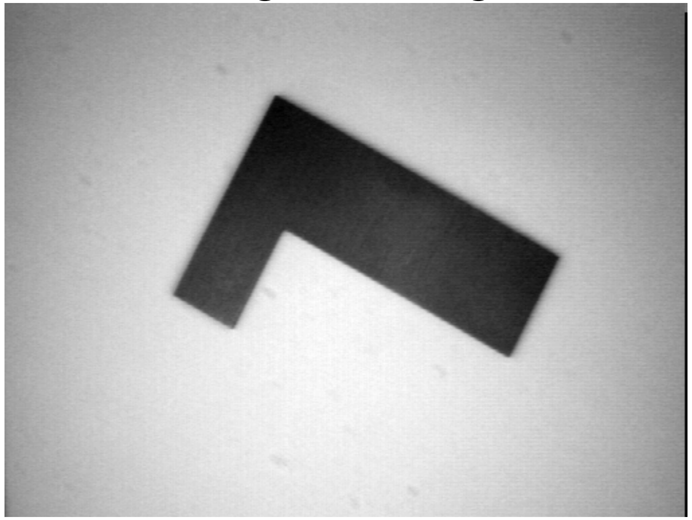
Machine Vision

Chapter 4: Curve Fitting

Dr. Martin Lauer
Institut für Mess- und Regelungstechnik

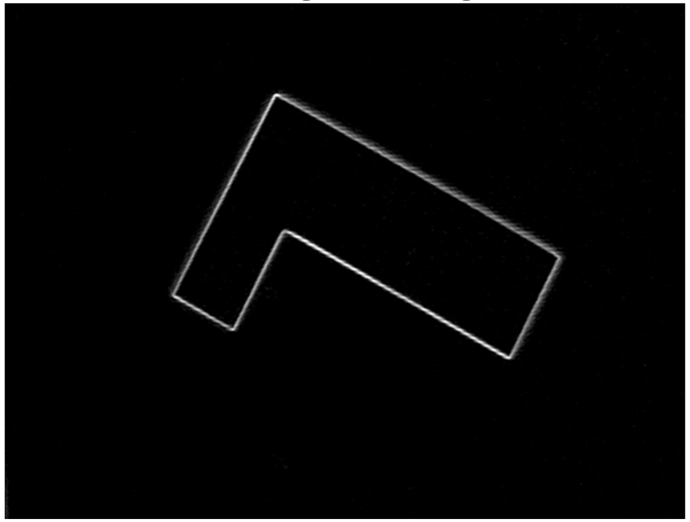
Contours

original image



edge detector

edge image



contour detector

polygon:
(195, 61) – (118,210) –
(163,237) – (201,162) –
(369,258) – (406,182)

geometric description

REPETITION: 2D GEOMETRY

2D Geometry

- dot product:
  - definition:

    $$\langle \vec{p}, \vec{q} \rangle = p_1 q_1 + p_2 q_2$$

  - bilinearity:

    $$\langle \alpha\vec{p}+\beta\vec{r}, \gamma\vec{q}+\delta\vec{s} \rangle = \alpha\gamma\langle \vec{p}, \vec{q} \rangle+\alpha\delta\langle \vec{p}, \vec{s} \rangle+\beta\gamma\langle \vec{r}, \vec{q} \rangle+\beta\delta\langle \vec{r}, \vec{s} \rangle$$

  - important property:

    $$\langle \vec{p}, \vec{q} \rangle = ||\vec{p}|| \cdot ||\vec{q}|| \cdot \cos\angle(\vec{p}, \vec{q})$$

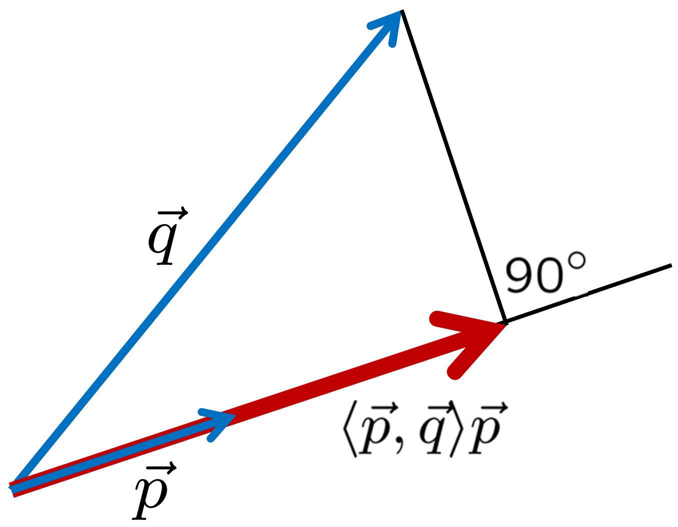
  - follows:

    $$\langle \vec{p}, \vec{p} \rangle = ||\vec{p}||^2$$

    $$\langle \vec{p}, \vec{q} \rangle = 0 \quad \text{if } \vec{p}\perp\vec{q}$$

  - projection on direction

    $$\langle \vec{p}, \vec{q} \rangle\vec{p} \quad \text{with } ||\vec{p}|| = 1$$



2D Geometry cont.

- lines and line segments
  - line segment with end points $\vec{p}$ and $\vec{q}$:

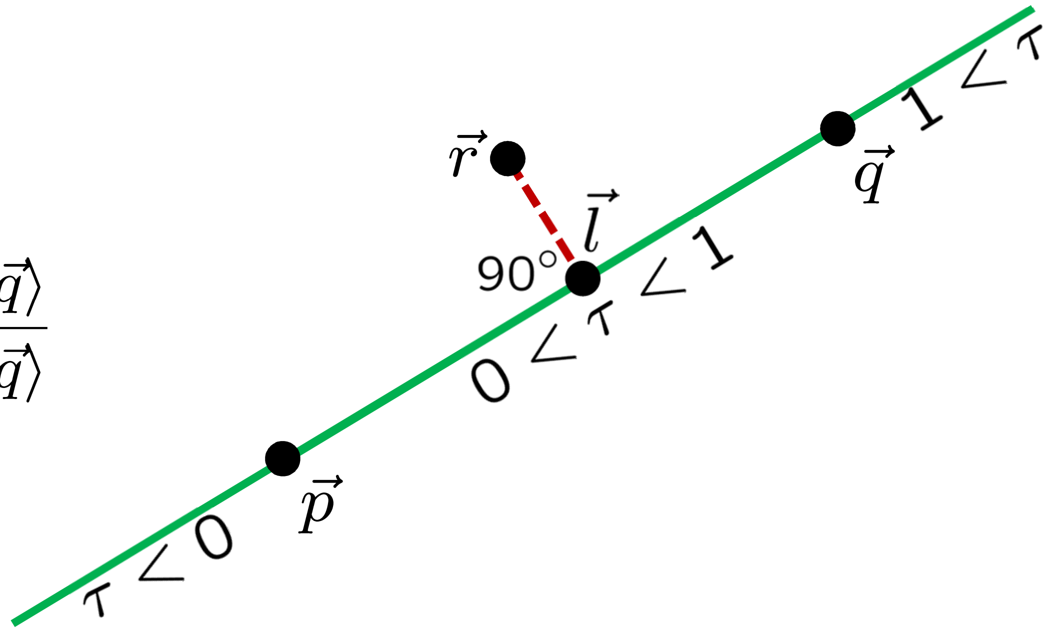
    $$\vec{x} = (1-\tau)\vec{p} + \tau\vec{q}, \quad \tau \in [0,1]$$
  - is part of line:

    $$\vec{x} = (1-\tau)\vec{p} + \tau\vec{q}, \quad \tau \in \mathbb{R}$$
  - perpendicular point:

    $$\vec{l} = (1-\tau)\vec{p} + \tau\vec{q}$$

    $$0 = \langle \vec{l} - \vec{r}, \vec{p} - \vec{q} \rangle$$

    $$\rightarrow \quad \tau = \frac{\langle \vec{p} - \vec{r}, \vec{p} - \vec{q} \rangle}{\langle \vec{p} - \vec{q}, \vec{p} - \vec{q} \rangle}$$



2D Geometry cont.

- line in normal form:

  $\vec{n}$ orthogonal unit vector, i.e. $||\vec{n}|| = 1, \langle \vec{n}, \vec{q} - \vec{p} \rangle = 0$

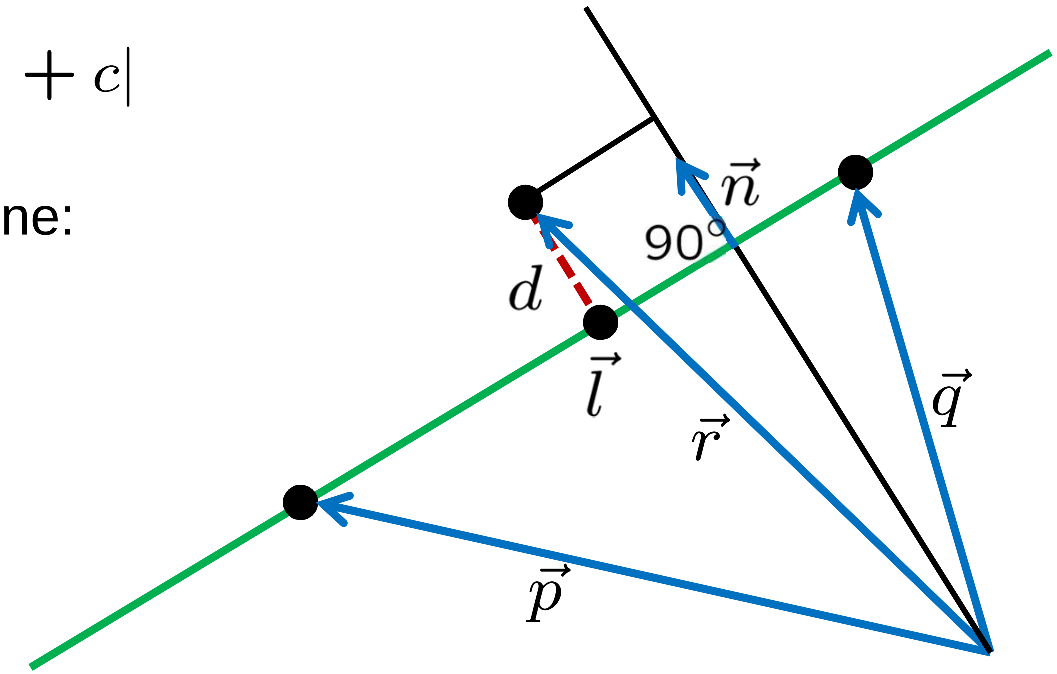
  $$0 = \langle \vec{n}, \vec{x} \rangle - \langle \vec{n}, \vec{p} \rangle$$
  $$= \langle \vec{n}, \vec{x} \rangle + c \qquad \text{(normal form)}$$

- distance of point from line:

  $$d = ||\vec{l} - \vec{r}|| = |\langle \vec{n}, \vec{r} \rangle + c|$$

- an (arbitrary) point on the line:

  $$-c\vec{n}$$



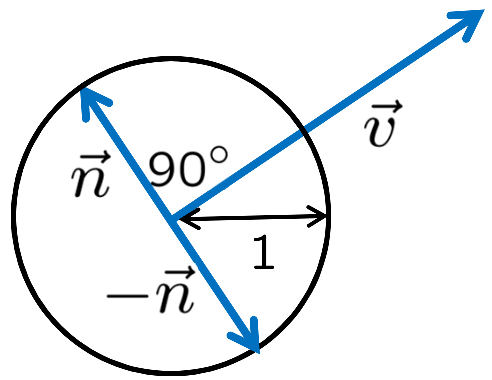
2D Geometry cont.

– unit normal vector:

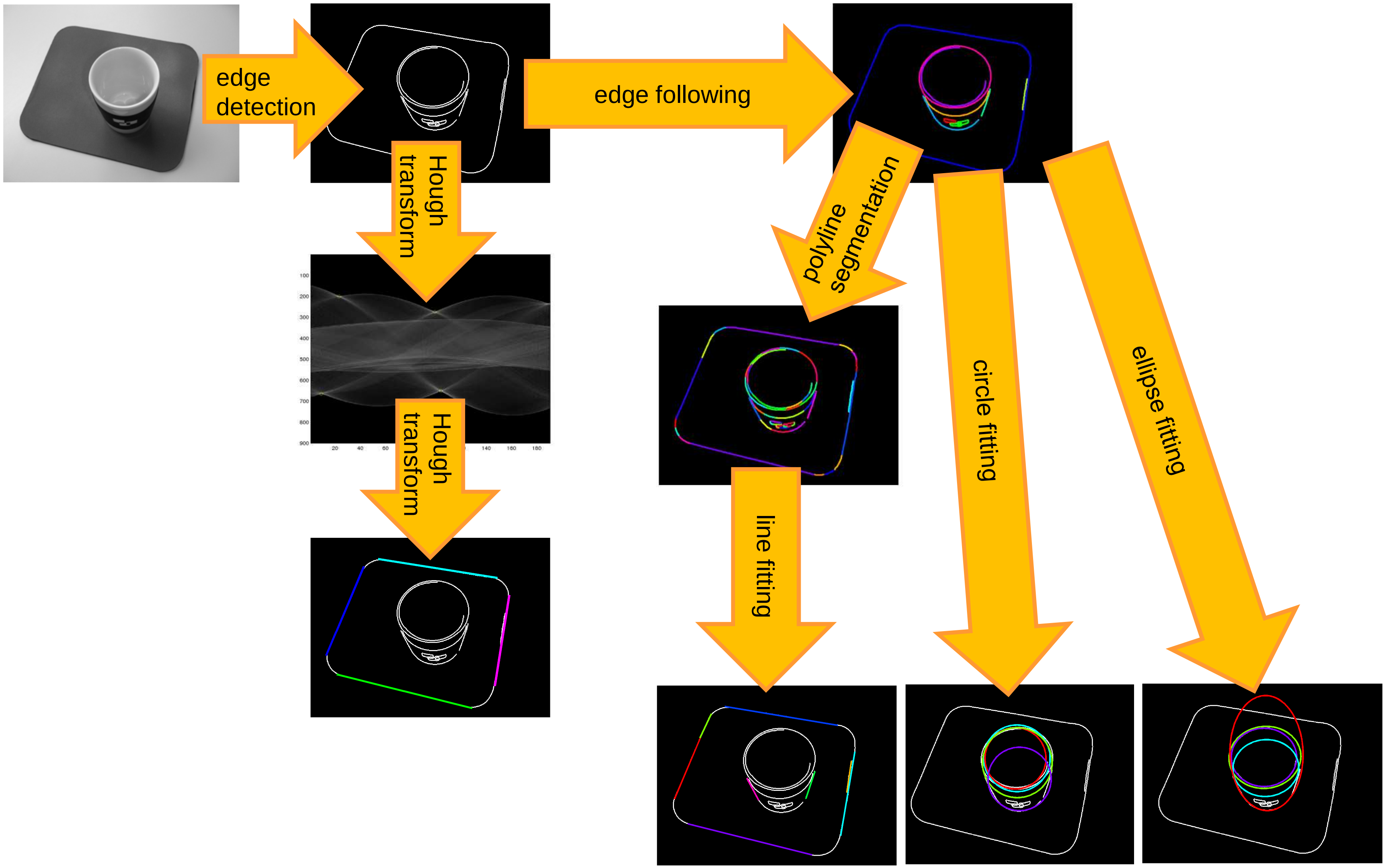
$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

$$\vec{n} = \frac{1}{||\vec{v}||} \begin{pmatrix} -v_2 \\ v_1 \end{pmatrix}$$

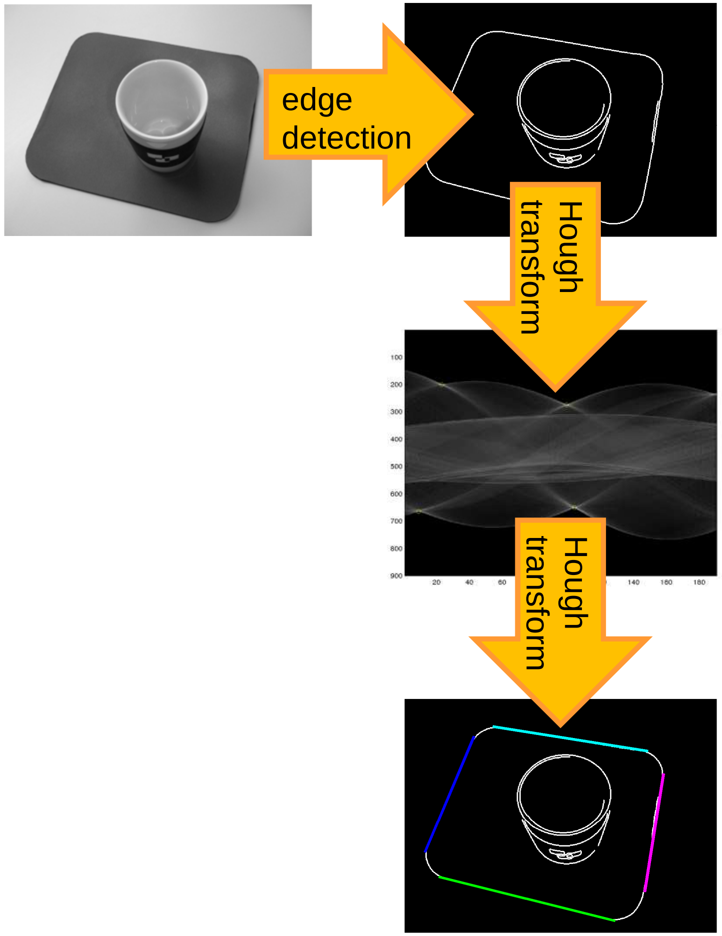
$$\rightarrow \qquad ||\vec{n}|| = 1, \vec{n} \perp \vec{v}$$



