Kapitel 6

Gelernte Robotersimulation

In diesem Kapitel zeige ich, wie die gelernte Roboterdynamik für ein Simulationssystem genutzt wird. Der Simulator sagt den Zustand des nächsten Frames voraus, indem lernbare Vorhersagesysteme integriert werden [Gloye, 2004]. Bei einer Änderung der Hardware des Roboters hat dies den Vorteil, dass keine Zeit aufgebracht werden muß, um die Parameter eines physikalischen Modells anzupassen oder gar ein nicht mehr ausreichendes physikalisches Modell durch ein anderes zu ersetzen. Dies ist zum Beispiel der Fall, wenn neue oder andere Motoren in die Roboter eingebaut oder wenn die Parameter des PID-Controllers auf dem Roboter optimiert werden (siehe Kapitel 9).

Ein Simulator ist sehr hilfreich, um schneller höhere Verhalten zu entwickeln und zu testen. Ein Test auf dem Spielfeld würde unter anderem Zeit für das Setup des physikalischen Systems erfordern: Die Roboter müssten geladen sein, mechanische Defekte müssten behoben worden sein, die Kameras müssen den Lichtverhältnissen angepasst werden, usw. Außerdem können mit Hilfe eines Simulators mehrere Personen gleichzeitig Verhalten auf gewöhnlichen Arbeitsplatzrechnern entwickeln; es gibt keine Konkurrenz um die Hardware. Dies ist gerade bei größeren Teams sehr wichtig.

Damit die entwickelten Verhalten auch gut in die Realität übertragbar sind, müssen die Modelle im Simulator angemessen die reale Hardware wiederspiegeln. Die Anpassung des Simulators an neue Roboter sollte auch möglichst schnell und einfach geschehen, denn die zur Verfügung stehende Zeit sollte lieber für anspruchsvolle Aufgaben, wie Verhaltensprogrammierung, genutzt werden. Doch nicht nur neue Roboter verlangen nach einer Anpassung des Simulators. Selbst ein anderer Teppich auf dem Spielfeld, zum Beispiel bei einem Wettbewerb, beeinflusst das physikalische Verhalten der Roboter. Die Räder können mehr oder weniger Haftung haben und der Roboter dadurch langsamer oder schneller beschleunigen. Daher ist es sinnvoll, ein neues Modell ohne weiteren Programmieraufwand lernen zu lassen.

In einer Gruppe um Andrew Ng von der Stanford University wurde für einen Simulator das Verhalten von Modell-Hubschraubern mit einem Vorhersagemodul trainiert [Ng, 2004]. Der Zustand des Helikopters lässt sich mit zwölf Variablen beschreiben: Position, Orientierung und Geschwindigkeiten im dreidimensionalen Raum. Die Manipulatoren haben fünf Freiheitsgrade, von denen nur vier benutzt werden. Die Rotorneigung längs- und seit-wärts, die Flügelstellung des Hauptrotors und die Flügelstellung des Heckrotors sind veränderlich; die Geschwindigkeit der Rotoren ist konstant. Für die Datenaufzeichnung zum Trainieren des Vorhersagemoduls wurde die Maschine von einem erfahrenen Piloten 479 Sekunden ferngesteuert. Die Daten wurden mit einer Frequenz von 50Hz aufgezeichnet. Damit wurde eine lineare Regression durchgeführt. Es wurden keine historischen Daten verwendet. Um das Delay zu überwinden wurde ein Kalmanfilter auf die Daten angewendet. Das gelernte Modell wurde dann für verschiedene Anwendungen benutzt. Mit Hilfe eines neuronalen Netzes, dessen Parameter mit einem Verstärkungslernverfahren trainiert wurde, konnte der Hubschrauber autonom fliegen. Die gestellten Aufgaben konnten von dem System besser bewältigt werden als von menschlichen Piloten, die den Simulator mit dem Joystick manövrierten. In einem späteren Experiment wurde auch ein realer Hubschrauber innerhalb von 72 Stunden simuliert, im Simulator trainiert und autonom geflogen [Ng, [2]2004]. Auf der Webseite[1] von Andrew Ng befinden sich einige Videos des autonomen Hubschraubers.

