# Computer Vision – Lecture 7

## Segmentation as Energy Minimization

13.11.2014

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de

leibe@vision.rwth-aachen.de

# Announcements

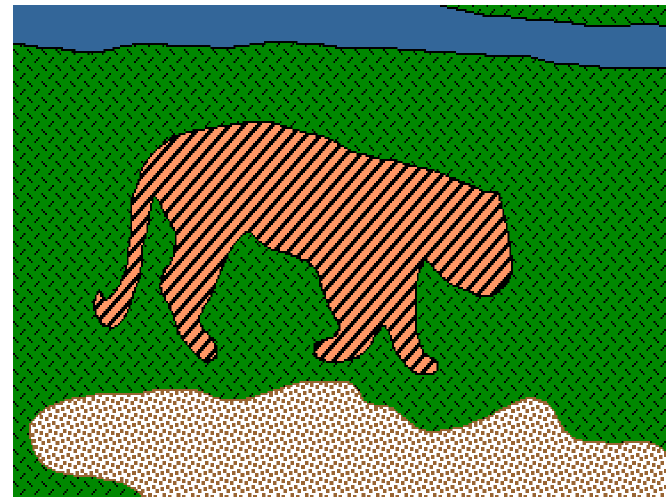
- **Please don’t forget to register for the exam!**
  - **On the Campus system**

# Course Outline

- Image Processing Basics
- Segmentation
  - Segmentation and Grouping
  - Segmentation as Energy Minimization
- Recognition
  - Global Representations
  - Subspace representations
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
- Motion and Tracking

# Recap: Image Segmentation

- Goal: identify groups of pixels that go together



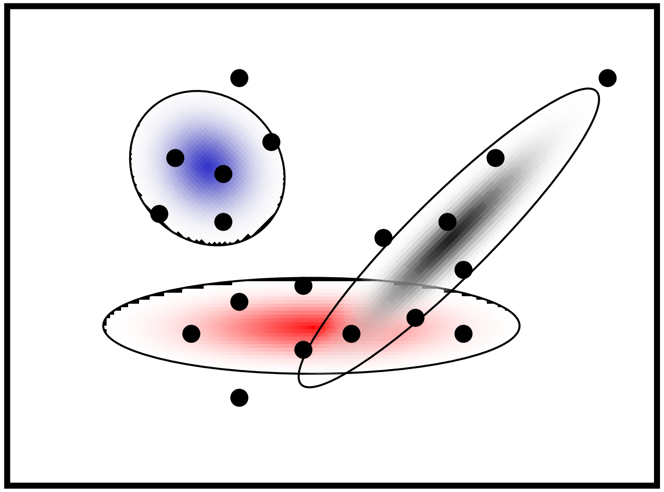
# Recap: K-Means Clustering

- **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:

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

# Recap: Expectation Maximization (EM)



- **Goal**
  - Find blob parameters $\theta$ that maximize the likelihood function:

$$p(data|\theta) = \prod_{n=1}^{N} p(\mathbf{x}_n|\theta)$$

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

see lecture Machine Learning!

# Recap: EM Algorithm

- **Expectation-Maximization (EM) Algorithm**
  - E-Step: softly assign samples to mixture components

$$\gamma_j(\mathbf{x}_n) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \qquad \forall j = 1,\ldots,K, \quad n = 1,\ldots,N$$

  - M-Step: re-estimate the parameters (separately for each mixture component) based on the soft assignments

$$\hat{N}_j \leftarrow \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)$$ = soft number of samples labeled $j$

$$\hat{\pi}_j^{\text{new}} \leftarrow \frac{\hat{N}_j}{N}$$

$$\hat{\boldsymbol{\mu}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)\mathbf{x}_n$$

$$\hat{\boldsymbol{\Sigma}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(\mathbf{x}_n)(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})^{\mathrm{T}}$$

Slide adapted from Bernt Schiele

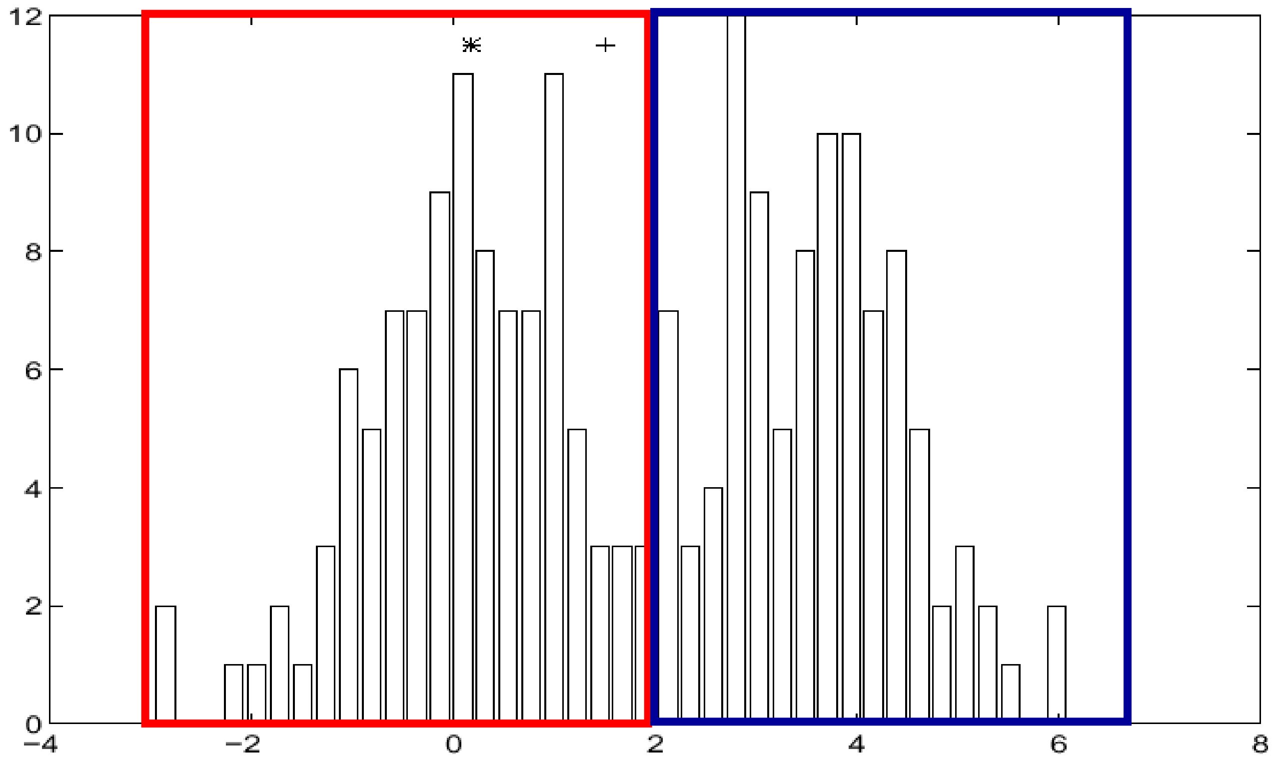
# MoG Color Models for Image Segmentation

(a) input image (b) user input (c) inferred segmentation

- **User assisted image segmentation**
  - User marks two regions for foreground and background.
  - Learn a MoG model for the color values in each region.
  - Use those models to classify all other pixels.

  $\Rightarrow$ Simple segmentation procedure
  (building block for more complex applications)

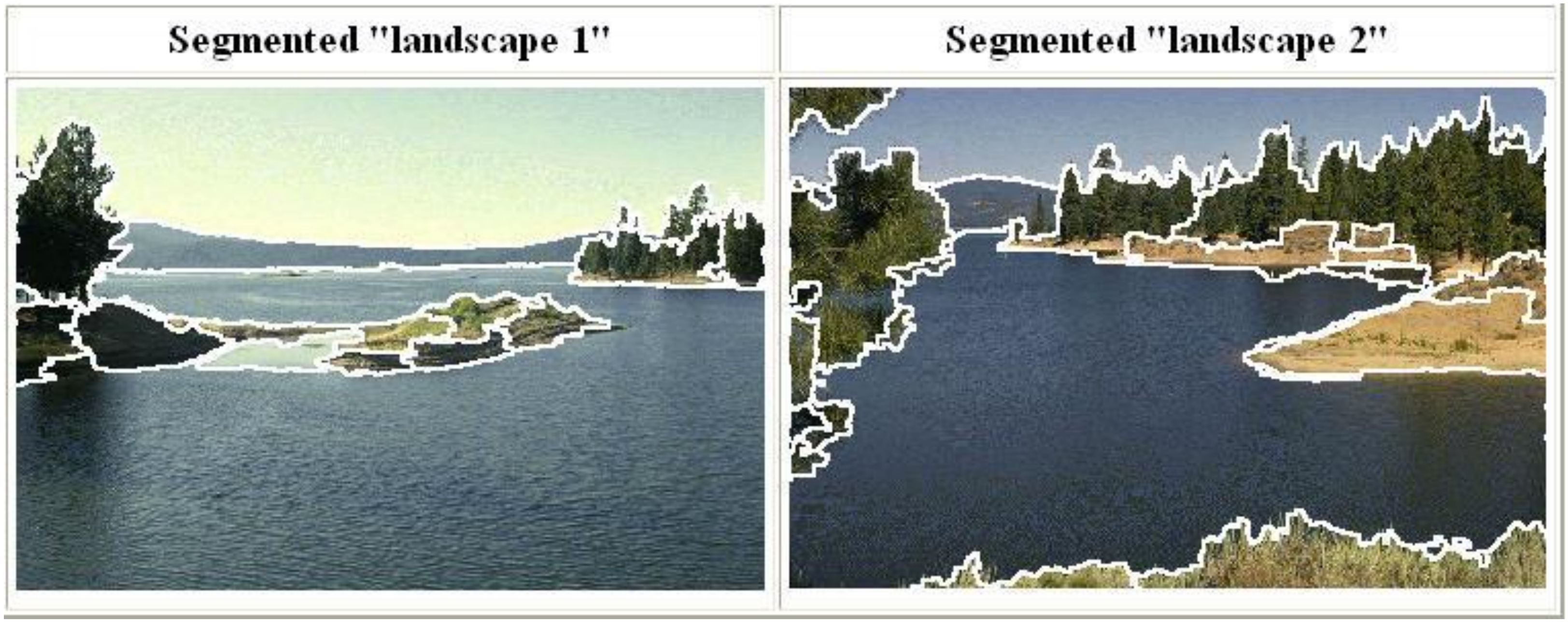
# Finding Modes in a Histogram




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

# Mean-Shift Segmentation

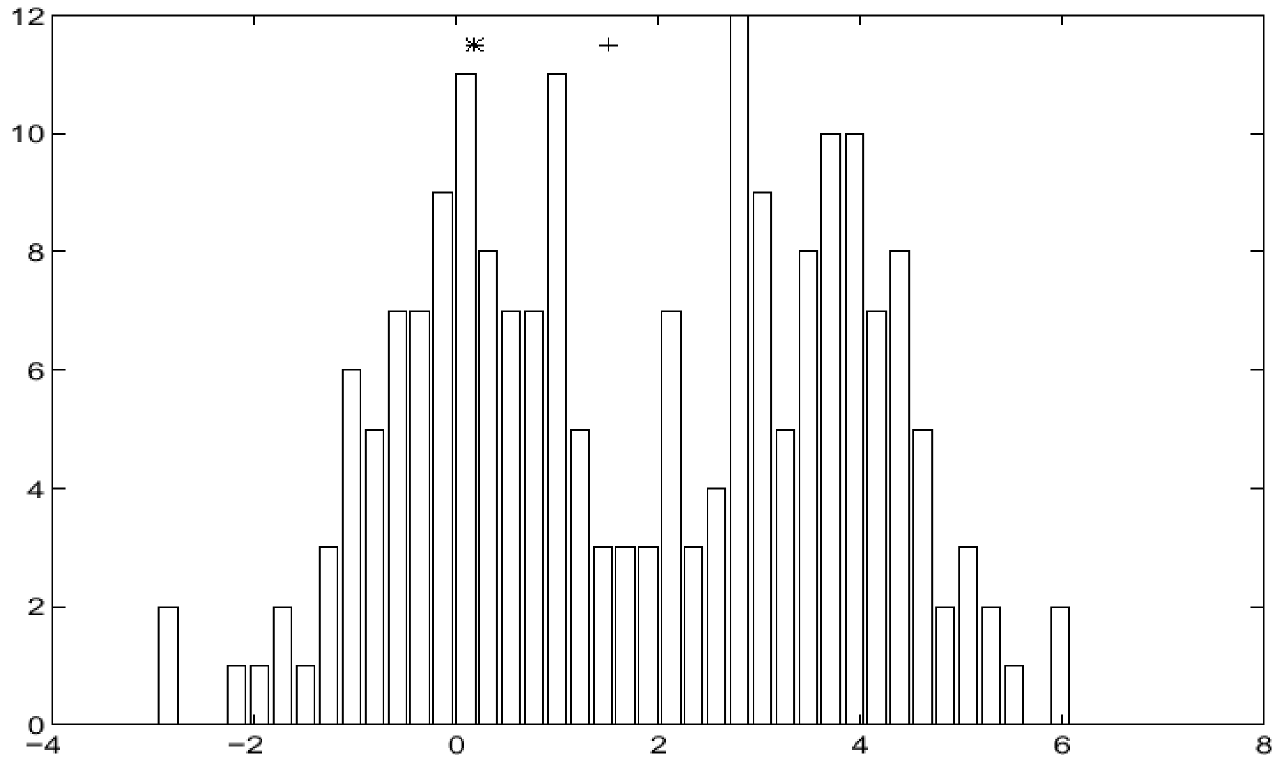
- An advanced and versatile technique for clustering-based segmentation




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

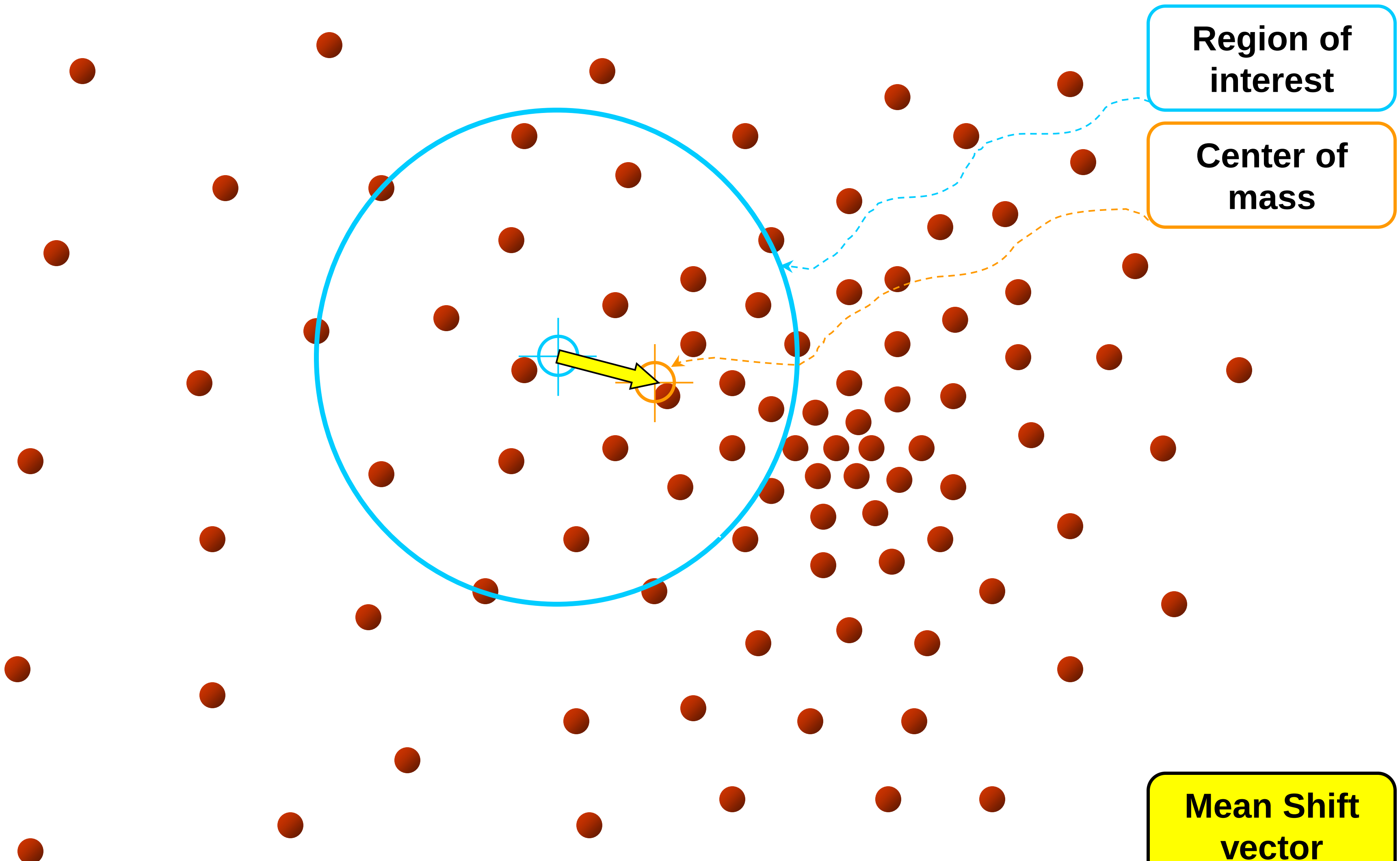
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

