CS5112: Algorithms and Data Structures for Applications

Lecture 21: Algorithmic issues in tracking

Ramin Zabih

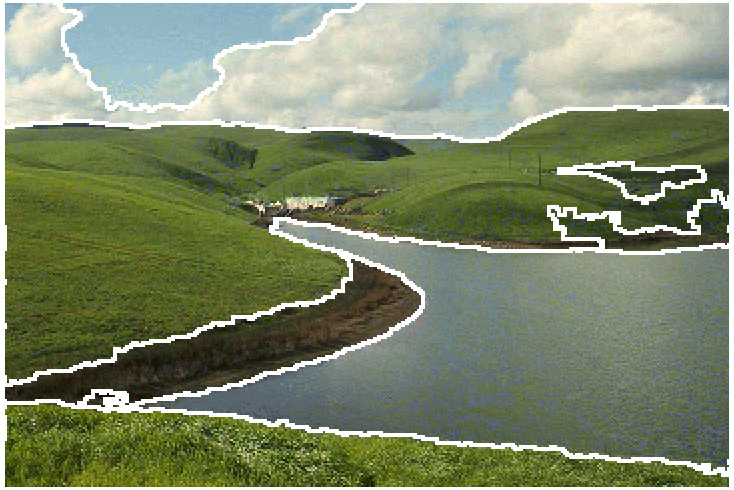
Some content from: Wikipedia/Google image search

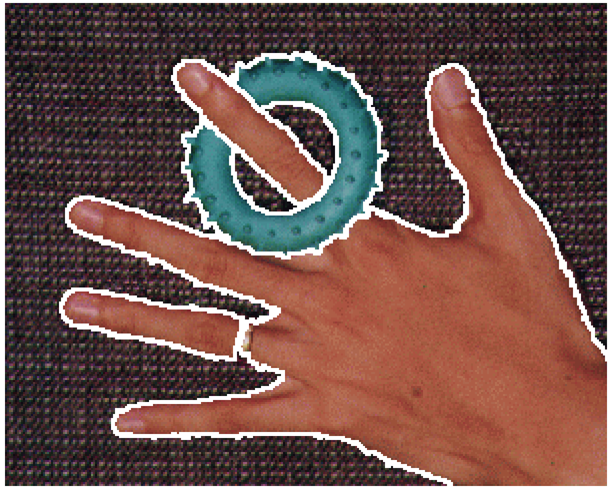
Lecture Outline

- Applying mean shift to tracking
- Expectation maximization
- RANSAC
- M-estimation
- Least squares fitting

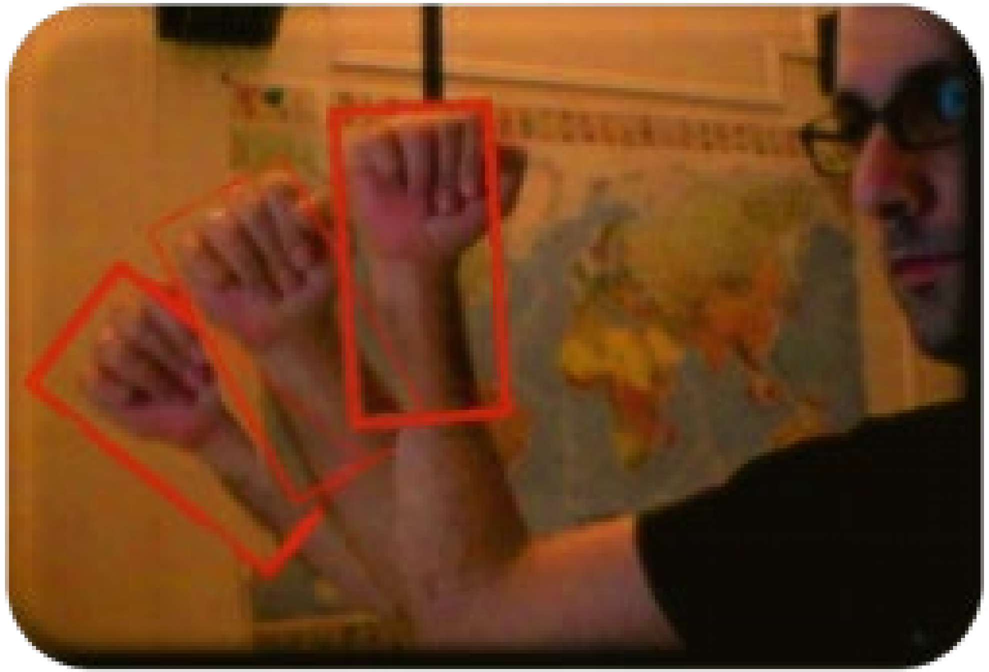
Mean shift segmentations







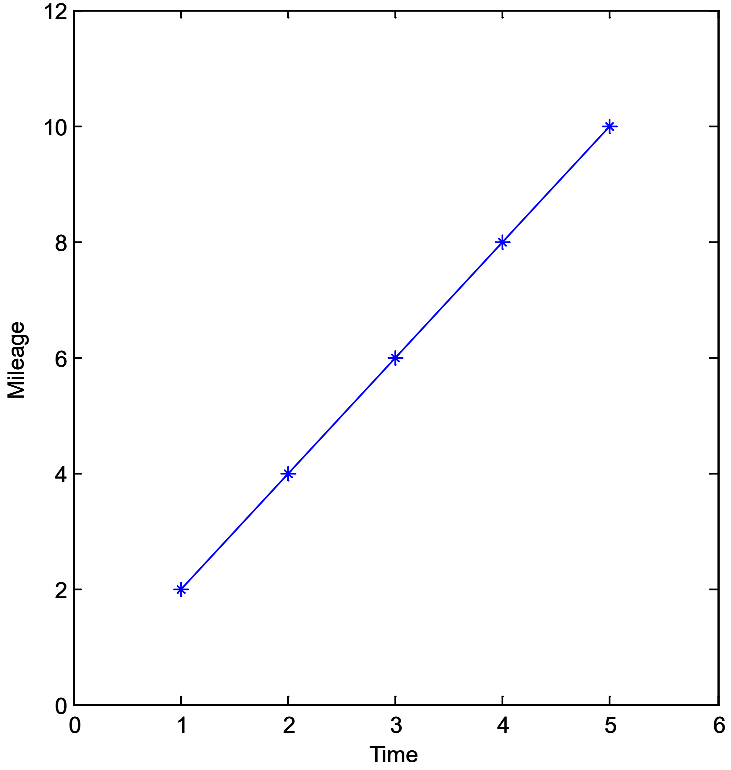
Mean shift tracking example



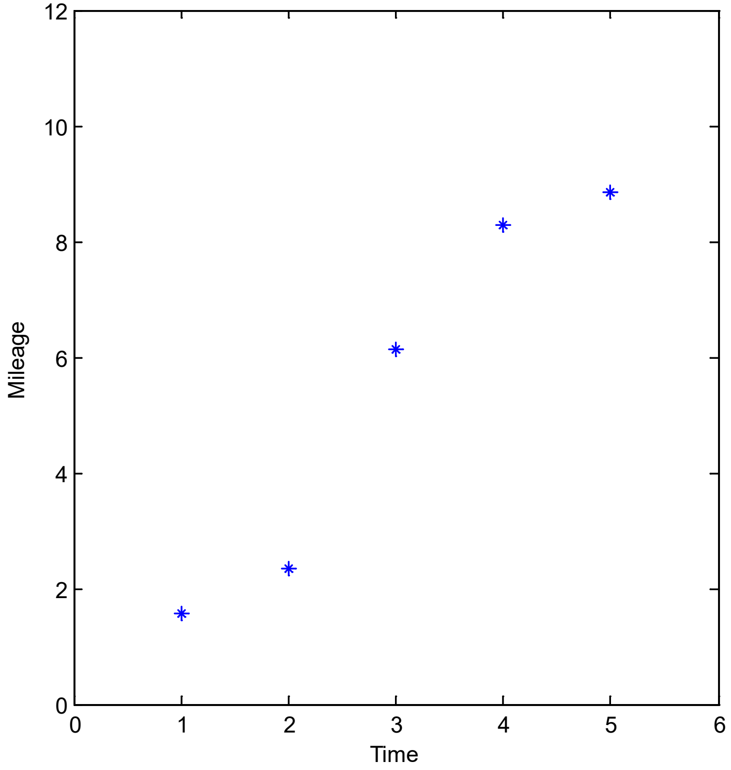
Tracking overview

- Suppose we want to figure out how fast and what direction (L-R) a robot is moving
- We can just run segmentation on consecutive frames
  - Not too much changes in 1/30 of a second
  - Usually you can 'carry over' some information from previous frame

The ideal robot



The real (lying) robot

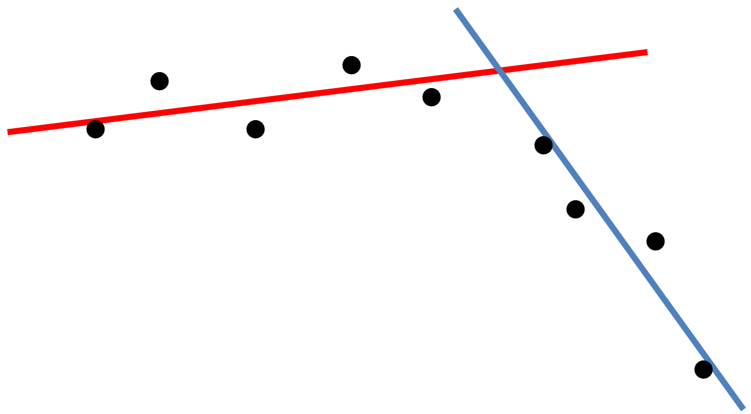


Real problem is much worse

- Some of the time we will track a different object
- This gives us outliers (from other object)
  - Versus inliers (from the object we want to track)

