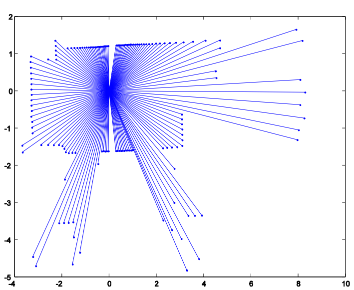
CS-8804 – Machine Learning for Robotics

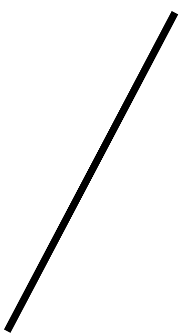
Finding models in noisy data

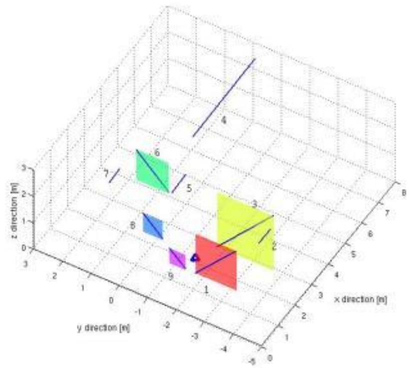
Cédric Pradalier

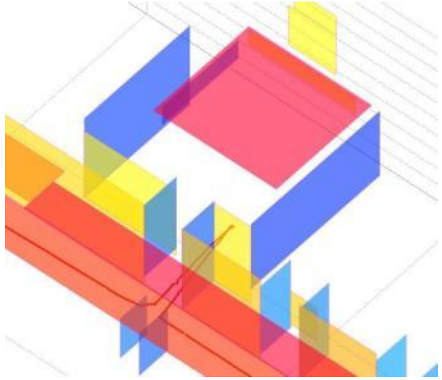
Features: Motivation

- Why Features:
  - Raw data: huge amount of data to be stored
  - Compact features require less storage (e.g. Lines, planes)
  - Provides rich and accurate information
  - Basis for high level features (e.g. more abstract features, objects)





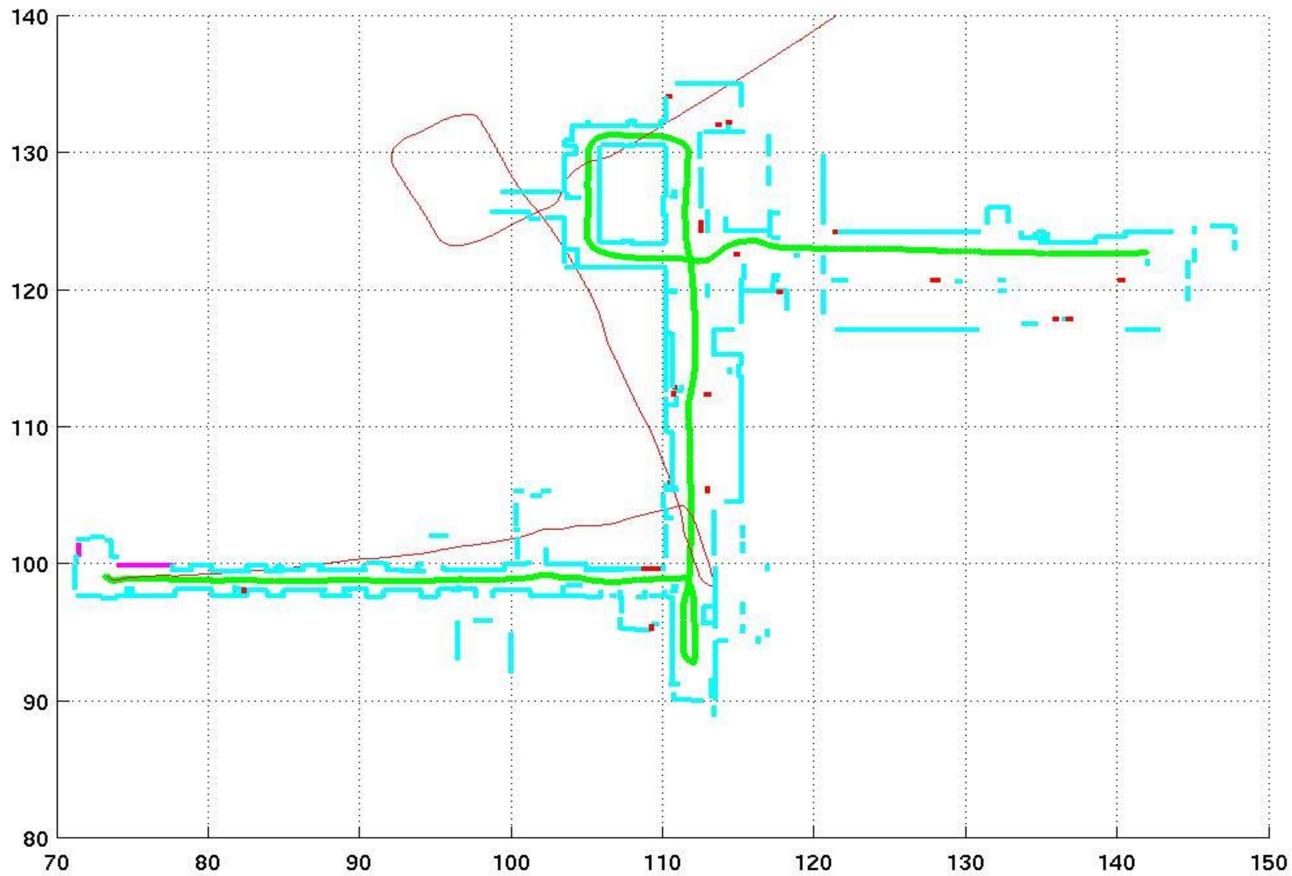




Line Extraction: Split-and-Merge

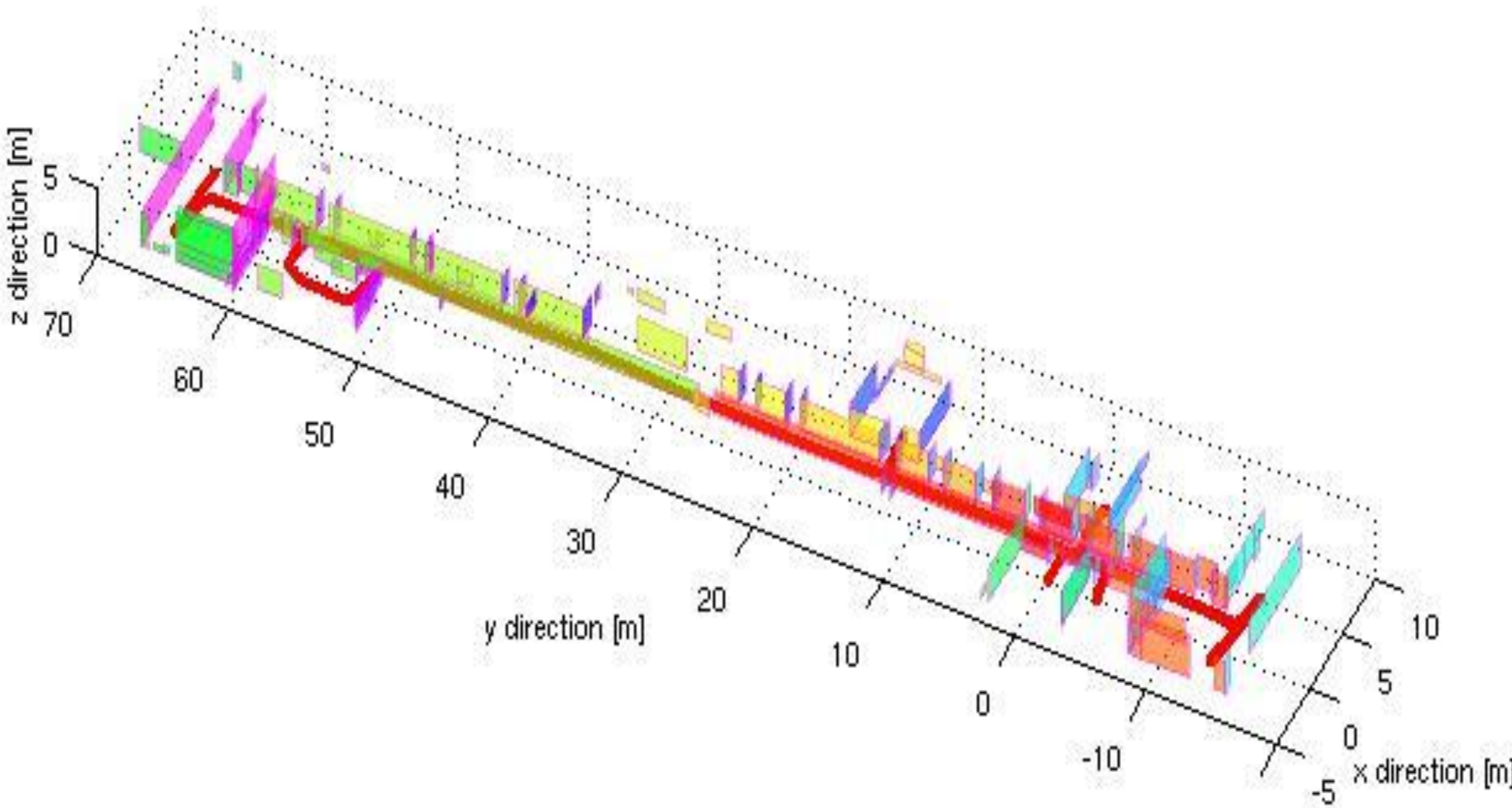
Line Extraction: Motivation

- Map of the ASL hallway built using line segments



Line Extraction: Motivation

- Map of the ASL hallway built using orthogonal planes constructed from line segments



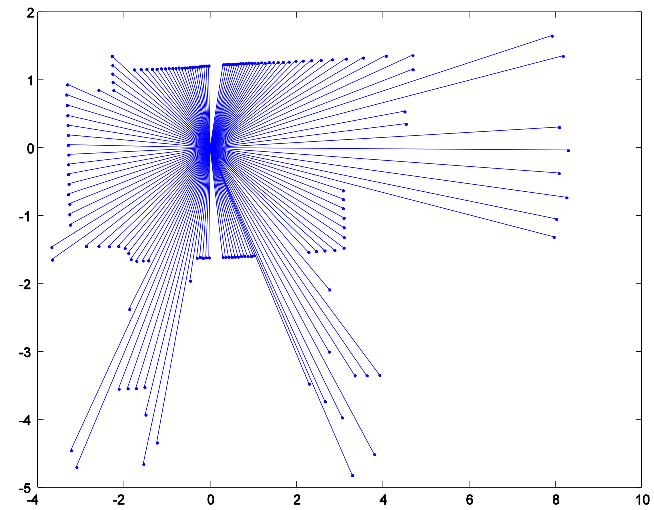
Line Extraction: Motivation

- Why laser scanner:
  - Dense and accurate range measurements
  - High sampling rate, high angular resolution
  - Good range distance and resolution.

- Why line segment:
  - The simplest geometric primitive
  - Compact, requires less storage
  - Provides rich and accurate information
  - Represents most office-like environment.

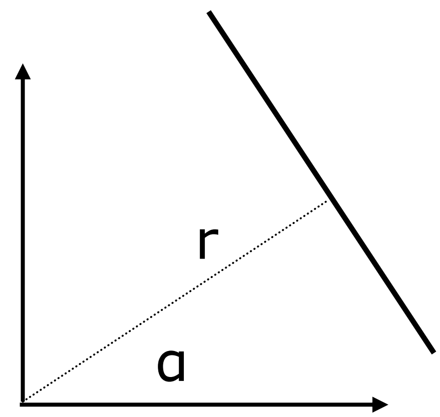
Line Extraction: The Problem

- Scan point in polar form: (ρ_i, θ_i)



- Assumptions:
  - Gaussian noise with (0, σ) for ρ
  - Negligible angular uncertainty

- Line model in polar form:
  - $x \cos \alpha + y \sin \alpha = r$
  - $-\pi < \alpha <= \pi$
  - $r >= 0$



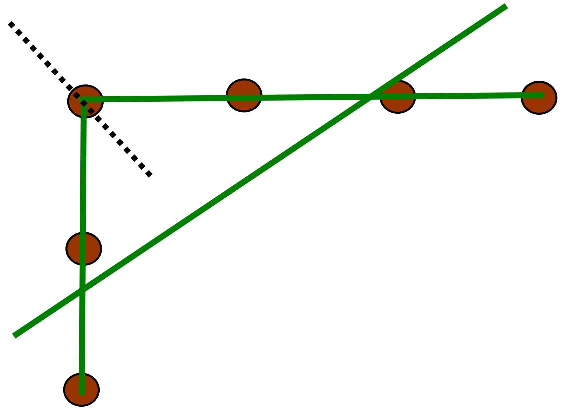
Line Extraction: The Problem (2)

- Three main problems:
  - How many lines ?
  - Which points belong to which line ?
    - This problem is called SEGMENTATION
  - Given points that belong to a line, how to estimate the line parameters ?
    - This problem is called LINE FITTING

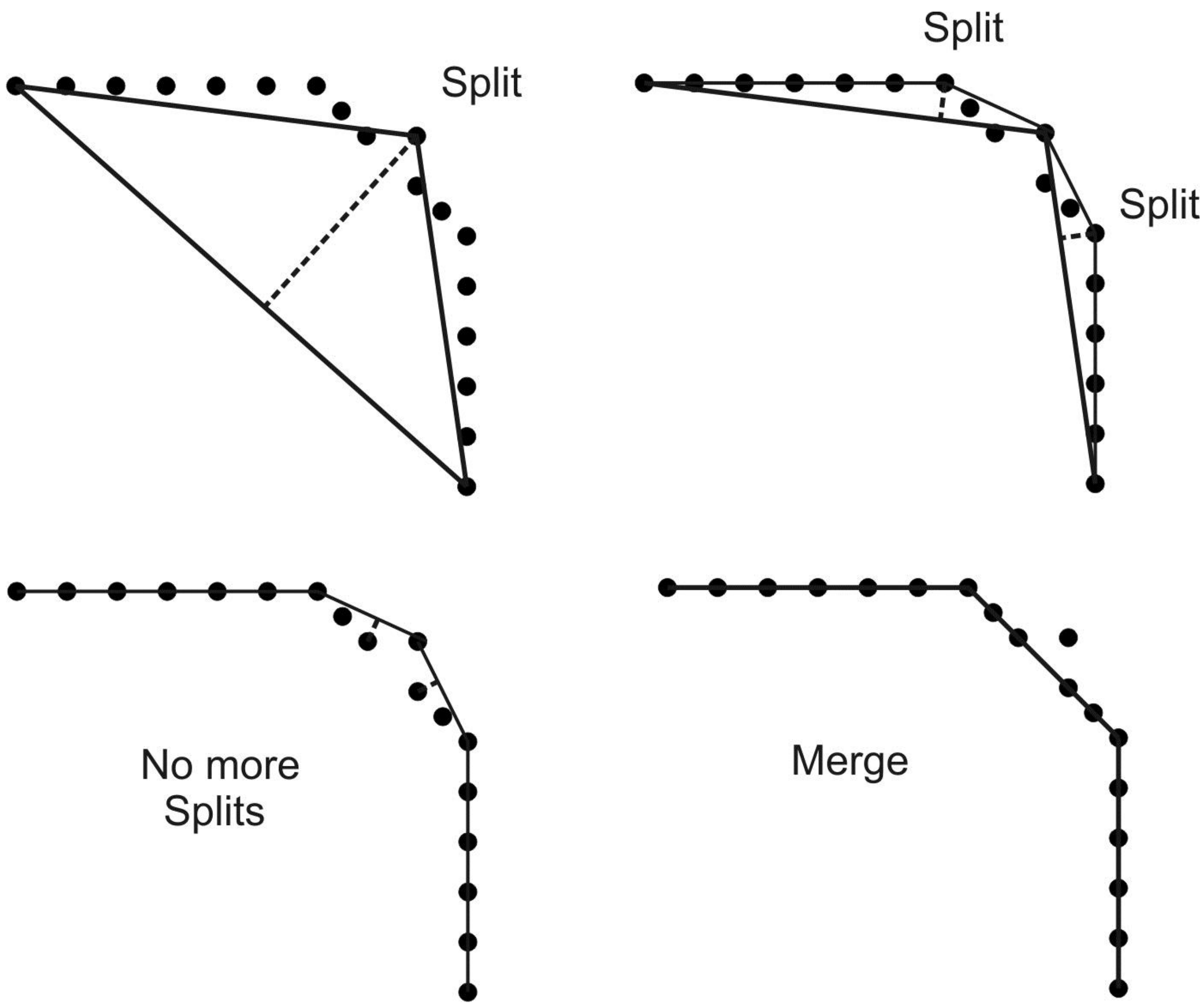
- The Algorithms we will see:
  1. Split and merge
  2. Linear regression
  3. RANSAC
  4. Hough-Transform

Algorithm: Split-and-Merge (standard)

- The most popular algorithm which is originated from computer vision.
- A recursive procedure of fitting and splitting.
- A slightly different version, called Iterative-End-Point-Fit, simply connects the end points for line fitting.



Algorithm: Split-and-Merge (Iterative-End-Point-Fit)

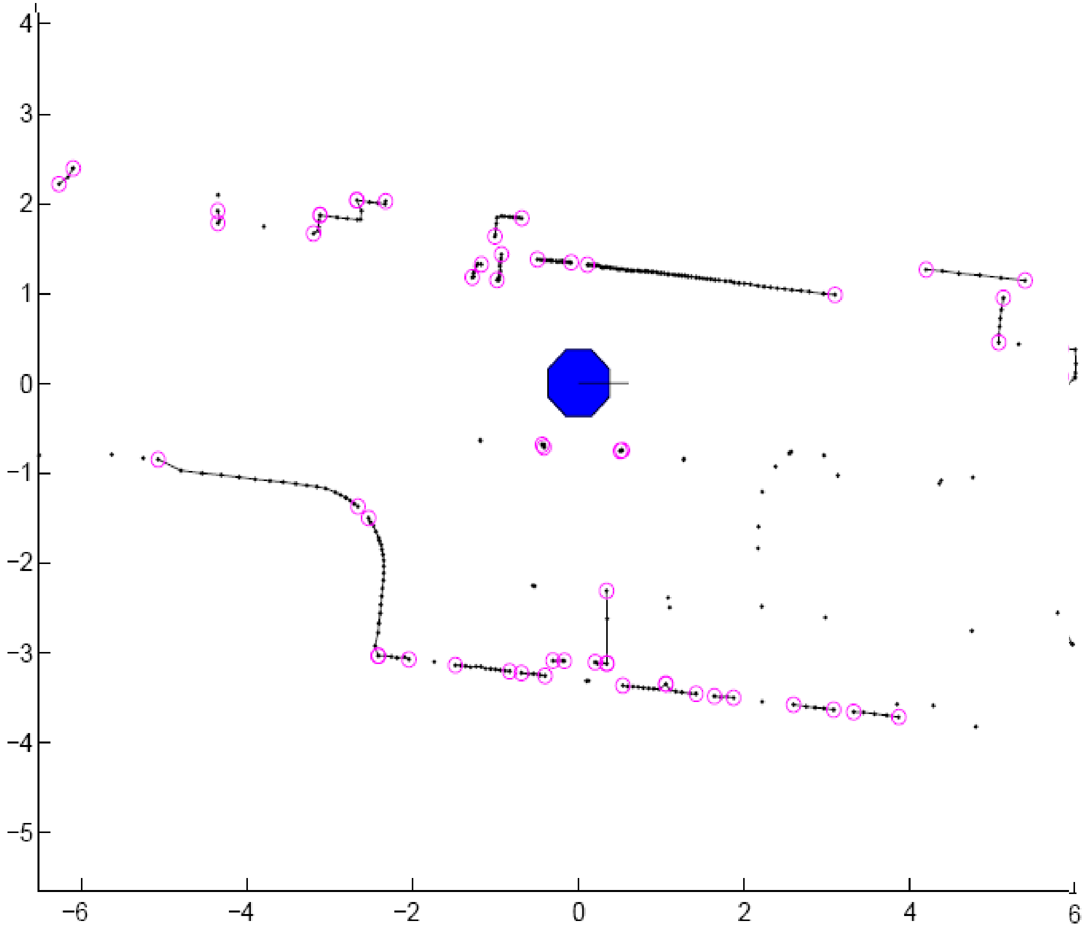


Algorithm: Split-and-Merge

Algorithm 1: *Split-and-Merge*

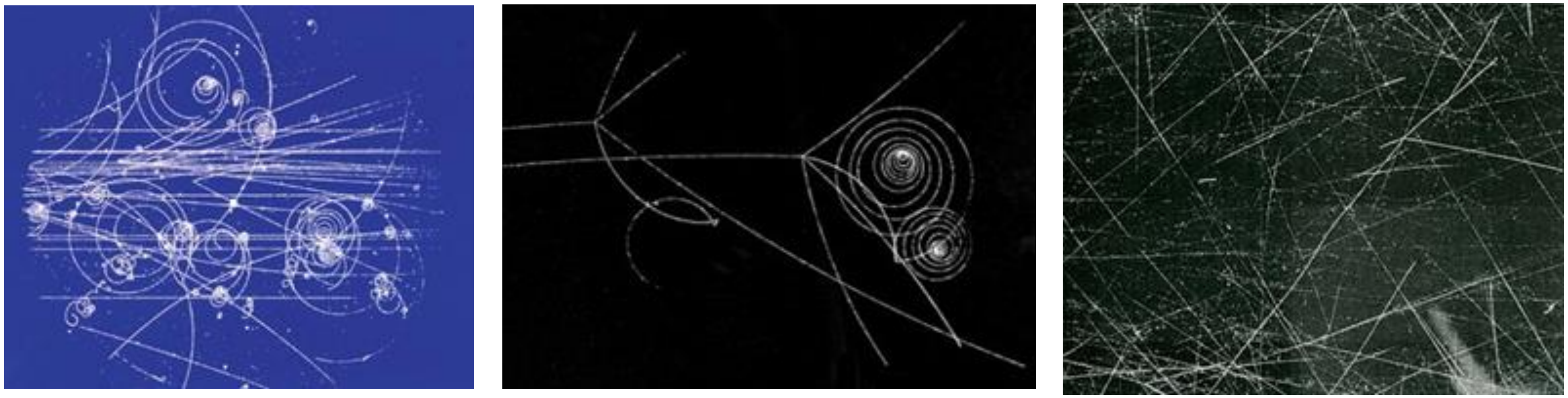
1. Initial: set s_1 consists of N points. Put s_1 in a list L
2. Fit a line to the next set s_i in L
3. Detect point P with maximum distance d_P to the line
4. If d_P is less than a threshold, continue (go to step 2)
5. Otherwise, split s_i at P into s_{i1} and s_{i2}, replace s_i in L by s_{i1} and s_{i2}, continue (go to 2)
6. When all sets (segments) in L have been checked, merge collinear segments.

Algorithm: Split-and-Merge: Example application



Model fitting using the Hough Transform

A bit of history

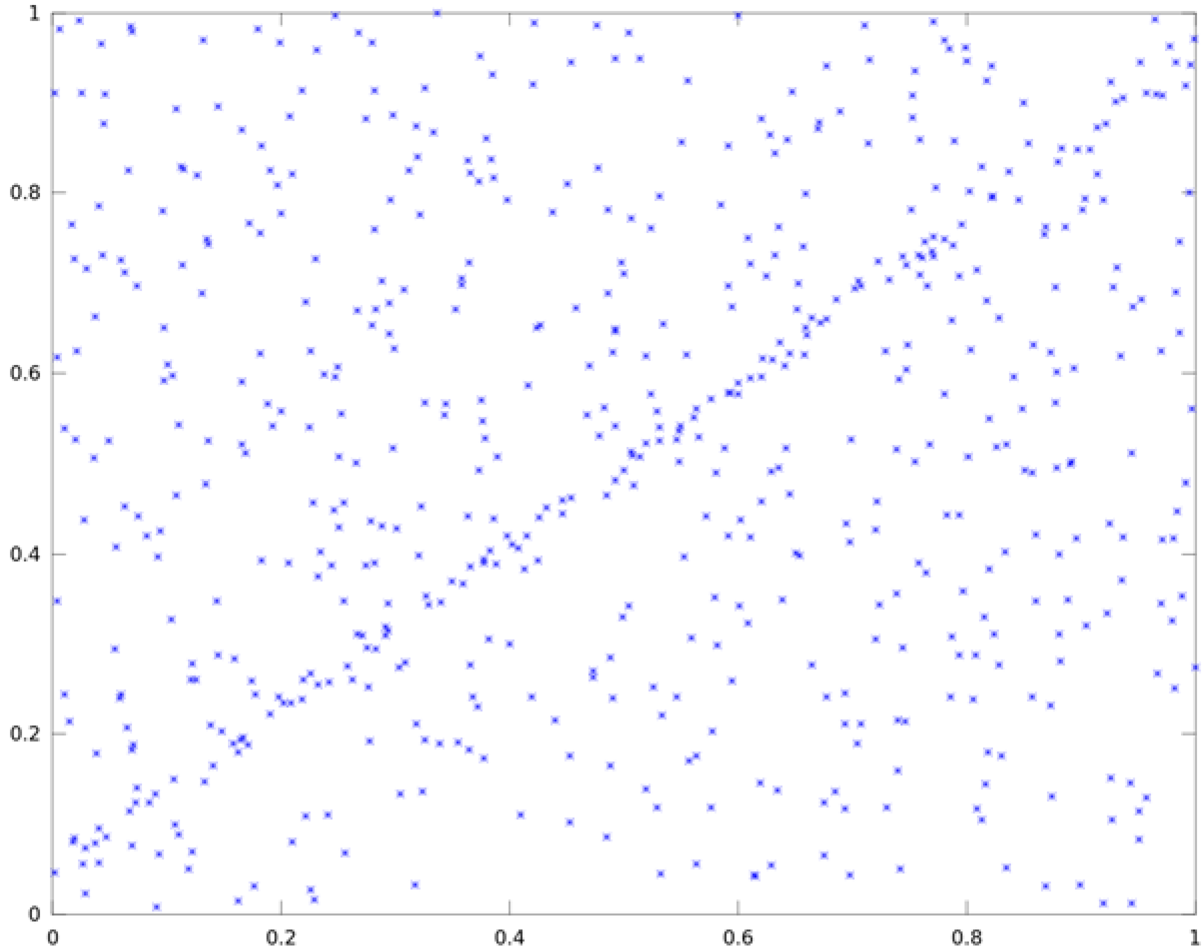


Bubble chamber and cloud chamber images

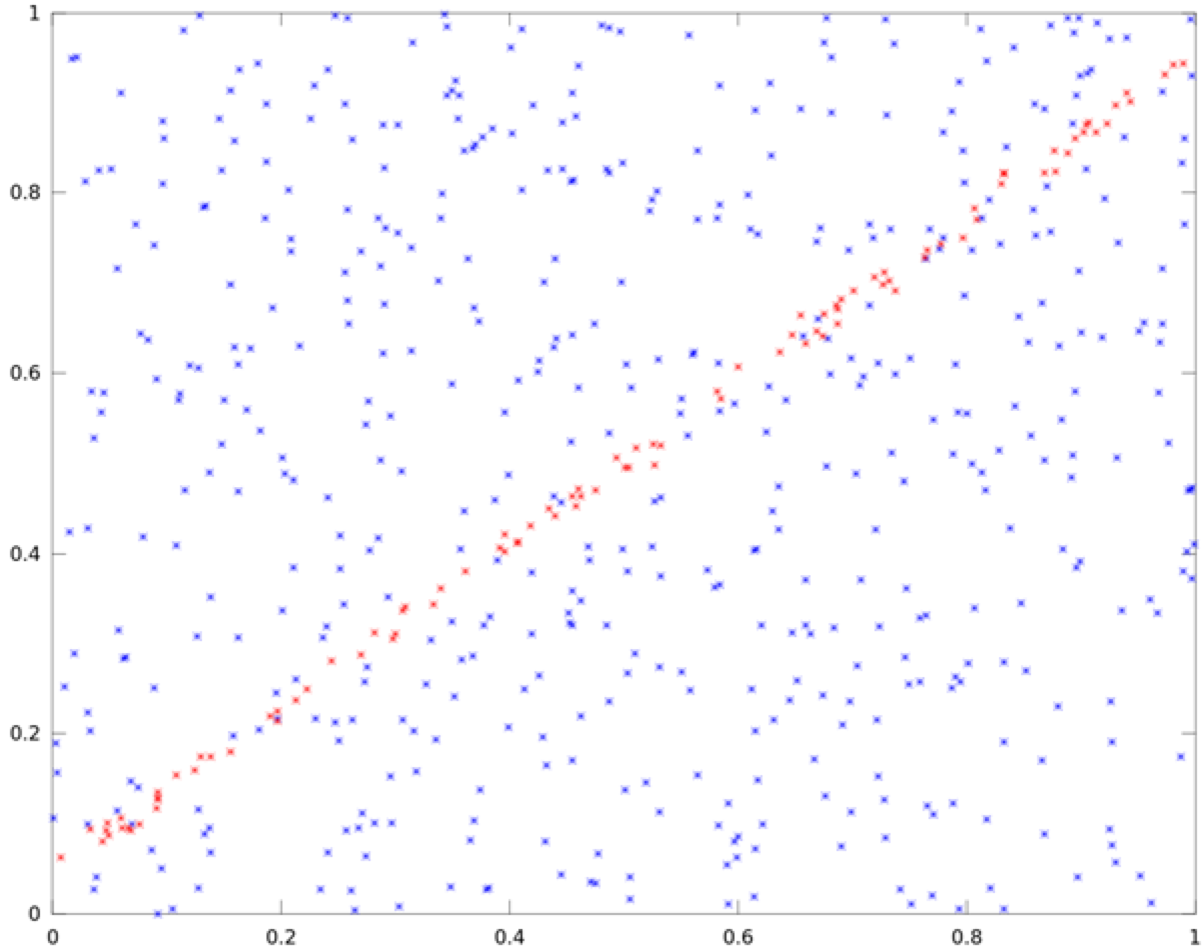
Further references:

http://en.wikipedia.org/wiki/Hough_transform
http://en.wikipedia.org/wiki/RANSAC
http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/RANSAC.shtml

