Institut für Informatik
Lehrstuhl für Künstliche Intelligenz
Prof. Dr. H.-D. Burkhard

Diplomarbeit

Thema: Modellierung bewegter Objekte durch mobile Agenten

Autor: Daniel Göhring

Betreuer: Prof. Dr. Burkhard,
Jan Hoffmann

1. Dezember 2004

Humboldt-Universität zu Berlin

Selbständigkeitserklärung

Hiermit erkläre ich, Daniel Göhring, die vorliegende Diplomarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst zu haben.

Berlin, den 1. Dezember 2004

Daniel Göhring

Einverständniserklärung

Ich bin damit einverstanden, dass ein Exemplar dieser Arbeit in der Bibliothek des Instituts für Informatik der Humboldt-Universität zu Berlin ausgestellt wird.

Berlin, den 1. Dezember 2004

Daniel Göhring

Abstract. Die Umweltmodellierung auf der Grundlage von Sensordaten spielt im Rahmen von Forschungsarbeiten zur Künstlichen Intelligenz, insbesondere auf dem Gebiet der Robotik, eine wichtige Rolle. Ein Problem besteht darin, dass zu jedem Zeitpunkt nur ein Teil, bzw. ein kleiner Ausschnitt der komplexen Umwelt wahrnehmbar ist und Sensordaten nach wie vor mit Messfehlern behaftet sind. Um diese Aufgabe zu lösen, werden in dieser Arbeit einige Modellierungsverfahren präsentiert. Daran anknüpfend werden ein viel versprechender Ansatz auf Basis eines Rao-Blackwellized Partikelfilters **erläutert**. Dieser Ansatz wird innerhalb der Sony-Liga – als einer komplexen dynamischen Umgebung unter Echtzeitbedingungen – **implementiert** und **experimentell** anderen Verfahren gegenübergestellt sowie die Ergebnisse **ausgewertet**. Erweitert wurde dieser Ansatz durch die Verwendung von Negativinformationen. Dabei ermöglichte die Nutzung von Hindernisinformationen eine Abgrenzung von sichtbaren zu nicht sichtbaren Gebieten. Basierend auf dieser Verwendung von Negativinformationen ist eine Aufmerksamkeitssteuerung für die Ballsuche im RoboCup implementiert, getestet und die Ergebnisse wiederum bewertet worden.
Nicht in gleichem Maße wesentlich, aber für Fortschritte bei der Umweltmodellierung unverzichtbar, sind alle Bemühungen, die Messfehler bei Sensordaten zu minimieren, um den Gesamtprozess der Umweltmodellierung immer besser zu beherrschen. Deshalb wurde auch der Analyse von Messfehlern bei Sensordaten in dieser Arbeit die entsprechend notwendige Aufmerksamkeit gewidmet.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Modellierung der Umwelt spielt, wie in vielen Wissenschaften, auch auf dem Gebiet der mobilen Robotik eine sehr wichtige Rolle. Mehr noch, bei der langfristigen Aktionsplanung kommt ihr sogar eine Schlüsselrolle zu, wobei hervorzuheben ist, dass es unzählige Modellierungsgegenstände gibt. Ohne eine detaillierte Modellierung von Umweltzuständen ist es schwer möglich, Voraussagen über Auswirkungen von Aktionen auf zukünftige Umweltzustände zu treffen. Der RoboCup bietet aufgrund seiner hochdynamischen Eigenschaften ein besonderes Testfeld für das Entwickeln neuer Modellierungsverfahren, da Modellgebungsverfahren schnell, robust gegen verrauschte Sensordaten und zugleich effizient sein müssen. In besonderer Weise stellt das Zusammenfügen der Weltmodelle verschiedener autonomer Agenten eine Herausforderung dar, weil verschiedene Agenten – aufgrund von ungenauen Sensordaten und unterschiedlichem Vorwissen – unterschiedliche Modelle ein und derselben Situation besitzen können.

1.1 Verwandte Arbeiten

In den letzten Jahren wurde der Forschungsaufwand für Lokalisierungs- und Modellierungsaufgaben sichtbar verstärkt. Probabilistische Ansätze haben sich in diesem Zusammenhang als sehr erfolgversprechend herausgestellt. Bereits 1998 wurde eine aktive Wahrnehmung auf Basis eines Markovmodells eingesetzt, um einem Roboter auf Bürofluren die Lokalisierung zu ermöglichen, Fox [22]. Weitere Arbeiten beschreiben die Verwendung von Partikelfiltern als eine neuartige Version der Markovlokalisierung für die Roboterlokalisierung [30, 29, 16]. So beschreibt Fox z.B. die Vorteile von **M**onte-**C**arlo-**P**artikel**f**iltern (kurz MCPF) gegenüber bisherigen

analytischen Verfahren [19]. Darauf aufbauend wird in [5] die Anwendung von Partikelfiltern zur Roboterlokalisierung im RoboCup dargestellt. Um die Partikelzahl so gering wie möglich zu halten, stellt Fox in [20] eine Methode vor, wie die Partikelanzahl über KLD-Sampling[1] dynamisch an die jeweilige Situation angepasst werden kann. Neuere Arbeiten beschäftigen sich verstärkt mit Weiterentwicklungen von Monte-Carlo Verfahren. Eine neue Generation von Partikelfiltern, die sog. **R**ao-**B**lackwellized **P**artikel**f**ilter (abgekürzt RBPF), stellt den Gegenstand vieler aktueller Forschungsarbeiten dar. In diesem Sinne beschreibt Khan in [31], wie die Dimension des Zustandsraums durch Unterteilung in Unterräume, die analytisch modelliert werden, verringert werden kann. Andrieu und Doucet zeigen in ihrer Arbeit [32] die Vorteile von RBPF gegenüber anderen Verfahren wie MCPF. Arbeiten zur Verwendung von RBPF für die Modellierung und Verfolgung von mehreren Objekten gleichzeitig[2] wurden von Schulz et al. [34] in Form der Modellierung von Personen sowie in vorrangig theoretischer Weise von Särkkä bzw. Freitas et al. [33, 35] durchgeführt. Anwendungen finden sich auch in Arbeiten zur Spracherkennung [36], in denen der Filter zum Training der Spracherkennung sowie zur Dekodierung benutzt wird.

1.2 Problembeschreibung für den RoboCup

Im RoboCup sind durch die Agenten im Wesentlichen vier Aufgabenbereiche zu bewältigen:

- Verarbeitung von Sensordaten
- Modellierung der Umwelt
- Aktionsplanung
- Ausführung von Laufbewegungen und Schüssen

Im Komplex dieser Aufgaben kommen der Bestimmung der eigenen Position sowie der Position und der Geschwindigkeit des Balls elementare Bedeutung zu. In [5] wird eine effiziente Lokalisierung mit Partikelfiltern anhand der Sony Liga

[1] KLD steht für die Fehlerabschätzung durch die **K**ullback-**L**eibler **d**istance.
[2] engl.: Multiple Target Tracking

a) b)

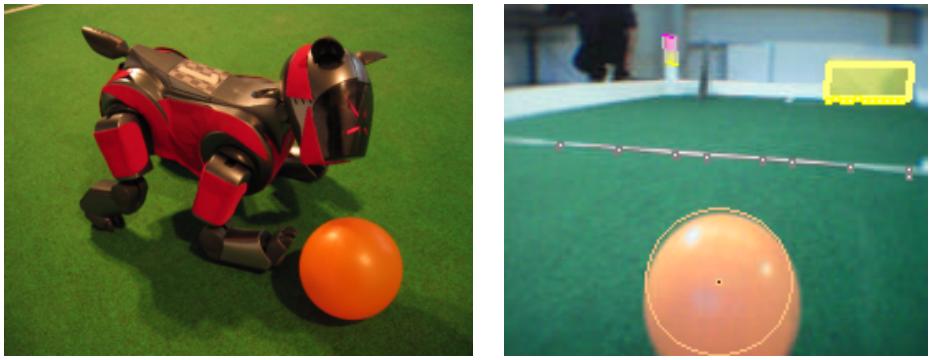


Abb. 1.1: *a) Roboter Aibo ERS-210A und Ball*
b) Kamerasicht des Roboters auf einen auf ihn zurollenden Ball

vorgestellt. Ein schwedisches Autorenkollektiv beschreibt Anwendungen von Partikelfiltern sowohl für die Navigation eines Flugzeuges als auch zur Positionsbestimmung eines Fahrzeugs auf einer Landkarte [15]. Einen Vergleich verschiedener Lokalisierungsmethoden auf Robustheit und Genauigkeit beinhaltet die Arbeit von Gutmann und Fox [18]. Die Anwendungsfelder für Lokalisierungsverfahren sind damit sehr vielfältig. Eine besondere Herausforderung besteht in dieser Hinsicht für die Sony Four Legged League, weil die Agenten der Sony-Liga – im Gegensatz zu den Robotern der Middlesize Liga – kein omnidirektionales Kamerabild besitzen. Die vierbeinigen Sony-Roboter müssen daher wie der Mensch ihre Aufmerksamkeit immer auf einen begrenzten Ausschnitt der Umwelt richten. Das ist in Anbetracht der Tatsache, dass der Ball eine sehr dynamische Größe bezüglich Ort, Geschwindigkeitsbetrag und -richtung darstellt, eine Schwierigkeit, der sich jedes Team stellen muss. Die einzelnen Teams des RoboCups, das wurde auch bei der Weltmeisterschaft in Portugal sowie den German Open 2004 deutlich, haben viele verschiedene Ansätze für die Ballmodellierung konzipiert und erprobt. Einige interessante Ansätze werden im weiteren Verlauf dieser Arbeit vorgestellt.

1.3 Zielstellung

Ziel dieser Arbeit ist die Erstellung und Evaluierung eines Modellierungsverfahrens auf Basis eines Rao-Blackwellized Partikelfilters innerhalb der Testumgebung des RoboCup. Der Roboter soll dabei in der Lage sein, Ort und Geschwindigkeit des Balls zuverlässig zu ermitteln. Zusätzlich wird das Modellierungsverfahren um die

Verwendung von Negativinformationen erweitert, wodurch der Roboter auch Informationen über ungesehene Objekte in einem Modell zusammenfassen kann. Der Roboter soll dadurch befähigt werden, seine Aufmerksamkeit auf Bereiche zu richten, in denen der Ball anhand der erhaltenen Negativinformationen vermutet wird.

1.4 Beitrag dieser Arbeit

Mit dieser Arbeit sollen Erkenntnisgewinn sowie weitere Erfahrungswerte zu folgenden Schwerpunkten erreicht werden:

1. Angelehnt an die von Fox und Kwok [7, 40] beim RoboCup-Symposium 2004 vorgestellte Objektverfolgung, wurde eine Objektmodellierung auf Basis von Rao-Blackwellized Partikeln im Rahmen der RoboCup-Domäne implementiert sowie eine Evaluierung durchgeführt.

2. Erweiterung der Objektmodellierung um die Verwendung von Negativinformationen mithilfe von Hindernisinformationen und eine darauf aufbauende Aufmerksamkeitssteuerung für die Ballsuche. Zu Evaluierungszwecken wurde ein Beispielverhalten implementiert und getestet.

3. Durchführung einer systematischen Messfehleranalyse der Ballerfassung, insbesondere bei ausgeführten Bewegungen des Roboters und Bewertung der Ergebnisse.

4. Beschreibung von Möglichkeiten der Kombination verschiedener Sensordaten zur Abschätzung der Messfehler sowie Darstellung einer möglichen Anwendung im Rahmen einer Ballmodellierung.

5. Eine systematische Gegenüberstellung von verschiedenen Filterverfahren und die Beschreibung der Ergebnisse anhand von ausgeführten Tests.

6. Aus der Sicht, dass es im Rahmen der Forschungsarbeiten des Aibo Teams Humboldt bereits ausführliche Arbeiten zu den Schwerpunkten Bildverarbeitung [4], Verhaltensmodellierung [42] sowie Bewegungsmodellierung [43] gibt, soll mit dieser Arbeit ein weiterer wichtiger Schwerpunkt zur Objektmodellierung und damit zu einer weiteren Abrundung der Gesamtforschung geleistet werden.

a) b)

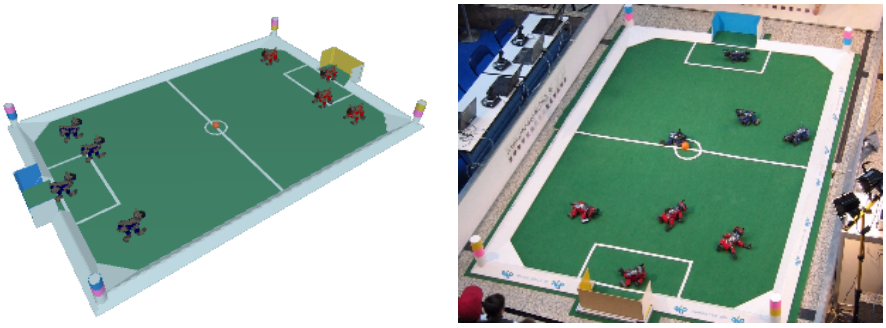


Abb. 1.2: *Das Spielfeld der Sony Four Legged League a) im Simulator, b) in der Realität*

1.5 RoboCup Domänenbeschreibung

Im Folgenden möchte der Verfasser einen kurzen Einblick in den Aufbau und die Gestaltung der Sony Four Legged League geben (Stand Juli 2004 [8]). Das Spielfeld erstreckt sich über eine 4,2 m lange sowie 2,7 m breite Fläche. Es gibt zwei Tore und entsprechende Spielfeldmarkierungen, analog zu normalen Fußballfeldern. Sowohl der Untergrund als auch alle anderen Gegenstände wie z.B. der Ball, die Tore und die Flaggen auf dem Spielfeld sind durch verschiedene Farben gekennzeichnet. Eine etwa 15 cm hohe und um 45° angewinkelte Bande verhindert, dass sowohl der Ball als auch die Roboter das Spielfeld verlassen. In jeder Ecke des Spielfeldes befinden sich farbmarkierte Zylinder, die die Lokalisierung des Roboters erleichtern sollen. Zusätzlich erfolgt die Lokalisierung über die Tore sowie die Feldlinien. Der Ball ist durch seine orange Farbe und seinen Durchmesser definiert (Abb. 1.2).

In jedem Spiel treten zwei Teams, bestehend aus jeweils vier autonomen Robotern des Typs Sony ERS-210 [2] bzw. ERS-210A oder des neuen Typs ERS-7 gegeneinander an (Abb. 1.3). Die ersten beiden Robotertypen unterscheiden sich in ihren Parametern nur geringfügig voneinander. Beim ERS-7 wurden nahezu alle Parameter (Tab. 1.1) verändert bzw. verbessert. Damit sind neue Herausforderungen und Chancen für die Entwickler gegeben.

Gespielt werden 2 Halbzeiten á 10 Minuten. Der Schiedsrichter kann Zeitstrafen für verschiedene Vergehen der Roboter verhängen, wie z.B. für zu langes Festhalten

a) b)

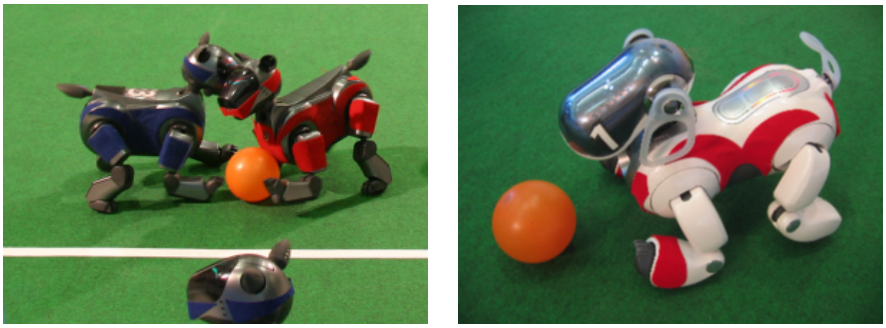


Abb. 1.3: *a) Roboter ERS-210 (blau) sowie ERS-210A (rot) b) ERS-7*

Kategorie	ERS-210/210A	ERS-7
Kameraauflösung in Pixeln	176*144	208*160
Bildwiederholfrequenz (fps)	25	30
Kameraöffnung vertikal (Grad)	48	44
Kameraöffnung horizontal (Grad)	58	55
CPU (MHz)	200/400	576
RAM (MByte)	32	64

Tab. 1.1: *Grobvergleich der Hardware von ERS-210/210A mit ERS-7*

des Balls, das Aufhalten der Verteidiger im eigenen Strafraum oder das Wegdrücken eines Spielers durch einen anderen. Die Regeln sind im Detail unter [8] nachzulesen.

Eine genaue Beschreibung verschiedener Sensoren, Problemstellungen und Lösungen in der Sony Four Legged League befindet sich im Anhang. Einen Gesamtüberblick über die Geschichte, Entwicklung und Perspektiven des RoboCup mit seinen verschiedenen Ligen gibt außerdem [1].

Kapitel 2

Grundlagen der Objektmodellierung

Wahrnehmung bezeichnet die Auswahl, Verarbeitung und Interpretation von Sensorinformationen. Sie dient als Grundlage für das Wissen eines Agenten über seine Umgebung. Auf der Basis dieses Wissens trifft der Agent Entscheidungen. Wahrnehmung lässt sich in einen aktiven und einen passiven Bereich unterteilen. Bei der passiven Wahrnehmung steht die Informationsverarbeitung im Mittelpunkt. Der Agent erhält mittels seiner Sensoren Informationen über Objekte seiner Umgebung. Die aktive Seite beschreibt die gezielte Ausrichtung der Sensoren, um Informationen über einen speziellen Bereich seiner Umgebung zu erhalten. So richtet z.B. ein Mensch, der einen Ball fängt, seine Augen gezielt auf das Objekt. Das ist in der Robotik ähnlich. Weil die meisten Agenten bzw. Roboter zu jedem Zeitpunkt nur einen kleinen Teil ihrer Umgebung wahrnehmen können, müssen sie Modelle ihrer Umwelt bilden und zusätzlich dazu ihre Wahrnehmung gezielt steuern. Erschwerend erweist sich dabei die Tatsache, dass jeder Sensor eine begrenzte Genauigkeit besitzt, weil z.B. Bilddaten eine begrenzte Auflösung besitzen, Teile des Bildes durch Artefakte verrauscht sein können, etc. pp. Einen Weg zur Überwindung dieser Einschränkung bezüglich der Sensoren stellt die Umweltmodellierung dar.
In diesem Zusammenhang hat sich in den letzten Jahren die Wahrscheinlichkeitstheorie zu einem wichtigen – wenn nicht zum wichtigsten – Werkzeug für eine robuste und exakte Modellierung von Umweltzuständen in der Robotik entwickelt. Die Idee dabei ist, das Wissen[1] des Agenten über die Weltzustände als Wahrscheinlichkeits-

[1] engl. belief

verteilung zu repräsentieren. Die Repräsentation als Wahrscheinlichkeitsverteilung erlaubt es dem Agenten bzw. Roboter, verschiedene Möglichkeiten des Umweltzustandes gleichzeitig zu verfolgen, ohne sich einer Schlussfolgerung vorzeitig anzuschließen. Wahrscheinlichkeitsverteilungen dienen auch der Aktivitätssteuerung. Der Roboter muss selbst entscheiden, welche Information für ihn zu jedem Zeitpunkt am wichtigsten ist. Diese Entscheidung ist nicht trivial und bedeutet, Kompromisse zu finden. Ein Roboter kann die Wahl haben, Informationen über seine eigene Position zu erhalten oder als Alternative, Informationen über den Ball zu bekommen. Blickt der Roboter auf den Ball, wird er oft keine neuen Daten über seine aktuelle Position erhalten. Richtet er hingegen seine Aufmerksamkeit zu lange auf Gegenstände für die Positionsbestimmung, kann er meist keine Informationen über den Ballzustand erhalten. Die Wahrscheinlichkeitstheorie kann dem Agenten helfen, „sinnvolle“ Kompromisse bei der Steuerung seiner Aufmerksamkeit einzugehen.
Im folgenden Abschnitt soll sowohl näher auf wichtige analytische Filterverfahren als auch auf Filterverfahren auf Repräsentantenbasis eingegangen werden, die in letzter Instanz alle unter dem Bayesfilter zu subsummieren sind.

2.1 Bayesfilter

Bayesfilter finden eine sehr verbreitete Anwendung in der Objektlokalisierung und -verfolgung [11, 40]. Sie eignen sich insbesondere bei Sensorungenauigkeiten sowie für die Sensorfusion, wenn verschiedene Sensordatenquellen zur Verfügung stehen. Grundlage für den Bayesfilter ist der Satz von Bayes, der besagt, dass

$$p(x|Z) = \frac{p(Z|x)P(x)}{p(Z)} \tag{2.1}$$

wobei x den Objektzustand und Z die Menge der Messungen dieses Zustandes darstellen. Bayesfilter repräsentieren den aktuellen Objektzustand zur Zeit t durch die Zufallsvariable x_t . Die Variable z_t beschreibt die Messung, die zur Zeit t durchgeführt wurde. Da der Roboter seine Umwelt aktiv beeinflussen kann, hängt jeder Zustand x auch von den durchgeführten Aktionen u ab. Zu jedem Zeitpunkt t wird der Objektzustand von x_t durch die Wahrscheinlichkeitsverteilung $Bel(x_t)$ über x_t angegeben [11]. $Bel(x_t)$ lässt sich als die Eintrittswahrscheinlichkeit von x_t unter der Bedingung der vorausgegangenen Sensorinformation $z_1, ..., z_t$ sowie der vorausgegangenen Aktionen $u_1, ..., u_t$ darstellen.

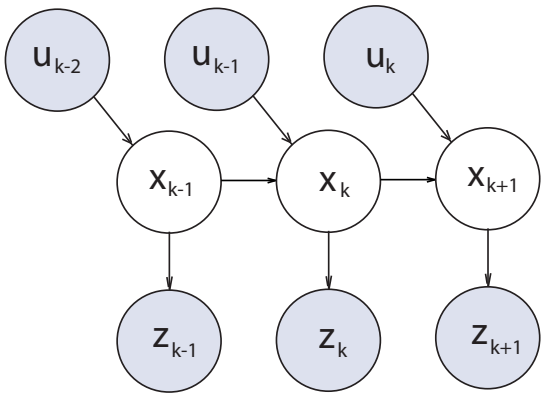


Abb. 2.1: *Diese Grafik beschreibt eine sequentielle Zustandsabschätzung, wie sie beim Bayesfilter vorkommt. Dieses Modell wird als Hidden Markov Modell bezeichnet, weil der Objektzustand x_t nicht exakt bestimmt werden kann. Der Roboter führt Aktionen u_t und Messungen z_t aus, anhand derer er sein Wissen über seinen Zustand aktualisiert.*

$$Bel(x_t) = p(x_t|z_t, u_{t-1}, z_{t-1}, u_{t-2}, ..., z_1, u_0, z_0) \tag{2.2}$$

Bayesfilter gehen von der Markovannahme aus, die besagt, dass aktuell ausgeführte Aktionen u_t sowie Messungen z_t bedingt unabhängig von vorhergegangenen Aktionen und Messungen bei gegebenem Objektzustand x_t sind (2.1). Mit dem Satz von Bayes (2.1) lässt sich Gleichung 2.2 umformen zu:

$$Bel(x_t) = \frac{p(z_t|x_t, u_{t-1}, ..., u_0, z_0)p(x_t|u_{t-1}, ..., u_0, z_0)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \tag{2.3}$$

Mit der Markovannahme (Abb. 2.1), wonach die aktuelle Messung z_t bedingt unabhängig von vorausgegangenen Messungen z und Aktionen u bei gegebenem Objektzustand x_t ist, folgt aus 2.3:

$$= \frac{p(z_t|x_t)p(x_t|u_{t-1}, ..., u_0, z_0)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \tag{2.4}$$

Mit dem Satz der totalen Wahrscheinlichkeit folgt durch Umstellung des Zählers:

$$= \frac{p(z_t|x_t)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \int p(x_t|x_{t-1}, u_{t-1}, ..., z_0)p(x_{t-1}|u_{t-2}, ..., z_0)dx_{t-1} \tag{2.5}$$

Mit Markov gelangt man zu:

$$= \frac{p(z_t|x_t)}{p(z_t|u_{t-1}, ..., u_0, z_0)} \int p(x_t|x_{t-1}, x_{u-1}) Bel(x_{t-1}) dx_{t-1} \tag{2.6}$$

Durch Einführung eines Normierungsfaktors η wird sichergestellt, dass sich die einzelnen Wahrscheinlichkeiten des Zustandsraums zu „1“ aufsummieren:

$$= \eta p(z_t|x_t) \int p(x_t|x_{t-1}, x_{u-1}) Bel(x_{t-1}) dx_{t-1} \tag{2.7}$$

Die resultierende Gleichung 2.7 ist eine rekursive Form des Bayesfilters, die es ermöglicht, eine inkrementelle Berechnung der neuen Zustandsfunktion aus der direkt vorhergehenden Zustandsfunktion durchzuführen. Diese Gleichung lässt sich in zwei Teilgleichungen zerlegen, je nachdem, ob Daten über ausgeführte Aktionen oder Sensordaten eintreffen. Falls Daten über ausgeführte Aktionen u_{t-1} eintreffen, wird der neue Zustand zunächst vorhergesagt[2]:

$$Bel^-(x_t) \longleftarrow \int p(x_t|x_{t-1}u_{t-1}) Bel(x_{t-1}) dx_{t-1} \tag{2.8}$$

Treffen Sensordaten z_t ein, wird die Zustandsabschätzung aktualisiert[3]:

$$Bel(x_t) \longleftarrow \eta p(z_t|x_t) Bel^-(x_t) \tag{2.9}$$

Um die Berechnung zu beginnen, wird $Bel(x_0)$ mit dem Vorwissen über Zustand x_0 initialisiert. Falls Nichts über x_0 bekannt ist, wird eine Gleichverteilung über alle möglichen Zustände angenommen. Es sind weiterhin das Wahrnehmungsmodell $p(z_t|x_t)$, die Systemdynamik $p(x_t|x_{t-1}, u_{t-1})$ sowie die Repräsentationsform von $Bel(x_t)$ festzulegen. Das Wahrnehmungsmodell beschreibt dabei die Wahrscheinlichkeit, eine Messung z_t bei gegebenem Zustand x_t zu erhalten. Die Systemdynamik, oft auch als Prozessmodell bezeichnet, beschreibt die Veränderung des Zustandes x_t bei ausgeführten Aktionen u_{t-1} im Zustand x_{t-1}. Es ist anzumerken, dass die Zufallsvariablen u, x und z mehrdimensionale Vektoren sein können. Bayesfilter stellen damit nur ein abstraktes Konzept für die Objektmodellierung dar, bei dem das Wahrnehmungsmodell sowie die Systemdynamik für eine spezielle Anwendung erst noch festgelegt werden müssen. In den nächsten Abschnitten werden gebräuchliche Anwendungen des Bayesfilters vorgestellt.

[2] engl. predicted
[3] engl. updated

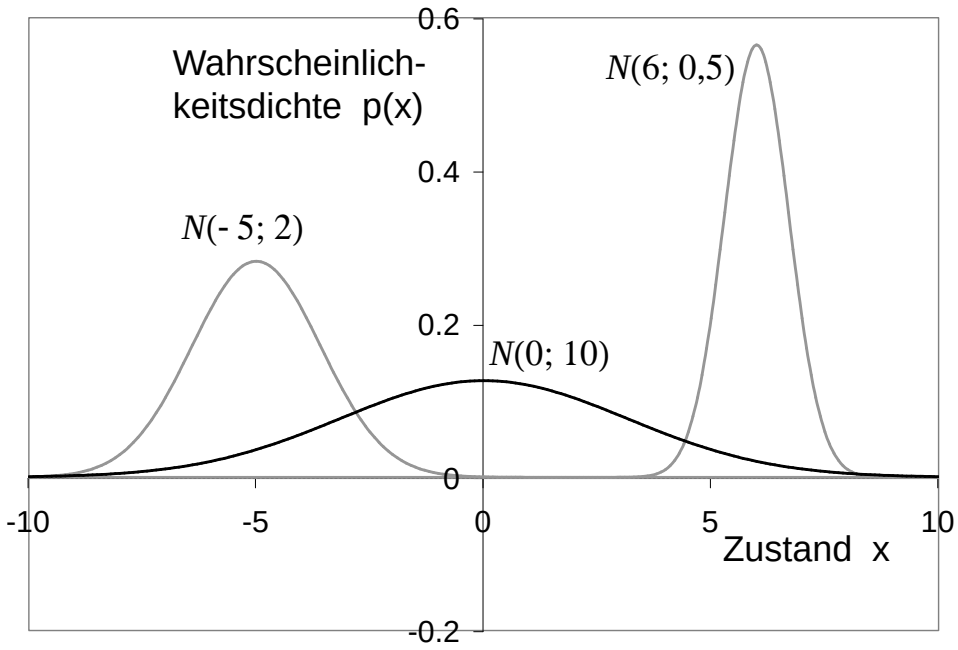


Abb. 2.2: *Drei Gaußfunktionen mit verschiedenen Parametern, je nach Varianz breiter oder spitzer. Der erste Parameter des Zweitupels gibt den Mittelwert, der zweite die Varianz an.*

2.2 Die Gaußverteilung

Bei mobilen Robotern kommen als Zustandsvariablen häufig die eigenen Positionen bzw. die Positionen von Objekten ihrer Umgebung in Frage. Wenn es darum geht, ein Objekt zu verfolgen und zu modellieren, werden häufig unimodale Verteilungsfunktionen verwendet, d.h., es wird genau eine Hypothese verfolgt, die genau eine mögliche Objektposition repräsentiert. Zu diesem Zweck werden des Öfteren Gaußfunktionen[4] verwendet, da sie die meisten Unimodalverteilungen geeignet zum Ausdruck bringen. Außerdem besitzen Gaußfunktionen günstige mathematische Eigenschaften, wie z.B. die Abgeschlossenheit bezüglich der Addition und Multiplikation. Daher sind Algorithmen, die auf Basis der Gaußfunktionen arbeiten, meist sehr effizient.

Eine Gaußverteilung $\mathcal{N}(\mu, \Sigma)$ repräsentiert die Wahrscheinlichkeitsdichte durch die ersten und zweiten Momente einer Verteilung. Bei einem Spaltenvektor X ist das erste Moment μ der Erwartungswert $E[X]$ und das zweite Moment Σ die Kova-

[4] auch Normalverteilung genannt

rianzmatrix $E[(X - E[X])(X - E[X])^T]$. Die Dichtefunktion einer n-dimensionalen Variable X berechnet sich bei einer Normalverteilung $\mathcal{N}(\mu, \Sigma)$ durch:

$$P(X) = \frac{1}{\sqrt{(2\pi)^n * det(\Sigma)}} exp^{-\frac{1}{2}(X-\mu)^T(\Sigma)^{-1}(X-\mu)} \tag{2.10}$$

Die Schreibweise $\mathcal{N}(X; \mu, \Sigma)$ wird verwendet, um die aus Gleichung 2.10 berechnete Wahrscheinlichkeit zu repräsentieren. Der Erwartungswert μ stellt den wahrscheinlichsten Wert der Verteilung dar. In der Anwendung bestimmt er den Ort des zu modellierenden Objektes. Die Kovarianzmatrix gibt Auskunft darüber wie verstreut oder eng zusammenliegend die gesamte Verteilung ist (Abb. 2.2), und ob es Zusammhänge zwischen einzelnen Dimensionen des Zustandsvektors gibt. Die visualisierten Kovarianzen einer bivariaten Verteilung bilden in der Ebene eine Ellipse, die Kovarianzen einer multivariaten Verteilung bilden einen Hyperellipsoid.

2.3 Kalmanfilter

Kalmanfilter [12] sind die wohl verbreitetste Form der Bayesfilter. Für das Sensor- und Prozessmodell verwenden sie Gaußfunktionen und sind daher sehr einfach und effizient zu implementieren. Sie sind optimal für zeitdiskrete lineare dynamische Prozesse, d.h. sie minimieren den quadratischen Fehler der Abschätzungen. Diese Eigenschaft geht allerdings verloren, wenn sie für nichtlineare Prozesse verwendet werden. Daher befassen sich notwendigerweise einige Arbeiten mit der Erweiterung von Kalmanfiltern für nichtlineare Prozesse [13]. Ein linearer Prozess für Gaußfunktionen wird wie folgt definiert:

Sei $x_t \in \Re^d$ der Zustand, der zum Zeitpunkt t modelliert werden soll, $\hat{x}_t$ die Abschätzung von x_t und P_t die Kovarianzmatrix, die den a-posteriori Modellierungsfehler abschätzt.

Das Kalmanmodell geht davon aus, dass sich ein Zustand x_t aus dem vorhergehenden Zustand x_{t-1} und den ausgeführten Aktionen u_{t-1} wie folgt linear berechnet:

$$x_t = Ax_{t-1} + Bu_{t-1} + \omega_{t-1} \tag{2.11}$$

$$\omega_{t-1} \sim \mathcal{N}(0, Q) \tag{2.12}$$

Die Matrix A setzt den alten Zustand x_{t-1} mit dem neuen Zustand x_t zur Zeit t in Beziehung. Die Matrix B lässt die seit dem letzten Zustand ausgeführten Aktionen

u_{t-1} in den neuen Zustand einfließen. Das dabei auftretende Prozessrauschen ω sei weiß, also normalverteilt, mit dem Mittelwert null und der Kovarianz Q.

Die Messung z_t ist ebenfalls linear vom Zustand x_t abhängig:

$$z_t = Hx_t + v_t \tag{2.13}$$

$$v_t \sim \mathcal{N}(0, R) \tag{2.14}$$

Die Matrix H transformiert den Zustand x_t in eine Messung z_t, die dem Sensorrauschen v unterliegt. Das Sensorrauschen v ist ebenfalls normalverteilt mit dem Mittelwert null und der Kovarianz R.

Im ersten der beiden Schritte „Vorausberechnung“ und „Aktualisierung“ wird aus dem letzten abgeschätzten Zustand $\hat{x}_{t-1}$ unter Berücksichtigung der seitdem vom Roboter ausgeführten Aktionen u_{t-1} der neue Zustand $\hat{x}_t^-$ ohne Nutzung von Sensordaten zunächst abgeschätzt.

Sei $\hat{x}_t^-$ der Erwartungswert von x_t vor dem Erhalt der Sensordaten z_t. Dann berechnet sich $\hat{x}_t^-$ aus $\hat{x}_{t-1}$ und u_{t-1} mit Gleichung 2.11 wie folgt:

$$\hat{x}_t^- \triangleq E[x_t|u_{t-1}, ..., z_0] \tag{2.15}$$

$$= E[Ax_{t-1} + Bu_{t-1} + w_{t-1}|u_{t-1}, ..., z_0] \tag{2.16}$$

$$= A\hat{x}_{t-1} + Bu_{t-1} \tag{2.17}$$

Der Umformungsschritt zu Gleichung 2.17 berücksichtigt, dass der Erwartungswert von ω, dem Prozessrauschen, null ist.

