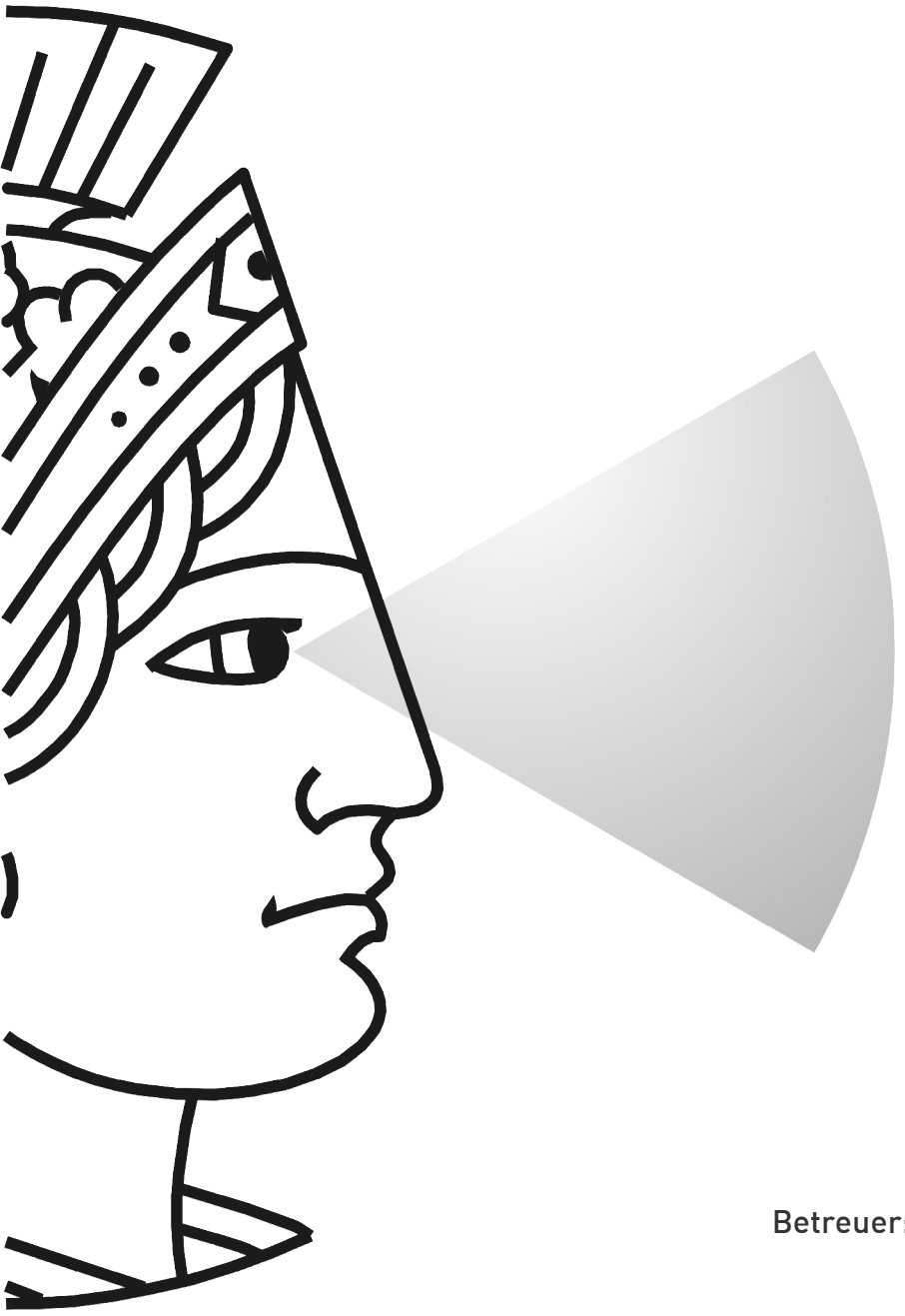
TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachgebiet
Simulation & Systemoptimierung
Prof. Dr. Oskar von Stryk

Gegnererkennung und -modellierung für Passspiel in der 4-Legged RoboCup League

Opponent Player Detection and Modelling for
Pass Playing in the 4-Legged RoboCup League



Diplomarbeit
Sommersemester 2005

Marc Dassler
Informatik, Matrikelnummer 562629

Betreuer: Dipl. Tech. Math. Maximilian Stelzer
Eingereicht am 26.09.2005

EHRENWÖRTLICHE ERKLÄRUNG

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, im September 2005 Marc Dassler

ZUSAMMENFASSUNG

Das Fachgebiet Simulation und Systemoptimierung der Technischen Universität Darmstadt beschäftigt sich mit der Forschung an mobilen autonomen Robotersystemen und nimmt mit dem Team der Darmstadt Dribbling Dackels am RoboCup in der 4-Legged Robot League teil.

In dieser Arbeit wurde für die bereits bestehende Softwarearchitektur GT200x eine Spielererkennung und eine Gegnermodellierung entwickelt, die ein Passspiel ermöglicht.

Die entwickelte Spielererkennung und Gegnermodellierung wurde auf dem RoboCup 2005 in Osaka/Japan beim Penalty Shootout erfolgreich eingesetzt.

ABSTRACT

The Simulation and System Optimization Group at the Technische Universität Darmstadt deals with research on mobile, autonomous robot systems and participates at the RoboCup in the 4-Legged Robot League with the Darmstadt Dribbling Dackels team.

Under this assignment, for the existing software architecture GT200x, 'Player Recognition' and 'Opponent Modelling' modules were developed to enable passes in the game.

These modules were also successfully employed in the penalty shootouts at the RoboCup 2005 held in Osaka, Japan.

DANKSAGUNG

Ich möchte meiner Mutter und meinem Vater danken, die mich moralisch und finanziell all die Jahre unterstützt und gefördert haben. Vielen, vielen Dank.

Mein Dank auch Gregor und Thorsten, die immer motivierende Worte gefunden haben. Max R., Dirk und Sebastian, die immer Zeit hatten, eine Idee mit mir zu durchdenken und bei Problemen in C++ immer hilfreich zur Seite standen. Mein Dank an Hannah, Carsten, Costa und Verena, die mir halfen, diese Arbeit von Rechtschreib- und Grammatikfehlern zu befreien.

Mein Dank an meinen Betreuer Max Stelzer, der immer Zeit für mich fand.

INHALTSVERZEICHNIS

1_EINLEITUNG

„By 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer.“

RoboCup Federation

1.1_Aufbau der Arbeit

Diese Arbeit soll eine nützliche Einführung in die Spielererkennung und Modellierung sein, um Interessenten die aufwendige Analyse der Programmcodes zu ersparen, und Anregungen für weitere Entwicklungen im RoboCup im Bereich Gegnermodellierung und Teamtaktik geben.

Die Arbeit ist in sechs Kapitel gegliedert. Im ersten Kapitel wird in den RoboCup eingeführt und die Arbeit in Bezug zu vorhandener Realisierung gesetzt. Im zweiten Kapitel wird auf die Grundlagen für diese Arbeit eingegangen und ein Blick in die reale Welt des Fußballs geworfen. Im dritten Kapitel wird die entstandene Realisierung der Spielererkennung, im vierten die der Gegnermodellierung und im fünften die Passberechnung, die Bewegungen und das Passverhalten erläutert. Im anschließenden Kapitel werden die Ergebnisse zusammengefasst.

1.2_RoboCup

Der RoboCup [12] ist eine internationale Vereinigung aus Universitäten und Forschungsanstalten, die es sich zum Ziel gesetzt haben, die Forschung und Entwicklung von Robotik und künstlicher Intelligenz voran zu treiben. Seit 1997 treffen sich dazu jährlich die Teilnehmer auf den nationalen (German Open, American Open, Australien Open und Japan Open) und internationalen Wettbewerben (RoboCup Weltmeisterschaft). Hier treten fußballspielende autonome Roboter und Softwareagenten in Wettbewerben gegeneinander an, um die Ergebnisse der Forschung im Bereich der Robotik zu messen.

Um die verschiedenen Forschungsschwerpunkte zu ermöglichen, ist der RoboCup in mehrere Ligen aufgeteilt: Small-Size, Mid-Size, 4-Legged, Humanoid und Simulation League. In der 4-Legged League, in deren Rahmen diese Arbeit entwickelt wurde, gilt die Regelung, dass an der Hardware, also den Robotern selbst, keine Veränderungen vorgenommen werden dürfen. Alle Entwicklungen beziehen sich daher nur auf die Software und die verwendeten Algorithmen. Dies garantiert die Chancengleichheit für alle Teams. Die Liga spielt mit den von SONY hergestellten Spielzeugrobotern ERS-210/ERS-210A und dem Nachfolgemodell ERS-7.

Die Darmstadt Dribbling Dackels sind seit 2001 auf nationalen Wettbewerben vertreten und bilden auf internationalen Wettbewerben mit den Universitäten Humboldt Berlin, Bremen und Dortmund das GermanTeam [10] und treten in der 4-Legged League an.

Das GermanTeam wurde 2004 in Lissabon/Portugal und 2005 in Osaka/Japan Weltmeister seiner Liga.

1.3_Roboter

Der zur Zeit eingesetzte Roboter ist der von SONY hergestellte ERS-7. In den Jahren zuvor wurde das ältere Modell ERS-210 bzw. die modifizierte Version ERS-210A mit schnellerem Prozessor eingesetzt. Er ist ausgestattet mit einer Farbkamera, Stereo-Mikrofonen, Tast-, Abstands-, Geschwindigkeits- und Vibrationssensoren [14]. Er verfügt über einen 576 MHz schnellen 64 bit RISC MIPS Prozessor mit 64 MB Hauptspeicher [15]. Jedes Bein wird von drei Motoren angetrieben und kann den ERS-7 bei der jetzigen eingesetzten Gangart auf 44 cm/s beschleunigen. Die eingesetzt Farbkamera besitzt eine Auflösung von 208 x 160 Pixel. Der Roboter besitzt zwei Infrarotabstandssensoren, die im Kopf und am Rumpf angebracht sind. Beschleunigungs- und Lagesensoren ermöglichen es ihm zu erkennen, ob er aufgerichtet ist oder nicht. Den Programmcode erhält er über einen speziellen SONY Memorystick, der in den Rumpf eingesetzt wird.

Abbildung 1: SONY ERS-210A.

Abbildung 2: SONY ERS-7. Quelle SONY Presseerklärung

1.4_Spielfeld

Das Spielfeld hat seit den Regeländerungen 2005 keine Banden mehr und wurde vergrößert auf 6 x 4 Meter. Dies macht eine gute Selbstlokalisierung unabdingbar, da unter anderem Zeitstrafen für das Verlassen des Spielfelds vergeben werden.

Einen Vorteil hingegen hat das Entfernen der Bande für die Spielererkennung, denn nun sind nur noch die Roboter und der untere Teil der Landmarken weiß im Gegensatz zu früher, als auch die Banden weiß waren. Dies lässt das Auffinden des Bodenpunkts eines Roboters erst zu (siehe 3.3). Auch die Art des Spiels hat sich geändert. Durch die fehlenden Banden neben dem Tor, das mit den neuen Regeln nun 60 cm misst, ist es jetzt nicht mehr möglich, den Ball in das Tor hineinzudrücken, wie es früher üblich war. Durch den zusätzlichen Raum und die damit verbundenen größeren Entfernungen ist das Passspiel interessanter geworden.

Abbildung 3: Spielfeld nach den Regeln 2005. Aufnahme aus dem Simulator.

1.5_Motivation und Zielsetzung

Zum Jahre 2050 sollen die Roboter des RoboCups in der Lage sein, den menschliche Fußballweltmeister zu schlagen. Seit langem spielen die Roboter des RoboCups Fußball. In den Ligen Mid-size und 4-Legged ist auch jeder Roboter für sich genommen autonom, und nur auf 4-Legged League beziehen sich die in dieser Diplomarbeit entwickelten Programmteile. Autonom heißt in diesem Zusammenhang, dass der Roboter plant und handelt selbständig nach seinem vorgegebenen Verhalten und wird nicht ferngesteuert.

Das Verhalten ist in der Realisierung des GermanTeams in ein Rollenverhalten aufgeteilt: Striker (Stürmer), Supporters (Mittelfeldspieler) und Goalie (Torwart). Diese Rollen wechseln dynamisch, je nach Entfernung des einzelnen Spielers zum Ball. Ausgenommen davon ist der Torwart. Zu Beginn dieser Arbeit existierte noch keine Teamtaktik in dem Sinne, die es den Robotern erlaubte, Spielzüge zu planen und durchzuführen. Zwar besitzt des GermanTeam ein Stellungsspiel, was sich auf die Positionen der Spieler im Bezug auf ihre Rollen (Striker, defensive und offensive Supporter) beziehen. Dieses Stellungsspiel umfasst allerdings keine Interaktion zwischen den Spielern erlaubt. Da eine erweiterte Teamtaktik mit Interaktion es ermöglichen würde Räume, zu überbrücken und den Gegner auszuspielen, ist sie höchst wünschenswert.

Um diesem Ziel näherzukommen, ist es notwendig, ein globales Weltbild der Spielsituation zu entwickeln, das koordinierte Spielzüge ermöglicht. Dies erfordert die Kommunikation der von jedem einzelnen Roboter gewonnenen Information und eine individuelle Fusion dieser Daten. In einigen Teilen wird dies bereits vom GermanTeam umgesetzt. Ein gutes Beispiel ist der Teamball, der es jedem Spieler ermöglicht, auch auf einen nicht selbst gesehenen Ball in gewissem Maße zu reagieren. Um nun eine globale Gesamtsicht auf die Spielsituation zu erhalten, ist es notwendig, die Position der gegnerischen Spieler zu erfassen und im Team zu kommunizieren bzw. zu modellieren.

Ziel dieser Arbeit ist es, ein globales Gegnermodell zu entwickeln, das es dem Team ermöglicht, kollisionsfreie Pässe zu planen und auszuführen. Dazu gehört die Entwicklung einer robusten Spielererkennung, eines Gegnermodells, einer Passplanung, einer Spielerkommunikation, die alle Informationen des Teams zusammenführt und eines Teamverhaltens, das den Pass ausführt.

1.6_Integration

Die in dieser Diplomarbeit entstandene Software ist eingebettet in die Software des GermanTeams 2004/2005 [11]. Während der Entwicklungszeit wurde sie ständig den Änderungen der Architektur angepasst und optimiert. Das GT200x Roboter-Framework ist modular aufgebaut, so dass jede anfallende Aufgabe in einem Modul repräsentiert wird. Diese Architektur garantiert die Wiederverwendbarkeit und die einfache Weiterentwicklung.

Für die Entwicklung des GT200x Framework wird die Sprache C++ eingesetzt. Neben dem entwickelten Framework werden drei Tools bereitgestellt, die zur Entwicklung eingesetzt werden.

1.6.1_RobotControl

RobotControl ist das älteste Werkzeug, das im GT200x Code zu Verfügung steht. Von den zahlreichen Funktionen die RobotControl bietet, wird nur auf die für diese Arbeit relevanten Teile eingegangen.

RobotControl hat im Gegensatz zu RobotControl2 und dem Simulator die Möglichkeit, sich zu mehreren Robotern zu verbinden und deren Wahrnehmungen senden zu lassen. Es bietet auch die Möglichkeit, einen Roboter zu simulieren und zuvor aufgenommene Logfiles[1] an den simulierten Roboter weiterzureichen. Mit RobotControl kann man sich auch mit der Deckenkamera [8] verbinden, mit Hilfe derer man die exakte Position der Roboter auf dem Spielfeld zu berechnen kann. Die Deckenkamera erweis sich als nützliches Werkzeug, um die Qualität der Gegnermodellierung zu bestimmen.

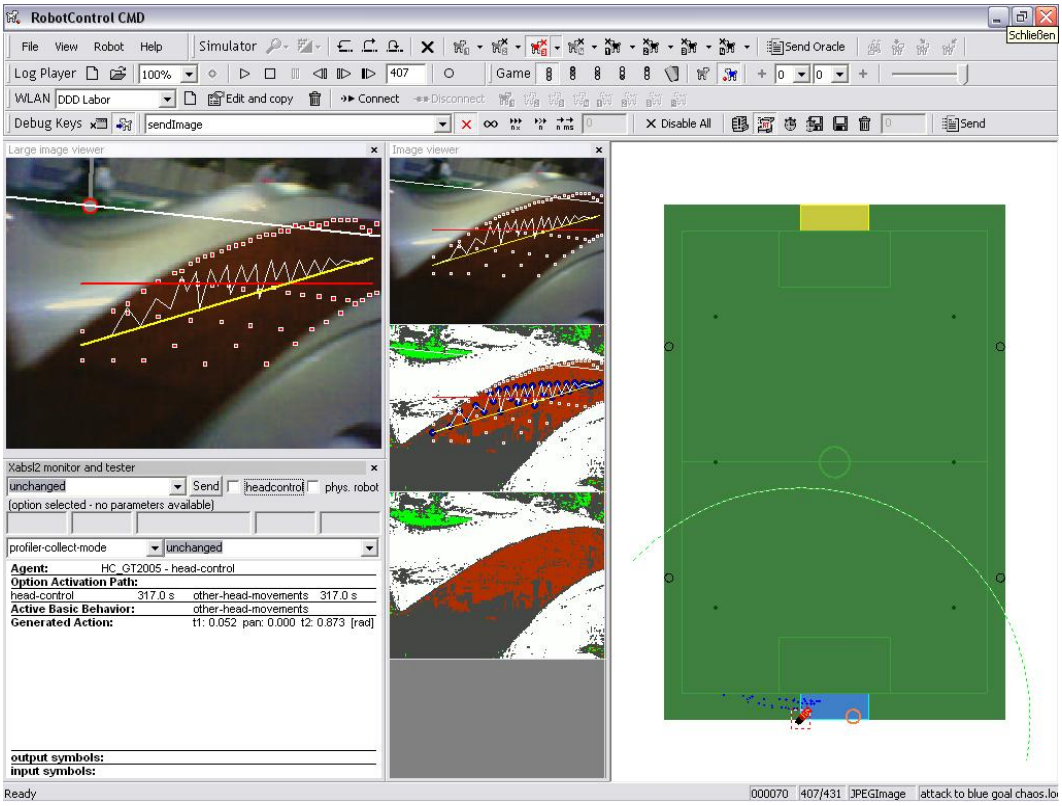


Abbildung 4: RobotControl

1 In einem Logfile können alle Informationen über einen Roboter gespeichert werden. Dazu gehören Kamerabilder, Gelenksensorendaten, Verhaltenszustände, etc.

1.6.2_RobotControl2

Da RobotControl durch den hohen Funktionsumfang sehr schwerfällig geworden ist, und somit es nicht sonderlich stabil und die Kompilierungszeit sehr hoch ist, wurde 2004 mit der Entwicklung von RobotControl2 begonnen. Es ist in C# entwickelt und enthält im Gegensatz zu seinem Vorgänger keine Simulation von Robotern. Es soll vor allem der Visualisierung und Analyse der vom Roboter gelieferten Daten dienen. Leider war es während der Entwicklung dieser Arbeit mit RobotControl2 nicht möglich, sich zu mehreren Robotern zu verbinden. Aus diesem Grund wurde größtenteils RobotControl und der Simulator genutzt.

