Image Segmentation

Vittorio Ferrari

ETH Zurich - HS 2019

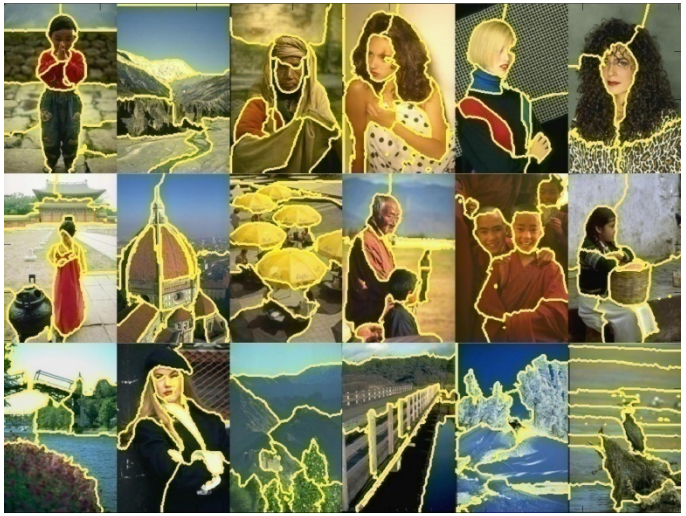
Slide credits:

K. Grauman, B. Leibe, S. Lazebnik, S. Seitz,
Y Boykov, W. Freeman, P. Kohli, L. Van Gool, and myself ;)

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

Examples of Grouping in Vision



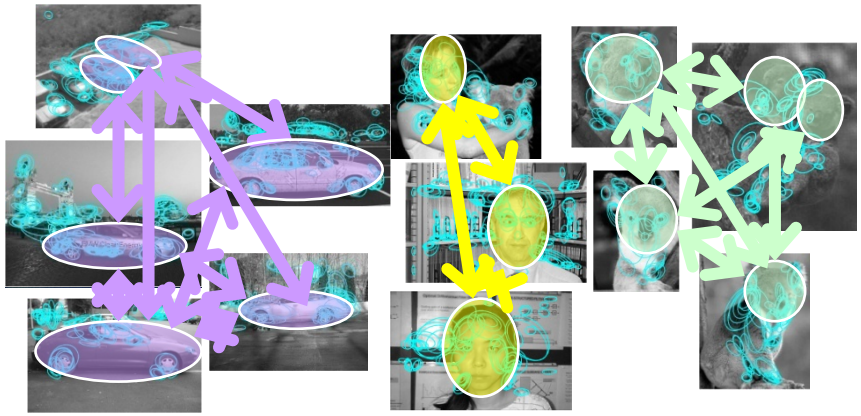
Determining image regions

Grouping video frames into shots

Figure-ground

What things should be grouped?

What cues indicate groups?



Object-level grouping

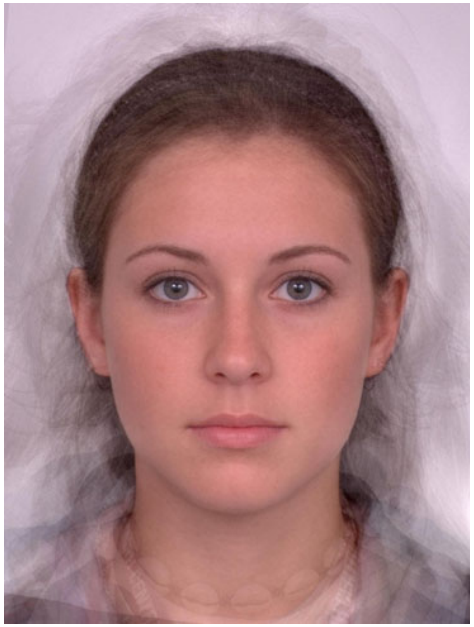
Slide credit: Kristen Grauman

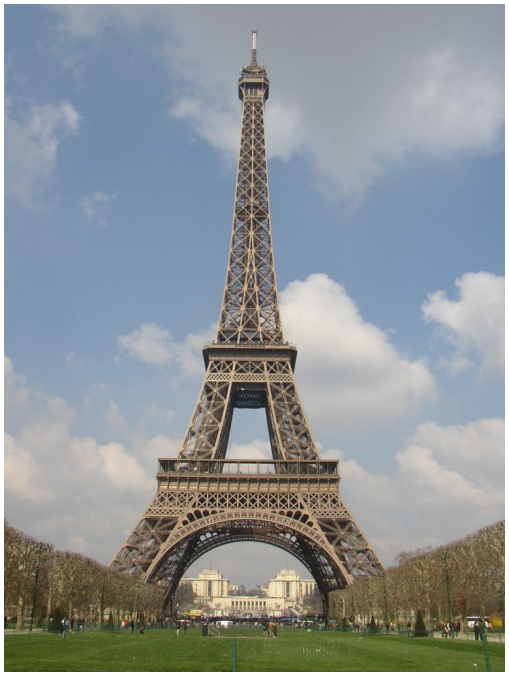
Similarity in appearance

Slide adapted from Kristen Grauman

http://chicagoist.com/attachments/chicagoist_alicia/GEESE.jpg, http://wwwdelivery.superstock.com/WI/223/1532/PreviewComp/SuperStock_1532R-0831.jpg

Symmetry





Slide credit: Kristen Grauman

http://seedmagazine.com/news/2006/10/beauty_is_in_the_processingtim.php

Common Fate

Image credit: Arthus-Bertrand (via F. Durand)

Slide credit: Kristen Grauman

Proximity

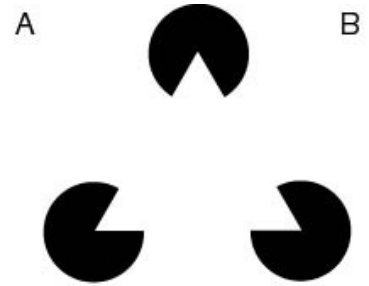
Slide credit: Kristen Grauman

http://www.capital.edu/Resources/Images/outside6_035.jpg

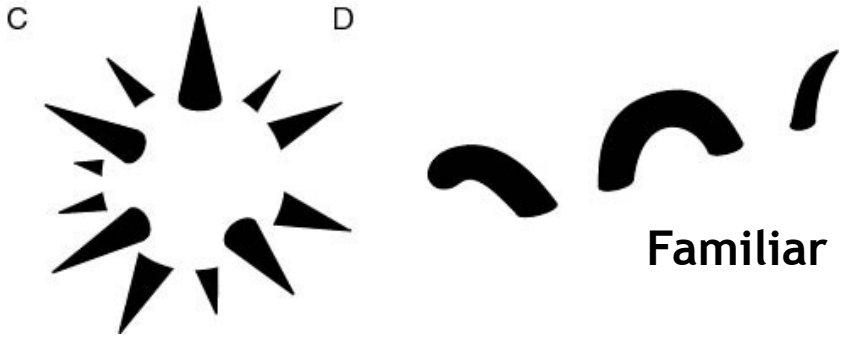
The Gestalt School

- Grouping is key to visual perception
- Elements in a collection can have properties that result from relationships
  - The whole is greater than the sum of its parts”

Illusory/subjective contours



Occlusion



Familiar configuration

http://en.wikipedia.org/wiki/Gestalt_psychology

Slide credit: Svetlana Lazebnik

Image source: Steve Lehar

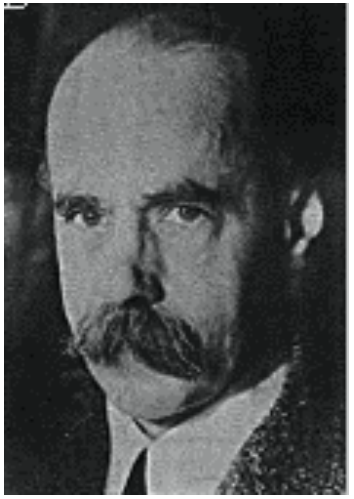
Gestalt Theory

- **Gestalt: whole or group**
  - **Whole is greater than sum of its parts**
  - **Relationships among parts can yield new properties/features**

- **Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)**

"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."

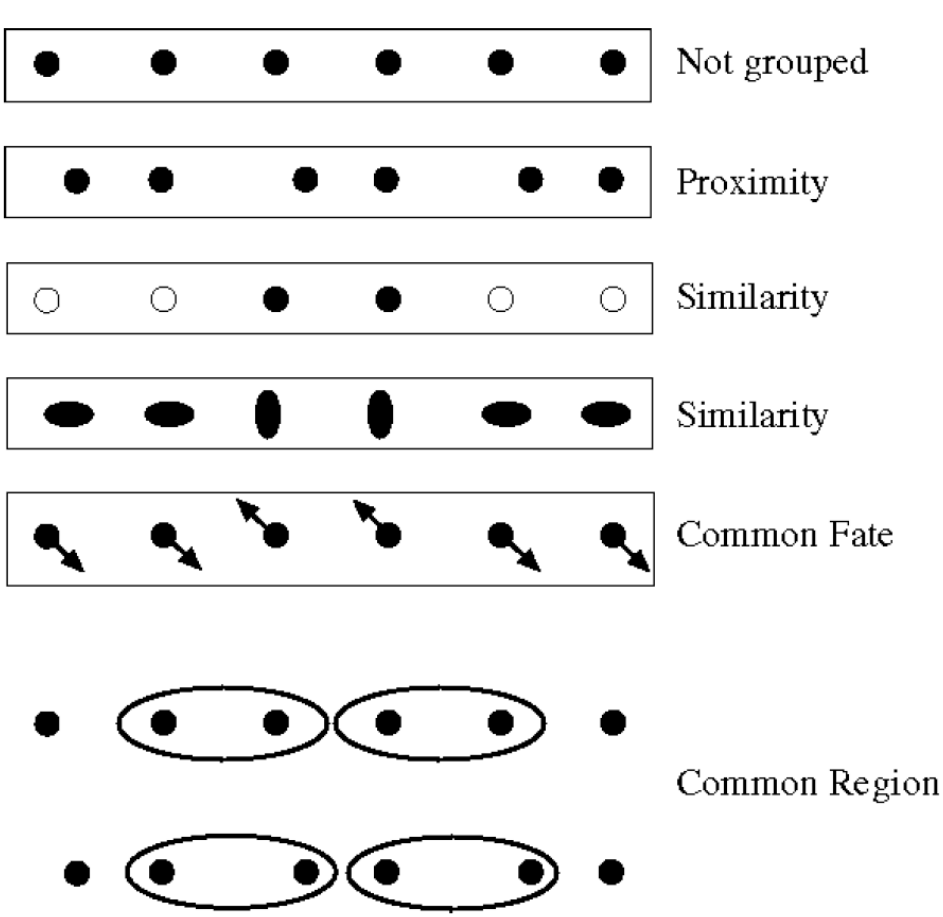
Max Wertheimer
(1880-1943)

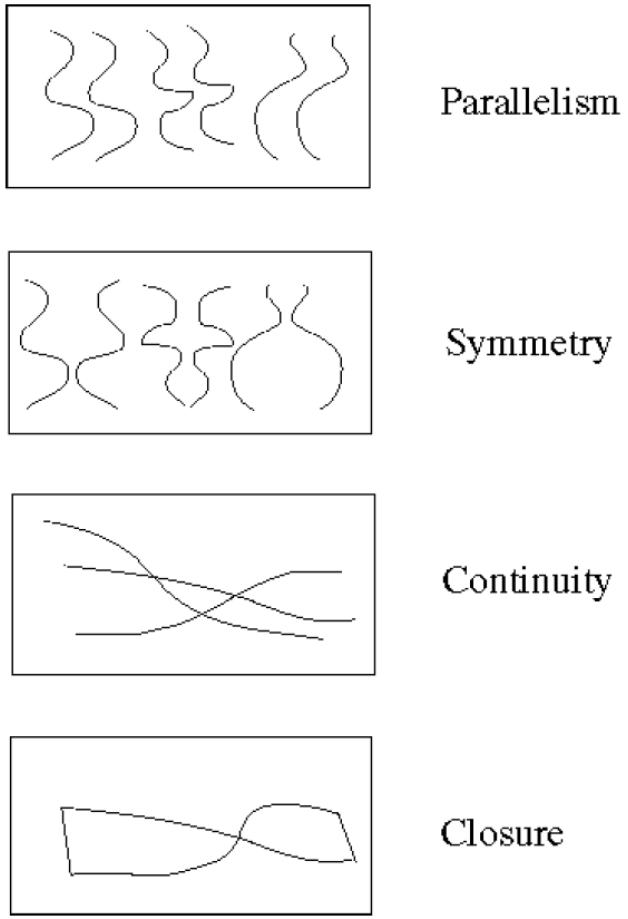


Untersuchungen zur Lehre von der Gestalt,
***Psychologische Forschung*, Vol. 4, pp. 301-350, 1923**
http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm

Slide credit: B. Leibe

Gestalt Factors





These factors make intuitive sense, but are very difficult to translate into algorithms.

Slide credit: B. Leibe

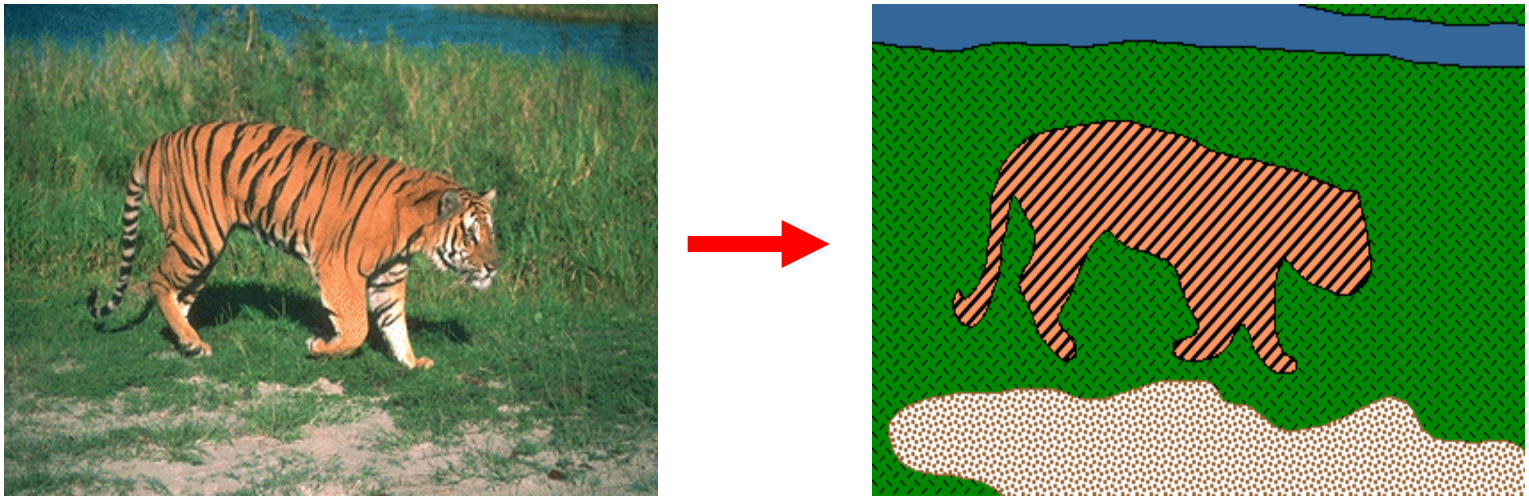
Image source: Forsyth & Ponce

The Ultimate Gestalt test

Slide adapted from B. Leibe

Image Segmentation

- **Identify groups of pixels that belong together**

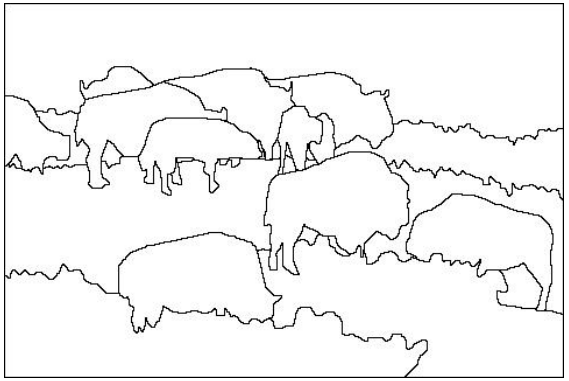


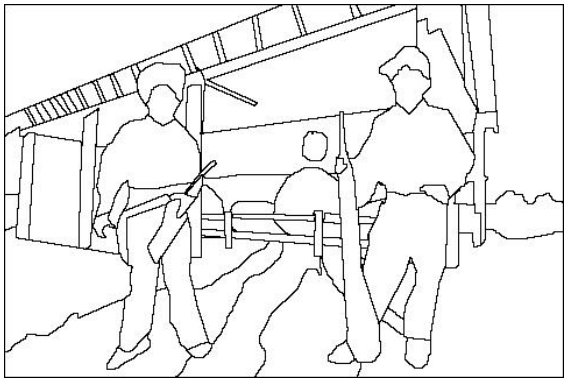
Slide credit: Steve Seitz, Kristen Grauman

The Goals of Segmentation

- **Delineate objects and background regions**

Image | **Human segmentation**

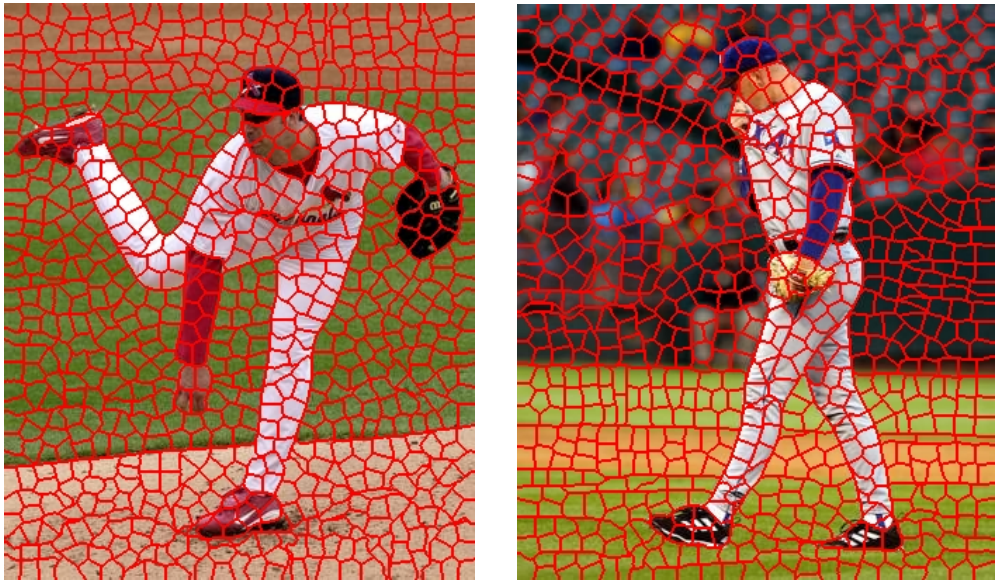
Slide credit: Svetlana Lazebnik

The Goals of Segmentation

- Delineate objects and background regions

- Group similar-looking pixels for efficiency of further processing

"superpixels"



X. Ren and J. Malik. Learning a classification model for segmentation. ICCV 2003.

Slide credit: Svetlana Lazebnik

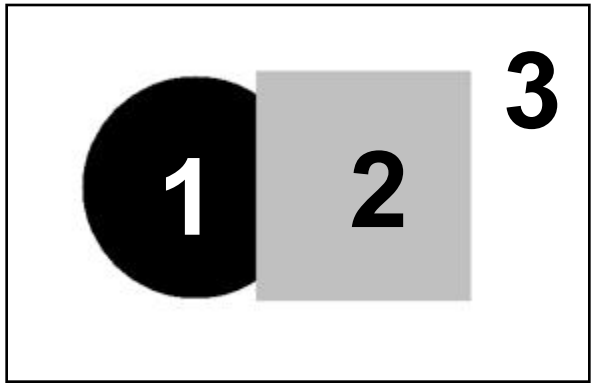
Recognition and segmentation

- They go hand in hand in the human visual system
- In computer vision traditional techniques are bottom-up, but there are several joint approaches
- We are going to see some deep learning approaches to this in the object class recognition lecture

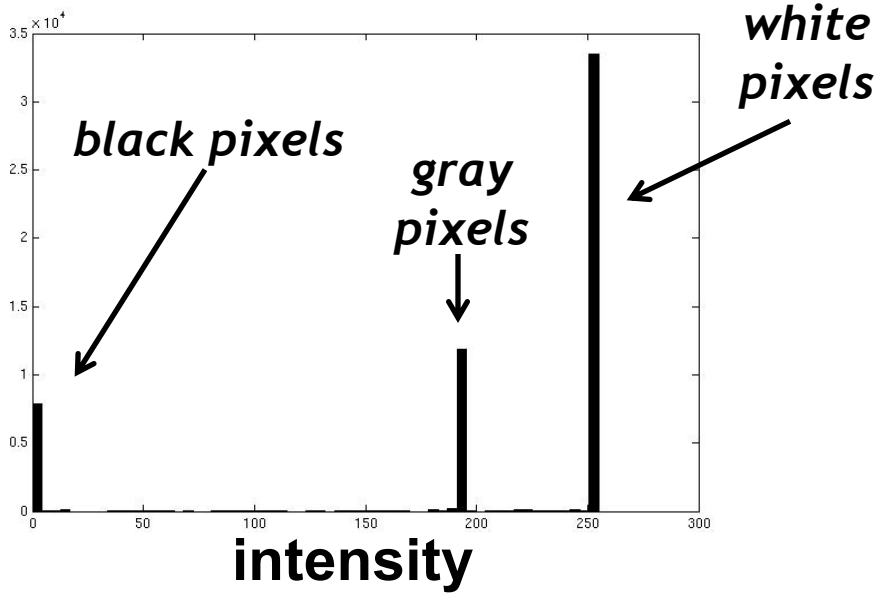
Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

Image Segmentation: Toy Example

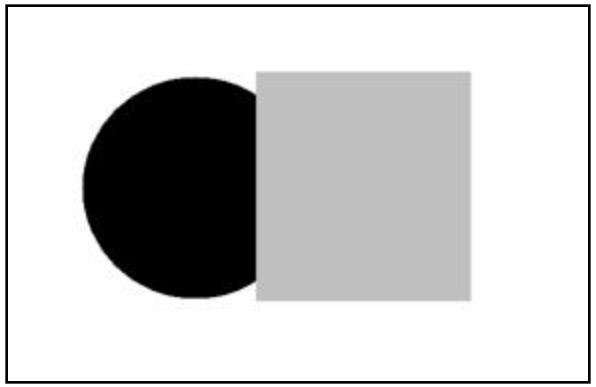


input image

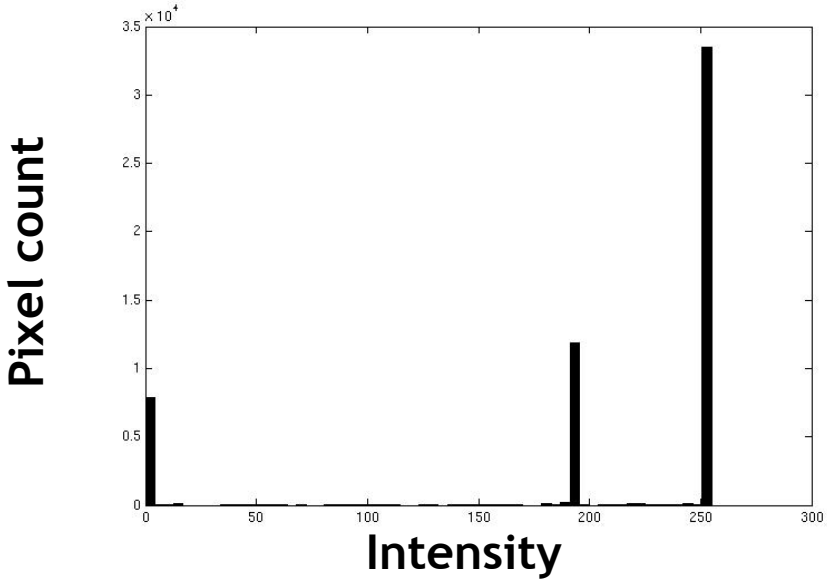


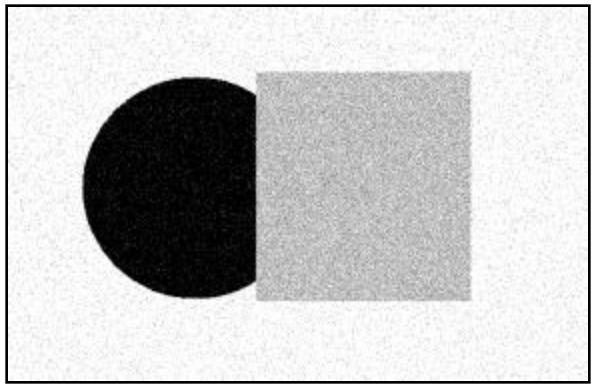
- These intensities define the three groups.
- We could label every pixel in the image according to which of these it is, i.e. segment the image based on the intensity feature.
- What if the image isn't quite so simple?

Slide credit: Kristen Grauman

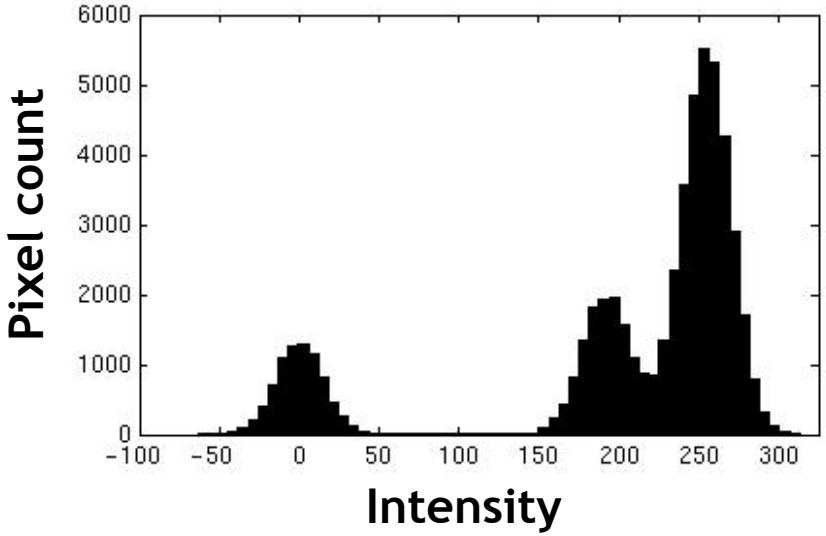


Input image

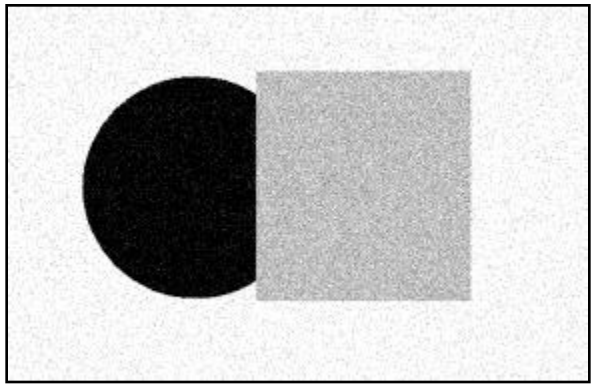




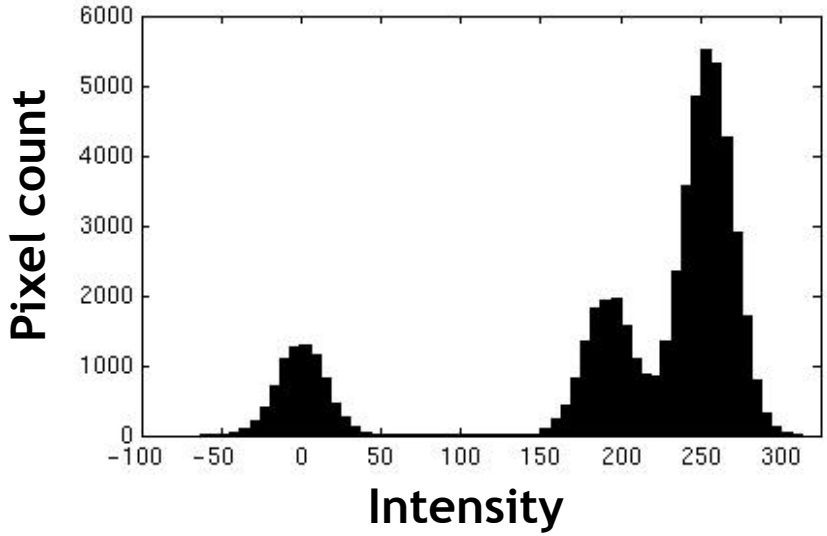
Input image



Slide credit: Kristen Grauman

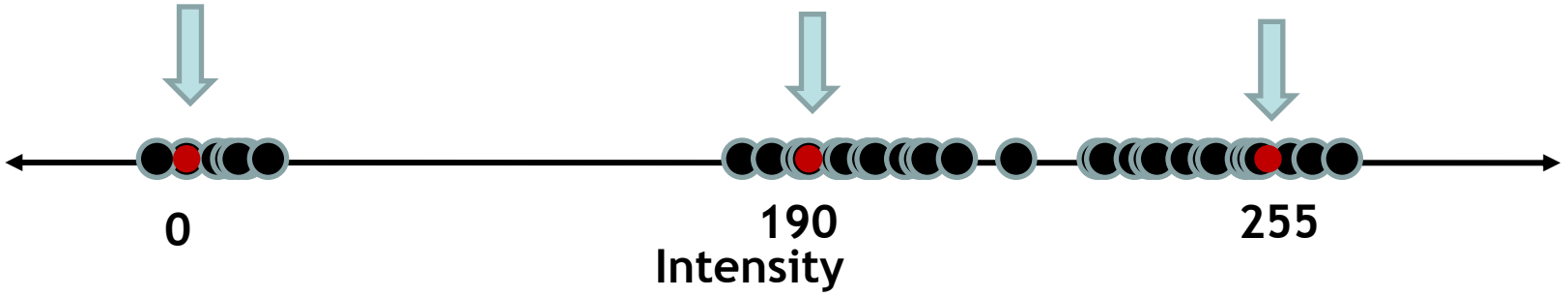


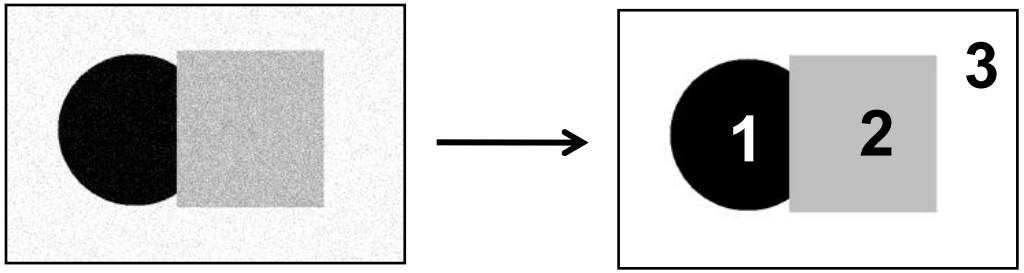
Input image



- Now how to determine the three main intensities that define our groups?
- We need to cluster.

