Studienarbeit

Entwicklung und Implementierung defensiver Spielstrategien für das RoboCup-Projekt des Teams „Tigers Mannheim“

Malte Mauelshagen

Mannheim, den 13.12.2010

Betreuer:
Dr. Mirko Wachs

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich
die vorliegende Arbeit selbstständig angefertigt
und keine weiteren, als die angegebenen Quellen
und Hilfsmittel benutzt habe.

Mannheim, den 15. Dezember 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AI ..... **A**rtificial **I**ntelligence

DECT ..... **D**igital **E**nhanced **C**ordless **T**elecommunications

KI ..... **K**ünstliche **I**ntelligenz

SSL ..... **S**mall-**S**ize **L**eague

1. Einleitung

Das Forschungsgebiet „Künstliche Intelligenz“ entstand Mitte der 50er Jahre im Zuge der Entwicklung von Computern und immer komplexeren Rechenmaschinen. Diese sollten gestellte Probleme selbständig lösen und analog zur Funktionsweise des menschlichen Gehirns lernfähig sein. Eine genaue Definition des Begriffs ist jedoch äußerst schwierig, da die Abgrenzung von rein logisch, mathematischem Denken zu emotionalen und kreativen Fähigkeiten nicht strikt getrennt ist.

Um ein Prüfmaß für Intelligenz von Computern zu finden, wurden oft Aufgaben gewählt, die wenige, einfache Regeln beeinhalten und somit relativ leicht zu implementieren sind. Exemplarisch für viele Intelligenztests war in den ersten Jahren das Schachspiel. Laut einer Vorhersage von *Herbert Simon* im Jahre 1957 werde „Innerhalb der nächsten zehn Jahre“ ein Computer den Schachweltmeister besiegen. Im Laufe der Zeit wurden viele der großen Erwartungen an die künstliche Intelligenz enttäuscht. Selbst realitätssimulierende Algorithmen, wie neuronale Netze oder Schwarmverhalten sind auch heute noch weit davon entfernt die kognitiven Fähigkeiten menschlicher Gehirne zu erreichen oder diese gar zu übertreffen. Erst 1997 wurde der amtierende Schachweltmeister *Gari Kasparov* von dem Schachprogramm *Deep Blue* geschlagen und damit über 30 Jahre später als vorausgesagt. [3]

In Anlehnung an diesen Meilenstein künstlicher Intelligenz entstand die Idee, den sportlichen Wettbewerb Mensch gegen Maschine nicht mehr nur auf rein logisch, rechnerischer Ebene auszutragen. Unter Berücksichtigung weiterer Bereiche, wie beispielsweise Robotik, Echtzeitsysteme, Planung auf Basis unvollständiger Informationen und Multiagentensysteme, soll bis 2050 der amtierende Fußballweltmeister von einer Mannschaft humanoider Roboter besiegt werden.

Hinter dem Namen **RoboCup** verbirgt sich eine Initiave, die dieses Vorhaben antreibt. Jährlich finden Meisterschaften fußballspielender Roboter in verschiedenen Ligen statt, wobei keine Nationalmannschaften, sondern Hochschulteams aus aller Welt gegeneinander antreten. Mittlerweile gehören auch Wettbewerbe außerhalb des Fußballs zum Programm, wie Robocup Rescue, wo Katastropheneinsätze simuliert werden und Roboter Rettungseinsätze übernehmen. [1]

1.1. Tigers Mannheim

Im Jahre 2009 wurde an der Dualen Hochschule Baden-Württemberg Mannheim das Team „Tigers Mannheim“ gegründet. Ziel des studentischen Projekts ist die Teilnahme am RoboCup 2011 in Istanbul. Als Spielklasse wurde die **Small-Size League (SSL)** gewählt, bei der fünf Roboter je Team in einem Fußballspiel gegeneinander antreten. Zur Teilnahme an der Weltmeisterschaft ist eine Qualifikation in Form eines „Description Papers“ und eines Team-Videos notwendig; Vorausscheidungsspiele gibt es keine.

Die zylindrisch geformten Roboter mit einer Höhe von nur 15 Zentimetern fahren auf vier sogenannten „Omni-Wheels“ und können den Spielball mit ihrer eingebauten Schusswalze auf bis zu $10\frac{m}{s}$ beschleunigen. Kameras über dem Spielfeld nehmen das aktuelle Geschehen auf und leiten es an einen Zentralrechner weiter, auf dem die eigentliche Software läuft.