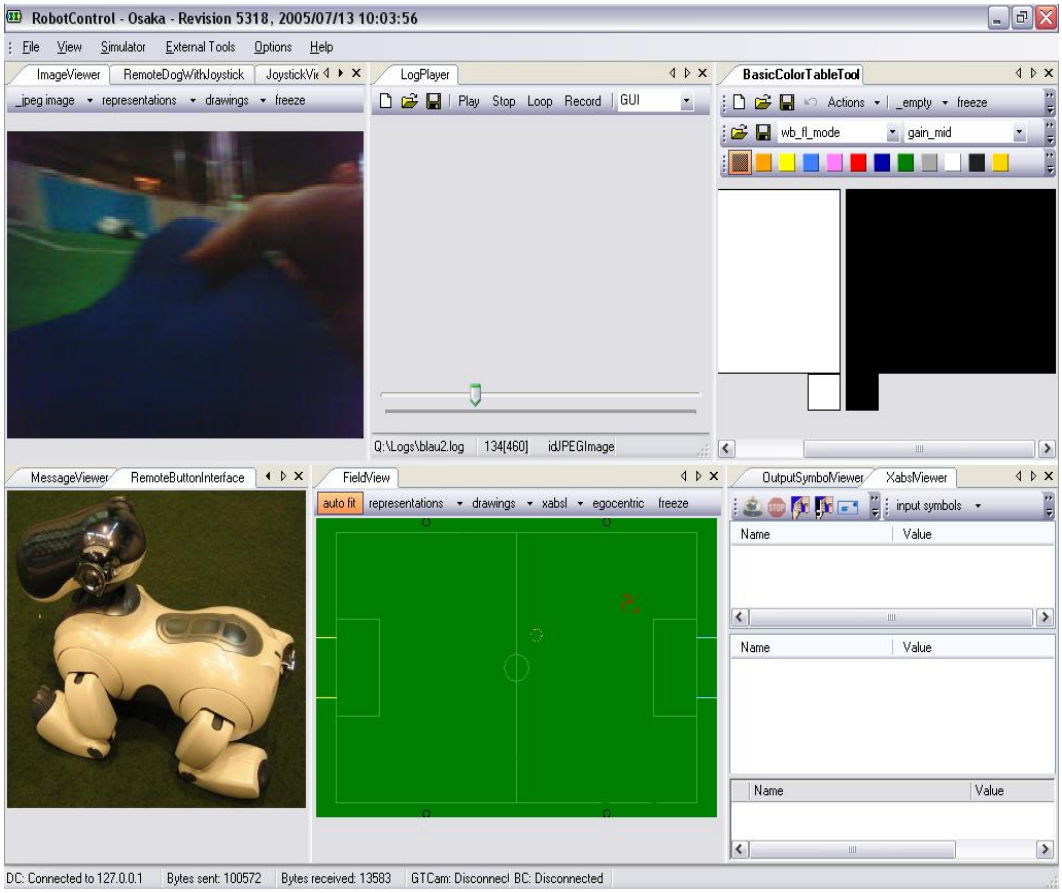


Abbildung 5: RobotControl2

1.7_Simulator

Der Simulator ist sehr hilfreich um Verhalten zu entwickeln, da hier alle Perzepte[2] vom Oracle[3] simuliert werden können. Auch ist es möglich, Ausgaben eines ganzen Moduls zu simulieren, wie z.B. die Selbstlokalisation. Der Simulator ist in der Lage, alle Roboter auf dem Feld zu simulieren und ermöglicht es, zu jedem Zeitpunkt anzuhalten und das Verhalten der einzelnen Spieler zu analysieren. Dies erspart viel Zeit, da die Alternative der Test am realen Roboter wäre. Wenn man die Interaktion der Roboter analysieren möchte, so ist dies fast ausschließlich im Simulator möglich, da man beim Test auf den Robotern zum einen nicht das Spiel anhalten kann, ohne den Status des Verhaltens und der Eingangsdaten zu ändern, und da zum anderen die Bandbreite des Wireless LAN (11 Mbps) sowie die Rechenleistung der Roboter beschränkt ist. Diese können nicht alle aufgenommen Bilder in JPEG-Bilder umwandeln und übermitteln, was für eine spätere Simulation in RobotControl wünschenswert wäre.

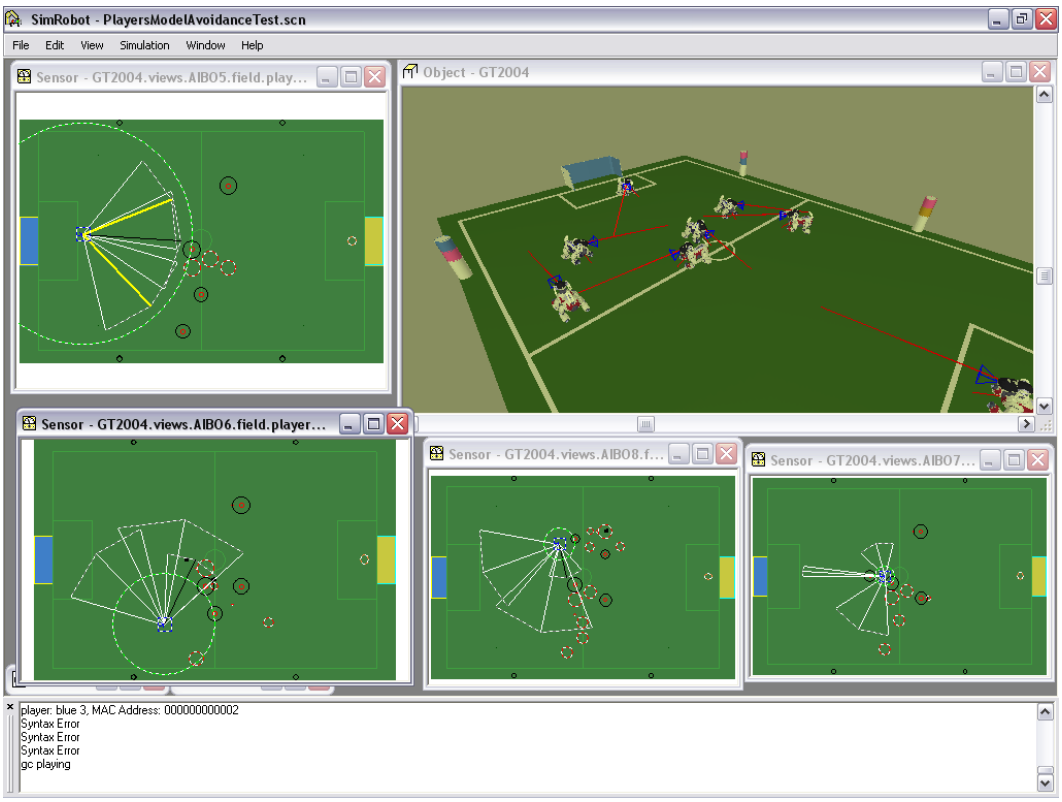


Abbildung 6: Simualtor mit den Fieldviews des PlayersLocator

2 Als Perzept werden die Wahrnehmungen des Roboters bezeichnet (Ball, Tor, Flagge, Gegner, etc.)

3 Das Oracle „prophezeit“ dem Roboter die Daten (Abstand, Winkel, etc.) der Objekte auf dem Spielfeld.

Der größte Teil der Entwicklung der beiden Gegnermodellierungsansätze wurde mit Hilfe des Simulators betrieben. Dazu wurde eine PlayersLocatorFieldview und ein PlayersPerceptOracle Klasse entwickelt.

1.7.1_GT200x Modulübersicht

Die folgende gekürzte Modulübersicht des GermanTeam-Codes GT200x zeigt alle vitalen Module, die für den Roboterfußball notwendig sind. Der Aufbau folgt dem bekannten Prinzip „Sense – Plan – Act“. Für diese Arbeit relevante Module sind: Selbstlokalisator (SelfLocator), Bildverarbeitung (ImageProcessor), Gegnermodellierung (PlayersLocator) und Verhaltenssteuerung (Behavior).

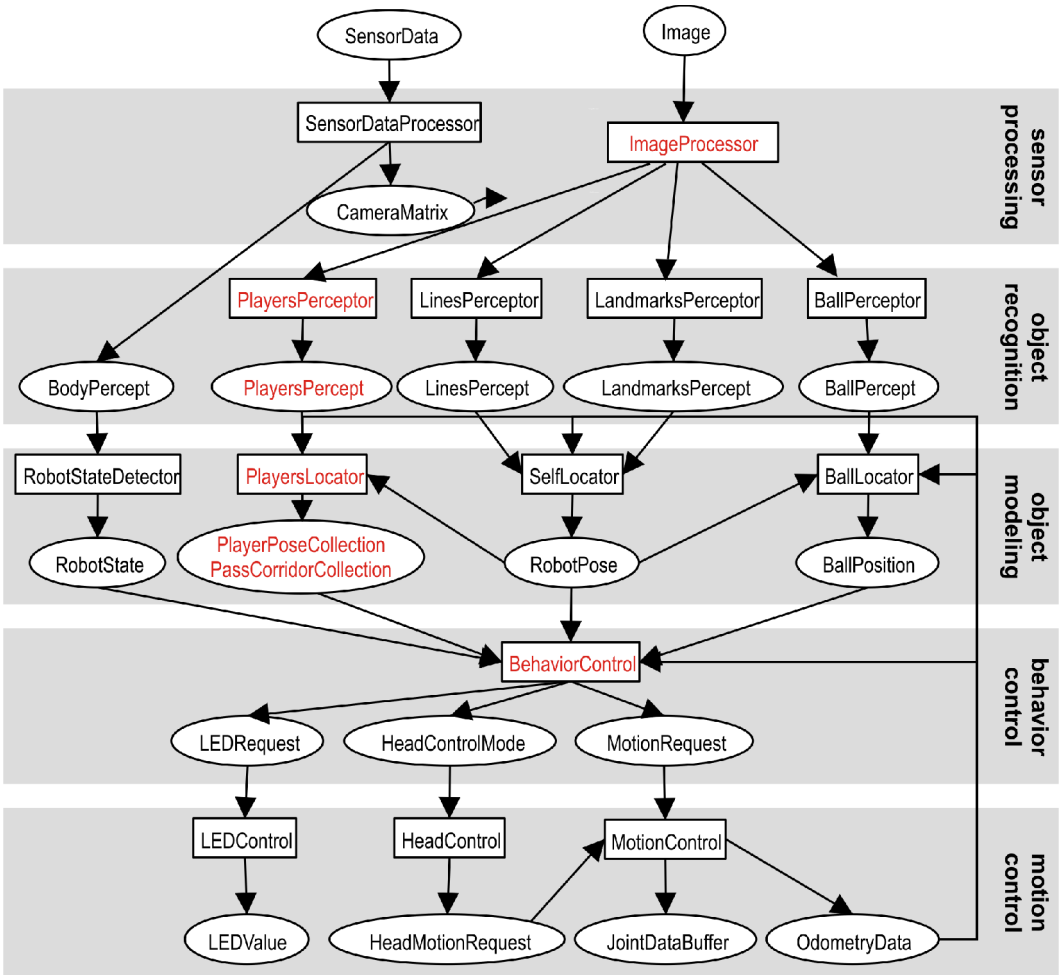


Abbildung 7: Modulübersicht. Quelle GT2004 Teambericht

In der Modulübersicht sind alle Module, die in dieser Arbeit entwickelt bzw. erweitert wurden, mit roter Schrift markiert. Die entwickelten Bewegungen sind nicht in der Übersicht zu sehen, da auf Grund der Speicherung in sog. MOF-Dateien keine Änderungen des Quellcodes nötig war. Das Gleiche gilt für die Verhaltenssteuerung. Da aber in dieser Arbeit neue InputSymbole entwickelt wurden, ist das BehaviorControl-Modul rot markiert.

1.8_Andere Ansätze

In der Mid-Size League wurde von der Universität Ulm eine Spielererkennung entwickelt, die mit zwei verschiedenen Ansätzen arbeitet. Zum einen eine Blob-based Detection, zum anderen eine Color-Edge-based Detection. Im Anschluss werden die erkannten Objekte durch ein neuronales Netzwerk gefiltert. Beide Ansätze sind nicht auf den GT200x-Code übertragbar, da ein zusätzliche Blob-Detection neben dem vorhandenen Scanline-Verfahren des GT200x-ImageProcessors zuviel Rechenzeit beanspruchen würde. Das gleiche gilt für einen zusätzlichen Kantenfilter. [9]

Allerdings benutzen die australischen 4-Legged League Teams ein Blob-based ImageProcessor und können, mit Hilfe eines trainierten neuronalen Netzwerks nahe Gegner gut erkennen. Unter anderem auch die Ausrichtung des erkannten Spielers. Nach Aussagen des Teamleiters der NUbots der University of Newcastle/Australien liegt die Erfolgsquote bei 80 %.

2_GRUNDLAGEN

„Some people think football is a matter of life and death. I don't like that attitude. I can assure them it is much more serious than that."

Bill Shankly

2.1_Geschichtliches

Grundlage für eine Teamtaktik sind unterschiedliche Aufstellungsformen. Schon in der Frühzeit (2694 v.u.Z.) gab es in China in dem dortigen fußballähnlichem Spiel „ts' u-chü“ (Stossball) Variationen in der Aufstellung. 1872, kurz nachdem ein allgemein gültiges Regelwerk für das uns bekannte Fußballspiel eingeführt wurde, standen sich die Mannschaften Schottland und England gegenüber und spielten in einer 2:2:6- bzw. 2:1:7-Formation[4]. Schottland nahm für die Verteidigung einen Stürmer zurück. Das System der annähernden ausgeglichenen Kette bewährte sich und das Spiel endete 0:0. Im Rückspiel übernahmen die Engländer die Aufstellung und gewannen 3:2. Klassische Aufstellungen besitzen meistens 4er Ketten (4:4:2 oder 4:3:3) und sind noch im heutigen Fußball die Grundlage für Mannschaftstrategien [1].

Auch im RoboCup hat sich ein Stellungsspiel entwickelt. Das GermanTeam arbeitet mit einem Goalie, Striker und mit einem Offensive Supporter und einem Defensive Supporter. Diese positionieren sich um den Stürmer herum, um mögliche Rückpässe abzufangen bzw. den Ball anzunehmen und selbst zum Stürmer zu werden, da die Rollen dynamisch nach Abstand zum Ball vergeben werden.

4 Die Aufstellung ist so zu verstehen: Verteidigung : Mittelfeld : Sturm.

2.2_Überblick

Um einen Pass zu ermöglichen, benötigt man eine möglichst globale Sicht auf die aktuelle Spielsituation: die Position des Balls, der gegnerischen und eigenen Spieler. Die Position des Balls und die Positionen der eigenen Spieler ist durch das BallModel bzw. durch den Teamball und die TeamMessages[5] gegeben. Allerdings weicht während des Spiels die tatsächliche Position der Spieler, die nahe am Ball sind, zeitweise stark ab. Dies ist bedingt durch die Nähe zum Ball und zu den gegnerischen Spielern. Zum einen müssen der Striker und der Offensive Supporter möglichst permanent den Ball beobachten und zum anderen versperren die gegnerischen Spieler die Sicht auf Landmarken, Tore und Linien. Ohne diese Landmarken kann der SelfLocator die eigene Position nur anhand der Odometrie bestimmen. Meistens wird der Striker von anderen Spielern bedrängt und die Odometrie wird dadurch stark verfälscht. Abgesehen von diesen Widrigkeiten fehlt noch eine effektive Spielererkennung um das Bild zu vervollständigen.

2.3_Taktiken

> *„Taktik ist also zunächst die Bezeichnung für den klugen Einsatz der eigenen Kräfte, für das planvolle und zielstrebige Verhalten in einer bestimmten Situation, wobei die Stärken und Schwächen des Gegners berücksichtigt werden müssen. Für das Fußballspiel abgewandelt, heißt das in jeder Spielsituation klug zu überlegen und schnell und umsichtig zu handeln, um im sportlich-fairen Wettkampf den Sieg zu erringen.“ [1]*

Um geeignete Taktiken für den RoboCup zu entwickeln, ist es sinnvoll, den menschlichen Fußball zu analysieren. Man unterscheidet zwischen Individualtaktik, die sich auf das Verhalten eines einzelnen Spielers am Ball bezieht, und Teamtaktik, die das Verhalten eines Teams beschreibt.

Die folgenden Merksätze stammen aus einem Lehrbuch für die Fußballjugend [1] aus dem Jahr 1969 und besitzen heute noch ihre Gültigkeit. Sie waren Grundlage für den Bewertung der möglichen Pässe und des entwickelten Passverhaltens.

5 Die TeamMessages enthalten unter anderem die Position der eigenen Spieler und werden periodisch bis zu 30 mal pro Sekunde an die Mitspieler versandt.

2.3.1_Individualtaktiken

2.3.1.1_Dribbling

Dribbling bedeutet Umspielen des Gegners, also das Führen des Balls, wenn man vom Gegner bedrängt wird. Ein guter Dribbler ist eine ernste Gefahr für jede Abwehr. Viele Torschüsse werden durch kluges Dribbling vorbereitet.

Merksätze:

1. Dribbelt nicht, wenn ein Mitspieler in guter Position freisteht! Abspielen geht vor Dribbling!
2. Das Mittelfeld verleitet oft zum unfruchtbaren Dribbling. Überwindet es schnell, indem ihr den Ball abspielt. Der Ball ist euer schnellster Mitspieler!
3. Dribbelt nicht in der Abwehr. Ein Ballverlust führt schnell zum Tor!

2.3.1.2_Torschuss

Der Torschuss ist ein individuelles Mittel der Angriffstaktik. Ein erfolgreicher Torschuss krönt eine Angriffskombination oder ein Dribbling.

Merksätze:

1. Der am günstigsten stehende Spieler muss den Ball zum Torschuss bekommen!
2. Schießt nicht auf das Tor, wenn die Entfernung zu weit, der Schusswinkel zu klein, oder die Schussbahn durch Spieler versperrt ist!
3. Schießt schnell!
4. Nicht allein die Schärfe ist entscheidend, wichtiger ist die Genauigkeit!

2.3.1.3_Positionswechsel (Freilaufen)

Im modernen Fußballspiel hat der Positionswechsel und das Sich-Anbieten (Freilaufen) - das sogenannte „Spiel ohne Ball“ - der Spieler große Bedeutung. Spieler ohne Ball müssen sich freilaufen, um ein erfolgreiches Abspiel zu ermöglichen.

Merksätze:

1. Handelt nicht plan- und ziellos. Verlasst eure Position nicht nur, um den Platz zu wechseln!
2. Befindet sich der Ball im Besitz der eigenen Mannschaft, dann lauft euch frei, um ein Abspiel zu ermöglichen!
3. Sucht den „freien Raum“, um euch für ein Abspiel anzubieten! Seid ihr im Ballbesitz, müsst ihr rechtzeitig den Spieler erkennen, der in einen „freien Raum“ läuft und ihm den Ball zuspielen!

2.3.2_Teamtaktiken

2.3.2.1_Abspiel (Zusammenspiel)

Bestimmt wird das Abspiel vom Können der einzelnen Spieler ohne Ball sowie von der Genauigkeit des Abspiels. Die verschiedenen Bezeichnungen des Abspiels beziehen sich auf die Entfernung (kurze, weite, lange Pässe) oder auf die Richtung des Abspiels (Quer-, Steil- oder Schrägpass).

Merksätze:

1. Spielt den Ball dem sich am klügsten anbietenden Mitspieler so zu, dass er ihn ohne Zeit zu verlieren und ohne seinen Lauf zu stoppen annehmen, weiterführen, abspielen oder aufs Tor schießen kann!
2. Zweckmäßig ist es, lange und kurze Pässe, flaches oder hohes Zuspiel, Quer- und Steilpässe abwechselnd anzuwenden. So kann der Gegner sich auf keine bestimmte Abwehr vorbereiten.
3. Stellung des Gegners zu den eigenen beachten!

