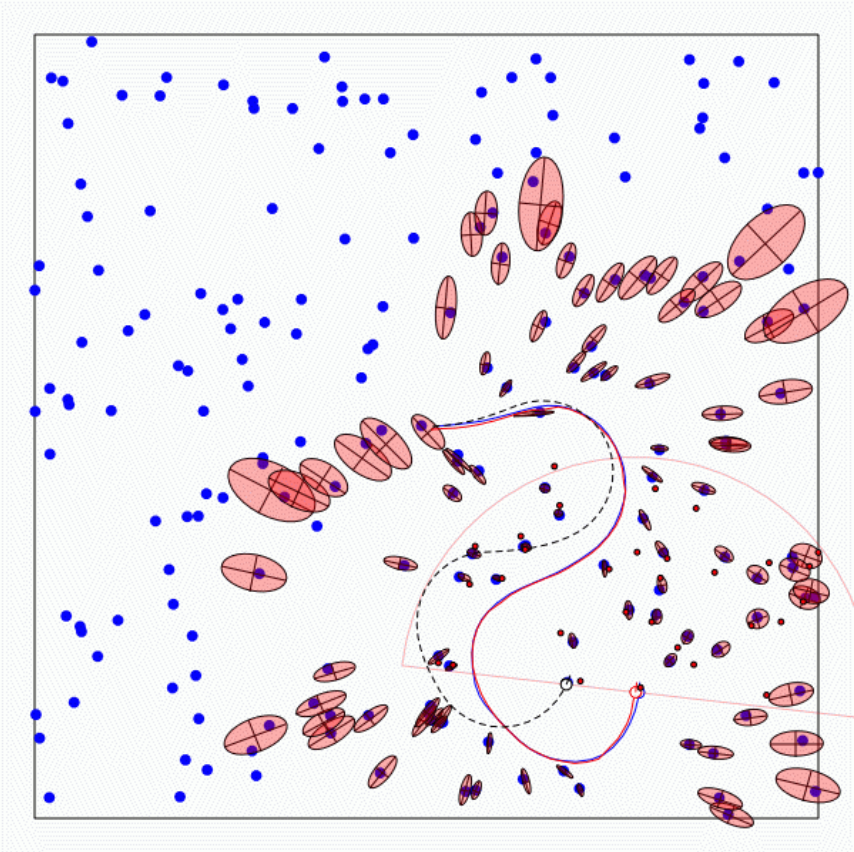
Fast-SLAM:
Synchrone Lokalisierung und Kartenerstellung mit einem Partikel-Filter

- Landmarkenbasiertes Fast-SLAM
- Gitterbasiertes Fast-Slam
- Optimierungen

Landmarkenbasiertes SLAM – Problemstellung

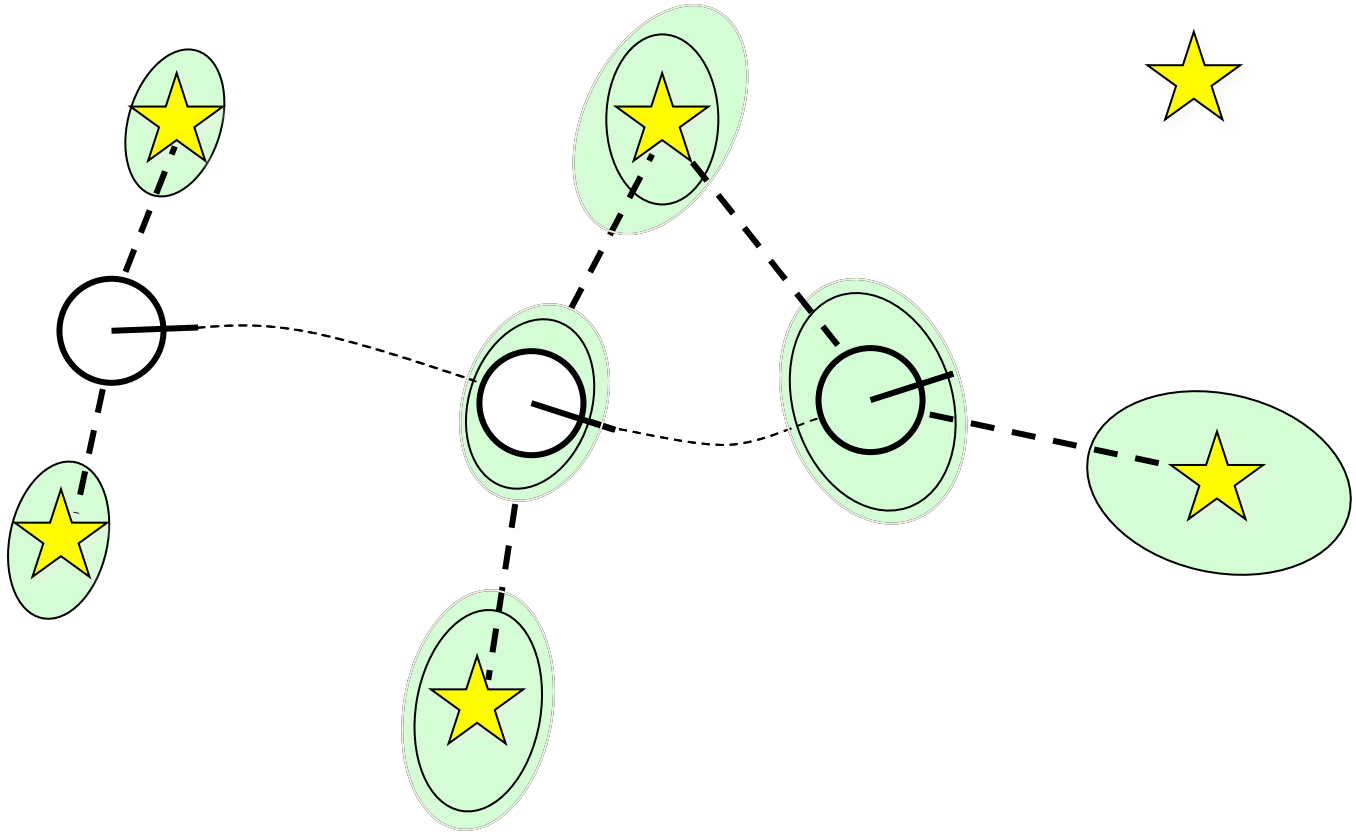
- Roboter exploriert eine unbekannte, statische Umgebung.



- Gegeben Sensor- und Steuerdaten:
  - $d = u_1, z_1, u_2, z_2, ..., u_k, z_k$
- Gesucht:
  - Karte m mit M Landmarken:
    $m = \ell_{1,x}, \ell_{1,y}, ..., \ell_{M,x}, \ell_{M,y}$
  - Weg des Roboters: $x_1, x_2, ..., x_k$

Warum ist SLAM ein schwieriges Problem?

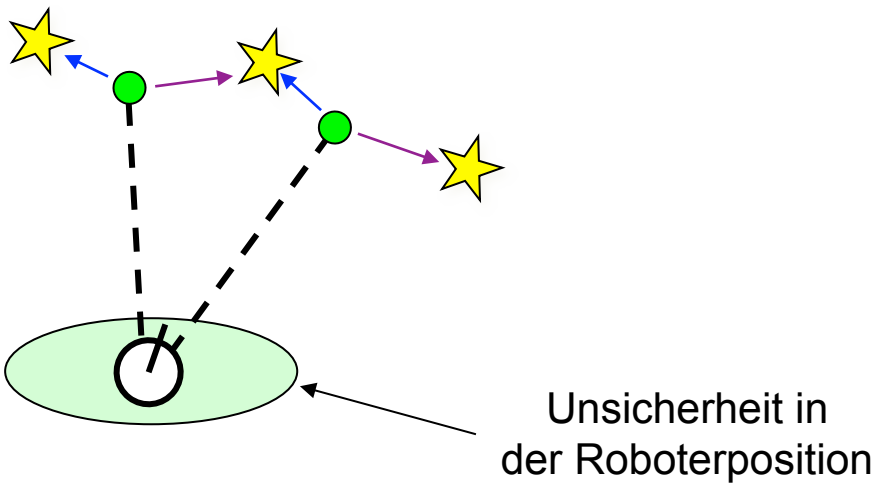
- Sowohl Positionen der Landmarken als auch Roboterweg sind unbekannt.



- Kartenfehler und Fehler im Roboterweg sind korreliert.

Warum ist SLAM ein schwieriges Problem?

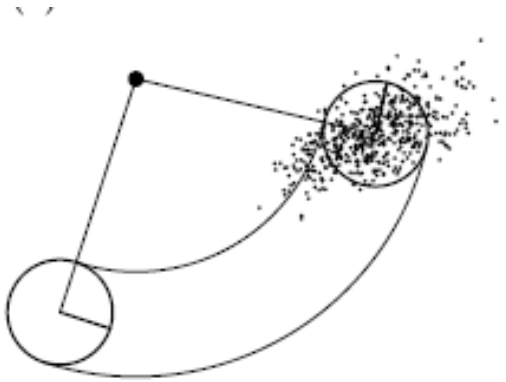
- Die Zuordnung von Messdaten zu Landmarken (data association) sind in der Regel unbekannt.
- Roboter muss entscheiden, ob Messdaten einer bereits beobachteten Landmarke zugeordnet werden können oder einer noch nicht gesehenen Landmarke.
- Zuordnungsproblematik wird durch Fehler im Roboterweg verstärkt.



