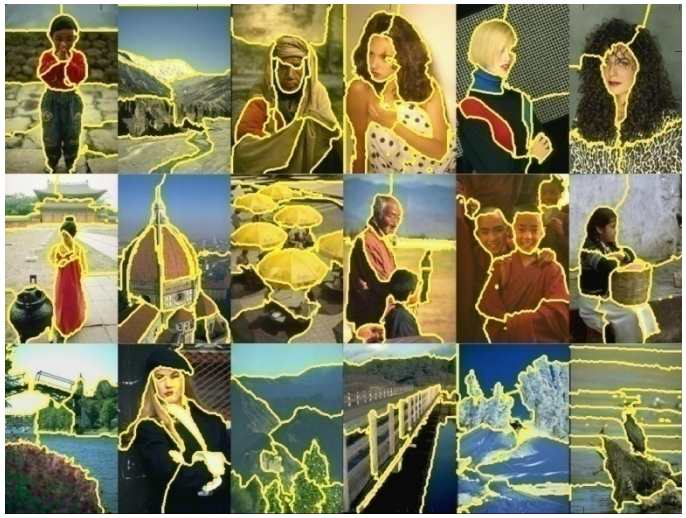
Segmentation, snakes, PDE, clustering

Grouping in vision

- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image or video parts

Examples of grouping in vision



[Figure by J. Shi]

Determine image regions

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

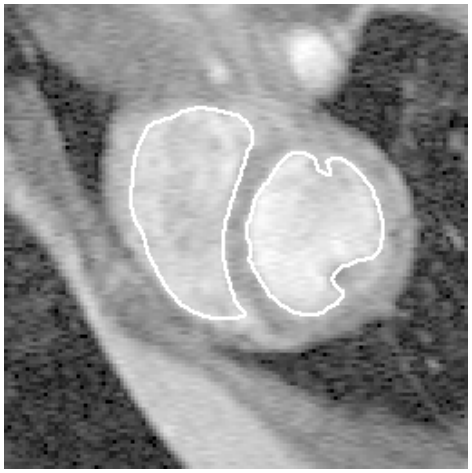
Group video frames into shots

[Figure by Wang & Suter]

Figure-ground

Image Segmentation

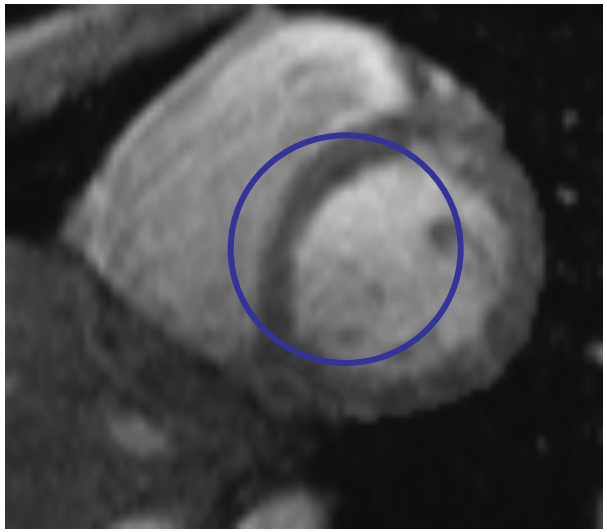
Deformable contours: sneak, level set



Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

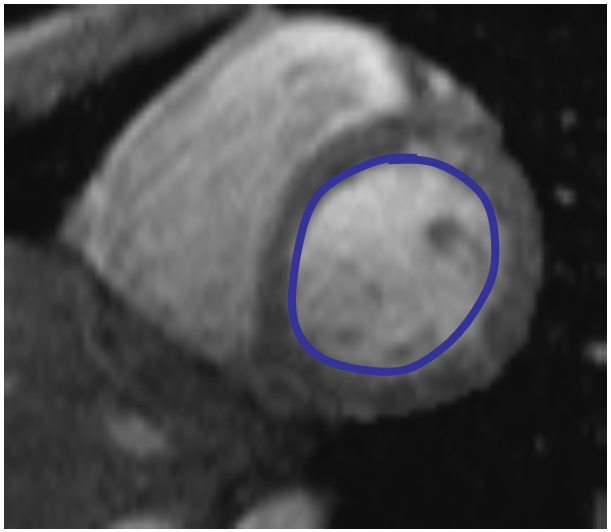


Deformable contours

a.k.a. active contours, snakes

Given: initial contour (model) near desired object

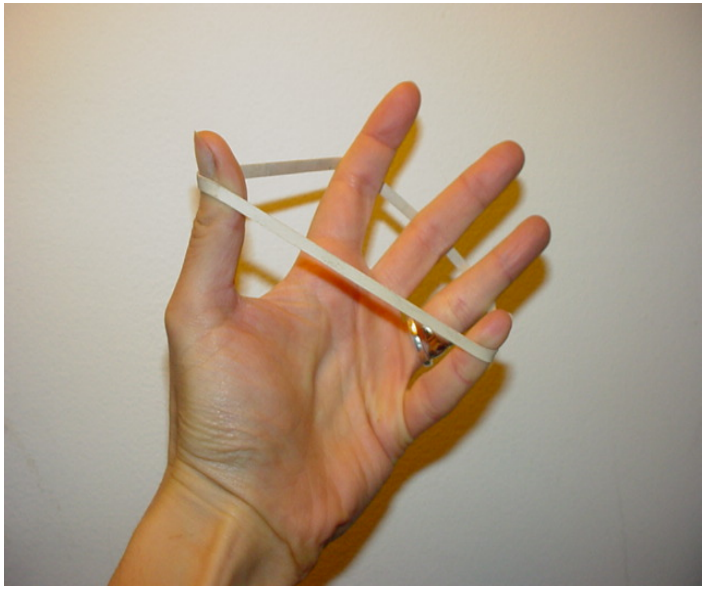
Goal: evolve the contour to fit exact object boundary

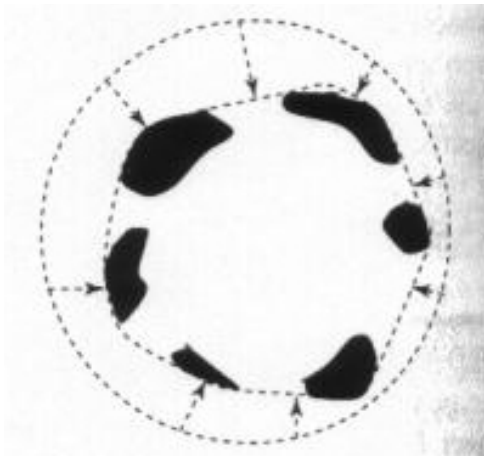


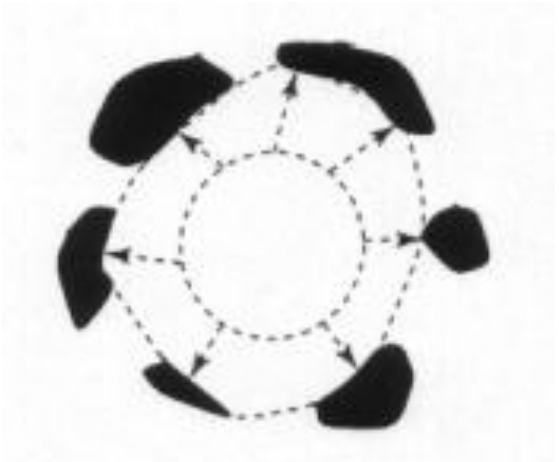
Main idea: elastic band is iteratively adjusted so as to

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

Deformable contours: intuition



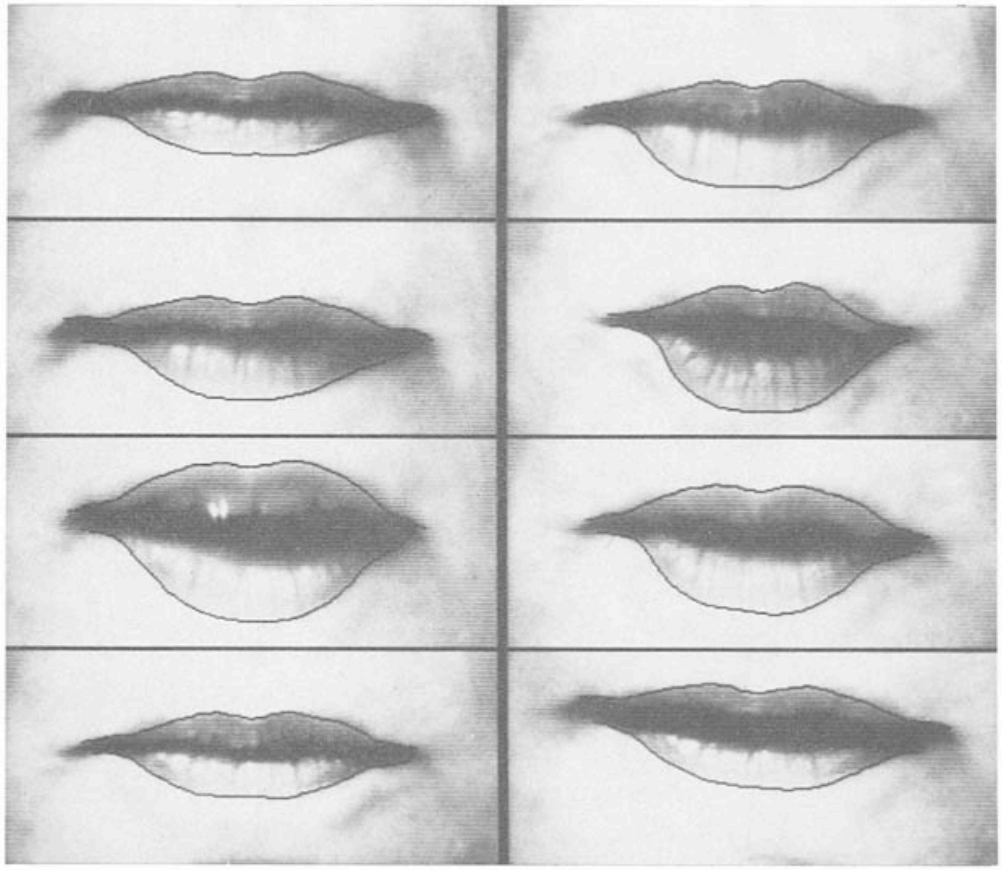




Why do we want to fit deformable shapes?

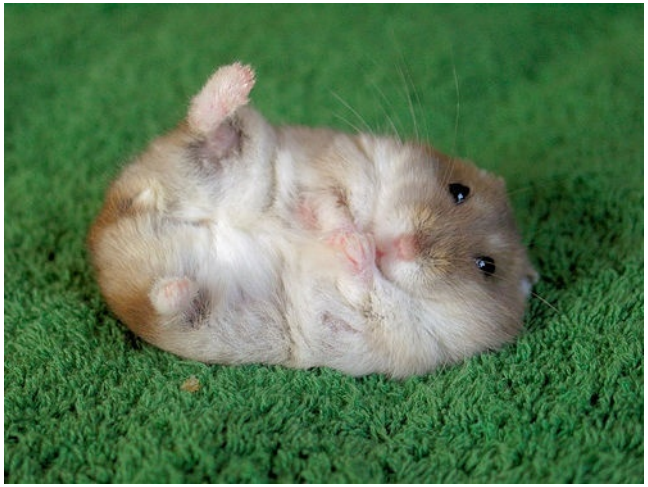
- Some objects have similar basic form but some variety in the contour shape.

Why do we want to fit deformable shapes?



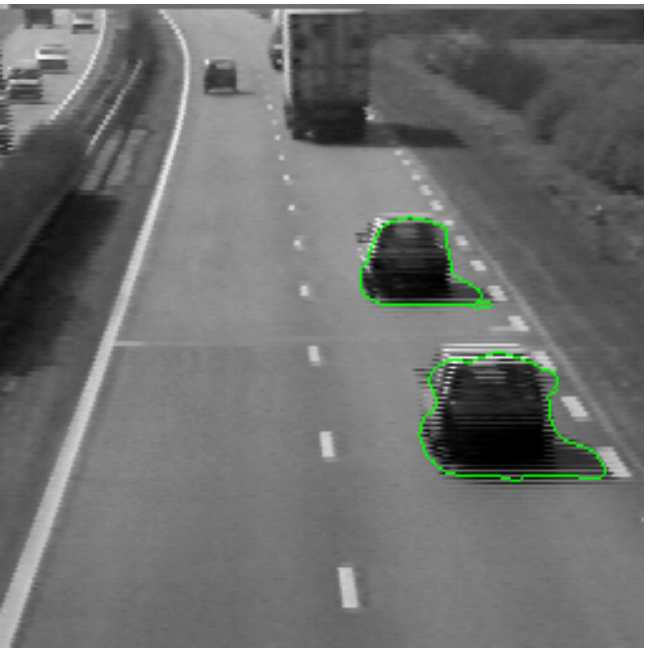
- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands…

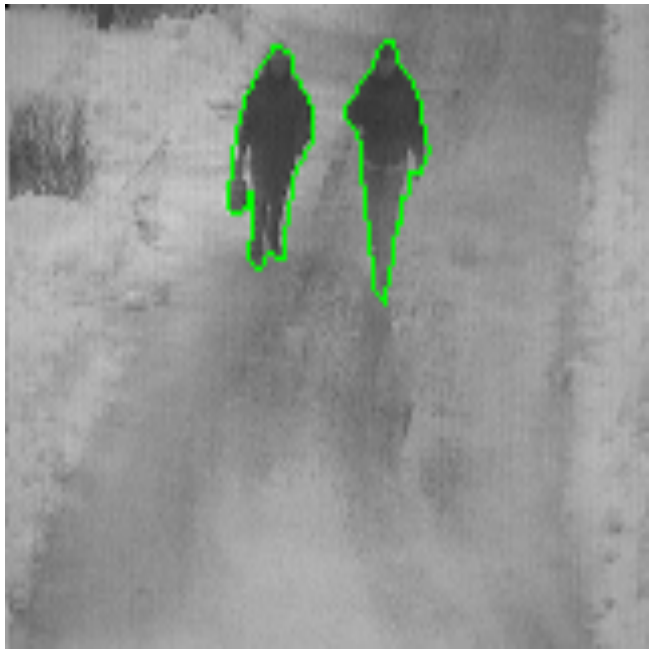
Why do we want to fit deformable shapes?

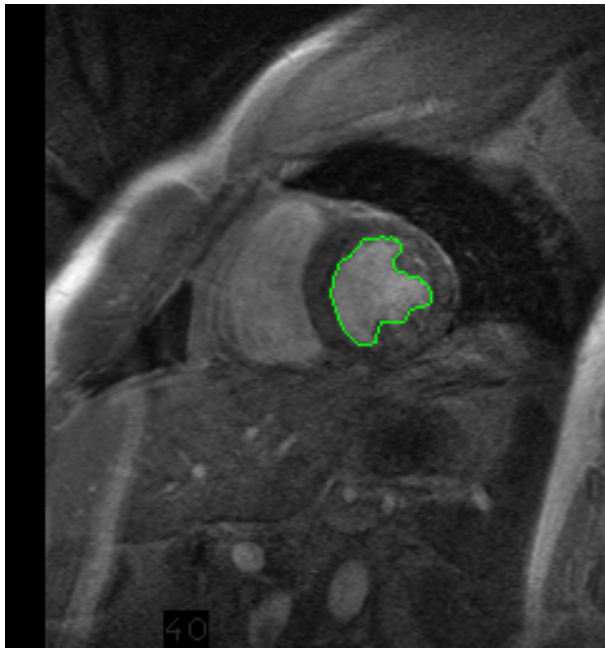


- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands…

Why do we want to fit deformable shapes?







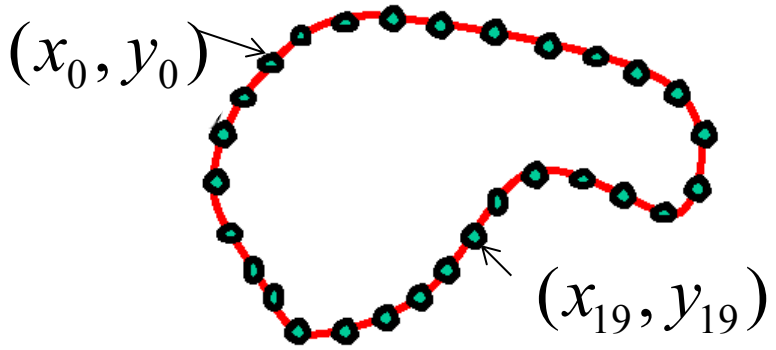
- Non-rigid, deformable objects can change their shape over time.

Aspects we need to consider

- Representation of the contours
- Defining the energy functions
  - External
  - Internal
- Minimizing the energy function

Representation

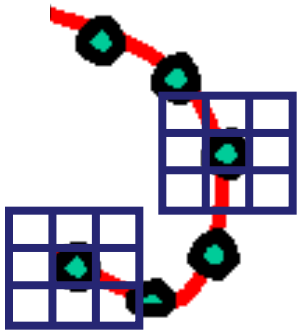
- We'll consider a discrete representation of the contour, consisting of a list of 2d point positions ("vertices").



$$v_i = (x_i, y_i),$$

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

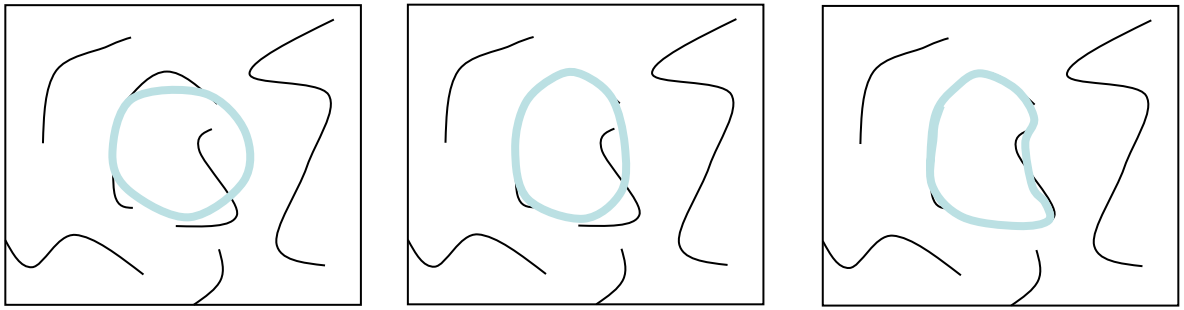
- At each iteration, we'll have the option to move each vertex to another nearby location ("state").



Fitting deformable contours

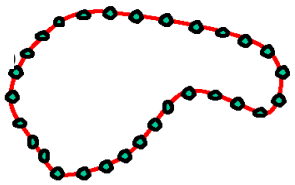
How should we adjust the current contour to form the new contour at each iteration?

- 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:



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

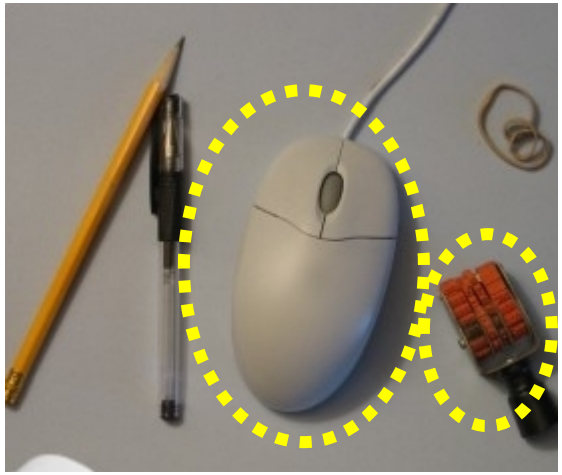
Internal energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

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

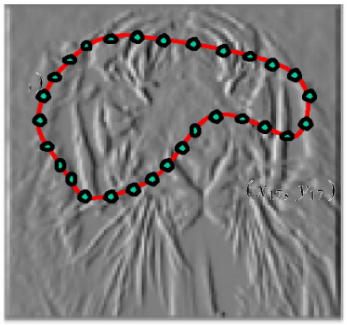
- Measure how well the curve matches the image data
- “Attract” the curve toward different image features
  - Edges, lines, texture gradient, etc.

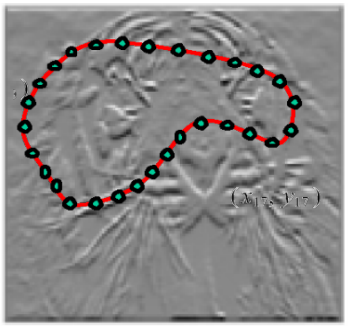


Think of external energy from image as gravitational pull towards areas of high contrast

External image energy

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





- External energy at a point on the curve is:

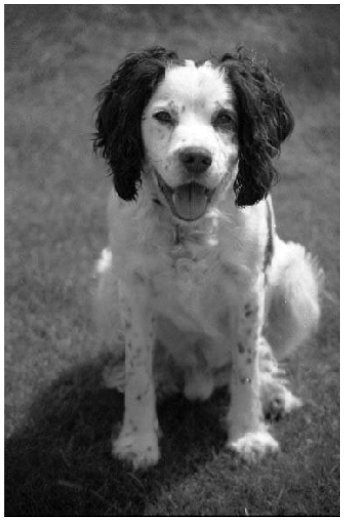
$$E_{external}(v) = -(|G_x(v)|^2 + |G_y(v)|^2)$$

- 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

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).



