Lecture 9:
Image Segmentation

Goal

- Goal: Identify groups of pixels that go together

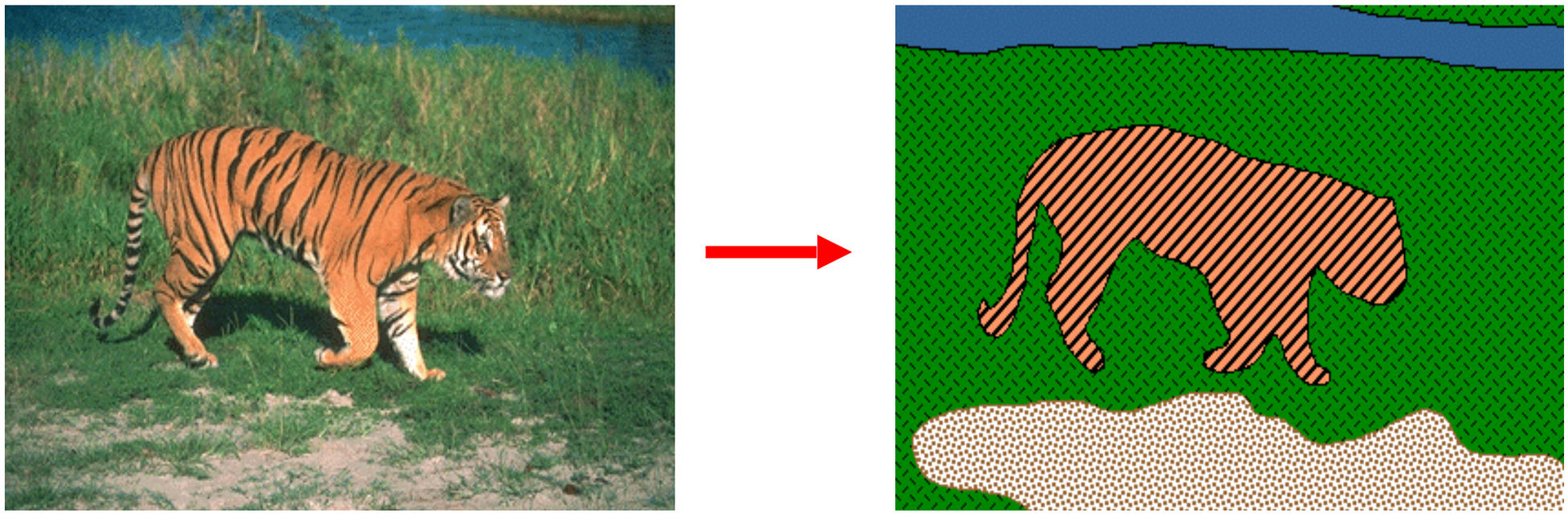


image credit: Steve Seitz, Kristen Grauman

Types of Segmentation

- Semantic Segmentation: Assign labels

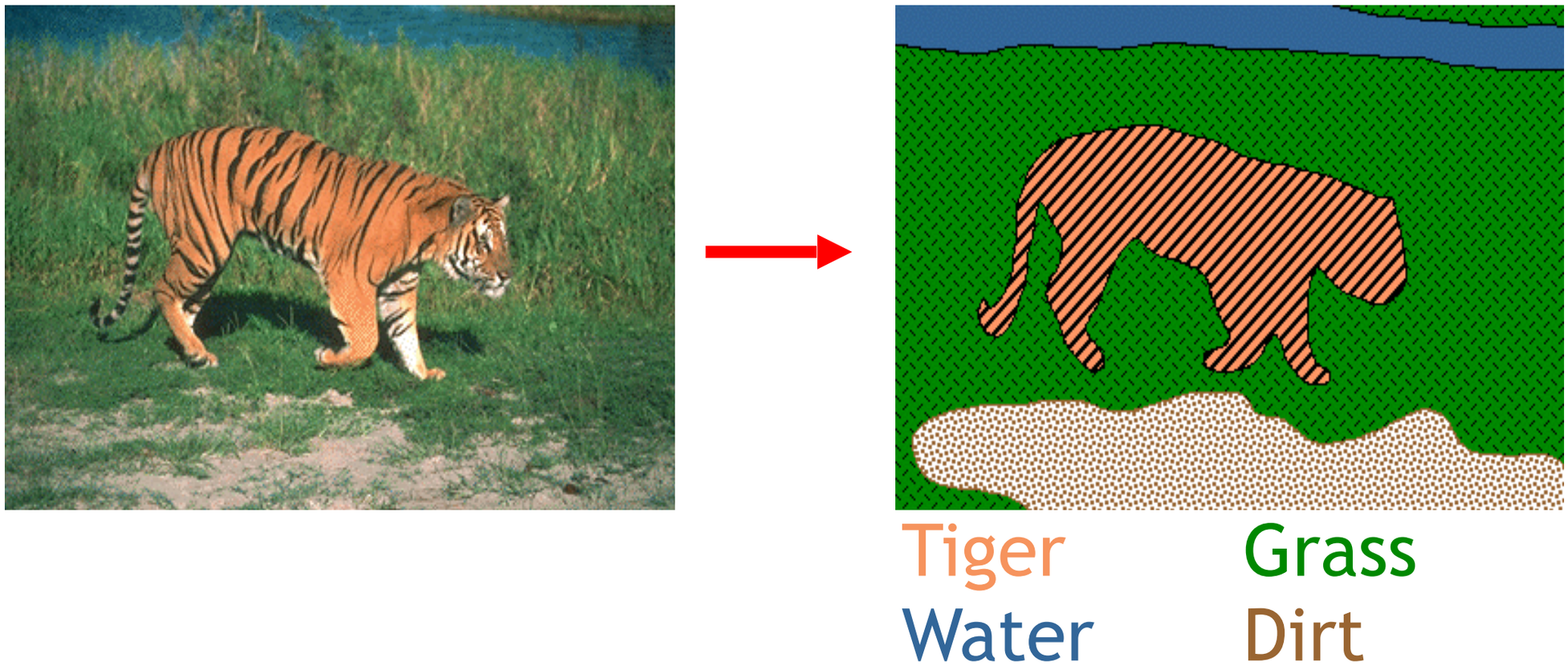


image credit: Steve Seitz, Kristen Grauman

Types of Segmentation

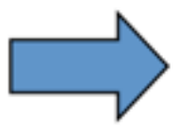
- Figure-ground segmentation: Foreground/background

image credit: Carsten Rother

Application: As a result

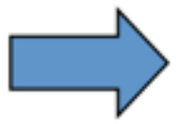
Rother et al. 2004

Application: Speed up Recognition



[Felzenszwalb and Huttenlocher 2004]

[Hoiem et al. 2005, Mori 2005]



[Shi and Malik 2001]

Slide: Derek Hoiem

History: Before Computer Vision

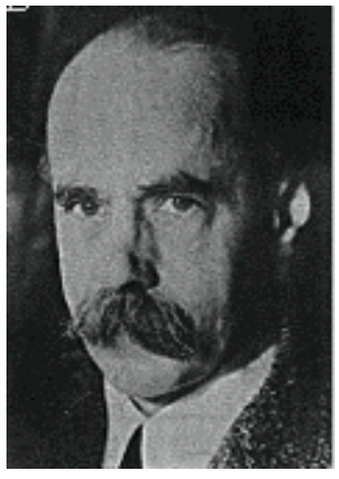
Gestalt Theory

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

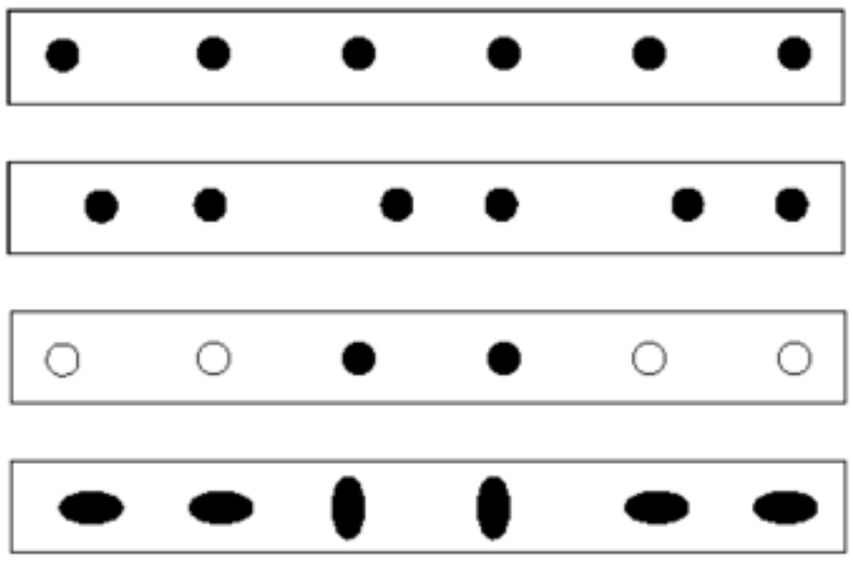
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

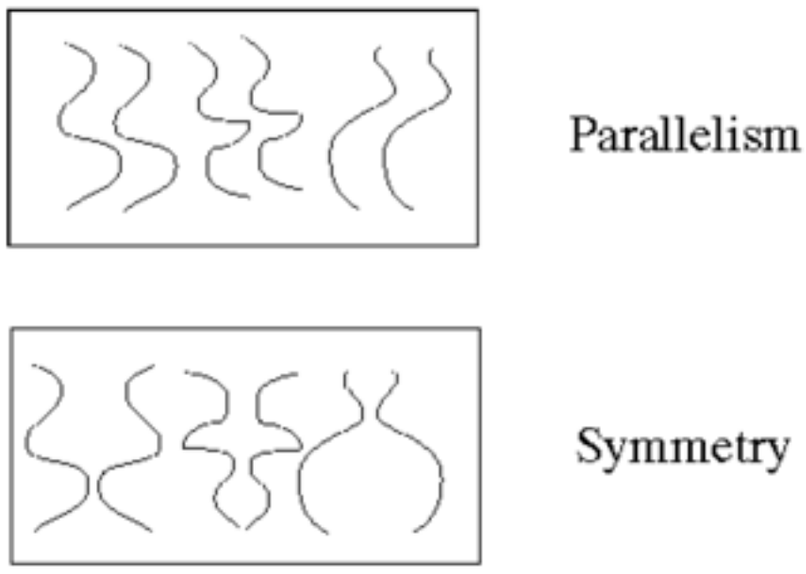
"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."

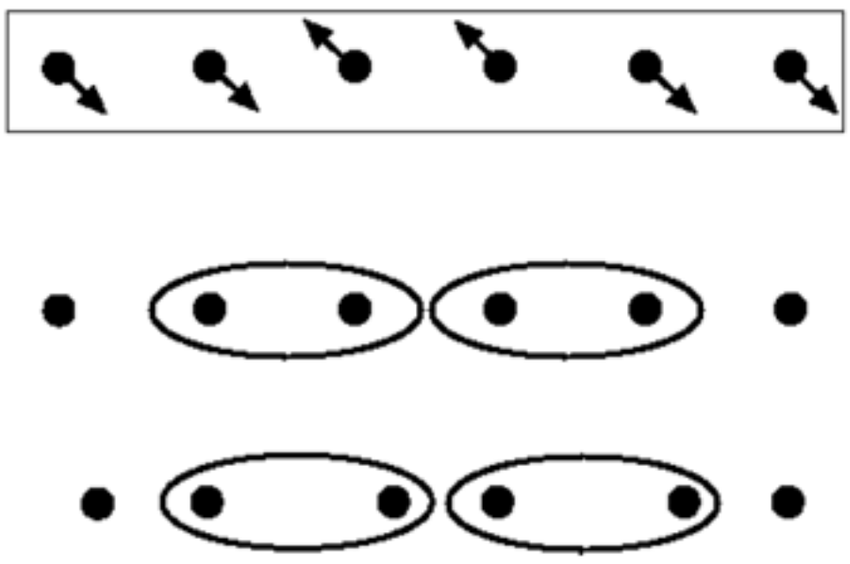
Max Wertheimer
(1880-1943)



Gestalt Factors







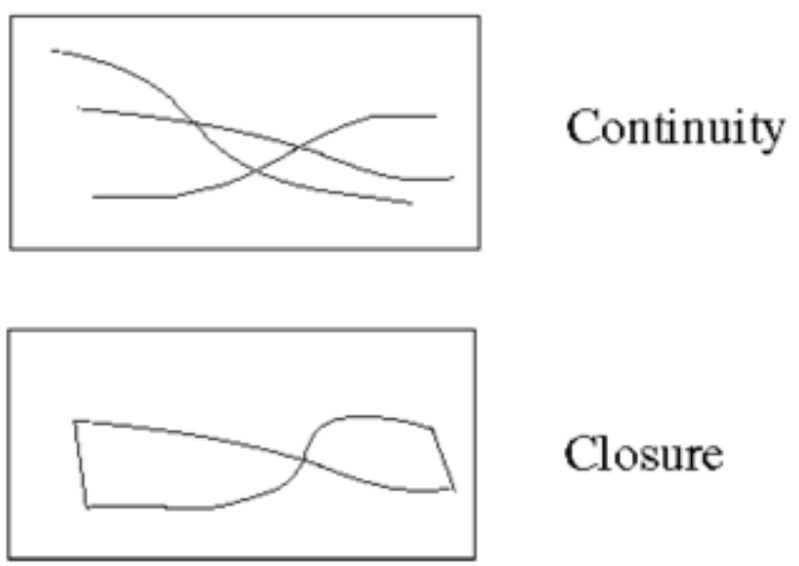


Image source: Forsyth & Ponce

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

Outline

1. Segmentation as clustering
2. Graph-based segmentation
3. Segmentation as energy minimization

Outline

1. Segmentation as clustering
    1. K-Means
    2. GMMs and EM
    3. Mean Shift
2. Graph-based segmentation
3. Segmentation as energy minimization

K-means clustering

1. Initialize ($t = 0$): cluster centers $c_1, \ldots, c_K$
   - Commonly used: random initialization
   - Or greedily choose K to minimize residual
2. Compute d^t: assign each point to the closest center
   - Typical distance measure:
     - Euclidean $sim(x, x') = x^T x'$
     - Cosine $sim(x, x') = x^T x' / (\|x\| \times \|x'\|)$
     - Others

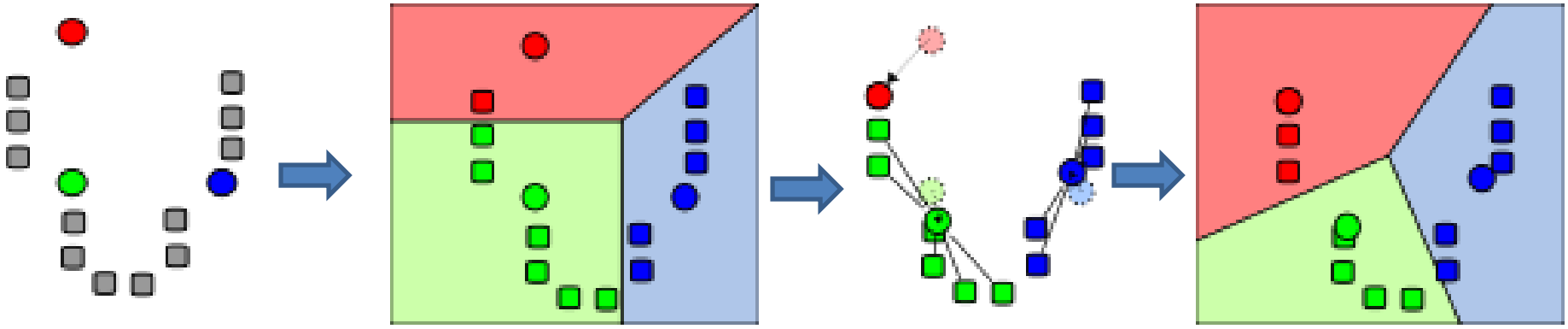
1. Computer c^t: update cluster centers as the mean of the points

$$c^t = \underset{c}{\operatorname{argmin}} \frac{1}{N} \sum_j^N \sum_i^K d_{ij} \left(c_i^{t-1} - x_j \right)^2$$

