64-544
Grundlagen der Signalverarbeitung und Robotik

http://tams.informatik.uni-hamburg.de/lectures/2012ss/vorlesung/GdSR

Jianwei Zhang

Universität Hamburg
Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Technische Aspekte Multimodaler Systeme

Sommersemester 2012

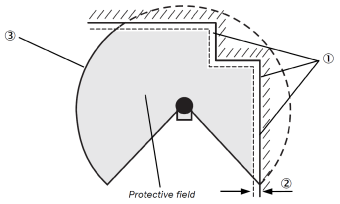
Gliederung

Agenda

Scandaten

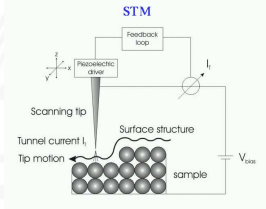
Makrorobotiker:

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



Nanotechnologe:

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



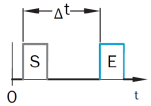
Verarbeitung von Abstandsmessungen

Für Messungen mit Lasermesssystemen gibt es Verfahren zur

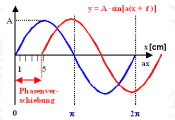
- Extraktion von Liniensegmenten
- Extraktion von Ecken
- Klassifikation von Scanpunkten
- Filterung von Scans:
  - Glätten
  - Datenreduktion

Exkurs: Laserscanner

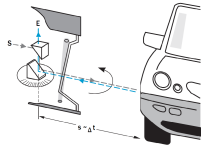
- Time of Flight
  - gepulster Laserstrahl wird ausgesendet
  - Zeit zwischen Aussenden und Empfang des Laserimpulses ist direkt proportional zur Entfernung



- 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

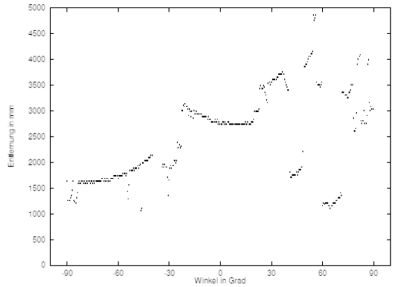
$$\left\{ m_i = (\alpha_i, r_i)^T | i = 0 \ldots n-1 \right\}$$

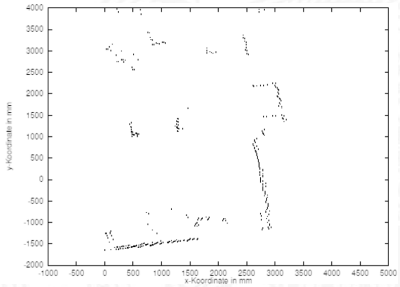
  welche in Polarkoordinaten $(\alpha_i, r_i)^T$ angegeben sind
- Ein Scanpunkt $m_i = (\alpha_i, r_i)^T$ kann für eine gegebene Aufnahmeposition $l = (x, y, \theta)^T$ in absolute Koordinaten umgerechnet werden

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

Scan (cont.)

- Polarkoordinaten (α_i, r_i)
- Koordinatensystem des Scanners
- absolute Koordinaten
- Aufnahmeposition $l = (x, y, \theta)^T$ berücksichtigt





Filtern von Scandaten

- Probleme von Scandaten:
  große Datenmenge, ungewünschte Scanpunkte
- daher oftmals vorab Filterung von Scandaten
- gängige Filter:
  - Medianfilter
  - Reduktionsfilter
  - Linienfilter

Medianfilter

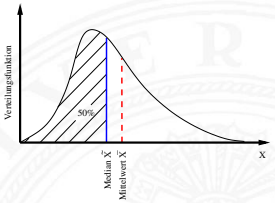
- Der Medianfilter erkennt Ausreißer und ersetzt diese durch eine geeignete Messung.
- Um jeden Scanpunkt wird ein Fenster gelegt, das die Messungen vor und nach dem Punkt enthält.
- Der Scanpunkt wird ersetzt durch einen Punkt, der denselben Aufnahmewinkel, aber den Median der Entfernungsmessungen des betrachteten Fensters als Entfernung hat.
- *wSize* bestimmt die Fenstergröße
  (Anzahl der Punkte im Medianfilter)
- große Fenstergröße $\longrightarrow$ 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.)

