Autonome Linux-Roboter: Das Robocup-Team der Ulm Sparrows

Roadrunner, Cojote & Co.

Beim Roboter-Fußball gibt es alles, was man auf dem Rasen erwartet: Flanken, Fouls, gelbe Karten – nur keine Kopfbälle: Die Spieler sind autonome Roboter mit Kameras, Sensoren und Linux unter der Haube. Das Team Ulm Sparrows nahm am Robocup teil und berichtet von der Technik und ihren Tücken. Gerd Mayer, Hans Utz, Philipp Baer

Abbildung 1: Vor dem Spiel und in der Halbzeitpause waren die Teams vollauf damit beschäftigt, ihre Roboter neu zu kalibrieren oder anderweitig umzubauen. Wo keine Laptops zur Verfügung standen, wurden beim Robocup mitunter komplette PCs auf das Spielfeld getragen.

Roadrunner, Cojote, Speedy und Gonzales sind die vier Fußball-Roboter der Ulm Sparrows, die vom 1. bis 4. April dieses Jahres bei den Robocup German Open [1], der offenen deutschen Roboter-Fußballmeisterschaft, im Heinz-Nixdorf-Forum in Paderborn gegen 149 andere Teams antraten. Der Robocup [2] ist eine internationale Forschungsinitiative, um den Fortschritt in der Robotik und verwandten Bereichen wie künstliche Intelligenz oder Bildverarbeitung voranzutreiben. Als gemeinsame Testplattform und Herausforderung dient Fußball mit autonomen Robotern.

Das Roboterteam der Universität Ulm, die Ulm Sparrows [3], nimmt seit 1998 mit Erfolg an diesen Fußballspielen teil. War unser Team 2003 noch Vizemeister, verpassten wir in 2004 das Viertelfinale: Deutscher Meister in der Middle Size League wurden die Brainstormer Tribots der Uni Osnabrück, Vizemeister ist das Team Persia aus dem Iran.

Vier gegen vier

Auf einem 12 mal 8 Meter großen Feld spielen 2 mal 10 Minuten lang vier Roboter pro Team, ein Torwart und drei Feldspieler. Alle Objekte auf dem Spielfeld sind farblich eindeutig gekennzeichnet: Der Ball ist ein handelsüblicher FIFA-Winterfußball in Orange, die Tore sind gelb und blau, der Boden ist grün mit weißen Linien. Kommunikation zwischen den Robotern ist erlaubt, nicht jedoch mit Entscheidungsinstanzen außerhalb des Spielfelds, etwa mit zusätzlichen Rechnern.

Die Spielfelder sind mit Studioscheinwerfern ausgeleuchtet, um eine gleich bleibende, wenn auch unregelmäßige Ausleuchtung zu garantieren. Da die Roboter bis zu drei Meter pro Sekunde fahren und den Ball dabei auch noch beschleunigen können, kann sich das Ganze zu einer sehr dynamischen Szene entwickeln.

Im Robocup spielen die Roboter vollkommen autonom, das heißt, sie werden nur zu Beginn des Spiels gestartet. Danach wird von außen keinerlei Einfluss mehr genommen – abgesehen von der Halbzeitpause, in der die Teams Veränderungen an den Robotern vornehmen dürfen (Abbildung 1). Die Roboter suchen den Ball, finden ihn hoffentlich, drehen sich zum Tor und versuchen einen Treffer zu erzielen.

Forschungsprobleme

Selbstverständlich ist Roboter-Fußball kein Selbstzweck, sondern eine Testumgebung, in der viele Probleme der Robotik praktisch untersucht werden können. Die Umgebung ist hochdynamisch, die Roboter müssen gleichzeitig sowohl mit mehr oder weniger kooperativen Team-Mitglieder zusammenarbeiten als auch gegnerischen Robotern ausweichen. Die Problemstellungen beim Roboter-Fußball decken sich mit vielen Anforderungen der Praxis – etwa wenn ein autonomer Roboter am Samstagvormittag auf einem belebten Marktplatz einkaufen geschickt würde.

