Computer Vision, Sensorfusion und Verhaltenssteuerung für Fussball-Roboter

Diplomarbeit
bei Prof. Dr. Raúl Rojas

vorgelegt von
Slav Petrov
slav@petrovi.de

am Fachbereich Mathematik und Informatik der
Freien Universität Berlin
Takustr. 9, 14195 Berlin

Berlin, den 17. Mai 2004

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig angefertigt habe, sämtliche Quellen, die verwendet wurden, angegeben habe und dass die Arbeit nicht schon an anderer Stelle zur Prüfung vorgelegt wurde.

Berlin, den 17. Mai 2004

Zusammenfassung

Diese Diplomarbeit beschreibt Teile der Steuerungssoftware der Fußball-Roboter der Freien Universität Berlin, **FU-Fighters**, welche in der Middle Sized League von RoboCup spielen. Fußball spielende Roboter haben vielfältige Herausforderungen zu bewältigen und es sollen einige Lösungen vorgestellt werden, die jedoch auch für mobile Roboter in anderen Umgebungen relevant sein können.

Viele der entwickelten Algorithmen bauen auf die von einer omnidirektionalen Kamera gelieferten Bilder auf. Die Qualität dieser Bilder wird durch einen automatischen Weißabgleich verbessert, der durch das Benutzen einer Weißreferenz eine weitgehende Farbkonstanz erzielt. Die Erkennung des Balles ist eine der wichtigsten Aufgaben im Roboter-Fußball. Anhand von Farbinformationen und Kenntnissen über die Ballform kann die Ballposition sehr genau und zuverlässig bestimmt werden. Desweiteren ist auch die Abschätzung der Ballgeschwindigkeit von großer Bedeutung. Der hier vorgestellte Algorithmus benutzt dazu ein Kalman-Filter und kann im Falle, dass der Ball verdeckt ist, aber frei rollt, zuverlässig zur Ballvorhersage verwandt werden.

Außerdem werden zwei Arten von Sensorfusion behandelt: einerseits die Fusion von Beobachtungen desselben Objektes aus verschiedenen Blickpunkten, d.h. durch verschiedene Roboter. Die Beobachtungen der einzelnen Roboter werden als Gauß-Verteilungen modelliert und werden von einem zentralen Rechner zu einem globalen Weltbild fusioniert. Andererseits die Fusion von Messwerten bezüglich desselben Systemzustandes durch verschiedene Sensoren auf einem Roboter, z.B. zur Selbstlokalisierung mit Hilfe von Odometer und Vision. Zur Fusion der von verschiedenen Sensoren stammenden Daten wird ein diskretes Kalman-Filter verwandt und eine neuartige Methode zur Behandlung von Daten mit unterschiedlichen Wahrnehmungsverzögerungen entwickelt.

Das Zusammenwirken dieser einzelnen Elemente wird am Beispiel des Torwart-Verhaltens verdeutlicht. Dank der verbesserten Bestimmung der Ballposition und -geschwindigkeit kann der Auftreffpunkt des Balles im Tor frühzeitig erkannt werden. Weiterhin führt die Sensorfusion zu einer genaueren Positionsbestimmung des Roboters und des Balles. Dadurch ist der Torwart in der Lage die meisten Schüsse auf das Tor abzuwehren.

"Künstliche Intelligenz ist das ehrgeizigste und gleichzeitig romantischste Großprojekt der Computerwissenschaften, und die gottgleiche Suche nach einer Form intelligenten Lebens bildet eine größere philosophische Herausforderung als die meisten anderen Aspekte der Cyber-Welt."

Stephan Vladimir Bugaj in "Was liest die Zukunft?"
Frankfurter Allgemeine Zeitung am 17. April 2001

Danksagungen

Danken möchte ich in erster Linie meinen Eltern für ihre alltägliche Unterstützung. Ohne sie wäre es mir nicht möglich gewesen mein Studium in so kurzer Zeit durch zu ziehen. Ein großer Dank gehört auch an meinen Bruder. Seinem fußballerischer Sachverstand war mir bei der Programmierung des Torwarts von großer Hilfe. Dankbar bin ich auch für seine Korrekturen an den ersten Versionen dieser Diplomarbeit.

Desweiteren möchte ich Raúl Rojas für das interessante Thema und die große Freiheit bei der Erstellung dieser Diplomarbeit danken. Felix von Hundelshausen und Fabian Wiesel möchte ich für ihre Unterstützung und die gute Zusammenarbeit danken.

Nicht zu letzt danke ich auch all den Freunden, die stets dafür gesorgt haben, dass ich genügend Ablenkung erhalte und auf keinen Fall zu viel Zeit in der Uni verbringe.

Außerdem danke ich Carlo Tomasi für die interessante und lehrreiche Zeit an der Duke University und für seine Unterstützung bei meiner Bewerbung für die Promotionsstelle in den USA.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Die folgende Diplomarbeit beschreibt Algorithmen und Methoden für mobile Roboter. Dabei werden Szenarien aus dem Umfeld des Roboter-Fußballs beschrieben, da die hier beschriebenen Methoden von den Fußball spielenden Robotern der Freien Universität Berlin (FU-Fighters) [38] benutzt werden. Bei der Steuerung mobiler Roboter in den verschiedensten Umgebungen treten ähnliche Probleme auf, wie sie auch beim Roboter-Fußball gelöst werden müssen. Daher sollte diese Arbeit nicht als reine Beschreibung eines Fußball spielenden Roboters verstanden werden.

1.1 Wieso Roboter-Fußball?

Nachdem sich schon in den Achtziger Jahren andeutete, dass das Computer-Schach-Problem bald gelöst sein würde, fingen Forscher aus dem Bereich der Künstlichen Intelligenz an, sich nach neuen Herausforderungen umzuschauen. Das Problem sollte komplex sein und in der realen Welt stattfinden. A. Mackworth listet in [23] die Grundannahmen der "klassischen" Künstlichen Intelligenz (KI) auf und beschreibt ein Szenario, in dem alle diese ursprünglich angenommenen Vereinfachungen verletzt sind. Als Beispiel benutzt er einen Roboter, der Fußball spielen soll. Im Gegensatz zu Computer-Schach nämlich befindet sich dieser in einer ständig wechselnden Umgebung, sein Wissen ist unvollständig und ungenau und er muss mit anderen Manschaftsmitgliedern kooperieren.

Das Problem stellt eine Herausforderung dar, da eine große Menge an Daten in Echtzeit verarbeitet werden muss und viele unvorhersehbare Situationen entstehen können und doch ist es mit angemessenem finanziellen und technischen Aufwand lösbar. Inzwischen hat sich Roboter-Fussball zu einem der Aushängeschilder der KI-Forschung entwickelt. Jährlich werden von der RoboCup Fe-

deration [14] Turniere ausgetragen, bei denen Teams aus allen Kontinenten gegeneinander antreten.

1.2 RoboCup

Gegründet wurde die RoboCup Federation [14] als internationales Gremium aus rund 150 Universitäten und Forschungszentren mit dem Ziel, Forschung und Entwicklung in den Bereichen Robotik und Künstliche Intelligenz durch einen auch für Laien zu verstehenden Wettbewerb zu fördern: es wurde ein Standardproblem gesucht, an dem eine große Zahl unterschiedlicher Technologien integriert und verglichen werden kann. Die Wahl eines solchen Anwendungsszenarios fiel auf das Fußballspiel. Das langfristige Ziel des RoboCup ist es, mit humanoiden Robotern den menschlichen Fußballweltmeister im Jahre 2050 zu schlagen. Noch ist man von diesem Ziel weit entfernt; Prognosen, die weiter als zehn Jahre reichen, sind wegen der rasanten technologischen Entwicklung sehr gewagt, so dass nicht abgesehen werden kann, ob das Ziel nur eine Utopie bleiben wird.

Seit der Gründung im Jahre 1997 findet jedes Jahr eine Weltmeisterschaft statt, wobei in vier verschiedenen Ligen gespielt wird:

- Small-Sized Robot League (F180) mit Mannschaften von ursprünglich je fünf Robotern (Durchmesser bis zu 18 cm), die auf einem Feld von der Größe eines Tischtennis-Tisches (180 Quadratzentimeter, daher der Name F180) gegeneinander antreten;
- Middle-Sized Robot League (F2000), in der anfangs Teams von je vier Robotern (Durchmesser bis zu 50 cm) auf einem Feld von 9 mal 6 Meter kämpften;
- Sony Legged Robot League mit Mannschaften aus drei vierbeinigen Robotern (Sony Aibo Robots [29]) und
- Simulation League, in der autonome Computerprogramme elf simulierte Roboter (Software-Agenten) virtuell über ein Spielfeld schicken.

Abbildung 1.1: Das offizielle RoboCup-Logo.

Jedes Jahr werden dabei das Spielfeld vergrößert und Erleichterungen (wie z.B. Banden am Spielfeldrand) abgeschafft, um dem ultimativen Ziel schrittweise näher zu kommen.

Damit ein Roboterteam tatsächlich Fußball spielen kann, müssen verschiedene Technologien integriert werden. Unter anderem Entwurfsprinzipien für autonome Agenten, Multi-Agenten-Kooperation, Strategielernen, Echtzeit-Planung, Robotik und Sensorfusion. RoboCup ist eine Aufgabe für ein Team von sich schnell bewegenden Robotern in einer dynamischen Umgebung. Lösungen für einige dieser Probleme sollen in dieser Diplomarbeit vorgestellt werden.

Für eine ausführlichere Einleitung zum Thema RoboCup wird auf [30] verwiesen.

Abbildung 1.2: Die aktuellen Roboter der FU-Fighters.

1.3 Middle-Sized League

Die Middle-Sized (kurz: Mid-Size) League wird als sog. “Königsklasse“ bezeichnet. Das liegt daran, dass jeder der eingesetzten Roboter komplett autonom handeln kann. Während es in der Small-Sized League eine (oder zwei) Kameras gibt, die das ganze Spielfeld überblicken und alle Roboter ohne eigene Intelligenz von einem zentralen Rechner gesteuert werden, existiert zwar auch in der Mid-Sized League meist ein zentraler Steuerungsrechner, jedoch ist die Intelligenz auf die einzelnen Roboter verteilt. Jeder Roboter hat eine eigene Kamera und lokalisiert sich und den Ball selbständig. Dabei reicht die Sichtweite nicht aus, um das gesamte Spielfeld zu überblicken; der Roboter muss

nur anhand der von seiner Position aus sichtbaren Linien eine Vermutung über seine Position aufstellen.

Nach jahrelangem Engagement in der Small-Size Liga und mehreren Vize-Weltmeisterschaften wurde 2002 ein Middle-Size Team an der Freien Universität gegründet. Der Name der erfolgreichen kleineren Mannschaft wurde dabei übernommen: FU-Fighters. Beim Bau der Roboter wurde ein minimalistisches Designprinzip verfolgt; die Roboter sollten leicht und dadurch schnell und wendig sein. Je weniger Komponenten benutzt werden, desto geringer sind die Kosten und desto unwahrscheinlicher ist ein Systemausfall. In Abbildung 1.2 ist der dabei entstandene Roboter zu sehen. Das Gesamtgewicht des Roboters beträgt etwa 10 kg, ein Bruchteil dessen, was andere Teams (Philips CFT [9], Minho [26], Allemaniacs [2]) auf die Waage bringen.

Um wie eine Mannschaft aufzutreten, bedarf es der zusätzlichen Kommunikation zwischen den einzelnen Spielern. Dies kann in einer paarweisen, direkten Kommunikation von Spieler zu Spieler erfolgen oder über einen Zentralrechner. Wie später erläutert werden wird, teilt jeder Roboter dem Zentralrechner seine Position sowie eine Hypothese über die Position des Balles mit. Diese Information ist mit einer gewissen Unsicherheit behaftet und es ist Aufgabe des Zentralrechners die teils widersprüchlichen Angaben der einzelnen Spieler zu fusionieren. anschließend kann der Zentralrechner den Spielern Anweisungen geben, z.B. wer zum Ball gehen soll und wer sich besser an einer anderen Stelle positionieren soll.

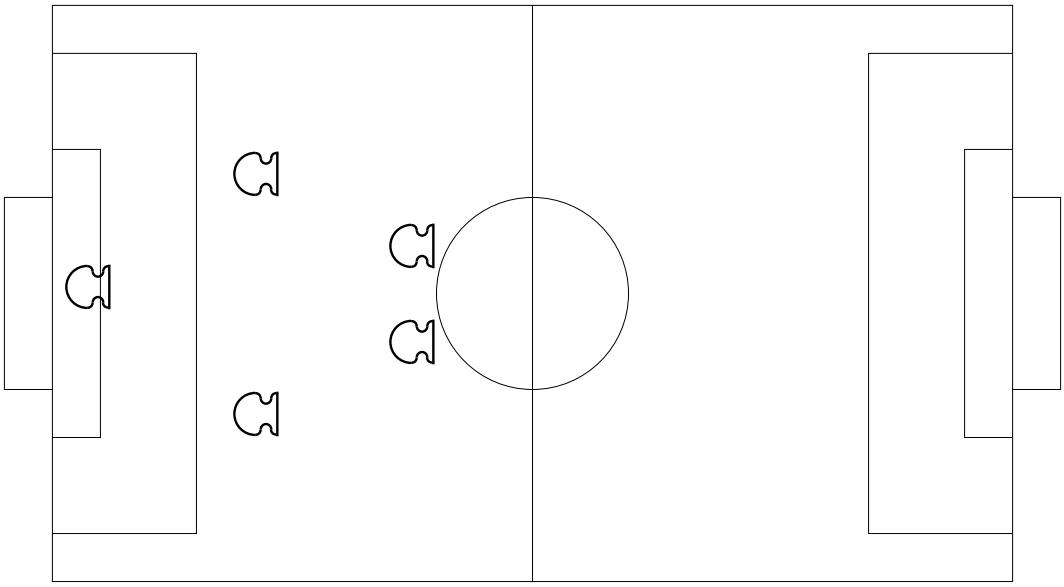


Abbildung 1.3: Das Spielfeld bei den German Open 2004 [1]. Eingezeichnet ist die Aufstellung beim gegnerischen Anstoß, wenn mit fünf Robotern gespielt wird.

Im folgenden beziehen sich die Größenangaben auf das Spielfeld bei den German Open 2004 [1] in Paderborn, bei denen die FU-Fighters den vierten Platz belegten. Das Spielfeld ist zehn Meter lang und sechs Meter breit. Die Tor- und Strafräume sowie der Mittelkreis sind proportional in Abbildung 1.3 eingezeichnet. An den Ecken sind farbige Eckpfeiler aufgestellt, die zur Lokalisierung benutzt werden können. Um leichter entscheiden zu können auf welches Tor gespielt wird, sind die Tore farbig markiert (das eine ist blau, das andere gelb).

1.4 Aufbau der Arbeit

Die Diplomarbeit ist wie folgt aufgebaut: Kapitel 2 beschreibt die benutzte omnidirektionale Kamera, sowie Verfahren zum automatischen Weißabgleich und zur Bestimmung der Abstands- und Positionsfunktion. Im darauf folgenden Kapitel wird beschrieben, wie die Ballposition exakt bestimmt werden kann und wie durch Anwendung eines Kalman-Filters der Ball verfolgt und seine Position vorhergesagt werden kann. Das Kapitel wird mit einem Verfahren zur Fusion von verschiedenen Beobachtungen abgeschlossen. Kapitel 4 behandelt die Sensorfusion mit Kalman-Filter und die automatische Synchronisation mehrerer Sensoren mit verschiedenen Wahrnehmungsverzögerungen. Wie diese Verfahren zusammenwirken, wird anhand der Torwart-Verhalten in Kapitel 5 gezeigt. Im letzten Kapitel wird werden die erzielten Ergebnisse zusammengefasst und ein Ausblick auf mögliche Erweiterungen gegeben.

Anhang A kann als Einstieg in die Theorie des an verschiedenen Stellen verwandten Kalman-Filters benutzt werden und legt die im Text benutzten Bezeichnungen fest. Anhang B erläutert das Prinzip des PID-Reglers.

Kapitel 2

Lokalisierung

Die Wahrnehmung der Umwelt ist eine notwendige Voraussetzung für ein sinnvolles Verhalten. In diesem Kapitel wird zuerst das sog. “catadioptrische Prinzip“ vorgestellt und die dabei eingesetzte Hardware erläutert. Für diese Kamerakonstruktion wird dann ein Weißabgleich entwickelt, der eine weitgehende Farbkonstanz erzielt. anschließend wird auf die Transformation von Bild- auf Weltkoordinaten eingegangen bevor am Ende des Kapitels kurz erklärt wird wie anhand der Feldlinien die Position auf dem Spielfeld ermittelt werden kann.

2.1 Das catadioptrische Prinzip

Beim Bau der Roboter wurde auf den Einsatz leichter Komponenten geachtet, um das Gesamtgewicht des Roboters minimal zu halten und dadurch eine gute Maneuvrierfähigkeit zu ermöglichen. Desweiteren sollten, soweit möglich, preiswerte Bauteile benutzt werden.

Während in den ersten Jahren des RoboCup-Wettbewerbes noch verschiedenste Kameramontierungen benutzt wurden, scheint sich nun der catadioptrische Ansatz durchgesetzt zu haben. Dabei wird die Kamera senkrecht auf dem Roboter befestigt und auf einen über ihr montierten Spiegel gerichtet. Diese Konstruktion ermöglicht es das Sichtfeld auf 360 Grad zu erweitern, d.h. Objekte die sich in einer beliebigen Richtung vom Roboter befinden, ohne Bewegen der Kamera zu erfassen. Wenn im Weiteren von omnidirektionalen Kameras gesprochen wird, dann ist das catadioptrische Prinzip gemeint. Als gute Einführung in dieses Thema eignet sich [31].

2.1.1 Die Kamera

Als Kamera wird eine handelsübliche Firewire-Webcam benutzt. Diese liefert eine Auflösung von 320 x 240 Pixel und kann wahlweise mit 30 oder 15 Bil-

dern pro Sekunde (fps) betrieben werden. Dabei hat es sich von großem Vorteil erwiesen, eine Kamera mit Firewire-Anschluss und nicht mit einem USB 2.0 - Anschluss zu verwenden. Während die Prozessorauslastung beim bloßen Anzeigen des aktuellen Bildes bei USB 2.0 Kameras bei etwa 30 Prozent liegt, verbrauchen Kameras mit Firewire Verbindung lediglich ca. 5 Prozent der verfügbaren Prozessorressourcen.

2.1.2 Der Spiegel

Omnidirektionale Kameras unterscheiden sich hauptsächlich durch die Form des eingesetzten Spiegels. üblich sind parabolische, hyperbolische, sphärische, elliptische und kegelförmige Spiegel [31].

Der Spiegel sollte so gestaltet sein, dass der Roboter alle relevanten Teile des Bildes mit möglichst geringer Verzerrung betrachten kann. Folgende Anfordungskriterien können demnach definiert werden:

- Wenn der betrachtete Punkt sehr nah am Roboter ist oder diesen sogar berührt, muss die Genauigkeit bezüglich Richtungswinkel und Entfernung sehr genau sein. Eine solche Situation tritt zum Beispiel auf, wenn der Roboter den Ball schießen möchte.
- Wenn der betrachtete Punkt wenige Meter vom Roboter entfernt ist, dann sollte der Fehler, sowohl bezüglich der Richtung, als auch bezüglich der Entfernung relativ gering sein und außerdem unabhängig von der Position des Punktes sein. Ein Beispiel ist die Selbst-Lokalisierung anhand von Linien, deren Position bekannt ist.
- Sobald der betrachtete Punkt weit weg ist, spielt die genaue Entfernung eine untergeordnete Rolle. Vielmehr ist die Richtung entscheidend, damit sich zum Beispiel der Roboter auf einen weit entfernten Ball zu bewegen kann.
- Außerdem sollte der Spiegel so gestaltet sein, dass die Markierungen, die jeder Roboter tragen muss, erkannt werden können. Diese Anforderung kann auch vernachlässigt werden.

Da keine der klassischen Formen all diese Eigenschaften hat, wurde ein Spiegel entwickelt, der aus einem kegelförmigen und einem sphärischen Teil besteht. Das hat den Vorteil, dass der Roboter selbst im Bild kaum zu sehen ist. Dadurch ist es möglich gleichzeitig sowohl die Position von Gegenständen in der näheren Umgebung des Roboters sehr genau zu bestimmen, als auch weit zu schauen. Bei der Konstruktion des Spiegels wurde zuerst die Art der

Abbildung 2.1: Kamera und Spiegel.

