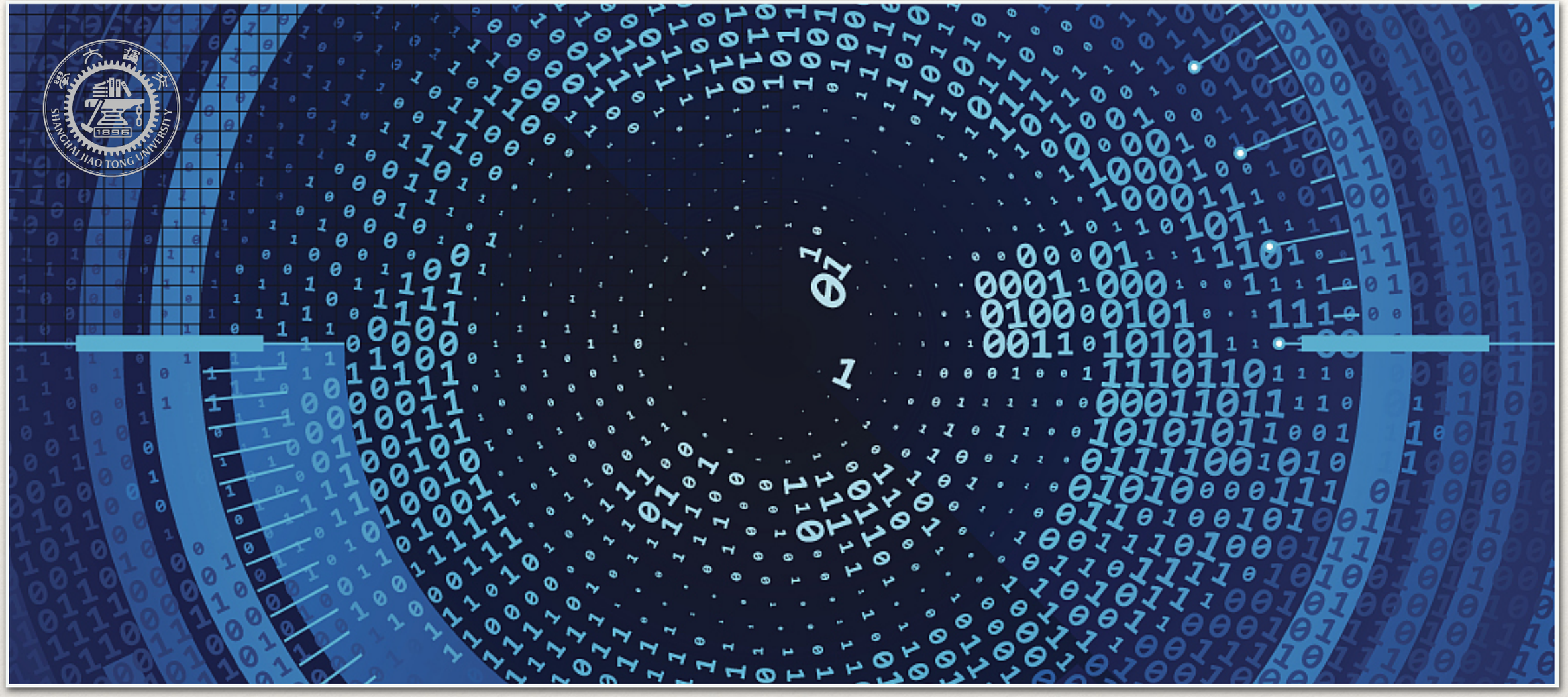


Shanghai Jiao Tong University

Computer Vision

Instructor: Xu Zhao

Class No.: C032703 F032528

Spring 2020

Xu Zhao @ Shanghai Jiao Tong university

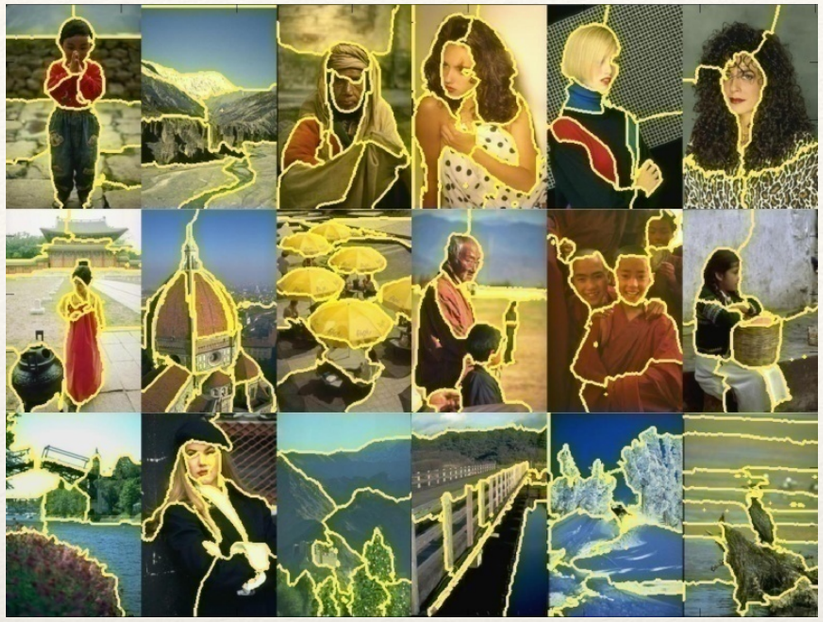
Lecture 9: Segmentation

Contents

- **Segmentation by clustering**
- **Deformable contours for segmentation**

Grouping in vision

- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image (video) parts
- Top down vs. bottom up segmentation
  - Top down: pixels belong together because they are from the same object
  - Bottom up: pixels belong together because they look similar
- Hard to measure success
  - What is interesting depends on the app



[Figure by J. Shi]

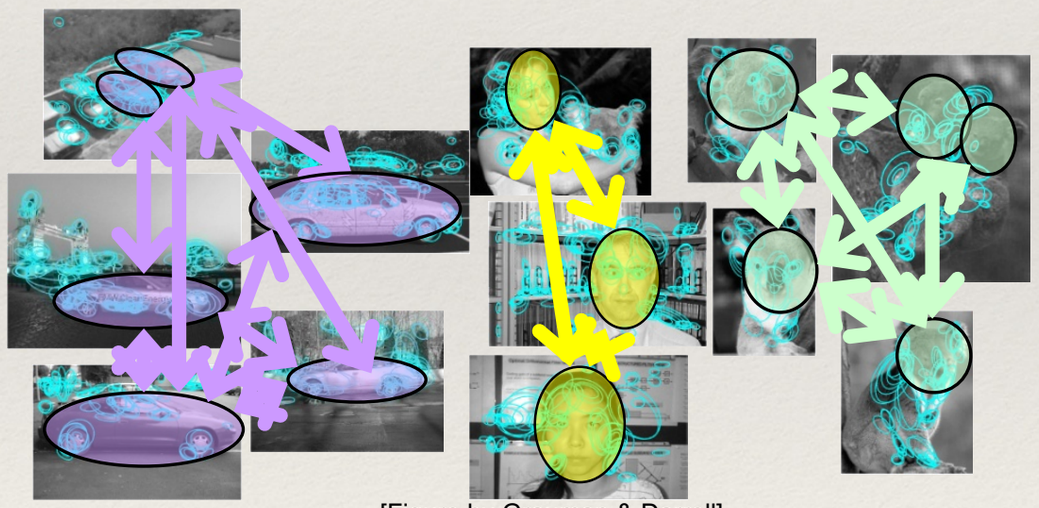
Determine image regions

[http://poseidon.csd.auth.gr/LAB_RESEARCH/Latest/imgs/SpeakDepVidIndex_img2.jpg]

Group video frames into shots

[Figure by Wang & Suter]

Figure-ground

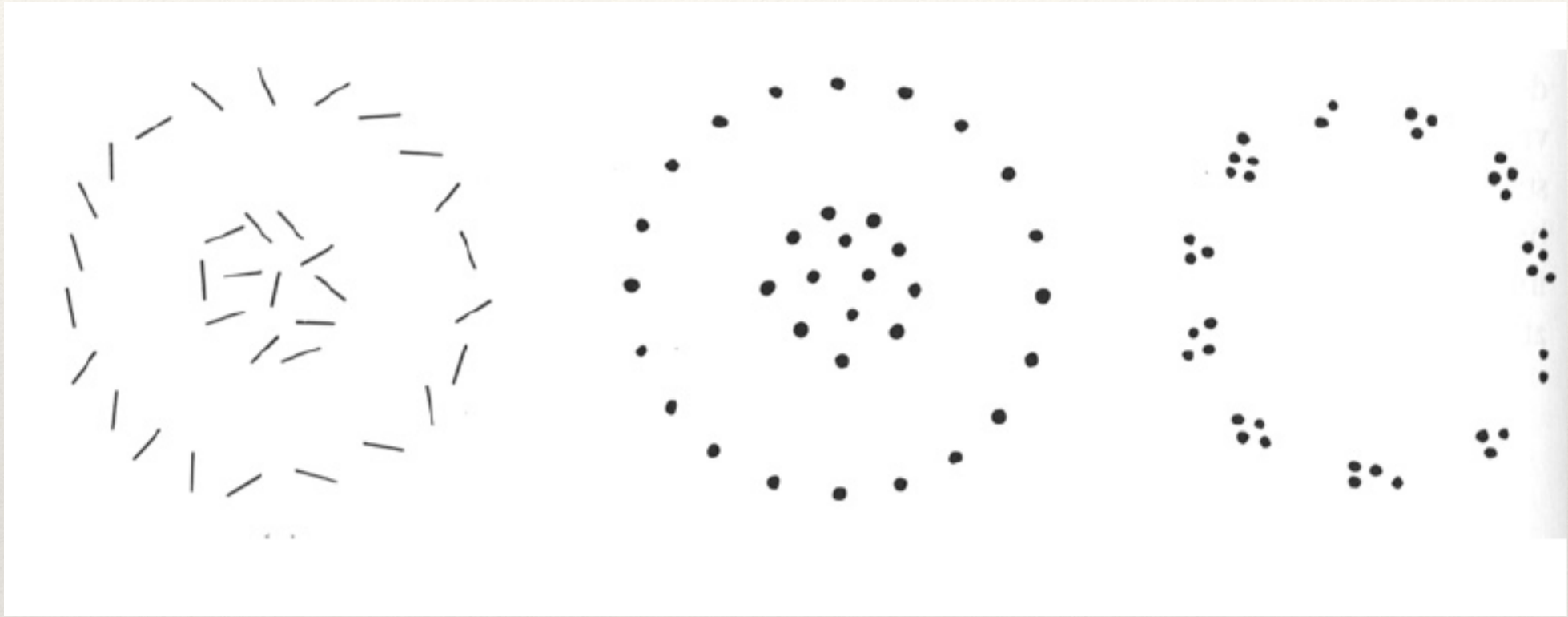


[Figure by Grauman & Darrell]

Object-level grouping

Slide credit: Kristen Grauman

What are meta-cues for grouping?



Slide credit: Kristen Grauman

Gestalt

- Gestalt: whole or group
  - Whole is greater than sum of its parts
    - Relationships among parts can yield new properties/ featuresHumans interpret image information collectively
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)
  - Humans interpret image information collectively

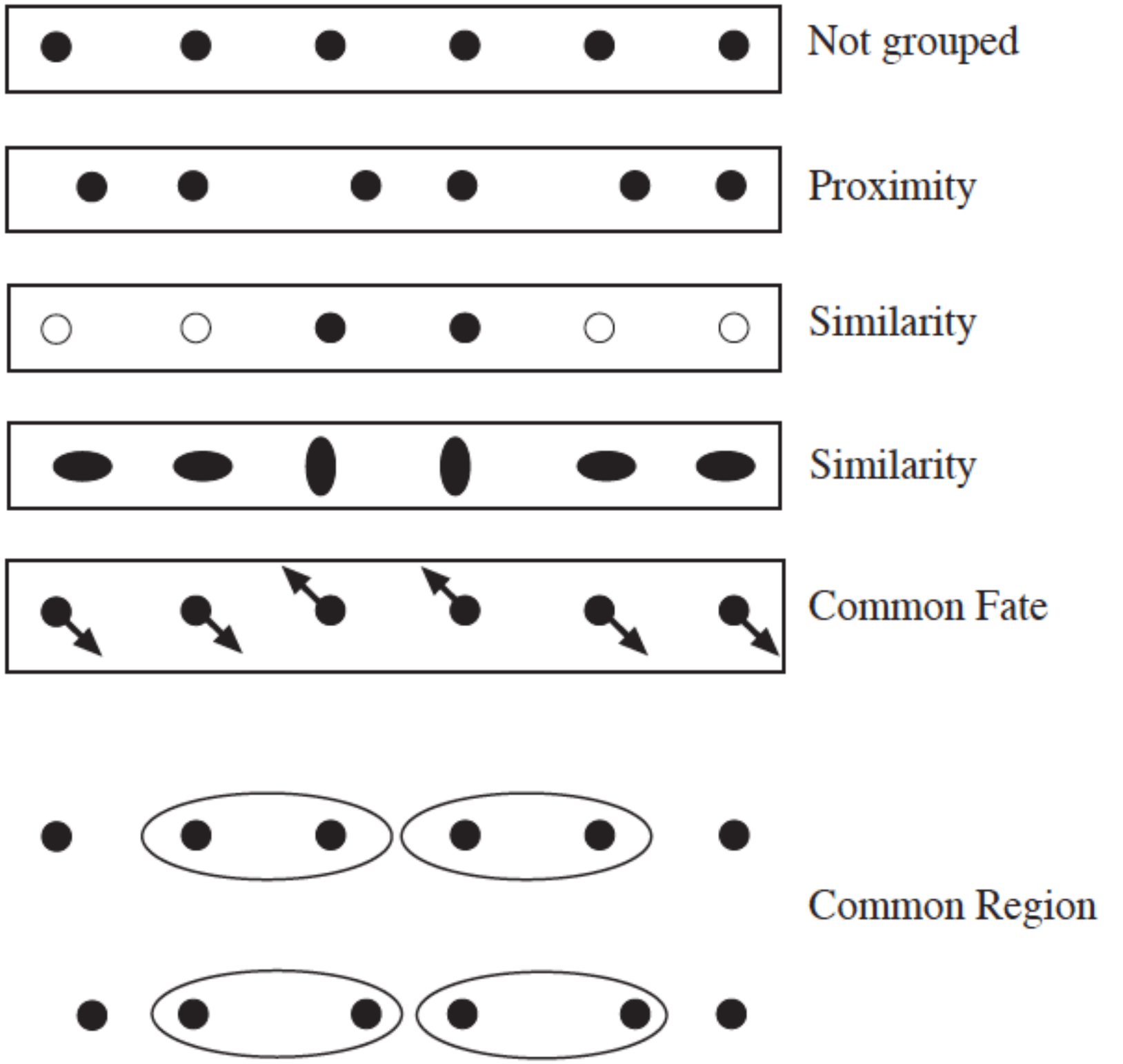
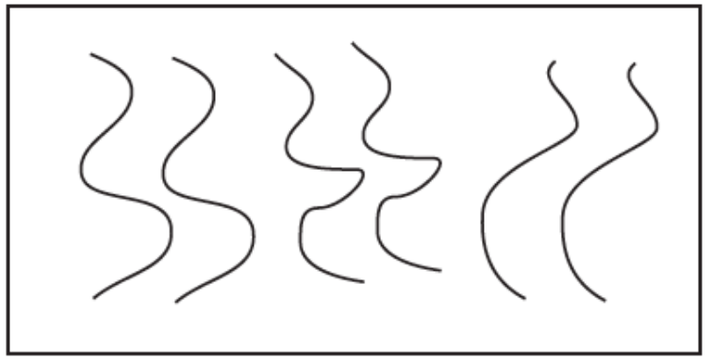
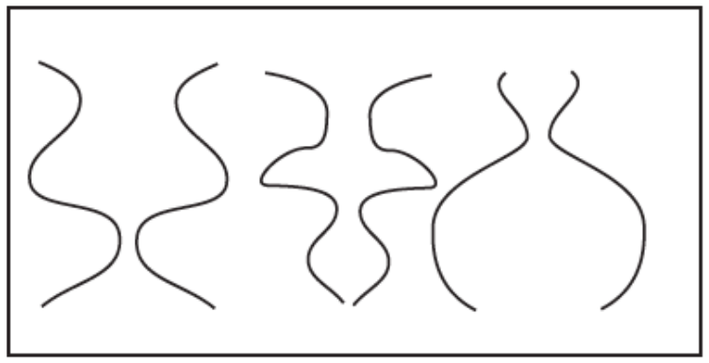


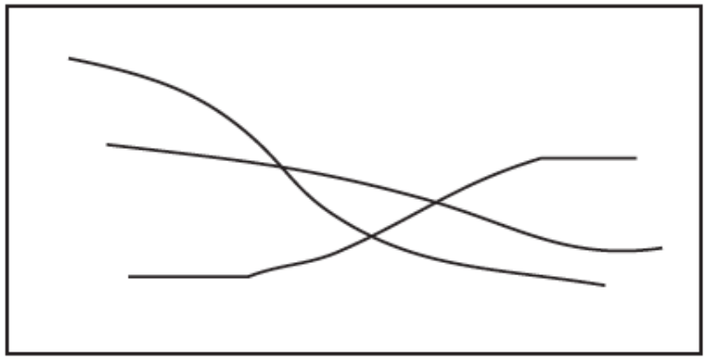
FIGURE 9.4: Examples of Gestalt factors that lead to grouping (which are described in greater detail in the text).



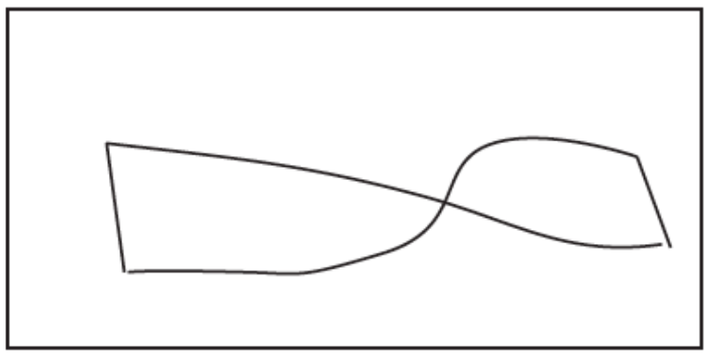
Parallelism



Symmetry



Continuity



Closure

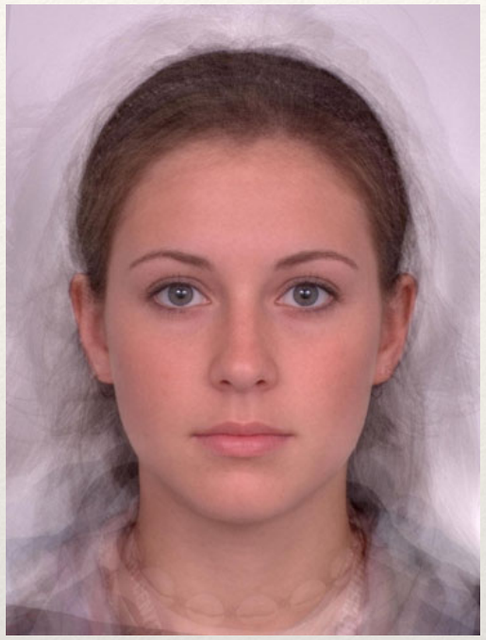
FIGURE 9.5: Examples of Gestalt factors that lead to grouping (which are described in greater detail in the text).

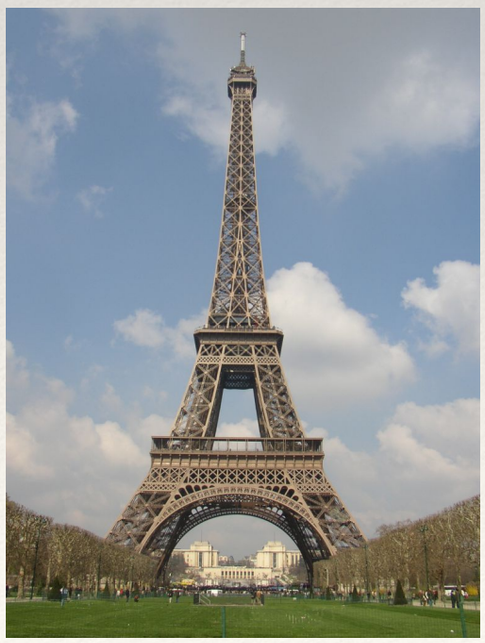
Similarity

Slide credit: Kristen Grauman

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

Symmetry





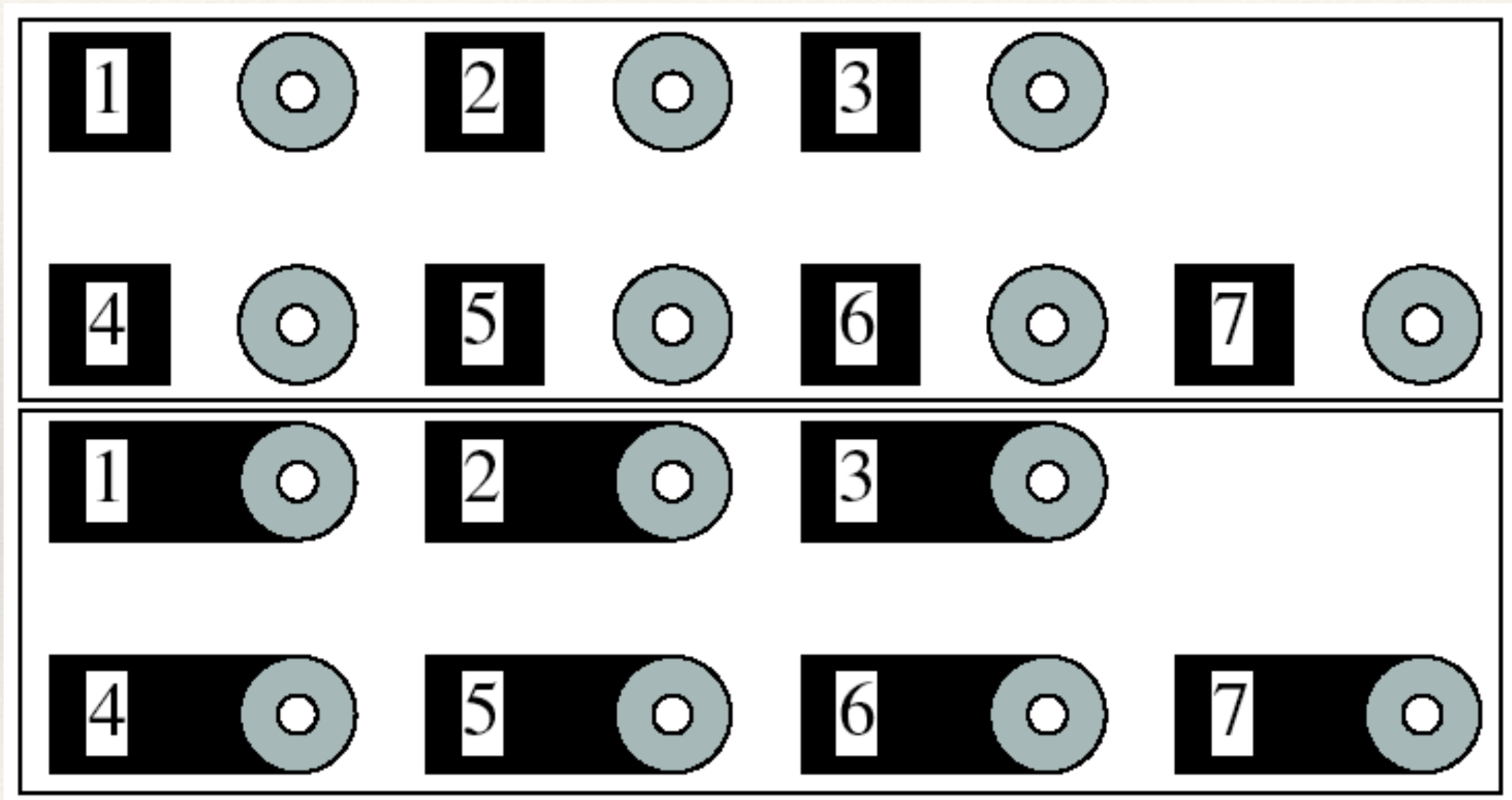
Slide credit: Kristen Grauman

Common fate

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

Slide credit: Kristen Grauman

Grouping phenomena in real life



Forsyth & Ponce, Figure 14.7

Gestalt

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/ featuresHumans interpret image information collectively
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)
  - Humans interpret image information collectively
- Inspiring observations/explanations; challenge remains how to best map to algorithms.

Bottom-up segmentation via clustering

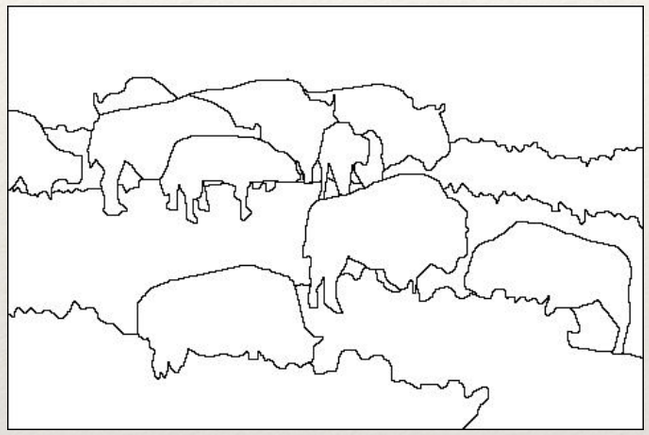
- Algorithms:
  - Mode finding and mean shift: k-means, EM, mean-shift
  - Graph-based: normalized cuts
- Features: color, texture, …
  - Quantization for texture summaries

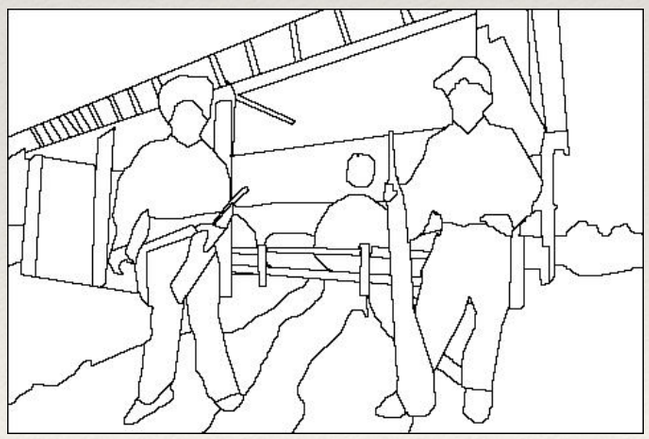
The goals of segmentation

Separate image into coherent “objects”

image

human segmentation





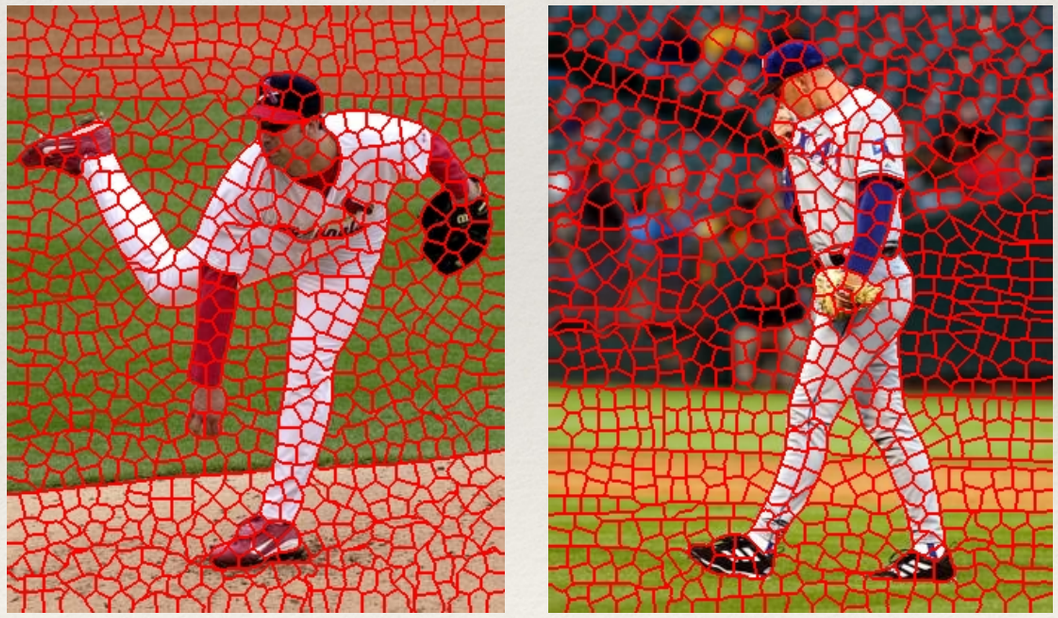
Source: Lana Lazebnik

The goals of segmentation

Separate image into coherent “objects”

Group together similar-looking pixels for efficiency of further processing

“superpixels”



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

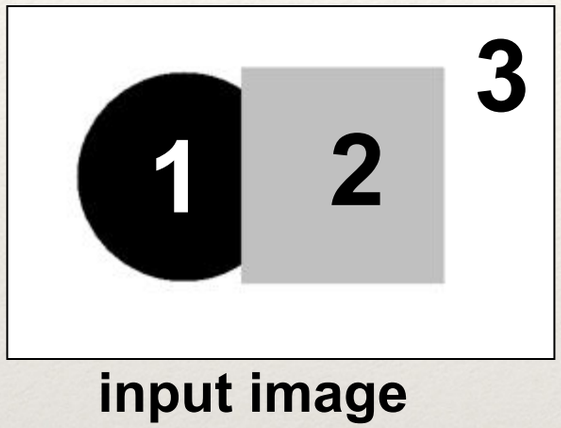
Source: Lana Lazebnik

Clustering

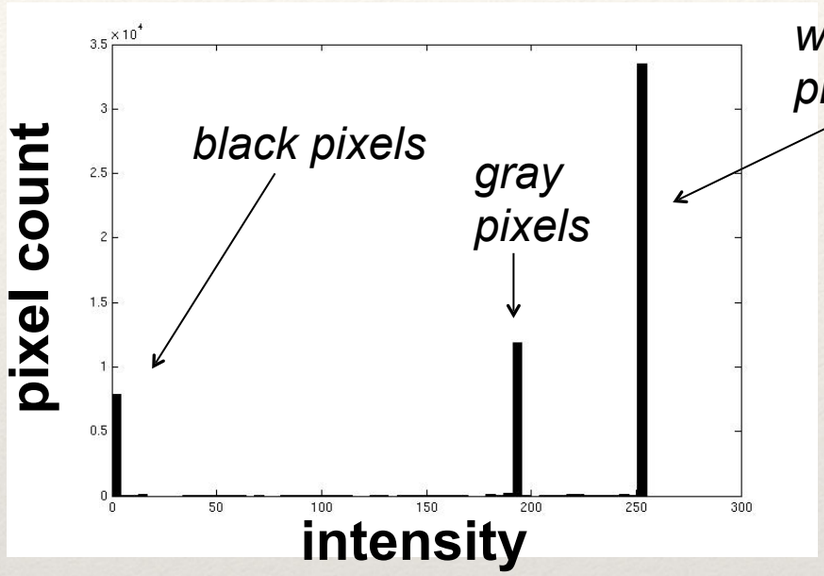
- Clustering algorithms:
  - Unsupervised learning
  - Detect patterns in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. group pixels into regions
- Useful when don’t know what you’re looking for
- Requires data, but no labels

Slide credit: Dan Klein

Image segmentation: toy example

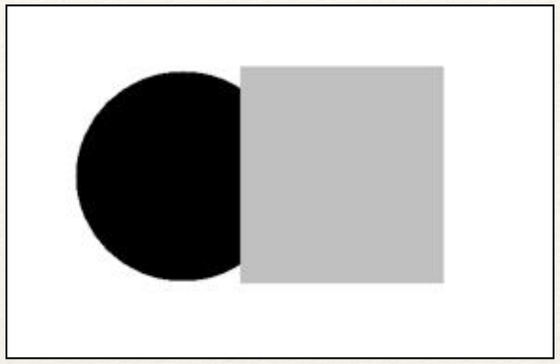


input image

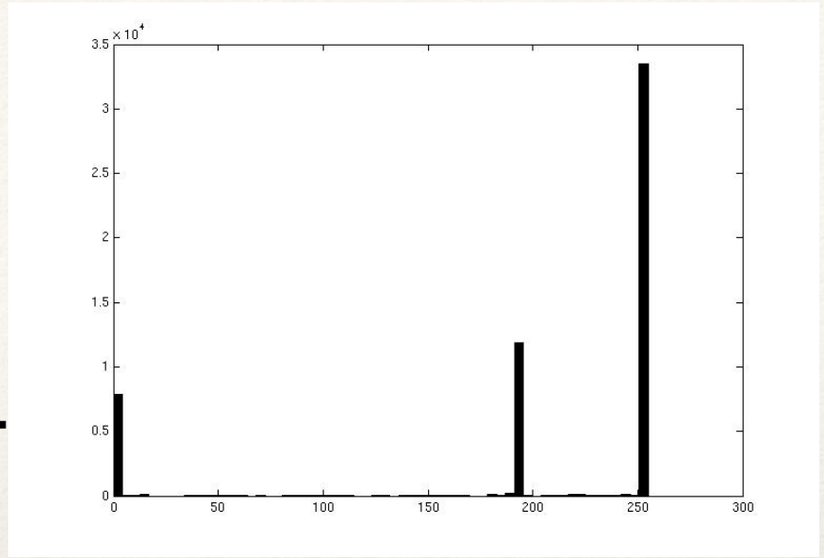


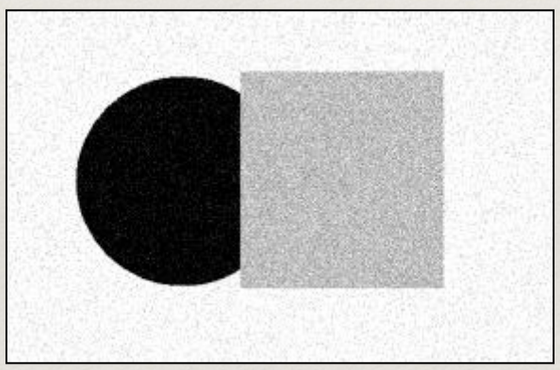
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Slide credit: Kristen Grauman

