Realisierung einer Sprachsteuerung für Roboter an der HAW-Hamburg

2. April 2002

Eine Studienarbeit von:

Arne Dieckmann und Andrej Gossen

Hochschule für angewandte Wissenschaften
Berliner Tor 3
20099 Hamburg, Germany

http://www.informatik.haw-hamburg.de/~pioneer

Inhaltsverzeichnis

INHALTSVERZEICHNIS 2

1 EINLEITUNG 3

2 ALLGEMEINE AUFGABENSTELLUNG 6

2.1 LEISTUNGSUMFANG 6
2.2 VORRAUSSETZUNGEN 6
2.3 PERSÖNLICHE MOTIVATION 6

3 DESIGN UND ARCHITEKTUR 8

3.1 ANSATZPUNKTE 8
3.2 GRUNDDESIGN 10

4 REALISIERUNG 20

4.1 AUSWAHL DER SPRACHERKENNUNGSSOFTWARE 20
4.2 AUSWERTUNG DER GESPROCHENEN SÄTZE 20
4.3 SCHNITTSTELLE ZWISCHEN SPRACHAUSWERTUNG UND ROBOTER 22
4.4 SCHNITTSTELLE ZUM ROBOTER 24
4.5 PARSER – DER DENKER – 26
4.5.1 IMPLEMENTIERUNG 26
4.5.2 PROGRAMMIERUNG 35
4.5.3 GRUNDSÄTZLICHER ABLAUF 37

5 ZUSAMMENFASSUNG / RESÜMEE 39

ANHANG 41

A SPRACHUMFANG 41
B LITERATUR- UND QUELLENVERZEICHNIS 42
C ABBILDUNGSVERZEICHNIS 43
D LIZENZBEDINGUNGEN 44
E SYSTEMANFORDERUNGEN 44

1 Einleitung

In den letzten Jahren haben sich durch die angestiegene Rechnerleistung mehrere Möglichkeiten ergeben, die Kommunikation zwischen Mensch und Maschine in einem erweiterten Umfeld zu nutzen. War es bisher üblich, über die Tastatur oder die Maus Daten in den PC zu bringen, werden heute immer mehr neue Verfahren eingesetzt und weiterentwickelt.

Zum einen werden die natürlichen Kommunikationsmittel des Menschen ausgewertet. In vielen Firmen z.B. wird schon die Augenerkennung als Zugangskriterium eingesetzt.

Auch die Spracherkennung ist heutzutage soweit fortgeschritten, das sich schon sehr viele Industriezweige mit der Idee der Sprachanalyse auseinandersetzen. In der Fahrzeugherstellung werden kleinere Aufgaben wie das Fenster öffnen oder die Radiobedienung bereits der Stimme des Fahrers überlassen. Da Nebengeräusche die Erkennung der Sprache sehr schwierig machen, ist hier aber noch sehr viel Forschungsarbeit zu investieren.

Es gibt aber auch schon Forschungsversuche die mit den Hirnstrommessungen arbeiten. So soll es vielleicht irgendwann möglich sein, mit den Gedanken einen kompletten Brief zu schreiben oder eine Maschinen zu steuern [Wheelesley, MIT].

Da die Erkennungsrate der Spracherkennung schon sehr weit fortgeschritten ist und es schon diverse frei erwerbliche Produkte auf dem Markt gibt, haben wir uns entschlossen, uns mit einem Projekt auf dem Gebiet der Sprachanalyse zu befassen. Am Anfang unserer Studienarbeit steht eine bereits bestehende Arbeit eines Büroboten [Löh-

ler/Boldt] mit Hilfe eines Pioneer 1 Roboters [ActivMedia] an der HAW-Hamburg.

Abbildung 1: Pioneer 1

Unter diesen Vorbedingungen steht nun diese Studienarbeit, die das Ziel verfolgt, einen Roboter mit einer Sprachsteuerung zu entwickeln. Diese Studienarbeit wurde sowohl im praktischen, als auch im textuellen Teil in Teamarbeit durchgeführt. Obwohl bei gut funktionierenden Teams die Einzelleistung kaum heraus separiert werden kann, ist doch grob eine Aufteilung der folgenden Art gegeben:

Sprachverarbeitung: Dieckmann / Gossen
Server / Client: Dieckmann
Befehlsumsetzer: Gossen

In Kapitel 2 werden wir die allgemeine Aufgabenstellung vorstellen. Die Architektur und das Design der gesamten Aufgabe wird in Kapitel 3 näher beschrieben. Im folgenden Kapitel 4 gehen wir dann genauer auf die konkrete Realisierung und Implementierung ein. Abschließend wird das ganze Projekt noch einmal zusammen gefaßt und beurteilt.