Abstandsfunktion festgelegt und anhand dieser ein Spiegel gefertigt. Die Abstandsfunktion hat zur Aufgabe Abstände im Bild in Abstände in der Welt abzubilden und umgekehrt. Pedro Lima beschreibt in [22] die Konstruktion eines solchen zweiteiligen Spiegels im Detail. Im nächsten Abschnitt wird auf die Abstandsfunktion näher eingegangen.

In Abbildung 2.1 ist die eingesetzte Konstruktion zu sehen. Der Spiegel ist auf drei Plexiglas-Stäben montiert, die im Bild kaum zu sehen sind und nahezu keine Verzerrung verursachen. Ein weiteres Merkmal ist, dass die Kamera so montiert ist, dass Objekte, die sich vor dem Roboter befinden, im Bild auf der Diagonalen erscheinen, da dadurch eine maximale Sichtweite erreicht werden kann.

2.2 Automatischer Weißabgleich

Eine weitgehende Farbkonstanz stellt eine wesentliche Voraussetzung für die meisten auf dem Gebiet der Computer Vision verwandten Algorithmen dar. Als Beispiel seien die Selbslokalisierung anhand der weißen Linien auf dem grünen Spielfeld oder die Balldetektion anhand seiner roten Farbe gennant. Diese Verfahren sind sehr anfällig gegen Änderungen der Beleuchtungsbedingungen. Außerdem führen Abstürze in der Elektronik der benutzten Kameras dazu, dass sich die Kameraparameter verstellen und diese mühsam manuell eingestellt werden müssen. In diesem Abschnitt soll daher ein Verfahren vorgestellt werden, das in der Lage ist, die Kamera nach einem Absturz wieder

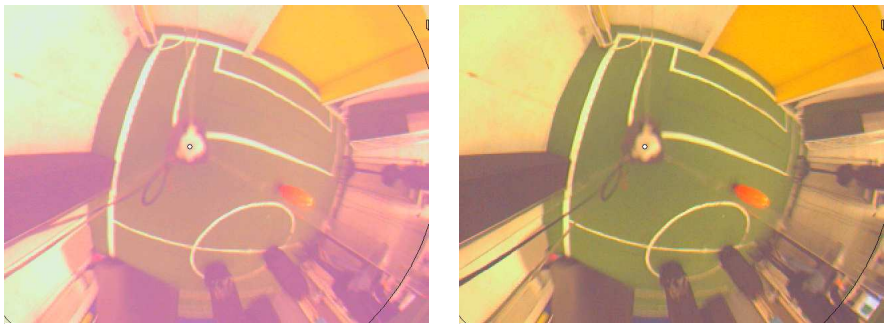


Abbildung 2.2: Mögliche Kamerabilder: (a) nach einem Systemabsturz, (b) nach Änderung der Lichverhältnisse.

neu zu konfigurieren und danach unter variierenden Beleuchtungsbedingungen eine weitgehende Farbkonstanz zu erreichen. Die Idee dazu stammt aus [40].

Abbildung 2.2 verdeutlicht wie unterschiedlich die Kamerabilder sein können. Obwohl die Kamera einen automatischen Weißabgleich vornehmen kann, ist es nicht sinnvoll diesen Kameramodus hier zu benutzen. Der automatische Weißabgleich der Kamera beruht nämlich auf G. Buchsbaum's "gray world assumption" [8], welche besagt, dass die durchschnittliche Oberflächenreflektion über das gesamte Bild grau ist. Obwohl diese Annahme oft korrekt ist, führt sie beim hier betrachteten Szenario zu sehr schlechten Ergebnissen: abhängig davon ob sich der Roboter in der Nähe des gelben oder des blauen Tores befindet, ändert sich, auch bei sonst identischen Beleuchtungsbedingungen, die Farbe des grünen Spielfeldes, wenn der automatische Weißabgleich der Kamera benutzt wird.

Um diesem Problem zu begegnen, musste daher ein anderer Weißabgleich entwickelt werden, der speziell für Bilder von einer omnidirektionalen Kamera optimiert sein sollte. Anstelle der "gray world assumption" sollte die Kamera mit einer Weißreferenz arbeiten. Die erste Idee bestand darin, die weißen Linien des Spielfeldes als Weißreferenz zu benutzen. Wenn die Kameraparameter verstellt sind, ist es jedoch nicht ohne weiteres möglich die Spielfeldlinien zu detektieren. Damit der Algorithmus außerdem zuverlässig in jeder beliebigen Umgebung angewandt werden kann, wurde die Kamera mit einem weißen Ring ausgestattet, der als Weißreferenz im Bild erscheint. Dieser weiße Ring ist so angebracht, dass er nichts vom Spielfeld verdeckt.

Das von der Kamera gelieferte Bild ist im YUV-Format und es ist für den Weißabgleich nicht nötig das Bild in ein anderes Format zu konvertieren (Das YUV-Format wurde ursprünglich entwickelt, um Farbfernsehen abwärtskompatibel zum Schwarzweißfernsehen zu machen). Es wird der Mittelwert für Y, U, und V über alle Pixel berechnet, die sich innerhalb der Weißrefe-

renz befinden. Mit der Differenz zu den theoretischen Sollwerten $U = 0$ und $V = 0$ werden mit Hilfe von zwei separaten PID-Reglern die Stellgrößen für die Verstärkungsfunktion für U und V für die verwendete Kamera bestimmt (Da die benutzte Kamera nicht das volle Farbspektrum abdeckt, mussten die Sollwerte manuell bestimmt werden). Außerdem wird der mittlere Wert von Y verwendet, um die Helligkeit der Kamera möglichst konstant zu halten. In Abbildung 2.3 ist das aus 2.2 nach dem automatischen Weißabgleich hervorgegangene Bild zu sehen.

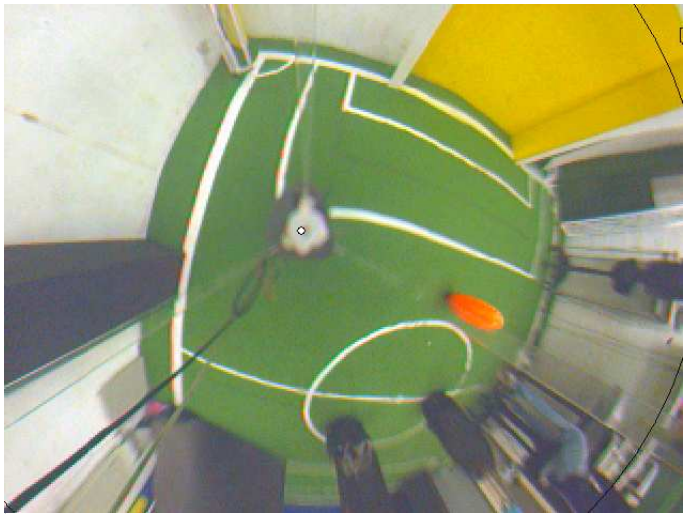


Abbildung 2.3: Das Kamerabild nach dem Weißabgleich.

2.3 Die Abstands- und Positionsfunktion

Durch die Form des Spiegels wird die Art der Abstandsfunktion vorgegeben: im sphärischen Teil des Spiegels ist die Abstandsfunktion linear, im kegelförmigen Teil ist sie dagegen exponentiell. Die Abstandsfunktion kann aus den Daten des Spiegels und der Kamera theoretisch hergeleitet werden. Jedoch ist es kaum möglich auf allen Robotern das Spiegelzentrum in exakt der gleichen Höhe und ganz genau über dem Zentrum der Kameralinse zu montieren. Zusätzlich ist der Spiegel oft leicht zur Seite geneigt. Diese kleinen Ungenauigkeiten und Unterschiede von wenigen Millimetern haben jedoch einen starken Einfluß und können zu Fehlern im Meterbereich führen. Daher weicht die theoretisch abgeleitete Abstandsfunktion zu stark von der Realität ab.

Nicht nur weicht die theoretisch abgeleitete Abstandsfunktion wegen dieser Ungenauigkeiten und Unterschiede stark von der Realität ab, sondern ist sie zusätzlich abhängig von der Richtung, in der der Abstand gemessen wird. Ein

Abstand von x Pixeln nach oben (im Bild) entspricht einem anderen realen Abstand (d.h. in der realen Welt), als ein Abstand von x Pixel nach unten (im Bild). Das hat zur Folge, dass eine Abstandsfunktion die nur Abstände im Bild betrachtet und diese in Abstände in der Welt umrechnet, nicht in alle Richtungen korrekt sein kann; die Richtung zum Objekt hin muss bei der Abstandsumrechnung mitberücksichtigt werden. Die Funktion, die sowohl Abstand als auch Richtung berücksichtigt, ist die Positionsfunktion. Die Positionsfunktion ist aus 180 radial vom Kamerazentrum ausgehenden Abstandsfunktionen konstruiert (jeweils eine á zwei Grad). Dabei sind benachbarte Abstandsfunktionen aneinander gekoppelt, und können an jeder Stelle nur um einen bestimmten Wert voneinader abweichen.

Die Abstandsfunktion wird einerseits dafür benutzt um die Entfernung zu Objekten auf dem Feld (der Ball oder andere Roboter) abzuschätzen. Andererseits wird sie von den zur Positionsbestimmung des Roboters benutzten Algorithmen verwandt. Die Idee dabei ist es die im Bild sichtbaren Feldlinien zu detektieren und diese dann mit einem vordefinierten Feldmodell zu vergleichen. An dieser Stelle werden die Abstands- und Positionsfunktion von großer Bedeutung. Wenn diese nicht korrekt kalibriert sind, ist es nicht möglich eine gute und korrekte übereinstimmung zwischen den gesehenen Linien und dem Modell zu finden, so dass die Positionsbestimmung fehlerhaft wird.

2.3.1 Empirische Bestimmung

Eine Abstandsfunktion wurde empirisch bestimmt und wird zur Initialisierung der Abstandsfunktionen in die verschiedenen Richtung benutzt. Für jeden einzelnen Roboter und jede Richtung wird die Positionsfunktion manuell optimiert. Dazu werden die gesehenen Linien entzerrt auf das Spielfeldmodell projeziert und der Benutzer hat die Möglichkeit die jeweilige Abstandsfunktion mit Hilfe der Maus anzupassen. In diesem Abschnitt wird eine Möglichkeit zur empirischen Bestimmung der Abstandsfunktion beschrieben:

Der Roboter wurde zur Spielfeldmitte schauend im Straufraum positioniert. Vor dem Roboter wurde eine Art “Zebrastreifen“ konstruiert: In regelmäßigen Abständen wechseln sich weiße und schwarze Papierstreifen ab. Da die Auflösung mit zunehmendem Abstand abnimmt, muss die Breite der Streifen auf dem Spielfeld mit dem Abstand zunehmen, damit die Streifen im Bild sichtbar bleiben. In Abbildung 2.4 ist das Spielfeld aus der Sicht des Roboters zu sehen. Die Auflösung beträgt nur 320 x 240 Pixel und die weit entfernten Streifen sind nur schwer sichtbar. Man beachte außerdem, dass die Streifen auf dem Spielfeld direkt vor dem Roboter aufgemalt wurden, im Bild jedoch auf der Diagonalen erscheinen. Die Kamera ist gezielt so montiert worden, um dadurch die Sichtweite zu maximieren.

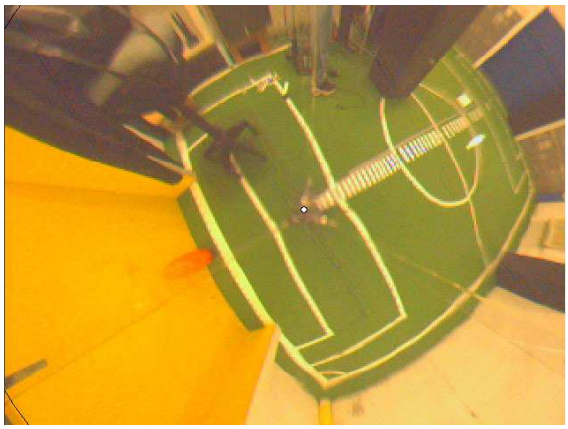


Abbildung 2.4: Roboter mit Streifenmuster auf dem Feld.

Manuell wurde in Abbildung 2.4 ein Pixel aus jedem der Streifen bestimmt. Diese wurden jeweils zur Berechnung der Entfernung zum manuell festgelegten Spiegelzentrum im Bild benutzt. Diese Entfernung und der Abstand des Streifens zum Roboter in der Welt sind in Abbildung 2.5 dargestellt. Die Abstandsfunktion a: $Pixelabstand \rightarrow Weltabstand$, welche Abstände im Bild in Abstände in der Welt umwandelt, ist im selben Graphen gezeichnet. Wie bereits erwähnt, besteht sie aus einem linearen Teil l mit dem Definitionsbereich $D(l) = [0; d]$ und einem exponentiellen Teil e mit dem Definitionsbereich $D(e) =]d; \infty[$. Bei der Herleitung der Zuordnungsvorschrift muss darauf geachtet werden, dass die Abstandsfunktion a stetig und differenzierbar ist. außerdem sollte sie anhand von Parametern verändert werden können, ohne dass diese beiden grundlegenden Eigenschaften verloren gehen. Als Parameter werden benötigt:

- m: die Steigung des linearen Abschnitts,
- n: der Achsenabschnitt des linearen Abschnitts,
- d: das Ende des linearen und der Anfang des exponentiellen Abschnitts,
- k: der Streckfaktor des exponentiellen Abschnitts.

Es sollte angemerkt werden, dass unter idealen Umständen kein Achsenabschnitt für den linearen Teil benötigt wird, da die Abbildungsfunktion durch den Ursprung verläuft. Da es aber für den Bereich unter 30 cm (in der Welt) keine Messergebnisse gibt (siehe Abbildung 2.5), wurde der Achsenabschnitt ergänzt. Das liegt daran, dass Abstände von weniger als 30 cm kaum messbar sind, da die Objekte dort sehr stark verzerrt werden und alles verwischt erscheint. Das wird zu einem Problem, wenn der Roboter sehr nahe am Ball ist

und die genaue Entfernung zum Ball bestimmen möchte. In diesem Bereich ist es wichtig die Ballentfernung sehr exakt zu bestimmen, da es ein großer Unterschied ist, ob der Ball berührt wird, 2 cm oder 5 cm entfernt ist, wenn der Ball geschossen oder gedribbelt werden soll. Wie trotz dieser niedrigen Bildqualität dieses Problem gelöst werden kann, wird im Abschnitt 3.1.3 über die genaue Positionsbestimmung des Balles erläutert werden.

In Abbildung 2.5 sind die Messwerte, sowie die daraus abgeleitete Abstandsfunktion a dargestellt, wobei a die folgende Zuordnungsvorschrift hat:

$$a(x) = \begin{cases} l(x) = mx + n, & x \leq d \\ e(x) = mx + n + \frac{m}{k}(e^{k(x-d)} - 1), & x > d \end{cases}$$

mit den folgenden Parametern: $m = 1.83, n = 10, k = 0.024, d = 45$.

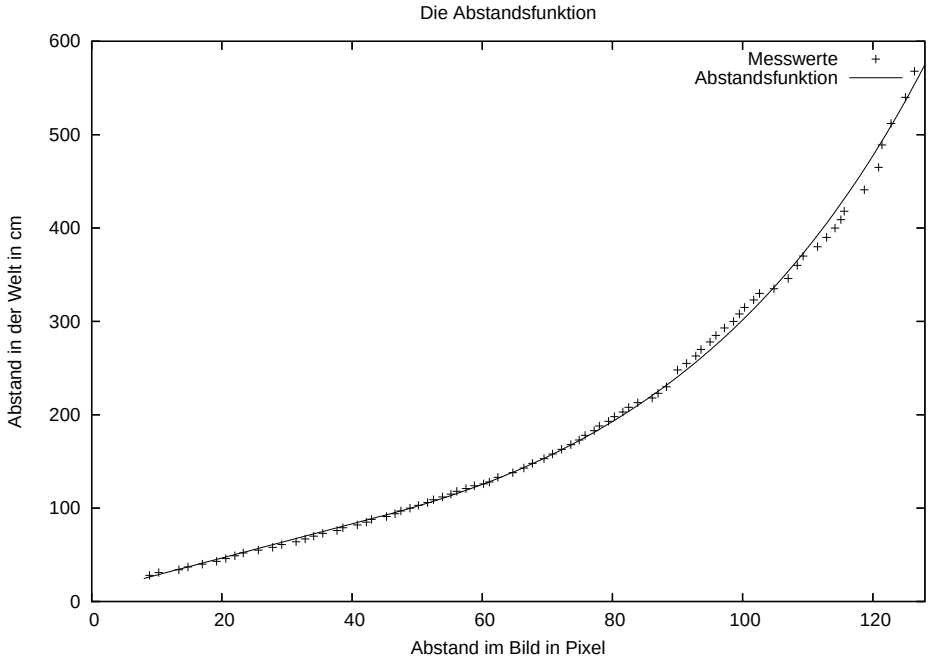


Abbildung 2.5: Die Abstandsfunktion.

2.3.2 Mathematische Eigenschaften der Abstandsfunktion

Behauptung: Unabhängig davon wie die Parameter gewählt werden, bleibt die Funktion erstens stetig und zweitens differenzierbar.

Beweis:

1. Die Funktion ist stetig, denn der linksseitige und rechtsseitige Grenzwert an der Stelle d stimmen überein.

a) $\lim_{x \to d^-} a(x) = \lim_{x \to d^-} l(x) = md + n$ und

b) $\lim_{x \to d^+} a(x) = \lim_{x \to d^+} e(x) = md + n + \frac{m}{k}(e^{k(d-d)} - 1) = md + n + \frac{m}{k}(e^0 - 1) = md + n + \frac{m}{k}(1 - 1) = md + n$.

□

2. Die Funktion ist differenzierbar.

Es ist bekannt, dass lineare und exponentielle Funktionen differenzierbar sind. Es muss nur gezeigt werden, dass a am übergang bei $x = d$ vom linearen in den exponentiellen Teil differenzierbar ist, d.h. dass die Ableitungsfunktion a' an der Stelle d stetig ist. Es gilt

a) Wegen $l'(x) = m$ gilt $\lim_{x \to d^-} a'(x) = \lim_{x \to d^-} l'(x) = m$ und

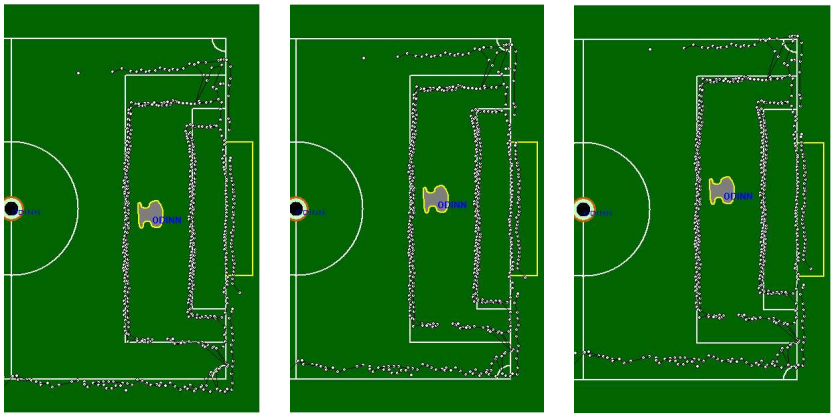
b) Es ist $e'(x) = m + m(e^{k(x-d)} - 1)$ und damit $\lim_{x \to d^+} a'(x) = \lim_{x \to d^+} e'(x) = m + m(e^{k(d-d)} - 1) = m + \frac{m}{d}(e^0 - 1) = m$.

□

Durch die Einschränkung der Abstandsfunktion auf die obige Form kann der Benutzer zur Laufzeit die Parameter beliebig anpassen, ohne dass die Funktion ihre Eigenschaften verliert.