Context

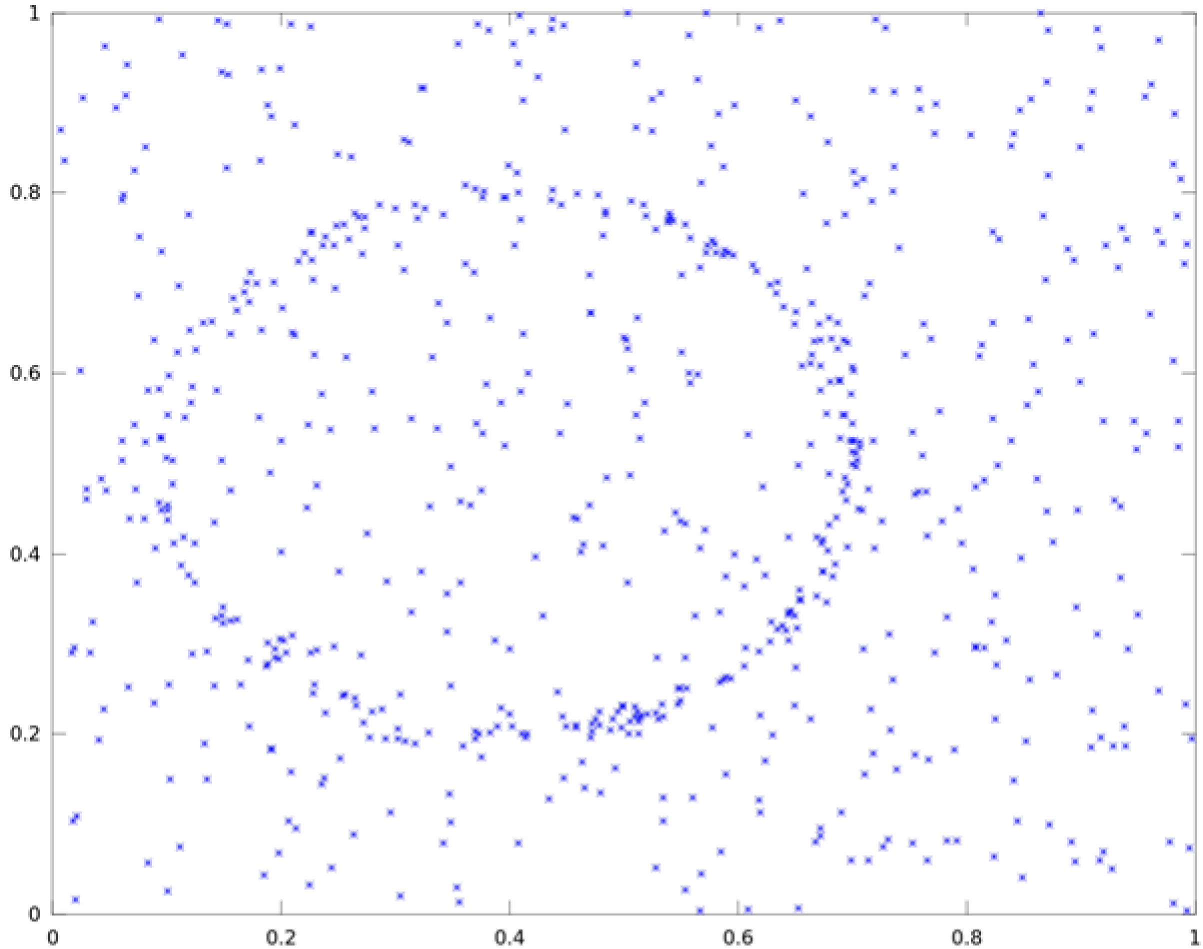


Context

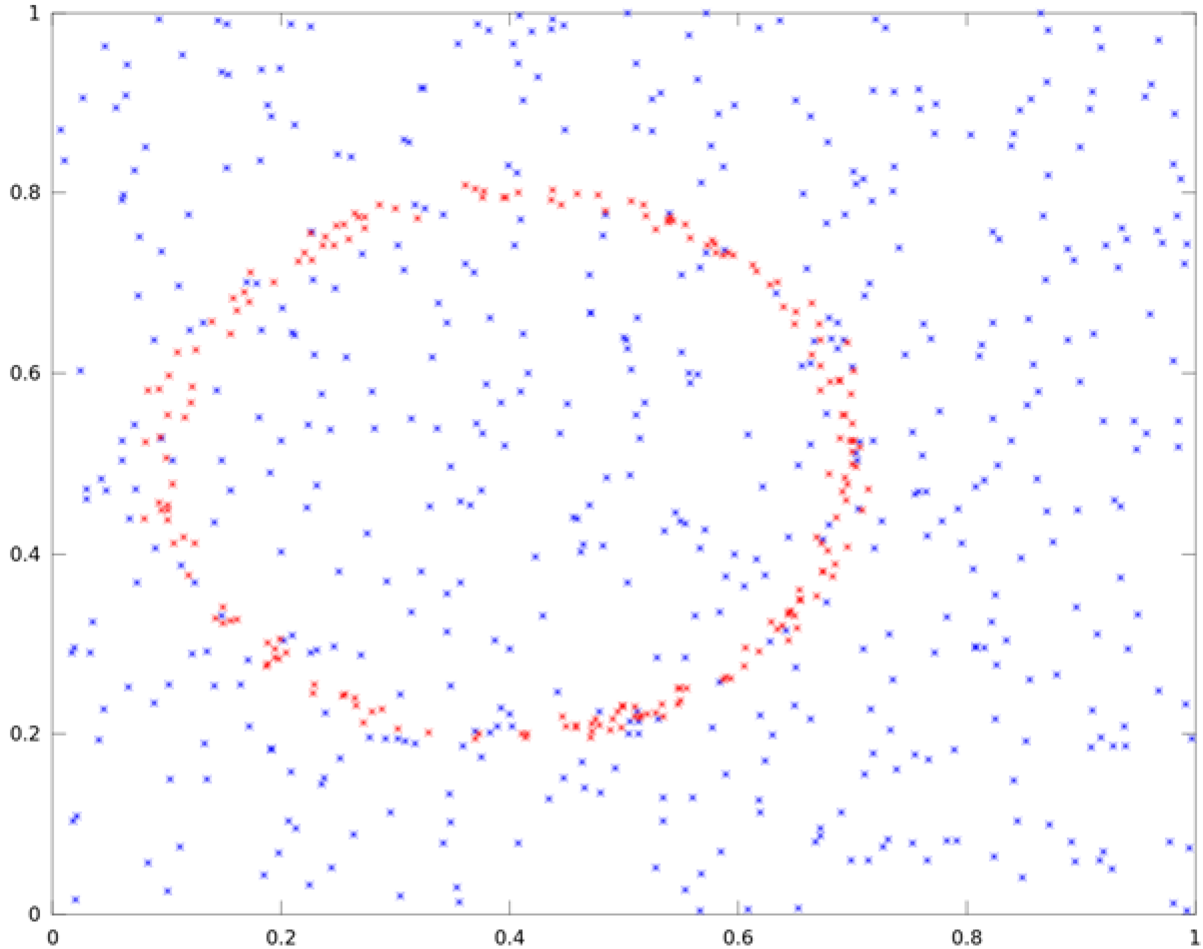


Y = 0.90 * x + 0.05 + 0.01*randn

Context



Context

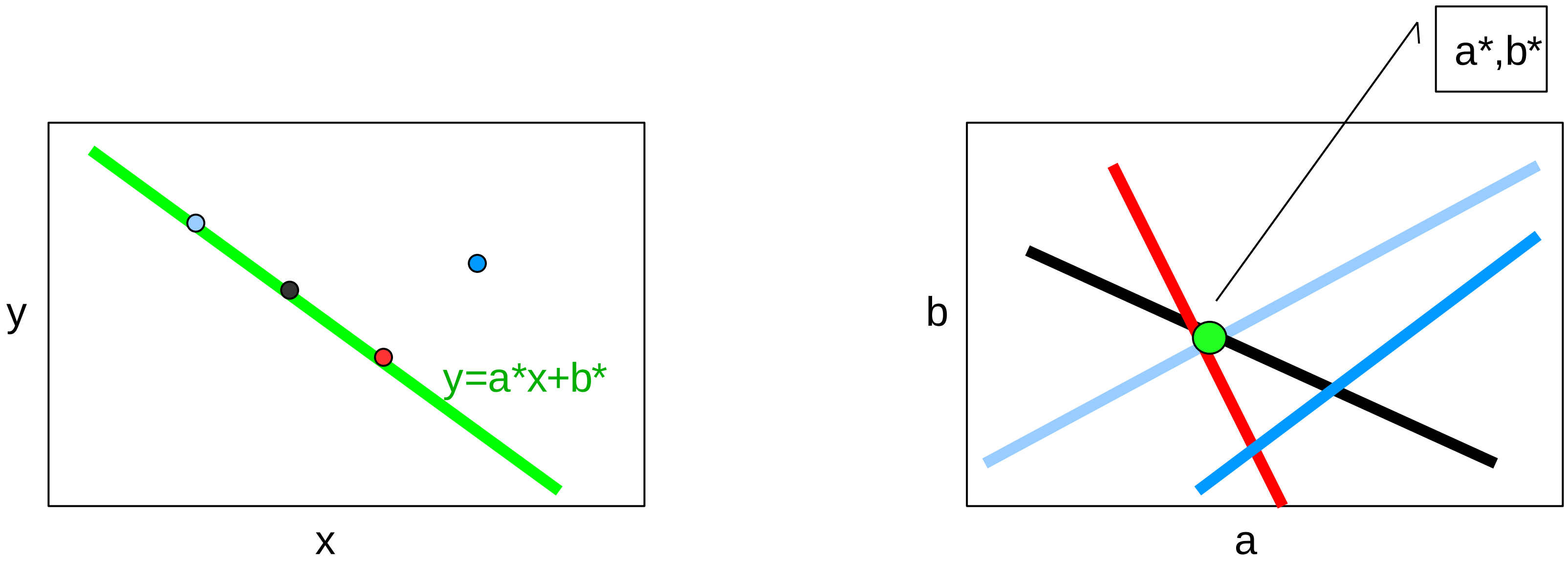


Principle

- Special case: line
  - $y = a\,x + b$
  - Equation for a line in (x,y) space
  - Also a linear equation in (a,b) space:
    - $b = -\,x\,a + b$
  - Each (x,y) data point defines a line in (a,b) space

- Principle
  - Two data points define a single parameter set (a,b)
  - Two lines in (a,b) space intersect to the common (a,b) value

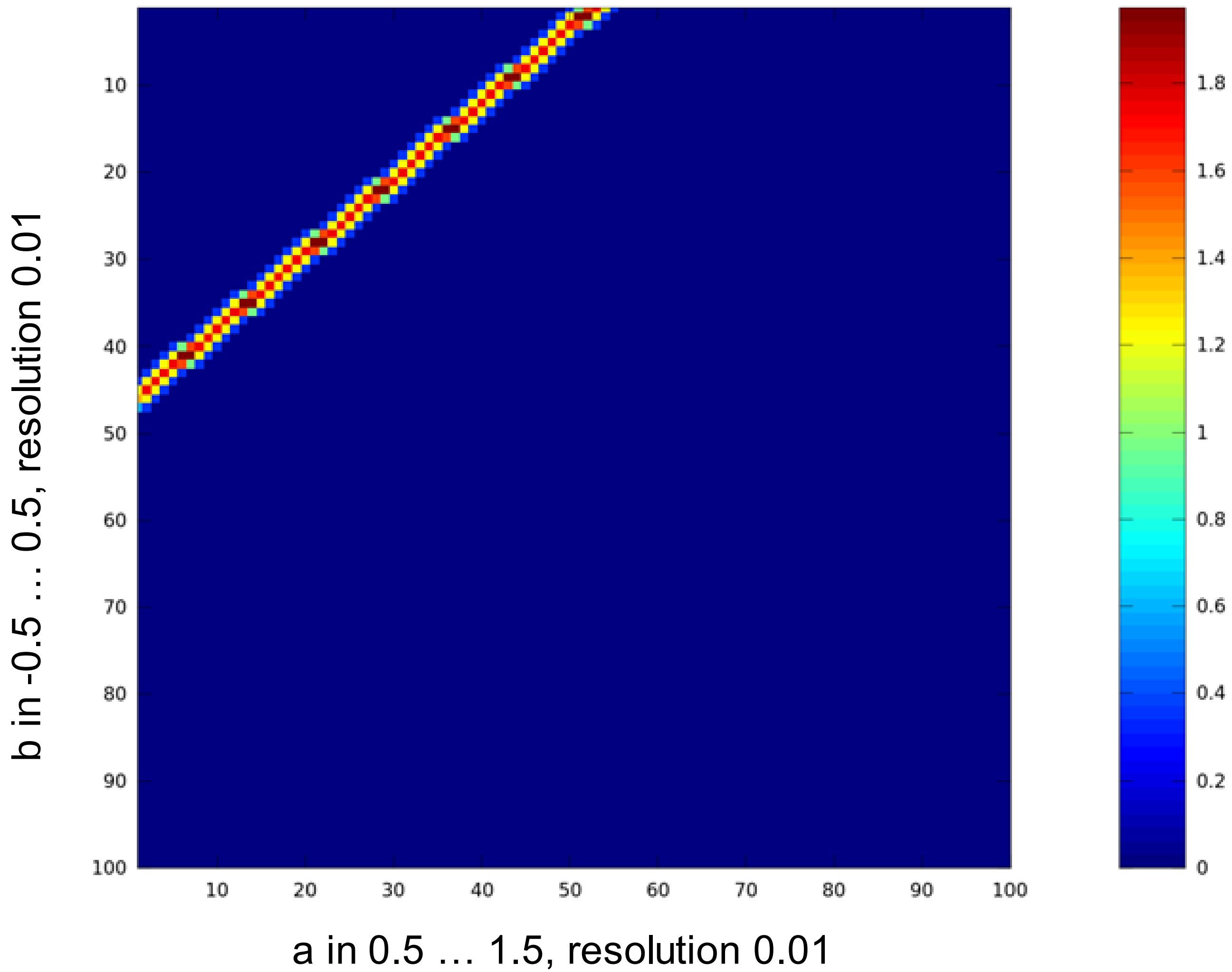
Illustration



Representation of the parameter space

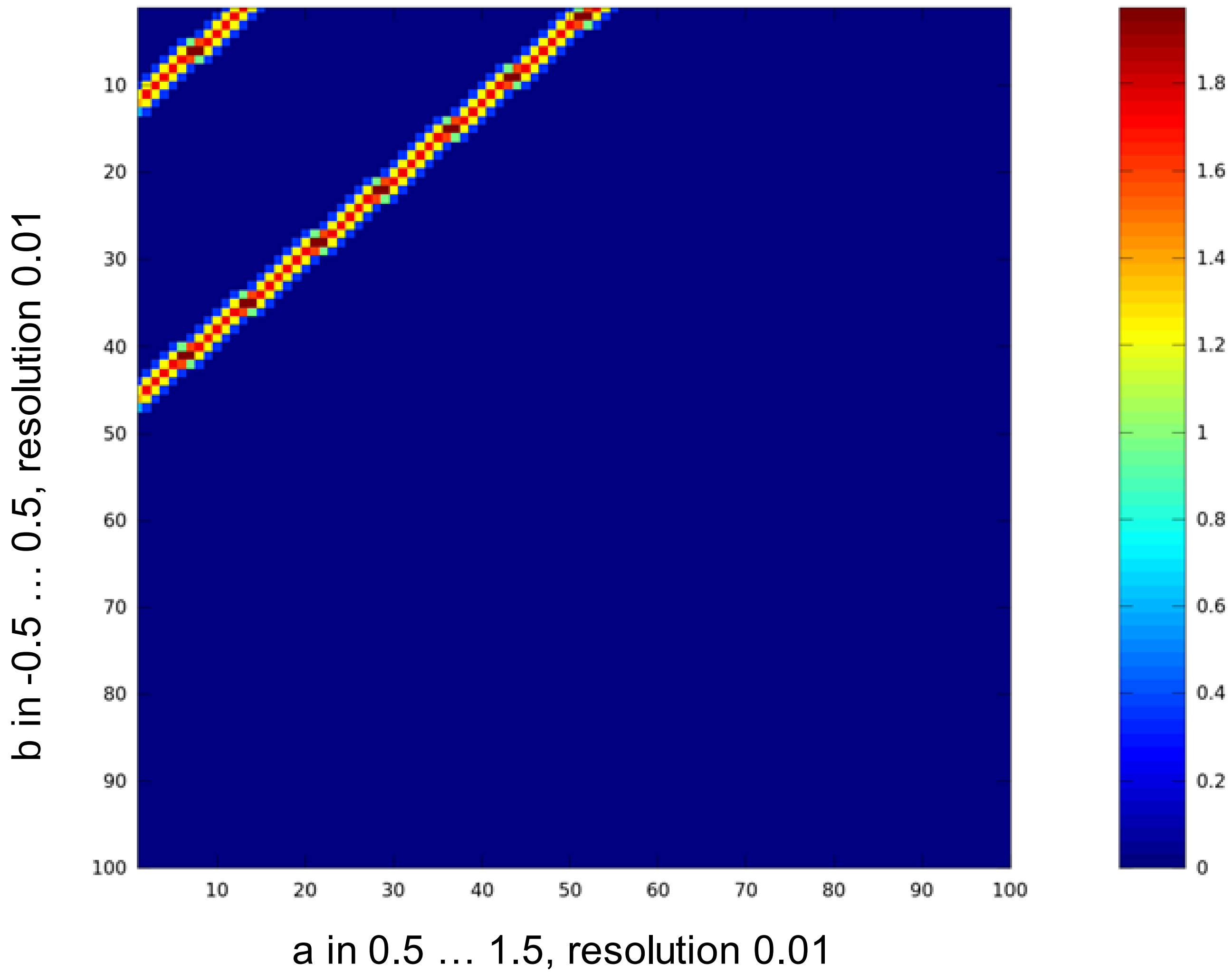
- How to find the parameter set which is consistent with the most data-points?
- Build an “accumulator” over the parameter space, initialised to zero
- For each data point,
  - For all suitable parameter,
    - accumulator[parameter] + 1
- Problem: requires to preset the range of parameters