# Mean-Shift Algorithm




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

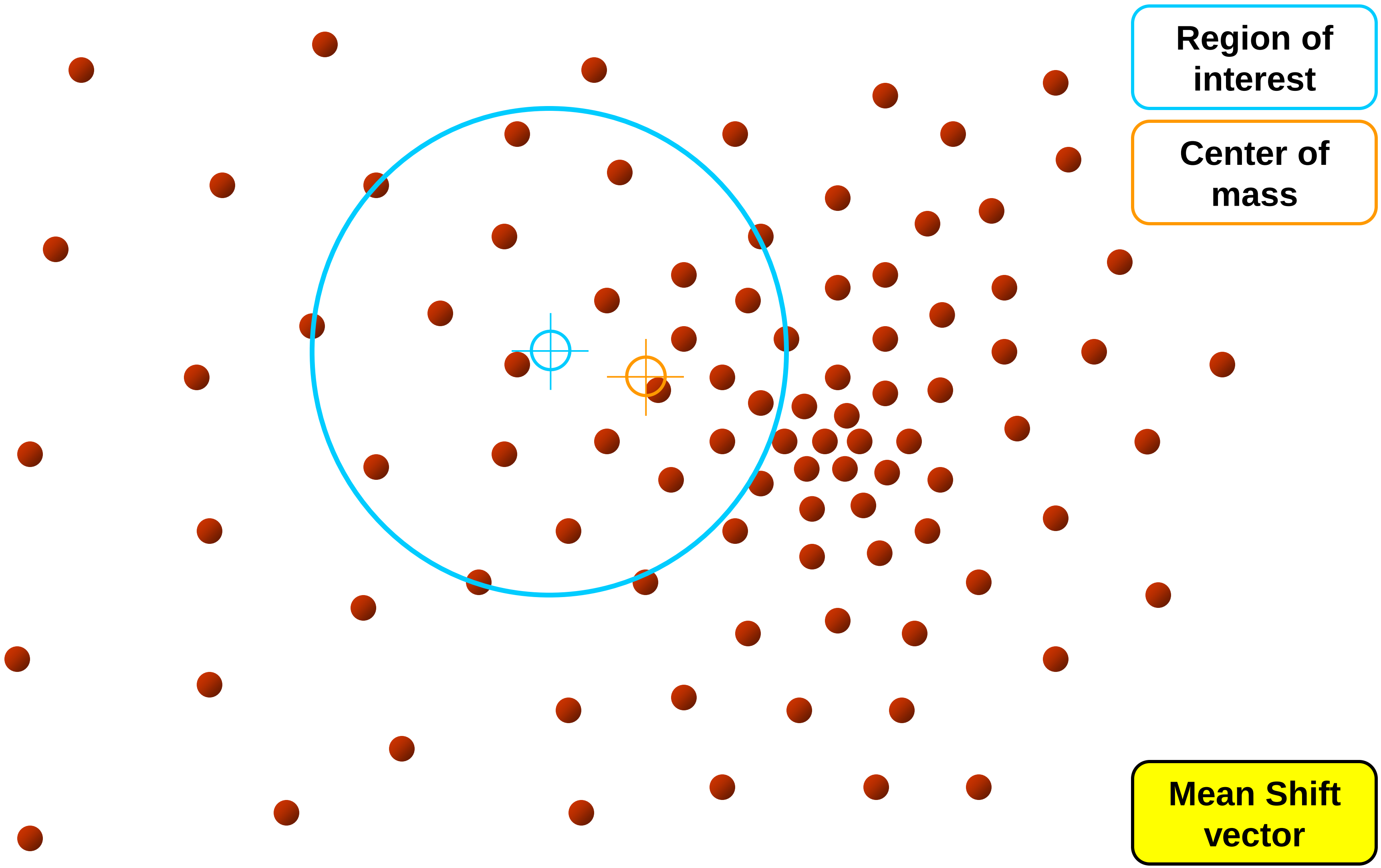
# Mean-Shift






Slide by Y. Ukrainitz & B. Sarel

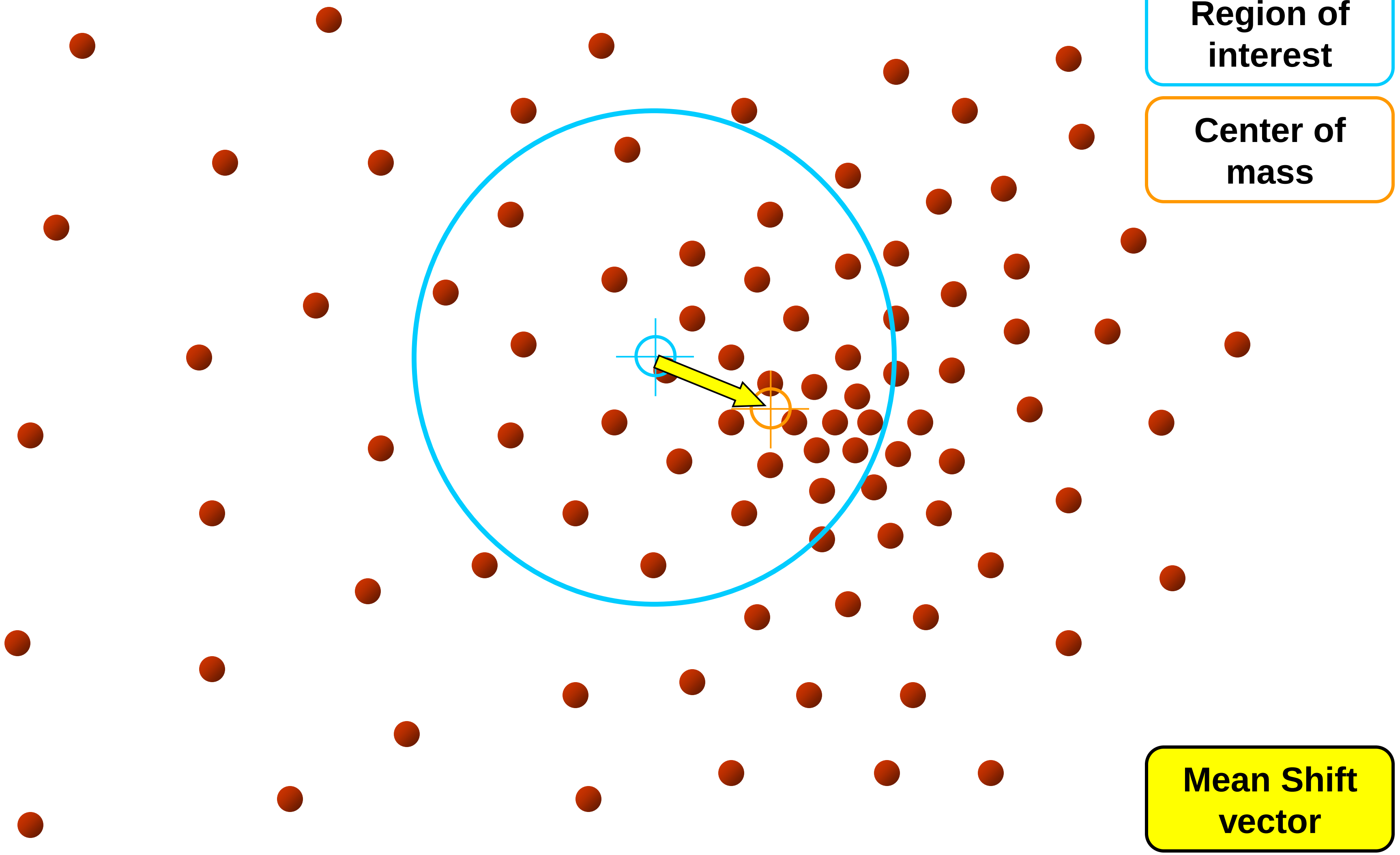
# Mean-Shift






Slide by Y. Ukrainitz & B. Sarel

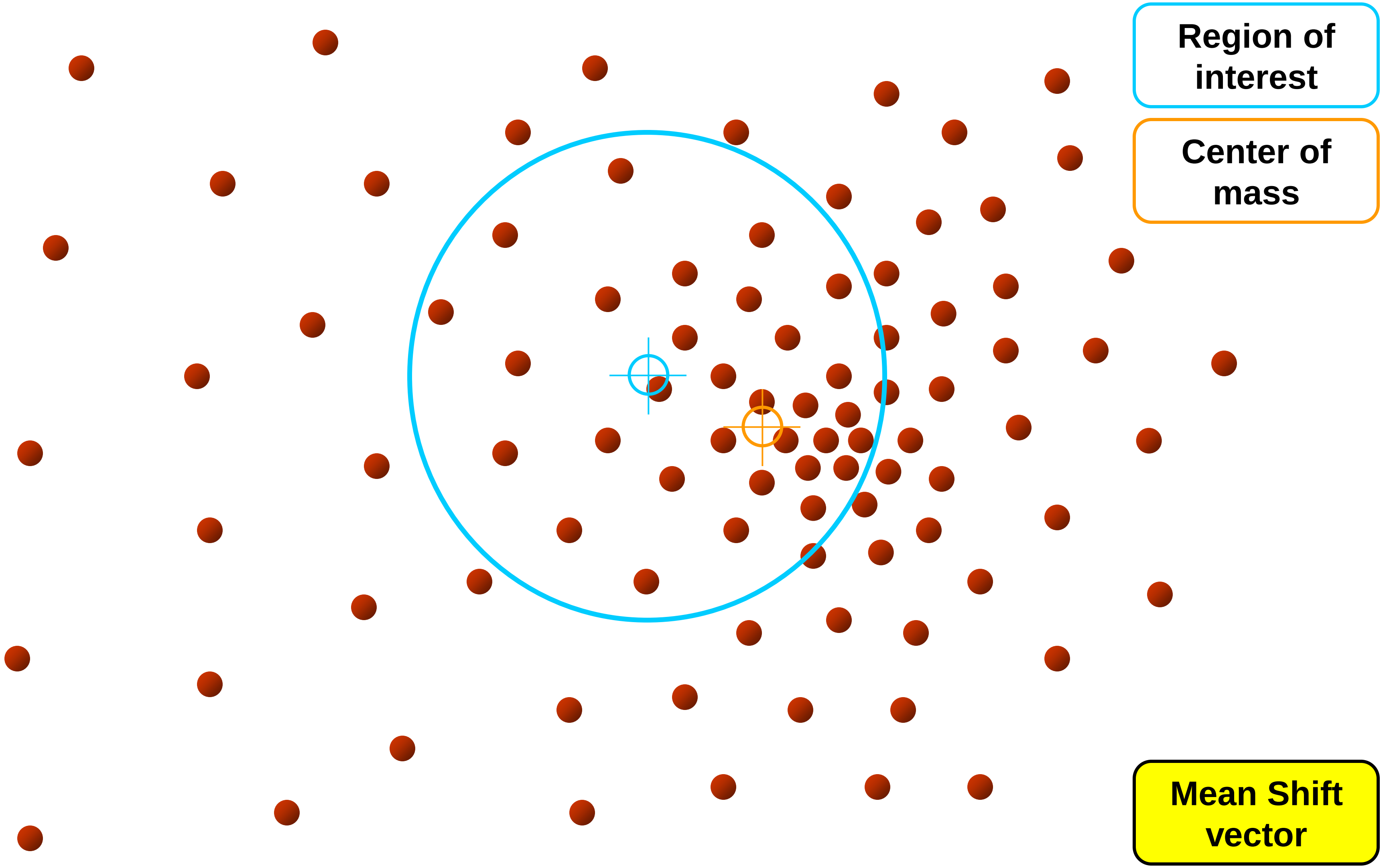
# Mean-Shift






Slide by Y. Ukrainitz & B. Sarel

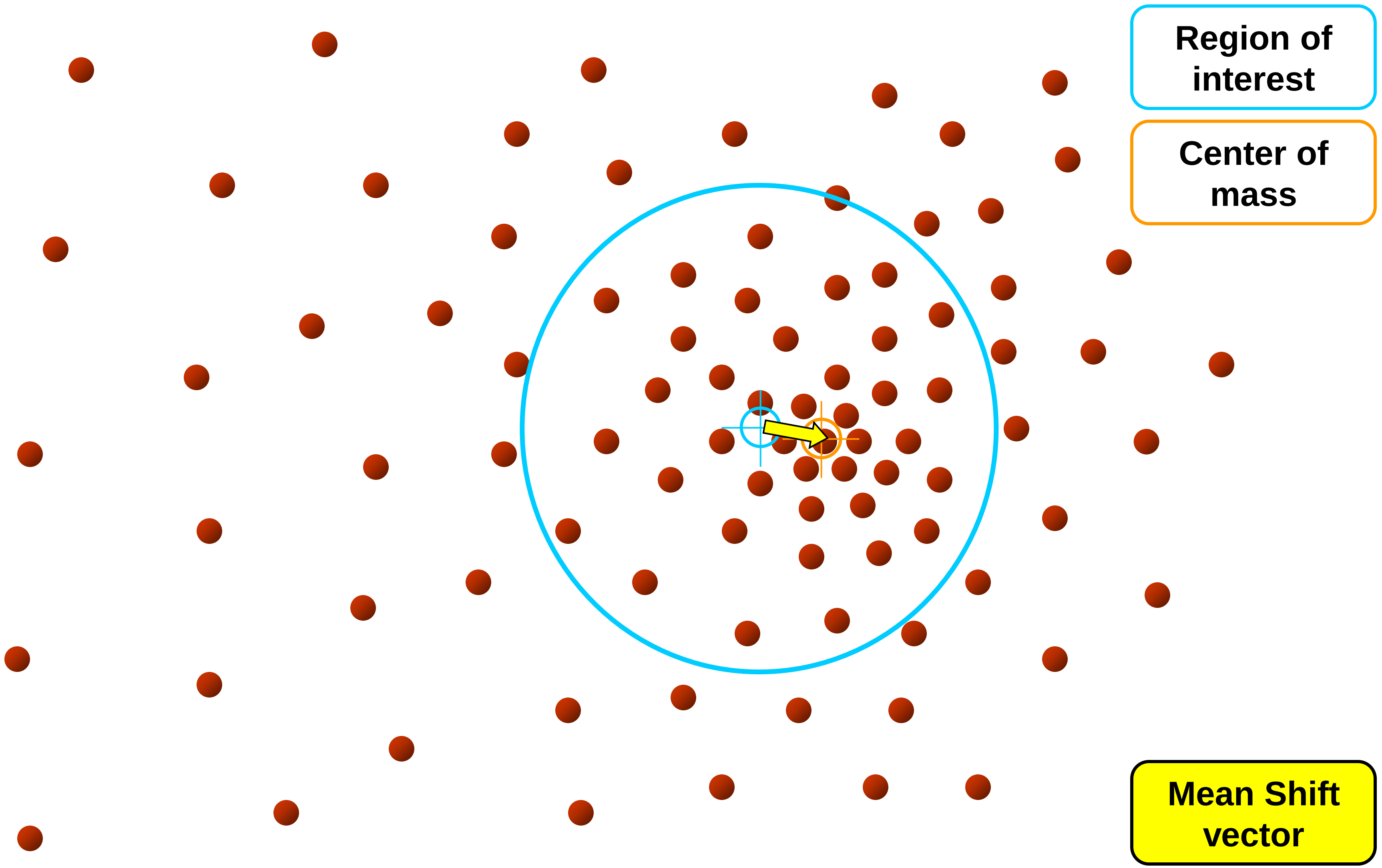
# Mean-Shift



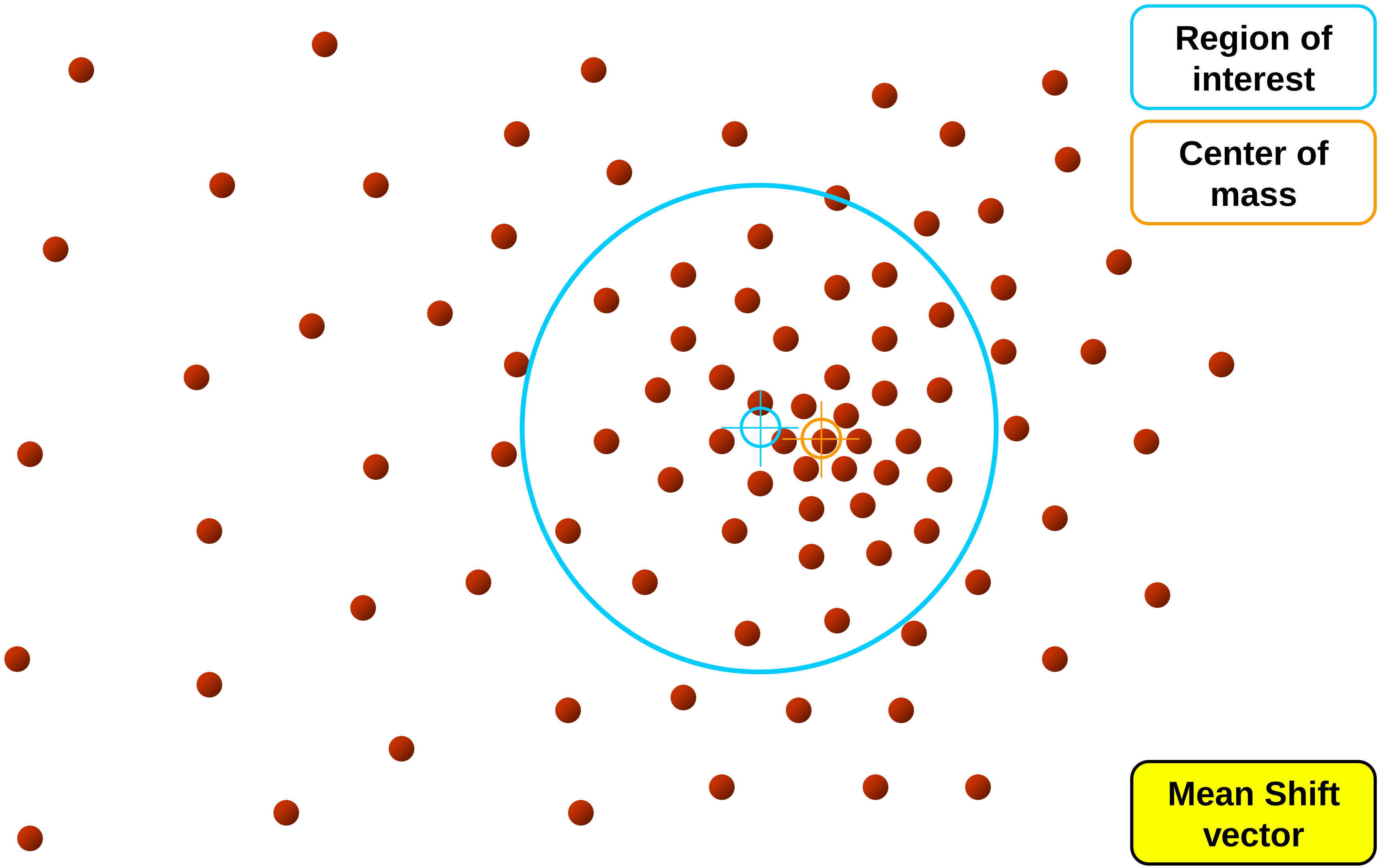



Slide by Y. Ukrainitz & B. Sarel

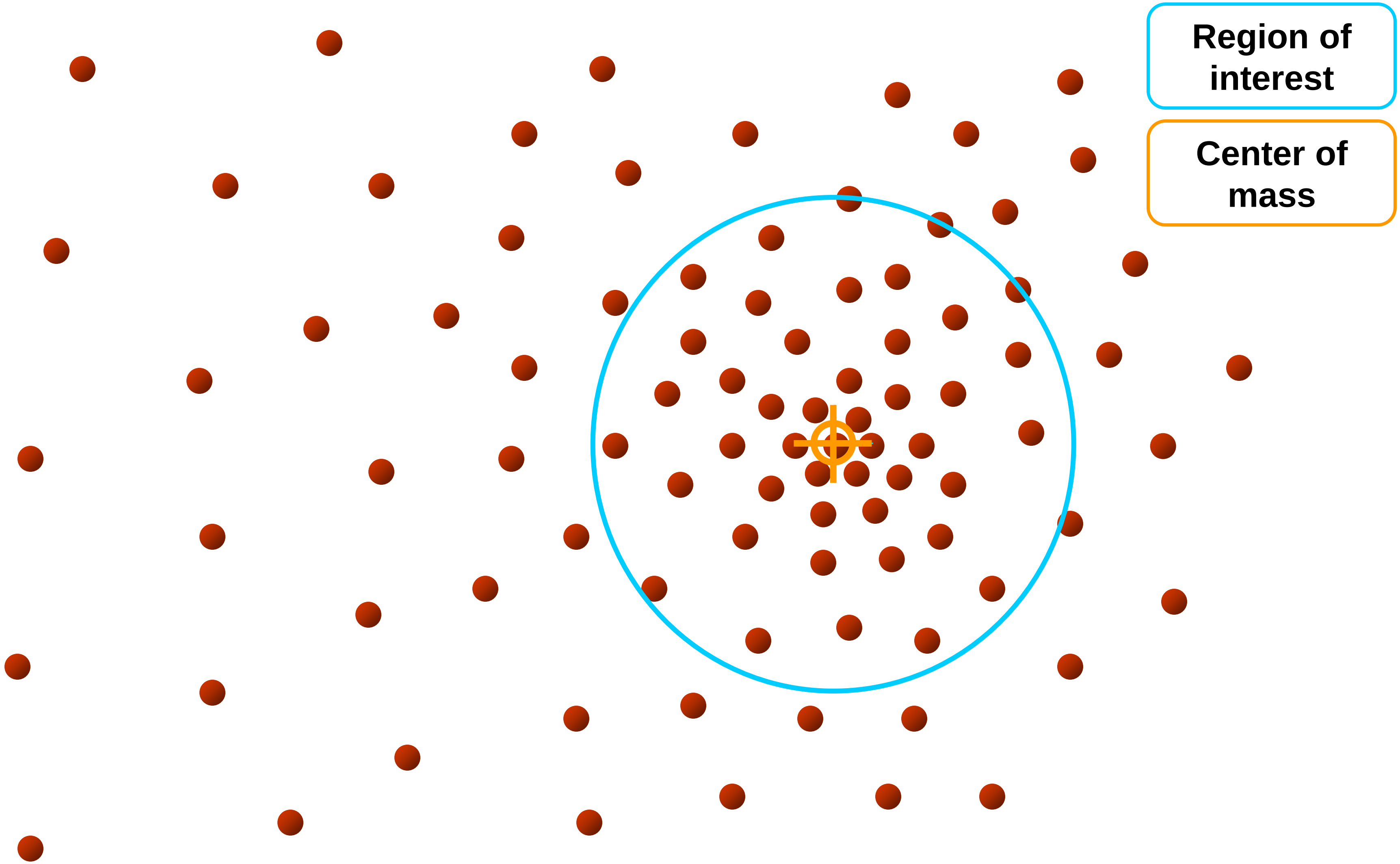
# Mean-Shift



# Mean-Shift



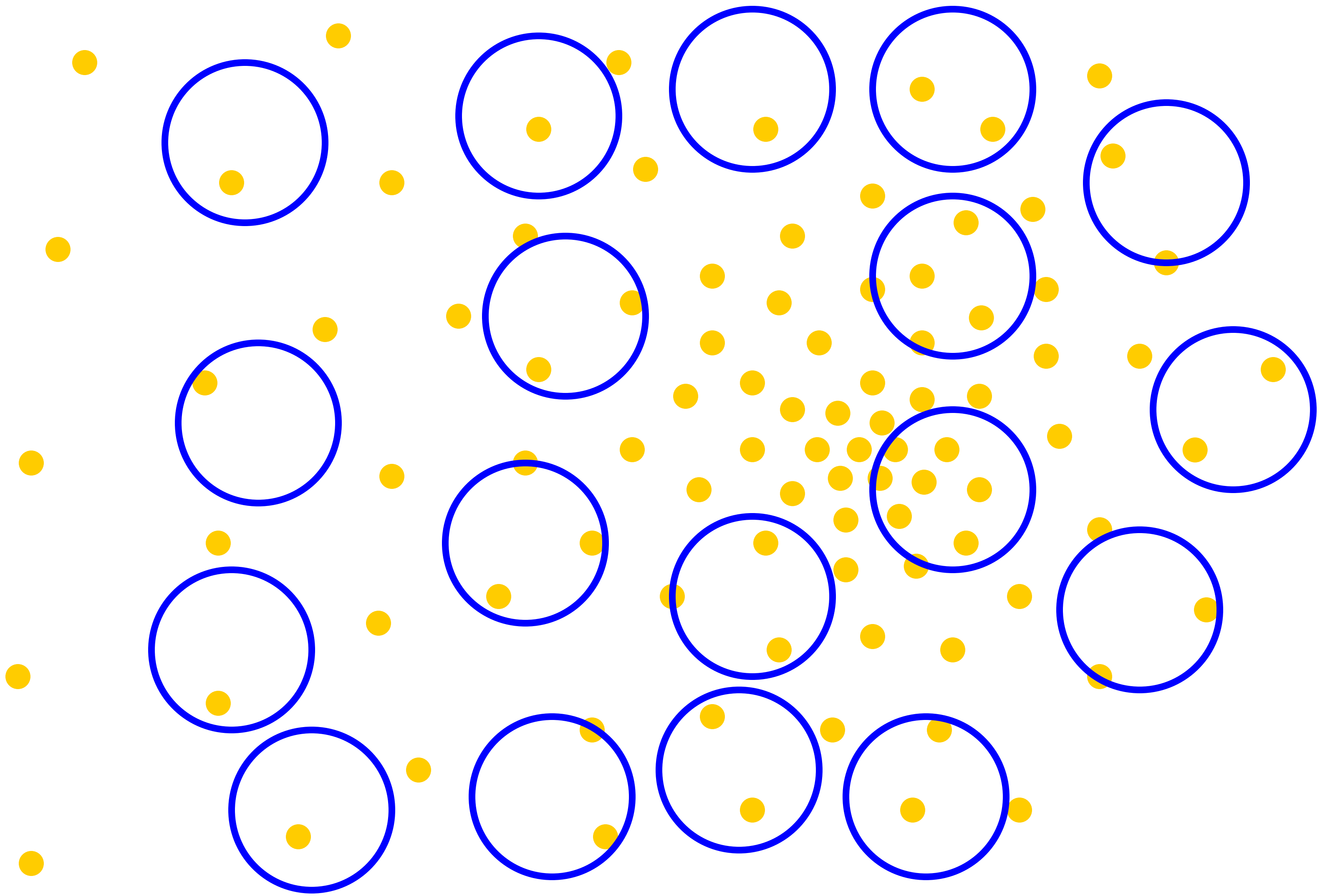
# Mean-Shift






Slide by Y. Ukrainitz & B. Sarel