6.1 Das verwendete Simulationssystem

Das Simulationssystem kommuniziert direkt mit dem Verhaltenssystem. Bildverarbeitung, Kommunikationsmodul, Kamera und Roboter werden ersetzt. Das Verhalten teilt dem Roboter die Fahrgeschwindigkeiten aller Roboter mit. Der Simulator berechnet die Positionen sowie Oientierungen der virtuellen Roboter und die Position des Balls und gibt die Daten im Takt der Bildwiederholrate an das Verhalten zurück. Die Daten werden zusätzlich verrauscht. Dies erhöht den Realismus des Systems. Es können unterschiedlich Modelle für die Roboter verwendet werden. Im Abschnitt 6.3 wird darauf genauer eingegangen. In Abbildung 6.1 ist eine Bildschirmkopie des Simulationsmoduls zu sehen. Über eine Menü können unterschiedliche Simulationsmodelle ausgewählt werden und deren Parmeter, wie zum Beispiel die Trägheit des Roboters oder das Rauschen der Position des Balls, eingestellt werden.

Die Simulation ist zwischen den Takten der künstlichen Bildwiederholrate ereignisbasiert. Das heißt, die Positionen werden im Zeitintervall einer Periode berechnet. Die Bewegungen der Objekte werden in dieser Zeit als geradlinig angenommen. Falls es in der Zeit zwischen der Periode zu einer Überschneidung der Objekte kam, wird bis zu dem Zeitpunkt der ersten Kollision zurück gegangen und das Ereignis behandelt. Dies wird so oft wiederholt, bis keine

[1] www.robotics.stanford.edu/~ang/rl-videos

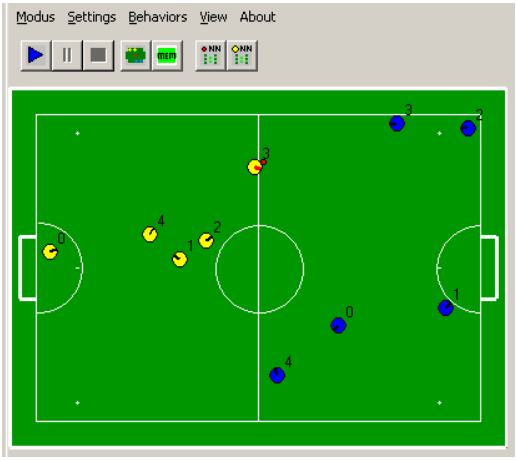

Abbildung 6.1: Bildschirmkopie des Simulationsmoduls. Roboter werden als Kreise dargestellt. Die Orientierung wird durch einen Strich vom Zentrum des Roboters zu seiner Vorderseite symbolisiert. Wenn der Schuss des Roboters aktiviert ist färbt sich der Orientierungsbalken rot; ansonsten ist er schwarz. Über das Menü können Modelle ausgewählt und die Parameter der Modelle eingestellt werden. Die Positionen sowie Orientierungen der Roboter und die Position sowie die Geschwindigkeit des Balls können mit der Maus frei gewählt werden.

Überschneidungen mehr auftreten. Anschließend werden die neuen Positionen und Orientierungen an das Verhaltenssystem zurückgegeben.

6.2 Adaption der gelernten Vorhersage

Die gelernte Vorhersage für das Roboterverhalten wurde in Kapitel 5 vorgestellt. Sie bekommt als Eingabe die Roboterpositionen und Orientierungen für die vergangenen sechs Frames, relativ zu den gegenwärtigen Werten. Die Ausgabe ist dann die Roboter-Position und -Orientierung vier Frames in der Zukunft. Die Vorhersage ist wichtig, um die Totzeit des Systems, die Zeit zwischen dem Erfassen des Roboters durch die Bildverarbeitung und die Ausführung der darauf basierenden berechneten Befehlen, zu überbrücken. Sie ermöglicht somit die Roboter schneller und präziser zu steuern.

Die Vorhersage macht nichts anderes, als die Dynamik des Roboters zu lernen, das heißt, wie er sich bei verschiedenen Fahrbefehlen in seiner Umwelt verhält. Es liegt also nahe, die so gelernten Informationen für eine Simulation des Roboters zu nutzen. Die Positionen und Orientierungen, die die Ausgabe der Vorhersage bilden, werden als die tatsächlichen, simulierten Werte für den Ro-

boter genommen. Es entfällt jedoch der Vergleich und die Korrektur zwischen Vorhersage und den Daten aus der Bildverarbeitung.

