64-544
Grundlagen der Signalverarbeitung und Robotik

http://tams.informatik.uni-hamburg.de/lectures/2011ss/vorlesung/GdSR

Jianwei Zhang

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik

Technische Aspekte Multimodaler Systeme

Sommersemester 2011

Gliederung

1. Einführung
2. Grundlagen der Robotik
3. Grundlagen der Sensorik
4. Scandaten verarbeiten
5. Rekursive Zustandsschätzung
6. Fuzzy-Logik
7. Steuerungsarchitekturen

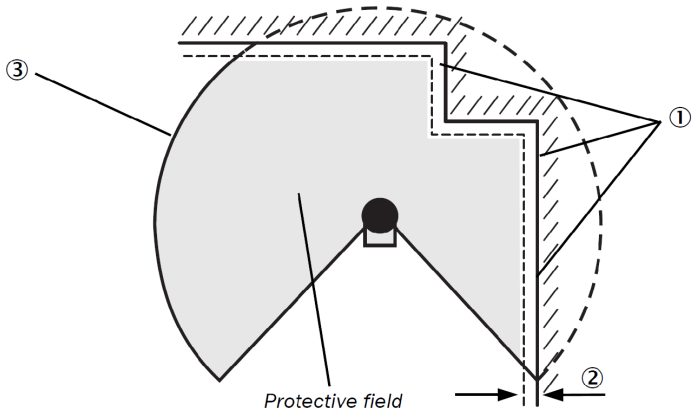
Gliederung

1. Einführung
2. Grundlagen der Robotik
3. Grundlagen der Sensorik
4. Scandaten verarbeiten
   - Filtern von Scandaten
   - Merkmalsextraktion
   - Anwendungen
   - Literatur
5. Rekursive Zustandsschätzung
6. Fuzzy-Logik
7. Steuerungsarchitekturen

Scandaten verarbeiten

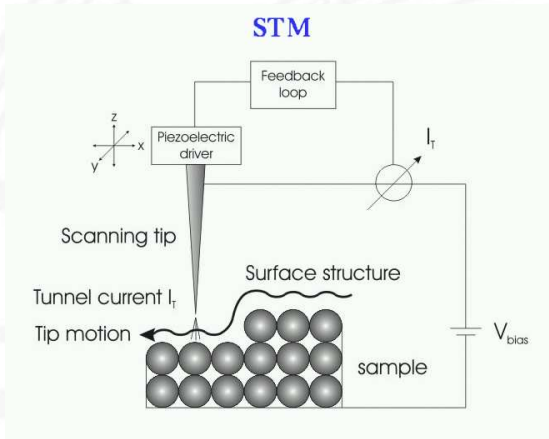
Makrorobotiker:

- z.B. 2D-Laserscan (Menge von Messwerten aus Winkel und Entfernung)



Nanotechnologe:

- Scanzeile eines Mikroskopes (Menge von Messwerten aus x-Koordinate und Höhe)



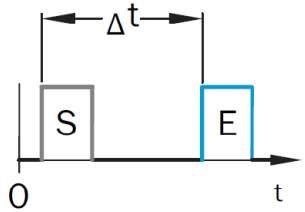
Verarbeitung von Abstandsmessungen

Für Messungen mit Lasermesssystemen gibt es Verfahren zur

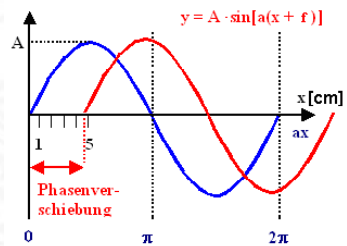
- Extraktion von Liniensegmenten
- Extraktion von Ecken
- Klassifikation von Scanpunkten
- Filterung von Scans:
  - Glätten
  - Datenreduktion

Exkurs: Laserscanner

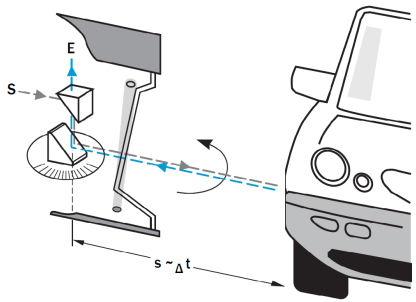
- Time of Flight
  - gepulster Laserstrahl wird ausgesendet
  - Zeit zwischen Aussenden und Empfang des Laserimpulses ist direkt proportional zur Entfernung



- Phasenmessung
  - Phasenverschiebung zwischen Referenzstrahl und empfangenem reflektierten Signal wird gemessen
  - Problem: falls Phasenverschiebung $\geq 360°$



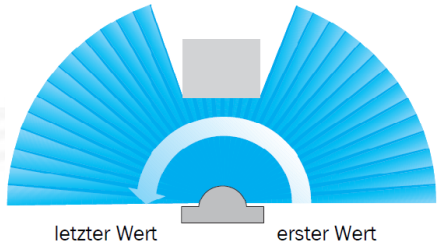
- interner Drehspiegel lenkt Laserstrahl ab (Laser Radar)



