SLAM mit Partikelfilter: FastSLAM

Montemerlot, Thrun & al.:
FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem.
Proc. AAAI-2002

T/B/F, Kapitel 13

- Verwende Partikel-Idee wie bei MCL
- Jedes Partikel hält aktuelle Pose (alternativ: kompletten Pfad mit aktueller Pose als letzter) und die Schätzung der Positionen der Landmarken (feste Landmarkenzahl n)
- Leider: Naive Anwendung wie zur Lok. klappt nicht, wegen exp. steigender #Partikel (in der Dim. des Zustandsraumen)
- Aber: Nutze Zusammenhang zwischen Pose (x,y,theta) und Karte (Landmarken) aus: Karte abhängig von Roboterpose!

Java-Applet zum Spielen: www.oursland.net/projects/fastslam/

„Rao-Blackwellisierung“

Ziel: Schätze Pose und n Landmarkenpositionen; löse einzelne Landmarkenpositionen mit EKFn, nichtlineare Pose mit Partikelfilter

$$\mathbf{P}\left(m, x_t \middle| z_{1:t}, u_{1:t-1}\right) =$$ gesuchte Verteilung

$$= \mathbf{P}\left(l_{1:n}, x_t \middle| z_{1:t}, u_{1:t-1}\right)$$ Karte = Landmarken

weiter umgeformt zu:

$$= \mathbf{P}\left(x_t \middle| z_{1:t}, u_{1:t-1}\right) \cdot \mathbf{P}\left(l_{1:n} \middle| x_{1:t}, z_{1:t}, u_{1:t-1}\right)$$ bed. W.keit, s. Folie 149

$$= \mathbf{P}\left(x_t \middle| z_{1:t}, u_{1:t-1}\right) \cdot \mathbf{P}\left(l_{1:n} \middle| x_{1:t}, z_{1:t}\right)$$ l unabh.v. u, gegeben x

$$= \underbrace{\mathbf{P}\left(x_t \middle| z_{1:t}, u_{1:t-1}\right)} \cdot \underbrace{\prod_{i=1}^{n} \mathbf{P}\left(l_i \middle| x_{1:t}, z_{1:t}\right)}$$ l_i untereinand. unabh.

posteriori-Poseschätzung, ähnlich Lokalisierung

Kartier.m. bekannten Posen

Partikelfilter – grobe Idee

- aktualisiere Landmarkenpositionen durch *n* „Mini-“EKF für Positionen der einzelnen, aktuell sichtbaren Landmarken:
  - Wegen Unabhängigkeit der Landmarken: Benötige nur Position und 2x2-Matrix zur entspr. Unsicherheit
  - => effiziente Berechnung der Mini-EKFs möglich (verglichen mit EKF-SLAM)
- errechne neue Pose durch Sampling vom Bewegungsmodell
- Vergleich Messung (Landmarken) mit lokaler Karte → Gewichtung der Partikel (niedrig in unbekanntem Gebiet, hoch bei Schleifenschluss) → Resampling

Partikelfilter – Implementierung

N Partikel, **jedes** mit n „kleinen EKF“ für n Landmarken
Beschränkung auf jeweils aktuelle Posen: Inkrementelles SLAM

	Pose (volles SLAM: Pfad)	Position Landm. 1	Position Landm. 2		Position Landm. n
Partikel k=1	$s^{[1]} = (x, z, \theta)^{T[1]}$	$\mu_1^{[1]}, \Sigma_1^{[1]}$	$\mu_2^{[1]}, \Sigma_2^{[1]}$	…	$\mu_n^{[1]}, \Sigma_n^{[1]}$
Partikel k=2	$s^{[2]} = (x, z, \theta)^{T[2]}$	$\mu_1^{[2]}, \Sigma_1^{[2]}$	$\mu_2^{[2]}, \Sigma_2^{[2]}$	…	$\mu_n^{[2]}, \Sigma_n^{[2]}$
⋮					
Partikel k=N	$s^{[N]} = (x, z, \theta)^{T[N]}$	$\mu_1^{[N]}, \Sigma_1^{[N]}$	$\mu_2^{[N]}, \Sigma_2^{[N]}$	…	$\mu_n^{[N]}, \Sigma_n^{[N]}$

FastSLAM-Algorithmus

Voraussetzung:
Alle Landmarken werden erkannt und nicht verwechselt!

Tu N-mal zur Zeit t:

- Ziehe gemäß Gewichtsverteilung Partikel k mit Pose $s^{[k]}$
- Ziehe für k neue Pose $s_t^{[k]} \propto P\left(s_t \middle| s_{t-1}^{[k]}, u_{t-1}\right)$
- Für jede Beobachtung z_t^i bestimme korrespondierende Landmarke j; aktualisiere $\mu_{j,t}^{[k]}$ und $\Sigma_{j,t}^{[k]}$ per EKF
- Berechne Partikelgewicht $W^{[k]}$ des neuen Partikels („Wie gut passen Landmarken-Positionen zu Pose/Pfad?“)

Ziehe so aus den N aktuellen Partikeln N frische Partikel
(analog Algorithmus MCL, Folie 287)

Video FastSLAM

Geschätzte Partikel-Trajektorien („individuelle Partikelhistorie“ – kein volles SLAM!) aufgetragen in je wahrscheinlichster Karte zu jeder Zeit t

Video von Sebastian Thrun,
robots.stanford.edu/videos.html

FastSLAM für Rasterkarten: GridMapper

… siehe Skript!

$$P(x_t | z_{1:t}, u_{1:t-1}) \cdot P(l_{1:n} | x_{1:t}, z_{1:t})$$

$$P(x_t | z_{1:t}, u_{1:t-1}) \cdot \prod^{n} P(l_i | x_{1:t}, z_{1:t})$$

… bei Rao-Blackwellisierung klappt der letzte Schritt (Faktorisierung) nicht, da Rasterzellen nicht unabhängig voneinander!

➥ jedes Partikel muss „seine eigene“ komplette Rasterkarte mitführen!

➥ arbeite an Reduktion der Zahl nötiger Partikel!