Um die Vorhersage für die Simulation benutzen zu können, müssen einige zusätzliche Eigenschaften und Besonderheiten berücksichtigt werden. Für die Vorhersage werden keine anderen Eingaben ausser den relativen Roboter-Positionen und -Orientierungen verwendet, wie zum Beispiel Abstände zu Wänden oder die Positionen von anderen Robotern. Dies ist für die Vorhersage auch nicht notwendig, da die Kollisions-Erkennung und -Vermeidung im Verhaltenssystem ein eigenständiges Modul ist. Dies führt zwar gelegentlich zu falschen Vorhersagen, diese werden aber durch das höhere Verhaltenssystem ausgeglichen.

Für ein Simulationssystem ist jedoch die Kollisionserkennung unvermeidlich. Sie ist relativ unabhängig von einem Roboter bzw. Robotermodell. Nur die Größe des Roboters ist eine wichtige Eigenschaft, die berücksichtigt werden muss. In der Regel läuft das Verhalten „frameweise“, also diskret ab. Das heißt, Kollisionen müssen auch zwischen den Frames erkannt werden, sonst könnte ein Roboter, wenn er schnell genug ist, durch einen anderen Roboter hindurch fahren.

Ausser der Kollisionserkennung ist auch die Ballvorhersage mit Reflektion an den Robotern und den Wänden ein wichtiger Bestandteil des Simulationssystems. Der Schussbefehl für die Roboter muss auch für die Ballsimulation berücksichtigt werden. Der Ball hat ein sehr einfaches Modell. Die nächste Position $p(t+1)$ berechnet sich aus einer gedämpften Geschwindigkeit $v(t+1)$ des Balles, indem auf die aktuelle Position $p(t)$ ein Teil der Geschwindigkeit $v(t)$ addiert wird:

$$\begin{aligned} v_x(t+1) &= \lambda v_x(t) \\ v_y(t+1) &= \lambda v_y(t) \\ p_x(t+1) &= p_x(t) + v_x(t+1) \\ p_y(t+1) &= p_y(t) + v_y(t+1) \end{aligned}$$

Der Dämpfungsfaktor λ kann frei gewählt werden, damit unterschiedliche Teppiche berücksichtigt werden können. Ein üblicher Faktor ist 0,995. Nach einer Sekunde hat dann der Ball noch 85% seiner Geschwindigkeit, bei einer Bildwiederholrate von 30Hz.

6.3 Simulationsansätze

In den nächsten Abschnitten werden die unterschiedlichen Ansätzen vorgestellt, die zur Simulation der Roboter bei den FU-Fighters benutzt werden. Als erstes

wird kurz auf das die physikalische Methode eingegangen. Anschließend stelle ich zwei Verfahren vor, die auf der gelernten Vorhersage basieren und die zu erheblich besseren Ergebnissen führen.

6.3.1 Vereinfachtes physikalisches Modell

Bei dem einfachen physikalischen Modell wird die Trägheit und die Geschwindigkeit des Roboters berücksichtigt. Ähnlich wie bei der Berechnung der neuen Ballposition wird die neue Roboterposition $p(t+1)$ und Orientierung $o(t+1)$ berechnet, indem auf die aktuelle Position der Geschwindigkeitsvektor addiert wird. Es wird jedoch nicht nur ein Dämpfungsfaktor verwendet, denn die Geschwindigkeit wird bezüglich der aktuellen Fahrbefehle $f(t)$ angepasst:

$$\begin{aligned}
v_x(t+1) &= (1-\alpha_f)\beta_f f_x(t) + \alpha_f v_x(t) \\
v_y(t+1) &= (1-\alpha_f)\beta_f f_y(t) + \alpha_f v_y(t) \\
v_\omega(t+1) &= (1-\alpha_f)\beta_d f_\omega(t) + \alpha_d v_x(t) \\
p_x(t+1) &= p_x(t) + v_x(t+1) \\
p_y(t+1) &= p_y(t) + v_y(t+1) \\
o(t+1) &= R_{v_\omega(t+1)} o(t)
\end{aligned}$$

Das Modell ist sehr einfach. Die Faktoren α_f, α_d spiegeln die Trägheit und β_f, β_d die Kraft der Fahrbefehle wieder; R_ξ ist eine Rotationsmatrix, die einen Vektor um den Winkel ξ dreht. Die Fahrbefehle f wirken sich linear auf die Geschwindigkeit v aus. Die Beschleunigung wird also als konstant angenommen. Um näher an der Realität zu bleiben, wird im nächsten Abschnitt ein besseres Modell verwendet.

