# 64-424
# Intelligente Roboter

http://tams.informatik.uni-hamburg.de/lectures/2011ws/vorlesung/ir

Jianwei Zhang

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

**Technische Aspekte Multimodaler Systeme**

Wintersemester 2011/2012

# Gliederung

# Verarbeitung von Abstandsmessungen

Für Messungen mit Lasermesssystemen gibt es Verfahren zur

- Filterung von Scans:
  - Glätten
  - Datenreduktion
- Extraktion von Liniensegmenten
- Extraktion von Ecken
- Klassifikation von Scanpunkten

# Scan

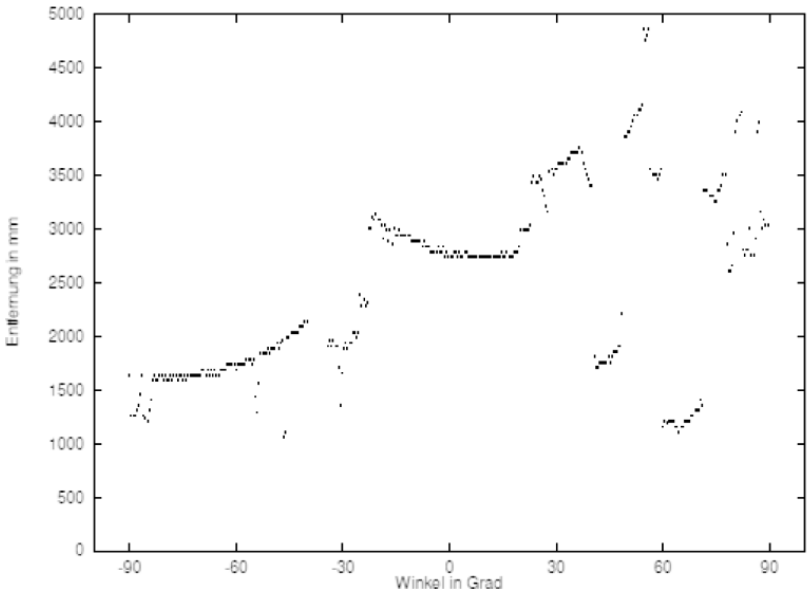
- Ein Scan ist eine Menge von Messwerten

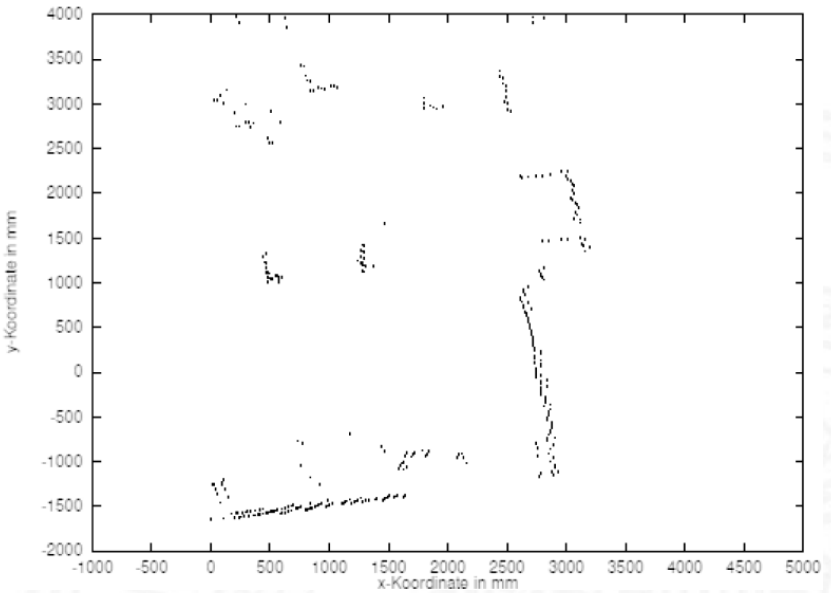
$$\left\{ m_i = (\alpha_i, r_i)^T | i = 0 \dots n-1 \right\}$$

  welche in Polarkoordinaten $(\alpha_i, r_i)^T$ angegeben sind
- Ein Scanpunkt $m_i = (\alpha_i, r_i)^T$ kann für eine gegebene Aufnahmeposition $l = (x, y, \theta)^T$ in absolute Koordinaten umgerechnet werden

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} r_i \cos\alpha_i \\ r_i \sin\alpha_i \end{bmatrix}$$

# Scan (cont.)


Entfernung in mm
Winkel in Grad
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
-90
-60
-30
0
30
60
90


y-Koordinate in mm
x-Koordinate in mm
4000
3500
3000
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
-1000
-500
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

# Filtern von Scandaten

- Problem von Scandaten: große Datenmenge, ungewünschte Scanpunkte
- Daher oftmals vorab Filterung von Scandaten
- gängige Filter:
  - Medianfilter
  - Reduktionsfilter
  - Winkelreduktionsfilter
  - Linienfilter

# Medianfilter

- Der Medianfilter erkennt Ausreißer und ersetzt diese durch eine geeignete Messung
- Um jeden Scanpunkt wird ein Fenster gelegt, das die Messungen vor und nach dem Punkt enthält
- Der Scanpunkt wird ersetzt durch einen Punkt, der denselben Aufnahmewinkel, aber den Median der Entfernungsmessungen des betrachteten Fensters als Entfernung hat
- *wSize* bestimmt die Fenstergröße (Anzahl der Punkte im Medianfilter)
- Ein großer Wert bedeutet eine starke Glättung
- Nachteil des Medianfilters: Ecken werden abgerundet

# Medianfilter (cont.)

**Algorithmus** Medianfilter

**Eingabe** Scan $s$, Fenstergröße *wSize*

**Ausgabe** Scan $s'$

**for** $i := 0$ **to** numpoints($s$)-1 **do**
  $p :=$ n-th-scanpoint($s$,$i$)
  **for** $j := 0$ **to** *wSize* **do**
    $k := (i + j$ - *wSize*/2) *mod* numpoints($s$)
    $p_k :=$ n-th-scanpoint($s$,$k$)
    $d(j) :=$ distance-value($p_k$)
  **endfor**
  $d_{median} :=$ median($d$)
  n-th-scanpoint($s'$,$i$) := (angle-value($p$),$d_{median}$)
**endfor**
**return** $s'$

# Medianfilter (cont.)

# Reduktionsfilter

- Der Reduktionsfilter fasst Punktwolken zu einem Punkt zusammen
- Eine Punktwolke wird durch einen Radius $r$ angegeben
- Der erste Punkt (Ausgangspunkt) eines Scans gehört zu einer Wolke
- Alle folgenden Punkte mit Abstand $d < 2 \cdot r$ werden zur Wolke hinzugefügt
- Beim ersten Punkt mit größerem Abstand wird eine neue Wolke angefangen
- Jede Wolke wird durch den Schwerpunkt der ihr zugeordneten Punkte ersetzt

