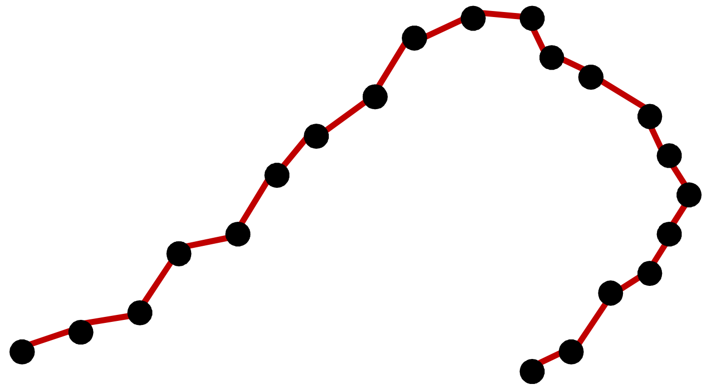
Slide 21

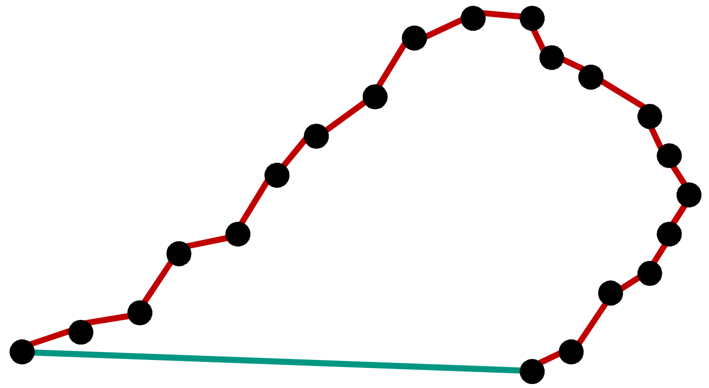
Ramer-Douglas-Peucker Algorithm

– basic idea: subdivide polyline recursively at the farthest vertex



Ramer-Douglas-Peucker Algorithm

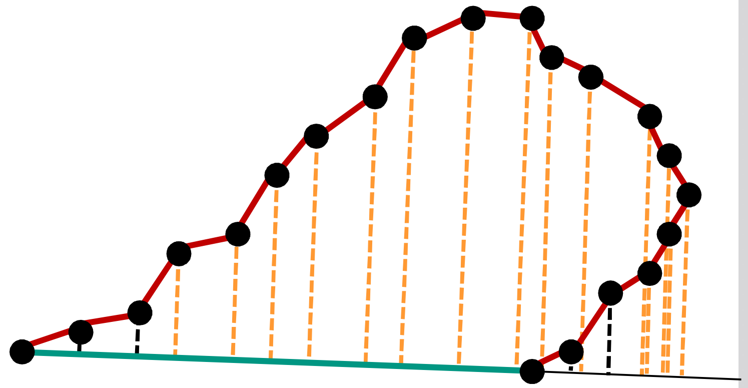
- basic idea: subdivide polyline recursively at the farthest vertex
  1. generate line from first to last pixel



Ramer-Douglas-Peucker Algorithm

– basic idea: subdivide polyline recursively at the farthest vertex

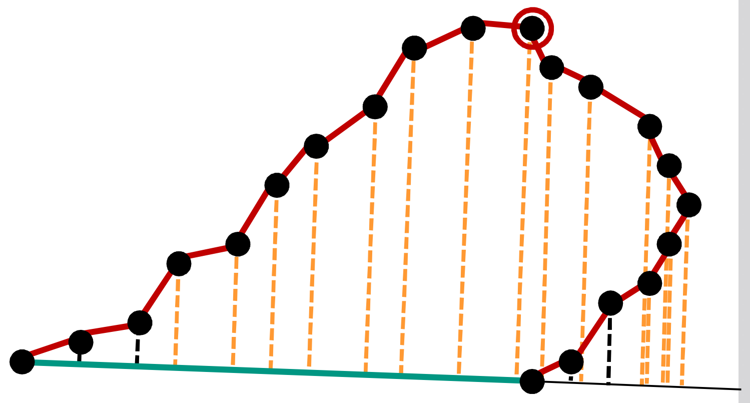
1. generate line from first to last pixel
2. calculate distance of pixels from the line



Ramer-Douglas-Peucker Algorithm

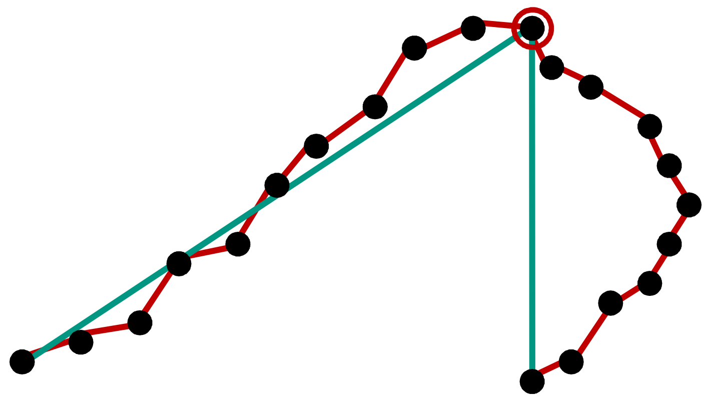
– basic idea: subdivide polyline recursively at the farthest vertex

1. generate line from first to last pixel
2. calculate distance of pixels from the line
3. if maximal distance is greater than tolerance, break edge list at farthest vertex and apply the algorithm to the two sublists



Ramer-Douglas-Peucker Algorithm

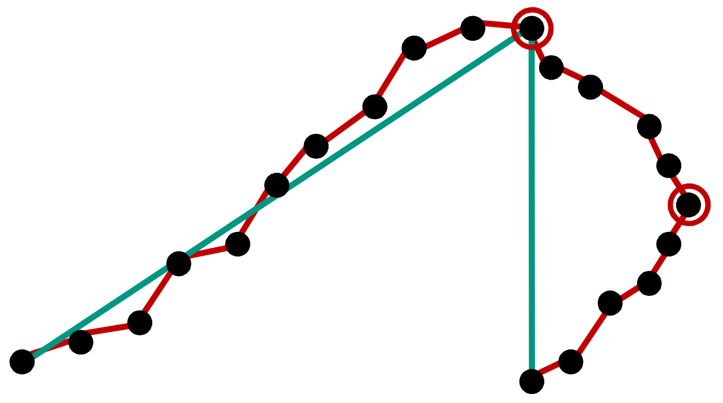
- basic idea: subdivide polyline recursively at the farthest vertex
  1. generate line from first to last pixel
  2. calculate distance of pixels from the line
  3. if maximal distance is greater than tolerance, break edge list at farthest vertex and apply the algorithm to the two sublists



Ramer-Douglas-Peucker Algorithm

– basic idea: subdivide polyline recursively at the farthest vertex

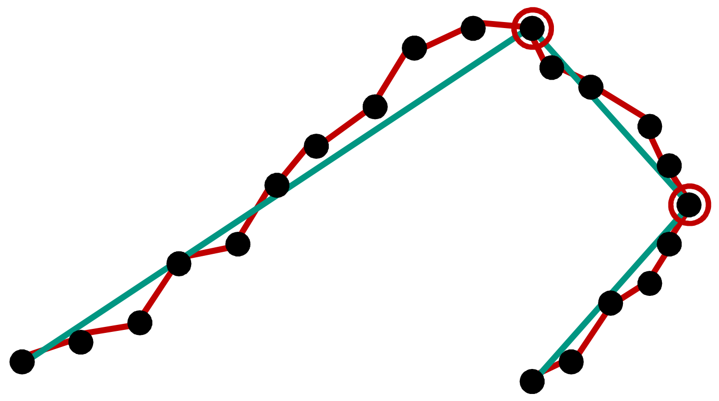
1. generate line from first to last pixel
2. calculate distance of pixels from the line
3. if maximal distance is greater than tolerance, break edge list at farthest vertex and apply the algorithm to the two sublists



Ramer-Douglas-Peucker Algorithm

– basic idea: subdivide polyline recursively at the farthest vertex

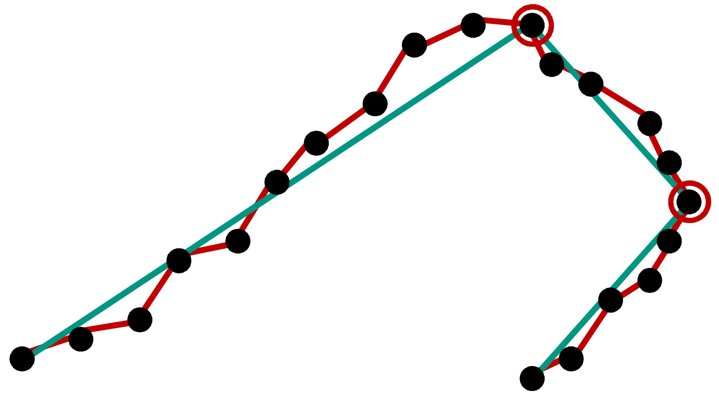
1. generate line from first to last pixel
2. calculate distance of pixels from the line
3. if maximal distance is greater than tolerance, break edge list at farthest vertex and apply the algorithm to the two sublists

