Partikelbasierte Ballmodellierung in einem Team autonomer, vierbeiniger Roboter

Particle-Based Ball Modeling in a Team of Autonomous, Four-Legged Robots

Diplomarbeit

im Studiengang `Informatik`

angefertigt am Fachbereich Informatik

der Technischen Universität Darmstadt

von `Jörg Brose`

Darmstadt, 31. März 2008

Betreuer: Prof. Dr. Oskar von Stryk, Max Risler

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfaßt habe.

Darmstadt, den 31. März 2008

Zusammenfassung

Mit der Verbreitung der kabellosen Kommunikation wird der Datenaustausch zwischen mehreren Robotern einfacher und öffnet nun den Weg zu komplexerem Teamwork. Die kooperative Objektmodellierung auf der Grundlage von Perzepten mehrerer, autonomer Agenten ist ein Teil davon. Für diese Aufgabe werden in der vorliegenden Arbeit einige Modellierungsverfahren präsentiert und ein Ansatz auf Basis des Austausches von Partikeln, die Position und Geschwindigkeit sowie eine Kovarianz des zu modellierenden beweglichen Objektes enthalten, erläutert. Dieser Ansatz wird innerhalb der RoboCup Four-Legged-League, die eine komplexe dynamische Umgebung unter Echtzeitbedingungen bietet, implementiert und dessen Effektivität und Nutzen ausgewertet.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Problembeschreibung 1

1.1.1 Umgebung RoboCup 2

1.1.2 GermanTeam 5

1.1.3 Zielstellung 5

1.2 Stand der Forschung 5

1.2.1 Vorhandene Lösungen 5

1.2.2 Verwandte Arbeiten 6

2 Grundlagen **7**

2.1 Objektmodellierung 7

2.2 Kalman-Filter 7

2.3 Ballmodell 10

2.4 Kommunikation 13

2.5 Selbstlokalisierung 14

2.6 Teamball 14

3 Partikelbasierte Teamballlokalisierung **16**

3.1 Idee 16

3.2 Einfluß von Fehlern 17

3.3 Vertrauen in die verschiedenen Partikel 17

3.4 Konsequenzen für das Verhalten 18

4 Theoretische Betrachtung der Methoden **20**

4.1 Gauß-Verteilung 20

4.2 Kalman-Filter 22

4.3 Multiple Hypothesen 23

4.4 4D- gegen 2D-Auswahl 25

5 Implementierung und Experimente **26**

5.1 ParticleTeamBallLocator 26

5.2 Implementierung der Gauß-Verteilung 32

5.3 Implementierung des Kalman-Filter 33

5.4 Implementierung der multiplen Hypothesen 38

5.5 Bestätigter Teamball 40

6 Ergebnisse und Ausblicke **44**

6.1 Mess- und Beobachtungsergebnisse 44

6.2 Ergebnisse bei Wettbewerben 45

6.2.1 Ergebnisse bei der GermanOpen 2008 in Hannover . . 45

6.2.2 Ergebnisse bei dem RoboCup 2008 in Suzhou, China . 46

6.3 Andere Anwendungsgebiete 47

6.3.1 Humanoide Roboter 47

6.4 Ausblick auf mögliche Weiterarbeit 49

6.4.1 Verfeinerung der mathematische Grundlagen 49

6.4.2 Weitere Sensoren 49

Literaturverzeichnis **50**

Kapitel 1

Einleitung

1.1 Problembeschreibung

Die Zusammenarbeit mehrerer autonomer Agenten wird, wie in vielen Informatikbereichen, auch auf dem Gebiet der mobilen Robotik zunehmend wichtig. Der Einsatz von Robotersystemen verbreitet sich in der Industrie [20] und auch Modelle für den privaten Gebrauch sind inzwischen auf dem freien Markt verfügbar. 2007 gab es 6,5 Millionen Roboter im weltweiten Einsatz, für das Jahr 2011 wird diese Zahl schon auf mehr als 18 Millionen geschätzt. Da die untere Grenze der Preisspanne beständig sinkt und die drahtlose Vernetzung von Systemen dank Funknetzwerken, UMTS und weiteren Techniken Standard geworden ist, gibt es eine steigende Zahl vernetzter Roboter in vielen Anwendungsgebieten, nun zunehmend auch mehrere Roboter in kooperativer Zusammenarbeit und ein funktionierendes Teamwork wird ein wichtiges Thema.

Ein Einsatzort für Teamwork ist die Umweltmodellierung. Ein einzelner Roboter für sich hat nur ein begrenztes Wissen über seine Umwelt. Die Anzahl

und Art der Sensoren sowie der Aufbau der Umwelt samt beweglicher und unbeweglicher Hindernisse bestimmt welchen Teilraum der Umgebung aktuell wahrgenommen werden. Zusätzlich zu der Eingeschränktheit des aktuellen Wissens führen auch Messungenauigkeiten zu Fehlern in der Umgebungswahrnehmung. Das Einbinden von Wissen anderer Roboter über die Umwelt kann dies stark verbessern.

Eine detaillierte Modellierung der Umwelt ist für ein autonomes Verhalten unerlässlich. Die Voraussage der Veränderungen in der Zukunft ist sehr wichtig. Ein einzelner Agent hat jedoch nur ein durch die Zahl und Qualität seiner Sensoren begrenztes Wissen. Deshalb kann auf diesem Gebiet der Einsatz von Teamarbeit sehr nützlich sein. Wenn Informationen über die Umwelt ausgetauscht und abgeglichen werden, führt das in der Regel zu einem besseren Weltmodell. Von anderen Robotern empfangene Informationen über Objekte, wie etwa Position, Größe oder Bewegung, führen zu einer höheren Präzision bei schon bekannten Objekten oder zur Hinzufügung neuer Objekte ins Weltmodell und damit zu einer Vergrößerung des modellierten Bereiches.

1.1.1 Umgebung RoboCup

Im Folgenden soll ein kurzer Überblick über den Aufbau, die Gestaltung und die Regeln der RoboCup Four-Legged-League (Stand Januar 2008 [1]) gegeben werden.

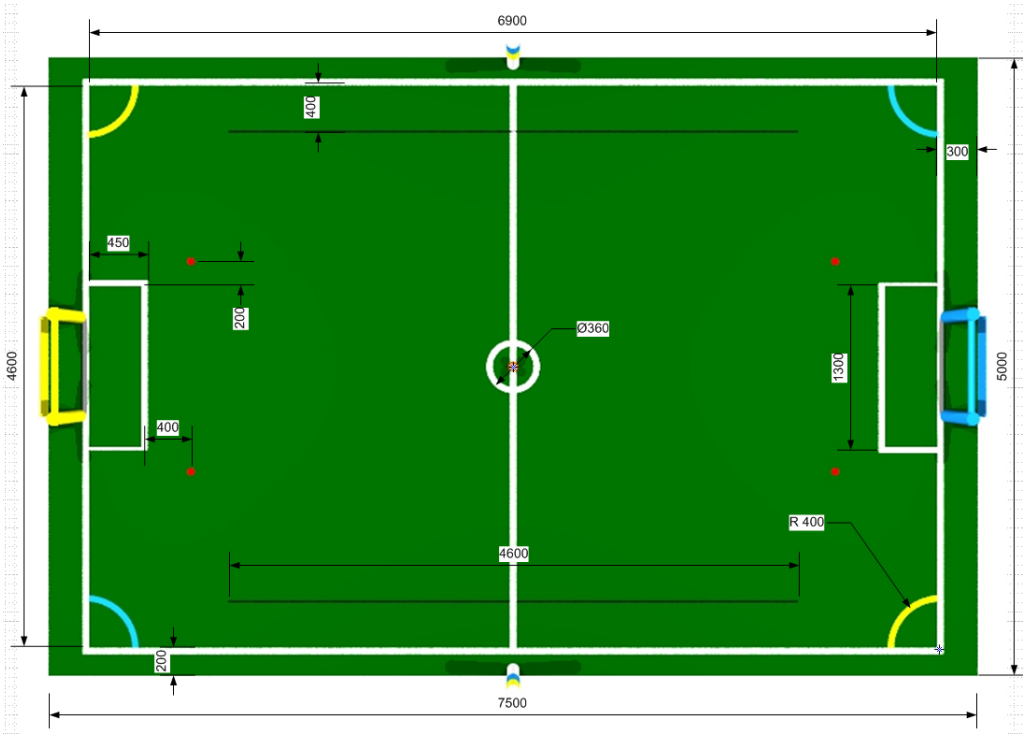

Abbildung 1.1: Skaliertes Diagramm des gesamten Feldes (Dimensionen in mm)

Das Spielfeld (siehe Abb. 1.1) hat die Maße 7,5m × 5m und besitzt Spielfeldlinien analog zu einem normalen Fussballfeld. Es gibt zwei farbige Tore und zwei farbig markierte Zylinder ("Flaggen"), die an den Enden der Mittellinie postiert sind und die Orientierung der Roboter unterstützen soll. Der Ball ist durchgehend orange und von einem festen Durchmesser von 8 cm.

Die Spiele werden zwischen zwei Teams á je 5 Spieler ausgetragen, wobei es je einen Torwart und 4 Feldspieler gibt. Diese Spieler sind autonome Roboter des Typs Sony Aibo ERS-7 [13]. Die Dauer eines Spieles beträgt 2 Halbzeiten á 10 Minuten.

Eine Menge von Spielregeln beschreibt Fouls, die der Schiedsrichter mit Zeitstrafen ahndet, wie z.B. für zu langes Festhalten (und damit Blockieren) des Balls ("Ball-Holding"), die Positionierung von Verteidigern (ausschließlich des Torwarts) im eigenen Strafraum ("Illegal Defender") oder das Drücken von anderen Spielern ("Player-Pushing")

Der RoboCup bietet seit 1997 [19] ein besonderes Testfeld für das Entwickeln neuer Modellierungsverfahren, da alle Modellierungen aufgrund der Konkurrenz mit anderen Teams möglichst effizient, reaktiv auf Veränderungen und robust gegen verrauschte Sensordaten sein müssen. Insbesondere gibt es vier Aufgabenbereiche die für den Erfolg in der Liga effektiv gelöst werden müssen:

- Verarbeitung von Sensordaten
- Modellierung der Umwelt
- Verhaltens- und Aktionsplanung
- Ausführung von Bewegungen

Die Bestimmung der Ballposition und -geschwindigkeit ist dabei bei der Modellierung der Umwelt von elementarer Bedeutung, da die Interaktion des Agenten mit dem Ball die einzige Möglichkeit ist um Tore zu erzielen.

