MITRA: Aktionsauswahl im Tischfußball

Diplomarbeit an der Universität Freiburg

Fakultät für Angewandte Wissenschaften

vorgelegt von:

Moritz Tacke

Betreuer: Prof. Dr. B. Nebel

2003

Inhaltsverzeichnis

Zusammenfassung

Tischfußball ist ein aufstrebendes neues Feld zum Testen von Verfahren der Künstlichen Intelligenz (KI) in der Realität. Vorreiter dieser Entwicklung ist der *Ki*cker-*Ro*boter *KiRo*, der es bereits heute möglich macht, Tischfußball gegen einen automatischen Gegner zu spielen. In der vorliegenden Arbeit wird das *KiRo*-System um eine Komponente zur Aktionsauswahl erweitert, die es ermöglicht, Abschätzungen über zukünftige Entwicklungen des Spieles in die Steuerung des automatischen Spielers einfließen zu lassen. Das vorgestellte System, MITRA, bewährte sich sowohl in der Realität als auch in Simulationsexperimenten.

Erklärung

Name: Moritz Tacke   Matrikelnummer: 9271551

Ich erkläre hiermit, diese Diplomarbeit selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet zu haben.

Freiburg, den 25. 6. 2003

Danksagungen

An dieser Stelle möchte ich mich vor allem herzlich bei Thilo Weigel bedanken, der diese Arbeit von der Aufgabenstellung bis zur Abgabe ständig begleitet und ihr Bild mitgeprägt hat. Für Hilfestellungen von kleinen Tipps zur Implementierung über Hinweise auf Literaturstellen bis zum Korrekturlesen dieser Arbeit gilt mein Dank Sebastian Kupferschmid, Matthias Güde, Maren Bennewitz, Dirk Zugenmaier, Johannes Seither sowie Maurus Tacke.

Schließlich danke ich Professor Nebel für die freundliche Betreuung und positiven Diskussionen.

Kapitel 1

Einleitung

Die Vision von selbständig handelnden Maschinen beflügelt die menschliche Phantasie schon seit Jahrhunderten. Während in früheren Zeiten die Versuche, solche Apparate tatsächlich zu konstruieren, entweder scheiterten oder über ein sehr einfaches Stadium nicht herauskamen, rückte die Verwendung von Computern die Realisierung dieser Vorstellungen in den Bereich des Möglichen. Von Anfang an waren diese Gedanken begleitet von der Idee des Wettkampfs zwischen Mensch und Maschine, wobei dieser sowohl auf friedlichen Wege, in Sport und Spiel, als auch – vor allem in der Sciencefictionliteratur – im Versuch gegenseitiger Vernichtung bestehen sollte. Diese Diplomarbeit befasst sich mit der ersten Variante.

Der klassische Schauplatz des Wettstreits zwischen Mensch und Maschine war und ist das Brettspiel, da es eine Abstraktion von der wirklichen Welt bietet, die den Möglichkeiten des Computers sehr entgegen kommt. Russel und Norvig schreiben hierzu [Russel und Norvig 1995]:

> Der Zustand eines Spieles ist einfach darzustellen, und die Agenten sind normalerweise auf eine kleine Anzahl wohldefinierter Aktionen eingeschränkt. [...] Weniger abstrakte Spiele, wie Kricket oder Fußball, haben nicht viel Interesse in der Künstlichen Intelligenz auf sich ziehen können.

Diese Einschätzung ist in den letzten Jahren relativiert worden: Vor allem das

Fußballspiel ist durch die RoboCup-Initiative [Kitano et al. 1997] verstärkt in den Fokus von Forschung und Öffentlichkeit geraten. Das Endziel ist hier, mit einer aus Robotern bestehende Fußballmannschaft im Jahre 2050 den amtierenden – menschlichen – Fußballweltmeister zu schlagen. Auf der Suche nach Möglichkeiten, Mensch und Maschine in etwas näherer Zukunft im sportlichen Wettkampf zu vereinen, gedieh die Idee des Tischfußballroboters: Das Spielfeld ist klein und der jeweilige Spielzustand schnell und kompakt zu beschreiben. Die Regeln sind einfach, die Anzahl möglicher Handlungen ist sehr gering – aber dennoch muss ein Roboter, der erfolgreich Tischfußball spielen soll, in der wirklichen Welt mit den in ihr vorkommenden Problemen wie ungenauen Sensoren und unpräzisen Aktuatoren bestehen und dabei anspruchsvolle sensormotorische Leistungen in Echtzeit erbringen. Ein weiterer Reiz liegt darin, dass Tischfußball ein populäres Spiel ist. Neben der weiten Verbreitung als Freizeitvergnügen existieren auch organisierte Wettkämpfe bis hin zur Weltmeisterschaft. Der erste Prototyp eines Tischfußballroboters ist das *KiRo*-System (*KiRo* steht für *Ki*cker-*Ro*boter), das momentan aufgrund laufender Verbesserungen auf dem bestem Wege ist, dem Menschen ein ernsthafter Konkurrent zu werden.

Mit dieser Arbeit soll ein Beitrag hierzu geleistet werden, indem die Aktionsauswahl, die im ursprünglichen *KiRo*-System rein reaktiv modelliert war, durch eine vorausschauend agierende ersetzt wird. Diese neue Aktionsauswahl besteht aus einer abgewandelten Version des aus dem Bereich der Brettspiele bekannten Minimax-Algorithmus.

1.1 Zielsetzung

Ziel ist, das Tischfußballspiel des bereits vorhandenen *KiRo*-Systems durch die Einführung einer fortgeschrittenen Aktionsauswahl zu verbessern. Zu Realisierung dieser im folgenden auch MITRA (für **Mi**nimaxbasierte **T**ischfussball**r**oboter**a**ktionsauswahl) genannten Aktionsauswahl waren gefordert:

- Ein erweiterbarer, gut handhabbarer Satz grundlegender Aktionsmöglichkeiten wie Schüsse, Blocks etc.
- Ein Planungsmodul, das die Güte ausgewählter Aktionen durch eine

Beurteilung ihrer Konsequenzen evaluiert. Dieses Planungsmodul soll die Verhältnisse am realen Tisch bestmöglich widerspiegeln.

- Weiterhin muss es die Möglichkeit bereitstellen, durch die Auswahl verschiedener Gegnermodelle eine einfache Anpassung an unterschiedliche Gegner zu schaffen.

- Da es sich bei Tischfußball um ein schnelles Spiel handelt, ist die Laufzeit dieses Verfahrens engen Beschränkungen unterworfen.

1.1.1 Gutes Tischfußballspiel

Um das Erreichen der in der Zielsetzung geforderten Verbesserung des Tischfußballspiels beurteilen zu können, muss zunächst Klarheit darüber geschaffen werden, was „gutes Tischfußballspiel“ bedeutet. In unseren Augen zeichnet sich ein gut Tischfußball spielender Roboter in zwei Kriterien aus:

1. **Spielstärke:** Dieses Kriterium ist das nächstliegende und das wichtigste: Ein Tischfußballroboter, der von jedem Gegner ohne die geringsten Probleme geschlagen wird, kann schwerlich als „gut tischfußballspielend“ bezeichnet werden.

   Die Spielstärke lässt sich im Prinzip sehr einfach quantitativ messen: Sowohl die Anzahl der erzielten Tore als auch der Schwerpunkt des Spieles lassen auf die Überlegenheit eines der Spieler schließen.

   Dennoch darf die Spielstärke nicht das einzige Kriterium bleiben: Ein mit starken Elektromotoren ausgestatteter Roboter, der die Stangen mit höchstmöglicher Geschwindigkeit rotiert und unvorhersehbar verschiebt, darf wohl als nahezu unbezwingbar gelten; dennoch wird ihn niemand als einen „guten“ Tischfußballspieler bezeichnen. Dies führt zur Einführung eines anderen Kriteriums:

2. **Ruhiges Spiel:** Der Roboter soll sich möglichst sparsam bewegen. Dieses Kriterium schließt sowohl den Verzicht auf das Rotieren der Stangen als auch den Versuch, diese so wenig und langsam wie möglich zu bewegen, ein. Durch hektische Bewegungen wird nicht nur der intuitive Eindruck, der Roboter spiele tatsächlich gut Tischfußball, zerstört, es ent-

steht auch eine erhebliche Lärmbelastung und die mechanischen Komponenten zeigen einen erhöhten Verschleiß, was sich vor allem durch den Bruch von Kupplungen bemerkbar macht.

Während das erste Kriterium einfach quantitativ messbar ist, ist dies bei der Forderung nach ruhigem Spiel schwieriger. Aus diesem Grunde wurde dieser Punkt als eine Richtschnur beim Entwurf und der Implementierung des Systems verwendet, indem diesem Anspruch nicht genügende Ausführungen der Komponenten des Systems fallengelassen wurden. In den Experimenten ist das Kriterium „ruhiges Spiel" deshalb nicht aufgeführt.

Leider entsteht durch die gleichzeitige Berücksichtigung beider Kriterien oft ein Zielkonflikt: Alle Bestrebungen, den Tischfußballroboter ruhiger spielen zu lassen, bauen letzten Endes auf ein erhöhtes Maß von Antizipation der Ballbewegung; diese aber ist mit den zur Verfügung stehenden Mitteln noch unzuverlässig und führt zu einer effektiv verminderten Spielstärke. Diese Güterabwägung bestmöglich vorzunehmen erwies sich als eine der Hauptschwierigkeiten bei der Implementierung der Aktionen.

1.2 Aufbau dieser Arbeit

Diese Arbeit ist folgendermaßen aufgebaut: Nach einem Überblick über verschiedene in der Literatur beschriebene Realisationen von Aktionsauswahlmechanismen in Kapitel 2 vervollständigt eine Beschreibung des *KiRo*-Systems in Kapitel 3 die Erläuterung der Grundlagen dieser Arbeit.

Kapitel 4 gibt dann eine Einführung in die Konzepte von MITRA. Im Kapitel 5 wird das für MITRA verwendete Weltmodell vorgestellt. Kapitel 6 erläutert die derzeit zur Verfügung stehenden Aktionen: Dies sind die atomaren Verhaltensweisen, deren Auswahl die Aufgabe von MITRA ist. Außerdem findet sich hier eine Beschreibung des Lernverfahrens, das verwendet wird, um die Auswirkungen der Aktionen abschätzen zu können. In Kapitel 7 wird beschrieben, wie diese Aktionsauswahl im Detail realisiert ist. Die Arbeit schließt mit der Beschreibung der experimentellen Ergebnisse in Kapitel 8 und einer Zusammenfassung nebst weiterem Ausblick in Kapitel 9.

Kapitel 2

Mechanismen zur Aktionsauswahl

Wie viele andere Fragestellungen im Kontext der Künstlichen Intelligenz wird das Aktionsauswahlproblem bezogen auf Agenten [Russel und Norvig 1995] formuliert. Pattie Maes tut dies folgendermaßen [Maes 1990]:

> Wie kann ein Agent zu einem bestimmten Zeitpunkt, in einer bestimmten Situation die „passendste“ oder „wichtigste“ nächste Aktion auswählen?

Ein Lösungsansatz zu diesem Problem wird als „Aktionsauswahlmechanismus“ oder, verkürzend, als „Aktionsauswahl“ bezeichnet.

Es existieren Unmengen verschiedener solcher Mechanismen; diese sind aus den unterschiedlichsten Gedankengängen motiviert. Häufig beispielsweise wird in der Begründung der Verfahren auf ähnliche Vorgehensweisen von Tieren oder auf psychologische Vorgänge beim Menschen hingewiesen. Durch die Vielzahl verschiedener Ansätze würde eine auch nur annähernd vollständige Aufzählung der wichtigsten Verfahren über den hierfür zur Verfügung stehenden Raum herausschießen; ein Artikel, der eine große Anzahl verschiedener Ansätze vorstellt sowie den Versuch einer Kategorisierung unternimmt, ist von Paolo Pirjanian [Pirjanian 1999]. Die folgende Aufführung basiert auf diesem Artikel sowie auf der Diplomarbeit von Klaus Müller [Müller 2000]. Eine kurze, gute Einführung in die Prinzipien der Aktionsauswahl bietet auch eine Veröffentlichung von Jay Gowdy [Gowdy 2000].

Allen im folgenden aufgeführten Verfahren ist eines gemeinsam: Sie funktionieren jeweils in einem spezifischen Kontext und sind somit relevant. Dies darf aber nicht darüber hinwegtäuschen, dass es sich oft um Vorgehensweisen handelt, die ihre Berechtigung ausschließlich hieraus beziehen: Allgemein gilt ein Verfahren dann als berechtigt, wenn es relevant oder elegant ist. Nicht alle im folgenden vorgestellten Prinzipien erfüllen die zweite Bedingung; insbesondere lassen einige den Eindruck entstehen, eher wegen ihrer einfachen Verständlichkeit für Menschen als aufgrund ihrer besonderen Eignung für die Bewältigung der entsprechenden Aufgaben zu existieren.

Eine Hauptkriterium zur Charakterisierung der unterschiedlichen Verfahren ist die Einteilung in die Kategorien „planend“ beziehungsweise „reaktiv“.

2.1 Planende Aktionsauswahl

Ein planend vorgehender Agent verfügt im Allgemeinen über eine interne Repräsentation der Umgebung und hat explizite Kenntnis über seine Ziele. Die Aktionsauswahl besteht in diesem Fall darin, für die gegebene Situation unter Verwendung eines Planungsalgorithmus einen Handlungsplan zu entwerfen, der zum Erreichen eines Zieles führt. Diesen Ansatz verfolgte beispielsweise der Roboter *Shakey* [Nilsson 1984], bei dem der *STRIPS*-Planungsformalismus zum Einsatz kam [Nilsson und Fikes 1971].

Die frühesten Verfahren zur Aktionsauswahl basierten auf Planungsalgorithmen; allerdings zeigte sich, dass diese für die reale Welt nur eingeschränkt einsetzbar sind: Sie sind im Allgemeinen sehr zeitaufwendig, wobei der Zeitverbrauch mit der Komplexität der Umgebung und der Schwierigkeit der Ziele steigt. Aus diesem Grunde muss versucht werden, die Darstellung der Umgebung so einfach wie möglich zu halten; je stärker allerdings vereinfacht wird, desto weiter entfernt sich das interne, zur Aktionsauswahl verwendete Weltmodell von der tatsächlichen Umgebung, und desto größer wird die Wahrscheinlichkeit, dass ein gefundener Plan sich in der Realität nicht bewähren wird. Weiterhin ist typischen Planungsalgorithmen gemein, dass sie von einer – außer durch die Ausführung des Planes – unbewegten Welt ausgehen. Auch diese Annahme hält der Realität nicht stand.

2.2 Reaktive Aktionsauswahl

Diese Schwierigkeiten führten zur Proklamation eines alternativen Ansatzes, der reaktiven Aktionsauswahl. Der Agent reagiert hierbei direkt auf die Reize aus seiner Umgebung, ohne ein Modell dieser zu besitzen oder auch nur explizite Ziele zu kennen. Als einer der konsequentesten Verfechter dieses Modells formuliert Brooks das Credo dieses Ansatzes folgendermaßen: *The world is its own best model* [Brooks 1986].

Eine weitere Überlegung ist, dass sich komplexe, zielgerichtete Verhaltensweisen aus dem Zusammenspiel mehrerer einfacherer ergeben. Das komplexere Verhaltensmuster wird dann auch als *emergent* bezeichnet. Diese Grundverhaltensweisen selbst reagieren direkt auf Vorgänge in der Umgebung des Agenten.

In ihrer einfachsten Form sind reaktive Verhaltensweisen durch hierarchisierte Regeln gegeben, die den Sensorinput auf Verhaltensmuster abbilden. Der Vorteil liegt darin, dass sich einfache Verhaltensweisen so sehr schnell – und zwar sowohl in Entwicklungs- wie auch Laufzeit – realisieren lassen. Dieser Ansatz zeigt Schwächen, wenn explizite Ziele erreicht werden müssen, und skaliert sehr schlecht mit der Komplexität der Aufgaben. Insbesondere ist die monolithische Realisierung von Nachteil, wenn neue Regeln eingefügt werden sollen, da in diesem Fall oft die gesamte durch die Schachtelung gegebene Hierarchie überprüft und gegebenenfalls verändert werden muss.

Einen möglichen Ausweg aus diesem Problem soll der *Subsumptionsansatz* bieten [Brooks 1991]. Hier werden sogenannte Verhalten wie z.B. die Kollisionsvermeidung eines mobilen Roboters in Schichten angeordnet, wobei die grundlegendsten Verhaltensweisen in den unteren Schichten zu finden sind und sich durch die weiter oben angesiedelten, komplexeren Verhalten hemmen lassen. Unter Verwendung dieses Ansatzes ließen sich tatsächlich Aufgaben wie das Explorieren einer Umgebung robust implementieren; jedoch krankt auch dieses Modell daran, dass mit dem Ansteigen der Anforderungen und der Anzahl der Schichten die Schwierigkeiten der Implementierung ansteigen. Insbesondere ist es bei mehreren, komplexen Verhaltensweisen nicht mehr einfach, eine klare Hierarchie unter diesen zu finden, je nach Kontext vielleicht sogar unmöglich.

Einen Ansatz zur Implementierung eines solchen Verhaltens bieten die *diskretes-Ereignis-Systeme* von [Košecká und Bajcsy 1993], die auf der Basis endlicher Automaten reaktive Verhaltensweisen realisieren.

Weiterhin bietet die *Fuzzylogik* einen Weg, um die Entscheidung für bestimmte Aktionen zu fällen; ein Beispiel für den Einsatz von Fuzzy-Lösungen ist die *Saphira*-Architektur [Konolige et al. 1997]. Inhärent ist diesen Herangehensweisen trotz ihrer weiten Verbreitung allerdings oft, dass die verwendeten „Fuzzy"-Funktionen einen eher willkürlichen Eindruck machen.

2.3 Gemischte Verfahren zur Aktionsauswahl

Neben rein reaktiven oder rein planenden Verfahren gibt es noch eine große Palette von Algorithmen, die sowohl reaktive als auch planende Komponenten beinhalten. Dies kann sowohl in einer Art und Weise geschehen, die die „Ratschläge" verschiedener Module in die Entscheidung für eine bestimmte Verhaltensweise einfließen lässt, als auch in einer einzelnen Komponente zusammengefasst geschehen.

Beispielhaft für letzteres sind die *Verhaltensnetzwerke* von [Maes 1990]: Diese lassen sich durch das Setzen von Parametern zwischen eher reaktivem und eher planendem Vorgehen einstellen – „eher" planend, da die Verhaltensnetzwerke ohne explizite Pläne auskommen, sondern die einzelnen Aktionen sich gegenseitig in ihrer Ausführbarkeit begünstigen oder hemmen. Das Grundprinzip ist hierbei, dass Aktionen ausgeführt werden, die sich durch die höchste „Aktivierungsenergie" auszeichnen. Die Aktivierungsenergie wird in jedem Zyklus anhand der Struktur des Netzwerks neu verteilt. Beispielsweise fließt Aktivierungsenergie von bestimmten Zielen zu Aktionen, die zum Erreichen dieser Ziele führen. Diese geben die Aktivierungsenergie teilweise an andere Aktionen weiter, die zur Erfüllung der Vorbedingungen der zielführenden Aktionen beitragen.

Eine Weiterentwicklung der Verhaltensnetzwerke, die sogenannten *Erweiterten Verhaltensnetzwerke* [Dorer 1999], werden im dem Tischfußball verwandten Kontext des Roboterfußballs eingesetzt. Die Mannschaft der Universität Freiburg, *CS Freiburg*, verwendet erfolgreich erweiterte Verhaltensnetzwerke zur Aktionsauswahl [Weigel et al. 2002].

Ein Verfahren zur Verknüpfung mehrerer – reaktiver oder planender – Module ist die *DAMN*-Architektur [Rosenblatt 1997], in der mehrere „Verhaltensweisen" für oder gegen zur Auswahl stehende Aktionen votieren. Die Verhaltensweisen sind hierbei gewichtet; die Gewichte müssen allerdings manuell vorgegeben werden. Die Abstimmung der Verhaltensweisen wird durch ein zentrales Modul, den sogenannten „Voter", interpretiert, das dann die als Sieger hervorgegangene Aktion ausführt. *DAMN* ist als verteilte Architektur realisiert, was zu einer hohen Toleranz gegenüber Ausfällen einzelner Bestandteile führt – falls der „Voter" nicht Opfer eines Ausfalles ist.

Neben der monolithischen Zusammenfassung von planenden und reaktiven Verhaltensweisen einerseits und der prinzipiell gleichwertigen Aufspaltung der Aktionsauswahl in mehrere Verhaltensweisen, die miteinander wetteifern, gibt es auch hierarchische Architekturen. Hierbei wird das System, ähnlich wie bei der Subsumptionsarchitektur, auf mehrere Schichten verteilt. Die oberen Schichten sind Sitz der explizit zielgerichteten, planenden Verhaltensweisen, während in den unteren Ebenen eher reaktive Modelle realisiert werden. Die Häufigkeit, mit der die einzelnen Schichten in den Lauf der Dinge eingreifen, nimmt von oben nach unten zu.

Ein Beispiel hierfür ist die *NASREM*-Architektur [Albus et al. 1987], die aus insgesamt sechs Schichten besteht. In diesem Fall verfügt jede Schicht über ein explizites Weltmodell, das um so abstrakter wird, je höher die Schichten angesiedelt sind.

Wie auch bei der Subsumptionsarchitektur stellt sich auch hier die Frage, nach welchen objektiven Kriterien eine solche Schichtung aufgebaut werden soll.

2.4 Nutzenbasierte Aktionsauswahl

Nutzenbasierte Ansätze können sowohl in planenden als auch in reaktiven Aktionsauswahlsystemen eingesetzt werden. Ihnen gemein ist, dass sie durch ihre Entscheidungen den Wert einer Nutzenfunktion zu maximieren trachten. Eine Nutzenfunktion bildet einen Zustand, eine Aktion oder eine Kombination beider auf die Menge der reellen Zahlen ab, wobei die Güte einer Eingabe als um so höher angesehen wird, je höher der für sie zurückgelieferte Wert ist. Eine nutzenbasierte Aktionsauswahl entscheidet sich dann für die am höchsten bewertete Alternative.

Stone und McAllester verwenden einen ganzen Satz manuell generierter Nutzenfunktionen zur Steuerung eines Teams in der *RoboCup*-Simulationsliga [Stone und McAllester 2001]. Das Vorgehen besteht hierbei darin, alle möglichen Optionen für den ballführenden Spieler zu untersuchen und sich anhand der zu erwartenden Konsequenzen für die am besten erscheinende zu entscheiden. Zu beachten ist hierbei, dass zu dieser Untersuchung die zukünftige Entwicklung des Spielgeschehens mit einbezogen wird. Dies geschieht durch die Berechnung der Zeitpunkte, zu denen die einzelnen Spieler frühestens in den Ballbesitz gelangen können. Es wird also mithin explizit ein – wenn auch begrenzter – Blick in die Zukunft geworfen.

Implizit enthalten ist dieser „Blick in die Zukunft“ in Aktionsauswahlverfahren, die Reinforcement Learning zur Bestimmung der besten Aktion verwenden [Sutton 1988]. Zwar ist die durch Reinforcement Learning gelernte Funktion nicht zwangsläufig identisch mit den anderen in diesem Abschnitt erwähnten Nutzenfunktionen, aber die Ähnlichkeit des Ansatzes ist doch so groß, dass eine Aufführung an diesem Ort gerechtfertigt erscheint. Die früheste – ohne den Einsatz von Computern funktionierende – solche Aktionsauswahl war Menace von Donald Michie [Michie 1961]. Menace lernte das Spielen von Tic-Tac-Toe. Auch im Roboterfußball wird Reinforcement Learning in verschiedenen Ligen zur Aktionsauswahl eingesetzt [Kleiner et al. 2002] [Riedmiller et al. 2001].

Ein explizit planender Algorithmus zur Aktionsauswahl ist das bei Brettspielen verwendete Minimax-Verfahren; dieses wird auch oft durch den Einsatz von Nutzenfunktionen an die Anforderungen der Realität angepasst. Da Minimax die Basis der vorgestellten Aktionsauswahl für das Tischfußballspiel darstellt, soll es im folgenden etwas tiefergehend beschrieben werden.

2.5 Minimax

Im Bereich der Brettspiele ist der (z.B. in [Shannon 1950] vorgestellte) Minimax-Algorithmus das klassische Verfahren, um den nächsten Zug in einer gegebenen Spielsituation zu bestimmen. Im folgenden sollen kurz die Grundlagen des allgemeinen Minimax-Algorithmus dargestellt werden, bevor im nächsten Kapitel die Modifikationen zur Anpassung von Minimax an das Tischfußballspiel vorgestellt werde.

2.5.1 Die Problemstellung

Der einfachste Fall, in dem Minimax eingesetzt wird, ist das Spiel zwischen zwei Gegnern, Min und Max, die abwechselnd aktiv sind, wobei Max beginnt. Eine Aktion von einem der beiden Spieler wird auch als ein „Halbzug" bezeichnet. Ferner sind für jeden Spielzustand die für den ziehenden Spieler legalen Züge sowie ihre Auswirkungen bekannt. Jedem möglichen Ausgang eines Spieles ist ferner ein Wert zugeordnet; im Falle von Spielen, die nur zwei Ausgänge, Sieg oder Niederlage, kennen, sind diese Werte typischerweise 1 und 0. Andere Spiele wie z.B. Skat kennen Siege und Niederlagen verschiedener Güteklassen, die im Allgemeinen durch irgendeine Art von Punktekonto bestimmt werden; in diesem Fall wird der Kontostand am Spielende als „Wert" des Spieles verwendet. Das Ziel von Max ist nun, den höchstmöglich bewerteten Spielausgang zu erreichen, während Min danach strebt, den Wert des Endzustandes möglichst klein zu halten.

2.5.2 Der einfache Minimax-Algorithmus

Um die oben beschriebene Situation mithilfe von Minimax lösen zu können, wird obige Ausführung als Beschreibung eines Suchproblems verstanden. Formal umfasst diese:

- Den Startzustand $gamestate \in \mathcal{S}$, entsprechend dem derzeitigen Spielzustand, wobei $\mathcal{S}$ die Menge legaler Spielzustände ist;
- Eine Testfunktion $finished(s), s \in \mathcal{S}$, die prüft, ob das Spiel beendet ist; diese ist durch die Spielregeln vorgegeben: So ist z.B. ein Skatspiel beendet, wenn keiner der Spieler mehr Karten auf der Hand hat, ein Mühlespiel, wenn einer der Gegner weniger als drei Steine besitzt.
- Eine Nutzenfunktion $utility(s), s \in \mathcal{S}$, die jedem Spielausgang einen Wert zuweist, und
- Einer Menge von Zügen, die jeweils einen Spielzustand $s_i \in \mathcal{S}$ in einen Nachfolgezustand $s_{i+1} \in \mathcal{S}$ überführen.

