Westfälische Wilhelms-Universität Münster

Thema:

Der RoboCup Soccer Server

Ausarbeitung
im Rahmen des Seminars "Agenten und Robotfußball"

im Fachgebiet Informatik
am Lehrstuhl für Informatik

Betreuer: Dr. Dietmar Lammers

vorgelegt von: Stephan Hagemann
shageman@uni-muenster.de

Vortrag: 30.06.2003

Inhaltsverzeichnis

1 Einleitung

Seit den frühen Neunziger Jahren gibt es die Idee, fußballspielende Roboter in der Forschung zu verwenden. Für 1997 wurde die erste “Roboter Fußballweltmeisterschaft” (RoboCup, Homepage unter [2]) angesetzt, die schließlich mit zugehörigen Konferenzen in Japan ausgerichtet wurde. Seitdem wird jährlich ein solches Turnier in verschiedenen Ländern ausgerichtet.

Die Idee hinter dem RoboCup ist, ein Standardproblem für künstliche Intelligenz und Roboterforschung zu bieten. Dieses Problem soll Forscher begeistern und ein günstiges Testumfeld für den Forschungsfortschritt bieten. Fußball wird in den meisten Teilen der Welt gespielt und hat relativ einfache Regeln. Es unterscheidet sich zu klassischen Forschungsfeldern der KI dadurch, dass es ein Mannschaftsspiel ist und am besten durch Kooperation der Spieler gelöst werden kann. Außerdem werden viele Randgebiete der Roboterforschung berührt, z.B. zeitkritische Systeme und Agenten.

Bei einem RoboCup werden mehrere parallele Turniere in verschiedenen Ligen ausgetragen. Neben den Ligen in denen Roboter unterschiedlicher Größen antreten, gibt es die Simulationsliga, in der eine Fußballsimulation gespielt wird.

Die Simulationsliga basiert auf dem Soccer Server, einem Server, der die Simulation eines Spielfelds mit Spielern übernimmt. Er unterstützt das Spiel mehrerer virtueller Spieler in einer unsicheren, semistrukturierten Multi-Agenten-Umgebung mit Echtzeitanforderungen. Dabei erlaubt das Abstraktionsniveau, sich auf die höheren Funktionen, wie Kooperation und Lernen zu konzentrieren. Grundlegende Fähigkeiten, wie Objektwahrnehmung oder Roboterbewegung müssen nicht programmiert werden. Auch andere Probleme, die bei den Roboter-Ligen auftreten (Hardwareprobleme mit Sensoren, Motoren oder Kommunikation), gibt es bei der Simulation nicht. Da das Verhalten der einzelnen Spieler in dieser Umgebung im Allgemeinen nicht fehlerfrei ist, benötigt man, wie im richtigen Spiel, einen Schiedsrichter. Dieser ist zum Teil als automatischer Schiedsrichter implementiert und erkennt bisher nur einfache Spielsituationen. Die noch nicht implementierten Funktionen werden durch einen menschlichen Schiedsrichter übernommen. Dieser und die anderen Zuschauer können das Spiel über die letzte wichtige Komponente der Simulation, die Monitore, verfolgen. Monitore erhalten vom Server Informationen über die Situation auf dem Spielfeld und stellen diese dann grafisch dar.

Die vorliegende Arbeit stellt den hinter der Simulation stehenden Soccer Server (Homepage unter [1]) vor. Dabei liegt der Schwerpunkt auf die im Server verwendeten Modelle und den Möglichkeiten der Spieler in der Fußballsimulation. In Abschnitt 2.2 wird die Wahrnehmung, in Abschnitt 2.3 die Aktionsmöglichkeiten der Spieler erklärt. In den Abschnitten 2.4 und 2.5 werden noch Erweiterungen des Modells und die Verwendung von Trainern vorgestellt.

2 Der Server

Der RoboCup Soccer Server ist ein Anwendungssystem, das autonomen Agenten erlaubt, gegeneinander simulierten Fußball zu spielen [RCF]. Die Organisation ist dabei ein Client-Server-System, in dem der Server das Spielfeld und alle Bewegungen verwaltet. Ein Agent kontrolliert immer nur einen Spieler.