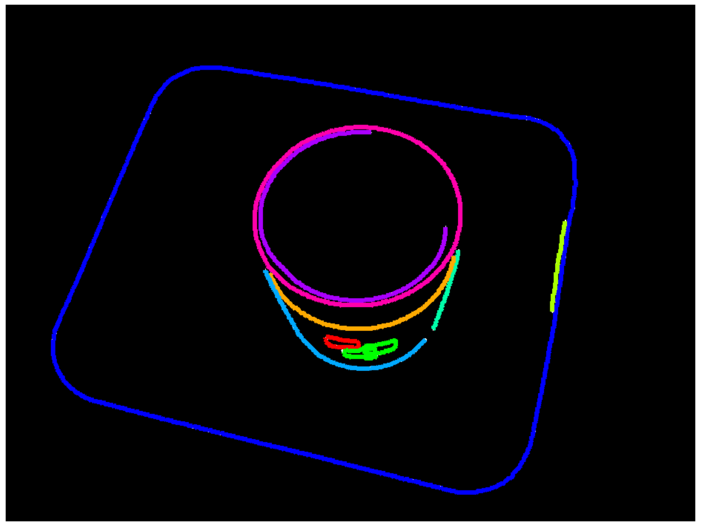


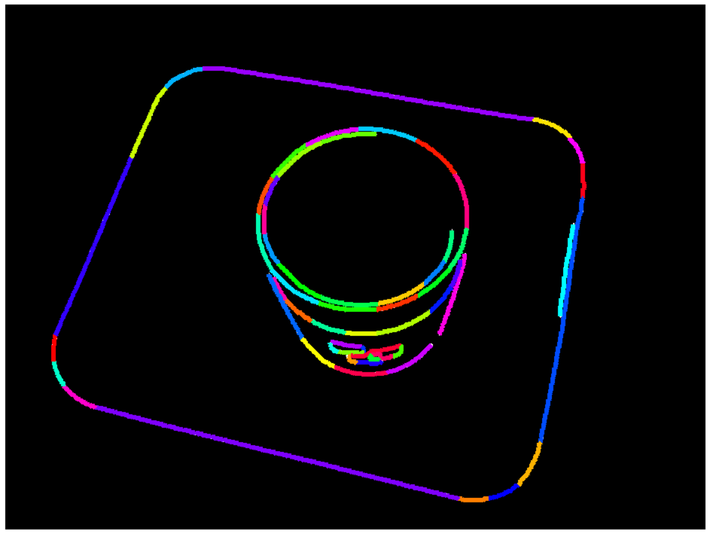
Ramer-Douglas-Peucker Algorithm

– basic idea: subdivide polyline recursively at the farthest vertex

1. generate line from first to last pixel
2. calculate distance of pixels from the line
3. if maximal distance is greater than tolerance, break edge list at farthest vertex and apply the algorithm to the two sublists



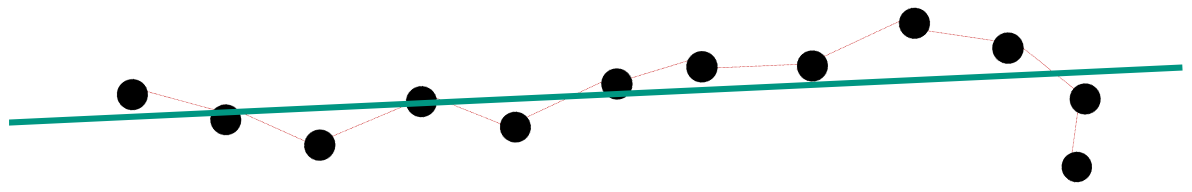




Slide 33

Line Estimation cont.

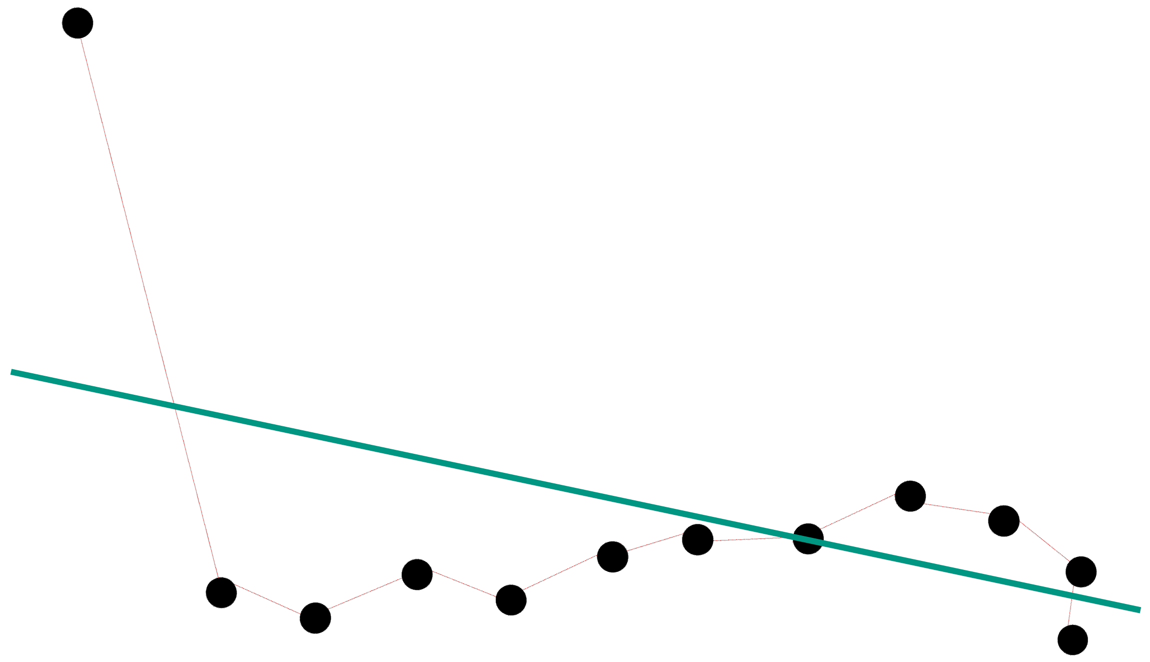
- robustness concerning outliers:



  - least squares estimation is easily distorted by outliers
  - outliers occur often in machine vision

Line Estimation cont.

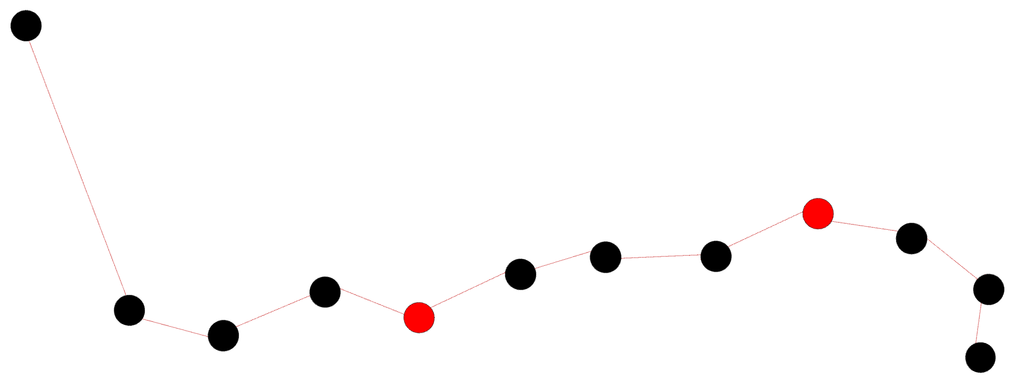
- robustness concerning outliers:

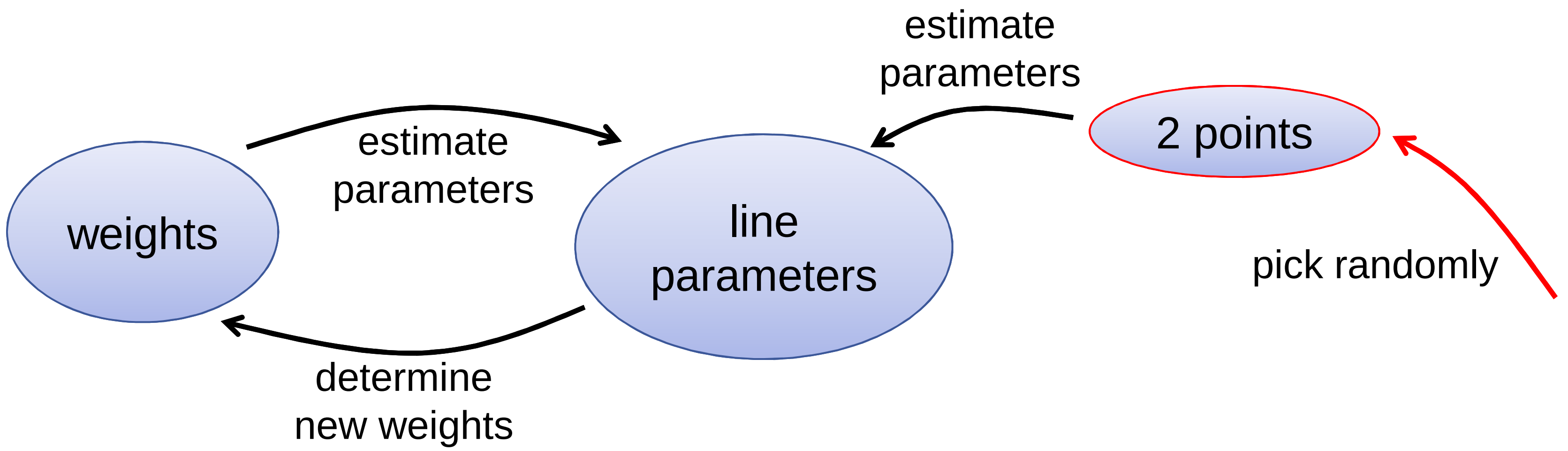


  - least squares estimation is easily distorted by outliers
  - outliers occur often in machine vision