Contours Detection



Contours Detection



Hough Transform

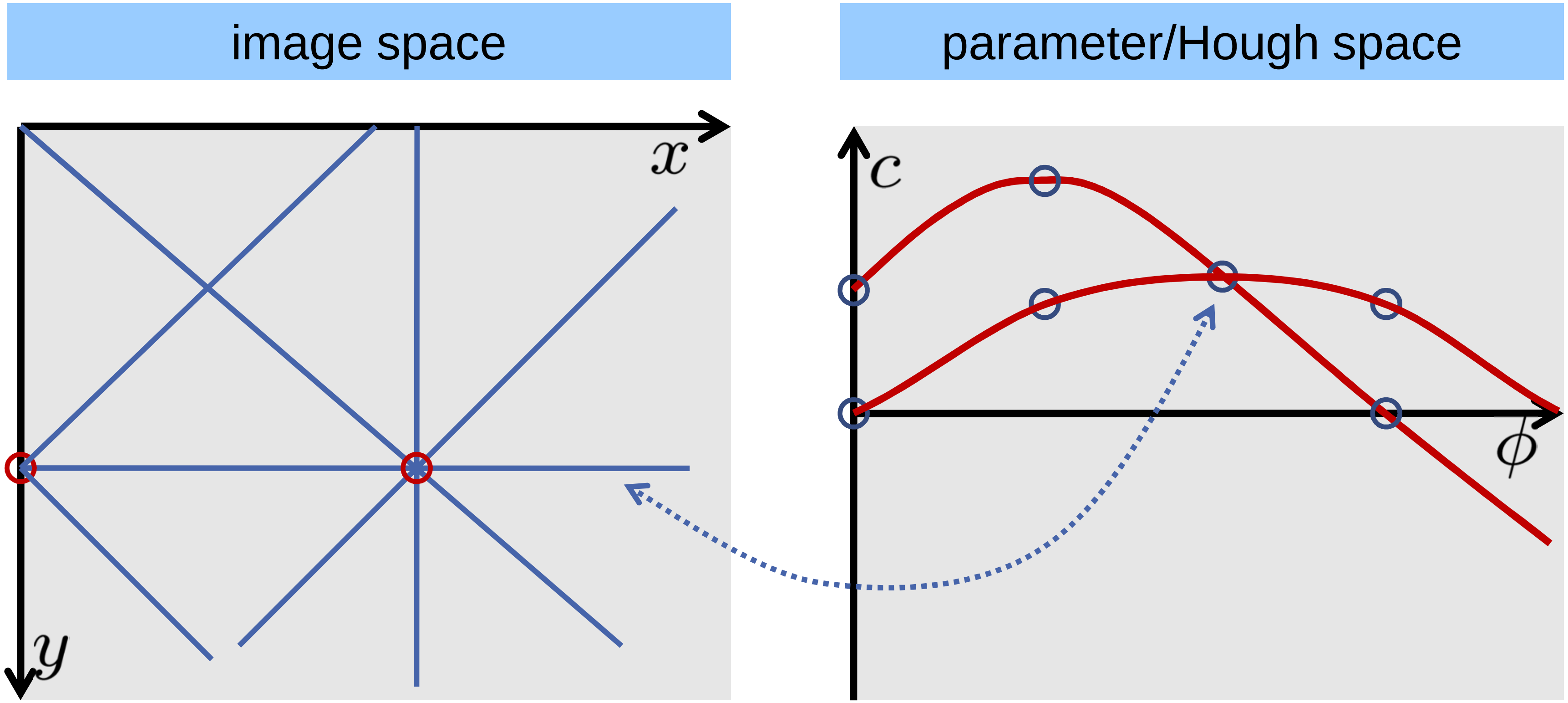
- find lines in edge bitmaps
  - idea: every line can be represented in 2D as:

    $x \cdot \cos\phi + y \cdot \sin\phi + c = 0$

    with $0^\circ \leq \phi < 180^\circ$ and $c \in \mathbb{R}$

  - 2D-space of all lines represented by (ϕ, c)

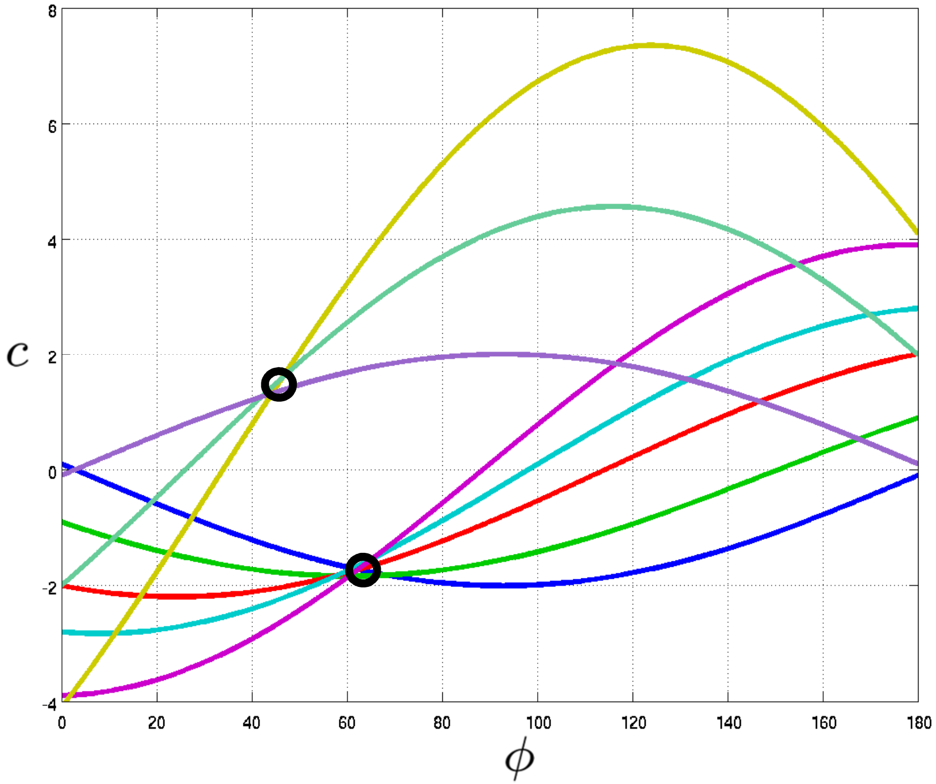
Hough Transform cont.



lines	⟷	points
points	⟷	sine curves
points on a line	⟷	intersecting sine curves

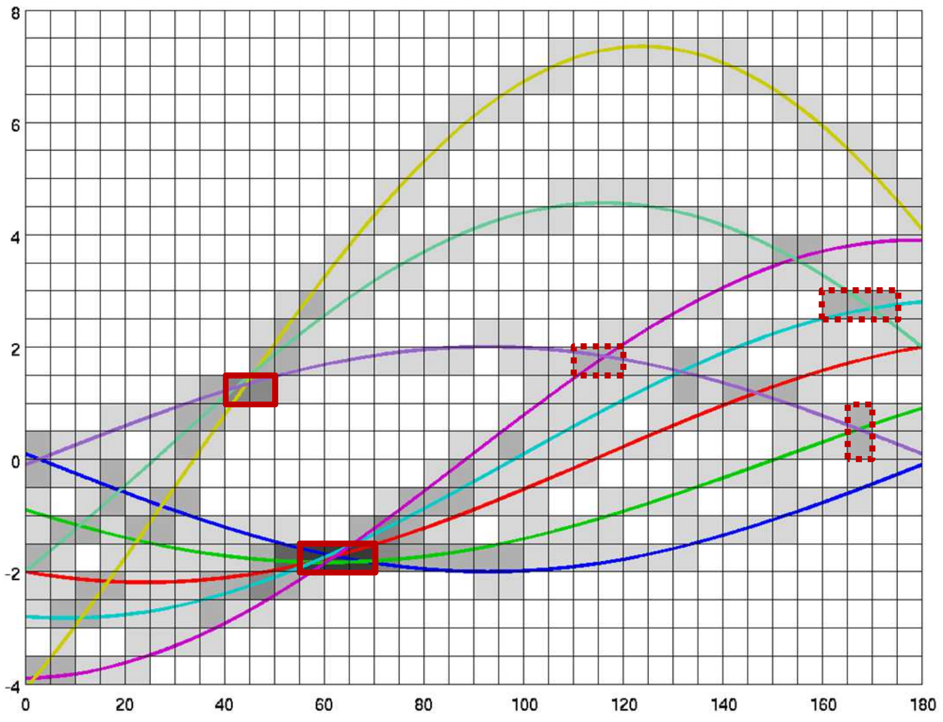
Hough Transform cont.

- basic procedure:
  1. calculate/draw sine curves in Hough space referring to edge points
  2. calculate point of intersection → line parameters
- in practice:
  - not unique point of intersection
  - mixture of several lines



Hough Transform cont.

- finding areas of “high density” in Hough space
  - use discrete array of accumulator cells
  - for every cell count the number of sine curves that go through
  - local maxima in accumulator array refer to line parameters



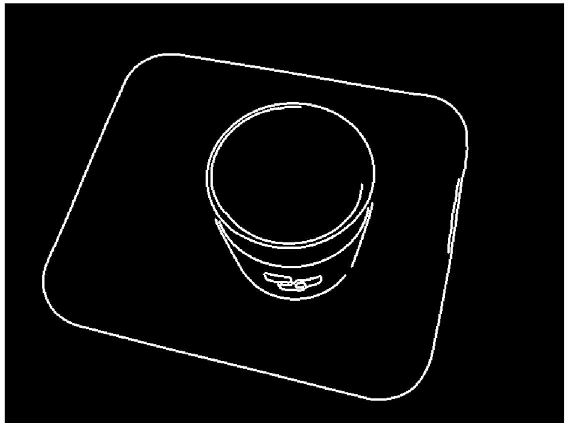
Hough Transform cont.

- Hough transform for many edge points on many edges:
  1. initialize accumulator array of adequate precision with 0
  2. increment all accumulator cells which satisfy the line equation
  3. find local maxima in accumulator array → parameters of most dominant lines in the image

- the mapping from image space to parameter space is also called *Radon transform*

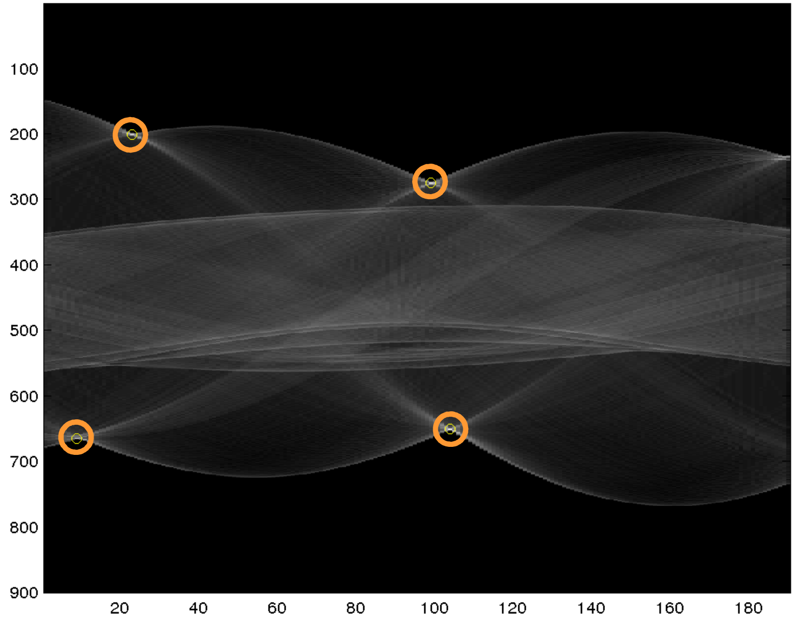
- after having found line parameters the edge pixels with small distance to the lines can be assigned to the line
  - determine starting point and end point of line
  - allow gaps of maximal size

Hough Transform cont.



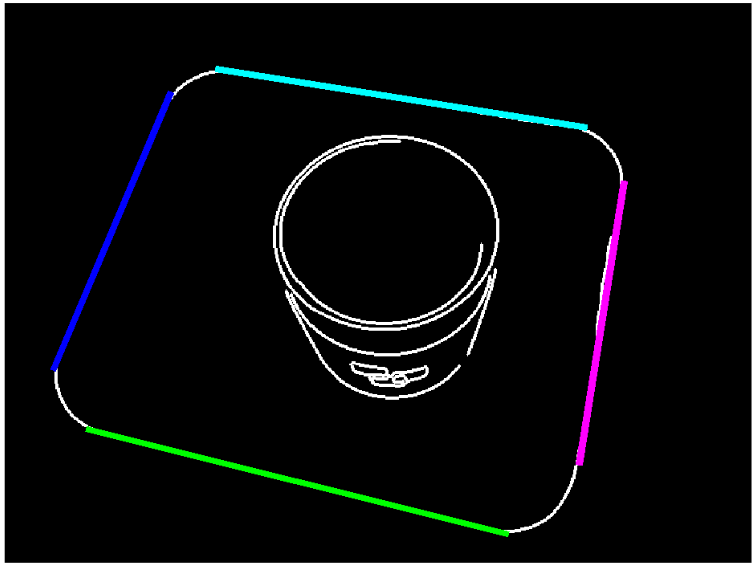
1. edge bitmap (with Canny)

2. Hough parameter space



3. Find local maxima

4. Determine lines belonging to local maxima

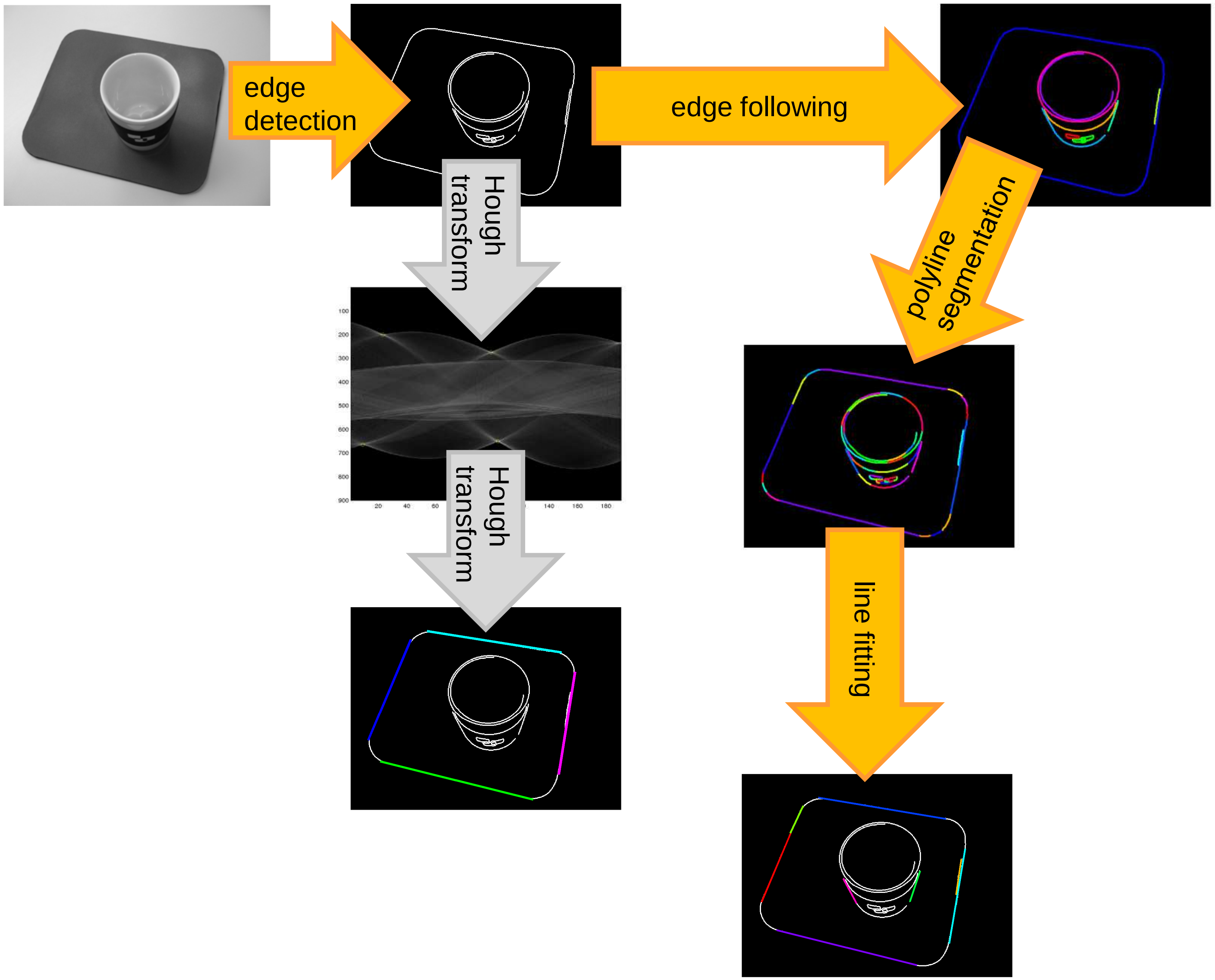


Hough Transform cont.