Step by step accumulation: iteration 1

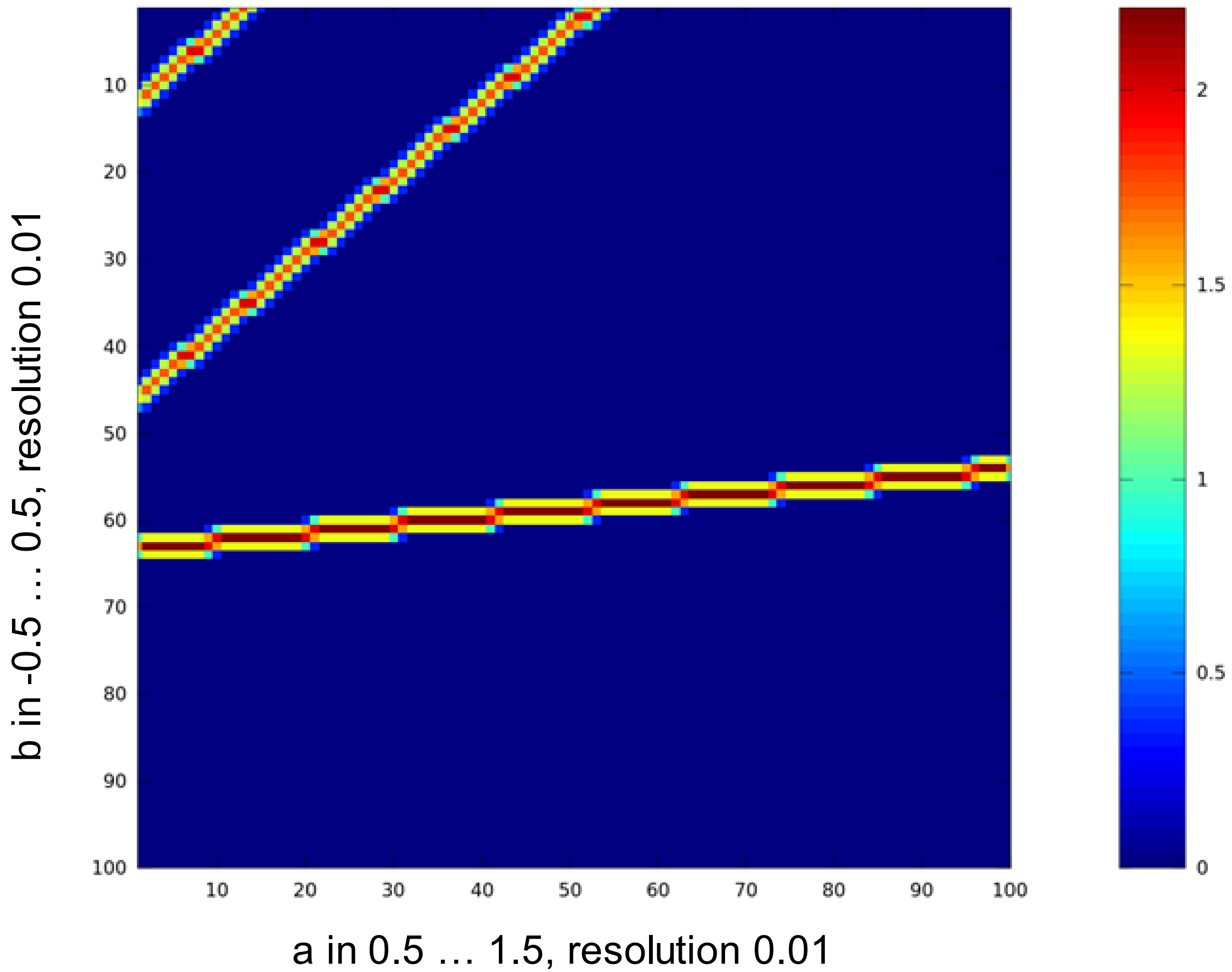


Georgia Tech Lorraine

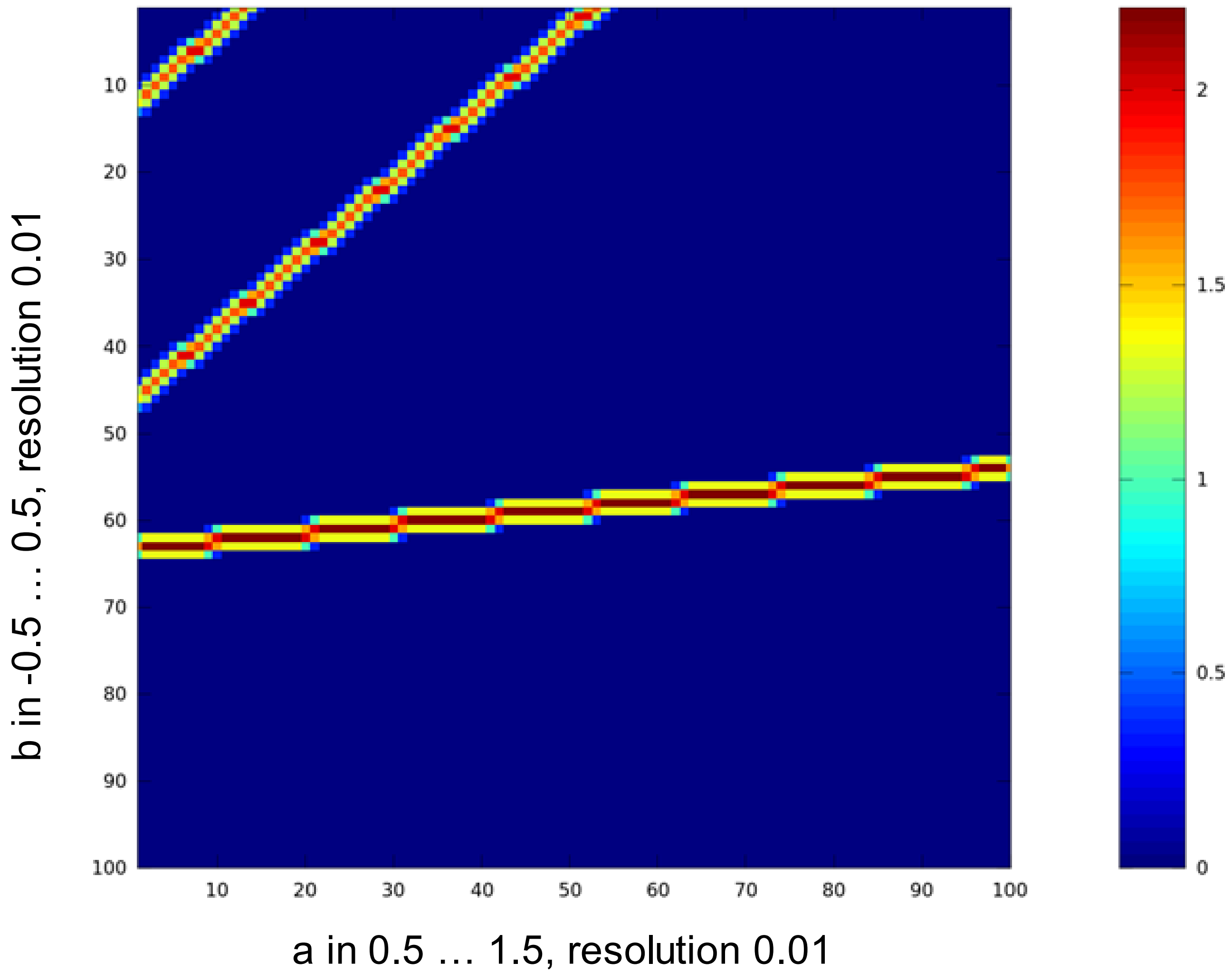
Step by step accumulation: iteration 2



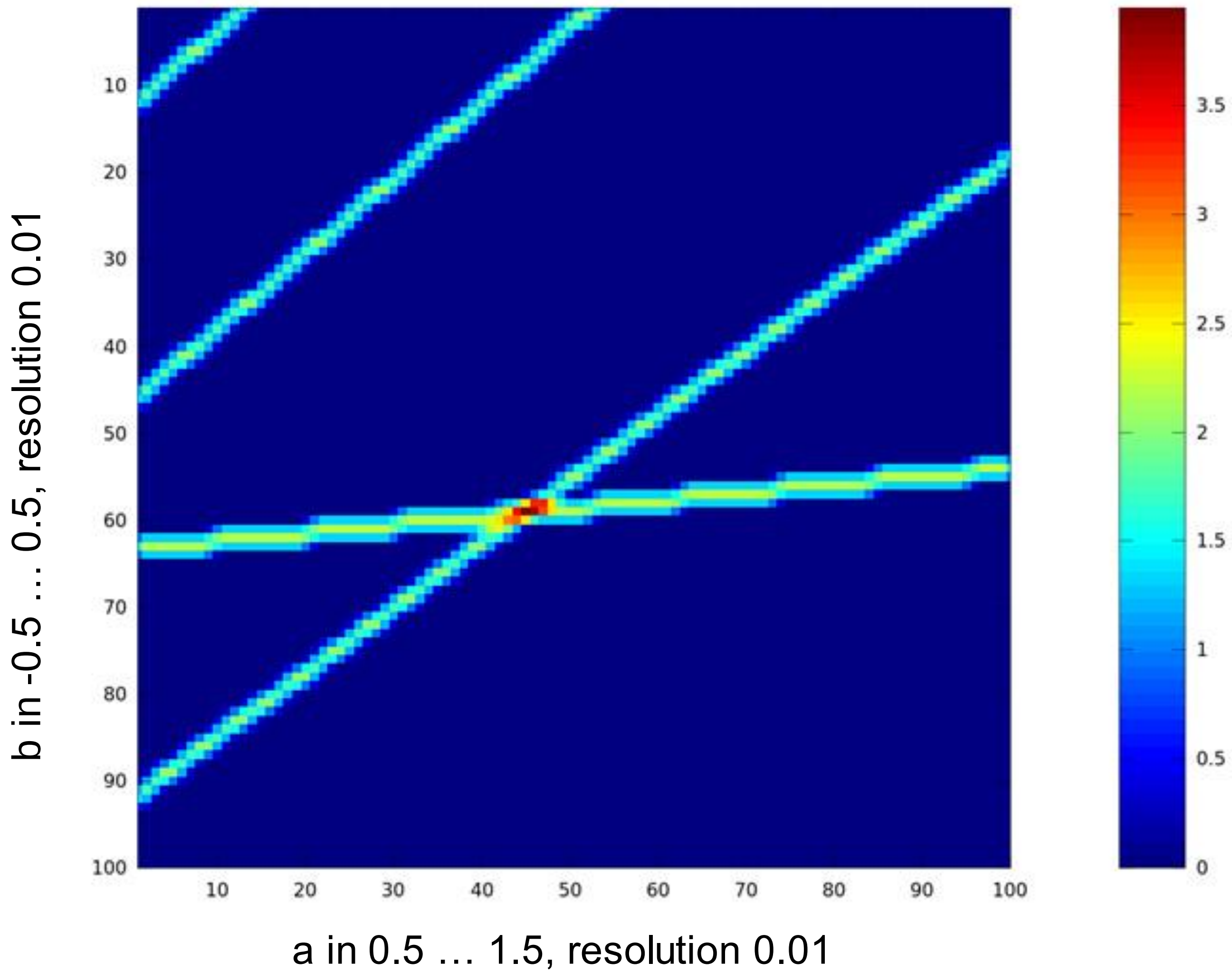
Step by step accumulation: iteration 3



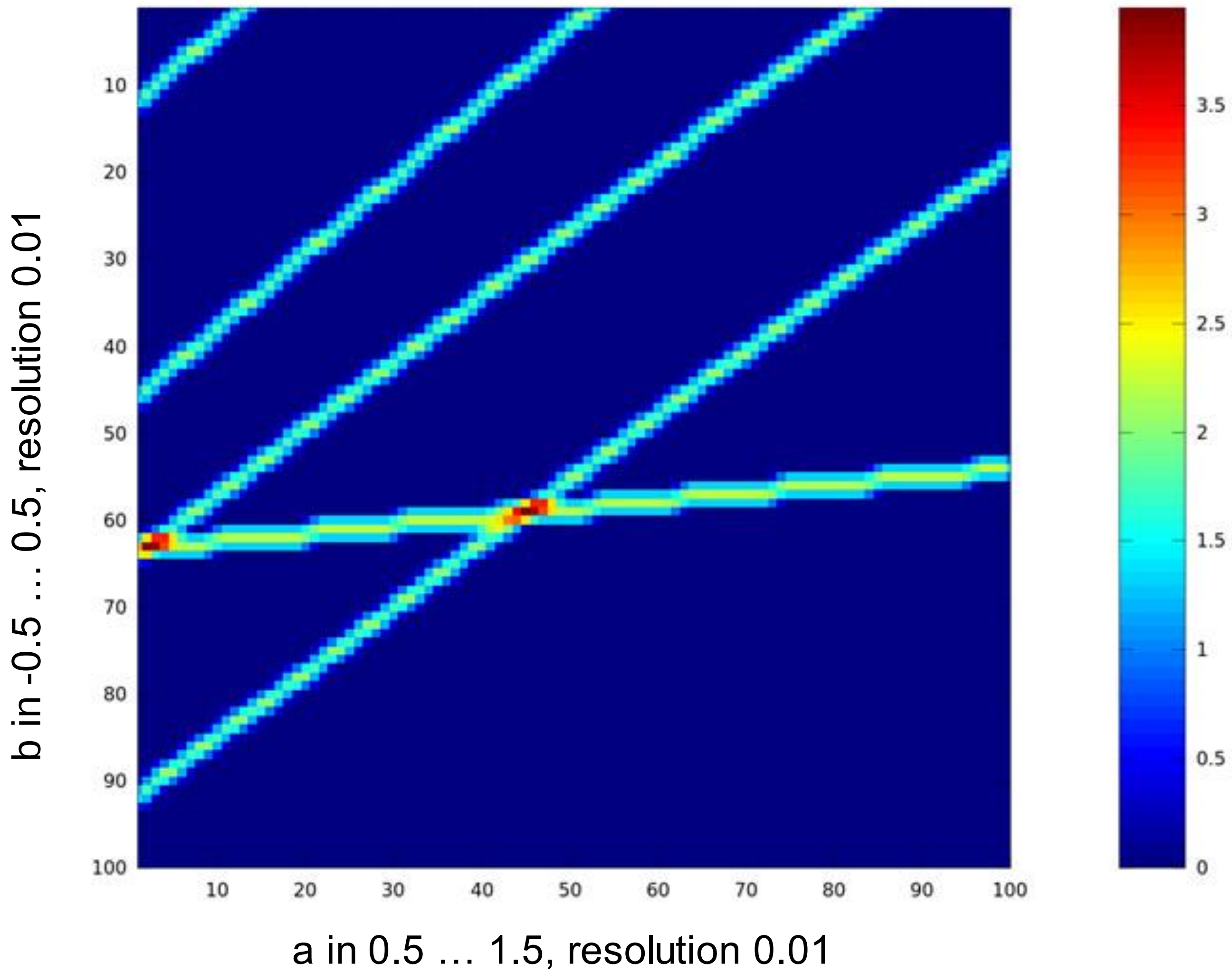
Step by step accumulation: iteration 4



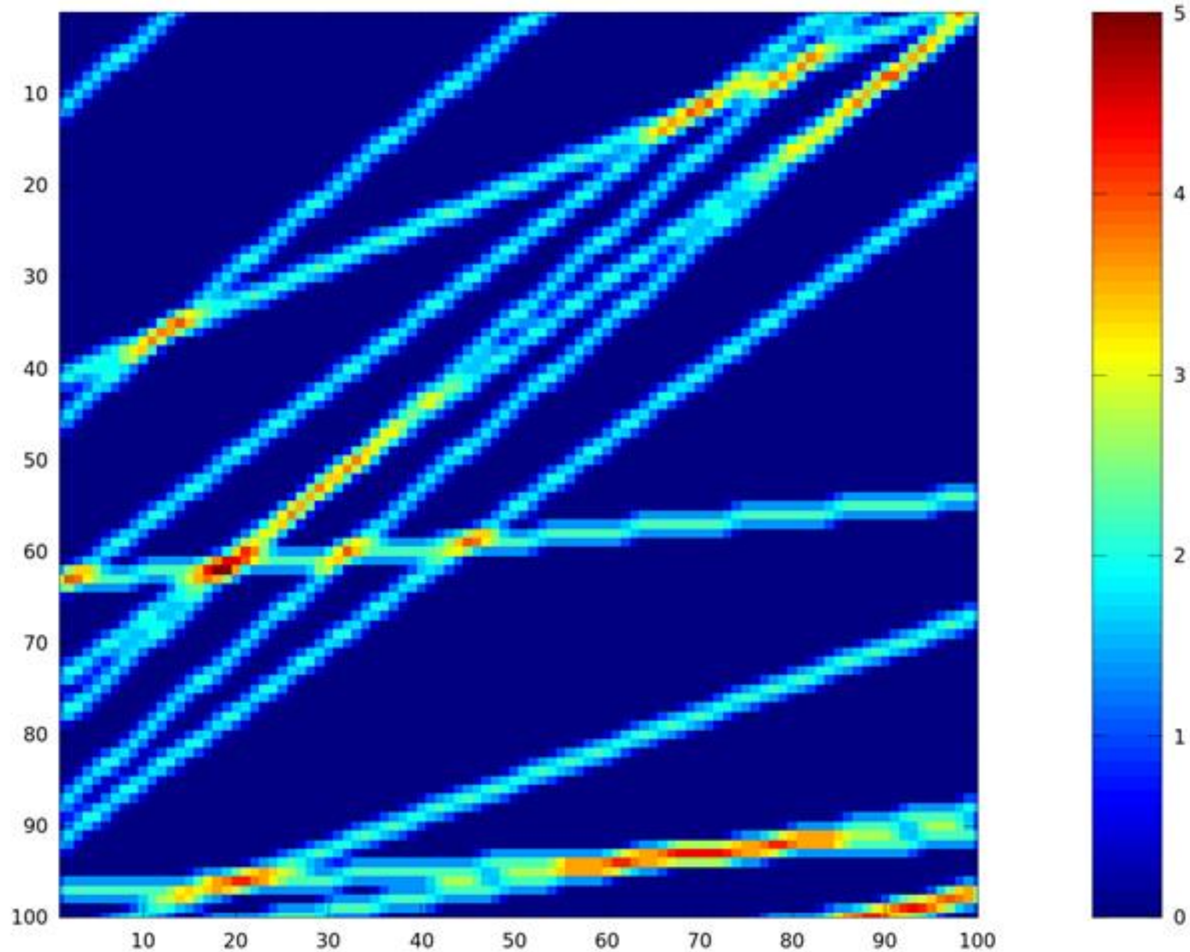
Step by step accumulation: iteration 5



Step by step accumulation: iteration 6



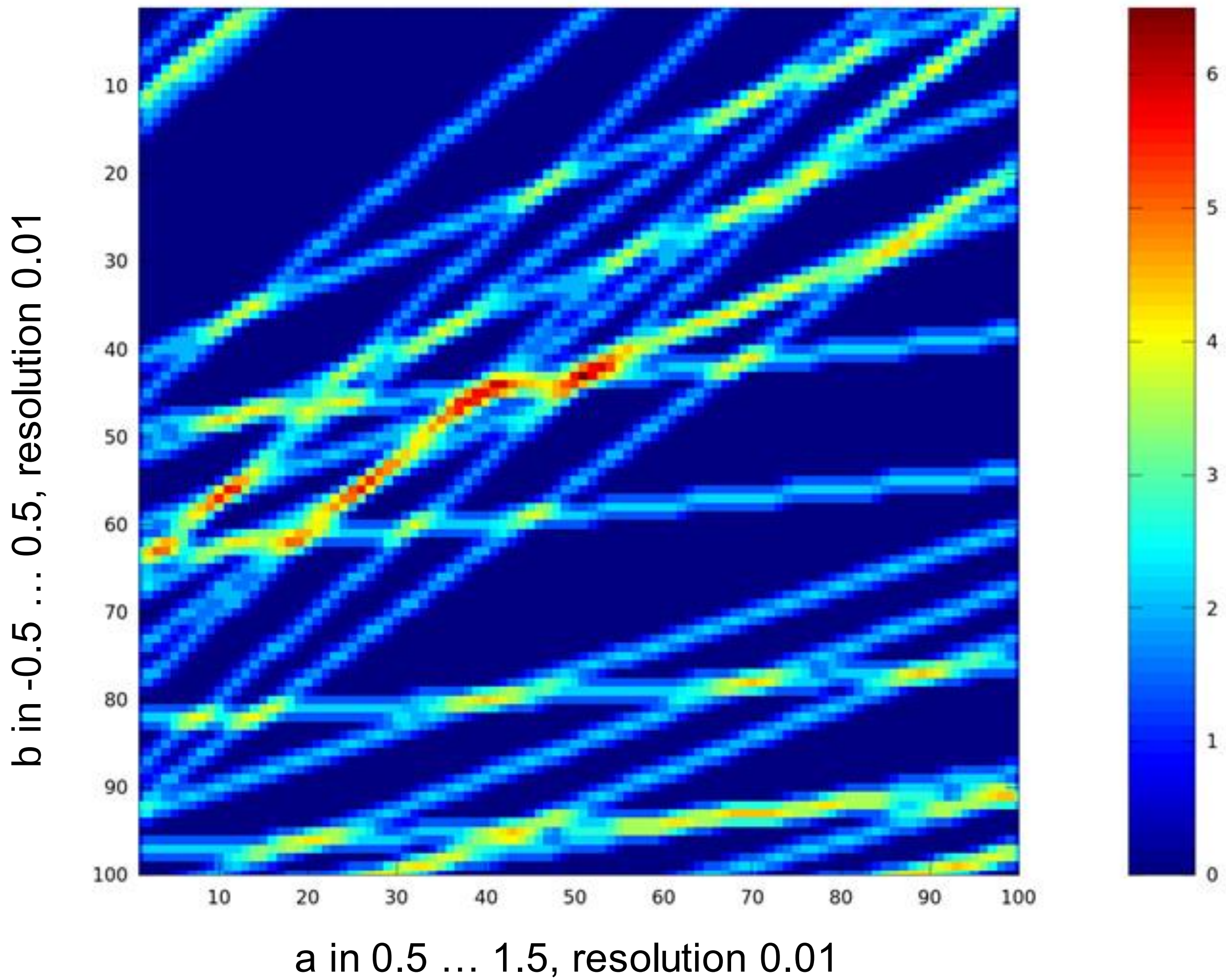
Step by step accumulation: iteration 15



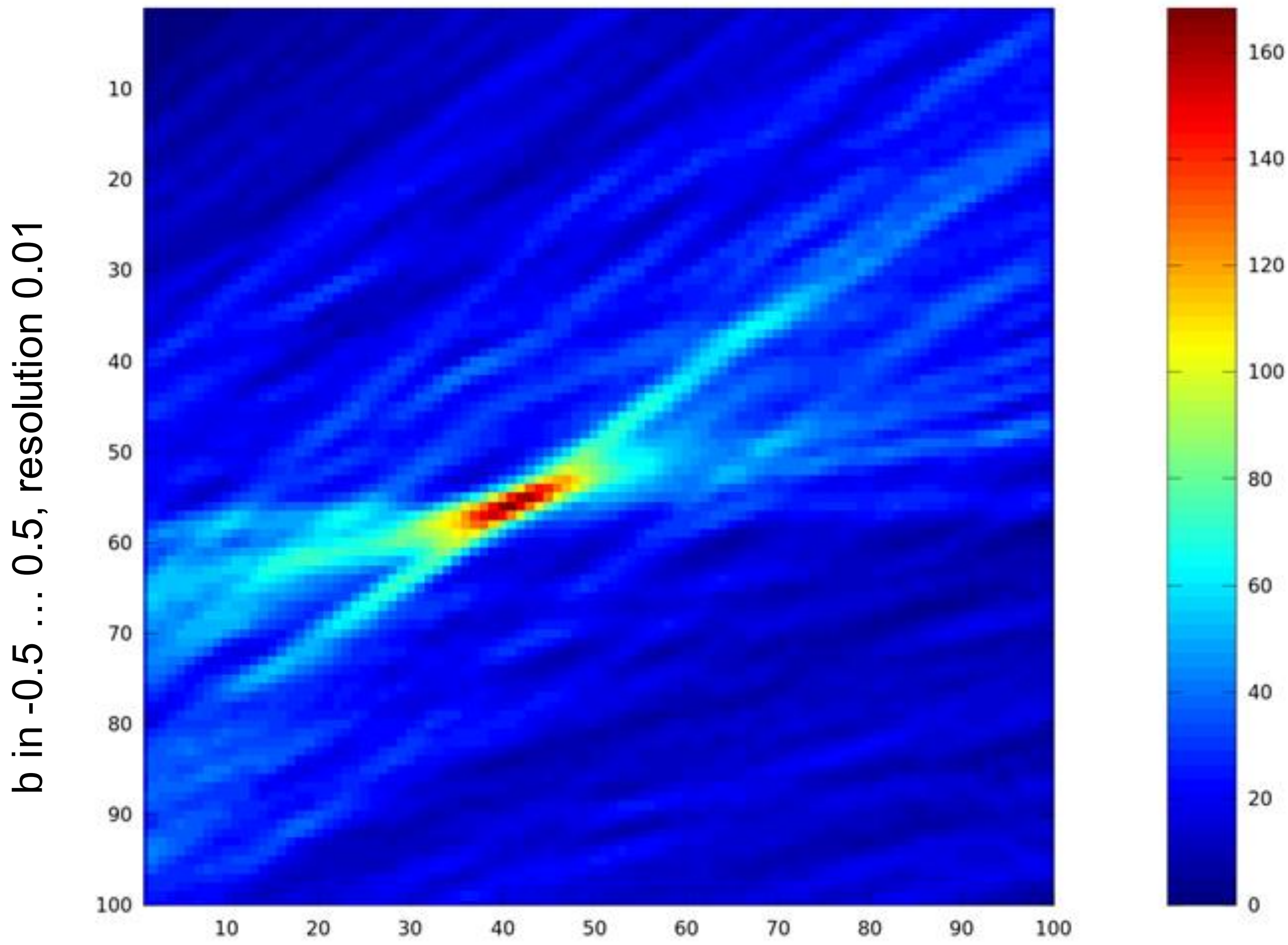
b in -0.5 … 0.5, resolution 0.01

a in 0.5 … 1.5, resolution 0.01

Step by step accumulation: iteration 30



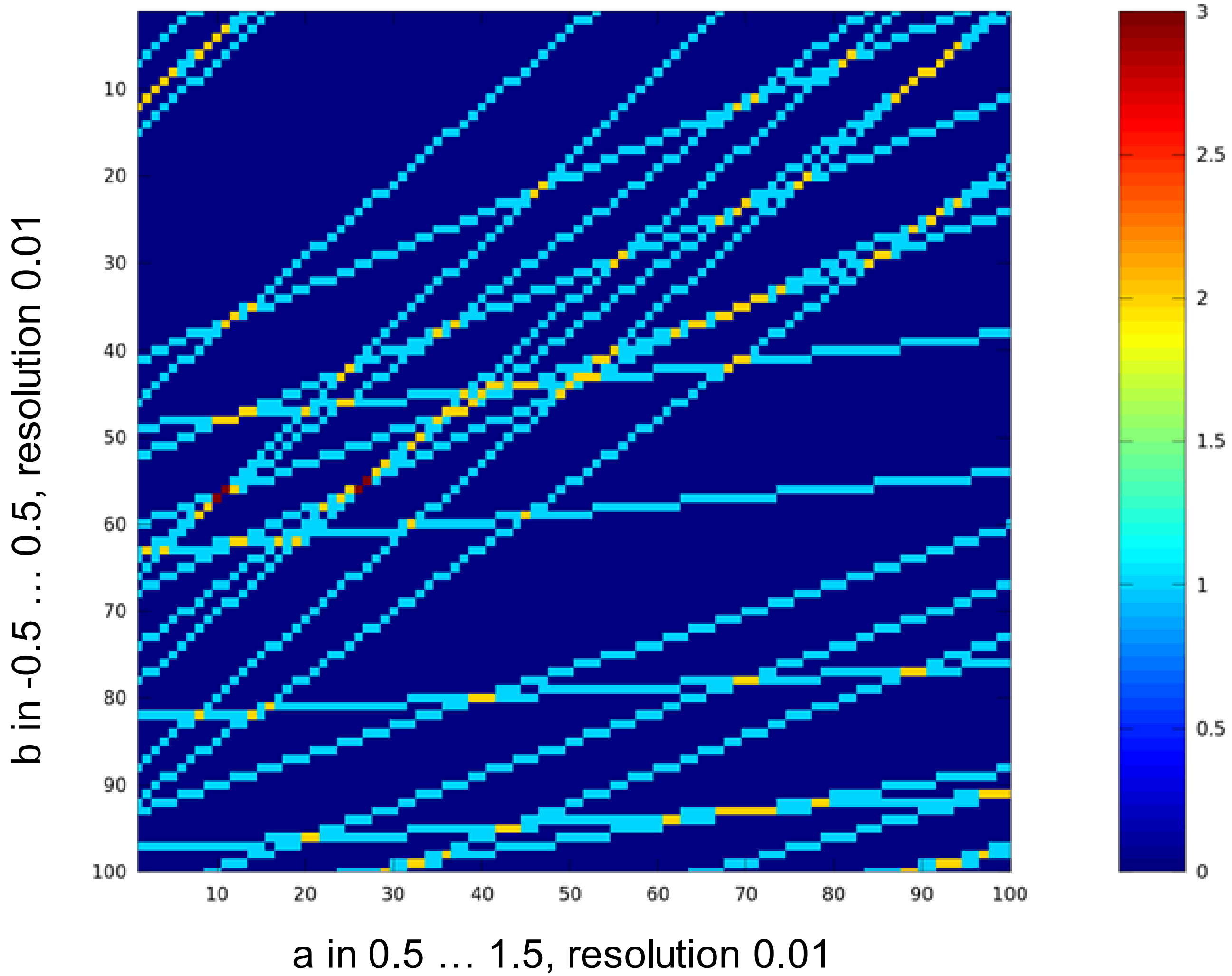
Step by step accumulation: final



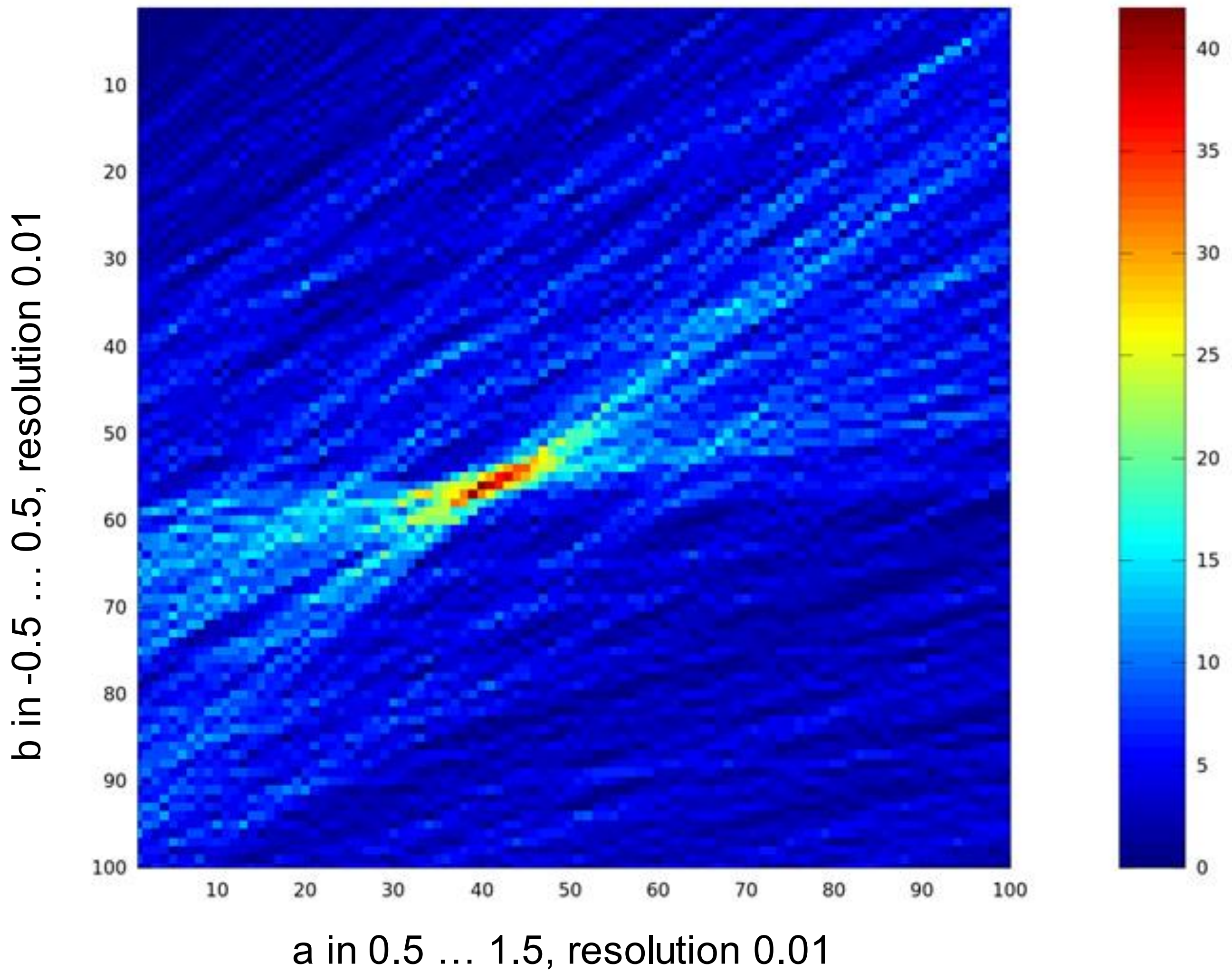
a in 0.5 … 1.5, resolution 0.01

Max at
a = 0.90
b = 0.05

Without smoothing: iteration 30

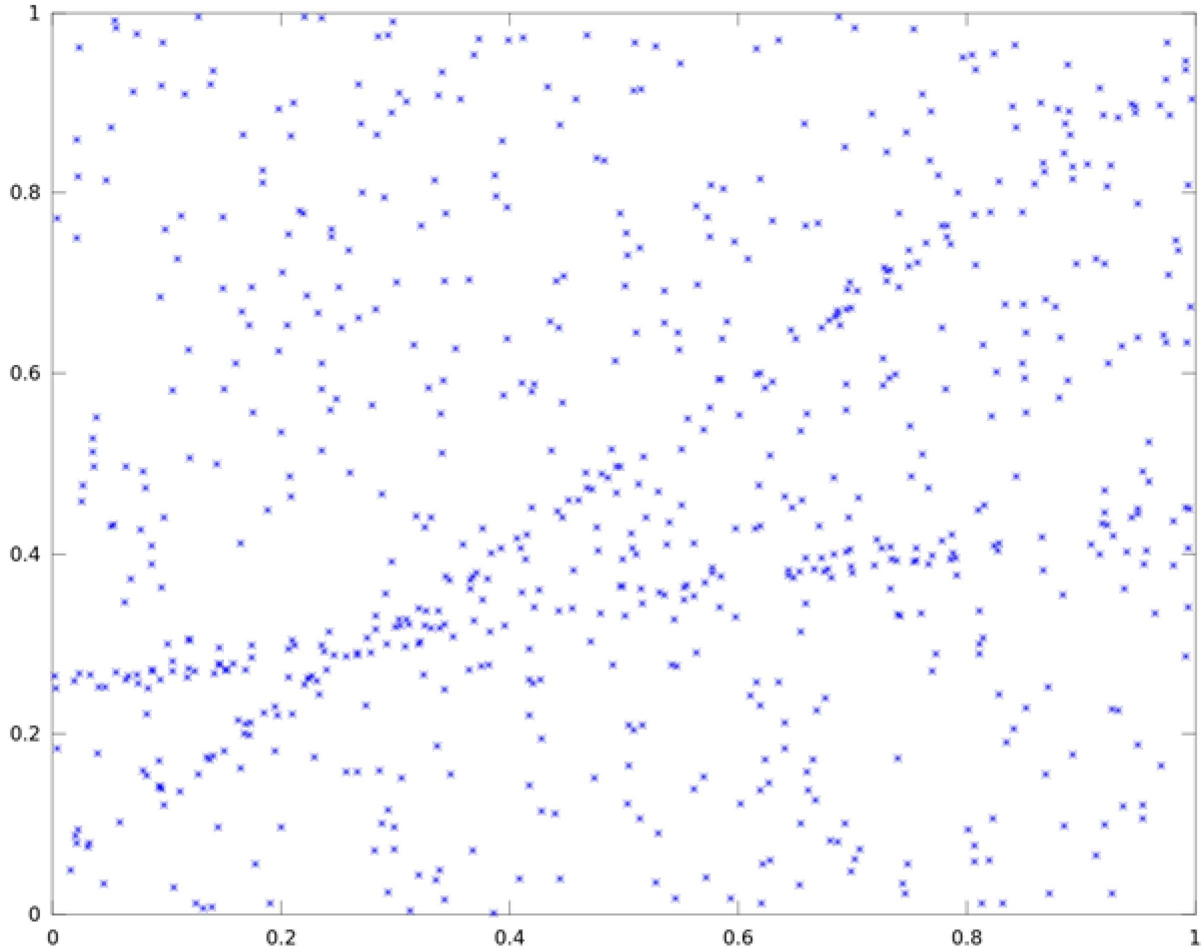


Without smoothing: final