Scan

- Ein Laserscan ist eine Menge von Messwerten

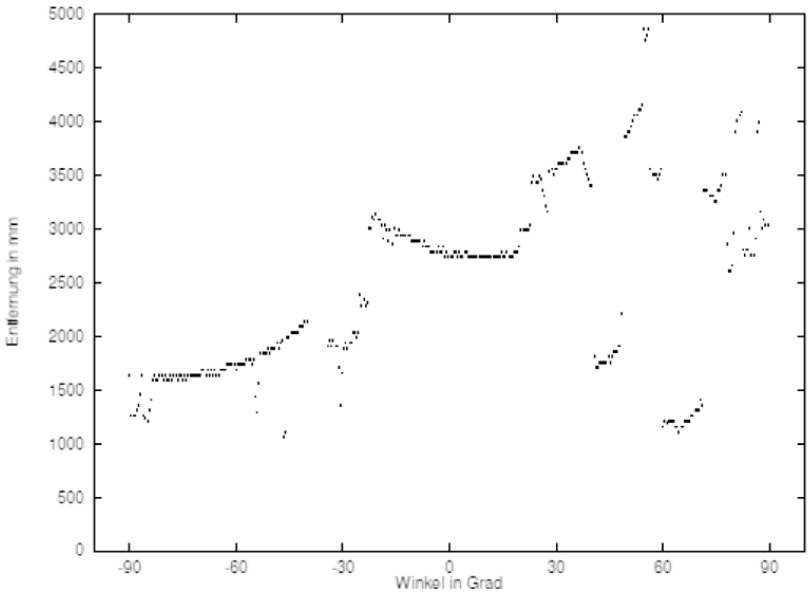
$$\left\{ m_i = (\alpha_i, r_i)^T | i = 0 \dots n-1 \right\}$$

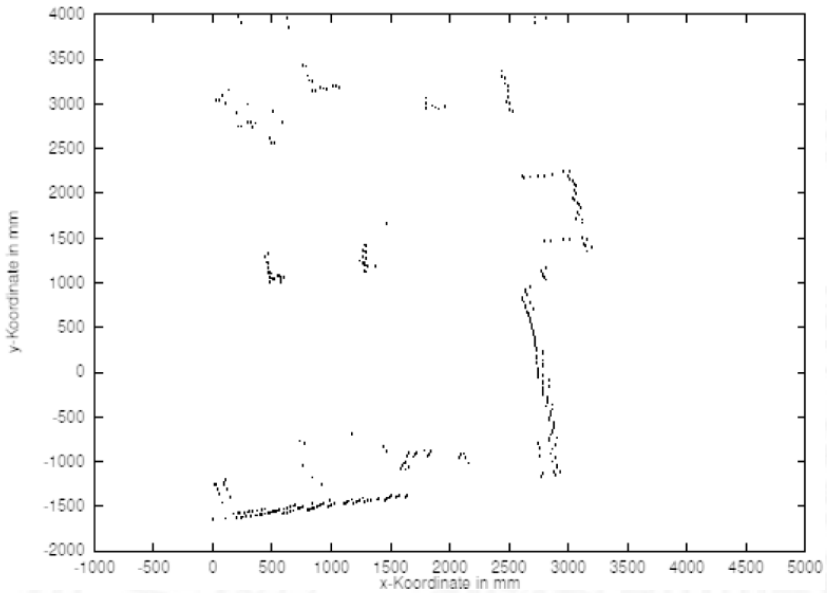


  welche in Polarkoordinaten $(\alpha_i, r_i)^T$ angegeben sind
- Ein Scanpunkt $m_i = (\alpha_i, r_i)^T$ kann für eine gegebene Aufnahmeposition $l = (x, y, \theta)^T$ in absolute Koordinaten umgerechnet werden

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

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
  - Linienfilter

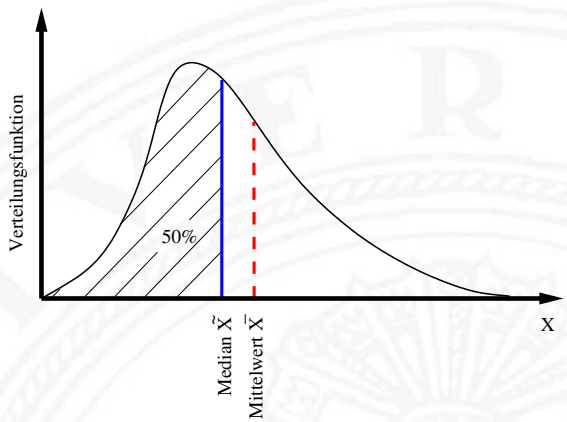
Medianfilter

- 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 der Fenstergröße bedeutet eine starke Glättung
- Nachteil des Medianfilters: Ecken werden abgerundet

Medianfilter (cont.)

Exkurs: Median

- Median (auch Zentralwert); Grenze zwischen zwei Werten
- in der Statistik:
  - stellt den Wert dar, der eine Verteilung in zwei gleich große Teile teilt
  - Median $\tilde{X}$ einer diskreten Stichprobe:
    - höchstens die Hälfte der Werte $< \tilde{X}$ und höchstens die Hälfte der Werte $> \tilde{X}$
    - für eine geordnete Stichprobe $\{x_1, x_2, x_3, ..., x_n\}$ mit n Messwerten gilt:



$$\tilde{X} = \begin{cases} x_{\frac{n+1}{2}} & n \text{ ungerade} \\ \frac{1}{2}(x_{\frac{n}{2}} + x_{\frac{n}{2}+1}) & n \text{ gerade} \end{cases}$$

Medianfilter (cont.)