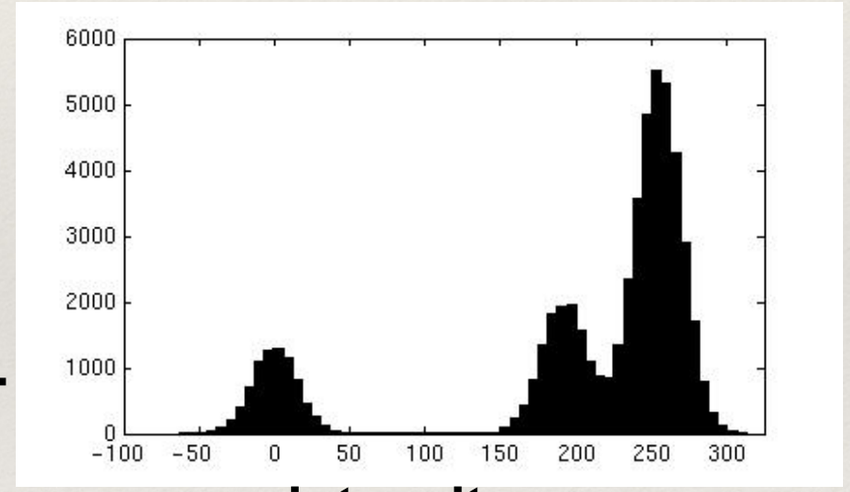


input image



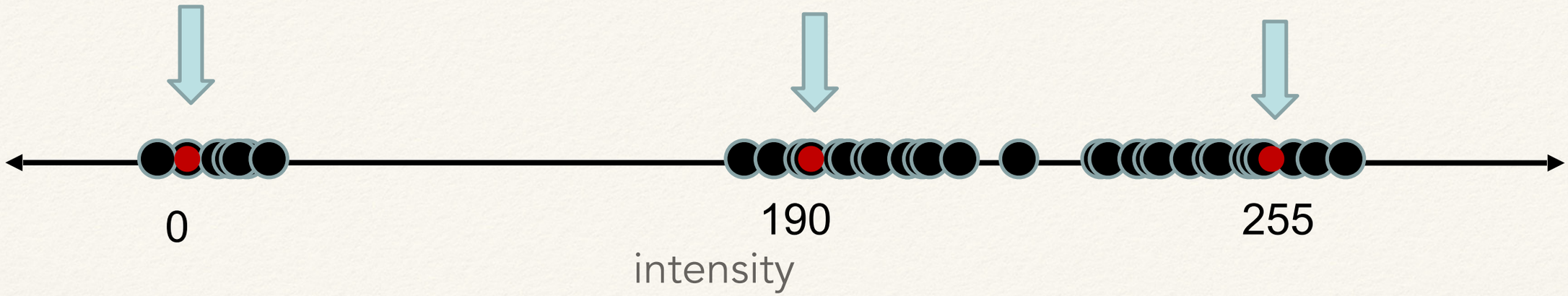


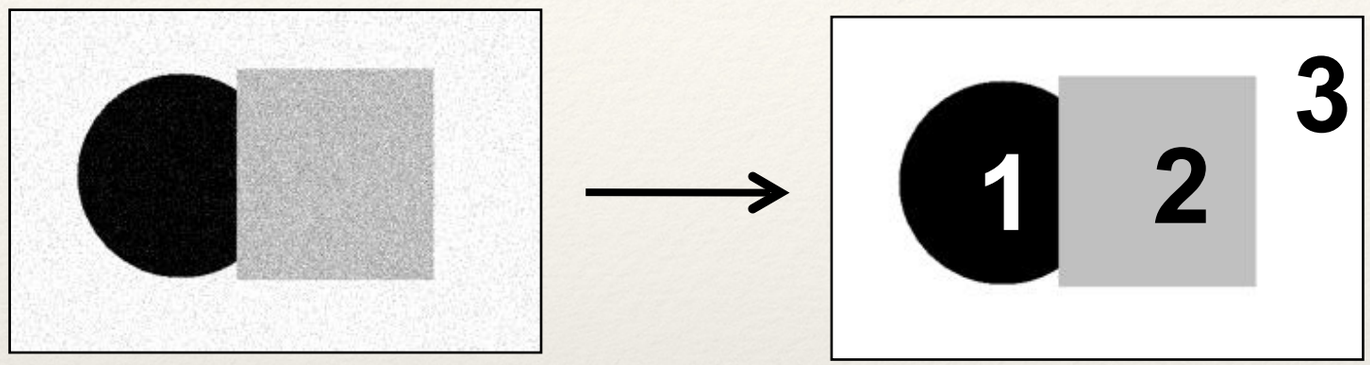
input image



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

Slide credit: Kristen Grauman





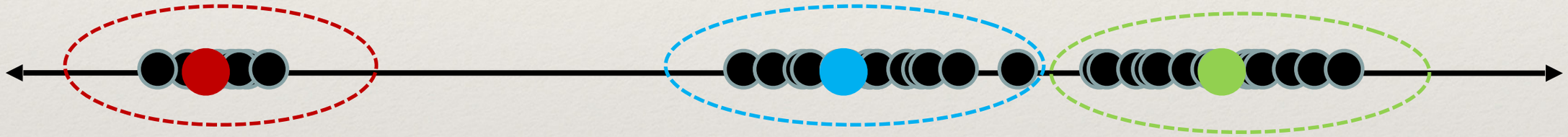
- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

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

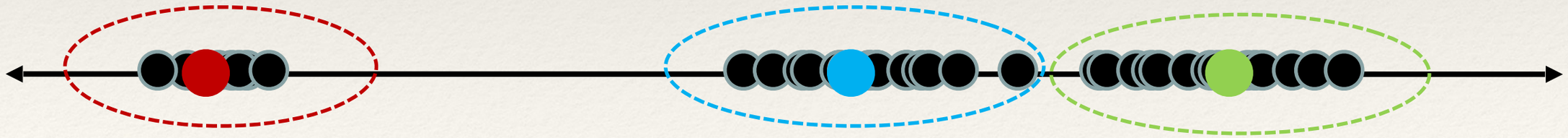
Slide credit: Kristen Grauman

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.



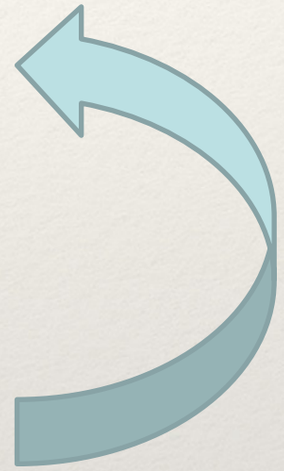
- If we knew the **group memberships**, we could get the centers by computing the mean per group.



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



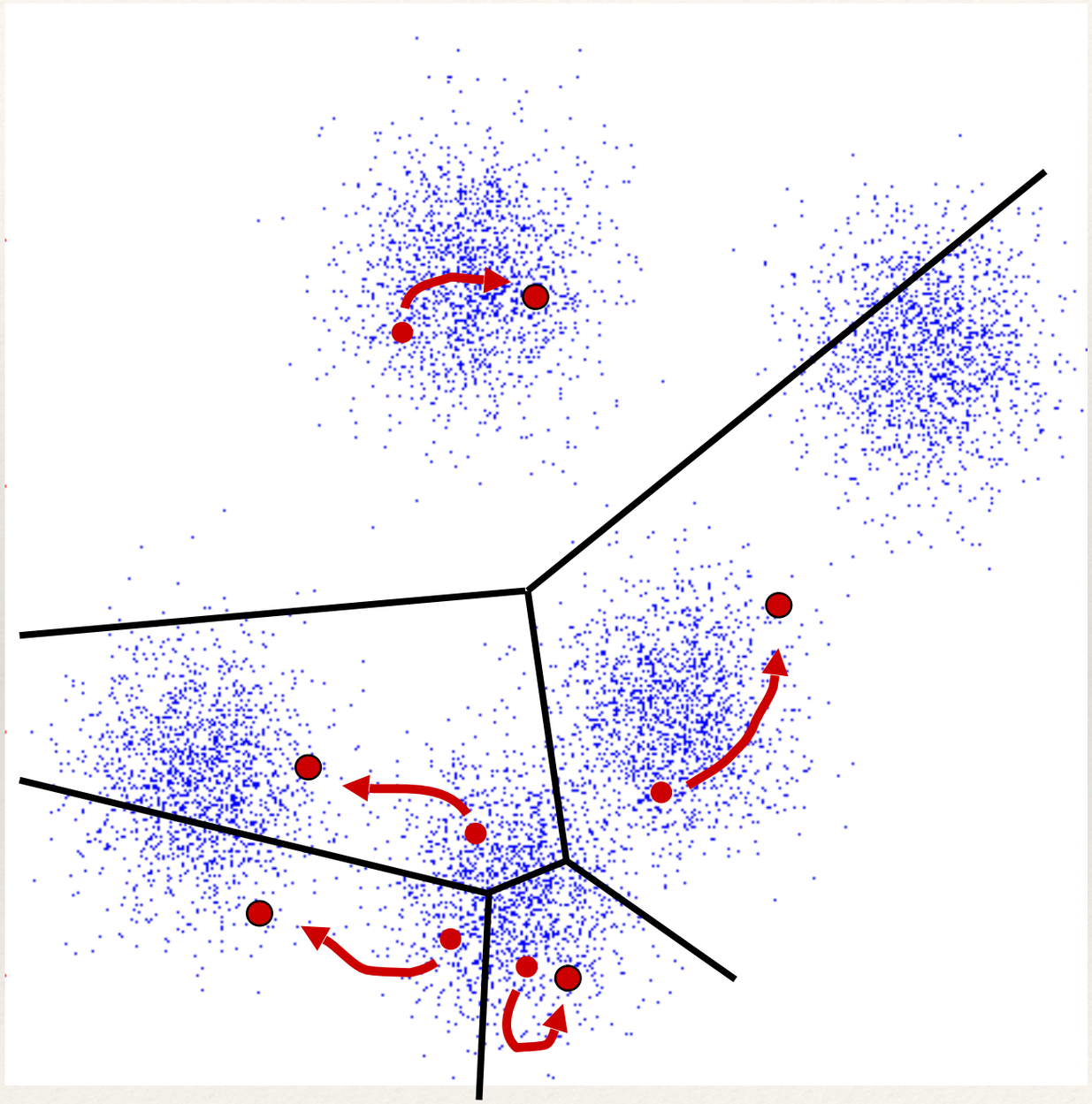
- Properties
  - Will always converge to *some* solution
  - Can be a "local minimum"
    - does not always find the global minimum of objective function:

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

Source: Steve Seitz

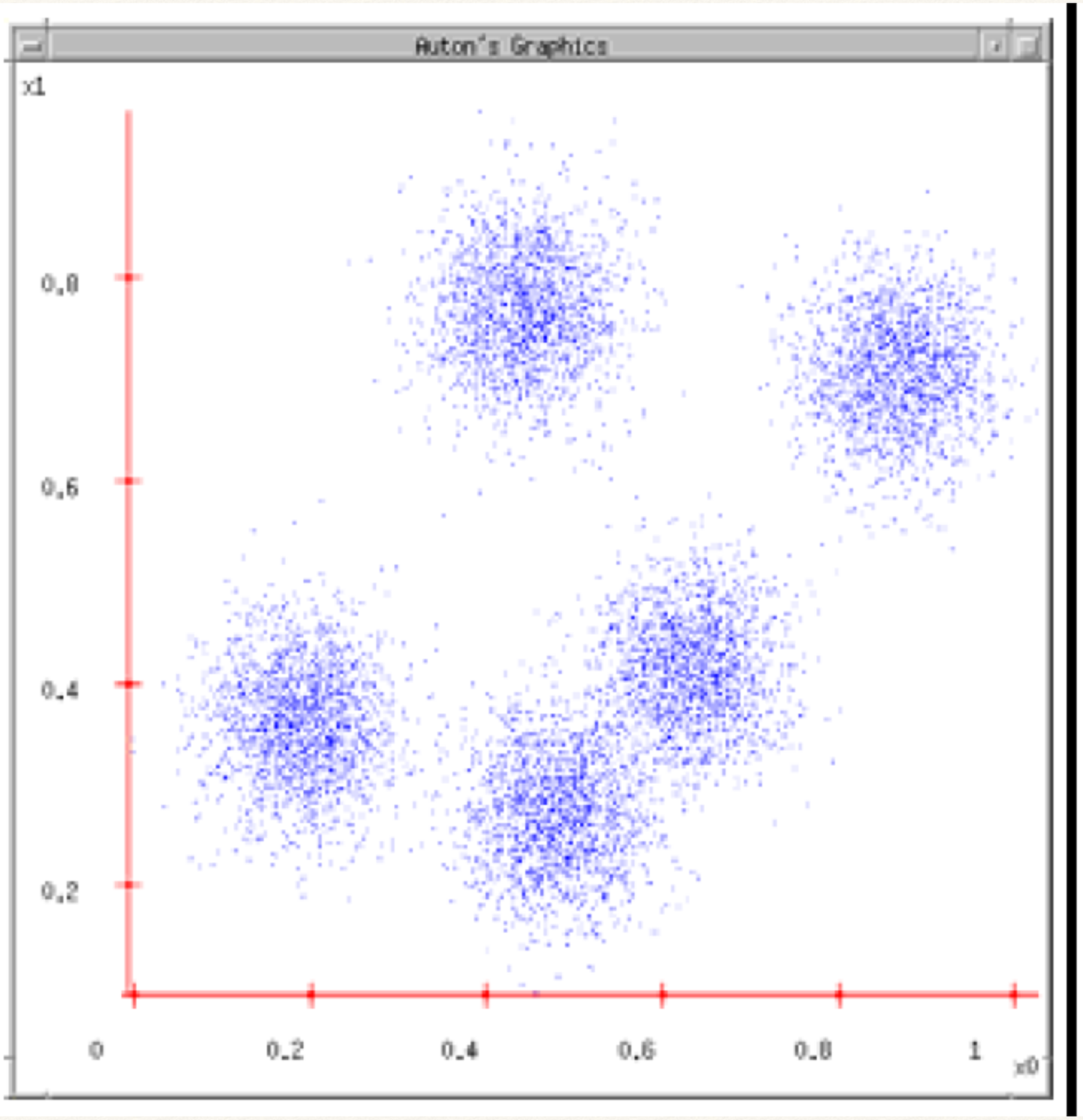
K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
- Stop when no points' assignments change



K-means

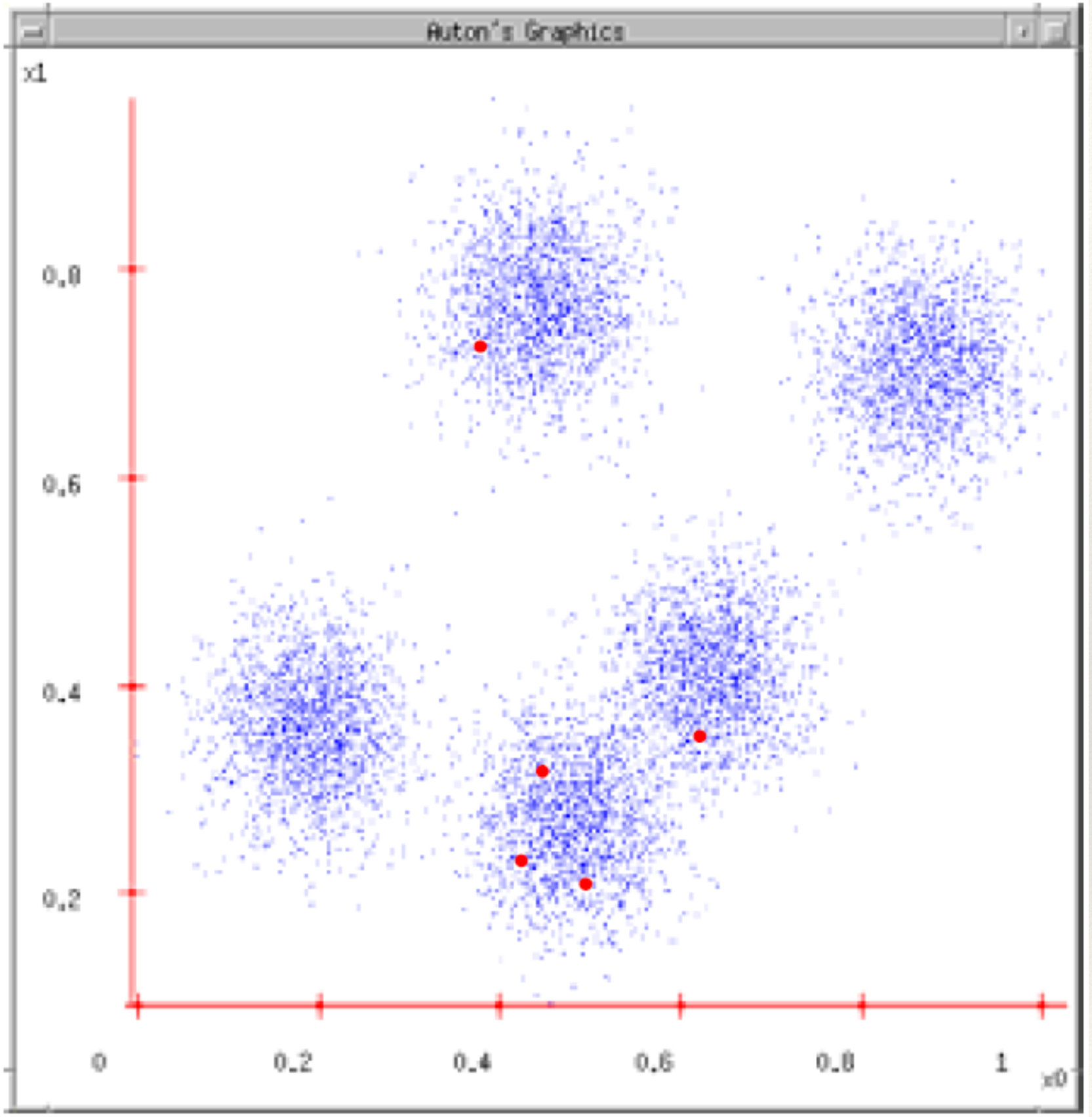
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Slide credit Andrew Moore

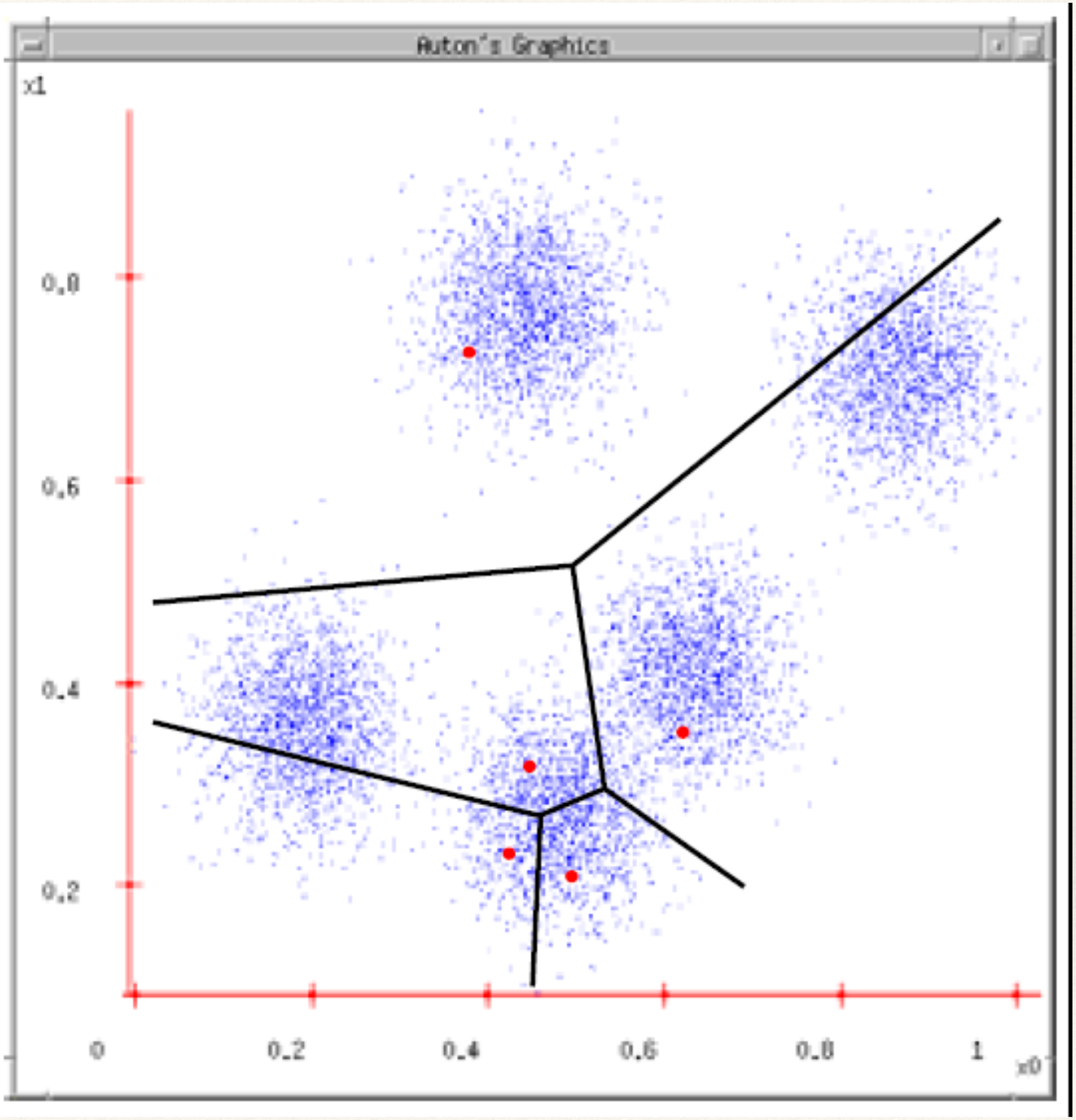
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



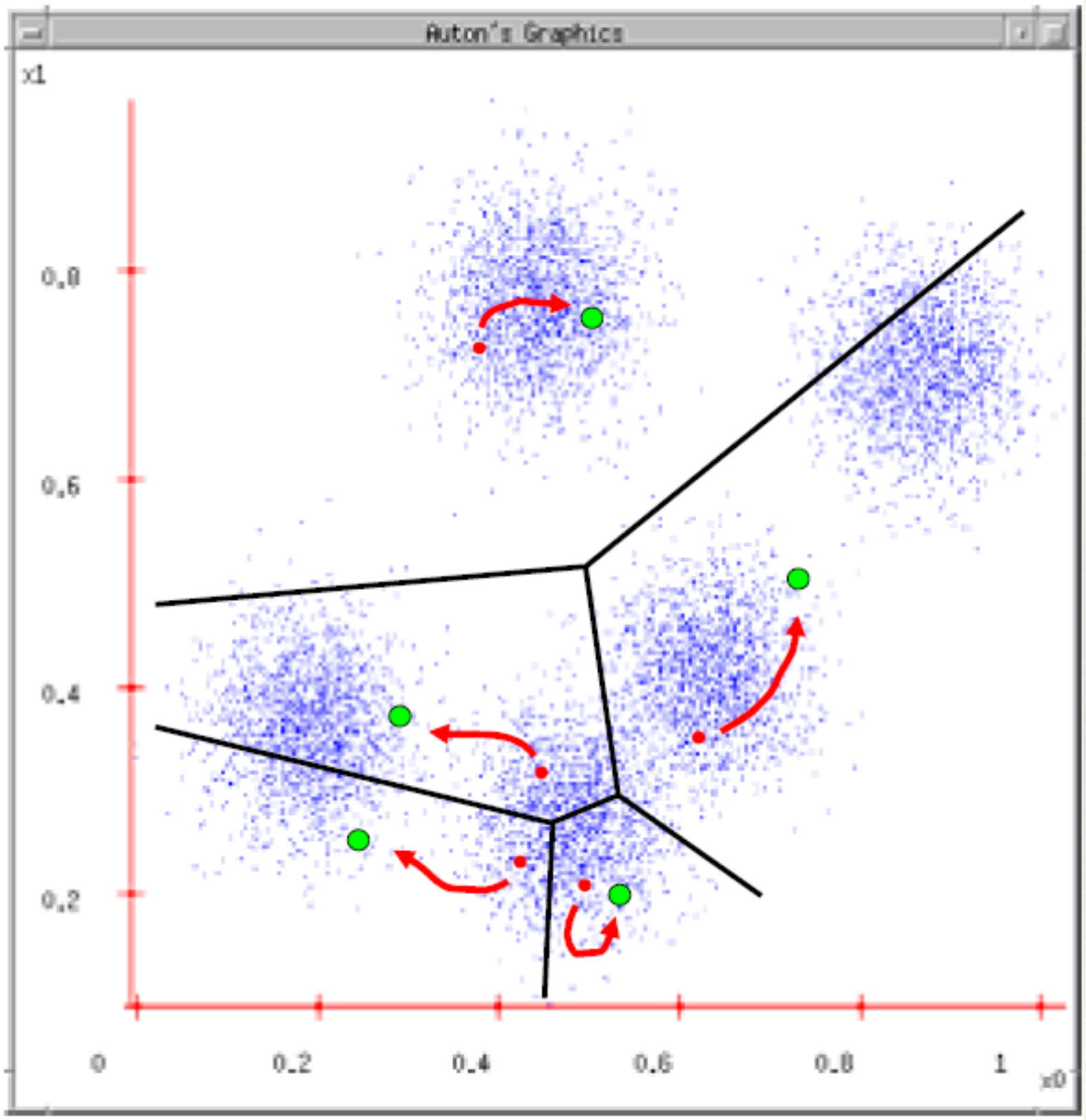
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



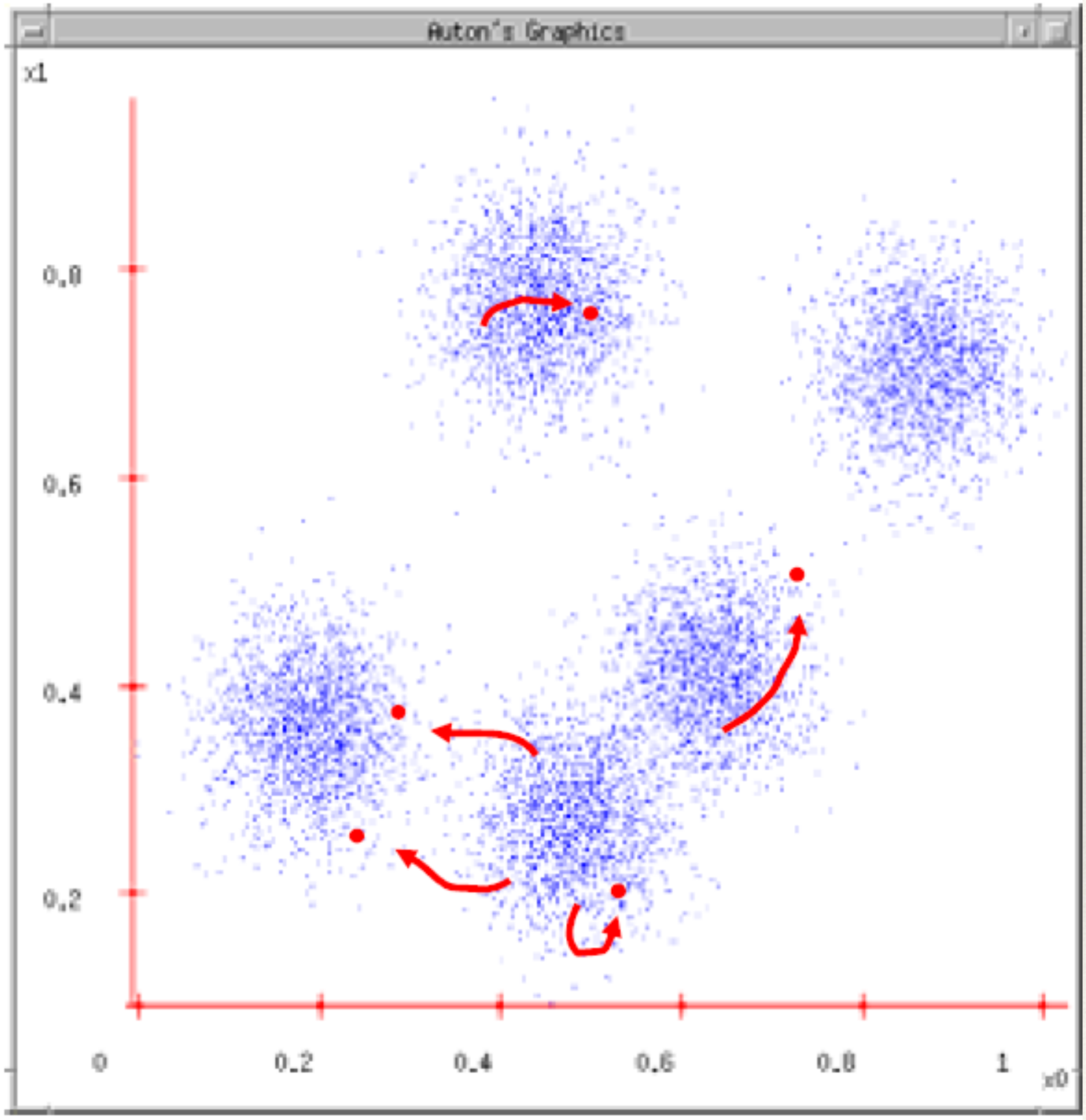
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



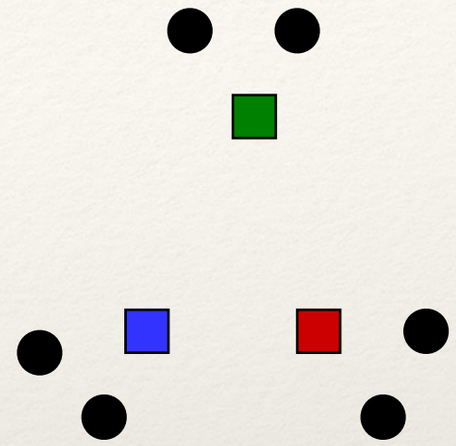
K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!

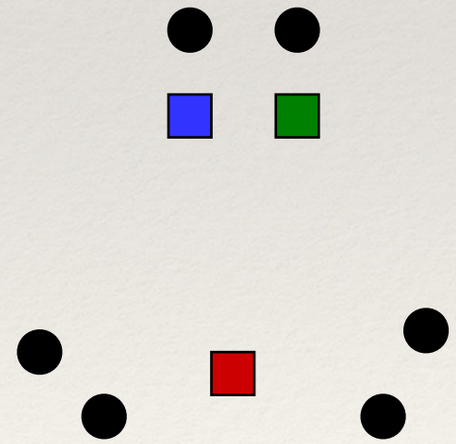


Initialization

- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!



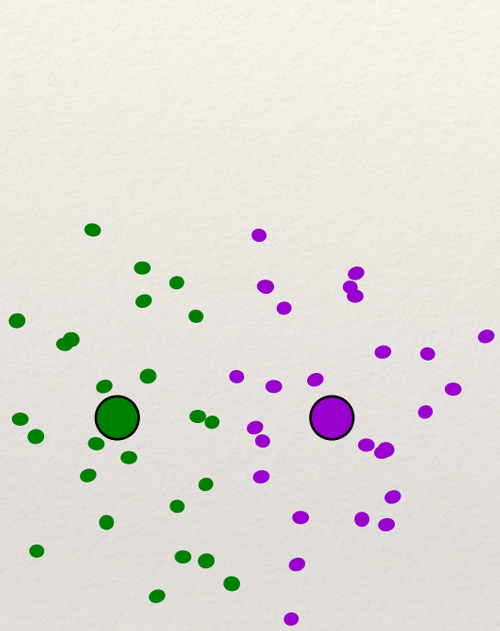
- What can go wrong?
- Various schemes for preventing this kind of thing

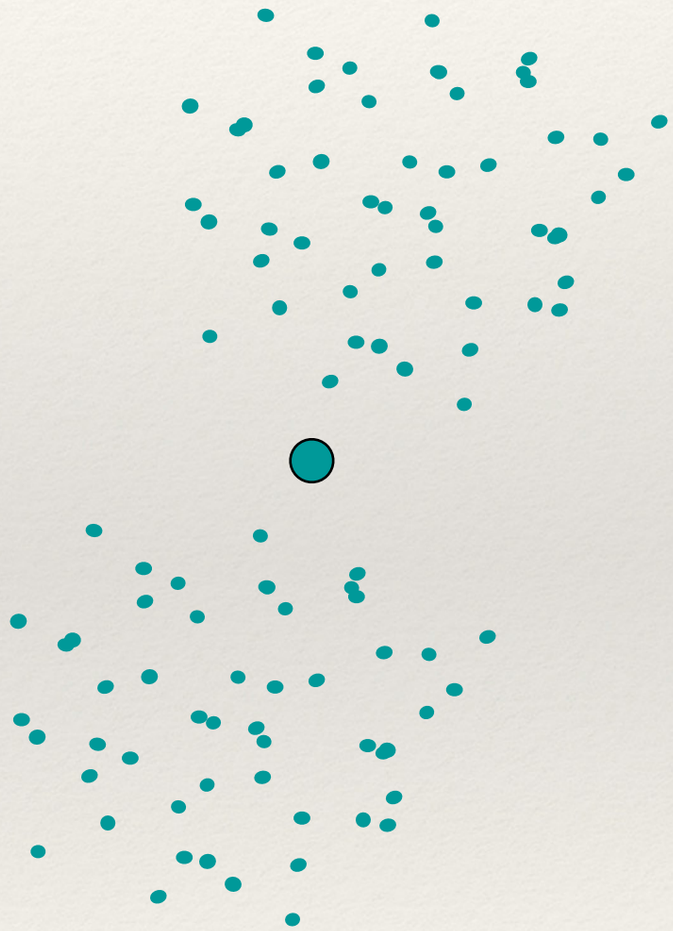


Slide credit: Dan Klein

K-Means getting stuck

- A local optimum:





Slide credit: Dan Klein

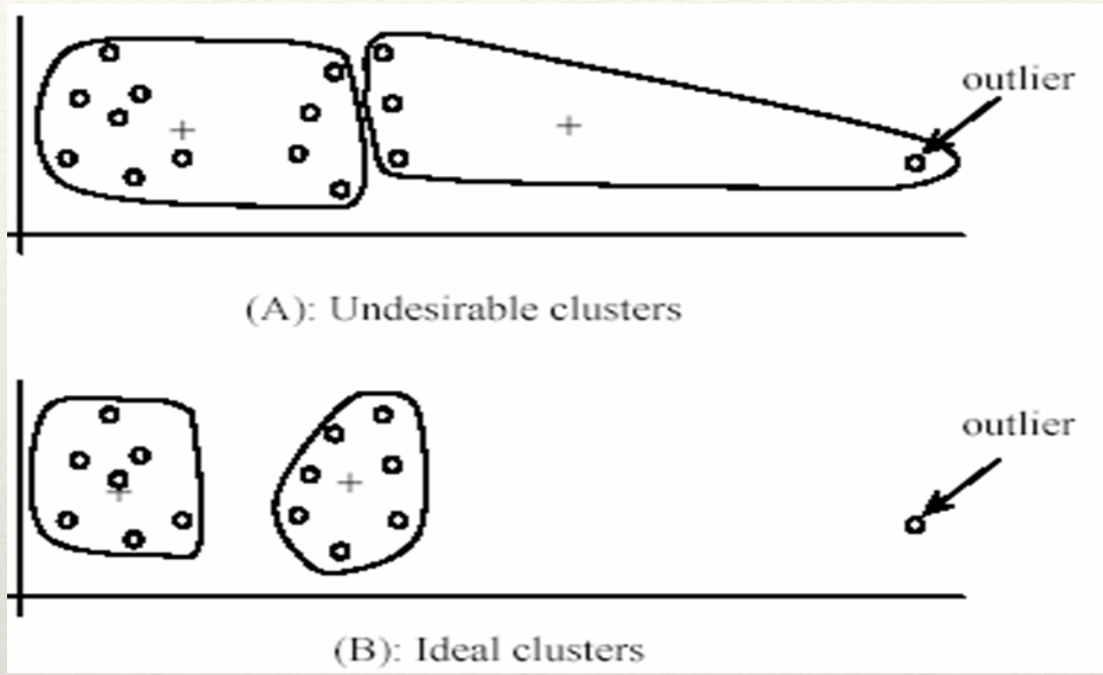
K-means: pros and cons

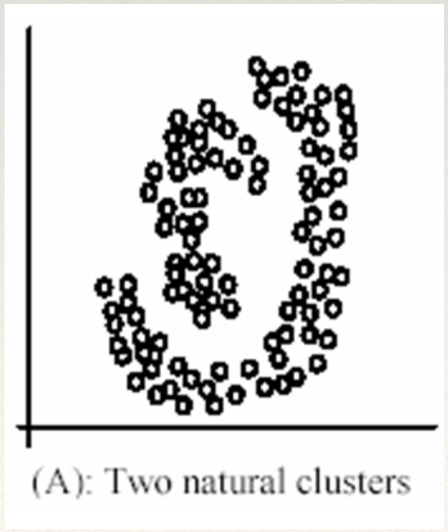
<u>Pros</u>

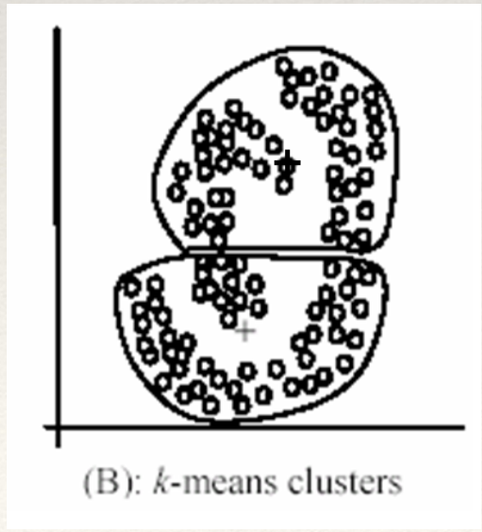
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

<u>Cons/issues</u>

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assuming means can be computed




(A): Two natural clusters


(B): *k*-means clusters

K-Means questions

- Will K-means converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
  - If the patterns are very very clear?
- Will it find something interesting?
- How many clusters to pick?
- Do people ever use it?