Varianten von FastSLAM

… gibt es inzwischen etliche

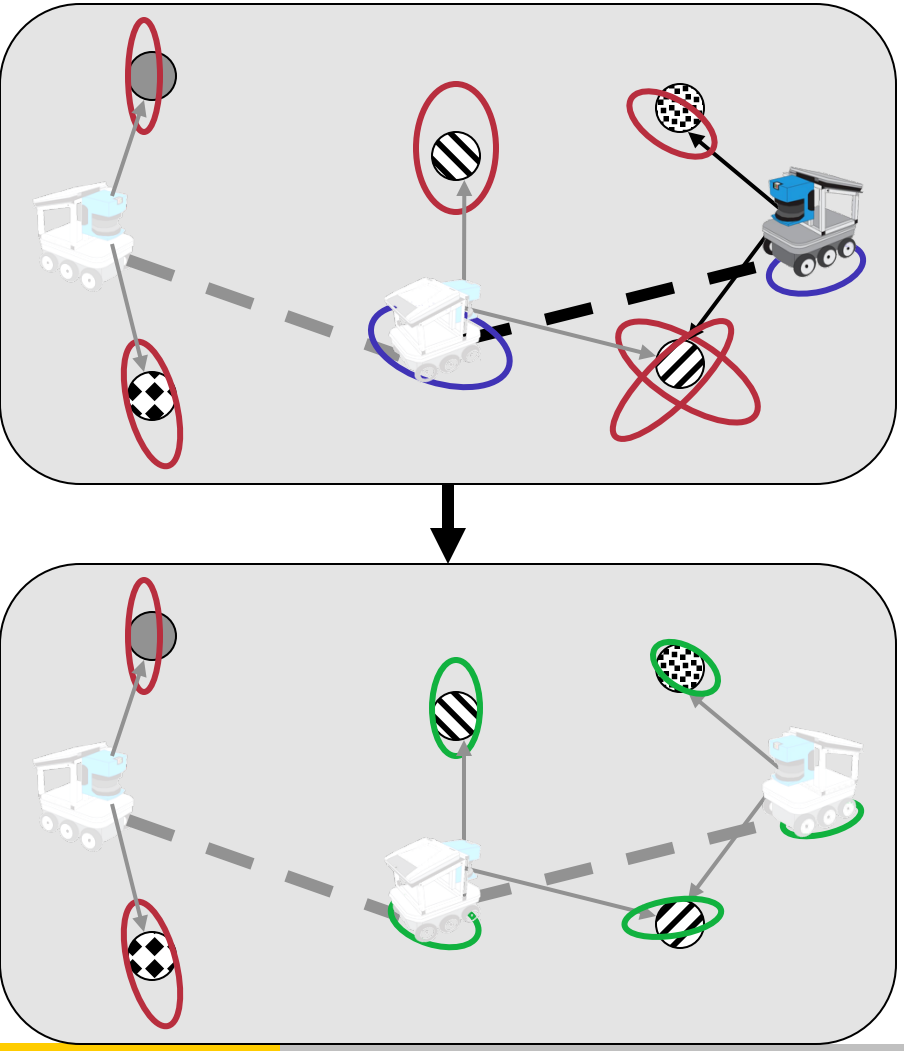
- für Rasterkarten
- unter Unsicherheit von Assoziationen
- mit Lösung des Assoziationsproblems (EM-Algorithmus)
- für multimodale Verteilungen einzelner Landmarken
- …

6.4 Vollständiges SLAM

- Inkrementelles SLAM ist vergleichsweise sparsam, weil die Pose-Historie wegfällt
- Globale Konsistenz der Information in der Karte resultiert aus Propagierung von Information (Kovarianzen) zwischen Landmarken (ausgelöst durch Schleifenschlüsse)
- Bei entfernten Landmarken kann das dauern
- Beim vollständigen SLAM bewirkt Posekorrektur beim Schleifenschluss direkte Korrektur aller Landmarkenpositionen entlang des Pfades: $\mathbf{P}\left(m, x_{1:t} \middle| z_{1:t}, u_{1:t-1}\right)$
- Karte kann schneller konvergieren. Pose kann konvergieren. Dafür sind die Algorithmen teurer