2.3.3 Automatische Kalibrierung

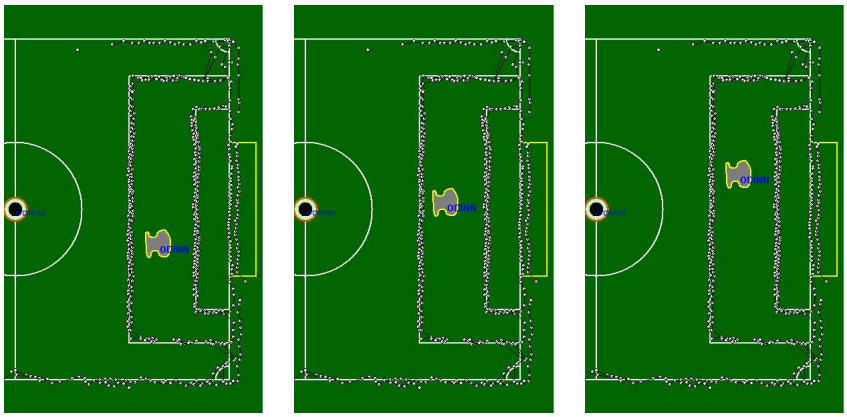
Eine automatische Kalibrierung der Abstandsfunktion ist zwar wünschenswert, jedoch nur schwer umsetzbar. Die hier empirisch bestimmte Abstandsfunktion reicht meist für eine grobe Lokalisierung aus. Die Unterschiede zwischen den gesehenen Linien und den Linien in Feldmodell könnten nun benutzt werden um die Abstandsfunktion zu optimieren. Das dabei auftretende Problem ist, dass die genaue Position des Roboters nicht bekannt ist und es daher mehrere Möglichkeiten gibt die Abstandsfunktion so anzupassen, dass die gesehenen Linien perfekt mit dem Feldmodell übereinstimmen. Angenommen der Roboter befindet sich zur Feldmitte schauend im Strafraum und der gesehene Strafraum ist in y-Richtung kleiner als der Strafraum im Modell. Dieser Fehler kann auf drei verschiedene Arten korrigiert werden: 1. Die Abstandsfunktion wird nur in die eine Richtung (zum Beispiel links) solange verändert bis die Linien mit



(a) Links (b) Mitte (c) Rechts

Abbildung 2.6: Die Positionsfunktion vor der Optimierung.

dem Modell übereinstimmen. 2. Alternativ kann die Abstandsfunktion mit dem selben Ergebnis nur in die andere Richtung (rechts) verändert werden oder 3. die Abstandsfunktion wird in beide Richtungen angepasst. Alle drei Varianten haben dasselbe Resultat: die gesehenen Linien stimmen mit dem Feldmodell genau überein. Die Position des Roboters wird jedoch unterschiedlich sein. Abhängig davon wo der Roboter sich auf dem Spielfeld befindet, wie der Spiegel geneigt ist und welche Linien gesehen werden, kann jede der drei erwähnten Varianten die richtige sein. Wie man in 2.6 und 2.7 wird der Fehler bei der Positionsbestimmung unter Umständen sogar vergrößert.



(a) Links (b) Mitte (c) Rechts

Abbildung 2.7: Drei Möglichkeiten die Positionsfunktion zu optimieren.

2.4 Positionsbestimmung anhand der Feldlinien

Abbildung 2.8: Das Feld aus der Sicht des Roboters

Eine präzise Positionsbestimmung ist für die Roboter von großer Bedeutung. Es gibt zwei Sensoren, die dafür relevante Informationen liefern können. Einerseits kann das in Kapitel 2 Abschnitt 2.1.1 beschriebene omnidirektionale Vision System benutzt werden. Dank farbiger Tore und Eckpfeiler, sowie der Feldlinien existieren verschiedene Orientierungshilfen, die zur Bestimmung der Position des Roboters auf dem Spielfeld benutzt werden können. Da die Eckpfeiler verdeckt sein können und die Tore für eine exakte Lokalisierung nicht ausreichen, haben sich die Feldlinien als bester Anhaltspunkt für die Positionsbestimmung erwiesen. Aus den Verschiebungen in aufeinander folgenden Bildern kann neben der aktuellen Position auch die relative Bewegung des Roboters bestimmt werden. Andererseits ist an jedem der drei Räder des omnidirektionalen Antriebes ein Encoder montiert, der die Raddrehung mißt. In Kapitel 4 wird es darum gehen wie die Fusion dieser Information umgesetzt werden kann.

In diesem Abschnitt soll beschrieben werden wie die Feldlinien im Bild erkannt werden können. Dafür stehen zwei alternative Methoden zur Verfügung. Diese sind jedoch nicht Teil dieser Diplomarbeit und werden daher nur kurz angerissen.

Die erste Methode besteht darin, anhand von einer manuell initialisierten Look-Up-Tabelle die grünen Regionen des Spielfeldes zu bestimmen und mit Hilfe des Region-Growing-Algorithmus' [37] zu verfolgen. Die detektierten Feldlinien werden mit einem vordefinierten Modell verglichen, so dass man die Position des Roboters erhält. Merkmalserkennung kann benutzt werden um den Mittelkreis oder Ecken zu detektieren und so die Positionsbestimmung

genauer zu machen. Dieses Verfahren hat den Vorteil, dass es sehr zuverlässig ist, jedoch müssen die Farben von Hand kalibriert werden. Durch zahlreiche Optimierungen kann eine Bildrate von 30 Bildern pro Sekunde erreicht werden.

Die zweite Methode wird "Radialscan" genannt. Man betrachtet dabei nur Punkte die auf 180 gleichmäßig radial vom Roboter ausgehende Strahlen liegen. Wenn die Helligkeit über einem Schwellwert liegt, gehört der Punkt zu einer weißen Feldlinie, ansonsten ist er Teil des Feldes oder eines (dunklen) Hindernisses. Schon anhand der ersten gesehenen weißen Linien kann der Roboter seine Position aktualisieren, muss jedoch auf die richtige Position initialisiert werden. Dies kann durch die erste Methode geschehen. Der Vorteil des Radialscans ist, dass nur ein Bruchteil des Bildes betrachtet werden muss und diese Methode daher sehr effizient ist und ohne Probleme mit einer Bildrate von 30 Bildern betrieben werden kann. Der Nachteil des Verfahrens ist, dass es nicht sehr robust ist.

Während anfangs mehr mit dem "Radialscan" gearbeitet wurde, hat sich schließlich das Verfolgen der grünen Regionen dank seiner größeren Robustheit durchgesetzt.

Kapitel 3

Wo ist der Ball?

Das Finden des Balles und die schnelle Erkennung seiner Bewegungsrichtung und -geschwindigkeit sind von großer Bedeutung. Nur so können die Roboter schnell reagieren und erfolgreich sein.

In diesem Kapitel wird zuerst auf die Lokalisierung des Balles im Bild anhand seiner spezifischen Farbe eingegangen. Der nächste Abschnitt führt ein Kalman-Filter zur verbesserten Abschätzung der Ballposition und -geschwindigkeit ein und zeigt auf, wie aus dem Bewegungsmodell des Kalman-Filters leicht eine Ballvorhersage generiert werden kann. Im letzten Abschnitt wird die Fusion der von jedem einzelnen Roboter mit einem gewissen Messfehler bestimmten Ballposition zu einer globalen Ballposition erläutert.

Bei den Ergebnissen in den einzelnen Abschnitten ist leider keine wirkliche Fehlerdiskussion möglich, da die exakte Ballposition stets unbekannt ist. Eine Möglichkeit bestünde darin, eine Kamera über dem Spielfeld zu montieren und die mit ihrer Hilfe bestimmte Ballposition zum Vergleich heranzuziehen. Doch so lange es diese Kamera nicht gibt, können die hier erzielten Ergebnisse nur mit anderen ebenfalls fehlerbehafteten Ergebnissen verglichen werden. Als Gütekriterium wird dabei die Glätte der Balltrajektorie benutzt: wenn der Ball frei rollt, dann bewegt er sich auf einer Geraden und das sollte die bestimmte Balltrajektorie auch wiederspiegeln.

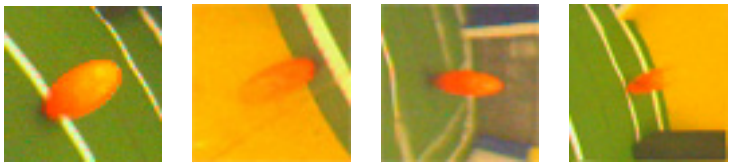


Abbildung 3.1: Verschiedene Ansichten des Balles

3.1 Ball-Tracking

Um den Ball zu finden, kann man sich in erster Linie auf Farbinformationen stützen. Laut RoboCup-Regeln [14] wird nämlich mit einem orangen Ball gespielt (dieser wird von der Fédération Internationale de Football Association (FIFA) [13] im professionellen Bereich bei Schnee eingesetzt). In nicht allzu ferner Zukunft soll mit einem regulären (schwarz-weißem Ball) gespielt werden und es gibt schon Ansätze zur Lösung dieses Problems [36], jedoch sind diese noch nicht ausgereift genug, um während eines Turnieres zuverlässig eingesetzt werden zu können. Wir konzentrieren uns hier auf die etwas leichtere Problemstellung, bei der ein orangener Ball benutzt wird.

Dadurch, dass der Ball sich durch seine Farbe deutlich vom Spielfeld abhebt, ist es meist einfach, ihn zu finden. Das Auffinden wird jedoch durch wechselnde Lichtverhältnisse, Schatten und Verdeckung durch andere Roboter erschwert.

Die Bestimmung der Ballposition erfolgt in zwei Stufen. Zuerst wird aus allen roten Pixeln eines ausgewählt, welches als ungefährer Punkt dient. Anschließend wird das Bild in der Umgebung von diesem Punkt genauer untersucht, um einen exakten Punkt zu finden, der zuverlässig wiedergefunden werden kann. Die Aufteilung in zwei Stufen wurde vor allem aus Gründen der Effizienz vorgenommen.

3.1.1 Ungefähre Positionsbestimmung

Die erste Stufe benutzt eine manuell mit möglichen Ballfarben initialisierte Look-Up Tabelle und klassifiziert anhand dieser jeden Punkt des Bildes, der nicht Teil einer grünen Region ist. Alternativ könnte jeder Pixel anhand seines Farbabstandes zur Referenzfarbe Rot klassifiziert werden, jedoch wird durch das benutzen einer Look-Up Tabelle die Klassifikation stark beschleunigt. Im gleichen Schritt wird auch die Ausdehnung des Spielfeldes bestimmt, so dass Bälle (oder Objekte mit ballähnlicher Farbe), die nicht auf dem Spielfeld sind, nicht weiter in Betracht gezogen werden. Die ballfarbenen Pixel werden zu Clustern zusammengefasst und der größte Cluster wird selektiert. Das Zentrum dieses Clusters wird noch mit der letzten bekannten Ballposition verglichen, um auszuschließen, dass ständig zwischen zwei verschiedenen Regionen hin und her gesprungen wird. Die ungefähre Ballposition ist das Zentrum des selektierten Clusters.

3.1.2 Genaue Positionsbestimmung

Die zweite Stufe hat das Ziel die genaue Ballposition zu bestimmen. Der Einfachheit halber wird die Höhe der Objekte nicht beachtet, sondern alles in zwei Dimensionen modelliert (Da manche Teams in der Lage sind, den Ball hoch zu schießen, wird durch diese Vereinfachung ein Fehler gemacht, jedoch kann mit nur einer Kamera nicht festgestellt werden, ob der in der Luft ist). Der Ball wird daher durch einen Kreis mit konstantem Radius repräsentiert, wobei das Kreiszentrum der im Bild nächste Ballpunkt ist. Die Bestimmung des nächsten Ballpunktes ist kritisch, da aus seiner Veränderung der Bewegungszustand des Balles bestimmt werden wird. Wegen der Verzerrungen durch den Spiegel und der schwankenden Lichverhältnisse ist die Bestimmung dieses Punktes problematisch. Die Schwierigkeit besteht darin, den Ball auf einen einzigen Pixel zu reduzieren und in aufeinander folgenden Bildern stets denselben Pixel zu selektieren. Um die Robustheit zu erhöhen, sollten daher alle Pixel, die zum Ball gehören, zur Bestimmung des nächsten Ballpunktes beitragen. Die manuell kalibrierte Look-Up Tabelle enthält jedoch bei weitem nicht alle Farbtöne, die zum Ball gehören und da bei der genauen Positionsbestimmung nur ein kleiner Bildausschnitt betrachtet wird, ist es nun angebracht für jeden Pixel den Farbabstand zu einer konstanten Referenzfarbe zu berechnen. Dabei reicht es nicht sich auf die Farbinformation ausschließlich zu verlassen, da der Ball an der oberen Seite wegen seiner leicht glänzenden Oberfläche nahezu weiß ist. Es ist jedoch bekannt, dass durch die Benutzung des omnidirektionalen Spiegels der Ball im Bild die Form einer Ellipse hat. Durch das Anpassen einer Ellipse um die rötlichen Pixel kann die Silhouette des Balles sehr gut bestimmt werden. Das Zentrum dieser Ellipse ist recht robust, da Information aus vielen Pixeln darin zusammengefasst ist. Im Folgenden wird der Vorgang genauer erläutert.

Das Bild wird als erstes mit einem 2D Gauß-Filter mit einer Standardabweichung von sieben Pixeln geglättet, um die von der Kamera verursachten Farbverfälschungen abzuschwächen. Anschließend wird das Bild aus dem RGB-Farbraum in den HSI / HSV-Farbraum (Hue = Farbton, $Saturation$ = Sättigung und $Intensity$ = Helligkeit oder $Value$ = Wert) transformiert. Der HSI-Farbraum ist eine nicht-lineare Transformation des RGB-Farbraumes und wurde 1978 von Alvy Ray Smith eingeführt [32]. Dieser Farbraum ist für Farbklassifizierungen besser geeignet, da sich Helligkeitsschwankungen im RGB-Farbraum auf alle drei Kanäle bemerkbar machen, der HSI-Farbraum aber so konzipiert ist, dass sich (theoretisch) reine Helligkeitsunterschiede nur auf dem I Kanal auswirken. In der Praxis hat es sich gezeigt, dass sich Änderungen der Lichtverhältnisse auch zum Teil auf den H und S Kanal auswirken. Das bei der Farbklassifikation im HSI-Farbraum erzielte Ergebnis ist dennoch deutlich besser als das im RGB-Farbraum erzielbare und rechtfertigt damit

den Wechsel des Farbraumes. Die Umrechnung von RGB nach HSI läßt sich durch eine Rotation und eine Normierung berechnen:

$$\begin{pmatrix} m_1 \\ m_2 \\ i_1 \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix},$$

$$I = \sqrt{3} \cdot i_1, \qquad S = \sqrt{m_1^2 + m_2^2}, \qquad H = \arctan\left(\frac{m_1}{m_2}\right).$$

Während der RGB-Farbraum durch einen Würfel repräsentiert werden kann (wobei $R, G, B \in [0; 255]$), entspricht der HSI-Farbraum einem Kegel. Punkte im HSI-Farbraum werden durch eine Art Polarkoordinaten angegeben, wobei H ein Winkel zwischen 0 und 360 Grad ist. Da jedoch nur ein Byte zum Speichern des Wertes benutzt werden soll, wird der Wert halbiert. S und I nehmen weiterhin Werte aus dem Intervall $[0; 255]$ an.

Da Unterschiede in der Helligkeit in unserem Fall irrelevant sind, werden die Pixel nur anhand des H und des S Kanales klassifiziert. Zur Klassifikation wird für jeden Pixel der euklidische Abstand zu einer vorher festgelegten Referenzfarbe, in diesem Falle Rot, berechnet. Dadurch erhält man ein Distanzbild D wie in Abbildung 3.2(c) zu sehen:

$$D = \sqrt{(H - 0)^2 + (S - 255)^2}.$$

Die Wahrscheinlichkeit, dass ein Pixel zum Ball gehört, ist hoch, wenn sein Abstand zur Referenzfarbe klein ist. Indem man das Distanzbild mit einem Schwellwert binarisiert erhält man eine Maske, die nur diejenigen Pixel enthält, deren Abstand kleiner als der Schwellwert ist. Die Wahl des Schwellwertes ist sehr wichtig und beeinflußt das Endergebnis sehr stark. Während ein zu kleiner Schwellwert dazu führt, dass nur das Ballzentrum als Ball erkannt wird, hat ein zu groß gewählter Schwellwert zur Folge, dass Regionen als Ball erkannt werden, die zu angrenzenden Objekten gehören. Ein zu kleiner Schwellwert führt zu einem Fehler, dieser ist jedoch nicht fatal. Dagegen führt ein zu großer Schwellwert, z.B. wenn der Ball vor dem gelben Tor ist, dazu, dass fast das gesamte Bild als Ball markiert wird. Wegen der wechselnden Lichtverhältnisse ist es nicht möglich einen konstanten Schwellwert zu benutzen. Eine einfache Lösung ist, den Schwellwert von der mittleren Bildhelligkeit (da man das Distanzbild betrachtet entspricht die mittlere Helligkeit dem durchschnittlichen

Abstand der im Bild auftretenden Farben zur Referenzfarbe) abhängig zu machen. Diese Lösung ist jedoch nicht robust genug und funktioniert nicht im Falle, dass der Ball vor dem gelben Tor ist.

Stattdessen wird das Histogramm h des Distanzbildes berechnet und der Schwellwert c so gewählt, dass in etwa eine Fläche von der zu erwartenden Größe des Balles selektiert wird [16]. Die Größe dieser Fläche ist abhängig von der Entfernung des Balles zum Roboter (da die Ballgröße mit wachsender Entfernung abnimmt). Empirisch wurde folgende Funktion für die erwartete Ballgröße $A(d)$ in Abhängigkeit von der Entfernung d zwischen Roboter und Ball bestimmt:

$$A(d) = 500 \cdot e^{\frac{38-d}{180}} + 40.$$

Das Histogramm h ist die Dichtefunktion der Abstände im Distanzbild und es gilt:

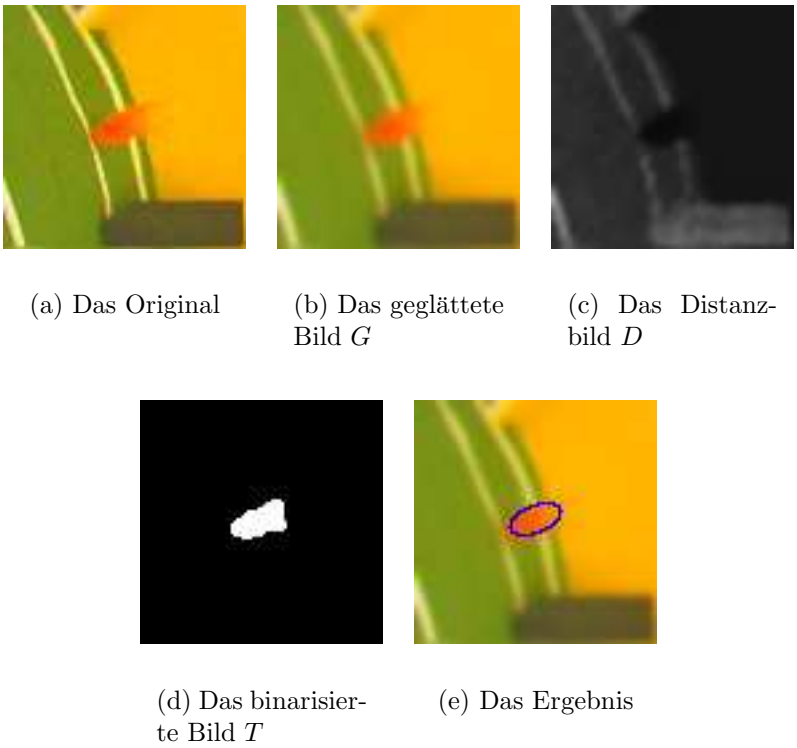
$$c = \underset{k}{\operatorname{argmax}} \sum_{i=0}^{k} h(k), \text{unter der Bedingung: } h(k) < A(d)$$

Für das binarisierte Bild T (Abbildung 3.2(d)) gilt dann: $T = D < c$. Wenn der Ball zum Teil verdeckt ist, wird ein zu großer Teil des Bildes selektiert. Dieser Fehler ist jedoch nicht problematisch, da die Teile des Roboters, die den Ball verdecken dunkel sind. Es werden also Pixel selektiert werden, die nicht zum Ball gehören, aber mit hoher Wahrscheinlichkeit auch nicht nahe am Ball sind. Im folgenden wird gezeigt werden, wie diese fälschlicher Weise selektierten Pixel beseitigt werden können.

Wegen Reflektionen auf der glatten Oberfläche des Balles erscheinen manche Pixel nahezu weiß und haben daher einen zu großen Abstand zum Referenzrot, obwohl sie Teil des Balles sind. Um die Silhouette des Balles zu erhalten, wurden zwei verschiedene Ansätze getestet:

1. Das Distanzbild wird mit einem relativ hohen Schwellwert binarisiert. Morphologisches Schließen [17] vervollständigt die Silhouette, indem es nahestehende aber nicht miteinander verbundene rote Pixel verbindet und dadurch .