Partikelfilter für Selbstlokalisierung -Wiederholung

Position als Partikelmenge:

- Wahrscheinlichkeitsverteilung für Position bel(x_k) wird durch eine Partikelmenge χ_k dargestellt.
- Vorteil: beliebige Verteilungen möglich.



Generierung einer neuen Partikelmenge:

- Wende auf jeden Partikel Steuerbefehl an.
- Stelle aufgrund der Sensordaten fest, wie wahrscheinlich das Gewicht jedes Partikels ist und gewichte jeden Partikel damit.
- Resampling: ziehe N zufällige Partikel nach ihrem Gewicht.

Bei Positionstracking genügen etwa 100 Partikel:

- Es ist lediglich ein dreidimensionaler Vekor (x,y,ϕ) zu schätzen.
- Ungefähre Position ist bei Positionstracking bekannt.

Partikelfilter für SLAM (1)

- Prinzipiell kann ein Partikelfilter auch verwendet werden, um das SLAM-Problem zu lösen.
- Zustandsraum:
  $\{(x_{1:k}, m) \mid \text{mit } m = \ell_{1,x}, \ell_{1,y}, \ldots, \ell_{M,x}, \ell_{M,y}\}$

Problem:

- Der Zustandsraum kann eine Dimension von einigen Hundert haben.
- Die Partikelmenge hängt exponentiell von der Dimension ab.

Partikelfilter für SLAM (2)

SLAM kann trotzdem mit einem Partikelfilter gelöst werden:

- x_1 = (0,0,0) ist bekannt. Daher Positions-Tracking (keine globale Lokalisierung)
- Es gibt Abhängigkeiten zwischen $x_{1:k}$ und m.
- Die Partikel können daher aus einem kleineren Zustandsraum gewählt werden.

Schlüssel liegt in folgender Gleichung:

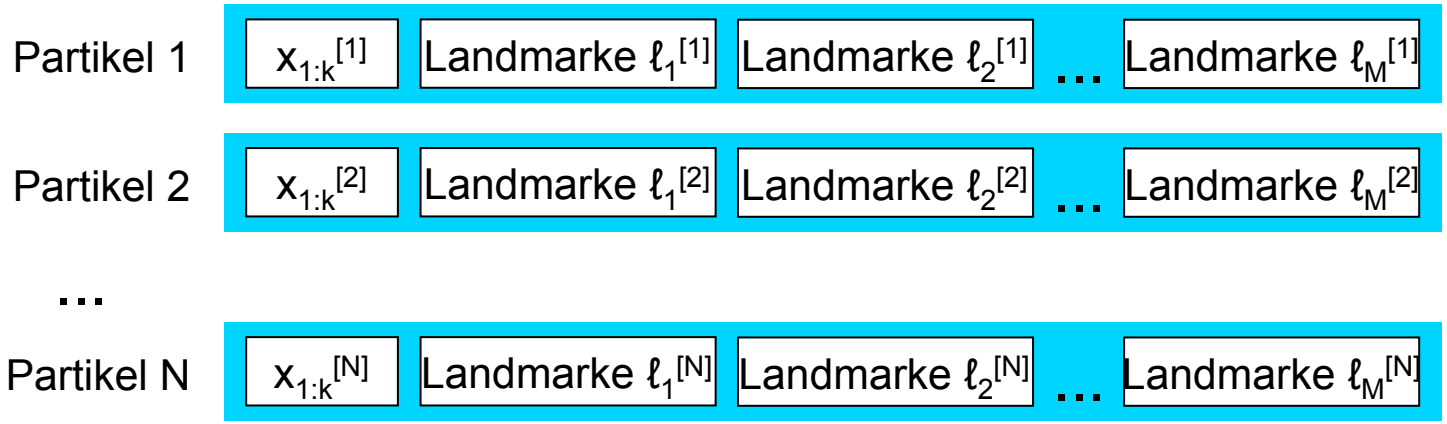
$$p(x_{1:k}, l_{1:M} \mid z_{1:k}, u_{1:k})$$
$$= p(x_{1:k} \mid z_{1:k}, u_{1:k}) \cdot p(l_{1:M} \mid x_{1:k}, z_{1:k})$$
$$= p(x_{1:t} \mid z_{1:t}, u_{1:t}) \cdot \prod_{i=1}^{M} p(l_i \mid x_{1:t}, z_{1:t})$$

Positionstracking-Problem

Bei bekannten Positionen und Sensordaten sind Landmarken-positionen unabhängig

SLAM-Verfahren (1)

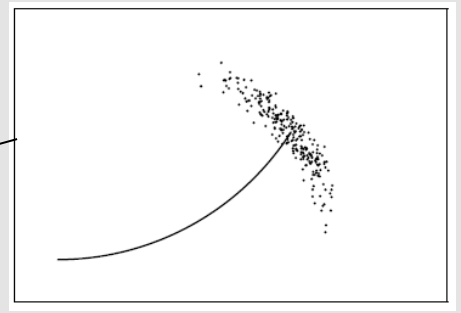
- Verwalte zu jedem Zeitpunkt t Partikelmenge χ_t:



- $x_{1:k}^{[p]}$ ist der im Partikel p gespeicherte Roboterweg.
- In jedem Partikel p wird jede Landmarkenschätzung als normalverteilt angenommen:
  $\ell_i^{[p]} \sim N(q_{i,k}^{[p]}, \Sigma_{i,k}^{[p]})$ (Position $q_{i,k}^{[p]}$ und Kovarianz $\Sigma_{i,k}^{[p]}$ sind 2-dimensional!)
- Jede Landmarkenschätzung wird mit einem EKF-Filter zu jedem Zeitpunkt aktualisiert. Damit verwaltet jeder Partikel M 2-dimensionale EKF-Filter.

SLAM-Verfahren (2)

Algorithmus FastSLAM(χ_{k-1}, u_k, z_k):



for p = 1 to N do

Integration des Steuerbefehls:

$x_k^{[p]} = \text{sampleMotionModel}(u_k, x_{k-1}^{[p]})$;

Integration der Sensordaten:

für jedes Sensordatum z_k^j ermittle zugehörige Landmarke ℓ_i und integriere z_k^j mit einem EKF, indem $q_{i,k}^{[p]}$ und $\Sigma_{i,k}^{[p]}$ aktualisiert werden;

Gewicht für jeden Partikel berechnen:

$w_k^{[p]}$ ergibt sich aus Wahrscheinlichkeit, dass von $x_k^{[p]}$ die Sensordaten z_k beobachtet werden;

endfor

Resampling:

$\chi_k = \varnothing$;

for i = 1 to N do

ziehe p zufällig mit Wahrscheinlichkeit $w_k^{[p]}$;

$\chi_k = \chi_k \cup \{ <x_{1:k}^{[p]}, q_{1,k}^{[p]}, \Sigma_{1,t}^{[p]}, \ldots, q_{M,k}^{[p]}, \Sigma_{M,k}^{[p]} > \}$;