6.3.2 Simulation durch Vorhersage vier Frames in die Zukunft

In diesem Ansatz wird das unveränderte Vorhersagesystem verwendet, welches die Position und Orientierung in dem vierten Frame in der Zukunft vorhersagt. Um die Position und Orientierung des Roboters ein Frame in der Zukunft zu schätzen, werden die Werte geviertelt (siehe Abbildung 6.2). Position und Orientierung werden getrennt voneinander berechnet. Für die Position wird ein Viertel des Wegs zwischen der gegenwärtigen und der vorhergesagten Position, für die Orientierung wird ein Viertel der Drehung genommen.

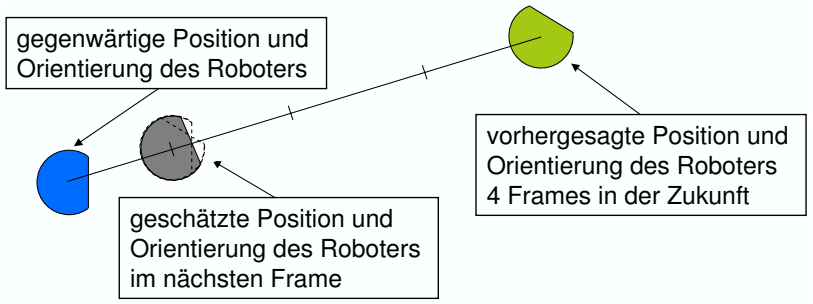

Abbildung 6.2: Um die Position und Orientierung des Roboters im nächsten Frame zu erhalten, werden die vorhergesagten Werte geviertelt. Für die Position wird ein Viertel des Weges zwischen der gegenwärtigen und der vorhergesagten Position, für die Orientierung ein Viertel des Drehungswinkels berechnet.

Die Kollisionserkennung wird weiterhin von dem Simulationssystem durchgeführt. Dafür werden die geschätzten Werte für die Roboter-Position und -Orientierung an das Simulationsmodul weitergegeben.

6.3.3 Simulation durch Vorhersage ein Frame in die Zukunft

Ein anderer Ansatz, die Vorhersage zur Simulation zu verwenden, ist eine spezielle Vorhersage zu lernen. Die in Abbildung 6.3 vorgestellte Lernmethode berechnet die Position des Roboters und seine Orientierung für das nächste Frame. In diesem Fall ist keine Korrektur notwendig, die Werte können direkt an das Simulationssystem für die Kollisionserkennung weitergegeben werden.

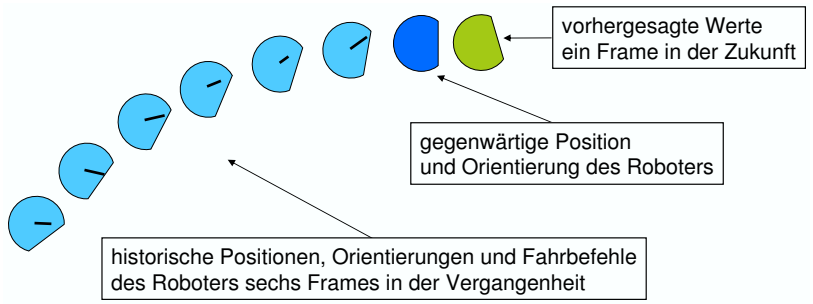

Abbildung 6.3: Diese Vorhersage lernt direkt die Position und Orientierung des Roboters im nächsten Frame. Eine anschließende Korrektur ist nicht notwendig.

