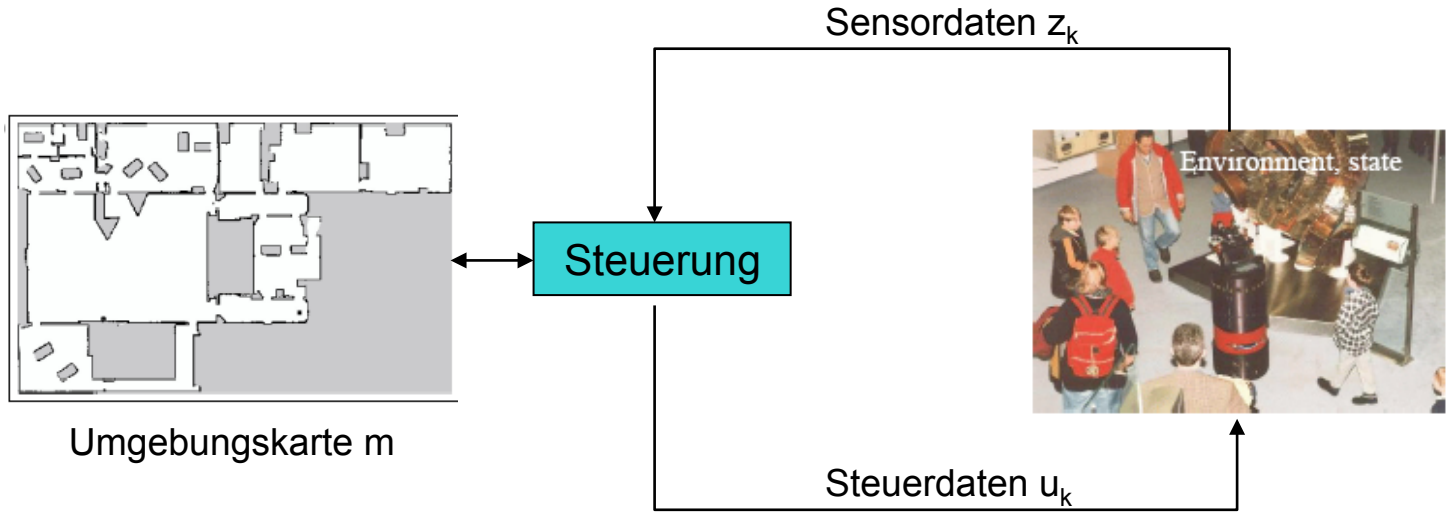
Monte-Carlo-Lokalisierung

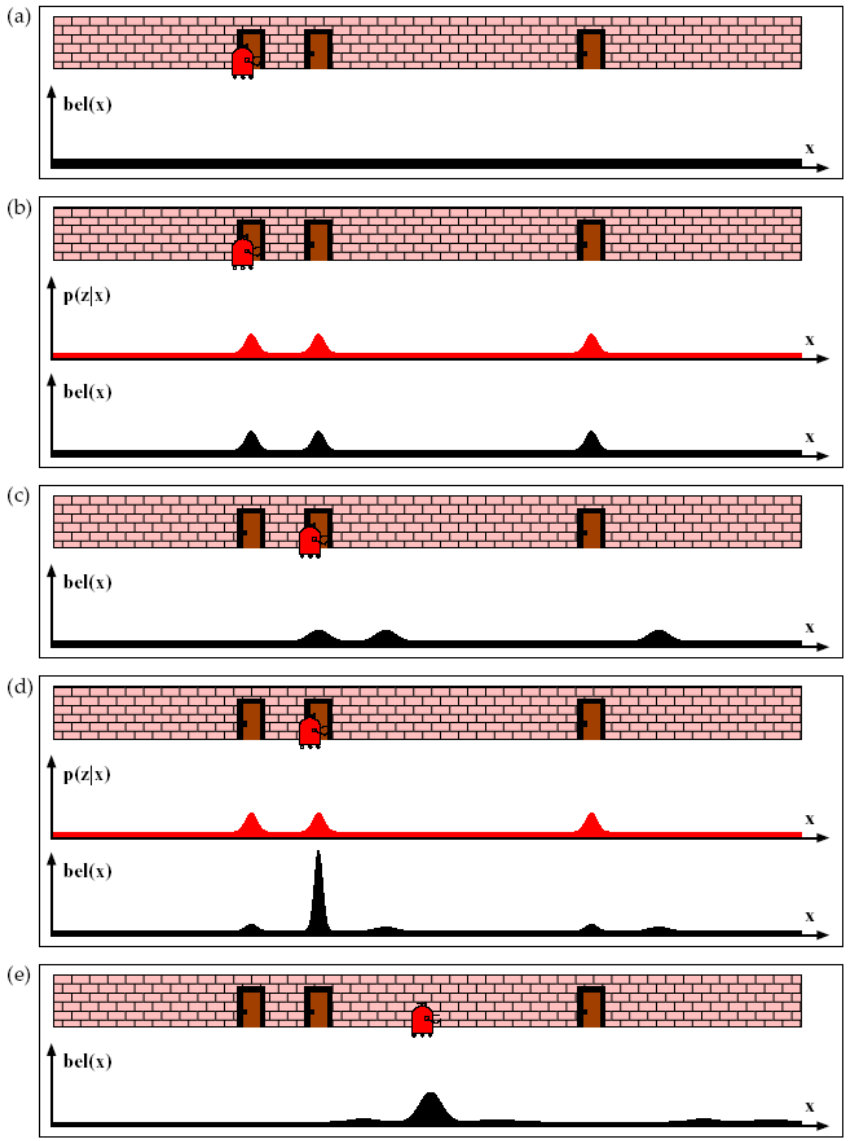
- Einleitung
- Modellierung von Unsicherheit mit Partikelmengen
- Algorithmus
- Beispiele
- Varianten

Allgemeine Problemstellung



- **Sensordaten**
  Der Roboter bekommt regelmäßig Sensordaten z_k - z.B. Abstandsinformationen.
  Sensordaten sind mit Unsicherheit behaftet!
- **Steuerdaten**
  Der Roboter führt regelmäßig Bewegungen mit den Steuerdaten u_k durch.
  Bewegungen entsprechen nicht exakt den vorgegebenen Steuerdaten!
- **Position und Umgebungsmodell**
  Aufgrund der Sensor- und Steuerdaten schätzt der Roboter seine globale Position $x_k = (x, y, \theta)$ bzgl. seiner Umgebungskarte m.
  Auch m enthält Ungenauigkeiten!

Wahrscheinlichkeitsbasierte Ansätze



- Roboterposition wird durch eine Wahrscheinlichkeitsverteilung bel(x) modelliert (bel = belief, Gewissheit).

  In (a) gibt es keine Gewissheit in der Position.

  In (e) wird die tatsächliche Position richtig geschätzt.

- Integration von Sensorwerte in die Positionsschätzung in (b) und (d):

  Der Roboter erkennt eine Türe und schliesst daraus mit Hilfe eines Umgebungs- und Sensormodells, welche Positionen am wahrscheinlichsten sind und verbessert damit sein Positionsschätzung.

- Vorhersage einer neuen Positionsschätzung mit Steuerdaten in (c) und (e):

  da die Steuerdaten unsicher sind, verschlechtert sich die Positionsschätzung.

Varianten

- **Lokale Selbstlokalisierung**
  - Die initiale Position des Roboters ist ungefähr bekannt.
  - Ziel ist es, die Position neu zu berechnen, sobald sich der Roboter bewegt (position tracking).

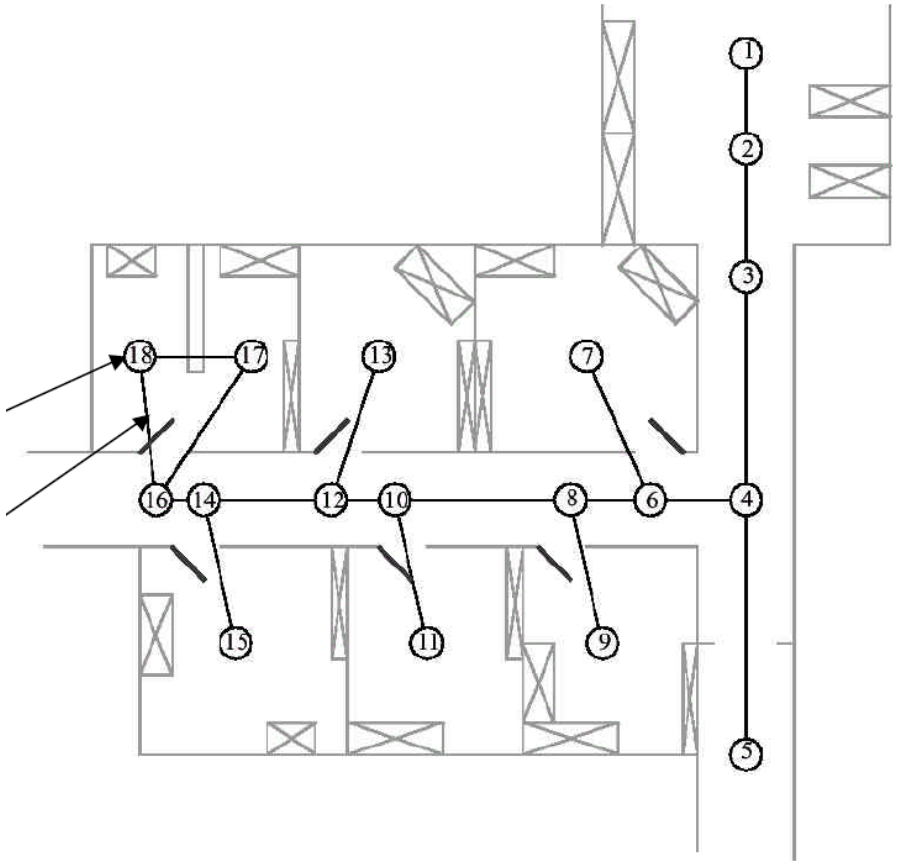
- **Globale Selbstlokalisierung**
  - Die initiale Position des Roboters ist nicht bekannt.
  - Ziel ist es, die Position aufgrund von Roboterbewegungen und neuen Sensordaten zu finden.
  - Schwieriger als lokale Selbstlokalisierung.
  - Als zusätzliche Problemstellung kann dazukommen, dass der Roboter während der Lokalisierungsphase willkürlich an eine andere Position platziert werden kann (kidnapped robot problem)

Verfahren

Verfahren	Koppel-navigation	Kalman-Filter	Gitterbasierte Lokalisierung	Monte-Carlo-Lokalisierung
Lokalisierungs-variante	lokal	lokal	lokal und global	lokal und global
Positionsschätzung	NDF	NDF	Gitter	Partikelmenge
Vorhersage mit Steuerdaten	ja	ja	ja	ja
Integration von Sensordaten mit Hilfe einer Umgebungskarte	nein	ja	ja	ja
Karte	nein	ja	ja	ja
Lokalisierungs-genauigkeit	Drift	sehr genau	genau	genau

Metrische und topologische Karten

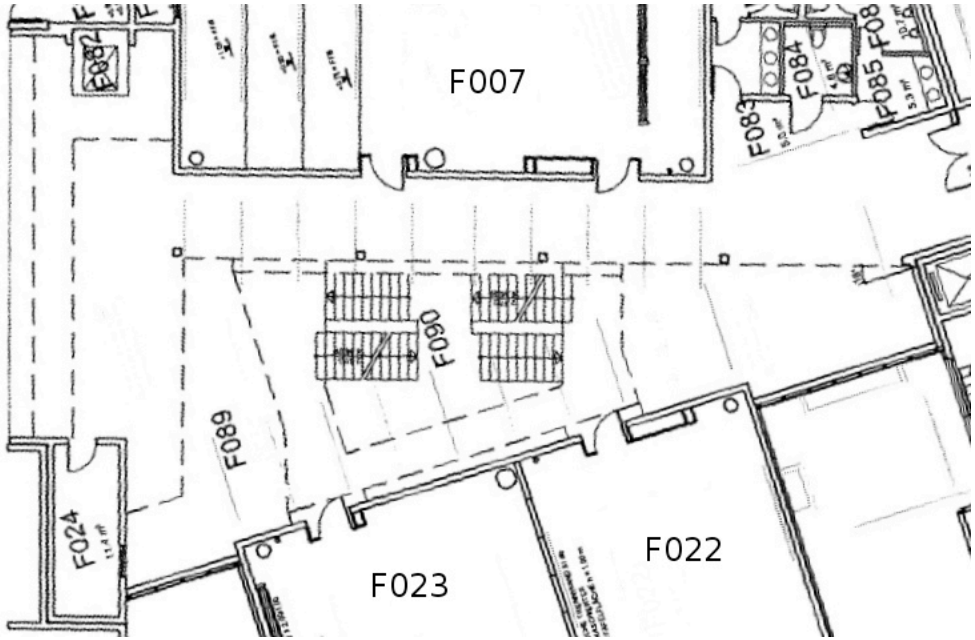
- Metrische Karten sind in einem KS eingezeichnet
- Topologische Karten beschreiben Nachbarschaftbeziehungen von relevanten Orten (Raum, Tür, Flur, ...) in der Umgebung als Graph



Gitterbasierte Karten

- Belegtheitsgitter
  schwarz = belegt, weiss = frei, grau = unbekannt.
- Je nach Auflösung sehr detailliert und speicheraufwendig.
- Typisches Ergebnis bei autonomer Kartenerstellung.
- Oft verwendet bei Planungs- und Navigationsverfahren.

Linienbasierte Karten



- Vektorgraphik.
- Wenig speicheraufwendig.
- Autonome Erstellung aufwendig.

Punktbasierte Karten

- Karte besteht aus einer Menge von Hindernispunkten.
- Wenig speicheraufwendig.
- Typisches Ergebnis bei autonomer Kartenerstellung.
- Hier: aus Laserscans zusammengesetzt.
  (Laserscan sind mit einem Laser aufgenommene 180°-Abstandsprofile).

Monte-Carlo-Lokalisierung

- Einleitung
- Modellierung von Unsicherheit mit Partikelmengen
- Algorithmus
- Beispiele
- Varianten

Idee der Monte-Carlo-Lokalisierung

Positionschätzung bel(x_k) wird durch ein Menge von Partikeln (vertikale Striche) dargestellt.

(a) Keine Information über die Anfangsposition; Partikel sind über alle x-Werte zufällig verteilt.

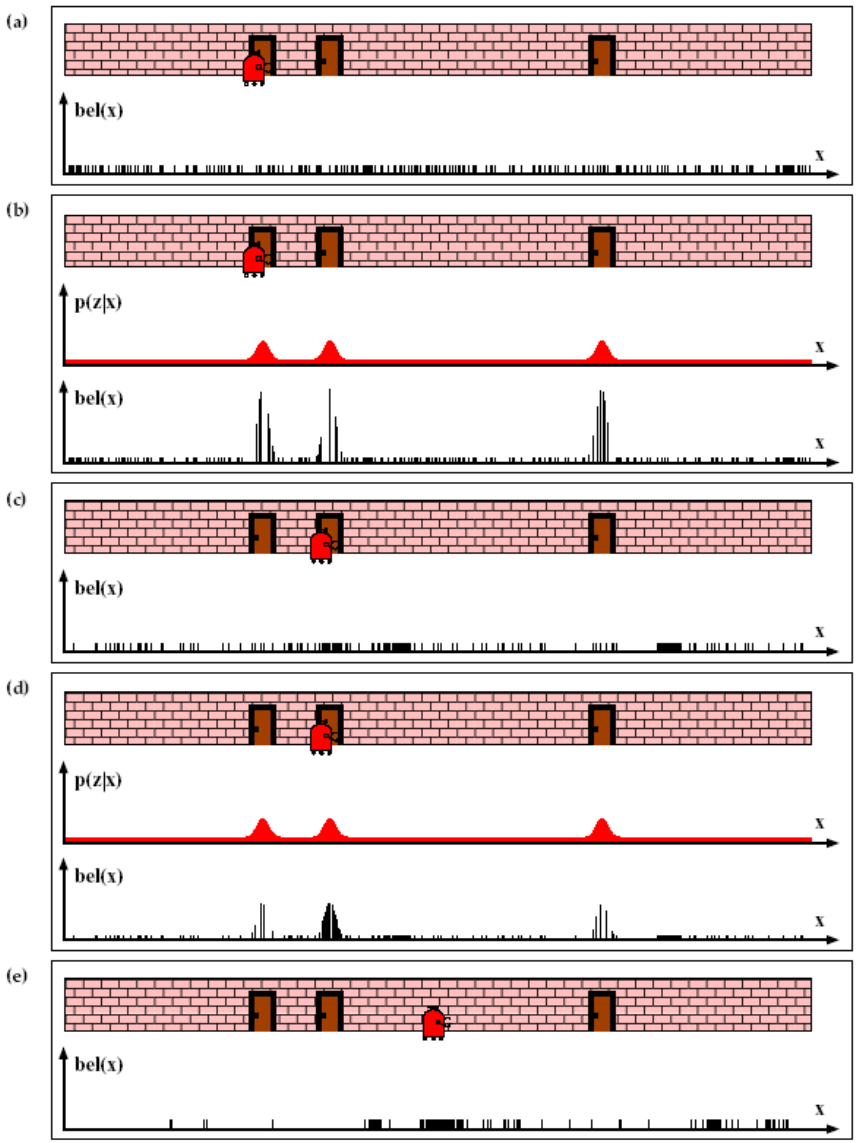
(b) Gewichtung:
Durch eine Sensormessung z werden die Gewichte (Strichhöhe) verändert.