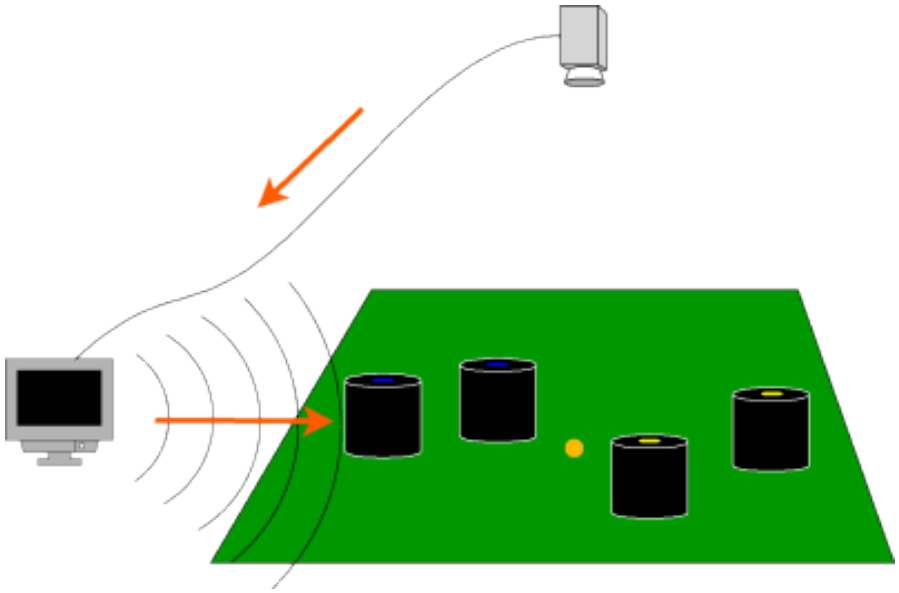

Abbildung 1.1.: Datenfluss während eines Spiels in der Small-Size League [2]

Dort werden auf Basis der Kameradaten sowie elektronisch übermittelten Schiedsrichterentscheidungen Analysen der Spielsituation vorgenommen. Nach Berechnung geeigneter Spielzüge werden die Ansteuerungsbefehle für die Roboter über eine Drahtlosverbindung verschickt.

2. Modul: Künstliche Intelligenz

Zur besseren Verständlichkeit des Datenflusses innerhalb meines Arbeitsbereiches, soll das Modul der „Artificial Intelligence“ innerhalb der Zentralsoftware kurz erläutert und der Aufbau grob skizziert werden.

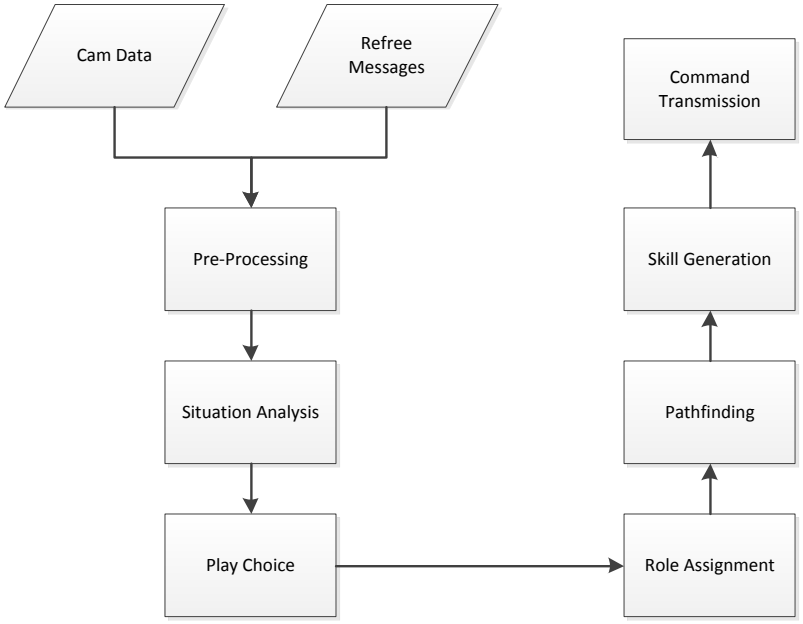

Abbildung 2.1.: Datenfluss und Verarbeitungsschritte des KI-Moduls

Alle 16 Millisekunden werden aktuelle Bilddaten von den Kameras rausgeschickt. Nach einer Vorverarbeitung werden sogenannte **Worldframes** generiert, die die Basis für die Arbeit der künstlichen Intelligenz bilden. Ein Worldframe beinhaltet die Positionen, Bewegungs- und Beschleunigungsvektoren aller Roboter sowie des Balls, außerdem einen Zeitstempel, eine eindeutige ID und mögliche Nachrichten des Schiedsrichters. Daraufhin erfolgt zunächst eine Analyse der aktuellen Situation mit Hilfe einer Rasterisierung des Spielfelds. Auf deren Grundlage muss sich danach für einen Spielzug entschieden werden, der letztendlich Roboterbefehle festlegt. Die höchste Abstraktionsebene bilden **Plays**. Hier werden komplexe Spielzüge definiert, die für die aktuelle Situation am sinnvollsten

erscheinen. Jedes Play beinhaltet 1-5 **Roles**, die spezifische Aufgaben übernehmen, wie beispielsweise „Decke das Tor ab!“ oder „Laufe dich frei!“. Diese werden danach in kleinere Verarbeitungsschritte zerlegt und nach Anwenden eines **Pathfinding**-Algorithmus durch Kombination oder Aneinanderkettung von **Skills** („Drehung um 30 Grad“, „Aktiviere die Schusswalze!“) dargestellt. Abschließend schickt das Programm die Ansteuerungsbefehle über eine DECT-Verbindung an den Mikrocontroller des Roboters, welcher sie dort weiterverarbeitet.

