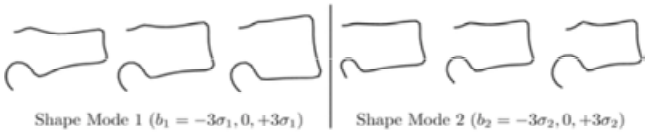
- Local optimization: may get stuck in local minimum Thus, needs good initialization near true boundary
- Susceptible to parameterization of energy function, must be set based on prior information, experience, etc.

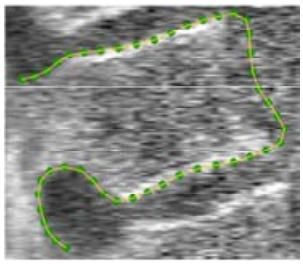
Computer Vision

Active Shape and Appearance Models

Shape model: Implicit energy via fitness to a statistical model

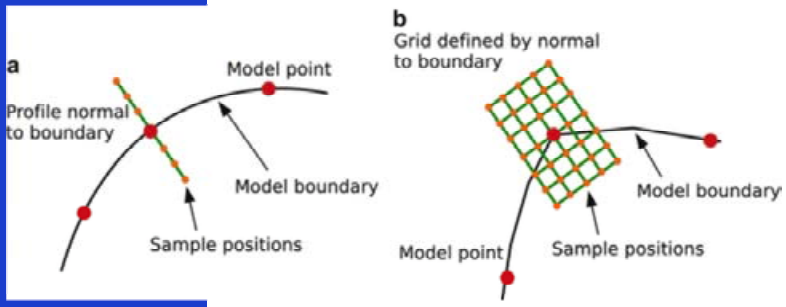
(remember PCA)

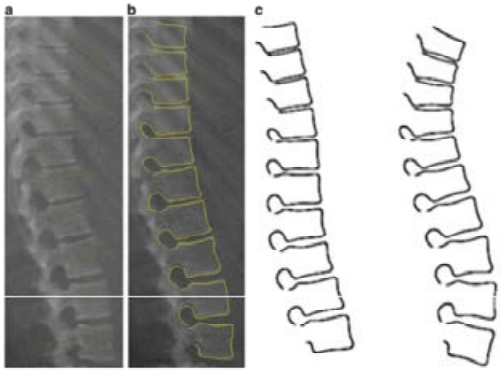




While updating the curve, project on model space to find closest shape model

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



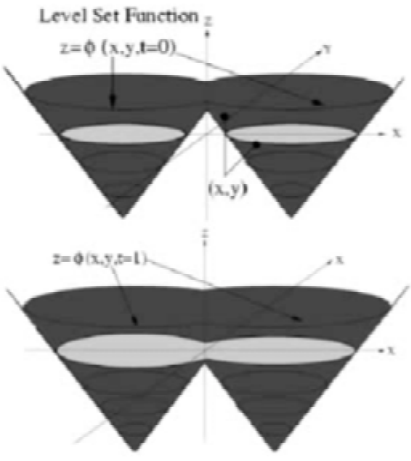


[Cootes et al., Handbook of Biomedical Imaging, 2015]

Computer Vision

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

Reading for those interested: Chan & Vese, Active Contours Without Edges, 2001

Computer Vision

Segmentation : Outline

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

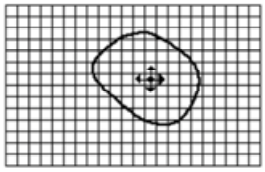
Computer Vision

Region growing : principle

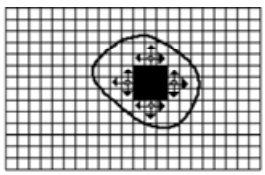
On the basis of segment homogeneity
(rather than inhomogeneity around edges)

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

These are grown pixel-by-pixel along borders considering some stopping (homogeneity) criterion



(a) Start of Growing a Region



(b) Growing Process After a Few Iterations

Choice of suitable homogeneity criterion is not straightforward

Computer Vision

Region growing : example

Seeds from low intensity variance are grown, for intensities between two (floating) thresholds and merging any similar segments

