Machine Vision

Chapter 4: Curve Fitting

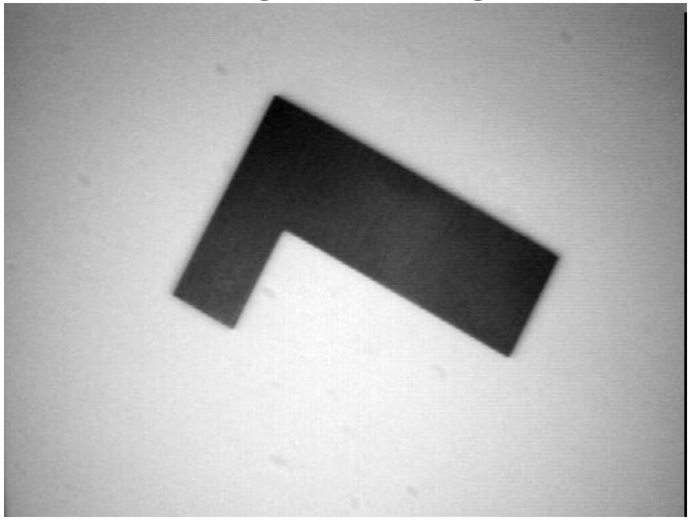
Dr. Martin Lauer

Institut für Mess- und Regelungstechnik

Karlsruhe Institute of Technology

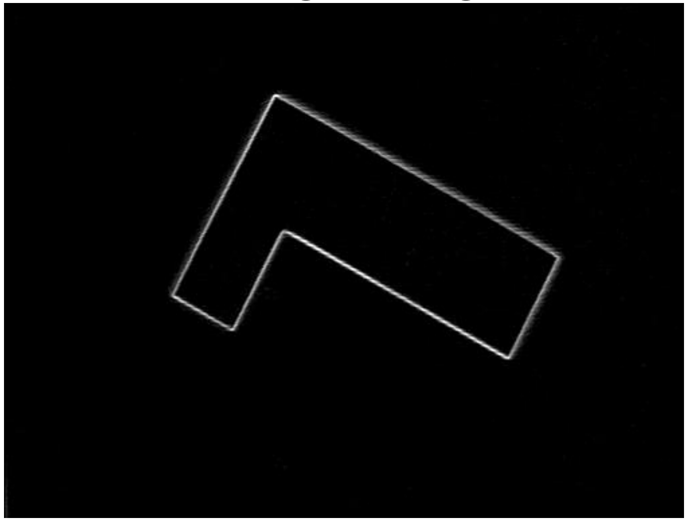
Contours

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:

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

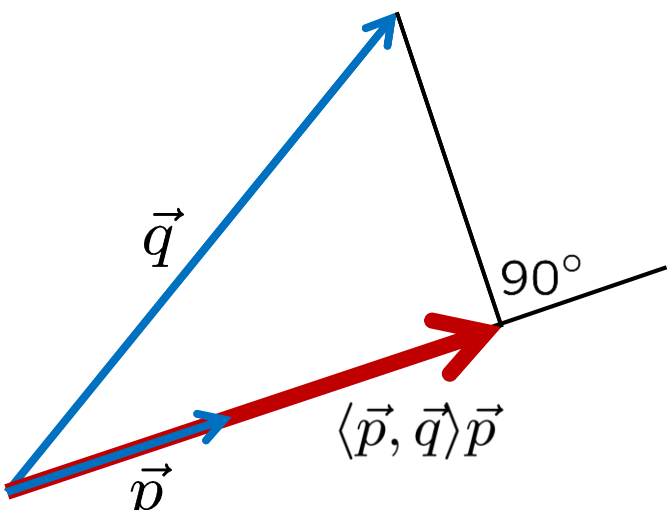
  - follows:

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

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

  - 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 $\vec{q}$:

    $$\vec{x} = (1-\tau)\vec{p} + \tau\vec{q}, \qquad \tau \in [0,1]$$

  - is part of line:

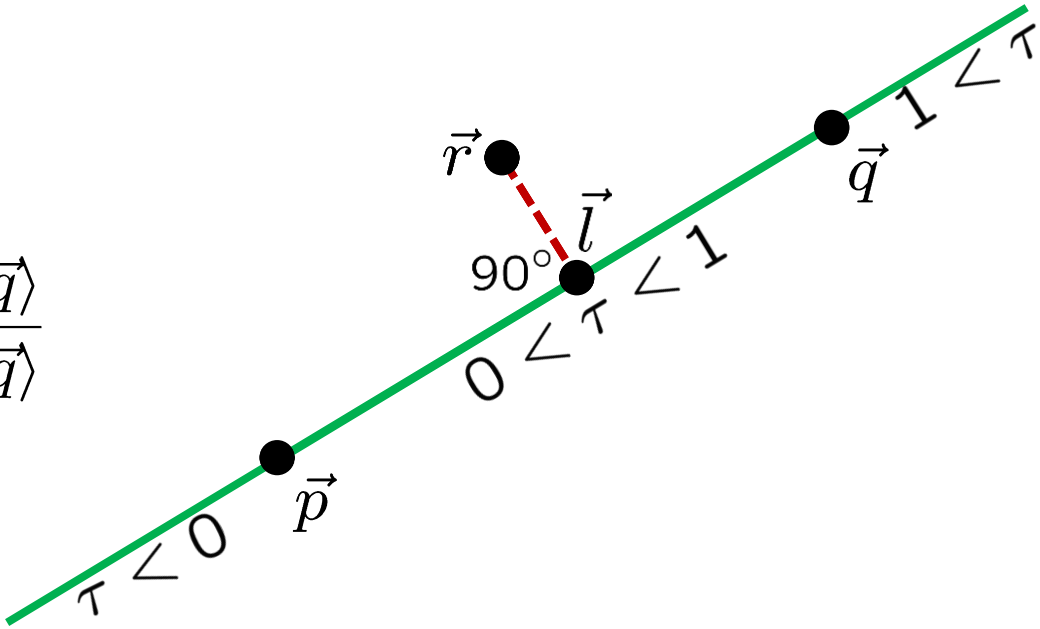
    $$\vec{x} = (1-\tau)\vec{p} + \tau\vec{q}, \qquad \tau \in \mathbb{R}$$

  - perpendicular point:

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

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

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



2D Geometry cont.

- line in normal form:

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

  $$\langle \vec{n}, \vec{x} \rangle = \langle \vec{n}, (1-\tau)\vec{p} + \tau\vec{q} \rangle = \langle \vec{n}, \vec{p} \rangle + \tau \langle \vec{n}, \vec{q} - \vec{p} \rangle = \langle \vec{n}, \vec{p} \rangle$$

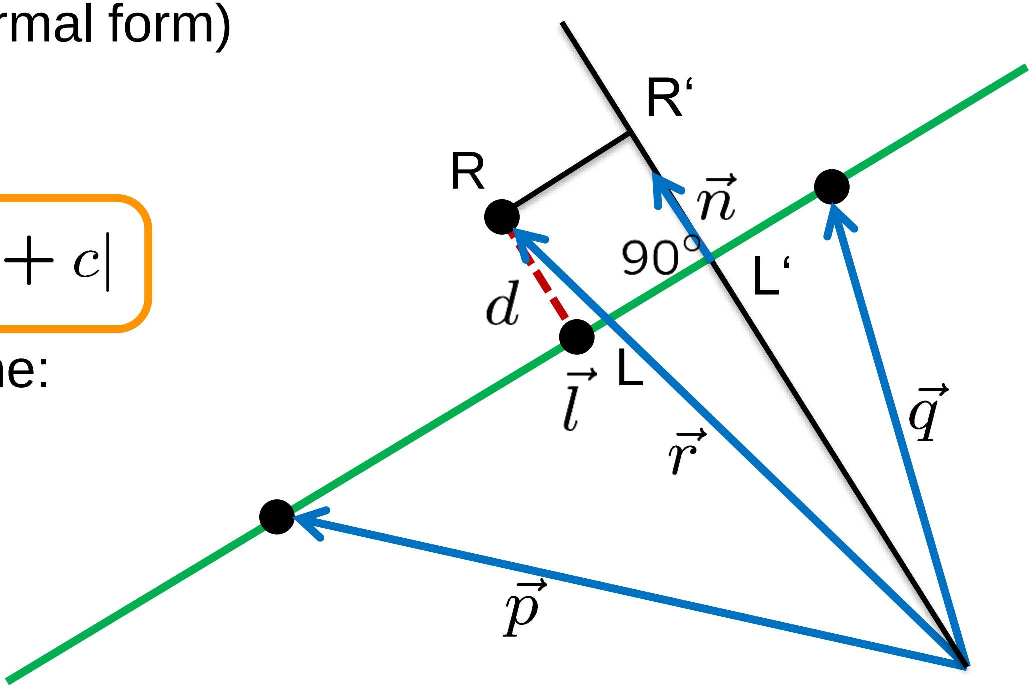
  $$\begin{aligned} 0 &= \langle \vec{n}, \vec{x} \rangle - \langle \vec{n}, \vec{p} \rangle \\ &= \langle \vec{n}, \vec{x} \rangle + c \quad \text{(normal form)} \end{aligned}$$

- distance of point from line:

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

- an (arbitrary) point on the line:

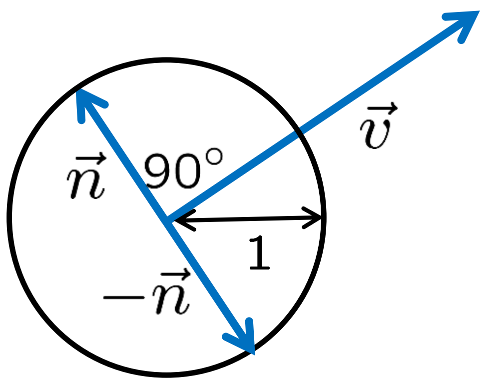
  $$\boxed{-c\vec{n}}$$



2D Geometry cont.

– unit normal vector:

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



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

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

– every vector has a polar representation as:

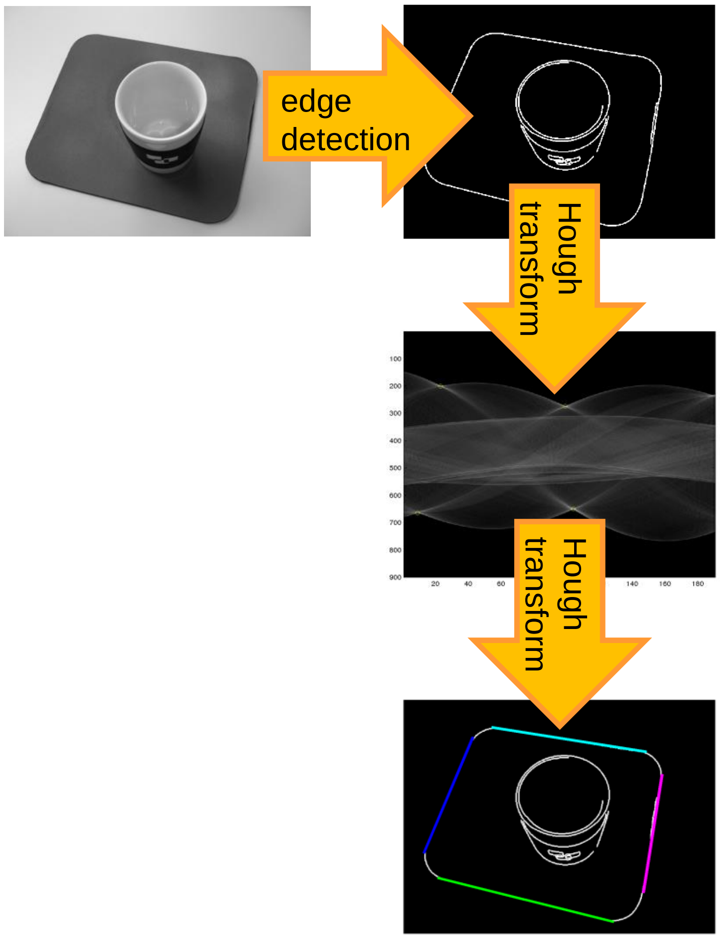
$$\vec{v} = r \cdot \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} \qquad \text{with } r \geq 0, \phi \in [0, 2\pi)$$

– determining the angle ϕ of a 2d-vector:

$$\phi = atan_2(v_1, v_2)$$

atan2为atan函数的拓展，不过不用关心

Contours Detection



Hough Transform 的核心思想是将图像空间中的点转换到参数空间中，
从而将检测特定形状的任务转化为寻找参数空间中的峰值问题。
例如，对于检测直线，Hough Transform 会将每个边缘点映射到表示所有可能经过该点的直线的参数空间。