Die komplette Software ist in der Programmiersprache Java geschrieben. Als Entwicklungsumgebung wurde *Eclipse* verwendet, ein Programm, welches bereits in der Vorlesung „Programmieren“ an der DHBW-Mannheim genutzt wurde. Zur Versionskontrolle und -verwaltung diente *Apache Subversion*.

3. Defensive Spielstrategien

Während des letzten Semesters habe ich mich mit der Entwicklung von defensiven Plays sowie der dafür nötigen Roles beschäftigt (vgl. A.1). Besonderer Fokus wurde hier auf die Qualifikationsanforderungen des Robocups gelegt. Exemplarisch sollen in diesem Bericht drei Spielzüge behandelt werden.

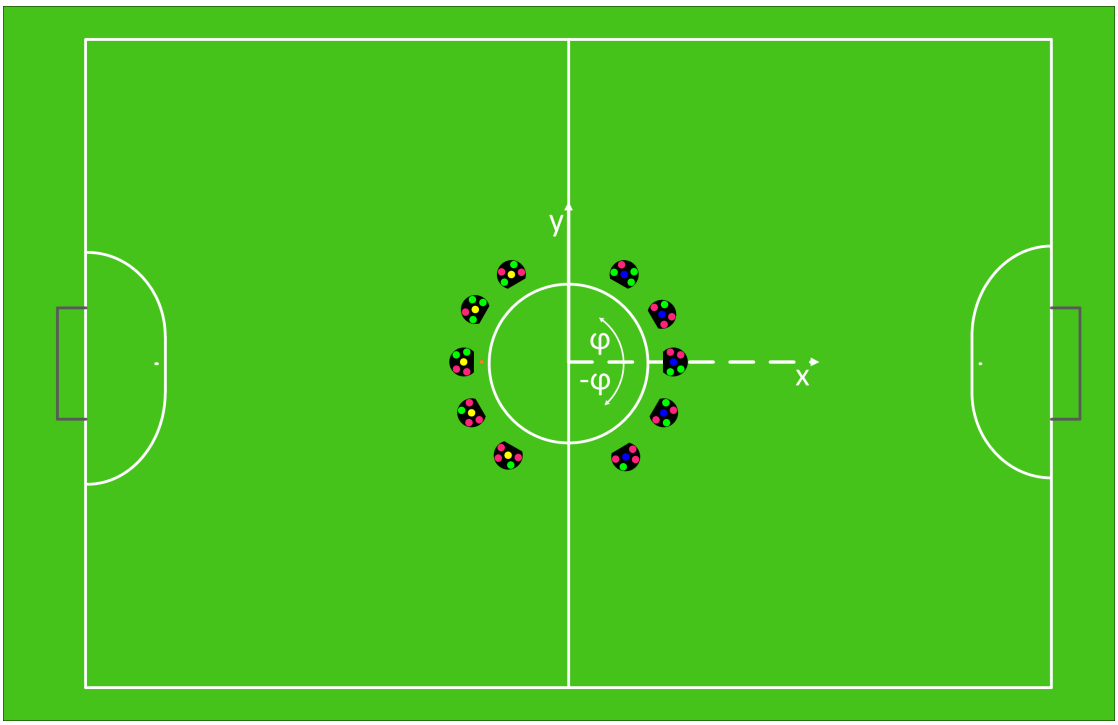

Abbildung 3.1.: Aufstellung der Roboter auf dem Spielfeld

3.1. KeeperSoloPlay

In den offziellen Bedingungen der Small-Size League zur Teilnahme an einer Weltmeisterschaft muss in den Qualifikationsvideos gezeigt werden, dass die Mannschaften zum aktiven Spiel fähig sind. So heißt die erste Anforderung für eine darzustellende Spielsituation:

a) One or more robots competing against an active robot goalkeeper. [4]

Da mein Aufgabenbereich den defensiven Part umfasst, habe ich ein **KeeperSoloPlay** mit einer dazugehörigen **KeeperSoloRole** für das KI-Modul geschrieben.

Laut offiziellen Regeln darf sich kein Feldspieler innerhalb des Torraums befinden. Um nicht von anderen Robotern gestört zu werden, soll sich der Bewegungsraum des Torwarts auf eben diesen Torraum beschränken. Für einen ersten einfachen Ansatz zur Positionierung wird ein Kreis mit Tormittelpunkt gedacht, dessen Radius r der Abstand von der Torlinie bis zur vorderen Torraumlinie bildet. Der so kreierte Halbkreis definiert vorerst die Punkte, die als mögliche Zielpunkte in Frage kommen. Wie Abbildung 3.2 zeigt, bestimmt der Schnittpunkt der Linie Tormittelpunkt (m) - Ballposition (b) mit dem Halbkreis den Verteidigungspunkt T des Torwarts. Er errechnet sich wie folgt:

$$\vec{T} = \vec{m} + \frac{r}{\left|\vec{b} - \vec{m}\right|} \cdot (\vec{b} - \vec{m}) \tag{3.1}$$