Internal energy

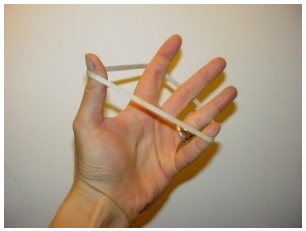
For a *continuous* curve, a common internal energy term is the “bending energy”.

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

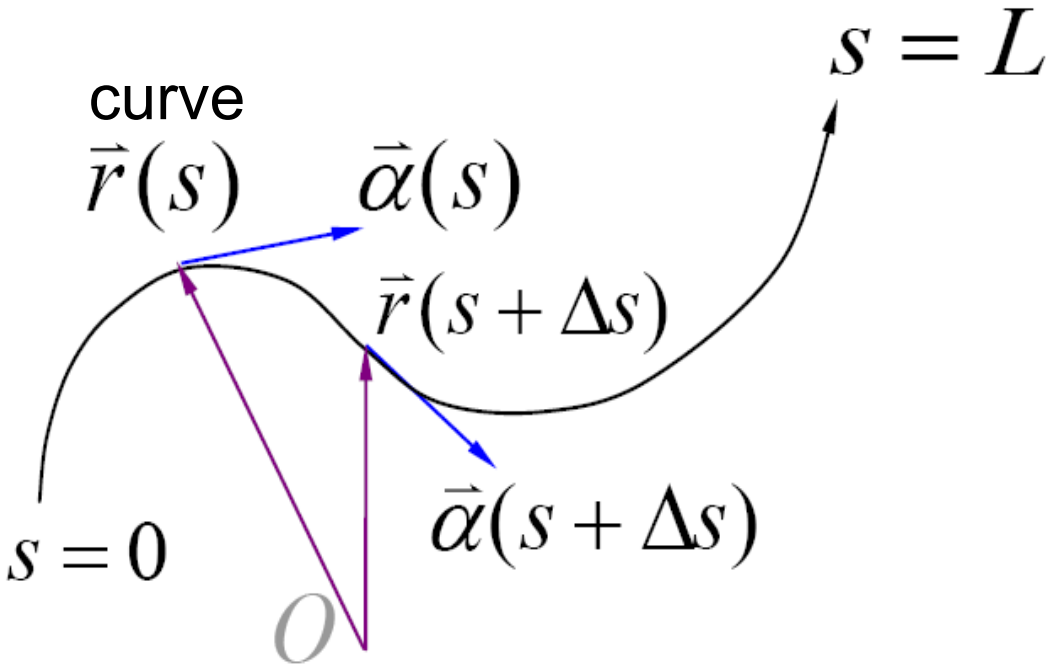
$$E_{internal}(v(s)) = \alpha \left| \frac{dv}{ds} \right|^2 + \beta \left| \frac{d^2 v}{d^2 s} \right|^2$$

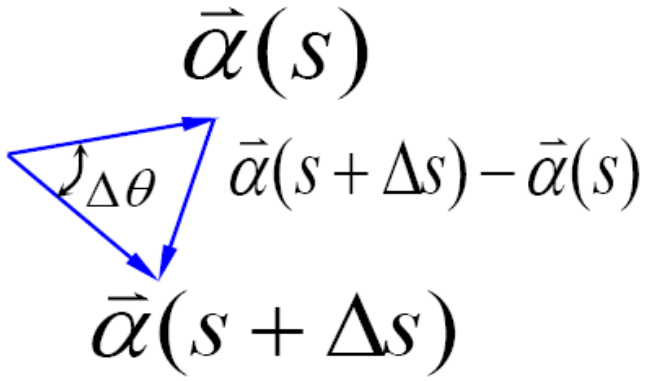
Tension, Elasticity

Stiffness, Curvature



Curvature

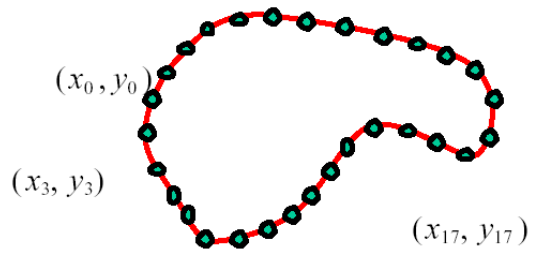




$$\left|\frac{d\vec{\alpha}}{ds}\right| = \lim_{\Delta s\to 0}\left|\frac{\Delta\vec{\alpha}}{\Delta s}\right| = \lim_{\Delta s\to 0}\left|\frac{1}{\Delta s}\left(\vec{\alpha}(s+\Delta s)-\vec{\alpha}(s)\right)\right|$$

$$= \lim_{\Delta s\to 0}\frac{2\left|\sin\left(\frac{\Delta\theta}{2}\right)\right|}{|\,\Delta s\,|} = \lim_{\Delta s\to 0}\left|\frac{\sin\left(\frac{\Delta\theta}{2}\right)}{\frac{\Delta\theta}{2}}\frac{\Delta\theta}{\Delta s}\right| = \lim_{\Delta s\to 0}\left|\frac{\Delta\theta}{\Delta s}\right|$$

Internal energy



- For our discrete representation,

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

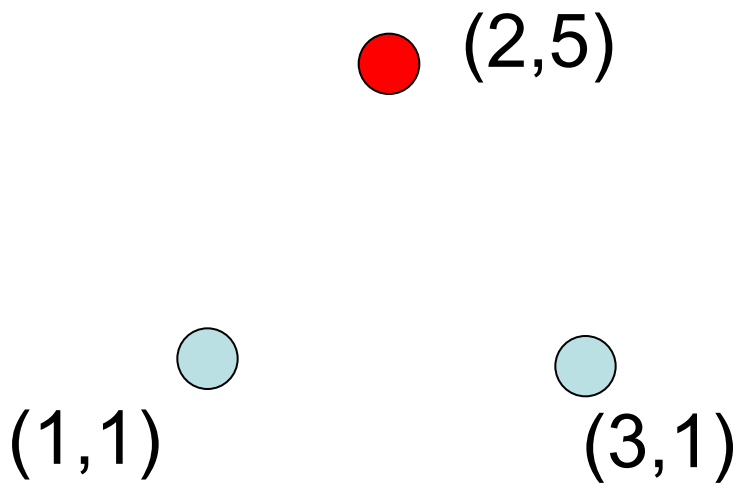
$$\frac{dv}{ds} \approx v_{i+1} - v_i \qquad \frac{d^2 v}{ds^2} \approx (v_{i+1} - v_i) - (v_i - v_{i-1}) = v_{i+1} - 2v_i + v_{i-1}$$

- Internal energy for the whole curve:

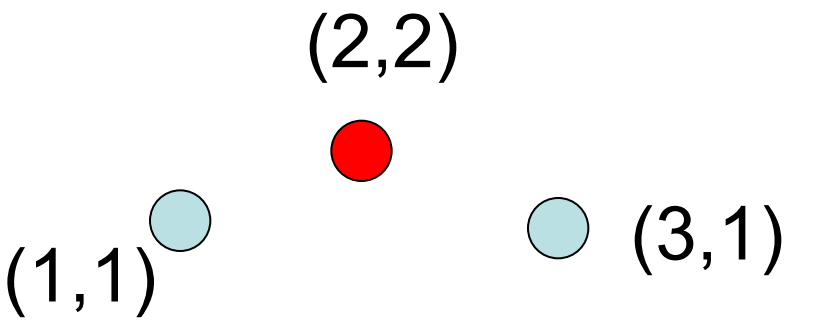
$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left\| v_{i+1} - v_i \right\|^2 + \beta \left\| v_{i+1} - 2v_i + v_{i-1} \right\|^2$$

Example: compare curvature

$$E_{curvature}(v_i) = \left\| v_{i+1} - 2v_i + v_{i-1} \right\|^2$$
$$= (x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2$$



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



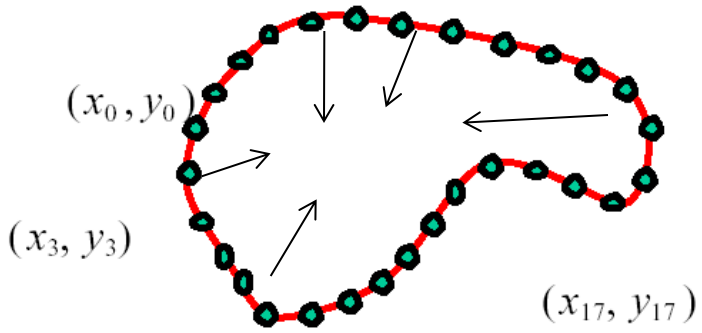
$$(3 - 2(2) + 1)^2 + (1 - 2(2) + 1)^2$$
$$= (-2)^2 = 4$$

Penalizing elasticity

- Current elastic energy definition uses a discrete estimate of the derivative:

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

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



What is the possible problem with this definition?

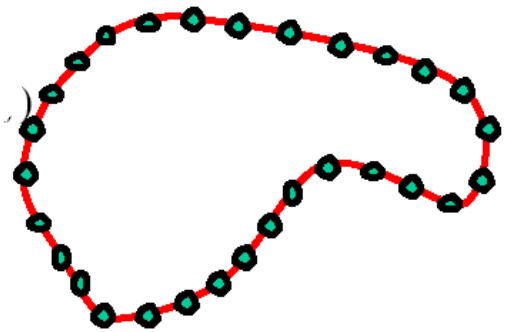
Penalizing elasticity

- Current elastic energy definition uses a discrete estimate of the derivative:

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

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.

Total energy: function of the weights

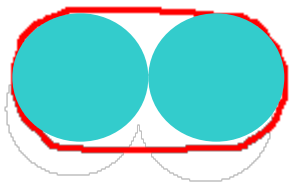
$$E_{total} = E_{internal} + \gamma E_{external}$$

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

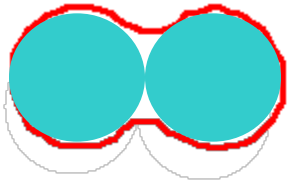
$$E_{internal} = \sum_{i=0}^{n-1} \alpha \left(\bar{d} - \|v_{i+1} - v_i\|\right)^2 + \beta \|v_{i+1} - 2v_i + v_{i-1}\|^2$$

Total energy: function of the weights

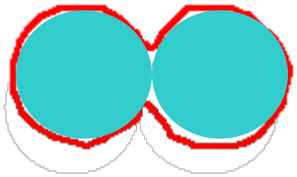
- e.g., α weight controls the penalty for internal elasticity



large α



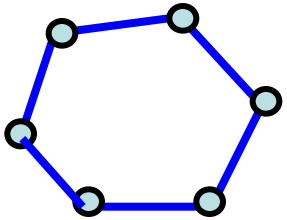
medium α



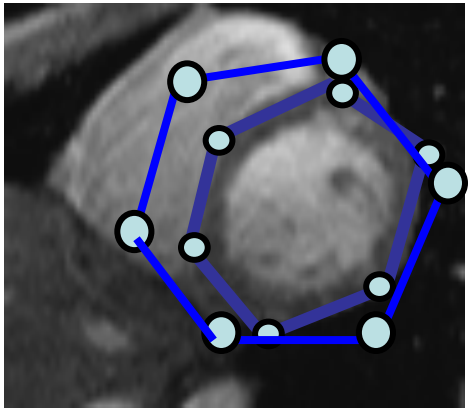
small α

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

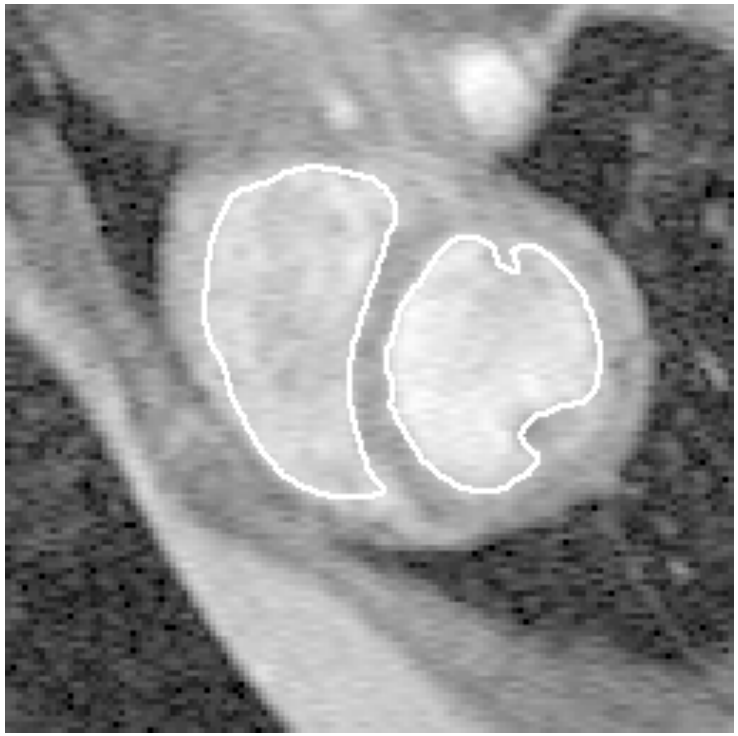


Energy minimization

- Several algorithms have been proposed to fit deformable contours.
  - Greedy search
  - Dynamic programming (for 2d snakes)
  - PDE

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)

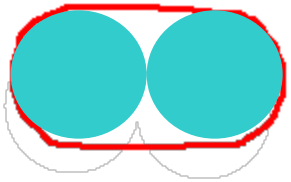
Tracking via deformable contours

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

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

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.

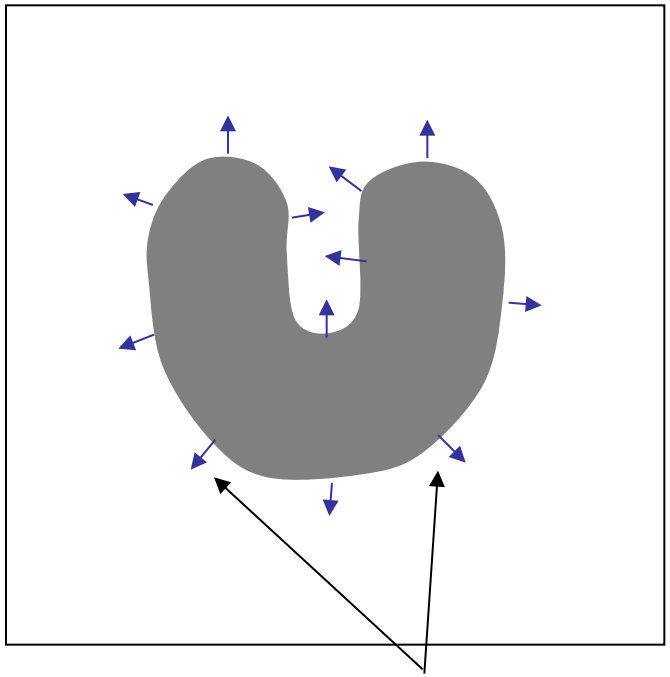
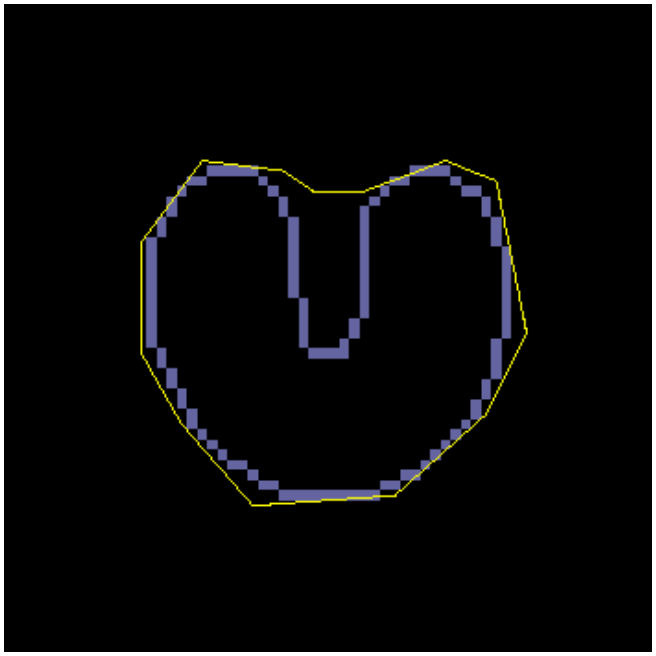


image gradients ∇I
are large only directly on the boundary



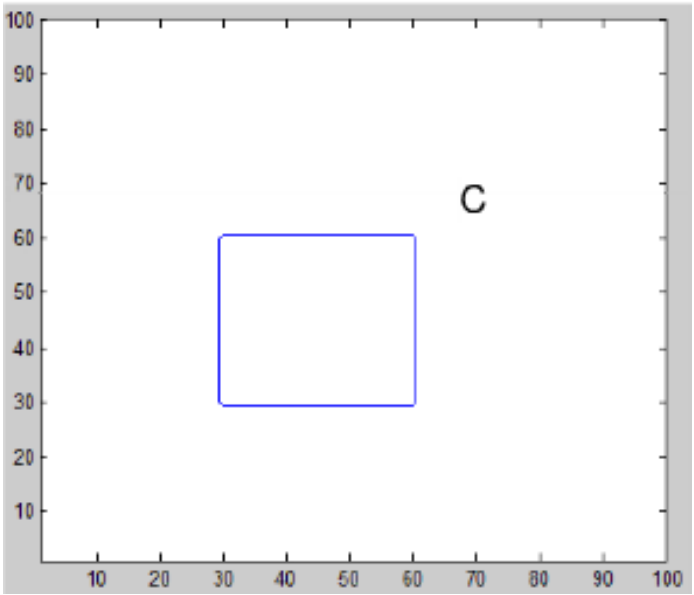
Level set

$$C = \{(x, y), f(x, y) = c\}$$

- Model the physical process and interface motion using partial derivative equations(PDE).
- Level Set Methods are used for the implementation of curve/interface evolution under various forces.