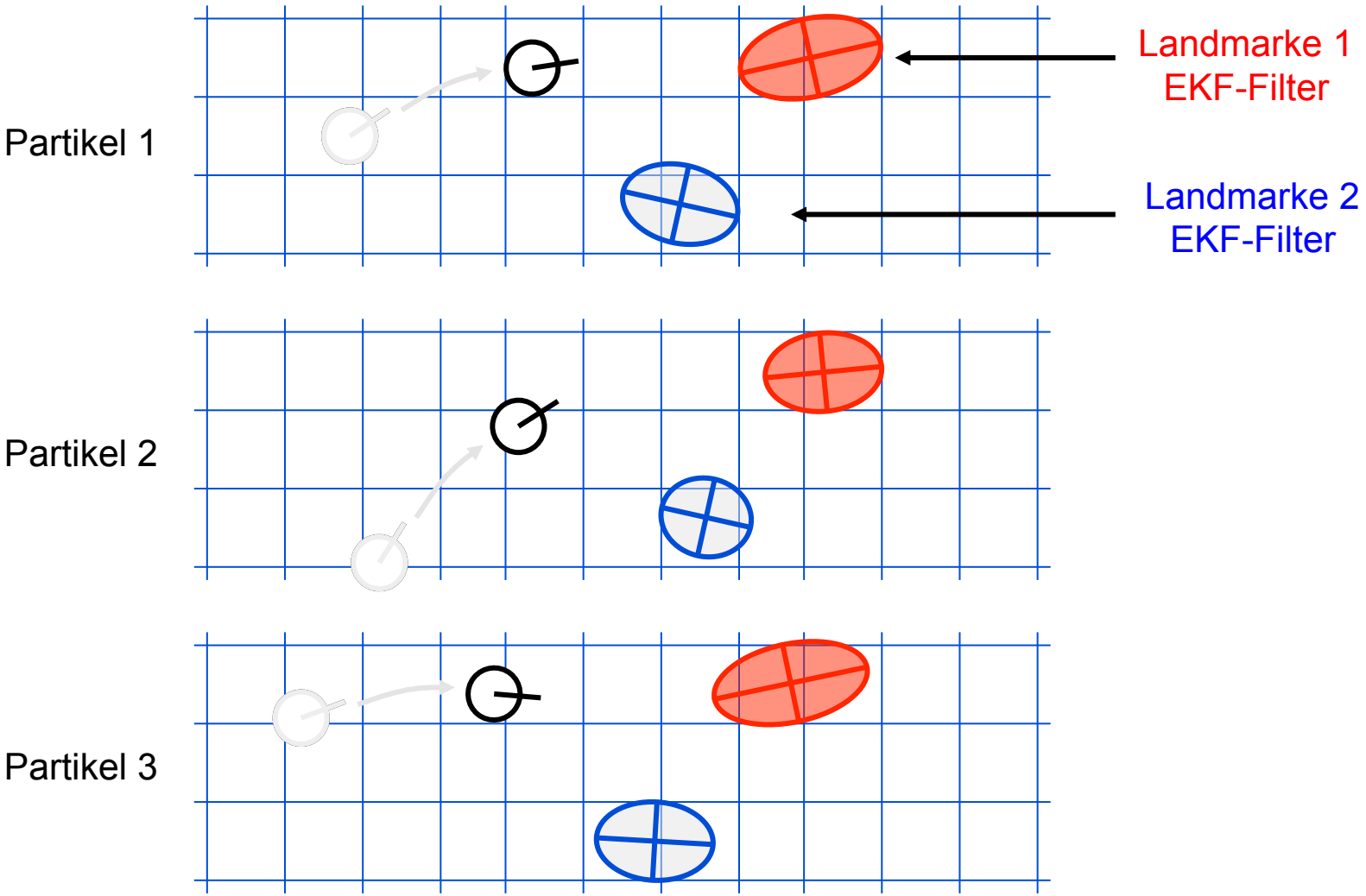
endfor

return χ_k;

EKF ohne Vorhersageschritt (d.h. ohne Bewegungsmodell)

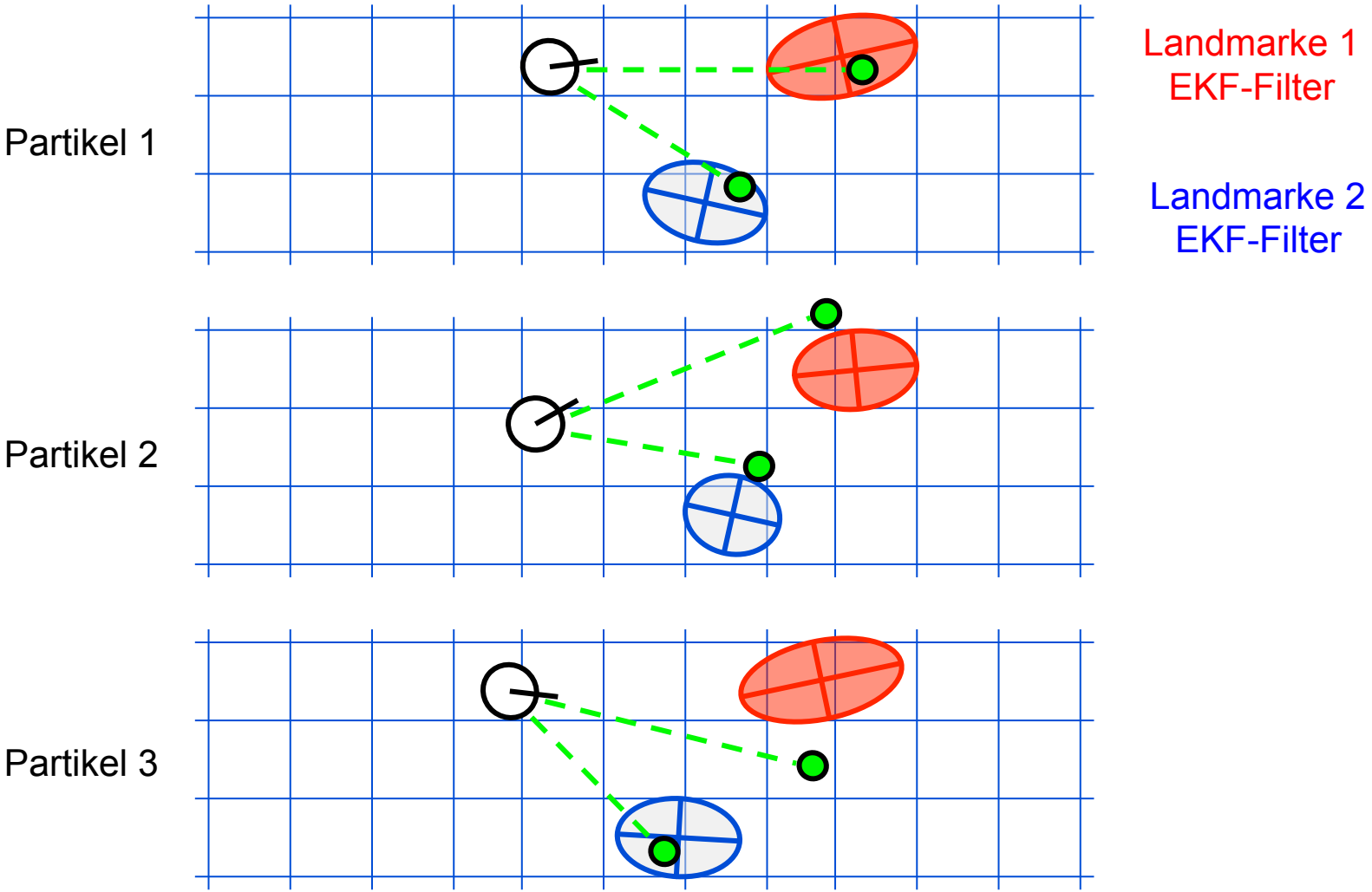
SLAM-Verfahren (3)

Integration des Steuerbefehls:



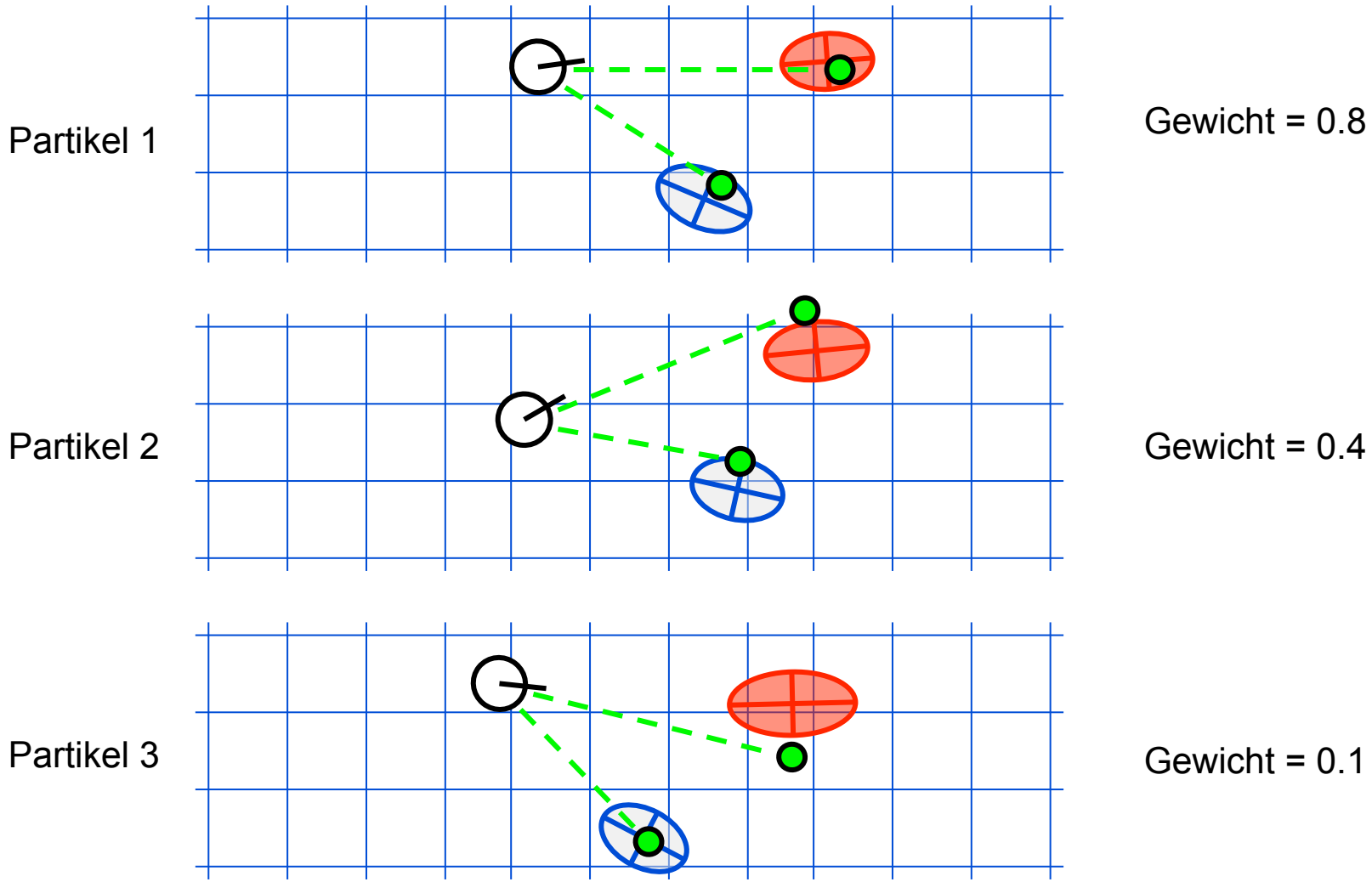
SLAM-Verfahren (4)