Max at
a = 0.89
b = 0.05

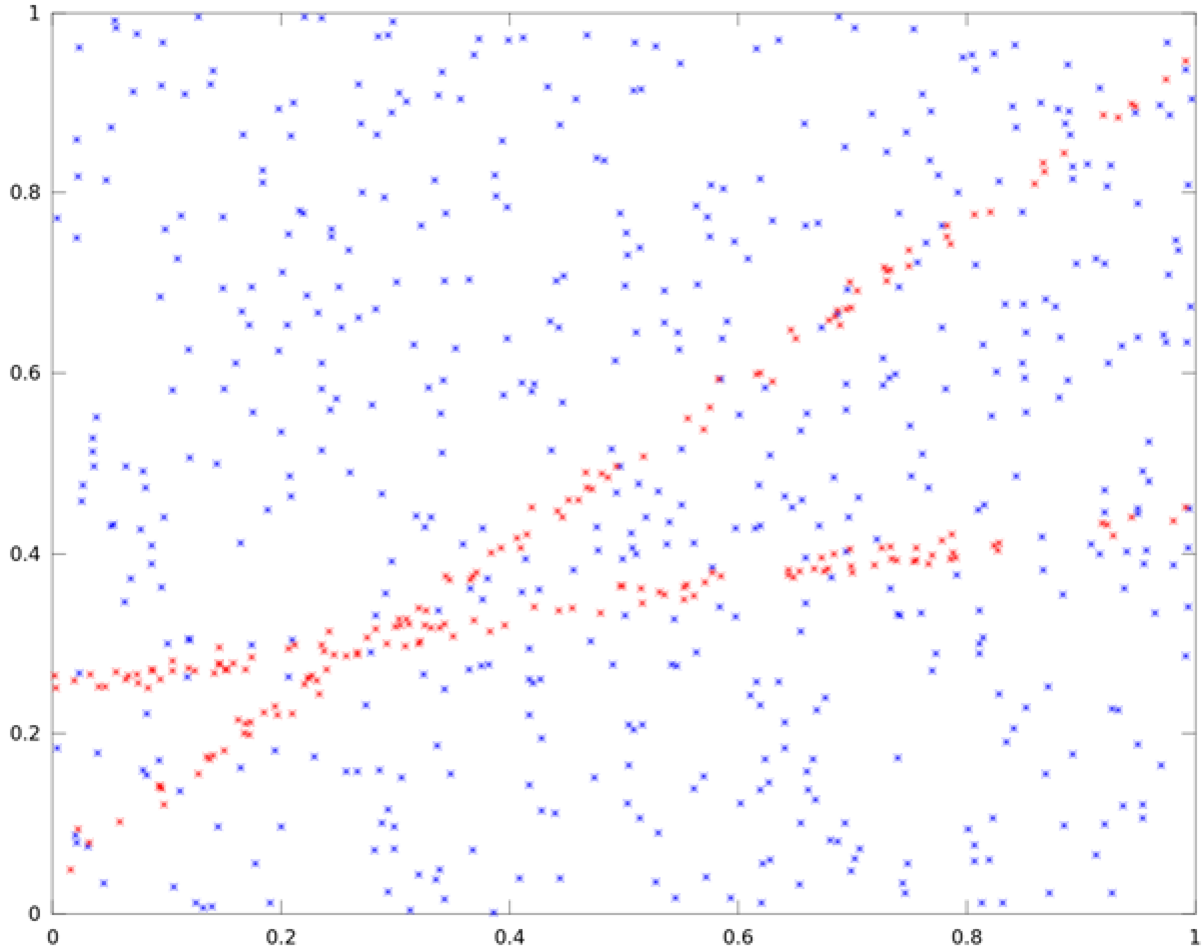
What if there are several lines?



Y = 0.90 * x + 0.05 + 0.01*randn

Y = 0.20 * x + 0.25 + 0.01*randn

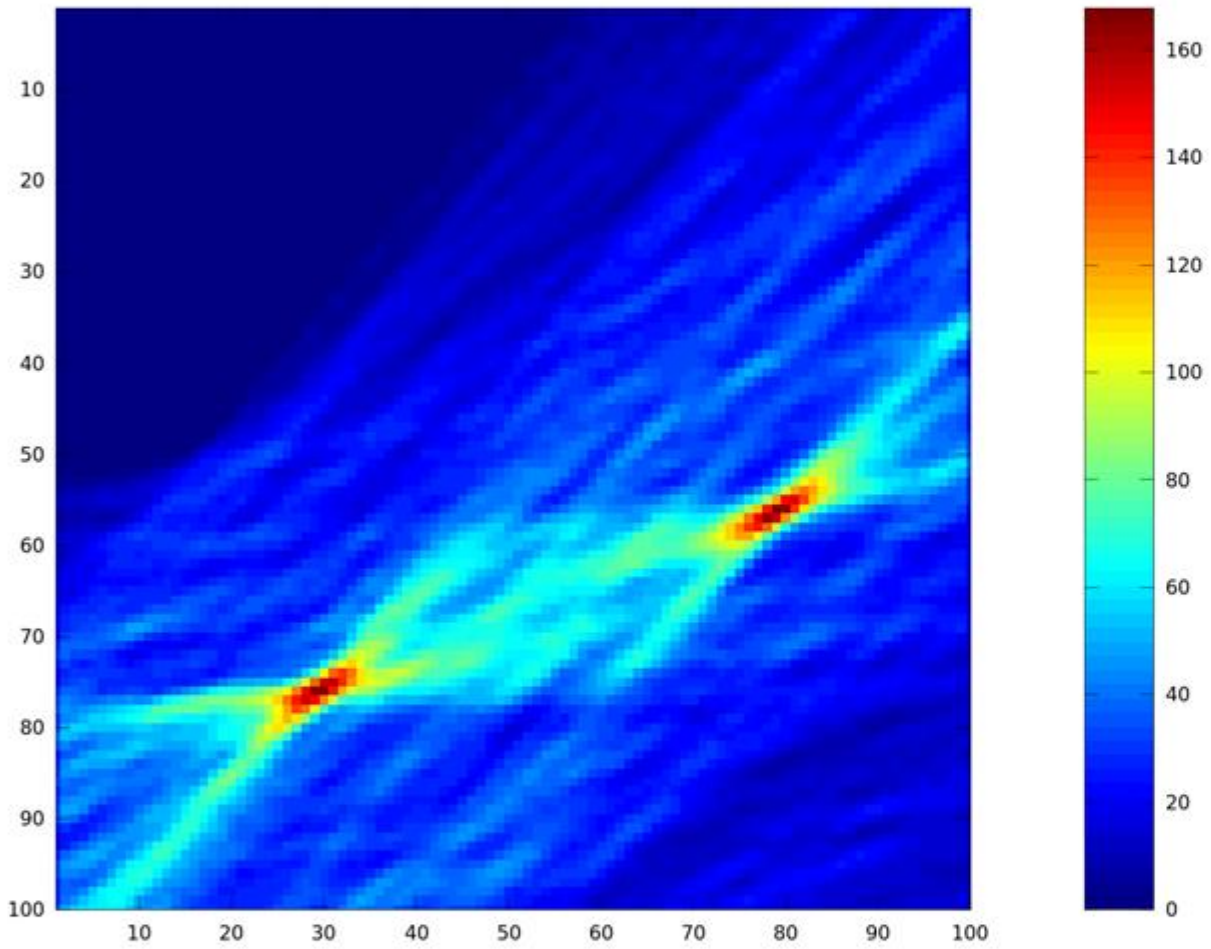
What if there are several lines?



Y = 0.90 * x + 0.05 + 0.01*randn

Y = 0.20 * x + 0.25 + 0.01*randn

What if there are several lines?



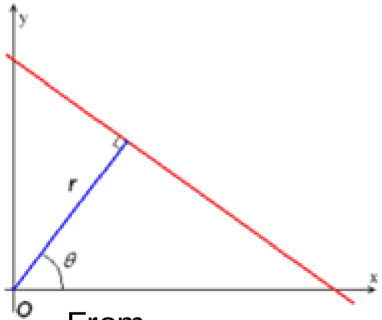
Max at
a = 0.19
b = 0.25

What if there are several lines

- Find multiple peaks
  - Thresholding...
  - Mean-shift...
  - Non-maxima suppression
  - …
- Find the first line, then remove all data points from this line, and run Hough again...
- Hacks and ad-hoc solutions

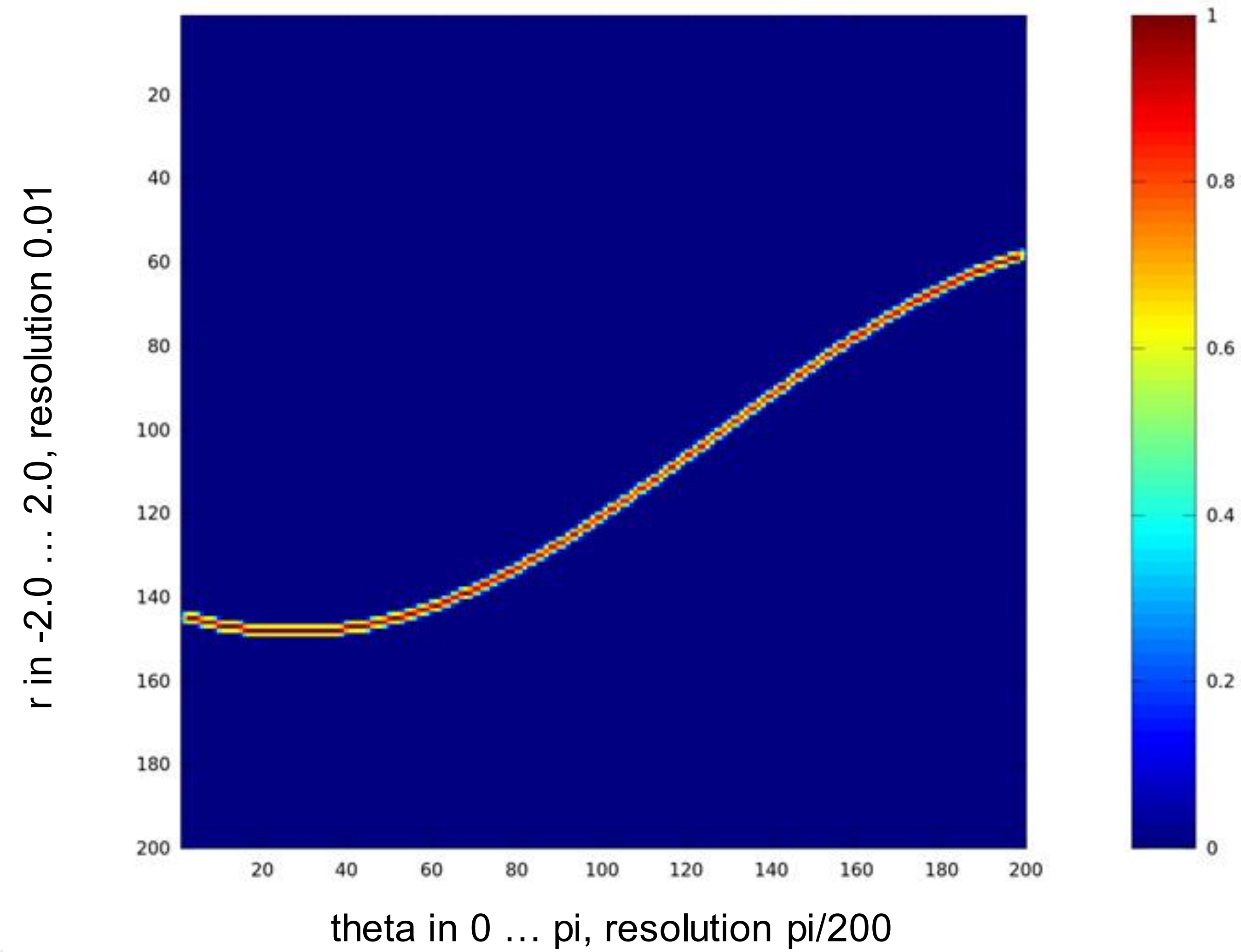
Other line representations

- Y = a*x+b: a is unbounded, so the accumulator needs to be unbounded as well.
- Alternative: change the representation
  - A line is defined by its distance to the origin and angle:
  - For a given point (x,y),
    - Lines passing through x,y verify