Slide credit: Kristen Grauman





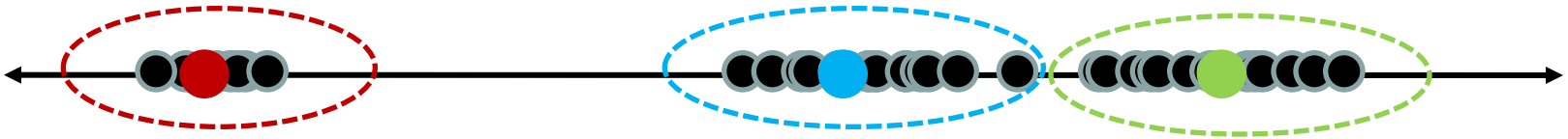
- **Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.**
- **Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:**

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

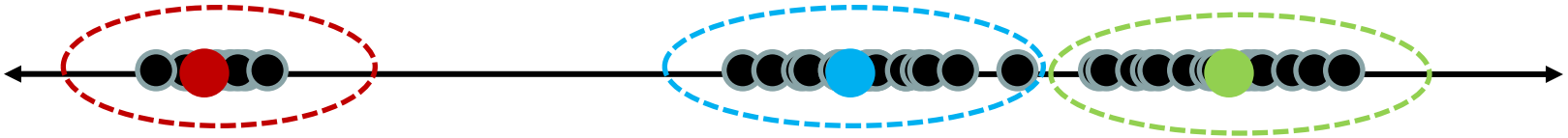
Slide credit: Kristen Grauman

Clustering

- With this objective, it is a "chicken and egg" problem:
  - If we knew the *cluster centers*, we could allocate points to groups by assigning each to its closest center.



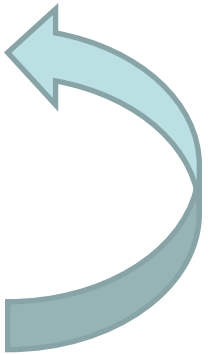
  - If we knew the *group memberships*, we could get the centers by computing the mean per group.



Slide credit: Kristen Grauman

K-Means Clustering

- **Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.**
  1. **Randomly initialize the cluster centers, $c_1, \ldots, c_K$**
  2. **Given cluster centers, determine points in each cluster**
     - **For each point p, find the closest c_i. Put p into cluster i**
  3. **Given points in each cluster, solve for c_i**
     - **Set c_i to be the mean of points in cluster i**
  4. **If c_i have changed, repeat Step 2**



- **Properties**
  - **Will always converge to *some* solution**
  - **Can be a "local minimum"**
    - **Does not always find the global minimum of objective function:**

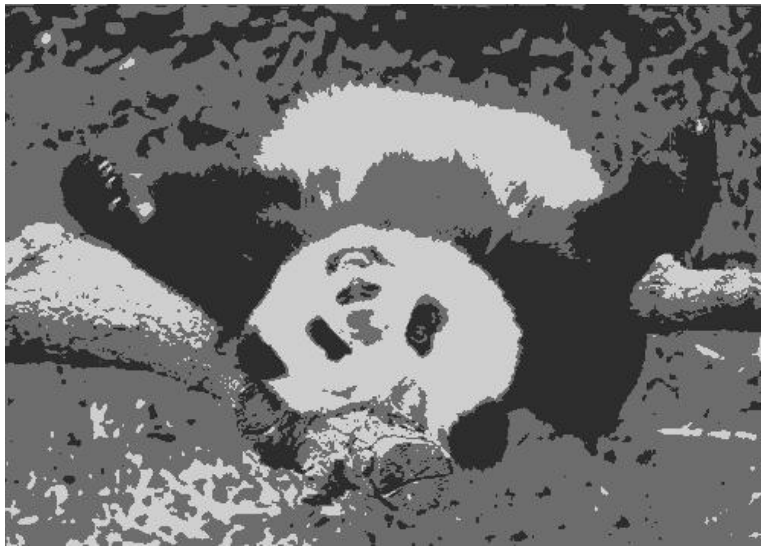
$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Slide credit: Steve Seitz

Segmentation as Clustering

K=2

K=3



```
img_as_col = double(im(:));
cluster_membs = kmeans(img_as_col, K);

labelim = zeros(size(im));
for i=1:k
   inds = find(cluster_membs==i);
   meanval = mean(img_as_column(inds));
   labelim(inds) = meanval;
end
```

Slide credit: Kristen Grauman

K-Means Clustering

- Java demo:

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

Feature Space

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on intensity similarity

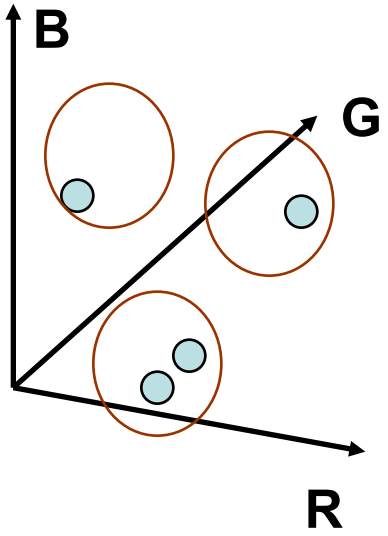
- Feature space: intensity value (1D)

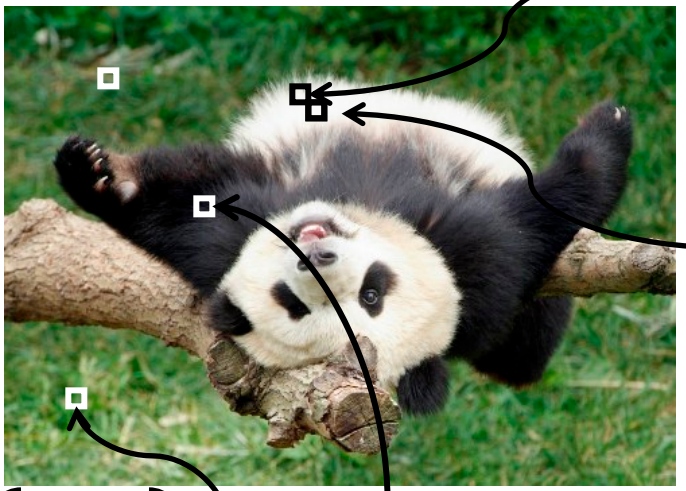
Slide credit: Kristen Grauman

Feature Space

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on color similarity





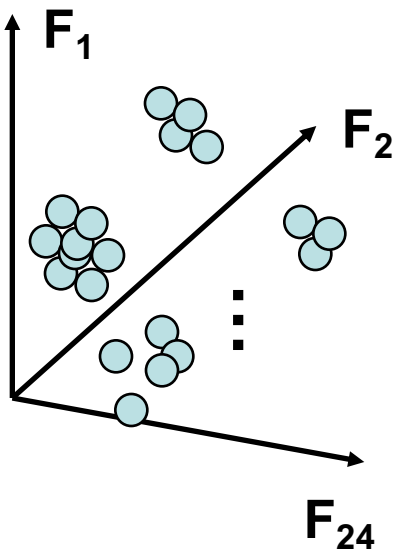
- Feature space: color value (3D)

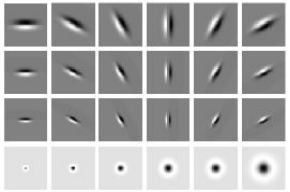
Slide credit: Kristen Grauman

Feature Space

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on texture similarity





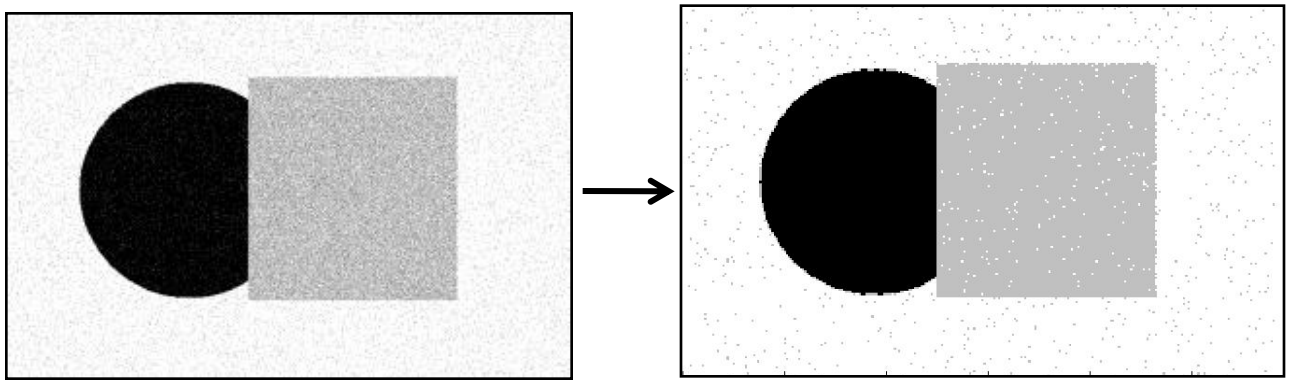
Filter bank of 24 filters

- Feature space: filter bank responses (e.g. 24D)

Slide credit: Kristen Grauman

Spatial coherence

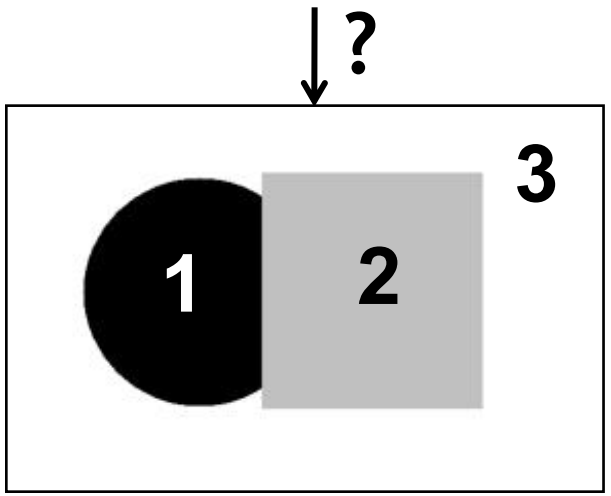
- Assign a cluster label per pixel → possible discontinuities



Original

Labeled by cluster center's intensity

- How can we ensure they are spatially smooth?

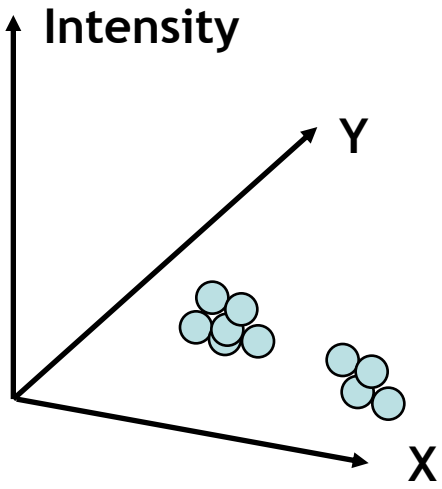


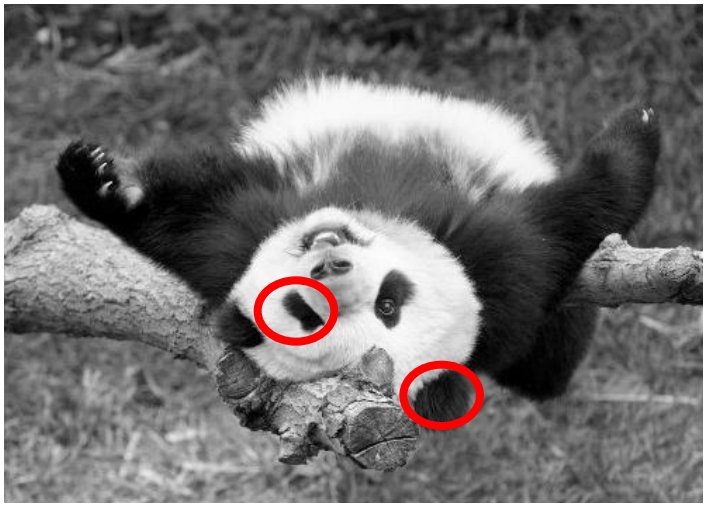
Slide adapted from Kristen Grauman

Spatial coherence

- Depending on what we choose as the *feature space*, we can group pixels in different ways.

- Grouping pixels based on *intensity+position* similarity





⇒ Way to encode both *similarity* and *proximity*.

Slide adapted from Kristen Grauman

K-Means without spatial information

- **K-means clustering based on intensity or color is essentially vector quantization of the image attributes**
  - **Clusters don't have to be spatially coherent**

Image

Intensity-based clusters

Color-based clusters

Slide adapted from Svetlana Lazebnik

Image source: Forsyth & Ponce

K-Means with spatial information

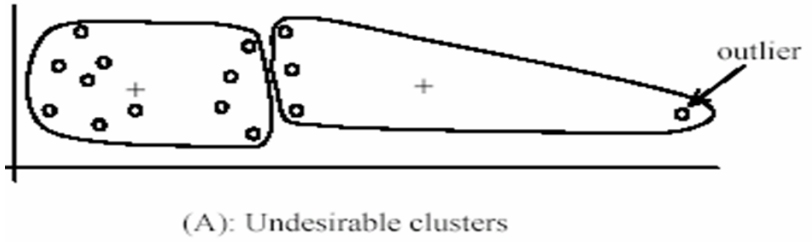
- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
  - Clusters don't have to be spatially coherent
- Clustering based on (r,g,b,x,y) values enforces more spatial coherence

Slide adapted from Svetlana Lazebnik

Image source: Forsyth & Ponce

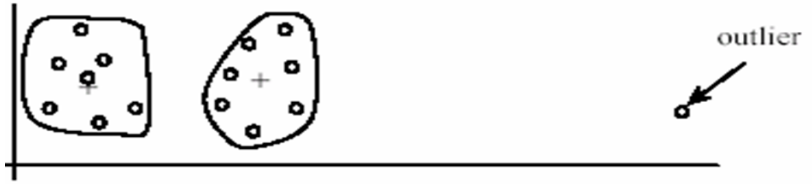
Summary K-Means

- **Pros**
  - Simple, fast to compute
  - Converges to local minimum of within-cluster squared error

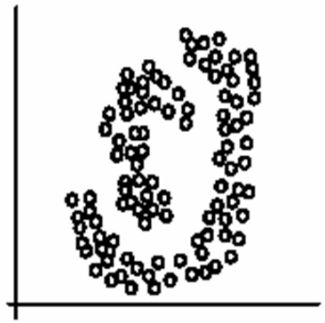


(A): Undesirable clusters

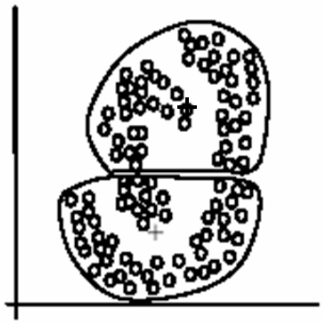
- **Cons/issues**
  - Setting k?
  - Sensitive to initial centers
  - Sensitive to outliers
  - Detects spherical clusters only
  - Assuming means can be computed



(B): Ideal clusters



(A): Two natural clusters



(B): *k*-means clusters

Slide credit: Kristen Grauman

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

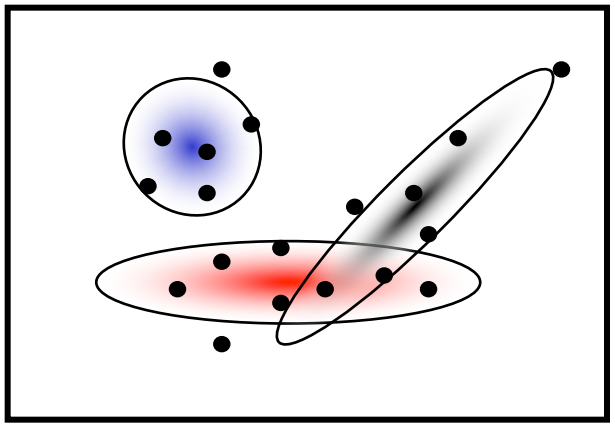
Probabilistic Clustering

- Basic questions
  - What's the probability that a point x is in cluster m?
  - What's the shape of each cluster?
- K-means doesn't answer these questions.

- Basic idea
  - Instead of treating the data as a bunch of points, assume that they are all generated by sampling a continuous function.
  - This function is called a generative model.
  - Defined by a vector of parameters θ

Slide credit: Steve Seitz

Mixture of Gaussians

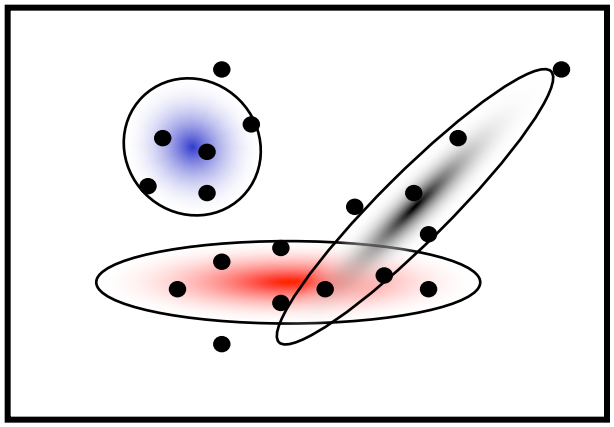


- One generative model is a mixture of Gaussians (MoG)
  - K Gaussian blobs with means μ_b covariance matrices V_b, dimension d
    - Blob *b* defined by: $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
  - Blob *b* is selected with probability α_b
  - The likelihood of observing x is a weighted mixture of Gaussians

$$P(x|\theta) = \sum_{b=1}^{K} \alpha_b P(x|\theta_b) \,, \quad \theta = [\mu_1, \ldots, \mu_K, V_1, \ldots, V_K]$$

Slide adapted from Steve Seitz

Training with Expectation Maximization (EM)



- **Goal**
  - **Find blob parameters θ that maximize the likelihood function overall all *N* datapoints $X = \{x_1, \ldots, x_N\}$**

$$P(X) = \prod_{x_i \in X} P(x_i|\theta)$$

- **Approach:**
  1. **E-step: given current guess of blobs, compute probabilistic ownership of each point**
  2. **M-step: given ownership probabilities, update blobs to maximize likelihood function**
  3. **Repeat until convergence**

Slide adapted from Steve Seitz

EM Details

- **E-step**
  - **Compute probability that point x is in blob *b*, given current guess of θ**

$$P(b|x,\mu_b,V_b) = \frac{\alpha_b P(x|\mu_b,V_b)}{\sum_{i=1}^{K}\alpha_i P(x|\mu_i,V_i)}$$

- **M-step**
  - **Compute overall probability that blob *b* is selected**

$$\alpha_b^{new} = \frac{1}{N}\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b) \qquad (N \text{ data points})$$

  - **Mean of blob *b***

$$\mu_b^{new} = \frac{\sum_{i=1}^{N} x_i P(b|x_i,\mu_b,V_b)}{\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)}$$

  - **Covariance of blob *b***

$$V_b^{new} = \frac{\sum_{i=1}^{N}(x_i-\mu_b^{new})(x_i-\mu_b^{new})^T P(b|x_i,\mu_b,V_b)}{\sum_{i=1}^{N} P(b|x_i,\mu_b,V_b)}$$

Slide adapted from Steve Seitz

Applications of EM

- Useful for all sorts of problems
  - Any clustering problem
  - Many model estimation problems
  - Missing data problems
  - Finding outliers
  - Segmentation problems
    - Segmentation based on color
    - Segmentation based on motion
    - Foreground/background separation
  - …

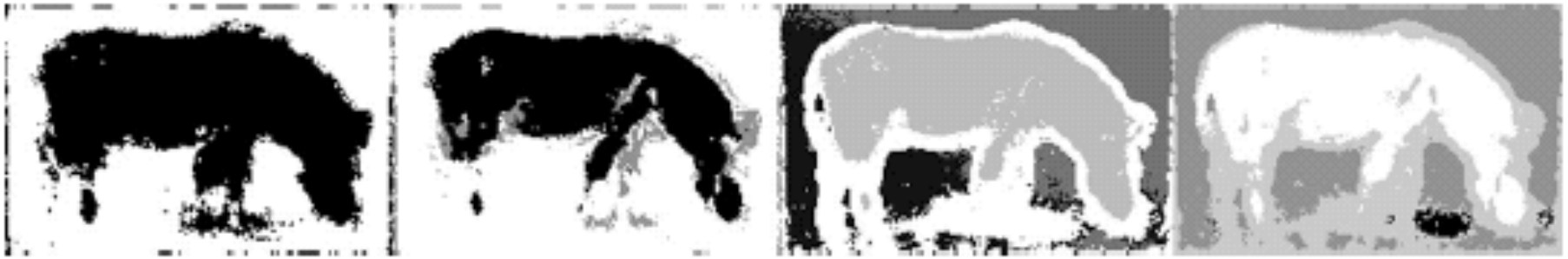
- EM demo
  - http://lcn.epfl.ch/tutorial/english/gaussian/html/index.html

Slide credit: Steve Seitz

Segmentation with EM

Original image

EM segmentation results



k=2 k=3 k=4 k=5

Slide credit: B. Leibe

Image source: Serge Belongie

Summary: Mixtures of Gaussians, EM

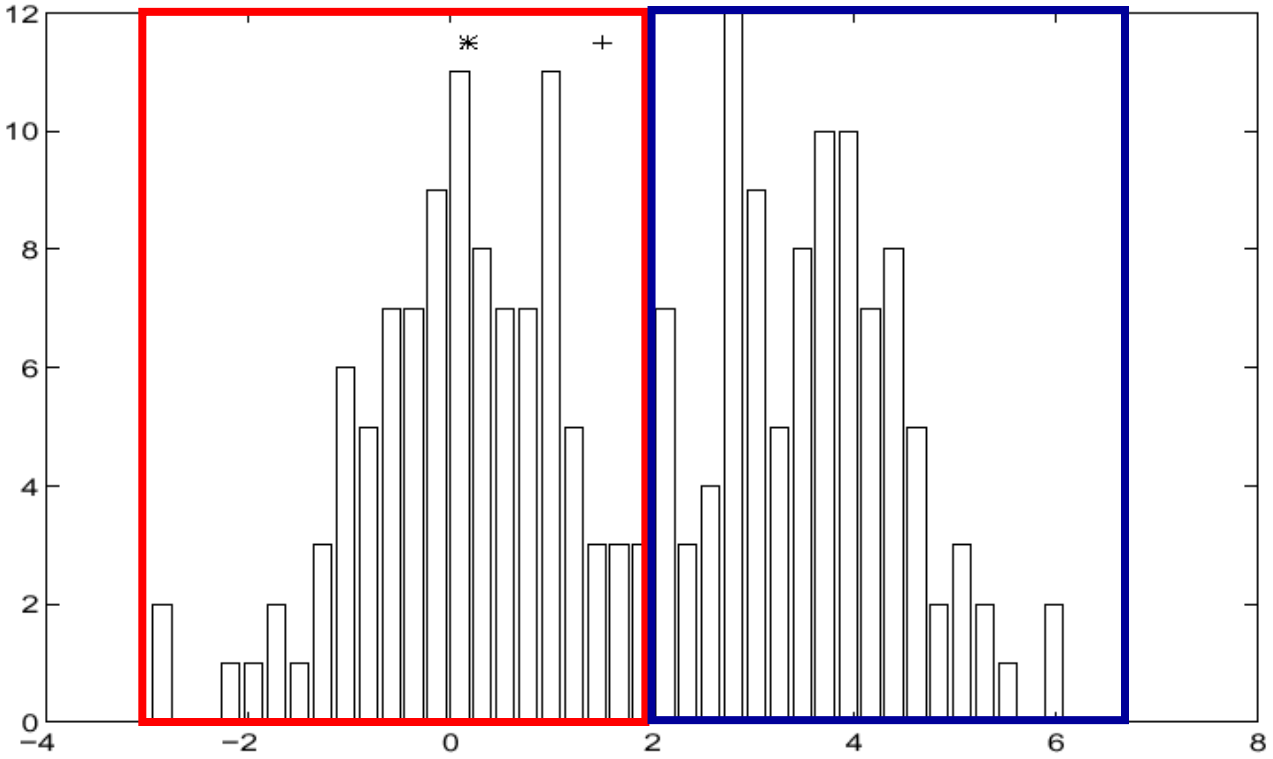
- **Pros**
  - Probabilistic interpretation
  - Soft assignments between data points and clusters
  - Generative model, can predict novel data points
  - Relatively compact storage ($O(K\,d^2)$)
- **Cons**
  - Initialization
    - often a good idea to start from output of k-means
  - Local minima
  - Need to know number of components *K*
    - solutions: model selection (AIC, BIC), Dirichlet process mixture
  - Need to choose blob generative model (math form of a cluster?)
  - Numerical problems are often a nuisance

Slide adapted from B. Leibe

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

Finding Modes in a Histogram

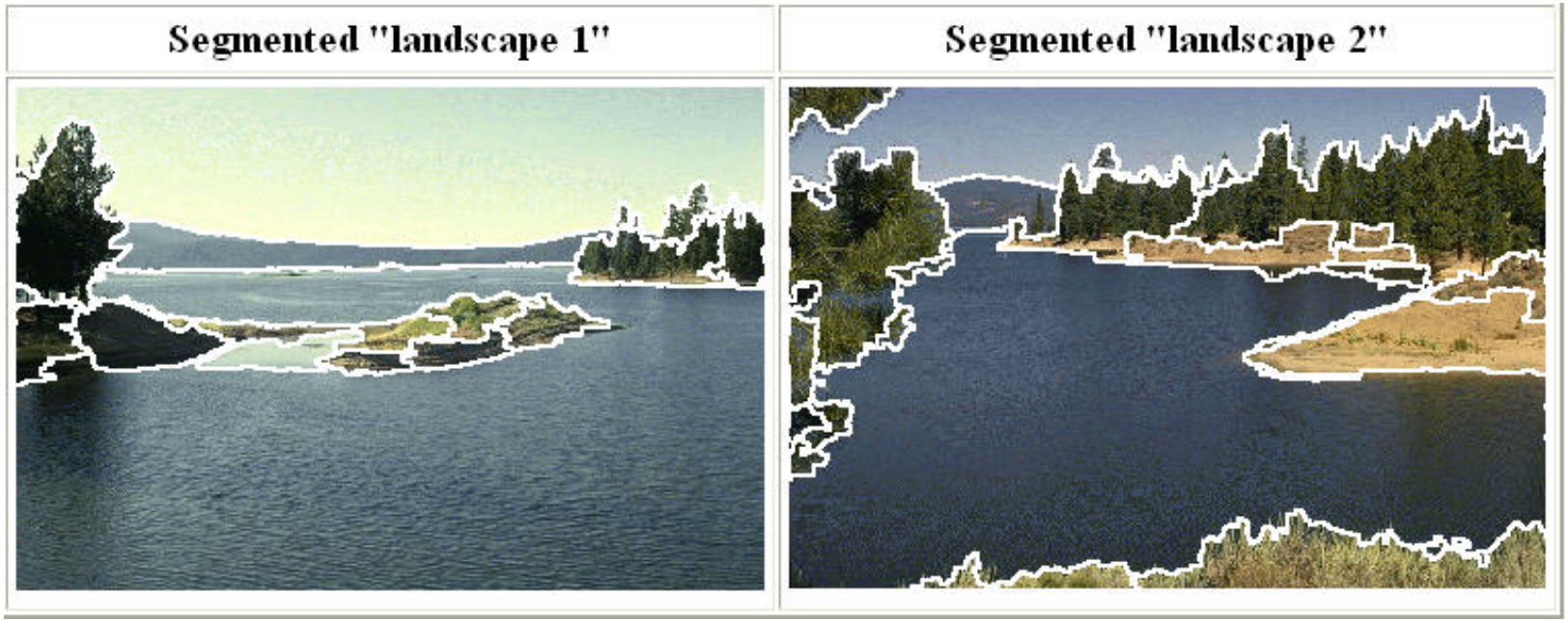


- How many modes are there?
  - *Mode* = local maximum of a given distribution
  - Easy to see, hard to compute