- properties of Hough transform:
  - result depends on size and precision of accumulator array
  - determining significant peaks in the accumulator array might be difficult in practice
  - gradient direction is ignored
  - accumulator array is flooded in “natural” scenes

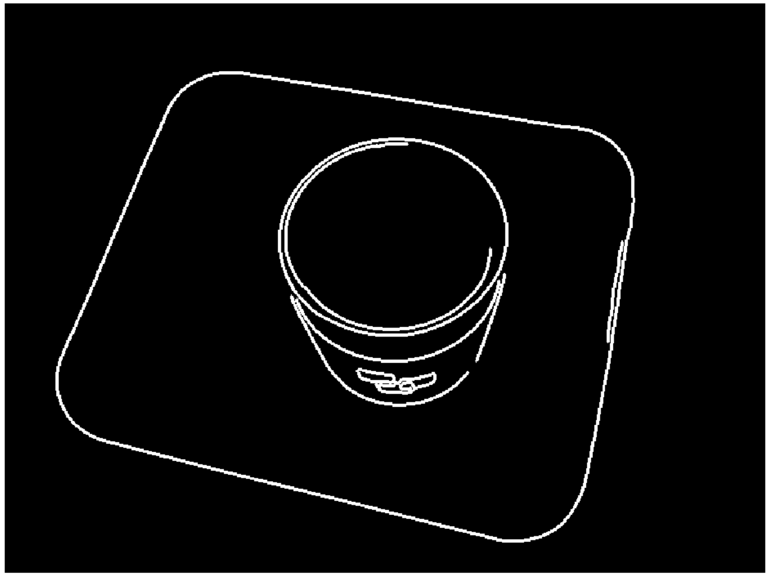
- extensions:
  - extension to other kind of parameterized curves (circles, ellipses, …)
  - randomized Hough transform
  - generalized Hough transform

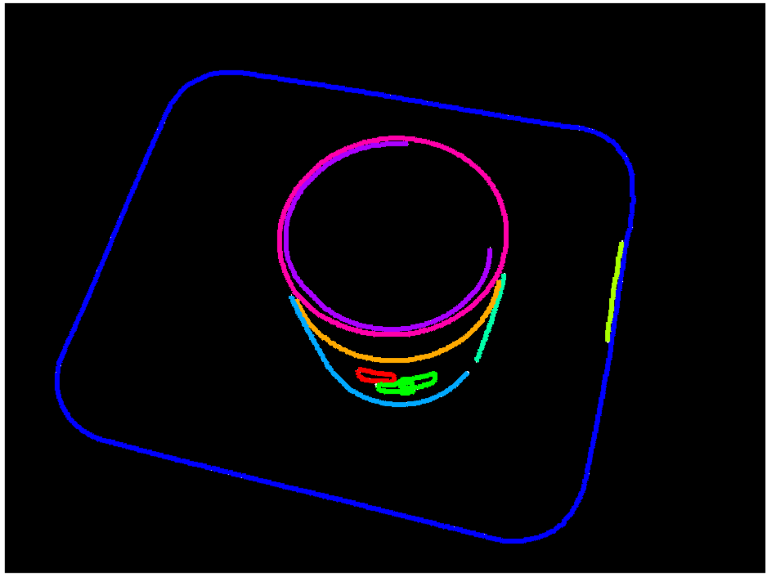
Contours Detection



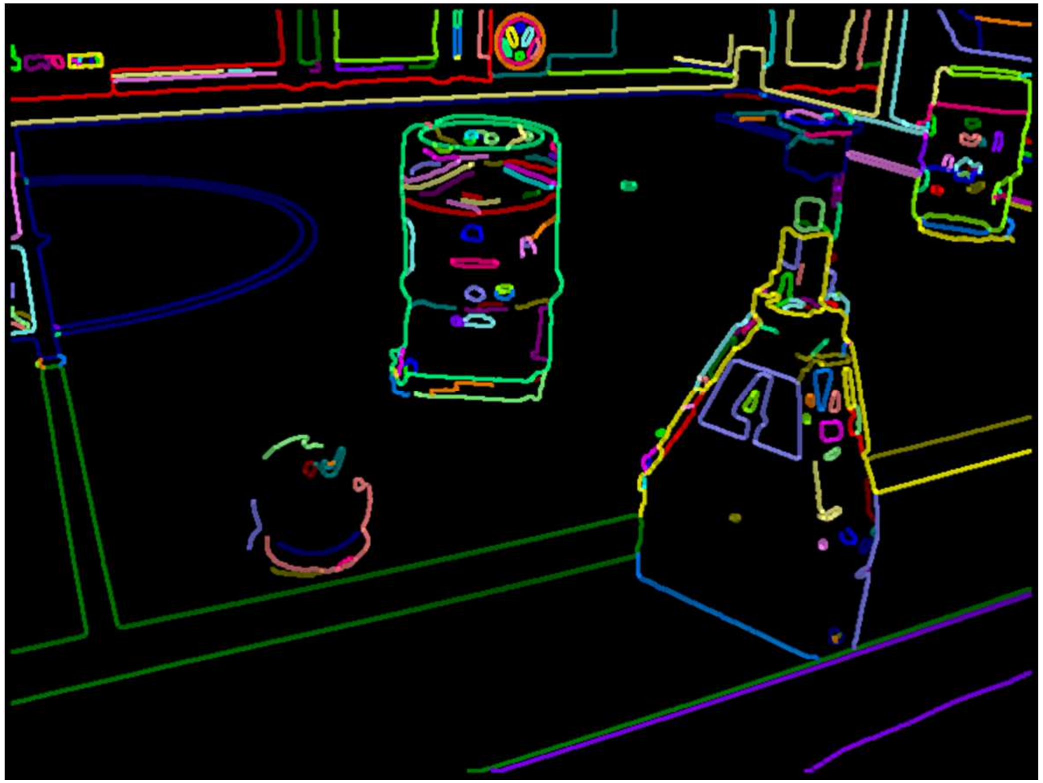
Edge Following

- edge detectors yield bitmaps with edge pixels
- collect all edge pixels and link them in topological order
- use gradient information (if available) for linking
- result: lists of edge pixels that describe a contours



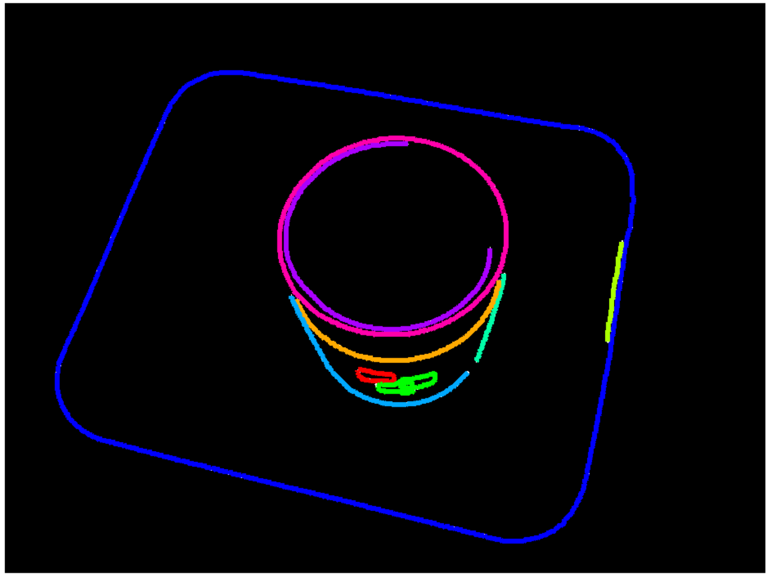


Edge Following cont.



Polyline Segmentation

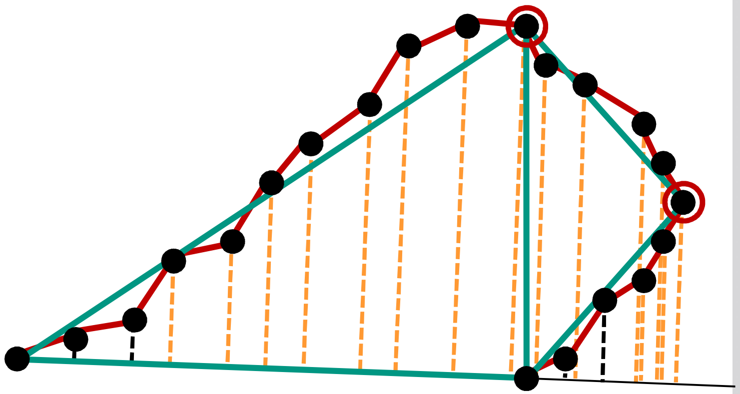
- edge following yields ordered lists of pixels
- these do not automatically represent straight lines
- **Task**: subdivide pixel lists in such a way that the sublists can be represented by line segments
- Several algorithms. Here, we only consider the Ramer–Douglas–Peucker algorithm

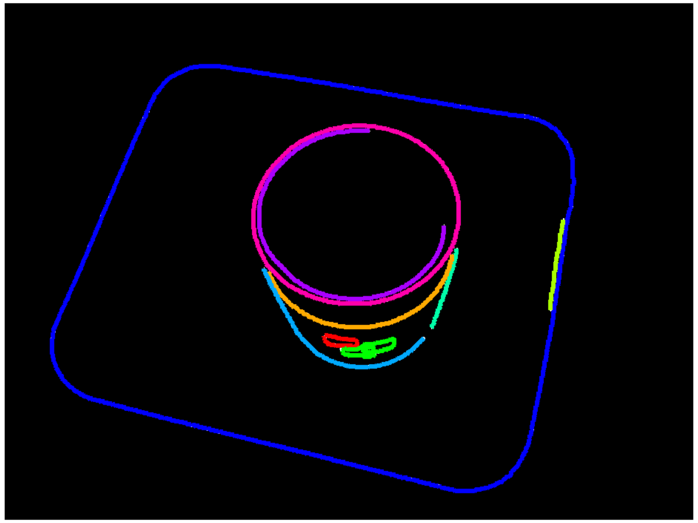


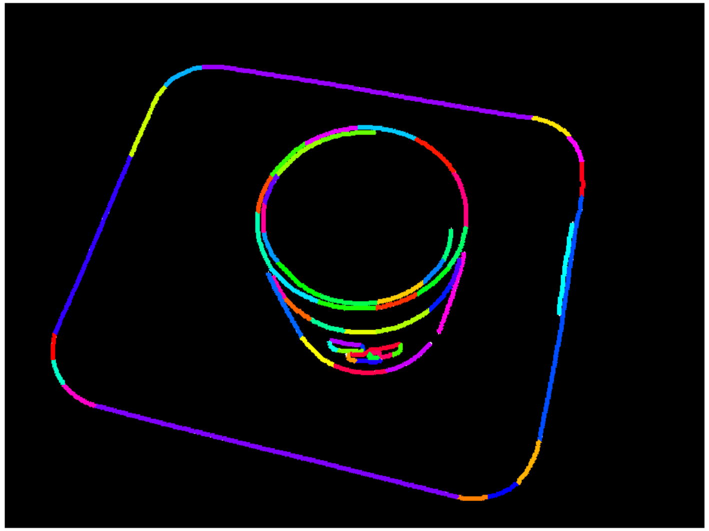
Ramer-Douglas-Peucker Algorithm

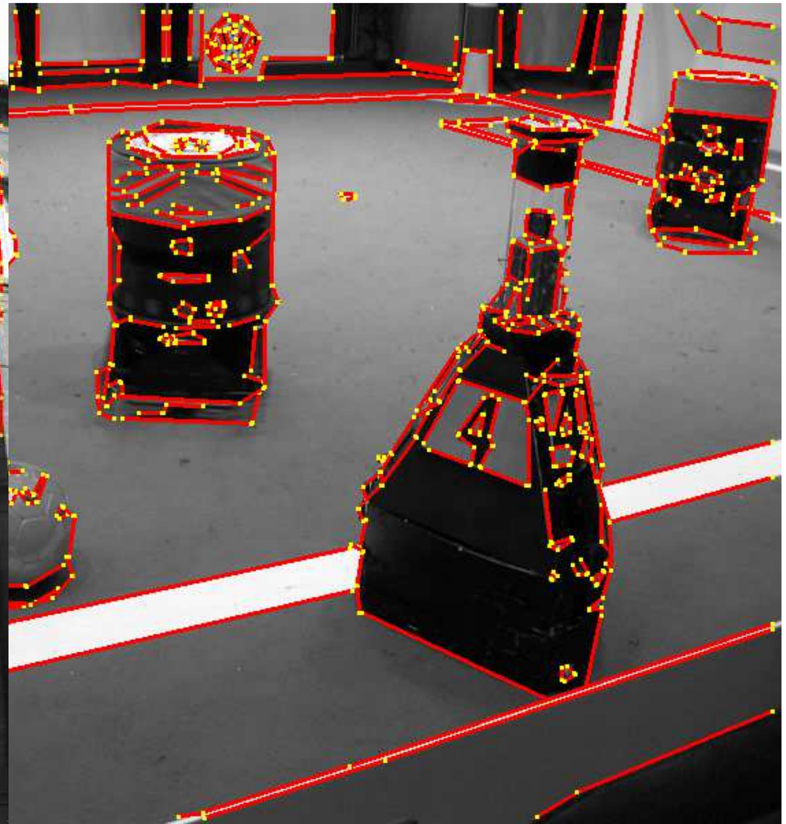
– basic idea: subdivide polyline recursively at the farthest vertex