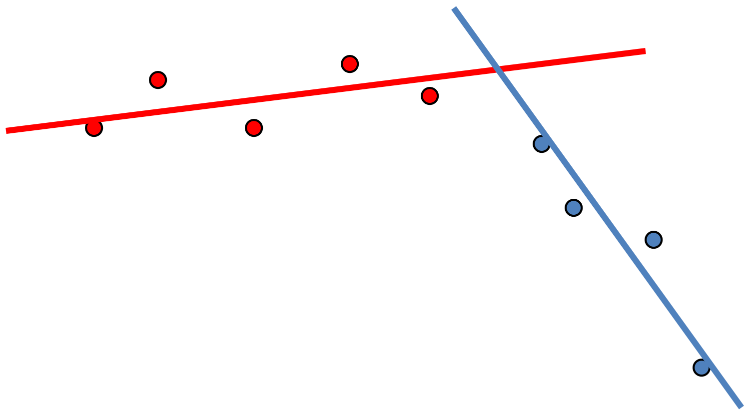
Simplified tracking problem – Line fitting

- Goal: To group a bunch of points into two “best-fit” line segments



“Chicken-egg problem”

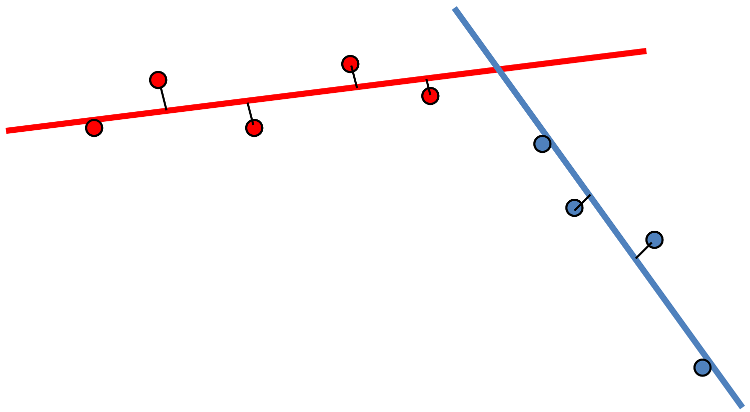
- If we knew which line each point belonged to, we could compute the best-fit lines.



CORNELL TECH

Chicken-egg problem

- If we knew what the two best-fit lines were, we could find out which line each point belonged to.

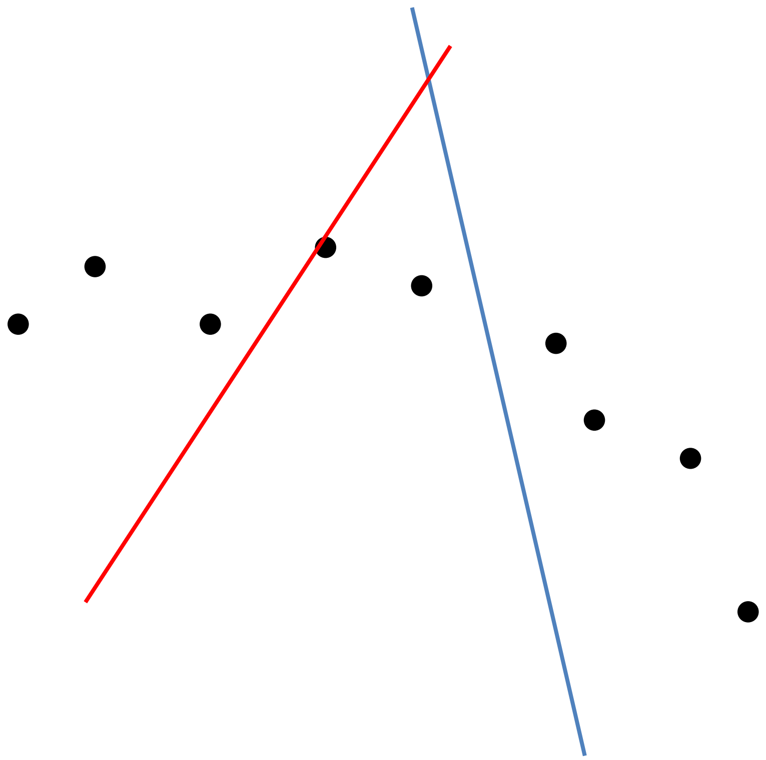


CORNELL TECH

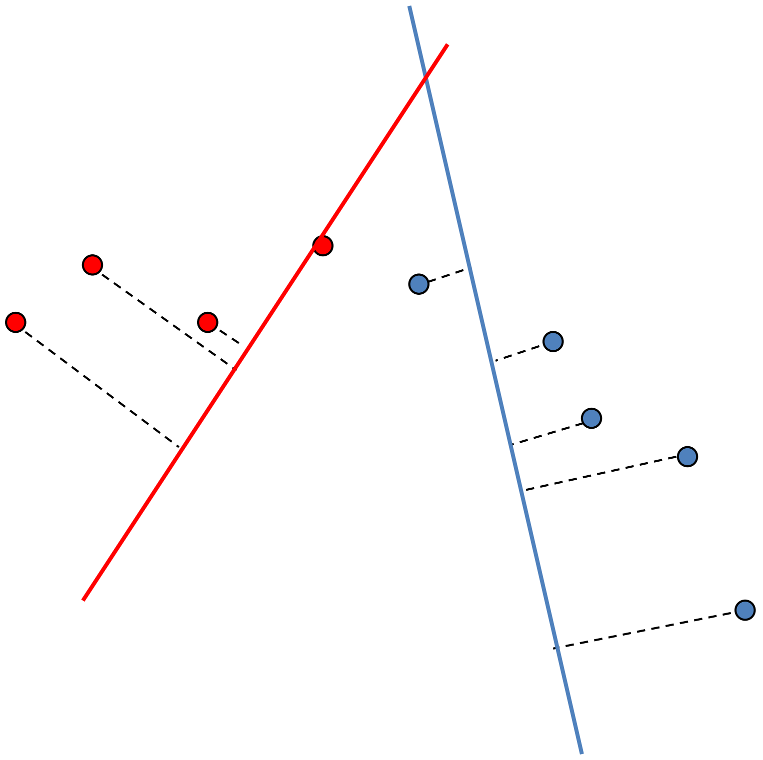
Expectation-Maximization (EM)

- Initialize: Make random guess for lines
- Repeat:
  - Find the line closest to each point and group into two sets. (Expectation Step)
  - Find the best-fit lines to the two sets (Maximization Step)
  - Iterate until convergence
- The algorithm is guaranteed to converge to some local optima

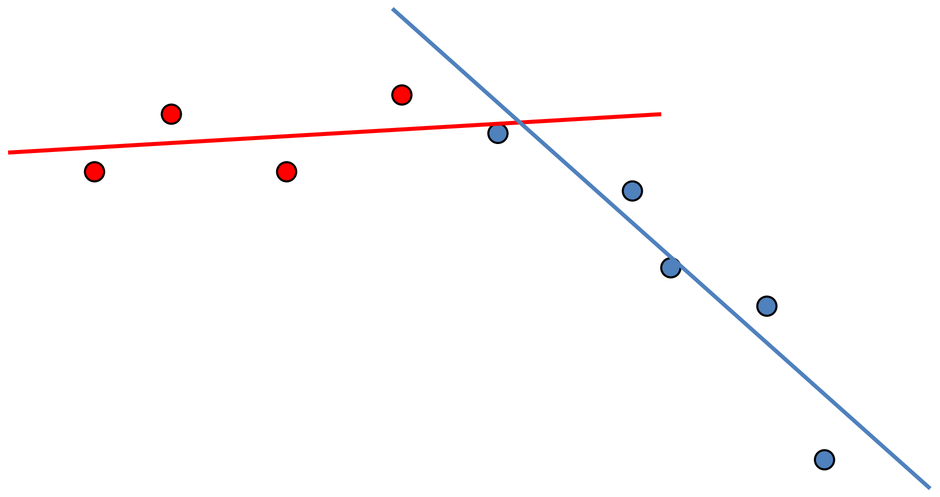
Example:



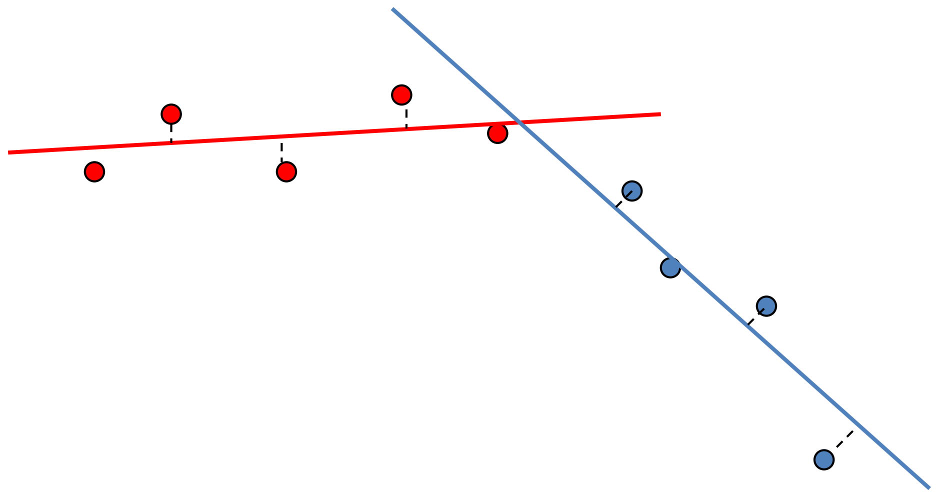
Example:



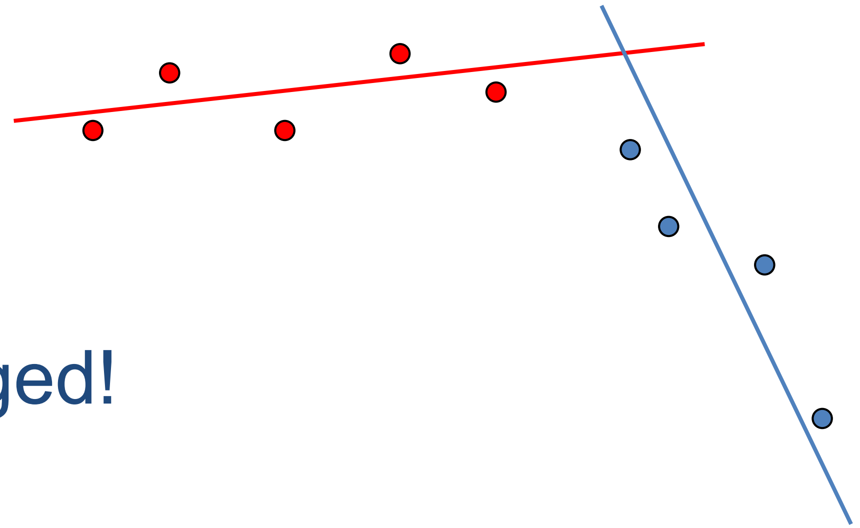
Example:



Example:



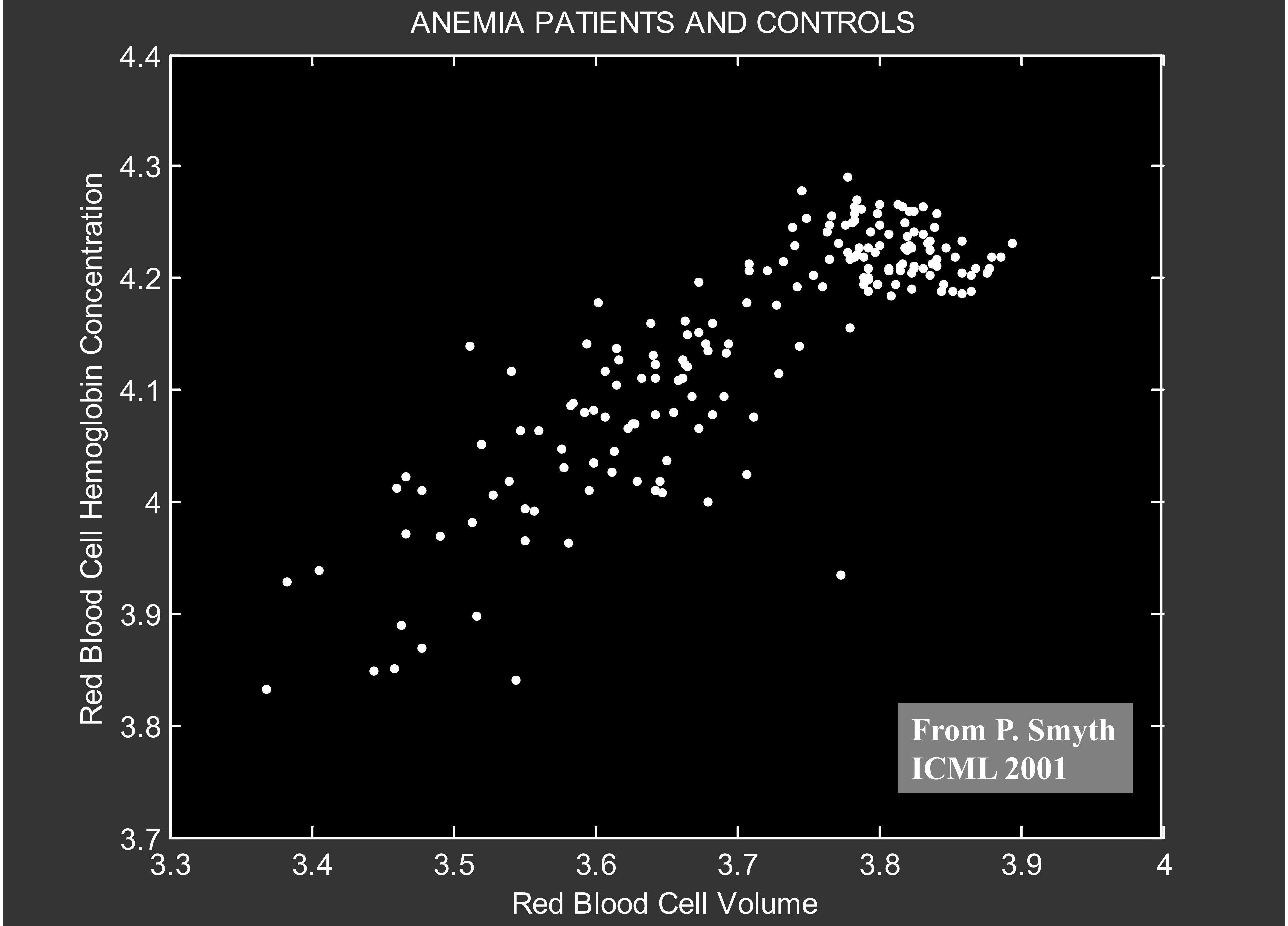
Example:



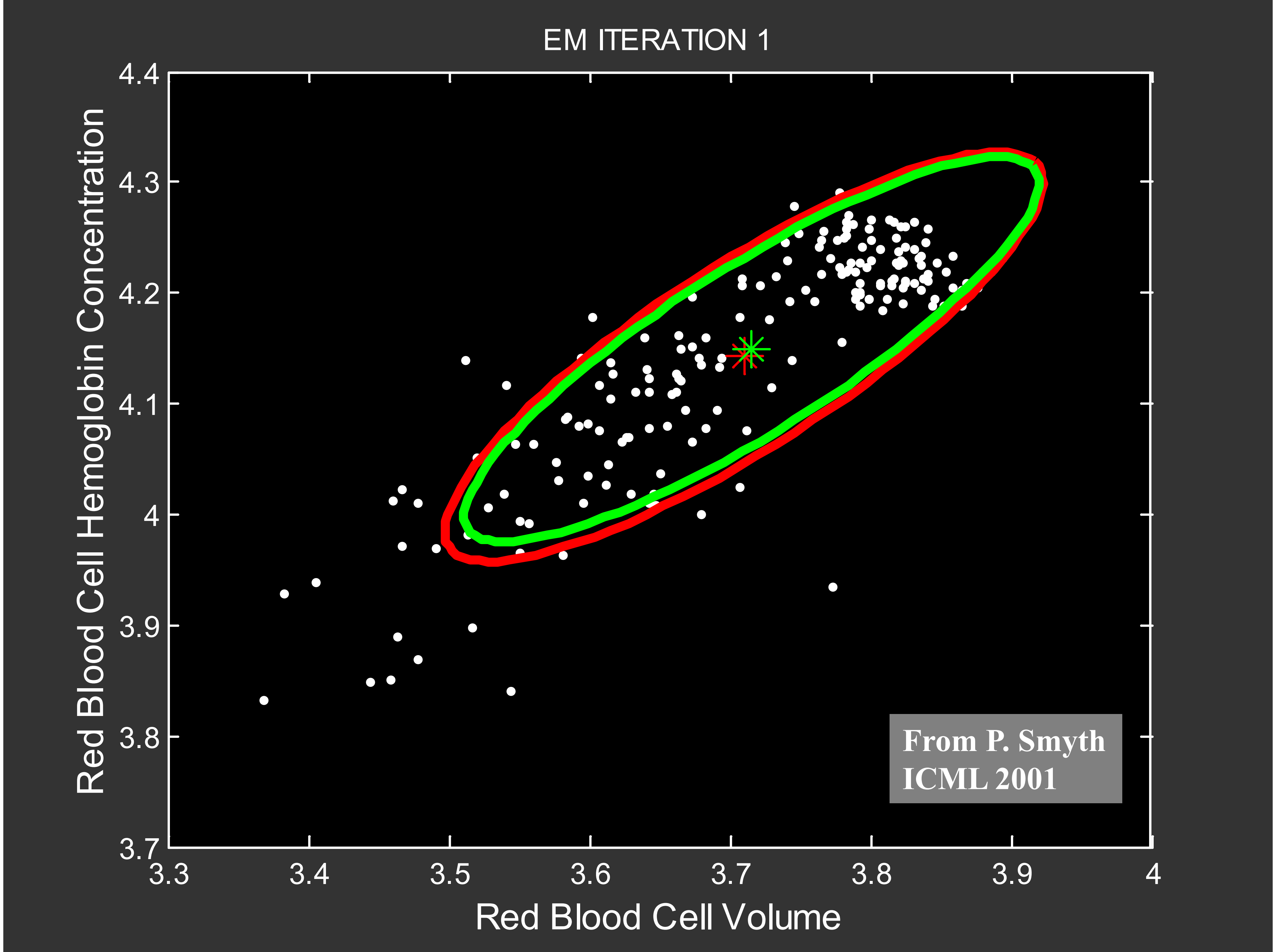
Converged!