Computer Vision

Region growing : example

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

Computer Vision

Region growing : remarks

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

solution : split-and-merge procedures

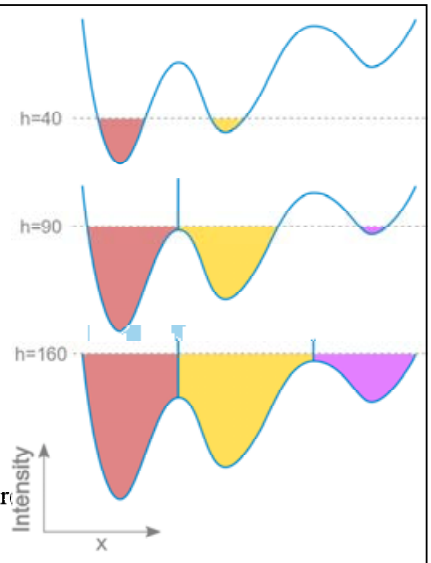
- merge similar touching regions (easy)
- split (more difficult, many ways to do)
  special representations, e.g. “quadtrees”, may help

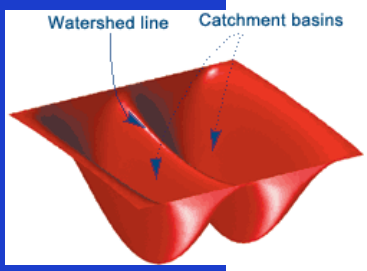
Region and edge based methods can be combined: hybrid approaches

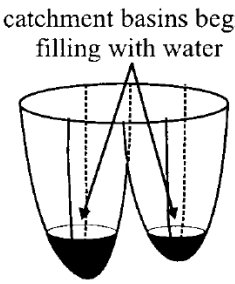
Computer Vision

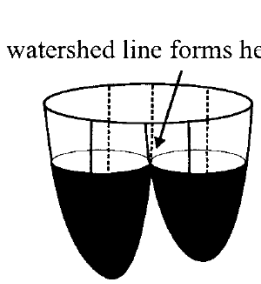
Watershed algorithm

Simulates the filling/drainage of an image topographical map, to find separated water catchment basins, e.g. (touching) objects







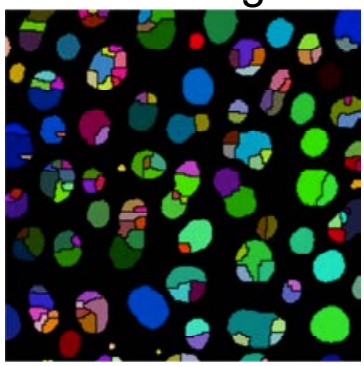


- Starting with local minima
- Different implementations of water filling
  (usually based on some type of region growing)
  Stop propagation at detected ridges (lines)
- Usually strong over-segmentation (many segments)