Hough Transform

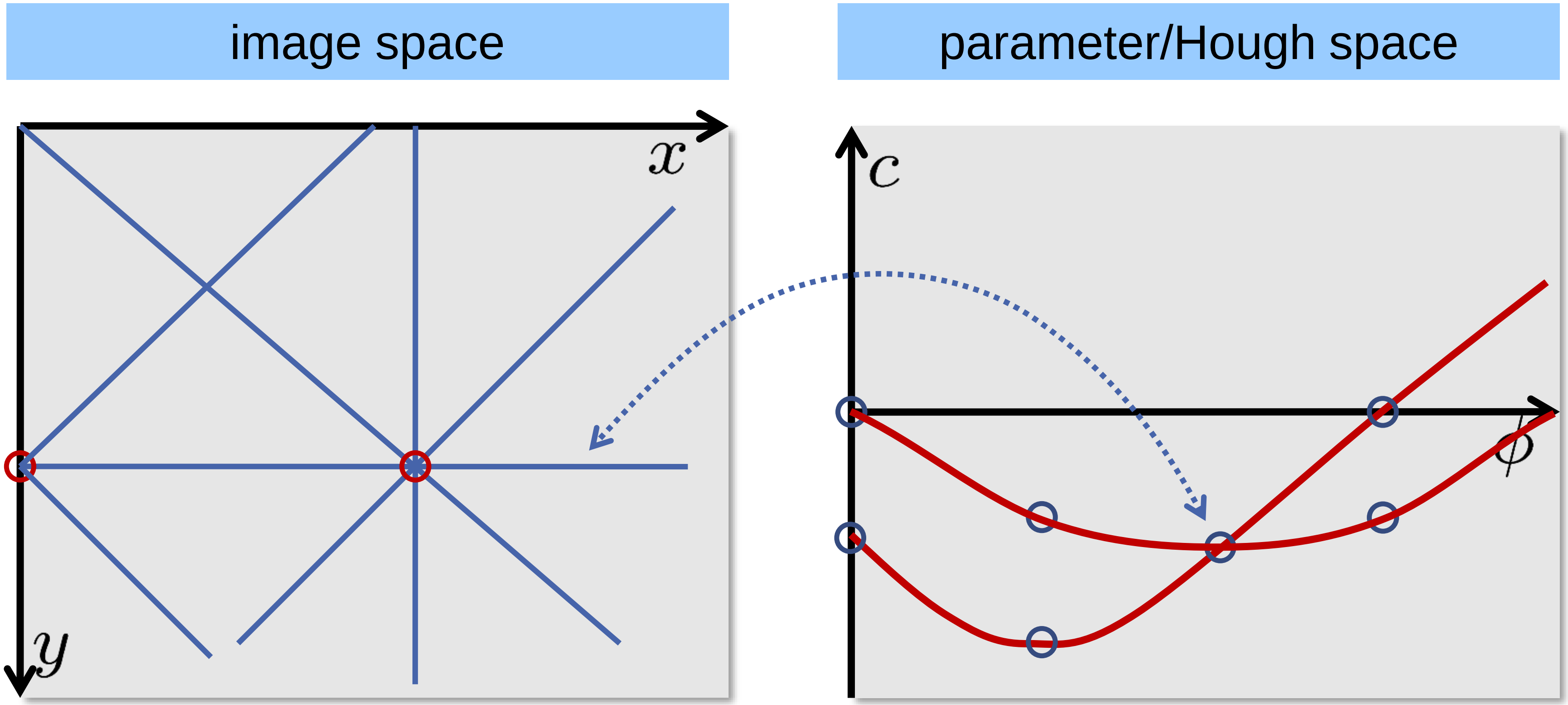
- find lines in edge bitmaps
  - idea: every line can be represented in 2D as:

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

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

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

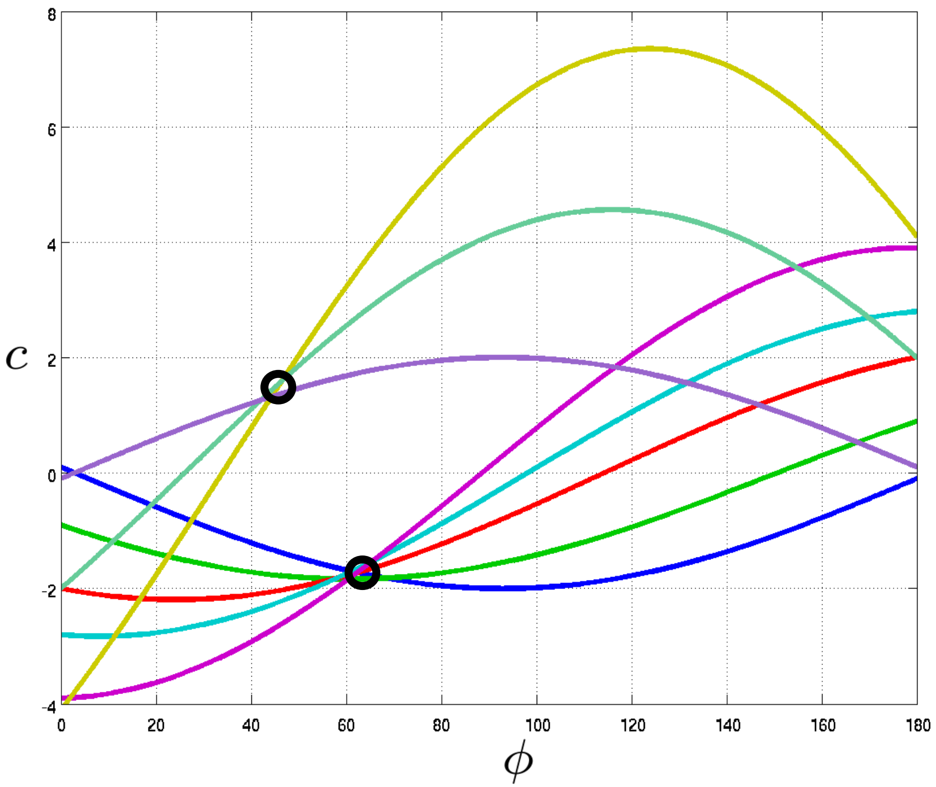
Hough Transform cont.



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

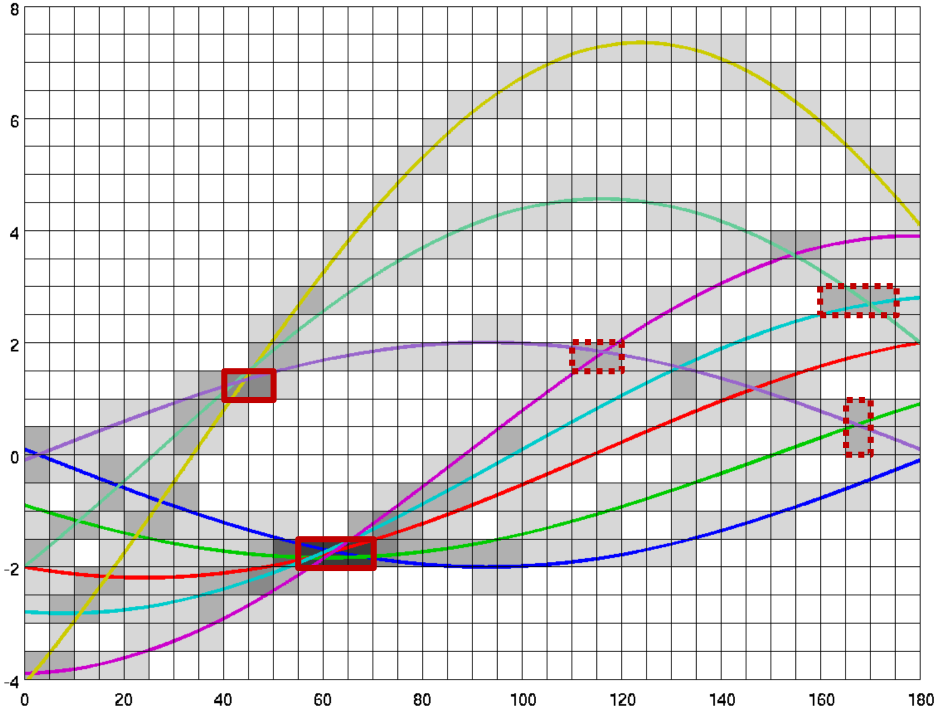
Hough Transform cont.

- 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



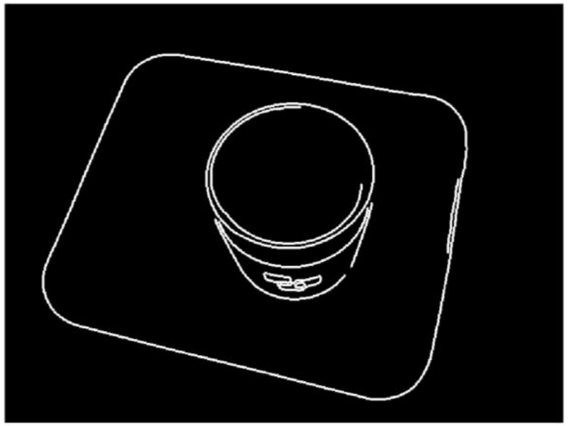
Hough Transform cont.

- 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*

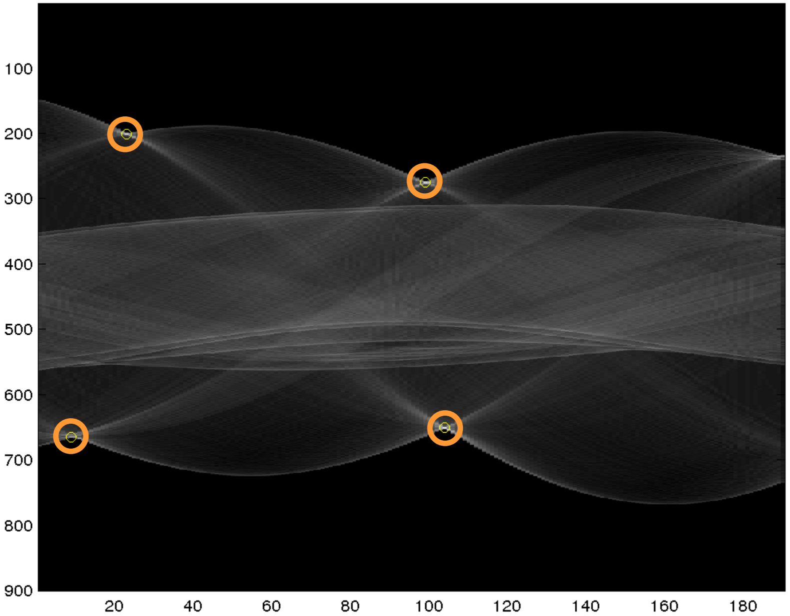
- after having found line parameters the edge pixels with small distance to the lines can be assigned to the line
  - determine starting point and end point of line
  - allow gaps of maximal size

Hough Transform cont.



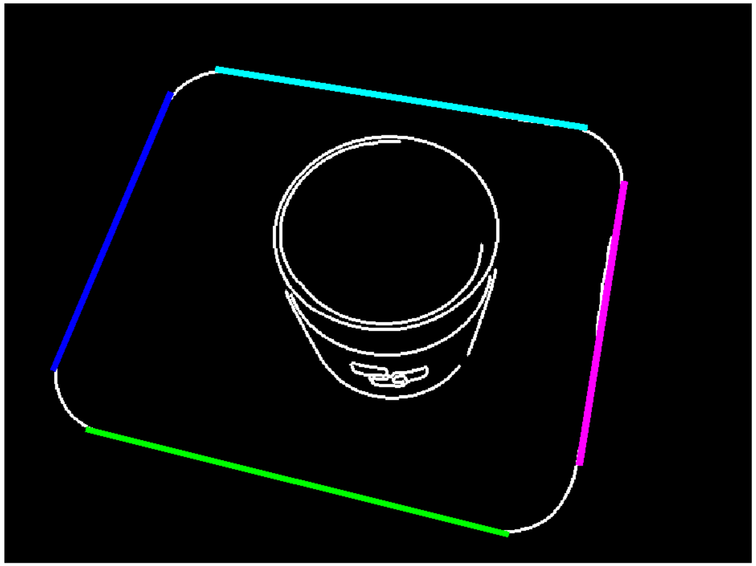
1. edge bitmap (with Canny)

2. Hough parameter space



3. Find local maxima

4. Deter-mine lines belonging to local maxima



Hough Transform cont.