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

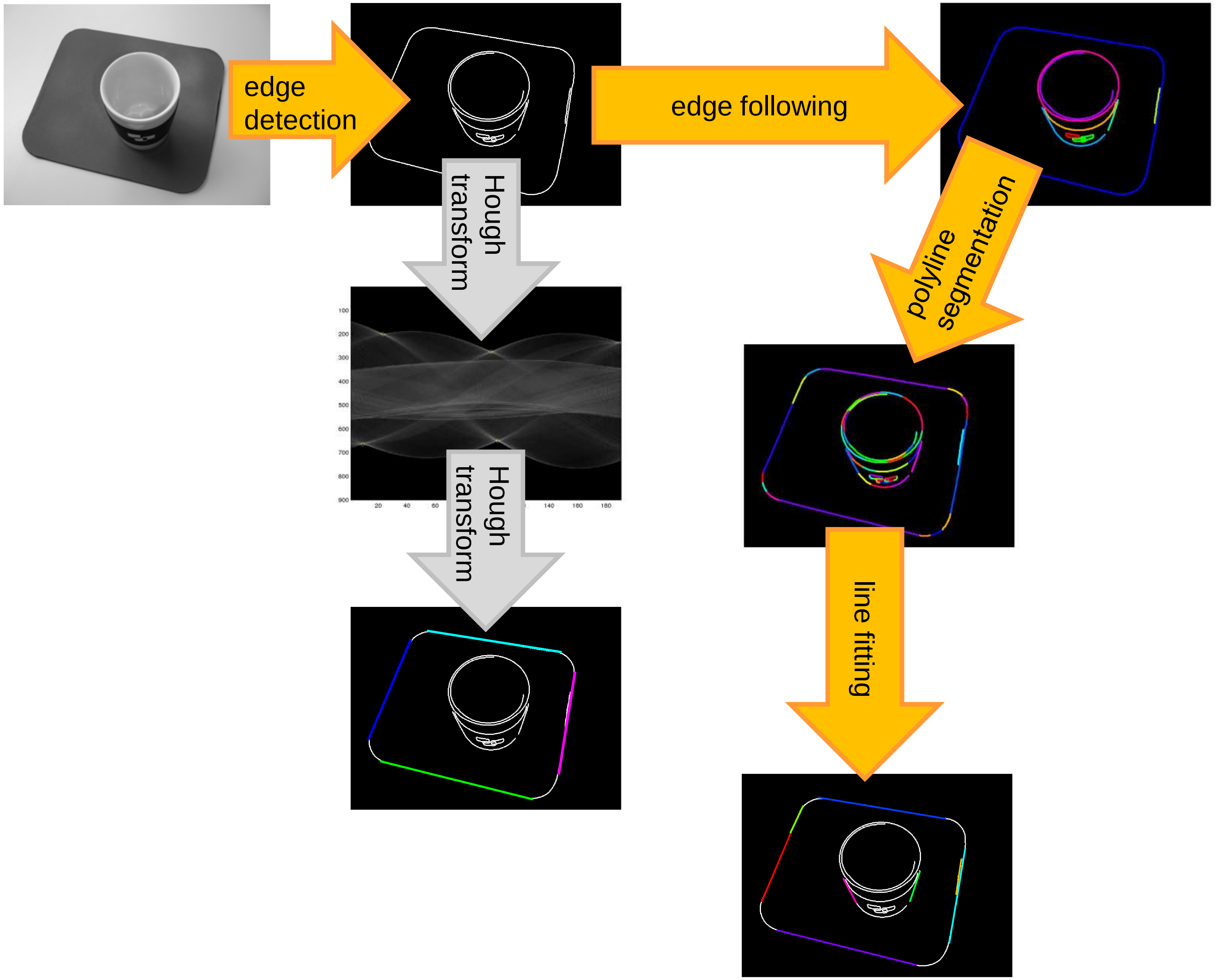








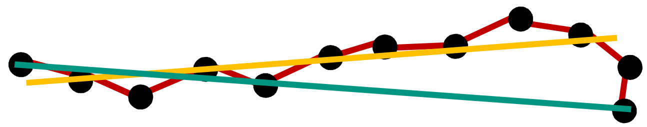
Contours Detection



LINE FITTING

Line Estimation

- result of polygonal chain splitting is suboptimal
- result of Hough transform may be suboptimal



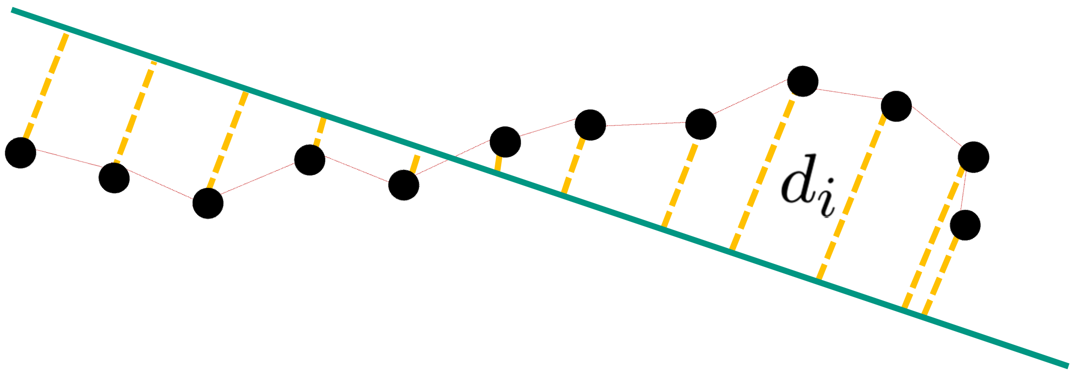
→ algorithms for accurate line estimation

Line Estimation cont.

- line parameters: $\vec{n}, c$
- which parameters are optimal ?
  - given $\vec{n}, c$ we can determine the distance of a point $\vec{x}_i$ to the line

    $d_i = |\langle \vec{n}, \vec{x}_i \rangle + c|$

  - search the line parameters that minimize $d_1, d_2, \ldots, d_n$



Total least squares

- total least squares approach

$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{N} d_i^2$$
$$subject\ to\ \langle \vec{n}, \vec{n} \rangle = 1$$

– Langrange function:

$$\mathcal{L}(\vec{n}, c, \lambda) = \sum_{i=1}^{N} d_i^2 - \lambda(\langle \vec{n}, \vec{n} \rangle - 1)$$
$$= \sum_{i=1}^{N} (\langle \vec{n}, \vec{x}_i \rangle + c)^2 - \lambda(\langle \vec{n}, \vec{n} \rangle - 1)$$

– zeroing partial derivative w.r.t. *c*:

$$\frac{\partial \mathcal{L}}{\partial c} = 2 \sum_i \langle \vec{n}, \vec{x}_i \rangle + 2Nc \stackrel{!}{=} 0$$
$$\rightarrow c = -\frac{1}{N} \sum_i \langle \vec{n}, \vec{x}_i \rangle$$

Total least squares cont.

– zeroing partial derivative w.r.t. n:

$$\frac{\partial \mathcal{L}}{\partial n_1} = 2(\sum_i x_{i,1}^2)n_1 + 2(\sum_i x_{i,1}x_{i,2})n_2 + 2(\sum_i x_{i,1})c - 2\lambda n_1 \stackrel{!}{=} 0$$

$$\frac{\partial \mathcal{L}}{\partial n_2} = 2(\sum_i x_{i,1}x_{i,2})n_1 + 2(\sum_i x_{i,2}^2)n_2 + 2(\sum_i x_{i,2})c - 2\lambda n_2 \stackrel{!}{=} 0$$

– substituting c by $-\frac{1}{N}\sum_i \langle \vec{n}, \vec{x}_i \rangle$:

$$\underbrace{(\sum_i x_{i,1}^2 - \frac{1}{N}(\sum_i x_{i,1})^2)}_{=:\alpha} n_1 + \underbrace{(\sum_i x_{i,1}x_{i,2} - \frac{1}{N}\sum_i x_{i,1}\sum_i x_{i,2})}_{=:\beta} n_2 = \lambda n_1$$

$$\underbrace{(\sum_i x_{i,1}x_{i,2} - \frac{1}{N}\sum_i x_{i,1}\sum_i x_{i,2})}_{=\beta} n_1 + \underbrace{(\sum_i x_{i,2}^2 - \frac{1}{N}(\sum_i x_{i,2})^2)}_{=:\gamma} n_2 = \lambda n_2$$

Total least squares cont.

– rewriting as matrix equation:

$$\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix} \vec{n} = \lambda \vec{n}$$

– hence: λ is Eigenvalue and $\vec{n}$ is Eigenvector of $\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}$

– two solutions of Eigenvalue problem: $\lambda_1 \geq \lambda_2 \geq 0$

$\rightarrow$ λ_2 minimises distances
$\rightarrow$ λ_1 maximises distances

Total least squares cont.

- recipe: line estimation with *total least squares*:
  1. calculate from all edge pixels:
     $$\sum_i x_{i,1},\ \sum_i x_{i,2},\ \sum_i x_{i,1}^2,\ \sum_i x_{i,2}^2,\ \sum_i x_{i,1}x_{i,2}$$
  2. calculate Eigenvalues and Eigenvectors of matrix $\begin{pmatrix}\alpha & \beta \\ \beta & \gamma\end{pmatrix}$
     $\rightarrow \vec{n}, \lambda \quad \text{(take the smaller Eigenvalue)}$
  3. calculate c from $\vec{n}$
  4. if you are interested in line segments, determine start and end point from edge pixels projected on the line

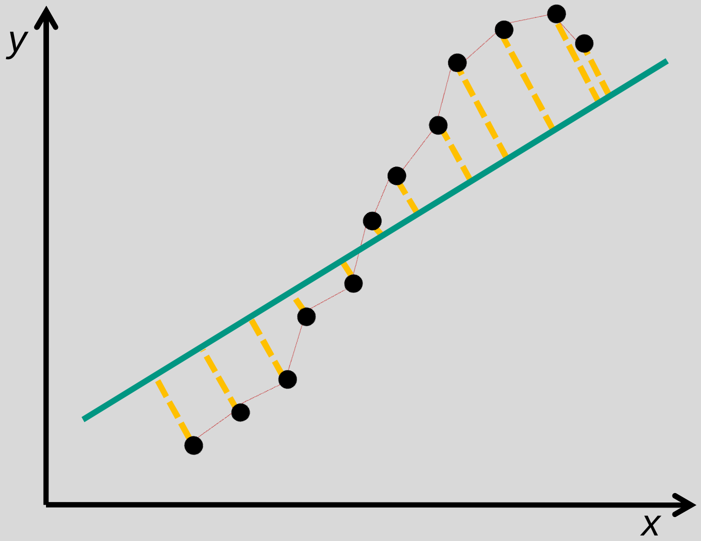
Total least squares cont.

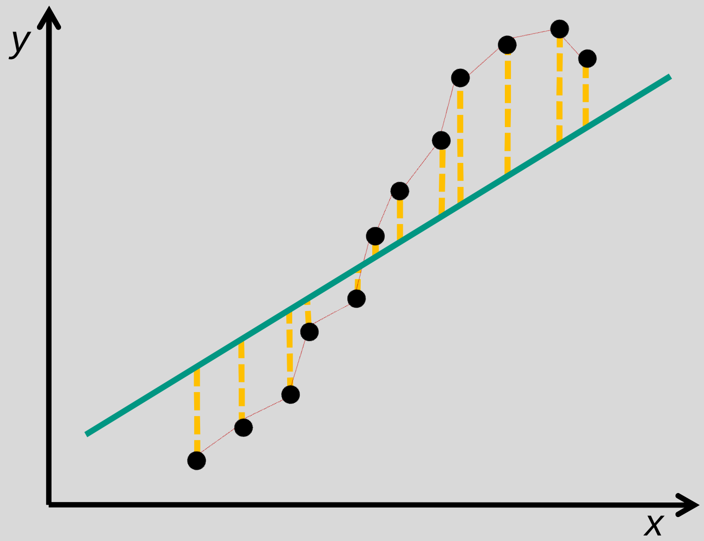
after line
splitting

after total
least squares

Total least squares cont.

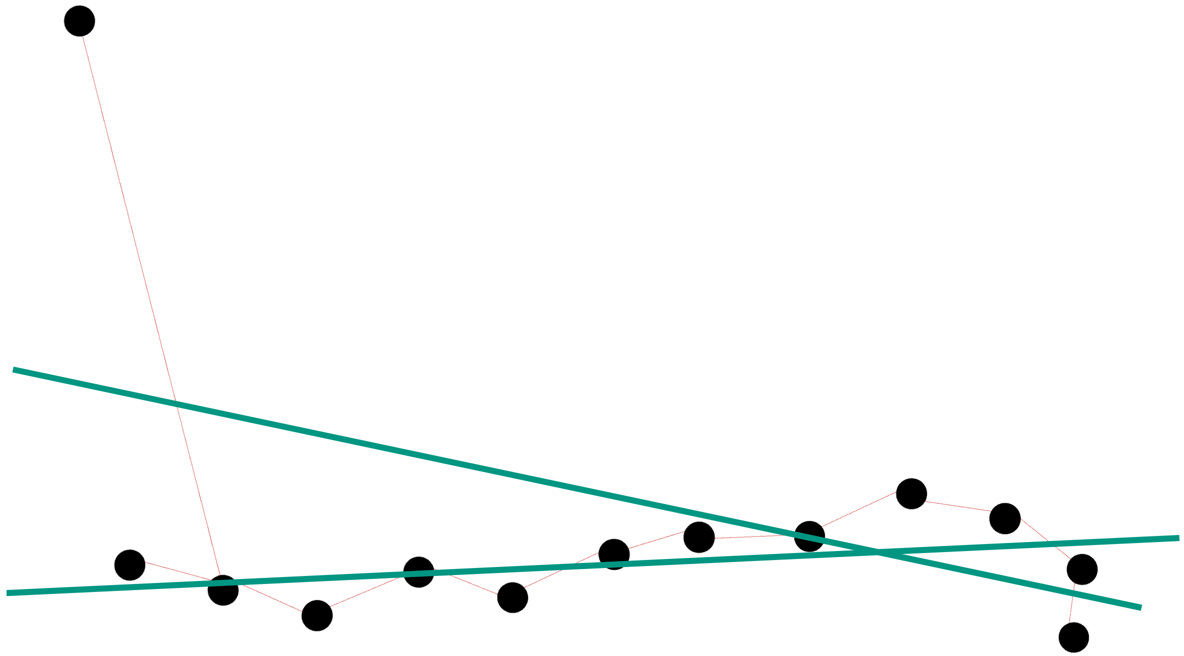
total least squares	ordinary least squares
• treat *x* and *y* alike • isotropic • minimize orthogonal distances	• interpret *y* = *y(x)* • anisotropic • minimize distances in y only





Line Estimation cont.

- robustness concerning outliers:



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

Line Estimation cont.

- robustness ideas:
  - reduce influence of gross outliers
    → weighted least squares, M-estimators
  - ignore outliers
    →LTS, RANSAC

Weighted Least Squares

- Idea:
  - edge points should have different influence
  - influence of outliers should be small, influence of reliable points large

- Approach:
  - introduce weights $w_i \geq 0$, one for each edge point
  - solve:

=1 for vanilla least squares

$$\begin{aligned} & \underset{\vec{n},c}{minimise} \sum_{i=1}^{N} w_i d_i^2 \\ & subject\ to\ \langle \vec{n}, \vec{n} \rangle = 1 \end{aligned}$$

Weighted Least Squares cont.

- Solution:
  - using Lagrange multipliers as before yields:

$$c = -\frac{1}{W}\sum_i w_i \langle \vec{n}, \vec{x}_i \rangle$$

$$\underbrace{\Big(\sum_i w_i x_{i,1}^2 - \frac{1}{W}\big(\sum_i w_i x_{i,1}\big)^2\Big)}_{=:\tilde{\alpha}} n_1 + \underbrace{\Big(\sum_i w_i x_{i,1} x_{i,2} - \frac{1}{W}\sum_i w_i x_{i,1} \sum_i w_i x_{i,2}\Big)}_{=:\tilde{\beta}} n_2 = \lambda n_1$$

$$\underbrace{\Big(\sum_i w_i x_{i,1} x_{i,2} - \frac{1}{W}\sum_i w_i x_{i,1} \sum_i w_i x_{i,2}\Big)}_{=\tilde{\beta}} n_1 + \underbrace{\Big(\sum_i w_i x_{i,2}^2 - \frac{1}{W}\big(\sum_i w_i x_{i,2}\big)^2\Big)}_{=:\tilde{\gamma}} n_2 = \lambda n_2$$

$$W = \sum_i w_i$$

  - yields Eigenvalue problem as before

$$\begin{pmatrix} \tilde{\alpha} & \tilde{\beta} \\ \tilde{\beta} & \tilde{\gamma} \end{pmatrix} \vec{n} = \lambda \vec{n}$$

M-estimators

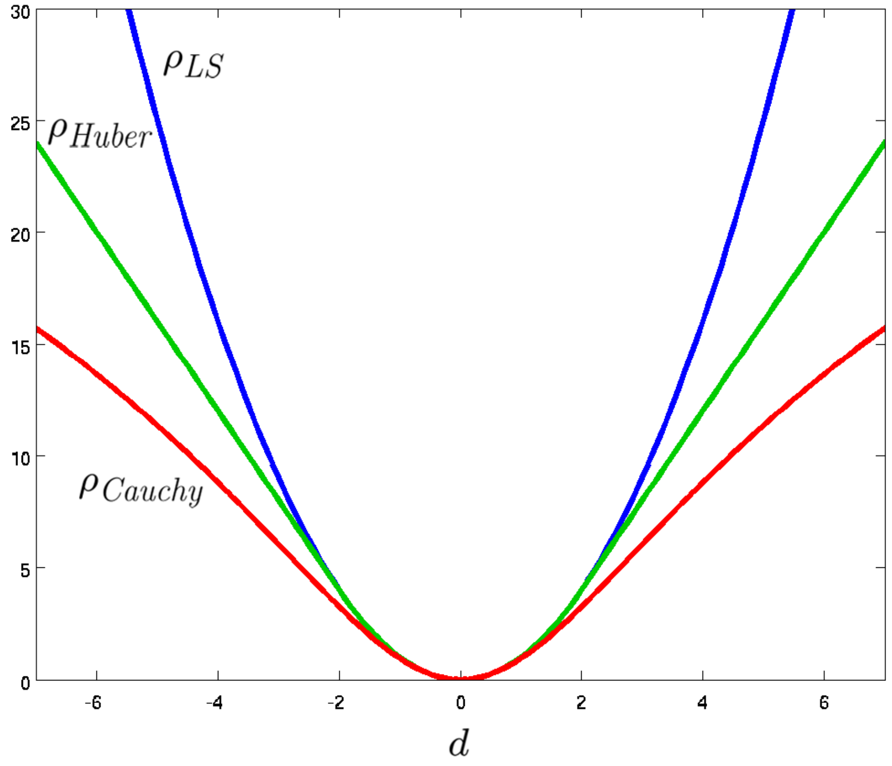
- Choice of w_i ?