Die a-priori[5] Kovarianzen, repräsentiert durch Matrix P^-, berechnen sich dementsprechend:

$$P_t^- \triangleq E[(x_t - E[x_t])(x_t - E[x_t])^T|u_{t-1}, ..., z_0] \tag{2.18}$$

$$= AE[(x_t - E[x_t])(x_t - E[x_t])^T]A^T + E[v_t v_t^T] \tag{2.19}$$

$$= AP_{t-1}A^T + Q \tag{2.20}$$

Der Korrekturschritt ist definiert als Produkt aus $p(z_t|x_t)$ und $Bel^-(x_t)$. Die erwartete Beobachtung $\hat{z}_t$ zum Zeitpunkt t berechnet sich analog zu Gleichung 2.13 wie folgt:

[5] a-priori bedeutet in diesem Zusammenhang: „vor der Messung“, a-posteriori dementsprechend: „nach der Messung“

$$\hat{z}_t = H\hat{x}_t^- \tag{2.21}$$

Der Unterschied zwischen der erwarteten Messung $\hat{z}_t$ und der tatsächlichen Messung z_t wird als Innovation[6] bezeichnet und gibt Auskunft darüber, wie stark die Vorhersage des Zustandes von der Messung abweicht. Seine Kovarianz S_t berechnet sich aus der Sensorkovarianz R_t und der a-priori Zustandskovarianz P_t^- :

$$S_t = (R + HP_t^- H^T) \tag{2.22}$$

Daraus berechnet sich das Kalmangain[7], welches bestimmt, wie stark die Innovation, d.h. die Differenz zwischen vermuteten und tatsächlichen Sensordaten für die Abschätzung des neuen Zustands $\hat{x}_t$, berücksichtigt werden soll.

$$K_t = P_t^- H^T S_t^{-1} \tag{2.23}$$

Nun kann die a-posteriori Abschätzung $\hat{x}_t$ für den Zustand x_t sowie die Berechnung der a-posteriori Kovarianzen P_t erfolgen:

$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - \hat{z}_t) \tag{2.24}$$

$$P_t = P_t^- - K_t S_t K_t^T \tag{2.25}$$

Wenn R, die Kovarianz der Sensordaten z.B. groß ist, hat die Vorhersage einen größeren Einfluss auf die neue Abschätzung als die Aktualisierung. Die Kovarianz der Innovation wird dann eher größer sein und sich in einem kleineren Kalmangain auswirken. Dadurch werden Abweichungen der Messwerte von den vorausgesagten Messwerten nicht so stark berücksichtigt. Ein solcher Kalmanfilter ist insbesondere für die Filterung von stark verrauschten Sensordaten geeignet. Ist R hingegen klein, erfolgt die Korrektur des a-priori Zustands $\hat{x}_t^-$ in einem höheren Maß um die Messung z_t. Kalmanfilter mit kleinen Kovarianzen der Sensordaten R reagieren schneller auf zeitliche Änderungen der Sensordaten.

Für die Filterung müssen sowohl die Prozesskovarianzmatrix Q als auch die Sensorkovarianzmatrix R bekannt sein. Nicht immer sind diese beiden Matrizen für jeden Zeitpunkt t konstant. Die Sensormatrix R kann durch Auswertung von Sensordaten erstellt werden. Die Prozessmatrix Q repräsentiert den Fehler, der bei Propagierung

[6] auch „Residual“

[7] engl. „gain“ bedeutet „Zuwachs“, „Ertrag“

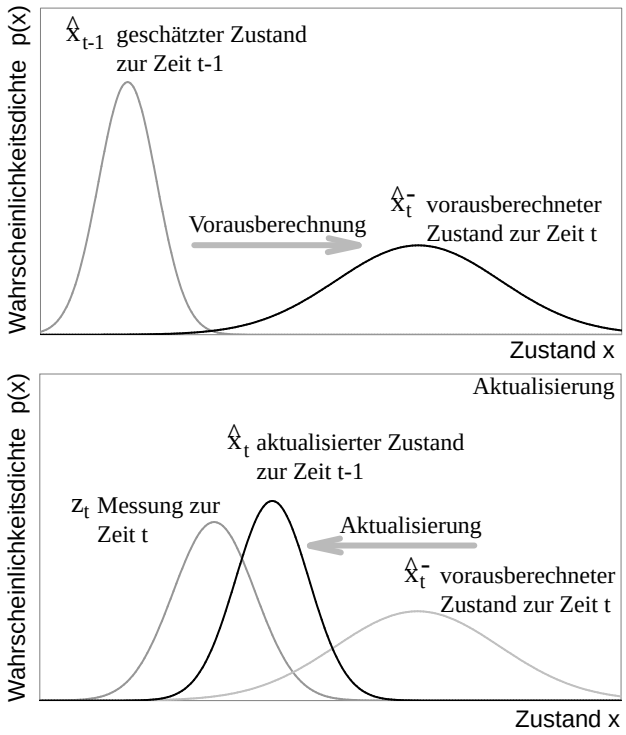


Abb. 2.3: *Die beiden Schritte Vorausberechnung und Aktualisierung am Beispiel eines eindimensionalen Kalmanfilters. Bei der Vorausberechnung wird die Varianz des Objektzustandes größer. Die Aktualisierung verringert die Varianz.*

eines Zustandes $\hat{x}_{t-1}$ in den nachfolgenden Zustand $\hat{x}_t^-$ entsteht. Sie kann nicht vom Roboter selbst ermittelt werden, da seine Sensordaten schon dem Sensorfehler unterliegen. Die Ermittlung der Prozessmatrix auf Datenbasis erfordert daher einen externen Beobachter. So erweist sich eine Deckenkamera beispielsweise für den RoboCup als hilfreich, um abschätzen zu können, inwieweit das Prozessmodell die reale Ballposition widerspiegelt.

Durch Anwendung des Kalmanfilters im Balllokator des German Teams im Jahr 2004 konnten die Ballgeschwindigkeiten erstmals sehr akkurat berechnet werden. In Kombination mit einem Torwart, der unnötige Laufbewegungen weitestgehend vermied, wurden damit vor allem in der Torverteidigung bemerkenswerte Fortschritte erzielt. Spieler und Torwart erkannten auf sie zurollende Bälle innerhalb kürzester Zeit und leiteten Gegenmaßnahmen ein. Nicht zuletzt deshalb gelang dem German Team der Gewinn der Weltmeisterschaft in Lissabon in der Sony Four Legged League.

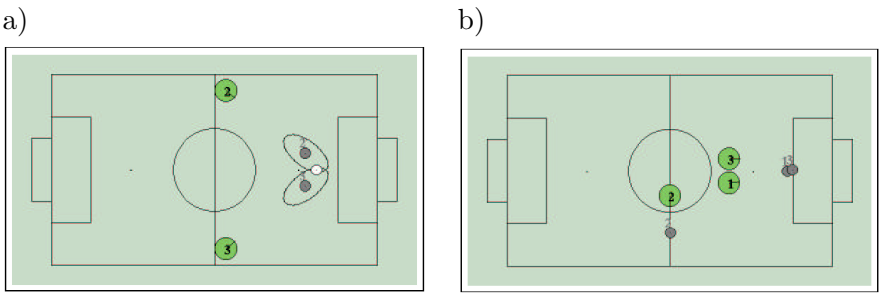


Abb. 2.4: *a) die Ballpositionen zweier Roboter werden fusioniert b) Roboter 2 hat eine falsche Ballposition, der Kalmanfilter kann diese jedoch herausfiltern, da Roboter 1 und 3 eine ähnliche Position ermitteln, Quelle: Dietl-Gutmann [14]*

Der Kalmanfilter kann auch dazu verwendet werden, Sensordaten aus verschiedenen Quellen zu modellieren. Dietl und Gutmann zeigen in ihrem Paper zum Thema der Multiagentenmodellierung [14] eine Möglichkeit, die Sensorwerte verschiedener Roboter zu fusionieren. Sie verwendeten dafür einen Kalmanfilter, der die Sensorwerte der verschiedenen Roboter als Eingangsdaten erhält. Das funktioniert mit drei Robotern auch dann noch, wenn einer der drei Roboter eine falsche Ballposition ermittelt. Die falsche Ballposition wird herausgefiltert, wenn die Ballpositionen der beiden anderen Roboter weitgehend übereinstimmen. In Abb. 2.4 wird der soeben beschriebene Ansatz verdeutlicht.

Diese Art Modellierung wurde auch auf die Sony Four Legged League übertragen. Die Roboter modellieren gemeinsam einen Ball aus den Informationen aller Roboter und des Weiteren einen Ball aus den eigenen Sensorinformationen. Jedoch ist bei kommunizierten Ballpositionen zu berücksichtigen, dass zusätzlich zum Fehler bei der Berechnung eines Ballperzepts der Selbstlokalisierungsfehler zweifach auftritt, da die kommunizierenden Roboter ihre eigenen Positionen in die Berechnung der allozentrischen[8] Ballposition mit einfließen lassen. Dadurch wird es unwahrscheinlich, dass zwei Ballpositionen konvergieren. Zu stark verrauschte Sensordaten, das zeigten die Experimente, können auch von einem Kalmanfilter nicht mehr verwendet werden. Es kann sogar vorkommen, dass im Ergebnis einer Filterung der modellierte Ball an Positionen entsteht, an denen er von keinem Roboter gesehen wurde (Abb. 2.5 b). Daher wird dieses Verfahren meist mit zusätzlichen Einschränkungen

[8]Das allozentrische Koordinatensystem ist ein globales, von der Roboterposition unabhängiges Koordinatensystem, mit Ursprung in der Mitte des Spielfeldes.

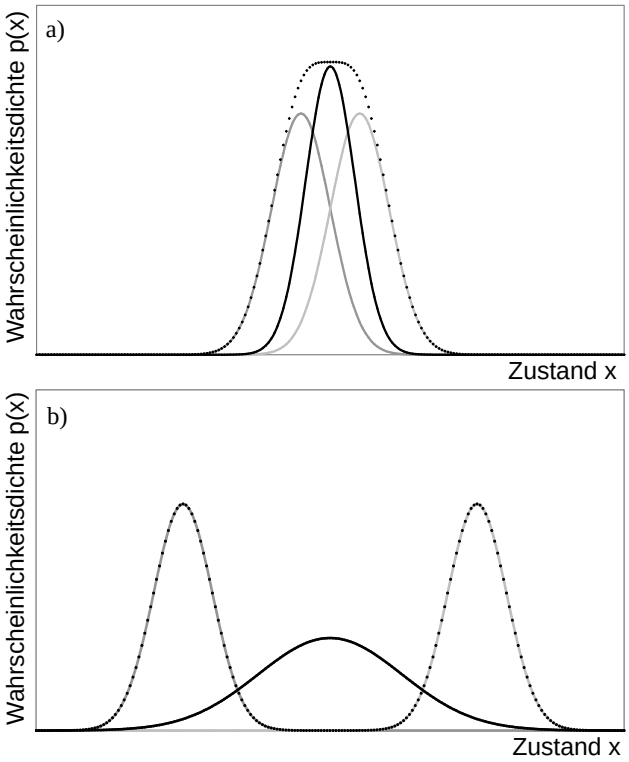


Abb. 2.5: *a) erfolgreiche Kombination zweier Gaußfunktionen (grau), die resultierende Funktion (schwarz) steht in guter Näherung zur Summe (gepunktet) der beiden Quellfunktionen; b) bei weit voneinander entfernten Gaußkurven (grau) besitzt die resultierende Funktion (schwarz) ihr Maximum in einem Bereich, in dem die Quellfunktionen nahezu null sind und steht damit in einer schlechten Näherung zur Summe (gepunktet) der Quellfunktionen*

verwendet, z.B. mit der Bedingung, dass die Sensordaten der einzelnen Roboter nicht zu stark voneinander abweichen dürfen.

2.4 Multihypothesis Tracking

Multi**h**ypothesis **T**racking[9] (kurz: MHT) überwindet die Schwäche von Kalmanfiltern, nur Unimodalverteilungen repräsentieren zu können [11]. Die Dichtefunktion des Objektzustandes *Bel* zur Zeit t wird als Summe von Gaußfunktionen repräsentiert:

[9] dt.: Multihypothesenverfolgung

$$Bel(x_t) = \sum_i w_t^{(i)} \mathcal{N}(x_t; \mu_t^{(i)}, \Sigma_t^{(i)}) \tag{2.26}$$

Jede Gaußfunktion wird durch einen Kalmanfilters aktualisiert. Aus der Höhe der Übereinstimmung[10] einer jeden Gaußfunktion $\mathcal{N}(\mu_t^{(i)}, \Sigma_t^{(i)})$ mit den Sensordaten berechnet sich ihre Gewichtung $w_t^{(i)}$. Zu jedem Zeitpunkt t wird also für jede Hypothese i eine Gewichtung ermittelt, die umso höher ausfällt, je besser die Hypothese in Einklang mit den Sensordaten steht. Durch ihre Fähigkeit, multimodale Verteilungen repräsentieren zu können, sind MHT-Algorithmen in einem breiteren Spektrum verwendbar als Kalmanfilter. Insbesondere bei Mehrdeutigkeiten, z.B. bei mehreren möglichen Positionen eines Objektes ist es wünschenswert, die verschiedenen Hypothesen mitzuführen. Diese Fähigkeit erfordert jedoch eine höhere Rechenleistung des Roboters. Es ist außerdem festzulegen, in welchen Fällen Hypothesen zu löschen bzw. neu zu generieren sind. Weil jede Hypothese auf Basis eines Kalmanfilters aktualisiert wird, gelten auch hier die Linearitätsannahmen.

2.5 Rasterbasierte Techniken

Bei rasterbasierten Techniken wird der Hypothesenraum[11] durch eine Rasterung[12] in endlich viele Unterabschnitte aufgeteilt. Dadurch kann ein kontinuierlicher Hypothesenraum in einen diskreten überführt werden. Durch rasterbasierte Ansätze können außerdem nahezu beliebige Verteilungen repräsentiert werden. Das Problem besteht allerdings darin, dass der Rechen- und Speicheraufwand für die Aktualisierung des Rasters exponentiell mit der Dimension des Hypothesenraums ansteigt. Das kann durch eine sehr geringe Auflösung des Rasters kompensiert werden [30]. Um diesen Nachteil zu umgehen, werden rasterbasierte Techniken hauptsächlich für niederdimensionale Probleme angewandt.

2.6 Monte-Carlo Partikelfilter

Monte-**C**arlo **P**artikel**f**ilter (abgekürzt MCPF) stellen eine besondere Alternative zu den bisher beschriebenen Modellierungsverfahren dar. Sie sind im Gegensatz zu Kalmanfiltern nicht auf lineare Prozesse beschränkt und können deshalb beliebige

[10] engl. likelihood
[11] die Potenzmenge aller möglichen Hypothesen
[12] engl. grid

multimodale Verteilungen repräsentieren. Weiterhin sind sie recheneffizienter als rasterbasierte Verfahren, weil sie ihr Auflösungsvermögen auf „wichtige“ Stellen konzentrieren können [30]. Dennoch sind bei MCPF die Objektzustände nicht diskretisiert. Deshalb kann mit MCPF eine nahezu beliebig hohe Auflösung der Wahrscheinlichkeitsdichtefunktion gewährleistet werden, im Gegensatz zu rasterbasierten Ansätzen. MCPF verwenden Repräsentanten[13], um die Wahrscheinlichkeitsdichteverteilung widerzuspiegeln. Durch die Verwendung von diskreten Repräsentanten arbeiten MCPF allerdings nicht so exakt bei linearen Prozessen wie Kalmanfilter. Sie sind dafür aber weitaus robuster bei nichtlinearen Prozessen. Diese Repräsentanten werden analog zum Bayesfilter rekursiv berechnet. Der Objektzustand berechnet sich aus der Menge der Repräsentanten. Sei Z^t die Menge der letzten Messungen $Z^t \triangleq \{Z_1 ... Z_t\}$ zur Zeit t und $P(X_t|Z^t)$ die a-posteriori Wahrscheinlichkeit. Wegen Bayes gilt wie in Gleichung 2.8:

$$P(X|Z_t) \propto P(Z_t|X_t) \int_{X_{t-1}} P(X_t|X_{t-1} P(X_{t-1}|Z^{t-1}) \tag{2.27}$$

Die Monte-Carlo Abschätzung der Dichtefunktion für X_t berechnet sich bei einer Menge von N Repräsentanten $\{X_{t-1}^{(i)}, \pi_{t-1}^{(i)}\}_{i=1}^{N}$ wie folgt [31]:

$$P(X_t|Z^t) \approx kP(X_t|Z_t) \sum_{(i)} \pi_{t-1}^{(i)} P(X_t|X_{t-1}^{(i)}) \tag{2.28}$$

wobei $\pi_{t-1}^{(i)}$ die Gewichtung für den Partikel $X_{t-1}^{(i)}$ darstellt und k als Normierungsfaktor dient. MCPF werden oft als „importance sampler“ bezeichnet. Damit wird beschrieben, dass an den Stellen, an denen die Dichtefunktion besonders hoch ist, in jedem Schritt überdurchschnittlich viele neue Repräsentanten generiert werden. Die „wichtigen“ Stellen in der Dichtefunktion, bspw. für den Aufenthaltsort eines Objektes, werden also durch mehr Partikel repräsentiert als unwichtige Stellen. Dadurch werden die Auflösung und damit die Genauigkeit für die Abschätzung erhöht. Die N neuen Repräsentanten $X_t^{(j)}$ berechnen sich[14] aus der abgeschätzten Verteilungsfunktion q:

$$X_t^{(j)} \sim q(X_t) \triangleq \sum_{(i)} \pi_{t-1}^{(i)} P(X_t|X_{t-1}^{(i)}) \tag{2.29}$$

[13] engl. samples

[14] Die Erzeugung einer neuen Repräsentantenmenge aus einer alten wird im Englischen als „Resampling“ bezeichnet.

und werden mit $\pi_t^{(j)}$ gewichtet. $\pi_t^{(j)}$ berechnet sich aus der Wahrscheinlichkeit für eine Messung Z_t unter der Annahme eines vom gegebenen Partikel repräsentierten Zustandes $X_t^{(j)}$:

$$\pi_t^{(j)} = P(Z_t|X_t^{(j)}). \tag{2.30}$$

MCPF liefern damit zu jedem Zeitpunkt t eine Zustandsabschätzung mit Hilfe einer Partikelmenge $\{X_t^{(i)}, \pi_t^{(i)}\}_{i=1}^N$ für den Zustand $P(X_t|Z^t)$.

Durch ihre Fähigkeit, nahezu beliebige Verteilungen repräsentieren zu können, eignen sich MCPF für solche Situationen, in denen viele Hypothesen gleichzeitig modelliert werden sollen. Allerdings erfordern MCPF einen exponentiell mit der Dimension des Zustandsraums ansteigenden Rechenaufwand. Ein neuartiges Verfahren, welches die Dimension des durch Partikel repräsentierten Zustandsraums verringert, ist der Rao-Blackwellized Partikelfilter.

2.7 Rao-Blackwellized Partikelfilter

Rao-Blackwellized Partikelfiter arbeiten wie gewöhnliche Partikelfilter mit dem Unterschied, dass sie einen Teil des Zustandes analytisch berechnen. Den verbleibenden Teil repräsentieren sie über Partikel. Die Partikel erhalten durch die Extraktion bestimmter Zustandsgrößen eine geringere Dimension und die Varianz des Partikelfilters verringert sich [30]. Es wird deshalb eine geringere Anzahl von Partikeln benötigt, weil ein Teil des Zustandes analytisch berechnet und nicht über aufwändiger zu berechnende Repräsentantenmengen abgeschätzt wird.

Um die Zerlegung eines Zustandsraumes in einen analytisch zu berechnenden und einen verbleibenden Teil mathematisch zu beschreiben, werden Erscheinungskoeffizienten eingeführt. Nunmehr besteht der Zustand $X_t = (l_t, a_t)$ aus zwei Anteilen; dem Lokationsanteil l_t, der die Position des Objektes über Partikelfilter modelliert sowie dem Erscheinungsanteil a_t, der die analytisch zu berechnenden bzw. abzuschätzenden Unterraumkoeffizienten enthält.

Die Gleichung 2.27 wird dadurch abgewandelt zu:

$$P(l_t, a_t|Z_t) = kP(Z_t|l_t, a_t) \times \int_{l_{t-1}} \int_{a_{t-1}} P(l_t, a_t|l_{t-1}, a_{t-1}) P(l_{t-1}, a_{t-1}|Z^{t-1}) \tag{2.31}$$

In der oberen Gleichung wird sowohl über den Lokationsanteil l_{t-1} als auch über den Erscheinungsanteil a_{t-1} zur Zeit $t-1$ integriert. Nun wird der Erscheinungsan-

teil a_t aus dem Zustand herausgenommen, es bleibt demzufolge ein Filter für den Lokationsanteil l übrig:

$$P(l_t|Z_t) = k \int_{a_t} P(Z_t|l_t, a_t) \times \int_{l_{t-1}} \int_{a_{t-1}} P(l_t, a_t|l_{t-1}, a_{t-1}) P(l_{t-1}, a_{t-1}|Z^{t-1}) \quad (2.32)$$

Im Weiteren soll eine Beispielmodellierung dargestellt werden, bei der ein Teil des Objektzustands analytisch und der verbleibende Teil mit MCPF abgeschätzt wird. Die Beschreibung der verschiedenen Zustandskomponenten erfolgt dazu mittels eines Bayes-Netzes.

2.7.1 Repräsentation als Bayes-Netz

Bayes-Netze werden bei RBPF verwendet, um die Abhängigkeiten der verschiedenen Größen des Zustandsraumes repräsentieren zu können. Die zu modellierenden Größen wie z.B. Roboter- und Ballposition haben Einfluss auf verschiedene Sensordaten. So hängt die Wahrscheinlichkeit für die Wahrnehmung einer Landmarke z_t^l ausschließlich von der Roboterposition r_t, nicht jedoch von der Ballposition b_t ab. Die Wahrnehmung des Balls z_t^b hingegen hängt sowohl von der Roboterposition r_t als auch von der Ballposition b_t ab. Mit Hilfe des Bayes-Netzes können die a-posteriori Dichtefunktionen für die Ball- und Roboterposition einfacher berechnet werden [9], als ohne ein solches Netz. Bedingte Unabhängigkeiten werden zugunsten einer vereinfachten Berechnung berücksichtigt. Die resultierende Dichtefunktion dient der Berechnung der Gewichtung der Partikel.

Das Beispielmodell in Abb. 2.6 zeigt einen Agenten, der Sensorinformationen für Landmarken[15] z_t^l bzw. für den Ball z_t^b erhält. Daraus errechnet er die eigene Position sowie die Position des Balls [7]. Für die egozentrische[16] Ballmodellierung verwendet der Agent einen Kalmanfilter. Die Roboterposition wird hingegen ausschließlich mit Hilfe von Partikeln modelliert. Um die Ballposition in globale Koordinaten umzurechnen, wird die Roboterposition benötigt. Man erhält das Netzwerk aus Grafik 2.6.

Die Dichtefunktion der Roboterposition r_t berechnet sich dementsprechend als:

[15] als Landmarken werden Gegenstände, die der Orientierung dienen bezeichnet, beim RoboCup sind dies z.B. die Tore und Flaggen

[16] Roboter ist Koordinatenursprung

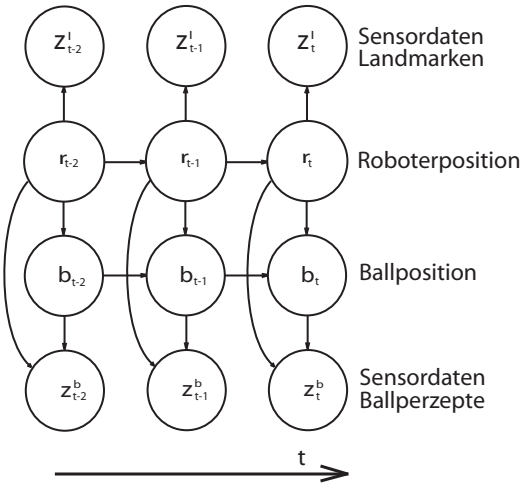


Abb. 2.6: *Dieses Bayes-Netz repräsentiert die Abhängigkeiten bei einer kombinierten Modellierung der Roboter- und Ballposition. Die Pfeile geben Aufschluss darüber, welche Abhängigkeiten zwischen den verschiedenen Größen existieren. Nicht eingezeichnete Pfeile deuten auf bedingte Unabhängigkeit hin.*

$$p(r_t|r_{t-1}^{(i)}, b_{t-1}^{(i)}, z_t) \propto p(z_t|r_t, r_{t-1}^{(i)}, b_{t-1}^{(i)}) p(r_t|r_{t-1}^{(i)}, b_{t-1}^{(i)}) \tag{2.33}$$

$$= p(z_t|r_t) p(r_t|r_{t-1}^{(i)}) \tag{2.34}$$

Die Gleichung 2.33 folgt aus dem Satz von Bayes bei gegebenem Bayes-Netz. Aus den in Abb. 2.6 dargestellten Abhängigkeiten der einzelnen Bestandteile des Zustandsraums gelangt man zu Gleichung 2.34.

Um einen Partikel zu repräsentieren, ist die Berechnung seiner Gewichtung π_t erforderlich. Die Berechnung hängt davon ab, ob eine Landmarke oder der Ball gesehen wurden. Bei einer erkannten Landmarke z_t^l gilt:

$$\pi_t^{(i)} \propto p(z_t^l|r_t^{(i)}, b_{t-1}^{(i)}) = p(z_t^l|r_t^{(i)}) \tag{2.35}$$

Bei einem wahrgenommenen Ballperzept z_t^b hingegen erhält man die Gleichung:

$$\pi_t^{(i)} \propto p(z_t^b|r_t^{(i)}, b_{t-1}^{(i)}) \tag{2.36}$$

Die resultierende Abschätzung approximiert dabei zu jedem Zeitpunkt eine Roboterposition und den Ballzustand gleichzeitig unter den gegebenen Abhängigkeiten.

2.8 Zusammenfassung

Bayesfilter stellen eine der erfolgreichsten Klassen von Filtern für die Objektmodellierung und Lokalisierung dar. Um sie wirksam einzusetzen, bedarf es vielfältiger mathematischer Grundlagen und Verfahren auf deren Basis die wichtigsten vorgestellten Filter arbeiten. Für lineare, zeitdiskrete unimodalverteilte Prozesse ist der Kalmanfilter, der auf Basis von Gaußverteilungen arbeitet, optimal. Auch für Abschätzungen von Zuständen mit vielen Dimensionen ist er sehr effizient anwendbar. Partikelfilter eignen sich für beliebige Verteilungsfunktionen. Sie zeichnen sich durch ihre Robustheit gegenüber stark verrauschten Daten aus. Ihre Schwäche besteht jedoch neben einer für lineare Prozesse geringeren Genauigkeit in ihrem höheren Rechenaufwand. Bei einem zu hoch dimensionierten Hypothesenraum gelangen Partikelfilter, durch die mit der Dimension exponentiell ansteigende Zahl benötigter Repräsentanten, schnell an ihre Grenzen. Rao-Blackwellized Partikelfilter kombinieren die Vorteile analytischer Verfahren wie Kalmanfiltern mit den Vorteilen von Monte-Carlo Partikelfiltern.
Nach der Beschäftigung mit den vorrangig theoretischen Grundlagen der Objektmodellierung sollen im Folgenden die Anwendung und Evaluierung eines Rao-Blackwellized Partikelfilters im Zentrum der Betrachtung stehen.

Kapitel 3

Rao-Blackwellized Partikelfilter

In diesem Kapitel wird eine Beispielanwendung für einen Rao-Blackwellized Partikelfilter anhand der RoboCup-Domäne vorgestellt, die auf Erkenntnissen von Kwok und Fox [40] basiert. Als Plattform dient ein Roboter vom Typ Sony Aibo ERS-210A bzw. ERS-7[1]. Im Kern geht es dabei um die zuverlässige Modellierung der Ballgeschwindigkeit und -position. Des Weiteren werden die Erweiterung des Verfahrens um die Verwendung von Negativinformationen vorgeschlagen und eine Aufmerksamkeitssteuerung für die Ballsuche präsentiert.

3.1 Motivation

Die Modellierung der Ballposition im RoboCup stellt eine besondere Herausforderung und damit eine geeignete Basis für viele Arbeiten im Rahmen der Objektmodellierung und -verfolgung[2] dar. Für einen Roboter ist es sehr wichtig zu wissen, wo er sich selbst befindet und an welcher Stelle sich Objekte befinden, mit denen er interagieren möchte.

Während eines RoboCup-Spiels sind sowohl Ballposition und -geschwindigkeit als auch die eigene Position eines Roboters permanenten Änderungen unterworfen. Insbesondere die Position und Geschwindigkeit des Balls verhalten sich oft hochgradig nichtlinear, was z.B. die Verwendung von Kalmanfiltern erschwert. Beispiel für nichtlineares Verhalten ist ein Ball, der von einem Roboter geschossen wird oder an anderen Robotern bzw. Begrenzungen des Spielfeldes abprallt. Geschwindigkeit und Position eines sich frei bewegenden Balls hingegen verhalten sich linear. Ein

[1] eine genaue Beschreibung dieser Plattform befindet sich in Anhang A

[2] engl. tracking

einfacher Ansatz beim Aufbau des Weltmodells[3] des Roboters ist vielfach die getrennte Modellierung von Selbstlokalisierung einerseits und von Objekten wie z.B. dem Ball andererseits [40]. Unsicherheiten der eigenen Positionsbestimmung werden dabei für die Objektverfolgung nicht berücksichtigt. Theorie und praktische Erfahrungen zeigen aber, dass der hohe Interaktionsgrad des Balls mit seiner Umgebung eine kombinierte Modellierung der eigenen Position und der Position von zu verfolgenden Objekten wie dem Ball erfordert.

Eine mögliche Arbeitsrichtung ist deshalb ein Rahmenkonzept auf der Basis eines RBPF, welches die Modellierung der Roboterposition sowie die der Ballposition kombiniert. Dabei werden die linearen Bewegungsanteile des Balls durch Kalmanfilter modelliert. Roboterposition und Interaktionen des Balls mit der Umgebung, die nichtlineares Verhalten zur Folge haben, werden durch Partikelfilter abgeschätzt. Dadurch sollen die Vorteile von linearen und nichtlinearen Modellierungsverfahren in einer Architektur kombiniert werden.

In einem ersten Schritt möchte deshalb der Verfasser eine mögliche Implementierung eines Kalmanfilters vorstellen, ähnlich wie sie auch sehr erfolgreich im German Team Code 2004 verwendet wurde. In einem zweiten Schritten wird darauf aufbauend ein Partikelfilter implementiert.

3.2 Kalmanfilter Referenzdesign

Wie bereits im 2. Kapitel beschrieben wurde, schätzen Kalmanfilter den Objektzustand optimal aus verrauschten, multidimensionalen Sensordaten linearer Prozesse ab. Da die Zustandsabschätzung zum Großteil auf der Verarbeitung von Kovarianzmatrizen basiert, ist eine exakte Ermittlung dieser Daten sehr wichtig. Die Kalmanfilterung erfolgt in zwei Schritten:

1. Vorausschau *(Predict)* des neuen Zustands aus dem alten Zustand und

2. Aktualisierung *(Update)* der vorausberechneten Daten anhand der Sensordaten und des Kalmangains.

Um die Aktualisierung durchführen zu können, muss dem System mitgeteilt werden, wie exakt die Sensordaten die Realität repräsentieren.

[3] ein Weltmodell enthält modellierte Größen wie die eigene Position bzw. die Position von Objekten der Umgebung

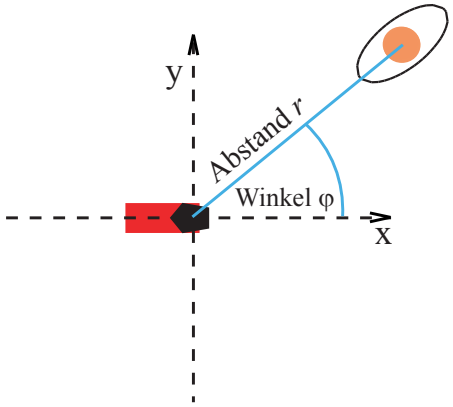


Abb. 3.1: *Ein Ballperzept enthält Daten über die Ballposition relativ zum Roboter in Form von Abstand und Winkel (blau). Die Ellipse repräsentiert die Messfehlermatrix R. Weiterhin ist die Darstellung des kartesischen Roboterkoordinatensystems erkennbar.*

3.2.1 Einige Bemerkungen zu Sensordaten

Eingangsgrößen bilden bei diesem Kalmanfilter die Ballperzepte z_t zum Zeitpunkt t, die aus Sensordaten gewonnen werden. Ein Ballperzept enthält Daten über den Ballabstand r_t sowie den Winkel des Balls φ_t relativ zum Roboter (Abb. 3.1)[4].

Die Geschwindigkeit v_t des Balls zum Zeitpunkt t wird über den Quotienten der örtlichen Differenz zweier Ballperzepte sowie ihrer zeitlichen Differenz abgeschätzt, wobei t die Bildtaktung und $\hat{t}$ die Zeit in [ms] darstellt:

$$v_t \approx \frac{z_t - z_{t-1}}{\hat{t}_t - \hat{t}_{t-1}} \tag{3.1}$$

Die zeitliche Differenz zweier Ballperzepte wird über die Bilderanzahl, die zwischen beiden Perzepten liegt, ermittelt, weil die Bilder beim Roboter stets äquidistant[5] erfasst werden. Die örtliche Differenz zweier relativ zur Roboterposition erfasster Perzepte wird zudem von Eigenbewegungen bereinigt.

[4] eine genaue Beschreibung der Berechnung der Ballperzepte aus den Sensordaten erfolgt in Abschnitt A.2.2

[5] der zeitliche Abstand zwischen zwei Bildern ist konstant

3.2.2 Entwurf des Zustandsraums

Als Eingangsgrößen für die Kalmanfilterung dienen zu jedem Zeitpunkt t jeweils ein Ballperzept z_t sowie die Zeit der Erfassung $\hat{t}_t$. Die Zeit ist für die Geschwindigkeitsberechnung erforderlich. Als Resultat der Ballmodellierung soll der Kalmanfilter sowohl die Ballposition s_t als auch die Ballgeschwindigkeit v_t modellieren.

Damit lässt sich der Filterprozess mit seinen Eingangs- und Ausgangsgrößen zusammenfassend in folgender Gleichung beschreiben:

$$\begin{pmatrix} s_{x_t} \\ s_{y_t} \\ v_{x_t} \\ v_{y_t} \end{pmatrix} = \textbf{Kalmanfilterung} \times \begin{pmatrix} z_{x_t} \\ z_{y_t} \\ \hat{t}_t \end{pmatrix} \tag{3.2}$$

Die Repräsentation der Ballposition erfolgt egozentrisch; Bezugssystem ist also das Roboterkoordinatensystem und nicht das Spielfeld. Damit werden Modellierungsfehler, die aus der Positionsbestimmung des Roboters auf dem Spielfeld resultieren, ausgeschlossen. Der Betrag der Geschwindigkeit soll relativ zum Spielfeld, die Richtung relativ zum Roboter berechnet werden.

3.2.3 Messkovarianz-, Prozesskovarianz-, Propagierungsmatrizen

Die Arbeitsweise eines Kalmanfilters wird maßgeblich durch seine Messfehler-, Prozessfehler- und Propagierungsmatrizen bestimmt. Durch diese Matrizen wird sowohl die Genauigkeit der Sensordaten als auch die Vorhersagepräzision eines aktuellen Zustands aus dem vorhergehenden dargestellt. Im Folgenden wird die Ermittlung der Werte dieser Matrizen beschrieben.

3.2.3.1 Messfehlermatrix R

Die Ermittlung der Messfehler kann durch einen auf der Stelle stehenden oder auf der Stelle laufenden Roboter geschehen. Der Roboter misst dabei über einen ausreichend großen Zeitraum mehrmals den Abstand zu einem sich vor ihm befindlichen Ball. Ein laufender Roboter ist einem stehenden zu bevorzugen, weil die Roboter während eines Spiels die meiste Zeit in Bewegung sind. Dadurch müssen sie verrauschtere Sensordaten verarbeiten als stehende Roboter. Es wurde im Rahmen von Messungen[6] festgestellt, dass die Kovarianzen der gemessenen Ballposition von der Ballposition

[6] Kapitel 4

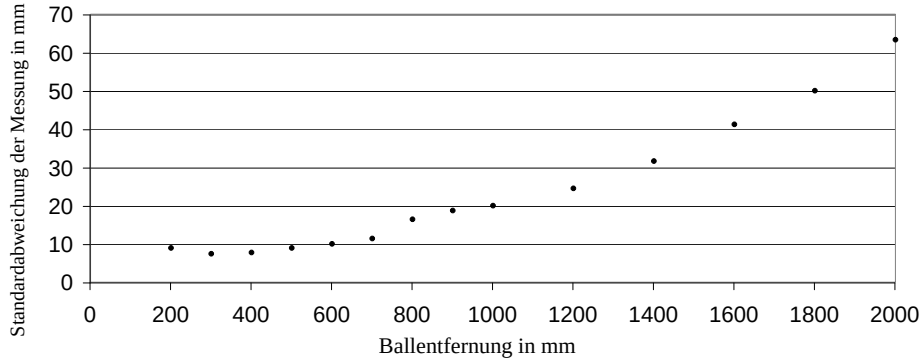


Abb. 3.2: *Standardabweichung der gemessenen Ballentfernung bei gegebenem Ballabstand.*

selbst, insbesondere von der Ballentfernung, abhängen. Eine Ursache dafür ist die begrenzte Kameraauflösung. Die Varianz für die Ballentfernung σ_r steigt, wie Messungen ergaben, in der zweiten Potenz mit der Ballentfernung r (Abb. 3.2). Für den Ballwinkel φ bleibt die Varianz bei unterschiedlichen Ballentfernungen und -winkeln konstant. Zudem kommt es in der Nähe des Roboters verstärkt zu Messfehlern, weil der Ball von der Kamera nicht mehr komplett erfasst werden kann. Die Messungen zeigten auch, dass die Ballabstands- und Ballwinkelmessfehler unkorreliert sind.