Die Anzahl der Roboter in einem Team nimmt über die Jahre hinweg zu und soll im Endeffekt die Größe einer menschlichen Fussballmanschaft betragen. Erste Versuche mit 22 Robotern auf einem Feld wurden im Rahmen des Turniers bereits unternommen, jedoch zeigte sich schnell, dass die Zusammenarbeit innerhalb eines Teams für ein solches Vorhaben verbessert werden muss.

1.1.2 GermanTeam

Diese Arbeit wurde im Rahmen des GermanTeam [17] entworfen. Das GermanTeam ist ein Zusammenschluss von drei deutschen Universitäten, der Humboldt-Universität zu Berlin, der Universität Bremen und der Technische Universität Darmstadt, und nimmt seit 2001 regelmäßig in der RoboCup Four-Legged-League teil. Sämtliche Implementierungen die im Rahmen dieser Arbeit vorgestellt werden, wurden im vorhandenen Quellcode des GermanTeam vorgenommen.

1.1.3 Zielstellung

Ziel dieser Arbeit ist die Erstellung und Evaluierung verschiedener Modellierungsverfahren für eine teambasierte Ballmodellierung im Rahmen der Four-Legged-League des RoboCup. Der Roboter soll dabei in der Lage sein zusätzlich zu seiner internen, auf eigenen Sensoren basierenden Ballmodellierung eine zweite, auf aus der Teamkommunikation gewonnenen Informationen basierende Ballposition zu gewinnen. Diese soll es ermöglichen auch bei fehlenden internen Sensorinformationen eine Theorie über die Ballposition aufstellen zu können, so dass diese im Verhalten während eines Spieles für verschiedene Aktionen (wie zum Beispiel die Ballsuche) benutzt werden können.

1.2 Stand der Forschung

1.2.1 Vorhandene Lösungen

Die vor dieser Arbeit beim GermanTeam benutzte Lösung [8] für den Teamball war die Nutzung der übertragenen Ballposition, bei der der über-tragende

Roboter sich am nächsten zum Ball glaubte. Da diese Position meist die Beste der übertragenen Positionen ist, führte dies zu annehmbaren, jedoch fehleranfälligen Werten, nutzte aber nicht alle zur Verfügung stehende Informationen. In anderen Ligen des RoboCups gab es auch Lösungen, von denen auch einige Grundideen in dieser Arbeit weiterverwendet wurden. In der Midsize League gibt es einen interessanten Ansatz von Pagello, DAngelo und Menegatti [11] mit einem multimodalen Kalman-Filter, der mehrere Filter parallel unterhält und diese über sämtliche Daten seit Beginn der Übertragungen regelmäßig neu berechnet (z.B. bei einer Fusion zweier Filter). Dieser wird effektiv eingesetzt, benutzt jedoch sehr viel Rechenleistung, eine Ressource die in der Four-Legged-League nicht im gleichen Umfang vorhanden ist.

1.2.2 Verwandte Arbeiten

Ein verwandter Forschungszweig zur Teamballmodellierung ist die Erfassung und Verfolgung eines Objektes über mehrere vernetzte Kamerasysteme. Hierbei gibt es feste Kamerapositionen und eine der hauptsächlichen Quellen für Fehlinformationen im RoboCup ist damit nicht vorhanden. Ebenso gibt es viele Untersuchungen zum Thema der Sensorfusion [12][14]. Bei dieser geht es darum Informationen, die von verschiedenen Sensoren kommen, zu einem Gesamtbild zusammenzufügen und wechselseitig zu kontrollieren. Die Teamballmodellierung kann als ein Spezialfall der Sensorfusion betrachtet werden, bei der alle Sensoren vom gleichen Typ (Kamera) sind und auf unterschiedlichen mobilen Systemen angebracht sind.

Kapitel 2

Grundlagen

2.1 Objektmodellierung

Die Grundlage für das Wissen eines Agenten über seine Umgebung ist die Wahrnehmung. Dieses Wissen wird durch Auswahl, Verarbeitung und Interpretation von Sensorinformationen erlangt. Es bildet die Basis für die Entscheidungen des Agenten.

2.2 Kalman-Filter

Der Kalman-Filter behandelt die Problemstellung den Zustand $x \in \mathbb{R}^n$ eines zeitlich diskret kontrolliertem System vorherzusagen, der durch die Gleichung

$$x_k = Ax_{k-1} + Bu_k + w_{k-1},$$

mit der Messung $z \in \mathbb{R}^m$

$$z_k = Hx_k + v_k$$

gegeben ist. Dabei sind:
$u_k \in \mathbb{R}^l$ die Steuereingabe im k-ten Zeitschritt,
$w_k \in \mathbb{R}^n$ das Systemrauschen im k-ten Zeitschritt,
$v_k \in \mathbb{R}^n$ das Messrauschen im k-ten Zeitschritt,
$A \in \mathbb{R}^{n \times n}$ und $B \in \mathbb{R}^{n \times l}$ beschreiben die lineare Systemfunktion und
$H \in \mathbb{R}^{m \times n}$ beschreibt die lineare Funktion der Messung

Der Kalman-Filter (siehe Abb. 2.1) schätzt hierbei den Systemzustand und verarbeitet danach Feedback, das in Form von (gestörten) Messungen vorliegt. Daraus ergeben sich zwei Phasen: Time-Update und Measurement-Update.

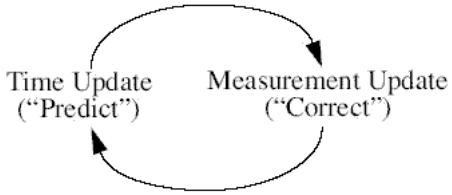

Abbildung 2.1: Der Kalman-Filter-Update-Zyklus

Für die beiden Phasen werden folgende Formeln benutzt:

Time Update Formeln des diskreten Kalman-Filters
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
$P_k^- = AP_{k-1}A^T + Q$

Die *Time Update*-Formeln projezieren den *a posteriori* Systemzustand $\hat{x}_{k-1}$ und die *a posteriori* Kovarianz P_{k-1} von Zeitschritt $k-1$ zu einem *a priori* Systemzustand $\hat{x}_k^-$ und einer *a priori* Kovarianz P_k^-, wobei Q in dieser Rechnung die Kovarianz des Systemrauschens ist.

Measurement Update-Formeln des diskreten Kalman-Filters
$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
$P_k = (I - K_k H) P_k^-$

Der erste Schritt während des *Measurement Update* ist es den *Kalman gain* K_k zu errechnen. Danach wird der eigentlich Systemzustand gemessen um z_k zu erhalten und durch dessen Einarbeitung ein *a posteriori* Systemzustand $\hat{x}_k$ generiert . Als letzter Schritt wird eine *a posteriori* Fehler-Kovarianz P_k errechnet. R beschreibt die Kovarianz des Messrauschens.

Nach jedem *Time* und *Measurement Update* Paar wird der Prozess wiederholt mit dem alten *a posteriori* Systemzustand um einen neuen *a priori* Systemzustand vorherzusagen (siehe Abb. 2.2).

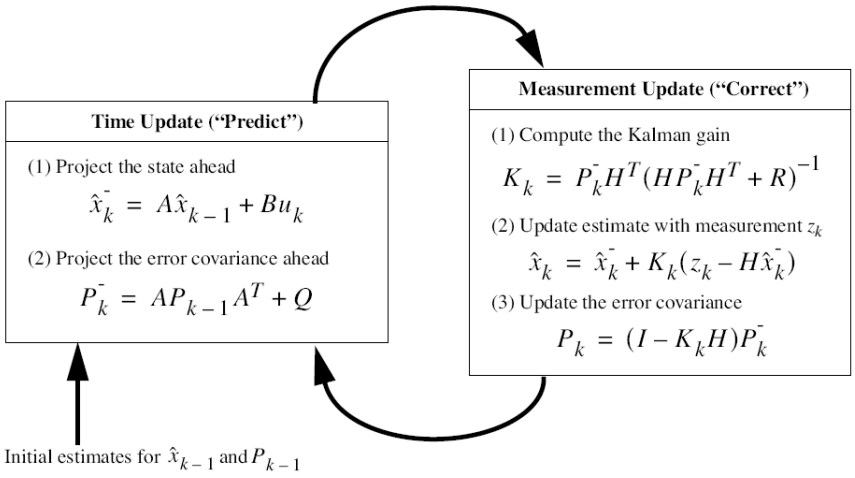

Abbildung 2.2: Der komplette Kalman-Filter-Update-Zyklus mit Formeln

2.3 Ballmodell

Eine Teamball-Modellierung ist stark abhängig von dem im einzelnen Roboter intern verwandten Ballmodell, da diese die Grundlage für die kommunizierten Nachrichten in diesem Zusammenhang bildet.

Das Ballmodell ist ein Modul, welches einen vierdimensionalen Zustand des Balles (Position $(x\ y)^T$ und Geschwindigkeit $(v_x\ v_y)^T$) bereitstellen soll. Dieser ist für die Entscheidungen über das weitere Spielverhalten wichtig und sollte deshalb möglichst genau die Wirklichkeit abbilden.

Als Grundlage zu der Erstellung eines Ballmodells dienten die von einem Vision-Modul gelieferte Berechnung der Ballposition aus den Sensoreninformationen (siehe Abb. 2.3), den Ballerkennungen z_t zu den Zeitpunkten t. Eine zweidimensionale Gauss-Wahrscheinlichkeitsverteilung ist dabei mit der Messung verbunden. Der Mittelpunkt der Gauss-Verteilung befindet sich an der

berechneten Position. Die Breite entlang der relativen Achsen (d: dem Abstand zum Ball, θ: der Winkel des Balls relativ zu dem Roboter) beschreibt die Unsicherheit der Messung entlang dieser Achsen (siehe Abb. 2.3.