Das Vorgehen von MINIMAX besteht nun darin, den vollständigen Suchbaum, der in diesem Fall als Spielbaum bezeichnet wird, aufzuspannen, und daraufhin aus diesem Spielbaum die optimale Aktion für MAX zu bestimmen. Der Spielbaum enthält drei Arten von Knoten, wobei jeder Knoten n einen Spielzustand $state(n)$ repräsentiert:

Min-Knoten, bei denen MIN am Zuge ist; die Menge der MIN-Knoten eines Baumes wird durch $\mathcal{I}$ bezeichnet; ist MAX an der Reihe, so handelt es sich um einen

Max-Knoten, die in der Menge $\mathcal{A}$ zusammengefasst sind, sowie

Terminalknoten: Für die Menge $\mathcal{T}$ der Terminalknoten gilt: $finished(state(t)) = \top \forall t \in \mathcal{T}$

Der Wurzelknoten ist laut Definition ein MAX-Knoten. Bei einem Spiel, in dem die Gegner alternierend ziehen, besteht ein einem Spielablauf entsprechender Pfad von der Wurzel zu einem Blatt also aus einer Folge von Knoten, die mit einem MAX-Knoten beginnt, mit einem Terminalknoten endet und innerhalb derer sich MIN- und MAX-Knoten abwechseln.

Neben der Information über den jeweiligen Spielzustand besitzt jeder Knoten einen Wert $minimax$, der sich je nach Knotenart unterschiedlich berechnet:

- Der Wert eines Terminalknotens ist der Nutzen des entsprechenden Spielausganges, also

$$minimax(n) = utility(n), \forall n \in \mathcal{T} \tag{2.1}$$

- ein MIN-Knoten n wird mit dem Minimalwert seiner Nachfolgerknoten $Successors(n)$ bewertet

$$minimax(n) = \min\{utility(s) | s \in Successors(n)\} \forall n \in \mathcal{I} \tag{2.2}$$

- einem MAX-Knoten wird der Maximalwert seiner Nachfolgerknoten zugeordnet:

$$minimax(n) = \max\{utility(s) | s \in Successors(n)\} \forall n \in \mathcal{A} \tag{2.3}$$

Diese Bewertung der einzelnen Knoten entspringt der Einsicht, dass eine perfekte Aktion für MIN diejenige ist, die in einer bestimmten Situation den für MAX maximal erreichbaren Wert minimiert, während MAX versucht, diesen Wert zu maximieren.

Der Ablauf von MINIMAX hat also folgende Form:

1. Der Spielbaum wird aufgespannt
2. Jedem Terminalknoten wird ein durch die Nutzenfunktion gegebener Wert zugewiesen
3. Von den Terminalknoten ausgehend werden für alle inneren (=MIN- und MAX-) Knoten die Werte berechnet.
4. MAX entscheidet sich für die Aktion, die zu dem MIN-Knoten mit dem höchsten Wert führt.

Diese Schritte sind in den Abbildungen 2.1 bis 2.2 dargestellt.

Der MINIMAX-Algorithmus liefert die beste Entscheidung für MAX unter der Voraussetzung, dass MIN optimal spielt; ist dies nicht der Fall, so kann sich der Nutzen für MAX nur erhöhen.

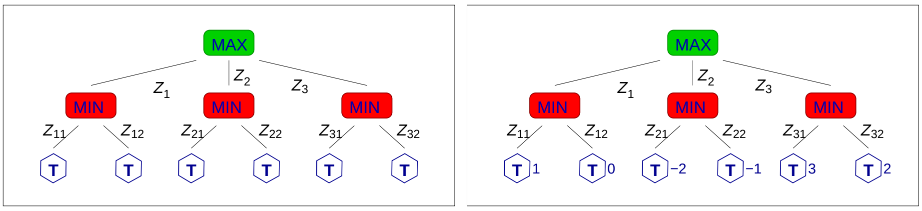


Abbildung 2.1: Aufspannung des Baumes und Bewertung der Terminalknoten. Die verschiedenen Zugmöglichkeiten sind durch die z_i bezeichnet.

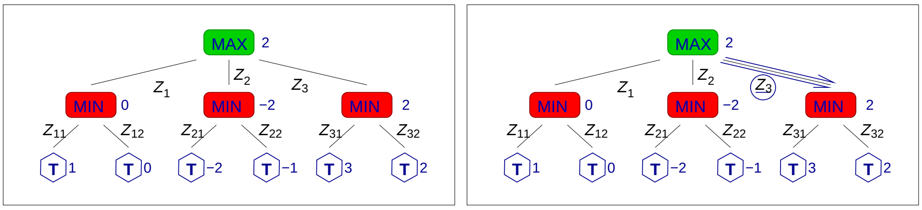


Abbildung 2.2: Bewertung der inneren Knoten und Auswahl der Aktion

Ein weiteres Charakteristikum von Minimax ist, dass die Nutzenfunktion den Terminalzuständen nicht die exakten Werte eines gewonnenen oder verlorenen Spieles zuweisen muss; wichtig ist lediglich, dass die richtige Reihenfolge unter den Werten für die einzelnen Endzustände gewährleistet ist, also ein für Max im Vergleich zu einem anderen besserer Spielausgang mit einem höheren Nutzen als dieser bewertet wird.

Theoretisch kann somit jedes Spiel, das den oben beschriebenen Bedingungen gehorcht, durch den Einsatz von Minimax perfekt durch den Computer gespielt werden. Allerdings wird der tatsächlichen Einsetzbarkeit des unmodifizierten Minimax-Algorithmus durch die Natur der meisten Spiele ein Riegel vorgeschoben.

2.5.3 Einsatz von Bewertungsfunktionen

Bei einem Verzweigungsfaktor v und einer Spiellänge von t Halbzügen liegt die Größe des Spielbaumes in $O(v^t)$. Daher ist es mit in absehbarer Zeit verfügbarer Hardware (und vermutlich weit darüber hinaus) nicht möglich, ein Spiel wie Schach, das einen durchschnittlichen Verzweigungsfaktor von ca. 35 und eine durchschnittliche Spiellänge von etwa 100 Halbzügen hat ([Russel und Norvig 1995]), durch Verwendung von Minimax vollständig durchzurechnen. Die Erfolge von Minimax-basierten Schachprogrammen in letzter Zeit zeigen jedoch, dass es trotzdem möglich ist, unter Verwendung des Minimax-Prinzips in derartigen Spielen zu bestehen.

Ein hierzu verwendetes Hilfsmittel sind die sogenannten Bewertungsfunktionen: Es sind dies Funktionen $eval(n)$, die für einen gegebenen Spielzustand n eine Schätzung des Nutzens dieses Zustandes vornehmen. Eine äußerst einfache Bewertungsfunktion für das Mühlespiel wäre z.B. die Anzahl der eigenen Steine auf dem Brett abgezogen der Anzahl gegnerischer Steine. Dieses Beispiel zeigt auch sofort die Problematik der Bewertungsfunktionen: Zwar scheint die Funktion auf den ersten Blick plausibel und weist am Ende des Spieles mit Sicherheit dem Sieger einen besseren Wert zu; jedoch existieren Situationen („Zwickmühlen“), in denen sie versagt. In diesen Situationen nimmt einer der Spieler in jedem Zug dem anderen einen Stein ab, die erwähnte Bewertungsfunktion, die also in einem Zustand s_i dem Spieler Min mit dem Wert -2 eine gute Spielstellung bescheinigt, kann in diesem Fall im Zustand s_{i+3} mit dem Wert 1 schon Vorteile für die Position von Max erkennen. Obwohl die aufgeführte Funktion besonders primitiver Natur ist, bleibt das Problem auch bei ausgeklügelteren Bewertungsfunktionen bestehen: Die Bewertungsfunktion ist eine Schätzung, und Schätzungen können danebenliegen. Eine perfekte Bewertungsfunktion hingegen macht Suche vollständig überflüssig, da Max sich nur für den Nachfolgezustand mit dem höchsten Wert der Bewertungsfunktion zu entscheiden braucht – beispielsweise verlassen sich Spielprogramme, die auf Reinforcement Learning basieren, auf die Qualität der gelernten Bewertungsfunktion.

Der Einsatz der Bewertungsfunktion geschieht nun folgendermassen: Der Suchbaum wird bis zu einer gewissen Tiefe, die durch eine Zeit- oder Suchtiefenbeschränkung gegeben sein kann, aufgespannt; die entstandenen Blätter werden mit der Bewertungsfunktion hinsichtlich ihres Nutzens abgeschätzt und im weiteren Verlauf von Minimax wie Terminalknoten behandelt. Allerdings führt der rein externen Kriterien folgende Abbruch der Baumkonstruktion mit nachfolgender Zustandsbewertung dann auf Probleme, wenn der Wert der Bewertungsfunktion sich in den Terminalknoten stark verändert; fortgeschrittene Methoden zur Begrenzung der Suchtiefe schneiden den Baum nur in ruhigen („quiescent“) Situationen ab.

Zur Verwendung im Zuge des MINIMAX-Algorithmus muss eine Bewertungsfunktion zwei Kriterien erfüllen: Sie sollte selbstverständlich eine möglichst gute Einschätzung des Spielgeschehens liefern, und sie sollte dies schnell tun: Dadurch, dass die Bewertungsfunktion in allen Blättern des Spielbaumes – und dadurch: oft – angewendet wird, schlagen Unterschiede in der Ausführungszeit der Bewertungsfunktion direkt auf die Leistungsfähigkeit des Algorithmus durch.

2.5.4 Baumbeschneidung

Neben dem heuristischen – und somit potentiell fehlerhaften – Abschneiden des Baumes existieren Möglichkeiten, die Leistungsfähigkeit von MINIMAX zu erhöhen, ohne dadurch Einbußen in der Spielstärke hinnehmen zu müssen.

Die *Alpha-Beta-Beschneidung* ist ein seit Anfang der sechziger Jahre bekanntes Verfahren zur Steigerung der Effizienz von MINIMAX. Das Vorgehen besteht darin, während dem Aufbau des Baumes den Wert der einzelnen Knoten zu beschränken, und aufgrund dieser Schranken zu erkennen, ob dieser Ast einen Einfluss auf den MINIMAXwert in der Wurzel haben kann oder nicht. Das Ergebnis eines unter Verwendung von Alpha-Beta-Beschneidung erzeugten Baumes ist identisch mit dem eines vollen MINIMAX-Baumes [KNUTH und MOORE 1975].

Einem grundsätzlich anderen Ansatz geht *SSS** nach: Während Alpha-Beta zum Aufbau des Baumes Tiefensuche verwendet, erfolgt dieser bei *SSS** nach „best-first"-Manier: Die „Best-First"-Vorgehensweise besteht darin, zu jedem Zeitpunkt denjenigen Knoten zu expandieren, der am vielversprechendsten zu sein scheint. *SSS** dominiert Alpha-Beta beweisbar, d.h. es kann gezeigt werden, dass *SSS** bei identischem Ergebnis auf jeden Fall weniger Knoten expandiert als Alpha-Beta [STOCKMAN 1979].

Der ursprünglich sehr komplizierte *SSS**-Algorithmus wurde mittlerweile als eine erweiterte Form der Alpha-Beta-Beschneidung erkannt und konnte stark vereinfacht werden [PLAAT et al. 1996].

2.5.5 Minimax mit Zufallsereignissen

Viele Spiele verwenden den Zufall als Spielelement; dies geschieht typischerweise durch das Würfeln und Mischen oder Ziehen von Karten. Neben reinen Glücksspielen lassen es die meisten derartigen Spiele trotzdem zu, dass Spieler richtige oder falsche Entscheidungen fällen - in diese Kategorie von Spielen gehören fast alle klassischen Brettspiele wie Risiko, Mensch-ärgere-dich-nicht und Monopoly. Wie kann man nun den MINIMAX-Algorithmus dahingehend verändern, dass er auch in solchen Spielen die Aktionen mit der größten Erfolgswahrscheinlichkeit auswählt? Die Antwort liegt in der Hinzunahme eines neuen Typs von Knoten, dem Würfelknoten $d \in \mathcal{D}$. Wann immer ein Zufallsereignis in den Lauf des Spieles eingreift, wird ein Würfelknoten erzeugt. Ein solcher Würfelknoten hat eine Anzahl von Nachfolgern, die der Anzahl $|\mathcal{R}|$ der möglichen verschiedenen Ausgänge des Zufallsereignisses entspricht, wobei jedem Nachfolger $succ(n, r)$ $n \in \mathcal{D}, r \in \mathcal{R}$ einer dieser Ausgänge $r \in \mathcal{R}$ sowie dessen Wahrscheinlichkeit $P(r)$ zugeordnet sind. Der MINIMAX-Wert eines Würfelknotens ist nun der Erwartungswert über die MINIMAX-Werte seiner Nachfolger, also

$$minimax(n) = \sum_{r_i \in \mathcal{R}} P(r_i) \cdot minimax(succ(n, r_i)) \forall n \in \mathcal{D} \qquad (2.4)$$

Der so modifizierte Algorithmus wird auch als EXPECTIMINIMAX bezeichnet.

Kapitel 3

KiRo

Bei *KiRo* handelt es sich um einen Tischfußballroboter, also um einen Tischfußballtisch, bei dem es in der vorliegenden Hardwareausführung möglich ist, gegen einen Computer Tischfußball zu spielen. Im folgenden soll der grundsätzliche Aufbau der Hard- und Software vorgestellt werden; die Informationen sind im wesentlichen [WEIGEL und NEBEL 2002] entnommen. Zu bemerken ist hierbei, dass die Bezeichnung *KiRo* sich sowohl auf das Gesamtsystem als auch auf das Steuerprogramm beziehen kann; soweit dies aus dem Kontext nicht eindeutig hervorgeht, wird es explizit unterschieden.

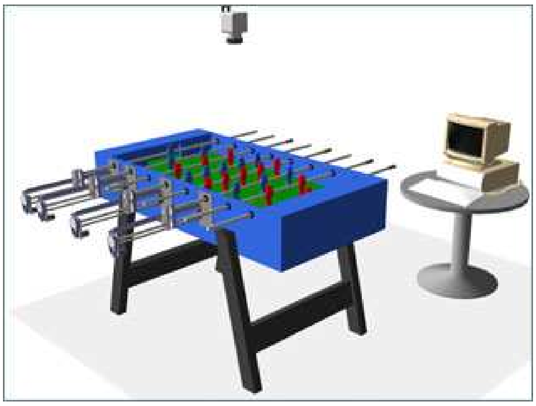


Abbildung 3.1: Das KiRo-System

3.1 Hardware

Die Entwicklung von *KiRo* fand unter dem Vorsatz statt, das gesamte System so einfach wie möglich zu halten. Diese Designentscheidung – die wohl maßgeblich dafür verantwortlich war, dass die Industrie mittlerweile die Eignung des Systems für die kommerzielle Produktion überprüft – spiegelt sich stark in der Hardwareausstattung wider:

Der Tischfußballtisch ist eine im Handel erhältliche Ausführung; es handelt sich hierbei um das Modell „ Home Soccer“ der Firma Löwen. Dieser Tisch wurde im wesentlichen unverändert in das System übernommen; die einzige Modifikation – neben der Anbringung von Bohrlöchern für weitere Komponenten – besteht darin, dass die normalerweise aus Stahl bestehenden Stangen, auf denen die Figuren befestigt sind, durch solche aus Aluminium ersetzt wurden, sofern es sich um vom Computer gesteuerte handelt. Dieser Schritt wurde unternommen, um die Leistungsaufnahme der verwendeten Elektromotoren so gering wie möglich zu halten; er ist problemlos, da die sehr viel robusteren Stahlstangen nur für menschliche Spieler benötigt werden: Die Ballkontrolle wird durch diese subjektiv besser und die vor allem in gastronomischen Betrieben nicht zu vermeidenden Fehlbelastungen wie z.B. das Lehnen auf die Stangen führen nicht zum sofortigen Bruch des Materials.

Zur Bewegung der Stangen sind außerhalb des Tisches Gestelle angebracht, in denen die Gleichstromelektromotoren platziert sind; diese werden via Motorcontroller des Typs *MCDC2805* der Firma *Faulhaber* von einem handelsüblichen PC mit einem mit 1.7 Gigahertz getakteten AMD-Prozessor gesteuert.

Dieser Rechner erhält die Daten über den momentanen Spielzustand durch eine oberhalb des Tisches montierte Kamera des Typs *Phytec VCAM-110*, durch die 50 Bilder pro Sekunde mit einer Auflösung von 324x288 Bildpunkten aufgenommen werden können.

Weiterhin ist auf dem Rand des Tisches aus Sicherheitsgründen ein Lichtgitter angebracht; wird dieses unterbrochen, so werden die Motoren blockiert.

3.2 Software

Das zur Steuerung von *KiRo* eingesetzte Programm verläuft in Zyklen, deren Länge durch die Bildwiederholfrequenz der Kamera vorgegeben sind: Die für einen Zyklus zur Verfügung stehende Zeit beträgt dementsprechend 20 ms.

Ein solcher Zyklus verläuft, wie in Abbildung 3.2 dargestellt, in vier Schritten: Zunächst wird ein von der Kamera geliefertes Bild interpretiert. Daraufhin werden die hieraus gewonnenen Informationen zusammen mit den aus anderen Quellen – wie z.B. dem Wissen über vorhergegangene Weltzustände – gewonnenen zum Weltmodell zusammengefügt. Auf dieses Weltmodell reagiert die Aktionsauswahl mit der Entscheidung für eine bestimmte Verhaltensweise, die dann durch die Aktionsausführung realisiert wird.

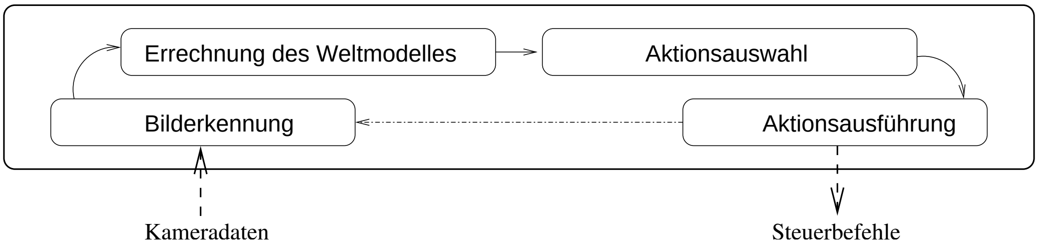


Abbildung 3.2: Ablauf eines Zyklus

3.2.1 Bilderkennung

Die Bilderkennung verläuft auf einfache, jedoch sehr schnelle und robuste Weise: Das Grundprinzip ist hierbei immer die Unterscheidung der einzelnen Merkmale durch die Farbgebung. Insgesamt kommen fünf Farben vor: Rot und Blau sind die Spieler der beiden Mannschaften, das Spielfeld ist grün, die darauf eingezeichneten Feldlinien sind weiß und der Ball ist gelb.

Zur Kalibrierung werden die Feldlinien gesucht: Der Durchmesser des Mittelkreises gibt Aufschluss darüber, wie groß das Spielfeld auf dem Bild erscheint, der Winkel der Mittellinie lässt die Rotation erkennen.

Zur Bestimmung der Positionen der Spielfiguren werden aus den auf dem Bild zu erkennenden Flecken der entsprechenden Farbe, vereint mit dem Wissen über den Abstand der Figuren zueinander, die Position und Rotation der Figuren geschätzt; während dies bei der Erkennung der Verschiebung sehr gut funktioniert, ist die Schätzung des Winkels der Figuren durch reine Bilderkennung sehr schwierig und fehlerbehaftet; klar erkannt werden kann aber, ob die Füße einer Spielfigur momentan nach unten oder seitwärts gerichtet sind, was für viele Zwecke ausreicht.

Die wichtigste Aufgabe der Bildverarbeitung aber ist die Erkennung des Balls, falls dieser im Spiel ist. Das Hauptproblem hierbei ist, dass dieser häufig ganz oder teilweise durch die Spielfiguren oder Stangen überdeckt ist; ist keines von beidem der Fall, so wird der Mittelpunkt des gelben Farbflecks als Ballposition angegeben. Falls der Ball nur teilweise überdeckt ist, so wird versucht, durch Wissen über die Größe des Balls und Positionen der Spielfiguren seine Lage zu errechnen. Trotz dieser Maßnahmen aber sind die bei verdecktem Ball bestimmten Werte von einer hohen Unzuverlässigkeit, die sich bei der derzeitigen Hardwareausstattung wohl auch nicht vermeiden lässt.

3.2.2 Weltmodellerstellung

Die durch die Bilderkennung gelieferten Informationen über den Ort der einzelnen Objekte werden im Verlauf der Weltmodellerstellung durch die Einbeziehung der Daten früherer Weltmodelle mit Angaben über Richtung und Geschwindigkeit des Balls und der Figuren angereichert. Im Falle des Balls geschieht dies derzeit durch den Einsatz eines Kalman-Filters.

Eine weitere Aufgabe der Weltmodellerstellung ist es, den Aufenthaltsort des Balls anzugeben, auch wenn dieser vollständig überdeckt ist. Zu diesem Zweck wird in früheren Weltmodellen eine Figur als „ballführend“ gekennzeichnet, wenn der Ball nah genug an ihr erkannt wird; falls der Ball in späteren Aufnahmen überhaupt nicht mehr zu erkennen ist, so wird er als direkt unterhalb dieser Figur befindlich angenommen.

Die Aufnahme und Analyse des Kamerabildes beanspruchen einen gewissen Zeitraum; aus diesem Grund wird die Entwicklung des erstellten Weltmodells vor der Weiterleitung an die Aktionsauswahl noch um diesen Zeitraum vorhergesagt.

Abbildung 3.3 zeigt eine Visualisierung des von *KiRo* verwendeten Weltmodells.

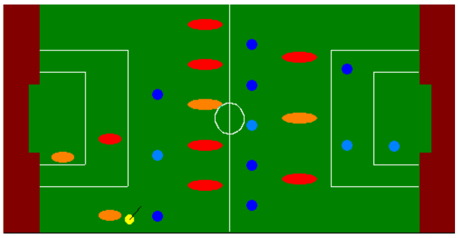


Abbildung 3.3: Das *KiRo* zur Verfügung stehende Weltmodell

3.2.3 Aktionsauswahl

Die von *KiRo* verwendete Aktionsauswahl ist in ihrer ursprünglichen Form rein reaktiv – es handelt sich um in einem Entscheidungsbaum zusammengefasste hierarchisierte Regeln. Zudem bezieht sie die gegnerischen Positionen in keiner Weise ein. Eine Aktion bezeichnet hierbei eine Verhaltensweise, die einer der vom Rechner gesteuerten Stangen zugewiesen wird.

Hauptkriterium für die Auswahl einer Aktion ist die Position des Balls zu der entsprechenden Stange: Allgemein gilt (mit der Ausnahme des Schusses), dass die Füße der Figuren nach unten deuten, wenn der Ball sich vor der entsprechenden Stange befinden; ist der momentane Ort des Balls hingegen hinter der Stange, so sind die Füße, um ein Passieren des Balls zu ermöglichen, hochgezogen.

Die Positionierung der Figuren zum Ball hängt von der Entfernung desselben ab: Ist er weit von der Stange entfernt, so wird diese auf eine strategisch als geeignet erscheinende Position bewegt, ansonsten wird versucht, eine zum Abfangen des Balls geeignete Position einzunehmen. Die Geschwindigkeit, mit der diese Positionierung abläuft, ist neben der Entfernung auch von der Geschwindigkeit des Balls abhängig. Ist sie erfolgreich, gelingt es also, die Stange so zum Ball zu positionieren, dass er von einer der Figuren zu erreichen ist, so wird ein Schuss nach vorne ausgelöst.

KiRo bewegt sich unter dieser Aktionsauswahl schnell und schießt, obwohl ungezielt, so häufig, dass er oft Tore erzielen kann und menschlichen Gegenspielern oft gleichwertig oder überlegen ist [Weigel und Nebel 2002].

3.2.4 Aktionsausführung

In der Aktionsausführung werden die von der Aktionsauswahl als Befehle der Art „Stürmerstange: Schuss" übermittelten Entscheidungen in Befehle für die Motoren übersetzt.

3.3 Simulator

Da die derzeitige Hardwareausstattung es nicht zulässt, verschiedene Programme gegeneinander antreten zu lassen, wurde ein Simulator entwickelt. Um diesen so realistisch wie möglich zu gestalten, wurde auf die ODE-Bibliothek[1] zurückgegriffen. Es handelt sich hierbei um eine frei verfügbare Softwarebibliothek, durch die es möglich ist, unter Angabe von physikalischen Eigenschaften einzelner Gegenstände wie z.B. deren Gewicht, Größe, Gestalt oder Geschwindigkeit deren Verhalten in der realen Welt nachzubilden. Weiterhin werden die errechneten Daten verrauscht, um die Ungenauigkeit der am realen Tisch verwendeten Sensoren nachzubilden. Es muss aber hinzugefügt werden, dass dieses injizierte Rauschen homogen über das ganze Spielfeld verteilt ist, während in der Realität die Schwierigkeiten der Ballerkennung vor allem in Figurennähe groß sind.

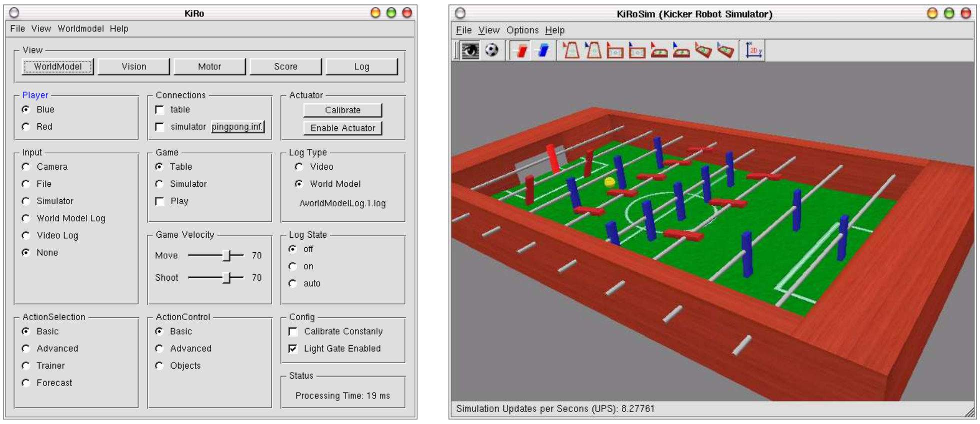
Abbildung 3.4(b) zeigt einen „Screenshot" des Simulators.