Slide credit: Dan Klein

Probabilistic clustering

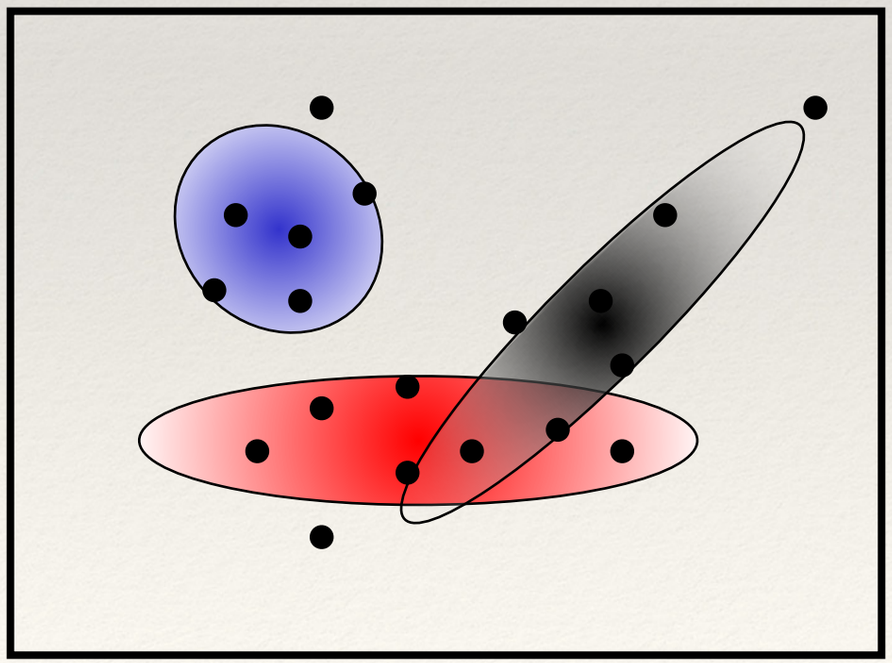
Basic questions

- 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

Probabilistic clustering (basic idea)

- Treat each cluster as a Gaussian density function

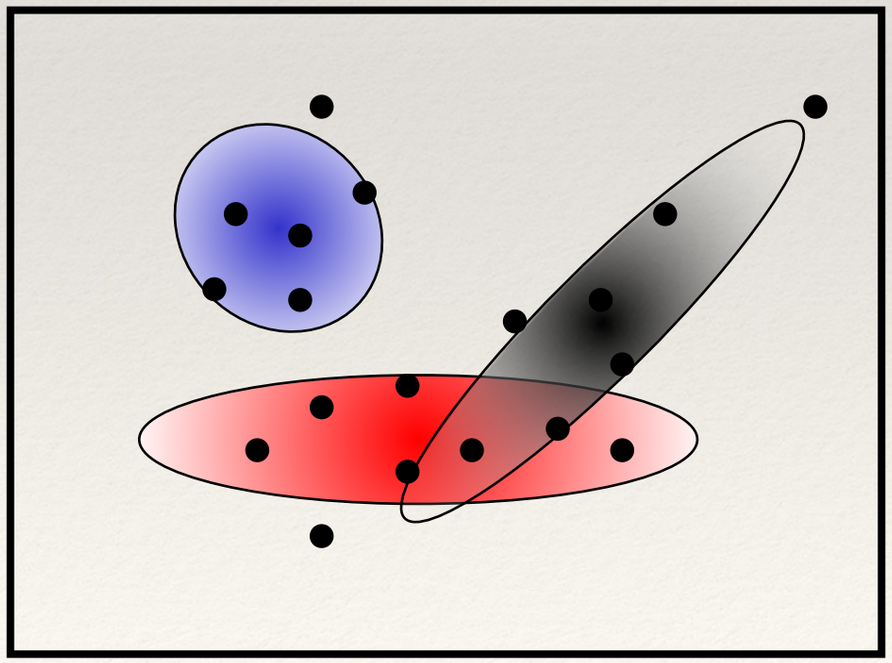


Slide credit: Steve Seitz

Expectation Maximization (EM)

A probabilistic variant of K-means:

- E step: "soft assignment" of points to clusters
  - estimate probability that a point is in a cluster
- M step: update cluster parameters
  - mean and variance info (covariance matrix)
- maximizes the likelihood of the points given the clusters



Slide credit: Steve Seitz

Segmentation as clustering

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

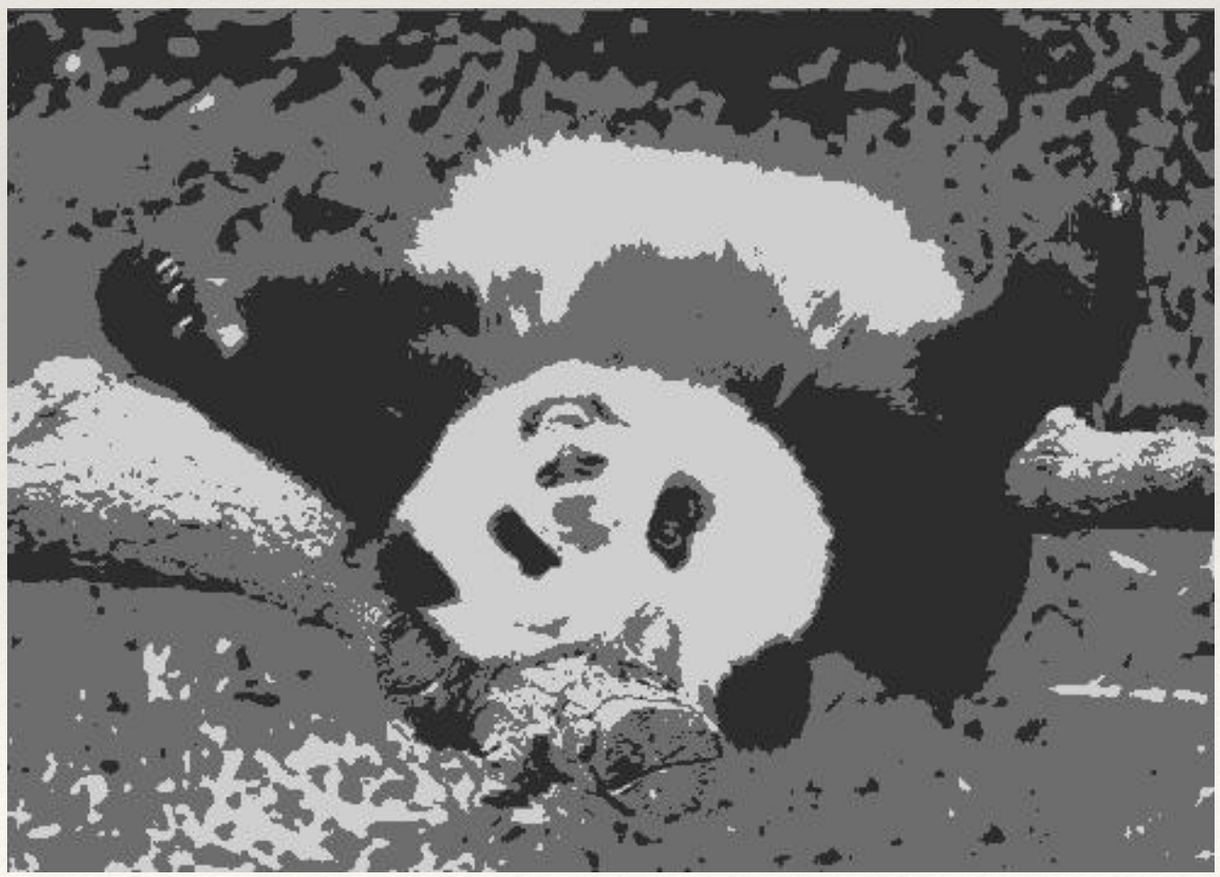
Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