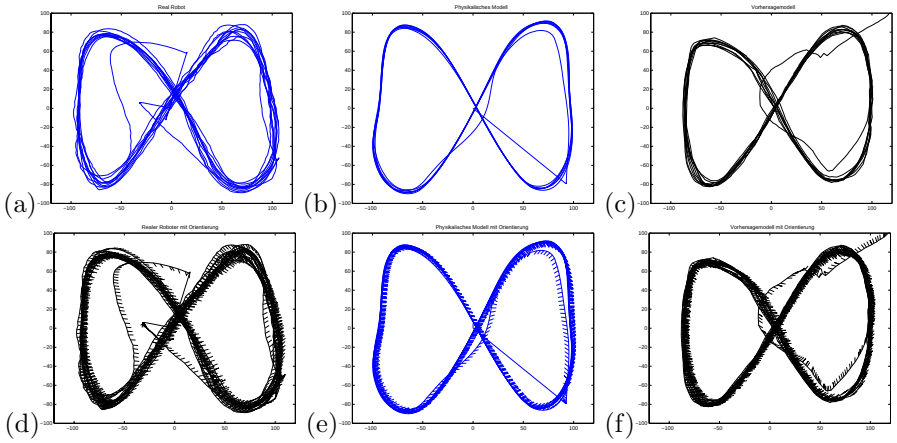

Abbildung 6.4: Ein Vergleich der Simulationsmethoden: (a) Der abgefahrene Pfad mit einem realen Roboter, (b) die Simulation mit einem einfachen physikalischen Modell und (c) die Simulation mit dem Vorhersagesystem. Die Grafiken der unteren Reihe zeigen die gleichen Pfade wie in der oberen Reihe, jedoch mit der Orientierung des Roboters. Wie deutlich zu sehen ist, simuliert das Vorhersagemodell den Roboter realistischer als das physikalische Modell.

6.4 Ergebnisse

Die Ergebnisse aller drei Simulationsmodelle sind in Abbildung 6.4 zu sehen. Die Aufgabe des Roboters ist es, dem Pfad einer liegenden 8 auf dem Spielfeld zu folgen. Wie deutlich zu erkennen, ist der Pfad des einfachen physikalischen Modells (mittlere Graphen) gegenüber dem realen Roboter (linke Graphen) zu „glatt“. Die Simulation mit Hilfe der Vorhersage (rechte Graphen) entsprechen schon eher dem realen Roboter. Das physikalische Modell könnte zwar durch ein realistischeres ersetzt werden, das auch dem unvorhersagbaren Verhalten des Roboters Rechnung trägt, aber dies wäre wieder mit Programmieraufwand verbunden, der durch den Einsatz des Vorhersagemodells entfällt.

6.5 Andere Ansätze

Der verbreitetste Ansatz zur Simulation von Robotern ist der Versuch, ein möglichst exaktes Modell der Roboter zu erstellen. Nicht nur im RoboCup, sondern allgemein in der Robotik. Dabei kann man zwei Klassen von Simulatoren unterscheiden. Die eine Art versucht spezifisch auf des Problem bezogene physikalische Modelle zu bauen und diese zu simulieren. Der andere Ansatz benutzt vorgefertigte physikalische Simulationssysteme, in denen nur die zu simulierenden Objekte modelliert werden müssen. Meist gibt es auch zusätzliche

Module für diese Systeme, die eine dreidimensionale Darstellung der Simulation ermöglichen.

Die Anzahl verfügbarer Simulatoren für Roboter ist unüberschaubar. Die Webseite Netinformations Computer Guide[2] führt 35 Simulatoren auf, die teilweise für sehr spezielle Roboter entwickelt wurden. Auf der Webseite von Michiel van de Panne[3] sind frei erhältliche sowie kommerzielle Simulatoren für physikalische Modelle aufgeführt. Hier wird nur eine kleine Auswahl von Simulatoren vorgestellt, die im RoboCup eingesetzt werden oder werden könnten.

Simulatoren werden schon in der Designphase benutzt, um auftretende Kräfte zu berechnen, die bei dem realen Roboter auftreten würden und die schon vorab die Parameter der Hardwarespezifikation prüfen. Dieser Ansatz wird auch verwendet, um Laufalgorithmen für Humanoide zu testen. Für einen 80cm hohen und 12kg schweren Humanoiden wird in [Buss, 2003] eine Simulation vorgestellt, in der das dynamische Modell mit Hilfe der MATLAB Simulationsumgebung SIMULINK analysiert wurde.