# Reduktionsfilter (cont.)

**Algorithmus** Reduktionsfilter

**Eingabe** Scan $s$, Radius $r$

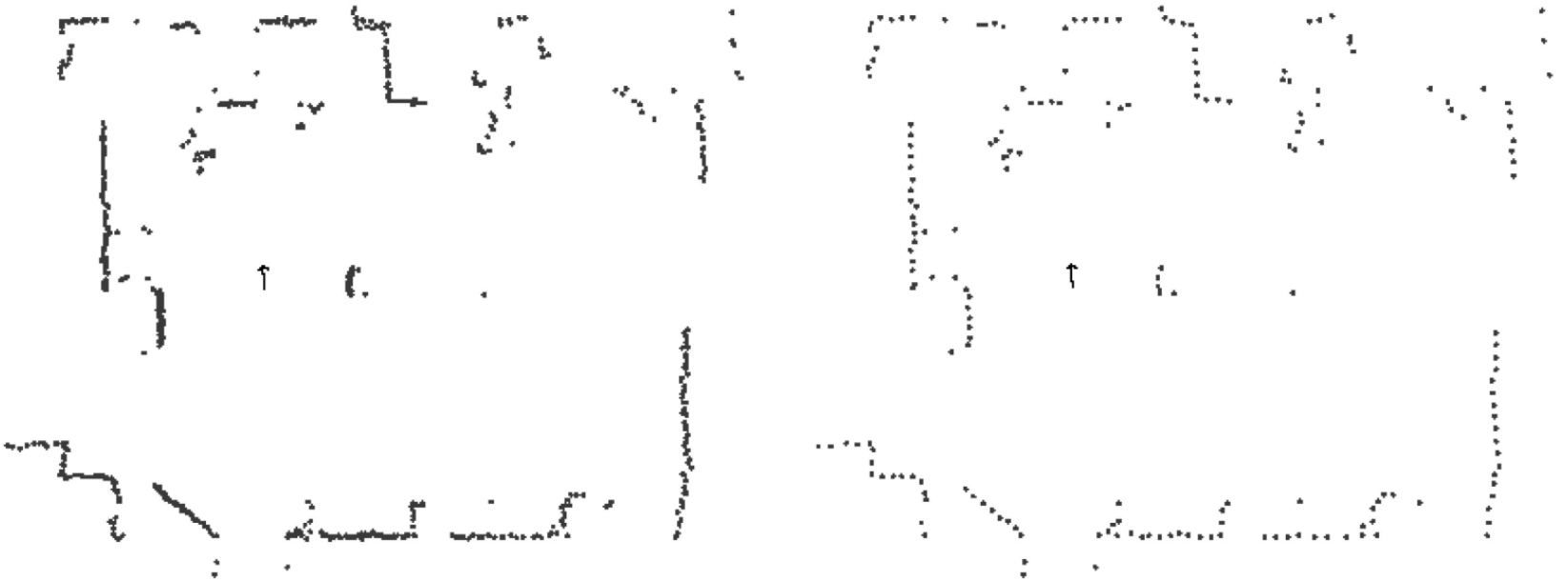
**Ausgabe** Scan $s'$

$j := 0$
$p_0 :=$ n-th-scanpoint($s$,0)
$p_{sum} := p_0$
$n := 0$
**for** $i := 1$ **to** numpoints($s$)-1 **do**
  $p :=$ n-th-scanpoint($s$,$i$)
  **if** distance($p_0$,$p$) $< 2r$ **then**
    $p_{sum} := p_{sum} + p$
    $n := n + 1$

# Reduktionsfilter (cont.)

**else**
n-th-scanpoint$(s',j) := p_{sum}/n$
$j := j + 1$
$P_0 := p$
$p_{sum} := p_0$
$n := 1$
**endif**
**endfor**
n-th-scanpoint$(s',j) := p_{sum}/n$
$j := j + 1$
numpoints$(s') := j$
**return** $s'$

# Reduktionsfilter (cont.)



# Reduktionsfilter (cont.)

- Der Algorithmus des Reduktionsfilters hat eine Zeitkomplexität von $O(n)$, wenn $n$ die Anzahl der Punkte ist
- Vorteile des Reduktionsfilters:
  - Reduzierung der Anzahl der Scanpunkte ohne Verlust wesentlicher Informationen
  - Dies führt zu kürzeren Laufzeiten bei der Nachbearbeitung eines Scans
  - Es ergibt sich eine bessere Gleichverteilung der Punkte
- Nachteile des Reduktionsfilters:
  - Extraktion von Merkmalen nicht mehr so einfach
  - Möglicherweise zu wenig Punkte für Merkmal
  - Daher besser: Merkmalsextraktion vor Reduktionsfilter

# Winkelreduktionsfilter

- Der Winkelreduktionsfilter ähnelt dem Reduktionsfilter
- Scanpunkte mit ähnlichem Aufnahmewinkel werden gruppiert und durch den Punkt ersetzt, dessen Entfernung gleich dem Median der Entfernungswerte ist
- Die Funktion $median - dist(q, n)$ liefert im folgenden Algorithmus diesen Punkt
- Die Zeitkomplexität ist $O(n)$, wenn $n$ die Anzahl der Scanpunkte ist
- Der Winkelreduktionsfilter wird eingesetzt um Scans mit hoher Winkelauflösung gleichmäßig zu reduzieren
- Dies wird z.B. bei der Verschmelzung mehrerer Scans zu einem Scan eingesetzt

# Winkelreduktionsfilter (cont.)

**Algorithmus** Winkelreduktionsfilter

**Eingabe** Scan $s$, Winkel $\alpha$

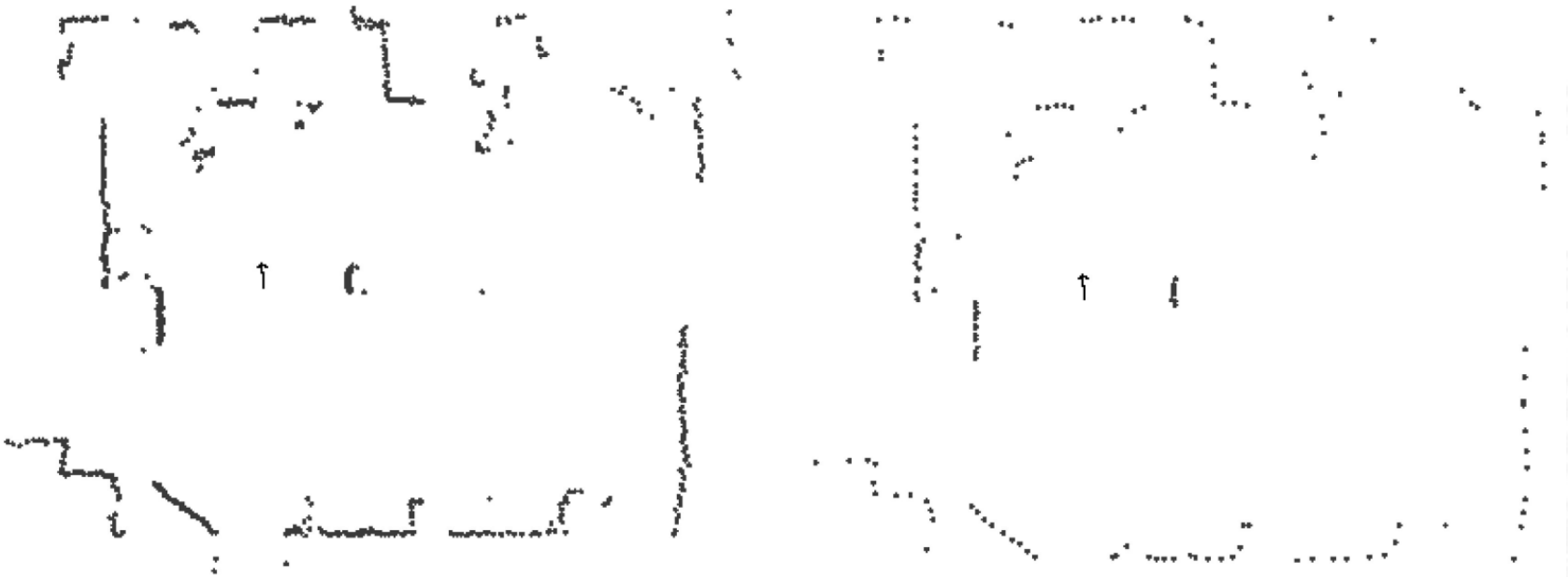
**Ausgabe** Scan $s'$

$j := 0$
$q(0) :=$ n-th-scanpoint($s$,0)
$n := 1$
**for** $i := 1$ **to** numpoints($s$)-1 **do**
  $p :=$ n-th-scanpoint($s$,$i$)
  **if** *abs*(angle-value($p$) - angle-value($q(0)$)) $< \alpha$ **then**
    $q(n) := p$
    $n := n + 1$

# Winkelreduktionsfilter (cont.)

**else**
n-th-scanpoint($s'$,$j$) := median-dist($q$,$n$)
$j := j + 1$
$q(0) := p$
$n := 1$
**endif**
**endfor**
n-th-scanpoint($s'$,$j$) := median-dist($q$,$n$)
$j := j + 1$
numpoints($s'$) := $j$
**return** $s'$