Weil das Weltmodell des Roboters nicht in Polarkoordinaten, sondern in kartesischen Koordinaten aufgebaut ist, muss die Messfehlermatrix R_{polar} (3.4) in kartesische Koordinaten umgerechnet und eine Koordinatenkreuztransformation entsprechend dem Ballwinkel φ durchgeführt werden (3.5). Dazu wird eine Transformationsmatrix T (3.3) generiert:

$$\mathbf{T} = \begin{pmatrix} cos(\varphi) & -sin(\varphi) & cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) & sin(\varphi) & cos(\varphi) \\ cos(\varphi) & -sin(\varphi) & cos(\varphi) & -sin(\varphi) \\ sin(\varphi) & cos(\varphi) & sin(\varphi) & cos(\varphi) \end{pmatrix} \tag{3.3}$$

$$\mathbf{R_{polar}} = \begin{pmatrix} \sigma^2_{R_{rr}} & \sigma^2_{R_{r\varphi}} & \sigma^2_{R_{r\dot{r}}} & \sigma^2_{R_{r\dot{\varphi}}} \\ \sigma^2_{R_{r\varphi}} & \sigma^2_{R_{\varphi\varphi}} & \sigma^2_{R_{\varphi\dot{r}}} & \sigma^2_{R_{\varphi\dot{\varphi}}} \\ \sigma^2_{R_{r\dot{r}}} & \sigma^2_{R_{\varphi\dot{r}}} & \sigma^2_{R_{\dot{r}\dot{r}}} & \sigma^2_{R_{\dot{r}\dot{\varphi}}} \\ \sigma^2_{R_{r\dot{\varphi}}} & \sigma^2_{R_{\varphi\dot{\varphi}}} & \sigma^2_{R_{\dot{r}\dot{\varphi}}} & \sigma^2_{R_{\dot{\varphi}\dot{\varphi}}} \end{pmatrix} \tag{3.4}$$

Die Transformationsgleichung 3.5 liefert als Resultat eine Messfehlermatrix auf der Basis von Roboterkoordinaten, siehe Gleichung 3.6.

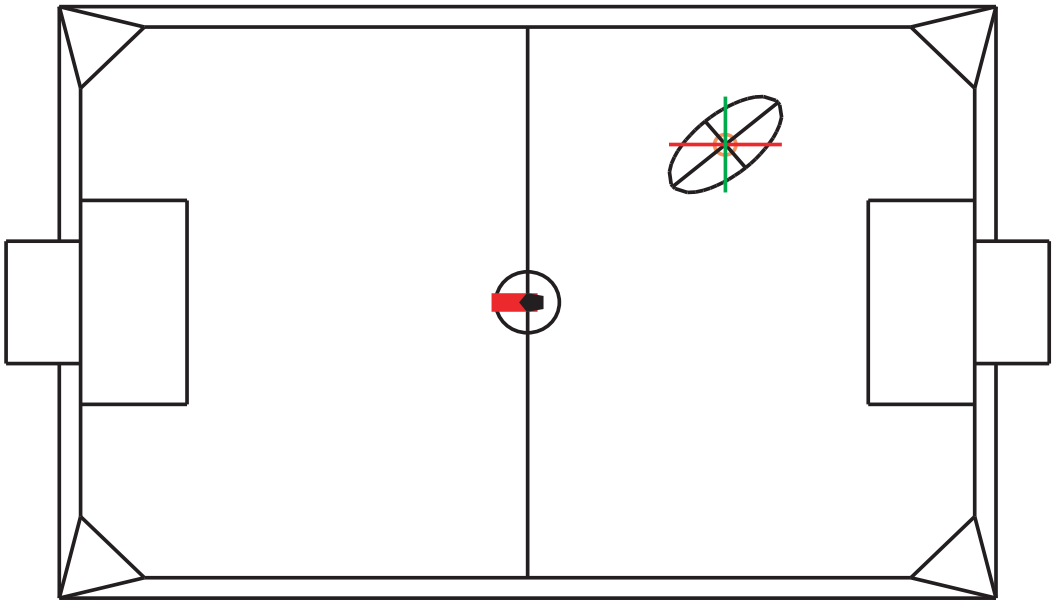


Abb. 3.3: *Messkovarianz als Ellipse visualisiert, nicht maßstabsgetreu. schwarz: die ursprünglichen Entfernungs- und Winkelstandardabweichung; rot:* σ_x*, grün:* σ_y *, durch Hauptachsentransformation gewonnen.*
Anmerkung: Blickt der Roboter auf einen im Winkel von 0° vor ihm befindlichen Ball, liegen keine Korrelationen zwischen σ_x *und* σ_y *vor. Bei einem im Winkel von 45° vor ihm liegenden Ball sind die Korrelationen aufgrund der in Polarkoordinaten berechneten Ballperzepte maximal.*

$$R_{cart} = T * R_{polar} * T^T \tag{3.5}$$

$$\mathbf{R_{cart}} = \begin{pmatrix} \sigma^2_{R_{xx}} & \sigma^2_{R_{xy}} & \sigma^2_{R_{xv_x}} & \sigma^2_{R_{xv_y}} \\ \sigma^2_{R_{xy}} & \sigma^2_{R_{yy}} & \sigma^2_{R_{yv_x}} & \sigma^2_{R_{yv_y}} \\ \sigma^2_{R_{xv_x}} & \sigma^2_{R_{yv_x}} & \sigma^2_{R_{v_xv_x}} & \sigma^2_{R_{v_xv_y}} \\ \sigma^2_{R_{xv_y}} & \sigma^2_{R_{yv_y}} & \sigma^2_{R_{v_xv_y}} & \sigma^2_{R_{v_yv_y}} \end{pmatrix} \tag{3.6}$$

3.2.3.2 Propagierungsmatrix A

Die Berechnung des neuen Ballzustandes $\hat{x}_t$ erfolgt aus dem alten abgeschätzten Ballzustand $\hat{x}_{t-1}$ mittels der Propagierungsmatrix A. Wenn sich nun der Roboter bewegt und damit die Position des Balls relativ zu sich selbst beeinflusst, muss

diese Bewegung mit Hilfe der Eingangsmatrix B berücksichtigt werden (siehe Gleichung 2.15). Die Propagierungsmatrix berücksichtigt die Zeitabstände zwischen zwei Zuständen implizit, d.h. die Zustandsabschätzung erfolgt in konstanten Zeitabständen. Die Propagierung der Position s_t zur Zeit t berechnet sich aus der Position s_{t-1}, der Geschwindigkeit v_{t-1} sowie der Zeitdifferenz zwischen dem letzten und dem aktuellen Zustand:

$$s_t = s_{t-1} + \int_{\hat{t}_{t-1}}^{\hat{t}_t} v_i di \tag{3.7}$$

$$s_t \approx s_{t-1} + v_{t-1}(\hat{t}_t - \hat{t}_{t-1}) \tag{3.8}$$

Die Geschwindigkeit v_t zur Zeit t berechnet sich aus der Geschwindigkeit v_{t-1} bei $t-1$ und nimmt einen exponentiell abfallenden Verlauf an:

$$v_t = v_{t-1} exp^{\gamma(\hat{t}_t - \hat{t}_{t-1})} \tag{3.9}$$

Der Diskontierungsfaktor $\gamma < 1$ ist experimentell durch Analyse eines auf dem Spielfeld ausrollenden Balls zu ermitteln. Er sorgt für einen realistischen Geschwindigkeitsabfall im Modell.

Weil die Sensordaten immer im gleichen Zeitabstand $\Delta t \triangleq \hat{t}_t - \hat{t}_{t-1}$ eintreffen, kann die Propagierungsfunktion vereinfacht werden zu:

$$s_t \approx s_{t-1} + v_{t-1}\Delta t \tag{3.10}$$

$$v_t = v_{t-1} exp^{\gamma \Delta t} \tag{3.11}$$

Werden die vom Roboter ausgeführten Eigenbewegungen außer Acht gelassen, gelangt man zu folgender Propagierungsfunktion:

$$\begin{pmatrix} s_{x_t} \\ s_{y_t} \\ v_{x_t} \\ v_{y_t} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & exp^{\gamma \Delta t} & 0 \\ 0 & 0 & 0 & exp^{\gamma \Delta t} \end{pmatrix} \times \begin{pmatrix} s_{x_{t-1}} \\ s_{y_{t-1}} \\ v_{x_{t-1}} \\ v_{y_{t-1}} \end{pmatrix} \tag{3.12}$$

Gleichung 3.12 entspricht einer Vereinfachung von Gleichung 2.15 in der Form:

$$\hat{x}_t = A\hat{x}_{t-1} \tag{3.13}$$

3.2.3.3 Eingangsmatrix B

Die Berücksichtigung der vom Roboter ausgeführten Bewegungen u_t zum Zeitpunkt t erfolgt mithilfe der Eingangsmatrix[7] B. Der Roboter kann sowohl Translations- als auch Rotationsbewegungen ausführen. Die aktuelle Ballposition muss durch die ausgeführten Roboterbewegungen aktualisiert werden, um die Ballgeschwindigkeit relativ zum Spielfeld repräsentieren zu können. Translationsbewegungen des Roboters werden in kartesischen Koordinaten durch die Variablen u_{x_t} sowie u_{y_t} ausgedrückt, demgegenüber werden Rotationsbewegungen durch u_{θ_t} repräsentiert. Die Translationsbewegung des Roboters kann durch Vektoraddition der Ballposition s_{t-1} mit der im letzten Zeitintervall zurückgelegten Roboterstrecke $\overrightarrow{u_{trans}}$ kompensiert werden. Bei einer Rotationsbewegung des Roboters müssen die Ballposition und die Ballgeschwindigkeit durch Multiplikation mit einer Rotationsmatrix angepasst werden. Für die Kompensation der Eigenbewegung u_t im Ballzustand $\hat{x}_t$ ergibt sich zu einem gegebenen Zeitpunkt t folgende Gesamtgleichung:

$$\begin{pmatrix} s'_x \\ s'_y \\ v'_x \\ v'_y \end{pmatrix} = \begin{pmatrix} \cos(u_\theta) & -\sin(u_\theta) & 0 & 0 \\ \sin(u_\theta) & \cos(u_\theta) & 0 & 0 \\ 0 & 0 & \cos(u_\theta) & -\sin(u_\theta) \\ 0 & 0 & \sin(u_\theta) & \cos(u_\theta) \end{pmatrix} \times \begin{pmatrix} s_x \\ s_y \\ v_x \\ v_y \end{pmatrix} + \begin{pmatrix} u_x \\ u_y \\ 0 \\ 0 \end{pmatrix} \tag{3.14}$$

3.2.3.4 Prozessfehlermatrix Q

Die Prozessfehlermatrix Q wird benötigt, um den Propagierungsfehler aufzuzeigen, der bei der Vorausberechnung des neuen Zustandes $\hat{x}_t^-$ zum Zeitpunkt t aus dem vorhergehenden abgeschätzten Zustand $\hat{x}_{t-1}$ entsteht. Die Matrix wird in kartesischen Koordinaten erstellt. Die experimentelle Herleitung der Prozessfehler erfordert eine externe Messquelle, z.B. eine Deckenkamera oder einen Laserscanner. Mit diesen externen Messinstrumenten können die vom Roboter vorausberechneten Werte mit den gemessenen Werten verglichen werden. Aus den mittleren Abweichungen von gemessenen zu vorausberechneten Werten kann die Prozessfehlermatrix erstellt werden. Da im Versuch keine äußeren Messinstrumente zur experimentellen Herleitung der Prozessfehler existierten, wurde der Prozessfehler geschätzt. Basis der Abschätzung waren Experimente mit einem rollenden Ball. Verschiedene Trajektorien realer Bälle

[7] engl. input matrix

wurden experimentell ermittelt und mit einem idealen Ball verglichen. Ein idealer Ball soll eine einmal eingeschlagene Rollrichtung beibehalten solange er nicht mit Gegenständen seiner Umgebung kollidiert. Der reale Ball in der Sony-Liga bewegt sich jedoch hin und wieder parabelförmig oder völlig nicht-deterministisch. Zum einen, weil er seinen Schwerpunkt nicht exakt im Zentrum besitzt und zum anderen, weil er z.T. auf welligem Untergrund rollt. Diese Abweichungen des Rollverhaltens eines realen zu einem idealen Ball spiegeln sich in der Prozessfehlermatrix wider. Je größer der Prozessfehler ist, desto höher wird die Aktualisierung des vorausberechneten Zustands x_t^- mit den Sensordaten z_t ausfallen.

3.3 Rao-Blackwellized Partikelfilter

Der Kalmanfilter modelliert das lineare Ballverhalten optimal. Für die Modellierung nichtlinearen Ballverhaltens ist er aber weniger geeignet. Deshalb sollen als Nächstes die wichtigsten Fälle für nichtlineares Ballverhalten vorgestellt werden, um darauf aufbauend eine Erweiterung des bisherigen Filterverfahrens auf Monte-Carlo Partikelbasis zu begründen. Einige wichtige Ansätze zur Modellierung von Objekten in nichtlinearen Umgebungen mit Hilfe von Partikelfiltern gehen auf die Arbeiten von Fox und Kwok [7, 40] zurück.
Die Hauptgründe für nichtlineares Ballverhalten sind:

1. Nichtlineare Bewegungen des Beobachters. Wegen Traktionsverlust bzw. Kollisionen [24, 25] mit anderen Robotern und Gegenständen sind die Bewegungsdaten des Roboters z.T. sehr verrauscht. Damit wird die Unterscheidung zwischen der Eigenbewegung des Roboters und der Bewegung des Balls erschwert.

2. Physikalische Interaktionen des Balls mit der Umwelt. Der Ball prallt häufig von der Spielfeldbande ab oder wird von Robotern geschossen. Dadurch entstehen ebenfalls nichtlineare Bewegungen.

3. Physikalische Interaktion des Balls mit dem Beobachter. In Fällen, in denen der beobachtende Roboter den Ball selbst greift oder wegschießt, ist die Ballposition in hohem Maße von den Aktionen dieses Roboters abhängig.

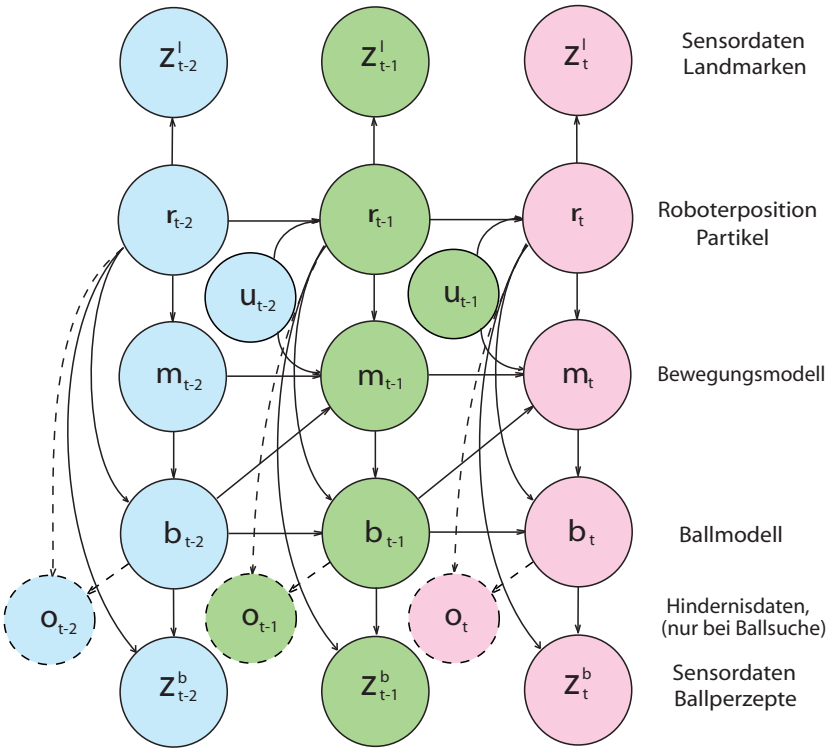


Abb. 3.4: *Bayes-Netz zusammen mit Bewegungsmodell und Hindernisdaten. Verschiedene Farben stehen für verschiedene Zeitschritte. Die Pfeile repräsentieren die direkten Abhängigkeiten der Variablen voneinander.*

3.4 Das Bayes-Netz für den RBPF

Im Folgenden Abschnitt sollen die Abhängigkeiten der Zustandsgrößen voneinander sowie von den Sensordaten im Zentrum der Betrachtung stehen. Dazu werden die Abhängigkeiten in einem Bayes-Netz dargestellt (Abb. 3.4).

In dem in Abb. 3.4 dargestellten Modell ist die Roboterposition der zuerst zu modellierende Teil im Lokalisierungsverfahren, weil die Wahrscheinlichkeitsdichte für das Wahrnehmen einer Landmarke lediglich von der Roboterposition abhängt. Die Roboterposition r_t zum Zeitpunkt t ergibt sich allein aus der vorhergehenden Roboterposition r_{t-1} sowie der ausgeführten Bewegung u_{t-1} zum Zeitpunkt $t-1$. Die Wahrscheinlichkeitsdichte $p(z_t^b)$ für das Wahrnehmen eines Balls hingegen ist indirekt vom Bewegungsmodell m_t sowie direkt von der Ballposition b_t und der Roboterposition r_t abhängig. Bei der Berechnung der einzelnen Wahrscheinlichkeitsdichten

sind jedoch nur die direkten Abhängigkeiten zu betrachten, wodurch die Berechnung sehr vereinfacht wird.

Ein Lokalisierungspartikel $r_t^{(i)}$ berechnet sich zunächst:

$$r_t^{(i)} \backsim p(r_t|r_{t-1}^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, z_t, u_{t-1}) \tag{3.15}$$

Das Bewegungsmodell $m_t^{(i)}$ berechnet sich dementsprechend:

$$m_t^{(i)} \backsim p(m_t|r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, z_t, u_{t-1}) \tag{3.16}$$

Die Berechnungen der Ballgeschwindigkeit und der -position $b_t^{(i)}$ erfolgen zum Schluss:

$$b_t^{(i)} \backsim p(b_t|r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, z_t) \tag{3.17}$$

Durch Verwendung von Bewegungsmodellen für nichtlineares Ballverhalten können nun die Ballposition und -geschwindigkeit analytisch durch Verwendung eines Kalmanfilters für jeden Partikel berechnet werden.

Die Berechnung der Roboterposition r_t bei Sensorinformation z_t erfolgt durch Bayes und mit Hilfe der bedingten Unabhängigkeiten:

$p(r_t|r_{t-1}^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, z_t, u_{t-1})$

$$\propto p(z_t|r_t, r_{t-1}^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) p(r_t|r_{t-1}^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) \tag{3.18}$$

$$= p(z_t|r_t, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) p(r_t|r_{t-1}^{(i)}, u_{t-1}) \tag{3.19}$$

Mit Bayes sowie den gegebenen Unabhängigkeiten berechnet sich die Wahrscheinlichkeit für ein Bewegungsmodell m_t zum Zeitpunkt t nach:

$p(m_t|r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, z_t, u_{t-1})$

$$\propto p(z_t|m_t, r_t^{(i)}, b_{t-1}^{(i)}) p(m_t|r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) \tag{3.20}$$

3.4.1 Gewichtung und Propagierung der Partikel bei gegebenem Bayes-Netz

Im Fall einer wahrgenommenen Landmarke z_t^l berechnet sich die Gewichtung π_t^i eines Partikels i wie folgt:

$$\pi_t^{(i)} \propto p(z_t^l|r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) \tag{3.21}$$

$$= p(z_t^l|r_t^{(i)}) \tag{3.22}$$

Bei einem wahrgenommenen Ballperzept z_t^b ergibt sich folgende Berechnung für die Gewichtung π_t^i eines Partikels i:

$$\pi_t^{(i)} \propto p(z_t^l | r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) \tag{3.23}$$

$$= \sum_{M_t} p(z_t^b | M_t, r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) p(M_t | r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) \tag{3.24}$$

$$= \sum_{M_t} p(z_t^b | M_t, r_t^{(i)}, b_{t-1}^{(i)}) p(M_t | r_t^{(i)}, m_{t-1}^{(i)}, b_{t-1}^{(i)}, u_{t-1}) \tag{3.25}$$

wobei M_t über alle Bewegungsmodelle iteriert wird. Dabei wird in jedem Summand die Wahrscheinlichkeit für das jeweilige Bewegungsmodell mit der Wahrscheinlichkeit für das Wahrnehmen eines Balls multipliziert.

Nach der Gewichtung aller Partikel und einer Normierung der Gewichte werden die Partikel neu generiert[8] Es gibt verschiedene Verfahren, um die Auswahl der in den nächsten Schritt zu übertragenden Partikel zu treffen [17], die sich insbesondere durch die konkrete Aufgabenstellung und den Berechnungsaufwand unterscheiden. Für die hier vorgestellte Ballmodellierung wurde ein Monte-Carlo Propagierungsverfahren für die Partikel angewandt. Hervorzuheben ist, dass die Wahrscheinlichkeit für die Propagierung eines Partikels i zur Zeit t proportional zu seiner Gewichtung π_t^i ist.

3.4.2 Zu einigen Bewegungsmodellen des Balls

Die folgenden Bewegungsmodelle sind angelehnt an Arbeiten von Kwok und Fox [7] und dienen der Repräsentation verschiedener Möglichkeiten für ein nichtlineares Ballverhalten. Dabei möchte der Autor auf 5 Hauptfälle eingehen.

3.4.2.1 Keine Interaktion mit der Umwelt

Das erste Modell beinhaltet den Fall, dass keine Interaktionen mit der Umwelt stattfinden. Hier bewegt sich der Ball frei auf dem Spielfeld oder er ruht. Für die Modellierung dieser Situation kann ein Kalmanfilter verwendet werden, da hier die Linearitätsannahmen gelten. Die Geschwindigkeit eines ausrollenden Balls fällt zwar exponentiell ab, der exponentielle Abfall kann jedoch in guter Näherung rekursiv linear berechnet werden, weil der Zeitabstand $\Delta t \approx 33ms$ zwischen je zwei Zustandsberechnungen sehr klein und konstant ist.

[8] engl. resampling

3.4.2.2 Der Ball prallt an einem Gegenstand ab

Beim zweiten Modell prallt der Ball von der Spielfeldbegrenzung (Bande) ab. Um abschätzen zu können, wann sich der Ball nahe genug an der Bande befindet, wird die allozentrische Ballposition aus der egozentrischen Ballposition in Verbindung mit der Roboterposition auf dem Spielfeld berechnet. Die Selbstlokalisierung eines Roboters ist mit einer bestimmten Ungenauigkeit verbunden, die bei der Umrechnung der Ballposition in Spielfeldkoordinaten berücksichtigt werden muss. Es werden dazu die Standardabweichungen $\sigma_x, \sigma_y, \sigma_\varphi$ als Maß für die Ungenauigkeit der Roboterposition sowie der Blickrichtung bestimmt. Die Berechnung der Standardabweichungen $\sigma_x, \sigma_y, \sigma_\varphi$ erfolgt mittels der Partikel der Selbstlokalisierung, die unabhängig von der Ballmodellierung bestimmt wurden. Den Ballhypothesen wird danach je nach Standardabweichung $\sigma_{x_t}, \sigma_{y_t}, \sigma_{\varphi_t}$ zum Zeitpunkt t eine verrauschte Roboterposition zugeordnet (Abb. 3.5). Damit kann die globale Position der Ballhypothese berechnet werden. Bei einem sehr ungenau lokalisierten Roboter streuen in der Folge die globalen Positionen der Ballhypothesen sehr stark.

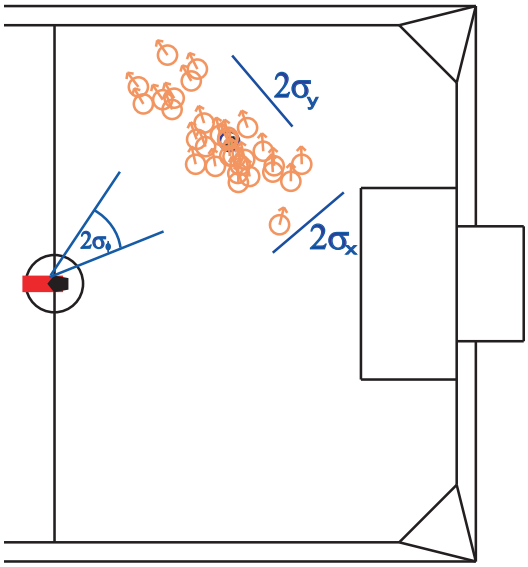


Abb. 3.5: Die auf die modellierten allozentrischen Ballpositionen propagierte Unsicherheit über die Roboterlokalisierung, $\sigma_x = \sigma_y = 100mm$, $\sigma_{rot} = 0,25rad$, 5 fach vergrößert

Es folgt die Überprüfung jeder Hypothese auf Kollision mit der Spielfeldbegrenzung. Falls sich die allozentrische Position einer Ballhypothese außerhalb des Spielfeldes befindet, wechselt das Bewegungsmodell der Hypothese von „Frei beweglich“ auf

„Abgeprallt“. Durch die verrauschten Roboterpositionen in den einzelnen Ballhypothesen gibt es oft widersprüchliche Bewegungsmodelle in verschiedenen Hypothesen (Abb. 3.6), die erst durch den Vergleich mit den Sensordaten bestätigt oder widerlegt werden können. Die Ballgeschwindigkeit und -richtung werden im Fall eines Abpralls unter der Annahme angepasst, dass sich die Ballgeschwindigkeit verringert und der Ausfallswinkel mindestens genauso groß ist wie der Einfallswinkel (Abb. 3.7). Diese Annahmen wurden durch zahlreiche Versuche des Verfassers und Erfahrungswerte in dieser Umgebung bestätigt. Nach der Modellierung des Abpralls wechselt das Bewegungsmodell der Hypothese wieder in „Frei beweglich“.

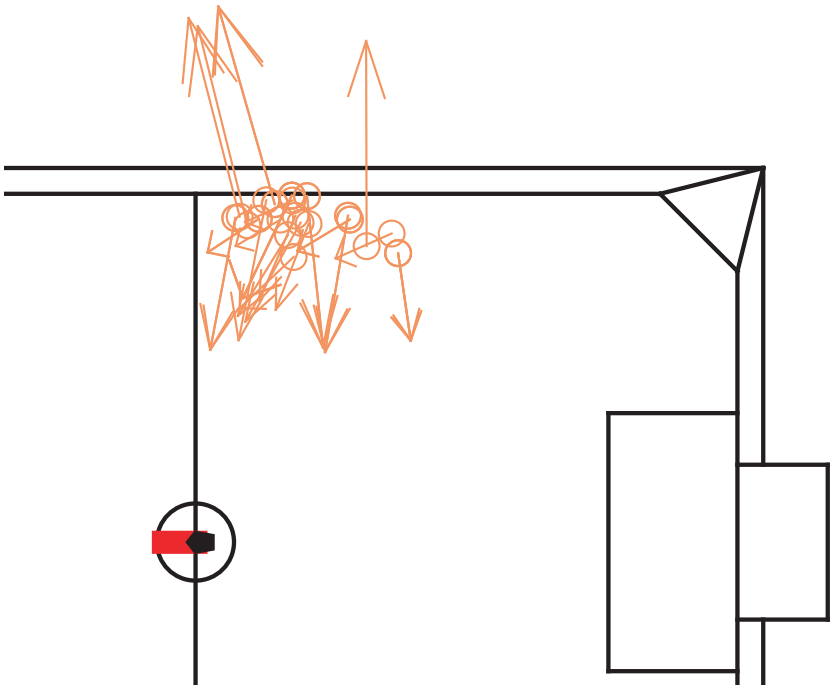


Abb. 3.6: *Beispiel mit 30 Partikeln. Je nach globaler Position eines jeden Partikels kommt es zu Interaktionen mit der Umgebung oder nicht. Die Pfeile geben Auskunft über Betrag und Richtung der Ballgeschwindigkeit des Partikels.*

3.4.2.3 Der Roboter greift den Ball

Im dritten Fall hat der beobachtende Roboter den Ball gefangen. In diesem Fall ist die Ballposition eng mit der Position des Roboters verbunden. Das Bewegungsmodell ändert sich auf „Gefangen“. Der Ball befindet sich in einem festen Abstand und Winkel zum Roboter. Da ein gegriffener Ball aufgrund seiner Nähe zum Roboter

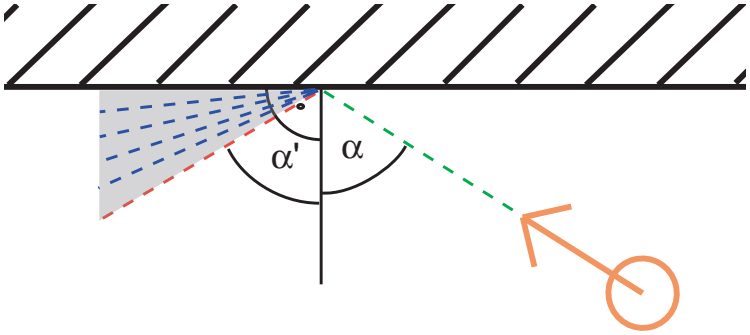


Abb. 3.7: *Modellierung eines Bandenabpralls, grün: Einfallsrichtung Winkel* α*;*
rot: ideale Reflexionsrichtung, Winkel $\alpha' = \alpha$*;*
blau/grau: reale Reflexionsrichtung, $\alpha' \leq \alpha$

nicht über die Kamera wahrgenommen werden kann, dient der PSD-Sensor in der Brust des Roboters zur Überprüfung, ob der Ball verloren wurde oder nicht. Wurde der Ball verloren, muss seine Position neu bestimmt werden und das Bewegungsmodell ändert sich von „Gefangen“ auf „Frei beweglich“.

3.4.2.4 Der Roboter schießt den Ball

Nachdem der Ball vom Roboter gefangen wurde, wählt der Roboter einen Schuss aus. Dies erfolgt im German Team mit Hilfe der sog. „Kick Selection Table“. Diese Tabelle besitzt für jeden verfügbaren Schuss eine stochastische Verteilung der Ballpositionen nach dem Schuss, die durch Beispielschüsse trainiert wurde (Abb. 3.8). Führt der Roboter einen Schuss aus, so können anhand dieser Daten Voraussagen darüber getroffen werden, mit welcher Wahrscheinlichkeit sich der Ball in eine bestimmte Richtung bewegt.

Das Bewegungsmodell ändert sich von „Gefangen“ auf „Geschossen“. Für das Partikelmodell bedeutet dies, dass nach Ausführung eines Schusses die Ballzustände des Kalmanfilters der Ballhypothesen unter Berücksichtigung der jeweiligen stochastischen Verteilung des Schusses auf die zu erwartende Geschwindigkeit und Richtung initialisiert werden (Abb. 3.9). Der Roboter kann dadurch schneller in die Richtung blicken, in die sich der Ball bewegt haben müsste. Gleichfalls kann es passieren, dass der Schuss missglückt ist. Daher bleibt das Bewegungsmodell einiger weniger Partikel nach einem Schuss unverändert im Zustand „Gefangen“.

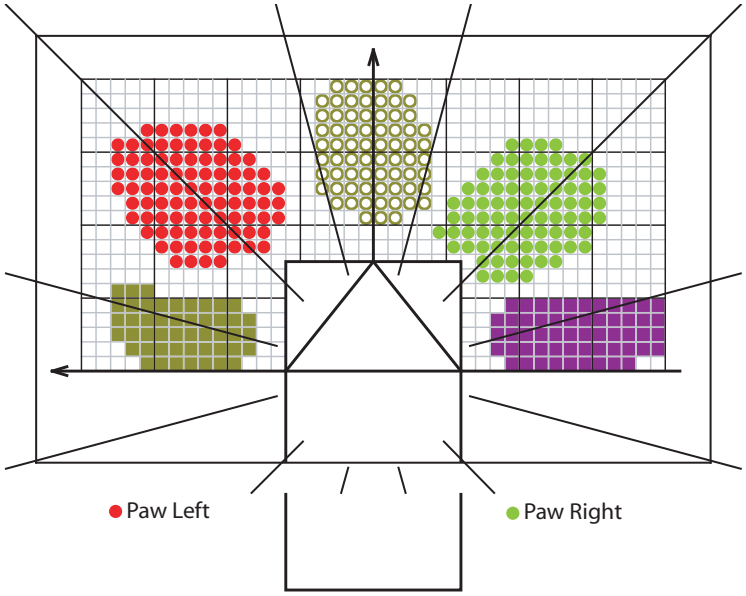


Abb. 3.8: *Die Kick Selection Table des German Teams. Im Diagramm ist die resultierende Ballposition nach verschiedenen Schüssen (verschiedene Farben) zu sehen. Der Roboter wählt mithilfe dieses Werkzeugs den passenden Schuss für eine bestimmte Richtung, in die er schießen möchte, aus.*

3.4.2.5 Der Ball ändert unvermittelt seine Richtung

Im 5. Fall ändert der Ball plötzlich seine Richtung, ohne mit der Bande kollidiert oder vom beobachtenden Roboter selbst geschossen worden zu sein. Die häufigsten Ursachen dafür sind ein Schuss eines anderen Roboters oder ein Abprall des Balls an einem anderen Roboter. In beiden Fällen ist es für den beobachtenden Roboter schwer, einzuschätzen, in welche Richtung sich der Ball im Weiteren bewegen wird, da eine präzise Mitspielermodellierung aufgrund der komplexen Roboterform und der geringen Kameraauflösung in der Sony Liga schwierig zu realisieren ist. In [7] werden solche Fälle stochastisch modelliert. Dabei hat sich die Annahme als sinnvoll herausgestellt, dass es in 5 bis 10 Prozent aller Spielsituationen zu solchen unvorhergesehenen Richtungsänderungen kommt. Daher lässt man zu jedem Zeitpunkt etwa 5 bis 10 Prozent aller Ballhypothesen die Richtung sowie den Betrag der Ballgeschwindigkeit ändern. Die neue Bewegungsrichtung ist dann völlig unabhängig von der vorhergehenden. Eine andere Möglichkeit für die Voraussage eines Abpralls des Balls an einem anderen Roboter stellt die Hinderniserkennung dar. Schüsse anderer Roboter sind jedoch vorerst auch dann nicht voraussagbar.

