Kamerabasierte Lageschätzung eines autonomen Roboters mittels Partikelfiltern

Studienarbeit

von

Slawomir Grzonka
grzonka@informatik.uni-freiburg.de

Autor	Slawomir Grzonka
Betreuer	Dipl.-Inf. Christian Plagemann
Gutachter	Prof. Dr. Wolfram Burgard
Abgabetermin	13.09.2005

Kurzfassung

Ziel der vorliegenden Arbeit ist die Realisierung eines Echtzeitsystems zur räumlichen Lageschätzung und zeitlichen Verfolgung einer physikalischen Markierung. In vielen Anwendungsgebieten der Robotik wird die Kenntnis der Position und Orientierung von Objekten benötigt. In dieser Arbeit wird ein szenenbasiertes Verfahren auf Grundlage der Partikelfilter entwickelt. Im Gegensatz zu rein bildbasierten Verfahren lässt sich hier das Bewegungsmodell der zu verfolgenden Markierung unmittelbar integrieren und eine spätere Erweiterung um zusätzliche Sensoren, wie z. B. Laser, wird erleichtert. Zunächst werden die Grundlagen der projektiven Geometrie, wie die Abbildung einer Szene auf die Bildebene und die Kamerakalibierung, sowie das Partikelfilter-Verfahren erläutert. Anschließend wird das entworfene System vorgestellt und dessen praktische Implementierung beschrieben. Die Leistungsfähigkeit des Systems in praxisrelevanten Situationen wird durch eine qualitative und quantitative Evaluierung gezeigt.

Inhaltsverzeichnis

1. Einleitung

Im Rahmen des SFB/TR-8[1] Projektes wird an der Albert-Ludwigs-Universität Freiburg in Kooperation mit der Universität Bremen ein System zur räumlichen Lokalisierung und Verfolgung eines Roboters entwickelt. Zum Einsatz kommt die visuelle Objektverfolgung zur Unterstützung der kooperativen Kartenerstellung zweier Roboter: ein kleiner Roboter, welcher kaum Rechenleistung besitzt aber sehr mobil ist und ein großer, welcher über mehr Kapazitäten verfügt und mit einer Kamera ausgestattet ist. Letztgenannter ist aufgrund seiner Größe nicht in der Lage, auf unebenem Untergrund wie z. B. Sand oder Gras zu manövrieren. Der an der Universität Bremen entwickelte kleine Scorpion-Roboter[2] ist in der Lage seinen Untergrund zu ertasten und kann diese Informationen an seinen „großen Bruder“ liefern. Dieser kann dadurch seine Umgebungskarte um die Information der Bodenbeschaffenheit ergänzen und so Pfade, die durch unpassierbares Gelände führen, aus der Karte entfernen.

Ziel der vorliegenden Arbeit ist die Entwicklung eines geeigneten visuellen Markers und die Implementierung eines Systems zur räumlichen Schätzung der Lage und Orientierung des Markers in Echtzeit. Ausgehend von einer initialen Lageschätzung soll der Marker anhand der Kamerabilder zeitlich im Raum verfolgt werden. Dazu werden mögliche Markerposition bezüglich eines probabilistischen Bewegungsmodells parallel verfolgt, mittels der visuellen Messungen bewertet und selektiert. Schwankende Lichtverhältnisse und der Wunsch nach weitreichender Robustheit gegenüber Störeinflüssen bei gleichzeitig vertretbarem Rechenaufwand und möglichst hoher Genauigkeit stellen hohe Anforderungen an das System.

Die vorliegende Arbeit beginnt mit der Erläuterung der Grundlagen in Kapitel 2. Dort wird die Kamerakalibrierung und der mathematische Hintergrund, der zur Beschreibung der Abbildung einer Weltszene auf die Bildebene benötigt wird, vorgestellt. Kapitel 3 veranschaulicht das Partikelfilter-Verfahren und seine Anwendung. Kapitel 3.4 stellt das anhand der gesteckten Ziele erstellte System zur räumlichen Schätzung von Orientierung und Lage vor. In Kapitel 4 wird auf die durchgeführten Experimente, ihre Ergebnisse und Deutungen eingegangen. Abschließend werden in Kapitel 5 mögliche Erweiterungen und Verbesserungen des Systems diskutiert.

[1] http://www.informatik.uni-freiburg.de/~sfbtr8/
[2] http://ag47.informatik.uni-bremen.de/ger/projekt.php?id=3&details=ja

2. Geometrische Grundlagen und Kamerakalibrierung

Bevor eine geeignete Marke zur Objektverfolgung konstruiert und zugehörige Algorithmen entwickelt werden können, ist die präzise Modellierung der Szenen- und Abbildunggeometrie sowie die Herleitung eines geeigneten Kameramodells nötig.

Eine Kamera bildet die dreidimensionale Welt (im Folgenden *Szene* genannt) auf ein zweidimensionales Bild (im Folgenden *Bild* genannt) ab. Die Art und Weise, wie die Kamera eine gegebene Szene auf ein Bild projiziert, wird durch die Parameter des Kameramodells beschrieben. Um Rückschlüsse vom 2D-Bild auf die 3D-Szene ziehen zu können, müssen die Parameter des Kameramodells bekannt sein. Die Bestimmung dieser Parameter nennt man Kamerakalibrierung.

Im Folgenden werden die geometrischen Grundlagen der projektiven Geometrie dargestellt und ein konkretes Verfahren zur Kamerakalibrierung entwickelt. In dieser Arbeit wird durchgängig das linkshändige Koordinatensystem verwendet, d. h. die x-Achse zeigt nach rechts, die y-Achse nach oben und die z-Achse vom Betrachter weg. Die Winkel sind positiv bei Rotation gegen den Uhrzeigersinn, falls entlang der Achse vom Betrachter weg geblickt wird. Die formalen Grundlagen sind weitestgehend an [Bur03] angelehnt.

2.1. Notation

Für den formalen Umgang mit den in diesem Kapitel beschriebenen Verfahren wird die folgende Notation verwendet.

$$\begin{aligned}\mathbf{s} &= (x_s, y_s, z_s) : \text{Szenenpunkt bzgl. des Szenenkoordinatensystems}\\ \mathbf{b} &= (x_b, y_b) : \text{Bildpunkt bzgl. des Bildkoordinatensystems}\\ \mathbf{c} &= (x_c, y_c, z_c) : \text{Szenenpunkt bzgl. des Kamerakoordinatensystems}\end{aligned}$$

In homogenen Koordinaten:

$$\begin{aligned}\tilde{s} &= (x_s, y_s, z_s, 1)\\ \tilde{b} &= (x_b, y_b, 1)\\ \tilde{c} &= (x_c, y_c, z_c, 1)\end{aligned}$$

H bezeichnet eine Homographie.

2.2. Projektive Koordinaten

Ein Punkt im n-dimensionalen euklidischen Raum $\mathbb{R}^n$ wird durch einen n-dimensionalen Vektor $\mathbf{x} = (x_1, ..., x_n)^T \in \mathbb{R}^n$ beschrieben. Bei dieser Darstellung spricht man auch von inhomogenen Koordinaten. Ein Punkt im n-dimensionalen projektiven Raum $\mathbb{P}^n$ wird durch einen $n+1$-dimensionalen Vektor $\tilde{\mathbf{x}} = (x_1, ..., x_n, x_{n+1})^T \in \mathbb{R}^{n+1} \backslash \mathbf{0}$ beschrieben. Bei dieser Darstellung spricht man auch von homogenen Koordinaten.

Zwei Vektoren $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}} \in \mathbb{P}^n$ sind genau dann identisch, wenn ihre homogenen Koordinaten äquivalent sind. Die Äquivalenz wird dabei definiert durch die Relation $\sim$ mit

$$\tilde{\mathbf{x}} \sim \tilde{\mathbf{y}} :\Leftrightarrow \exists \lambda \in \mathbb{R}\backslash\{0\} : \tilde{\mathbf{x}} = \lambda \tilde{\mathbf{y}} \,.$$

Die euklidischen Koordinaten eines Punktes lassen sich in projektive umwandeln:

$$\mathbf{x} = (x_1, ..., x_n) \;\rightarrow\; \tilde{\mathbf{x}} = (\lambda x_1, ..., \lambda x_n, \lambda),\; \lambda \in \mathbb{R}\backslash\{0\} \,.$$

2.3. Projektive Transformation

Eine projektive Transformation ist definiert als invertierbare Abbildung im projektiven Raum. Sie bildet Punkte auf Punkte und Geraden auf Geraden ab. Sie kann als lineare Abbildung in homogenen Koordinaten beschrieben werden:

$$\tilde{\mathbf{x}}' \sim \mathbf{H}\tilde{\mathbf{x}} \,.$$

Die einzige Invariante ist das Doppelverhältnis. Das Doppelverhältnis ist der Quotient der Teilverhältnisse von vier Punkten auf einer Geraden. Seien A, B, C und D Punkte auf einer Geraden und A', B', C' und D' die transformierten Punkte, so gilt

$$\frac{\frac{\overline{AC}}{\overline{BC}}}{\frac{\overline{AD}}{\overline{BD}}} = \frac{\frac{\overline{A'C'}}{\overline{B'C'}}}{\frac{\overline{A'D'}}{\overline{B'D'}}} \tag{1}$$

Eine projektive Transformation wird auch Homographie genannt.

2.4. Der Abbildungsprozess

Der physikalische Abbildungsvorgang wird in dieser Arbeit durch ein vereinfachtes Modell beschrieben. Zunächst wird das Modell einer Lochkamera vorgestellt. Dieses wird anschließend zum verwendeten Abbildungsmodell verfeinert. Für viele praktische Anwendungen beschreibt dieses Modell den Abbildungsvorgang hinreichend genau.

Eine Lochkamera bildet Szenenpunkte mittels Zentralprojektion auf Bildpunkte ab.
Wie in Abbildung 1 dargestellt, liegen Szenenpunkt $\mathbf{c}$ und Bildpunkt $\mathbf{b}$ auf einer Geraden, die durch das Kamerazentrum geht. Ein Szenenpunkt $\mathbf{c}$ wird durch die Koordinaten (x_c, y_c, z_c) dargestellt. Die Bildebene ist die Ebene, für die bezüglich des Kamerakoordinatensystems $z_c = f$ gilt. f bezeichnet die Brennweite der Kamera. Die optische Achse wird durch den Richtungsvektor $(0, 0, 1)$ und Aufpunkt $(0, 0, 0)$ gegeben. Sie trifft die Bildebene im Hauptpunkt $(0, 0, f)$. Der Bildpunkt $\mathbf{b} = (x_b, y_b)$ hat somit die 3D-Koordinaten (x_b, y_b, f). Anwenden des Strahlensatzes liefert

$$\frac{x_b}{x_c} = \frac{y_b}{y_c} = \frac{f}{z_c} \tag{2}$$

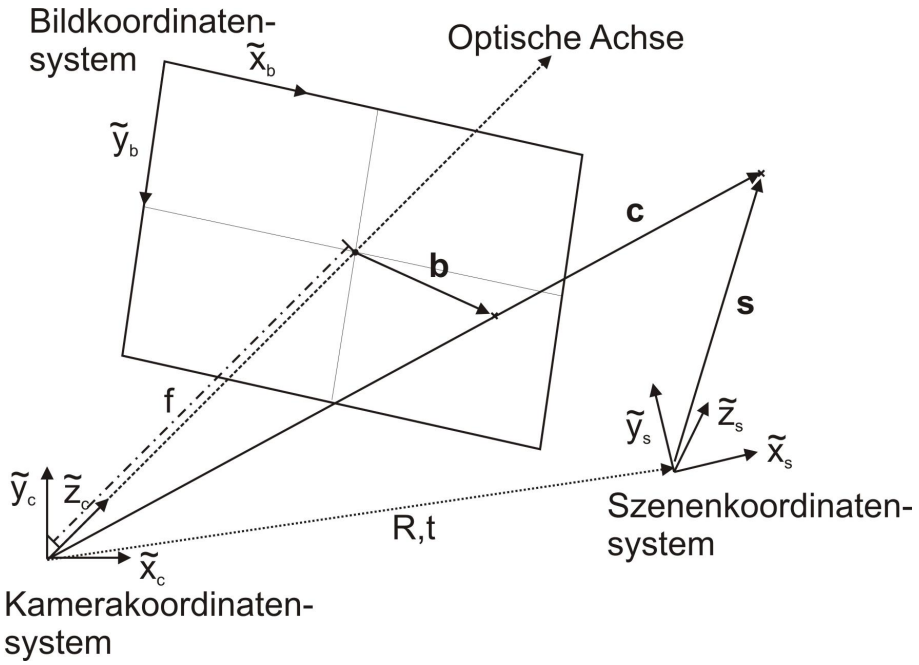

Abbildung 1: Abbildung eines Szenenpunktes **s** auf den Bildpunkt **b**

und daher gilt für die Abbildung eines Szenenpunktes **c** auf den Bildpunkt **b**

$$(x_c, y_c, z_c) \rightarrow (\frac{f}{z_c}x_c, \frac{f}{z_c}y_c)\,.$$

In homogener Koordinatendarstellung gilt

$$(\frac{f}{z_c}x_c, \frac{f}{z_c}y_c, 1) \sim (fx_c, fy_c, z_c)\,. \tag{3}$$

In Matrixschreibweise wird die Linearität der Abbildung deutlich:

$$\begin{pmatrix} fx_c \\ fy_c \\ z_c \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} \Leftrightarrow \tilde{\mathbf{b}} \sim \mathbf{P}\tilde{\mathbf{c}}\,. \tag{4}$$