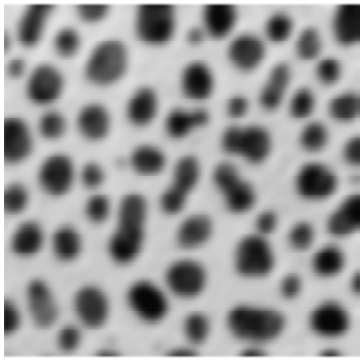
Computer Vision

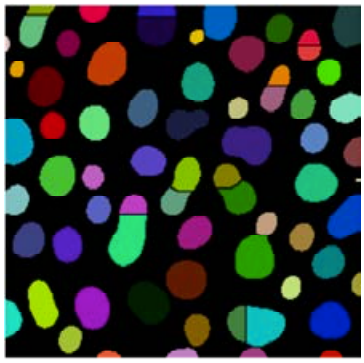
Watershed segmentation: Example

Watershed on original image



Watershed on Gaussian blurred image





Computer Vision

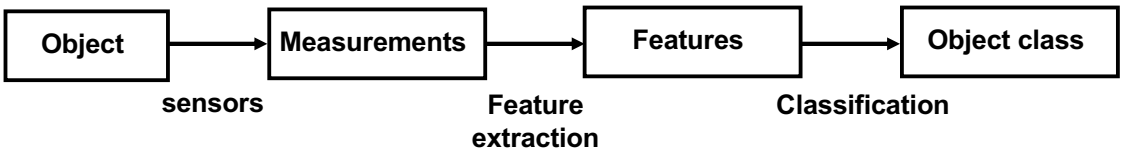
Segmentation : Outline

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

Computer Vision

Statistical pattern recognition

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



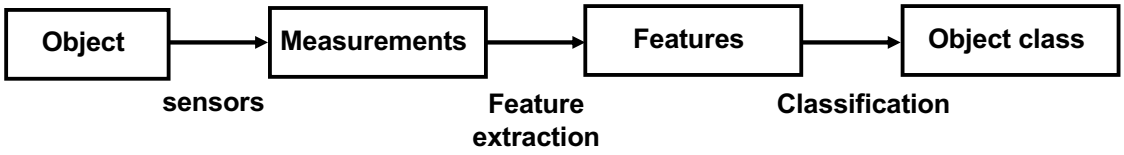
Three alternatives:

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

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

Computer Vision

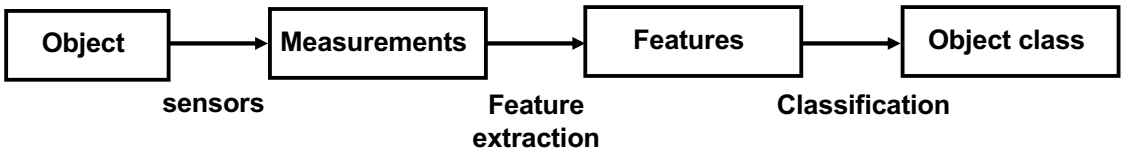
Example application : 1



Objects: Frog
Sensors: Camera
Measurements: Pixels intensity
Features: Color
Object classes: Foreground / Background

Computer Vision

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

Computer Vision

Basic Notation

The set of all possible classes:

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

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

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

Examples of features:
- Color – 3 dimensional
- X-ray attenuation – 1 dimensional
- Pixel location – 2 dimensional
- Combinations…

Computer Vision

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)

Computer Vision

Unsupervised clustering: K-means Algorithm

A popular and widely used algorithm.

Given *M* measurements (samples) of length-*n* features:

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

Choose K centers / means

$$m_i \in \mathbb{R}^n, \;\; i = 1, \ldots, K$$

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

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

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

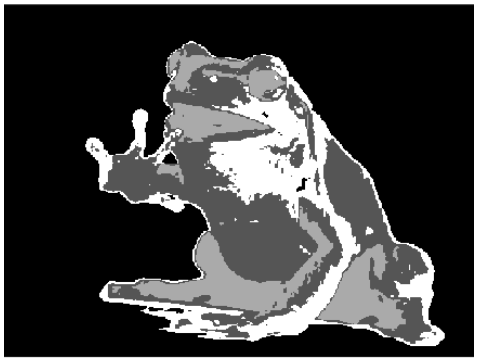
Computer Vision

K-means analysis

(for K classes)

K=2

K=3



K=4

Computer Vision

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

Computer Vision

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

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

$P \equiv p$

Computer Vision

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) → Bayes' theorem

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

Computer Vision

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

Computer Vision

Supervised Generative Models : principle

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

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

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

$\vec{v}$: grey level intensity

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

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

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

Computer Vision

a-posteriori distribution

Compute posterior distributions via Bayes theorem:

Can be learnt Prior

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

Constant

Final segmentation by maximum a-posterior (MAP)

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

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

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

Computer Vision

Importance of distributions

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

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

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

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

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

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

Computer Vision

Likelihood instead of posterior

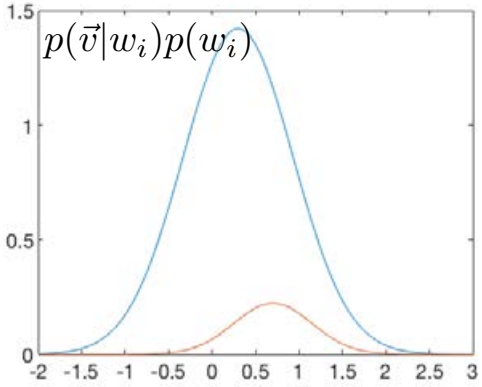
$$P(w_i|\vec{v}) = \frac{p(\vec{v}, w_i)}{p(\vec{v})} = \frac{p(\vec{v}|w_i)P(w_i)}{p(\vec{v})}$$