# Real Modality Analysis

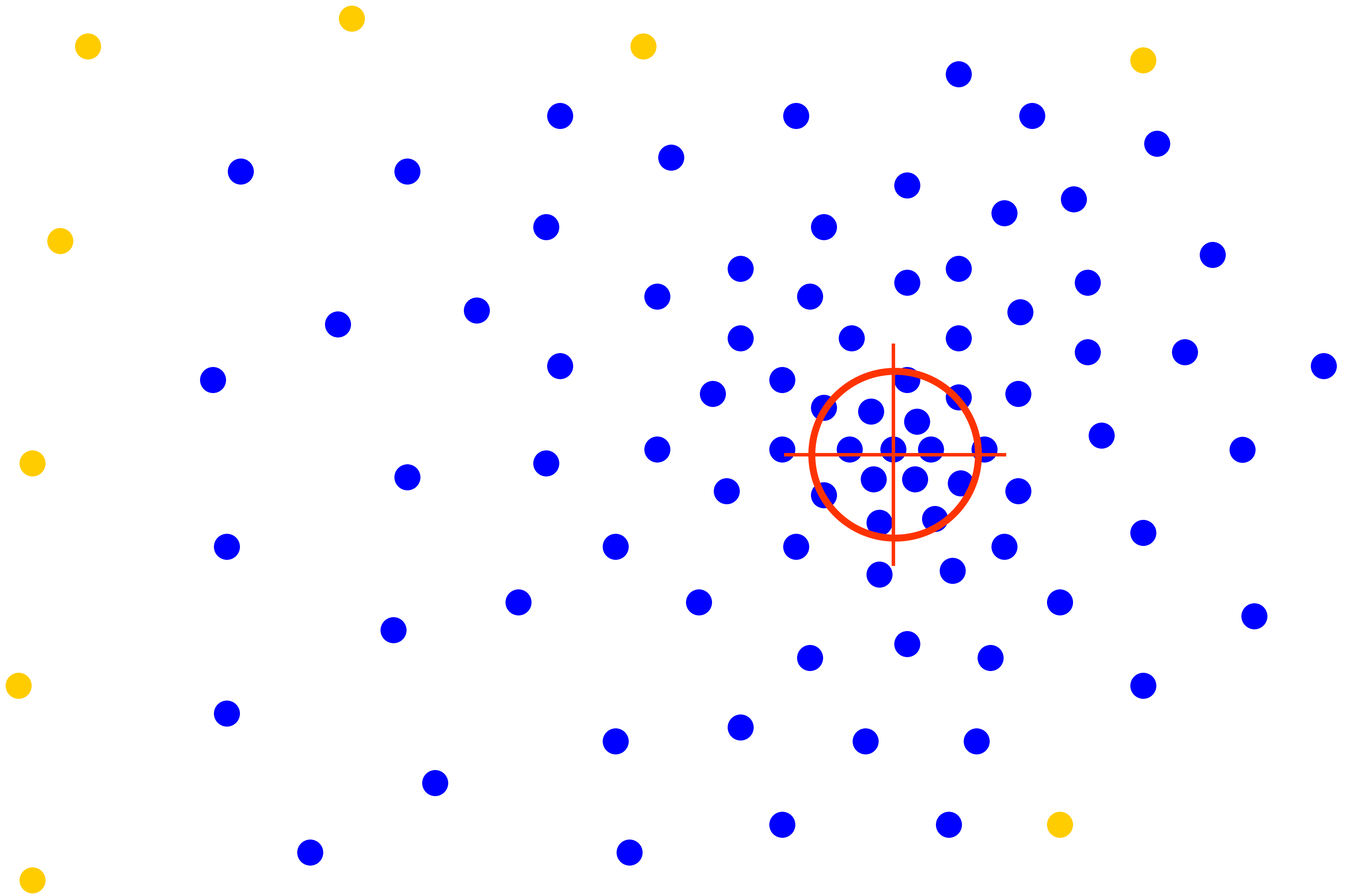


Tessellate the space with windows

Run the procedure in parallel

Slide by Y. Ukrainitz & B. Sarel

# Real Modality Analysis



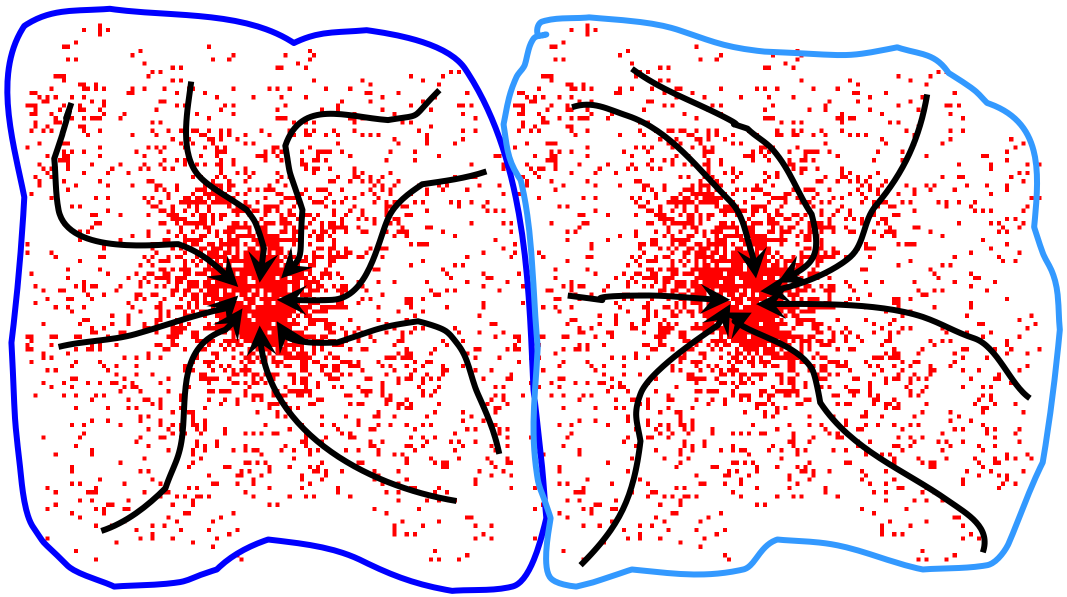
The blue data points were traversed by the windows towards the mode.



Slide by Y. Ukrainitz & B. Sarel

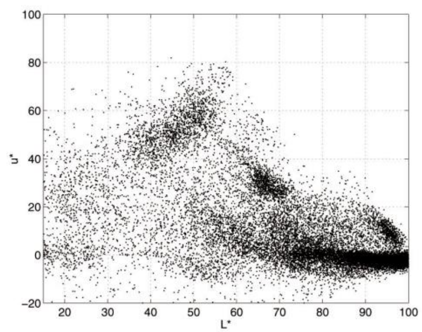
# Mean-Shift Clustering

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

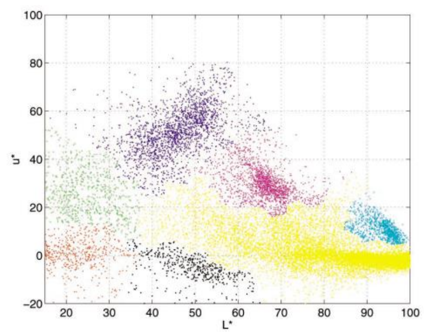


# Mean-Shift Clustering/Segmentation

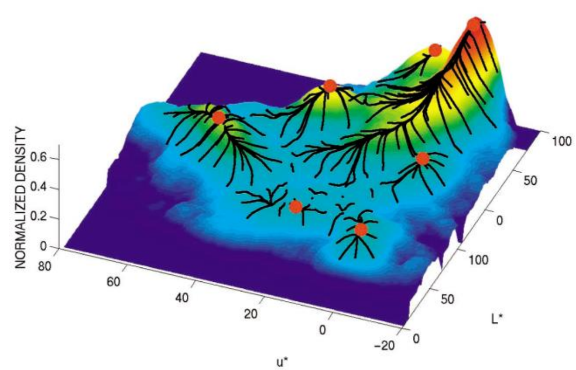
- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode



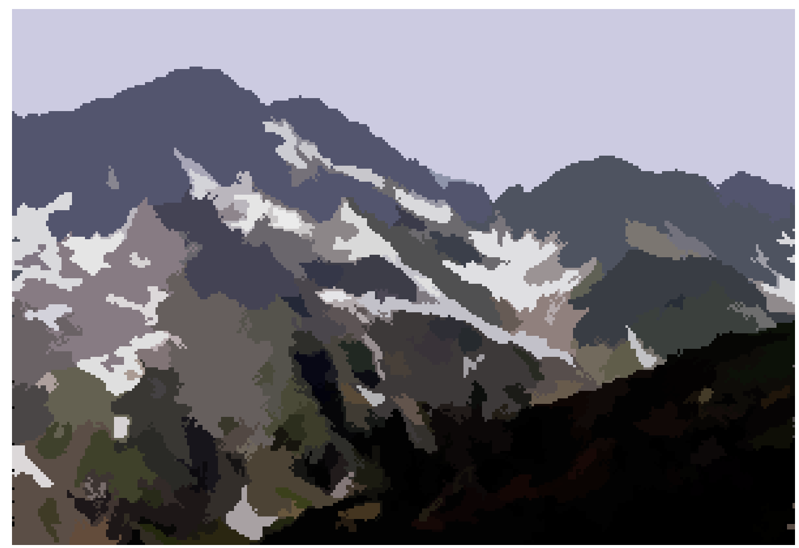

(a)