Die Matrix **P** lässt sich schreiben als:

$$\mathbf{P} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \mathbf{K}\mathbf{P}_\mathbf{p}\,. \tag{5}$$

Dabei bezeichnet **K** die *internen* Kameraparameter. In der Praxis nimmt die Matrix **K** nicht diese einfache Form an, weshalb sie im Folgenden auf die allgemeine Form gebracht werden soll. Da die optische Achse im Allgemeinen nicht durch den Ursprung des

Bildkoordinatensystems geht, werden die Bildkoordinaten (x, y) um einen Offset (x_0, y_0) verschoben. Zudem kann es vorkommen, dass der Bildsensor nicht aus quadratischen Pixeln aufgebaut ist. Dadurch wird das Bild in x- und y-Richtung unterschiedlich stark skaliert, d. h. die Brennweiten in x- und y-Richtung sind verschieden. Im Allgemeinen liegt die optische Achse nicht senkrecht zur Ebene des Bildsensors. In diesem Fall tritt eine Scherung auf, die durch den Scherungsparameter s ausgedrückt wird. Somit erweitert sich $\mathbf{K}$ zu

$$\mathbf{K} = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6}$$

Der Zusammenhang zwischen Szenen- und Kamerakoordinatensystem wird durch eine euklidische Transformation beschrieben:

$$\mathbf{c} = \mathbf{R}\mathbf{s} + \mathbf{t} \tag{7}$$

$$\text{mit } \mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}\text{: Isometrie (Rotation) und}$$

$$\mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}\text{: Translation}$$

Die Rotationsmatrix $\mathbf{R}$ und der Translationsvektor $\mathbf{t}$ beschreiben die *externen* Kameraparameter. Die Projektionsmatrix $\mathbf{P}$ erweitert sich zu:

$$\mathbf{P} = \mathbf{K}\mathbf{P_p}\begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} = \mathbf{K}(\mathbf{R}|\mathbf{t}),\ \mathbf{0} \in \mathbb{R}^3 \tag{8}$$

mit

$$(\mathbf{R}|\mathbf{t}) := \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix}. \tag{9}$$

Im Allgemeinen treten radiale Linsenverzerrungen auf. Geraden werden dann auf Kurven abgebildet. Die Stärke der Linsenverzerrung ist abhängig vom Abstand des Bildpunktes vom Bildmittelpunkt. Seien $\mathbf{b} = (x_b, y_b)$ die linearen (unverzerrten) Bildkoordinaten und $\mathbf{b}_d = (x_{bd}, y_{bd})$ die realen (verzerrten) Bildkoordinaten. Das Zentrum der Verzerrung sei der Bildhauptpunkt $\mathbf{b}_0 = (x_{b0}, y_{b0})$. Dann kann die Linsenverzerrung modelliert werden durch

$$(\mathbf{b}_d - \mathbf{b}_0) = L(d)(\mathbf{b} - \mathbf{b}_0) \tag{10}$$

mit einer geeigneten Linsenfunktion L und $d = \|\mathbf{b} - \mathbf{b}_0\|$, wobei $L(0) = 1$ gilt. Die unbekannte Funktion L lässt sich durch eine Taylorreihe um 0 ausdrücken:

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \kappa_4 r^4 + \cdots \quad . \tag{11}$$

Für praktische Anwendungen lässt sich (11) hinreichend genau durch ein Polynom zweiten Grades approximieren:

$$\mathbf{b}_d = \mathbf{b} + (\mathbf{b} - \mathbf{b}_0)(\kappa_1 r + \kappa_2 r^2)\,. \tag{12}$$

Es gilt somit, neben $\mathbf{K}(\mathbf{R}|\mathbf{t})$ auch die Parameter (κ_1, κ_2) zu bestimmen.

2.5. Kamerakalibrierung

Im Folgenden wird ein Verfahren zur Bestimmung der Matrix $\mathbf{K}$ und der Parameter der radialen Linsenverzerrung, die so genannte 2D-Kalibreirung, vorgestellt. Alternative Verfahren wie 3D-Kalibrierung oder Selbstkalibrierung werden in [Bur03] beschrieben. Der Einfluss der Linsenverzerrung kann unabhängig von der Bestimmung von $\mathbf{K}$ betrachtet werden. Da die interne Kalibriermatrix $\mathbf{K}$ unabhängig von der externen Kalibriermatrix $(\mathbf{R}|\mathbf{t})$ ist, können Bedingungen für $\mathbf{K}$ durch Punktkorrespondenzen zwischen Bild- und Szenenpunkten bezüglich unterschiedlicher Szenenkoordinatensystemen abgeleitet werden. Seien Punkte auf einer Ebene gegeben, so kann o. B. d. A. die x-y-Ebene des Szenenkoordinatensystems auf diese Ebene gelegt werden. Bezüglich des Szenenkoordinatensystems besteht die Ebene aus den Punkten $(x_s, y_s, 0)$ mit $x_s, y_s \in \mathbb{R}$.
Nach Umschreibung von (9) in $(\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{t})$ mit $\mathbf{r_j} := (r_{1j}, r_{2j}, r_{3j})^T$ kann die Projektion einer Ebene durch

$$\begin{pmatrix} x_b \\ y_b \\ 1 \end{pmatrix} \sim \mathbf{K}(\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}, \mathbf{t}) \begin{pmatrix} x_s \\ y_s \\ 0 \\ 1 \end{pmatrix} \tag{13}$$

beschrieben werden. Der Spaltenvektor $\mathbf{r_3}$ hat keinen Einfluss auf das Ergebnis. Daher können wir die Projektion durch eine reine 2D-2D-Transformation beschreiben:

$$\begin{pmatrix} x_b \\ y_b \\ 1 \end{pmatrix} \sim \mathbf{K}(\mathbf{r_1}, \mathbf{r_2}, \mathbf{t}) \begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix}\,. \tag{14}$$

Es gilt:

$$\tilde{\mathbf{b}} \sim \mathbf{H}\tilde{\mathbf{s}}_{Ebene}\,. \tag{15}$$

Die Homographie $\mathbf{H}$ ist demnach eine 3×3-Matrix, die abzüglich des Freiheitsgrades für Skalierungsinvarianz acht Freiheitsgrade besitzt. Da jede Punktkorrespondenz zwei Gleichungen liefert, sind vier Punkte nötig, um $\mathbf{H}$ zu bestimmen. Ausgehend von $\mathbf{H}$ können nun Bedingungen für $\mathbf{K}$ abgeleitet werden [Bur03].

Ist $\mathbf{K}$ bekannt, so kann mittels $\mathbf{H}$ die Position und Orientierung der Ebene berechnet werden:

$$\mathbf{H} = (\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}) = \lambda \mathbf{K}(\mathbf{r_1}, \mathbf{r_2}, \mathbf{t}). \tag{16}$$

Die Rotationsmatrix $\mathbf{R}$ ist eine orthogonale Matrix mit $\|\mathbf{r_i}\| = 1$ für $i \in \{1, 2, 3\}$ und $\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}$. $\mathbf{K}$ ist definitionsgemäß invertierbar. Mit $\lambda = \|\mathbf{K}^{-1}\mathbf{h_1}\|$ folgt

$$(\mathbf{r_1}, \mathbf{r_2}, \mathbf{t}) = \frac{\mathbf{K}^{-1}\mathbf{H}}{\|\mathbf{K}^{-1}\mathbf{h_1}\|}. \tag{17}$$

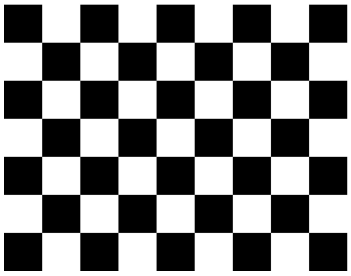

Abbildung 2: Das verwendete Kalibriermuster

Abbildung 3: Zwei der 16 Aufnahmen mit überlagerten Punkten, die vom Algorithmus für Punktkorrespondenzen verwendet werden (rechts unten: erster Punkt, links oben: letzter Punkt).

Somit kann mit Hilfe von vier bekannten Punkten in der Ebene und einer kalibrierten Kamera die Lage dieser Ebene im Raum bestimmt werden.

2.6. Praktische Umsetzung der Kamerakalibrierung

Als Kalibriermuster wird ein Schachbrettmuster (siehe Abbildung 2) verwendet. Für Punktkorrespondenzen werden die inneren Ecken des Schachbrettmusters extrahiert. Das Muster wurde aus 16 verschiedenen Positionen aufgenommen. Abbildung 3 zeigt zwei der 16 Aufnahmen mit markierten Punkten, die für die Punktkorrespondenzen verwendet wurden. Die Realisierung erfolgte mit Hilfe der frei verfügbaren C++ Bibliothek *OpenCV*[3]. Das Verfahren wurde dabei so implementiert, dass der Anwender angeben kann durch wieviele Aufnahmen die Kalibrierung gestützt werden soll. Der Algorithmus entscheidet bei jeder Aufnahme, ob die gesuchte Anzahl an Punktkorrespondenzen gefunden wurde und verwertet oder verwirft die Aufnahme entsprechend.

[3] http://sourceforge.net/projects/opencvlibrary/

3. Szenenbasierte Objektverfolgung mit Partikelfiltern

In Kapitel 2 wurden die mathematischen Grundlagen, die bei der Abbildung einer Weltszene mit einer Kamera auf die Bildebene gegeben sind, beschrieben. Die Kamera als Sensor unterstützt das Lokalisierungsverfahren bei der Bildung einer a-posteriori-Wahrscheinlichkeitsverteilung im 6D-Szenenraum. Diese Wahrscheinlichkeitsverteilung kann z. B. aufgrund der Bewegung des Roboters und der verrauschten Messung beliebig kompliziert und multimodal werden. Eine parametrisierte Darstellung, wie z. B. durch eine einfache Gaußkurve, ist deshalb meist unzureichend. Im gesamten Zustandsraum befinden sich zudem nur wenige in Frage kommende 6D-Positionen. Eine Aktualisierung der a-posteriori-Wahrscheinlichkeitsverteilung auf dem gesamten Raum ist deshalb nicht nötig und meist nicht praktikabel. Partikelfilter sind ein elegantes Werkzeug für die kontinuierliche Approximation von Wahrscheinlichkeitsverteilungen. Metropolis und Ulam [UM49] legten 1949 die Grundsteine für die modernen Partikelfilter-Varianten. Isard und Blake [BI96] übertrugen diesen Ansatz mit dem Condensation-Algorithmus (**Con**ditional **Dens**ity Propag**ation**) auf die Partikelfilter im Bereich der Bildverarbeitung, mit denen es möglich ist, mehrere Hypothesen zeitgleich zu verfolgen. In diesem Kapitel werden die für die Studienarbeit relevanten Bereiche der Partikelfilter erläutert. Die formalen Grundlagen sind an [TBF05] und [RAG04a] angelehnt.

3.1. Notation

Für den formalen Umgang mit den in diesem Kapitel beschriebenen Verfahren sowie für Anhang A wird die folgende Notation verwendet.