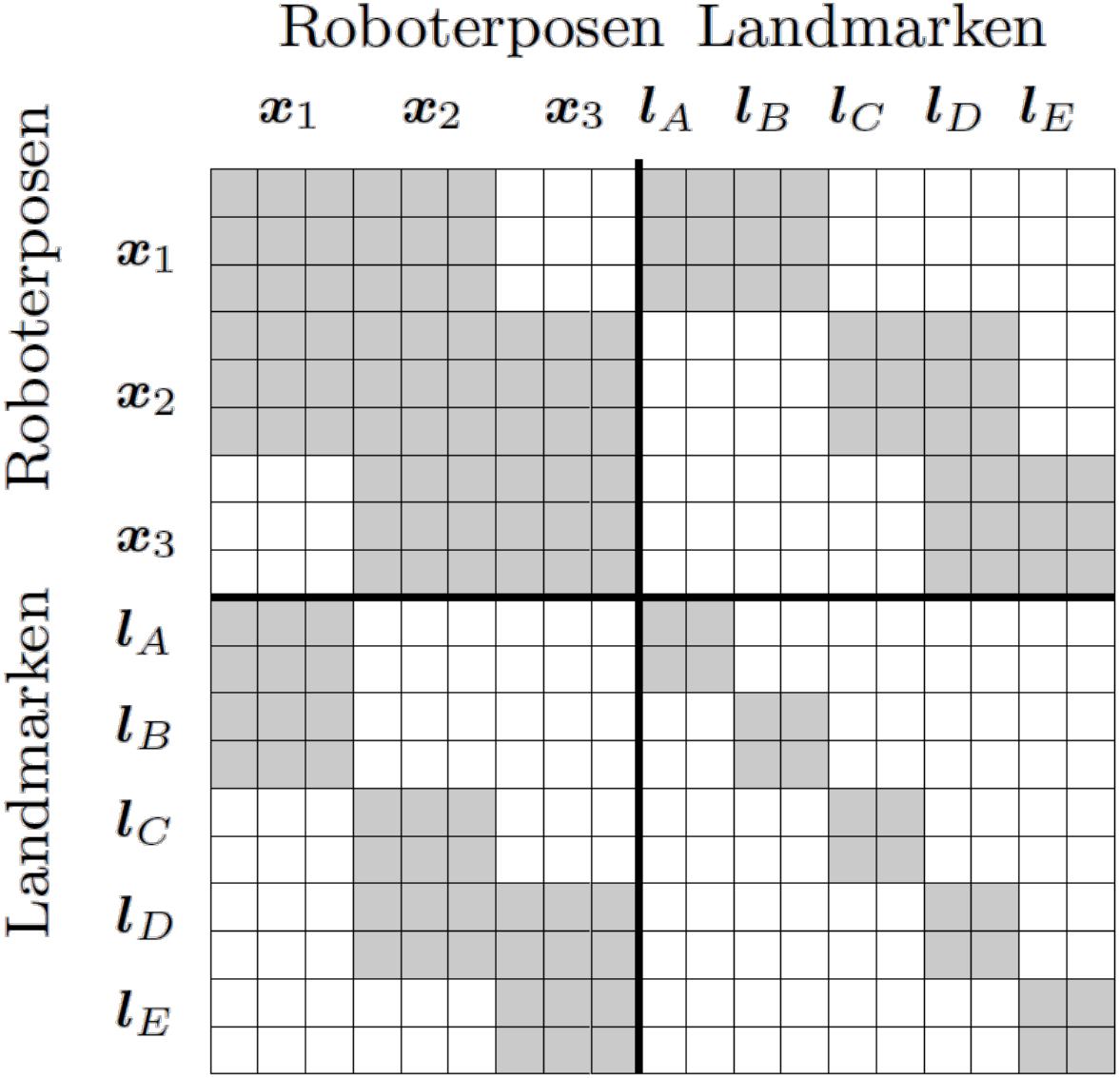
GraphSLAM – die Idee

Erinnerung Folie 273



- Bei inkr. SLAM ist Pose-Historie nicht Teil des Zustands: Karte enthält nur Landmarken-Positionen
- **GraphSLAM** erweitert Zustandsraum um alle Posen
- Den wahrscheinlichsten Zustand (Vektor aus Posen & Landmarkenpositionen) schätze als den Vektor mit minimalem Abstand von den jeweiligen Erwartungswerten
- Karte enthält auch Posefolge!