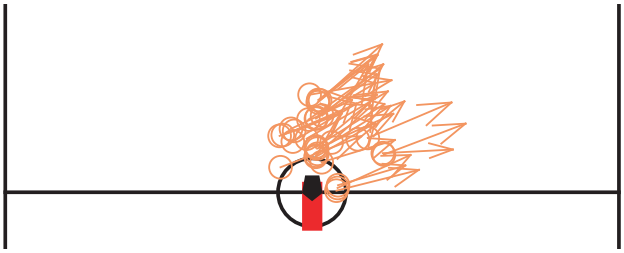


Abb. 3.9: *Der Roboter führt einen Schuss aus. Der Betrag und die Richtung aller Ballhypothesen werden unter Berücksichtigung der Varianz der Schussstärke und Richtung initialisiert.*

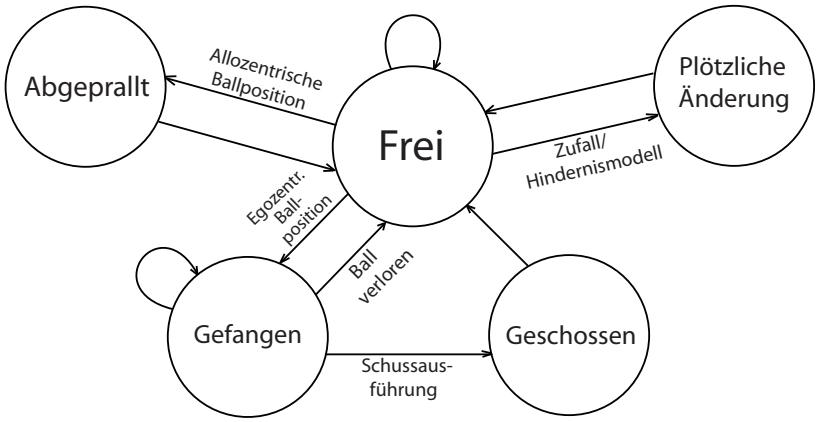


Abb. 3.10: *Graph der verschiedenen Bewegungsmodelle und der Transitionen*

In Abb. 3.11 wurden die verschiedenen Bewegungsmodelle als Zustandsdiagramm abgebildet. Anfangs befindet sich der Ball im Zustand „Frei“. Berechnet sich aus der egozentrisch modellierten Ballposition in Verbindung mit der Position des Roboters auf dem Spielfeld, dass sich der Ball außerhalb des Spielfelds befindet, erfolgt ein Übergang in den Zustand „Abgeprallt“. Dabei wird die Ballgeschwindigkeit so verändert als ob ein Abprall an der Spielfeldbegrenzung stattgefunden hätte. Danach erfolgt ein Übergang zurück in den Zustand „Frei“. Sobald der Roboter den Ball gefangen hat, findet ein Übergang in den Zustand „Gefangen“ statt. Dieser Zustand bleibt solange aktiv bis der Roboter den Ball geschossen oder verloren hat. Im Fall einer Schussausführung wechselt der Ball in den Zustand „Geschossen“ und im nächsten Zeitpunkt sofort in den Zustand „Frei“. Sollte der Ball an einem anderen Roboter abprallen oder von einem anderen Roboter geschossen werden, wechselt der Zustand des Balls in „Plötzliche Änderung“. Dabei wird die neue Ballgeschwindig-

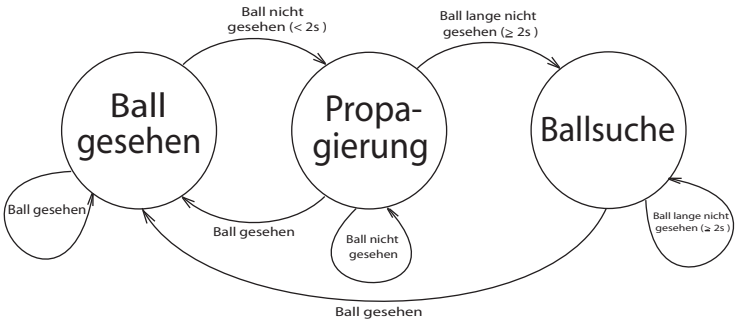


Abb. 3.11: *Zustände und Zustandsübergänge der Ballmodellierung. Bei kontinuierlichem Sensordateneingang erfolgt die Modellierung des Ballzustands über kalmangefilterte Rao-Blackwellilzed Partikel (Zustand links). Bei Ausbleiben der Sensorinformation wird der Ballzustand propagiert (Mitte). Bei lange nicht gesehenem Ball werden Negativinformationen zur Ballsuche verwendet (rechts).*

keit zufällig initialisiert. Im nächsten Schritt erfolgt wieder der Rücksprung in den Zustand „Frei“.

3.4.3 Zur Nutzung von Negativinformationen

Die bisherigen Verfahren sind für die Modellierung bewegter Objekte bei kontinuierlichem Sensordateneingang konzipiert. Es gibt jedoch auch Situationen, in denen Sensordaten für kurze oder längere Zeit ausbleiben. In Abb. 3.11 wird ein Zustandsdiagramm für die Modellierung bei ausbleibenden Sensorinformationen schematisch dargestellt. Die folgenden Ausführungen sollen derartige Modellierungsansätze für kurzzeitig ausbleibende Sensorinformationen und für die Behandlung von Negativinformationen detailliert vorstellen.

3.4.3.1 Modellierung eines gesehenen Balls

In Situationen, in denen der Ball für den Roboter sichtbar ist, kann der Roboter mithilfe des RBPF Ort und Geschwindigkeit des Balls berechnen. Jeder Partikel des RBPF besitzt einen Kalmanfilter, um Ort und Geschwindigkeit bei gegebenem Bewegungsmodell und gegebener Roboterposition zu berechnen. Die Berechnung der Ballposition s_t zur Zeit t erfolgt relativ zum Roboter, d.h. egozentrisch. Der Betrag der Ballgeschwindigkeit v_t wird relativ zum Spielfeld und die Ballrichtung relativ zum Roboterkoordinatensystem modelliert. Der Roboter verwendet zur Be-

stimmung der Ballgeschwindigkeit auf dem Spielfeld sogenannte Odometriedaten, die ihm mitteilen, wie schnell er sich gerade bewegt. Durch diese Art Modellierung ist die Ballzustandsbestimmung robuster gegenüber Fehlern in der Selbstlokalisierung, als wenn der Kalmanfilter auf der Grundlage von globalen Koordinaten arbeiten würde. Die Aktualisierung der Ballposition und -geschwindigkeit erfolgt bei jedem Schritt, in dem Sensordaten mit Ballinformationen eintreffen.

3.4.3.2 Modellierung bei kurzzeitig nicht gesehenem Ball

Es gibt Situationen, in denen der Roboter den Ball für kurze Zeit aus seinem Blickfeld verliert. Meistens tritt dieser Fall dann ein, wenn der Roboter nach Landmarken für die Orientierung Ausschau hält. Dieses Abscannen der Umgebung dauert jedoch selten länger als zwei Sekunden. Danach blickt der Roboter wieder auf die Stelle, an der der Ball vermutet wird. Innerhalb dieser Zeit propagiert die Ballmodellierung die Ballposition und -geschwindigkeit unter Berücksichtigung der Eigenbewegung des Roboters weiter, sodass der Roboter nicht mehr zu der Stelle blickt, an der er den Ball zuletzt gesehen hat, sondern auf die Stelle, zu der sich der Ball bewegt haben müsste. Für die Modellierung im Kalmanfilter bedeutet eine fehlende Sensorinformation, dass der Aktualisierungsschritt entfällt und die Kovarianzen mit fortlaufender Zeit zunehmen. Auch Bandenabpraller, ausgeführte Schüsse sowie alle restlichen Bewegungsmodelle werden im Fall eines nicht gesehenen Balls weitermodelliert. Allerdings kann keine Gewichtung der Partikel mittels Sensordaten erfolgen.

3.4.3.3 Modellierung bei lange nicht gesehenem Ball

Wird der Ball selbst nach zwei Sekunden nicht wiedergefunden, gilt er als verloren und der Roboter fängt an, Suchbewegungen auszuführen. Für die Modellierung bedeutet dies, vom egozentrischen auf ein allozentrisches Ballmodell umzuschalten. Das hat den Vorteil, dass Modellierungsfehler für Ballpartikel, die durch eine ungenaue Odometrie entstehen und die nach relativ kurzer Zeit sehr groß werden, ausgeschlossen werden können. Die anstelle dessen auftretenden Selbstlokalisierungsfehler fallen nicht so schwer ins Gewicht. Dafür kann der Roboter das Feld systematisch absuchen und sogar eine Heuristik[9] einführen, anhand derer er abschätzt, welche Stellen sich eher abzusuchen lohnen als andere. Die letzte beobachtete Ballgeschwindigkeit und -position wird in das allozentrische Modell übertragen und zusätzlich werden

[9] griech., von heuriskein, „finden“ bzw. „erfinden“

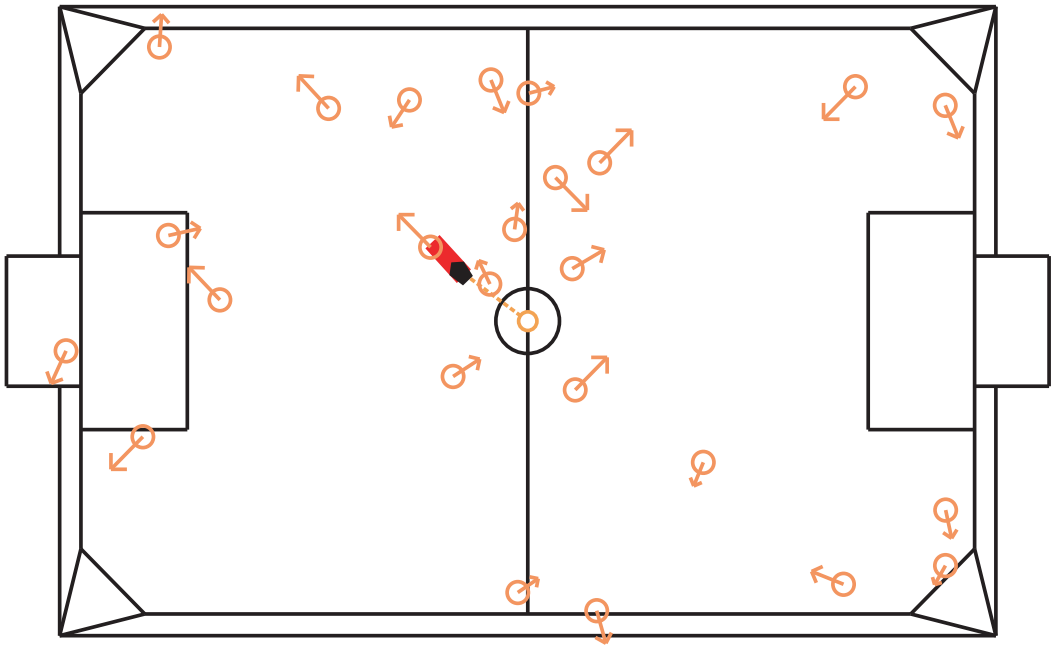


Abb. 3.12: *Propagierung der Ballhypothesen. Wenn der Ball lange nicht gesehen wurde, verteilen sich die Hypothesen gleichmäßig auf dem Spielfeld, die Pfeile geben sowohl die Richtung als auch den Betrag der Geschwindigkeiten der Hypothesen an.*

die Positionen der Ballhypothesen möglichst gleichmäßig über das Spielfeld verteilt (Abb. 3.12).

Der Roboter sucht bei dieser Modellierung zunächst die Umgebung ab, in der der Ball vermutet wurde. Alle Ballhypothesen i, die sich zum Zeitpunkt t im Sichtbereich des Roboters befanden, und in dem kein Ball war, bekommen eine sehr geringe Gewichtung ω_t^i. Die Gewichtungen aller Hypothesen, die sich außerhalb des Sichtbereichs befanden, bleiben unverändert, da über sie keine neuen Informationen vorliegen. Nach der Berechnung der Gewichtungen ω_t^i werden die Hypothesen je nach Höhe der Gewichtungen neu erzeugt. Auf diese Art kann der Roboter die Bereiche seiner direkten Umgebung systematisch absuchen. Die Sichtinformation wird außerdem dahingehend untersucht, ob Hindernisse im Blickfeld des Roboters waren oder nicht [26, 27]. Damit kann der Roboter beurteilen, inwieweit sein Blickfeld durch Hindernisse eingeschränkt wurde. Wird der Ball wiedergefunden, startet der Kalmanfilter neu und der Roboter verwendet die egozentrische Modellierung mit den Bewegungsmodellen. Die allozentrisch modellierten Hypothesen werden verworfen.

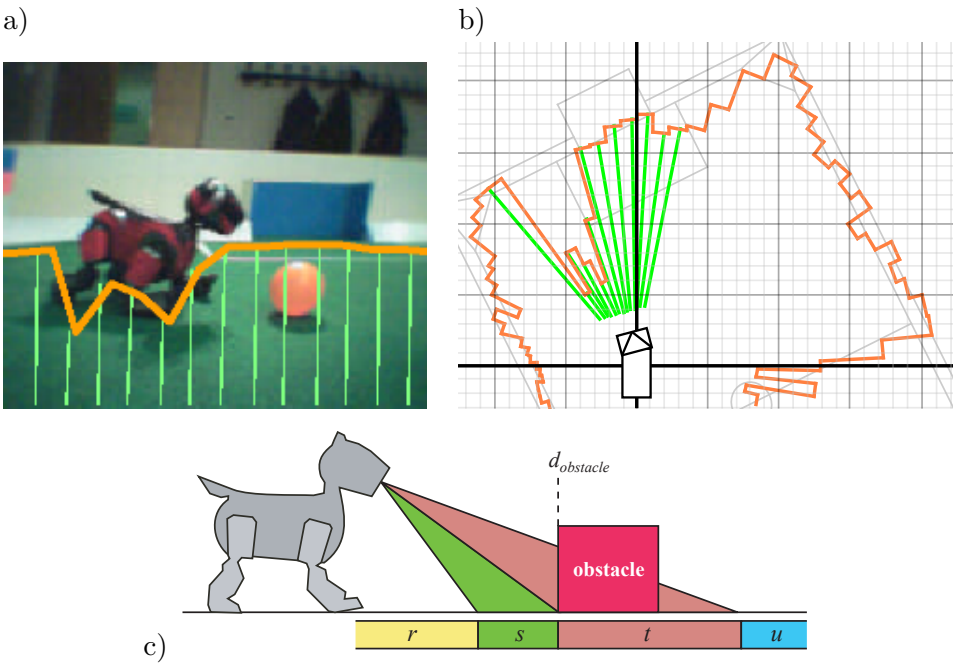


Abb. 3.13: *a) freier Sichtbereich ins Kamerabild des Roboters projiziert – der orange Ball zählt nicht zu den Hindernissen, b) die grünen Linien geben Auskunft über freie Bereiche im Sichtbereich des Roboters, c) die verschiedenen zu beachtenden Bereiche bei der Hinderniserkennung aus Bildinformationen*

3.4.3.4 Zur Funktionsweise der Hinderniserkennung

Die Hinderniserkennung wurde entwickelt, um andere Roboter oder Gegenstände zu erkennen und beim Bewegen über das Spielfeld zu umlaufen, d.h. Kollisionen auf dem Spielfeld zu vermeiden. Im Ergebnis können damit Kollisionen sehr effizient vermieden werden. In Verbindung mit der Suche nach dem Ball ist dadurch detektierbar, welche Bereiche in der Umgebung des Roboters sichtbar, bzw. welche Bereiche durch Hindernisse verdeckt sind (Abb. 3.13 a).

Für die Ballmodellierung bedeutet dies, dass nur Ballhypothesen als abgesucht gelten, die nicht von Hindernissen verdeckt werden, also solche, die sich im Sichtbereich der Kamera befinden. Umgekehrt werden verdeckte Ballhypothesen nicht aktualisiert, weil sie nicht gesehen werden können, wie Abb. 3.14 zeigt.

Im Detail überprüft der Roboter in seinem Kamerabild mit Hilfe von Scanlines (Abb. 3.13 b) die grünen Bereiche im Kamerabild. In Abb. 3.13 c ist erkennbar, dass der Roboter aufgrund des Kamerabildrands beim Geradeausblick nicht beliebig nahe Gegenstände erfassen kann. Dieser Nahbereich kann aber wiederum über

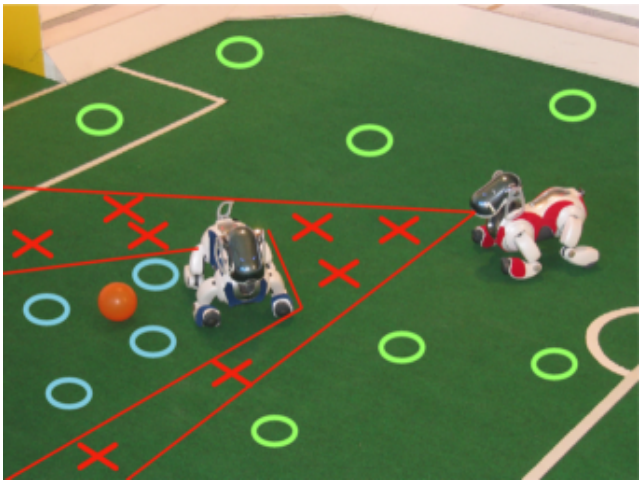


Abb. 3.14: *Beispielszene beim Suchen des Balls durch den rechten Roboter. Es werden nur diejenigen Hypothesen gelöscht (Kreuze) und an anderer Stelle neu erzeugt, die sich im direkten Sichtbereich der Kamera befinden. Hypothesen (blau) des Bereiches hinter dem linken Roboter, in dem sich der Ball befindet, werden nicht verändert, genauso wie die Hypothesen (grün) außerhalb des Sichtbereiches des rechten Roboters.*

den PSD-Sensor erfasst werden. Grüne Bereiche, die sich ggf. hinter einem Hindernis befinden, sind von Bereichen vor dem Hindernis zu unterscheiden. Zur Unterscheidung wird angenommen, dass lediglich die grünen Bereiche vor dem Hindernis am unteren Bildrand anfangen. Wird in einem von Hindernissen freien Bereich, in dem eine Ballhypothese modelliert wurde, kein Ball gefunden, so wird diese Hypothese gelöscht und an anderer Stelle, außerhalb des sichtbaren Bereichs, neu generiert (Abb. 3.15). Als Resultat erhält man durch dieses einfache Verfahren eine Hypothesenanhäufung in Bereichen, die lange nicht mehr auf das Vorhandensein des Balls überprüft wurden.

3.4.4 Bestimmung der Ballposition anhand der Partikelverteilung

Eine weit verbreitete Gruppe von Methoden zur Bestimmung der Objektposition aus der Partikelverteilung stellen Clusterverfahren dar. Durch Clusterbildung können verschiedene Bereiche des Hypothesenraums mit einer hohen Repräsentantenzahl gefunden und gegeneinander abgewogen werden. Das Zentrum des am höchsten bewerteten Clusters wird meist als die geschätzte Objektposition betrachtet. Die Bewertung eines Clusters setzt sich aus Anzahl bzw. Gewichtung der Partikel zu-

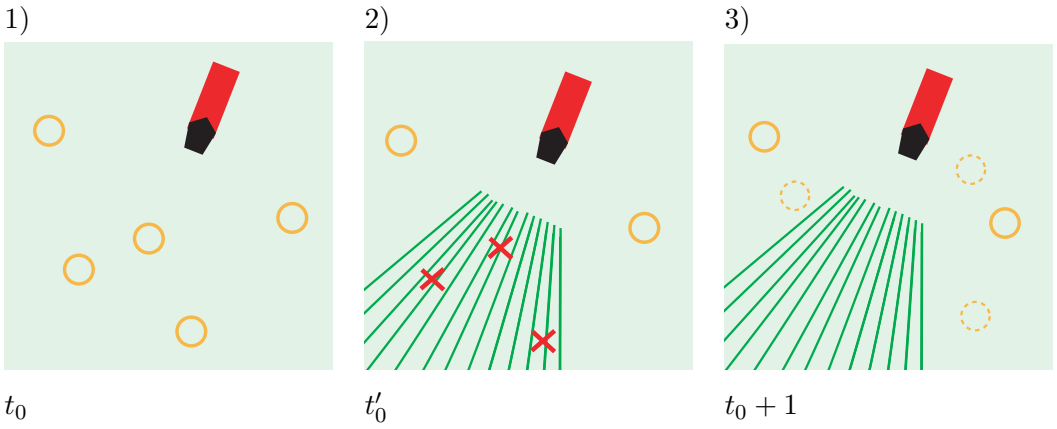


Abb. 3.15: *Darstellung des Hypothesenupdates mit Hilfe von Sichtinformation:*
1) Hypothesenverteilung bei Ausführungsbeginn
2) Herauslöschen der Hypothesen im Sichtbereich
3) Neugenerierung von Hypothesen außerhalb des Sichtbereichs

sammen. Die Clusterbildung kann auf verschiedene Weisen erfolgen. In Abb. 3.16 werden zwei verbreitete Ansätze aufgezeigt.

Zum einen kann der Hypothesenraum in Raster aufgeteilt werden [40]. Jedes Raster wird für sich bewertet und aneinander grenzende Raster, die einen bestimmten Schwellenwert für ihre Bewertung überschreiten, zusammengefasst.

Ein anderes Verfahren stellt die agglomerative[10] Clusterbildung [37, 38, 39] dar, welche ohne Rasterung auskommt. Für die Ballpositionsbestimmung wurde ein agglomeratives Clusterverfahren verwendet, deren Arbeitsweise weiter unten beschrieben wird.

3.5 Aufmerksamkeitssteuerung

Die Aufmerksamkeitssteuerung dient in diesem Kontext einem systematischen Suchverhalten des Roboters nach dem Ball.
Im Weiteren soll gezeigt werden, wie eine Aufmerksamkeitssteuerung als Anwendung für das Hypothesenmodell bei nicht gesehenem Ball aussehen könnte. Das Ziel dabei ist es, möglichst schnell die Bereiche der verschiedenen, unter Umständen auch weit voneinander entfernt liegenden Ballhypothesen, visuell abzusuchen. Damit wird es

[10] glomus = Knäuel; Cluster werden vereinigt. Gegenteil: divisiv; Cluster werden zerlegt.

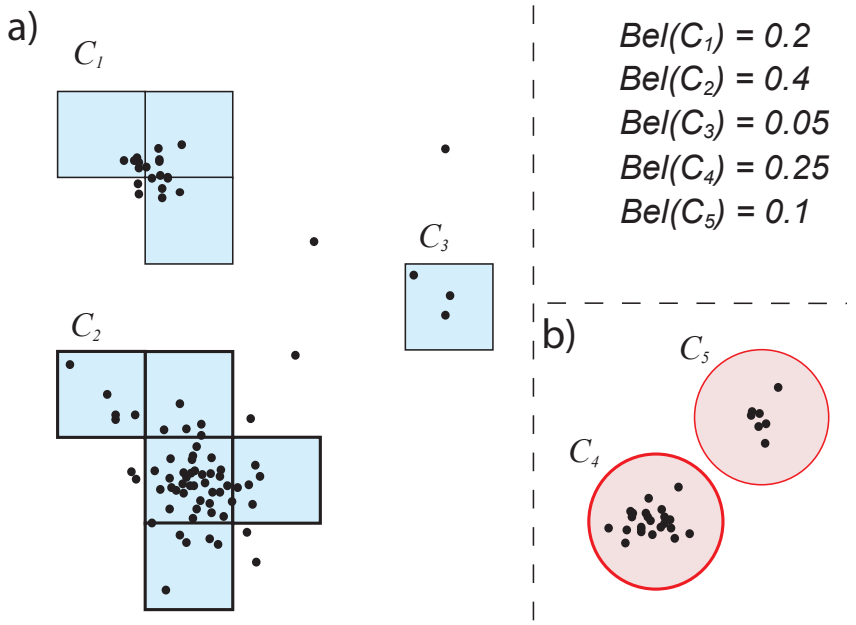


Abb. 3.16: *Zwei häufig angewandte Clusterverfahren zur Zustandsabschätzung anhand der Partikelverteilung. a) rasterbasiertes Clusterverfahren. Die Partikel eines jeden Rasters werden gezählt. Cluster, die aneinander grenzen und eine Mindestanzahl von Partikeln enthalten, werden vereinigt und erhalten eine Gesamtwahrscheinlichkeit Bel(C).*
b) agglomeratives Clusterverfahren, rasterfrei. Unter der Voraussetzung, dass die Partikel bestimmte Abstände zueinander nicht überschreiten, werden sie zu Clustern zusammengefasst.

möglich, den Informationsgewinn pro Zeiteinheit zu erhöhen sowie den Ball möglichst schnell wiederzufinden. Es soll dabei eine einfache, aber effiziente Heuristik[11] für ein systematisches Suchverhalten vorgestellt werden.

3.5.1 Problemstellung

Ein Problem bei der Erstellung einer Aufmerksamkeitssteuerung ist die Handlungsstabilität. Der Roboter soll einerseits einmal gefasste Entschlüsse nicht voreilig wieder aufgeben, andererseits soll er flexibel genug bleiben, um auf Änderungen der Umwelt reagieren zu können. Wenn der Roboter einen bestimmten Bereich nach dem Ball abgesucht hat, so ist es wünschenswert, dass er danach Gebiete absucht, die in seiner Nähe sind, um Zeit zu sparen. Eine Entfernungskomponente soll daher

[11] Heuristik bezeichnet hier eine Schätzfunktion für den Informationsgewinn

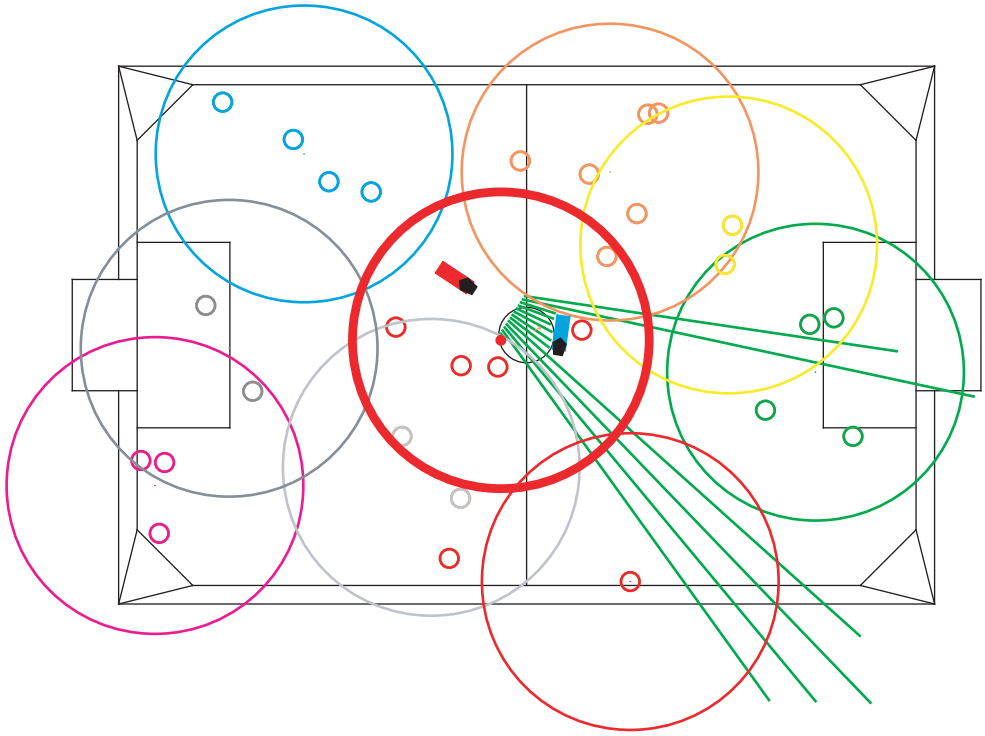


Abb. 3.17: *Schwerpunktclusterung von Ballhypothesen - Clusterabstand 800 mm; Schwerpunktmethode, Beginn der Suche. Die grünen, vom Roboter (rot) ausgehenden Linien repräsentieren freie Bereiche vor dem Roboter, Hypothesen in diesem Bereich sind direkt verifizierbar, also nicht durch Hindernisse, wie z.B. durch den Roboter (blau), verdeckt.*

Bestandteil der Heuristik sein. Aber auch die Anzahl der vermuteten Hypothesen ist von entscheidender Bedeutung bei der Auswahl von Gebieten, die als Nächstes abgesucht werden sollen.

3.5.2 Ansatz zur Bestimmung der Suchkriterien

Die Hypothesen repräsentieren die Aufenthaltswahrscheinlichkeit des Balls auf dem Spielfeld und stellen damit ein wichtiges Kriterium für das Suchverhalten dar. Eine hohe Hypothesendichte deutet auf eine hohe Aufenthaltswahrscheinlichkeit des Balls hin. Um Bereiche mit vielen Hypothesen zu ermitteln, werden die Ballhypothesen durch das agglomerative Clusterverfahren zu Einheiten zusammengefasst.

Anfangs stellt jede Ballhypothese jeweils eine ein-elementige Clustermenge dar. Bei der Schwerpunkt-Clustermethode werden ein Abstandsmaß sowie ein Schwellenwert vorgegeben. Das ist hier der euklidische Abstand, der kleiner als 100 mm

sein soll. Jeweils zwei Cluster, deren Schwerpunkte weniger als 100 mm voneinander entfernt sind, werden zu einem neuen Cluster mit neuem Schwerpunkt zusammengefasst. Nachdem der Schwellenwert von 100 mm für alle Clusterpaare überprüft wurde, beginnt der Durchgang von Neuem mit einem Abstandsmaß von 200 mm. Dies wurde im Beispiel bis zu einem Abstand von 800 mm durchgeführt. Das Ergebnis ist in Abb. 3.17 dargestellt.
Single-Linkage- und Complete-Linkage-Verfahren sind Alternativen zur Schwerpunktmethode. Beim Single-Linkage Verfahren wird der Abstand zwischen zwei Clustern anders berechnet. Um zu ermitteln, ob zwei Cluster C_i und C_j fusioniert werden können, wird das Minimum d_{min} der euklidischen Abstände $d(el_{i_k}, el_{j_l})$ aller Elementpaare $< el_{i_k}, el_{j_l} >$ aus diesen beiden Clustern ermittelt:

$$d_{min} = \min\{d(el_{i_k}, el_{j_l}) | el_{i_k} \in C_i; el_{j_l} \in C_j; 1 \leq k \leq card(C_i), 1 \leq l \leq card(C_j), \} \tag{3.26}$$

Dieses Minimum wird mit einem Schwellenwert verglichen. Falls das Minimum größer als der Schwellenwert ist, können beide Cluster fusioniert werden. Das Complete-Linkage Verfahren verfährt analog dazu, mit dem Unterschied, dass das Maximum d_{max} aller Elementpaare gebildet wird:

$$d_{max} = \max\{d(el_{i_k}, el_{j_l}) | el_{i_k} \in C_i; el_{j_l} \in C_j; 1 \leq k \leq card(C_i), 1 \leq l \leq card(C_j), \} \tag{3.27}$$

Im Gegensatz zur Schwerpunktmethode benötigen diese Verfahren jedoch immer eine große Menge von Vergleichen. Bei der Schwerpunktmethode reicht es, die Abstände der Schwerpunkte zweier Clustermengen zu vergleichen. Ein Nachteil des Single-Linkage Verfahrens ist zudem, dass es zu sogenannten Kettenbildungen kommt, wenn viele Cluster mit einem relativ geringen Abstand zueinander eine lange Reihe bilden. Dies kann dazu führen, dass der Maximalabstand verschiedener Individuen innerhalb dieses Clusters sehr groß wird. Praktisch bedeutet dies, dass die Cluster beim Single-Linkage Verfahren für den Roboter unüberschaubar werden können.

3.5.3 Ausführung der Suche

Mit der Schwerpunktmethode im Rahmen eines Clusterverfahrens ist es möglich, Gebiete mit Ballhypothesen zu Einheiten zusammenzufassen und je nach Individu-

enzahl zu bewerten. Im Test wurde der Cluster mit den meisten Individuen selektiert und als Suchziel ausgewählt. Der Roboter bewegte sich nun auf den Schwerpunkt dieses Clusters zu. Vom Roboter abgesuchte Hypothesen werden aus dem Cluster gelöscht und an anderer Stelle neu generiert, falls kein Ball gesehen wurde. Die Bewertung von anderen Clustern steigt dabei an, weil gelöschte Hypothesen mit hoher Wahrscheinlichkeit in Bereichen bereits bestehender Cluster neu erzeugt werden. Abbruchkriterium für das Absuchen eines Clusters kann der Vergleich der Bewertung des ausgewählten mit der Bewertung von anderen Clustern darstellen. Neues Suchziel ist dabei immer der am höchsten bewertete Cluster. Um Oszillationen der Clusterauswahl bei vielen ähnlich hoch bewerteten Clustern zu vermeiden, kann dabei ein Hysteresismaß eingeführt werden. Im hier dargestellten Beispiel sollte der Roboter den Schwerpunkt des selektierten Clusters so lange ansteuern, bis es einen Cluster mit einer doppelt so hohen Bewertung wie der aktuellen gibt. Da der Roboter die Hypothesen, die er bereits abgesucht hat, sofort löscht und an anderer Stelle neu generiert, wird die Bewertung des selektierten Clusters mit fortschreitender Zeit immer kleiner.

Kombinieren lässt sich diese Aufmerksamkeitssteuerung mit statischem Suchverhalten. Denkbar ist ein Verhalten, bei dem sich der Roboter anfangs einmal komplett im Kreis dreht, um die direkte Umgebung abzusuchen und danach Bereiche ansteuert, die ggf. durch Sichthindernisse verdeckt waren. Ein verbessertes Resultat beim Suchverhalten kann erreicht werden, wenn für die Bewertung eines Clusters zusätzlich zur Hypothesenzahl n der euklidische Abstand r des Roboters zum Clusterschwerpunkt sowie der Winkel φ zwischen der Blickrichtung des Roboters und der Richtung zum Clusterschwerpunkt herangezogen werden. Der Roboter wird daraufhin eher Cluster in seiner Umgebung absuchen, auch wenn die Anzahl der Hypothesen in diesem Cluster nicht maximal ist. Bei gegebener Hypothesenzahl n_i eines Clusters C_i, dem Abstand r_i vom Cluster zum Roboter und einem Winkel von φ_i lautet die Berechnungsvorschrift für die Clusterbewertung val_{C_i} damit wie folgt:

$$val_{C_i} = \frac{n_i}{(c_r + r_i)(c_\varphi + \varphi_i)} \tag{3.28}$$

Die Variablen c_r sowie c_φ bestimmen, inwieweit der Abstand r_i bzw. der Winkel φ_i in die Bewertung val_{C_i} des Clusters C_i eingehen.

3.6 Zusammenfassung

Aus aktueller Sicht und bisherigen Erfahrungen lässt sich feststellen, dass der Kalmanfilter optimal für die Modellierung linearer Prozesse ist. Um jedoch Situationen modellieren zu können, in denen das zu betrachtende Objekt sowohl mit dem Beobachter als auch mit der Umgebung interagiert, wurde die Erweiterung des Kalmanfilters um ein Partikelmodell notwendig. Deshalb wurde ein Rao-Blackwellized Partikelfilter für die RoboCup Domäne vorgestellt, der sowohl die Interaktionen der Roboter mit dem Ball, als auch Interaktionen des Balls mit seiner Umgebung modellieren kann. Auf der Grundlage von Informationen über Hindernisse im Sichtbereich des Roboters wurde zusätzlich ein Verfahren zur Verwendung von Negativinformationen präsentiert, das zur Aufmerksamkeitssteuerung, z.B. zur Ballsuche, verwendet werden kann. Diese Kombination stellt zurzeit ein gewisses Optimum in der Sony-Liga dar und führte sowohl bei Tests als auch bei den anwendenden Teams im Rahmen des RoboCup 2004 zu guten bis sehr guten Ergebnissen.

Kapitel 4

Implementierung und Experimente

Die nachfolgenden Ausführungen dienen vorrangig der Evaluierung des im Kapitel 3 vorgestellten Rao-Blackwellized Partikelfilters. Ziel ist in diesem Zusammenhang die Abgrenzung des RBPF von einem reinen Kalmanfilter bzw. gegenüber einer nicht vorhandenen Filterung auf Basis von Testergebnissen im Rahmen des RoboCup. Zunächst wird ein implementierter Algorithmus des RBPF angegeben, der als Grundlage für die durchgeführten Versuche diente. Danach werden die verschiedenen Ursachen für verrauschte Sensordaten dargestellt. Anschließend erfolgt die Vorstellung einiger interessanter und für die Sony Liga repräsentativer Szenarien.