$x_t^{[m]}$	Zustand von Partikel m zum Zeitpunkt t
χ_t	$:= \{x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}\}$
$w_t^{[m]}$	Gewicht des Partikels m zum Zeitpunkt t
u_t	Kommando zum Zeitpunkt $(t-1) \rightarrow t$
z_t	Messung zum Zeitpunkt t
$x_{0:t}^{[m]}$	$:= \{x_0^{[m]}, x_1^{[m]}, ..., x_t^{[m]}\}$
$u_{0:t}$	$:= \{u_0, u_1, ..., u_t\}$
$z_{0:t}$	$:= \{z_0, z_1, ..., z_t\}$
$bel(x_t)$	= Verteilung von x zum Zeitpunkt t

3.2. Partikelfilter

Partikelfilter werden zur fortlaufenden Aktualisierung einer Wahrscheinlichkeitsfunktion durch neue Messungen und Kontrollkommandos verwendet. Sie sind eine nichtparametrisierte Implementierung des Bayes-Filters [Mit97]. Partikelfilter approximieren die a-posteriori-Verteilung von Zustandswahrscheinlichkeiten eines Systems durch eine endliche Menge an Parametern. Die Grundidee ist, eine Wahrscheinlichkeitsfunktion durch eine Stichprobenmenge, die Partikel, zu approximieren. Abbildung 4a veranschaulicht dies. Im Gegensatz zu alternativen Ansätzen wie z. B. dem Kalmanfilter können Partikelfilter durch ihre nicht-parametrische Form beliebige Verteilungen approximieren. In

der Praxis wird dies allerdings dadurch eingeschränkt, dass nur eine endliche Menge von Partikeln zur Verfügung steht. Ein weiterer Vorteil dieser Repräsentation ist die Möglichkeit, nichtlineare Transformationen darauf anzuwenden. Die Kernidee des mathematischen Korrektheitsbeweises wird in Anhang A hergeleitet. Der Grundalgorithmus nach [TBF05] lautet:

1: **Algorithmus Partikelfilter**(χ_{t-1}, u_t, z_t) :
2: $\overline{\chi_t} = \chi_t = \emptyset$
3: for $m = 1$ to M do
4: ziehe $x_t^{[m]}$ aus $p(x_t|u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t|x_t^{[m]})$
6: $\overline{\chi_t} = \overline{\chi_t} \cup \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: ziehe i mit Wahrscheinlichkeit proportional zu $w_t^{[i]}$
10: $\chi_t = \chi_t \cup x_t^{[i]}$
11: endfor
12: return χ_t

Tabelle 1: Grundversion des Partikelfilter-Algorithmus

Im Detail lässt sich der Algorithmus wie folgt erklären:
Zeile 4 generiert den Folgezustand $x_t^{[m]}$ für Zeitpunkt t basierend auf dem bisherigen Zustand $x_{t-1}^{[m]}$ und dem Kommando u_t (z. B. Bewegung). Da das Bewegungsmodell in nichtdeterministischer Form gegeben ist, wird aus $p(x_t|u_t, x_{t-1}^{[m]})$ gezogen.

Zeile 5 berechnet für jedes Partikel das Gewicht (engl. *importance factor*) $w_t^{[m]}$. Das Gewicht ist die Wahrscheinlichkeit, die Messung z_t zu erhalten, gegeben die Hypothese, die durch das Partikel x_t^m repräsentiert wird. Also

$$w_t^{[m]} = p(z_t|x_t^{[m]}) \,.$$

Die Menge der gewichteten Partikel repräsentiert die a-posteriori-Verteilung.

Zeilen 8-11 implementieren den *resampling*-Schritt (*importance sampling*). Dieser Schritt bewirkt eine Konzentration der Partikelmenge auf die wahrscheinlichsten Bereiche des Zustandsraumes. Das Auswahlprinzip gleicht dem „der Stärkste überlebt“, wie es auch bei Genetischen Algorithmen angewendet wird. Ein Partikel überlebt (kommt in die nächste Iteration) mit einer Wahrscheinlichkeit proportional zu seinem Gewicht.

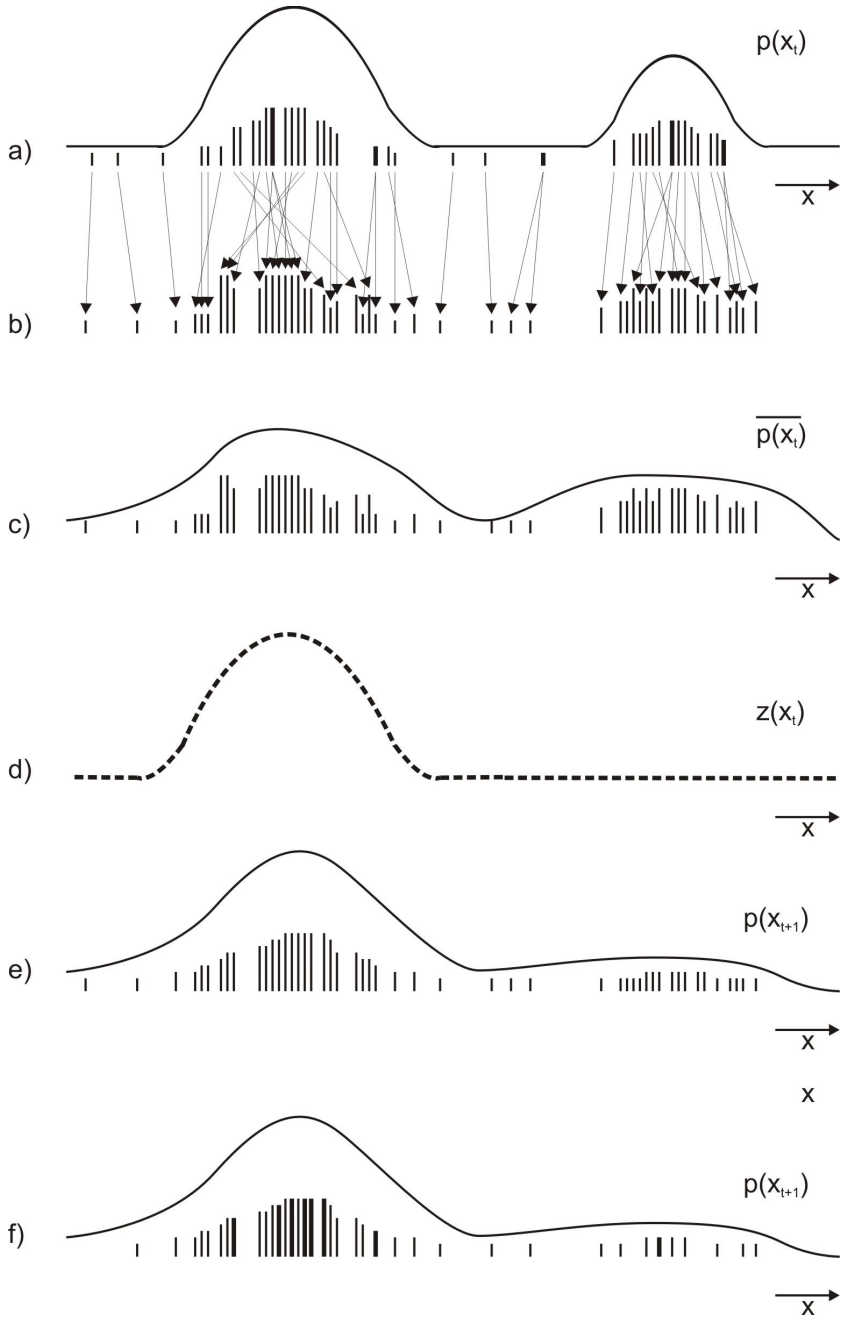

Abbildung 4: Der Partikelfilter-Algorithmus: a) Die wahre a-priori-Verteilung der Zustandswahrscheinlichkeiten und die Repräsentation durch Partikel. Das Gewicht der einzelenen Partikel wird durch ihre Strichlänge visualisiert. b) Der Filterzustand nach Anwendung des Bewegungsmodells und (c) die dadurch approximierte Verteilung. d) Die Messung in der aktuellen Iteration e) Die Gewichte der Partikel wurden durch die Messung angepasst f) Resampling-Schritt: Partikel werden proportional zu ihrem Gewicht (ggf. mehrfach oder gar nicht) kopiert.

1:	**Algorithmus Low_Variance_Sampler**$(\chi_\mathbf{t}, \mathbf{W_t})$:
2:	$\overline{\chi}_t = \emptyset$
3:	$r = rand(0, M^{-1})$
4:	$c = w_t^{[1]}$
5:	$i = 1$
6:	for $m = 1$ to M do
7:	$U = r + (m-1)M^{-1}$
8:	while $U > c$ do
9:	$i = i + 1$
10:	$c = c + w_t^{[i]}$
11:	endwhile
12:	$\overline{\chi}_t = \overline{\chi t} + x_t^{[i]}$
13:	endfor
14:	return $\overline{\chi}_t$

Tabelle 2: der *low variance sampling*-Algorithmus

3.3. Der Resampling-Schritt

Eine wichtige Fehlerquelle bei der Anwendung der Partikelfilter liegt in der durch das wiederholte Iterieren verursachten Variation der Partikel. Da die Partikel die Originalverteilung approximieren, herrscht immer ein Unterschied zwischen der Verteilung, die durch die Partikel repräsentiert wird, und der Originalverteilung. Diese Variation wird auch *Varianz* genannt.
Die Varianz wird im Laufe der Iterationen durch das stetige Resampling verstärkt. Aufgrund der zufälligen Ziehung von einer Iteration in die andere führt es aufgrund des zufälligen Ereignisses zwangsweise dazu, dass einige Partikel mehrfach gezogen werden, während andere aussterben. Daher kann es vorkommen, dass auch „gute“ Partikel aussterben. Um dies zu verhindern, wird in dieser Studienarbeit das Resampling nach dem Prinzip des *low variance sampling* [TBF05] verwendet. Hierzu wird ein Partikel nicht einfach unabhängig von den anderen Partikeln proportional zu seinem Gewicht gewählt, welches einem Rouletteziehen gleich kommt (siehe Abbildung 5a). Das verwendete Verfahren besteht aus drei Schritten: zuerst werden die Partikel durch ihre Gewichte aneinandergereiht. Anschließend wird zufällig ein Startpartikel gewählt und von diesem aus die restliche Anzahl an Partikeln zu gleichen Abständen entlang der Gewichtsachse gezogen. Tabelle 2 zeigt den Algorithmus und Abbildung 5b veranschaulicht dessen Arbeitsweise. Ein weiterer Vorteil liegt in der linearen Laufzeit des Verfahrens.

3.4. Visuelle Objektverfolgung

In diesem Abschnitt wird beschrieben, wie das theoretische Gerüst der letzten Abschnitte im Rahmen dieser Arbeit verwendet werden kann. Dazu wird das implementierte System vorgestellt und Umgebungsvoraussetzungen deutlich gemacht. Die im letzten

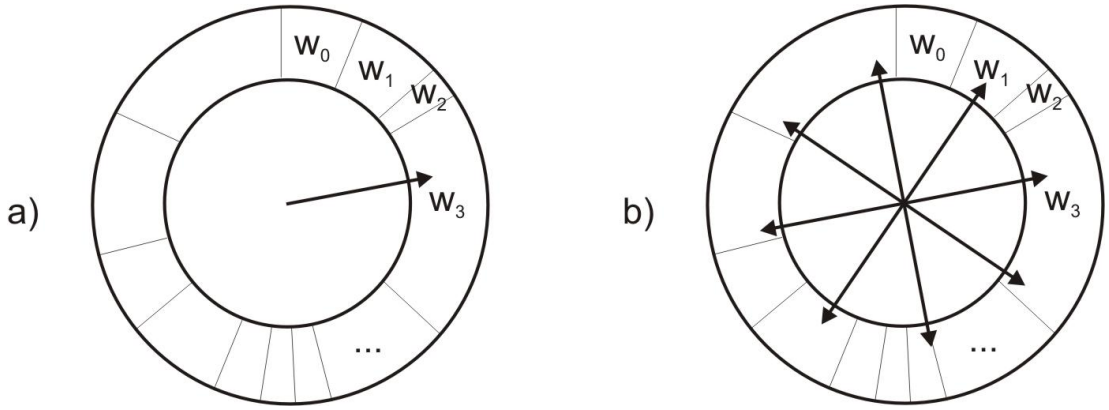

Abbildung 5: Der Resampling-Schritt: a) einfaches Resampling proportional zum Gewicht (Roulette-Resampling) b) Resampling nach dem *low variance sampling*-Verfahren

Abschnitt vorgestellten Partikelfilter leisten hierbei die Verfolgung der Markierung im 6D-Szenenraum anhand von 2D-Eingangsbildern. Im Folgenden wird ein Überblick über alle benötigten Elemente zur räumlichen Schätzung einer Markierung gegeben.