Slide credit: Kristen Grauman

K=2

K=3



quantization of the feature space; segmentation label map

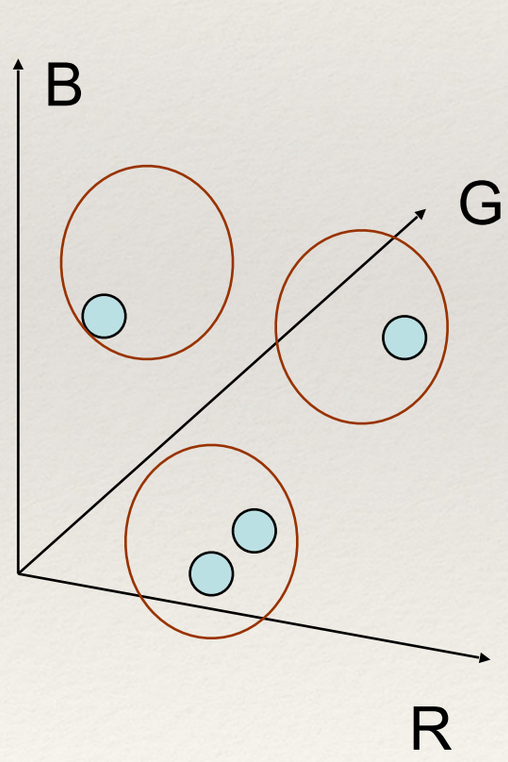
```
img_as_col = double(im(:));
cluster_membs = kmeans(img_as_col, K);

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

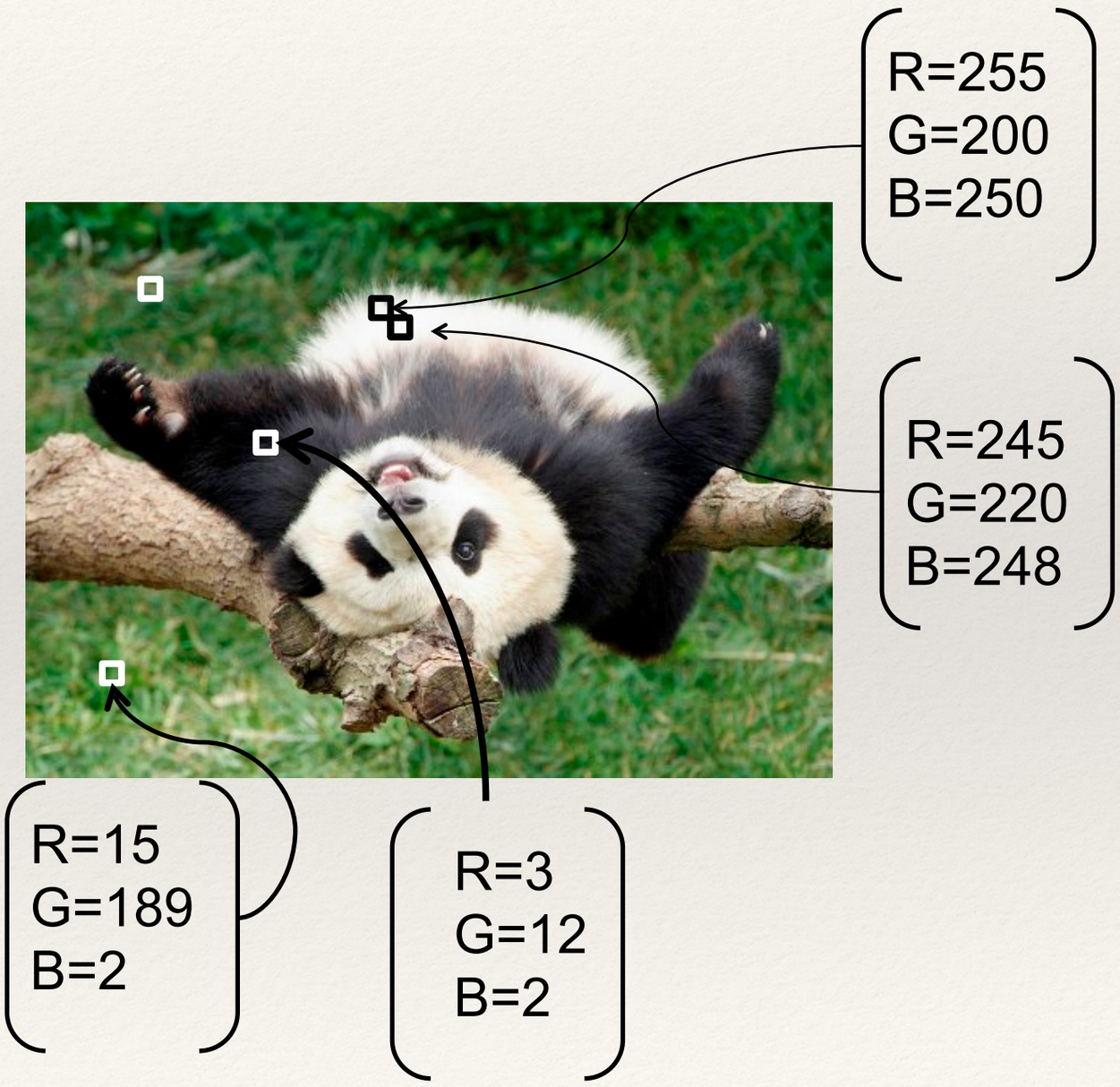
Segmentation as clustering

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 (3-d)



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** similarity

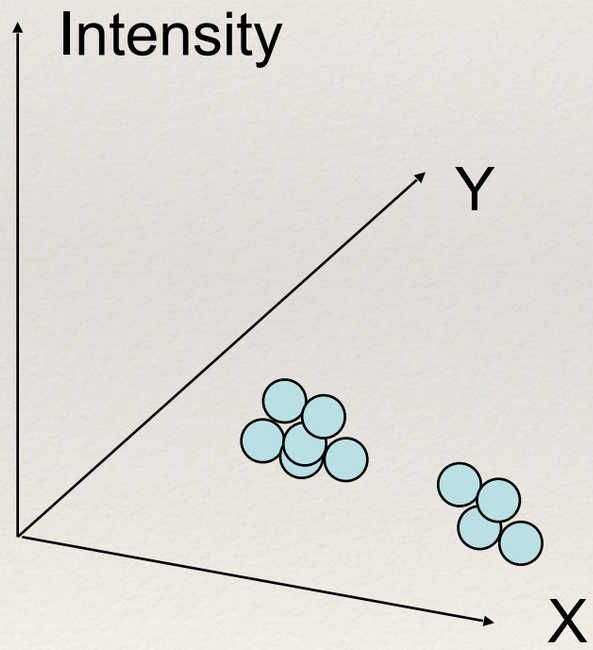
Clusters based on intensity similarity don't have to be spatially coherent.

Slide credit: Kristen Grauman

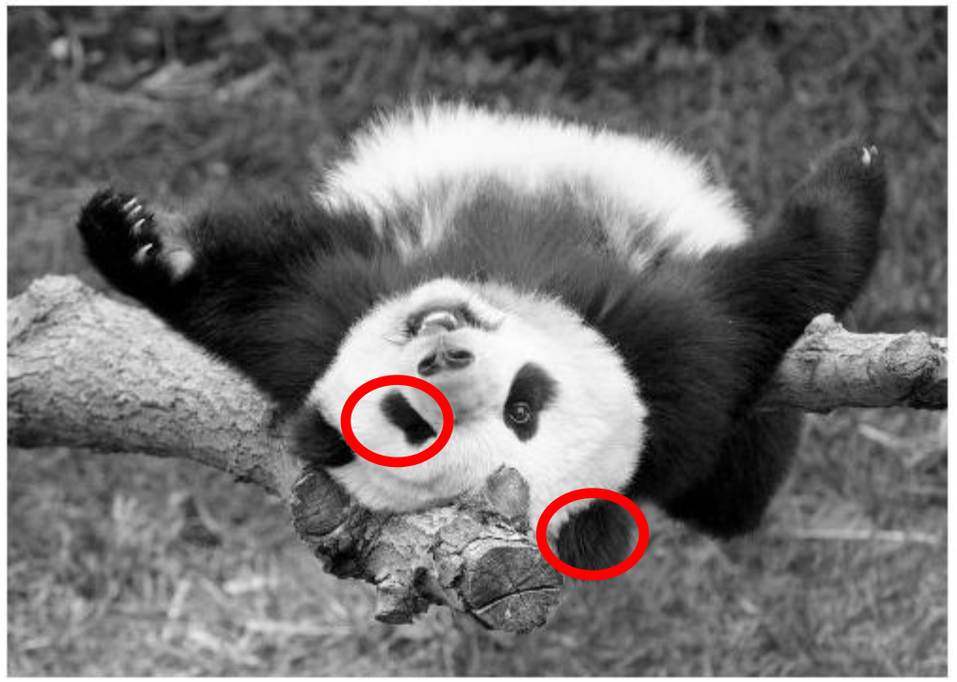
Segmentation as clustering

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

Grouping pixels based on **intensity+position** similarity



Slide credit: Kristen Grauman



Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

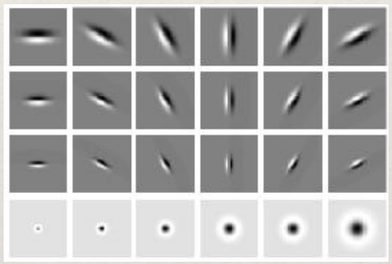
Segmentation as clustering

- Color, brightness, position alone are not enough to distinguish all regions…

Segmentation as clustering

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

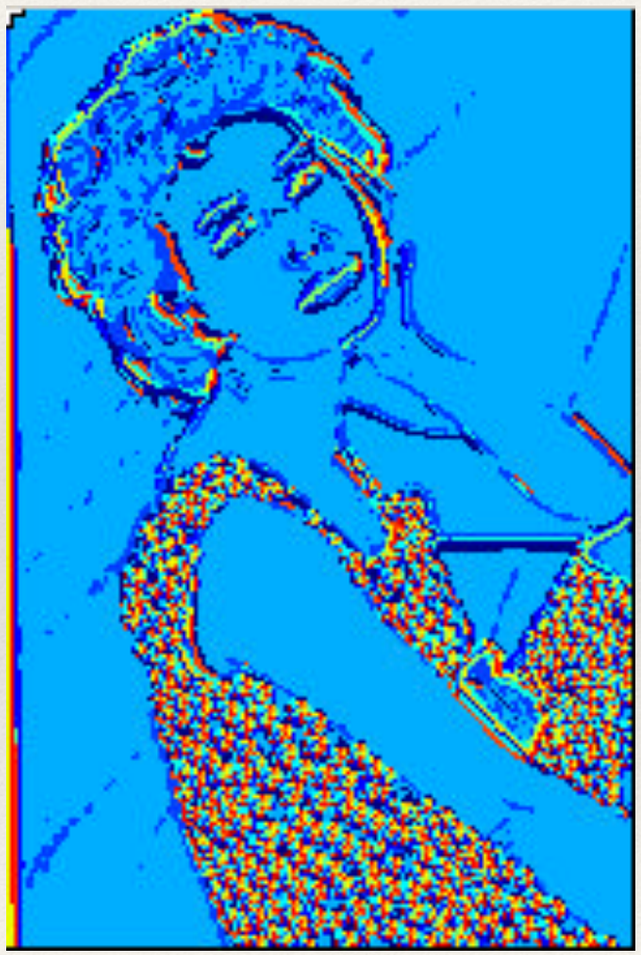
Feature space: filter bank responses (e.g., 24-d)

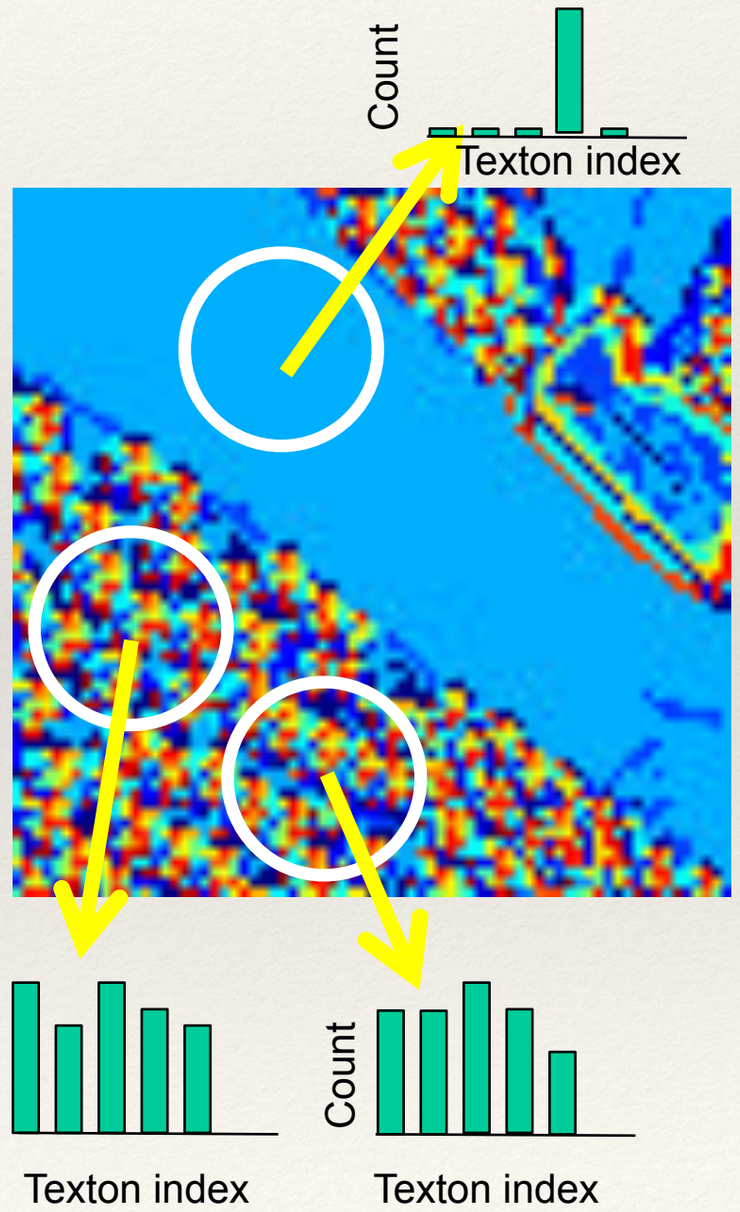
Segmentation with texture features

- Find "textons" by **clustering** vectors of filter bank outputs
- Describe texture in a window based on *texton histogram*

Image

Texton map

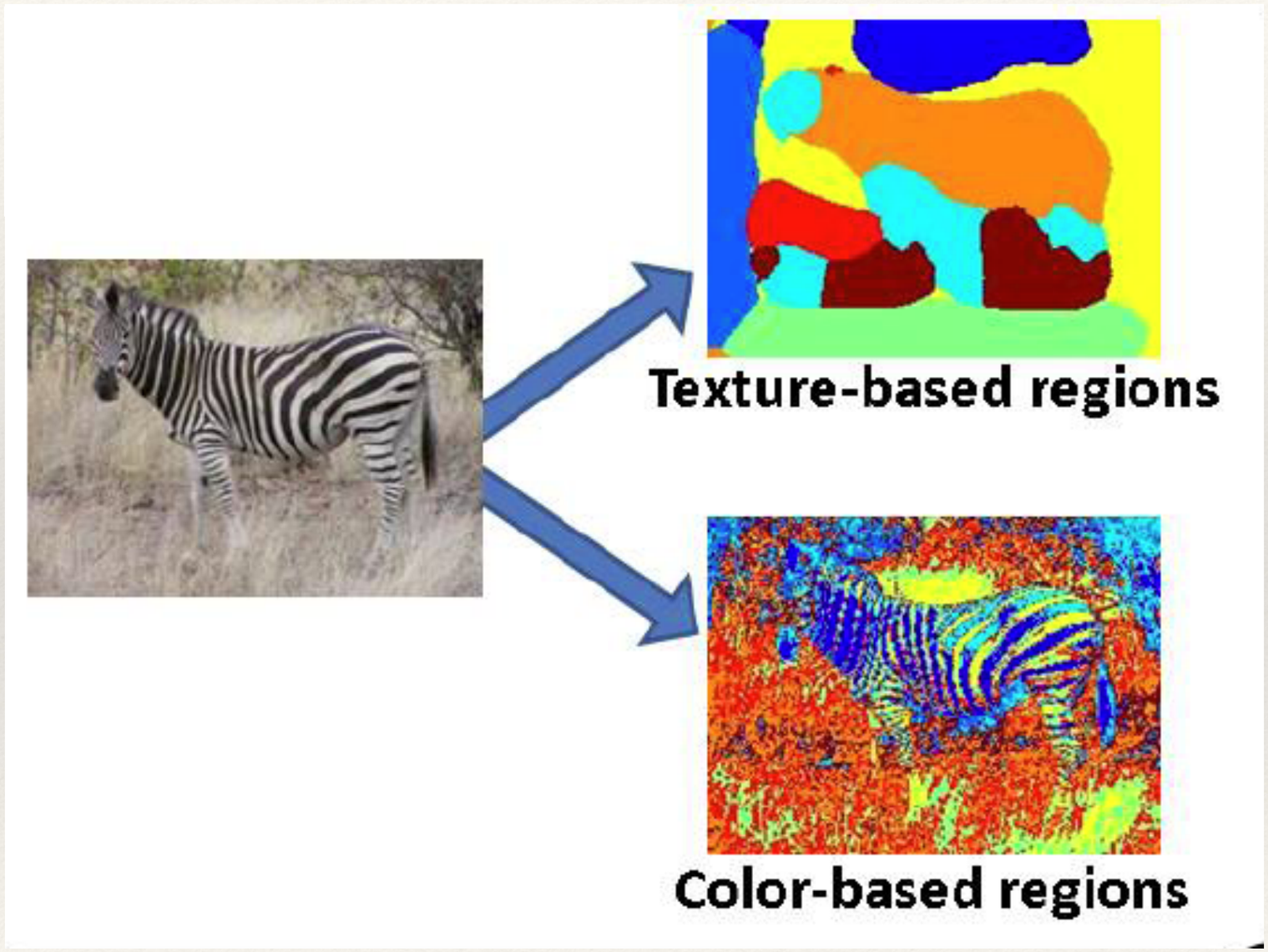




Malik, Belongie, Leung and Shi. IJCV 2001.

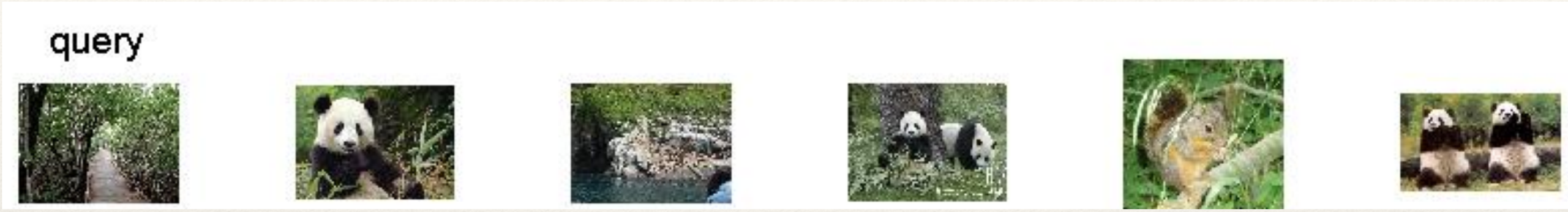
Adapted from Lana Lazebnik

Image segmentation example



Slide credit: Kristen Grauman

Pixel properties vs. neighborhood properties



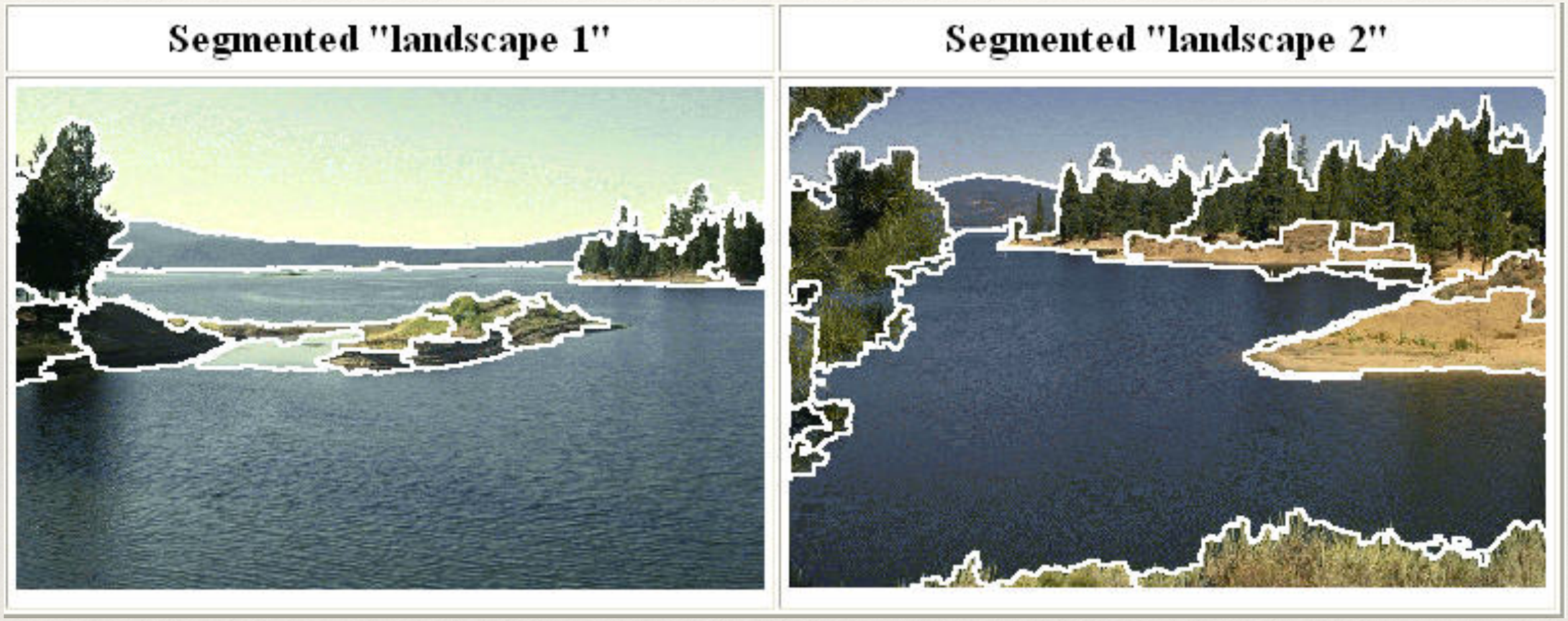
These look very similar in terms of their color distributions (histograms).

How would their *texture* distributions compare?

Slide credit: Kristen Grauman

Mean shift clustering and segmentation

- An advanced and versatile technique for clustering-based segmentation



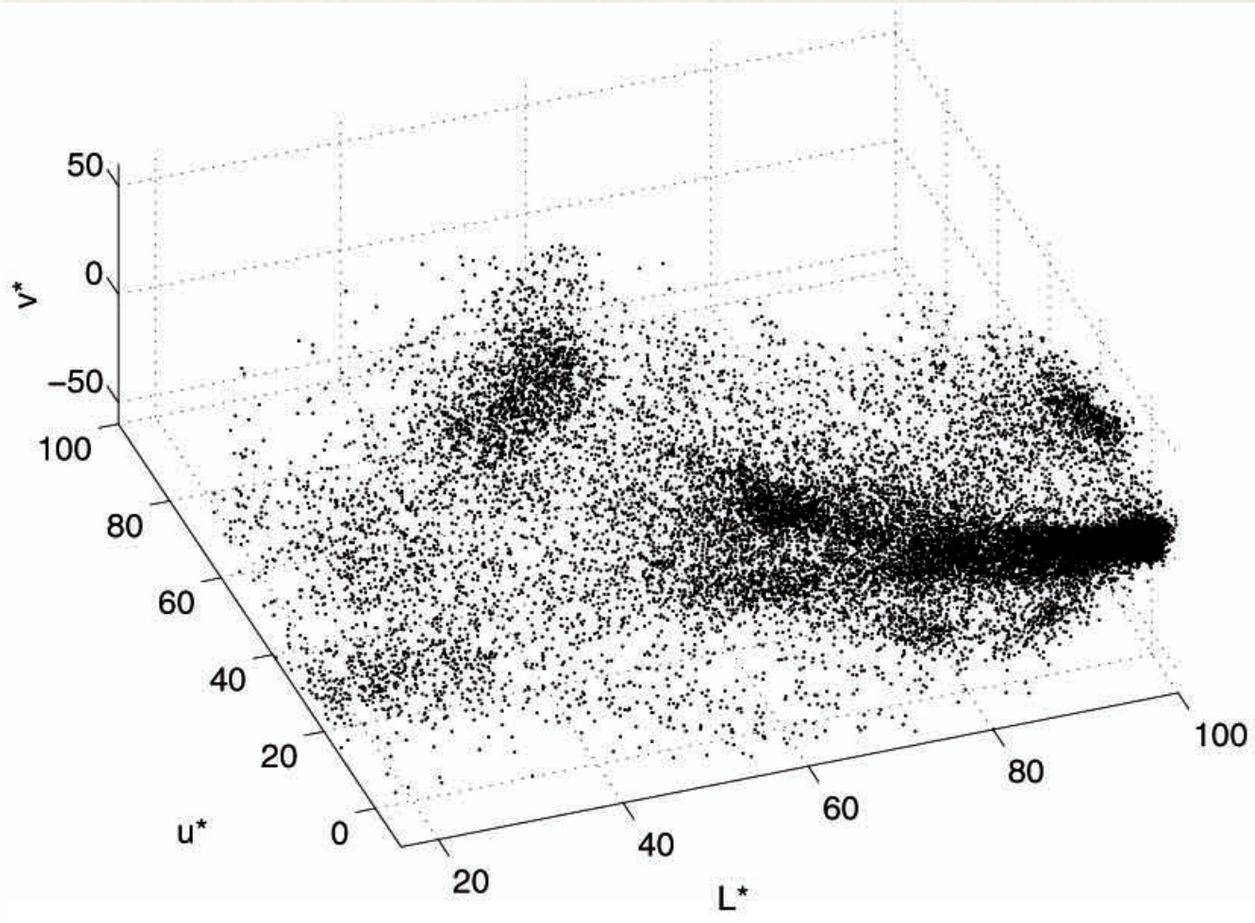
http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

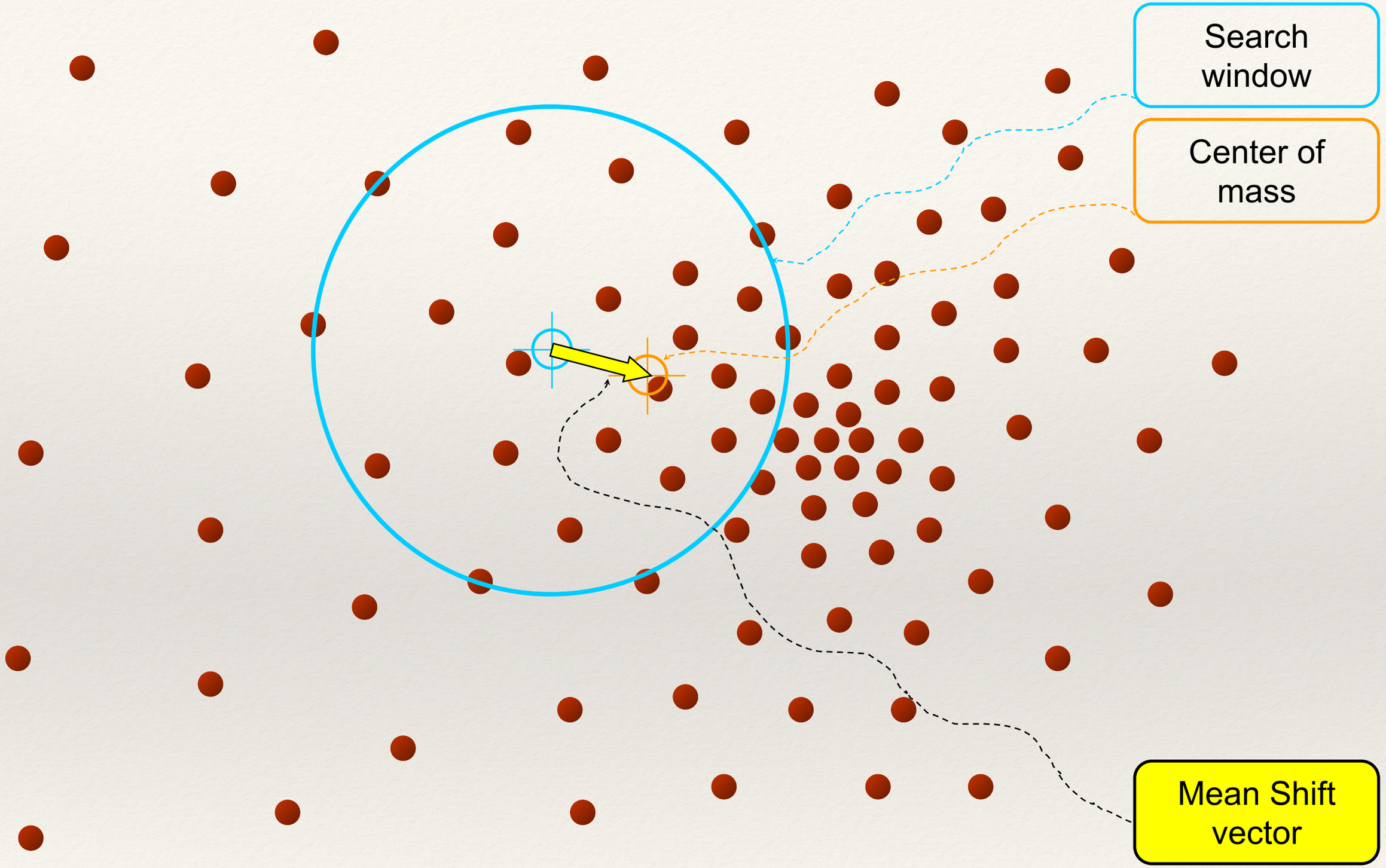
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, PAMI 2002.

Mean shift algorithm

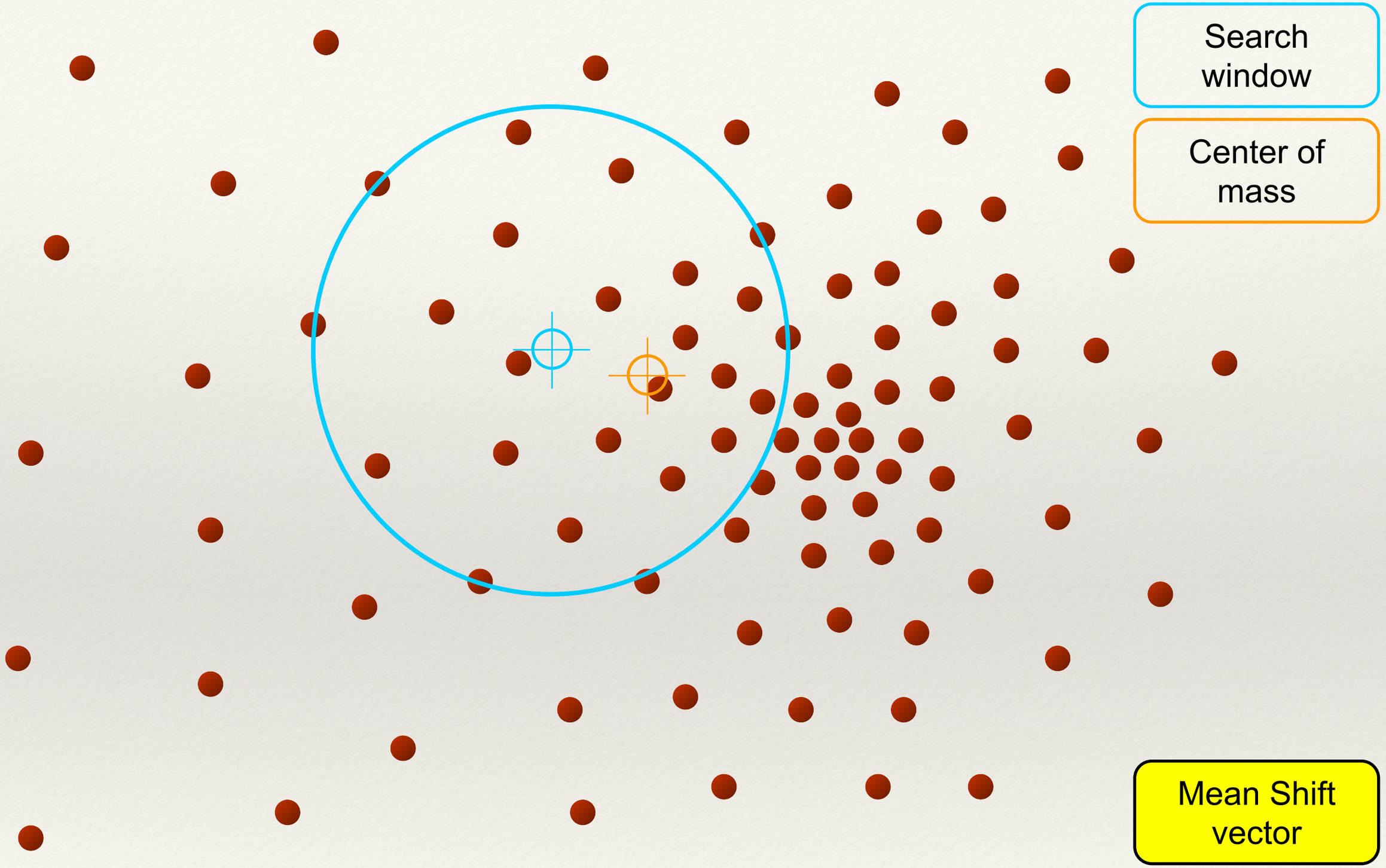
image

Feature space
(L*u*v* color values)

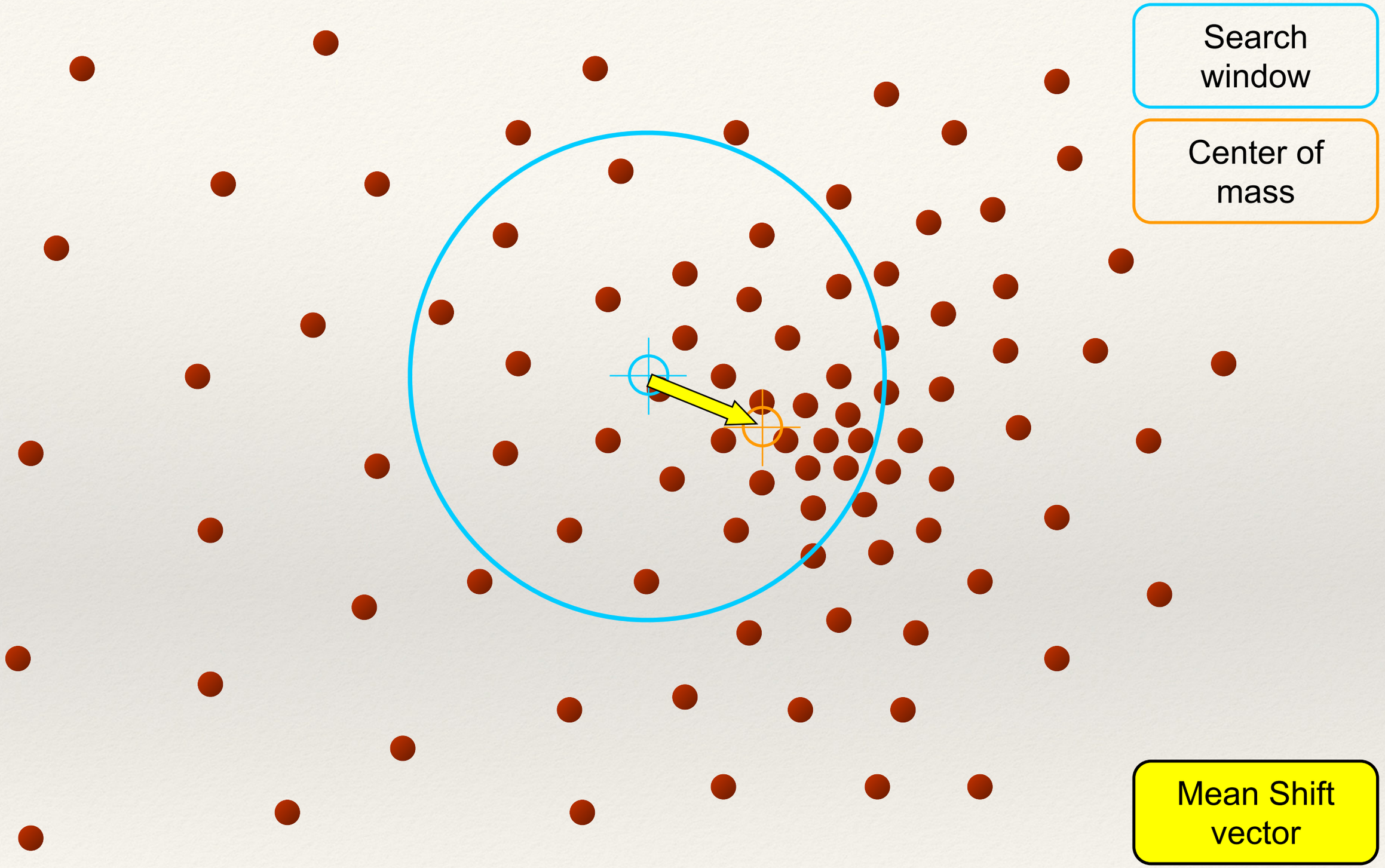




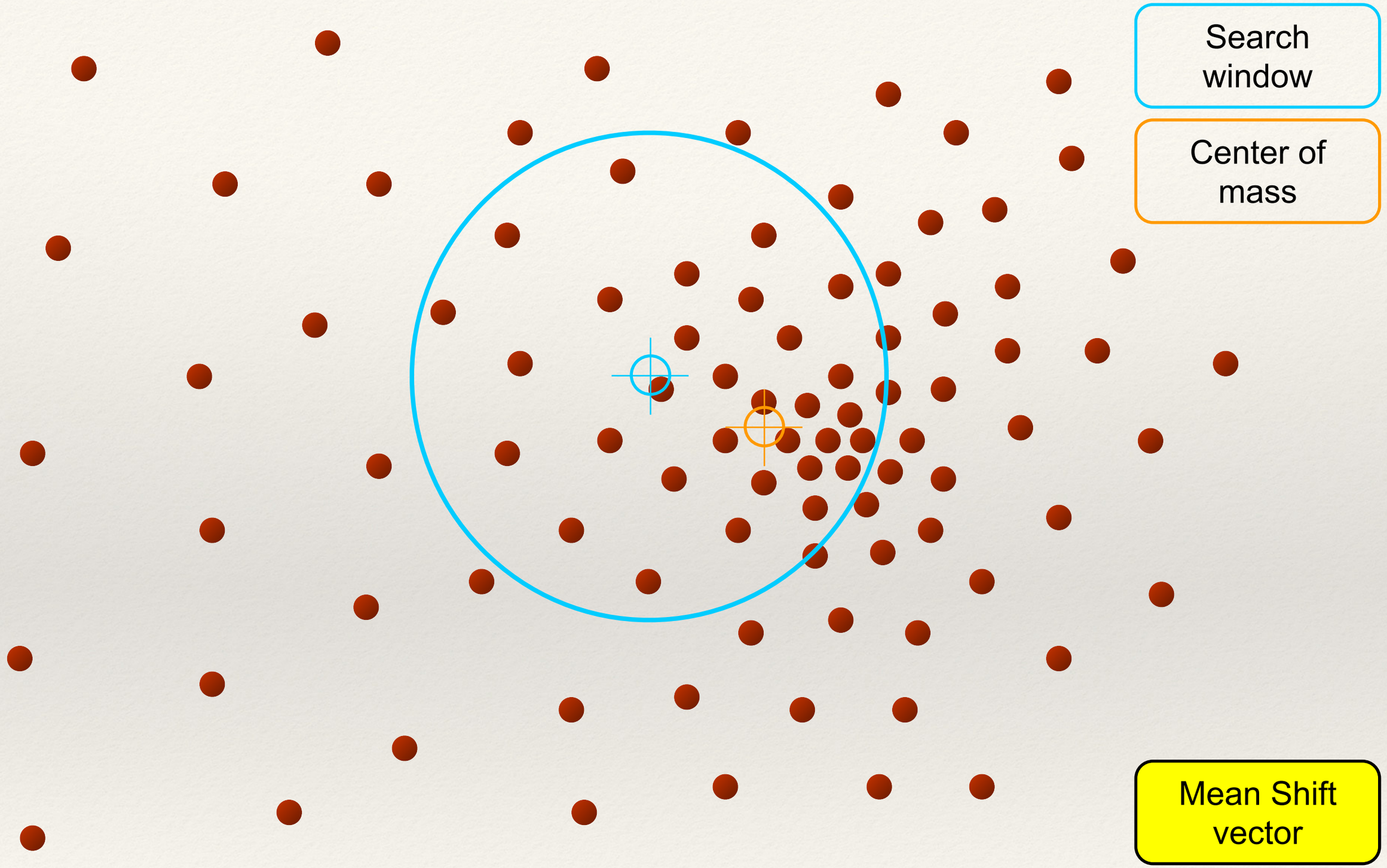
Slide by Y. Ukrainitz & B. Sarel



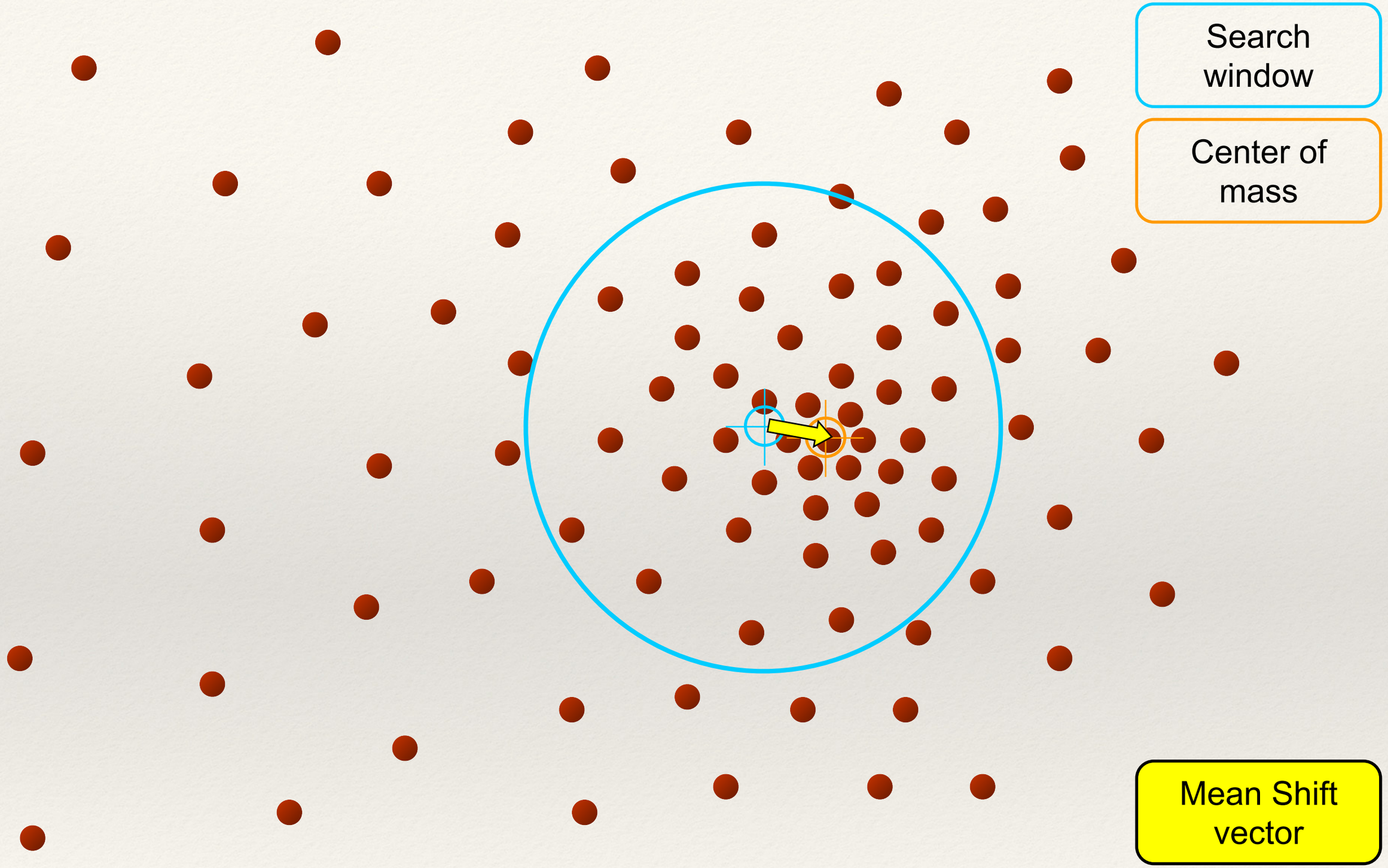
Slide by Y. Ukrainitz & B. Sarel



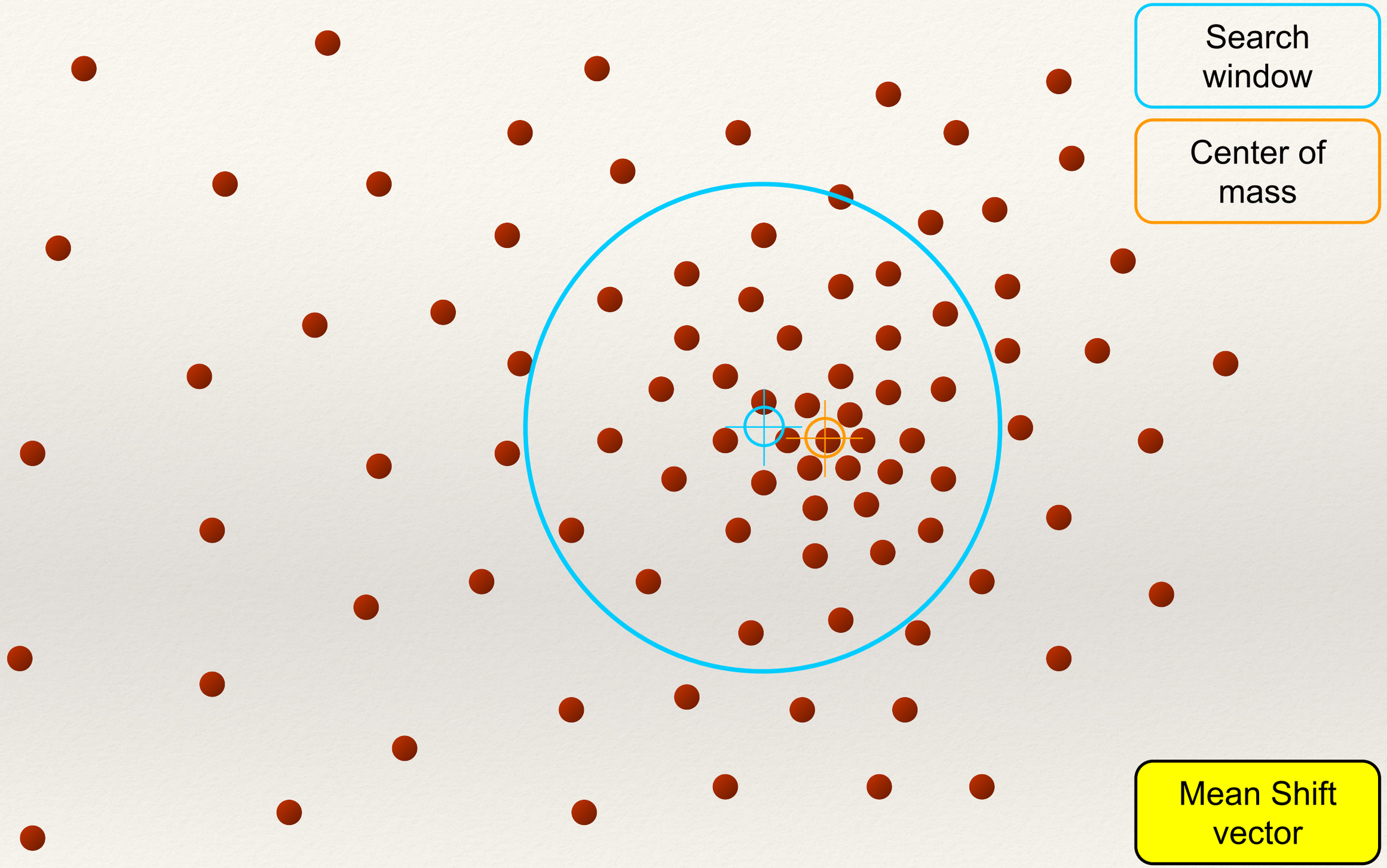
Slide by Y. Ukrainitz & B. Sarel



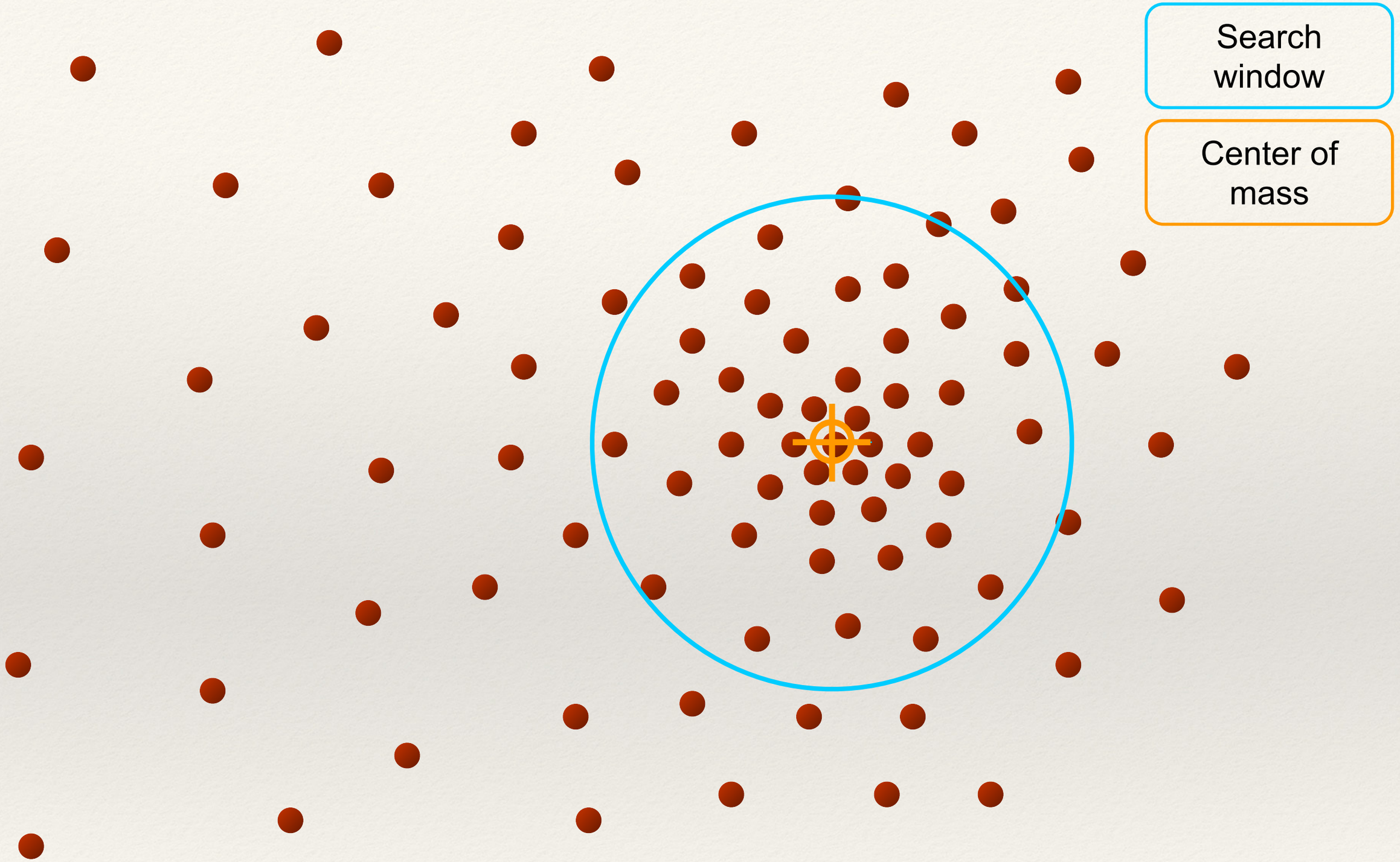
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel



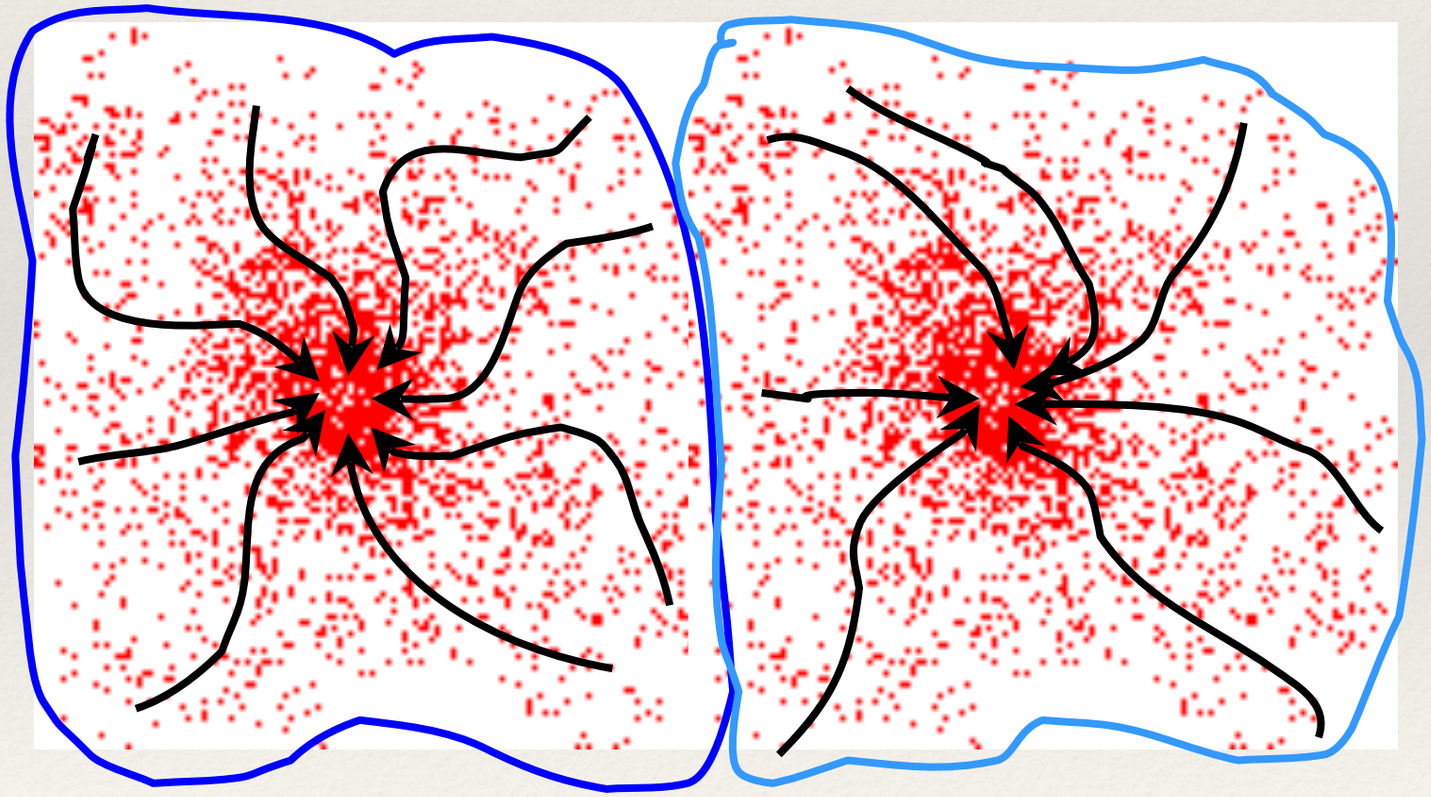
Slide by Y. Ukrainitz & B. Sarel



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

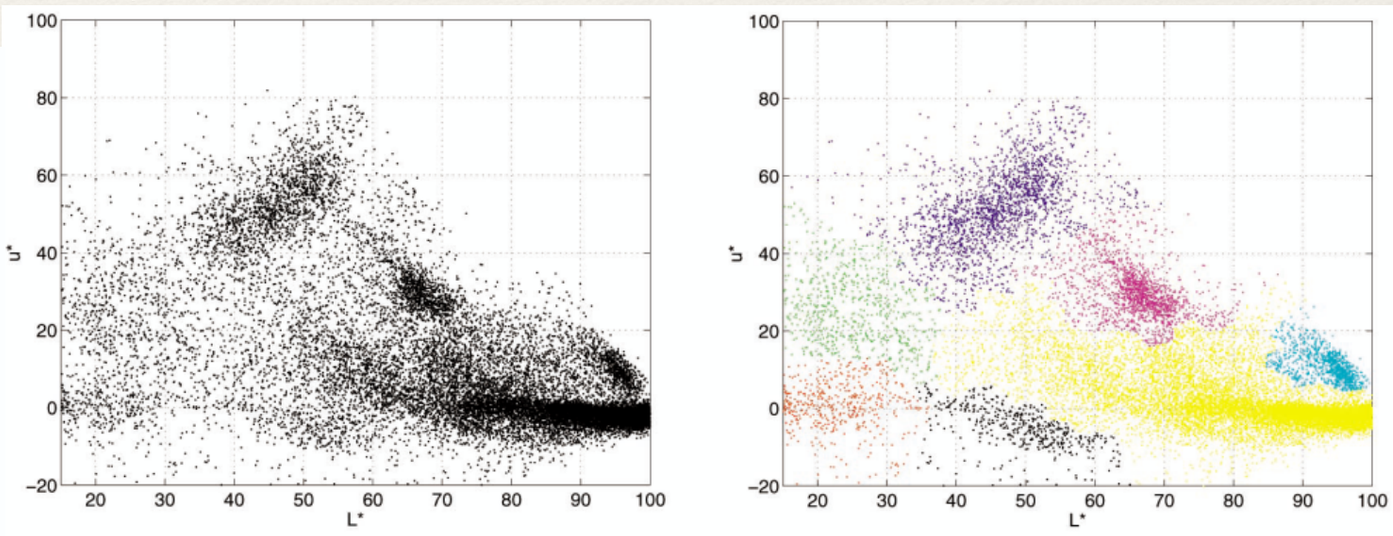
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



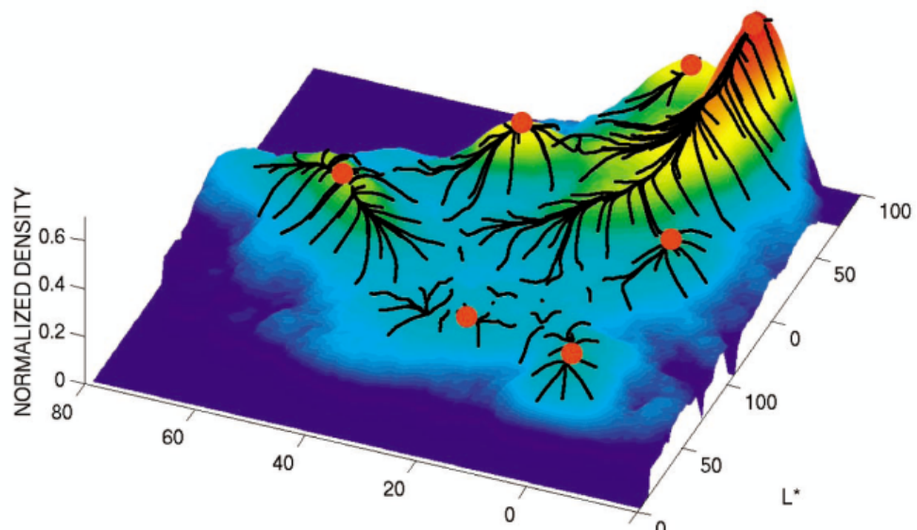
Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering/segmentation

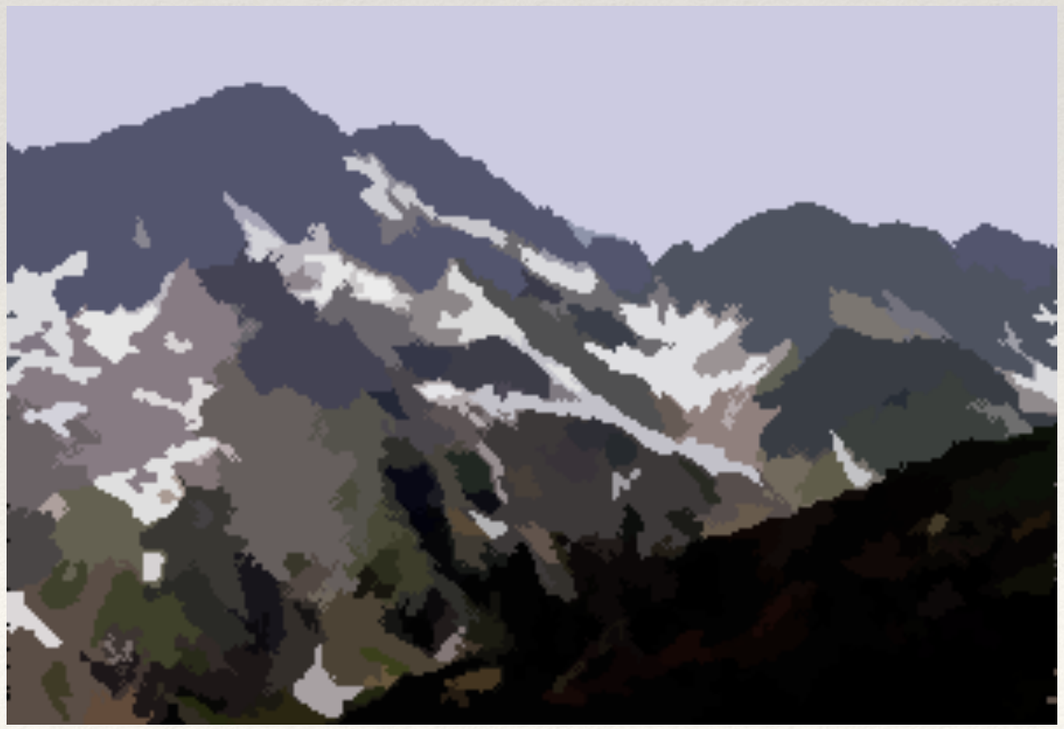
- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode



(a) (b)



Mean shift segmentation results



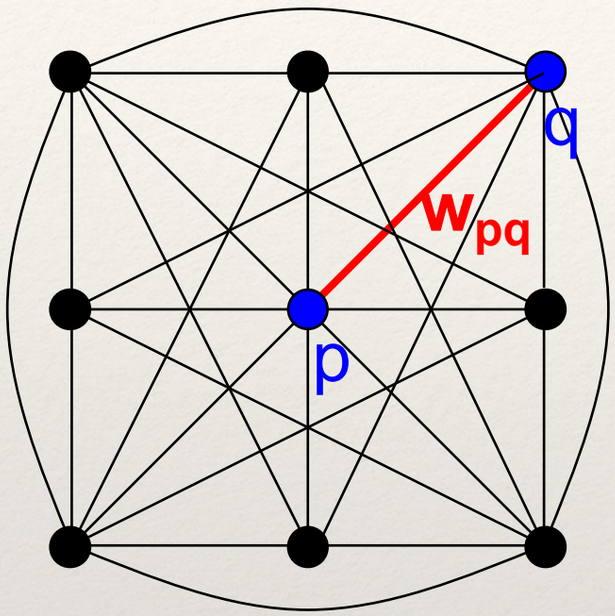
Mean shift segmentation results

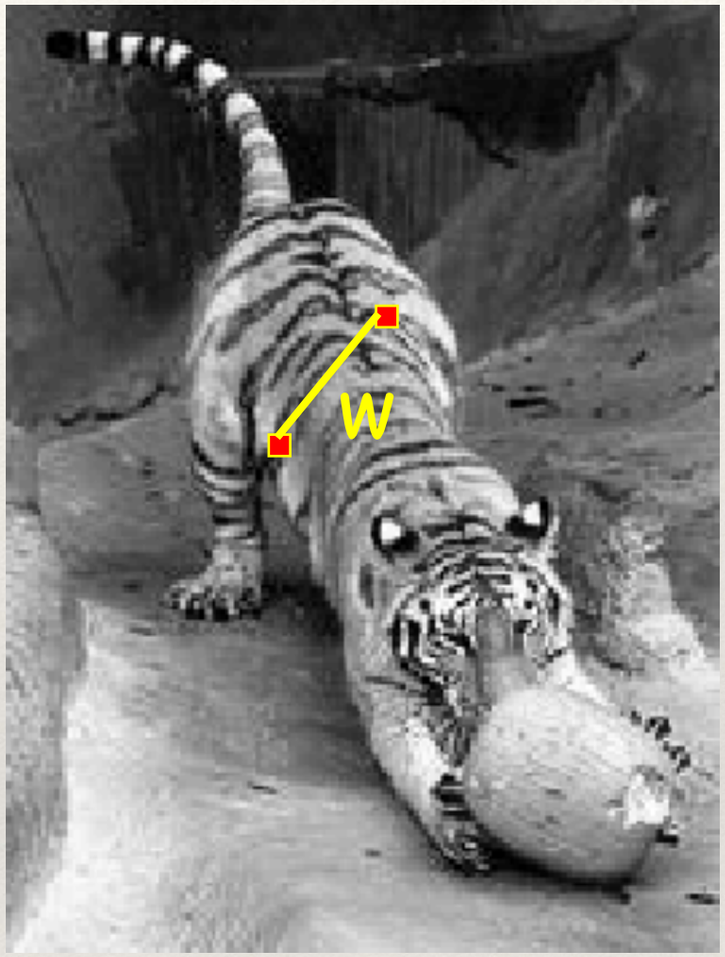
Mean shift segmentation results

Mean shift

- Pros:
  - Does not assume shape on clusters
  - One parameter choice (window size, aka “bandwidth”)
  - Generic technique
  - Find multiple modes
- Cons:
  - Selection of window size
  - Does not scale well with dimension of feature space

Images as graphs



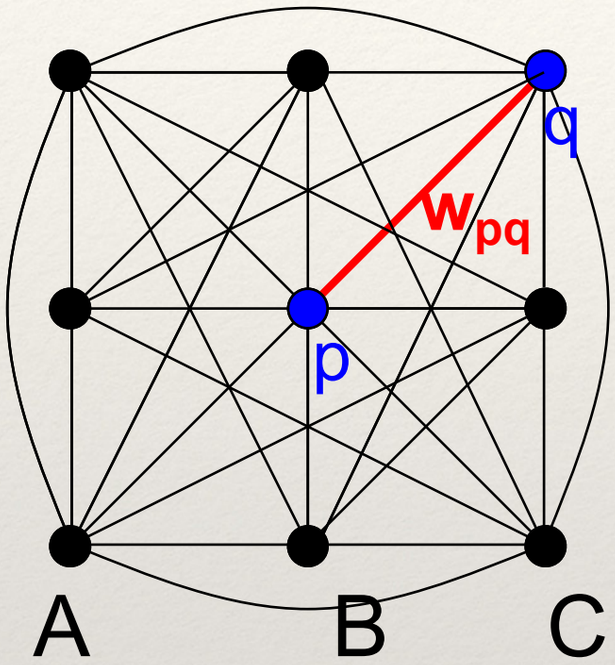


Fully-connected graph

- node (vertex) for every pixel
- link between *every* pair of pixels, **p**,**q**
- affinity weight $\mathbf{w_{pq}}$ for each link (edge)
  - $\mathbf{w_{pq}}$ measures *similarity*
    - similarity is *inversely proportional* to difference (in color and position…)

Source: Steve Seitz

Segmentation by Graph Cuts



Break Graph into Segments

- Want to 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

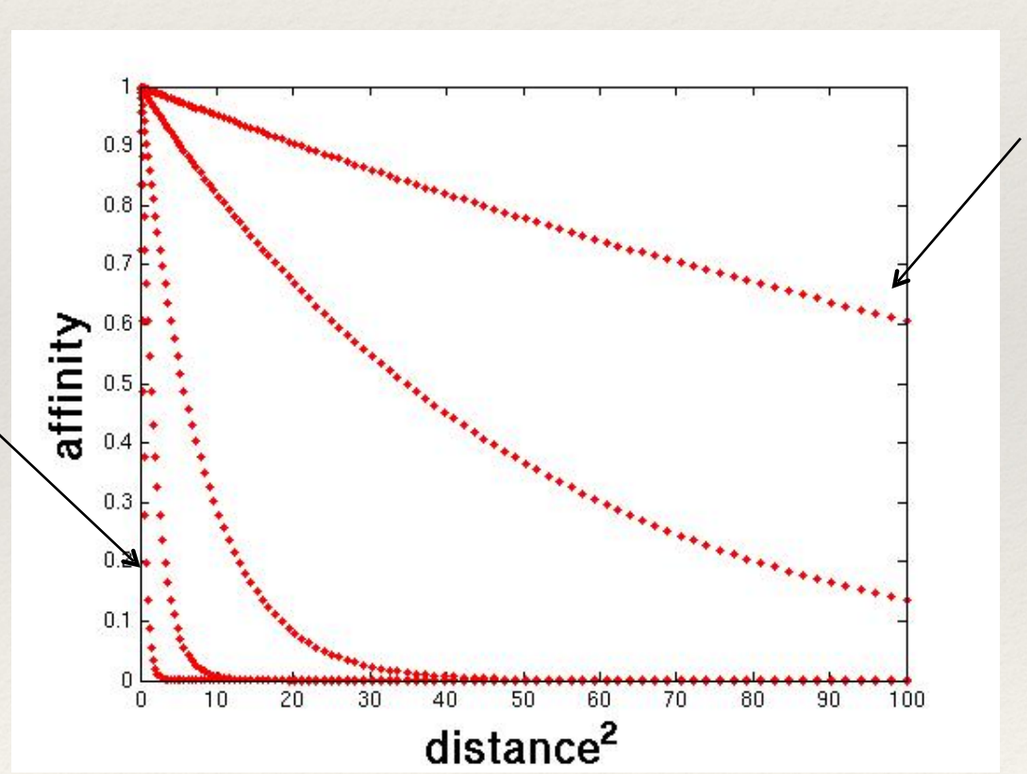
Source: Steve Seitz

Measuring affinity

- One possibility:

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

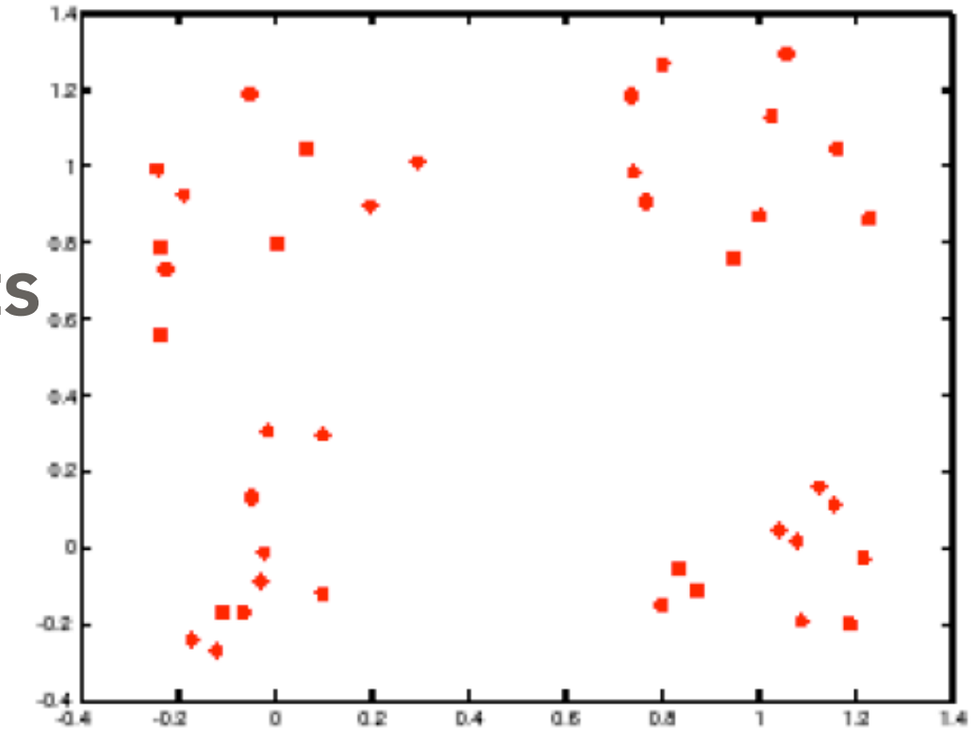
Small sigma: group only nearby points



Large sigma: group distant points

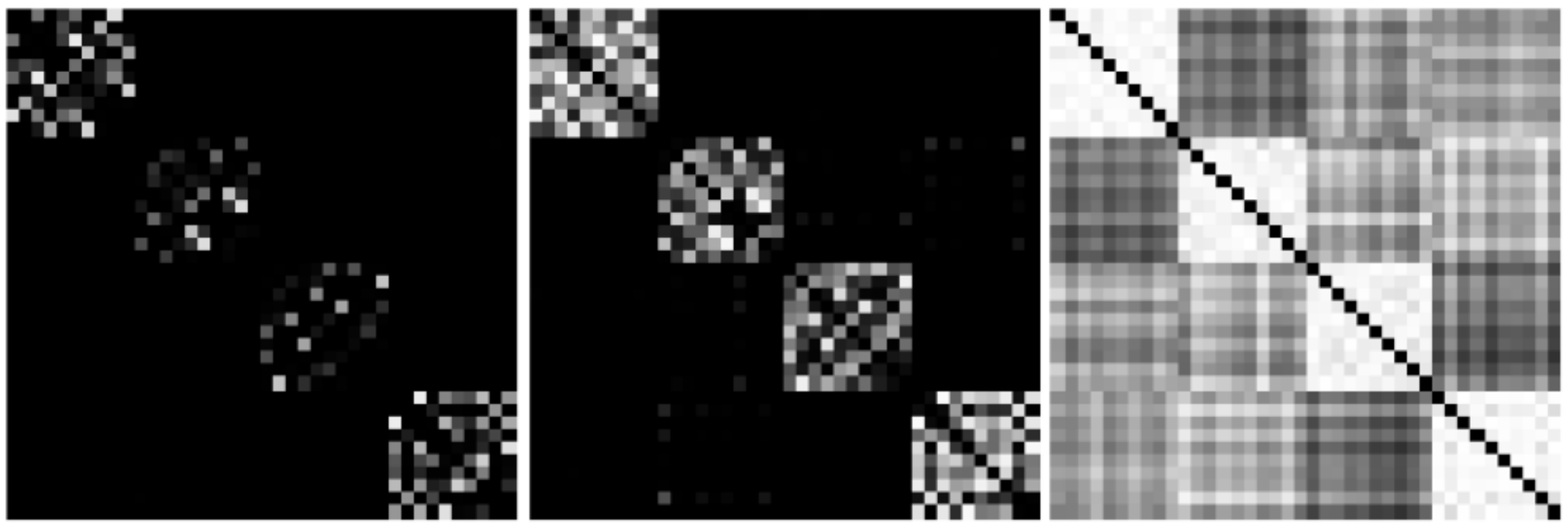
Measuring affinity

Data points



σ=.2

Affinity matrices



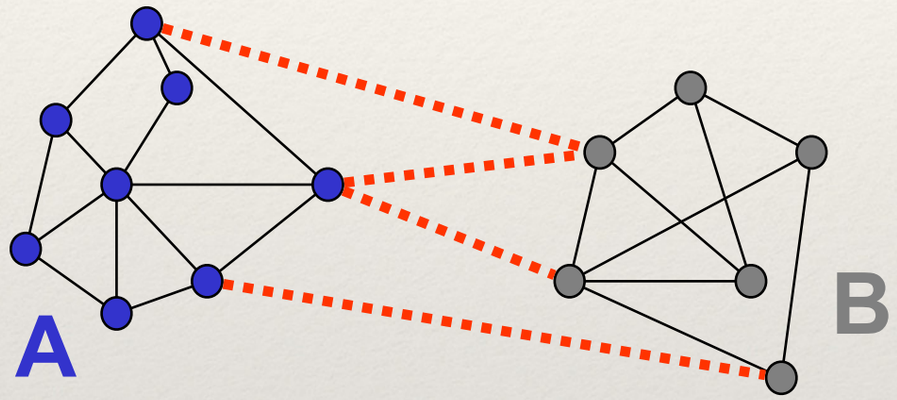
σ=.1 σ=.2 σ=1

Slide credit: Kristen Grauman

Cuts in a graph: Min cut

Link Cut

- set of links whose removal makes a graph disconnected
- cost of a cut:



$$cut(A,B) = \sum_{p \in A, q \in B} w_{p,q}$$

Find minimum cut

- gives you a segmentation
- fast algorithms exist for doing this

Minimum cut

- Problem with minimum cut:

  Weight of cut proportional to number of edges in the cut; tends to produce small, isolated components.

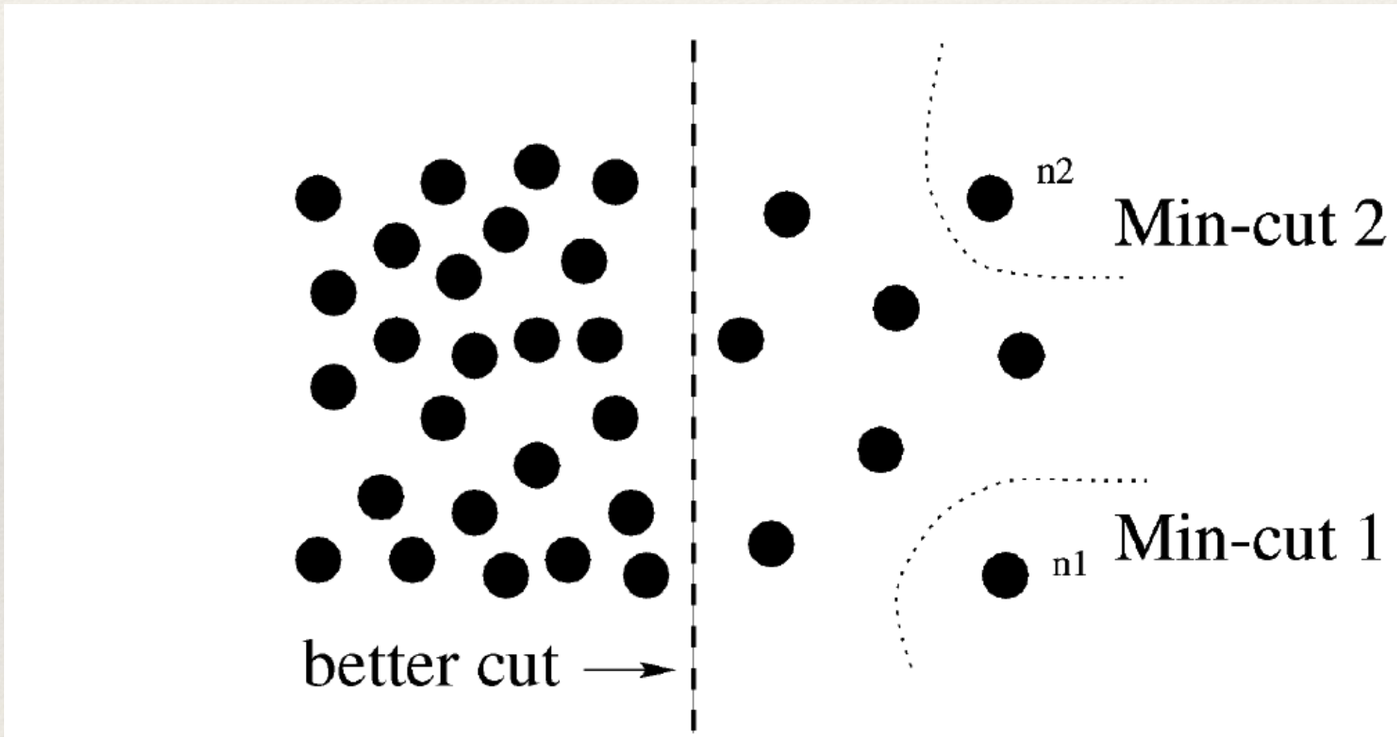
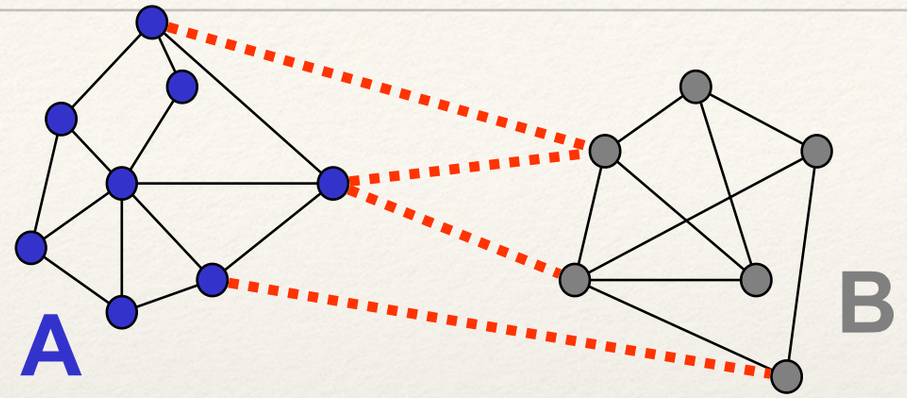


Fig. 1. A case where minimum cut gives a bad partition.

[Shi & Malik, 2000 PAMI]

Cuts in a graph: Normalized cut

Normalized Cut