2. Update $t = t + 1$, Repeat Step 2-3 till stopped
   - c^t doesn't change anymore.

Slide: Derek Hoiem

K-means clustering



1. Initialize Cluster Centers

2. Assign Points to Clusters

3. Re-compute Means

Repeat (2) and (3)

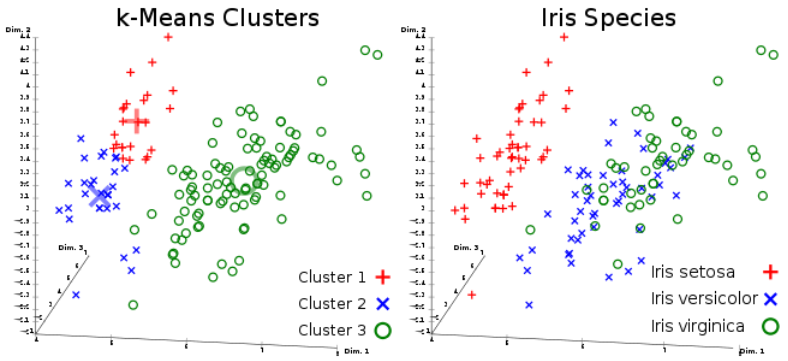
- Java demo:

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

K-means clustering

- Converges to a *local minimum* solution
  - Initialize multiple runs

- Better fit for spherical data

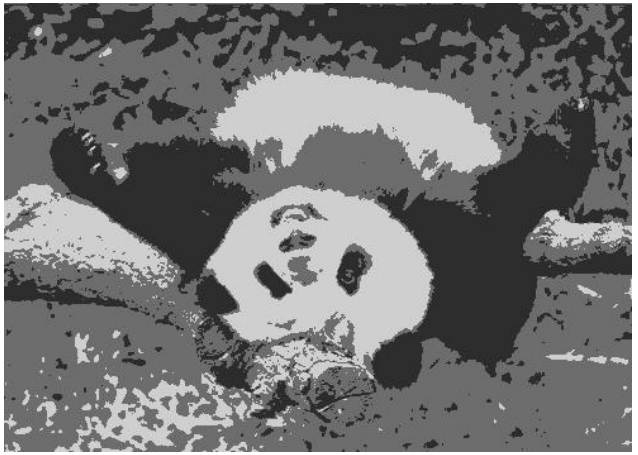


- Need to pick K (# of clusters)

Segmentation as Clustering

Original image

2 clusters



3 clusters

Feature Space

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

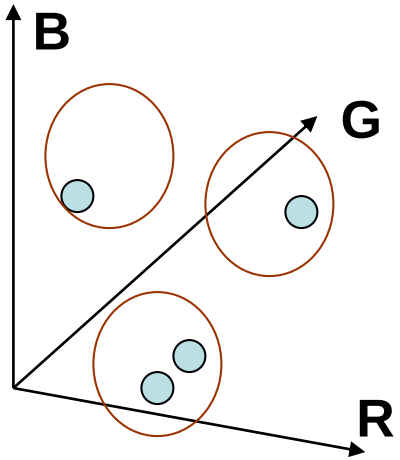
- Grouping pixels based on intensity similarity

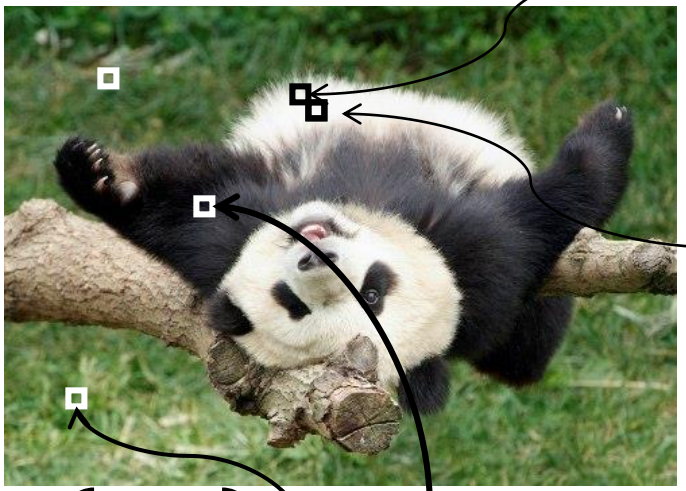
- Feature space: intensity value (1D)

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



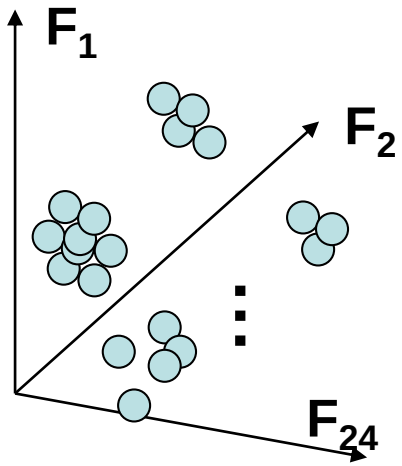


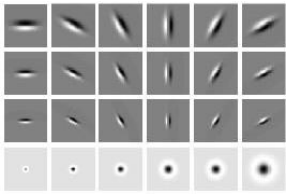
- Feature space: color value (3D)

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





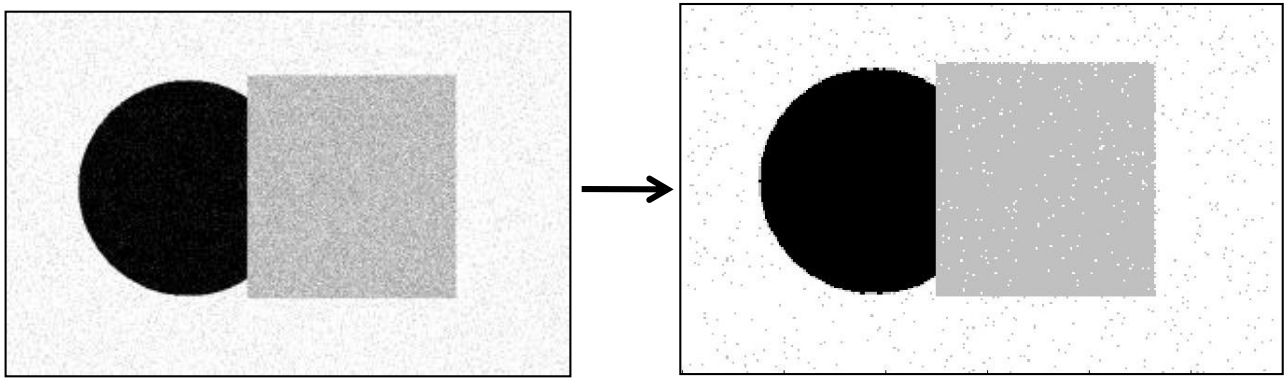
Filter bank of 24 filters

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

Slide credit: Kristen Grauman

Smoothing Out Cluster Assignments

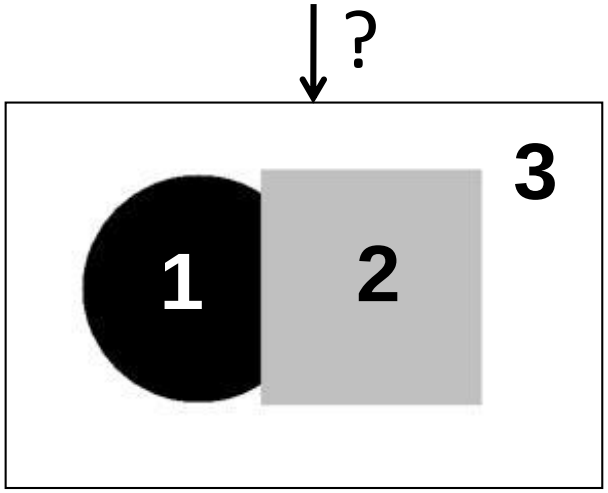
- Assigning a cluster label per pixel may yield outliers:



Original

Labeled by cluster center's intensity

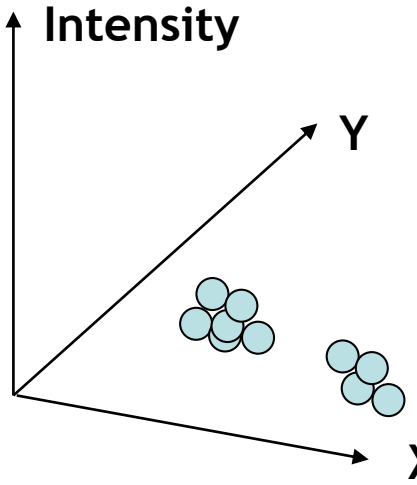
- How can we ensure they are spatially smooth?

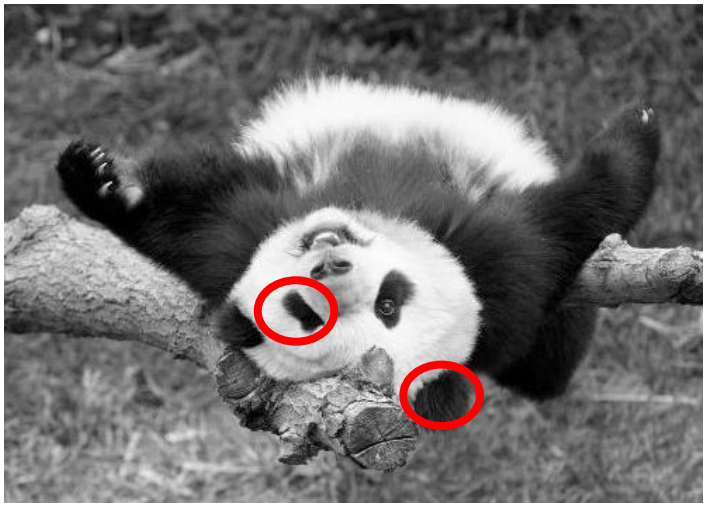


Slide credit: Kristen Grauman

Segmentation as Clustering

- Depending on what we choose as the *feature space*, we can group pixels in different ways.
- Grouping pixels based on *intensity+position* similarity





$\Rightarrow$ Way to encode both *similarity* and *proximity.*

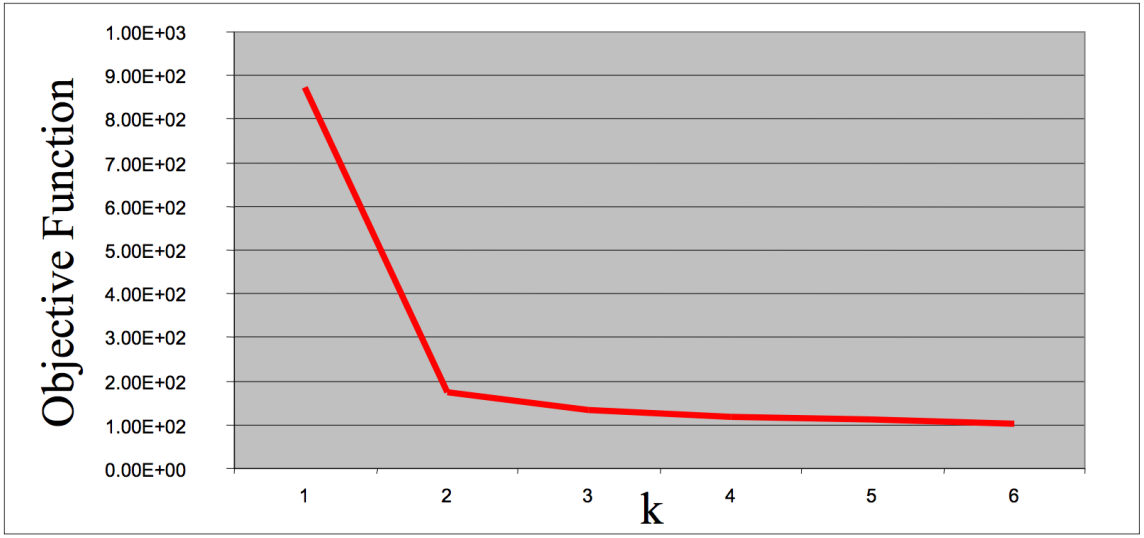
Slide credit: Kristen Grauman

How to choose the number of clusters?

Try different numbers of clusters in a validation set and look at performance.

We can plot the objective function values for k equals 1 to 6…

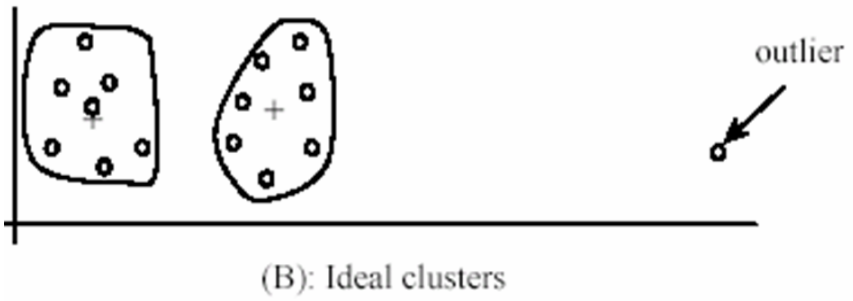
The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as "knee finding" or "elbow finding".