Die allgemein verbreitetsten Systeme zur Echtzeit-Simulation von Festkörpern sind Karma[4] und Havok Physics[5] [Kögler, 2003]. Sie werden vorwiegend zur Simulation in Spielen eingesetzt, beispielsweise im Ego-Shooter Unreal[6]. Der Nachteil dieser Systeme ist, dass sie sehr teuer sind und für den Einsatz in Spielen entwickelt wurden. Ein Bezug zur Realität ist also nur bedingt vorhanden.

Das Swarm-Bot[7] Projekt, ein Multi-Agenten System für kooperierende Roboter, verwendet für das Simulationssystem Swarmbot3d das Vortex Toolkit der kanadischen Firma Critical Mass Labs[8] zur dynamischen Simulation mit einer 3D Grafikausgabe [Dorigo, 2004]. Die grafische Ausgabe wird als Sensoreingabe für die simulierte Kamera der Roboter benutzt. Alle Sensoren und Aktoren werden dabei simuliert: Greifarme mit Kräften, durchdrehende Raupenräder, jeder einzelne Abstandssensor, sogar die LEDs, die sich an den Robotern befinden. Die Simulation läuft langsamer ab als in der Realität. Sie wird nur deshalb benutzt, weil der Aufwand für die Wartung der Roboter zu groß ist.

Ein weiteres kommerzielles Simulationssystem, dass in der hunamoiden RoboCup-Liga eingesetzt wird ist NovodeX[9] [Behnke, 2004].

Ein sehr verbreitetes und frei erhältliches Simulationssystem ist ODE[10] (Open Dynamics Engine) von Russel Smith, der vorher Teile des Karma Simulators geschrieben hatte. Die ursprüngliche Idee, die Echtzeitsimulation von Spielwelten, ist leider erhalten geblieben. Es wird kein besonders großer Wert auf physika-

[2] www.netinformations.com
[3] www.cs.ubc.ca/~van/simulators.html
[4] udn.epicgames.com
[5] www.havok.com
[6] www.unreal.com
[7] www.swarm-bots.org
[8] www.criticalmasslabs.com
[9] www.novodex.com
[10] ode.org

lische Exaktheit oder physikalische Einheiten gelegt, was die Programmierung von physikalischen Modellen nicht gerade vereinfacht. Die Stabilität des Systems ist (noch) nicht ganz perfekt, doch die „Community“ in diesem Bereich ist so umfangreich, dass es sicher große Fortschritte in dieser Hinsicht geben wird.

Anwender von ODE innerhalb von RoboCup sind beispielsweise das AIBO-Team der Universität Bremen mit SimRobot[11] (siehe Abbildung 6.5), bei dem Kollisionen zwischen den Hunden berücksichtigt werden. Es werden jedoch keine Kräfte der Aktuatoren und deren Einfluss auf die Umwelt berücksichtigt.

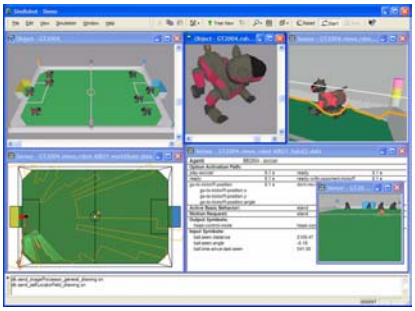

Abbildung 6.5: Eine Bildschirmkopie der Simulationssoftware SimRobot der Universität Bremen für die vierbeinigen AIBO Roboter. (Bild aus der Präsentation von Thomas Röfer zum SPP 1125 Workshop 27. und 28. November 2003)

Ein universell einsetzbares, kommerzielles Simulationssystem ist Webbots, der Firma Cyberbotics[12] [Michel, 2004]. Es enthält einige vordefinierte Roboter (zum Beispiel AIBO, HOAP-2, Khepera, Pioneer2) kann aber auch für eigene Roboter konfiguriert werden (siehe Abbildung 6.6). Für einige Roboter können Programme in der Simulationsumgebung getestet und anschließend auf den realen Roboter übertragen werden. Zur physikalischen Simulation ist ODE als Modul integriert. Im Vergleich zu den Preisen des Khepera Roboters ist das Simulationssystem auch recht preiswert.