Die Verbindung zwischen dem Simulator und dem *KiRo*-Programm wird über ein Rechnernetz hergestellt; es ist also möglich, mehrere Programme auf unterschiedlichen Rechnern über das Netzwerk gegeneinander antreten zu lassen.

[1] http://q12.org/ode/

3.4 Bedienung

Sowohl das *KiRo*-Programm als auch der zugehörige Simulator sind über eine graphische Benutzerschnittstelle intuitiv zu bedienen. Abbildung 3.4 zeigt für beide Programme den Hauptbildschirm.



(a) Die Benutzerschnittstelle des *KiRo*-Programms

(b) Der Simulator

Abbildung 3.4: Zwei Screenshots

3.5 Anmerkungen

Das *KiRo*-System ist Gegenstand ständiger Weiterentwicklung; zu denen in jüngster Vergangenheit erfolgten Modifikationen gehören, abgesehen von ständigen Verbesserungen und Berichtigungen von Details, die Einführung und laufende Verfeinerung des Simulators sowie die Veränderung der Ballerkennung. Durch das erwachende Engagement der Privatwirtschaft sind in naher Zukunft tiefgreifende Modifikationen von Hard- und Software zu erwarten: Die derzeitigen Probleme bei der Ballerkennung beispielsweise sollten durch die Detektion des Balls mittels einer unter der Spielfläche angebrachten Infrarotkamera nebst Infrarot-Leuchtdioden in naher Zukunft der Vergangenheit angehören.

Kapitel 4

Minimax im Tischfußballspiel

4.1 Warum Minimax?

Wie bereits eingangs erwähnt basiert das in dieser Arbeit vorgestellte Verfahren auf MINIMAX. Es stellt sich aber die Frage, was den Einsatz von MINIMAX, das ja für eine Umgebung, in der „alle relevanten Details bekannt“ sind [RUSSEL und NORVIG 1995], das Brettspiel, entworfen und geeignet ist, in der Robotik, also einer Umgebung, in der dies typischerweise nicht gegeben ist, rechtfertigt.

Auf den ersten Blick scheinen die Unterschiede zwischen einem normalen, MINIMAX-tauglichen Brettspiel und dem Tischfußballspiel derart groß zu sein, dass die Idee, MINIMAX am Tischfußballtisch anzuwenden, unrealistisch erscheint. Die am stärksten ins Auge fallenden Unterschiede sind:

- Vom **Weltmodell**, bei normalen Brettspielen vollkommen zugänglich und exakt bekannt, existiert im Falle des Tischfußballspieles aufgrund der unvermeidlichen Messfehler der eingesetzten Sensoren nur eine Annäherung. Insbesondere ist die Schätzung der Bewegung des Balls fehlerbehaftet. Von der Kenntnis der Bewegung aber hängt jeder nichttriviale Versuch, erfolgreich Tischfußball zu spielen, entscheidend ab.
- Das Weltmodell ist beim Tischfußball im Gegensatz zu Brettspielen

stetig und ändert sich im Falle des bewegenden Balls auch ohne Zutun der einzelnen Spieler.

- Für **Aktionen** der einzelnen Spieler gilt aufgrund des verrauschten Weltmodells und der notwendigerweise nicht hundertprozentig genauen Ausführung der Steuerbefehle in noch stärkerem Masse, dass ihre Auswirkungen nicht exakt bekannt sind; im Gegensatz zur Grundannahme von EXPECTIMINIMAX ist die Menge der möglichen verschiedenen Konsequenzen einer Aktion nicht durch eine feste Anzahl von Hypothesen, sondern durch eine Wahrscheinlichkeitsverteilung gegeben.

- Die **Spieler** können zu jedem Zeitpunkt, insbesondere gleichzeitig, das Weltmodell verändern, indem sie ihre Stangen bewegen; beim klassischen MINIMAX greifen die Spieler in einer vorgegebenen Reihenfolge in das Spielgeschehen ein.

- Der Versuch, einen Spielbaum aufzubauen, der in jedem Knoten alle möglichen Aktionen der einzelnen Spieler evaluiert, ist von vorneherein zum Scheitern verurteilt: Die Anzahl möglicher Eingriffe in den Weltzustand ist unbegrenzt.

Diese Punkte legen die Vermutung nahe, dass es nicht möglich ist, den Verlauf des Spieles in einen MINIMAXartigen Spielbaum zu fassen. Für den Einsatz einer modifizierten MINIMAX-Version, die es verdient, das Wort MINIMAX im Namen zu tragen, spricht aber, dass das Tischfußballspiel einige Charakteristika von Brettspielen beinhaltet: Es gibt einen Gegner, dessen Motivation mit der von MIN identisch ist, es gibt gute und schlechte Spielzustände, und diese lassen sich relativ einfach erkennen.

Trotz dieser Parallelen scheint dennoch intuitiv eine Reinforcement-Learning-basierte Aktionsauswahl vorteilhaft zu sein, da in dieser sämtliche Aspekte des Spieles inklusive der Unschärfe der Sensorinformationen „umsonst“ in der gelernten Funktion enthalten sind. Gegen Reinforcement Learning sprechen allerdings Argumente: Der Zustandsraum ist ungeheuer groß, so dass nur das Lernen in Partien gegen sich selbst, also „offline “, möglich erscheint. Tischfußball aber ist ein Spiel, dessen Charakter immer stark von der Spielweise des Gegners abhängt. Die durch „offline“-Lernvorgänge gewonnene Funktion zur Aktionsauswahl aber wäre zu einer Adaption an den Gegner nicht möglich. Bei einem explizit planenden Verfahren hingegen besteht die Hoffnung, durch adaptive Gegnermodelle „online“ eine Anpassung an den Charakter des Spieles zu erreichen.

Im folgenden werden Modifikationen vorgestellt, die trotz der erwähnten Schwierigkeiten den Einsatz von MINIMAX zum Tischfußballspielen ermöglichen.

4.2 Die Vorwärtssimulation

Im Falle von Brettspielen werden zu jedem Knoten alle möglichen Nachfolgezustände erzeugt; bei den MIN- und MAX-Knoten ist deren Zahl gegeben durch die Anzahl legaler Züge, bei den Würfelknoten durch die der möglichen Ergebnisse eines Zufallsereignisses. Weiterhin sind die entstehenden Weltzustände mit denen identisch, die durch die entsprechenden Züge oder Würfelergebnisse aus den als Eingabe verwendeten Spielpositionen erstellt würden.

Beides muss im Falle des Tischfußballspieles geopfert werden: Weder kann der Anspruch erhoben werden, alle Zugmöglichkeiten des Gegners in Betracht zu ziehen, noch ist davon auszugehen, dass die Folgen der einzelnen Aktionen exakt abgebildet werden können. Dennoch soll dargelegt werden, dass es auch unter diesen Bedingungen möglich ist, Spielbäume zu erzeugen, die zur Aktionsauswahl geeignet sind. Die Werkzeuge hierzu sind die Gegnermodellierung und die Vorwärtssimulation.

Wie weiter oben bereits erwähnt, ist ein durch die Interpretation von Sensorwerten in Echtzeit erstelltes Weltmodell in jedem Fall ungenau, wobei gilt, dass die Schätzungen für Positionen von höherer Zuverlässigkeit sind als die von Geschwindigkeiten. Eine Simulation wiederum kann nicht den Anspruch haben, die Entwicklungen in einer realen Umgebung präzise widerzugeben. Aus diesen zwei Gründen scheint schon alleine der Ansatz, die Veränderungen im Verlaufe eines Tischfußballspieles durch eine auf fehlerhaften Startwerten aufbauende unrealistische Simulation zu prognostizieren von vorneherein zu akkumulierenden Fehlern, einer starken Divergenz zum realen Spiel und somit zum völligen Verlust der Aussagekraft verdammt zu sein. Diesen Einwand glauben wir aus zwei Gründen entkräften zu können:

- Die Simulation greift nur sehr kurz in die Zukunft vor, so dass die tatsächlich nicht zu vermeidenden sich gegenseitig verstärkenden Fehler im Verlauf der Simulation im Griff gehalten werden können.

- Ein Kernpunkt der Simulation, die Abschätzung der Konsequenzen der eigenen Aktionen, kann durch den Einsatz von Verfahren des maschinellen Lernens stark an die Realität angenähert werden.

- Weiterhin ist absolute Genauigkeit bei der Vorwärtssimulation nicht unbedingt erforderlich: Die Frage, auf welchen x- und y-Koordinaten sich der Ball zu einem bestimmten Zeitpunkt befindet ist weniger wichtig als die Frage, ob der Ball von einem Spieler erreicht wird, wenn ja, von welchem, oder sich nach einer gewissen Zeitspanne im Tor befindet. Im Normalfall werden diese Fragen auch bei einer Abweichung von mehreren Zentimetern noch korrekt beantwortbar sein; der Ball trifft dann zwar um einige Zentimeter versetzt das Tor oder wird ein wenig verschoben abgefangen, die Kernfrage – Tor oder kein Tor – wird trotzdem identisch beantwortet.

Doch selbst unter der Annahme, dass es möglich ist, den Spielverlauf über einen ausreichenden Zeitraum vorhersehen zu können, bleiben einige Probleme ungelöst: Insbesondere gehört hierzu die Unmöglichkeit, vorherzusagen, wie der Gegner sich verhält, falls er im Ballbesitz ist. Dieses Problem lässt sich nicht lösen; allerdings lässt es sich ignorieren: Da alle Vorausplanung vergeblich ist, muss sich der reagierende Spieler darauf beschränken, so schnell wie möglich durch eine möglichst günstige Aufstellung der eigenen Figuren in den Ballbesitz zu gelangen, und dies am besten, ohne in der Zwischenzeit ein Tor zu kassieren. Diese Aufstellung aber ist ohne Vorwärtssimulation reaktiv bestimmbar. Die Konsequenz aus diesen Überlegungen ist diese: Falls der Gegner im Verlauf der Spielbaumkonstruktion in Ballbesitz gelangt, so wird die Vorwärtssimulation an dieser Stelle mit einem Aufruf der Evaluierungsfunktion abgebrochen; ist dies sogar im Wurzelknoten, also im tatsächlichen Spielzustand, der Fall, so wird auf die Konstruktion des Spielbaumes völlig verzichtet; die Aktionsauswahl und -ausführung ist in diesem Falle also rein reaktiv. Anders formuliert: Bei dem im Spielbaum formalisierten Spiel handelt es sich nicht um ein vollwertiges Tischfußballspiel: Die Aufgabe von Max ist es, den Ball so nahe wie möglich an das gegnerische Tor zu befördern; sein Gegner muss hingegen nur in den Ballbesitz gelangen.

Zur Verwendung von Spielbäumen nach diesen Grundsätzen müssen am Minimax-Algorithmus tiefgreifende Modifikationen vorgenommen werden. Diese sollen im folgenden erläutert werden.

4.3 Anpassungen von Minimax an das Tischfußballspiel

Die zur Anwendung auf das Tischfußballspiel notwendigen Anpassungen des Minimax-Algorithmus zur **Mi**nimaxbasierten **T**ischfussball**r**oboter**a**ktionsauswahl MITRA umfassen:

- Die Einführung eines neuen Knotentyps
- Ein im Gegensatz zu Minimax grundlegend unterschiedliches Gegnermodell
- Die Veränderung der Struktur des Baumes

4.3.1 Simulationsknoten

Um die Parallelität der Eingriffe der Spieler sowie die ständige Veränderung des Weltzustandes zu modellieren wird ein neuer Knotentyp eingeführt: der Simulationsknoten. Alle Veränderungen des Weltzustandes laufen in ihm ab; er erfüllt also zwei Aufgaben:

- Unter Verwendung eines Modells der in der wirklichen Welt geltenden physikalischen Gesetze berechnet er aus den Informationen über den derzeitigen Weltzustand den zukünftigen Lauf des Balls.
- Er materialisiert durch die Interpretation der von beiden Spielern gegebenen Steuerbefehle die Eingriffe dieser in das Spielgeschehen.

Letzterer Punkt beinhaltet insbesondere, dass die von Min bzw. Max getroffenen Entscheidungen – ihre Züge – den Weltzustand nicht direkt beeinflussen, sondern erst im durch den Simulationsknoten dargestellten Verlauf der Zeit wirksam werden.

4.3.2 Gegnermodellierung

Weiter oben wurde bemerkt, dass es aussichtslos ist, die Aktionen des ballführenden Gegners in den Spielbaum aufnehmen zu wollen; seine Absichten bleiben völlig opak. Dies muss aber auch andersherum gelten: Max kennt seine eigenen Möglichkeiten, Min kann diese höchstens abschätzen. Aus diesem Grunde *darf der modellierte Gegner sich nicht aufgrund von Informationen aus tieferen Ebenen des Baumes für eine bestimmte Aktion entscheiden.* Die Notwendigkeit dieser Feststellung lässt sich mit einem einfachen Beispiel demonstrieren:

Angenommen, der Spielbaum wäre zwar nach obigen Kriterien konstruiert, und die Bewertungsfunktion der Min-Knoten entspräche der in Gleichung 2.2 dargestellten. Weiterhin wollen wir annehmen, dass die „körperlichen“ Fähigkeiten des Gegners ausreichen, um einen von Max abgeschossenen Ball abzufangen, falls die Figuren zu der richtigen Abfangposition gezogen werden. Unter diesen Umständen gilt: Min kann zu jedem Zeitpunkt – in jedem Min-Knoten – seine Figuren der entsprechenden Stange zu jedem möglichen Punkt ziehen; also insbesondere auch zu dem Punkt, an dem der von Max geschossene Ball diese Stange passiert, egal, wohin Max schießt. Der optimale Zug für Max wäre unter diesen Umständen also, den Ball zu halten, denn Max kann nicht darauf hoffen, durch irgendeine Aktion die Bewertung des Spielzustandes zu erhöhen.

Aus diesem Dilemma bieten sich mehrere Auswege an: Eine Möglichkeit ist es, die Gegnermodellierung dergestalt zu implementieren, dass Min sich auf wenige „rationale“ Züge beschränkt; der Nachteil dieser Variante ist aber, dass selbst wenige Optionen ausreichen, um alle Schussmöglichkeiten von Max zuverlässig abzudecken. Diesen Nachteil vermeidet eine andere, radikalere Variante: Die Züge des Gegners werden durch Würfelknoten modelliert. Die Gegnermodellierung und ihre Auswertung bestehen also nicht darin, alle möglichen Züge von Min zu erzeugen und sie unter der Annahme, Min spiele optimal, zu bewerten; stattdessen liefert das Gegnermodell Hypothesen darüber, wie dieser auf eine gegebene Spielsituation reagieren könnte, wobei die Unsicherheit darüber, für welche der Optionen sich der Gegner tatsächlich entscheidet, durch Wahrscheinlichkeitswerte ausgedrückt wird. Der so modellierte Gegner kann also immer noch jede beliebige Position auf dem Spielfeld einnehmen; je mehr von dieser Möglichkeit aber Gebrauch gemacht wird, je unentschlossener die Gegnermodellierung also ist, desto geringer wird das Gewicht der einzelnen Hypothesen. Da der Gegner sich also nicht mehr für den Zug entscheiden darf, der den Spielwert für Max minimiert, wird er im folgenden nicht mehr mit Min, sondern mit Opp (für „*opp*onierender Spieler“) bezeichnet.

Wieder lassen sich die Konsequenzen dieser Maßnahme am besten mit einem Beispiel illustrieren; wieder sei der Gegner dergestalt modelliert, dass er gleichverteilt jede Position auf dem Spielfeld einnimmt. Dieser so modellierte Gegner entspricht also einem Spieler, der seine Figuren auf nicht vorhersehbare, vom Spielzustand unabhängige Weise mit maximaler Geschwindigkeit bewegt. Unter diesen Umständen wäre die Wahrscheinlichkeit, dass Opp mit seiner Stürmerstange einen vom eigenen Verteidiger geschossenen Ball abfängt, deutlich geringer, als in dem Falle, in dem die Aufgabe darin besteht, mit der figurenreichen Mittelfeldstange einen Ball zu erreichen. In ersterem Fall ist die Wahrscheinlichkeit, dass ein Spieler genau auf der Balltrajektorie platziert wird, deutlich geringer, als im zweiten Fall, in dem dieser Wert bedingt durch die höhere Figurendichte und die geringere Amplitude deutlich höher liegt. Dies deckt sich mit den Beobachtungen vom realen Tisch.

Weiterhin scheint diese Art der Gegnermodellierung geeigneter, um datenbasierte Gegnermodelle einsetzen zu können: Prinzipiell für die Gegnermodellierung geeignete Ansätze wie [Bennewitz et al. 2002] liefern zusätzlich zu ihren Abschätzungen, welche Intention eine beobachtete Person verfolgt, auch die Wahrscheinlichkeit, dass dem so ist.

4.3.3 Veränderung der Baumstruktur

Wie soeben beschrieben, ändert sich das Weltmodell nur innerhalb der Simulationsknoten – dies ist gleichbedeutend damit, dass in der durch den Spielbaum gegebenen Vorwärtssimulation außerhalb der Simulationsknoten die Zeit stillsteht. Also ist es unerheblich, in welcher Reihenfolge Opp und Max ziehen, wenn folgende Bedingungen erfüllt werden:

1. Auf dem Pfad zwischen den entsprechenden Opp- und Max-Knoten liegt kein Simulationsknoten.
2. Die Entscheidungen, die innerhalb der Knoten gefällt werden, beruhen ausschließlich auf dem gegebenen Weltzustand.

Auf diese Art und Weise lassen sich also die in Wirklichkeit parallelen „Züge" der Spieler äquivalent seriell realisieren. Dies führt zu der in Abbildung 4.1 dargestellten Baumstruktur, in der ein „Zug" durch drei Ebenen von Knoten repräsentiert wird: In einer ersten Ebene von Max-Knoten reagiert Max auf den Weltzustand; hierauf folgt eine Schicht von Knoten, in denen Opp zieht. Abgeschlossen wird der „Zug" durch die Materialisierung der beiden „Halbzüge" in einer Ebene von Simulationsknoten.

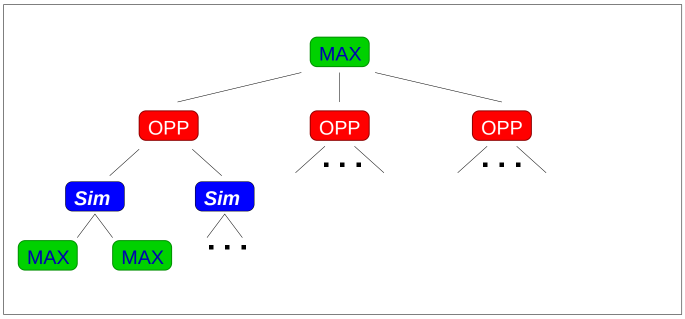


Abbildung 4.1: Die Baumstruktur für das Tischfußballspiel

Eine Kernfrage zu diesem Baumaufbau ist die, wie lange die Zeit in den Simulationsknoten vorweggenommen werden soll; in der Realität haben die Spieler ja die Möglichkeit, zu jedem Zeitpunkt ihre Stangen zu bewegen;

ein Zyklus des gesamten *KiRo*-Programms inklusive Aufnahme und Interpretierung eines Bildes usw. dauert 20 ms; ein solcher Zeitraum scheint also angemessen zu sein, um eine möglichst große Realitätstreue der Simulation zu erreichen. Gegen ein derartig kurzes Intervall spricht allerdings, dass die gesamte Konstruktion des Spielbaumes zwischen zwei „Zügen" von *KiRo*, also innerhalb des Anteils dieser 20 ms, die noch nicht von anderen Komponenten des Programms beansprucht wird, stattfinden muss; selbst bei einem moderaten Verzweigungsfaktor des mit einer solchen feinkörnigen zeitlichen Auflösung erstellten Baumes ist es unrealistisch, auch nur annähernd eine Sekunde vorauszuplanen. Die vorhergesagten Intervalle müssen also deutlich verlängert werden, wobei die Eingriffsmöglichkeiten der Spieler möglichst unbeschadet zu bleiben haben. Im Kapitel 7.6 werden Maßnahmen beschrieben, durch welche die Simulationsschrittdauer um mehrere Größenordnungen erhöht werden kann, ohne dass die Aktionsmöglichkeiten der Spieler zu stark beschnitten werden.

Kapitel 5

Das verwendete Weltmodell

Sowohl die Vorwärtssimulation als auch die Aktionen alleine bauen auf einem einheitlichen Format des Weltmodells auf. Jedes dieser Modelle $s \in \mathcal{S}$ beinhaltet zwei Arten von Informationen: Die Koordinaten der auf dem Spielfeld vorhandenen Objekte sowie allgemeinere Angaben über den Spielzustand und über das Weltmodell selbst.

Letztere umfassen:

- `echt` $\in \{\top, \bot\}$: Nimmt den Wert $\top$ genau dann an, wenn das Weltmodell aus dem derzeitigen tatsächlichen Spielzustand extrahiert wurde. Sobald das Weltmodell verändert oder auch nur kopiert wird, wird es als Kopie gebrandmarkt, indem `echt` auf $\bot$ gesetzt wird. Durch diese Variable wird die ansonsten konsequent durchgehaltenen Transparenz, ob es sich bei einem gegebenen Spielzustand um den momentan tatsächlich vorliegenden oder einen im Verlauf der Vorwärtssimulation erzeugten handelt, verletzt; der Grund für die Einführung dieses Wertes ist, dass die verschiedenen zur Steuerung der Aktionsauswahl eingesetzten Lernverfahren durch die Berücksichtigung modifizierter Weltmodelle nicht mehr auf die korrekten Werte konvergieren könnten.

  Außerdem muss während der Aktualisierung der Aktionen auf verschiedene Werte von `echt` reagiert werden: Während sich beim Spiel am realen Tischfußballtisch das Misstrauen in die von den Sensoren gelieferten Werte dadurch ausdrückt, dass die Bewegungen der Stangen geglättet werden, würde diese Glättung bei den großen Vorhersageintervallen während der Vorwärtssimulation dazu führen, dass der Ball von den Stangen nicht mehr erreicht wird. Diese Glättung wird dementsprechend genau dann aktiviert, wenn `echt` den Wert $\top$ annimmt.

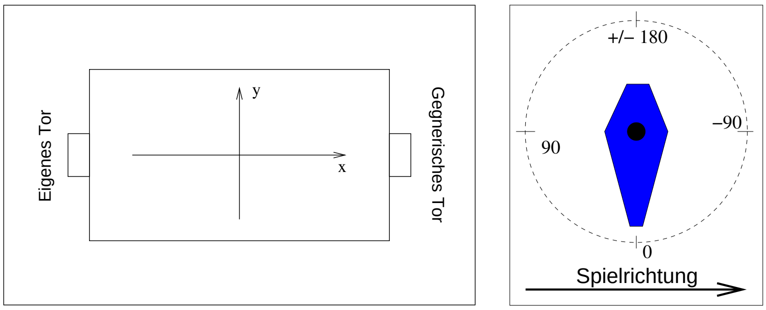
- `zustand` $\in \{ungültig, geschätzt, beobachtet, tor, gegentor\}$: Gibt eine grobe Information über den derzeitigen Spielzustand; während die ersten drei Werte ihre Verwendung ausschließlich bei der Erstellung des Weltmodells finden, wird auf die letzten beiden durch die Bewertungsfunktion zurückgegriffen.

- `zeitpunkt` $\in (\mathbb{N}, \mathbb{N})$. Kennzeichnet den Zeitpunkt der Erstellung des vorliegenden Weltmodells. Die Darstellung in zwei Zahlen hat ihren Ursprung im vom Computer verwendeten Zeitformat.

Neben diesen Angaben allgemeinerer Natur sind im Weltmodell alle Daten vorhanden, die notwendig sind, um Position, Bewegungsrichtung und Geschwindigkeit jeder Figur und des Balls zu rekonstruieren.

Das zugrundeliegende Koordinatensystem für alle Werte mit Ausnahme der Rotation der Stangen hat seinen Ursprung auf dem Mittelpunkt des Spielfeldes, die positive x-Achse geht entlang der Längsachse des Spielfeldes in Richtung des gegnerischen Tors und die positive y-Achse verläuft orthogonal zur x-Achse auf den gegnerischen Spieler zu. Die Rotation der Stangen wird dergestalt gemessen, dass der Wert einer Figur, deren Füße senkrecht nach unten zeigen, 0° und der einer waagrecht mit den Füßen in Spielrichtung des Roboters zeigenden -90° beträgt. Diese Koordinatensysteme sind in Abbildung 5.1 nochmals dargestellt.

Intern sind, da die Werte für die einzelnen Figuren einer Stange sich entweder gar nicht (z.B. der Winkel) oder aber auf vom Spielgeschehen unabhängige Weise (die y-Koordinate) unterscheiden, nur die Koordinaten für die Figur mit dem jeweils niedrigsten Index einer Stange gespeichert; in der nun folgenden Auflistung wird allerdings nicht die interne Verwirklichung, sondern die externe Schnittstelle beschrieben; diese beinhaltet Zugriffsroutinen für einzelne Figuren. Dies ist auch der Grund, aus dem die – für jede einzelne Stange unveränderlichen – x-Koordinaten der einzelnen Stangen hier ange-



(a) (b)

Abbildung 5.1: Die dem Weltmodell zugrundeliegenden Koordinatensysteme für (a) Orte auf dem Spielfeld und (b) die Rotation der Stangen

geben werden; tatsächlich werden diese natürlich nicht in jedem Weltmodell von neuem gespeichert.

Um auf die einzelnen Figuren oder Stangen zugreifen zu können müssen diese eindeutig identifizierbar sein. Dies geschieht im Falle einer Figur durch die Angabe des Spielers, der die entsprechende Figur kontrolliert, der Stange, auf welcher sich die Figur befindet, sowie des Index der Figur auf der entsprechenden Stange. Für die Unterscheidung der Stangen reichen die ersten beiden Spezifikationen. Die Menge $\mathcal{G}$ der Mitspieler umfasst die Elemente *kiro* und *gegner*, Stange und Index auf dieser sind in den Mengen $\mathcal{T} = \{torwart, verteidiger, mittelfeld, stürmer\}$ sowie $\mathcal{I} = \{0, 1, 2, 3, 4\}$ enthalten.

- `ballX, ballY` $\in \mathbb{R}$: Geben den Ort des Balls in Millimetern an.
- `ballVX, ballVY` $\in \mathbb{R}$: Geschwindigkeit und Richtung des Balls werden in die jeweiligen Geschwindigkeiten in x- und y- Richtung parametrisiert; dieser Darstellung wurde der Vorzug vor der äquivalenten Darstellung durch explizite Angabe von Richtung und Geschwindigkeit der Vorzug gegeben, da im Verlauf der Vorwärtssimulation praktisch ausschließlich auf die Geschwindigkeiten entlang der Koordinatenachsen zugegriffen wird. Alle Geschwindigkeiten werden in Millimetern pro Sekunde angegeben.