4.1 Beispielimplementierung

Der im 3. Kapitel vorgestellte Partikelfilter modelliert sowohl die Roboter- als auch die Ballposition in einem Schritt. Das ist deshalb notwendig, weil die Ballposition von der Roboterposition abhängig ist und umgekehrt. Die Roboterposition demgegenüber ist für die Ballmodellierung erforderlich, um zu erkennen, ob der Ball an der Bande abgeprallt ist oder nicht. Analog dazu wird ein Partikel mit falscher Roboterposition gelöscht, wenn sich der Ball bei dieser Position außerhalb des Feldes befinden würde. Allerdings ist diese Art von Modellierung sehr rechen- und speicherintensiv. Es werden für den Aibo-Roboter in jedem Schritt etwa 350-500 Partikel benötigt, um gute Resultate zu erzielen [7], weil für jede einzelne Roboterposition verschiedene Bewegungsmodelle repräsentiert werden müssen. Hinzu kommt der Umstand,

dass bei weitem mehr Ballperzepte zum Roboter gelangen als Landmarkenperzepte. Daher hatten die Ballperzepte weit größeren Einfluss auf die Roboterlokalisierung als die Perzepte der Landmarken, obwohl die Landmarken wesentlich wichtiger für die Roboterlokalisierung sind als der Ball [7]. Deshalb wurde das einheitliche Modell für die Ball- und Roboterposition in zwei getrennte Modelle unterteilt. Die Roboterposition hat weiterhin Einfluss auf das Ballmodell, aber Balldaten werden für die Modellierung der Roboterposition ignoriert. Die Verwendung von zwei weitgehend unabhängigen Modellen benötigt – und das ist von eminenter Bedeutung – zudem deutlich weniger Partikel und weniger Ressourcen.

Algorithm 1 RBPF-Implementierung, 1. Initialisierung

1: $S_{t-1} =< R_{t-1}, B_{t-1} >$ //*Input der vorherigen Zustände*
2: $R_{t-1} = \{r_{t-1}^{(i)} | i = 1, ..., N\}$ //*Roboterposition*
3: $B_{t-1} = \{< b_{t-1}^{(i)}, m_{t-1}^{(i)}, \rho_{t-1}^{(i)} > | i = 1, ..., L\}$ //*Ballmodell*
//$b = Ballzustand$, $m = Bewegungsmodell$, $\rho = Roboterposition$
4: u_{t-1} //*Odometriedaten*
5: z_t //*Sensordaten*
6: $R_t := 0, B_t := 0;$ //*Initialisierung*

Der implementierte RBPF arbeitet in vier Schritten. Im ersten Schritt (Alg. 1) erfolgt die Initialisierung der Ball- und Roboterpartikelmengen B_t bzw R_t. Die Ball- und Roboterpartikelmengen B_{t-1}, R_{t-1} aus dem letzten Takt müssen bekannt sein. Ein Ballpartikel ist ein 3-Tupel, welches aus dem Ballzustand b_t^i, einem Bewegungsmodell m_t^i und einer Roboterposition ρ_t^i besteht. Zusätzlich dazu hat der Roboter bereits die aktuellen Sensorinformationen z_t sowie Odometriedaten u_{t-1} erhalten.

Die Ausführung des nächsten Schritts hängt davon ab, ob in den letzten zwei Sekunden ein Ball gesehen wurde oder nicht. Wurde kurz zuvor ein Ball gesehen, setzt der Algorithmus bei Schritt 2 fort. Wurde kein Ball gesehen, erfolgt die Fortsetzung bei Schritt 3. Im zweiten Schritt (Alg. 2) erfolgt für jeden Ballpartikel i die Abschätzung der neuen Roboterposition ρ_t^i anhand der Odometriedaten u_{t-1} und der vorherigen Position des Roboters ρ_{t-1}^i. Danach wird das neue Bewegungsmodell m_t^i abgeschätzt. Mithilfe der Roboterposition und dem Bewegungsmodell kann der Kalmanfilter nun die neue Ballposition und im Anschluss daran die Gewichtungen ω_t^i für jeden Partikel i bestimmen. Liegen keine Sensordaten vor, propagiert der Kalmanfilter die Ballpositionen und alle Gewichtungen werden auf den gleichen Wert gesetzt. Am Ende von Schritt zwei erfolgt die Erzeugung einer neuen Partikelmenge B_t aus den soeben modellierten Partikeln b_t^i und deren Gewichtungen ω_t^i.

Algorithm 2 RBPF-Implementierung, 2. Ball gesehen

1: **if** [letztes Ballperzept z_{last} nicht älter als zwei Sekunden] **then**
2: **for** $i := 1, ..., L$ **do**
3: Sample $\rho_t^{(i)} \sim p(\rho_t|\rho_{t-1}, u_{t-1})$ *//Roboterposition odometrieabgeschätzt*
4: Sample $m_t^{(i)} \sim p(m_t|m_{t-1}, \rho_t^{(i)}, b_{t-1}^{(i)}, u_{t-1})$ *//Bewegungsmodell abgeschätzt*
5: $b_t^{(i)} := Kalman\_update(\rho_t^{(i)}, m_t^{(i)}, b_{t-1}^{(i)}, z_t)$ *//Ballmodell aktualisiert*
6: $B_t := B_t \cup \{< b_t^{(i)}, m_t^{(i)}, \rho_t^{(i)} >\}$ *//Partikel zu B_t hinzufügen*
7: $\omega_t^{(i)} := p(z_t|m_t^{(i)}, \rho_t^{(i)}, b_t^{(i)})$ *//Gewichtungen für die //Ballsamples berechnen*
8: Resampling von B_t mithilfe der Gewichtungen ω_t
9: **end for**
10: **end if**

Algorithm 3 RBPF-Implementierung, 3. Ball nicht gesehen - Suche

1: **if** [letztes Ballperzept z_{last} älter als zwei Sekunden] **then**
2: **for** $i := 1, ..., L$ **do**
3: **if** [Ballhypothese $b_t^{(i)}$ sichtbar] **then**
4: $\omega_t^{(i)} := 0$ *//Ballhypothese wurde abgesucht, //kein Ball gefunden, Gewichtung null*
5: **else**
6: $\omega_t^{(i)} := \omega_{t-1}^{(i)}$ *//Ballhypothese nicht sichtbar, //Gewichtung unverändert*
7: **end if**
8: **end for**
9: Resampling von B_t mithilfe der Gewichtungen ω_t
10: **end if**

Der dritte Schritt wird immer dann ausgeführt, wenn der Ball lange nicht gesehen wurde. Der Roboter überprüft für jeden Partikel b_t^i, ob er sich im Sichtbereich des Roboters befindet. Dabei achtet er darauf, ob Hindernisse im Weg stehen und dadurch bestimmte Bereiche verdeckt sind. Befindet sich ein Partikel im direkten Sichtbereich des Roboters, so erhält dieser Partikel eine sehr kleine Gewichtung, vorzugsweise von null. Ballhypothesen, die sich nicht im Sichtbereich des Roboters befinden, behalten ihre bisherigen Gewichtungen. Damit ist die Modellierung der Ballpartikel beendet und die Modellierung der Roboterposition beginnt.

Algorithm 4 RBPF-Implementierung, 4. Roboterposition modellieren

1: **for** $i := 1, ..., N$ **do**
2: Sample $r_t^{(i)} \sim p(r_t | r_{t-1}^{(i)}, u_{t-1})$ *//Roboterposition odometrieabgeschätzt*
3: $R_t := R_t \cup \{r_t^{(i)}\}$ *//Partikel zu R_t hinzufügen*
4: **end for**

5: **if** [z_t ist eine Landmarke] **then**
6: **for** $i := 1, ..., N$ **do**
7: $\omega_t^{(i)} := p(z_t | r_t^{(i)})$ *//Berechne Gewichtungen*
//für die Robotersamples
8: Resample R_t
9: **end for**
10: Berechne neue Roboterposition R und Varianz σ_R

11: **for** $i := 1, ..., L$ **do**
12: Aktualisiere $< b_t^{(i)}, m_t^{(i)}, \rho_t^{(i)} >$ mit Roboterposition R und σ_R
13: **end for**
14: **end if**

15: return $S_t =< R_t, B_t >$

Im vierten und letzten Schritt erfolgt die Modellierung der Roboterposition. Dazu werden die Zustände der neuen Partikel r_t aus den alten Partikeln r_{t-1} und der Odometrie u_{t-1} berechnet. Wurde außerdem eine Landmarke gesehen, so erfolgt eine Neugewichtung $\omega_t^{(i)}$ der Partikel entsprechend ihrer Übereinstimmung mit den Sensordaten z_t sowie eine Neugenerierung der Partikel anhand ihrer Gewichtung $\omega_t^{(i)}$. Die Partikel der Ballmodellierung werden zum Schluss mit der neuen Roboterposition aktualisiert.

Nach diesen Darlegungen zur Implementation des RBPF, möchte der Autor nunmehr auf das Problem auftretender Sensorfehler eingehen.

4.2 Das Fehlermodell des Ballperzepts

Der Roboter berechnet die Position von Objekten aus Bilddaten sowie aus den Daten über seine Gelenkwinkel. Anhand der Stellung der Kopfgelenke weiß der Roboter, aus welcher Richtung er das Bild aufgenommen hat. Mit den Daten über seine Beingelenke sowie durch den Neigungssensor kann der Roboter einen künstlichen Horizont berechnen, der ihm hilft, die Entfernung zu gesehenen Objekten zu bestimmen. Die detaillierte Beschreibung der Entfernungsberechnung befindet sich im Anhang A. Im Folgenden sollen nur die wichtigsten Fehlerarten bei der Berechnung von Ballperzepten vorgestellt werden.

4.2.1 Farbfehlklassifikation und Bildverzerrungen

Fehler in der Bildverarbeitung können durch Farbfehlklassifikationen entstehen wie Abb. 4.1 f zeigt. Verzerrungen, verursacht durch die Optik, können vorher gemessen und durch geeignete Verfahren herausgerechnet werden. Ein anderes Phänomen, das gemeinhin als Bildschiefe bezeichnet wird, tritt durch die Bewegung der Roboterkamera relativ zu den Bildobjekten auf. Der obere Teil des Bildes wird zeitlich deutlich früher abgetastet als der untere. Im Ergebnis staucht eine sich nach unten bewegende Kamera das Bild, eine sich nach oben bewegende Kamera streckt es. Ein laufender Roboter sieht deshalb den Ball eher oval als rund (Abb. 4.1 g, h). Das erschwert die genaue Positionsbestimmung, insbesondere wenn versucht wird, einen Kreis in den im Bild als Ellipse erscheinenden Ball hinein zu interpretieren.

4.2.2 Entfernungsfehler wegen begrenzter Kameraauflösung

Als Einschränkung für die Objektgrößenbestimmung erweist sich die Tatsache, dass das Kamerabild gerastert ist und zudem eine geringe Auflösung besitzt. Die Entfernung zu sehr weit entfernten Bällen kann nicht mehr differenziert genug bestimmt werden (Abb. 4.3 a), und es kommt zu Messfehlern (b). Die Ballränder im Bild sind zudem nicht orange – sondern eine Mischung aus den Farben des Hintergrunds und orange – und können daher in die Messung der Objektgröße nicht mit einbezogen werden. Fallen Ballrand und Pixelrand zusammen, kann es außerdem zu Oszillationen des gemessenen Ballrandes kommen und demzufolge zu Oszillationen in der gemessenen Ballentfernung. Ein Ball, der n orange Pixel breit ist, kann demnach auch fast $n + 2$ Pixel breit sein, wenn sich an zwei Ballrändern der Messfehler aufaddiert (Abb. 4.2 a).

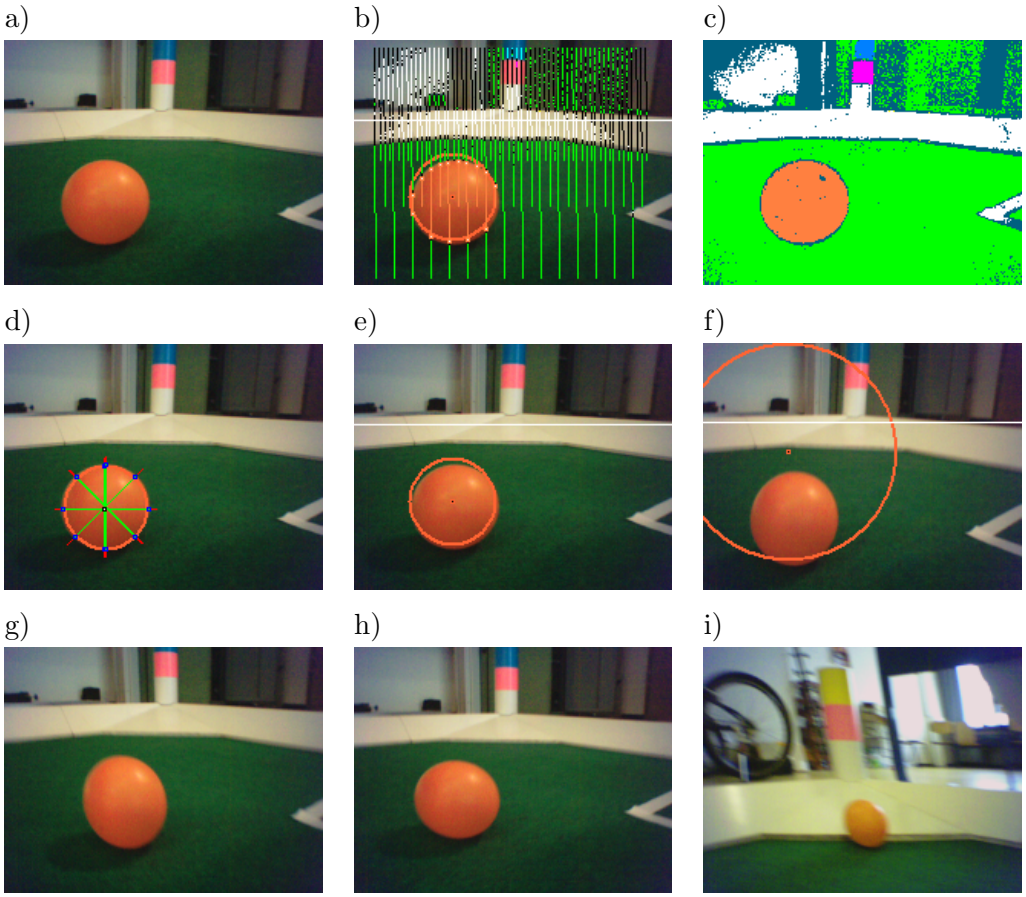


Abb. 4.1: *ERS7, Ballerfassung: a) Kamerabild, b) Scanlines, c) Farbklassifikation, d) Ballagent, e) korrekt gefundenes Ballperzept, f) fehlerhaftes Ballperzept aufgrund von Farbfehlklassifikation*
Bildschiefe: g) Bild gestreckt, weil sich Kamera nach oben bewegt, h) Bild gestaucht, wenn sich Kamera während der Aufnahme nach unten bewegt, i) seitliche Verzerrung bei sich seitlich bewegendem Kopf.

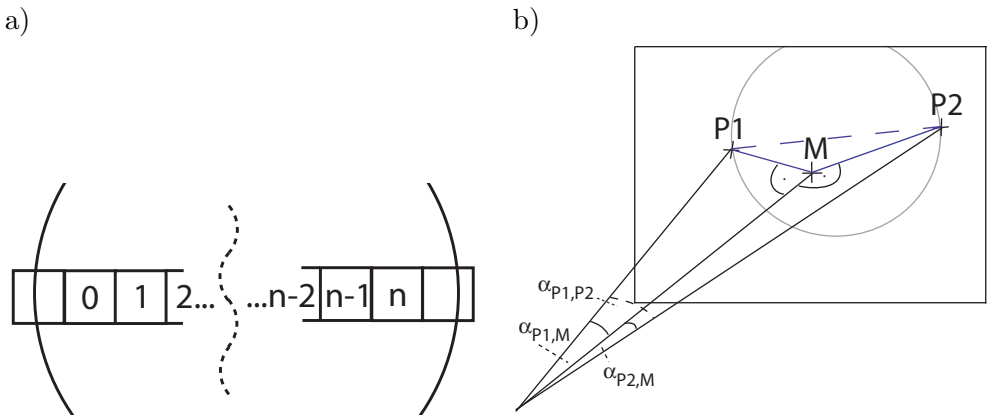
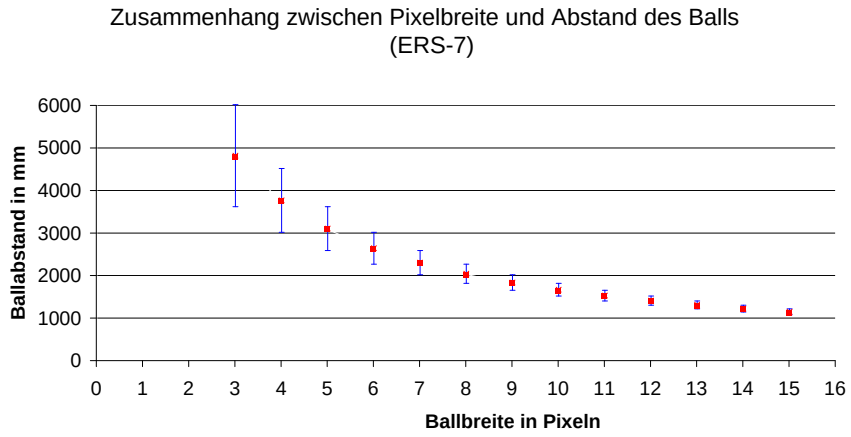


Abb. 4.2: *a) Pixelbreite des Balls, Randpixel werden bei der Messung aufgrund ihres Farbwertes nicht berücksichtigt.*
b) Raumwinkelberechnung für zwei Bildpunkte. Bei zwei gegebenen Randpixeln P_1 und P_2 eines Objekts berechnet sich der Raumwinkel $\alpha_{P1,P2}$ zwischen diesen beiden Pixeln aus dem Raumwinkel $\alpha_{M,P1}$, der zwischen P_1 und dem Bildmittelpunkt M sowie aus $\alpha_{M,P2}$, der zwischen P_2 und dem Bildmittelpunkt aufgespannt wird.

Im Mittel wird er jedoch $n + 1$ Pixel breit sein. Diese Überlegung ging auch in Abb. 4.3 a mit ein. Ein und dieselbe Ballentfernung kann sich im Bild durch mehrere Pixelausdehnungen widerspiegeln. Durch Betrachtung mehrerer Durchmesser und deren Mittelung, resultierend aus mehreren Randpunkten, kann die zu erwartende Genauigkeit für die Ballentfernungsbestimmung dennoch erhöht werden.

Weit entfernte Bälle können auch bei der peilungsbasierten Entfernungsmessung nicht exakt bestimmt werden, da der Peilungswinkel zwischen Ball und künstlichem Horizont dann nahe null und nicht mehr differenziert genug messbar ist wie Abb. 4.4 zeigt. Demgegenüber können Entfernungen zu verdeckten Bällen recht gut bestimmt werden, da der Rand des Balls nicht unbedingt erkannt werden muss. Der Nachteil dieses Verfahrens besteht jedoch darin, dass der Horizont nicht aus Bilddaten ermittelt werden kann und deshalb aus der Körperneigung und Kopfhaltung errechnet werden muss. Diese Berechnung ist jedoch nicht sehr exakt und wird oft durch Laufbewegungen verrauscht. In der Folge wird der Ball bezüglich seiner wahren Position manchmal näher und manchmal weiter weg erkannt. Dieses Verfahren sollte deshalb nur für nahe Bälle und für Bälle, deren Ränder zum großen Teil verdeckt sind, verwendet werden.

a)



b)

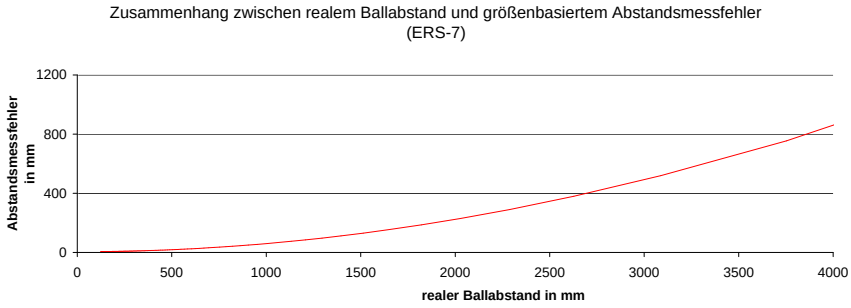
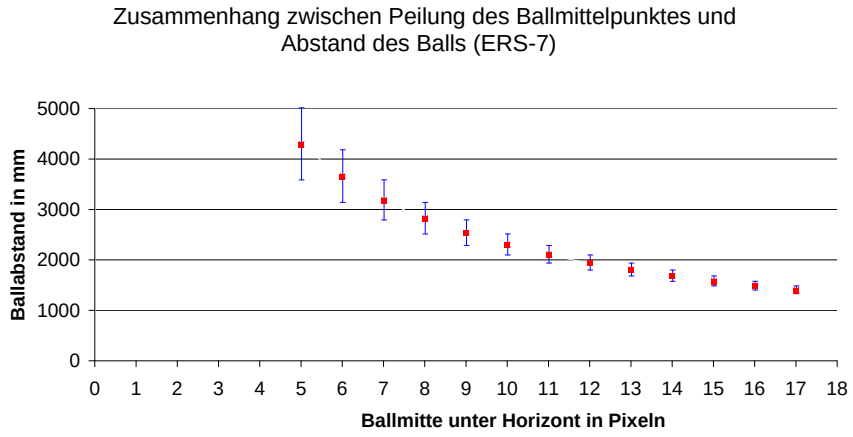


Abb. 4.3: *a) stellt den Zusammenhang zwischen Ausdehnung des Balls im Kamerabild in Pixeln und realer Ballentfernung dar*
b) grafische Darstellung des Messfehlers in Abhängigkeit von der Ballentfernung

4.2.3 Fehler bei der Ballwinkelmessung

Der Winkelfehler aus den reinen Bilddaten ist bei beiden Messverfahren maximal $2 * \sqrt{\frac{\alpha_{hor}}{2} + \frac{\alpha_{vert}}{2}}$, also $2 * \sqrt{\frac{0.29°}{2} + \frac{0.29°}{2}} = 0.41°$. Es kommt hierbei jedoch noch zu Fehlern in der Winkelbestimmung von bis zu 10° vertikal, verursacht durch die Laufbewegungen des Roboters. Im Zusammenhang mit der langen Auslesezeit der CCD-Kamera von 30 ms kommt es zudem zu geometrischen Verzerrungen, die wiederum in beiden Dimensionen bis zu 10°, also insgesamt fast 15° betragen können. Die Kameramatrix stellt für das gesamte Bild jedoch nur einmal Sensordaten zur Verfügung, die nur für den unteren Bildbereich gelten. Für eine exakte Zuordnung des Kopfwinkels zu erkannten Bildobjekten ist damit eine weitere Modellierung notwendig, die die aktuelle Kameramatrix an den aktuellen Bildabschnitt, in dem das erkannte Objekt gefunden wurde, anpasst.

a)



b)

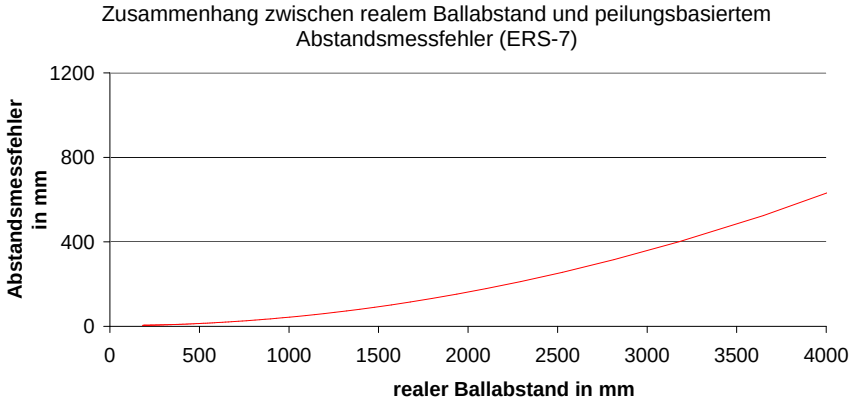


Abb. 4.4: . *a) stellt den Zusammenhang zwischen Abstand des Balls vom Horizont und Ballentfernung dar*
b) grafische Darstellung des Messfehlers in Abhängigkeit von der Ballentfernung

4.2.4 Vergleich der Berechnungsmethoden

In Abb. 4.5 wurden die Ballentfernungen mit Hilfe der Ballgröße einerseits und dem Peilungswinkel andererseits errechnet und gegenübergestellt. Erkennbar ist dabei, dass die Ballentfernungen, die basierend auf der ermittelten Ballgröße berechnet wurden, weitaus weniger schwanken als die Entfernungen, die mit Hilfe des Peilungswinkels festgestellt wurden. Dieses Ergebnis steht im Einklang mit der zu erwartenden Robustheit der größenbasierten Ballentfernungsmessung gegenüber Laufunruhen und der damit verbundenen Veränderung des künstlichen Horizontes des Roboters.

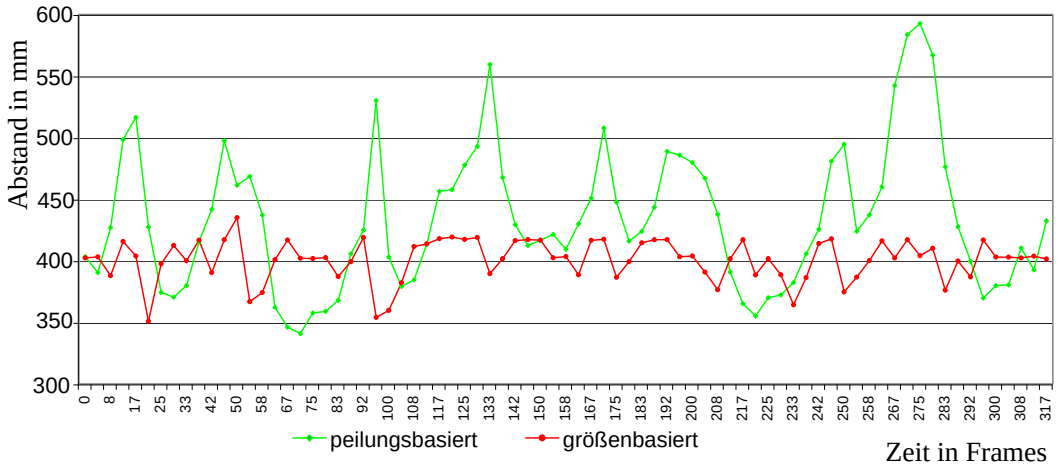


Abb. 4.5: *Vergleich von größen- mit peilungsbasierten Entfernungsmessverfahren, Roboter läuft auf der Stelle;*
rot: größenbasierte Ballentfernungsmessung, $\sigma = 16.84mm$*;*
grün: peilungswinkelbasierte Ballentfernungsmessung, $\sigma = 57.16mm$

Speziell beim ERS-7 Roboter führt die Tatsache, dass sich 30 statt wie bisher 25 Bilder innerhalb einer Sekunde auf 125 Frames[1] aufteilen, zu dem Problem, dass die verschiedenen Bilder nichtäquidistante Framenummerndifferenzen zu ihren Nachbarbildern besitzen. Im Detail sei die Framenummerndifferenz anfangs 5, dann haben die nächsten 5 Bilder zueinander einen Abstand von 4 Frames. Danach gibt es wieder eine größere Pause von 5 Frames, etc. pp. Tests haben jedoch gezeigt, dass die Bilder selbst zu äquidistanten Zeitabschnitten aufgenommen wurden. Die zugeordneten Framenummern geben den Aufnahmezeitpunkt daher nur ungenau wieder wie Abb. 4.6 zeigt. Insbesondere für die Ballgeschwindigkeitsberechnung kann dies zu Fehlern von bis zu 20% führen (Abb. 4.7).

[1] Als ein Frame wird ein Abarbeitungsschritt des Roboters, speziell des Motionprozesses, bezeichnet. Es gibt 125 Frames pro Sekunde, ein Frame dauert also 8 ms.

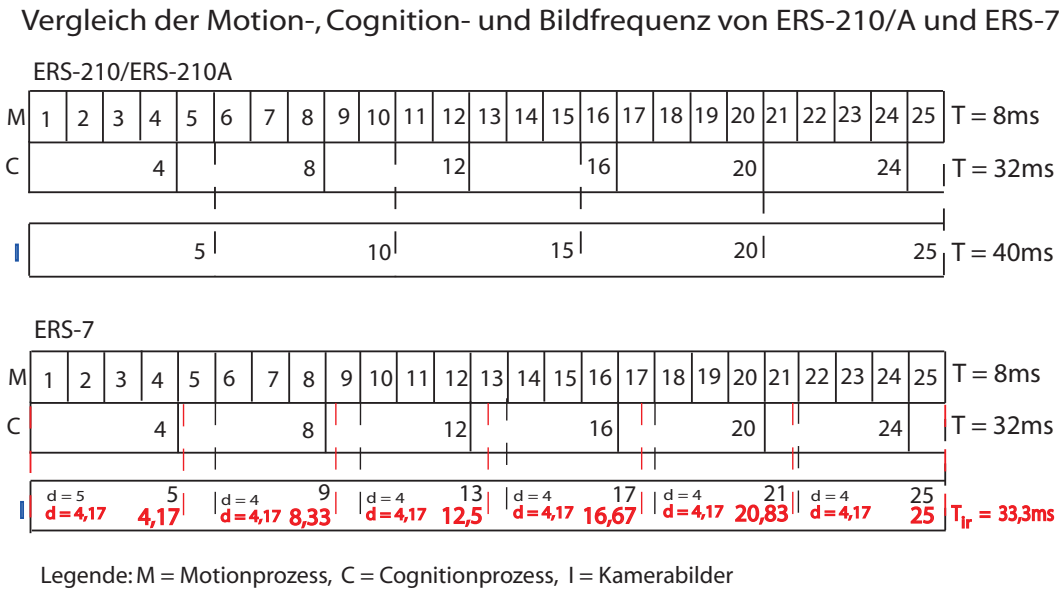


Abb. 4.6: *Vergleich der Bilderzuordnungen zwischen ERS-210/210A und ERS-7: Bei der 210er Reihe gibt es zu jedem Bild eine genaue Framenummer aus dem Motionprozess. Beim ERS-7 werden diese Nummern zugeordnet (schwarz), entsprechen aber nicht der exakten Bilderfassungszeit (rot). Die Bildabstände sind beim ERS-7 wie schon beim ERS-210 äquidistant, lassen sich aber nicht in ganzen Framezahlen ausdrücken, da 30 kein Teiler von 125 ist.*

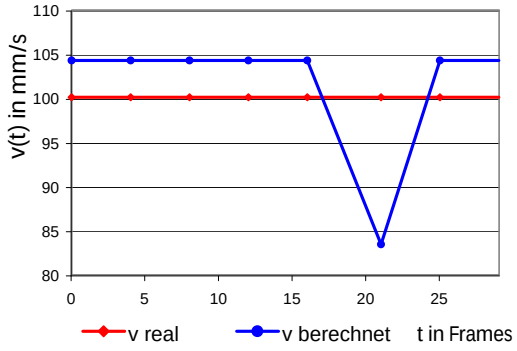


Abb. 4.7: *Geschwindigkeitsberechnungsfehler, die durch falsch zugeordnete Framenummern entstehen können. Ein Frame entspricht 1/125 Sekunde.*

4.2.5 Tabellarische Aufstellung der Fehlerarten bei der Ballerfassung

Die möglichen Fehlerquellen für die Ballperzeptberechnung werden in Tabelle 4.1 konzentriert dargestellt. Es ist festzustellen, dass der Gesamtfehler für die Bestimmung des Ortes bis zu 15° betragen kann. Diese Abweichung stellt m. E. ein Problem dar und wirkt sich negativ sowohl für die Bestimmung der Ballposition als auch für die Bestimmung der Ballgeschwindigkeit aus.

4.3 Kurze Beschreibung der Versuche

Als Referenzmodell für die Evaluation des Rao-Blackwellized Partikelfilters wird ein Kalmanfilter verwendet. Die Ergebnisse beider Verfahren werden mit den Rohdaten verglichen, um Stärken, bzw. Schwächen in einzelnen Bereichen der Modellierung besser analysieren zu können.

Folgende Versuche wurden durchgeführt:

1. Ein Roboter läuft auf der Stelle und misst die Varianz der Ballentfernung a) anhand der Rohdaten (Perzepte) und b) mit Hilfe eines Kalmanfilters (Abb. 4.8-1). Durch die Laufbewegungen werden die Ballperzepte verrauscht und entsprechen damit den Ballperzepten während eines Fußballspiels.

2. Ein Roboter läuft auf der Stelle und misst die Entfernung und Geschwindigkeit eines auf ihn zurollenden Balls a) anhand von Rohdaten, b) mit Hilfe eines Kalmanfilters bzw. RBPF (Abb. 4.8-2).

Ursache	Fehlerart	Quantifiz.	Behebung
Farbfehl-klassifizierung	Positions-fehler	von Güte der Farbtabelle abhängig	Bildverarbeitung, bessere Klassifi-zierungsalgorithmen
Geringe Kamera-auflösung	Positions-fehler	$\sigma_{horiz} \approx 0.3°$ $\sigma_{vert} \approx 0.25°$	Statistik über mehrere Randpixel
Bildverzerrung wegen langer Kameraauslese-zeiten und Lauf-bewegungen	Positions-fehler	$\sigma_{horiz} \approx 5°$ $\sigma_{vert} \approx 5°$	Bildentzerrung auf Odometriedatenbasis
Laufbewegungen verändern Roboteraus-richtung	Positions-fehler	$\sigma_{horiz} \approx 10°$ $\sigma_{horiz} \approx 10°$	Genaueres Roboter-modell, das die Roboterausrichtung exakter berechnet
Ungenaue Zuordnung von Framenum-mern zu Bildern	Geschwindig-keitsfehler	$\sigma_v \approx 8\%$	Verwendung von Bild-statt Framenummern

Tab. 4.1: *Die verschiedenen Fehlerarten bei der Ballperzeptberechnung.* $\sigma_{horiz}, \sigma_{vert}$ *stehen für horizontale und vertikale Winkelfehler bei der Perzeptberechnung.* σ_v *repräsentiert den Fehler bei der Geschwindigkeitsberechnung. Der Fehler, der durch Laufbewegungen entsteht, ist für die Ballperzeptberechnung am größten.*

Dieses Experiment dient dem Zweck abzuschätzen, wie gut und konstant der Kalmanfilter die Geschwindigkeit trotz verrauschter Rohdaten berechnen kann.

3. Ein stehender Roboter verfolgt einen Ball, der kurzzeitig durch ein Hindernis verdeckt wird (Abb. 4.8-3). Nur wenn es dem Roboter gelingt, die Geschwindigkeit des Balls vor der Verdeckung korrekt zu ermitteln, wird er in der Lage sein, den Ball bis zum Ende der Verdeckung zu verfolgen und wiederzufinden.

4. Ein stehender Roboter verfolgt einen Ball, der an einer Spielfeldbegrenzung abprallt (Abb. 4.8-4). Dieses Experiment soll zeigen, inwieweit der RBPF in der Praxis nichtlineare Bewegungen des Balls modellieren kann. Das Vergleichsmodell ist ein Kalmanfilter.

5. Ein stehender Roboter verfolgt einen Ball, der hinter einem Hindernis an der Spielfeldbegrenzung abprallt und auf derselben Seite hinter dem Hindernis hervorkommt (Abb. 4.8-5). Zu erwarten ist, dass ein gewöhnlicher Kalmanfilter den Ball in die „Bande propagiert“, nachdem er ihn hinter dem Hindernis verloren hat. Der RBPF sollte den Ball auf der richtigen Seite des Hindernisses erwarten.

6. Ein Ball wird von einem Roboter geschossen und soll danach direkt auf den Ball blicken (Abb. 4.8-6). Gemessen wird die Zeit, die der Roboter für das Wiederfinden des Balls nach dem Schuss benötigt.

7. Ein Ball wird hinter einem Hindernis abgelegt, sodass ihn der Roboter nicht sehen kann (Abb. 4.8-7). Durch Verwendung von Negativinformationen soll der Roboter den Ball möglichst schnell wiederfinden.