- fix bias of Min Cut by **normalizing** for size of segments:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

assoc(A,V) = sum of weights of all edges that touch A

- Ncut value small when we get two clusters with many edges with high weights, and few edges of low weight between them
- Approximate solution for minimizing the Ncut value : generalized eigenvalue problem.

Source: Steve Seitz



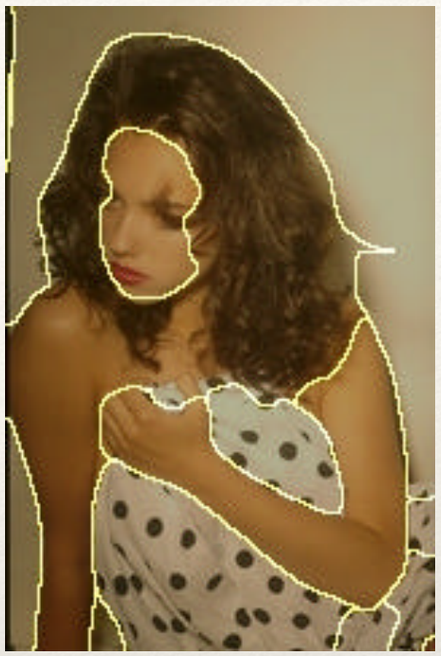


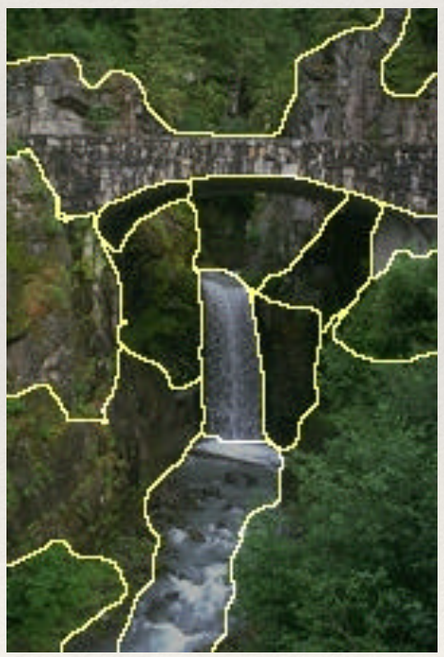


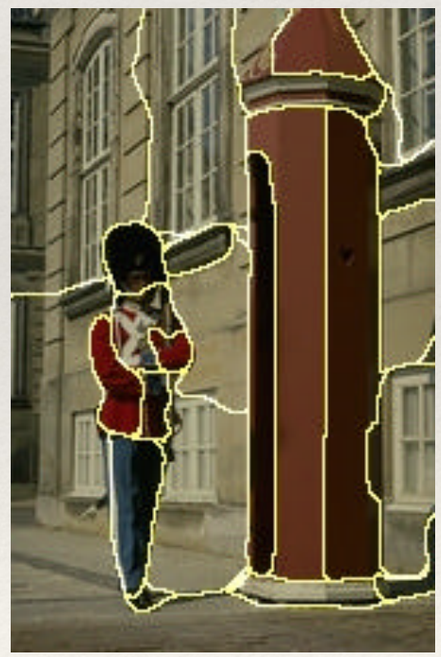










http://www.cs.berkeley.edu/~fowlkes/BSE/

Normalized cuts: pros and cons

Pros:

- Generic framework, flexible to choice of function that computes weights ("affinities") between nodes
- Does not require model of the data distribution

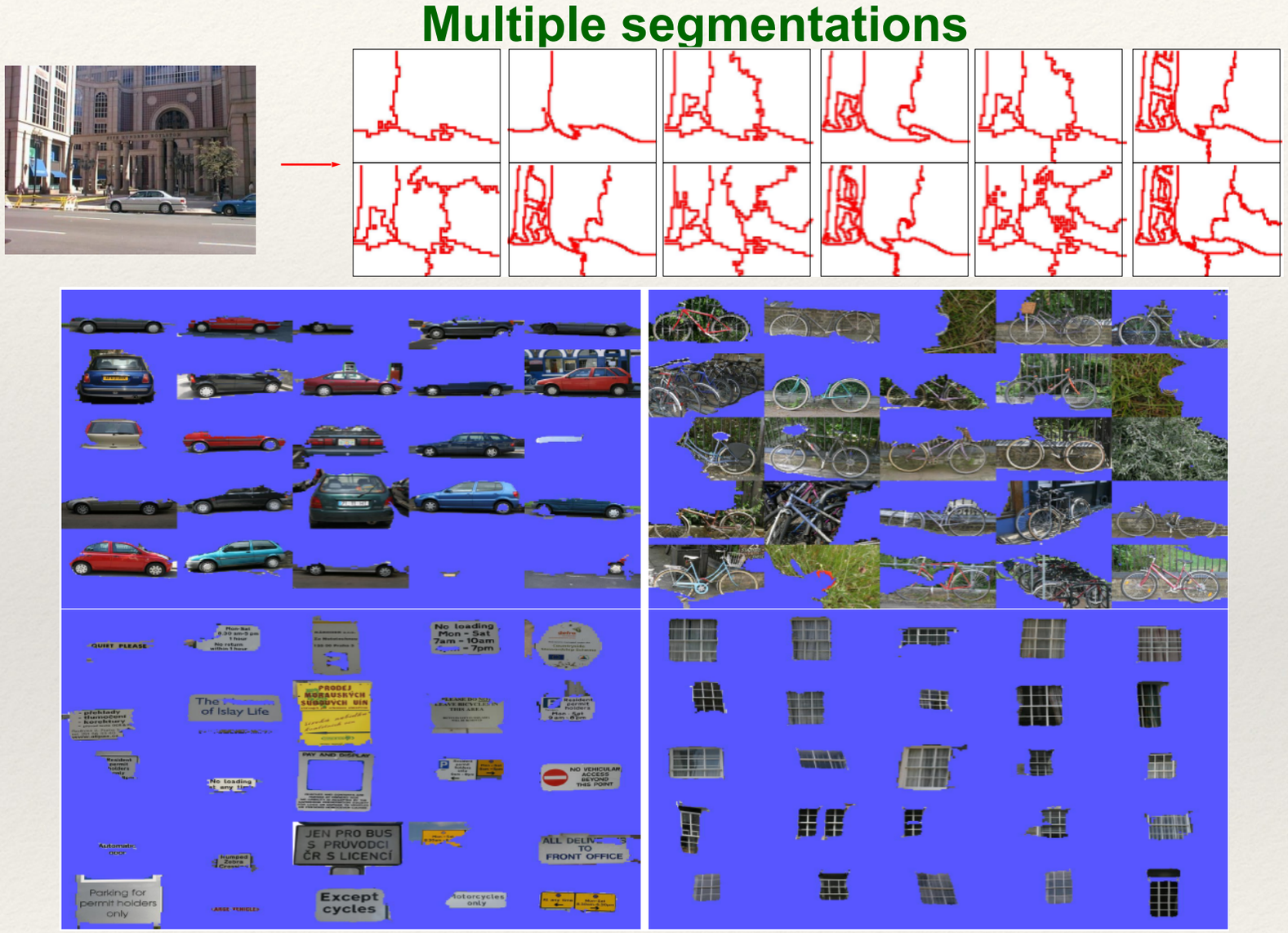
Cons:

- Time complexity can be high
  - Dense, highly connected graphs → many affinity computations
  - Solving eigenvalue problem
- Preference for balanced partitions

Summary

- Segmentation to find object boundaries or mid-level regions, tokens.
- Bottom-up segmentation via clustering
  - General choices -- features, affinity functions, and clustering algorithms
  - Grouping also useful for quantization, can create new feature summaries
    - Texton histograms for texture within local region
- Example clustering methods
  - K-means
  - Mean shift
  - Graph cut, normalized cuts

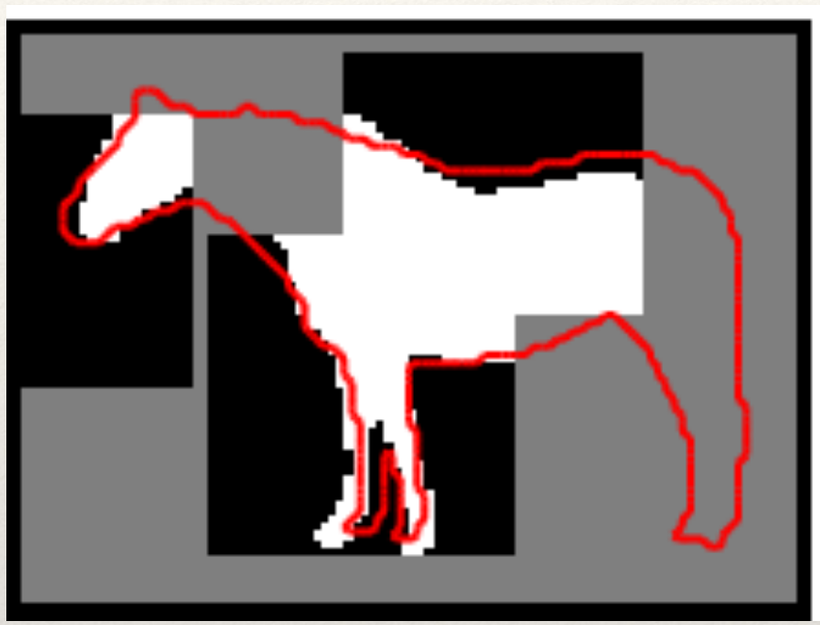
Segments as primitives for recognition

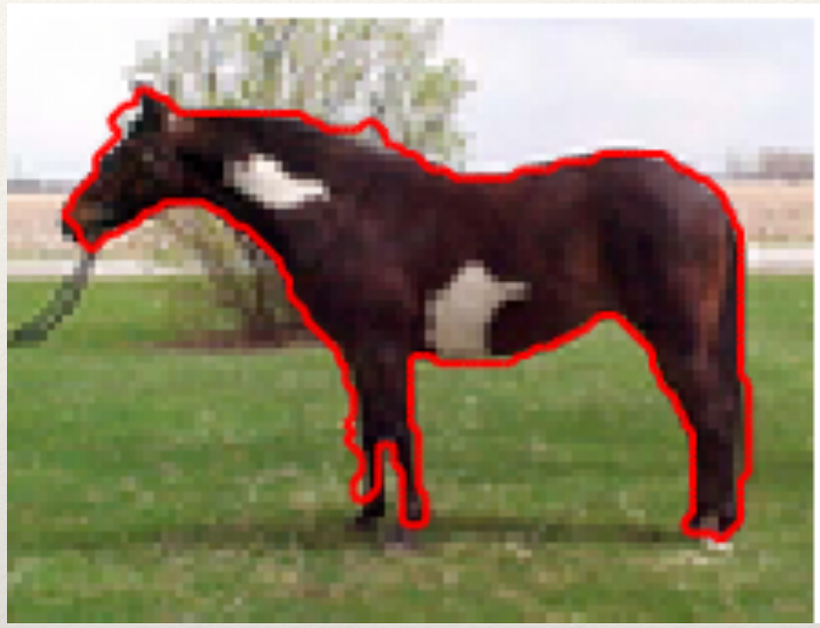


B. Russell et al., [“Using Multiple Segmentations to Discover Objects and their Extent in Image Collections,”](#) CVPR 2006

Slide credit: Lana Lazebnik

Top-down segmentation



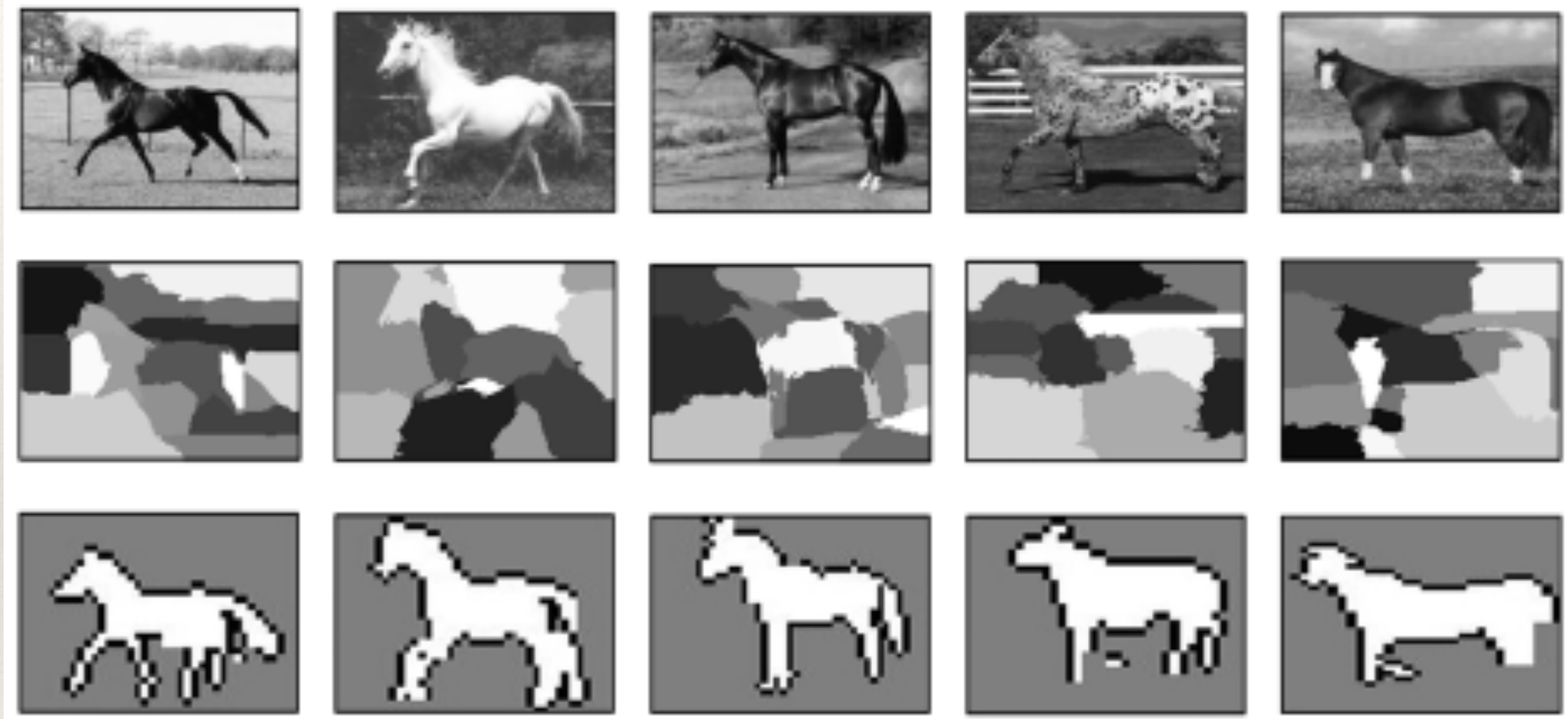


E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," ECCV 2002

A. Levin and Y. Weiss, "Learning to Combine Bottom-Up and Top-Down Segmentation," ECCV 2006.

Slide credit: Lana Lazebnik

Top-down segmentation



E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," ECCV 2002

A. Levin and Y. Weiss, "Learning to Combine Bottom-Up and Top-Down Segmentation," ECCV 2006.

Slide credit: Lana Lazebnik

Motion segmentation

Input sequence

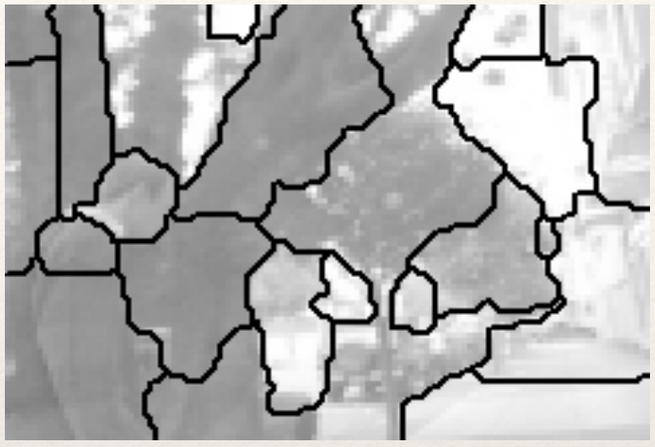
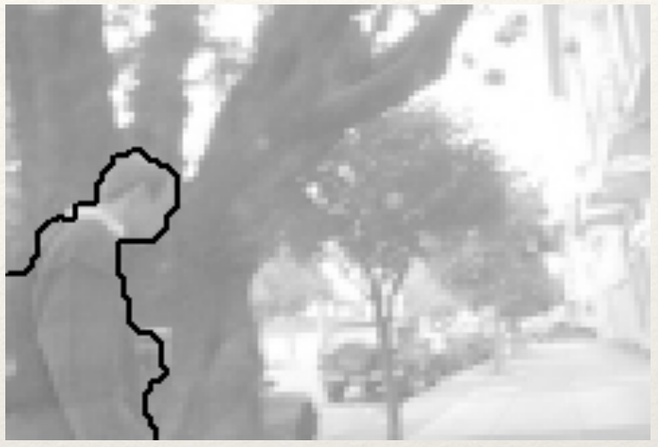


Image Segmentation



Motion Segmentation

Input sequence

Image Segmentation

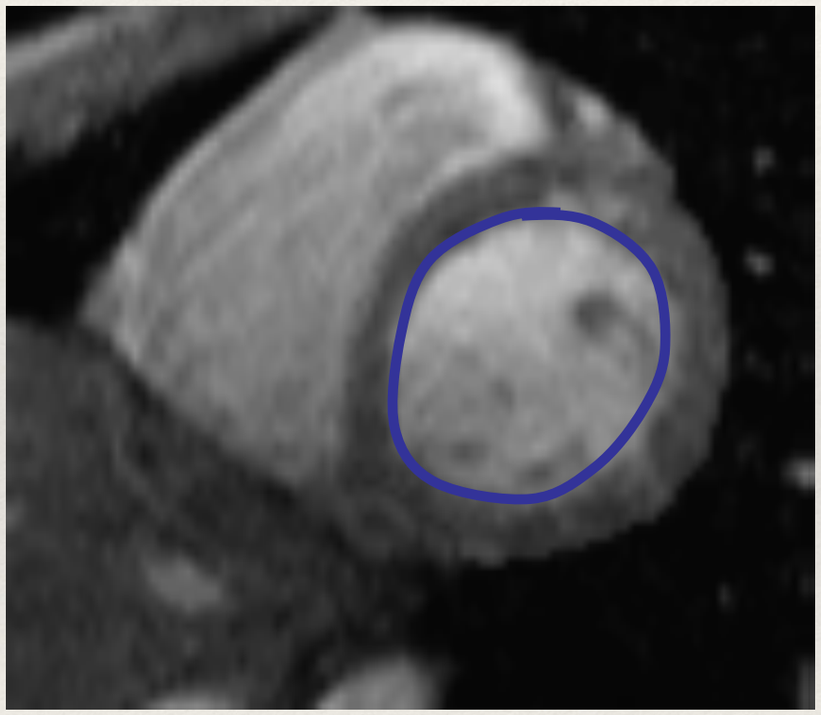
Motion Segmentation

A.Barbu, S.C. Zhu. Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities, *IEEE Trans. PAMI*, August 2005.

Deformable contours

Given: initial contour (model) near desired object

Goal: evolve the contour to fit exact object boundary



Main idea: elastic band is iteratively adjusted so as to

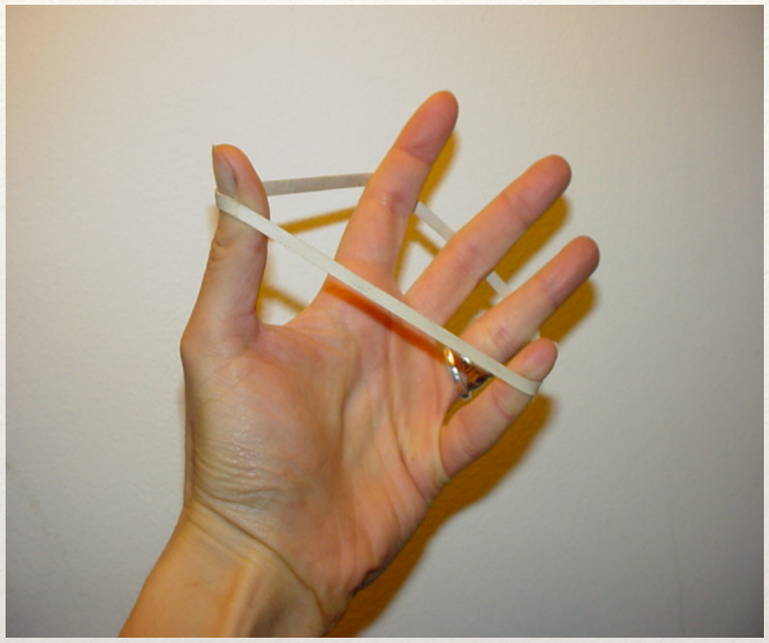
- be near image positions with high gradients, **and**
- satisfy shape "preferences" or contour priors

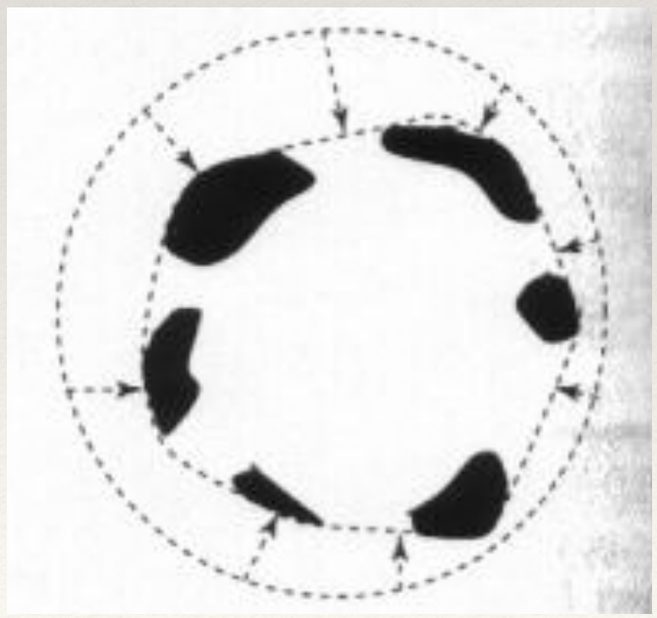
a.k.a. active contours, snakes

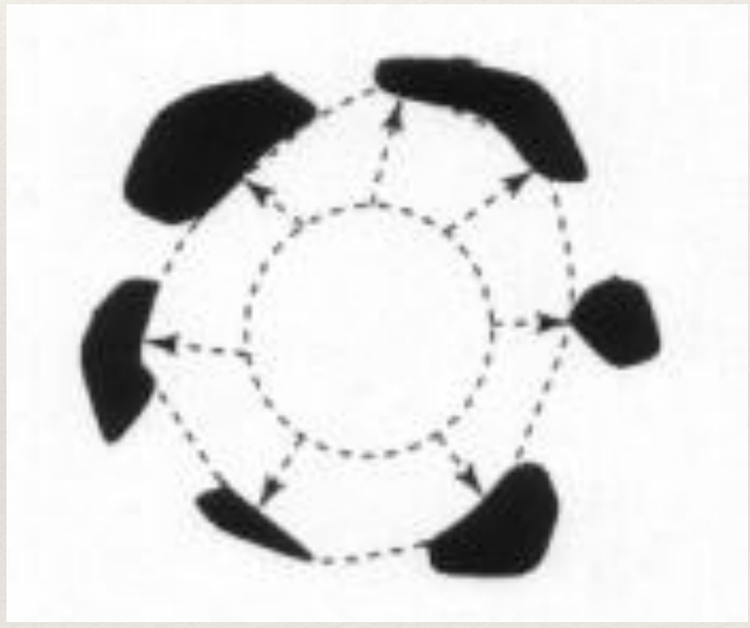
[Snakes: Active contour models, Kass, Witkin, & Terzopoulos, ICCV1987]

Figure credit: Yuri Boykov

Deformable contours: intuition





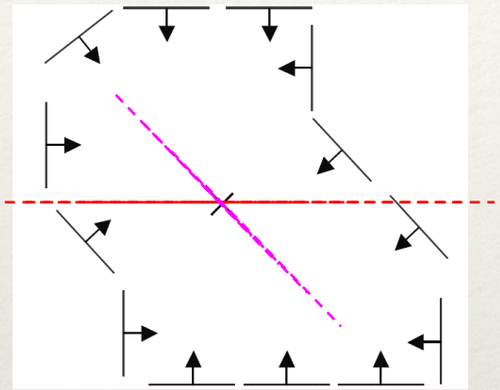


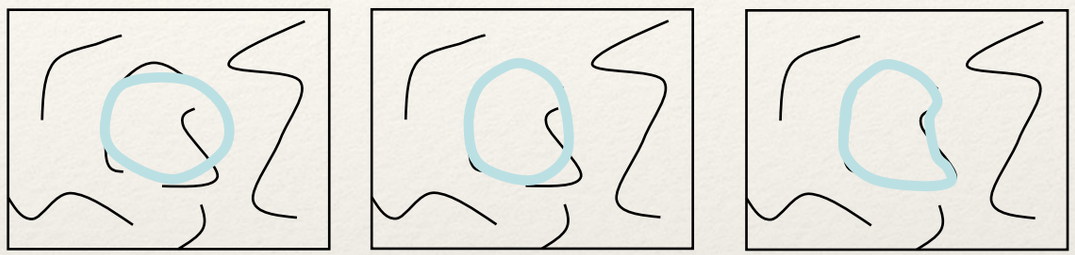
Slide credit: Kristen Grauman

Image from http://www.healthline.com/blogs/exercise_fitness/uploaded_images/HandBand2-795868.JPG

Deformable contours vs. Hough

Like generalized Hough transform, useful for shape fitting; but





Hough

Rigid model shape

Single voting pass can detect multiple instances

Deformable contours

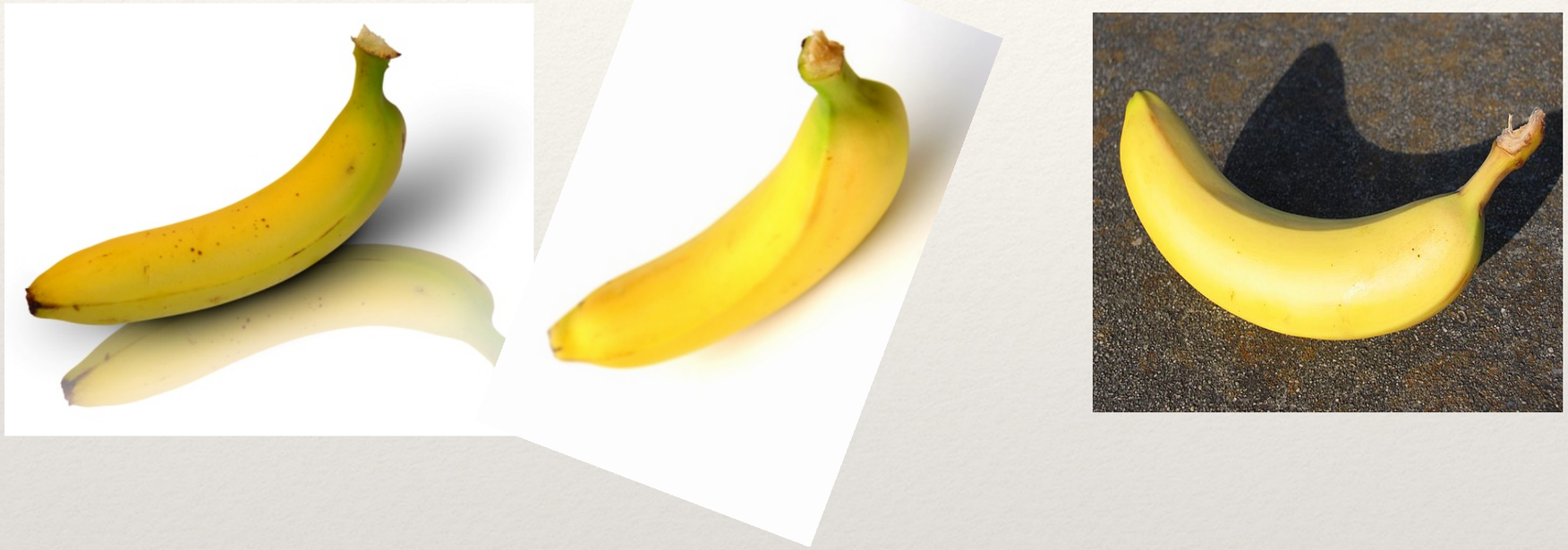
Prior on shape types, but shape iteratively adjusted (*deforms*)

Requires initialization nearby

One optimization “pass” to fit a single contour

Slide credit: Kristen Grauman

Why do we want to fit deformable shapes?



- Some objects have similar basic form but some variety in the contour shape.

Slide credit: Kristen Grauman

Why do we want to fit deformable shapes?

- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands…

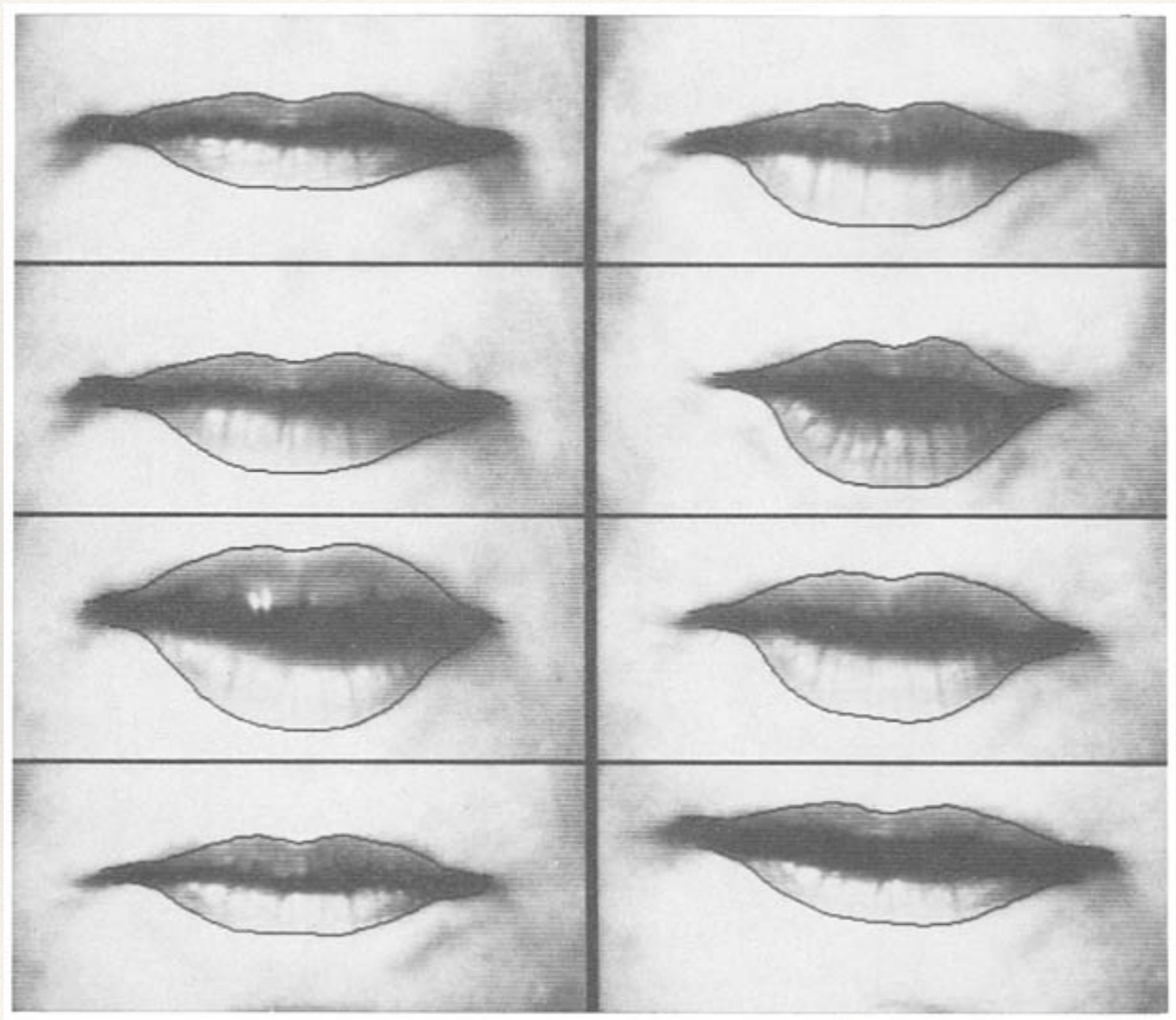
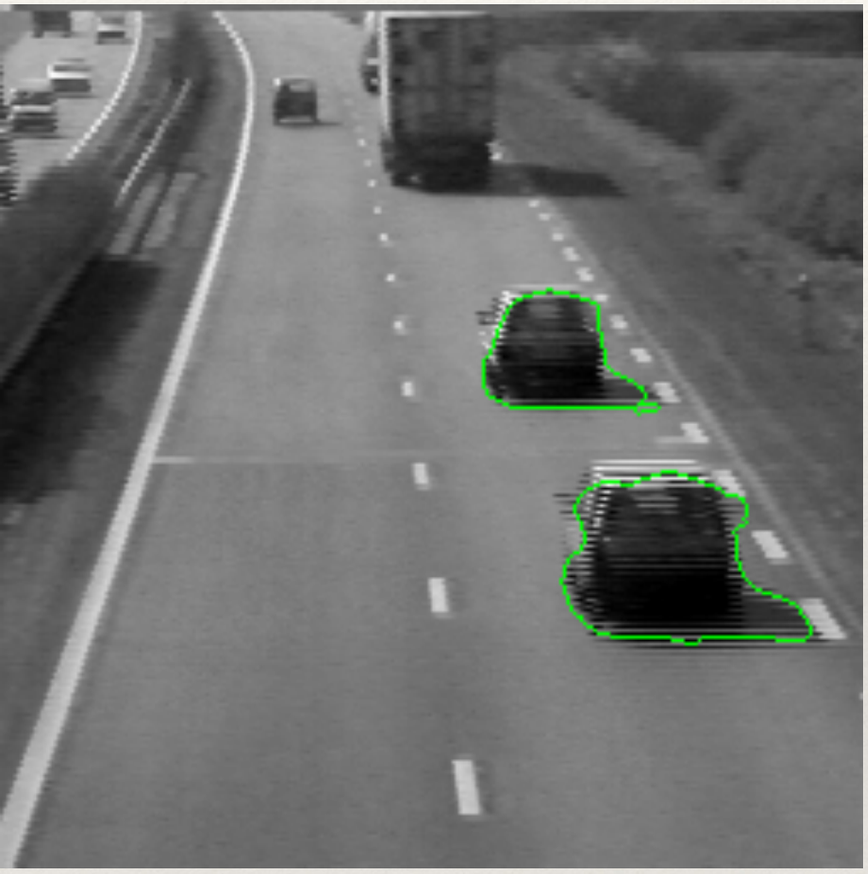
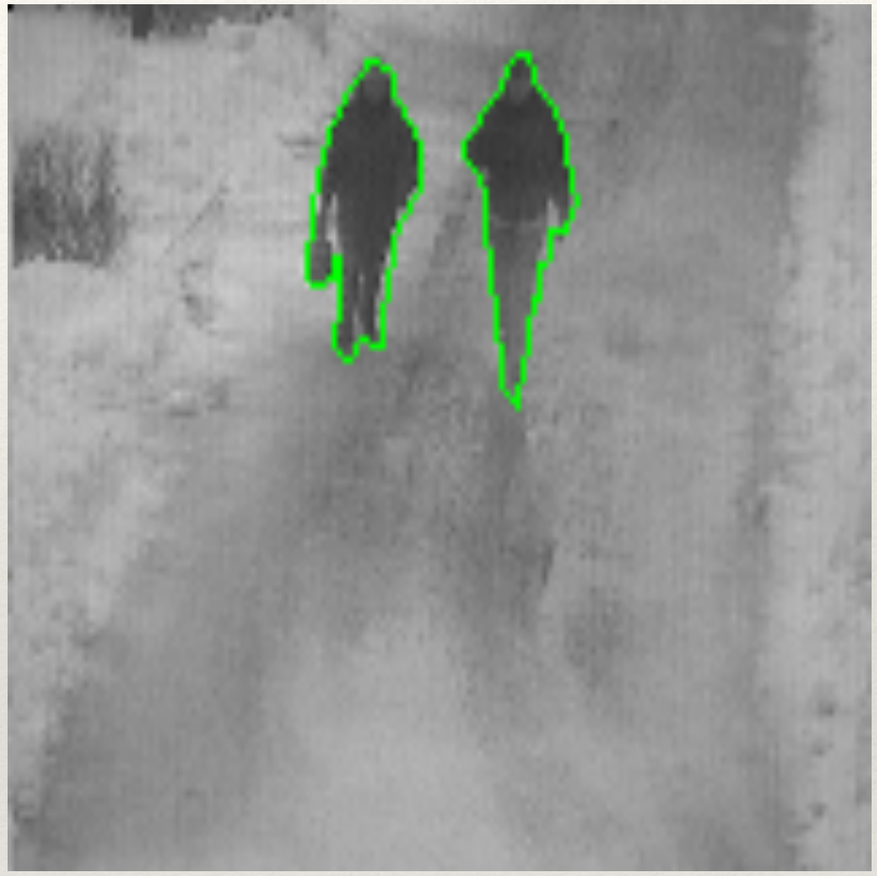


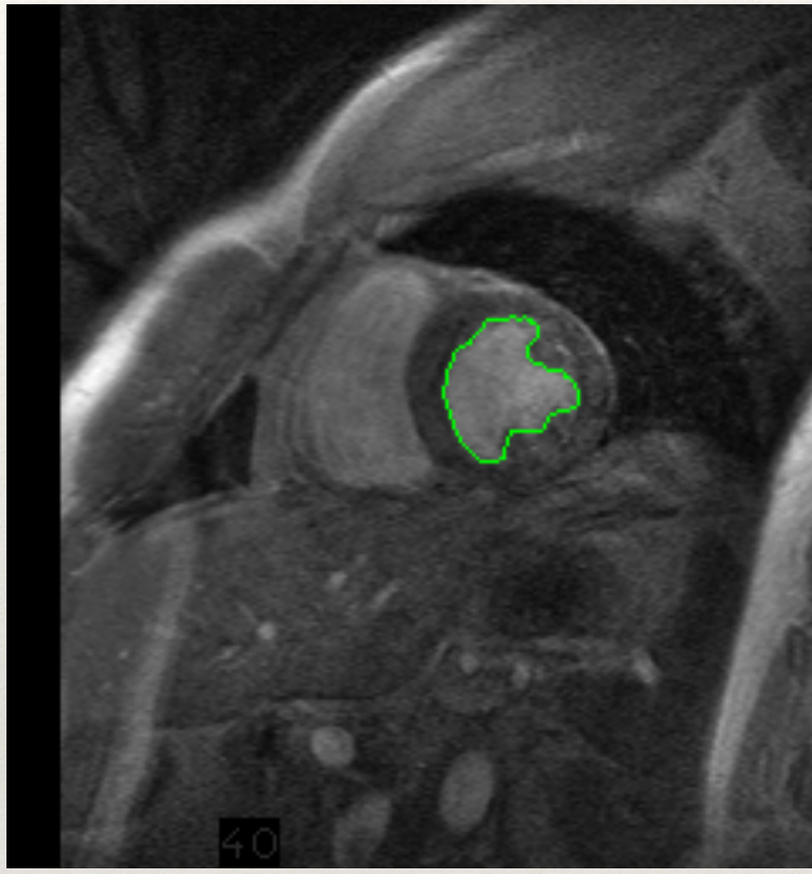
Figure from Kass et al. 1987

Slide credit: Kristen Grauman

Why do we want to fit deformable shapes?







- Non-rigid, deformable objects can change their shape over time.

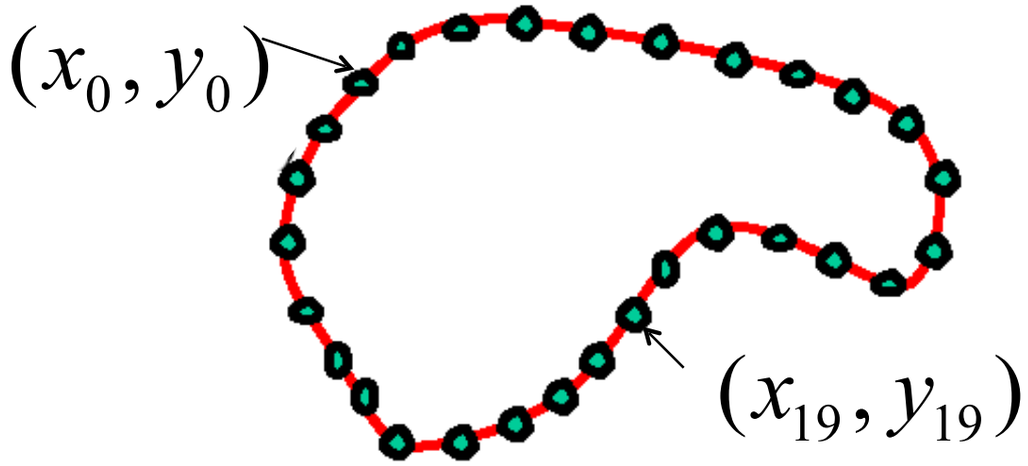
Figure credit: Julien Jomier

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

Representation

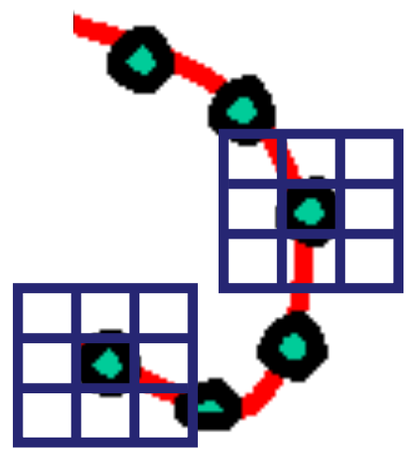
- We'll consider a discrete representation of the contour, consisting of a list of 2d point positions ("vertices").



$$\mathbf{v}_i = (x_i, y_i),$$

for $i = 0, 1, \ldots, n-1$

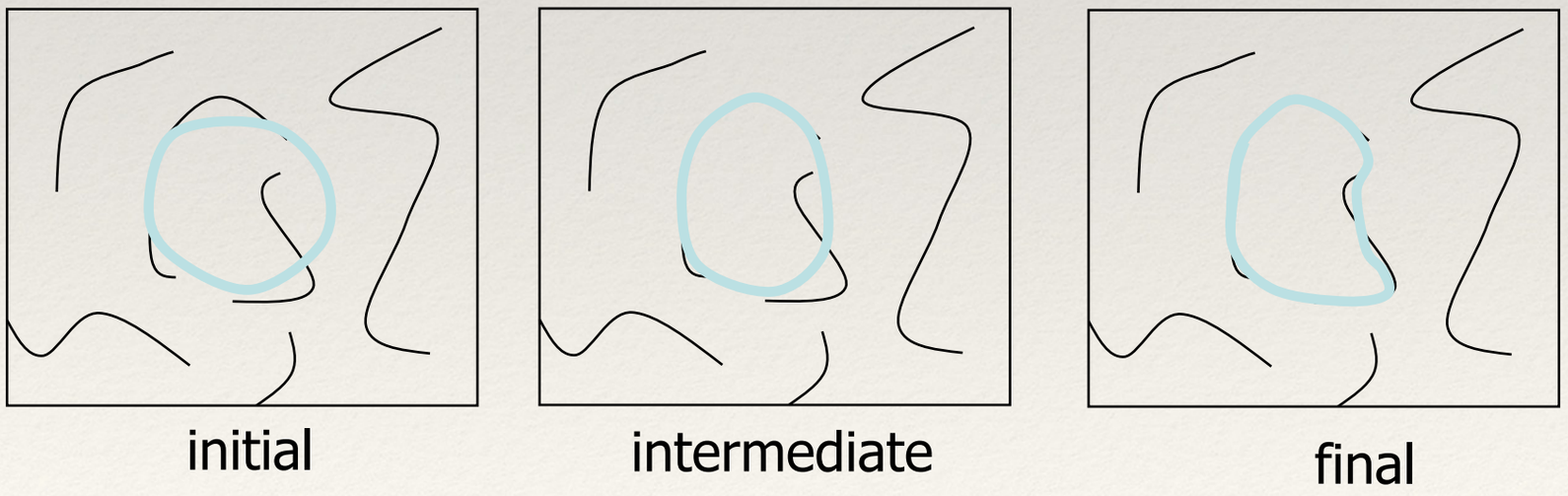
- At each iteration, we'll have the option to move each vertex to another nearby location ("state").



Fitting deformable contours

How should we adjust the current contour to form the new contour at each iteration?

- Define a cost function ("energy" function) that says how good a candidate configuration is.
- Seek next configuration that minimizes that cost function.



Energy function

The total energy (cost) of the current snake is defined as:

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

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

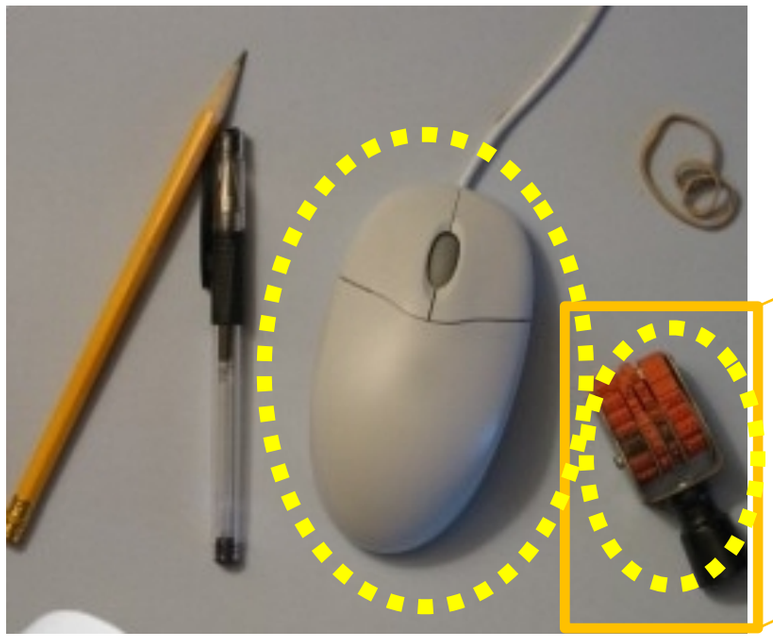
External energy ("image" energy): encourage contour to fit on places where image structures exist, e.g., edges.

A good fit between the current deformable contour and the target shape in the image will yield a **low** value for this cost function.

External energy: intuition

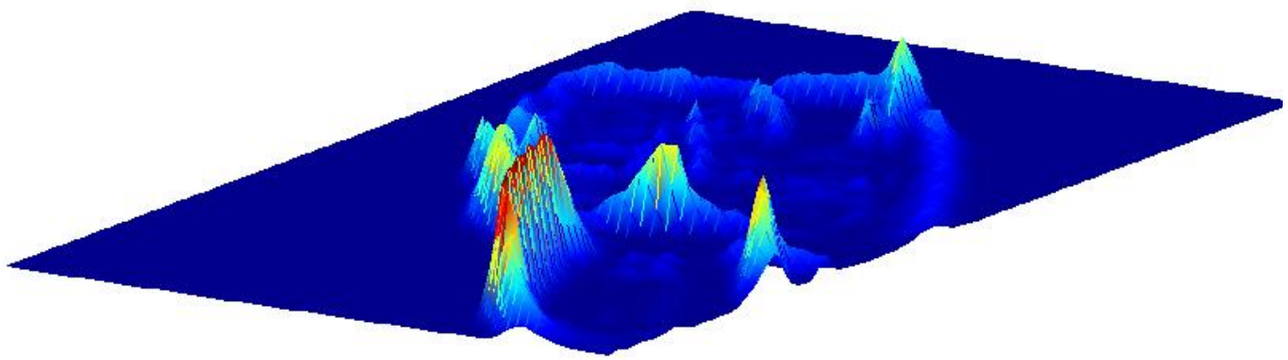
- Measure how well the curve matches the image data
- "Attract" the curve toward different image features
  - Edges, lines, texture gradient, etc.

External image energy



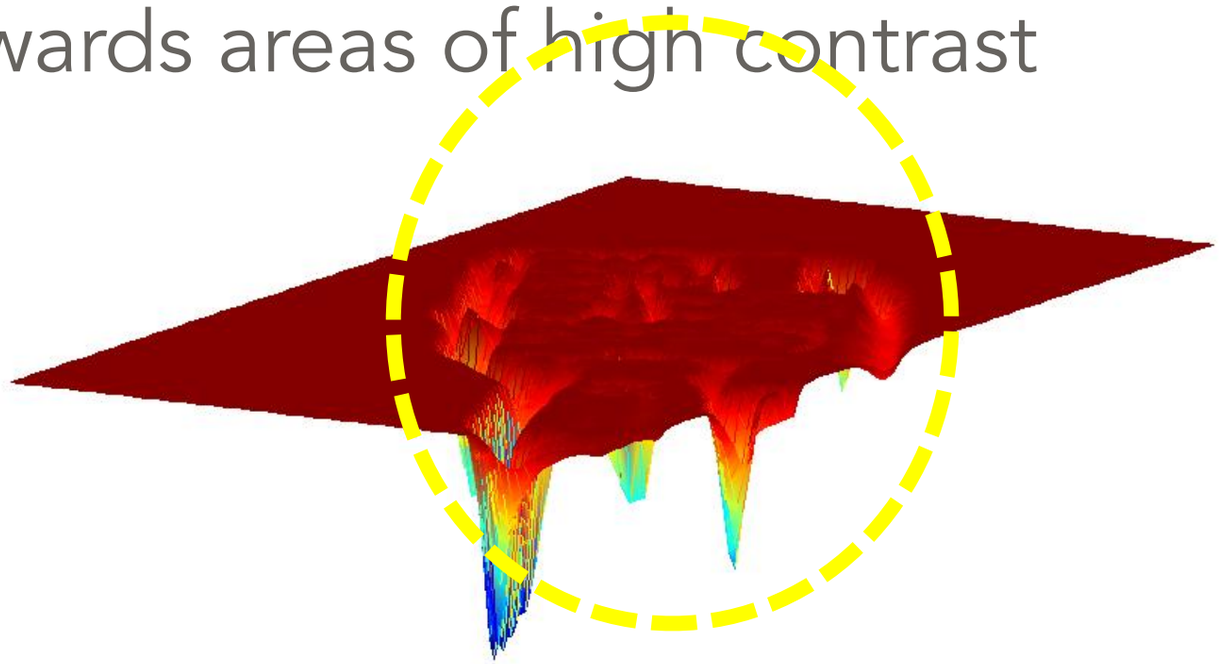
How do edges affect "snap" of rubber band?

Think of external energy from image as gravitational pull towards areas of high contrast



Magnitude of gradient

$$G_x(I)^2 + G_y(I)^2$$



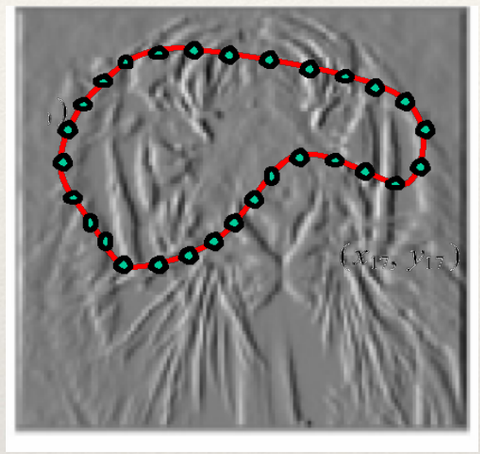
- (Magnitude of gradient)

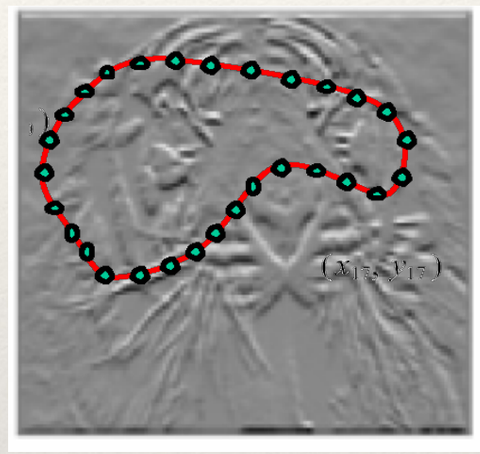
$$-\left(G_x(I)^2 + G_y(I)^2\right)$$

Slide credit: Kristen Grauman

External image energy

- Gradient images $G_x(x,y)$ and $G_y(x,y)$





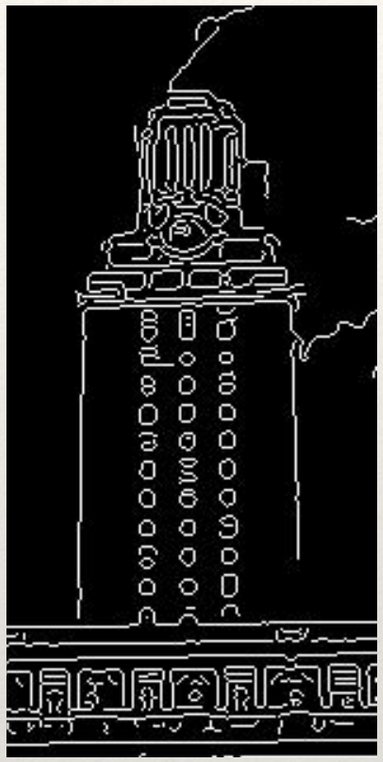
- External energy at a point on the curve is:

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

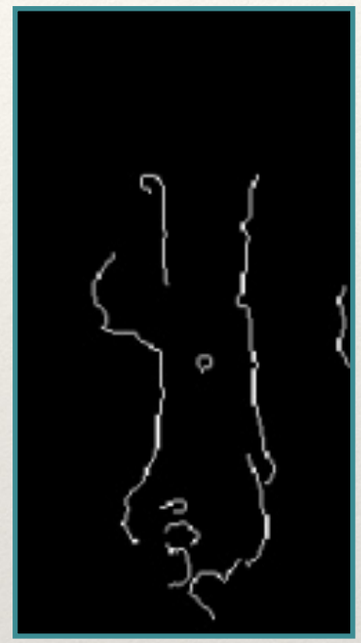
- External energy for the whole curve:

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

Internal energy: intuition



What are the underlying boundaries in this fragmented edge image?

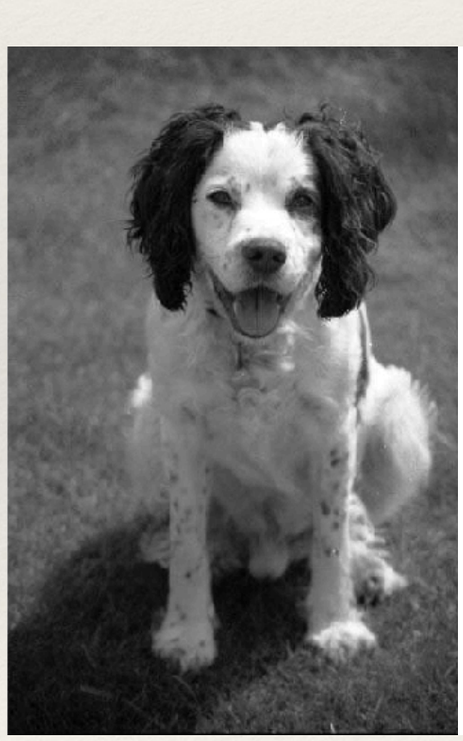


And in this one?

Slide credit: Kristen Grauman

Internal energy: intuition

A priori, we want to favor **smooth** shapes, contours with **low curvature**, contours similar to a **known shape**, etc. to balance what is actually observed (i.e., in the gradient image).



Slide credit: Kristen Grauman

Internal energy

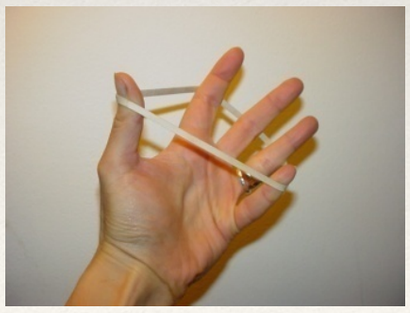
For a *continuous* curve, a common internal energy term is the "bending energy".

At some point *v(s)* on the curve, this is:

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

Tension, Elasticity

Stiffness, Curvature

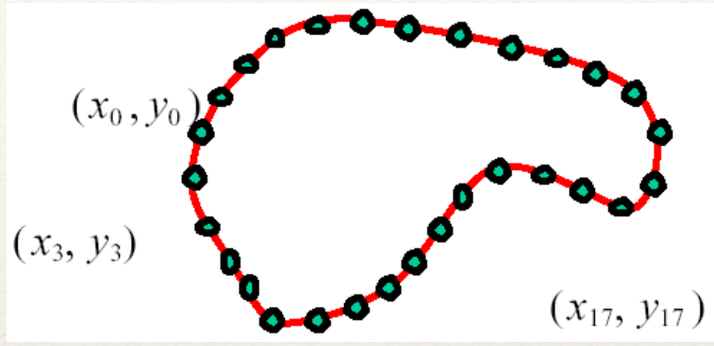


Slide credit: Kristen Grauman

Internal energy

- For our discrete representation,

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



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

*Note these are derivatives relative to **position**---not spatial image gradients.*

- Internal energy for the whole curve:

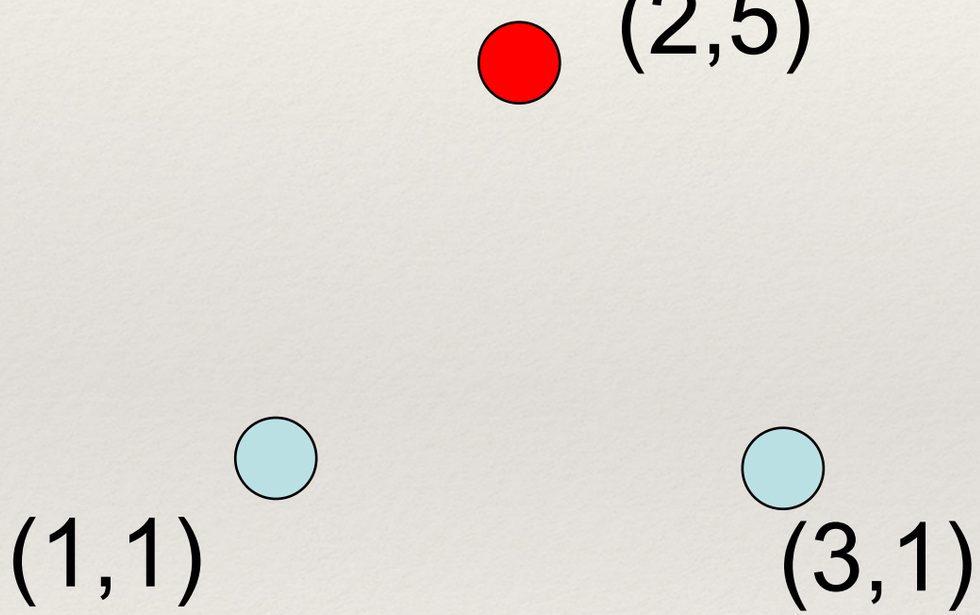
$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left\| \mathbf{v}_{i+1} - \mathbf{v}_i \right\|^2 + \beta \left\| \mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1} \right\|^2$$

*Why do these reflect **tension** and **curvature**?*

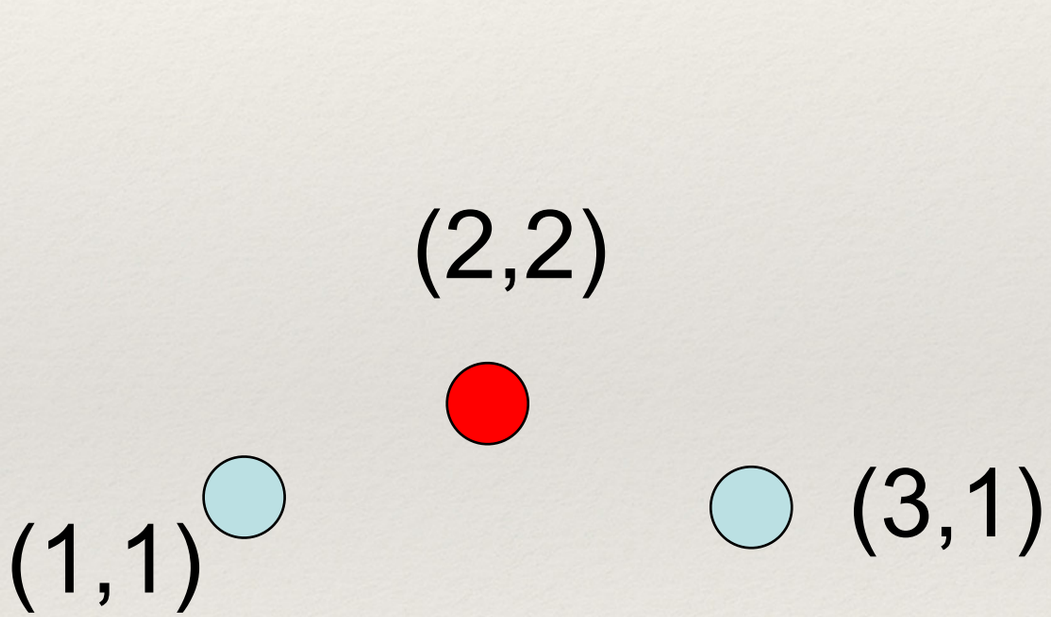
Example: compare curvature

$$E_{curvature}(v_i) = \|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$$

$$= (x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2$$



$$(3 - 2(2) + 1)2 + (1 - 2(5) + 1)2$$
$$= (-8)2 = 64$$



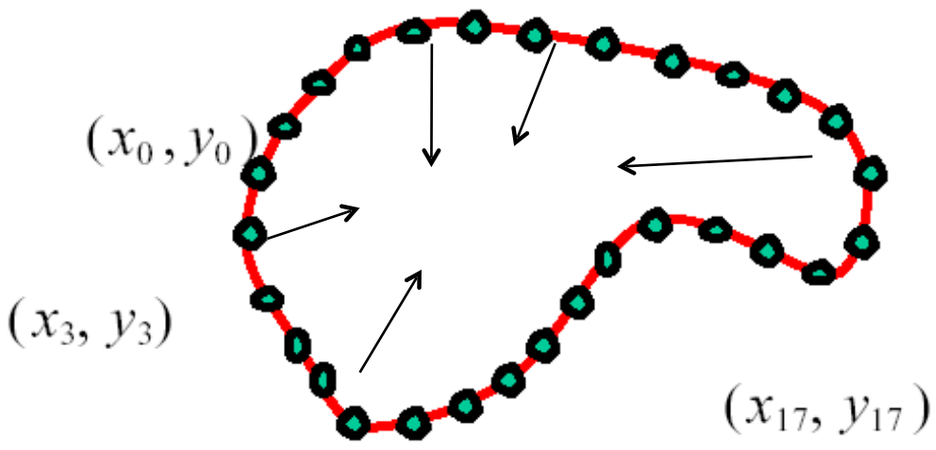
$$(3 - 2(2) + 1)2 + (1 - 2(2) + 1)2$$
$$= (-2)2 = 4$$

Slide credit: Kristen Grauman

Penalizing elasticity

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

$$= \alpha \cdot \sum_{i=0}^{n-1} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$



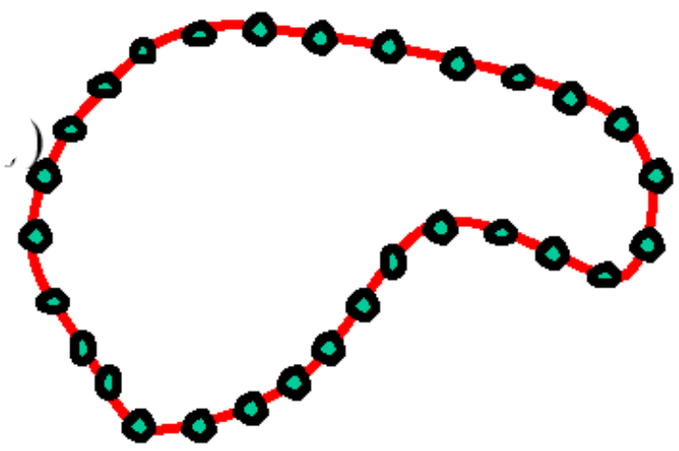
What is the possible problem with this definition?

Penalizing elasticity

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

Instead:

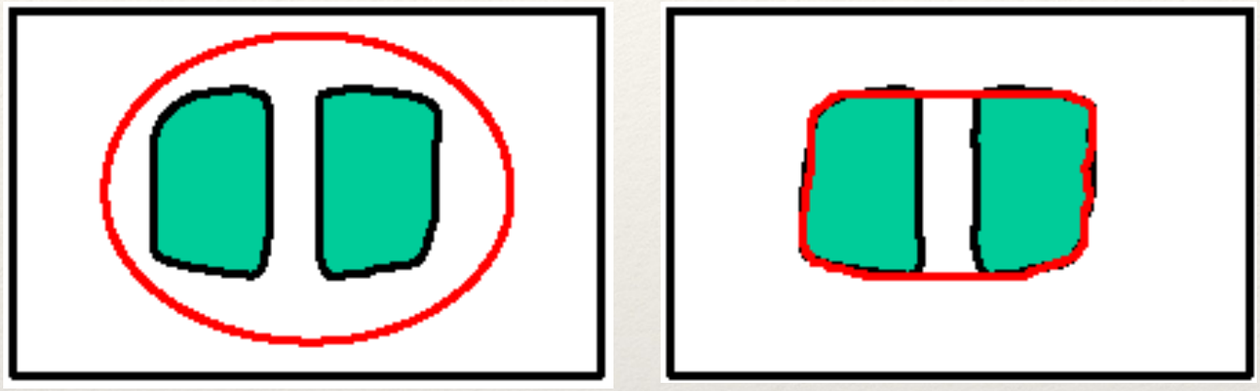
$$= \alpha \cdot \sum_{i=0}^{n-1} \left((x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 - \bar{d} \right)^2$$

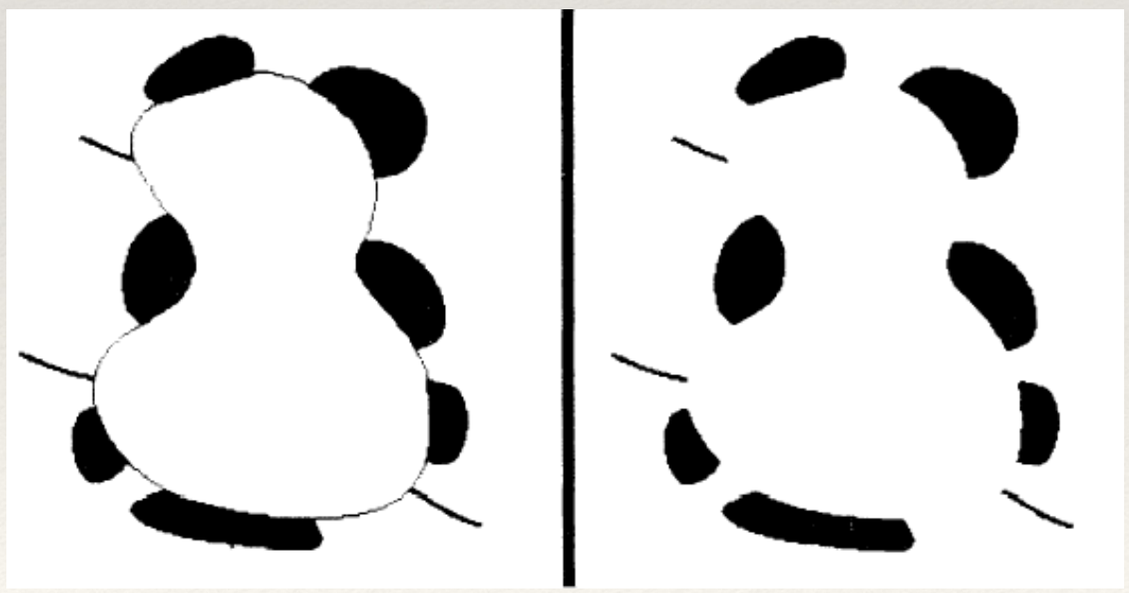


where *d* is the average distance between pairs of points – updated at each iteration.

Dealing with missing data

- The preferences for low-curvature, smoothness help deal with missing data:



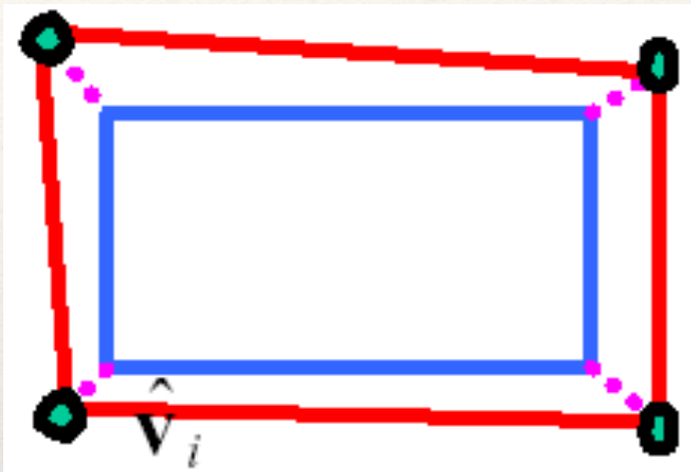


[Figure from Kass et al. 1987]

Illusory contours found!

Extending the internal energy: capture shape prior

- If object is some smooth variation on a known shape, we can use a term that will penalize deviation from that shape:



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

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

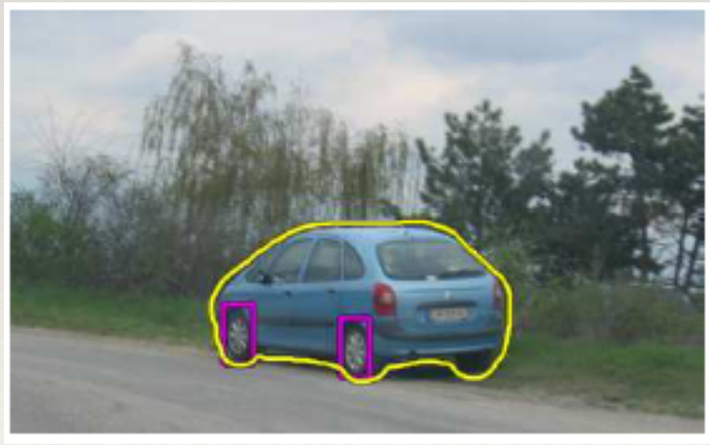


Fig from Y. Boykov

Total energy: function of the weights

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

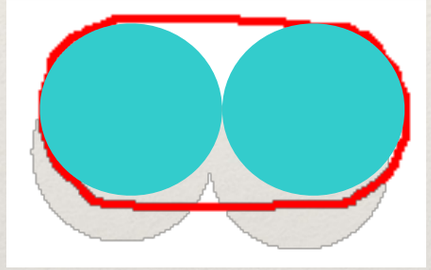
$$E_{external} = -\sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

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

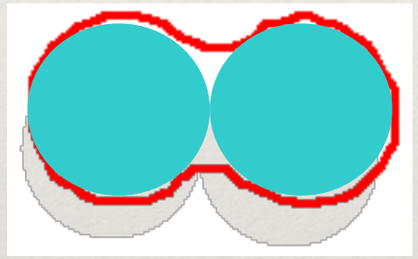
Total energy: function of the weights

e.g., weight controls the penalty for internal elasticity

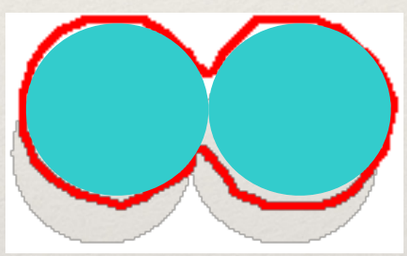
α



large α



medium α

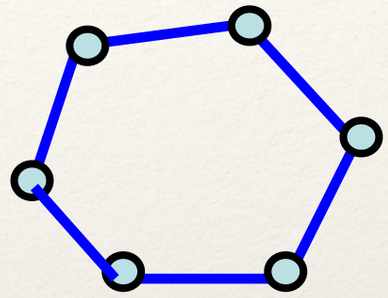


small α

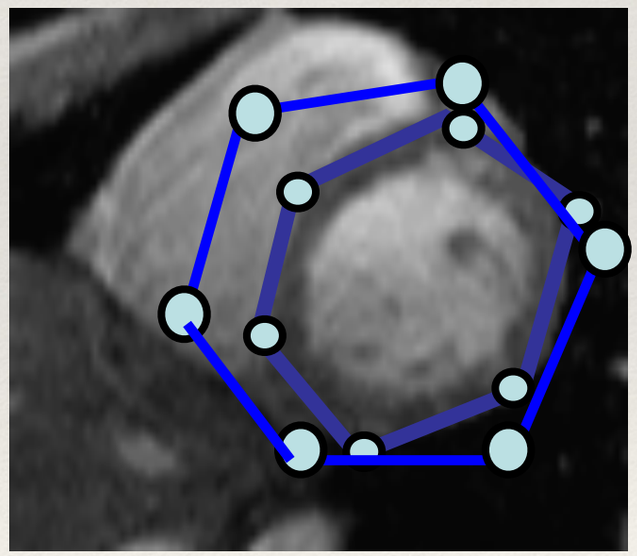
Fig from Y. Boykov

Recap: deformable contour

- A simple elastic snake is defined by:
  - A set of *n* points,
  - An internal energy term (tension, bending, plus optional shape prior)
  - An external energy term (gradient-based)



- To use to segment an object:
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy

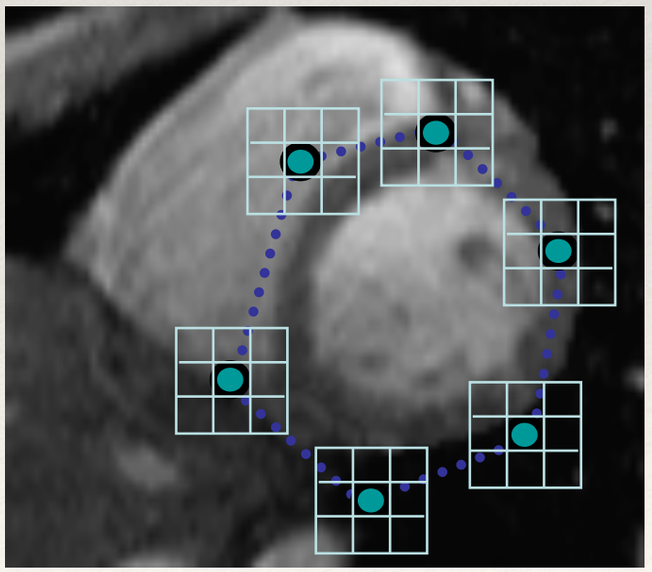


Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We'll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

Energy minimization: greedy

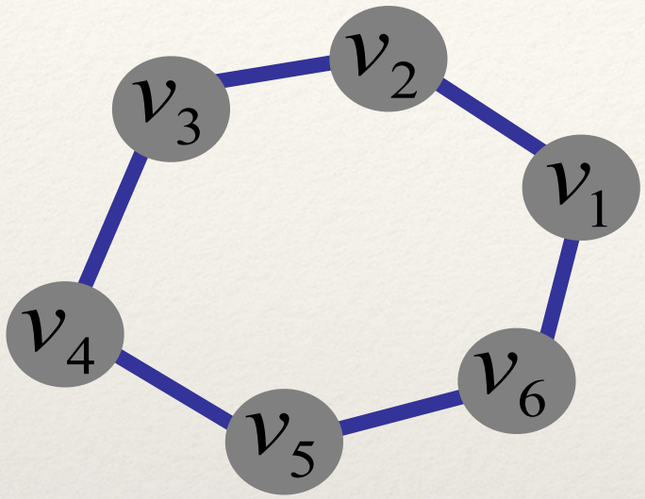
- For each point, search window around it and move to where energy function is minimal
  - Typical window size, e.g., 5 x 5 pixels
- Stop when predefined number of points have not changed in last iteration, or after max number of iterations
- Note:
  - Convergence not guaranteed
  - Need decent initialization

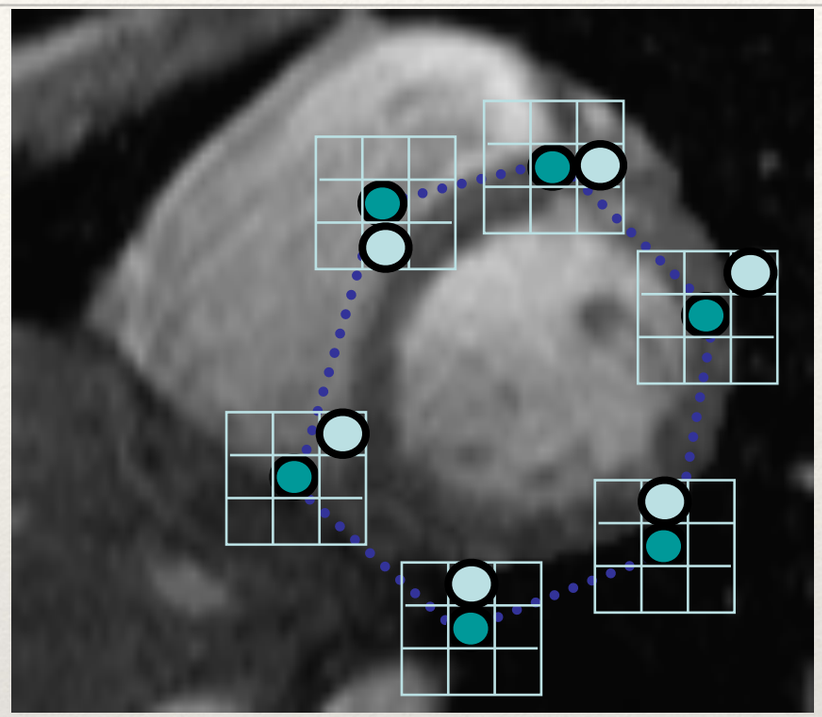


Energy minimization

- Several algorithms have been proposed to fit deformable contours.
- We'll look at two:
  - Greedy search
  - Dynamic programming (for 2d snakes)

Energy minimization: dynamic programming





- With this form of the energy function, we can minimize using dynamic programming, with the *Viterbi* algorithm.
- Iterate until optimal position* for each point is the center of the box, i.e., the snake is optimal in the local search space constrained by boxes.
- *optimal within resolution of considered window

Fig from Y. Boykov
[Amini, Weymouth, Jain, 1990]

Energy minimization: dynamic programming

- Possible because snake energy can be rewritten as a sum of pair-wise interaction potentials:

$$E_{total}(\mathbf{v}_1,\ldots,\mathbf{v}_n) = \sum_{i=1}^{n-1} E_i(\mathbf{v}_i,\mathbf{v}_{i+1})$$

- Or sum of triple-interaction potentials.

$$E_{total}(\mathbf{v}_1,\ldots,\mathbf{v}_n) = \sum_{i=1}^{n-1} E_i(\mathbf{v}_{i-1},\mathbf{v}_i,\mathbf{v}_{i+1})$$

Snake energy: pair-wise interactions

$$E_{total}(x_1,\ldots,x_n,y_1,\ldots,y_n) = -\sum_{i=1}^{n-1} |G_x(x_i,y_i)|^2 + |G_y(x_i,y_i)|^2 + \alpha\cdot\sum_{i=1}^{n-1}(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2$$

Re-writing the above with $v_i = (x_i, y_i)$ *:*

$$E_{total}(\mathbf{v}_1,\ldots,\mathbf{v}_n) = -\sum_{i=1}^{n-1} \|G(\mathbf{v}_i)\|^2 + \alpha\cdot\sum_{i=1}^{n-1} \|\mathbf{v}_{i+1}-\mathbf{v}_i\|^2$$

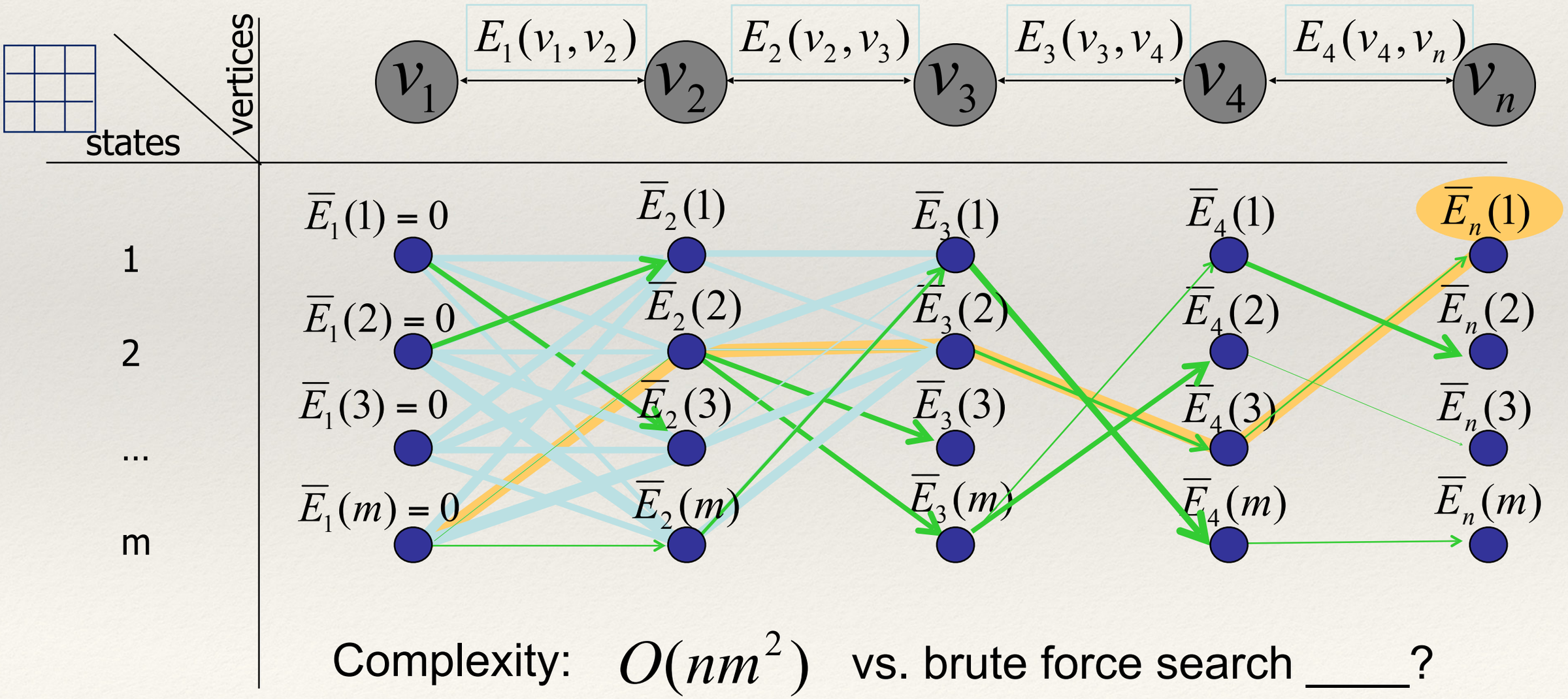
$$E_{total}(\mathbf{v}_1,\ldots,\mathbf{v}_n) = E_1(v_1,v_2)+E_2(v_2,v_3)+\ldots+E_{n-1}(v_{n-1},v_n)$$

where $E_i(\mathbf{v}_i,\mathbf{v}_{i+1}) = -\|G(\mathbf{v}_i)\|^2 + \alpha\|\mathbf{v}_{i+1}-\mathbf{v}_i\|^2$

Viterbi algorithm

Main idea: determine optimal position (state) of predecessor, for each possible position of self. Then backtrack from best state for last vertex.

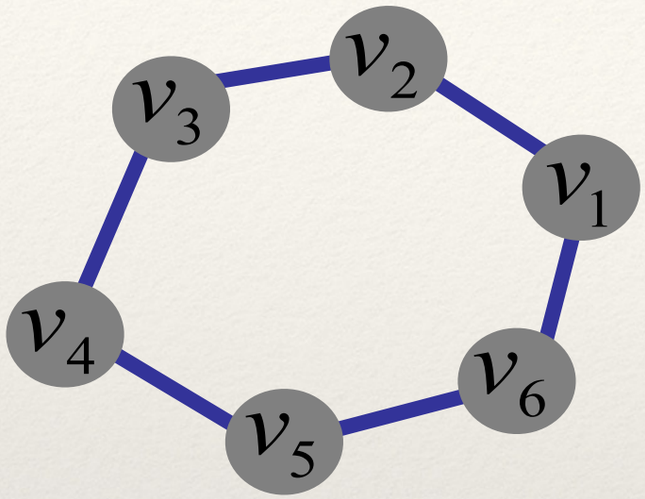
$$E_{total} = E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n)$$

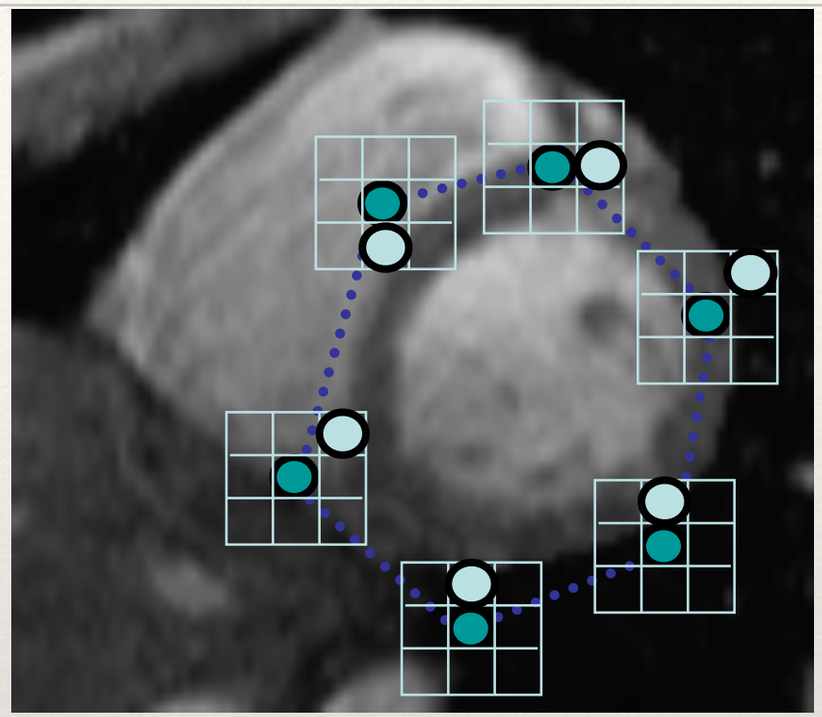


Complexity: $O(nm^2)$ vs. brute force search ____?

Example adapted from Y. Boykov

Energy minimization: dynamic programming





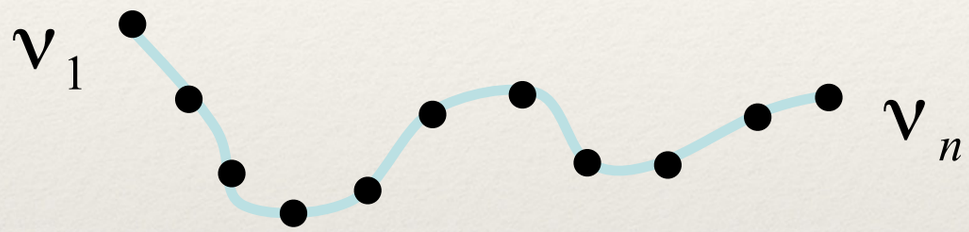
- With this form of the energy function, we can minimize using dynamic programming, with the *Viterbi* algorithm.
- Iterate until optimal position for each point is the center of the box, i.e., the snake is optimal in the local search space constrained by boxes.

Fig from Y. Boykov
[Amini, Weymouth, Jain, 1990]

Energy minimization: dynamic programming

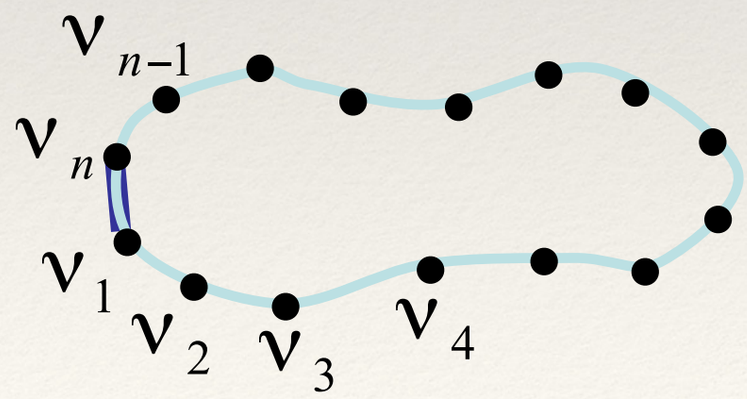
DP can be applied to optimize an open ended snake

$$E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n)$$



For a closed snake, a "loop" is introduced into the total energy.

$$E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n) + E_n(v_n, v_1)$$



Work around:

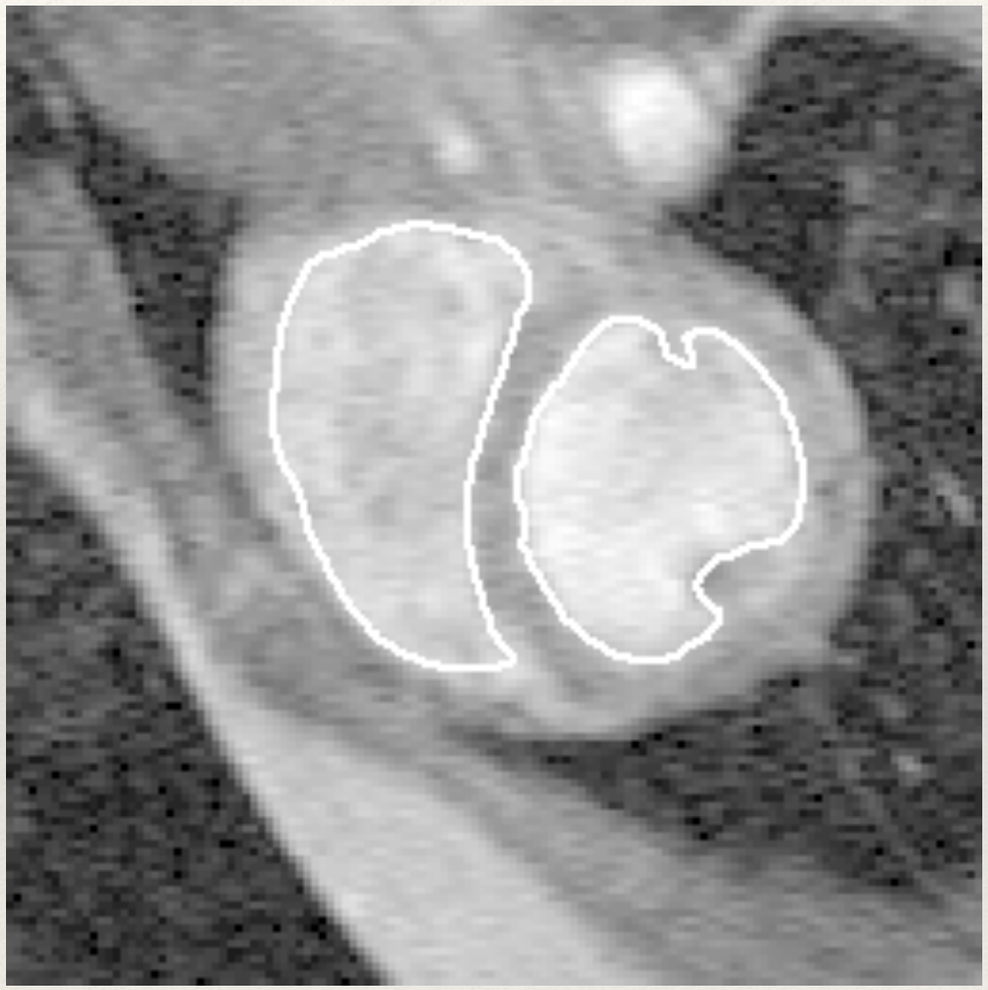
1) Fix v_1 and solve for rest .
2) Fix an intermediate node at its position found in (1), solve for rest.

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function
- Extensions:
  - Tracking
  - Interactive segmentation

Tracking via deformable contours

1. Use final contour/model extracted at frame *t* as an initial solution for frame *t+1*
2. Evolve initial contour to fit exact object boundary at frame *t+1*
3. Repeat, initializing with most recent frame.



Tracking Heart Ventricles
(multiple frames)

Tracking via deformable contours

Visual Dynamics Group, Dept. Engineering Science, University of Oxford.

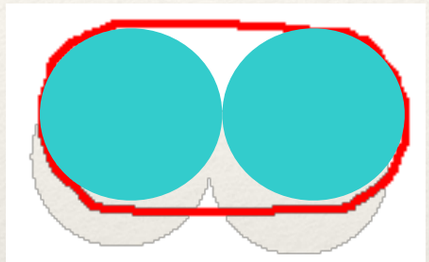
Applications: Traffic monitoring
Human-computer interaction
Animation
Surveillance
Computer assisted diagnosis in medical imaging

3D active contours

- Jörgen Ahlberg
- http://www.cvl.isy.liu.se/ScOut/Masters/Papers/Ex1708.pdf

Limitations

- May over-smooth the boundary



- Cannot follow topological changes of objects

Limitations

- External energy: snake does not really "see" object boundaries in the image unless it gets very close to it.

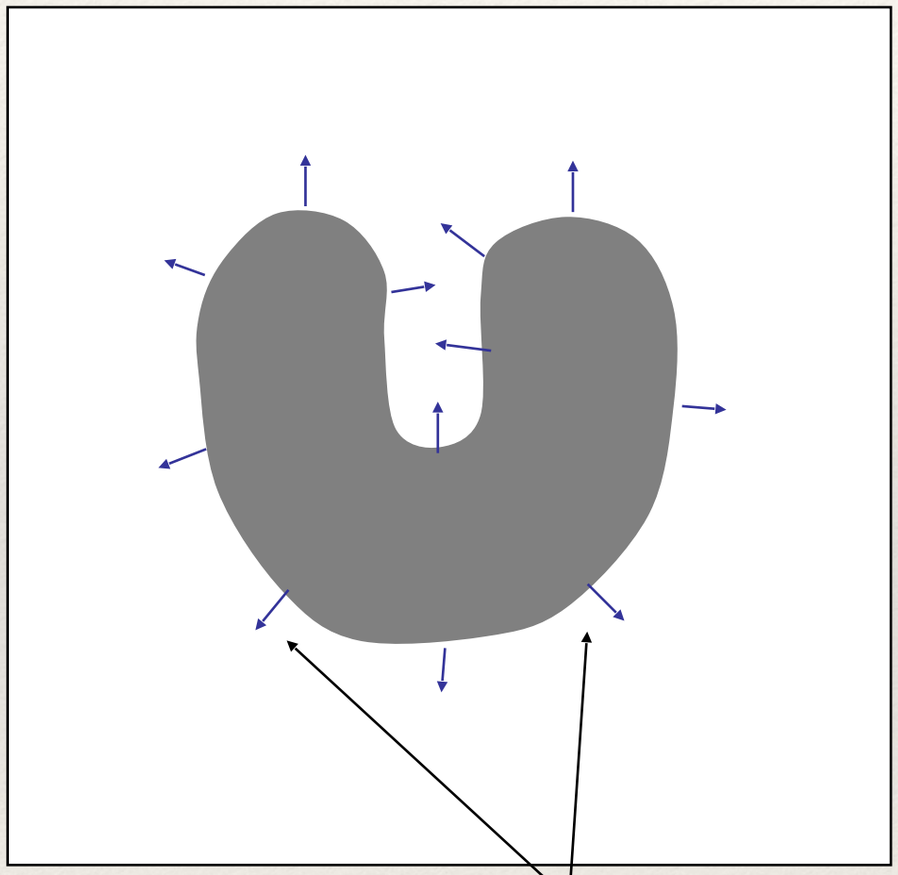
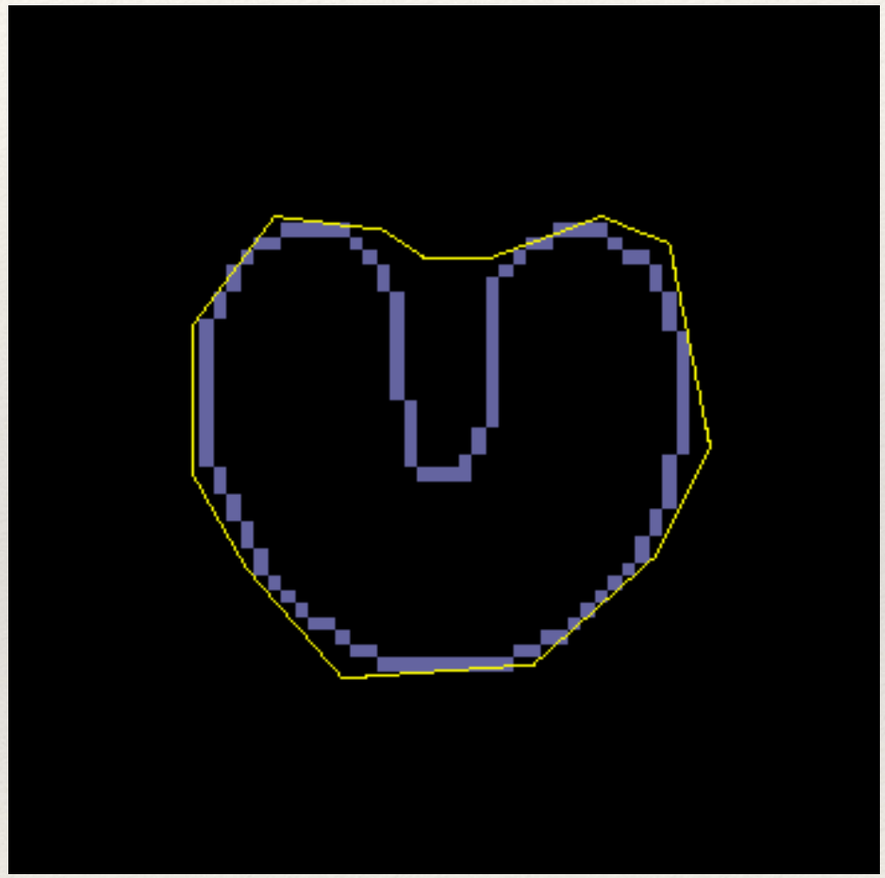
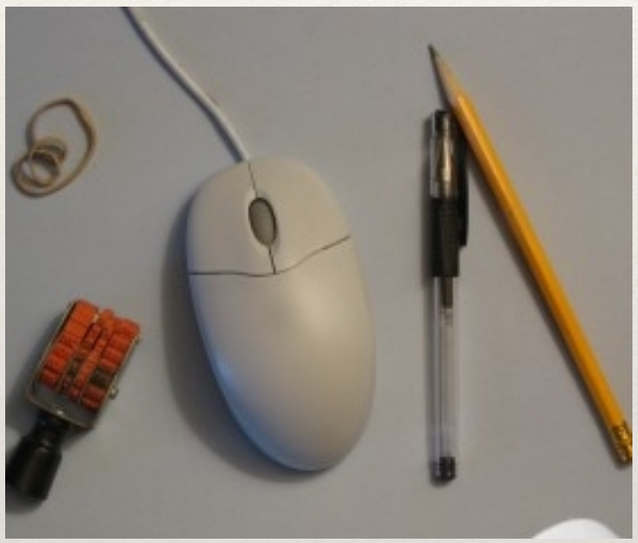


image gradients ∇I
are large only directly on the boundary



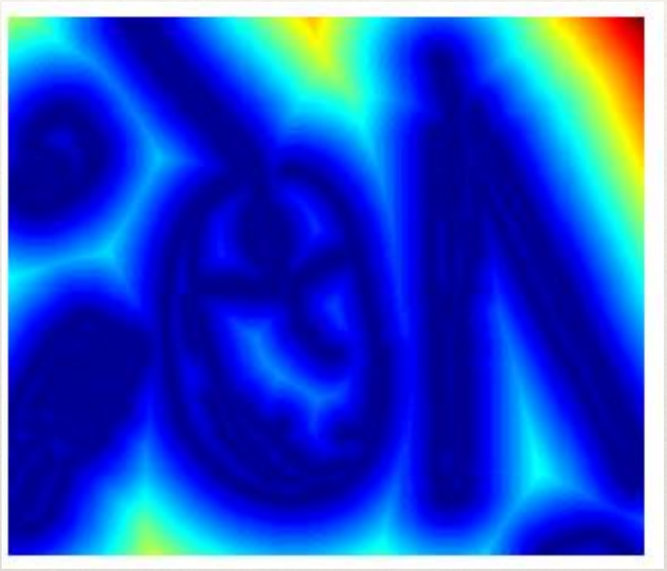
Distance transform

- External image can instead be taken from the **distance transform** of the edge image.

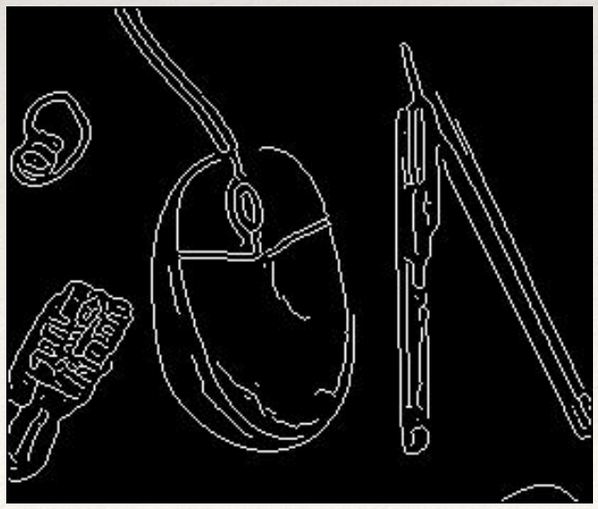


original

-gradient



distance transform



edges

Value at (x,y) tells how far that position is from the nearest edge point (or other binary mage structure)