Im Detail sind die Themen etwa Selbstlokalisierung oder Objekterkennung. Die Klärung der Frage, wo genau der Roboter sich auf dem Feld befindet, ist Voraussetzung für eine Unterscheidung zwischen der Rolle als Verteidiger oder Angreifer. Weitere wichtige Punkte sind die

Handlungsplanung, also welche Aktion zu einem bestimmten Zeitpunkt ausgeführt werden soll, sowie die Koordination der Roboter untereinander.
Von Jahr zu Jahr ändert sich das Reglement des Robocup, um den Fortschritt zu erzwingen. Wurden zum Beispiel vor einigen Jahren noch Wände um das Spielfeld herum aufgestellt, stehen Zuschauer jetzt direkt an der Seitenlinie, was die gesamte Bildverarbeitung extrem erschwert.

Roboter-Hardware

Unsere aktuellen Roboter, die Sparrows 03, gehören bereits zur zweiten Generation selbst entworfener und gebauter Roboter unseres Teams. Bewährt haben sich eine runde Form, damit sie sich wenig verhaken, sowie ein pneumatischer Kicker mit einem Presslufttank, der genug Luft für ein Spiel enthält. Abbildung 2 zeigt eine Schemazeichnung der aktuellen Robotergeneration.
Um Probleme mit der Stromversorgung zu umgehen, werden alle Roboter mit einem kleinen Laptop betrieben. Der Hauptsensor der Roboter ist die Firewire-Kamera DFW V-500 von Sony. Firewire hat im Gegensatz zu analogen Kameras mit Framegrabber-Karten den Vorteil, dass es keine Probleme mit dem Zusammensetzen der Halbbilder (De-Interlacing) gibt. Speziell bei Eigenbewegungen und Rotationen liefern die Firewire-Kameras ein deutlich besseres Bild. Weitere Sensoren sind Infrarot-Abstandsmesser sowie Umdrehungszähler für die Räder (so genannte Odometrie).
Die Komponenten innerhalb der Roboter sind per CAN-Bus verbunden. Das ist ein serieller Bus aus dem Automobilbau, der mit ungeschirmten Leitungen auskommt und sehr fehlertolerant ist. Zusätzlich besteht die Möglichkeit, dass sich Komponenten am Bus direkt miteinander unterhalten, ohne jeweils über den Rechner zu gehen. Der CAN-Bus ist über einen externen Controller am Parallelport des Laptops angeschlossen.

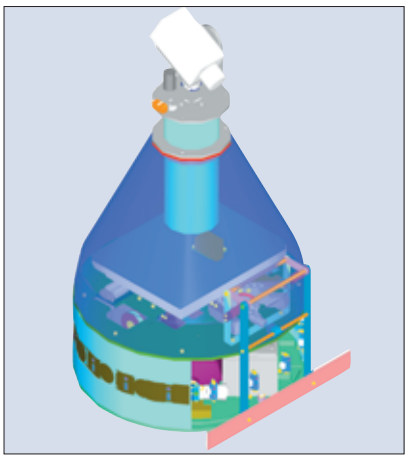

Abbildung 2: Der zentrale Rechner des Roboters ist ein IBM-Notebook, mit blauem Deckel und grauem Gehäuse dargestellt. Ein Stockwerk darunter sind der Antrieb zusammen mit der Batterie (grau) und dem Druckluftbehälter (violett) untergebracht.

Steuerung und Bildverarbeitung

Ein solch komplexes Projekt wie eine komplette Roboteranwendung erfordert eine durchdachte Software-Architektur. Die Software soll möglichst unabhängig von der tatsächlichen Roboterhardware funktionieren, damit Komponenten wie zum Beispiel die Selbstlokalisierung nicht nur im Roboter-Fußball, sondern zum Beispiel auch in einer Büroumgebung funktionieren.
Die von uns entwickelte Middleware für Roboter, kurz Miro genannt, ist einerseits eine solche Abstraktionsebene für die Hardware-Ansteuerung und bietet andererseits verschiedene Softwarerahmen für die Roboteransteuerung und die Verarbeitung der Videobilder. Miro basiert auf Corba [4], wodurch unterschiedliche Programmteile leicht auf verschiedene Rechner verteilt werden können, was für rechenintensive Aufgaben oder das Debugging sehr nützlich ist. Miro erfreut sich wachsender Beliebtheit, ist unter der GPL freigegeben und auf [5] erhältlich.