Algorithmus Medianfilter

Eingabe Scan s, Fenstergröße $wSize$

Ausgabe Scan s'

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

s, s': Feld von Messwerten;

Messwert: Tupel aus (angle, distance);

numpoints(s): berechnet number of points von s;

scanpoint(s, i): liefere Messwert i aus s;

distance-value(i): liefere distance aus Messpaar i;

angle-value(i): liefere Winkel aus Messpaar i;

median(d): berechne den Median der Werte des Feldes d;

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'

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

% beginne neue Punktewolke

% wenn Abweichung der Entfernungen $<$ ∅ füge p hinzu; summiere Winkel und Entfernungen separat

Reduktionsfilter (cont.)

```
        ...
    else
        scanpoint(s',j) := p_sum/n      % schreibe Schwerpunkt der Punkte-
                                          wolke als neuen Punkt nach s'
        j := j + 1
        p_0 := p                        % initialisiere neue Punktewolke
        p_sum := p_0
        n := 1
    endif
endfor
scanpoint(s',j) := p_sum/n              % schreibe Schwerpunkt der letzten
                                          Punktewolke nach s'
j := j + 1
numpoints(s') := j                      % übergebe neue Anzahl der Punkte
return s'
```

Reduktionsfilter (cont.)

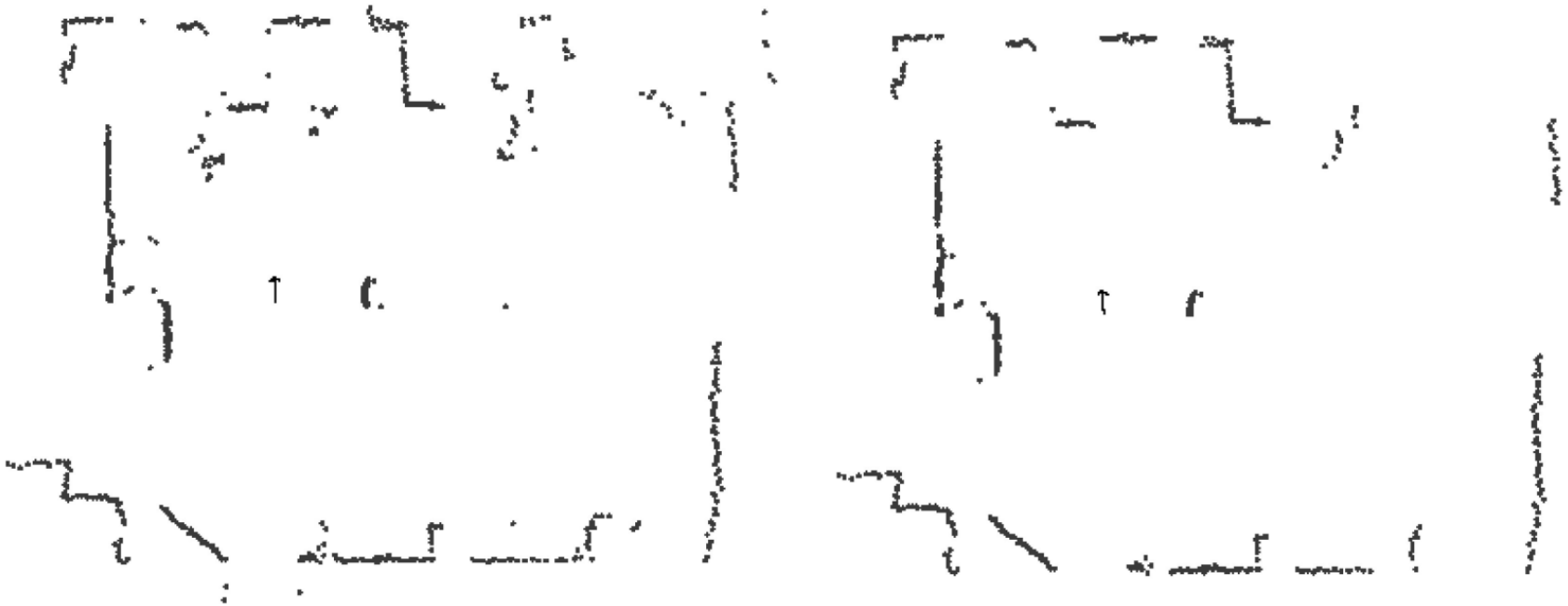
Reduktionsfilter (cont.)

- 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

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

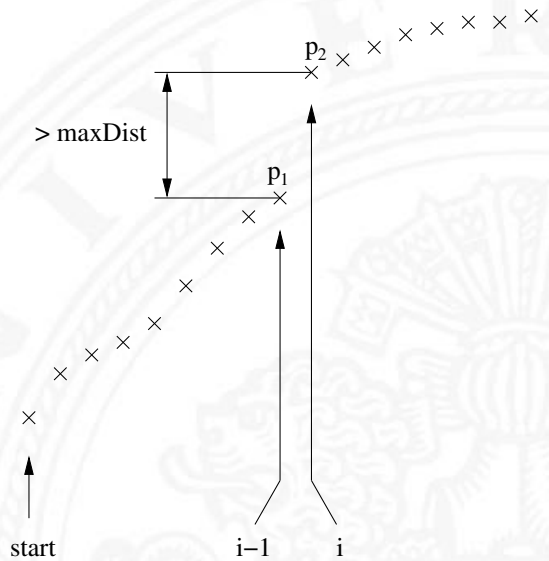
Linien

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 := scanpoint(s,i-1)
  p_2 := 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** % Ausgleichsgerade durch *start* u. *end* berechnen
  if σ(*line*) $<$ *maxSigma* **then** % Ausgleichsgerade erfüllt Längenkriterium?
    $l := l \cup \{line\}$ % Ausgleichsgerade erfüllt Abweichungskriterium?
  else $/ * teilen * /$ Fkt. σ(*line*) berechnet Abweichung der Ausgleichsgeraden vom tatsächlichen Kurvenzug zwischen *start* u. *end*
    p_{start} := scanpoint(s,*start*)
    p_{end} := scanpoint(s,*end*)
    i_{split} := *start*
    $d := 0$

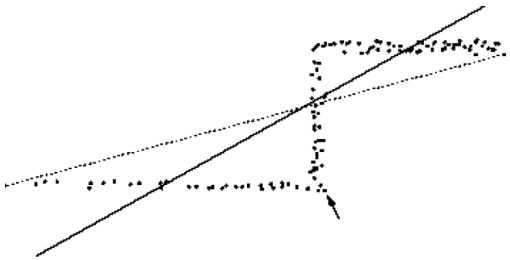
Linien (cont.)