(c) Resampling:
Aus der Partikelmenge werden zufällig aber entsprechend ihrem Gewicht Partikeln gezogen.

Anschließend wird der Steuerbefehl (Bewegung) u_k integriert.

(d) erneute Gewichtung mit neuem Sensorwert

(e) erneutes Resampling und Integrierung des Steuerbefehls

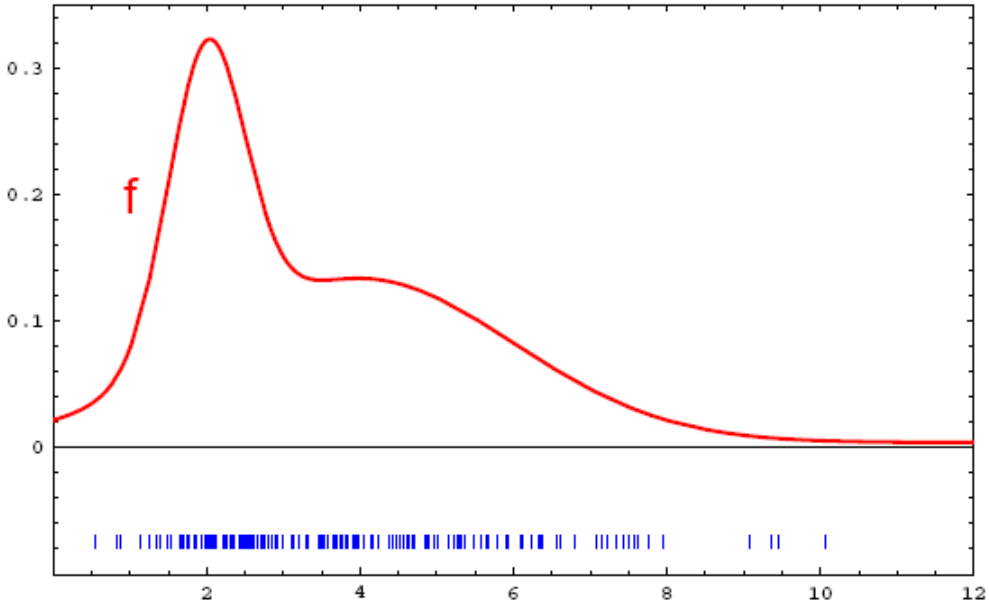


Partikelmengen

- Jeder Partikel stellt eine Hypothese (d.h. einen konkreten Wert) für den Zustand x dar.
- Partikelmenge:

  $\chi = \{x^{[1]}, x^{[1]}, .., x^{[M]}\}$,

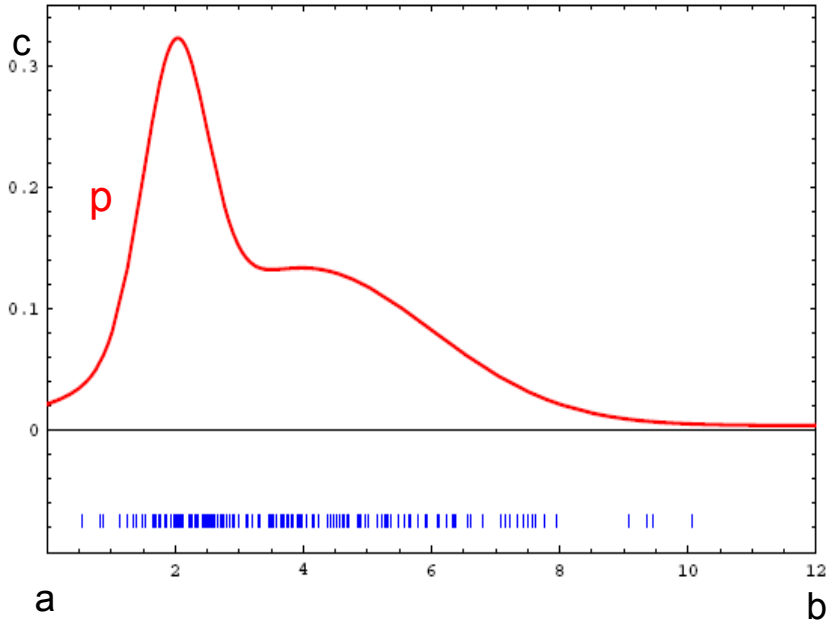
  wobei $x^{[m]}$ ein Partikel ist.
- M ist üblicherweise groß (z.B. M = 1000)



Generierung von Partikelmengen aus Dichten

- Generererierung eine Partikelmenge χ aus einer Wahrscheinlichkeitsdichte p:

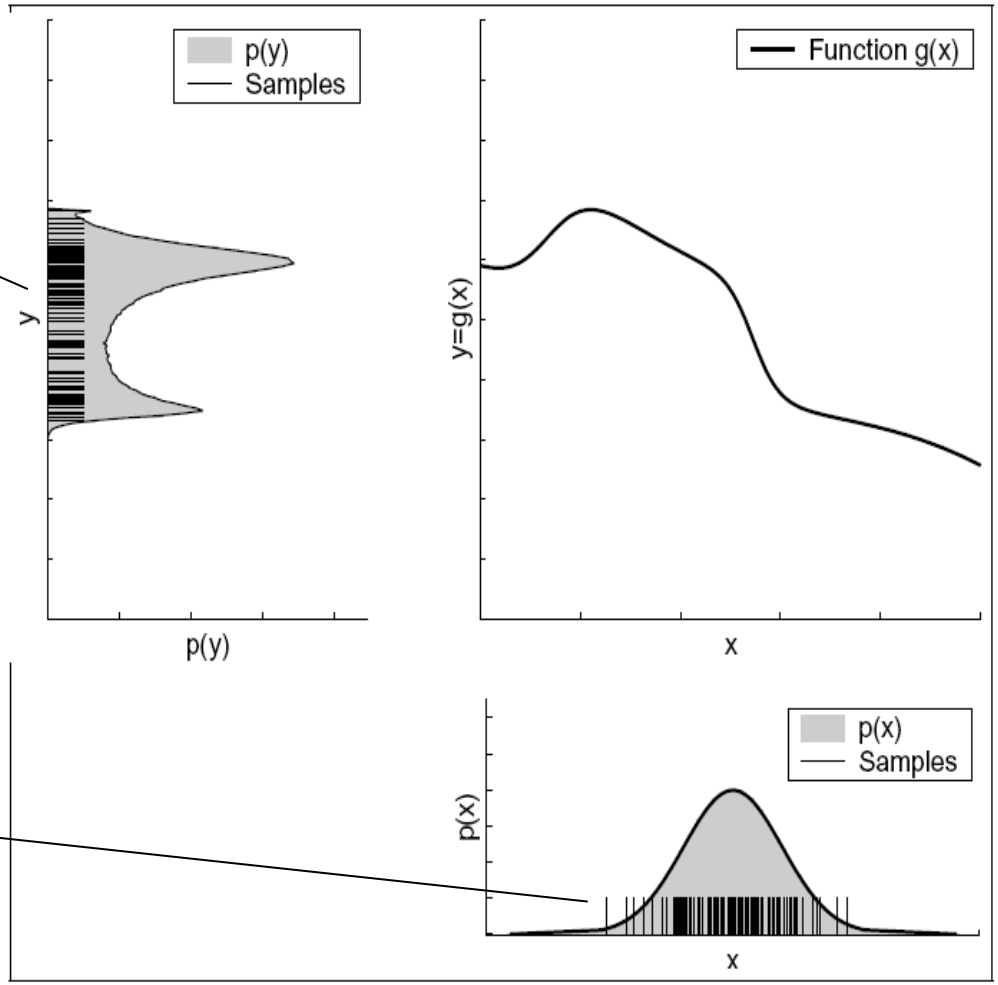
```
Algorithm generateParticle(p):
    χ = ∅;
    i = 0;
    while i < M do
        generiere Zufallszahl x aus [a,b];
        generiere Zufallszahl q aus [0,c];
        if q < p(x) then
            i = i+1;
            χ = χ ∪ {x};
        endif
    endwhile
return χ;
```



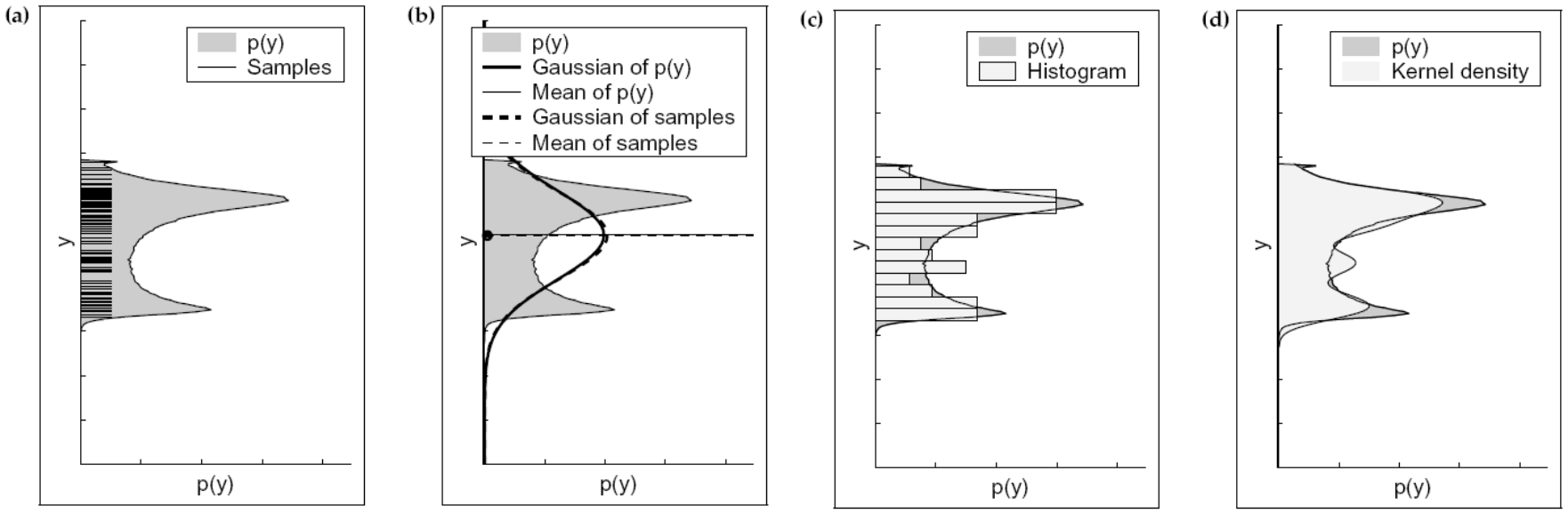
Transformation einer Partikelmenge

Wahrscheinlichkeitsverteilung und Partikelwolke nach Anwendung der Funktion g.

Menge von Partikeln die gemäß der Normalverteilung p(x) zufällig generiert wurden.



Generierung von Dichten aus Partikelmengen



(a) Menge von Partikeln die gemäß der Verteilung p(x) zufällig generiert wurden.

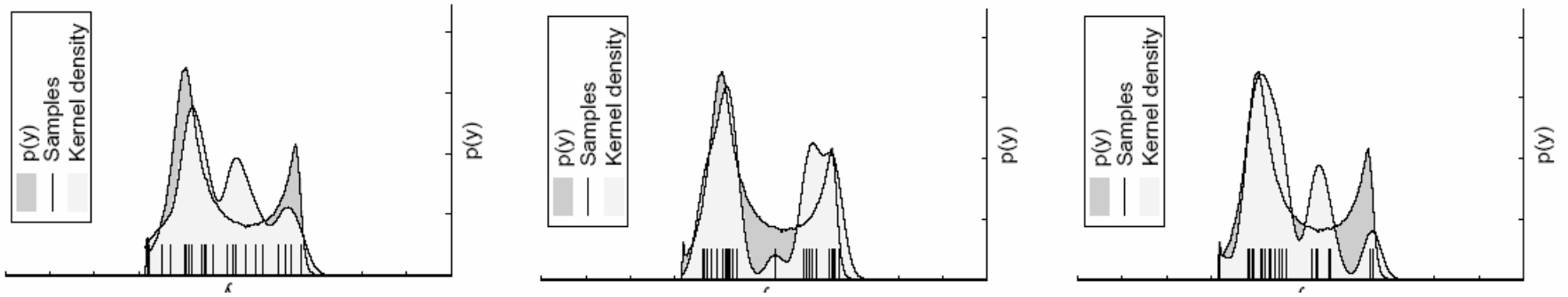
(b) Aus der Partikelmenge gewonnene Normalverteilung, indem aus der Partikelmenge Mittelwert und Varianz ermittelt wird.

(c) Aus Partikelmenge gewonnenes Histogramm.

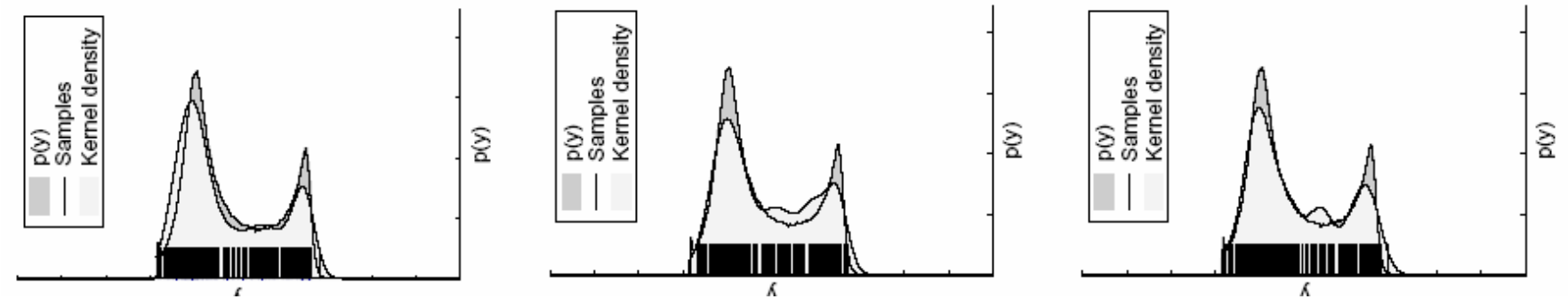
(d) Stetige Variante (Kerndichteschätzer, kernel density estimation)
Bilde Summe aus Normalverteilungen, wobei jeder Partikel eine Normalverteilung bildet.

