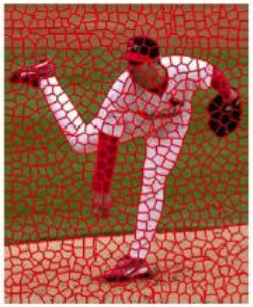
Unit 5

Image Segmentation

Image and object segmentation

- **Image Segmentation**
  - Group pixels into regions that share some similar properties

Superpixels
(Ren ICCV 2003



- **Segmenting images into meaningful objects**
  - Object-level segmentation: accurate localization and recognition

Object segmentation: applications

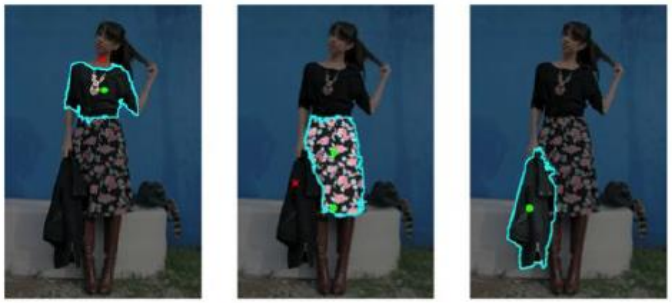
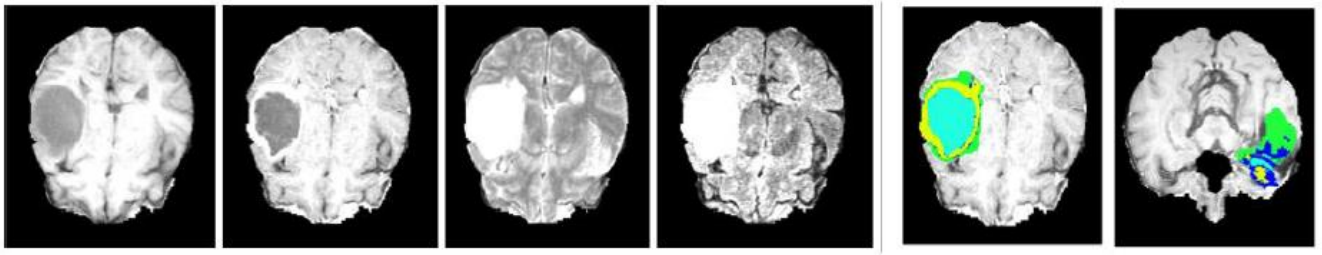


Image editing and composition (Xu, 2016)

Robotics

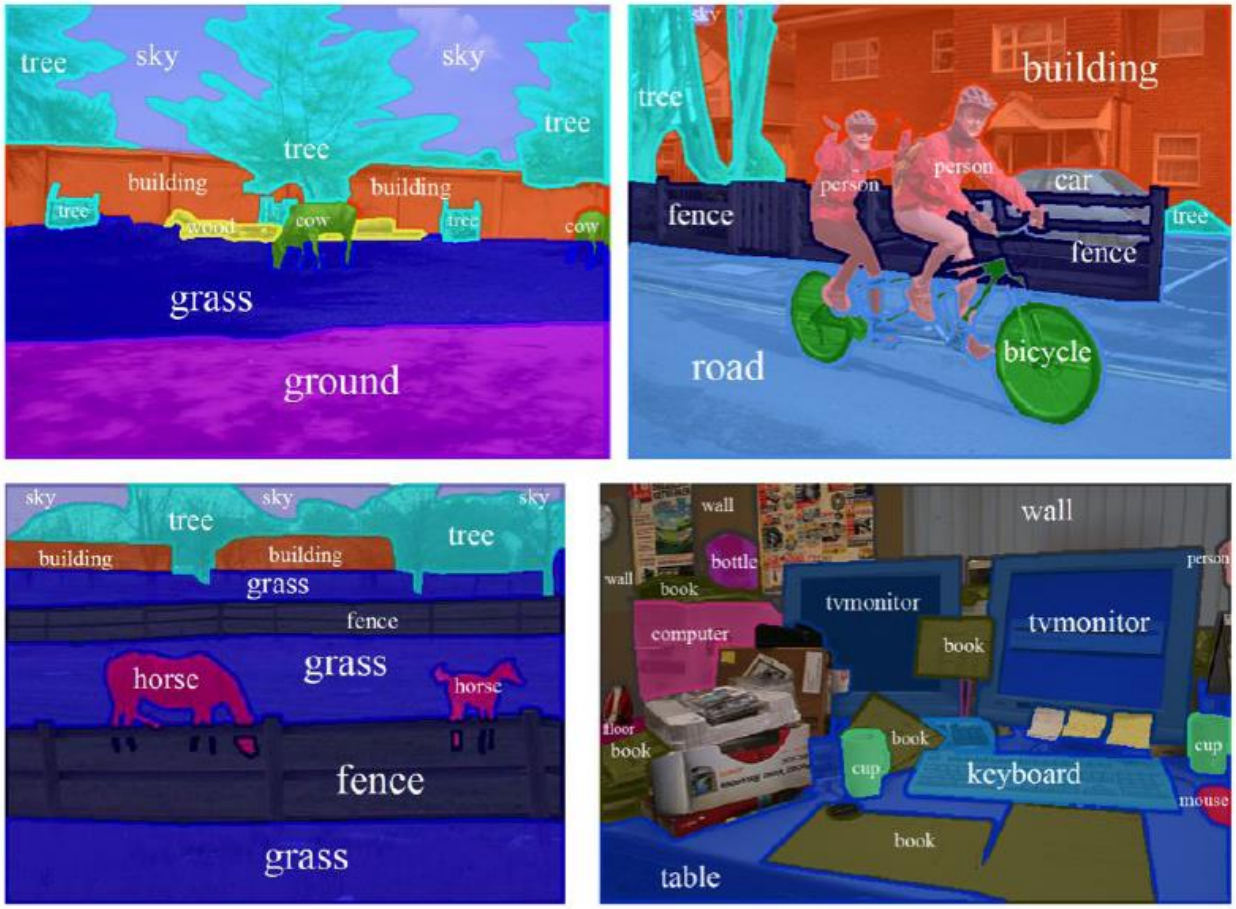
Autonomous driving
(cordts, 2016)

Medical image analysis
(Casamitjana, 2017)



Semantic segmentation

- Label every pixel: recognize the class of every pixel
- Do not differentiate instances

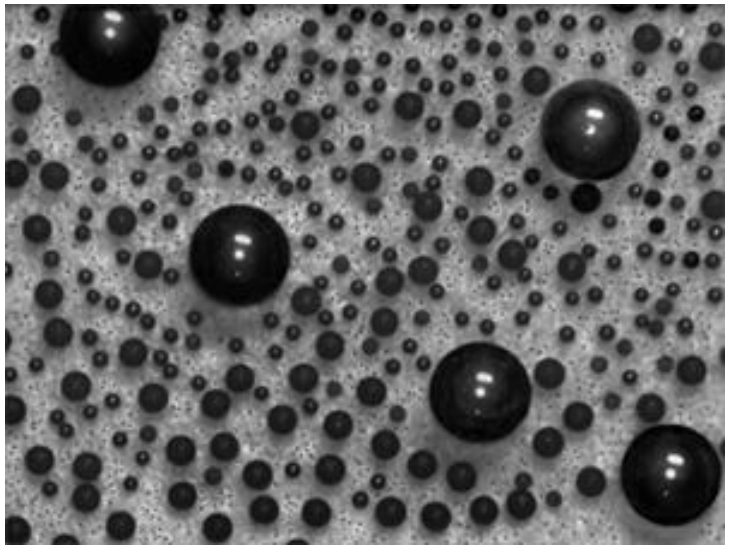


Instance segmentation

- Detect instances, categorize and label every pixel
- Labels are class-aware and instance-aware

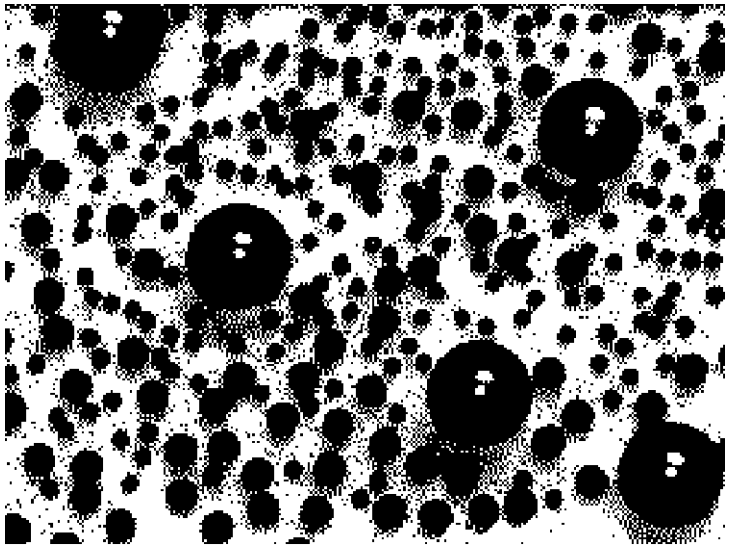
Histogram-based segmentation

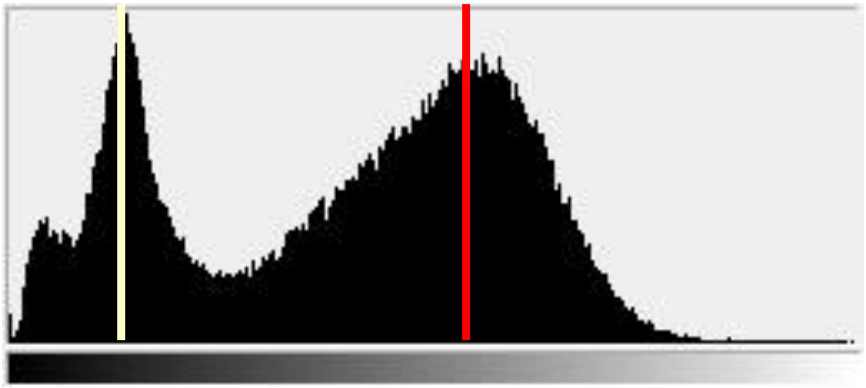
- Goal
  - Break the image into K regions (segments)
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color



Histogram-based segmentation

- Goal
  - Break the image into K regions (segments)
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color

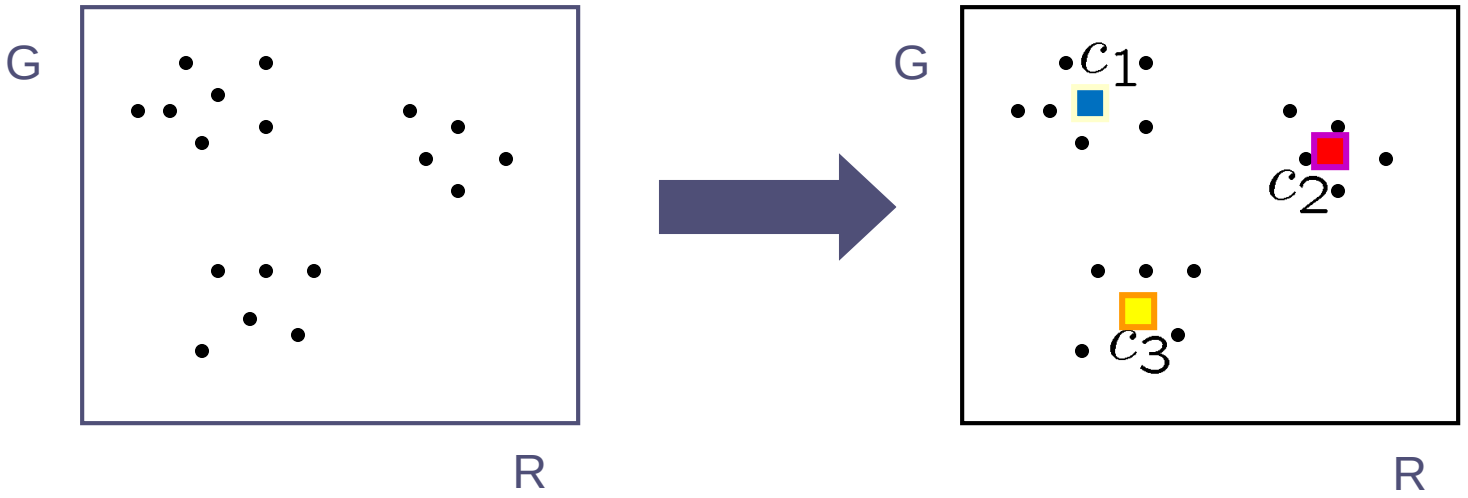




Here's what it looks like if we use two colors

Clustering

- How to choose the representative colors?
  - This is a clustering problem!



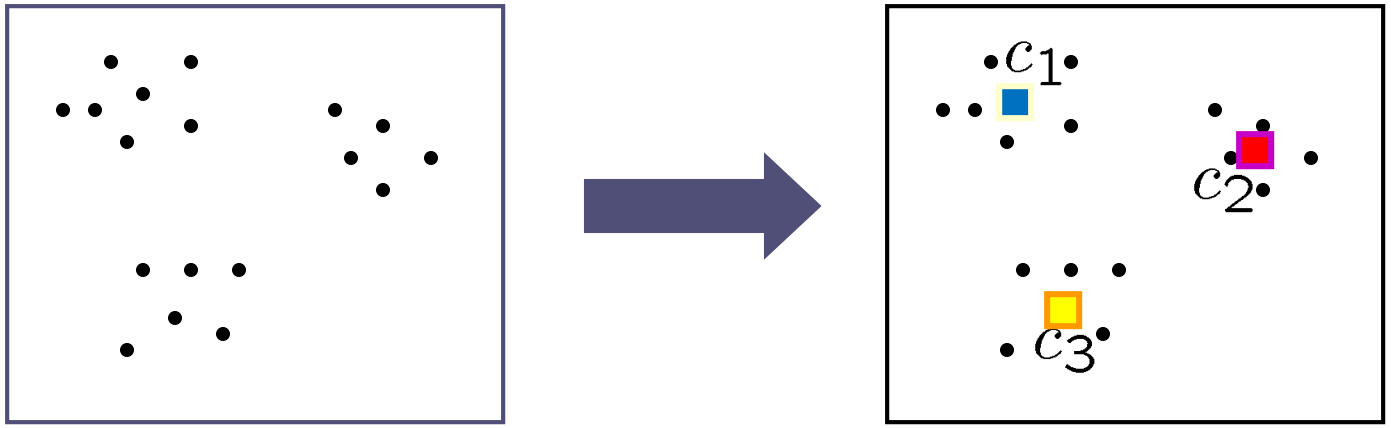
Objective

- Each point should be as close as possible to a cluster center
  - Minimize sum squared distance of each point to closest center

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

Break it down into subproblems

- Suppose I tell you the cluster centers c_i
  - Q: how to determine which points to associate with each c_i?
  - A: for each point p, choose closest c_i



Suppose I tell you the points in each cluster

- Q: how to determine the cluster centers?
- A: choose c_i to be the mean of all points in the cluster

K-means clustering

- K-means clustering algorithm
  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, update c_i to be the mean of all 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++

- Can we prevent arbitrarily bad local minima?

1. Randomly choose first center.
2. Pick new center with prob. proportional to: $\|p - c_i\|^2$
   (contribution of *p* to total error)
3. Repeat until *k* centers.

Probabilistic clustering

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

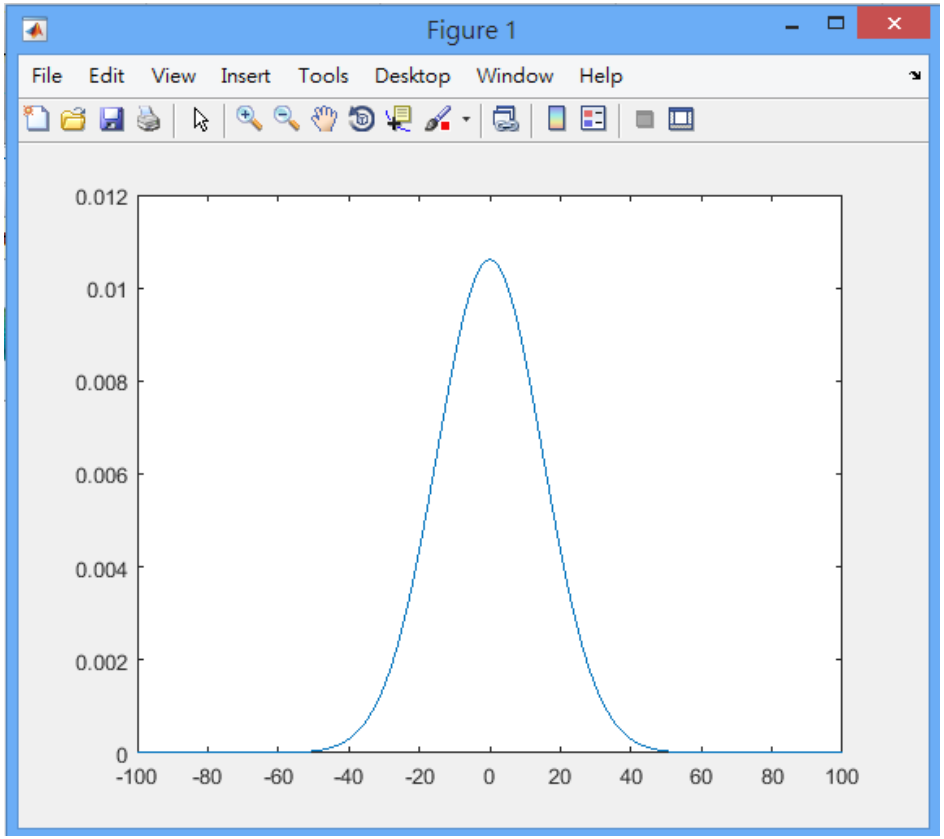
- Basic idea
  - instead of treating the data as a bunch of points, assume that they are all generated by sampling a continuous function
  - This function is called a **generative model**
    - defined by a vector of parameters θ

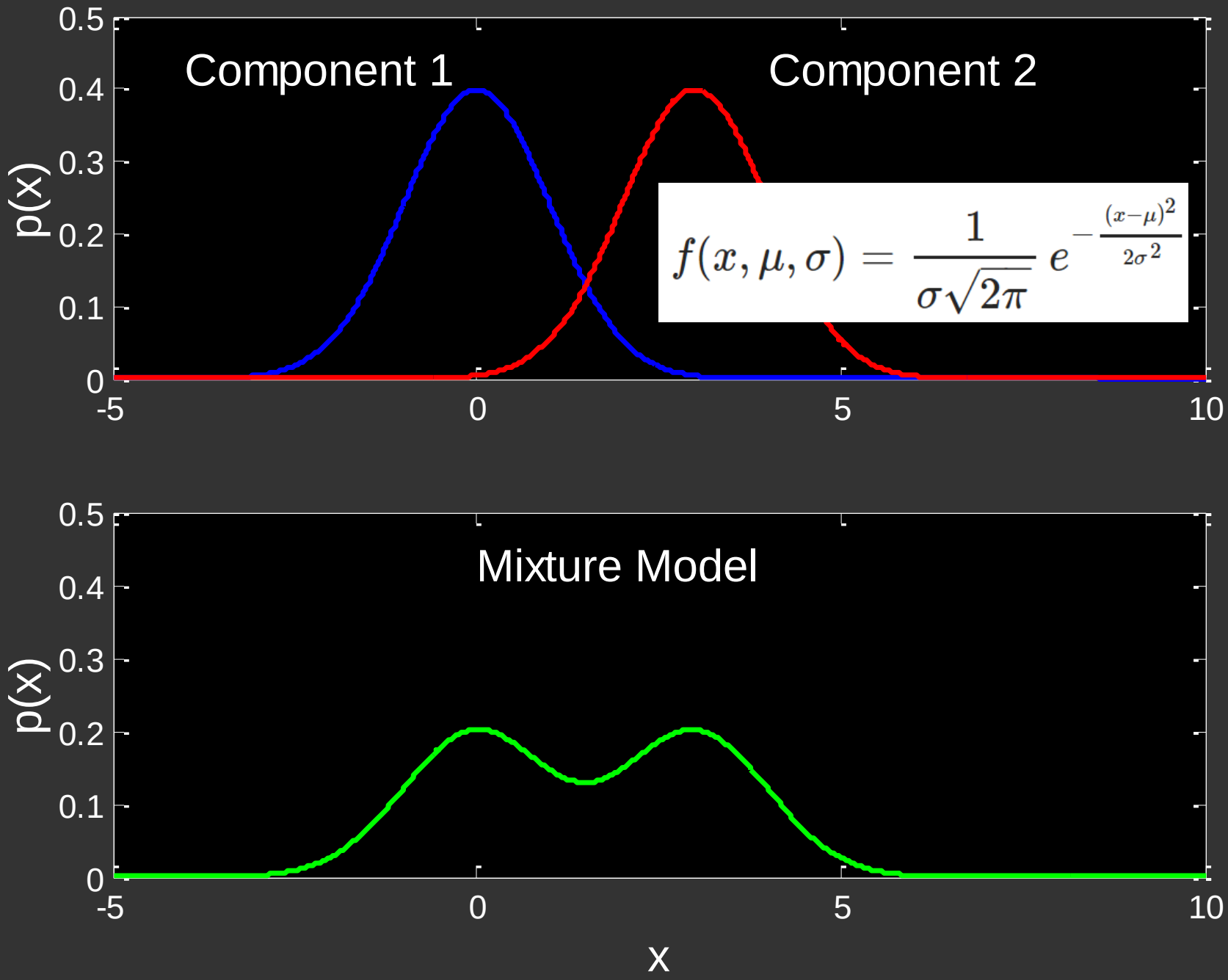
Generative Model

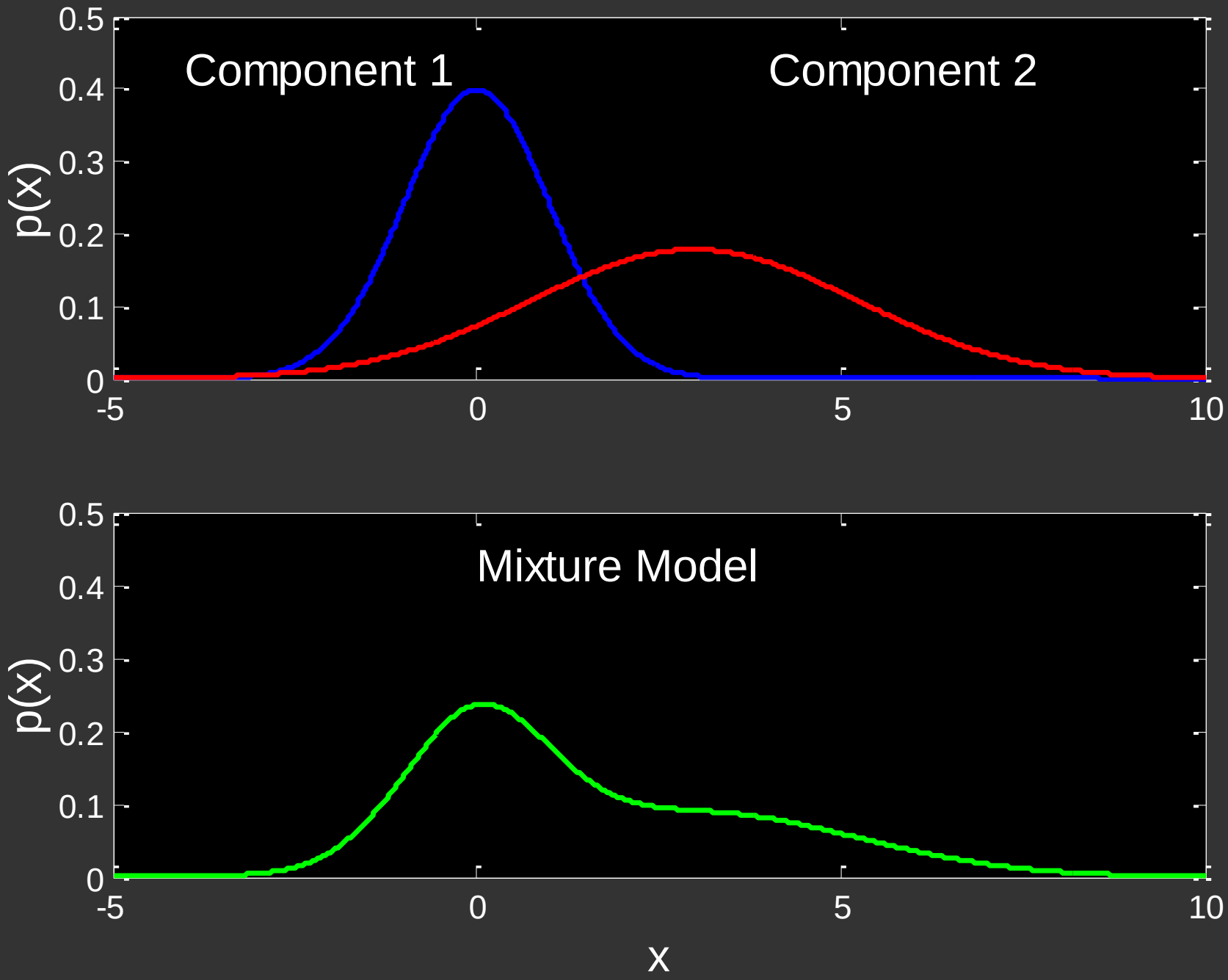
$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

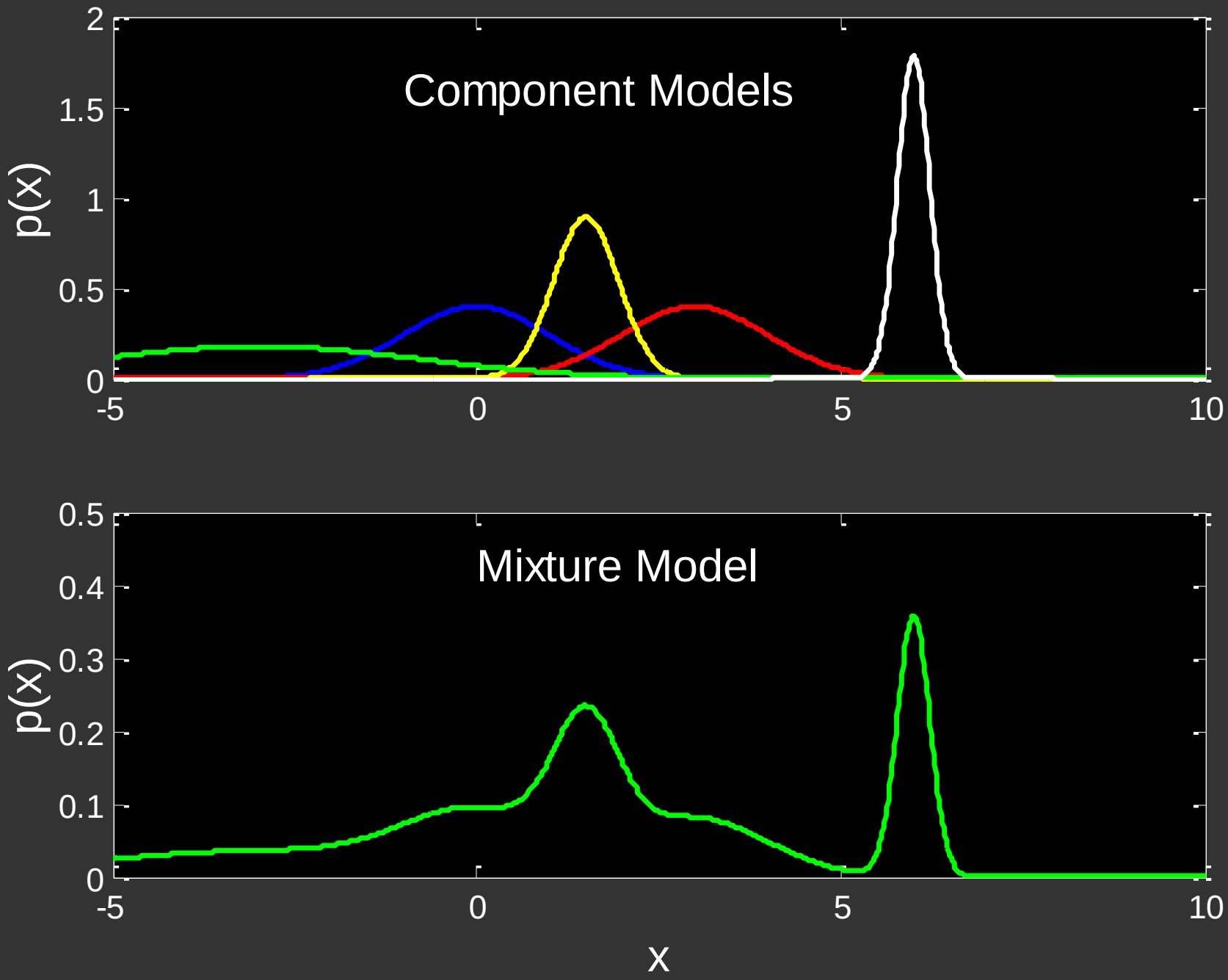
- x=-100:0.5:100
- y=(1/(15*2*pi))*exp(-(x.*x)/(2*15*15))
- plot(x, y)

- If you have the mean and variance, you have the Gaussian distribution.
- Given the distribution, you can use any x to calculate its probability y.