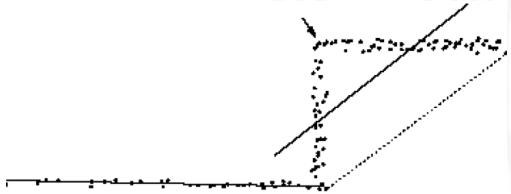
```
    for i := start+1 to end-1 do
      p := scanpoint(s,i)
      if dist-to-line(p,p_start,p_end) > d then
        i_split := i
        d := dist-to-line(p,p_start,p_end)
      endif
    endfor
    l := l ∪ split(s,start,i_split)
    l := l ∪ split(s,i_split,end)
  endif
endif
return l
```

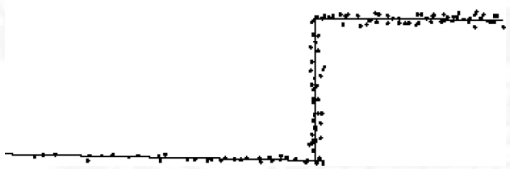
% suche Punkt i_{split} der größten Abweichung zur Ausgleichsgeraden

% Fkt. dist-to-line berechnet Abstand des Punktes p zur Ausgleichsgeraden

% teile bei i_{split} und wende *split* auf linken und rechten Kurvenzug an





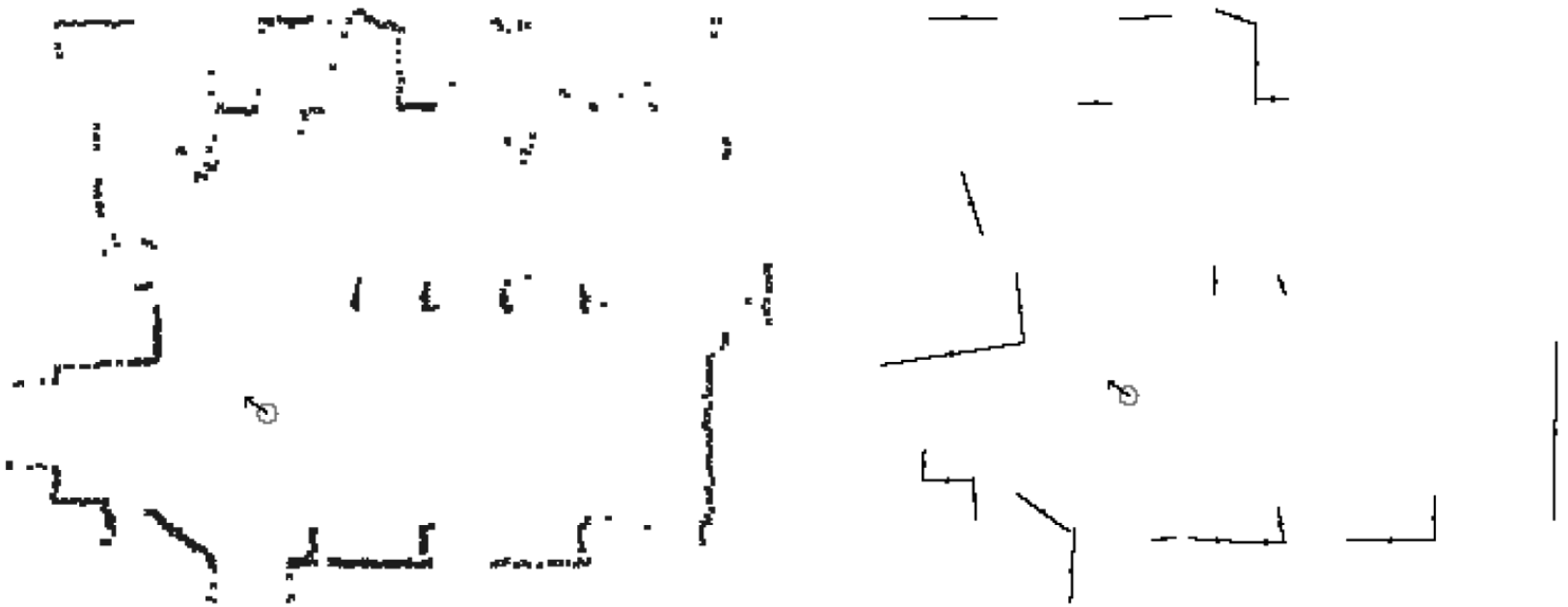


Linien (cont.)

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

Linien (cont.)

- Zeitkomplexität ähnlich *Quicksort*: $O(n^2)$ im schlechtesten, $O(n \log n)$ im mittleren Fall (n: Anzahl der Scanpunkte)



Ecken

- Ähnlich wie Linienalgoritmus
- Aufeinander folgende Linien werden miteinander geschnitten
- Lediglich Ersetzen der Funktion *split* durch die Funktion *corner* notwendig
- Gleiche Zeitkomplexität

Algorithmus corner(s,$start$,end)

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

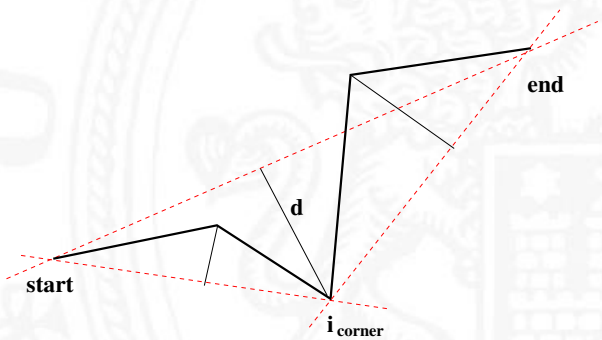
Ausgabe Menge von Ecken e

Ecken (cont.)