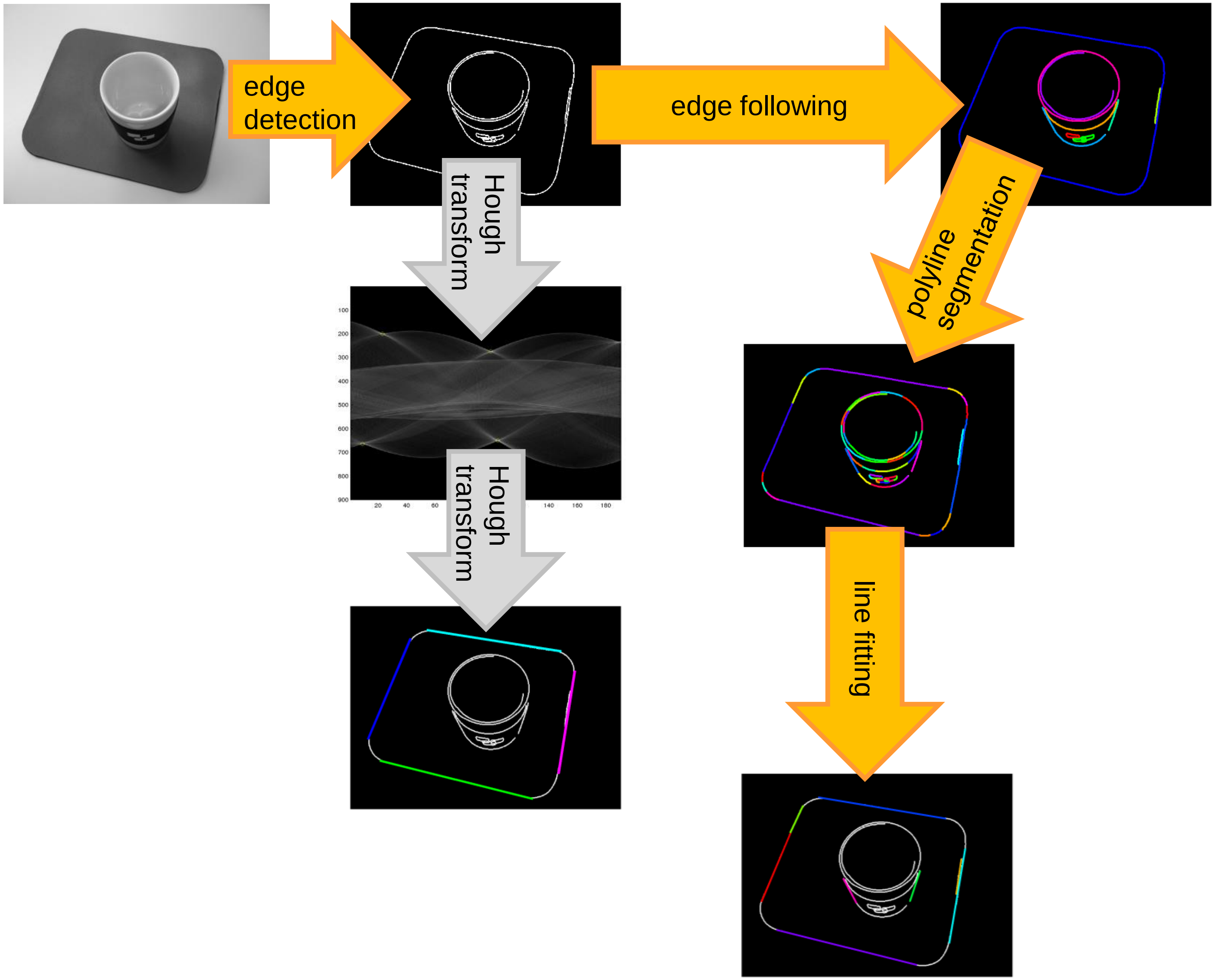
- properties of Hough transform: 结果依赖于累加器数组的大小和精度：
  - result depends on size and precision of accumulator array
  - 在累加器数组中确定显著峰值可能困难
  - determining significant peaks in the accumulator array might be difficult in practice
  - gradient direction is ignored 忽略梯度方向
  - accumulator array is flooded in “natural” scenes

  在“自然”场景中累加器数组容易被填满(noise)

- extensions: 扩展到其他参数化曲线
  - extension to other kind of parameterized curves (circles, ellipses, …)
  - randomized Hough transform 随机化 Hough Transform （Randomized Hough Transform）
  - generalized Hough transform 广义 Hough Transform （Generalized Hough Transform）

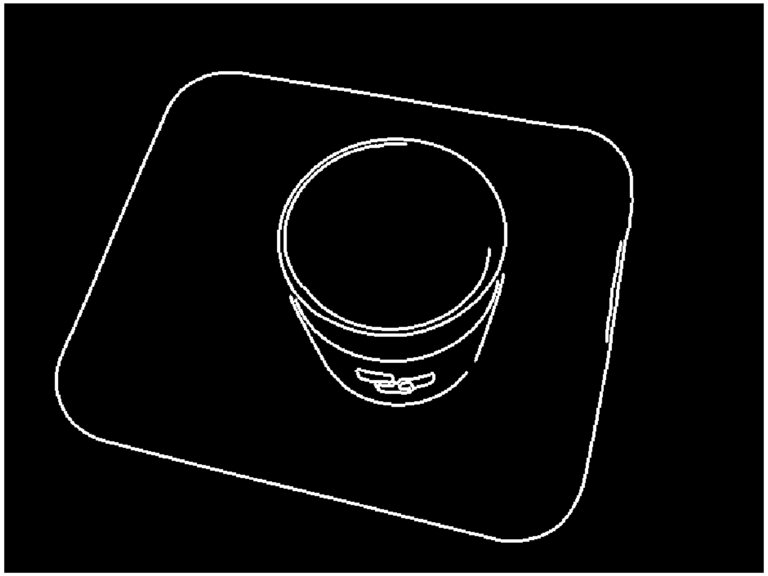
都是拓展Hough_Transform，使其可以检测更多类型的线（曲线，环形之类的）

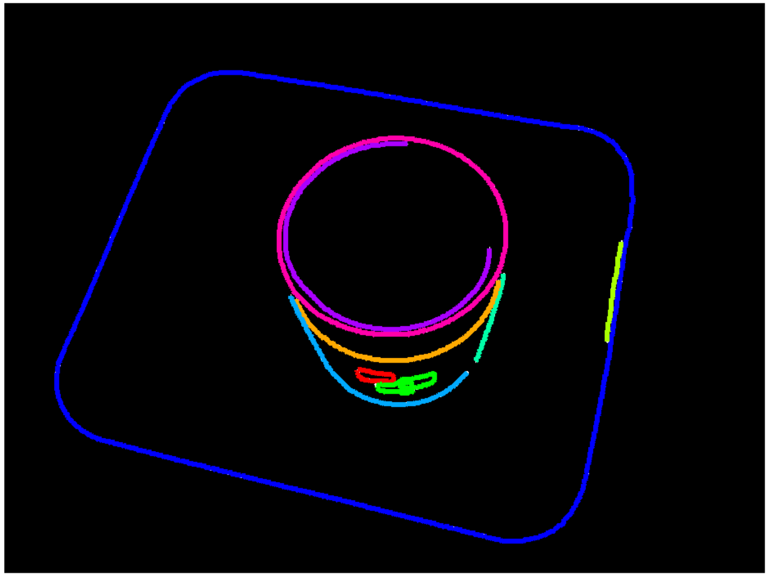
Contours Detection



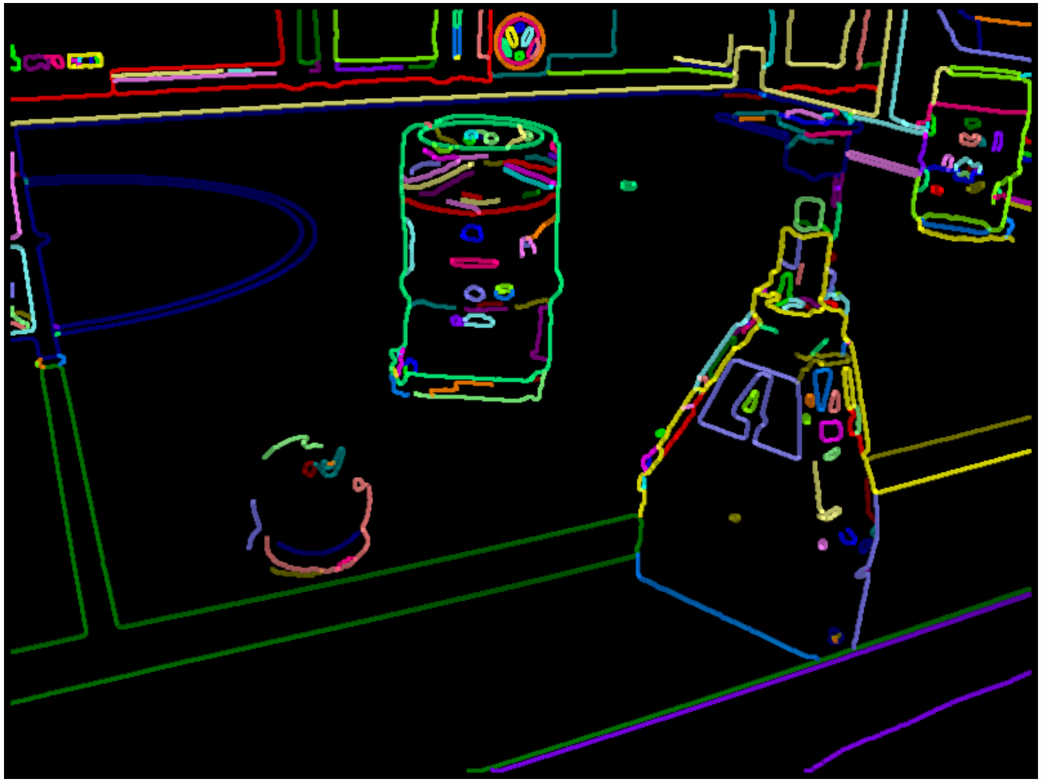
Edge Following

- 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



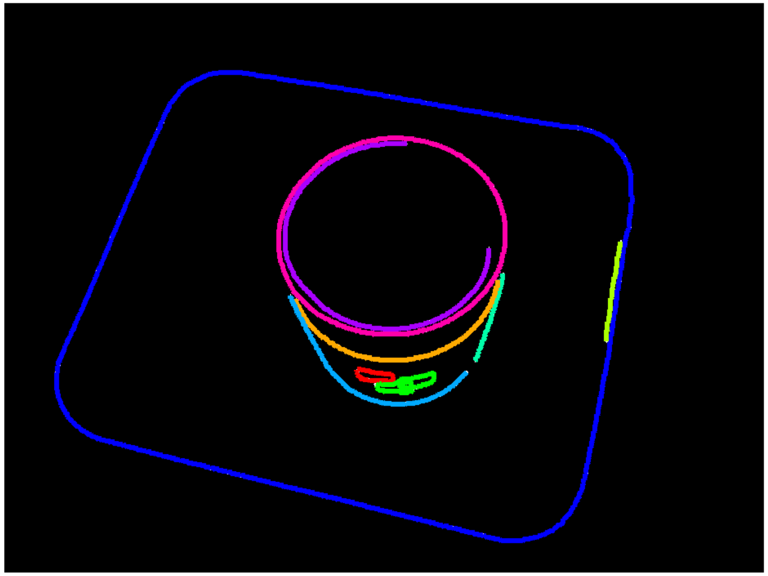


Edge Following cont.