Component 1
Component 2
$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
Mixture Model
p(x)
p(x)
0.5
0.4
0.3
0.2
0.1
0
-5
0
5
10
x


Component 1
Component 2
p(x)
0.5
0.4
0.3
0.2
0.1
0
-5
0
5
10
Mixture Model
p(x)
0.5
0.4
0.3
0.2
0.1
0
-5
0
5
10
x


Component Models
p(x)
2
1.5
1
0.5
0
-5
0
5
10
Mixture Model
p(x)
0.5
0.4
0.3
0.2
0.1
0
-5
0
5
10
x

Finite Mixture Models

- Assume we have K Gaussian distribution. Each can be treated as a component.
- The density function of GMM is considered as the linear combination of all component:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x} \mid k) p(k)$$

$$= \sum_{k=1}^{K} \alpha_k \, p(\mathbf{x} \mid \mu_k, \Sigma_k)$$

$$= \sum_{k=1}^{K} \alpha_k \, N(\mathbf{x} \mid \mu_k, \Sigma_k)$$

Each mixture component is a multidimensional Gaussian with its own mean μ_k and covariance Σ_k ,e.g., K=2:

We have parameter set $\{\theta, \alpha\} = \{\mu_1 , \sigma_1 , \mu_2 , \sigma_2 , \alpha_1 , \alpha_2\}$

Variance

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$$

- Variance only operates on 1 dimension
  - So each dimension is calculated independently of the other dimensions.
  - However, it is useful to have a similar measure to find out how much the dimensions vary ***with respect to each other***.

Covariance

- Covariance is measured between 2 dimensions
  - calculate the covariance between one dimension and itself, you get the variance.

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

  - if you had a 3-dimensional data set *(x, y, z)*, then you could measure the covariance between x and y, y and z, x and z.

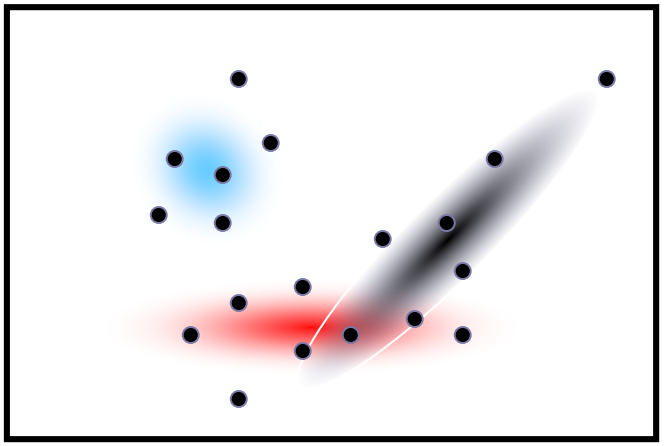
$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

The Covariance Matrix

- Recall that covariance is always measured between 2 dimensions.
- If we have a data set with 3 dimensions, we can calculate cov(x, y), cov(x, z), cov(y, z)
- The covariance matrix is formed like this:

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

Mixture of Gaussians



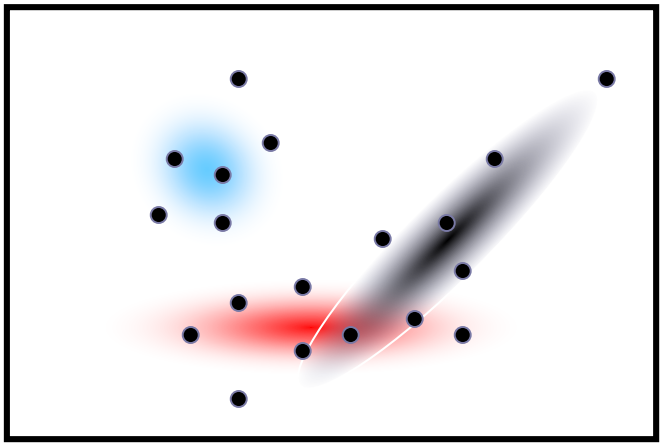
- One generative model is a mixture of Gaussians (MOG)
  - K Gaussian blobs with means $\boldsymbol{\mu_b}$ covariance matrices $\mathbf{V_b}$, dimension d
    - blob *b* defined by $P(x|\mu_b, V_b) = \frac{1}{\sqrt{(2\pi)^d |V_b|}} e^{-\frac{1}{2}(x-\mu_b)^T V_b^{-1}(x-\mu_b)}$
  - blob *b* is selected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

$$P(x|\theta) = \sum_{b=1}^{K} \alpha_b P(x|\theta_b)$$

  - where

$$\theta = [\mu_1, \ldots, \mu_n, V_1, \ldots, V_n]$$

Expectation maximization (EM)



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

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

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

EM details

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

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

- M-step
  - If we know the probability that point x is in distribution b, the probability can be considered to be the contribution of distribution *b* that generates x. In other words, component *b* generates point

$$P(b|x_1,\mu_b,V_b)x_1,\ P(b|x_2,\mu_b,V_b)x_2,\ \ldots\ldots,\ P(b|x_N,\mu_b,V_b)x_N,$$

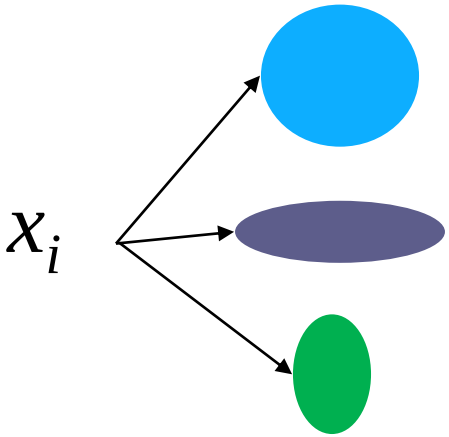
  - The mean of blob b can be expressed as:

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

Example

$\alpha_b P(x|\mu_b, V_b)$

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

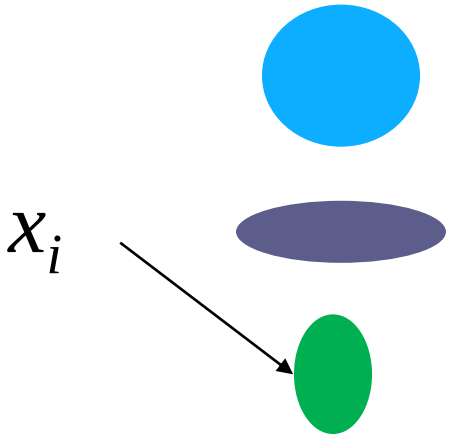


0.5 0.5 / (0.5+0.2+0.8) = 0.33

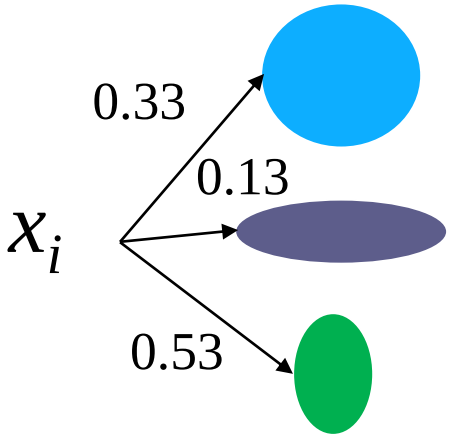
0.2 0.2 / (0.5+0.2+0.8) = 0.13

0.8 0.8 / (0.5+0.2+0.8) = 0.53

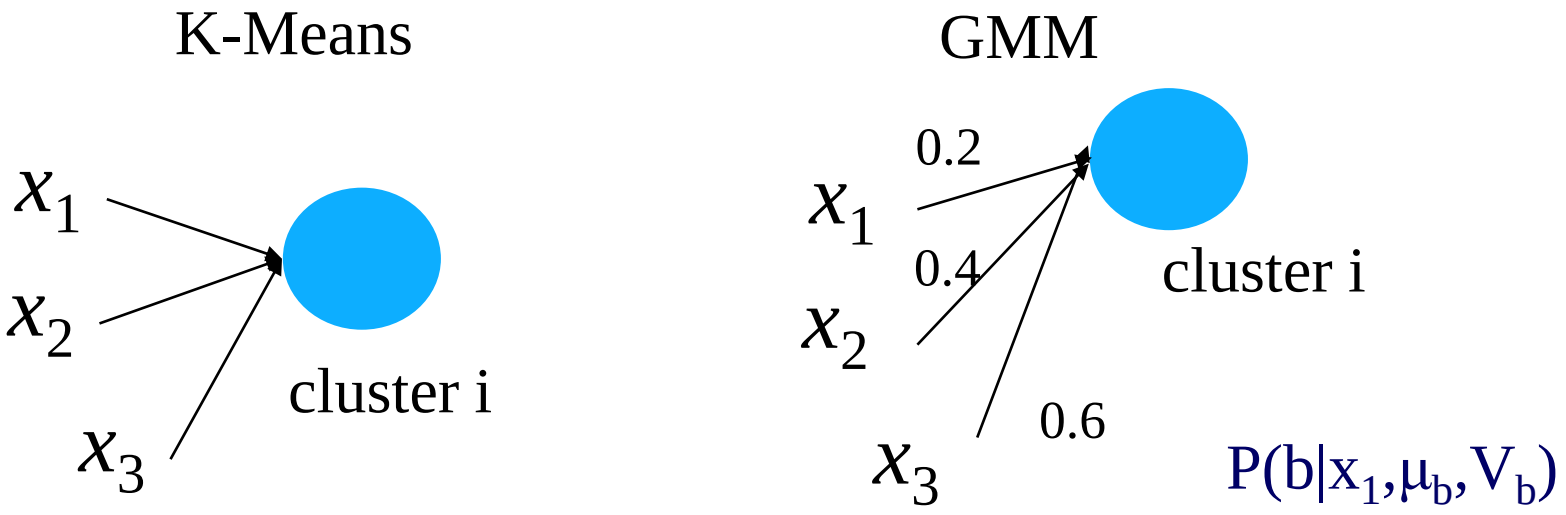
K-Means



GMM



Example



K-means Mean = (x_1 *1 + x_2 * 1 + x_3 * 1) / (1+1+1)

GMM Mean = (x_1 *0.2 + x_2 * 0.4 + x_3 * 0.6) / (0.2+0.4+0.6)

$P(b|x_1,\mu_b,V_b)x_1$, $P(b|x_2,\mu_b,V_b)x_2$, ……, $P(b|x_N,\mu_b,V_b)x_N$,

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

EM details

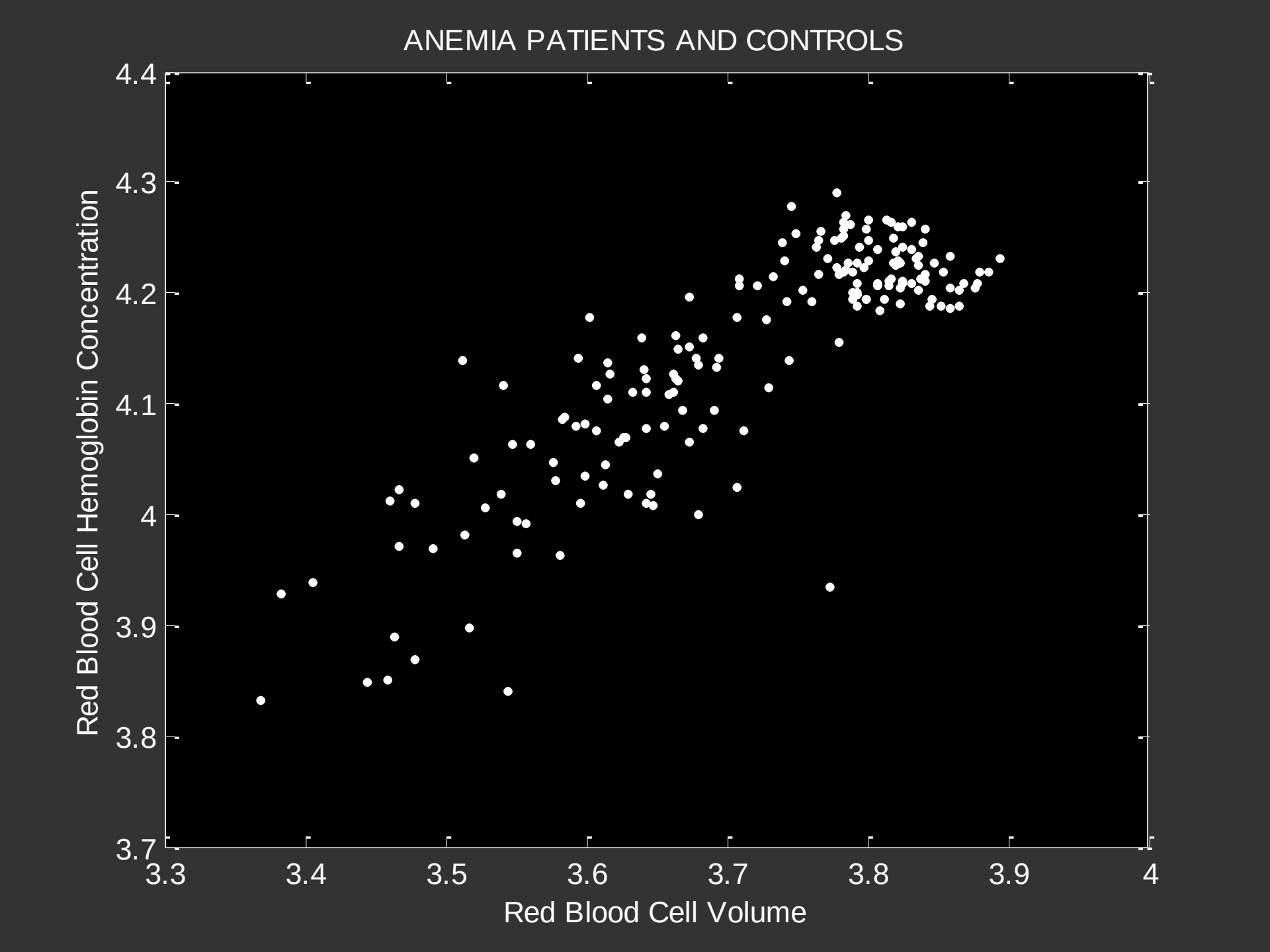
- M-step
  - variance of blob b

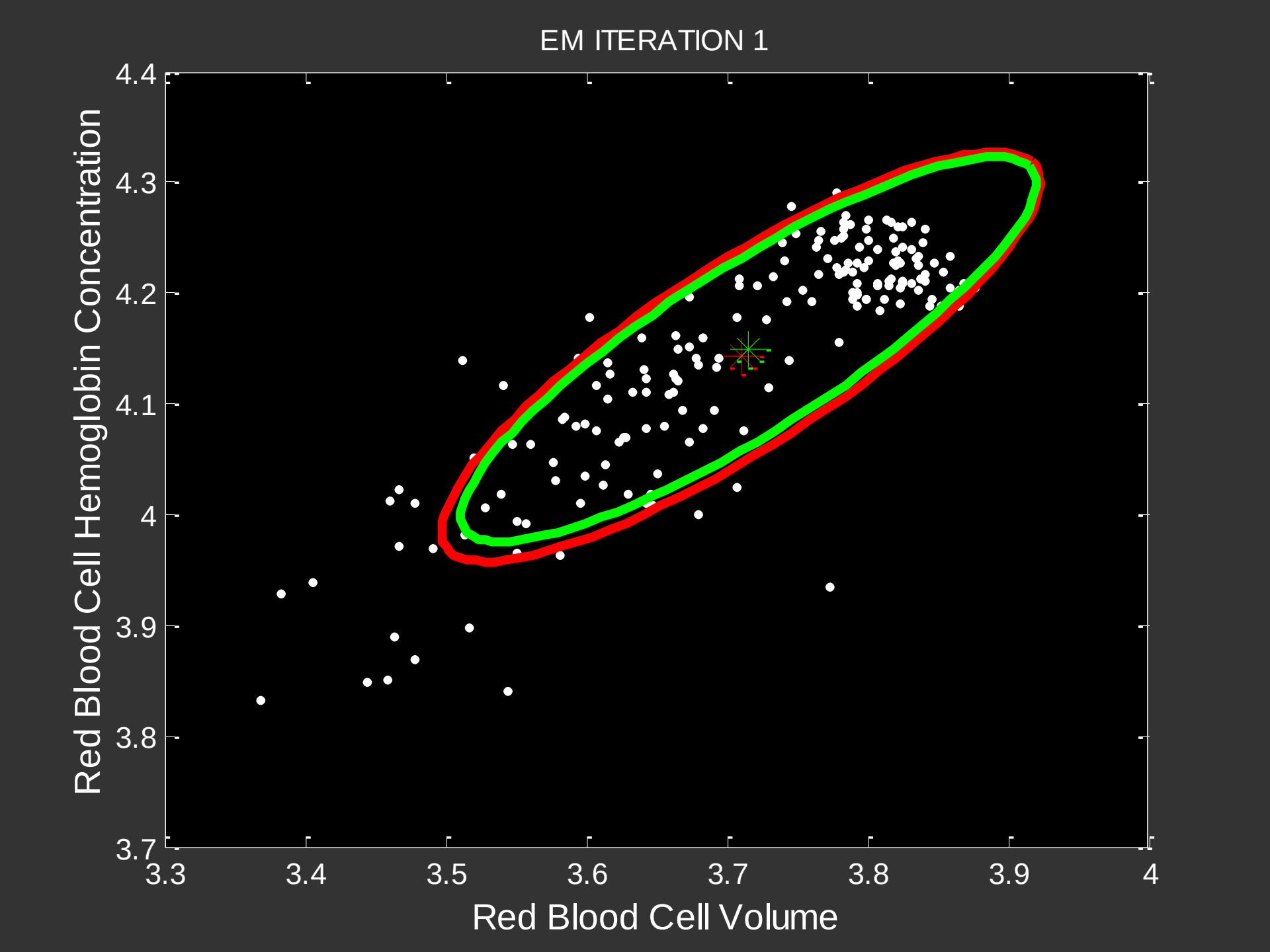
$$V_b^{new} = \frac{\sum_{i=1}^{N}(x_i - \mu_b^{new})(x_i - \mu_b^{new})^T P(b|x_i, \mu_b, V_b)}{\sum_{i=1}^{N} P(b|x_i, \mu_b, V_b)}$$

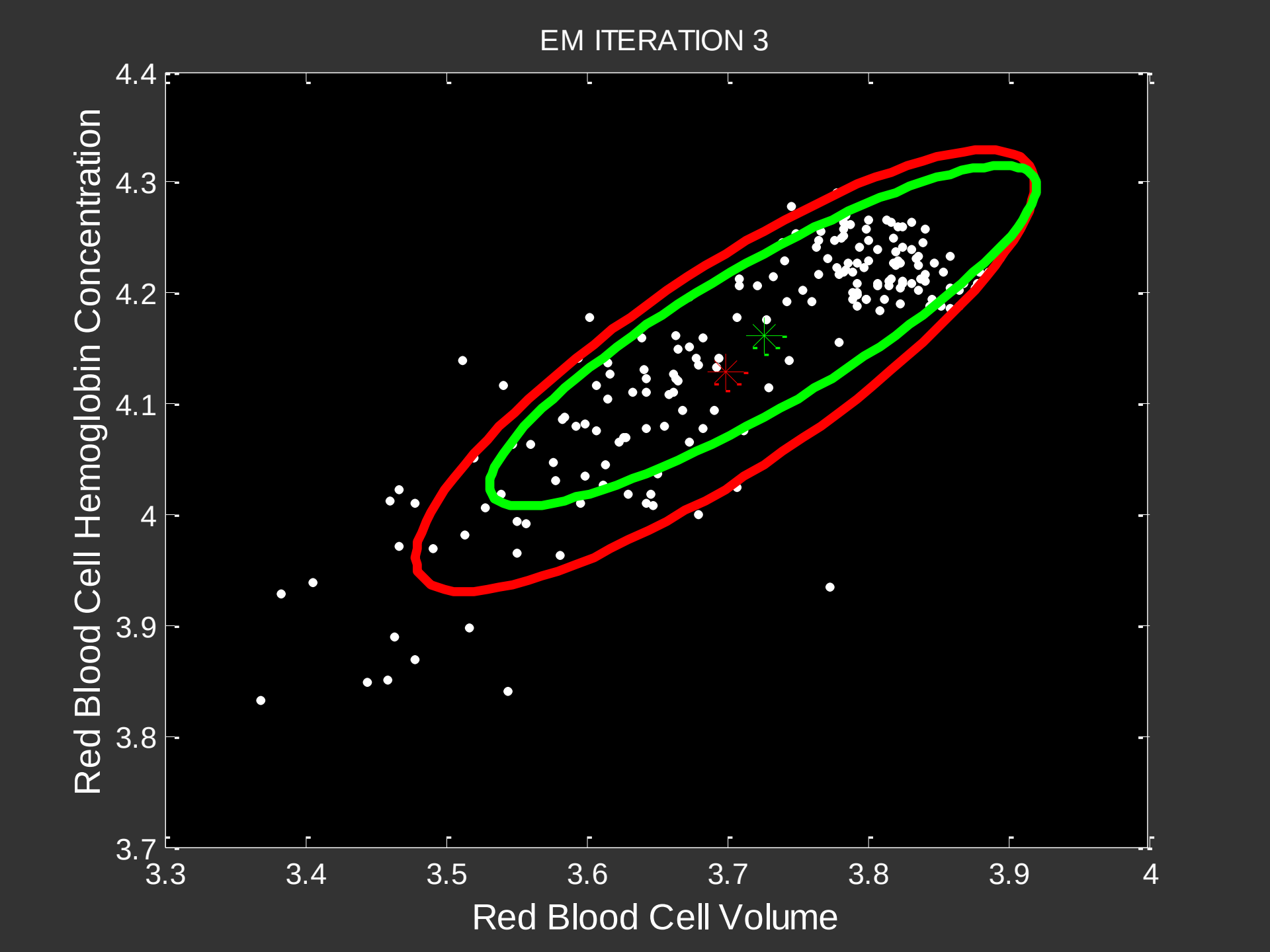
  - compute probability that blob b is selected