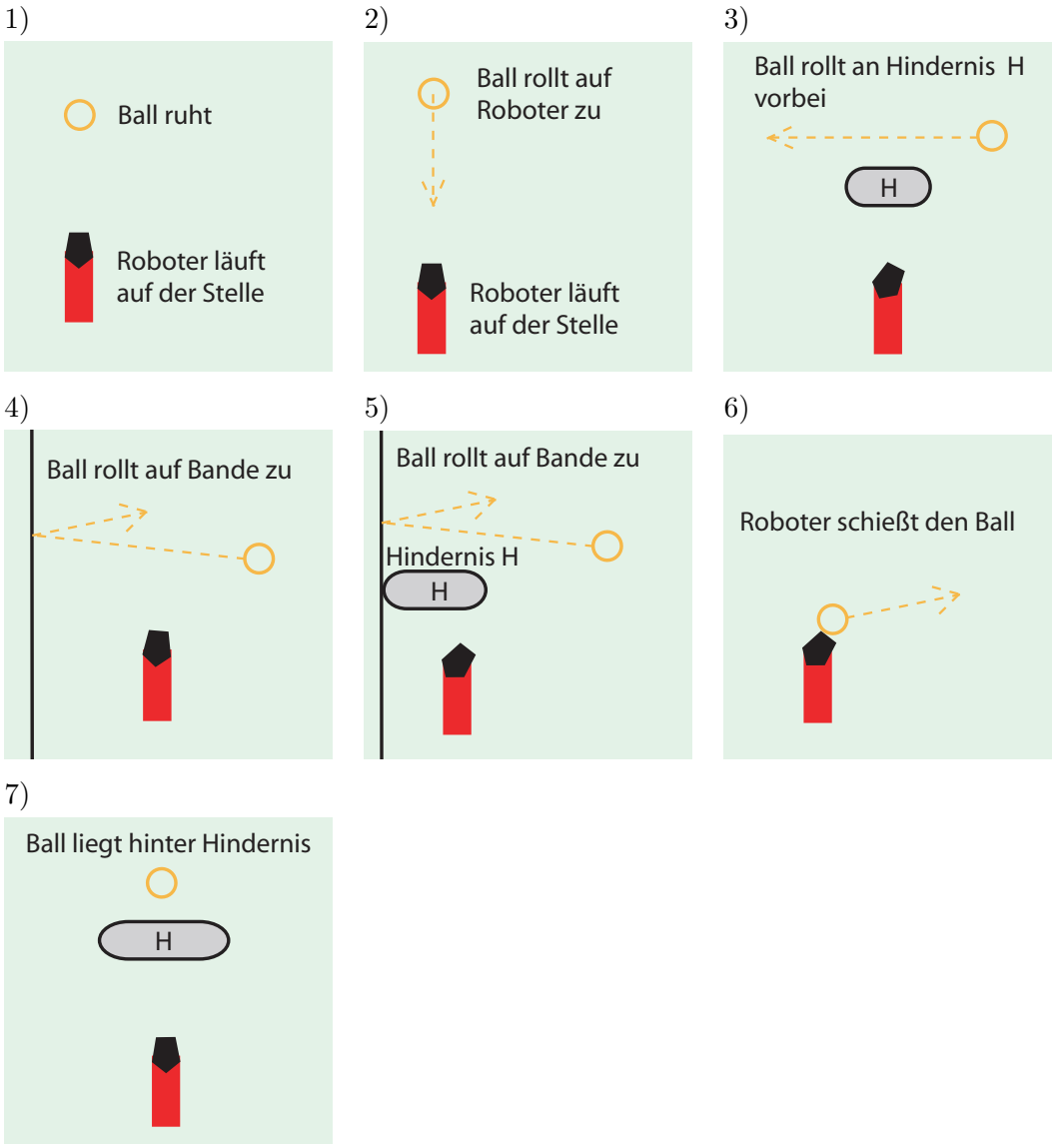


Abb. 4.8: Grafische Darstellung der 7 Experimente

4.4 Vergleich: Rohdaten vs. Kalmanfilterung

1. Versuch: Im ersten Versuch wurde die Varianz von ungefilterten Balldaten eines laufenden Roboters ermittelt. Die Entfernung zwischen Roboter und Ball blieb dabei zeitlich konstant. Insbesondere durch die Laufbewegungen ist die hohe Varianz von $\sigma^2 = 250mm^2$ für die gemessenen Ballentfernungen zu begründen (Abb. 4.9 a,b).

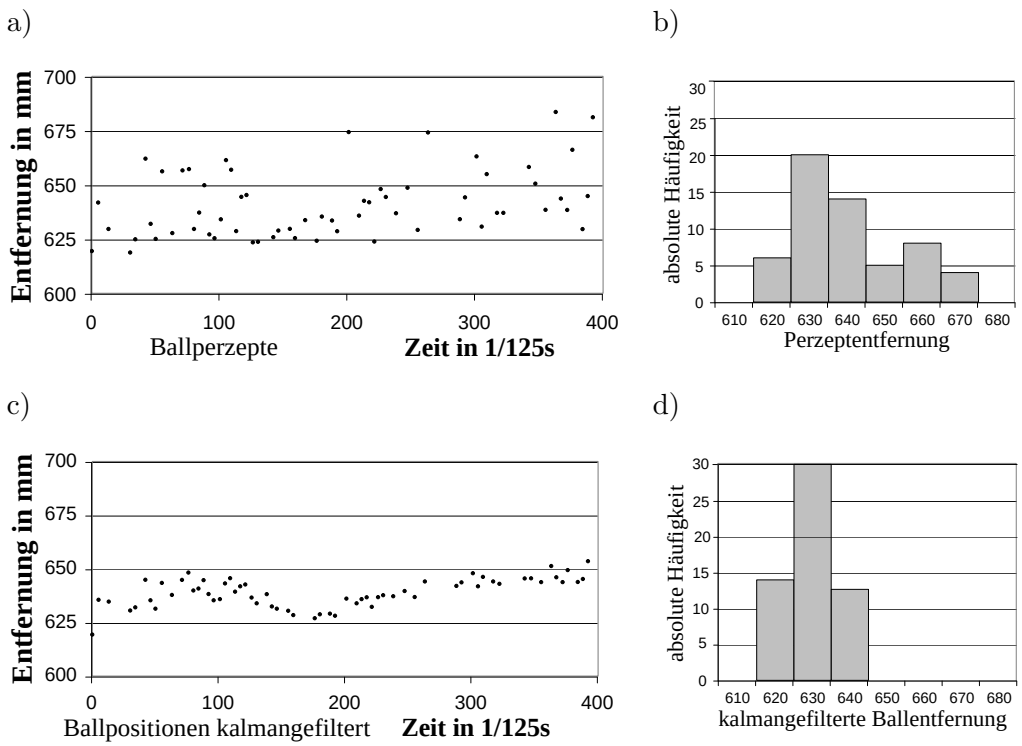


Abb. 4.9: *Exp. 1: a) Entfernungs-Zeit-Diagramm für Ballperzepte eines auf der Stelle laufenden Roboters*
b) Histogramm für die absoluten Häufigkeiten der gemessenen und gerundeten Ballentfernungen bei einer Varianz von $\sigma^2_{ungefiltert} = 250mm^2$
c) Entfernungs-Zeit-Diagramm für kalmangefilterte Ballpositionen
d) Histogramm der kalmangefilterten und gerundeten Ballentfernungen bei einer Varianz von $\sigma^2_{Kalman} = 48mm^2$

Für die Filterung der gemessenen Ballpositionen eignete sich der implementierte Kalmanfilter im Versuch sehr gut. Die Varianz der Ballposition betrug nach der Filterung $\sigma^2_{Kalman} = 48mm^2$, was etwa 20% des Ausgangswertes entspricht (Abb. 4.9 c,d).

2. Versuch: Im zweiten Versuch blickte ein laufender Roboter in Richtung eines auf ihn zurollenden Balls. Diesmal wurde zusätzlich zur Ballposition auch die Ballgeschwindigkeit berechnet. Der Versuch bestätigte die Vermutung, dass ungefilterte Balldaten für eine Geschwindigkeitsberechnung ungeeignet sind. Die Standardabweichung der Geschwindigkeit betrug in diesem Fall $\sigma_{v-ungefiltert} = 990mm/s$. Bei einer realen Ballgeschwindigkeit von etwa $350mm/s$ wurden Werte im Bereich von -2000 bis $2000mm/s$ ermittelt. Der Kalmanfilter bestätigte seine optimalen Filtereigenschaften und ermittelte die korrekte Ballgeschwindigkeit bei einer Standardabweichung von $\sigma_{v-kalmangefiltert} = 60mm/s$.

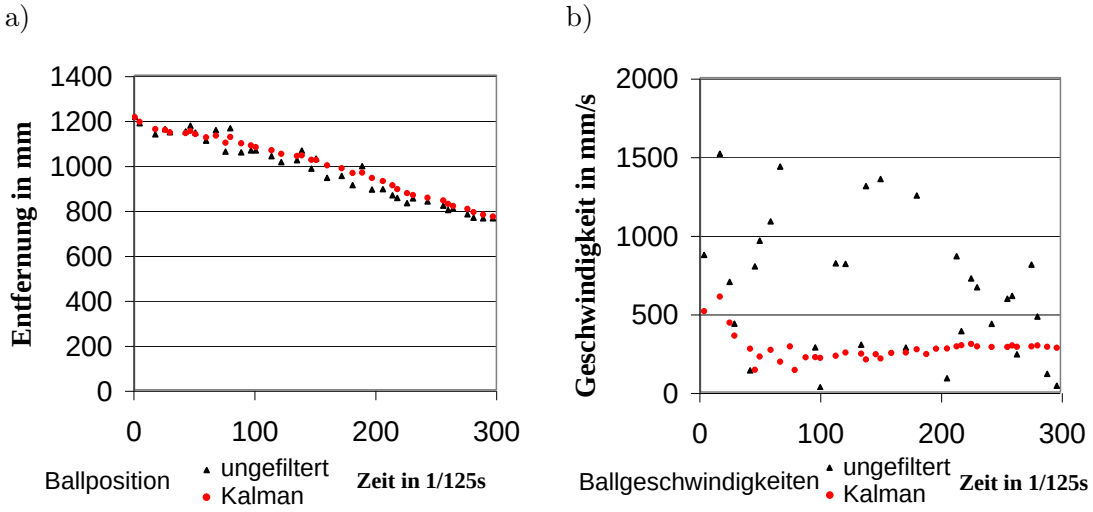


Abb. 4.10: *Exp. 2: a) Entfernungs-Zeit-Diagramm für ungefilterte und kalmangefilterte Ballpositionen eines auf den Roboter zurollenden Balls*
b) Geschwindigkeits-Zeit-Diagramm für ungefilterte und kalmangefilterte Ballgeschwindigkeiten desselben Experiments (bei ungefilterten Geschwindigkeiten wurden die Beträge dargestellt); $\sigma_{v-ungefiltert} = 990mm$; $\sigma_{v-kalmangefiltert} = 60mm$

3. Versuch: Im dritten Versuch wurden die Propagierungseigenschaften des Kalmanfilters unter der Bedingung, dass keine neuen Sensorinformationen zum Roboter gelangen, getestet. Der Roboter musste einen anfangs sichtbaren Ball hinter einem Hindernis verfolgen. Wichtig für einen erfolgreichen Versuch war dabei, dass der Roboter das verdeckte Objekt mit der exakten Geschwindigkeit verfolgt, um es auf der

anderen Seite des Hindernisses nicht zu spät oder zu früh zu erwarten und damit aus dem Blickfeld zu verlieren.

Es wurden zehn Versuche durchgeführt, wobei der Ball eine Geschwindigkeit von 800 mm/s besaß und vor der Verdeckung etwa eine Sekunde lang sichtbar war. Das Hindernis hatte eine Länge von etwa 40 cm. In acht von zehn Fällen gelang es dem Roboter, den Ball auf der anderen Seite des Hindernisses wiederzufinden (Abb. 4.11). Auf der Basis von ungefilterten Balldaten gelang es dem Roboter zweimal, den Ball hinter dem Hindernis zu orten. Das heißt, dass der Roboter mit einem Kalmanfilter die Ballgeschwindigkeit deutlich besser bestimmen kann als auf Basis von Rohdaten.

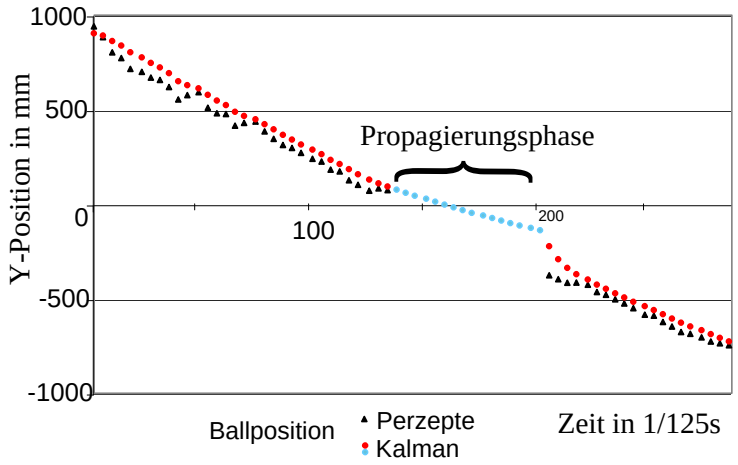


Abb. 4.11: *Exp. 3: Propagierung des Ballzustandes. Zu Beginn des Experiments erhält der Roboter Ballperzepte und berechnet die Geschwindigkeit. Bei fehlenden Sensordaten wird die neue Ballposition anhand der Geschwindigkeit und der alten Ballposition propagiert (blaue Punktereihe).*

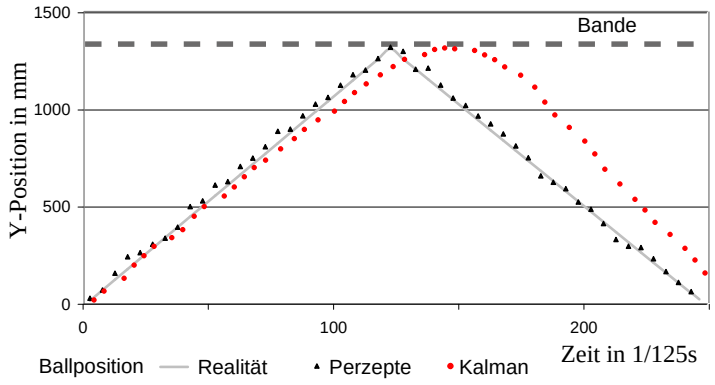
4.5 Vergleich: Kalmanfilter vs. RBPF

Zu den Vorzügen dieser Filter wurde an verschiedenen Stellen schon Wesentliches theoretisch gesagt. Nunmehr sollen anhand experimenteller Untersuchungen die Vor- und Nachteile beider Filter dargestellt werden.

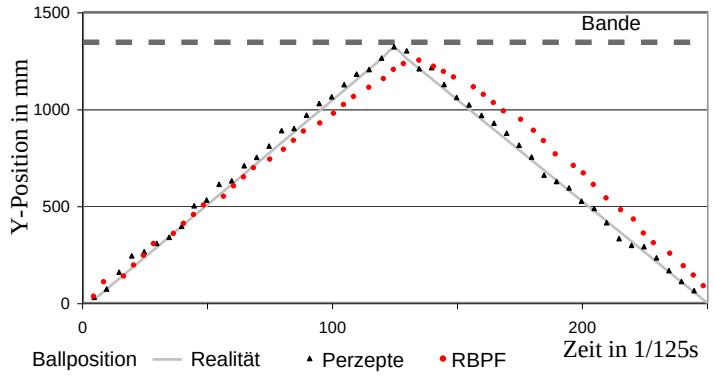
4. Versuch: Im vierten Experiment rollt ein Ball auf die Spielfeldbegrenzung zu und prallt ab. Der Versuch wurde sowohl mit einem Kalmanfilter als auch mit einem RBPF durchgeführt. Im Experiment benötigte der Kalmanfilter etwa zwei Sekunden, um die neue Geschwindigkeit und Ballposition zu modellieren. Ursache dafür ist das Prozessmodell des Kalmanfilters, welches an dem Ballzustand vor dem Abprall - auch bei widersprechenden Sensorinformationen - noch für einen gewissen Zeitraum festhält. Deshalb wurde der Ball nach der Kollision noch für etwa eine Sekunde weiter in Richtung Bande propagiert (Abb. 4.12 a).

Der Rao-Blackwellized Partikelfilter konnte trotz ungenauer Selbstlokalisierungsdaten $\sigma_{pos} = 100mm$ die Bandenkollision modellieren und sich auf die Richtungsänderung des Balls einstellen (Abb. 4.12 b).

a)



b)



c)

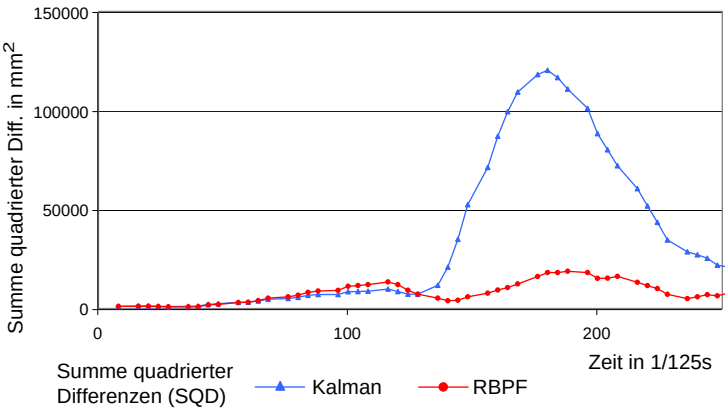


Abb. 4.12: *Exp. 4: Der Ball prallt von der Bande ab. Die Ballgeschwindigkeit ist konstant.*
a) Der Kalmanfilter kann auf die nichtlineare Bewegungsänderung des Balls erst nach etwa einer Sekunde reagieren.
b) Der Rao-Blackwellized Partikelfilter modelliert den Abprall des Balls von der Bande explizit, dementsprechend schnell aktualisiert er die neue Ballposition und -geschwindigkeit schon nach etwa 0.3 Sekunden.
c) Summe quadrierter Differenzen aus jeweils fünf Messwertepaaren; bei der Kalmanfilterung ergeben sich deutlich höhere Differenzen zwischen gefilterten und realen Positionen als beim RBPF.

5. Versuch: Der gleiche Versuch wird nun mit einem die Sicht versperrenden Hindernis an der Bande durchgeführt. Der Roboter kann die Kollision des Balls mit der Bande nicht direkt sehen. Nur wenn er die Kollision aufgrund seines Weltmodells vorausberechnen kann, wird er den Ball auf derselben Seite des Hindernisses erwarten, an der er ihn das letzte Mal gesehen hat.

Der Kalmanfilter propagierte den Ball hinter dem Hindernis mit der richtigen Geschwindigkeit. Anschließend wurde der Ball bis weit hinter die Bande modelliert (Abb. 4.13 a).
Der RBPF propagierte den Ball korrekt bis zur Bande. Danach sank die Geschwindigkeit stark ab und es kam zu kleinen Sprüngen in der Ballposition (Abb. 4.13 b). Nach kurzer Zeit konnte der Roboter den Ball wiederfinden.

Die Positionssprünge nach dem Abprall des Balls von der Bande sind dadurch zu erklären, dass die einzelnen Partikel bei der modellierten Kollision des Balls von der Bande in unterschiedliche Richtungen abprallten. Ohne Sensoraktualisierung waren keine Gewichtungen der Partikel möglich und die Partikelmenge konnte sich nicht an bestimmten Stellen konzentrieren. Die Partikelverteilung wurde deshalb breiter.

6. Versuch: Der Roboter schießt einen vorher gefangenen Ball und soll ihn möglichst schnell wiederfinden. Ein Roboter mit RBPF Ballmodellierung benötigte für das Wiederfinden des Balls etwa 0.7 Sekunden (Abb. 4.14). Mit einem gewöhnlichen Kalmanfilter dauerte dieser Vorgang bis zu 1.5 Sekunden, da der Roboter dafür seine gesamte Umgebung absuchen musste.

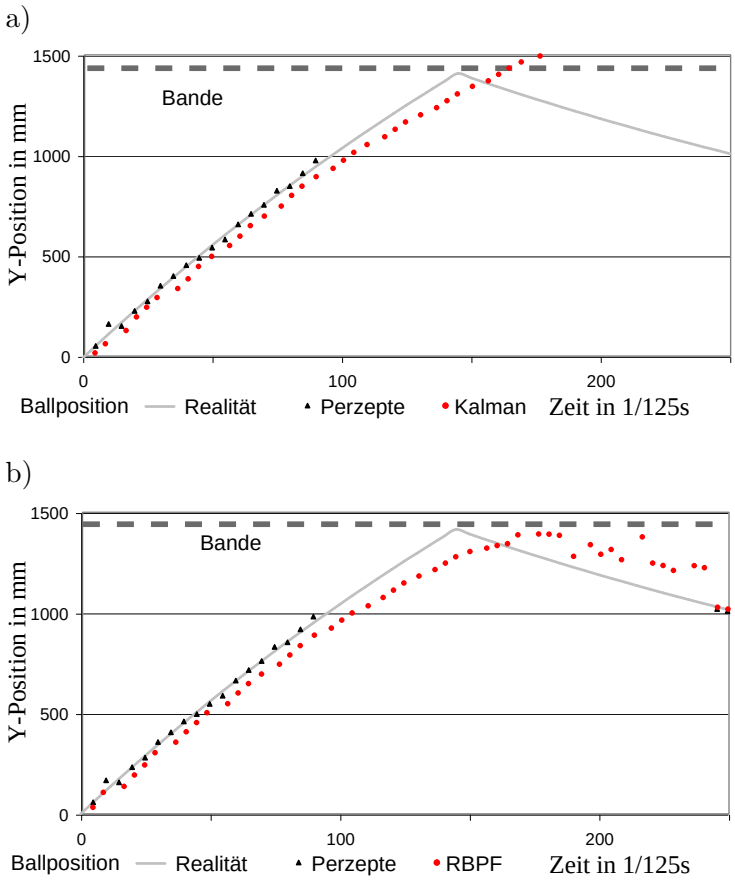


Abb. 4.13: *Exp. 5: Der Ball prallt hinter einem Hindernis und damit nicht sichtbar von der Bande ab und verliert an Geschwindigkeit.*
a) Der Kalmanfilter propagiert die Ballposition hinter die Bande und verliert dadurch den Ball.
b) Der RBPF modelliert die Kollision; aufgrund von fehlender Sensorinformation und Positionierungsunsicherheit weist die modellierte Ballposition kleinere Sprünge auf. Der Roboter findet jedoch den Ball wieder, weil er ihn in der Nähe der realen Position vermutet.

4.6 Zur Ballsuche mittels Negativinformationen

7. Versuch: In diesem Versuch wurde der Ball hinter einem Hindernis platziert, sodass er für den Roboter nicht sichtbar war. Der Roboter hatte nun die Aufgabe, den Ball durch Absuchen seiner Umgebung zu lokalisieren. Je nach Ausgangsposition und Lage benötigte er dafür zwischen 15 Sekunden und einer Minute. Die dabei entstehende Konvergenz der Hypothesen in einem kleinen Bereich wird in Abb. 4.15 gezeigt. Das Finden eines Balls, der sich hinter einem Hindernis befindet, wird in Abb. 4.16 dargestellt. Im Vergleich zu Ansätzen, bei denen sich der Roboter zufällig im Kreis dreht, geht der Roboter bei dieser Methode systematisch vor und unterscheidet bereits abgesuchte von noch nicht abgesuchten Bereichen. Ein Roboter, der sich im Kreis dreht, kann Objekte auf der anderen Spielfeldhälfte nur sehr klein oder gar nicht erkennen und wird einen weit entfernten Ball daher nicht wiederfinden. Das im Experiment getestete Suchverhalten wurde in Abb. 4.17 schematisch beschrieben. Zunächst erfolgte die Auswahl eines Bereichs, d.h. Clusters mit einer hohen Aufenthaltswahrscheinlichkeit des Balls (Abb. 4.17 a). Der Roboter steuerte diesen Bereich an und wich dabei den im Weg stehenden Hindernissen aus (Abb. 4.17 b). Nachdem der ausgewählte Bereich ergebnislos abgesucht wurde, wählte der Roboter sein nächstes Ziel aus.

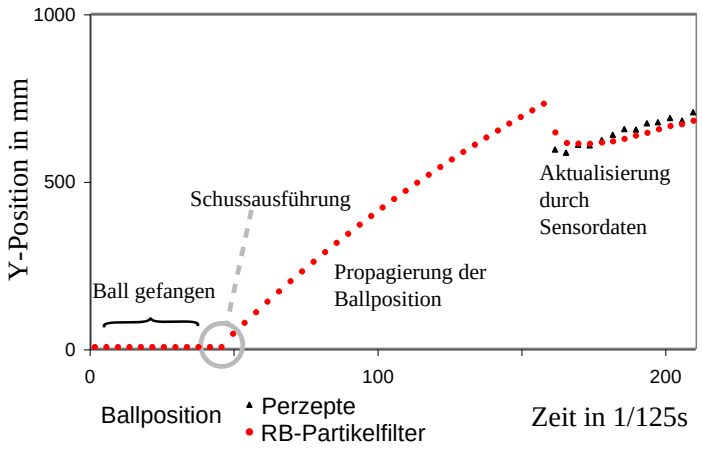


Abb. 4.14: *Exp. 6: Der Ballzustand wird nach der Schussausführung in die wahrscheinlichste Richtung propagiert. Nach Beendigung des Schusses blickt der Roboter auf die propagierte Ballposition und findet den Ball wieder.*

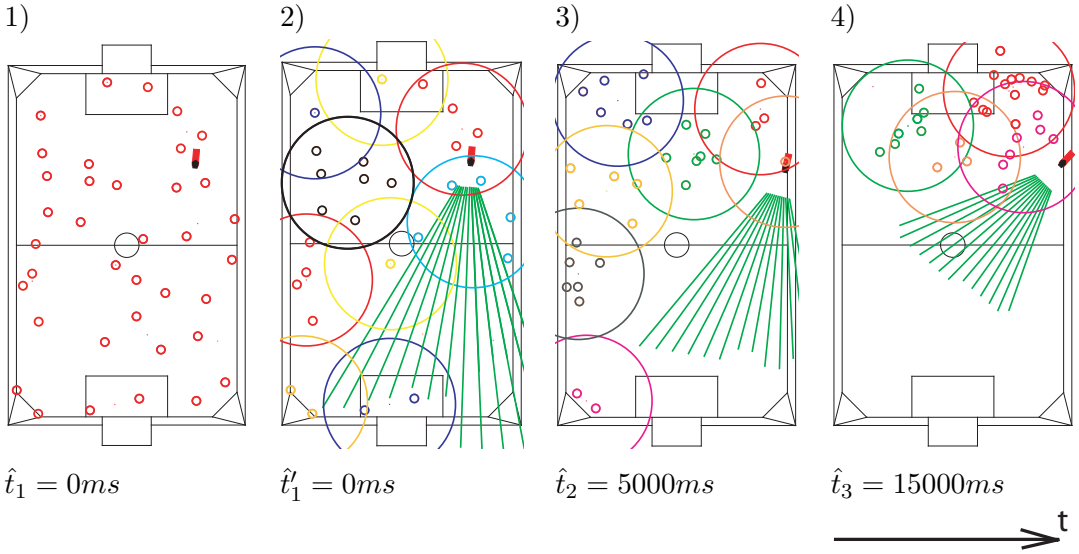


Abb. 4.15: *Eingrenzung des Hypothesenraums durch Negativinformationen. Für den Roboter sichtbare Bereiche sind durch die grünen Linien repräsentiert*
1) Gleichverteilung unmittelbar nach der Initialisierung
2) Clusterung der Ballhypothesen
3) Eingrenzung des möglichen Aufenthaltsbereichs
4) Verdichtung der Hypothesen in Bereichen, die noch nicht abgesucht wurden

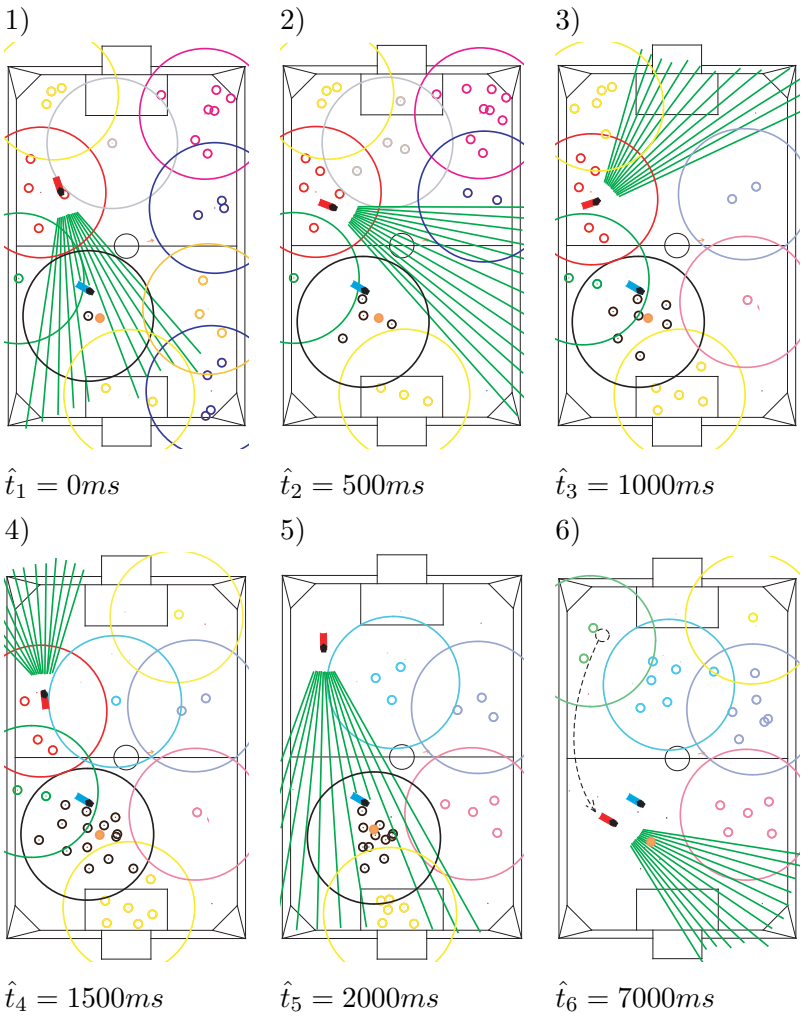


Abb. 4.16: *Ballsuche mithilfe von Negativinformationen. Der blaue Roboter verdeckt den Ball für den roten Roboter.*
1) - 4) Der rote Roboter sucht zunächst seine unmittelbare Umgebung nach dem Ball ab.
5) Es bilden sich Bereiche heraus, die nicht im Sichtbereich des Roboters liegen, insbesondere der Bereich hinter dem blauen Roboter.
6) Der rote Roboter bewegt sich nun in Richtung des vom blauen Roboter verdeckten Bereichs (gestrichelte Linie).

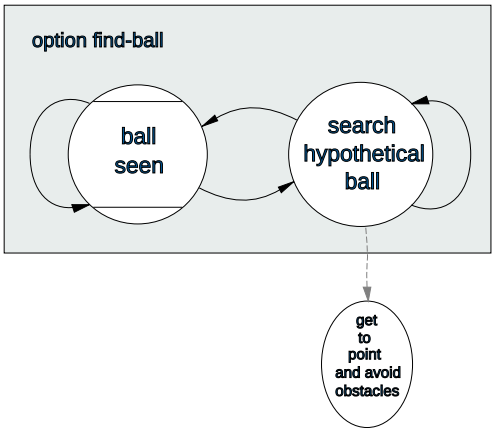


Abb. 4.17: *Beschreibung eines Suchverhaltens mit der Verhaltensbeschreibungssprache Xabsl [41]. Die Ballsuchoption „find-ball“ wählt einen abzusuchenden Bereich aus und steuert ihn mit der Unteroption „get-to-ball-and-avoid-obstacles an, bis der Ball gesehen wird.*

4.7 Zusammenfassung

Die Rohdaten der Ballposition sind für eine exakte Geschwindigkeitsbestimmung oder für die Propagierung der Ballposition nicht verwendbar. Ein Kalmanfilter erfüllt diese Aufgaben sehr gut solange der Ball nicht in Interaktion mit anderen Objekten wie der Bande oder Robotern tritt. Die Verwendung von Rao-Blackwellized Partikeln zeigte im Test hinsichtlich der nichtlinearen Bewegungsänderungen des Balls vielversprechende Resultate. Insbesondere für die Modellierung von Bandenkollisionen, ausgeführten Schüssen sowie für die Verwendung von Negativinformationen ist der Rao-Blackwellized Partikelfilter einem Kalmanfilter deutlich überlegen.

Kapitel 5

Auswertung und Ausblick

5.1 Auswertung

Die Objektmodellierung unterteilt sich grundsätzlich in zwei Hauptaufgabenbereiche: Die Zustandsabschätzung und die aktive Wahrnehmung. Ein wesentliches Problem für eine akkurate Zustandsabschätzung sind ungenaue und unvollständige Sensordaten. Deshalb ist die Schaffung fehlerrobuster Algorithmen eine sehr bedeutsame Aufgabe.
Die im Rahmen dieser Arbeit vorgestellte Verwendung von Kalmanfiltern zeigte in den meisten Situationen sehr gute Ergebnisse bezüglich der Ballmodellierung. Das wurde auch in besonderer Weise bei der RoboCup WM 2004 in Form eines äußerst reaktionsschnellen Roboter-Torwarts sichtbar, dessen Ballmodellierung auf der Grundlage eines Kalmanfilters funktionierte.
Eindeutig belegt wurde in dieser Arbeit, dass die Verwendung eines Rao-Blackwellized Partikelfilters im Allgemeinen zu Verbesserungen führte. In besonderer Weise werden diese Qualitätsteigerungen bei Interaktionen des Balls mit der Spielfeldbegrenzung sowie beim Wiederfinden des Balls nach ausgeführten Schüssen sichtbar. Eine noch zu lösende Aufgabe stellt die Modellierung von Interaktionen des Balls mit anderen Robotern dar, da die Mitspielermodellierung auf dem Aibo-Roboter noch keine befriedigenden Resultate liefert.
Hervorhebenswert ist auch die Tatsache, dass durch die Verwendung von Partikelfiltern der Einsatz von Negativinformationen in Kombination mit Hindernisdaten ermöglicht wurde und in diesem Zusammenhang auch gute Resultate zeigte, wenn es darum ging, den Ball wiederzufinden. Im Gegensatz zu bisherigen statischen Im-

plementierungen, bei denen der Roboter den Ball auf festgelegten, oft intuitiv bestimmten Routen sucht, konnte der Roboter mittels des hier vorgestellten Verfahrens seinen Suchweg selbst festlegen und schnell auf Situationsänderungen reagieren. Hindernisinformationen waren also sehr wirksam, freie Bereiche zu erkennen und über längere Zeit verdeckte Bereiche direkt anzusteuern.
In der Arbeit durchgeführte Untersuchungen über Kombinationsvarianten von verschiedenen Sensordaten weisen auf neue Möglichkeiten bzw. Anwendungen für Kalmanfilter und Rao-Blackwellized Partikelfilter hin. Dabei ist zu beachten, dass eine genaue Messfehleranalyse die Grundlage für eine optimale Anpassung der Filterparameter an die Umgebung, in der die verschiedenen Filter operieren sollten, darstellt.
Abschließend sei bemerkt: Obwohl die hier vorgestellten Algorithmen in einer speziellen Domäne angewandt wurden, stellen sie keine besonderen Bedingungen an die zu verwendenden Sensoren. Daher sind sie als Lösungsmöglichkeiten für eine Vielzahl von Applikationen zu betrachten, in denen es vorrangig darum geht, Informationen in dynamischen Umgebungen schnell und effizient zu verarbeiten.