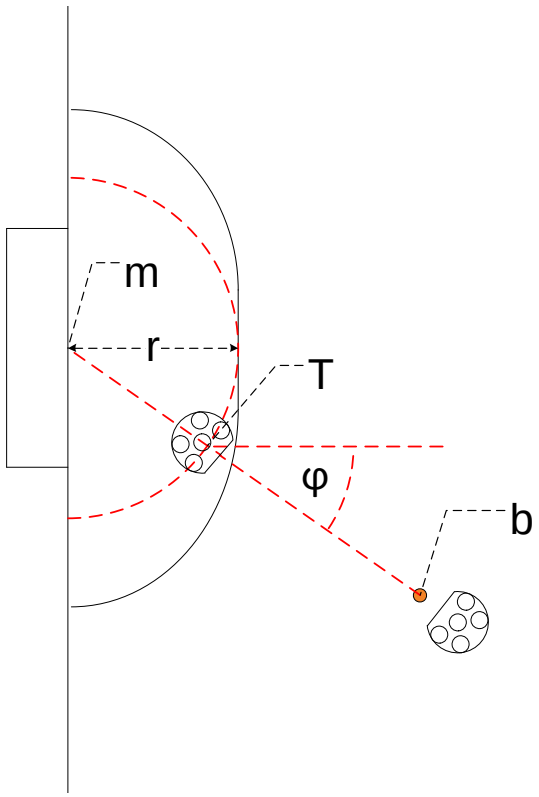

Abbildung 3.2.: Positionierung eines einzeln verteidigenden Torwarts

Desweiteren soll die Vorderseite des Roboters stets zum Ball gerichtet sein, um abgefangene Bälle möglichst schnell kontrollieren zu können. Wie in Abbildung 3.1 dargestellt, werden Winkel unterhalb der X-Achse negativ, oberhalb als positiv gewertet und befinden sich somit immer im Bereich von $[-\pi; \pi]$. Bezüglich Abbildung 3.2 ist der Ausrichtwinkel also $-\varphi$.

Aus dieser Rolle lassen sich demnach, wie im Übrigen bei allen Defensiv-Rollen, genau 2 Skills ableiten. Der **MoveToXY**-Skill erhält die Zielposition als Parameter, der **Rotate**-Skill den Drehungswinkel. Das Finden eines geeigneten Pfades, sowie die parallele Abarbeitung der Skills übernimmt das **Skillsystem**, der High-Level KI Programmierer muss sich darum nicht kümmern.

3.2. KeeperPlus2DefenderPlay

Bei defensiven Spielsituationen, insbesondere bei gegnerischem Ballbesitz sollen jeweils zwei Feldspieler zum Blocken des Tores abgestellt werden, während die restlichen beiden aktive Mann- oder Raumdeckung betreiben. Das **KeeperPlus2DefenderPlay** bestimmt das Verhalten des Torwarts und 2 Verteidigern, die gemeinsam versuchen das Tor großflächig abzudecken.

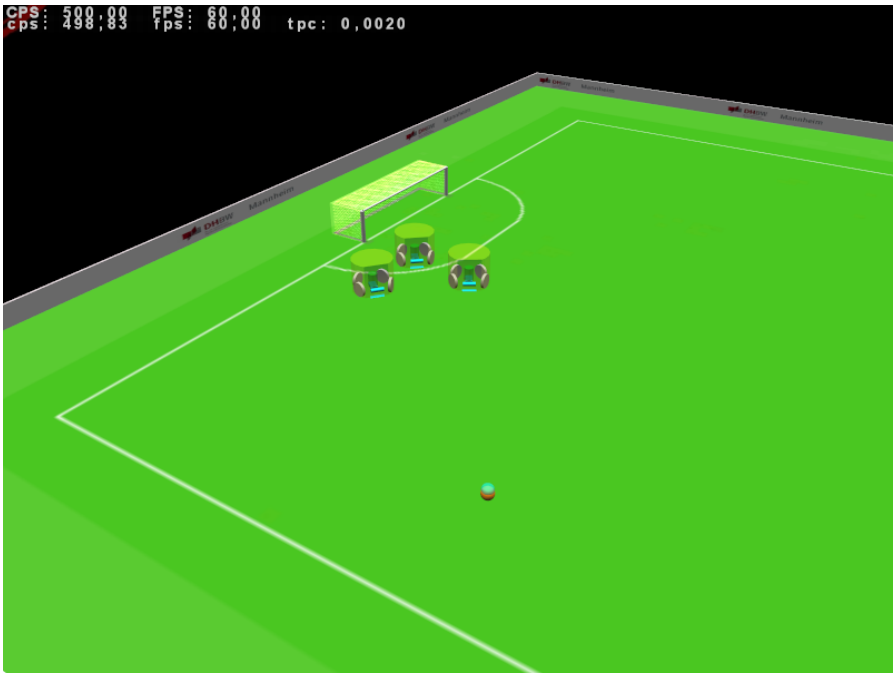

Abbildung 3.3.: Darstellung des KeeperPlus2DefenderPlay im Simulator

Um eine korrekte Synchronisation der drei Roboter zu gewährleisten, wurde der Torwart als eine Art Master-Rolle definiert. In den Rollen der Verteidiger steckt keine eigene Logik; sie richten sich stets nach der Position ihres Hintermanns.