Integration der Sensordaten:



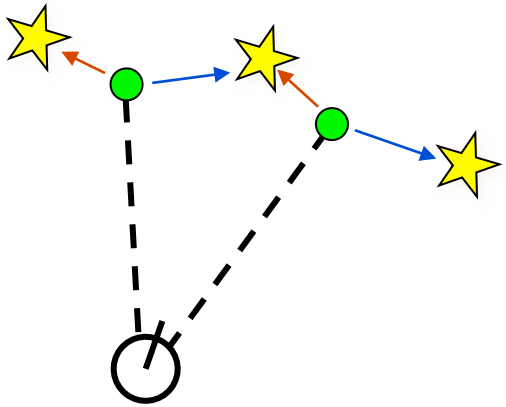
SLAM-Verfahren (5)

Integration der Sensordaten und Gewichtung der Partikeln:

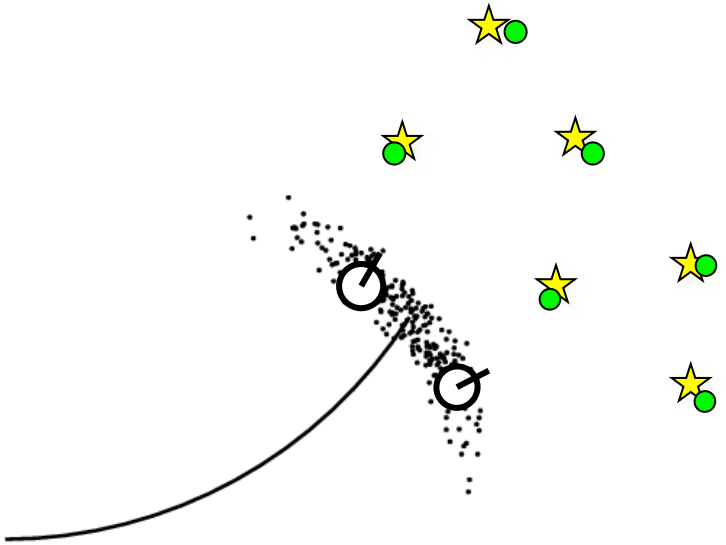


Zuordnung von Meßdaten zu Landmarken (1)

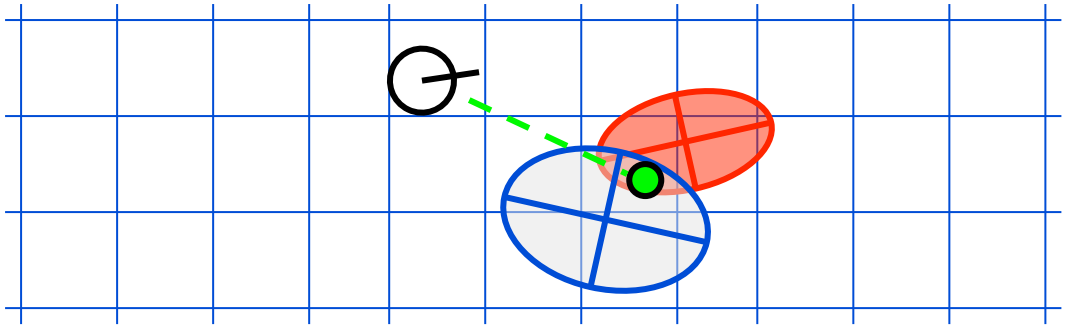
- Zu welchen Landmarken gehören die einzelnen Meßdaten:



- Die Zuordnungen werden für jeden Partikel getrennt durchgeführt:



Zuordnung von Meßdaten zu Landmarken (2)



Wurde der Sensorwert durch die rote oder die blaue Landmarke verursacht?

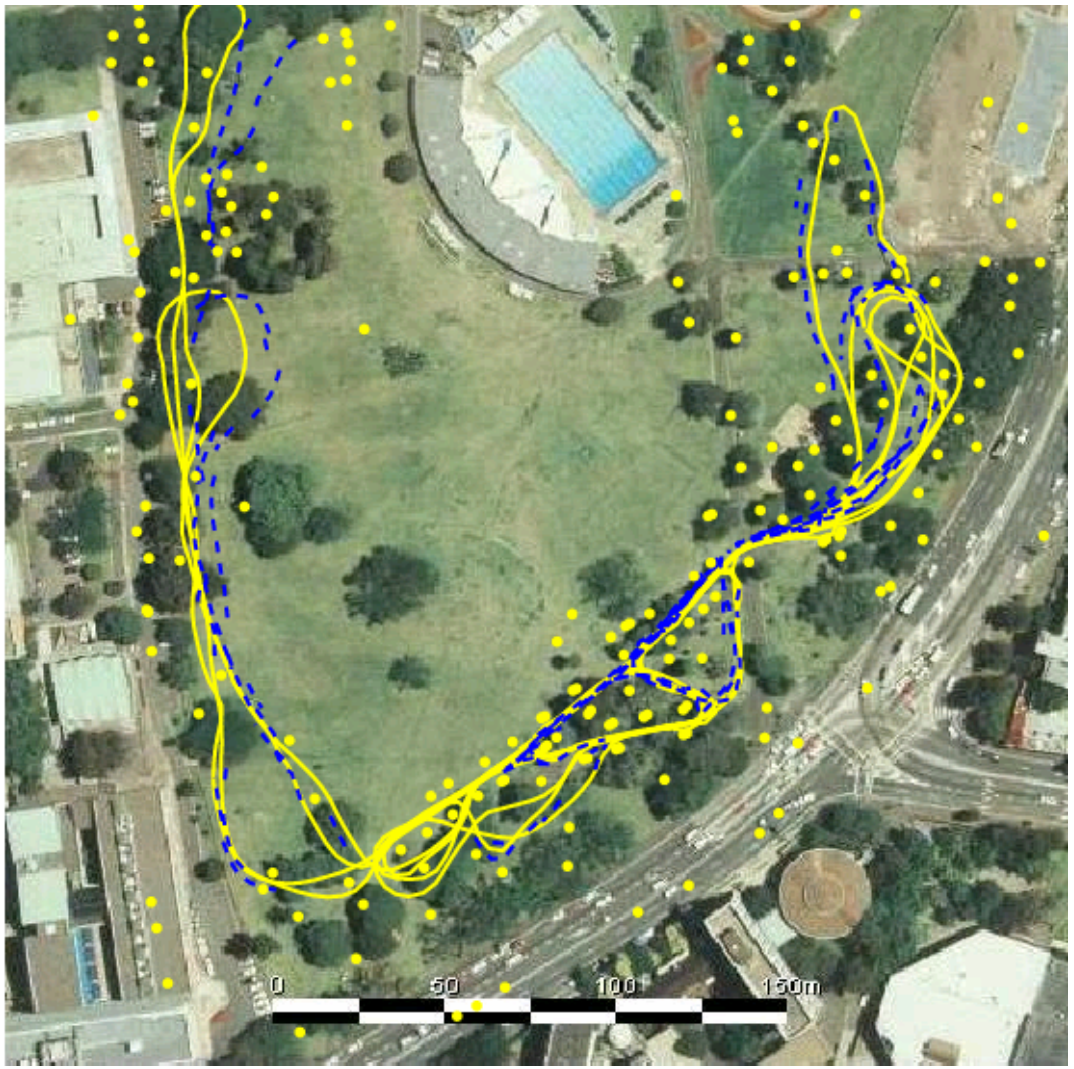
P(observation|red) = 0.3 P(observation|blue) = 0.7

- Wähle die wahrscheinlichste Zuordnung
  oder erzeuge eine zufällige Zuordnung gewichtsgesteuert.
- Falls die Wahrscheinlichkeit unter einem vorgegebenen Schwellenwert liegt, wird eine neue Landmarke eingeführt.