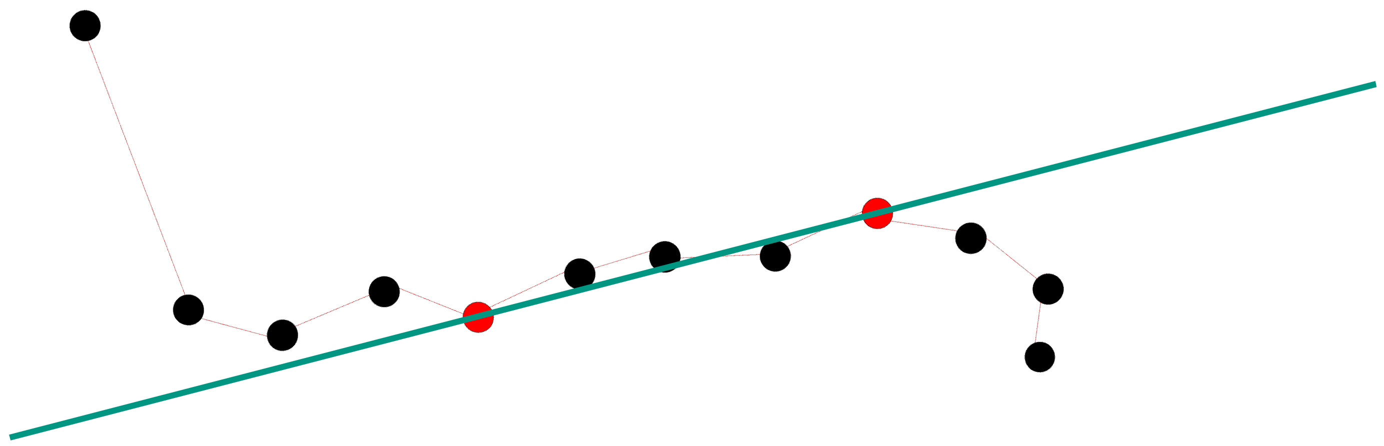
Slide 43

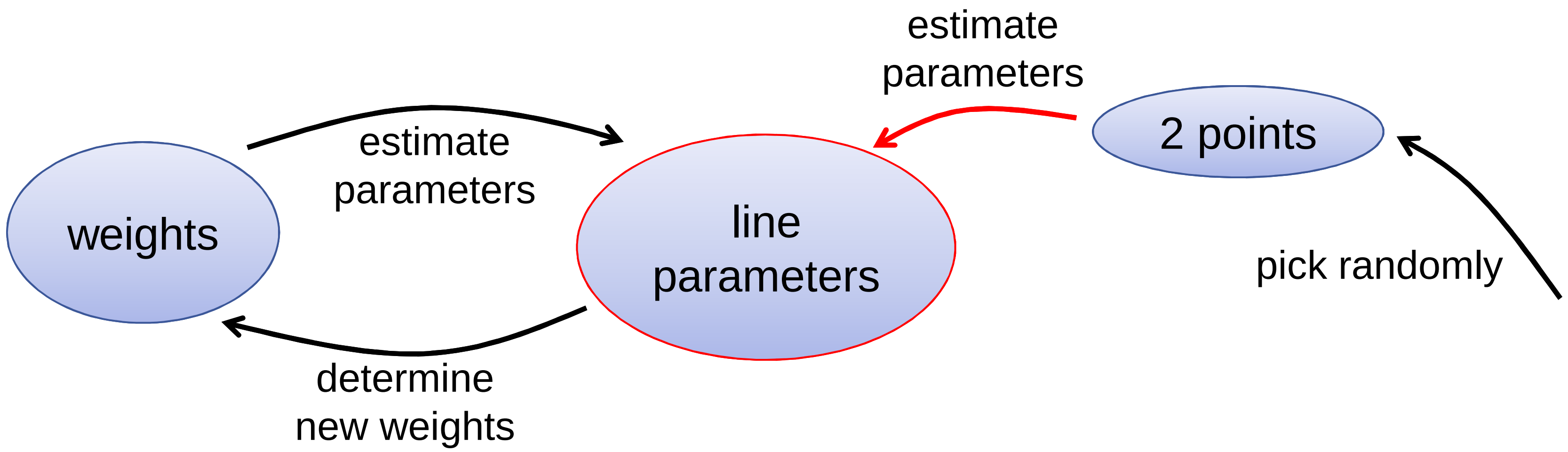
M-Estimators cont.



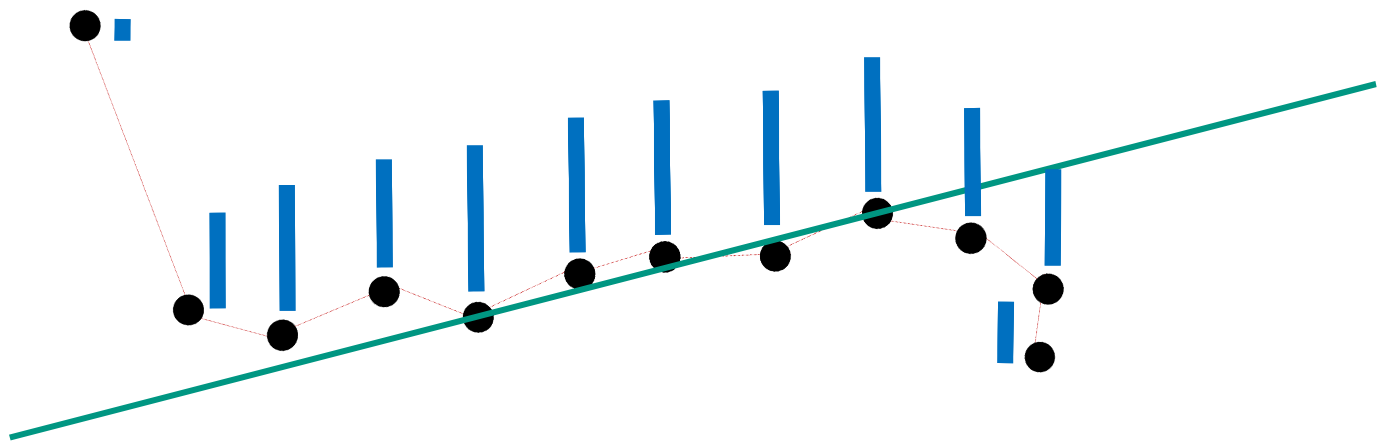


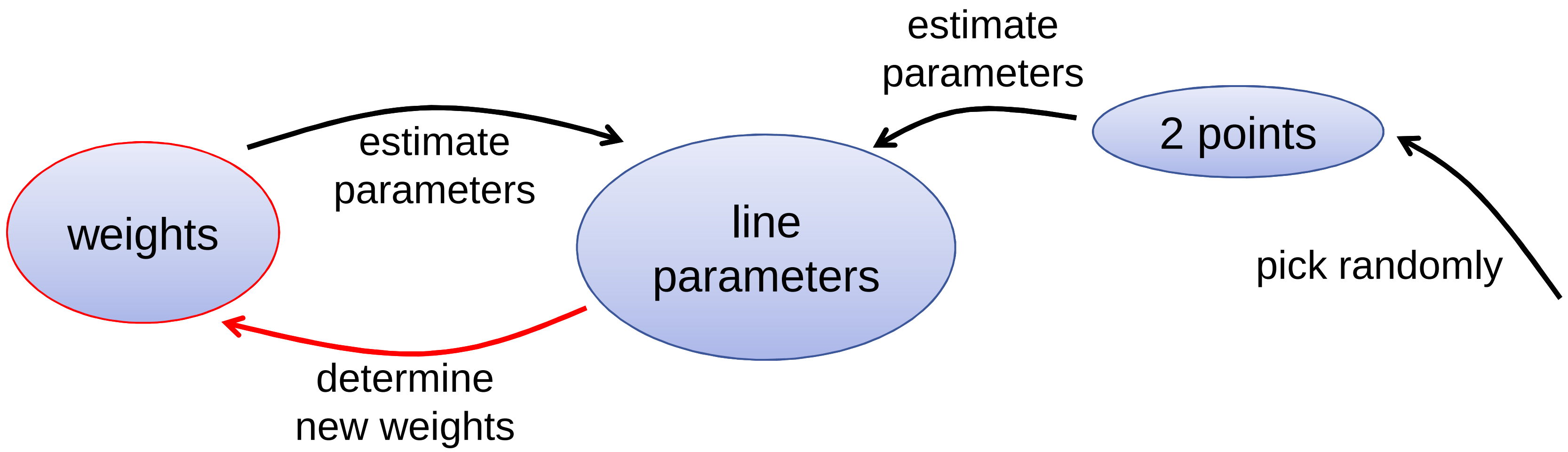
M-Estimators cont.