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

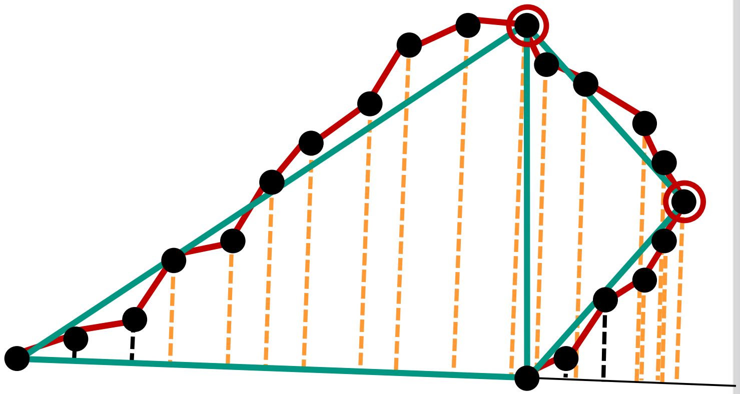


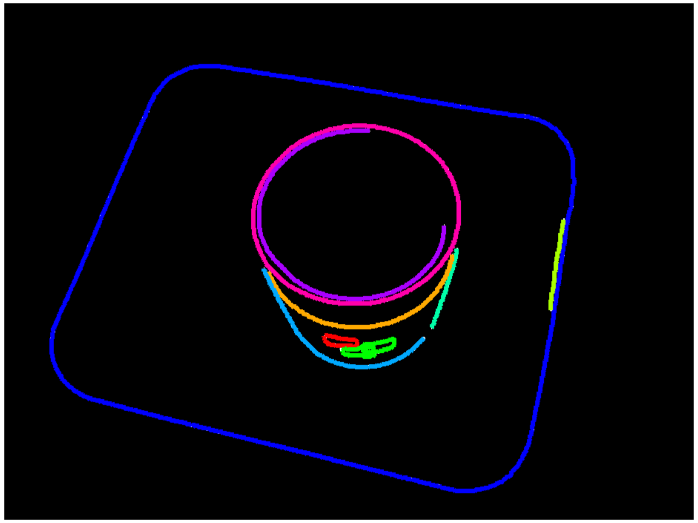
Ramer-Douglas-Peucker Algorithm

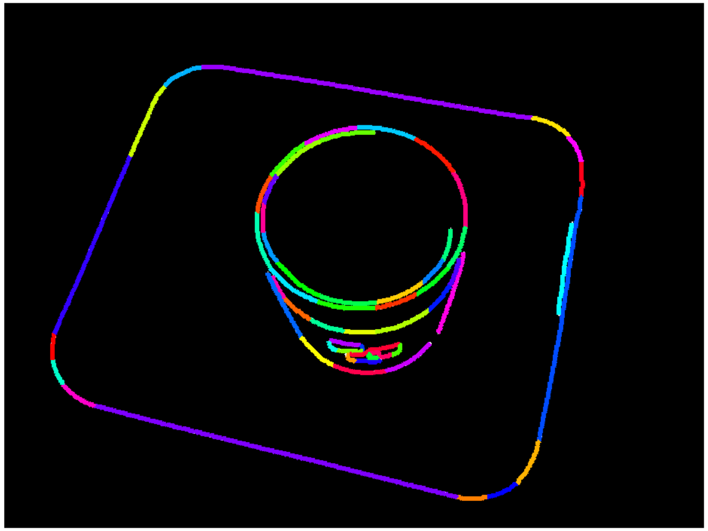
一种曲线拟合的简单方法（重复到没有超过阈值的长度）

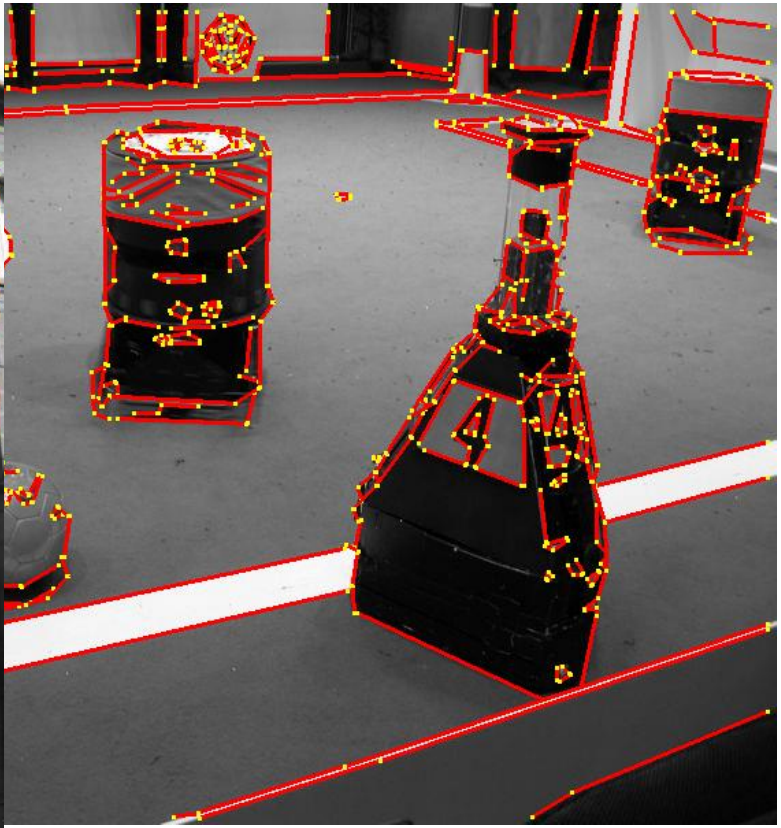
– basic idea: subdivide polyline recursively at the farthest vertex

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

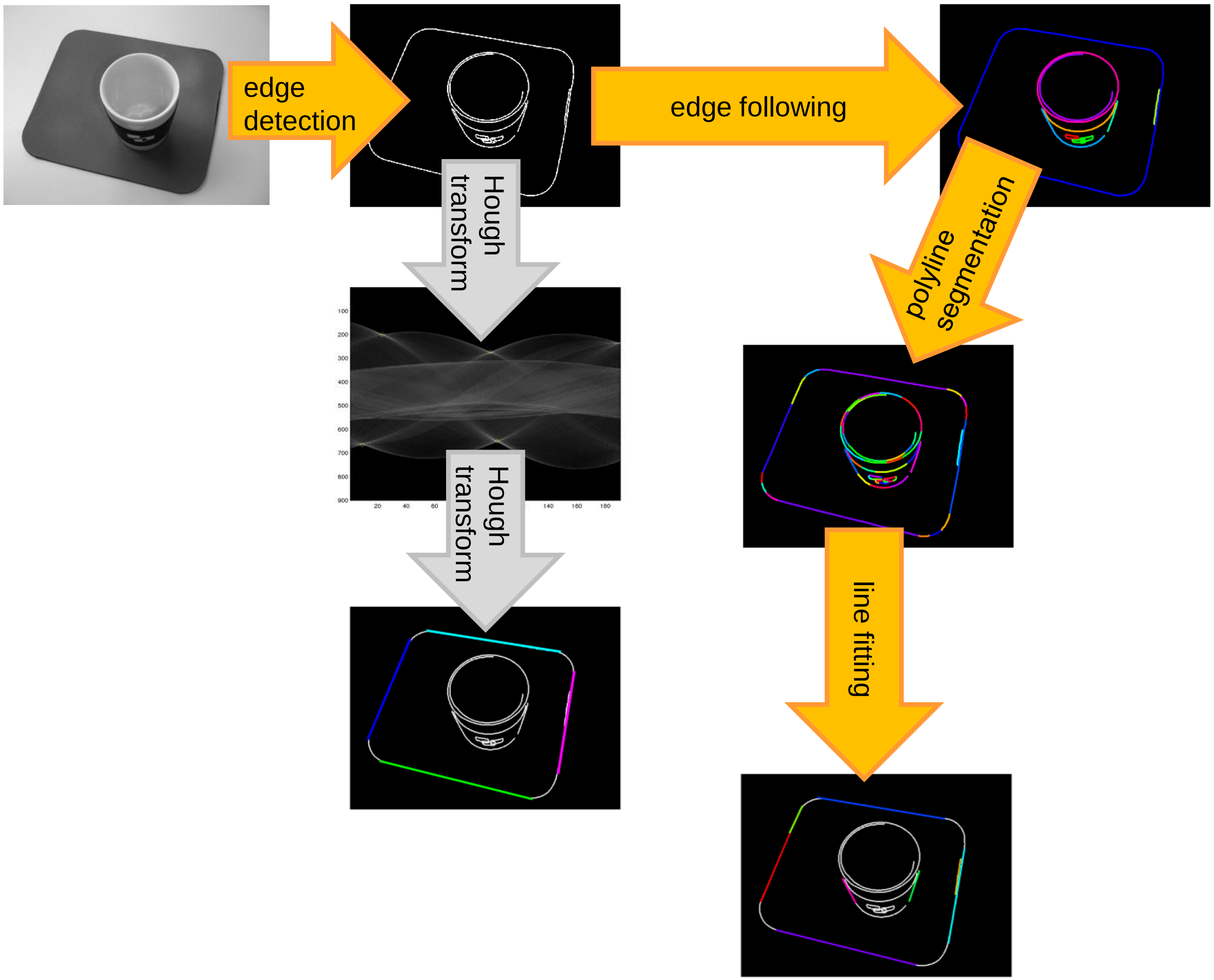








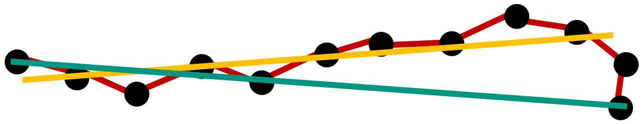
Contours Detection



LINE FITTING

Line Estimation

- result of polyline segmentation is suboptimal
- result of Hough transform may be suboptimal



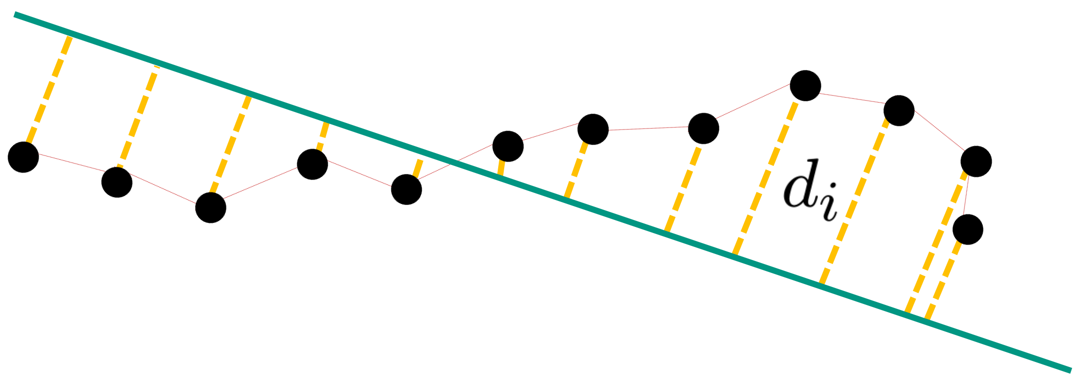
→ algorithms for accurate line estimation

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 $d_1, d_2, \dots, d_N$



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:

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

Total least squares cont.

– zeroing partial derivative w.r.t. n_1, n_2 :

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

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

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

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

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

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: $\lambda_1 \geq \lambda_2 \geq 0$

  $\rightarrow \lambda_2$ minimises distances
  $\rightarrow \lambda_1$ maximises distances

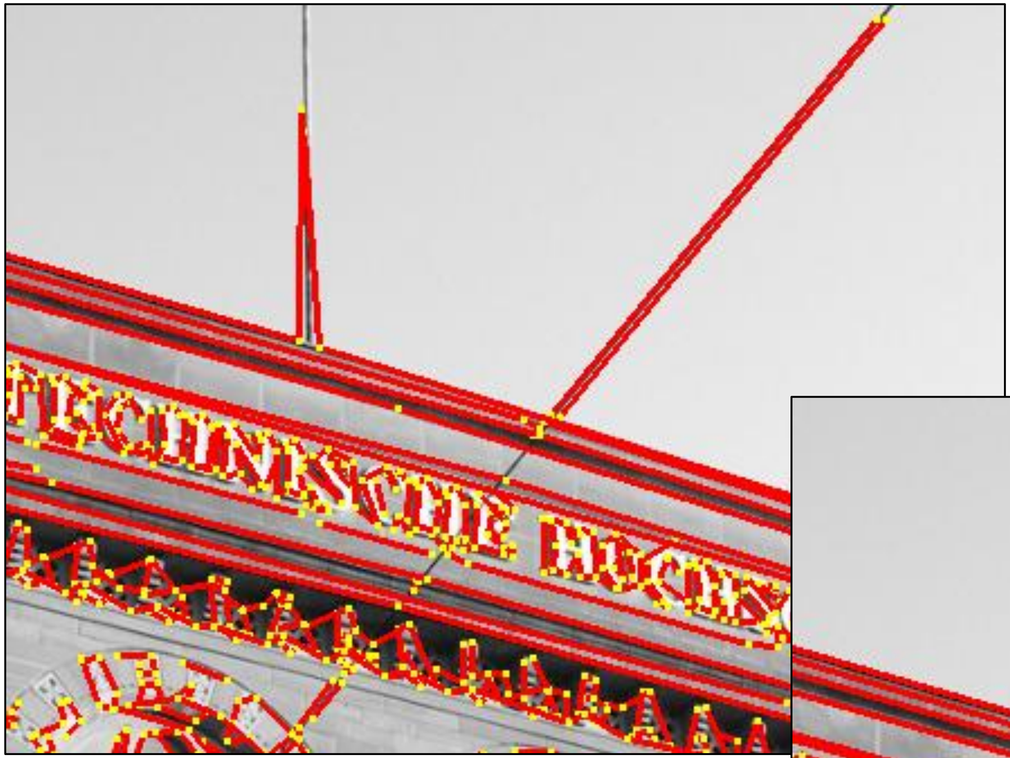
Total least squares cont.

- 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 $\begin{pmatrix}\alpha & \beta \\ \beta & \gamma\end{pmatrix}$
     $\rightarrow \vec{n}, \lambda \quad$ (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.