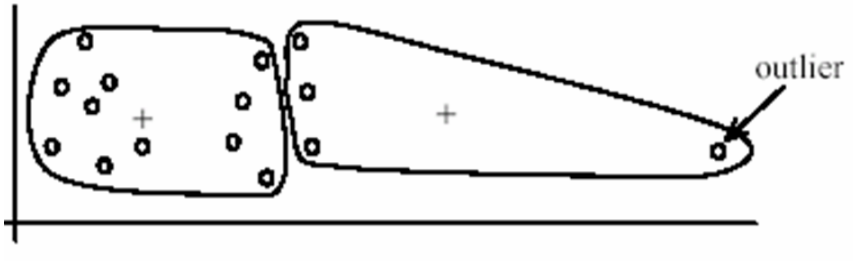
Slide: Derek Hoiem

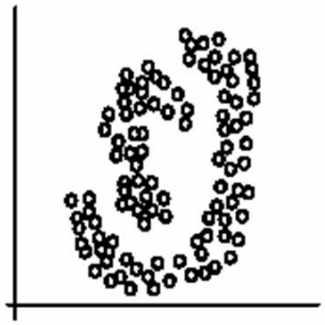
K-Means pros and cons

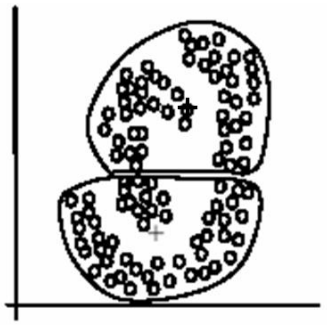
- Pros
  - Finds cluster centers that minimize conditional variance (good representation of data)
  - Simple and fast, Easy to implement
- Cons
  - Need to choose K
  - Sensitive to outliers
  - Prone to local minima
  - All clusters have the same parameters (e.g., distance measure is non-adaptive)
  - *Can be slow: each iteration is O(KNd) for N d-dimensional points
- Usage
  - Unsupervised clustering
  - Rarely used for pixel segmentation



(B): Ideal clusters




(A): Two natural clusters


(B): *k*-means clusters

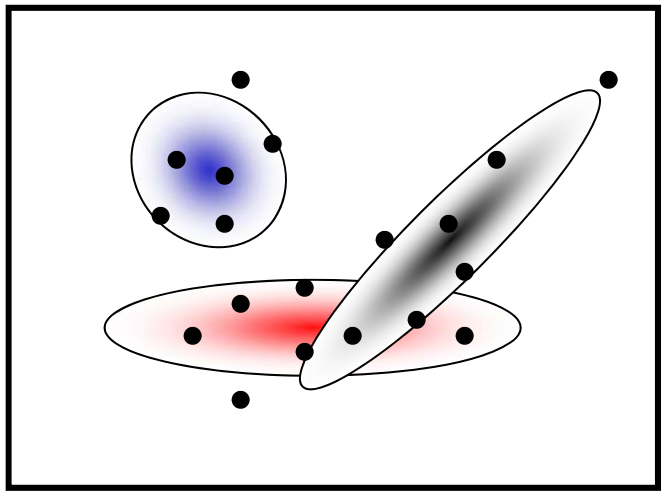
Probabilistic Clustering

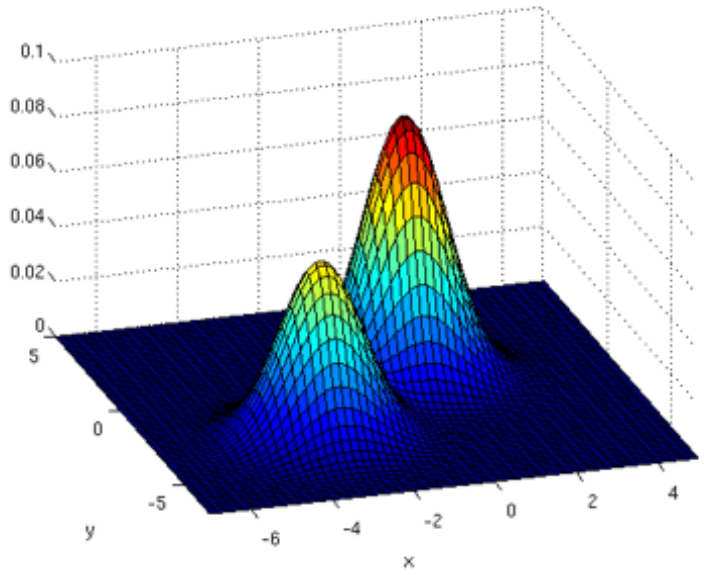
- Basic questions
  - What's the probability that a point x is in cluster m?
  - What's the shape of each cluster?
- K-means doesn't answer these questions.

- 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



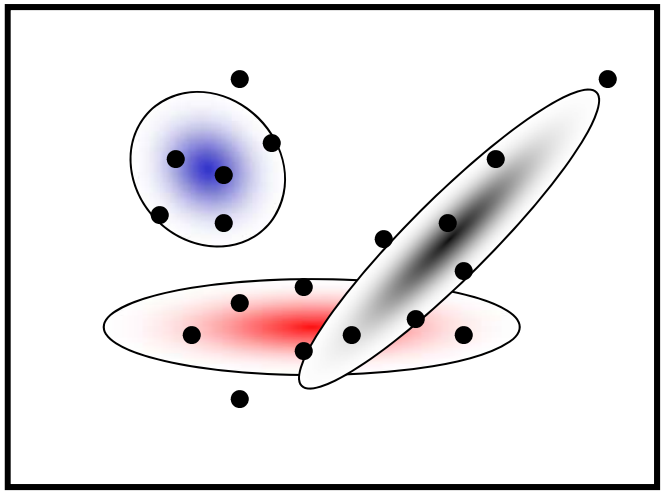


- One generative model is a mixture of Gaussians (MoG)
  - K Gaussian blobs with means μ_b covariance matrices V_b, dimension d
    - Blob *b* defined by: $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
  - Blob *b* is selected with probability α_b
 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

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

Slide credit: Steve Seitz

Expectation Maximization (EM)



- Goal
  - Find blob parameters θ that maximize the likelihood function:

$$P(data|\theta) = \prod_x P(x|\theta)$$

- 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

Slide credit: 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 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 credit: Steve Seitz

Clustering: Expectation Maximization (EM)

GMMs

Pro:

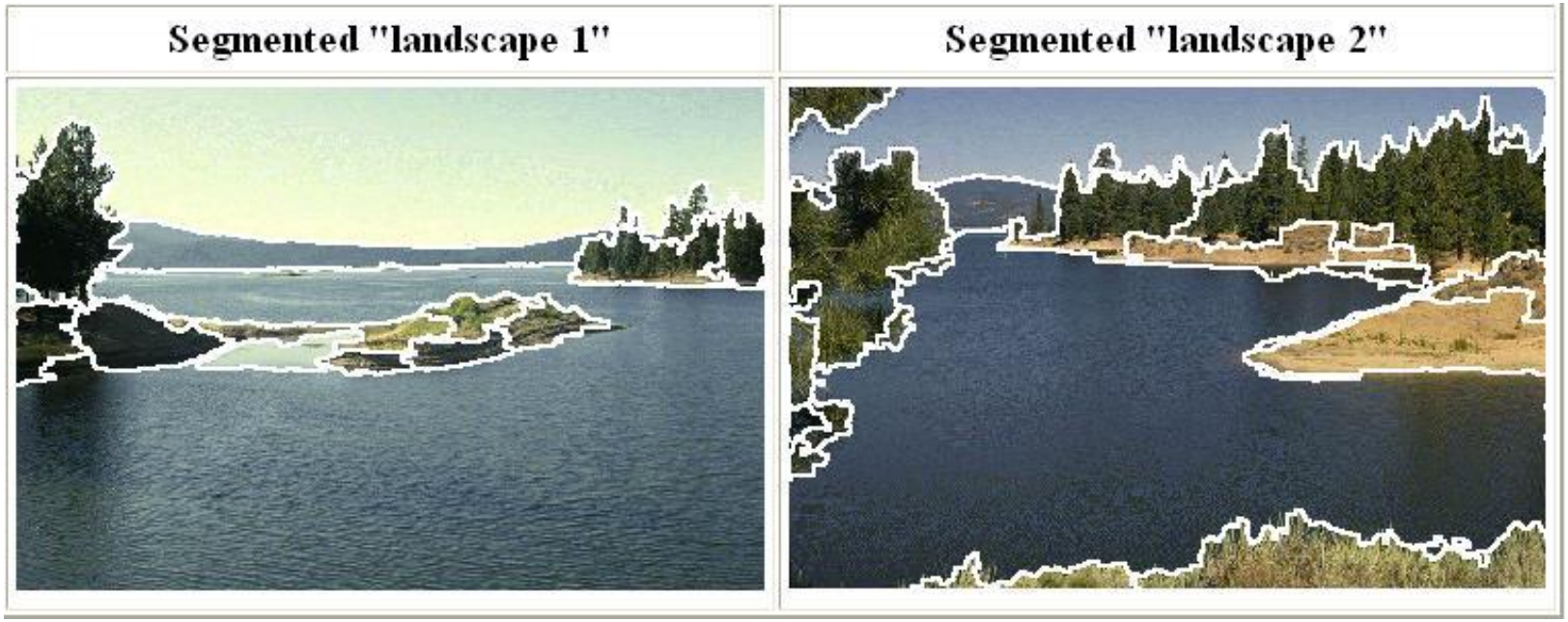
- Still fairly simple and efficient
- Model more complex distributions

Con:

- Need to know number of components in advance — hard to know unless you're looking at the data yourself!