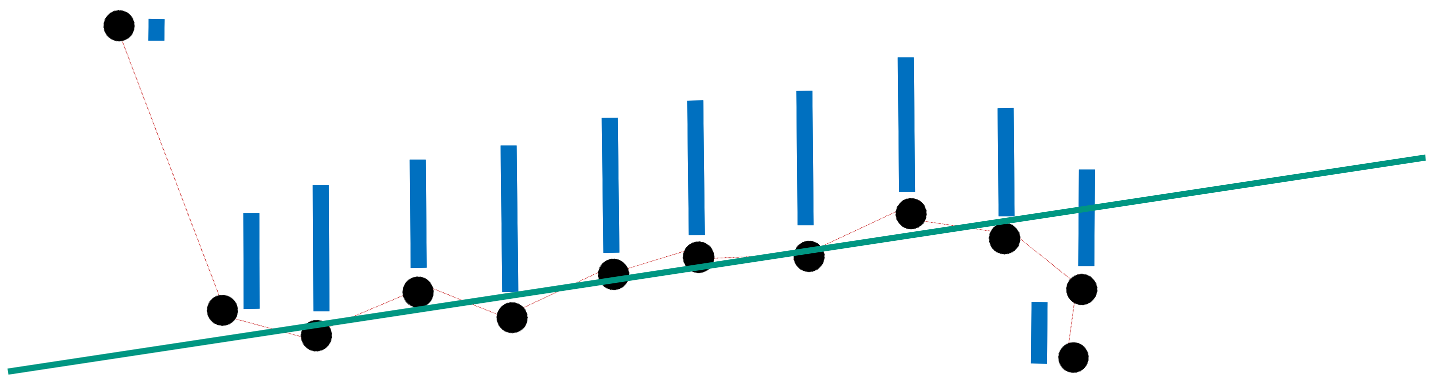


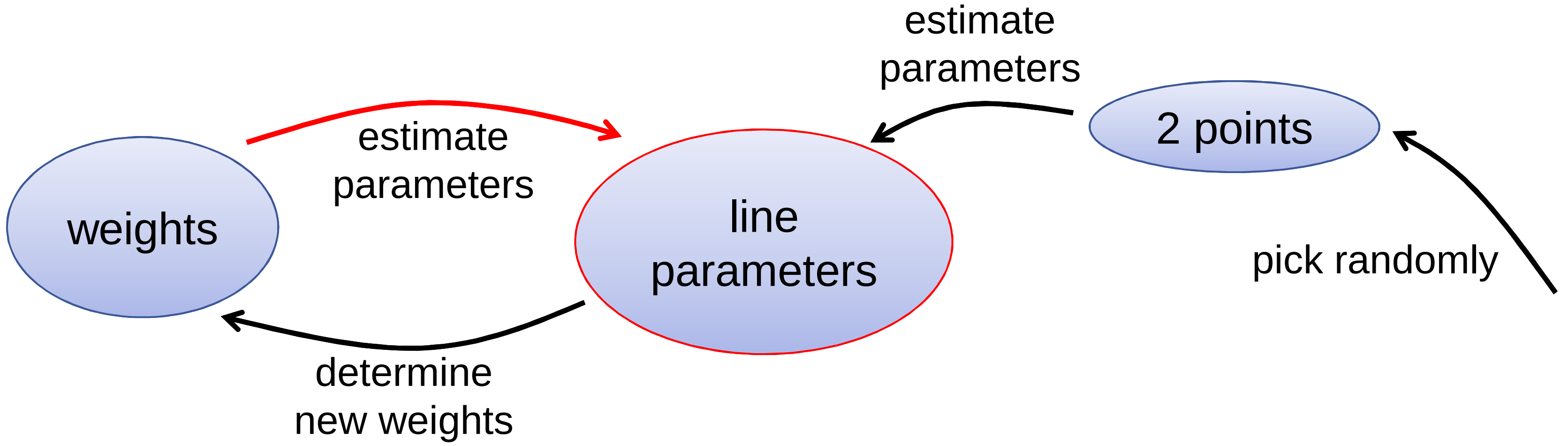
M-Estimators cont.



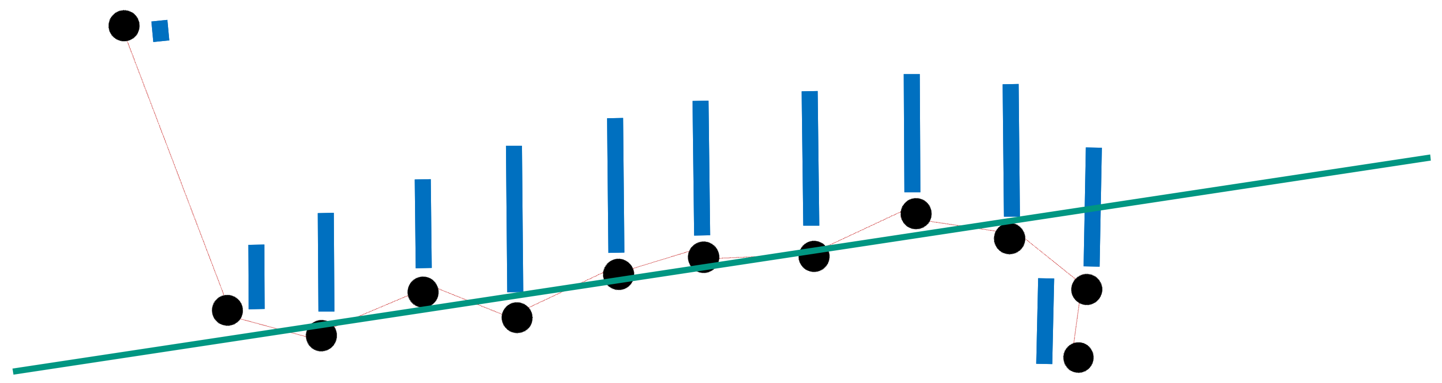


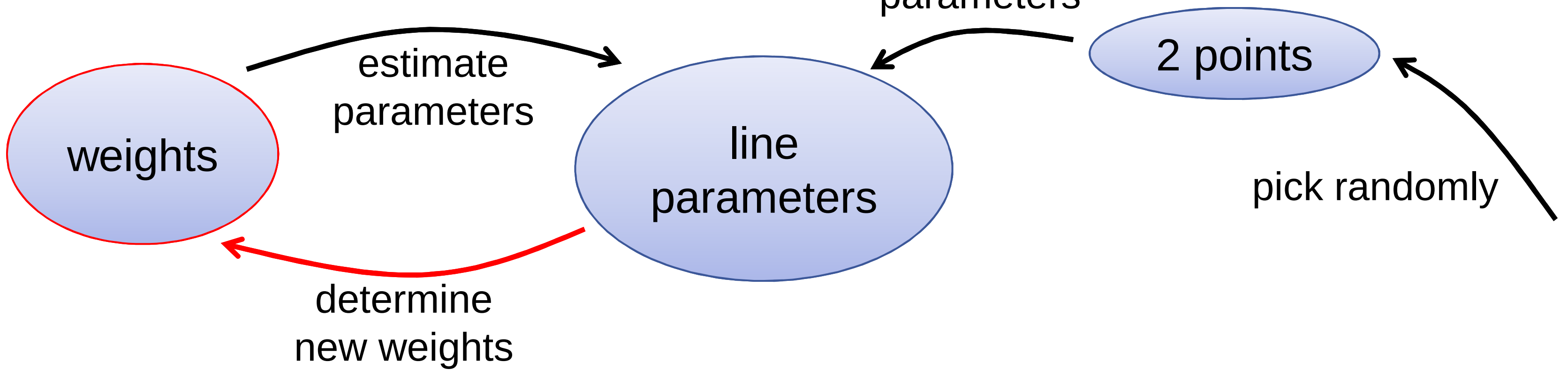
M-Estimators cont.



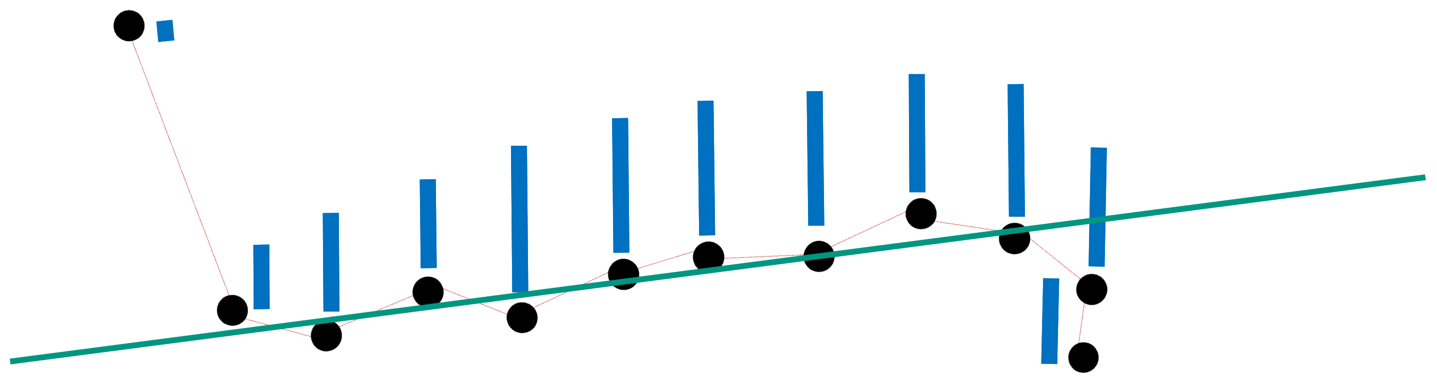


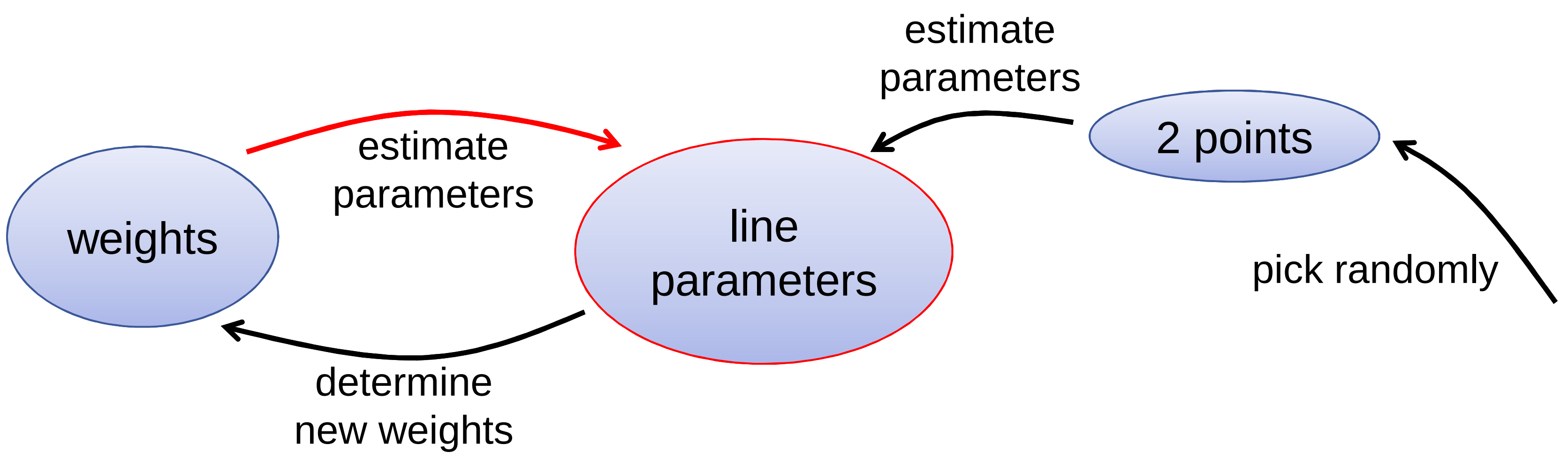
M-Estimators cont.