(b)



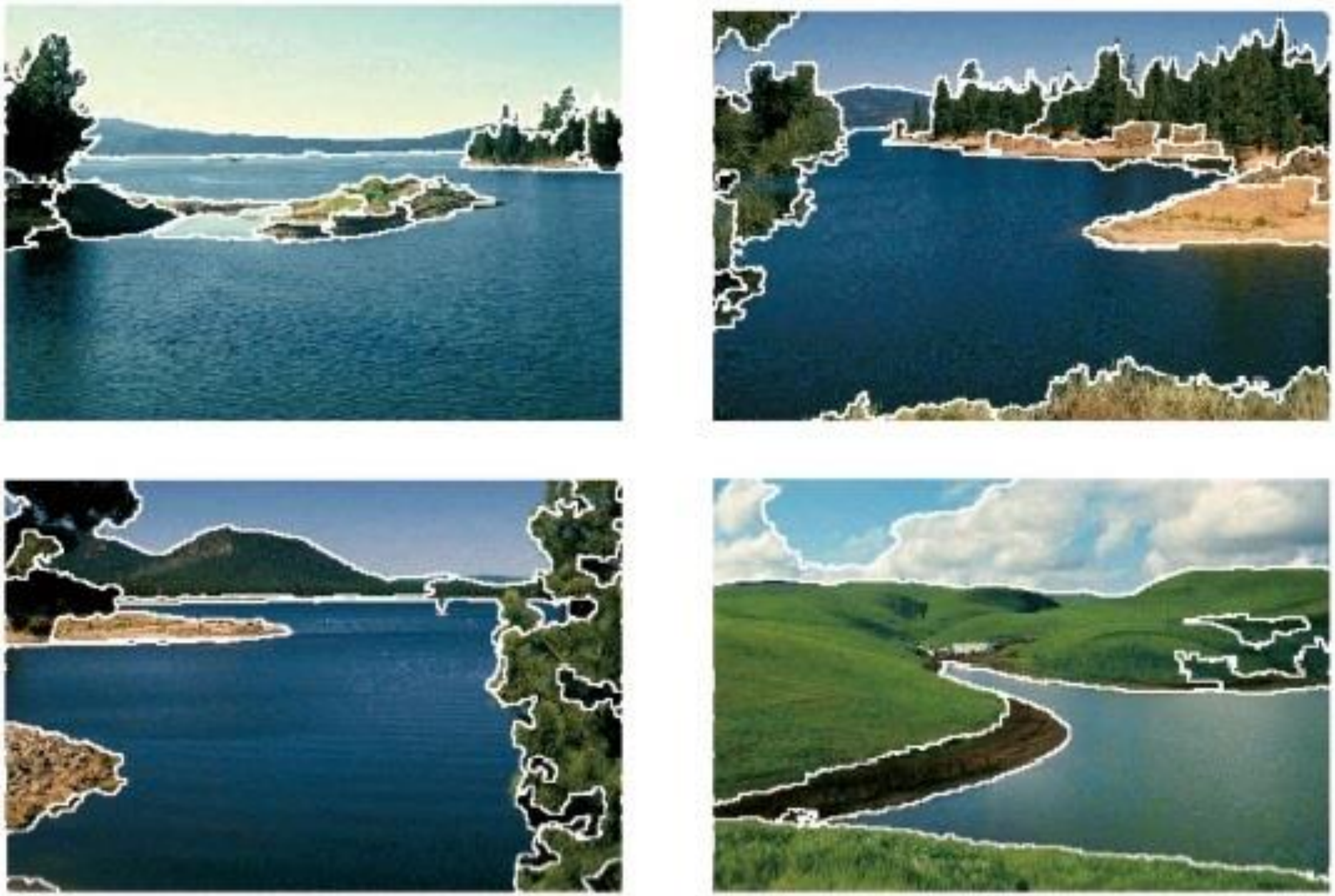
# Mean-Shift Segmentation Results



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

Slide credit: Svetlana Lazebnik

# More Results



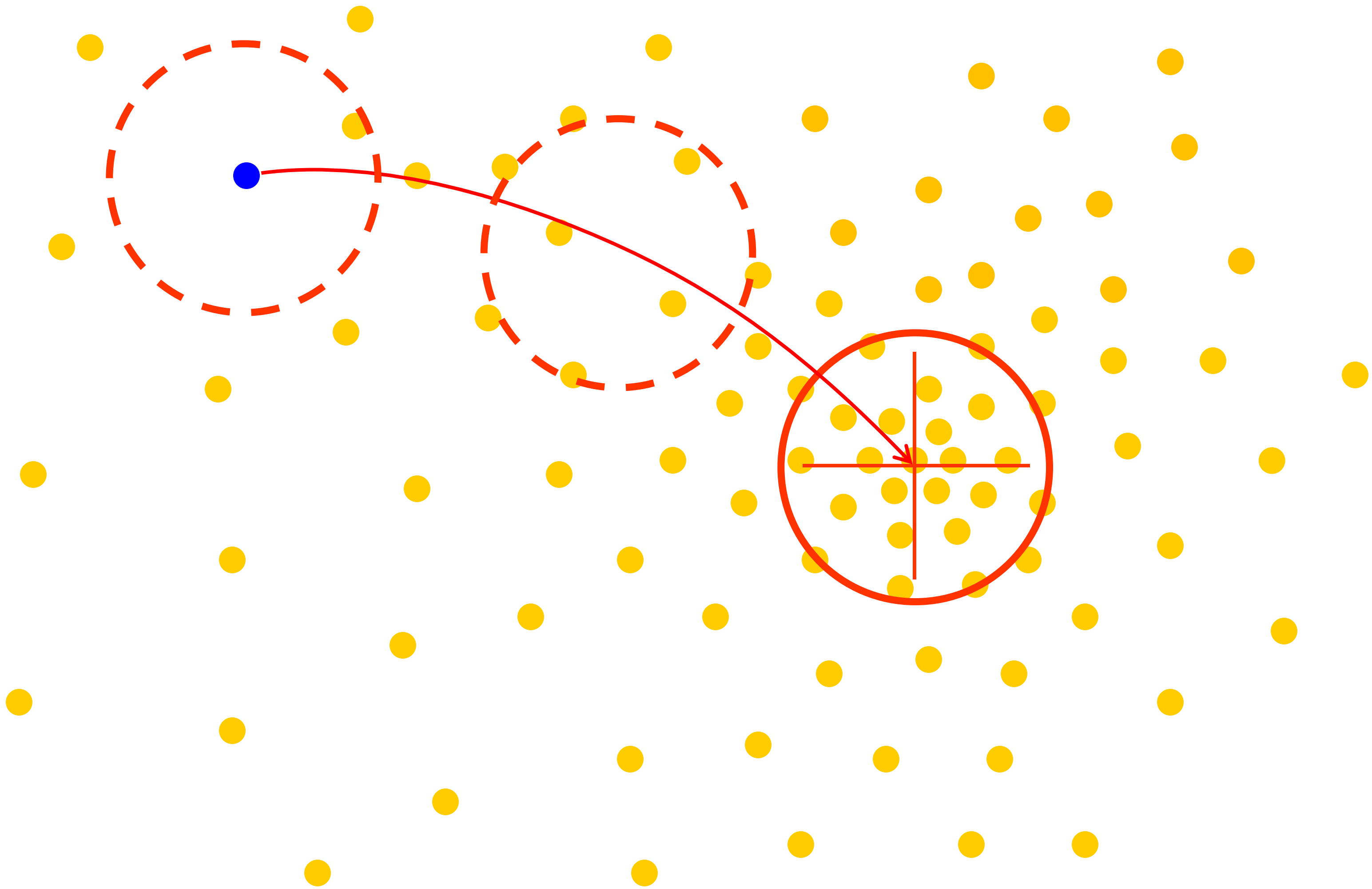
Slide credit: Svetlana Lazebnik

# More Results

Slide credit: Svetlana Lazebnik

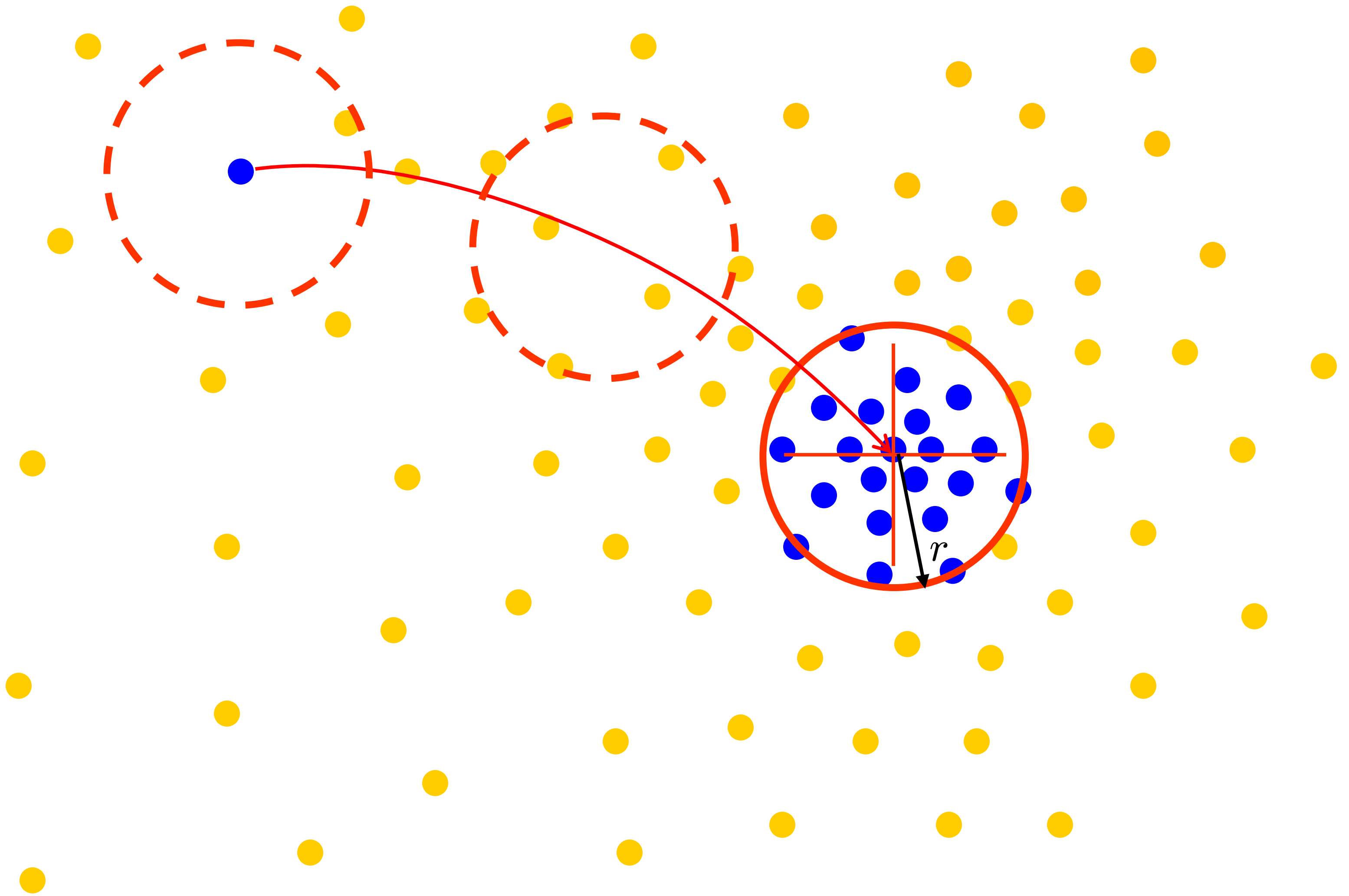
RWTH AACHEN UNIVERSITY

# Problem: Computational Complexity



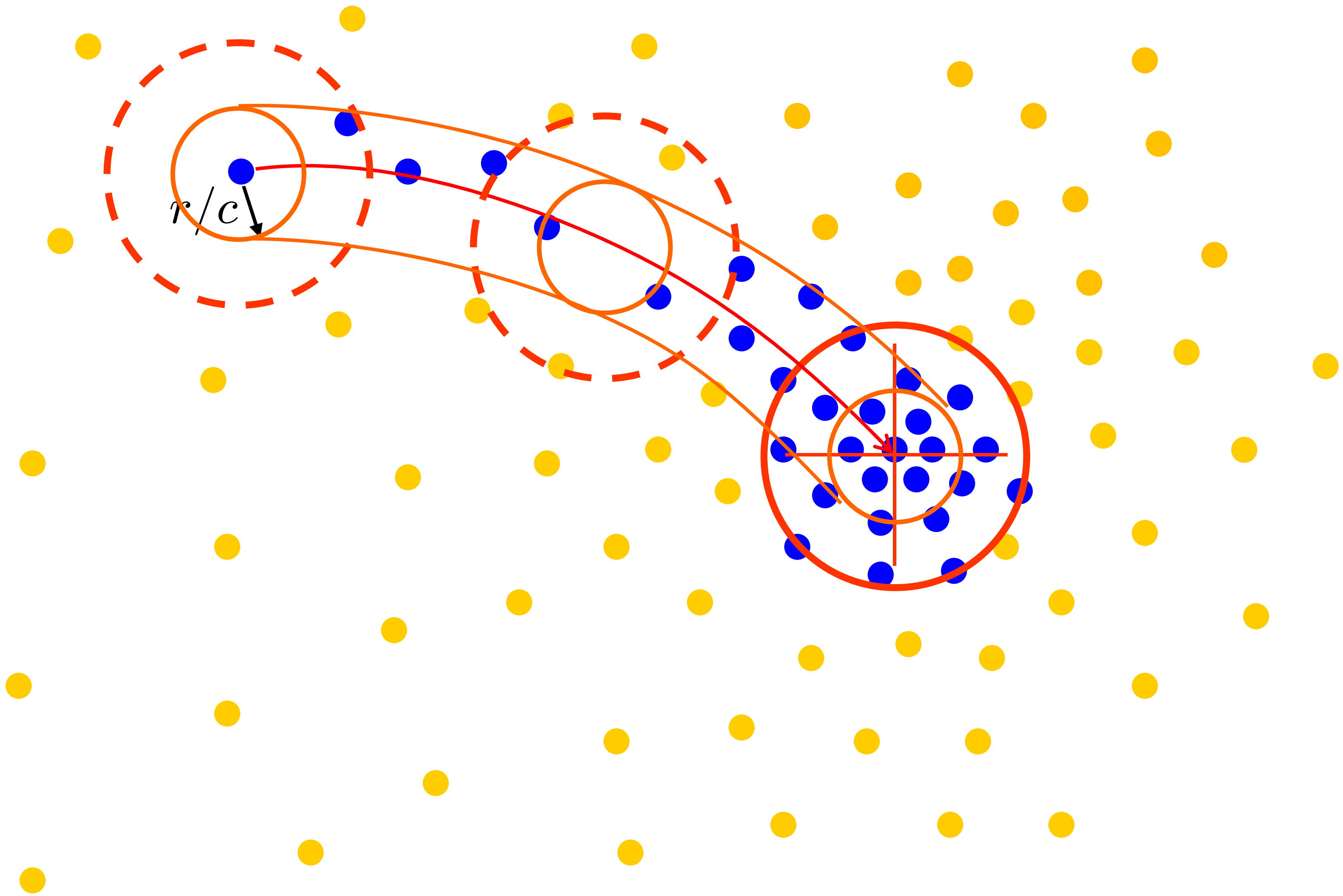
- Need to shift many windows…
- Many computations will be redundant.