Flexible Miro-Module

Ein weiteres Projekt nutzt das Videoverarbeitungs-Framework von Miro und enthält die eigentliche Bildverarbeitung und Objekterkennung. Durch die Abspaltung in ein eigenes Projekt ist es auch hier leicht, die Komponenten in unterschiedlichen Anwendungsszenarien zu benutzen. Als letztes Modul folgt anschließend eine geeignete Software, um Fußball zu spielen. Hierzu gehören Teile wie die Weltmodellierung, die

Selbstlokalisierung und die einzelnen kleinen Bewegungssteuerungen (Behaviours) für das Anfahren des Balls, das Kicken und so weiter.

Ganz normale Probleme

Probleme in der Robotik haben oft leider wenig mit wissenschaftlichen Fragestellungen zu tun. Die Auswahl an Laptops etwa, die in die Roboter passen, ist begrenzt. Letztlich haben wir uns für Laptops aus IBMs X31-Reihe entschieden, also die kleinsten von IBM mit einem 12-Zoll-Display. Unsere Geräte sind mit einem Pentium 4 mit 1,4 GHz und Centrino ausgestattet, was ein guter Kompromiss zwischen Akkulaufzeit und Rechenleistung ist.

Damit handelt man sich aber auch die Probleme des Intel-WLAN-Moduls ein. Mit Hilfe des Ndiswrapper-Projekts [6] und der Windows-XP-Treiber lässt sich das Intel-WLAN-Modul betreiben. In der neuesten Version wurde zudem ein Fehler behoben, der die Kommunikation mit IP-Multicasting unmöglich machte. Dies war für uns deshalb so wichtig, weil einige Teile der Software auf IP-Multicasting aufbauen.

Leider scheint der Betrieb des Intel-WLAN-Moduls in Verbindung mit IP-Multicasting recht aufwändig zu sein, denn der Treiber verbrauchte jeweils einen ansehnlichen Teil der Ressourcen. Immerhin zwischen 20 und 40 Prozent, wenn Daten per IP-Multicasting ausgetauscht wurden. Die derzeit verfügbare Betaversion des nativen Open-Source-WLAN-Treibers von Intel [7] ist bisher auch keine wirkliche Alternative. Er beanspruchte bei unseren Tests noch mehr Rechenzeit. Da aber beinahe im Wochenrhythmus neue Versionen erscheinen, können wir wohl noch berechtigte Hoffnung auf Besserung haben.

Leistungshungriger CAN-Bus

Noch schlechter als bei WLAN war die Situation bei unserer CAN-Bus-Anbindung. Während die WLAN-Treiber nur dann Rechenzeit benötigen, wenn auch wirklich etwas gesendet wird, beansprucht der CAN-Treiber [8] permanent CPU-Zeit. Wir wandten uns dann direkt an den Entwickler des Treibers, der uns umgehend mit einer neuen Betaversion ausgeholfen hat. Dadurch konnten wir auch auf Kernel 2.6 umsteigen, bei den Vorrundenspielen in Paderborn benutzten wir Kernel 2.6.5-pre3-bk. Durch die Preemption-Patches, die der Kernel 2.6 enthält, konnten wir auch noch eine deutliche Steigerung der Systemperformance feststellen.

Die Firewire-Kameras werden, anders als Consumergeräte, nicht über die »libdv1394«, sondern über »libdc1394« respektive »libraw1394« angesprochen. Unter Kernel 2.4 hatten wir immer wieder damit Probleme, dass das System plötzlich und nicht reproduzierbar seine Mitarbeit verweigerte. Danach ließ sich nicht einmal mehr das Kernelmodul entladen, weshalb nur noch ein Neustart des Rechners half. Mit Kernel 2.6 scheinen diese Probleme praktisch nicht mehr oder zumindest deutlich seltener aufzutreten.