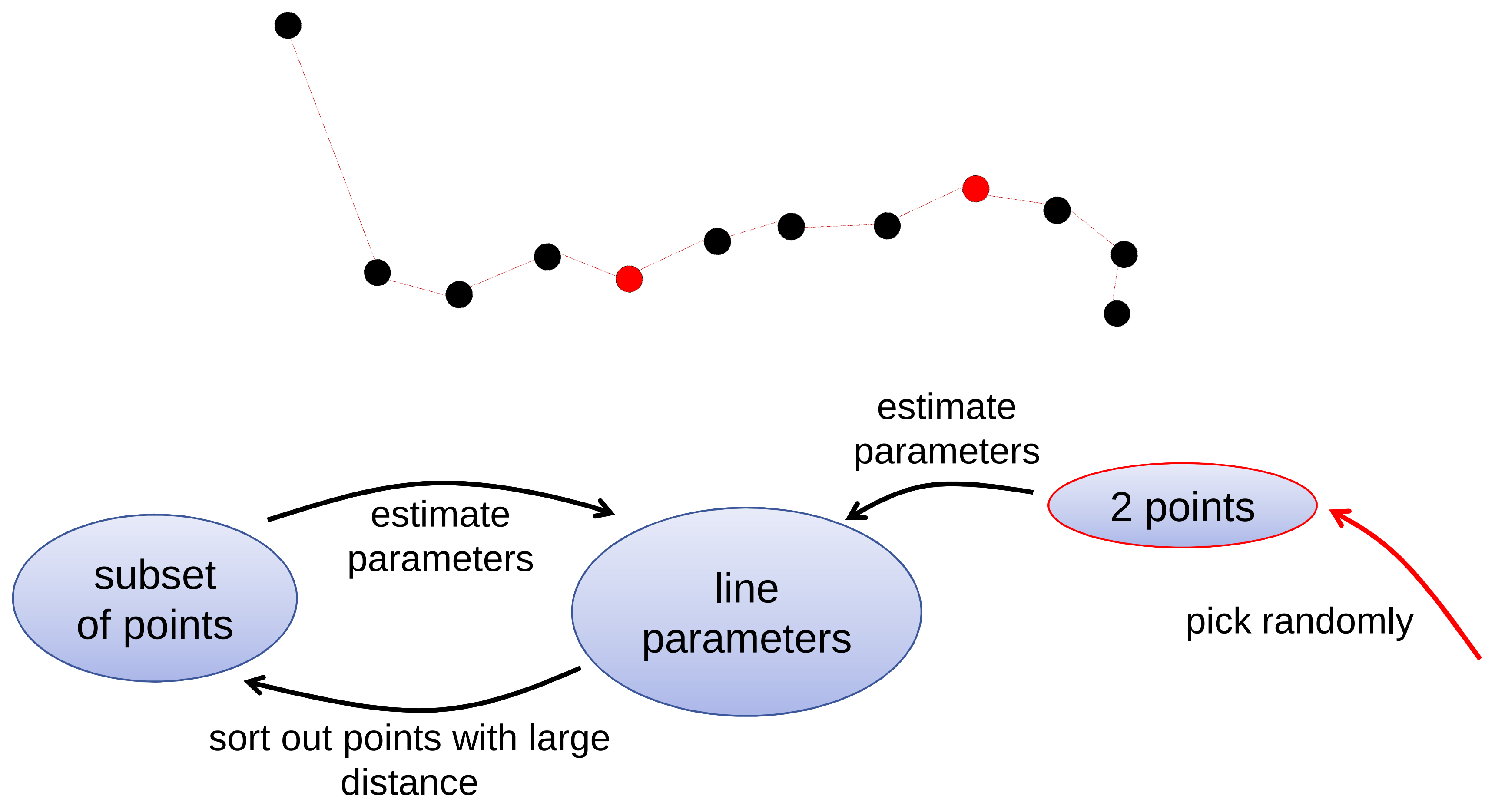
M-Estimators cont.



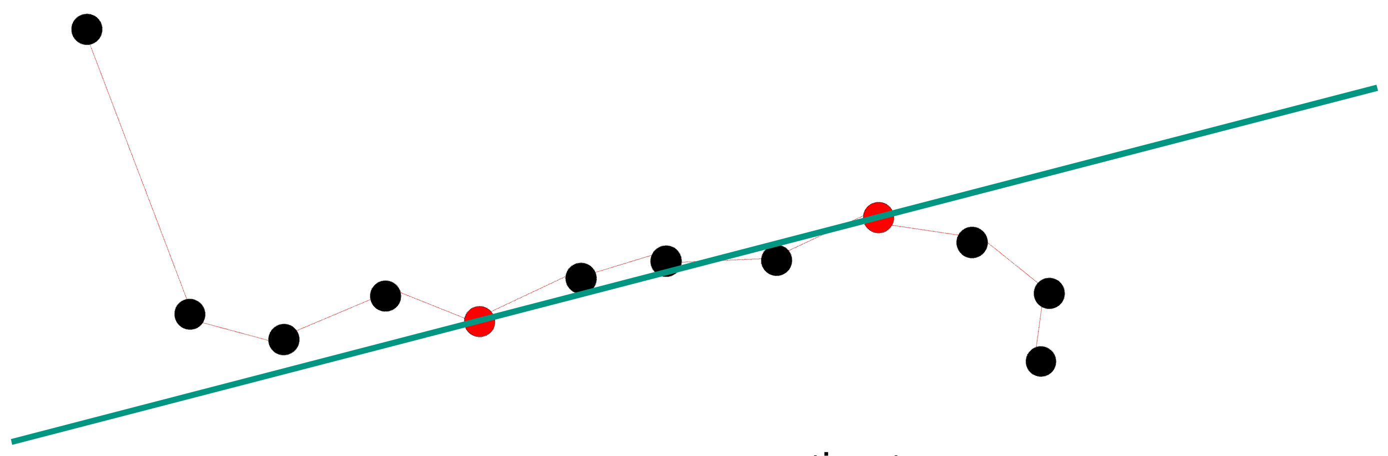


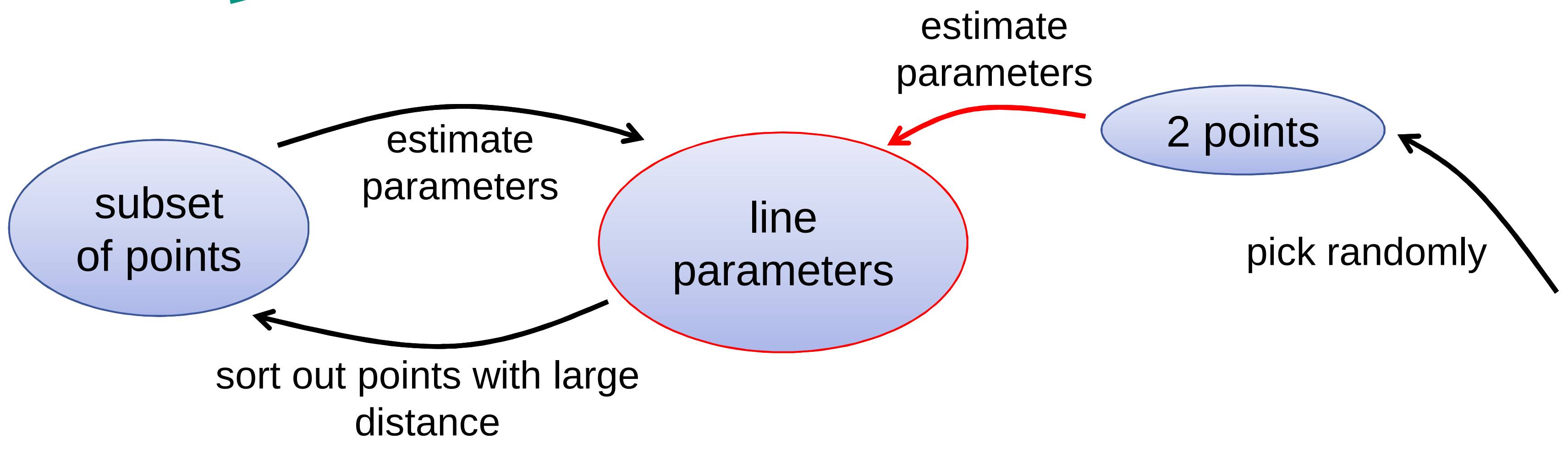
Slide 46

LTS cont.