# Speedups: Basin of Attraction




1. **Assign all points within radius r of end point to the mode.**

# Speedups




**2. Assign all points within radius r/c of the search path to the mode.**

# Summary Mean-Shift

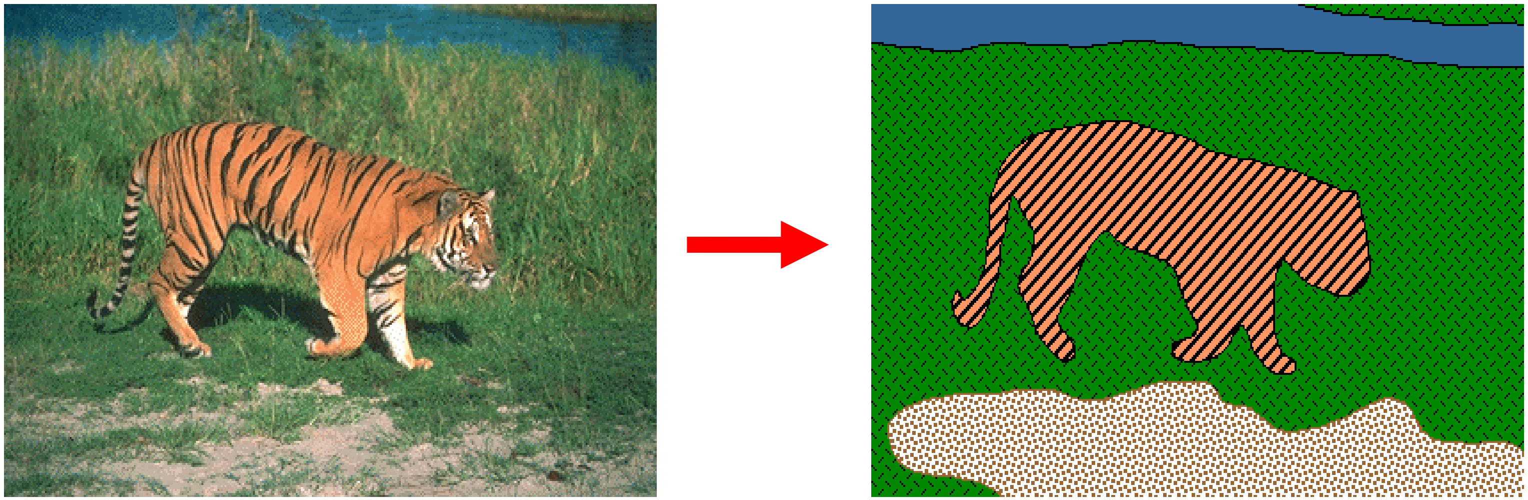
- Pros
  - General, application-independent tool
  - Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
  - Just a single parameter (window size h)
    - h has a physical meaning (unlike k-means)
  - Finds variable number of modes
  - Robust to outliers

- Cons
  - Output depends on window size
  - Window size (bandwidth) selection is not trivial
  - Computationally (relatively) expensive (~2s/image)
  - Does not scale well with dimension of feature space

Slide credit: Svetlana Lazebnik

# Back to the Image Segmentation Problem…

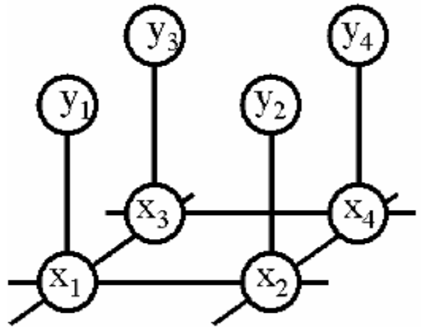
- Goal: identify groups of pixels that go together



- Up to now, we have focused on ways to group pixels into image segments based on their appearance…
  - Segmentation as clustering.
- We also want to enforce region constraints.
  - Spatial consistency
  - Smooth borders

# Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation

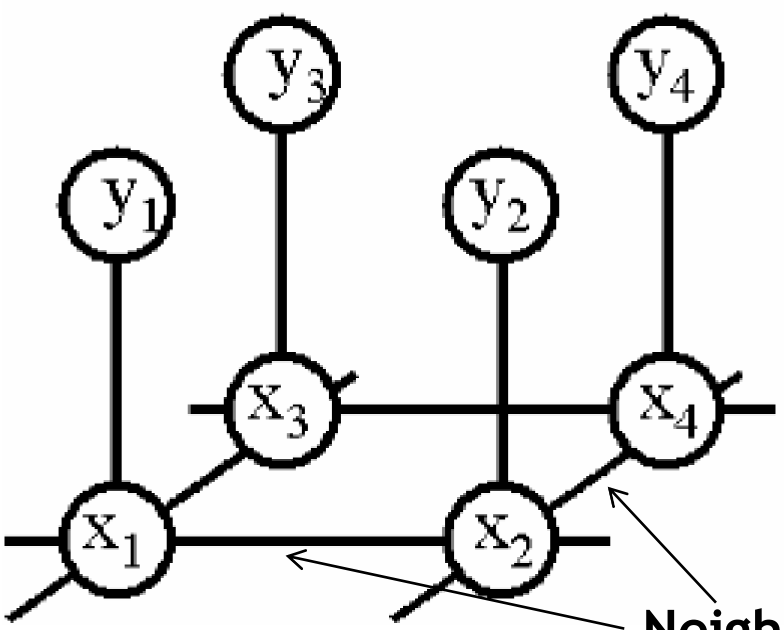



- Graph cuts for image segmentation
  - Basic idea
  - s-t Mincut algorithm
  - Extension to non-binary case

- Applications
  - Interactive segmentation

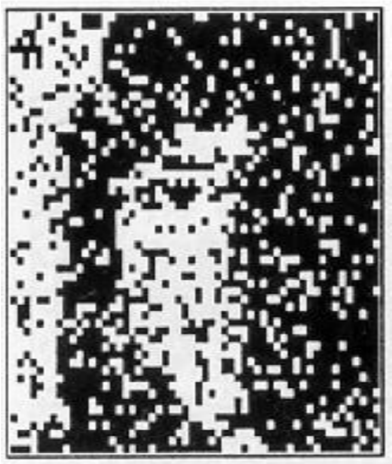
# Markov Random Fields

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



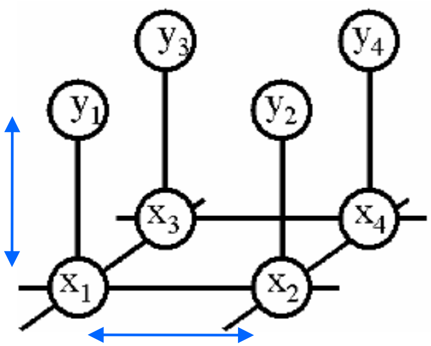
# MRF Nodes as Pixels

Original image



Degraded image

Reconstruction from MRF modeling pixel neighborhood statistics




$\Phi(x_i, y_i)$

$\Psi(x_i, x_j)$

# Network Joint Probability

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

Scene

Image

Image-scene compatibility function

Local observations

Scene-scene compatibility function

Neighboring scene nodes

# Energy Formulation

- **Joint probability**

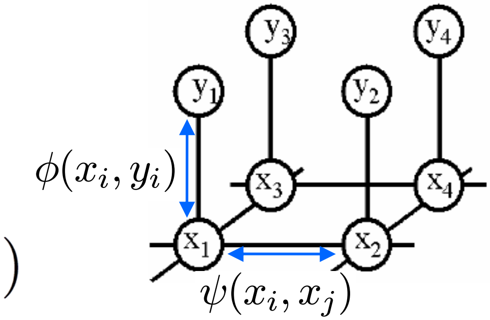
$$p(\mathbf{x},\mathbf{y}) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j)$$

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

$$\begin{aligned} -\log p(\mathbf{x},\mathbf{y}) &= -\sum_i \log \Phi(x_i, y_i) - \sum_{i,j} \log \Psi(x_i, x_j) \\ E(\mathbf{x},\mathbf{y}) &= \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j) \end{aligned}$$

- **This is similar to free-energy problems in statistical mechanics (spin glass theory). We therefore draw the analogy and call $E$ an *energy function*.**
- **$\phi$ and $\psi$ are called *potentials*.**

# Energy Formulation

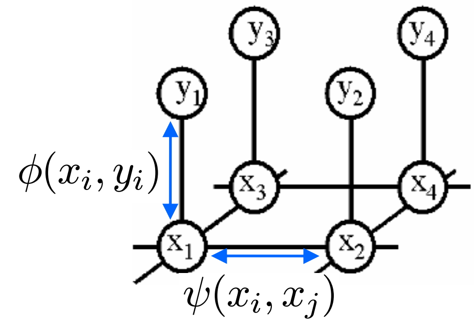



- Energy function

$$E(\mathbf{x}, \mathbf{y}) = \underbrace{\sum_i \phi(x_i, y_i)}_{\text{Single-node potentials}} + \underbrace{\sum_{i,j} \psi(x_i, x_j)}_{\text{Pairwise potentials}}$$

- Single-node potentials $\phi$ ("unary potentials")
  - Encode local information about the given pixel/patch
  - How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?
- Pairwise potentials $\psi$
  - Encode neighborhood information
  - How different is a pixel/patch's label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)

# Energy Minimization



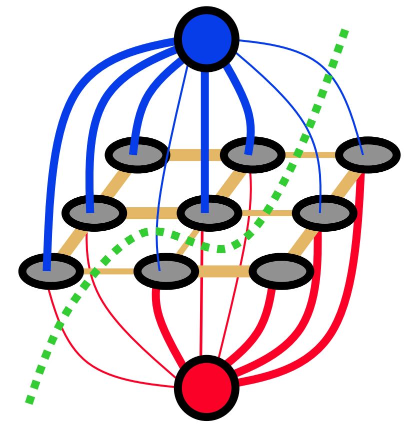

- **Goal:**
  - Infer the optimal labeling of the MRF.
- **Many inference algorithms are available, e.g.**
  - Gibbs sampling, simulated annealing
  - Iterated conditional modes (ICM)
  - Variational methods
  - Belief propagation
  - Graph cuts

see lecture Machine Learning!

- **Recently, Graph Cuts have become a popular tool**
  - Only suitable for a certain class of energy functions
  - But the solution can be obtained very fast for typical vision problems (~1MPixel/sec).

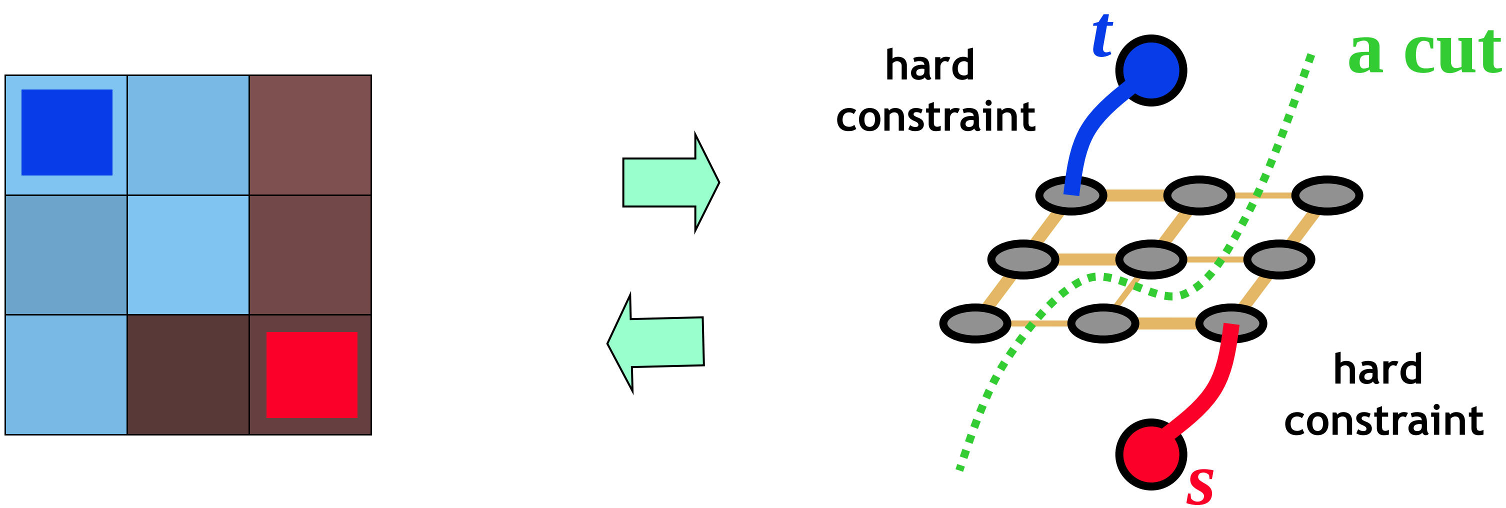
# Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation
- Graph cuts for image segmentation
  - Basic idea
  - s-t Mincut algorithm
  - Extension to non-binary case
- Applications
  - Interactive segmentation



# Graph Cuts for Optimal Boundary Detection

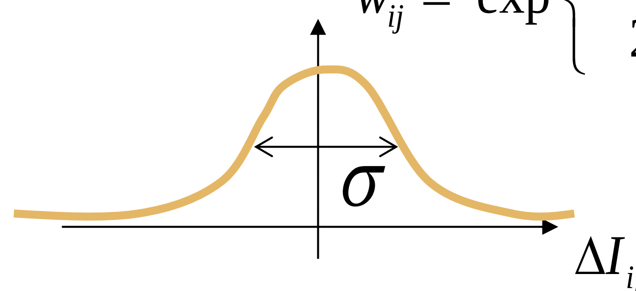
- **Idea: convert MRF into source-sink graph**




**Minimum cost cut can be computed in polynomial time**

**(max-flow/min-cut algorithms)**

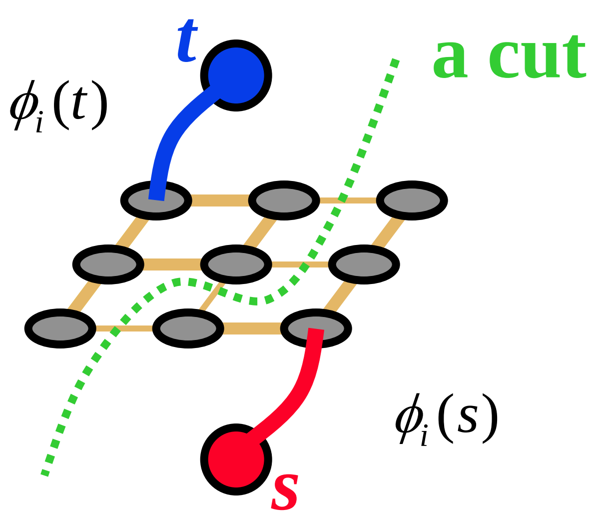
$$w_{ij} = \exp\left\{ -\frac{\Delta I_{ij}}{2\sigma^2} \right\}$$



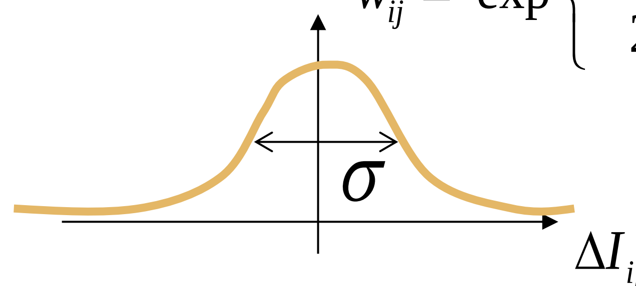
# Simple Example of Energy

$$E(\mathbf{x},\mathbf{y}) = \sum_i \phi_i(x_i) + \sum_{i,j} w_{ij} \cdot \delta(x_i \neq x_j)$$

**Unary terms** **Pairwise terms**



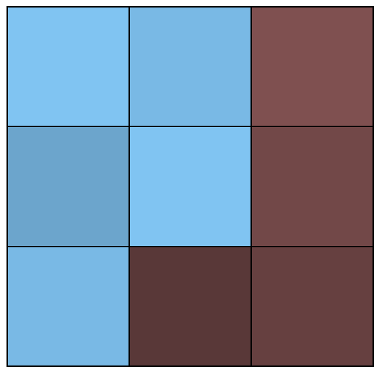

$$w_{ij} = \exp\left\{-\frac{\Delta I_{ij}}{2\sigma^2}\right\}$$

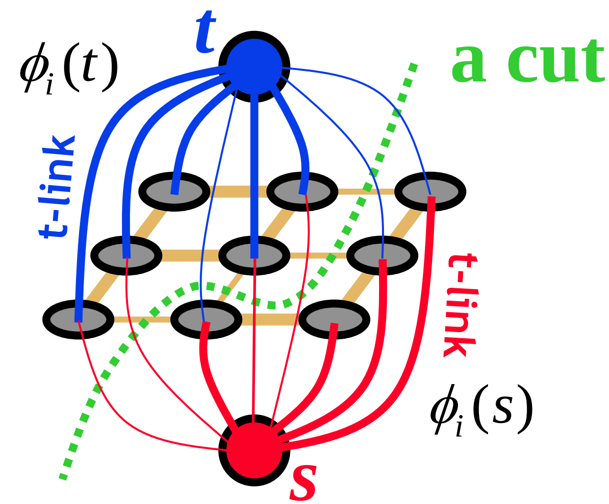



$$x \in \{s, t\}$$

**(binary object segmentation)**

# Adding Regional Properties






**Regional bias example**

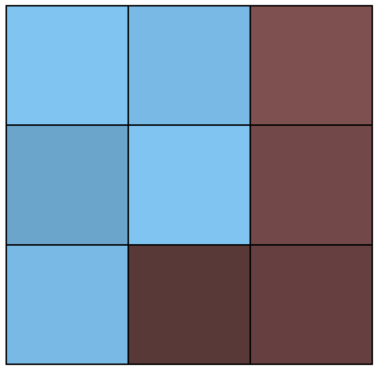
**Suppose $I^s$ and $I^t$ are given "expected" intensities of object and background**

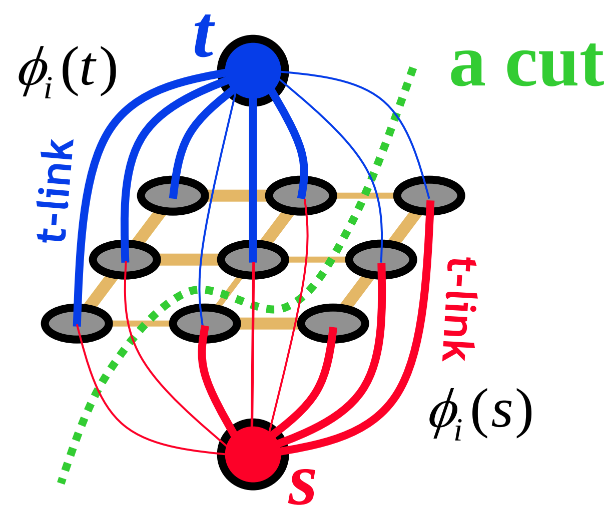
$$\phi_i(s) \propto \exp\left(-\| I_i - I^s \|^2 / 2\sigma^2\right)$$

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

**NOTE: hard constrains are not required, in general.**

# Adding Regional Properties






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

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

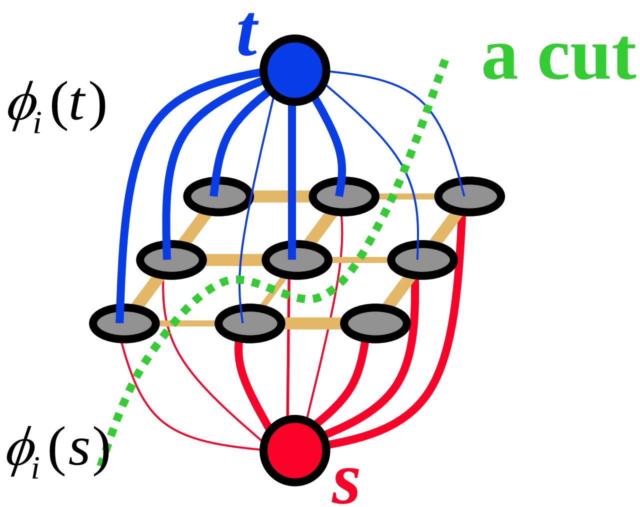
$$\phi_i(t) \propto \exp\left(-\| I_i - I^t \|^2 / 2\sigma^2\right)$$