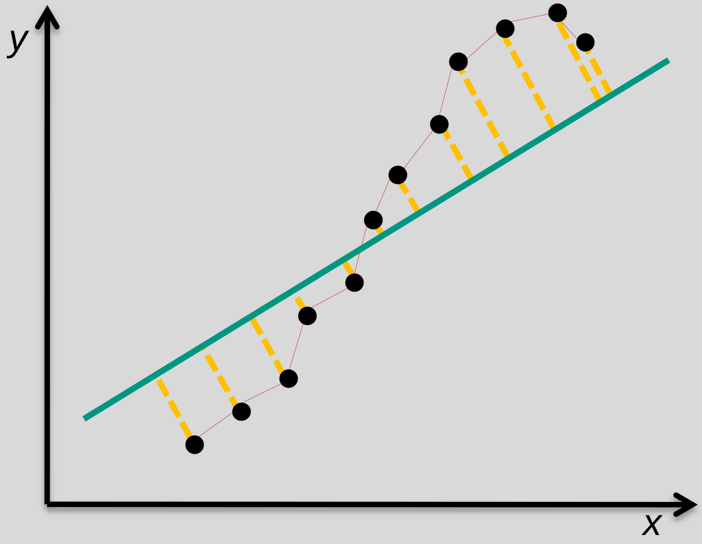


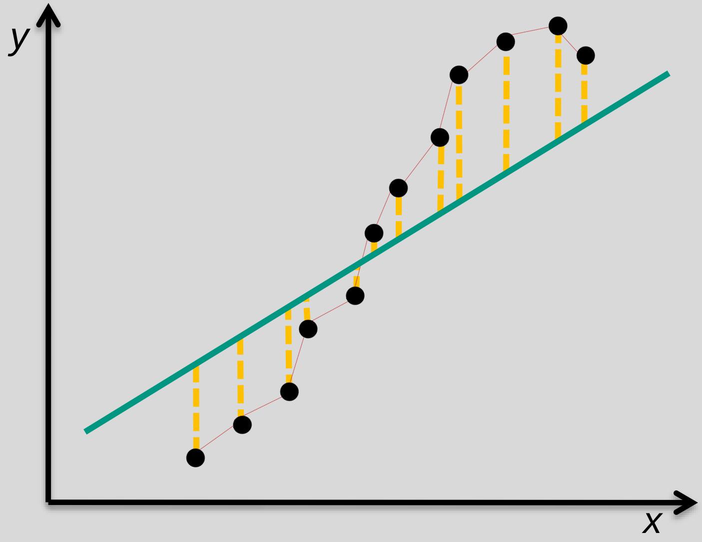
after line splitting

after total least squares

Total least squares cont.

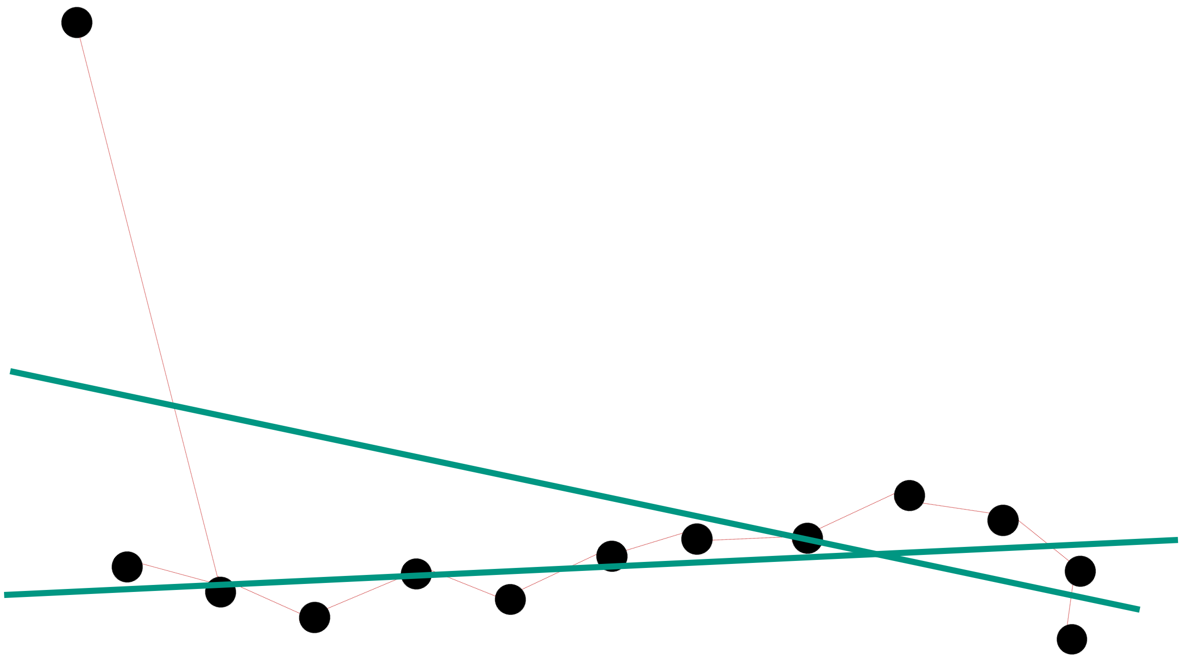
total least squares	ordinary least squares
• treat *x* and *y* alike • isotropic • minimize orthogonal distances	• interpret *y* = *y(x)* • anisotropic 不考虑x方向的误差 • minimize distances in y only





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

    →M-estimators, …
  - ignore outliers

    →RANSAC, …

M-Estimators

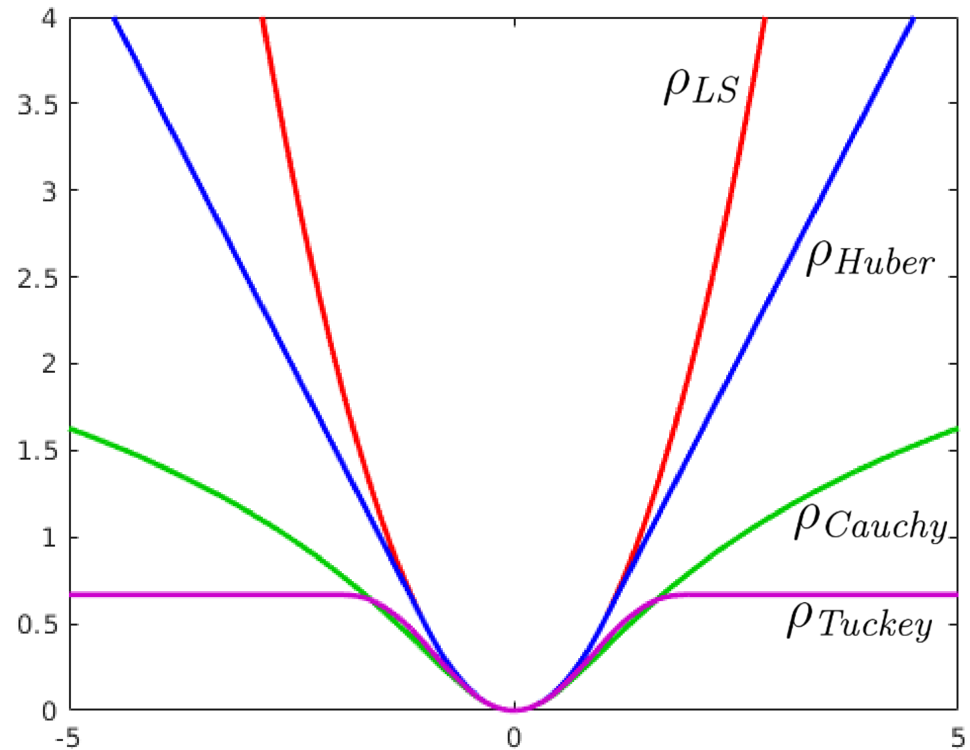
区别在Lossfunction， 目标是给定不同阶数的Lossfunction,使不同的Loss最小

Total least squares:

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

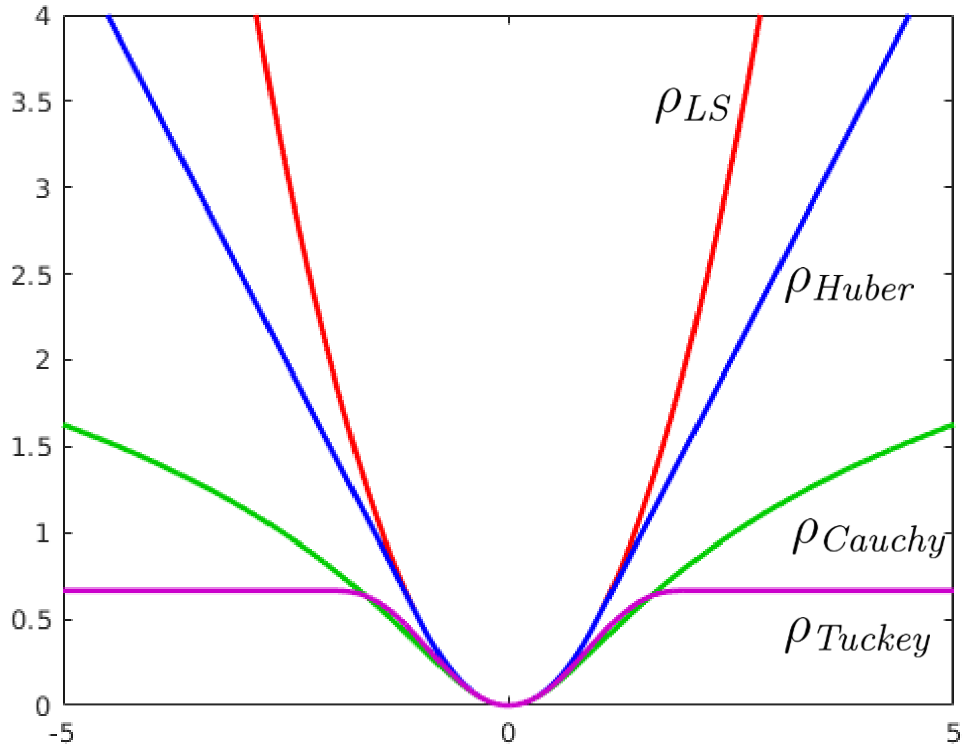
M-estimators:

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



Choose ρ such that outliers have lower impact

M-Estimators



$$\rho_{LS} \; : \; d \mapsto \frac{1}{2} d^2$$

$$\rho_{Cauchy} \; : \; d \mapsto \frac{\kappa^2}{2} \log\left(1 + \frac{d^2}{\kappa^2}\right)$$

$$\rho_{Huber} \; : \; d \mapsto \begin{cases} \frac{1}{2} d^2 & \text{if } |d| \leq k \\ k|d| - \frac{1}{2}k^2 & \text{otherwise} \end{cases}$$

linear function(线性部分（Idl > k）**降低了异常值的影响)

$$\rho_{Tuckey} \; : \; d \mapsto \begin{cases} \frac{a^2}{6}\left(1 - \left(1 - \frac{d^2}{a^2}\right)^3\right) & \text{if } |d| \leq a \\ \frac{a^2}{6} & \text{otherwise} \end{cases}$$

线性部分（Idl > k）**降低了异常值的影响

RANSAC

不是穷举，有一个上限的

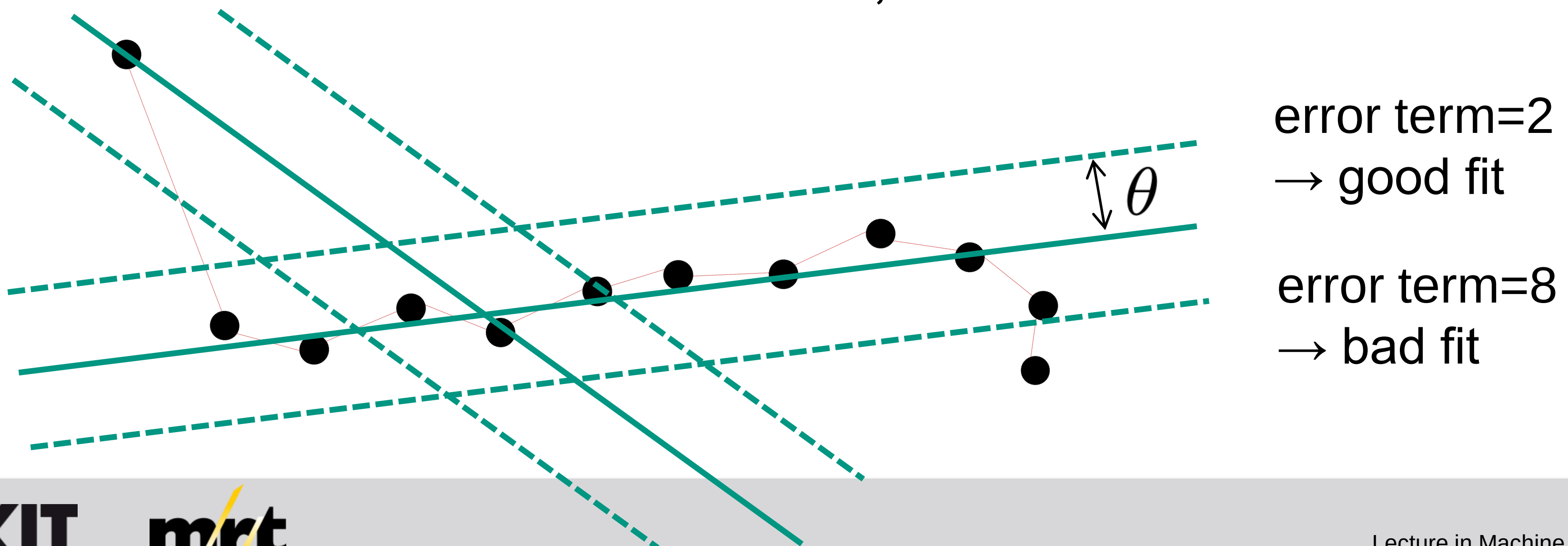
- idea: search a line that passes nearby as many points as possible