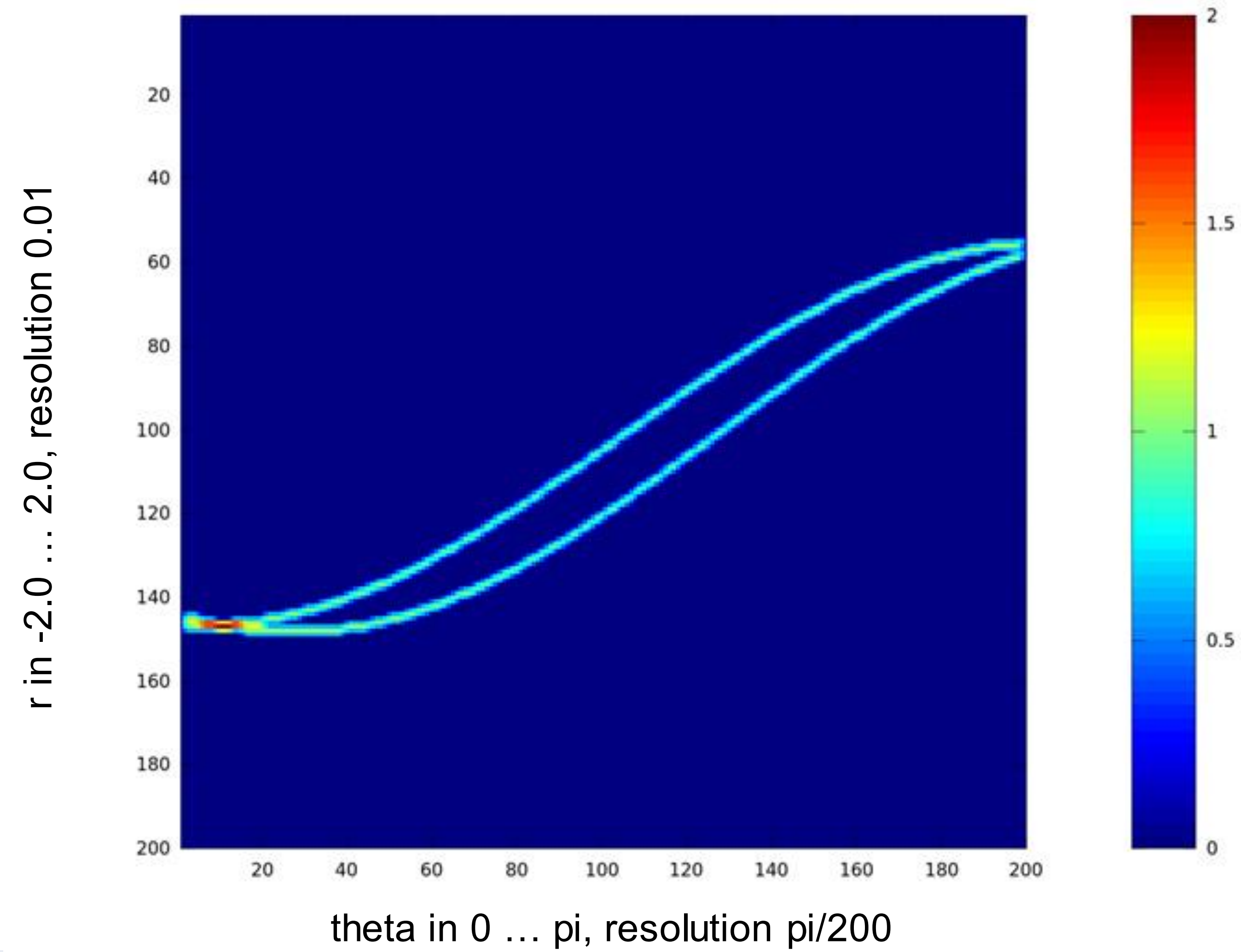


From wikipedia

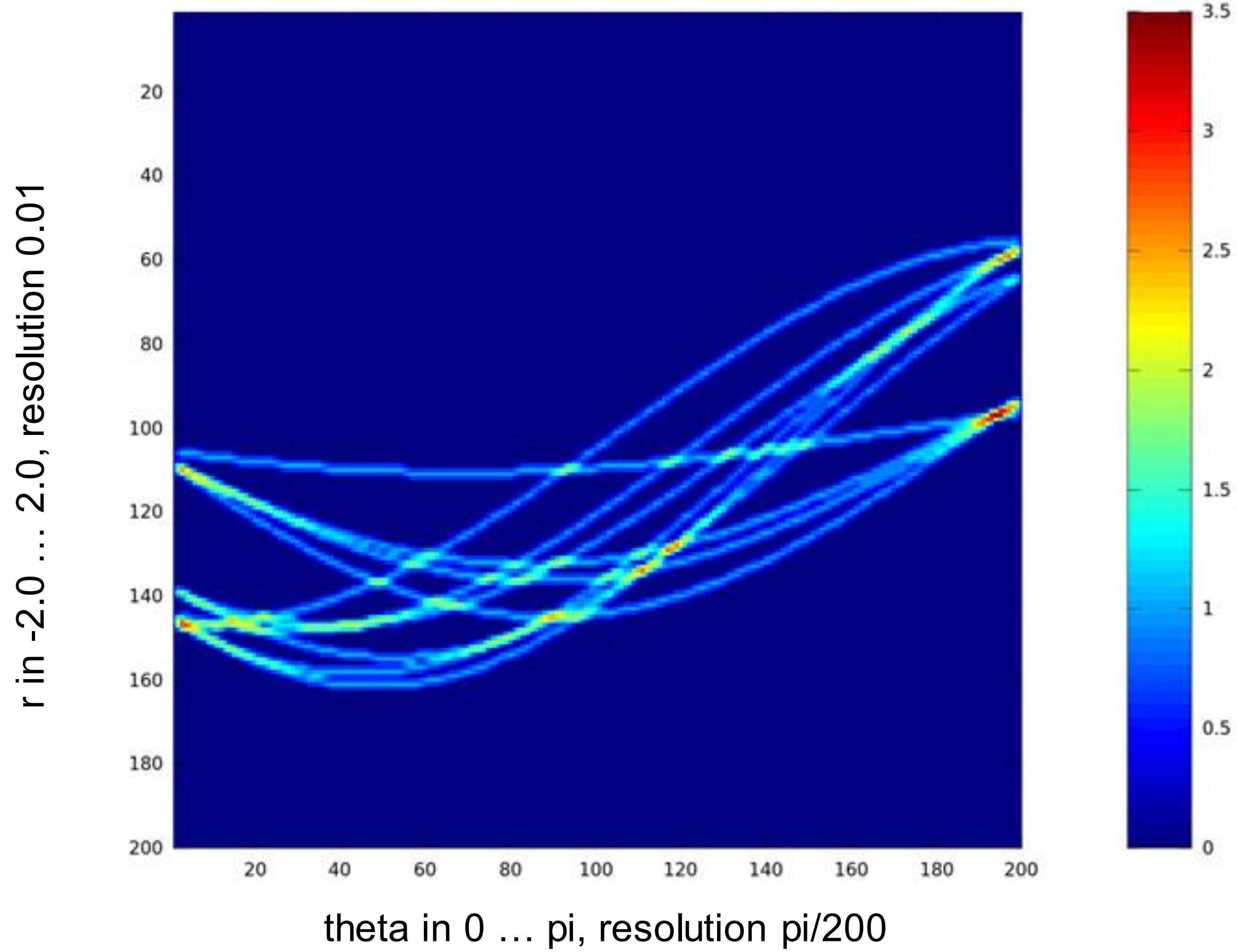
Step by step accumulation: iteration 1



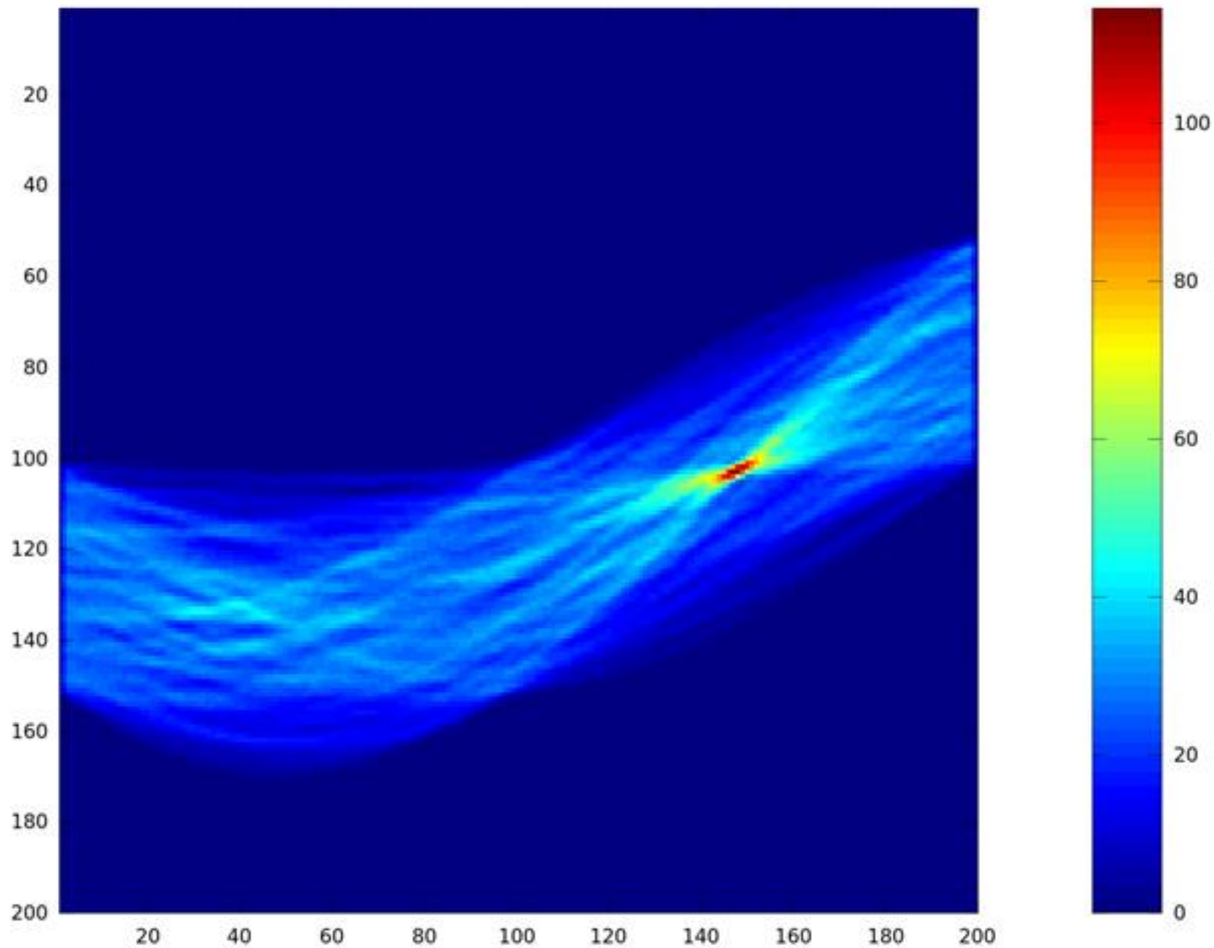
Step by step accumulation: iteration 2



Step by step accumulation: iteration 10



Step by step accumulation: final

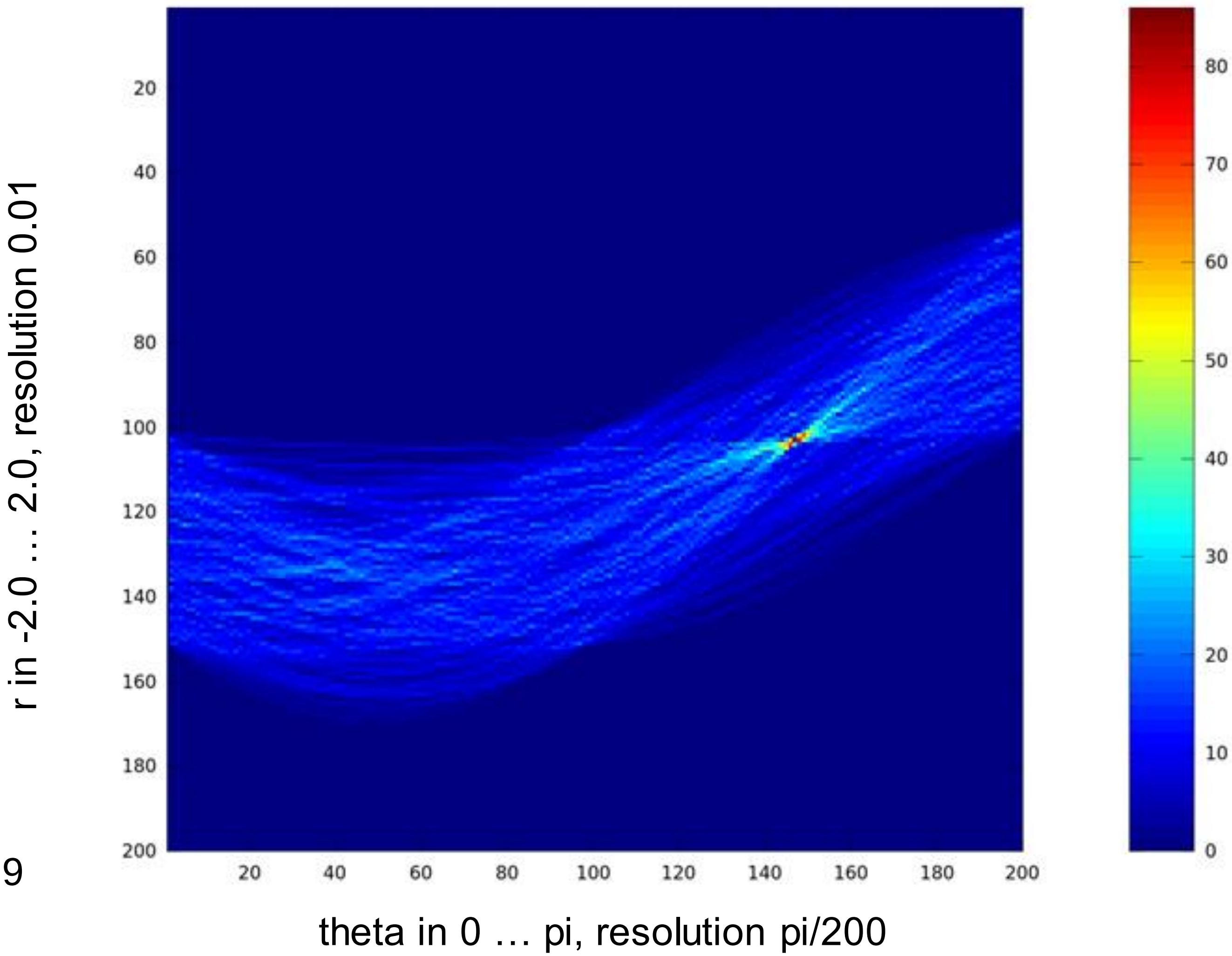


r in -2.0 … 2.0, resolution 0.01

theta in 0 … pi, resolution pi/200

r = 0.04
theta = 2.29
a = 0.88
b = 0.05

Step by step accumulation: final, no smoothing



r = 0.04
theta = 2.29
a = 0.88
b = 0.05

Application to circles

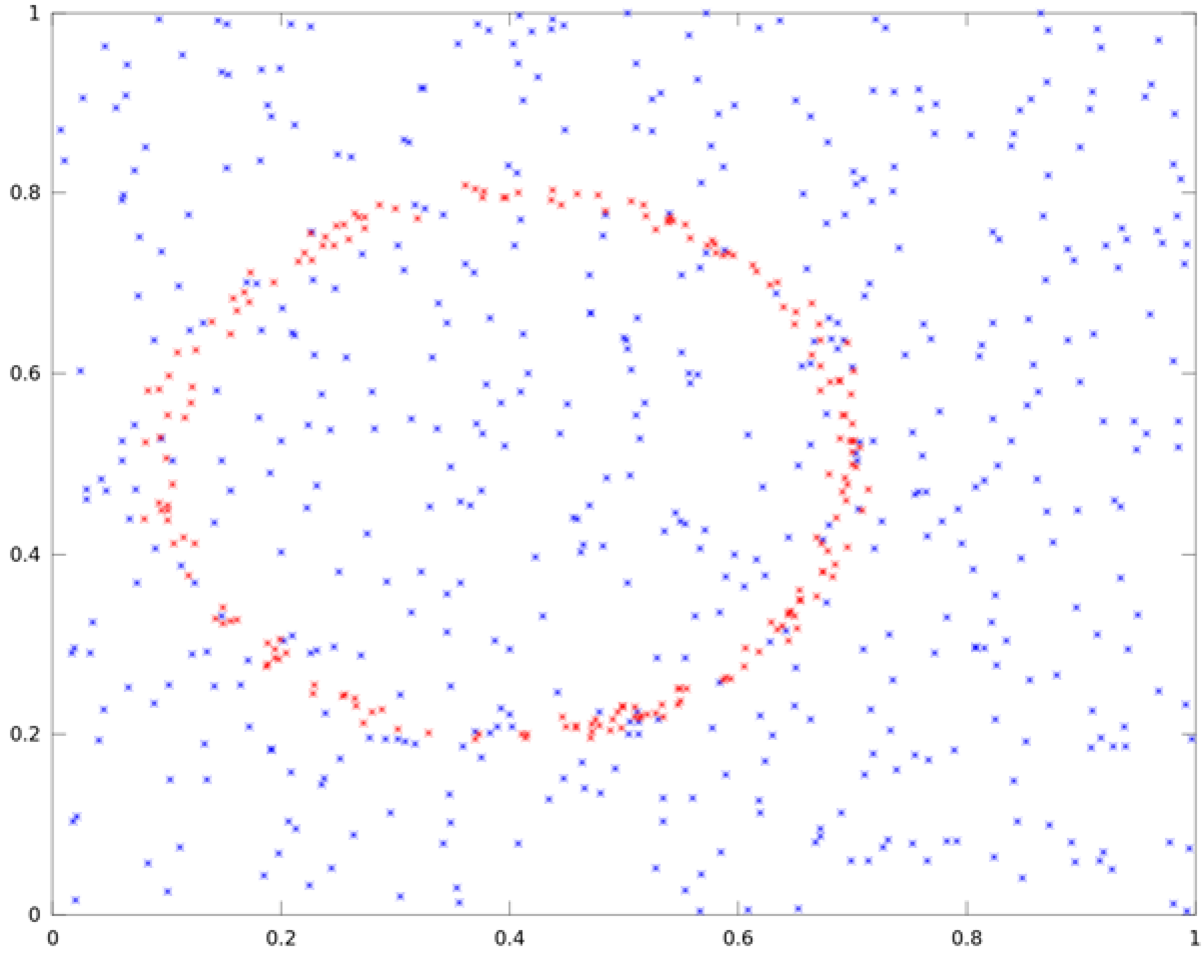
- Circle equation

$$(x-a)^2+(y-b)^2 = r^2$$

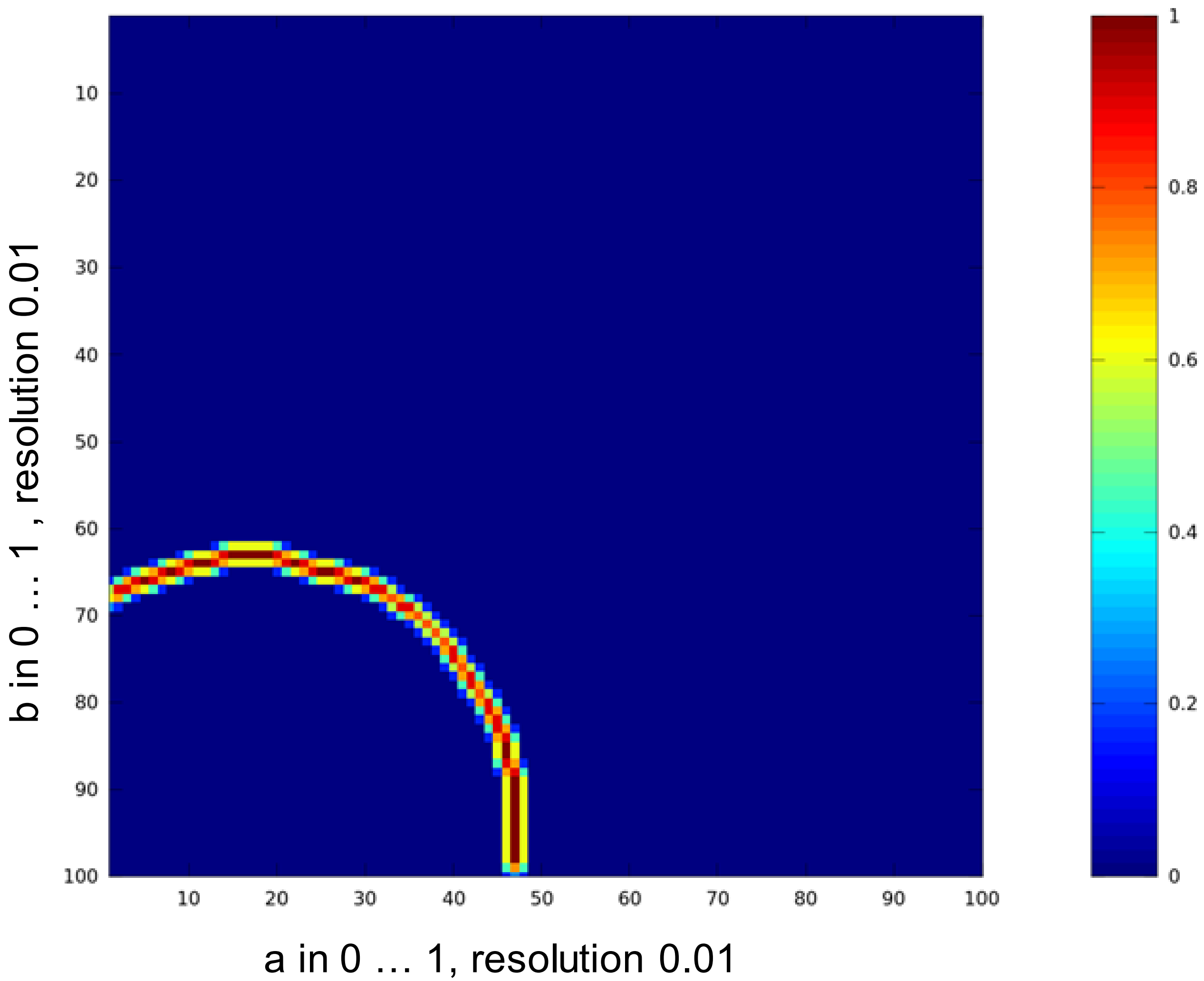
  - Parameter center (a,b), radius r
  - If we assume r known, it is also a circle in (a,b) space

- About r:
  - r influences the possible/expected number of votes per parameter set.
  - Either search in 3D space (a,b,r)
  - Or run a search in (a,b) space for multiple candidate r

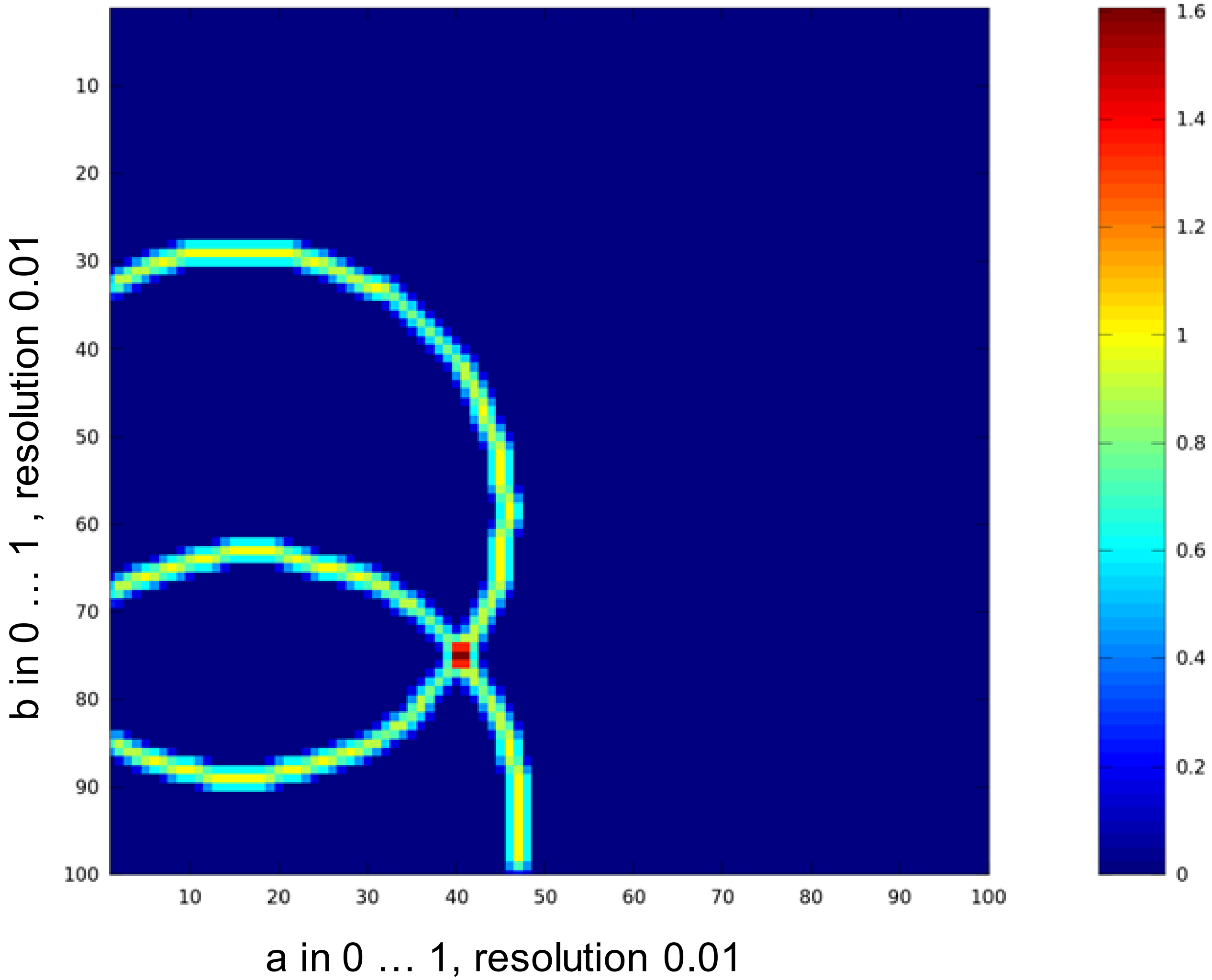
Context



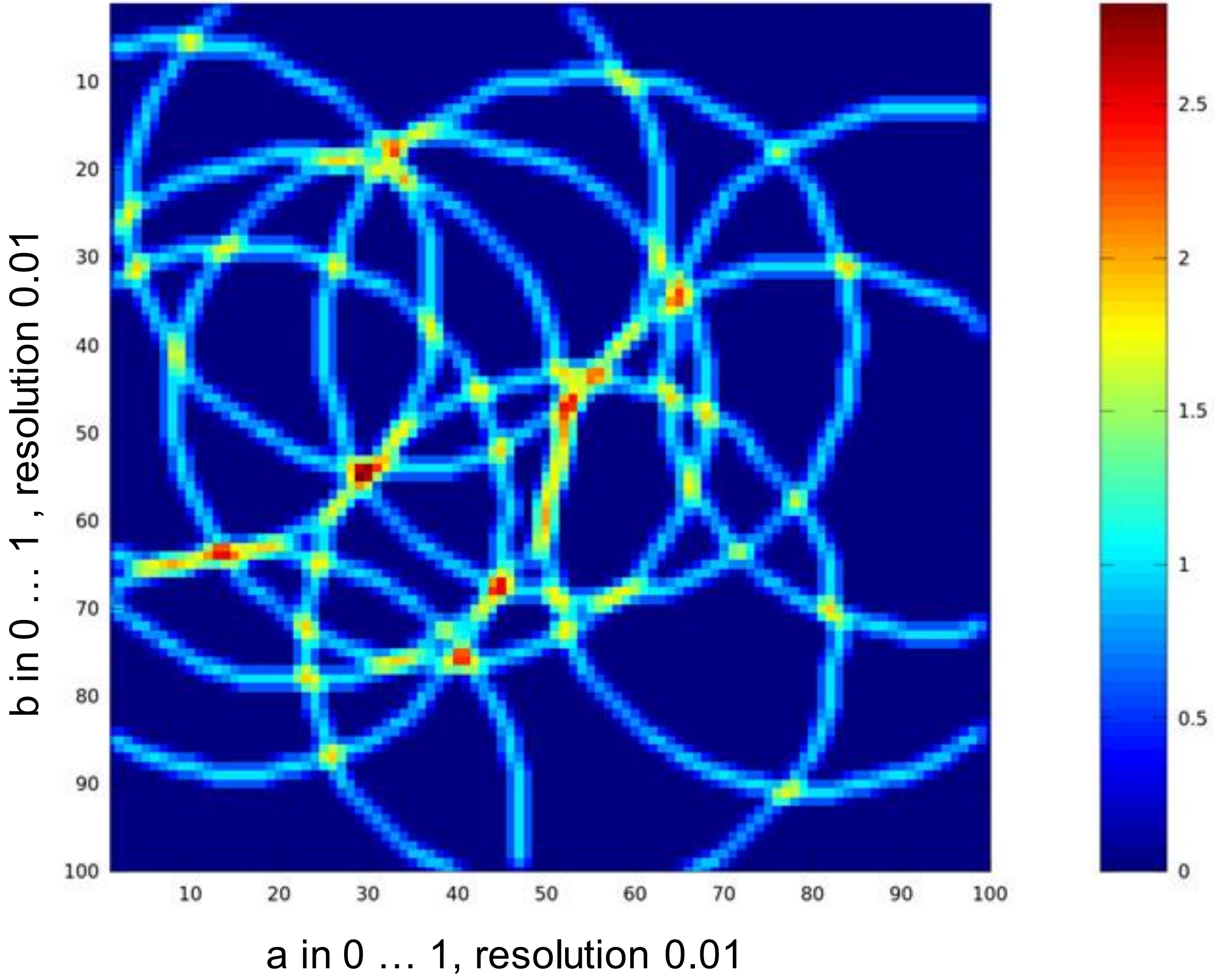
Step by step accumulation: iteration 1



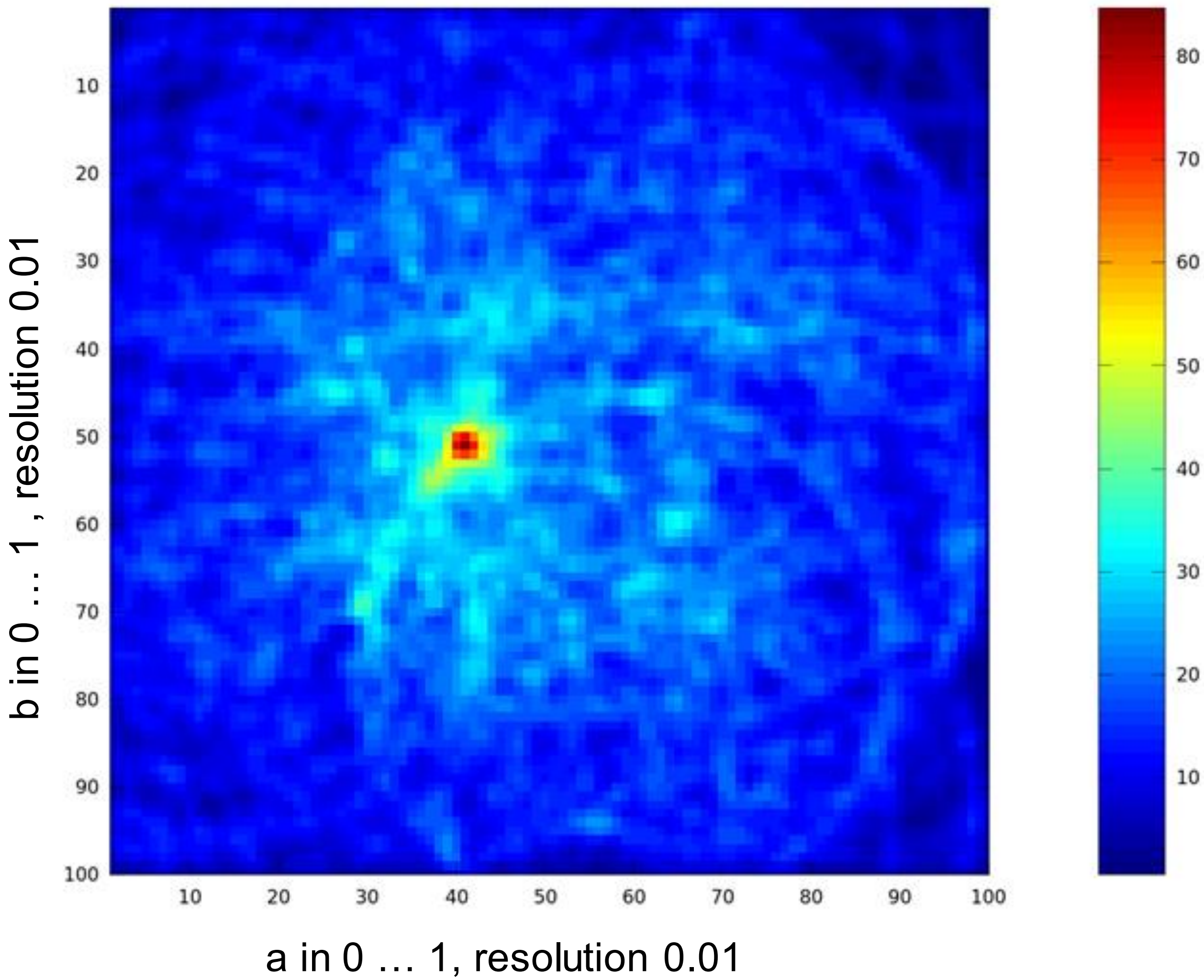
Step by step accumulation: iteration 2



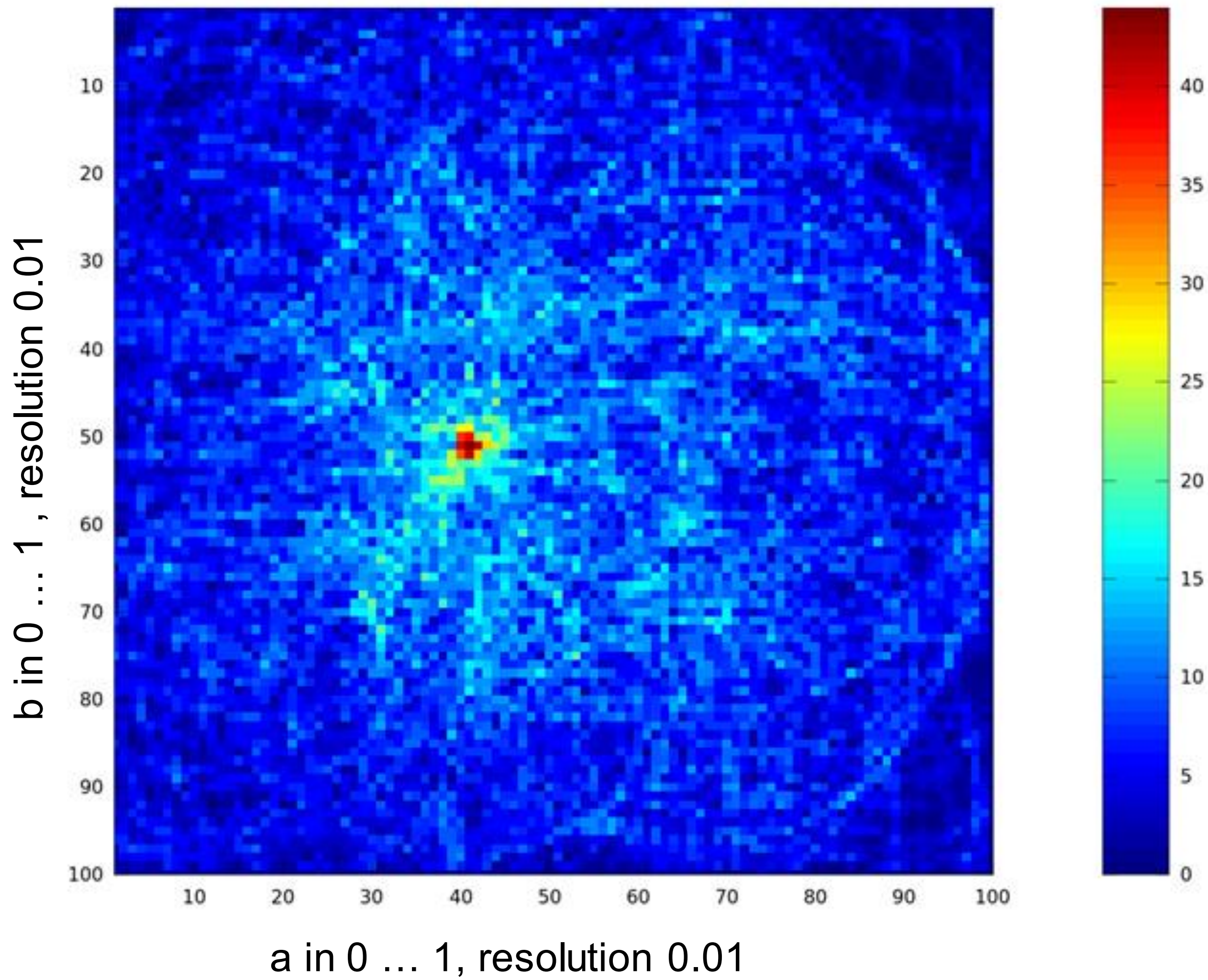
Step by step accumulation: iteration 3



Step by step accumulation: final

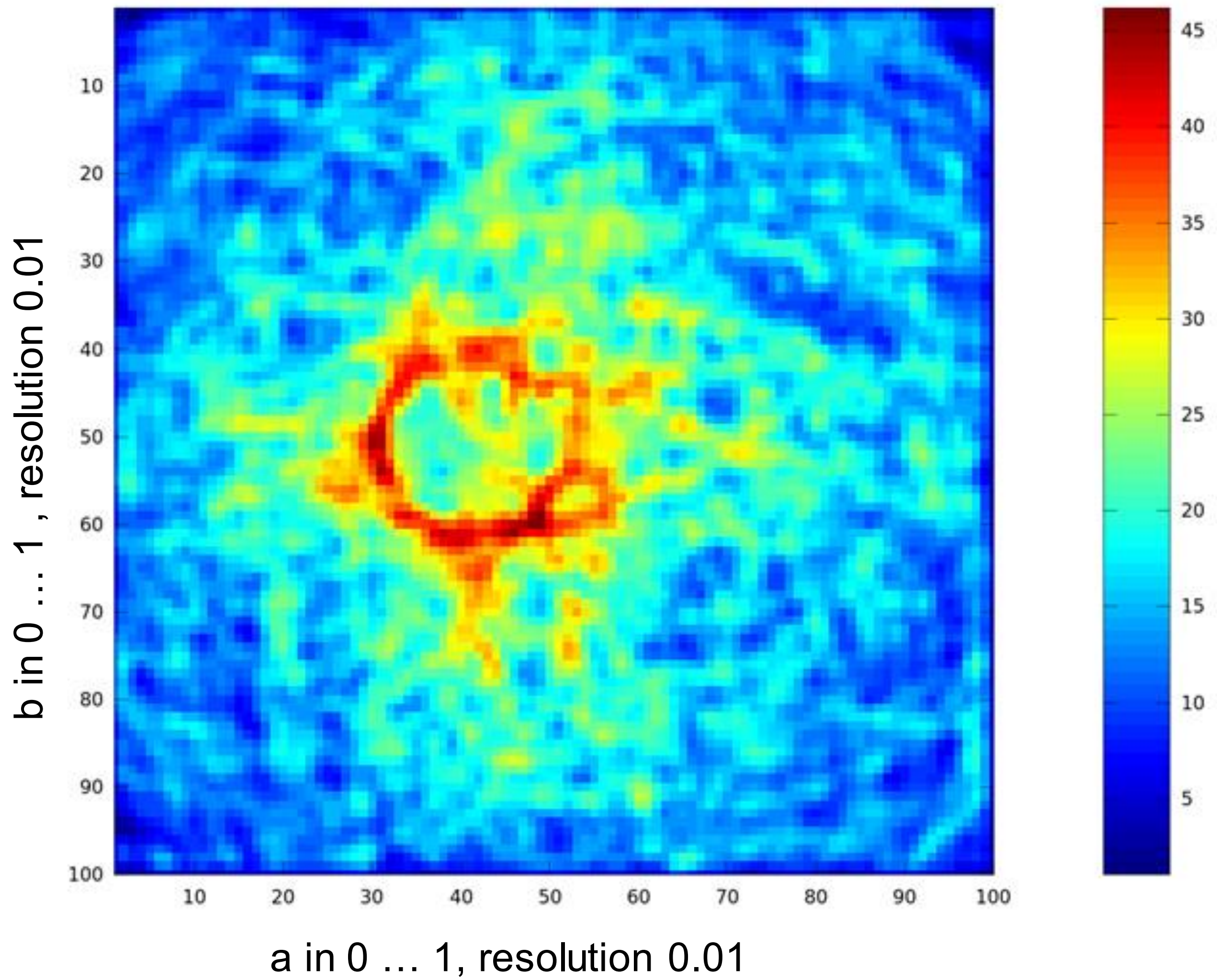


Step by step accumulation: final, no smoothing



Step by step accumulation: wrong radius

R=0.4 instead of R=0.3



Hough Transform: summary

- Very robust detector
  - Low sensitivity to noise
  - Low sensitivity to outliers
- Only really works for low number of parameters
  - Exponential memory requirement
  - Needs a bounded parameter space
- Not very often used for real applications, except with a strong prior reducing the search space
  - Looking for nearly vertical lines
  - Looking for circles of know radius