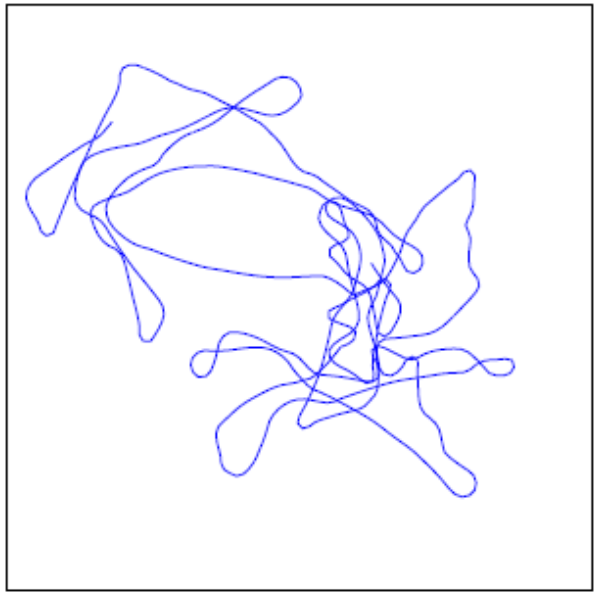
Kartierung des Viktoria-Parks, Toronto (1)

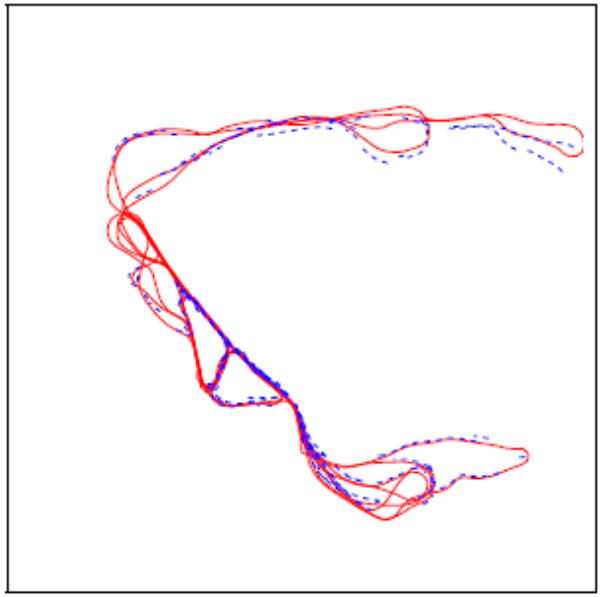
- 4 km langer Weg
- < 5 m RMS Positionsfehler
- 100 Partikeln

- Blau: GPS
- Gelb: FastSLAM



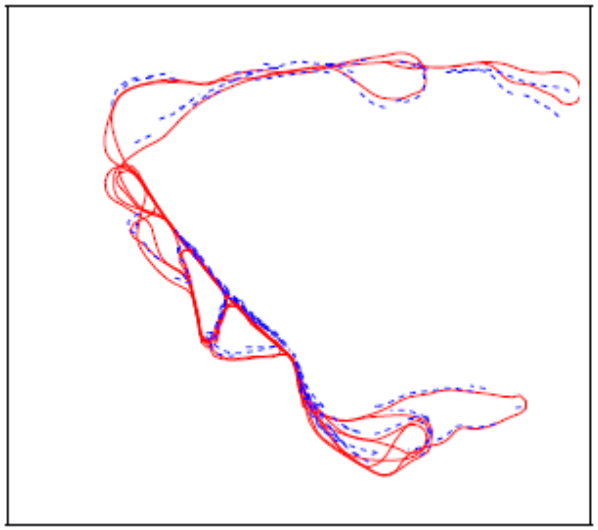
Kartierung des Viktoria-Parks, Toronto (2)





Linkes Bild:
mit Odometrie
gemessener Weg.

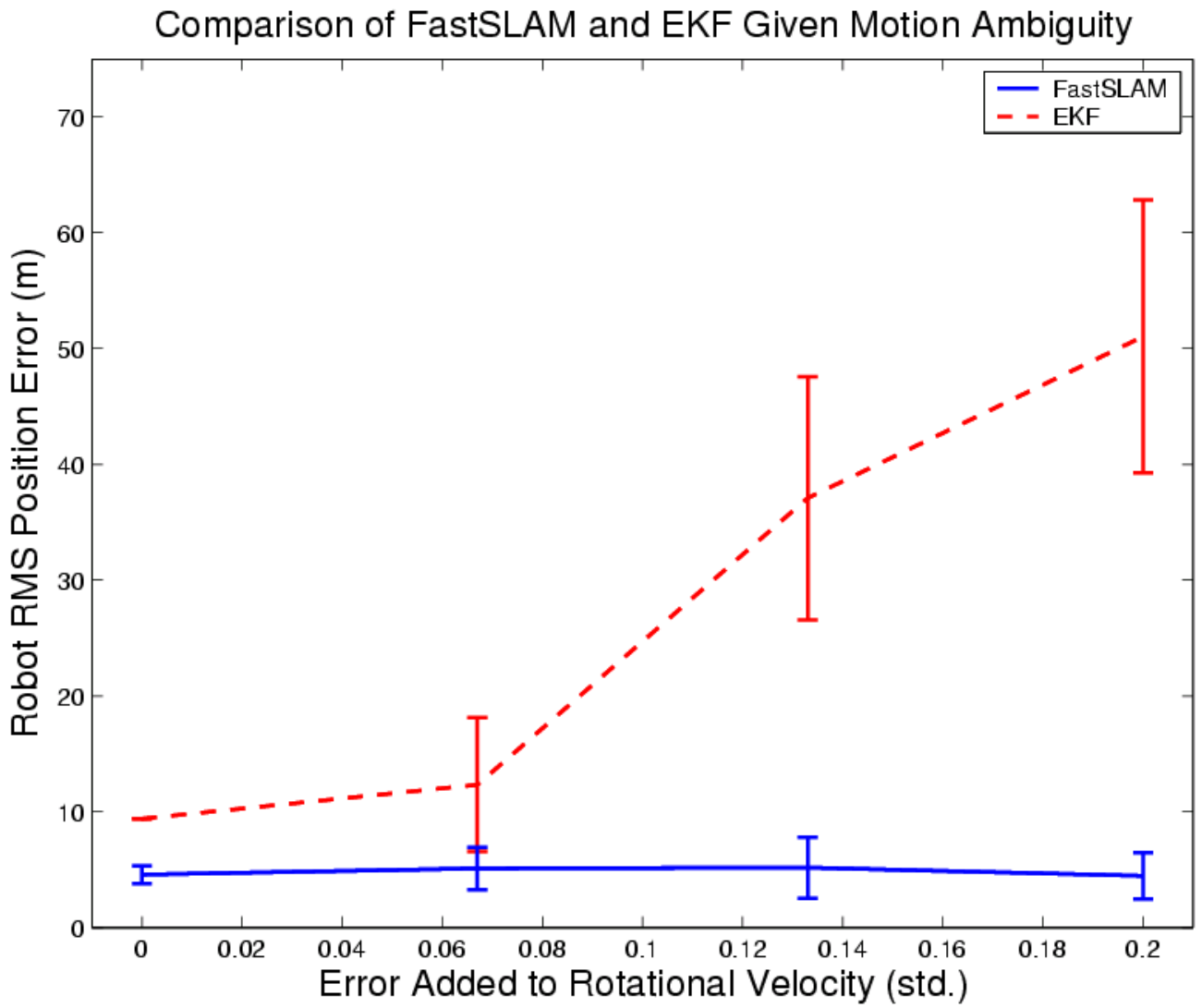
Rechtes Bild:
Blau: GPS
Rot: FastSLAM



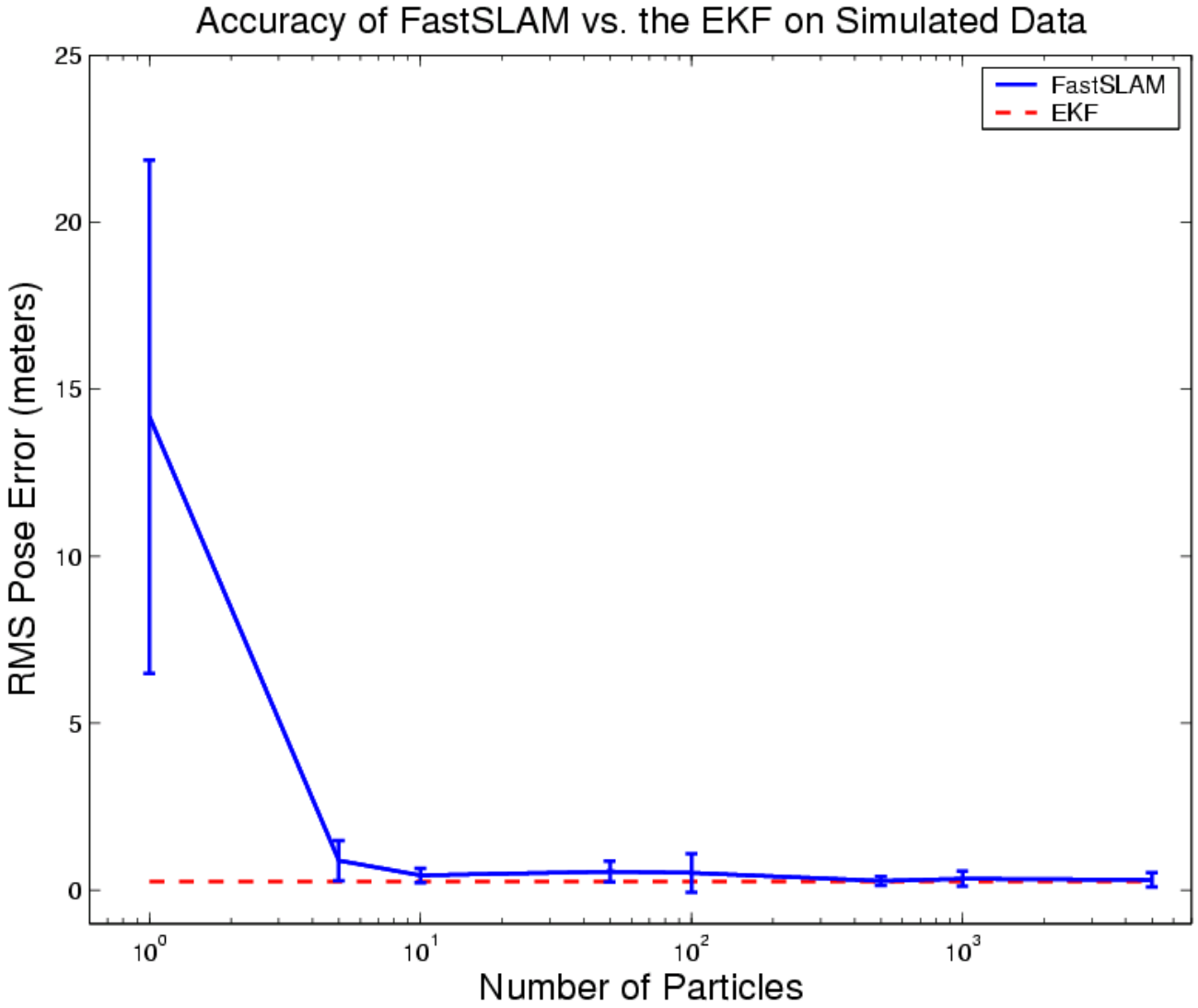
Blau: GPS

Rot: FastSLAM, wobei Odometriewerte
durch zufällige Daten ersetzte wurden.
D.h. es wurde eine extrem unzuverlässige
Odometrie verwendet.
Es zeigt sich die Robustheit des Verfahrens.