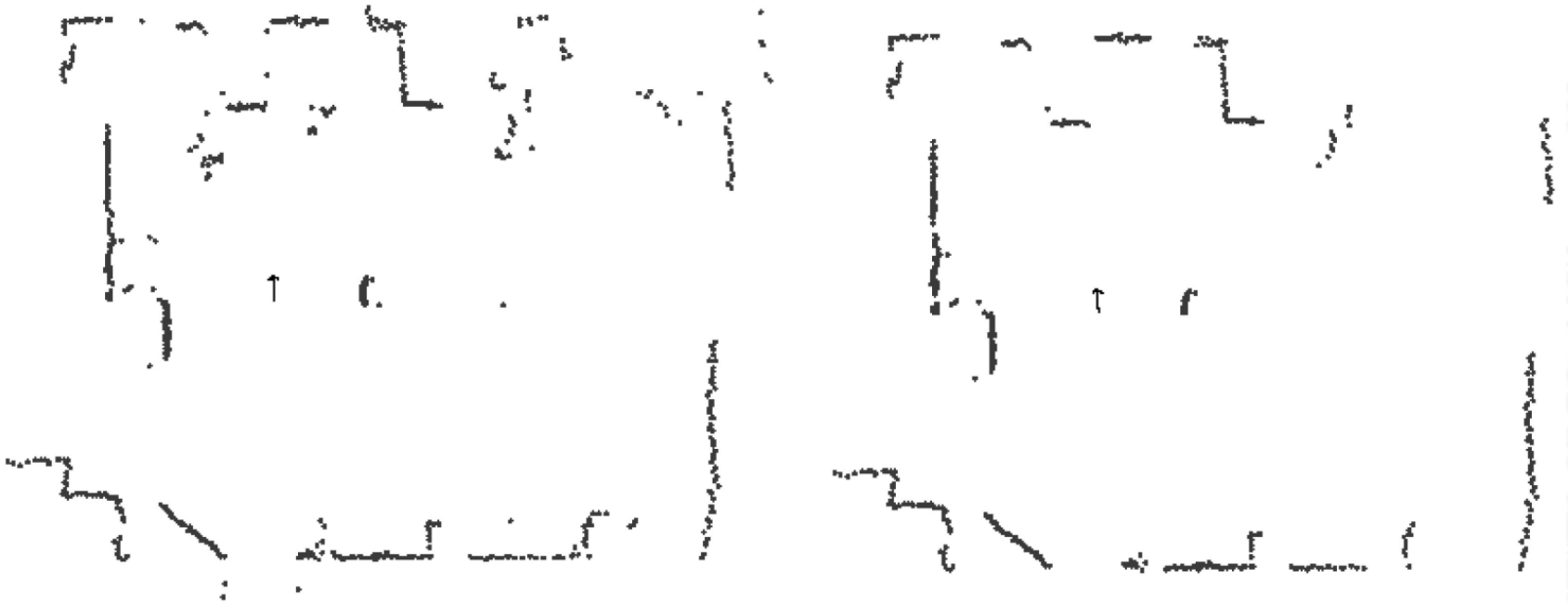
# Winkelreduktionsfilter (cont.)



# Linienfilter

- Der Linienfilter nutzt das später vorgestellte Verfahren zur Linienextraktion
- Die Scanpunkte, die keinem Liniensegment zugeordnet wurden, werden entfernt
- Die Zeitkomplexität ist gleich der Komplexität der Linienextraktion ($O(n \log n)$ im mittleren Fall)
- Der Filter wird angewendet, wenn anschließend angewandte Algorithmen polygonale Umgebungen erfordern

# Linienfilter (cont.)



# Merkmalsextraktion

- Keine Verarbeitung von kompletten Scans sondern Merkmalsextraktion
- Häufige Merkmale: Linien, Ecken
- *maxDist* im folgenden Algorithmus ist der maximal zulässige Abstand zweier aufeinander folgender Punkte für die Gruppierung

# Linien

**Algorithmus** Linienextraktion

**Eingabe** Scan $s$, Parameter *maxDist*

**Ausgabe** Menge von Linien $l$

$l$ := empty
*start* := 0
**for** $i$:=1 **to** numpoints($s$)-1 **do**
  $p_1$ := n-th-scanpoint($s$,$i$-1)
  $p_2$ := n-th-scanpoint($s$,$i$)
  **if** distance($p_1$,$p_2$) > *maxDist* **then**
    $l$ := $l$ ∪ split($s$,*start*,$i$-1)
    *start* := $i$
  **endif**
**endfor**
$l$ := $l$ ∪ split($s$,*start*,numpoints($s$)-1)
**return** $l$

# Linien (cont.)

**Algorithmus** split($s$,*start*,*end*)

**Eingabe** Scanpunkte, durch $s$, *start* und *end* festgelegt
Parameter *minPointsOnLine*, *maxSigma*

**Ausgabe** Menge von Linien $l$

$l$ := empty
*line* := make-line($s$,*start*,*end*)
**if** numpoints(*line*) $\geq$ *minPointsOnLine* **then**
  **if** $\sigma$(*line*) $<$ *maxSigma* **then**
    $l := l \cup \{line\}$
  **else**
    $p_{start}$ := n-th-scanpoint($s$,*start*)
    $p_{end}$ := n-th-scanpoint($s$,*end*)
    $i_{split}$ := *start*
    $d := 0$

# Linien (cont.)

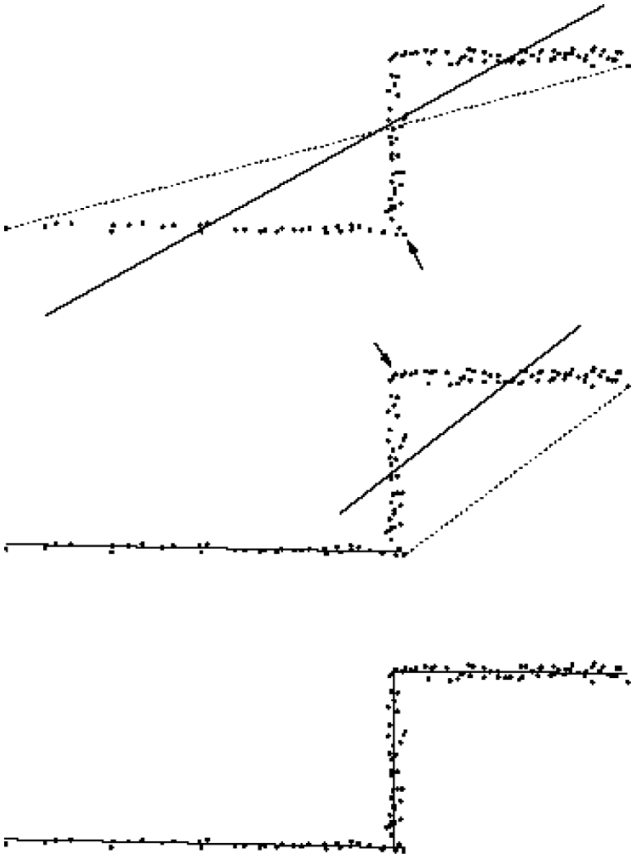
**for** $i$ := $start$+1 **to** $end$-1 **do**
  $p$ := n-th-scanpoint($s$,$i$)
  **if** distance-to-line($p$,$p_{start}$,$p_{end}$) $>$ $d$ **then**
    $i_{split}$:= $i$
    $d$ := distance-to-line($p$,$p_{start}$,$p_{end}$)
  **endif**
**endfor**
$l$ := $l$ $\cup$ split($s$,$start$,$i_{split}$)
$l$ := $l$ $\cup$ split($s$,$i_{split}$,$end$)
**endif**
**endif**
**return** $l$

- Die *split*-Funktion ist rekursiv
- Zuerst wird eine Ausgleichsgerade durch die Punkte gelegt

# Linien (cont.)

- Ist die Abweichung $\sigma(line)$ zu groß wird die Punktmenge aufgeteilt und für die neuen Mengen die Funktion *split* aufgerufen
- Der Punkt an dem aufgeteilt wird, ist der Punkt mit dem größten Abstand zur Geraden durch *start* und *end*
- *minPointsOnLine* und *maxSigma* bestimmen die Anzahl und Qualität der Linien
- Die Linienextraktion ist ein typischer *divide and conquer* Algorithmus
- Zeitkomplexität ähnlich *Quicksort*: $O(n^2)$ im schlechtesten, $O(n \log n)$ im mittleren Fall ($n$: Anzahl der Scanpunkte)

# Linien (cont.)



# Linien (cont.)

# Ecken

- Ähnlich wie Linienalgoritmus
- Aufeinander folgende Linien werden miteinander geschnitten
- Nur Ersetzen der *split*-Funktion notwendig
- Gleiche Zeitkomplexität