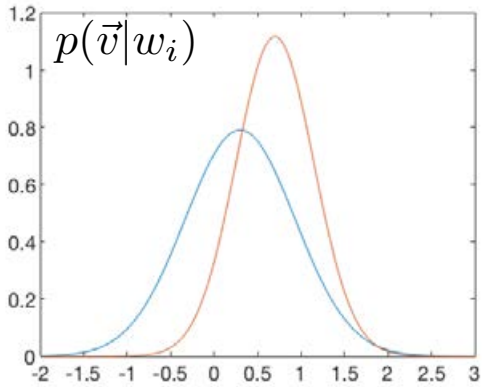
Posterior distribution can be difficult to compute

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

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

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





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

Computer Vision

Remarks on Generative Supervised Models

- Mathematically sound
- Can be extended to various features
- Flexible and generic
- Links to Bayesian methods

- Features are very important
- Optimization and inference can be hard
- For high dimensional features
  estimation of distributions is problematic
  (use of latent variables)

Computer Vision

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

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

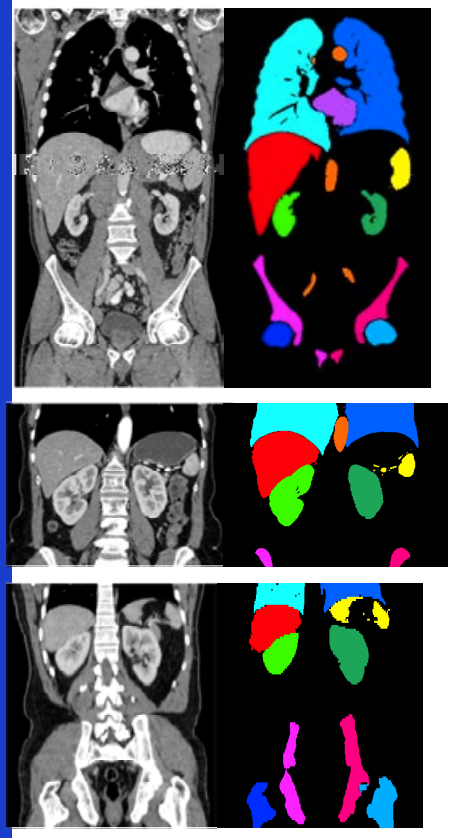
In terms of probabilities, this is similar to modeling posterior or its maximum

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

But, we “learn” a direct “mapping” from examples

Computer Vision

Discriminative learning: 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

Computer Vision

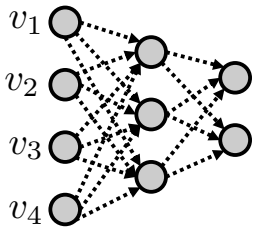
Discriminative learning: Finding a mapping

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

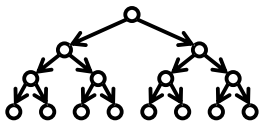
- Mapping is a parametric model
- KNN – K-nearest neighbors
- Logistic regression (of binary class)

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

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



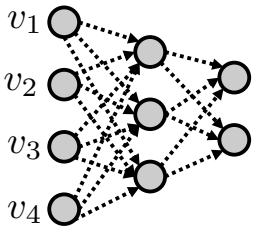
- Decision trees - random forests
- …



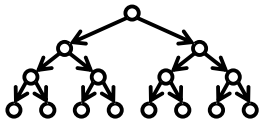
Computer Vision

Discriminative learning: Learning / **Training** Phase

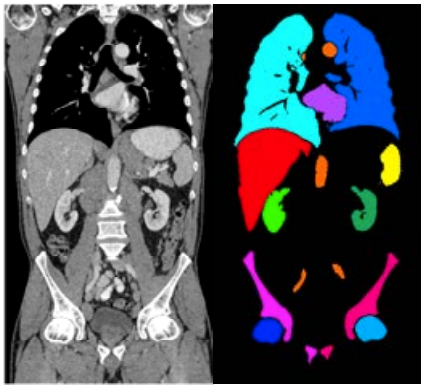
$$c_j = \frac{1}{1+\exp\left(-\beta_0 - \beta^T \vec{v}\right)}$$