Die Kommunikation zwischen Client und Server erfolgt üblicherweise über ein LAN. Es gibt aber auch während des gesamten Jahres einen ständigen Wettbewerb über das Internet, der sich zu einem normalen Turnier nur darin unterscheidet, dass die Zeit etwas langsamer fortgeschrieben wird. In beiden Fällen werden die Daten zwischen den Rechnern über UDP ausgetauscht. UDP ist ein verbindungsloses Internetprotokoll, das keine Übertragungsgarantie bietet, das heißt, es gibt keinen Mechanismus, der sicherstellt, dass versandte Pakete auch beim Empfänger ankommen. Dieser Nachteil hat sich als nicht gravierend herausgestellt gegenüber dem Vorteil der sich durch den geringeren Overhead gegenüber TCP ergibt.

Der Inhalt der Nachrichten hat immer die Form von S-Ausdrücken. Diese Ausdrücke sind die Basis von funktionalen Programmiersprachen wie LISP (die erste Version des Soccer Server war in LISP geschrieben) oder Scheme. S-Ausdrücke sind geschachtelte Klammerausdrücke, zu denen man folgende BNF Regeln angeben kann:

$$
\begin{array}{lll}
SExp & \rightarrow (S) & \\
S & \rightarrow SS \quad | \quad (S) \quad | \quad char^* &
\end{array}
$$

Der Torhüter der Mannschaft “S04” würde z.B. folgende Nachricht zur Initialisierung an den Server senden: `(init S04 (version 9) (goalie))`. Der Server antwortet auf eine solche Initialisierung, wenn alles in Ordnung ist, mit mehreren langen S-Ausdrücken, die dem Spieler Informationen über die Parameter des Servers geben.

Der Server ist der einzige der beteiligten Rechner, der das Weltmodell (aufgrund von Aktionen der Spieler) verändern kann, außerdem ist nur diesem alles auf dem Spielfeld vollständig und fehlerfrei bekannt. Die Spieler haben dagegen ein egozentrisches und fehlerbehaftetes Bild des Teils des Spielfeldes, der in ihrem Blickfeld liegt.

Die Simulation wird zeitlich diskret durchgeführt. Eine Halbzeit besteht aus 3000 Zyklen. Standardmäßig ist die Länge eines Zyklus 100ms. In der Internetsimulation wird für einen Zyklus etwas mehr Zeit gegeben: 150ms. Am Ende eines Zyklus wertet der Server die bis dahin aufgelaufenen Aktionen der Spieler aus und berechnet die Positionen, Richtungen und Geschwindigkeiten neu. Wichtig zu bemerken ist, dass die Simulation nur zweidimensional berechnet wird.

Es gibt zahlreiche Einstellungsmöglichkeiten, mit denen das Verhalten angepasst werden kann. So kann vom Turniermodus in einen Trainingsmodus umgeschaltet werden, oder die

Abseitsregel ausgeschaltet werden. Im Folgenden sind diese Parameter nicht im Einzelnen aufgeführt, sondern es wird der Standardfall vorgestellt.

2.1 Spielregeln

In der RoboCup Simulationsliga wird im Wesentlichen nach den offiziellen Regeln des Fußballweltverbandes FIFA gespielt. Einige Änderungen ergeben sich z.B. aus der Zweidimensionalität des Spielfeldes.

Der automatische Schiedsrichter erkennt viele Spielsituationen und korrigiert eventuelle Probleme. Vor dem Anstoß müssen sich die Spieler auf der eigenen Spielhälfte aufhalten. Den Spielern wird etwas Zeit gegeben, um sich richtig zu positionieren, dann wird das Spiel gestartet. Sollte ein Spieler auf die falsche Hälfte kommen, wird er vom Schiedsrichter an eine zufällige Stelle in der eigenen Hälfte gesetzt. Der Schiedsrichter erkennt Tore und leitet dann den Anstoß ein. Bei einem Aus wird der Ball entsprechend positioniert (Seitenaus, Abstoß oder Ecke). Bei Standardsituationen wird der 9,15 m Abstand zum Ball kontrolliert. Des Weiteren überwacht der automatische Schiedsrichter Abseits, Rückpässe und die Spielzeit. Andere Situationen, wie zum Beispiel Blockieren von Tor oder Ball, Sperren ohne Ball oder offensichtlich fehlerhaftes Verhalten, müssen von einem menschlichen Schiedsrichter beobachtet und sanktioniert werden.