Vergleich EKF-SLAM und Fast-SLAM (1)



Vergleich EKF-SLAM und Fast-SLAM (2)



Fast-SLAM:
Synchrone Lokalisierung und Kartenerstellung mit einem Partikel-Filter

- Landmarkenbasiertes Fast-SLAM
- Gitterbasiertes Fast-Slam
- Optimierungen

Gitterbasiertes SLAM - Problemstellung

- Roboter exploriert eine unbekannte, statische Umgebung
  z.B. mit Laser-Scanner.

- Gegeben Sensor- und Steuerdaten:
  - $d = u_1, z_1, u_2, z_2, ..., u_k, z_k$
- Gesucht:
  - Belegheitsgitter m
  - Weg des Roboters: $x_1, x_2, ..., x_k$

Schlüsselüberlegung zu gitterbasiertes SLAM

Belegheitsgitter

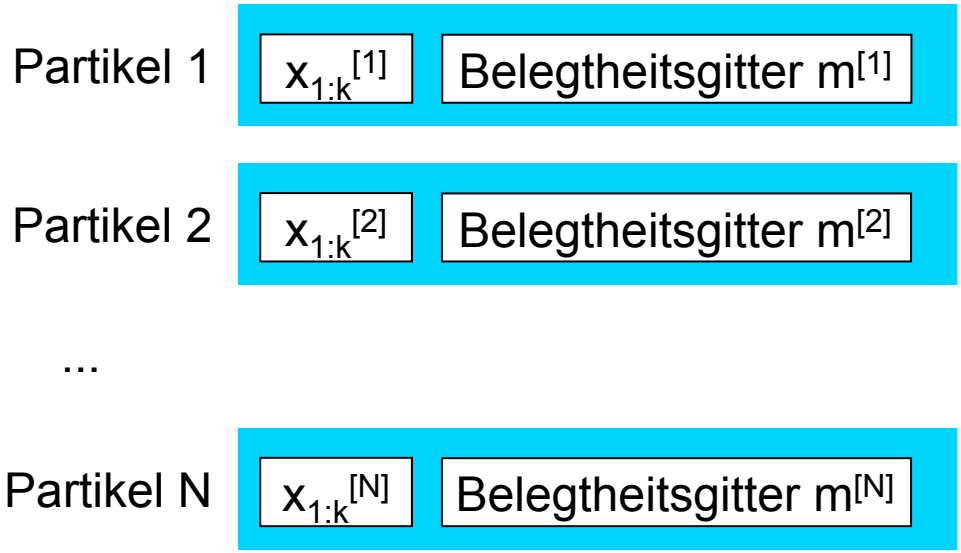
$$p(x_{1:k}, m \mid z_{1:k}, u_{1:k})$$
$$= p(x_{1:k} \mid z_{1:k}, u_{1:k}) \cdot p(m \mid x_{1:k}, z_{1:k})$$

Das ist Positionstracking. Benutze MCL.

GridMapping bei bekannten Positionen.

Gitterbasiertes SLAM-Verfahren (1)

- Verwalte zu jedem Zeitpunkt t_k Partikelmenge χ_k:



- $x_{1:k}^{[p]}$ ist der im Partikel p gespeicherte Roboterweg.

Gitterbasiertes SLAM-Verfahren (2)

Algorithmus FastSLAM_OccupancyGrids(χ_{k-1}, u_k, z_k):

for p = 1 to N do

Integration des Steuerbefehls:

$x_k^{[p]}$ = sampleMotionModel(u_k,$x_{k-1}^{[p]}$);

Gewicht für jeden Partikel brechnen:

$w_k^{[p]}$ = measurementModel(z_k, $x_k^{[p]}$, $m_{k-1}^{[p]}$);

Integration des Sensordaten:

$m_k^{[p]}$ = updateOccupancyGrid($m_{k-1}^{[p]}$, $x_k^{[p]}$, z_k);

endfor

Resampling:

$\chi_k = \emptyset$;

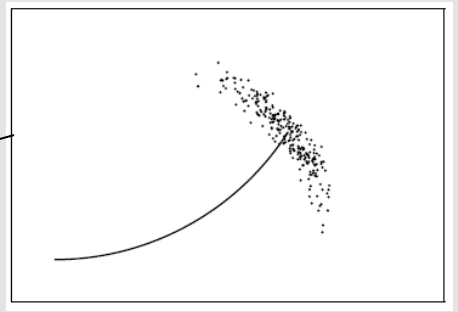
for i = 1 to N do

ziehe p zufällig mit Wahrscheinlichkeit $w_k^{[p]}$;

$\chi_k = \chi_k \cup \{ <x_{1:k}^{[p]}, m_k^{[p]} > \}$;

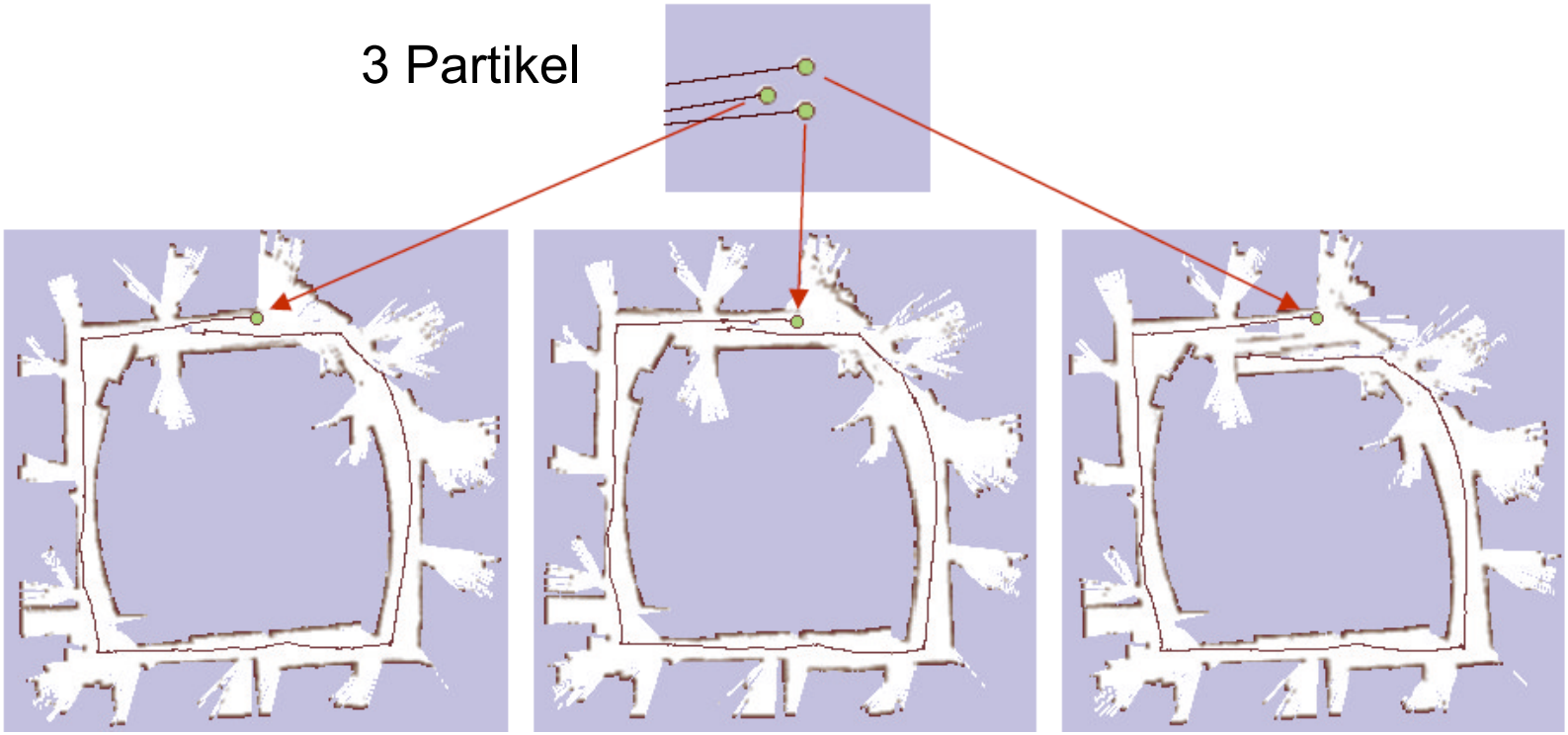
endfor

return χ_k;