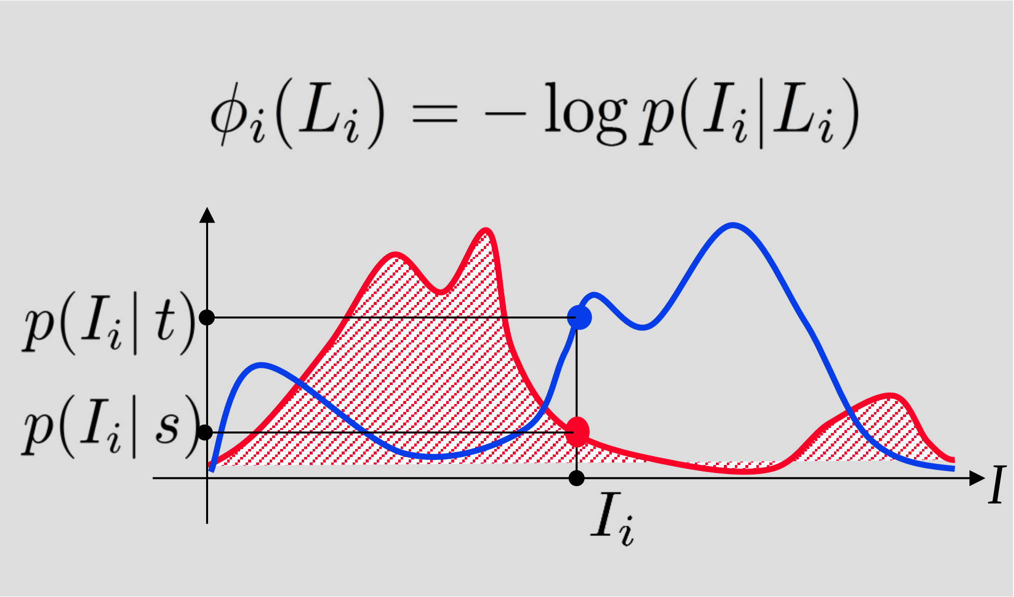
**EM-style optimization**


[Boykov & Jolly, ICCV'01]

# Adding Regional Properties

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




given object and background intensity histograms

# How to Set the Potentials? Some Examples

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

$$\phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) p(k|x_i) \mathcal{N}(y_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

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

$$\psi(x_i, x_j, g_{ij}(\mathbf{y}); \theta_\psi) = -\theta_\psi g_{ij}(\mathbf{y}) \delta(x_i \neq x_j)$$

**where**

$$g_{ij}(\mathbf{y}) = e^{-\beta \|y_i - y_j\|^2} \qquad \beta = \frac{1}{2}\left(\text{avg}\left(\|y_i - y_j\|^2\right)\right)^{-1}$$

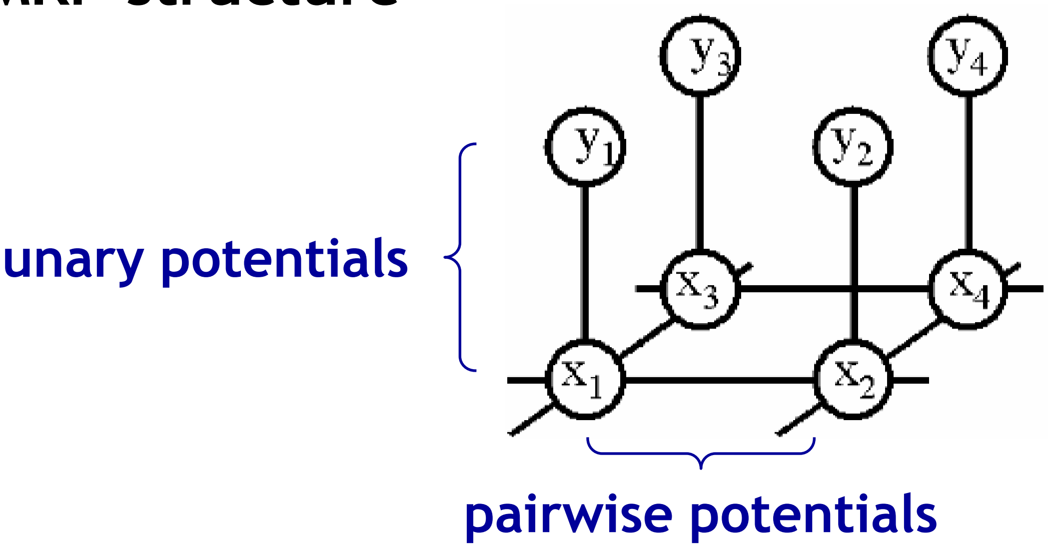
- **Parameters $\theta_\phi$, $\theta_\psi$ need to be learned, too!**



[Shotton & Winn, ECCV'06]

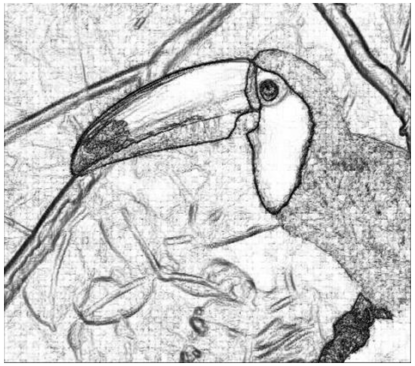
# Example: MRF for Image Segmentation

- **MRF structure**




Data (D)

Unary likelihood



Pair-wise Terms

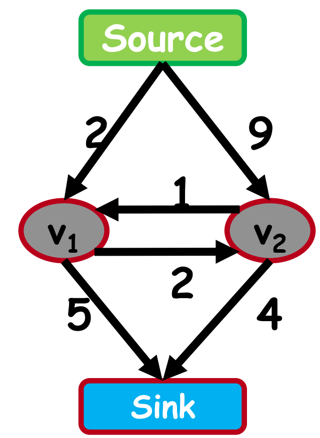
MAP Solution



Slide adapted from Phil Torr

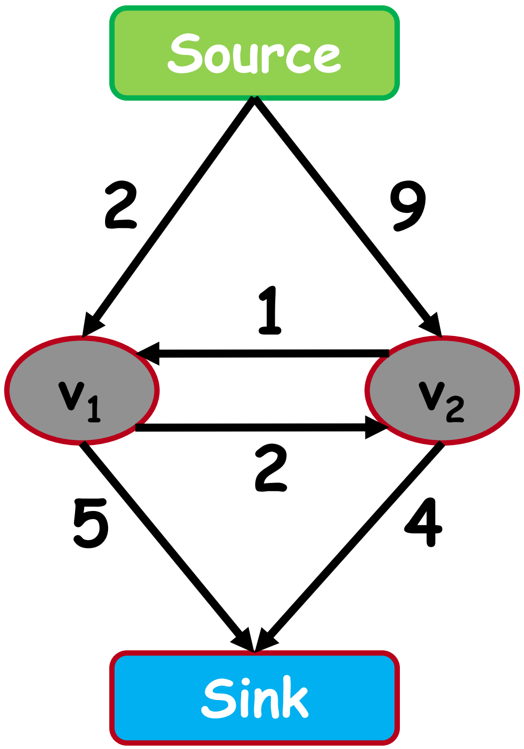
# Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation
- Graph cuts for image segmentation
  - Basic idea
  - s-t Mincut algorithm
  - Extension to non-binary case
- Applications
  - Interactive segmentation



RWTH AACHEN UNIVERSITY

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



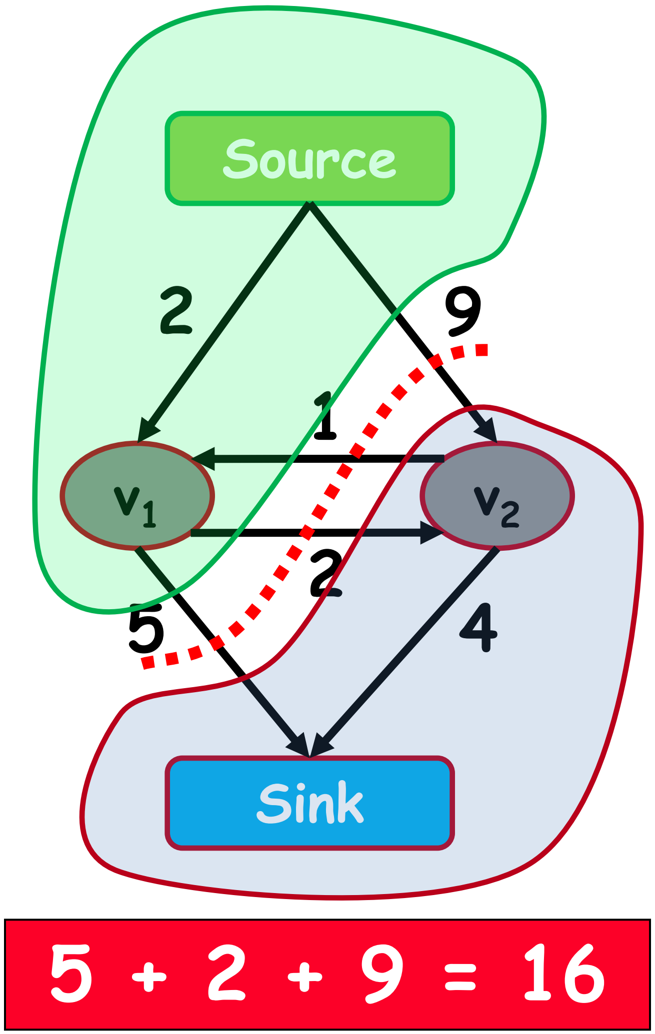

**Graph (V, E, C)**

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

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

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

# The s-t-Mincut Problem




## What is an st-cut?

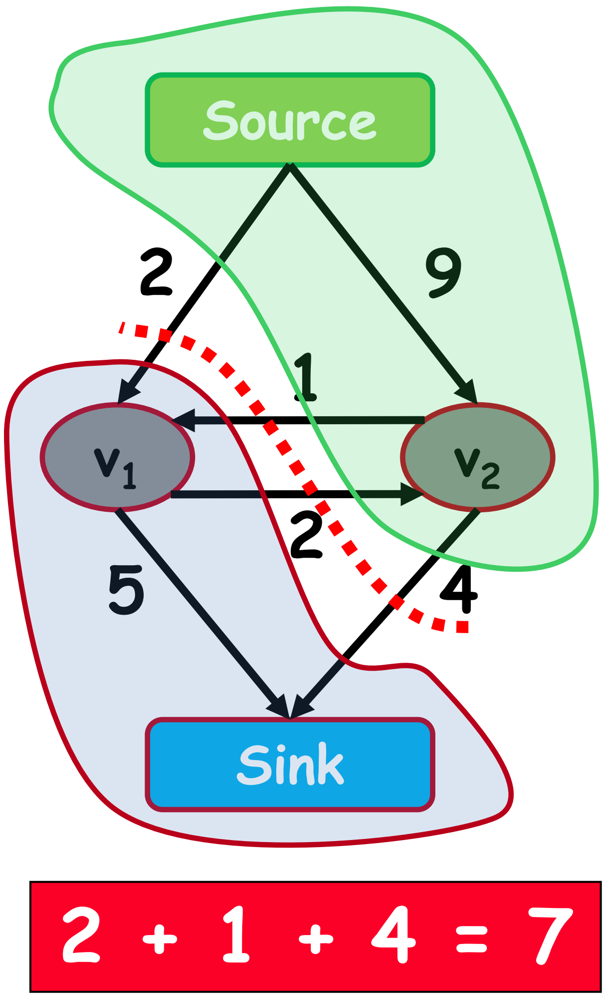
An st-cut (S,T) divides the nodes between source and sink.

## What is the cost of a st-cut?

Sum of cost of all edges going from S to T

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?**

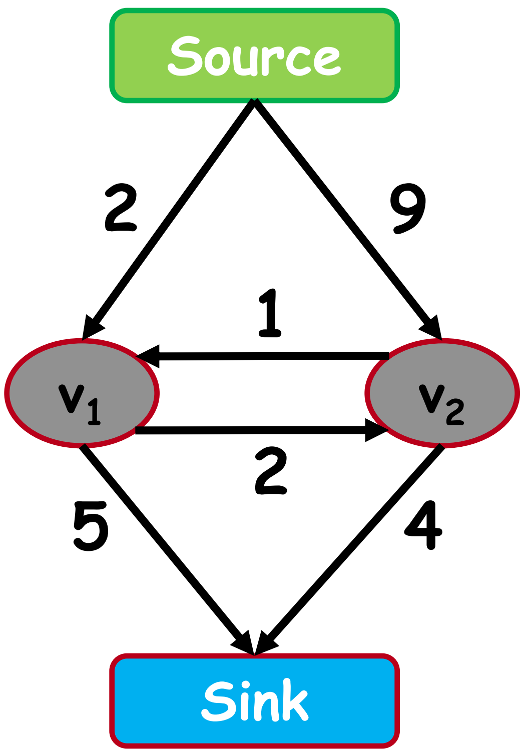
Sum of cost of all edges going from S to T

**What is the st-mincut?**

st-cut with the minimum cost

Slide credit: Pushmeet Kohli

# 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

# 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 | $O(m^{3/2} \log(n^2/m) \log U)$<br>$O(n^{2/3}m \log(n^2/m) \log U)$ |