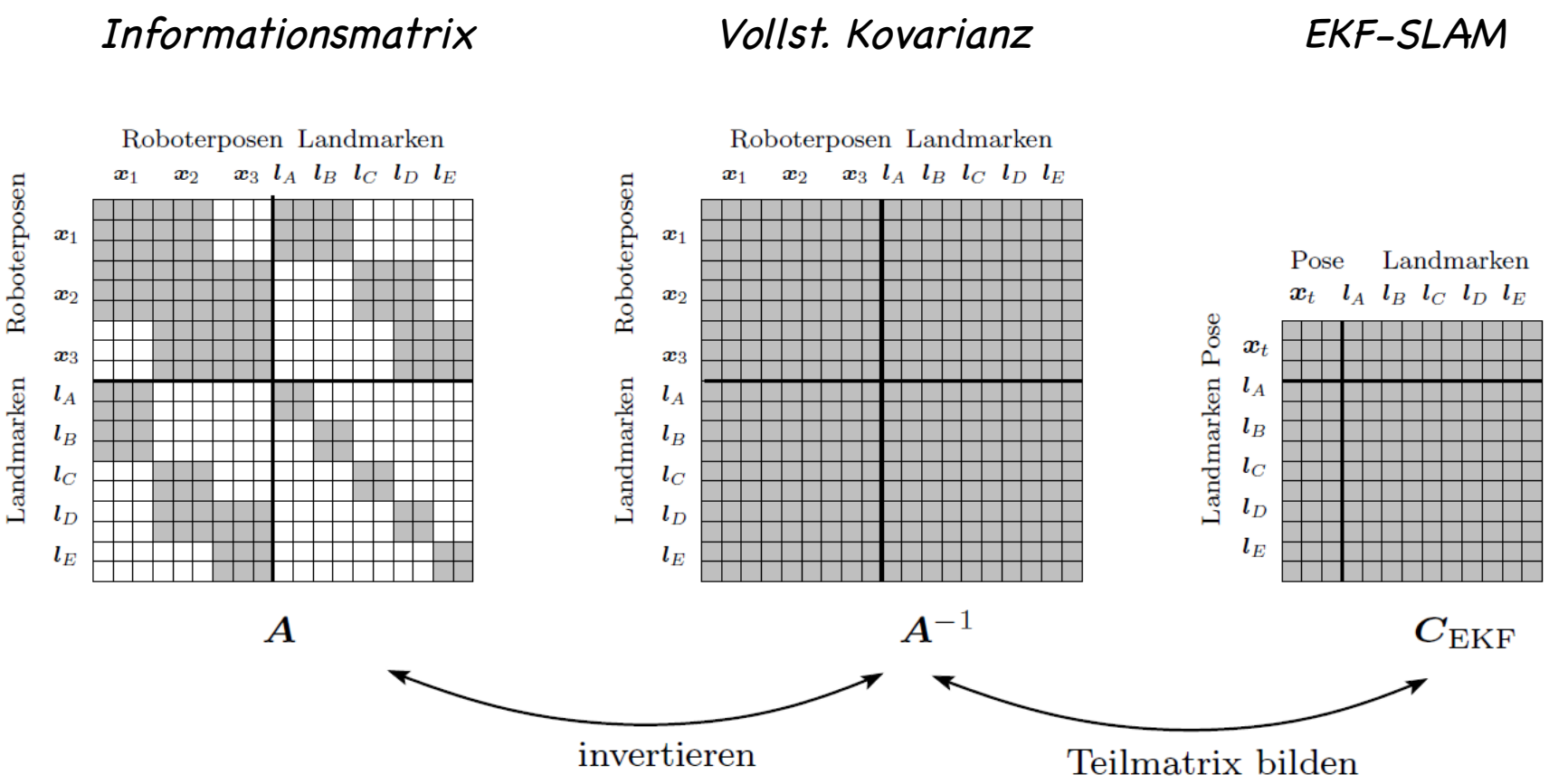
Die Informationsmatrix



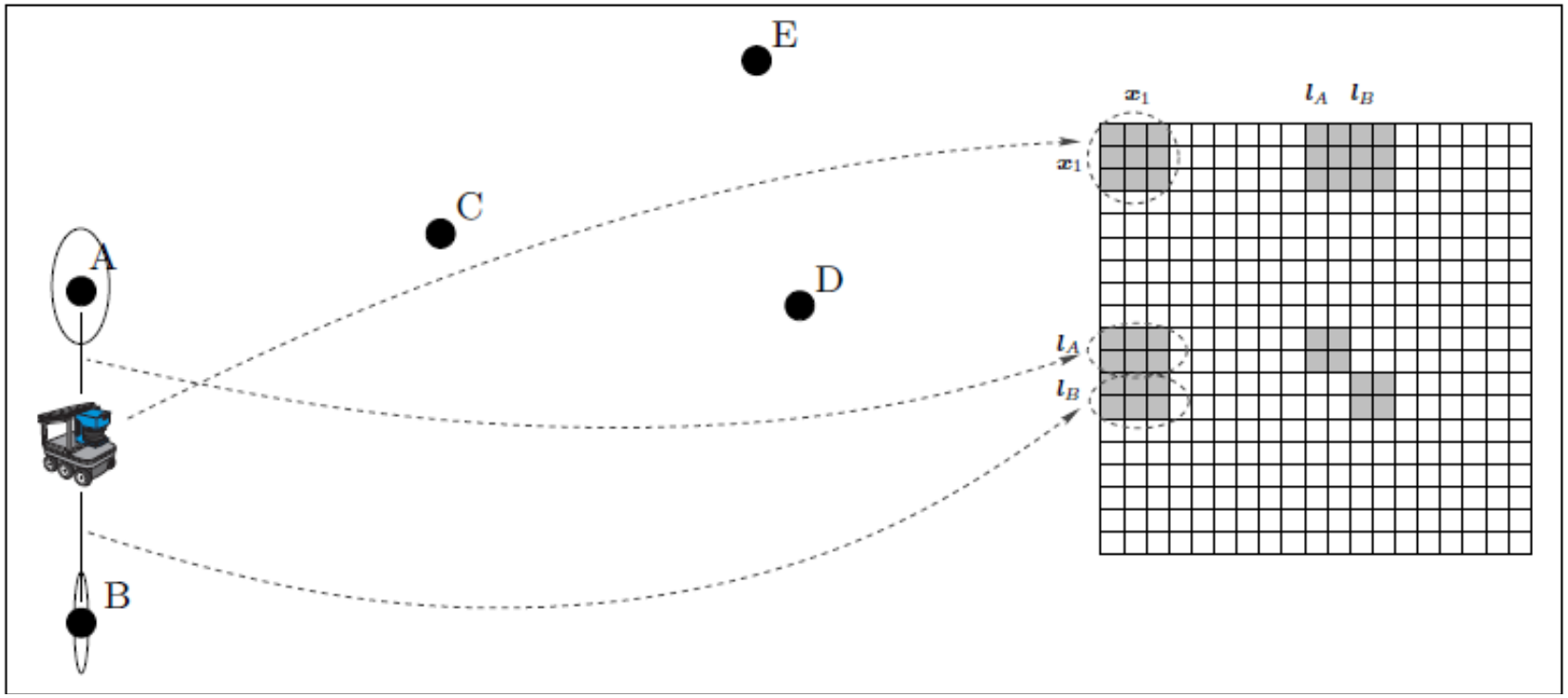
- Vollständiges SLAM braucht die vollständige Information!
- GraphSLAM hält zusätzlich Graphen korrelierter Posen
- Dadurch Info.-Matrix dünn besetzt
- (Relativ) effizient trotz voller Information

Details & Herleitungen s. Skript!