Curve representation

- A closed curve cannot be represented as a 1-D function y=f(x)
- A parametric representation can be used: C={x(p),y(p)} p is in [0,1]



Curve representation

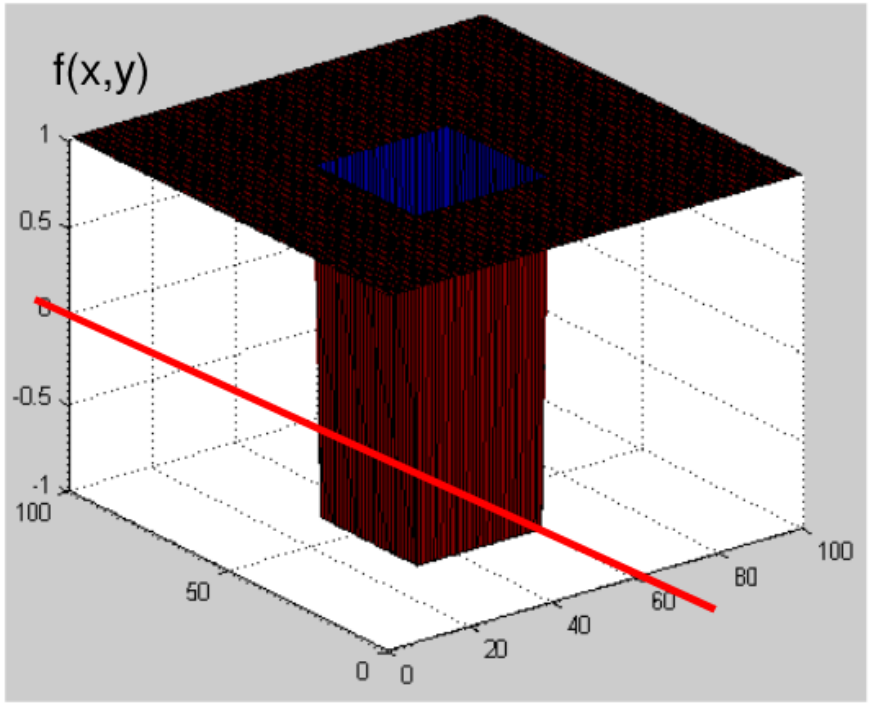
- Explicit (parametric) representation

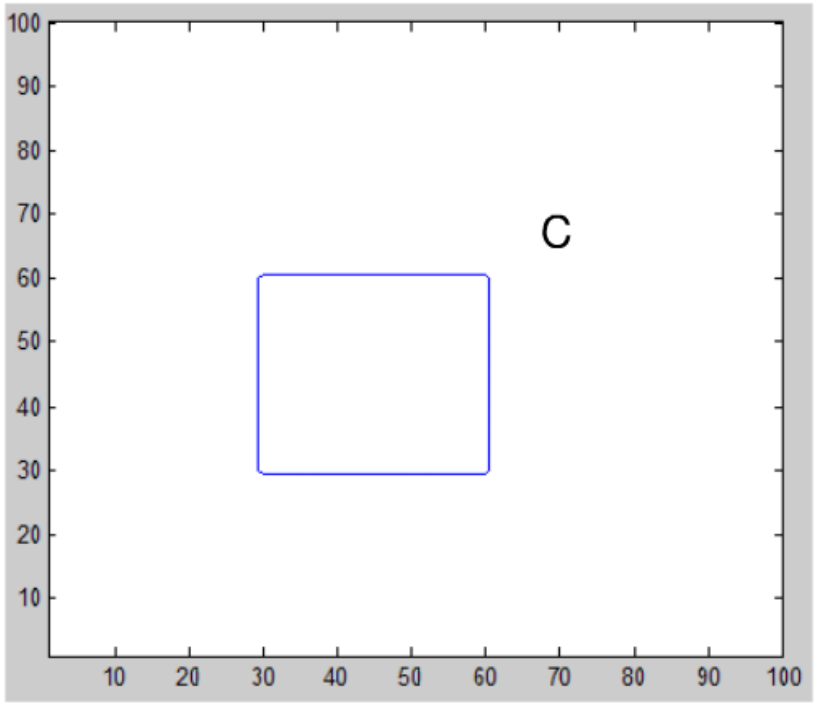
- Implicit representation (Level set methods)

Embed the curve C into a 2-D function
$z=f(x,y)$

Implicit methods

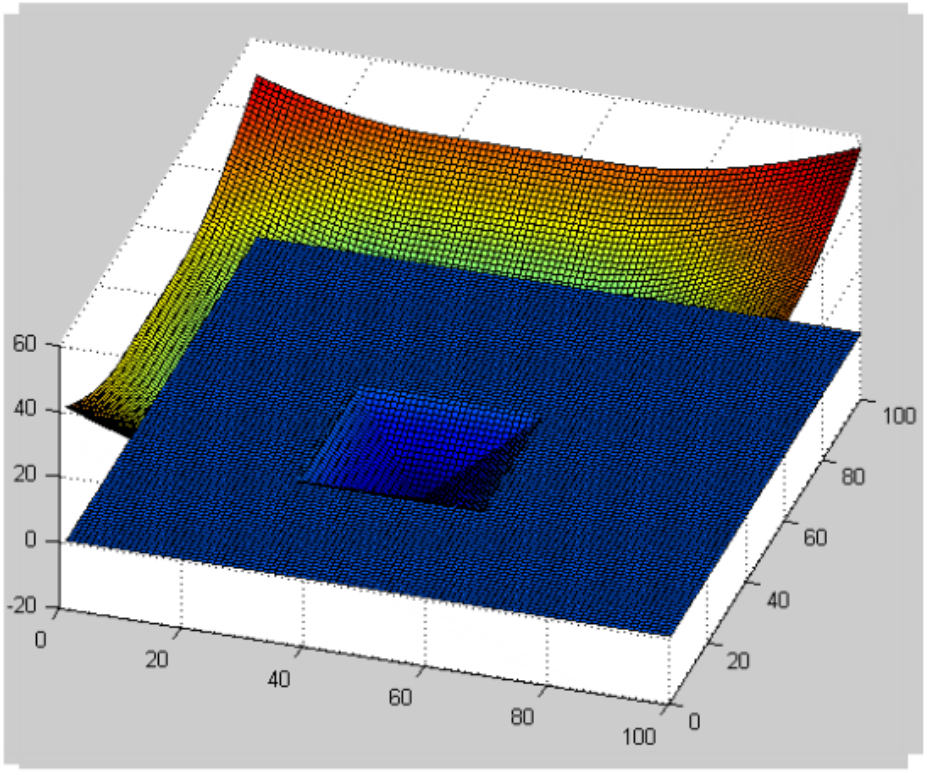
- Curve is embedded as the 0 level set
- C={(x,y)| f(x,y)=0}

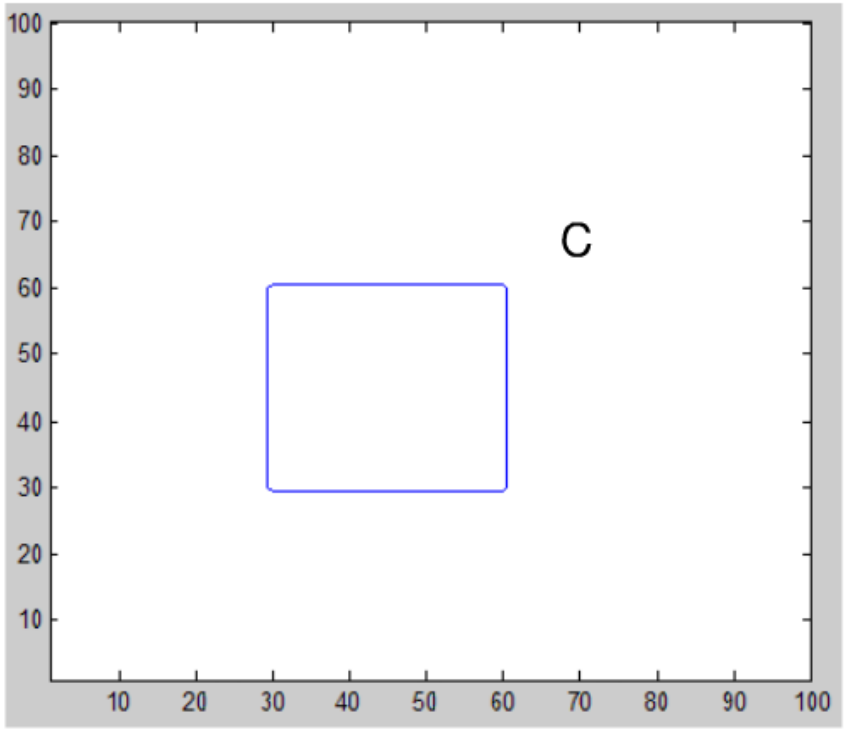




Implicit Methods

- A better embedding function is smoother





Level Set Methods

- Main idea:

Evolve the embedding function f(x,y)

Keep track of its zero level set

Evolution Forces

- Types of forces

A force in the normal direction to the curve.

An external vector field.

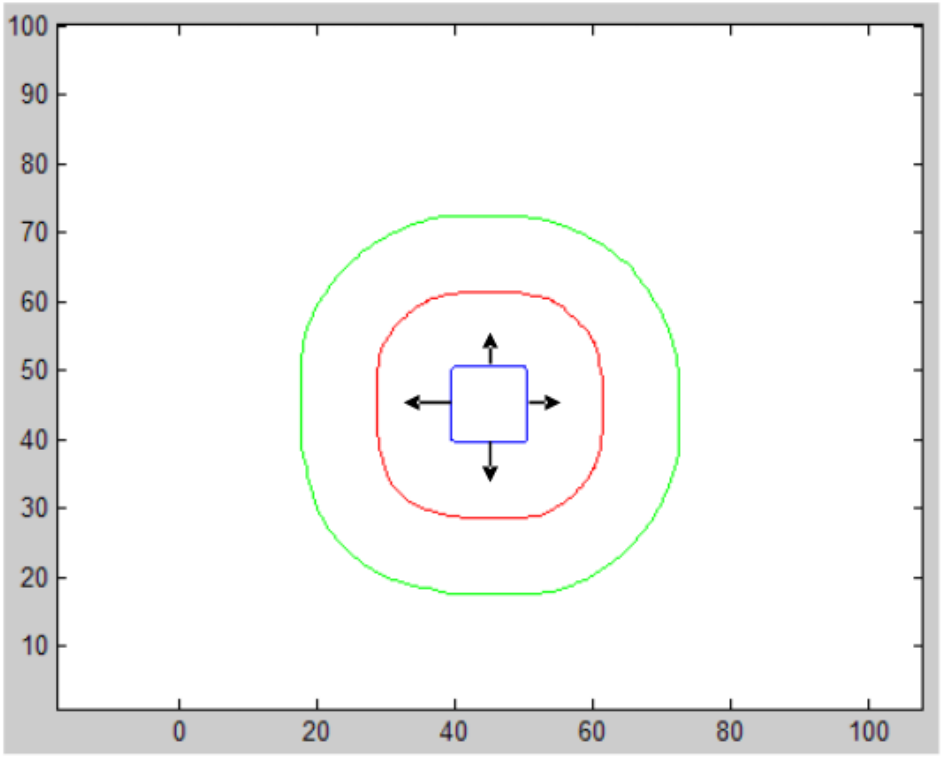
A force based on the curvature of the curve.

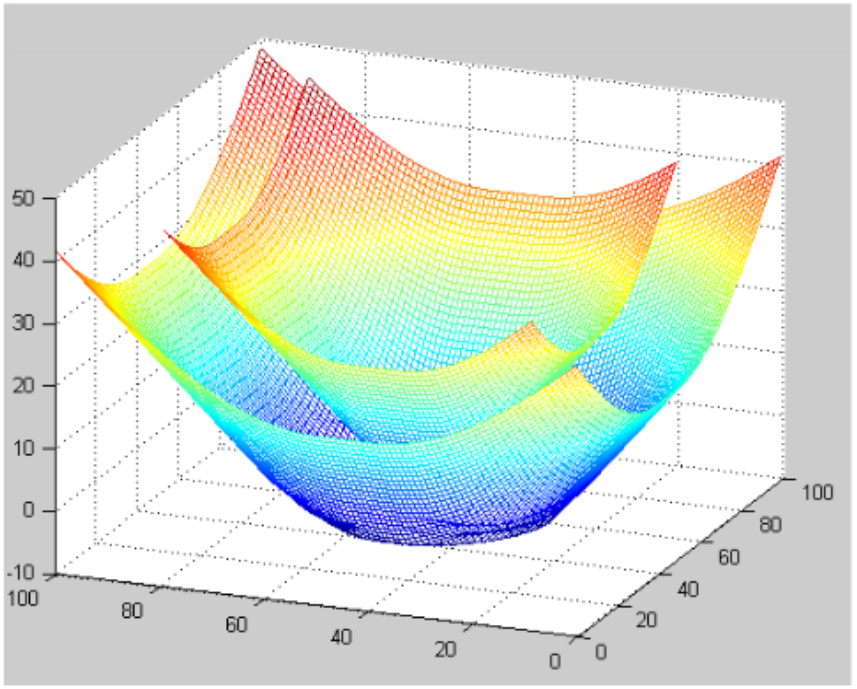
- Partial Differential Equation:

$$\frac{\partial f}{\partial t} + \underbrace{\vec{S} \cdot \nabla f}_{\text{Vector Field Based}} + \underbrace{V_N \left|\nabla f\right|}_{\text{In Normal Direction}} = \underbrace{b\kappa \left|\nabla f\right|}_{\text{Curvature Based}}$$

Force in Normal Direction

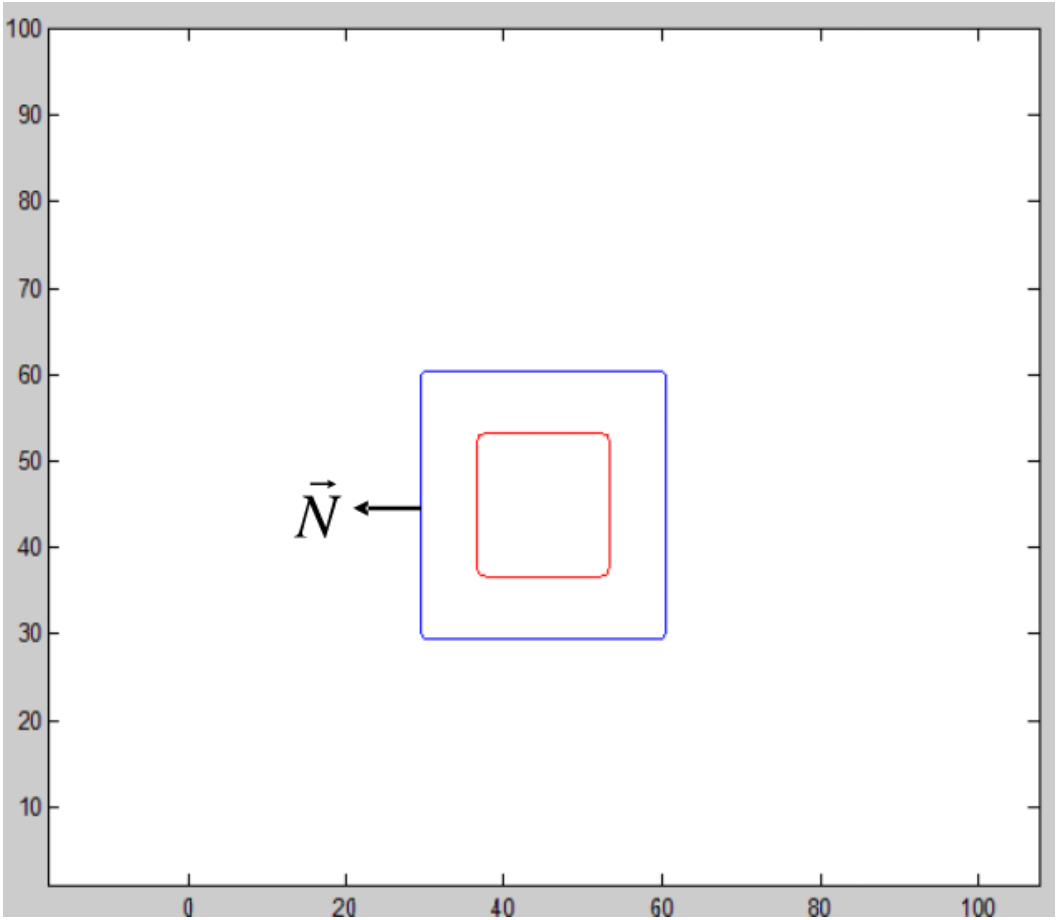
$$\frac{\partial f}{\partial t} + V_N \left| \nabla f \right| = 0 \qquad V_n = 1$$