2. Durch starkes Glätten des Distanzbildes mit einem neun Pixel großen Gaußschen Kernel und anschließendes Binarisieren mit einem etwas niedrigeren Schwellwert kann man die Silhouette ebenfalls erhalten.



(a) Das Original

(b) Das geglättete Bild G

(c) Das Distanzbild D

(d) Das binarisierte Bild T

(e) Das Ergebnis

Abbildung 3.2: Die Zwischenschritte bei der Balllokalisierung

Da morphologische Operationen teurer sind und oft auch schlechtere Ergebnisse als das zusätzliche Glätten geliefert haben, wurde die zweite Variante gewählt.

Wenn der Ball zum Teil verdeckt ist, oder es andere rötliche Objekte im Bild gibt, kann es jedoch passieren, dass diese auch als Ball erkannt werden. Da Roboter keine rote Farbe tragen dürfen, ist es jedoch ausgeschlossen, dass der Ball durch einen rötlichen Gegenstand verdeckt wird. Es werden daher die zusammenhängenden Flächen durchnummeriert und anschließend deren Flächen bestimmt. Man sucht nur einen Ball und behält daher nur die größte, zusammenhängende Komponente, da angenommen wird, dass der Ball das größte rote Objekt auf dem Spielfeld ist.

Da die Silhouette immernoch nicht perfekt ist, wird sie am Ende durch eine Ellipse umschlossen [21]. Der Mittelpunkt dieser Ellipse wird als das Ballzentrum benutzt. Um die Abstandsfunktion anwenden zu können, wird ein Punkt auf dem Spielfeld benötigt (also mit Höhe Null). Die Hauptachse der Ellipse zeigt auf den nächsten Punkt des Balles, jedoch sollte diese nicht benutzt werden, da die Ballposition dadurch instabil wird. Alle selektierten Pixel tragen zur Bestimmung des Zentrums bei, wodurch dieses gegen schwankende Lichtverhältnisse und Farbverfälschungen am Rand durch die niedrig aufgelöste

Kamera resistent ist. Dagegen ändert sich gerade die Größe der Ellipse durch die schwankenden Lichtverhältnisse stark, da die Pixel am Rand teilweise selektiert und teilweise nicht selektiert werden. Auch an dieser Stelle kann eine Heuristik mehr Stabilität bringen. Wie schon erwähnt, ist die zu erwartende Ballgröße bekannt, es ist aber auch das Verhältnis von Haupt- zu Nebenachse der Ellipse bekannt. Daher läßt sich die zu erwartende Länge der Hauptachse leicht berechnen. Das Ballzentrum wird um diesen Wert in Richtung der Spiegelmitte verschoben, wodurch man einen robusten Punkt auf dem Spielfeld erhält.

3.1.3 Optimierung für nahe Bälle

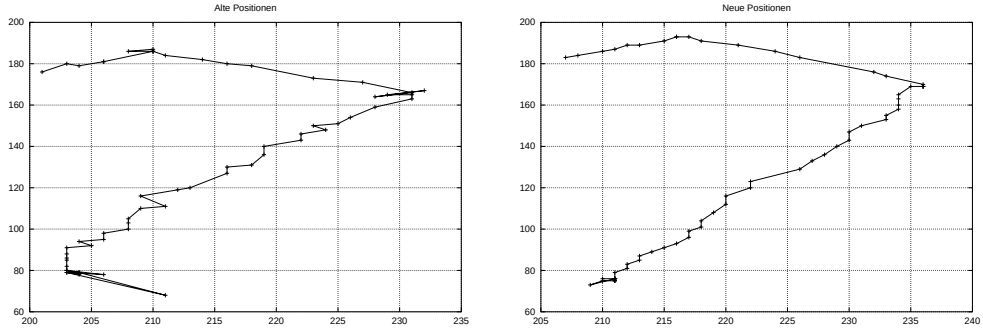
Um den Ball platziert zu schießen oder zu dribbeln, ist es sehr wichtig den Abstand zwischen Ball und Roboter möglichst exakt zu kennen. Wie in Abschnitt 2.3 erwähnt, ist es jedoch nicht möglich Abstände unter 30 cm genau zu bestimmen, da die Objekte in dieser Entfernung durch den Spiegel sehr stark verzerrt werden. Die Roboter haben einen Radius von ca. 20 cm, auf der vorderen Seite etwas weniger, so dass Bälle, die nah am Roboter liegen genau in diesen Bereich fallen. Die genaue Positionsbestimmung wird zusätzlich erschwert, da in diesem Bereich der Ball im Bild vom Roboter verdeckt wird.

Um dieses Problem zu lösen, musste eine weitere Heuristik in den Algorithmus zur genauen Positionsbestimmung ergänzt werden. Die Idee dabei ist sich am weitesten entfernten Ballpunkt zu orientieren und nicht wie zuvor am nächsten, da dieser nicht sichtbar ist. Der am weitesten entfernte Ballpunkt ist stets sichtbar (da der Ball nur zu weniger als einem Drittel verdeckt werden darf) und daher robust bestimmbar. Außerdem ist er weiter entfernt vom Roboter, so dass die Abstandsfunktion angewandt werden kann. Da der Ballradius bekannt ist, kann aus dem am weitesten entfernten Ballpunkt auf den nächsten geschlossen werden.

Um diese Heuristik benutzen zu können, musste lediglich empirisch bestimmt werden, ab welcher Entfernung der Ball vom Roboter verdeckt wird und wie die Ballgröße im Bild abnimmt.

3.1.4 Ergebnisse

In Abbildung 3.3 ist die Ballposition in Bildkoordinaten dargestellt. Es fällt auf, dass die neue Lokalisierung des Balles robuster und der Graph der Ballpositionen viel glatter als zuvor ist. Dadurch lassen sich die Bewegungsrichtung und -geschwindigekit des Balles mit einem kleineren Fehler vorhersagen. Zur Berechnung der alten Ballposition wurde der nächste Ballpunkt nur anhand der manuell initizialisierten Look-Up Tabelle bestimmt.



(a) Alte Ball-Positionen (b) Neue Ball-Positionen

Abbildung 3.3: Alte und neue Ball-Positionen in Bild-Koordinaten.

3.2 Kalman-Filter zur verbesserten Positionsabschätzung

Im vorhergehenden Abschnitt wurde erläutert, wie der Ball im Bild gefunden wird und ein möglichst zuverlässig wiedererkennbarer Punkt bestimmt werden kann. Wie die Ergebnisse zeigen, ist die Ballposition durch diese Methode recht robust bestimmbar. Wenn jedoch aus zwei aufeinander folgenden Ballpositionen die Ballgeschwindigkeit bestimmt werden soll, zeigt sich, dass die Messungen zu unsicher und ungenau sind. Eine Möglichkeit die Ballgeschwindigkeit genauer zu berechnen besteht darin ein Kalman-Filter zu benutzen. Für Bezeichnungen und eine kurze Einführung in die Theorie des Kalman-Filters wird auf Anhang A verwiesen.

3.2.1 Bewegungsmodell

Als erstes wird der im Bild bestimmte Punkt aus den Bildkoordinaten mit Hilfe der Positionsfunktion in Weltkoordinaten, d.h. in einen Punkt auf dem Spielfeld, umgerechnet. Der Systemzustand x_k des Balles ist durch seine Position auf dem Feld (x und y Koordinate) und seiner Geschwindigkeit v eindeutig bestimmt (man beachte den Unterschied zwischen Systemzustand $x_\mathbf{k}$ und der Koordinate x):

$$x_k = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}.$$

Da das Bewegungsmodell linear ist, kann ein einfaches Kalman-Filter benutzt werden (man benötigt kein erweitertes Kalman-Filter). Es wird angenommen, dass der Ball sich in der selben Richtung weiterbewegen wird, jedoch durch die Reibung (als Reibungskoeffizient wird $\rho = 0.99$ angenommen) leicht abgebremst werden wird. Da keine Zusammenstöße mit anderen Objekten modelliert werden, ist die Eingabe stets Null. Exemplarisch sei hier die Zustandspropagation dargestellt, welche sich durch die Annahmen zu der folgenden Beziehung vereinfacht:

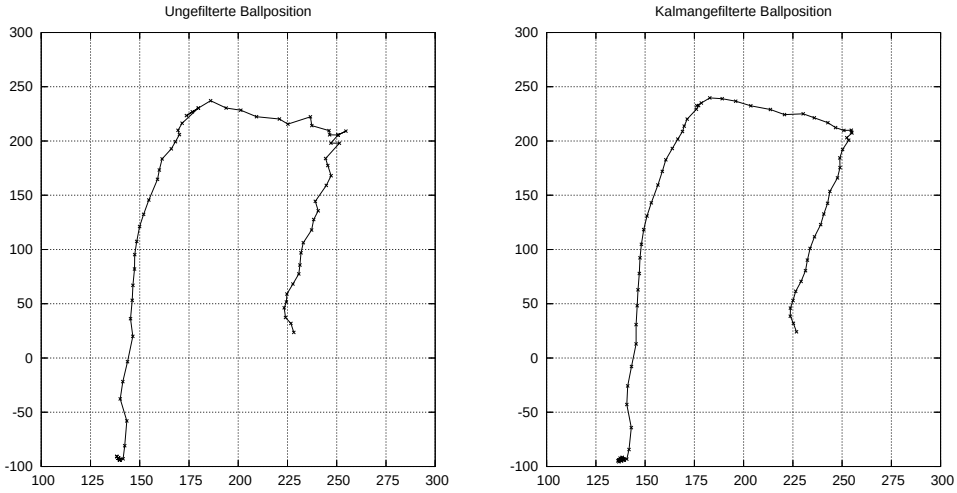
$$\bar{x_k} = \begin{pmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & \rho & 0 \\ 0 & 0 & 0 & \rho \end{pmatrix} x_{k-1},$$

wobei t die Zeit zwischen zwei Zeitschritten ist.

Da der Fehler in der Messung mit steigender Entfernung wächst, wird die Kovarianzmatrix der Messung in Abhängigkeit von der Ballentfernung neu berechnet, wobei angenommen wird, dass der Messfehler proportional zur Ballentfernung ist.

Die Kovarianzmatrix gibt an, wie sehr man der Messung vertraut und beeinflußt das Kalman-Filter stark. Einerseits möchte man, dass wenn der Ball ruht, die Ballposition konstant ist. Es wird zwar so sein, dass der im Bild als nächster Ballpunkt bestimmte Punkt um ein oder zwei Pixel schwankt, jedoch sollte die gefilterte Position unverändert bleiben. In diesem Fall ist eine größere Trägheit von Vorteil: der Messung wird eine geringe Bedeutung gegeben und es wird mehr dem Bewegungsmodell vertraut (welches in dieser Situation eine ruhende Position vorhersagen würde). Andererseits ist es vor allem für den Torwart von größter Bedeutung, so schnell wie möglich festzustellen, dass der Ball angefangen hat sich zu bewegen. Dazu muss der Messung stärker vertraut werden und es wird in Kauf genommen dadurch keine glatte Position mehr zu erhalten, wenn der Ball ruht. Es ist nicht möglich die Trägheit des Kalman-Filters so zu wählen, dass in allen Situationen ein optimales Ergebnis erreicht wird.

Eine mögliche Erweiterung des Bewegungsmodells besteht darin, die Kraft, die beim Dribbeln und beim schießen auf den Ball wirkt, als Eingangsgröße einfließen zu lassen. Da der Kontakt mit dem Ball wegen der schwachen Kameraauflösung nicht genau bestimmt werden kann, wurde vorerst auf diese Erweiterung verzichtet.



(a) Ungefilterte Ballposition (b) Kalmangefilterte Ballposition

Abbildung 3.4: Ungefilterte und kalmangefilterte Ballposition in Spielfeld-Koordinaten.

3.2.2 Ergebnisse

In Abbildung 3.4(a) ist die ungefilterte, aus dem Bild heraustransferierte Ballposition dargestellt. In 3.4(b) dagegen ist die kalmangefilterte Position aufgezeichnet. Die Verbesserungen durch das Kalman-Filter sollten sofort sichtbar sein.

3.2.3 Ballvorhersage

Der Ball ist oft für den Roboter nicht sichtbar, da er durch einen anderen Roboter verdeckt ist. Dank eines Zentralrechners, mit dem jedes Teammitglied kommuniziert, ist es möglich die Ballposition, die von den anderen Mitspielern bestimmt worden ist, zu erhalten. Jedoch ist die Kommunikation über den Zentralrechner mit einer hohen Verzögerung belastet und außerdem wegen oft auftretender Probleme mit dem Funknetz nicht gesichert. Daher ist es hilfreich im Fall, dass kein Ball im Bild gefunden werden kann, die Ballposition anhand von Daten aus der Vergangenheit vorherzusagen.

Dank des benutzten Kalman-Filters ist dies ohne größere Umstände möglich. Das Kalman-Filter durchläuft in jeder Iteration einen zweistufigen Prozess. Zuerst wird die neue Ballposition und -geschwindigkeit vorhergesagt und anschließend diese durch die Messung aktualisiert. Wenn das Kalman-

Filter zur Vorhersage benutzt werden soll, wird einfach auf die Aktualisierung durch die Messung verzichtet (da diese nicht existiert).

Abbildung 3.5: Spielfeld mit Hindernis aus der Sicht des Roboters

Versuche haben gezeigt, dass wenn der Ball in mindestens zehn aufeinander folgenden Bildern gesehen wurde, das Bewegungsmodell meist gut genug ist, um die Ballposition für ca. 20 weitere Bilder vorherzusagen, unter der Annahme, dass der Ball frei rollt.

Ergebnisse

Um die Ballvorhersage zu testen, wurde ein Hindernis vor dem Roboter positioniert und der Ball hinter dem Hindernis entlang gerollt. Die Sicht des Roboters ist in Abbildung 3.5 zu sehen. Die Position von Roboter und Hindernis, sowie die Ballposition sind in Abbildung 3.6 zu sehen, wobei der vorhergesagte Teil markiert ist. Der Fehler in der vorhergesagten Ballposition akkumuliert sich, so dass ein kleiner Sprung entsteht, sobald der Ball hinter dem Hindernis wieder hervorkommt (rot umkreiste Stelle). Dieser Fehler kann auf zwei Arten erklärt werden:

1. Der angenommene Reibung $\rho = 0.99$ ist zu stark, so dass der Ball im Modell stärker abgebremst wird als in der Realität. Experimente haben jedoch gezeigt, dass ein schwächerer Reibungskoeffizient dazu führen kann, dass die Vorhersage überschießt, d.h. der vorhergesagte Ball eine größere Entfernung zurücklegt, als der reale Ball.

2. Die angenommene Ballgeschwindigkeit ist zu gering. Das Berechnen der Ballgeschwindigkeit wird entscheidend von der im Kalman-Filter benutzten Kovarianzmatrix beeinflußt. Eine zu geringe Ballgeschwindigkeit ist

ein Indiz dafür, dass das Kalman-Filter zu träge ist. Um die Trägheit zu verringern müsste das Vertrauen in die Messung erhöht werden, was jedoch dazu führt, dass die Ballposition nicht mehr so stabil sein wird (siehe Abschnitt 3.2.1).

Aus diesen beiden Gründen ist es schwer, die Ballvorhersage zu verbessern, ohne dafür andere Nachteile in Kauf nehmen zu müssen. Obwohl die vorhergesagte Ballposition mit einem Fehler behaftet ist, ist sie aber ausreichend, um den Roboter sinnvoll zu positionieren. Sobald der Roboter nah am Ball ist, wird dieser wieder sichtbar werden und die Ballposition dadurch wieder exakt bestimmbar.

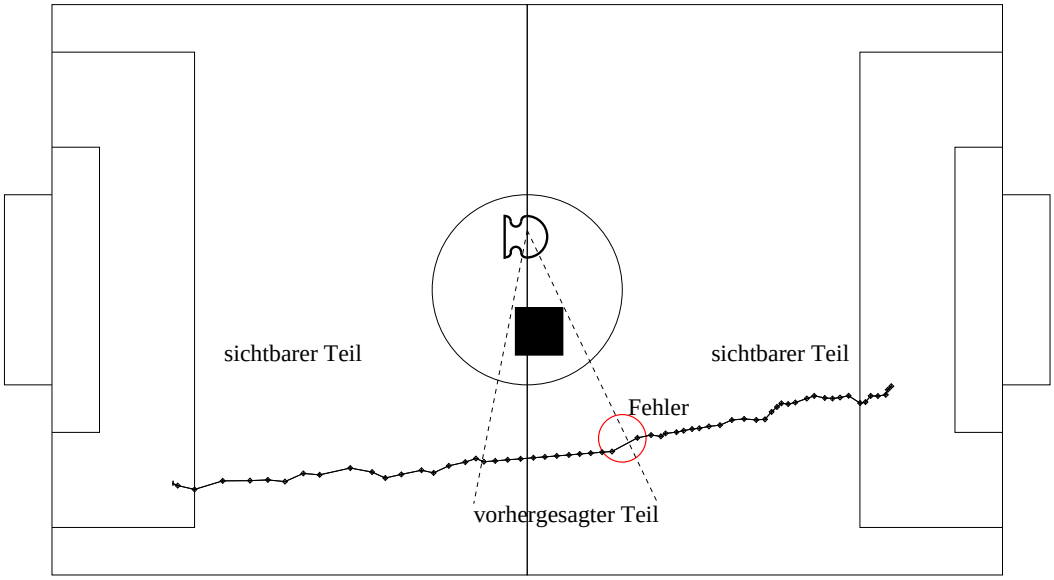


Abbildung 3.6: Versuchsaufbau und vorhergesagte Ballpositionen

3.2.4 Ausblick

Sobald Hindernisse auf dem Spielfeld (also Roboter der eigenen und der gegnerischen Mannschaft) zuverlässig erkannt und verfolgt werden können, sollte jedes Objekt mit einem Kalman-Filter assoziiert werden. Das Kalman-Filter liefert genauere Information über die Position und die Bewegungsrichtung der Objekte. Von diesem Informationsgewinn würde in erster Linie die Pfadplanung profitieren: Beim Planen eine Pfades könnten dann nicht nur die aktuellen Hindernispositionen berücksichtigt werden, sondern auch deren aktuelle Geschwindigkeit. Dadurch könnte die Hindernisposition zu einem späteren Zeitpunkt vorhergesagt werden und Kollisionen würden vermieden.

3.3 Ballfusion

In diesem Abschnitt wird eine Methode zur Repräsentation, Kommunikation und Fusion von verteilten, verrauschten und unsicheren Beobachtungen eines Objektes durch verschiedene Roboter präsentiert. [28] und [34] adressieren dasselbe Problem und benutzen einen ähnlichen Ansatz, jedoch werden teilweise einschränkende Annahmen gemacht, die bei unserem Ansatz nicht nötig sind.

üblicherweise überblickt jeder einzelne Roboter zu einem bestimmten Zeitpunkt nur einen Teil des Spielfeldes. In einer dynamischen Umgebung wird die vorher gesammelte Information über zur Zeit nicht sichtbare Objekte schnell ungenau und unbrauchbar. Das Austauschen von Information zwischen den Robotern erhöht die effektive Sichtweite und erlaubt so eine genauere Modellierung der Welt. Wenn Informationen, die aus verschiedenen Blickpunkten gemacht worden sind, effizient ausgetauscht werden, kann zusätzlich die Genauigkeit der Messungen erhöht werden. Um den zeitlichen Anforderungen der sich schnell verändernden Umgebung zu genügen, müssen die ausgetauschte Information, sowie der Rechenaufwand zur Kombination mehrerer Beobachtungen minimal sein. Der hier vorgeschlagene Ansatz verwendet eine zweidimensionale statistische Repräsentation der Beobachtungen und verlangt wenige Rechenschritte zur Fusion.

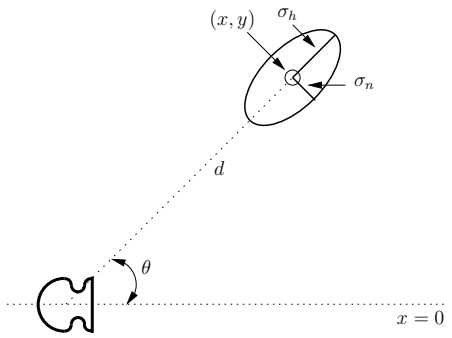


Abbildung 3.7: Die Parameter der Gauß-Verteilung: Mittelwert (x, y), Winkel der Hauptachse θ, Standardabweichung entlang der Hauptachse (σ_h) und der Nebenachse (σ_n), sowie der Abstand d zwischen Roboter und Mittelwert.