3.4.1. Der Marker

Das Verfahren soll sowohl im Außenbereich als auch in Innenräumen einsetzbar sein. Dabei soll der entwickelte Marker auf mindestens vier Meter Entfernung erkennbar sein. Das System soll möglichst robust gegenüber Störungen sein und in Echtzeit funktionieren. Für diese Voraussetzungen hat sich ein Marker mit folgenden Eigenschaften als geeignet erwiesen: Eine Styroporkugel mit einem Durchmesser von 10 cm. Im Abstand von je 90° sind zwei Markierungen angebracht (siehe Abbildung 6). Die oberen Markierungen besitzen einen Durchmesser von 2.1 cm, die unteren einen Durchmesser von 3.5 cm. Die Markierungen liegen auf der oberen Halbkugel, da der Marker im praktischen Einsatz in Bodennähe aufzufinden ist, während sich die Kamera 150 cm über dem Marker befinden wird. Die Farben der Markierung wurden gemäß dem Graycode-Verfahren [Boo64] angebracht. Dies gewährleistet, dass keine Rotationssymmetrie entsteht. Abbildung 6 zeigt vier Aufnahmen der Kugel als 360° Rundumsicht. Jeder Farbkreis wird im Verfolgungsverfahren durch fünf Punkte repräsentiert: ein Punkt in der Mitte und je einer in Nord-, Süd-, Ost- und West-Richtung im Abstand von 0.5 cm vom Mittelpunkt bei den oberen Markierungen und 1.2 cm vom Mittelpunkt bei den unteren.

3.4.2. Farbsegmentierung

Um die Merkmale auf der Kugel, also die farblichen Markierungen, erkennen zu können, wird das Eingangsbild einer Farbsegmentierung unterzogen. Hierzu wird das Bild in den

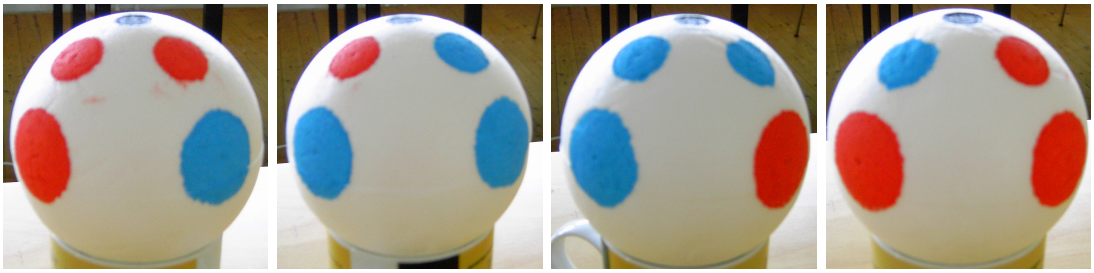

Abbildung 6: Der verwendete Marker aus verschiedenen horizontalen Blickrichtungen

Abbildung 7: a) Ausschnitt eines Eingansbildes. b) Eingangsbild überlagert mit der Farbsegmentierung „rot“. c) Binärbild nach Farbsegmentierung „rot“. d) Faltung des Binärbildes mit einem 5×5-Gaußkernel.

HSV-Farbraum transformiert. Um Helligkeitsinvarianz zu erhalten, wird es bezüglich der Helligkeitskomponente (V) derart normalisiert, dass die mittlere Helligkeit einen Wert von 127 bei einer Skala von von 0 bis 255 annimmt. Der Vorteil der Konvertierung in den HSV-Farbraum ist, dass dort Helligkeitsbereiche einfach selektiert werden können. Während im RGB-Farbraum der Unterschied zwischen einem hellen und einem dunklen Farbton in allen drei Farbkomponenten sichtbar ist, ändert sich im HSV-Farbraum nur die V-Komponente (engl. *value*=Helligkeit). Um einen Farbton helligkeitsinvariant zu erkennen, werden sechs Schwellwerte angegeben: $hue_{min/max}, saturation_{min/max}, value_{min/max}$. Abbildung 7a zeigt den Ausschnitt eines Eingangsbildes und die zugehörige Segmentierung bezüglich des Farbtons „rot“ (7b und 7c).

3.4.3. Der Sensor

In jeder Iteration müssen die Gewichte der Partikel aktualisiert werden. Dazu muss die bedingte Wahrscheinlichkeit $p(z_t|x_t)$ berechnet werden. Nach der Farbsegmentierung liegt für jeden Farbkanal ein Binärbild vor. Die Messunsicherheit wird durch eine Gaußverteilung approximiert. In der Anwendung wird dies effizient durch die Faltung der binären Farbkanalbilder mit einem Gaußkernel (hier: 5×5) erreicht. Das tiefpassgefiltere

Bild kann nun als eine Gewichtung bezüglich des Abstandes interpretiert werden. Abbildung 7d zeigt das Bild aus Abbildung 7c nachdem es mit dem Gaußkernel gefaltet wurde. Alternativen zu einem auf Farbsegmentierung basierendem Sensor wären Kantenfilter, wie sie z. B. in [IB96] verwendet werden.

3.4.4. Das Bewegungsmodell

Sowohl der Marker, als auch die Kamera können sich frei bewegen. Daher unterliegen die Partikel in jeder Iteration einer Bewegung. Es gibt mehrere Möglichkeiten, solche Mischbewegungen zu modellieren. Eine Möglichkeit ist, alle Bewegungen nacheinander auszuführen. Eine andere Möglichkeit, wie sie von Isard und Blake [IB98] verwendet wird, besteht aus nur einem Bewegungsmodell, welches automatisch aus einer Menge von Modellen ausgewählt wird. In dieser Arbeit wird davon ausgegangen, dass die Kamera fest ist und nur der Marker eine Bewegung vollzieht. Die Bewegung wird durch eine Gauß'sche Verteilung approximiert. Da aufgrund der Aufgabenstellung der Marker zwischen zwei Iterationen nur eine geringe Bewegung vollzieht, ist das Bewegungsmodell unabhängig von der Bewegungshistorie. Der Mittelwert jeder Gaußverteilung ist also Null. Die Varianzen hängen vom Einsatzgebiet (Geschwindigkeit des Markers) und von der Dauer der Verarbeitungsschritte je Iteration (Dauer zwischen zwei Messungen) ab.

3.4.5. Das Verfolgungsverfahren

In den oberen Abschnitten wurden die Voraussetzungen für eine Verfolgung mittels eines Partikelfilters geschaffen. Geschätzt werden soll die genaue Lage des Markers, also seine Position (x, y, z) und die zugehörigen Winkel (*Neigungswinkel* (engl. *pitch*), *Gierwinkel* (engl. *yaw*), *Rollwinkel* (engl. *roll*)). Dazu wird eine szenenbasierte Verfolgung im sechsdimensionalen Raum vollzogen. Alternativ ließe sich das zweidimensionale Abbild auf der Bildebene verfolgen und dies als Grundlage für eine sechsdimensionale Schätzung verwenden. In dieser Arbeit wird der direkte Weg eingeschlagen und als Grundraum der sechsdimensionale Raum aller möglichen Markerpositionen in der Szene verwendet. Dazu wird zuerst die 6D-Position des Markers geschätzt. Anschließend wird die Schätzung mit dem aktuellen Kamerabild verglichen. Gerade wegen der hohen Dimensionalität und der verrauschten Messungen ist es wichtig, mehrere Hypothesen zeitgleich verfolgen zu können. Falls sich der Marker zwischen zwei Iterationen zum Beispiel nur gering nach links bewegt, so ist es aufgrund des Kamerabildes zunächst nicht ersichtlich, ob sich der Marker nach links bewegt oder nur leicht um die eigene Achse gedreht hat. Der Kalmanfilter [RAG04b], der nur eine Hypothese verfolgen kann, ist deshalb hier nicht einsetzbar. Die Unsicherheit könnte zwar durch Modellierung einer hohen Varianz kompensiert werden, dadurch wäre die Genauigkeit der Positionsschätzung jedoch gering. Tabelle 3 veranschaulicht die Arbeitsweise des Verfolgungsalgorithmus in Pseudocode.

```
1:  Particle_Filter_Tracker ():
2:      particles = initializeParticleSet (numParticles, initialState)
3:      cp = loadCameraParameters ()
4:      cam = connectToCamera (ip, port)
5:      while (true) do
6:          image = getNextFrame (cam)
7:          for i = 1 to numParticles do
8:              applyMotionModel (particle[i])
9:          endfor
10:         convertToHSVAndNormalize (image)
11:         sred = segmentImage (image, colorRed)
12:         sblue = segmentImage (image, colorBlue)
13:         gaussRed = applyGaussianKernel (sred)
14:         gaussBlue = applyGaussianKernel (sblue)
15:         for i = 1 to numParticles do
16:             spherePoints = calculatePositionAndOrientationOfSphere (particle[i])
17:             visibleRed = removeInvisiblePoints (spherePoints, red)
18:             visibleBlue = removeInvisiblePoints (spherePoints, blue)
19:             projRed = calculateProjectionToCamera(visibleRed, cp)
20:             projBlue = calculateProjectionToCamera(visibleBlue, cp)
21:             for every actualPoint in projRed do
22:                 p = lookupProbability (gaussRed, actualPoint)
23:                 // == p(z_t = red|x_t) die Wahrscheinlichkeit an der
24:                 // erwarteten Position ein rotes Pixel zu finden.
25:                 weight(particle[i]) = weight(particle[i]) · p
26:             endfor
27:             for every actualPoint in projBlue do
28:                 p = lookupProbability (gaussBlue, actualPoint)
29:                 // == p(z_t = blue|x_t) die Wahrscheinlichkeit an der
30:                 // erwarteten Position ein blaues Pixel zu finden.
31:                 weight(particle[i]) = weight(particle[i]) · p
32:             endfor
33:         endfor
34:         resampleWithLowVarianceResampler (particles)
35:     endwhile
```

Tabelle 3: Pseudocode des Verfolgungsverfahrens

Im Detail:
Zeile 2: Initialisierung der Partikelmenge gemäß einer Startverteilung. Ansätze zur Selbstinitialiserung werden im Ausblick angegeben.

Zeilen 7 bis 9: Jedes Partikel wird entsprechend des Bewegungsmodelles in allen Dimensionen bewegt.

Zeilen 10 bis 14: Das Eingangsbild wird in den HSV-Farbraum transformiert und bezüglich der Helligkeitskomponente normalisiert. Anschließend werden Farbfilter (hier: rot und blau) auf das Bild angewendet, was je ein Binärbild für den jeweiligen Farbkanal produziert. Die Binärbilder werden mit einem Gaußkernel gefaltet. Dadurch kann das Bild als Gewichtsverteilung bezüglich des Abstandes interpretiert werden.

Zeilen 16 bis 20: für jedes Partikel wird nun die Position der Kugel berechnet. Dazu wird die Kugel gemäß der 3 Rotationswinkel gedreht und verschoben. Anschließend wird berechnet, welche der Punkte der Kugel auf der Vorderseite bezüglich der Kamera liegen. Diese Punkte werden gemäß der internen Kameraparameter auf die Bildebene projiziert. Angenommen ein Partikel würde den aktuellen Weltstatus richtig schätzen, so müsste im aktuellen Kamerabild an den projizierten Stellen im jeweiligen Farbkanal eine hohes Gewicht vorhanden sein.

Zeilen 21 bis 32: Die Gewichte, wie gut die projizierten Punkte mit dem Weltbild zusammenpassen, können nun direkt in den Farbkanalbildern abgelesen werden.

Zeile 34: Resampling nach der *low-variance*-Methode, wie sie in Kapitel 3.3 beschrieben wurde.

Abbildung 8: Der für das Experiment verwendete Roboter mit angebrachtem Marker. Die Fahrtgeschwindigkeit betrug ca. 3.5 cm/s, die Rotationsgeschwindigkeit ca. 50°/s.