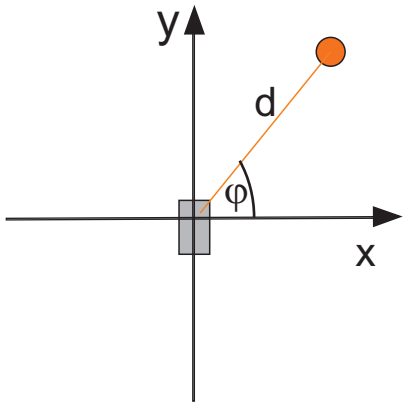

Abbildung 2.3: Beispieldarstellung eines Ballperzeptes in dem kartesischen Roboterkoordinatensystems

Die Geschwindigkeit v_t des Balls zum Zeitpunkt t kann bei exakten und zeitlich äquidistanten Perzepten über den Quotienten der Differenz mit der vorangegangenen Ballerkennung sowie dem Zeitabstand zwischen zwei Ballerkennungen abgeschätzt werden:

$$v_t \approx \frac{z_t - z_{t-1}}{dt}$$

Während eines RoboCup-Spieles verändern sich Ballposition und Ballgeschwindigkeit jedoch fortlaufend. Da die Geschwindigkeit des Balles nicht aus einem einzelnen Bild erkannt werden kann und auch die gesehene Position aufgrund von Fehlerkennungen und kurzzeitig verdecktem Sichtfeld auf den Ball nicht robust ist, muss das Modell vergangene Erkennungen mit verarbeiten und somit einen fließenden, über einzelne Iterationen hinausgehenden Ballzustand liefern.

Da das interne Ballmodell nur die Ballposition relativ zu dem Roboter und nicht absolut auf dem Feld angeben muss, ist im Gegensatz zu der Teamballmodellierung keine Aussage über die absolute Roboterposition nötig, sondern lediglich eine Aussage über die Bewegung des Roboters, damit diese im Ballmodell ausgeglichen werden kann.

In der betrachteten Implementierung wird ein Rao-Blackwellized-Partikel-Filter [21] benutzt (siehe Abb. 2.4), welcher einen Kalman-Filter und einer Monte-Carlo-Lokalisierung vereinigt. Zusätzlich werden Partikel üeber veschiedene Modi erstellt, die die verschiedenen Interaktionsmöglichkeiten des Balles mit den Robotern darstellen.

Diese Ball-Modi ergaben im Vergleich zu den früheren Ansätzen mehrere Vorteile:

- Höhere Flexibilität des Ballzustandes
- Abbildung von Einfluss der Umwelt auf den Ball in die Berechnung möglich
- Verschiedene Gegebenheiten können unterschiedlich gehandhabt werden

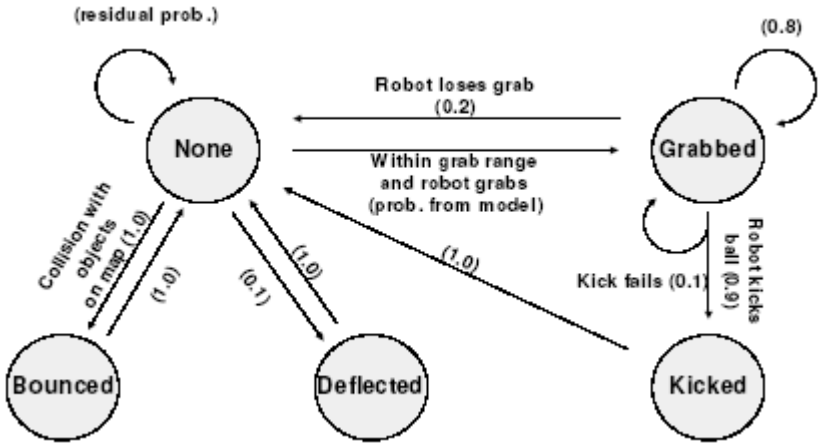

Abbildung 2.4: Übergangsgraph der Ball-Modi des verwendeten Rao-Blackwellized-Partikel-Filters

2.4 Kommunikation

Die im Rahmen dieser Arbeit benutzten Roboter können untereinander über eine integrierte Wireless-LAN-Netzwerkkarte kommunizieren [13]. Über diese Verbindung wird regelmäßig von jedem Roboter ein Datenpaket mit verschiedenen Informationen gesendet, unter anderem Informationen über die von diesem Roboter angenommene Ballposition. Diese Information benutzen wir für diese Arbeit. Die Übertragung der Daten erfolgt nicht zu Nullzeit und benötigt zusätzlich eine Bearbeitungszeit bei dem empfangenden Roboter bis die Informationen intern zur Verfügung stehen. Zusätzlich können Störungen wie dichte Auslastung des Netzwerkes zu zusätzlicher Verzögerung einzelner Pakete führen. Diese Verzögerung führt dazu dass wir im folgenden stets die Aktualität der erhaltenen Informationen prüfen müssen.

2.5 Selbstlokalisierung

Zum Zeitpunkt der vorliegenden Arbeit wird im GermanTeam ein Monte-Carlo Self-Locator verwendet, welcher eine Markov-Lokalisierung mit der sogenannten Monte-Carlo Methode benutzt. In diesem probabilistischem Ansatz wird die Position $pose = (x\ y\ \Theta)^T$ als die Dichte eines Sets von Partikeln modelliert. Jeder Partikel kann als Hypothese betrachtet werden, dass sich der Roboter an Position $(x\ y)^T$ und mit Ausrichtung Θ befindet.

Eine solche Markov-Lokalisierung benötigt sowohl ein Modell der Beobachtungen (von dem Vision-Modul gesehene Landmarken, Linien und sonstige Anhaltspunkte über die Position) als auch ein Modell der Bewegung (Wissen über die möglichen Veränderungen der Roboterposition bei der aktuell ausgeführten Aktion, wie z.B. eine Drehung).

Mit diesem Wissen wird bei der Lokalisierung zuerst alle Partikel im internen Modell an die Bewegung der vorangegangenen Aktion angepasst. Danach werden die Wahrscheinlichkeiten sämtlicher Partikel anhand des Beobachtungsmodells neu berechnet. Anhand dieser Wahrscheinlichkeiten wird ein Resampling, eine erneute Auswahl einer Teilmenge aus den vorhandenen Partikeln, durchgeführt und abschliessend der Durschschnitt der Wahrscheinlichkeitsverteilung ausgerechnet, der die beste Abschätzung der aktuellen Roboterposition darstellt.

2.6 Teamball

Der "Teamball" ist in Verbindung mit der vorliegenden Arbeit und den vorher in dem GermanTeam vorhandenen teambasierten Balllokalisatoren die

Zusammenfassung aller durch Teamkommunikation vorhandenen Informationen über den Ballzustand in einem Ballzustand. Dabei wird versucht eine subjektiv optimale Position zu bestimmen.

Vom Team übermittelte Informationen können auf verschiedene Weise für die Umweltmodellierung benutzt werden, im Rahmen der vorliegenden Arbeit werden zwei dieser betrachtet:

Ergänzung: Die zusätzliche Informationen werden dafür benutzt das Umweltmodell aus den eigenen Sensorinformationen um weitere Objekte zu ergänzen. Im Falle des Teamballes ist dies die Ergänzung der Ballinformationen bei fehlenden roboterinternen Informationen über den Ballzustand.

Korrektur: Die durch die eigenen Sensoren erhaltenen Informationen im Umweltmodell werden durch die zusätzlichen Informationen korrigiert.

Kapitel 3

Partikelbasierte Teamballlokalisierung

3.1 Idee

Wie in Abschnitt 2.3 beschrieben hat die Nutzung von Partikeln, die die möglichen Zustände des Balles beschreiben, einige Vorteile. Die daraus resultierende Aufgabenstellung ist es, diese Vorteile auf die Modellierung des Teamballs zu übertragen und weiter zu nutzen. Dazu wird zusätzlich zu dem vierdimensionalen Zustand auch noch die zugehörige 4×4-Matrix der Kovarianz und das Gewicht des vom einzelnen Roboter ausgesuchten Partikels üebertragen. Dabei ist zu beachten, dass diese Angaben vor der Versendung auf absolute Feldkoordinationen umgerechnet werden müssen.

Somit erhält jedes Teammitglied im Idealfall von jedem anderen Teammitglied den von der lokalen Ballmodellierung ermittelten Partikel, der aussagt in welchem (vierdimensionalen) Gebiet dieses den Ballzustand einschätzt und wie sicher diese Aussage getroffen wurde.

3.2 Einfluß von Fehlern

Aufgrund der Abhängigkeit von anderen Modulen haben Fehler aus diesen Einfluß auf die Leistung und Zuverlässligkeit der Teamballmodellierung. Dies sind vor allem:

- Fehler aus der Ballmodellierung der einzelnen Roboter und
- Fehler aus der Selbstlokalisierung.

Auch wenn die Fehlerquellen unterschiedlich sind haben beide Fehlerarten ähnliche Auswirkungen. Da die interne Ballmodellierung die Ballinformationen relativ zum Roboter benutzt und erst füer die Teamkommunikation diese zu einer Darstellung relativ zum Feld umgerechnet werden führt ein Fehler in der Selbstlokalisierung zu einem Fehler in der übermittelten Ballposition und durch Abhängigkeit der Daten überträgt sich dieser Fehler auch auf die Geschwindigkeit und die Kovarianz des übertragenen Partikels.

3.3 Vertrauen in die verschiedenen Partikel

Wenn von mehreren Teammitgliedern gültige Partikel übertragen wurden, ist das übertragene Gewicht nicht die einzige Möglickeit die Glaubwürdigkeit der einzelnen Partikel zu bewerten. Auch andere Weltumstände sollten dies beeinflußen. Dazu gehören:

- **Abstand des übertragenden Roboters zum kommunizierten Ball:** Je weiter ein Roboter vom Ball entfernt ist, desto ungenauer kann aus dem Kamerabild die Position des Balles bestimmt werden und

der modellierte Ball weicht stärker von der Wirklichkeit ab. Demnach sollte einem Roboter, der näher am (vermeintlichen) Ball positioniert ist mehr Vertrauen geschenkt werden als einem weiter entfernten.

- **Bewegung des Roboters:** Wenn ein Roboter sich bewegt, wird die Ballerkennung abhängig von der Bewegung ungenauer. Die verschiedenen Bewegungsarten (Laufen, Drehen, Kopfbewegung) haben dabei verschiedene Effekte.