2 Allgemeine Aufgabenstellung

Die erste Frage, die sich bei der Bearbeitung der Studienarbeit stellte, lautet:
„Mit welchen Mitteln der HAW-Hamburg ist es möglich, einen Roboter mit Hilfe von Sprache agieren zu lassen.“

2.1 Leistungsumfang

Der Roboter soll sich mittels Sprachkommandos orts- und rechnerunabhängig steuern und kontrollieren lassen. Weiterhin muß er sich soweit orientieren können, dass er sich selbstständig in seiner Welt zurecht finden kann. Als dritter Aspekt kommen dann noch Funktionen wie Objekte greifen hinzu.

2.2 Vorraussetzungen

Zum einen sollte es möglich sein, die Sprache über eine beliebige evtl. vorhandene Software in den PC zu bekommen, die dann von einer Schnittstelle weiterverarbeitet werden kann.
Zum anderen muß der Roboter in der Lage sein, sich selber in seiner Umgebung zurechtzufinden und eventuellen Hindernissen auszuweichen.

2.3 Persönliche Motivation

Die bereits vorhandene Erfahrung mit den Pioneer-Robotern [WPP Sommersemester 2001], jedoch nur auf der Basis der Saphira Programmierumgebung [Saphira] für die Implementierung unterschiedli-

cher Aufgaben, weckte in uns ein größeres Interesse das Gesamtsystem kennen zu lernen.
Die Funktionalität der Schnittstelle Roboter-Computer-Mensch hatte für uns zusätzlich den Reiz etwas "lebendig" agieren zu lassen und folglich die Resultate physisch sehen zu können.
Ebenso erhoffte wir uns die Aneignung von Erfahrung bei der Implementierung einer Schnittstelle, die für die Interaktion zwischen Roboter und Mensch dienen sollte. Dabei war es für uns ganz besonders wichtig, die Möglichkeiten und Problemfälle kennen zu lernen.

3 Design und Architektur

3.1 Ansatzpunkte

Um den Implementierungsaufwand, der nötig war, abschätzen zu können, mussten zuerst ein paar Vorüberlegungen gemacht werden. Vor allem sollte in dieser Phase des Projektes die grundsätliche Angehensweise herausgearbeitet werden.
Aus den ersten wagen Ideen sollten die sinnvollen und realiserbaren ausgewält und als die Grundlage des Projektes festgelegt werden.
Die erste Frage die wir uns bei den ersten Vorüberlegungen gestellt hatten war:
„wie bekommt man die Sprache in den Computer und was genau wurde gesagt“
Da die akkustische Sprach-Erfassung und Verarbeitung nicht der Bestandteil unserer Studienarbeit war, haben wir uns sofort für die Benutzung einer kommerziellen Lösung entschieden.
Weiter stellte sich die Frage:
„Was wollen wir mit der Information machen“
Diese Frage sollte mehr die Art der Steuerung die hinter der Implementierung steckte, beantworten. Unter anderem müsste hier klar gestellt werden, in wie weit die Information die Einwirkung in das Programmgefüge hat und wie wichtig diese ist.
Ebenso stellten wir uns die Frage:
„Was wird von der erfassten Information erwartet und wie wird diese weiterverarbeitet“.

Hier sollte eine grobe Struktur oder das Aussehen und die Art der Information erkannt und festgelegt werden. Dies war in unseren Augen erforderlich, denn der logische Hintergrund der allgemeinen Steuerung von Maschinen erfordert eine bestimmte Grundstruktur aus der die Kommandos gebildet werden können.
Als Beispiel sei hier ein Gespräch zwischen zwei Personen angegeben, der mit der Steuerung einer Maschine nichts zu tun hat und somit eine nicht relevante Information darstellt, die nicht weiter verarbeitet werden soll. Oder im Gegensatz dazu ein Satz der gezielt eine bestimmte Aktion bei der Maschine auszulösen versucht.
Die Frage der Weiterverarbeitung stellte uns vor die Problematik der konkreten Umsetzung des Gesprochenen auf die für die Maschine verständliche Kommandos.

Im weiteren sollten möglichst alle beteiligten Elemente des Projektes erkannt werden. Hier war uns wichtig die Grundstruktur zusammen zu tragen, die alle Projekt relevanten Hard- und Software Komponenten enthalten sollte.
Nach Möglichkeit sollte auch gleich die Interaktion zwischen den Komponenten heraus gearbeitet werden, um die Abhängigkeiten erkennen zu können.
Durch diese Ansatzpunkte wurden die ersten Grundsteine, eine Art Fundament gelegt auf dem die weitere Implementierung basieren sollte.