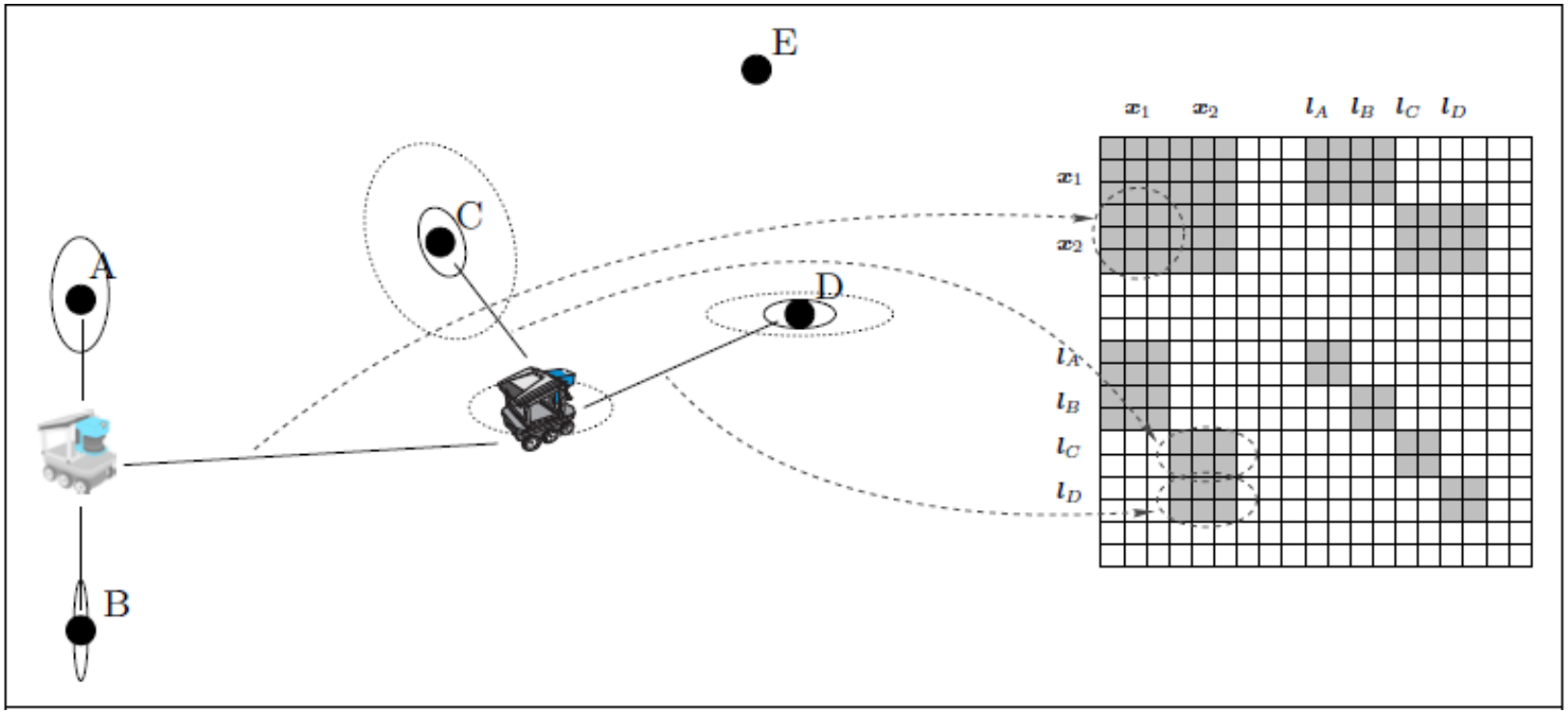
Informationsmatrix – Motivation



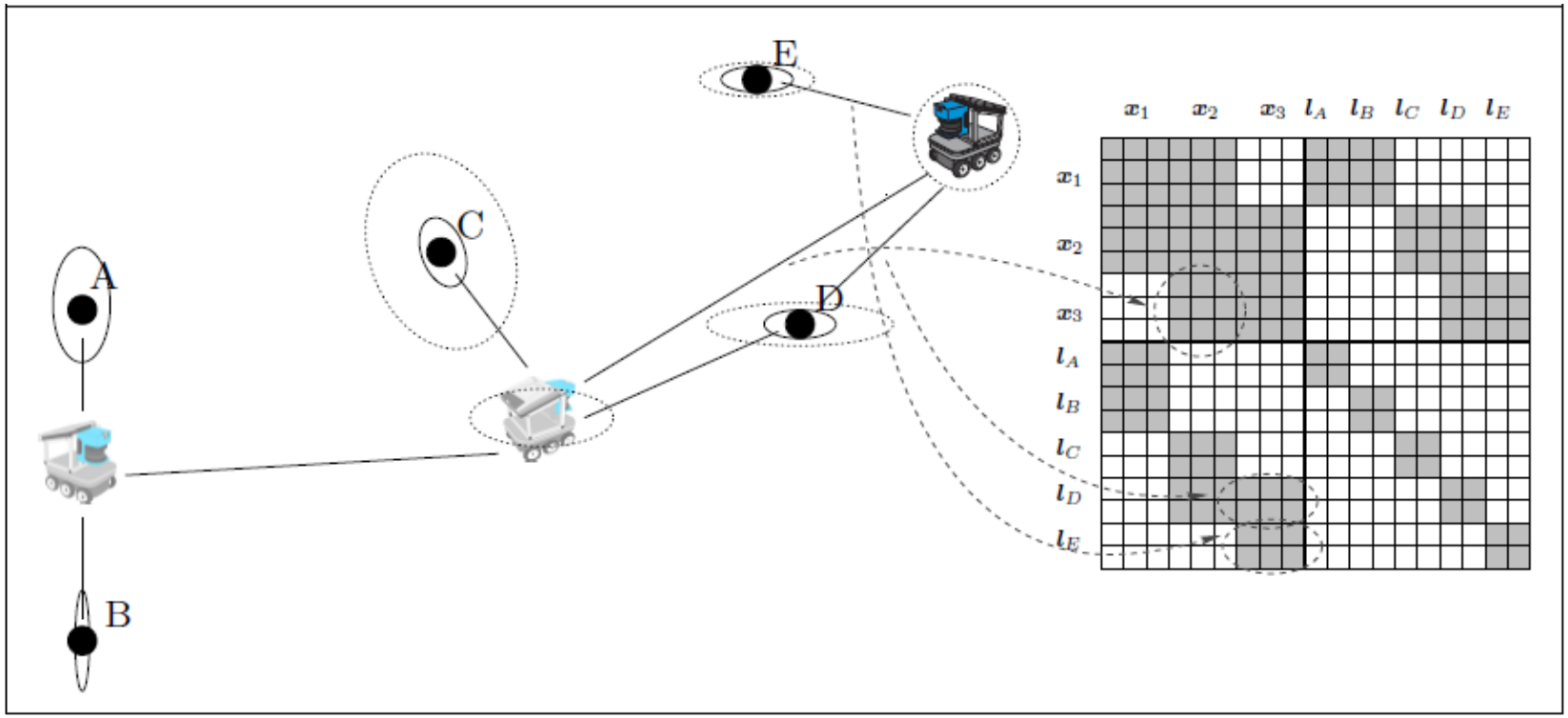
Informationsmatrix – Entwicklung (1)



Informationsmatrix – Entwicklung (2)



Informationsmatrix – Entwicklung (3)



ICP in 6D

A. Nüchter & al.:
6D SLAM with Approximate Data Association. Proc. ICAR-2005
kos.informatik.uni-osnabrueck.de/download/icar2005_2.pdf

Erinnerung an Folie 222ff

- ICP nutzt Punktkorrespondenzen $w_{i,j}$ in 2 Scans (nach Nähe)
- Sei M (Modellscan), D (Datenscan) Scans mit Punkten (3D!) $\mathbf{m}_i, \mathbf{d}_i$. Finde $\mathbf{R}, \mathbf{t}$ (jetzt in 6D!), sodass $E(\mathbf{R},\mathbf{t})$ minimal wird:

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{|M|}\sum_{j=1}^{|D|} w_{i,j} \left\| \mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t}) \right\|^2$$

Tricks zur Beschleunigung

- Datenreduktion (z.B. Octree-basiert)
- Berechnung von Rotation & Translation in geschlossener Form
- *kD*-Bäume für Nachbarpunktsuche

Rotation & Translation in geschlossener Form

$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{|M|}\sum_{j=1}^{|D|} w_{i,j} \left\| \mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t}) \right\|^2$$

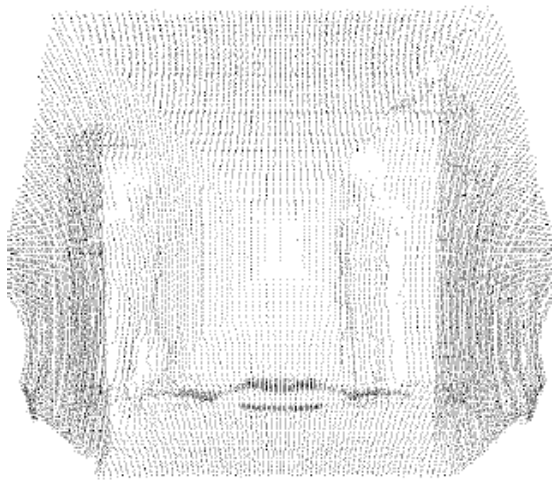
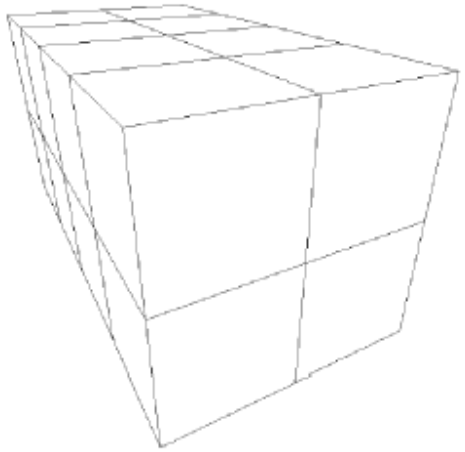
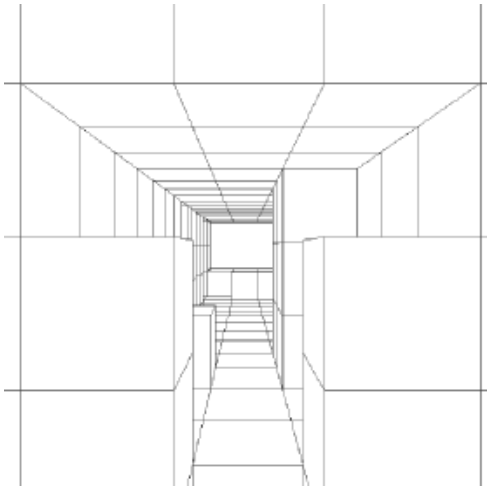
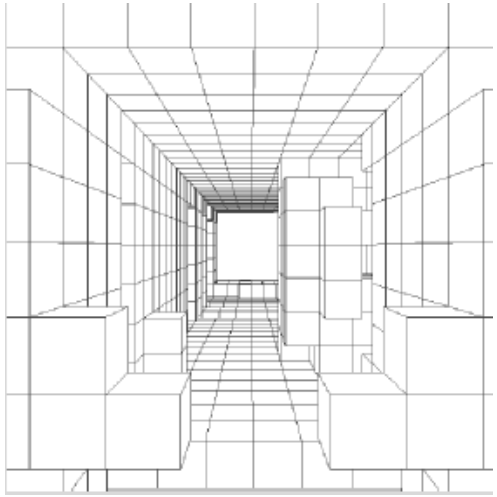
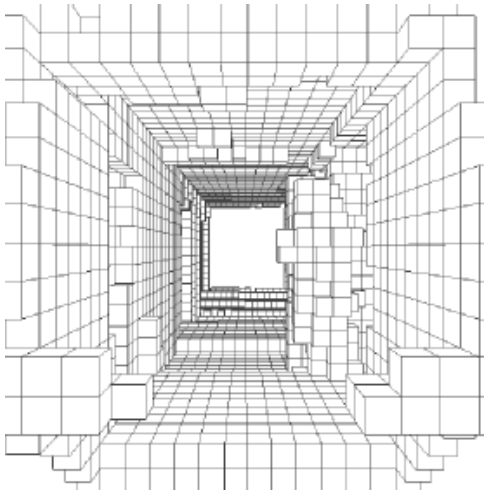
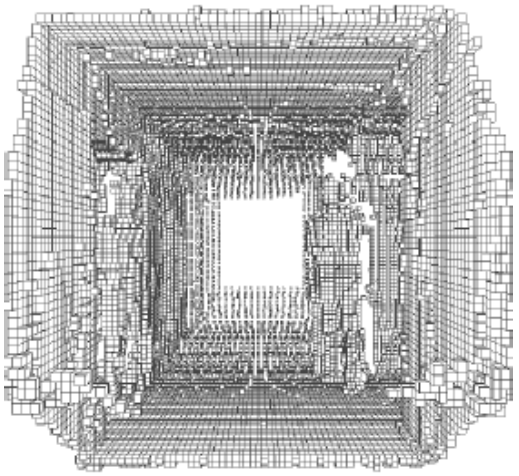
… bei HAYAI (Folie 238ff) separiert und gelöst für Merkmale