Model fitting using RANSAC

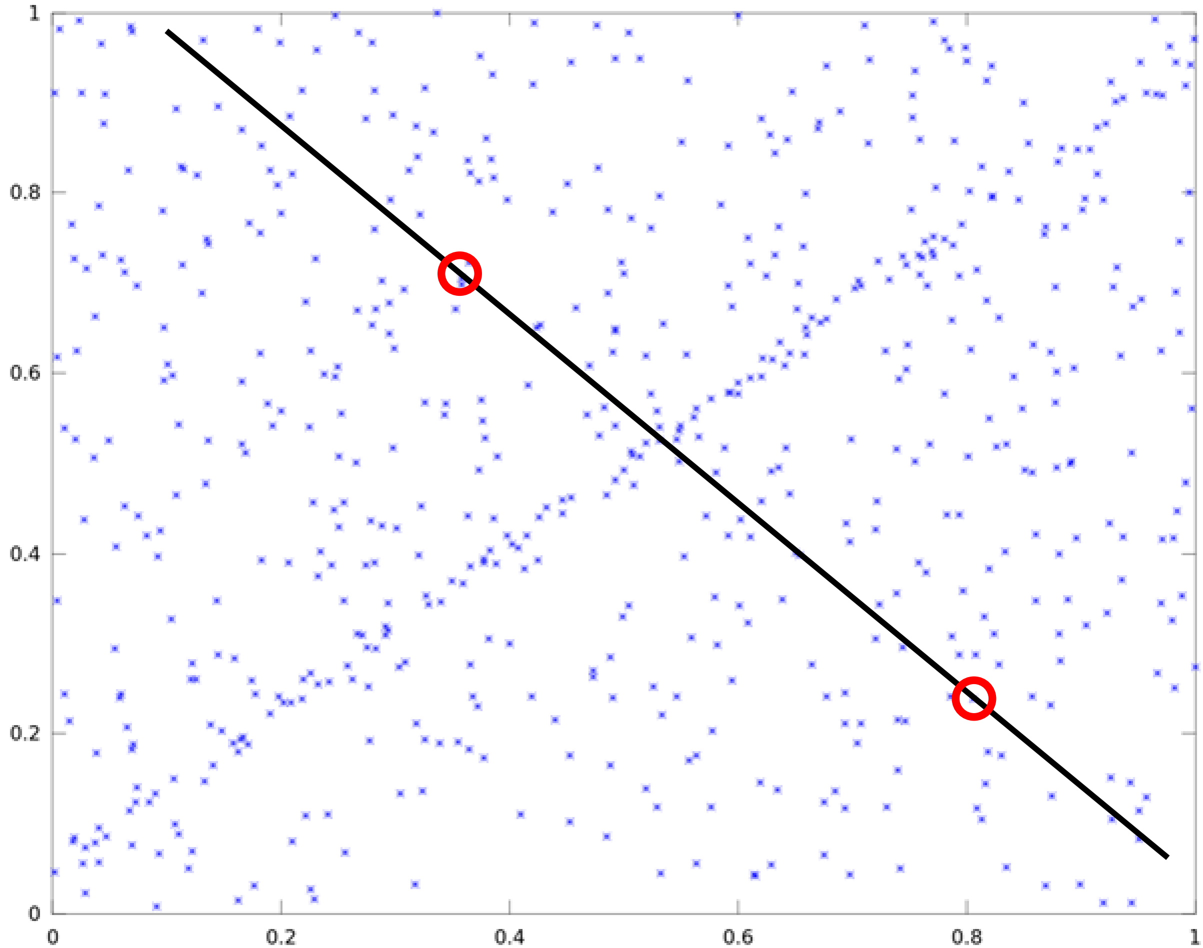
From Hough Transform to RANSAC

- How to deal with
  - Higher number of parameters
  - Possibly unbounded parameters
  - Limited memory requirements
- Accepting that
  - Probabilistic guarantees are sometimes enough
- RANSAC
  - Random Sampling Consensus
  - Intelligent sampling of the parameter space

RANSAC principle

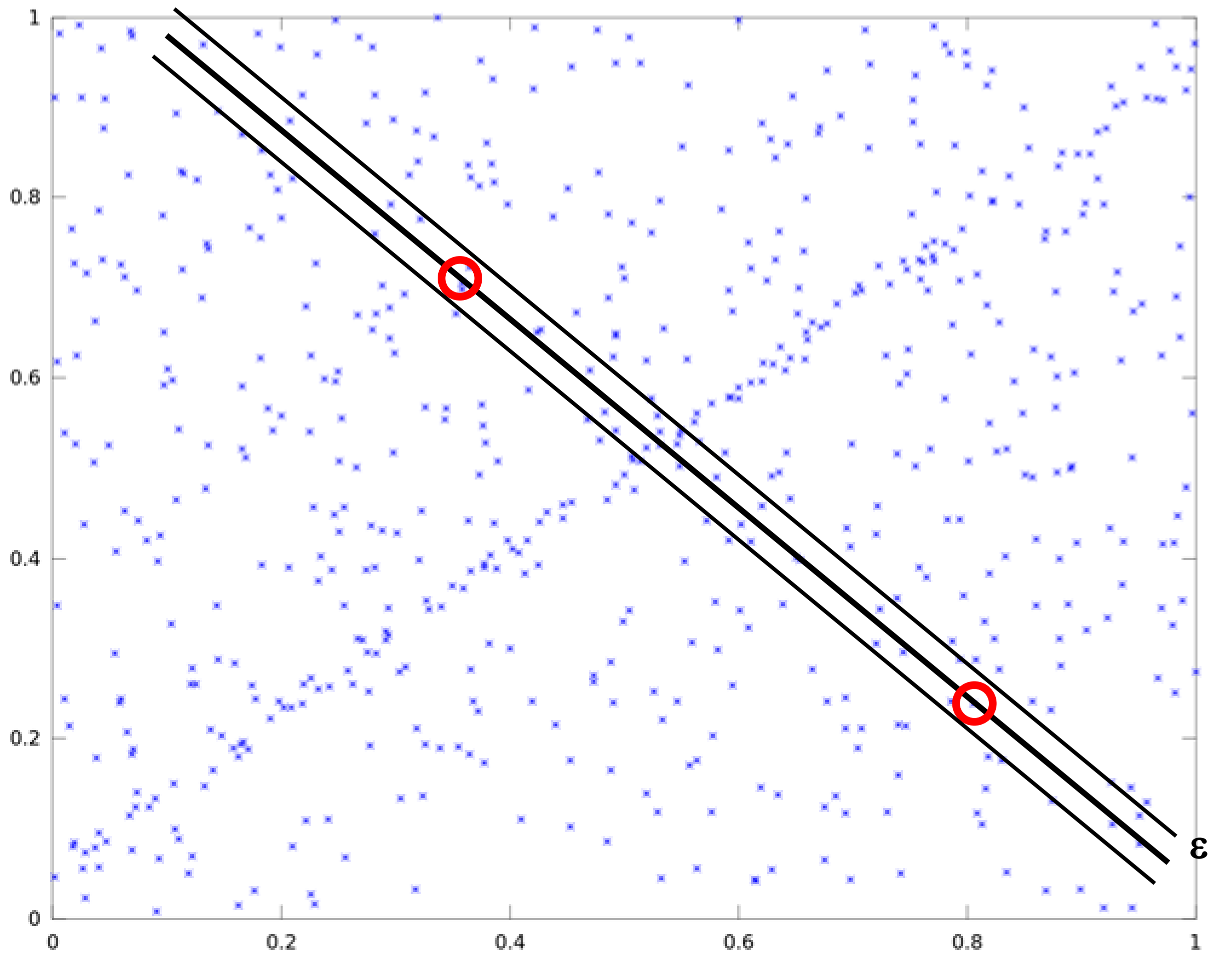
- Objective:
  - Estimate a model with p parameters using n data points
- Assumption:
  - Using q data points ($q << n$) , a closed form of the p parameter can be estimated (2D line: $q = 2$)
- Repeat enough time:
  - Sample q data points and estimate model $M(q)$
  - Count the number S of data points consistent with the model ($M(q)(x_i) < \varepsilon$)
  - If S is better that previous score, keep this model

Example: $S_{init} = 0$



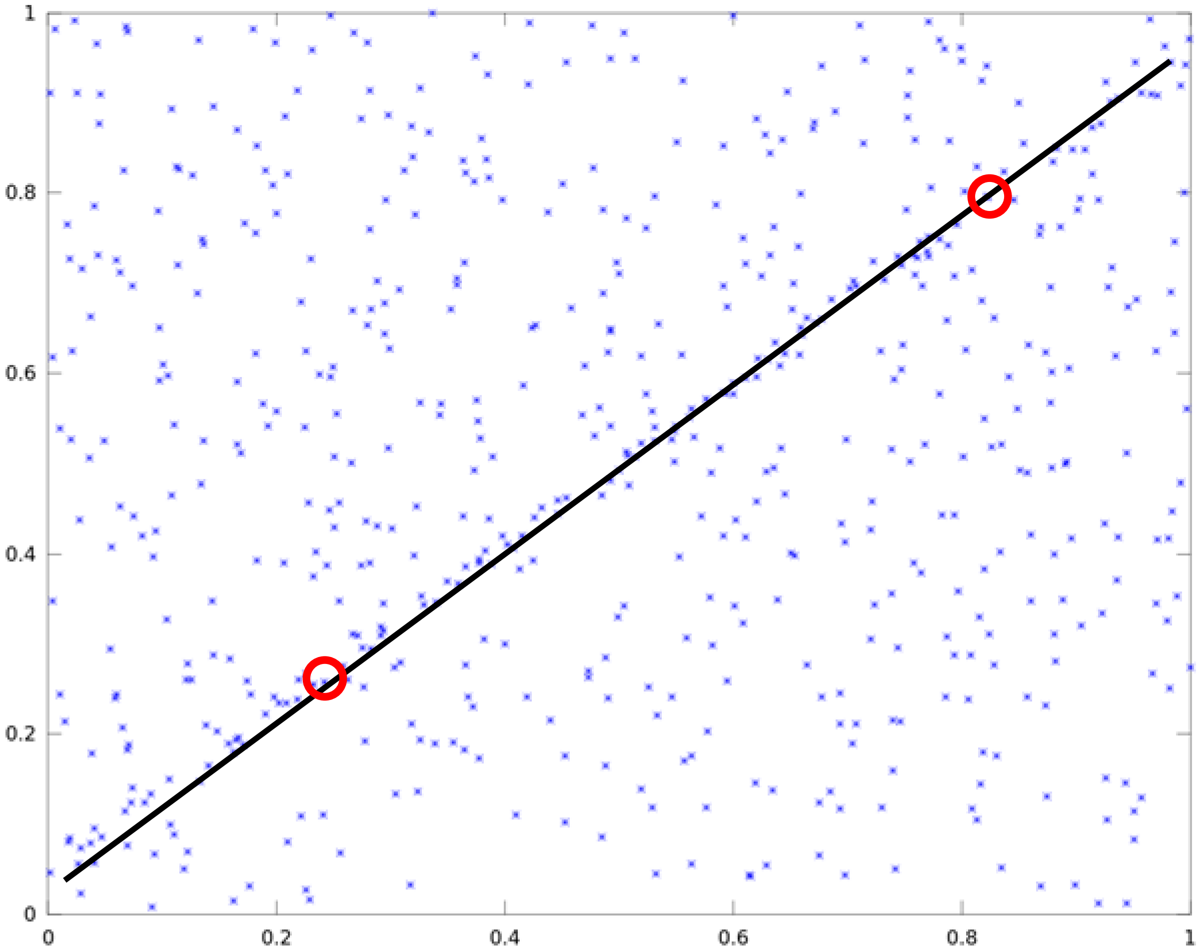
Georgia Tech Lorraine

Example: $S_{init} = 0$

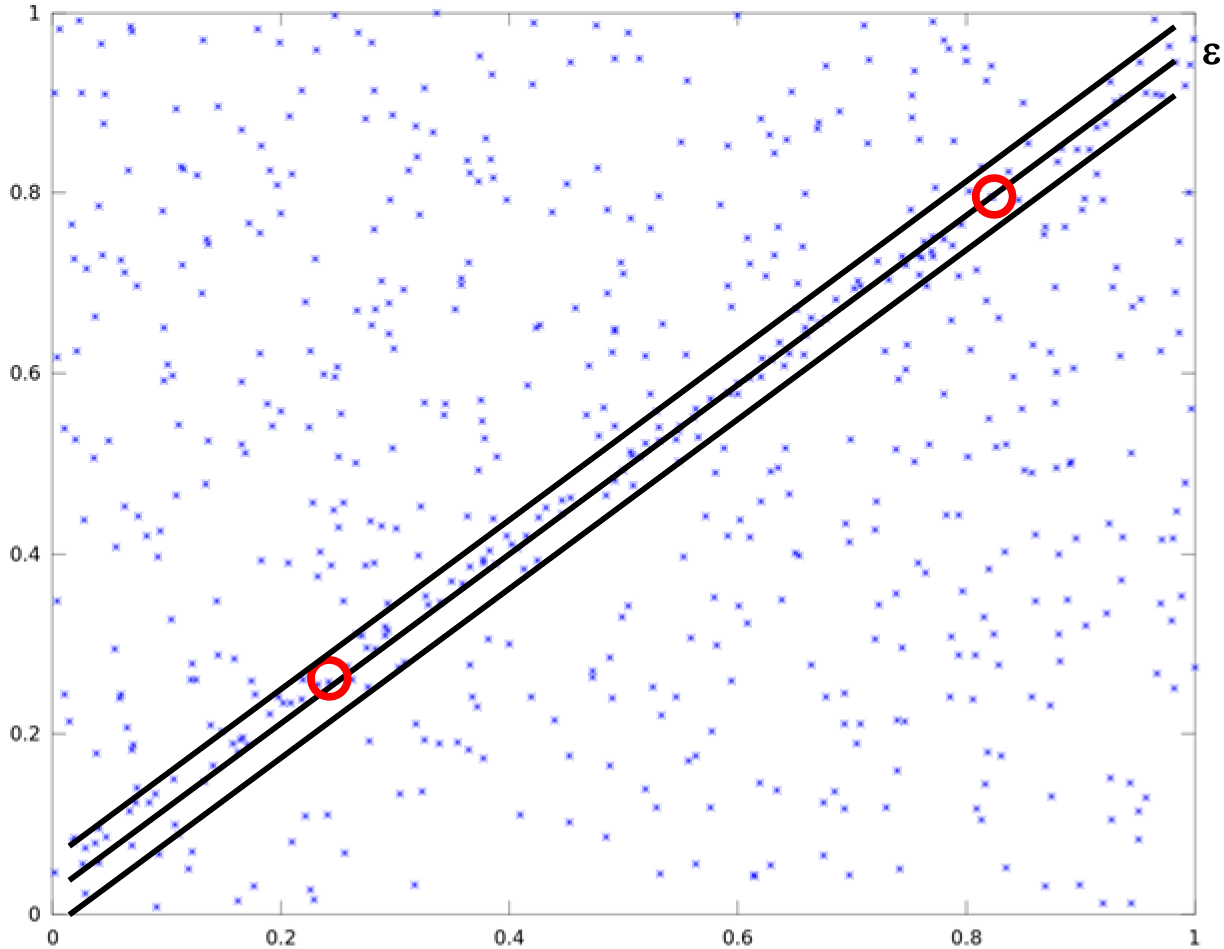


S = 31

Example: $S_{best} = 31$



Example: S_{best} = 31



S = 120

Alternative scoring

- Counting the number of consistent data point is equivalent to minimising:

$$S(M)=\sum_{i=1}^{n}\rho(M(q_i))$$

- Where: $$\rho(M(q_i))=\begin{cases}0 & |M(q(i))|<\delta\\ 1 & otherwise\end{cases}$$

- Why not something smoother?

$$\rho(M(q_i))=\begin{cases}M(q(i)) & |M(q(i))|<\delta\\ \delta & otherwise\end{cases}$$

- MSAC: M-Estimator Sample Consensus

Number of iterations

- Let *P* be the probability of selecting a good subset of q data point.
- Let *h* be the number of iterations
- Let ε be the desired probability of having sampled at least one good subset after h iteration.
- We want: $(1-P)^h \leq \grave{o}$
- Hence: $h \geq \left\lceil \frac{\log \grave{o}}{\log(1-P)} \right\rceil$ where $\log(1-P) \leq 0$
- How to compute P?

How to compute P?

- Assume we know the number of inliers N_I
- If all points have the same probability of being selected,

$$P = \frac{\binom{N_I}{q}}{\binom{N}{q}} = \frac{N_I!(N-q)!}{N!(N_I-q)!} = \prod_{i=1}^{q-1} \frac{N_I - i}{N - i}$$

- If $N \gg q$ and $N_I \gg q$

$$P = \prod_{i=1}^{q-1} \frac{N_I - i}{N - i} \approx \left(\frac{N_I}{N} \right)^q$$

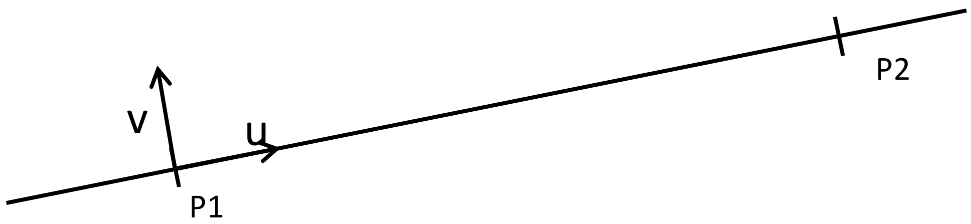
- But we don’t know N_I…

How to compute P?

- Let $\hat{N}_I$ be the biggest number of inliers seen so far.
- We have: $$\hat{N}_I \leq N_I \Rightarrow P(\hat{N}_I) \leq \mathrm{P}(\mathrm{N}_I)$$
$$\Rightarrow \log(1-P(\hat{N}_I)) \geq \log(1-P(N_I))$$
$$\Rightarrow \frac{1}{\log(1-P(\hat{N}_I))} \leq \frac{1}{\log(1-P(N_I))}$$
$$\Rightarrow \frac{\log \grave{o}}{\log(1-P(\hat{N}_I))} \geq \frac{\log \grave{o}}{\log(1-P(N_I))}$$
- Hence: $h = \left\lceil \frac{\log \grave{o}}{\log(1-P(\hat{N}_I))} \right\rceil$ is a conservative required number of iterations

Example 1: Finding lines

- Line model: $a \cdot x + b \cdot y + c = 0$
- Minimum number of points: 2



$$u = \frac{P_2 - P_1}{\| P_2 - P_1 \|}$$

$$a \cdot x + b \cdot y + c = 0 \Rightarrow u = [-b, a], c = -u \cdot P_1$$

- Fitness measure: distance from a point to the line: $$M(x, y \mid a, b, c) = \left| a \cdot x + b \cdot y + c \right|$$
  - This implies/requires $\left\| \begin{pmatrix} a \\ b \end{pmatrix} \right\| = 1$

Alternative model computation

- Line construction from projective geometry

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} P_1.x \\ P_1.y \\ 1 \end{pmatrix} \times \begin{pmatrix} P_2.x \\ P_2.y \\ 1 \end{pmatrix}$$

- Normalisation

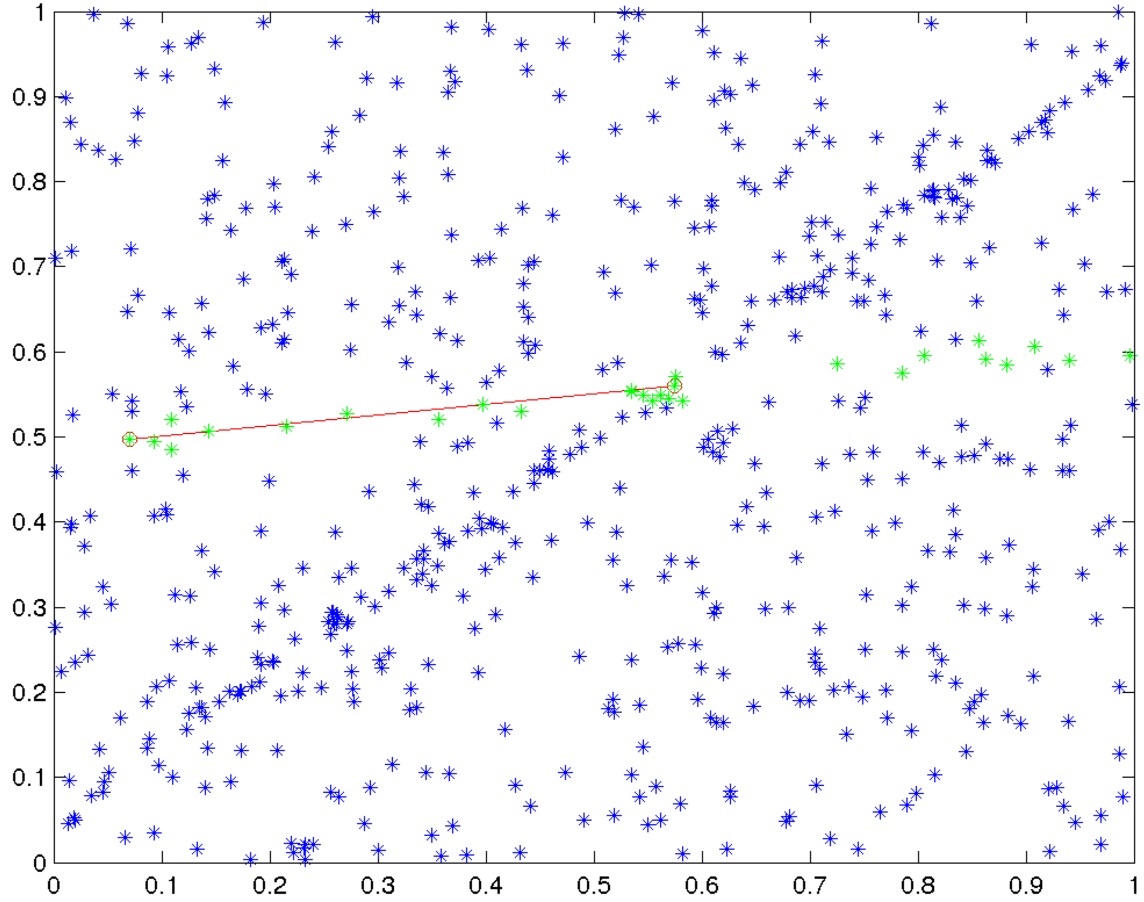
$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \frac{1}{\sqrt{a'^2 + b'^2}} \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix}$$

Matlab implementation

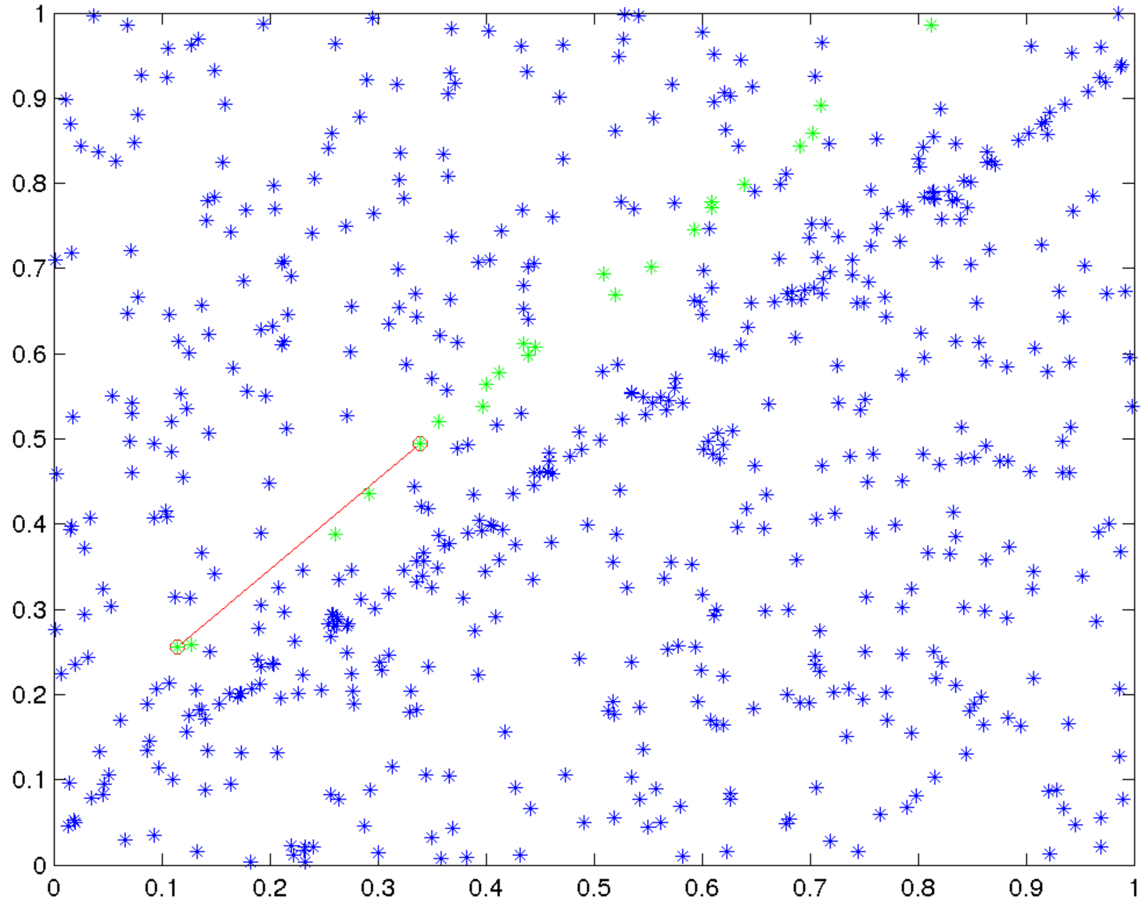
```
% Compute line model
param = line_param(P1,P2);

% Evaluate the model on all points
score_i = abs(param(1)*xl+param(2)*yl+param(3));
idx = find(score_i < max_error);

% Number of inliers
count = size(idx,1);

% Model score
score = sum(score_i(idx)) + (n-count)*max_error

if score < best_consensus
  best_consensus = score;
  best_param = param;
  best_inliers = count;
end
```

$$\rho(M(q_i)) = \begin{cases} M(q(i)) & |M(q(i))| < \delta \\ \delta & otherwise \end{cases}$$