4. Evaluierung

Die Anwendbarkeit des Verfahrens sowie die erreichbare Schätzgenauigkeit wurden in mehreren Experimenten untersucht. In diesem Kapitel werden zwei der Experimente und ihre quantitativen Ergebnisse im Detail dargestellt. Der entwickelte Marker wurde auf einem kleinen, autonomen Roboter angebracht. Der Roboter ist in Abbildung 8 mit aufgesetztem Marker dargestellt. Er bewegt sich mit einer Fahrtgeschwindigkeit von ca. 3.5 cm/s und besitzt eine Rotationsgeschwindigkeit von ca. 50°/s. Das Experiment wurde abends bei Bürobeleuchtung durchgeführt. Die Kamera befand sich ca. 95 cm über (y-Achse) dem Roboter und zwischen 430 cm und 370 cm vor diesem (z-Achse). Bezüglich der x-Achse bewegte sich der Roboter im Bereich von ± 65 cm. Bei der Kamera handelt es sich um eine Sony DFW-SX900 mit einer Auflösung von 960×1280 Pixeln. Die Aufzeichnungsrate lag bei ca. 0.8 Bildern pro Sekunde. Die Partikel besitzen ein festes Bewegungsmodell mit den Mittelwerten Null und den folgenden Standardabweichungen:

x	: 4.5 cm	y	: 0.5 cm	z	: 4.5 cm
$pitch$	: $0.005\,\pi$	yaw	: $0.15\,\pi$	$roll$	: $0.005\,\pi$

Abbildung 9 zeigt alle Navigationspunkte des Roboters. Für die Strecke benötigte der Roboter zwischen Iteration 23 und 145 ca. 60 Sekunden. Die ersten 23 Iterationen und die letzten 16 stand der Roboter still. Die Experimente wurden auf einem PC mit $2.80\,GHz$ und $1\,GB$ RAM unter Linux ausgeführt. Die Ergebnisse der Versuche mit 50 und 350 Partikeln werden im folgenden beschrieben. Die Laufzeit des Algorithmus ist linear in der Anzahl der Partikel und benötigt 0.6 Sekunden pro Iteration bei 50 Partikeln und 2.73 Sekunden bei 5000 Partikeln. Im Folgenden wird mit dem Mittelwert der gewichtete Mittelwert der Partikelmenge bezeichnet. Die Standardabweichung bezeichnet die gewichtete Standardabweichung der Partikelmenge. Für kontrastreichere Darstellungen wurden in den folgenden Abbildungen die Skalen der Achsen entsprechend angepasst.

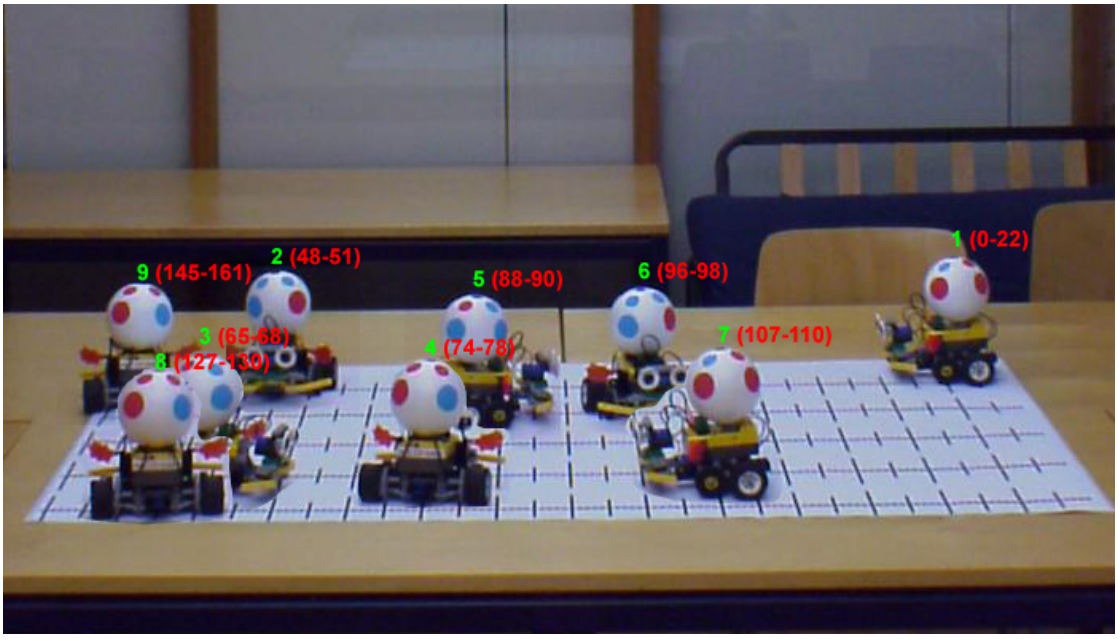

Abbildung 9: Die Wegpunkte 1 bis 9 auf dem Pfad des Roboters und die zugehörigen Iterationen 0 bis 161. Das Bild ist eine Überlagerung von Eingangsbildern. Für die Strecke benötigte der Roboter ca. 60 Sekunden. Auf dem Untergrund sind lange Striche im Abstand von je 10 cm in vertikaler Richtung angebracht. Der Abstand der horizontalen Strahlen beträgt 15 cm. Das Bewegungsfeld hat eine Größe von ca. 145 cm×85 cm.

4.1. Untersuchung der Lokalisierungsgenauigkeit

Die in Abschnitt 4 aufgezeichneten Bilder wurden für eine Verfolgung mit dem Partikelfilter-Algorithmus aus Kapitel 3.4 mit 350 Partikeln verwendet[4]. Der Algorithmus benötigte im Schnitt 0.71 Sekunden pro Iteration. Abbildung 10 zeigt die Positionen des Roboters mit eingezeichneten Partikeln bei den Iterationen 0, 2, 37, 49, 55, 65, 68, 80, 93, 100, 111 und 131. Zur Visualisierung der geschätzten Verteilung ist die Helligkeit des Farbtons rot/blau proportional zu dem Gewicht des Partikels, welches die Position repräsentiert. In Abbildung 11 sind die Rotationen und ihre Standardabweichungen aufgetragen. Die Drehungen sind deutlich am Abfall und Anstieg um ca. 1.6 rad bezüglich der Gierkurve zu erkennen. Neigungs- und Roll-Winkel bleiben wie erwartet fast konstant. Die Standardabweichung bezüglich des Gier-Winkels liegt deutlich unter 0.1 rad (ca. 5.7°). Abbildung 12 zeigt die gemessenen und die aus den Eingangsbildern approximierten x-Positionen des Markers. Die Störungen bei der Rotation (Mittelwert des Gierwinkels in Abbildung 11) besitzen zwei Ursachen: zum einen die geringe Anzahl an Partikel.

[4]Die Ausgabe dieses Experiments inklusive aller Ausgabebilder befinden sich auf der DVD im Ordner „./output/27.08.05_12:46:35“. Die Aufzeichnung befindet sich im Ordner „./output/25.08.05_17:07:54“

Bei einer Partikelanzahl von 50000 treten diese Störungen kaum auf. Zum anderen sind geringe Bewegungen entlang der x-Achse von einer Rotation entlang der Gierachse im Bild nur schwer zu erkennen. Dennoch wird eine hohe Genauigkeit bezüglich der x-Koordinate erzielt. Abbildung 13 und Abbildung 14 zeigen die Schätzung bezüglich der y- und z-Koordinate. Deutlich zu erkennen sind die Mehrdeutigkeiten in der Messung. Bewegungen entlang der z-Achse können nur schwer von Bewegungen entlang der y-Achse unterschieden werden. Bewegt sich der Marker ausschließlich zur Kamera hin oder von ihr weg, so ändert sich sein visuelles Abbild nur wenig. Aus diesem Grund treten Abweichungen der geschätzten z-Position auf, was wiederum eine fehlerhafte Schätzung der anderen Koordinaten zur Folge haben kann. Die Abweichung entlang der z-Achse betrug jedoch nie mehr als 40 cm. In Abbildung 15 und 16 ist der geschätzte Pfad und die aus den Eingangsbildern rekonstruierte Grundwahrheit als Pfad aufgetragen. Abbildung 16 zeigt zusätzlich die Standardabweichungen in z-Richtung. Die Unsicherheit in z-Richtung wirkt sich entsprechend auf den Pfad aus.

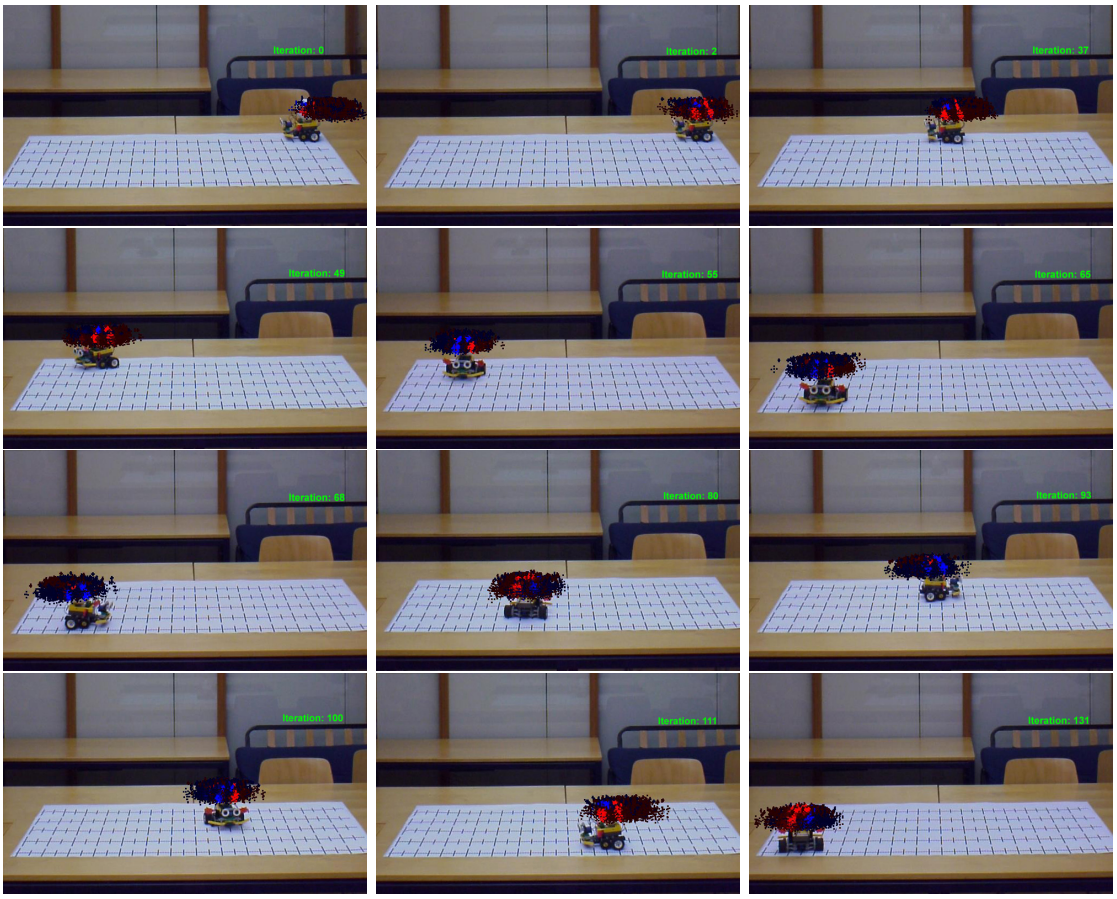

Abbildung 10: Bilder aus dem Experiment mit 350 Partikeln. Alle Partikel wurden eingezeichnet. Je heller der Farbton des Partikels, desto höher sein Gewicht.

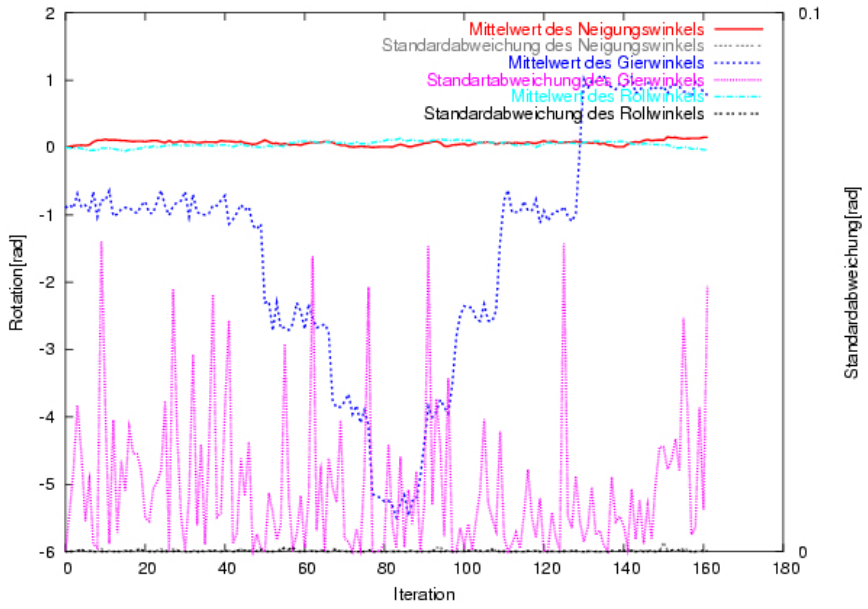

Abbildung 11: Geschätzte Rotationen bei 350 Partikeln und ihre Standardabweichungen.