```
>> help bwdist
```

Slide credit: Kristen Grauman

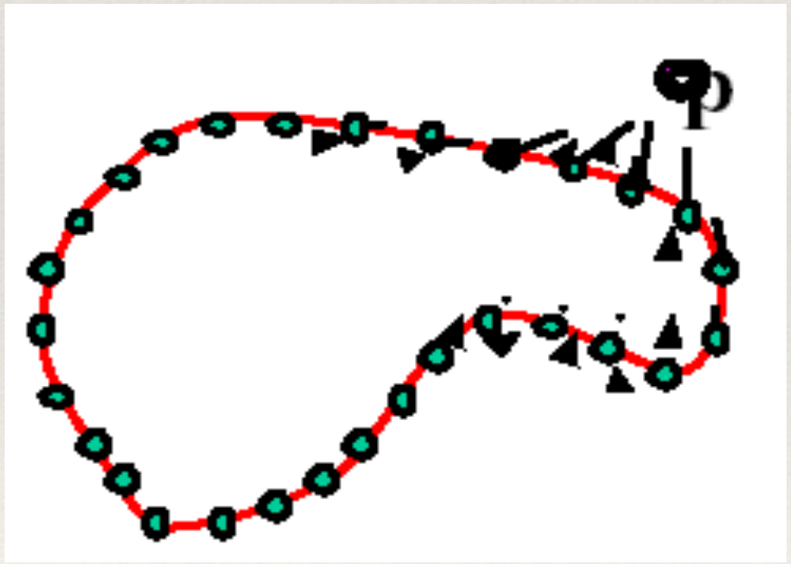
Interactive forces

How can we implement such an *interactive* force with deformable contours?

Slide credit: Kristen Grauman

Interactive forces

- An energy function can be altered online based on user input – use the cursor to push or pull the initial snake away from a point.
- Modify external energy term to include:



$$E_{push} = \sum_{i=0}^{n-1} \frac{r^2}{|\mathbf{v}_i - p|^2}$$

Nearby points get pushed hardest

Intelligent scissors

Another form of interactive segmentation:

Compute optimal paths **from every point to the seed** based on edge-related costs.

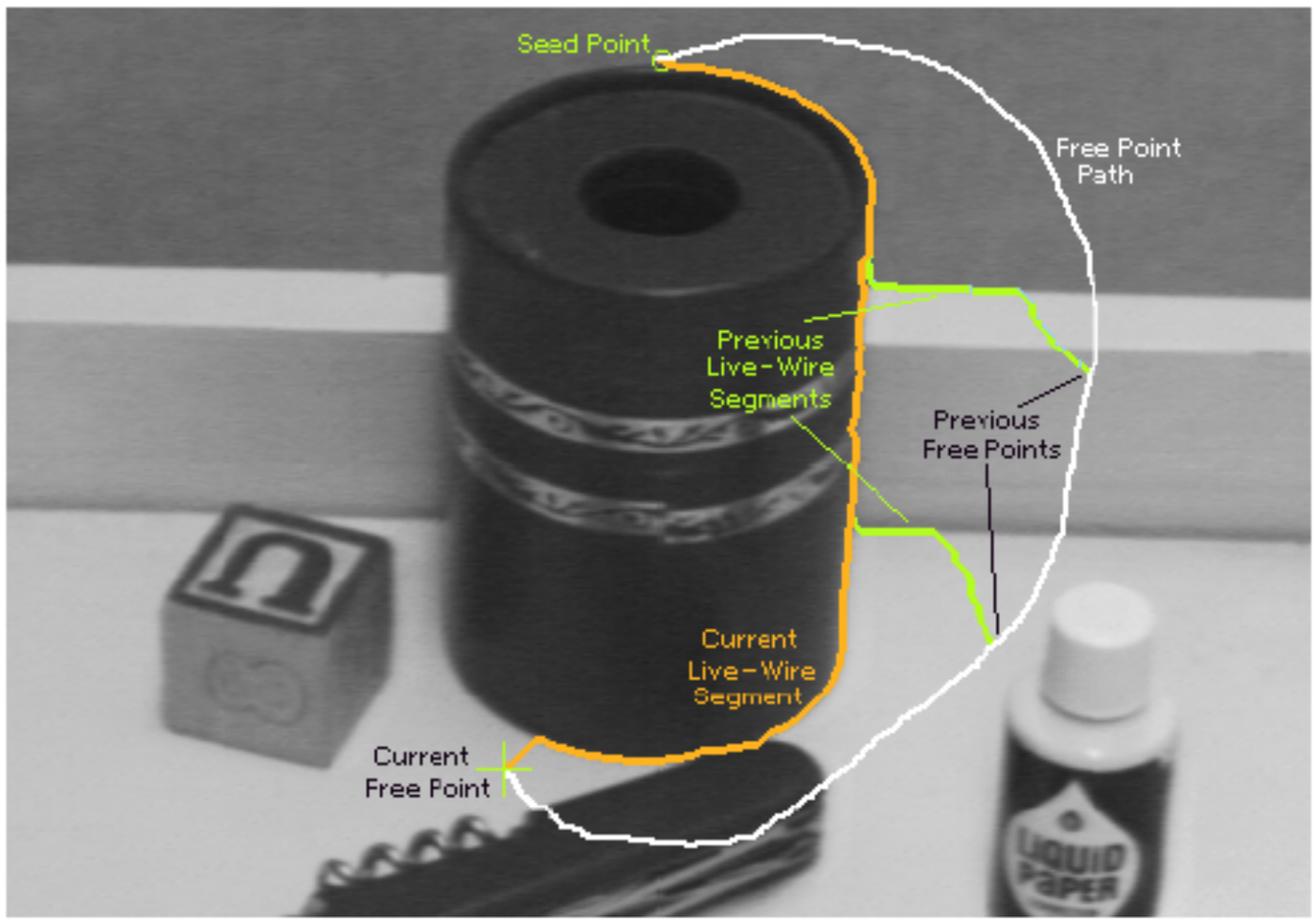
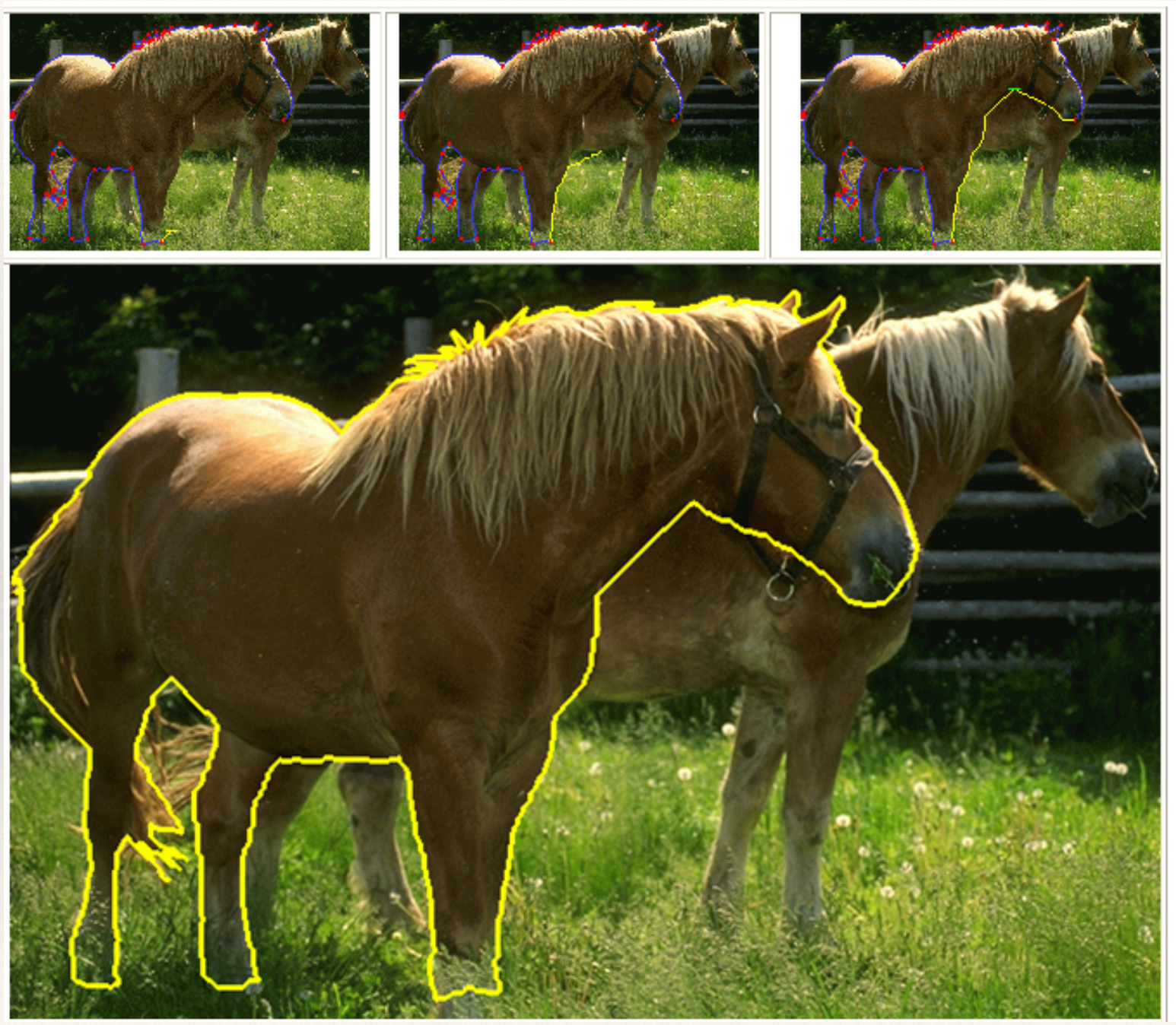


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

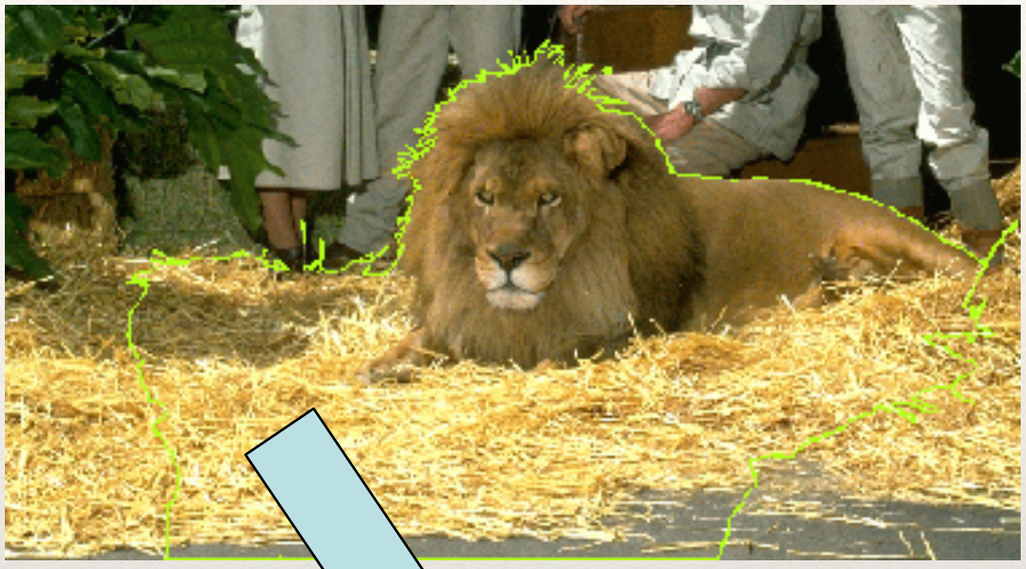
[Mortensen & Barrett, SIGGRAPH 1995, CVPR 1999]

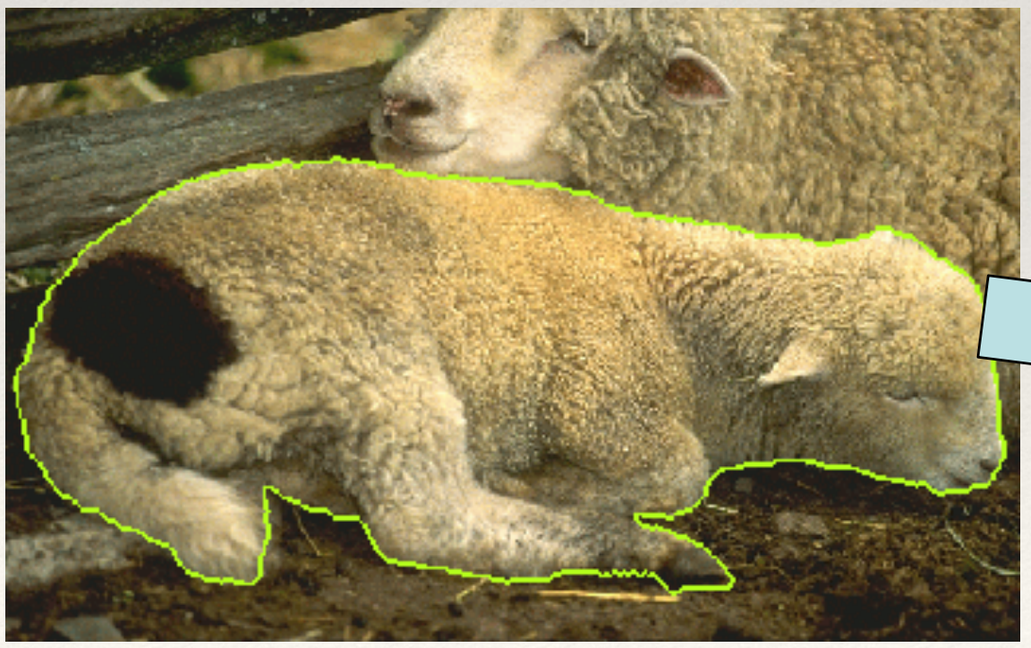
Intelligent scissors



- http://rivit.cs.byu.edu/Eric/Eric.html

Intelligent scissors





Deformable contours: pros and cons

Pros:

- Useful to track and fit non-rigid shapes
- Contour remains connected
- Possible to fill in “subjective” contours
- Flexibility in how energy function is defined, weighted.

Cons:

- Must have decent initialization near true boundary, may get stuck in local minimum
- Parameters of energy function must be set well based on prior information

Slide credit: Kristen Grauman

Summary

- Deformable shapes and active contours are useful for
  - Segmentation: fit or “snap” to boundary in image
  - Tracking: previous frame’s estimate serves to initialize the next
- Fitting active contours:
  - Define terms to encourage certain shapes, smoothness, low curvature, push/pulls, …
  - Use weights to control relative influence of each component cost
  - Can optimize 2d snakes with Viterbi algorithm.
- Image structure (esp. gradients) can act as attraction force for *interactive* segmentation methods.