Der Server kann auf vielen Wegen hintergangen werden, auch wenn die Spielregeln eingehalten werden. Es ist zum Beispiel üblich, dass die Programme, die die Spieler implementieren, auf einem Rechner laufen. Es wäre ein Leichtes, Prozesse oder Threads vorbei am Soccer Server miteinander Daten austauschen zu lassen. Es gilt als Gentlemans Agreement, dass man dies nicht tut.

2.2 Wahrnehmung

Spieler haben drei Sensoren, mit denen sie ihre Umwelt wahrnehmen können. Der wichtigste dieser Sensoren ist der Sehsinn, aber je mehr die Agenten kooperieren wollen, desto wichtiger wird für sie das Hören. Der dritte Sinn ist der Körpersensor, der den aktuellen Körperstatus liefert.

Das Abstraktionsniveau der Wahrnehmung ist so hoch, dass es für Entwickler nicht um Sensorprobleme geht. Es besteht sofort die Möglichkeit aus den Wahrnehmungen höhere Funktionen, wie Weltmodelle und Taktiken zu entwickeln.

2.2.1 Hören

Nachrichten können von Spielern, den Trainern oder dem Schiedsrichter versandt werden. Sie werden in der Form `(say Message)` gesendet. Alle hörenden Agenten erhalten dann sofort eine Nachricht der Form `(hear Time Sender Message)`. Dabei enthält Message jeweils den Inhalt der Nachricht (darf nicht länger als 512 Zeichen sein), Time den aktuellen Zyklus und Sender entweder Schiedsrichter, Trainer oder die Richtung (-180° bis 180°), aus der das Signal kommt.

Aber nicht alle Agenten hören eine Nachricht, denn das Hören ist in Reichweite und Kapazität eingeschränkt. Man kann Gesagtes bis zu 50 m weit hören. Bis zu dieser Weite tritt keine Hörverschlechterung ein, danach hört ein Spieler allerdings nichts mehr. Die Aufnahmefähigkeit eines Spielers ist auch zeitlich begrenzt. Mit der normalen Einstellung kann nur jeden zweiten Zyklus eine Nachricht von jedem Team gehört werden. Um dies zu implementieren, gibt es eine Kapazitätsvariable, die in jeden Zyklus bis zu einem Maximum um ein Inkrement erhöht wird und beim Hören einer Nachricht verringert wird. Eine Nachricht kann nur gehört werden, wenn die Kapazität dabei positiv bleibt, sonst ist ihr Inhalt undefiniert. Nachrichten des Schiedsrichters kommen allerdings immer bei einem Spieler an.

2.2.2 Sehen

Der Sehsinn ist bei weitem der komplexeste Sensor eines Spielers. Er nimmt die verschiedenen Objekte auf dem Spielfeld wahr. Dazu gehören neben den Mitspielern die Außenlinien, die Tore und Markierungen besonderer Punkte des Spielfeldes. Periodisch erhält jeder Spieler Informationen über die Objekte in seinem Sichtfeld in der Form `(see (ObjName Distance Direction [DistChng DirChng [BodyDir HeadDir]]+))`. ObjName ist der kodierte Name eines Objektes (für den oben erstellten Torwart z.B. `(p S04 1)`). Abbildung 1 zeigt die Objektnamen der Tore, Linien und Markierungen. Distance, Direction enthalten Entfernung und Richtung des Objekts. Falls es ein bewegtes Objekt ist, enthalten DistChng und DirChng die Veränderungen von Distanz und Richtung. Ist es sogar ein Spieler, wird auch die relative Richtung von Körper und Kopf (BodyDir und HeadDir) zurückgegeben.

Informationen über gesehene Objekte sind nicht fehlerfrei: sie können verfälscht oder unvollständig sein. Entfernungen werden tendenziell überschätzt. Für Spieler beträgt der Fehler bis zu 10 %. Im besten Fall sieht man von einem Spieler die Mannschaftszugehörigkeit und die Trikotnummer. Ab einer bestimmten Entfernung nimmt allerdings die Wahrscheinlichkeit, mit der man die Trikotnummer wahrnimmt, linear bis auf null ab. In größerer Entfernung gilt das Gleiche für den Mannschaftsnamen. Mit den Standardwerten kann die Nummer ab 40 m und die Mannschaft ab 60 m nicht mehr gesehen