Genauigkeit der Verteilungsapproximation in Abh. von der Partikelanzahl

- 25 Partikel



- 250 Partikel



Beispiel

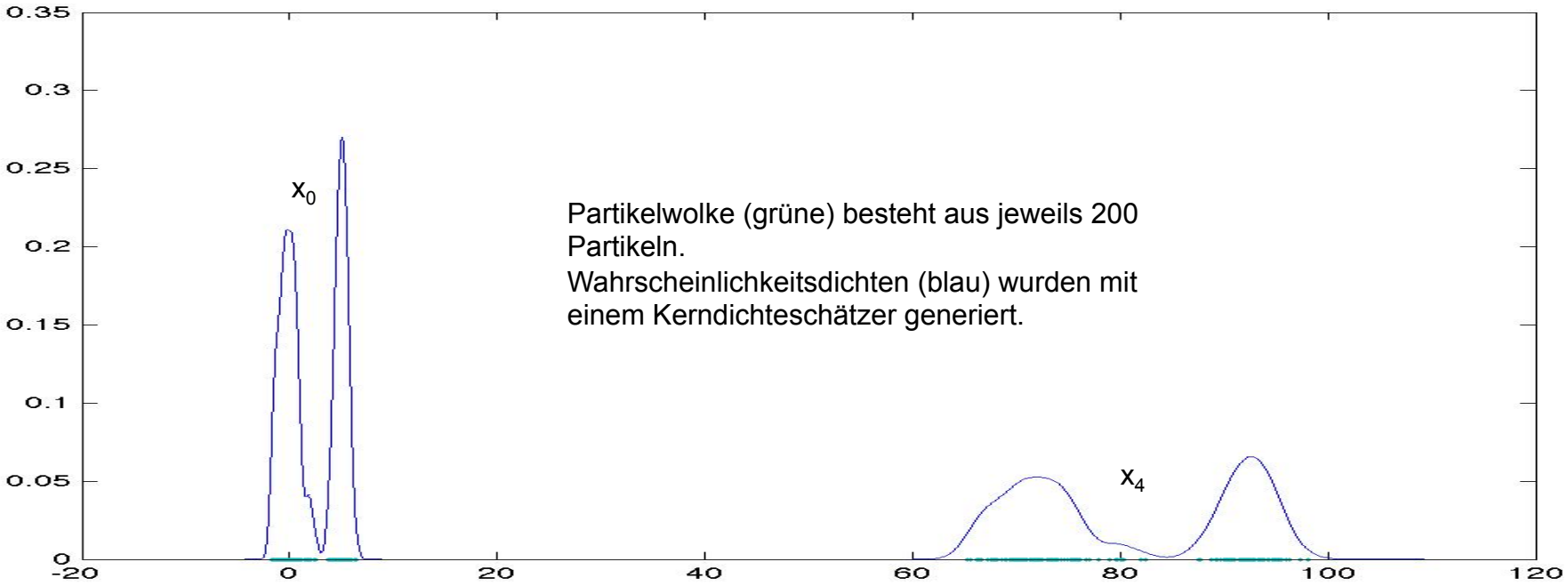
- lineares, zeitdiskretes System:

$$x_{k+1} = \sqrt{2}x_k + 10$$

- Anfangszustand x_0 ist nur mit einer gewissen Unsicherheit bekannt:

$$x_0 \sim 0.5 * N(x_0;0,1) + 0.5 * N(x_0;5,0.25)$$

- Partikelmenge wird gemäß der Verteilung des Anfangszustands generiert.
- Auf jeden Partikel wird solange Systemgleichung angewendet, bis x_4 erreicht wird.



Monte-Carlo-Lokalisierung

- Idee
- Modellierung von Unsicherheit mit Partikelmengen
- Algorithmus
- Beispiele
- Varianten

Monte-Carlo-Lokalisierung (MCL)

Algorithm **MCL**(χ_k, u_k, z_{k+1})

$\chi_{k+1}' = \chi_{k+1} = \varnothing$;

Integration des Steuerbefehls u_k und
Gewichtung mit Sensorwert z_{k+1}:

for i = 1 to M do

$x_{k+1}[i]$ = sampleMotionModel(u_k,$x_k[i]$);

w_i = measurementModel(z_{k+1},$x_{k+1}[i]$,m);

$\chi_{k+1}' = \chi_{k+1}' \cup \{(x_{k+1}[i] , w_i)\}$;

endfor

Resampling:

for i = 1 to M do

ziehe i zufällig mit Wahrscheinlichkeit w_i;

$\chi_{k+1} = \chi_{k+1} \cup \{x_{k+1}[i]\}$;

endfor

return χ_{k+1};

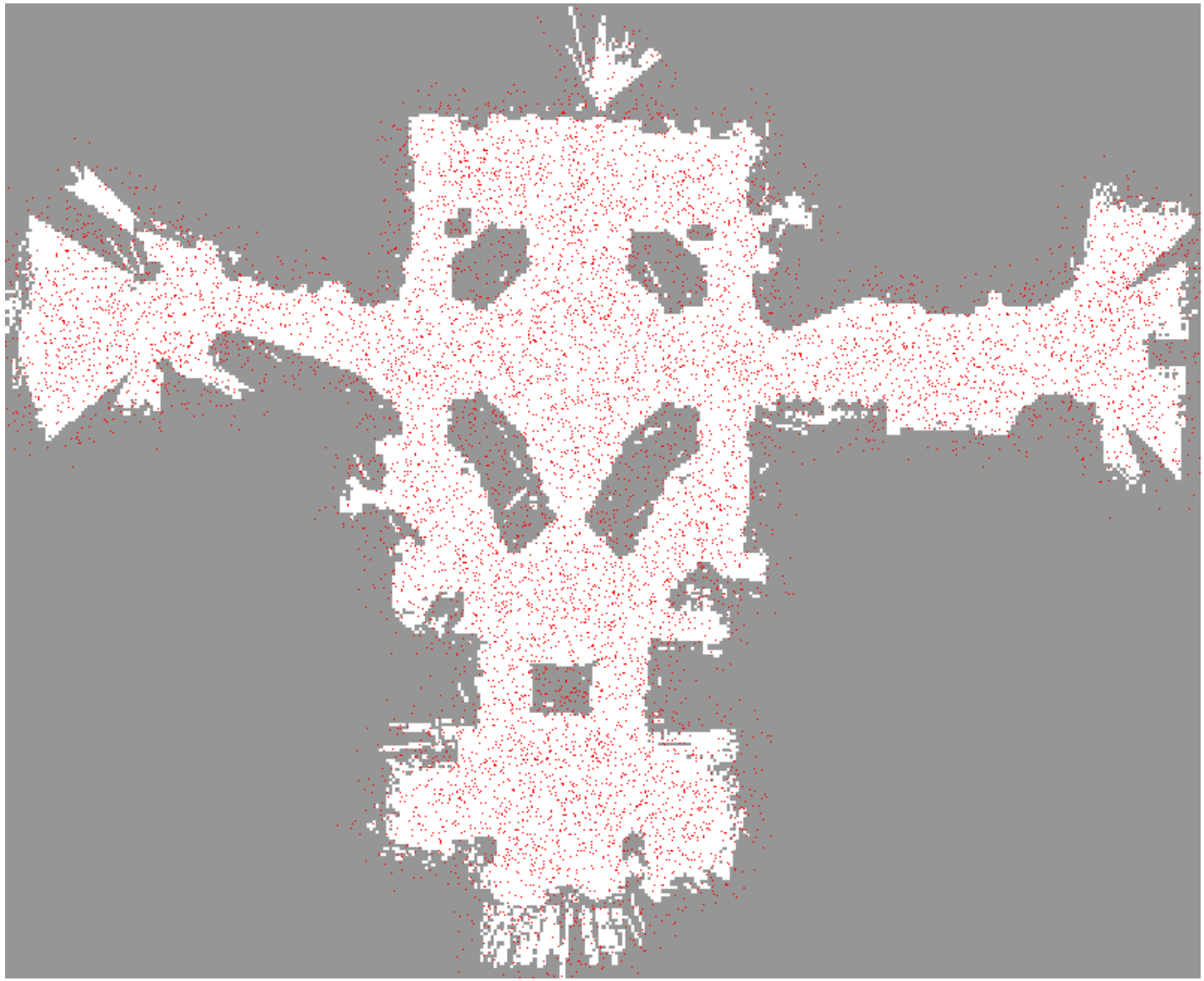
Positionsschätzung dargestellt durch Partikelmenge $\chi_k = \{x_k[1], x_k[2], .., x_k[M]\}$.

Bewegungsmodell:
Hier wird auf Partikel $x_k[i]$ ein zufällig generierter Steuerbefehl angewendet.

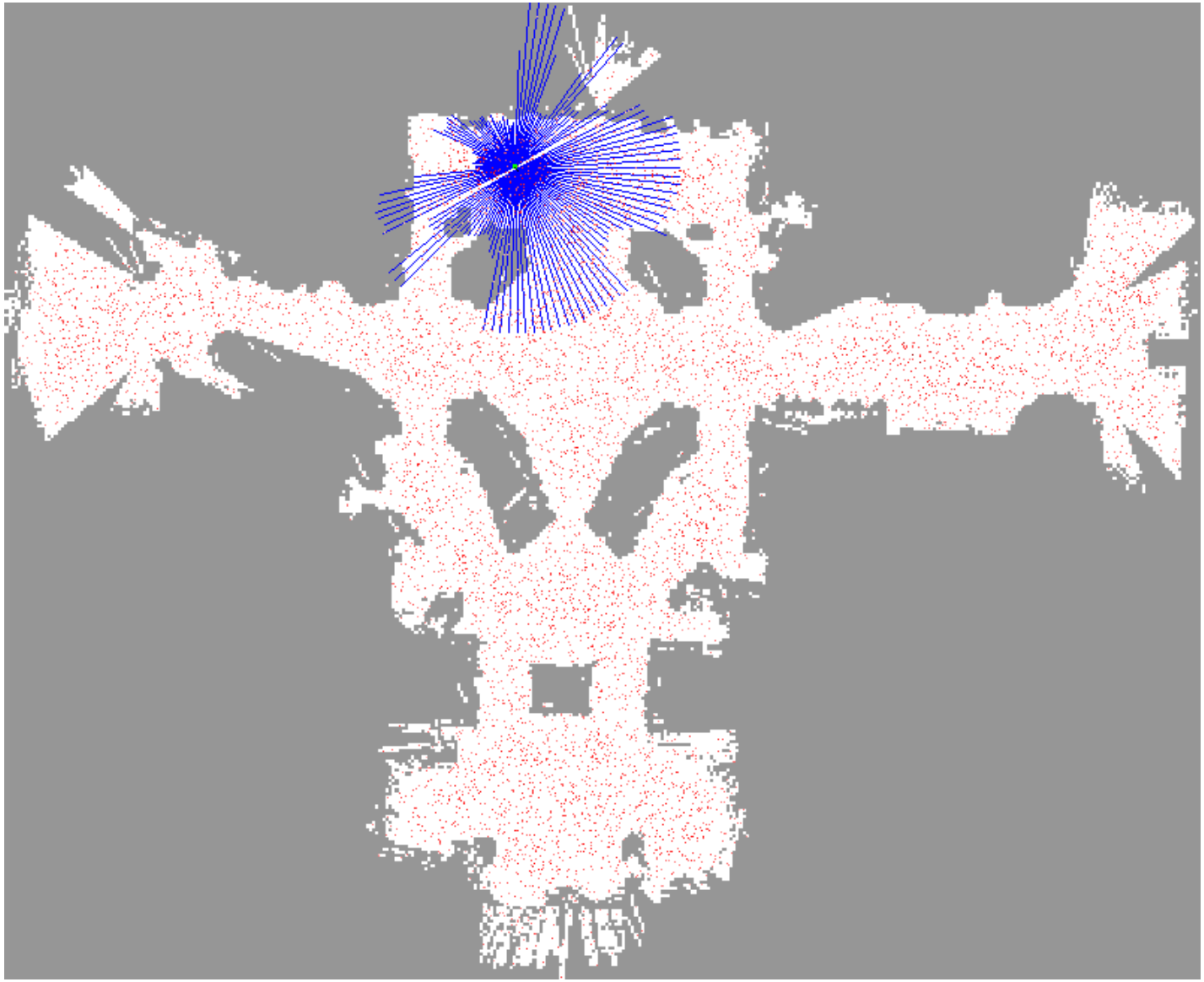
Messmodell:
Hier wird geprüft, wie gut die gemessenen Sensorwerte z_{k+1} zur Position $x_{k+1}[i]$ in einer Umgebungskarte (map) m passen.

Neue Positionsschätzung

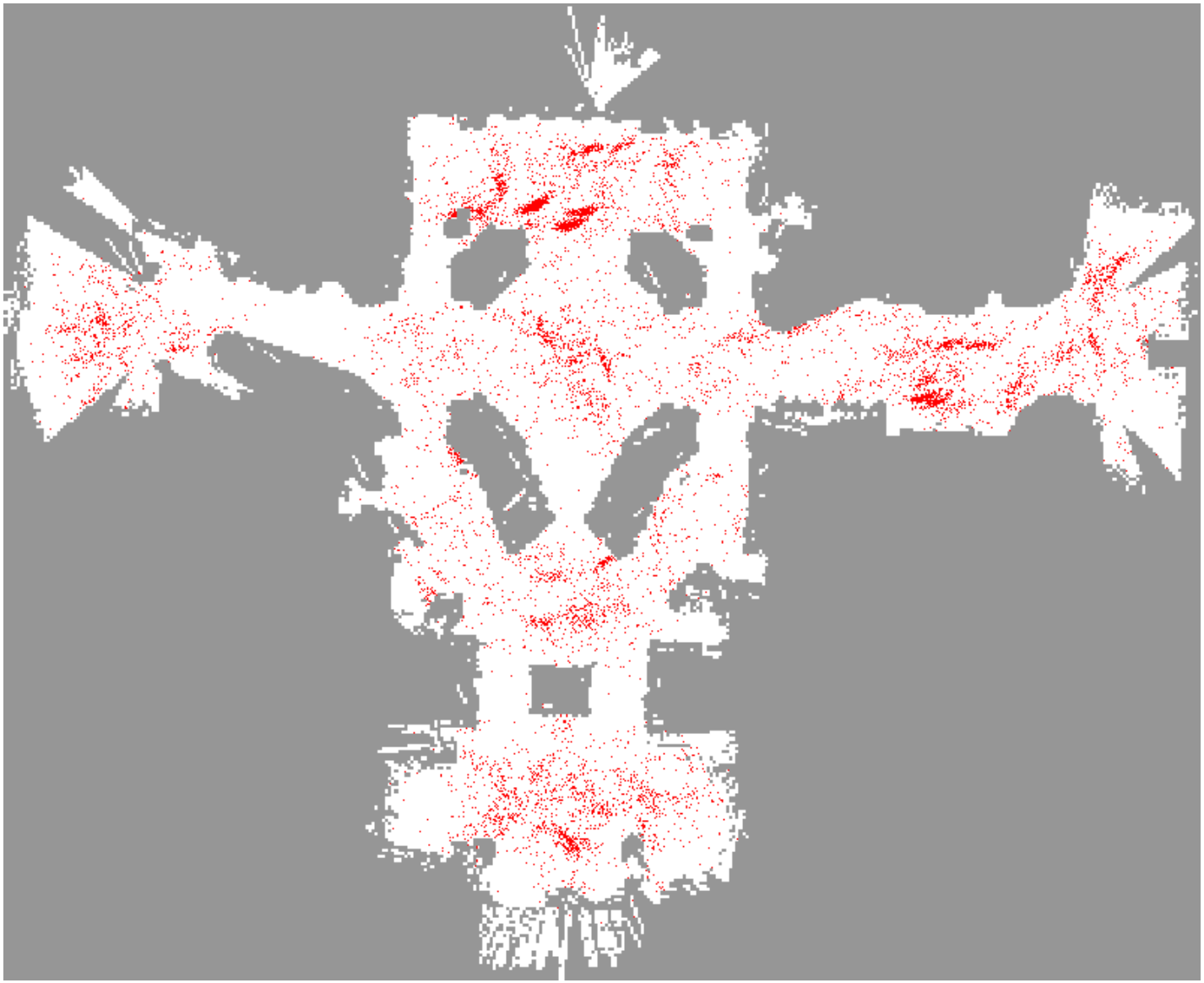
Beispiel (1)