Für den Torwart könnte hier prinzipiell wieder die KeeperSoloRole eingesetzt werden, aber zur logischen Trennung der Spielsituationen wurde eine eigene Torwart-Rolle für das Play geschrieben. Außerdem kann nun auf einen alternativen Ansatz zur Positionierung eines zentral verteidigenden Torwarts eingegangen werden. Anstatt die Linie Tormittelpunkt-Ballposition zu verwenden, wird hier die Winkelhalbierende der beiden

Strecken linker Pfosten-Ballposition und rechter Pfosten-Ballposition eingesetzt. Dies erlaubt dem Torwart eine effektivere Positionierung besonders bei seitlichen Angriffen. Zur einfacheren Berechnung der Winkelhalbierenden dient die Eigenschaft, dass in einem Dreieck der Schnittpunkt der Winkelhalbierenden mit der gegenüberliegenden Seite diese im Verhältnis der anliegenden Seiten teilt. In Abbildung 3.4 gilt daher:

$$\frac{a_1}{a_2} = \frac{s_1}{s_2} \tag{3.2}$$

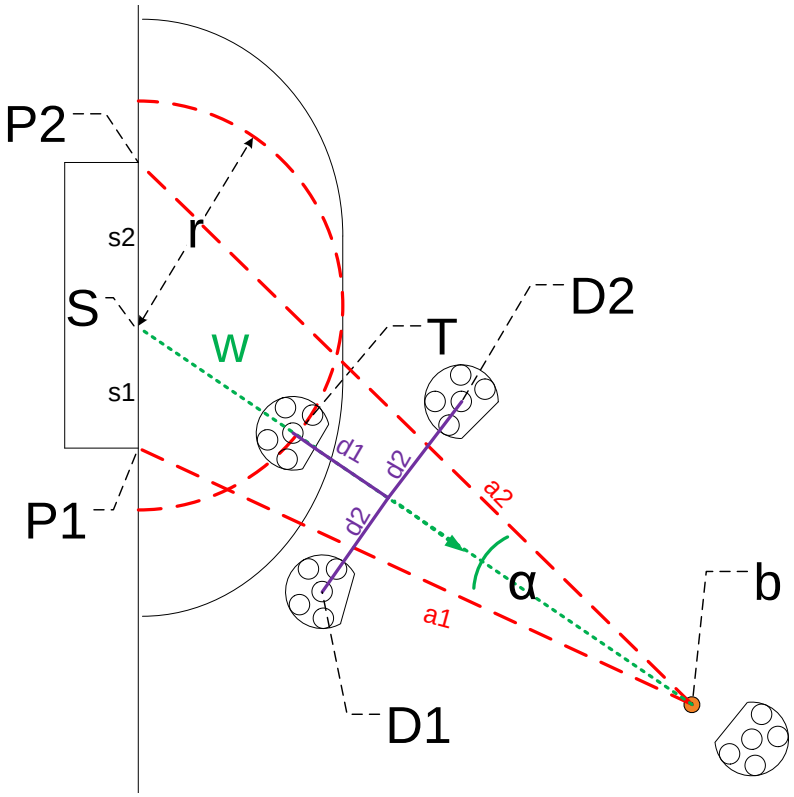

Abbildung 3.4.: Positionierung von einem Torwart und 2 Verteidigern

Der Schnittpunkt S der Winkelhalbierenden w mit der Torlinie bildet nun den neuen Ausgangspunkt zur Positionierung des Torwarts. Addiert man dazu den Richtungsvektor der Winkelhalbierenden und skaliert diesen auf die Länge r erhält man den Zielpunkt T.

$$\begin{aligned} \vec{T} &= \vec{S} + \lambda\vec{w} \\ &= \vec{S} + \frac{r}{\left|\vec{S} - \vec{b}\right|} \cdot (\vec{S} - \vec{b}) \end{aligned}$$

S ergibt sich als Addition von $\vec{P_1}$ mit einem senkrechten Vektor $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ der Länge s_1.

$$\vec{S} = \vec{P_1} + s_1 \tag{3.3}$$

s_1 ist nach Gleichung 3.2:

$$s_1 = \frac{a_1}{a_1 + a_2} \cdot s_{gesamt} \tag{3.4}$$

Wobei gilt:

$$s_{gesamt} = |\vec{P_1} - \vec{P_2}| \tag{3.5}$$

Die Positionen der beiden Verteidiger lassen sich mit jeweils 2 Verschiebungen der Position des Torwarts einmal in Richtung Ball $\vec{w}$ und einmal orthognonal zur Winkelhalbierenden $\vec{w}_o$ berechnen.

$$\vec{D_1} = d_1\vec{w} + d_2\vec{w}_o \tag{3.6}$$

$$\vec{D_2} = d_1\vec{w} - d_2\vec{w}_o \tag{3.7}$$

$$\vec{w}_o = \begin{pmatrix} w_{ox} \\ w_{oy} \end{pmatrix} = \begin{pmatrix} w_y \\ w_x \end{pmatrix} \tag{3.8}$$

Die Länge der Verschiebungsvektoren wurde vorerst willkürlich festgelegt, sollte aber später auf Basis von Erfahrungswerten angepasst werden. Über eine dynamische Änderung in Abhängigkeit der Positionen von Gegner und Ball ist ebenfalls nachzudenken.