5.2 Ausblick

In der Forschungsarbeit im Rahmen der KI, insbesondere der Robotik, wurden in den letzten Jahren interessante Arbeiten zu Themen wie Bildverarbeitung, Verhaltensmodellierung und Bewegungssteuerung angefertigt und z.T. erfolgreich in die Praxis umgesetzt. Die vorliegende Arbeit zur Umweltmodellierung mit ihren vielen Nebenaspekten stellt eine Ergänzung zu diesen Forschungsthemen dar. Davon ausgehend sind weitere Forschungsarbeiten auf dem Gebiet der Multiagentenmodellierung und den damit verbundenen Kooperationsstrategien zu leisten. Dabei geht es m. E. um Fragestellungen, inwieweit die verschiedenen Roboter ihre Weltbilder kombinieren und diese kombinierten Informationen zur gemeinsamen Handlungsplanung, wie z.B. einer verteilten Aufmerksamkeitssteuerung für die Modellierung verwenden können. Es sollten auch Fragen zur gemeinsamen Modellierung von Negativinformationen im Mittelpunkt der Betrachtung stehen.
Andere Fragestellungen gibt es im Bereich der Sensorik, insbesondere der Bildverarbeitung. Ziel ist es dabei, verschiedene erkannte Objekte schon in der Bildverarbeitung in Beziehungen zu setzen; zum einen um Fehler zu erkennen und zum anderen für eine genauere Modellierung verschiedener Objekte zueinander.
Die Kombination verschiedener Sensordaten, welche in dieser Arbeit anhand eines

Beispiels (eines Balllokators) vorgestellt wurde, könnte auch in Anlehnung an die menschliche Wahrnehmung, die insbesondere durch Kombination verschiedenster Sensoren funktioniert, neue Erkenntnisse für die Robotik bringen.
Lernstrategien können dem Agenten helfen, sich selbstständig an verschiedene Umgebungen anzupassen und ihre Aktionsplanung zu optimieren.
Für die weitere Forschung im RoboCup stellt sich die Frage, inwieweit Verfahren zur Ballsuche auf Basis von Hindernisinformationen, z.B. mit Hilfe einer Hindernismodellierung vervollkommnet werden können (Abb. 5.1). Verfahren wie Potenzialfelder [44] können in diesem Zusammenhang die Entwicklung von effizienten Wahrnehmungsstrategien unterstützen, ebenso wie die Erstellung von Heuristiken, die die Kosten von Wahrnehmungshandlungen determinieren.
Speziell für die Aibo-Liga würde eine Feldspielererkennung und -modellierung neue Forschungsgebiete eröffnen, wenn es darum geht, Handlungsstrategien des gegnerischen Teams zu durchschauen bzw. einzelne Handlungen vorherzusagen, wie es bereits in der Simulationsliga des RoboCup möglich ist.
Allein aus den hier vorgeschlagenen Forschungsthemen wird deutlich, dass es auf dem Gebiet der Objektmodellierung noch eine Vielzahl wichtiger, interessanter sowie notwendiger Aufgaben zu lösen gibt.

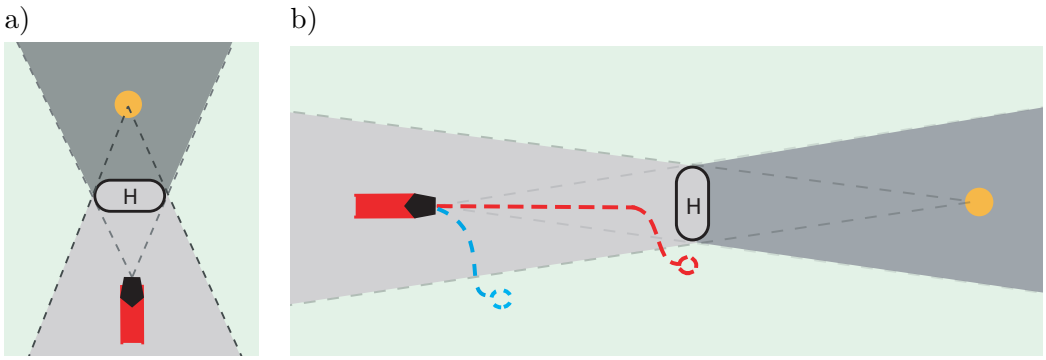


Abb. 5.1: *Vorschlag für ein effizienteres Suchverhalten durch eine Hindernis- und Sichtbarkeitsmodellierung. a) Aufteilung des Spielfelds in Sichtbarkeitsbereiche,*
dunkelgrau: vom Roboter nicht sichtbarer Bereich; hellgrau: Schattenbereich, innerhalb dessen die Position der Ballhypothese (orange) nicht sichtbar ist;
b) mögliche Suchverhalten: rot versucht direkt zur Hypothese zu gelangen (bisheriges Verhalten), blau verlässt direkt den Schattenbereich und kann damit eher die Hypothese auf Vorhandensein eines Balls testen

5.3 Danksagung

Mein Dank gilt in erster Linie Prof. Burkhard, der den RoboCup an der Humboldt-Universität ins Leben gerufen, bzw. zu bemerkenswerten Ergebnissen geführt und auch diese Arbeit betreut hat. Des Weiteren möchte ich mich bei Jan Hoffmann für die konstruktive Betreuung dieser Arbeit, bei Uwe Düffert, der mir stets mit Rat und Tat zur Seite stand sowie bei Matthias Jüngel, Martin Lötzsch, Thomas Röfer, Max Risler und dem gesamten German Team bedanken, deren kreative und produktive Arbeit im Rahmen des RoboCup eine wichtige Grundlage für den Erfolg des Projekts darstellt.
Danken möchte ich auch meiner Familie für ihre moralische Unterstützung.

Anhang A

Objektmodellierung im Rahmen der Sony Liga

Dieses Kapitel hat die Beschreibung von Objektmodellierungsparadigmen zum Inhalt, die speziell bei der Referenzarchitektur Aibo ERS-210/210A sowie deren Weiterentwicklung Aibo ERS-7 auftreten. Die Sensordatenerfassung und -verarbeitung sowie die damit verbundenen Schwierigkeiten sollen dabei im Mittelpunkt der Betrachtung stehen. Darüber hinaus werden einige weitere speziell für diesen Roboter entwickelte Modellierungsalgorithmen vorgestellt. Zunächst erfolgt die Beschreibung der für die Ballmodellierung relevanten Sensordaten des Aibo-Roboters.

A.1 Sensordaten

Die beiden für die Ballerkennung relevanten Sensoren sind der PSD-Sensor und die CCD-Kamera des Roboters. Obwohl die CCD-Kamera bedeutend wichtiger für die Ballerkennung ist, gibt es Situationen, in denen die Kamera den Ball nicht erfassen kann und der PSD-Sensor hilfreiche Daten über die Ballposition liefert.

A.1.1 CCD-Kamera

Der wichtigste Sensor der Aibo-Roboter für die Ballerkennung ist die CCD-Kamera. Tabelle A.1 zeigt die wesentlichen Kameradaten des ERS-7. Daraus geht hervor, dass der Roboter durch den geringen Kameraöffnungswinkel von etwa 55° nur einen begrenzten Teil seiner Umgebung wahrnehmen kann.

Kategorie	ERS-7
Kameraauflösung in Pixeln	208*160
Bildwiederholfrequenz (fps)	30
Kameraöffnung vertikal (Grad)	44
Kameraöffnung horizontal (Grad)	55
Farbraum	YUV

Tab. A.1: *Kameraparameter beim ERS-7 Roboter*

A.1.2 Der PSD-Sensor

Der PSD-Sensor[1] ermöglicht es dem Roboter, Objekte zu erkennen, die sich vor ihm befinden. Der Sensor arbeitet mit Infrarotfrequenzen. Da es sich um einen Punktsensor handelt, kann immer nur die Entfernung zu einem Punkt gemessen werden. Für die Erkennung von Hindernissen wurde der PSD-Sensor jedoch schon erfolgreich eingesetzt. Der in der Sony-Liga benutzte Ball bewirkt, dass die Infrarotstrahlen absorbiert werden. Im Ergebnis misst der Sensor bei vorhandenem Ball eine größere Entfernung zum Boden als ohne Ball (Abb. A.1). Dadurch ist erkennbar, ob sich der Ball oder ein anderer Roboter in unmittelbarer Nähe vor dem Roboter befinden. Dies ist insbesondere deshalb von Vorteil, weil der Roboter beim Greifen des Balls mit der Kamera über den Ball schaut und damit nicht visuell feststellen kann, ob er den Ball nach etwaigen Drehungen verloren hat oder ob der Greif-Versuch erfolgreich war.

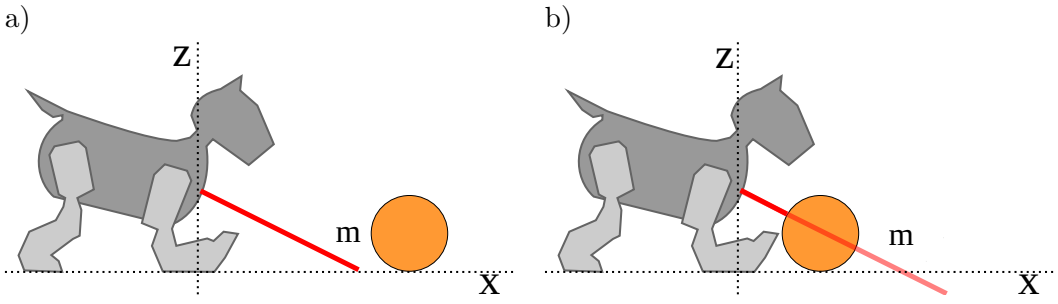


Abb. A.1: *a) der PSD-Sensor des Roboters ohne Gegenstand in der Nähe misst den Abstand m zum Boden (ERS-7),*
b) ein Ball vor dem Roboter bewirkt die Messung eines erhöhten Abstandes m, weil die Infrarotstrahlen vom Ball absorbiert werden

[1] PSD steht für Position Sensing Detector

A.2 Perzeptberechnung aus Bild- und Gelenkdaten

Dieser Abschnitt gibt einen Überblick über die Schritte, die zur Extraktion der Balldaten aus den Bildinformationen erforderlich sind. Im Weiteren wird beschrieben, wie der Roboter zum einen seine Ausrichtung sowie zum anderen die der Kamera relativ zum Feld bestimmen kann, um aus den Bilddaten die Entfernung zu erkannten Objekten berechnen zu können. Darauf aufbauend wird in einem letzten Abschnitt die Bildverarbeitung erläutert, die dazu dient, aus den Bilddaten Objektinformationen zu extrahieren.

A.2.1 Stellung der Kopfgelenke und Beine

A.2.1.1 Daten der Kopfgelenke

Sobald der Roboter ein Bild verarbeitet und daraus Datenstrukturen über erkannte Objekte generieren soll, muss er berücksichtigen, aus welcher Blickrichtung die Bilddaten empfangen wurden. Nur so ist es im späteren Verlauf möglich, die aus den Bildinformationen gewonnenen Daten angemessen ins Weltmodell einfließen zu lassen. Um aus der Position des Balls bzw. der Landmarken im Bild auf die reale Position relativ zum Roboter schließen zu können, werden u.a. die Gelenkwinkeldaten der drei Kopf- bzw. Halsgelenke benötigt. Unter Nutzung einer Kameramatrix, die die geometrischen Eigenschaften der einzelnen Kopfgelenke speichert, kann aus den aktuellen Gelenkwinkeln des Kopfes direkt auf die momentane Blickrichtung des Roboters geschlossen werden. Daten aus dem zweidimensionalen Bild können zusammen mit dem Wissen über die Größe der einzelnen Objekte in dreidimensionale Koordinaten relativ zum Roboter zurückgerechnet werden. Da der Auslesevorgang des Bildes etwa 30 ms benötigt und die Sensordaten zur Erstellung der Kameramatrix erst am Ende des Auslesens ermittelt werden, ist die Kameramatrix nur für den unteren Teil des Bildes näherungsweise zeitlich präzise.

A.2.1.2 Neigung des Roboters

Bestimmte Verfahren zur Abstandsmessung setzen Informationen über die Körperneigung des Roboters voraus. Die Berechnung der Körperneigung kann auf zwei verschiede Arten erfolgen.

Die erste Möglichkeit stellt das Abfragen der Neigungssensoren des Aibo dar. Diese Sensoren messen die verschiedenen Beschleunigungen des Roboters für alle

drei Raumdimensionen. Ein stehender Roboter wird in der Regel nur die Erdbeschleunigung messen, woraus die Körperneigung zurückgerechnet werden kann. Die Roboterhöhe ist mit diesem Verfahren nicht verifizierbar. Sie muss daher manuell gemessen und dem Roboter mitgeteilt werden. Durch ruckartige Laufbewegungen können die Beschleunigungsdaten zudem verrauscht sein.

Eine zweite Möglichkeit für die Berechnung der Körperneigung stellt die Vorwärtskinematik dar. Aus der Abfrage der Beingelenkwinkel können die Körperneigung sowie die Körperhöhe berechnet werden. Der Nachteil bei der Neigungsbestimmung über die Beingelenke besteht darin, dass der Roboter nicht feststellen kann, ob er auf einer schiefen Ebene steht. Dieser Fall kommt während eines Spiels allerdings sehr selten vor. Der Beschleunigungssensor kann solche Situationen wiederum erkennen. Die Vorteile der Neigungsbestimmung aus den Beingelenken, insbesondere die höhere Robustheit gegen Laufunruhen, überwiegen jedoch die Nachteile, weshalb dieses Verfahren im aktuellen Code des German Teams [3] Anwendung findet.

Wurden alle oben beschriebenen Daten ausgewertet, können nicht nur die aus dem Bild extrahierten Daten räumlich zugeordnet werden. Auch die Berechnung eines künstlichen Horizontes wird damit möglich. Das bedeutet für den Roboter eine zusätzliche Hilfe, wenn es darum geht, nach bestimmten Objekten im Bild wie dem Ball oder nach Landmarken zu suchen. Mit anderen Worten, wenn bestimmte Annahmen getroffen werden, wie z.B., dass sich ein Ball immer unterhalb oder eine Flagge immer oberhalb des Horizontes befinden, können Suchverfahren an die Umgebung angepasst werden und damit effizienter ablaufen. Zu berücksichtigen ist jedoch, dass der berechnete künstliche Horizont nur bedingt zuverlässig ist und je nach Implementation mehr oder weniger Abweichungen vom realen Wert aufweist (Abb. A.2).

A.2.2 Ballerkennung durch die Bildverarbeitung

An dieser Stelle sollen die Teile der Bildverarbeitung auszugsweise vorgestellt werden, die zur Extraktion von Informationen über den Ball aus Bilddaten erforderlich sind.

Um den Ball bzw. andere Objekte wie Flaggen, Tore oder Feldlinien im Bild erkennen zu können, wird das Bild mittels eines Linienrasters, den sog. „Scanlines“ parallel (Abb. 4.1 b) abgetastet. Das im German Team verwendete Verfahren mit Scan-Linien hat den Vorteil, dass nur ein Teil aller Bildpunkte verarbeitet werden muss und dadurch Rechenzeit gespart wird. Wie oben erwähnt, ist der künstliche

a) b)

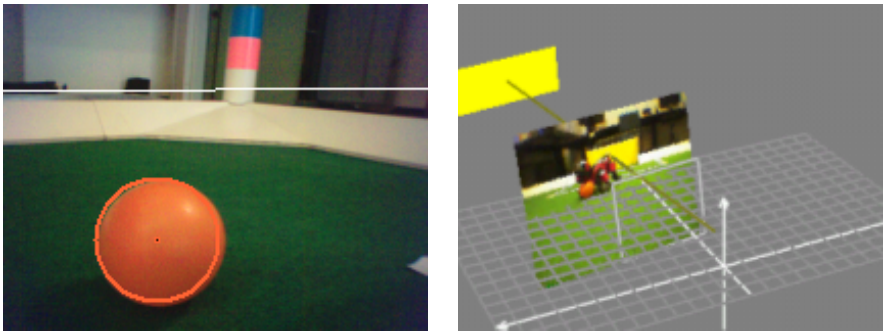


Abb. A.2: *a) der ins Kamerabild eingeblendete künstliche Horizont (ERS-7), b) anschauliche Projektion des Kamerabildes in das Blickrichtungskoordinatensystem des Roboters [4]*

Horizont im Bild durch die Daten der Robotergelenke sowie durch den Neigungssensor bestimmbar. Dadurch kann das Bild senkrecht zum Horizont abgesucht werden. Ein paralleles Abtasten des Bildes mit Scanlines hat ein radiales Abtasten des Feldes zur Folge (Abb. A.3). Weit entfernte Objekte werden ggf. nicht mehr erkannt, wenn sie zwischen zwei Scanlines liegen. Zu viele Scanlines beeinflussen jedoch den Berechnungsaufwand negativ. Eine Lösung stellt die Erhöhung der Scanlines in der Nähe des Horizonts dar, um auch weit entfernte Objekte zu erfassen.

a) b)

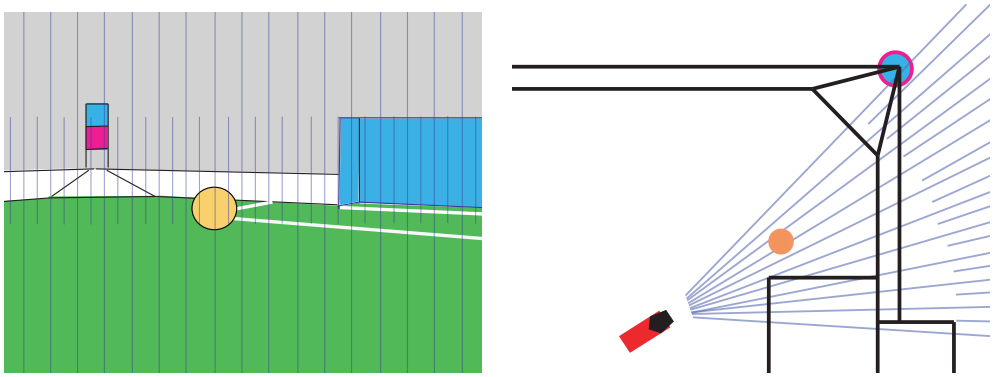


Abb. A.3: *a) Parallele Scanlines im Bild, b) dieselben Scanlines verlaufen radial auf dem Spielfeld.*

Die Bildpunkte der Scanlinien werden farbklassifiziert [4, 28, 23]. Das geschieht durch eine manuell oder automatisch [4] erstellte Farbtabelle. Für die Ballerkennung

sind orange klassifizierte Punkte von Interesse. Auch Flaggen, Tore oder Feldlinien werden anhand von farbklassifizierten Bildpunkten erkannt. Werden ein oder mehrere Bildpunkte der Farbklasse „Orange“ gefunden, ist es die Aufgabe eines eigens dafür entwickelten „Ballspezialisten“ zu entscheiden, ob es sich bei dem Punkt um einen Teil des Balls handelt oder nicht. Dazu wird die unmittelbare Umgebung des gefundenen Punktes nach weiteren Merkmalen abgesucht, die auf einen Ball schließen lassen. Im positiven Fall liefert die Bildverarbeitung Informationen über Größe und Mittelpunkt des Balls im Bild zurück. Zusammen mit den Daten über die Roboter- und Kameraausrichtung ist es möglich, die reale Position des im Bild erkannten Objektes zu berechnen. Der dabei entstehende Datentyp, der die Entfernung und den Winkel des erkannten Objektes zum Roboter enthält, wird als Perzept bezeichnet. Informationen über einen erkannten Ball werden durch ein Ballperzept gespeichert.

Um Fehler bei der Perzeptberechnung auszuschließen, wird zusätzlich zur reinen Bildverarbeitung auch Hintergrundwissen benutzt, z.B. dass ein Ball nur unterhalb des Horizontes und über grünem Rasen gesehen werden kann. Außerdem ist die Spielfeldgröße begrenzt, der Ball kann somit nicht beliebig weit vom Roboter entfernt sein und muss eine Mindestgröße besitzen.

A.3 Ballperzepte

Um ein Fehlermodell für die Messfehler der Ballperzepte erstellen zu können, ist eine Betrachtung der Kameraauflösung sowie des Öffnungswinkels nötig. Wie Abb. A.4 zeigt, spannt jeder Pixel jeweils einen horizontalen und vertikalen Raumwinkel auf. Je kleiner dieser Winkel ist, desto genauer ist die Auflösung innerhalb der Ebene auf der er liegt. Eine hohe horizontale Auflösung bedeutet daher eine hohe Messgenauigkeit für die horizontale Ausdehnung von Objekten. Es ist in Abb. A.4 zu erkennen, dass die Pixelwinkel am Bildrand kleiner werden. Der Ball nimmt am Bildrand mehr Pixel ein, als in der Bildmitte, was zu Verzerrungen am Bildrand führt. Es ist deshalb nicht ausreichend, die reine Ausdehnung eines zu messenden Objektes in Pixeln zu betrachten, sondern vielmehr den Winkel, den es aufspannt. Abb. A.4 soll das verdeutlichen.

Grundlage der Ballmodellierung stellen die aus den Sensordaten gewonnenen Ballperzepte dar. Die Abb. A.5 zeigt visualisierte Ballperzepte, die aus einem am Roboter vorbeirollenden Ball entstanden sind. Der Abstand des Balls ist dabei auf verschiedene Arten ermittelbar. Die zwei im German Team gebräuchlichen Verfahren

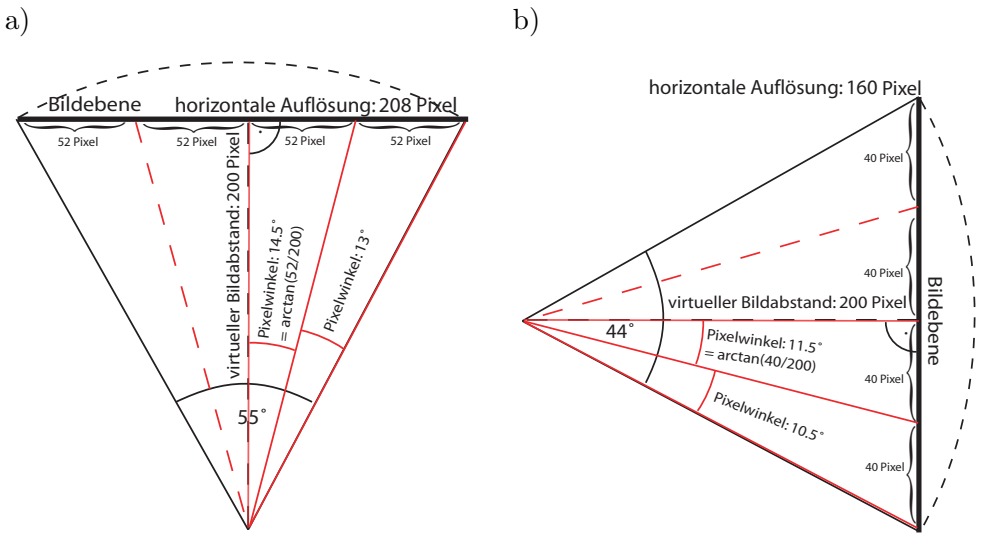


Abb. A.4: *a) horizontale Pixelwinkel für die mittlere Pixelzeile, b) vertikale Pixelwinkel für die mittlere Pixelspalte;*
Die Bildzeile bzw. -spalte wurde zur Veranschaulichung in vier gleich lange Teile zerlegt und im Anschluss daran die dazugehörigen Pixelwinkel berechnet. Es wird deutlich, dass die Pixelwinkel zur Bildmitte größer und zum Rand hin kleiner werden. Pixelanzahl und Winkel stehen damit in keinem festen Verhältnis

sind die Bestimmung der Ballentfernung sowohl über die Bildgröße als auch über den Peilungswinkel. Diese beiden Verfahren werden in den nächsten Abschnitten behandelt.

A.3.1 Die größenbasierte Ballentfernungsbestimmung

Eine erste Möglichkeit der Ballentfernungsbestimmung ergibt sich aus der Bestimmung der Größe sowie der Lage des Balls im Kamerabild. Ein Vorteil dieses Verfahrens ist die Robustheit gegenüber Laufbewegungen und veränderlicher Körperneigung[2]. Die Entfernungsbestimmung bedarf hierbei nahezu keinerlei weiterer Informationen über die Stellung der Beine oder die Ausrichtung des Kopfes, welches durch die Tatsachen begünstigt wird, dass der Ball rund ist und bei konstanter Entfernung aus allen Richtungen des Raums die gleiche Ausdehnung aufweist.

Die Ballentfernung f zur Kamera berechnet sich bei gegebener Winkelausdehnung γ und bekanntem Radius r wie folgt:

[2]engl. body tilt

a) b)

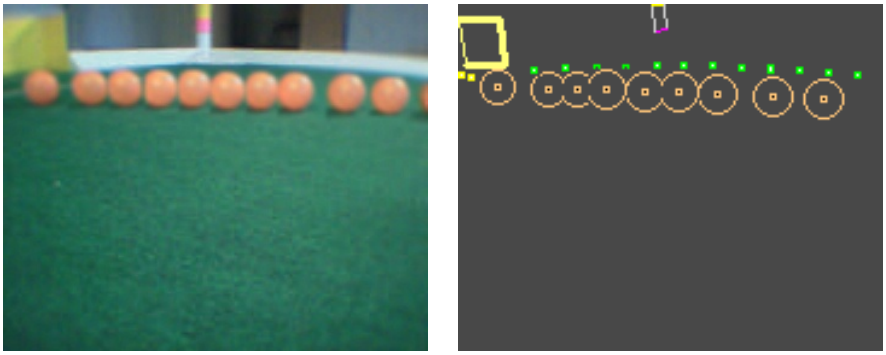


Abb. A.5: *a) Zusammenschnitt aller den Ball enthaltenden Bildausschnitte für einen am Roboter vorbeirollenden Ball im zeitlichen Verlauf.*
b) Die aus den Bildinformationen berechneten und visualisierten Ballperzepte im Zusammenschnitt.

$$f = \frac{r}{\sin(\frac{\gamma}{2})} \tag{A.1}$$

Grundvoraussetzung für die Anwendung dieser Messmethode ist darüber hinaus die genaue Kenntnis über die Ausdehnung des Balls im Bild. Diese ist jedoch manchmal nur schwer ermittelbar, z.B. wenn der Ball am Rand des Bildes bzw. sehr nah vor dem Roboter liegt und dadurch die Krümmung des Balls und damit seine Größe nicht immer eindeutig bestimmbar sind.

A.3.2 Die peilungsbasierte Ballentfernungsbestimmung

Als zweite Möglichkeit zur Bestimmung der Ballentfernung bietet sich an, den Peilungswinkel ausgehend von der Roboterkamera auf das Ballzentrum zu bestimmen. Das Ballzentrum kann aus dem Schwerpunkt aller orangen Pixel im Bild gewonnen werden oder mit Hilfe des Schnittpunktes der Mittelsenkrechten der Tangenten des Ballrandes. Aus dem Winkel $\varphi \triangleq \alpha + \beta$ der Geraden durch den Ballmittelpunkt zum Lot auf den Boden, ausgehend von der Kamera bei gegebener Kamerahöhe h, berechnet sich die Ballentfernung d zum Roboter als:

$$d = \tan(\varphi) * h \tag{A.2}$$

Aufgrund der begrenzten Kameraauflösung wird die Bestimmung des Peilungswinkels φ bei steigender Ballentfernung immer ungenauer. Laufbewegungen verrau-

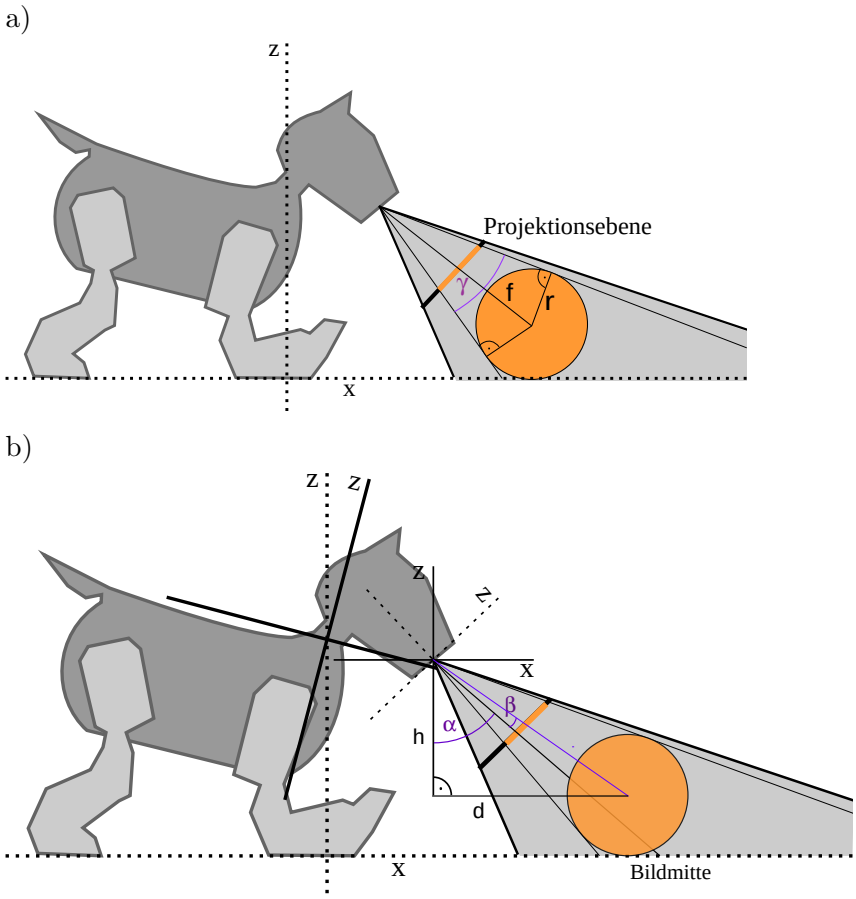


Abb. A.6: *a) größenbasierte Ballabstandsmessung,* $f = r/sin(\gamma/2)$
b) peilungswinkelbasierte Ballabstandsmessung, $d = tan(\alpha + \beta) * h$

schen den Peilungswinkel und eine genaue Bestimmung der Kopf- und Körperneigung des Roboters ist daher nicht möglich. Abb. A.6 zeigt die grafische Berechnung der Ballentfernung.

A.4 Physikalische Eigenschaften des Balls

Um die dynamischen Eigenschaften des Balls der Sony Four Legged League modellieren zu können, sollen nun kurz die grundlegenden physikalischen Eigenschaften des Balls vorgestellt sowie ein kinetisches Modell für die Geschwindigkeitsabschätzung und -propagierung entwickelt werden. Damit kann jederzeit die aktuelle Geschwindigkeit des Balls aus vorhergehenden Ballgeschwindigkeiten approximiert werden.

Aus Vereinfachungsgründen wurde von der zum Teil gekrümmten Rollbahn des Balls abstrahiert; es wurde von einem sich geradlinig bewegenden Ball ausgegangen.

Größe	Wert
Masse [m]	$26g$
Radius [r]	$43mm$
Trägheitsmoment [I]	$32.05 * 10^{-6} kgm^2$

Tab. A.2: *die physikalischen Ballgrößen*

Damit ergeben sich für die kinetische Energie E_{Kin}, für die Rotationsenergie E_{Rot} sowie für die Gesamtenergie E_{Ges} bei gegebener Geschwindigkeit v folgende Berechnungsvorschriften:

$$E_{Kin} = \frac{1}{2}mv^2 = 0.0130kg * [v]^2 \tag{A.3}$$

$$E_{Rot} = \frac{1}{2}I\omega^2 \tag{A.4}$$

Mit $\omega = \frac{1}{r}v$ und $I = \frac{2}{3}mr^2$ eingesetzt folgt für E_{Rot}:

$$E_{Rot} = \frac{1}{2r^2}Iv^2 = 0.0087kg * [v]^2 \tag{A.5}$$

Die Gesamtenergie E_{Ges} des Balls beträgt somit

$$E_{Ges} = E_{Kin} + E_{Rot} = 0.0217kg * [v]^2 \tag{A.6}$$

Nun soll die Verzögerung des Balls, verursacht durch den Teppich, experimentell ermittelt werden. Diese ist später unverzichtbar, um aus einer gegebenen Ballgeschwindigkeit v_0 unter der Annahme keinerlei weiterer äußerer Einwirkungen die Ballgeschwindigkeit zum Zeitpunkt t_i möglichst exakt vorauszuberechnen. Es wird ein physikalisches Modell zugrunde gelegt, bei dem sich die Geschwindigkeit v_t zum Zeitpunkt t bei einer Anfangsgeschwindigkeit von v_0 und einer seitdem vergangenen Zeit Δt folgendermaßen verhält:

$$v_t \approx v_0 e^{c\Delta t} \tag{A.7}$$

Dabei ist c experimentell zu ermitteln. Es wird folgender Versuchsaufbau erstellt: Der zunächst in Ruhe befindliche Ball rollt von einer Schussrampe mit einer Höhe von $0.1m$ auf das Spielfeld hinab. Damit sind die anfängliche Ballgeschwindigkeit sowie

die Richtung sehr exakt und wiederholbar festgelegt. Der Ball hat für $h = 0.1m$ die Anfangsenergie:

$$E_{Pot} = mgh = 0.0255\frac{kgm^2}{s^2} \tag{A.8}$$

$$-E_{Pot} + E_{Kin} + E_{Rot} = 0 \tag{A.9}$$

Aus (A.9) und mit (A.6) nach v umgestellt folgt, dass der Ball am Ende der Rampe eine Geschwindigkeit von etwa $1.08\frac{m}{s}$ besitzt. Nach etwa 3.5 Sekunden kommt der Ball auf dem Spielfeld zum Stehen. Mithilfe der Exponentialregression, angewandt auf die Messwerte, ermittelt man für c einen Wert von etwa -0.9. Bei einer Anfangsgeschwindigkeit von v_0 berechnet sich die Geschwindigkeit v_t zum Zeitpunkt t demnach als:

$$v_t \approx v_0 e^{-0.9\Delta t} \tag{A.10}$$

wobei Δt die seit der letzten Berechnung verstrichene Zeit darstellt. Abb. A.7 stellt Gleichung A.10 grafisch dar.

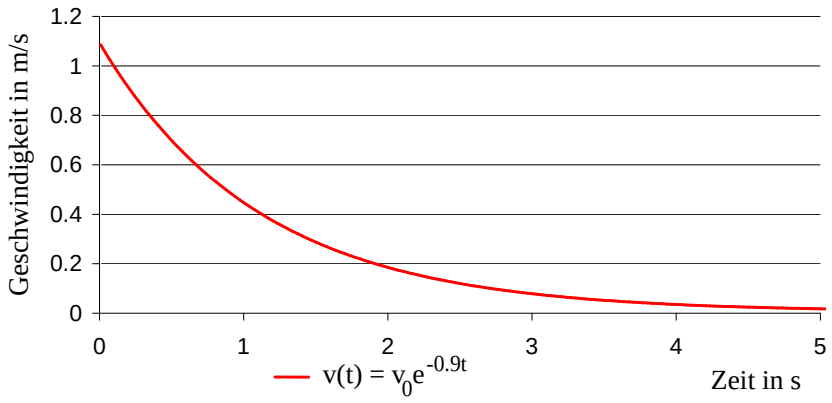


Abb. A.7: *der zeitliche Verlauf der Ballgeschwindigkeit*

A.5 Selbstlokalisierung im RoboCup

Die Lokalisierung des Roboters auf dem Feld ist sowohl für die Positionierung des Balls auf dem Spielfeld als auch für die Handlungsplanung erforderlich. Nur durch eine gute Lokalisierung können die relativ zum Roboter erfassten Ballperzepte in einem

weiteren Schritt in globale Koordinaten des Feldes umgerechnet werden. Besonders wichtig ist die Berechnung der globalen Ballposition für die Teamkommunikation. Die Teamkommunikation ist für die Kooperation der Roboter einer Mannschaft elementare Bedingung. Per WLAN werden u.a. errechnete Ballpositionen von einem Roboter zu einem anderen übertragen, z.B. wenn für einen Roboter der Ball nicht sichtbar ist. Im Weiteren werden zwei der gebräuchlichsten Lokalisierungsverfahren des German Teams vorgestellt.