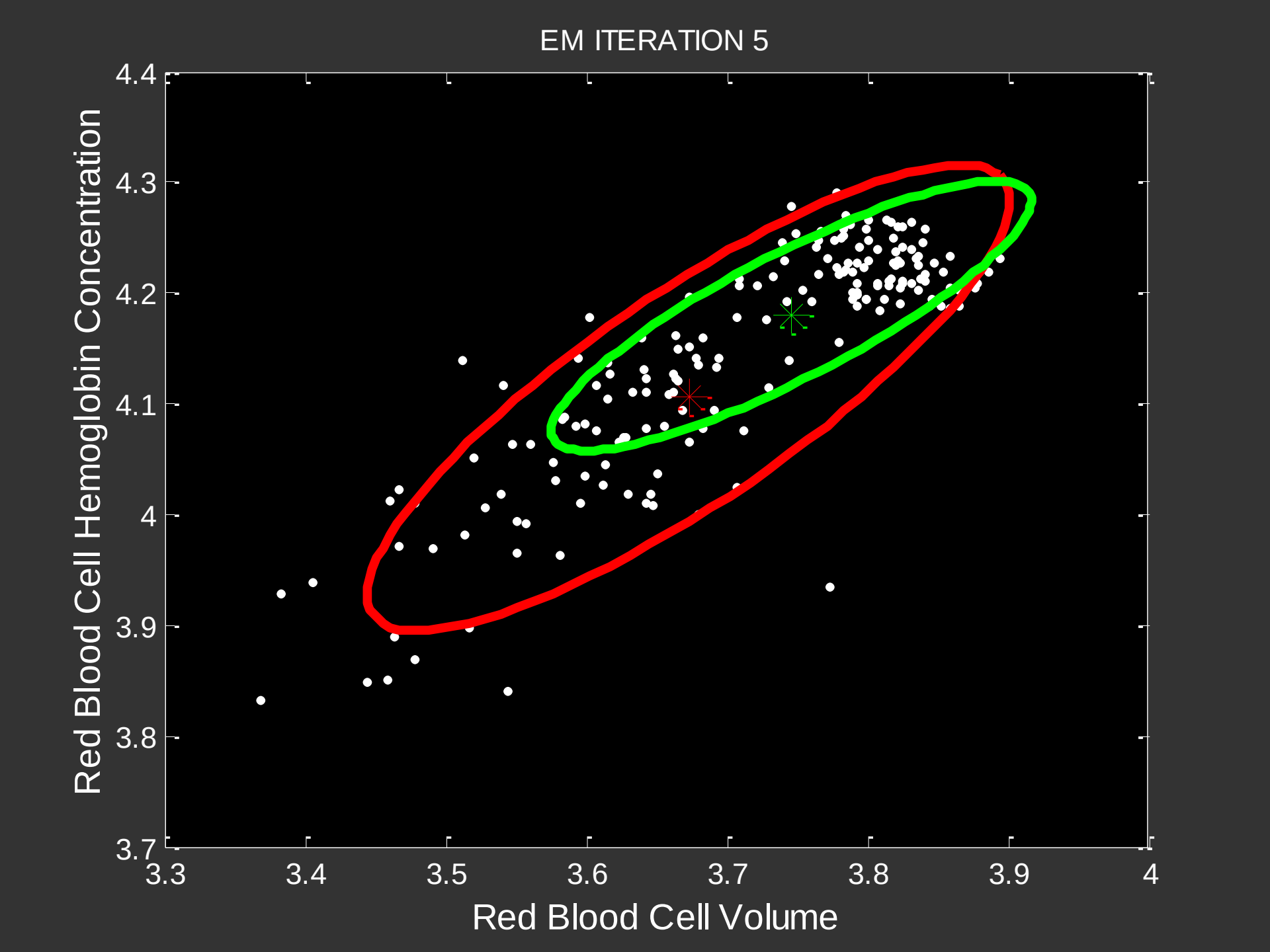
$$\alpha_b^{new} = \frac{1}{N}\sum_{i=1}^{N} P(b|x_i, \mu_b, V_b)$$

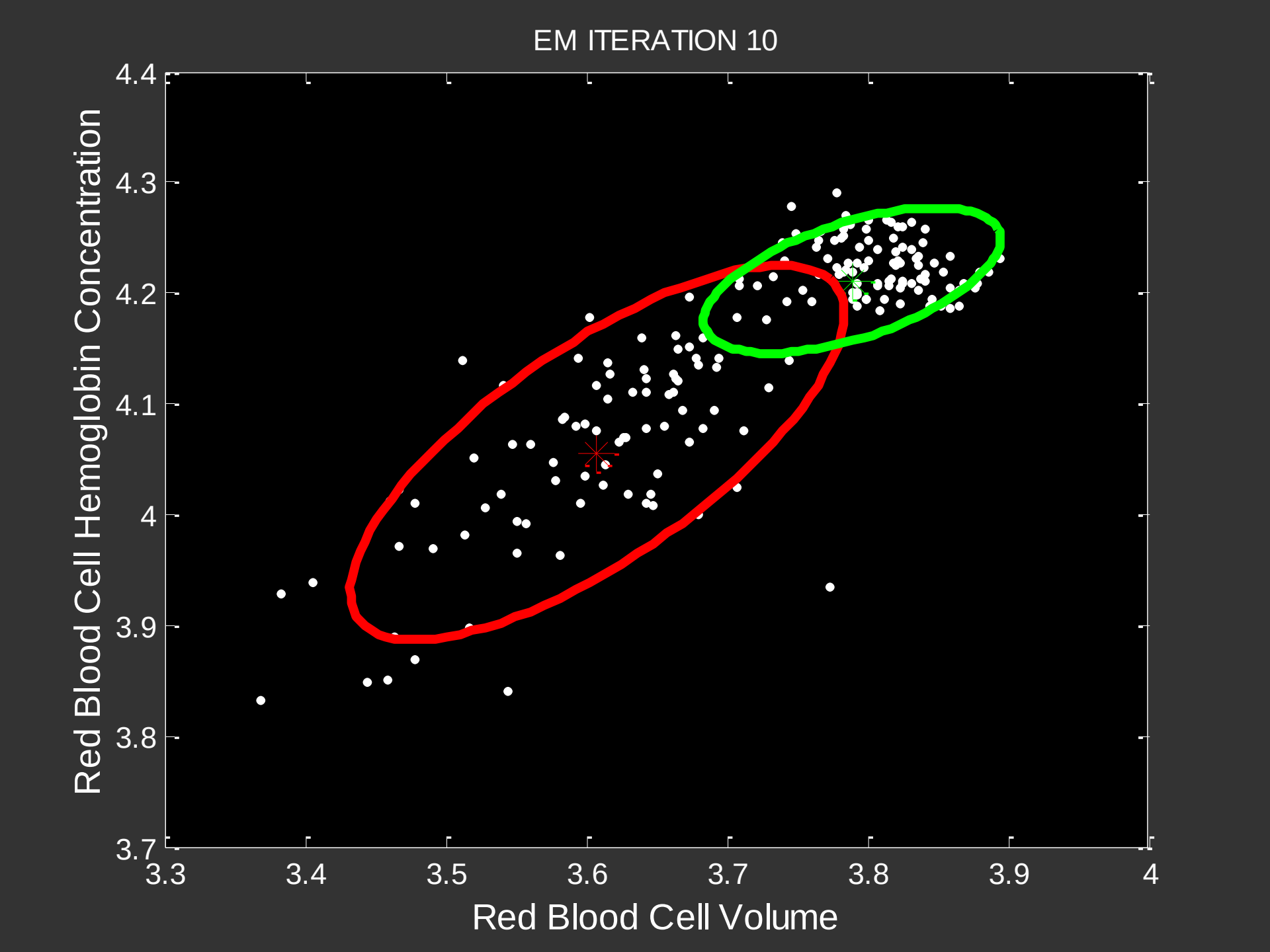
- For clustering, each component *k* in GMM is a cluster.
  - Calculate the probability of x_i to each component *k*, then we know which cluster it belongs to.

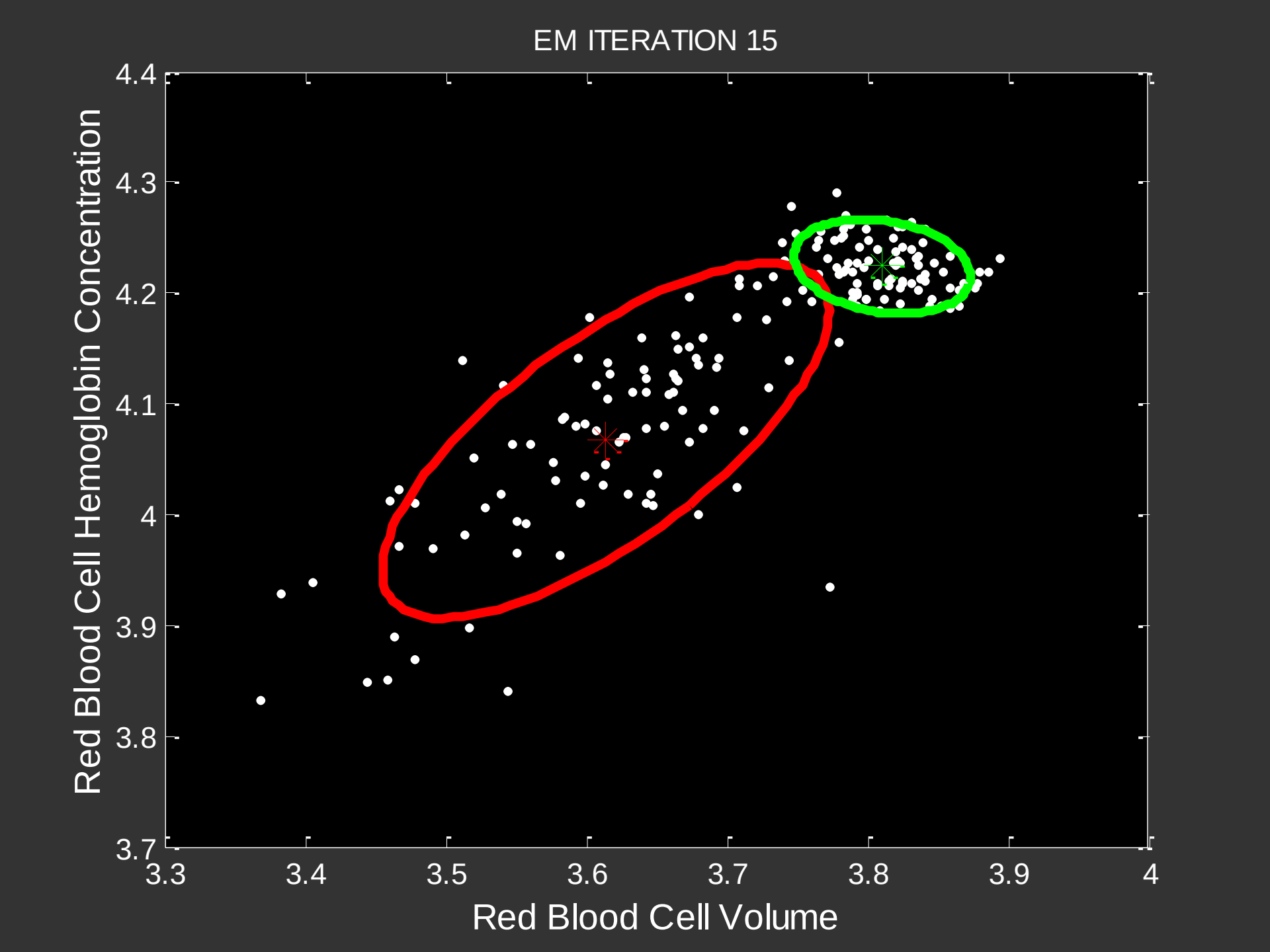

ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

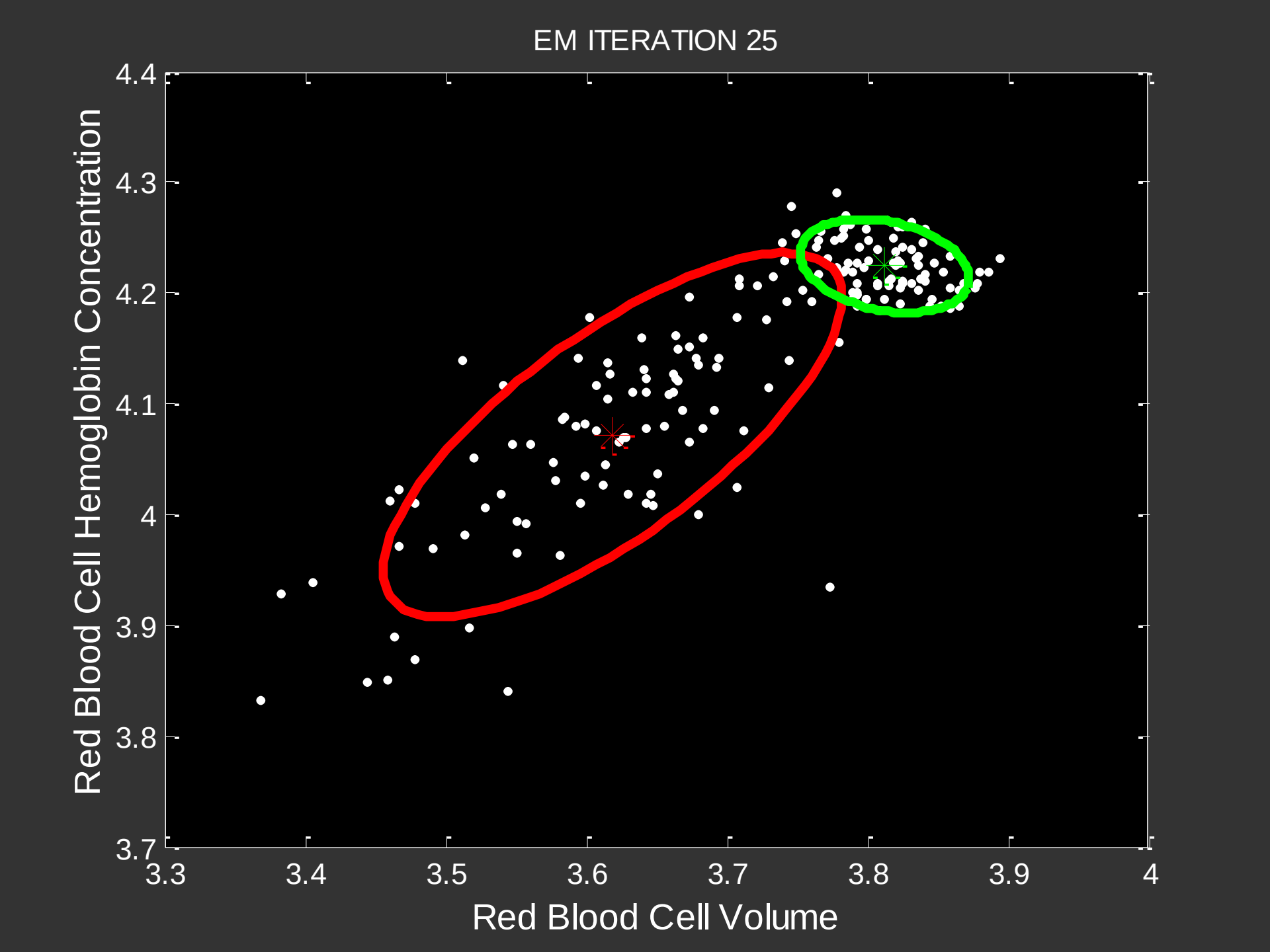

EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

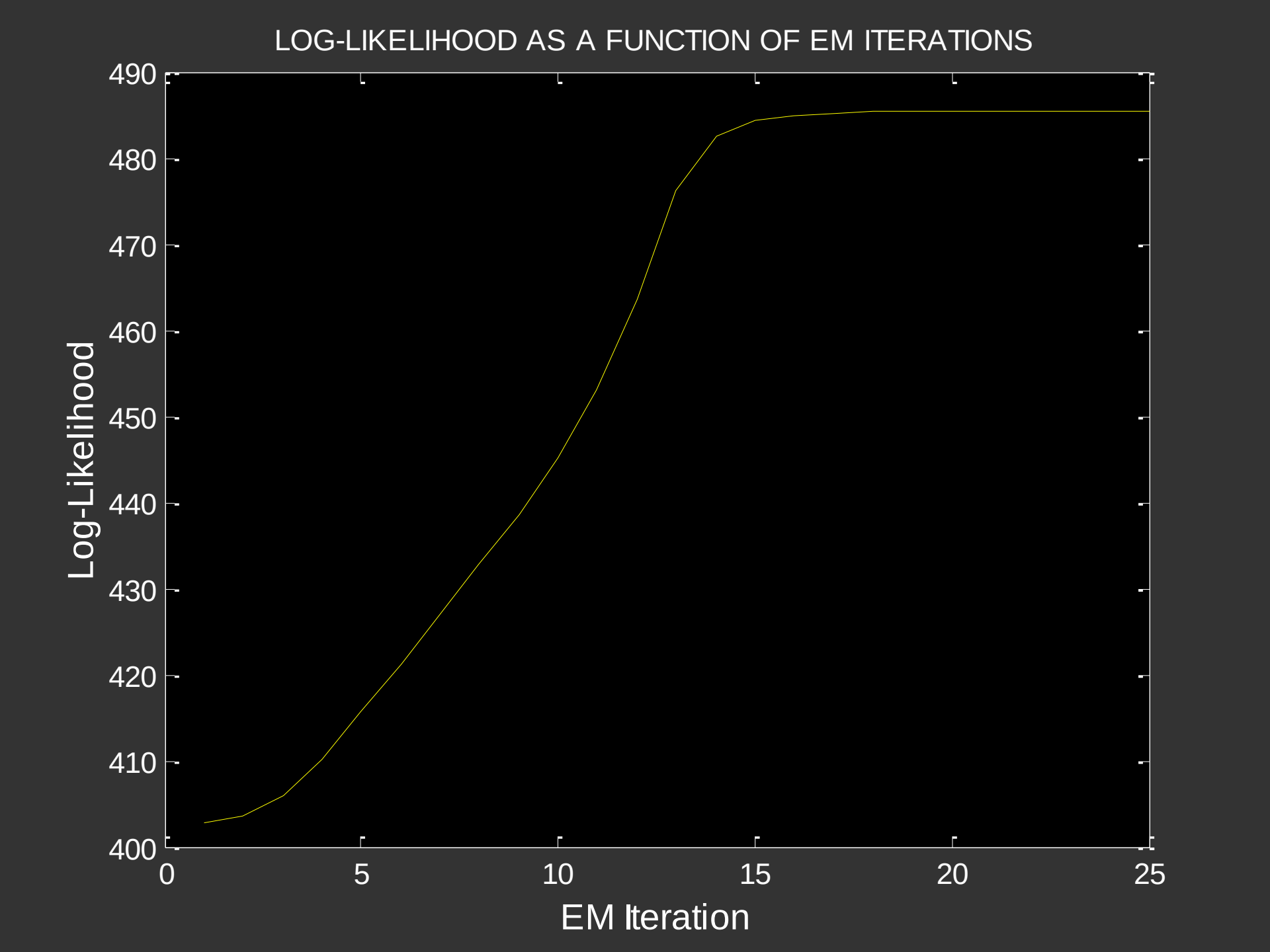

EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

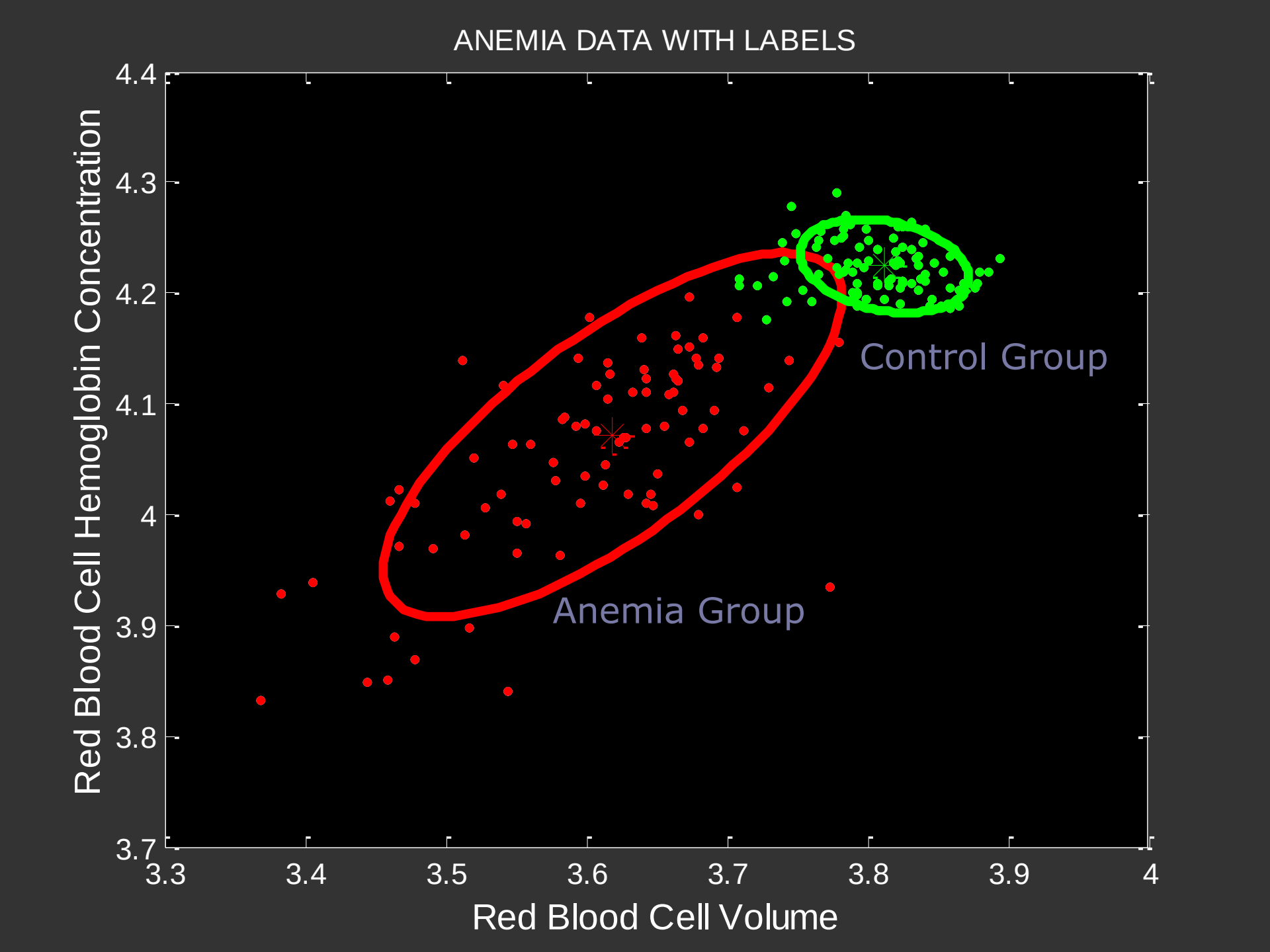

EM ITERATION 5
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 10
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 15
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4


EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4

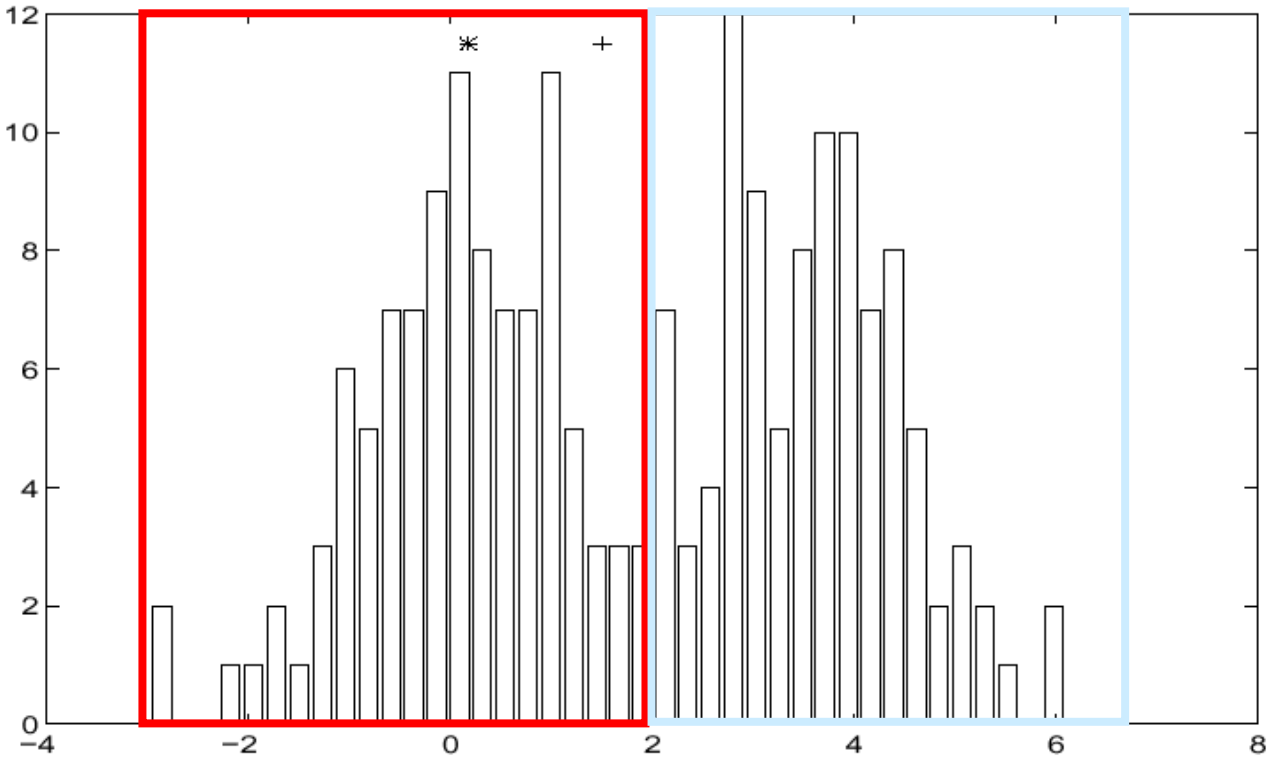

LOG-LIKELIHOOD AS A FUNCTION OF EM ITERATIONS
Log-Likelihood
EM Iteration
490
480
470
460
450
440
430
420
410
400
0
5
10
15
20
25


ANEMIA DATA WITH LABELS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
Control Group
Anemia Group

Applications of EM

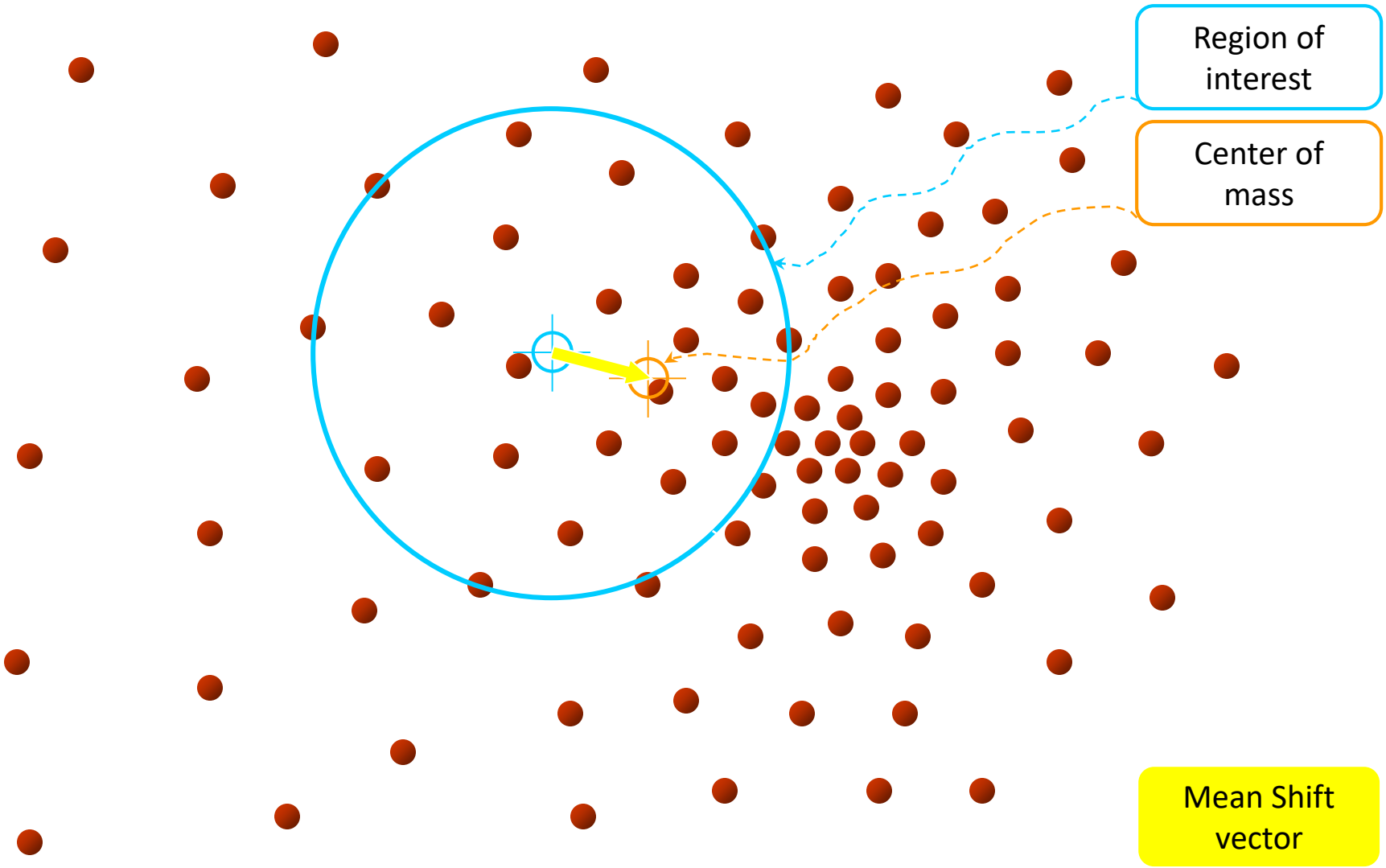
- Turns out this is useful for all sorts of problems
  - any clustering problem
  - any model estimation problem
  - missing data problems
  - finding outliers
  - segmentation problems
    - segmentation based on color
    - segmentation based on motion
    - foreground/background separation
  - ...