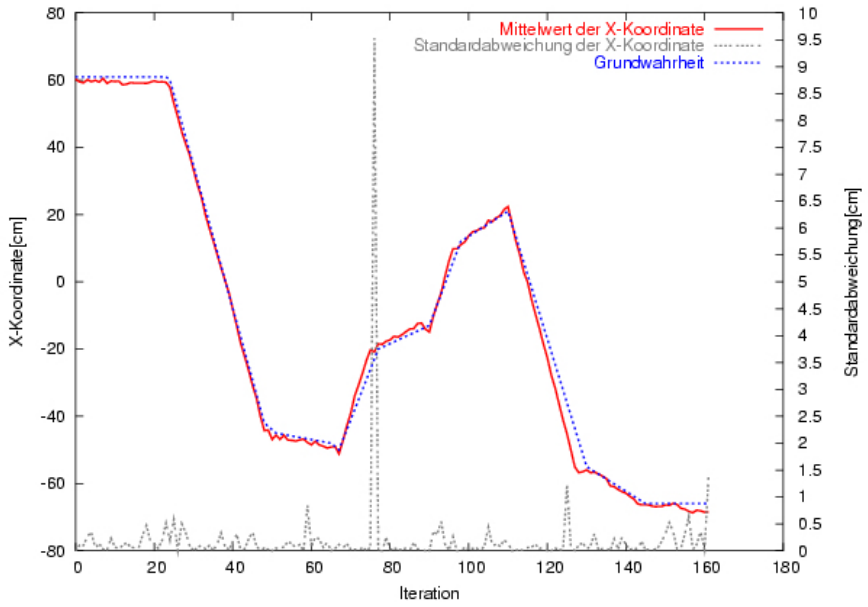

Abbildung 12: Geschätzter Weg in x-Richtung, die durch die Eingangsbilder rekonstruierte Grundwahrheit und die Standardabweichung in den Partikeln bezüglich der x-Position bei 350 Partikeln.

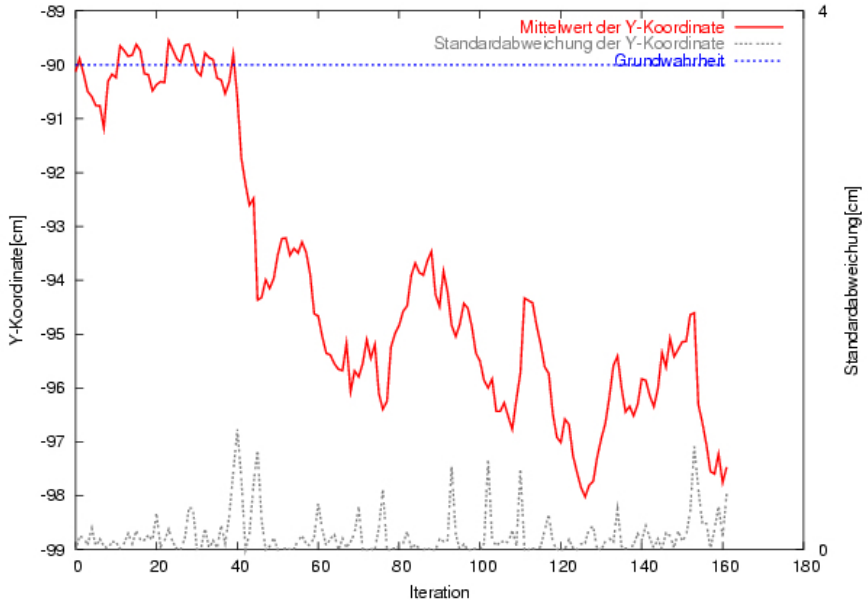

Abbildung 13: Geschätzter Weg in y-Richtung, die durch die Eingangsbilder rekonstruierte Grundwahrheit und die Standardabweichung in den Partikeln bezüglich der y-Position bei 350 Partikeln.

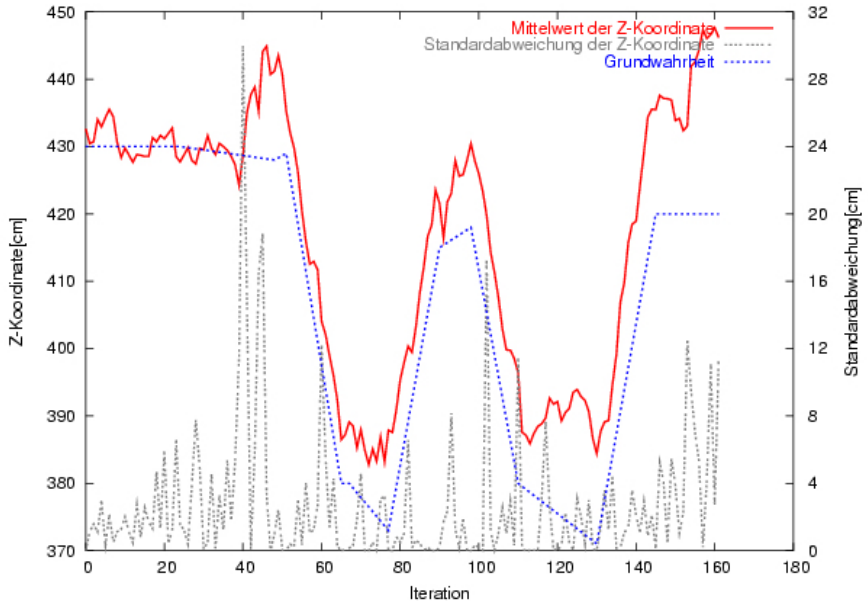

Abbildung 14: Geschätzter Weg in z-Richtung, die durch die Eingangsbilder rekonstruierte Grundwahrheit und die Standardabweichung in den Partikeln bezüglich der z-Position bei 350 Partikeln.

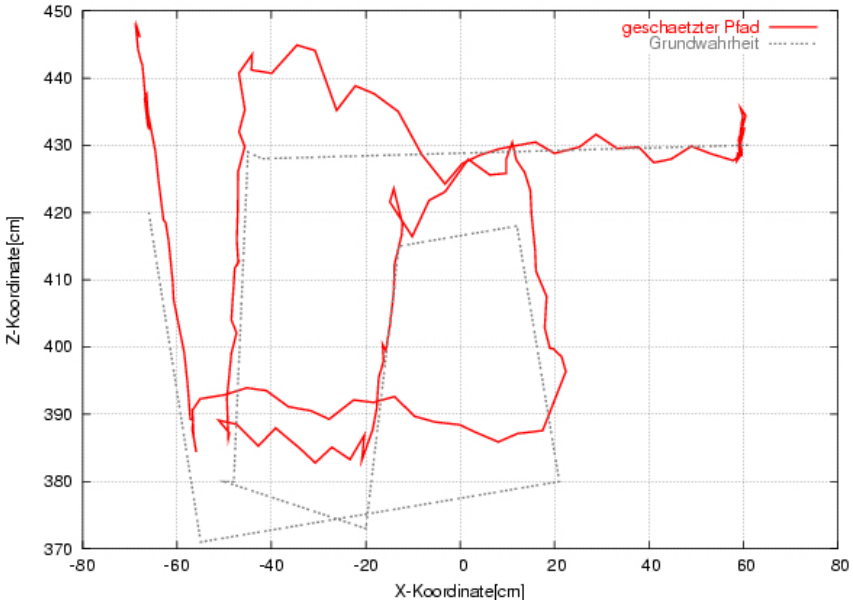

Abbildung 15: Geschätzter Pfad in x-z-Richtung und die Grundwahrheit bei 350 Partikeln.

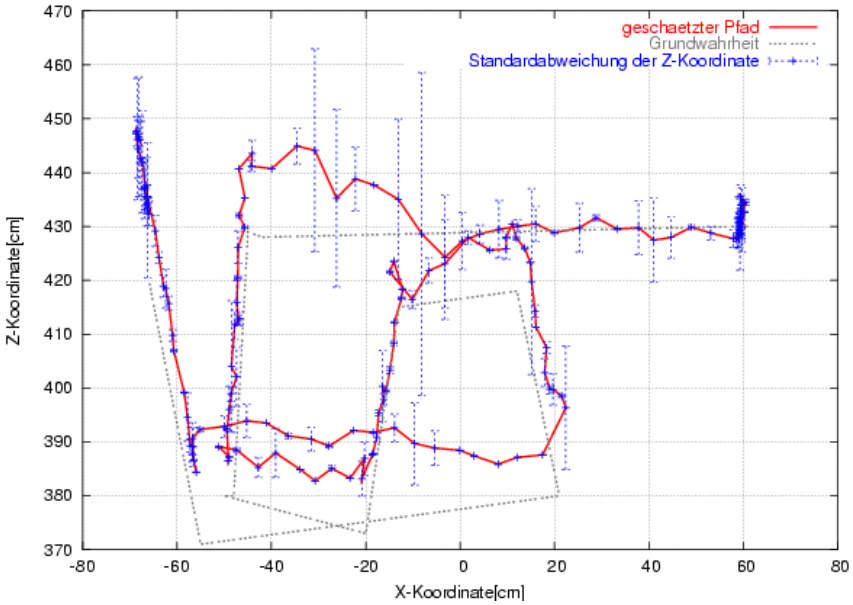

Abbildung 16: Bild aus Abbildung 15 mit eingezeichneten Standardabweichungen bezüglich der z-Koordinate.

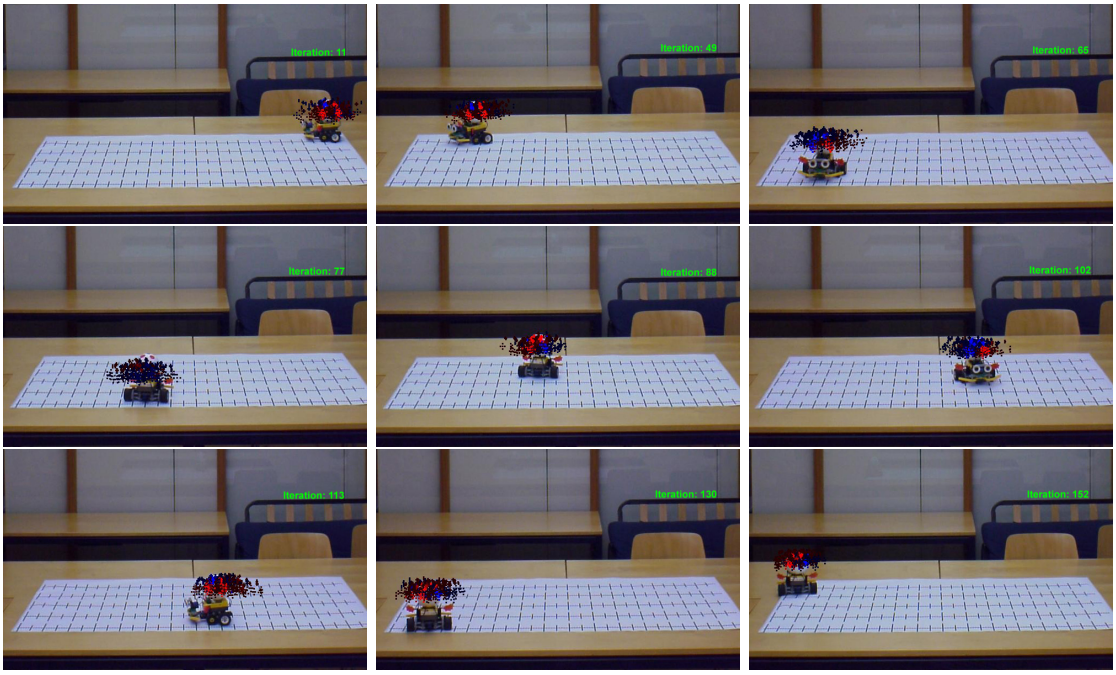

Abbildung 17: Bilder aus dem Experiment mit 50 Partikeln. Alle Partikel wurden eingezeichnet. Je heller der Farbton des Partikels, umso höher sein Gewicht.