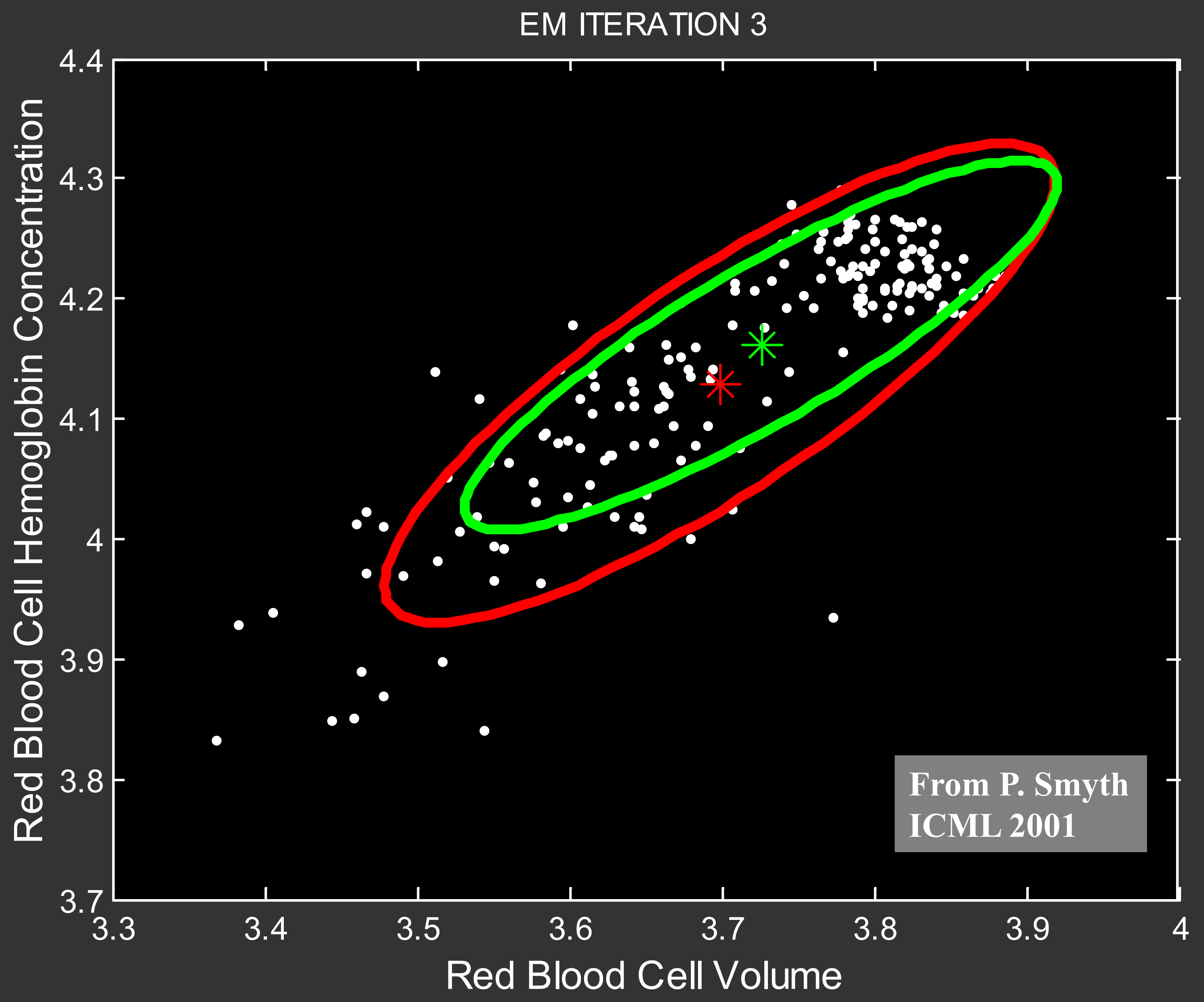
Expectation maximization

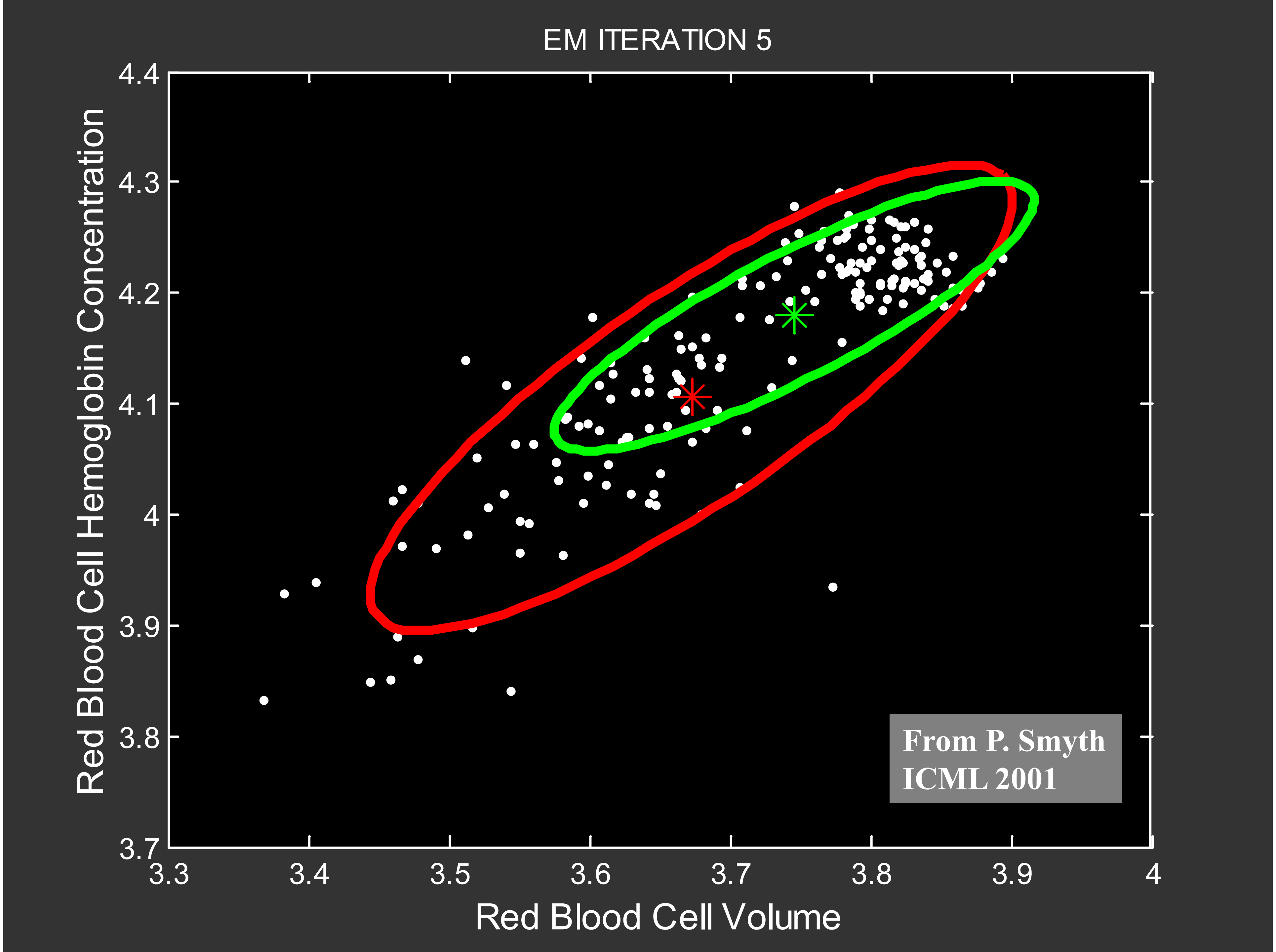
- Standard use case is to separate two gaussians (MOG)
- If we knew which data is water and which is beer, we could compute the mean and variance separately
- If we know the mean and variance were for beer and water, we could figure out which data is water and which is beer
- But we don’t know anything!
- So, just like in k-means, we guess and iterate

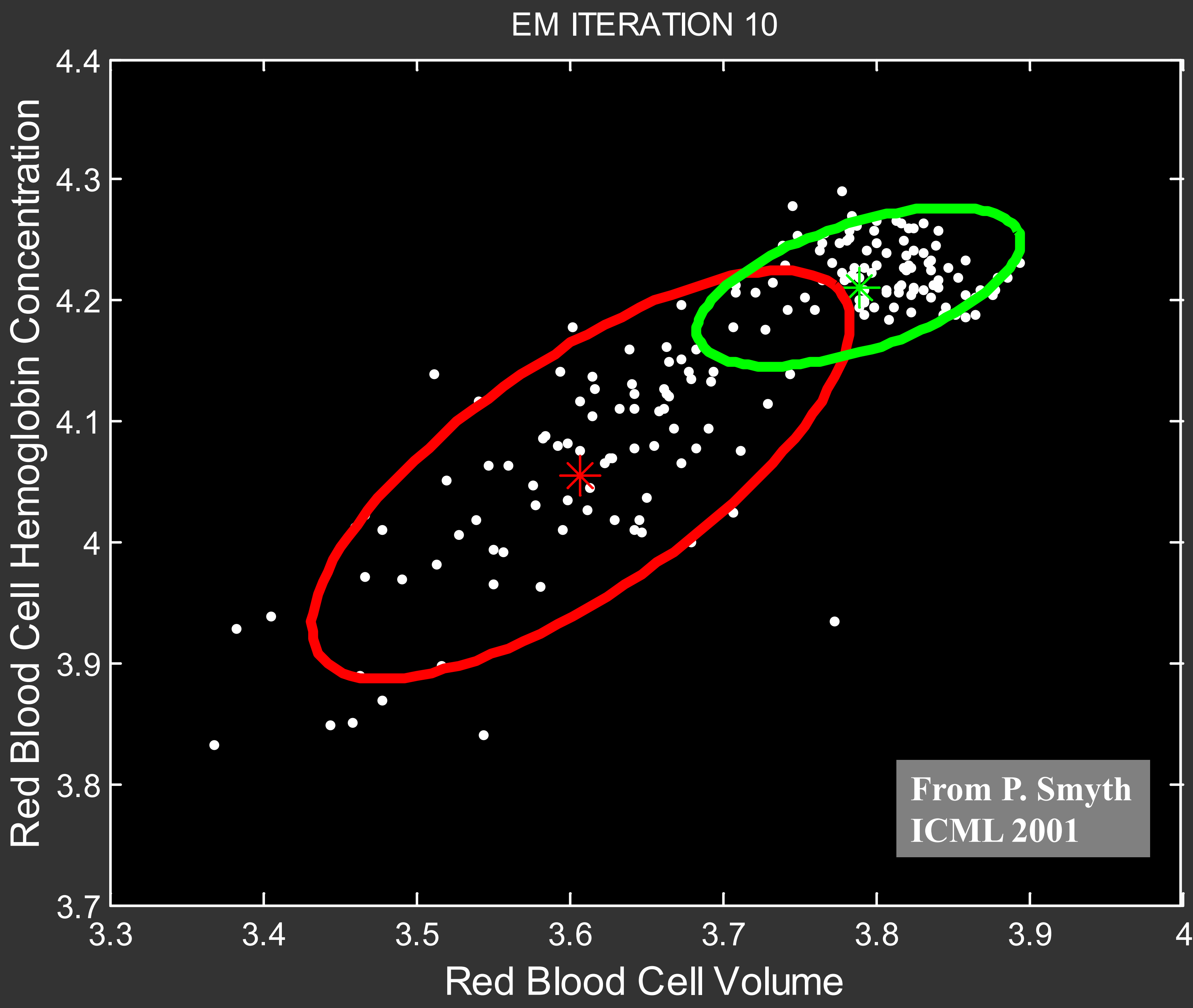


ANEMIA PATIENTS AND CONTROLS
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001