3.3.1 Vereinfachtes Beispiel

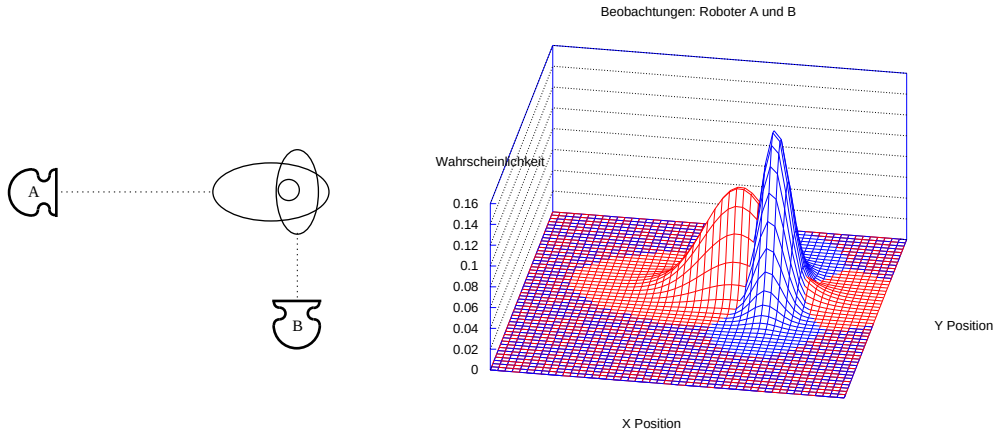
An einem vereinfachten Beispiel soll die Grundidee veranschaulicht werden, bevor die mathematischen Details dargelegt werden.

Die Position des Balles kann als eine zweidimensionale Gauß-Verteilung modelliert werden. Das Zentrum oder der Mittelwert der Verteilung ist der

in Abschnitt 3.1.2 mit dem Kalman-Filter bestimmte Punkt und die Standardabweichungen entlang der Haupt- und Nebenachse korrespondieren zur Messunsicherheit in die jeweilige Richtung. Der Wert der Verteilung an einem beliebigen Punkt entspricht der Wahrscheinlichkeit, dass sich das Objekt unter Voraussetzung der Beobachtung dort befindet (siehe Abbildung 3.7).

Bei omnidirektionalen Kameras ist die Messunsicherheit bei der Richtungsbestimmung stets geringer als die Messunsicherheit bei der Abstandsbestimmung. Deswegen ist die Projektion der Gauß-Verteilung stark ellipseförmig. Außerdem nimmt die Messunsicherheit proportional zur Entfernung zu.

Die Fusion ist symmetrisch und assoziativ, weil die Fusion zweier Gauß-Verteilungen wiederum eine Gauß-Verteilung ist. Dies ermöglicht die Fusion beliebig vieler Beobachtungen. Die Idee ist also, die Beobachtungen der einzelnen Roboter zu sammeln, die korrespondieren Gauß-Verteilungen zu fusionieren und dadurch eine bessere Abschätzung der Position und des Messfehlers zu erhalten.



(a) Modelaufbau (b) Gauß-Verteilungen der Beobachtungen

Abbildung 3.8: Zwei Roboter sehen den Ball aus verschiedenen Richtungen und Entfernungen.

3.3.2 Fusion von Gauß-Verteilungen

Die kanonische Form einer zweidimensionalen Gauß-Verteilung $p(X)$ hängt vom Mittelwert $\overline{X}$ und der Kovarianzmatrix Σ ab, wobei sich Σ durch die Standardabweichungen σ_x, σ_y und dem Korrelationsfaktor ρ berechnen läßt:

$$p(X) = \frac{1}{2\pi|\Sigma|} e^{-\frac{1}{2}\left(X-\overline{X}\right)^T \Sigma^{-1}\left(X-\overline{X}\right)}, \qquad \Sigma = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}.$$

Seien $G_1 = (\overline{X_1}, \Sigma_1)$ und $G_2 = (\overline{X_2}, \Sigma_2)$ die Gauß-Verteilungen der beiden Beobachtungen. Die fusionierte Normalverteilung $G_3 = (\overline{X_3}, \Sigma_3)$ ist schlicht das Produkt von G_1 und G_2. Mathematische Umformungen führen zur Beziehung $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1}$. Smith und Cheesmans zeigen jedoch in [33] einen effizienteren Weg zur Berechnung der Kovarianzmatrix und des Mittelwertes.

$$\Sigma_3 = \Sigma_1 - \Sigma_1 \cdot (\Sigma_1 + \Sigma_2)^{-1} \cdot \Sigma_1,$$

$$\overline{X_3} = \overline{X_1} - \Sigma_1 \cdot (\Sigma_1 + \Sigma_2)^{-1} \cdot \left(\overline{X_2} - \overline{X_1}\right).$$

In Abbildung 3.9 ist die fusionierte Gauß-Verteilung aus dem simulierten Beispiel dargestellt. Durch die Berücksichtigung der beiden Beobachtungen hat sich der Mittelwert leicht verschoben, zusätzlich sind die Standardabweichungen (der Messfehler) kleiner geworden und die Genauigkeit beim Mittelwert hat sich erhöht.

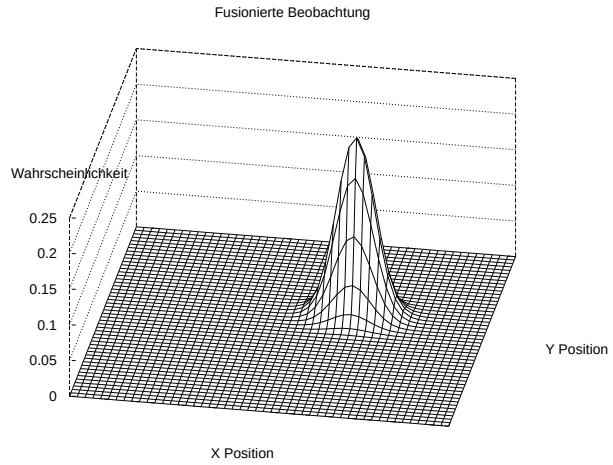


Abbildung 3.9: Die fusionierte Gauß-Verteilung mit kleineren Standardabweichungen (Fehler) und höherer Genauigkeit beim Mittelwert.

3.3.3 Konkrete Umsetzung

Die Ballposition ist zweifellos die wichtigste Information über den Ball. In vielen Situation ist jedoch die Ballgeschwindigkeit von genauso großer Bedeutung.

Daher wird anstatt der zweidimensionalen Normalverteilung eine vierdimensionale verwandt.

Dadurch dass jeder Roboter ein stets aktuelles Bewegungsmodell des Balles mit Hilfe eines Kalman-Filters berechnet, ist die benötigte Information über die Messunsicherheit schon vorhanden. Von Vorteil erweist es sich auch, dass das Bewegungsmodell des Kalman-Filters mit dem globalen Koordinatensystem arbeitet (und nicht im lokalen Koordinatensystem, d.h. relativ vom Roboter aus betrachtet). Dies ermöglicht die direkte Fusion der Beobachtungen, ohne vorherige Koordinatentransformationen.

Im vorherigen Abschnitt über die Fusion von Gauß-Verteilungen wurde angenommen, dass die Roboter perfekt lokalisiert sind. Dies ist im Allgemeinen nicht der Fall. Wenn der Fehler in der Lokalisierung ebenfalls als Gauß-Verteilung modelliert werden kann, kann er auf ähnliche Art wie die Fusion einbezogen werden. Wie in Kapitel 4 erläutert werden wird, wird zur Lokalisierung des Roboters unter anderem ebenfalls ein Kalman-Filter eingesetzt. Um die Unsicherheit in der Lokalisierung des Roboters in die Fusion einfließen zu lassen, wird jede Messung zusätzlich verrauscht. Da die Kalman-Filter mit dem selben globalen Koordinatensystem arbeiten reicht es die Kovarianzmatrix der Lokalisierung (Σ_L) auf die der Messung (Σ_M) auf zu addieren:

$$\Sigma_M = \Sigma_M + \Sigma_L$$

anschließend können die Messungen wie in Abschnitt 3.3.2 fusioniert werden.

Die Roboter stehen in ständigem Kontakt mit dem Zentralrechner, kommunizieren jedoch nicht direkt untereinander. Die Entscheidung, die Ballfusion im Zentralrechner zu berechnen, hat Vorteile als auch Nachteile. Der gravierendste Nachteil ist die etwas größere Verzögerung, da nach der Berechnung der Fusion, die Information noch an die Roboter gesandt werden muss. Vorteilhaft ist aber, dass deutlich weniger Information ausgetauscht werden muss: anstatt des Austausches der Information zwischen jedem Paar von Robotern, muss jeder Roboter lediglich dem Server melden ob und wenn ja wo und mit welcher Messunsicherheit der Ball lokalisiert worden ist. Im Gegenzug sendet der Server ein fusioniertes Weltbild an alle Roboter, wodurch sichergestellt ist, dass alle Roboter dasselbe globale Weltbild haben.

3.3.4 Behandlung von Ausreißern

Obwohl die Fusion meist zu einer Verbesserung der Genauigkeit von Ballposition und -geschwindigkeit führt, gibt es auch Situationen, in denen sie nicht sinnvoll ist. Wenn sich ein Roboter sehr schlecht lokalisiert hat oder einen

falschen Ball gefunden hat, würde eine Fusion mit den anderen Beobachtungen zu einer Verschlechterung der korrekten Messungen führen. Daher wird ein Mechanismus benötigt, der feststellt, welche Beobachtungen bei der Fusion berücksichtigt werden sollten.

In [28] wird eine Nutzenfunktion definiert, anhand derer entschieden wird, ob eine Fusion vernünftig ist. Wenn die Nutzenfunktion nach der Fusion unimodal ist, dann sollten die Beobachtungen fusioniert werden, ansonsten nicht.

In Anlehnung daran haben wir festgelegt, dass zwei Beobachtungen nur dann miteinander fusioniert werden, wenn deren Mahalanobis Abstand kleiner als der zweifache Balldurchmesser ist. Der Mahalanobis Abstand unterscheidet sich vom euklidischen Abstand darin, dass der Abstand in jeder Komponente mit der Inversen der Varianz in dieser Dimension skaliert wird. Das hat zur Folge, dass ein Abstand x in einer Richtung, in der die Streuung gering ist, sich stärker auf den Gesamtabstand auswirkt als derselbe Abstand x in einer Richtung mit großer Varianz.

3.3.5 Ausblick

Ein globales Weltbild sollte nicht nur das Spielfeld und den Ball enthalten, sondern auch die Positionen der Roboter der eigenen und der gegnerischen Mannschaft. Jeder Roboter versucht die Hindernisse auf dem Spielfeld zu erkennen und ihnen auszuweichen. Da die Roboter jedoch nicht das ganze Spielfeld überblicken können und ihnen teilweise die Sicht versperrt ist, ist diese Information nie vollständig. Das Prinzip der Ballfusion läßt sich auch auf andere Objekte erweitern. Dabei tritt jedoch ein neues Problem auf. Es gibt nur einen Ball, so dass problemlos die einzelnen Beobachtungen fusioniert werden können. Bei Hindernissen auf dem Spielfeld muss zuerst entschieden werden, welche Beobachtungen sich auf dasselbe Objekt beziehen. Ein nicht triviales Problem, das in Zukunft angegangen werden sollte.

Kapitel 4

Sensorfusion

4.1 Einführung

Viele robotische Systeme kombinieren Vision und Odometrie, um eine bessere Lokalisierung zu erreichen [3], [12]. Vision allein hat eine relativ lange Verzögerung und liefert Ergebnisse mit starkem Rauschen. Dagegen führt eine Lokalisierung nur anhand des Odometers wegen durchdrehender Räder, Unebenheiten auf dem Spielfeld und der Diskretisierung der Raddrehungen zu einem unbeschränkten Positionsfehler. Ansätze zur Korrektur dieser Fehler existieren zwar ([5], [10]), sind aber nicht ausreichend, um eine zuverlässige Lokalisierung für längere Zeit zu garantieren. Wenn dagegen beide Sensoren kombiniert werden, treten synergetische Effekte auf. Einerseits wird die visuelle Verarbeitung einfacher, da durch die Odometrie eine bessere Vorhersage der zu erwartenden Positionen von Objekten im Bild möglich ist und andererseits wird das Rauschen durch die zusätzlichen Daten des Odometers verringert. Beides ermöglicht eine genauere Lokalisierung.

Sehr häufig wird zur Fusion der Daten ein diskretes Kalman-Filter (siehe Anhang A, sowie [39]) benutzt [18], [20], [15]. Dabei wird die Behandlung von Daten mit unterschiedlichen zeitlichen Verzögerungen zu einem der Hauptprobleme. Da der Odometer eine kürzere Wahrnehmungsverzögerung als die Kamera hat, gibt es odometrische Daten zu denen noch keine visuellen Daten zur Verfügung stehen, wie in Bild 4.1 zu sehen ist.

Das folgende Beispiel soll das auftretende Problem illustrieren: Wenn der Roboter anfängt sich zu bewegen, wird diese Bewegung sehr schnell (mit einer Verzögerung von etwa 10 ms) vom Odometer angezeigt werden. Wegen der deutlich längeren visuellen Verzögerung (etwa 70 ms) werden sich die Bilder jedoch zu Beginn der Bewegung nicht ändern. D.h. es gibt eine Zeitspanne, in der die odometrischen Daten eine Bewegung anzeigen, aber die visuellen Daten konstant bleiben.

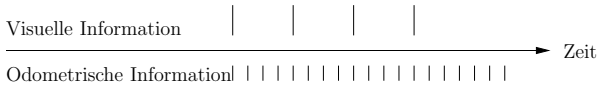


Abbildung 4.1: Da der Odometer eine kürzere Wahrnehmungsverzögerung als die Kamera hat, gibt es odometrische Daten zu denen noch keine visuellen Daten zur Verfügung stehen.

Wenn ein diskretes Kalman-Filter zur Fusion von Daten von verschiedenen Sensoren benutzt werden soll, dann müssen die Messungen vom selben Systemzustand (in der Zeit) stammen. Normalerweise ist dies nicht der Fall. Im obigen Beispiel, bei dem der Roboter anfängt sich zu bewegen, stammt die odometrische Information von einem Systemzustand, in dem sich der Roboter bewegt. Dagegen stammt die visuelle Information vom Zustand, als der Roboter noch ruhte.

Der Unterschied zwischen den Verzögerungen der Sensoren ist oft klein (etwa 60 ms) und wird daher in der Literatur meist vernachlässigt. Verschiedene Ansätze beschäftigen sich mit der Fusion von asynchronen Sensoren ([15], [24]), aber vernachlässigen dabei die Wahrnehmungsverzögerungen der Sensoren. Soweit uns bekannt, haben sich Larsen et al. [20] als einzige mit der expliziten Modellierung dieser Verzögerung beschäftigt. Sie benutzen ein diskretes Kalman-Filter zur Fusion und extrapolieren die verzögerten Daten, um dadurch Messungen zu erhalten, die vom gleichen Systemzustand stammen. Unserer Meinung nach ist es nicht sinnvoll die Daten zu extrapolieren, da dadurch die Unsicherheit in den Daten erhöht wird.

Dagegen schlagen wir einen zwei-stufigen Fusionsprozess vor. Dabei werden die odometrischen Daten in zwei Teile aufgespalten: einen Teil zu dem es korrespondierende visuelle Daten gibt und einen Teil zu dem es keine gibt. Im ersten Schritt werden die zusammengehörenden Daten mit einem Kalman-Filter fusioniert. Dadurch erhält man eine Position des Roboters (die sog. $FusedPos$), die der Position zu dem Zeitpunkt als das Bild gemacht wurde, entspricht. Daher wird diese Position dazu verwendet um die im Bild lokalisierten Objekte aus der lokalen Sicht in die globale herauszutransferieren. Im zweiten Schritt wird erneut ein Kalman-Filter benutzt, um aus den übriggebliebenen odometrischen Daten die aktuellste Position des Roboters zu berechnen (die sog. $OdoPos$). Diese Position ist sehr genau und wird zur Steuerung des Roboters auf einem exakten Pfad benutzt. Im nächsten Schritt wird dann aber wieder von der $FusedPos$ ausgegangen und unter Berücksichtigung der neuen odometrischen und visuellen Messwerten eine neue $FusedPos$ berechnet. Die aktuellsten odometrischen Messwete fließen dann in die Aktualisierung der $OdoPos$ ein. Bei der Berechnung der neuen $OdoPos$ benutzt das Kalman-Filter nicht die vorherige $OdoPos$, sondern die genauere $FusedPos$. Dieser Vorgang ist in 4.4 illustriert und wird später noch im Detail beschrieben werden.

Die zitierten Arbeiten haben alle gemeinsam, dass zu jedem Zeitpunkt genau ein aktueller Systemzustand, in unserem Fall eine Roboterposition, existiert. Mit der Zeit wird aus jedem Systemzustand ein neuer Systemzustand generiert und es entsteht eine Art Kette von Zuständen. Die Besonderheit an unserem Ansatz ist, dass stets zwei aktuelle Systemzustände ($FusedPos$, $OdoPos$) berechnet werden. Abhängig vom Kontext wird dann entschieden welcher Systemzustand benutzt werden sollte. Desweiteren wird ein Verfahren zur automatischen Bestimmung der Wahrnehmungsverzögerungen der Sensoren präsentiert.

Dieses Kapitel ist wie folgt strukturiert: Nachdem ein Gerüst von Begriffen etabliert ist wird der zwei-stufige Fusionsprozess im Detail erläutert. Anschließend wird eine Methode zur automatischen Kalibrierung und Synchronisierung vorgestellt. Schließlich werden einige Ergebnisse gezeigt. Eine Zusammenfassung und ein Ausblick auf mögliche Arbeiten in der Zukunft schließen das Kapitel ab.

4.2 Zeit und Ereignisse

Die folgenden Definitionen stammen aus dem Net-Lexikon [27].

Je nach Kontext werden unterschiedliche Nuancen des Begriffes *Ereignis* betont. Die folgende Definition enthält alle in unserem Zusammenhang relevanten Aspekte: “Ein Ereignis findet immer dann statt, wenn sich etwas verändert, wenn etwas passiert. In der Physik wird ein durch Ort und Zeit festgelegter Punkt der Raumzeit als Ereignis bezeichnet.“ Desweiteren unterscheiden wir zwischen *physikalischen Ereignissen* in der Welt, z.B. wenn das Rad eines Roboters anfängt sich zu drehen, und *mentalen Ereignissen* im Computer, z.B. wenn die odometrische Information, die das Drehen des Rades reflektiert, verfügbar wird. Die Zeit, die zwischen physikalischem und mentalem Ereignis vergeht, wird als *Wahrnehmungsverzögerung* bezeichnet. Die Zeit wird mit Hilfe von Intels Performance Counter [11] mit einer Genauigkeit von mehr als $10^{-6}s$ gemessen.

Die Wahrnehmungsverzögerungen der verschiedenen Sensoren ist unbekannt und nur schwer bestimmbar. In diesem Kapitel soll eine Möglichkeit zur Bestimmung der relativen Unterschiede zwischen den Wahrnehmungsverzögerung verschiedener Sensoren präsentiert werden. Dieser relative Unterschied ist hinreichend, um ein optimales Fusionsergbenis zu garantieren.

4.3 Kalman-Filter zur Fusion

Ein diskretes Kalman-Filter wird für die Fusion der Messwerte des visuellen und des odometrischen Sensors verwendet. Leser, die nicht mit der Theorie des Kalman-Filters vertraut sind seien auf die kurze Einführung im Anhang A, sowie auf [39] verwiesen.

Der Systemzustand x_k des Roboters zum Zeitpunkt t_k läßt sich durch einen sechsdimensionalen Vektor beschreiben:

$$x_k = \begin{pmatrix} x(t_k) \\ y(t_k) \\ \phi(t_k) \\ v_x(t_k) \\ v_y(t_k) \\ \omega(t_k) \end{pmatrix} \begin{array}{l} \text{Position in x-Richtung,} \\ \text{Position in y-Richtung,} \\ \text{Orientierung,} \\ \text{Geschwindigkeit in x-Richtung,} \\ \text{Geschwindigkeit in y-Richtung,} \\ \text{Drehgeschwindigkeit.} \end{array}$$

4.3.1 Messungen

Im Falle von diskreten Messungen ohne Verzögerung kann das Kalman-Filter direkt angewandt werden. In unserem System jedoch werden Messungen desselben Systemzustandes von Sensoren mit unterschiedlichen Verzögerungen vorgenommen.