4.2. Lokalisierungsgenauigkeit bei reduzierter Partikelanzahl

Bei einem Versuch[5] mit nur 50 Partikeln benötigt der Algorithmus im Schnitt 0.6 Sekunden pro Iteration. Im Gegensatz zum obigen Versuch wurde das Objekt nicht bei jedem Anlauf mitverfolgt. Die Rotation von ca. 90° innerhalb von zwei Iterationen konnte teils wegen der geringen Anzahl der Partikel nicht mitverfolgt werden. Abbildung 17 zeigt Ausschnitte aus der Bildsequenz. In Iteration 77 ist zu sehen, wie die Partikelmenge den Marker fast verliert. Die folgenden Bilder beschreiben das Versuchsergebnis wie in Abschnitt 4.1. Trotz der geringen Anzahl an Partikeln und der hohen Bewegung konnte die Lage und Orientierung des Roboters fast genauso gut geschätzt werden wie mit 350 Partikeln. Wie beim Versuch mit 350 Partikeln lässt sich ein Zusammenhang zwischen der Messung in y-Richtung und der Messung in z-Richtung feststellen. Bei Bewegungen entlang der z-Achse kann aufgrund der Messung nicht eindeutig eine Bewegung in z-Richtung validiert werden. Die Bewegung fällt deshalb zum Teil auf die y-Bewegung. Die geringe Standardabweichung in z-Richtung (Abbildung 21) ist durch die geringe Anzahl an Partikeln zu erklären.

[5] Die Ausgabe dieses Experiments inklusive aller Ausgabebilder befinden sich auf der DVD im Ordner „./output/27.08.05_13:30:25“

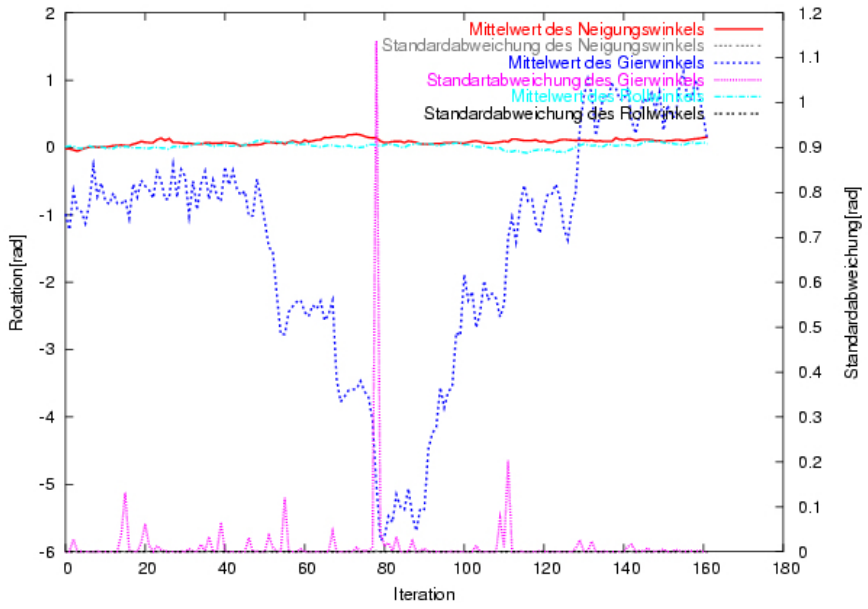

Abbildung 18: Geschätzte Rotationen bei 50 Partikeln und ihre Standardabweichungen.

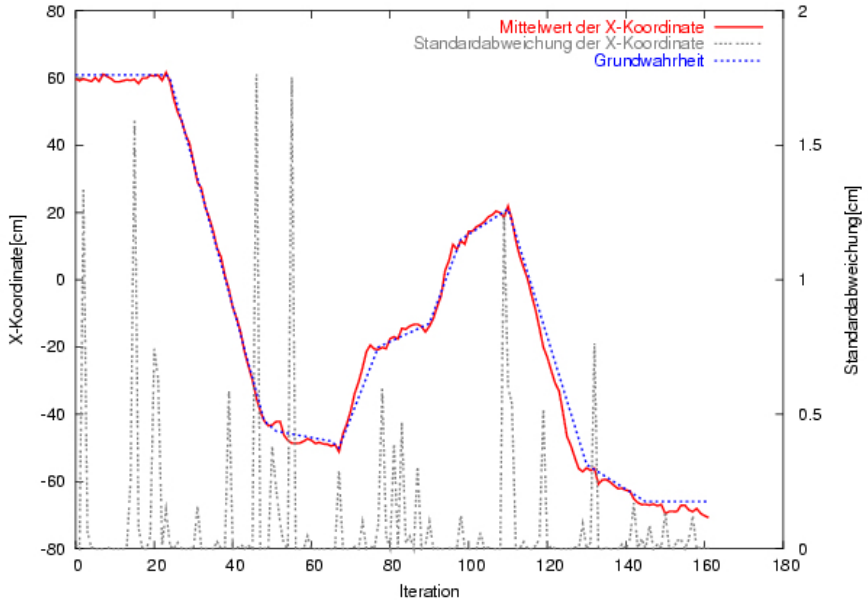

Abbildung 19: Geschätzter Weg in x-Richtung, die durch die Eingangsbilder rekonstruierte Grundwahrheit und die Standardabweichung in den Partikeln bezüglich der x-Position bei 50 Partikeln.

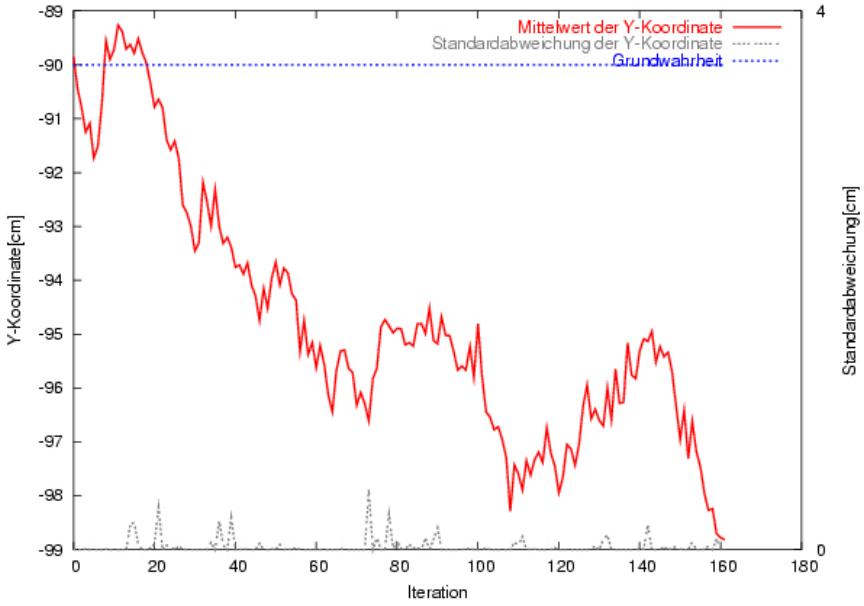

Abbildung 20: Geschätzter Weg in y-Richtung, die durch die Eingangsbilder rekonstruierte Grundwahrheit und die Standardabweichung in den Partikeln bezüglich der y-Position bei 50 Partikeln.

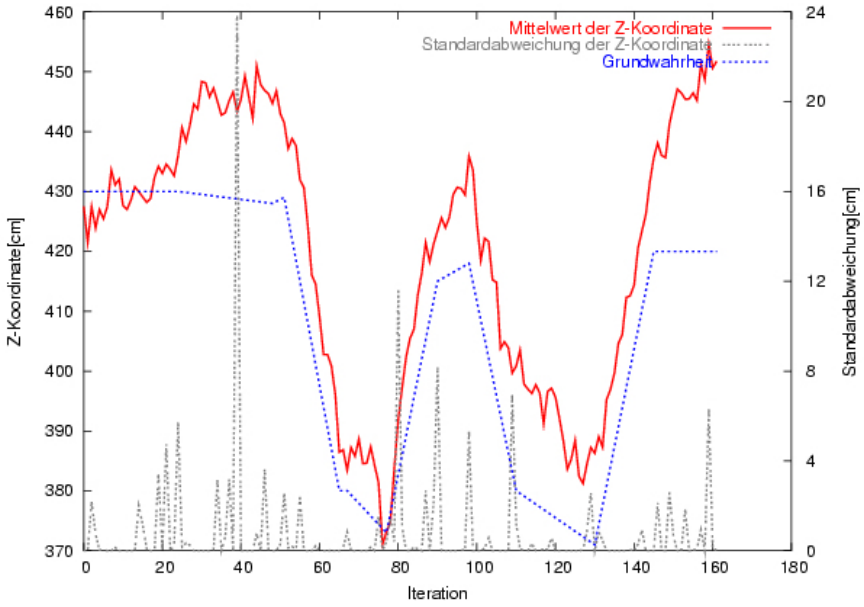

Abbildung 21: Geschätzter Weg in z-Richtung, die durch die Eingangsbilder rekonstruierte Grundwahrheit und die Standardabweichung in den Partikeln bezüglich der z-Position bei 50 Partikeln.

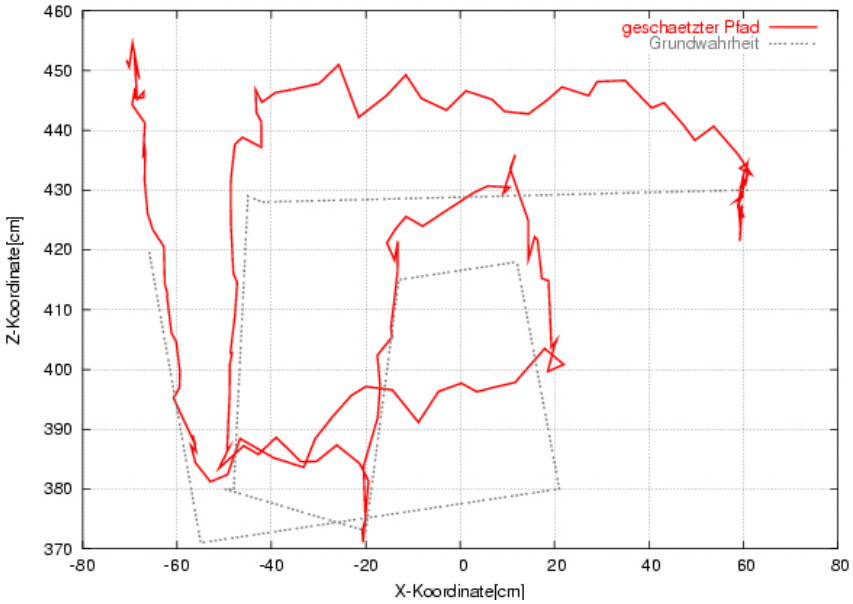

Abbildung 22: Geschätzter Pfad in x-z-Richtung und die Grundwahrheit bei 50 Partikeln.

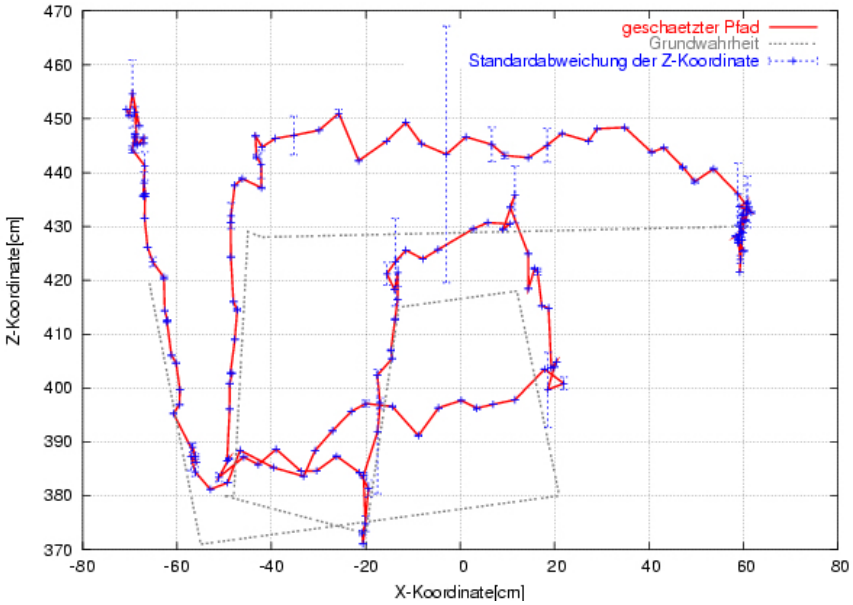

Abbildung 23: Bild aus Abbildung 22 mit eingezeichneten Standardabweichungen bezüglich der z-Koordinate.