Finding Modes in a Histogram



- How Many Modes Are There?
  - Easy to see, hard to compute

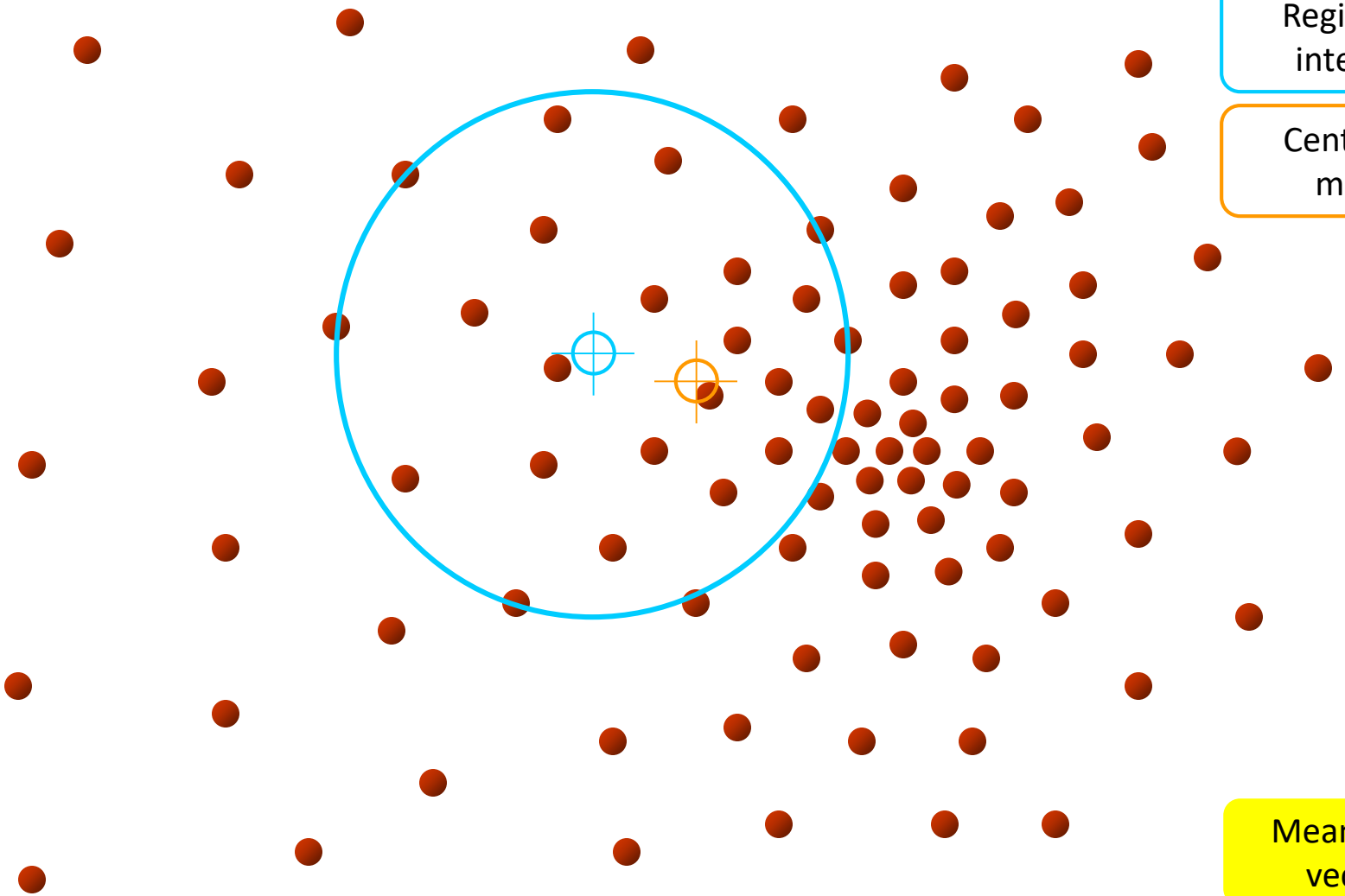
Mean Shift Theory



Objective : Find the densest region

Distribution of identical billiard balls

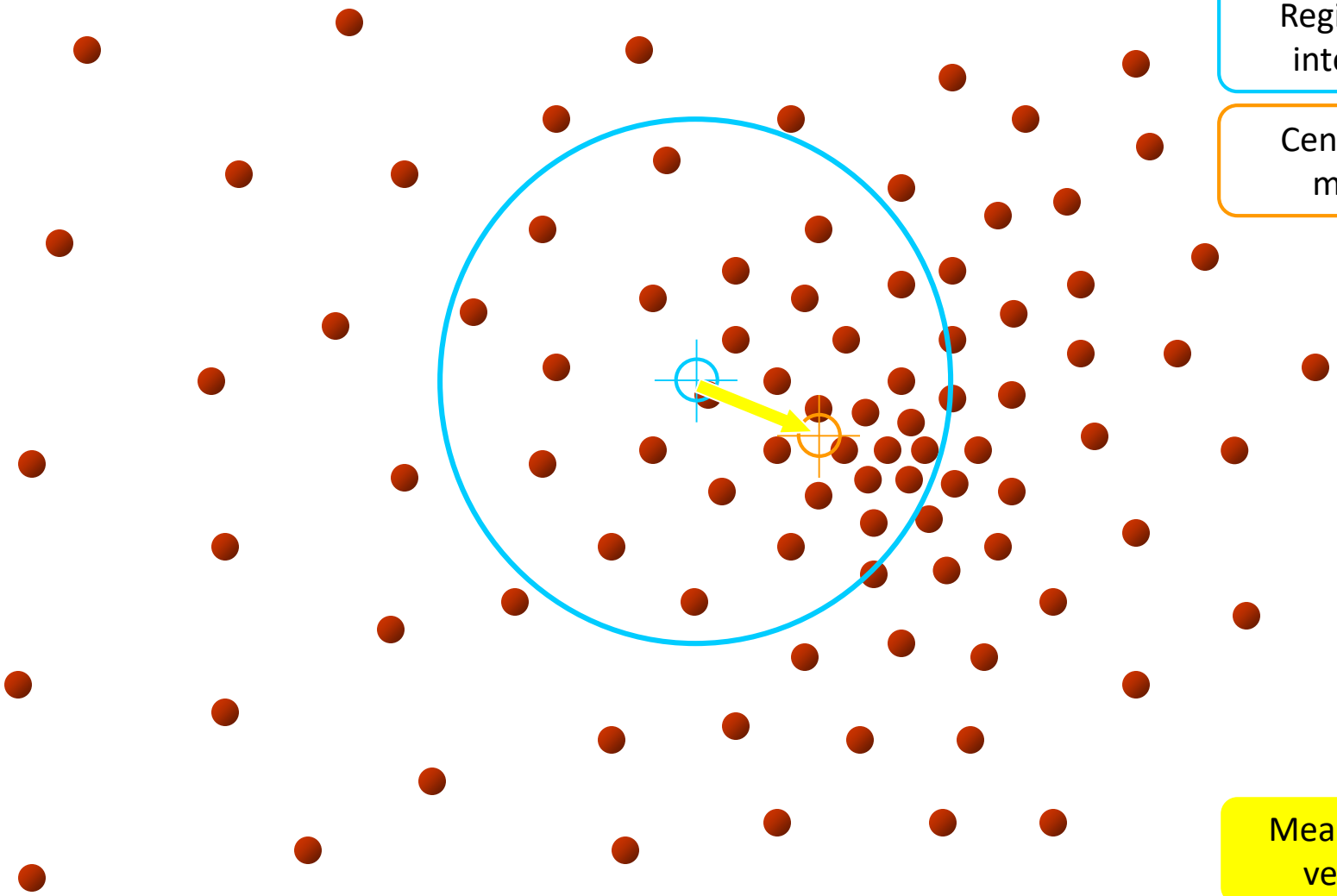
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

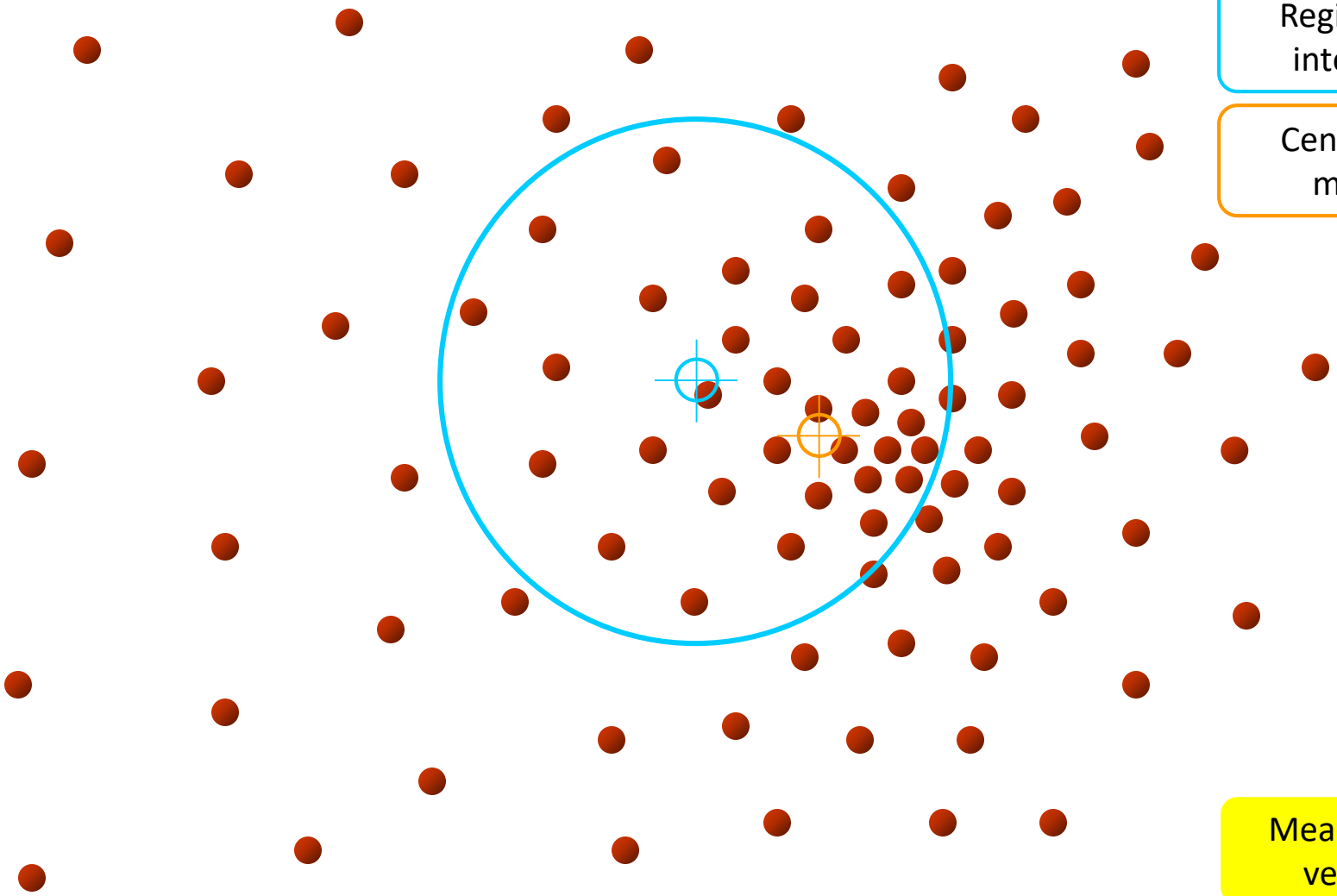
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

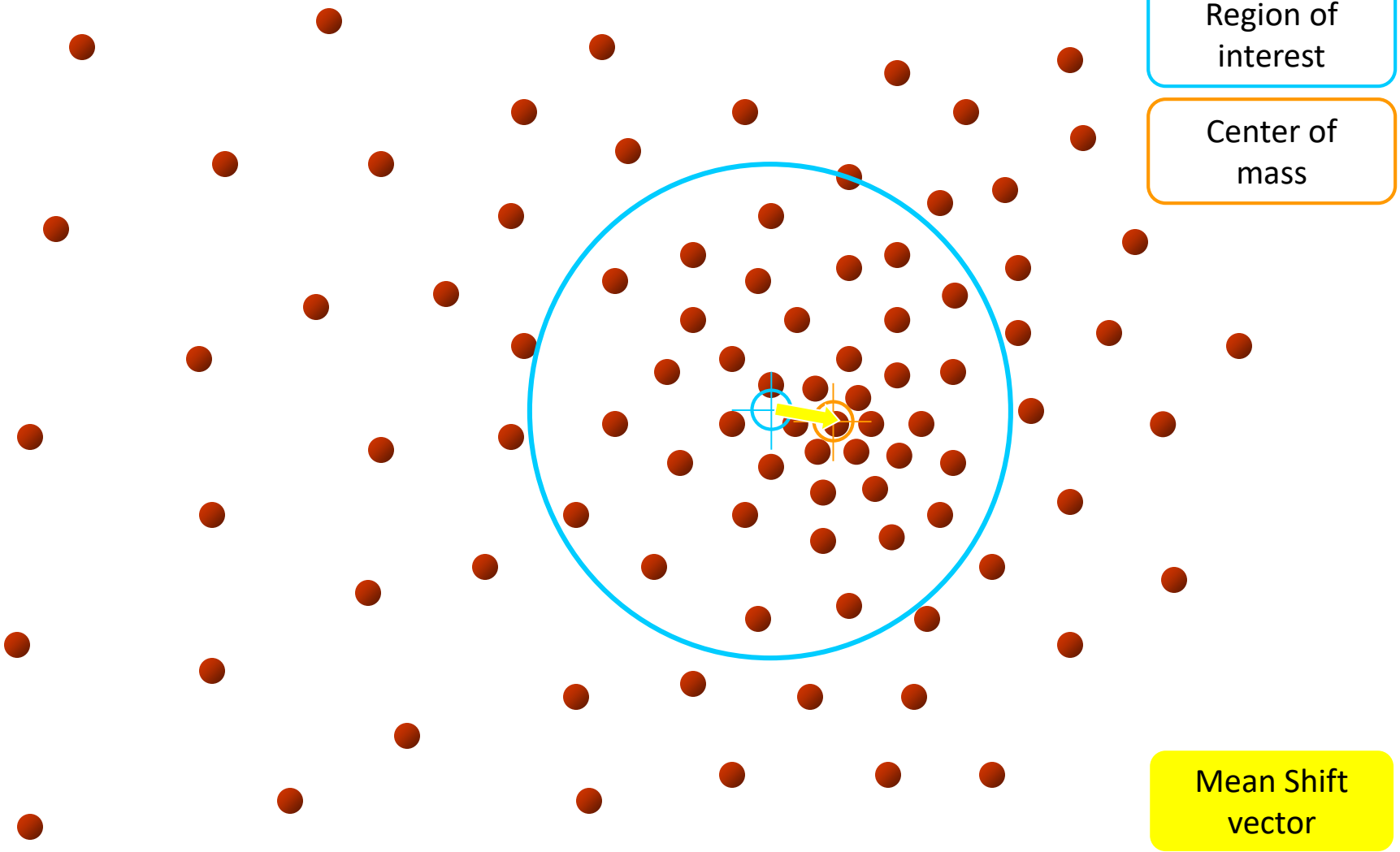
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

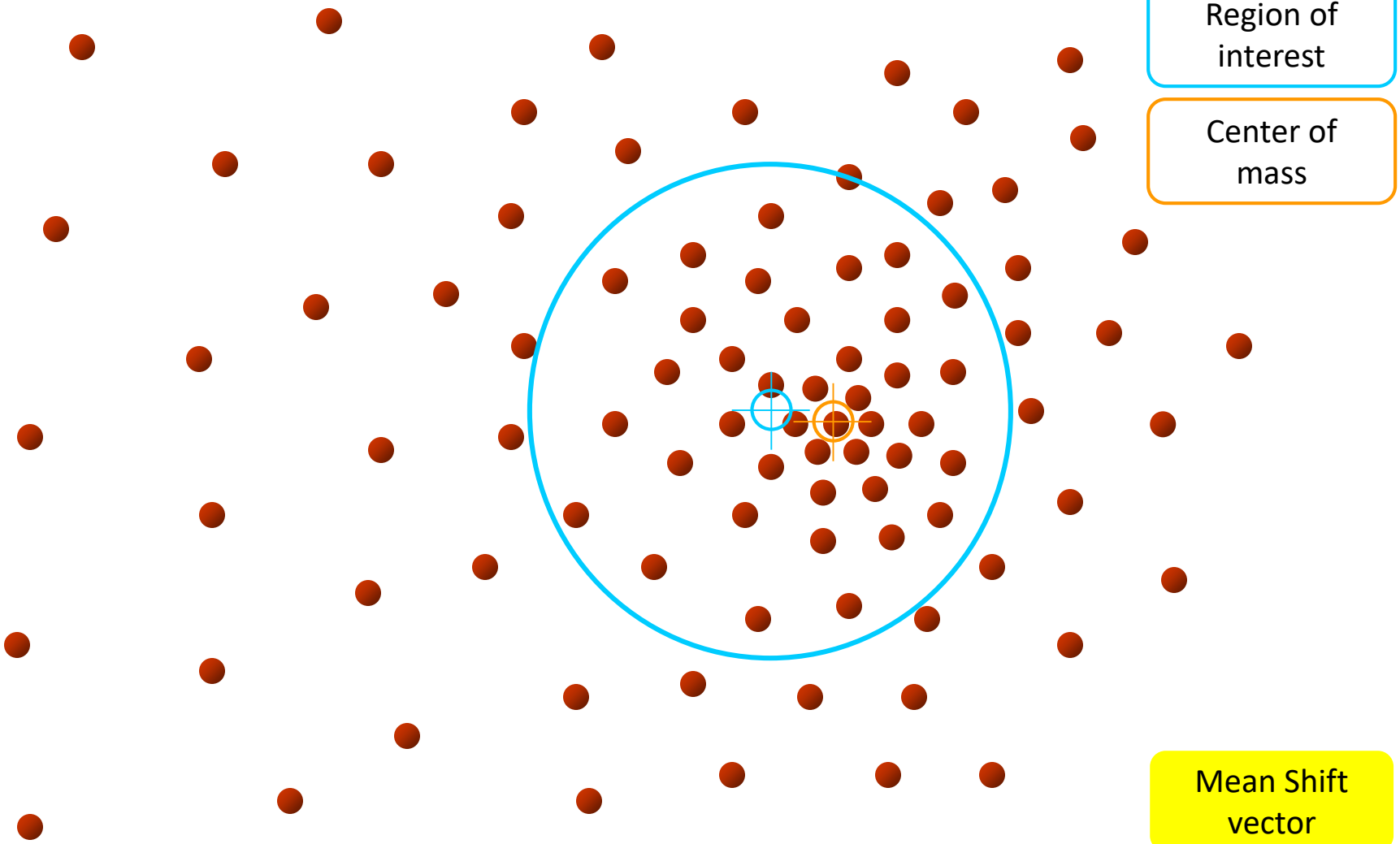
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

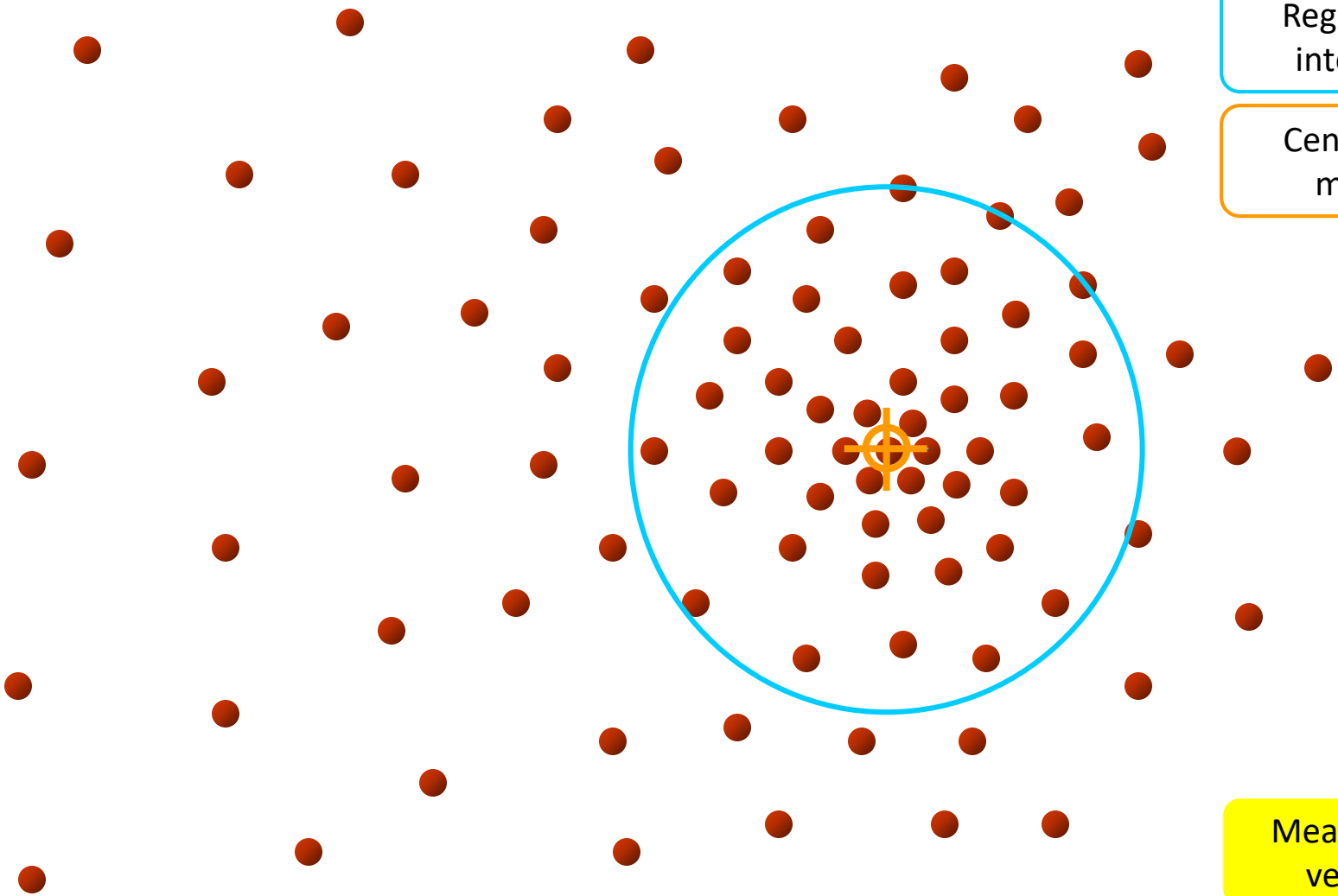
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