**Algorithmus** split($s$,$start$,$end$)

**Eingabe** Scanpunkte, durch $s$, $start$ und $end$ festgelegt
Parameter $minPointsCorner$, $maxSigma$

**Ausgabe** Menge von Ecken $e$

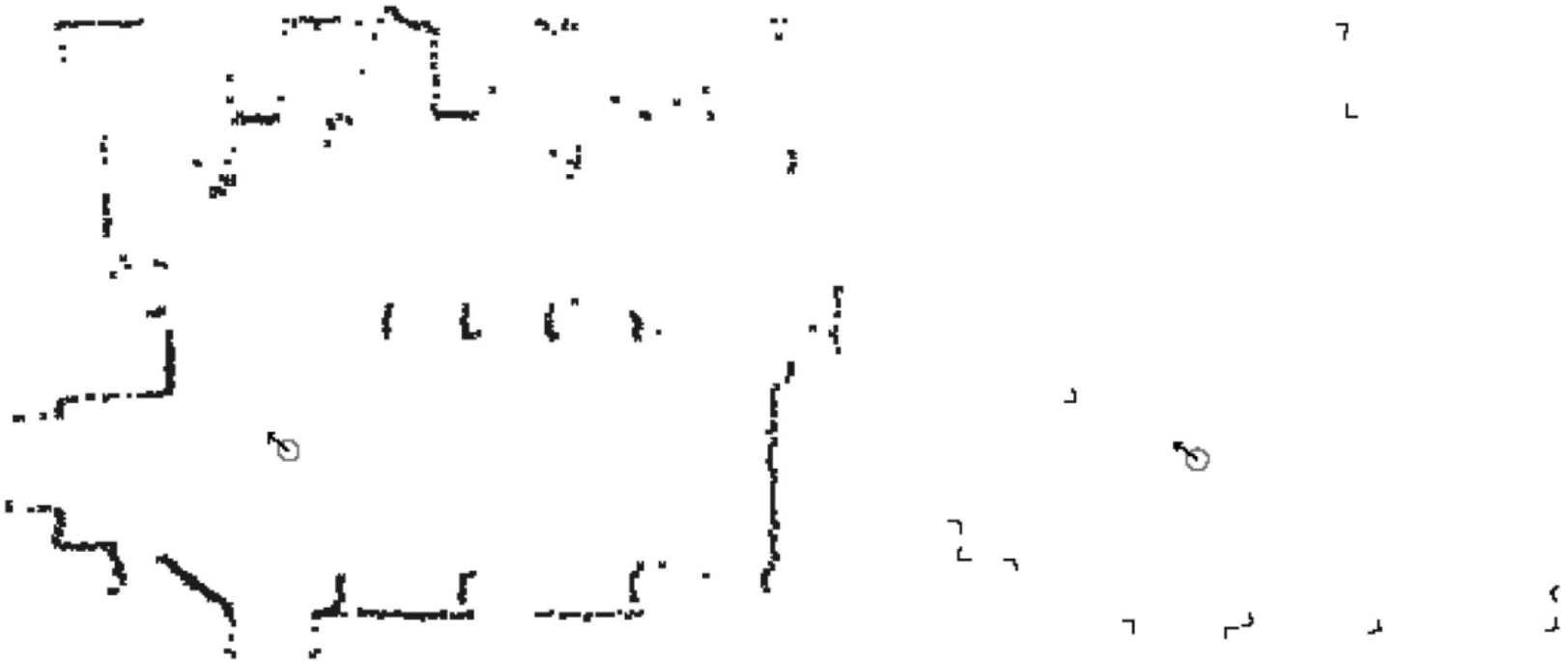
# Ecken (cont.)

$e$ := empty
**if** ($end$ -$start$) $\geq 2 \cdot minPointsCorner$ **then**
  $p_{start}$ := n-th-scanpoint($s$,$start$)
  $p_{end}$ := n-th-scanpoint($s$,$end$)
  $i_{split}$ := $start$
  $d$ := 0
  **for** $i$ := $start$+1 **to** $end$-1 **do**
    $p$ := n-th-scanpoint($s$,$i$)
    **if** distance-to-line($p$,$p_{start}$,$p_{end}$) $>$ $d$ **then**
      $i_{split}$ := $i$
      $d$ := distance-to-line($p$,$p_{start}$,$p_{end}$)
    **endif**
  **endfor**

# Ecken (cont.)

**if** $(i_{split}$ - $start) \geq minPointsCorner$ **and**
$(end$ - $i_{split}) \geq minPointsCorner$ **then**
$line_1$ := make-line($s$,$i_{split}$ - $minPointsCorner$,$i_{split}$)
$line_2$ := make-line($s$,$i_{split}$, $i_{split} + minPointsCorner$)
**if** $\sigma(line_1) < maxSigma$ and $\sigma(line_2) < maxSigma$ **then**
$e := e \cup \{$make-corner($line_1$,$line_2$)$\}$
**endif** **endif**
$e := e \cup$ split($s$,$start$,$i_{split}$)
$e := e \cup$ split($s$,$i_{split}$,$end$)
**endif**
**return** $e$

# Ecken (cont.)



# Scan-Matching

- In der mobilen Robotik werden Laserscans häufig eingesetzt, um die Position des Roboters in einer Karte zu finden
- Dazu wird der Scan mit Hilfe der Linienextraktion in eine Menge von Linien umgewandelt
- Die gemessene Anordnung der Linien wird in einer Karte durch Überdeckung gesucht
- Dieses Verfahren wird Scan-Matching genannt
- Es kann auch direkt auf den Abstandsmesswerten durchgeführt werden

# Verfahren von Cox

- Einer der ersten Vorschläge zum Überdecken von Scandaten und einem *a priori* Linienmodell stammt von Cox (1990, 1991)
- Hierbei wird jedem Scanpunkt eine Linie des *a priori* Modells zugeordnet
- Aus der Zuordnung lässt sich die Rotation und Translation gegenüber dem Linienmodell bestimmen
- Das Verfahren benötigt eine ungefähre Anfangsschätzung der Aufnahmeposition z.B. anhand der Odometriedaten

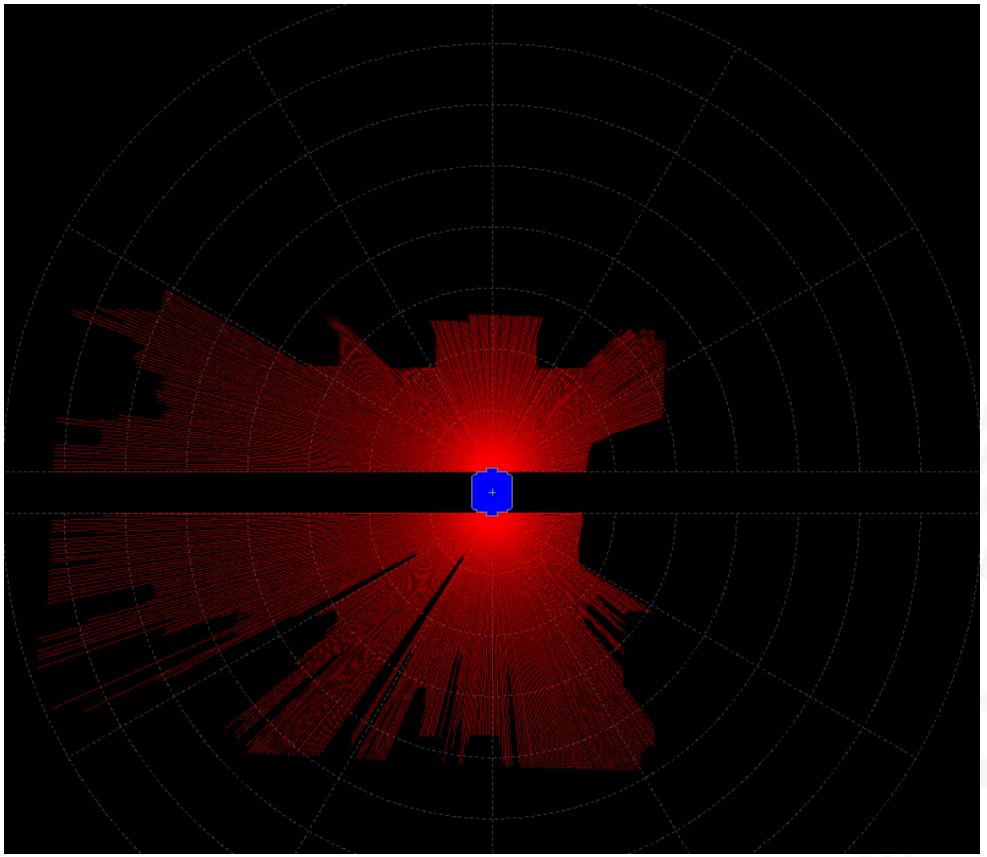
# Verfahren von Cox (cont.)