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

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

间断的

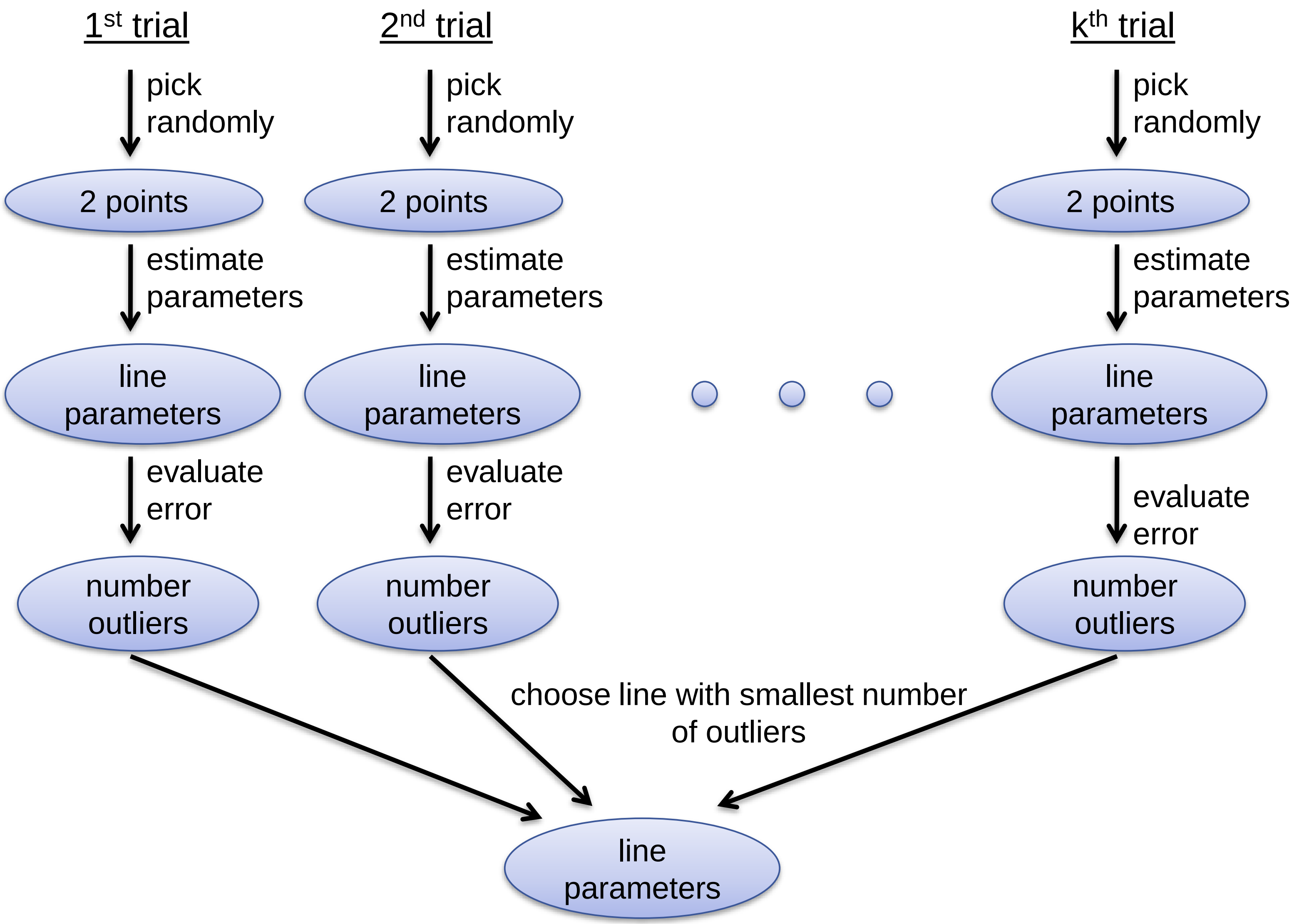
- definition similar to M-estimator, but σ is discontinuous



RANSAC cont.

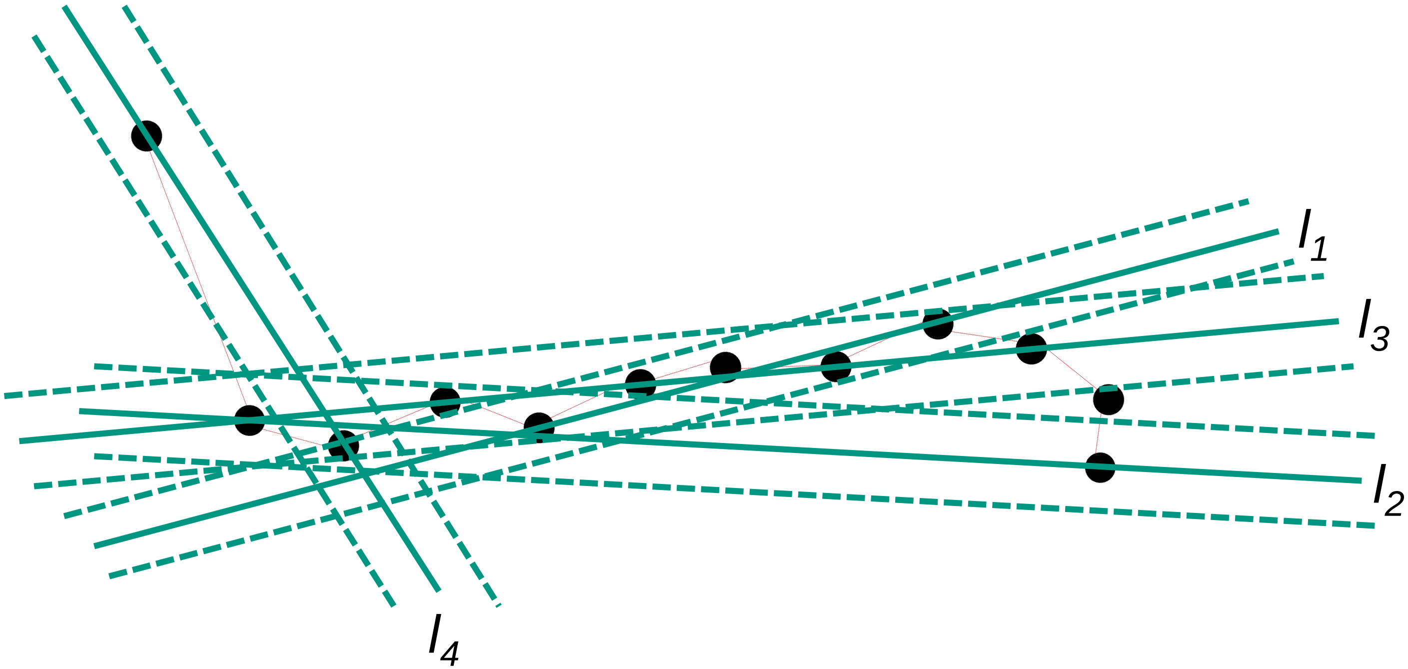
- algorithm:
  - pick randomly two points
  - fit line
  - check the number of points outside the tolerance band (=number of outliers)
  - repeat the process several times with different points
  - select the line with the smallest number of outliers
- RANSAC=<u>ran</u>dom <u>sa</u>mple <u>c</u>onsensus

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.

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

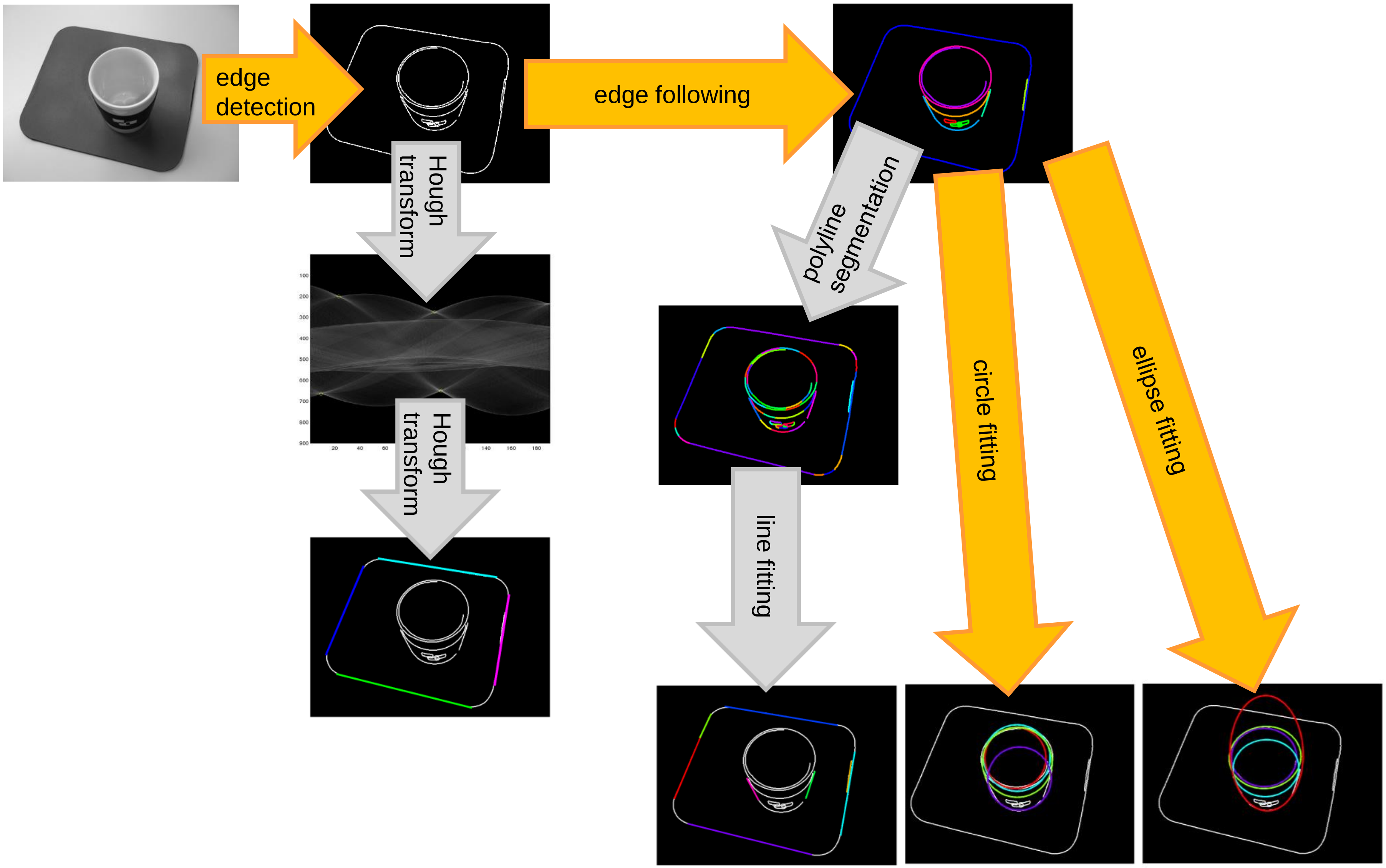


Robust Estimation

	M-estimator	RANSAC
• idea	reweight points according to their distance	ignore points with distance larger than a threshold
• parameters	error term, width parameter	acceptance threshold, number of trials
• algorithm	iterated weighted least squares	repeated guesses from pairs of points