Typische Fehler:

1. Ball hinter den Lauf des Mitspielers abgeben.
2. Abspiel an einen gedeckten Mitspieler.

2.3.2.2_Stellungsspiel

Das richtige Stellungsspiel erfordert viel Aufmerksamkeit und gute Spielübersicht, aber auch genaue Kenntnis aller Spielregeln. Ein kluges Stellungsspiel ist die Grundlage erfolgreicher Handlungen im Angriff, so beim Platzwechsel und Freilaufen oder in der Verteidigung beim Stören, bei der Ballabnahme und beim Decken.

Merksätze:

1. In jedem Augenblick des Spiels muss die Position des Spielers so gewählt sein, dass er die grössten Aussichten hat, den Ball zugespielt zu bekommen, ihn zu erreichen oder in Besitz zu nehmen bzw. dem Gegner den Weg zum Tor zu versperren!
2. Stellt euch in der Abwehr so zwischen dem eigenen Tor und dem Gegner auf, dass ihr euch in einer gedachten Linie zwischen der Mitte des Tores und dem Gegner befindet!
3. Beachte beim Stellungsspiel nicht nur den Gegner, sondern auch die Stellung der eigenen Mitspieler, um sie bei plötzlichen und unvorhergesehenen Aktionen des Gegners zu sichern!
4. Die Entfernung des Balls vom eigenen Tor, die Schnelligkeit eurer Gegenspieler und ihr technisches Können bestimmen, wie weit entfernt ihr vom Gegner stehen müsst!

3_SPIELERERKENNUNG

„Es gibt nur einen Ball. Wenn der Gegner den Ball hat, stellt sich die Frage: Warum hat er den Ball?“

Giovanni Trapattoni

3.1_Überblick

Für diese Arbeit ist das Modul Gegnermodellierung (PlayersLocator in der Version DDD2005PlayersLocator[6]), ein Spezialist für die Spielererkennung GT2005PlayersSpecialist im ImageProcessor (GT2005ImageProcessor) und Behavior InputSymbols, PassSymbols und PlayersSymbols entwickelt worden.

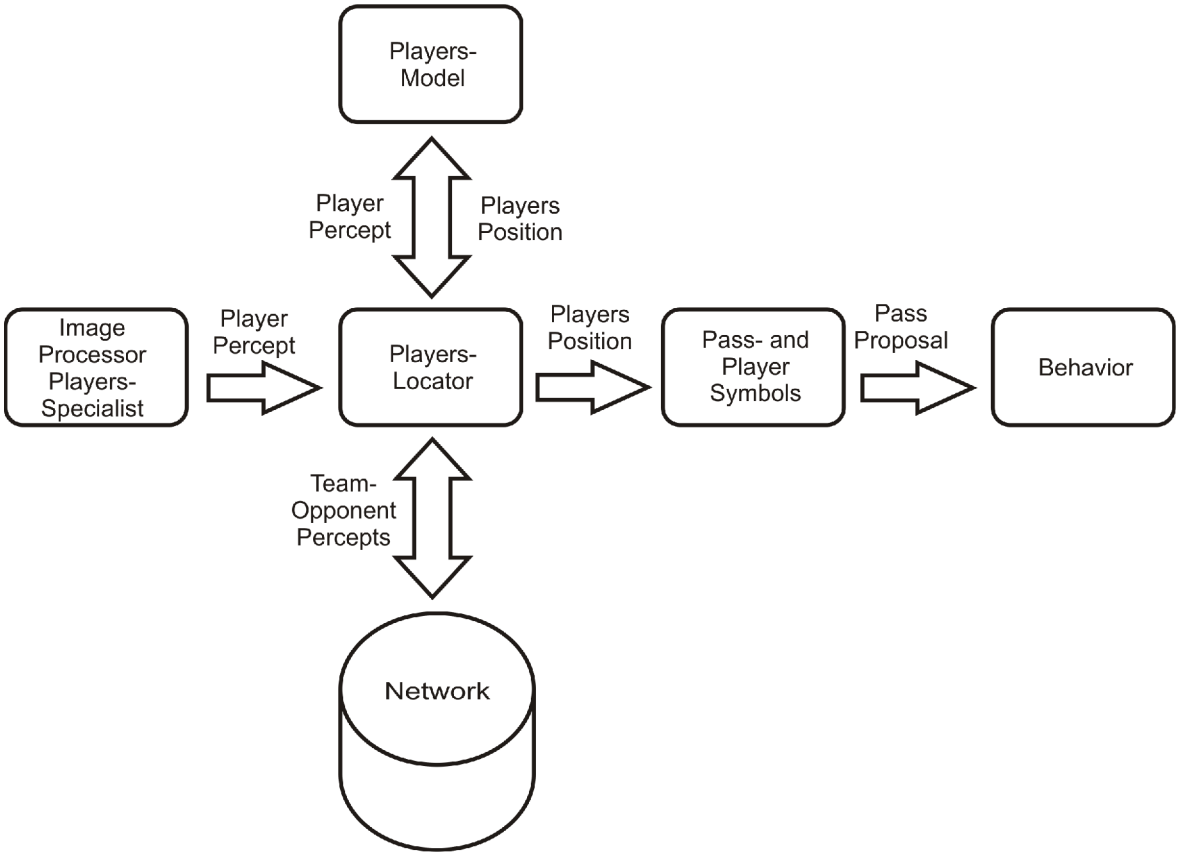


Abbildung 8: Übersicht der Module und der zugehörige Informationsfluss

Die bereits vorhandene Teamkommunikation wurde so erweitert, dass sie die gesammelten Informationen des einzelnen Roboters über die Sichtung von Gegnern an die anderen Mitspieler weitergibt. Das entwickelte Verhalten ermöglicht es, die aus dem globalen Weltbild entwickelten Pässe umzusetzen.

6 Im GermanTeam werden die Module nach den jeweiligen Teams und dem Jahr der Entwicklung benannt. Nach den nationalen Meisterschaften werden die besten Module zusammengeführt für den Einsatz auf der RoboCup Weltmeisterschaft vorbereitet.

3.2_Ausgangsbasis

Die zuvor eingesetzte Erkennung der Spieler basierte ausschließlich auf den Trikotelementen des Roboters. Um aber nicht das gesamte Kamerabild nach farbigen Pixel zu durchsuchen, werden Scanlinien eingesetzt. Die Scanlinien sind Teile eines Rasters, welches um den Horizont herum engmaschiger, unterhalb und oberhalb des Horizonts weiter ist. Der Horizont ist eine gedachte Ebene auf der Höhe der Kamera, parallel zum Boden. Die Höhe der Kamera wird durch die Stellung der Gelenke berechnet. Weit entfernte Objekte sind kleiner im Kamerabild und liegen näher am Horizont. Nahe Objekte erscheinen größer und nehmen mehr Fläche im Bild ein. Sie sind somit leichter zu erkennen und benötigen daher kein so feines Raster.

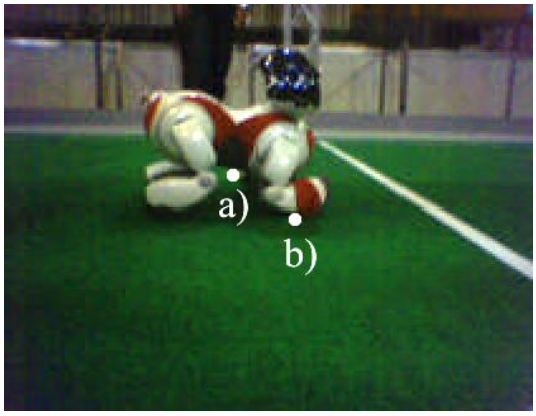


Abbildung 9: a) zeigt den gefunden Bodenpunkt unterhalb des Körpertrikots, b) bezeichnet den tatsächlichen Bodenpunkt des Roboters.

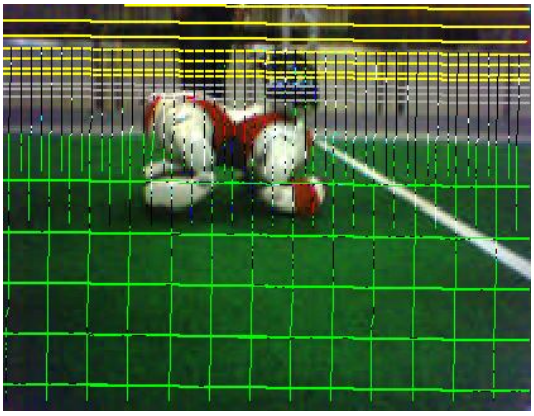


Abbildung 10: Kamerabild mit Scanlines

Auf Basis der Scanlinien wurden LinePercepts der Robotertrikots erzeugt. Diese wurden dann geclustered und die Entfernung des Roboters errechnet [11]. Ein LinePercept ist ein vertikaler Abschnitt einer Scanline und wird verallgemeinert durch den oberen und unteren Pixel, und einem LinieTyp (Tor, Ball, Gegner, etc.) charakterisiert. Um die Position zu errechnen, wurde der erste grüne Pixel unter dem farbigen LinePercept als Bodenpunkt des Roboters angenommen. Da die Position der Kamera, also Blickrichtung und Höhe über dem Feld bekannt ist, kann man die Entfernung zum Gegner errechnen. Zu jedem LinePercept wurde die Entfernung berechnet und aus dem nahegelegensten ein PlayersPercept erzeugt.

Dies geht von der Überlegung aus, dass das Fußtrikot in der Menge der LinePercepts liegt und damit der Bodenpunkt des Roboters gefunden worden ist. Falls dies nicht der Fall ist und nur Körpertrikots gefunden wurden, das sich etwa 5-10 cm über dem Boden befindet, ist der erste grüne Punkt unter einem LinePercept nicht der Bodenpunkt sondern ein Punkt weiter hinter dem Roboter (Punkt a), Abbildung 9). Denn der Beobachter sieht zwischen die Beine des Roboters hindurch und sieht somit einen Punkt weit hinter dem eigentlichen Bodenpunkt des Roboters (Punkt b), Abbildung 9).

Da das Fußtrikot bedingt durch die geringe Größe eher selten erkannt wird, ist diese recht genaue Entfernungsschätzung leider unbrauchbar. Eine andere Möglichkeit war die Erkennung des Körpertrikots. Der damalige Ansatz arbeitete folgendermaßen: Es wurde angenommen, wenn ein Trikotelement mit mehr als 20 trikotfarbigen Pixeln erkannt wurde, dass es sich hierbei um ein Körpertrikot handelt. Da dieses sich über dem Boden befindet, wurde der unterste trikotfarbene Pixel als Bodenpunkt angenommen. Um die Höhendifferenz auszugleichen, wurde die Kameramatrix um 10 cm auf der z-Achse verschoben.

Das damals eingesetzte Verfahren hatte leider den Nachteil, dass die Ergebnisse stark verrauscht waren und bis zu 40-80 cm sprangen. Erkennungen auf einem verrauschten Hintergrund, wie er durch die nun fehlende Bande auftritt, waren nicht zu gebrauchen und konnten auch nicht mehr durch die Gegnermodellierung aufgefangen werden. Aus diesem Grund wurde die Spielererkennung grundlegend überarbeitet.

3.3_Erkennung

Um eine exakte Erkennung zu erhalten, wurde in dieser Arbeit ein anderen Ansatz gewählt. Als Ausgangsbasis wurden die erkannten LinePercepts genutzt, die aus den Scanlinien [11] des ImageProcessor entstehen. Ausgehend davon wird versucht, eine Signatur des Trikots zu erstellen und den Bodenpunkt zu finden. Es handelt sich hierbei also eigentliche um eine Gegnererkennung, da im Spiel nur nach der gegnerischen Trikotfarbe gesucht wird, um keine Rechenzeit zu verschwenden. Denn schließlich erhält man über die TeamMessages die Position der eigenen Spieler, deren Positionsbestimmung besser ist, als die selbst erkannte.

Insgesamt besteht der Prozess der Spielererkennung aus den folgenden vier Schritten: Clustering, Beine finden, Bodenpunkt finden und Perzept erzeugen.

3.3.1_Clustering

Aus den LinePercepts wird der mittlere Pixel, also der Pixel der in der Mitte zwischen dem obersten und dem untersten erkannten Pixel liegt, errechnet und eine Verbindungslinie zwischen dem äußersten linken und dem äußersten rechten LinePercept über die mittleren Pixel gezogen. In Abbildung 13 ist die Verbindungslinie schwarz eingezeichnet, da hier eine detailierte Bearbeitung der Bilddaten erfolgt.

Tritt nämlich dabei eine Pixeldistanz größer als 20 Pixel zwischen dem n-ten und dem n+1-ten LinePercept auf, so wird überprüft, ob sich auf der direkten Linie zwischen den beiden mittleren Pixeln der LinePercepts Weiß, „keine“ Farbe[7] oder die Trikotfarbe befindet. Zusätzlich sind noch die Farben Pink und Himmelblau erlaubt, da diese nahe den Trikotfarben Rot und Blau in der Farbtabelle liegen. Falls dies nicht der Fall ist, also sich dazwischen Grün oder zuviel „keine“ Farbe befindet, wird ein neues Segment angefangen, da vermutet wird, dass es sich um zwei verschiedene Roboter handelt. Der Grund für den Abbruch bei zuviel „keine“ Farbe, liegt darin, dass es sich um Pixel handelt, die von Objekten außerhalb des Spielfelds stammen könnten.

Abbildung 11: Unbearbeitetes Bild

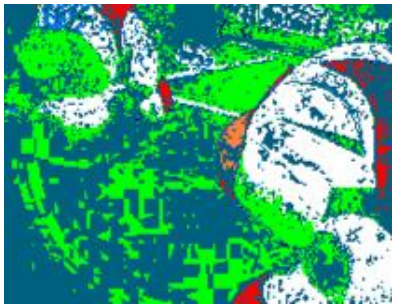


Abbildung 12: Segmentiertes Bild

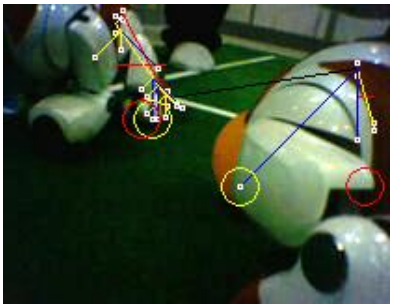


Abbildung 13: Bild mit Debuginformationen

7 „keine“ Farbe bezeichnet eine nicht zu einer Farbklasse [11] zugeordnete Farbe in der Farbtabelle.

3.3.2_Beine finden

In diesem Schritt wird ausgehend von den Erkenntnissen im ersten Schritt versucht, die Beine zu finden. Hierzu wird zwischen dem linken und dem rechten LinePercept auf der Horizontalen nach der Farbe Weiß der Beine gesucht. Beim ERS-7 handelt es sich um Weiß. Es ergeben sich drei denkbare Situationen.

1. Zwischen den linken und dem rechten LinePercept befindet sich nur die Trikotfarbe. Dies ist der Blick auf die Vorder- bzw. Hinterseite des Roboters.

2. Zwischen dem linken und dem rechten LinePercept befindet sich ein weißer Abschnitt. Dies ist der Blick seitlich schräg auf den Roboter.

3. Zwischen dem linken und dem rechten LinePercept befinden sich zwei weiße Abschnitte. Dies ist der Blick seitlich auf den Roboter.

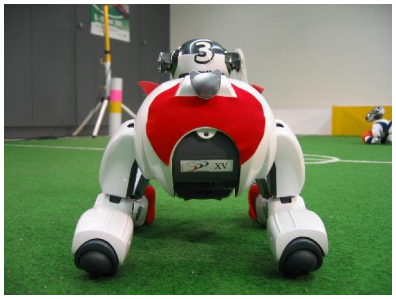


Abbildung 14: Fall 1. Heckansicht.

Abbildung 15: Fall 2. Schräge Ansicht.

Abbildung 16: Fall 3. Seitliche Ansicht.

Für die Erkennung sind nur die Fälle 1. und 2. interessant, da Fall 3. vom zweiten mit abgedeckt werden kann. Ausgehend von den gefundenen Punkten, nämlich erster trikotfarbiger, erster weißer, zweiter weißer und zweiter trikotfarbiger Pixel in der Horizontalen werden nun Suchstrahlen in Richtung Rasen geschickt, da vermutet wird, dass sich unter diesen Punkten Beine befinden, die wiederum auf dem Rasen enden.

3.3.3_Bodenpunkt finden

Um aufwendige Berechnungen zu vermeiden, werden die Strahlen jeweils diagonal nach rechts unten, gerade nach unten und diagonal nach links unten in Bildkoordinaten ausgesandt. Somit ist kein Bresenham notwendig, der viel Rechenzeit benötigt. Ein Strahl wird solange rekursiv in seiner Richtung verfolgt, solange kein Grün auftritt. Sobald Grün auftritt, werden wieder in die drei Richtungen Strahlen ausgesandt, bis die letzte Bildzeile erreicht ist oder alle Strahlen auf einem grünen Punkt enden. Die Koordinate des Punkts wird dann zurückgeben. Der Algorithmus arbeitet rekursiv und wird – um einen Stack Overflow zu vermeiden – in der 5. Rekursion abgebrochen.

Aufgrund der Breite eines geclusterten Roboters wird abgeschätzt, wie viele Pixel es bis zum Fußpunkt sein müssten. Mithilfe dieser Schätzung wird bis zur Hälfte der Höhe des Roboters jeweils ein Pixel übersprungen, und erst in der unteren Hälfte jeder Pixel betrachtet. Dies beschleunigt die Suche erheblich, da die meisten der gestarteten Strahlen schnell aus dem Roboterumriss herauslaufen. In den Bildern mit Debugdrawings sind die Suchstrahlen, die vollständig mit jeweils einem übersprungen Pixel abgelaufen wurden, gelb eingezeichnet.

3.3.4_Perzept erzeugen

Aus der Menge der gefundenen Bodenpunkte wird der tiefste ermittelt und das Mittel auf der Horizontalen zwischen dem erkannten Anfang und dem Ende des Roboters als Fußpunkt des Roboters angenommen.

Abbildung 17: Unbearbeites Kamerabild

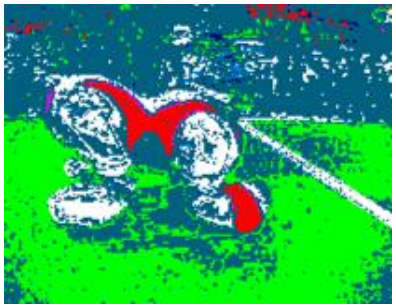


Abbildung 18: Segmentiertes Kamerabild

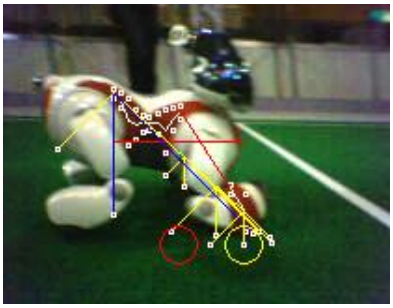


Abbildung 19: Bild mit Debuginformationen

Diese Aufnahmen stammen von einem Logfile der WM 2005 in Osaka/Japan. In Abbildung 19 liegt der tiefste erkannte Bodenpunkt im Mittelpunkt des gelben Kreises. Die rote Linie ist das Mittel der Horizontalen der erkannten Anfangs- und Endpunkte. Der errechnete Mittelpunkt des Roboters ist mit dem roten Kreis markiert.

Um die Ausrichtung des Roboters abzuschätzen, wird der Winkel der Geraden zwischen dem linken und rechte LinePercepts und dem Horizonts berechnet. Steht der gegnerische Roboter schräg zum beobachteten Roboter, liegt der Winkel im Bereich von 15° - 40°, da das erkannte Fußtrikot die Gerade neigt. Ist der Winkel größer 40°, so steht der Roboter seitlich. Zeigt der Roboter seine Front oder sein Heck, so ist der Winkel kleiner 15°.