4.3. Allgemeine Ergebnisse

Zusätzlich zu den beschriebenen quantitativen Experimenten wurde die Leistungsfähigkeit des Verfahrens anhand mehrerer Versuche im Innen- und Außenbereich (siehe Abbildung 24) untersucht. Dadurch ergaben sich folgende Erkenntnisse:

- Sofern sich Störungen in einem Gebiet auf einen Farbkanal konzentrieren und die Farbbereiche des Markers gut zu erkennen sind, wirken sich diese nur bedingt auf das Ergebnis aus. Eine menschliche Hand zum Beispiel ist solch eine Störung. Diese macht sich nur im Rotkanal bemerkbar. Der blaue Farbbereich auf dem Marker gewährleistet deutlich höhere Gewichte für Partikel, die die Position des Markers der Position der Hand vorziehen. Existieren in einer näheren Umgebung des Markers jedoch Störungen in beiden Farbkanälen, die zudem im Bild ähnlich zu einer Konstellation des Markers liegen, kann ein Teil oder gar die gesamte Partikelmenge zur Störung hingezogen werden. Der Verlust des Markers erfolgt jedoch nur dann, wenn dieser in einigen aufeinanderfolgenden Iterationen nur schlecht oder gar nicht zu erkennen war. Gerade im Außenbereich war dies teils gegeben, da sich z. B. durch Sonneneinstrahlung Glanzpunkte auf dem Marker gebildet haben.

- Die benötigte Anzahl an Partikeln ist abhängig vom verwendeten Bewegungsmodell. Im obigen Versuch waren 350 Partikel stets für eine Verfolgung des Markers ausreichend. Bei dem Versuch mit 50 Partikeln verlor das Verfahren den Marker in zwei von drei Fällen.

- Die Abweichung der Schätzung in x-Richtung betrug nie mehr als 3 cm. In y-Richtung waren die Abweichungen unter 15 cm. Die größten Abweichungen befanden sich in der Schätzung der z-Koordinate und betrugen nicht mehr als 40 cm. Die maximale Abweichung der Rotation betrug 7°.

- Das implementierte Verfahren erreicht bei 350 Partikeln eine Iterationsfrequenz von 1.4 Bildern pro Sekunde

Zu den erzielten Ergebnissen ist festzuhalten, dass das verwendete Bewegungsmodell nur eine geringe Bewegung in y-Richtung, Neigung und Rollen erlaubt. Die fest eingestellte Blende des Roboters erschwert zudem Arbeitsübergänge vom Innenbereich in den Außenbereich.

Zusammenfassend gilt für die Ergebnisse dieser ersten Untersuchungen: Die Schätzung der Position und Lage verläuft in den untersuchten Fällen zuverlässig. Je nach eingesetzter Partikelanzahl können größere Bewegungen zwischen aufeinanderfolgenden Iterationen mitverfolgt werden. Die Implementierung erlaubt eine robuste Anwendung des Verfahrens in Echtzeit bei 1.4 Bildern pro Sekunde.

Abbildung 24: Mehrere Experimente wurden sowohl im Außen- als auch im Innenbereich durchgeführt.

5. Abschlussbetrachtungen

5.1. Zusammenfassung

Ziel der vorliegenden Arbeit war die Entwicklung eines Markers und eines Systems zur Schätzung der Position und Orientierung des Markers. Dazu sollte ein System entwickelt werden, welches robust gegenüber Störungen, im Innen- und Außenbereich anwendbar und in Echtzeit ausführbar ist. Dazu wurde eine Styroporkugel mit einem Durchmesser von 10 cm mit insgesamt acht farblichen roten und blauen Markierungen nach dem Graycode-Verfahren angebracht. Partikelfilter wurden für eine szenenbasierte Lageschätzung des Markers verwendet. Aus den Aufnahmen der Kamera wurden durch verschiedene Farbfilter (hier: rot und blau) Messungen extrahiert um die Schätzung eines jeden Partikels zu gewichten. In mehreren Experimenten wurde gezeigt, dass das verwendete Verfahren eine hinreichend genaue Schätzung der Position und Orientierung des Markers liefert. Es ist dabei robust gegenüber Störungen, solange sich diese nicht in allen Farbkanälen auswirken und dabei eine mögliche Lage des Markers nahe dem tatsächlichen Aufenthaltsort vortäuschen. Bei Verwendung von 350 Partikeln ist die robuste Verfolgung in Echtzeit bei 1.4 Bildern pro Sekunde möglich.

5.2. Ausblick

Eine Unabhängigkeit von einer Initialposition würde die Anwendung des Sytems erleichtern. Beispielsweise könnte ein Fenster variabler Größe im Blau- und Rotkanal nach übereinstimmenden Bereichen suchen. Diese ließen sich dann aufgrund der variablen Größe des Fenster einschränken. An allen gefunden Positionen könnten Partikel in Gier-Richtung zu den Winkeln 45°, 135°, 225° und 315° erstellt werden (die anderen Rotationen haben aufgrund der Aufgabenstellung keinen großen Einfluss auf die Messung und können mit einem konstanten Wert initialisiert werden). Die verbleibenden Dimensionen x, y und z könnten dann mit Hilfe der Fensterposition eingeschränkt werden. Hier würde man entlang des eingeschränkten Bereiches Partikel erzeugen.

Weitere Markierungen könnten zu einer erhöhten Sicherheit bezüglich der z-Achse führen, beispielsweise drei Punkte mit unterschiedlichen Radien, die zwischen den bisherigen Punkten liegen. Dabei würden von den Punkten nur die Ränder gezeichnet, diese anschließend von Kantenfilter extrahiert und mit den geschätzten Rändern verglichen. Da nur der Rand sichtbar ist, würden sich geringe Abweichungen in z-Richtung stärker auf das Gewicht auswirken.

Aktive Markierungen, wie LEDs, könnten die Sichtbarkeit des Markers auch unter schlechter Beleuchtung gewährleisten.

Die simultane Verfolgung mehrerer Roboter bzw. Marken könnte eine weiterführende Anwendung des Verfahrens sein. Hier müssten dann Cluster von Partikeln mit jeweils eigenem Bewegungsmodell betrachtet werden. Tweed und Calway [TC02] verwenden hierzu den *Subordinated CONDENSATION*-Algorithmus.

Abbildung 25: Entwurf eines einfarbigen Markers.

Bei Betrachtung eines Bildausschnittes (engl. *region of interest* = ROI) in der Messung würde sich die Laufzeit des Algorithmus drastisch verkürzen. Die Verarbeitung des Eingangsbildes nimmt zur Zeit etwa 0.5 Sekunden in Anspruch. Bei einer Verarbeitungszeit von ca. 0.1 Sekunden wäre die Frequenz der Kamera, die bei 7.5 Bildern pro Sekunde liegt, voll ausschöpfbar.

Adaptives Anpassen des Bewegungsmodells würde zu einer erhöhten Effizienz des Algorithmus beitragen. Mit Hilfe der aktuellen Orientierung könnten insbesondere Annahmen über Bewegung in z-Richtung getroffen werden. Kenntnis über die Kontrollkommandos des kleinen Roboters würden zu präzieseren Bewegungsmodellen führen. Falls in der Anwendung die Geschwindigkeit des Markers vergleichsweise so hoch ist wie in den Experimenten, würden auch historienbedingte Bewegungsmodelle zu einer gesteigerten Genauigkeit beitragen.

Die Transformation in den HSV-Farbraum ist ein rechenintensiver Arbeitsschritt des Algorithmus. Um Rotationssymetrien zu vermeiden werden zwei Farben für die Markierungen verwendet. Eine Alternative hierzu wäre ein Marker, welcher nur Markierungen einer Farbe beinhaltet: schwarze Linien verschiedener Höhe. Ein einfacher Hell-Dunkel-Filter könnte für die Extraktion eingesetzt werden. Die Höhe der horizontalen Linien würde sich je 90° ändern und wären gemäß dem Graycode-Verfahren angebracht. Zusätzliche vertikale Linien und Blöcke, angebracht in nichtäquidistanten Abständen, würden die Genauigkeit bezüglich der Rotation und der z-Achse steigern. Abbildung 25 skizziert die Idee.

A. Mathematische Analyse des Partikelfilter-Verfahrens

Per Induktion lässt sich zeigen, dass der in Kapitel 3.2 beschriebene Partikelfilter-Algorithmus bei $M \rightarrow \infty$ gegen die a-posteriori-Verteilung der Zustandswahrscheinlichkeiten konvergiert. Dieser Abschnitt beschreibt die Kernidee hinter den Partikelfiltern nach [TBF05]. Um die Partikel mathematisch herzuleiten, betrachten wir Historien von Partikeln $x_{0:t}^{[m]}$. Der Algorithmus aus Tabelle 1 kann einfach durch Hinzufügen der Historie erweitert werden. Der Partikelfilter berechnet nun die a-posteriori-Wahrscheinlichkeitsverteilung über alle Sequenzen von Partikeln:

$$bel(x_{0:t}) = p(x_{0:t}|u_{1:t}, z_{1:t})$$

Es gilt:

$$\begin{aligned} p(x_{0:t}|u_{1:t}, z_{1:t}) &\overset{Bayes}{=} \eta\ p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t})\ p(x_{0:t}|z_{1:t-1}, u_{1:t}) \\ &\overset{Markow}{=} \eta\ p(z_t|x_t)\ p(x_{0:t}|z_{1:t-1}, u_{1:t}) \\ &= \eta\ p(z_t|x_t)\ p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t})\ p(x_{0:t-1}|z_{1:t-1}, u_{1:t}) \\ &\overset{Markow}{=} \eta\ p(z_t|x_t)\ p(x_t|x_t - 1, u_t)\ p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1}). \end{aligned}$$

Die Herleitung erfolgt nun mittels Induktion. Da $bel(x_0)$ durch Ziehen von der a-priori-Wahrscheinlichkeit $p(x_0)$ entstanden ist, ist der Induktionsanfang erfüllt.
Induktionsvoraussetzung: zum Zeitpunkt $t-1$ sind die Partikel entsprechend $bel(x_{0:t-1})$ verteilt.
Induktionsschritt: aus dem m-ten Partikel $x_{0:t-1}^{[m]}$ wird nun in Zeile 4 des Algorithmus der Folgezustand entsprechend der Verteilung

$$p(x_t|x_{t-1}, u_t)bel(x_{0:t-1}) = p(x_t|x_{t-1}, u_t)p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1}) \tag{18}$$

generiert. Mit

$$w_t^{[m]} = \frac{Zielverteilung}{generierte Verteilung}$$

folgt aus (18)

$$\begin{aligned} p(x_t|x_{t-1}, u_t)bel(x_{0:t-1}) &= \frac{\eta\ p(z_t|x_t)\ p(x_t|x_t - 1, u_t)\ p(x_{0:t-1}|z_{1:t-1}, u_{1:t})}{p(x_t|x_{t-1}, u_t)\ p(x_{0:t-1}|z_{1:t-1}, u_{0:t-1})} \\ &= \eta\ p(z_t|x_t) \end{aligned}$$

mit η als Normalisierungskonstante. Durch das Resampling und der Gewichtung mit $w_t^{[m]}$ sind die Partikel entsprechend durch

$$\eta w_t^{[m]} p(x_t|x_{t-1}, u_t)p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1}) = bel(x_{0:t}) \tag{19}$$

verteilt.

Literatur

[BI96] Andrew Blake and Michael Isard. The condensation algorithm - conditional density propagation and applications to visual tracking. In *NIPS*, pages 361–367, 1996.

[Boo64] J. Boothroyd. Acm algorithm 246: Graycode. *Commun. ACM*, 7(12):701, 1964.

[Bur03] Hans Burkhardt. *Bildverarbeitungspraktikum I, Praktikumsanleitungen*, chapter 2, pages 53–82. Lehrstuhl für Mustererkennung und Bildverarbeitung (Universität Freiburg), 2003.

[IB96] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In Bernard F. Buxton and Roberto Cipolla, editors, *ECCV (1)*, volume 1064 of *Lecture Notes in Computer Science*, pages 343–356. Springer, 1996.

[IB98] Michael Isard and Andrew Blake. A mixed-state condensation tracker with automatic model-switching. In *ICCV*, pages 107–112, 1998.

[Mit97] Tom Mitchell. *Machine Learning*, chapter 6, pages 154–199. McGraw-Hill Companies, Inc., 1997.

[RAG04a] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter, Particle Filters for Tracking Applications*, chapter 3, pages 35–66. Artech House Publishers, 2004.

[RAG04b] Branko Ristic, Sanjeev Arulampalam, and Neil Gordon. *Beyond the Kalman Filter, Particle Filters for Tracking Applications*, chapter 1,2, pages 1–34. Artech House Publishers, 2004.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*, chapter 4, pages 97–115. The MIT Press, 2005.

[TC02] David Tweed and Andrew Calway. Tracking many objects using subordinated condensation. In Paul L. Rosin and A. David Marshall, editors, *BMVC*. British Machine Vision Association, 2002.

[UM49] S. Ulam and N. Metropolis. The monte carlo method. *J. Am. Stat. Assoc.*, 1949.