- **Anzahl der Sichtungen von Landmarken:** Die Qualität der Selbstlokalisierung ist proportional abhängig von der Anzahl und Neuheit von Landmarken-Sichtungen eines Roboters. Gleichermaßen sinkt auch die Fehlerübertragung ins das Ballmodell und somit dem übertragenen Partikel.

3.4 Konsequenzen für das Verhalten

Eine genaue und zuverlässliche Teamball-Position ermöglicht neue Verhaltensweise im Vergleich zur reinen Benutzung des internen Ballmodells. Zum Zeitpunkt des Startes dieser Arbeit wurde die Teamball-Position nur an einer Stelle des Verhaltens benutzt, dem Suchverhalten. Wenn kein gültiges internes Ballmodell vorhanden ist, versucht der Roboter für ein Zeitfenster von wenigen Sekunden die Teamball-Position anzusehen. In vielen Fällen dauert jedoch das Drehen zu dieser Position länger als dieses Zeitfenster zulässt und weil die Unzuverlässigkeit der bisherigen Teamball-Position die normale Suchroutine effektiver macht als eine längere Drehung wird diese nicht mehr ausgeführt. Wenn die Teamball-Position nun genauer und robuster wird, kann die Effektivität sich zugunsten der Drehung verändern und somit das Zeitfenster erweitert werden.

Zusätzlich dazu ergeben sich jedoch auch noch neue Möglichkeiten zum Einsatz der Teamball-Position. Eine dieser ist die Aufstellung von defensiven Spielern anhand von Team-Informationen.

Kapitel 4

Theoretische Betrachtung der Methoden

Die Ziel dieser Arbeit ist, die Vorteile der partikelbasierten Ballmodellierung auf die Teamballmodellierung zu übertragen. Während die Form der Partikel durch die lokale Ballmodellierung bestimmt wird, ist die interne Weiterverarbeitung der empfangenen Daten vielseitig gestaltbar. Im Rahmen dieser Arbeit werden drei Ansätze untersucht, die schon auf anderen Bereichen der Robotik und Informatik allgemein vielseitige und erfolgreiche Anwendung gefunden haben.

4.1 Gauß-Verteilung

Die Gauß-Verteilung ("Gaussian Distribution") ist ein typisches Modell für Signale und Geräusche in vielen Einsatzgebieten der Informatik. Auch bei der kooperativen Errechnung der Ballposition können wir einen Ansatz [9] benutzen, der auf dieser beruht.

Jeder der von anderen Agenten übertragene Partikel beschreibt dabei eine Gaussche Dichtefunktion um einen bestimmten Punkt (den 'Mittelpunkt' des Partikels) und damit die Wahrscheinlichkeiten möglicher Feldpositionen. Die Dichtefunktion in der Dimension n ist bestimmt durch die Formel:

$f_x : \mathbb{R}^n \mapsto \mathbb{R}, (x_1, ..., x_n) \mapsto \frac{1}{\sqrt{(2\pi)^n}} \exp(\frac{-1}{2} \sum_n^(i = 1) x_i^2).$

Da wir in der vorliegenden Arbeit mit der Position des Balles arbeiten, benötigen wir eine zweidimensionale Funktion. Die zweidimensionale Dichtefunktion (siehe Abb. 4.1) der Normalverteilung mit einem Korrelationskoeffizienten ϱ ist:

$f_x(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\varrho^2}} \exp - \frac{1}{2(1-\varrho^2)}((\frac{x_1-\mu_1}{\sigma_1})^2 - \varrho\frac{x_1-\mu_1}{\sigma_1}\frac{x_2-\mu_2}{\sigma_2} + (\frac{x_2-\mu_2}{\sigma_2})^2)$

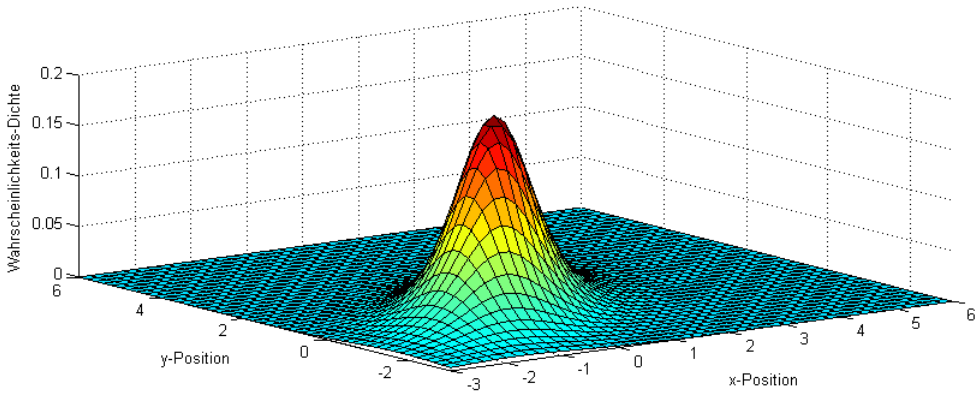

Abbildung 4.1: Zweidimensionale Dichtefunktion

Diese Wahrscheinlichkeitsverteilungen können sich auf dem Feld überschneiden. Wenn dies geschieht, können die Wahrscheinlichkeiten auf der Schnittfläche (siehe Abb. 4.2) addiert werden. Das Ergebnis ist eine komplette Wahrscheinlichkeitsverteilung über dem gesamten Feld mit lokalen und einem absoluten Maximum.

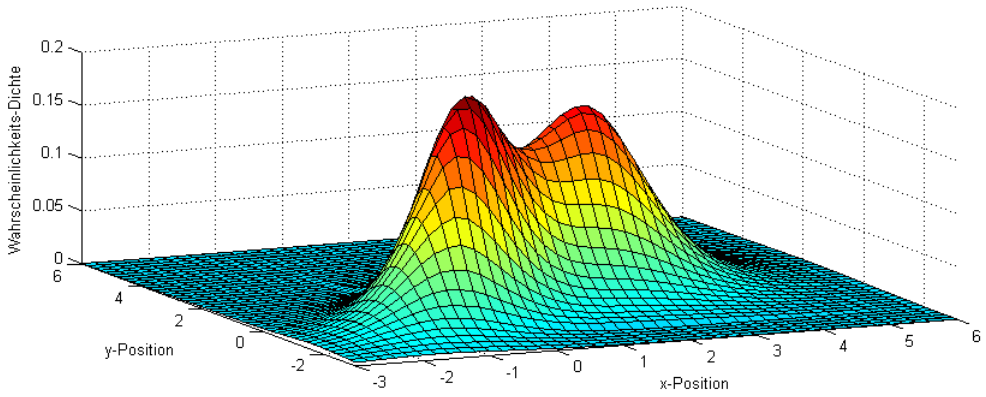

Abbildung 4.2: Schneiden zweier Verteilungen

Das absolute Maximum kann nun mit recht geringem Aufwand bestimmt werden, da der größte Teil des Feldes von keinem übertragenen Partikel abgedeckt wird und somit die Suche auf die Gebiete um die Mittelpunkte der Partikel eingeschränkt werden kann. Dieses bildet die mit dieser Methode zu ermittelnden 'beste' Ballposition aus den Team-Informationen.

4.2 Kalman-Filter

Ein Kalman-Filter (siehe Kapitel 2.4) bietet Methoden, um viele der Nachteile einer reinen Wahrscheinlichkeitsaddition auszugleichen. Techniken die auf einem Kalman Filter basieren haben auf verwandten Anwendungsgebieten gezeigt, dass sie eine robuste und genaue Objektposition modellieren können. Da wir in unserem vorliegenden Fall eines Teamballes keine unabhängige Zeitschritte haben sondern im Idealfall die Trajektorie eines Balles abbilden, kommt uns vor allem die Haupteigenschaft des Kalman-Filters zugute: Der Kalman-Filter bildet den aktuellen Systemzustand nicht allein anhand der

in diesem Zeitschritt verfügbaren Informationen (wie der Gauß-Verteilungs-Ansatz), sondern benutzt den errechneten Systemzustand samt Varianz aus dem letzten Zeitschritt weiter und damit rekursiv auch alle vorangegangenen.

In dem Fall einer Teamballmodellierung sind jedoch im Gegensatz zu der normalen Anwendung des Kalman-Filters mehrere Messungen pro Zeitschritt vorhanden (jeder übertragene Partikel gilt als eine Messung). Dies führt zu mehreren Möglichkeiten der Verarbeitung, vor allem bei der Anzahl der Messungen die mit dem Kalman-Filter verarbeitet werden. Von diesen möchten wir zwei betrachten:

1. Nur eine Messung pro Zeitschritt wird verarbeitet. Dies sollte wenn möglich der subjektiv beste übermittelte Partikel sein.
2. Alle Partikel werden als voneinander unabhängige Messungen innerhalb eines Zeitschritts angenommen und für jede ein Measurement-Update durchgeführt.

Obwohl diese 2 Möglichkeiten unterschiedliche Auswirkungen haben können wir bei guter Objektorientierter Programmierung die selben Kalman-Filter-Methoden für beide Ansätze verwenden.

4.3 Multiple Hypothesen

Da die verfügbaren Informationen für die Teamballmodellierung aus voneinander unabhängigen Quellen kommen und des öfteren stark und gleichmäßig voneinander abweichen (z.B. bei der Fehllokalisierung eines Roboters), kann es sein dass sie mehr als eine Theorie der Ballposition auf dem Feld abbilden. Da ein einfacher Kalman-Filter jedoch von nur einer Theorie die durch alle

Informationen erzeugt wird ausgeht ist dieser in dem Aufgabenfeld dieser Arbeit nur bedingt einsetzbar.

Wie verschiedene Versuche zeigen [10] [11] ist es aber auch möglich mehrere Kalman-Filter parallel auf verschiedenen Theorien arbeiten zu lassen. Dieses 'Multihypothesis Tracking' (kurz: MHT) repräsentiert die Dichtefunktion eines Objektzustandes B zur Zeit t als Summe von Gaußfunktionen:

$$B(x_t) = \sum_i \omega_t^i \eta(x_t; \mu_t^i, \Sigma_t^i) \tag{4.1}$$

Jede dieser Gaußfunktionen wird durch einen Kalmanfilter aktualisiert und erhält eine Gewichtung ω_t^i je nach Wahrscheinlichkeit (eng: likelihood) dieser Theorie bei den gegebenen Sensordaten. Durch diese Fähigkeit multimodale Verteilungen repräsentieren und bewerten zu können sind MHT-Algorithmen in mehr Fällen einsetzbar als ein einfacher Kalmanfilter.