Um zu unterscheiden, ob die Vorderseite oder die Hinterseite beobachtet wird, wird die durchschnittliche Höhe der mittleren Pixel der LinePercepts errechnet (rote Linie in Abbildung 47). Liegt diese nahe dem Horizont (weiße Linie) oder darüber, so beobachtet man die Hinterseite. Liegt sie weit darunter, so handelt es sich um die Vorderseite.

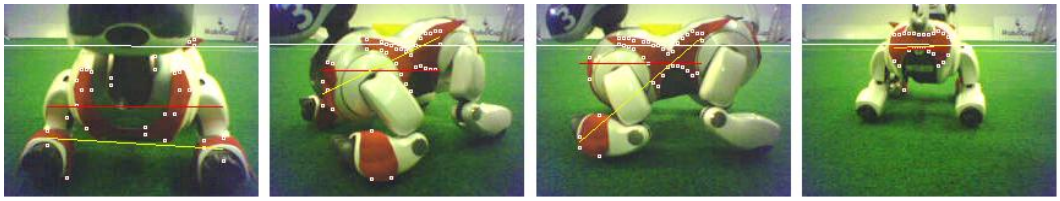


Abbildung 20: Die weiße Linie zeigt den Horizont, die rote die durchschnittliche x-Höhe der LinePercepts und die gelbe die Verbindung zwischen ersten und letzten LinePercept.

Allerdings ist dabei der Schatten des Kopfs ausschlaggebend. Schaut der Kopf zur Seite, dann werden mehr Bildpunkte in der oberen Hälfte des Brusttrikots erkannt. Dies stört die Entscheidung, ob es sich um die Vorderseite oder die Hinterseite handelt.

Um zu entscheiden, ob der Roboter von schräg vorne oder schräg hinten beobachtet wird, kann man den Schnittpunkt der Geraden der mittleren Höhe der LinePercepts und der Geraden des ersten und letzten LinePercept in Betracht ziehen. Dieser wandert je nach Blickwinkel über die Gerade der mittleren Höhe der LinePercepts, wie in den folgenden Bilder zu sehen ist.

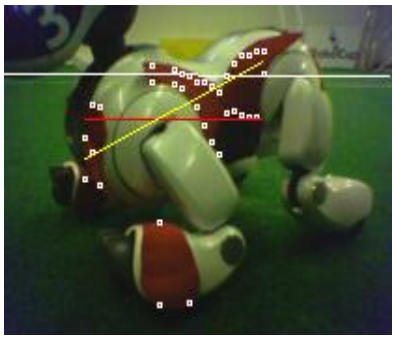


Abbildung 21: Schräg vorne

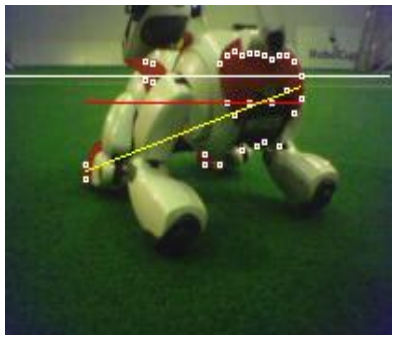


Abbildung 22: Schräg hinten

Da die Ausrichtung von der Modellierung zur Zeit nicht genutzt wird, sind diese Berechnung nicht im Code enthalten, da sie nur wertvolle Rechenzeit kosten würden.

3.4_Qualität und Aussagekraft

Die Spielererkennung arbeitet in einem Bereich von 20 cm bis 150 cm Entfernung zuverlässig. Bei näheren Robotern, kann die Kamera den Roboter nicht von Kopf bis Fuß erfassen. Das führt dazu, dass der Bodenpunkt nicht richtig errechnet werden kann, da Teile des Fußes außerhalb des Bildes liegen. Die Erkennung wird an der untersten Bildzeile abgebrochen und diese als tiefster Punkt angenommen. Bei weiter als 150 cm entfernten Robotern sind meistens nicht ausreichend farbige Pixel des Trikots vorhanden, um eine weitere Auswertung auszuführen. Hinzu addiert sich noch der Fehler der Entfernungsberechnung durch den schwankenden Horizont[8]. Zwar wird der Roboter oft von den Scanlinien gefunden, doch erbringt die weitere Auswertung keine nennenswert besseren Ergebnisse. Außerdem ist die Wahrscheinlichkeit hoch, dass es sich um Störpixel handelt, die außerhalb des Spielfelds liegen. Dieses Manko wird aber durch die kommunizierten Perzepte ausgeglichen, da diese mit in die Gegnermodellierung einfließen.

8 Der Horizont wird durch die Stellung der Gelenke des Roboters kinematisch errechnet. Da die Gelenke Spiel aufweisen sowie durch Erschütterungen beim Laufen treten Messfehler auf.

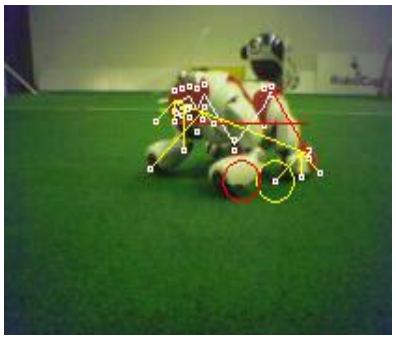


Abbildung 23: Entfernung 50cm

Abbildung 24: Entfernung 100cm

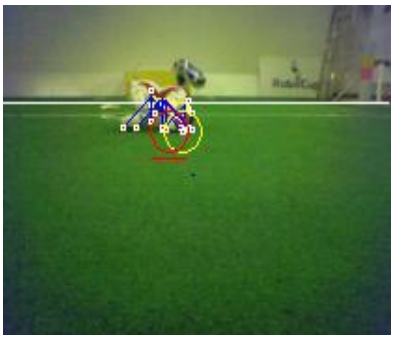


Abbildung 25: Entfernung 150cm

Die Qualität der Ausrichtung ist bei nahen Robotern gut, allerdings schwindet die Erkennungsrate der Ausrichtung mit zunehmender Entfernung, da kein Fußtrikot mehr erkannt wird und somit sich die Gerade des ersten und letzten LinePercept nicht mehr neigt. Ob der Roboter nun mit dem Vorder- oder Hinterteil zum beobachtenden Roboter steht, ist stark anhängig von der Schatteneinwirkung des Kopfes und ist deswegen nicht sonderlich aussagekräftig. Es fehlt an einer Unterscheidung, ob das Heck oder die Frontl betrachtet wird. Solange diese Möglichkeit nicht gegeben ist, müsste dies dann die Modellierung kompensieren, die aber zur Zeit die Ausrichtung nicht berücksichtigt. Zum einen ist dies der Fall, da die Gegnermodellierung zeitlich vor der jetzigen Spielererkennung entstand, weil zu Beginn versucht wurde, mit der damaligen Erkennung zu arbeiten, diese sich aber dann als unzureichend herausgestellt hat. Zum anderen hat sich erst spät gezeigt, dass die Ausrichtung überhaupt bestimmbar ist.

3.4.1_Einfluss der Trikotfarbe

Da Rot nur als Trikotfarbe vorkommt und keine anderen Elemente im Spiel markiert, sind die Störungen in der Erkennung minimal. Allerdings muss das Pink der Landmarken bei der Suche nach dem Bodenpunkt miteinbezogen werden, da es in der Farbtabelle in der Nähe von Rot liegt und bei nicht optimalen Lichtbedingungen auch auf dem Trikot erkannt wird. Dass eine Landmarke als Gegner erkannt wird, scheidet aus, da eine Landmarke immer oberhalb des Horizonts liegt und ein Spieler immer unterhalb bzw. darauf.

Das blaue Trikot ist leider nicht so einfach von dem himmelblauen Tor und dem dunklen Hintergrund außerhalb des Spielfeldrands zu unterscheiden. In beiden Fällen treten eher vereinzelte blaue Pixel auf als flächige blaue Bereiche.

Um diese Fehlerkennungen möglichst zu reduzieren, werden die erkannten LinePercepts auf ihren Zusammenhalt geprüft. Dies geschieht beim Clustern (siehe 3.3.1). Dabei weist der Hintergrund oder das himmelblaue Tor keinen hohen Zusammenhalt auf und wird deswegen aus der weiteren Analyse herausgenommen. Das Verfahren hat sich bewährt und reduziert die Anzahl der Fehlperzepte drastisch. Nachteil des Verfahrens ist, dass teilweise ein naher Roboter in separat wahrgenommene unterteilt wird und somit für beide Teile eine weitere Analyse notwendig ist. Daraus entstehen dann zwei PlayersPercepte, die an die Gegnermodellierung weitergeben werden. Da aber beide Perzepte nah aneinander liegen, ist dies nicht weiter tragisch und wird in der Modellierung schnell kompensiert.

3.4.2_Robustheit

Um dem Leser möglichst anschauliche Bilder zu zeigen, wurde auf Bilder von Robotern in blauen Trikots verzichtet, da diese sich nicht gut vom Hintergrund abheben. Um aber zu zeigen, dass die Spielererkennung auch mit blauen Trikots arbeitet, folgt hier ein worst-case Beispiel:

Abbildung 26: Unbearbeitetes Bild

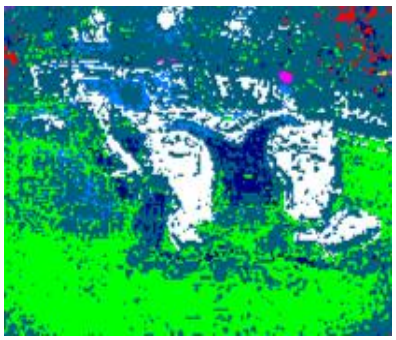


Abbildung 27: Segmentiertes Bild

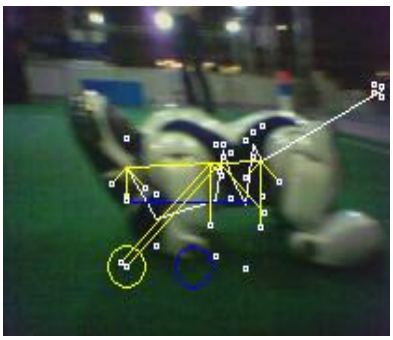


Abbildung 28: Bild mit Debugdrawings

Dieses Bilder stammen aus einem Logfile mit der vor ort aktuellen Farbtabelle von der RoboCup WM 2005 in Osaka/Japan. Wie man sieht, findet der Algorithmus auch in diesem stark gestörten Bild präzise den Bodenpunkt des Roboters.

3.4.3_Schatten

Bei einem nicht optimal ausgeleuchteten Bild werfen die Oberschenkel Schatten auf das Körpertrikot. Dies kann bei einer Ansicht von schräg vorne bei einer mittleren Distanz zu einer Nichterkennung führen, da das Brusttrikot durch den Schattenwurf des Kopfes selten auf mittlere Distanz erkannt wird. Da dann nur noch das Körpertrikot übrig bleibt, ist die Gesamtzahl der LinePercepts zu gering und muss dann verworfen werden. Ein anderer negativer Effekt durch den Schatten des Kopfs tritt bei nahen Robotern auf. Durch den Schatten ist die Anzahl der trikotfarbigen Pixel auf dem Brusttrikot zu gering und führt dazu, dass beim Clustering die LinePercepts in zwei Gruppen unterteilt werden und daraus dann auch zwei Roboterperzepte entstehen. Dieser Effekt wird aber durch die Modellierung kompensiert, da nur das jeweils näheste Perzept in die Modellierung eingeht (siehe 4.8.3).

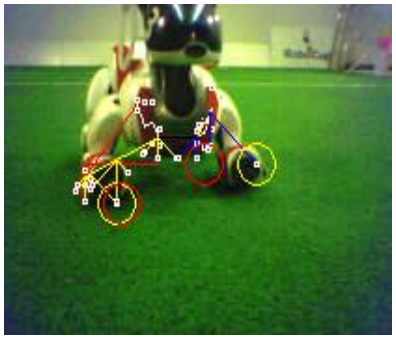


Abbildung 29: Schatten - Kopf gerade

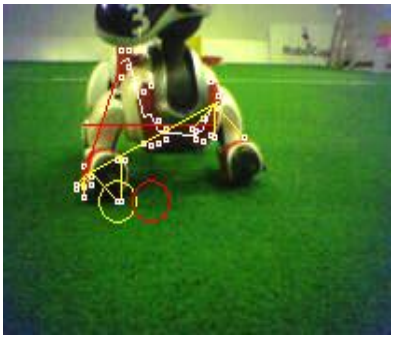


Abbildung 30: Schatten - Kopf links

3.5_Rechenzeit

Nach einigen Optimierungen, die vor allem die weitere Analyse eines möglichen Roboters schon bei bereits ausreichend guten Ergebnissen abbrechen und damit viel Zeit sparen, liegt die Rechenzeit bei weniger als 0,3 ms pro Roboter (siehe 6.2). Diese Zeit ist natürlich abhängig von der Nähe des erkannten Roboters zur Kamera. Je größer die farbige Fläche, desto mehr Pixel müssen untersucht werden.

4_GEGNERMODELLIERUNG

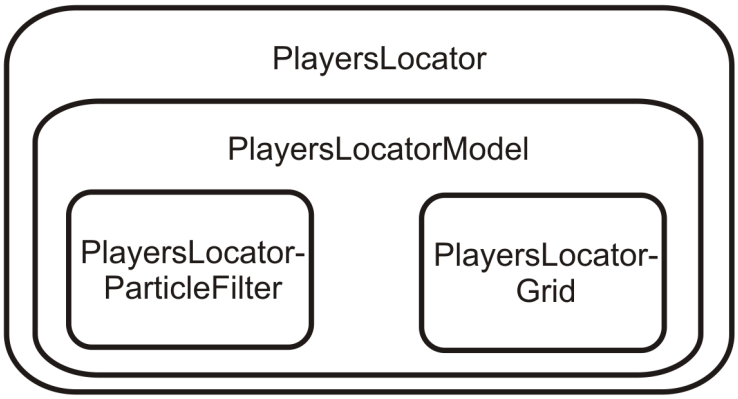
„Beim Fußballspielen verkompliziert sich alles durch das Vorhandensein der gegnerischen Mannschaft“

Jean-Paul Sartre

4.1_Einleitung

Die Gegnermodellierung ist aufgeteilt in zwei separate Funktionseinheiten. Die erste Einheit ist der PlayersLocator. Dies ist ein eigenständiges Modul und durch eine Konfigurationsdatei an- und abschaltbar [11]. Das Modul kümmert sich um die Versorgung der Modellierung mit den gesammelten Eingangsdaten und erzeugt aus den Ergebnissen der Modellierung mögliche Pässe und verwertbare Informationen für das Verhalten, wie z.B. Abstand zum nächsten Gegner zu allen Seiten.

Die Modellierung ist die zweite Funktionseinheit. Sie kümmert sich um die Errechnung der gegnerischen Spielerposition und kann ohne größere Anpassung des PlayersLocators ausgetauscht werden. So ist es möglich, eine neue Modellierung hinzuzufügen oder – mit ein paar geringfügigen Änderungen – zwei oder mehrere Modellierungen parallel zu betreiben um Vergleiche zu ermöglichen. Dies wurde genutzt, um zwei unterschiedliche Ansätze der Modellierung zu verfolgen, die hier vorgestellt werden: den Partikelansatz und den Gitteransatz.



Zeichnung 1: Aufbau des PlayersLocator Modul

4.2_Ausgangsbasis

Zuvor beruhte die Modellierung der gegnerischen Perzepte auf der Klasse ValidityAndAge. Sie basierte auf der Idee, dass gesehene Roboterpositionen eine Validität zugeordnet bekommen und diese über die Zeit hinweg altert. Die vier Positionen mit den höchsten Validitäten wurden im Anschluss aus der Menge der ValidityAndAge-Punkte (VA-Punkte) herausgesucht und als Roboterpositionen angenommen.

Durch die damalige Ungenauigkeiten der Spielererkennung wurden die Perzepte eines beobachteten Roboters um ihn herum gestreut. Daher wird eine Menge an VA-Punkten erzeugt, die um die tatsächlichen Positionen des Roboters herumliegt. Aus dieser Menge wird eine Mittelpunkt und eine Abweichung errechnet. Bewegt sich der Roboter nun aus diesem Bereich heraus, so nimmt der Anteil der neu hinzukommenden VA-Punkten an der Mittelpunktberechung nur langsam zu. Ergebnis ist, dass die errechnete Position der eigentlichen hinterher hinkt. Ein weiteres Problem tritt auf, wenn zwei Roboter sich nahe stehen: Sie verschmelzen zu einem Roboter mit einer hohen Abweichung. Da alle Punkte konstant alterten, entstand eine Spur von der aktuellen Position hin zum ältesten VA-Punkt. Dies bildet dann eine Art „Mauer“ aus Gegnern.

4.3_Ablauf des PlayersLocator

Zuerst werden dem Spielermodell die eigenen und im Anschluss die von den Mitspielern erkannten PlayersPercepte hinzugefügt. Daraufhin wird die Auswertung gestartet und die gegnerischen Spielerpositionen in die PlayersPoseCollection eingetragen. Dann erfolgt die Berechnung der Pässe und die Bewertung dieser sowie der Versand der gewonnen Informationen an die Teammitglieder.

4.4_Überblick

Die Aufgabe der Modellierung ist es, die Perzepte zu filtern und die kommunizierten gegnerischen Perzepte in Relation zu setzen. Aus diesen Eingangsdaten werden die möglichen gegnerischen Positionen errechnet.

Generell führen beide Ansätze – Gitter- und Partikel- – die gleichen Schritte aus; nur die Repräsentation ist unterschiedlich. Zuerst werden die gewonnenen relativen Perzepte mit Hilfe der Selbstlokalisierung auf eine absolute Position auf dem Feld umgerechnet. Dann werden sie in die Modellierung eingefügt und im Anschluss wird das Modell gefiltert sowie die Maxima gesucht. Im nächsten Schritt werden die möglichen Positionen gealtert.

4.5_Informationsquellen

Das Modell erhält seine Informationen primär aus den Gegnerperzepten. Dies sind zum einen die eigenen, zum anderen auch die der Mitspieler. Allerdings gehen letztere nicht mit der gleichen Gewichtung wie die eigenen Perzepte in das Modell ein. Der Grund liegt in der Zeitverzögerung durch das Netzwerk und in dem Fehler der Positionsbestimmung. Denn liegt bei dem sendenden und dem empfangenden Roboter ein Fehler in der Selbstlokalisierung vor, addieren sich diese auf und bilden erkannte Roboter an falschen Positionen ab. Zur Zeit werden die von anderen Mitspielern erkannten Roboter zu 90 % gewichtet, im Gegensatz zu den eigenen, die mit 100 % in das Modell eingehen.

4.5.1_PlayersPercepte und ObstaclesModel

Das ObstaclesModel [13] arbeitet auf Basis erkannter Grünflächen. Dazu werden Strahlen ausgehend vom unteren Bildrand ausgeschickt, bis sie einen nicht grünen Punkt erreichen – kurze weiße Pixelabschnitte werden wegen der weißen Spielfeldlinien ignoriert. Der nicht grüne Pixel wird als Hindernis angenommen und der relative Abstand zum Roboter berechnet. Das erhaltene ObstaclesModel gibt eine gute Schätzung der Hindernisse im Umfeld des Roboters ab. Die Kombination aus positiver Information (Spielererkennung) und negativer Information (kein Hindernis erkannt) kann der Gegnermodellierung helfen, die Qualität der errechneten gegnerischen Positionen zu verbessern bzw. Geisterroboter, die durch Fehler in der Selbstlokalisierung auftreten, zu entfernen (siehe 4.8.5).