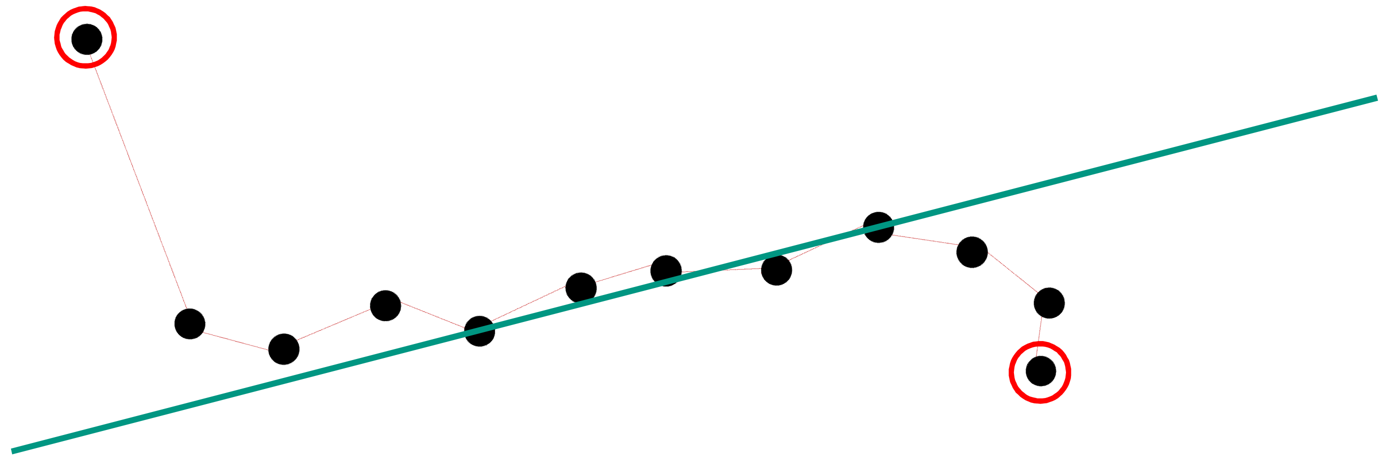

estimate
parameters
2 points
estimate
parameters
subset
of points
line
parameters
pick randomly
sort out points with large
distance

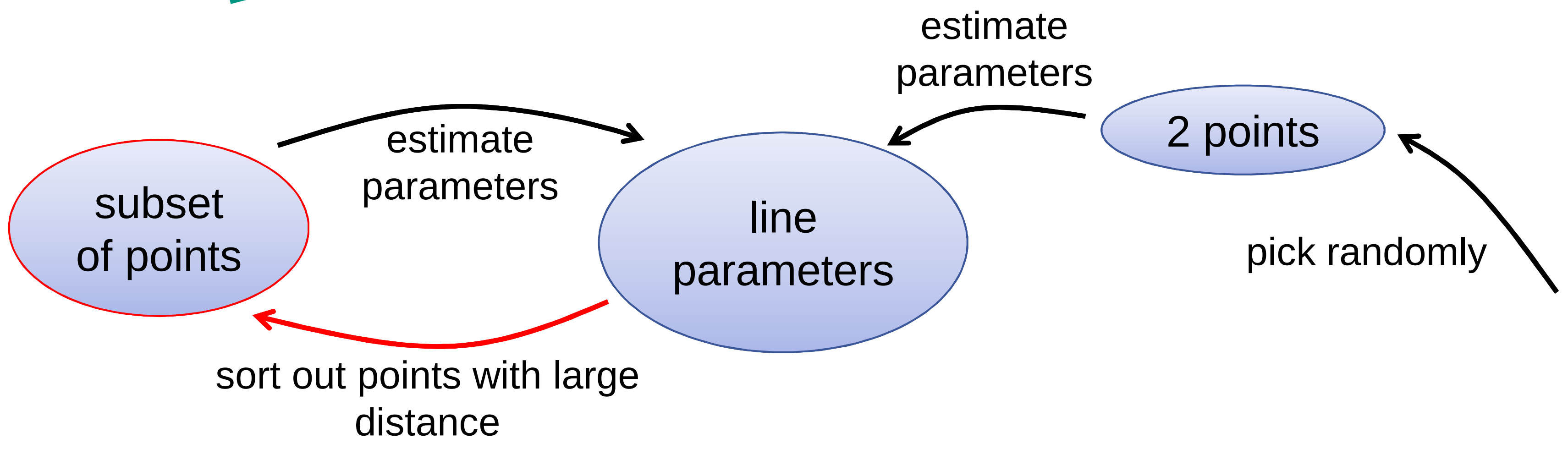
LTS cont.



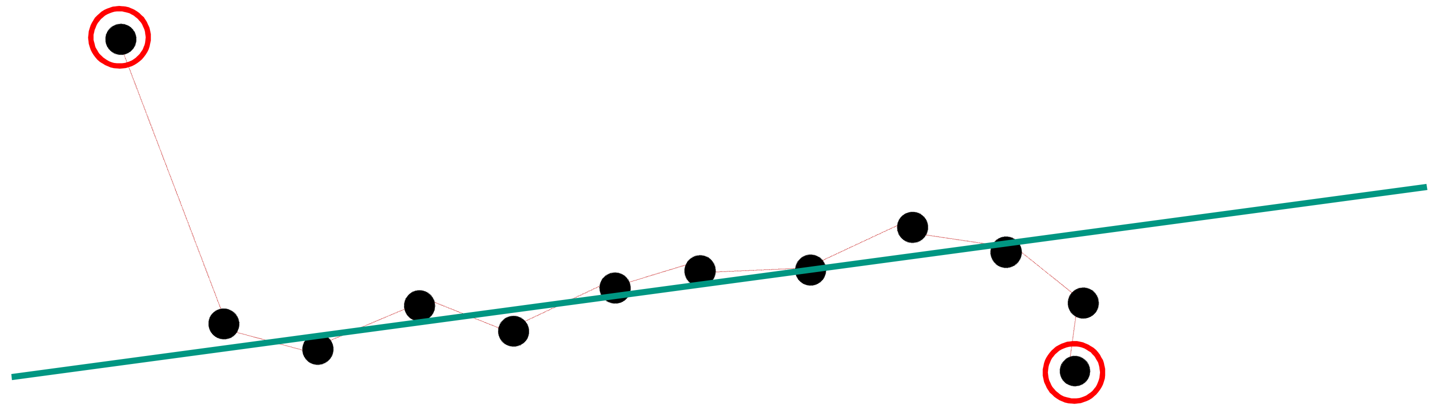

subset of points
estimate parameters
line parameters
sort out points with large distance
estimate parameters
2 points
pick randomly

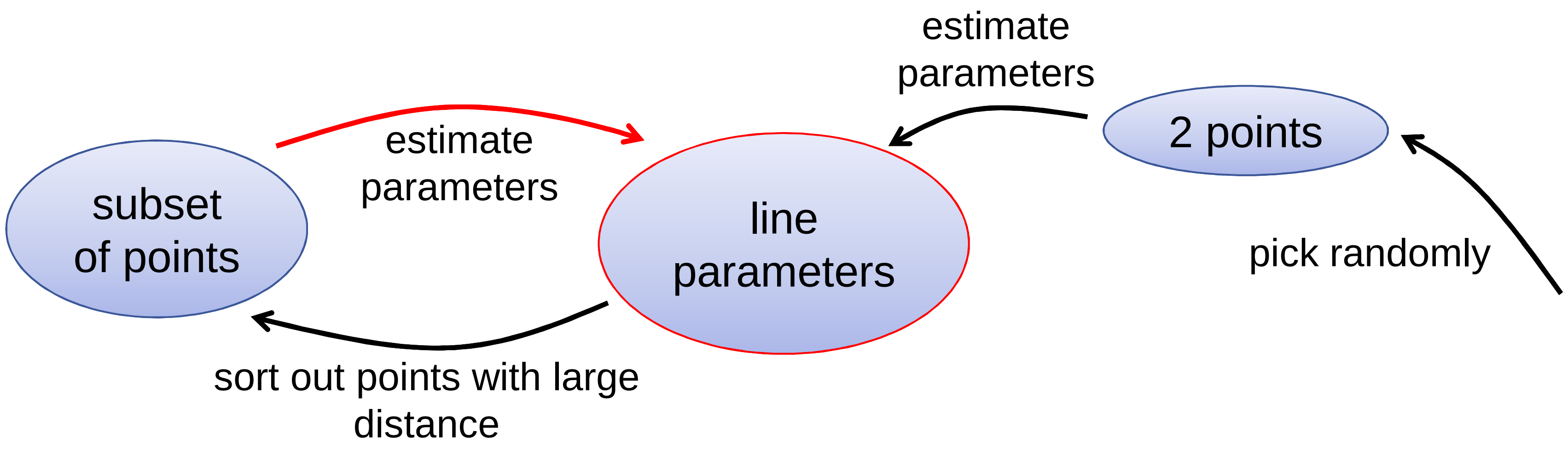
LTS cont.