Der Simulator DDSim[13] des Fraunhofer AIS, der unter anderem für deren Middle-Size-Roboter eingesetzt wird, benutzt zur Zeit kein physikalisches Modell für die Aktuatoren des Roboters. Es können jedoch mehrere Sensoren wie Laserscanner, Farbkameras, usw. simuliert werden (siehe Abbildung 6.7). Über eine XML Beschreibungssprache können verschiedene Roboter modelliert werden. Die Modelle der Roboter können in einem allgemeinen Format (VRML, 3DS) eingelesen und über CORBA auch auf anderen Rechner visualisiert werden. Das Sytem ist in Java geschrieben. Zu Forschungszwecken ist das Pro-

[11] www.informatik.uni-bremen.de/simrobot
[12] www.cyberbotics.com
[13] www.ais.fraunhofer.de/BE/env/ddsim

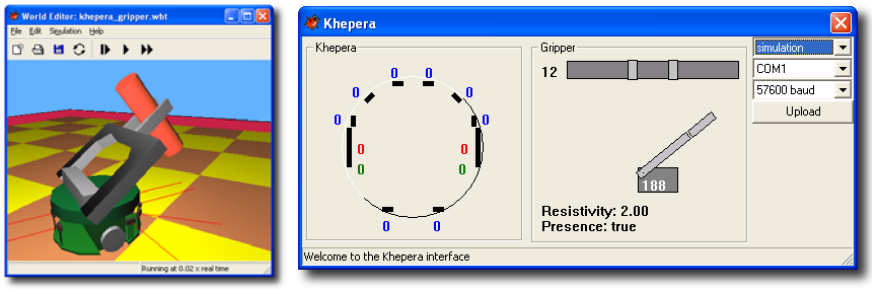

Abbildung 6.6: Ein mit der Software Webbots simulierter Khepera II Roboter mit Infrarotsensoren und einem Greifarm. Bilder von der Cyberbotics Webseite.

gramm frei erhältlich, jedoch nicht für den kommerziellen Einsatz. Nach Aussagen von Thomas Wisspeintner, einem Mitarbeiter des RoboCup-Teams des AIS, wird zur Zeit ein physikalischer Simulator implementiert.

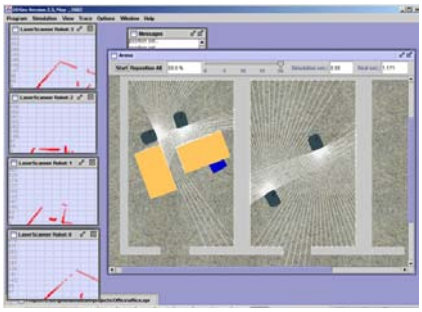

Abbildung 6.7: Ein Screenshot der konfigurierbaren Simulationssoftware DDSim des Fraunhofer Instituts für autonome intelligente Systeme. (Bild von der DDSim Webseite)

Der SimSrv[14] der Universitäten Stuttgart und Freiburg ist eine Simulationsumgebung, die netzwerkfähig, für unterschiedlichste Robotertypen konfigurierbar und durch eine Modulschnittstelle individuell erweiterbar [Kleiner, [2]2003] ist. Auch hier kann die 3D-Ausgabe als Eingabe für das System benutzt werden. Dies verursacht jedoch erhebliche Performanzprobleme. Der Simulator benutzt kein universelles physikalisches Simulationssystem. Zur Modellierung stehen verschiedene Module zur Verfügung. Unter anderem ein sogenanntes Blackbox-Modul, ein neuronales Netz, das trainiert werden kann. Auch ein Modul als physikalisches Modell, das durchdrehende Räder und Odometriefehler berücksichtigt ist verfügbar.

[14] kaspar.informatik.uni-freiburg.de/~simsrv

In der Small-Size-Liga hat die CMU einen aufwändigen physikalischen Simulator ÜberSim[15] geschrieben, der als Festkörper-Simulations-System auch ODE verwendet [Browning, 2003]. ÜberSim berücksichtigt die Reibung zwischen Oberflächen und die Massenverteilungen innerhalb der Roboter. In Abbildung 6.8 ist ein Vergleich zwischen dem Simulator und einem realen Roboter gezeigt. Zum Vergleich der hohen Genauigkeit dieser extrem guten Simulation sei auf die simple Simulation in Abbildung 6.4 verwiesen. Eine Anwendung, die den hohen Aufwand der Implementierung eventuell rechtfertigt, und die auch in ÜberSim durchgeführt wurde, ist die Simulation eines Segway RMP Roboters, bei dem die Simulation des Gleichgewichts sicher einer physikalischen Modellierung bedarf [Go, 2004].