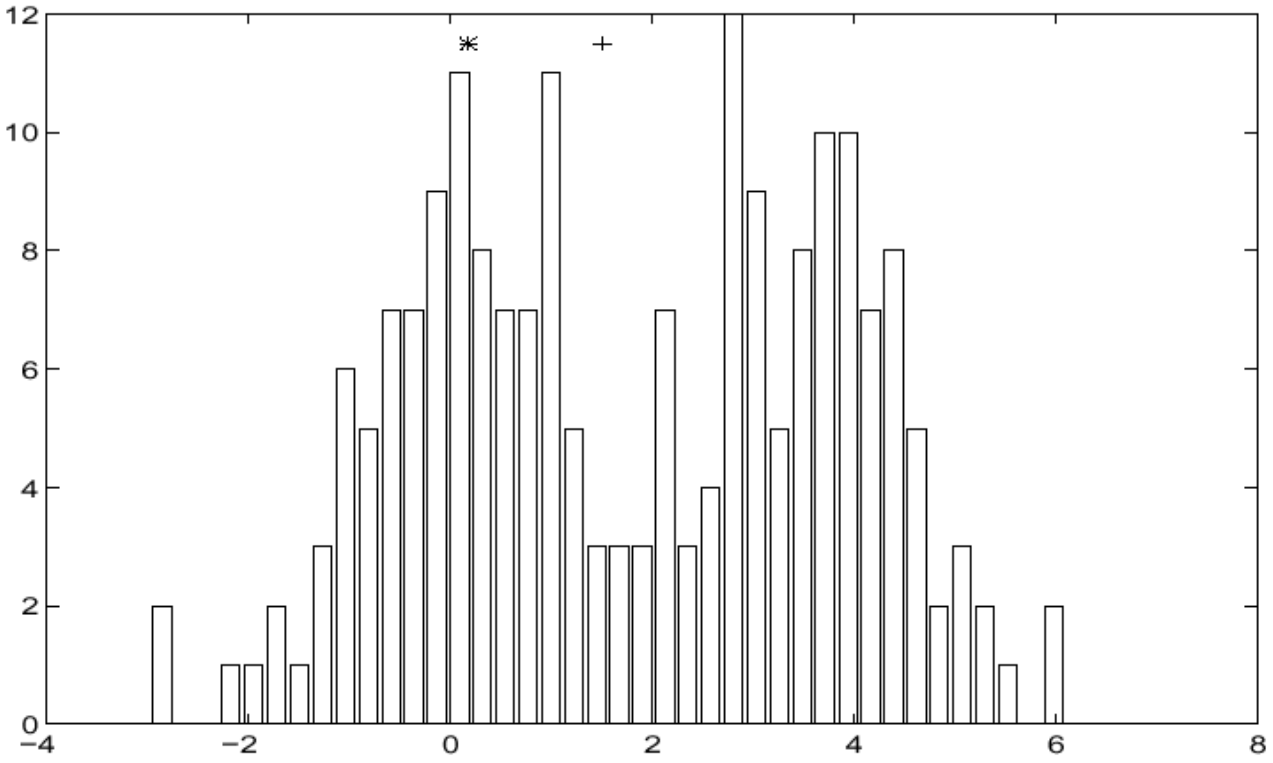
Intuitive Description



Objective : Find the densest region

Distribution of identical billiard balls

Mean Shift [Comaniciu & Meer]

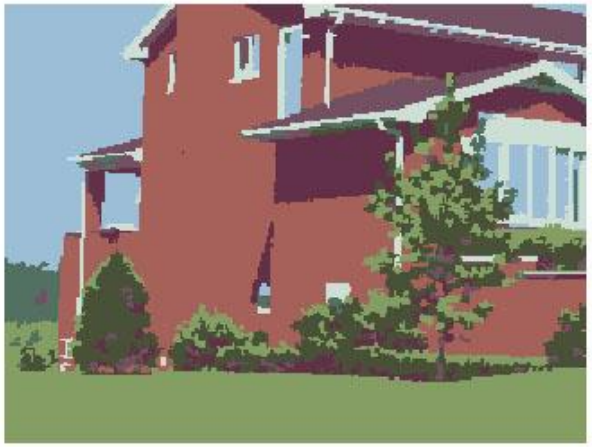


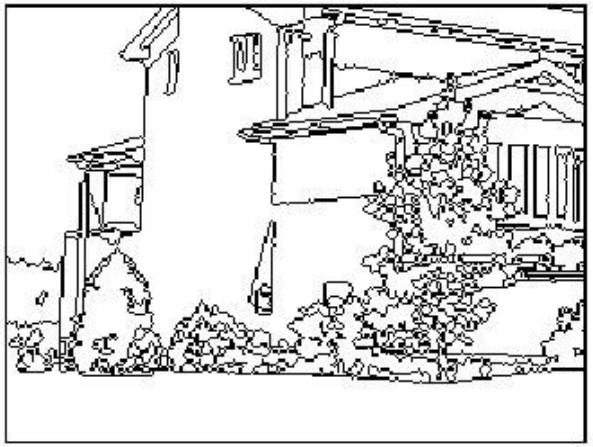
Iterative Mode Search

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

Mean-shift for image segmentation

- Useful to take into account spatial information
  - instead of (R, G, B), run in (R, G, B, x, y) space
  - D. Comaniciu, P. Meer, Mean shift analysis and applications, *7th International Conference on Computer Vision*, Kerkyra, Greece, September 1999, 1197-1203.
    - http://www.caip.rutgers.edu/riul/research/papers/pdf/spatmsft.pdf

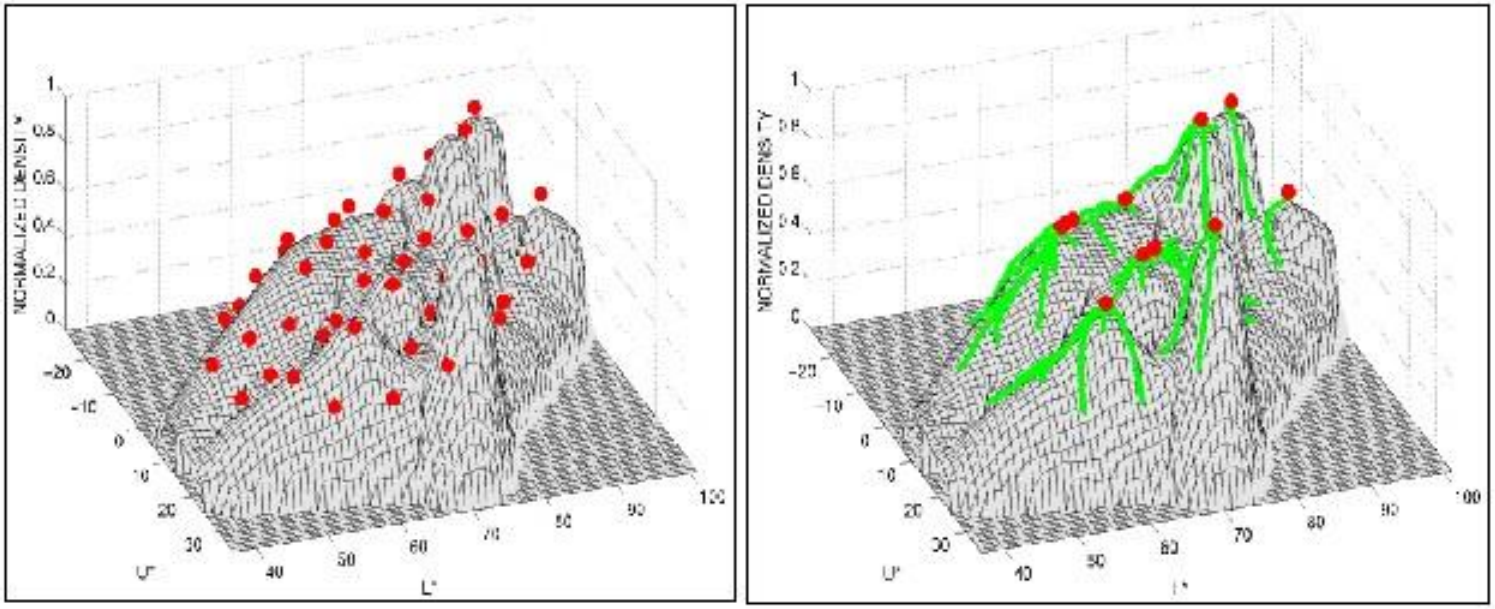




http://www.caip.rutgers.edu/~comanici/segm_images.html

Mean-Shift

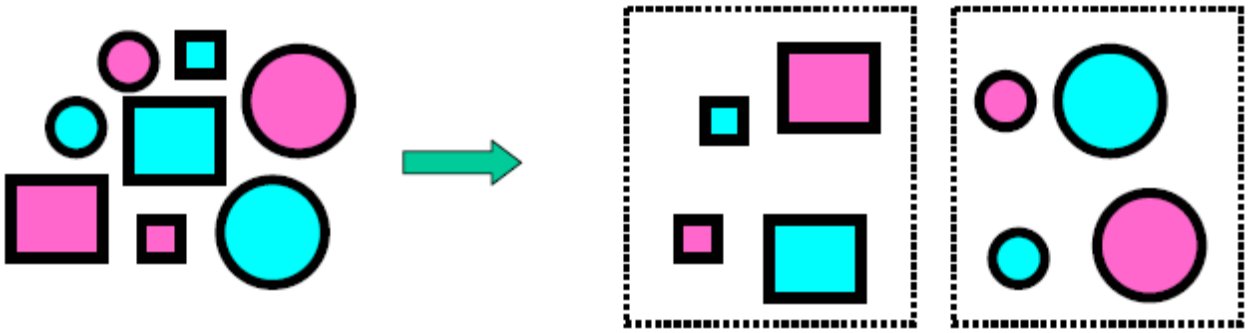
- Approach
  - Initialize a window around each point
  - See where it shifts—this determines which segment it's in
  - Multiple points will shift to the same segment



Mean shift trajectories

Interactive Graph Cuts

- **Introduction**
  - **Segmentation problem**
    - **Grouping objects** by some criteria, such that those within a group will respond similarly and those in a different group will respond differently.



Whole objects

"Segment 1" "Segment 2"

Interactive Graph Cuts

- **Introduction**
  - **Automatic or Semi-automatic**
    - **Fully automatic segmentation**
      - (which seems to) Never be perfect…
    - **Interactive segmentation (semi-auto)**
      - (is evaluated) More reliable…

Interactive Graph Cuts

- **Introduction**
  - **Goal**
    - A general purpose interactive segmentation technique that divides and image into two segments: "**object**" and "**background**".



Interactive Graph Cuts

- **Introduction**
  - **Segmentation method**
    - **Approximate solution**
      - Snake, Deformable templates
      - Shortest path, Ration regions
        - Intensity, Edge (locally minimum)

      ⇨ **Imperfect**

    - **Global optimal solution**
      - MAP-MRF estimation (Graph-Cut)

      ⇨ **Reliable**

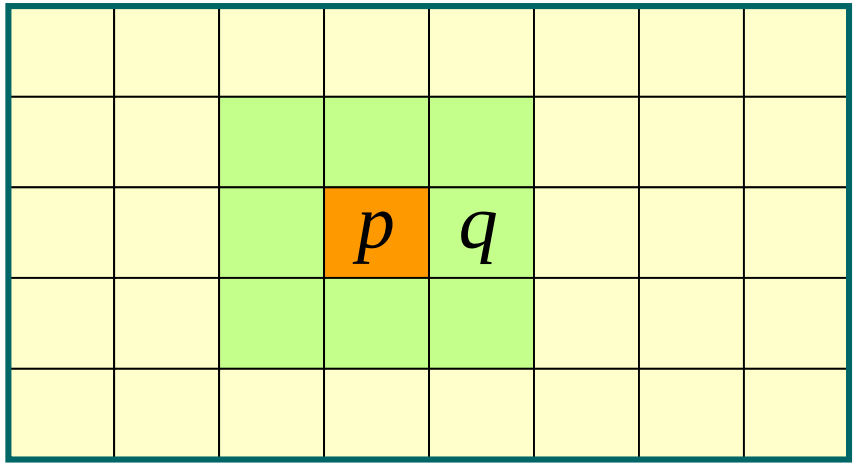
Notations

- **Segmentation**
  - Notation

P : arbitrary set of data elements

N : neighborhood set of all unordered pair $\{p, q\}$

5×8 **Image**

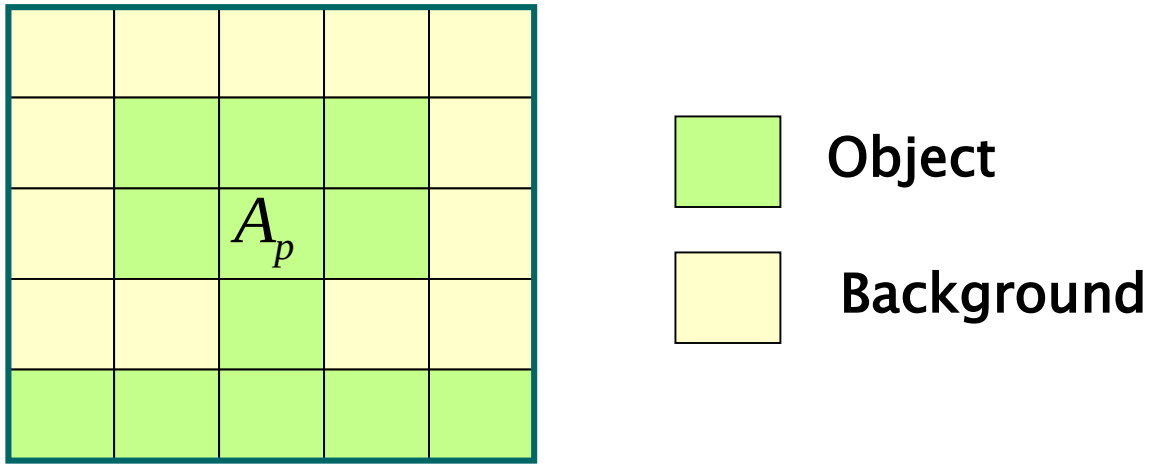


N contain all unordered pairs of neighboring pixels (or voxels) under a standard 8-(or26) neighborhood system.

Notations

- **Segmentation**
  - Notation

$$A = (A_1, \ldots, A_p, \ldots, A_{|p|}) : \text{label vector}$$



Segmentation problem
= allocation problem of proper labeling value to A_p

Formulation

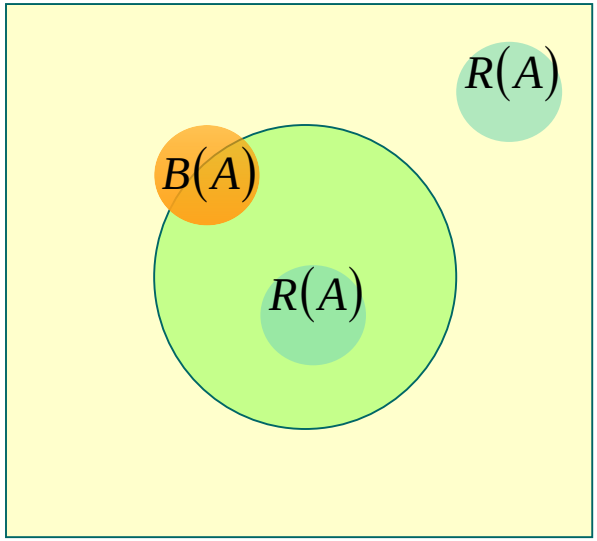
- **Segmentation**
  - Cost function

$$E(A) = \lambda \cdot R(A) + B(A)$$

Coefficient : relative importance

Region properties term

Boundary properties term

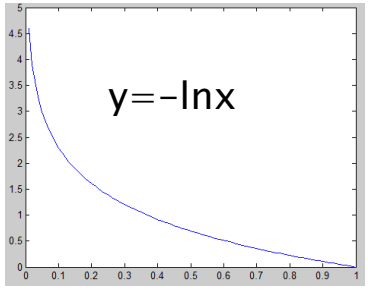


Region Property

- **Segmentation**
  - Regional property term
    - Individual penalties for assigning pixel ***p*** to "object" and "background".

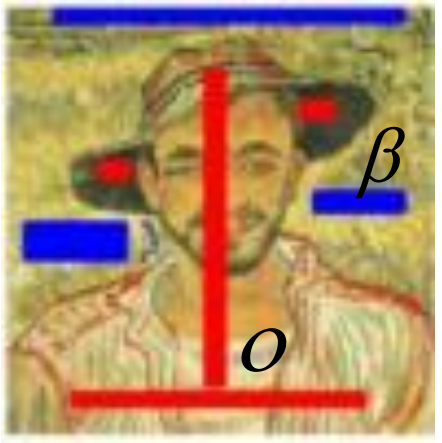
$$R(A)=\sum_{p\in P}R_p(A_p)$$

$R_p(\cdot)$ may reflect on how the intensity of pixel ***p*** fits into a known intensity model (histogram) of object and background.



$$R_p("obj")=-\ln \Pr(I_p \mid o)$$

$$R_p("bkg")=-\ln \Pr(I_p \mid \beta)$$



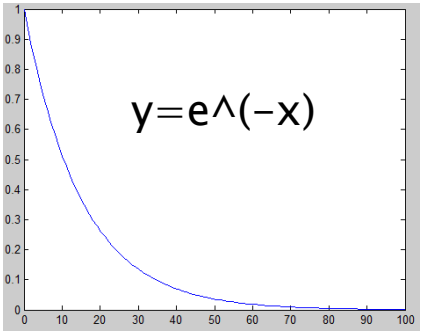
Boundary Property

- **Segmentation**
  - Boundary property term

$$B(A) = \sum_{\{p,q\} \in N} B_{\{p,q\}} \cdot \delta(A_p, A_q)$$

$$\delta(A_p, A_q) = \begin{cases} 1 & if \quad A_p \neq A_q \\ 0 & otherwise \end{cases}$$

For the boundary



$B_{\{p,q\}}$ = discontinuity penalty.

$B_{\{p,q\}}$ is large when pixel p and q are similar.

$B_{\{p,q\}}$ is close to zero when the two are very different.

$$B_{\{p,q\}} \propto \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) \cdot \frac{1}{dist(p,q)}$$

Cost Function

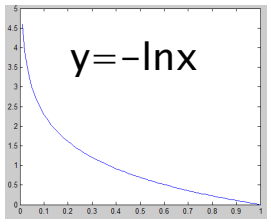
- **Segmentation**
  - **How to select A : minimize the objective function**

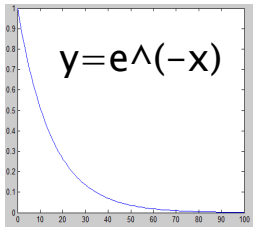
$$E(A) = \lambda \cdot \sum_{p \in P} R_p(A_p) + \sum_{\{p,q\} \in N} B_{\{p,q\}} \cdot \delta(A_p, A_q)$$

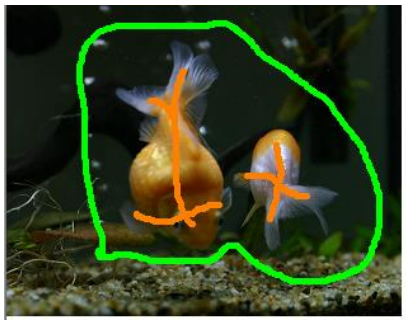
$$R_p(\text{"}obj\text{"}) = -\ln \Pr(I_p \mid o)$$

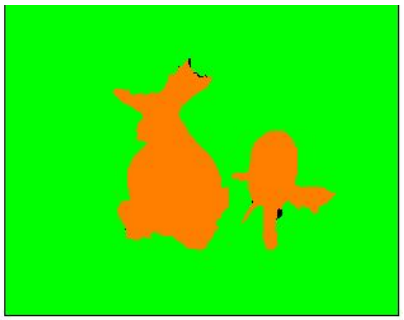
$$R_p(\text{"}bkg\text{"}) = -\ln \Pr(I_p \mid \beta)$$

$$B_{\{p,q\}} \propto \exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right) \cdot \frac{1}{dist(p,q)}$$





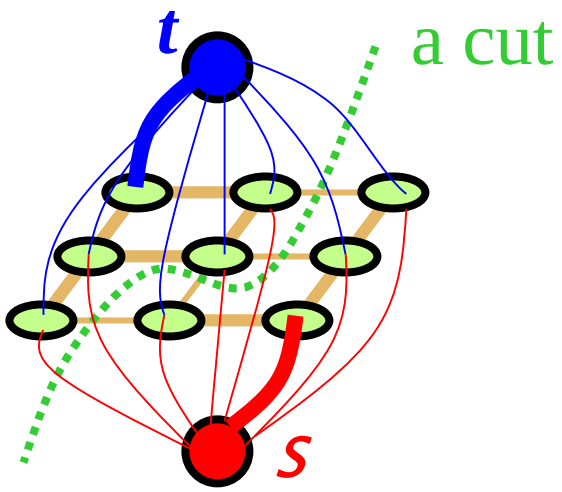




Graph Cuts

- **Graph-Cut based method**
  - Graph

$$G = (V, E)$$



V : nodes

$$V = P \cup \{S, T\}$$

P = pixel node

S = terminal(source)node / object

T = terminal(sink) node / background

E : the edges connecting nodes

$$E = N \bigcup_{p \in P} \{\{p, S\}, \{p, T\}\}$$

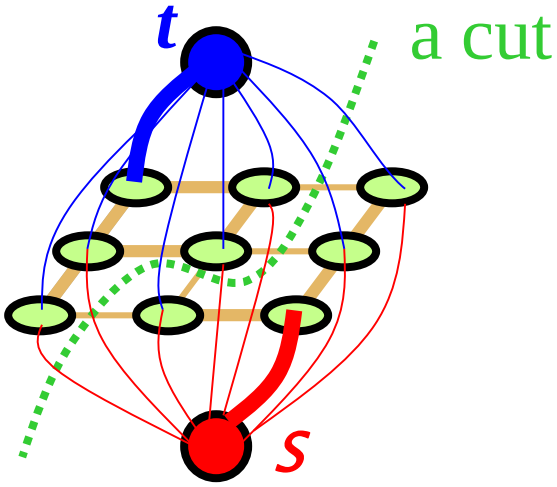
N = n - links (neighborhood links)

$\{p, S \text{ or } T\}$ = t - link (terminal link)

Graph Cuts

- **Graph-Cut based method**
  - Graph

$$G = (V, E)$$



- Each edge $e \in E$ is assigned a nonnegative weight w_e .
- C is set of cut edges that separate the terminals on the graph. $C \subset E$

$$|C| = \sum_{e \in C} w_e$$

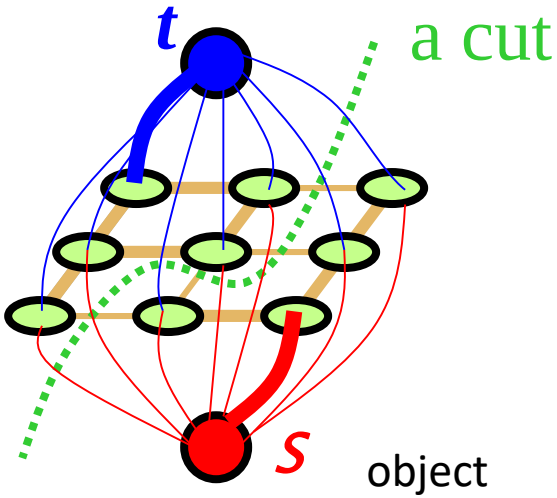
Defining Weights

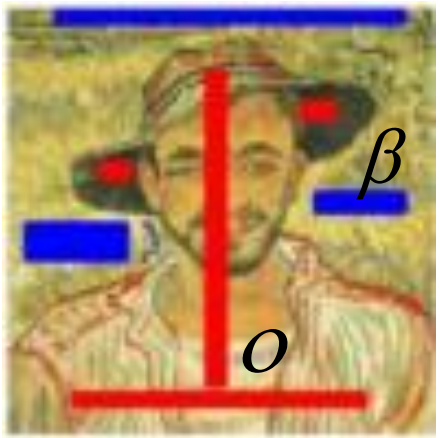
- **Graph-Cut based method**
  - How to set the weight

edge	weight (cost)	for
$\{p,q\}$	$B_{\{p,q\}}$	$\{p,q\} \in \mathcal{N}$
$\{p,S\}$	$\lambda \cdot R_p(\text{“bkg”})$	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
$\{p,T\}$	$\lambda \cdot R_p(\text{“obj”})$	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$

where

$$K = 1 + \max_{p \in \mathcal{P}} \sum_{q:\ \{p,q\} \in \mathcal{N}} B_{\{p,q\}}.$$

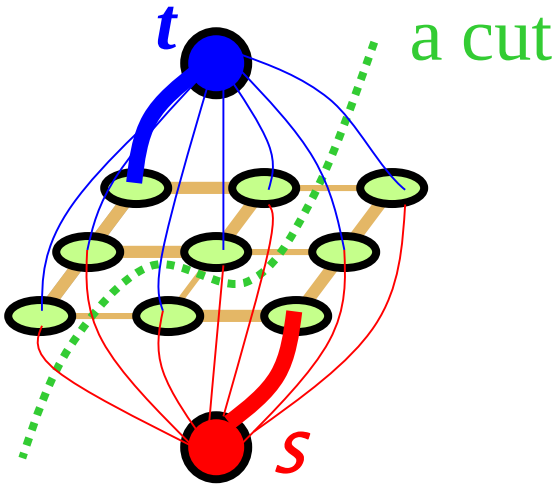




Max-Flow/Min Cut Algorithm

- **Graph-Cut based method**
  - Finding min-cut

$$|C| = \sum_{e \in C} w_e$$

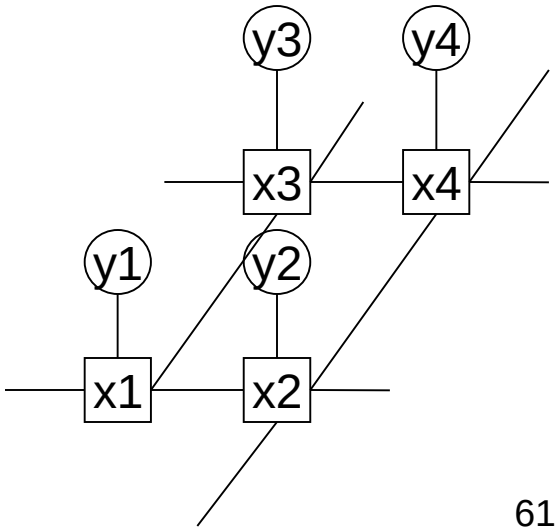


  - To be specific, assume that a max-flow algorithm is used to determine the minimum cut on G.

- An Experimental Comparison of Min-Cut/Max-Flow Algorithm for Energy Minimization in Vision
  - Stefan Roth, Michael J. Black (PAMI Sept. 2004)

Markov Random Field (MRF)

- Markov Random Field (MRF)
  - MRFs are a kind of statistical model
  - Easily describe local relationship
  - MRFs replace temporal dependency of Markov chains with spatial dependency



Markov Random Field (MRF)

- The undirected graph which is often used in many vision problems
  - Observation node: y
    - Represent some visible information in low-level problem
  - Hidden node: x
    - Each node own a corresponding observation node *y*

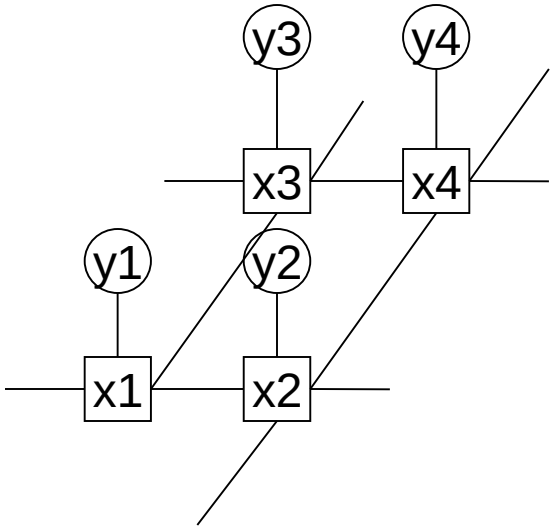
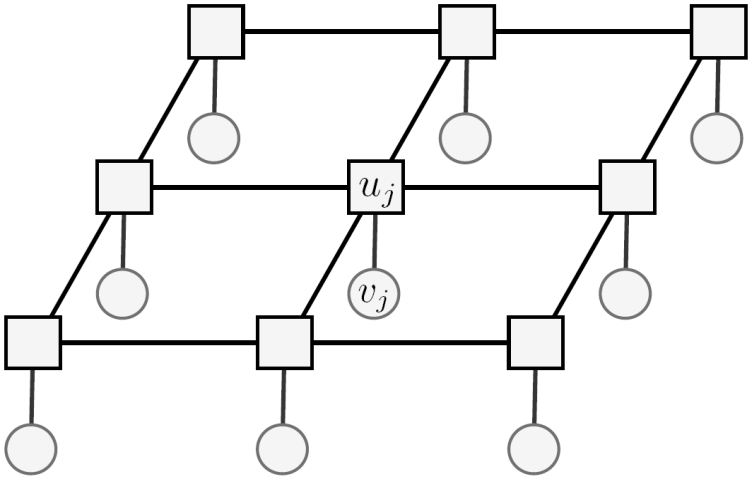


Image Denoising

- Consider image restoration: Given a noisy image v, perhaps with missing pixels, recover an image u that is both smooth and close to v.