Visueller Sensor

Der visuelle Sensor ist das in Kapitel 2 Abschnitt 2.1.1 beschriebene omnidirektionalen Vision System. Die Kamera kann wahlweise mit 30 oder 15 Bildern pro Sekunde betrieben werden und die typische Totzeit der Kamera (Verzögerung bis das Kamerabild verfügbar wird) beträgt ca. 70 ms.

In jedem Bild werden die Feldlinien detektiert und aus der relativen Veränderung dieser Linien in zwei aufeinander folgenden Bildern wird die relative Bewegung des Roboters $\Delta x_v(t_k)$, $\Delta y_v(t_k)$ und die Rotation $\Delta\phi_v(t_k)$ berechnet.

Odometrischer Sensor

An jedes der drei Räder des omnidirektionalen Antriebes ist ein Encoder montiert, der die Drehung mißt. Etwa alle 10 ms wird eine neue Messung vorgenommen. Aus dem Messergebnis werden die relativen Bewegungen $\Delta x_o(t_k)$, $\Delta y_o(t_k)$ und die Rotation $\Delta\phi_o(t_k)$ des Roboters zwischen dem Zeitpunkt t_k der aktuellen Messung und t_{k-1} der letzten Messung berechnet.

An dieser Stelle sind die Gleichungen zur Berechnung der realtiven Verschiebung und Drehung aus den einzelnen Messwerten irrelevant. Es sei nur festgehalten, dass durch eine einfache Matrixmultiplikation aus den Drehungen der einzelnen Räder die Geschwindigkeiten berechnet werden können. Desweiteren sollte festgehalten werden, dass das hier vorgestellte Verfahren unabhängig vom Fahrwerk ist. Es kann zwei, drei oder vier Räder haben, relevant ist nur die Verschiebung und Drehung zwischen zwei Messungen.

4.3.2 Korrespondenz der odometrischen und visuellen Information

Um das hier beschriebene Verfahren zur Fusion von odometrischer und visueller Information anzuwenden, muss exakt festgestellt werden können, welche Messungen sich auf den selben Systemzustand in der Zeit beziehen. In diesem Abschnitt soll der zeitliche Zusammenhang zwischen Ereignis in der Welt, Messung und dem Eintreffen der Ergebnisse der Messungen der verschiedenen Sensoren erläutert werden.

Angenommen, die Kamera wird mit einer Bildrate von 30 Bildern pro Sekunde betrieben, dann erhält das System etwa alle 33 ms ein neues Bild. Dagegen gibt es etwa alle 10 ms neue odometrische Messwerte, so dass es mehr odometrische als visuelle Messwerte gibt. Zusätzlich haben die beiden Sensoren unterschiedliche Verzögerungszeiten. Da die visuelle Verzögerung deutlich größer als die odometrische ist, entspricht die zeitliche Reihenfolge der eintreffenden Messergebnisse nicht der zeitlichen Reihenfolge der zugrundeliegenden Systemzustände.

Ein physikalisches Ereignis (z.B. ein Photon, das vom Ball reflektiert wird) ereignet sich in der realen Welt zu einem bestimmten Zeitpunkt I, der dem System unbekannt ist. Jedes Bild wird mit einem Zeitstempel J versehen, jedoch tritt zwischen I und J eine Verzögerung ΔI auf, diese wird als die Wahrnehmungsverzögerung des visuellen Sensors bezeichnet, siehe dazu auch Abbildung 4.2.

Immer wenn der Roboter sich bewegt, wird diese Bewegung durch den Odometer registriert. Diese Information wird an das System weitergegeben, wobei eine kurze Wahrnehmungsverzögerung ΔO zwischen dem Ereignis und dessen Wahrnehmung durch das System auftritt.

Die Wahrnehmungsverzögerungen ΔI und ΔO sind von großer Bedeutung, da sie die Reihenfolge, in der physikalische Ereignisse wahrgenommen werden, verändern können. Wenn Informationen von verschiedenen Sensoren fusioniert werden sollen, dann dürfen nur zeitlich zu einander passende Messwerte miteinander verrechnet werden. Da die odometrische Verzögerung kürzer als die

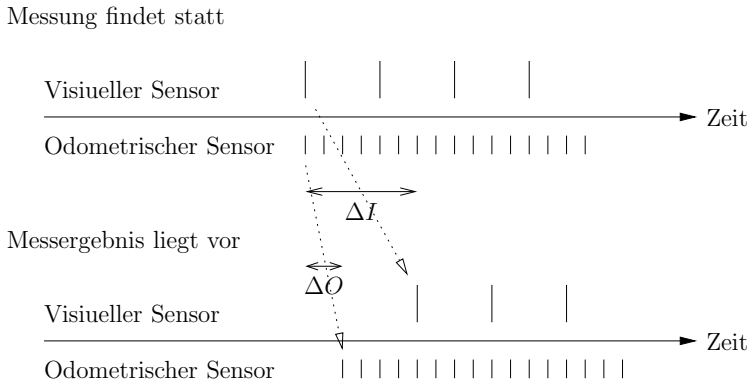


Abbildung 4.2: Die Wahrnehmungsverzögerung ΔI und ΔO. Die Messergebnisse von Messungen, die gleichzeitig stattfinden, liegen zu unterschiedlichen Zeitpunkten vor. Dagegen stammen zwei Messergebnisse, die zur gleichen Zeit fertiggestellt werden, nicht vom selben Systemzustand.

visuelle ist, wird es immer aktuellere odometrische Informationen geben, die jedoch keineswegs für die Fusion verwandt werden darf.

4.3.3 Datenfusion

Um ein diskretes Kalman-Filter zur Fusion von Daten, die von verschiedenen Sensoren stammen, zu benutzen, muss sichergestellt sein, dass die Daten vom selben Systemzustand in der Zeit stammen. Wegen der verschiedenenen Verzögerungszeiten der Sensoren ist das im Allgemeinen nicht der Fall.

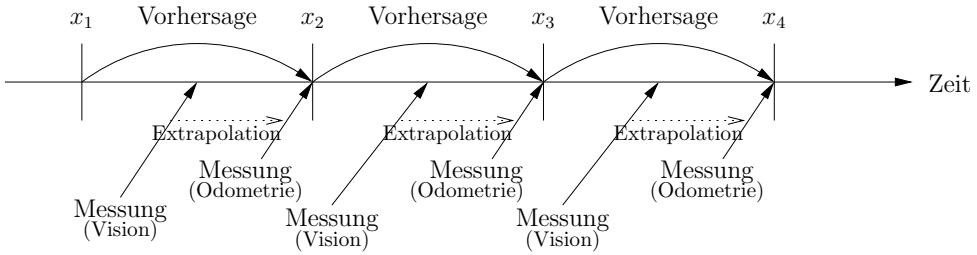


Abbildung 4.3: Larsen et al. extrapolieren die verzögerten Messwerte des visuellen Sensors, damit sie sich auf den selben Systemzustand beziehen, wie die Messwerte des odometrischen Sensors. Ein Kalman-Filter fusioniert anschließend diese Messung mit der Vorhersage.

Das einzige uns bekannte Verfahren zur Fusion von Messwerten, die mit verschiedenen Verzögerungen behaftet sind, wird von Larsen et al. in [20] dargelegt. Die Idee dabei ist es, diejenigen Messwerte, die mit einer längeren Verzögerung behaftet sind, zu extrapolieren (siehe Abbildung 4.3). Larsen

et al. argumentieren, dass ihre Methode "statistisch gesehen optimal zur Fusion extrapolierter Messwerte" sei, "aber die Methode der Extrapolation *selbst* nicht optimal" sei. Das Problem ist, dass die extrapolierten Messwerte nur eine Schätzung sind. Unserer Meinung nach ist es daher sinnvoller nur die vorhandenen Messwerte zu benutzen.

Wir schlagen einen zwei-stufigen Fusionsprozess vor. Dabei werden die odometrischen Daten in zwei Teile aufgespalten: einen Teil, zu dem es korrespondierende visuelle Daten gibt und einen Teil, zu dem es keine gibt. Wie diese Aufspaltung erreicht wird, wird im nächsten Abschnitt erläutert.

Im ersten Schritt werden die zusammengehörenden Daten mit einem Kalman-Filter fusioniert. Dadurch erhält man eine Position des Roboters, die der Position zu dem Zeitpunkt, als das Bild gemacht wurde, entspricht ($FusedPos$). Daher wird diese Position dazu benutzt um die im Bild lokalisierten Objekte aus der lokalen Sicht in die globale herauszutransferieren. Die allgemein bekannten Kalman-Filter Gleichungen (siehe Anhang A) können direkt angewandt werden. Dabei besteht die Messung z_k aus den relativen Bewegungen in x- und y-Richtung und der Änderung der Orientierung seit der letzten Anwendung des Kalman-Filters. Typischerweise wird das Kalman-Filter immer dann angewandt, wenn ein neues Bild ankommt. Die odometrischen Messungen, die in dieser Zeitperiode gemacht worden sind, werden akkumuliert:

$$z_k = (\Delta x_o(t_k), \Delta y_o(t_k), \Delta \phi_o(t_k), \Delta x_v(t_k), \Delta y_v(t_k), \Delta \phi_v(t_k)).$$

Die so berechnete Position des Roboters ist jedoch veraltet und muss noch aktualisiert werden. Im zweiten Schritt wird erneut ein Kalman-Filters benutzt um aus den übriggebliebenen odometrischen Daten die aktuellste Position des Roboters zu berechnen ($OdoPos$). Diese Position ist sehr genau und wird zur Steuerung des Roboters auf einem exakten Pfad benutzt. Die Messung z_k^+ besteht in diesem nur aus den odometrischen Daten: $z_k^+ = (\Delta x_o^+(t_k), \Delta y_o^+(t_k), \Delta \phi_o^+(t_k))$.

Dieser Vorgang ist in Bild 4.4 illustriert.

In der nächsten Iteration wird derselbe zwei-stufige Fusionsprozess benutzt. Dabei geht der erste Schritt aber wieder von der $FusedPos$ aus und berechnet unter Berücksichtigung der neuen odometrischen und visuellen Messwerte eine neue $FusedPos$. Die aktuellsten odometrischen Messwerte fließen dann in die Aktualisierung der $OdoPos$ ein. Bei der Berechnung der neuen $OdoPos$ benutzt das Kalman-Filter nicht die vorherige $OdoPos$, sondern die genauere $FusedPos$.

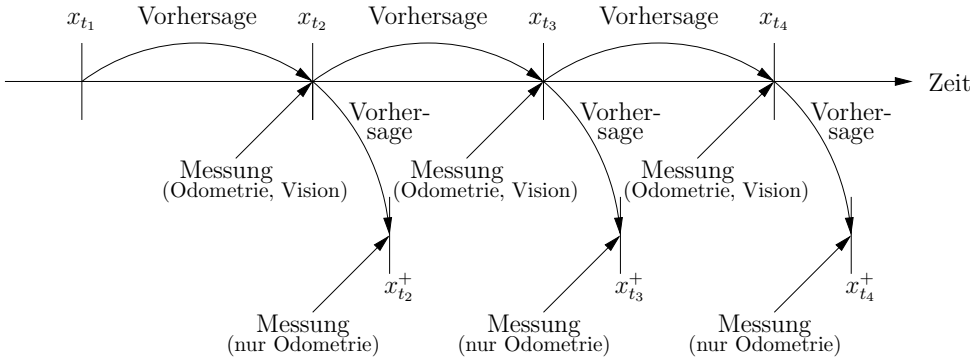


Abbildung 4.4: Der zwei-stufige Fusionsprozess: Zuerst werden die zeitlich korrespondierenden visuellen und odometrischen Messwerte mit Hilfe eines Kalman-Filters fusioniert. Im zweiten Schritt werden die übriggebliebenen odometrischen Messwerte zur Aktualisierung der Roboterposition benutzt.

4.4 Automatische Kalibrierung und Synchronisierung

Unabhängig vom Fusionsverfahren sollten stets nur zeitlich zu einander korrespondierende Messwerte fusioniert werden. Die Aufgabe des im Folgenden präsentierten Verfahren ist die Bestimmung des relativen Unterschiedes in den Wahrnehmungsverzögerungen der beiden Sensoren. Nur wenn diese Differenz bekannt ist, kann bestimmt werden, welche odometrischen und welche visuellen Daten zueinander gehören, d.h. vom gleichen Systemzustand in der Zeit stammen.

Die Idee bei der Synchronisierung ist, denselben Systemzustand mit beiden Sensoren zu messen. Zum Beispiel kann die Änderung des Orientierungswinkels mit dem Odometer und mit der Kamera bestimmt werden. Wenn man die Messwerte der beiden Sensoren vergleicht, kann dann aus deren Unterschied die Differenz in den Wahrnehmungsverzögerungen bestimmt werden, da bekannt ist, dass beide Sensoren denselben Systemzustand messen.

Die Änderung des Orientierungswinkels kann direkt aus den odometrischen Daten berechnet werden. Um sie mit der Kamera zu messen, muss ein Referenzpunkt ausgewählt und verfolgt werden. Der Einfachheit halber kann der Ball als Referenzobjekt benutzt werden. Da die ganze Prozedur nur wenige Sekunden dauert, ist es keine große Einschränkung, dass der Ball während der Prozedur ruhen muss. Die Prozedur läuft wie folgt ab: Man positioniert den Ball nahe am Roboter und schickt dann Befehle zum Roboter, dass er etwa 180 Grad um die eigene Achse rotieren und dann wieder zurückdrehen soll. Während dieser Rotation berechnet man aus den odometrischen Daten die akkumulierte Änderung des Orientierungswinkels $\phi(t_k)$ nach jedem neuen

Messwert zum Zeitpunkt t_k. In jedem Bild wird die Position des Bildes bestimmt und ebenfalls die akkumulierte Änderung des Winkels $\psi(t_k)$ zum Ball berechnet. Wichtig ist dabei, dass dieser Winkel direkt aus dem Bild bestimmt werden kann und nicht von der Lokalisierung des Roboters abhängt.

Die Messwerte werden in zwei Listen gespeichert und mit Zeitstempeln markiert. Diese Zeitstempel beziehen sich auf den Zeitpunkt, zu dem das Messergebnis vorlag, d.h. zwei Messwerte von verschiedenen Sensoren mit dem selben Zeitstempel können von zwei unterschiedlichen Systemzuständen stammen. Die Zeitstempel müssen korrigiert werden, so dass zeitlich korrespondierende Daten auch den selben Zeitstempel tragen. Die nächsten beiden Abschnitte werden erklären wie diese automatische Kalibrierung und Synchronisierung realisiert werden kann.

4.4.1 Automatische Kalibrierung

Da die Räder während der Bewegung keine optimale Haftung haben, neigen sie dazu leicht durchzudrehen. Das hat zur Folge, dass die vom Odometer gemessene Veränderung stets größer als die tatsächliche ist. In diesem Fall würde eine zu starke Rotation des Roboters gemessen werden. Die tatsächliche Rotation kann jedoch aus den visuellen Messungen bestimmt werden. Experimente haben gezeigt, dass das Verhältnis zwischen gemessener und tatsächlicher Rotation nahezu konstant ist. Die Konstante

$$\rho = \frac{max(\phi(t))}{max(\psi(t))}$$

wird daher als Rotationskorrekturfaktor verwandt. Zur Korrektur werden daher die vom Odometer gelieferten Messwerte mit dieser Konstanten multipliziert, bevor die Rotation des Roboters berechnet wird (Analog werden auch Seitwärts- und Vorwärtskorrekturkonstanten verwandt, welche jedoch auf andere Weise bestimmt werden).

4.4.2 Automatische Synchronisierung

Eine genaue Synchronisierung ist eine notwendige Voraussetzung für das benutzte Fusionsverfahren. Die Verschiebung zwischen den gemessenen Winkeln läßt sich auf die längere Wahrnehmungsverzögerung des visuellen Sensors zurückführen und entspricht ΔT in Bild 4.2.

ΔT ist auch diejenige Verschiebung der einen Kurve, durch die, die Fläche zwischen den beiden Kurven minimiert wird. Im Sinne des kleinsten quadratischen Fehlers erhält man:

$$\Delta T = \underset{\Delta t}{\operatorname{argmin}} \left(\sum_{T_0+\Delta t \leq t \leq T_n} \left(\phi\left(t_k\right) - \psi\left(t_k - \Delta t\right) \right)^2 \right),$$

wobei T_0 und T_n die Anfangs- bzw. Endzeit der Kalibrierungsprozedur sind. Diese Gleichung kann für Δt durch Gradientenabstieg gelöst werden. Da es sich um diskrete Messwerte handelt und die Zeitintervalle zwischen zwei Messwerten schwanken können, muss an den Stellen, an denen kein Messwert vorhanden ist, linear interpoliert werden. Dadurch dass die Messwerte sehr nah aneinander sind und die zugrunde liegenden Kurven stetig sind, ist der dabei entstehende Fehler vernachlässigbar klein.

ΔT ist also die Differenz in den Wahrnehmungsverzögerungen der beiden Sensoren. Wie bereits erwähnt werden alle hereinkommenden Messwerte mit einem Zeitstempel versehen. Dieser Zeitstempel wird benutzt, um die zeitliche Korrespondenz zwischen den Messwerten des Odometers und der Kamera festzustellen. Da die Kamera eine um ΔT längere Wahrnehmungsverzögerung hat, werden die Zeitstempel der visuellen Messwerte durch die Subtraktion der Zeit ΔT korrigiert.

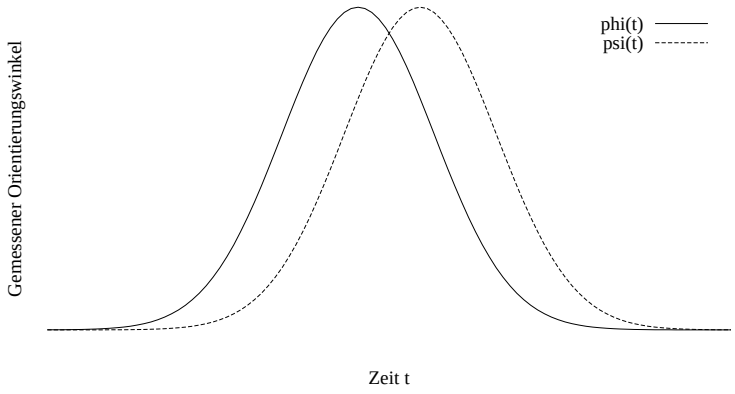
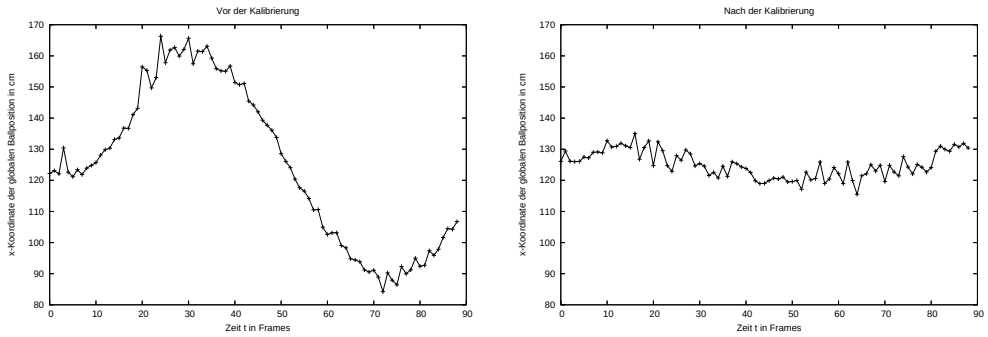


Abbildung 4.5: Der vom Odometer gemessene Orientierungswinkel $\phi(t)$ und der aus den visuellen Messwerten bestimmte akkumulierte relative Winkel zum Ball $\psi(t)$ (idealisierte und übertrieben dargestellte Kurven).