A.5.1 Prozessmodellbasierte Selbstlokalisierung

Die prozessmodell- oder auch odometriebasierte Selbstlokalisierung basiert auf der Tatsache, dass der Roboter selbst Daten über seine aktuelle Fortbewegungsart in Form von Translations- und Rotationsgeschwindigkeiten besitzt. Diese Tatsache macht es möglich, dass der Roboter aus seiner letzten Position und seiner momentanen Geschwindigkeit auf seine aktuelle Position schließen kann. Diese Positionsbestimmung wird mit größerer Zeitdauer jedoch immer unzuverlässiger, da sich Abweichungen der beabsichtigten von der ausgeführten Bewegung im Laufe der Zeit aufaddieren. Ohne Aktualisierung der modellierten Roboterposition unter Verarbeitung von Sensordaten wird die Diskrepanz zwischen modellierter und realer Position innerhalb weniger Sekunden sehr groß. Für kurze Zeitintervalle stellt die odometriebasierte Selbstlokalisierung jedoch eine sinnvolle Ergänzung zu sensorischen Lokalisierungsverfahren dar, denn sie ermöglicht es dem Roboter, auch bei verdeckten Landmarken oder bei ungünstiger Blickrichtung die Lokalisierung für eine gewisse Zeit aufrechtzuerhalten. Ein Beispiel für eine gute Odometrie stellt die Situation dar, bei der sich der Roboter mit einem gegriffenen Ball dreht und - eine gute Vermessung der Bewegungsgeschwindigkeiten und keine Hindernisse vorausgesetzt - nach Vollendung der Drehung ohne neue Sichtinformation den Ball in die richtige Richtung schießt. Allerdings stellen Kollisionen und Traktionsverlust ein Problem für odometriebasierte Verfahren dar [24, 25].

A.5.2 Monte-Carlo Selbstlokalisierung

Die Monte-Carlo Selbstlokalisierung, erstmals vorgestellt durch Fox et al. in [29, 30], beruht auf der Verwendung von Multihypothesenmodellen [6]. Grundlage dafür ist die Verwendung eines Bayesfilters. Die Wahrscheinlichkeitsdichte für die Roboterposition und -ausrichtung wird durch Partikel repräsentiert [10] wie weiter oben bereits

beschrieben. Der Roboter berechnet bei Eingang von Sensorinformationen z_t^l zum Zeitpunkt t die Wahrscheinlichkeit $p(z_t^l|r_t^i)$ für jeden einzelnen Partikel i. Anhand von $p(z_t^l|r_t^i)$ wird jedem Partikel seine Gewichtung π_t^i zugeordnet:

$$\pi_t^i \propto p(z_t^l|r_t^i) \tag{A.11}$$

Um Bewertungssprünge der Partikel gering zu halten, erfolgt in der Selbstlokalisierung des German Teams eine schrittweise Anhebung bzw. Verringerung der Bewertung einzelner Partikel [5]. Man erhält die gefilterte Gewichtung $\hat{\pi}_t^i$ des i. Partikels zur Zeit t. Die Positionsbestimmung wird dadurch stabiler, insbesondere bei kurzzeitig falscher Sensorinformation. Sei π_t^i die auf Basis von Gleichung A.11 ermittelte Gewichtung. $\hat{\pi}_t^i$ berechnet sich dann nach:

$$\hat{\pi}_t^i = \begin{cases} \hat{\pi}_{t-1}^i + 0.1 & \text{falls } \pi_t^i > \hat{\pi}_{t-1}^i + 0.1 \\ \hat{\pi}_{t-1}^i - 0.05 & \text{falls } \pi_t^i < \hat{\pi}_{t-1}^i - 0.05 \\ \pi_t^i & \text{sonst} \end{cases} \tag{A.12}$$

In jedem weiteren Durchgang erfolgt eine Selektion der Partikel relativ zu ihrer Bewertung. Nach der Propagierung der ausgewählten Partikel unter Berücksichtigung von Odometriedaten beginnt der Algorithmus im nächsten Schritt wieder von vorn. Die Kondensation der einzelnen Partikel an Punkten hoher Aufenthaltswahrscheinlichkeit im zeitlichen Verlauf zeigt Abb. A.8. Die Stärke dieses Verfahrens liegt in seiner Fähigkeit, den Roboter innerhalb kürzester Zeit (3 Sekunden) mit einer Genauigkeit von etwa 5-10 cm zu lokalisieren, selbst wenn die Position vorher völlig unbekannt[3] war. Ein Nachteil dabei ist jedoch, dass das Verfahren unstetig arbeitet. Wenn sich Positionswahrscheinlichkeiten ändern, kommt es zu Positionssprüngen.

A.6 Methoden zur Filterung von Messdaten

Es werden an dieser Stelle einige Ansätze für einfache Ballmodellierungen vorgestellt, denen gemeinsam ist, dass sie ausschließlich positive Ballinformationen verarbeiten, also Informationen über gesehene Bälle.
Zunächst ist es interessant zu erfahren, welche Daten der Roboter nach dem Bildverarbeitungsprozess über den Ball erhält und wie brauchbar diese Daten sind.

[3] ist die Anfangsinformation über die eigene Position gleich null, spricht man auch vom „kidnapped robot“

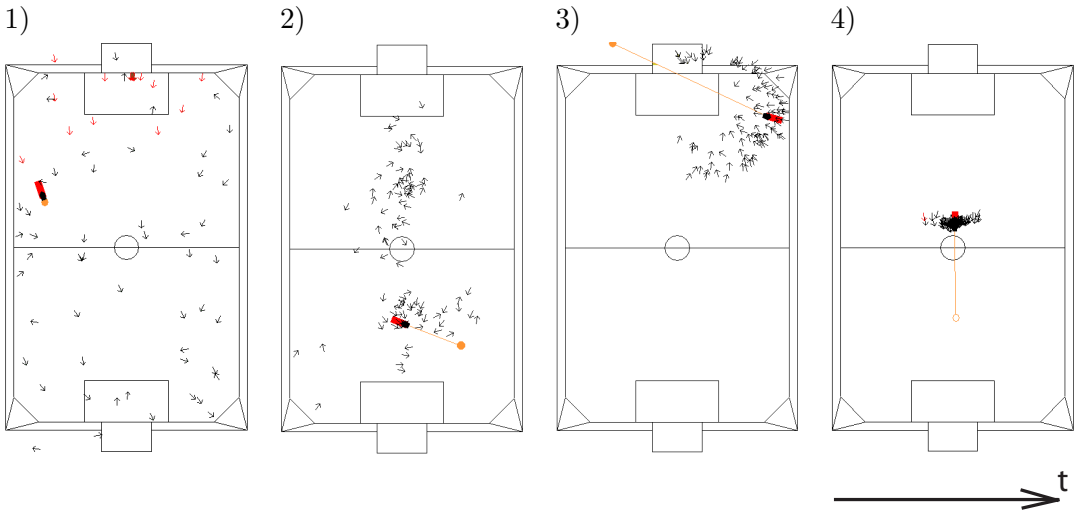


Abb. A.8: Selbstlokalisierung mit 100 Partikeln [5]:
1) Gleichverteilung unmittelbar nach der Initialisierung
2) Beginn der Konvergenz
3) Entstehung eines Gebiets erhöhter Wahrscheinlichkeit
4) Nach kurzer Zeit stellt sich eine sehr hohe Konzentration der Partikel auf einen bestimmten Bereich ein

A.6.1 Keine Filterung

Die wohl einfachste Möglichkeit einer Ballmodellierung besteht darin, die Ballperzepte unverarbeitet, also zustandslos, in Ballpositionen umzurechnen. Diese Methode erweist sich als sehr reaktiv, da keinerlei Rauschfilterung vorgenommen wird. Dieser Vorteil ist aber zugleich auch Nachteil, weil der Roboter keine Unterscheidung von verrauschten zu unverrauschten Daten treffen kann. Jedes neue Ballperzept aktualisiert die momentane Ballposition ohne Rücksicht auf bisheriges Wissen oder auf die physikalischen Eigenschaften des Balls.

A.6.2 Tiefpassfilter

Balllokatoren, die mehrere Ballperzepte auf die eine oder andere Art aufsummieren, um damit Rauschen herauszufiltern, werden an dieser Stelle als Tiefpassfilter bezeichnet. Zwei wichtige dieser Tiefpassfilter sollen kurz skizziert werden.

A.6.2.1 Mittelwertfilter

Eine bewährte Methode für den Umgang mit verrauschten Sensordaten ist die Tiefpassfilterung. Dabei werden n-Tupel von Ballperzepten zusammengefasst und der Mittelwert aus allen Ballperzepten gebildet. Je nach Anzahl der zu mittelnden Ballperzepte wird die Varianz der modellierten Ballpositionen geringer. In der Abb. A.9 wird die aus einer solchen Mittelung resultierende Ballposition und -geschwindigkeit im Vergleich zu unbehandelten Daten gezeigt. Es ist deutlich weniger Rauschen in den Daten zu erkennen. Leider repräsentieren diese gefilterten Daten immer nur einen veralteten Teil der Sensorinformationen und machen den Roboter damit weniger reaktiv für plötzlich auftretende Änderungen. Zudem erfordert diese Art der Ballmodellierung eine höhere Anzahl an Ballperzepten, um die Filterung durchführen zu können. In der Praxis kann jedoch nicht zu jeder Zeit garantiert werden, dass eine bestimmte Anzahl von Ballperzepten vorhanden ist.

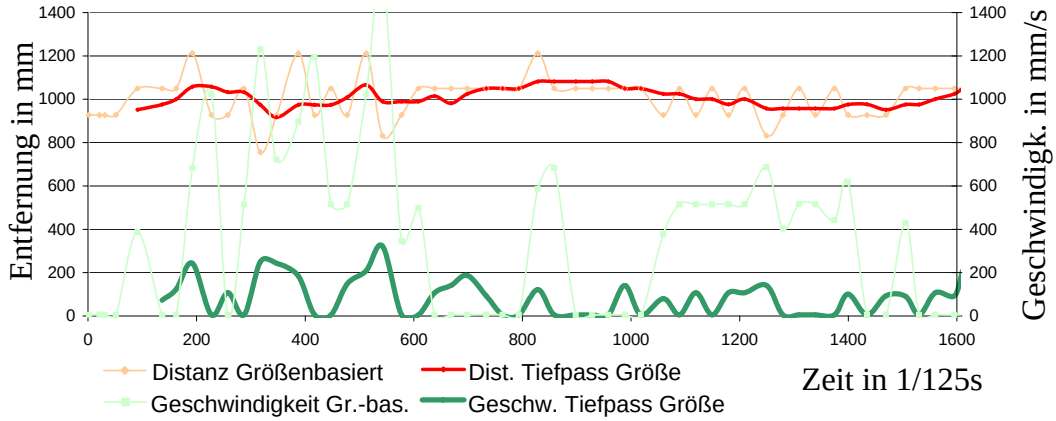
A.6.2.2 PID - Regler

Ein PID - Regler verwendet für die Berechnung der Ballpositionen drei Komponenten P, I und D. Der proportionale Anteil berechnet sich aus der Abweichung der vermuteten von der beobachteten Ballposition und wird mit P gewichtet. Der Integralanteil berechnet sich im diskreten Fall aus der Summe aller Differenzen von beobachteter zu vermuteter Ballposition und sorgt damit dafür, dass vorhandenes Wissen aus der Vergangenheit über Abweichungen der gemessenen Ballpositionen zu den vermuteten Ballpositionen nicht verworfen wird. Der differentiale Anteil misst, um wieviel sich die Differenz zwischen vermuteter und beobachteter Ballposition im letzten Durchgang verändert hat. Er wird mit D gewichtet, die Wichtung des Integralanteils erfolgt durch I. Die eigentliche Berechnung erfolgt iterativ, d.h. die modellierten Balldaten des $n+1$. Taktes hängen von denen des n. Taktes ab. Die Berechnung erfolgt dabei über diskrete Zeitintervalle, weil der Roboter die Ballperzepte nur alle 40 ms (ERS-210) bzw. 33 ms (ERS-7) erfassen kann. Sei d_t die Differenz zwischen alter modellierter Ballposition $\hat{x}_{t-1}$ und neuem Ballperzept z_t, dann berechnet sich die neue modellierte Ballposition $\hat{x}_t$ nach der Formel:

$$\hat{x}_t = \hat{x}_{t-1} + P * d_t + I * \sum_{k=0}^{t} d_k + D * (d_t - d_{t-1}) \tag{A.13}$$

Dabei sind P, I, D die Gewichte der einzelnen Summanden der PID-Summe. Je

a)



b)

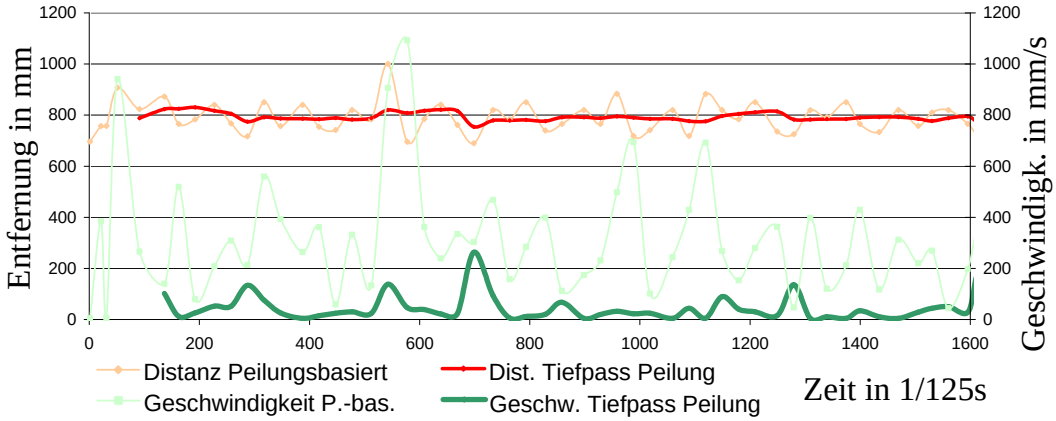


Abb. A.9:
a) Tiefpassfilterung/Mittelwertberechnung aus jeweils fünf größenbasierten Ballperzepten, b) Tiefpassfilterung/Mittelwertberechnung aus jeweils fünf peilungsbasierten Ballperzepten. Die hellroten bzw. -grünen Graphen repräsentieren ungefilterte Daten, dunkelrot bzw. -grün repräsentieren gefilterte Daten.

nach Höhe der Gewichte, die vorzugsweise zwischen 0 und 1 liegen sollten, ist die modellierte Ballkurve eher reaktionsträge, schwingungsfreudig oder reaktiv für Änderungen der Geschwindigkeit des zu beobachtenden Objektes. Im Ergebnis erhält man eine harmonisch geglättete, modellierte Ballkurve wie in Abb. A.10. Das Finden der Gewichte P,I,D stellt dabei eine Schwierigkeit dar. Da es sich aber um eine Funktion mit drei Freiheitsgraden handelt, ist der Lösungsraum überschaubar und durch Gradientenabstieg ermittelbar. Aufgrund dieser Einfachheit ist dieses Verfahren in seiner Anwendung jedoch beschränkt. Es kann, wie beim Perzept - Balllokator, immer nur eine Ballhypothese modelliert werden. Einen Konfidenz-Indikator für die Sicherheit bzw. die Unsicherheit der modellierten Ballposition – wie beim Kalmanfilter – gibt es nicht.

A.6.3 Lineare Regression

Bei der linearen Regression werden die letzten n Messdaten durch eine Gerade interpoliert. Die Summe der Fehlerquadrate zwischen Interpolationsgerade g und Messpunkten z soll für die Zeitpunkte t_1 bis t_2 minimal sein. Eine Variation der Berechnung besteht darin, ältere Messwerte weniger stark zu gewichten als aktuelle. Für g mit Gewichtungsfunktion π gilt:

$$\min \leftarrow \sum_{t=t_1}^{t_2} \pi(t)(g(t) - z_t)^2 \tag{A.14}$$

Wie bei vielen Filterverfahren ist auch bei der linearen Regression ein Kompromiss einzugehen zwischen möglichst vielen Sensordaten über die letzten Ballpositionen für die Filterung und einer möglichst zeitnahen Berechnung der Ballgeschwindigkeit, welche nur dann möglich ist, wenn man sich auf wenige zeitnahe Ballpositionen bei der Berechnung beschränkt. Wie der Name andeutet ist die lineare Regression nur für lineare Prozesse anwendbar. Über den Korellationskoeffizienten aus Zeit und Ort können Rückschlüsse über die Genauigkeit der Berechnung gezogen werden.

A.6.4 Votum-basierte Filterung

Die Votum-basierte Filterung verwendet für die Ballmodellierung die letzten n Ballperzepte. Das Verfahren besitzt gewisse Ähnlichkeit zur linearen Regression, jedoch mit dem Unterschied, dass Punktepaare z_i, z_j, die die Ballperzepte repräsentieren, zur Geraden $g_{i,j}$ verbunden werden. Jede dieser Geraden erhält von den verblei-

benden Perzepten Bewertungen[4], die umso höher ausfallen, je kleiner die Beträge der Abstände der Perzepte zur Geraden sind. Die Votings aller an der Evaluierung beteiligten Punkte werden gewichtet aufaddiert. Weiter in der Vergangenheit zurückliegende Punkte haben weniger Einfluss auf die Evaluierung der Hypothese als aktuellere. Die Gerade $g_{i,j}$ mit der höchsten Bewertung wird zur Abschätzung der aktuellen Ballposition verwendet. Die Abb. A.11 repräsentiert verschiedene Orts-Zeit-Geraden, die evaluiert werden. Sei $g_{i,j}$ die Gerade, die durch Verbinden der Perzepte z_i und z_j entsteht. Weiterhin sei die Menge der zu verwenden Ballperzepte z_t durch t_1 und t_2 festgelegt und sei π_t das Gewicht von z_t. Die Zielfunktion über alle $t_1 \leq i < j \leq t_2$ beschreibt sich wie folgt:

$$\max \leftarrow vote(g_{i,j}) \triangleq \sum_{t=t_1}^{t_2} \frac{1}{\pi_t (g_{i,j}(t) - z_t)^2 + 1} \tag{A.15}$$

A.6.5 Perzept-Validity-Filter

Um einige der Hauptschwächen der bisherigen Balllokatoren zu beseitigen, hat der Autor eine alternative Form der Balllokalisierung entwickelt. Problem der bisherigen Verfahren war zumeist, dass die Güte der Ballperzepte selbst nicht bestimmt werden konnte. Die Idee war dabei die Entwicklung eines Balllokators, der bei verrauschten Ballperzepten sowohl Filterfunktionen übernimmt und bei unverrauschten Ballperzepten trotzdem reaktiv bleibt.

Der Validity-Balllokator bestimmt in diesem Sinne vor der Modellierung ein Gütemaß für die Ballperzepte, auch Validität genannt. Dies wird durch die doppelte Messung der Ballentfernung möglich. Die Annahme ist dabei, dass nur bei korrekter Messung die Ballentfernungsermittlung durch das peilungsbasierte Verfahren mit der größenbasierten Messung übereinstimmt. Dem liegt die Vermutung zugrunde, dass sich viele der Fehler in der größenbasierten Messung nicht direkt auf Fehler in der peilungsbasierten Messung auswirken. Die Fehler beider Messverfahren sind meist unterschiedlicher Natur, wie z.B. unvollständig gesehene Ballränder oder falsch berechnete Roboter- bzw. Kopfneigungen. Deshalb ist eine kleine Übereinstimmung beider Messungen ein Indikator für fehlerhafte Sensordaten. Verrauschte Sensorinformationen können als solche erkannt und in der Modellierung hinreichend berücksichtigt werden.

[4] engl. votes

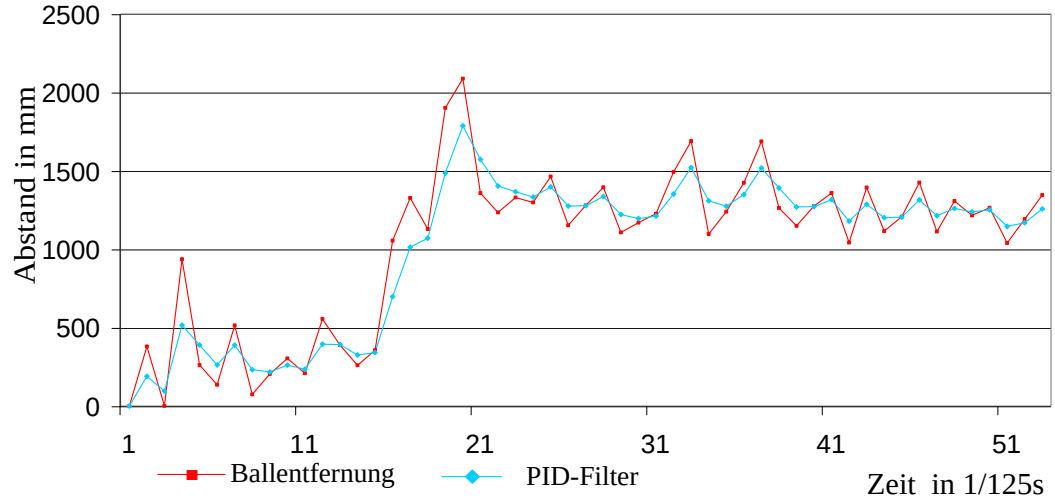


Abb. A.10: *PID-Filterung eindimensionaler Balldaten;*
blau: Ballperzepte, rot: modellierte Ballpositionen;
Gewichte: $P = 0.5$, $I = 0.0$, $D = 0.0$

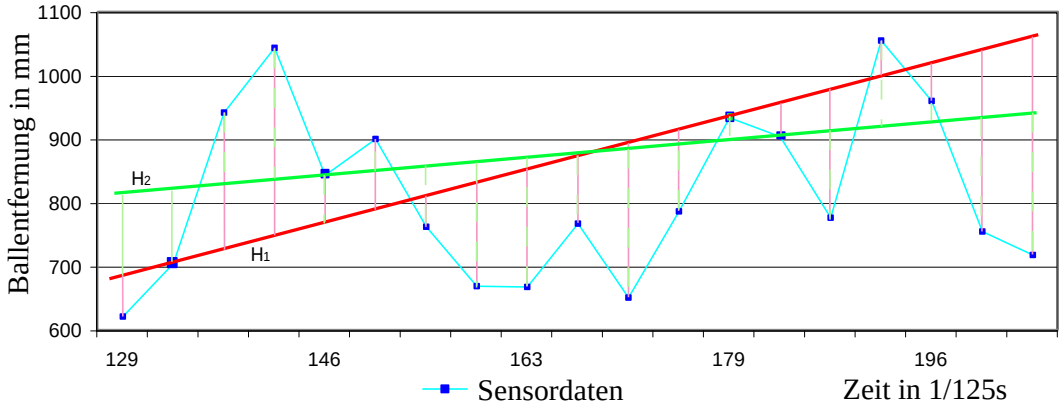


Abb. A.11: *Beispiel für Votum-basierte Filterung;*
blau: Ballperzepte; rot, grün: Verbindungsgeraden, die über die Abstände zu den Ballperzepten gewichtet werden

Die Ballvalidität $val(t)$ zum Zeitpunkt t wird definiert als:

$$val(t) \triangleq \min\left(\frac{z_{t_{peilung}}}{z_{t_{größe}}}, \frac{z_{t_{größe}}}{z_{t_{peilung}}}\right) \tag{A.16}$$

Wenn das Verhältnis zwischen peilungsbasierten $z_{t_{peilung}}$ und größenbasierten $z_{t_{größe}}$ Ballperzepten etwa 1 ist, weist dies auf eine hohe Genauigkeit der Sensordaten hin. Von Verhältnissen größer als 1 wird der Kehrwert gebildet, um $val(t)$ auf Werte zwischen 0 und 1 zu normieren.

Bei gegebenem $\hat{x}_{t-1}$ für den Ballzustand in $t-1$ sowie den Ballperzepten $z_{t_{peilung}}$ und $z_{t_{peilung}}$ zum Zeitpunkt t berechnet sich der neue abgeschätzte Ballzustand $\hat{x}_t$ als:

$$\hat{x}_t = (1 - val(t)) * \hat{x}_{t-1} + val(t) * \left(\frac{z_{t_{peilung}} + z_{t_{größe}}}{2}\right) \tag{A.17}$$

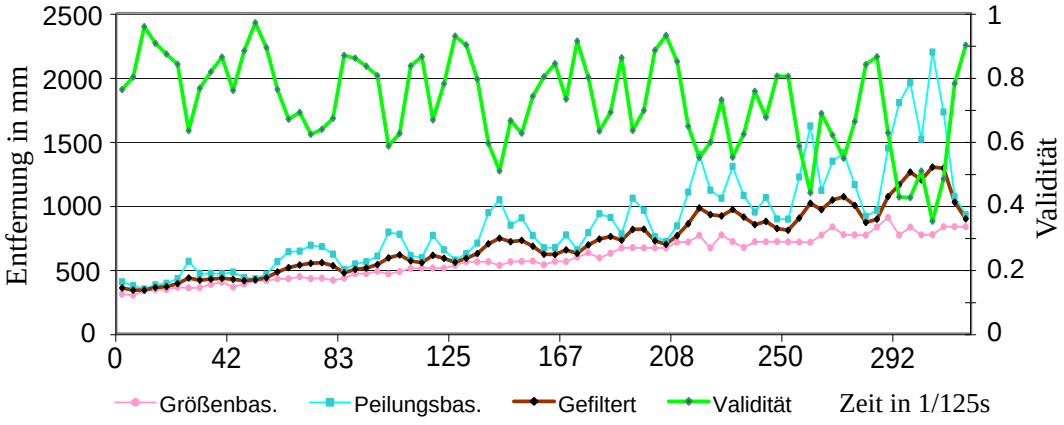


Abb. A.12:

Die Vorteile dieses Verfahrens bestehen darin, dass es sehr einfach ist, ohne komplizierte Berechnungen auskommt und dennoch sehr effizient ist. Die Betrachtung der Ballvaliditäten kann sich des Weiteren auch bei der Berechnung der Ballgeschwindigkeiten als nützlich erweisen. Es wird dabei eine lineare Regression durchgeführt bei der die Anzahl der benötigten Eingangsdaten variabel und von den Perzeptvaliditäten abhängt. Liegen einige hoch bewertete und zeitlich aktuelle Ballperzepte vor, so kann die Ballgeschwindigkeit durch Verwendung von wenigen Ballperzepten bestimmt werden. Das bietet sich vor allem für sich wenig bewegende Spieler, wie

z.B. den Torwart, an. Für Feldspieler, die normalerweise viel in Bewegung sind, werden mehr Ballperzepte betrachtet und die Geschwindigkeit durch lineare Regression ermittelt.

In Algorithmus 5 wurde das bis hierher beschriebene Verfahren implementiert. Es gibt zudem auch andere Möglichkeiten, die Perzeptvaliditäten zur Bestimmung der Ballgeschwindigkeit zu benutzen. So kann die lineare Regressionsgleichung dahingehend angepasst werden, dass die Ballperzepte mit den Validitätswerten gewichtet werden und in die Regressionsgleichung eingehen. Es wird damit erreicht, dass stark verrauschte Ballperzepte weniger Einfluss auf die Geschwindigkeitsberechnung haben als kaum verrauschte.
Die Verwendung von mehreren Sensordatenquellen hat im Ergebnis den Vorteil, dass eine effiziente Abschätzung der momentanen Sensorgenauigkeit möglich ist. In Kombination mit anderen Filterverfahren, wie z.B. einem Kalmanfilter, kann somit die Anpassung der Messfehlermatrizen direkt zur Laufzeit anstatt a-priori beim Entwurf des Filterverfahrens geschehen. In diesem Sinne stellt die Kombination solcher Verfahren mit Kalmanfiltern bzw. RBPFs eine interessante und vielversprechende Aufgabe dar.

Algorithm 5 Berechnung der Geschwindigkeit durch lineare Regression bei variabler Messwerteanzahl

```
1: for i := 1, ..., N do
2:    if i = 1 then
3:       averageValidity[1] =
         (validity[1] + validity[2] + validity[3])/3
                                  //Berechne Mittel über den
                                  //letzten drei Ballvaliditäten
4:    else
5:       averageValidity[i] =
         (averageValidity[i - 1] * (i + 1) + validity[i]) / (i + 2)
                                  //Berechne die N mittleren Validitäten
                                  //averageValidity[i] über die letzten
                                  //N + 2 Ballperzeptvaliditäten
6:    end if
                                  //Vergleich, ob die mittleren Validitäten
                                  //größer als ein bestimmter Grenzwert
                                  //sind, der bei steigender Perzeptanzahl
                                  //schrittweise (hier in 0.05er Schritten)
                                  //kleiner wird
7:    if averageValidity[i] > (1 - i * 0.05) then
8:       v = linearRegression(i + 2 Perzepte)
                                  //Berechne die Ballgeschwindigkeit v
                                  //über den letzten i + 2 Ballperzepten
                                  //durch lineare Regression
9:    end if
10: end for
```

Literaturverzeichnis

[1] H. D. Burkhard, H. A. Marsiske. *Endspiel 2050* Heise, 2003.

[2] M. Fujita, H. Kitano. *Development of an Autonomous Quadruped Robot for Robot Entertainment* Autonomous Robotics 5 (1998) 7-18.

[3] Th. Röfer, H.-D. Burkhard, U. Düffert, J. Hoffmann, D. Göhring, M. Jüngel, M. Lötzsch, O. v. Stryk, R. Brunn, M. Kallnik, M. Kunz, S. Petters, M. Risler, M. Stelzer, I. Dahm, M. Wachter, K. Engel, A. Osterhues, C. Schumann, and J. Ziegler. *GermanTeam RoboCup 2003 Technical report* 2003.

[4] M. Jüngel. *A Vision System for RoboCup: Diploma Thesis* Humboldt-Universität zu Berlin, 2004.

[5] T. Röfer, M. Jüngel. *Vision-Based Fast and Reactive Monte-Carlo Localization* Universität Bremen, Humbold-Universität zu Berlin (ICRA-2003).

[6] C. P. Robert. *Monte Carlo Statistical Methods* University of Paris Dauphine, Springer 1999.

[7] C. Kwok, D. Fox. *Map-based Multiple Model Tracking of a Moving Object* Department of Computer Science & Engineering, University of Washington, Seattle, WA, VIII. RoboCup Symposium, 2004.

[8] RoboCup Technical Comittee. *Sony Four Legged Robot Football League Rule Book* (December, 2003).

[9] A. Doucet, N. de Freitas, K. Murphy, S. Russell. *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks* Engineering Dept., Cambridge University & Computer Science Dept., UC Berkeley

[10] A. Doucet, N. J. Gordon, V. Krishnamurthy. *Particle Filters for State Estimation of Jump Markov Linear Systems* IEEE Transactions on Signal Processing, 49(3), 2001.

[11] D. Fox, J. Hightower, L. Liao, D. Schulz; G. Boriello. *Bayesian Filtering for Location Estimation* University of Washington and Intel Research Seattle

[12] G. Welch, G. Bishop. *An Introduction to the Kalman Filter* University of North Carolina at Chapel Hill, 2003.

[13] Y. Bar-Shalom, X. Rong Li, T. Kirubarajan *Estmation with Applications to Tracking and Navigation* John Wiley, 2001.

[14] M. Dietl, J.-S. Gutmann, B. Nebel. *Cooperative Sensing in Dynamic Environments* Universität Freiburg, (IROS 2001).

[15] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, P.-J. Nordlund. *Particle filters for positioning, navigation and tracking* IEEE Transactions on Signal Processing, 50(2), 2002.

[16] S. Thrun, D. Fox, F. Dellaert, W. Burgard. *Particle filters for mobile robot localization*

[17] N. Bergman, A. Doucet. *Markov Chain Monte Carlo Data Association for Target Tracking* IEEE Conference on Acoustics, Speech and Signal Processing, 2000.

[18] J.-S. Gutmann, D. Fox. *An Experimental Comparison of Localization Methods Continued* Sony Corporation, Tokyo, Japan / University of Washington, Seattle, WA, USA (IROS 2002).

[19] F. Dellaert, D. Fox, W. Burgard, S. Thrun. *Monte Carlo Localization for Mobile Robots* Carnegie Mellon University, Pittsburgh PA 15213 / Universität Bonn

[20] D. Fox *Adapting the Sample Size in Particle Filters Through KLD-Sampling* International Journal of Robotic Research (IJRR), pp. 985-1003 (19), 22(12), 2003.

[21] C. F. Olson. *Probabilistic Self-Localization for Mobile Robots* IEEE Transactions on Robotics and Automation, 16(1), 2000.

[22] D. Fox, W. Burgard. *Active Markov Localization for Mobile Robots* Universität Bonn, Carnegie Mellon Univesity, 1998.

[23] N. Barnes, D. Cameron. *Knowledge-based autonomous dynamic colour calibration* 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences) Springer, 2004.

[24] J. Hoffman, D. Göhring. *Sensor-Actuator-Comparison as a Basis for Collision Detection for a Quadruped Robot* VIII. RoboCup Symposium, 2004.

[25] M. J. Quinlan, C. L. Murch, R. H. Middleton, S. K. Chalup. *Traction Monitoring for Collision Detection with Legged Robots* VII. RoboCup Symposium, 2003.

[26] J. Hoffmann, M. Jüngel, M. Lötzsch. *A Vision Based System for Goal-Directed Obstacle Avoidance used in the RC '03 Obstacle Avoidance Challenge* VIII. RoboCup Symposium, 2004.

[27] S. Lenser, M. Veloso. *Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision* In Proceedings of IROS '03, 2003.

[28] J. Bruce, T. Balch, M. Veloso. *Fast and inexpensive color image segmentation for interactive robots* In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00), volume 3, pages 2061-2066, 2000.

[29] F. Dellaert, W. Burgard, D. Fox, S. Thrun. *Using the condensation algorithm for robust, vision-based mobile robot localization* In Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 1999.

[30] D. Fox, W. Burgard, F. Dellaert, S. Thrun. *Monte Carlo localization: Efficient position estimation for mobile robots* In Proc. of the National Conference on Artificial Intelligence, 1999.

[31] Z. Khan, T. Balch, F. Dellaert. *A Rao-Blackwellized Particle Filter for Eigen-Tracking* Georgia Institute of Technology College of Computing

[32] C. Andrieu, A. Doucet. *Particle Filtering for partially observed Gaussian state space models* University og Bristol, University of Cambridge, UK, 2002.

[33] S. Särkkä, A. Vehtari, J. Lampinen. *Rao-Blackwellized Monte Carlo Data Association for Multiple Target Tracking* Helsinki University of Technology, Finland

[34] D. Schulz, D. Fox, J. Hightower. *People tracking with anonymous and id-sensors using rao-blackwellised particle filters.* Intl. Joint Conf. on Artificial Intelligence (IJCAI), 2003.

[35] N. de Freitas. *Rao-Blackwellised particle filtering for fault diagnosis.* IEEE Aerospace, 2002.

[36] A-V. I. Rosti, M. J. F. Gales. *Rao-Blackwellized Gibbs Sampling For Switching Linear Dynamical Systems* Tech. Rep. CUED/FINFENG/TR.461, Cambridge University Engineering Department, 2003.

[37] C. Fraley, A. E. Raftery. *How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis* Technical Report No. 329, Department of Statistics University of Washington, Seattle, USA, 2004.

[38] G. Henrion. *Beispiele zur Datenanalyse* Berlin, Dt. Verlag der Wissenschaften, 1988.

[39] K. Fukunaga. *Introduction to Statistical Pattern Recognition* London, Academic, 1990.

[40] C. T. Kwok. *Robust Real-time Perception for Mobile Robots, Dissertation Thesis* University of Washington, USA, 2004.

[41] M. Lötzsch, J. Bach, H.-D. Burkhard, M. Jüngel. *Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL* 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences) Springer, 2004.

[42] M. Lötzsch *XABSL - A Behavior Engineering System for Autonomous Agents, Diploma Thesis* Humboldt-Universität zu Berlin, 2004.

[43] U. Düffert *Modellierung und Optimierung von Roboterbewegungen, Diplomarbeit* Humboldt-Universität zu Berlin, 2004.

[44] T. Laue, T. Röfer. *A Behavior Architecture for Autonomous Mobile Robots Based on Potential Fields* VIII. RoboCup Symposium, 2004.