- Vanilla least squares:
  - distances enter error term as squares

$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{N} d_i^2$$
$$subject\ to\ \langle \vec{n}, \vec{n} \rangle = 1$$

  - idea: replace d_i^2 by a term that grows more slowly

M-Estimators cont.



$$\rho_{LS} : d \mapsto d^2$$

$$\rho_{Cauchy} : d \mapsto c^2 \log(1 + \frac{d^2}{c^2})$$

$$\rho_{Huber} : d \mapsto \begin{cases} d^2 & \text{if } |d| \leq k \\ 2k|d| - k^2 & \text{otherwise} \end{cases}$$

M-Estimators cont.

- *M-estimators:*

$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{N} \rho(d_i)$$
$$subject\ to\ \langle \vec{n}, \vec{n} \rangle = 1$$

with suitable function ρ

- Lagrange function:

$$\mathcal{L}_M(\vec{n}, c, \lambda) = \sum_{i=1}^{N} \rho(d_i) - \lambda(\langle \vec{n}, \vec{n} \rangle - 1)$$

compare with Lagrange function of weighted least squares:

$$\mathcal{L}_W(\vec{n}, c, \lambda) = \sum_{i=1}^{N} w_i d_i^2 - \lambda(\langle \vec{n}, \vec{n} \rangle - 1)$$

M-Estimators cont.

- Derivatives of Lagrange function:

$$\left.\begin{aligned}\frac{\partial \mathcal{L}_M(\vec{n}, c, \lambda)}{\partial c} &= \sum_{i=1}^{N} \frac{\partial \rho(d_i)}{\partial d_i} \cdot \frac{\partial d_i}{\partial c} \\ & \hphantom{= \sum_{i=1}^{N} \frac{\partial \rho(d_i)}{\partial d_i} \cdot {}} = \\ \frac{\partial \mathcal{L}_W(\vec{n}, c, \lambda)}{\partial c} &= \sum_{i=1}^{N} w_i \cdot 2d_i \cdot \frac{\partial d_i}{\partial c}\end{aligned}\right\} \text{equal if } w_i = \frac{\frac{\partial \rho(d_i)}{\partial d_i}}{2d_i}$$

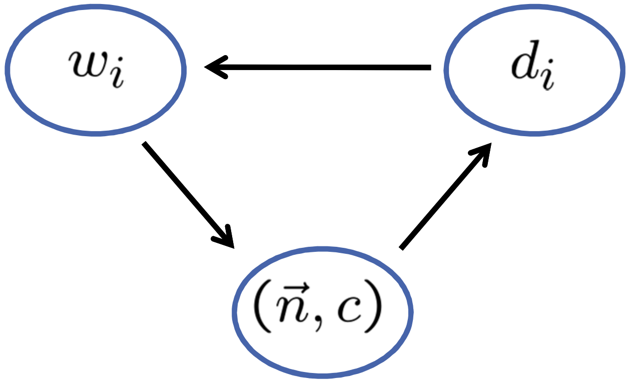
analogous reasoning for $\frac{\partial \mathcal{L}}{\partial \vec{n}}$

$\rightarrow$ running weighted least squares with appropriate weights implements M-estimators

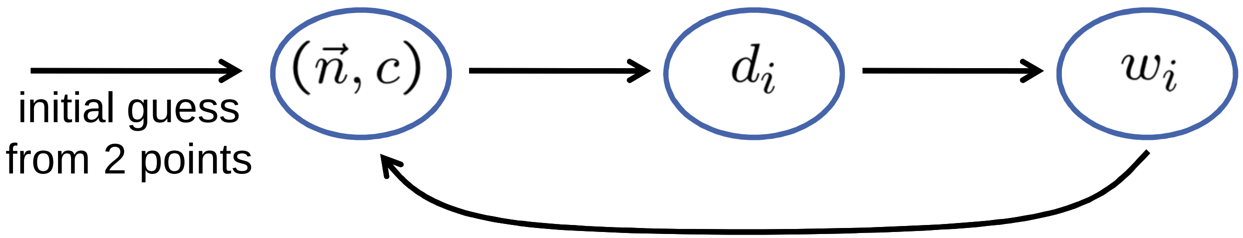
$\rightarrow$ choose weights $w_i = \frac{\frac{\partial \rho(d_i)}{\partial d_i}}{2d_i}$

M-Estimators cont.

- Recurrent dependency



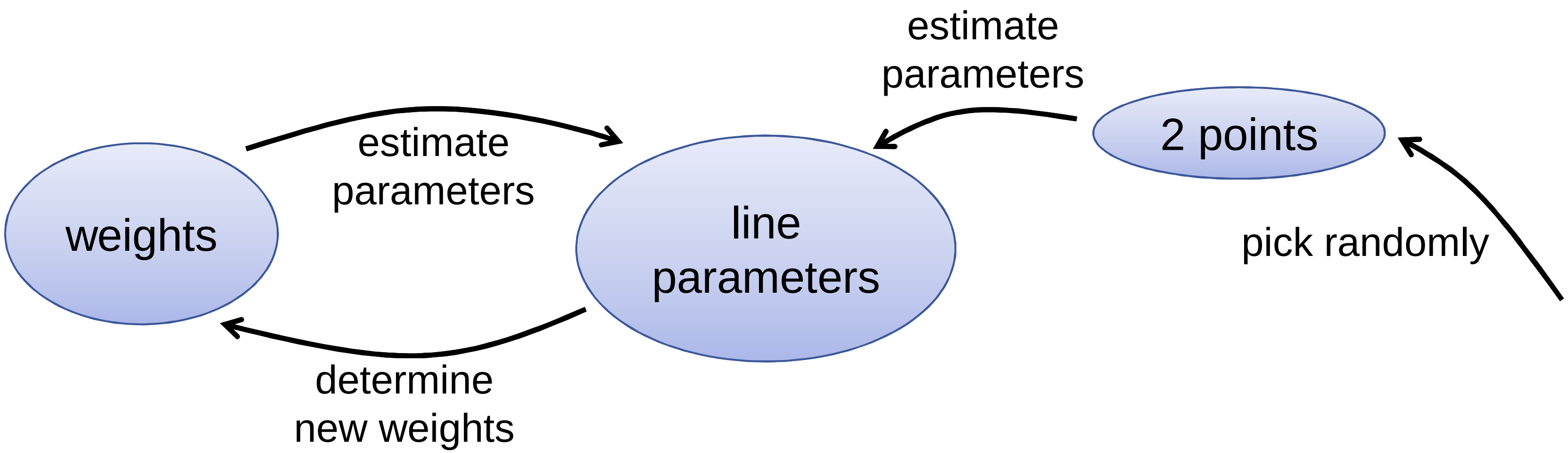
$\rightarrow$ iterative calculation



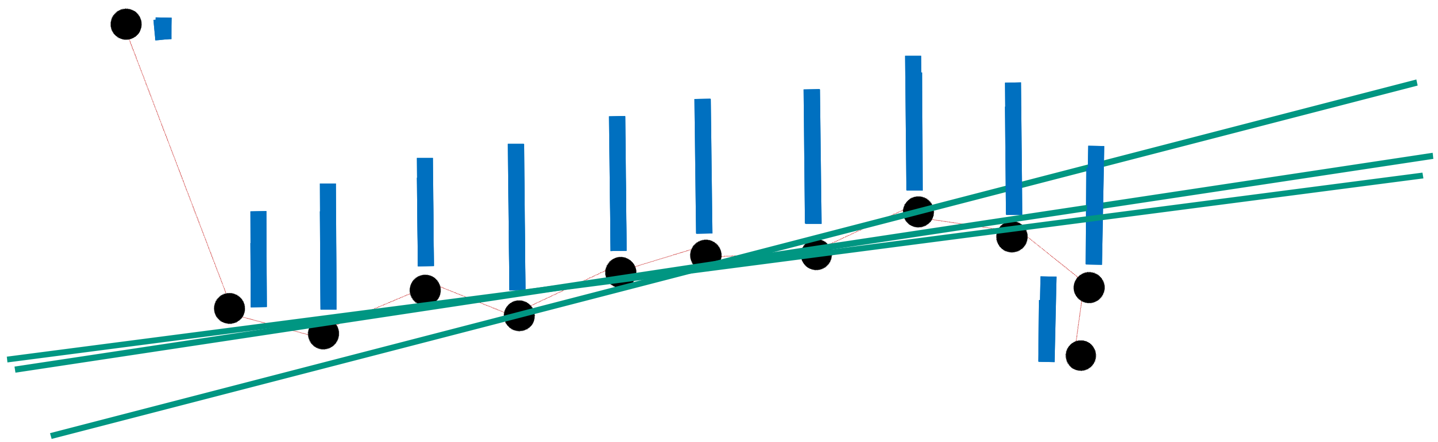
M-Estimators cont.

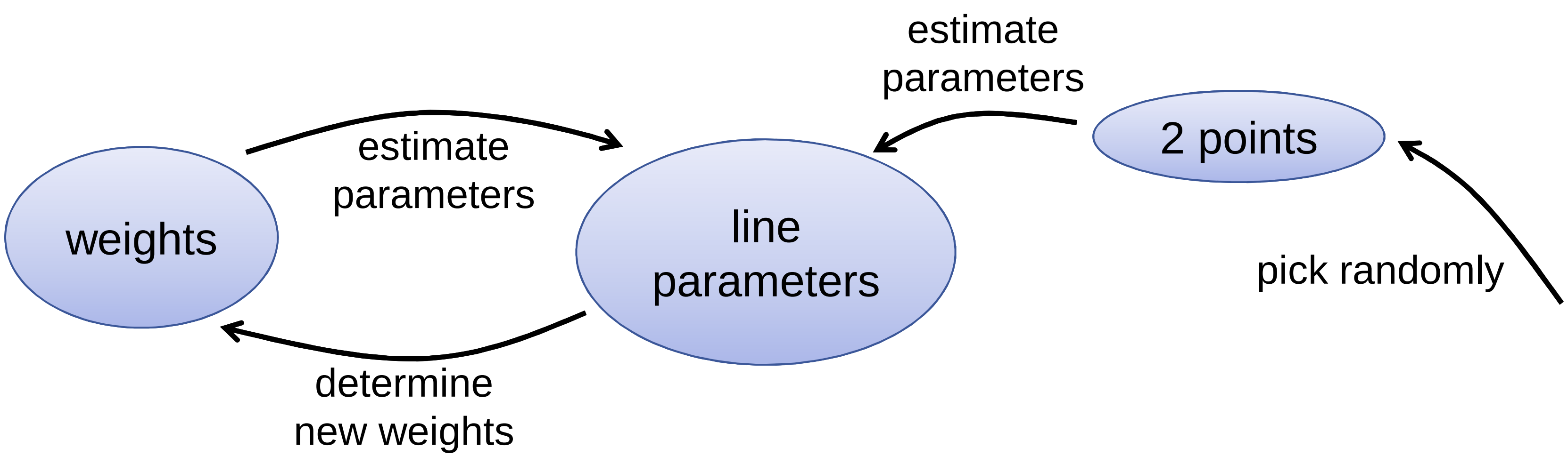
– iterative algorithm:

1. calculate initial guess of line parameters from two (arbitrary) points
2. based on the line parameters found, calculate weights
3. based on the weights, recalculate the line parameters
4. repeat steps 2 and 3 until convergence



M-Estimators cont.





LTS

- *least sum-of-squares estimator* (LS):

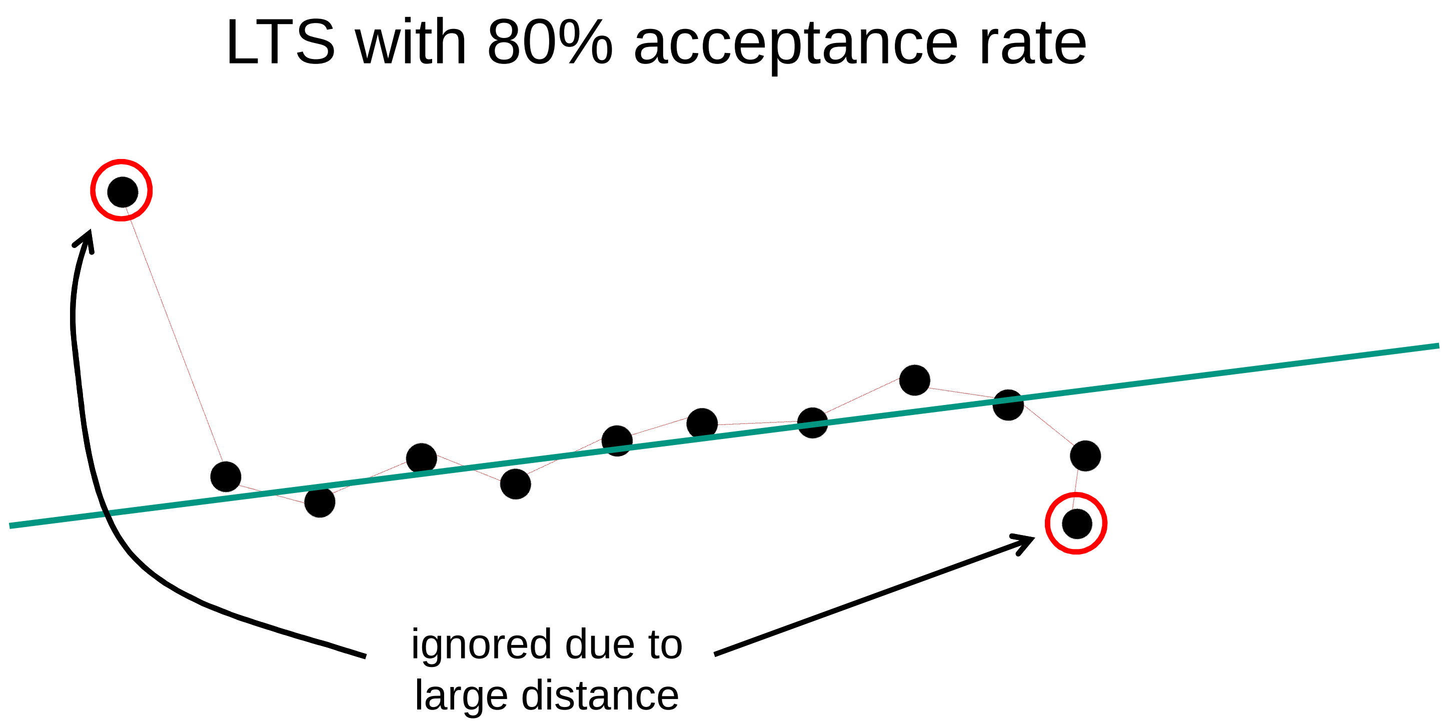
$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{N} d_i^2$$

  is sensitive to outliers because all points contribute to the sum
- *least trimmed-sum-of-squares estimator* (LTS):