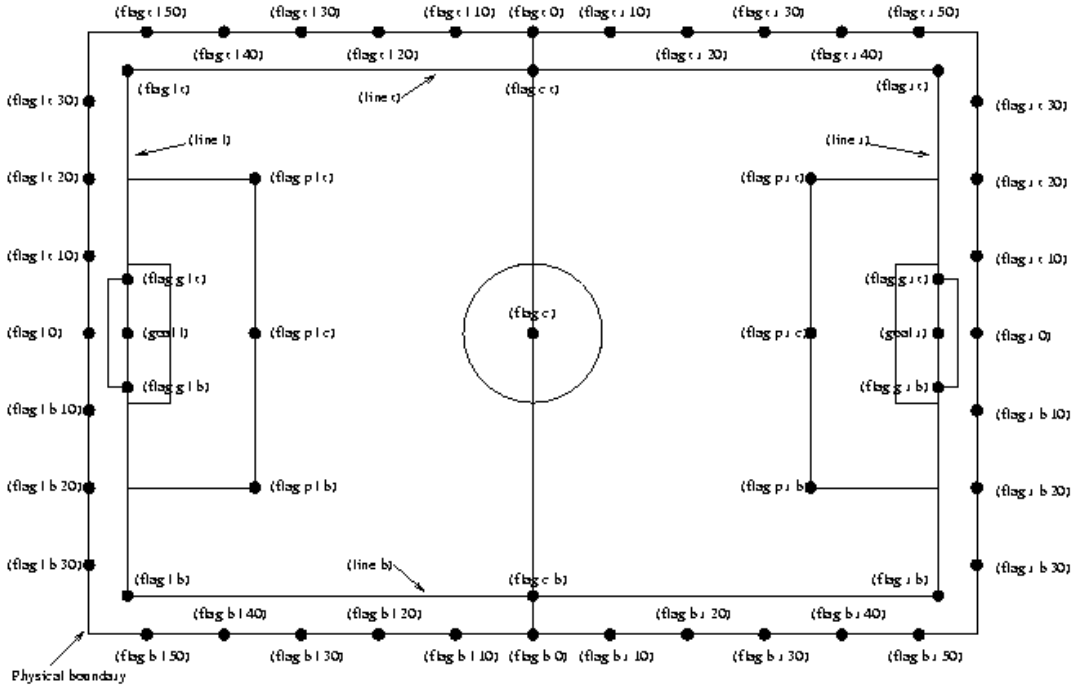

Abbildung 1: Markierungen und Linien auf dem Spielfeld
Das Spielfeld ist 105 m lang und 68 m breit. Das Tor ist etwa 14 m breit (doppelt so breit, wie ein normales Tor, da es keine Höhe hat). Die Markierungen stehen 5 m hinter den Außenlinien.

werden.

Der Sehsinn dient darüber hinaus auch als Nahsensor. In einer engen Umgebung (3 m) um den Spieler nimmt er alle Objekte ihrem Typ nach wahr, auch wenn sie nicht in seinem Blickfeld liegen. Ein Beispiel für die Sicht eines Spielers zeigt Abbildung 2.

Die Eigenschaften der Sicht werden durch die Qualität des Sehens und die Breite des Sichtfeldes bestimmt. Durch die Nachricht `(change_view` *`Width Quality`*`)` kann ein Spieler diese Werte verändern. Dabei kann die Breite 45°, 90° oder 180° betragen. Die Qualität kann hoch oder niedrig sein. Abhängig von diesen Werten ergibt sich die Frequenz, mit der der Spieler die Umgebung wahrnimmt. Im Normalfall nimmt ein Spieler alle 150 ms ein Bild von seiner Umwelt auf.

2.2.3 Körpersensor

Der Körpersensor übermittelt einem Spieler in jedem Zyklus die Daten, die der Server über ihn hat: `(sense_body` *`Time (view_mode ViewQuality ViewWidth) (stamina Stamina Effort)...`*`)` Die Nachricht enthält noch weitere Punkte, die es insgesamt dem Spieler erlauben, die Selbsteinschätzung des eigenen Zustandes mit dem Bild des Servers abzugleichen. Es kommt zu Unterschieden zwischen diesen Sichten durch die fehlerbehaftete Umwelt und eventuell durch untergegangene UDP Pakete.