4.5.2_Selbstlokalisierung

Da die Gegnermodellierung ein absolutes Weltmodell ist, also die Darstellung der Gegner nicht relativ zum Roboter erfolgt, ist sie direkt abhängig von der Selbstlokalisierung. Ist die eigene Position falsch, so werden alle relativen erkannten Gegner auf einer falschen Position abgebildet. Daher ist das Spielermodell nur dann vollständig gültig, wenn die Selbstlokalisierung in der letzten Zeit also solange bis falsche Perzepte vollkommen veraltet sind und aus dem Modell entfernt wurden sind, gültig war.

Über die Jahre der Entwicklung des GermanTeam-Codes ist es nicht gelungen eine hinreichend genaue allgemeine Aussage über Genauigkeit der Selbstlokalisierung (Validität) zu entwerfen, die die Realität wiederspiegelt. Der Roboter kann zu fast jedem Zeitpunkt während des Spiels seine Position bestimmen, aber keine hinreichend genau Aussage darüber treffen, welche Güte die Positionsbestimmung hat. Zur Zeit wird die Validität anhand der Streuung der Partikel errechnet und auf ein Intervall zwischen 0 und 1 abgebildet. Auf Wettbewerben wird oft noch stark – indirekt oder direkt – an der Selbstlokalisation gearbeitet. Änderungen an den Parametern des Selbstlokalisationsalgorithmus führen zur Änderungen der Intervallgrenzen. Deswegen wurde nach einigen Tests auf die Verrechnung der Selbstlokalisierungsvalidität mit der der Perzepte verzichtet und es wurden nur noch die Perzeptvalidiäten[9] eingesetzt.

4.6_Modellierungsvarianten

Während der Entwicklung wurden zwei Ansätze verfolgt, ein partikelbasierter Ansatz und ein Gittermodell, die nun auf den folgenden Seiten vorgestellt werden. Der Grund für die parallele Entwicklung ist folgender: Während der Entwicklung des Partikelansatzes hat sich angedeutet, dass dieser nicht die gewünschten Ergebnisse liefern konnte (siehe 4.9). Deshalb wurde parallel die Entwicklung des Gitteransatzes begonnen, um einen einsatzfähigen PlayersLocator für die RoboCup-WM 2005 zu erhalten.

9 Dabei nimmt die Validität der Perzepte linear zur Entfernung ab.

4.7_PlayersLocatorParticleFilter

Das Partikelsystem funktioniert grob folgendermaßen: Eine konstante Menge an Partikeln, die eine mögliche Position eines Roboters auf dem Feld repräsentieren, wird nach Maxima, also einer Häufung von Partikeln durchsucht. Ein Partikel besteht aus dem Tupel (x,y) der Position und einer Wahrscheinlichkeit der Position [9]. Für weitere Details wird auf geeignete Literatur verwiesen.

Der programmtechnische Ablauf des Partikelsystems besteht aus der Gewichtung der Partikel anhand ihres Abstands zu den Perzepten, der Normalisierung der Partikel, der Maximasuche, dem Resampling der Partikel und der Streuung der Partikel.

Zu Beginn wird das Partikelfeld mit zufällig gewählten Positionen gefüllt, die alle die gleiche Wahrscheinlichkeit haben, so dass die Summe der Wahrscheinlichkeiten 1 ergibt.

4.7.1_Einfügen der Perzepten

Das Einfügen von neuen Perzepten erfolgt durch das Erhöhen der Wahrscheinlichkeiten von Partikeln in der Region des gesehenen Perzepts. Dabei wird eine gestreckte Gaußfunktion eingesetzt, da der Fehler in der Spielererkennung in der relativen Abstandsberechnung auftritt und eher selten in der relativen Winkelberechnung. Da es vorkommen kann, dass in dieser Region keine Partikel liegen, da sie alle bereits in der Region von vorhanden Partikelhäufungen sind, so werden beim Resampling zufällig einige Partikel in dieser Region eingefügt. Diese ersetzen Partikeln in der Menge der Partikel. Jedoch werden sie nicht an der gleichen Position der entfernten Partikel auf dem Feld eingesetzt.

4.7.2_Gewichtung

Das Ziel der Gewichtung ist der Abgleich der erkannten Informationen, d.h. absoluten Position der Gegner auf dem Feld, mit dem im Modell enthaltenen Informationen.

Bei der Gewichtung werden drei Schritte pro Partikel durchgeführt. Zuerst wird der Abstand zu allen Perzepten und dem aktuell zu bearbeitenden Partikel errechnet um die Wahrscheinlichkeit der Position des Partikels bewerten zu können. Denn liegt ein Partikel nahe an einem Perzept, so ist seine Positionsrepräsentation wahrscheinlicher als die eines Partikels, der weiter weg liegt. Liegt der Partikel innerhalb eines Mindestabstands von 30 cm zu einem Perzept, so wird er durch eine zweidimensionale Gaußfunktion gewichtet. Der höchste Wert der Gausfunktion liegt in der Mitte des Perzepts und flacht zu den Seiten hin ab. Zudem ist sie leicht in der Länge gestreckt, da Abweichungen des gesehen Perzepts und der realen Position des Roboters in der Abstandsberechnung auftreten und eher selten im relativen Winkel zum Roboter.

Dabei wird der Partikel zu dem Perzept hin angezogen, deren Mindestabstand er unterschritten hat. Je näher der Partikel liegt, desto stärker wird er in Richtung Perzept versetzt. Dies soll bewirken, dass sich Partikel an den Perzepten häufen. Im nächsten Schritt wird der Partikel mit der errechneten Wahrscheinlichkeit der Positionsrepräsentation mutipliziert. Im letzten Schritt wird das gesamte Partikelfeld normalisiert, so dass die Summe aller Wahrscheinlichkeiten wieder 1 ergibt.

4.7.3_Maximasuche

Um die nun vorhandenen Partikelhäufungen (Maxima) zu finden wird ein Raster der Größe 20 x 15 Zellen erzeugt und über das Spielfeld gelegt. Jeder Zelle werden nun die Partikel zugeordnet, die innerhalb dieser liegen und die Summe der Partikelwahrscheinlichkeiten innerhalb der Zelle wird errechnet. Die Zellen mit den höchsten Wahrscheinlichkeiten werden herausgesucht und in Verbindung mit den Nachbarzellen der Schwerpunkt des Partikelhaufens gesucht. Die vier wahrscheinlichsten werden dann als Spielerpositionen an den PlayersLocator weitergegeben.

Abbildung 31: Spielsituation

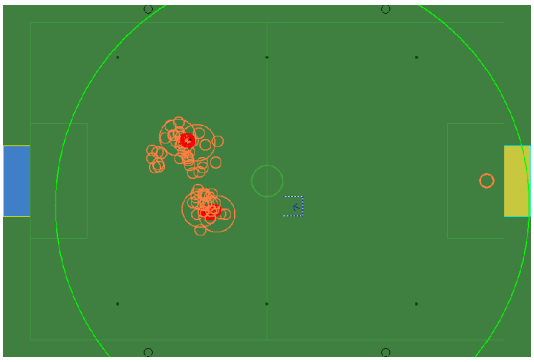


Abbildung 32: PlayersLocator Field View

Die orangen Kreise in Abbildung 32 sind die erzeugten Partikel um die Perzepte herum. Die roten Kreis stellen besonders hohe Wahrscheinlichkeiten dar, die im folgenden Resampling vermehrt werden. Die großen orangen Kreise zeigen die Region an, die von der Gaußfunktion bewertet wird. Der Mittelpunkt der großen orangen Kreise entsprechen auch den Positionen der gesehenen Perzepte.

4.7.4_Resampling

Beim Resampling werden weniger wahrscheinliche Positionen aus dem Partikelfeld entfernt und höher wahrscheinliche vermehrt. Zusätzlich werden noch zufällig Partikel an gesehenen Positionen und zufällig im Feld eingestreut. Dabei bleibt die Menge der Partikel konstant. Dies hat den Sinn, dass bei neuen Beobachtungen Partikel an dieser Stelle vorhanden sind und Partikel nicht nur um Maxima herum liegen. Denn liegt dort kein Partikel, kann er nicht bewertet werden und somit ergibt sich keine Häufung, also auch kein Maximum.

4.7.5_Streuung

Wenn für ein Maximum längere Zeit keine Informationen in Form von beobachteten Gegnern vorlagen, dann könnte sich der Gegner in irgendeine Richtung bewegt haben. Um dies im Partikelfeld widerzuspiegeln, werden die Partikel noch ein Wenig zufällig in irgendeine Richtung verschoben.

Geplant war, die Streuung ein Wenig in Richtung Ball zu richten. Denn man kann davon ausgehen, dass sich ein Gegner zum Ball bewegt. Ferne Gegner tun dies weniger stark als solche, die nah am Ball sind. Allerdings wurden dadurch viele Partikel zwischen Gegner und Ball gelegt, was dazu führte, dass der Roboter sich langsam Richtung Ball bewegte, obwohl er beobachtet wurde und nach einer gewissen Zeit wieder auf seine tatsächliche Position zurücksprang.

Dieser Effekt ist folgendermaßen zu erklären: Da alle Partikel des Haufens durch die Streuung erfasst werden, bewegt sich der Schwerpunkt zum Ball hin. Neue Partikel, die die reale Position widerspiegeln, werden hinter dem Maximum erzeugt. Diese wiegen aber nicht so schwer wie das Maximum und werden zudem noch von diesem im nächsten Schritt angezogen. Erst wenn es sich der Haufen durch die zufällige Streuung weit genug verteilt hat, haben die beobachteten Perzepte die Möglichkeit eine neue Häufung zu bilden.

Wenn man die Stärke des „magnetischen“ Effekts verringerte, können die Partikel dem Maxima nicht entkommen. Dies ist ein Beispiel für die in 4.9 beschriebenen Probleme bei der Entwicklung eines Partikelansatz.

4.7.6_Ergebnisse

Das Partikelsystem hat sich als präziser Ansatz zur Gegnermodellierung erwiesen, wenn man nur einen Gegner hat. Bei mehreren werden die Partikel von den anderen Gegnern abgezogen und „verwaschen“. Hinzu kommt, dass das Partikelfeld seit kurzer Zeit nicht beobachtete Roboter schnell vergisst.

Auf dem GermanTeam Workshop 2005 in Bremen zur Vorbereitung auf die WM in Osaka/Japan wurde von einigen Teammitgliedern[10] der Darmstadt Dribbling Dackels eine Erweiterung des Partikelansatzes umgesetzt, der die oben beschriebenen Probleme nicht aufweist. Die Idee war, dass jeder Partikel alle vier Gegnerpositionen repräsentiert, also ein vierer Tupel mit (x,y), und damit der Abzug der Partikel von einem Gegner zum anderen unterbunden wird. Leider hat es sich als nicht viel stabiler und als rechenintensiver erwiesen.

10 Zu nennen sind hier Dorian Scholz und Tobias Oberlies in Zusammenarbeit mit Max Risler.

4.8_PlayersLocatorGrid

Die Idee des Gitteransatzes ist simpel. Das Spielfeld wird in Zellen zerlegt und jede einzelne Zelle repräsentiert eine mögliche Präsenz eines Gegners. Derzeit beträgt die Kantenlänge der Rasterzellen 30 cm und das entspricht 266 Zellen. Die Kantenlänge ist so gewählt, dass ca. ein Roboter in eine Zelle passt. Allerdings tritt so das Problem auf, dass ein Roboter, der auf einer Zellengrenze beobachtet wird, immer zwischen diesen beiden Zellen hin und her springt. Dies entsteht durch kleine Variationen in der Erkennung. Um den entstehenden Positionsfehler zu verringern, kann man die Kantenlänge des Gitters auf 15 cm reduzieren. Dadurch wird zwar der Effekt des Springens verstärkt, aber dafür wird der Positionsfehler kleiner.

4.8.1_Einfügen der Perzepte

Ein neues Perzept wird hinzugefügt, indem der Wert der Zelle, die der des beobachteten Gegners entspricht, auf 1 angehoben wird. Die Werte der benachbarten Zellen werden zusätzlich leicht angehoben. Dazu werden die Werte je nach Abstand zum Perzept mit den Werten der Matrix multipliziert.

0,10	0,20	0,25	0,20	0,10
0,20	0,40	0,50	0,40	0,20
0,25	0,50	1,00	0,50	0,25
0,20	0,40	0,50	0,40	0,20
0,10	0,20	0,25	0,20	0,10

Tabelle 1: Filtermatrix Maximumbildung

Bei der jetzigen Gittergröße von 30 cm werden nur die unmittelbaren Nachbarn angehoben. Bei einer geringeren Gittergröße, wie z.B. 15 cm, muss die Schrittweite angepasst werden, bei noch kleineren Schrittweiten die gesamte Matrix.

4.8.2_Saugfunktion

Die Saugfunktion des Gitteransatzes hat analog zum Partikelsystem den Zweck, Häufungen zu bilden. Wenn ein Maximum vorliegt, dann sollen die Validitäten der angrenzenden Zellen verringert und auf das Maximum aufaddiert werden. So gesehen wird „aufgekehrt“. Perzepte, die um die eigentliche Position des Roboters entstehen werden durch Ungenauigkeiten in der Erkennung (siehe 3.4) zu dem Maximum hinzugefügt und verstärken dieses. Dazu wird die gleiche Matrix verwendet, wie auch beim Hinzufügen der Perzepte.

4.8.3_Hillclimbing

Um die vorhandenen Maxima zu finden, wird das Gitter vollständig abgelaufen. Sobald der Wert einer Zelle größer als 0,1 ist, wird ein Hillclimbing-Verfahren gestartet. Es werden die Werte der umliegenden Zellen betrachtet und in die Zelle gewechselt, die den höchsten Wert hat. Sind alle Werte kleiner, so ist das Maximum erreicht. Falls nicht, beginnt der Prozess von vorne.

Von den Ergebnissen des Hillclimbings werden nur die vier größten Werte gespeichert. Dabei wird auf ein Mindestabstand zwischen den Maxima geachtet und neu erkannte nicht gespeichert, falls diese zu dicht liegt und kleiner sind, als bereits ein bekanntes Maximum. Denn durch den in 4.8 erwähnten Effekt werden leicht zwei Maxima eines Roboters erzeugt.

Die Laufzeit des Hillclimbing-Algorithmus ist nahezu linear. Hingegen hätte an Quicksort Sortierungsalgorithmus mit der Laufzeit von n(log n) und der anschließenden Abstandsüberprüfung innerhalb der gefunden Maxima mehr Rechenzeit in Anspruch genommen.

Stehen zwei Maxima in einer Sichtlinie zum beobachtenden Roboter, so wird nur das Maximum aufgenommen, welches näher am Beobachter ist. Dies trägt zwei Situationen Rechnung:

1. Wenn ein Teammitglied, das quer zum Beobachter steht, einen Gegner wahrnimmt und dieses dem Beobachter mitteilt, aber seine eigene Position nicht exakt stimmt, dann wird verhindert, dass zwei Roboter in gleicher Linie an das Verhalten weitergeben werden.

2. Bei weit entfernten Robotern springt die Entfernungsberechnung stärker und es kommt vor, dass ein und derselbe Roboter Perzepte in unterschiedlichen Distanzen erzeugt. Für das Verhalten ist aber nur der nahe Roboter interessant.

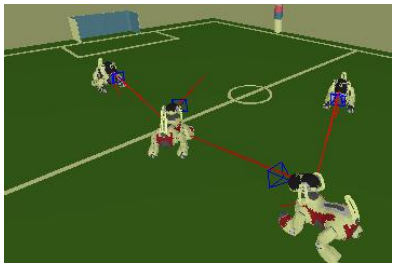


Abbildung 33: Zwei rote Roboter in einer Sichtlinie aus dem Simulator

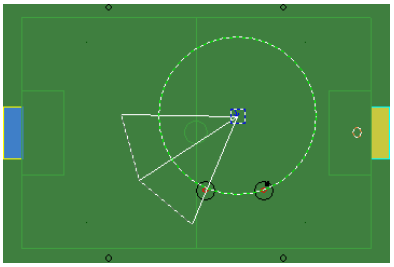


Abbildung 34: PlayersLocator View des seitlich beobachtenden Roboters

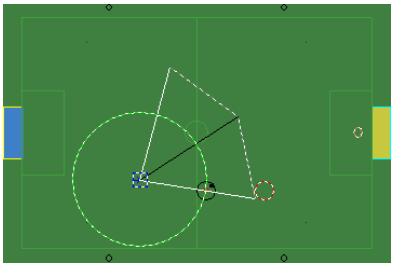


Abbildung 35: PlayersLocator View des in einer Sichtline beobachtenden Roboters

4.8.4_Aging

Im Gegensatz zum Partikelsystem, ist beim Gitteransatz das Aging die einzige Möglichkeit, Zellen aus der Suche nach Maxima zu entfernen. Beim Aging wird jede Zelle um einen Wert verringert, so dass ein voller Ausschlag innerhalb von vier Sekunden wieder Null wird. Dazu wird die vergangene Zeit zwischen dem jetzigen und dem letzten Frame dividiert durch die Zeit, die ein Roboter im Modell unbeobachtet existieren soll, berechnet und von jeder Zelle im Gitter abgezogen.

$$Z_j = \frac{t(f_{(i)}) - t(f_{(i+1)})}{L} * Z_j$$

Formel 1: Alterungsformel. Z bezeichnet die Zelle, L ihre Lebenszeit, f den Frame und t die Zeitfunktion.

4.8.5_Verbesserung durch ObstaclesModel

Um die Qualität des Gegnermodells noch zu steigern, kann das ObstaclesModel [13] eingesetzt werden. Hierbei werden die Hindernissektoren, die das ObstaclesModel errechnet, auf das Gittermodell abgebildet. Vom beobachtenden Roboter aus werden Linien für jeden gerade betrachteten Sektor mit Hilfe eines Bresenham Linienalgorithmus auf die Zellen des Gittermodells abgebildet und der Wert der Zelle reduziert. Bei einem typischen Schwenk des Kopfs – wie in den Head-Control Modien [7] searchAuto und searchForLandmarks üblich – wird jede Zelle mehrfach durch die Scanlinien des Bresenham bestrichen. Deshalb wird bei einer aus der Sicht des ObstaclesModels leeren Zelle, diese nur mit dem Faktor 0,99 multipliziert, da sonst durch das mehrfache bestreichen der Zelle, die Alterung zu stark fortschreiten würde..

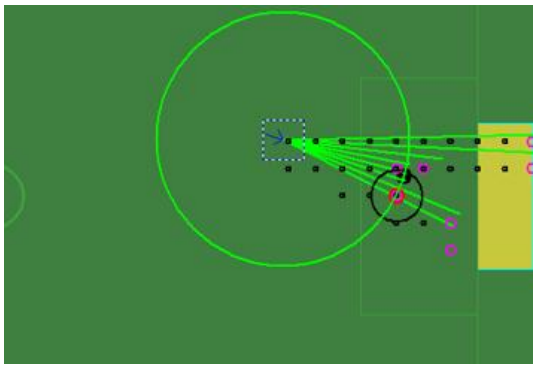


Abbildung 36: PlayersLocator Fieldview mit Einsatz des ObstaclesModel zur Verbesserung der Modellierung.

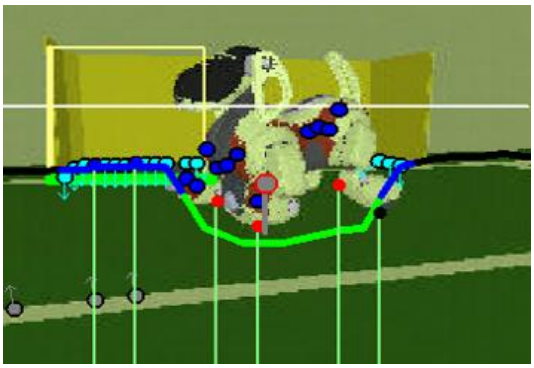


Abbildung 37: Kamerabild aus dem Simulator mit DebugDrawings.