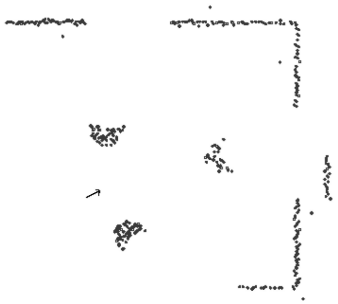
Algorithmus Medianfilter
Eingabe Scan s, Fenstergröße *wSize*
Ausgabe Scan s'

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.)

Orginaldaten



mediangefilterte Daten

(Darstellung in absoluten Koordinaten)

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 < ∅ (euklidischer Abstand) füge p hinzu; summiere Winkel und Entfernungen separat

Reduktionsfilter (cont.)

```
        ...
    else
        scanpoint(s',j) := p_sum/n
        j := j + 1
        p_0 := p
        p_sum := p_0
        n := 1
    endif
endfor
scanpoint(s',j) := p_sum/n
j := j + 1
numpoints(s') := j
return s'
```

% schreibe Schwerpunkt der Punktewolke als neuen Punkt nach s'
% initialisiere neue Punktewolke

% schreibe Schwerpunkt der letzten Punktewolke nach s'
% übergebe neue Anzahl der Punkte

Reduktionsfilter (cont.)

Orginaldaten

Punktwolken

(Darstellung in absoluten Koordinaten)

Reduktionsfilter (cont.)

- Zeitkomplexität des Reduktionsfilters: $O(n)$, mit n Anzahl der Punkte
- Vorteile des Reduktionsfilters:
  - Reduzierung der Anzahl der Scanpunkte ohne Verlust wesentlicher Informationen
  - führt zu kürzeren Laufzeiten bei der Nachbearbeitung eines Scans
  - ergibt eine bessere Gleichverteilung der Punkte
- Nachteile des Reduktionsfilters:
  - Extraktion von Merkmalen nicht mehr so einfach
  - möglicherweise zu wenig Punkte für ein Merkmal
  - daher besser: Merkmalsextraktion vor Reduktionsfilter

Linienfilter

- Linienfilter nutzt das später vorgestellte Verfahren zur Linienextraktion
- Scanpunkte, die keinem Liniensegment zugeordnet wurden, werden entfernt
- 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.)

Orginaldaten

liniengefiltert

(Darstellung in absoluten Koordinaten)

Merkmalsextraktion

- keine Verarbeitung von kompletten Scans, sondern Merkmalsextraktion
- häufige Merkmale: Linien, Ecken

- Beispiel: Linienextraktion
  - *maxDist* ist der maximal zulässige Abstand zweier aufeinander folgender Punkte für die Gruppierung

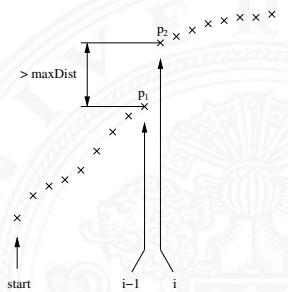
Linien

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($line$) ≥ *minPointsOnLine* **then**
  if σ($line$) < *maxSigma* **then**
    l := l ∪ {$line$}
  else /* *teilen* */
    p_{start} := scanpoint(s,$start$)
    p_{end} := scanpoint(s,end)
    i_{split} := $start$
    d := 0

% Ausgleichsgerade durch *start* u. *end* berechnen
% Ausgleichsgerade erfüllt Längenkriterium?
% Ausgleichsgerade erfüllt Abweichungskriterium?
Fkt. σ($line$) berechnet Abweichung der Ausgleichsgeraden vom tatsächlichen Kurvenzug zwischen *start* u. *end*