1. Setze $(\hat{x}, \hat{y}, \hat{\theta})^T = (s_x, s_y, s_\theta)^T$, wobei $(s_x, s_y, s_\theta)^T$ die initiale Positionsschätzung der Scanaufnahme anhand der Odometrie ist
2. Verschiebe und drehe Scan auf Position $(\hat{x}, \hat{y}, \hat{\theta})^T$
3. Bestimme für jeden Scanpunkt die Modellinie, die dem Punkt am nächsten liegt. Diese Modellinie wird im folgenden Ziellinie genannt
4. Berechne die Transformation $\hat{b} = (\delta x, \delta y, \delta\theta)^T$, welche die Summe der Abstandsquadrate zwischen den Scanpunkten und der jeweiligen Ziellinie minimiert
5. Setze $(\hat{x}, \hat{y}, \hat{\theta})^T = (\hat{x}, \hat{y}, \hat{\theta})^T + (\delta x, \delta y, \delta\theta)^T$

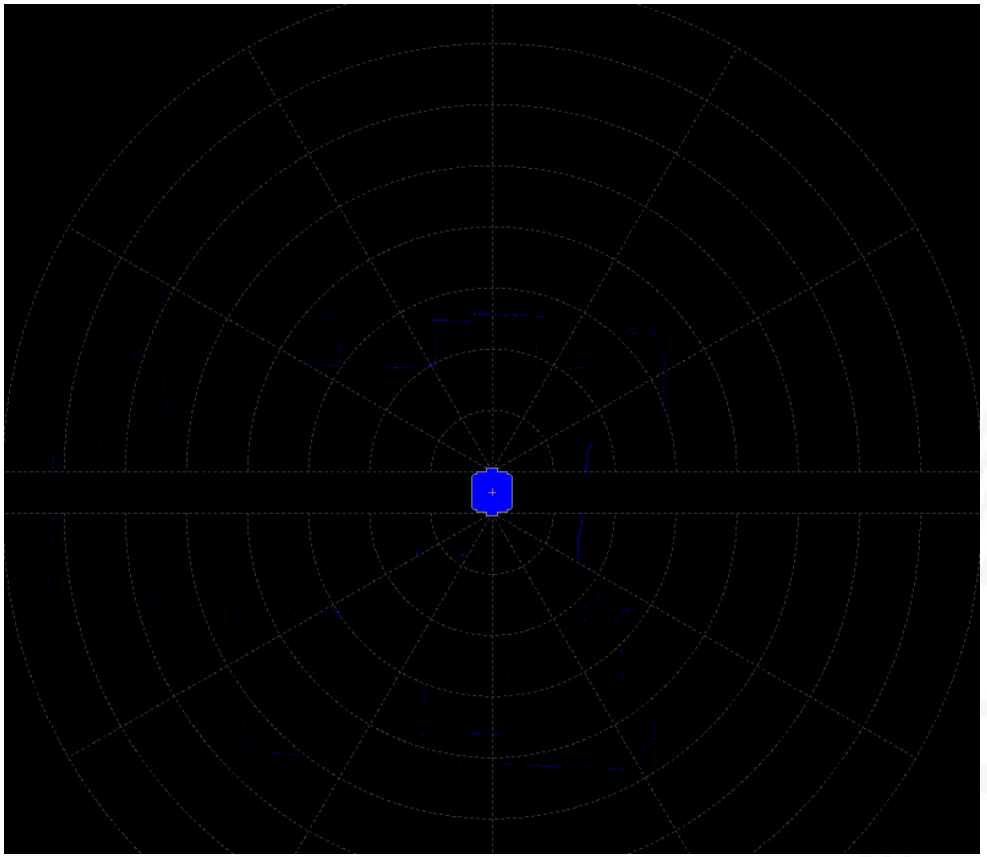
# Verfahren von Cox (cont.)

6. Wiederhole Schritte 2-5 bis das Verfahren konvergiert
   Das Ergebnis der Überdeckung ist $(\hat{x}, \hat{y}, \hat{\theta})^T$
7. Berechne die Fehlerkovarianzmatrix $\sum_{match}$

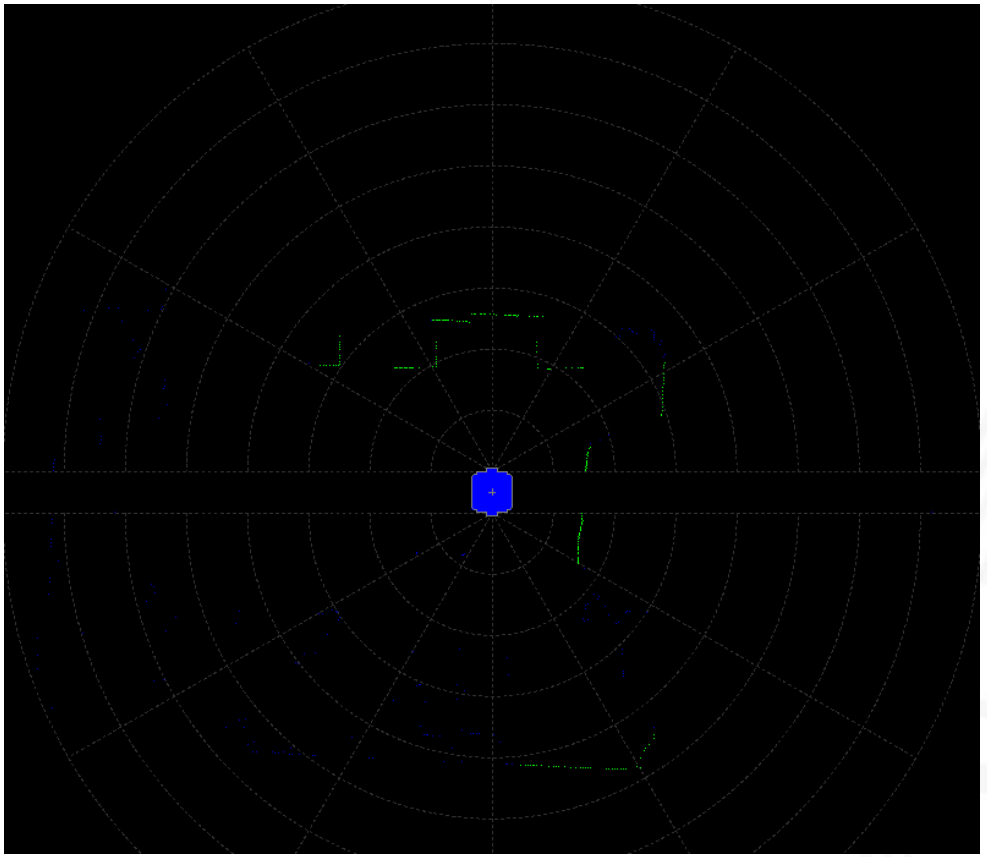
# Zusammenfassen von Linien-Segmenten



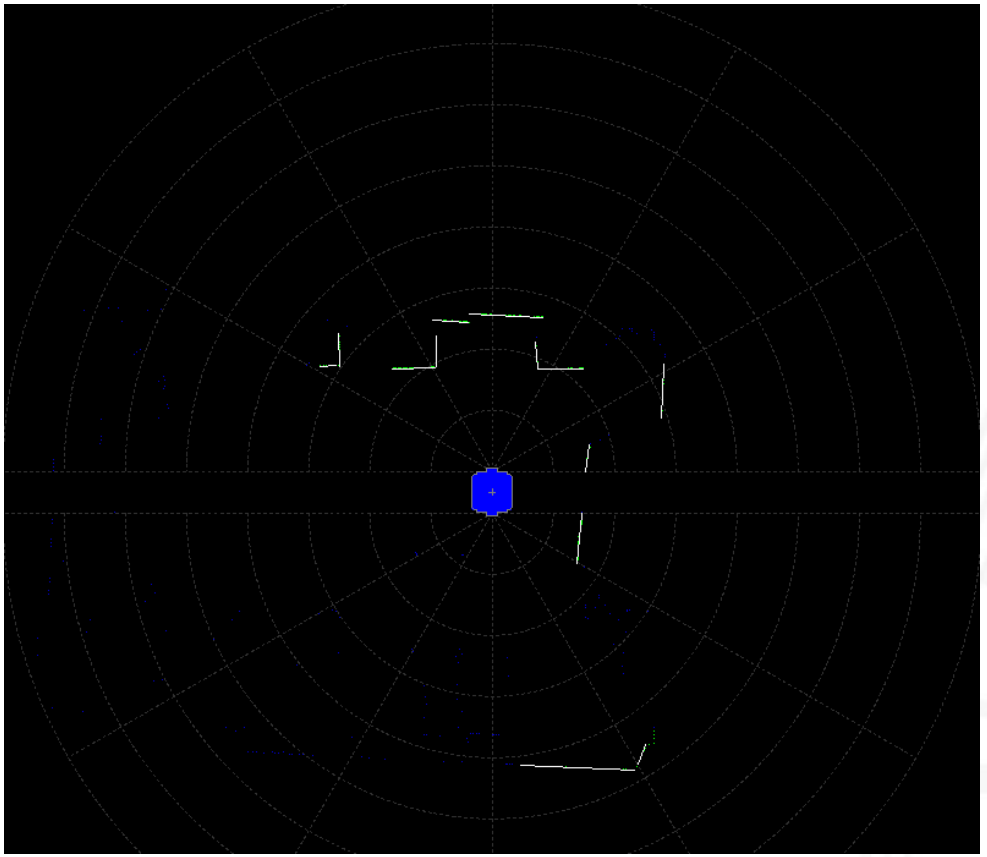
# Zusammenfassen von Linien-Segmenten (cont.)