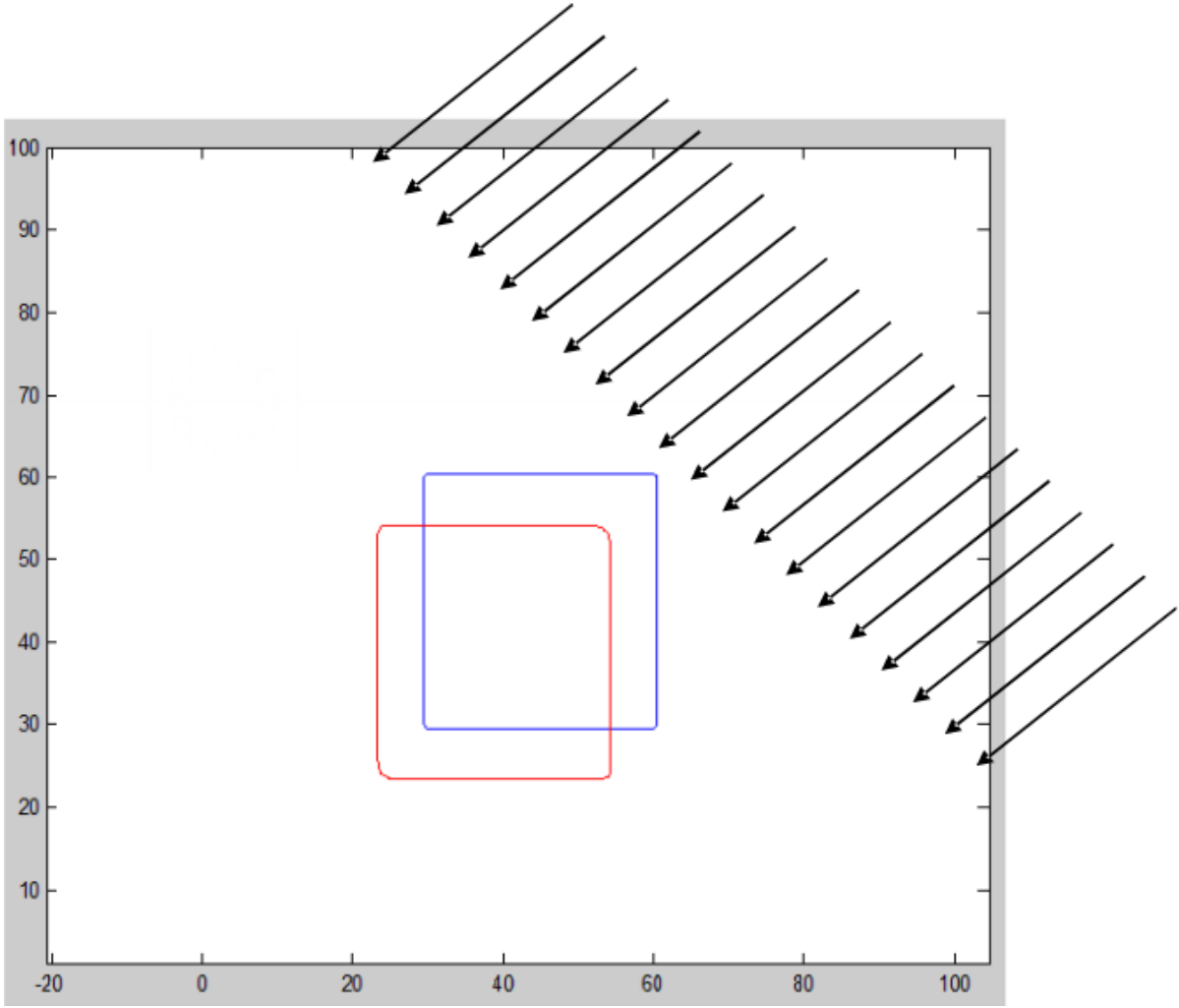
Shrink in Normal Direction

$V_n = -1$



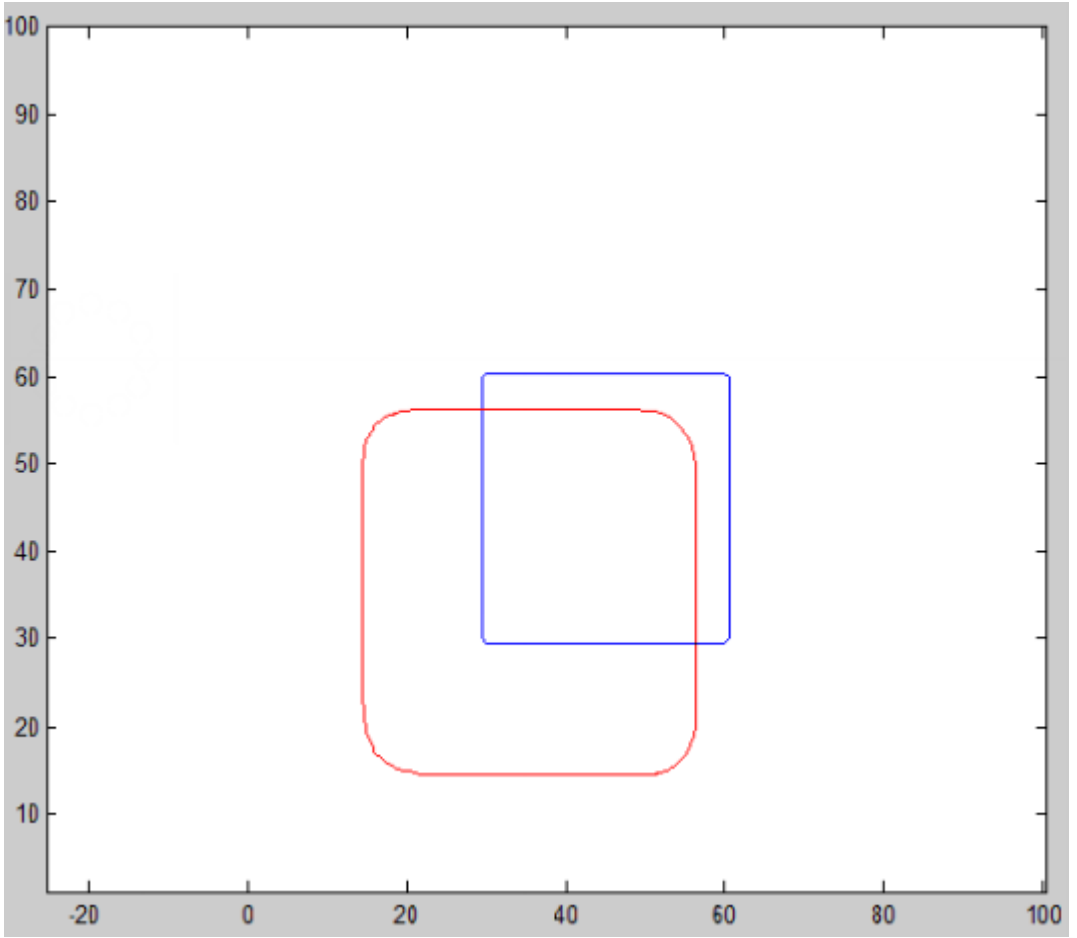
External Vector Field

$$\frac{\partial f}{\partial t} + \vec{S} \cdot \nabla f = 0$$

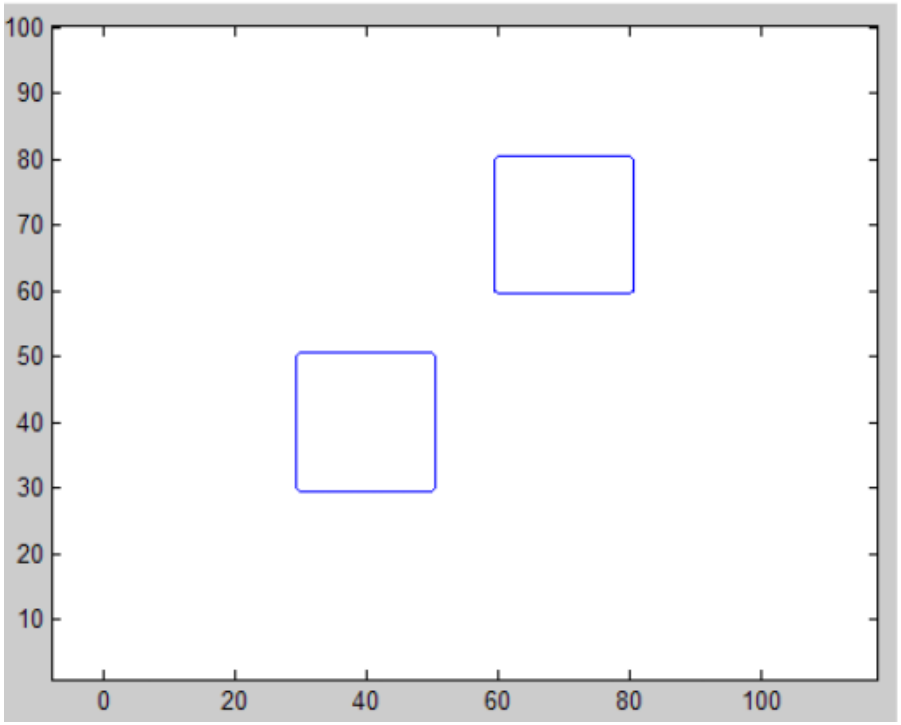


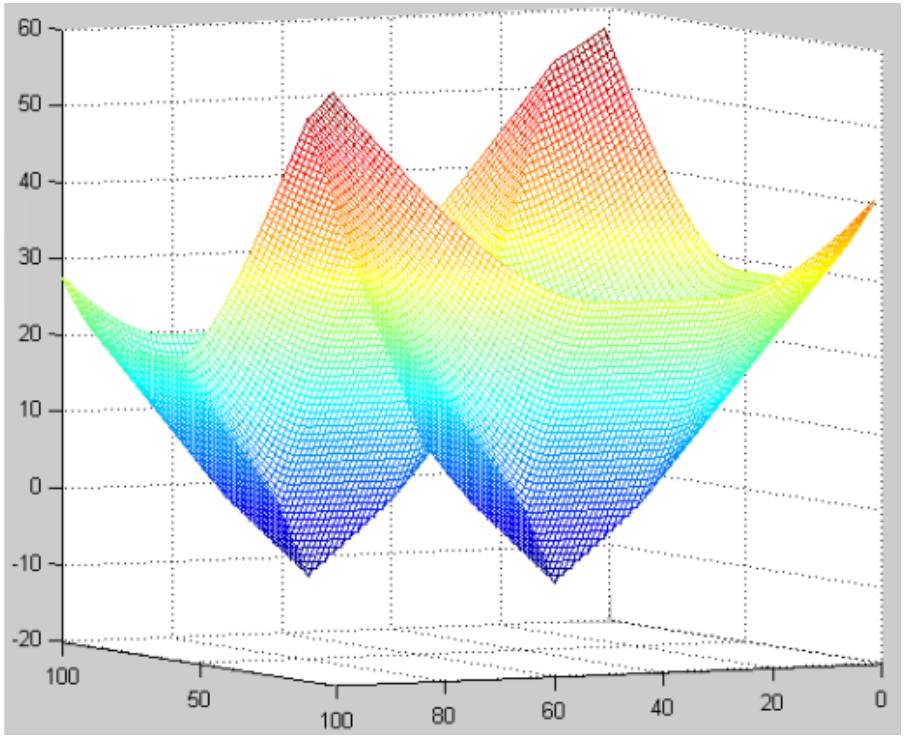
Simultanous Translation and Expansion

$$\frac{\partial f}{\partial t} + \vec{S} \cdot \nabla f + V_N \left| \nabla f \right| = 0$$



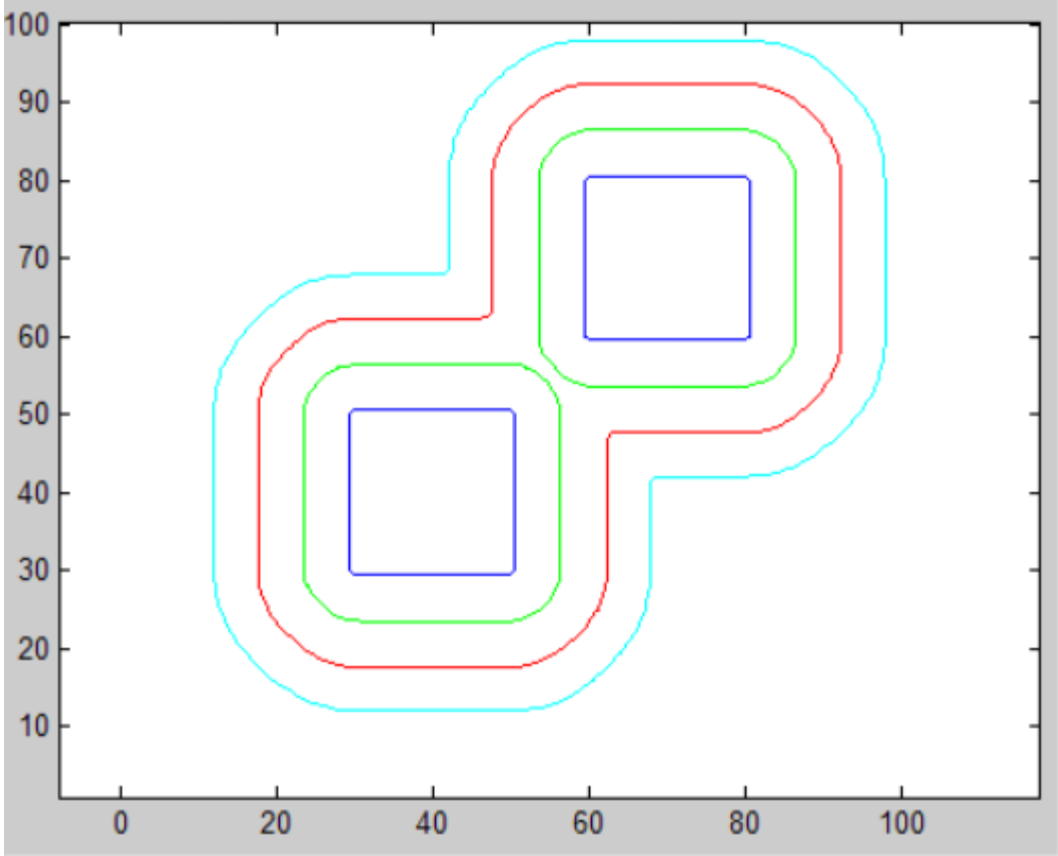
Multi-part curves





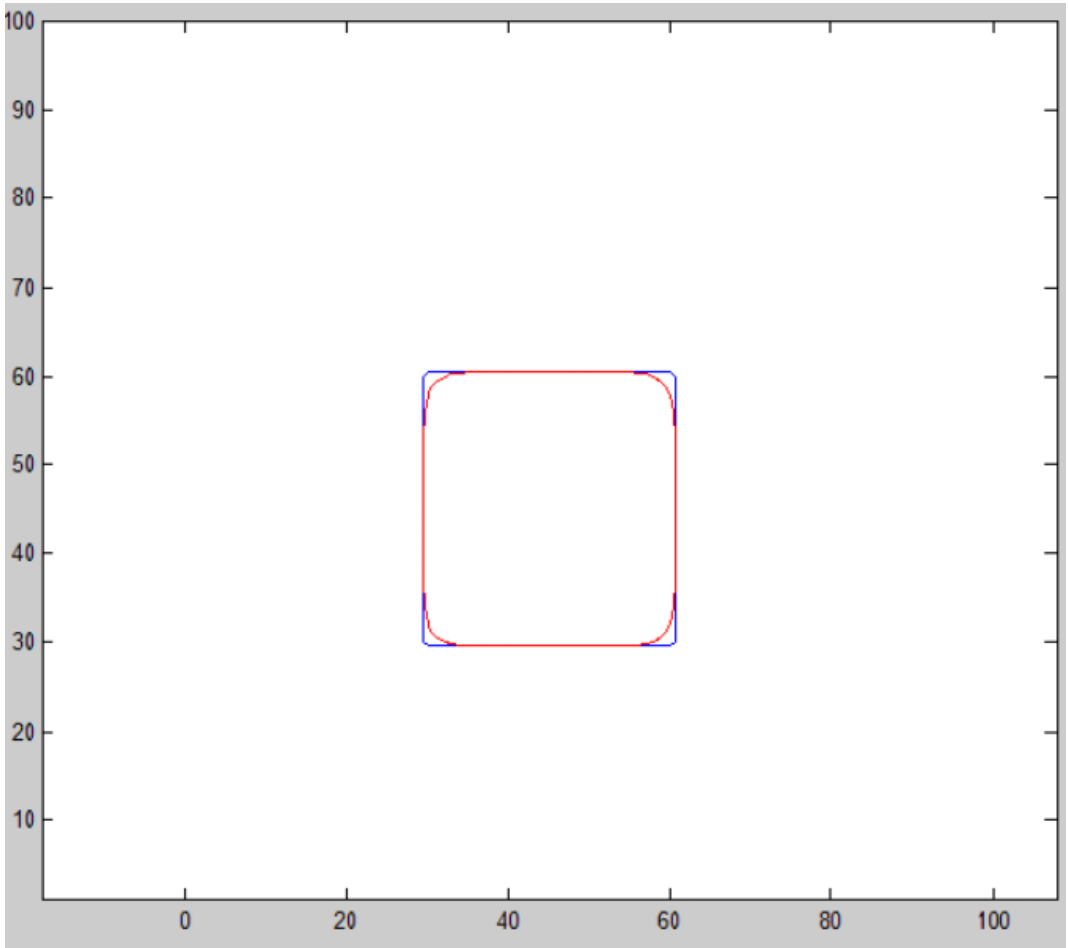
Merging of curves

- Topological changes such as merging and splitting are taken care of by level set.



Curvature-based force

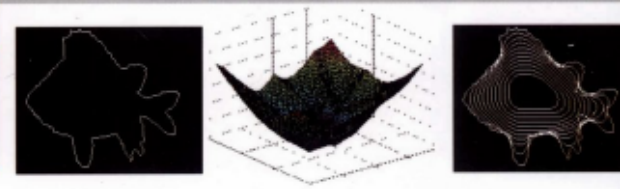
$$\frac{\partial f}{\partial t} = b\kappa|\nabla f|$$



研究生推荐用书

图像处理的偏微分方程方法

王大凯　侯榆青　彭进业　编著



科学出版社
www.sciencep.com

Chunming Li's homepage

http://www.engr.uconn.edu/~cmli/

Chunming Li

Ph.D. in Electrical Engineering (UConn)
M.S. in Mathematics (Fudan)

News:

My IEEE TIP'08 paper (700+ citations since 2008) received *2013 IEEE Signal Processing Society Best Paper Award*

Vistors since 2005

Publications | Research | Google Scholar Profile | Award | Professional activities | Collaborators | Software | Contact me | Personal

Selected Publications (with 4100+ citations since 2006)

Research themes: *Level set method, image segmentation, image registration, MRI bias field correction (intensity inhomogeneity correction), brain, illumination and reflectance, variational methods, partial differential equation for image processing and computer vision...*

Papers and code (in Matlab/C/C++)

Papers	Code
• **C. Li**, J.C. Gore, and C. Davatzikos, "Multiplicative intrinsic component optimization (MICO) for MRI bias field estimation and tissue segmentation", ***Magnetic Resonance Imaging*** , vol. 32 (7), pp. 913-923, 2014 (pdf, BibTex,website)	Code available here
• **C. Li**, C. Xu, C. Gui, and M. D. Fox, "Distance Regularized Level Set Evolution and its Application to Image Segmentation", ***IEEE Trans. Image Processing***, vol. 19 (12), pp. 3243-3254, 2010 (pdf, BibTex,**400+ citations**, website) (This paper generalizes and improves our preliminary work in CVPR'05, which has received **1300+ citations** since 2006)	Code available here
• **C. Li**, R. Huang, Z. Ding, C. Gatenby, D. N. Metaxas, and J. C. Gore, "A Level Set Method for Image Segmentation in the Presence of Intensity Inhomogene ities with Applicatio n to MRI", ***IEEE Trans. Image Processing***, vol. 20 (7), pp. 2007-2016, 2011.	Matlab code: .rar, (include two and

Learning PDE

PDE Methods in Computer Vision

- PDEs have been applied to
  - Denoising, Deblurring, Inpainting, Segmentation, Stereo, …
- Roughly have two kinds of methodologies for PDE design
  - Direct design: write down PDEs directly using some insights from physics

Now is **Winter** for PDE methods!!

Anisotropic Diffusion: $\frac{\partial u}{\partial t} = \mathrm{div}(c(\|\nabla u\|)\nabla u),\ u|_{t=0} = f,\ \frac{\partial f}{\partial n}|_{\partial D} = 0$

  - Variational design: energy functional $\longrightarrow$ Euler-Lagrange equation

Total Variation: $\min_u \int_\Omega \|\nabla u\| + \lambda \|f - u\|^2$

Not a hot topic in computer vision society recently!

Why PDEs fell out of favor?

- PDE is a powerful tool for data analysis
  - PDE is indeed a general regressor to approximate the nonlinear mapping between input and output vision data.
  - Differential system should be more efficient than linear equation for data analysis.
- Unfortunately…
  - It is challenging to design PDE in handcraft way (huge feasible solution set)
    - Deep insights for the problem and the data
    - High mathematical skills for building a PDE system
  - Fixed PDE formulation and/or boundary conditions is lack of flexibility

Beautiful mathematical formulation, BUT poor performance and limited application!!

New Perspective: PDE + Learning

Tow possible ways to incorporate “learning” ideas into PDE designing:

•For a specific kind of PDE system (e.g., heat diffusion), we may adaptively learn its form and boundary for different data and tasks

- Risheng Liu, Guangyu Zhong, Junjie Cao, Zhouchen Lin, Shiguang Shan, Zhongxuan Luo. Learning to Diffuse: A New Perspective to Design PDEs for Visual Analysis. TPAMI: 2457-2471 (2016)
- Risheng Liu, Junjie Cao, Zhouchen Lin and Shiguang Shan. Adaptive Partial Differential Equation Learning for Visual Saliency Detection. CVPR’2014 (Oral)

•For using PDE to address general vision tasks, we may learn the PDE form from the “**dictionary of differential operators**” for different image processing and analysis tasks

- Risheng Liu, Zhouchen Lin, Wei Zhang, Kewei Tang and Zhixun Su. Toward Designing Intelligent PDEs for Computer Vision: An Optimal Control Approach. Image and Vision Computing, 2013
- Risheng Liu, Zhouchen Lin, Wei Zhang and Zhixun Su. Learning PDEs for Image Restoration via Optimal Control. ECCV’2010

Learning-Based PDE for Computer Vision

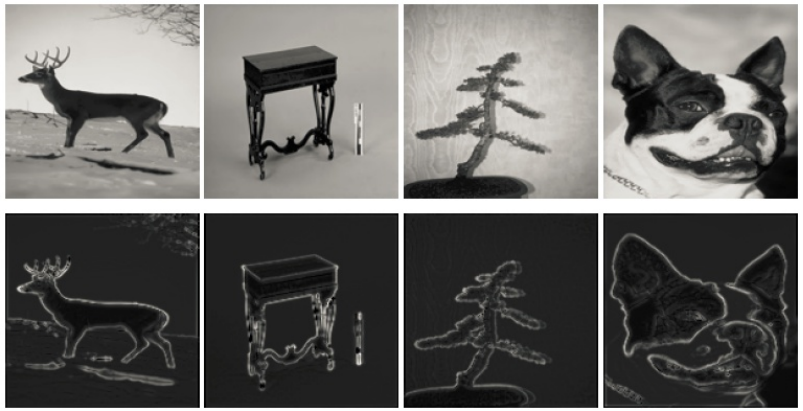
$$J(\{u_k\}_{k=1}^K, \mathbf{a}) = \frac{1}{2}\sum_{k=1}^{K}\int_{\Omega}(u_k(T_f) - \tilde{u}_k)^2 d\Omega + \frac{1}{2}\sum_{i=0}^{5}\alpha_i \int_0^{T_f} a_i^2(t)dt,$$

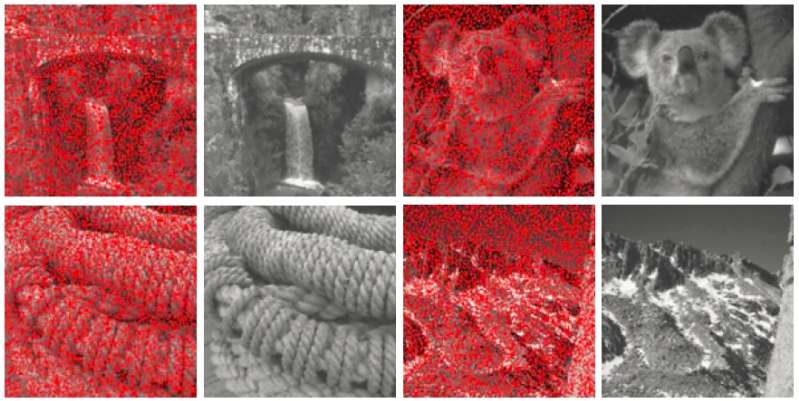
$$\min_{\mathbf{a}} J(\{u_k\}_{k=1}^K, \mathbf{a}), \quad s.t. \begin{cases} \frac{\partial u_k}{\partial t} = L(u_k, \mathbf{a}), & (x, y, t) \in Q, \\ u_k = 0, & (x, y, t) \in \Gamma, \\ u_k|_{t=0} = f_k, & (x, y) \in \Omega. \end{cases}$$

$$L(u, \mathbf{a}) = \kappa(u) + F(u, \mathbf{a}) \qquad F(u, \mathbf{a}) = \mathbf{a}(t)^T \mathbf{inv}(u),$$

i	$\mathrm{inv}_i(u)$
0	f
1	u
2	$\|\nabla u\|^2 = u_x^2 + u_y^2$
3	$\mathrm{tr}(\mathbf{H}_u) = u_{xx} + u_{yy}$
4	$\mathrm{tr}(\mathbf{H}_u^2) = u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2$
5	$(\nabla u)^T \mathbf{H}_u (\nabla u) = u_x^2 u_{xx} + 2u_x u_y u_{xy} + u_y^2 u_{yy}$

Learning-Based PDE for Computer Vision





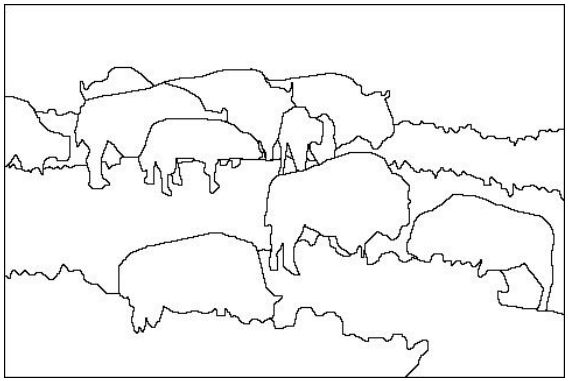
- Risheng Liu, Zhouchen Lin, Wei Zhang, Zhixun Su. Learning PDEs for Image Restoration via Optimal Control. ECCV, 2010
- Risheng Liu, Zhouchen Lin, Wei Zhang, Kewei Tang, Zhixun Su. Toward Designing Intelligent PDEs for Computer Vision: A Data-based Optimal Control Approach, Image and Vision Computing, 2012.