Slide adapted from Steve Seitz

Mean-Shift Segmentation

A versatile technique for clustering-based segmentation

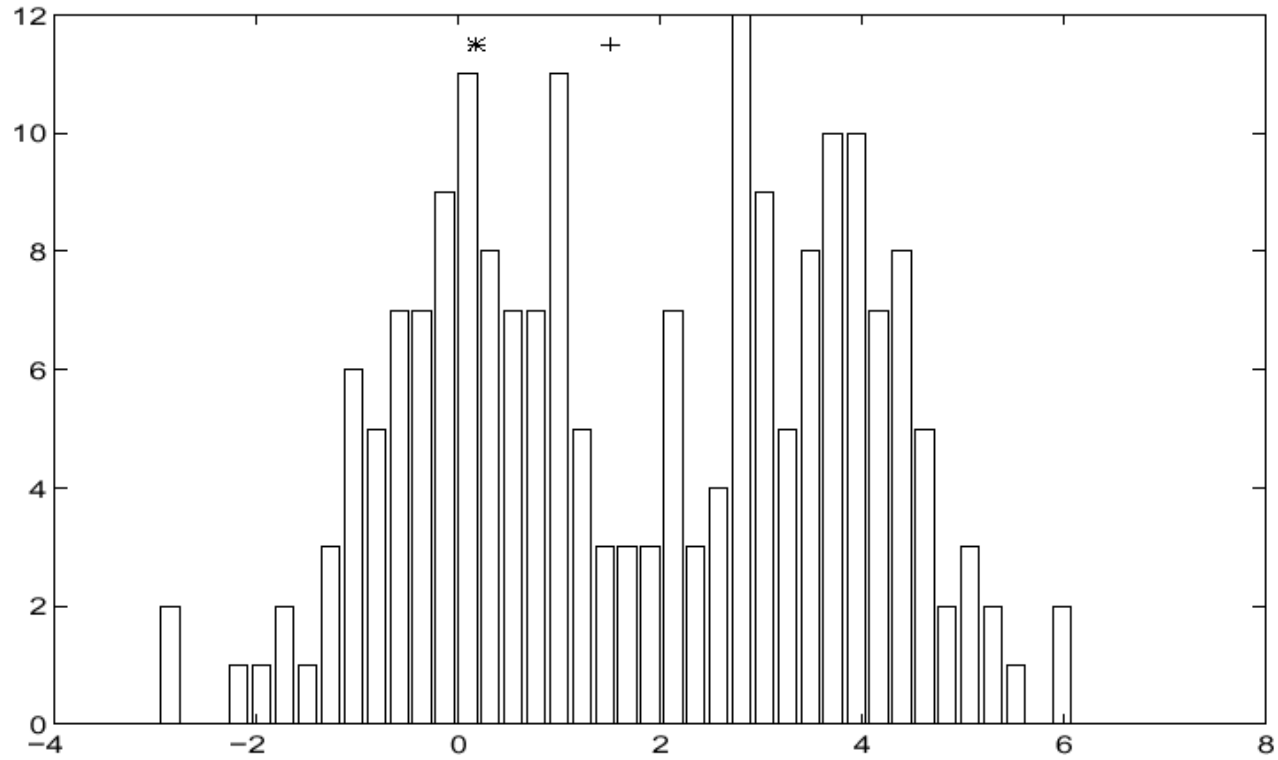


http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Slide credit: Svetlana Lazebnik

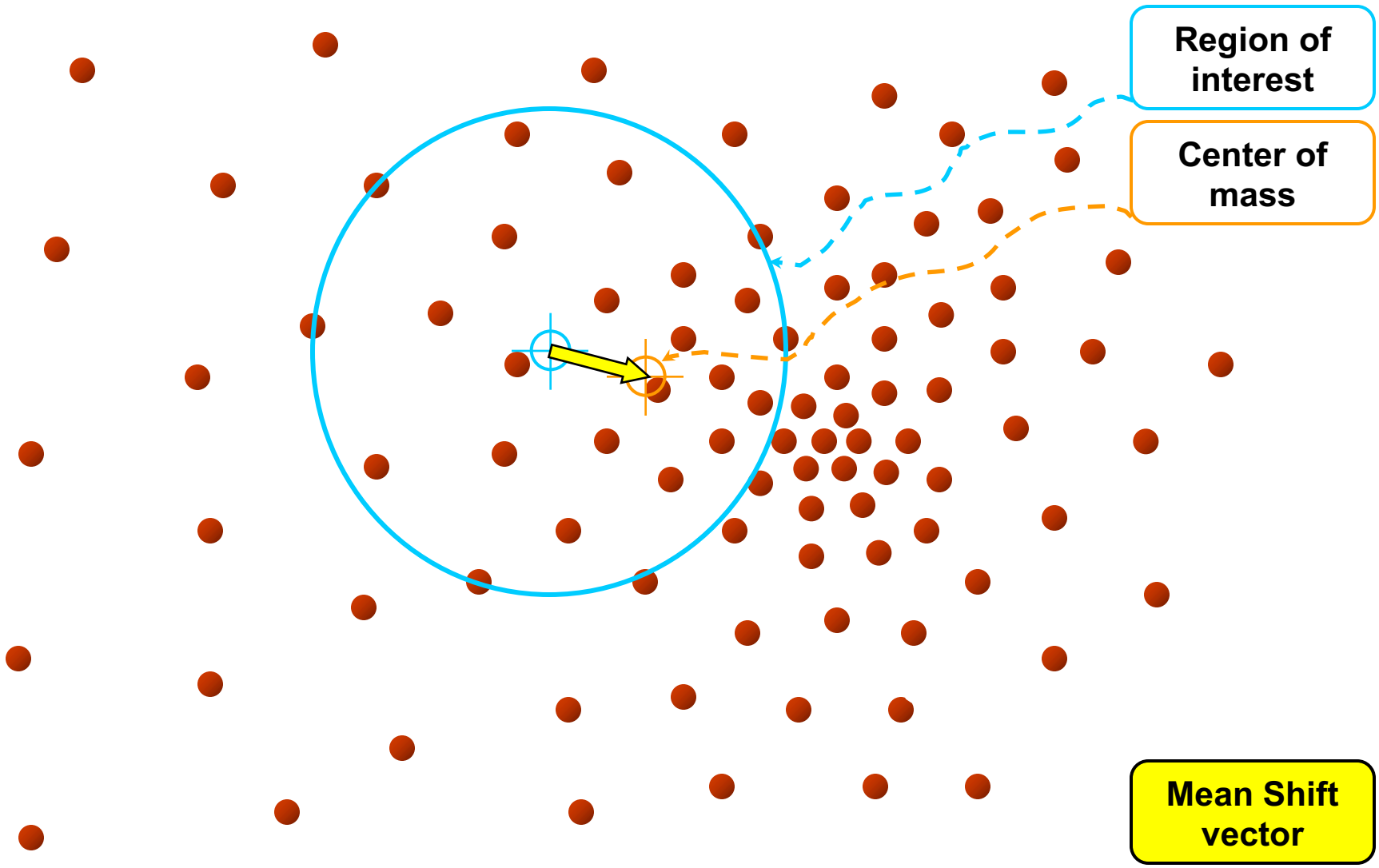
Mean-Shift Algorithm



- **Iterative Mode Search**
  1. **Initialize random seed center and window W**
  2. **Calculate center of gravity (the "mean") of W:** $\sum_{x \in W} xH(x)$
  3. **Shift the search window to the mean**
  4. **Repeat steps 2+3 until convergence**

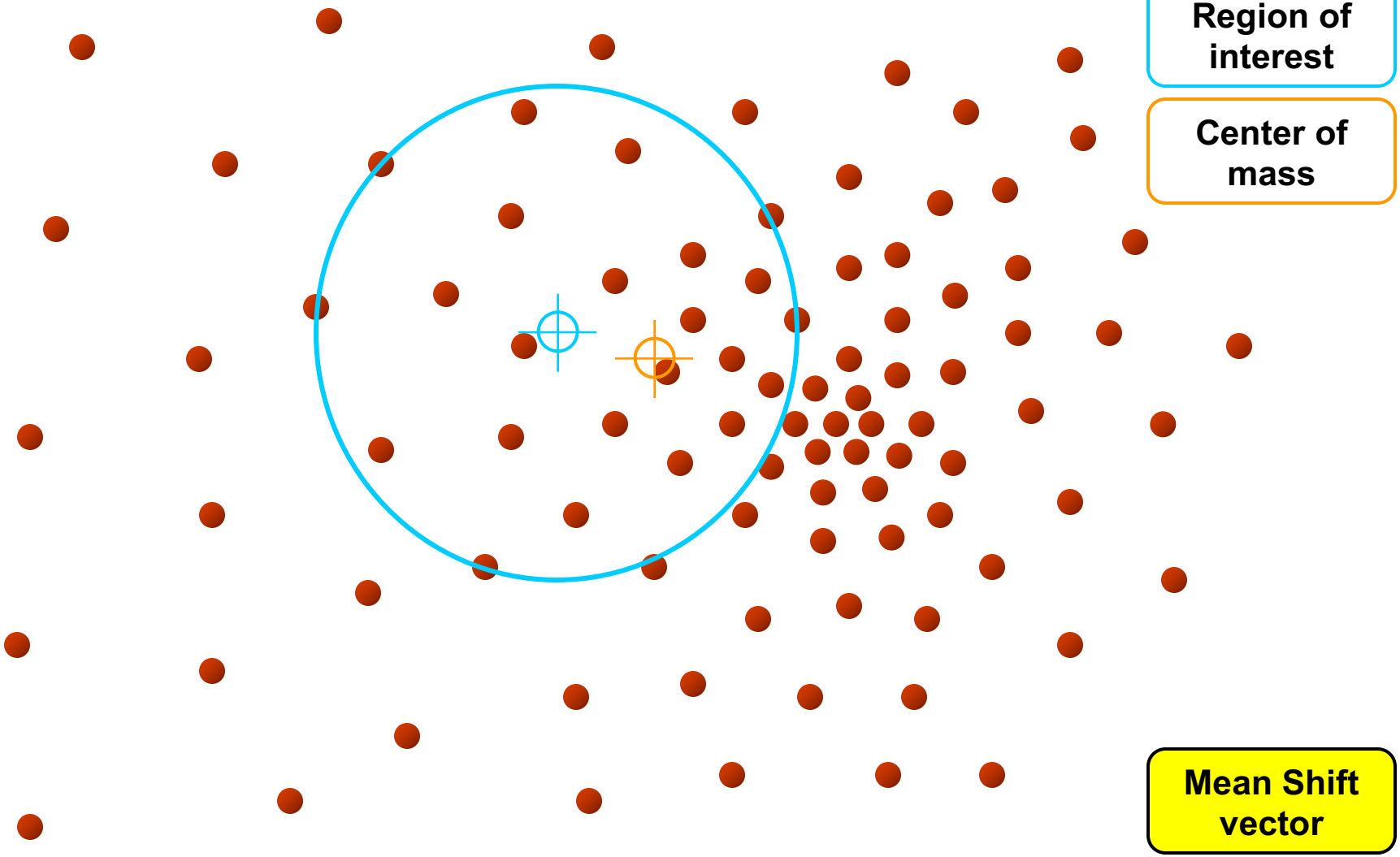
Slide adapted from Steve Seitz

Mean-Shift



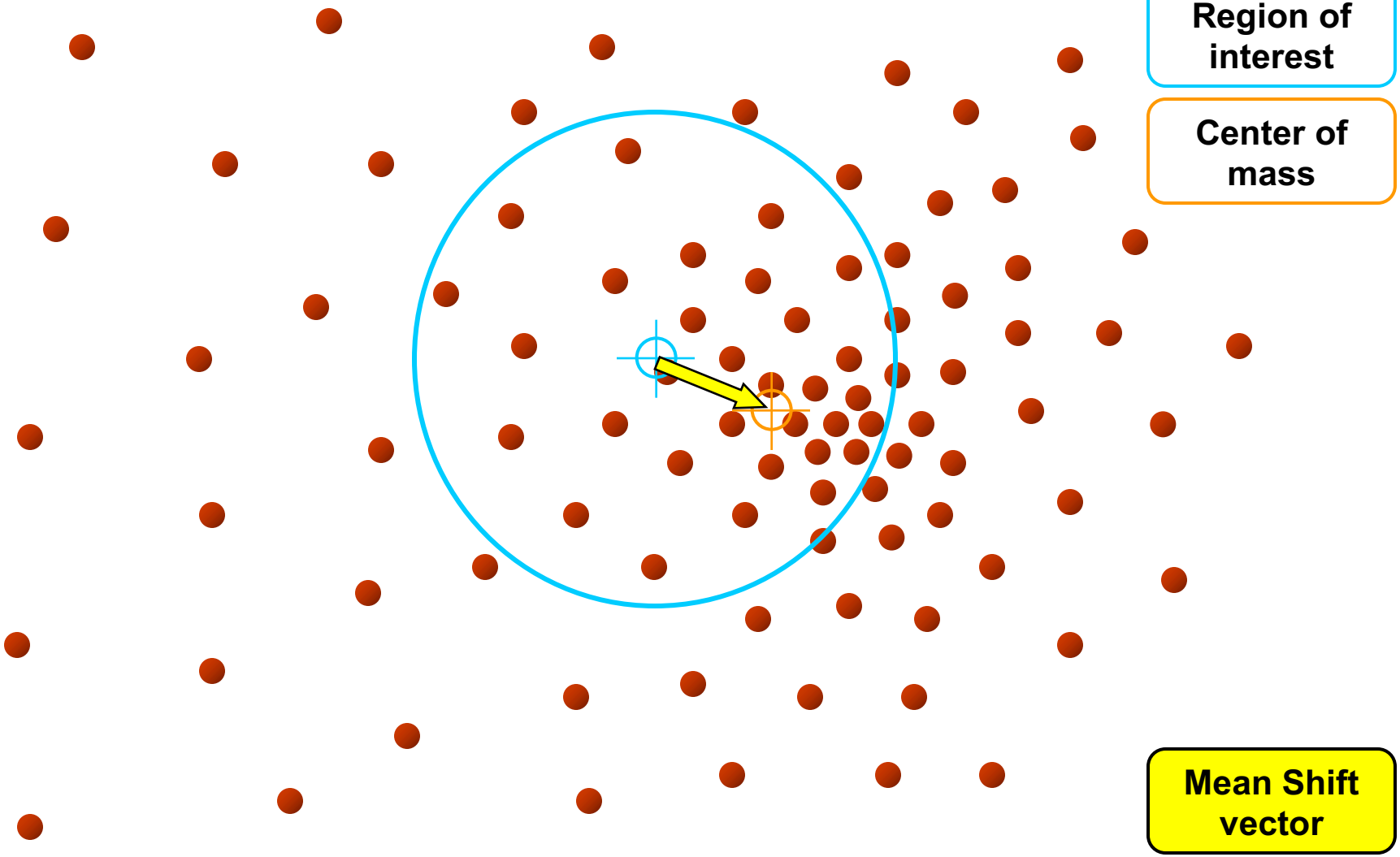
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



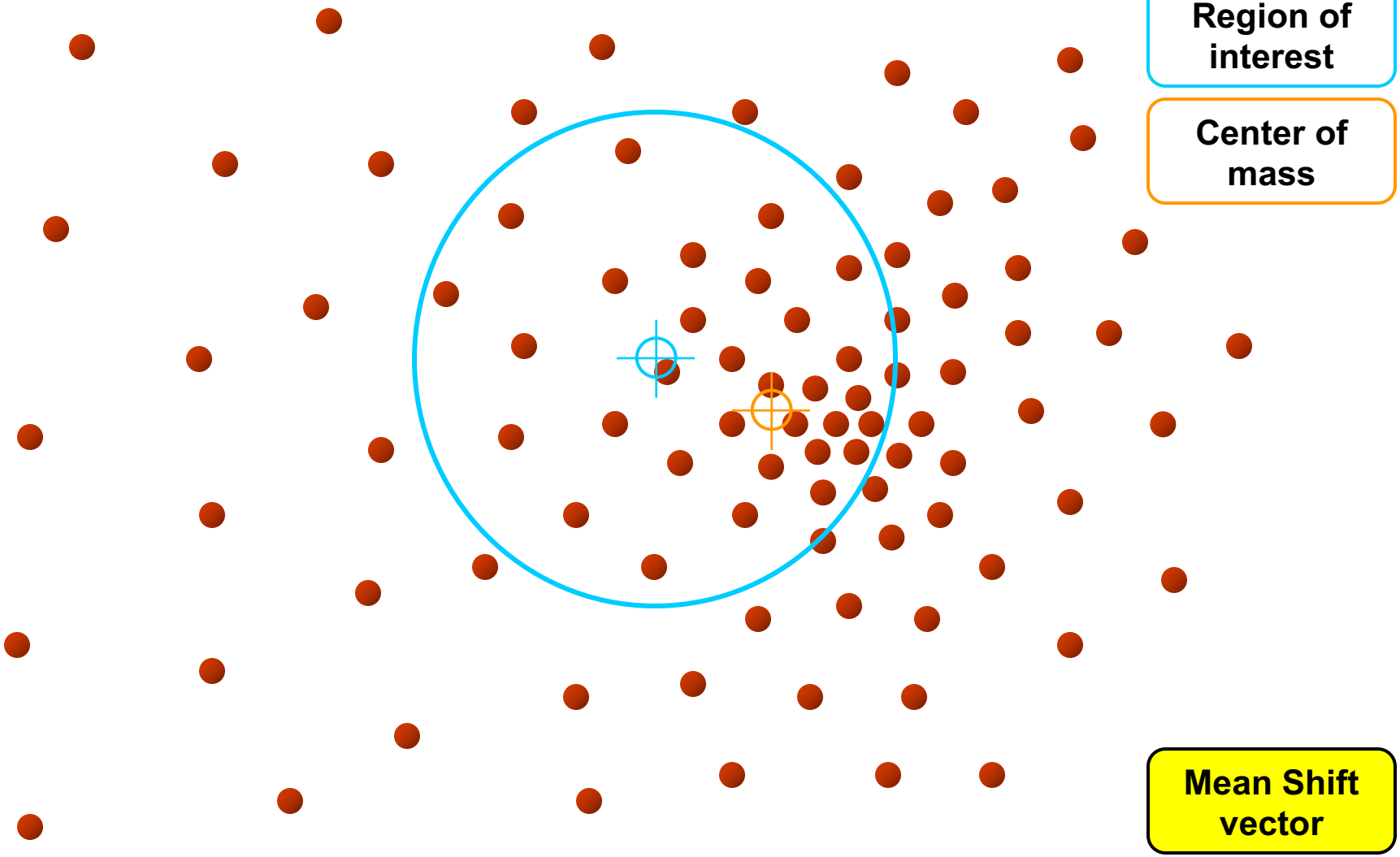
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



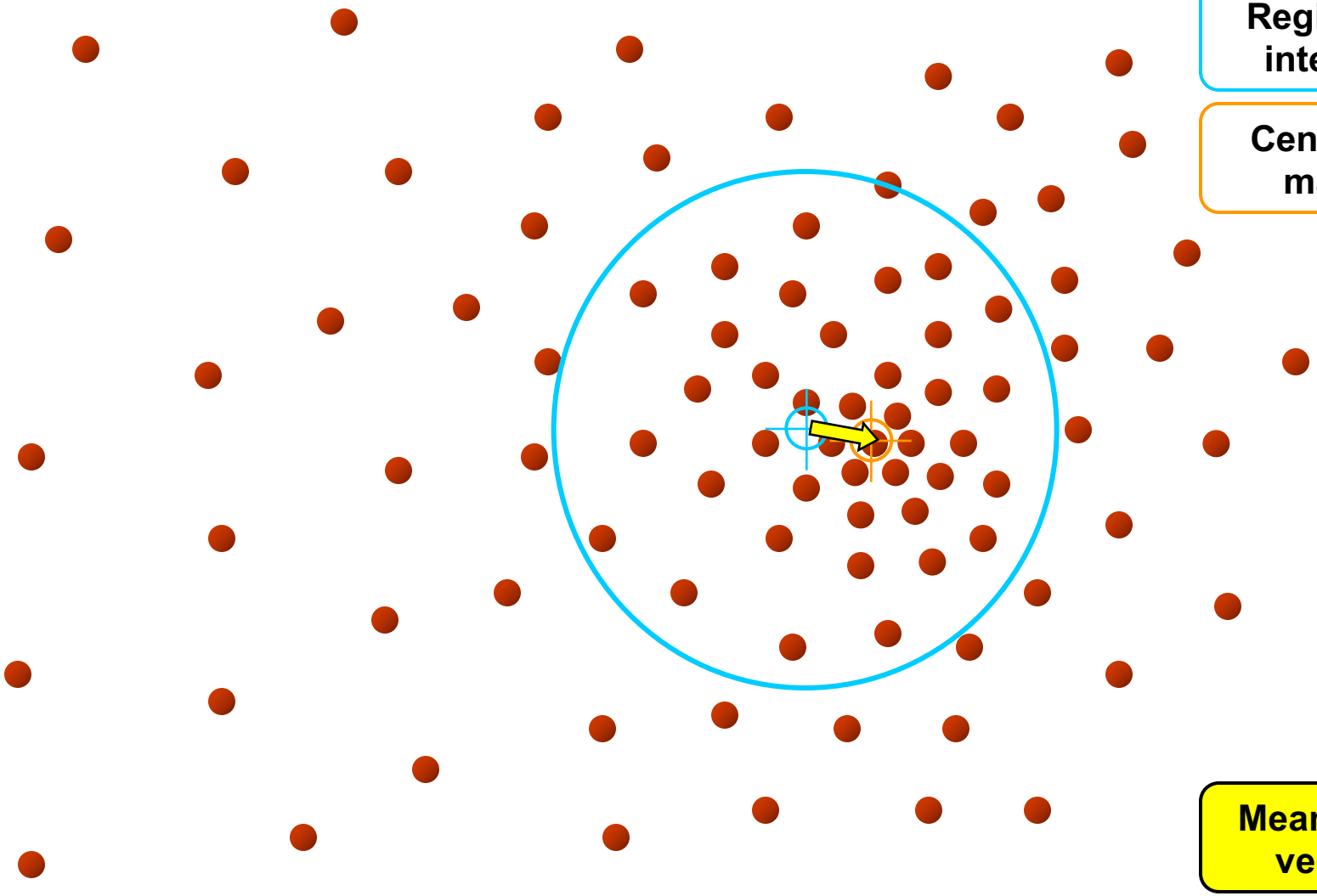
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



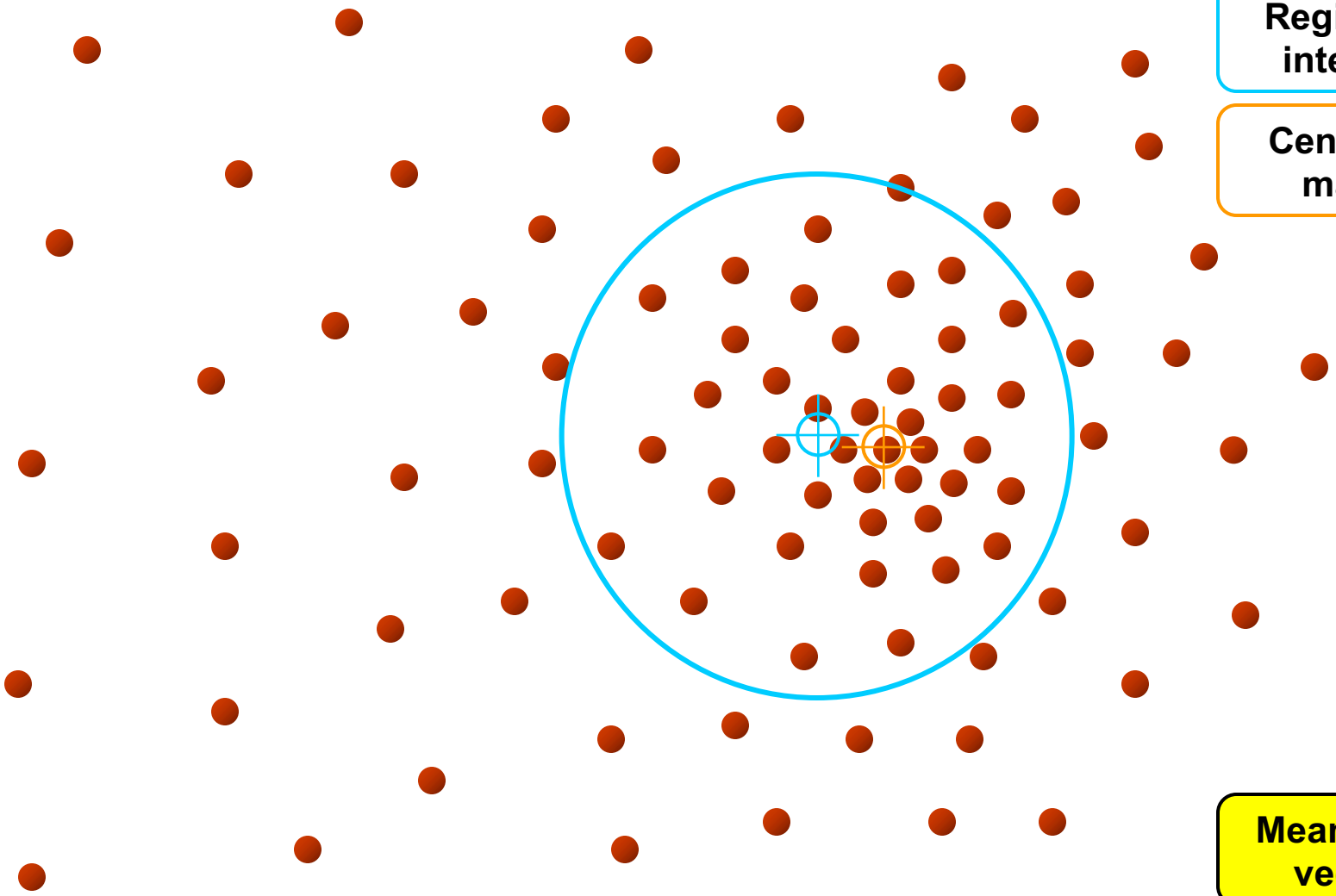
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



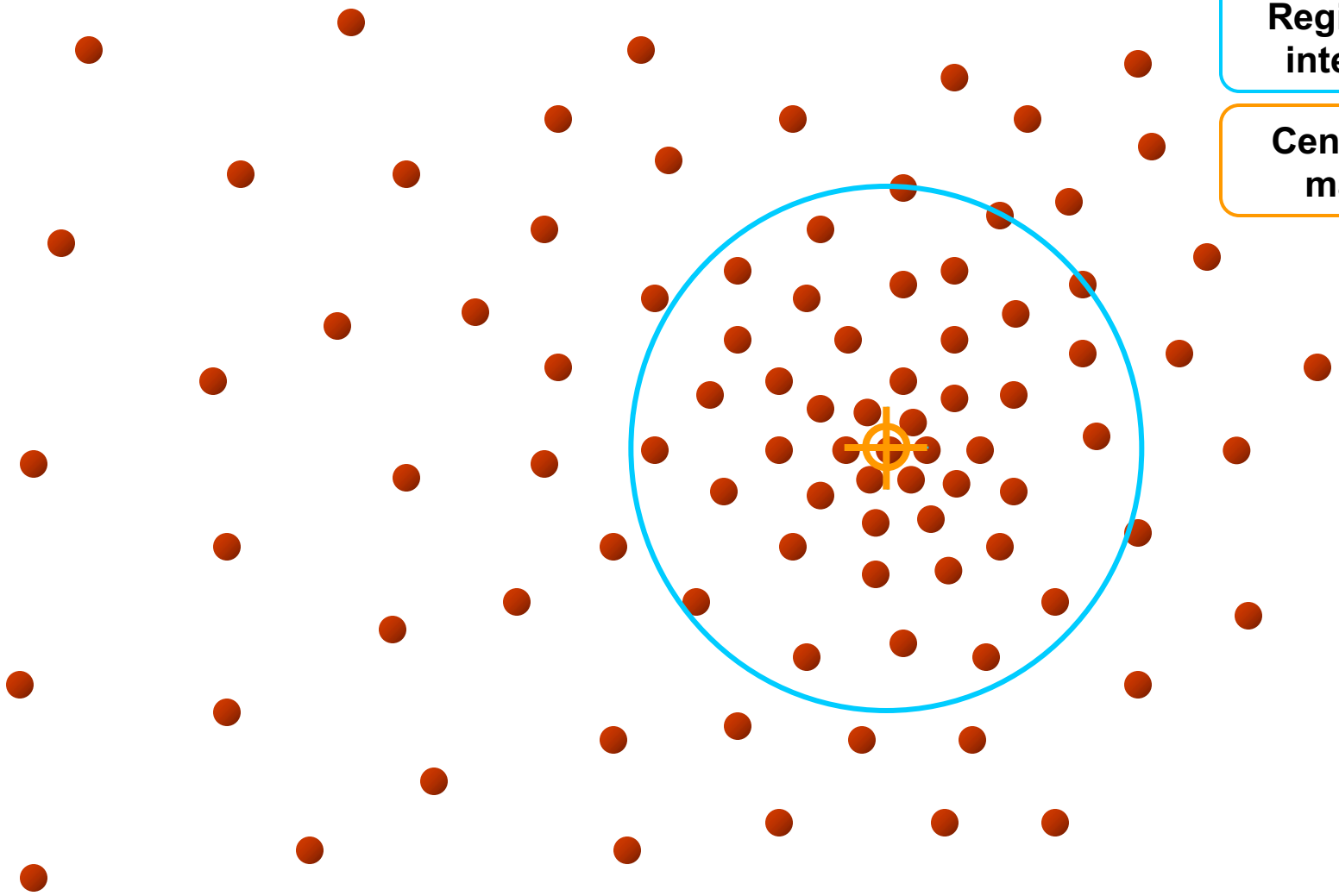
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