→ demo tool

Contours Detection



Estimating Circles and Ellipses

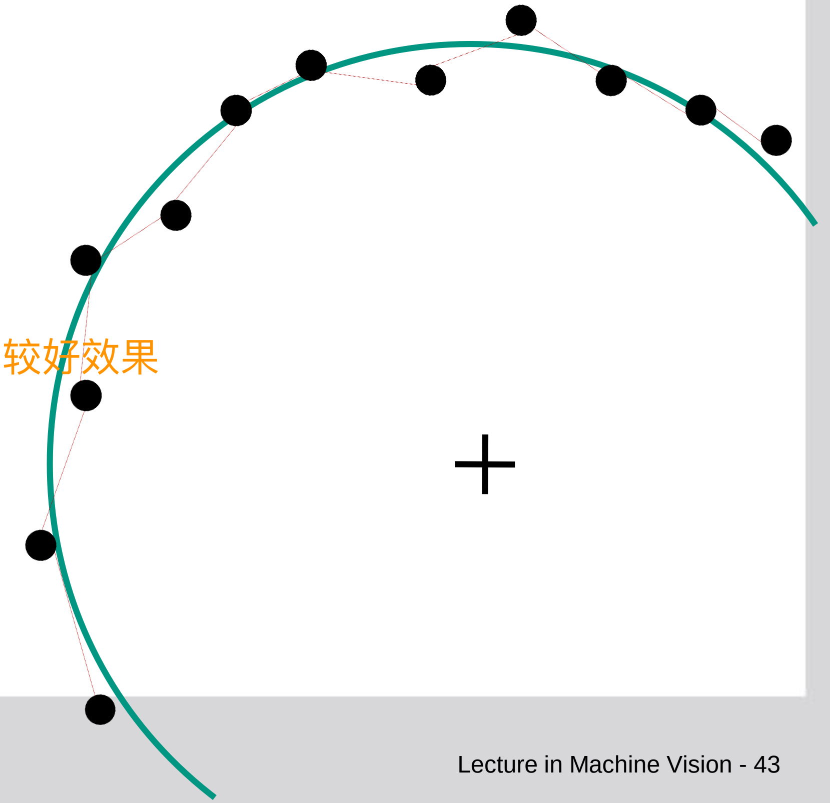
- determining parameters of circles/ ellipses that describe a curved contour from points

椭圆：确认5个点-
（因为拟合椭圆的隐式形式需要至少 5个点 来唯一确定 A, B, C, D, E, F 六个参数。）

满足椭圆约束条件：

4AC - B^2 > 0

随机采样：需要 5 个点确定椭圆的参数。
参数拟合：通过线性代数计算隐式方程参数
内点筛选：根据误差筛选内点。
迭代优化：使用 RANSAC 提升鲁棒性，对噪声和异常值具有较好效果



Estimating Circles

- parametric representation of circles:

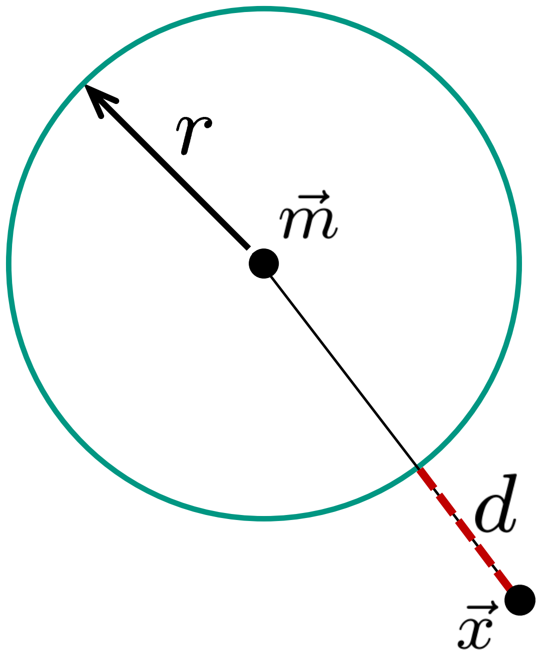
$$(x - m_1)^2 + (y - m_2)^2 - r^2 = 0$$

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

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

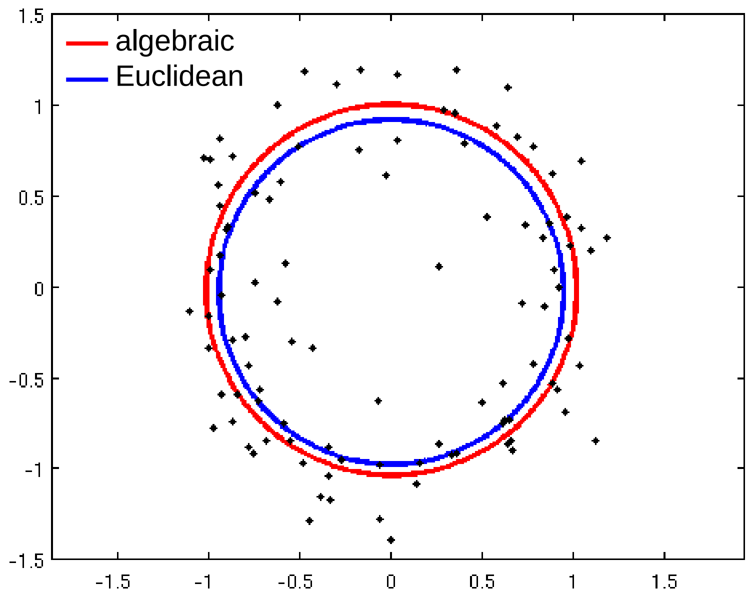
- algebraic distance:

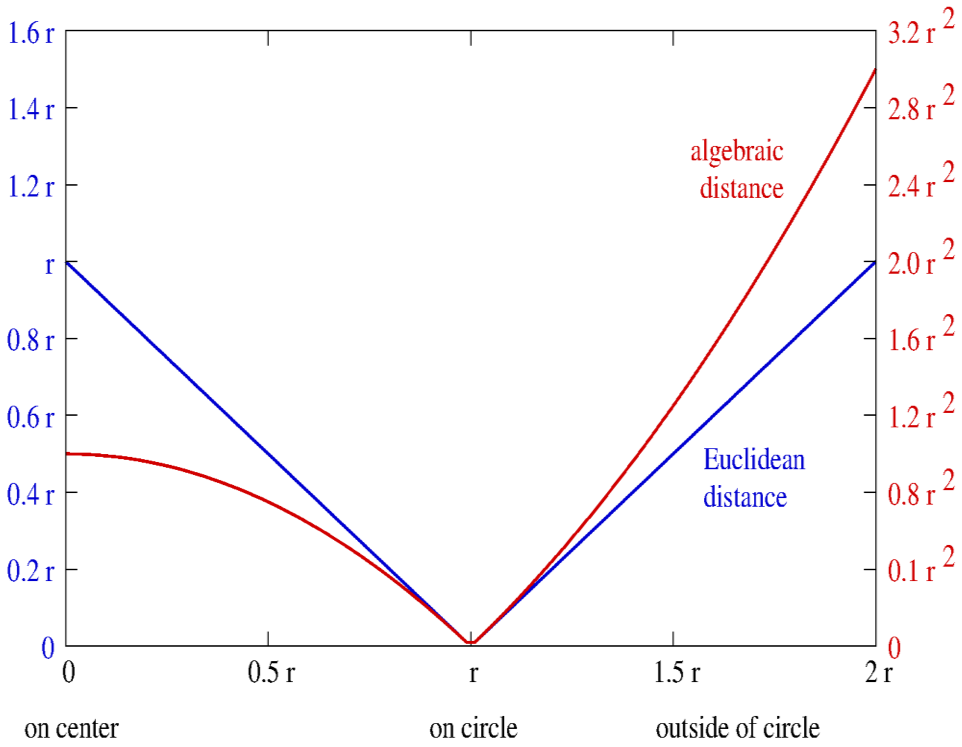
$$d_A = \left| (x - m_1)^2 + (y - m_2)^2 - r^2 \right|$$



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 → numerical optimization necessary
- 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:

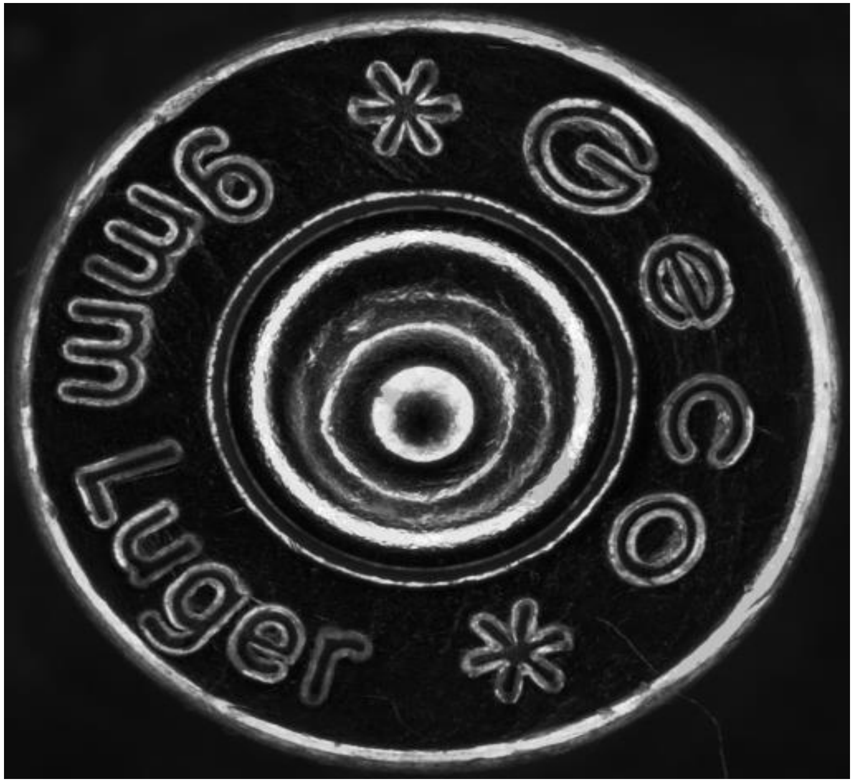
$$
\begin{pmatrix}
\sum_i x_i^2 & \sum_i x_i y_i & \sum_i x_i \\
\sum_i x_i y_i & \sum_i y_i^2 & \sum_i y_i \\
\sum_i x_i & \sum_i y_i & N
\end{pmatrix}
\begin{pmatrix} A \\ B \\ C \end{pmatrix}
=
\begin{pmatrix}
-\sum_i x_i(x_i^2+y_i^2) \\
-\sum_i y_i(x_i^2+y_i^2) \\
-\sum_i (x_i^2+y_i^2)
\end{pmatrix}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

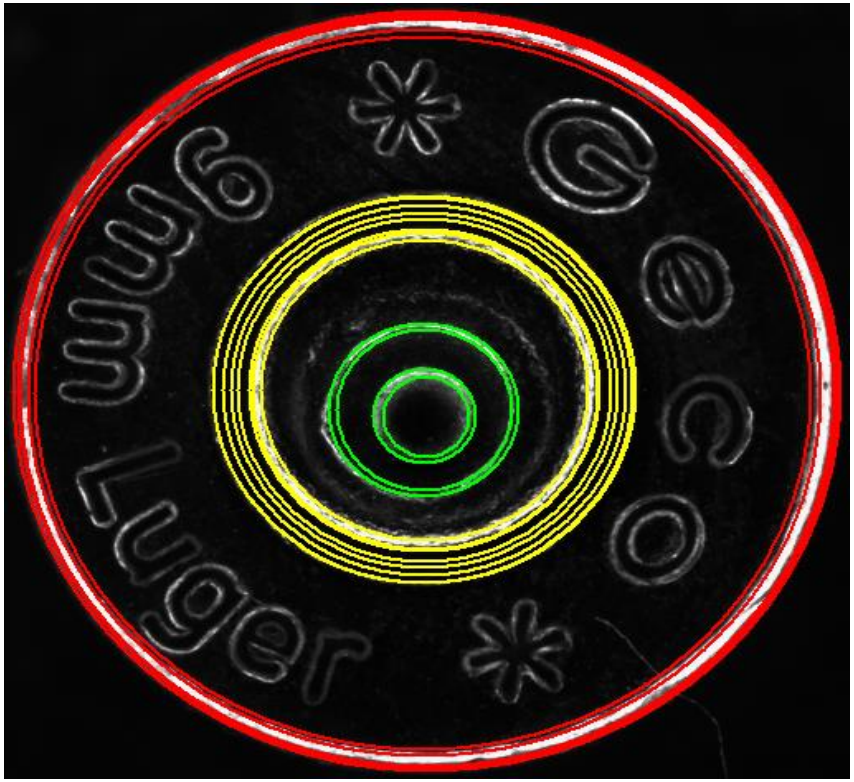
Estimating Circles cont.

– 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.