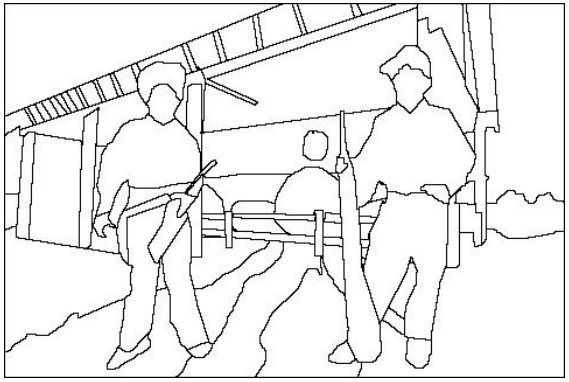
The goals of segmentation

- Separate image into coherent “objects”

image

human segmentation

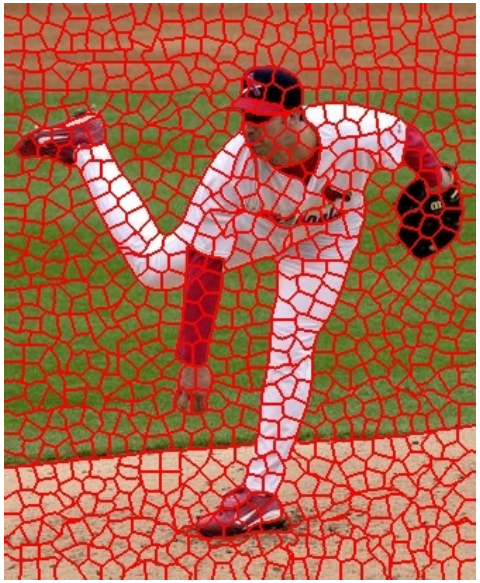




The goals of segmentation

- Group together similar-looking pixels for efficiency of further processing

"superpixels"



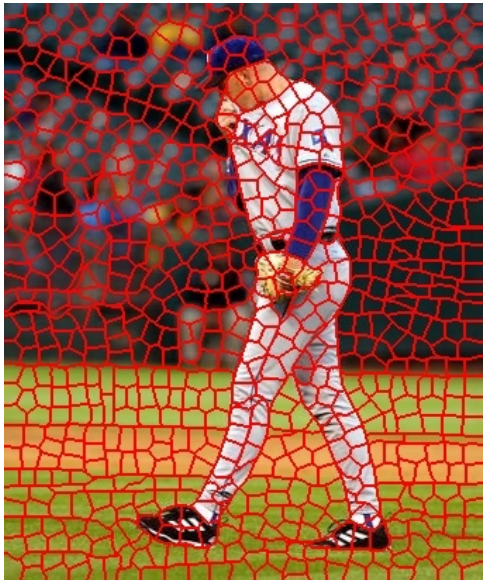
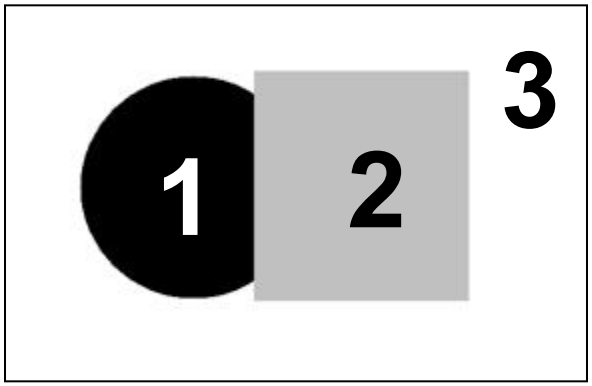
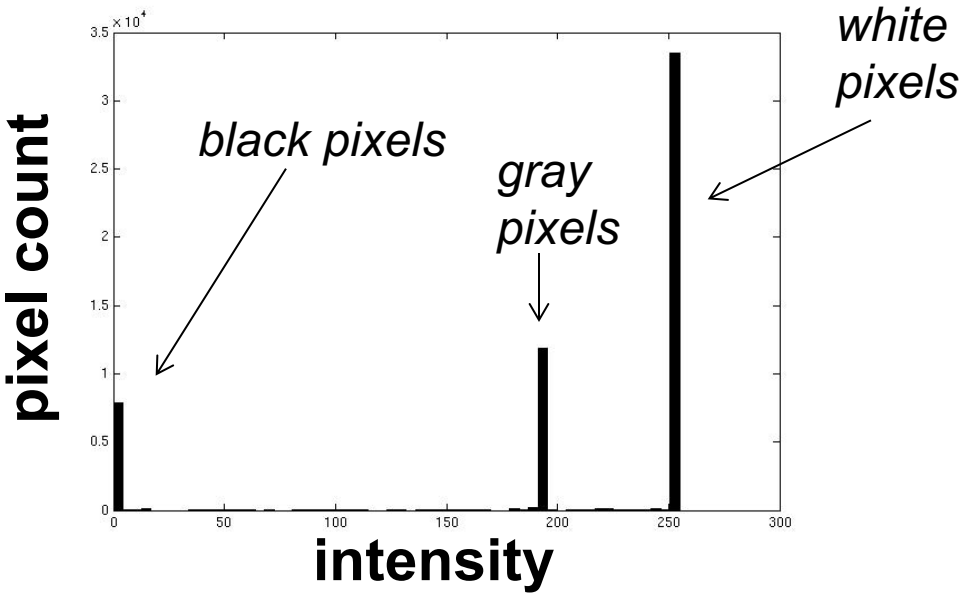


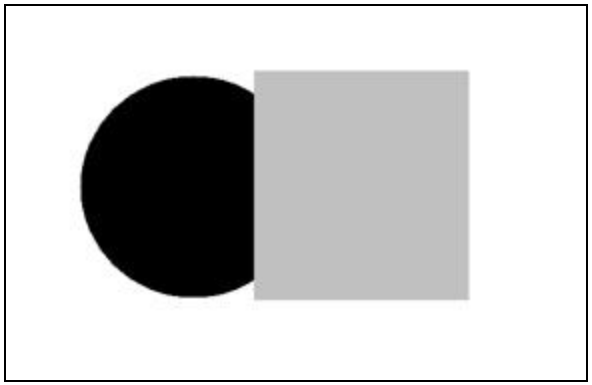
Image segmentation: toy example



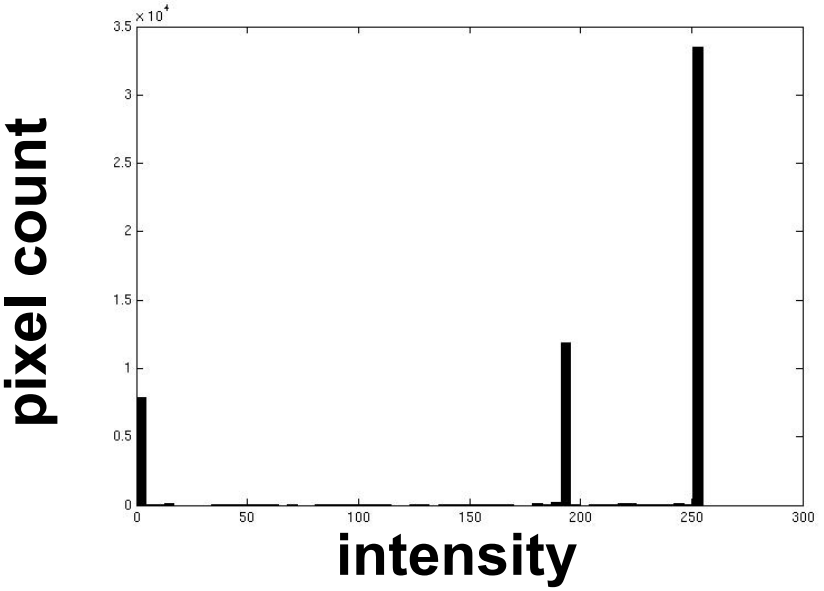
input image

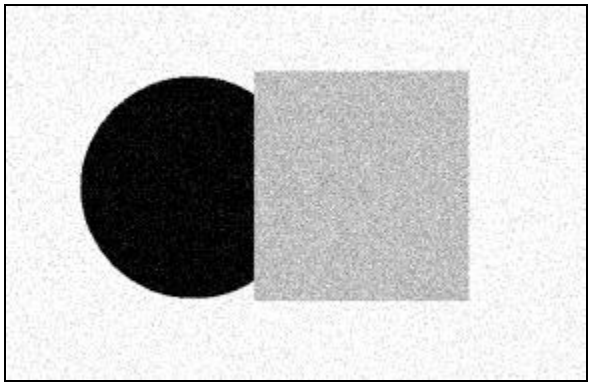


- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

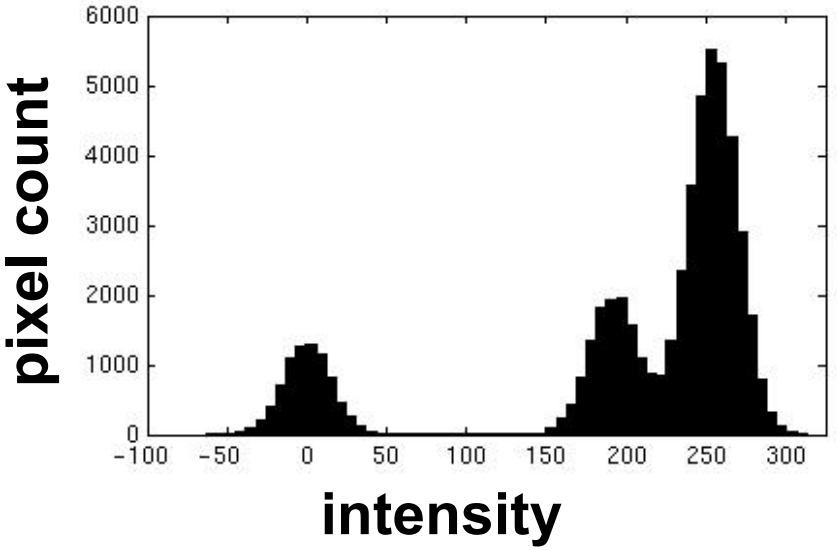


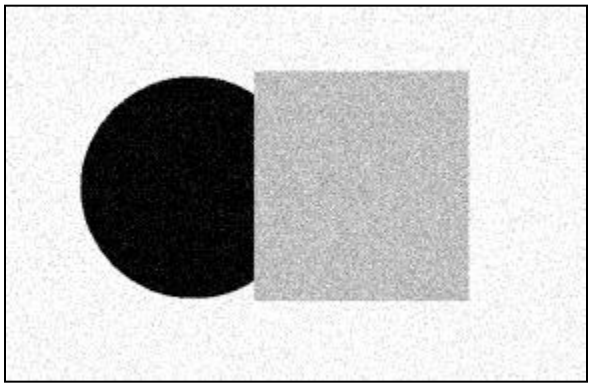
input image



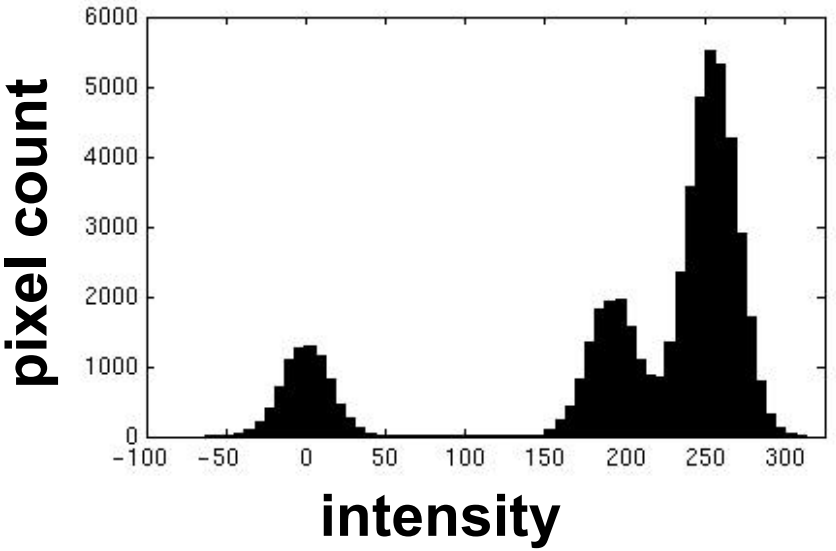


input image

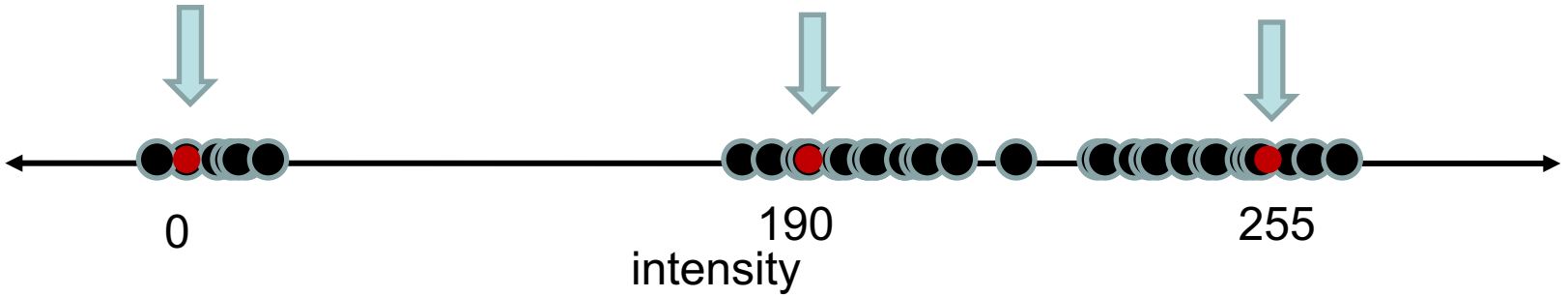


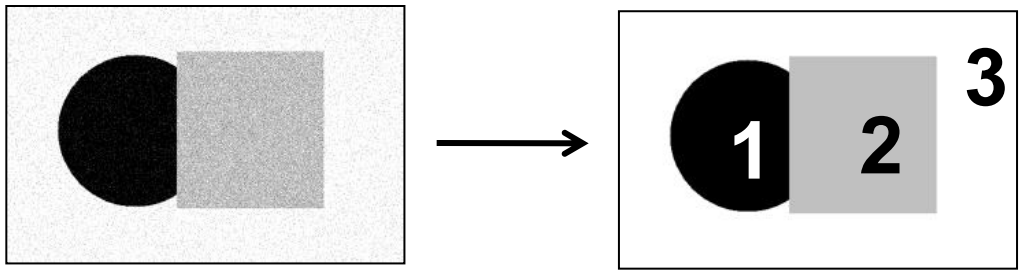


input image



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



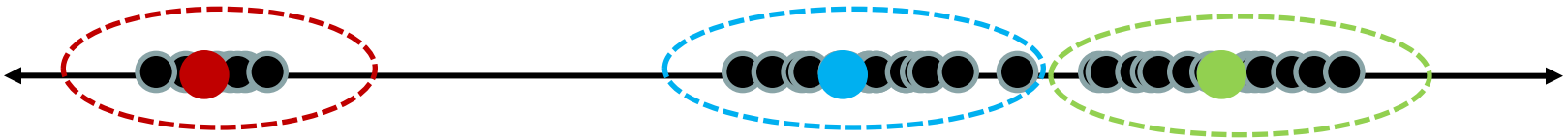


- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

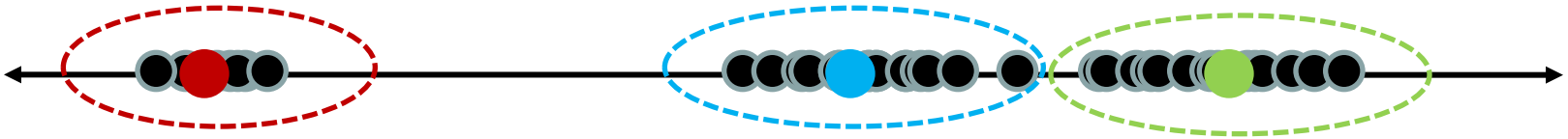
$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

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.



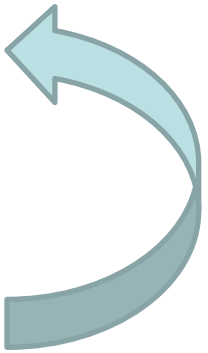
  - If we knew the **group memberships**, we could get the centers by computing the mean per group.



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, ..., 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