… die wir hier nicht haben (nur 3D-Punktewolken)

… doch $E(\mathbf{R},\mathbf{t})$ minimiert man ähnlich wie bei HAYAI

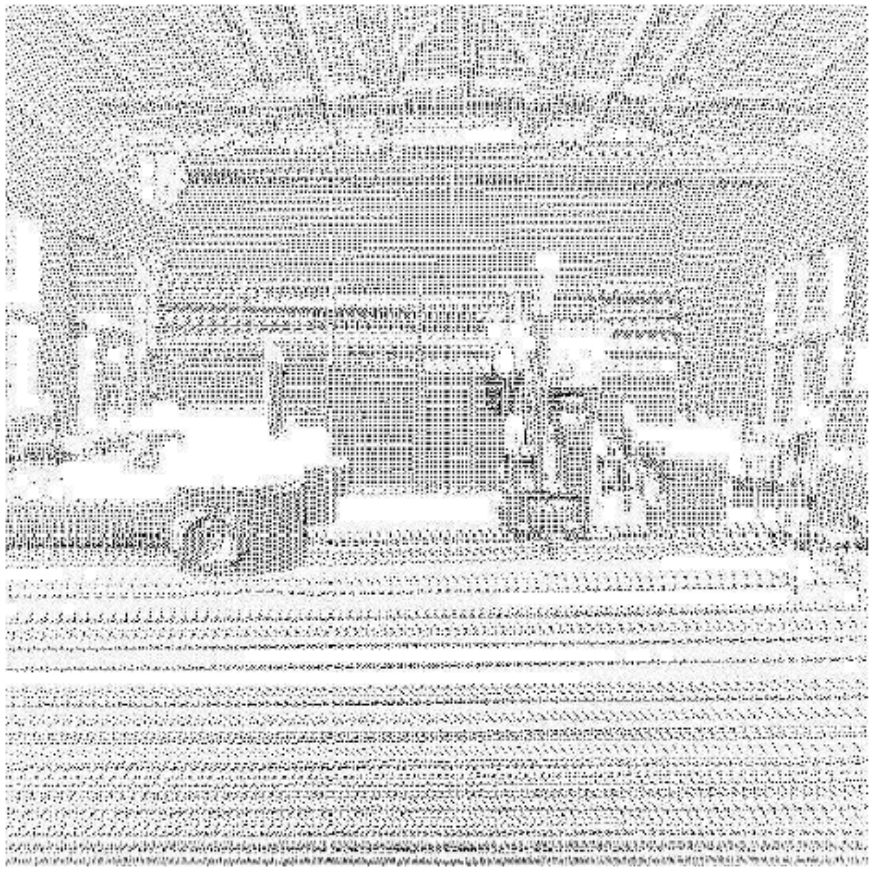
Im Detail im Buch (6.4.3)!

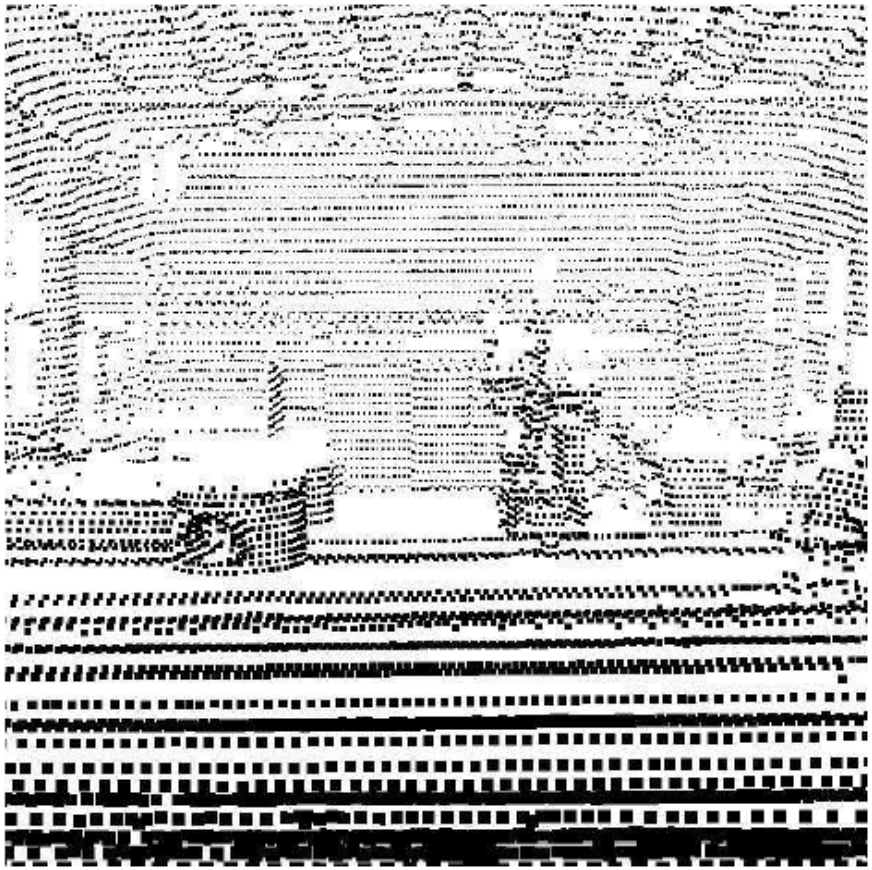
Octree-Repräsentation (z.B. zur Datenreduktion)