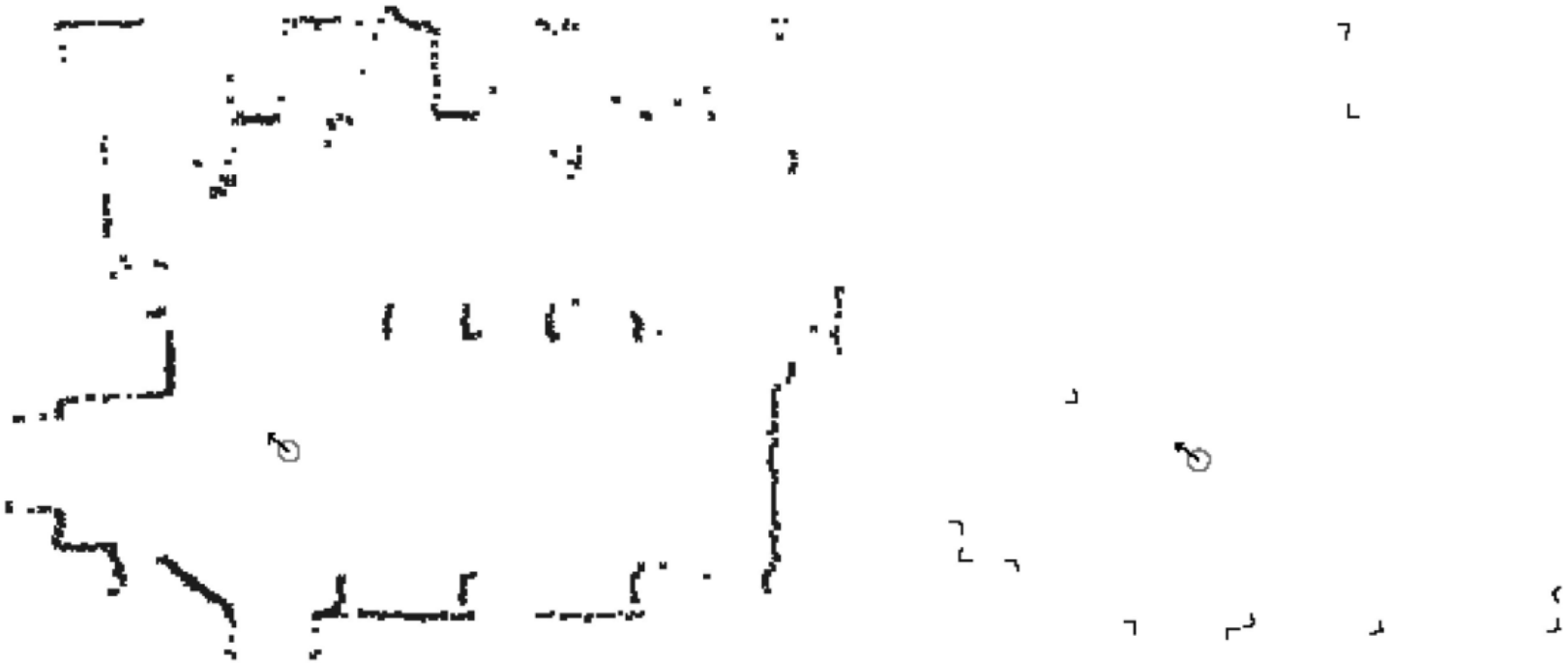
e := empty
if $(end\ -start) \geq 2 \cdot minPointsCorner$ **then** % noch mind. zwei Ecken möglich?
  p_{start} := scanpoint(s,$start$)
  p_{end} := scanpoint(s,end)
  i_{corner} := $start$
  d := 0
  for i := $start$+1 **to** end-1 **do** % Punkt mit größtem Abstand suchen
    p := scanpoint(s,i)
    if distance-to-line(p,p_{start},p_{end}) > d **then**
      i_{corner} := i
      d := distance-to-line(p,p_{start},p_{end})
    endif
  endfor

Ecken (cont.)

if $(i_{corner}$ - $start) \geq minPointsCorner$ **and**
$(end$ - $i_{corner}) \geq minPointsCorner$ **then**
$corner\_line_1 :=$
make-line$(s, i_{corner}$ - $minPointsCorner, i_{corner})$ % generiere Ausgleichsgerade durch *corner* und $(corner - mPC)$
$corner\_line_2 :=$
make-line$(s, i_{corner}, i_{corner} + minPointsCorner)$ % generiere Ausgleichsgerade durch *corner* und $(corner + mPC)$
if $\sigma(corner\_line_1) < maxSigma$ **and**
$\sigma(corner\_line_2) < maxSigma$ **then**
$e := e \cup \{$make-corner$(corner_line_1, corner_line_2)\}$ % falls beide *corner_lines* Qualitätskriterium erfüllen, generiere Ecke
endif
endif
$e := e \cup$ corner$(s, start, i_{corner})$
$e := e \cup$ corner(s, i_{corner}, end)
endif
return e



Ecken (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
- Mögliche 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