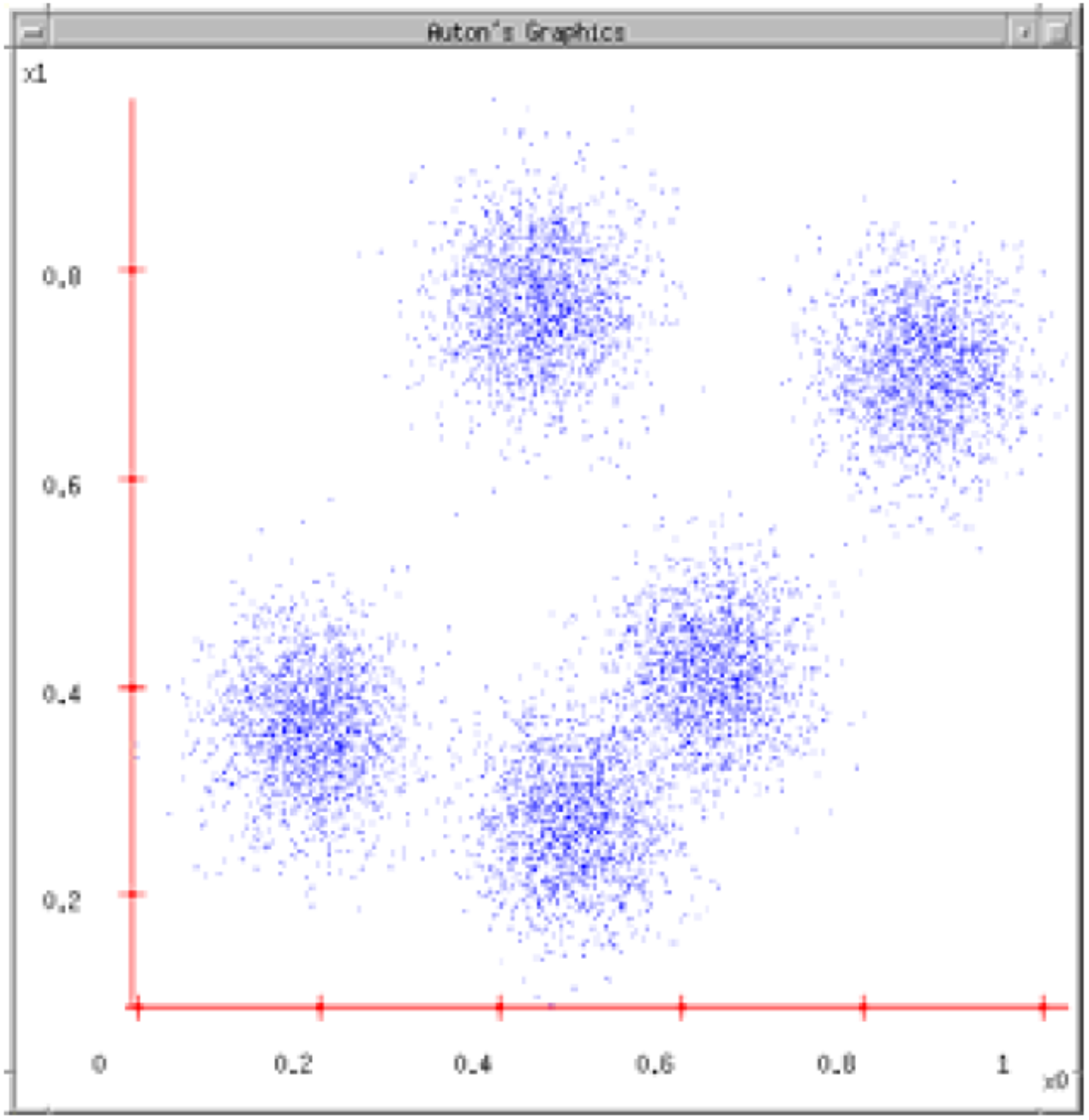
Properties

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

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

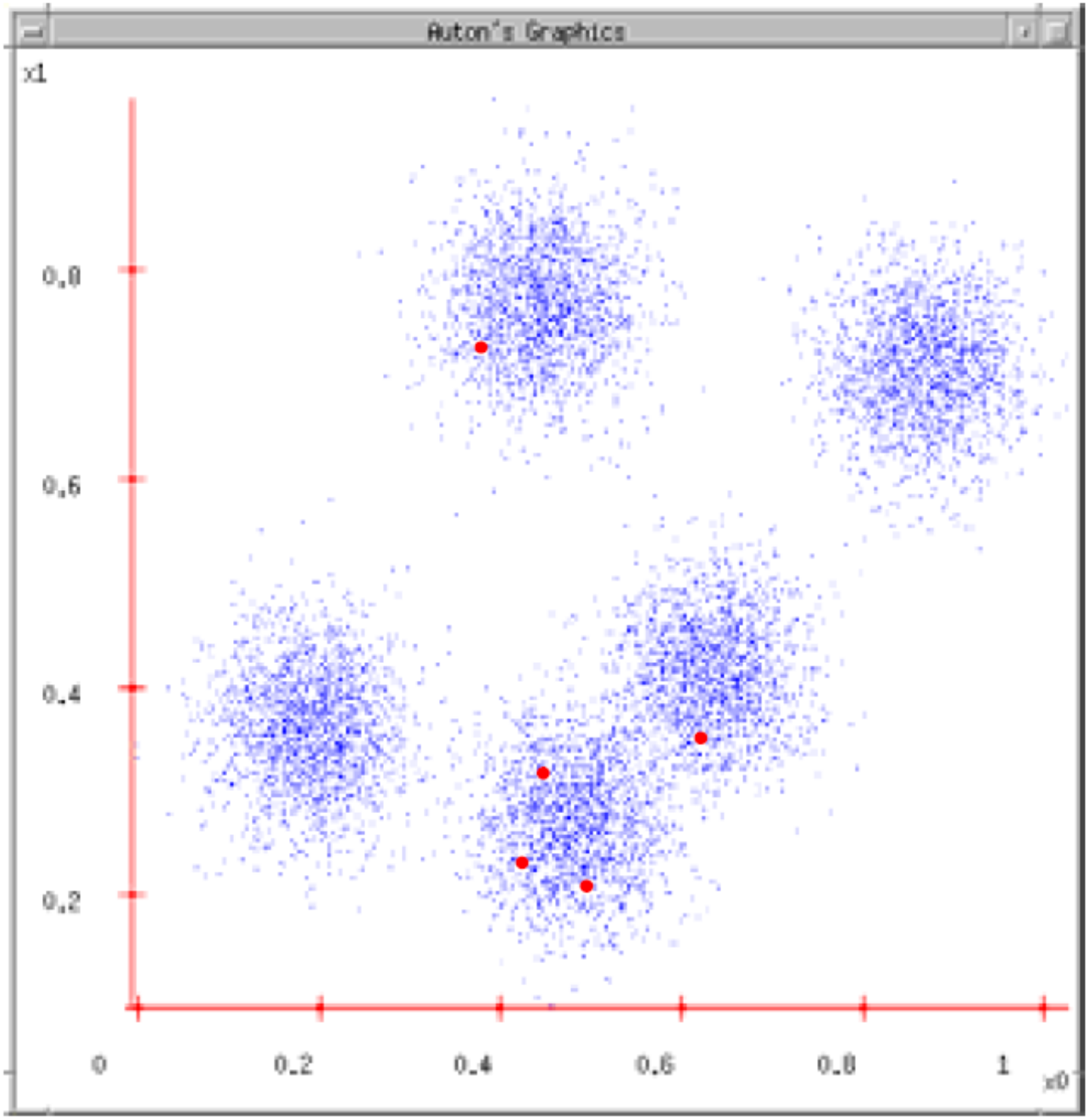
K-means

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



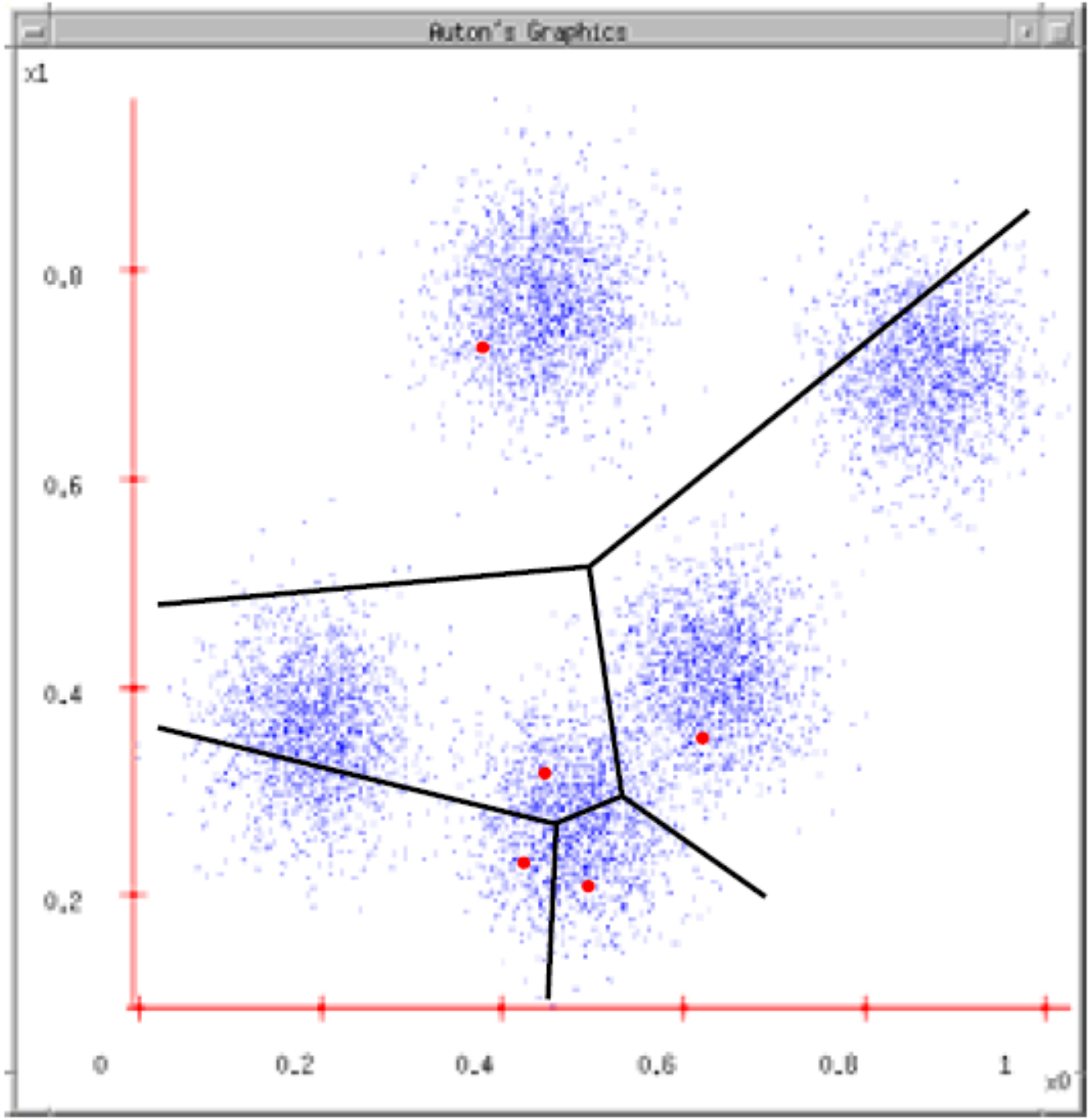
K-means

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



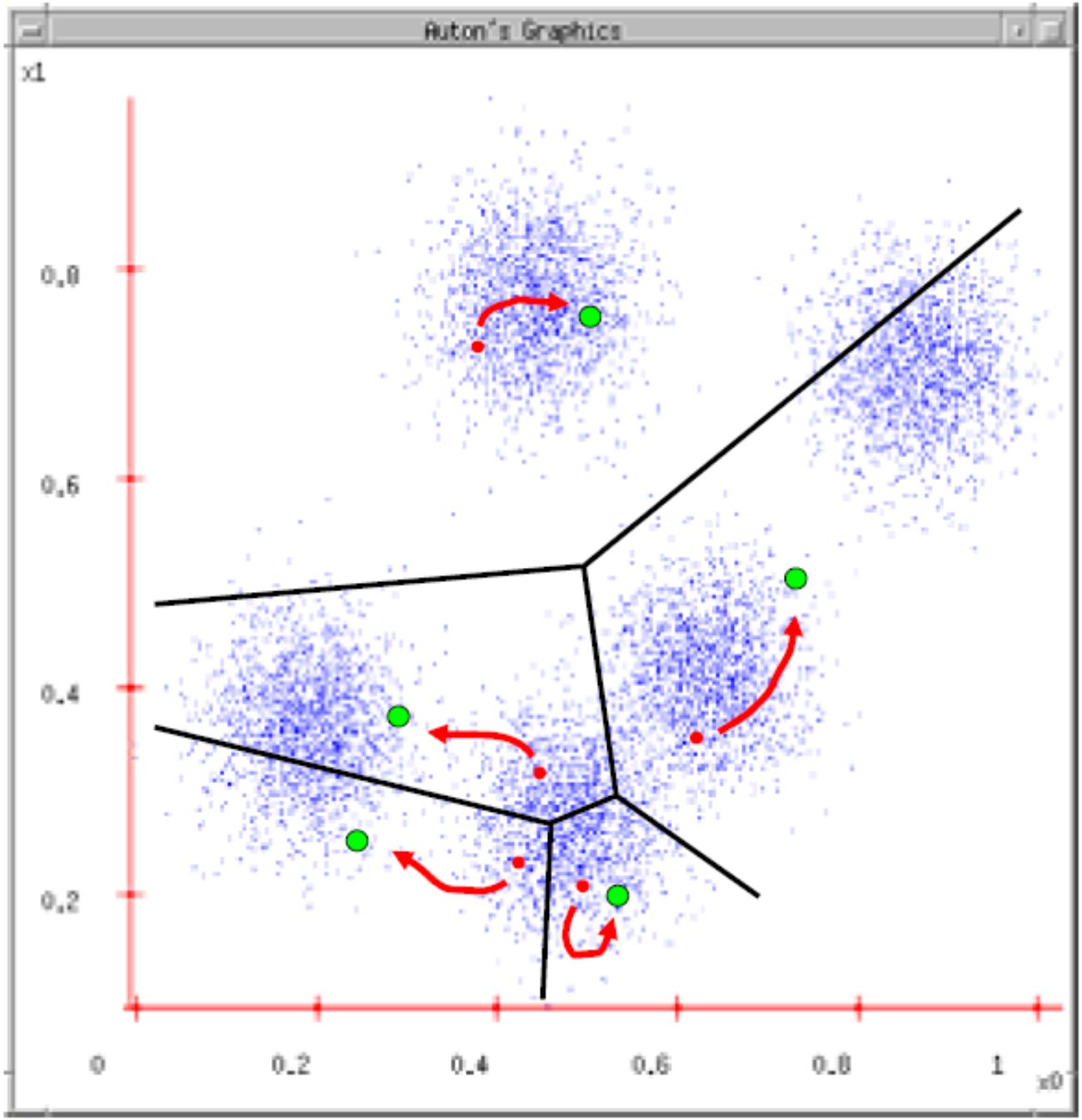
K-means

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)



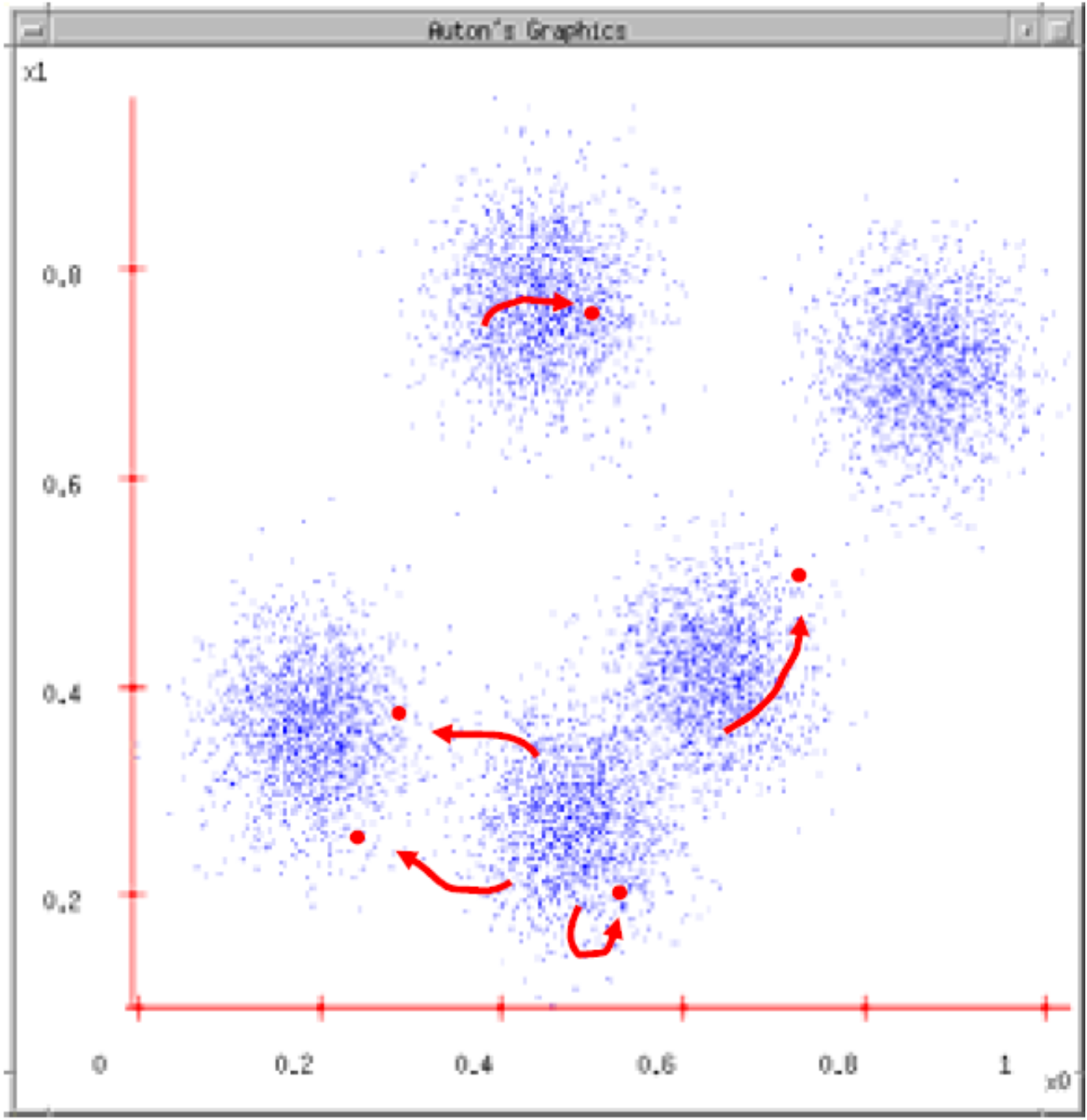
K-means

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!



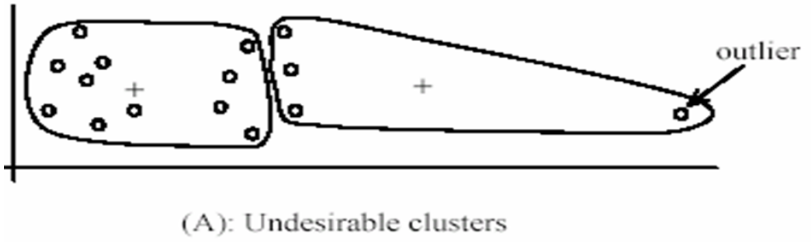
K-means: pros and cons

Pros

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

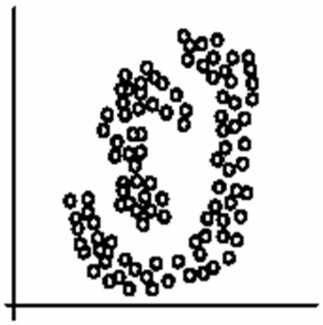
Cons/issues

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers

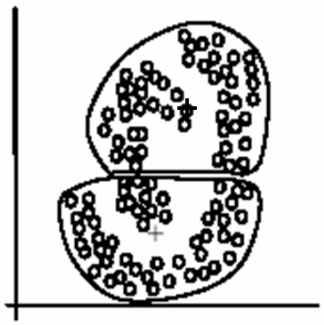


(A): Undesirable clusters

(B): Ideal clusters



(A): Two natural clusters



(B): *k*-means clusters

K-means clustering (MATLAB)

```
>> help kmeans
 kmeans K-means clustering.
    IDX = kmeans(X, K) partitions the points in the N-by-P data matrix X
    into K clusters.  This partition minimizes the sum, over all clusters, of
    the within-cluster sums of point-to-cluster-centroid distances.  Rows of X
    correspond to points, columns correspond to variables.  Note: when X is a
    vector, kmeans treats it as an N-by-1 data matrix, regardless of its
    orientation.  kmeans returns an N-by-1 vector IDX containing the cluster
    indices of each point.  By default, kmeans uses squared Euclidean
    distances.

    kmeans treats NaNs as missing data, and ignores any rows of X that
    contain NaNs.

    [IDX, C] = kmeans(X, K) returns the K cluster centroid locations in
    the K-by-P matrix C.
```

Segmentation as clustering

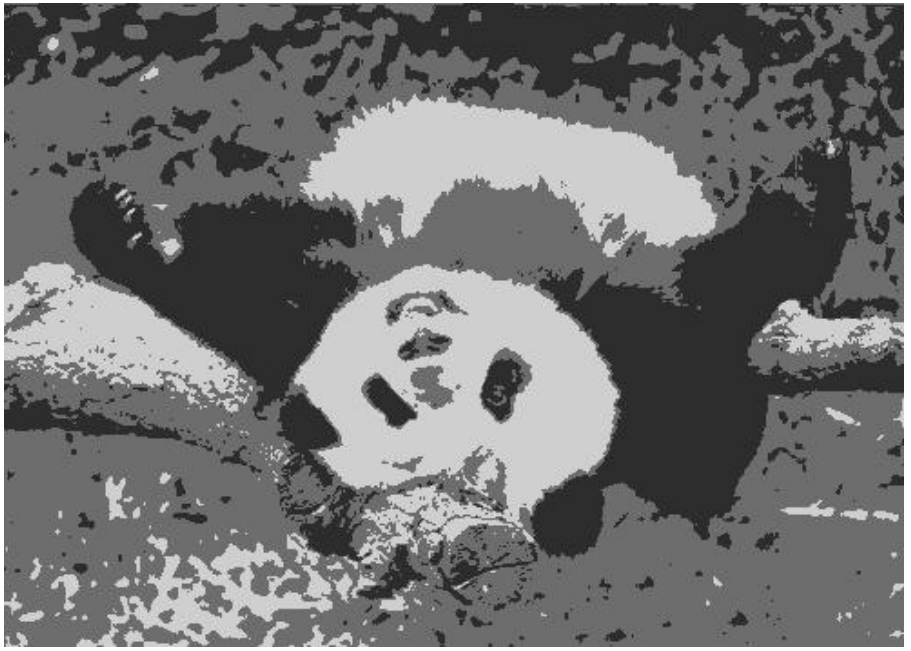
Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity** similarity

Feature space: intensity value (1-d)

K=2

K=3

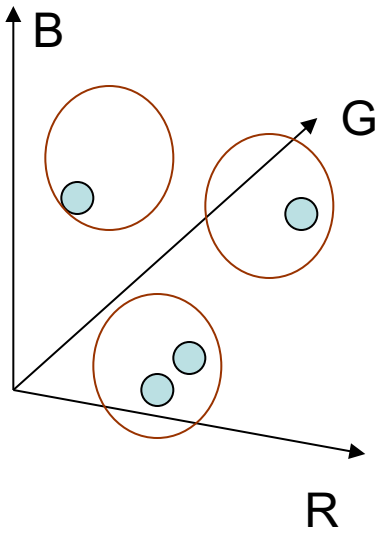


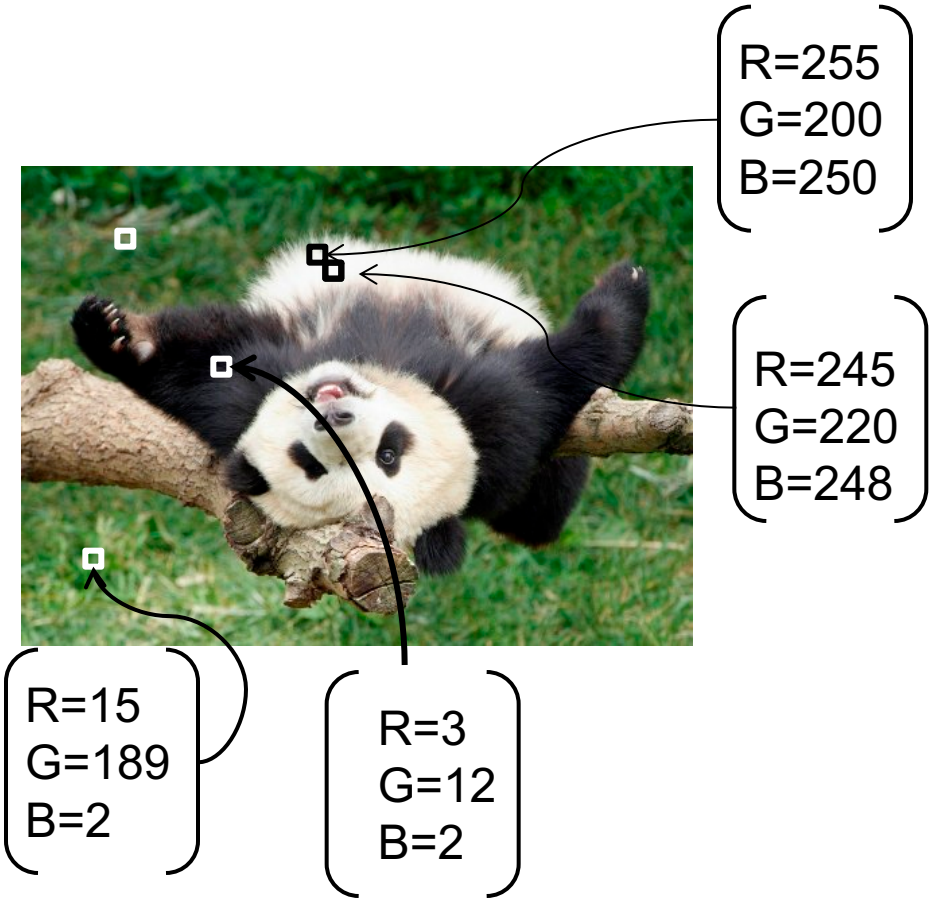
quantization of the feature space;
segmentation label map

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)

Segmentation as clustering

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

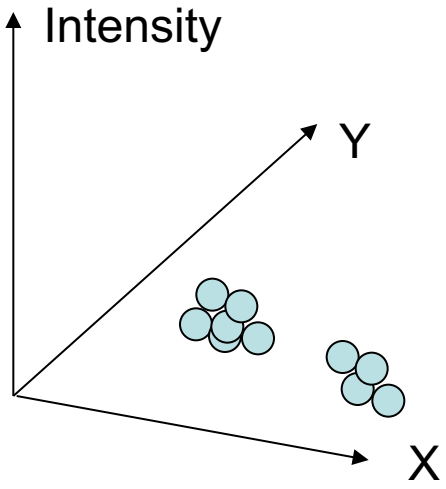
Grouping pixels based on **intensity** similarity

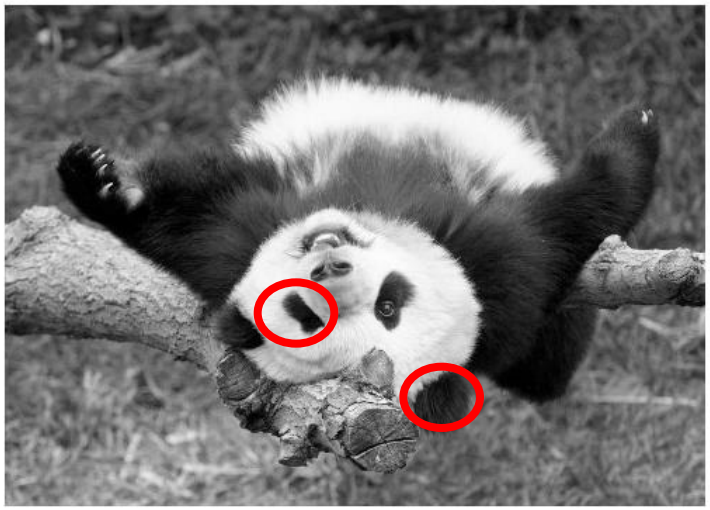
Clusters based on intensity similarity don't have to be spatially coherent.