$n$: #nodes

$m$: #edges

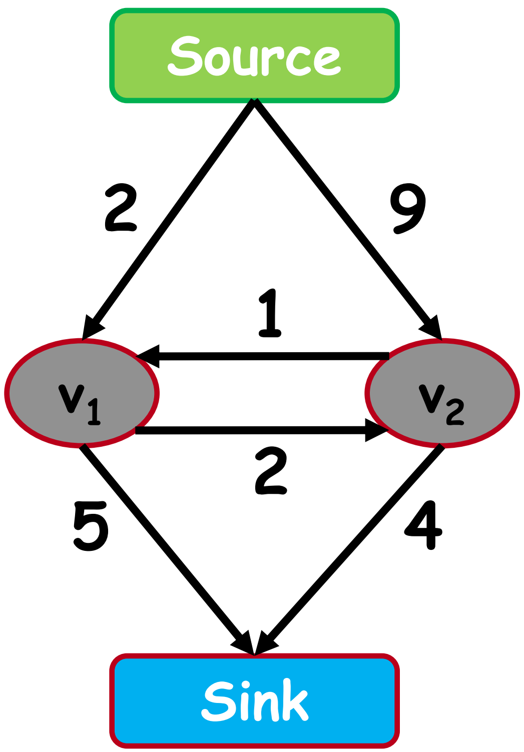
$U$: maximum edge weight

**Algorithms assume non-negative edge weights**

Slide credit: Andrew Goldberg

# Maxflow Algorithms

Flow = 0

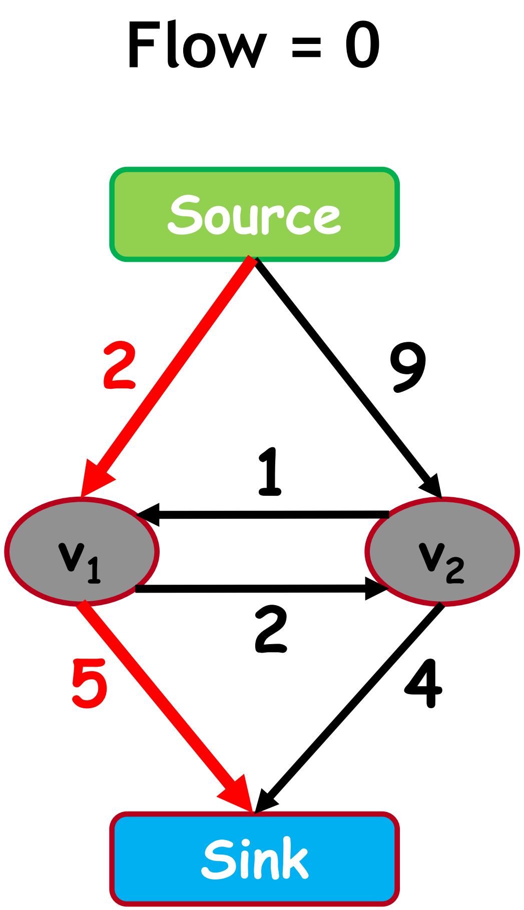



Augmenting Path Based Algorithms

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**

# Maxflow Algorithms



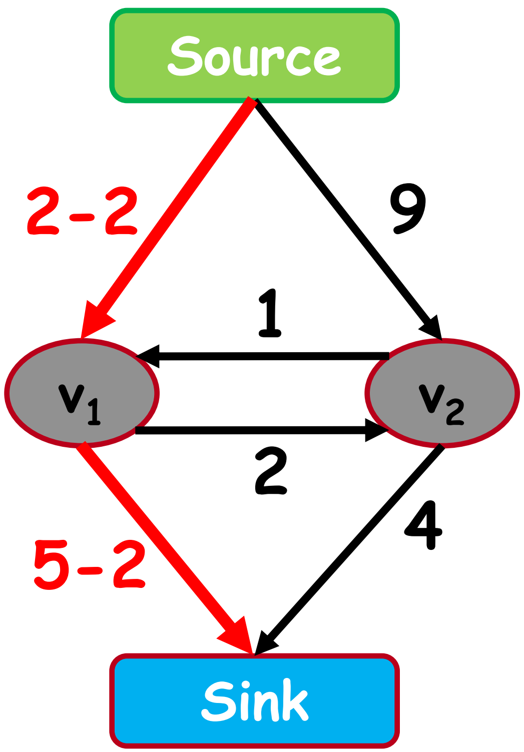

Augmenting Path Based Algorithms

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**

# Maxflow Algorithms

Flow = 0 + 2



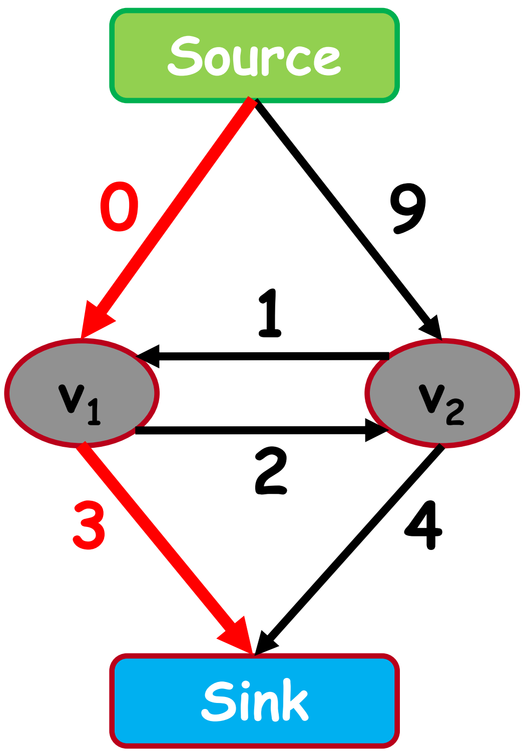

Augmenting Path Based Algorithms

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

# Maxflow Algorithms

Flow = 2

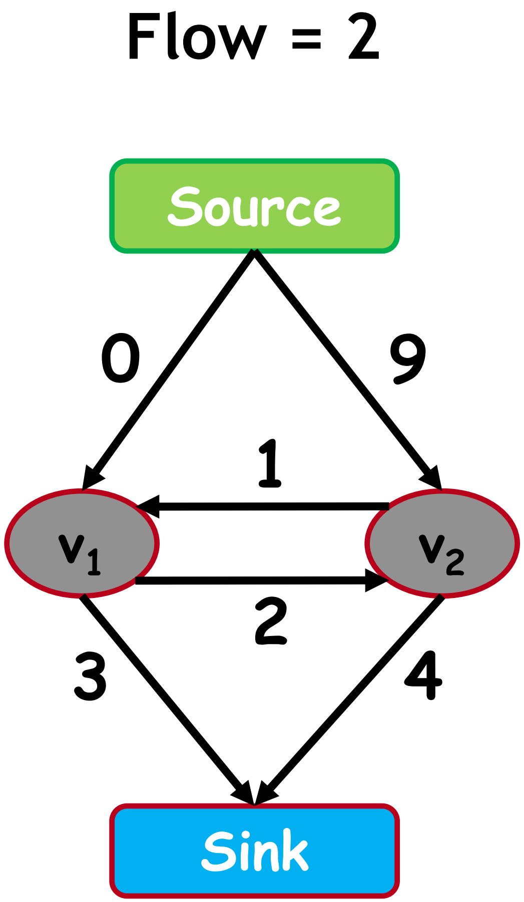



## Augmenting Path Based Algorithms

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**

# Maxflow Algorithms



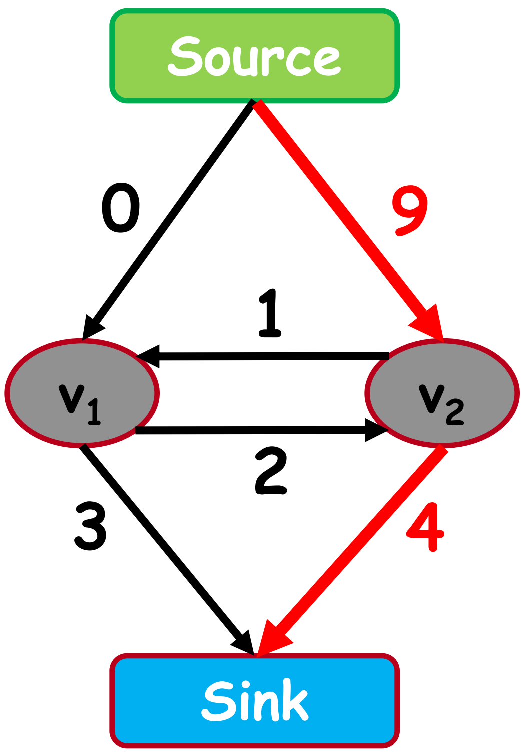

Augmenting Path Based Algorithms

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**

# Maxflow Algorithms

Flow = 2



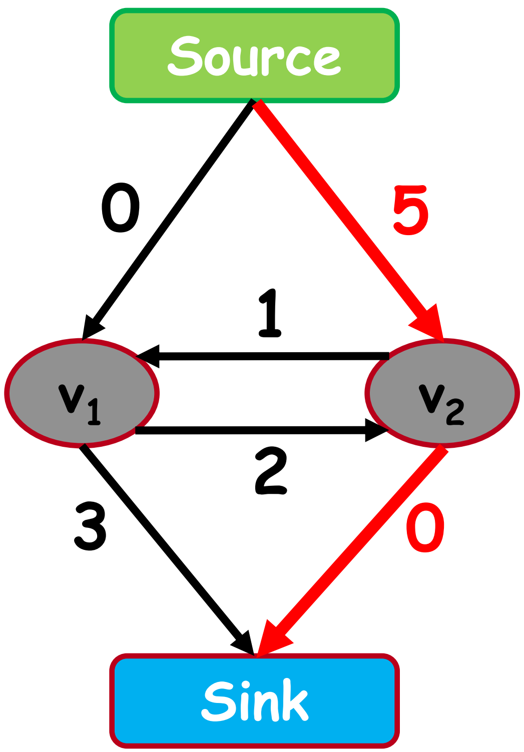

## Augmenting Path Based Algorithms

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**

# Maxflow Algorithms

Flow = 2 + 4




Augmenting Path Based Algorithms

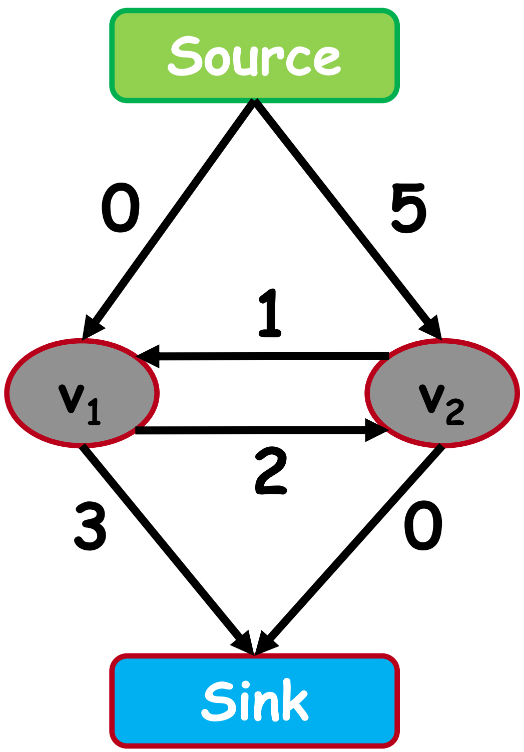
1. Find path from source to sink with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

**Algorithms assume non-negative capacity**

Slide credit: Pushmeet Kohli

# Maxflow Algorithms

Flow = 6



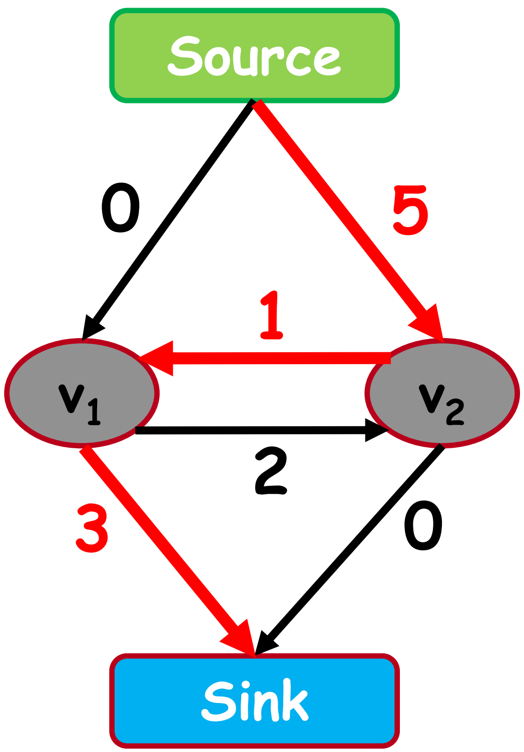

Augmenting Path Based Algorithms

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

# Maxflow Algorithms

Flow = 6



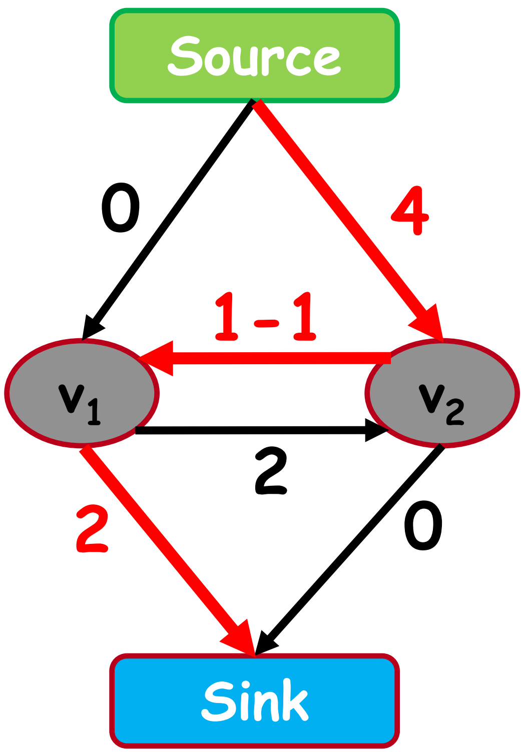

Augmenting Path Based Algorithms

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

# Maxflow Algorithms

Flow = 6 + 1

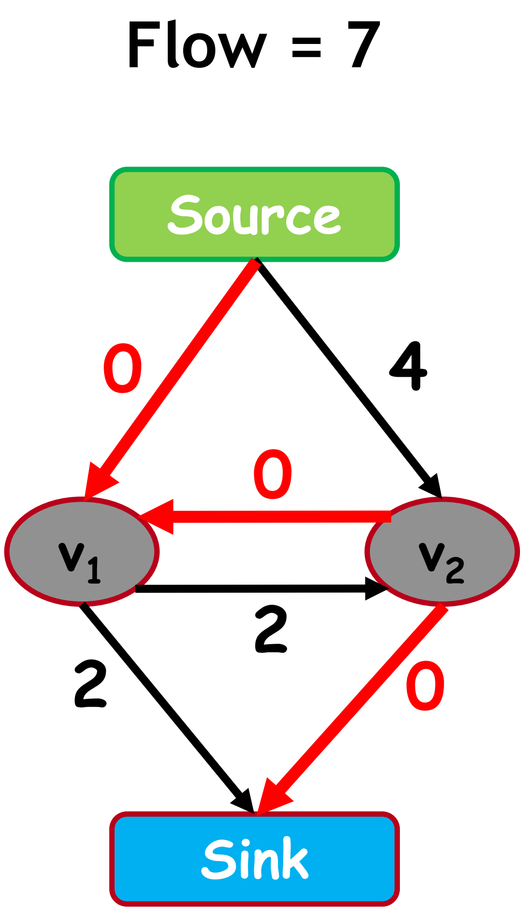



## Augmenting Path Based Algorithms

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

# Maxflow Algorithms

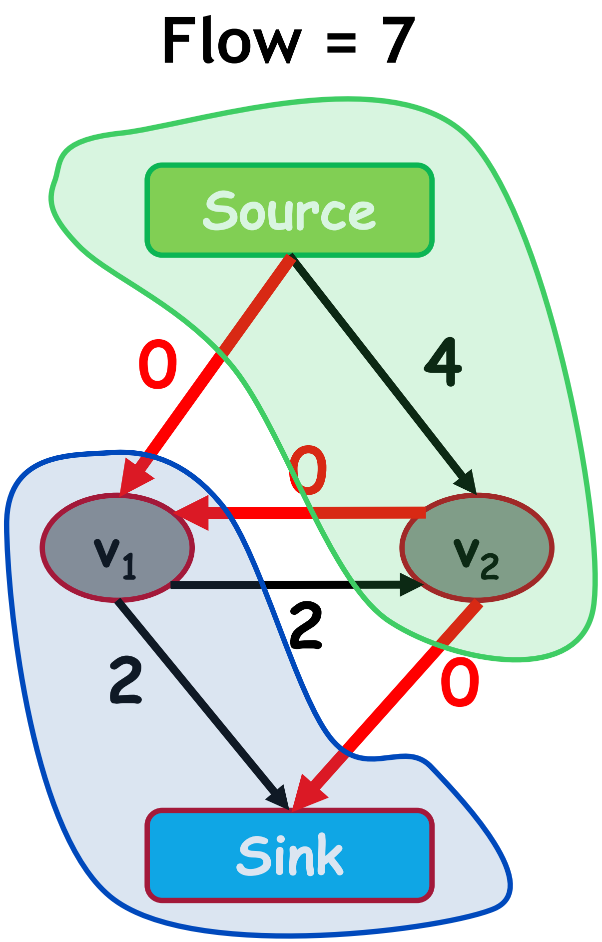



Augmenting Path Based Algorithms

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

# Maxflow Algorithms



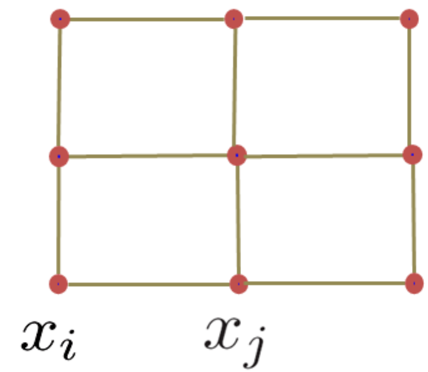

## Augmenting Path Based Algorithms

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

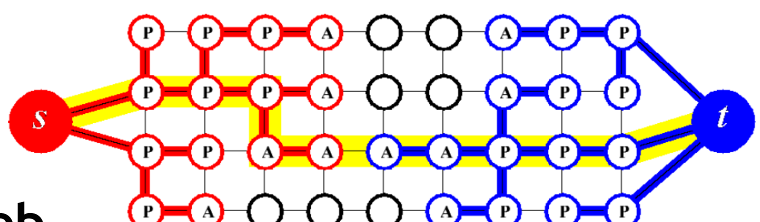
**Algorithms assume non-negative capacity**

# Applications: 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.cs.ucl.ac.uk/staff/V.Kolmogorov/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]

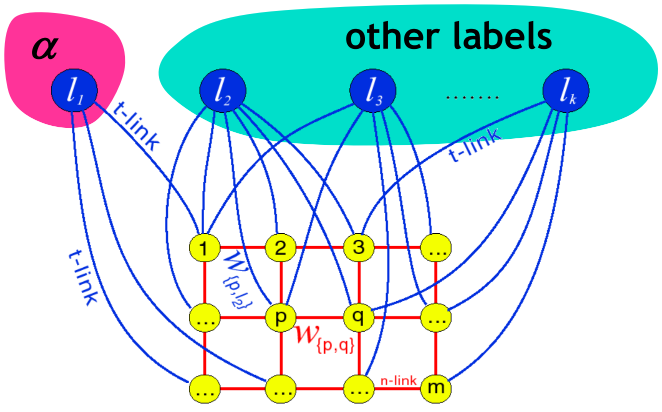
***E(L)*** **can be minimized by *s-t* graph cuts** $\Leftrightarrow$ $E(s,s) + E(t,t) \leq E(s,t) + E(t,s)$

Submodularity ("convexity")

- Non-submodular cases can still be addressed with some optimality guarantees.
  - Current research topic

# Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation
- **Graph cuts for image segmentation**
  - **Basic idea**
  - **s-t Mincut algorithm**
  - **Extension to non-binary case**
- Applications
  - Interactive segmentation

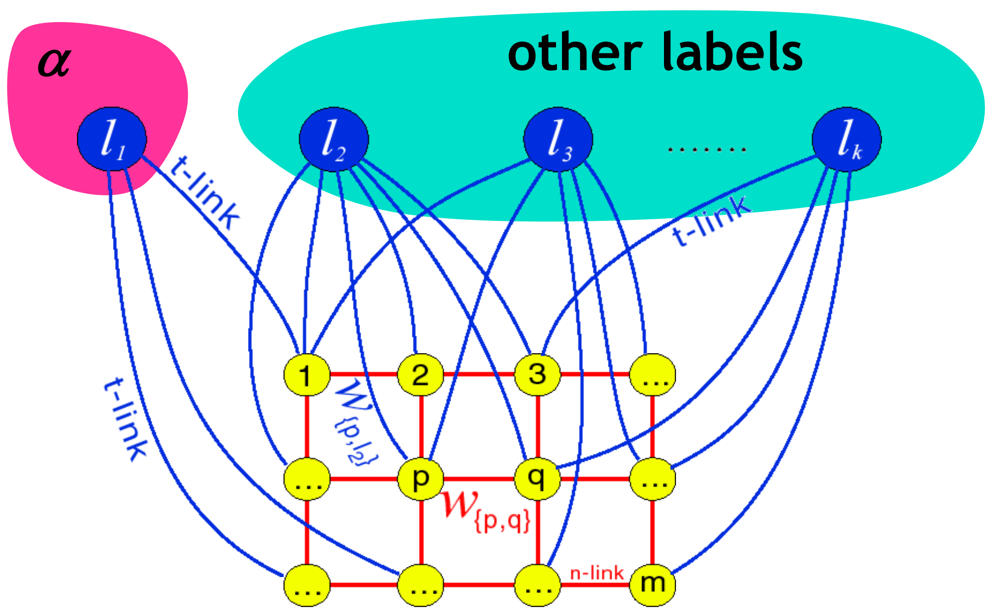


# Dealing with Non-Binary Cases

- Limitation to binary energies is often a nuisance.
  - ⇒ E.g. binary segmentation only…
- We would like to solve also multi-label problems.
  - The bad news: Problem is NP-hard with 3 or more labels!
- There exist some approximation algorithms which extend graph cuts to the multi-label case:
  - $\alpha$-Expansion
  - $\alpha\beta$-Swap
- They are no longer guaranteed to return the globally optimal result.
  - But $\alpha$-Expansion has a guaranteed approximation quality (2-approx) and converges in a few iterations.