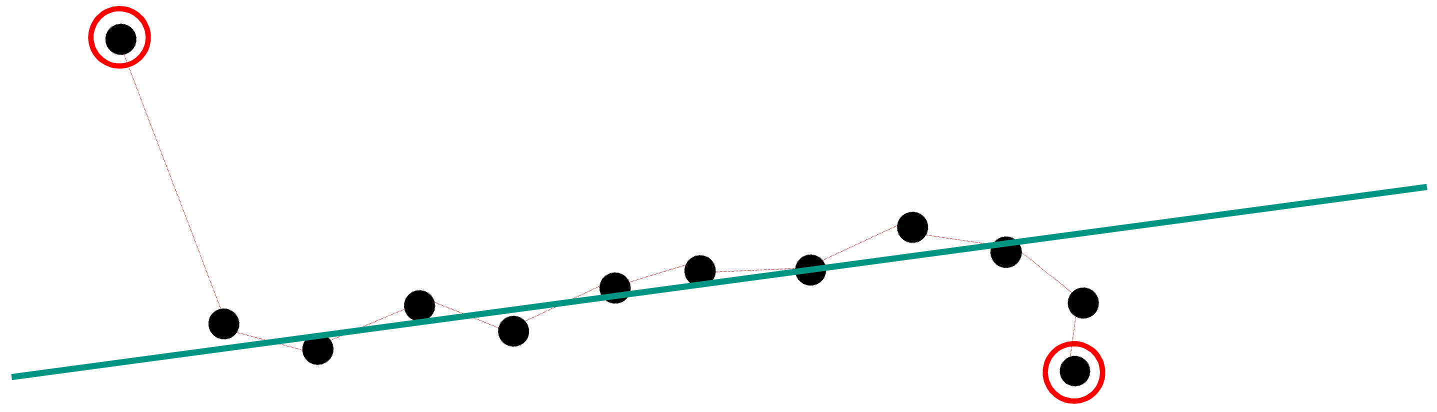

estimate
parameters
2 points
estimate
parameters
subset
of points
line
parameters
pick randomly
sort out points with large
distance

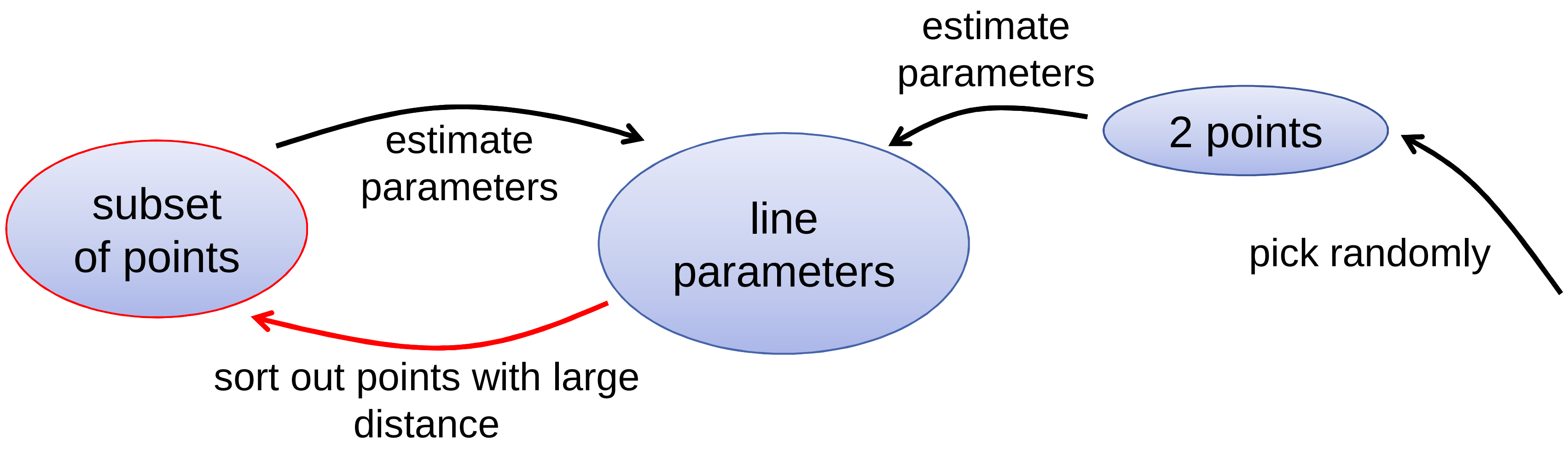
LTS cont.




estimate
parameters
subset
of points
estimate
parameters
line
parameters
2 points
pick randomly
sort out points with large
distance

LTS cont.





- to overcome convergence into a local minimum, the whole process is repeated several times

Slide 47

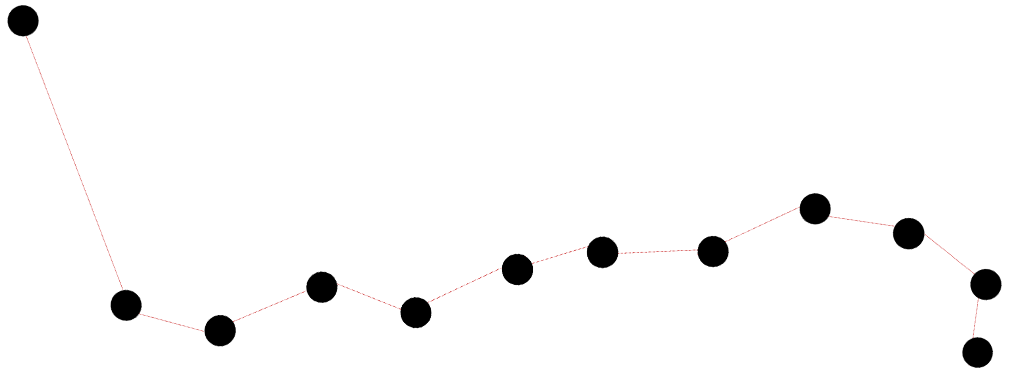
RANSAC

- idea: search a line that passes nearby as many points as possible

$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{N} \sigma(d_i)$$

$$\text{with } \sigma(d_i) = \begin{cases} 0 & \text{if } |d_i| \leq \theta \\ 1 & \text{if } |d_i| > \theta \end{cases}$$

- definition similar to M-estimator, but σ is discontinuous



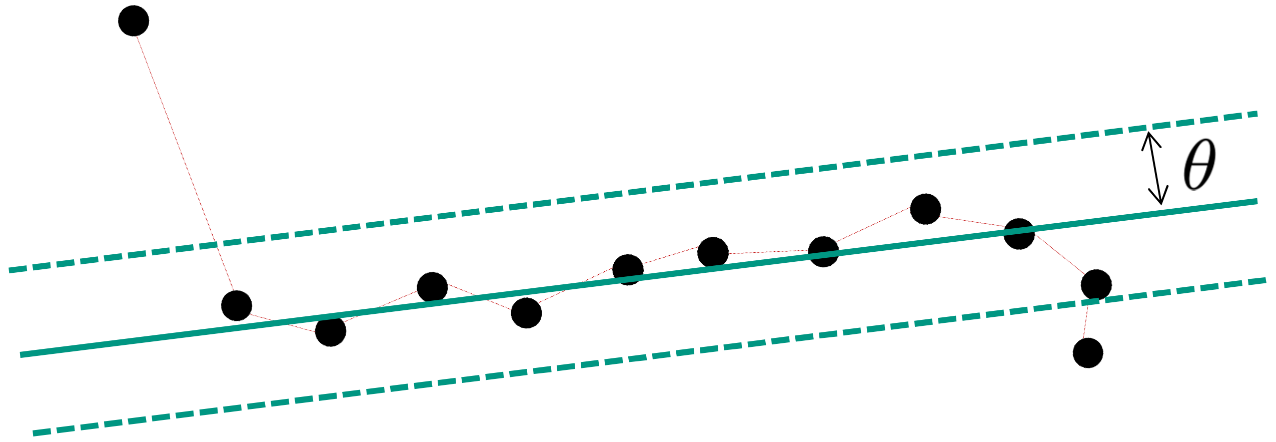
RANSAC

- idea: search a line that passes nearby as many points as possible

$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{N} \sigma(d_i)$$

$$\text{with } \sigma(d_i) = \begin{cases} 0 & \text{if } |d_i| \leq \theta \\ 1 & \text{if } |d_i| > \theta \end{cases}$$

- definition similar to M-estimator, but σ is discontinuous



error term=2
→ good fit

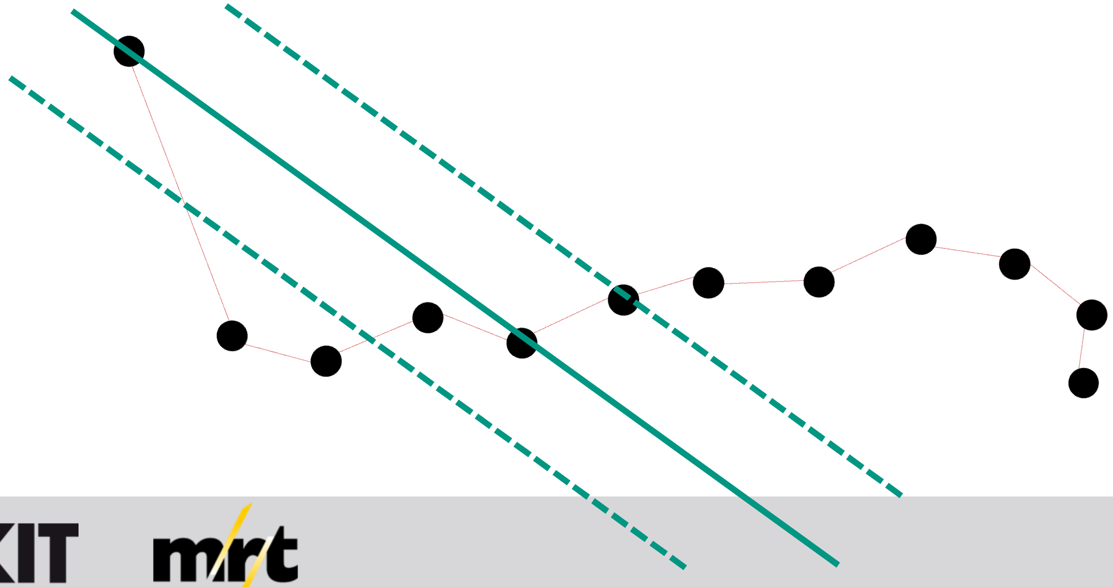
RANSAC

- idea: search a line that passes nearby as many points as possible

$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{N} \sigma(d_i)$$

$$\text{with } \sigma(d_i) = \begin{cases} 0 & \text{if } |d_i| \leq \theta \\ 1 & \text{if } |d_i| > \theta \end{cases}$$