In dem Einsatzgebiet des Teamballs muss jedoch das MHT-Grundmodell angepasst werden. Da wir nicht nur einen Satz von Sensordaten pro Zeitschritt (genauer: gemessene Ballpositionen von Robotern) haben sondern mehrere Sätze die jeweils nicht für alle Thesen einsetzbar sind. Durch den Abstand des Mittelpunkts des übertragenden Partikels zu dem Mittelpunkt der These lassen sich Rückschlüsse auf die Einsetzbarkeit schließen, da der Einbezug eine Ballbewegung suggestieren würde, die physikalisch durch Robotereinwirkung nicht möglich ist.

Da wir bei starken Fehlern in der Selbstlokalisierung oder anderen Modulen längere, gleichbleibende Abstände zwischen den Übertragungen verschiedener Roboter bekommen können und wir nicht sofort einschätzen können, welche die reale Position ist (falls es überhaupt eine der übertragenen gibt, die eine richtige Abschätzung bietet), ist diese Methode mit einigen Vorteilen verbunden:

- Wir können mehrere mögliche Positionen speichern
- "falsch" erscheinende Daten können weiterverwendet werden ohne das Ergebnis sofort zu verändern
- frühere Daten können unter neuen Gesichtspunkten neu bewertet werden

4.4 4D- gegen 2D-Auswahl

Außer der Position des Balles ist aus dem Ballmodell Informationen über zwei weitere Dimensionen enthalten: Die x- und y-Achsen-Geschwindigkeit. Die Frage die unmittelbar aus diesem Fakt hervorgeht ist: Kann die Geschwindigkeit auch bei einer Teamballmodellierung betrachtet werden? In dem vorliegenden Fall ist diese Frage schnell zu beantworten: Unter den in der Four-Legged-League herrschenden Umständen ist eine 4D-Betrachtung des Balles nicht sinnvoll. Schon bei der auf nur einem Roboter basierenden Ballmodellierung ist die Geschwindigkeit schwer abschätzbar. Dies liegt zum einem an dem verhältnismäßig schlechtem Kamerabild des Sony Aibos (350.000 Pixel [13]) und andererseits auch an den nichtlinearen Bewegungen des standardmäßig festgelegten Balles der Liga (siehe [3]). Schwankungen in der von der Ballerkennung zurückgegebenen Entfernung zum Ball (durch Erkennen von nur Teilen des Balles) treten in der Realität sehr häufig auf. Dies führt zu sprunghaften Veränderungen der wahrgenommenen Geschwindigkeit. Diese Umstände ergeben ein zu ungenaue Grundlage für eine erfolgreiche Teammodellierung.

Kapitel 5

Implementierung und Experimente

In diesem Kapitel wird eine Beispielanwendung für einen Partikel-basierten Teamballlokator im Kontext der Four-Legged-League vorgestellt. Dabei wird nicht ein spezieller Ansatz als Basis benutzt, sondern mit einem Grundgerüst für einen Partikel einlesenden Locator gearbeitet, in den verschiedene Methoden zur Verarbeitung dieser integriert werden. Dies ermöglicht das Testen und Evaluieren von mehreren Ansätzen in einer konstanten Entwicklungsumgebung. Des Weiteren werden Anwendungen der verbesserten teambasierten Lokalisierung des Balls im Spielverhalten der Agenten vorgeschlagen.

5.1 ParticleTeamBallLocator

Der im dritten Kapitel vorgestellte Teamballlokator empfängt aus den Ballmodellen der anderen Agenten auf feldrelative Koordinaten umgerechnete Daten. Da von jedem Agenten A_i mit $i\epsilon\{1..\text{Anzahl der Agenten}\}$ in regelmäßi-

gen Abstand eine Nachricht m_i gesendet wird und jede dieser Nachrichten im Idealfall von jedem Teammitglied empfangen wird, ist es effektiver einmal vor dem Senden die roboterrelativen Koordinaten der zu übertragenen Position in absolute Feldkoordinaten umzurechnen. Die Umrechnung geschieht in der internen Ballmodellierung, in diesem Fall dem Rao-Black-Particle-Filter. Es wird eine Nachricht erstellt die aus folgenden aus dem internen Modell entnommenen Informationen besteht:

- Der Ballposition $\begin{pmatrix} x \\ y \end{pmatrix}$
- Der Ballgeschwindigkeit $\begin{pmatrix} v_x \\ v_y \end{pmatrix}$
- Der Ball-Kovarianz $P = \begin{pmatrix} p_{x,x} & p_{x,y} & p_{x,vx} & p_{x,vy} \\ p_{y,x} & p_{x,y} & p_{y,vx} & p_{y,vy} \\ p_{vx,x} & p_{vx,y} & p_{vx,vx} & p_{vx,vy} \\ p_{vy,x} & p_{vy,y} & p_{vy,vx} & p_{vy,vy} \end{pmatrix}$
- Einen boolschen Wert ob der Ball gesehen wurde (auch wenn der Ball nicht gesehen wurde, kann das Modell gültig sein)
- Dem Abstand zum Ball d
- Dem Gewicht des Partikels ω

Am Anfang jeder Iteration des Teamballlocators werden diese Informationen für jeden der anderen Agenten aus der Teamkommunikation ausgelesen. Zur Überprüfung der Aktualität jeder der übertragenen Daten werden die Zeitstempel der Nachricht an die lokale Zeit des Agenten angepasst und mit der Systemzeit verglichen, wobei eine Differenz von bis zu zwei Sekunden akzeptiert wird. Zusätzlich wird überpruft ob die übertragene Ballposition

in einem sinnvollen Rahmen liegt (den Feldmaßen). Sind beide Tests positiv verlaufen wird der Partikel in einem Array von Teamballpartikeln gespeichert (siehe Abb. 5.1). Diese Objektklasse wurde speziell für diese Implementierung geschrieben und enthält Speicher für alle oben genannten Informationen.

```
Particle-Test()
{
  for(i=1, ..., numberOfRobots-1)
    if(Partikel ist j"unger als 2 Sekunden AND Ball-Position
        des Partikels liegt innerhalb des Feldes)
        TeamBallParticle ballPosition[i] = message.ballPosition;
    else
        verwerfe Partikel;
}
```

Abbildung 5.1: Pseudo-Code für den Test der übertragenen Partikel

Als nächster Schritt werden nun die Gewichte ω_i der Partikel angepasst. Der Abstand des Agenten zu dem gesehenen Ball vergält sich linear zu der daraus ensteenden Ungenauigkeit der Ballerkennung und wird als Faktor behandelt: $\omega_i^* = d_i * factor_{distance}$. Eine Bewegung des Agenten ($b_{movement}$ sowie eine Bewegung des Kopfes ($b_{headmovement}$) sind falls vorhanden konstant und gehen mit einer konstanten Addition ein: $\omega_i^+ = b_{movement} * const_{movement} + b_{headmovement} * const_{movement}$. Daraus wird dann das neue Gewicht ausgerechnet: $\omega_i^{neu} = \omega_i^* * \omega_i^{alt} + \omega_i^+$.

Der wichtigste Schritt in der Bestimmung der Teamballposition ist die Auswahl dieser aus dem Raum der möglichen Teamballpositionen. Dazu haben wir im vorhergehenden Kapitel 4 mehrere Möglichkeiten vorgestellt und zeigen im Folgenden Implementierungen und deren erste Ergebnisse.

Für die folgenden Ergebnisse wurden außer aufgenommen Daten aus echten Spielen auch folgende zwei Tests benutzt:

1. Standardaufstellung eines fünfköpfigen, verteidigenden Teams mit Blick auf den Ball auf dem Mittelpunkt (siehe Abb. 5.2).

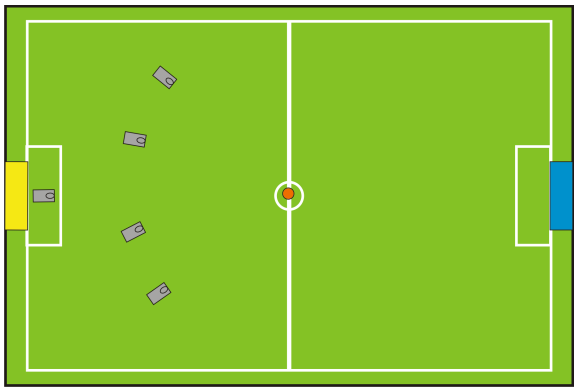

Abbildung 5.2: Standardaufstellung der Roboter (graue Rechtecke, Ellipse symbolisiert den Kopf) mit Ball

2. Wie 1. nur mit einem spielenden Roboter der den Ball bewegt (siehe Abb. 5.3).

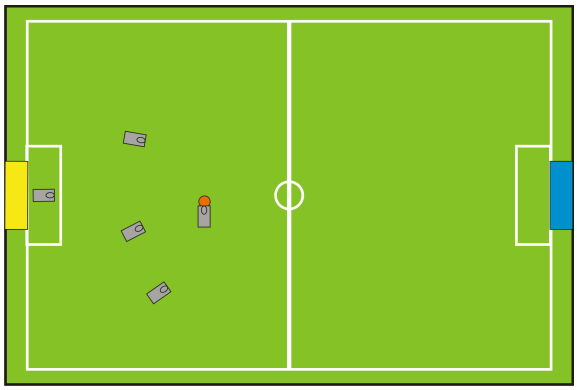

Abbildung 5.3: Standardaufstellung mit einem spielenden Roboter

Die beiden Testfällen und der erwähnten Daten aus echten Spielen wurden in Form von Logfiles aufgenommen, die später über eine Simulation [7] ausgewertet werden konnten (siehe Abb. 5.4).