wie beim MCL-Algorithmus aus Kap. 8

Beispiel

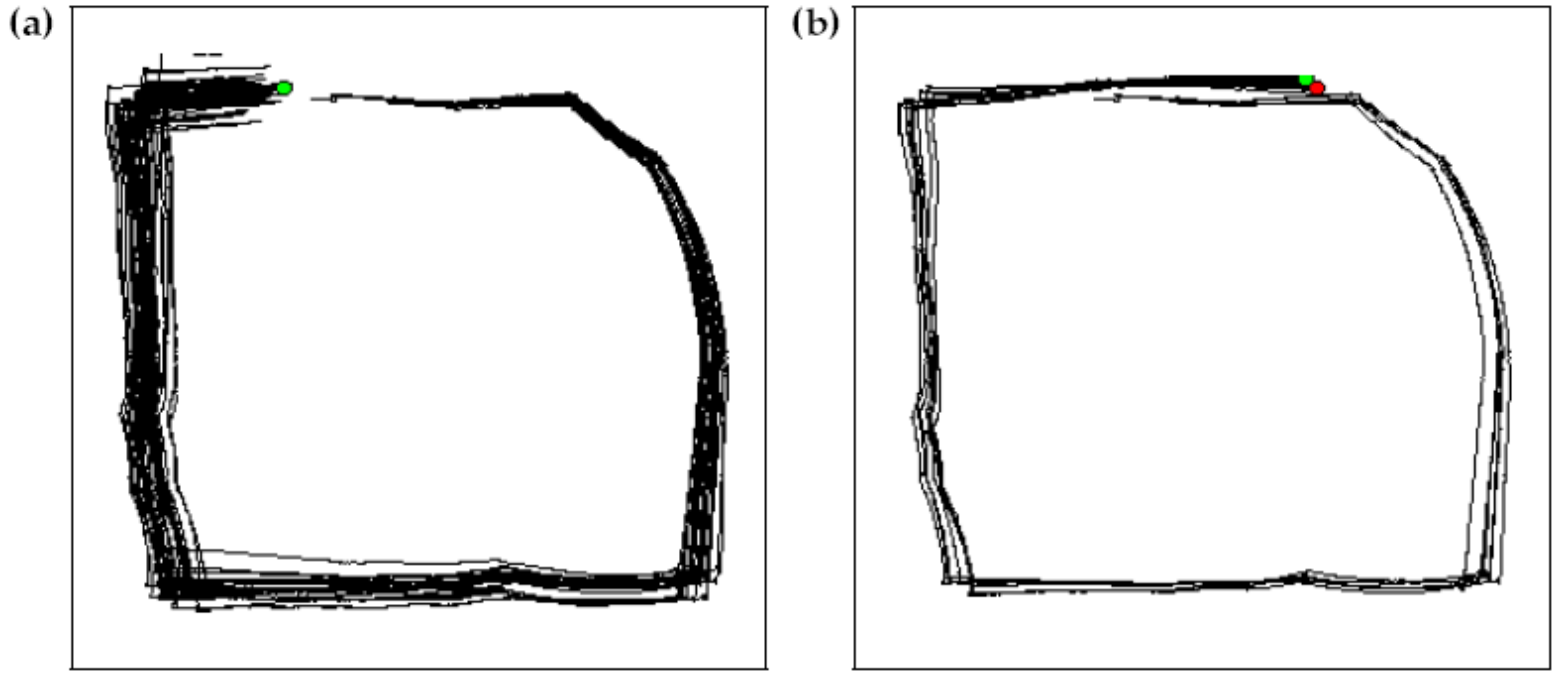


Roboterweg und Karte von Partikel 1

Roboterweg und Karte von Partikel 2

Roboterweg und Karte von Partikel 3

Beispiel



(a) Aufgrund der langen Strecke steigt die Unsicherheit in der Position.
Die Partikel spreizen stark.

(b) Die Schleife wird geschlossen.
Der Roboter gelangt in einem Kartenbereich, der mit größerer Genauigkeit zu Beginn aufgenommen wurde.
Viele unwahrscheinliche Partikel fallen beim Resampling weg.
Die Partikel spreizen jetzt weniger stark.

Optimierungen

Problem:

- bei der gitterbasierten Kartenerstellung kann ein Gitter sehr groß werden.
- Jeder Partikel enthält ein eigenes Belegtheitsgitter.
- Daher müssen die Anzahl der Partikel klein gehalten werden!

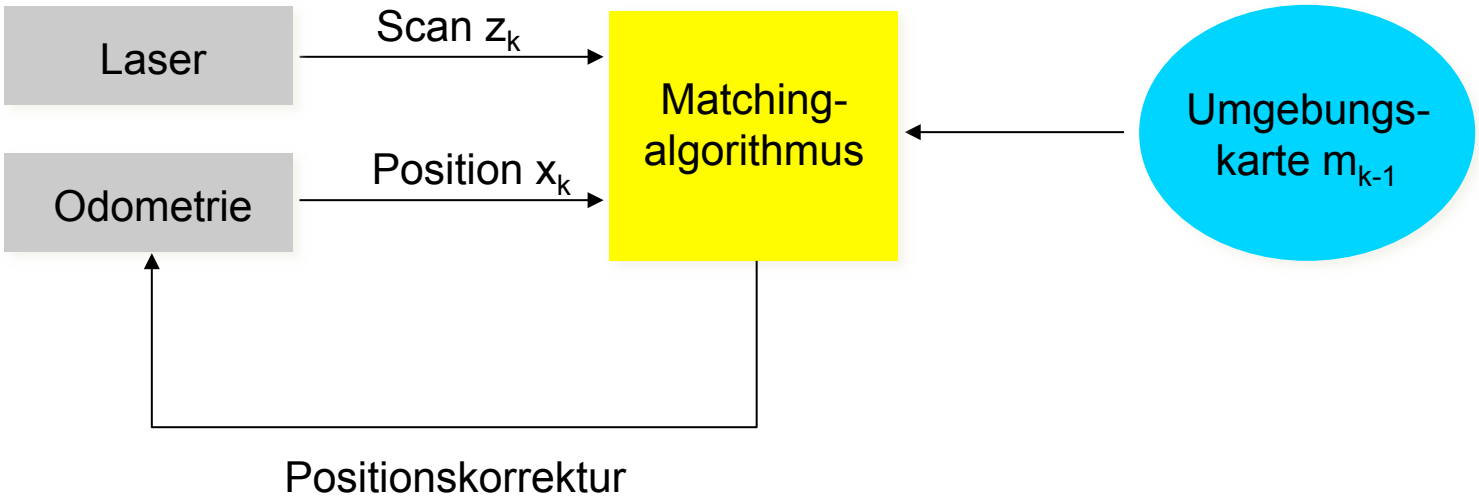
Lösungsansätze:

- Verbesserung der Odometrie mit Scan-Matching
- Verbesserung von sampleMotionModel: auf Sensordaten ausgerichtete Generierung von Positionen

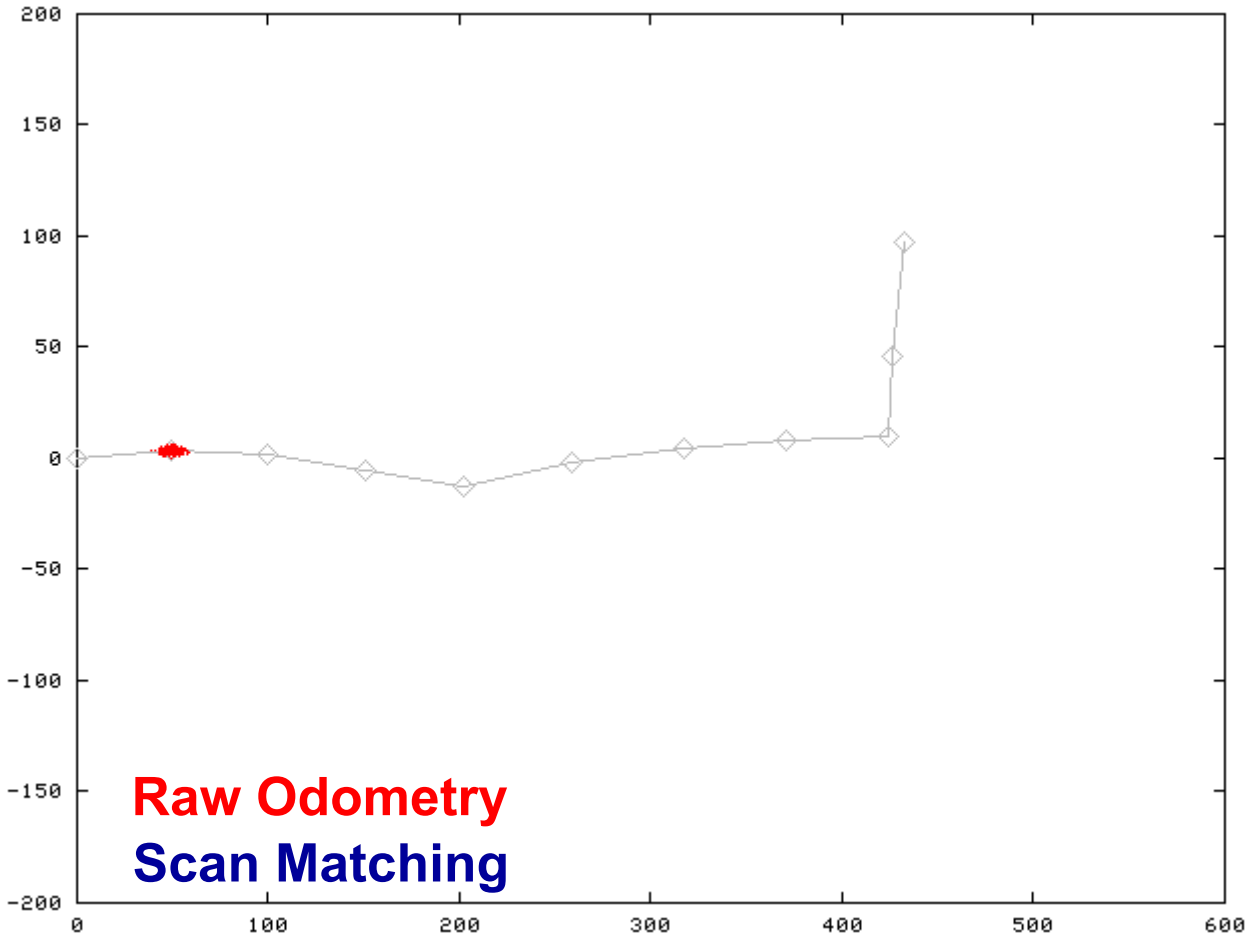
Verbesserung der Odometrie durch Scanmatching