- definition similar to M-estimator, but σ is discontinuous

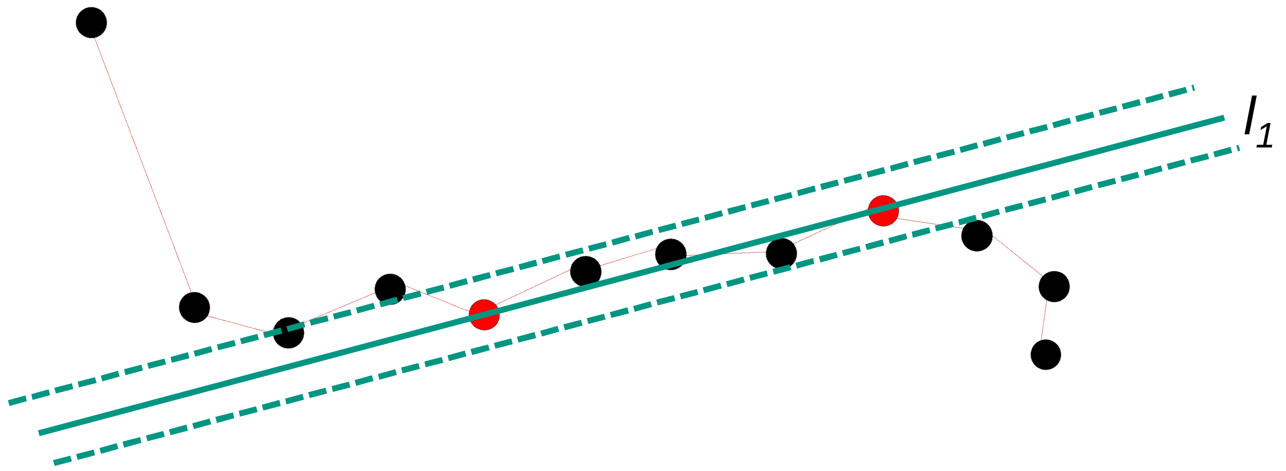


error term=8
→ bad fit

Slide 50

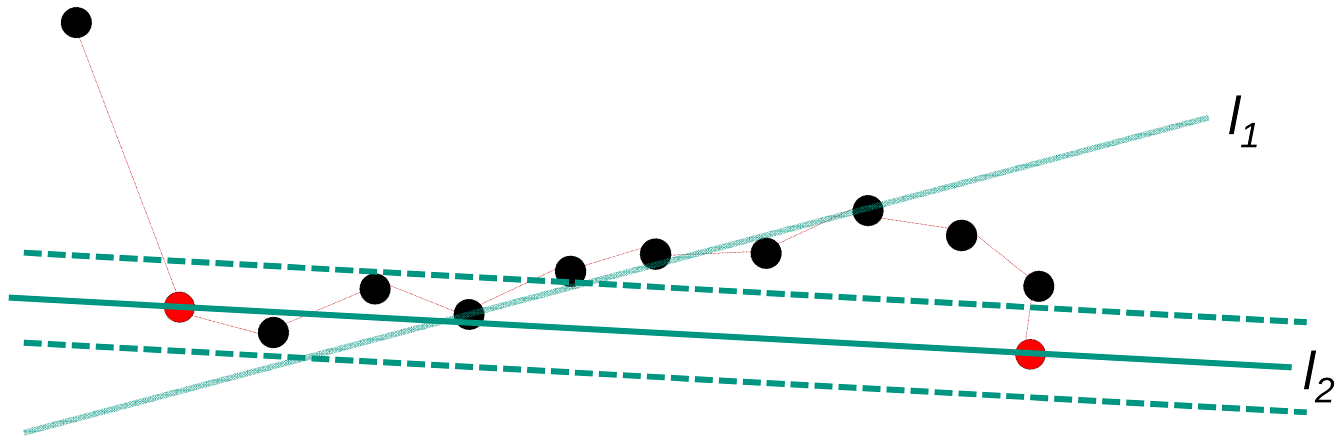
RANSAC cont.

- 1st trial: 6 outliers



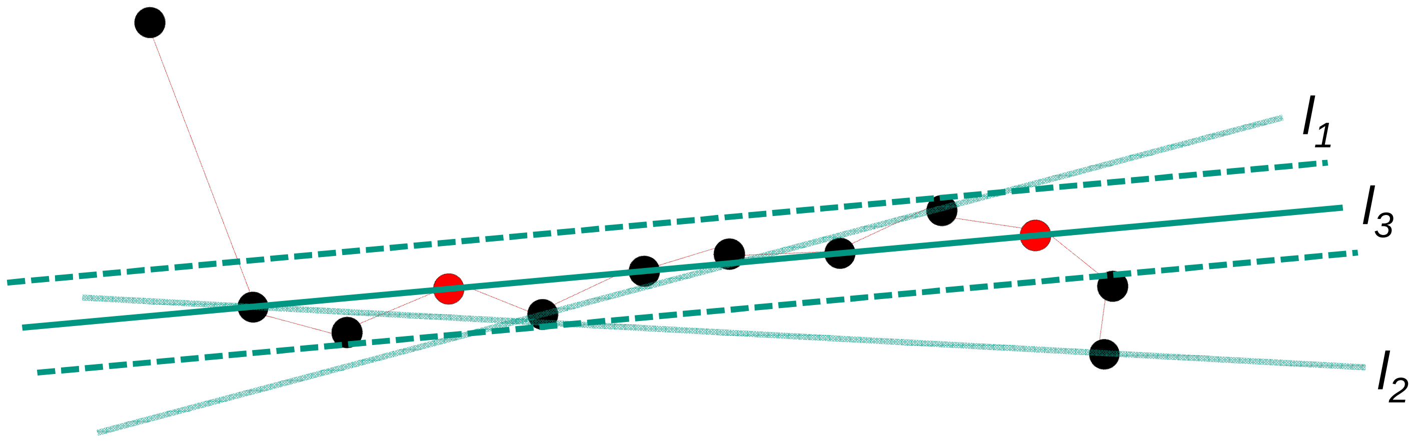
RANSAC cont.

- 1st trial: 6 outliers
- 2nd trial: 7 outliers



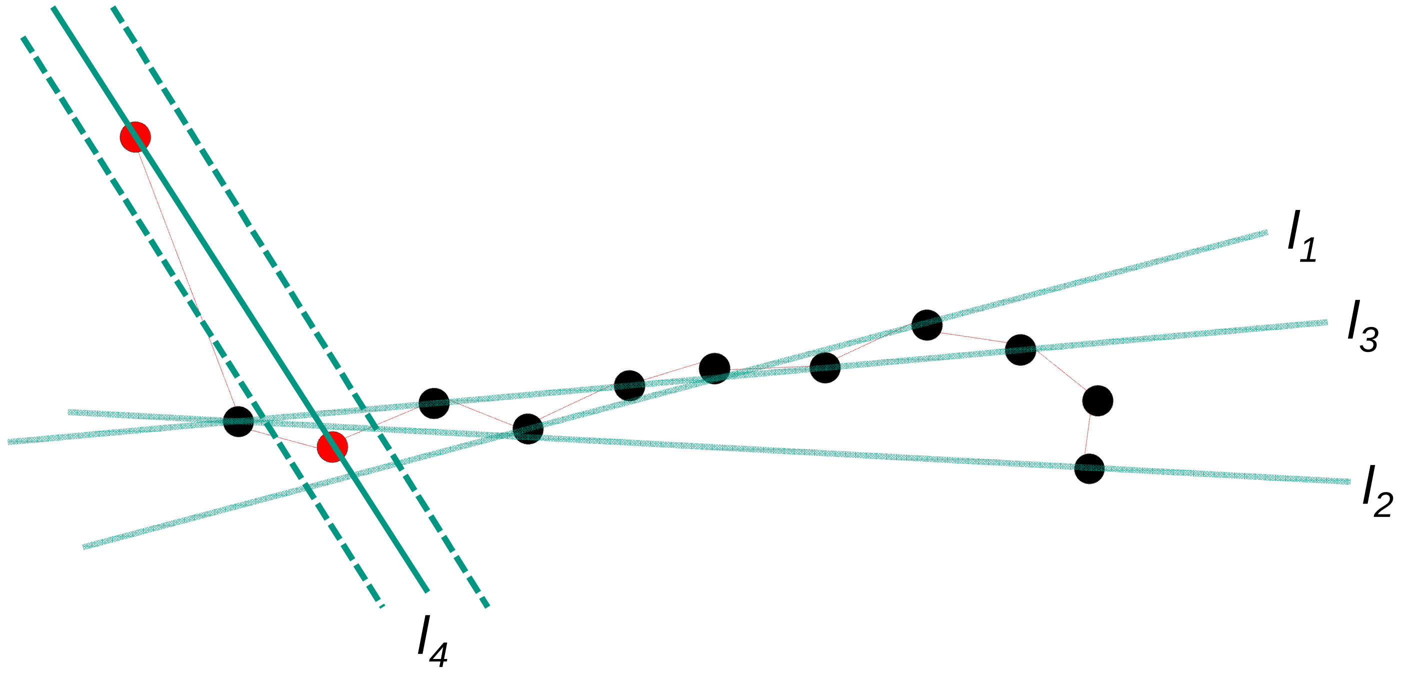
RANSAC cont.

- 1st trial: 6 outliers
- 2nd trial: 7 outliers
- 3rd trial: 3 outliers



RANSAC cont.

- 1st trial: 6 outliers
- 2nd trial: 7 outliers
- 3rd trial: 3 outliers
- 4th trial: 10 outliers



RANSAC cont.

- 1st trial: 6 outliers
- 2nd trial: 7 outliers
- 3rd trial: 3 outliers
- 4th trial: 10 outliers