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 Verbindungsgeraden(p_{start},p_{end})
% Fkt. dist-to-line berechnet Abstand des Punktes p zur Verbindungsgeraden

% 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 *start*, *end* gelegt
- falls Abweichung $\sigma(line)$ zu groß:
  - Aufteilung der Punktmenge
  - Aufruf von *split* für die neuen Mengen
  - Punkt an dem aufgeteilt wird, ist der Punkt mit dem größten Abstand zur Geraden durch *start* und *end*
- *minPointsOnLine* und *maxSigma* bestimmen Anzahl und Qualität der Linien
- 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*

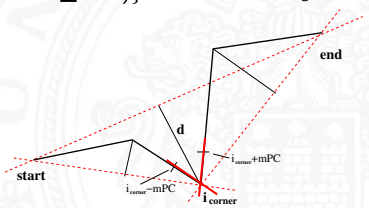
Ausgabe Menge von Ecken e

Ecken (cont.)

```
e := empty
if (end-start) ≥ 2·minPointsCorner then      % noch mind. zwei Ecken möglich?
  p_start := scanpoint(s,start)
  p_end := scanpoint(s,end)
  i_corner := start
  d := 0                                     % = 0 sinnvoll?
  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.)

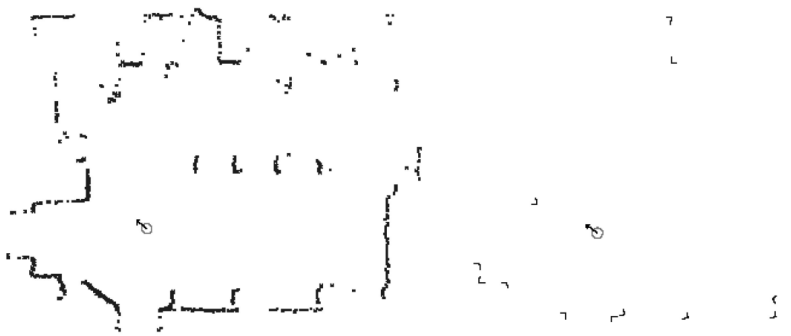
if $(i_{corner}$ - $start) \geq minPointsCorner$ **and**
$(end$ - $i_{corner}) \geq minPointsCorner$ **then**
$corner\_line_1 :=$
make-line(s,i_{corner} - $minPointsCorner$,i_{corner}) % generiere Ausgleichsgerade durch $corner$ und $(corner - mPC)$
$corner\_line_2 :=$
make-line(s,i_{corner}, $i_{corner}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



Ecken (cont.)

Orginaldaten

extrahierte Ecken



(Darstellung in absoluten Koordinaten)

Hough-Transformation

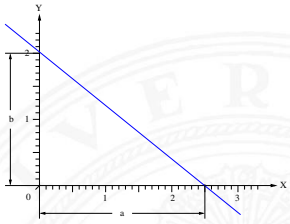
- Verfahren zur Erkennung von Merkmalen wie:
  - Geraden
  - Kreisen
  - sonstigen geometrischen Figuren ...
- in der Bildverarbeitung häufig angewandt auf Gradientenbildern
- duale Transformation vom Bild- in Parameterraum
- mögliche Parameter:
  - Gerade: Steigung und y-Achsen-Abschnitt
    (bei Wahl anderer Geradengleichungen, andere Parametersätze)