– 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

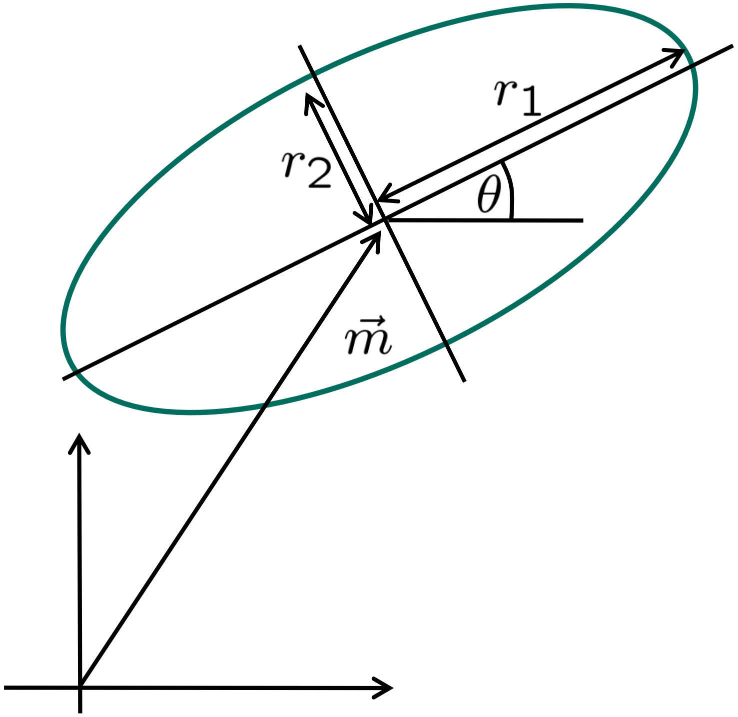
椭圆估计的方法

- 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 $4AB - H^2 > 0$ 椭圆的前提条件
- 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)

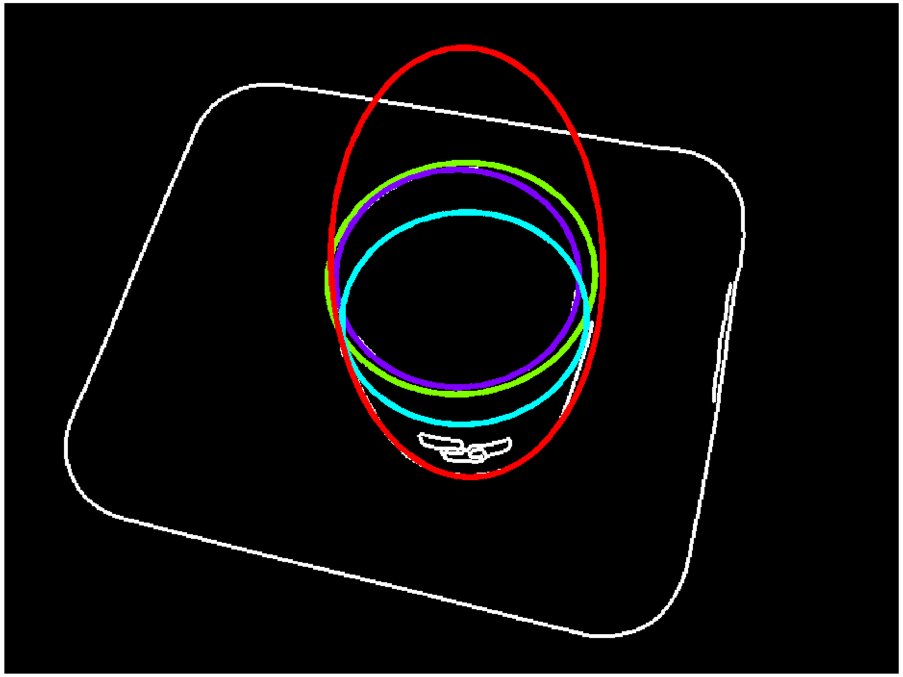
引入约束条件以减小一个自由度（左边是几种方法）

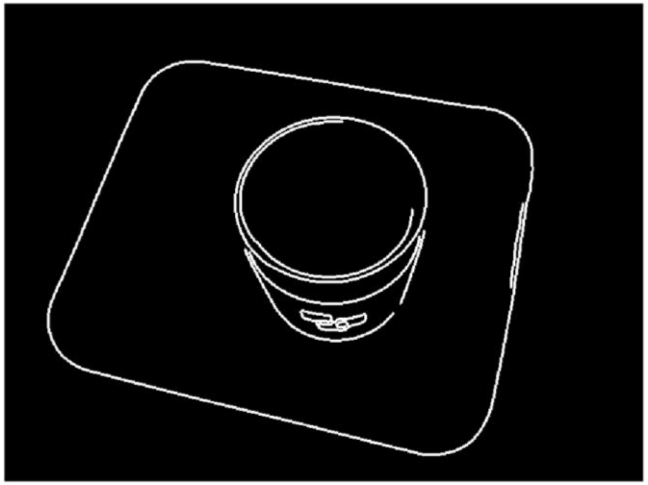


Estimating Ellipses

- 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.

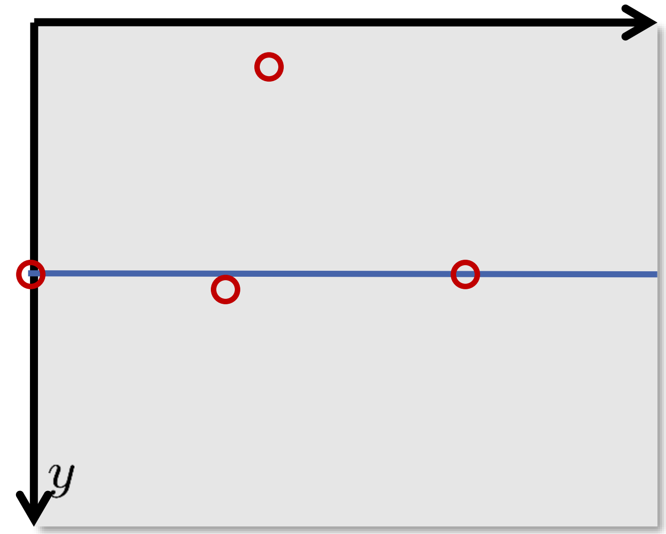


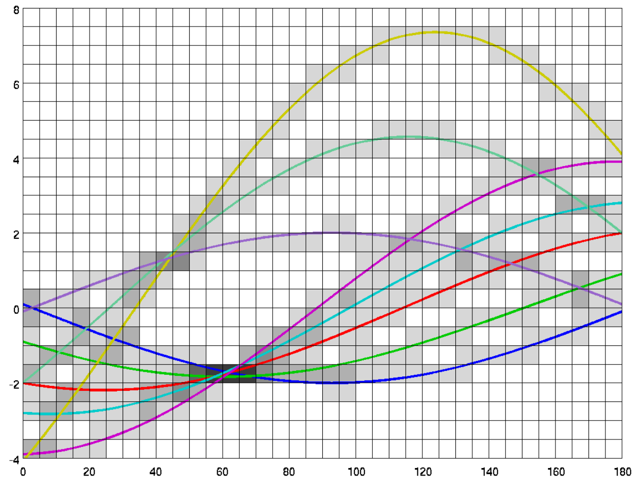


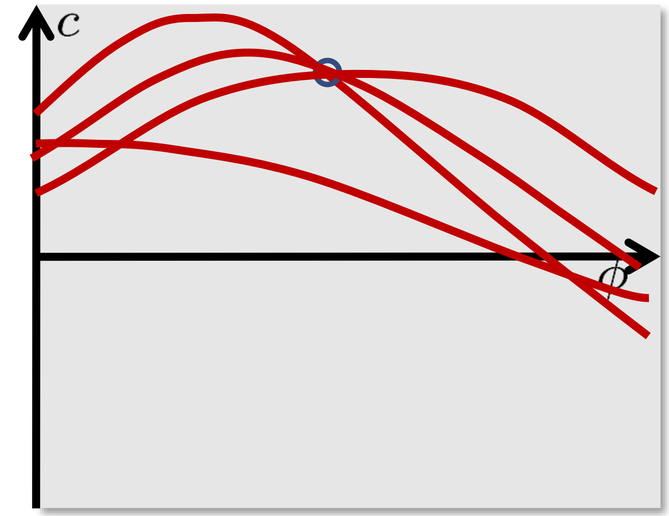
SUMMARY: CURVE FITTING

Summary cont.

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

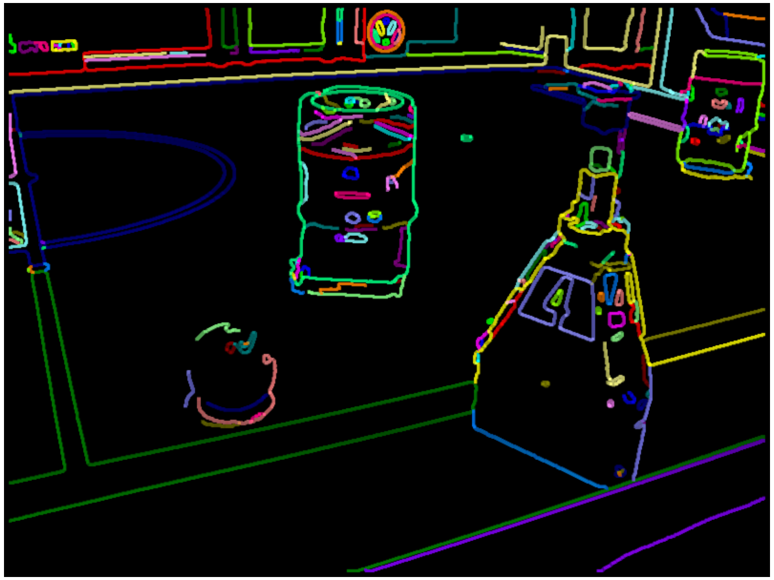


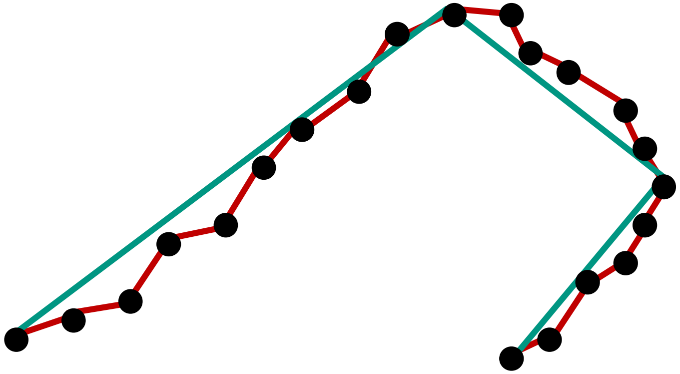




Summary cont.

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





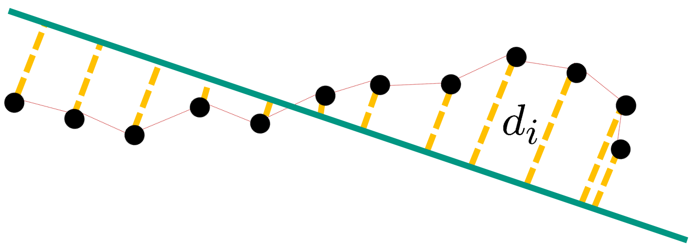
Summary cont.

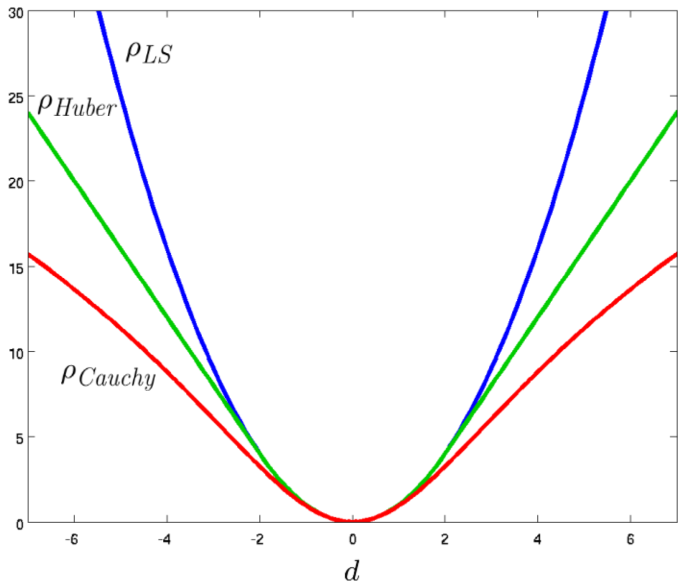
– **Hough transform**

– **polyline segmentation**

– **line estimation**
  - total least squares
  - M-estimators
  - RANSAC

– **circle and ellipse fitting**





Summary cont.

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