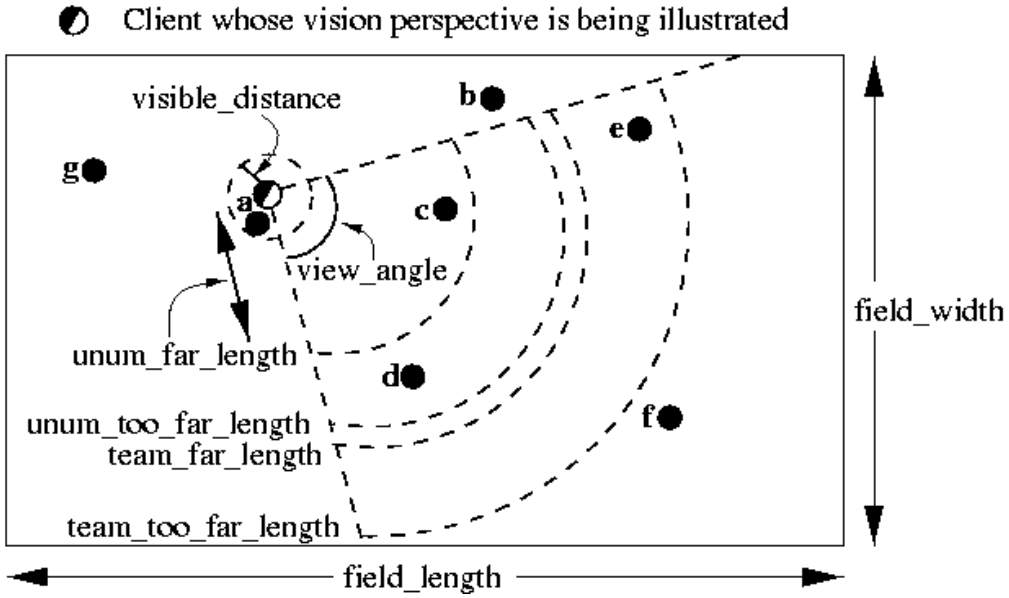

Abbildung 2: Sichtfeld Beispiel

2.3 Aktionen

Hat ein Client sich einen Plan davon gemacht, was er erreichen will, muss er diesen über seine Aktionen noch umsetzen. Dazu hat ein Spieler beim Soccer Server einige Kommandos zur Auswahl. Das Abstraktionsniveau ist wie bei der Wahrnehmung relativ hoch, so dass nicht elementare Körperfunktionen wie Gleichgewicht oder Muskelspannungen implementiert werden müssen. Die zur Auswahl stehenden Kommandos sind: Beschleunigen, Schießen, Fangen, Körper drehen, Kopf drehen. Für die Bewegung außerhalb der Spielzeit gibt es noch eine absolute Positionierungsmöglichkeit. Alle diese Kommandos dürfen pro Zyklus nur einmal an den Server gesendet werden.

2.3.1 Das Bewegungsmodell

Das Bewegungsmodell des Soccer Server ist recht einfach. In fünf Phasen wird die Bewegung aller mobilen Objekte in jeder Phase gleichzeitig durchgeführt:

- Die Beschleunigung wird angewendet
- die Bewegung wird durchgeführt
- die Geschwindigkeit wird reduziert
  Bälle behalten etwa 94% ihrer Geschwindigkeit, Spieler etwa 40%.
- die Beschleunigung wird auf null gesetzt
- Kollisionen werden behoben
  Spieler und Ball werden in der Simulation als Kreise dargestellt. Sollten sich nach der eigentlichen Bewegung mobile Objekte überlagern, werden sie mit

einem Zehntel ihrer Geschwindigkeit zurückbewegt, bis die Kollision aufgehoben ist.

Alle Bewegungen werden mit einem Fehler belegt. Diese Fehler können auf allen Ebenen eingestreut werden: beim Wert des Parameters bei der Beschleunigung, beim Berechnen der Wirkung einer Aktion.

2.3.2 Beschleunigen

Das Dash Kommando beschleunigt einen Spieler in die Richtung, in die aktuell sein Körper zeigt. Das Kommando lautet `(dash` *`Power`*`)`, wobei der Parameter Power angibt, wie viel Kraft der Spieler in die Beschleunigung setzt.