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



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

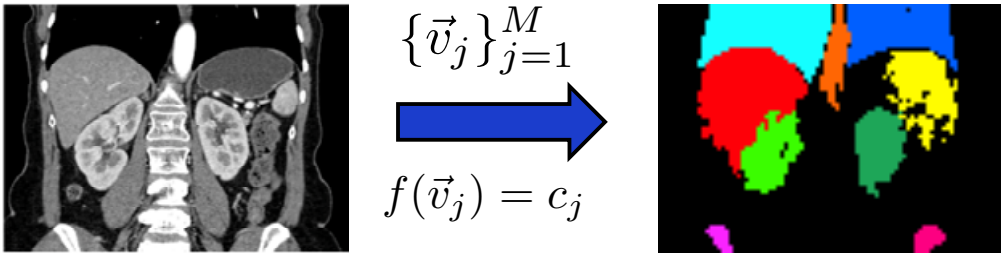


Training data: $\{(\vec{v}_n, c_n)\}_{n=1}^{N}$

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

Computer Vision

Discriminative learning: Segmentation **Testing** phase

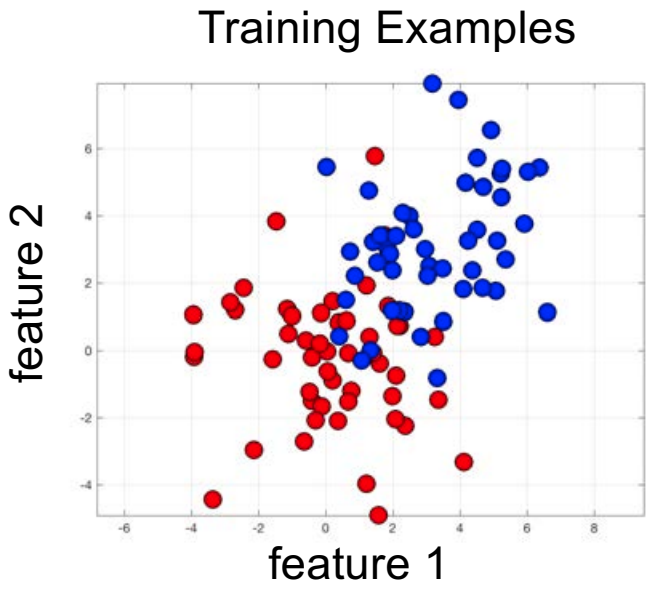


- Extract the features used during training
- Use the mapping learned before

Computer Vision

K-nearest neighbors - KNN

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



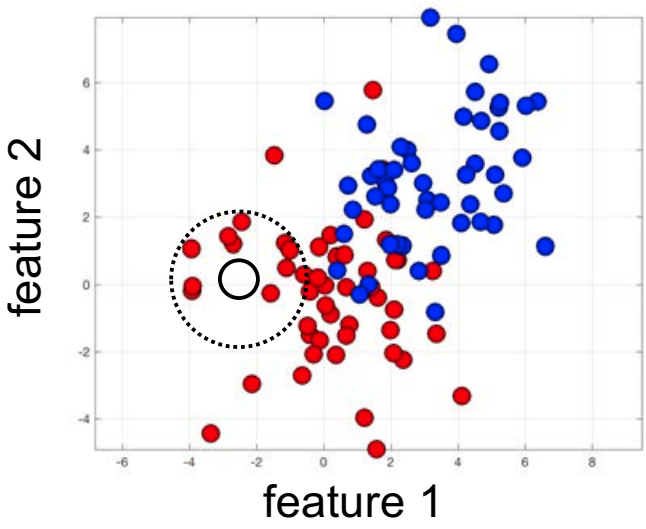
Computer Vision

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

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

Computer Vision

K-nearest neighbors – KNN: parameterization

- Define the term “nearest” → distance
- Define the mapping $f(\vec{v}) = f(c_1, \ldots, c_K)$
  - 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…