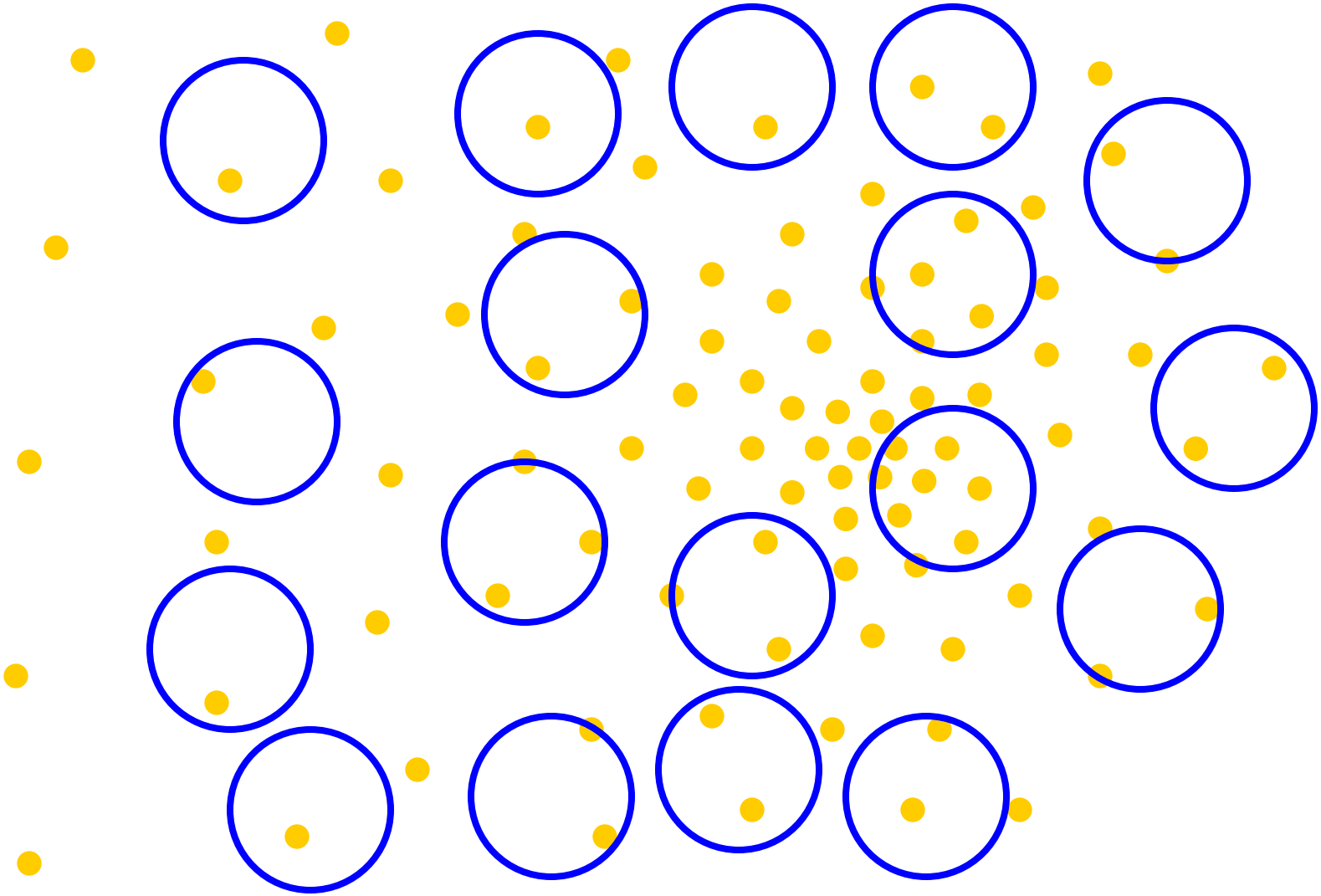
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



Slide by Y. Ukrainitz & B. Sarel

Real Modality Analysis

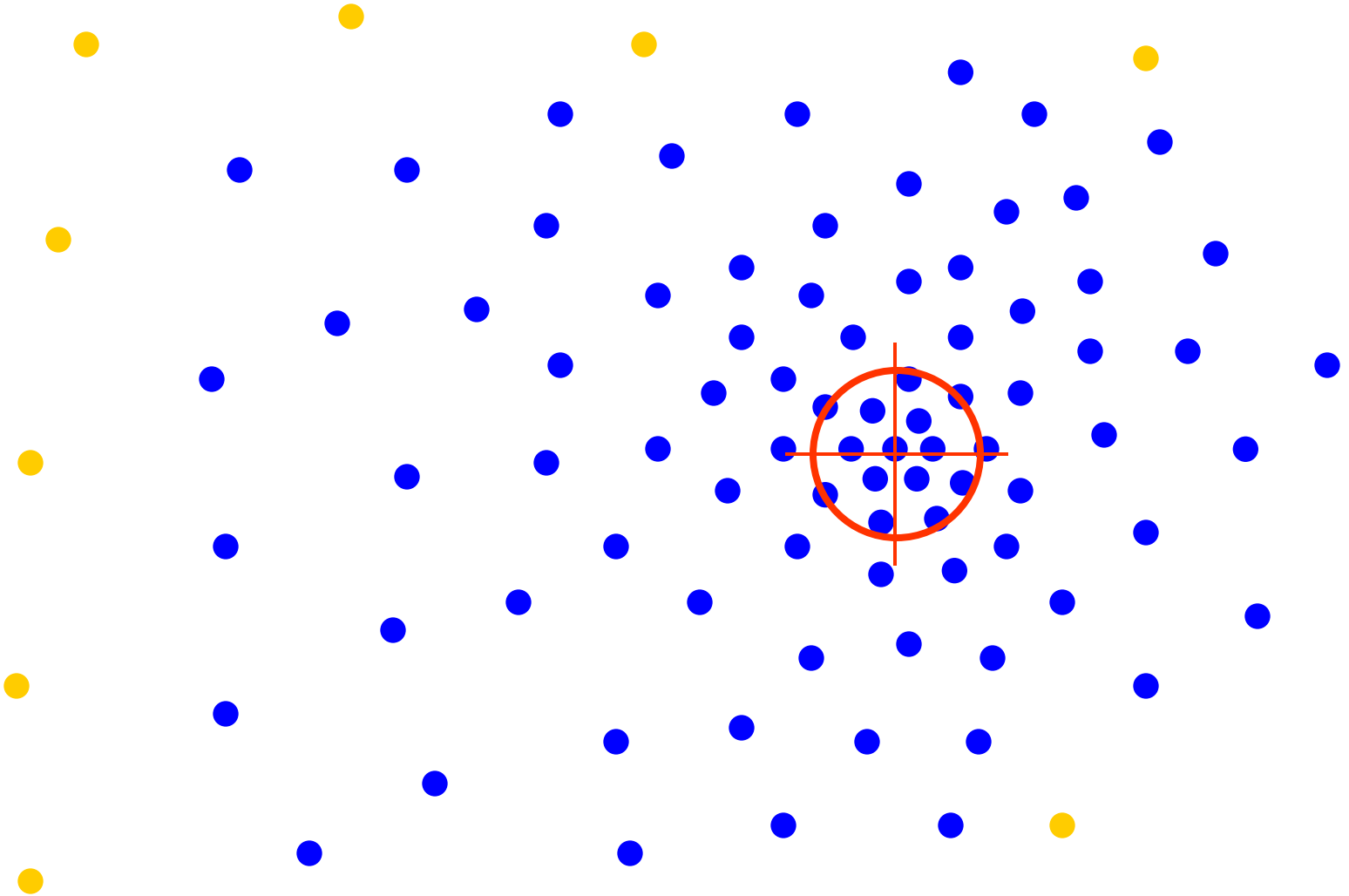


Tessellate the space with windows (or start from every point)

Run the procedure in parallel

Slide by Y. Ukrainitz & B. Sarel

Real Modality Analysis

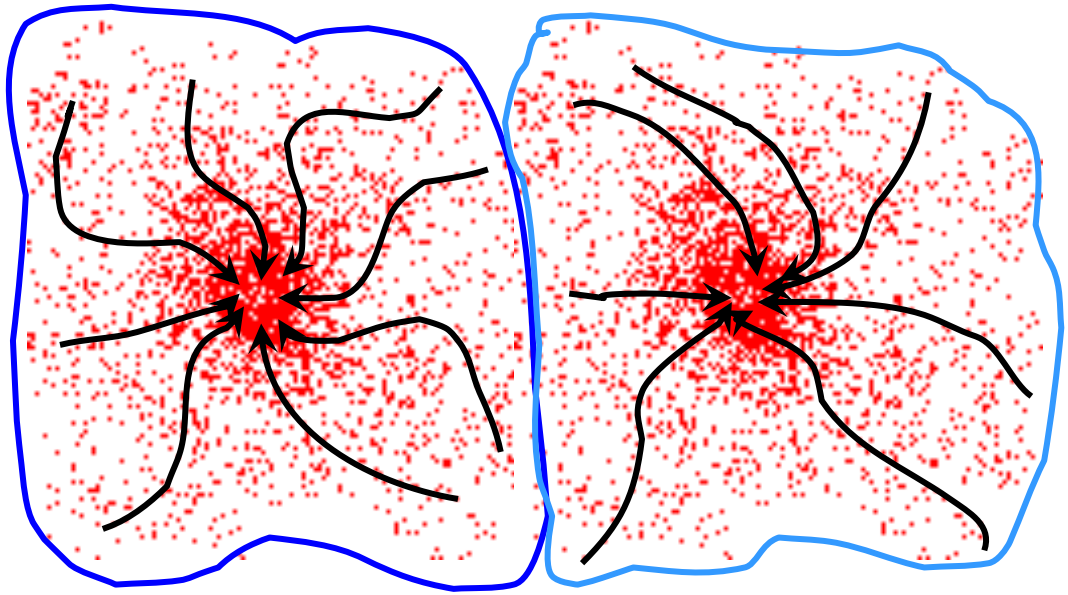


The blue data points were traversed by the windows towards the mode.

Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering

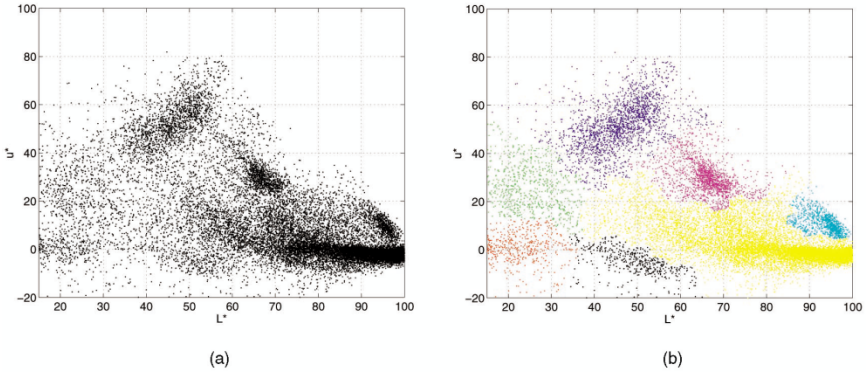
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

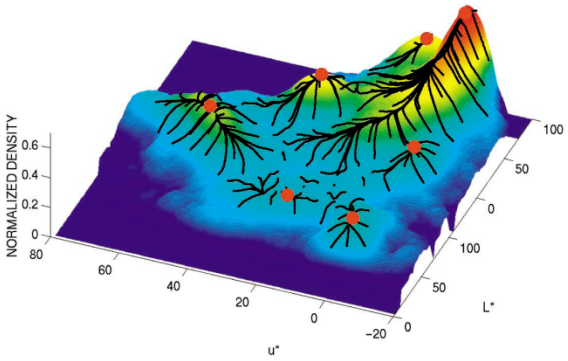


Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Segmentation: overall algorithm

- Choose features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Start mean-shift from each window until convergence
- Merge windows that end up near the same “peak” or mode

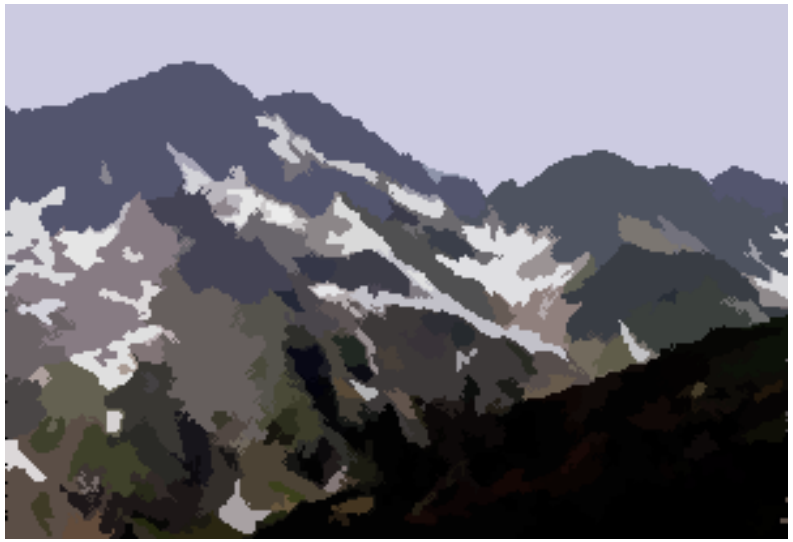




Slide adapted from Svetlana Lazebnik

Mean-Shift Segmentation Results



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Slide credit: Svetlana Lazebnik

More Results

Slide credit: Svetlana Lazebnik

Summary Mean-Shift

- **Pros**
  - General, application-independent tool
  - Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
  - Just a single parameter (window size h)
    - h has a physical meaning (unlike k-means) == scale of clustering
  - Finds variable number of modes given the same h
  - Robust to outliers

- **Cons**
  - Output depends on window size h
  - Window size (bandwidth) selection is not trivial
  - Computationally rather expensive
  - Does not scale well with dimension of feature space

Slide adapted from Svetlana Lazebnik

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

The Hough transform : principle

Uses the structure of shapes to extract them in a parameter space of limited dimension

For general planar shapes the specification of their position in a plane requires 3 parameters

Not so for these simple shapes :

1. Straight lines
2. Circles of fixed radius

2 parameters suffice for the positioning of each

Hough transform : straight lines

Suppose we would like to detect straight lines

These are important given their omnipresence in man-made scenes

The analysis is as if the lines are of infinite length, but we will be able to extract line segments with the Hough approach…

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 :

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

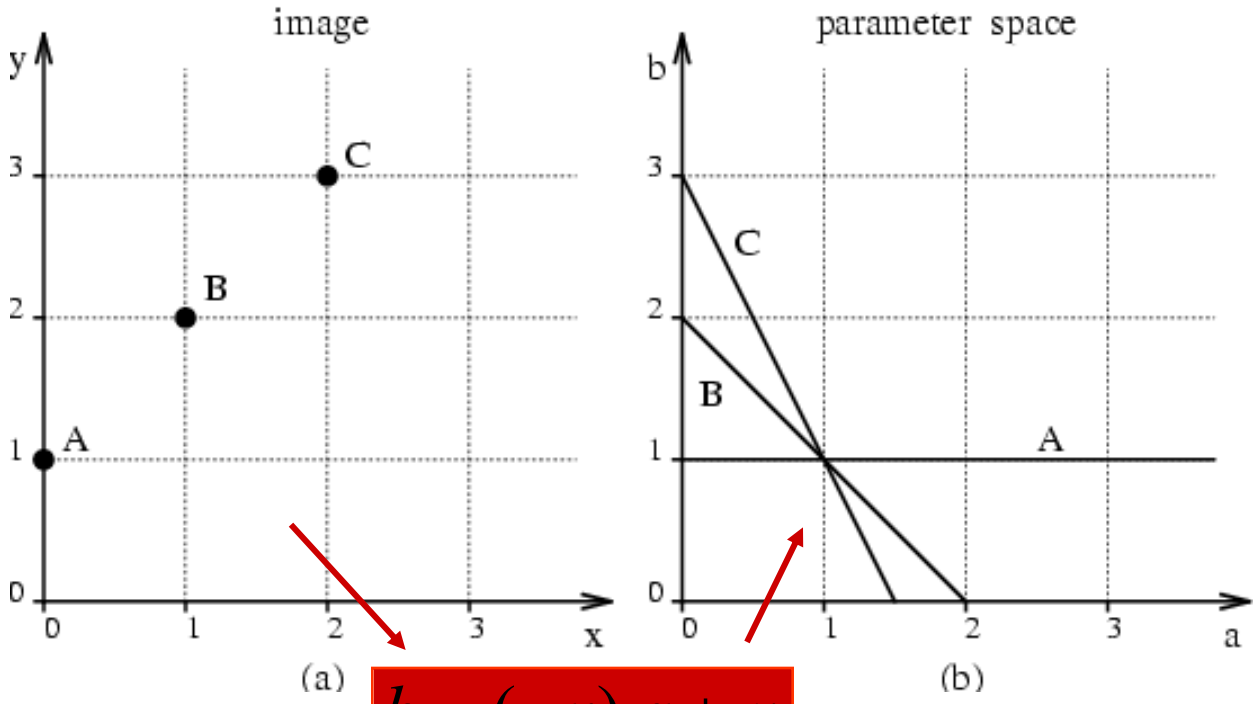
The Hough transform :

- 1. Scrutinize all edge pts
- 2. For each point draw the above line in

  (a,b) - parameter space

  (corresponding to all lines passing by x,y)

Hough transform : straight lines



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

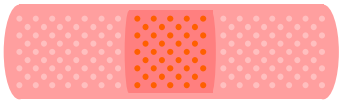
implementation :

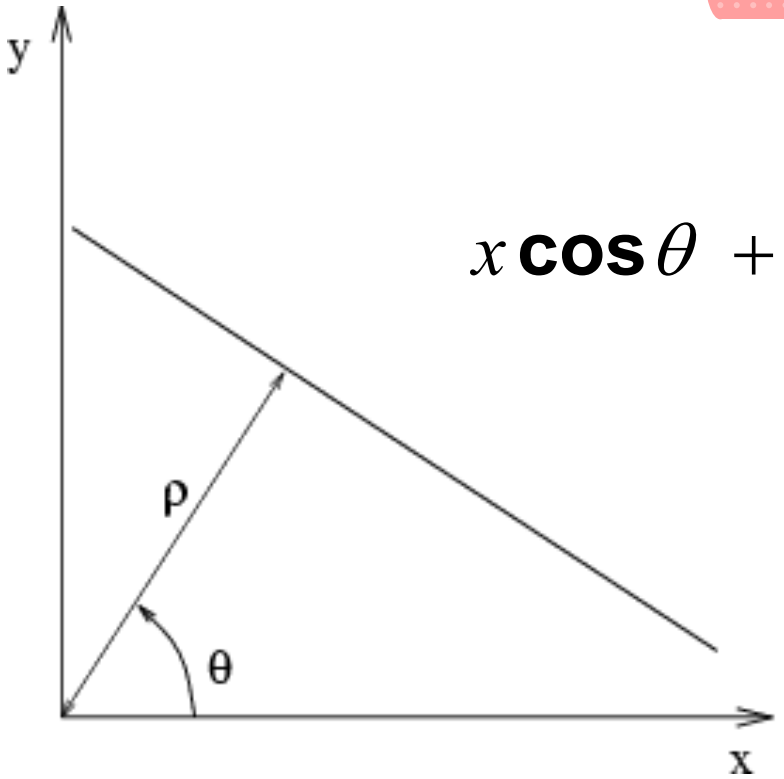
1. the parameter space is discretised
2. a counter is incremented at each cell where the lines pass
3. peaks are detected

Hough transform : straight lines

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

alternative representation:



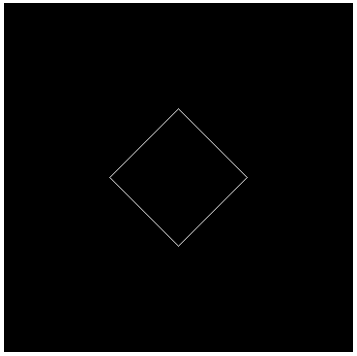


$$x\cos\theta + y\sin\theta = \rho$$

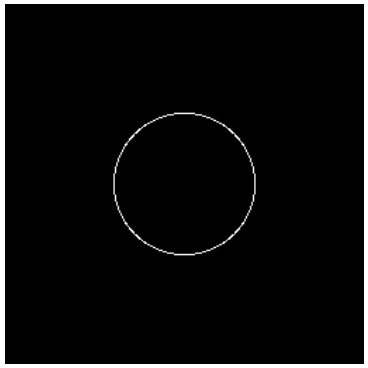
Each point will add a cosine function in the (θ,ρ) parameter space

Hough transform : straight lines

Square:

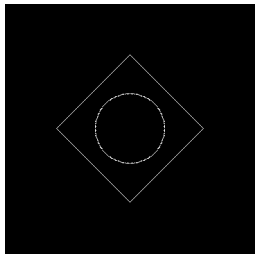


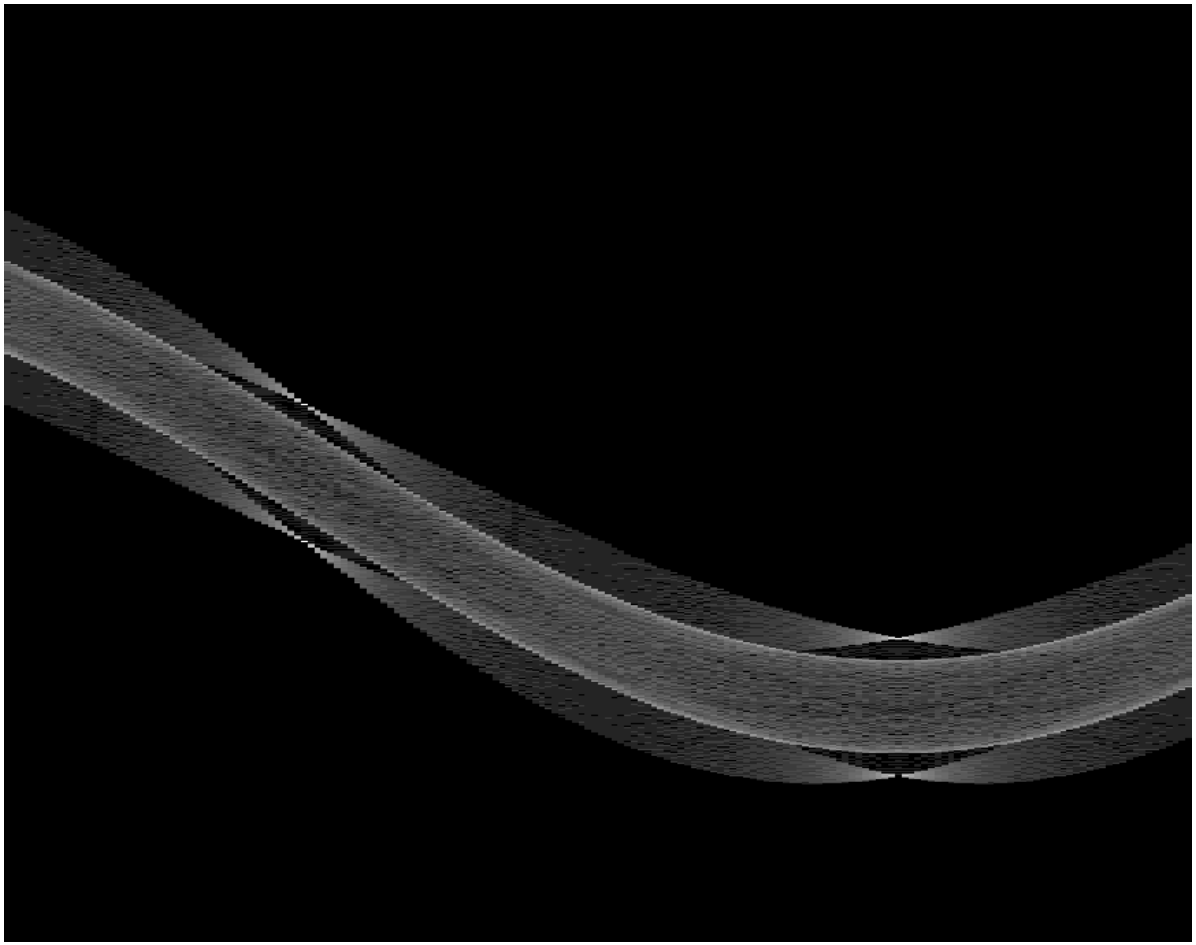
Circle:



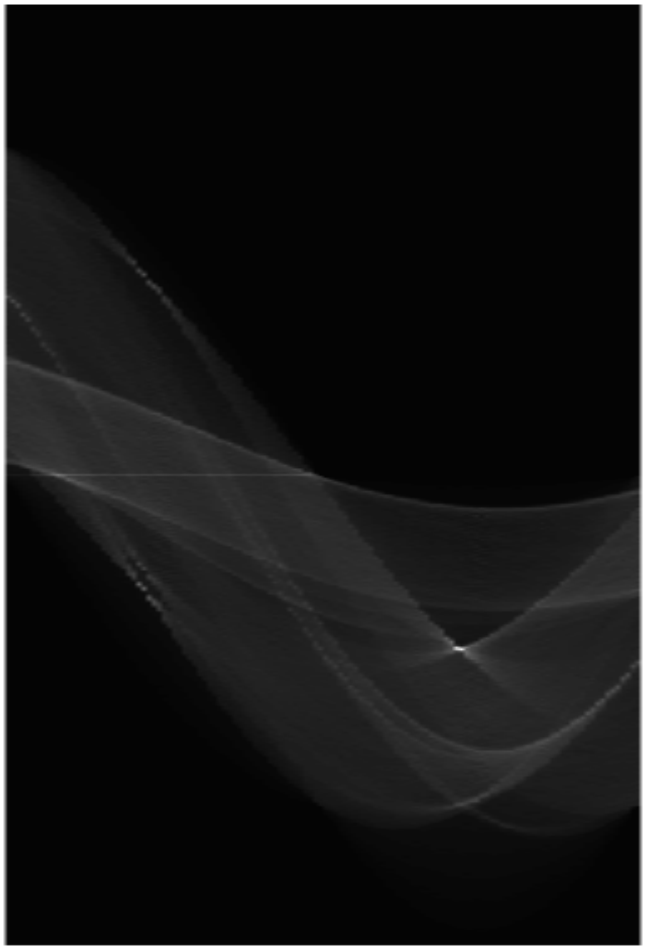
Hough transform : straight lines

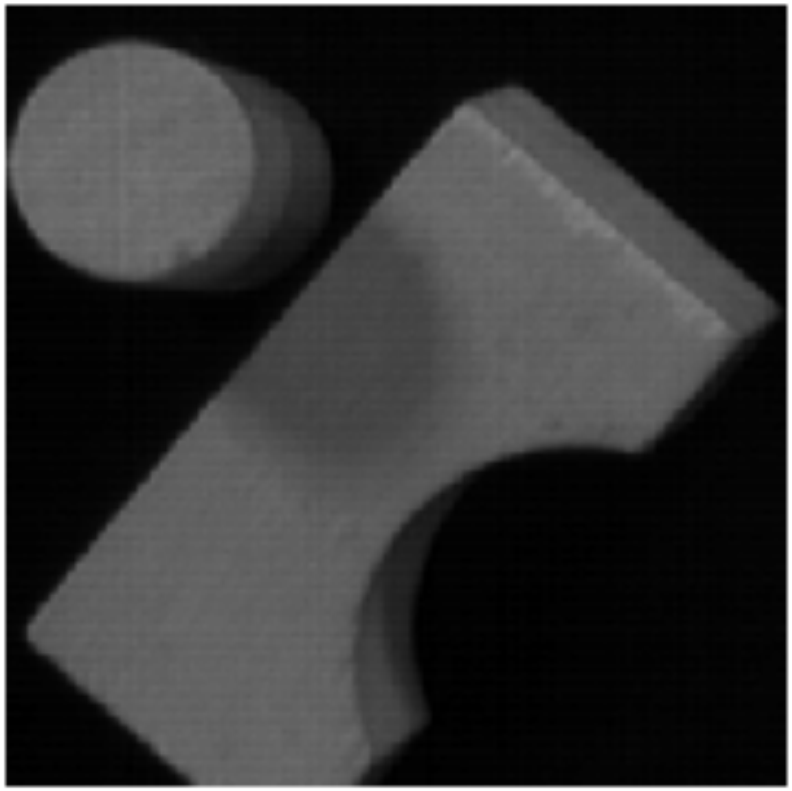
combined:





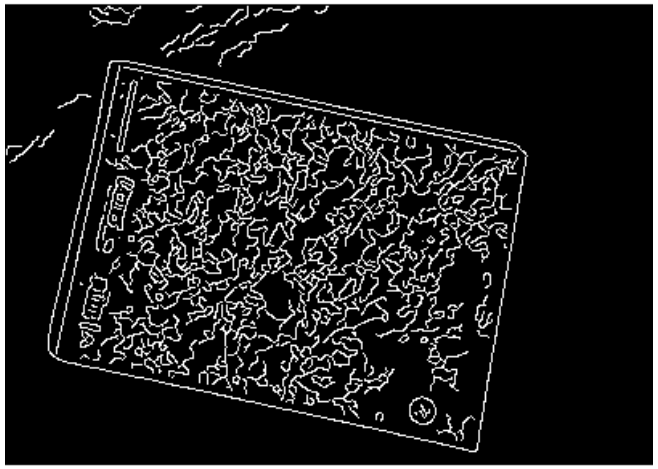
Hough transform : straight lines

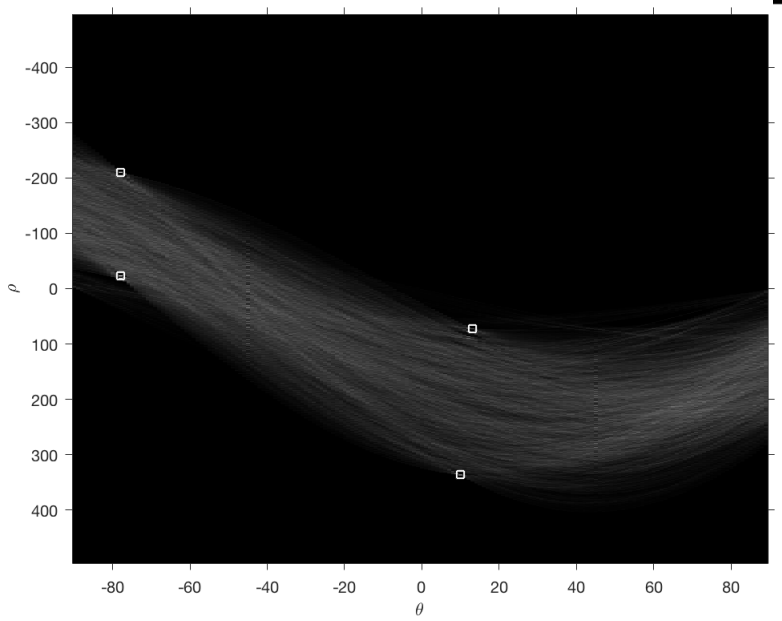


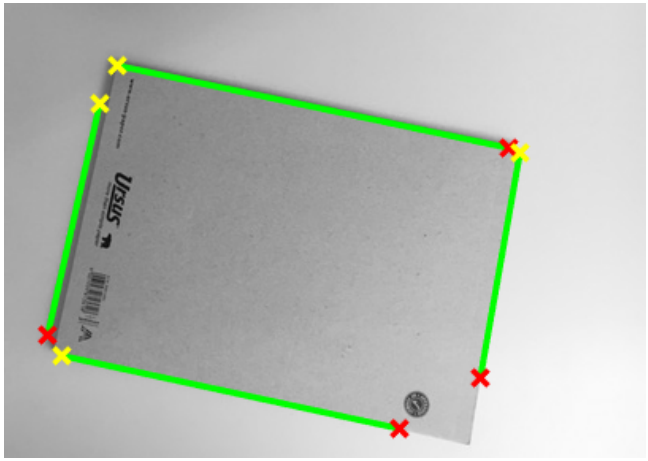


Now on a real image

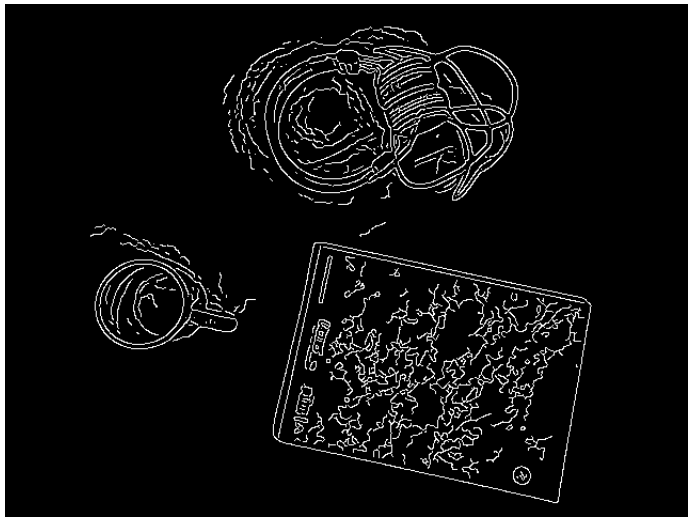
Hough transform : detecting the book

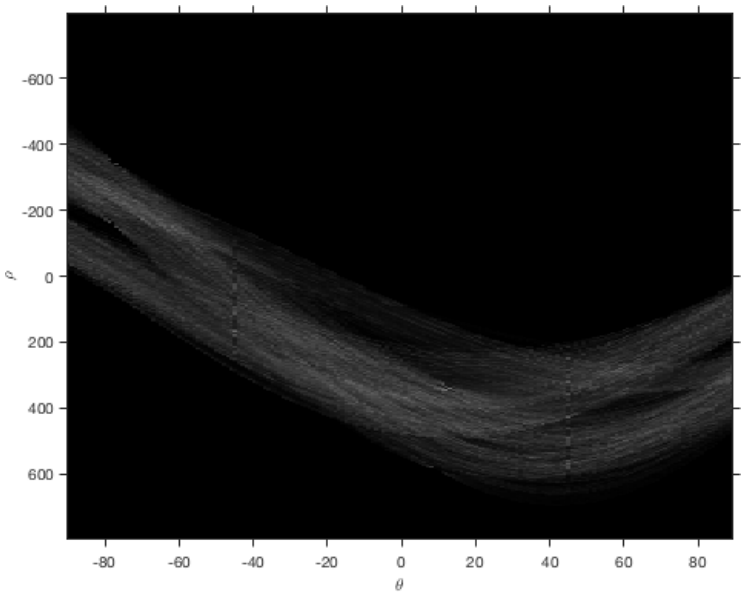


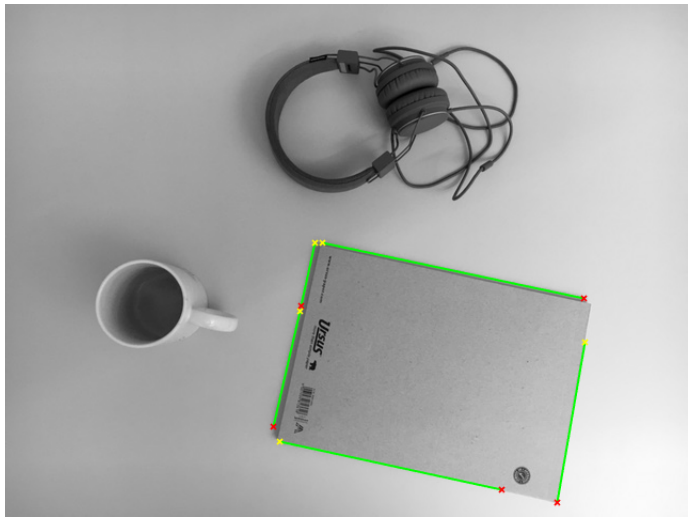




works with other objects present







Hough transform : remarks

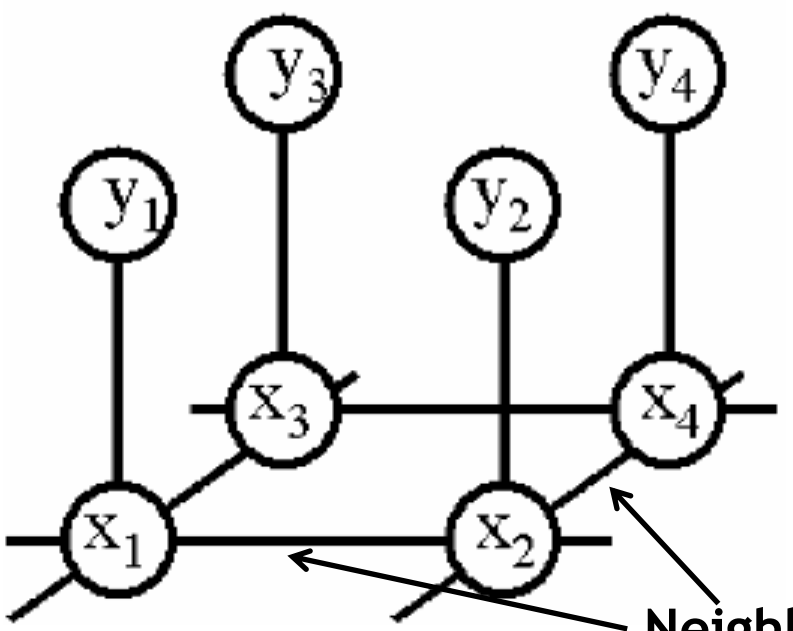
- 1. time consuming
- 2. robust to outliers and lots of points not on a line
- 3. peak detection is difficult with noise in edge coordinates
- 4. Robustness of peak detection increased by weighting contributions (votes that are proportional with intensity gradient magnitude)
- 5. can do it also for circles and even higher-order parametric shapes (but it quickly gets harder as number of shape parameters grows)

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

Markov Random Fields

- Allow rich probabilistic models for images
- But built in a local, modular way
  - Learn local effects, get global effects out



Slide credit: William Freeman

MRF Nodes as Pixels (or Patches)

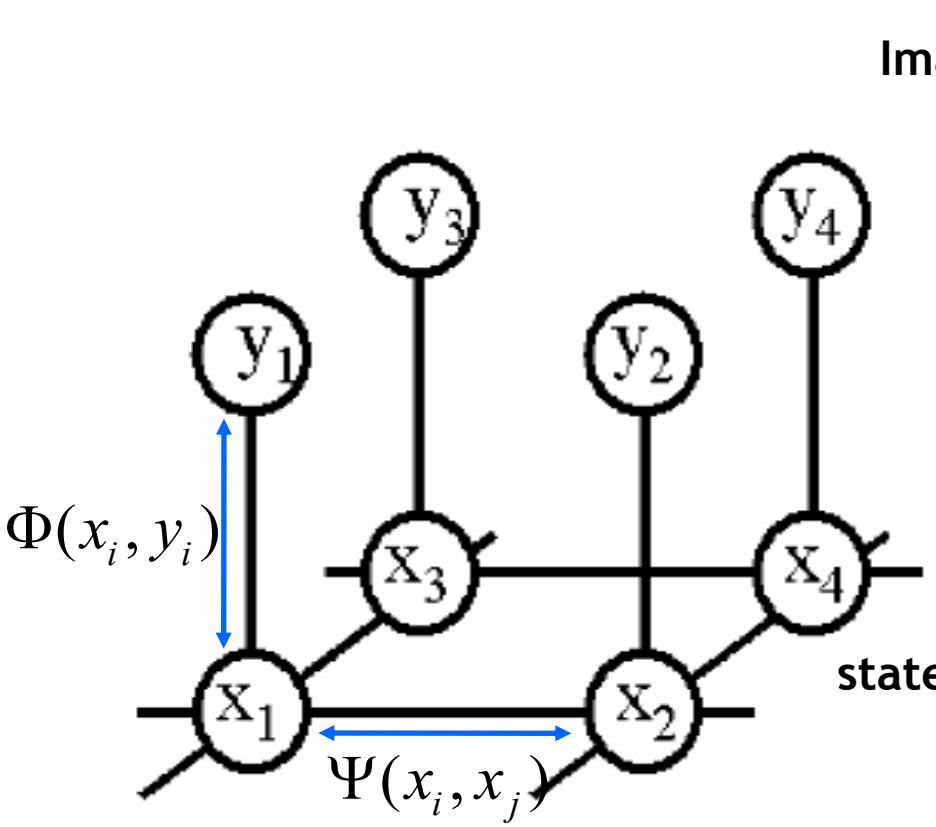
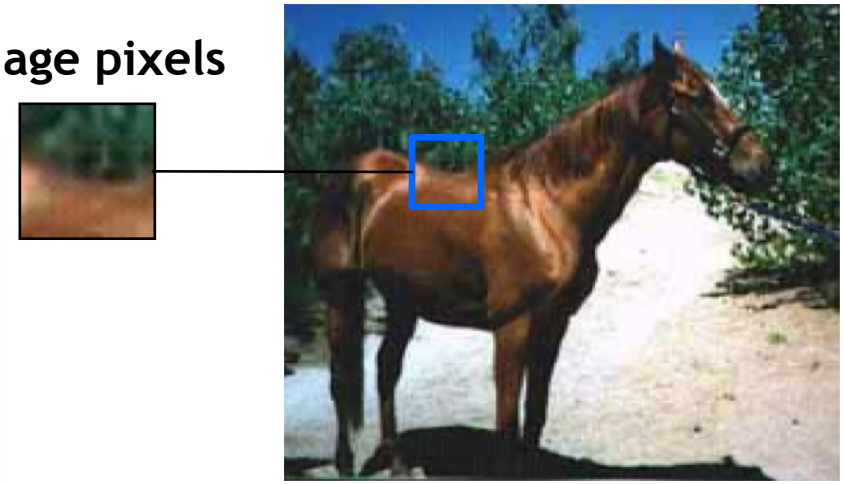


Image pixels



Image

states (e.g. foreground/background)

Slide adapted from William Freeman

Field Joint Probability

$$P(x, y) = \prod_{i} \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j)$$

states

Image

Image-state compatibility function

Local observations

state-state compatibility function

States of neighboring nodes

Slide adapted from William Freeman

Energy Formulation

- **Joint probability**

$$P(x,y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j)$$

- **Maximizing the joint probability is the same as minimizing the log**

$$\log P(x,y) = \sum_i \log\ \Phi(x_i, y_i) + \sum_{i,j} \log\ \Psi(x_i, x_j)$$

$$E(x,y) = \sum_i \varphi(x_i, y_i) \qquad + \sum_{i,j} \psi(x_i, x_j)$$

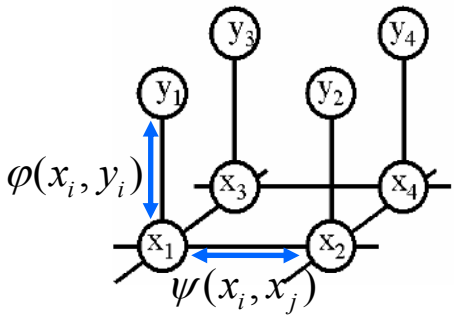
- **This is similar to free-energy problems in statistical mechanics (spin glass theory). We therefore draw the analogy and call E an *energy function*.**
- **φ and ψ are called *potentials* (unary and pairwise)**

Slide credit: B. Leibe

Energy Formulation

- **Energy function**

$$E(x,y) = \sum_i \underbrace{\varphi(x_i, y_i)}_{\text{Unary potentials}} + \sum_{i,j} \underbrace{\psi(x_i, x_j)}_{\text{Pairwise potentials}}$$

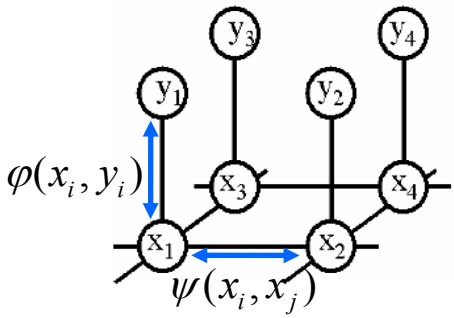


- **Unary potentials** φ
  - Encode local information about the given pixel/patch
  - How likely is a pixel/patch to be in a certain state ? (e.g. foreground/background)?

- **Pairwise potentials** ψ
  - Encode neighborhood information
  - How different is a pixel/patch's label from that of its neighbor? (e.g. here independent of image data, but later based on intensity/color/texture difference)

Slide adapted from B. Leibe

Energy Minimization

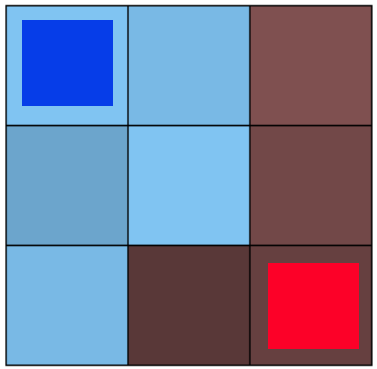


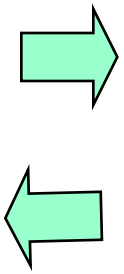
- **Goal:**
  - Infer the optimal labeling of the MRF
- **Many inference algorithms are available, e.g.**
  - Gibbs sampling, simulated annealing
  - Iterated conditional modes (ICM)
  - Variational methods
  - Belief propagation
  - Graph cuts
- **Graph Cuts have become a popular tool**
  - Globally optimal suitable for a certain class of energy functions
  - The solution can be obtained very fast for typical vision problems (~1MPixel/sec).

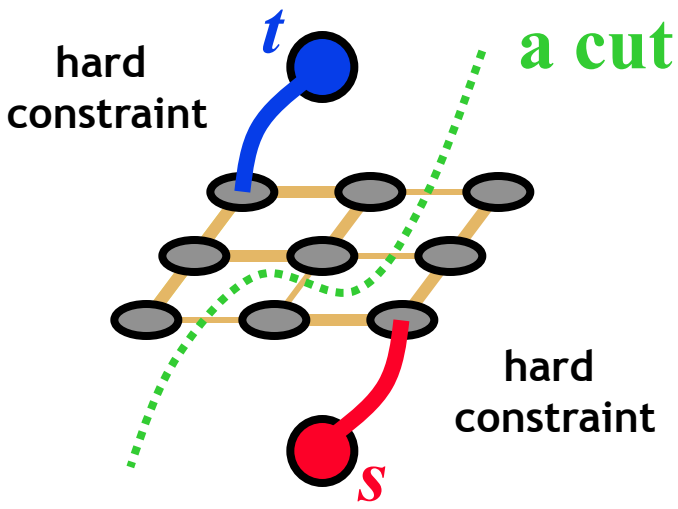
Slide credit: B. Leibe

Graph Cuts for Optimal Boundary Detection

- **Idea: convert MRF into source-sink graph**







Minimum cost cut can be computed in polynomial time

(max-flow/min-cut algorithms)

Slide adapted from Yuri Boykov

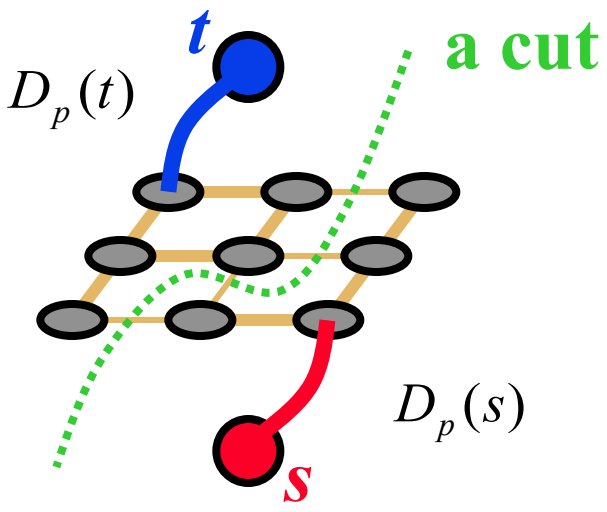
[Boykov & Jolly, ICCV'01]

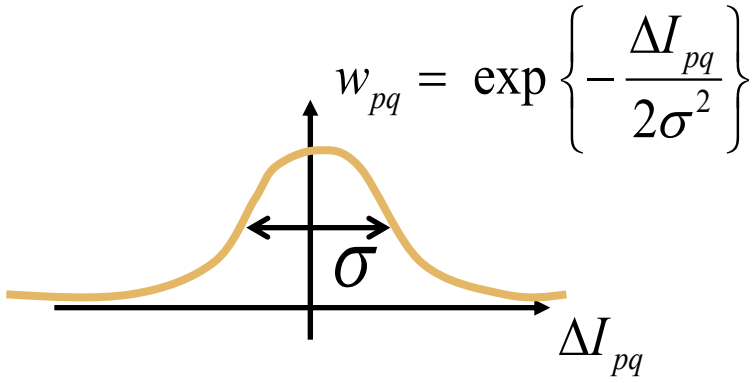
Simple Example of Energy

Regional term **Boundary term**

$$E(L) = \sum_{p} D_p(L_p) + \sum_{pq \in N} w_{pq} \cdot \delta(L_p \neq L_q)$$

t-links **n-links**



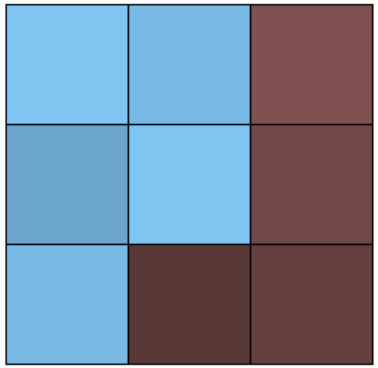


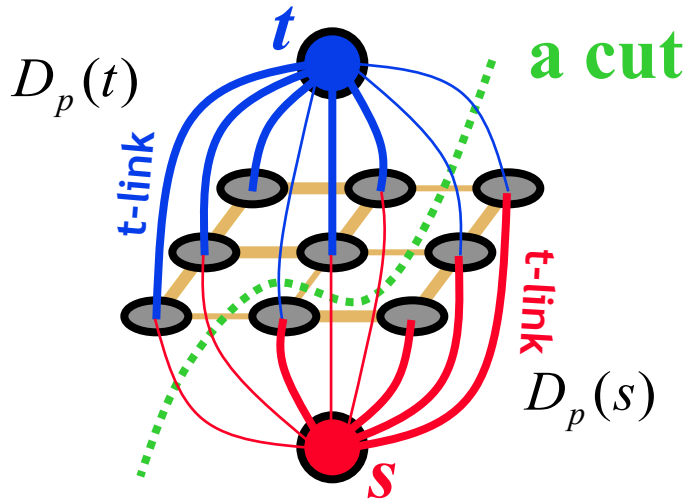
$$L_p \in \{s, t\}$$

(binary segmentation)

Slide credit: Yuri Boykov

Adding Regional Properties





Regional bias example

Suppose I^s and I^t are given "expected" intensities of object and background

$$D_p(s) \propto \exp\left(-\| I_p - I^s \|^2 / 2\sigma^2\right)$$

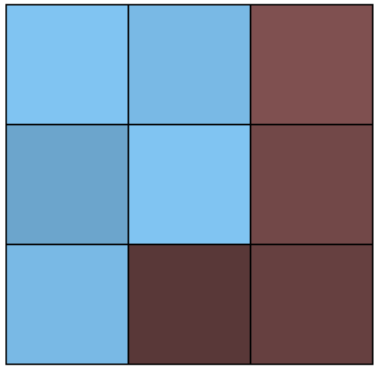
$$D_p(t) \propto \exp\left(-\| I_p - I^t \|^2 / 2\sigma^2\right)$$

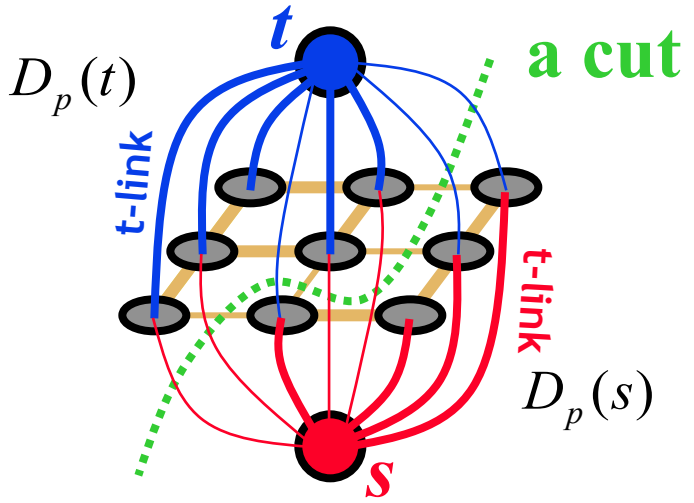
NOTE: hard constrains are not required, in general.

Slide credit: Yuri Boykov

[Boykov & Jolly, ICCV'01]

Adding Regional Properties





"expected" intensities of
object and background
I^s and I^t
can be re-estimated

$$D_p(s) \propto \exp\left(-\| I_p - I^s \|^2 / 2\sigma^2\right)$$
$$D_p(t) \propto \exp\left(-\| I_p - I^t \|^2 / 2\sigma^2\right)$$

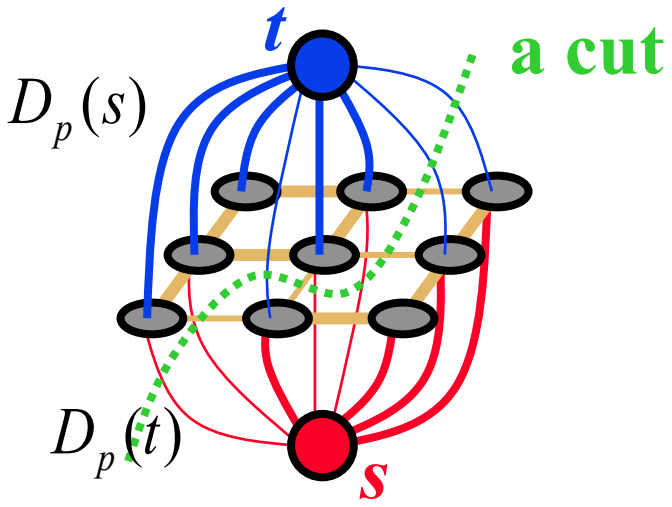
EM-style optimization

Slide credit: Yuri Boykov

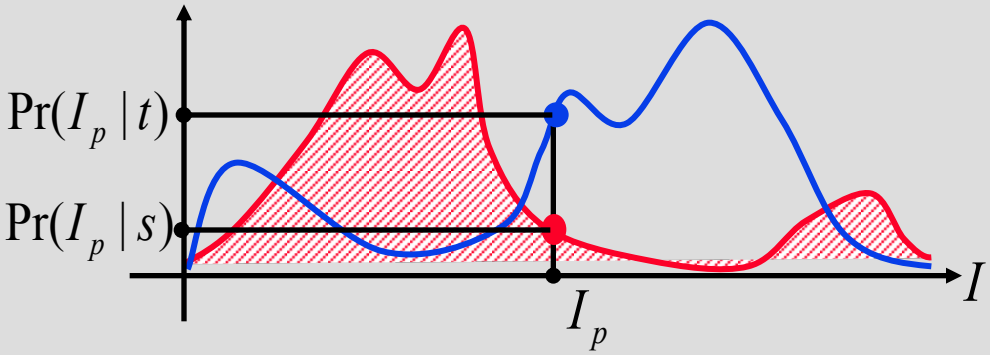
[Boykov & Jolly, ICCV'01]

Adding Regional Properties

- More generally, regional bias can be based on any intensity models of object and background



$$D_p(L_p) = -\log \Pr(I_p \mid L_p)$$



given object and background intensity histograms

Slide credit: Yuri Boykov

[Boykov & Jolly, ICCV'01]

How to Set the Potentials? Some Examples

- **Color potentials**
  - **e.g. modeled with a Mixture of Gaussians**

$$\pi(x_i, y_i; \theta_\pi) = \log \sum_k \theta_\pi(x_i, k) P(k \mid x_i) N(y_i; \bar{y}_k, \Sigma_k)$$

- **Edge potentials**
  - **e.g. a "contrast sensitive Potts model"**

$$\phi(x_i, x_j, g_{ij}(y); \theta_\phi) = -\theta_\phi^T g_{ij}(y) \delta(x_i \neq x_j)$$

where

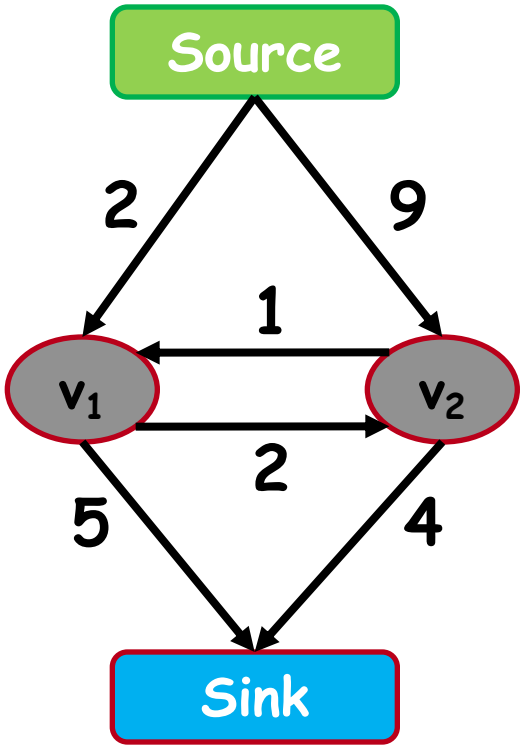
$$g_{ij}(y) = e^{-\beta \left\| y_i - y_j \right\|^2} \qquad \beta = 2 \cdot avg\left(\left\| y_i - y_j \right\|^2 \right)$$

- **Parameters θ_π, θ_ϕ need to be learned, too!**

Slide credit: B. Leibe

[Shotton & Winn, ECCV'06]

How Does it Work? The s-t-Mincut Problem



Graph (V, E, C)

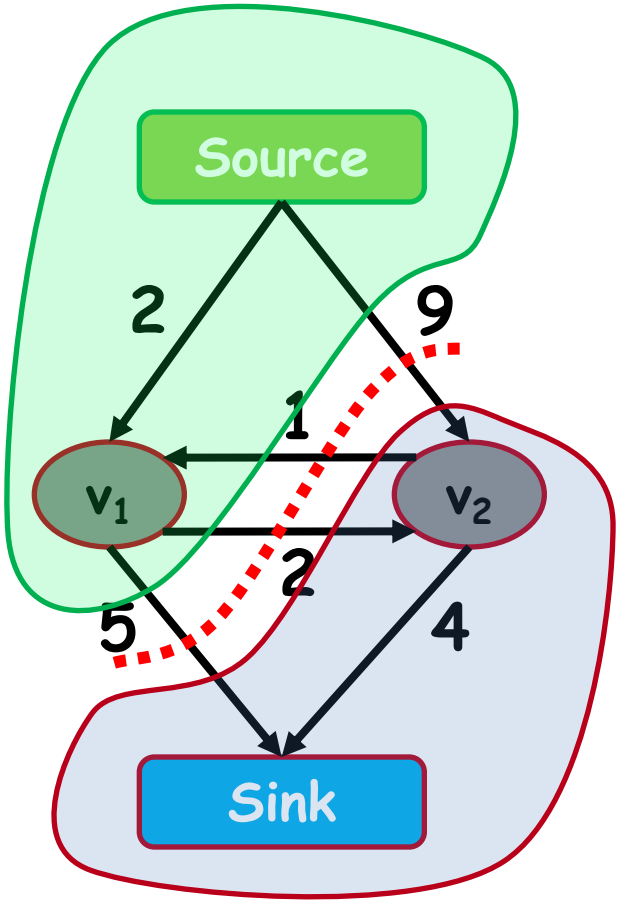
Vertices $V = \{v_1, v_2 \ldots v_n\}$

Edges $E = \{(v_1, v_2) \ldots.\}$

Costs $C = \{c_{(1, 2)} \ldots.\}$

Slide credit: Pushmeet Kohli

The s-t-Mincut Problem



5 + 2 + 9 = 16

What is an st-cut?