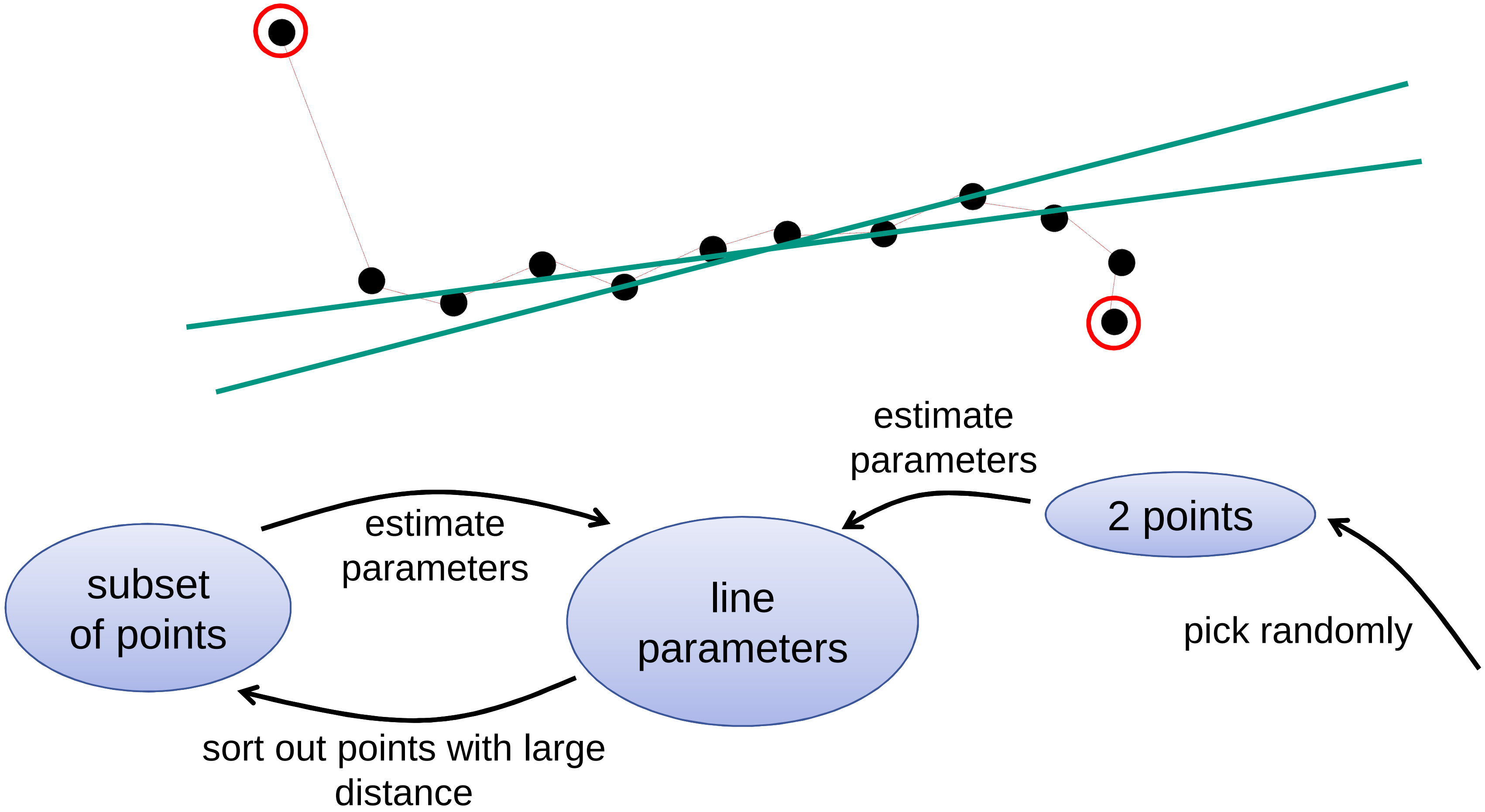
$$\underset{\vec{n},c}{minimise} \sum_{i=1}^{p} d_{i:N}^2$$

  with $p<N$ and $d_{i:N}$ the i-th element in the ordered list of point-distances
  p is typically given as a percentage of N, e.g. 80% of N
- LTS is robust since outliers with large distance do not contribute to the sum

LTS cont.



LTS cont.



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

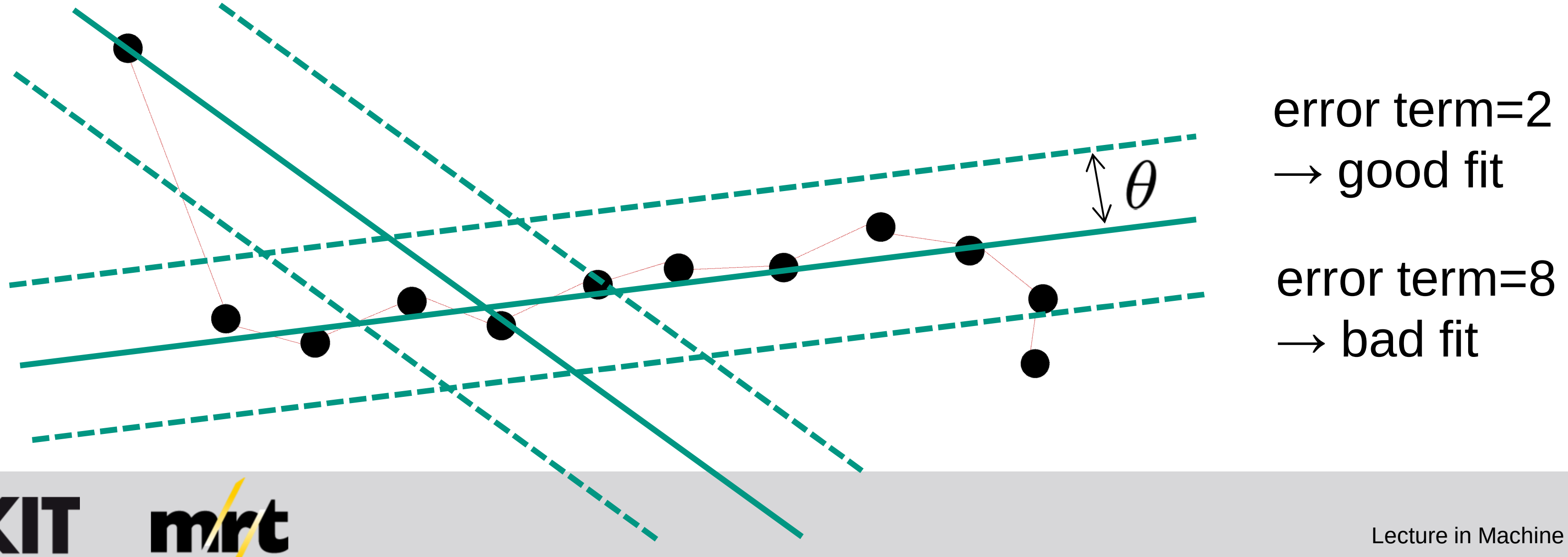
RANSAC

- 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

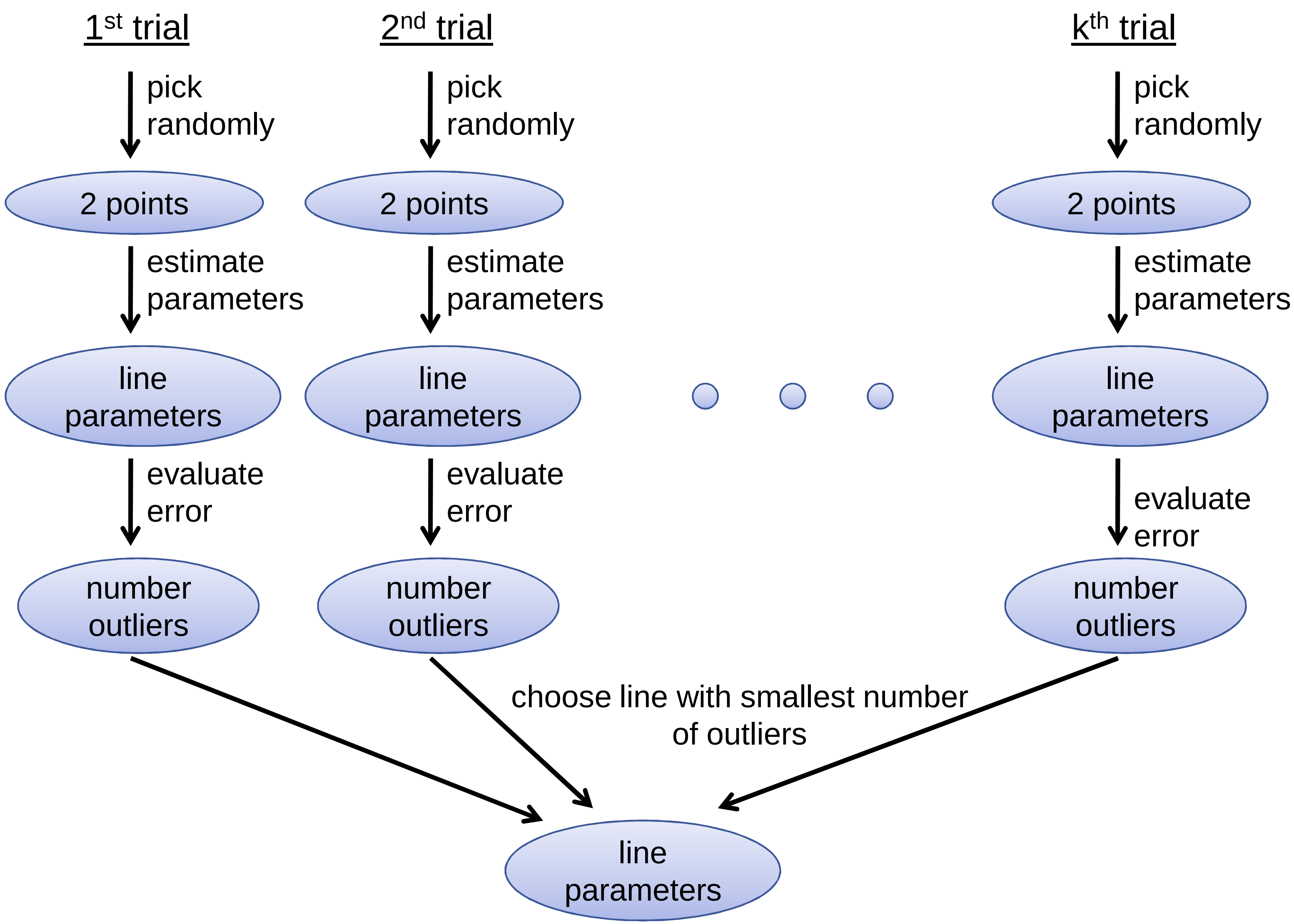


RANSAC cont.

- algorithm:
  - pick randomly two points
  - fit line
  - check the number of points outside the tolerance band (=number of outliers)

  - repeat the process several times with different points
  - select the line with the smallest number of outliers
- RANSAC=<u>ran</u>dom <u>sa</u>mple <u>c</u>onsensus

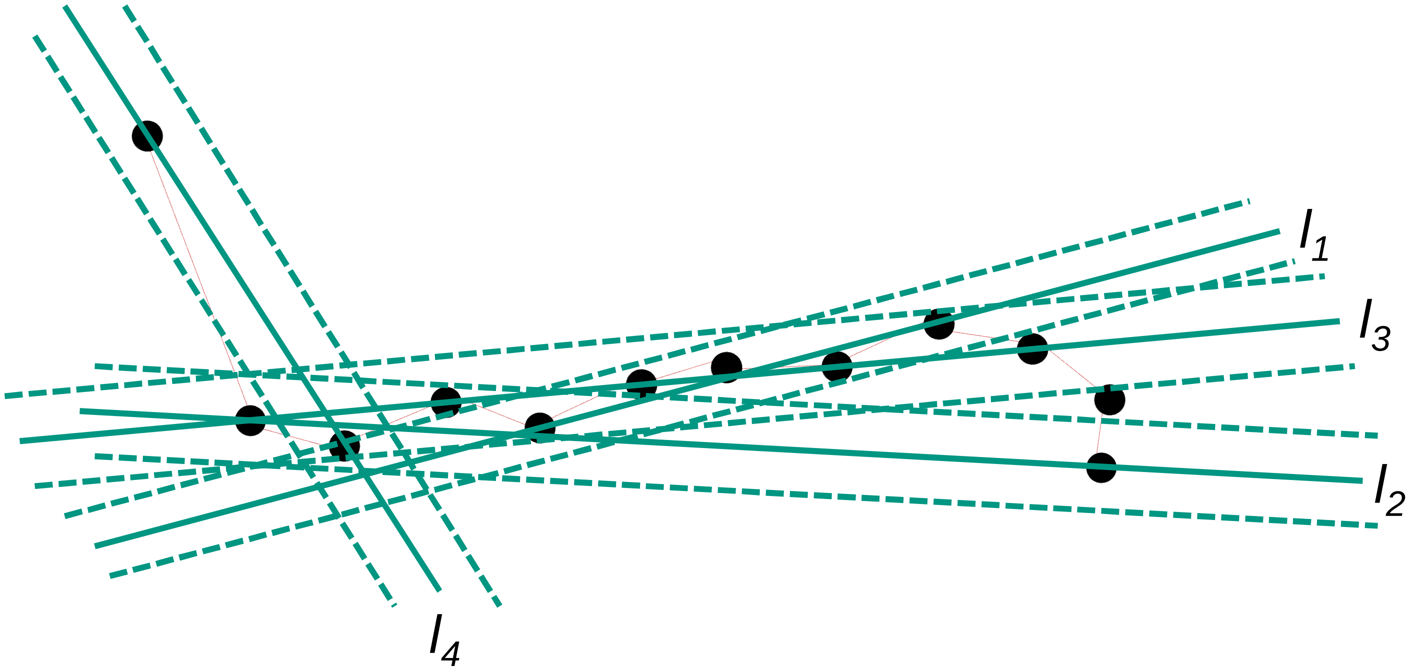
RANSAC cont.



1st trial
2nd trial
kth trial
pick randomly
pick randomly
pick randomly
2 points
2 points
2 points
estimate parameters
estimate parameters
estimate parameters
line parameters
line parameters
line parameters
evaluate error
evaluate error
evaluate error
number outliers
number outliers
number outliers
choose line with smallest number of outliers
line parameters

RANSAC cont.