Computer Vision

Remarks on KNN

Pros:

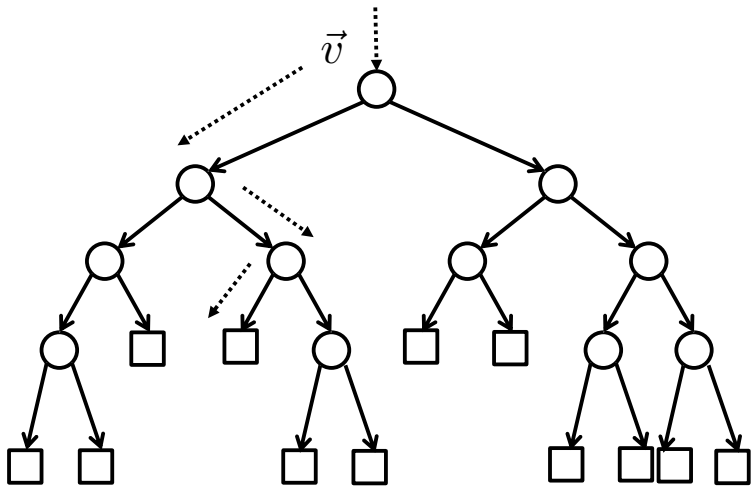
- Very simple to implement
- Very simple to understand
- Efficient implementations possible
  approximate nearest neighbors,…
- Distance definition is flexible

Cons:

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

Computer Vision

Discriminative learning: Decision Trees



At each internal node n, there is a binary question on the features

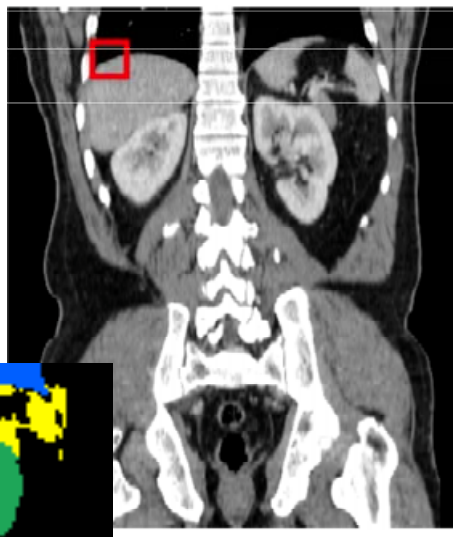
$$f_n(\vec{v}) = \begin{cases} 0 \rightarrow & \text{go left} \\ 1 \rightarrow & \text{go right} \end{cases}$$

At each leaf node, there is a prediction and it changes from leaf node to leaf node

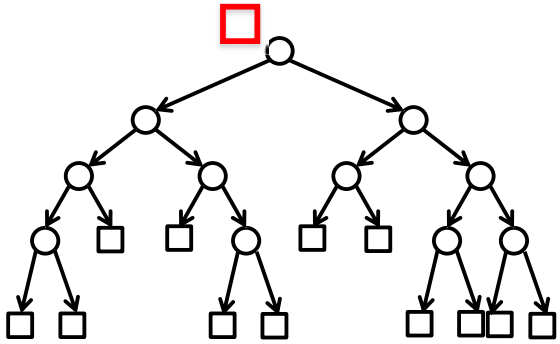
$$f_l(\vec{v}) = c$$

Computer Vision

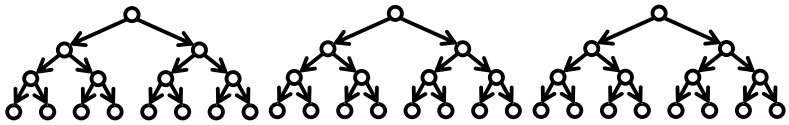
Random forests: During Segmentation



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



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



Computer Vision

Random Forests: remarks

Pros:

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

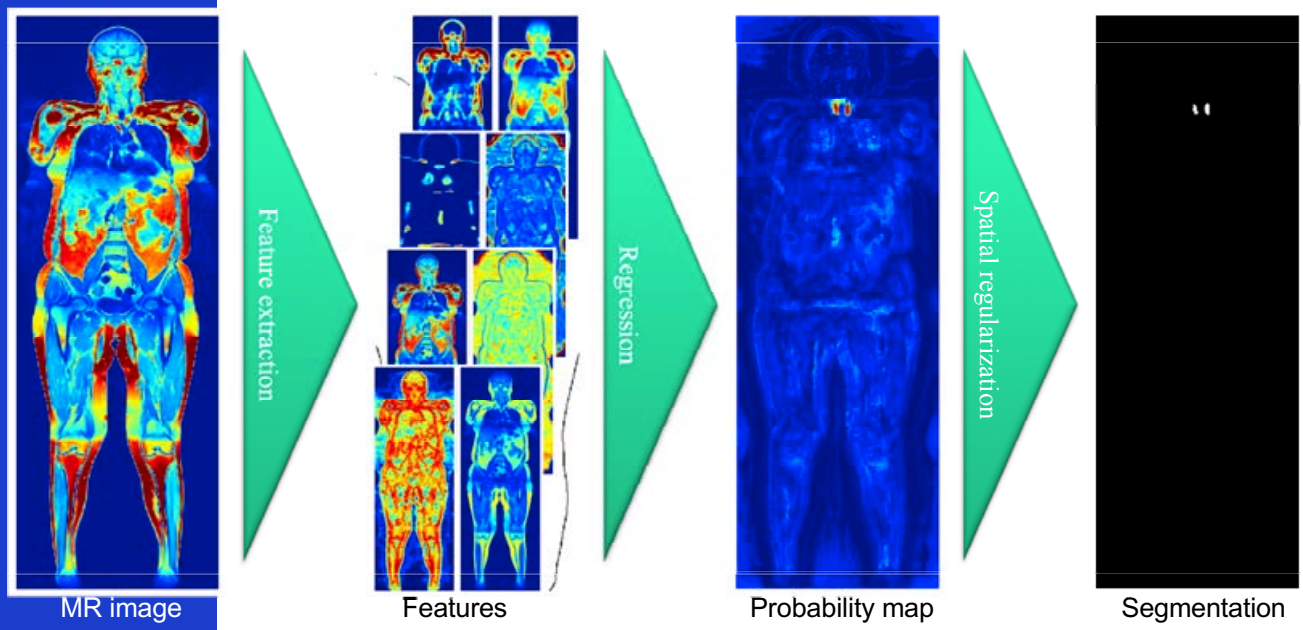
Cons:

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

Computer Vision

Example: Illustrating pipeline

Directly classify from features,
or alternatively regress to some probability map for further processing



Computer Vision

Summary of Segmentation

- Voxel-based:
  - Thresholding: Otsu (automatic threshold determination)
  - Individual voxel decisions often need post-processing with morphological operators
- Edge-based:
  - Hough transform: Parametric models to vote for shapes in lower dimensional space
  - Elastically deformable shapes: Iteratively update a contour to fit borders
  - Connected components:
- Region based:
  - Region growing: Starts from seeds, grow homogeneous regions (splitting/merging/…)
  - Watershed: Simulate flooding, e.g., gradient magnitude as natural borders/ridges
- Pattern Recognition based

Computer Vision

Appendix

Computer Vision

Outline

- **Supervised generative learning:**
  From individual pixels to combinations
  Markov Random Fields
  Gibbs sampling
  Graph-cuts

- **Supervised discriminative learning for segmentation**
  KNN
  Random Forests

Computer Vision

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?