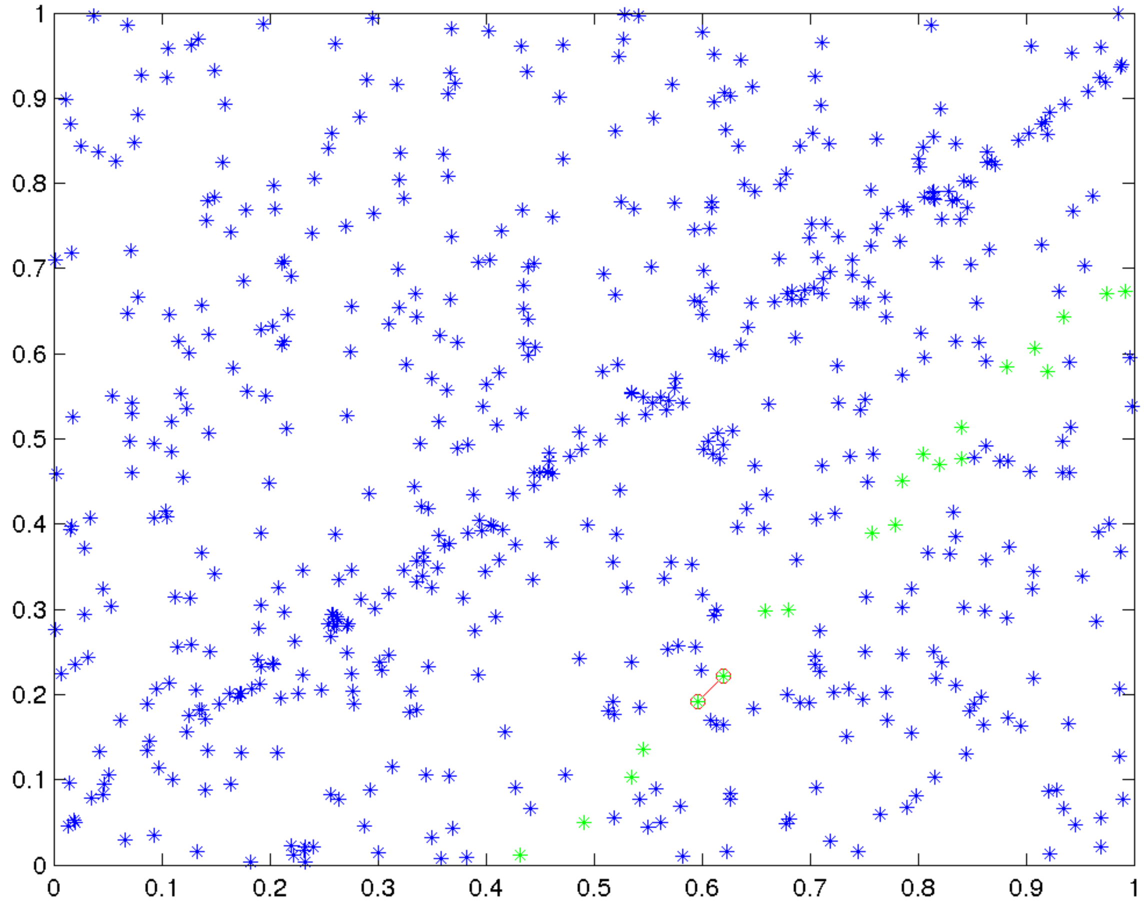
Example 1: Finding lines



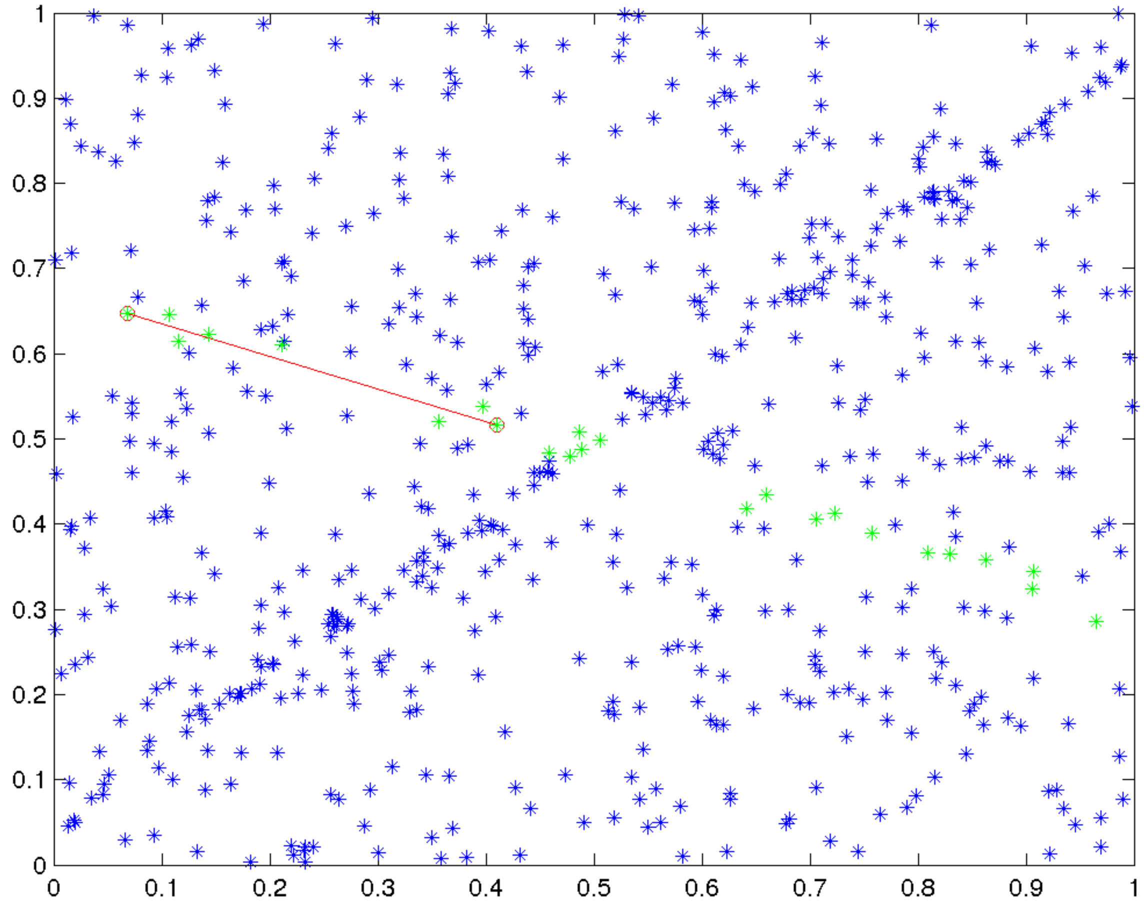
Example 1: Finding lines



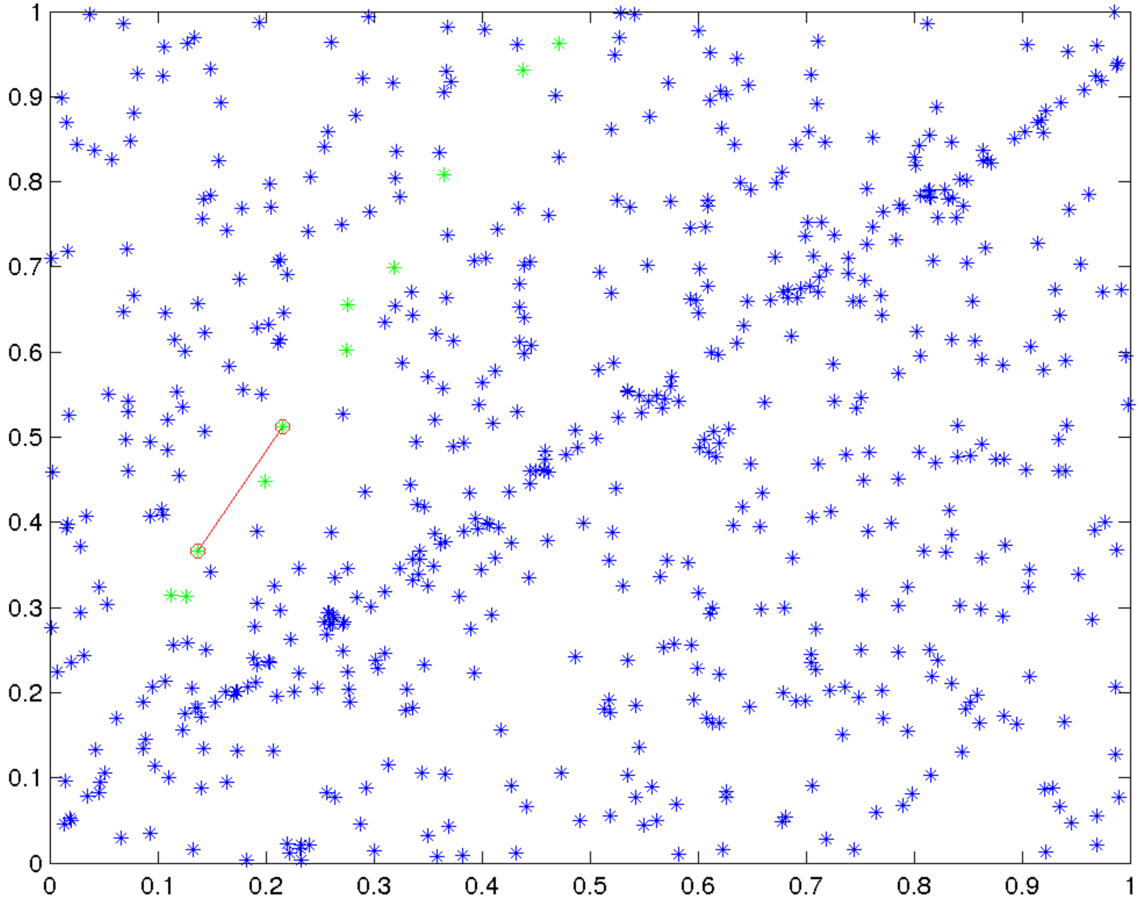
Example 1: Finding lines



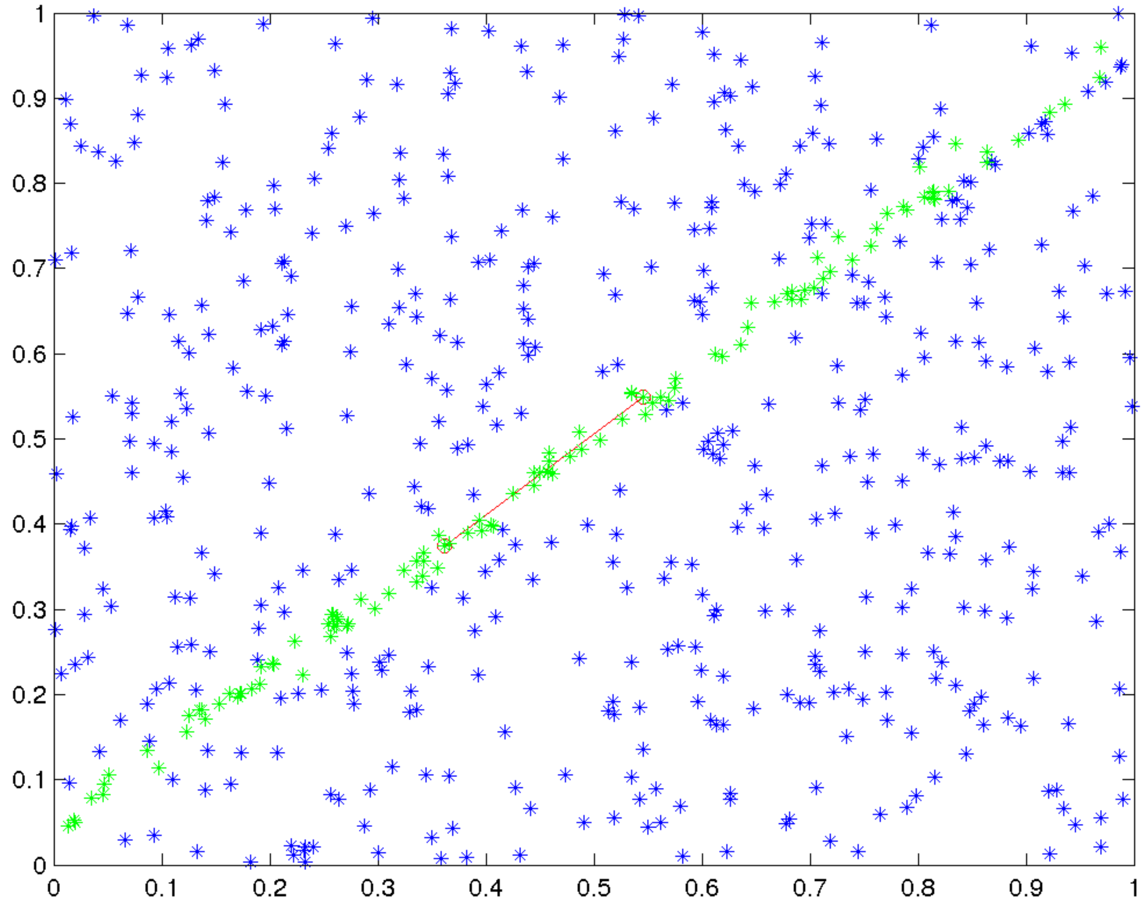
Example 1: Finding lines



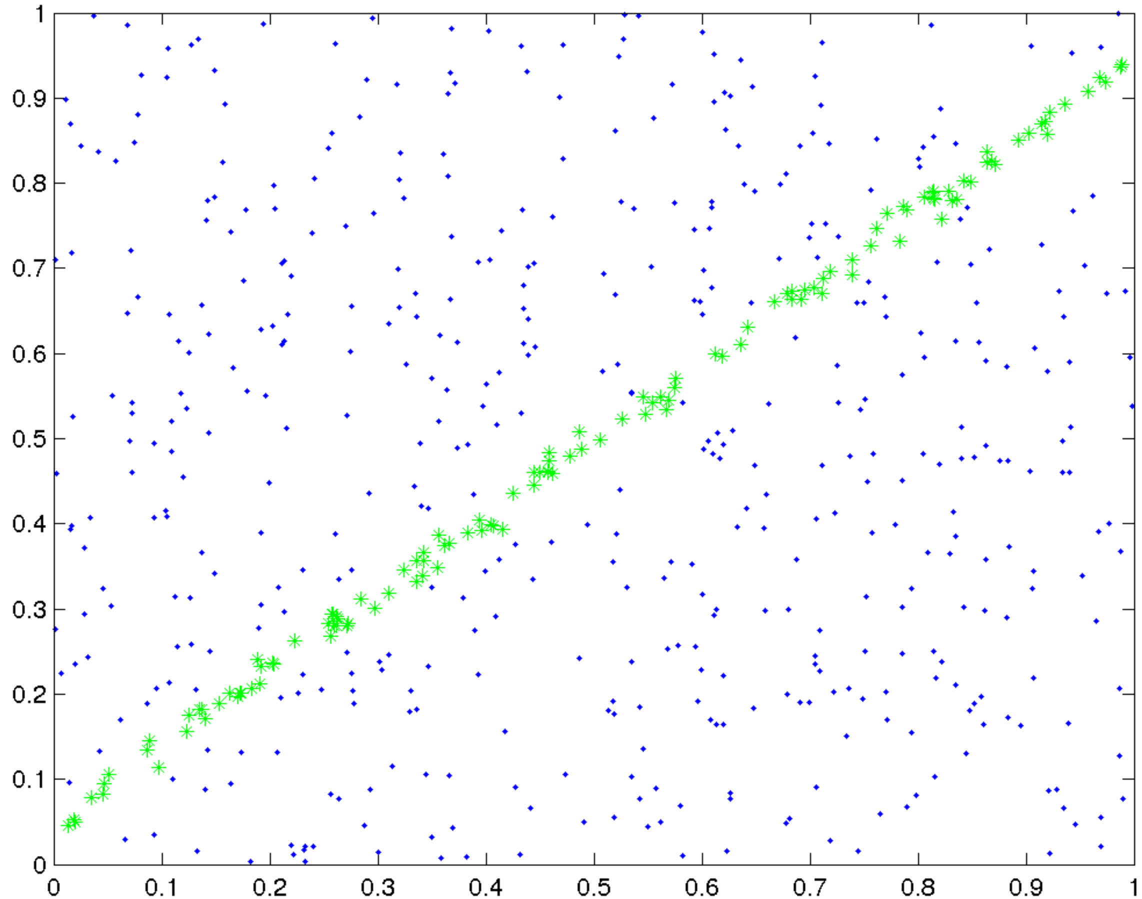
Example 1: Finding lines



Example 1: Finding lines



Example 1: Final Model



Max at
a = 0.8957 (0.90)
b = 0.0470 (0.05)

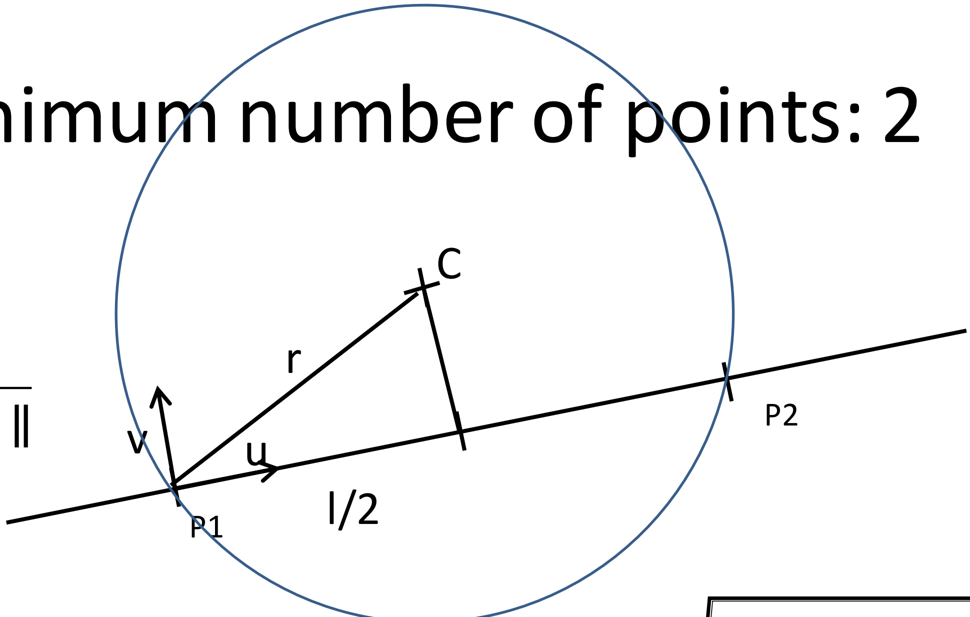
Score: 10.33
Inliers: 127

Example 2: Finding a circle, r known

- Circle model: $(x-a)^2+(y-b)^2 = \mathrm{r}^2$

- Minimum number of points: 2

$$u=\frac{P_2-P_1}{\|P_2-P_1\|}$$



$$C=P_1+\frac{l}{2}u+\sqrt{r^2-\left(\frac{l}{2}\right)^2}\,v$$

Example 2: Finding a circle, r known

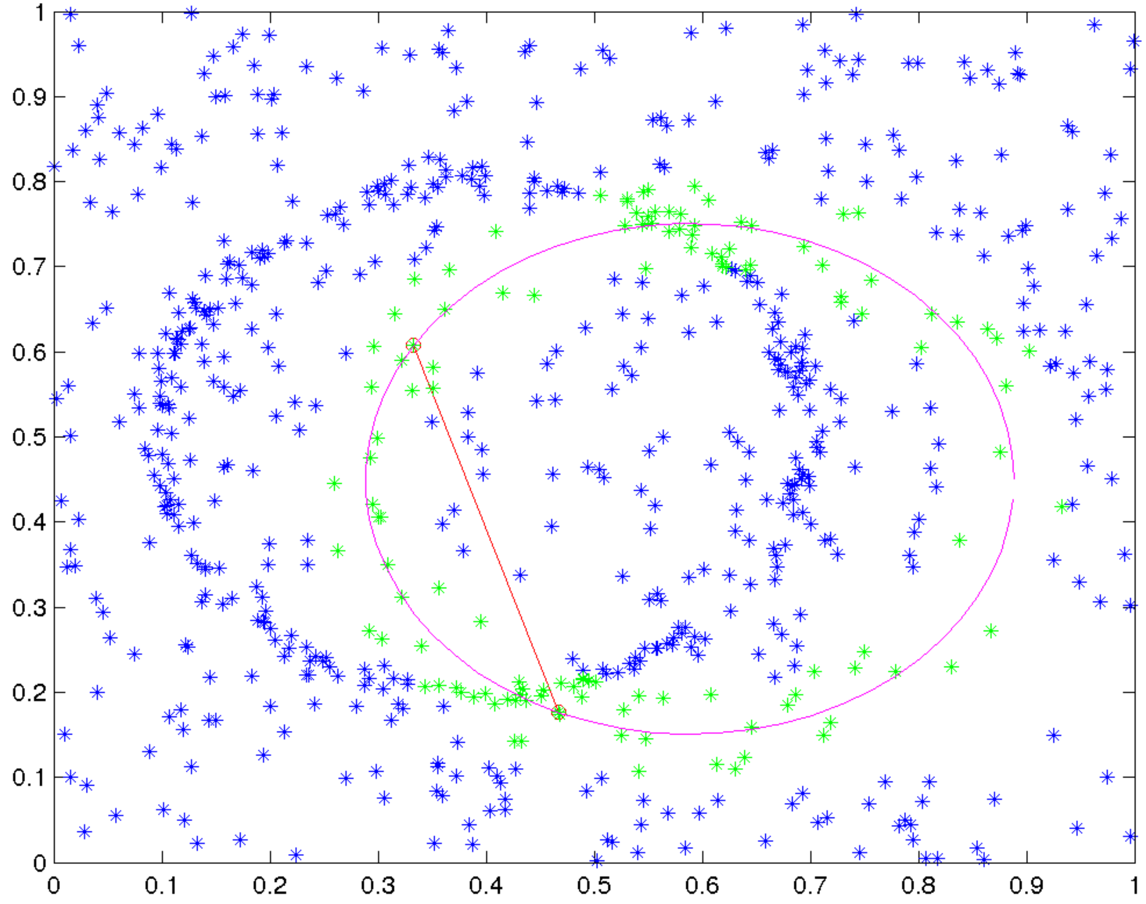
- Circle model: $(x-a)^2+(y-b)^2 = \mathrm{r}^2$

- Fitness measure: distance from a point to the circle:

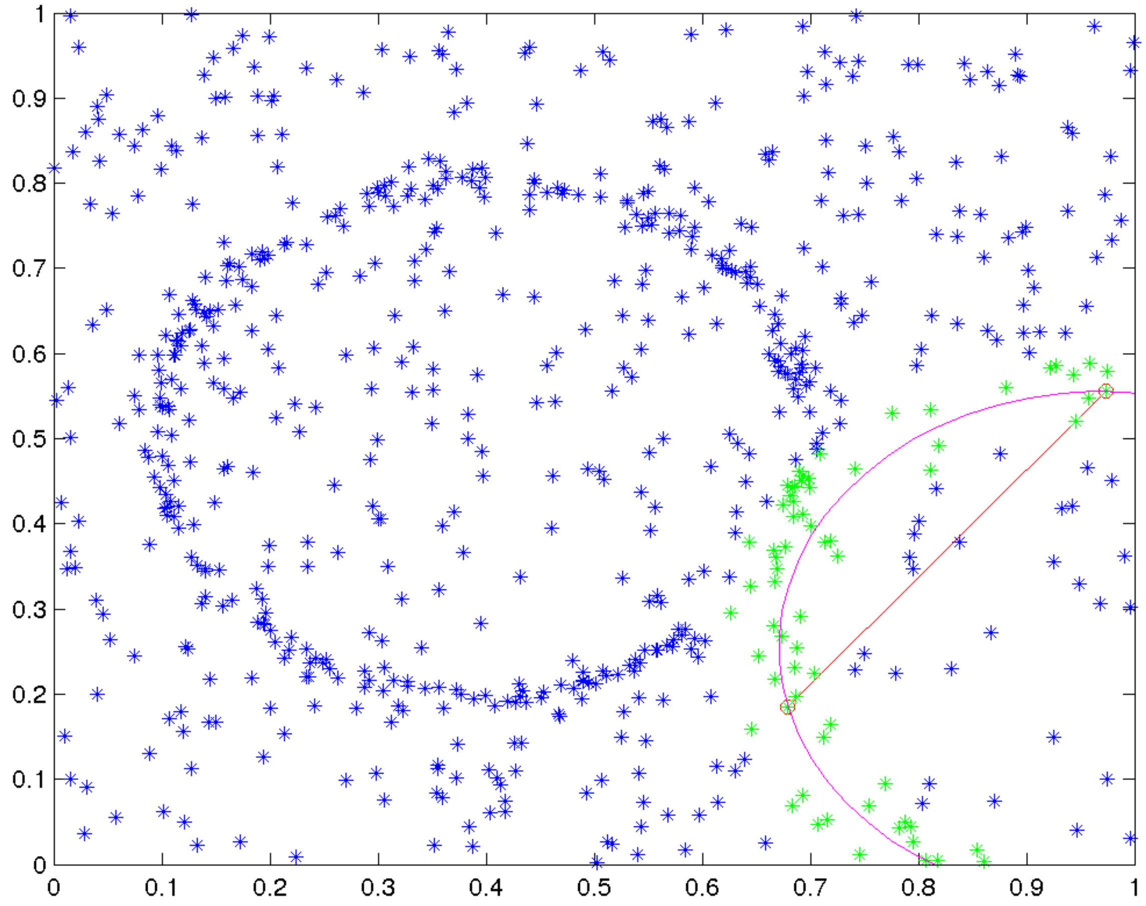
$$M(x,y\,|\,a,b)=\left|r-\sqrt{(x-a)^2+(y-b)^2}\right|$$

  – The sqrt is not strictly necessary, but gives M a metric meaning.