  - Kreis: Radius und Mittelpunkt
- Punkt im Bildraum entspricht einer Figur im Parameterraum
- gesuchtes Merkmal findet sich bei Häufungen im Parameterraum

Geradenerkennung (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: $\frac{2}{5}x + \frac{1}{2}y - 1 = 0$ oder
$2x + \frac{5}{2}y - 5 = 0$ oder
$-2x - \frac{5}{2}y + 5 = 5$ oder ...

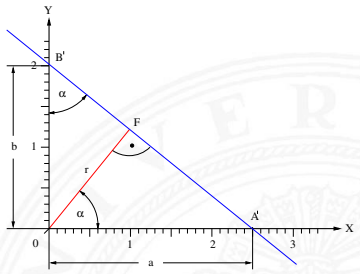
$$Ax + By + C = 0$$
$$Ax + By = -C$$
$$\frac{A}{-C}x + \frac{B}{-C}y = 1$$
$$\frac{x}{a} + \frac{y}{b} = 1 \text{ mit } a = \frac{-C}{A};\ b = \frac{-C}{B}$$

Geradenerkennung (Exkurs: Geradengleichungen) (cont.)

Hessesche Normalform

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

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



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

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

Beispielgerade:

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

Geradenerkennung

- **Parameter:** Steigung und y-Achsen-Abschnitt
- **Nachteil:** Geraden mit unendlicher Steigung können nicht abgebildet werden
- **besser:** Gerade in Hessescher Normalform

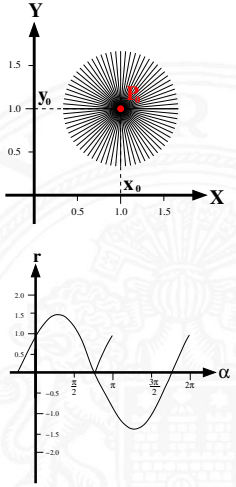
$$r = x \cdot \cos(\alpha) + y \cdot \sin(\alpha)$$

mit

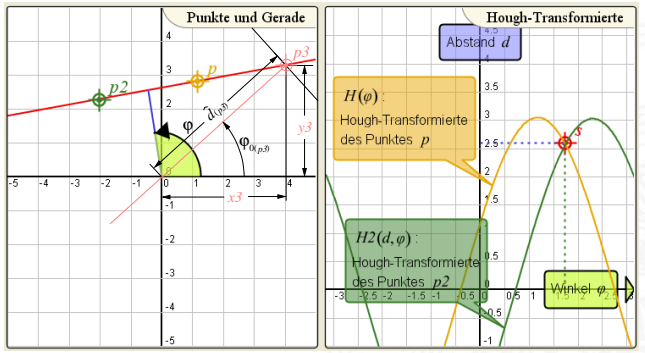
- α: Winkel zwischen der x-Achse und der Normalen der Geraden; Vollkreiswerte möglich oder Beschränkung auf $[0, \pi[$
- r: Abstand vom Ursprung zur Geraden; falls Vollkreiswerte, negative Werte erforderlich, z. B. positiv, falls r in Richtung der Normalen und negativ, falls r entgegen der Richtung der Normalen

Geradenerkennung (cont.)

- Annahme:
  - Punktemenge im Parameterraum $\{(\alpha_0, r_0) \ldots (\alpha_n, r_n)\}$ *mit* $n \to \infty$
  - alle genügen der Geradengleichung $r = x_0 \cdot \cos(\alpha) + y_0 \cdot \sin(\alpha)$
- alle Punkte entsprechen jeweils einer Geraden im x-y Raum, die durch den Punkt (x_0, y_0) gehen