Idee:

verbessere die durch Odometrie ermittelte Position x_k, indem der aktuelle Laserscan z_k mit der bisherigen Karte m_{k-1} abgeglichen wird.

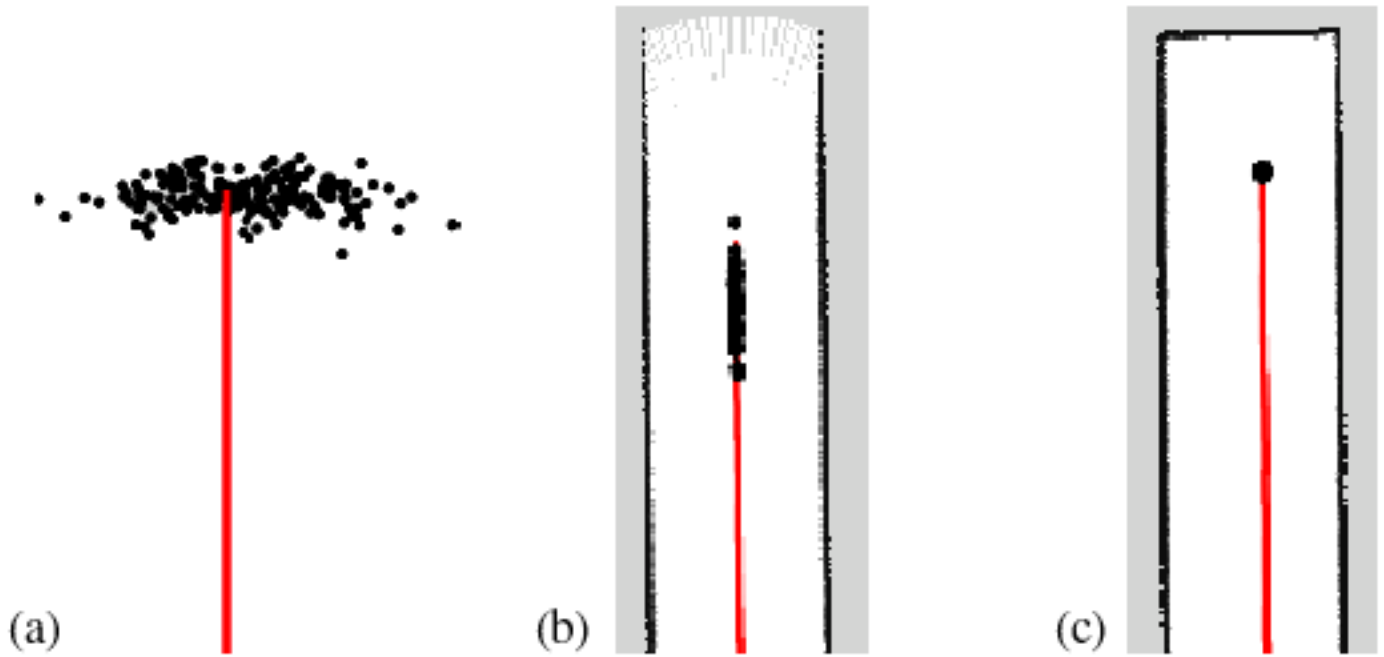


Illustrierung der Positionsverbesserung beim Scanmatching



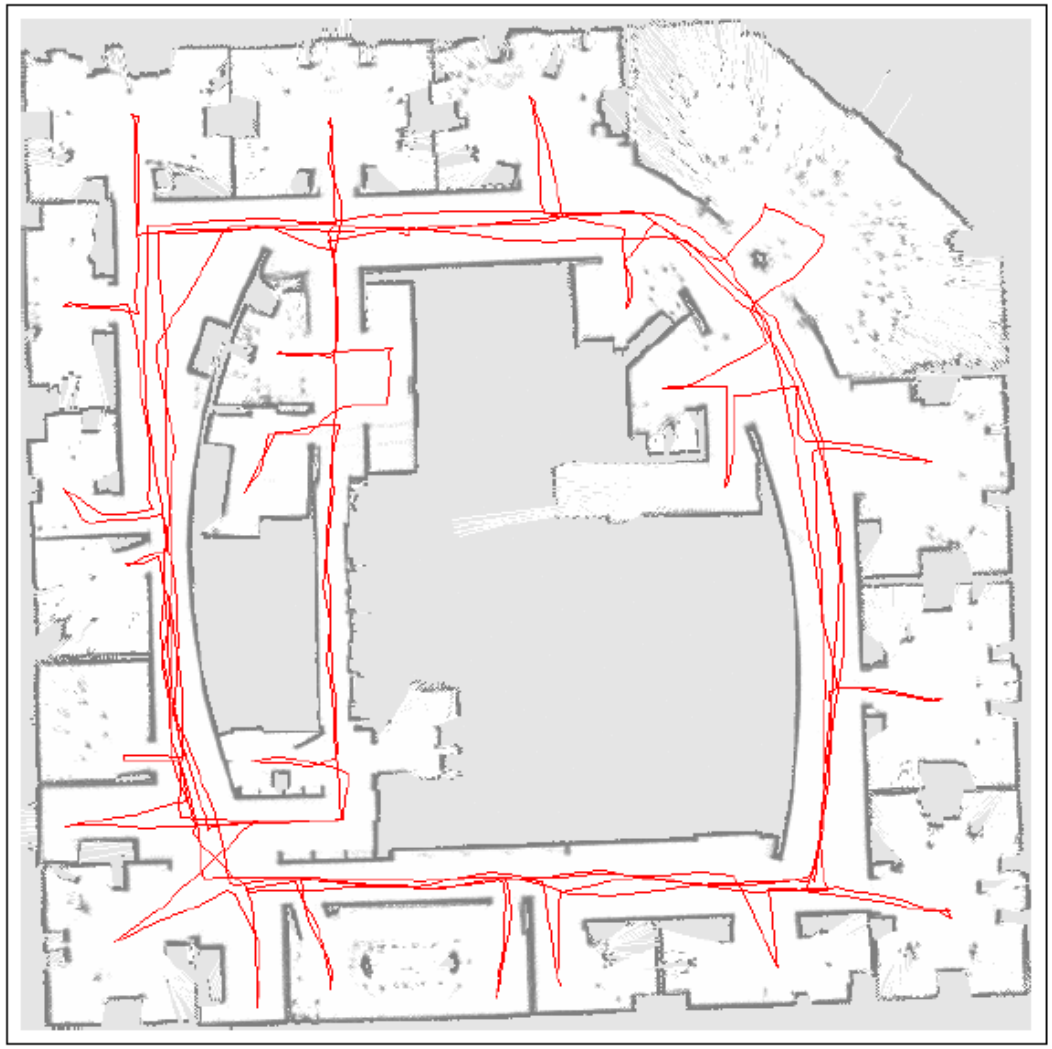
Verbesserung von sampleMotionModel

- von Sensordaten (d.h. Umgebung) abhängige Generierung von Positionen



Intel-Lab mit Standard-FastSlam (1)

- Belegheitsgitter für den Partikel mit höchsten (akkumuliertem) Gewicht
- 500 Partikel
- Roboterpfad: 491m
- Umgebung: 28m * 28m
- Durchschnitts-geschwindigkeit: ca. 0.2 m/sec



Intel-Lab mit Standard-FastSlam (2)

- Darstellung der reinen Odometriedaten und des damit erzeugten Belegtheitsgitters

Intel-Lab mit FastSlam und Scanmatching

- 15 Partikel
- 5 cm Gitter-Auflösung während des Scanmatchings
- 1 cm Auflösung in der endgültigen Karte

Outdoor Campus Map

- 30 Partikel
- 250m x 250m
- ca. 1,7 km (Odometrie)
- 20 cm Auflösung während des Scanmatchings
- 30 cm Auflösung in der endgültigen Karte