4.5 Ergebnisse

Die Verbesserung, die durch die automatische Synchronisierung erreicht werden kann, soll durch einen vorher-nachher Vergleich verdeutlicht werden. In Abbildung 4.6 ist die globale Ballposition eines ruhenden Balles, während sich der

Roboter dreht, dargestellt (exemplarisch ist nur die x-Koordinate in Abhängigkeit von der Zeit aufgetragen). Wenn der Roboter anfängt zu rotieren, dann wird das Odometer eine Rotation zu einem Zeitpunkt anzeigen, zu dem das Bild noch unverändert ist. Wenn die Sensoren nicht synchronisiert sind, wird bei der Berechnung der Roboterposition diese neue odometrische Information berücksichtigt werden, so dass sich die Roboterposition ändern wird. Diese Roboterposition wird dann zum heraustransferieren der im Bild gefundenen Objekte benutzt, was zur Folge hat, dass sich der Ball mit dem Roboter mitbewegt, wie in (a) zu sehen. Nach der Synchronisierung wird zu Anfang der Rotation die aktuelle Roboterposition anhand der odometrischen Daten aktualisiert werden, aber zum Heraustransferieren der Objekte im Bild wird die zu den visuellen Daten korrespondierende Roboterposition benutzt, so dass der Ball ruhig liegen bleibt, siehe (b).



(a) (b)

Abbildung 4.6: Globale Ballposition (x-Koordinate) während der Drehung des Roboters: (a) vor der Synchronisierung, (b) danach.

Anmerkung: Bei der Berechnung der globalen Ballposition wird der im Bild bestimmte Ballpunkt aus dem lokalen Koordinatensystem des Roboters in das globale Koordinatensystem heraustransferiert. Dabei wird natürlich die Roboterposition benutzt, so dass sich ein Fehler in der Roboterposition auch auf die Ballposition auswirkt. Wenn sich der Roboter bewegt, wackelt die Kamera und Fehler im Bereich von $20cm$ sind möglich. Daher ist selbst in (b) die Ballposition nicht absolut ruhig.

Um die Zuverlässigkeit des Verfahrens zu überprüfen wurde die automatische Kalibrierung mehrmals hintereinander ausgeführt. Dabei wurden die folgenden Differenzen Δt in den Wahrnehmungsverzögerungen der beiden Sensoren gemessen:

Wiederholung	Δt
1	56
2	56
3	57
4	56
5	58
.	...
25	57
Durchschnitt	56,6

Wie die Kalibrierungsergebnisse zeigen, läßt sich die Differenz der Wahrnehmungsverzögerungen zuverlässig und ohne größe Schwankungen bestimmen. Daher werden die Roboter mit dem Wert $\Delta t = 56,6$ ms initialisiert, vor jedem Spiel jedoch werden die Sensoren noch ein Mal kalibriert und synchronisiert.

Kapitel 5

Verhalten

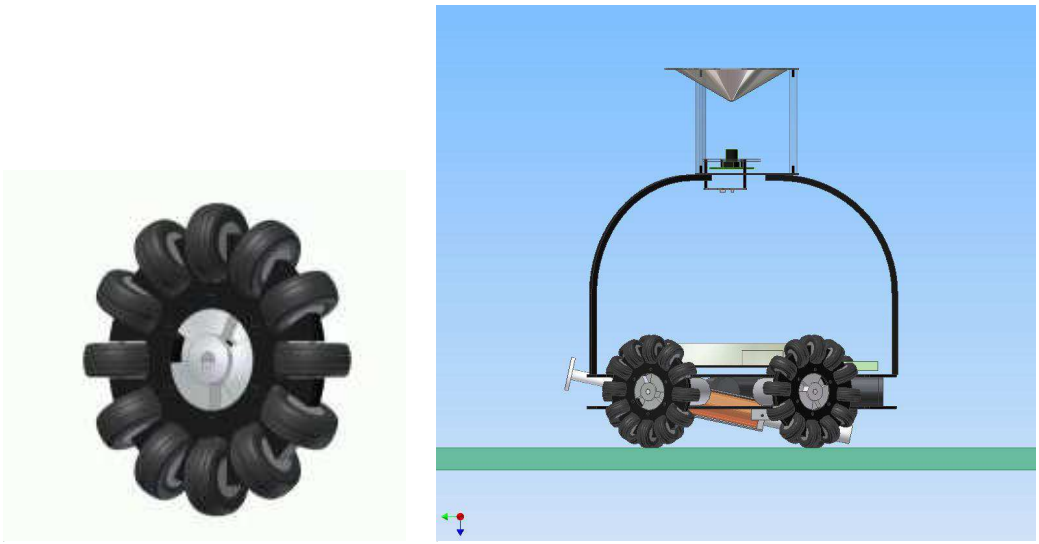
Das Verhalten basiert auf einem System, das schon seit Langem von den FU-Fighters verwandt wird und ursprünglich von Sven Behnke entwickelt wurde [4]. Dabei wird eine hierarchische Architektur zur Steuerung benutzt. Einfache und schnelle Verhalten befinden sich auf den unteren Stufen der Hierarchie, während langsamere und komplexere Verhalten auf den höheren Ebenen angesiedelt sind. Die Verhalten können unabhängig sein, es können aber Hemmungs- und aktivierungsbeziehunegn zwischen ihnen existieren, ähnlich wie in [6] von R. A. Brooks vorgeschlagen.

Jede Ebene ist aus drei separaten Modulen aufgebaut. Erstens, einem Sensoren-Modul, welches die sich schnell ändernden Werte der Sensoren aggregiert. Zweitens, einem Aktivierungs-Modul, das an Hand der Sensoren entscheidet, ob bestimmte Verhalten (die sog. Aktoren) aktiviert werden sollen und drittens dem Aktoren-Modul, welches das aktuelle Verhalten bestimmt, so dass es der berechneten Ziel-Dynamik entspricht.

Im Folgenden soll kurz die relevante Hardware erläutert werden. Anschließend werden einige grundlegende Verhaltensmuster skizziert.

5.1 Omnidirektionales Fahrwerk

Das omnidirektionale Fahrwerk besteht aus drei speziell gefertigten Rädern und erlaubt es in jede beliebige Richtung zu fahren ohne dabei die Orientierung des Roboters ändern zu müssen. Anstelle einer symmetrischen Anordnung der Räder beträgt der Winkel zwischen den beiden vorderen Rädern 150 Grad und ermöglicht eine höhere Geschwindigkeit in der vorwärts und rückwärts Bewegung. Es werden Räder aus Eigenproduktion benutzt, wobei jedes Rad aus zwölf 38mm Lufträder besteht, die normalerweise im Flugzeugmodellbau eingesetzt werden. Diese Räder haben einen relativ großen Durchmesser und bieten dadurch mehr Haftung.



(a) Ein Rad (b) Seitenansicht des Roboters im Modell

Abbildung 5.1: Die Hardware.

5.2 Verhaltensmuster für den Torwart

Wie weiter oben erklärt, gibt es zwischen den verschiedenen Verhaltensmustern eine Hierarchie. Beim Torwart sind die Verhalten alle auf derselben Ebene, da es keine komplexen, strategischen Gesichtspunkte zu beachten gibt. Meist ist das “Torwart: Stellen“ Verhalten aktiv, wenn sich der Ball im eigenen Strafraum befindet, wird es durch das Verhalten “Torwart: Ball-Aus-Strafraum“ gehemmt. Schließlich wenn sich der Ball aufs eigene Tor zubewegt wird das Verhalten “Torwart: Ball-Halten“ aktiv, welches wiederum die beiden anderen Torwart Verhalten hemmt.

Die Steuerung des Torwarts wird dadurch erleichtert, dass andere Roboter nur kurzzeitig in den Strafraum hineinfahren dürfen und dabei den Torwart nicht berühren dürfen. Daher braucht der Torwart keine speziellen Verhalten zur Hindernisvermeidung.

5.2.1 Torwart: Stellen

Das Verhaltensmuster “Torwart: Stellen“ ist immer dann aktiv, wenn kein anderes Torwart-Verhalten aktiv ist. Das Ziel dieses Verhaltens ist es, den Torwart so zu positionieren, dass es der gegnerischen Mannschaft erschwert wird ein Tor durch einen Fernschuss zu erzielen. Die Position wird dabei abhängig

von der Ballposition bestimmt. Wenn der Ball frontal vor dem Tor liegt, stellt sich der Torwart auf Höhe des Balles kurz vor der Torlinie. Ist der Ball dagegen seitlich zum Tor, ist es wichtig die sog. kurze Ecke zu decken. Dazu stellt sich der Torwart so nah an den zum Ball näheren Pfosten, dass kein Ball zwischen Pfosten und Torwart durchpasst. Der Torwart richtet sich dabei stets zum Ball aus.

Im Allgemeinen bewegt sich der Torwart auf einer Ellipse, wie in Abbildung 5.2 dargestellt. Die Haupt- und Nebenachse dieser Ellipse, sowie die Verschiebung nach vorne sind als editierbare Sensoren definiert und können daher im laufenden Betrieb geändert werden.

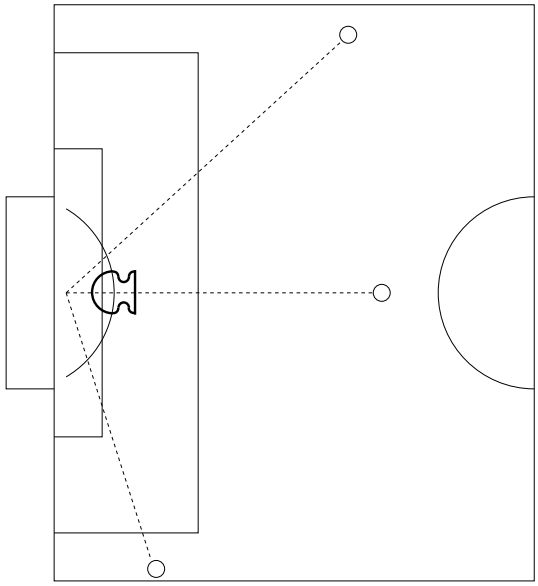


Abbildung 5.2: Das Verhaltensmuster “Torwart: Stellen“.

Ein Sonderfall tritt dann auf, wenn der Ball innerhalb der eingezeichneten Ellipse ist, aber noch nicht über die Torlinie gerollt ist. Es ist dann große Vorsicht geboten, um kein Eigentor zu erzielen. Der Torwart versucht in einer solchen Situation hinter den Ball zu fahren, wobei stets ein Mindestabstand gewahrt wird. Sobald der Torwart hinter dem Ball ist, schiebt er den Ball aus dem Torraum heraus.

5.2.2 Torwart: Ball-Halten

In jeder Iteration wird aus der aktuellen Ballposition und der Ballgeschwindigkeit ein möglicher Schnittpunkt der Balltrajektorie mit den äußeren Feldlinien berechnet. Wenn dieser Schnittpunkt im Tor oder knapp daneben liegt, bewegt sich der Ball in Richtung Tor und es ist Vorsicht geboten. Wenn der Ball

zusätzlich eine maximale Entfernung zum Tor nicht überschreitet und eine Mindestgeschwindigkeit hat wird das Verhalten "Torwart: Ball-Halten" aktiviert. Diese beiden Bedingungen sollen sicherstellen, dass der Torwart nicht unnötiger Weise sich in die Ecke bewegt, nur weil der Ball eine zufällige BewegungsÄnderung erfährt. Um den Ball zu halten, bewegt sich der Torwart auf einer Linie die meist 50 cm vor der Torlinie ist (der Abstand zur Torlinie ist über einen Sensor zur Laufzeit veränderbar).

5.2.3 Torwart: Ball-Aus-Strafraum

Obwohl es Aufgabe der Abwehrspieler ist den Ball nicht in den eigenen Strafraum rollen zu lassen, kann es vorkommen, dass der Ball in den eigenen Strafraum rollt und dort liegen bleibt. In einer solchen Situation muss der Torwart entschlossen zum Ball gehen und diesen aus dem Strafraum schieben.

5.2.4 Elfmeter-Halten

Das "Elfmeter-Halten" Verhalten basiert auf dieselbe Idee wie das "Torwart: Ball-Halten" Verhalten. Es kann nur aktiviert werden, wenn der Roboter in einem bestimmten Modus ist. Dieser Modus wird vom Zentralrechner gesetzt. Anfangs befindet sich der Torwart auf der Torlinie. Sobald der Ball losrollt darf sich der Torwart bewegen. Aus der Ballposition und der Ballgeschwindigkeit wird eine Abfangsposition auf der Haltelinie (50 cm vor der Torlinie) bestimmt und der Roboter versucht so schnell wie möglich an diesen Punkt zu gelangen.

5.3 Ergebnisse

Das Zusammenwirken der in diese Diplomarbeit entwickelten Komponenten kann am Beispiel des Torwart-Verhaltens gut verdeutlicht werden. Dank der verbesserten Balllokalisierung und -geschwindigkeitsbestimmung kann der Auftreffpunkt des Balles im Tor frühzeitig erkannt werden. Weiterhin führt die Sensorfusion zu einer genaueren Positionsbestimmung des Roboters und des Balles. Dadurch ist der Torwart in der Lage sich gut im Tor zu positionieren und die meisten Schüsse auf das Tor abzuwehren. Ein Video, welches den Torwart in Aktion zeigt, kann unter *http://www.petrovi.de/slav/diplom/* angeschaut werden.

Wie man im Video sieht, reagiert der Torwart angemessen schnell und fängt den Ball meist zügig ab. Das Hauptproblem bei Schüssen mit großer Geschwindigkeit ist die Wahrnehmungsverzögerung, bedingt durch die Totzeit der Kamera: der Ball ist meist schon im Tor, wenn die Bilder anfangen eine

Bewegung in Richtung Tor anzuzeigen. Wegen der zusätzlichen Trägheit des Roboters werden daher selbst durch weitere Optimierung und Beschleunigung der Algorithmen solche Schüsse nicht haltbar werden. Die von uns angestrebte Lösung zielt darauf ab, ein Teamverhalten zu entwickelt, welches die Abwehrspieler mit einbezieht. Je ein Abwehrspieler deckt eine der Ecken ab, so dass der Torwart nur Schüsse im mittleren Bereich des Tores abwehren muss. Diese Strategie hat sich als sehr erfolgreich erwiesen.

Kapitel 6

Schlussbemerkungen

Um erfolgreich am RoboCup Wettbewerb teilzunehmen müssen Probleme aus verschiedenen Bereichen gelöst werden. Einerseits gilt es bezüglich der Hardware viele technische Probleme zu meistern, diese sind jedoch nicht Teil dieser Diplomarbeit. Vielmehr wurden vielfältige in Software zu realisierende Herausforderungen angegangen und zufriedenstellend gelöst. Die meisten Algorithmen bauen auf die von einer omnidirektionalen Kamera gelieferten Bilder auf. Die Qualität dieser Bilder wird stark durch die Einstellungen der Kamera beeinflußt. Daher wurde ein Weißabgleich entwickelt, der speziell für Bilder von einer omnidirektionalen Kamera optimiert ist und eine weitgehende Farbkonstanz erzielt. Anstelle der oft benutzten "gray world assumption" wird mit einer Weißreferenz gearbeitet, die in der Form eines weißen Ringes um die Kameralinse realisiert ist. Das Bild wird von der Kamera im YUV geliefert und braucht nicht in ein anderes Format umgewandelt werden. Es wird einfach der Mittelwert für Y, U, und V über alle Pixel berechnet, die sich innerhalb der Weißreferenz befinden. Mit Hilfe von zwei separaten PID-Reglern werden die Verstärkungsfunktionen für U und V für die verwendete Kamera bestimmt. Außerdem wird der mittlere Wert von Y verwendet, um die Helligkeit der Kamera möglichst konstant zu halten.

Es wurde gezeigt wie anhand von Farbinformation und Wissen über die Ballform die Ballposition sehr genau und zuverlässig bestimmt werden kann. Dazu wird das mit einem Gauß-Filter geglättete Bild in den HSI-Farbraum konvertiert und für jeden Pixel der Farbabstand zur Referenzfarbe rot berechnet. Ein adaptives Binarisierungverfahren benutzt dann Information über die zu erwartende Ballgröße und das Histogramm der Farbabstände, um einen geeigneten Schwellwert zu bestimmen. Eine Ellipse wird um die auf diese Weise erhaltene Silhouette gelegt, um die genaue Ballposition zu erhalten. Neben der Ballposition ist auch die Abschätzung der Ballgeschwindigkeit von großer Bedeutung. Der hier vorgestellte Algorithmus benutzt dazu ein Kalman-Filter

und kann im Falle, dass der Ball verdeckt ist, aber frei rollt, zuverlässig zur Ballvorhersage verwandt werden.

Einen möglichen Ansatzpunkt für Erweiterungen bietet das dem Kalman-Filter zugrunde liegende Bewegungsmodell. Zur Zeit wird angenommen, dass der Ball frei rollt. Viele Roboter sind jedoch inzwischen in der Lage den Ball zu führen und zu schießen. Daher kann es sich als sinnvoll erweisen, die beim Dribbeln und beim Schießen auf den Ball wirkende Kraft als Eingangsgrößen einfließen zu lassen. Da der Kontakt mit dem Ball wegen der schwachen Kameraauflösung nicht genau bestimmt werden kann, wurde jedoch vorerst auf diese Erweiterung verzichtet.

Die Beobachtungen der einzelnen Roboter bezüglich des Balles werden als Gauß-Verteilungen modelliert und können von einem zentralen Rechner zu einem globalen Weltbild fusioniert werden, wobei die Genauigkeit der Messungen erhöht wird. Sobald Hindernisse auf dem Spielfeld (also Roboter der eigenen und der gegnerischen Mannschaft) zuverlässig erkannt und verfolgt werden können, kann das für den Ball benutzte Fusionsverfahren auf die Hindernisse ausgeweitet werden. Dadurch wäre ein vollständiges globales Weltbild konstruierbar und eine exakte Pfadplanung möglich. Bevor die Beobachtungen fusioniert werden können, müsste entschieden werden, welche Beobachtungen sich auf dasselbe Objekt beziehen. Ein nicht triviales Problem, das in Zukunft angegangen werden sollte.

Das Wissen über den Bewegungszustand der gegnerischen Roboter ist sehr wichtig für die eigene Verhaltenssteuerung. Daher sollte jedes detektierte Hindernis mit einem Kalman-Filter assoziiert werden. Das Kalman-Filter liefert genauere Information über die Position und die Bewegungsrichtung der Objekte. Von diesem Informationsgewinn würde in erster Linie die Pfadplanung profitieren: Beim Planen eine Pfades könnten dann nicht nur die aktuellen Hindernispositionen berücksichtigt werden, sondern auch deren aktuelle Geschwindigkeit. Dadurch könnte die Hindernisposition zu einem späteren Zeitpunkt vorhergesagt werden und Kollisionen würden vermieden.

Viele robotische Systeme kombinieren Vision und Odometrie, um eine bessere Lokalisierung zu erreichen. Während Vision allein eine relativ lange Verzögerung hat und Ergebnisse mit starkem Rauschen liefert, führt eine Lokalisierung nur anhand des Odometers zu einem unbeschränkten Positionsfehler. Zur Fusion der Daten wird ein diskretes Kalman-Filter verwandt und es wurde eine neuartige Methode zur Behandlung von Daten mit unterschiedlichen Wahrnehmungsverzögerungen vorgestellt. Wir haben einen zwei-stufigen Fusionsprozess entwickelt, bei welchem die odometrischen Daten in zwei Teile aufgespalten werden: einen Teil, zu dem es korrespondierende visuelle Daten gibt und einen Teil, zu dem es keine gibt. Im ersten Schritt werden die zusammengehörenden Daten mit einem Kalman-Filter fusioniert. Dadurch erhält

man eine Position des Roboters, die der Position zu dem Zeitpunkt als das Bild gemacht wurde, entspricht. Daher wird diese Position dazu benutzt um die im Bild lokalisierten Objekte aus der lokalen Sicht in die globale herauszutransferieren. Im zweiten Schritt wird erneut ein Kalman-Filter benutzt, um aus den übriggebliebenen odometrischen Daten die aktuellste Position des Roboters zu berechnen. Diese Position ist sehr genau und wird zur Steuerung des Roboters auf einem exakten Pfad benutzt. Unabhängig vom Fusionsverfahren sollten nur zeitlich zu einander korrespondierende Messwerte fusioniert werden. Daher wurde ein automatisches Synchronisierungsverfahren zur exakten Bestimmung der Differenz der Wahrnehmungsverzögerungen der Sensoren entwickelt.