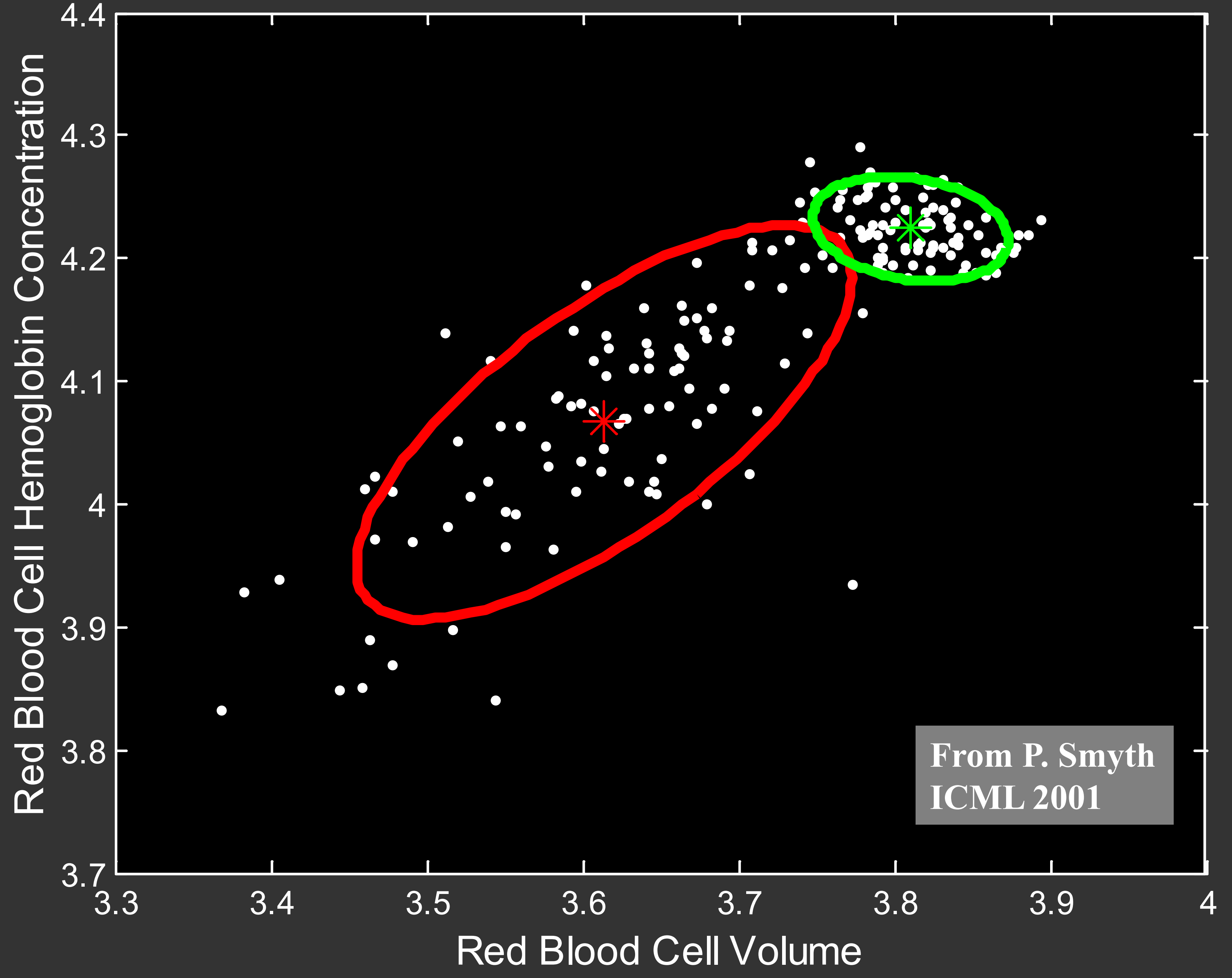

EM ITERATION 1
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
From P. Smyth
ICML 2001

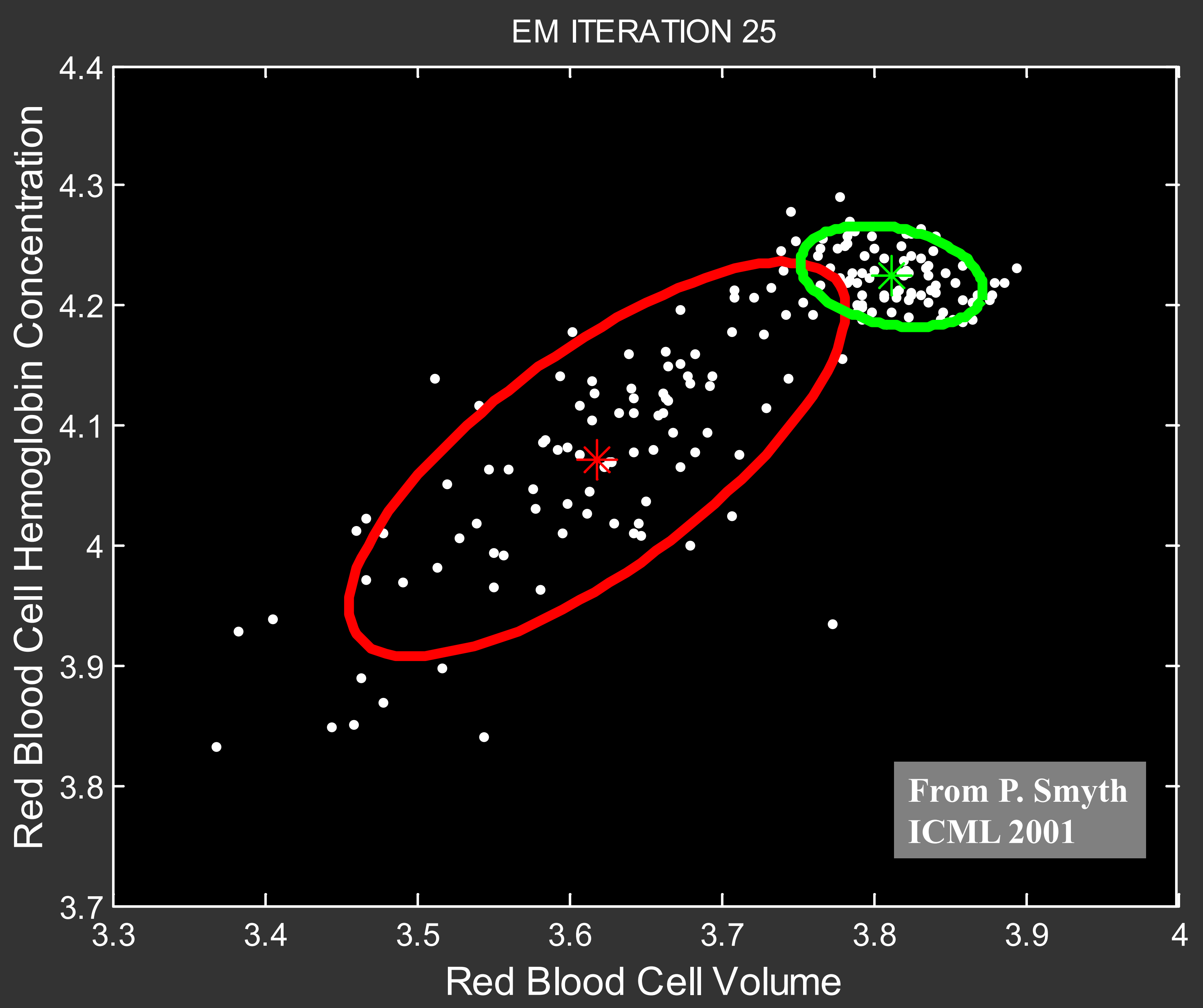

EM ITERATION 3
Red Blood Cell Hemoglobin Concentration
4.4
4.3
4.2
4.1
4
3.9
3.8
3.7
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
Red Blood Cell Volume
From P. Smyth
ICML 2001


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
From P. Smyth
ICML 2001


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
From P. Smyth
ICML 2001

EM ITERATION 15





EM ITERATION 25
Red Blood Cell Hemoglobin Concentration
Red Blood Cell Volume
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4
3.7
3.8
3.9
4
4.1
4.2
4.3
4.4
From P. Smyth
ICML 2001

RANSAC algorithm

Run k times:

How many times?

(1) draw n samples randomly

How big?
Smaller is better

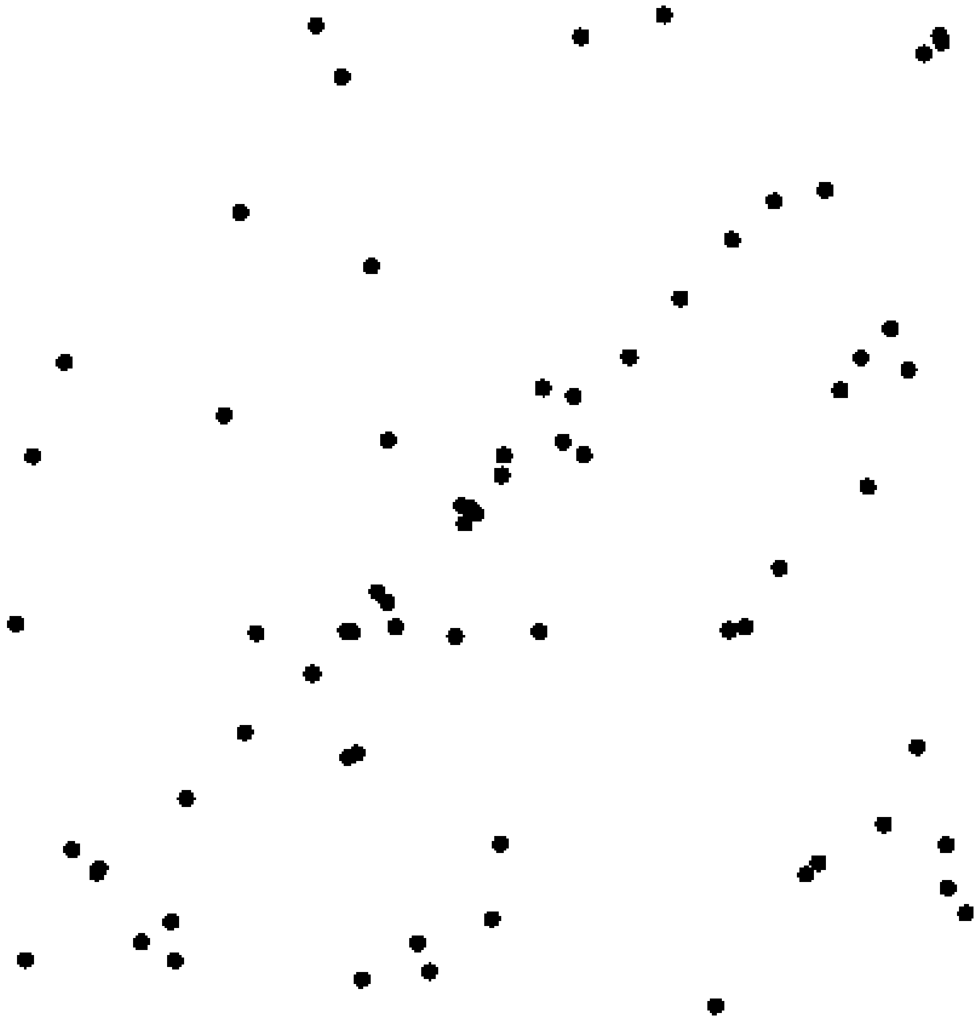
(2) fit parameters Θ with these n samples