Nachteilig ist dabei, dass durch das ObstaclesModel auch Zellenwerte reduziert werden, die tatsächlich einen Roboter repräsentieren. Oft tritt der Fall auf, dass das ObstaclesModel den freien Raum unterhalb eines Roboters erkennt. Steht ein Roboter seitlich zum Beobachter, so kann der Beobachter zwischen seinen Beinen hindurchsehen, wie man in Abbildung 37 an der zweiten grünen Linie von rechts erkennen kann. Bei einem Kopfschwenk wird dadurch der Wert der Zelle, die einen Roboter in der realen Welt wiederspiegelt, mehrfach mit dem Faktor 0,99 multipliziert. Dies kann zur Nichterkennung führen.

Da bei weiter entfernten Robotern Schwankungen in der Positionsbestimmung auftreten, führt die Anwendung der ObstaclesModel zu eher negativen Ergebnissen und wurde auch aus Gründen der Rechenzeit nicht im WM-Code 2005 genutzt.

4.9_Partikel- versus Gittermodell

Das Partikelmodell spart Speicher und Rechenzeit und ist der elegantere Ansatz. Nachteilig ist allerdings, dass die Partikel schwerer zu kontrollieren sind und somit die Entwicklung erschweren. Fügt man eine Regel hinzu um einen lokalen Effekt zu beeinflussen, so wirkt sie sich auf das gesamte Partikelfeld aus und ändert die Gewichtung der Partikel. Dies führt zu vollkommen anderen Ergebnissen. Zuvor entwickelte Regeln müssen teilweise wieder angepasst werden und beeinflussen wiederum das Verhalten der Partikel. Ein sehr negativer Effekt ist der schnelle Abzug von Partikeln von unbeobachteten Roboterpositionen bei vielen neuen Perzepten der gleichen Position. Das Partikelsystem vergisst sehr schnell. Außerdem werden durch die Ansaugfunktion der Maxima die Partikel von nah liegenden Robotern zum Teil komplett abgezogen.

Das Gittermodell teilt das Spielfeld in kleine Zellen auf. Je kleiner die Kantenlänge, desto mehr Zellen müssen „angefasst“ werden um nach Maxima zu suchen bzw. um die Alterung der Werte durchzuführen. Bei der zur Zeit verwendeten Kantenlänge von 30 cm entspricht dies 266 Quadraten. Bei 15 cm Kantenlänge sind es bereits 1066 Quadrate. Dadurch steigt die benötigte Rechenzeit des PlayersLocator von 1,0 ms pro Aufruf auf 1,6 ms auf dem SONY ERS-7.

Die Entwicklung des Gittermodells hat sich als simpler erwiesen als die des Partikelsatzes. Lokal angewandte Regeln beeinflussen nicht das ganze Modell und erlauben es im Gegensatz zum Partikelansatz, Verfeinerungen ohne einen kompletten Test einzupflegen. Das Gittermodell hat sich als robuster und stabiler erwiesen und kam deshalb beim RoboCup 2005 in Osaka/Japan zum Einsatz.

5_PASSPLANUNG, BEWEGUNGEN UND VERHALTEN

„Das Runde muss in das Eckige“

Helmut Schulte

5.1_Einleitung

Um Pässe zu spielen benötigt das Verhalten Angaben über die Qualität der möglichen Pässe zu den anderen Spielern. Grundsätzlich sind zu allen Spielern Pässe möglich, doch nicht jeder ist spielbar, da er z.B. durch einen Gegner blockiert ist oder keinen nennenswerten taktischen Vorteil erbringt.

5.2_Passberechnung

Das PlayersModel übernimmt auch die Berechnung der Pässe, da es sich bei der Entwicklung als sinnvoll erweisen hat, dies nicht als selbstständiges Modul zu entwickeln, da der Funktionsumfang eher gering ist.

Da als Passpartner nur die drei weiteren Mitspieler zur Verfügung stehen wobei ein Rückpass zum Torwart nicht wünschenswert ist, da die Passspielgenauigkeit nicht sehr hoch ist (siehe 5.6.2) bleiben zum überwiegenden Teil nur die Pässe zu den beiden anderen Feldspielern übrig.

Ein Pass ist nur dann sinnvoll, wenn man selbst den Ball besitzt und ein gegnerfreier Korridor zum Passpartner existiert. Da das Gegnermodell alle Informationen über die Position der eignen und der generischen Spieler besitzt, werden im Anschluss der Gegnermodellierung die Korridore zu den Mitspielern errechnet und bewertet.

5.2.1_Gegnerfreie Zone

Bei der Berechnung der Pässe wird auch der umgebende freie Raum des Roboters berechnet. Dieser wird im Verhalten genutzt, um zu entscheiden, ob ein Passspiel erfolgreich sein könnte oder direkt schon beim Abspiel gestört werden könnte. Auch kann dieser Wert in anderen Teilen des Verhaltens eingesetzt werden, um z.B. eine Aussage darüber zu treffen, ob der Roboter sich im „Getümmel“ befindet oder frei steht. Die Gegnerfreie Zone wird als InputSymbole free-space-around-robot (in mm) an das Verhalten weitergegeben. Die folgenden Aufnahmen aus dem Simulator zeigen anschaulich das Ergebnis der Berechnungen.

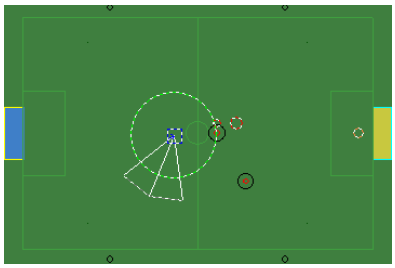


Abbildung 38: PlayersLocator View Aibo1

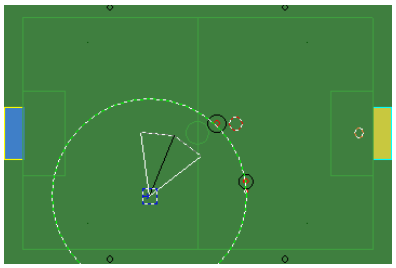


Abbildung 39: PlayersLocator View Aibo2

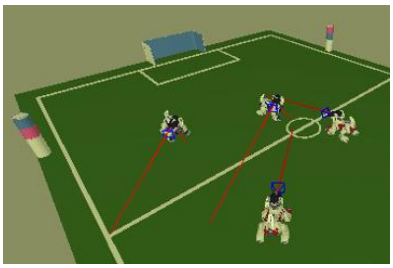


Abbildung 40: Spielsituation

Der grün-weiße Kreis um die Roboter herum bezeichnet die gegnerfreie Zone. Die beiden schwarzen Kreise mit dem roten Mittelpunkt stellen die erkannten Gegner dar. Nicht als Maximum akzeptierte Maxima sind als weiß-rote Kreise eingezeichnet (siehe 4.8.3).

5.2.2_Passkorridor-Berechnung

Um einen Korridor zu berechnen, werden die Abstände zu allen gegnerischen Robotern und der Öffnungswinkel zwischen den Geraden Passgeber-Passpartner und Passgeber-Gegner errechnet. Die Ergebnisse werden in der PassCorridorCollection gespeichert, die folgende Werte enthält:

- Öffnungswinkel
- Abstand zum Passpartner
- Abstand zwischen Passgeraden und Gegner
- Spielernummer des Passpartners

Ist der Öffnungswinkel zu allen Gegnern größer als 60°, so wird der Winkel auf 60° begrenzt.

5.3_Passbewertung

Nachdem die Passkorridore errechnet wurden, müssen die Pässe nun anhand der aktuellen Spielsituation bewertet werden. Als Ausgangsbasis für die Bewertungskriterien wurde Bezug genommen auf die Merksätze in Kapitel 2.3.

Um die Qualität eines Passspiels in einer Zahl auszudrücken, kann man viele Kriterien heranziehen. Damit steigt allerdings die Dimension des Lösungsraums und die Anschaulichkeit des Ergebnisses nimmt ab. Deshalb wurden für die Bewertungen nur die wichtigsten Kriterien herangezogen und gehen in die Passqualität mit unterschiedlichen Gewichten ein. Die Wertungen liegen in den Intervallen [0..1] bzw. [-1..1]. Nachdem alle Wertungen errechnet wurden, gehen sie nach ihren Gewichten in die Wertungssumme ein und werden im Anschluss normalisiert.

Die Wertungen „Raumgewinn“ bzw. „Entfernung zum Passempfänger“ können, im Gegensatz zu den anderen Wertungen, auch negativ sein. Denn es ist sinnvoll, eine zu hohe Entfernung zum Passpartner und damit die Wahrscheinlichkeit den Ball zu verlieren als Malus zu werten. Das Gleiche gilt auch für den Raumgewinn.

Die Berechnung der Wertung erfolgt im Spielermodell und speichert jeden Wert in der Passkorridorrepräsentation. Die Wertung des besten Korridors wird dem Verhalten als skalarer Wert zur Verfügung gestellt.

5.3.1_Raumgewinn

Der Raumgewinn bezeichnet den Vorteil den man erhält wenn man abspielt und Raum zum gegnerischen Tor gutmacht. Ist also der Abstand des Passpartner zum gegnerischen Tor geringer als der eigene, so hat man Raum gutgemacht und erhält eine positive Bewertung. Somit ergeben Querpässe keinen Raumgewinn und Rückpässe negative Ergebnisse. Berechnet wird die Differenz zwischen den Abständen Passpartner-Tor und Passgeber-Tor.

5.3.2_Passwinkel zur Spielfeldlängsachse

In der eigenen Spielfeldhälfte sind Querpässe uninteressant, da sie selten zum Spielvorteil führen. Falls der Ball beim Pass verloren geht, kann es schnell zu einem ungewollten Tor kommen. Je näher man an das gegnerische Tor kommt, desto interessanter werden Querpässe und können einen taktischen Vorteil verschaffen, wenn der Ball die Spielfeldseite wechselt und somit Verteidiger und Torwart umspielt. Daher werden Passwinkel größer 45° zur Spielfeldlängsachse mit null bewertet, für den Winkel 0° wird die 1 vergeben und dazwischen linear skaliert, falls sich der Passgeber nicht in der Nähe des gegnerischen Strafraums aufhält. Ist er in der Nähe des Strafraums, wo ein Querpass erlaubt und erwünscht ist, ergibt sich eine Bewertung von 0 bei 90° und bereits 1 bei 65°. Dazwischen wird wieder linear skaliert.

5.3.3_Entfernung zur Aussenlinie

Je mehr Raum zur Aussenlinie hinter dem Passempfänger vorhanden ist, desto unwahrscheinlicher ist bei einer verpatzten Annahme ein „Aus“ des Balls und der damit verbundene Nachteil für das Team, dass der Ball wieder in Richtung des eigenen Torraums eingesetzt wird [2]. Dabei wird der Abstand zur Aussenlinie (y Spielerposition) im Verhältnis zur halben Spielfeldbreite linear skaliert.

5.3.4_Entfernung zum Passempfänger

Die Entfernung zum Passempfänger ist ein zweischneidiges Kriterium. Ist der Empfänger nah, so bleibt kaum Zeit für den Empfänger sich auszurichten und den Ball anzunehmen. Ist der Empfänger weit weg, so kann die gegebene Streuung des Schusses und die nicht-gradlinige Laufbahn des Balls zum Ballverlust führen. Da sich 1 m Abstand zwischen Passgeber und Passempfänger als ein guter Wert (experimentell ermittelt) gezeigt hat und bei einem Abstand von 2 m der Pass oft fehlschlug, wurde eine quadratische Funktion zur Wertung genutzt, die auf das Intervall [-1..1] begrenzt :

$$f(x) = \frac{-x^2 + 2x}{1000}$$

Formel 2: Abstandsbewertung

5.3.5_Breite des Passkorridors

Je breiter der Korridor ist, in dem sich kein gegnerischer Spieler befindet, desto höher sind die Chancen auf ein erfolgreiches Passspiel. Bei 30° Öffnungswinkel und bei 1 m Abstand zum Partner ist der Korridor beim Partner für eine Ballannahme ausreichende 50 cm breit. Deswegen ist die Wertung bei 30° gleich 1. Sie nimmt linear ab bis zum Winkel vom 0° und der Wertung 0.

5.3.6_Qualität des Gegnermodells

Bei einem schlechten Spielermodell muss das Passspiel nahezu unmöglich werden, da die geplanten Pässe auf zu schlechten Daten fußen und somit der Ballverlust droht. Da das Spielermodell direkt abhängig ist von der Selbstlokalisation, hat es wenig Sinn, diesen Wert in die Passbewertung mit einzurechnen. Deswegen ist es sinnvoller, nur Pässe zu spielen, wenn auch die Selbstlokalisation stimmt. Diese Entscheidung wird im Verhalten getroffen.

5.3.7_Ergebnisse

Insgesamt ergibt sich folgende Wertung und Gewichtung für die Passkorridore:

		Wertung		Gewichtet	
Bezeichnung	**Gewichtung**	**Min**	**Max**	**Min**	**Max**
Passwinkel zur Spielfeldlängsachse	1,0	0,0	1,0	0,0	1,0
Entfernung zur Aussenlinie	0,5	0,0	1,0	0,0	0,5
Entfernung zum Passempfänger	1,0	-1,0	1,0	-1,0	1,0
Raumgewinn	3,0	-1,0	1,0	-3,0	3,0
Breite des Passkorridors	1,0	0,0	1,0	0,0	1,0
Summe	**6,5**			**-4,0**	**6,5**
Normalisiert				**-0,6**	**1,0**

Tabelle 2: Gewichtung der Passbewertung

Wie bereits erwähnt, wird die Summe normalisiert und auf das Intervall [-0,6..1] abgebildet. Durch andere Gewichtungen lässt sich die Spielweise leicht ändern. Mit diesen Werten wird recht oft ein Passspiel provoziert. Wenn man leichte Änderungen an der Gewichtung „Passwinkel zur Spielfeldlängsachse“ und „Breite des Korridors“ vornimmt, so führt dies zu einem vorsichtigeren Passspiel.

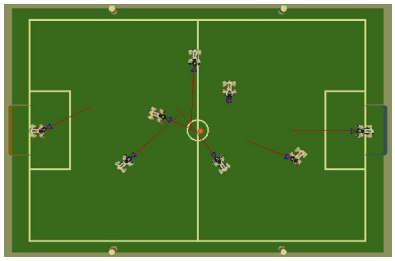


Abbildung 41: Spielsituation. Rote Spieler werden simuliert, blaue sind passiv

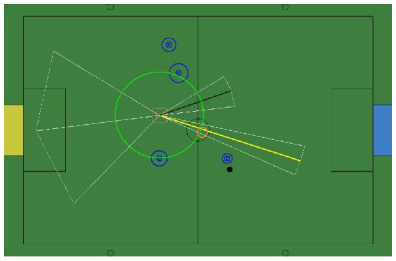


Abbildung 42: PlayersLocator FieldView Aibo1

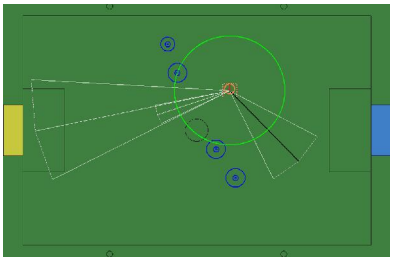


Abbildung 43: PlayersLocator FieldView Aibo2

Die in Abbildung 42 gezeigte FieldView des Spielers in der eigenen Hälfte, zeigt drei berechnete Passkorridore. Der zum Torwart ist nicht farbig und damit negativ bewertet. Der zum Mitspieler oben in der gegnerischen Hälfte ist schwarz markiert und damit als gut befunden. Der gelbe Passkorridor zeigt den Korridor, der am besten bewertet ist. Dieser wird dem Verhalten vorgeschlagen. In der FieldView von Abbildung 43 kann man sehen, das kein Pass herausragend gut ist, da der Raumgewinn und der Winkel des Passes keine Besserung der Spielsituation mit sich bringen würde.

5.4_Kommunikation

Die gewonnenen Daten eines einzelnen Spielers reichen nicht aus, um ein Gesamtbild der Spielsituation zu erstellen. Daher werden unter den Spielern die erkannten gegnerischen Roboter und die eigenen Positionen ausgetauscht. Mit einer Verzögerung von wenigen Millisekunden erhalten alle Spieler somit alle erforderlichen Daten, um ein vollständiges Spielermodell zu errechnen. Die Information über die Gegner wird mittels der schon existierenden TeamMessages übermittelt. In den TeamMessages werden neben diesen Informationen die RobotPose, SeenBallState und BehaviorTeamMessages übertragen.

Zusätzlich - zur Zeit allerdings nicht aktiviert - wird das Obstacles-Model übertragen. Dies soll es später ermöglichen, die Negativinformationen, die das ObstaclesModel über die Position von gegnerischen Robotern liefert, auch den anderen Mitspielern bereitzustellen. Dabei ist besonders das ObstaclesModel des Torwarts interessant. Denn dieser hat, von einem Feldspieler aus gesehen, immer eine rückwärtige Ansicht auf das Spielfeld und kann dazu beitragen, das Spielermodell der Feldspieler zu verbessern, da sie diesen Teil des Feldes weniger häufig betrachten. Allerdings müssen zuerst die bereits erwähnten Probleme (siehe 4.8.5) behoben werden.

5.5_Bewegungen

Um eine Ballannahme zu ermöglichen, muss natürlich die exakte Position und Geschwindigkeit des Balls bekannt sein. Das Problem der Ermittelung der Ballposition ist seit 2004 gelöst. Allerdings ist die Ermittlung der Ballgeschwindigkeit alles andere als trivial. Durch das Schwanken des Kopfs und den Ungenauigkeiten bei der Berechnung des Ballmittelpunkts, kommt es zu starken Schwankungen bei der Abstandsberechnung zum Ball.

Das Problem wurde bereits in den Jahren zuvor angegangen und die Lösung wurde immer wieder erweitert und verbessert. Der erste Ansatz zur Ermittlung der Ballgeschwindigkeit wurde von drei Teammitgliedern im Rahmen eines Praktikums der Darmstadt Dribbling Dackels, unter der Leitung des Autors dieser Arbeit entwickelt und kam 2004 auf der RoboCup-WM in Lissabon zum ersten Mal zum Einsatz [4].

Um ein Passspiel zu ermöglichen, müssen die Bewegungen der Roboter optimiert werden, vor allem die des Passempfängers. Da die Ausarbeitung dieser Bewegungen den Rahmen einer Diplomarbeit sprengen würden, wurde dies von zwei Teammitgliedern der Darmstadt Dribbling Dackels im Rahmen eines Praktikums erarbeitet und vom Autor dieser Arbeit erdacht und betreut [3].

Im folgenden Abschnitt wird deshalb nur oberflächlich auf die Bewegungen eingegangen. Der geneigte Leser wird auf den Parktikumsbericht verwiesen.

5.5.1_Bewegungen Passgeber

Für den Passgeber sind alle notwendigen Bewegungen im aktuellen Code des GermanTeams enthalten und konnten ohne Änderungen für den Passgeber einsetzt werden.

5.5.1.1_Positionierung

Wenn das Verhalten ein Passspiel auslöst, so muss der Roboter sich zuvor am Ball ausrichten, bzw. den Ball greifen und sich dann zum Passpartner ausrichten. Da der Roboter beim Ballgreifen den Kopf nach vorne neigt, ist seine Sicht auf das Spielfeld stark eingeschränkt. Daraus resultiert, dass die Lokalisierung nicht mehr möglich ist. Dies ist aber notwendig, da der Roboter meistens von anderen Spielern bedrängt wird und er nach dem Greifen nicht mehr die gleiche Position und Ausrichtung wie vorher hat. Ein weiteres Manko ist, dass eine Korrektur der Ausrichtung mit gegriffenem Ball ungleich schwerer ist, da der Roboter nur noch die Hinterbeine einsetzten kann.