Datenreduktion

… entspr. Folie 254 (Reduktion für Lokalisierung durch Scanmatching)





2D-Rendering: Original + Bild mit reduzierten (+größer gemalten) Punkten

kD-Bäume, hier: *k*=3

Tricks zur Beschleunigung

- Datenreduktion ✓
- Berechnung von Rotation & Translation in geschlossener Form ✓
- *kD*-Bäume für Nachbarpunktsuche

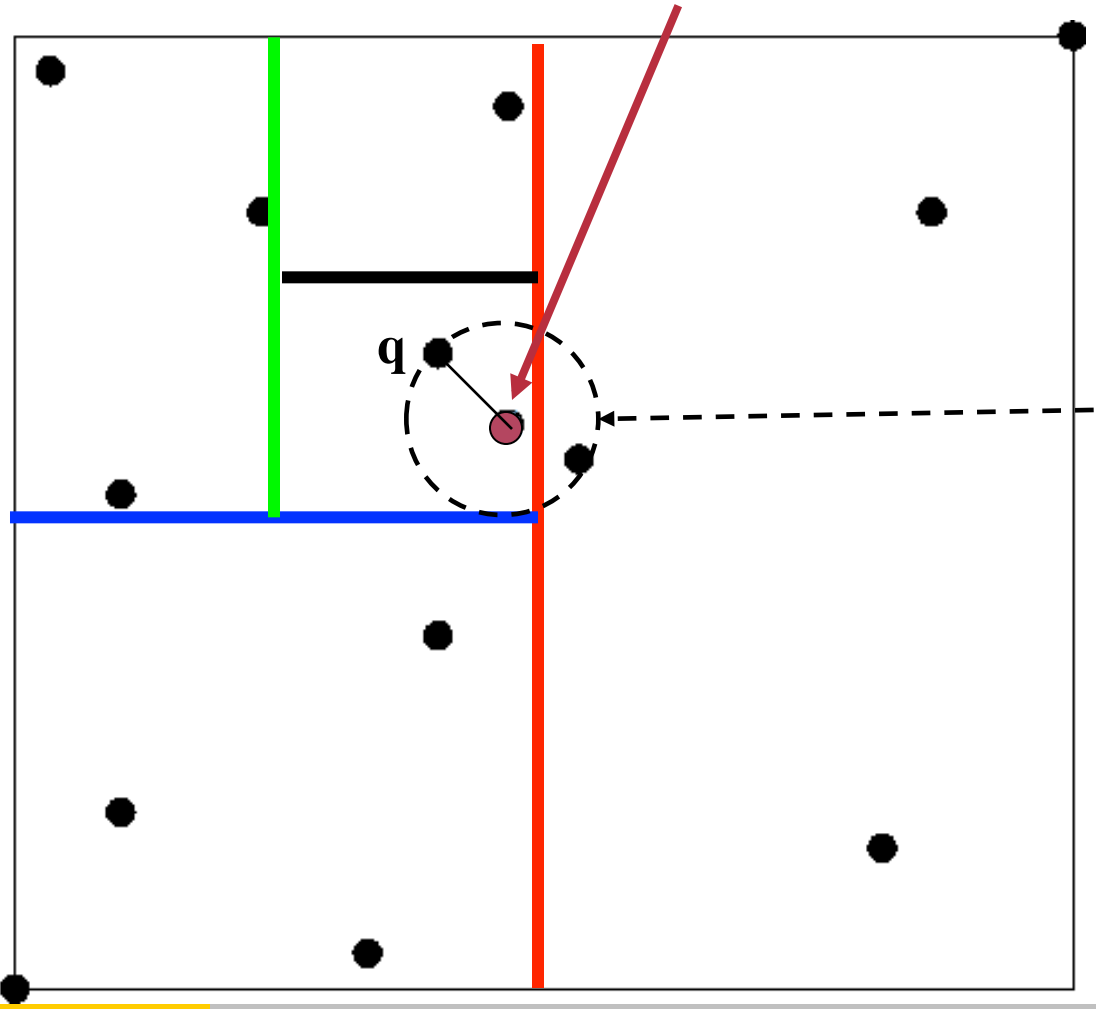
Nearest Neighbor Suche zu Punkt p

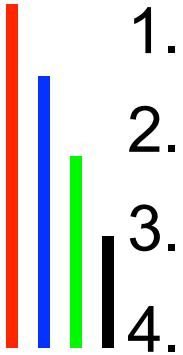
1. Zerlege durch eine Hyperebene ($(k-1)$-dimensional) den zu durchsuchenden Raum (default: gleichmäßige Zerlegung)
2. Betrachte die Hälfte, in der $\mathbf{p}$ liegt
   - liegen max. b („*bucket size*“) Punkte im selben k-Volumen, ermittle nächsten Nachbarpunkt $\mathbf{q}$; weiter bei 3.
   - sonst teile weiter (Schritt 1)
3. Nachprozessierung (s.u.)

z.B. R. Sedgewick: *Algorithmen in C*, Kap. 26
oder: www-lehre.inf.uos.de/~dbs/2005/PDF/skript-05.pdf

Beispiel: Nachbarsuche im 2D

Sei b=1 Finde Nachbarn dieses Punktes **p**!