3.3. KeeperPlus1DefenderPlay

Das Studium des Spielverhalten anderer Teams hat gezeigt, dass Tore oft durch Fernschüsse bei eigenem Ballverlust im Angriff fallen. Aufgrund der hohen Ballgeschwindigkeiten hat ein einzelner Torwart meist keine Chance diese Treffer noch zu verhindern. Bei eigenem Ballbesitz und Angriff soll daher stets ein Verteidiger hinten bleiben um dem Keeper zur Seite zu stehen.

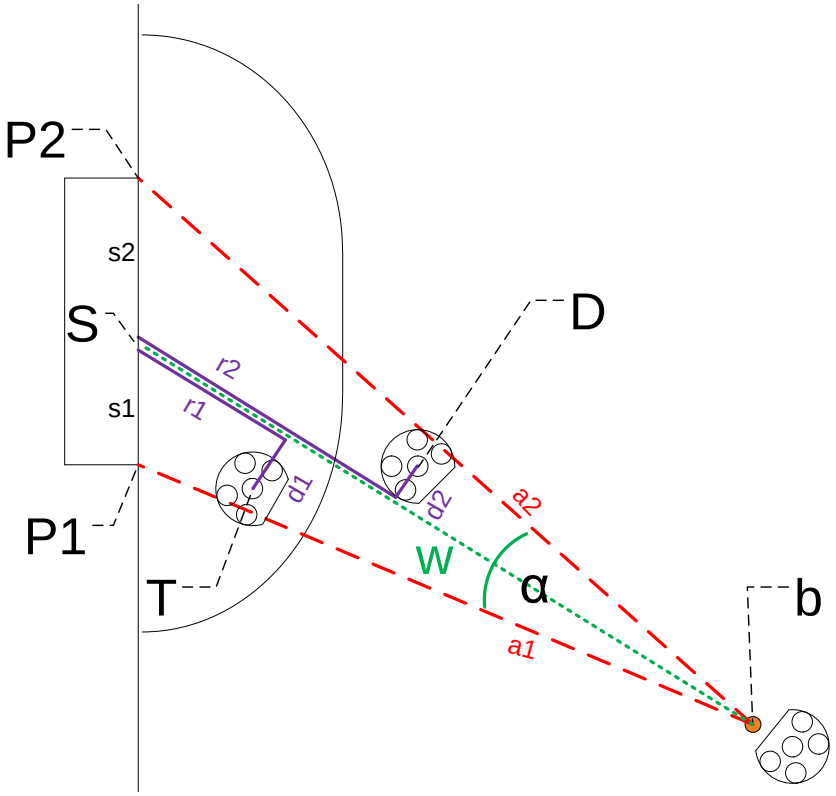

Abbildung 3.5.: Positionierung von einem Torwart und einem Verteidiger

Der Torwart fungiert nicht wie bisher als zentraler Blocker, sondern ersetzt den zweiten Innenverteidiger, allerdings aufgrund der Torraumbeschränkung etwas nach hinten versetzt, was zu einer asymmetrischen Aufstellung führt. Wie im vorigen Play wird eine Winkelhalbierende aufgestellt, die als Orientierungsgerade verstanden werden kann. Mittels der Parameter r und d kann die genaue Positionierung erfolgen, wobei r parallel und d orthogonal verschiebt. Zur Berechnung der Zielpunkte vergleiche Gleichung 3.6 und 3.7.

Um eine hohe Variabilität zu erhalten, wird erst zur Laufzeit entschieden, ob der Torwart die linke Seite und der Verteidiger die rechte deckt oder umgekehrt. Dazu wird einmalig nach Auswahl des Plays im derzeitigen Worldframe überprüft, welcher Roboter sich weiter links befindet. Daraufhin erfolgt die Zuweisung der Seite an die Rollen.

4. Zusammenfassung und Ausblick

Dank der guten Softwarestruktur und der Repräsentation von algebraischen Figuren wie Vektoren und Geraden als Java-Klassen war die Umsetzung der mathematischen Gleichungen in übersichtliche Algorithmen relativ einfach. Außerdem fanden während der Entwicklungszeit mehrere KI-Meetings statt, deren Diskussionsergebnisse den allgemeinen Aufbau von Roles und Plays, sowie deren Interaktion untereinander optimieren konnten.