Computer Vision

From individual pixels to combinations: general formulation

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

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

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

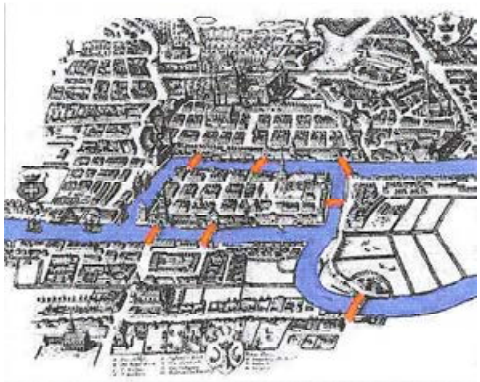
Joint distribution when pixels are not independent

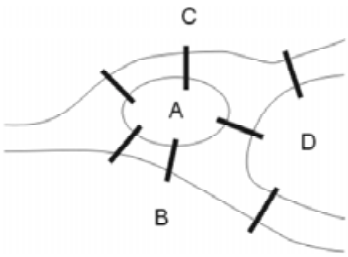
$$p(\mathbf{c}, \vec{\mathbf{v}}) = \prod_{j=1}^{M} p(\vec{v}_j|c_j)P(\mathbf{c})$$

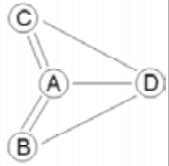
Computer Vision

Topology

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





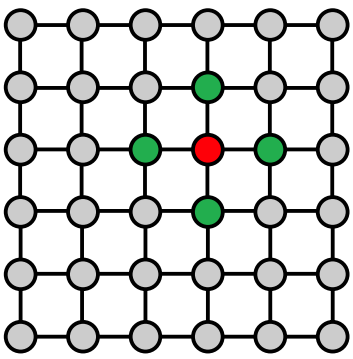


Computer Vision

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

Computer Vision

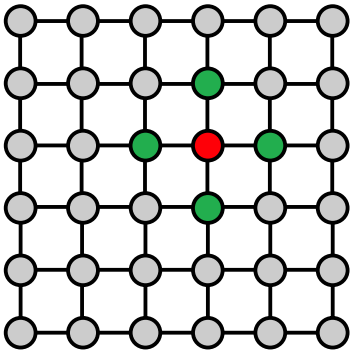
Markov Random Fields: Energies and Gibbs distributions

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

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

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

$$d(c_j, c_k) : \text{distance between } c_j \text{ and } c_k$$



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!

Computer Vision

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**

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

Computer Vision

Optimization

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

Equivalently

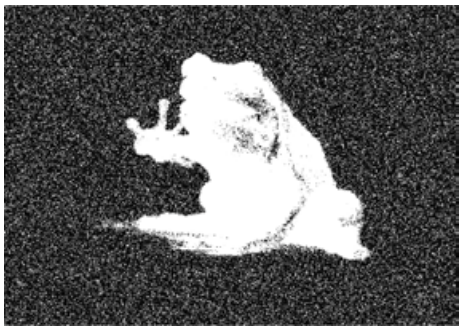
$$\arg_{\mathbf{c}} \min \sum_{j=1}^{M} (\vec{v}_j - \mu_{c_j})^T \Sigma_{c_j}^{-1} (\vec{v}_j - \mu_{c_j}) + \lambda \sum_{j=1}^{M} \sum_{k \in G_j} \delta(c_j \neq c_k)$$

- 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]
  …

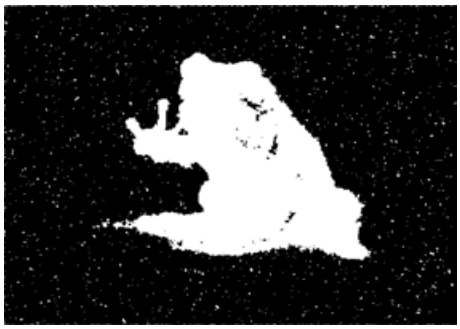
Computer Vision

Stochastic Relaxation / Gibbs Sampling: in action

Noisy image



Initial segmentation



After 1 iteration of Gibbs

After 25 iterations

Computer Vision

Stochastic Relaxation / Gibbs Sampling: analysis with respect to λ

Noisy image

$\lambda = 0$

$\lambda = 5$

$\lambda = 20$

Computer Vision

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