Bei jeder Beschleunigung wird Energie verbraucht. Wie viel Kraft dabei effektiv in die Beschleunigung geht, hängt von der Ausdauer ab, die der Spieler noch hat. Durch die Ausdauer ergibt sich der Einsatz, mit dem ein Spieler agiert. Unterschreitet die Ausdauer 30% des Maximums, reduziert sich dieser und damit die effektive Beschleunigung. Darüber steigt der Einsatz, solange er unter dem Maximum ist. Daneben gibt es die Erholungsrate, mit der die Ausdauer regeneriert wird. Sinkt die Ausdauer unter 30%, sinkt sie langsam dauerhaft ab, bis auf ein Minimum von 50%. Die Erholungsrate wird nur in der Halbzeit auf den Ursprungswert zurückgesetzt.

2.3.3 Schießen

Das Kommando zum Schießen des Balles lautet `(kick` *`Power Direction`*`)`. Dabei gibt Power die Kraft des Schusses an und Direction die Richtung, in die der Ball geschossen wird. Die Richtung kann zwischen -180° und 180°, also auch als “Hackentrick” nach hinten, betragen. Wie bei der Beschleunigung kann die effektive Kraft niedriger sein als die eingesetzte. Je mehr der Ball hinter dem Spieler liegt, desto geringer ist die Kraftübertragung. Dies kann die effektive Schusskraft um bis zu 25% senken. Nur wenn der Ball sich in einem nahen Abstand zum Spieler befindet kann dieser ihn schießen. Bei der größten noch erreichbaren Entfernung beträgt der Kraftverlust 25%. Beide Reduktionen werden addiert, so dass es im schlechtesten Fall zu einer Kraftreduktion um 50% kommt.

2.3.4 Fangen

Der Torwart ist der einzige Spieler, der den Ball auch fangen kann. Um einen Fang zu versuchen, schickt er eine Nachricht der Form `(catch` *`Direction`*`)` an den Server. Die Richtung gibt an, in welche Richtung relativ zu seiner Blickrichtung er versuchen möchte zu fangen. In dieser Richtung wird, wie in Abbildung 3 zu sehen, ein Fangbereich

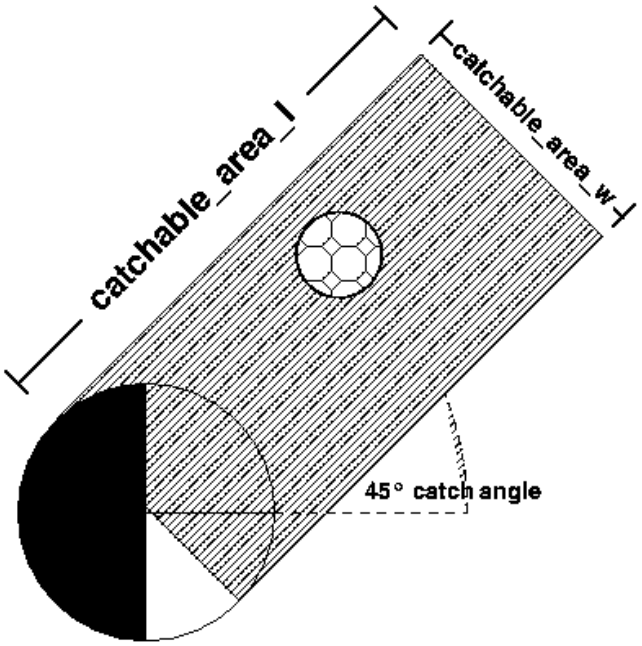

Abbildung 3: Der Fangbereich

aufgespannt. Üblicherweise ist dieser Bereich 2m lang und 1m breit. Befindet sich der Ball am Ende eines Zyklus innerhalb dieses Bereiches, fängt der Torwart den Ball mit einer bestimmten Wahrscheinlichkeit (die normalerweise allerdings 1 ist).

Sollte ein Fangversuch erfolglos sein, dann können für eine bestimmte Zeit keine weiteren Versuche gemacht werden. Die Interpretation ist einfach: wenn ein Torwart nach dem Ball springt, um ihn zu fangen, muss er erst wieder aufstehen, bevor er es erneut versuchen kann.

Eine weitere Besonderheit gibt es, wenn das Fangen erfolgreich war. Der Torwart darf dann innerhalb des Strafraums bis zu zweimal die absolute Positionierung über `move` durchführen. Diese Regel könnte eingeführt worden sein, um es dem Torwart zu ermöglichen, etwas Platz für einen Abschlag zu bekommen.