Der ausgegebene Teamball-Wert wurde dabei verglichen mit den Daten einer Deckenkamera, die die Position der Roboter und des Balles auf dem Feld bis auf 2-3 cm genau ausgeben kann. Da die komplette Weltmodellierung eines jeden Roboters starken Schwankungen unterliegt, geben wir im Folgenden nur einen Durchschnittswert über alle Messungen eines Testes aus. Sämtliche Versuche einen Verlauf des Fehlers über verschiedene Dimensionen (wie z.B: Zeit oder Abstand zum Ball) zu zeichnen haben leider unaussagekräftige Graphen geliefert. Die Durchschnittswerte lieferten jedoch gute Vergleichswerte und waren auch über mehrere Testversuche reproduzierbar.

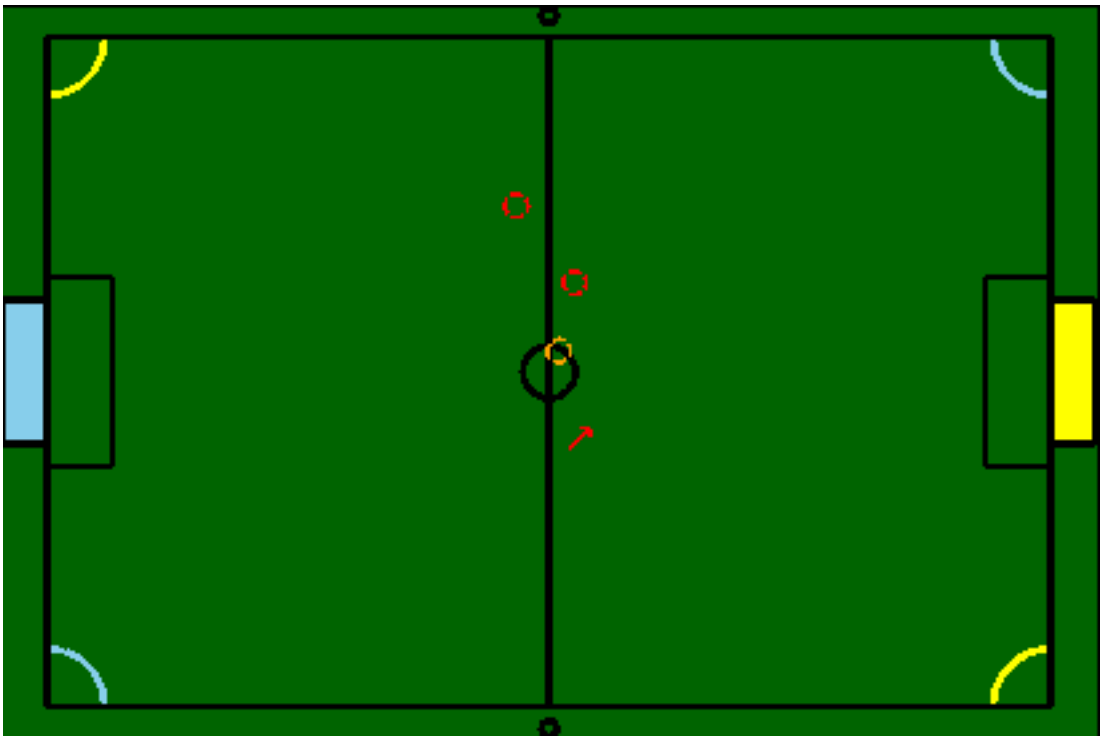

Abbildung 5.4: Übermittelte Ballpositionen (Kreise) und durch Modul gewählte Teamball-Position (orange anstatt rot)

Um einen Vergleich zu dem Zustand vor der vorliegenden Arbeit zu ermöglichen, wurden zuerst Messungen mit dem ursprünglichen Teamballlokator (wie in Kapitel 1.2.1 beschrieben) vorgenommen. Diese ergaben einen durchschnittlichen Fehler von:

- für die Standardaufstellung: 40cm
- für einen spielenden Roboter: 80cm
- für Logfiles aus Spielen: 93cm

5.2 Implementierung der Gauß-Verteilung

Bei der Implementierung des Ansatzes "Gauß-Verteilung" (siehe Abb. 5.5) ist durch das oben beschriebene Grundgerüst außer der Auswahl der besten Position kaum weitere Arbeit notwendig.

```
Gauss-Distribution()
{
  if(at least one valid Teamball received)
    set Teamball-timestamp;
    for each particle
        sum the probabilities of the Gauss Distribution;
    choose most probable position
    set Ball Data with position
  else
    no teamball avaiable
}
```

Abbildung 5.5: Pseudo-Code für Gauss-Ansatz

Die Messungen für den ersten Ansatz ergaben einen durchschnittlichen Fehler

- für die Standardaufstellung: 73cm
- für einen spielenden Roboter: 114cm
- für Logfiles aus Spielen: 133cm

Die Simulationen zeigen zusätzlich, dass die Ausgaben dieser Methode einige Probleme zur Folge haben. Diese sind vor allem:

- Ein isolierter, aber stark gewichteter Partikel hat mehr Einfluss als mehrere, nicht stark gewichtete Partikel die die selbe Position beschreiben

- Das Ergebnis ist unabhängig von Ergebnissen aus vorherigen Iterationen und damit auch extrem sprunghaft von Iteration zu Iteration

- Das Ergebnis ist sehr fehleranfällig, da stark abweichende Positionen auch als Ergebnis herauskommen können.

- Die Methode bringt insgesamt nur wenig Vorteile gegenüber einer reinen Übernahme der Ballposition des Agenten, der am nähsten am Ball ist

5.3 Implementierung des Kalman-Filter

Die Implementierung des wie in Kapitel 2.2 beschriebenen Kalman-Filters sieht von der Struktur wie in Abb. 5.6 aus.

```
kalman-Filter()
{
  time-update for the model;
  if(at least one valid Teamball received)
    set Teamball-timestamp;
    for each valid particle
        measurement-update of the model with this particle;
    return data from model;
  else
  {
    if model was updated not too long ago
    {
        time-update for the model;
        return data from model;
    }
    else
        no teamball avaiable
  }
}
```

Abbildung 5.6: Pseudo-Code für Kalman-Ansatz mit mehreren Measurement-Updates

Bei diesem Ansatz ergaben die Messungen durchschnittliche Fehler

- für die Standardaufstellung: 56cm
- für einen spielenden Roboter: 130cm
- für Logfiles aus Spielen: 215cm

Dabei ist jedoch zu beachten, dass, obwohl manche dieser Werte niedriger als die aus dem ersten Ansatz sind, die Varianz der Fehler auch viel geringer ist. Dies bedeutet zwar dass einerseits der Fehler selten einen Wert mehr als 150% der Durchschnittsfehler annimmt, aber auch dass der Fehler ebenso selten unter 50% desselben fiel, außer es gab für längere Zeit nur ein übertragendes Teammitglied.

In diesem Ansatz werden mit jedem gültigen Partikeln je ein Measurement-Update auf dem Modell ausgeführt. Es werden zwar sämtliche Partikel verwendet und auch über mehrere Iterationen weiterverarbeitet jedoch zeigt sich in den Tests schnell, dass es ein entscheidendes Problem in diesem Ansatz gibt. Wie sich zeigt entsteht durch die Vielzahl der Measurement-Updates ein Ergebnis dass irgendwo zwischen den gesendetet Positionen liegt und bis auf seltene Ausnahmen in keinster Weise deckungsgleich zu einem der übertragenen Partikel ist (siehe Abb. 5.7). Dies ist jedoch nicht sinnvoll, da die in der Praxis am häufigsten vorkommenden Situationen die ist, in der ein oder mehrere Partikel die richtige Position beschreiben und ein oder mehrere Partikel eine falsche Position.

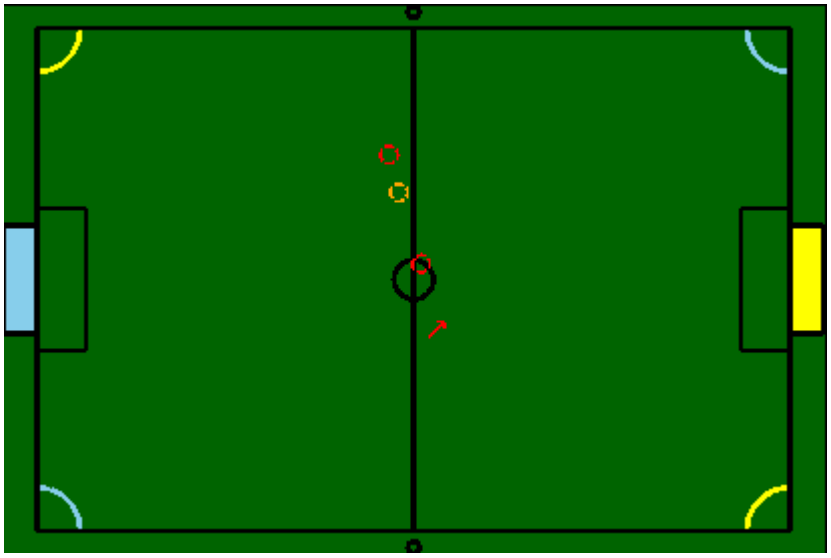

Abbildung 5.7: Durch Kalman (Version 1) übermittelte Ballpositionen (rot) und durch Kalman-Filter gewählte Teamballposition (orange)

Dieses Problem können wir lösen in dem wir in jedem Zeitschritt nur mit einem der Partikel ein Measurement-Update ausführen. Dafür wählen wir jeweils einen besten Partikel analog zu Kapitel 5.2 (siehe Abb. 5.8).

```
kalman-Filter()
{
  time-update for the model;
  if(at least one valid Teamball received)
    set Teamball-timestamp;
    for each valid particle
        sum the probabilities of the Gauss Distribution;
    choose best position;
    measurement-update for the model with best position;
    return data from model;
  else
  {
    if model was updated not too long ago
    {
        time-update for the model;
        return data from model;
    }
    else
        no teamball avaiable
  }
}
```

Abbildung 5.8: Pseudo-Code für Kalman-Ansatz

Die Ergebnisse für diesen Ansatz ergaben

- für die Standardaufstellung: 42cm
- für einen spielenden Roboter: 63cm
- für Logfiles aus Spielen: 75cm

Das Verhalten der ausgegebenen Teamballposition ist um einiges ruhiger und genauer. Allerdings ergeben Beobachtungen des Fehlerverlaufes dass trotz der relativ guten Ergebnisse es nach wie vor einige verbesserungswürdigen Verhaltensweisen der Teamballmodellierung gab, vor allem der sehr träge Trennungsvorgang wenn sich ein vorher mehrmals als bester Partikel-standort ausgewählte Position als falsch erweist. Rückzuführen ist dies unter anderem darauf, dass sämtliche Informationen über gesendete Partikel die in dem zugehörigen Zeitschritt als nicht optimal eingestuft komplett verworfen wurden. Auch bei späterer Bestätigung durch neue Partikel können diese Informationen nicht mehr in die Ausgabe mit einbezogen werden, da sie in keiner Form gespeichert wurden.