(3) for each of other N-n points, calculate its distance to the fitted model, count the number of inlier points, c

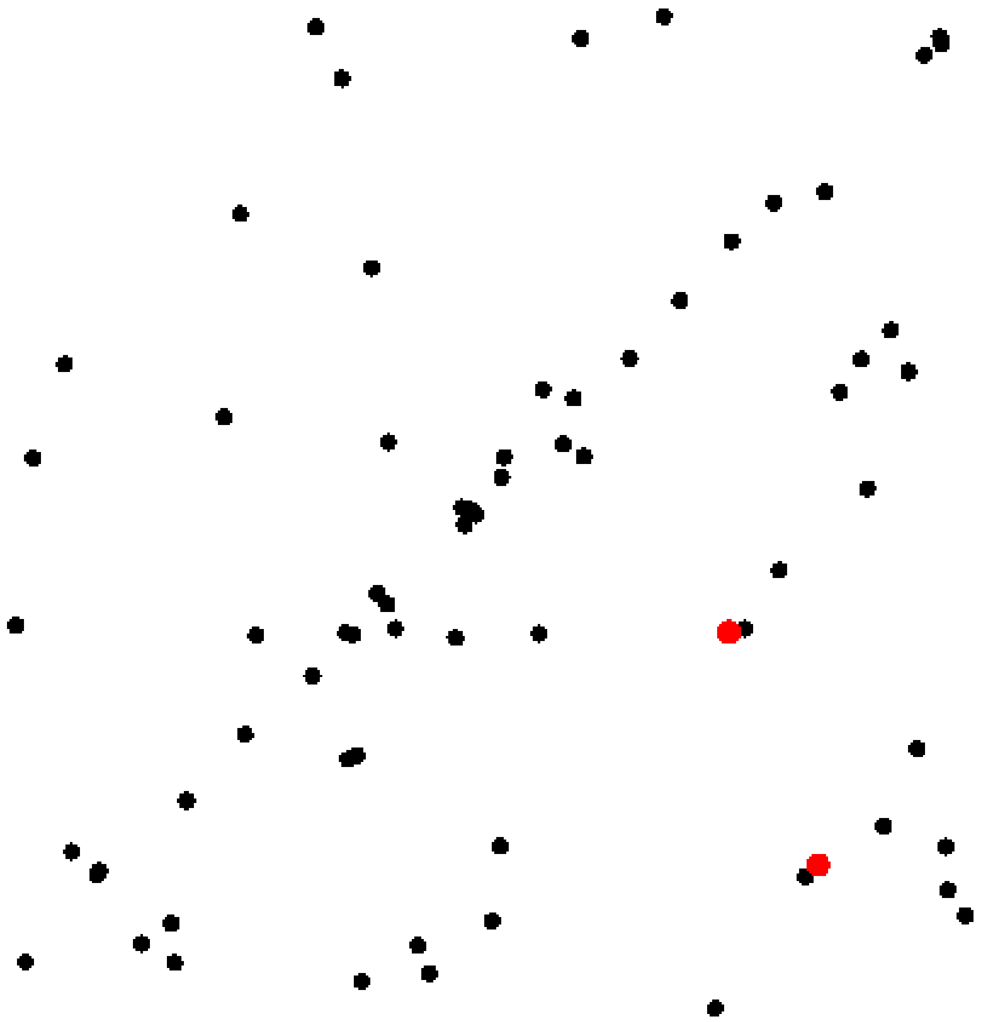
Output Θ with the largest c

How to define?
Depends on the problem.