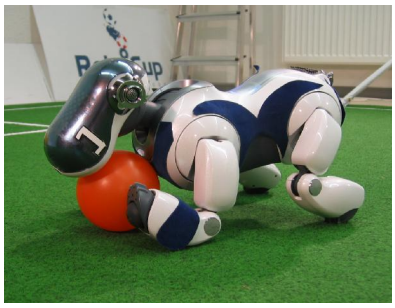


Abbildung 44: Roboter mit GrabBall

Abbildung 45: Kamerabild vor Grab

Abbildung 46: Kamerabild nach Grab

Deswegen wird auf die zuvor beschriebene Variante nur zurückgegriffen, wenn kein passender Schuss gefunden wurde und der Ball in Greifweite liegt. Ansonsten versucht der Roboter, sich zum Ball und zum Partner zu positionieren. Dazu beobachtet der Roboter den Ball, nähert sich ihm auf Schussweite und läuft seitwärts um ihn herum bis er zum Passpartner ausgerichtet ist.

5.5.1.2_Abgabe

Bei der Abgabe werden zwei verschiedene Schussarten eingesetzt. Ist der Passpartner nah, so wird auf einen etwas leichteren Schuss zurückgegriffen, der den Ball ca. 1 m weit befördert. Ist der Partner weiter entfernt, so wird der zur Zeit härteste Schuss eingesetzt, der den Ball bis zu 3m trägt. Leider ist der Schuss weniger präzise und kann – auch durch die Unebenheiten des Spielfeldrasens – zu großen Abweichungen führen, die der Passempfänger ausgleichen muss.

5.5.2_Bewegungen Passempfänger

Die Ballannahme gliedert sich in drei Sequenzen: Ausrichtung zum Passgeber, Ausgleich des Passfehlers und Ballannahme.

5.5.2.1_Ausrichtung

Wenn der Passgeber mit Hilfe der TeamMessages die Verhaltensnachricht schickt, dass eine Ballabgabe unmittelbar bevor steht, richtet sich der Passempfänger zum Passgeber aus. Das Ausrichten erfolgt so lange, bis er vom Passgeber die Nachricht erhält, dass das Abspiel erfolgt ist oder wenn der Passempfänger einen Ball auf sich zu rollen sieht (siehe 5.6.3).

5.5.2.2_Ausgleich des Passfehlers

Um den Fehler durch einen ungenauen Pass auszugleichen, wurde das Seitwärtslaufen eingeführt. Der Roboter läuft seitwärts, während er den Ball beobachtet. Ist der Fehler zu groß, also kann der Roboter die Strecke nicht überbrücken bevor der Ball an ihm vorbeirollt, bricht er den Versuch ab und geht in das normale Spielverhalten über.

5.5.2.3_Ballannahme

Das normale GT Aufstehen nach dem Block des Balls hat den Nachteil, dass dabei die Vorderbeine vor den Roboter gezogen werden um ihm besseren Halt zu geben. Liegt der Ball allerdings in der Reichweite der Roboterarme, wird er weggeschlagen.

Die neue Ballannahme wurde daraufhin entwickelt, den Ball nach dem Stoppen nicht durch eine Berührung beim Aufstehen des Roboters abzustoßen. Infolge dessen, muss der Ball während des Blocks beobachtet werden, um im Anschluss zu entscheiden, zu welcher Seite der Roboter aufsteht. Liegt der Ball dabei außerhalb der Reichweite, so wird das normale Aufstehen verwendet, welches etwas schneller ist als das entwickelte vorsichtige Aufstehen. Liegt der Ball nahe, so dreht sich der Roboter zum Ball und zieht die vorderen Beine zurück, um nicht den Ball zu berühren, und steht dann erst auf.

Für den Fall, dass der Ball vor der Brust liegt, wurde ein Aufstehen entwickelt, welches ein Wenig vom Ball zurückweicht, bevor der Roboter die Aufstehbewegung durchführt.

Die folgenden Bilder stammen aus einem Video der RoboCup-WM 2004 in Lissabon/Portugal. Das GermanTeam spielt in rot und man erkennt deutlich, dass der Ball nach der Annahme weit vom Roboter entfernt liegt.

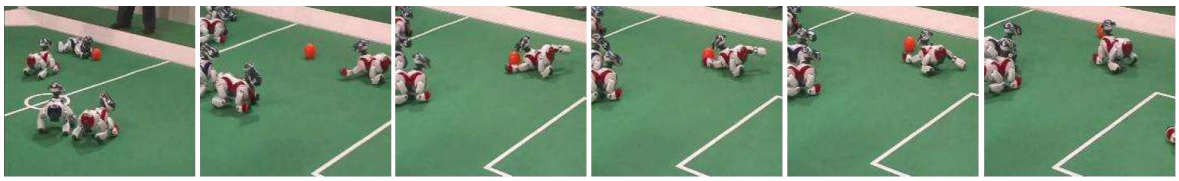


Abbildung 47: Bildsequenz alte Ballannahme

Die nun folgenden zwei Bildsequenzen stammt aus einem ERL-Spiel Darmstadt Dribbling Dackels in rot gegen S.P.Q.R. Sie zeigen die Ballannahme eines Feldspielers und des Torwarts.

Abbildung 48: Bildsequenz neue Ballannahme Feldspieler

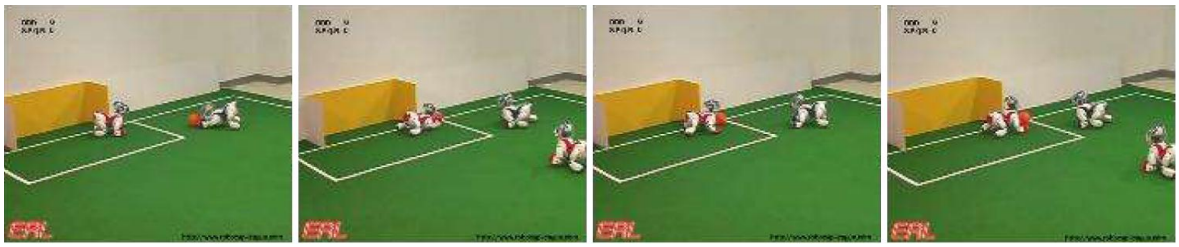


Abbildung 49: Bildsequenz neue Ballannahme Torwart

5.5.3_Ergebnisse

Die neue Ballannahme hat in Versuchen und im Spiel gezeigt, dass sie sowohl bei den Feldspielern als auch beim Torwart vorteilhaft ist. Der Ball liegt vor dem empfangenden Spieler für eine direkte Aktion bereit, ohne dass sich der Spieler dem Ball zeitraubend nähern muss.

5.6_Verhalten

Wie in den vorherigen Kapiteln erläutert wurde, stehen nun dem Verhalten Informationen über Positionen und Entfernungen der Gegner, Pässe und deren Wertung. Sowie Bewegungen des Passgebers und Passempfängers zur Verfügung. Daraus wurde im Rahmen dieser Arbeit ein Verhalten entwickelt, das die Ballabgabe und -annahme ermöglicht.

5.6.1_InputSymbols

Um dem Verhalten Informationen mitzuteilen, werden so genannte InputSymbols verwendet. Diese können boolesch oder numerisch sein [11]. Für das Passspiel wurden zwei Klassen von InputSymbols entwickelt, die nun näher erläutert werden.

5.6.1.1_PassSymbols

Die PassSymbols beziehen sich rein auf die für das Passspiel notwendigen Informationen und enthalten folgende Werte:

Name	Typ	Maß	Beschreibung
pass.found	Boolesch	-	Zeigt an, ob ein Pass gefunden wurde.
pass.quality	Numerisch	-	Gibt die Qualität des Pass wieder. Maximum 1,0
pass.distance	Numerisch	mm	Distanz zwischen den Passpartnern.
pass.angle-to-sender	Numerisch	Grad	Der relative Winkel zum Passgeber.
pass.angle-to-receiver	Numerisch	Grad	Der relative Winkel zum Passempfänger.
pass.player-number-of-pass-receiver	Numerisch	-	Enthält die Spielernummer des Passempfänger. Dies ist nützlich für die Entwicklung vom Passverhalten.

Tabelle 3: PassSymbols

Alle Werte werden vom PlayersLocator errechnet und entsprechend in den InputSymbols gesetzt.

5.6.1.2_PlayersModelSymbols

Die vom PlayersModel errechneten Informationen werden dem Verhalten nicht vollständig zur Verfügung gestellt, da es bis dato noch keine weitere Verwendung für diese Informationen gab. Während der RoboCup-WM 2005 in Osaka/Japan sind Abstandsinformationen und Winkel hinzugekommen, die im Penalty-Schützen eingesetzt wurden, um den gegnerische Torwart zu umspielen. Hier sind noch weitere Informationen sinnvoll, die aus dem Gegnermodell errechnet werden können, wie z.B. der Winkel zum gegnerfreien Raum vor dem Roboter.

Name	Typ	Maß	Beschreibung
players.angle-to-own-detected-player[11]	Numerisch	Grad	Relativer Winkel zum eigenen gesehenen Spieler im Sichtfeld. Dieser Wert dient zur Korrektur des Lokalisierungsfehlers.
players.time-since-detected-own-player	Numerisch	ms	Vergangene Zeit seit dem ein eigener Spieler gesehen wurde.
players.free-space-around-robot	Numerisch	mm	Freier Raum um den Roboter herum.
players.opp-player-ahead	Boolesch	-	Wahr, wenn ein Gegner ±45° relativ vor dem Roboter steht.
players.angle-to-opp-player-ahead	Numerisch	Grad	Winkel zum gegnerischen Spieler voraus.
players.distance-to-opp-player-ahead	Numerisch	mm	Abstand zum gegnerischen Spieler voraus.
players.distance-to-opp-player-behind	Numerisch	mm	Abstand zum gegnerischen Spieler rückwärtig.
players.distance-to-opp-player-left	Numerisch	mm	Abstand zum gegnerischen Spieler links.
players.distance-to-opp-player-right	Numerisch	mm	Abstand zum gegnerischen Spieler rechts.

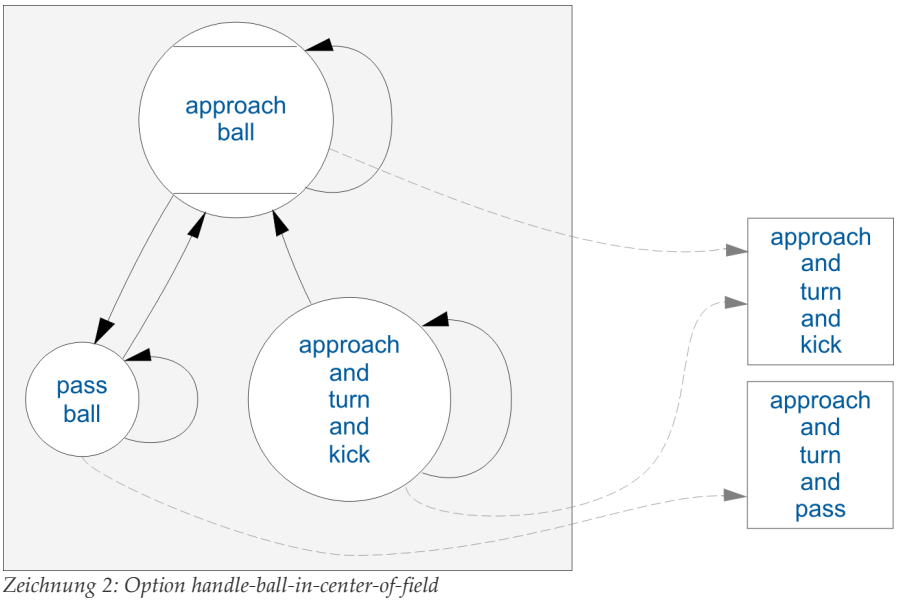
Tabelle 4: PlayersSymbols

11 Dieser Wert kann nur genutzt werden, wenn man im ImageProcessor das Erkennen der eigenen Spieler zulässt. Im normalen Spiel ist dies deaktiviert, um Rechenzeit zu sparen.

5.6.2_Verhalten Passgeber

Das Verhalten des Passgebers gliedert sich in zwei Schritte: Positionierung und Ballabgabe. Dabei liegt der Schwerpunkt auf der Positionierung. Abweichungen vom Idealwinkel zum Mitspieler haben drastische Folgen für die Ballannahme. Ein Beispiel: Bei einer zu überbrückenden Distanz von 1,5 m zwischen den Spielern und eine Abweichung bei der Passabgabe von 10° beträgt der Fehler bereits 26 cm. Diese Entfernung kann vom Roboter innerhalb einer Sekunde überwunden werden. Da bereits ein langsamer Ball 80 cm/s erreicht, bleibt weniger als eine Sekunde um zu reagieren. Hinzu kommt, dass der empfangende Roboter die Ballgeschwindigkeit und die Abweichung bei großen Entfernungen nicht ausreichend exakt bestimmen kann. Tests zeigten, dass er den Ball erst bei einer Entfernung von ca. 80 cm ausreichend genau erkennt, um die notwendigen Korrekturmaßnahmen einzuleiten. Oft beträgt der Fehler leider mehr als 10°, was zum einen von einer ungenauen Lokalisierung, zum anderen von dem nicht geraden Ballverlauf herrührt. Des Weiteren benötigt der Roboter noch ca. 400 ms um eine Abfangbewegung durchzuführen.

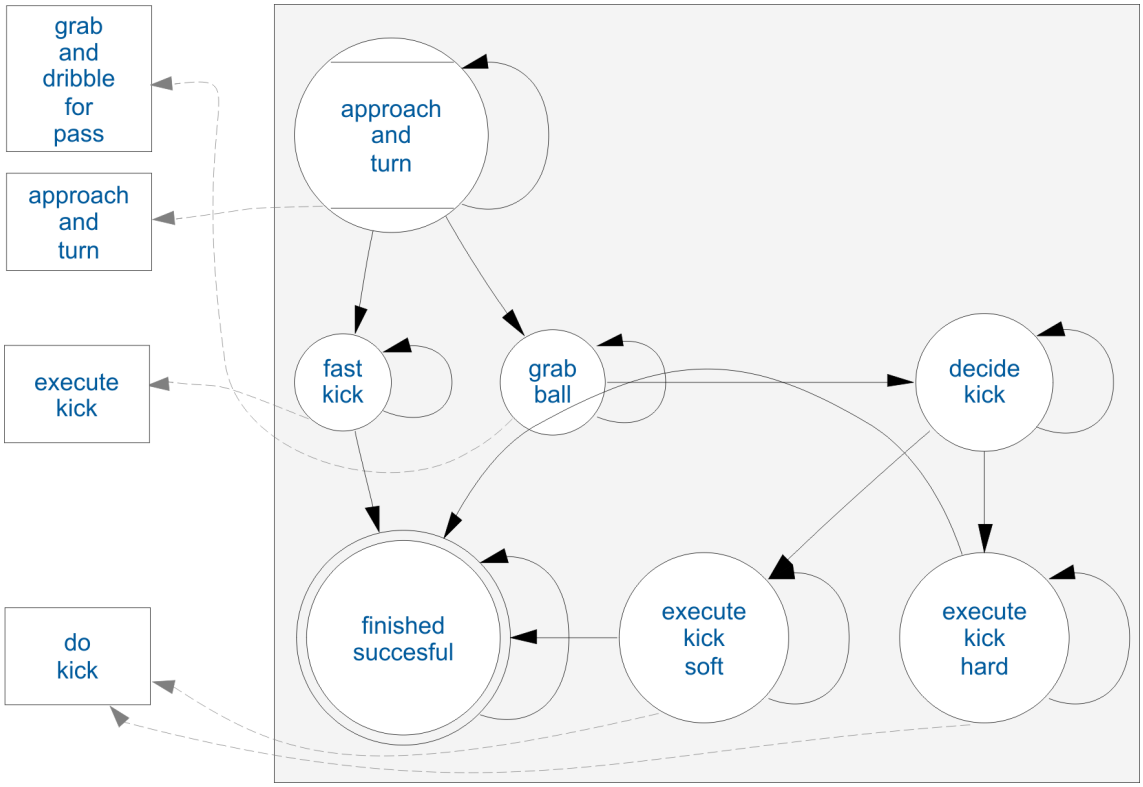
Im nun folgenden Teil wird auf die XABSL-Realisierung eingegangen. XABSL steht für eXtensible Agent Behavior Specification Language und kann grob umschrieben werden als eine State Machine. Für weitere Details bezüglich XABSL wird auf den GermanTeam Report [11] und auf die Diplomarbeit XABSL [6] verwiesen.



Zeichnung 2: Option handle-ball-in-center-of-field

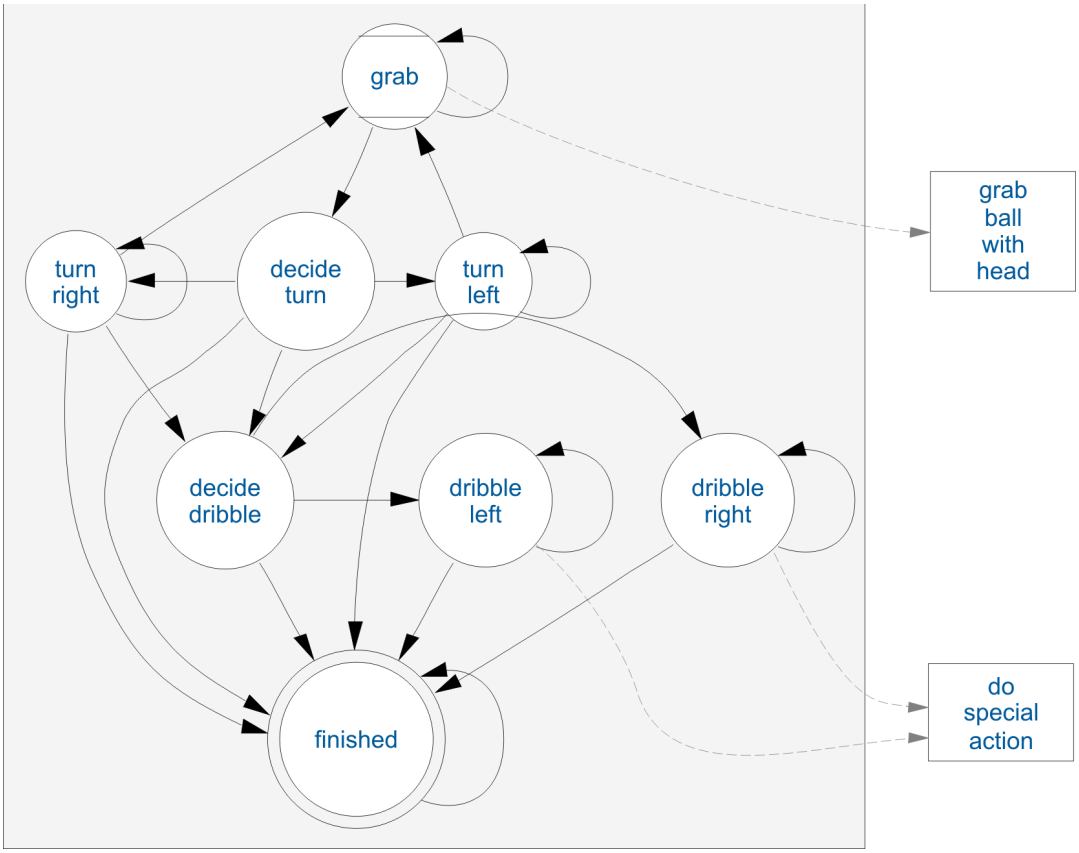
Um das Passspiel in einer beliebigen Spielsituation einzusetzen, ist es als eigenständige Option *approach-and-turn-and-pass* realisiert. Zur Zeit wird sie von der Option *handle-ball-in-center-of-field* aufgerufen. Der vollständige Pfad ist dem aktuellen Verhalten zu entnehmen.