- `mannY[g][t][i]` $\in \mathbb{R}$, `g` $\in \mathcal{G}$, `t` $\in \mathcal{T}$, `i` $\in \mathcal{I}$: Die y-Koordinate der durch die Parameter spezifizierten Figur.

- `mannX[g][t]` $\in \mathbb{R}$, `g` $\in \mathcal{G}$, `t` $\in \mathcal{T}$: Der x- Wert der entsprechenden Stange.

- Die Einträge `rotation[g][t]` $\in [-180, 180]$, `VRot[g][t]` $\in \mathbb{R}$ sowie `VY[g][t]` $\in \mathbb{R}$ kennzeichnen für jede Stange die derzeitige Rotation in °, die Rotationsgeschwindigkeit (in °/s) und die Geschwindigkeit der Bewegung in Richtung der y-Achse in mm/s. Für die Indizes gilt in jedem dieser Fälle: `g` $\in \mathcal{G}$, `t` $\in \mathcal{T}$

5.1 Analyse des Weltmodells

Die im Weltmodell zusammengefassten Daten werden teilweise in ihrer rohen Form verwendet; vor allem für die Aktionen ist es aber wichtig, diese Daten in Bezug auf die vermutete Weiterentwicklung des Spieles zu interpretieren. Ein Großteil dieser Interpretationen ist eher spezieller Natur und daher in den Aktionen selbst angesiedelt. Andere Untersuchungen werden entweder so häufig benötigt oder sind derartig kompliziert, dass eine zentrale Analyse des Weltmodells in diesen Punkten von Vorteil ist. Allen dieser Routinen ist gemeinsam, dass sie auf oft groben Vereinfachungen der tatsächlichen physikalischen Vorgänge basieren; ein weiterer Vorteil der zentralen Zusammenfassung ist, dass Verbesserungen dieser Methoden, inklusive Annäherungen an die tatsächliche Welt, sofort global zur Verfügung stehen. Die derzeit zentral durchgeführten Analysen werden im folgenden kurz beschrieben.

5.1.1 Ballvorhersagen

Die unter diesem Punkt zusammengefassten Methoden geben Schätzungen über den zukünftigen Lauf des Balls ab, wobei davon ausgegangen wird, dass *die Figuren den Lauf des Balls nicht beeinflussen.* Weiterhin gelten folgende vereinfachenden Annahmen:

- Verlustfreie Bewegung: Unabhängig davon, wie lange der Ball sich bewegt, und wie oft er mit den Wänden kollidiert, verändert sich seine Geschwindigkeit nicht (wobei sich die Werte für `ballVX` und `ballVY` durchaus ändern können; entscheidend ist, dass $\texttt{ballVX}^2 + \texttt{ballVY}^2$ konstant bleibt). Diese Annahme ist bei den für das Tischfußballspiel typischen schnellen Aktionswechseln sinnvoll.

- „Einfallswinkel = Ausfallswinkel" gilt bei Kollisionen mit den Wänden uneingeschränkt. Die Abweichungen sind in der Realität im Allgemeinen gering und hängen von der Rotation des Balls um die senkrechte Achse ab. Diese aber ist unbekannt und durch die gegebene Hardwareausstattung auch nicht zu messen.

- Die Ecken sind beim realen Tischfußballtisch hochgezogen, um ein Liegenbleiben des Balls im von keiner Figur erreichbaren Niemannsland zu verhindern. Dies wird bei der Ballvorhersage ignoriert, kann aber bei weiteren Verfeinerungen relativ einfach modelliert werden.

Die krasseste Vereinfachung ist allerdings:

- Der Ball wird von den Querwänden (= Torwänden) nicht reflektiert. Diese Einschränkung ist notwendig, um für einen gegebene X-Koordinate eine eindeutige Schätzung zu erhalten, wann der Ball sie passieren wird – oder passiert hat.

Aufgrund dieser teilweise starken Vereinfachungen ist klar, dass alle aus der Analyse des Weltmodells gewonnenen Aussagen systematische Fehler aufweisen. Ihre Berechtigung finden sie jedoch darin, dass ihre Bestimmung nur in Kurzzeitvorhersagen liegt: Bei kurzen Vorhersageintervallen aber gehen die durch diese Vereinfachungen entstandenen Fehler im Rauschen der Sensoren unter.

Zeitvorhersage in X-Richtung

Zu einer gegebenen X-Koordinate x und einem gegebenen Weltzustand s wird errechnet, wann der Ball bei konstanter $s.\texttt{ballVX}$ x erreicht oder erreicht hat. $s.\texttt{ballVX}$ muss dabei ungleich Null sein.

$$\texttt{zeitBisX}(x,s) = \frac{x - s.\texttt{ballX}}{s.\texttt{ballVX}} \tag{5.1}$$

Zeitabhängige Vorhersage der Y-Koordinate des Balls

Durch den Aufruf von $\texttt{yVorhersage}(\texttt{s},\texttt{t})$ für einen Weltzustand $s \in \mathcal{S}$ und einen Zeitraum $\texttt{t} \in \mathbb{R}$ (der Buchstabe „t“ wird sowohl als Bezeichnung für die Zeit als auch als Index für die Stangen benutzt, da Verwechselungen des unterschiedlichen Kontextes wegen nicht möglich sind) findet die Berechnung der Y-Koordinate des Balls nach $\texttt{t}$ Sekunden unter den oben beschriebenen vereinfachenden Annahmen statt. Dies geschieht in zwei Schritten: zunächst wird die „virtuelle“ Y-Koordinate des Balls errechnet; es ist dies die Koordinate, die der Ball erreichen würde, wenn das Spielfeld keine Seitenwände hätte. Daraufhin wird der Ball wieder in das Spielfeld zurückprojiziert.

5.1.2 Ballanalysen relativ zu den Spielfiguren

Unter dieser Rubrik finden sich Prozeduren, die Aussagen über den derzeitigen oder zukünftigen Ort des Balls relativ zu den Spielfiguren treffen. Diese bauen teilweise stark auf den oben aufgeführten Ballprognosen auf und teilen deshalb auch deren Schwächen.

Kürzester Y-Abstand zu einer Figur

Eine regelmäßig auftretende Frage ist die nach dem kürzesten Y-Abstand einer Figur einer bestimmten Stange zum Ball in einem gegebenen Weltzustand s, also

$$\texttt{minAbstand}(g,t,s) = \min_{i \in \mathcal{I}} |s.\texttt{mannY}[g][t][i] - s.\texttt{ballY}| \tag{5.2}$$

für $s \in \mathcal{S}, g \in \mathcal{G}, t \in \mathcal{T}$. Die unterschiedliche Anzahl von Figuren pro Stange wird innerhalb dieser Funktion gehandhabt.

Nächster Einflussbereich

Bälle, deren Abstand zu einer Stange ein gewisses Maß unterschreitet, können von dieser kontrolliert gespielt werden. Durch die Funktion `bereich`(s), $s \in \mathcal{S}$ wird ein Index der Stange, deren Einflussbereich der Ball als nächstes erreicht, angegeben; aus diesem Index lassen sich der Besitzer und Typ dieser Stange eindeutig feststellen. Diese Aktion berücksichtigt die Reflexion des Balls an den Querwänden, d.h. die nächste Stange, in deren Einflussbereich ein hinter einer Torwartstange befindlicher und sich von dieser (dementsprechend nach hinten) entfernender Ball eintreten wird, ist die Torwartstange selbst. In diesem Fall wird die Reflexion also implizit mitberücksichtigt.

Festzuhalten ist, dass diese Funktion keine Aussage darüber trifft, durch welche Stange der Ball tatsächlich gespielt werden kann, sondern nur, in welchem Einflussbereich sich der Ball demnächst aufhalten wird. Die Frage nach der tatsächlichen Spielbarkeit eines Balls wird im folgenden Punkt behandelt.

Erreichbarkeit

Ob, wo und wann der Ball von einer gegebenen Stange erreicht werden kann ist nur unter Verwendung einer weiteren Annahme möglich: Die Annahme, dass es eine bekannte Maximalgeschwindigkeit gibt, mit der die Spieler ihre Stangen in Y-Richtung bewegen können. Für die Richtigkeit dieser Annahme spricht, dass die Geschwindigkeit und Beschleunigung, mit denen der Roboter seine Stangen bewegen kann, höher als bei Menschen sind. Diese Maximalwerte aber sind bekannt. Vereinfachend wird bei der Errechnung der Erreichbarkeit auf die Berücksichtigung der Beschleunigung verzichtet; der hierdurch entstehende Fehler erscheint hinnehmbar, da die tatsächlich sehr hohe Beschleunigung (in der Größenordnung von über 10000 mm/s^2) den Fehler marginalisiert.

Die Berechnung der Erreichbarkeit vollzieht sich nun folgendermassen: Gegeben ist ein Weltzustand $s \in \mathcal{S}$; der Zeitpunkt s.`zeitpunkt` heißt im folgenden t_0. Der Verlauf der Y-Koordinate des Balls ist nun durch die Funktion `yVorhersage`(s, t) eindeutig bestimmt. Desweiteren existieren für eine gegebene Stange r Abschätzungen t_e respektive t_a für die Zeitpunkte, zu denen der Ball unter Beibehaltung der derzeitigen Bewegungsparameter in den Einflussbereich der Stange eintritt bzw. aus diesem austritt. Liegt der Zeitpunkt t_a in der Zukunft, so gibt es für jede Figur auf r ein Intervall auf der Y-Achse, das durch diese bis zum Zeitpunkt t_a abgedeckt werden kann. Grafisch verdeutlichen lässt sich dieser Sachverhalt durch ein Koordinatensystem, bei dem als Abszisse der Zeitverlauf aufgetragen ist; hier lässt sich der abdeckbare Bereich zu jedem Zeitpunkt durch einen Kegel darstellen, bei dem der Y-Wert des Schnittpunktes der Schenkel durch die Y-Koordinate der jeweiligen Figur gegeben ist und ihr Y-Abstand zum Zeitpunkt t_0 genau der Breite der Figur entspricht; die Steigung der Schenkel ist durch die Maximalgeschwindigkeit gegeben. Schneidet sich die Bahn des Balls nun zu einem Zeitpunkt t_{s_1} mit $t_{s_1} \leq t_a$ mit einem der Schenkel und zu einem Zeitpunkt t_{s_2} mit $t_{s_2} \geq t_a$ mit dem anderen, so ist ein Erreichen des Balls durch die entsprechende Figur frühestens zum Zeitpunkt $\max\{t_a, t_{s_1}\}$ möglich; sind diese Bedingungen nicht erfüllt, so ist der Ball für diese Figur nicht erreichbar. Der frühste Zeitpunkt der Erreichbarkeit des Balls durch eine Stange ist das Minimum der derartigen Zeitpunkte für die Figuren auf der Stange. Abbildung 5.2 verdeutlicht dies nochmals.

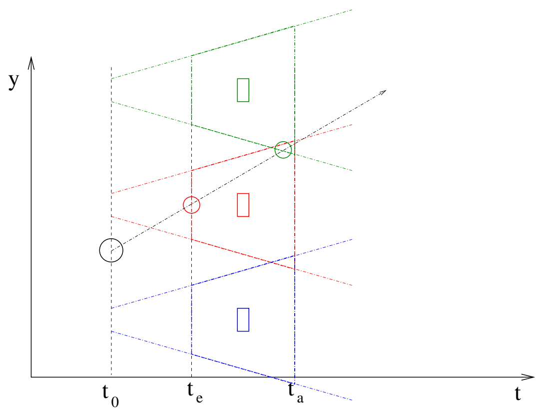


Abbildung 5.2: Beispiel für die Erreichbarkeit eines Balls

Aktiver Spieler

Der aktive Spieler ist derjenige, der als nächster den Ball erreichen kann.

5.1.3 Divergierende Weltzustände

Durch das der Bilderkennung immanente und dem Simulator injizierte Rauschen ist es oft der Fall, dass sich der tatsächliche Spielzustand inkonsistent zu den im Weltmodell zusammengefassten Werten verändert. Bis zu einem gewissen Grad muss dies von den Aktionen ausgeglichen werden können; überschreiten die Sprünge dieses Maß hingegen, so kann dieser Anspruch nicht mehr gestellt werden, so dass aktive Aktionen u.U. abgebrochen werden müssen oder das entsprechende Weltmodell nicht mehr in die Datenbasis für die verwendeten Lernalgorithmen gelangen darf.

Um diese divergierenden Folgen von Weltzuständen erkennen zu können, wurde das Prädikat `divergierend(prae,post)` eingeführt, das für die Weltzustände `prae` und `post` aufgrund von Vergleichen der Zeitstempel, der Geschwindigkeiten sowie des Unterschiedes zwischen der aus den Daten aus `prae` errechneten Hypothese über die Ballposition zu `post.zeitpunkt` und der tatsächlichen Ballposition in `post` die Beurteilung, ob die Abfolge der Weltzustände divergiert, übernimmt.

Kapitel 6

Der Aktionensatz

Grundbausteine des Spieles sind die in der Menge $\mathcal{A}$ zusammengefassten Aktionen. Jede Stange des Spieles wird von genau einer Aktion $a \in \mathcal{A}$ gesteuert, wobei durch die Aktion die Reaktionen der entsprechenden Spielfiguren auf die auftretenden Situationen festgelegt werden. Die Aktionen sind in der Absicht, eine hohe Wartbarkeit und einfache Erweiterbarkeit sowie eine einfache Handhabung innerhalb des Spieles zu gewährleisten, als Objekte implementiert. Grundlegende Eigenschaft aller Aktionen ist es, dass sie ihre Informationen nur explizit von der jeweiligen Umgebung erhalten und die Reaktionen auf diese sich nur intern abspielen; diese Eigenschaft ermöglicht es, zu jedem Zeitpunkt eine unbegrenzte Anzahl von Aktionen zu erzeugen, verwenden oder vernichten, ohne dass es zu Interferenzen kommt, und noch dazu jeder dieser Aktionen gezielt Informationen zukommen zu lassen.

6.1 Allgemeine Eigenschaften

Um eine einheitliche Schnittstelle zu gewährleisten gibt es einen Satz von Methoden, der von jeder Aktion bereitgestellt wird. Durch diesen Aufbau wird eine Kapselung erreicht, die es ermöglicht, gewisse, allgemeine Informationen über die Aktion zu erhalten oder bestimmte Modifikationen an ihr vorzunehmen, ohne den Typ der Aktion oder ihre Implementierung kennen zu müssen. Weiterhin verfügt jede Aktion, unabhängig von ihrem Typ, ein Grundwissen über sich selbst. Dieses Grundwissen umfasst

- Den eigenen Zustand: Jede Aktion befindet sich zu jedem Zeitpunkt in genau einem Zustand. Mögliche Zustände sind:

  Init: Die Aktion ist frisch erzeugt oder zurückgesetzt worden, sie hat insbesondere noch keine Informationen über den derzeitigen Spielzustand erhalten

  Vorbereitend: Die Aktion ist in den Spielverlauf eingebettet, ist aber noch nicht dazu gekommen, in diesen einzugreifen

  Ausführend: Die Aktion ist im Moment dabei, in den Spielverlauf gezielt einzugreifen

  Beendet: Die Intervention ins Spiel hat stattgefunden und ist vorbei; aus diesem Zustand lässt sich aber nicht herauslesen, ob die Aktion erfolgreich war.

- Den eigenen Typ: Dieses Wissen ist weniger für die Aktion selbst als für externe Abfragen vorhanden.

- Den Typ der von der Aktion kontrollierten Stange `stange` und dadurch die Position `x` auf der Längsachse des Spielfeldes.

- Eine Zeitabschätzung, wann der Eingriff in das Spielgeschehen stattfinden wird, sowie

- Die von der Aktion selbst gesetzten Steuerbefehle. Die Aktionen steuern die jeweiligen Stangen nicht direkt, sondern legen nur Befehle an, die von der Umgebung, innerhalb derer die Aktionen agieren – wobei es sich um den realen Tisch, eine Simulation oder den Spielbaum handeln kann – ausgelesen und ausgeführt werden.

Über dieses Grundwissen hinaus verfügen die einzelnen Aktionen über jeweils individuelles Wissen über sich selbst oder die Umgebung betreffend; soweit dieses zum Verständnis der Aktionen notwendig ist, wird es an entsprechender Stelle Erwähnung finden.

6.1.1 Allgemeine Handhabung

Die Aktionen sind Objekte, die während des Aufbaus des Spielbaumes, während des Spiels selber oder im Verlauf von Analyseroutinen laufend Vervielfältigungen und Modifikationen erfahren. Diese werden durch folgende Methoden bewerkstelligt:

Aktualisierung: Im Verlauf der Aktualisierung erhält die Aktion Kenntnis über den aktuellen Weltzustand und verändert ihren internen Zustand inklusive der Steuerbefehle dementsprechend. Insbesondere werden die momentan den realen Tischfußballtisch steuernden Aktionen in jedem Zyklus einmal aktualisiert.

Klonierung: Bei der Klonierung wird die Aktion vollständig kopiert, inklusive aller internen Variablenbelegungen.

Rücksetzung: Hierdurch werden alle veränderlichen Werte auf die Startwerte zurückgesetzt.

Neuerzeugung: Die Neuerzeugung einer gegebenen Aktion entspricht ihrer Klonierung und der nachfolgenden Rücksetzung der Kopie.

6.1.2 Aktivität und Erfolg

Zur Beurteilung der Auswirkungen einer Aktion auf das Spiel existieren zwei Kriterien: Die Aktivität und der Erfolg.

Eine Aktion heißt dann aktiv in einem Weltzustand, wenn sie im Begriff ist, gezielt in den Spielverlauf einzugreifen; im Falle eines Schusses wäre Aktivität beispielsweise dann gegeben, wenn durch die Drehung der Stange der Schuss selbst ausgeführt wird. Voraussetzung für Aktivität ist also eine gelungene Positionierung der Spielfiguren zum Ball.

Ein Erfolg ist dann gegeben, wenn eine Aktion aktiv war und der Eingriff in den Spielverlauf in der beabsichtigten Art und Weise erfolgt ist. Aktivität ist also eine Vorbedingung für einen Erfolg: Hieraus folgt, dass ein Erfolg einer Aktion immer mit einem Eingriff in den Lauf des Balls verbunden ist. Aus diesem Grunde ist die **ClearBall**-Aktion, deren Zweck es ist, den Ball passieren zu lassen, die also nie aktiv wird, definitionsgemäß niemals erfolgreich.

Zur Feststellung, ob eine Aktion $a_i \in \mathcal{A}$ an der Stange i in einem Weltzustand aktiv ist, existiert ein Prädikat $\texttt{aktiv}_{a_i}(s)$, $s \in \mathcal{S}$, das dies zu einem gegebenen Weltzustand beurteilt. Dieses Prädikat wird sowohl zur internen Steuerung als auch zur externen Analyse der Aktion eingesetzt.

Da das entsprechende Prädikat zur Erkennung eines Erfolges eine Veränderung des Spielzustandes beurteilen muss, wird es mit zwei Parametern aufgerufen: $\texttt{erfolg}_{a_i}(s_1, s_2)$, $s_1, s_2 \in \mathcal{S}$ ist dann wahr, wenn die Aktion im Weltzustand s_1 aktiv ist und sich die den Ball betreffenden Parameter des Weltzustandes s_2 im Gegensatz zu s_1 in einer Form verändert haben, die als Erfolg erkannt wird. Um sicherzugehen, dass zwischen s_1 und s_2 ausschließlich die betreffende Aktion eingegriffen haben kann, muss gelten, dass die Aktion in s_1 bereits aktiv ist und der Zustand s_2 so kurz wie möglich nach der Ausführung der Aktion aufgenommen wurde; tatsächlich muss das Prädikat sogar dann korrekte Ergebnisse liefern, wenn die Aktion gerade ihre Ausführung beendet hat. Da in diesem Fall die von der Bilderkennung gelieferten Werte für die Bewegung des Balls oft äußerst ungenau sind, weichen die Implementierungen der Prädikate unter Umständen von der intuitiv „richtigen“ Form ab.

Der Zeitraum, innerhalb dessen eine Aktion die Steuerung einer Stange vornimmt, wird in Phasen eingeteilt. Eine solche Phase beginnt mit der Auswahl der Aktion oder dem Ende der vorhergehenden Phase; das Ende einer Phase wird festgestellt, wenn

- Die Aktion aktiv war;
- Der Ball den Einflussbereich der entsprechenden Stange passiert hat, ohne dass die Aktion aktiv werden konnte;
- Die Aktion abgelöst wurde, ohne dass einer der beiden vorigen Fälle eingetreten ist.

Um beurteilen zu können, wie die Konsequenzen der Auswahl einer Aktion a_i an der Stange i in einem Weltzustand s sein werden, stellt jede Aktion zwei Methoden zur Verfügung, die die Wahrscheinlichkeiten $P(\texttt{aktiv}_{a_i}|s)$ und $P(\texttt{erfolg}_{a_i}|s)$, ob a_i in der mit s beginnenden Phase aktiv, respektive erfolgreich sein wird, zurückliefern. Die zurückgelieferten Werte gelten nur unter der Voraussetzung, dass die Aktion nicht abgebrochen wird, bevor sie aktiv war oder der Ball ihren Einflussbereich passiert hat.

Unter Verwendung der Prädikate `aktiv` und `erfolg` lassen sich diese Wahrscheinlichkeiten durch die Analyse des Spielverlaufs schätzen.

6.2 Die Schätzung der Erfolgswahrscheinlichkeiten

Im folgenden soll die eingesetzte Methode zum Lernen der Erfolgs-und Aktivitätswahrscheinlichkeiten $P(\texttt{erfolg}_{a_i}|s)$ und $P(\texttt{aktiv}_{a_i}|s)$ vorgestellt werden. Dabei wird zunächst das eingesetzte Schätzverfahren demonstriert und werden die zum Einsatz notwendigen Daten benannt; daraufhin wird erklärt, wie diese benötigten Daten gesammelt werden.

Um die Menge benötigter Daten überschaubar zu halten werden aus dem Weltzustand s folgende Informationen extrahiert:

- Distanz des Balls zur Stange i in X-Richtung
- Geschwindigkeit des Balls in X-Richtung relativ zur Stange i (also „näherkommend“, „entfernend“)
- Minimale Distanz des Balls zu einer der Figuren auf der Stange i in Y-Richtung
- Geschwindigkeit des Balls in Y-Richtung relativ zu dieser Figur

Diese Werte werden nun diskretisiert, indem der Wertebereich in jeweils sieben überschneidungsfreie Intervalle aufgespalten wird; diese Aufteilung bei den Distanzen zeigt Abbildung 6.1.

Jeder Weltzustand lässt sich jetzt als ein 4-Tupel von Intervallkennzeichnern darstellen. Wichtig für das Verständnis des folgenden ist, sich zu vergegenwärtigen, dass nicht die Erfolgswahrscheinlichkeit in einem Zustand errechnet werden soll, sondern die Erfolgswahrscheinlichkeit in einer Phase, die mit diesem Zustand beginnt.

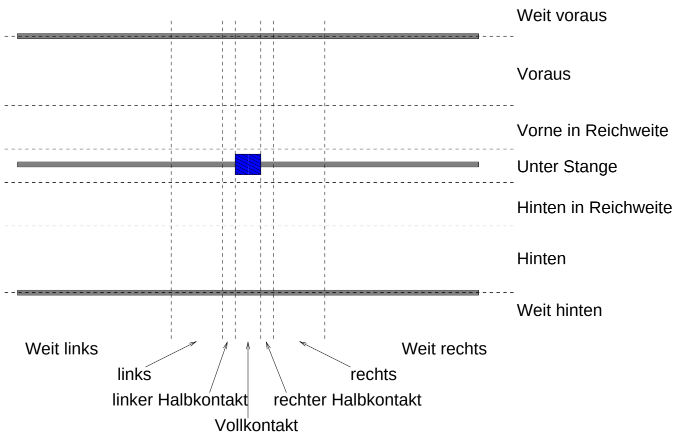


Abbildung 6.1: Die Einteilung der Ballentfernung in Intervalle

Das derzeit eingesetzte Verfahren zum Lernen der Erfolgs- und Aktivitätswahrscheinlichkeiten ist der *naive Bayesschätzer*[MITCHELL 1997]. Da sich das Vorgehen zur Schätzung der beiden Werte durch nichts unterscheidet, soll es hier nur anhand seiner Anwendung bei der Erfolgswahrscheinlichkeit vorgestellt werden; die Schätzung des Wertes für die Aktivität verläuft analog. Um die Schreibweise kompakt zu halten ist „$\texttt{erfolg}_{a_i}$“ im folgenden verkürzend durch „E“ ersetzt.

Gegeben:

- Eine Menge von überschneidungsfreien Entfernungsintervallen in X-Richtung $\mathcal{X}$
- Eine Menge von überschneidungsfreien Entfernungsintervallen in Y-Richtung $\mathcal{Y}$
- Eine Menge von überschneidungsfreien Geschwindigkeitsintervallen in X-Richtung $\mathfrak{X}$

- Eine Menge von überschneidungsfreien Geschwindigkeitsintervallen in Y-Richtung $\mathfrak{Y}$
- Eine in diese Intervalle klassifizierte Ballbeobachtung $\langle d_x, d_y, v_x, v_y \rangle, d_x \in \mathcal{X}, d_y \in \mathcal{Y}, v_x \in \mathfrak{X}, v_y \in \mathfrak{Y}$

Gesucht ist nun:

Die Erfolgswahrscheinlichkeit $P(E|d_x, d_y, v_x, v_y)$.

Nach der Bayesschen Regel gilt:

$$P(E|d_x, d_y, v_x, v_y) = \frac{P(d_x, d_y, v_x, v_y|E) \cdot P(E)}{P(d_x, d_y, v_x, v_y)} \tag{6.1}$$

An dieser Stelle werden zwei Unabhängigkeitsannahmen getroffen:

1. Die Ereignisse aus $\mathcal{X}, \mathcal{Y}, \mathfrak{X}$ und $\mathfrak{Y}$ sind unabhängig voneinander, es gilt also
$$P(d_x, d_y, v_x, v_y) = P(d_x) \cdot P(d_y) \cdot P(v_x) \cdot P(v_y)$$
Diese Annahme ist gerechtfertigt. Sie gilt für ein homogenes Spielfeld, und nicht in den ansteigenden Ecken. Dort aber sind Aktionen ohnehin nicht möglich.