Leider war bis zum Abschluss der Studienarbeit noch kein Roboter voll einsatzfähig, so dass es nicht möglich war, die verschiedenen Szenarien in der realen Welt zu testen. Dafür stand als passendes Werkzeug der vom Sim-Team programmierte 3D-Simulator bereit. Diese Testplattform versucht eine möglichst detailgetreue Abbildung der Wirklichkeit mit Hilfe der Physics Engine *Java Monkey Engine*, um KI-Entwicklern eine einfache Möglichkeit zu bieten, Algorithmen und Spielzüge ortsunabhängig zu testen. Über eine Steuerungsfunktion im grafischen Interface der Zentralsoftware (vgl. Abbildung A.2) ist es möglich, den automatischen PlayFinder zu deaktivieren und spezifische Plays manuell auszuwählen und diese zu erproben. Dafür wurde der Ball (derzeit einziger Einflussparameter für DefensePlays) im Simulator bewegt und die Reaktion der Roboter beobachtet. Ausnahmslos alle Test liefen erfolgreich, die Bewegungsresultate waren die vorhergesagten. Diesbezüglich scheinen die Defensiv-Plays korrekt implementiert und die Aufgabe gut erledigt.

Dennoch gibt es einige Verbesserungsmöglichkeiten und wünschenswerte Änderungen, die zwar nicht obligatorisch in Bezug auf die Qualifikationsanforderungen, aber für ein erfolgreiches Abschneiden bei der Weltmeisterschaft von Nöten sind: Zunächst einmal sollte neben der Position des Balls auch dessen Bewegungsvektor, sowie die Positionen und Bewegungen der gegenerischen Roboter Einfluss auf die Positionierung der Verteidiger nehmen. Insbesondere ist hier auf die Ausrichtung (Wohin zielt der Angreifer?) zu achten. Zur Verbesserung der bisherigen Plays gehört darüber hinaus eine exaktere Beschreibung der Situationen, beispielsweise das Aufheben der Approximation des Torraums mit Hilfe eines Halbkreises. Desweiteren sollten Spiele gegen die eigene KI simuliert werden. „Offensiv gegen Defensiv“ wäre ein Szenarion aus dem beide Seiten nur positive Erfahrungen ziehen können, um Fehler ihrer Software oder Verbesserungsmöglichkeiten zu finden. Eine Art „Freundschaftsspiel“ gegen andere Teams ist ebenfalls denkbar, wenn

auch in fernerer Zukunft. Um Ecken, Elfmetern und Freistößen adäquat zu begegnen, sind Defense Plays für diverse Standardsituationen unumgänglich.

Wie gut die TIGERS Mannheim in Istanbul letztendlich bestehen werden, steht noch ungewiss. Die Qualifikation wird bei funktionierender Robotermechanik aber wohl keine Probleme bereiten.

Literaturverzeichnis

[1] About robocup. http://www.robocup.org/about-robocup/. [Online; Stand 13. Dezember 2010].

[2] Beispielhafte darstellung eines spiels in des small-size league. http://small-size.informatik.uni-bremen.de/_media/dataflow.png. [Online; Stand 13. November 2010].

[3] Computerschach. http://www.geocities.com/SiliconValley/Lab/7378/comphis.htm. [Online; Stand 13. Dezember 2010].

[4] Submission of your team video. http://small-size.informatik.uni-bremen.de/robocup2010:qualification. [Online; Stand 13. November 2010].