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

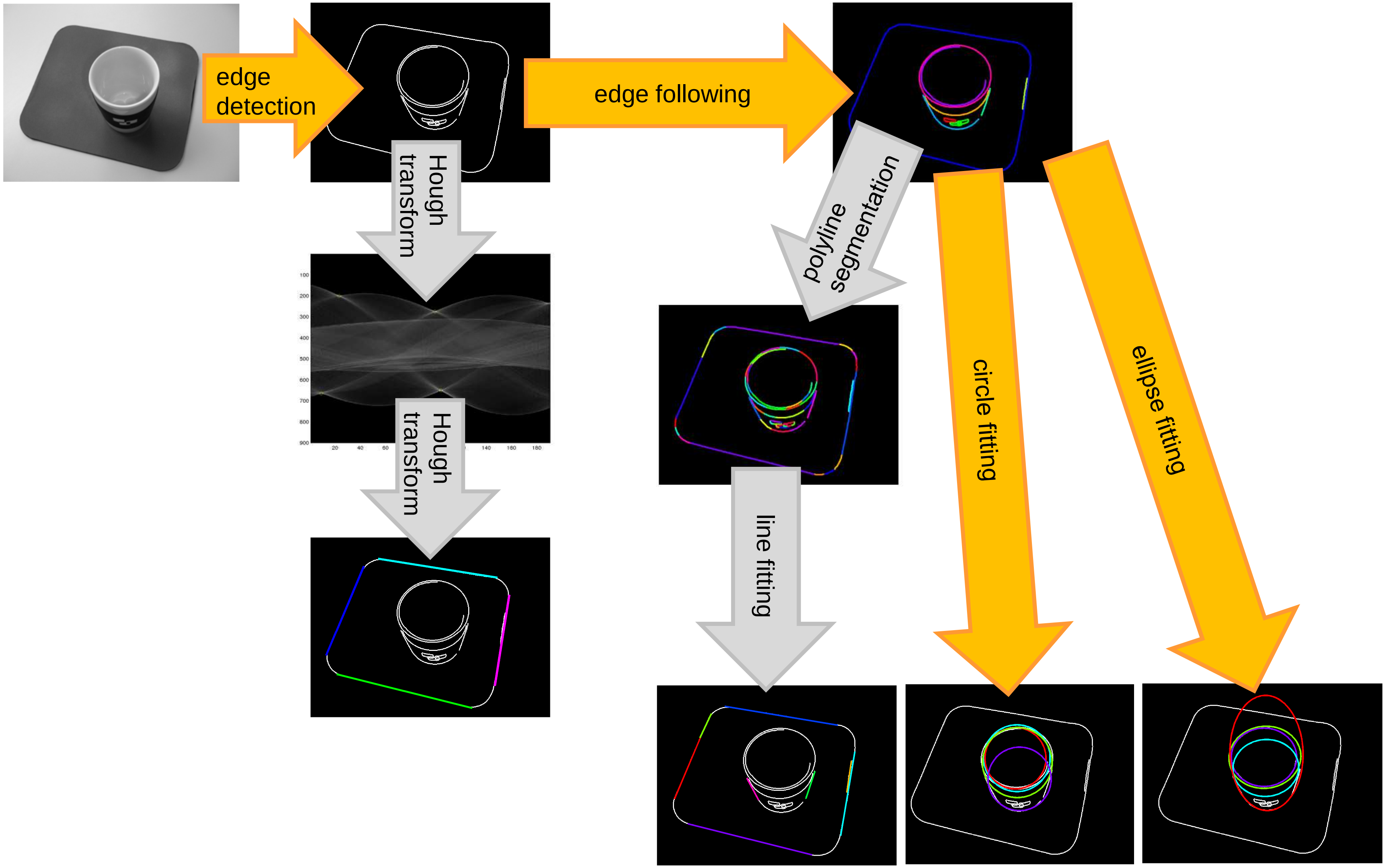


Robust Estimation

	M-estimator	LTS	RANSAC
• idea	reweight points according to their distance	ignore percentage of points with largest distances	ignore points with distance larger than a threshold
• parameters	error term, width parameter	acceptance rate	acceptance threshold
• algorithm	iterated weighted least squares	iterated least squares, several repetitions	repeated guesses from pairs of points

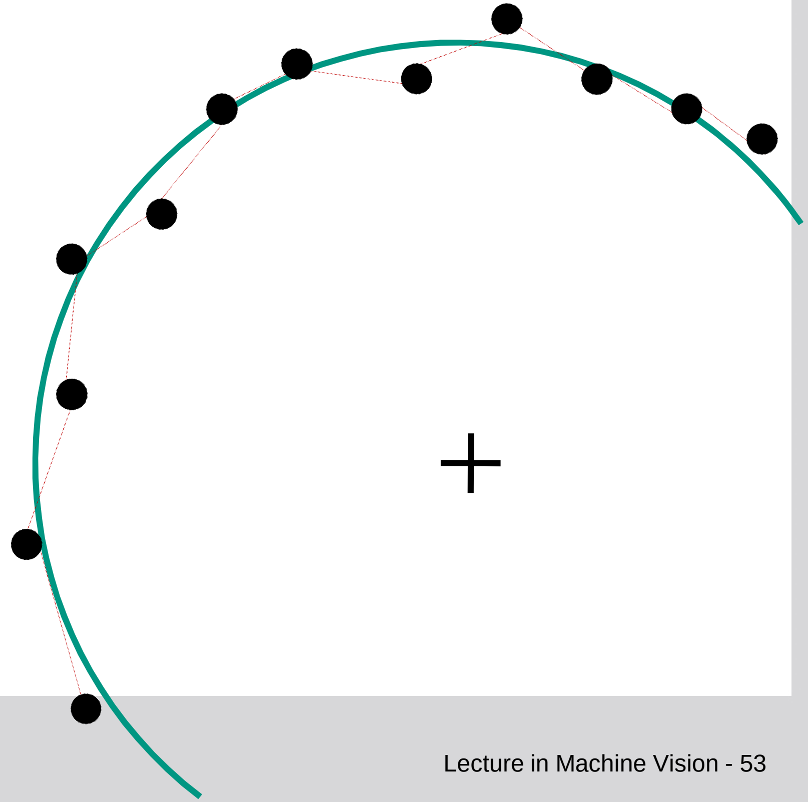
→ demo tool

Contours Detection



Estimating Circles and Ellipses

- determining parameters of circles/ ellipses that describe a curved contour from points



Estimating Circles

- parametric representation of circles:

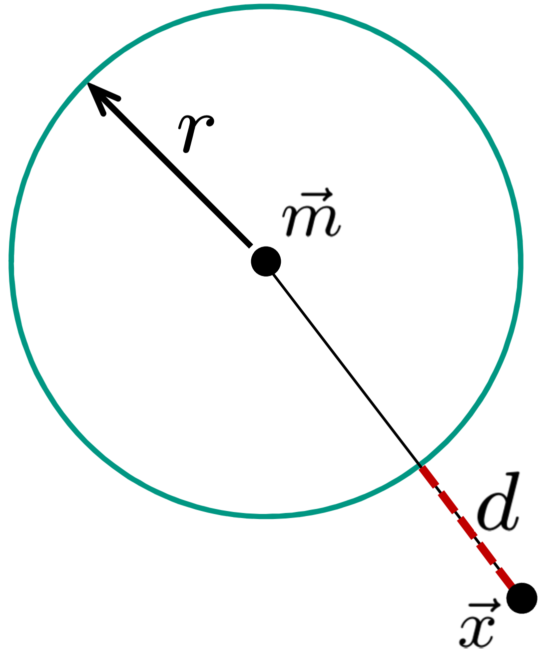
$$(x - m_1)^2 + (y - m_2)^2 - r^2 = 0$$

- Euclidean distance of point (x,y) from the circle:

$$d_E = \left| \sqrt{(x - m_1)^2 + (y - m_2)^2} - r \right|$$

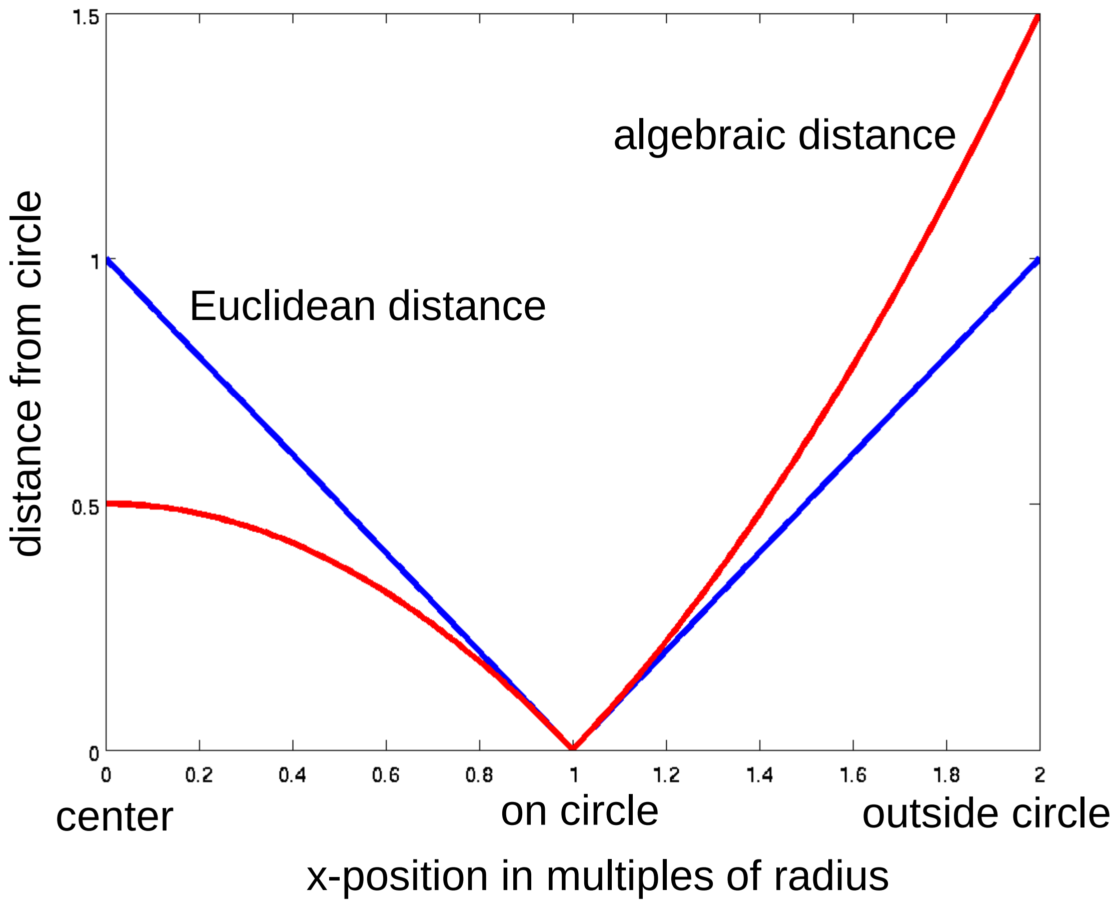
- algebraic distance:

$$d_A = \left| (x - m_1)^2 + (y - m_2)^2 - r^2 \right|$$



Estimating Circles cont.

- algebraic distance is asymmetric
- for points close to the circle both are similar



Estimating Circles cont.

- minimizing Euclidean distance:
  - cannot be solved analytically
- minimizing algebraic distance:
  - rewriting algebraic distance

$$\begin{aligned}(x-m_1)^2+(y-m_2)^2-r^2 &= (x^2+y^2)+(m_1^2+m_2^2-r^2)+(-2m_1)x+(-2m_2)y \\ &= Ax+By+C+(x^2+y^2)\end{aligned}$$

$$\text{with } A=-2m_1, B=-2m_2, C=m_1^2+m_2^2-r^2$$

  - minimizing

$$\sum_{i=1}^{N}(Ax_i+By_i+C+(x_i^2+y_i^2))^2$$

  - zeroing partial derivatives yields:

$$\begin{pmatrix}\sum_i x_i^2 & \sum_i x_i y_i & \sum_i x_i \\ \sum_i x_i y_i & \sum_i y_i^2 & \sum_i y_i \\ \sum_i x_i & \sum_i y_i & N\end{pmatrix}\begin{pmatrix}A\\B\\C\end{pmatrix}=\begin{pmatrix}-\sum_i x_i(x_i^2+y_i^2)\\ -\sum_i y_i(x_i^2+y_i^2)\\ -\sum_i(x_i^2+y_i^2)\end{pmatrix}$$

Estimating Circles cont.

– after having found A,B,C we get:

$$m_1 = -\frac{A}{2}$$
$$m_2 = -\frac{B}{2}$$
$$r^2 = m_1^2 + m_2^2 - C$$

Estimating Circles cont.

- minimising the Euclidean distance by iterative reweighting
  - optimal fit would be:

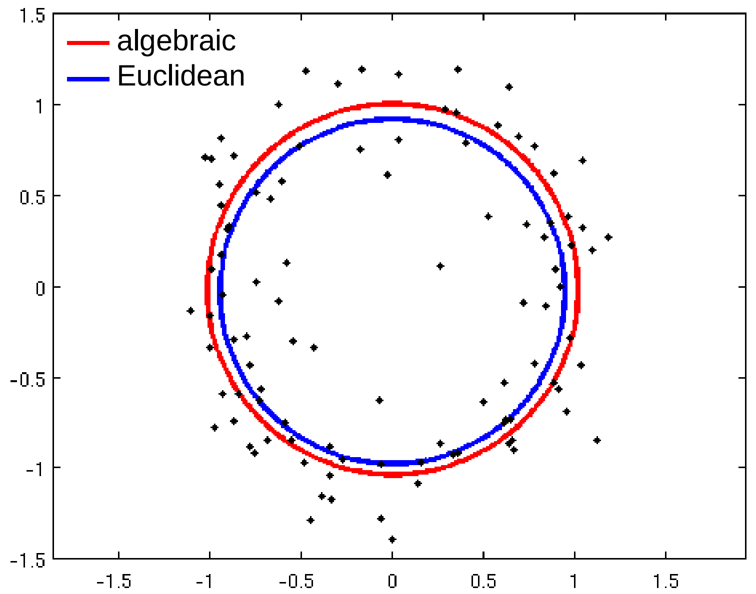
(1) $$\mathit{minimise} \sum_{i=1}^{N} (d_{E,i})^2$$

  - easily calculated:

(2) $$\mathit{minimise} \sum_{i=1}^{N} (d_{A,i})^2$$

  - weighted algebraic distance:

(3) $$\mathit{minimise} \sum_{i=1}^{N} w_i (d_{A,i})^2$$

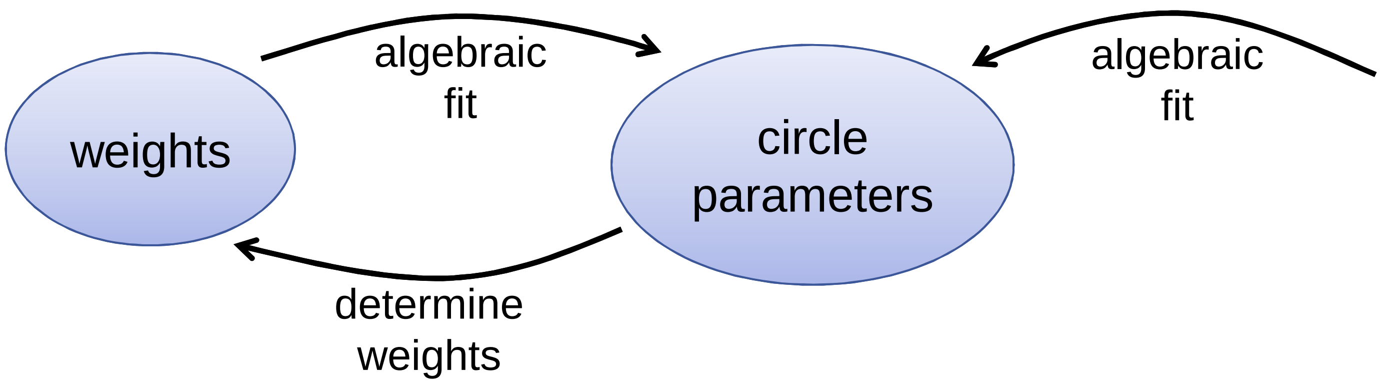


    - choose w_i so that (1) and (3) take their minimum at the same values
    - incrementally recalculate w_i and circle parameters

Estimating Circles cont.