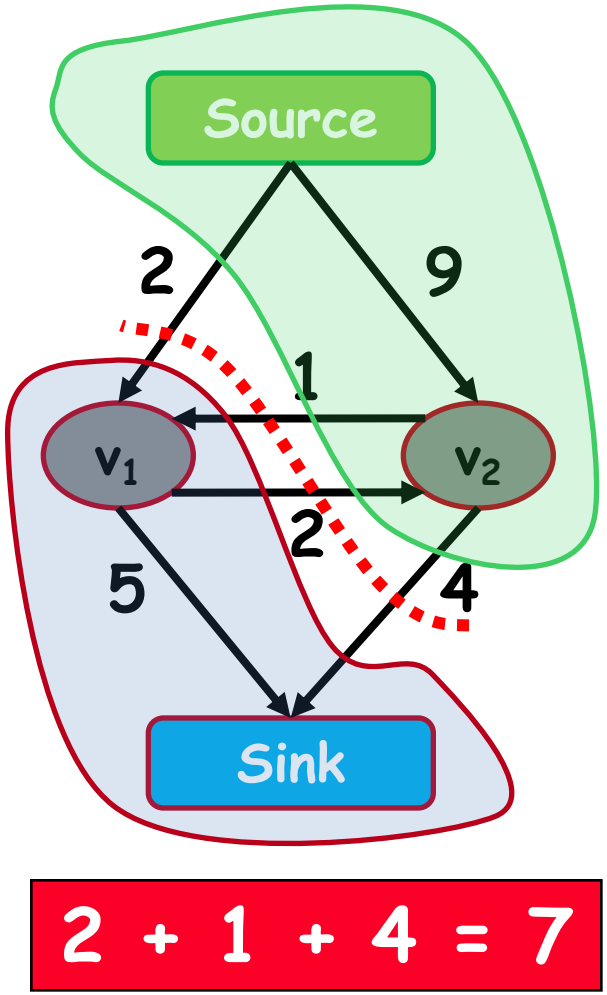
An st-cut (S,T) divides the nodes between source and sink.

What is the cost of a st-cut?

Sum of cost of all edges going from S to T

Slide credit: Pushmeet Kohli

The s-t-Mincut Problem



What is an st-cut?

An st-cut (S,T) divides the nodes between source and sink.

What is the cost of a st-cut?

Sum of cost of all edges going from S to T

What is the st-mincut?

st-cut with the minimum cost

Slide credit: Pushmeet Kohli

History of Maxflow Algorithms

Augmenting Path and Push-Relabel

year	discoverer(s)	bound
1951	Dantzig	$O(n^2mU)$
1955	Ford & Fulkerson	$O(m^2U)$
1970	Dinitz	$O(n^2m)$
1972	Edmonds & Karp	$O(m^2 \log U)$
1973	Dinitz	$O(nm \log U)$
1974	Karzanov	$O(n^3)$
1977	Cherkassky	$O(n^2m^{1/2})$
1980	Galil & Naamad	$O(nm \log^2 n)$
1983	Sleator & Tarjan	$O(nm \log n)$
1986	Goldberg & Tarjan	$O(nm \log(n^2/m))$
1987	Ahuja & Orlin	$O(nm + n^2 \log U)$
1987	Ahuja et al.	$O(nm \log(n\sqrt{\log U}/m))$
1989	Cheriyan & Hagerup	$E(nm + n^2 \log^2 n)$
1990	Cheriyan et al.	$O(n^3/\log n)$
1990	Alon	$O(nm + n^{8/3} \log n)$
1992	King et al.	$O(nm + n^{2+\epsilon})$
1993	Phillips & Westbrook	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$
1994	King et al.	$O(nm \log_{m/(n \log n)} n)$
1997	Goldberg & Rao	$O(m^{3/2} \log(n^2/m) \log U)$ $O(n^{2/3}m \log(n^2/m) \log U)$

n: #nodes

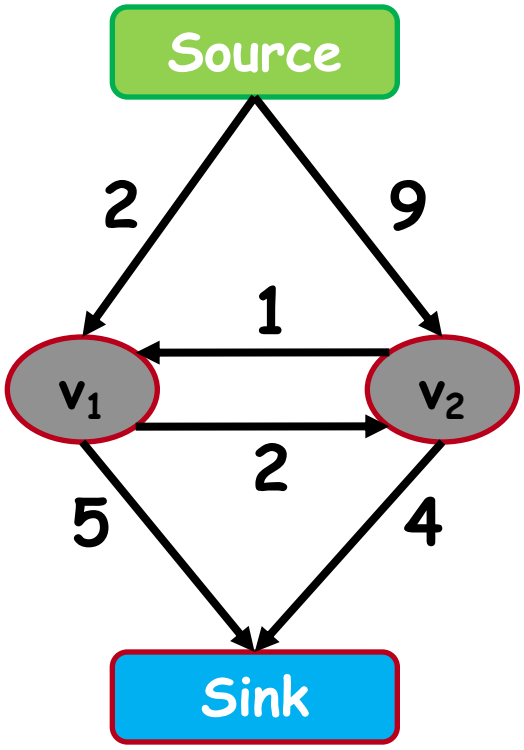
m: #edges

U: maximum edge weight

Algorithms assume non-negative edge weights

Slide credit: Andrew Goldberg

How to Compute the s-t-Mincut?



Solve the dual maximum flow problem

Compute the maximum flow between Source and Sink

Constraints

Edges: Flow < Capacity

Nodes: Flow in = Flow out

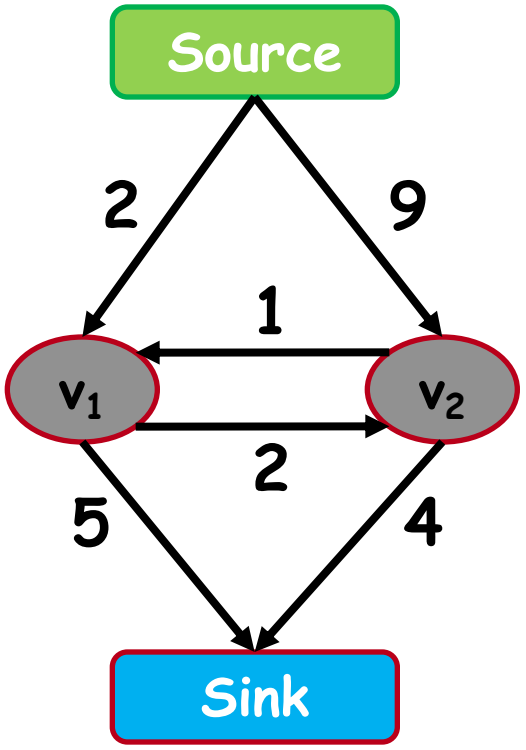
Min-cut/Max-flow Theorem

In every network, the maximum flow equals the cost of the st-mincut

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 0



Augmenting Path Based Algorithms

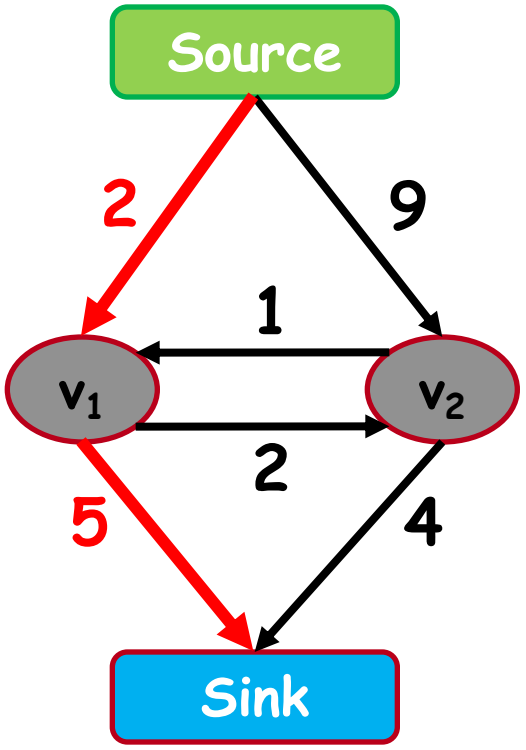
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 0



Augmenting Path Based Algorithms

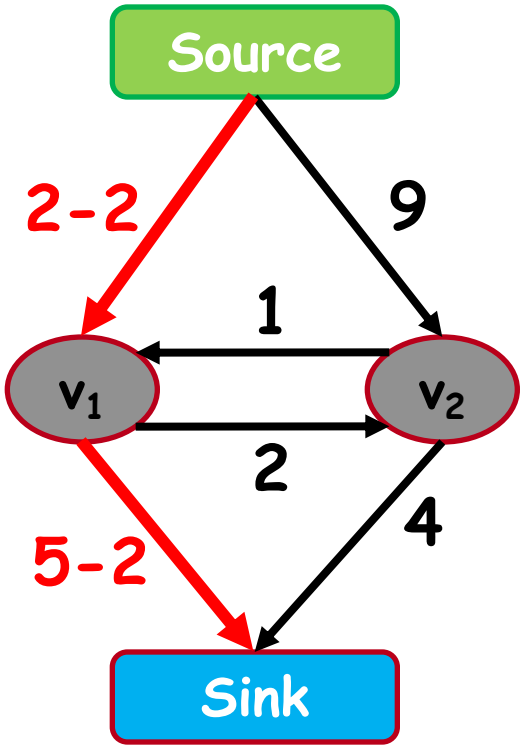
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 0 + 2



Augmenting Path Based Algorithms

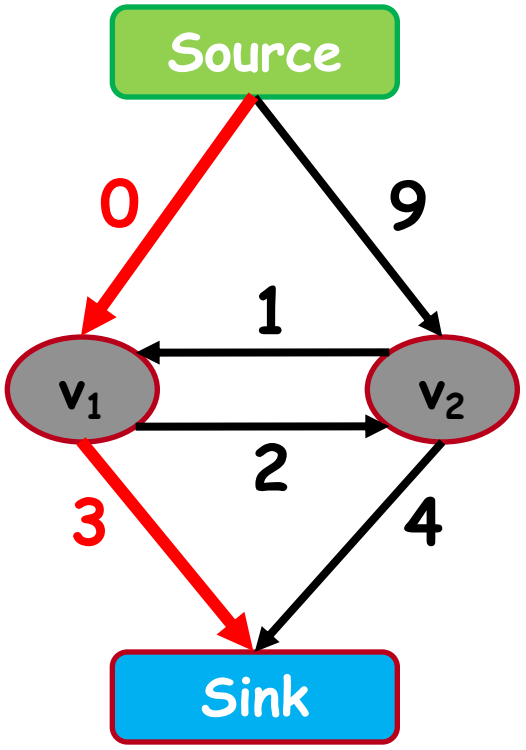
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 2



Augmenting Path Based Algorithms

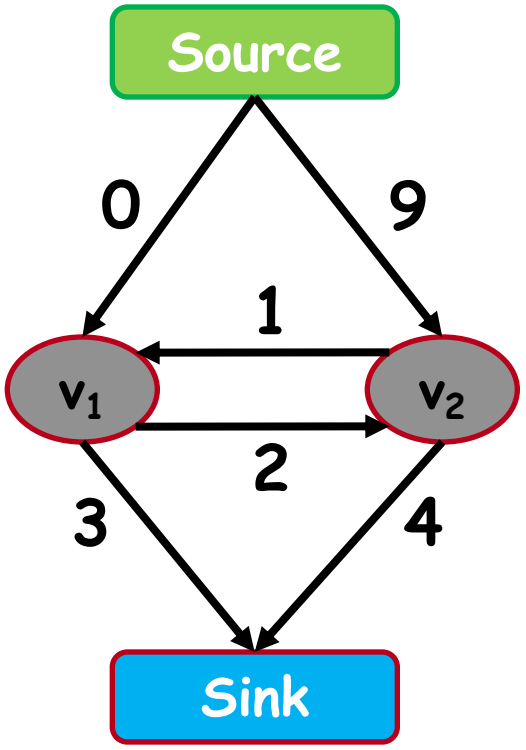
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 2



Augmenting Path Based Algorithms

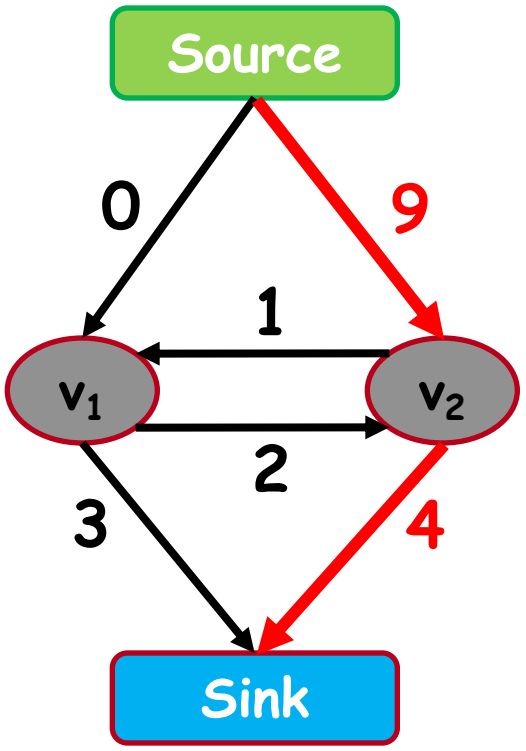
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 2



Augmenting Path Based Algorithms

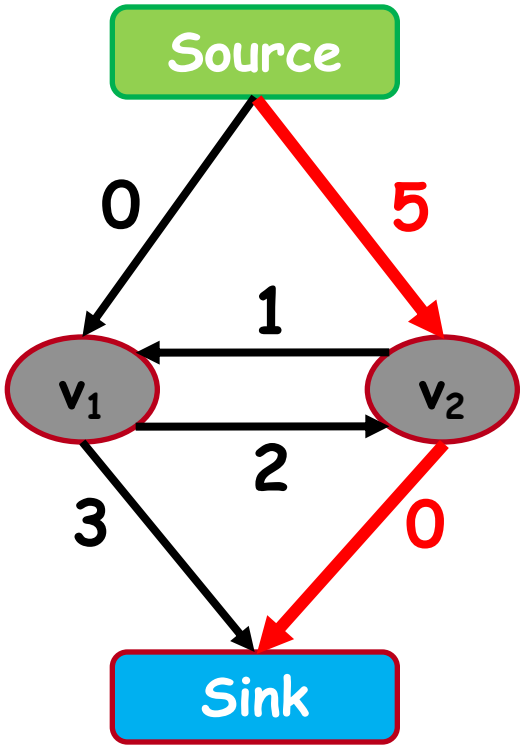
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 2 + 4



Augmenting Path Based Algorithms

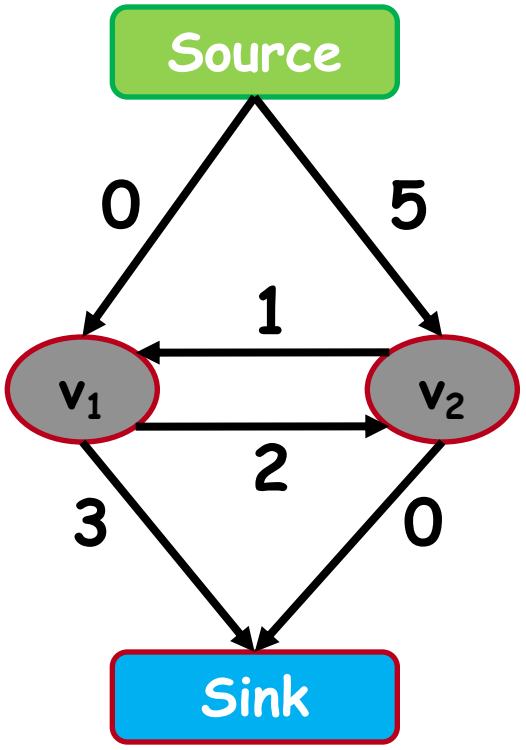
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 6



Augmenting Path Based Algorithms

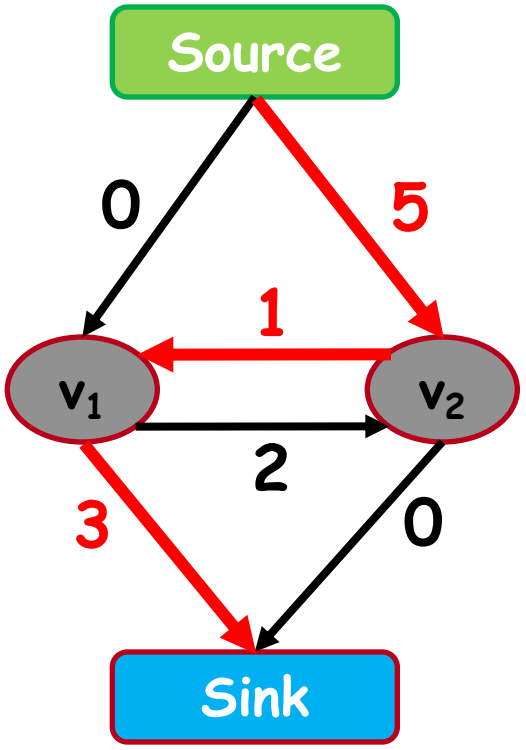
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 6



Augmenting Path Based Algorithms

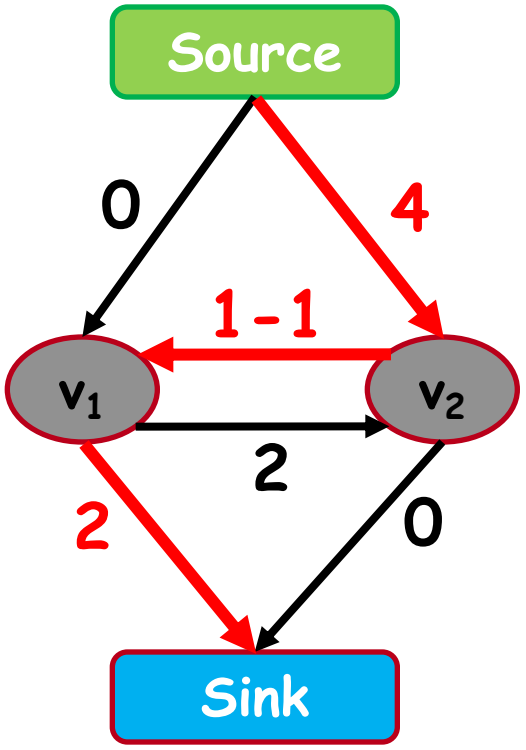
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 6 + 1



Augmenting Path Based Algorithms

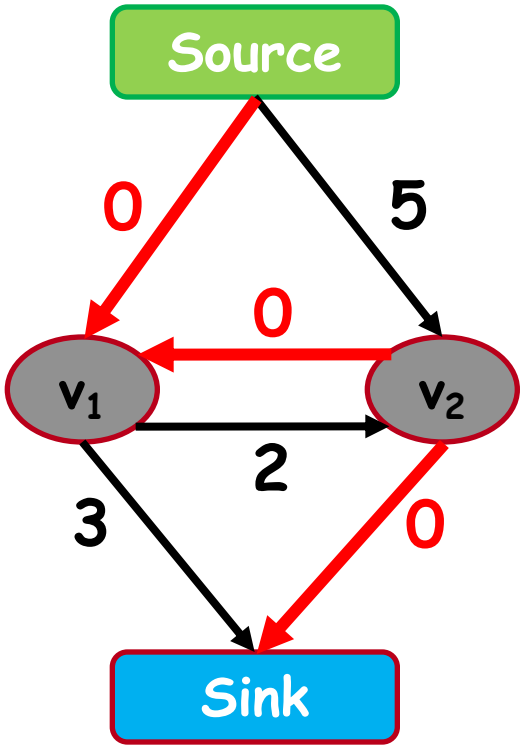
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms

Flow = 7



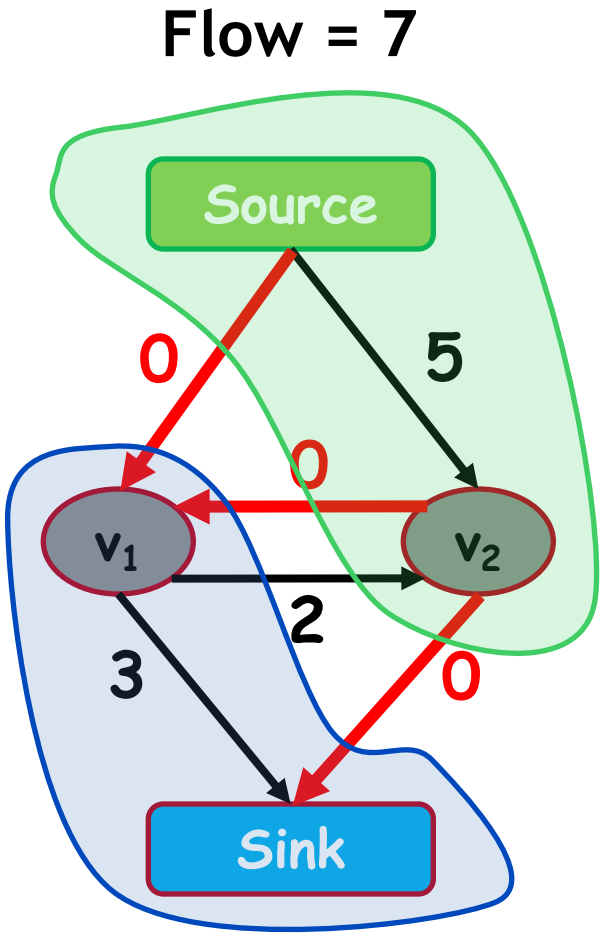
Augmenting Path Based Algorithms

1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow Algorithms



Augmenting Path Based Algorithms

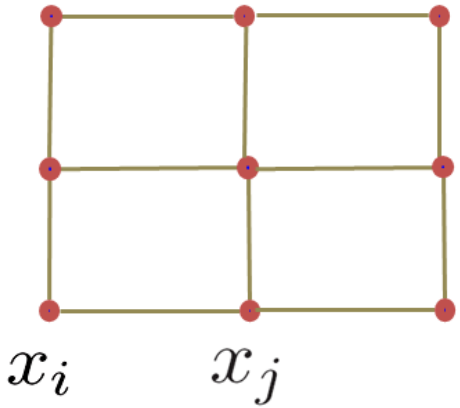
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

Algorithms assume non-negative capacity

Slide credit: Pushmeet Kohli

Maxflow in Computer Vision

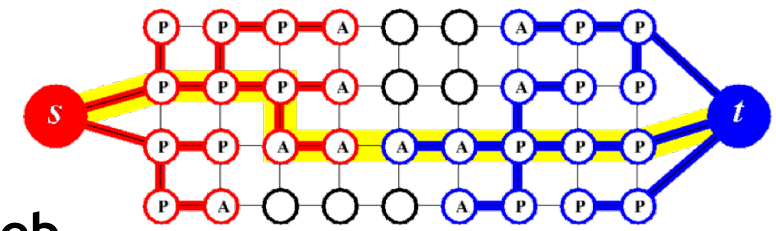
- Specialized algorithms for vision problems
  - Grid graphs
  - Low connectivity (m ~ O(n))



- Dual search tree augmenting path algorithm

  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently
  - High worst-case time complexity
  - Empirically outperforms other algorithms on vision problems
  - Efficient code available on the web

    http://www.adastral.ucl.ac.uk/~vladkolm/software.html



Slide credit: Pushmeet Kohli

When Can s-t Graph Cuts Be Applied?

Regional term Boundary term

$$E(L) = \sum_{p} E_p(L_p) + \sum_{pq \in N} E(L_p, L_q)$$

t-links n-links

$$L_p \in \{s, t\}$$

- s-t graph cuts can only globally minimize binary energies that are submodular. [Boros & Hummer, 2002, Kolmogorov & Zabih, 2004]

E(L) can be minimized by *s-t* graph cuts $\Leftrightarrow$ $E(s,s) + E(t,t) \leq E(s,t) + E(t,s)$

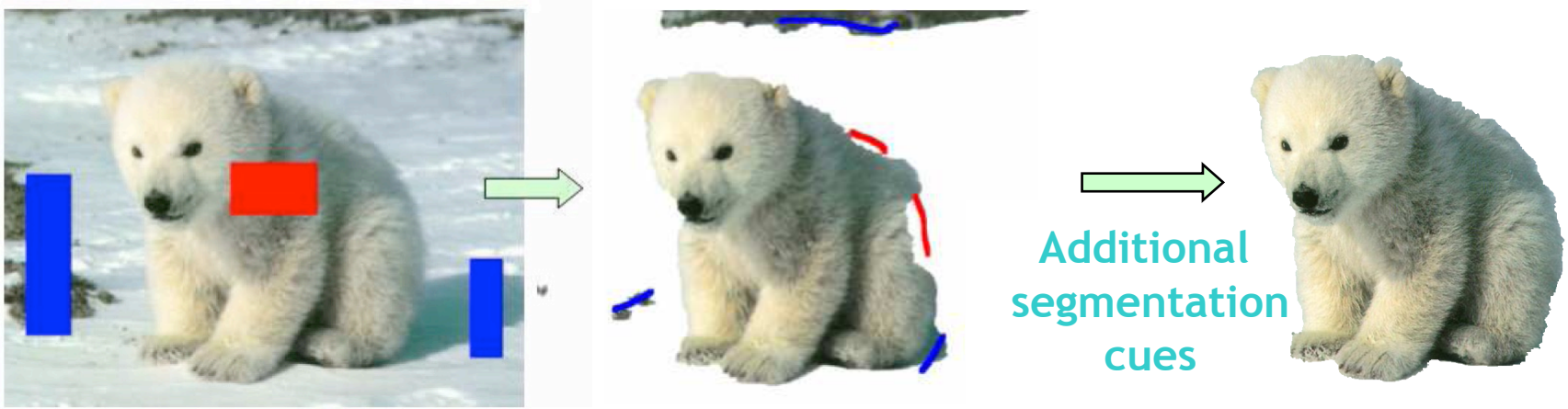
Submodularity ("convexity")

- Non-submodular cases can still be addressed approximately, and with strong guarantees/bounds (e.g. alpha-expansion algorithm, not in this lecture)

Slide credit: B. Leibe

GraphCut Applications: "GrabCut"

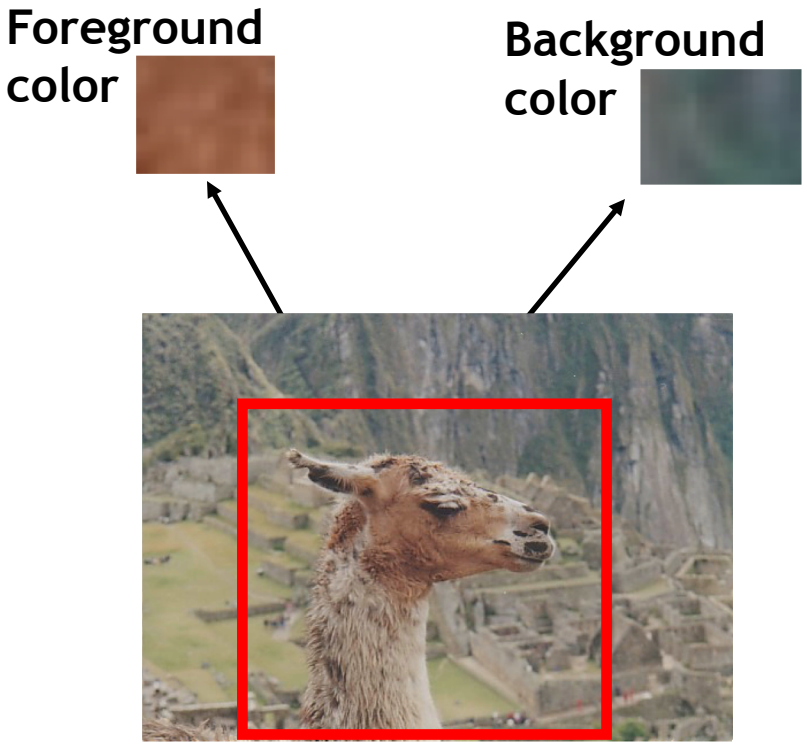
- **Interactive Image Segmentation** [Boykov & Jolly, ICCV'01]
  - Rough region cues sufficient
  - Segmentation boundary can be extracted from edges
- **Procedure**
  - User marks foreground and background regions with a brush → get initial segmentation → correct by additional brush strokes



User segmentation cues

Slide adapted from Matthieu Bray

GrabCut: Data Model



Global optimum of the unary energy

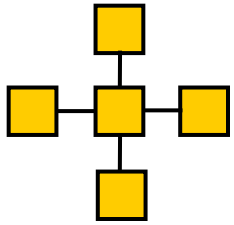
- Obtained from interactive user input
  - User marks foreground and background regions with a brush
  - Alternatively, user can specify a bounding box

Slide adapted from Carsten Rother

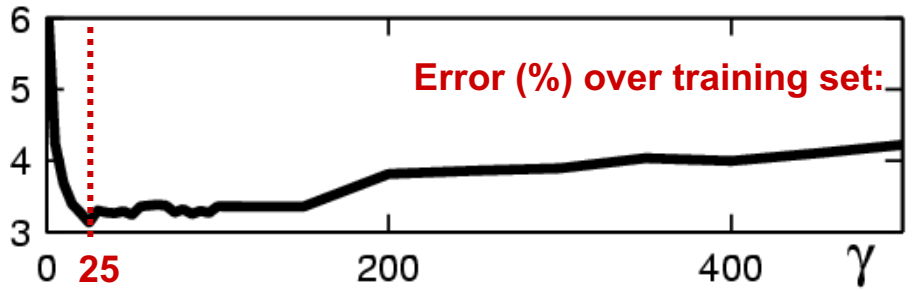
GrabCut: Spatial coherence Model

- **An object is a coherent set of pixels:**

$$\psi(x,y) = \gamma \sum_{(m,n)\in C} \delta\left[x_n \neq x_m\right] \mathrm{e}^{-\beta\|y_m - y_n\|^2}$$



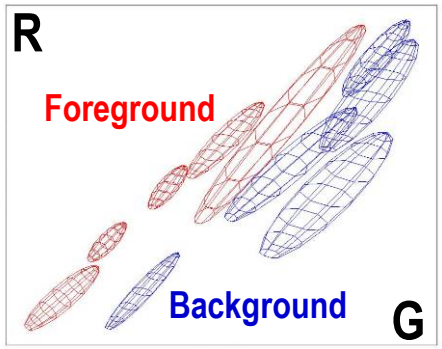
How to choose γ?