Mean-Shift Segmentation

- An advanced and versatile technique for clustering-based segmentation

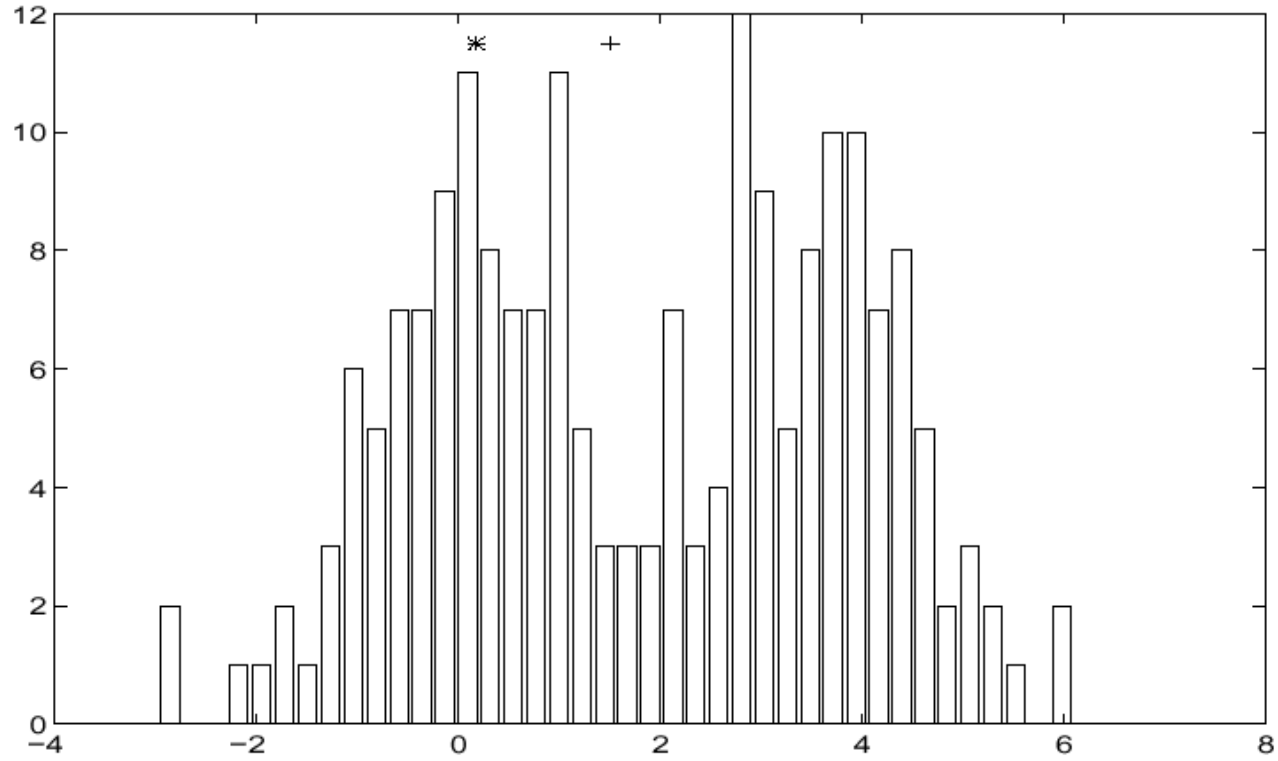


http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

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

Slide credit: Svetlana Lazebnik

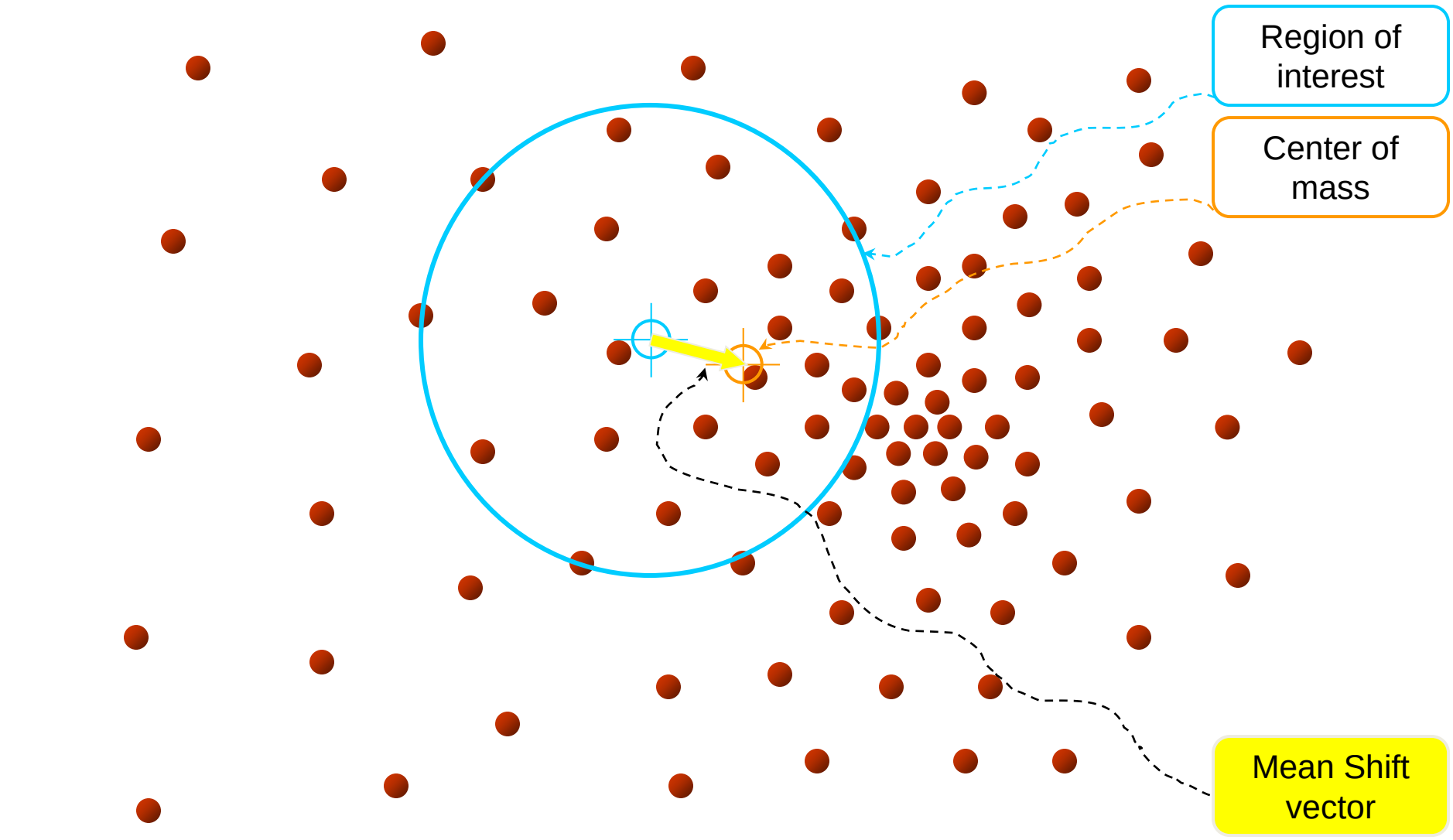
Mean-Shift Algorithm



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

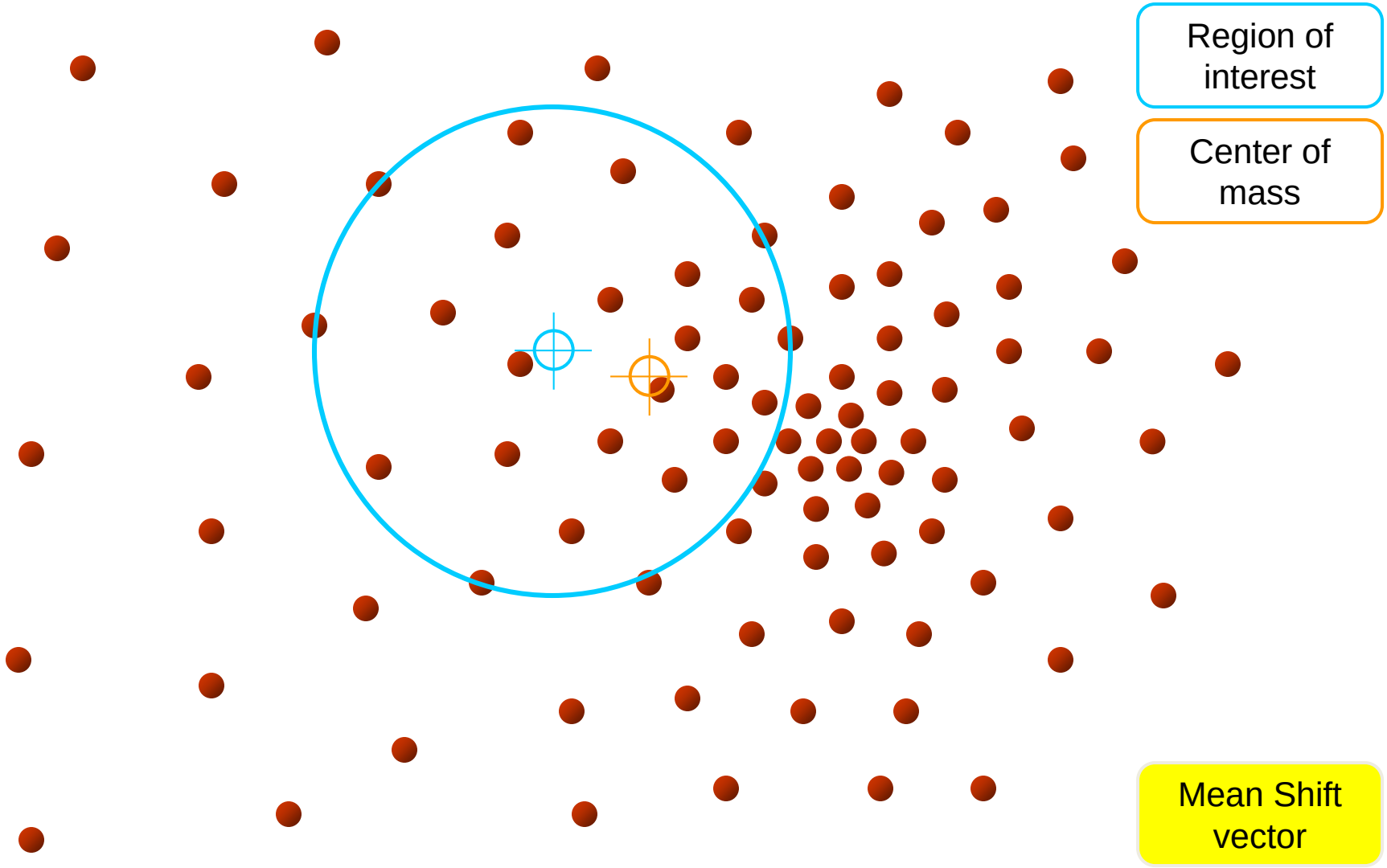
Slide credit: Steve Seitz

Mean-Shift



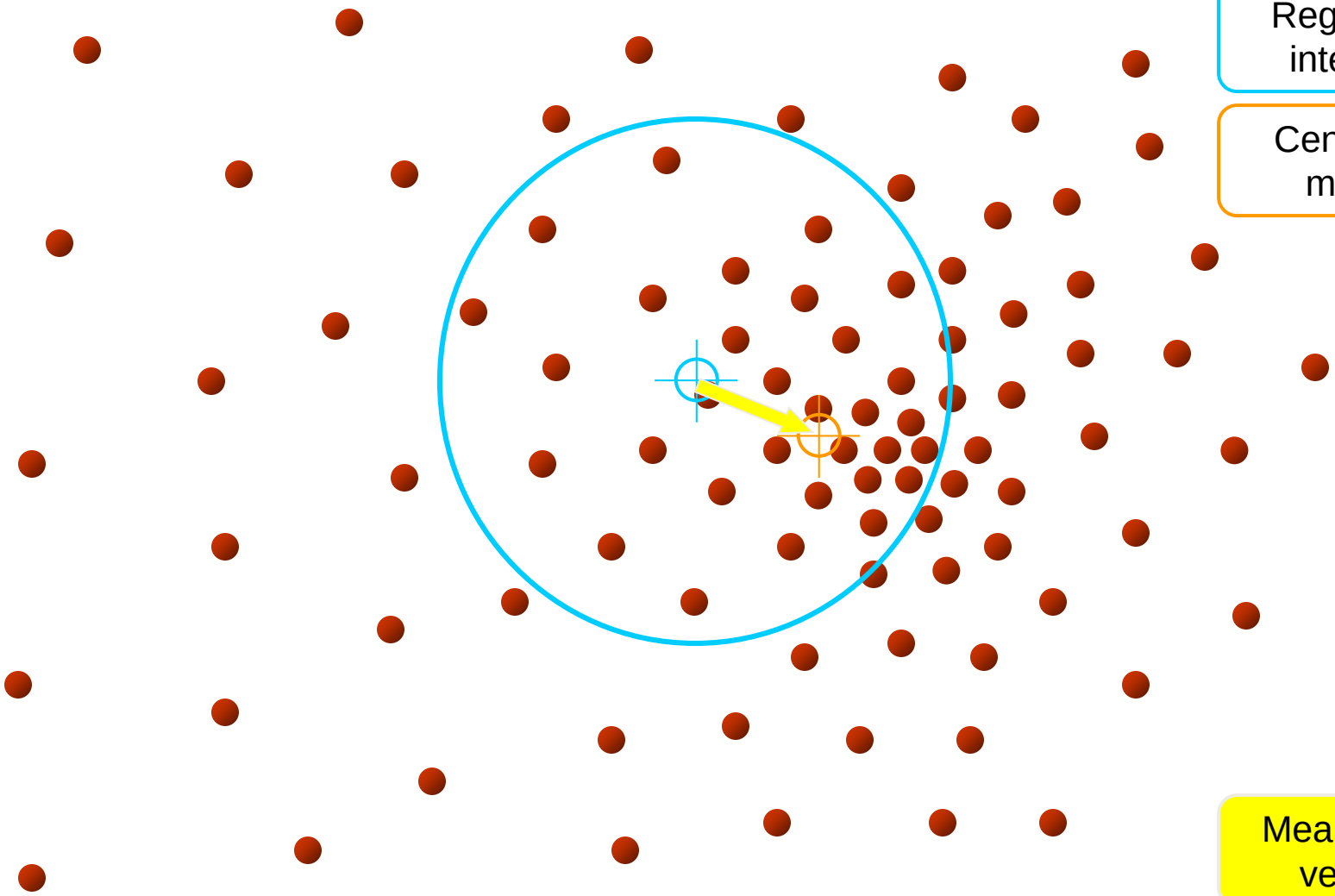
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



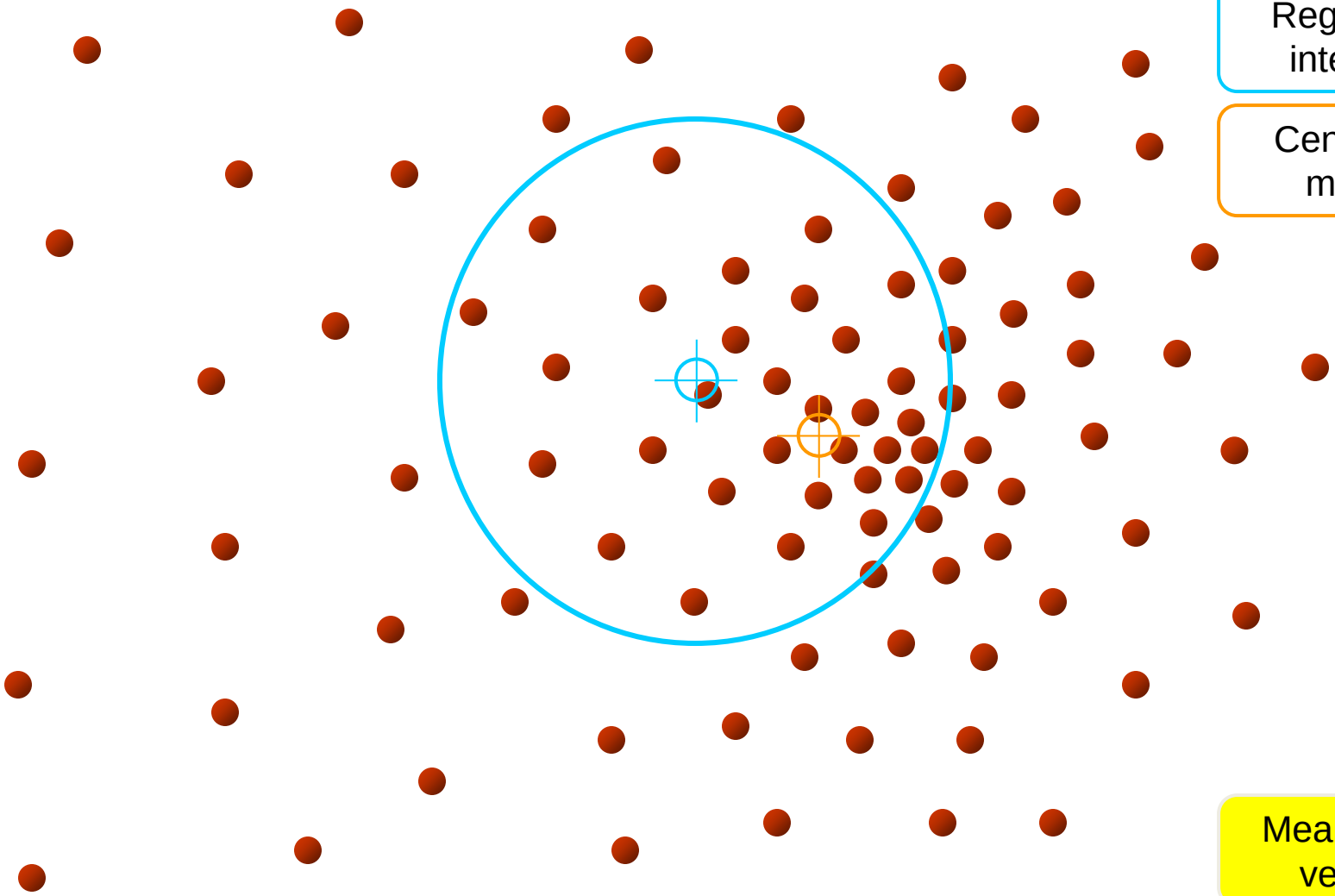
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



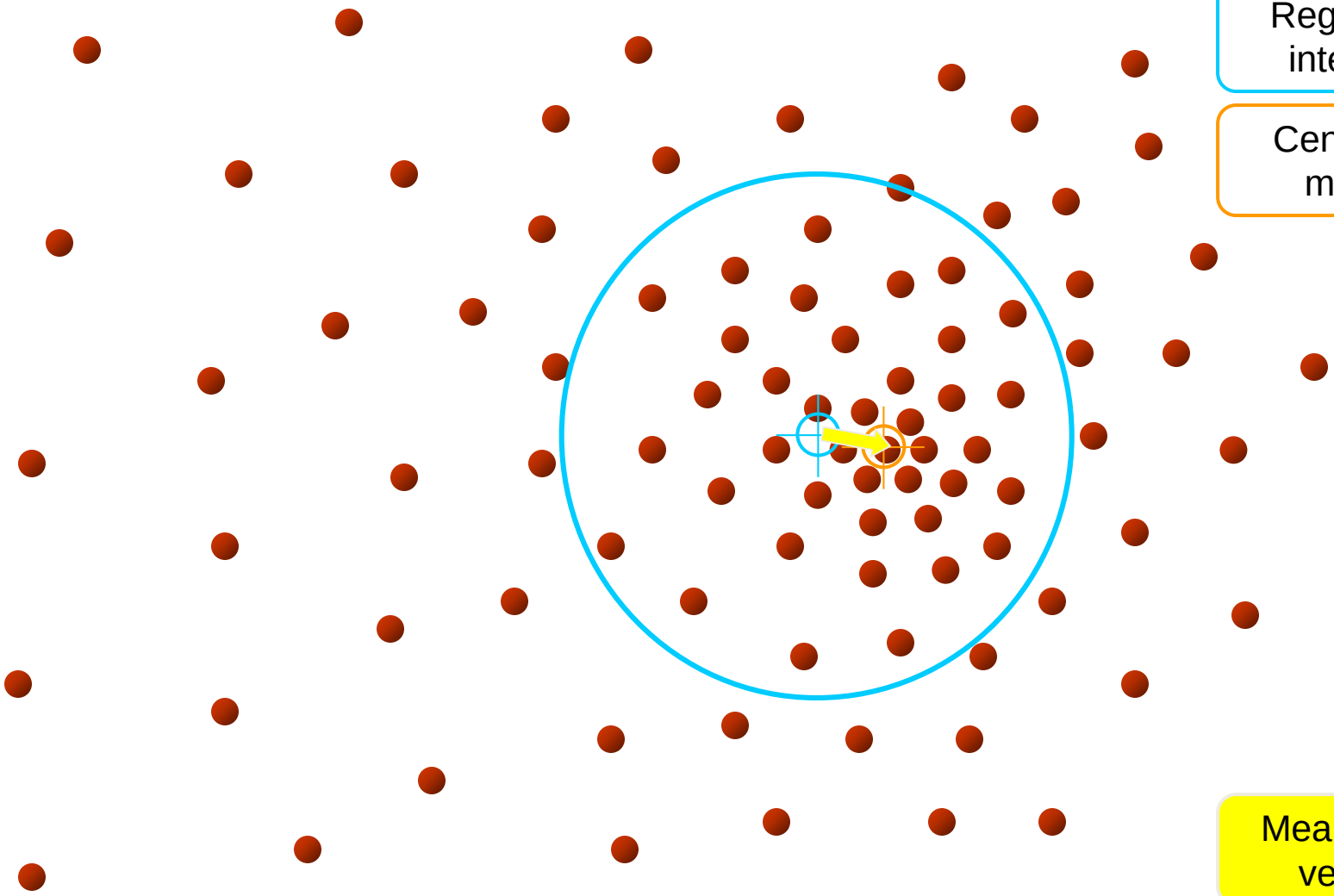
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