Example: line fitting

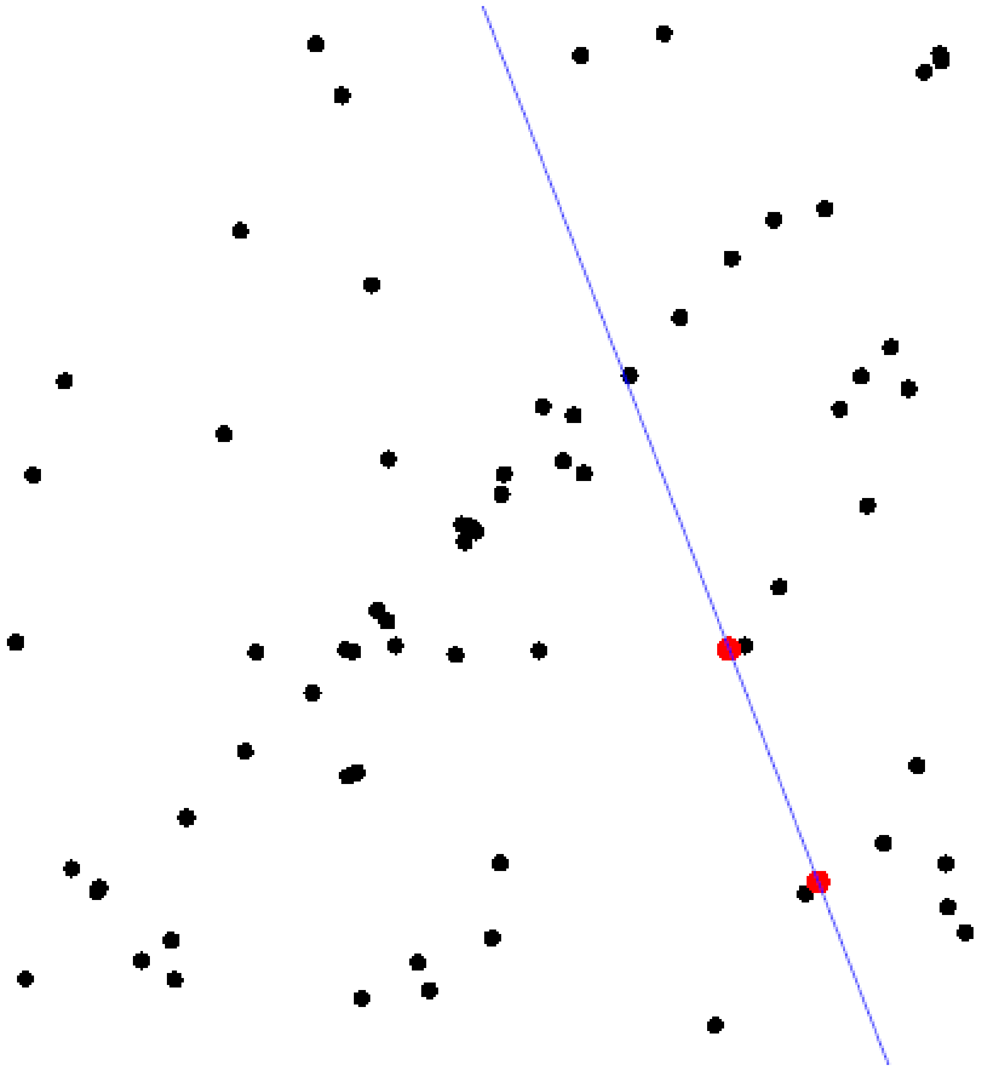


Example: line fitting

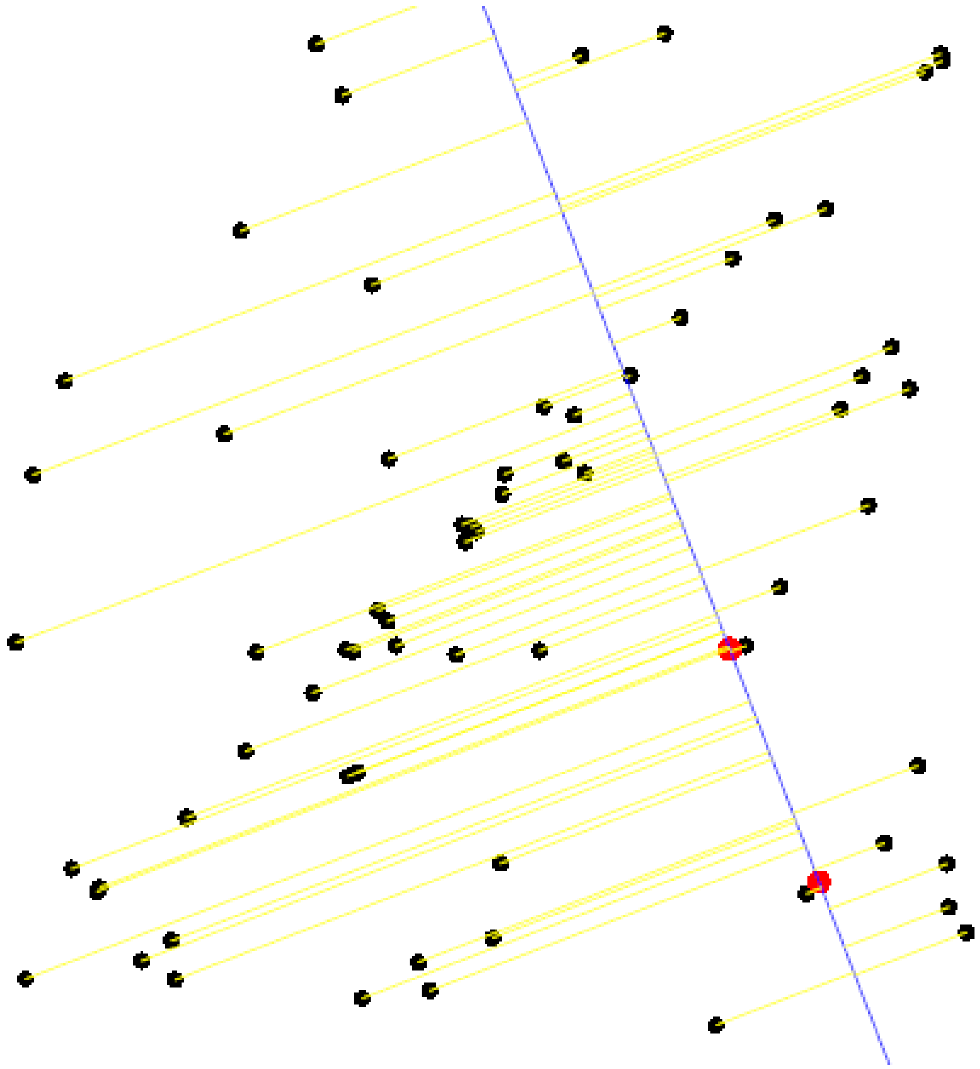


$n=2$

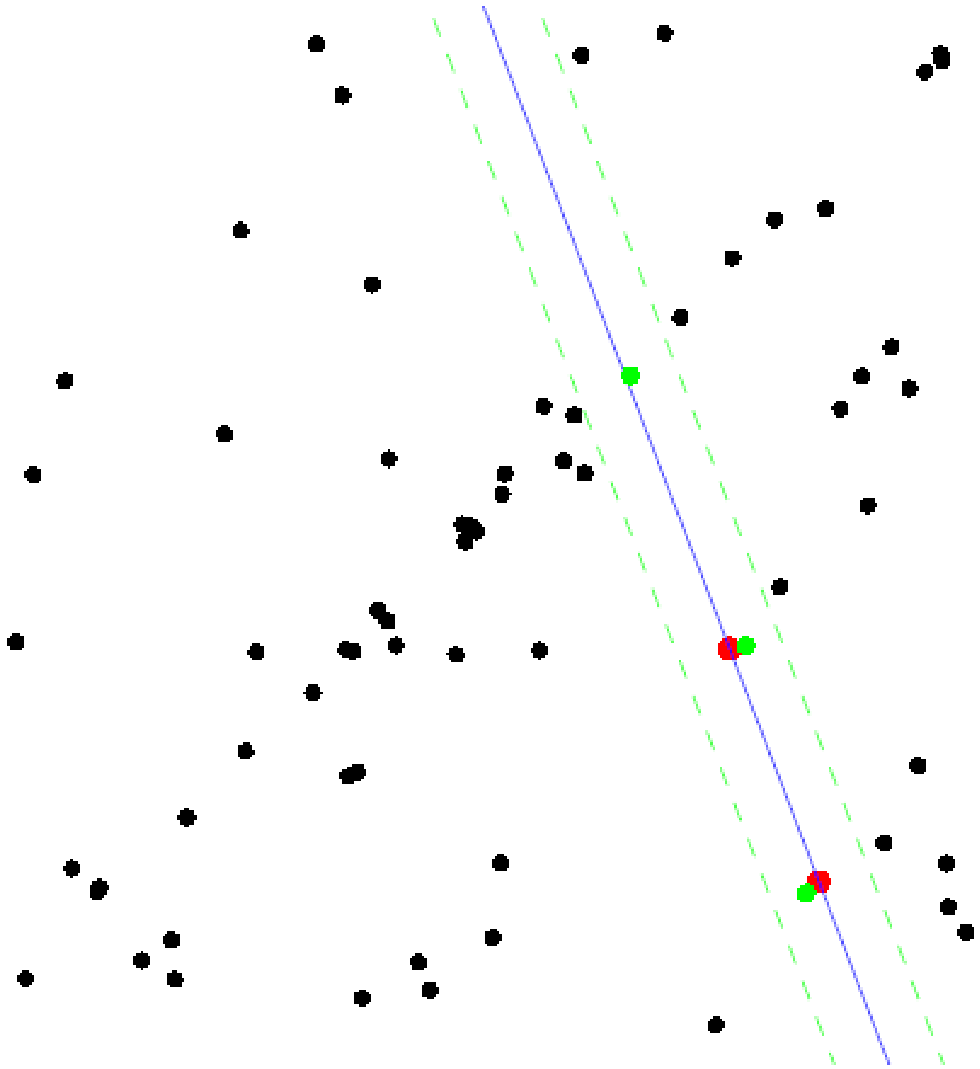
Model fitting



Measure distances

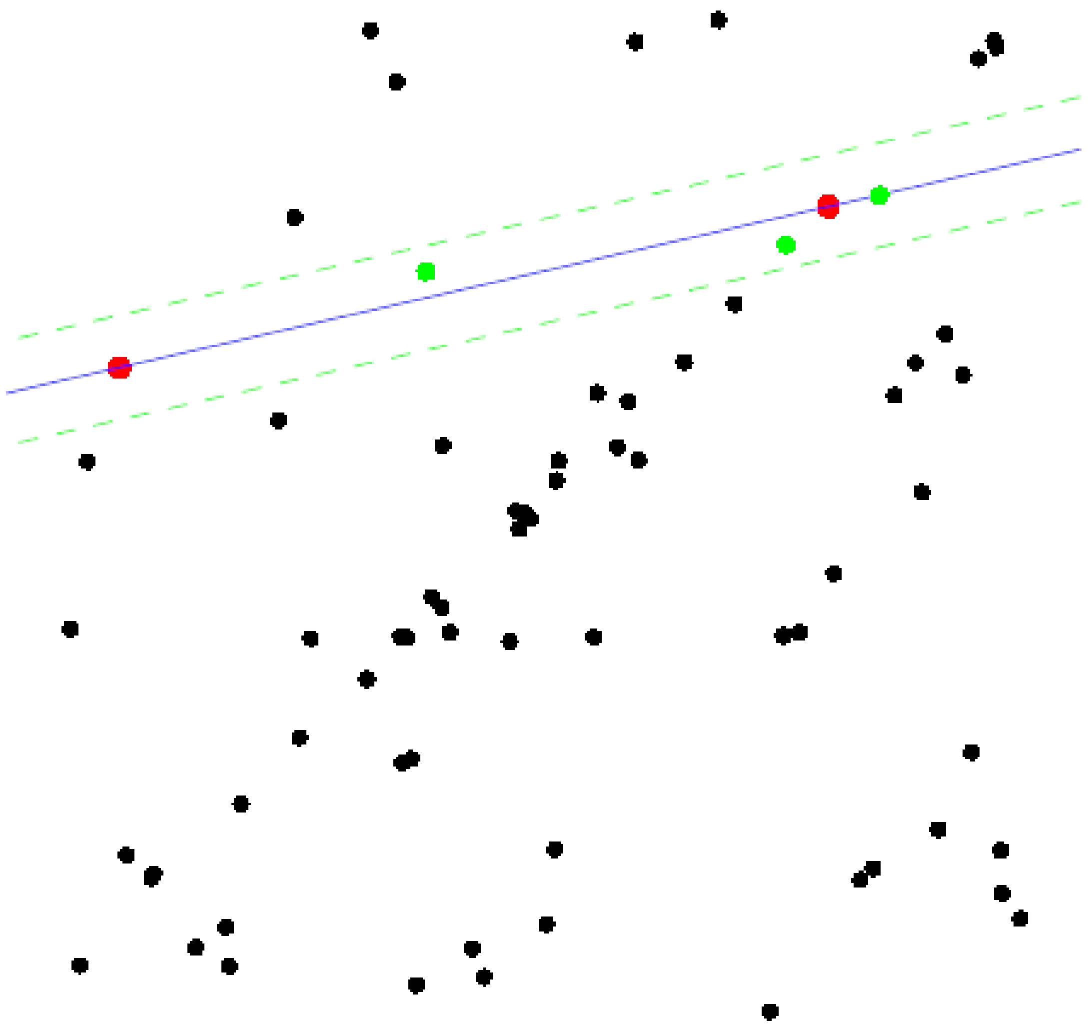


Count inliers



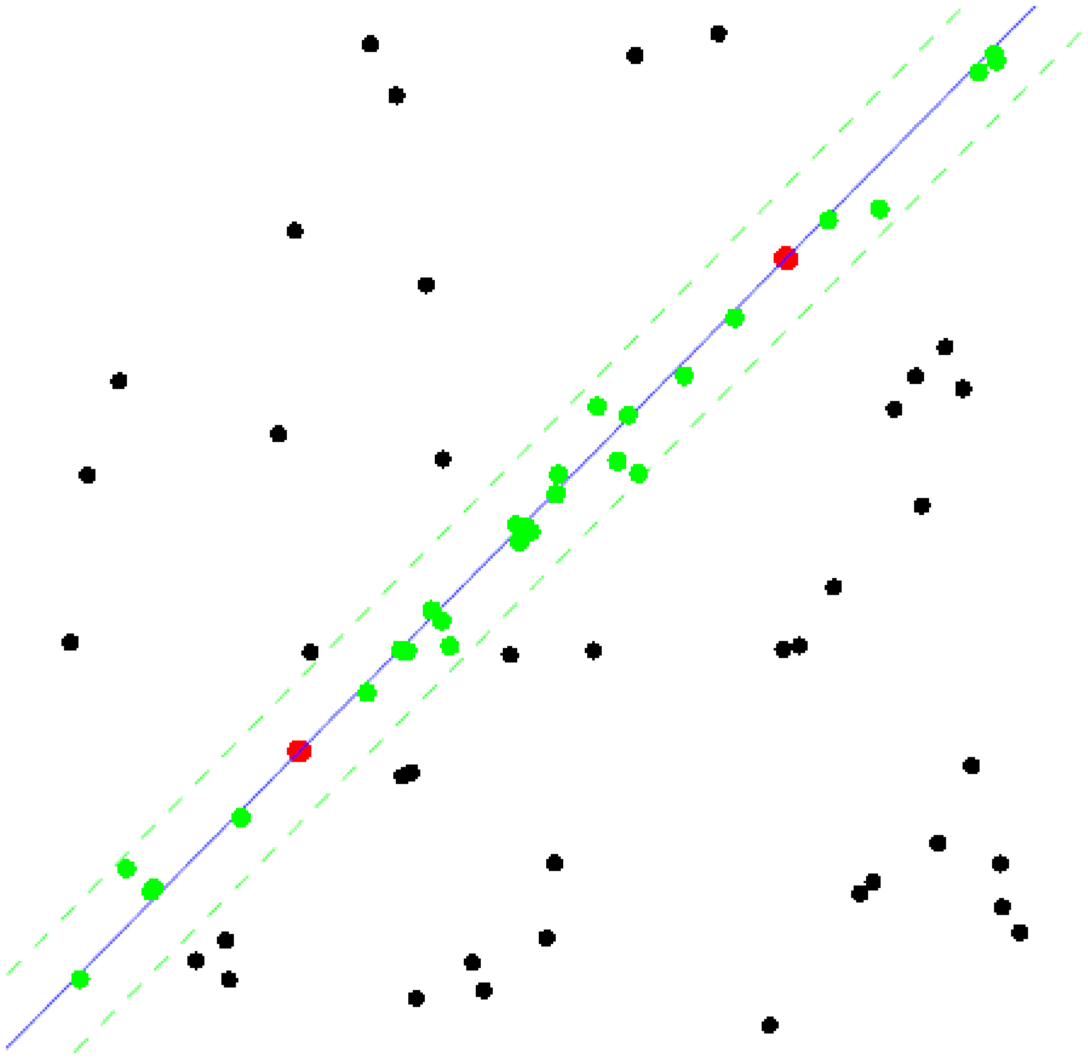
$c=3$

Another trial



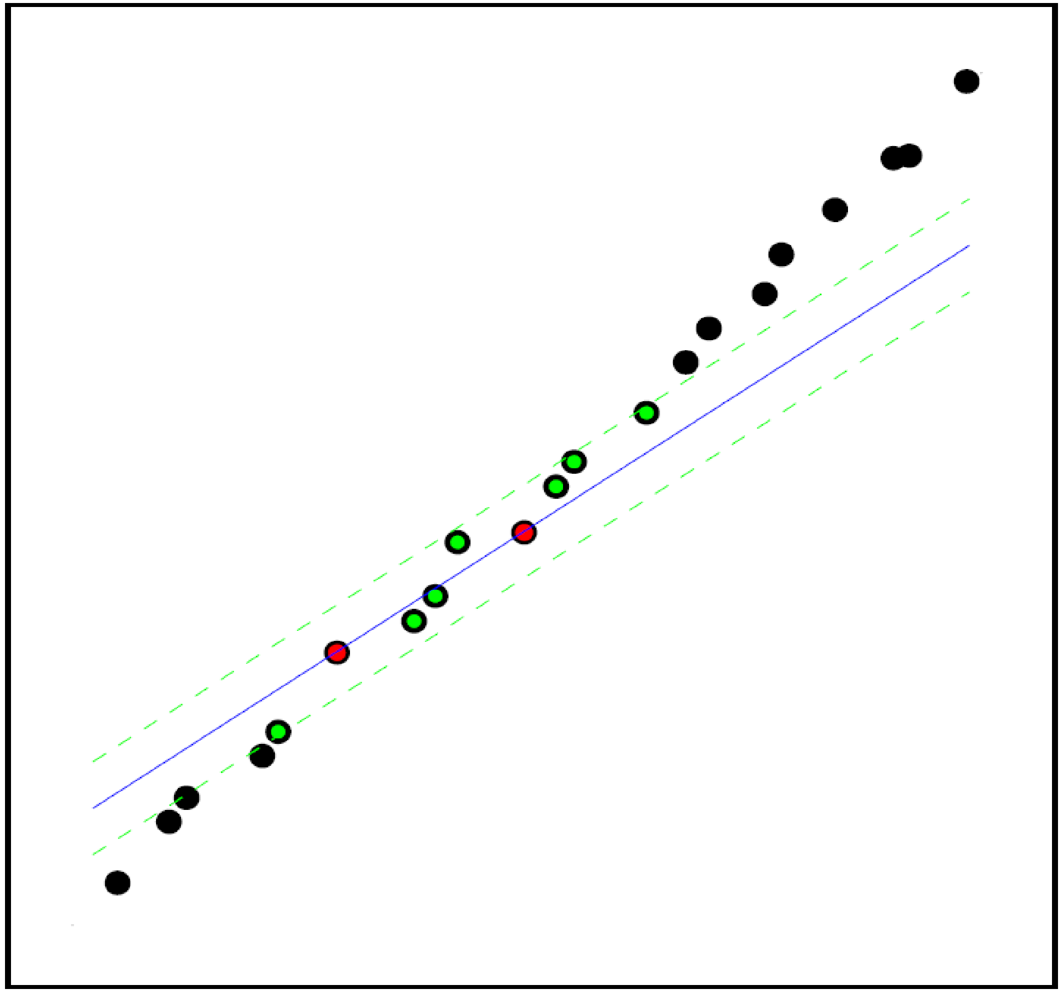
$c=3$

The best model



$c=15$

RANSAC failure mode



Not every all-inlier sample gives a model consistent with all inliers

General model fitting problem

- We have some data points (x_i, y_i) and some possible models, each of which has some parameters θ
  - Example: line fitting, $y = mx + b$
  - A model predicts $M(x; \theta)$
- What set of parameters gives the best fit to the data?
- For a particular θ, the residuals are

$$r_i = y_i - M(x_i; \theta)$$

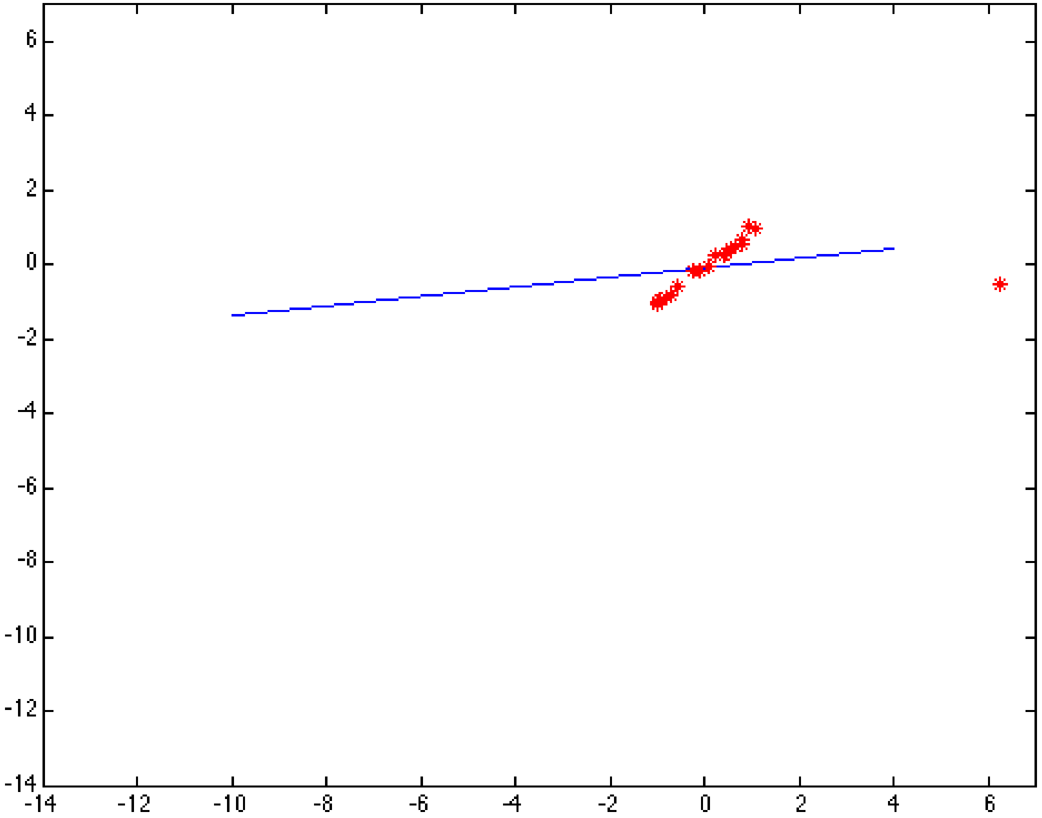
Least squares fit

- The least squares fit says that the best fit minimizes

$$\sum_i r_i^2$$

  – Sum of the squared residuals
- At the correct selection of points, what are the residuals?
  – They are generally small and gaussian

1 bad point can ruin your whole line



Example c/o Kim Boyer, OSU

Problem is subtle

- You can’t simply do an LS fit and then declare the worst-fitting point to be “bad”
  - There are examples where the bad data is fit better than the good data
- Robust statistics addresses this problem
  - A robust fitting method tolerates outliers
    - Obviously, LS is not robust
  - Note that in vision, the term “robust” sometimes simply means “good”

Robust model fitting

- There are two problems with the LS fit
  - We square the residuals
  - We sum up these squares
- The main approaches in robust statistics address each of these problems