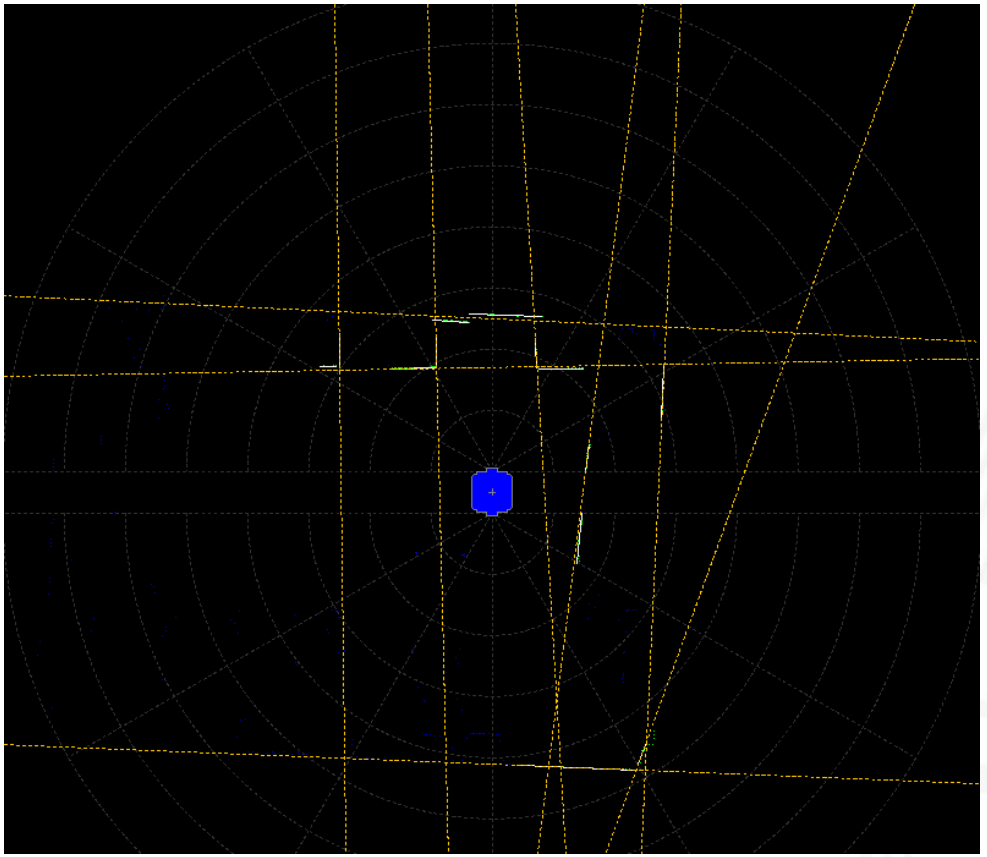
# Zusammenfassen von Linien-Segmenten (cont.)



# Zusammenfassen von Linien-Segmenten (cont.)



# Zusammenfassen von Linien-Segmenten (cont.)



# Hough-Transformation

- Verfahren zur Erkennung von
  - Geraden
  - Kreisen
  - beliebigen anderen geometrischen Figuren
- Häufig in der Bildverarbeitung angewandt auf Gradientenbildern
- Punkt im Bild wird in einen Parameterraum abgebildet
- Geeignete Parameter:
  - Gerade: Steigung und y-Achsen-Abschnitt
  - Kreis: Radius und Mittelpunkt
- Punkt im Bildraum entspricht einer Figur im Parameterraum
- Gesuchte Figur findet sich bei Häufungen im Parameterraum

# Geradenerkennung

- **Parameter:** Steigung und y-Achsen-Abschnitt
- **Nachteil:** Geraden mit unendlicher Steigung können nicht abgebildet werden

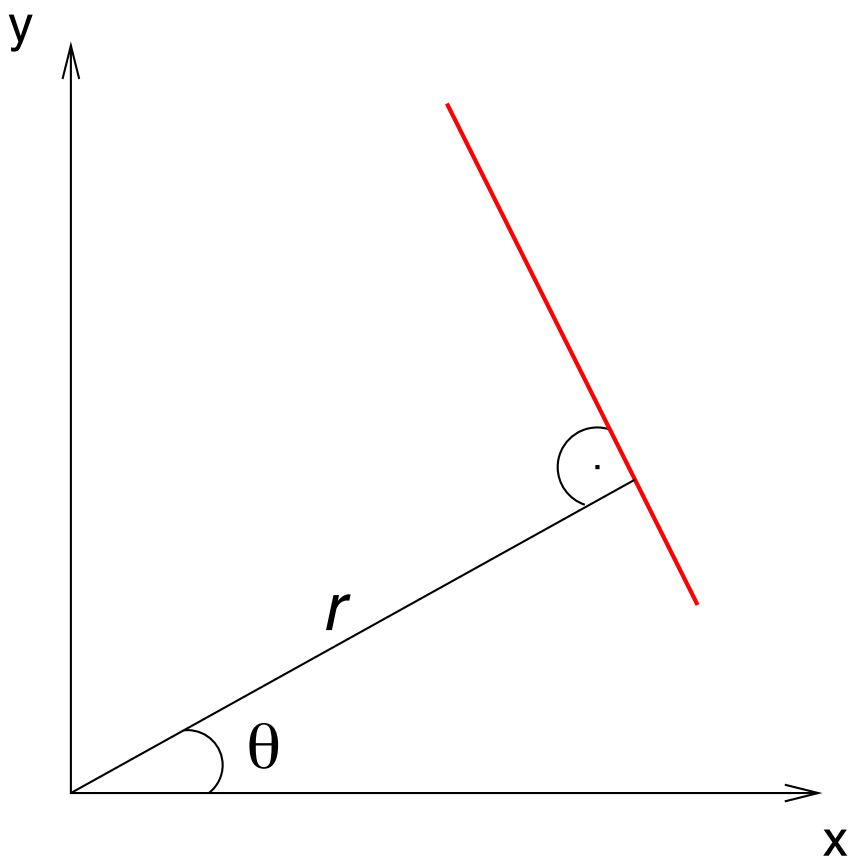
- **Besser:** Gerade in Hessescher Normalform