2. Die Naive-Bayes-Annahme gilt, d.H. die Wahrscheinlichkeiten der einzelnen Ereignisse sind unabhängig voneinander, gegeben, dass `erfolg` gilt, also
$$P(d_x, d_y, v_x, v_y|E) = P(d_x|E) \cdot P(d_y|E) \cdot P(v_x|E) \cdot P(v_y|E)$$
Diese Annahme ist – wie es bei der Verwendung des Naive-Bayes-Ansatzes zum guten Ton gehört – nicht gerechtfertigt, wird aber getroffen, da sie durch eine deutliche Einschränkung der Darstellungskomplexität des Verbundwahrscheinlichkeitsraumes die Berechnung stark vereinfacht. In unserem Falle mit nur sieben Ereignissen pro Kategorie würde alleine die Repräsentation der $P(d_x, d_y, v_x, v_y)$ $7^4 = 2401$ Einträge fordern; um jeden dieser Einträge vernünftig lernen zu können, müssen mehrere Ereignisse mit dieser Konfiguration beobachtet werden

– das Lernen verliefe dementsprechend langsam und wäre in frühen Phasen äußerst fehleranfällig. Falls man vorhätte, jeden dieser Werte von Hand einzutragen, kann davon ausgegangen werden, dass die entstehende Wahrscheinlichkeitsverteilung zumindest stark fehlerhaft, wenn nicht vollkommen bedeutungslos, wäre. Durch die Verwendung der Naive-Bayes-Annahme schmilzt die Anzahl benötigter Einträge auf 28, also auf ca. 1 %, zusammen. Dies vereinfacht den Einsatz von Lernalgorithmen und ermöglicht die Verwendung manuell eingegebener Werte und ist dadurch als Näherung sinnvoll.

Unter diesen Annahmen lässt sich die Gleichung 6.1 durch die Errechnung der Naive-Bayes-Wahrscheinlichkeit P_{NB} folgendermassen nähern:

$$P_{NB}(E|d_x, d_y, v_x, v_y) \hat{=} \frac{P(d_x|E) \cdot P(d_y|E) \cdot P(v_x|E) \cdot P(v_y|E) \cdot P(E)}{P(d_x) \cdot P(d_y) \cdot P(v_x) \cdot P(v_y)} \tag{6.2}$$

Der Naive-Bayes-Klassifikator verwendet diese Konstruktion, um gegebene Beispiele zu klassifizieren. Bei einem gegebenen Attributvektor $\langle a_1, a_2, \ldots, a_n \rangle$ und Hypothesenraum $\mathcal{V}$ entscheidet sich der Naive-Bayes-Klassifikator also für die Hypothese

$$h_{NB} = \arg\max_{v \in \mathcal{V}} P_{NB}(v|a_1, a_2, \ldots, a_n)$$

Für die Verwendung als Klassifikator reicht es also, wenn gilt:

$$\begin{aligned} P(v_1|a_1, a_2, \ldots, a_n) &< P(v_2|a_1, a_2, \ldots, a_n) \\ \Rightarrow P_{NB}(v_1|a_1, a_2, \ldots, a_n) &< P_{NB}(v_2|a_1, a_2, \ldots, a_n) \\ &\qquad \forall v_1, v_2 \in \mathcal{V} \end{aligned}$$

Im Falle der expliziten Wahrscheinlichkeitsschätzung reicht es aber nicht aus, wenn diese Bedingung erfüllt ist; hier müssen weitere Forderungen gestellt werden, nämlich

$$\begin{aligned} \forall v \in \mathcal{V} : 0 \leq P_{NB}(v|a_1, a_2, \ldots, a_n) &\leq 1 \\ \sum_{v \in \mathcal{V}} P_{NB}(v|a_1, a_2, \ldots, a_n) &= 1 \end{aligned}$$

Durch den in Gleichung 6.2 gegebenen einfachen Schätzer wird diese Bedingung verletzt; insbesondere nehmen die durch ihn vorhergesagten Wahrscheinlichkeiten oft Werte größer als 1 an – kleiner als Null werden sie dahingegen nie.

Normierung

Da im gegebenen Kontext nur die Wahrscheinlichkeiten binärer Ereignisse abgeschätzt werden müssen, besteht eine Möglichkeit der Normierung darin, unter Verwendung der für korrekte Wahrscheinlichkeitswerte für diesen Fall geltenden Aussage $P(x) = \frac{P(x)}{P(x)+P(\neg x)}$ die Ergebnisse P_{NB} des Naive-Bayes-Verfahrens folgendermassen zu den obigen Bedingungen gehorchenden, einsetzbaren Wahrscheinlichkeitswerten P_{NormNB} normieren:

$$P_{NormNB}(E|d_x, d_y, v_x, v_y) = \frac{P_{NB}(E|d_x, d_y, v_x, v_y)}{P_{NB}(E|d_x, d_y, v_x, v_y) + P_{NB}(\neg E|d_x, d_y, v_x, v_y)} \tag{6.3}$$

Der Wert für $P_{NB}(\neg E|d_x, d_y, v_x, v_y)$ berechnet sich hierbei nach Gleichung 6.2:

$$P_{NB}(\neg E|d_x, d_y, v_x, v_y) \hat{=} \frac{P(d_x|\neg E) \cdot P(d_y|\neg E) \cdot P(v_x|\neg E) \cdot P(v_y|\neg E) \cdot P(\neg E)}{P(d_x) \cdot P(d_y) \cdot P(v_x) \cdot P(v_y)} \tag{6.4}$$

Die dieser Normierung zugrundeliegende Annahme ist die, dass die durch den Naiven Bayes-Ansatz ermittelten Werte nicht nur in der Reihenfolge, sondern auch in der Relation der Absolutbeträge ausreichend nah an den realen Werten liegen.

Benötigte Daten

Welche Daten sind notwendig, um dieses Verfahren einsetzen zu können, und welche Bedingungen müssen sie erfüllen?

Gleichung 6.2 fordert Kenntnis über folgende Größen:

- $P(d_x), P(d_x|E), d_x \in \mathcal{X}$

- $P(d_y), P(d_y|E), d_y \in \mathcal{Y}$
- $P(v_x), P(v_x|E), v_x \in \mathfrak{X}$
- $P(v_y), P(v_y|E), v_y \in \mathfrak{Y}$
- $P(E)$

Da $P(\neg E) = 1 - P(E)$ gilt, werden für die Gleichung 6.4 nur noch folgende Werte benötigt:

- $P(d_x|\neg E), d_x \in \mathcal{X}$
- $P(d_y|\neg E), d_y \in \mathcal{Y}$
- $P(v_x|\neg E), v_x \in \mathfrak{X}$
- $P(v_y|\neg E), v_y \in \mathfrak{Y}$

Das Verfahren zur Herleitung dieser Werte ist bei den einzelnen Einträgen dasselbe, deshalb soll die Herleitung von dieser anhand von $P(d_x|\neg E)$ dargestellt werden.

Nun gilt nach $P(A|B) = \frac{P(A \wedge B)}{P(B)}$:

$$\begin{aligned} P(d_x|\neg E) &= \frac{P(d_x \wedge \neg E)}{P(\neg E)} \\ &= \frac{P(d_x \wedge \neg E)}{1 - P(E)} \end{aligned}$$

Da $P(d_x \wedge \neg E) + P(d_x \wedge E) = P(d_x)$ gilt, folgt:

$$P(d_x|\neg E) = \frac{P(d_x) - P(d_x \wedge E)}{1 - P(E)}$$

und durch $P(d_x \wedge E) = P(d_x|E) \cdot P(E)$

$$P(d_x|\neg E) = \frac{P(d_x) - P(d_x|E) \cdot P(E)}{1 - P(E)}$$

Es werden also für die Normierung keine zusätzlichen Werte benötigt, vorausgesetzt, die gegebenen Werte sind realistisch, insbesondere muss die Konsistenzbedingung

$$P(d_x) > P(d_x|E) \cdot P(E) \tag{6.5}$$

gelten. Diese aber ist durch die willkürliche Eingabe der Werte durch einen menschlichen Editor gefährdet.

Es stellt sich nun die Frage, wie diese Werte in der Implementierung am besten dargestellt werden. Die Kriterien, nach denen dies beurteilt werden muss, sind folgende:

- Korrektheit: Die atomaren Wahrscheinlichkeitswerte müssen den üblichen Bedingungen für Wahrscheinlichkeitswerte entsprechen
- Konsistenz: Die Werte müssen widerspruchsfrei abgelegt werden
- Verständlichkeit: Die Form, in der die Werte abgelegt werden, sollte nicht nur für den Computer les- und interpretierbar sein; insbesondere soll die manuelle Optimierung einzelner Parameter möglich sein.
- Eignung für das maschinelle Lernen: Die abgelegten Werte sollten durch während des Spielverlaufs gewonnene Erfahrungen erweiterbar sein.

Prinzipiell gibt es hierzu zwei Alternativen, nämlich die Darstellung als Wahrscheinlichkeitswerte sowie die Darstellung durch Ereignistabellen

Erstere hat den Vorteil, dass die Wahrscheinlichkeitswerte direkt ausgelesen und verwendet werden können; des weiteren sind sie als für Menschen lesbar anzusehen. Die üblichen Bedingungen für Wahrscheinlichkeitswerte, also z.B. $\sum_{v_< \in \mathcal{V}} P(v) = 1$ können auch bei manueller Eingabe der Werte einfach garantiert werden; größere Probleme zeigen sich hier bei der Konsistenz: Die Einhaltung der Bedingung 6.5 ist für den menschlichen Editor, der die Werte nach Augenmaß eingibt, nicht sofort offensichtlich. Weiterhin müssen zur Verwendung im Kontext des maschinellen Lernens Hilfsvariablen eingeführt werden, um festzulegen, in welchem Umfang die gegebenen Werte sich durch Hinzunahme neuer Erfahrungen verändern.

Die Probleme bei der Verwendung von Ereignistabellen sind anders geartet; der Aufbau einer Ereignisse für die beispielhafte Menge $\mathcal{X} = \{d_{x_1}, \cdots, d_{x_i}\}$

Ereignis	Gesamtanzahl	Anzahl von Erfolgen $d_{x_k} \wedge E$
d_{x_1}	n_1	m_1
$\cdots$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$
d_{x_i}	n_i	m_i

Tabelle 6.1: Format einer Ereignistabelle für Ereignisse aus $\{d_{x_1}, \cdots, d_{x_i}\}$

ist in Tabelle 6.1 dargestellt. Zur Errechnung der benötigten Wahrscheinlichkeitswerte gilt nun:

$$
\begin{aligned}
P(d_{x_k}) &= \frac{n_k}{\sum_{h=1}^{i} n_h} \\
P(d_{x_k}|E) &= \frac{m_k}{\sum_{h=1}^{i} m_h} \\
P(E) &= \frac{\sum_{h=1}^{i} m_h}{\sum_{h=1}^{i} n_h}
\end{aligned}
$$

Zur Einhaltung der Korrektheitsbedingungen genügt nun, dass $\forall k : n_k \geq m_k$ und $n_k > 0$ gilt; schwieriger gestaltet sich die Erfüllung der Konsistenzbedingungen: Da sich aus jeder der Tabellen für die Mengen $\mathcal{X}, \mathcal{Y}, \mathfrak{X}, \mathfrak{Y}$ der Wert für $P(E)$ herleiten lässt, müssen die Tabellen in einer Form zusammengesetzt werden, dass dieser Wert für alle Tabellen identisch ist. Am einfachsten lässt sich dies dadurch erreichen, dass die Gesamtanzahl der Ereignisse sowie die Gesamtanzahl der Erfolge in jeder Tabelle identisch ist: Hierdurch ist die Gleichheit der Werte für $P(E)$ garantiert. Des weiteren kommt so eine weitere Stärke der Ereignistabellen zum Tragen: Im Zuge des Spieles aufgezeichnete Ballbeobachtungen können, nachdem sie in ein 4-Tupel der Form $\langle d_x, d_y, v_x, v_y \rangle, d_x \in \mathcal{X}, d_y \in \mathcal{Y}, v_x\mathfrak{X}, v_y \in \mathfrak{Y}$ klassifiziert wurden, direkt in die Ereignistabellen aufgenommen werden; die Anzahl der in der Tabelle bereits aufgeführten Ereignisse entspricht also der Trägheit der Tabelle: Tabellen mit wenigen Einträgen reagieren schnell auf neue Beobachtungen, während sich die Werte bei „schweren“, mit großen Mengen an Beobachtungen aufgefüllten Tabellen nur sehr langsam verändern.

Aus diesem Grunde wurde der Repräsentation durch Ereignistabellen der

Vorzug gegeben.

Durchführung des Lernens

Die für die Schätzung der Wahrscheinlichkeitswerte notwendigen Daten werden während des Spieles gesammelt: Jedem Aktionstyp ist ein Schätzer zugeordnet, der aus den dieser Aktion während ihrer Aktualisierungen mitgeteilten Weltzuständen die erforderlichen Informationen extrahiert. Diese Extraktion verläuft mehrstufig: Zunächst werden die gelieferten Weltzustände nach einer Filterung in einer Liste gesammelt, um dann en bloque analysiert zu werden.

Die Vorgang der Filterung durchläuft mehrere Stufen:

1. Prüfung auf Echtheit: Mithilfe des im Weltzustand mitgeführten `echt`-Stempels wird erkannt, ob die betreffende Aktion momentan tatsächlich spielt oder ihre Aktualisierung im Verlauf der Vorwärtssimulation stattfindet; in letzterem Fall wird der gelieferte Weltzustand ignoriert.

2. Prüfung auf Divergenz: Die Ballerkennung ist, wie bereits erwähnt, stark fehlerbehaftet. Da solche Fehler im schlimmsten Falle – z.B. wenn der Ball zwischen zwei Weltzuständen „springt“ – den Erfolg der Aktion unmöglich machen, ohne dass dies auf die Aktion selbst zurückzuführen wäre, müssen solche Weltzustände herausgefiltert werden. Außerdem wird über solche Sprünge erkannt, ob die Aktion zwischenzeitlich inaktiv war.

   Das Kriterium zur Erkennung eines solchen „divergierenden“ Weltzustandes ist in Kapitel 5.1.3 dargestellt worden. Nichtdivergierende Weltzustände werden sofort in die Liste aufgenommen; wird ein divergierender Weltzustand erkannt, so ergeht eine weitere Untersuchung, ob der Ball sich innerhalb der Reichweite der Aktion befindet: ist der Fall, so wird der Weltzustand trotz der Abweichung in die Liste aufgenommen. Der Grund hierfür liegt darin, dass die Ballerkennung in Figurennähe mit hoher Wahrscheinlichkeit fehlerbehaftet ist; dies müssen die Aktionen kompensieren können, wollen sie erfolgreich sein.

Wird ein `echt`er Weltzustand nicht in die Liste aufgenommen, so wird diese analysiert und anschließend geleert.

Zur Analyse wird die Liste von Weltzuständen in die in Abschnitt 6.1.2 eingeführten Phasen zerlegt; ist eine Phase dann in ihrer vollen Länge bekannt, kann unter Verwendung der Aktivitäts- und Erfolgsprädikate der entsprechenden Aktion erkannt werden, ob die Aktion in der Sequenz aktiv oder erfolgreich sein konnte. Hierauf wird für jeden Weltzustand der Phase das 4-Tupel von Klassifikationen errechnet; falls (wie es häufig vorkommt) eines dieser Tupel mehrfach hintereinander in der Liste auftaucht, werden die Duplikate entfernt, um zu verhindern, dass z.B. Situationen, in denen der Ball sich langsam bewegt, aufgrund der größeren Anzahl von Weltzuständen in der Phase die Schätzung dominieren und verfälschen. Diese Gefahr besteht aufgrund der in Gleichung 6.2 als gültig angenommenen unkorrekten Unabhängigkeitsannahmen.

Die somit entstehende Liste von 4-Tupeln wird in die Ereignistabelle übernommen, indem für jedes Vorkommen einer Klassifikation in einer der vier Kategorien in der entsprechenden Ereignistabelle der Eintrag an der korrespondierenden Stelle in der „Gesamtanzahl“-Spalte um eins erhöht wird; ist die Phase als „aktiv“ oder „ erfolgreich“ beurteilt worden, so werden die Werte in den jeweiligen Spalten ebenfalls inkrementiert.

Initialisierung der Lernalgorithmen

Da im Verlauf der Abschätzung der Wahrscheinlichkeiten das Produkt der atomaren Wahrscheinlichkeitswerte verwendet wird, müssen diese mit Werten ungleich Null initialisiert werden, da das Auftreten einer einzigen Null zwangsläufig dazu führt, dass das Ergebnis der Schätzung entweder ungültig (bei einer Division durch Null) oder gleich Null wird; letzteres hätte zur Folge, dass die entsprechende Aktion nicht ausgewählt wird und somit die Ereignistabelle unverändert bleibt. Die entsprechende Aktion stünde also in allen Spielsituationen, in denen der dieser Wert Null annimmt, nicht zur Verfügung. Einen Ausweg aus dieser Misere bietet das sogenannte m-estimate [Cestnik 1990]: Hierbei werden a-priori-Abschätzungen der einzelnen Werte, gewichtet mit dem Vertrauen in diese Abschätzung, zur Initialisierung des Lernalgorithmus verwendet. Da die Wahrscheinlichkeitswerte durch Ereignistabellen repräsentiert werden, besteht in diesem Fall das Initialisieren mittels m-estimate in der Einfügung von Abschätzungen über die Häufigkeit von Ereignissen; das Vertrauen in die eigenen Schätzungen wird mittels der „Trägheit" der Tabelle ausgedrückt.

In der vorliegenden Implementierung wurden die Abschätzungen folgendermaßen vorgenommen: Jeder „Gesamtanzahl"-Eintrag wird mit 4 initialisiert, jeder „Aktivitäts"-Eintrag mit 2 und jeder „Erfolg"-Eintrag mit 1. Dies bedeutet also: Am Anfang sind alle Aktionen gleich bewertet, ihre Aktivitätswahrscheinlichkeit ist situationsunabhängig mit 0,5 und die Erfolgswahrscheinlichkeit mit 0.25 initialisiert. Durch den niedrigen Gesamtbetrag werden diese Werte sehr schnell durch die aus dem Spielverlauf gewonnenen Informationen verändert.

„Unsaubere" Modifikationen

Bedingt durch den Aufbau des Schätzers und die Initialisierung der Werte liefert der Schätzalgorithmus immer Werte ungleich Null zurück. Dies gilt auch für Situationen, in denen der Ball objektiv nicht mehr erreichbar ist. Falls ein derartiger Fall eintritt – was sich mittels des in 5.1.2 vorgestellten Verfahren erkennen lässt – wird der Schätzer nicht befragt, sondern Nullwerte für Aktivitäts- und Erfolgswahrscheinlichkeit zurückgegeben.

Nach der Einführung der den Aktionen zugrundeliegenden Prinzipien folgt nun die Vorstellung der einzelnen Aktionen.

6.3 KickBall

6.3.1 Beschreibung

Die **KickBall**-Aktion ist die Kernaktion des gesamten Tischfußballspieles: ihr Ziel ist es, den Ball so stark wie möglich von einem bestimmten Punkt aus in eine bestimmte Richtung zu schießen. Dies lässt sich auf verschiedenen Wegen bewerkstelligen, so dass die **KickBall**-Aktion in mehreren Ausprägungen aufgerufen werden kann. Diese Ausprägungen unterscheiden sich in zwei Kategorien: Eine **KickBall**-Aktion ist entweder statisch oder dynamisch und sie kann geradeaus oder nach rechts respektive links schießen.

6.3.2 Schussrichtung

Die Schussrichtung kann drei verschiedene Werte annehmen: Links, rechts und geradeaus. Durch möglichst exakte Positionierung schräg bzw. genau hinter dem Ball wird versucht, den Ball in die spezifizierte Richtung zu schießen. Siehe hierzu Abbildung 6.2. Ob dies gelingt, hängt offensichtlich stark davon ab, wie genau sich die seitliche Versetzung des Balls zum Spieler zum Zeitpunkt des Schusses vorhersagen und feststellen lässt. Ein sich schnell seitwärts bewegender Ball lässt daher die Wahrscheinlichkeit, dass der Ball sich nach dem Schuss tatsächlich in die geplante Richtung bewegt, stark abfallen. In diesem Fall hat der gerade Schuss die höchste Erfolgswahrscheinlichkeit, da sich zum Gelingen dieser Variante der Ball in einem breiteren Bereich aufhalten darf.

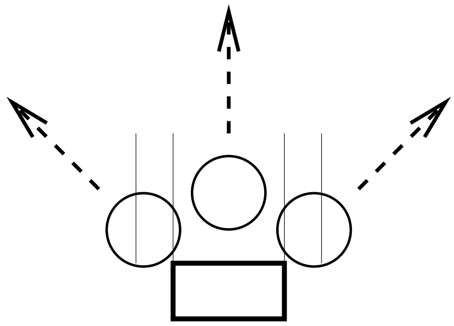


Abbildung 6.2: Die verschiedenen Schussrichtungen

6.3.3 Statische und dynamische Schüsse

Eine *dynamische* **KickBall**-Aktion errechnet in jedem Schritt die Position, an der sie den Ball am frühesten erreichen kann; daraufhin legt sie Steuerbefehle an, um rechtzeitig an der entsprechenden Position zu stehen. Da diese

Berechnung bei jeder Aktualisierung vorgenommen wird, ist die Aktion in hohem Maße in der Lage, das Rauschen der Ballerkennung zu kompensieren.

Einen grundlegend anderen Ansatz verfolgt der *statische* Schuss: Hier wird die Aktion mit einem Parameter aufgerufen, der die gewünschte Schussposition kennzeichnet. Die Aktion errechnet aus dem Weltmodell die notwendige Geschwindigkeit, um die angegebene Position vor dem Ball zu erreichen. Dort verharren die Figuren, bis der Ball an der entsprechenden Position ankommt; hierauf wird der Schuss ausgelöst. Vorteile und Nachteile der statischen Schussvariante liegen klar auf der Hand: Da im Gegensatz zum dynamischen Schuss nur der Ball in Bewegung ist, kann dieser präziser getroffen werden; dies wird allerdings mit einer gesteigerten Abhängigkeit von der Güte der Ballerkennung erkauft, da die Aktion nicht mehr in der Lage ist, gravierende Fehler dieser durch eine Veränderung der Positionierung auszugleichen.

6.3.4 Erfolg und Misserfolg

Der Erfolg der **KickBall**-Aktion ist intuitiv einfach festzustellen: Im Falle eines geraden Schusses bewegt sich der Ball vom Abschusspunkt aus mehr oder weniger genau geradeaus und tut dies mit hoher, also durch irgendeinen Schwellwert abgrenzbarer Geschwindigkeit. Leider stellte sich heraus, dass sich ein vernünftiger Schwellwert nicht finden ließ: Während bei den ständig mitlaufenden Statistiken über die Ballgeschwindigkeit Werte von über 2000 mm/s keine Seltenheit sind (und intuitiv klar ist, dass der Ball auch noch weit höhere Werte als diese zwei Meter pro Sekunde erreicht) wurden selbst Schwellwerte von unter 200 mm/s bei den globalen Messungen direkt im Anschluss an erfolgreiche Schüsse oft unterschritten.

Der Grund für dieses Phänomen ist, dass es sich beim Nachfolgeweltzustand um den Weltzustand direkt nach der Ausführung des Schusses handelt; die Bilderkennung, die ohne Wissen über die ausgeführten Aktionen die Werte der Ballbewegung ausschließlich aus den aufgenommenen Bildern errechnet, hat zu diesem Zeitpunkt noch nicht genügend Informationen, um die neuen Bewegungsparameter des Balls vernünftig abschätzen zu können. Dies ist in Darstellung 6.3 dargestellt: Die Kreise markieren den Ort des Balls zu den angegebenen Zeitpunkten, die durchgezogene Linie den tatsächlichen Weg des Balls, und die gestrichelte Linie zeigt die von der Ballerkennung angenommene Bahn des Balls.

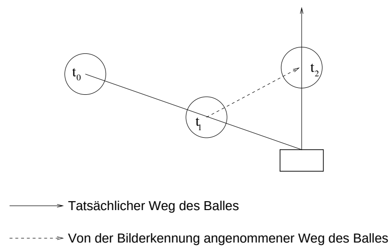


Abbildung 6.3: Fehler der Ballerkennung direkt nach Schüssen

Feststellbar ist allerdings in jedem Fall eine Erhöhung der Geschwindigkeit im Vergleich zur letzten Ballgeschwindigkeit vor dem Abschuss, und dies auch bei einer groben Geschwindigkeitsschätzung nur aufgrund der vom Weltmodell gelieferten Ballkoordinaten. Diese Schätzung der Nachschussgeschwindigkeiten `grobVX` und `grobVY` geschieht folgendermassen:

$$\begin{aligned} \texttt{grobVX} &= \frac{\texttt{post.ballX} - \texttt{prae.ballX}}{\texttt{post.zeitpunkt} - \texttt{prae.zeitpunkt}} \\ \texttt{grobVY} &= \frac{\texttt{post.ballY} - \texttt{prae.ballY}}{\texttt{post.zeitpunkt} - \texttt{prae.zeitpunkt}} \end{aligned}$$

Mit Hilfe dieser Abschätzungen lassen sich nun die Erfolgsprädikate für die unterschiedlichen Schussrichtungen links, rechts und geradeaus, indiziert durch kl, kr und kg derart definieren:

$$\begin{aligned} \texttt{erfolg}_{kg}(\texttt{prae}, \texttt{post}) &= (\texttt{grobVX} - \texttt{prae.ballVX} > 0) \\ &\quad \wedge (|\texttt{grobVY} \cdot 2| < \texttt{grobVX}) \qquad (6.6) \\ \texttt{erfolg}_{kr}(\texttt{prae}, \texttt{post}) &= (\texttt{grobVX} - \texttt{prae.ballVX} > 0) \\ &\quad \wedge (\texttt{grobVY} > 0) \qquad (6.7) \\ \texttt{erfolg}_{kl}(\texttt{prae}, \texttt{post}) &= (\texttt{grobVX} - \texttt{prae.ballVX} > 0) \\ &\quad \wedge (\texttt{grobVY} < 0) \qquad (6.8) \end{aligned}$$

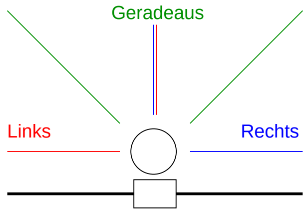


Abbildung 6.4: Zur Erfolgsbeurteilung der **KickBall**-Aktion

6.4 BlockBall

6.4.1 Beschreibung

Während es sich bei **KickBall** um eine rein offensive Aktion handelt (theoretisch ist es zwar kein Problem, rückwärts zu schießen, aber auf die Verwirklichung dieser Möglichkeit im Programm wurde aus Mangel an vernünftigen Gründen verzichtet), ist der Daseinszweck der **BlockBall**-Aktion ein rein defensiver: mit starr nach unten gerichteten Füßen wird versucht, den Ball mit größtmöglicher Sicherheit am Passieren der Stange zu hindern.