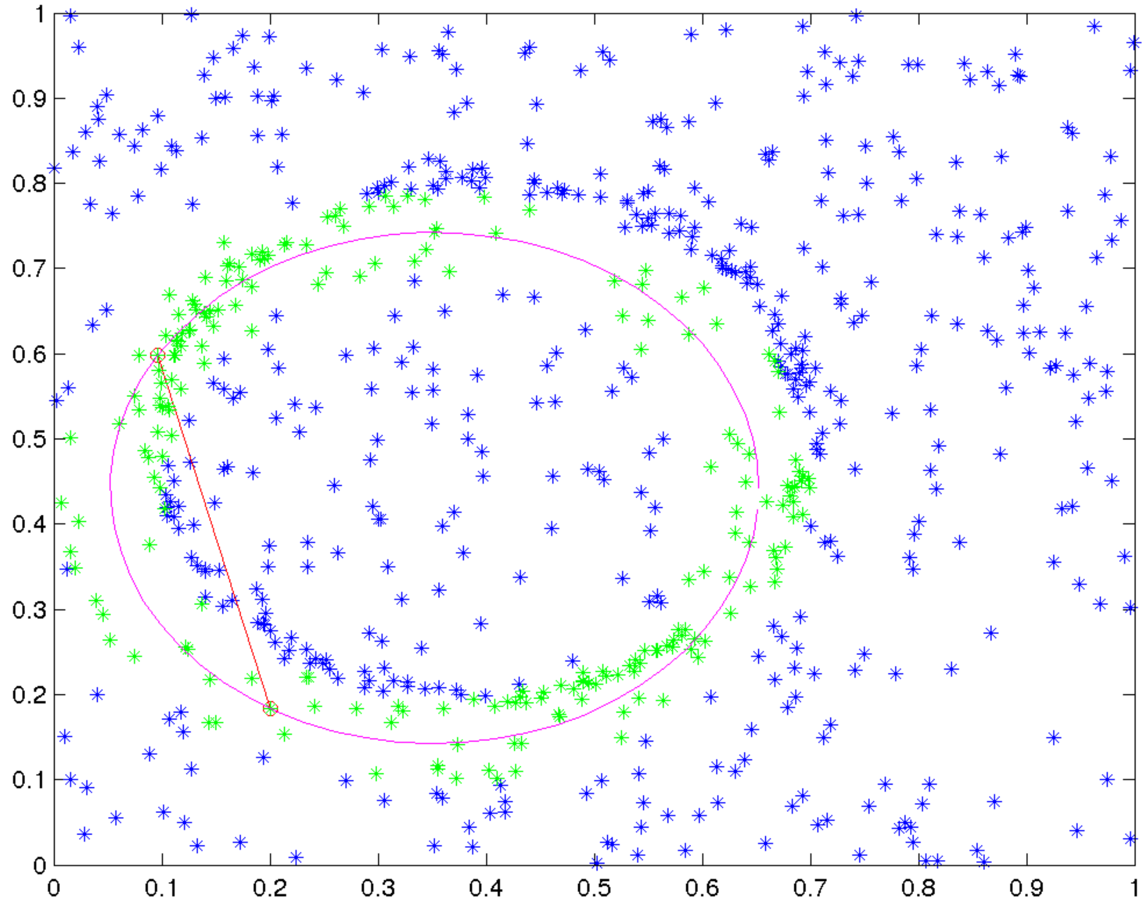
Example 2: Finding a circle, r known



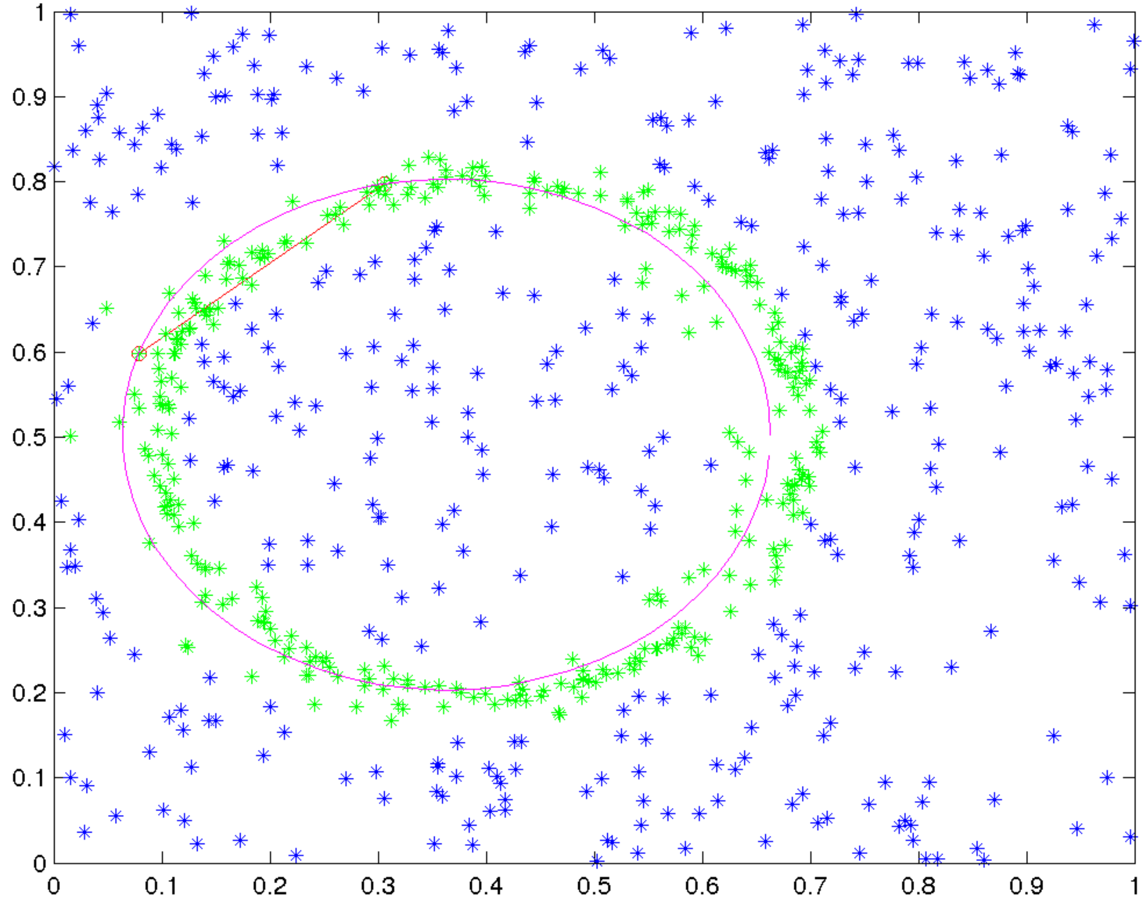
Example 2: Finding a circle, r known



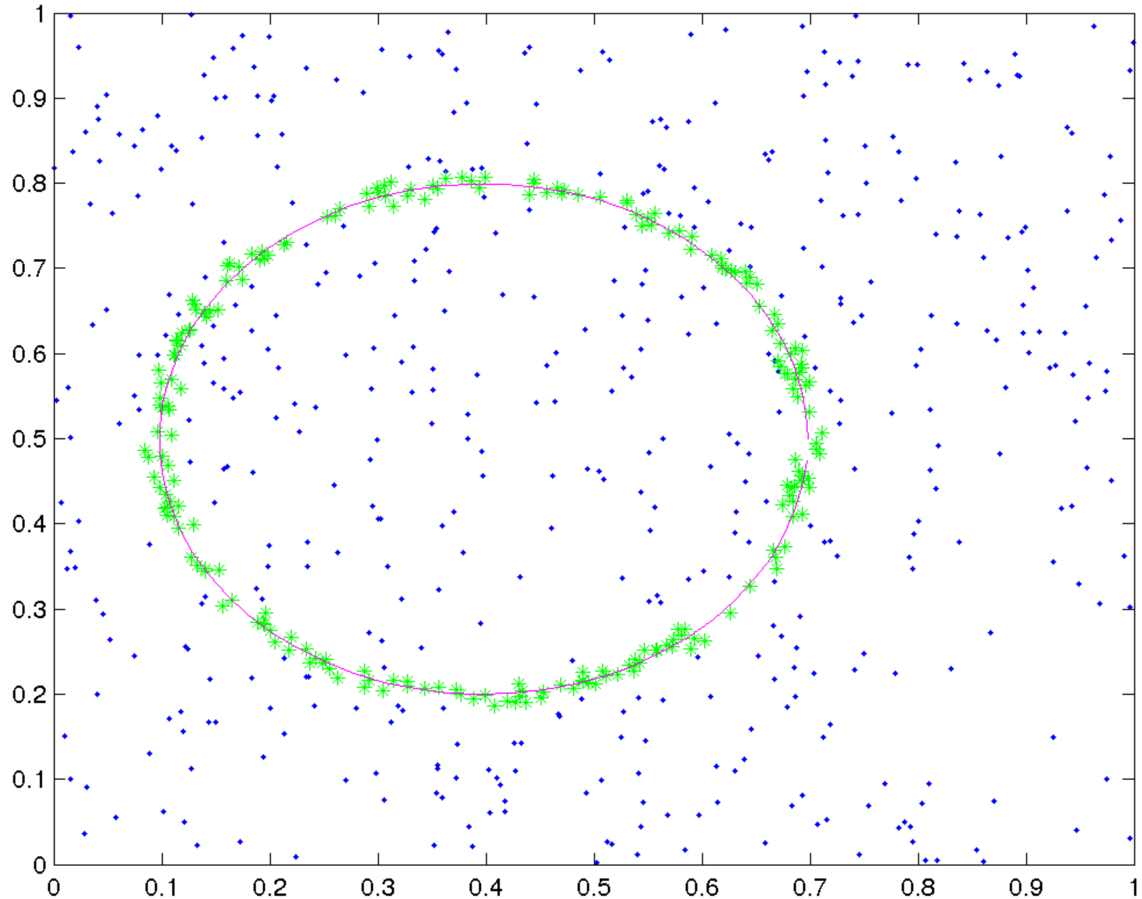
Example 2: Finding a circle, r known



Example 2: Finding a circle, r known



Example 2: Final Model



Max at
a = 0.4026 (0.40)
b = 0.5022 (0.50)

Score: 18.76
Inliers: 318

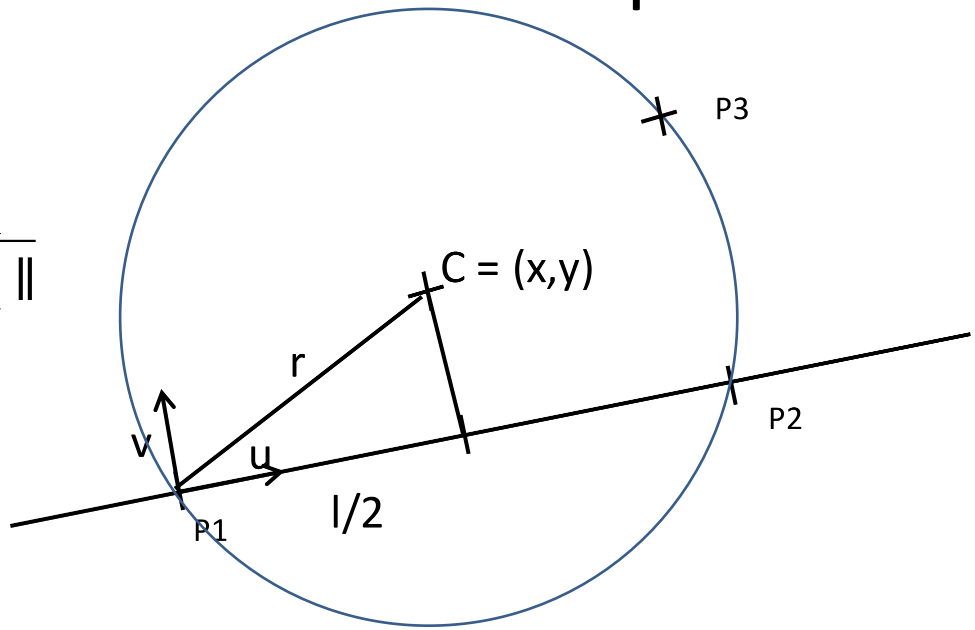
Example 3: Finding a generic circle

- Circle model:

$$(x-a)^2+(y-b)^2 = \mathrm{r}^2$$

- Minimum number of points: 3

$$u = \frac{P_2 - P_1}{\| P_2 - P_1 \|}$$



In reference frame (u,v):

$$x = \frac{l}{2}$$

$$y = \frac{1}{2}\frac{x_3^2 - l \cdot x_3 + y_3^2}{y_3}$$

$$r = \sqrt{x^2 + y^2}$$

Example 3: Finding a generic circle

- Circle model:

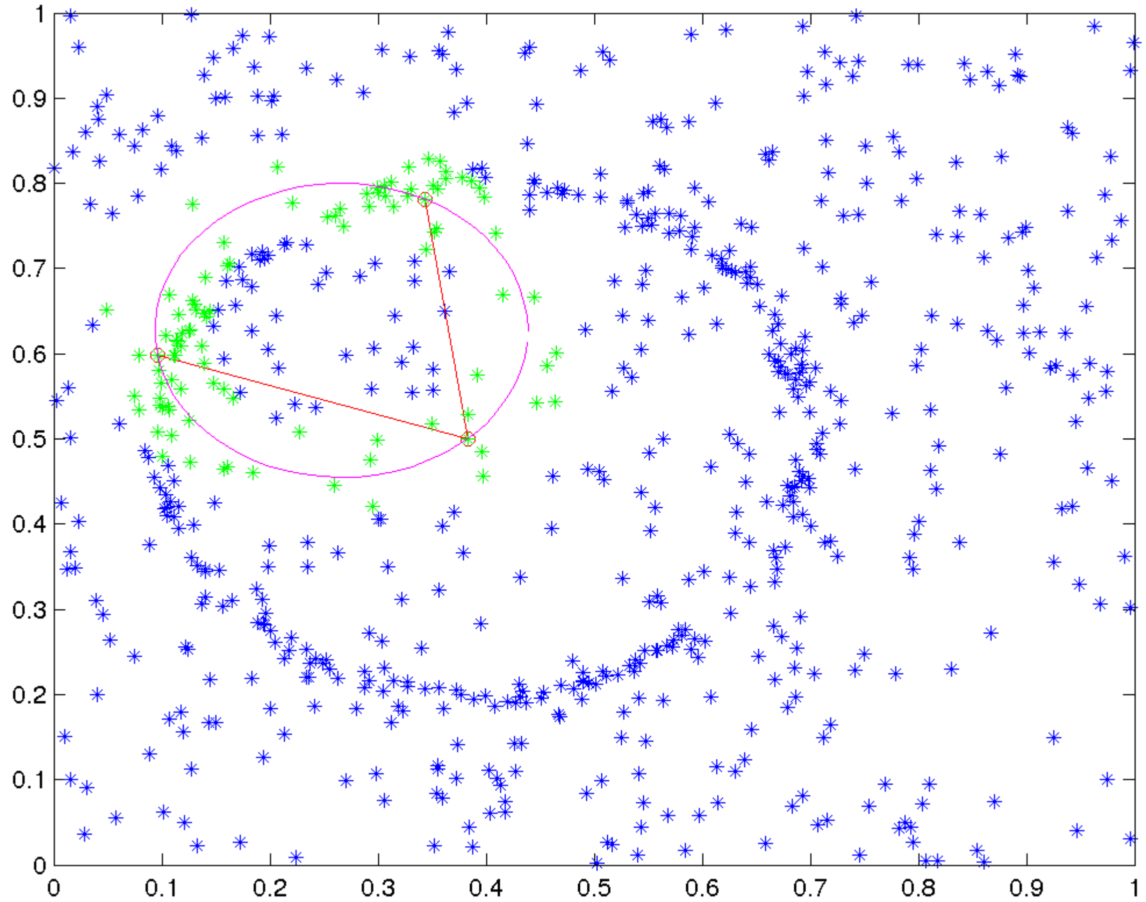
$$(x-a)^2+(y-b)^2 = r^2$$

- Fitness measure: distance from a point to the circle:

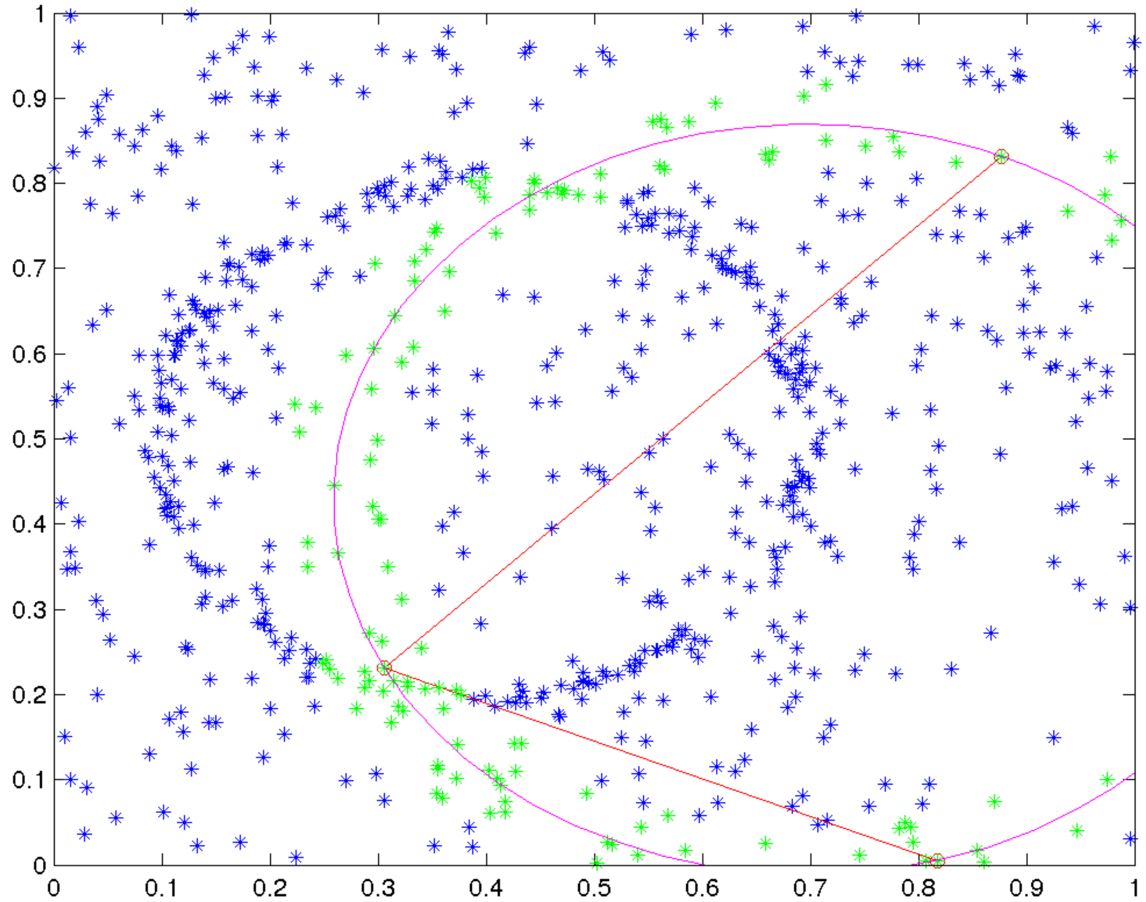
$$M(x,y\,|\,a,b)=\left|r-\sqrt{(x-a)^2+(y-b)^2}\right|$$

  – The sqrt is not strictly necessary, but gives M a metric meaning.

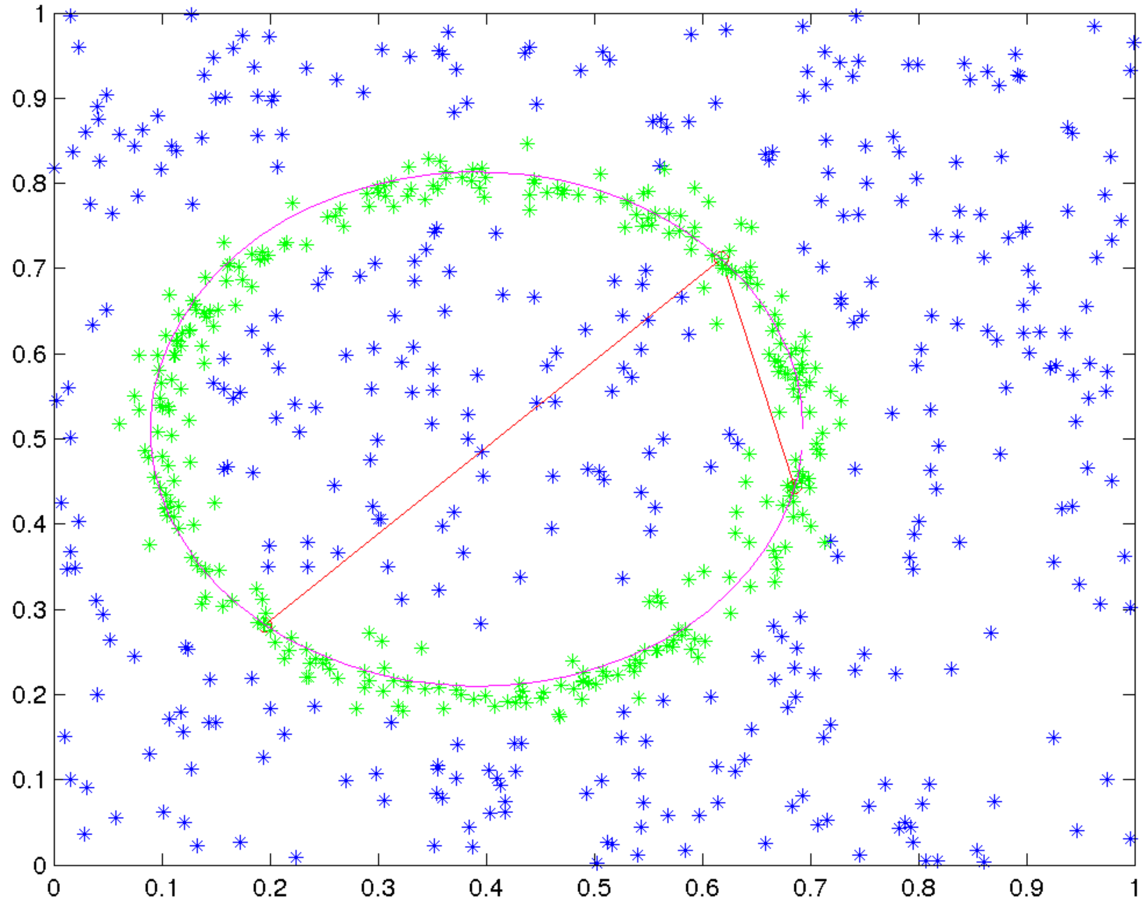
Example 3: Finding a generic circle



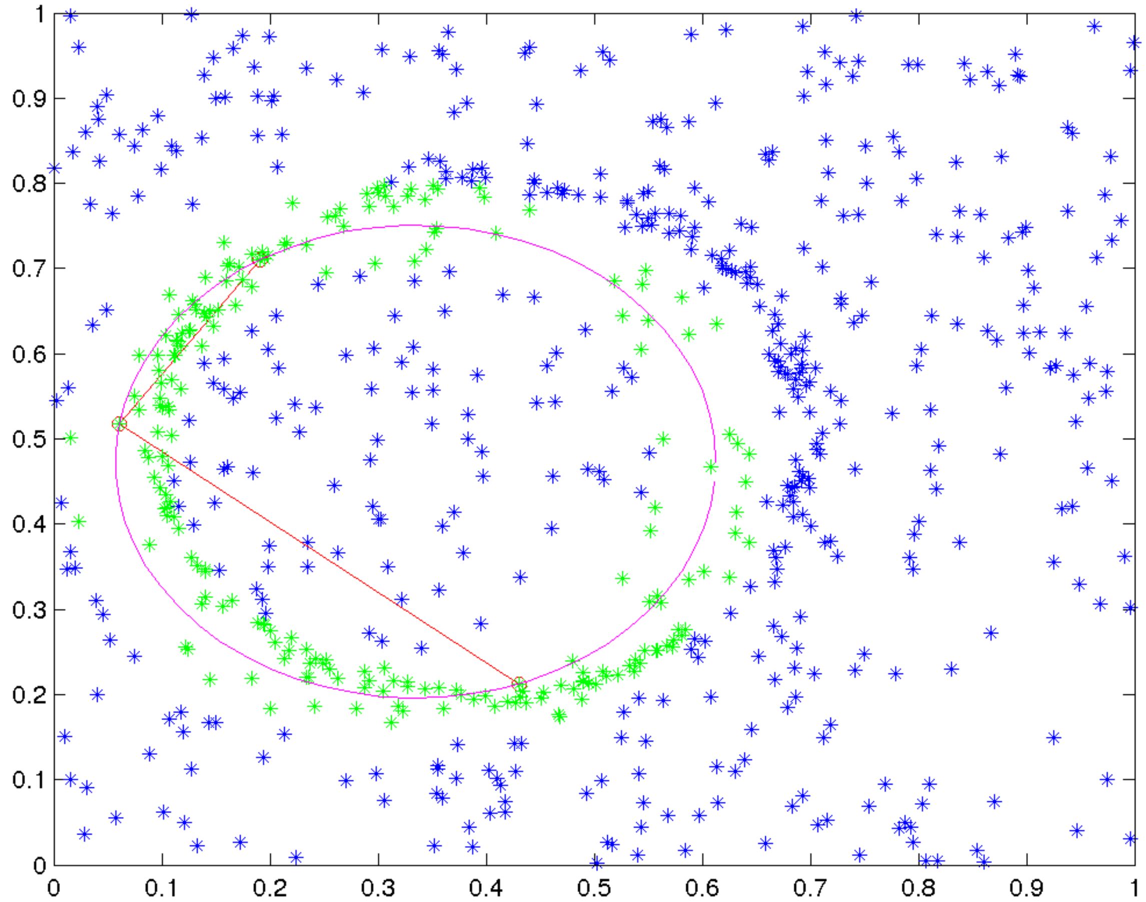
Example 3: Finding a generic circle



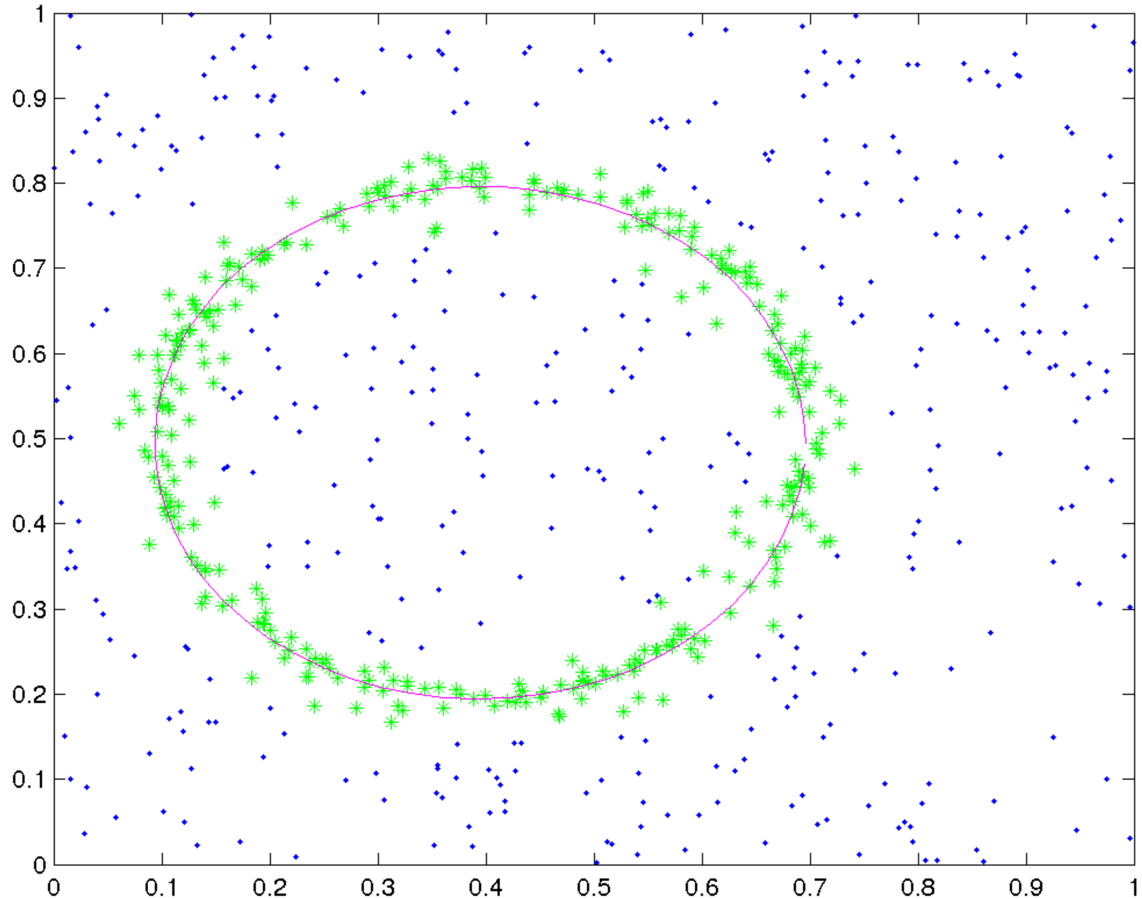
Example 3: Finding a generic circle



Example 3: Finding a generic circle



Example 3: Final



Max at
a = 0.4026 (0.40)
b = 0.5022 (0.50)
r = 0.3010 (0.30)

Score: 23.21
Inliers: 318

RANSAC Summary

- Also a very robust estimator
  - But only probabilistic guarantees to find the optimum
- No constraints on the parameter space
- Less sensitive to the dimensionality if the inliers are dominants
- Choosing between HT and RANSAC:
  - RANSAC is most of the time a better choice.
  - HT is more exhaustive and better encodes multiple candidates.

CONCLUSION

Conclusion

- Features: powerful way to extract most important information from raw data
- Reduction of dimension, hopefully without loss of information
- Investment of computation power with the hope that later tasks (matching, localisation, mapping, …) will become easier.
- Tons of features in computer vision.