- ein Punkt im x-y Raum entspricht im Parameterraum einer Sinuskurve
- ein Punkt im Parameterraum entspricht im x-y Raum einer Geraden



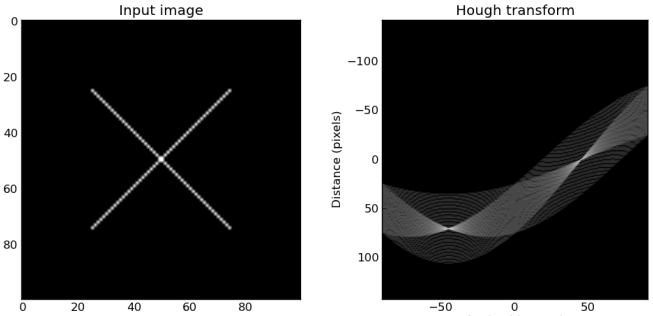
Geradenerkennung auf Punktmengen



$$\varphi_0 = arctan(\tfrac{y_i}{x_i})$$
$$\hat{d} = \frac{y_i}{sin\varphi_0}$$

$$d = \hat{d} \cdot cos(\omega t + \varphi_0)$$

Geradenerkennung auf Punktmengen (cont.)



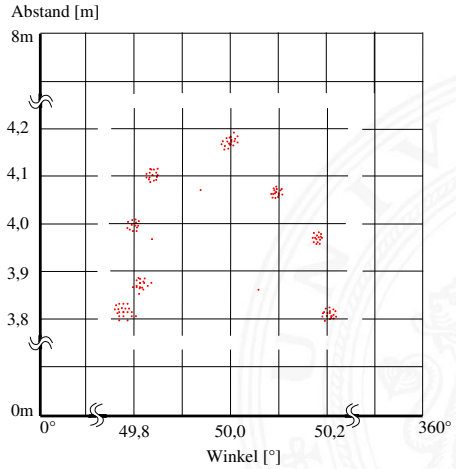
Quelle: http://scikits-image.org

- Parameterraum (Dualraum) aufgespannt durch (φ, d)
- Dualraum repräsentiert als Matrix
- Geradengleichung mit (φ_i, d_i) inkrementiert Zähler der Zelle φ_i, d_i

Geradenerkennung auf Liniensegmenten

- Linien-Segmente z. B. aus der Linienerkennung lassen sich in Hessescher Normalform darstellen
- jedem Linien-Segment wird im Parameterraum ein α-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 auf Liniensegmenten (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. zueinander am nächsten liegenden Cluster werden zusammengefasst
3. wiederhole Schritt 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.

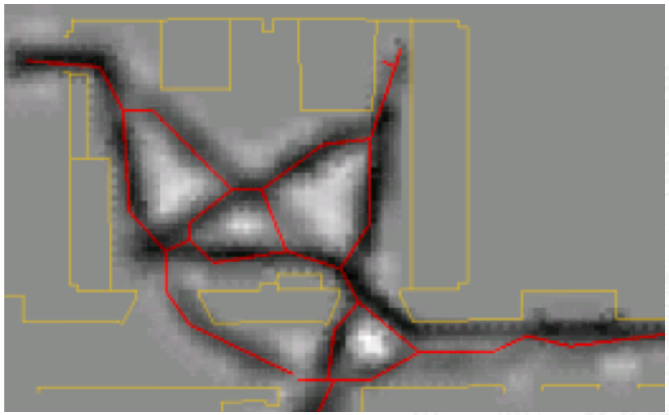
Gruppierung und Tracking

- 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.)



Praxisbeispiel: Verarbeitung von STM-Scandaten

Aufgabe: Wie lässt sich der gleiche Punkt auf der Oberfläche mehrfach ansteuern?

- Voraussetzung für autonome Manipulation
- Position der Spitze ist nicht direkt kontrollierbar/messbar
- Regelspannung des Z-Piezos liefert Informationen über Topographie
- aber: tatsächliche Spitzenposition ist unsicher wegen
  - Temperaturdrift
  - Piezoartefaktefakte
    - Hysterese
    - Kriechen (creep)

Übertragene Aufgabe: Wie lassen sich in Scans etwa der gleichen Region identische Punkte zuordnen?

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Lösung des Zuordnungsproblems

Lokalisierung identischer Punkte in verschiedenen Scans möglich:

- Ermittlung der Parameter möglich für:
  - Temperaturdrift
  - Piezoartefakte
  - veränderte Spitzengeometrie

  erhöht Wahrscheinlichkeit für korrekte Positionierung
- Identifizierung von Ausschnitten
- Zusammenfügen von Teilscans

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Lösungsansatz

Bereitstellung elementarer Lokalisierungsroutinen

- Ermittlung und Einsatz “natürlicher Landmarken“
- Berechung der Transformationen zwischen den Scans

Herausforderungen an die Lokalisation:

- unterschiedliche Vergrößerungen oder Auflösungen möglich
- Oberfläche kann sich partiell verändert haben
- elektrische Eigenschaften der Piezos führen zu Artefakten in der Bildgebung

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

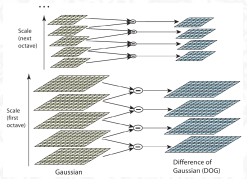
Scale Invariant Feature Transform (SIFT)-Algorithm

Scale Invariant Feature Transform (SIFT)

- eingeführt 1999 durch D. Lowe
- Ermittlung und Auswahl sog. "*keypoints*" eines Bildes
- Beschreibung eines Keypoints durch 128 Werte (Häufigkeitsverteilung metrisch skalierbarer Merkmale)
- invariant zu Abbildungsmaßstab, Translation und Rotation
- Keypoints lassen sich durch Berechung des quadratischen Abstandes ihrer Merkmale vergleichen

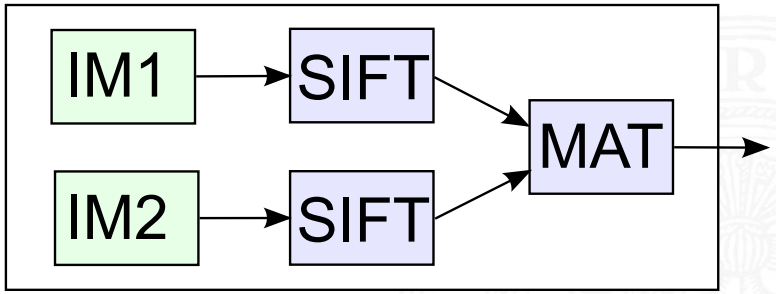
SIFT-feature beliebt in Computer/Robot-Vision:

- Object Erkennung
- Lokalisation
- Tracking



Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Elementares Vorgehen



- Features werden für jedes Bild unabhängig ermittelt
- Feature der Listen werden verglichen (Feature Matching)

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Feature Matching

- finde die beste Entsprechung eines jeden Features im anderen Bild
- teste, ob gefundene Entsprechung eindeutig (zur Liste der Matches hinzufügen/entfernen)
- wende RANSAC (Random Sample Consensus) auf die Liste der Matches an
  - iteratives Verfahren
  - ermittle Ausreißer