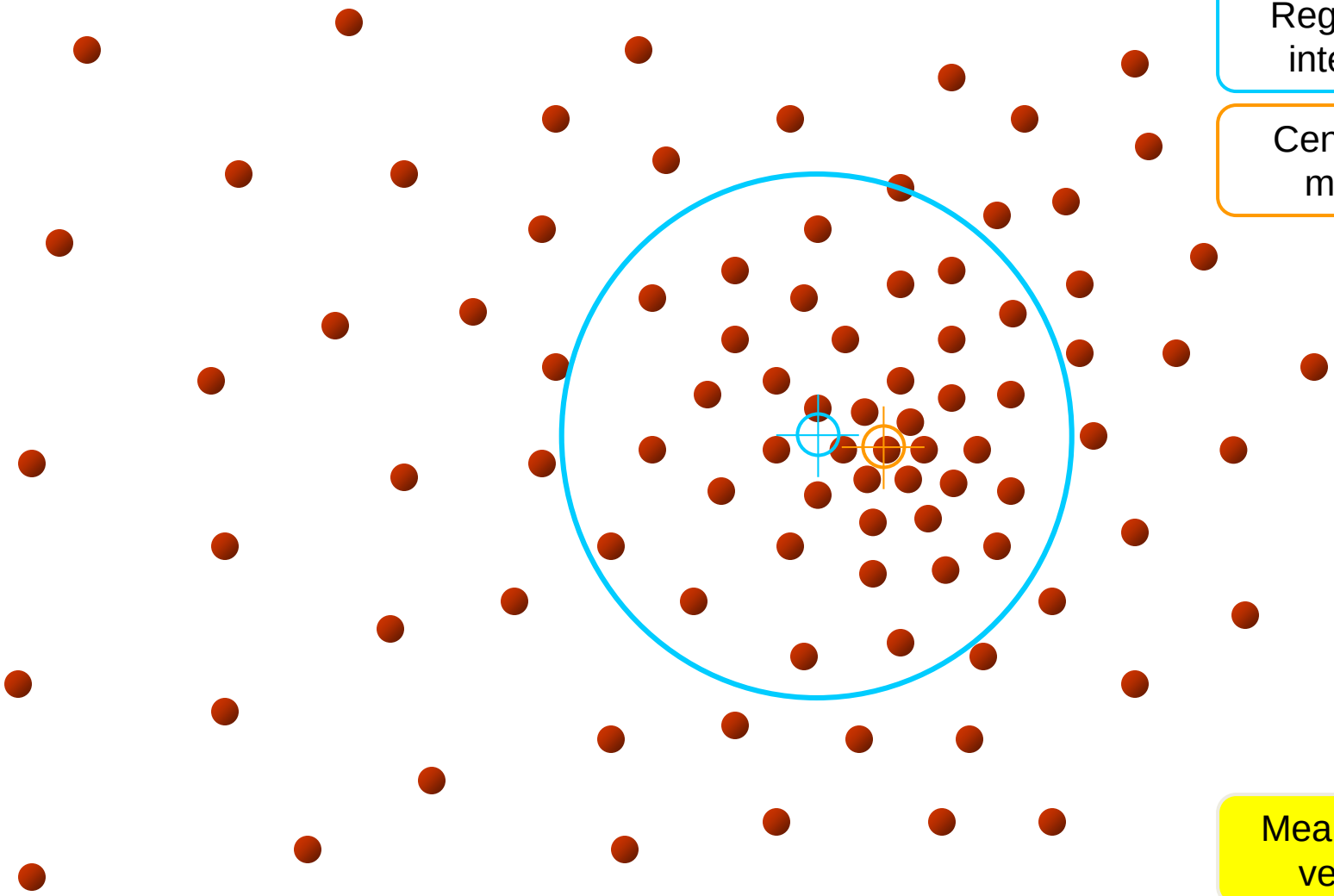
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



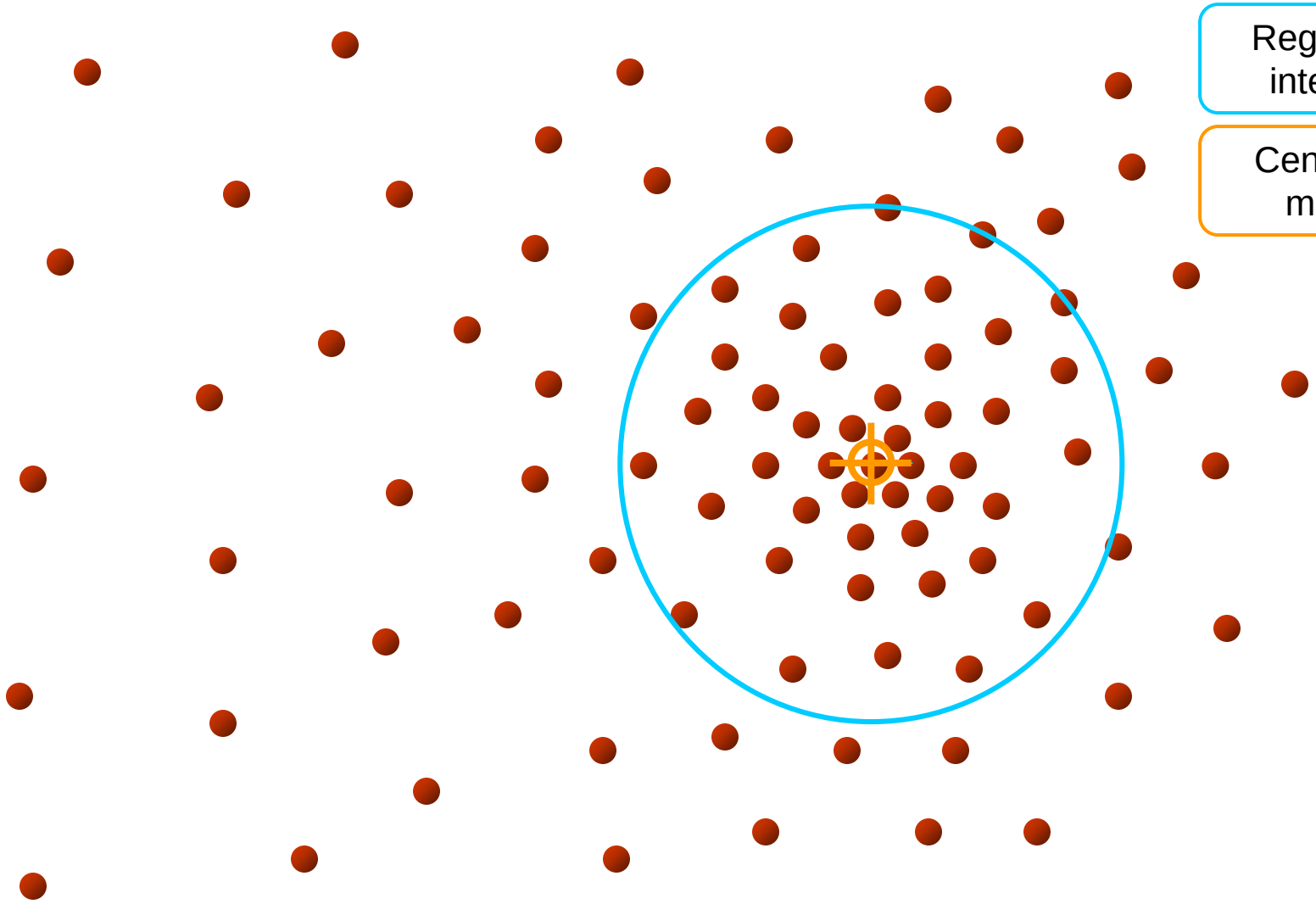
Slide by Y. Ukrainitz & B. Sarel

Mean-Shift



Slide by Y. Ukrainitz & B. Sarel

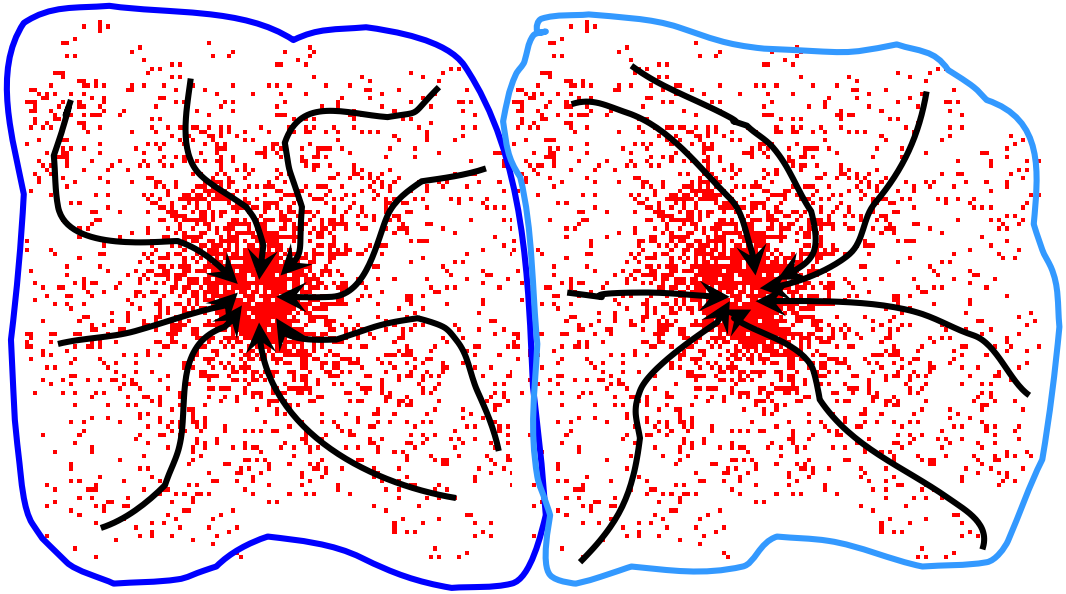
Mean-Shift



Slide by Y. Ukrainitz & B. Sarel

Mean-Shift Clustering

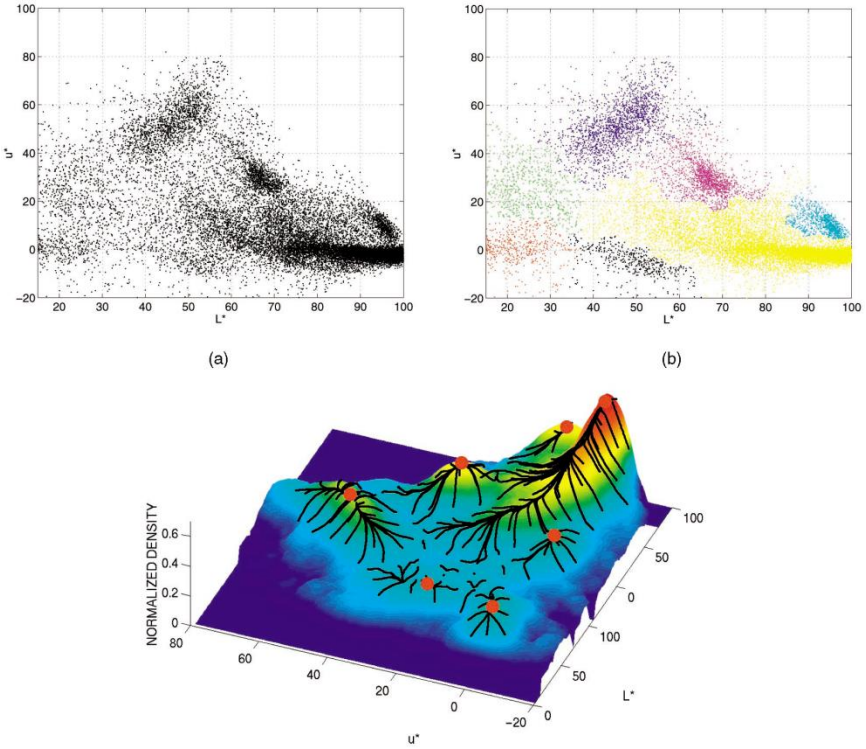
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Slide by Y. Ukrainitz & B. Sarel

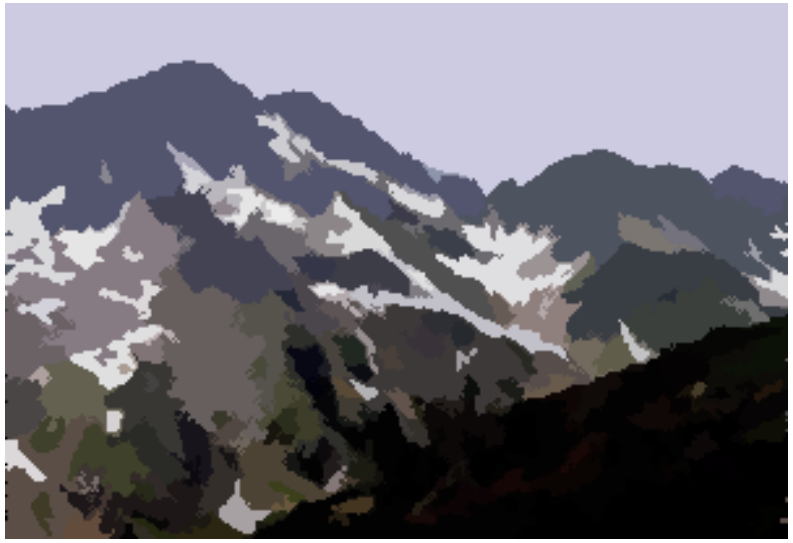
Mean-Shift Clustering/Segmentation

- 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



Slide credit: Svetlana Lazebnik

Mean-Shift Segmentation Results



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

Slide credit: Svetlana Lazebnik

More Results

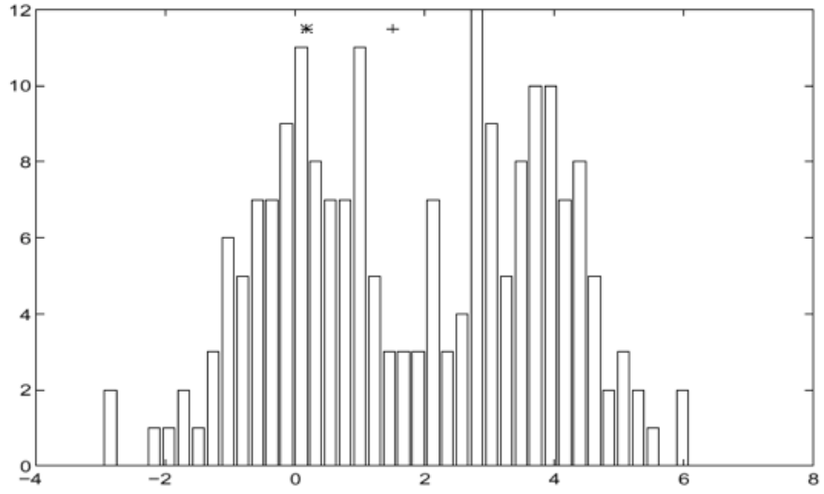
Slide credit: Svetlana Lazebnik

More Results

Mean Shift

Pro:

- No number of clusters assumption
- Handle unusual distributions
- Simple



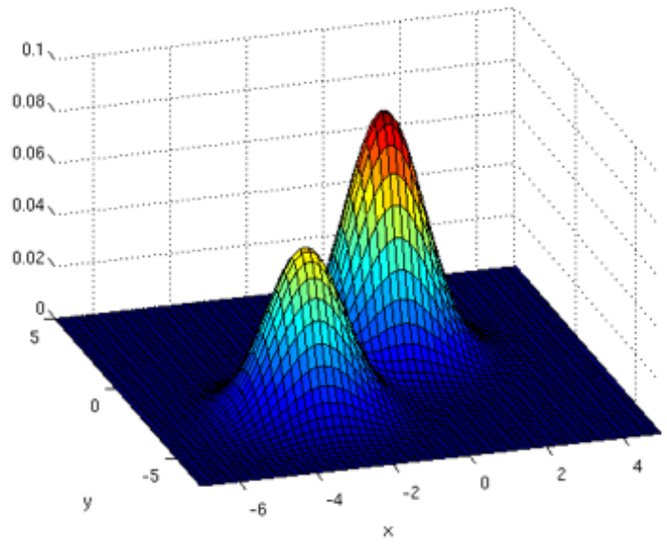
Con:

- Choice of window size
- Can be somewhat expensive

Clustering

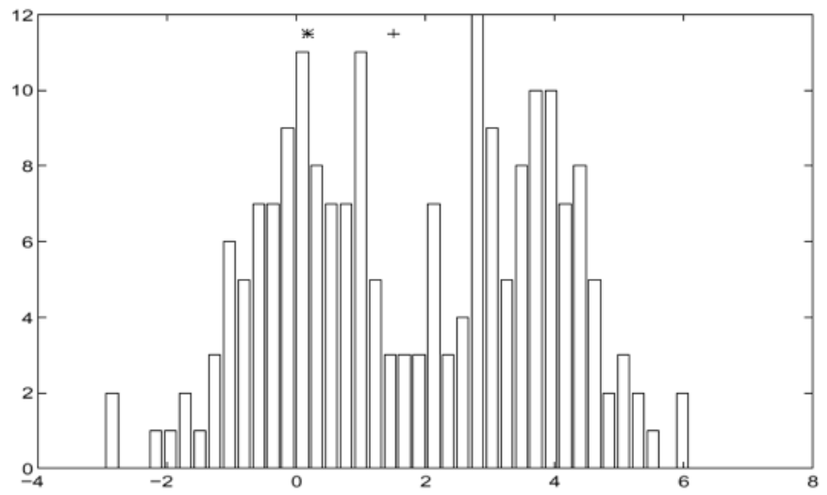
Pro:

- Generally simple
- Can handle most data distributions with sufficient effort.