Accordingly, the energy function is given by

$$E(u) = \sum_{i \in \mathcal{V}} D(u_i) + \sum_{(i,j) \in \mathcal{E}} V(u_i, u_j)$$

Image Denoising

Let v be the sum of a smooth 1D signal u and IID Gaussian noise e:

$$v = u + e \ , \tag{4}$$

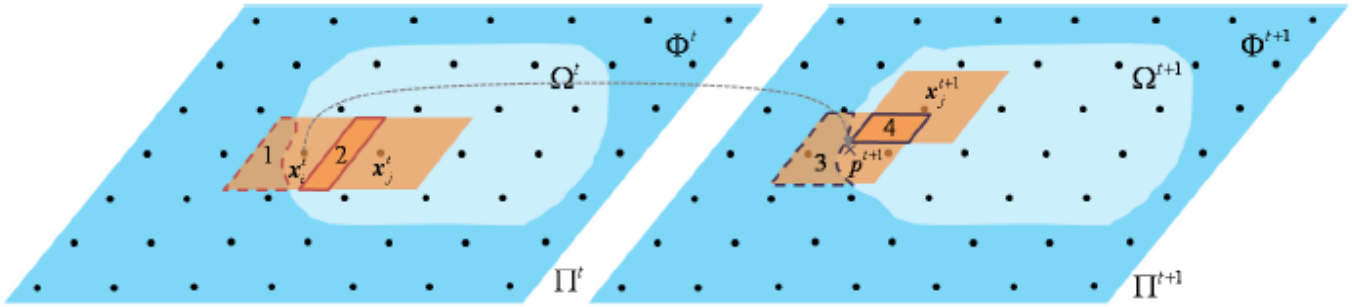
where $u = (u_1, ..., u_N)$, $v = (v_1, ..., v_N)$, and $e = (e_1, ..., e_N)$.

With Gaussian IID noise, the negative log likelihood provides a quadratic *data term*. If we let the *smoothness term* be quadratic as well, then up to a constant, the log posterior is

$$E(u) = \sum_{n=1}^{N} (u_n - v_n)^2 + \lambda \sum_{n=1}^{N-1} (u_{n+1} - u_n)^2 \ . \tag{5}$$

Video Completion

- Goal: to restore the spatial temporal missing regions of a video in a visually plausible way.



$$E(X) = E_s(X) + \alpha E_t(X),$$

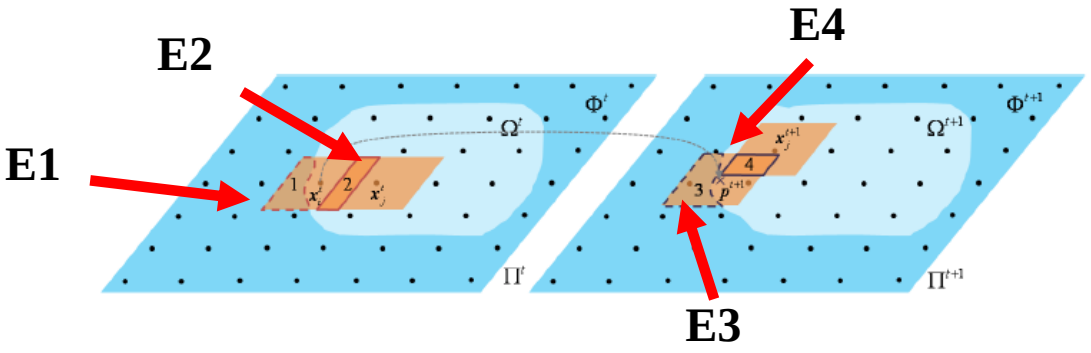
Video Completion

- **The Spatial Term**

$$E_s(X) = \sum_{v_i^t} E_1(x_i^t) + \sum_{(v_i^t, v_j^t) \in \mathcal{E}_s} E_2(x_i^t, x_j^t),$$

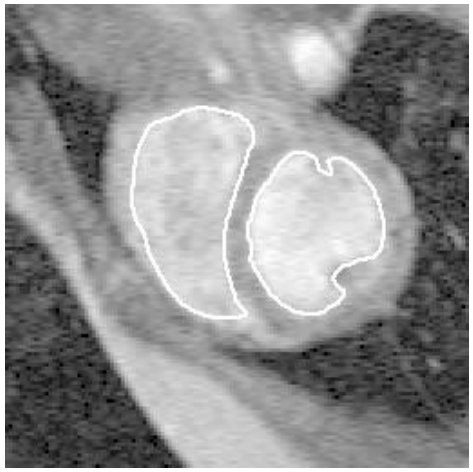
$$E_1(x_i^t) = C_i^t \cdot d(x_i^t, \Phi^t),$$

$$E_2(x_i^t, x_j^t) = \left[\frac{C_i^t + C_j^t}{2}\right] \left[\lambda_1 E_2'(x_i^t, x_j^t) + \lambda_2 E_2''(x_i^t, x_j^t)\right],$$



Active Contour Model - Snake

- Energy-minimizing spline guided by external constraint forces and pulled by image forces toward features.



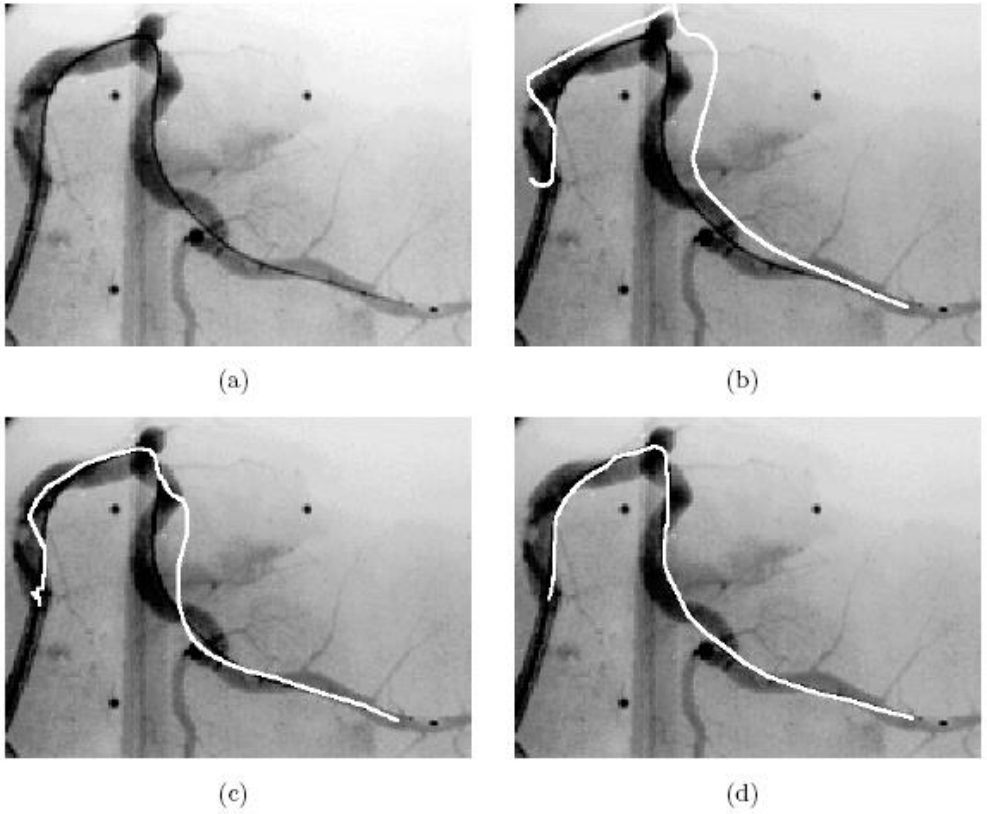
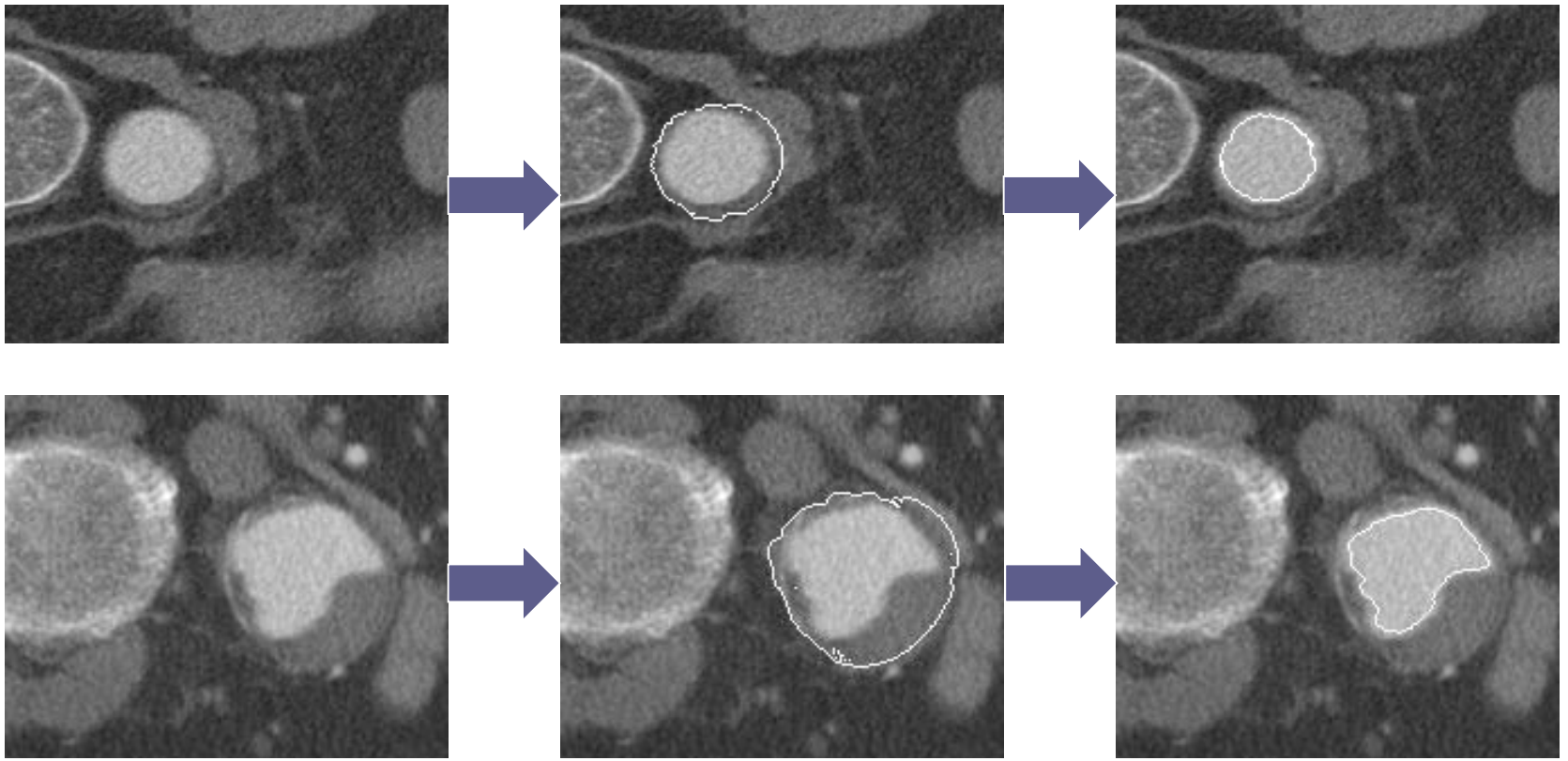


Figure 7.7: Snake-based detection of the intravascular ultrasound catheter (dark line positioned inside the coronary artery lumen) in an angiographic X-ray image of a pig heart. (a) Original angiogram. (b) Initial position of the snake. (c) Snake deformation after 4 iterations. (d) Final position of the snake after 10 iterations.

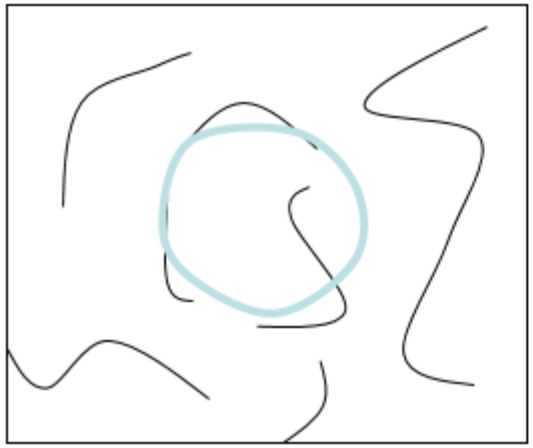
Introduction

- First an initial spline (snake) is placed on the image, and then its energy is minimized.
- Local minima of this energy correspond to desired image properties.

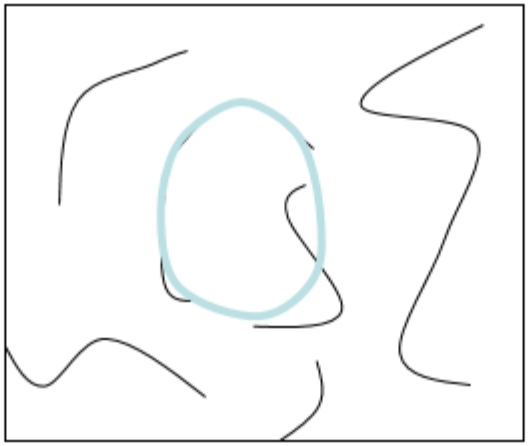


“Snakes”

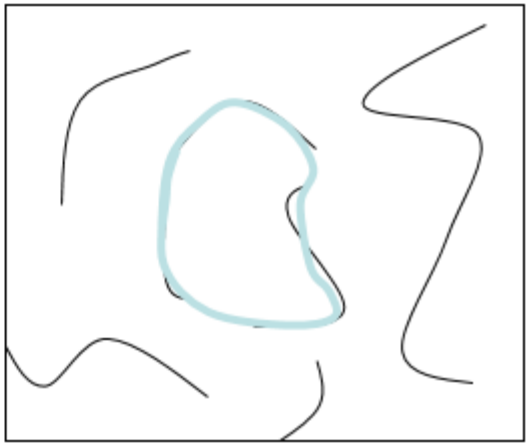
- A smooth 2D curve which matches to image data
- Initialized near target, iteratively refined
- Can restore missing data



initial



intermediate



final

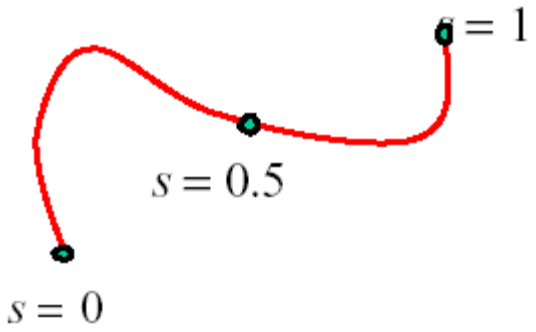
Snake Behavior

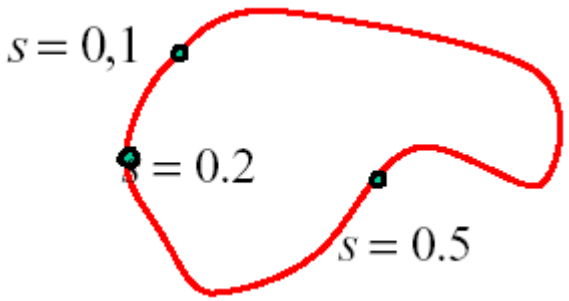
- A snake falls into the closest *local* energy minimum.
- The local minima of the snake energy comprise the set of alternative solutions
- A higher level knowledge is needed to choose the "*correct one*" from these solutions
  - High level reasoning
  - User interaction
- These high-level methods can *interact* with the contour model by pushing it toward an appropriate local minimum

Parametric Curve Representation
(continuous case)

- A curve can be represented parametrically

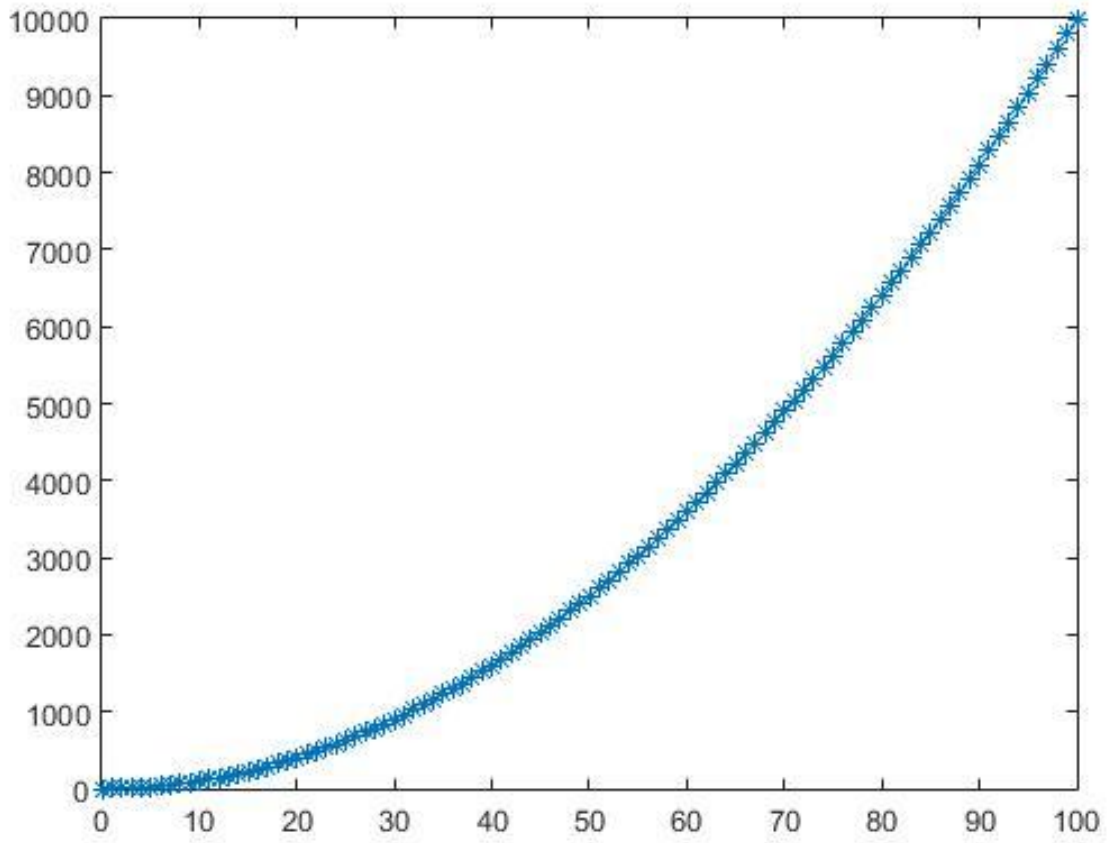
$$v(s) = (x(s), y(s)) \qquad 0 \le s \le 1$$





Parametric Curve Representation

- Let t = 0:1:100;
- $x = t$; $y = t$.* t;
- Plot(x, y, '*');



Snake Algorithm

- The snake is defined parametrically as v(s)=[x(s),y(s)], where s∈[0,1] is the normalized arc length along the contour. The energy functional to be minimized may be written as

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds$$

$$= \int_0^1 E_{\text{int}}(v(s))ds + \int_0^1 E_{image}(v(s))ds + \int_0^1 E_{forces}(v(s))ds$$

E_{int}: internal energy of the spline due to bending.
E_{image}: image forces pushing the snake toward image features, such as edges.
E_{forces} : external constraints forces responsible for putting the snake near the desired local minimum. It may come from: Higher level interpretation or User interaction, etc.

Internal Energy

- The snake is a *controlled continuity* spline
  - Regularizes the problem

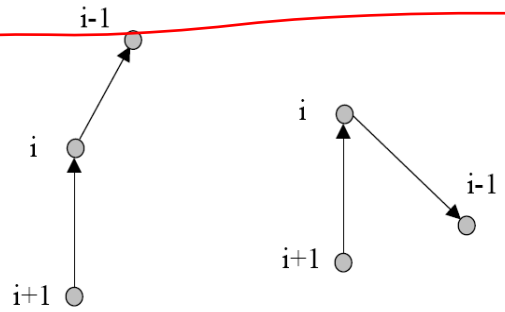
$$E_{\text{int}} = (\alpha(s)\,|\,v_s(s)\,|^2 + \beta(s)\,|\,v_{ss}(s)\,|^2)/2$$

  - The first order derivative *Vs(s)* makes the spline act like a *membrane* ("elasticity").
  - The second order derivative *Vss(s)* makes it act like a *thin-plate* ("rigidity").
  - *α(s)* and β*(s)* controls the relative importance of membrane and thin-plate terms
  - Setting *β(s)=0* for a point allows the snake to become second-order discontinuous and develop a corner.

Internal Energy

- Shrinkage : first-order derivative
  - minimization makes neighboring points closer.
- Smooth: second-order derivative
  - $|v_{i-1}-2v_i+v_{i+1}| = |(v_{i-1}-v_i)-(v_i-v_{i+1})|$
  - It can be seen as the difference of vectors $V_{i-1, i}$ and $V_{i, i+1}$.
  - If the difference is small, it is smooth. Otherwise, the angle of direction may be large.

$$E_{\text{int}} \approx \sum_{i=0}^{n-1}[\alpha_i \frac{|v_i - v_{i-1}|}{h} + \beta_i \frac{|v_{i-1} - 2v_i + v_{i+1}|}{h^2}]$$

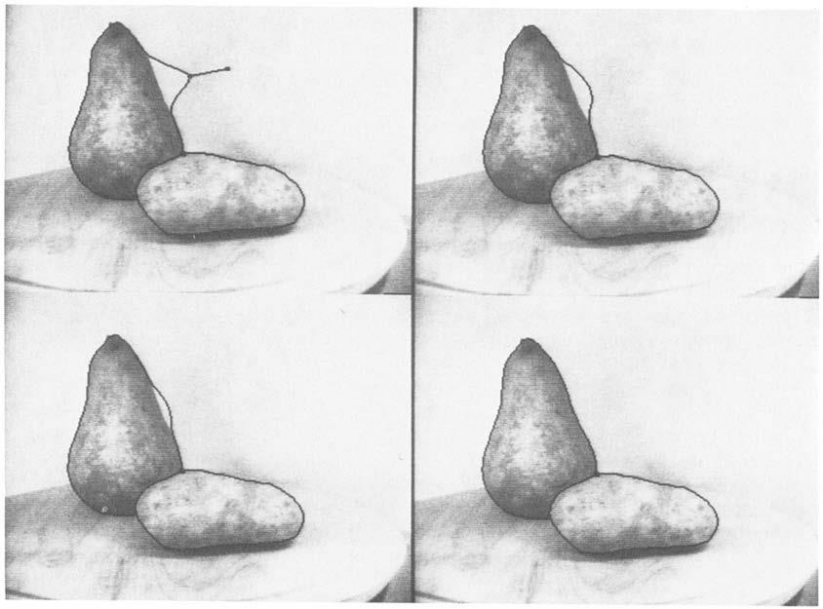


Simple Edge Strength

- Suppose we have an image *I(x,y)*
- Generate gradient images $G_x(x,y)$ & $G_y(x,y)$
- External energy based on edge strength at a point is then

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

- Then, the snake is attracted to contours with large image gradients.



Line Functional

$$E_{\text{line}} = I(x, y)$$

- depending on the sign of w_{line}, the snake will be attracted either to light lines or dark lines.
- Subject to its other constraints, the snake will try to align itself with the lightest or darkest nearby contour.