5.4 Implementierung der multiplen Hypothesen

Das zuletzt genannte Problem aus dem letzten Ansatz ist bei einer Implementierung des multiplen Hypothesenansatzes nicht mehr vorhanden.

```
Multi-Kalman()
{
  for all existing models
    if (model too old) delete model;
    else time-update;
  if(at least one Teamball received)
    set Teamball-timestamp;
    for each valid particle {
      choose nearest model;
      measurement update of that model with particle; }
    choose best model and reutrn that data;
  else {
    for all existing models
      if (model too old) delete model;
      else time-update for the model;
    if at least one model exists
      return data of best model
    else no teamball avaiable }
}
```

Abbildung 5.9: Pseudo-Code für Multiple-Hypothesen-Ansatz

Dieser Ansatz war der am zu schwierigsten zu programmierende und benötigte eine komplexe Objektorientierte Struktur, lieferte jedoch auch die besten Ergebnisse mit durchschnittlichen Fehlern

- für die Standardaufstellung: 13cm
- für einen spielenden Roboter: 24cm
- für Logfiles aus Spielen: 26cm

Die Ergebnisse dieses Ansatzes erwiesen sich nicht nur als numerisch gut sondern zeigten auch ein erwünschenswertes Verhalten bei Beobachten des Teamball-Verlaufes. Die Vorteile umfassen vor allem:

- Auch als falschëingestufte Informationen sind innerhalb Alternativ-Hypothesen für längere Zeit verfügbar
- Wenn sich eine verfolgte Hypothese als falsch erweist kann es einen schnellen Sprung zu einer anderen Hypothese geben
- Zwei sich annähernde Hypothesen werden verschmolzen mit resultierende stärkeren Gewicht
- Kurzzeitige Fehler müssten länger aktiv sein um sich durchzusetzen gegen fundierte Hypothesen

5.5 Bestätigter Teamball

Während der ersten echten Spielen im Einsatz der GermanOpen 2008 6.2.1 stellte sich heraus, dass es bestimmte Bereiche des Verhalten gibt, bei denen eine fast 100 prozentige Sicherheit des Teamballes nötig ist um darauf entsprechende Aktionen zu starten.

Der errechnete Teamball ist trotz der Verbesserungen wie jedes Modul in der ungewissen Realität jedoch nicht immer korrekt. Besonders wenn nur ein gültiger Teamballpartikel bei der Berechnung der Teamballposition eingeflossen ist, gibt es eine direkte Abhängigkeit zu der Korrektheit der Lokalisierung des sendenden Roboters.

Dementsprechend ist in den oben genannten Fällen eine zusätzliche Aussage über die Korrektheit des Teamballes notwendig. Sehr wahrscheinlich richtig

ist der Teamball wenn während einer Iteration zwei Teamballpartikel von zwei unterschiedlichen Robotern erhalten wurden, die trotzdem ungefähr die gleiche Position darstellen. In diesem Fall sprechen wir in dieser Arbeit davon dass die Position des ersten Roboters von dem Partikel des zweiten Roboters bestätigt wurde.

Es ist sehr unwahrscheinlich dass zwei Roboter die gleiche absolute Balposition übertragen und in beiden der gleiche Fehler enthalten ist. So ist bei einem Fehler in der Lokalisierung dieser auf jedem Roboter als voneinander unabhängig zufällig zu betrachten. Dass dieser Fehler einen relativ zum Roboter richtig gesehenen Ball auf der absoluten Felddarstellung in dem Maße falsch verschiebt dass es genau mit der fehlerhaften Ball-Felddarstellung eines zweiten Roboters übereinstimmt ist vernachlässigbar unwahrscheinlich.

Ein anderer Fall wäre eine Fehlerkennung des Balls. Hier müssen wir wiederum zwei Fälle getrennt betrachten. Zum einen kann die Berechnung der Ballposition aus der Ballerkennung falsch sein, was zum Beispiel aus im Bild verdeckten Teilen des Balls entstehen kann. Hier ist es wieder vernachlässigbar unwahrscheinlich dass zwei Roboter auf unterschiedlichen Positionen und damit unterschiedlichen Blickwinkeln auf den Ball den gleichen Fehler in der Ballerkennung haben oder zwei unterschiedliche Fehler die zur gleichen feldrelativen Ballposition führen.

Eine absolute Fehlerkennung wäre die andere Form eines Fehlers. Diese wird verursacht durch einen vom Ball unterschiedlichen Gegenstand, der als Ball erkannt wird, vor allem auftretend in dunklen Ecken des gelben Tores (siehe Abbildung 5.10), Kleidungsstücken der Zuschauer und der Hand des Schiedsrichters. All diese Fälle werden jedoch in echten Spielen an anderer Stelle abgefangen oder können abgefangen werden (wie z.B. eine gute Farbtabelle, Sichthindernisse auf Roboterhöhe zu den Zuschauern) oder kommen in

Spezialfällen vor die den Teamball unbedeutend machen (Hand des Schiedsrichter auf Kamerahöhe bedeutet meistens eine Herausnahme des Balles vom Spielfeld und Einsetzen an einer anderen Stelle).

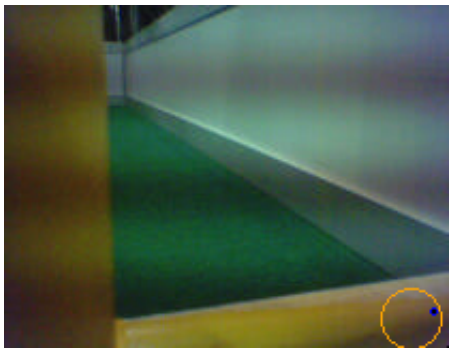

Abbildung 5.10: Gesehener Ball im Tor (verursacht durch schlechte Farbwerte)

Die Implementierung des bestätigten Teamballs erfolgte in den Ansatz der multplen Hypothesen. Dort kann in jeder Iteration des Teamballlocator überprüft werden, ob an einer Hypothese zweimal durch je einen Partikel ein Measurement-Update vorgenommen wurde. Ist dies der Fall wird diese Hypothese als bestätigt für diese Iteration gekennzeichnet und auf jeden Fall als Ausgabe des Teamballes mit einer zusätzlichen boolschen Variable als bestätigt bearbeitet (siehe Abb. 5.11).

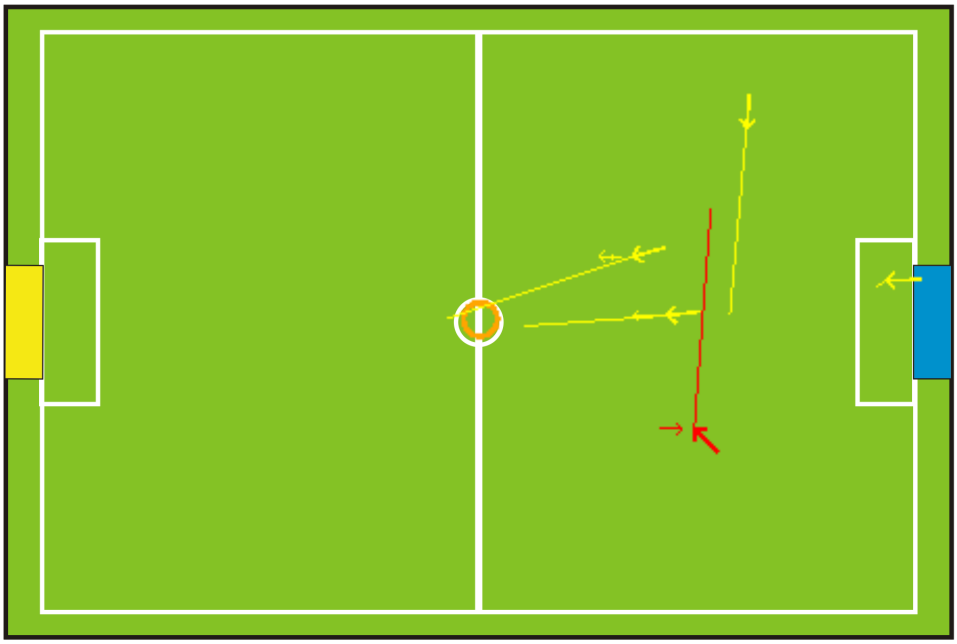

Abbildung 5.11: Der Ball wird von zwei Robotern gesehen und dadurch bestätigt (abgebildet durch größeren orangenen Kreis)

Kapitel 6

Ergebnisse und Ausblicke

In diesem Kapitel soll ein Überblick über die Ergebnisse des Einsatzes des neuen Teamball-Lokators und ein Ausblick auf weitere möglichen Arbeiten in dem Themengebiet gegeben werden.

6.1 Mess- und Beobachtungsergebnisse

Die in Kapitel 5.1 vorgestellten Messversuche ergaben eine deutliche Verringerung des durchschnittlichen Fehlers in der Teamball-Modellierung verglichen von der im Rahmen dieser Arbeit erstellten und dem vorher (siehe Kapitel 1.2.1) vorhandenen Modell. Die letztlich aktive Modellierung mit multiplen Hypothesen (siehe Kapitel 4.3) zeigte in diesem Vergleich eine Verbesserung des durchschnittlich gemessenen Fehlers der Teamball-Position zur von der Deckenkamera erfassten Position in den verschiedenen Kategorien von:

- für die Standardaufstellung: von 40cm auf 13cm (Eine Verbesserung um ca. 307%)

- für einen spielenden Roboter: von 80cm auf 24cm (Eine Verbesserung um ca. 333%)

- für Logfiles aus Spielen: von 93cm auf 26cm (Eine Verbesserung um ca. 357%)

Zusätzlich ergaben Beobachtungen während der Wettbewerbe (siehe folgende Abschnitte) und die Analyse der Spiellogfiles, dass ein Teamball der bestätigt wurde in weniger als einem von hundert Fällen eine größere Abweichung als 50cm von der tatsächlichen Ballposition hatte, was eine verhältnismäßig zuverlässige Basis für Handlungen aufgrund der Teamball-Modellierung darstellte. Dabei lag der durchschnittlich gemessene Prozentsatz an vorhandenen Teamballpositionen in den realen Spielen zwischen 60 und 80 Prozent und der Prozentsatz der bestätigten Teamballpositionen zwischen 40 und 65 Prozent der aktiven Spielzeit (Aufstellung und andere Spielpausen ausgeschlossen).