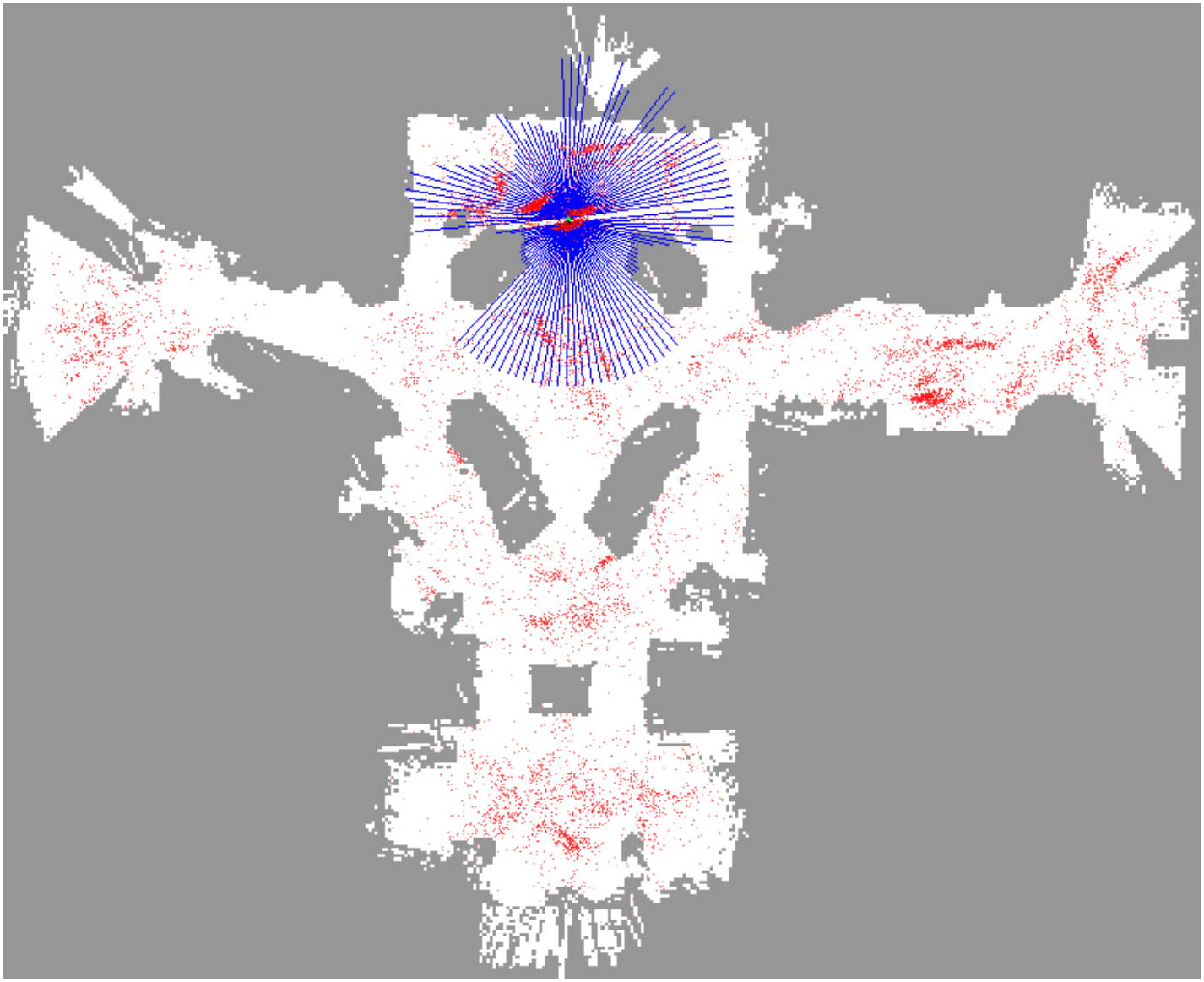
Beispiel (2)



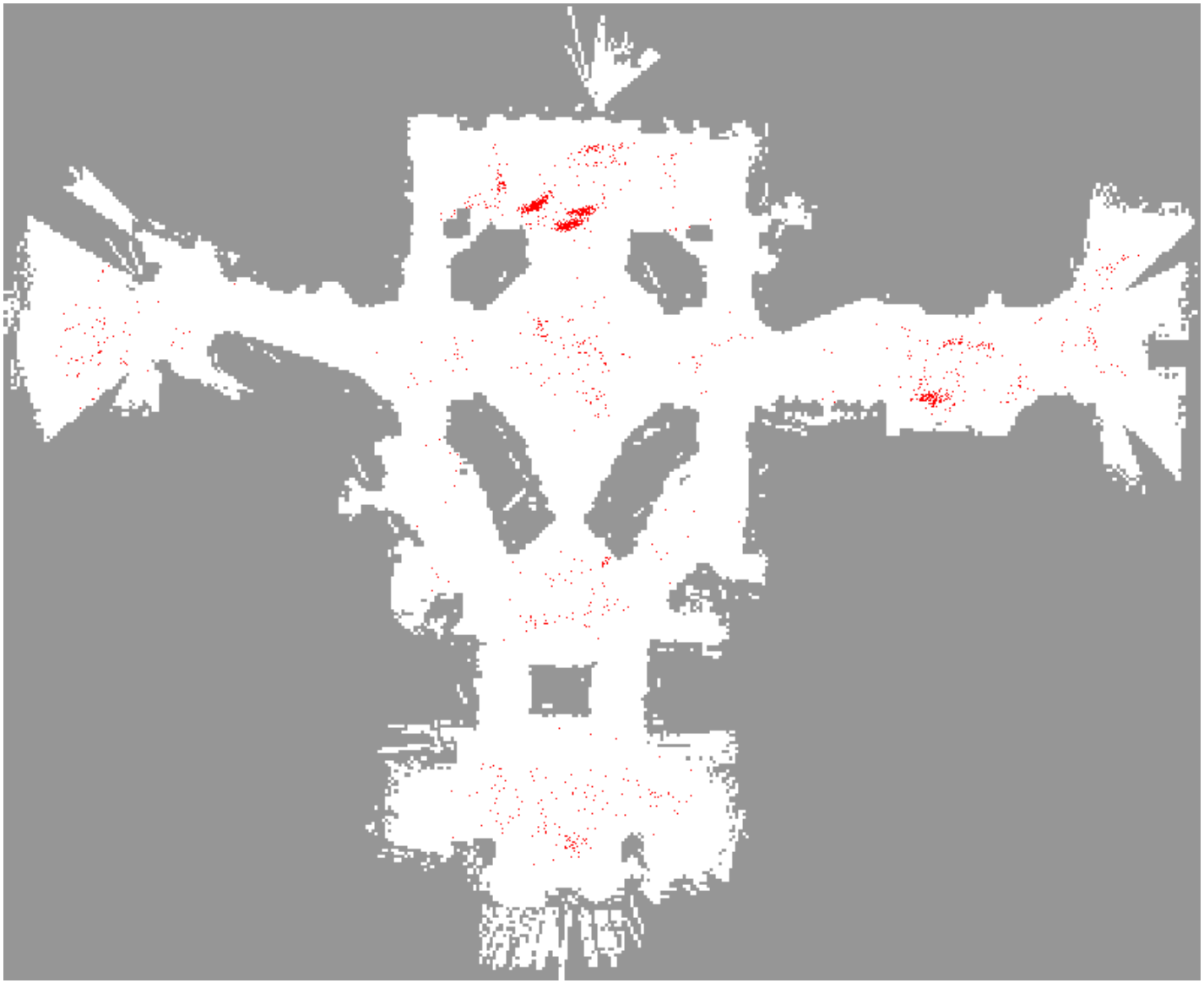
Beispiel (3)



Beispiel (4)

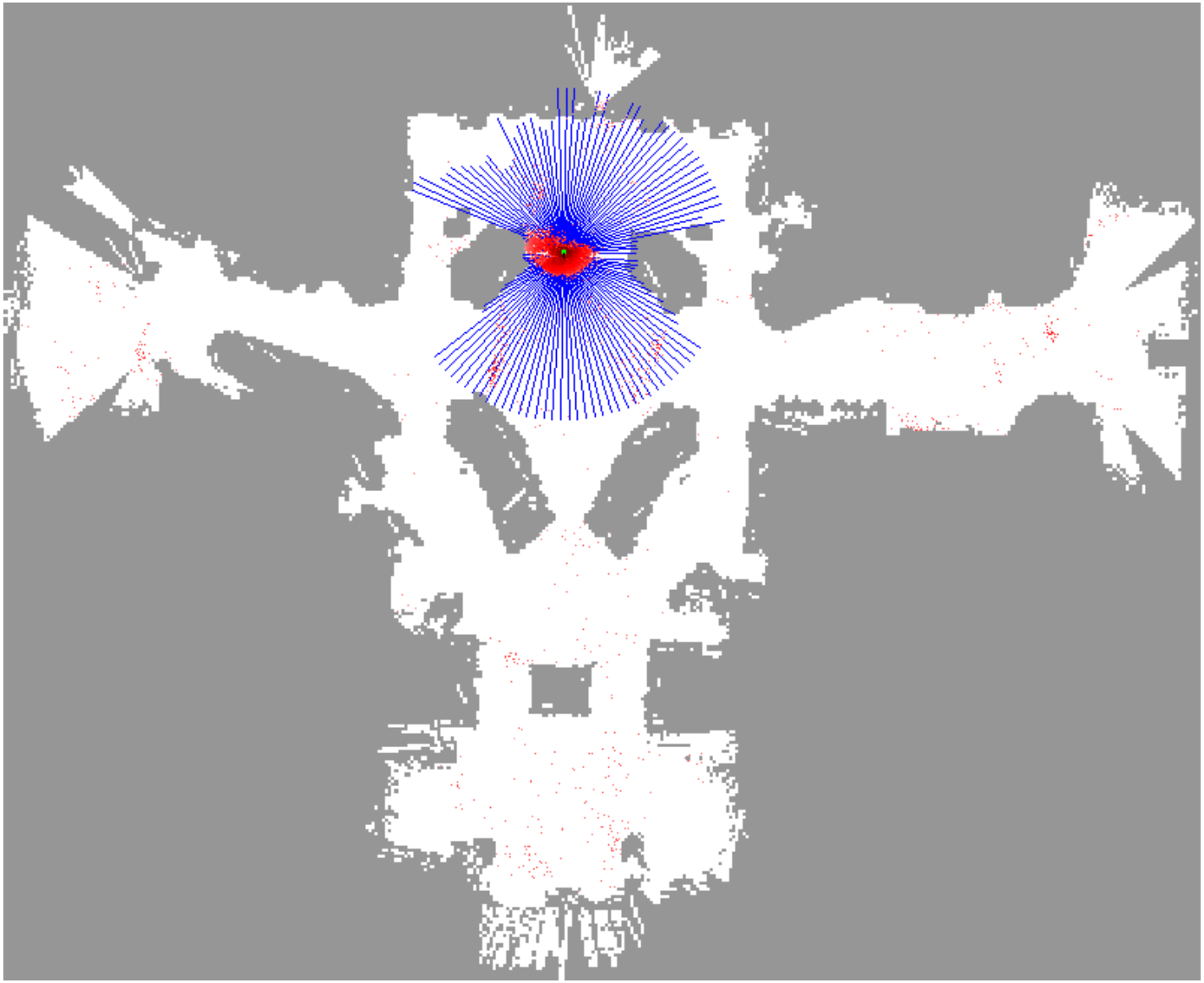


Beispiel (5)

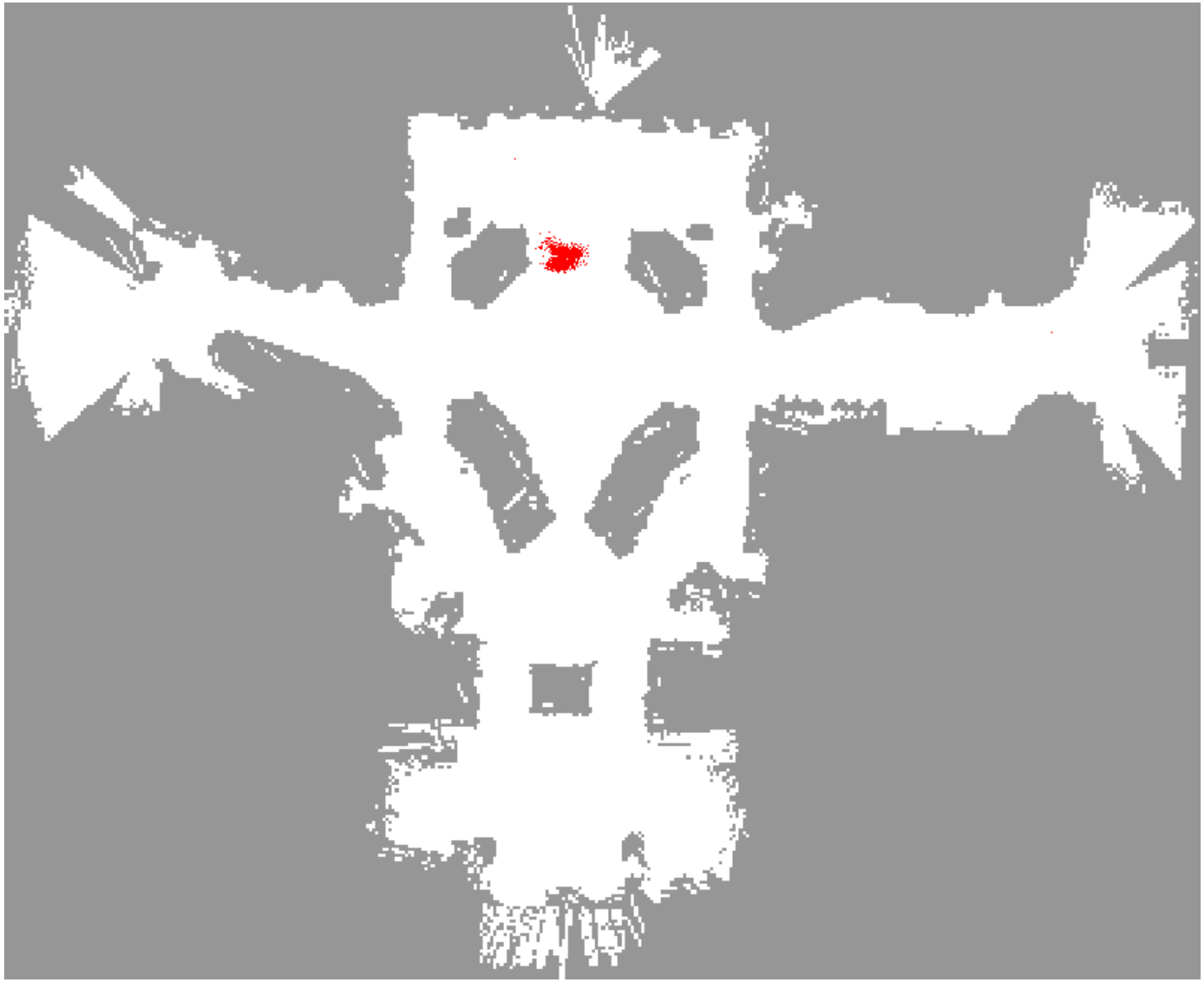


Beispiel (6)

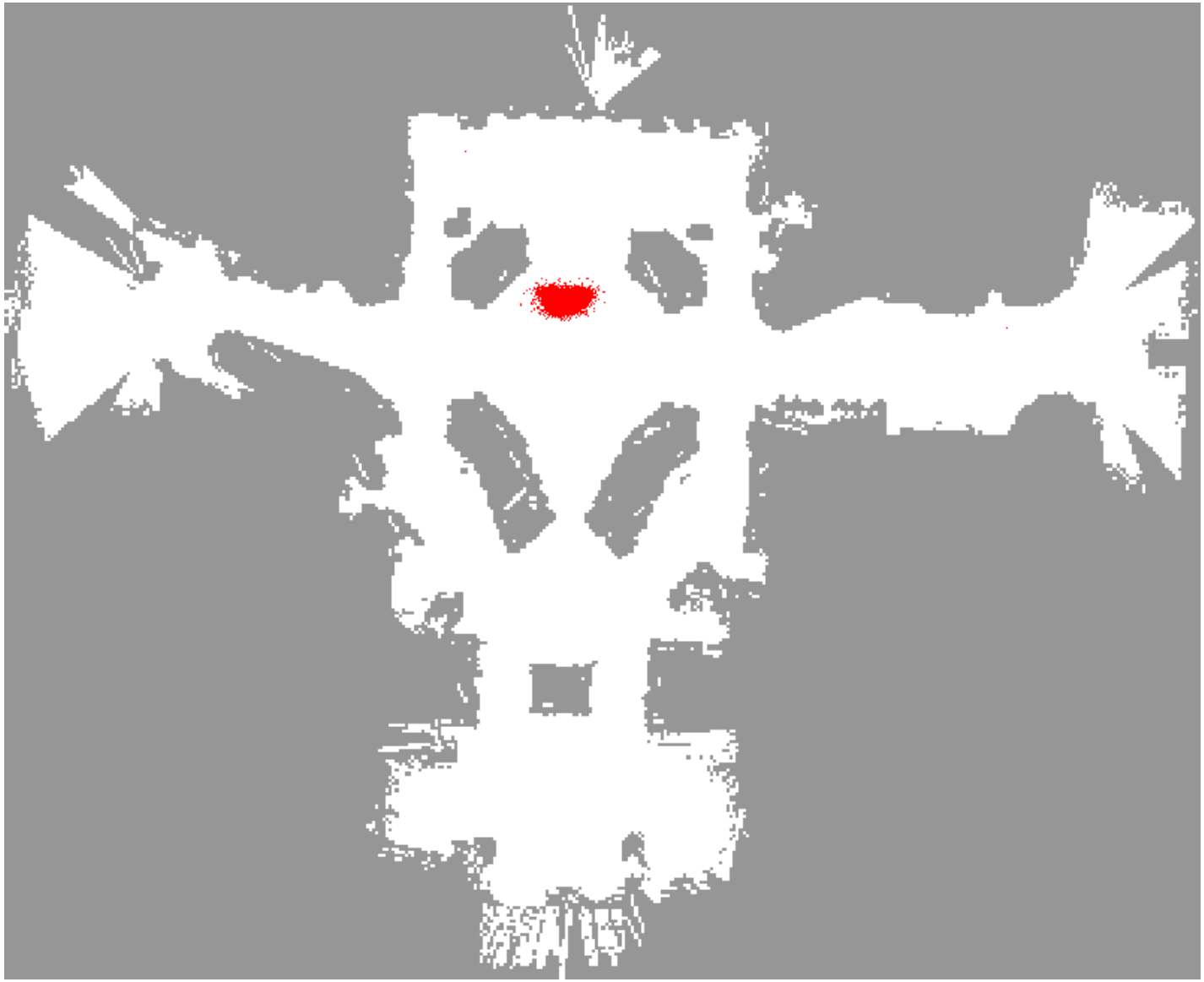
Beispiel (7)