# α-Expansion Move

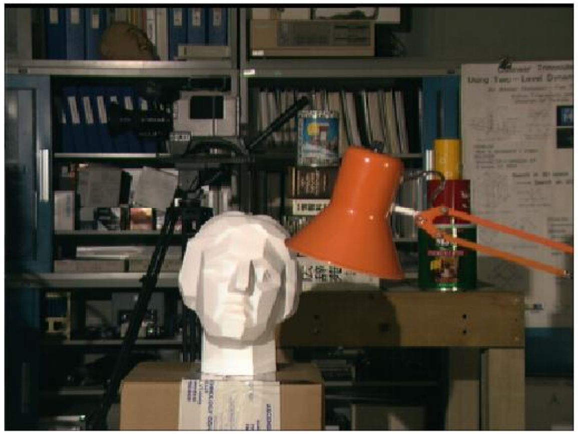
- **Basic idea:**
  - Break multi-way cut computation into a sequence of binary s-t cuts.

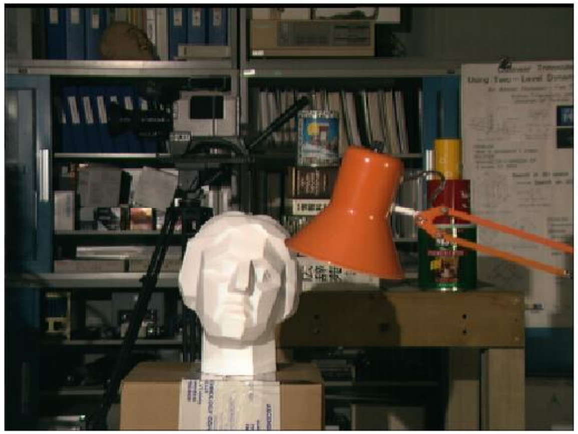


# $\alpha$-Expansion Algorithm

1. Start with any initial solution
2. For each label "$\alpha$" in any (e.g. random) order:
   1. Compute optimal $\alpha$-expansion move (s-t graph cuts).
   2. Decline the move if there is no energy decrease.
3. Stop when no expansion move would decrease energy.

# Example: Stereo Vision





Original pair of "stereo" images


Depth map

# α-Expansion Moves

- In each α-expansion a given label "α" grabs space from other labels

initial solution

-expansion

-expansion

-expansion

-expansion

-expansion

-expansion

-expansion

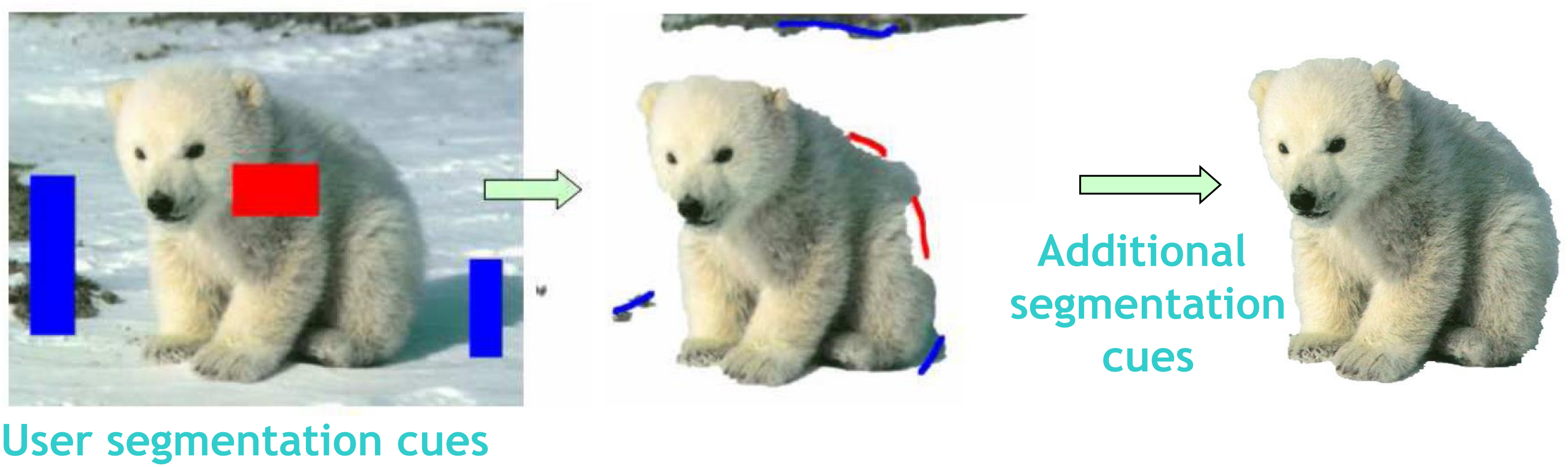
For each move, we choose the expansion that gives the largest decrease in the energy: ⇒ binary optimization problem

# Topics of This Lecture

- Segmentation as Energy Minimization
  - Markov Random Fields
  - Energy formulation
- Graph cuts for image segmentation
  - Basic idea
  - s-t Mincut algorithm
  - Extension to non-binary case
- **Applications**
  - **Interactive segmentation**

# 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.
  - This is used to create an initial segmentation which can then be corrected by additional brush strokes.

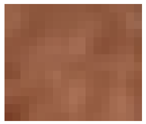




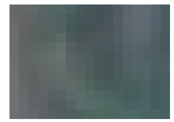

Slide credit: Matthieu Bray

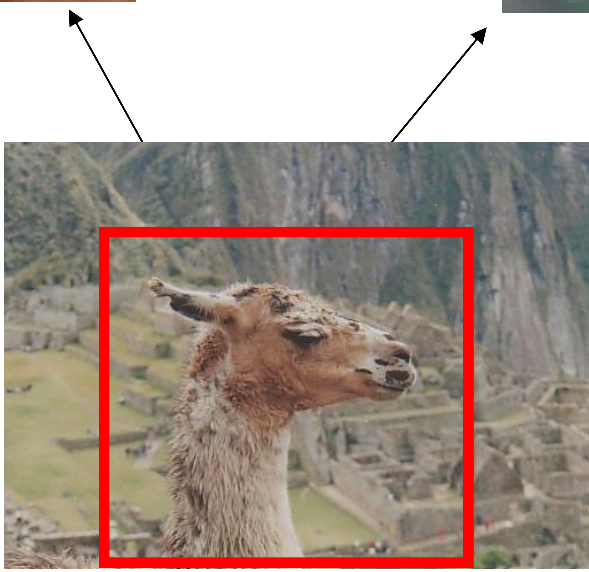
# GrabCut: Data Model

Foreground color



Background color





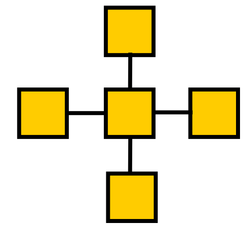
Global optimum of the energy

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

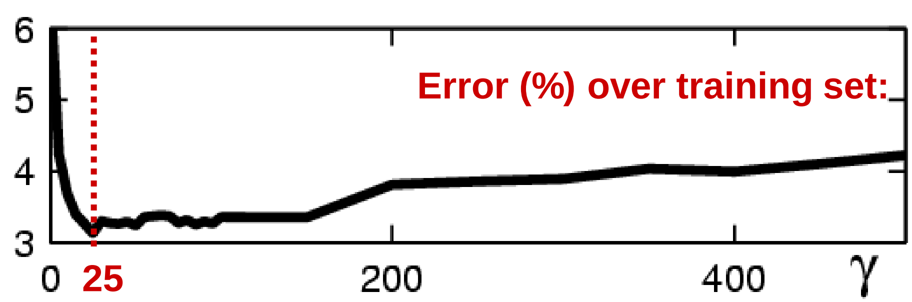
# GrabCut: Coherence Model

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

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

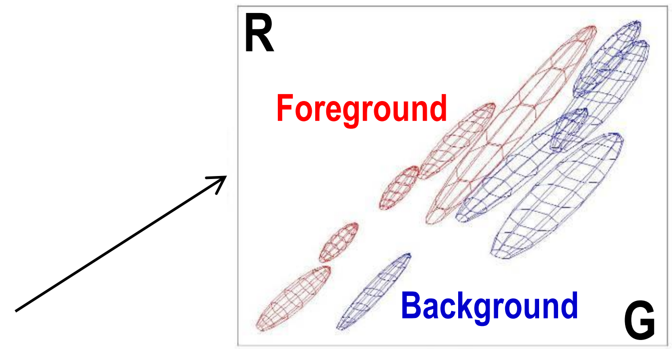


**How to choose $\gamma$?**

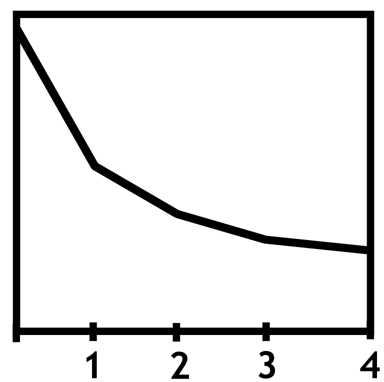


# Iterated Graph Cuts

Result

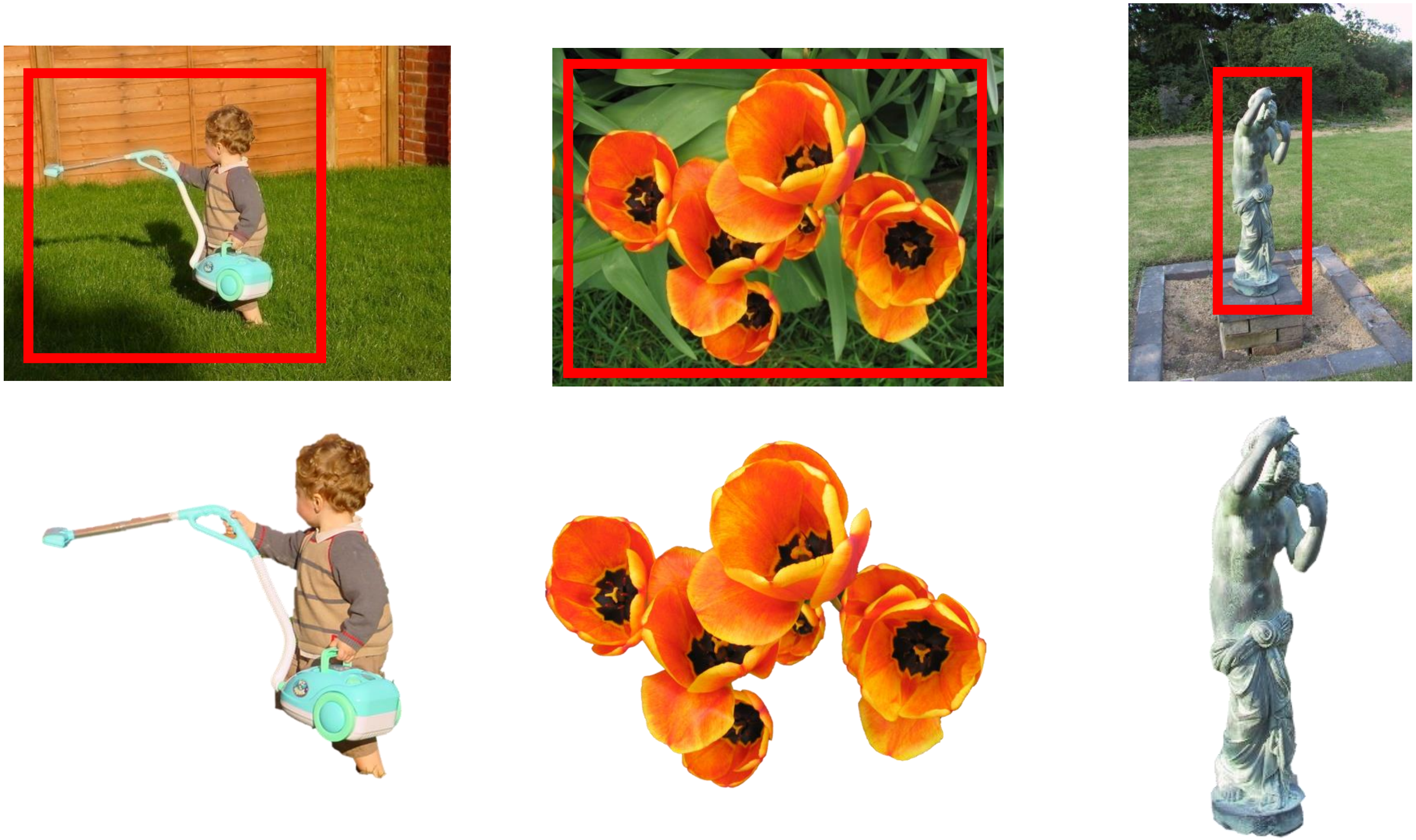



Color model
(Mixture of Gaussians)




Energy after
each iteration

# GrabCut: Example Results

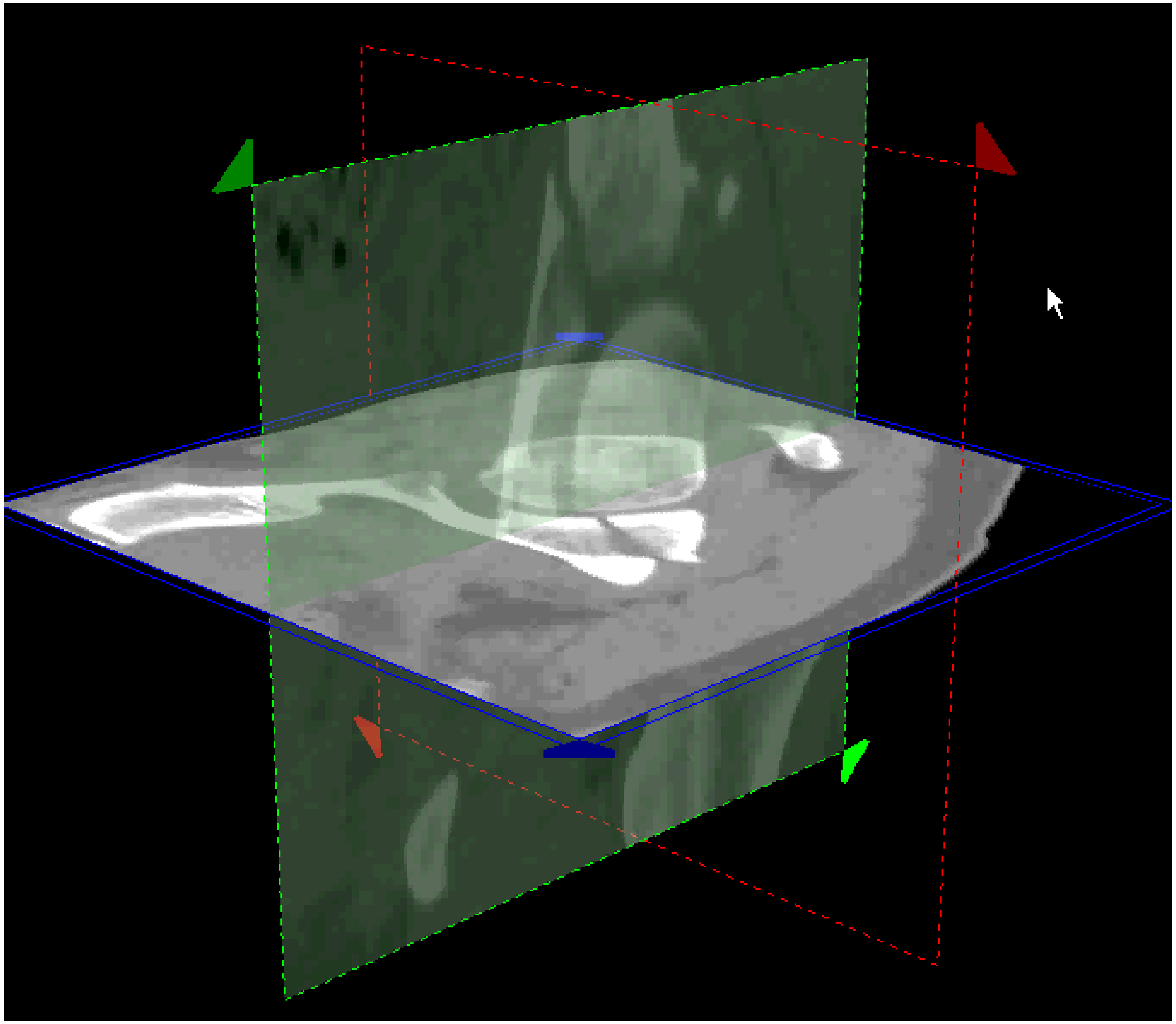


- ***This is included in the newest version of MS Office!***



Image source: Carsten Rother

# Applications: Interactive 3D Segmentation



# 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 only solve a limited class of models
    - Submodular energy functions
    - Can capture only part of the expressiveness of MRFs
  - Only approximate algorithms available for multi-label case

# References and Further Reading

- A gentle introduction to Graph Cuts can be found in the following paper:
  - Y. Boykov, O. Veksler, Graph Cuts in Vision and Graphics: Theories and Applications. In *Handbook of Mathematical Models in Computer Vision*, edited by N. Paragios, Y. Chen and O. Faugeras, Springer, 2006.
- Read how the interactive segmentation is realized in MS Office 2010
  - C. Rother, V. Kolmogorov, Y. Boykov, A. Blake, Interactive Foreground Extraction using Graph Cut, Microsoft Research Tech Report MSR-TR-2011-46, March 2011
- Try the GraphCut implementation at

  http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html