$$r = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

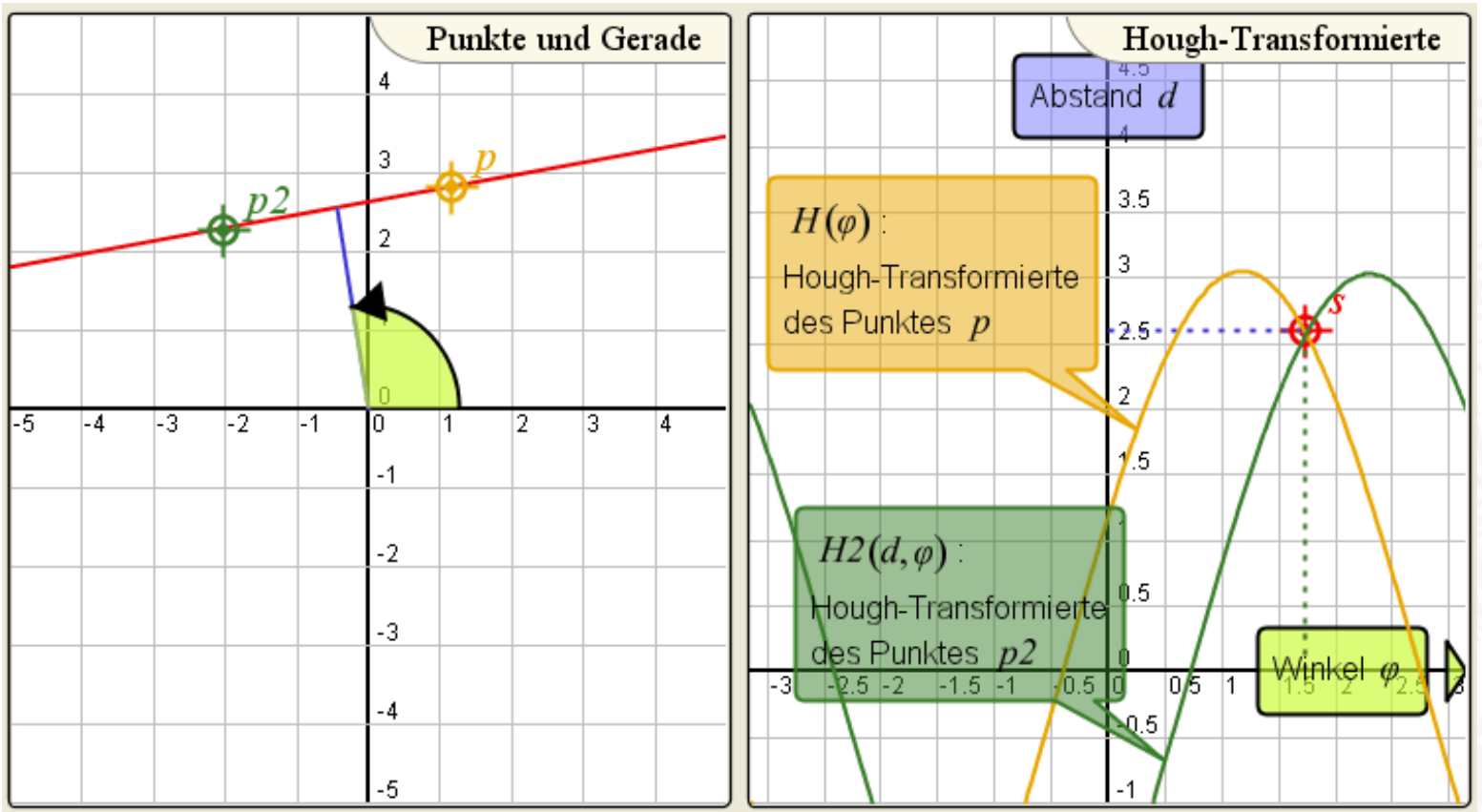
mit

- $\theta$: Winkel zwischen der x-Achse und der Normalen der Geraden
- $r$: Abstand vom Ursprung zur Geraden

# Geradenerkennung (cont.)



# Geradenerkennung (cont.)



# Geradenerkennung (cont.)

- Alle Linien-Segmente aus der Linienerkennung lassen sich in Hessescher Normalform darstellen
- Für jedes Linien-Segment wird im Parameterraum ein $\theta$-$r$-Punkt eingetragen
- Für Häufungen (engl. *Cluster*) wird ein stellvertretender Schwerpunkt berechnet
- Parameter des Schwerpunktes beschreiben eine Gerade auf der (in etwa) die Liniensegmente liegen
- Schwerpunkte können mit Hierarchischem Clustering gefunden werden

# Hierarchisches Clustering

Zwei grundsätzliche Verfahren:

- **anhäufende Verfahren** (engl. *agglomerative clustering*)
  - In der Praxis häufig eingesetzt
  - Es werden schrittweise einzelne Elemente zu Gruppen zusammengefasst
- teilende Verfahren (engl. *divisive clustering*)
  - Es wird eine große Gruppe in kleinere Gruppen unterteilt

# Agglomerative Clustering

1. Alle Elemente sind einzelne Cluster
2. Die zueinander am nächsten liegenden Cluster werden zusammengefasst
3. Wiederhole 2. bis
   - alle Cluster eine bestimmte Distanz zueinander überschreiten

oder

   - eine minimale Anzahl an Clustern erreicht ist

- Es muss eine Distanzfunktion d für den Abstand zweier Cluster gegeben sein

# Typische Distanzfunktionen

Für den Abstand zweier Cluster $A$ und $B$ werden oft folgende Distanzfunktionen gewählt:

- Single Linkage Clustering: Minimaler Abstand zweier Elemente

$$\min_{a \in A, b \in B} \{d(a, b)\}$$

- Complete Linkage Clustering: Maximaler Abstand zweier Elemente

$$\max_{a \in A, b \in B} \{d(a, b)\}$$

- Average Linkage Clustering: Durchschn. Abstand aller Elemente

$$\frac{1}{|A||B|} \sum_{a \in A, b \in B} d(a, b)$$

# Typische Distanzfunktionen (cont.)

- Average Group Linkage: Durchschnittlicher Abstand aller Elemente der Vereinigung von $A$ und $B$

$$\frac{1}{|C|} \sum_{x,y \in C, C=A \cup B} d(x,y)$$

- Centroid Method: Abstand der Mittelwerte

$$d(\bar{x}, \bar{y})$$

- Ward's Method: Zunahme der Varianz beim Vereinigen von $A$ und $B$

$$\frac{d(\bar{x}, \bar{y})}{1/|A| + 1/|B|}$$

- und zahlreiche weitere Funktionen...

# Personen-Tracking mit Lasermesssystemen

- Fest installierte Lasermesssysteme können zum Verfolgen (engl. *Tracking*) von Personen genutzt werden
- Übliche Verfahren arbeiten in drei Schritten:
  1. Unterteilen der Messdaten in Vordergrund und Hintergrund durch Background Subtraction
  2. Gruppierung der Vordergrund-Daten
  3. Verfolgen der Gruppen in konsekutiven Scans

# Background Subtraction

- Zuerst wird ein Hintergrundmodell erstellt:
  - Zu jedem Winkel wird für einen bestimmten Zeitraum ein Histogramm der Entfernungsmessungen ermittelt
  - Das Histogramm für jeden Winkel $\alpha$ wird durch eine Verteilungsfunktion beschrieben (meistens: Gauß-Verteilung mit Mittelwert $\mu_\alpha$ und Standardabweichung $\sigma_\alpha$)
- Mit Hilfe des Hintergrundmodells werden die Scans während des Betriebs gefiltert:
  - Abstandsmessungen die kleiner als $\mu_\alpha - n \cdot \sigma_\alpha$ werden als Vordergrund klassifiziert
  - Die Vordergrund-Messungen können ähnlich wie bei der Linien-Extraktion zu Gruppen zusammengefasst werden

# Background Subtraction (cont.)

- Verändert sich der Hintergrund muss das Verfahren neu initialisiert werden
- Dies kann mit Hilfe eines gleitenden Hintergrunds umgangen werden
- Objekte die als Vordergrund klassifiziert werden, sich aber nicht bewegen, werden nach einiger Zeit zum Hintergrundmodell hinzugefügt
- Background Subtraction wird häufig auch für das Tracking von Objekten in der Bildverarbeitung verwendet