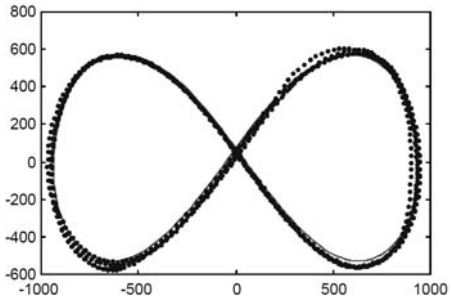

Abbildung 6.8: Ein Vergleich der virtuellen Fahrt eines Small Size Roboters in dem ÜberSim Simulator (durchgezogene Linie) mit dem realen Roboter (gepunkteter Pfad). Zufällig sollte die gleiche Aufgabe ausgeführt werden, die auch die FU-Fighters für die Positionsvorhersage (Abbildung 5.9) und den Simulator (Abbildung 6.4) nutzen. (Bild aus [Browning, 2003])

Man könnte auch den Simulator der RoboCup Simulationsliga [Noda, [2]1998] als einen Simulator für Roboter auf einer sehr abstrakter Ebene betrachten. Der in der entstehenden 3D-Simulationsliga verwendete Ansatz ist noch mehr als Roboter-Simulation zu verstehen, denn das Ziel dieses Projekts ist es, irgendwann Menschen als Fußballspieler zu simulieren. Der erste Prototyp der Universität Koblenz wurde mit Hilfe von ODE geschrieben. Die Agenten sind bisher nur einfache Kugeln. Die Simulation ist jedoch trotzdem so langsam, dass eine zehnminütige Halbzeit mehrere Stunden dauert.

6.6 Zusammenfassung

Der Vergleich existierender Simulationssysteme zeigt, dass der Aufwand, der bei der Erstellung physikalischer Modelle entsteht, erheblich ist. Es ist auch nicht

[15] www-2.cs.cmu.edu/~robosoccer/ubersim

in allen Bereichen, in denen man eine Simulation benötigt, sinnvoll, eine so hardwarenahe Simulation zu benutzen. Zum Testen höherer Verhalten genügt beispielsweise ein abstrakteres Modell. Ein weiterer gravierender Nachteil physikalisch exakter Simulationssysteme ist, dass sie bei etwas komplexeren Modellen nicht mehr echtzeitfähig sind, d. h. die Simulation ist langsamer als die Realität. Mitunter werden dann zeitaufwändig zusätzlich einfache Modelle in den Simulator integriert, die die Simulation (etwas) beschleunigen [Pettinaro, 2003].

Der hier vorgestellte Ansatz spart viel Zeit für den Modellierungsaufwand, läuft in Echtzeit, ist adaptiv und erreicht gute Ergebnisse. Es musste nur der vorhandene Simulator angepasst werden. Der Einsatz eines fremden Simulationssystems hätte viele Anpassungen bedurft, die sehr zeitaufwändig gewesen wären. Außerdem sind die universell einsetzbaren Simulationssysteme selbst sehr komplex und umfangreich. Eine vollständige 3D-Simulation ist für einen Simulator in der Small-Size-Liga nicht notwendig. Ausserdem kann eine 3D Animation auch unabhängig von einem Simulator verwendet werden, wie dies in dem FU-Fighters-Framework auch der Fall ist. Wenn jedoch Hochschüsse in der Small-Size-Liga selbstverständlich werden, wäre eine Erweiterung auf die dritte Dimension durchaus wünschenswert.

Wie sinnvoll ein adaptives, lernbares Simulationssystem ist, zeigt sich in mehrfacher Weise in den folgenden Kapiteln. Im nächsten Kaptel zeige ich, wie die Fahrbefehle des Roboters optimiert werden können, damit ein Roboter in einer Weise reagiert, wie es von ihm erwartet wird. Durch Beobachtung ist aufgefallen, dass die verwendeten Roboter meist schräg seitlich anfahren, auch wenn nur Fahrbefehle für eine Geradeausfahrt gesendet werden. Hier hätte ein physikalisches Modell im Simulator angepasst werden müssen. Durch die Verwendung des Vorhersagesystems war dies nicht nötig.