2.3.5 Körperdrehung

Turn ermöglicht es dem Spieler, seine Richtung zu wechseln. Das Kommando dazu lautet einfach `(turn` *`Moment`*`)`, wobei Moment den gewünschten Drehwinkel angibt. Dabei ist in die Bewegung ein Trägheitsfaktor eingebaut, so dass Drehungen bei einer hohen Geschwindigkeit nicht so schnell geschehen können. Bei höchster Geschwindigkeit kann die effektive Drehung nicht größer als 60° sein. Aus dem Stand kann man sich in einem Zyklus um 90° drehen.

2.3.6 Kopfdrehung

Der Spieler kann den Kopf unabhängig vom Körper bewegen. Das Kommando ist völlig analog zum Kommando zur Drehung des Körpers: `(turn_neck` *`Moment`*`)`. Der maximale Winkel des Kopfes zum Körper ist 90° nach links und rechts. Dabei ist der Kopf in einem Zyklus voll beweglich, kann also jeden Winkel erreichen. Wichtig zu wissen ist, dass sich der Kopf ohne ein `turn_neck` fest zum Körper ist. Das bedeutet, dass bei einer Körperdrehung der Kopf sich automatisch auch bewegt.

2.3.7 Move

Move ist ein einfaches Kommando, das eine absolute Positionierung des Spielers an die Position `(X,Y)` ermöglicht: `(move` *`X Y`*`)`. Damit ist Move die einzige Aktion, die nicht egozentrisch ist. Das ist auch der Grund, warum es nicht im normalen Spielmodus verwendet werden darf. Nur zu Beginn der Halbzeiten und nach Toren dürfen Spieler über dieses Kommando bewegt werden. Durch dieses Kommando wird außerdem auch Energie verbraucht.

2.4 Erweiterung des Modells

Die Faszination von Fußball lebt auch davon, dass es unterschiedliche Spielertypen gibt. Nicht alle Spieler sind gleich stark oder haben die gleichen Fähigkeiten. Dies kann ein wenig im Soccer Server nachgebildet werden. Das Modell erlaubt es offensichtlich nicht, Links- von Rechtsfüssern zu unterscheiden, aber es gibt auch auf dem Abstraktionsniveau des Soccer Server viele Möglichkeiten. Prinzipiell können sich Spieler in fast allen Parametern, die einen Spieler betreffen, unterscheiden: bei der Beschleunigung, Kraft, Ausdauer oder Erholung.

Dazu legt der Server vor dem Spiel verschiedene Spielertypen an und meldet diese an jeden Client, der sich verbindet. Wichtig ist, dass die Gesamtstärke der Teams dadurch gleich bleibt, dass beide Teams die gleichen heterogenen Spieler erhalten.

Die Fehler, die bisher in der Simulation auftauchen, kommen alle aus der Egoperspektive der Spieler. Man kann sagen, sie können Wahrnehmung und Aktionen nicht völlig genau steuern. Dieser Fehler ist aber unsystematisch. Wind verändert auch das Ergebnis von Bewegung, allerdings systematisch. Standardmäßig ist der Einfluss von Wind zwar ausgeschaltet, aber man kann Wind einführen und dazu seine Richtung, die Stärke einstellen. Sogar Böen sind möglich, durch einen zufälligen Faktor bei der Erzeugung des Windeinflusses.

2.5 Trainer

Bevor Mannschaften in der Simulationsliga spielen können, müssen sie natürlich entwickelt werden. Das manuelle Testen der Spieler in der Simulationsumgebung ist zu Anfang vielleicht nur unhandlich, aber spätestens beim Einsatz von neuronalem Netzen zum Lernen völlig ungeeignet. Es liegt auf der Hand, dass ein Tool zum automatischen Steuern des Servers sinnvoll ist. Dazu wird beim Soccer Server, wie für die Spieler auch, ein Client verwendet: der Trainer.

Der Trainer hat sehr weite Möglichkeiten, den Ablauf der Simulation im Server zu steuern. Er kann alle Objekte mit ihren absoluten Positionen fehlerfrei auf dem Spielfeld sehen. Er kann den Spielstatus verändern und auch die Positionen und Geschwindigkeiten mobiler Objekte neu setzen. Darüber hinaus kann er mit den Spielern kommunizieren. Es ist sogar möglich, ein Spiel ohne den automatischen Schiedsrichter zu spielen.