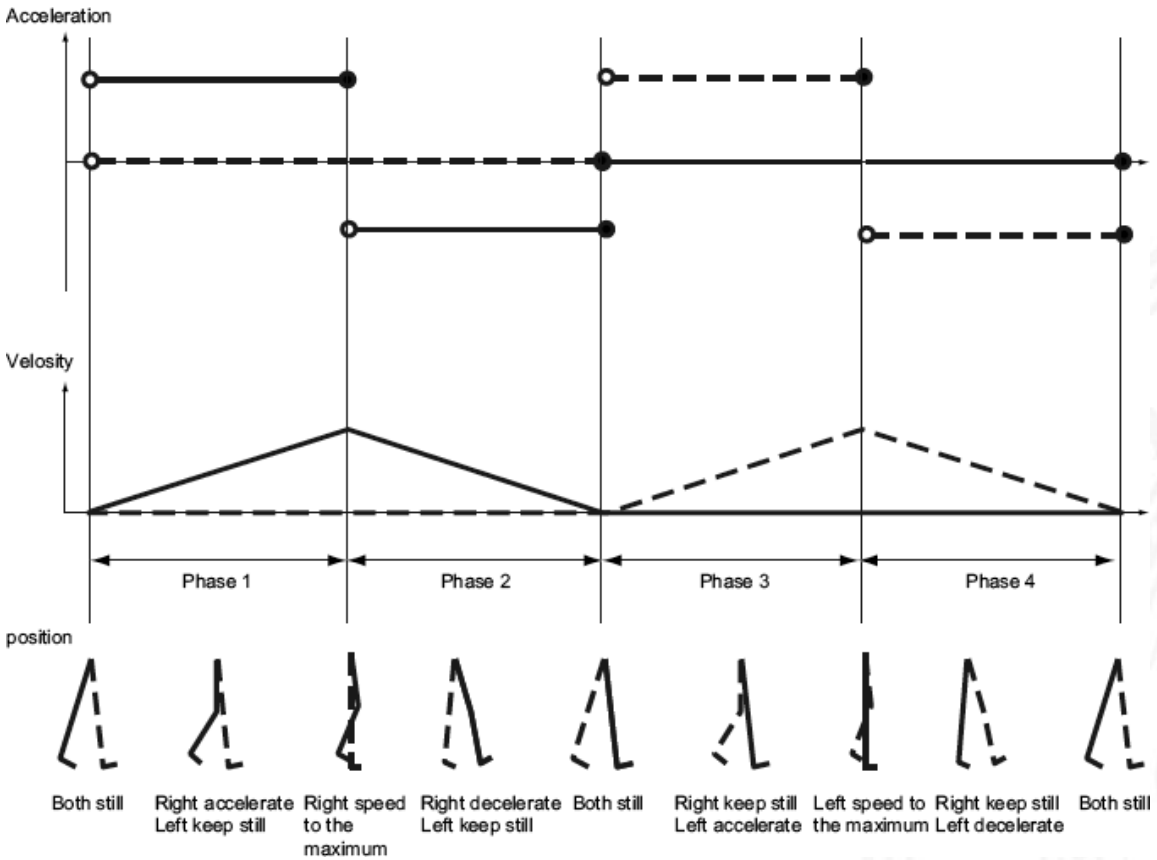
# Gruppierung der Vordergrund-Gruppen

- Die Gruppierung erfolgt ähnlich wie bei der Extraktion der Linien
- In der Praxis sind die Lasermesssysteme of nahe des Bodens montiert
- Daher sind meist nur die Beine von Personen sichtbar
- Vordergrund-Gruppen der Durchmesser größer als 20-30 cm ist, können für die Personen-Detektion ignoriert werden
- Mittels der Hough-Transformation wäre es möglich auch nach (Halb-)Kreisen zu suchen
- Je nach Abstand vom Laserscanner haben die Vordergrundgruppen unterschiedlich viele Messpunkte

# Tracking

- **Schwierigkeit:** Welches Paar Beine gehört zusammen?
- Durch Verdeckung sind nicht immer beide Beine zu sehen
- Meist steht ein Bein während das andere sich bewegt
- In der Literatur werde komplexe Bewegungsmodelle vorgestellt
- Die Verfolgung in konsekutiven Scans erfolgt dann zumeist über Kalman- oder Partikel-Filter und ähnliche Verfahren

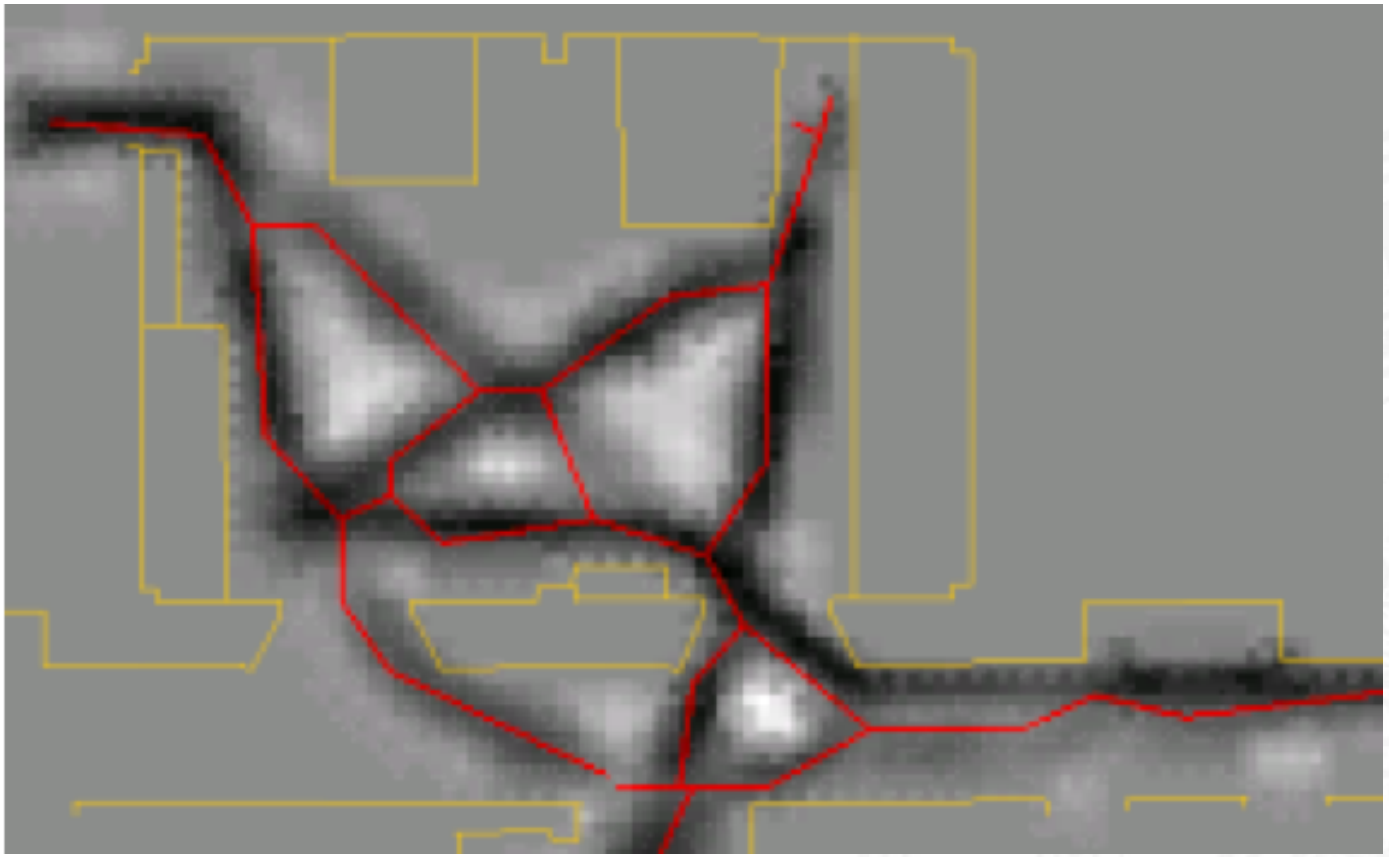
# Tracking (cont.)



# Lernen eines Bewegungsgraphen

- Werden in einem größeren Bereich über einen längeren Zeitraum Bewegungen verfolgt, kann man die Häufigkeit ermitteln, mit der bestimmte Bereiche begangen werden
- Aus dieser Häufigkeitsverteilung lässt sich ein Graph generieren der die Pfade beschreibt die gegangen werden
- An den Knotenpunkten können Wahrscheinlichkeiten ermittelt werden, in welche Richtung abgebogen wird
- Dadurch lassen sich Vorhersagen machen, wohin eine Person geht

# Lernen eines Bewegungsgraphen (cont.)



[Gut00] J.-S. Gutmann:
*Robuste Navigation autonomer mobiler Systeme*.
Universität Freiburg, Diss., 2000. –
Kapitel 3, Seite 21-58. –
URL `www.informatik.uni-freiburg.de/~gutmann/papers/thesis-steffen.ps.gz`

[NMTS] Viet Nguyen, Agostino Martinelli, Nicola Tomatis, Roland Siegwart:
A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics.
In: *Proceedings of the IEEE/RSJ Intenational Conference on Intelligent Robots and Systems, IROS 2005*.
Edmonton, Canada, . –

URL `asl.epfl.ch/index.html?content=epfl/publications.php`

[NZS+04] Katsuyuki Nakamura, Huijing Zhao, Ryosuke Shibasaki, Kiyoshi Sakamoto, Tomowo Ooga, Naoki Suzukawa:
Tracking Pedestrian by using Multiple Laser Range Scanners.
In: *Proceedings of the XXth ISPRS Congress*.
Istanbul, Turkey, 12–23 July 2004, S. 1260–1265

[Wes06] Martin Weser:
*Multimodales Tracking und Trajektorien-Vorhersage*.
Universität Hamburg, Department Informatik, TAMS, Diplomarbeit, 2006