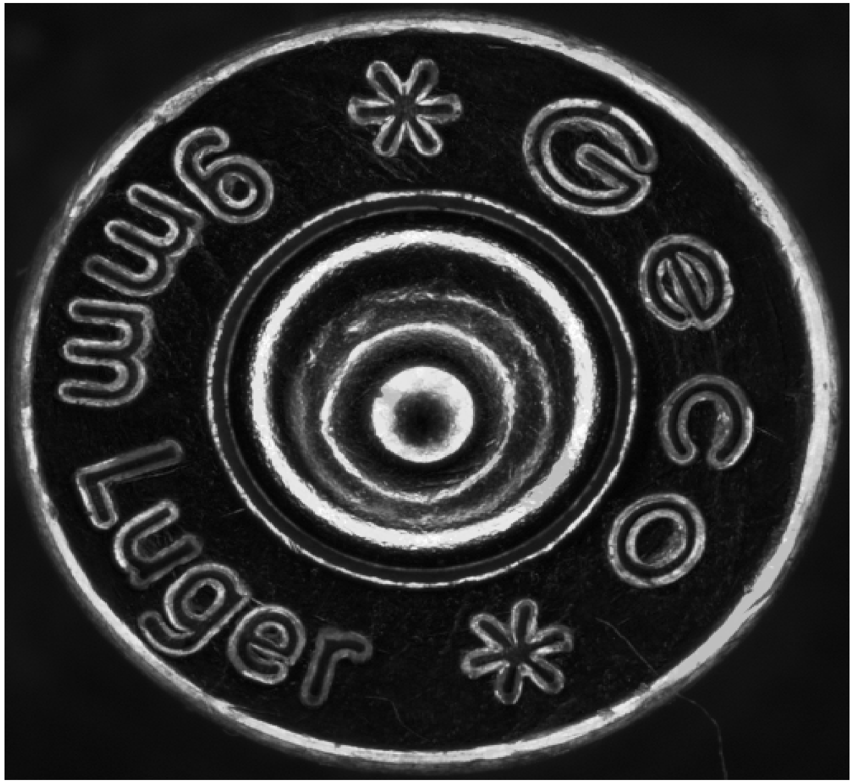
- recurrent dependency:
  - circle parameters depend on weights
  - weights depend on circle parameters

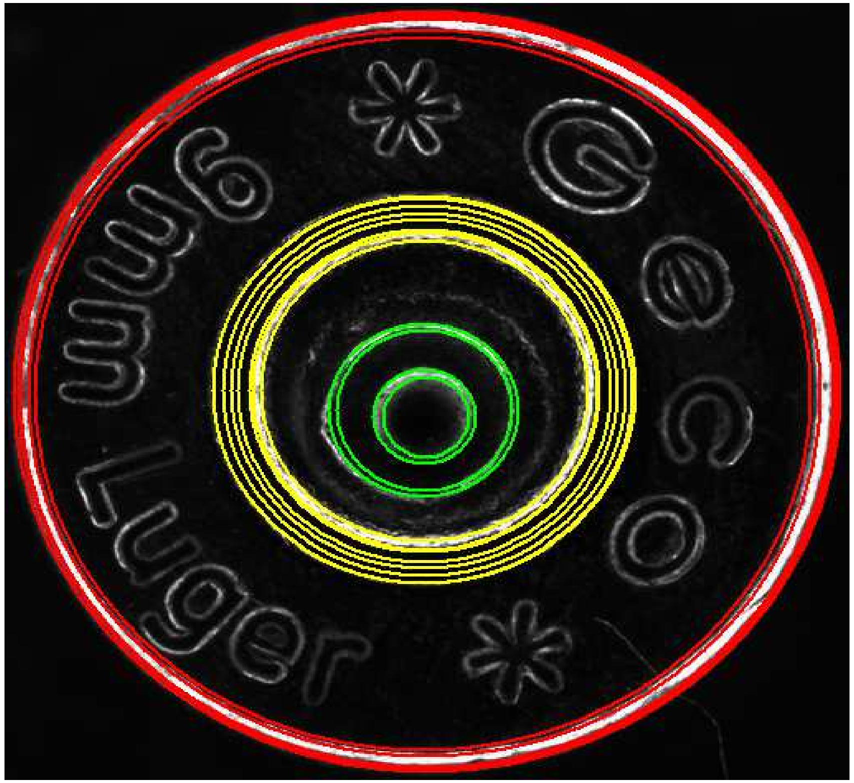
- iterative algorithm:



- estimation of circle parameters can be combined with robust techniques like M-estimators, LTS, RANSAC

Estimating Circles cont.



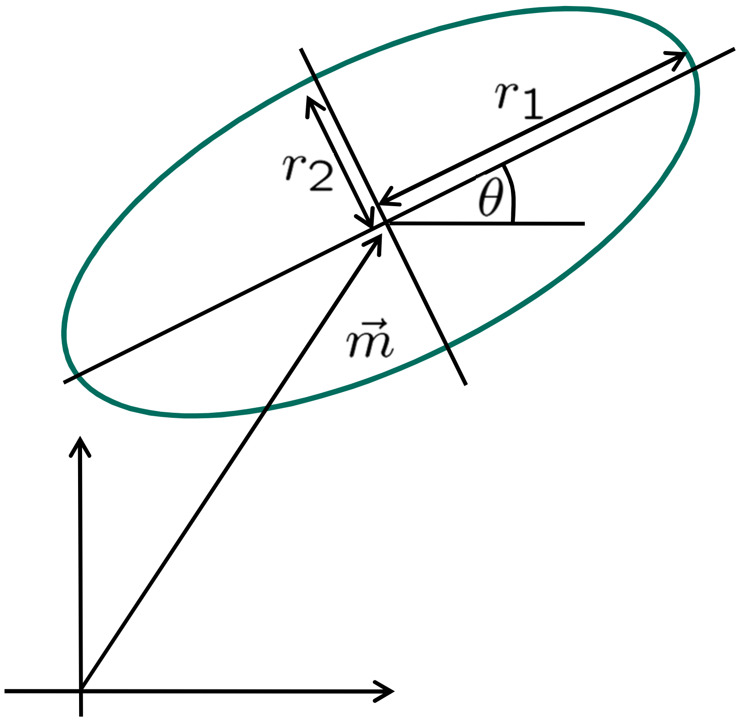


- example: estimating circles in the images of bullet casings
- techniques: randomized Hough transform + circle fitting with algebraic distance

  (work of Dr.-Ing. Christoph Speck, MRT)

Estimating Ellipses

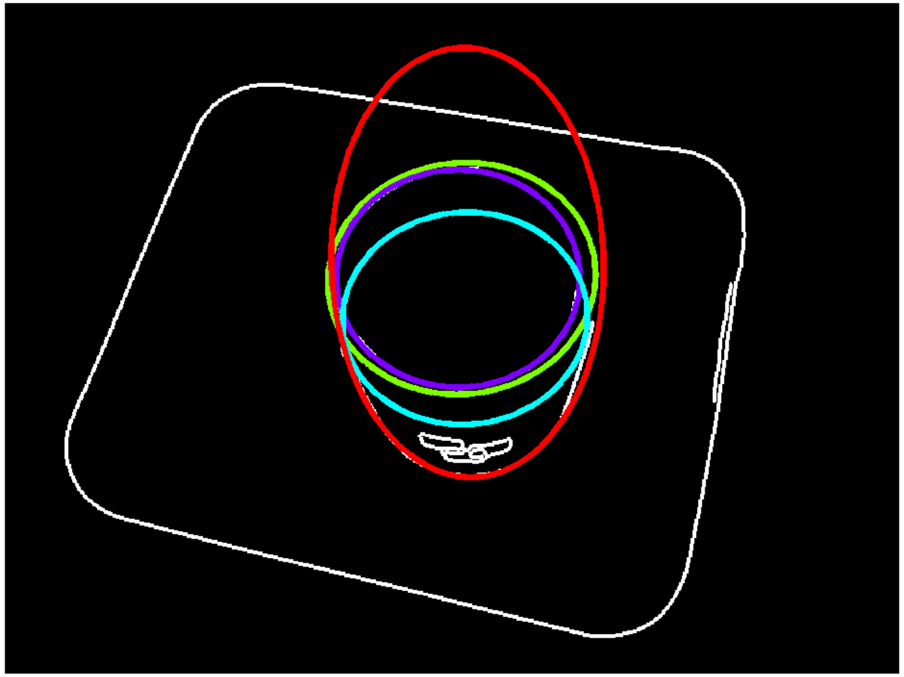
- ellipses:
  - extension (radius) r_1, r_2
  - center $\vec{m}$
  - turning angle θ
- parametric representation:
  $Ax^2 + Hxy + By^2 + Gx + Fy + C = 0$
  with $4AB - H^2 > 0$
- eliminating one degree of freedom:
- $A = 1$
- or $A + B = 1$
- or $A^2 + B^2 + C^2 + F^2 + G^2 + H^2 = 1$
- or $C = 1$ (not invariant to translation)

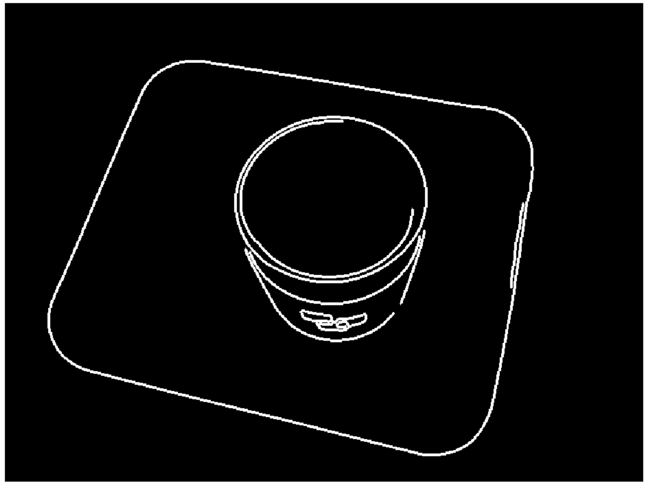


Estimating Ellipses

- approach of Fitzgibbon, Pilu, and Fisher (1999)
  - minimize squared algebraic distance
  - subject to constraint $4AB - H^2 = 1$

- yields a generalized Eigenvalue problem.
  Solution provides optimal ellipse parameters.

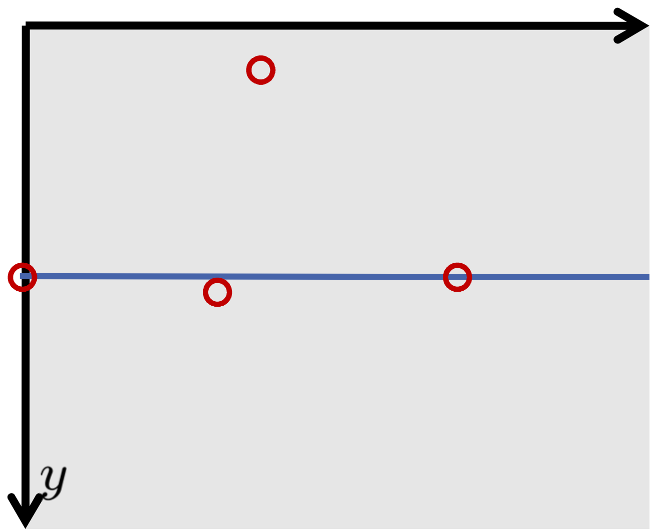


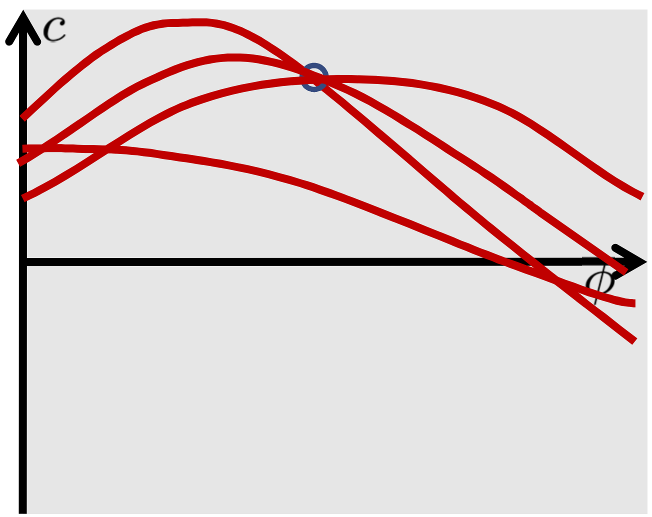


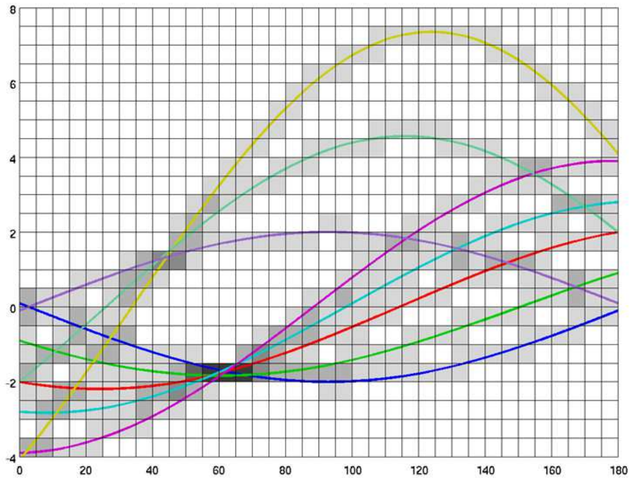
SUMMARY: CURVE FITTING

Summary cont.

- **Hough transform**
  - 2D geometry of lines
  - Hough transform
- **polyline segmentation**
- **robust line estimation**
- **circle and ellipse fitting**

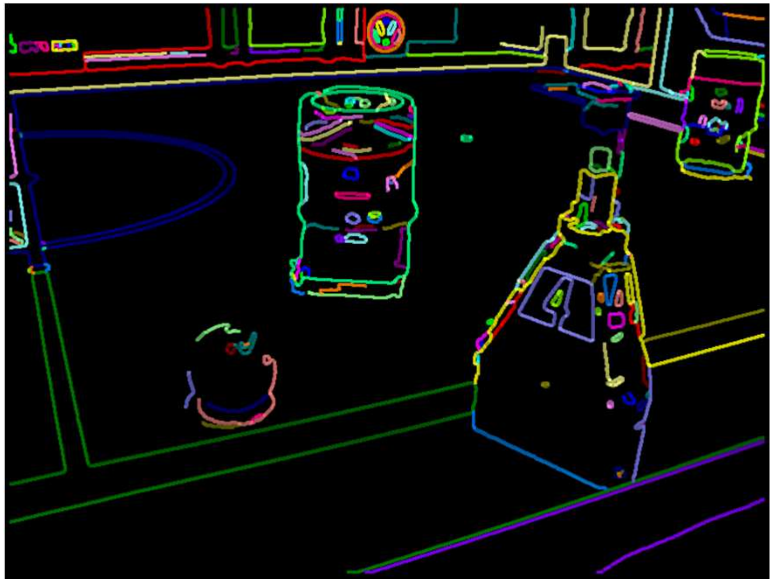


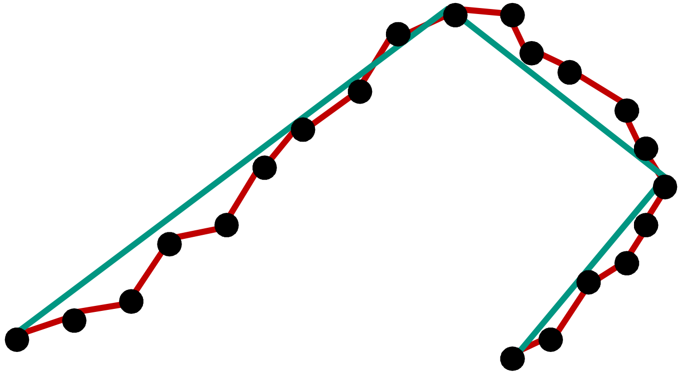




Summary cont.

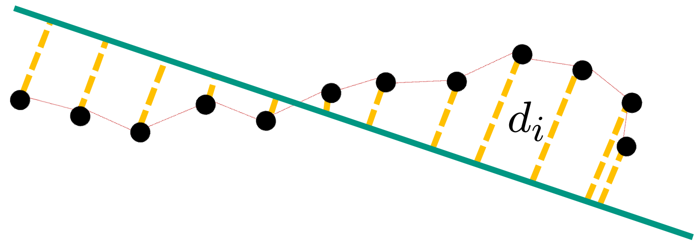
- **Hough transform**
- **polyline segmentation**
  - edge following
  - Ramer-Douglas-Peucker alg.
- **robust line estimation**
- **circle and ellipse fitting**

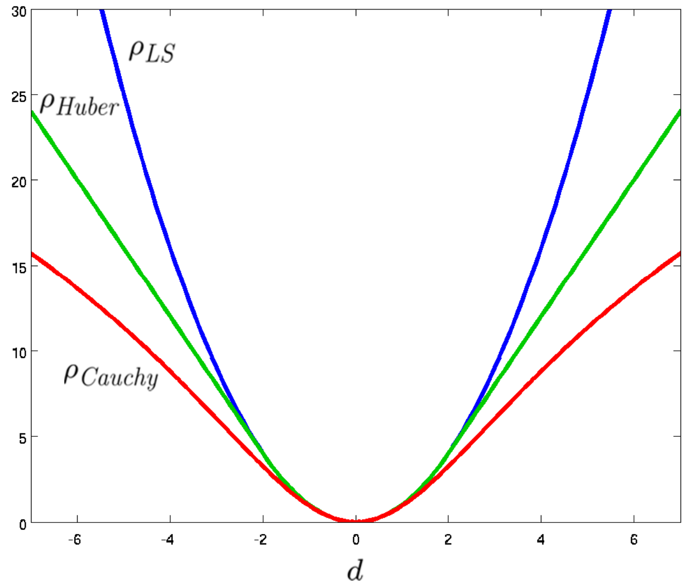




Summary cont.

- Hough transform
- polyline segmentation
- robust line estimation
  - total least squares
  - weighted least squares
  - M-estimators
  - LTS
  - RANSAC
- circle and ellipse fitting





Summary cont.

- **Hough transform**
- **polyline segmentation**
- **robust line estimation**
- **circle and ellipse fitting**
  - parametric representation
  - algebraic and Euclidean distance
  - iterative estimation