Der Trainer darf nicht bei Turnieren verwendet werden, man kann ihn aber hervorragend beim Testen verwenden, um z.B. immer wieder die Situation “zwei Stürmer gegen den Torwart” zu üben.

Zwar darf der Trainer sich während offizieller Spiele nicht mit dem Server verbinden, aber es gibt einen weiteren Client, den Coach, der seine Rolle dort übernehmen kann. Dessen Privilegien sind gegenüber denen des Trainers allerdings stark eingeschränkt. Die Idee ist, dass es einen Client gibt, der nicht seine Zeit dafür aufbringen muss auf seine Umwelt zu reagieren, sondern sich mehr mit der Spielstrategie befassen kann.

Der Coach sieht genau wie der Trainer die Positionen und Geschwindigkeiten aller Objekte. Er darf eingeschränkt mit den Spielern kommunizieren. Auf den Spielverlauf selbst hat er keinen Einfluss.

Die Kommunikation zwischen Coach und den Spielern ist mehrfach eingeschränkt. Zum einen darf der Coach nur alle 300 Zyklen Nachrichten versenden. Damit soll verhindert werden, dass er die Steuerung der Spieler kontrollieren kann. Des Weitern ist die Sprache der Nachrichten, die der Coach versenden darf, vorgegeben. Diese soll die Forschung auf diesem Gebiet vorantreiben und es erlauben, dass Coaches unabhängig von Teams entwickelt werden können. In dieser Sprache sind z.B. Konstrukte zur Definition von Bedingungen, Aktionen und Regionen gegeben. Dann können Regeln auf diesen Definitionen aufbauen.

3 Bewertung und Ausblick

Fußball ist eine Sportart, für die sich auf der Welt viele Menschen begeistern können. Die Regeln des Spiels sind einfach genauso wie die ersten Versuche es zu spielen. Aber das Spiel zur Perfektion zu bringen, ist sehr schwierig, auch weil es erforderlich ist in einer Mannschaft mit 11 Spielern ein erfolgreiches Zusammenspiel aufzubauen. Fußball ist aus diesen Gründen sehr gut geeignet, als Standardproblem der Roboter- und KI-Forschung zu dienen.

Der Soccer Server bietet eine sehr gute Simulation von Fußball als Multiagentensystem. Er zeichnet sich besonders durch seine große Verfügbarkeit aus. Da er freie Software ist, ist sein Source Code verfügbar und frei auf verschiedene Systeme portierbar (wenn nicht sogar schon eine Binärdatei existiert). Das Programm ist einfach zu bedienen und braucht zur Ausführung keine besonderen Ressourcen. Die Verwendung von UDP macht die Server Architektur von den Clientsystemen unabhängig, zur Implementierung kann jede UDP-fähige Programmiersprache verwendet werden.

Das System könnte noch so gut sein, es würde nicht genutzt, falls es kein Interesse, also keine interessanten Forschungsfelder, ansprechen würde. Doch um den Soccer Server ergeben sich viele Felder, in denen zurzeit aktiv entwickelt wird.

- Verbesserung der Monitore, vielleicht mit integriertem Spielkommentator
- Erweiterung der Funktionen des automatischen Schiedsrichters
- Weiterentwicklung der Coaches
- Grundlagen der Durchführung einer solchen Simulation

Der Soccer Server ist jetzt schon eine sehr gute Plattform für das Forschen mit Multiagentensystemen. Es sind Weiterentwicklungen denkbar, in denen das Weltmodell nicht mehr nur zweidimensional ist oder in denen die Bewegung und die Ausdauer realistischer modelliert werden. Vielleicht könnte der Schiedsrichter auch vom Server getrennt werden und als eigener Client auf dem Spielfeld agieren? Wem Fußball Spaß macht, dem fällt sicherlich noch mehr ein.

Literatur- und Webseitenverzeichnis

[1] *RoboCup Homepage* – URL http://www.robocup.org – Zuletzt abgerufen am 01.07.2003

[2] *RoboCup Soccer Simulator Homepage* – URL http://sserver.sourceforge.net/ – Zuletzt abgerufen am 01.07.2003

[RCF] The RoboCup Federation: *RoboCup Soccer Server Users Manual (for Soccerserver Ver. 7.07 and later)*, 11.02.2003