Exkurs: Geradengleichungen

Normalform

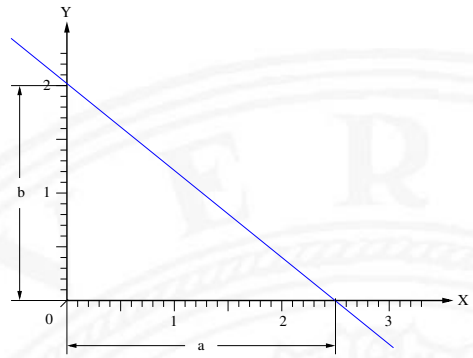
$$y = mx + b$$

Beispielgerade: $y = -\frac{4}{5}x + 2$

Achsenabschnittsgleichung

$$\frac{x}{a} + \frac{y}{b} = 1$$

Beispielgerade: $\frac{2x}{5} + \frac{y}{2} = 1$



Allg. Geradengleichung

$$Ax + By + C = 0$$

Beispielgerade:

$$\frac{2}{5}x + \frac{1}{2}y - 1 = 0 \text{ oder}$$
$$2x + \frac{5}{2}y - 5 = 0 \text{ oder}$$
$$-2x - \frac{5}{2}y + 5 = 5 \text{ oder } \ldots$$

$$Ax + By + C = 0$$
$$Ax + By = -C$$
$$\frac{A}{-C}x + \frac{B}{-C}y = 1$$
$$\frac{x}{a} + \frac{y}{b} = 1$$
$$\text{mit } a = \frac{-C}{A};$$
$$b = \frac{-C}{B}$$

Exkurs: Geradengleichungen (cont.)

Hessesche Normalform

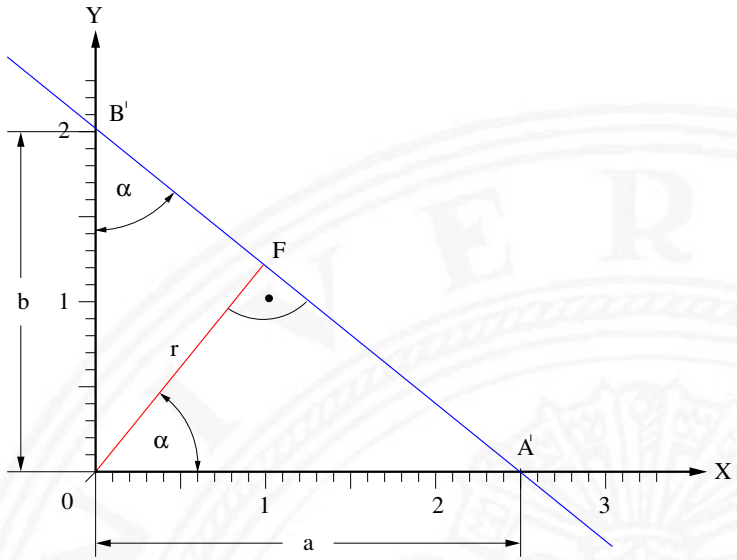
$$x\cos\alpha + y\sin\alpha - r = 0$$

$$\begin{aligned} Ax + By + C &= 0 \quad | -C \\ -\tfrac{A}{C}x - \tfrac{B}{C}y - 1 &= 0 \quad | \; a = -\tfrac{C}{A}, \; b = -\tfrac{C}{B} \\ \tfrac{1}{a}x + \tfrac{1}{b}y - 1 &= 0 \quad | \cdot r \end{aligned}$$

$$x\cos\alpha + y\sin\alpha - r = 0$$

Beispielgerade:

$$\alpha = \tan^{-1}\left(\tfrac{B}{A}\right) = 38,7°$$
$$r = a\cos\alpha = 1,95$$



$$a = \overline{0A'} = -\frac{C}{A}$$
$$b = \overline{0B'} = -\frac{C}{B}$$

$$cos\alpha = \frac{r}{a}$$
$$sin\alpha = \frac{r}{b}$$

Geradenerkennung

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

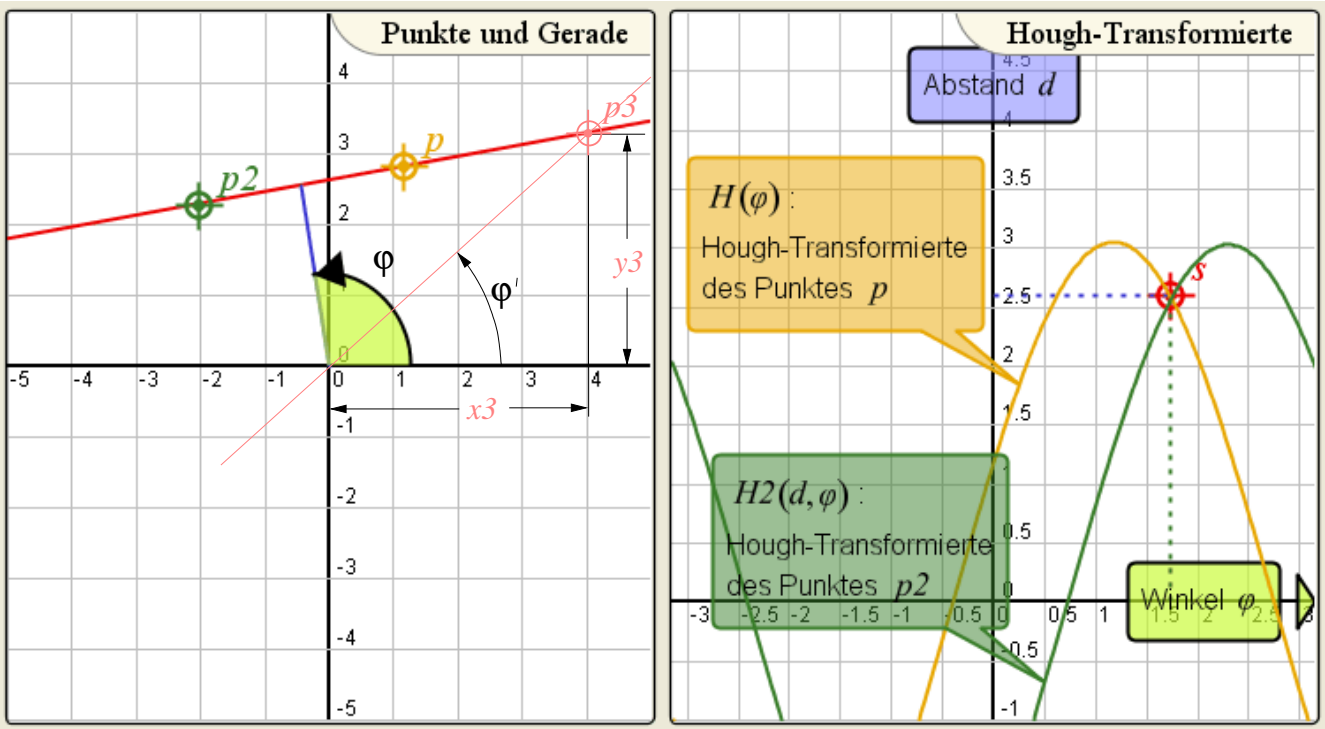
- **Besser:** Gerade in Hessescher Normalform