1. Partition
2.
3.
4.

Ball-within-bounds-Test

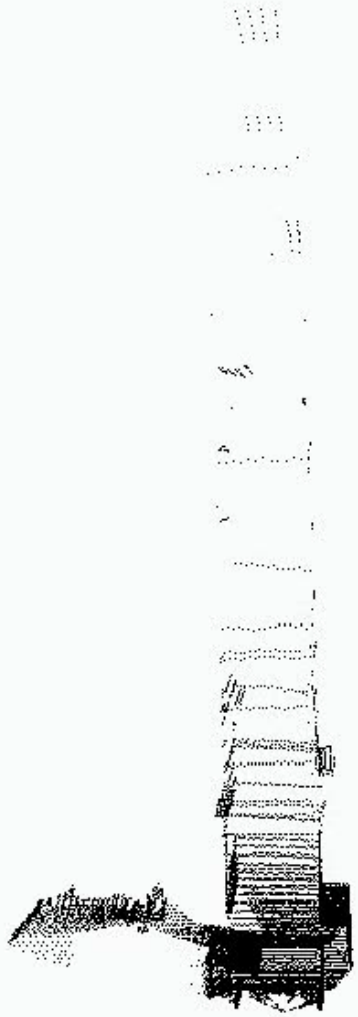
q suboptimal!

- Backtracking über alle Zerlegungen im Radius **p→q**

oder

- nimm Approximation

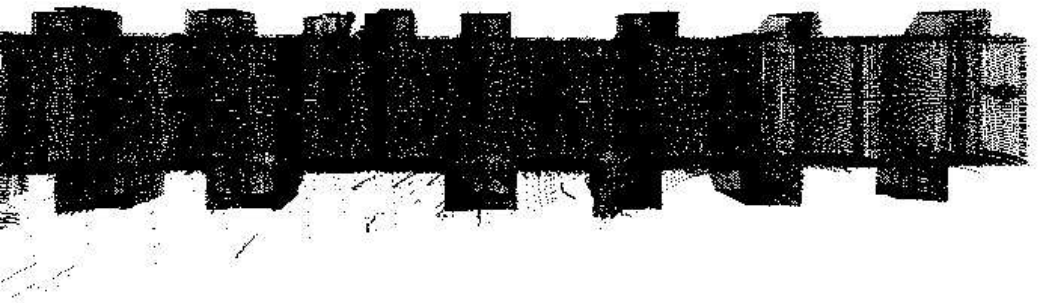
Registrierung einzelner Flur-Scans



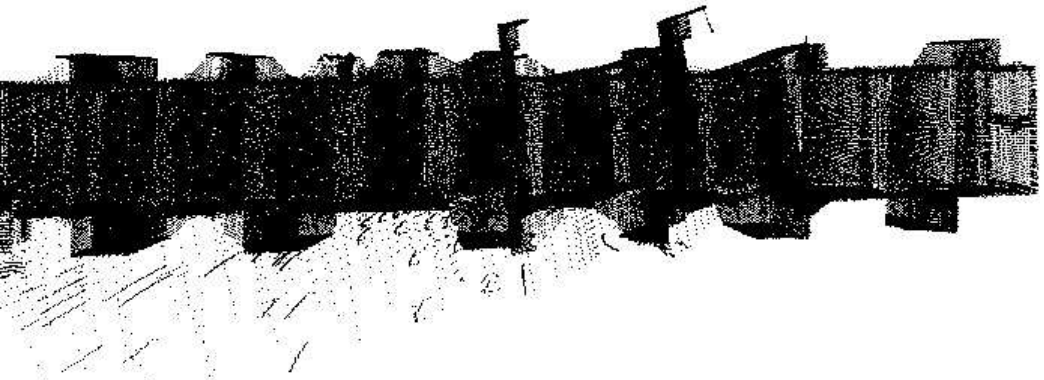
Algorithmenanimation
3D-Scans in Aufsicht auf Flur

- läuft online (zwischen Aufnahme von 2 Scans)
- nutze $(\mathbf{R},\mathbf{t})$ zur Korrektur der Poseschätzung des letzten Scans

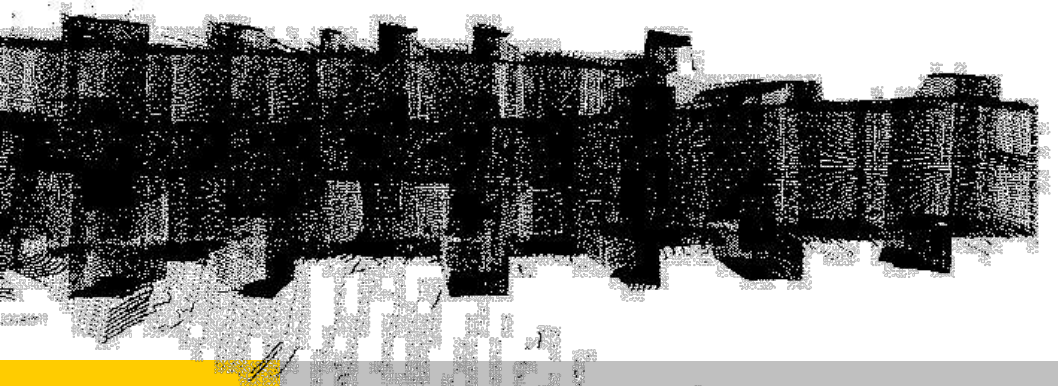
Registrierung mehrerer Scans



Paarweises Matching
Registriere mit maximal überlappendem Alt-Scan

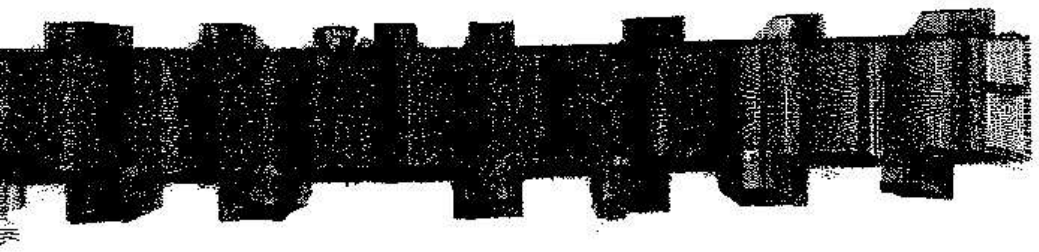


Inkrementelles Matching
Registriere mit **Metascan** (Resultat aller früheren Registrierungen)



Merkmalbasiertes Matching
Registriere auf Basis von Merkmalsmengen, hier: Linien-Endpunkte

Registrierung mehrerer Scans (Forts.)



Simultanes Matching
Registriere iterativ (bis zur globalen Relaxierung) alle Scans bzgl. aller Nachbarn; fixiere den ersten Scan (Ursprung)

Registrierung von 20 Scans (~50,000 Pkte., reduziert) der Büroflur-Szene wie dargestellt (Laufzeiten relativ, Messungen auf alter Hardware)

Match-Methode	Laufzeit
Paarweise	47 sec
Inkrementell	59 sec
Merkmalbasiert	40 sec
Simultan	17:44 min

Pragmatische Lösung

- Paarweises Matching on-line auf dem Roboter
- Simultanes Matching off-line zum Kartenbau

Beispiel: 3D-Karte aus dem Schloss