Der Roboter nähert sich dem Ball und wechselt bei einem entsprechend gut bewerteten Pass in die Option approach-and-turn-and-pass. In diesem Zustand bleibt er dann bis das InputSymbols strategy.ball-is-handled-at-the-moment nicht mehr gesetzt ist. Dies soll garantieren, dass der Pass vollständig ausgeführt wird.



Zeichnung 3: Option approach-and-turn-for-pass

In dieser Option richtet sich der Roboter zum Ball aus und löst einen schnellen Schuss aus, falls der Roboter zum Passpartner ausgerichtet steht. Falls der Ball in Greifweite liegt, führt er die folgende Option grab-and-dribble-for-pass aus und dreht sich mit dem Ball zum Passpartner. Im Anschluss wird in Abhängigkeit von der Entfernung zum Passpartner entschieden, ob ein harter oder weicher Schuss ausgeführt wird.



Zeichnung 4: Option grab-and-dribble-for-pass

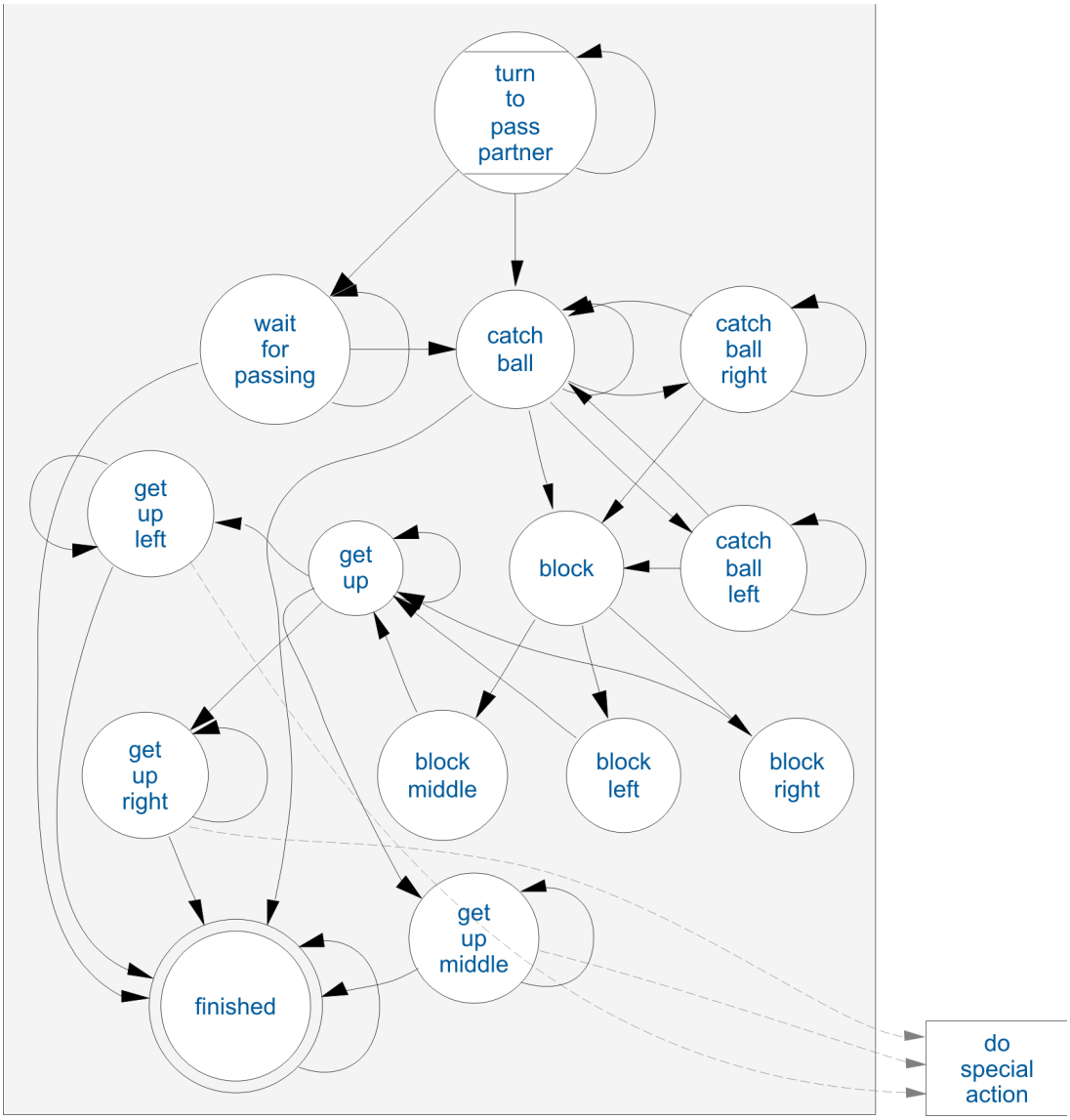
Nach dem Drehen mit dem Ball wird noch geprüft, ob ein Hindernis vor dem Roboter steht. Falls also ein Gegner im Passkorridor steht, wird mit wenigen seitlichen Schritten versucht, dem Hindernis auszuweichen - auf Kosten der Passgenauigkeit. Im Anschluss richtet sich der Roboter auf und führt dann in der approach-and-turn-for-pass den Schuss aus.

5.6.3_Verhalten Passempfänger

Der Passempfänger erhält über die TeamMessages die Nachricht, dass er in der nächsten Zeit einen Pass erhält. Deswegen ist die Option receive-pass in der Option playing-supporter verankert.

Sobald receive-pass ausgeführt wird, richtet sich der Roboter zum Passgeber aus und wartet 5 Sekunden auf den Pass. Ist die Zeit abgelaufen, wechselt er in das normale Spielverhalten. Wenn er den Ball sieht, wechselt er in den Zustand catch-ball und beginnt mit Seitwärtsschritten, falls der vorhergesagte Abstand auf der y-Achse des Koordinatensystems des Roboters bezüglich des Balls zu groß wird. Ist der Abstand zu groß, wechselt abermals der Roboter in das normale Spielverhalten.

Je nachdem auf welcher Seite der Ball ankommt, blockt der Roboter und führt im Anschluss das entwickelte Aufstehen aus.



Zeichnung 5: Option receive-pass

6_RESULTATE

„Nach dem Spiel ist vor dem Spiel“

Sepp Herberger

6.1_Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Passspiel entwickelt, welches zeigt, dass eine Teamtaktik in der 4-Legged League möglich ist. Während der Entwicklungszeit wurden viele Ansätze erdacht, entwickelt, getestet und wieder verworfen. Beispiele dafür sind der Partikelansatz und die Integration des ObstaclesModel in das Gittermodell. Die implementierten Module liefern in dieser ersten arbeitsfähigen Version gute Ergebnisse, die mit Verbesserungen und Erweiterungen die Art des Spiels der 4-Legged League bereichern können. Denn die Fähigkeit, Pässe zu spielen und damit Räume zu überbrücken, eröffnet gänzlich neue Möglichkeit.

6.2_Performance

Die Tabelle Tabelle 5 zeigt die durchschnittliche Rechenzeit des ImageProcessors. Der zusätzlich Mehraufwand für die Spielererkennung liegt bei 0,53 ms.

	Ø (ms)	Min (ms)	Max (ms)	Frequenz (Hz)	Abstand (cm)	Roboter
	16,98	15	22	16		0
	17,48	16	21	16,5	20	1
	17,36	16	21	17,2	60	1
	17,24	16	22	17,6	90	1
	17,5	16	24	19,3	150	1
	18,38	16	22	15,6	60	2
	17,66	16	21	15,6	90	2
Ø	**17,51**	**16**	**24**	**16,83**		

Tabelle 5: Rechenzeit des ImageProcessor mit blauen Robotern. Gemittelt über 50 aufgenommene Bilder mit Debugcode.

In der folgenden Tabelle sind die Rechenzeiten der einzelnen Module des GT2005 WM-Codes aufgelistet. Die Gegnermodellierung benötigt bei einem hochaufgelösten 15 cm Gridansatz nur 4,03 % der Gesamtrechenzeit. Bei einem Gridansatz mit 30 cm halbiert sich ca. die Rechenzeit.

Modul	Ø (ms)	Min (ms)	Max (ms)	Frequenz (Hz)
ImageProcessor	18,57	9	44	15,2
SensorDataProcessor	1,51	0	9	29
ballLocator	1,89	0	11	15,2
teamBallLocator	1,78	0	8	15,2
selfLocator	6,96	0	24	15,2
playersLocator	1,63	0	11	15,2
obstaclesLocator	3,1	1	10	15,2
behaviorControl	3,26	1	14	17,9
motionControl	0,68	0	4	121,2
soundControl	0,09	0	5	121,2
specialVision	0,01	0	2	15,2
sensorBehaviorControl	0,03	0	2	17,9
headControl	0,28	0	2	121,2
collisionDetector	0,43	0	5	29
robotStateDetector	0,13	0	2	29
Prozentualer Anteil	**4,03**			

Tabelle 6: Rechenzeit des PlayersModel bei einem 15 cm Grid. Gemittelt über 200 aufgenommene Bilder mit Debugcode. Blickrichtung ist das blaue Tor mit dem HeadControlMode search-for-ball. Zwei blaue Roboter werden detektiert.

6.3_Bewertung der Ergebnisse

Zu Beginn dieser Arbeit galten in der 4-Legged League noch andere Spielregeln und die Anpassung des Programmcodes an das neue Roboter Modell SONY ERS-7 war noch nicht so weit fortgeschritten, da er erst wenige Monate erhältlich war. Das hatte zum einen Vorteile für die Entwicklung der Spielererkennung, die nur mit den neuen Regeln 2005 [2], mit dem Spiel ohne weiße Bande in dieser Form möglich war. Denn ohne die Bande heben sich die weißen Roboter vom Hintergrund ausreichend gut ab. Eine Erkennung mit Bande wäre sicherlich auch möglich gewesen, hätte aber auch den Rechenaufwand erhöht, um falsche Perzepte herauszufiltern.

Als Vorteil für das Spiel, aber als Nachteil für das Passspiel, erwiesen sich die Ergebnisse der Laufoptimierungen, die nicht nur beim GermanTeam traumhafte Ergebnisse lieferten. Zum RoboCup 2004 in Lissabon/Portugal lag die Geschwindigkeit der Roboter noch bei 22-25 cm/s. Auf den GermanOpen 2004 in Paderborn besaßen die Microsoft Hellhounds von der Universität Dortmund [5] eine Laufgeschwindigkeit von 42 cm/s bzw. eine Sprintgeschwindigkeit von 50cm/s. Auch andere Teams der Liga können jetzt mit Laufgeschwindigkeiten teilweise über 40 cm/s aufwarten. Dies hat drastische Folgen für das Passspiel: Um einen Pass durchzuführen, ist Raum um den Passgeber und den Passempfänger notwendig. Da der Roboter keine Möglichkeit hat, aus einer beliebigen Position heraus einen Pass in einem bestimmten Winkel zu spielen, ist er auf die exakte Positionierung zum Ball hin angewiesen. Das kostet Zeit und gibt dem Gegner Gelegenheit das Passspiel noch vor der Ausführung zu stören. Auch beim Passempfänger ist die Ballannahme zeitraubend. Durch das Hinwerfen wird zwar die Reichweite erhöht, aber der Roboter benötigt wiederum Zeit um aufzustehen. Die zusätzliche Spielfläche von 72 %, die von dem größeren Feld in den Regeln von 2005 herrühren, ändern an der Situation nicht viel. Die z.B. 90 cm mehr in der Spielfeldbreite werden von den Robotern in knapp 2 Sekunden überbrückt. Das Aufstehen nach dem Block benötigt alleine etwas mehr als ein Sekunde. Fazit: Je schneller der Gegner, desto geringer die Chance auf einen erfolgreichen Pass.

Die Ergebnisse aus den einzelnen Modulen sind im Gegenteil zum Passspiel ermutigender. Die Spielererkennung arbeitet solide, schnell, ist nicht rechenintensiv und arbeitet mit einer geringen Zahl an Fehlerkennungen. Die Gegnermodellierung sichert ein effektives Verfolgen der Gegner und führt die Erkenntnisse der Roboter in einem globalen Gegnermodell sinnvoll zusammen.

Die Entwicklungen haben im Penalty Shootout auf der RoboCup WM 2005 gezeigt, dass sie zuverlässig auch den schwierigen Fall - der Erkennung eines blauen Gegners - meistern und haben dazu beigetragen, einen erfolgreichen Strafstossschützen zu entwickeln.

6.4_Ausblick

Um dem großen Ziel des RoboCups näher zu kommen, im Jahre 2050 den menschlichen Fußballweltmeister zu schlagen, wurde im Rahmen dieser Arbeit ein Passspiel entwickelt, das in einer zukünftigen integrativen Teamtaktik eingesetzt werden kann. Dabei sind zahlreiche Funktionen entwickelt worden, die auch für sich alleine genommen einen Zugewinn für das Spiel des GermanTeams darstellen. Die Spielererkennung ermöglicht es, einem Gegner auszuweichen, zu umlaufen oder gar ihn zu überholen. Das fusionierte globale Gegnermodell kann genutzt werden, um sich auf dem Spielfeld besser zwischen Gegner und dem Ball zu positionieren. Das verbesserte Ballblocken ermöglicht dem Torwart und dem defensive Supporter ein anderes, aggressiveres Spiel. Auch ist der Einsatz des neuen Aufstehens besser geeignet für den Torwart und könnte durch ein darauf optimiertes Verhalten einen schnellen, reaktiven Torwart erzeugen.

Bei der Spielererkennung kann eine Steigerung der Erkennungsrate durch die Nutzung von flächigeren Bewertungsfunktionen erreicht werden. Bewertungsfunktionen, die auf Basis eines Pixels arbeiten, so wie sie im GT200x Code üblich sind, führen oft zu schlechten Ergebnissen. Für das Clustern der möglichen Roboter wurde zeitweise eine Funktion eingesetzt, die auch die benachbarten Pixel in die Berechnung einbezieht. Die Qualität der Ergebnisse waren merklich besser. Fehlperzepte traten nur noch sehr selten auf, nahe aneinander stehende Roboter wurden besser als separate Roboter erkannt und die Abstandsberechnung wurde vor allem bei weit entfernten Robotern genauer. Leider vervierfachte sich die benötigte Rechenzeit. Ähnlich gute Ergebnisse wurde auch bei der Suche nach dem Bodenpunkt durch Einsatz der Flächenbewertungsfunktion erzielt. Ein Rechenzeit sparender Ansatz würde die Qualität der Spielererkennung erheblich steigern.

Des weiteren wäre eine Erkennung des Kopfes hilfreich, um eine genau Aussage über die Ausrichtung des Roboters zu erhalten. Wie in 3.3.4 beschrieben, ist die Erkennung ob es sich um Front oder Heck handelt recht ungenau.

Gegenwärtig wird in der Modellierung die Ausrichtung des Gegners noch nicht genutzt, weil die Umsetzung in der Modellierung zeitlich nicht mehr möglich war. Dies kann aber vorteilhaft für das Fällen der Entscheidung des Passspiels sein, da dann in Betracht gezogen werden kann, ob der Gegner genug Zeit hat, das Passspiel zu stören, da er sich z.B. erst selbst zum Ball ausrichten muss, weil der Ball seitlich zu ihm liegt, oder er den Ball selbst nicht sieht, weil der Ball hinter ihm liegt.

Wie in 4.8.5 beschrieben, ist der Einsatz des ObstaclesModels zur Verbesserung der Modellierung interessant. Zwar ist die Rate der fehlerhaft erkannten Roboter nicht sonderlich hoch, doch erzeugen die Abweichungen in der Selbstlokalisierung immer wieder Geisterroboter und stören damit das Gegnermodell zeitweise erheblich. Die Nutzung der Negativinformation kann die Qualität des Gegnermodell erheblich steigern, wie sich in Versuchen im Simulator gezeigt hat.

Eine weitere Verbesserung wäre es, wenn während der Ausrichtung zum Passpartner, der players.angle-to-own-detected-player eingesetzt würde. Dies setzt allerdings voraus, das der Roboter während dem GrabBall das Spielfeld beobachten kann und das eigene Team Perzepte im ImageProcessor erzeugt (siehe 3.3). Teams wie den NU-Bots ist dies bereits möglich. Somit muss nicht vollständig auf die Selbstlokalisierung vertraut werden und ein genaueres Passspiel kann erreicht werden.

Weit in die Zukunft gedacht, wäre die Nutzung einer Spielermodellierung, in der sowohl eigene Spieler als auch Gegner modelliert werden denkbar. Diese Modellierung könnte dann mit der Selbstlokalisierung zusammengeführt werden und somit die eigene Lokalisierung verbessern, da z.B. ein eigener Spieler seine eigene Position exakt kennt und die Position seines eigenen Teammitglieds besser bestimmen kann als der gerade beobachtende Spieler. Dieser Fall tritt häufig ein, wenn ein Roboter gerade aus einem „Getümmel“ kommt.

Sinnvoll wäre es auch, wenn man durch die Gegnermodellierung erkannte „Infight“-Situation in der Selbstlokalisierung nutzt und somit z.B. die Bewertung der Partikel durch erkannte Landmarken verringert, da diese im „Getümmel“ nicht aussagekräftig sind.

ANHANG

„Football is a simple game; 22 men chase a ball for 90 minutes and at the end, the Germans win.“

Gary Lineker

LITERATURVERZEICHNIS

[1] Fussball, Schülersport,
Dr. N. Rogolaski, Dr.E.-G. Degel, Sportverlag Berlin, 1969

[2] Sony Four Legged Robot Football League Rule Book ,
RoboCup Technical Committee, 2005

[3] Praktikumsbericht Ballannahme, Dribbling, Sidestep und Schußvarianten ,
Marko Borazio, Patrick Winkler, 2005

[4] Praktikumsbericht Kalman-Filter Ballmodellierung ,
Stefan Uhrig, Marcus Schobbe, Patrick Stamm, 2004

[5] Microsoft Hellhounds Homepage,
http://www.fussballhun.de/, besucht am 18.9.2005

[6] XABSL: A Behavior Engineering System for Autonomous Agents ,
Martin Lötzsch, 2004

[7] Directed Vision by autonomous HeadControl in the SONY 4-legged League ,
Marc Dassler, 2004

[8] Ein globales Sichtsystem zur Unterstützung der Entwicklung autonomer Fußballroboter,
Ronnie Brunn, Michael Kunz, 2005

[9] Visual Robot Detection in RoboCup using Neural Networks ,
Ulrich Kaufmann, Gerd Mayer, Gerhard Kraetzschmar, and Günther Palm, 2004

[10] Homepages des GermanTeam,
http://www.germanteam.org/, besucht am 2.9.2005

[11] GermanTeam Report 2004,
http://www.germanteam.org/GT2004.pdf, besucht am 2.9.2005

[12] RoboCup Federation,
http://www.robocup.org, besucht am 2.9.2005

[13] A Vision Based System for Goal-DirectedObstacle Avoidance ,
Jan Hoffmann, Matthias Jüngel and Martin Lötzsch, 2004

[14] OPEN-R SDK Model Information ERS-7
SONY Cooperation, 2004

[15] 3rd Generation Of Sony's Entertainment Robot Evolves ,
SONY Cooperation, 2003