Con:

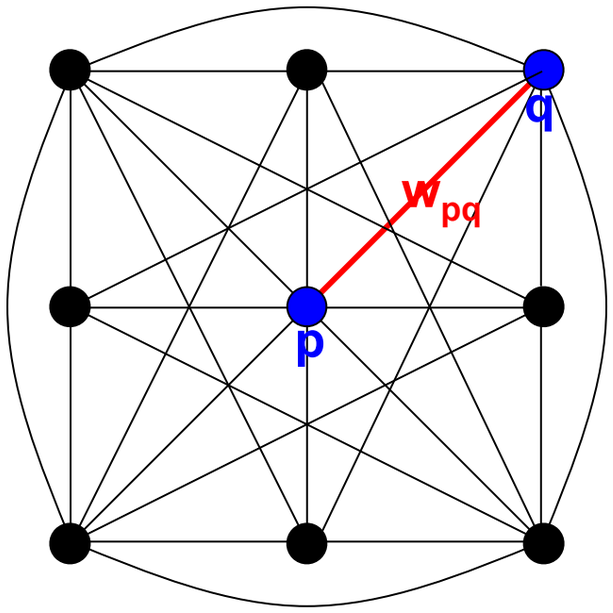
- Hard to capture global structure
- Performance is limited by simplicity

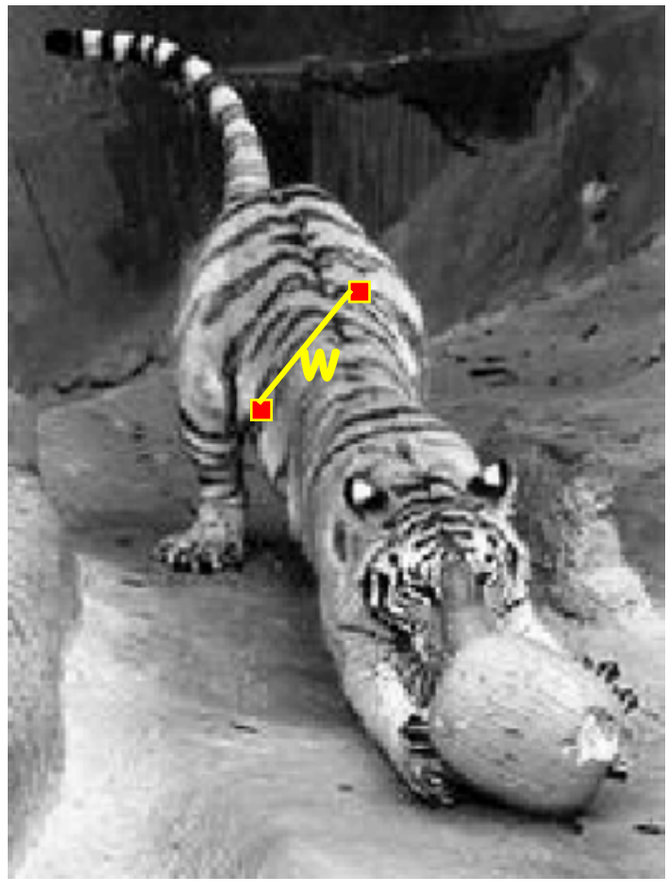


Outline

1. Segmentation as clustering
2. Graph-based segmentation
    1. General Properties
    2. Spectral Clustering
    3. Min Cuts
    4. Normalized Cuts
3. Segmentation as energy minimization

Images as Graphs





- Node (vertex) for every pixel
- Edge between pairs of pixels, (p,q)
- Affinity weight w_{pq} for each edge
  - w_{pq} measures similarity
  - Similarity is inversely proportional to difference (in color and position…)

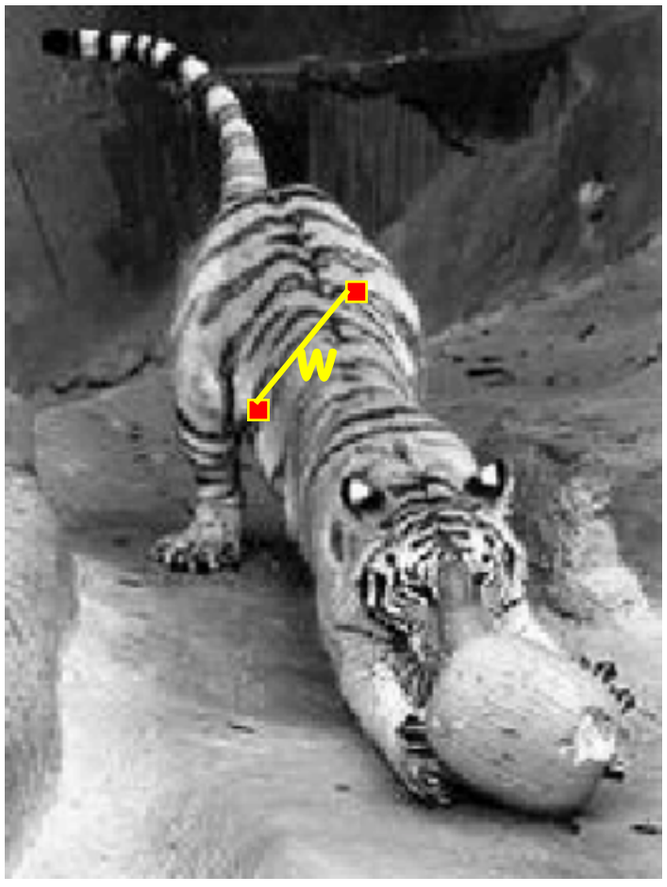
slide credit: Steve Seitz

Images as Graphs

Which edges to include?

Fully connected:

- Captures all pairwise similarities
- Infeasible for most images



Neighboring pixels:

- Very fast to compute
- Only captures very local interactions

Local neighborhood:

- Reasonably fast, graph still very sparse
- Good tradeoff

Measuring Affinity

- In general: $aff(x, y) = \exp\left(-\frac{1}{2\sigma_d^2}\|f(x) - f(y)\|^2\right)$

- Examples:
  - Distance: $f(x) = location(x)$
  - Intensity: $f(x) = intensity(x)$
  - Color: $f(x) = color(x)$
  - Texture: $f(x) = filterbank(x)$

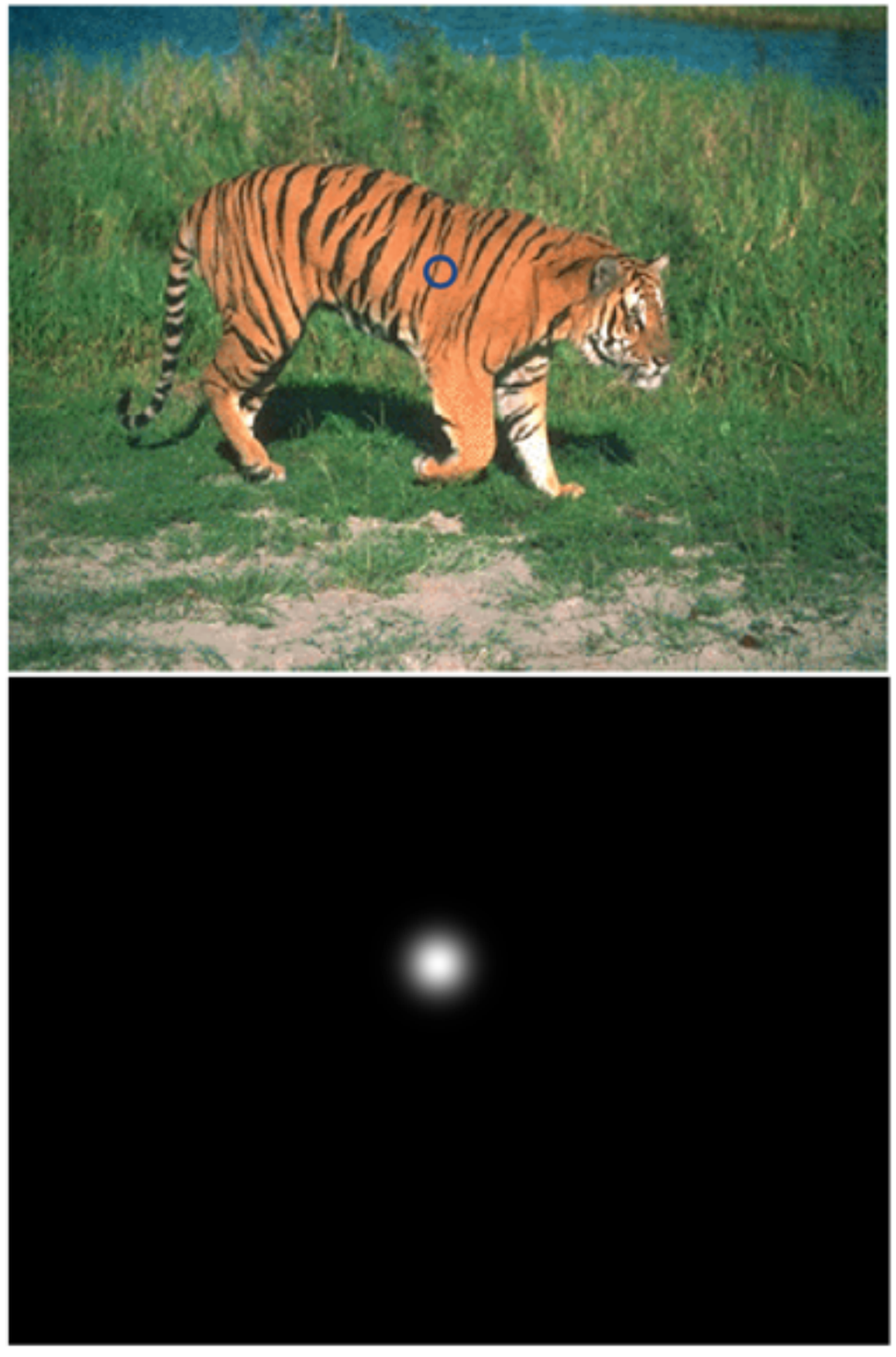
- Note: Can also modify distance metric

slide credit: Forsyth & Ponce

Measuring Affinity

Distance:

$$f(x) = location(x)$$

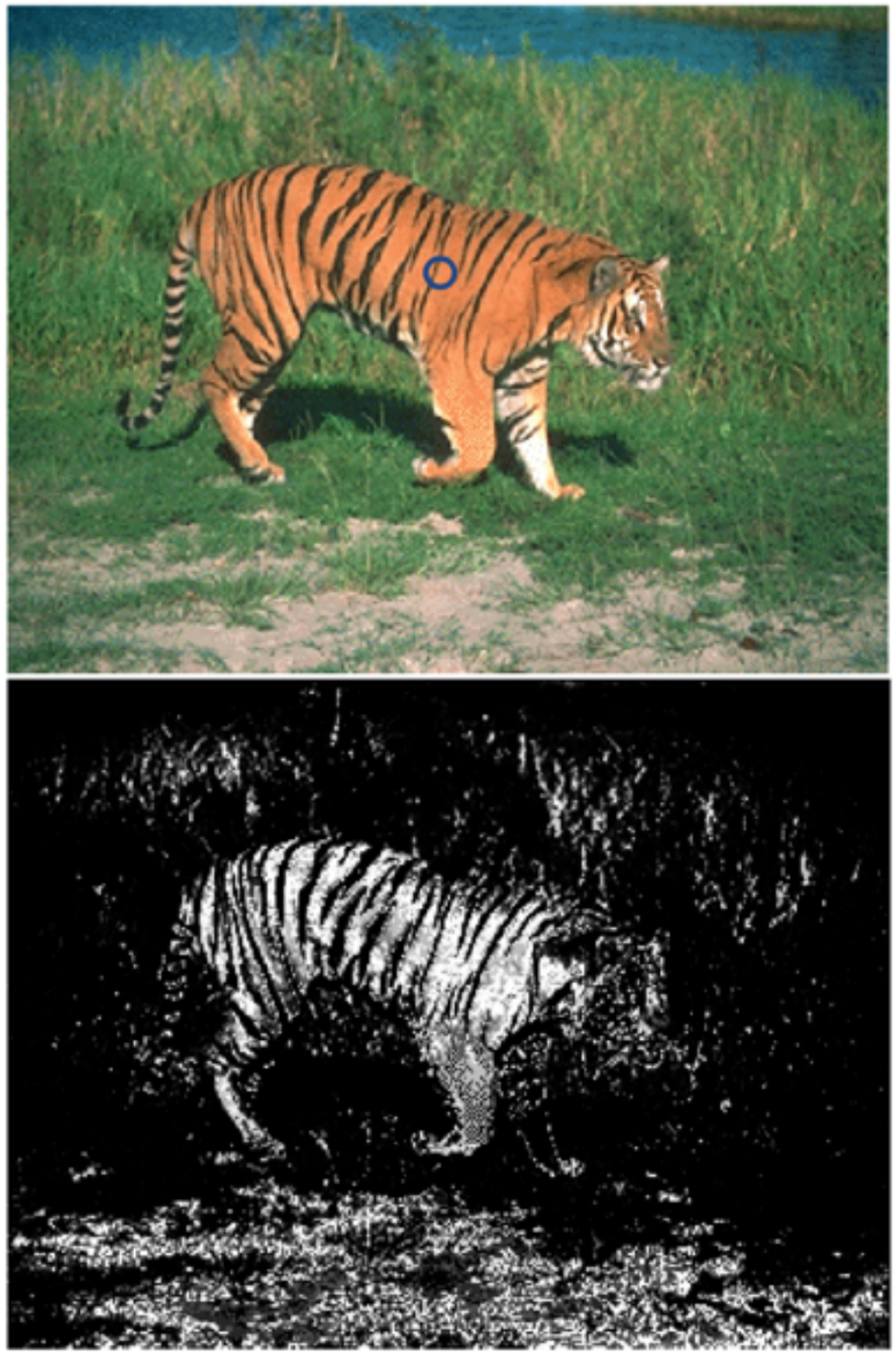


slide credit: Forsyth & Ponce

Measuring Affinity

Intensity:

$$f(x) = intensity(x)$$



slide credit: Forsyth & Ponce

Measuring Affinity

Color:

$$f(x) = color(x)$$

slide credit: Forsyth & Ponce

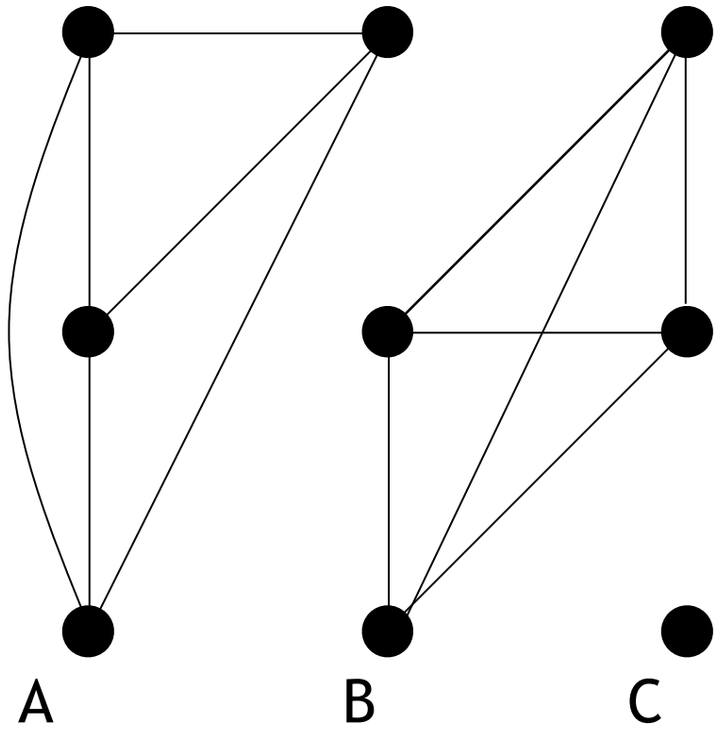
Measuring Affinity

Texture:

$$f(x) = filterbank(x)$$

slide credit: Forsyth & Ponce

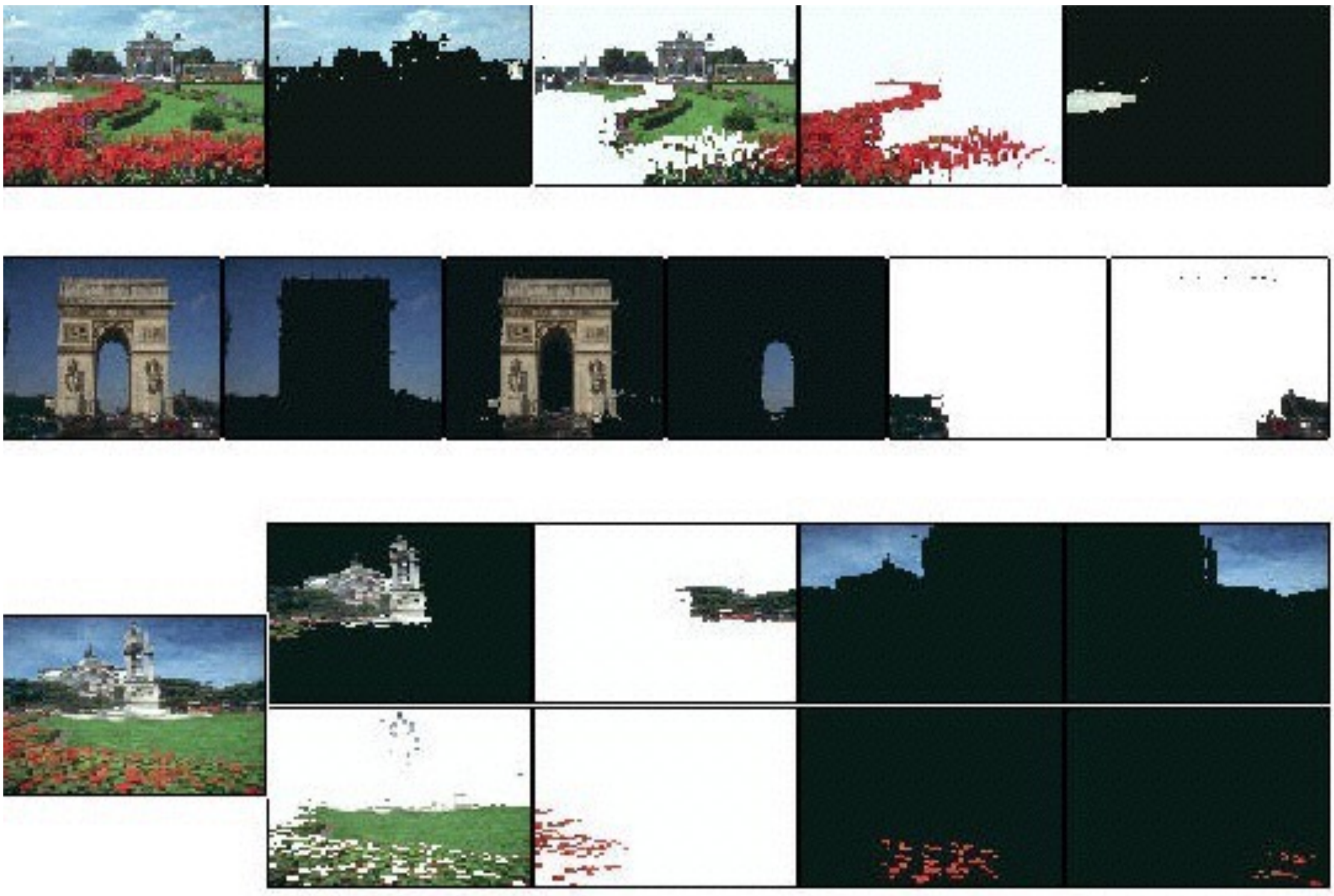
Segmentation as Graph Cuts



- Break Graph into Segments
  - Delete links that cross between segments
  - Easiest to break links that have low similarity (low weight)
    - Similar pixels should be in the same segments
    - Dissimilar pixels should be in different segments

slide credit: Steve Seitz

NCuts examples



NCuts examples

NCuts Pro and Con

- Pro
  - Flexible to choice of affinity matrix
  - Generally works better than other methods we've seen so far

- Con
  - Can be expensive, especially with many cuts.
  - Bias toward balanced partitions
  - Constrained by affinity matrix model

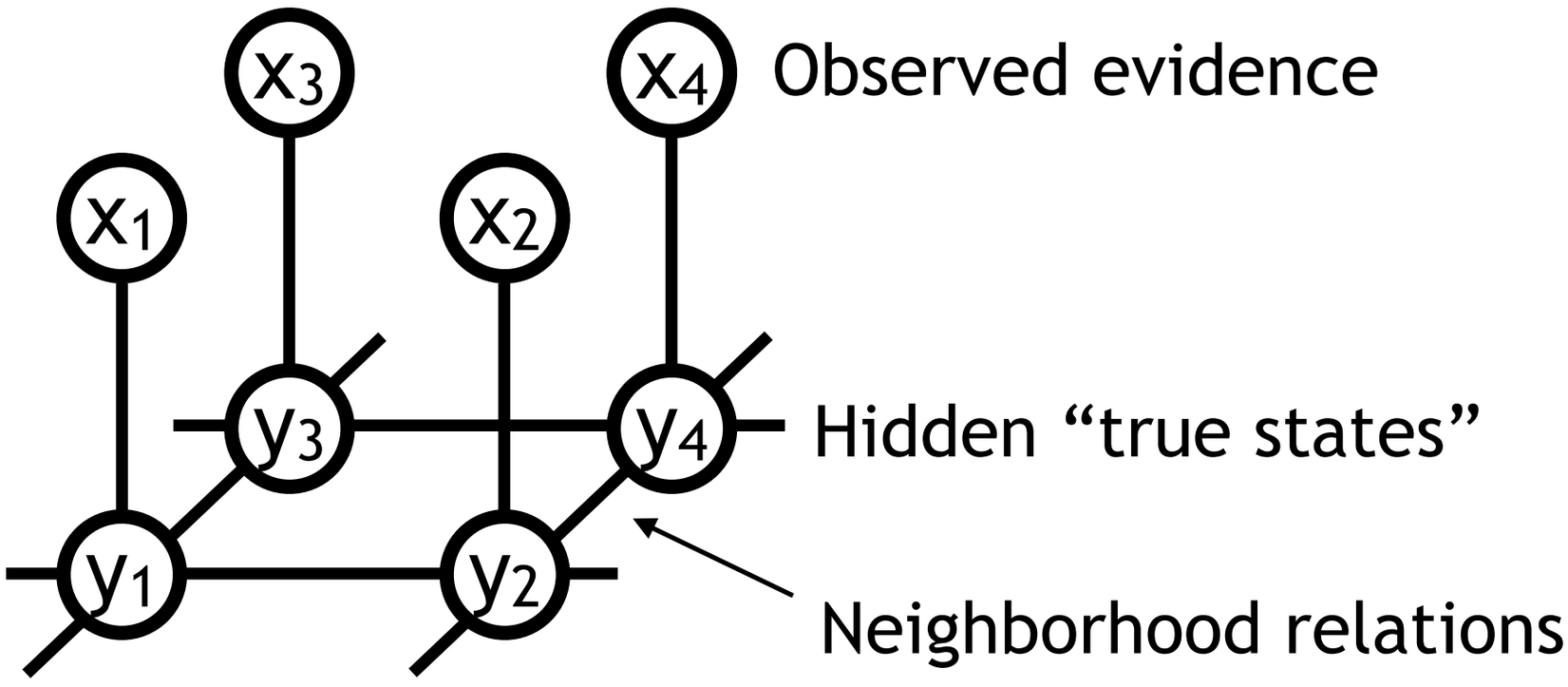
Slide source: Kristen Grauman

Outline

1. Segmentation as clustering
2. Graph-based segmentation
3. Segmentation as energy minimization
    1. MRFs + CRFs
    2. Segmentation with CRFs
    3. GrabCut

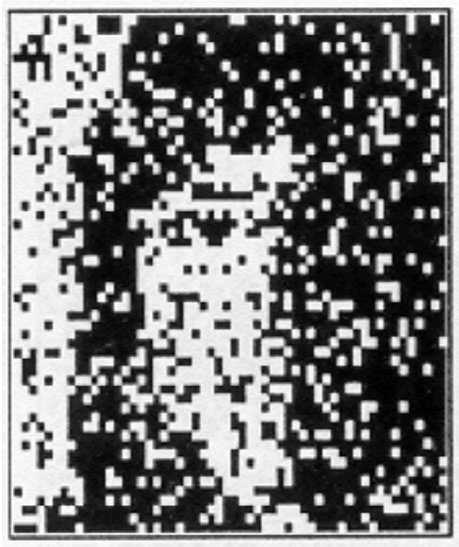
Conditional Random Fields (CRFs)

- Rich probabilistic model for images
- Built in local, modular way
  - Get global effects from only learning/modeling local ones
- After conditioning, get a Markov Random Field (MRF)

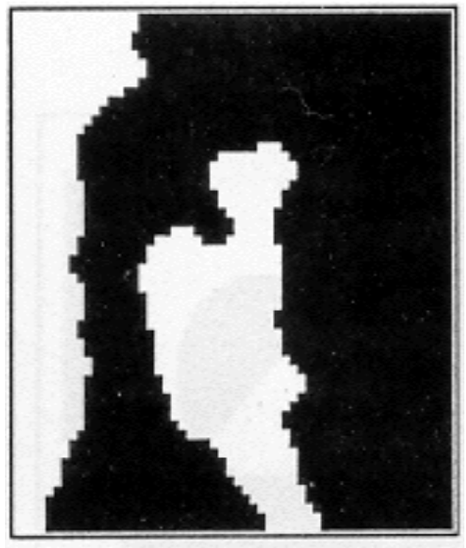


Pixels as CRF Nodes

Original image



Degraded image



Reconstruction from MRF modeling pixel neighborhood statistics

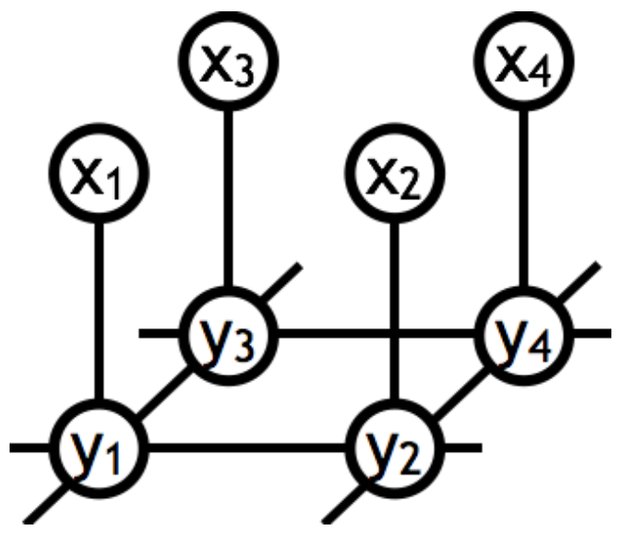


Image source: Bastian Liebe

CRF Probability

$$P(x, y) = \frac{1}{Z} \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(y_i, y_j)$$

Image

Scene

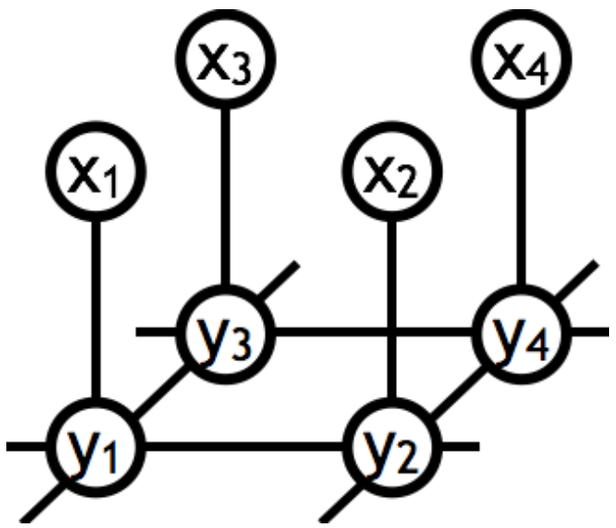
Partition Function

Image-scene compatibility function

Local observations

Scene-scene compatibility function

Neighboring scene nodes



Energy Formulation

$$P(x, y) = \frac{1}{Z} \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(y_i, y_j)$$

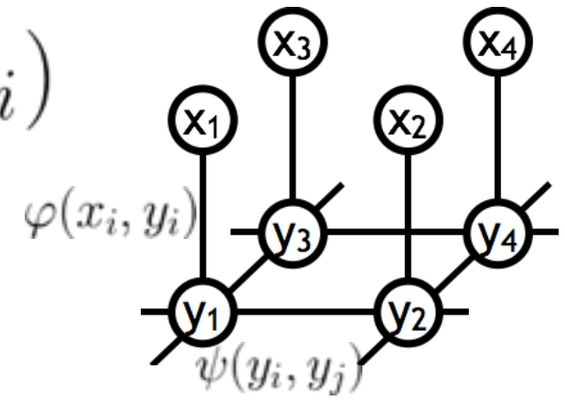
take logs, drop Z

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

- We call E an *energy function*
  - named from free-energy problems in statistical mechanics
- Individual terms are *potentials*
- *Note: Derived this way, it's energy maximization. Be careful and check each formulation individually.*

Energy Formulation

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



- Unary potentials φ
  - Local information about each pixel
  - e.g. how likely a pixel/patch belongs to a certain class

- Pairwise potentials ψ
  - Neighborhood information, enforces consistency
  - e.g. how different a pixel is from its neighbor in appearance