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

mit

- α: Winkel zwischen der x-Achse und der Normalen der Geraden; Vollkreiswerte möglich
- r: Abstand vom Ursprung zur Geraden; auch negative Werte mölgich, z. B. positiv, falls r in Richtung der Normalen und negativ, falls r entgegen der Richtung der Normalen

Geradenerkennung (cont.)



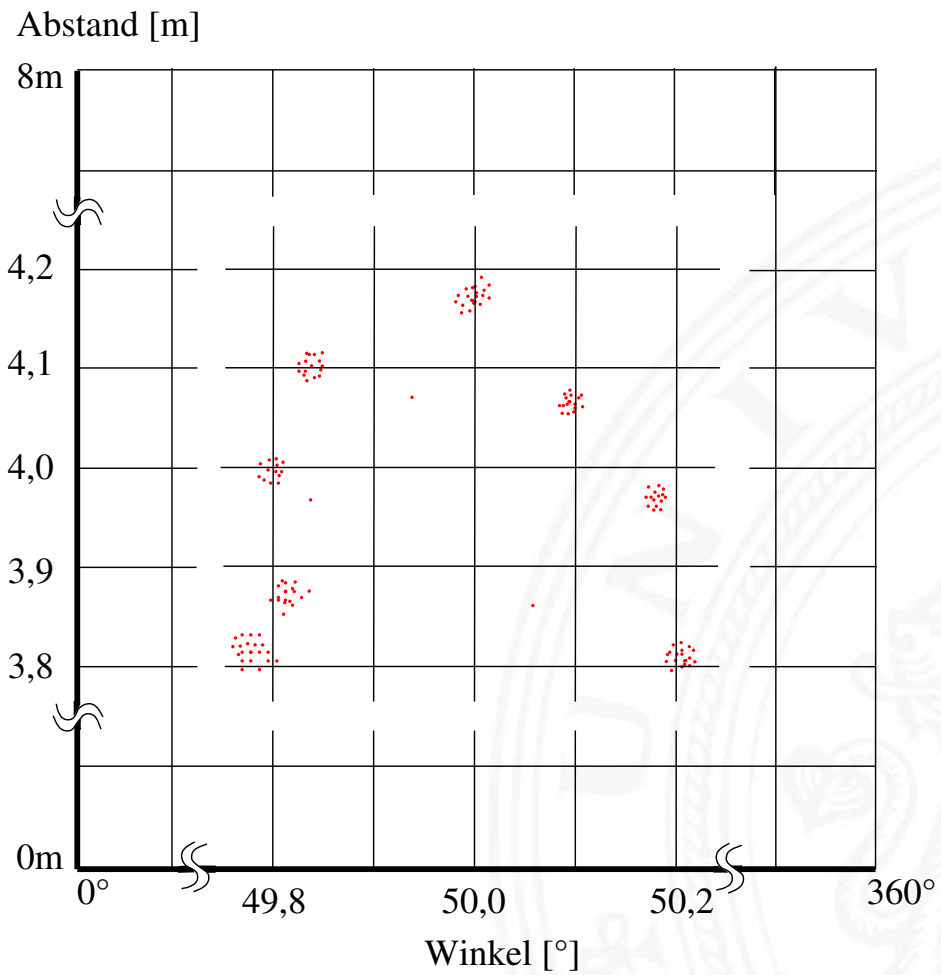
$$y = A \cdot sin(\omega t + \varphi)$$

$$\varphi = arctan(\frac{y_i}{x_i})$$
$$y_i = A \cdot sin\varphi$$
$$A = \frac{y_i}{sin\varphi}$$

Geradenerkennung (cont.)

- Alle Linien-Segmente aus der Linienerkennung lassen sich in Hessescher Normalform darstellen
- Für jedes Linien-Segment wird im Parameterraum ein θ-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

Geradenerkennung (cont.)



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

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

- Complete Linkage Clustering: Maximaler Abstand zweier Elemente

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

- Average Linkage Clustering: Durchschn. Abstand aller Elemente

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

Typische Distanzfunktionen (cont.)