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

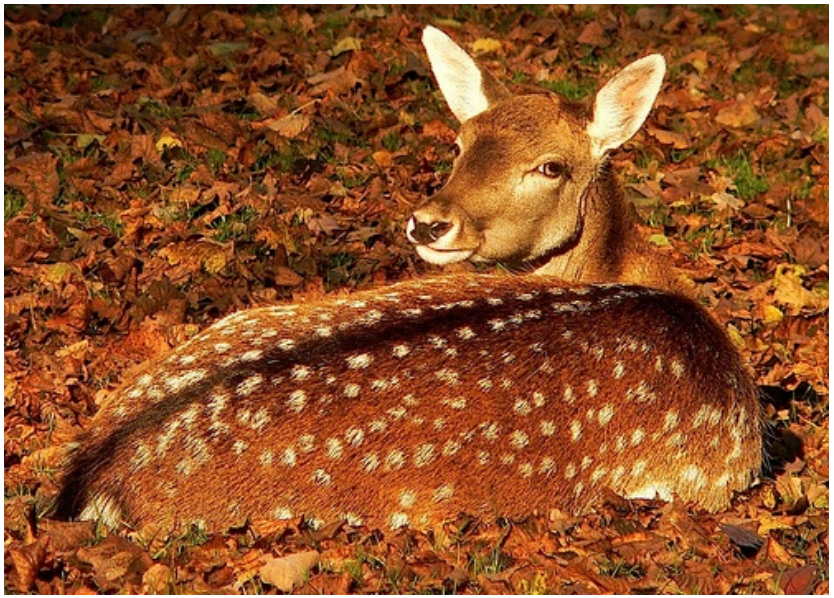




Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Segmentation as clustering

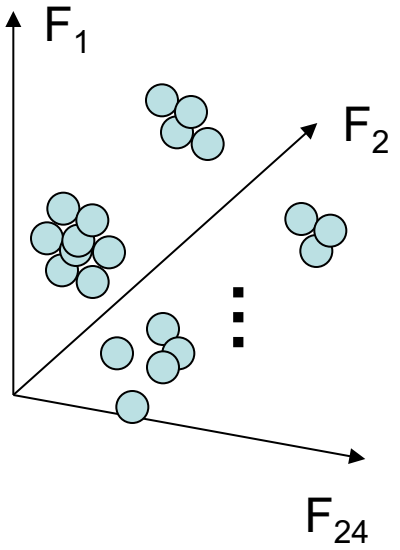
- Color, brightness, position alone are not enough to distinguish all regions…

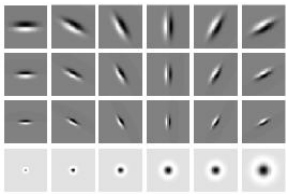


Segmentation as clustering

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

Grouping pixels based on **texture** similarity

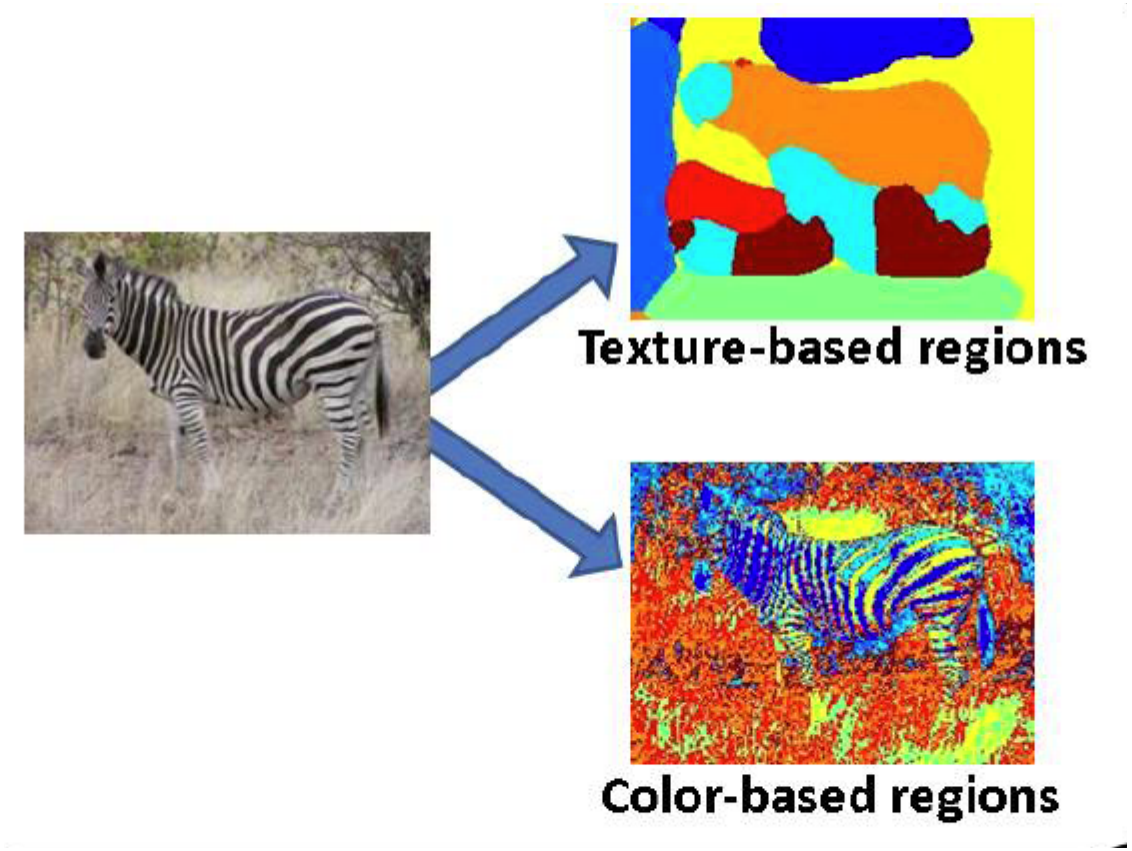




Filter bank of 24 filters

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

Image segmentation example

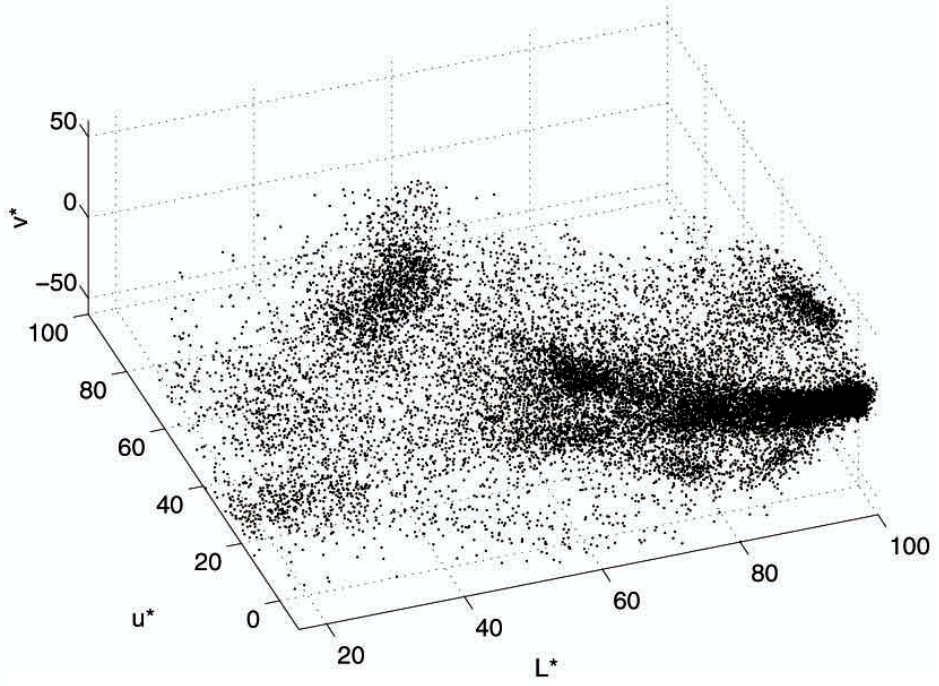


Mean shift algorithm

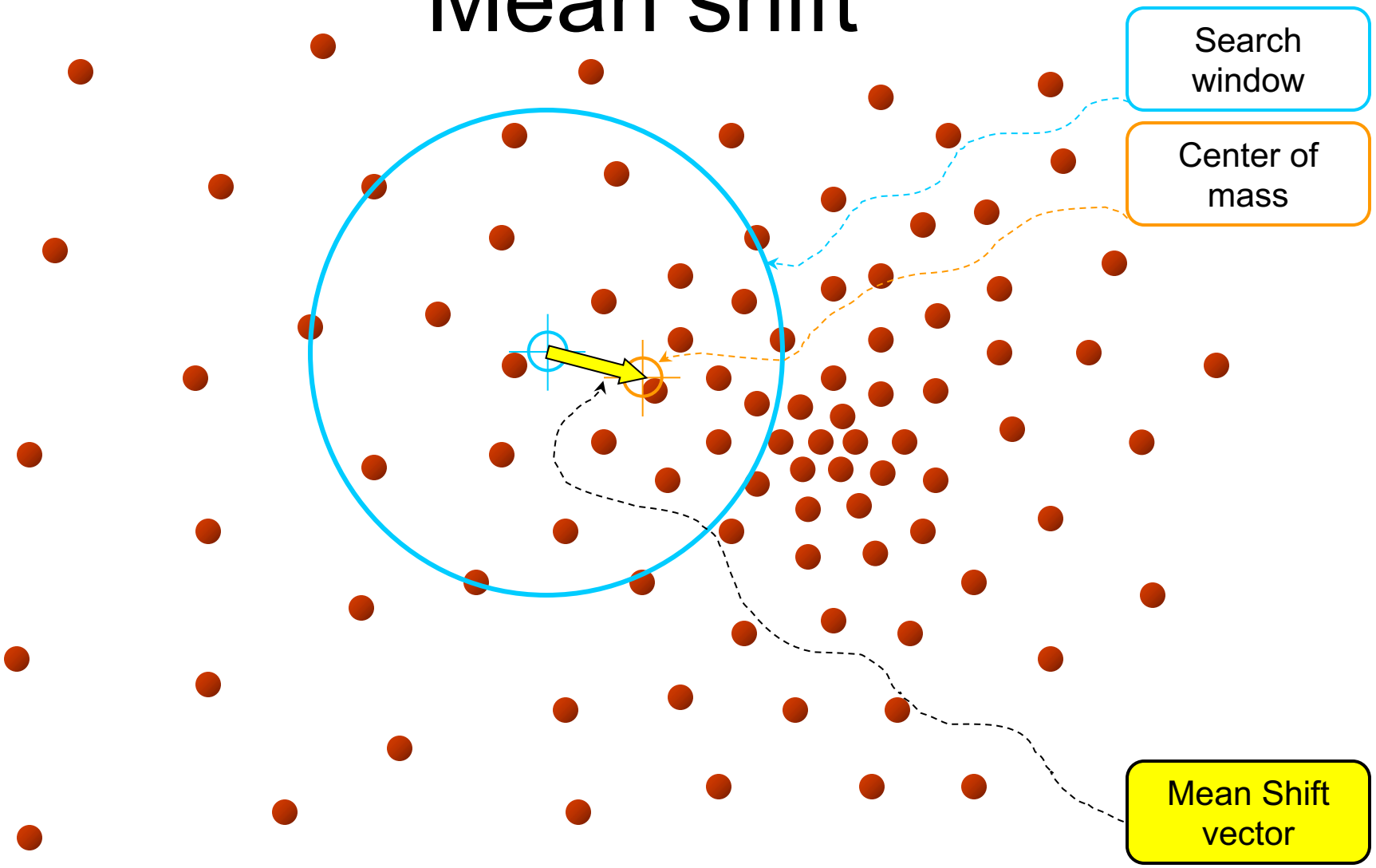
- The mean shift algorithm seeks *modes* or local maxima of density in the feature space

image

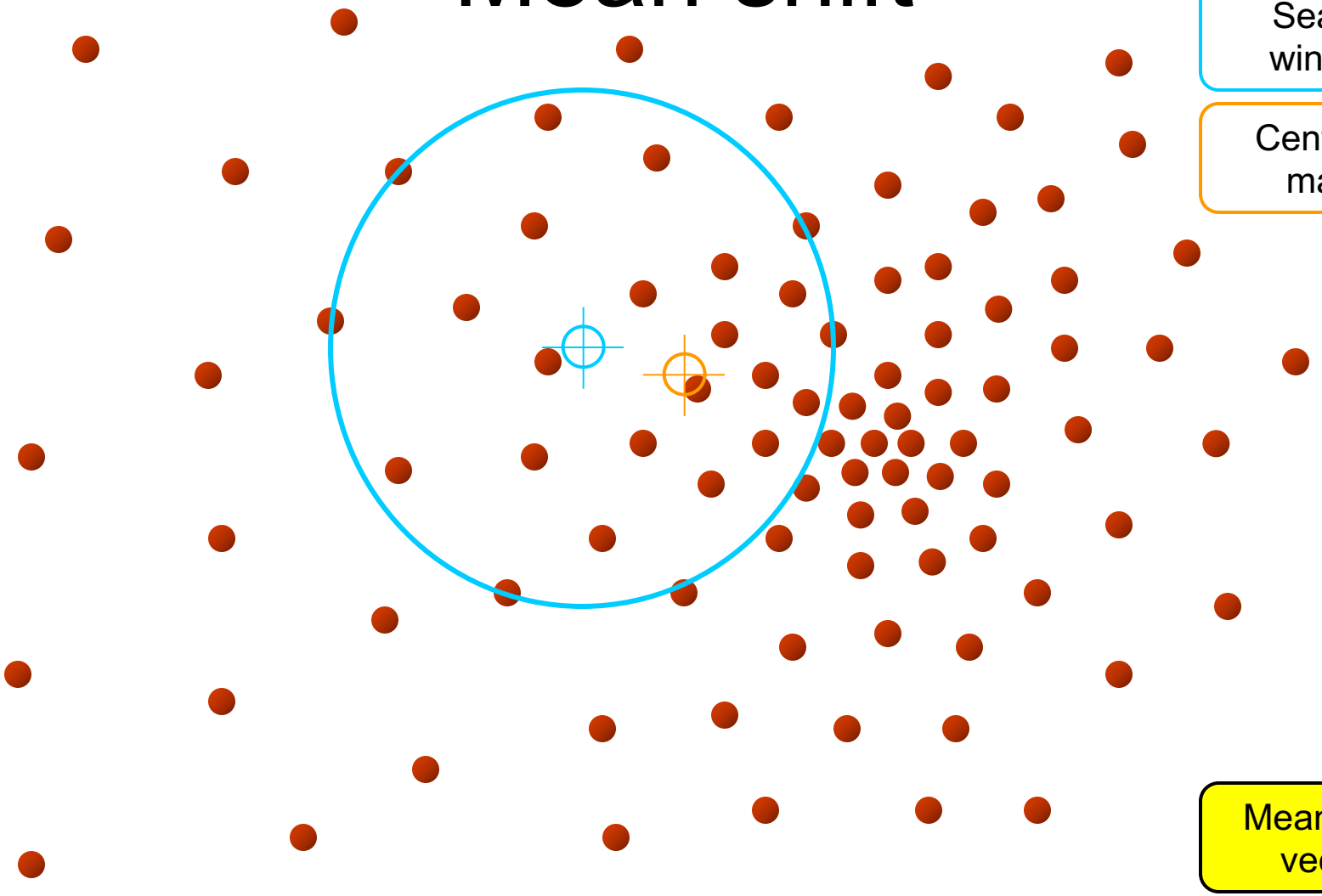
Feature space (L*u*v* color values)