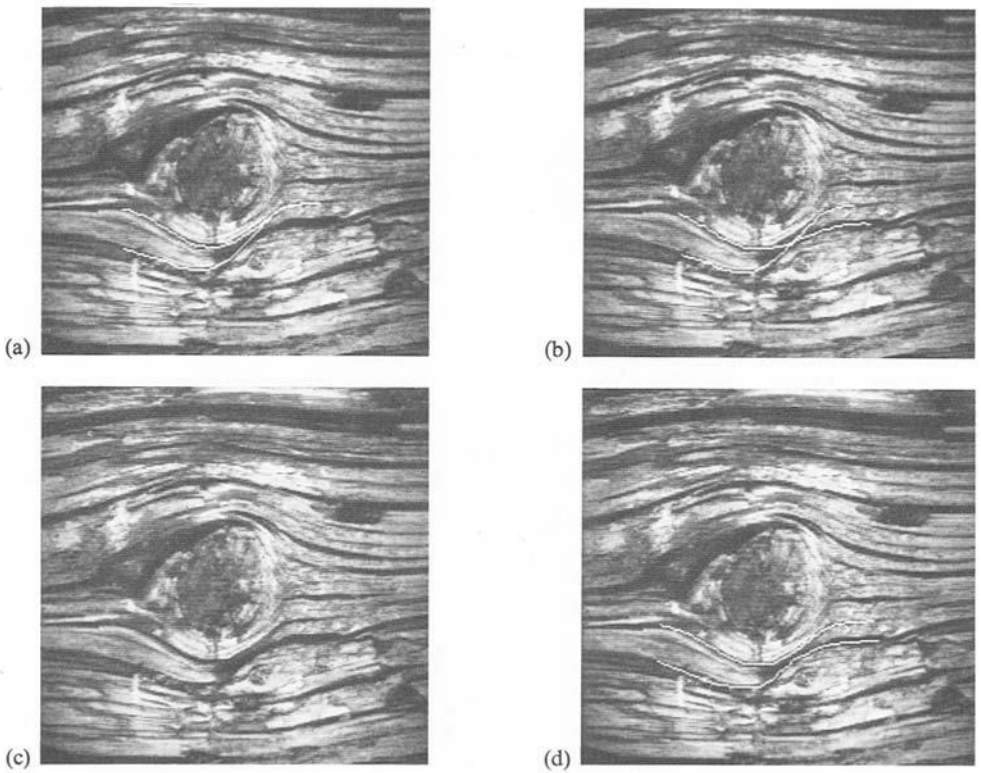


Image Forces

- The image forces E_{image} are derived from the image data over which the snake lies.
- Three important features the snake can be attracted to are line, edge and termination functions. The total image energy can be expressed as a weighted combination of the three.

$$E_{image} = \omega_{line} E_{line} + \omega_{edge} E_{edge} + \omega_{term} E_{term}$$

Discrete approach

discrete image

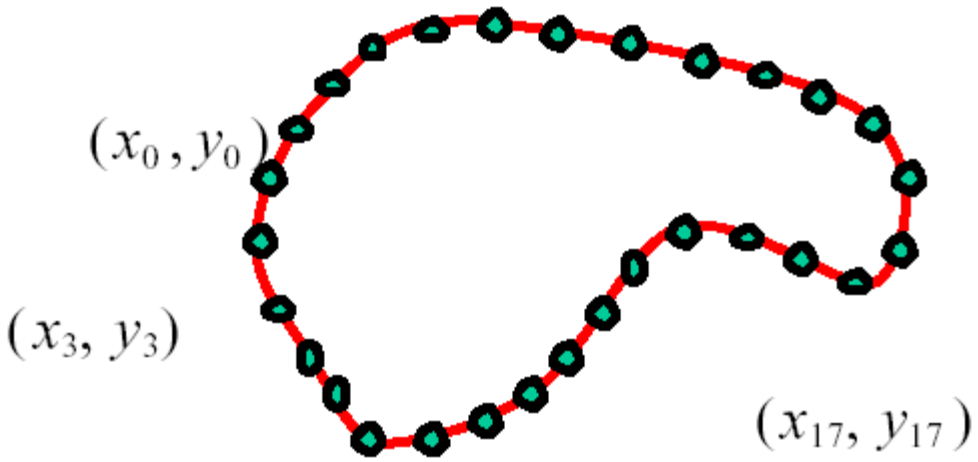
discrete snake representation

discrete optimization (dynamic programming)

Parametric Curve Representation
(discrete case)

- Represent the curve with a set of n points

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



Discrete Representation

- If the curve is represented by *n* points

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

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

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

Elasticity

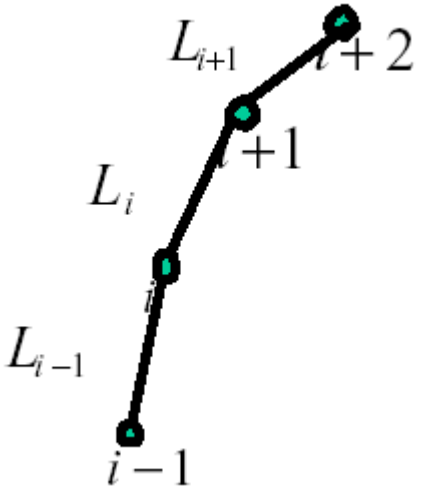
Stiffness

Simple Elastic Curve (example)

- For a curve represented as a set of points a simple elastic energy term is

$$E_{in} = \alpha \cdot \sum_{i=0}^{n-1} L_i^{\,2}$$

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



Simple Edge Strength

- An external energy term for a (discrete) snake based on image edge

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

Simple Elastic Snake

- A simple elastic snake is thus defined by
  - A set of *n* points,
  - An internal elastic energy term
  - An external edge based energy term
- To use this to locate the outline of an object
  - Initialize in the vicinity of the object
  - Modify the points to minimize the total energy

Simple Elastic Snake

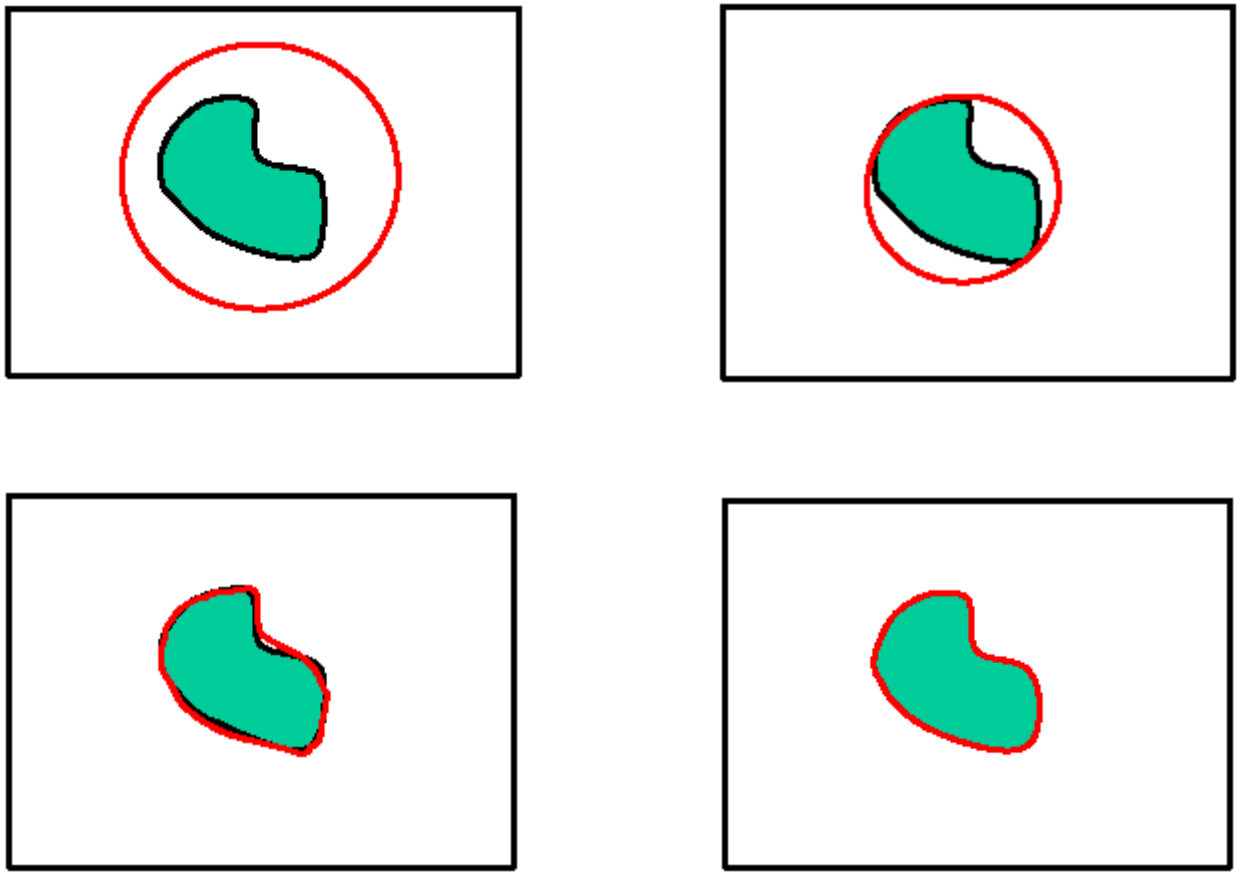
- In this case we have

$$E_{total}(x) = E_{in}(x) + E_{ex}(x)$$

  - where $x = (x_0, \cdots, x_{n-1}, y_0, \cdots, y_{n-1})^T$

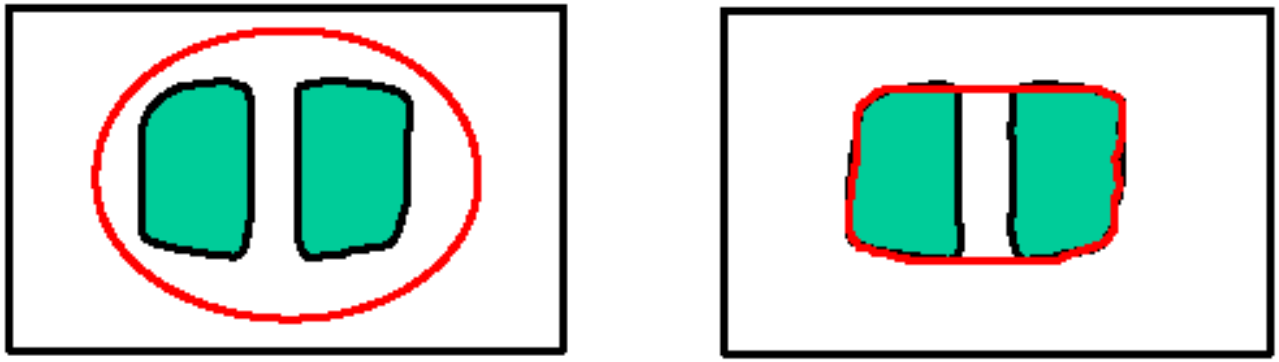
- Optimization problem (*2n* variables)
  - note that we can easily compute the derivatives, which allows efficient optimization converging to a local minima

Synthetic example



Dealing with missing data

- The smoothness constraint can deal with missing data (sometimes maybe wrong!):

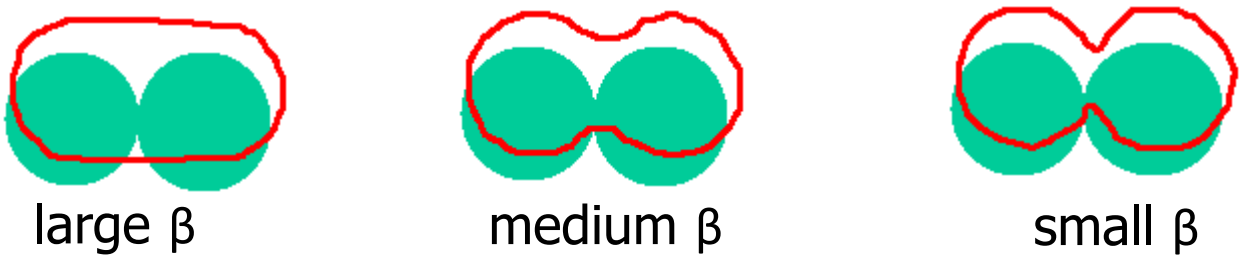


Relative weighting

- Notice that the strength of the internal elastic component can be controlled by a α parameter,

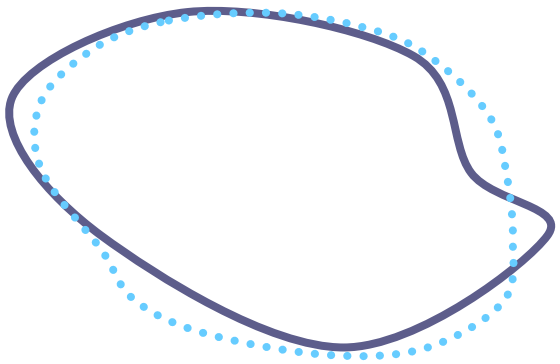
$$E_{in} = \alpha \cdot \sum_{i=0}^{n-1} L_i^{\,2}$$

- Increasing β increases stiffness of curve



Discrete Snakes Optimization

- At each iteration we compute a new snake position within proximity to the previous snake
- New snake energy should be smaller than the previous one
- Stop when the energy can not be decreased within local neighborhood of the snake (local energy minima)



Optimization Methods

1. *Gradient Descent*
2. *Dynamic Programming*

Snake energy: pair-wise interactions

Example: simple elastic snake energy

$$E_{total}(x_0,\ldots,x_{n-1},y_0,\ldots,y_{n-1}) = -\sum_{i=0}^{n-1} |G_x(x_i,y_i)|^2 + |G_y(x_i,y_i)|^2 + \alpha\cdot\sum_{i=0}^{n-1}(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2$$

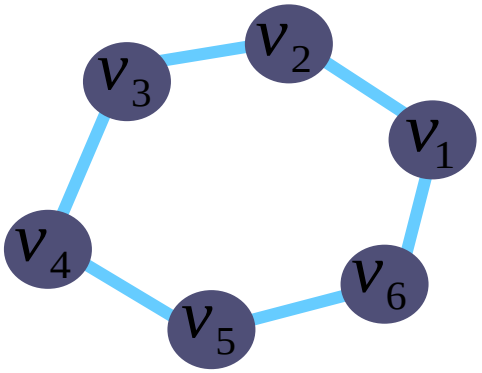
$$E_{total}(v_0,\ldots,v_{n-1}) = -\sum_{i=0}^{n-1} \|G(v_i)\|^2 + \alpha\cdot\sum_{i=0}^{n-1} \|v_{i+1}-v_i\|^2$$

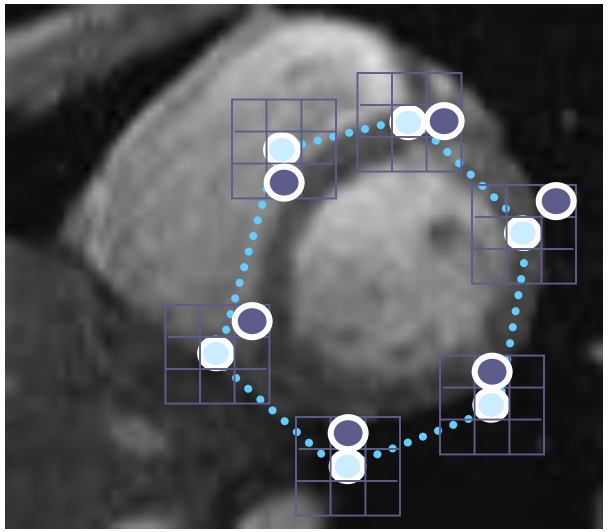
$$E_{total}(v_0,\ldots,v_{n-1}) = \sum_{i=0}^{n-1} E_i(v_i,v_{i+1})$$

$$E_i(v_i,v_{i+1}) = -\|G(v_i)\|^2 + \alpha\|v_i - v_{i+1}\|^2$$

DP Snakes

[Amini, Weymouth, Jain, 1990]





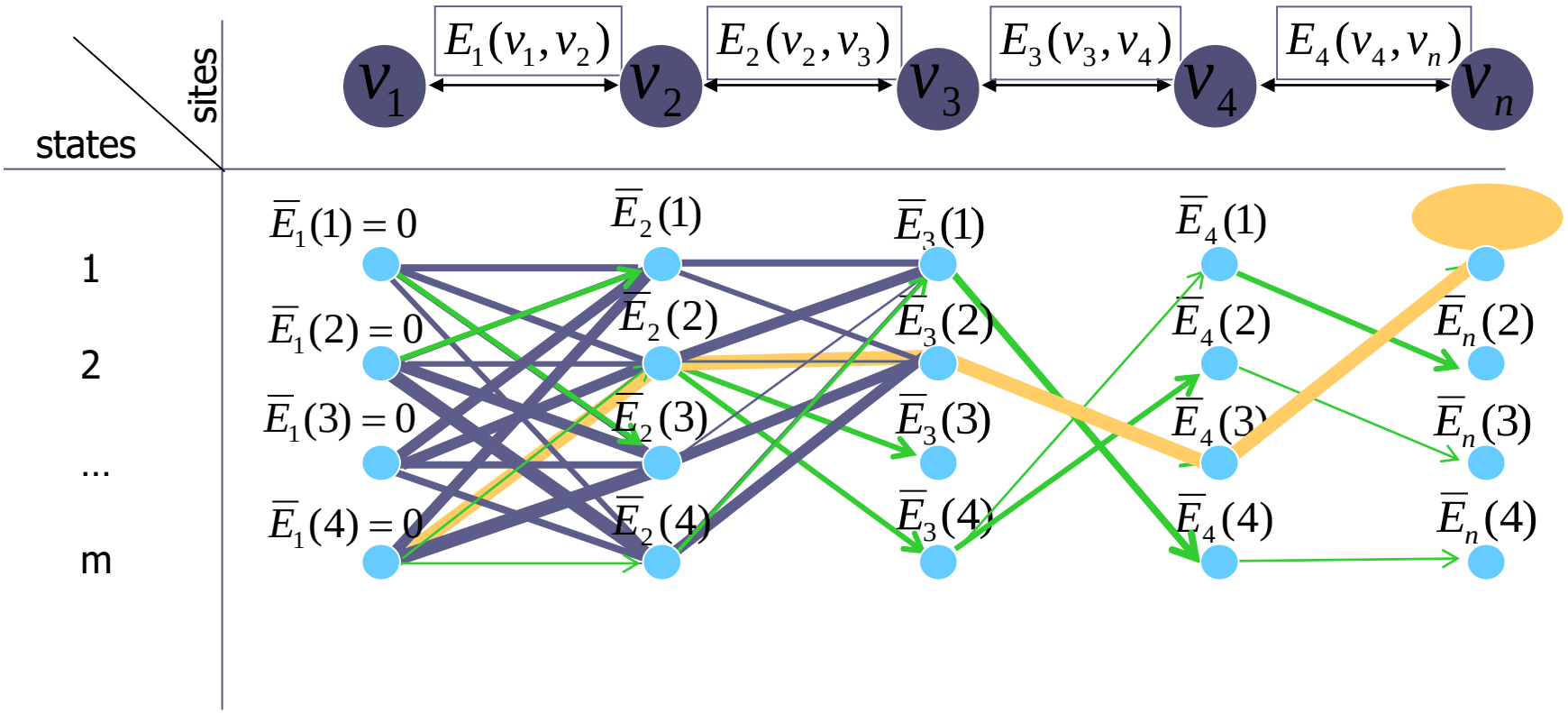
$$E(v_1, v_2, \ldots, v_n) = E_1(v_1, v_2) + E_2(v_2, v_3) + \ldots + E_{n-1}(v_{n-1}, v_n)$$

Energy E is minimized via Dynamic Programming

Dynamic Programming (DP)

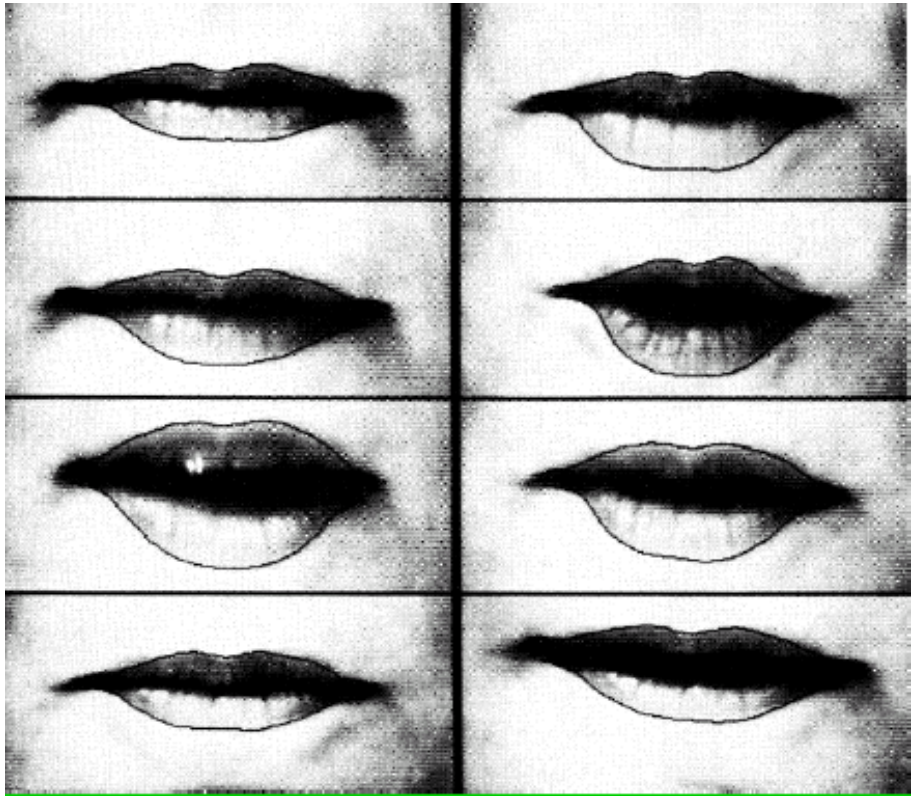
Here we will concentrate on **first-order interactions**

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

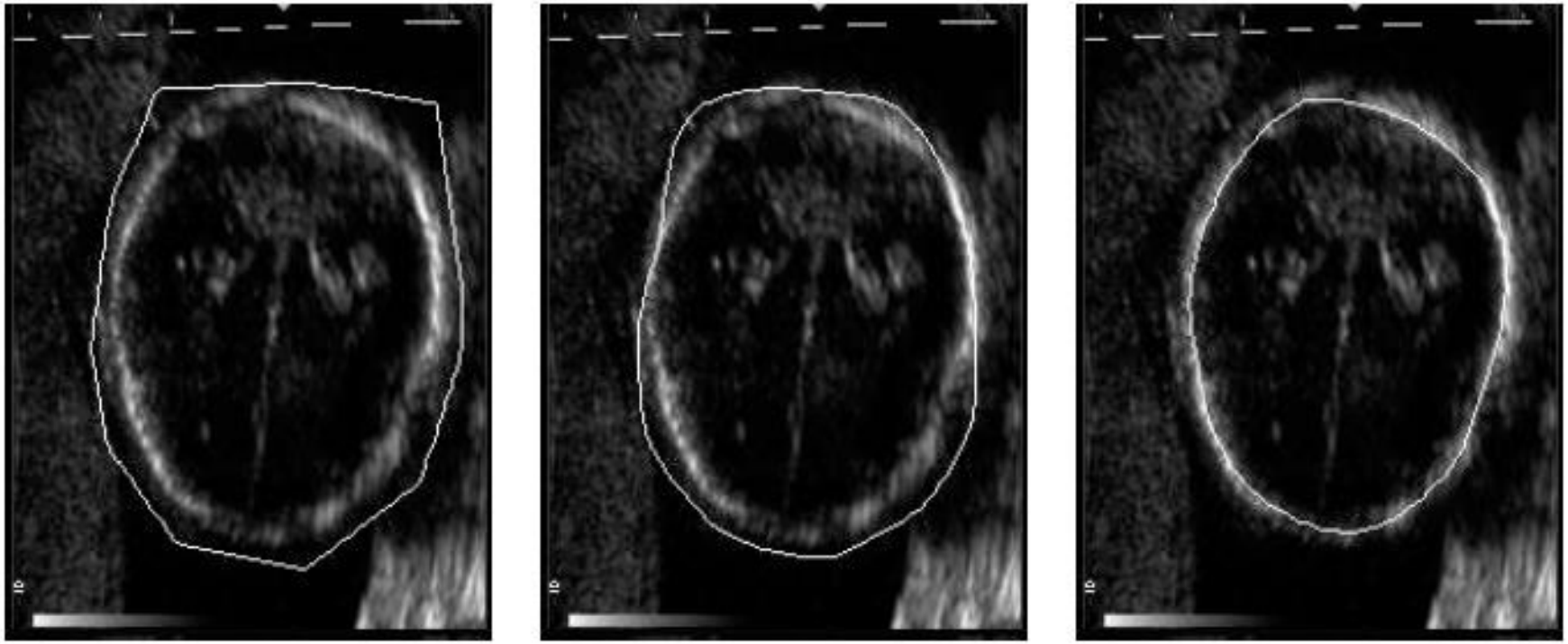


Motion Tracking

- Once a snake finds a feature, it "locks on".
- If the feature begins to move, the snake will track the same local minimum.
- Fast motion could cause the snake to flip into a different minimum.