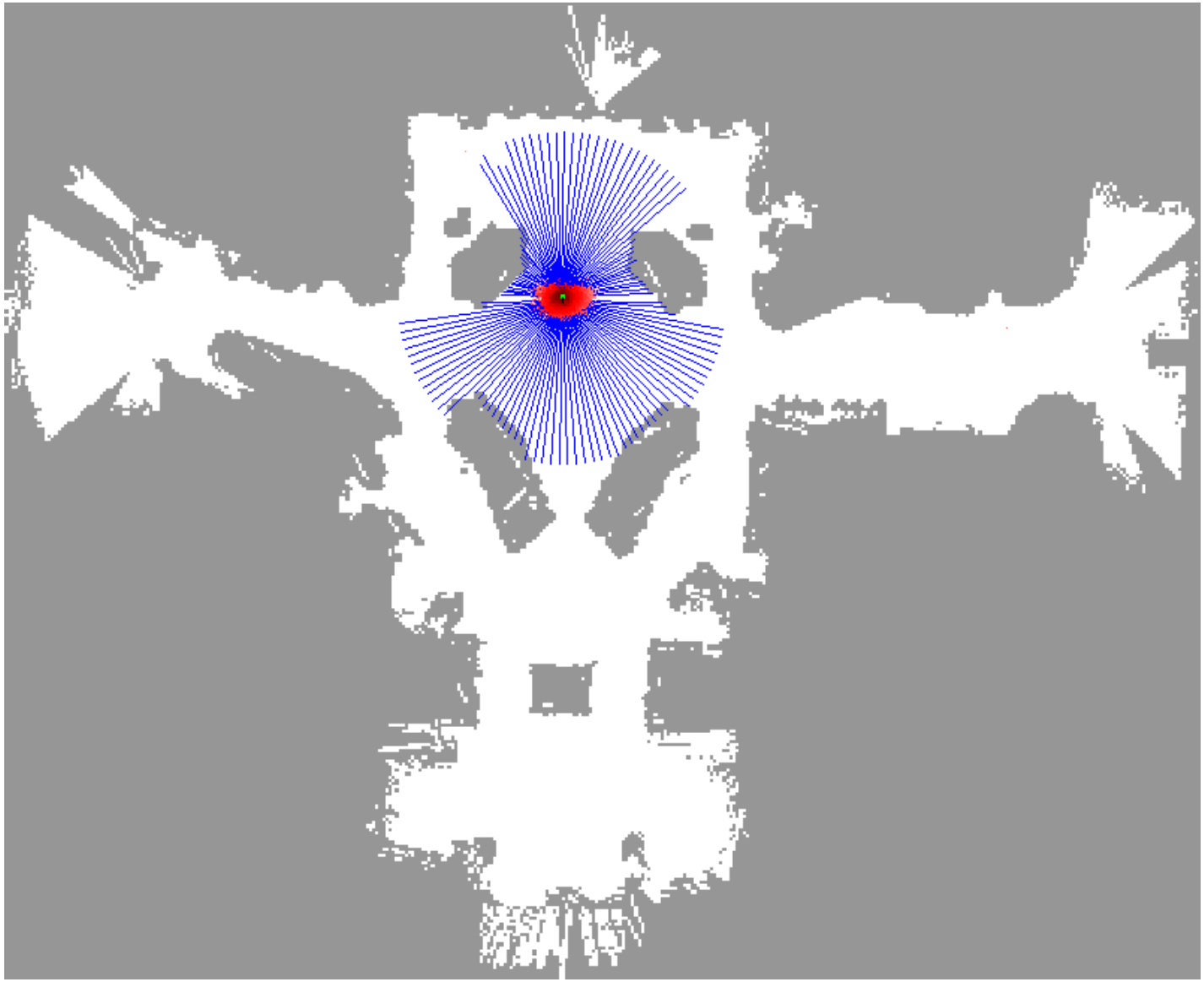
Beispiel (8)



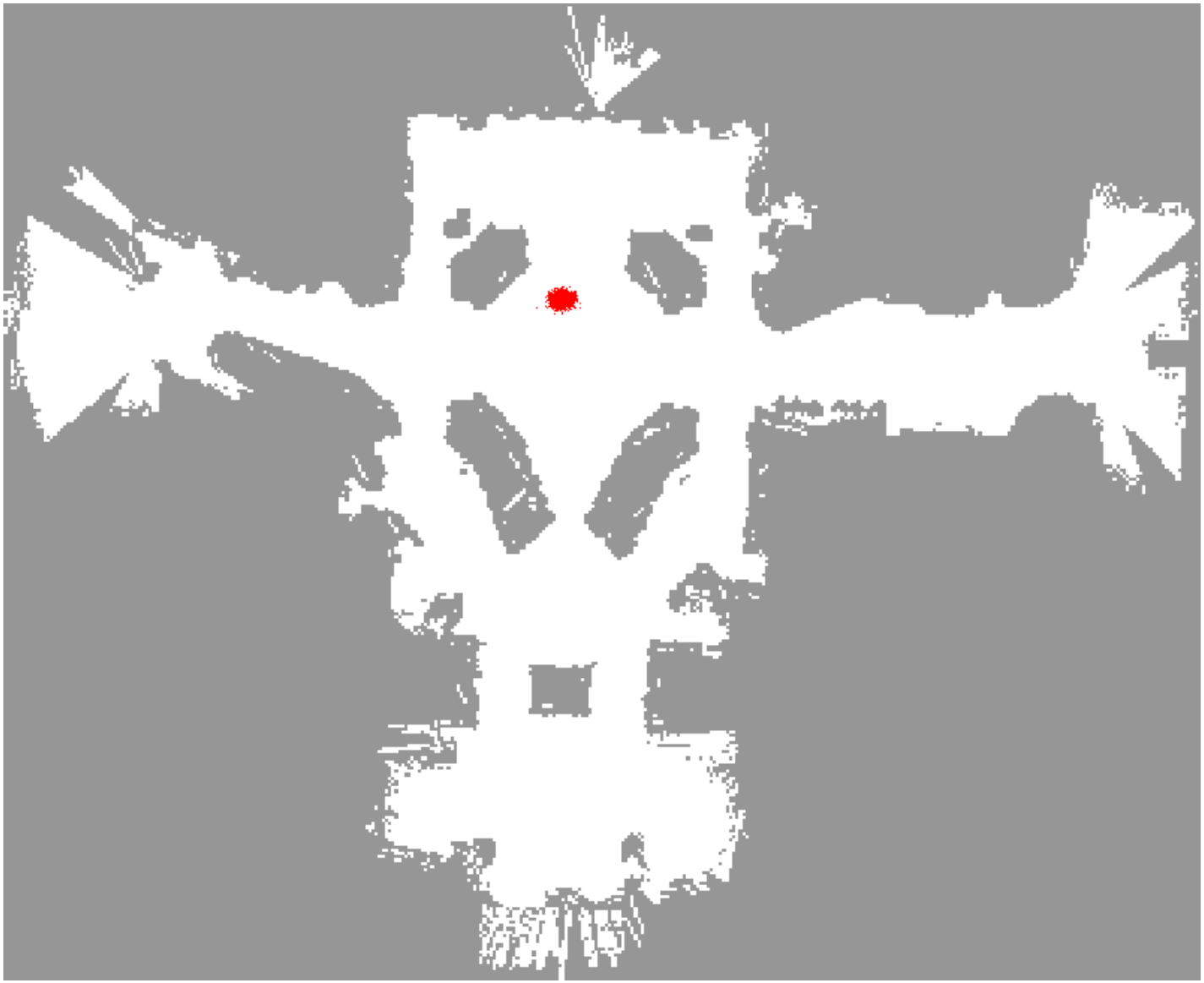
Beispiel (9)



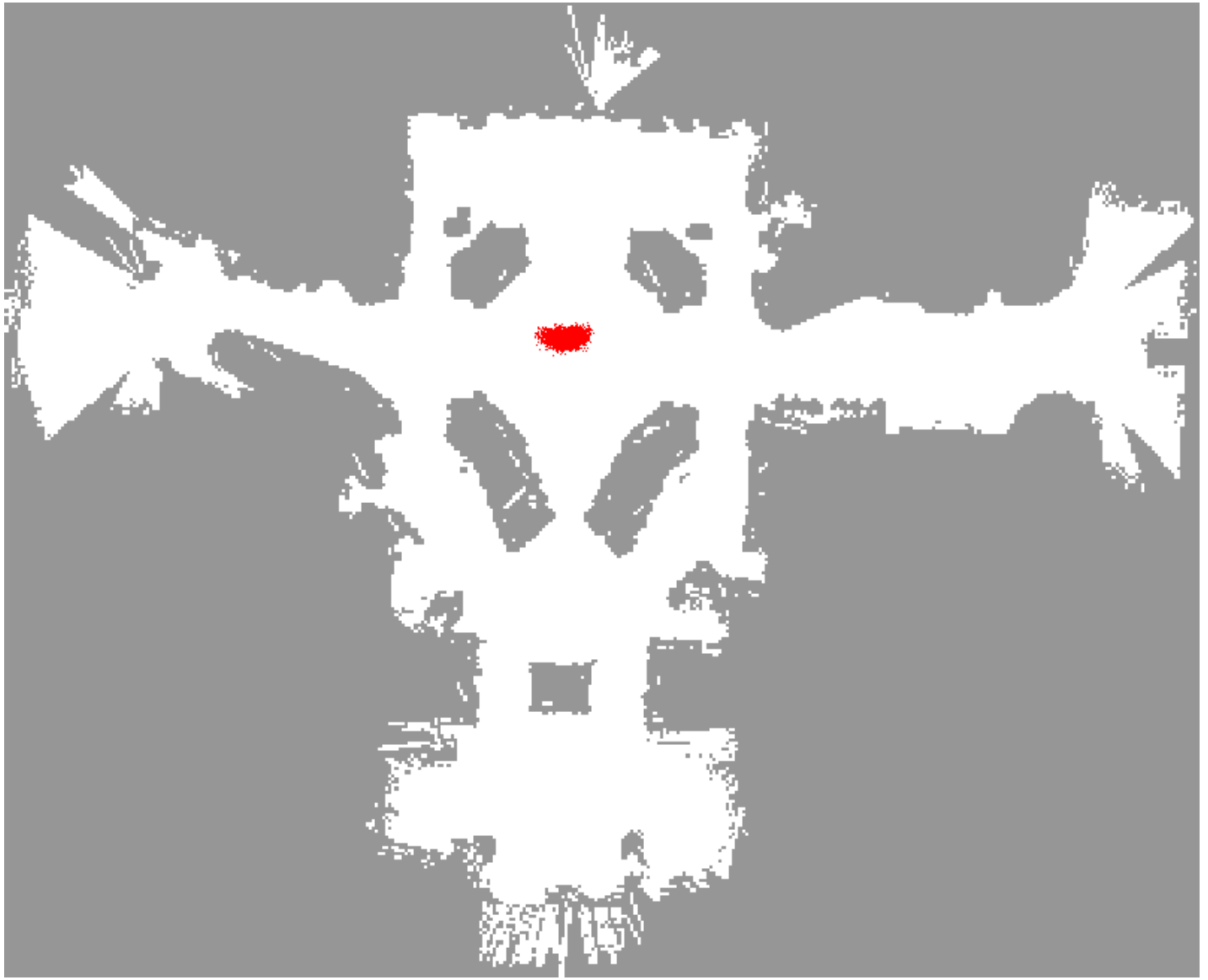
Beispiel (10)



Beispiel (11)



Beispiel (12)

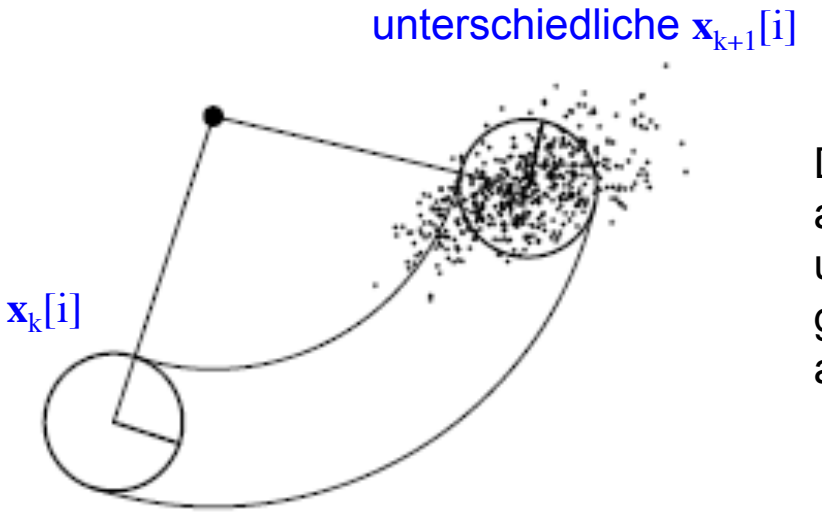


Bewegungsmodell

- Die Funktion sampleMotionModell addiert zunächst zum Steuerbefehl $\mathbf{u}_k$ einen zufällig generierten Rauschterm mit der Varianz $\Sigma_{\mathbf{u}}$
- Auf den verrauschten Steuerbefehl $\tilde{\mathbf{u}}_k$ und einem Partikel $\mathbf{x}_k[i]$ wird dann die Funktion g des Bewegungsmodells angewendet:

$$\mathbf{x}_{k+1}[i] = g(\mathbf{x}_k[i], \tilde{\mathbf{u}}_k)$$

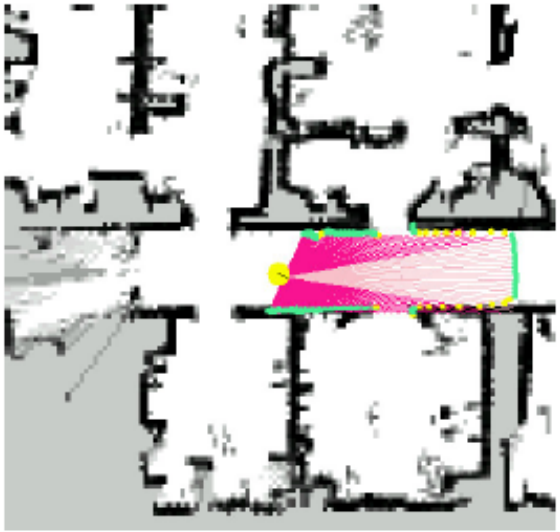
- Das Bewegunsgmodell g ist üblicherweise eines der Koppelnavigationsmodelle.



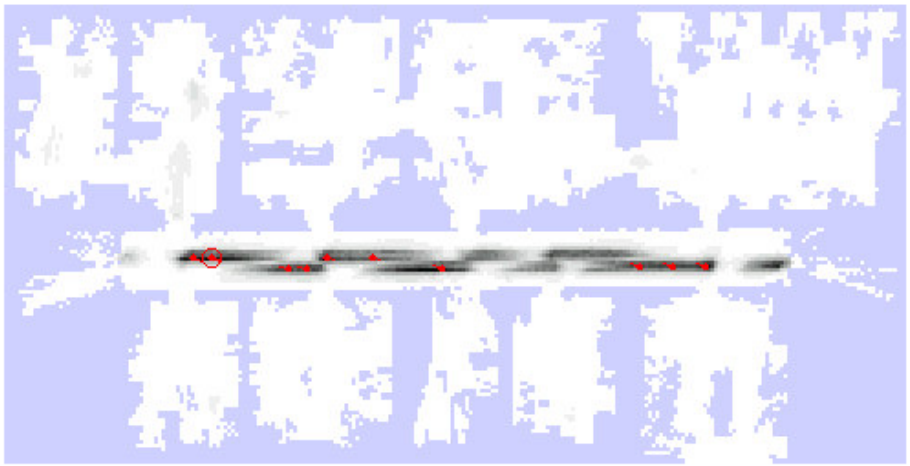
Das Bewegungsmodell g wurde auf ein Partikel $\mathbf{x}_k[i]$ und unterschiedlichen, zufällig generierten Steuerbefehlen $\tilde{\mathbf{u}}_k$ angewendet.