6.2 Ergebnisse bei Wettbewerben

6.2.1 Ergebnisse bei der GermanOpen 2008 in Hannover

Die GermanOpen 2008 war der erste praktische Einsatzort für den neuen Teamball-Lokator. Ausgetragen wurde der Wettbewerb zwischen internationalen Teams in Hannover. Ab dem ersten Testspiel vorort wurde das das neue Teamball-Lokator Software-Modul angeschaltet. Zu Beginn ergaben sich Probleme wenn der Teamball nur durch einen Partikel im aktuellen Bild bestimmt wird, wie in Kapitel 5.5 beschrieben. Durch Einführung des bestätigt-Status eines Partikels wurde diesem Problem entgegengewirkt und

der Teamball Lokator lieferte gute Ergebnisse während den Folgespielen. Mit dieser Neuentwicklung und weiteren Verbesserungen (siehe [17]) konnte die GermanOpen vom GermanTeam gewonnen werden.

6.2.2 Ergebnisse bei dem RoboCup 2008 in Suzhou, China

Nach dem Erfolg bei der GermanOpen 2008 wurden sämtliche Neuentwicklungen konsequent weiter verbessert (bei dem Teamball Lokator unter anderem die Auswahlwahrscheinlichkeiten der Partikel). Mit diesen Grundlagen konnte auch unter den schwierigen Verhältnissen auf der Austragungsfläche (siehe Abb. 6.1) des RoboCups 2008 (unter anderem gab es es Probleme mit den Frequenzen der Scheinwerfer und dem durch ein Dachfenster eindringenden Tageslicht) wieder ein Erfolg gefeiert werden. Nach schwierigen Duellen mit 9 weiteren Teams aus 7 Ländern wurde das GermanTeam Weltmeister 2008 in der Four-legged-League.

Abbildung 6.1: Ein Spielfeld der Four-Legged-League beim RoboCup 2008 in Suzhou, China

6.3 Andere Anwendungsgebiete

Der im Umfang dieser Arbeit entworfene und implementierte Teamball-Lokator kann durch seinen modularen Aufbau und allgemeine Theorie-Grundlage auf andere Gebiete der Robotik und der künstlichen Intelligenz übertragen werden. Besonders da die bisher verwendete Plattform, der Sony Aibo, seit 2006 nicht mehr hergestellt wird und die zugehörige Liga im Robocup auf eine andere Hardwarebasis (den Nao von Aldebaran Robotics, siehe [16] für Details) umgestellt wird.

Einige dieser Anwendungsgebiete auf die diese Arbeit übertragen werden könnte sollen im folgenden vorgestellt werden.

6.3.1 Humanoide Roboter

Der Sektor der humanoiden Roboter ist ein Teilgebiet der Robotik, das immer größeren Stellenwert im RoboCup bekommt. Neben inzwischen zwei Humanoiden-Ligen (kidSize und teenSize) wird (wie im vorhergehenden Abschnitt erwähnt) die Standard-Platform-League (frühere Four-Legged-League) zur Zeit auf einen humanoiden Roboter umgestellt und eine weitere Simulationsliga, in der humanoide Roboter simuliert gegeneinander antreten, wurde eingeführt. Eine Portierung des Teamballlokators ist deshalb sinnvoll. Ein Team in der Humanoiden Liga sind die Darmstadt Dribblers (siehe [15]), die eine verwandte Grundstruktur zu dem hier verwendeten modularen Programmierungsansatz benutzen, weswegen die Umstellung besonders günstig wäre.

Abbildung 6.2: Zwei humanoide Roboter der Darmstadt Dribblers

Die humanoide Plattform bietet einige Vorteile für dieses Modul:

- Bessere Kamerabilder
- Höher gelegener Blickwinkel und damit genauere relative Ballpositions-Bestimmung
- Leistungsfähigere und aufrüstbare Recheneinheit

Jedoch gibt es bisher auch einige Nachteile:

- Langsamerer Spielablauf und kleinere Teams (zur Zeit drei Roboter pro Team), womit der Teamball weniger Einfluß hat
- Ungenaue und schlecht vorhersehbare Bewegungen, die die absolute Ballpositions-Bestimmung verschlechtern
- Leistungsfähigere und upgradbare Recheneinheit

Mit fortlaufender Entwicklung werden diese negativen Punkte jedoch immer schwächer und die Vorteile immer größer.

6.4 Ausblick auf mögliche Weiterarbeit

In dieser Sektion sollen kurz einige mögliche Ansätze für eine Weiterarbeit an dem Thema gegeben werden:

6.4.1 Verfeinerung der mathematische Grundlagen

Sämtliche feste Werte die zur Berechnung der Teamballposition benutzt werden, können unter anderen Gegebenheiten neu berechnet werden. Dies wäre unter anderem vor allem bei dem Wechsel der Plattform sinnvoll. Auch können neue ausgefeilte Berechnungs- und Messmethoden oder automatische Optimierungsprozesse benutzt werden um genauere Werte zu ermitteln.

6.4.2 Weitere Sensoren

Die Verwendung von Daten von weiteren Sensoren könnte als Ansatz für weitere Partikel benutzt werden. Im GermanTeam gab es einen von David Becker entwickelten Ansatz [18] um über akustische Signale den Abstand zwischen verschiedenen Robotern festzustellen. Dies wäre eine geeignete Information über die man die Abhängigkeit des Teamball Lokators zu der Selbstlokalisierung der anderen Roboter abschwächen könnte.

Literaturverzeichnis

[1] RoboCup Technical Committee, "RoboCup Four-Legged League Rule Book", 2008, avaiable from http://www.tzi.de/4legged/pub/Website/Downloads/AiboRules2008.pdf

[2] C. Kwok, D. Fox, "Map-based Multiple Model Tracking of a Moving Object",Department of Computer Science&Engineering, University of Washington, 2004

[3] J. Brose, C. Werner, "Abschlussbericht des Praktikums: Ballmodellierung durch Rao-Blackwellised-Partikel-Filter", Department of Computer Science, TU Darmstadt, 2007

[4] Sony, "ERS-7M2 Users Guide", elektronisch erhältlich von http://support.sony-europe.com/aibo/

[5] G. Welsch, G.Bishop, "The Discrete Kalman Filter", An Introduction to the Kalman Filter, 2001

[6] F. Dellaert, D. FOx, W. Burgard, S. Thrun, "Monte Carlo Localization for Mobile Robots", Department of Computer Science & Engineering, Canergie Mellon University, 2000

[7] T. Röfer, T. Laue, H.-D. Burkhard, J. Hoffmann, M. Jüngel, D. Göhring, M. Lötzsch, U. Düffert, M. Spranger, B. Altmeyer, V. Goetzke, O. von Stryk, R. Brunn, M. Dassler, M. Kunz, M. Risler, M. Stelzer, D. Thomas, S. Uhrig, U. Schwiegelshohn, I. Dahm, M. Hebbe, W. Nistico, C. Schumann, M. Wachter,, "GermanTeam 2004, Technical Report of the RocoCup-Team in the Sony Legged Robot League 2004", elektronisch verfügbar von http://www.sim.informatik.tu-darmstadt.de/publ

[8] T. Röfer et al "GermanTeam Team Description Paper 2006", elektronisch verfügbar von http://www.sim.informatik.tu-darmstadt.de/publ

[9] Martin Grothaus, Yuri G. Kondratiev and Ludwig Streit, "Regular generalized functions in Gaussian analysis", Universitat Bielefeld, 1997

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

[11] E. Pagello, A. DÁngelo, E. Menegatti,"Cooperation Issues and Distributed Sensing for Multirobot Systems", Proceedings of the IEEE vol 94, 2006

[12] Yu Ding et al"Distributed Sensing for Quality and Productivity Improvements", IEEE Transactions on automation science and engineering vol 3, 2006

[13] Sony, "Sony Aibo ERS-7M2 User's Guide", elektronisch verfügbar von http://support.sony-europe.com/aibo/

[14] R.C. Luo, C.C. Yih and K. L. Su, "Multisensor fusion and integration: approaches, applications, and future research directions", IEEE Sensors J., vol 2, 2002

[15] Martin Friedmann, Karen Petersen, Sebastian Petters, Katayon Radkhah, Dirk Thomas, Oskar von Stryk, "Darmstadt Dribblers: Team Description for Humanoid KidSize League of RoboCup 2008",elektronisch verfügbar von http://www.sim.informatik.tu-darmstadt.de/publ/download/2008-tdp-hum.pdf

[16] Aldebaran Robotics, "Robot Humanoide - Le Monde de Nao-Concept", elektronisch verfügbar von http://www.aldebaran-robotics.com/pageProjetsNao.php

[17] David Becker, Jörg Brose, Daniel Göhring, Matthias Jüngel, Max Risler, Thomas Röfer, "GermanTeam 2008 - The German National RoboCup Team", elektronisch verfügbar von http://RoboCup.informatik.tu-darmstadt.de/publications.php

[18] David Becker, "Aktive Schall-Lokalisation in Teams autonomer, mobiler Roboter und Ansätze zur kooperativen Selbstlokalisierung", elektronisch verfügbar von http://RoboCup.informatik.tu-darmstadt.de/publications.php

[19] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, "RoboCup: The Robot World Cup Initiative", elektronisch verfügbar von: http://www.RoboCup.org/overview/RoboCup-Agent97.pdf

[20] IFR statistics department, "IFR statistic press release", elektronisch verfügbar unter http://www.worldrobotics.org

[21] C. Kwok, D. Fox, "Map-based Multiple Model Tracking of a Moving Object",Department of Computer Science and Engineering, University of Washington, 2004