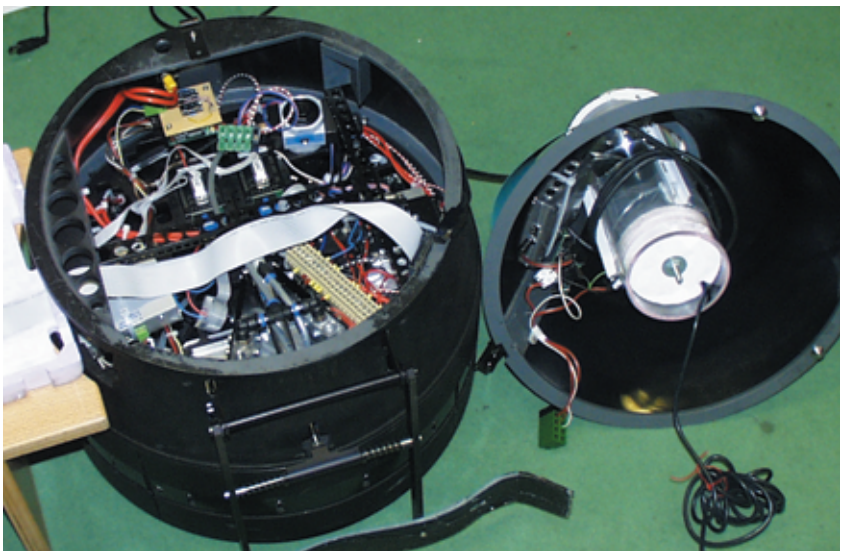

Abbildung 3: Ein Blick unter die Haube von Gonzales: Im Deckel (rechts) sind die Halterung und der Stellmotor für die Firewire-Kamera zu sehen, links der Robot-Körper von vorn. Trotz CAN-Bus und eines IBM-Notebooks als Steuerrechner ist die Verkabelung überaus komplex.

Ausblick

Nachdem wir jetzt die meisten technischen Probleme ausgemerzt haben, wollten wir bei der kommenden Weltmeisterschaften in Lissabon, die parallel zur „humanoiden“ Fußballweltmeisterschaft stattfindet, zeigen, was in unseren Robotern steckt. Leider können wir aus finanziellen Gründen dort nicht als komplettes Team antreten.

Da sich aber aus jeder Not eine Tugend machen lässt, planen wir ein gemischtes Team zusammen mit Mannschaften anderer Universitäten aufzustellen. Das wäre wohl das erste Mal, dass Roboter kooperieren, die nicht von Grund auf dafür konzipiert wurden. Auch daraus ergeben sich interessante wissenschaftliche Fragestellungen. *(mdö)*

Infos

[1] Robocup German Open 2004: [http://ais.gmd.de/GO/2004]
[2] Internationale Robocup-Homepage: [http://www.robocup.org/]
[3] Homepage der Ulm Sparrows: [http://www.sparrows.uni-ulm.de/]
[4] The Ace Corb (Corba): [http://www.theaceorb.com/]
[5] Middleware für Roboter (Miro): [http://smart.informatik.uni-ulm.de/Miro/]
[6] Projektseite Ndiswrapper: [http://ndiswrapper.sourceforge.net/]
[7] Centrino-WLAN-Treiber von Intel: [http://ipw2100.sourceforge.net/]
[8] CAN-Bus-Treiber für Linux: [http://www.peak-system.com/linux/]

Die Autoren

Gerd Mayer beschäftigt sich hauptsächlich mit der Bilderkennung und -Verarbeitung der Roboter, Hans Utz ist der Hauptentwickler von Miro, zudem sind beide die Coaches der Ulm Sparrows. Philipp Baer gehört ebenfalls zum Team und kümmert sich um das Wohlergehen der Roboter.