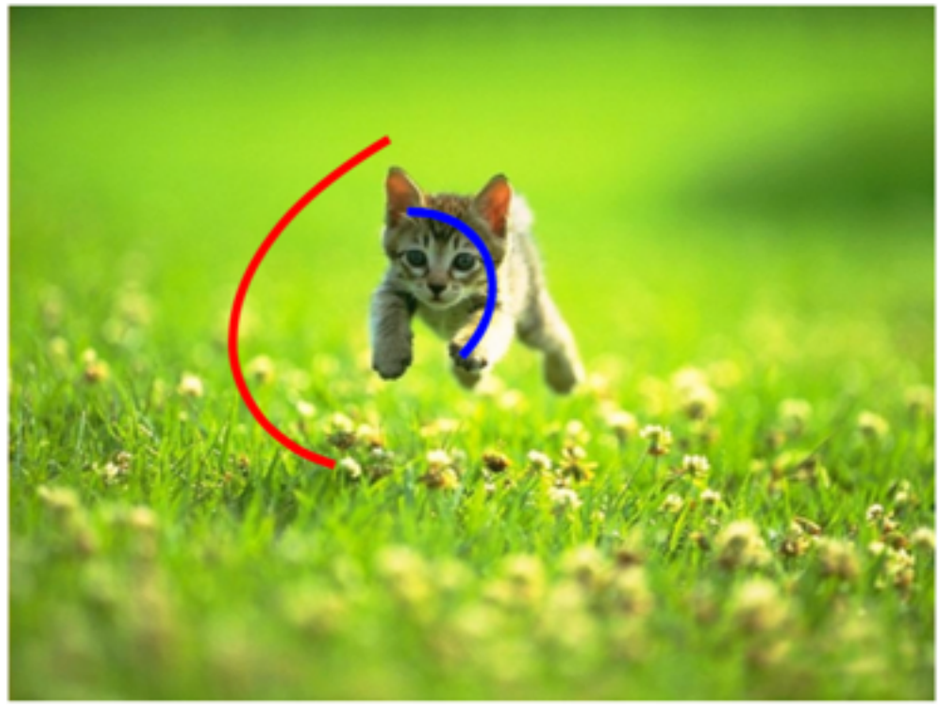
slide credit: Bastian Liebe

CRF segmentation example

- Boykov and Jolly (2001)

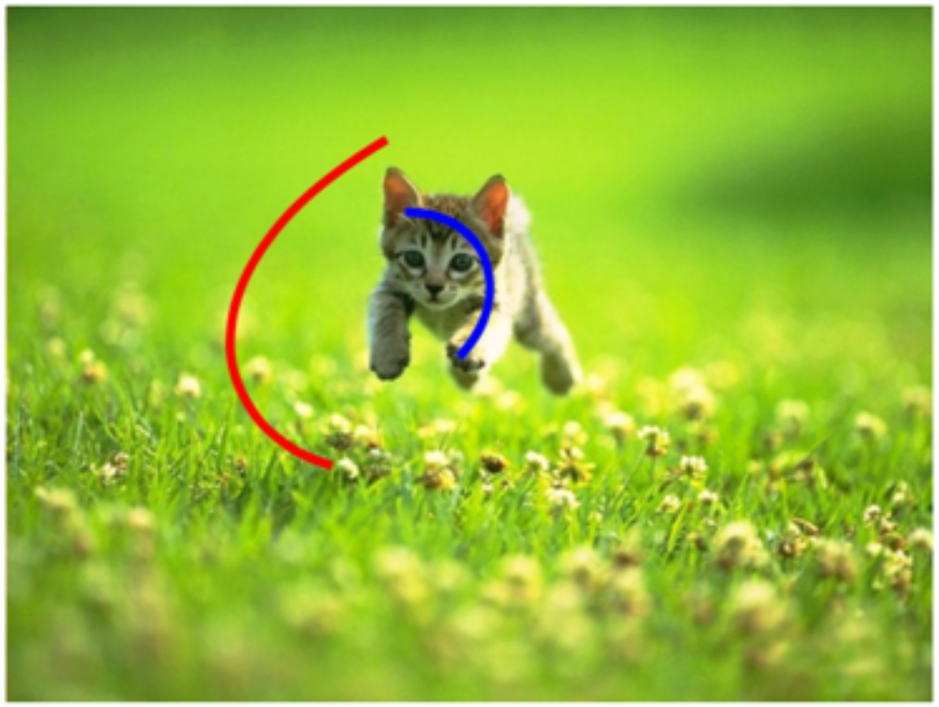
$$E(x,y) = \sum_i \varphi(x_i, y_i) + \sum_{i,j} \psi(y_i, y_i)$$

- Variables
  - x_i : Annotation (Input)
    - **Foreground**/**background**/empty
  - y_i : Binary variable
    - Foreground/background
- Unary term
  - $\varphi(x_i, y_i) = K[x_i \neq y_i]$
  - Penalty for disregarding annotation
- Pairwise term
  - $\psi(y_i, y_j) = [y_i \neq y_j] w_{ij}$
  - Encourage smooth annotations
  - w_{ij} is affinity between pixels *i* and *j*

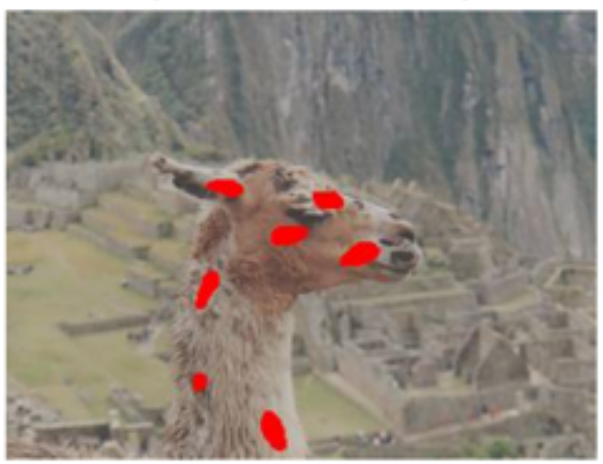
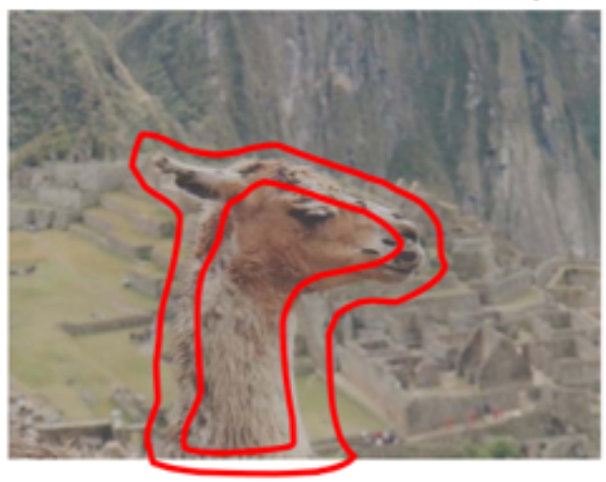
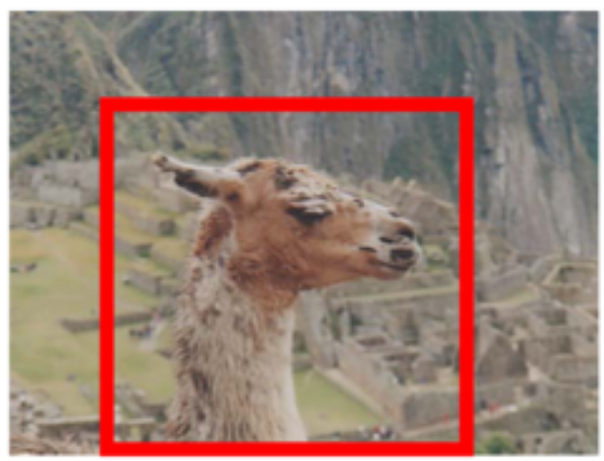


Solving Efficiently

- Grid structured random fields
  - Maxflow/mincut
  - Optimal for binary labeling
    - submodular energy functions
  - Boykov & Kolmogorov, “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision”, PAMI 2004
- Fully connected models
  - Efficient solution with convolution mean-field
  - Krähenbühl and Koltun, “Efficient Inference in Fully-Connected CRFs with Gaussian Edge Potentials”, NIPS 2011



GrabCut: Interactive Foreground Extraction

	Magic Wand (Adobe, 2002)	Intelligent Scissors Mortensen and Barrett (1995)	GrabCut
User Input			
Result	 Regions	 Boundary	 Regions & Boundary

GrabCut formulation

$$E(x, y, \theta, k) = \sum_i \varphi(x_i, y_i, \theta, k_i) + \sum_{ij} \psi(y_i, y_j, x_i, x_j)$$

- Variables
  - x_i: pixel
  - $y_i \in \{0,1\}$: foreground/background label {0,1}
  - $k_i \in \{0, \ldots, K\text{-}1\}$: GMM mixture component
  - θ: GMM model parameters
  - $\mathbf{I} = \{z_1, \ldots, z_m\}$: RGB image
- Unary Term $\varphi(x_i, y_i, \theta, k_i)$
  - -log of GMM probability
- Pairwise Term

$$\psi(y_i, y_j, x_i, x_j) = \gamma[y_i \neq y_j] \exp\left(-\beta \|x_i - x_j\|^2\right)$$

GrabCut - Iterative Optimization

1. Initialize Mixture Models based on user annotation
2. Assign GMM components

$$k_i = \arg\min \varphi(x_i, y_i, \theta, k_i)$$

3. Learn GMM parameters

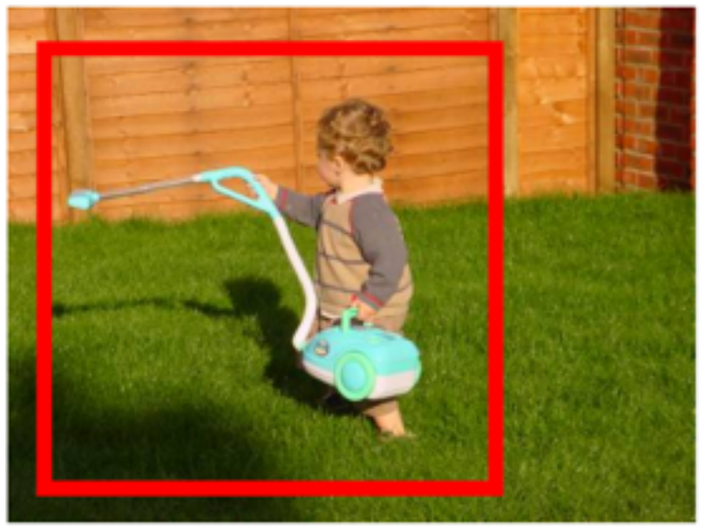
$$\theta = \arg\min \sum_i \varphi(x_i, y_i, \theta, k_i)$$

4. Estimate segmentations (mincut)

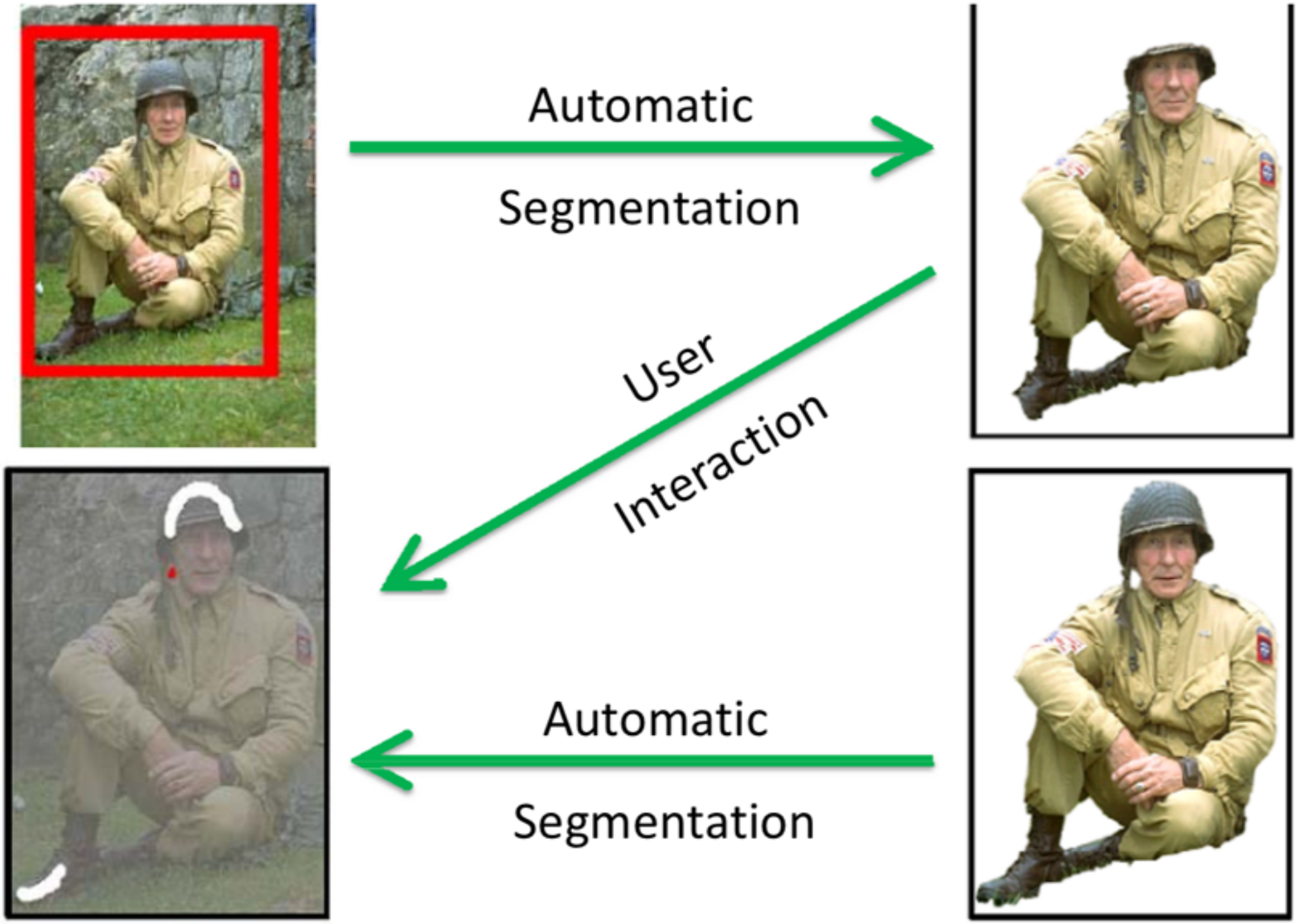
$$y = \arg\min E(x, y, \theta, k)$$

5. Repeat 2-4 until convergence

GrabCut results



Further editing with GrabCut



Summary: Graph Cuts with CRFs

- Pros
  - Very powerful, get global results by defining local interactions
  - Very general
  - Rather efficient
  - Becoming more or less standard for many segmentation problems (GrabCut was 2004!)
- Cons
  - Only works for sub modular energy functions (binary)
  - Only approximate algorithms work for multi-label case

Extra: Improving Segmentation Efficiency

- Images contain a lot of pixels
  - Even efficient methods can be slow
- Efficiency trick: Superpixels
  - Group together similar pixels
  - Cheap and local over segmentation
  - Must be high precision!
  - Many methods exist, try several.
- Another trick: Resize the image
  - Do segmentation in lower-res version, then scale back up to high-res.