Messmodell (1)

- Es muss geprüft werden, wie gut die Sensorwerte z_{k+1} zur Partikel-Position $x_{k+1}[i]$ in einer Umgebungskarte m passen.



Sensorwerte z_{k+1}
(z.B Laser-Scan)



Wahrscheinlichkeit für jede Position $x_{k+1}[i]$, dass dort Laser-Scan z_{k+1} gemessen wurde. Je dunkler, desdo wahrscheinlicher.

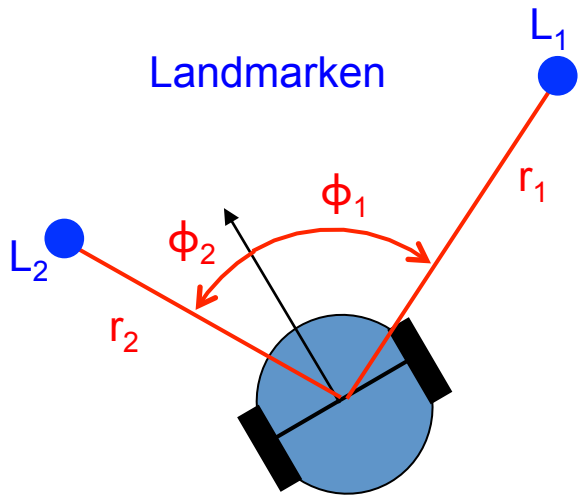
Messmodell (2)

- Es gibt in der Literatur zahlreiche Verfahren,
  wovon einige hier kurz behandelt werden.
- Wir wollen davon ausgehen, dass außer beim Landmarkenbasiertem
  Verfahren die Sensordaten aus einem Laser-Scan bestehen.
  Das sind typischerweise 181 Abstandswerte in 1 Grad Auflösung.

Karte	Verfahren
Menge von Landmarken	Landmarkenbasiertes Verfahren
Metrische Karte	Strahlenmodell
Metrische Karte	Likelihood-Field

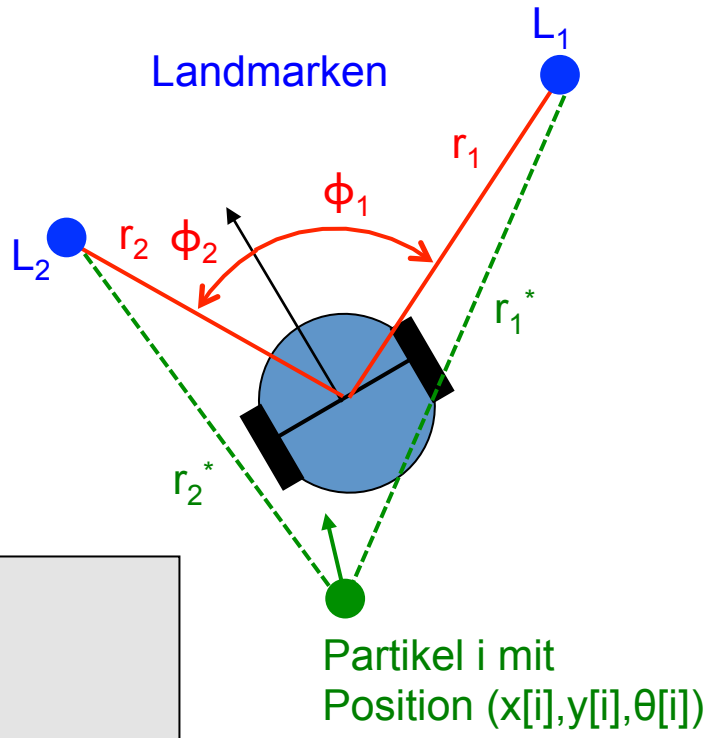
Landmarkenbasiertes Verfahren (1)

- Sensor misst zur j-ten Landmarke Entfernung r_j und relative Orientierung ϕ_j.
- Dabei wird ein normalverteilter Fehler angenommen:
  - $err_r \sim N(0,\sigma_r^2)$
  - $err_\phi \sim N(0,\sigma_\phi^2)$
- Es kann auch nur Entfernung oder Orientierung gemessen werden.



Landmarkenbasiertes Verfahren (2)

- Um das Gewicht w_i des Partikels i zu bestimmen, wird ermittelt wie wahrscheinlich die Sensorwerte an der Partikelposition (x[i],y[i],θ[i]) sind:

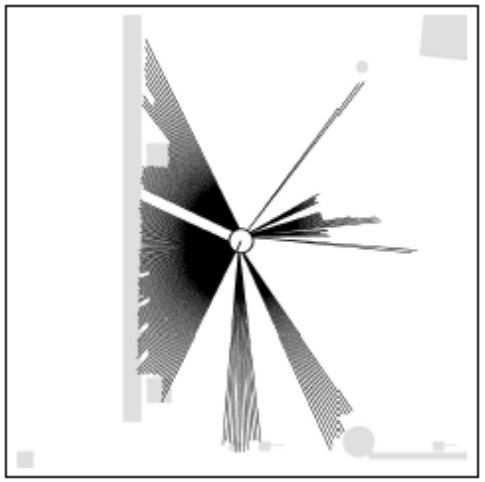



```
p = 1;
for all Sensorwerte z_j = (r_j, ϕ_j) do
      rechne für Partikelposition (x[i],y[i],θ[i])
      erwartete Entfernung r_j* und relative
      Orientierung ϕ_j* zur j-ten Landmarke aus;
      p = p * N(0,σ_r^2)(r_j – r_j*) * N(0,σ_ϕ^2)(ϕ_j - ϕ_j*) ;
endfor
```

Strahlenmodell

- Laser misst strahlenförmig Abstände z = d_1,d_2, …, d_n zu den nächsten Hindernissen.
- Es wird die Wahrscheinlichkeit p geschätzt, dass an der Partikelposition (x[i],y[i],θ[i]) diese Abstandsmessungen beobachtet werden.

```
p = 1;
for k = 1 to n do
    berechene in Richtung der Messung d_k durch
    Strahlverfolgung von Position (x[i],y[i],θ[i]) aus
    in der Karte m den Abstand d_k*
    zum ersten Hindernispunkt;
    p = p * N(0,σ²)(d_k - d_k*);
return p;
```



Beobachtete Abstands-messungen $z_t = z_1,z_2, \ldots, z_n$

Likelihoodfield (1)

- Für eine Lasermessung z = d_1,d_2, …, d_n und eine Partikelposition (x[i],y[i],θ[i]) werden Hindernispunkte (x_{z1}, y_{z1}), …, (x_{zn}, y_{zn}) im globalen Koordinatensystem berechnet.
- Es wird die Wahrscheinlichkeit p geschätzt, dass an der Partikelposition (x[i],y[i],θ[i]) die Lasermessung beobachtet wird, indem berechnet wird, wie nah die Hindernispunkte von tatsächlichen Hindernissen in der Umgebungskarte entfernt sind.

```
p = 1;
for k = 1 to n do
    berechne aus lokalen Polarkoordinaten (d_k,φ_k) globale Koordinaten (x_zk, y_zk);
    dist = Abstand von (x_zk, y_zk) zum nächstgelegenen Hindernispunkes in m;
    p = p * N(0,σ²)(dist);
end
```

Likelihoodfield (2)

- Der Abstand dist von (x_{zk}, y_{zk}) zum nächstgelegenen Hindernispunkt läßt sich vorab berechnen.
- Damit wird der Algorithmus (auf Kosten von Speicherplatz!) wesentlich beschleunigt.

Links:
Umgebungskarte m.

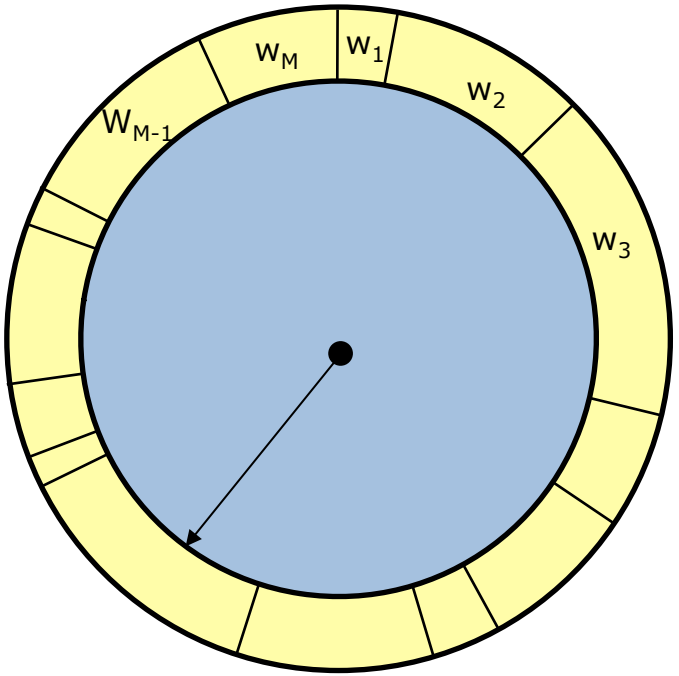
Rechts:
im Voraus berechnetes Likelihood-Field.
Je dunkler desdo unwahrscheinlicher ist es, dort ein Hindernispunkt zu beobachten.

Resampling mit Roulette-Rad-Verfahren

- Aus der Partikelmenge {x[1], x[2], .., x[M]} mit Gewichten w_1,..., w_M sollen M Partikeln zufällig entsprechend ihrem Gewicht gezogen werden.
- Bilde dazu M Intervalle (gelbe Streifen im Roulette-Rad):

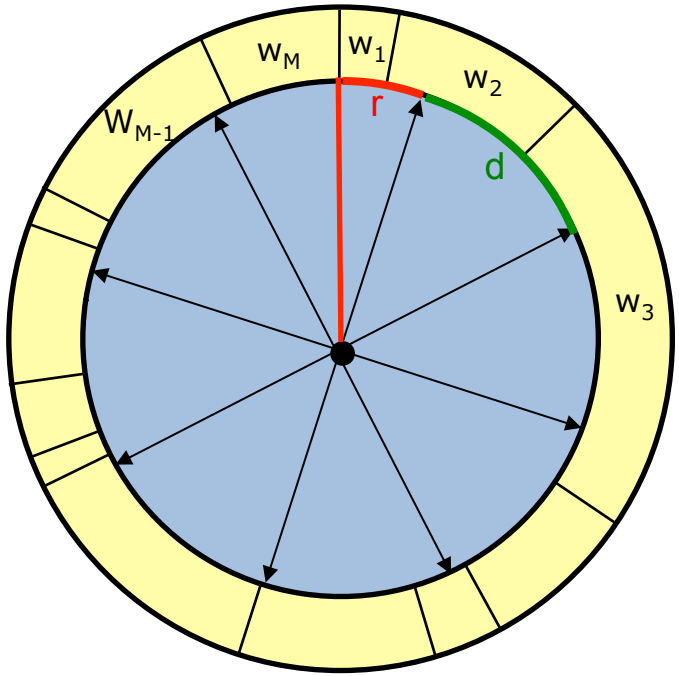
  $I_1 = [0, w_1]$
  $I_2 = [w_1, w_1+w_2]$
  ...
  $I_M = [w_1+...+w_{M-1}, w_1+...+w_M]$

- Führe folgende Schritte M-mal aus:
  - Generiere gleichverteilte Zufallszahl z aus $[0, w_1+...+w_M]$ (drehe Roulette-Rad);
  - Ermittle Intervall $I_{n,}$ in dem z liegt mit binärer Suche;
  - Wähle Partikel x[n];
- Beachte: ein Partikel kann mehrfach ausgewählt werden.
- Laufzeit: O(M log(M)).



Resampling mit Speichenrad-Verfahren

- Aus der Partikelmenge {x[1], x[2], .., x[M]} mit Gewichten w_1,..., w_M sollen M Partikeln zufällig entsprechend ihrem Gewicht gezogen werden.
- Ordne Gewichte w_1,..., w_M auf einer kreisförmigen Skala an.
  Es sei $W = w_1+...+w_M$ die Gewichtsumme.
- Bilde ein Speichenrad mit M Speichen und einem Speichenabstand von $d = W/M$.
- Drehe 1. Speiche auf einen zufälligen Wert $r \in [0, d]$.
- Wähle jeden Partikel x[i] aus, auf dessen Gewicht w[i] eine Speiche zeigt.
  Zeigen mehrere Speichen auf ein Partikel, dann wird der Partikel mehrfach ausgewählt.
- Laufzeit: O(M).



Diversität der Partikelmenge

- Wünschenswert ist eine gewisse Streuung der Partikelmenge (Diversität).
  Im Idealfall ist die Partikelmenge normalverteilt um die tatsächliche Pose.
- Falls sich der Roboter im Extremfall über viele Schritte nicht bewegt
  (d.h. keine Integration eines Steuerbefehls) und darüberhinaus alle Gewichte immer gleich sind, dann tendiert das Roulette-Verfahren dazu, mit M Kopien des gleichen Partikels zu enden.

  Die Posenvarianz der Partikel wird 0. D.h. die Diversität verschwindet komplett.
  Die Wahrscheinlichkeit eines Lokalisierungsfehlers ist jedoch sehr groß.
- Das Speichenrad-Verfahren behält dagegen eher die Diversität bei.
  Haben alle Partikel das gleiche Gewicht, dann bleibt die Partikelmenge unverändert.
- Die Diversität lässt sich auch durch eine Verringerung der Resampling-Frequenz erhalten.

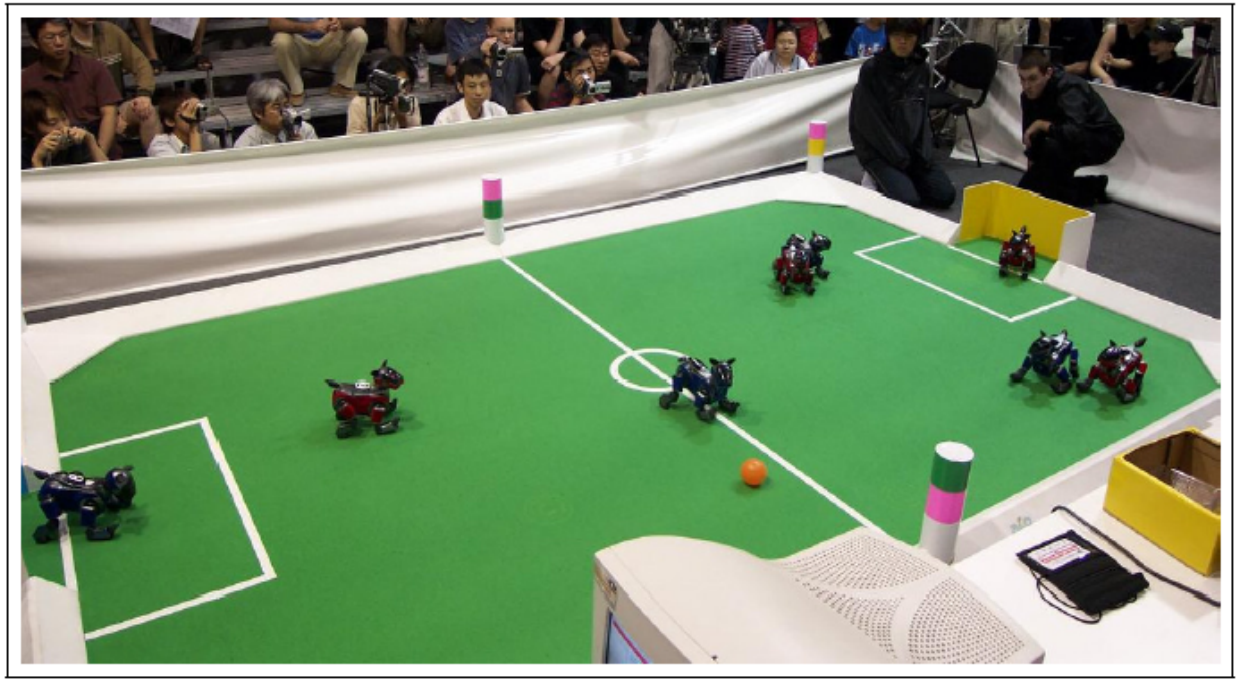
  Es wird nur in jedem N-ten Schritt ein Resampling durchgeführt.
  In den Zwischenschritten werden die Partikeln zwar gewichtet, jedoch werden die Gewichte je Partikel nur aufmultipliziert. Das Resampling wird über das Gewichtsprodukt durchgeführt. Die Steuerbefehle werden dagegen in jedem Schritt integriert.