Je nach Spielsituation ändern sich die Art und Weise, auf die **BlockBall** versucht, dieses Ziel zu erreichen: In einem ersten Schritt untersucht die Aktion den zur Aktualisierung übergebenen Weltzustand, ob sie selbst den sogenannten „Frontblock“ stellt, sie also diejenige Stange kontrolliert, die der Ball als nächstes passiert. Ist dies der Fall, so unterscheidet sich das Verhalten der Aktion je nachdem, ob der Gegner der momentan aktive Spieler ist (zur Definition des aktiven Spielers siehe 5.1.2). Unter diesen Umständen positionieren Aktionen, die die Stürmer- respektive Mittelfeldstange kontrollieren, ihre Figuren auf der Y-Koordinate des Balls, während die Verteidigerstange in dieser Situation den Weg zum Tor blockiert. Die Torwartstange kann in einer Situation, in der der Gegner der aktive Spieler ist, nicht den „Frontblock“ stellen.

Falls eine **BlockBall**-Aktion sich selbst als „Frontblock“ identifiziert und feststellt, dass der Gegner nicht mehr in der Lage ist, den Ball zu erreichen, so ist klar, dass der Ball seine derzeitige Bahn weiterverfolgen wird. Diese versucht die Aktion zu antizipieren, um den Ball mit größtmöglicher Sicherheit abzufangen.

Kommt einer **BlockBall**-Aktion nicht die Aufgabe zu, den „Frontblock“ zu realisieren, so positioniert sie ihre Figuren in einer Art und Weise, die das Tor so gut wie möglich abdeckt. Das Verfahren hierzu verläuft folgenderweise: Die Aktion nimmt an, dass es sich bei den „hinter“ ihr agierenden Aktionen ebenfalls um **BlockBall**-Aktionen handelt, die auch das Tor abdecken. Die Abdeckung des Tors verläuft von der kurzen (ballnahen) zur langen Ecke: jede Aktion kann also abschätzen, bis wohin das Tor durch die hinter ihr befindlichen Stangen bereits abgedeckt wird, und stellt ihre Figuren dergestalt auf, dass dieser abgedeckte Bereich in Richtung der langen Ecke verlängert wird. Dieses Schema ist in Abbildung 6.5 verdeutlicht.

Um zu einer Entscheidung zu gelangen, wie schnell die Stangen sich bewegen müssen, um rechtzeitig die zugewiesenen Positionen zu erreichen, wird die minimale Zeit, bis der Block stehen muss, errechnet. Dies ist das Minimum aus der Zeit, die der Ball bis zum Erreichen der Stange bei unveränderter Bewegung benötigt, und der Zeit, die vergeht, bis der Gegner den Ball erreichen kann, zuzüglich der (sehr kurzen) Zeitspanne, die vergeht, bis der ohne Verzögerung vom Gegner geschossene Ball die kontrollierte Stange passiert.

6.4.2 Erfolg und Misserfolg

Der Erfolg einer **BlockBall**-Aktion `block` ist einfach festzustellen: Der Ball darf die Stange nicht passiert haben, also

$$\texttt{erfolg}_{bb}(\texttt{prae}, \texttt{post}) = ((\texttt{prae.ballX}-\texttt{block.x} > 0) \wedge (\texttt{post.ballX}-\texttt{block.x} > 0))$$

6.5 ClearBall

ClearBall nimmt im Aktionensatz insofern eine Sonderstellung ein, als es das Ziel dieser Aktion ist, den Lauf des Balls nicht zu beeinflussen; trotzdem ist diese Aktion nicht vollständig untätig, sondern sie versucht, diejenige Aktion, durch die sie irgendwann einmal abgelöst werden wird, in eine

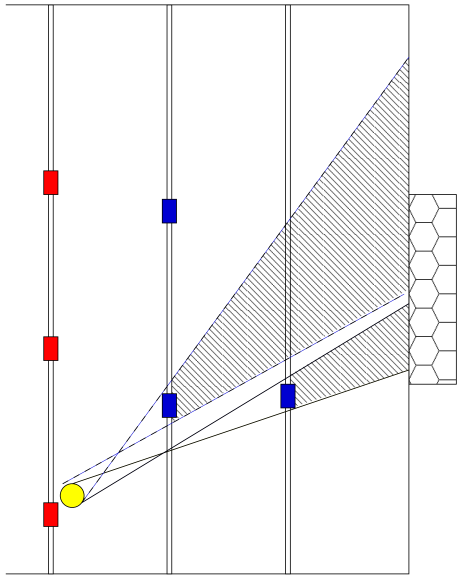


Abbildung 6.5: Abdeckung des Tors durch Verteidiger-und Torwartstange

möglichst gute Ausgangssituation zu bringen. Dies geschieht, indem **ClearBall** auf dieselbe Art dem Verlauf des Balls folgt beziehungsweise diesen zu antizipieren versucht, wobei die Figuren in die waagrechte Position gebracht werden. Durch ihren ansonsten passiven Charakter bedingt ist diese Aktion nie aktiv und greift nie erfolgreich ins Spielgeschehen ein.

6.6 StopBall

Die Aufgabe der **StopBall**-Aktion ist es, den Ball durch Einklemmen zwischen einer Figur und dem Spielfeld zu stoppen. Zu diesem Zweck passiert der Ball die Stange, um dann, während er sich bereits wieder von dieser entfernt, durch eine schnelle Drehung der Spielfiguren eingeklemmt zu werden.

StopBall ist äußerst selten erfolgreich: Am realen Tisch sind, da der Ball eine gewisse Strecke unterhalb einer Spielfigur, also für die Kamera unsichtbar, zurücklegen muss, die Schätzungen über den Ballort vollkommen unzuverlässig; der Simulator krankt – noch – daran, dass die in Wirklichkeit abgerundeten Füße der Figuren als Quader modelliert sind; der Ball spritzt also fast immer unkontrolliert weg.

Kapitel 7

Implementierung der Vorwärtssimulation

7.1 Züge

Die Definition eines Zuges ist im Falle des auf das Tischfußballspiel optimierten MINIMAX-Algorithmus anders als im klassischen Fall: Während normalerweise ein Spielzug einer Veränderung des Spielzustandes entspricht, lässt hier die Entscheidung eines Spielers für eine bestimmte Handlung den Weltzustand zunächst unverändert: Erst, wenn beide Spieler ihre Handlungsabsichten kundgetan haben, werden im Simulationsschritt deren Auswirkungen bestimmt. Ein „Zug“ entspricht in diesem Fall also eher einem Kommando als einer tatsächlichen Handlung.

Während MAX im Verlaufe der Vorwärtssimulation die Aufgabe zukommt, das Spiel konstruktiv zu seinen Gunsten zu gestalten, muss OPP nur versuchen, MAX hierbei zu behindern. Diese Asymmetrie in den Aufgaben spiegelt sich auch in einer Asymmetrie in den Möglichkeiten wieder: OPP beschränkt sich darauf, seine Figuren so zu verschieben, dass der Ball mit größtmöglicher Wahrscheinlichkeit abgefangen wird; MAX hingegen muss mit seinen Entscheidungen nicht nur ein Erreichen des Balls durch seine Figuren gewährleisten, sondern auch dafür Sorge tragen, dass der Ball in einer Art und

Weise behandelt wird, die den Spielverlauf in eine für Max günstige Richtung weist. Aus diesem Grund besteht ein Zug von Max aus einem 4-Tupel $\langle a_0, a_1, a_2, a_3 \rangle \in \mathcal{A}^4$ von Aktionen. Diese Aktionstupel, die jeder Stange eine Aktion zuweisen, werden im folgenden als Aktionszuweisungen bezeichnet.

Opp hingegen gibt jeder seiner Stangen einen direkten Steuerbefehl.

7.2 Implementierung der Aktionsauswahl

Um die in der Zielsetzung geforderte Erweiterbarkeit und Wartbarkeit zu erreichen, wurde die Aktionsauswahl in verschiedene Komponenten gegliedert: Neben dem immergleichen Aufbau des Spielbaumes, wobei sich „Aufbau“ in diesem Fall tatsächlich nur auf die Struktur des Baumes bezieht, werden die Züge von Opp und Max sowie die Bewertung der Weltzustände in isolierten Modulen vorgenommen, die sich auch während des Spieles auswechseln lassen.

Diese Austauschbarkeit ist durch die Definition von Schnittstellen für die einzelnen Komponenten gewährleistet:

Aktionserzeugung: Diese entspricht der Erzeugung der Züge von Max. Die Eingabe besteht aus einem Weltzustand; Ausgabe ist eine Menge von Aktionszuweisungen.

Gegnermodell: als Reaktion auf einen gegebenen Weltzustand liefert dieses Modul eine Menge von Steuerbefehlen für die Stangen von Opp.

Bewertungsfunktion: Die heuristische Bewertungsfunktion, die für einen gegebenen Weltzustand die Abschätzung dessen Nutzens für Max zurückliefert.

Vorhersagekomponente: Dieses Modul leistet die Hauptarbeit im Verlauf der Vorwärtssimulation, indem es die Auswirkungen der Züge von Opp und Max abschätzt. Als Eingabe für dieses Modul dienen die Züge sowie der Weltzustand, in dem diese angewendet werden sollen; zurückgeliefert wird eine Menge von möglichen Nachfolgezuständen. Weiterhin modifiziert die Vorhersagekomponente einige zur Steuerung des Baumaufbaues verwendete Größen.

Das Zusammenspiel der Komponenten ist in Abbildung 7.1 dargestellt: Über einen in den MAX-Knoten erfolgenden Aufruf der Aktionserzeugung werden die Züge von MAX generiert und fließen in die OPP-Knoten ein; von dort aus erzeugt das Gegnermodell unter Hinzufügung der Steuerbefehle von OPP die Simulationsknoten. Im nächsten Schritt werden in der Vorhersagekomponente unter Verwendung der angesammelten Informationen über den Weltzustand sowie die Züge beider Gegner mögliche Nachfolgeweltzustände erzeugt, mit denen eine neue Schicht von MAX-Knoten initialisiert wird. Ist ein MAX-Knoten als Blatt markiert, so ruft er seine eigene Bewertung durch die Evaluierungsfunktion auf.

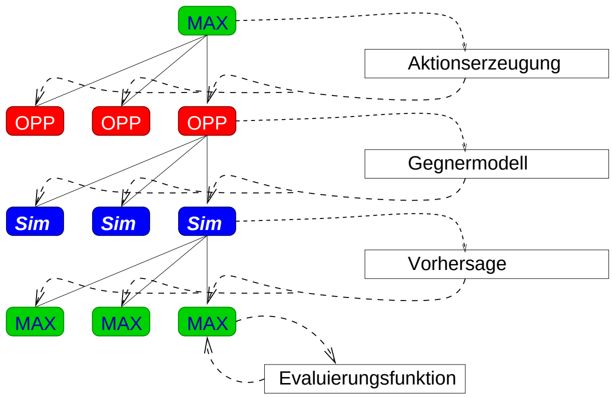


Abbildung 7.1: Der Gesamtaufbau von MITRA

7.3 Der Aufbau des Spielbaumes

Der Aufbau des Spielbaumes geschieht durch Tiefensuche. Diese ist in der vorliegenden Implementierung rekursiv realisiert; der durch einen Aufruf entstehende Baum wird im Gegensatz zum normalen Vorgehen nicht nur implizit, sondern tatsächlich erzeugt. Der Grund für die Auswahl dieser bei äquivalenten Ergebnissen langsameren Alternative liegt darin, dass eine so erzeugte

Datenstruktur sich deutlich besser analysieren lässt und das Funktionieren der einzelnen Komponenten leichter überprüfbar ist.

Im Spielbaum kommen Opp-, Max- und Simulationsknoten vor. Für die Terminalknoten wurde in der Implementierung kein eigener Knotentyp eingeführt. Der Aufruf der Bewertung der entstandenen Zustände erfolgt gegebenenfalls durch die Max-Knoten.

Unabhängig vom Typ enthält jeder Knoten k folgende Daten:

- `minimaxwert` $\in \mathbb{R}$: Der Minimax-Wert für k.
- `baumgröße` $\in \mathbb{N}$: Die Größe des Teilbaumes, dessen Wurzel k ist. Diese Information wird nur zu Analysezwecken verwendet.
- `vorhersagezeit` $\in \mathbb{R}$: Gibt in Sekunden an, wie weit die Vorwärtssimulation dem tatsächlichen Weltzustand in k bereits vorgegriffen hat. Da mit zunehmendem zeitlichem Abstand die Genauigkeit der Vorwärtssimulation drastisch abnimmt, führt die Überschreitung eines gewissen Schwellwerts (derzeit 5 Sekunden) durch diesen Wert zu einem Aufruf der Bewertungsfunktion und somit zu einem Abbruch der Baumkonstruktion unterhalb von k.
- `maxVorhersagen` $\in \mathbb{N}$: In jedem Simulationsknoten wird dieser Wert um mindestens eins verringert; ansonsten erben die Knoten diesen Wert von ihren Vorgängern. Max-Knoten, in denen `maxVorhersagen` einen negativen Wert hat, bilden die Blätter des Baumes. Durch diesen Wert ist also die Größe des Spielbaumes nach oben beschränkt.
- `weltzustand` $\in \mathcal{S}$: Der k zugeordnete Weltzustand.
- `aktionen` $\in \mathcal{A}^4$: Eine Aktionszuweisung; im Falle des Max-Knoten handelt es sich hierbei um die vor `weltzustand` aktiv gewesenen Aktionen, ansonsten um die in diesem anzuwendenden.

Weiterhin enthält jeder Knoten neben einem Zeiger auf den jeweiligen Vorgänger sowie einer Liste von Zeigern auf die Nachfolger Informationen darüber, durch welche Module seine Nachfolger erzeugt bzw. bewertet werden müssen.

Die Unterschiede zwischen den verschiedenen Knotentypen zeigen sich vor allem in der Erzeugung der Nachfolger und der Errechnung des Wertes. Letztere geschieht bei den MAX-Knoten in der von EXPECTIMINIMAX gewohnten Form. Simulationsknoten und OPP-Knoten werden wie Würfelknoten gehandhabt.

Die weiteren Unterschiede sind:

Max-Knoten: Die Erzeugung der Nachfolgerknoten geschieht über einen Aufruf der Aktionserzeugung. Von den anderen Knoten unterschiedet den MAX-Knoten die Möglichkeit, sich selbst als ein Blatt des Spielbaumes zu erkennen und die Bewertungsfunktion aufzurufen. Weiterhin gestattet der MAX-Knoten den Zugriff auf die Aktionszuweisung des höchstbewerteten Nachfolgers.

Opp-Knoten: Dieser ruft das **Gegner**-Modul auf, um unter Verwendung der Reaktionsmöglichkeiten von OPP die nächste Schicht von Knoten zu erzeugen.

Simulationsknoten: Der Simulationsknoten enthält alle zum Aufruf der **Vorhersage** notwendigen Informationen; durch diesen Aufruf wird die nächste Generation von MAX-Knoten erzeugt.

7.4 Die Aktionserzeugung

Die Aktionserzeugung hat die Aufgabe, zu einem gegebenen Weltzustand $s \in \mathcal{S}$ eine Menge von geeigneten Aktionszuweisungen $\mathcal{Z} = \{Z_0, \cdots, Z_i\}$ zu erzeugen. Die Einbindung in die Konstruktion des Spielbaumes ist in Abbildung 7.2 dargestellt: Die Aktionserzeugung erhält als Eingabe einen im aufrufenden MAX-Knoten gespeicherten Weltzustand, unter dessen Verwendung sie dann eine Menge von Aktionszuweisungen zur Erstellung der nächsten Ebene von OPP-Knoten erzeugt. Die Aktionszuweisungen sind in der Darstellung durch die Anfangsbuchstaben der Aktionen repräsentiert; *CSBB* steht also für eine Aktionszuweisung der Form ⟨**ClearBall, StopBall, BlockBall, BlockBall**⟩.

$\mathcal{Z}$ soll folgenden Ansprüchen genügen:

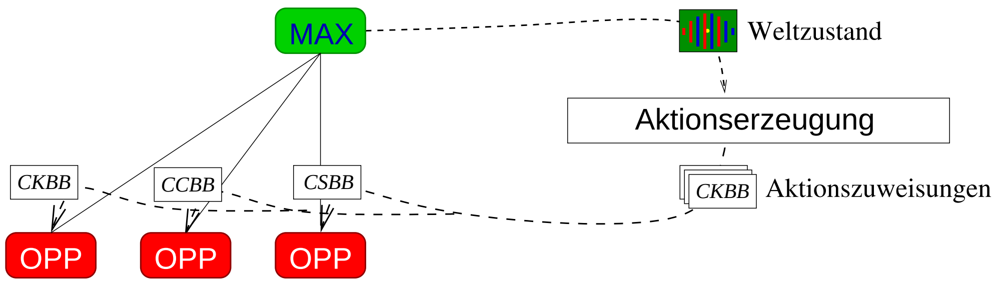


Abbildung 7.2: Funktionsweise der Aktionserzeugung

Erschöpfung: Jede Aktionszuweisung, deren Anwendung in s nützlich zu sein scheint, soll in $\mathcal{Z}$ enthalten sein.

Minimalität: Um den Verzweigungsfaktor so gering wie möglich zu halten, sollen sich die Konsequenzen der Aktionszuweisungen in $\mathcal{Z}$ paarweise deutlich unterscheiden.

Die Erfüllung dieser Kriterien kann selbstverständlich nur intuitiv beurteilt werden, da der einzige Weg, jede mögliche Aktionszuweisung automatisch auf ihre Eignung für den Einsatz in einer gegebenen Situation zu überprüfen, im Ausprobieren aller dieser Zuweisungen besteht. Dies allerdings soll durch die Forderung nach Minimalität ja gerade unterbunden werden.

Die Aktionserzeugung hat also die Aufgabe, aus der Menge aller möglichen Aktionszuweisungen die relevanten herauszufiltern. Allerdings darf nicht vergessen werden, dass der Spielbaum nur im offensiven Fall bei der Auswahl der richtigen Aktionen hilft. In der Defensive darf die Aktionserzeugung nur eine einzige Aktionszuweisung generieren. Die Eignung dieser Zuweisung kann nicht aus dem Spiel begründet werden, es handelt sich hierbei also um durch den Programmierer eingebrachtes Weltwissen.

Falls *KiRo* in der Offensive ist, kann dennoch maximal eine Stange direkt auf den Ball einwirken. Allerdings bestünde die Möglichkeit, durch geeignetes Positionieren einer anderen Stange (beispielsweise über eine Art von Kommunikation zwischen den Aktionen) den Wissensvorsprung, was in nächster Zukunft mit dem Ball geschehen wird, zu nutzen. Auf diese Möglichkeit wurde aus Gründen der Vereinfachung in der vorliegenden Implementierung verzichtet. Wenn das einzige Augenmerk auf der Einwirkung auf den Ball liegt, so unterscheiden sich alle Aktionszuweisungen in $\mathcal{Z}$ mindestens an dieser Stange.

Da die anderen Aktionen weitgehend irrelevant sind, haben wir uns, um ein ruhiges Spiel zu ermöglichen, dafür entschieden, allen nicht den Ball spielenden Aktionen die **ClearBall**-Aktion zuzuweisen, falls sie sich vor dem Ball befinden; andernfalls fällt die Entscheidung für die **BlockBall**-Aktion.

Alle Aktionen in $\mathcal{Z}$ unterscheiden sich also nur durch die Aktion, die an der ballführenden Stange anliegt. Für diese Aktion gibt es folgende Alternativen:

- **BlockBall:** Der Ball soll auf keinen Fall die Stange passieren. Diese Zuweisung ist auch die einzige, die in der Defensive erzeugt wird.
- **ClearBall:** Der Ball soll auf jeden Fall die Stange passieren.
- **KickBall:** Die geradeaus schießende **KickBall**-Aktion wird auf jeden Fall erzeugt; falls sich der Ball nicht zu schnell seitwärts bewegt, werden auch Seitwärtsschüsse in beide Richtungen angeboten. „Zu schnell“ bedeutet in diesem Fall, dass sich der Ball zwischen zwei aufeinanderfolgenden Bildern der Kamera um mehr als seinen eigenen Radius bewegt. Die Wahrscheinlichkeit, einen erfolgreichen schrägen Schuss durchzuführen, ist in diesem Fall äußerst gering.
- **StopBall:** Falls der Ball sich bewegt, wird auch eine Aktionszuweisung mit **StopBall**-Aktion erzeugt.

7.5 Die Gegnermodelle

Abbildung 7.3 demonstriert die Rolle, die den Gegnermodellen im Aufbau des Spielbaumes zukommt: Aufgerufen durch einen OPP-Knoten, erstellen sie einen oder mehrere Züge von OPP, wobei einer dieser Züge aus Steuerbefehlen für jede der Stangen sowie der Wahrscheinlichkeit, dass der Gegner genau diese Befehle anlegt, besteht. Aus den im aufrufenden OPP-Knoten vorhandenen Informationen und jeweils einem dieser Züge wird dann ein neuer Simulationsknoten erzeugt.

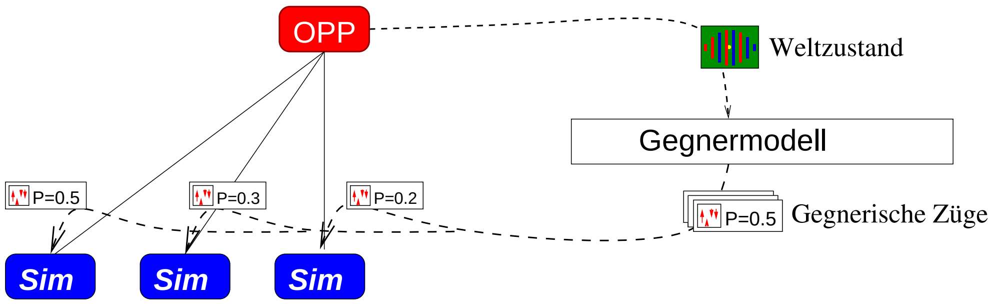


Abbildung 7.3: Einbindung des Gegnermodells

Wie bereits oben erwähnt entsprechen die Züge der Gegner mit Wahrscheinlichkeiten gewichteten Steuerbefehlen; diese werden durch die Vorwärtssimulation identisch zu den jeweiligen Kommandos der Aktionen interpretiert. Eine nicht notwendige, in der derzeitigen Fassung des Programms aber aus Vereinfachungsgründen getroffene Vereinfachung ist die in Abschnitt 5.1.2 bereits erwähnte Annahme der körperlichen Gleichwertigkeit: Dem Gegner stehen also tatsächlich genau die Geschwindigkeits- und Beschleunigungsoptionen zur Verfügung, die auch die eigenen Aktionen einsetzen können. Allerdings wird bei der Gegnermodellierung darauf verzichtet, die Rotation der Stangen zu berücksichtigen; es wird davon ausgegangen, dass der Gegner immer in der Lage ist, die Füße seiner Figuren in eine angebrachte Stellung zu rotieren, falls er es schafft, die Stangen dergestalt zu verschieben, dass der Kontakt zum Ball nur noch vom Winkel, in dem die Figuren stehen, abhängt.

Wie in 4.3.2 ausgeführt kann das Gegnermodell eine beliebige Anzahl von verschiedenen Zügen erzeugen, wobei allerdings die Summe der Wahrscheinlichkeiten, die den einzelnen Zügen zugeordnet werden, eins ergeben muss.

In der momentanen Version stehen drei verschiedene Gegnermodelle zur Auswahl; allen gemeinsam ist, dass sie nur sehr wenige Züge erzeugen.

Der statische Gegner: Dieser lässt seine Figuren immer dort stehen, wo sie momentan sind; dies geschieht mit der Wahrscheinlichkeit eins.

Der blockierende Gegner: Dieser Gegner legt mit hundertprozentiger Wahrscheinlichkeit dasselbe Verhalten an den Tag wie das MITRA, wenn sie mit jeder Stange zu blockieren versucht – tatsächlich besteht die Implementierung dieses Gegnermodells aus vier **BlockBall**-Aktionen, die als Eingabe gespiegelte Weltmodelle erhalten und deren Steuerbefehle wiederum gespiegelt werden, bevor sie in die Simulation einfließen.

Diese beiden Gegnermodelle sind für sich genommen zu einfach aufgebaut: Während der statische Gegner unrealistisch schwach ist – und die Ausgaben des Spielbaumes dementsprechend durch „Schiessen aus allen Lagen" beschrieben werden können – zeigt sich, dass der blockierende Gegner in der fehlerfreien Welt der Simulation zu gut funktioniert und die allermeisten Bälle abfängt. Aus diesem Grund versucht MITRA bei der Verwendung dieses Gegnermodells, den Ball zu halten. Auch dies kann nicht als wünschenswertes Verhalten angesehen werden. Da sich also die Stärken und Schwächen der beiden Modelle komplementieren, wurden diese zusammengefügt zum

Fifty-Fifty-Gegner: Dieser entscheidet sich in jedem Zug mit fünfzigprozentiger Wahrscheinlichkeit für die Untätigkeit und mit demselben Wert für blockierendes Verhalten. Das Spielverhalten ist bei der Verwendung dieses Gegnermodells das ansprechendste; allerdings wird dies dadurch erkauft, dass der Verzweigungsfaktor des Spielbaumes im Vergleich mit den einfacheren Gegnermodellen verdoppelt wird; die bei gleicher Beanspruchung der Ressourcen erreichbare Suchtiefe sinkt also beträchtlich.

7.6 Die Vorhersagekomponente

Aufbauend auf den in einem Simulationsknoten angesammelten Informationen erzeugt die Vorhersagekomponente eine Menge neuer Weltzustände, durch die eine neue Ebene von Max-Knoten erzeugt und somit eine neue Runde in der Konstruktion des Baumes eingeläutet wird. Abbildung 7.4 stellt dies graphisch dar: Die Eingabe für die Vorhersagekomponente besteht aus einem Weltzustand, sowie den für diesen in den weiter oben erläuterten Komponenten erzeugten Zügen von Max und Opp. Als Ausgabe liefert die

Vorhersagekomponente eine Menge von mit ihren jeweiligen Wahrscheinlichkeiten gewichteten Weltzuständen, durch die jeweils ein neuer MAX-Knoten initialisiert wird.