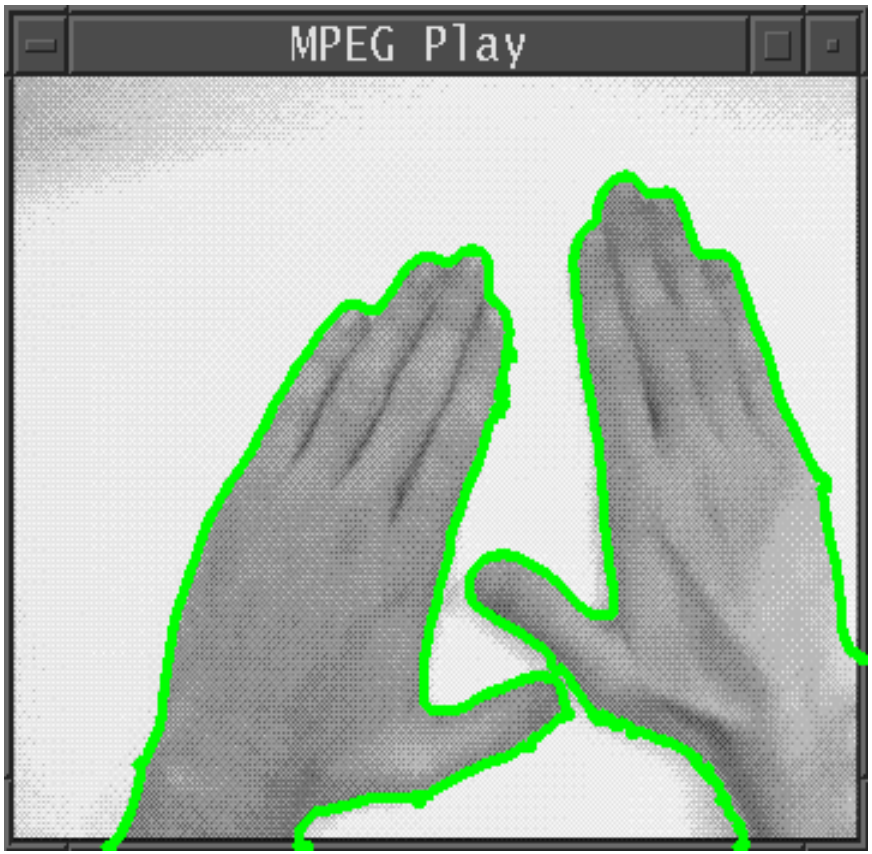
Balloon Snake



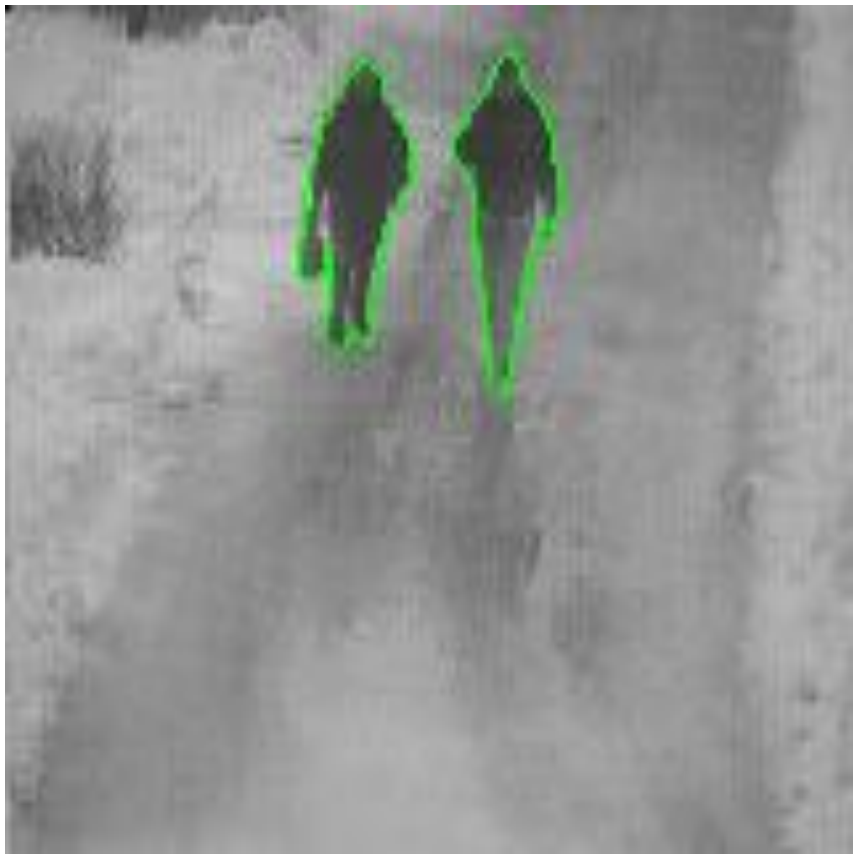
(a) (b) (c)

Figure 7.10: Balloon-based image segmentation of an ultrasound image of a fetal head. (a) Initial position of the balloon. (b) Balloon deformation after 10 iterations. (c) Final position of the balloon after 25 iterations. *Courtesy of V. Chalana.*

Examples



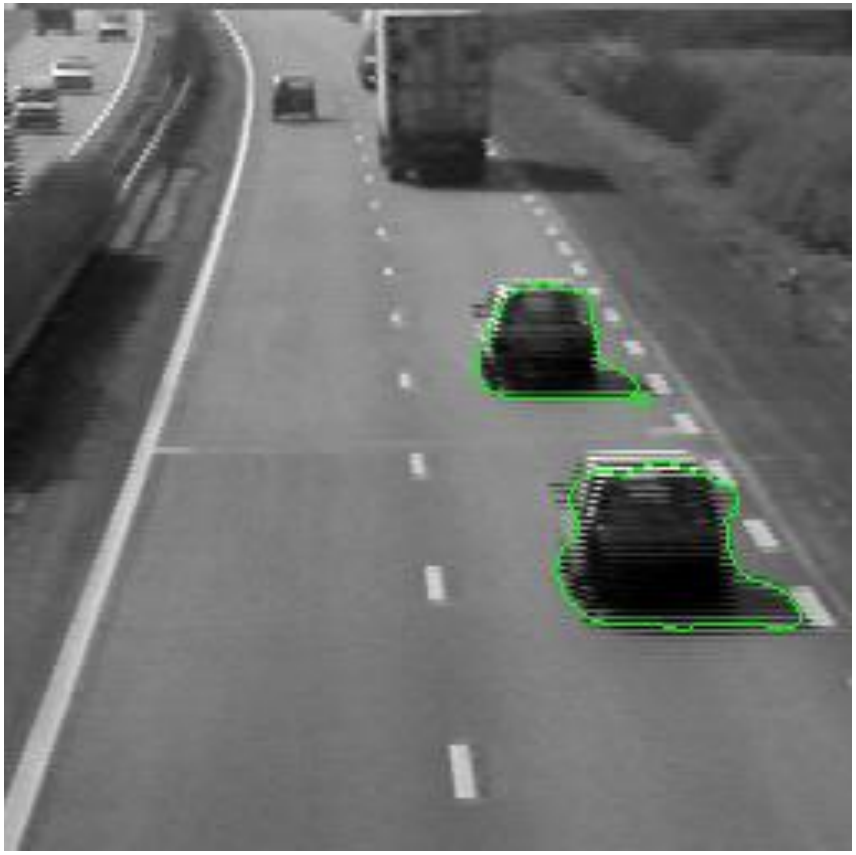
Hands



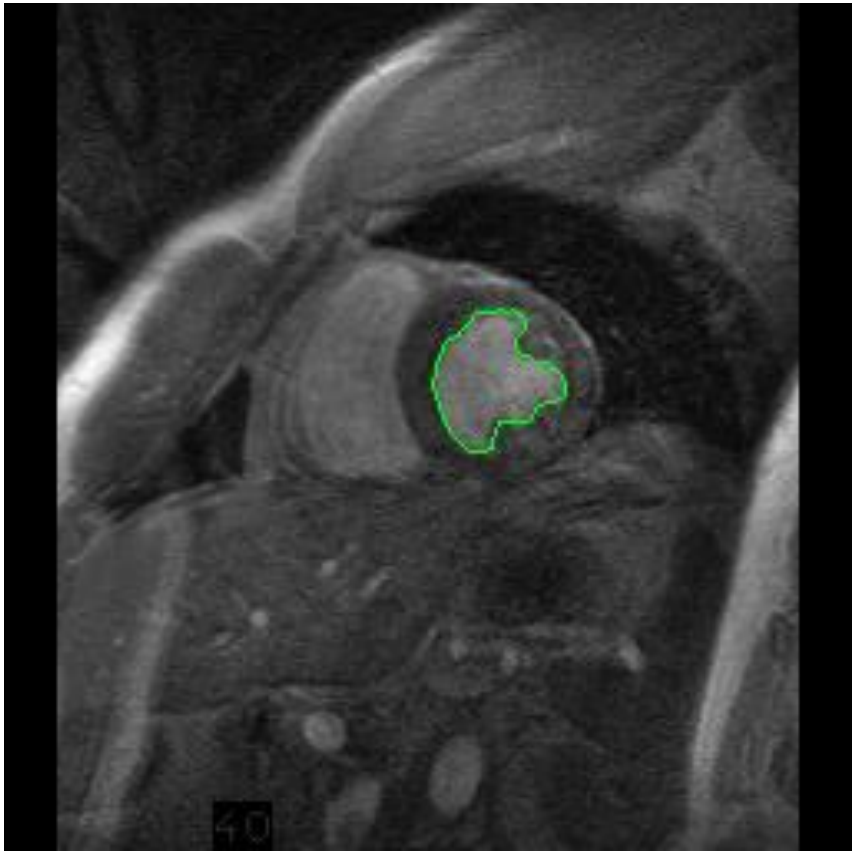
People

Examples from Julien Jomier

Examples



Highway



Heart

Problems with snakes

- Snakes sometimes degenerate in shape by shrinking and flattening.
- Stability and convergence of the contour deformation process may be unpredictable.

Solution: Add some constraints: External forces or Physical properties

- Initialization is not straightforward.
  Solution: Manual and Statistics

Weakness of traditional snakes

- Extremely sensitive to parameters.
- Small capture range.
- No external force acts on points which are far away from the boundary.
- Convergence is dependent on initial position.

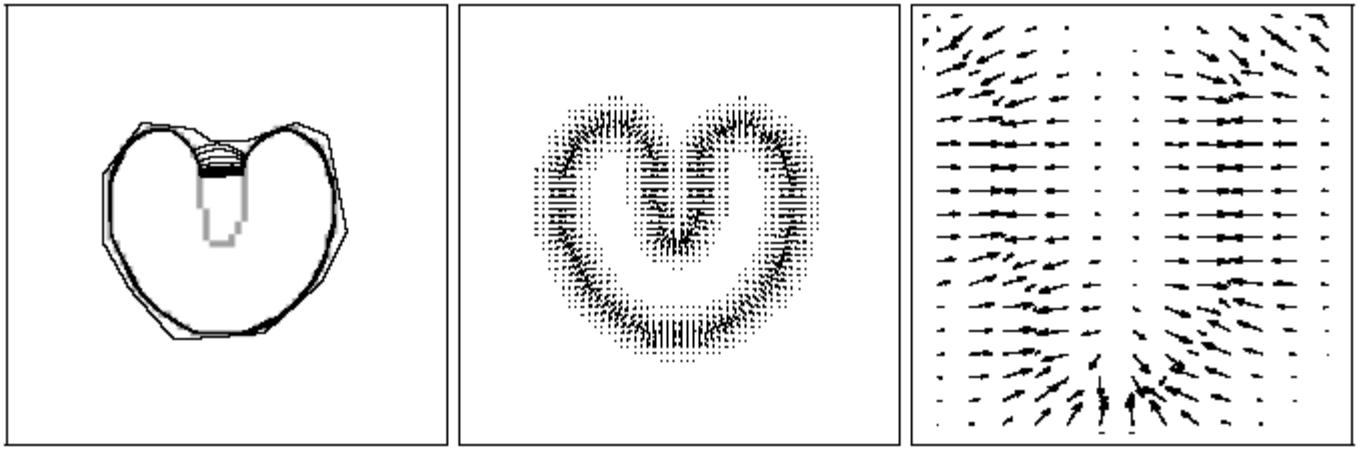
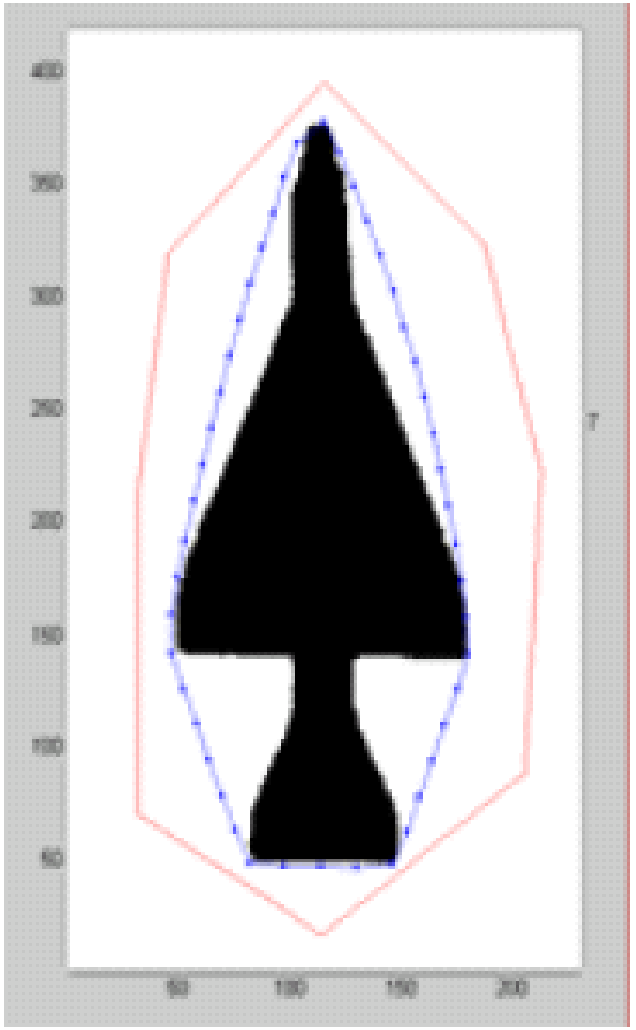


Figure 7.11: Classic snake convergence. (a) Convergent sequence of snake locations. Note that the snake fails segmenting the concave boundary. (b) Classic external forces. (c) Close-up of the concave object region. No forces exist capable of pulling the snake inside the bay. *Courtesy of J. L. Prince and C. Xu, Johns Hopkins University, ©1998 IEEE [Xu and Prince, 1998].*

Weakness of traditional snakes

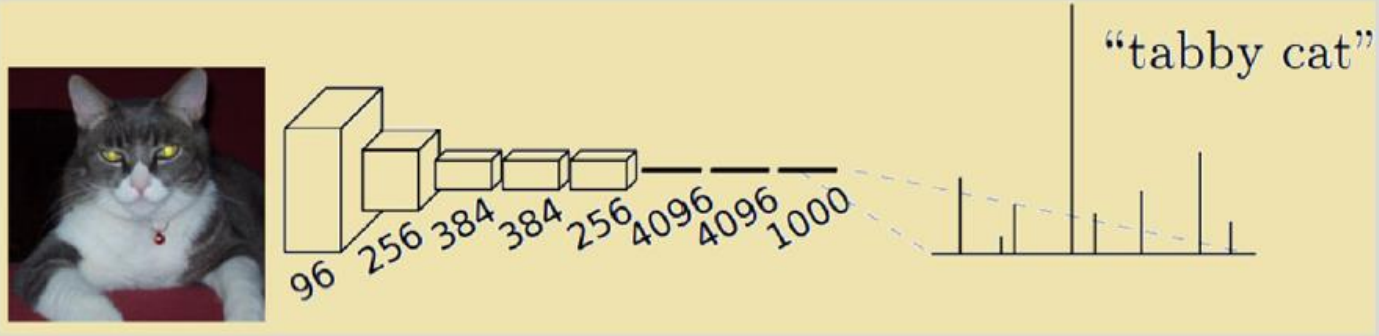
- Fails to detect concave boundaries. External force can't pull control points into boundary concavity.



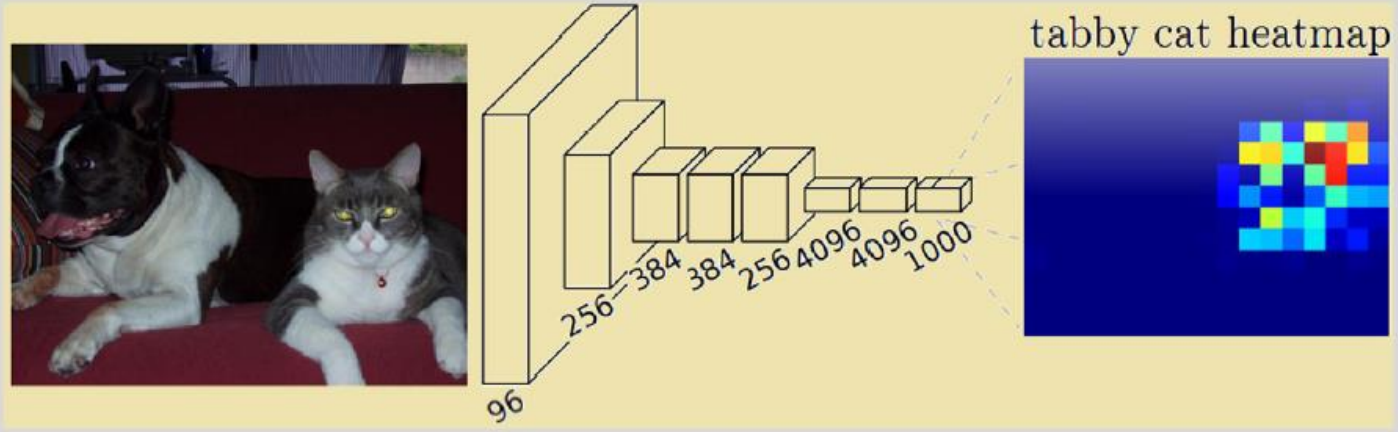
From classification to semantic segmentation

CNN for Pixel-wise Labelling

- Usual convolutional networks



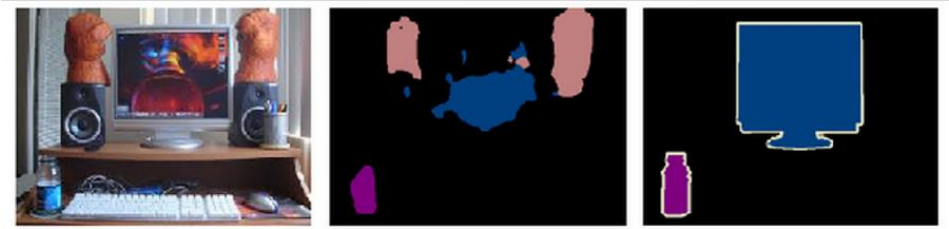
- Fully convolutional networks

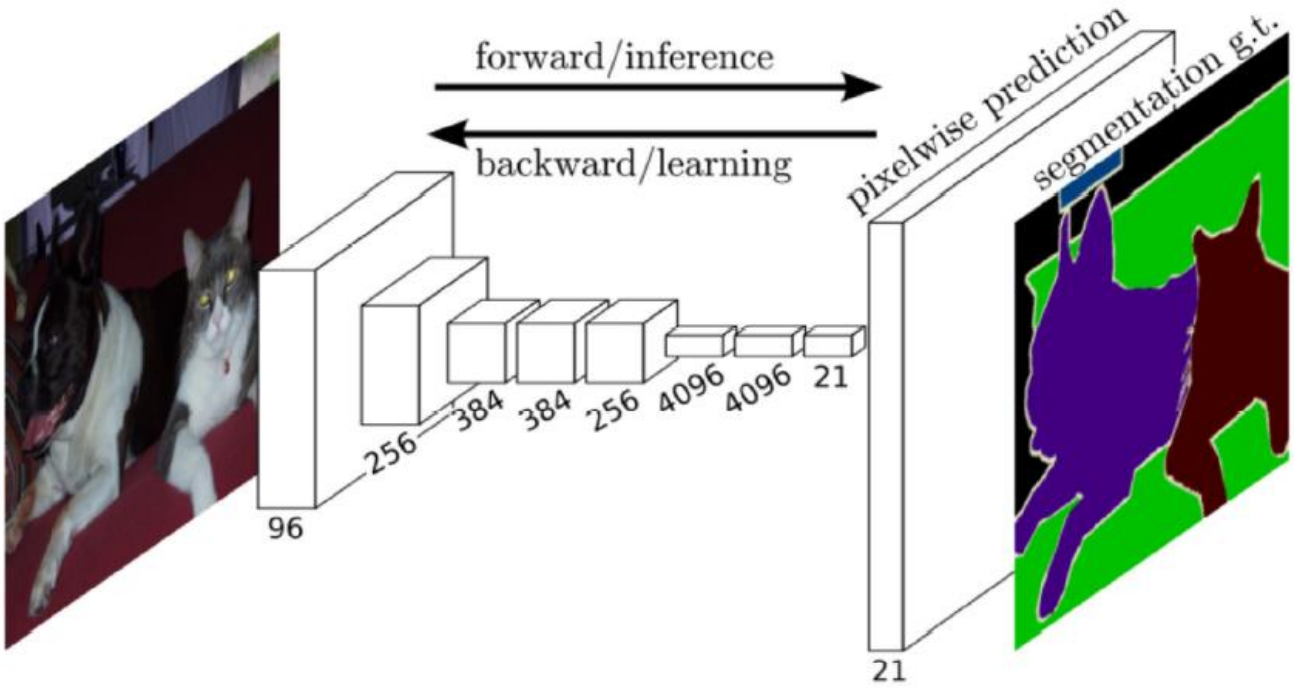


Fully Convolutional Networks

+ Significantly segmentation.

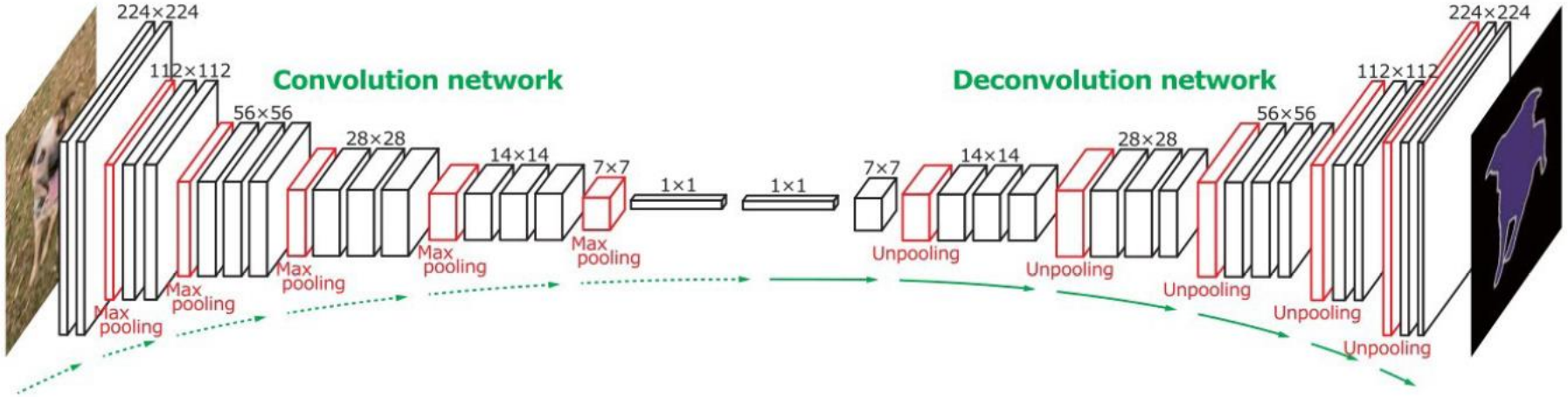
- Poor object neglected.





DeconvNet

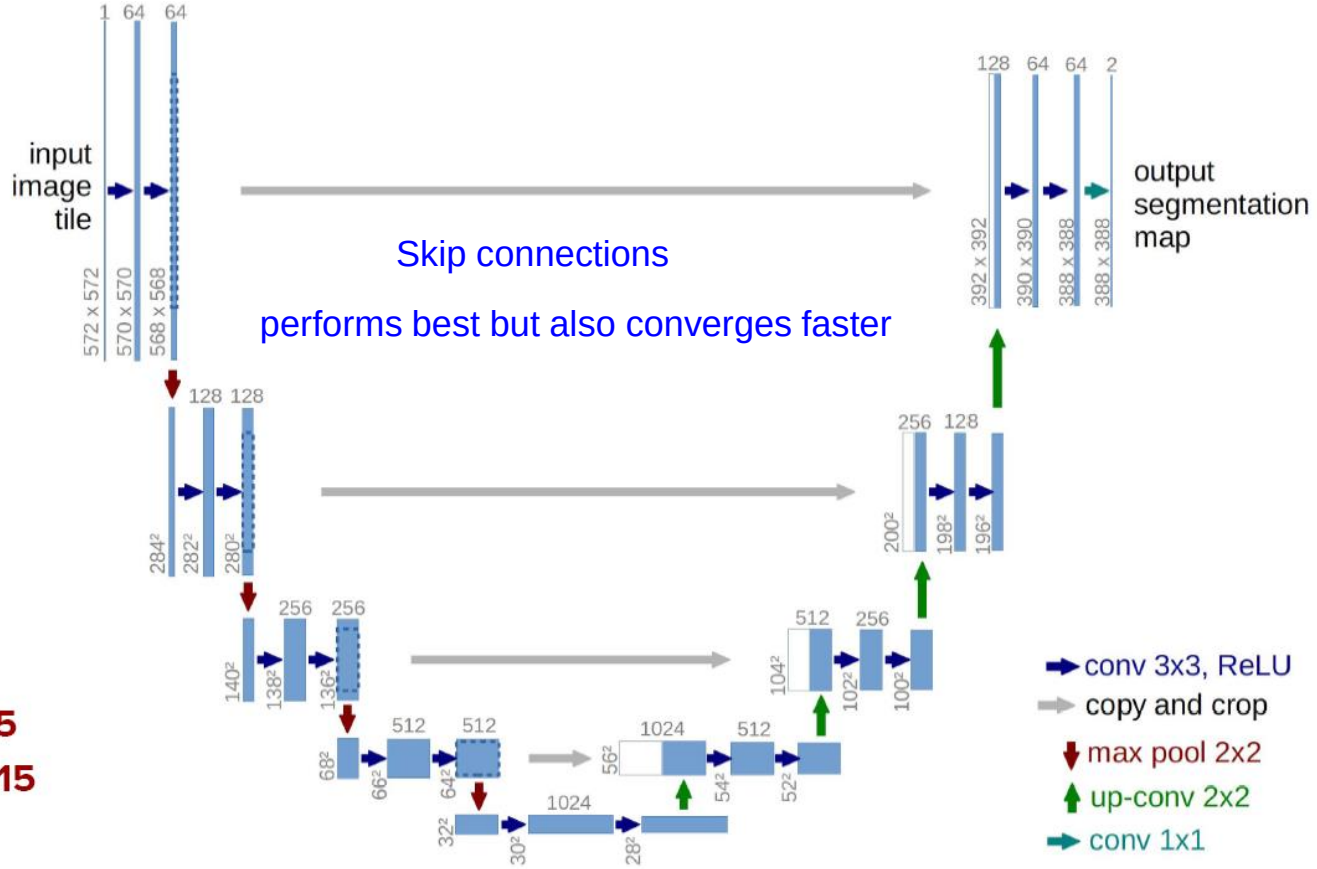
- Use image as input and output the segmentation map of the network.



Method	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbk	person	plant	sheep	sofa	train	tv	mean
EDeconvNet+CRF	**93.1**	**89.9**	**39.3**	**79.7**	**63.9**	**68.2**	**87.4**	**81.2**	**86.1**	**28.5**	**77.0**	**62.0**	**79.0**	**80.3**	**83.6**	**80.2**	**58.8**	**83.4**	**54.3**	**80.7**	**65.0**	**72.5**
DeepLab-CRF	93.1	84.4	54.5	81.5	63.6	65.9	85.1	79.1	83.4	30.7	74.1	59.8	79.0	76.1	83.2	80.8	59.7	82.2	50.4	73.1	63.7	71.6
TTI-Zoomout-16	89.8	81.9	35.1	78.2	57.4	56.5	80.5	74.0	79.8	22.4	69.6	53.7	74.0	76.0	76.6	68.8	44.3	70.2	40.2	68.9	55.3	64.4
FCN8s	91.2	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
MSRA-CFM	87.7	75.7	26.7	69.5	48.8	65.6	81.0	69.2	73.3	30.0	68.7	51.5	69.1	68.1	71.7	67.5	50.4	66.5	44.4	58.9	53.5	61.8
Hypercolumn	88.9	68.4	27.2	68.2	47.6	61.7	76.9	72.1	71.1	24.3	59.3	44.8	62.7	59.4	73.5	70.6	52.0	63.0	38.1	60.0	54.1	59.2

U-Net

- Adds convolutions in the upsampling path (“symmetric” net)
- Skip connections: concatenation of feature maps



Winner of
CAD Caries challenge ISBI 2015
Cell tracking challenge ISBI 2015