  - The problem with squaring the residuals is that the squared values get too large

M-estimation

- Suppose that our measure of goodness of fit is $\sum_i \rho(r_i)$, where

$$\rho(r_i) = \min(r_i^2, s^2)$$

  – Here, s is a scale parameter
  – All residuals worse than s count like s
- The scale parameter essentially controls the boundary between inliers and outliers
  – We expect outliers to have residuals larger than s, but not inliers
  – How do we pick s?

Computing a robust fit

- It's possible to perform M-estimation fairly efficiently using a variant of least squares
- Think of Az, where A is a matrix and z is a vector, as a linear combination of the columns of A, weighted by elements of z
- Example for the model $y = mx + b$ and data (x_i, y_i)
  - A has $(x_i \; 1)$ rows, one per data point
  - $z = (m \; b)^T$

$$Az = (y_1, \ldots, y_n)^T = d$$

CORNELL TECH

Computing a LS fit

- If we consider all possible choices of z we span a subspace.
- The solution to $Az = d$ is the “coordinates” of d in terms of the columns of A
- What if d isn’t in the subspace?
- We can ask for the point in the subspace that is as close as possible to d (the least squares fit)

Solving least squares

- The least squares solution to $Az \ = d$ is
$$\arg\min_z ||d - Az||$$
- An elegant result, due to Gauss, is that the solution to this is the pseudoinverse $(A^T A)^{-1} A^T d$
  – Easy to re-derive: $A^T A$ is square!
- If we weight each residual by w_i we get
$$\arg\min_z \left||W(d - Az)|\right| = (A^T W^2 A)^{-1} A^T W d$$
  – Here, W is a diagonal matrix of w_i

Iterative reweighted least squares

- IRLS algorithm
  - Start with all weights being 1
  - Compute least squares fit and residuals
  - Adjust W to reduce the weighting of the large residuals
  - Re-fit and repeat