Das Zusammenwirken dieser einzelnen Elemente wurde am Beispiel des Torwart-Verhaltens verdeutlicht. Dank der verbesserten Balllokalisierung und -geschwindigkeitsbestimmung kann der Auftreffpunkt des Balles im Tor frühzeitig erkannt werden. Weiterhin führt die Sensorfusion zu einer genaueren Positionsbestimmung des Roboters und des Balles. Dadurch ist der Torwart in der Lage die meisten Schüsse auf das Tor abwehren.

Wie diese Arbeit zeigt, lassen sich einige der wichtigsten Probleme beim Roboter-Fußball inzwischen zufriedenstellend lösen. Die Roboter lokalisieren sich selbst und den Ball nahezu perfekt und zeigen einfache Formen von sinnvollem Mannschaftsverhalten. Es ist jedoch noch ein langer Weg zu gehen, bevor auch nur daran gedacht werden kann, gegen eine menschliche Mannschaft anzutreten. Allerdings ist bis zum Jahre 2050 auch noch viel Zeit...

Anhang A

Das diskrete Kalman-Filter

Das Kalman-Filter ist ein Satz von mathematischen Gleichungen zur effizienten (rekursiven) Lösung von Problemen, die nach der Methode des kleinsten quadratischen Fehlers berechnet werden können. Das Filter ist sehr mächtig, da es die Abschätzung von Zuständen in der Vergangenheit, Gegenwart und Zukunft erlaubt und das ohne die genaue Kenntnis des modellierten Systems vorauszusetzen.

Das Kalman-Filter ist benannt nach Rudoplh E. Kalman, der es in seiner 1960 publizierten Veröffentlichung [19] einer rekursiven Lösung zur linearen Filterung diskreter Daten erstmalig beschreibt.

A.1 Begriffe

Die Definition einiger Begriffe ist zum Verständnis des Kalman-Filters von großer Bedeutung.

Dynamisches System

In seiner allgemeinsten Bedeutung bezieht sich der Begriff *System* auf eine physikalische Einheit, auf welche eine Aktion ausgeübt wird. Die Aktion hat dabei eine Eingabe u_k, das System reagiert darauf und produziert eine Ausgabe y_k. Bei einem *dynamischen* System ereignen sich diese Phänomene in einer zeitlichen Reihenfolge.

Ein einfaches Beispiel wäre ein elektrischer Schaltkreis. Die Eingabe ist die Spannung, die an einen Zweig angelegt wird und die Ausgabe ist die Spannung zwischen zwei Knoten.

Es wird weiterhin zwischen *diskreten* und *kontinuierlichen* Systemen unterschieden, abhängig davon ob die Zeit in diskreten Zeitschritten oder stetig fortschreitet.

Systemzustand

Bei der Modellierung eines dynamischen Systems müssen die Eingaben (Ursachen) mit den Ausgaben (Effekten) korreliert werden. Die Ausgabe y_k zu einem Zeitpunkt kann dabei nicht nur aus der Eingabe u_k zum gleichen Zeitpunkt bestimmt werden, sie ist vielmehr das Resultat aller Eingaben aus der Vergangenheit des Systems. Der Systemzustand x_k fasst diese Information aus der Vergangenheit und der Gegenwart des Systems zusammen. Die Kenntnis des Systemszustandes x_k und der Eingabe u_k ermöglicht die Berechnung der Ausgabe y_k.

In einem dynamischen System beeinflusst also die Eingabe den Systemzustand, während die Ausgabe eine Funktion des Systemzustandes ist.

A.2 Gleichungen

Die Gleichungen des Kalman-Filters lassen sich in zwei Gruppen aufteilen: 1. Zeitaktualisierung oder Vorhersage und 2. Messungsaktualisierung oder Korrektur.

Die Zeitaktualisierung ist für die Projektion des aktuellen Systemzustandes und des Fehlers in die Zukunft zuständig. Dadurch erhält man eine *a priori* Abschätzung für den nächsten Zeitschritt. Die Messungsaktualisierung stellen eine Art Rückkopplung dar und verarbeiten die Messung mit der *a priori* Abschätzung in eine verbesserte *a posteriori* Abschätzung.

A.2.1 Vorhersage

Es wird der aktuelle Systemzustand x_k^- und der vermutete Fehler P_k^- anhand der Eingänge u und dem Systemmodell A vorhergesagt:

$$\text{Zustandspropagation: } x_k^- = Ax_{k-1} + Bu_k,$$
$$\text{Fehlerpropagation: } P_k^- = AP_{k-1}A^T + Q.$$

Die Matrix A beschreibt das Verhalten des Systems. Sie beschriebt den Zusammenhang zwischen dem Systemzustand im vorangehenden Zeitschritt $k-1$ und dem aktuellen Systemzustand fü den Fall dass keine neue Eingabe und kein Rauschen existiert. Die Matrix B verknüpft die optionale Eingabe u mit dem Systemzustand x, sie unterscheidet sich von der Einheitsmatrix wenn die Werte des Vektors u verschieden gewichtet in das Filter eingehen sollen.

A.2.2 Messung

Zum Zeitpunkt t_k wird eine Messung mit dem Ergebnis z_k vorgenommen. Durch verschiedene Faktoren kann es sich hierbei jedoch nicht um den exakten Wert, sondern die Messung kan nur einen Bereich angeben, in der sich der mit einer gewissen Wahrscheinlichkeit befindet.

A.2.3 Korrektur

Im zweiten Teil wird die Messung z_k mit dem im vorherigen Schritt geschätzten Systemzustand x_k^- verrechnet und das Modell für die nächste Schätzung angepasst.

$$\begin{gathered}\text{Stärke der Korrektur: } K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}, \\ \text{Schätzung aktualisieren: } x_k = x_k^- + K_k(z_k - H x_k^-), \\ \text{Fehler aktualisieren: } P_k^- = (I - K_k H) P_k^- .\end{gathered}$$

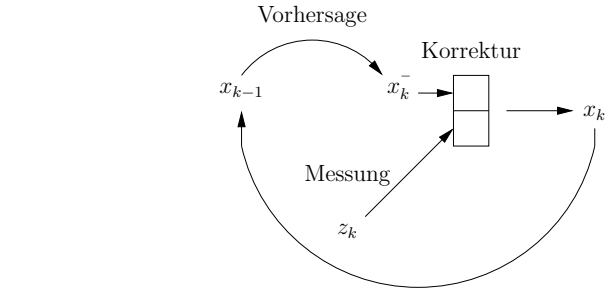
Die Matrix H beschreibt wie die aktuelle Messung z_k in den Systemzustand eingeht. Schließlich bezeichnet die Matrix K den Kalman Zuwachs, welche die *a posteriori* Fehler Kovarianzmatrix P minimiert.

Die beiden Kovarianzmatrizen Q und R beschreiben das Rauschen innerhalb des Systems und bei der Messung. Das System- und das Messrauschen werden als weiß und normalverteilt mit Erwartungswert Null angenommen. Die Annahme der Normalverteilung begründet sich aus dem zentralen Grenzwertsatz der Statistik, der aussagt, dass eine Summe von n Zufallsvariablen gliecher Streuung und Erwartungswerte für $n > 30$ gegen die Normalverteilung konvergiert [25], [7].

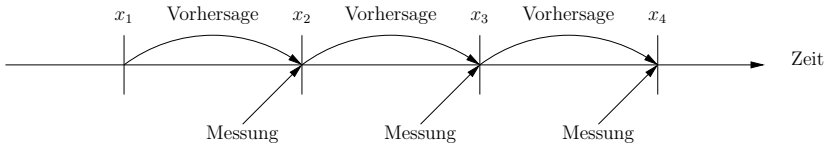
A.3 Das erweiterte Kalman-Filter

Das Kalman-Filter adressiert das allgemeine Problem der Abschätzung eines Zustandes eines diskreten dynamischen Systemes, das sich durch eine *lineare* stochastische Differenzgleichung beschreiben läßt. Wenn das System bzw. das Verhältnis von Messung zum System nicht linear ist, kann ein einfaches Kalman-Filter nicht benutzt werden. Vielmehr muss ein Filter, das über den aktuellen Mittelwert und der aktuellen Kovarianz linealisiert eingesetzt werden; ein solches Filter wird erweitertes Kalman-Filter genannt.

Ohne auf weitere Details einzugehen soll hier nur erwähnt werden, dass im Falle eines erweiterten Kalman-Filters die Matrizen A und B in der Zustandspropagation durch einen nicht-lineare Funktion f ersetzt werden. Die Matrix



(a)



(b)

Abbildung A.1: (a) Der Kalman-Filter in einer Schleife: Zuerst findet die aus Zustands- und Fehlerpropagation bestehende Vorhersage x_k^- vom vorherigen Zeitschritt t_{k-1} zum aktuellen Zeitschritt t_k statt. Anschließend werden der Zustand und der Fehler korrigiert, indem Messung z_k und Vorhersage x_k^- zum neuen Zustand x_k fusioniert werden. (b) wie (a) jedoch ist die wiederholte Anwendung des Kalman-Filters über die Zeit dargestellt.

H muss ebenefalls durch eine nicht-lineare Funktion h ersetzt werden. außerdem müssen die Jacobischen Matrizen A, W, H und V berechnet werden. A ist die Jacobische Matrix der partiellen Ableitungen von f bezüglich x: $A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(x_k, u_k, 0)$. W ist die Jacobische Matrix von f bezüglich w; H ist die Jacobische Matrix von h bezüglich x; V ist die Jacobische Matrix von h bezüglich x;

Die Gleichungen für das erweiterte Kalman Filter sind:

$$\begin{gathered}
\text{Zustandspropagation: } x_k^- = f(x_{k-1}, u_k, 0), \\
\text{Fehlerpropagation: } P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T, \\
\text{Stärke der Korrektur: } K_k = P_k^- H^T (H P_k^- H^T + V_k R V_k^T)^{-1}, \\
\text{Schätzung aktualisieren: } x_k = x_k^- + K_k(z_k - h(x_k^-, 0)), \\
\text{Fehler aktualisieren: } P_k^- = (I - K_k H) P_k^- .
\end{gathered}$$

Diese kurze Einführung in die Theorie des Kalman-Filters basiert auf [39], [35].

Anhang B

Der PID-Regler

Regler im Allgemeinen werden zur Steuerung von *Prozessen* verwandt. Der Prozess hat dabei zum Zeitpunkt t eine *Eingabe* W_t und eine davon abhängende *Ausgabe* T_t. Dabei kann zwischen einer Änderung in der Eingabe und deren Auswirken auf die Ausgabe eine gewisse Zeit vergehen, die sog. Totzeit. Zudem existiert ein von außen vorgegebener *Sollwert* T_s für die Ausgabe. Aufgabe des Reglers ist es, die Eingabe so zu modifizieren, dass die Ausgabe dem Sollwert entspricht. Die Differenz zwischen Sollwert und Ausgabe wird *Fehler* gennant.

Regler müssen zwei widersprüchlichen Anforderungen genügen: Einerseits soll bei einer Änderung des Sollwertes der Regler schnell reagieren. Andererseits ist es erwünscht, dass die Ausgabe möglichst konstant bleibt, sobald der Sollwert erreicht ist.

PID-Regler betrachten zwei Arten von Fehler: den absoluten Fehler und die relative FehlerÄnderung. Der absolute Fehler ist die aktuelle Differenz zwischen Sollwert und Ausgabe. Die relative FehlerÄnderung gibt eine Auskunft darüber, ob der absolute Fehler geringer oder größer geworden ist.

B.1 Proportionaler (P-) Regler

Eine der ersten Ideen wenn es um den Design einer automatische Prozessregelung geht, ist was allgemein als proportionaler Regler bekannt ist. Gemeint ist damit, dass die Stärke der Korrektur von der Größe des Fehlers abhängt: ist der Fehler klein, sollte auch nur eine kleine Korrektur vorgenommen werden; ist dagegen der Fehler groß, so sollte auch die Korrektur groß ausfallen.

$$W_t = P \cdot (T_s - T_t)$$

Der proportionale Regler ist jedoch nicht ausreichend, da er eine Oszillation oder eine bleibende Regelabweichung nicht beseitigen kann.

B.2 Integrierender (I-) Regler

Wird statt der Stellgröße, wie beim P-Regler, die Stellgeschwindigkeit von der Regelabweichung beeinflußt, dann entsteht ein integrierender Regler.

Dabei wird über die Zeit die Summe der Fehler gebildet und in Abhängigkeit davon geregelt. Eine bleibende Regelabweichung würde zu einem unendlichen Fehler führen und wird daher von einem I-Regler beseitigt.

$$W_t = I \cdot \int (T_s - T_t)\, dt$$

B.3 Differenzierender (D-) Regler

Der differenzierende Regler berücksichtigt die Tatsache, dass eine Änderung in der Eingabe sich erst nach einer gewissen Zeit auf die Ausgabe auswirken wird. Die Berücksichtigung der Totzeit führt dazu, dass Änderungen in der Eingabe für eine Zeit zurückgehalten werden, da bekannt ist, dass schon gemachte Änderungen sich noch nicht ausgewirkt haben.

Die Hinzunahme eines D-Reglers führt dazu, dass der Regler weniger stark überschießt.

Ein solcher Regler wird üblicherweise nur in Verbindung mit einem P- oder I-Regler verwandt, um die Dämpfung und damit die Stabilität des Systems zu verbessern:

$$W_t = D \cdot \frac{d}{dx} (T_s - T_t)\,.$$

Reine D-Regler sind technisch nicht realisierbar. Beim differentiell wirkenden Anteil in einem zusammengesetzten Regler wird in der Sprungantwort ein nadelförmiger Peak erzeugt.

B.4 PI-Regler

P-Regler bieten im Regelkreis den Vorteil, dass sie kurze Einregelzeiten haben, besser gedämpft regeln als andere Elemente und eine kleine überschwingweite bewirken. Sie haben aber den großen Nachteil einer *bleibenden Regelabweichung* (Bei P-Reglern kommt es im Regelkreis häufig dazu, dass der Regler einen scheinbar stationären Zustand in der Stellwertausgabe erreicht, bei dem aber auf der Strecke noch kein Ist- Sollwertausgleich erreicht wurde).

I-Regler dagegen haben längere Einregelzeiten, eine schlechtere Dämpfung und größere überschwingweiten. Sie weisen aber *keine bleibende Regelabweichung* auf.

Der einfachste zusammengesetzte Regler ist daher der PI-Regler:

$$W_t = P \cdot \left((T_s - T_t) + I \cdot \int (T_s - T_t)\, dt \right).$$

B.5 PID-Regler

PI-Regler kann durch die Synthese der Merkmale in den häufigsten Fällen ein gestelltes Problem grundsätzlich bewältigen. Der im nachfolgenden beschriebene PID-Regler führt lediglich in einigen Fällen zu einer Beschleunigung.

Der PID-Regler stellt die Summe aus den Signalen einzelner P-, I- und D-Regler dar:

$$W_t = P \cdot \left((T_s - T_t) + I \cdot \int (T_s - T_t)\, dt + D \cdot \frac{d}{dx} (T_s - T_t) \right).$$

Literaturverzeichnis

[1] German Open 2004. *http://www.ais.frauenhofer.de/GO/2004/.*

[2] Team AllemaniACs. *http://www.robocup.rwth-aachen.de/.*

[3] Kai O. Arras, Nicaola Tomatis, Björn T. Jensen und Roland Siegwart. Multisensor on-the-fly localization. In *IEEE International Conference on Robotics and Autonomous Systems*, Seiten 131–143. 2001.

[4] Sven Behnke und Raúl Rojas. A hierarchy of reactive behaviors handles complexity. In M. Hannebauer et al. (Herausgeber), *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, Seiten 125–136. 2001.

[5] J. Borenstein und L. Feng. Umbmark - a method for measuring, comparing and correcting dead-reckoning errors in mobile robots. In *IEEE Transactions on Robotics and Automation*. 1996.

[6] R. A. Brooks. A robust layered control system for a mobile robot, 1986.

[7] R. G. Brown und P. Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. Wiley and Sons Inc., 1996.

[8] G. Buchsbaum. A spartial processor model for object color perception. In *Journal of the Franklin Institute, Volume 310*, Seiten 1–26. 1980.

[9] Team Philips CFT. *http://www.cft.philips.com/robocup/.*

[10] K. S. Ching und L. Kleeman. Accurate odometry and error modelling for a mobile robot. In *IEEE International Conference on Robotics and Automation*. 1997.

[11] Intel Performance Counter. *http://www.intel.com/software/products/vtune/vpa/countermon.htm.*

[12] J. L. Crowley und F. Chenavier. Position estimation for a mobile robot using vision and odometry. In *IEEE International Conference on Robotics and Automation*, Seiten 2588–2593. 1992.

[13] Offizielle Homepage der Fédération Internationale de Football Association (FIFA). *http://www.fifa.com/*.

[14] Offizielle Homepage der RoboCup Federation. *http://www.robocup.org/*.

[15] L. Drolet, F. Michaud und J. Cote. Adaptable sensor fusion using multiple kalman filters. In *IEEE International Conference on Intelligent Robots and Systems*. 2000.

[16] C. A. Glasbey. An analysis of histogram-based thresholding algorithms. In *Graphical Models and Image Processing*, Seiten 532–537. 1993.

[17] R. M. Haralick und L. G. Shapiro. *Computer and Robot Vision, Volume I*. Addison Wesley, Reading, MA, 1992.

[18] F. H. Hsiao und S. T. Pan. Robust kalman filter synthesis for uncertain multiple time-delayed stochastic systems. In *Journal of Dynamic Systems, measurement and control*. 1996.

[19] R. E. Kalman. A new approach to linear filtering and prediction problems. In *Transaction of ASME - Journal of Basic Engineering, 82 (Series D)*, Seiten 35–45. 1960.

[20] Thomas Dall Larsen, Nils A. Andersen, Ole Ravn und Niels K. Poulsen. Incorporation of time delayed measurements in a discrete-time kalman filter.

[21] Intel Open Source Computer Vision Library. *http://www.sourgeforge.net/projects/opencvlibrary*.

[22] Pedro Lima, Andrea Bonarini, Carlos Machado, Fabio Marchese, Carlos Marques und Fernando Ribeiro. Omni-directional catadioptric vision for soccer robots, 2001.

[23] A. Mackworth. On seeing robots. In *Vision Interface*. Vancouver, May 1992.

[24] D. T. Magill. Optimal adaptive estimation of sampled stochastic processes. In *IEEE Transactions on Automatic Control*, Seiten 434–439. 1995.

[25] P. S. Maybeck. Stochastic models, estimation and control. In *Vol. 141*. 1979.

[26] Team Minho. *http://www.robotica.dei.uminho.pt/robocup/*.

[27] NetLexikon. *http://www.netlexikon.de*.

[28] Pedro Pinheiro und Pedro Lima. Bayesian sensor fusion for cooperative object localization and world modeling. In *8th Conference on Intelligent Autonomous Systems*. 2003.

[29] Sony Aibo Roboter. *http://www.sony.net/Products/aibo/*.

[30] Raúl Rojas. Robocup - Das Buch.

[31] Patrick Schmider. Einsatz einer omnidirektionalen Kamera im RoboCup, 2002.

[32] Alvy Ray Smith. Color gamma transform pairs. In *SIGGRAPH 78, Vol. 12, No. 3*, Seiten 12–19. 1978.

[33] R. C. Smith und P. Cheesman. On the representation and estimation of spatial uncertainty. In *The International Journal of Robotics Research*, Seiten 56–68. 1986.

[34] Ashley W. Stroupe, Martin C. Martin und Tucker Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Technical Report CMU-RI-TR-00-30*. Carngie Mellon University, 2000.

[35] Carlo Tomasi. Stochastic state estimation. In *CS 296.4 Class Notes*. Duke University, 2002.

[36] Andre Treptow und Andreas Zell. Real-time object tracking for soccer-robots without color information, 2002.

[37] Felix von Hundelshausen, Sven Behnke und Raúl Rojas. An omnidirectional vision system that finds and tracks color edges and blobs. *Birk, A., Coradeschi, S., Tadokoro, S. (eds): RoboCup-01: Robot Soccer World Cup V, Springer, 2001.*, 2001.

[38] Felix von Hundelshausen, Sven Behnke und Raúl Rojas. FU-Fighters team description. In D. Polani et al. (Herausgeber), *RoboCup - Proceedings of the International Symposium*. 2003.

[39] Greg Welch und Gary Bishop. An introduction to the kalman filter. In *SIGGRAPH*. 2001.

[40] T. Wilhelm, J. Kludas, H.-J. Böhme und H.-M. Gross. Automatischer weißabgleich für eine omnidirektionale kamera. In *Technical Report, Technische Universität Ilmenau*. 2002.