A. Anhang

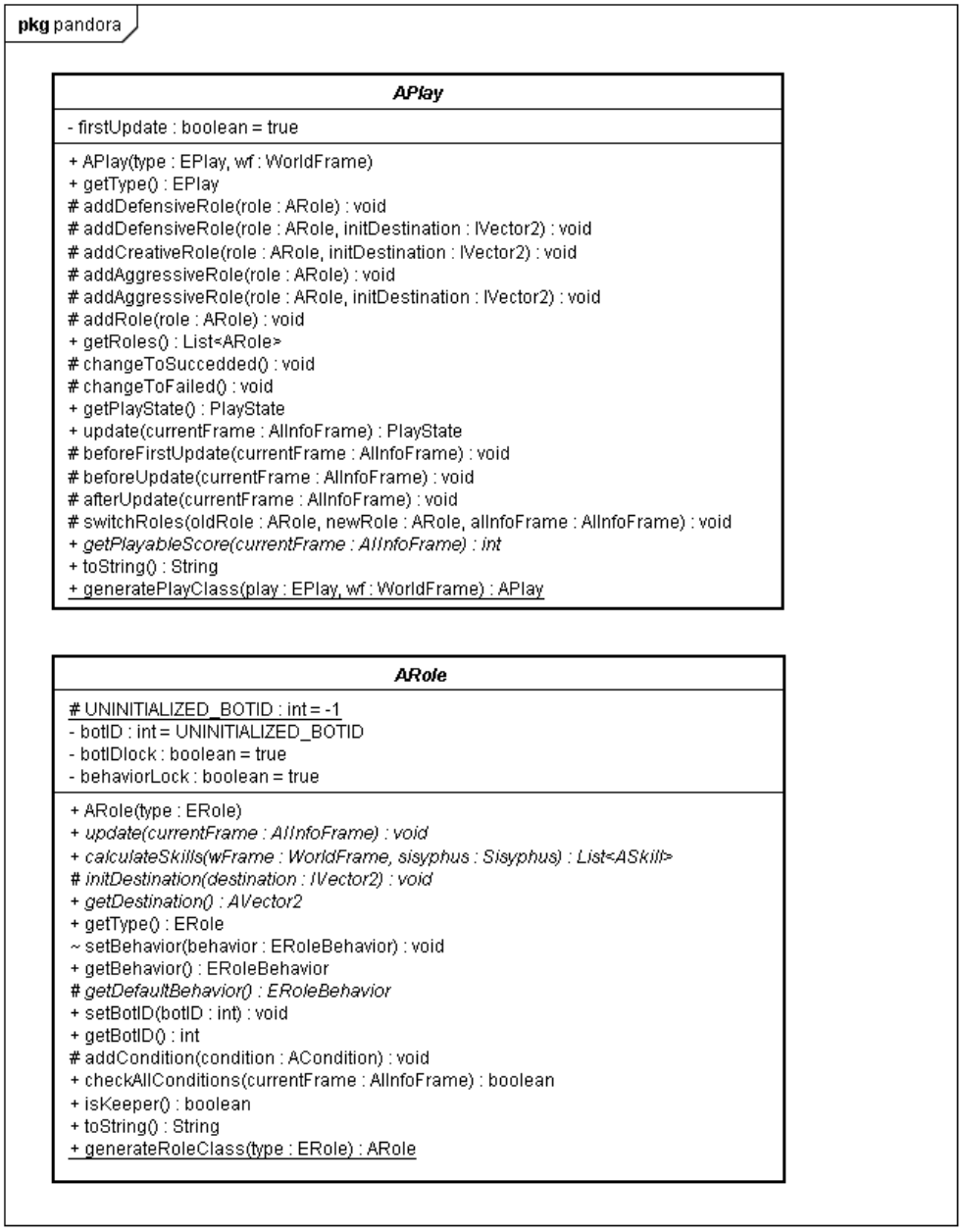

Abbildung A.1.: Klassendiagramm von APlay und ARole, den abstrakten Oberklassen aller Plays und Roles

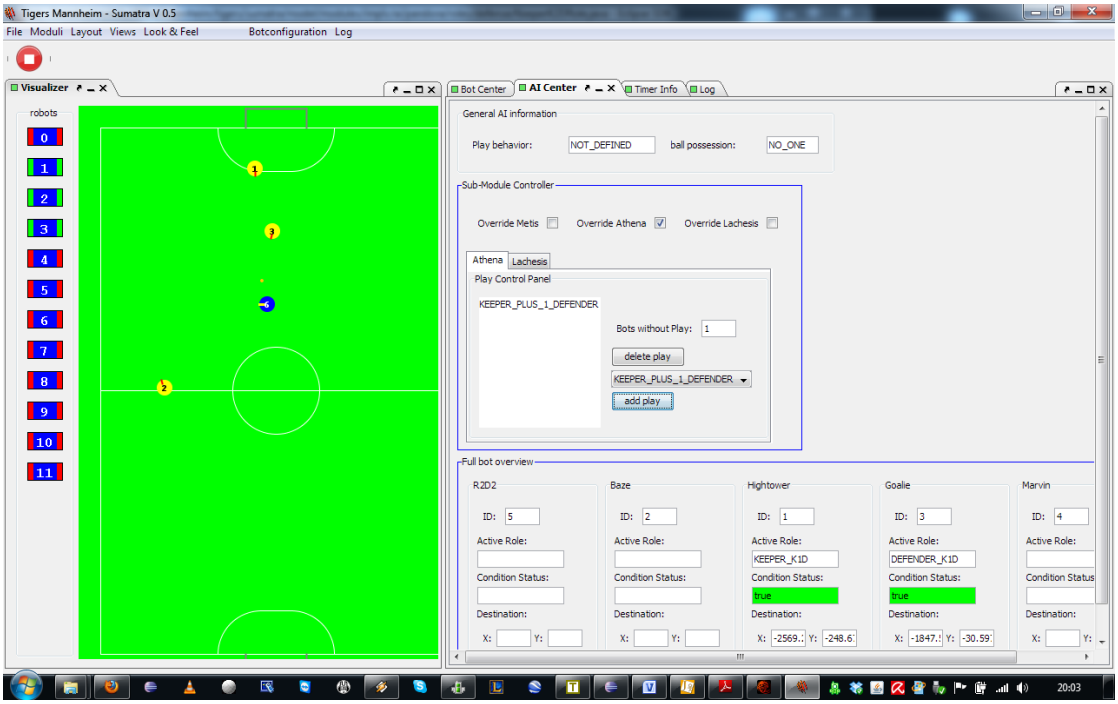

Abbildung A.2.: Grafische Ansicht der Zentralsoftware *Sumatra*