- Centroid Method: Abstand der Mittelwerte

$$D_{CM} = d(\bar{x}, \bar{y})$$

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

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

- und zahlreiche weitere Funktionen...

Tracking auf Scandaten

- Informationen über Bewegungsverläufe (engl. *Tracking*) können aus Scandaten gewonnen werden
- Übliche Verfahren arbeiten in drei Schritten:
  1. Unterteilen der Messdaten in Vordergrund und Hintergrund durch Background Subtraction
  2. Gruppieren der Vordergrund-Daten
  3. Verfolgen der Gruppen in konsekutiven Scans

Background Subtraction

- Zuerst wird ein Hintergrundmodell erstellt:
  - Zu jedem Winkel bzw. Koordinate (je nach Datenmodell) und im folgenden Messpunkt genannt wird für einen bestimmten Zeitraum ein Histogramm der Entfernungsmessungen ermittelt
  - Das Histogramm für jeden Messpunkt α wird durch eine Verteilungsfunktion beschrieben (meistens: Gauß-Verteilung mit Mittelwert μ_α und Standardabweichung σ_α)
- 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 und Tracking

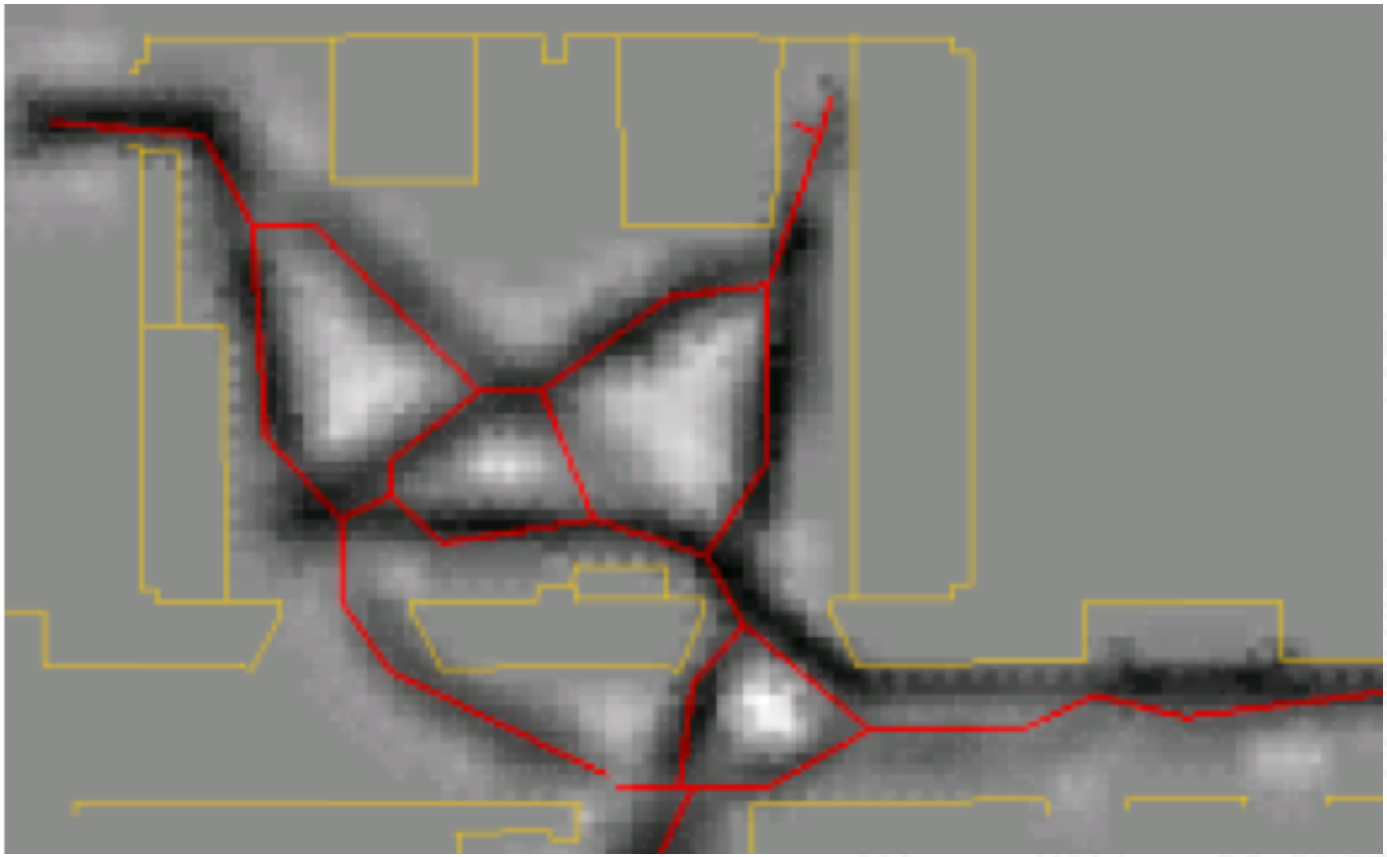
- Die Gruppierung erfolgt ähnlich wie bei der Linien-Extraction
- Ausnutzen von Hintergrundinformationen bzw. bekannte Eigenschaften der Vordergrundgruppen
- Mittels der Hough-Transformation wäre es möglich nach best. Formen zu suchen

- Die Verfolgung in konsekutiven Scans erfolgt dann zumeist über Kalman- oder Partikel-Filter und ähnliche Verfahren

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 beschritten werden
- An den Knotenpunkten können Wahrscheinlichkeiten ermittelt werden, in welche Richtung abgebogen wird
- Dadurch lassen sich Vorhersagen über Bewegungsabläufe machen

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