  Ein guter Indikator, um festzustellen, ob ein Resampling durchgeführt werden soll, ist die Gewichtsvarianz. Nur bei hoher Gewichtsvarianz sollte ein Resampling durchgeführt werden.

Monte-Carlo-Lokalisierung

- Einleitung
- Modellierung von Unsicherheit mit Partikelmengen
- Algorithmus
- Beispiele
- Varianten

Beispiel – Landmarkenbasierte Navigation (1)



- Aibo und Robo-Soccer
- 6 Landmarken am Spielfeldrand; farblich gekennzeichnet

- Bewegungsmodell auf 4-Bein-Antrieb angepasst.
- Sensormodell auf Landmarkenerkennung angepasst. Sensorwert z_t besteht aus Richtung, Abstand und Landmarkennummer

Beispiel – Landmarkenbasierte Navigation (2)

(c) Partikelwolken
zu unterschiedlichen Zeitpunkten.
Jeweils vor und nach Resampling.

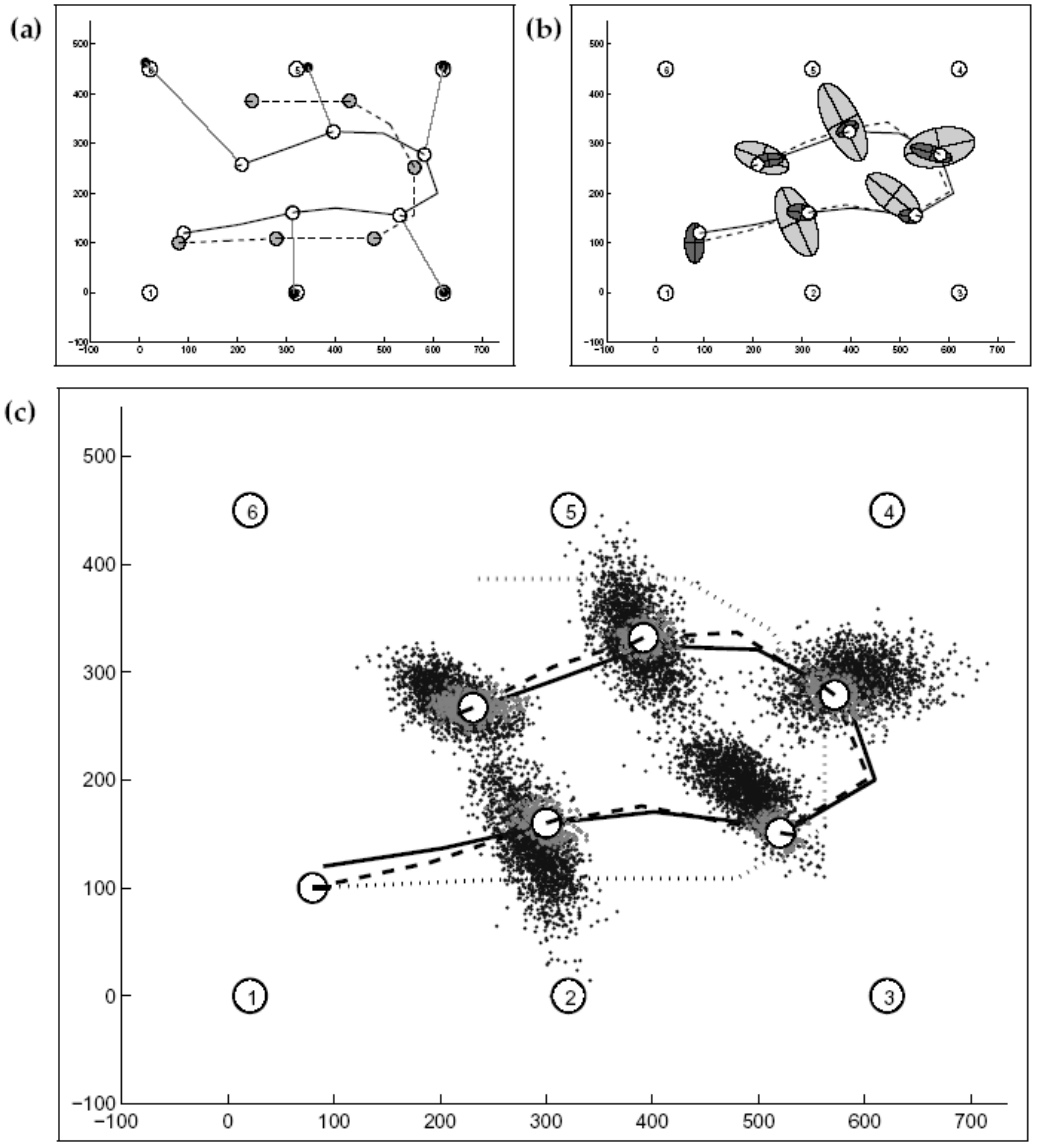
Durchgezogene Line:
tatsächlicher Weg

Gepunktete Linie:
über Odometrie gemessener Weg

Gestrichelte Linie:
über MCL ermittelter Weg

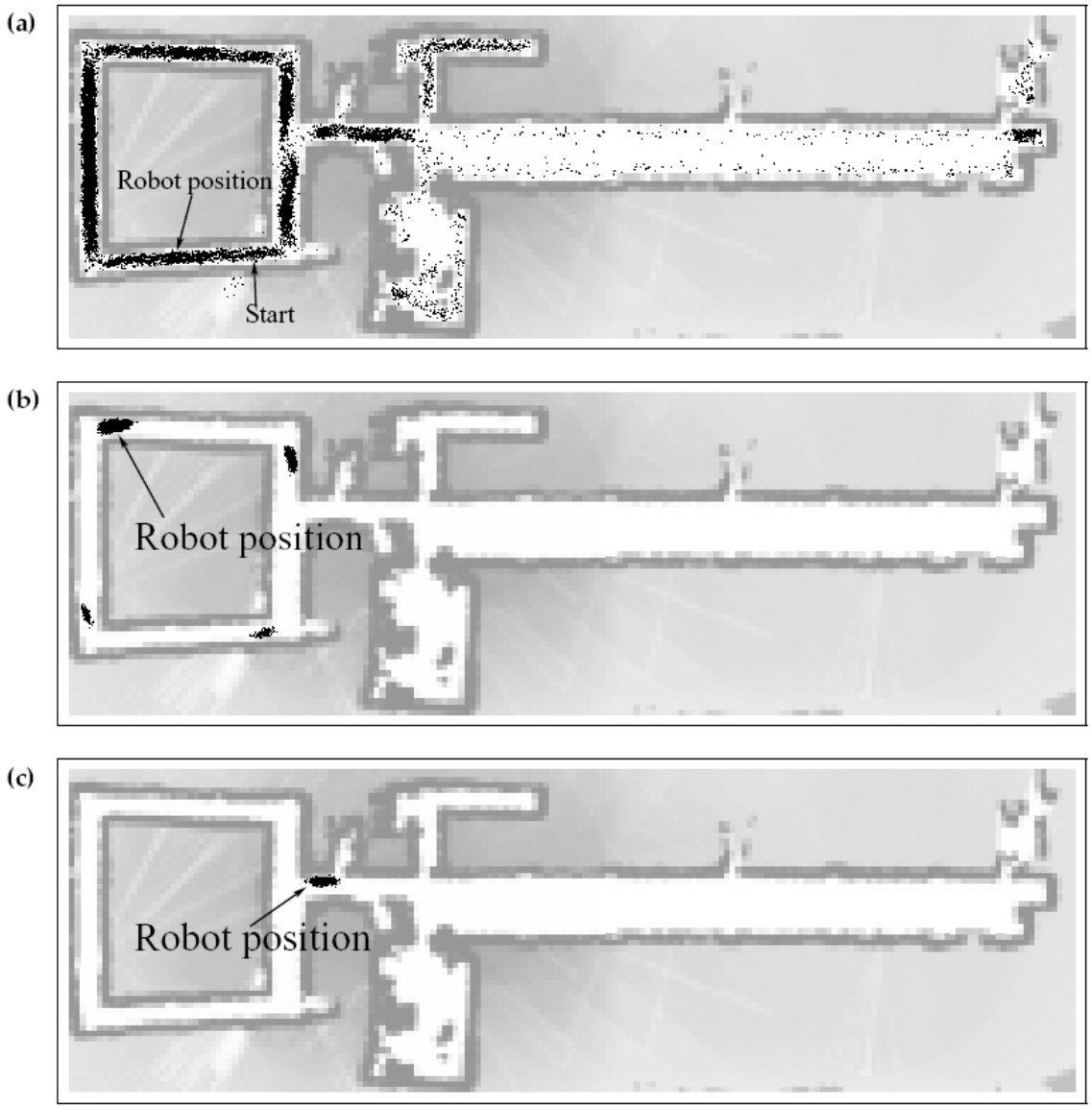
(a) gibt an welche Landmarke
zu den einzelnen
Zeitpunkten erkannt wird.

(b) Kovarianzen zu den
Partikelwolken aus (a)

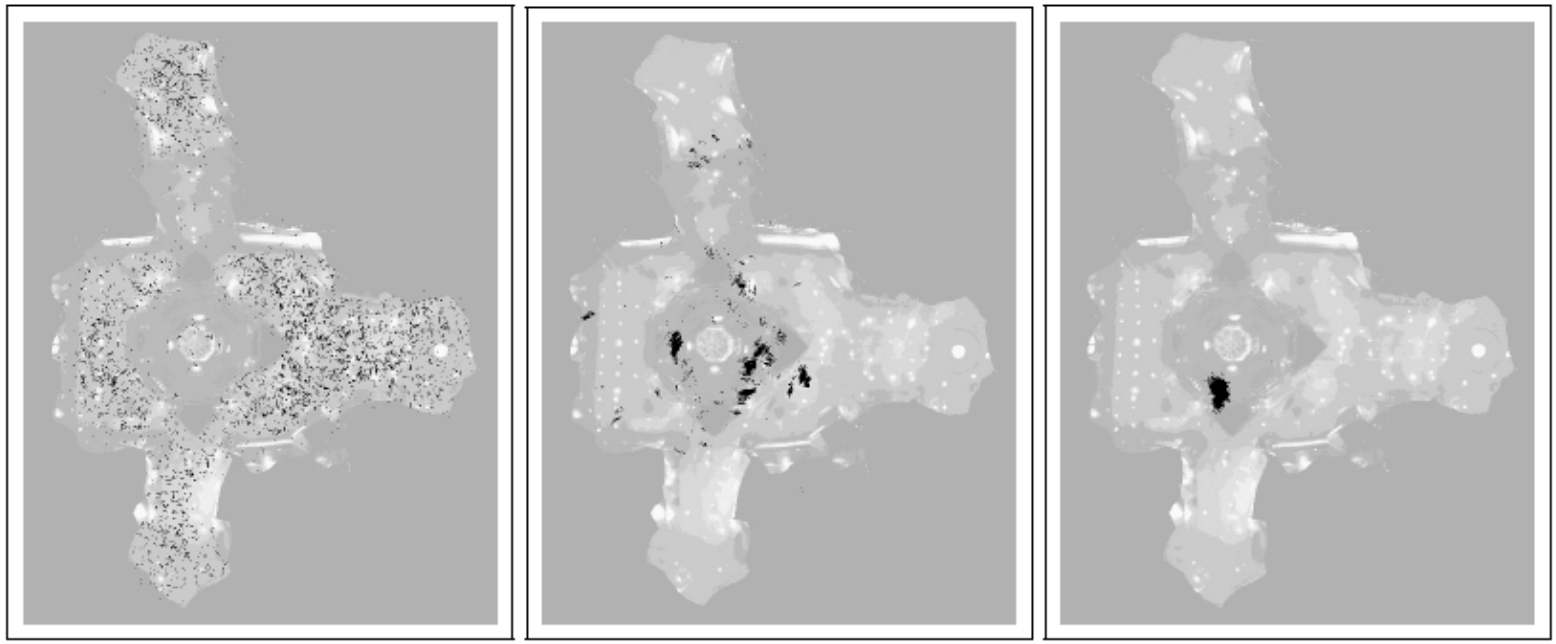


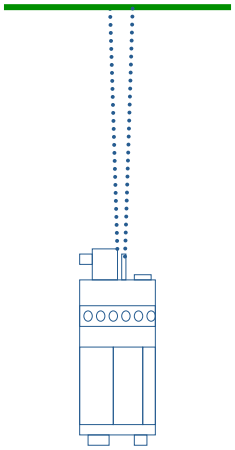
Beispiel – Gobale Selbstlokalisierung mit Ultraschall

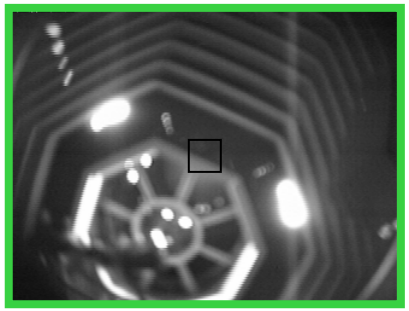
- Büroumgebung mit 54m x 18m.
- Nach 5m Fahrt ist die Roboterposition noch sehr ungewiss (a).
- Nachdem der Roboter im linken oberen Eck angelangt ist, gibt es noch 4 mögliche Positionen (b).
- Nach 55m Fahrt ist die Position gewiss (c).



Beispiel – Gobale Selbstlokalisierung mit Kamera in Himmelsrichtung

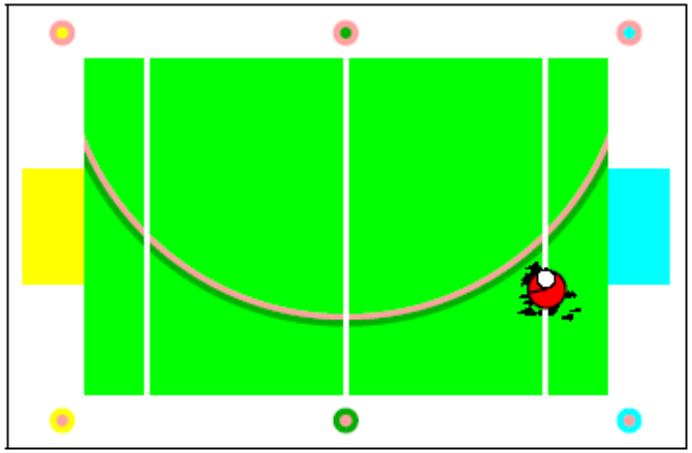


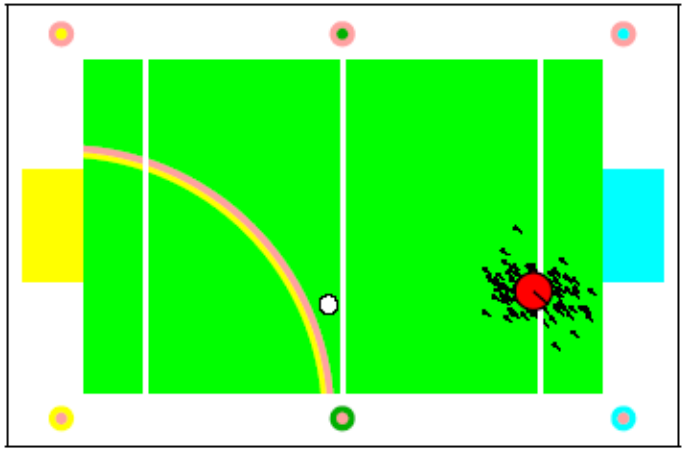




Varianten: MCL mit Zufallspartikeln (1)

- Kidnapping-Problematik kann nicht gelöst werden, sobald Partikeln sich an einer Stelle konzentrieren.
- Abhilfe: erzeuge zufällige Partikel.
- Die Anzahl kann von der aktuellen Lokalisierungsgüte (Durchschnitt der Gewichte) abhängig gemacht werden.
- Die Partikelpositionen können durch die Sensorwerte beeinflusst werden