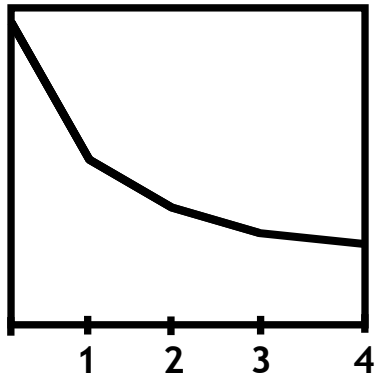
Slide credit: Carsten Rother

Iterated Graph Cuts

Result



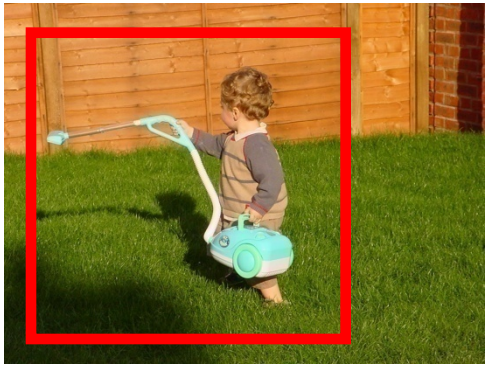
Color model
(Mixture of Gaussians)



Energy after
each iteration

Slide credit: Carsten Rother

GrabCut: Example Results



Summary: Graph Cuts Segmentation

- **Pros**
  - Powerful technique, based on probabilistic model (MRF).
  - Applicable for a wide range of problems.
  - Very efficient algorithms available for vision problems.
  - Becoming a de-facto standard for many segmentation tasks.

- **Cons/Issues**
  - Graph cuts can solve a limited (but useful) class of models
    - Submodular energy functions
    - Can capture only part of the expressiveness of MRFs
    - Only approximate algorithms available for multi-label case (except for special cases with particular topology and/or pairwise potentials)

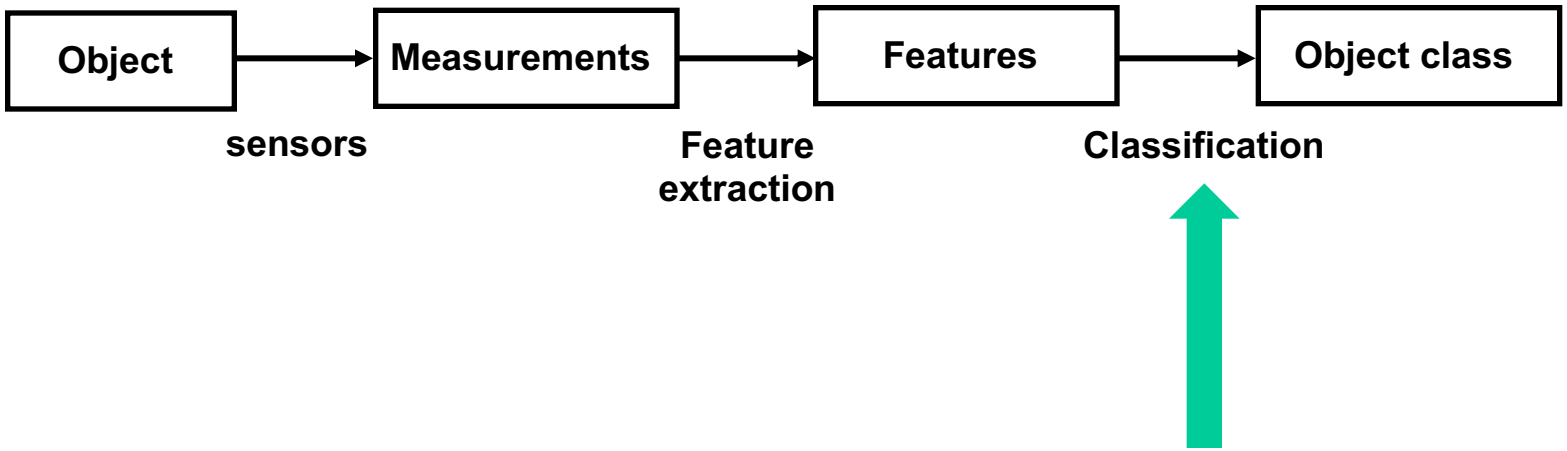
Slide credit: B. Leibe

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

Statistical pattern recognition

- General scheme
- Feature based (fixed, pre-deep-learning)
- Learn classifier



Discriminative learning: principles

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

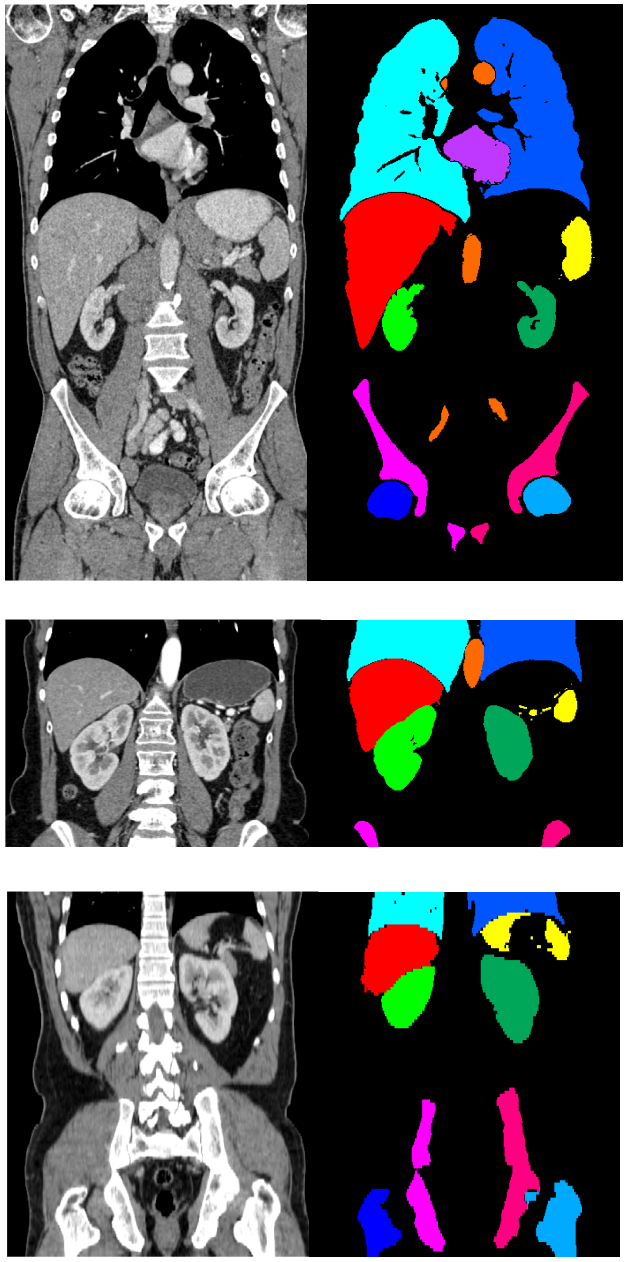
$$f(\vec{v}_j) = c_j$$

In terms of probabilities 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: training data



- Composed of images and the corresponding segmentations of the objects
- Application dependent
- The “ground-truth” segmentations are often done manually or with a semi-automatic algorithm
- Can be VERY expensive
- VERY valuable

Discriminative learning: Segmentation Training phase

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

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

- Multiple types of mappings are possible…
- but they contain parameters to be set
- and that is what training is about

Discriminative learning: Segmentation Testing phase

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

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

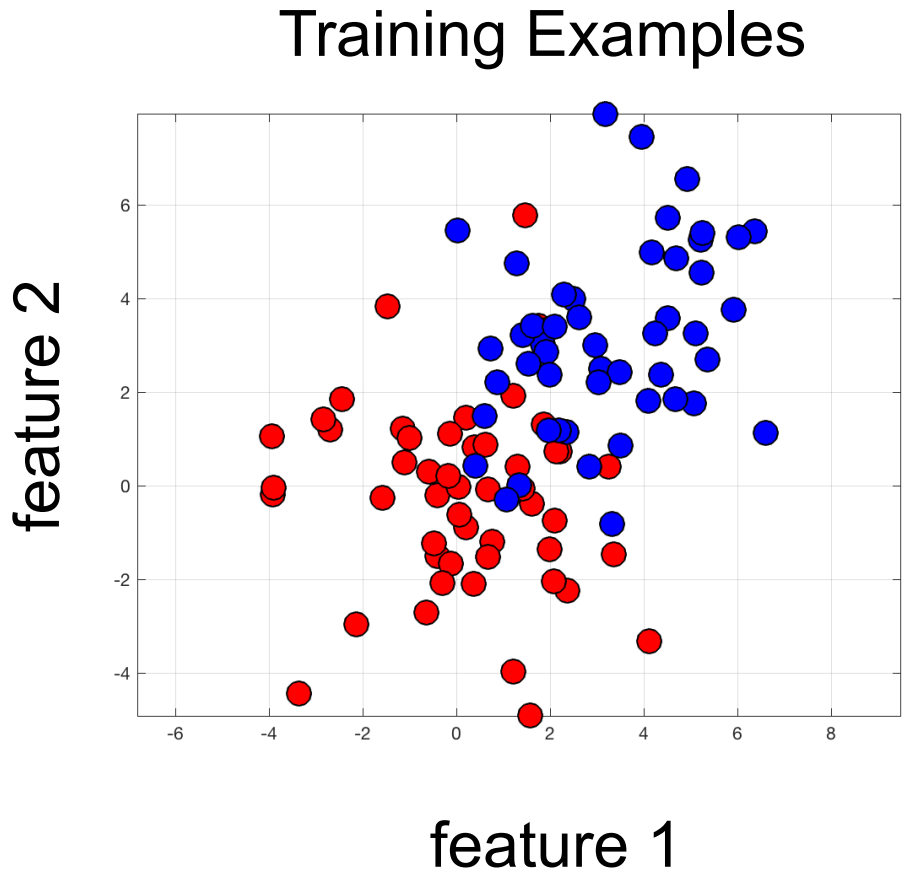
- Extract the same features used during training
- Use the mapping learned before to label

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

K-nearest neighbors - KNN

- Find the K nearest neighbors within the training dataset for a sample (in feature space)
- For training examples we know the features and the labels

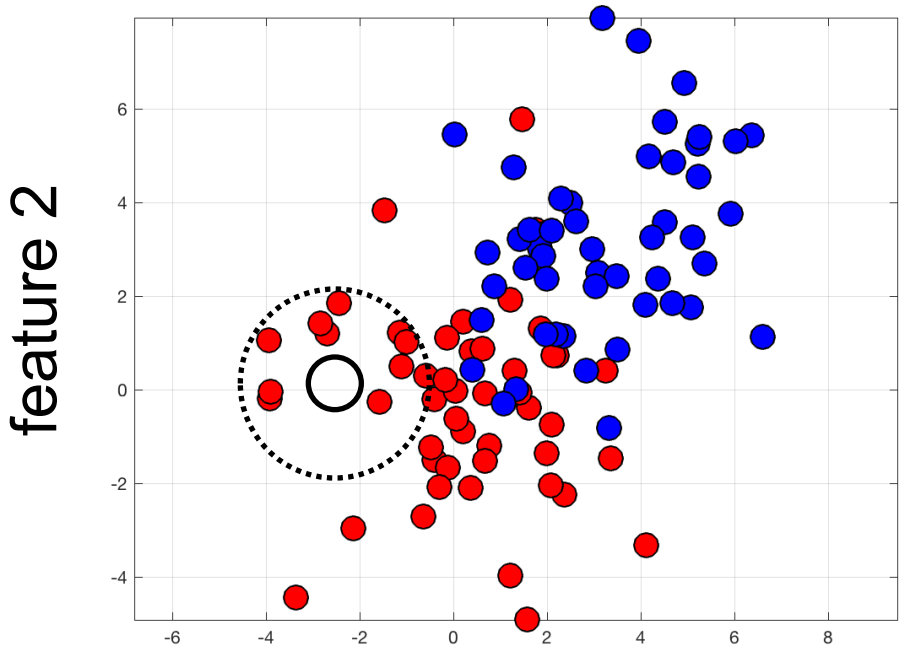


K-nearest neighbors - KNN

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

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

Training Examples



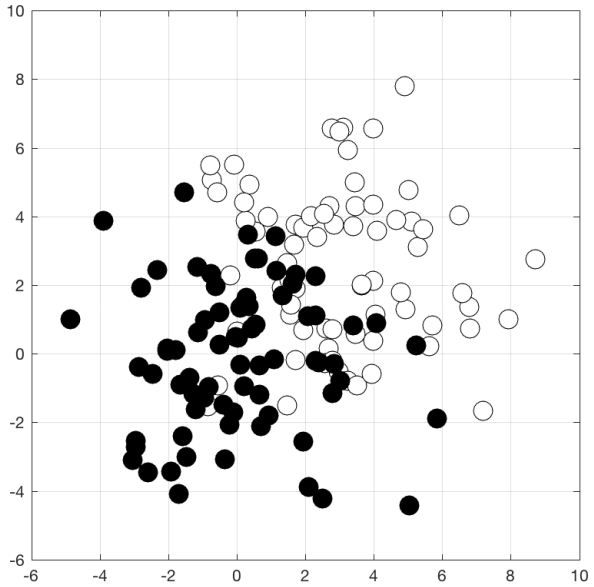
Find the K training examples with minimum distance

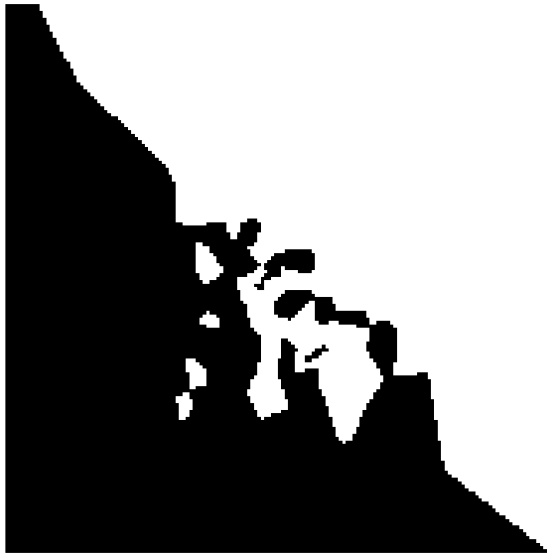
$$d(\vec{v}, \vec{v}_n)$$

Mapping is the function of the corresponding labels

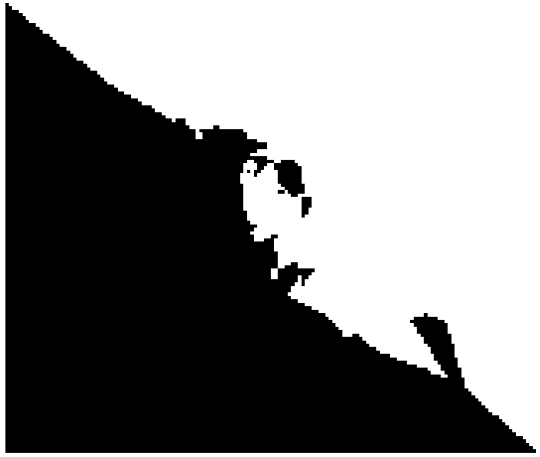
$$f(\vec{v}) = f(c_1, \ldots, c_K)$$

KNN defines a partitioning – majority voting

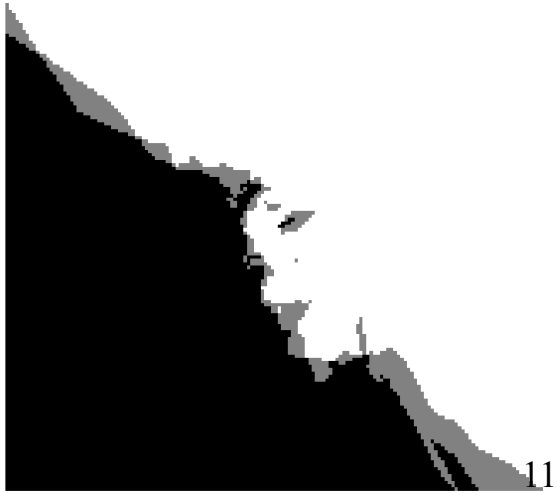




K=1

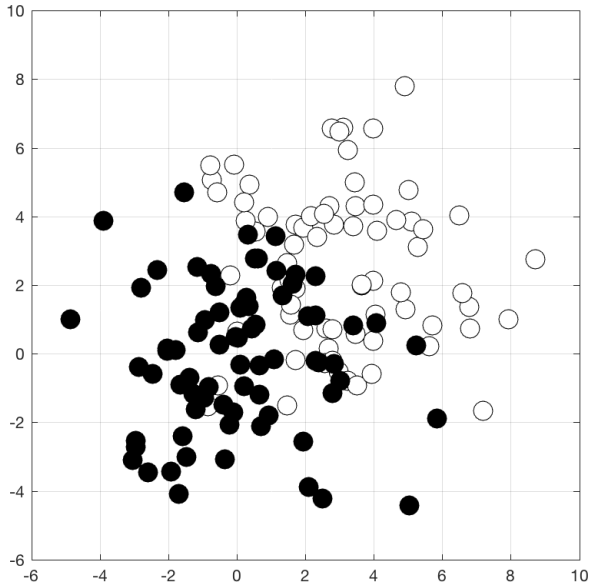


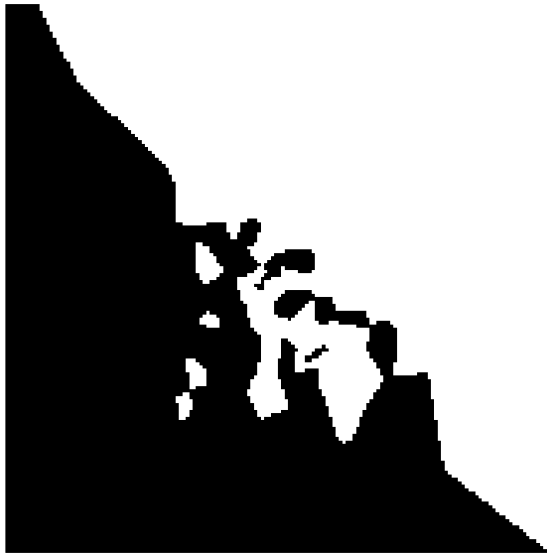
K=5



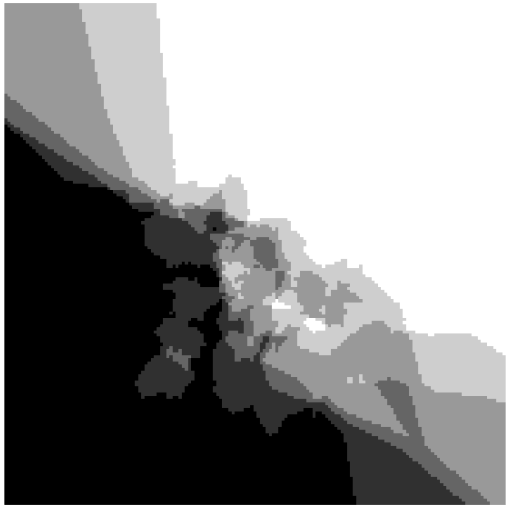
K=10

Probabilities for partitioning - proportions

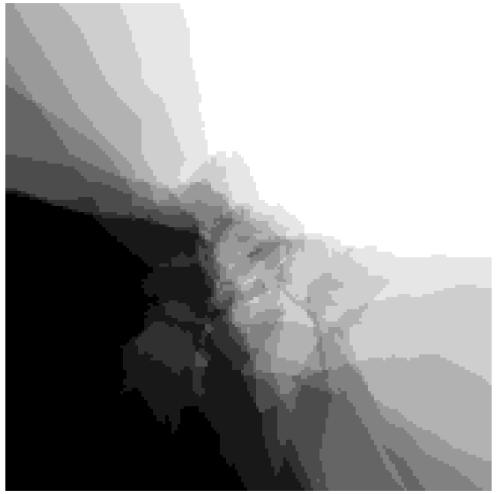




K=1



K=5



K=10

Remarks on KNN

Pros

- Very simple to implement
- Very simple to understand
- Efficient implementations possible for approx. NNs
- Distance definition is flexible

Cons

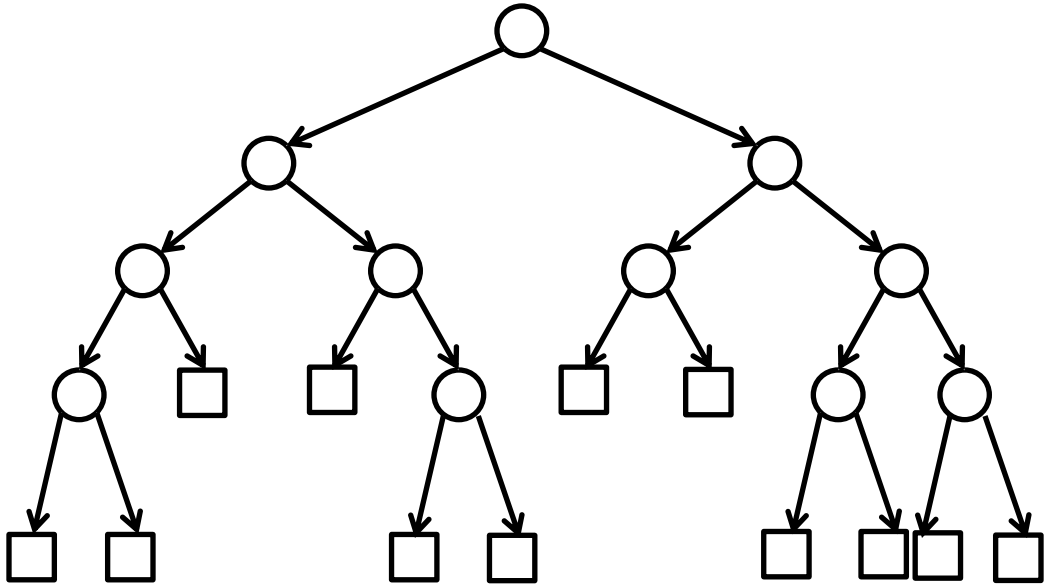
- Highly depends on the definitions and K
- Need to keep the entire data in memory for distance computations
- For high dimensional problems (with high d) might need many training samples for accuracy
- Other methods have better generalization ability

Topics of This Lecture

- Introduction
- Segmentation as clustering
  - k-Means
  - Mixture of Gaussians, EM
  - Model-free clustering: mean-shift
- Hough transforms (edge-based)
- Interactive Segmentation with GraphCuts
- Learning-based approaches:
  - K-nearest neighbor
  - Random forests
  - Deep learning (in ‘object class recognition’ lecture)

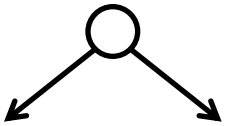
Discriminative learning: Random Forests

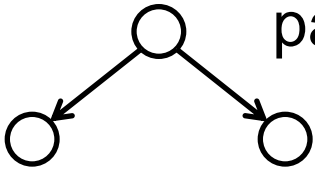
- Binary decision trees



: internal nodes

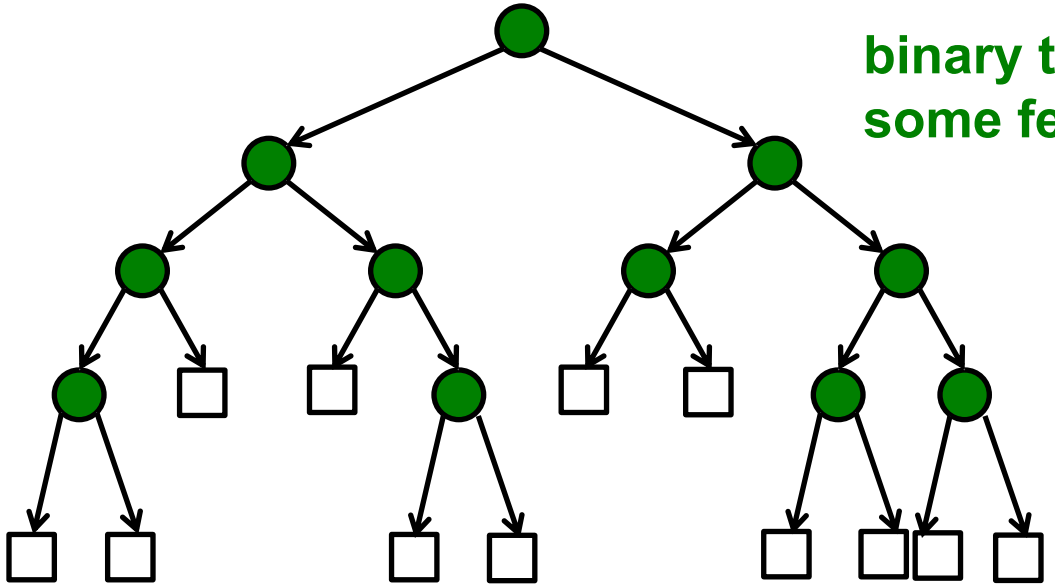

: leaf nodes

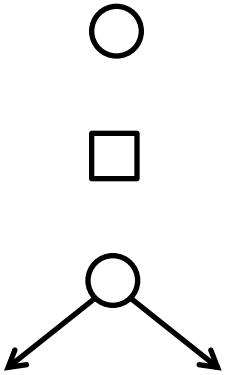

: splits

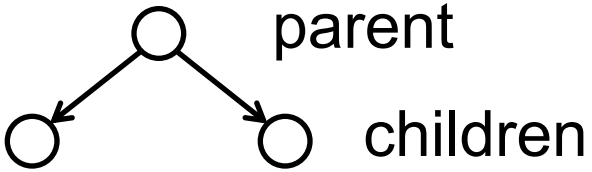


Discriminative learning: Random Forests

- Binary decision trees

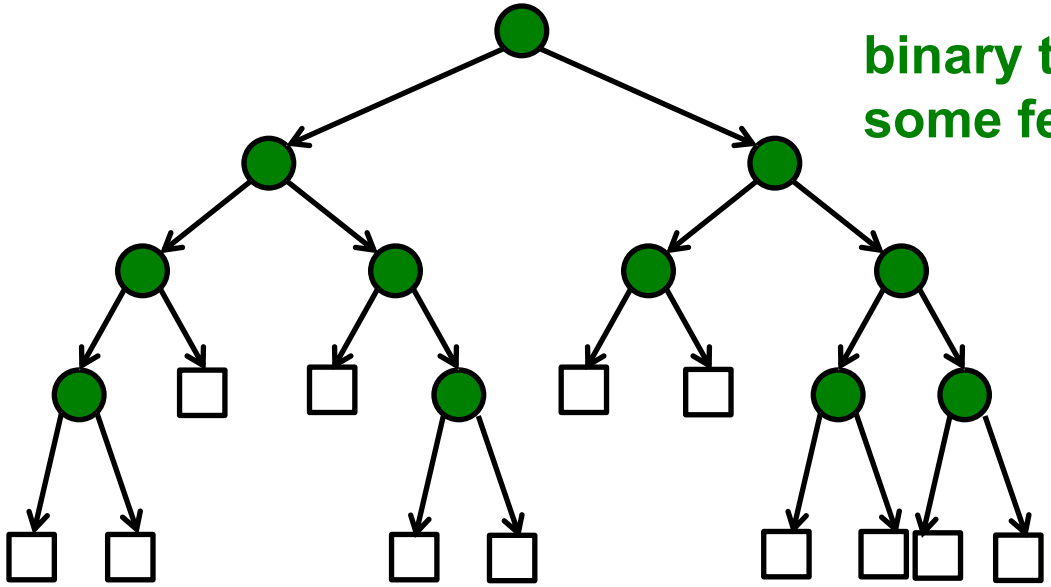






Discriminative learning: Random Forests

- Binary decision trees

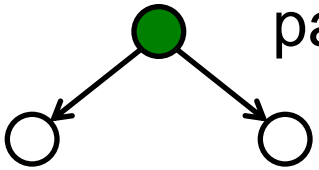


binary test on some feature

: internal nodes

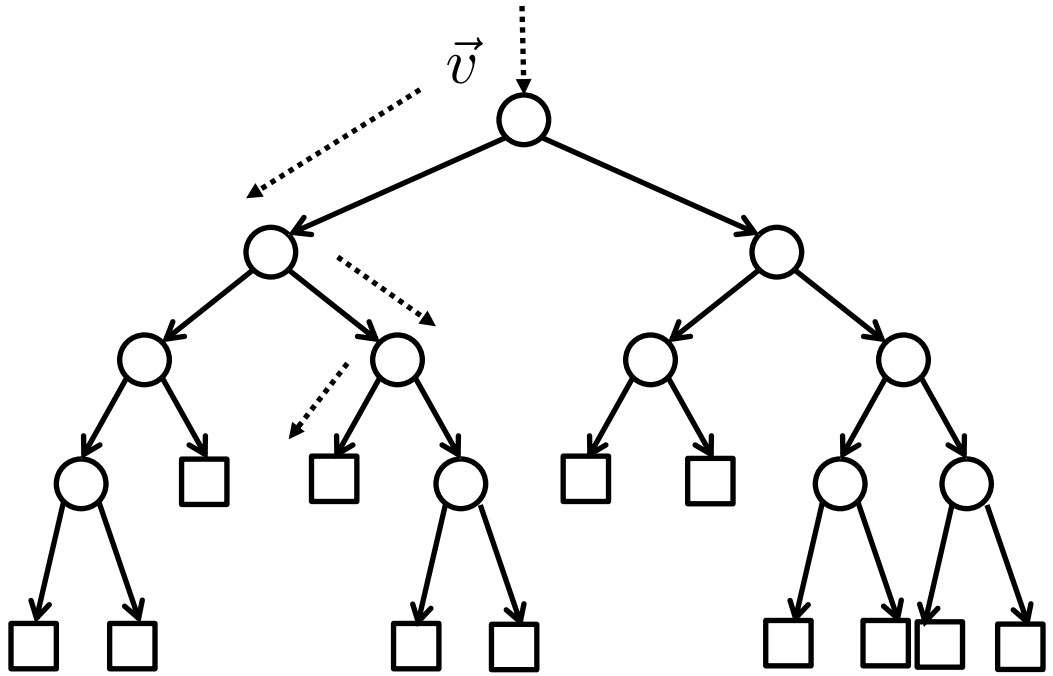
: leaf nodes

: splits



depending on test outcome the sample continues to the left or the right child

Discriminative learning: Decision Trees



A sample – e.g. an image patch - is dropped in at the top node.