  - entferne Ausreißer
- geringfügige Modifikationen beim Feature Matching auf STM-Scans erforderlich

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

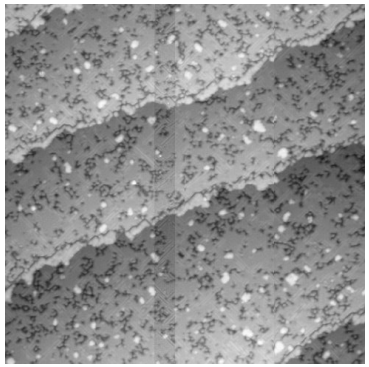
Exkurs: RANSAC

Random Sample Consensus (RANSAC)

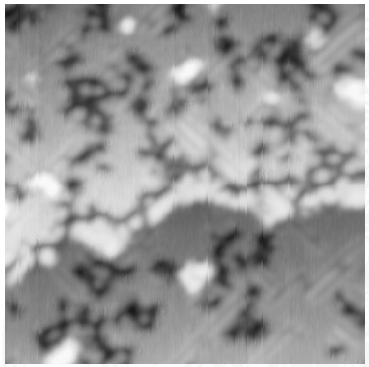
- vorgestellt 1981 von Martin A. Fischler und Robert C. Bolles
- Selektion zuverlässiger Messwerte (Eliminierung von Ausreißern)
  - Bestimmung einer Ausgleichsfunktion
    - wähle minimal nötige Anzahl Messpunkte zufällig aus
    - berechne Parameter der Ausgleichsfunktion
    - Consensus-Set: trage für jeden Messwert ein, ob er Modell unterstützt (Abstand $<$ gegebener Schranke)
  - berechne n weitere Ausgleichsfunktionen (n Verfahrensparameter)
  - diejenigen Messwerte, die die meisten Modelle unterstützen, scheinen verlässlich
- anschließend ggf. Berechnung der Ausgleichsfunktion auf diesen von Ausreißern bereinigten Werten

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Beispiel Scans



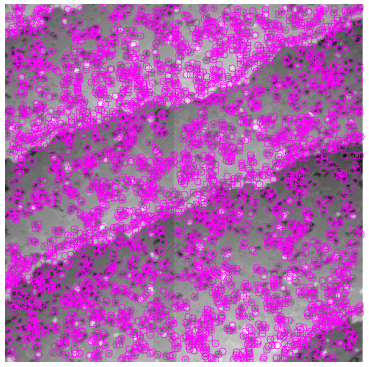
Iron on Tungsten
Overview: 200 nm x 200 nm

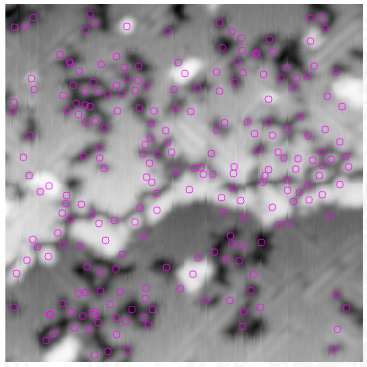


Detail: 45 nm x 45 nm

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Beispiel Scans - detektierte Features

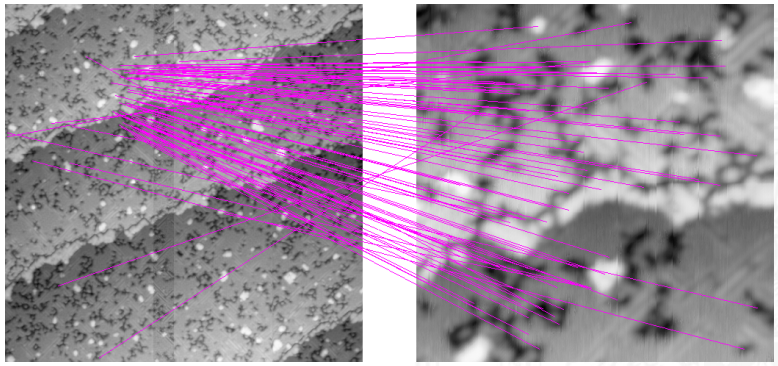




Detected Features: 4497/299

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

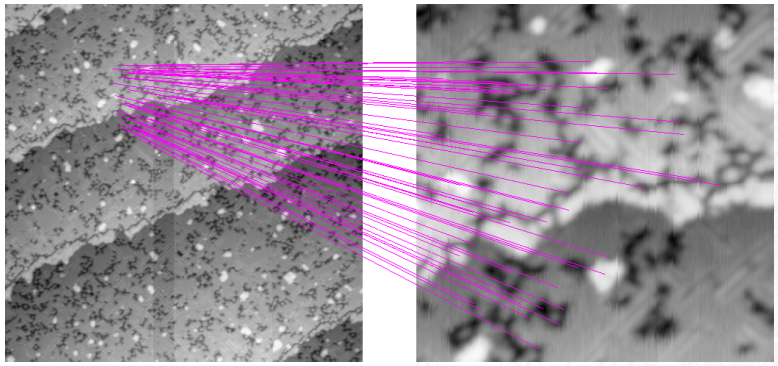
Beispiel Scans - initiale Matches



82 Matches

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Beispiel Scans - gefilterte Matches



55 verbleibende Matches (Eleminierung doppelter Matches + RANSAC)

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Berechnung der Transformation

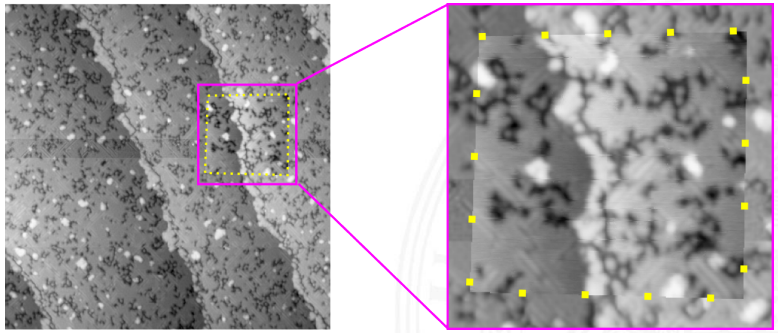
- Berechnung der homogenen Transformationsmatrix (3x3)
- Matrix trägt Information über Translation, Skalierung und Rotation

Transformatiosmatrix des Beispiels:

$$T = \begin{pmatrix} 0.222433 & -0.012754 & 302.727908 \\ -0.010619 & 0.234036 & 136.412458 \\ -0.000065 & -0.000019 & 1.000000 \end{pmatrix}$$

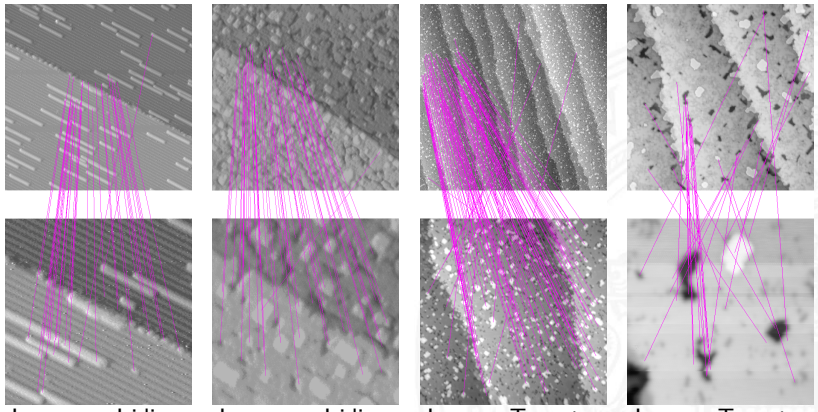
Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Rückprojektion



Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Experimentelle Ergebnisse



Iron on Iridium
Matches: 42

Iron on Iridium
Matches: 62

Iron on Tungsten
Matches: 111

Iron on Tungsten
Matches: 29

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Anwendungen

Erstellen einer "Karte" der Probe:

- Beginne mit Übersichtsscan
- scanne interessante Bereiche mit hoher Auflösung
- berechne Transformationen zwischen Übersichtsbild und Ausschnittsscans

Bestimmung der Position der Messspitze::

- scanne einen kleinen Bereich und vergleiche ihn mit vorigen Scans
- auch anwendbar nach System-Crash oder in-situ Evaporation von Atomen

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Applications (cont.)

Drift Bestimmung

- im laufenden Betrieb möglich
- ermöglicht quasi online Rekalibrierung des STM

Ermittlung der Veränderungen zwischen zwei Scans

- align Scans zuerst mittels Projektion
- berechne Unterschied Pixel für Pixel
- Unterschied zeigt z. B. Erfolg der Manipulation

Panorama Stitching

- Erzeuge große Scans durch zusammenfügen von Einzelscans
- Auflösung darf variieren

Praxisbeispiel: Verarbeitung von STM-Scandaten (cont.)

Beispiel zeigt, wie etablierte Verfahren aus Makro-Robotik bzw. Bildverarbeitung mit geringen Modifikationen direkt in Anwendungen in der Nanotechnologie (hier STM-Scans) übernommen werden kann.

[DH71] R.O. Duda, P.E. Hart:
Use of the Hough Transformation to Detect Lines and Curves in Pictures.
AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025, Apr 1971, Forschungsbericht 36.
SRI Project 8259 Comm. ACM, Vol 15, No. 1

[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