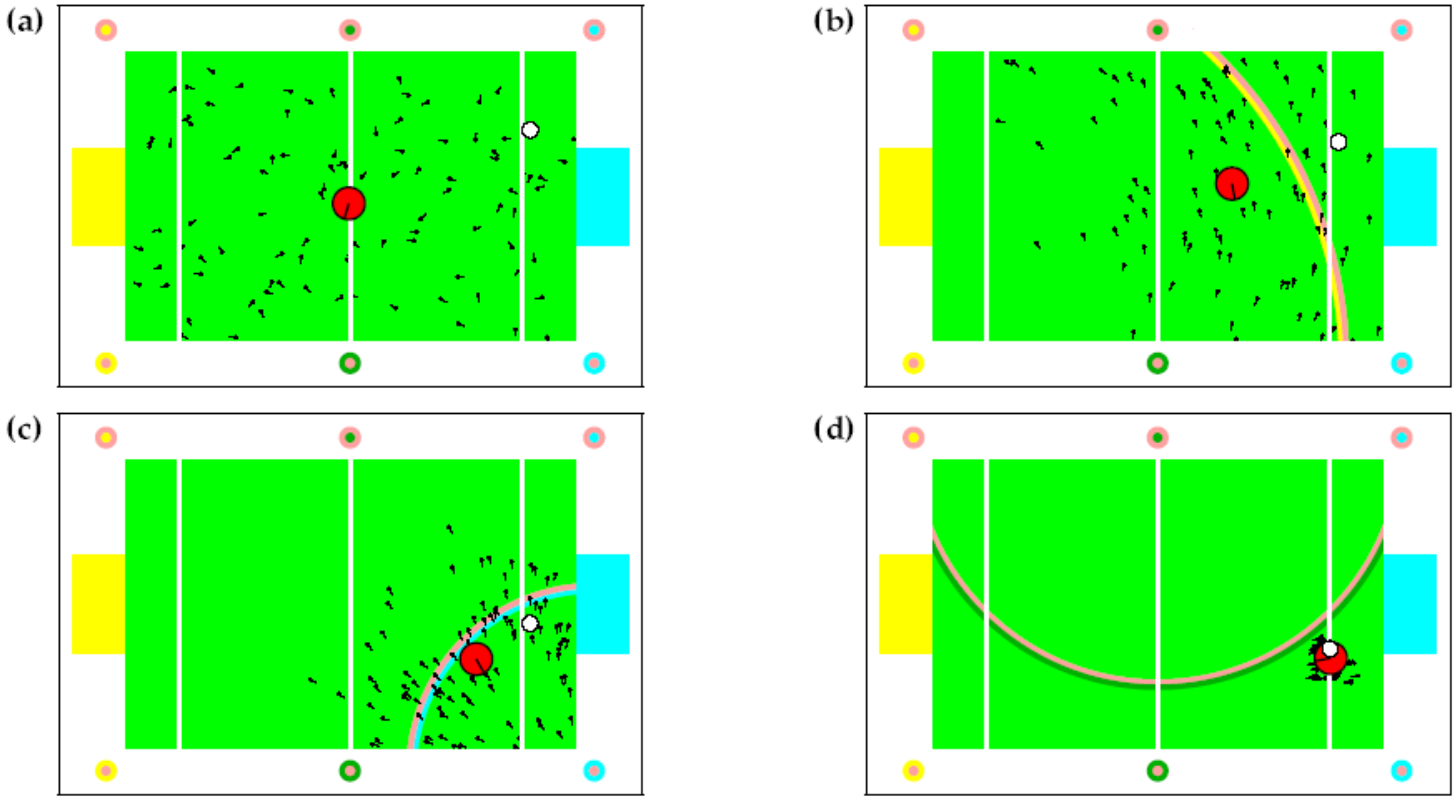




Roboter wurde gekidnapped.
Weißer Punkt gibt die neue Position an.

Varianten: MCL mit Zufallspartikeln (2)

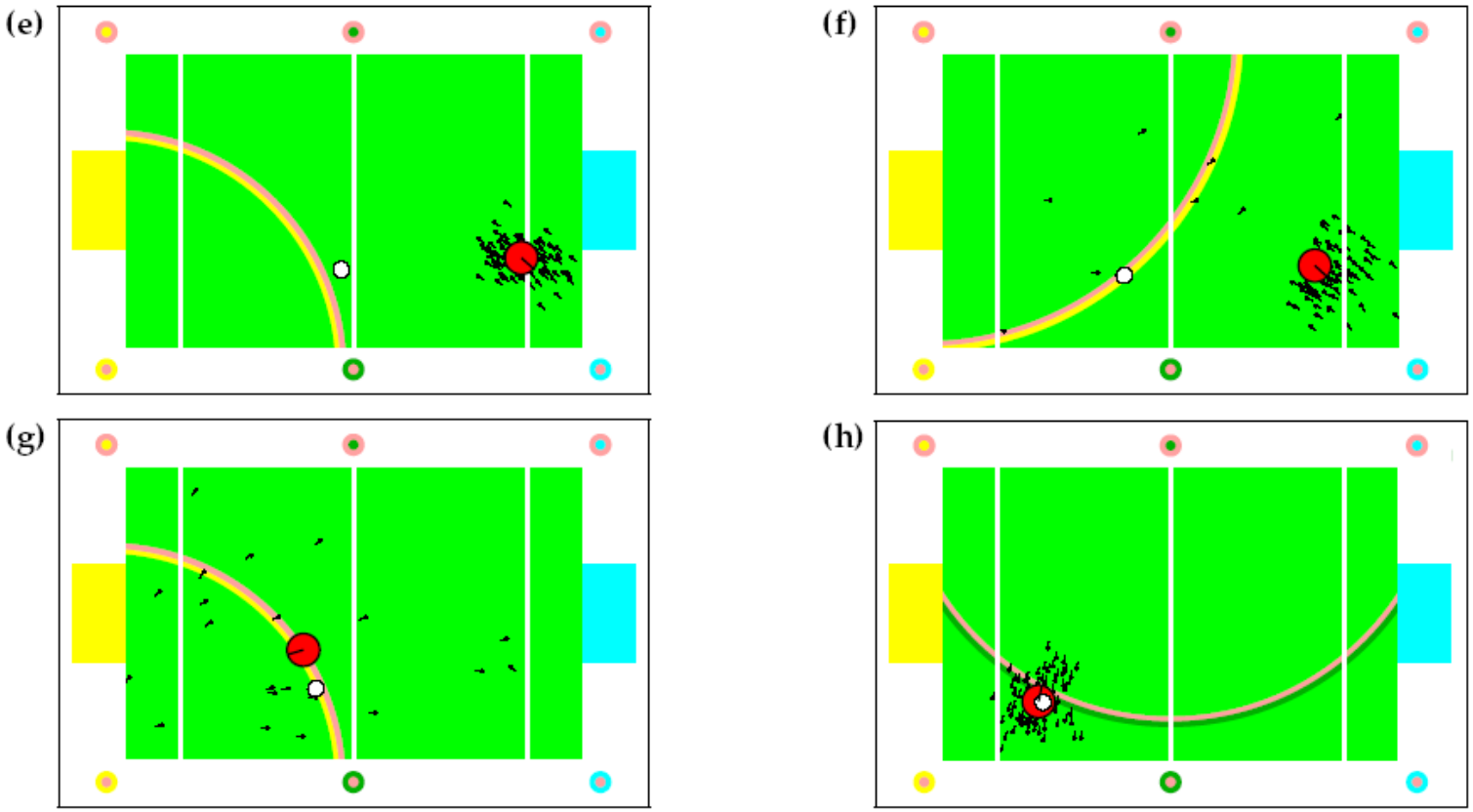
Beispiel: Aibo und RoboSoccer



- (a) bis (d) globale Selbstlokalisierung
- Schwarze Punkte: Partikel haben ein kleine Linie, die die Richtung kennzeichnet.
- Roter Punkte: Mittelwert der Partikel.
- Weißer Punkt: tatsächliche Position
- Sensormesswerte sind durch Kreisbögen mit Landmarke als Mittelpunkt dargestellt.

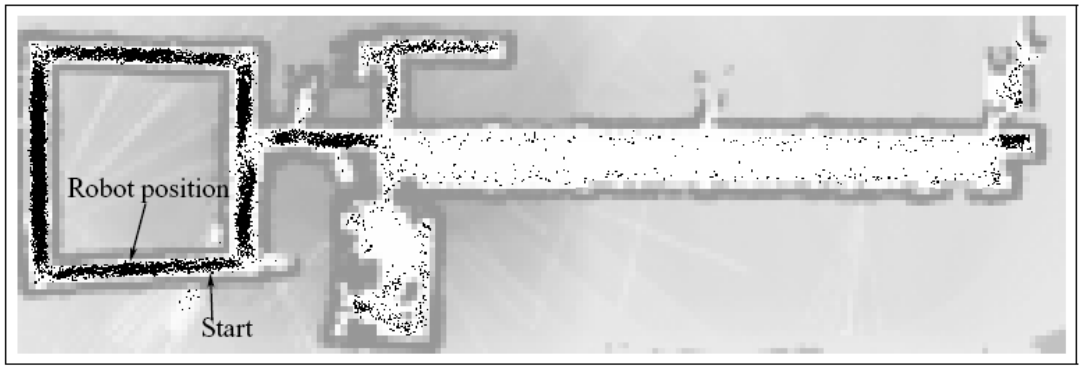
Varianten: MCL mit Zufallspartikeln (3)

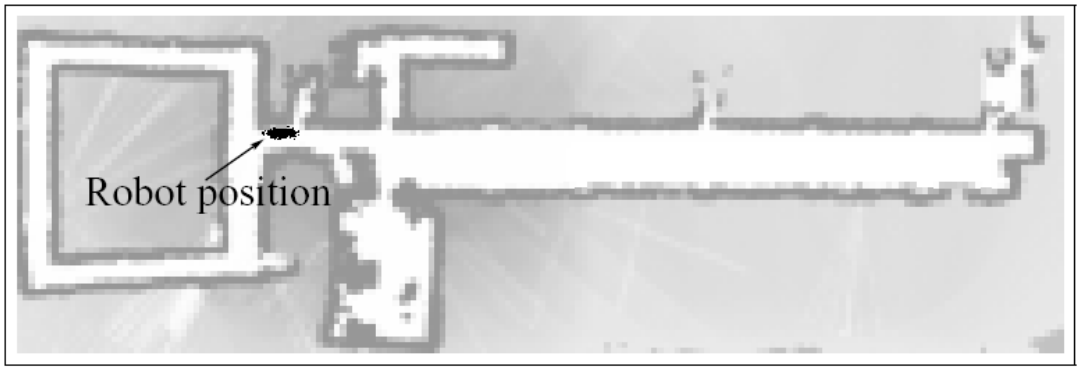
Aibo und RoboSoccer – Relokalisierung nach Kidnapping



KLD-Sampling: Anpassung der Partikelzahl

- Falls die Gewissheit der Position sehr klein ist (globale Selbstlokalisierung), dann sind sehr viele Partikel notwendig (z.B. 100.000).
- Falls die Gewissheit der Position größer wird (d.h. Partikel sind auf wenige Stellen konzentriert), dann genügen wesentlich weniger Partikel (z.B. einige hundert).
- Adaptives Verfahren mit KLD-Samplig: mit einer Histogrammtechnik wird die Verteilung der Partikeln gemessen und damit die Anzahl der neuen Partikeln gesteuert.

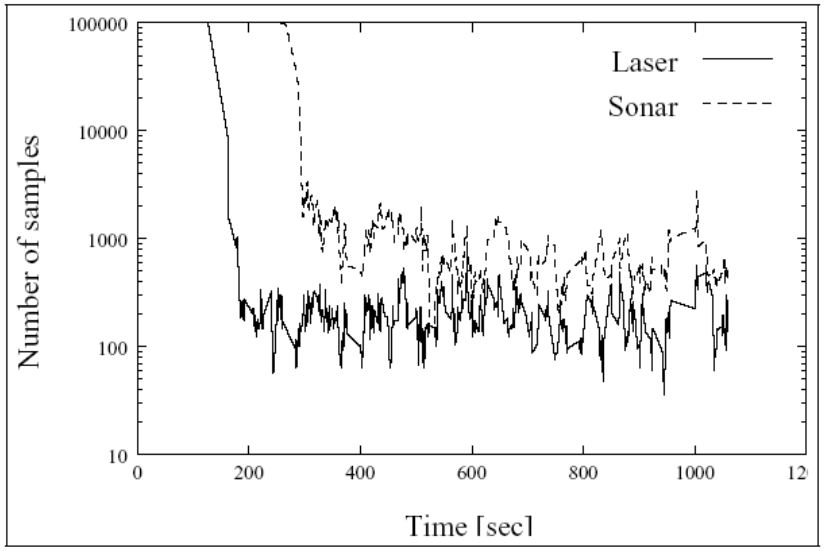




Globale Selbstlokalisierung mit 100.000 Partikeln.

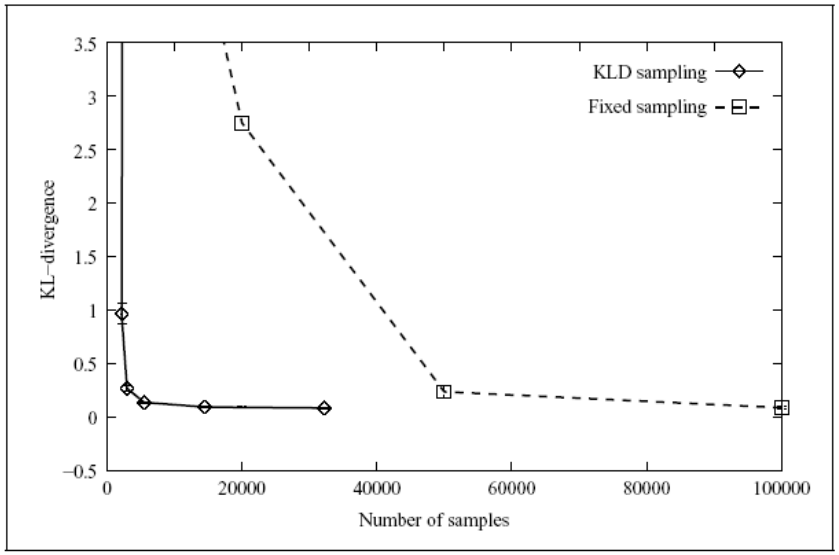
KLD-Sampling: Leistungsfähigkeit

Partikelanzahl und KLD-Sampling



Approximationsgüte von MCL mit konstanter Partikelanzahl und KLD-Sampling

Approximationsgüte (y-Achse) wird berechnet durch Divergenz zur optimalen Positionsschätzung



KLD-Sampling Algorithmus (1)

- mit einer Histogrammtechnik wird die Verteilung der Partikeln gemessen und damit die Anzahl der neuen Partikeln gesteuert.

```
Algorithm KLD_Sampling_MCL(χ_k, u_k, z_{k+1})
    χ_{k+1}= ∅; M = 0; M_χ = 0;
    k = 0;
    for all b in H do
        b = empty
    endfor
    do
        ziehe Partikel i mit Wahrscheinlichkeit w_k[i]
        x_{k+1}[i] = sampleMotionModel(u_k,x_k[i]);
        w_{k+1}[i] = measurementModel(z_{k+1},x_{k+1}[i],m);
        χ_{k+1} = χ_{k+1} ∪ {(x_{k+1}[i] , w_{k+1}[i]};
        M = M + 1;
        aktualisiere Histogramm H
        und passe M_χ an;
    while M < M_χ or M < M_χmin
    return χ_{k+1};
```

- χ_{k+1} ist die neue Partikelmenge und M die Partikelanzahl.
- M_χ ist die Partikelanzahl, die erreicht werden soll.
- Partikel werden mit Gewichten abgespeichert!

- H ist das Histogramm und besteht aus einer Menge von Behältern (bins) über den Posen-Raum.
- k ist die Anzahl der nicht-leeren Behälter

- siehe nächste Folie

- $M_{\chi min}$ ist die gewünschte Untergrenze der Partikelanzahl

KLD-Sampling Algorithmus (2)

aktualisiere Histogramm H und passe M_χ an:

if $x_{k+1}[i]$ fällt in einen leeren Behälter b then
    k = k + 1;
    b = non-empty;
    if k > 1 then

$$M_X = \frac{k-1}{2\varepsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\}^3$$

endif

- KLD-Sampling bestimmt die geforderte Anzahl der Partikel M_χ so, dass mit einer Wahrscheinlichkeit von 1-δ die Partikelmenge die tatsächliche Verteilung mit einem Fehler von ε approximiert.
- In der Praxis: ε = 0.05 und 1-δ = 0.99
- $z_{1-\delta}$ gibt das (1-δ)-Quantil der Standard-Normalverteilung N(0,1) an.
  Z.B. ist $z_{0.99} = 2.33$
- Mχ wächst ungefähr linear in der Anzahl der nicht-leeren Behälter k
- Typische Behältergröße: 0.5m*0.5m*15°
- Details siehe auch: *Dieter Fox, KLD-Sampling: Adaptive Particle Filters (2001)*