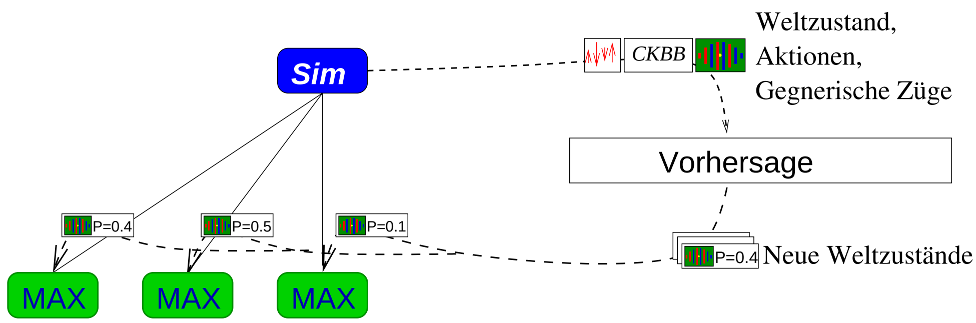


Abbildung 7.4: Einbindung der Vorhersagekomponente

Der Vorhersagekomponente kommt die essentielle Aufgabe zu, aus den gegebenen Informationen über den aktuellen Spielzustand s_i sowie über die Züge beider Parteien die nachfolgenden Spielzustände $s_{i+1_1}, \cdots s_{i+1_n}$ zu errechnen.

Wie bereits weiter oben erwähnt kommt der Dauer $\Delta_k t = s_{i+1_k}.\texttt{zeitpunkt} - s_i.\texttt{zeitpunkt}$ der in der Simulation vorweggenommenen Zeiträume eine zentrale Bedeutung zu: Da sich die Spieler während des Vorhersageschritts nicht für eine andere Verhaltensweise entscheiden können, ist es wünschenswert, die $\Delta_k t$ so klein wie möglich zu halten, um den am realen Tisch herrschenden Verhältnissen möglichst nahe zu kommen – in der Realität haben ja beide Spieler zu jedem Zeitpunkt die Möglichkeit, sich umzuentscheiden.

Kleine Werte für $\Delta_k t$ aber führen zu einem anderen Problem: Da die Baumkonstruktion inklusive aller Vorhersageschritte in einem sehr kurzen Zeitraum – bei der vorliegenden Implementierung 20 ms – vollendet werden muss, der Aufwand hierzu aber exponentiell in der Baumtiefe steigt, muss diese so gering wie möglich gehalten werden. Die Anzahl der zur Verfügung stehenden Vorhersageschritte ist also begrenzt. Überbrücken diese nun zusätzlich nur einen sehr kleinen Zeitraum, so ist es nicht möglich, in der zur Verfügung stehenden Zeit einen Spielbaum zu konstruieren, der die zukünftigen Entwicklungen des Spieles genügend weit berücksichtigt. Die Aussagekraft eines solchen Baumes und damit seine Verwendbarkeit zur Aktionsauswahl sind somit minimal.

Es existiert hier also ein Zielkonflikt: Um eine größtmögliche Realitätstreue zu erreichen sind kleine Werte für $\Delta_k t$ wünschenswert, Erwägungen über die Einsetzbarkeit des Spielbaumes fordern aber, diese Zeitspanne so groß wie möglich zu gestalten. Eine Möglichkeit hierzu liegt in der – begrenzten – Voraussicht der Aktionen: Die einzelnen Aktionen haben, ebenso wie der Gegner, ein Modell von den Vorgängen in ihrer Umgebung; eine **Kick-Ball**-Aktion beispielsweise besteht ja nicht nur im Vorschnellen des Fußes der entsprechenden Spielfigur, sondern übernimmt davor auch noch die Positionierung anhand der im Weltmodell gegebenen Werte über den Ort und die Bewegung des Balls. Jede Aktion a_i trägt also eine rudimentäre Vorhersagekomponente mit sich; sie setzt zu jedem Zeitpunkt t_0 die Steuerbefehle, die notwendig sind, um zu einem späteren Zeitpunkt t_{a_i} richtig für die Ausführung des den Ball beeinflussenden Eingriffes positioniert zu sein; weiterhin kennt jede Aktion den Zeitpunkt t_{a_i}. Analog hierzu ist die Situation des Gegners: Auch bei diesem ist für jede Stange i der Zeitpunkt t_{o_i} bekannt, zu dem sie den Ball erreichen kann, und welche Steuerbefehle der Gegner bei dieser anlegt, um seine Figuren rechtzeitig auf den entsprechenden Koordinaten positioniert zu haben. Unter der Bedingung, dass sich keiner der beiden Gegner für eine andere Aktion entscheidet und zudem sich die Welt auch genau so entwickelt, wie es die rudimentären Vorhersagekomponenten prophezeien, kann also gefahrlos bis zum Zeitpunkt t_{min} vorwärtssimuliert werden, für den gilt: $t_{min} = \min\{t_{a_i} | i \in [0, \cdots, 3]\} \cup \{t_{o_i} | i \in [0, \cdots, 3]\}$. Zu diesem Zeitpunkt muss dann überprüft werden, ob entweder Opp bei dem Versuch, den Ball abzufangen, Erfolg beschieden sein wird, oder aber Max in der Lage ist, eine seiner Aktionen auszuführen. Im ersten Fall muss der entsprechende Ast des Spielbaumes mit einem Aufruf der Bewertungsfunktion abgeschnitten werden, im anderen ist die Ausführung der betreffenden Aktion zu simulieren.

Die Frage ist nun: unter welchen Bedingungen sind die für obiges Vorgehen angegebenen Voraussetzungen nicht erfüllt, wann also muss der Zeitpunkt t_{sim}, bis zu dem simuliert wird, kleiner als t_{min} gewählt werden, um den Gegnern die Möglichkeit zu geben, sich für neue Eingriffsoptionen zu entscheiden? Allgemein lässt sich diese Frage folgendermassen beantworten: Die Simulation muss zu jedem Zeitpunkt unterbrochen werden, zu dem der weitere Verlauf des Spieles von einer der beiden Parteien nicht mehr vorherzusehen ist. Die Situationen, in denen dies der Fall ist, sind:

- Erzielte Tore
- Die Ausführung einer den Lauf des Balls verändernden Aktion
- Die Veränderung der Aufstellung der Figuren der einzelnen Spieler

Problematisch ist hierbei der letzte Fall: Beide Spieler verschieben ihr Stangen ständig, und somit kann zu jedem Zeitpunkt eine Aktion in ihrer Bewertung plötzlich schwanken. Wir haben uns allerdings entschlossen, den letzten Punkt zu ignorieren und uns bei den Kriterien zur Auswahl des Endzeitpunktes eines Vorhersageschrittes vollkommen auf den Ball zu konzentrieren; erst diese Einschränkung in der Realitätstreue der Vorhersagekomponente macht den Weg frei für die Erzielung von verwendbaren $\Delta_k t$.

Weiterhin haben wir uns dafür entschlossen, ein weiteres Kriterium zur Einleitung einer neuen Runde von Aktionserzeugungen einzuleiten, nämlich die Reflexion an einer der Stirnwände. Dies geschieht aus zwei Gründen: Zum einen vereinfacht dieser Schritt die Verwirklichung des ersten Kriteriums, des erzielten Tors, zum anderen entspricht diese Reflexion auch immer einer Umkehrung der Spielrichtung und somit einer Veränderung der Gesamtsituation.

Der Ablauf eines Simulationsschritts besteht also aus mehreren Stufen: Zunächst wird der Zeitpunkt t_{sim}, bis zu dem gefahrlos vorwärtssimuliert werden kann, bestimmt; daraufhin erfolgt die Simulation bis zu t_{sim}. Der Weltzustand zu t_{sim} ist definitionsgemäß der erste, zu dem der Lauf des Balls auf unvorhersagbare Weise verändert werden kann; anhand dieses Weltzustandes wird also beurteilt, ob Eingriffe in den Lauf des Balls stattfinden können, und, falls ja, was ihre Konsequenzen sind. Falls mehrere Nachfolgeweltzustände erzeugt werden müssen, so ist dieser Weltzustand zu t_{sim} derjenige, an dem die Aufspaltung stattfindet. Nach dieser Beurteilung und eventuellen Aufspaltung ist es unter Umständen notwendig, einen weiteren kurzen Simulationsschritt einzulegen; diese sogenannte „Schrecksekunde“ wird weiter unten erläutert.

Nach Absolvierung dieser Schritte (oder durch ein währenddessen erkanntes zwingendes Abbruchkriterium) werden in jedem Fall ein oder mehrere neue mit ihrer Wahrscheinlichkeit gewichtete MAX-Knoten eingefügt; in diesen wird entweder eine neue Runde von Aktionserzeugungen eingeleitet, oder, falls der Knoten durch einen negativen Wert von `maxVorhersagen` als Blatt markiert ist, durch einen Aufruf der Evaluierungsfunktion der entsprechende Ast abgeschnitten.

7.6.1 Errechnen der Vorhersagezeit

Zur Bestimmung des Zeitpunktes t_{sim}, bis zu dem vorwärtssimuliert werden kann, wird zunächst untersucht, ob es einen aktiven Spieler gibt, und, falls dem so ist, um welchen es sich handelt.

Gibt es keinen aktiven Spieler, so liegt einer von zwei Fällen vor: Entweder bewegt sich der Ball auf eine der Stirnwände zu und ist von keiner Stange mehr erreichbar, oder er bewegt sich überhaupt nicht in X-Richtung und befindet sich außerhalb der Reichweite aller Figuren. In letzterem Fall ist das Weiterverfolgen der Vorwärtssimulation sinnlos. Es wird also ein als Blatt markierter MAX-Knoten erzeugt, dem die Wahrscheinlichkeit 1,0 zugewiesen wird. Im ersten Fall wird bestimmt, wann der Ball die Stirnwand erreicht; dieser Zeitpunkt wird dann als t_{sim} gewählt.

Handelt es sich beim Gegner um den aktiven Spieler, so wird berechnet, wann dieser den Ball erreichen kann. Diese Berechnung wird rein anhand von externen Kriterien, also der Bewegung des Balls und der x-Koordinate der gegnerischen Figuren, vorgenommen. Hierdurch unterscheidet sie sich von der Bestimmung von des entsprechenden Zeitpunktes für MAX: In diesem Fall entspricht t_{min} der Abschätzung der ballführenden Stange, wann sie in das Spielgeschehen eingreifen wird.

Der Grund für diese unterschiedliche Behandlung der beiden Spieler liegt darin, dass der Gegner den Ball nur erreichen muss, und davon auszugehen ist, dass er dies nicht zu einem anderen als dem frühestmöglichen Zeitpunkt versucht; MAX hingegen kann absichtlich seinen Eingriff in das Spielgeschehen verzögern, um beispielsweise in eine vorteilhaftere Schussposition zu gelangen. Um diese Möglichkeit in der Simulation nicht zu verbauen, wurde dieser asymmetrischen Zeitberechnung der Vorzug vor einer Gleichbehandlung beider Parteien gegeben.

Weiterhin wird, falls MAX am Zuge ist, die ballführende Aktion abgefragt, wie groß die Wahrscheinlichkeit ist, in der gegebenen Situation erfolgreich ins

Spiel einzugreifen; in welcher Form diese Abschätzung vor sich geht, wurde in Kapitel 6.2 erläutert. Das Ergebnis dieser Abfrage wird im weiteren Ablauf des Vorhersageschrittes verwendet.

7.6.2 Simulationsschritt

Der eigentliche Simulationsschritt ist sehr einfach aufgebaut: Der Ball und die Spielfiguren werden um die im vorigen Schritt berechnete Zeitspanne weiterbewegt. Die Vorhersage der Ballbewegung verläuft dabei nach den in Kapitel 5 angegebenen Prinzipien, die Bewegung der Figuren beider Spieler werden unter Verwendung eines Modells der Semantik der Steuerbefehle abgeschätzt.

7.6.3 Beurteilung der entstandenen Spielsituation

Je nachdem, ob das Ende des Simulationsschritts durch einen Kontakt mit einer Stirnwand, einem Eingriff von Opp oder von Max gegeben ist, werden die Konsequenzen der entstandenen Spielsituation berechnet:

- Der Ball wird an einer Stirnwand reflektiert: Hierbei gibt es zwei Alternativen, nämlich die, dass die „Reflexion“ zwischen den Torpfosten stattfindet – und somit die Bewertungsfunktion aufgerufen werden kann – oder aber der Ball tatsächlich gegen die Stirnwand prallt und reflektiert wird. Diese Reflexion wird einfach durch einen Vorzeichenwechsel der Ballgeschwindigkeit in X-Richtung modelliert. Diese sehr einfache Realisierung weist zwei Ungenauigkeiten auf: Der Einfallswinkel ist in der Realität leider nicht immer gleich dem Ausfallswinkel, weiterhin besteht in der wirklichen Welt die Möglichkeit, dass der Ball durch den Torpfosten schräg reflektiert wird.

- Opp hat die Möglichkeit, den Ball zu erreichen: Falls es Opp gelungen ist, eine seiner Figuren ausreichend nahe zum Ball zu bewegen, so wird der Simulationsschritt beendet und ein als Blatt markierter und der Wahrscheinlichkeit eins gewichteter Max-Knoten als Nachfolger erzeugt; ansonsten bleiben die Werte des Balls unverändert.

- MAX kann auf den Ball einwirken: Es ist dies der bei weitem komplexeste Fall, und derjenige, dessen Implementierung am stärksten vereinfacht ist. Falls die oben festgestellte Erfolgswahrscheinlichkeit der ballführenden Aktion im gegebenen Weltzustand größer als Null ist, so wird ein mit dieser gewichteter Nachfolgeweltzustand erzeugt, in dem die Ballbewegung entsprechend dem Charakter der ausgeführten Aktion verändert wird. Beim derzeitigen Aktionensatz bedeutet dies: Nach einem Schuss bewegt sich der Ball schnell in die Schussrichtung; **BlockBall** reflektiert den Ball, und **StopBall** bringt ihn zum Stillstand.

Während die erfolgreichen Fälle einfach zu behandeln sind, ist das Vorgehen bei Erfolglosigkeit deutlich schwerer handhabbar. In der vorliegenden Implementierung wird davon ausgegangen, dass der Ball bei Nichterfolg der Aktion seine Bahn unverändert weiterverfolgt, es wird also ein mit der Wahrscheinlichkeit der Erfolglosigkeit gewichteter Nachfolgeweltzustand erzeugt, in dem der Ball die Stange ungestört passiert. Von dieser Annahme kann aber unter realistischen Bedingungen nur dann mit Sicherheit ausgegangen werden, wenn die Aktion nicht einmal aktiv wurde.

Deshalb wurde ursprünglich ein Ansatz verfolgt, der den Ball bei Nichtaktivität weiterlaufen ließ und bei Fehlschlag die Ballbewegung unter Einbeziehung der Geschwindigkeit von Ball und spielender Figur zufällig verändert wurde. Es stellte sich aber heraus, dass der Ball bei vielen Fehlschlägen, obwohl die Aktion aktiv wurde, ungestört weiterlief; da dies in der Vorhersage nicht adäquat modelliert wurde, wurde zunächst die **KickBall**-Aktion als sichere Abwehrmethode für den Torwart angesehen, da der Ball sich bei Fehlschlägen zwar nicht in Richtung des gegnerischen Tors bewegte, aber doch fast immer vom eigenen Tor abgelenkt zu werden und sich meistens noch im Bereich der eigenen Stangen zu befinden schien. Tatsächlich waren die Abwehrversuche mittels **KickBall** eine unerschöpfliche Quelle von Gegentoren.

Die derzeit verwendete Realisierung weist diesen Fehler nicht auf. Sie ist dennoch weit von der Perfektion entfernt, wie folgendes Beispiel demonstriert:

Passiert der Ball die eigenen Verteidigerstange in Richtung auf das gegnerische Tor, so scheint die schlimmste Konsequenz einer fehlgeschlagenen **StopBall**-Aktion die zu sein, dass der generische Stürmer in Ballbesitz gelangt.

Im Spiel hingegen stellte sich immer wieder heraus, dass missratene Stopversuche zu Schüssen auf das eigene Tor missraten (Zur Erinnerung: Um den von hinten nach vorne rollenden Ball zu stoppen, lässt die Aktion ihn erst unter der Stange passieren, um ihn dann mit einer Drehung der Spielfiguren gegen die Laufrichtung des Balls einzuklemmen. Kritisch ist hierbei das Timing der Drehung: erfolgt diese zu früh, so wird der Ball nicht eingeklemmt, sondern gegen seine ursprünglichen Laufrichtung zurückgeschlagen). Diese Schüsse sind spektakulär erfolgreich.

Trotz der erkannten Schwäche des aktuellen Ansatzes wurde er nicht weiter verbessert: Das Auftreten von Fehlentscheidungen ist hinnehmbar selten, so selten, dass die weitere Verzweigung des Baumes und dadurch erfolgende Effizienzminderung nicht gerechtfertigt erscheinen.

7.6.4 Schrecksekunde

Falls Max den Ball gespielt hat, so wird der Spielverlauf mit den geänderten Ballparametern noch um einen kurzen Zeitraum weitersimuliert. Dies geschieht aus zwei Gründen:

1. In der Realität gibt es eine nicht zu vermeidende Reaktionszeit oder Schrecksekunde, die sowohl beim Menschen wie auch beim Computer notwendig ist, um die neue Bewegungsrichtung des Balls zu erkennen. Siehe hierzu auch Abbildung 6.3. Es ist also unrealistisch, anzunehmen, dass beiden Spielern direkt nach dem Eingriff in den Lauf des Balls bereits bekannt ist, was für Konsequenzen dieser hatte.

2. Der zweite Grund liegt im Aufbau der Vorwärtssimulation: Da im vorigen Schritt nur die Bewegung des Balls, nicht aber sein Ort modifiziert wurde, würde die entsprechende Aktion im nächsten Vorhersageschritt den Ball als sofort spielbar erkennen und durch diese Einschätzung verhindern, dass weitersimuliert werden kann.

7.7 Die Evaluierungsfunktion

Darstellung 7.5 zeigt die Einbindung der Evaluierungsfunktion: Ein als Blatt markierter MAX-Knoten ruft die Funktion mit dem im Knoten gespeicherten Weltzustand als Parameter auf und bekommt einen Zahlwert zurückgeliefert, der dem Wert des Knotens entspricht.

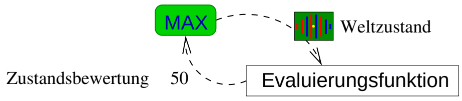


Abbildung 7.5: Einbindung der Bewertungsfunktion

Die verwendete Evaluierungsfunktion weist einem Weltzustand $s \in \mathcal{S}$ einen Wert zwischen 0 und 100 zu, wobei ein erzieltes Tor den Wert 100 erhält, ein Gegentor den Wert 0. Das Erkennen eines Tors kann auf zwei Wegen geschehen: Falls der Wert von $s.\texttt{zustand}$ ein Tor oder Gegentor ausweist, oder aber der Ball von keiner Stange mehr erreicht werden kann und sich auf dem Weg zu einem Tor befindet, wird auf ein erzieltes Tor erkannt.

Falls der Ball in s weder bereits im Tor noch unaufhaltbar auf dem Weg dorthin ist, so muss der Wert von s heuristisch abgeschätzt werden. Das Vorgehen hierzu fußt auf diesen Beurteilungen:

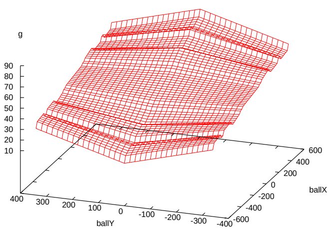
- Je größer der Wert von $s.\texttt{ballX}$ ist, je weiter der Ball also vom eigenen Tor entfernt ist, desto besser;
- Je größer $|s.\texttt{ballX}|$ ist, je geringer also die Distanz des Balls zu einer der Stirnwände ist, desto wichtiger werden der Wert von $s.\texttt{ballY}$ und die Identität des ballführenden Spielers für die Bewertung von s.

Die hier gewählte Abschätzungsfunktion g hat folgende Form:

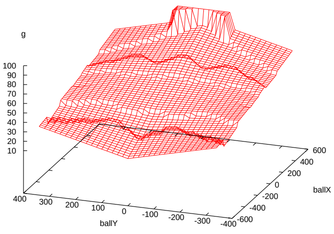
$$g(s) = A + B \cdot \frac{s.\texttt{ballX} - c \cdot v \cdot |s.\texttt{ballY}|}{0.5 \cdot h_f} + C \cdot c \cdot k \tag{7.1}$$

A bezeichnet den Wert, der einem unbewegt in der Mitte des Spielfeldes liegendem Ball zugewiesen wird; in der vorliegenden Version beträgt dieser 50.

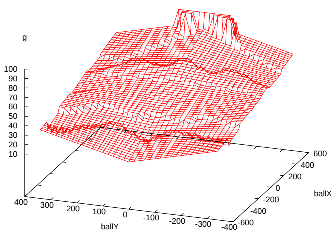
Der ruhig in der Mitte liegende Ball ist der Fußpunkt der Abschätzungsfunktion. B ist ein Faktor, der den Einfluss des Ortes des Balls auf den Wert des Spielzustandes gewichtet; dieser ist momentan auf 40 festgesetzt. C bezeichnet das Gewicht, das die Identität des ballführenden Spielers hat; der Wert von C beträgt momentan 5. Weiterhin ist h_f die Spielfeldlänge in X-Richtung, $h_f \cdot 0.5$ dementsprechend der Betrag der X-Koordinaten beider Tore, v das Vorzeichen von $s.\texttt{ballX}$, für den Gewichtungsfaktor c gilt $c = \frac{|s.\texttt{ballX}|}{h_f \cdot 0.5}$, und k schließlich nimmt den Wert 1 an, wenn *KiRo* „am Zug“ ist, -1, falls dies für den Gegner gilt, und ansonsten 0.



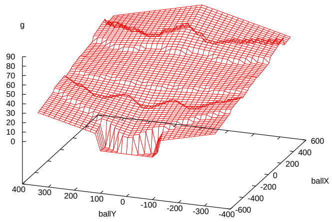
(a)



(b)



(c)



(d)

Abbildung 7.6: Demonstration der von der Evaluierungsfunktion gelieferten Werte; der Ball bewegt sich in den Darstellungen (a) gar nicht, (b) gerade und (c) schräg in positiver X-Richtung sowie (d) gerade in negativer X-Richtung

Dass diese etwas willkürlich wirkende und an unbegründeten Zahlwerten reiche Funktion tatsächlich vernünftige Werte liefert ist nicht auf den ersten

Blick zu erkennen; aus diesem Grunde finden sich in Abbildung 7.6 einige Plots dieser Funktion bei unterschiedlichen Bewegungsparametern des Balls.

Kapitel 8

Experimentelle Ergebnisse

8.1 Laufzeit des Verfahrens

Ein Kriterium für die Einsetzbarkeit von Algorithmen ist immer deren Laufzeit. Da das vorgestellte Verfahren in einem Kontext eingesetzt wird, in dem schnelle Reaktionen von höchster Bedeutung sind, gelten engste Beschränkungen für die zur Verfügung stehende Zeit. Ein Durchlauf des Hauptzyklus von *KiRo* darf in der vorliegenden Version nicht länger als 20 ms dauern; knapp unter 15 ms hiervon stehen der Aktionsauswahl maximal zur Verfügung. Auf dem als Referenz benutzten Rechner, der mit einem 1,7 GHz AMD-Prozessor ausgestattet ist, wurden folgende Laufzeiten der Aktionsauswahl festgestellt:

Anzahl Vorhersageschritte	Laufzeit
1	0,3 ms
2	10 ms
3	23 ms
4	45 ms
5	80 ms

Bei diesen Werten handelt es sich jeweils um die im schlimmsten Fall auftretenden. Da der Spielbaum nicht aufgebaut wird, wenn der Gegner am Zuge ist, ist die für den besten Fall benötigte Zeit nahe Null; der durchschnittliche Fall hängt hingegen wesentlich von der Spielweise des Gegners ab. Aus diesem Grunde wurde darauf verzichtet, die entsprechenden Werte aufzuführen.

Klar zu erkennen ist, dass die Aktionsauswahl bis zu einer Suchtiefe von zwei Vorhersageschritten – was einer Baumtiefe von sechs Knoten entspricht – problemlos mit der zur Verfügung stehenden Zeit auskommt. Ab einer Tiefe von drei Vorhersageschritten überschreitet MITRA die zur Verfügung stehende Zeit. Experimente zeigten, dass dies im Falle der Suchtiefe drei noch zu tolerieren ist; ab der Suchtiefe vier hingegen wurde das Spiel von *KiRo* durch die Zeitüberschreitungen deutlich in Mitleidenschaft gezogen.

8.2 Durchgeführte Experimente

Es wurden zwei Arten von Experimenten durchgeführt: Zum einen musste sich MITRA im Simulator mit den alten Komponenten messen, zum anderen wurden beide Ausführungen dahingehend verglichen, wieviele Tore sie am realen Tisch in einer vorgegebenen Zeitspanne schießen, wenn der Gegner seine Stangen nicht bewegt. Dies ist notwendig, um garantieren zu können, dass Unterschiede im Abschneiden der verschiedenen Versionen nicht in der unterschiedlichen Stärke der jeweiligen Gegner begründet sind.

Der direkte Vergleich zwischen verschiedenen Programmen ist derzeit nur in der Simulation möglich; da die im Simulator erzeugte Welt immer von der realen abweicht, sind die hier erzielten Ergebnisse nicht ohne weiteres in die reale Welt zu übertragen. Dennoch legt erfahrungsgemäß die im Simulator festgestellte drückende Überlegenheit eines Programms über ein anderes nah, dass ersteres sich auch am realen Tisch besser bewährt.

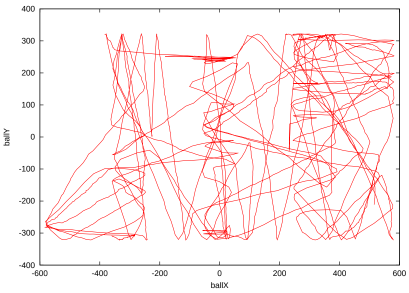
8.2.1 Ergebnisse am Simulator

Die Ergebnisse der Experimente am Simulator waren gut: Die neue Art der Aktionsauswahl beherrschte die alte deutlich.

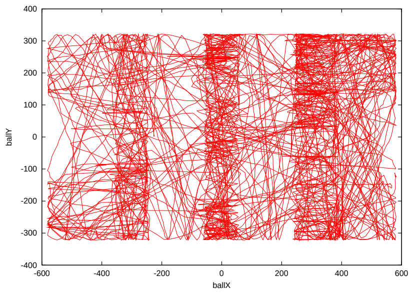
Zunächst wurde eine Serie von Partien gespielt, in denen die maximale Suchtiefe von MITRA auf zwei Vorhersageschritte gesetzt war – da jeder Vorhersageschritt drei Arten von Knoten umfasst war der entstandenen Suchbaum also maximal 6 Knoten tief. Diese Einstellung wurde bei allen Experimenten beibehalten, soweit im folgenden nicht explizit auf Ausnahmen hiervon hingewiesen wird.