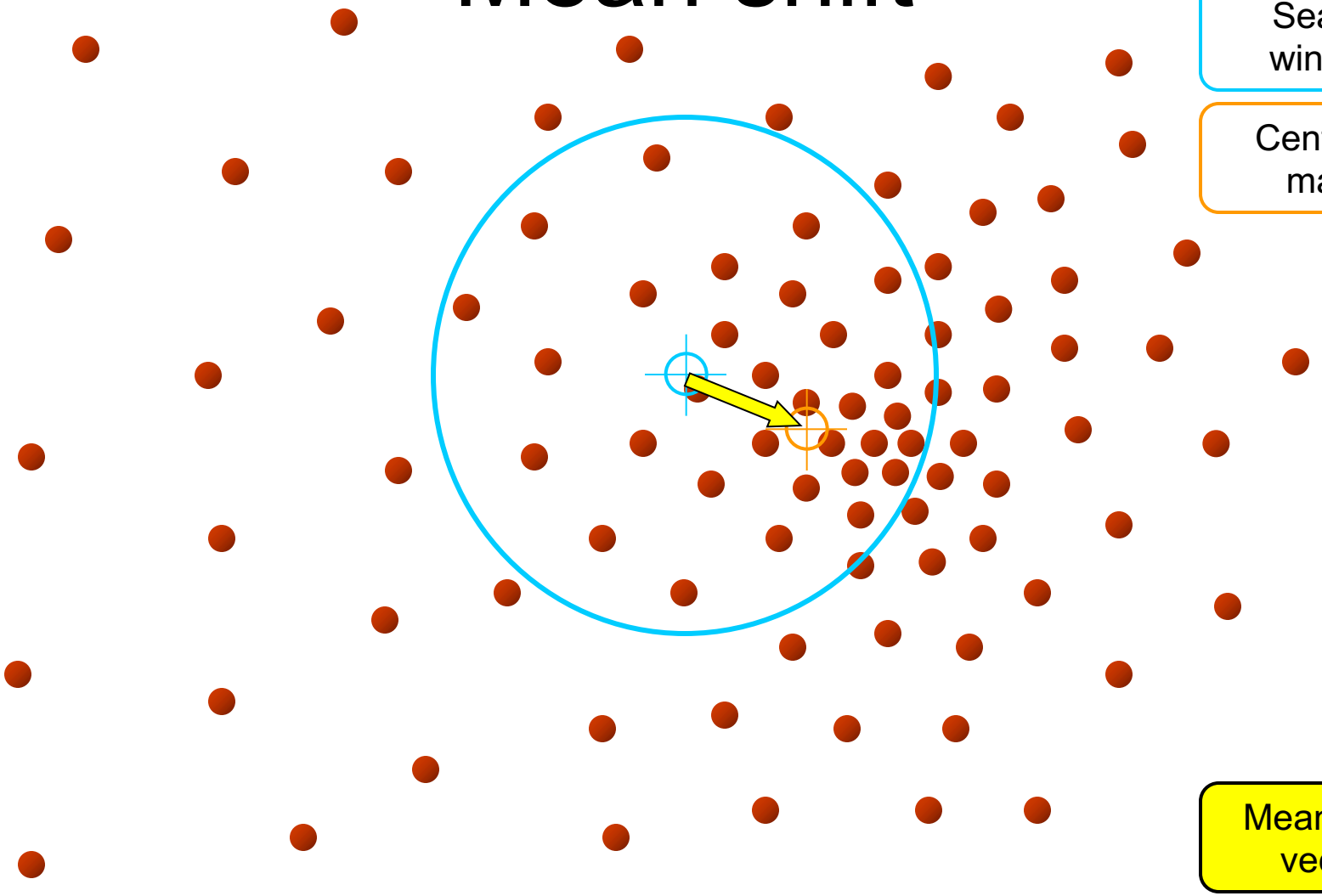
Mean shift



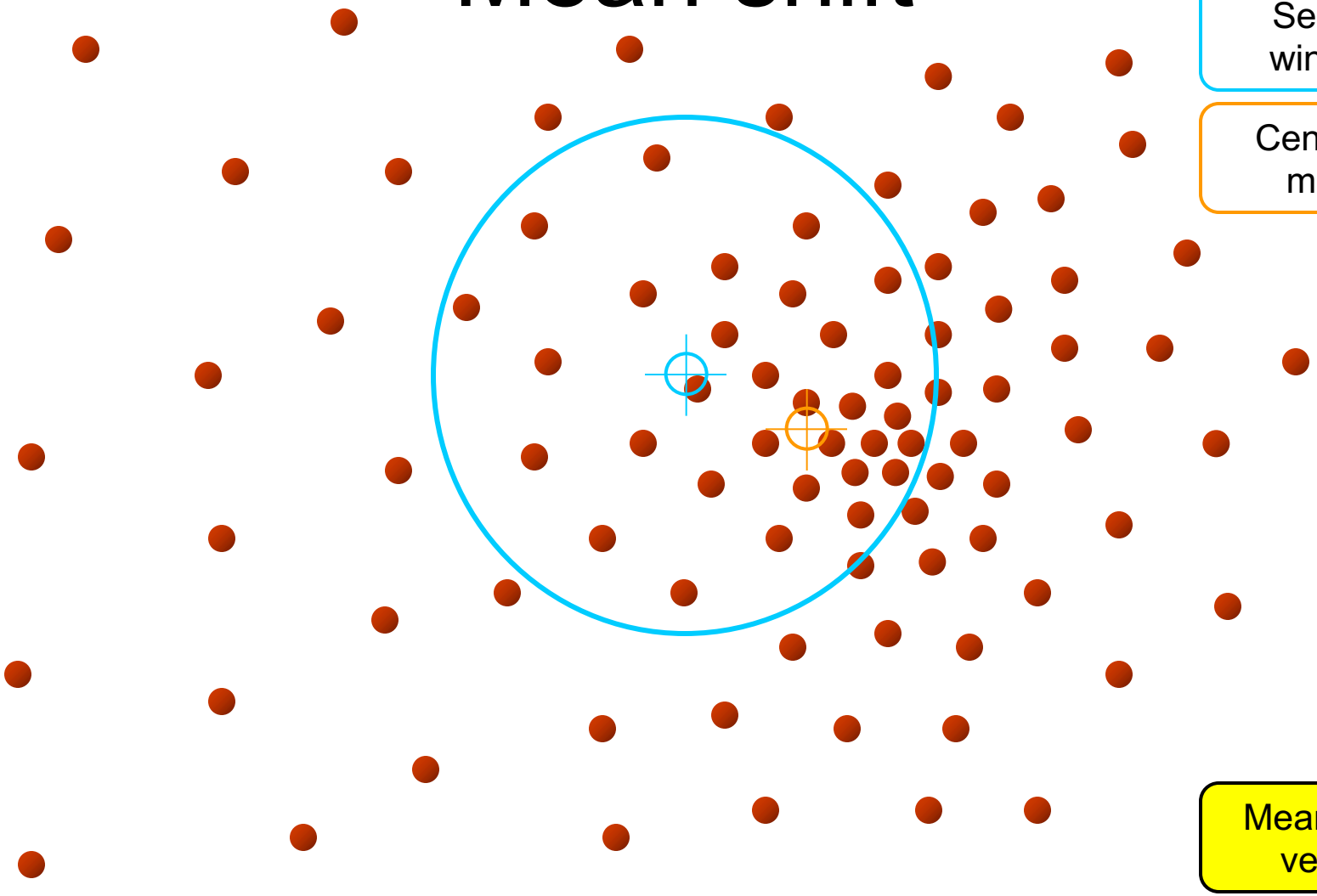
Mean shift



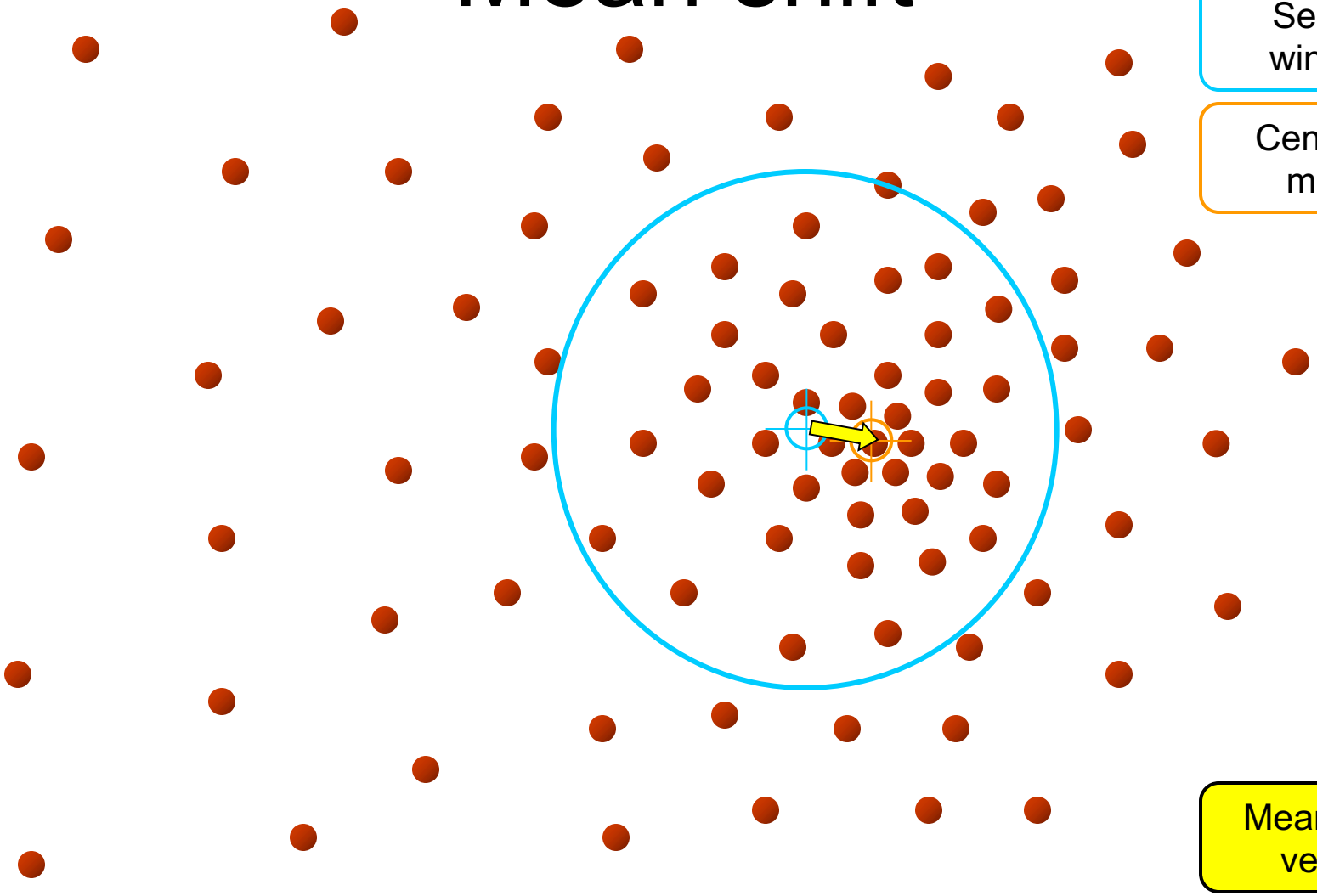
Mean shift



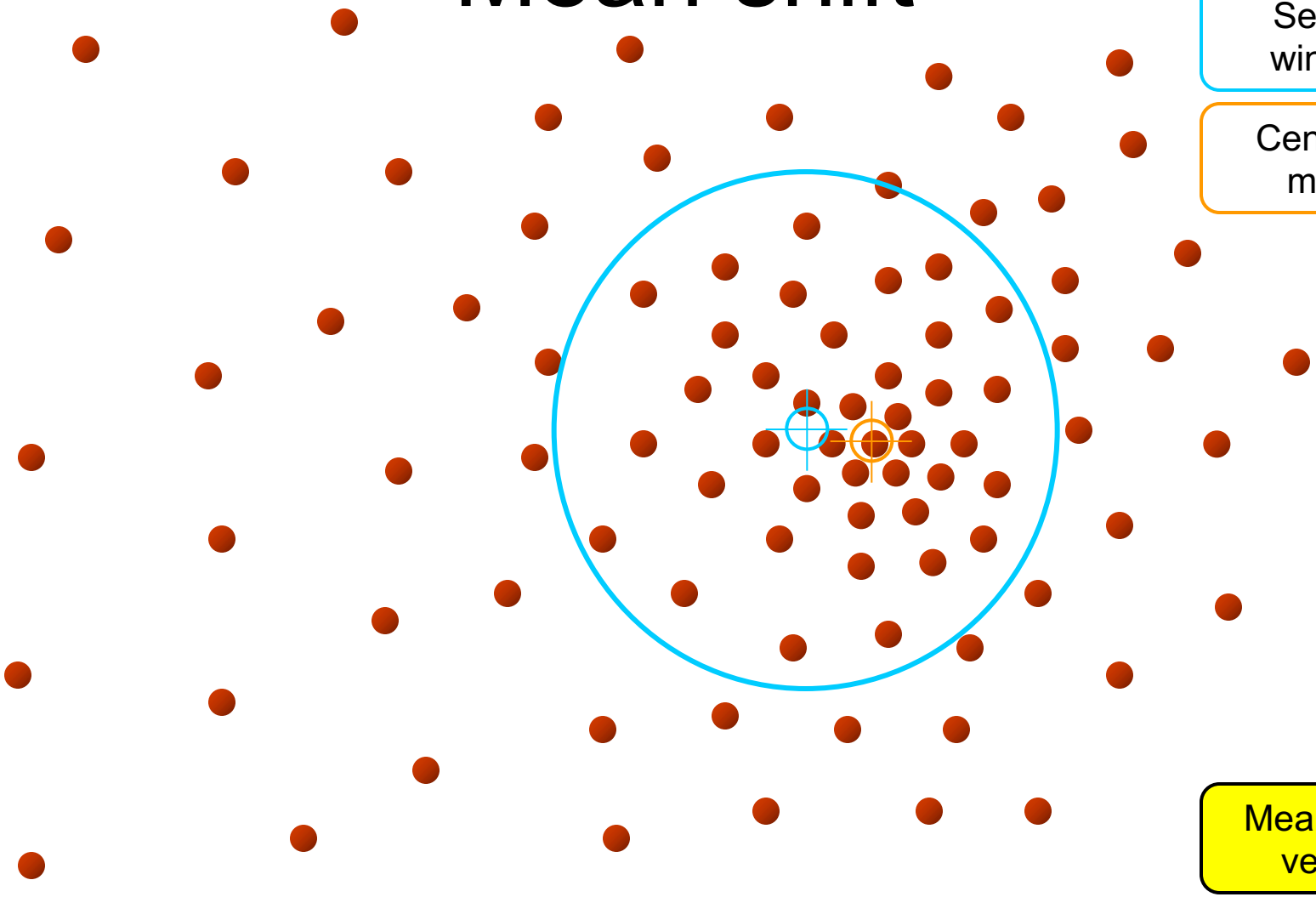
Mean shift



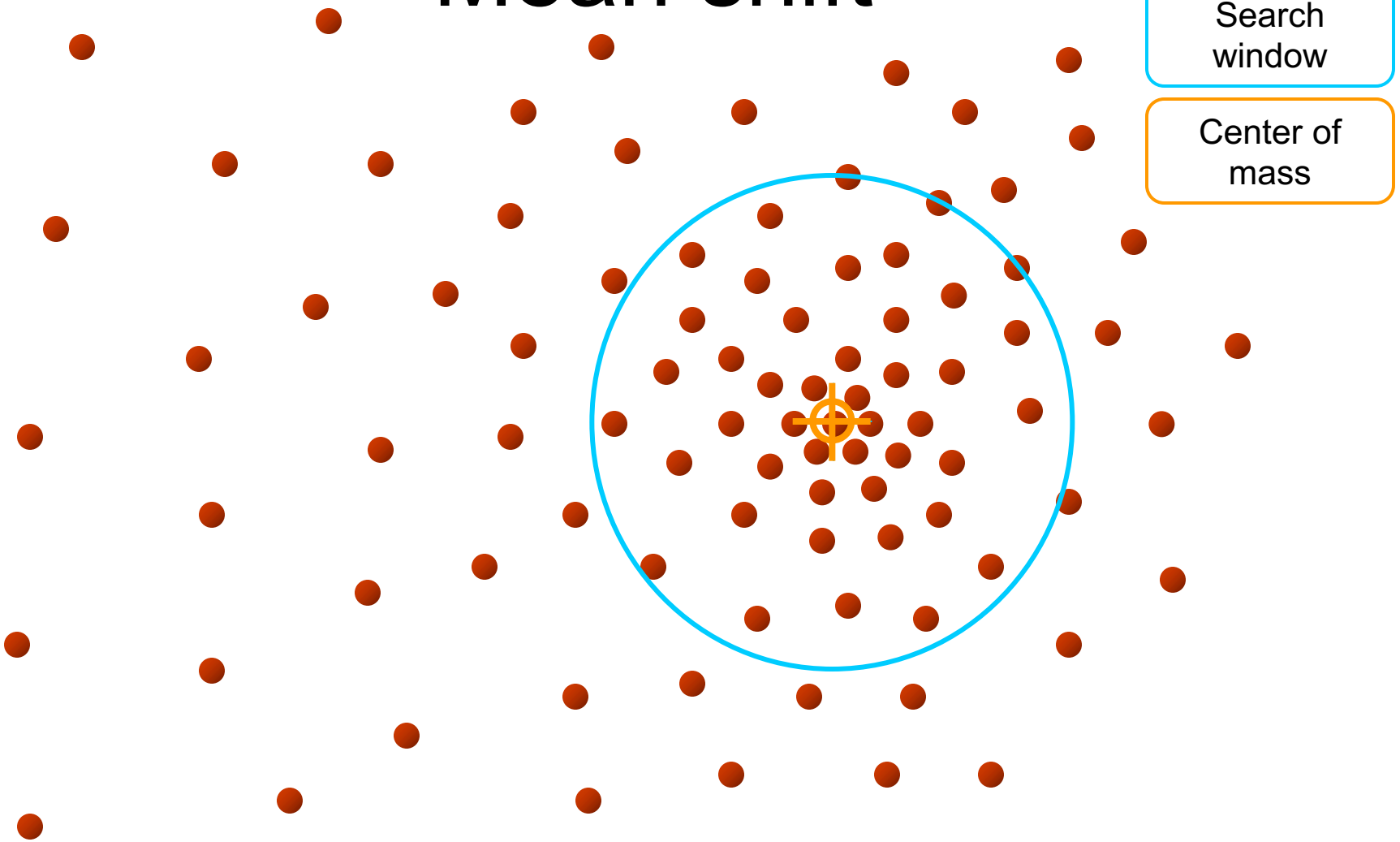
Mean shift



Mean shift

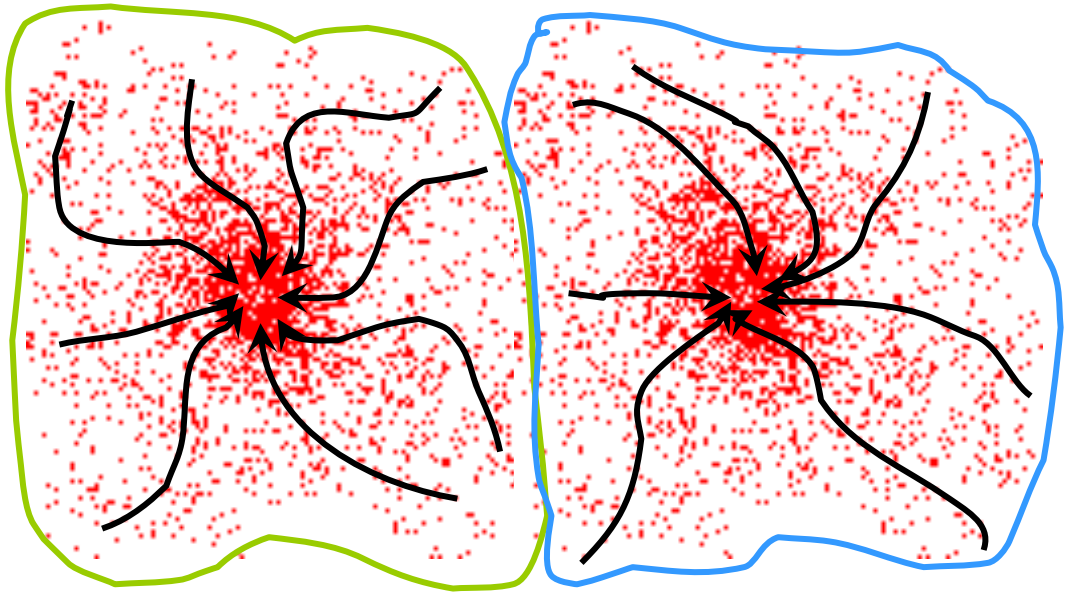


Mean shift



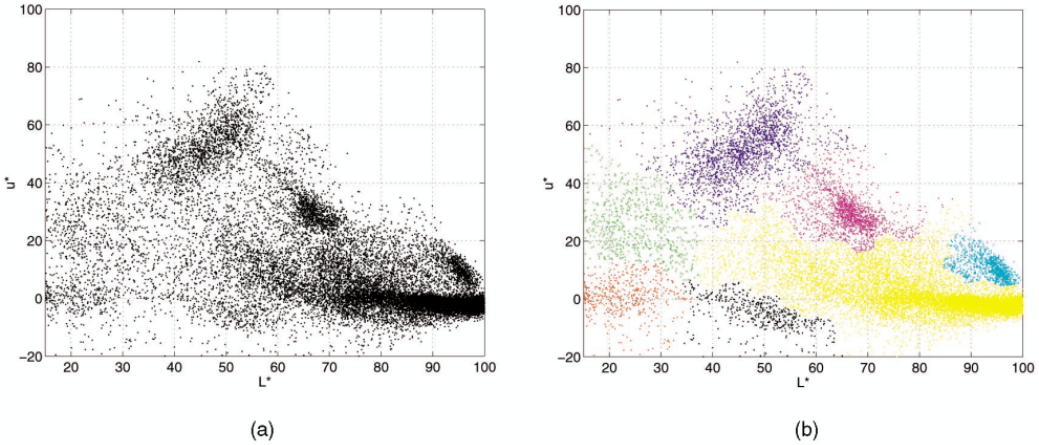
Mean shift clustering

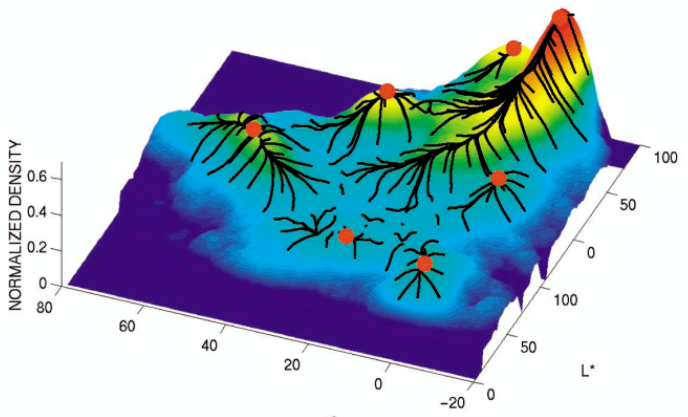
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



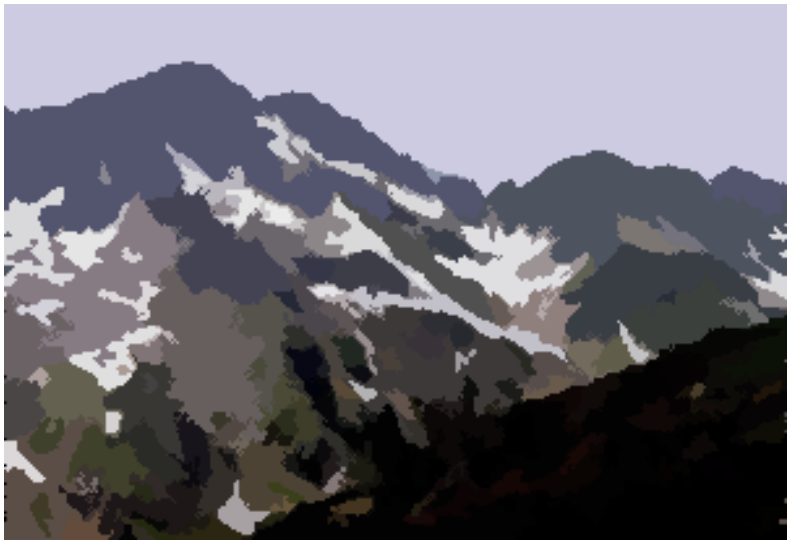
Mean shift segmentation

- 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





Mean shift segmentation results



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

Mean shift segmentation results

Mean shift

- Pros:
  - Does not assume shape on clusters
  - One parameter choice (window size)

- Cons:
  - Selection of window size
  - Does not scale well with dimension of feature space