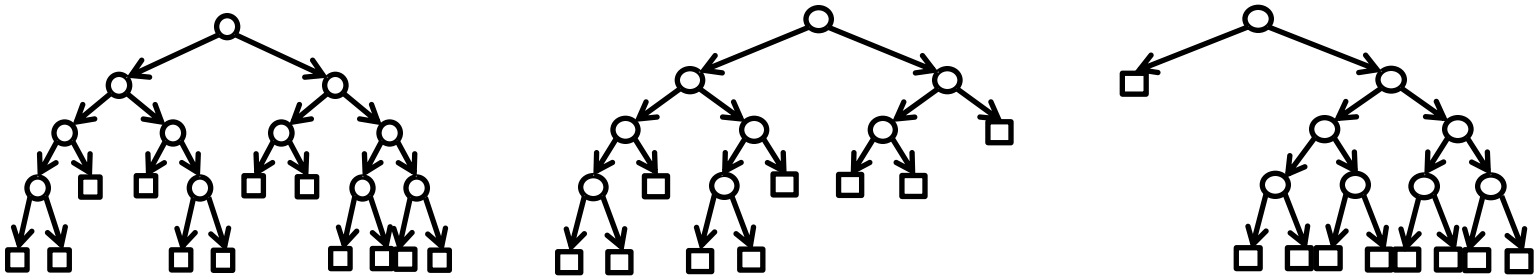
At each internal node there is a binary question on the features extracted from the patch.

The sample will end up in one of the node leafs.

At each leaf node there is a prediction about the class/label for patches getting there; and it changes from leaf node to leaf node.

Discriminative learning: Random Forest

- A `random forest' consists of an ensemble of different decision trees – hence *forest*



- There are several aspects of randomness in how the trees are trained, hence *random*

- Once the forest has been trained, a sample is put in at the top of every tree, and the different lea in which it ends decide on class membership (typically averaging predictions of each tree)

Discriminative learning: Random Forest

The are two phases

1. *Training*: setting up all the trees in the forest

2. *Testing/Inference*: applying the trees to all samples of a new incoming image, one sample at a time

Random Forests are slow to train, but fast to apply during testing. Of course, it is mainly during testing that speed matters

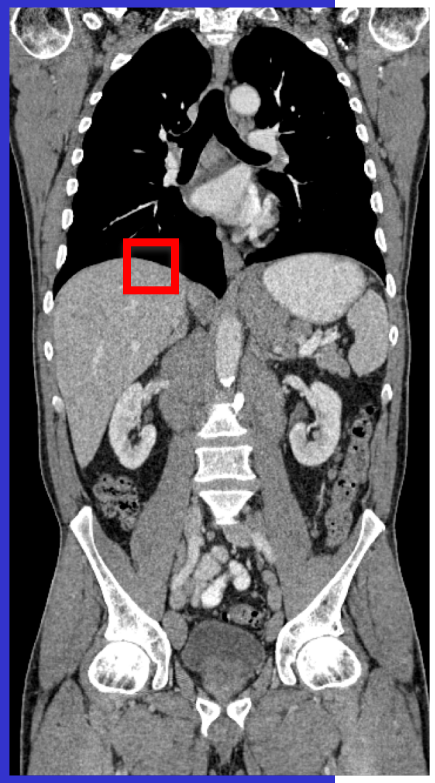
Discriminative learning: Random Forest

T R A I N I N G

Two aspects:

- deciding on the (binary) node tests
- deciding on when to stop splitting

Random Forests: training



We need training data, e.g. segmentations performed by a human expert… expensive !

$\rightarrow \vec{v}_1 = \{v_{11}, \ldots, v_{1d}\}$

$\rightarrow \vec{v}_2 = \{v_{21}, \ldots, v_{2d}\}$

$\rightarrow \vec{v}_n = \{v_{n1}, \ldots, v_{nd}\}$

For the different samples from the segmented regions, the expert indicates their segmentation class

Random Forests: parameterization

For measurement vector $\vec{v}$, 2 levels of parameterization

- What are the tests (for internal node n) ?

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

- How are the predictions done (for leaf node l) ?

$$f_l(\vec{v}) = c \quad \text{(where } c \text{ is a class)}$$

Both of these are learned during training

Various alternatives for both

Random Forests: the node tests

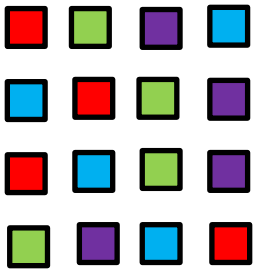
A simple node test that is often used

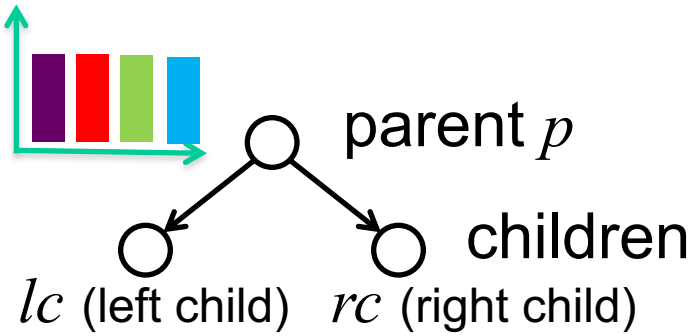
- Binary stump (thresholding on one feature)

$$f_n(\vec{v}) = \begin{cases} 0, & f_k > \tau \\ 1, & f_k \leq \tau \end{cases}$$

- During training the parameters k (choice of the feature) and τ are determined for each node

Random Forests: the node tests



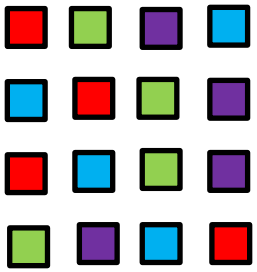


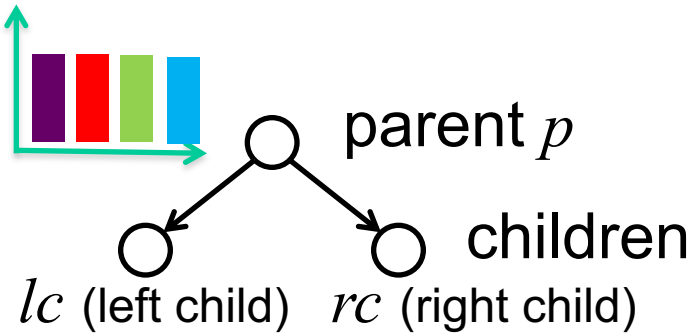
During training, samples are used for which the true class is known.

The goal is to find a node test that maximally reduces the uncertainty of class membership for samples ending up in the child nodes, compared to at their parent node.

Uncertainty can be quantified through the entropy of the class distribution at a node.

Random Forests: the node tests

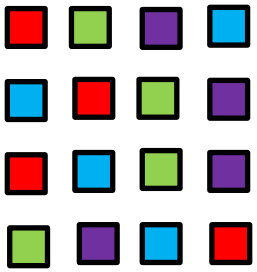


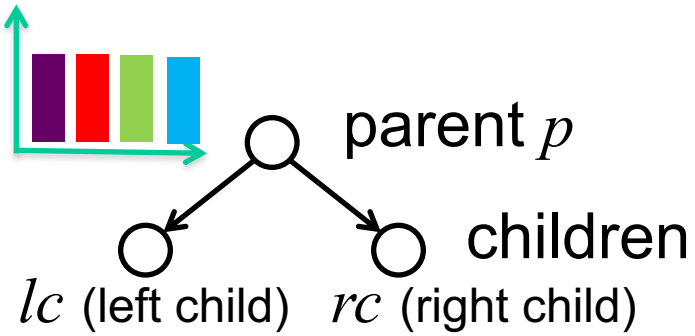


Having arrived at a node, one randomly generates a set of possible tests to carry out in order to split it up.

Among the possibilities one selects the test that maximally reduces the uncertainty.

Random Forests: the node tests

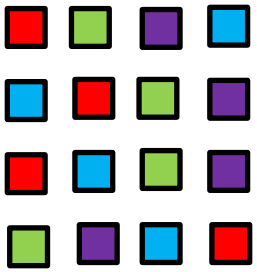


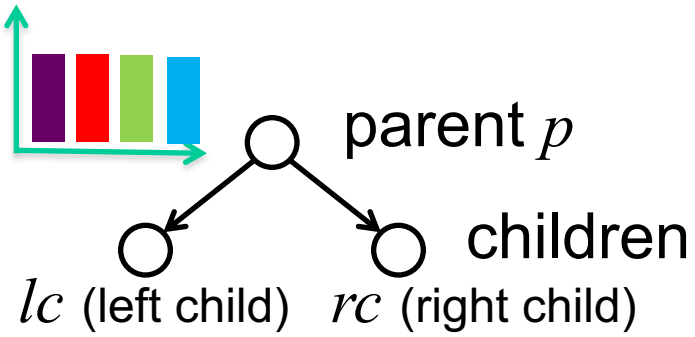


$$
\begin{aligned}
D_n &= \{(\vec{v}_n, c_n) \in \text{node n}\} \quad \text{(sets of training samples)} \\
G(k,\tau) &= H(\text{parent}) - \frac{|D_{rc}|}{|D_p|} H(\text{rc}) - \frac{|D_{lc}|}{|D_p|} H(\text{lc}) \\
H(\text{node n}) &= \sum_{i=1}^{K} P(w_i) \log P(w_i) \\
P(w_i) &= \sum_{n=1}^{|D_n|} \delta(c_n = w_i)/|D_n| \\
k^*, \tau^* &= \arg\max G(k,\tau)
\end{aligned}
$$

G quantifies the gain in class certainty, by going from a parent to its children

Random Forests: the node tests





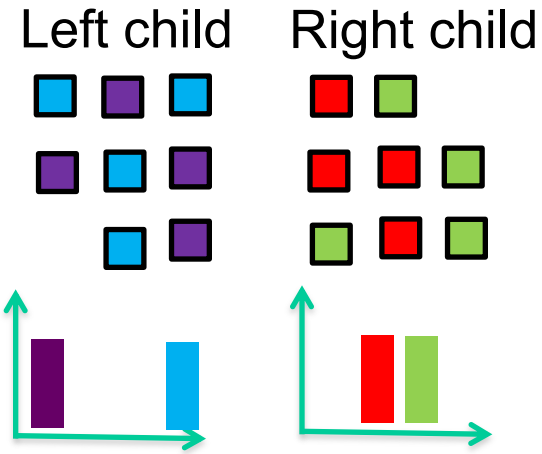
$$D_n = \{(\vec{v}_n, c_n) \in \text{node n}\} \quad \text{(sets of training samples)}$$

$$G(k,\tau) = H(\text{parent}) - \frac{|D_{rc}|}{|D_p|}H(\text{rc}) - \frac{|D_{lc}|}{|D_p|}H(\text{lc})$$

$$H(\text{node n}) = \sum_{i=1}^{K} P(w_i)\log P(w_i)$$

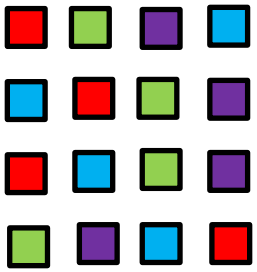
$$P(w_i) = \sum_{n=1}^{|D_n|} \delta(c_n = w_i)/|D_n|$$

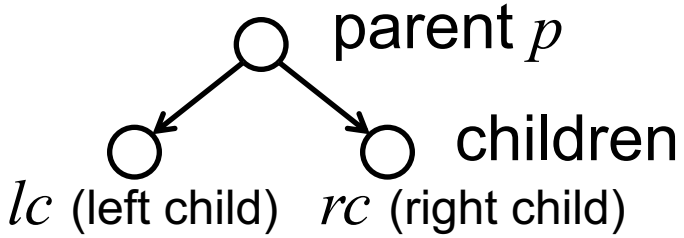
$$k^*, \tau^* = \arg\max G(k,\tau)$$



In this case, samples in the children nodes belong to 1 of only 2 classes, compared to possibly 4 at parent level

Random Forests: the node tests

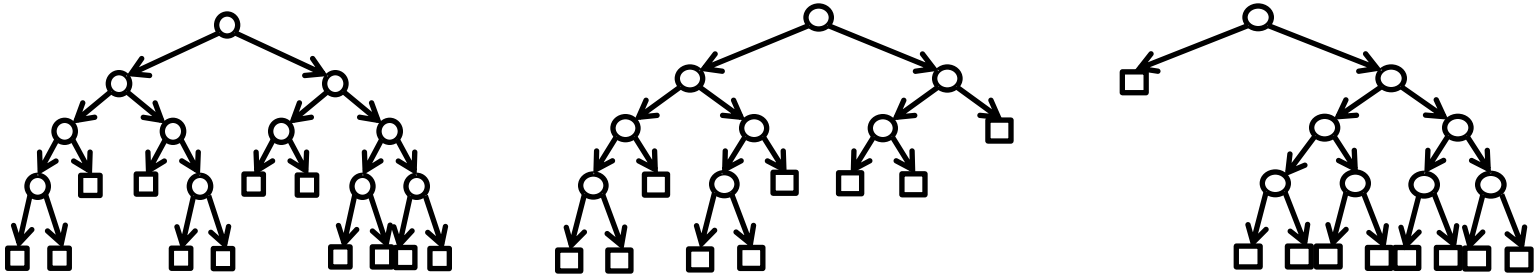




$$
\begin{aligned}
D_n &= \{(\vec{v}_n, c_n) \in \text{node n}\} \quad \text{(sets of training samples)} \\
G(k,\tau) &= H(\text{parent}) - \frac{|D_{rc}|}{|D_p|} H(\text{rc}) - \frac{|D_{lc}|}{|D_p|} H(\text{lc}) \\
H(\text{node n}) &= \sum_{i=1}^{K} P(w_i) \log P(w_i) \\
P(w_i) &= \sum_{n=1}^{|D_n|} \delta(c_n = w_i)/|D_n| \\
k^*, \tau^* &= \arg\max G(k,\tau)
\end{aligned}
$$

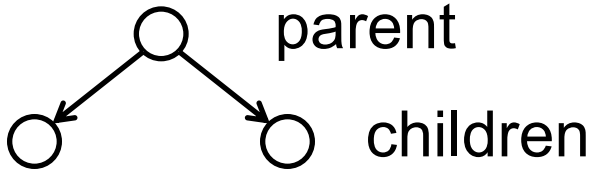
Select from a set of randomly generated tests the one that maximally reduces uncertainty about the classes found in lc & rc

The Randomness in Random Forests



- We saw that the node tests result from randomly generated sets of candidate tests.
- Another aspect of the randomness is that different trees can be trained with different, random subsets of the training samples
- The randomness is beneficial for the performance

Random Forests: stopping criterion



A child node is typically declared `leaf node' if

1) the samples in it all belong to 1 class/label only, or

2) the nmb of samples falls below a threshold such that their number is too small to extract meaningful statistics

Discriminative learning: Random Forest

T E S T I N G

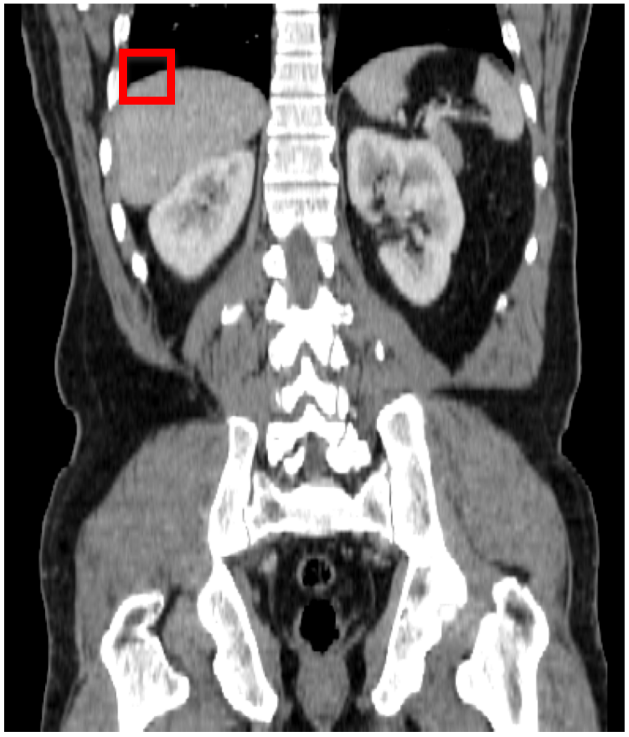
Once the trees – i.e. the entire forest – has been trained, we can apply those trees to the samples of images to be segmented automatically.

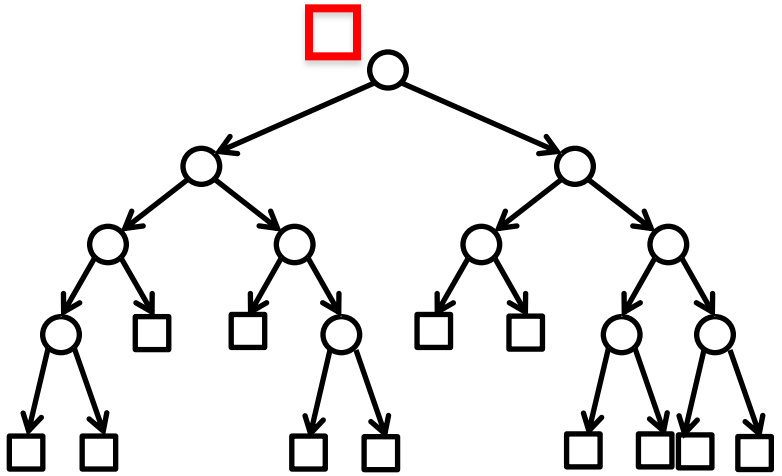
Random Forests: testing

- Feed the current sample into every tree
- See in which leaf node it ends -> (most probable) class
- Combine the results, e.g. which class/label was found most often

Random forests: testing (i.e. during segmentation)

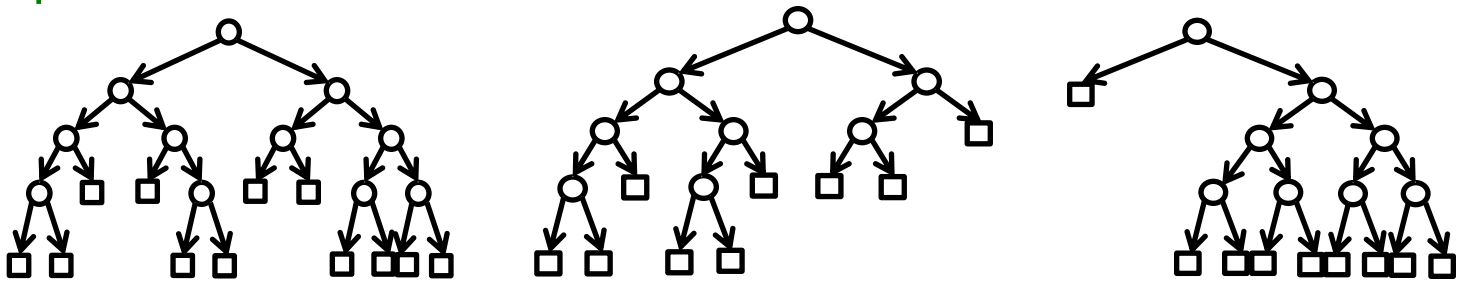
$= \vec{v} = \{v_1, \ldots, v_d\}$



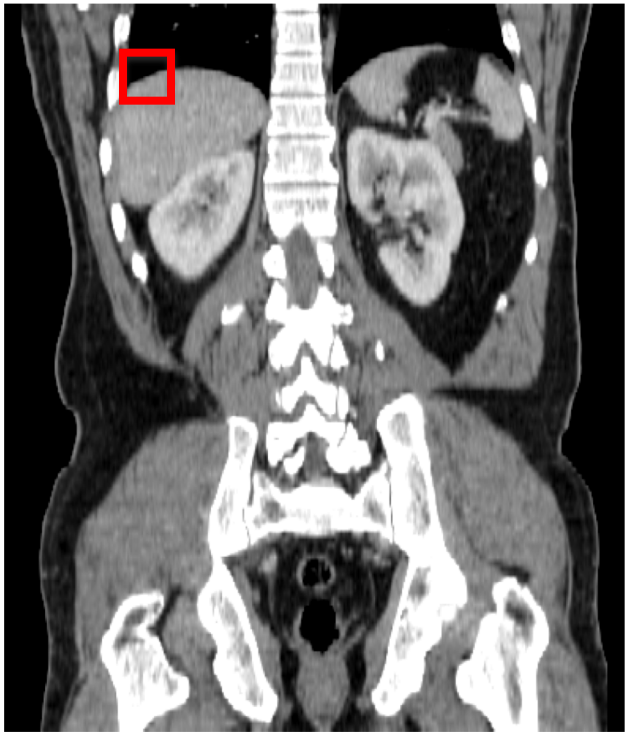


$$f_n(\vec{v}) = \begin{cases} 0 \rightarrow & \text{go left} \\ 1 \rightarrow & \text{go right} \end{cases} \quad f_l(\vec{v}) = c$$

Repeat this for all trees in the forest…

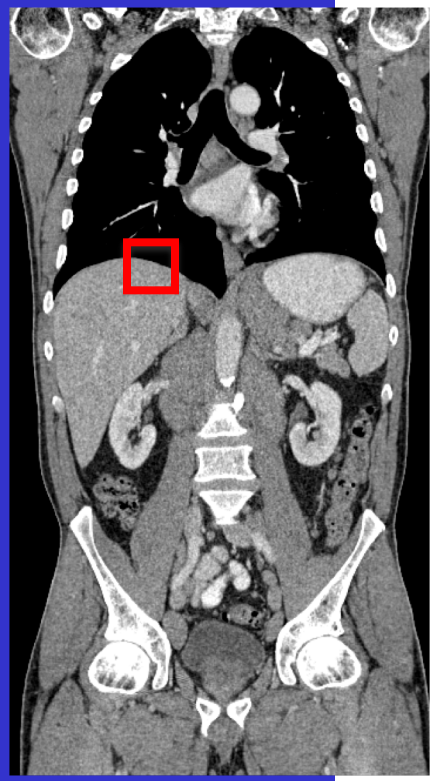


Random forests: During Segmentation

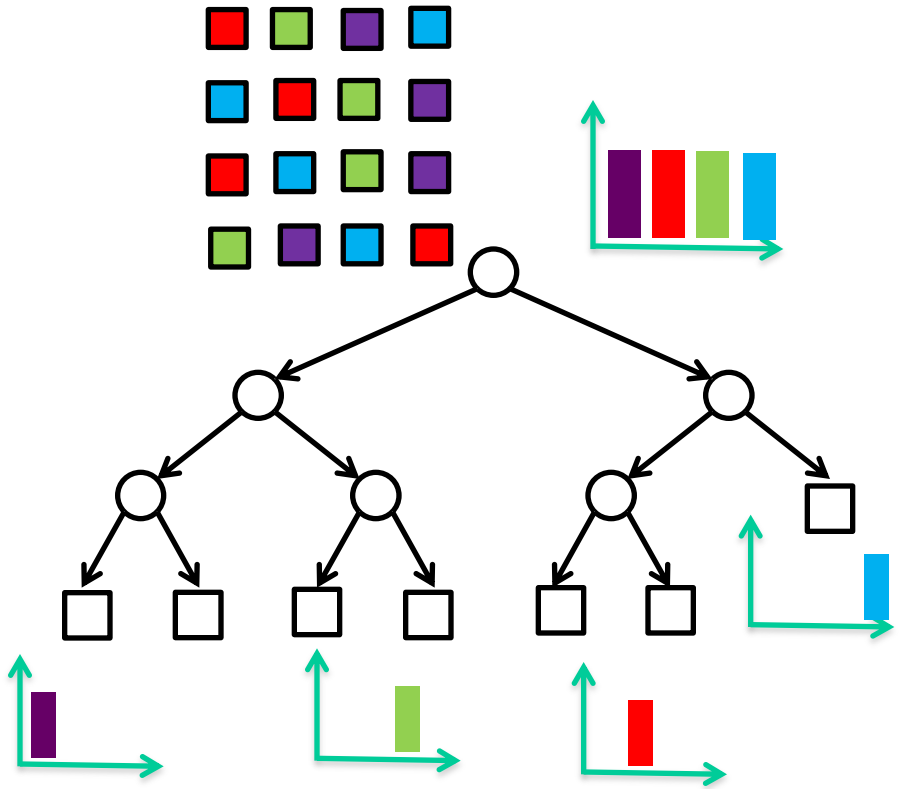


Carry out the process on the previous slide for all the patches involved in the segmentation

Random Forests: testing



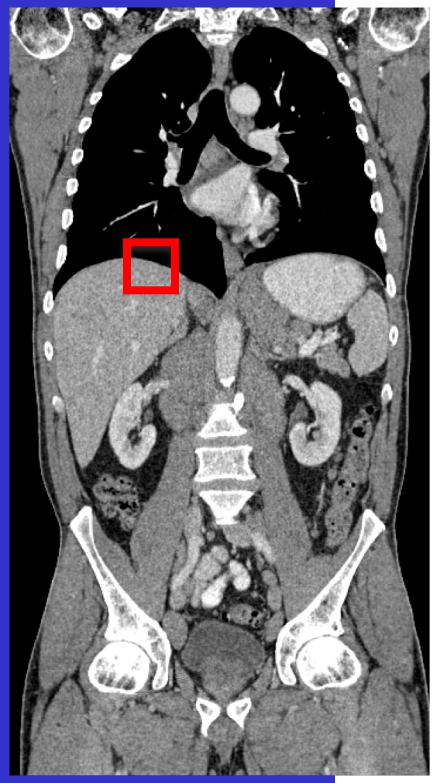
$$\blacksquare \rightarrow \vec{v}_1 = \{v_{11}, \ldots, v_{1d}\}$$
$$\blacksquare \rightarrow \vec{v}_2 = \{v_{21}, \ldots, v_{2d}\}$$
$$\blacksquare \rightarrow \vec{v}_n = \{v_{n1}, \ldots, v_{nd}\}$$



When using the system, a point's sample will end up in some leaf node of every tree => combine…

$$f_l(\vec{v}) = \mathbb{E}[c|\text{leaf l}]$$

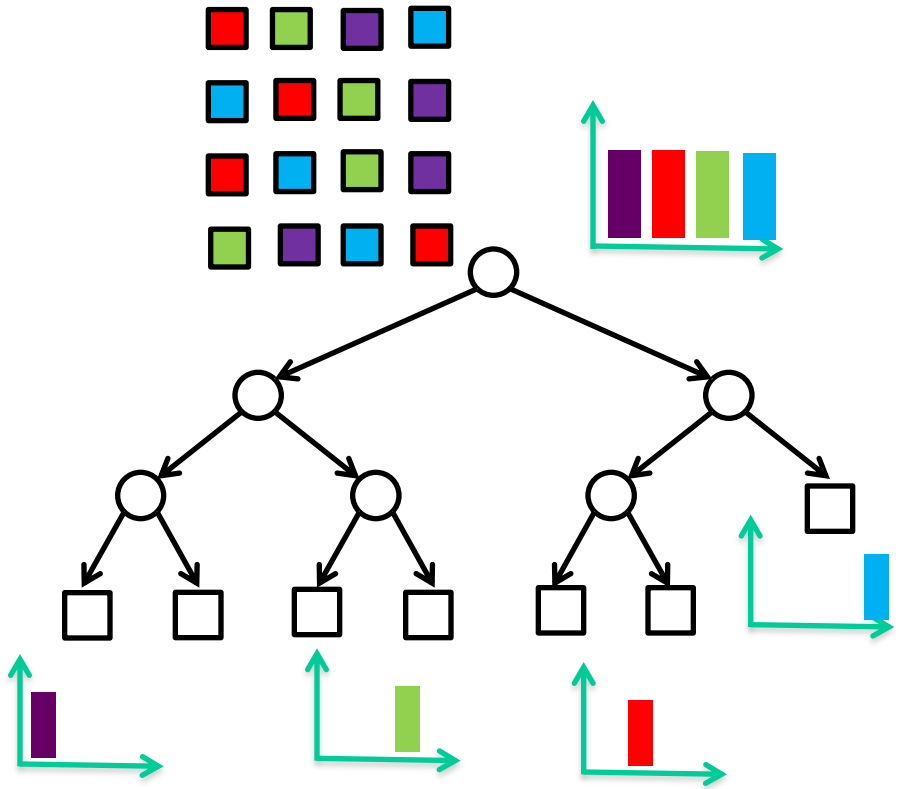
Random Forests: testing



$\rightarrow \vec{v}_1 = \{v_{11}, \ldots, v_{1d}\}$

$\rightarrow \vec{v}_2 = \{v_{21}, \ldots, v_{2d}\}$

$\rightarrow \vec{v}_n = \{v_{n1}, \ldots, v_{nd}\}$



E.g. select the class that most often appeared as most probable class in the visited leaf nodes.

$$f_l(\vec{v}) = \mathbb{E}[c|\text{leaf l}]$$

Random Forests: remarks

Pros

- Easy to implement
- Very efficient during testing
- Can easily use diverse features
- Can handle high dimensional spaces

Cons

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

PS Method behind Kinect body part classification (earlier version)