Von 20 durchgespielten Partien, wobei eine Partie beendet ist, sobald eine Spielpartei neun Tore geschossen hat, gewann MITRA alle; die alte Aktionsauswahl schoss im Durchschnitt 1,34 Tore pro Partie.

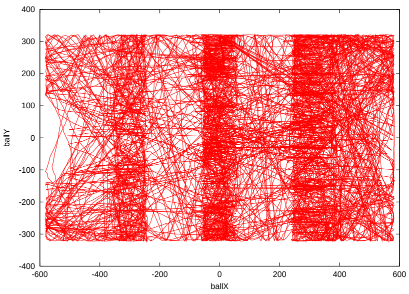
Neben der Frage nach der Gesamtanzahl geschossener Tore ist deren Häufigkeit über die Zeit von Interesse. MITRA war in der Lage, $0,67$ Tore pro Minute zu erzielen, während die alte Programmversion einen Wert von $0,10$ erreichte.



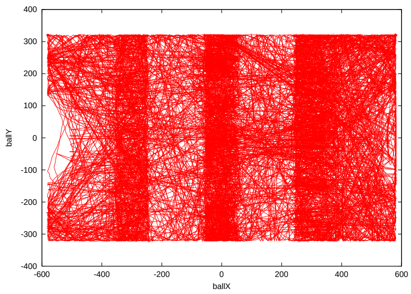
(a) Nach einer Minute



(b) Nach fünf Minuten



(c) Nach zehn Minuten



(d) Nach zwanzig Minuten

Abbildung 8.1: Balltrajektorien im Verlauf eines Spieles

Allerdings müssen diese Ergebnisse insofern abgeschwächt werden, als viele der MITRA zugesprochenen Tore Eigentore der alten Aktionsauswahl waren. Diese Eigentore fielen immer auf dieselbe Weise: Der Torwart schoss den hinter ihm parallel zur Torlinie rollenden Ball in das eigene Tor. Dass dennoch berechtigt von einer Überlegenheit von MITRA gesprochen werden kann, verdeutlicht die Abbildung 8.1, in der der Weg des Balls während einer zwanzigminütigen Partie aufgezeichnet ist – die Bilder entsprechen der Balltrajektorie nach (a) einer, (b) fünf, (c) zehn und (d) zwanzig Minuten.

Problemlos ist zu erkennen, dass der Schwerpunkt des Spieles sich in der Hälfte der alten Aktionsauswahl befindet (MITRA spielt von links nach rechts). Bei der Interpretation dieser Bilder muss beachtet werden, dass der Ball, falls er von keiner Figur mehr erreicht werden kann, in die nächstgelegene Ecke gelegt wird, von der aus er in das Spielgeschehen zurückrollt.

Die in den Abbildungen 8.1 und 8.2 aufallende Häufung von Ballbeobachtungen an bestimmten Stellen ist dadurch begründet, dass es sich um die Räume zwischen zwei in die entgegengesetzten Richtungen spielenden Stangen handelt.

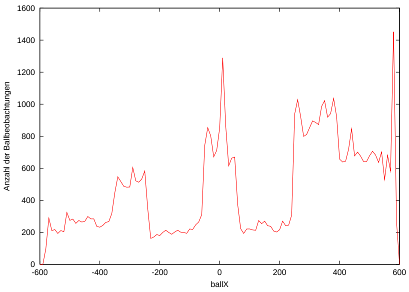
8.2.2 Einfluss der Suchtiefe auf die Spielstärke

Der Einfluss der Suchtiefe auf die Spielstärke von MITRA wurde am Simulator anhand von Partien gegen die alte Aktionsauswahl untersucht. Die Histogramme über den X-Wert des Balls finden sich in Abbildung 8.2.

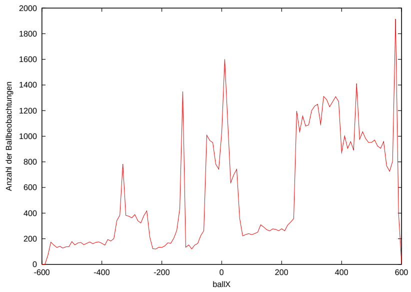
Darstellung 8.2(a) zeigt die Situation bei einer Tiefe von einem Vorhersageschritt. Obwohl in diesem Fall von einer Vorhersage im eigentlichen Sinne noch gar nicht gesprochen werden kann, da die Vorwärtssimulation direkt nach der Ausführung einer einzigen Aktion abbricht (es wird also immer die Aktion ausgewählt, deren Gelingen am wahrscheinlichsten ist), gelingt es MITRA bereits, den Ball 64 % der Zeit in der gegnerischen Hälfte zu halten.

Die Erhöhung der maximalen Suchtiefe um einen Vorhersageschritt (siehe Abbildung 8.2(b)) lässt die Überlegenheit nochmals sprunghaft ansteigen: Der Ball befindet sich nun 72 % der Zeit in der gegnerischen Hälfte, ein Wert, der sich durch eine weitere Erhöhung der maximalen Suchtiefe auf drei

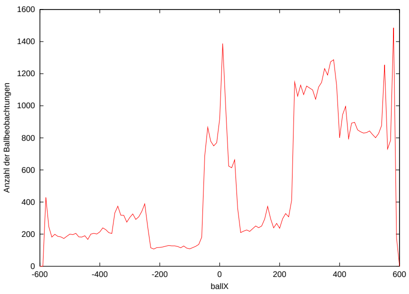
Vorhersageschritte nochmals auf 74 % steigern lässt (Abbildung 8.2(c)).



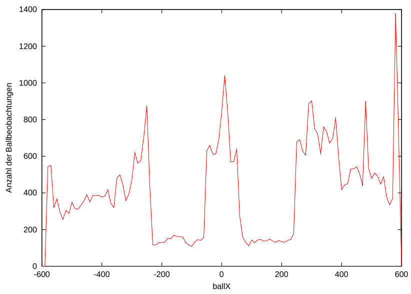
(a) Suchtiefe: Ein Vorhersageschritt



(b) Suchtiefe: Zwei Vorhersageschritte



(c) Suchtiefe: Drei Vorhersageschritte



(d) Suchtiefe: Vier Vorhersageschritte

Abbildung 8.2: Histogramme über den X-Wert des Balls. An der Abszisse aufgetragen ist die X-Koordinate des Balls, an der Ordinate die Häufigkeit der entsprechenden Ballbeobachtungen

Bei einer weiteren Inkrementierung der maximalen Suchtiefe auf vier Vorhersageschritte allerdings bricht die Leistungsfähigkeit von MITRA ein: Wie in Darstellung 8.2(d) erkenntlich liegt der Schwerpunkt des Spieles nur noch knapp (57 % der Zeit) in des Gegners Hälfte.

Der Grund hierfür ist darin zu suchen, dass bei der Suchtiefe von vier Vorhersageschritten, die einer maximalen Baumtiefe von zwölf Knoten entspricht, der Aktionsauswahlschritt zu viel Zeit benötigt. Da der Ablauf eines Zyklus des *KiRo*-Programms aber seriell verläuft, zieht dies das gesamte Spiel in

Mitleidenschaft. So werden bei Überschreitungen der zur Verfügung stehenden Zeit beispielsweise von der Kamera gelieferte Bilder nicht berücksichtigt, was zu einer deutlich verschlechterten Schätzung der Ballbewegung führt.

Einfluss der Lernalgorithmen

Die durch die Lernalgorithmen ermittelten Abschätzungen der Effizienz der Aktionen beeinflussen den Spielcharakter beträchtlich. Beurteilen lässt sich dies anhand des Diagramms in Abbildung 8.3. In diesem Histogramm wird verglichen, wie häufig die einzelnen Aktionen im Verlauf der ersten respektive letzten fünf Minuten eines 25-minütigen Spieles ausgewählt wurden. Dieses Spiel wurde mit „frischen", nur auf die oben erwähnte Art mit Gleichverteilungen initialisierten Lernalgorithmen begonnen.

Deutlich springt zunächst ins Auge, dass die Anzahl von **BlockBall**-Aktionen auf Kosten der **ClearBall**-Aktionen zunimmt; dies lässt sich allerdings weniger durch den direkten Einfluss der Lernalgorithmen begründen, sondern vielmehr dadurch, dass der Ball sich verstärkt in der gegnerischen Hälfte aufgehalten hat. Wie in Abschnitt 7.4 erwähnt wird ja jeder Stange, die nicht direkt in das Spielgeschehen eingreift, falls sich der Ball hinter dieser befindet, die **ClearBall**-Aktion zugewiesen. Ist der Ball zwischen dem gegnerischen Tor und der Stange, so erhält sie ihre Steuerbefehle von einer **BlockBall**-Aktion; das Verhältnis zwischen den Häufigkeiten dieser Aktionen lässt also Rückschlüsse darauf zu, wo der Schwerpunkt des Spielgeschehens sich befindet. Man kann also indirekt darauf schließen, dass die Spielstärke von MITRA durch die Verbesserung der Güte der Effektivitätsabschätzungen deutlich zugenommen hat.

Weiterhin fällt in dem Diagramm allerdings auf, dass der Anteil der **StopBall**-Aktionen deutlich (um 43 %) abgenommen hat; dies aber ist eindeutig auf die im Verlauf des Lernens immer schlechter werdenden Effektivitätsabschätzungen für diese Aktion zurückzuführen.

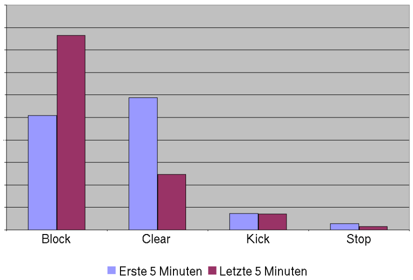


Abbildung 8.3: Einfluss der Lernalgorithmen. Gezeigt ist die Veränderung der Häufigkeit aller Aktionen zu Beginn und am Ende eines Spieles

8.2.3 Ergebnisse am realen Tisch

Die derzeitige Ausführung der Hardware von *KiRo* lässt Spiele verschiedener Programme gegeneinander momentan noch nicht zu. Aus diesem Grund bestanden die Experimente am realen Tisch aus Partien gegen den einzigen garantiert unparteiischen, den „statischen“ Gegner. Die alte Aktionsauswahl und MITRA mussten also bei unbewegten, allerdings in der senkrechten Position gehaltenen gegnerischen Stangen sich in der Menge der Tore pro Zeit messen.

Die ermutigenden Ergebnisse vom Simulator ließen sich am realen Tisch leider nicht bestätigen: Hier war die alte Aktionsauswahl MITRA sogar knapp überlegen. Erstere war in den Experimenten in der Lage, 1,48 Tore pro Minute zu erzielen, während MITRA nur auf 1,43 Tore kam. Allerdings muss dieses Ergebnis, wie auch die Ergebnisse aus den am Simulator durchgespielten Partien, mit Vorsicht genossen werden: Viele der gefallenen Tore wären gegen reale Gegner so nie zustande gekommen; dies gilt für beide Aktionsauswahlverfahren gleichermassen. Häufig waren beispielsweise Tore, die entstanden, indem der an der Stirnwand abgeprallte Ball von hinten gegen die Verteidigerfiguren stieß und so in das Tor zurückrollte.

Bei der Beobachtung des Spieles fiel allerdings auf, dass MITRA häufig in der Lage war, mit gezielten Schüssen nach rechts oder links aus schwierigen Situationen direkte Tore zu erzielen, während die alte Aktionsauswahl im allgemeinen nur dann Tore schoss, wenn der Ball sich zufällig in der Spielfeldmitte befand. Dass die Anzahl der so geschossenen Tore dennoch leicht höher war als die von MITRA erklärt sich dadurch, dass die alte Aktionsauswahl deutlich häufiger schoss.

Zur Bewertung dieser Ergebnisse wurden auch am Simulator Partien gegen den „statischen“ Gegner durchgespielt; in diesen war MITRA mit 2,45 Toren pro Minute gegenüber 1,85 Toren/Minute aus Seiten der alten Aktionsauswahl überlegen.

8.3 Interpretation der Ergebnisse

Wie lassen sich diese Ergebnisse nun interpretieren? Festzuhalten bleibt: Es wurden zwei verschiedene Arten von Experimenten durchgeführt, die beide spezifische Schwächen haben: Der Simulator ist nun einmal nur eine Annäherung an die Verhältnisse in der Realität, und das Spiel gegen einen unbewegten Gegner kann auch nicht als besonders realitätsnah angesehen werden.

Zudem schienen die Ergebnisse der jeweils schwächeren Aktionsauswahl – der alten am Simulator, MITRA am realen Tisch gegen den statischen Gegner – zumindest teilweise durch behebbare Mängel verschlimmert geworden zu sein: Am Simulator schießt die alte Aktionsauswahl (beziehungsweise der von dieser gesteuerte Torwart) häufig Eigentore, am realen Tisch gelingt es von MITRA ausgewählten **KickBall**-Aktionen häufig nicht, bei einem von hinten nach vorne rollenden Ball den Abschusszeitpunkt zu erkennen.

Alles in allem aber scheint vor allem die deutliche Feldüberlegenheit von MITRA am Simulator für eine Berechtigung dieses Ansatzes zu sprechen. Hierfür sprechen zwei Argumente: Das simulierte Spiel gegen einen agierenden Gegner hat eine größere Aussagekraft als das reale Spiel gegen einen statischen, und die Probleme der Bilderkennung, die eine vorausschauende Aktionsauswahl natürlich schwerer treffen als eine rein reaktive, werden in Zukunft wohl eher abnehmen (siehe Abschnitt 3.5). Es ist also eine Annäherung der Verhältnisse in der Realität an die im Simulator zu erwarten. Hier aber zeigte sich MITRA überlegen.

Kapitel 9

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde gezeigt, dass es möglich ist, das Tischfußballspiel des Tischfußballroboters *KiRo* durch die Verwendung einer vorausplanenden Aktionsauswahl MITRA zu verbessern. Dieses Verfahren basiert auf dem MINIMAX-Algorithmus, dessen Einsatzgebiet normalerweise die Brettspiele sind. Er wird dort zur Bestimmung des optimalen Zuges in einer gegebenen Spielsituation benutzt. Eine Grundannahme ist hierbei, dass der Gegenspieler seinerseits ebenfalls optimal spielt. MINIMAX wurde folgenderweise modifiziert:

- Die MINIMAX zugrundeliegende Annahme des optimalen Gegenspielers wurde fallengelassen. Stattdessen werden probabilistische Modelle des Gegners eingesetzt, um dessen Handlungen so gut wie möglich nachzubilden.
- Ein neuer Typ von Knoten, der Simulationsknoten, wurde eingeführt. In diesem wird die Entwicklung des Spieles vorwärtssimuliert, um die Auswirkungen der zur Verfügung stehenden Aktionen abschätzen zu können.
- Die Modellierung der Züge beider Spieler unterscheidet sich von der von den Brettspielen gewohnten Form: Während dort ein Zug einer Änderung des Spielzustandes entspricht, werden bei MITRA die Konsequenzen eines Zuges auf den Weltzustand erst im Verlauf der Simulation

wirksam.

Aufbauend auf bereits bestehenden Komponenten des *KiRo*-Systems wurde ein vollständig neues Verfahren aus der Taufe gehoben. Als atomare Bausteine wurde eine Menge von Aktionen definiert, die nach völlig neuen Kriterien entworfen wurde und daher im Aufbau von den bis dato existierenden Aktionen grundverschieden ist. Der Grund hierfür liegt darin, dass es zur Verwendung im Zuge der Vorwärtssimulation notwendig war, eine große Zahl von Aktionen parallel zu erzeugen, um sich dann für die besten darunter zu entscheiden. Die hierfür zur Vermeidung von Interferenzen notwendige Kapselung war in den vorgefundenen Aktionen nicht gegeben. Eine der ersten und grundlegendsten Herausforderungen während der Arbeit war es, diese Aktionen in einer sowohl auf dem Tisch als auch in der Simulation funktionierenden Weise zu erstellen, da sich oft zeigte, dass in der Simulation erfolgversprechende Aktionen in der Realität nicht einsetzbar waren.

Jede dieser Aktionen verfügt über zwei Prädikate, `erfolg(prae, post)` und `aktiv(s)`. Durch von `aktiv(s)` wird beurteilt, ob die Aktion im Spielzustand `s` in das Spielgeschehen eingreift. Mit `erfolg(prae, post)` kann erkannt werden, ob eine zwischen den Spielzuständen `prae` und `post` ausgewählte Aktion erfolgreich war. Während `aktiv` auch zur Steuerung der Aktionen eingesetzt wird, handelt es sich bei `erfolg` um eine reine Analysefunktion.

Die Prädikate `aktiv` und `erfolg` ermöglichen den Einsatz von Lernalgorithmen. Die von diesen gelieferten Abschätzungen über die Effektivität der Aktionen werden in der Vorwärtssimulation zur Erstellung von Nachfolgezuständen verwendet. Sind also `aktiv` und `erfolg` bei einer Aktion korrekt implementiert, so fließen die Veränderungen der Effektivität der Aktion, die z.B. bei einer besseren Abstimmung erreicht werden, automatisch in die Vorhersagekomponente ein. MITRA kennt also die Güte der zur Verfügung stehenden Aktionen, selbst wenn die Funktionsweise dieser verändert wurde.

Als für die Leistungsfähigkeit von MITRA essentiell stellte sich die Ausführung der Bewertungsfunktion dar. Schnell zeigte sich, dass triviale Ansätze direkt zu ungenügendem Spiel führten. Die in dieser Arbeit präsentierte Funktion zeigte unter einer Vielzahl getesteter Kandidaten die besten Ergebnisse. Sie bezieht in die Bewertung eines Spielzustandes sowohl den Ort des Ball als auch die Identität des ballführenden Spielers mit ein. Über Bestimmung dieses Spielers wird auch die Bewegung des Balls in die Bewertungsfunktion mit einbezogen.

Bei im Zuge der Implementierung zu fällenden Designentscheidungen wurde meistens der einfachsten funktionierenden Variante der Vorzug gegeben. Der Vorteil dieser Vorgehensweise war, dass diese Arbeit mit einem funktionierenden, erfolgreichen System abgeschlossen werden konnte. Hieraus folgt aber auch, dass viele Komponenten Raum für Verbesserungen lassen: Die Aktionen selbst sind noch optimierbar und die Gegnermodelle sehr grob. Hier scheint der Einsatz datenbasierter Modelle vielversprechend: Auf diesem Weg ließe sich eine Adaptivität von MITRA an den Gegner erreichen. Außerdem ist die Modellierung der physikalischen Vorgänge in der tatsächlichen Welt derzeit äußerst primitiv. Der für die Experimente verwendete Simulator kann allerdings über eine Schnittstelle zur Vorwärtssimulation eingesetzt werden. In der momentanen Implementierung basiert die Bestimmung der durch die Vorwärtssimulation überbrückbaren Zeiträume momentan allerdings darauf, dass sich die Welt in der Vorwärtssimulation genau so verhält, wie die internen Vorhersagekomponenten der Aktionen dies prognostizieren. Ob sich dieses Verfahren auch für die Verwendung von externen Vorhersagekomponenten eignet, oder in diesem Fall ein alternativer Weg gefunden werden kann, ist momentan nicht absehbar. Vermutlich wird der Weg eher über eine direkte Integrierung von Komponenten des Simulators führen, da auf diesem Weg die bei Aufrufen über das Netzwerk nicht zu vermeidenden Latenzzeiten umgangen werden können.

Alle oben aufgeführten Maßnahmen müssen beim derzeitigen Aufbau des *KiRo*-Systems allerdings die Bedingung einhalten, dass der Vorgang der Aktionsauswahl mit den durch die Zyklusdauer begrenzten Ressourcen auskommt; dieses Problem lässt sich möglicherweise durch eine nebenläufige Implementierung der Aktionsauswahl, z.B. durch Threads, lösen. Ein derartig tiefgreifender Eingriff in das Programm würde aber große Mengen zusätzlicher Fragen aufwerfen, so z.B. wann ein Auswahlverfahren angefangen oder unter welchen Umständen es abgebrochen werden kann. Fest steht allerdings, dass die Anforderungen an die Genauigkeit der Abschätzung der Ballbewegung zunehmen, je länger der Vorgang der Aktionsauswahl dauert.

Die vorgestellte Aktionsauswahl konnte sich in den durchgeführten Experimenten bewähren. Insgesamt erscheint die Weiterentwicklung des Ansatzes von MITRA sinnvoll. Substantielle Gegenargumente ergaben sich im Laufe der Arbeit nicht. An einigen Stellen der Arbeit wurden heuristische Annahmen gemacht, die optimiert oder modifiziert werden können. Um allerdings zu einer abschließenden Beurteilung von MITRA zu gelangen, ist es unausweichlich, die Möglichkeit bereitzustellen, dass mehrere Automaten in der Realität gegeneinander spielen. Erst dieses Experiment kann Aufschluss darüber geben, welche Form der Aktionsauswahl für das Tischfußballspiel am besten geeignet ist.

Abbildungsverzeichnis

Literaturverzeichnis

[Albus et al. 1987] Albus, J. S., H. G. McCain und R. Lumia (1987). *NASA/NBS standard reference model for telerobot control*. Technischer Bericht 1235, NBS.

[Bennewitz et al. 2002] Bennewitz, M., W. Burgard und S. Thrun (2002). *Using EM to Learn Motion Behaviors of Persons with Mobile Robots*. In: *IROS*.

[Brooks 1986] Brooks, Rodney A. (1986). *A robust layered control system for a mobile robot*. IEEE Journal of Robotics and Automation, 2(1).

[Brooks 1991] Brooks, Rodney A. (1991). *Intelligence without representation*. Artificial Intelligence, 47:139–159.

[Cestnik 1990] Cestnik, B. (1990). *Estimating Probabilities: A crucial task in machine learning*. In: *Proceedings of the Ninth European Conference on Artificial Intelligence*, S. 147–149, London. Pitman.

[Dorer 1999] Dorer, Klaus (1999). *Behavior Networks for Continuous Domains using Situation-Dependent Motivations*. In: *IJCAI*, S. 1233–1238.

[Gowdy 2000] Gowdy, Jay (2000). *Emergent Archtiectures: A Case Study for Outdoor Mobile Robots*. Doktorarbeit, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3890.

[Kitano et al. 1997] Kitano, Hiroaki, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa und H. Matsubara (1997). *RoboCup: A Challenge Problem for AI*. AI Magazine, 18(1):73–85.

[Kleiner et al. 2002] Kleiner, A., M. Dietl und B. Nebel (2002). *Towards a Life-Long Learning Soccer Agent*. In: *Proceedings of the International RoboCup Symposium '02*, Fukuoka, Japan.

[Knuth und Moore 1975] Knuth, D. E. und R. W. Moore (1975). *An Analysis of Alpha-Beta Pruning*. Artificial Intelligence, 6:293–326.

[Konolige et al. 1997] Konolige, K., K. L. Myers, E. H. Ruspini und A. Saffiotti (1997). *The Saphira Architecture: A Design for Autonomy*. Journal of experimental & theoretical artificial intelligence: JETAI, 9(1):215–235.

[Koŝecká und Bajcsy 1993] Koŝecká, Jana und R. Bajcsy (1993). *Discrete Event Systems for Autonomous Mobile Agents*. In: *Proceedings Intelligent Robotic Systems*, S. 21–31, Zakopane.

[Maes 1990] Maes, P. (1990). *How to do the Right Thing*. Connection Science Journal, Special Issue on Hybrid Systems, 1.

[Michie 1961] Michie, Donald (1961). *Trial and Error*. Science Survey, part two, S. 129–145.

[Mitchell 1997] Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill.

[Müller 2000] Müller, Klaus (2000). *Roboterfussball: Multiagentensystem CS Freiburg*. Diplomarbeit, Universität Freiburg.

[Nilsson und Fikes 1971] Nilsson, N. und R. Fikes (1971). *Strips: A new approach to the application of theorem proving to problem solving*. Artificial Intelligence, 2:189–208.

[Nilsson 1984] Nilsson, N. J. (1984). *Shakey the robot*. Tech. Note 323, SRI AI Center, Menlo Park, CA.

[Pirjanian 1999] Pirjanian, P. (1999). *Behavior Coordination Mechanisms - State-of-the-art*.

[Plaat et al. 1996] Plaat, Aske, J. Schaeffer, W. Pijls und A. de Bruin (1996). *Best-First Fixed-Depth Minimax Algorithms*. Artificial Intelligence, 87(1-2):255–293.

[Riedmiller et al. 2001] Riedmiller, M., A. Merke, D. Meier, A. Hoffman, A. Sinner, O. Thate, und R. Ehrmann (2001). *Karlsruhe Brainstormers — A Reinforcement Learning Approach to Robotic Soccer*. In: Stone, Peter, T. Balch und G. Kraetszchmar, Hrsg.: *RoboCup-2000: Robot Soccer World Cup IV*, S. 367–372. Springer Verlag, Berlin.

[Rosenblatt 1997] Rosenblatt, Julio (1997). *DAMN: A Distributed Architecture for Mobile Navigation*. Doktorarbeit, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

[Russel und Norvig 1995] Russel, Stuart und P. Norvig (1995). *Artificial Intelligence - A Modern Approach*. Prentice Hall International Editions.

[Shannon 1950] Shannon, Claude E. (1950). *Programming a computer for playing chess*. Philosophical Magazine, 41(4):256–257.

[Stockman 1979] Stockman, G. C. (1979). *A Minimax Algorithm Better than Alpha-Beta?*. Artificial Intelligence, 12:179–196.

[Stone und McAllester 2001] Stone, Peter und D. McAllester (2001). *An architecture for action selection in robotic soccer*. In: Müller, Jörg P., E. Andre, S. Sen und C. Frasson, Hrsg.: *Proceedings of the Fifth International Conference on Autonomous Agents*, S. 316–323, Montreal, Canada. ACM Press.

[Sutton 1988] Sutton, Richard S. (1988). *Learning to Predict by the Methods of Temporal Differences*. Machine Learning, 3:9–44.

[Weigel et al. 2002] Weigel, Thilo, J.-S. Gutmann, M. Dietl, A. Kleiner und B. Nebel (2002). *CS-Freiburg: Coordinating Robots for Successful Soccer Playing*. IEEE Transactions on Robotics and Automation, 18(5):685–699.

[Weigel und Nebel 2002] Weigel, Thilo und B. Nebel (2002). *KiRo - An Autonomous Table Soccer Player*. In: *Proceedings of the International RoboCup Symposium '02*, Fukuoka, Japan.