3.2 Grunddesign

Die ersten Vorüberlegungen führen unentwegt zu der Idee einer Komponenten Implementierung, wobei jede einzelne Komponente den jeweiligen Aspekt der Aufgabenstellung implementieren sollte.
So sind aus der logischen Sicht 3 Hauptkomponenten erkennbar:

- Spracherfassung
- Sprachverarbeitung
- Ausführung

Der Einsatz einer kommerziellen Lösung für die Spracherfassung und –erkennung zog eine Ausarbeitung von Informationsrückgabemöglichkeiten, die die Information liefern sollte, nach sich.
Da es nicht vorausgesetzt werden konnte, dass alle Benutzer die gleiche kommerzielle Produkte besitzen würden, sollte die Spracherfassungskomponente die Unterstützung unterschiedlichster kommerzieller Produkte erlauben, die nur einige wenige Bedingungen erfüllen sollten.

Als nächster Punkt in der Designentwicklung sollte die Sprachverarbeitungskomponente betrachtet werden.
Die Sprachverarbeitung als eine der wichtigsten Teilaspekte der Implementierung, die die tatsächliche Informationsverarbeitung implementiert, hat als Ziel die erfasste Sprache auf die Relevanz zu prüfen und daraus die sinnvolle Information zu selektieren.

Die Ausführungskomponente ist für die Zusammenstellung der Kommandos für die zu steuernde Maschine, in diesem Fall Pioneer-Roboter, zuständig.

Somit kann man die Bezüge durch folgendes Diagramm darstellen, selbst wenn es so simpel erscheint ist jedoch die grafische Darstellung einleuchtender.

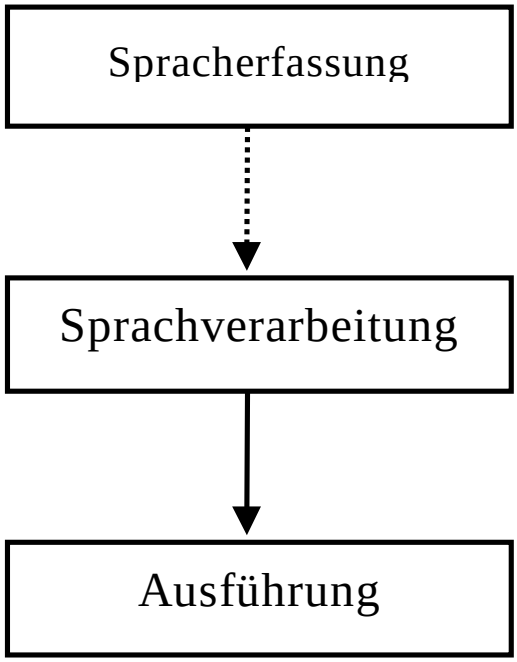

Abbildung 2: logischer Aufbau

Die strikte Abgrenzung der Spracherfassung von den anderen Komponenten hatte in unseren Augen zusätzlich den Vorteil einer allgemeineren Implementierung, die nicht nur einem bestimmten Zweckeinsatz folgt, sondern eine Austauschmöglichkeit erlaubt, wo die Sprache durch andere Eingabemöglichkeiten ersetzt werden könnte, z.B. um mal den futuristischen Fall anzunehmen, die Steuerung des Roboters mittels Gedanken oder auch einen Fall einer Steuerung mittels Gesten, diese Möglichkeit würde auch die Steuerung des Roboters einem stummen Menschen erlauben.

Das Grundkriterium bei der Wahl des Spracherkennungsprogramms war die sofortige Möglichkeit der Informationsangabe. Es sollten also keine zusätzlichen Steuerbefehle für das Spracherkenungsprogramm angegeben werden.

Als Beispiel könnte man sich folgendes Szenario vorstellen:

- ➔ „Datei öffnen“
- ➔ „Texteingabe“
- ➔ „ich möchte, dass du 20 cm nach vorne fährst“
- ➔ „Datei speichern/senden“

Wie man sieht, ist mit der relevanten Information „ich möchte, dass du 20 cm nach vorne fährst“ ein Zusatzaufwand notwendig, der eher die Kommunikation mit dem Programm als Hintergrund hat und nicht mit der Maschine. In unserer Lösung sollte dieser Overhead vermieden werden, dies setzt jedoch voraus dass andere Spracherkennungsprogramme ähnliche Möglichkeiten bieten sollten.

Als weiterer „Meilenstein“ in dem Projekt war die Sprachverarbeitung zu betrachten. Hier sollten die grundsätzliche Ansätze für die Sprachverarbeitung festgelegt und definiert werden.
Dabei sind wir auf zwei unterschiedliche Richtungen aufmerksam geworden:
Zum einen konnte man die Sprache semantisch analysieren und diese auf Richtigkeit und die Relevanz zu überprüfen um die Interaktion auf einem menschlichen Nevau zu halten.
z.B. man würde aus dem Sprachkontext, der auf unterschiedliche Weise das Gleiche Kommando festlegt, den Befehl herausselektieren:

- ➔ „Fahre bitte nach vorne“
- ➔ „Ich möchte, dass du fährst“
- ➔ „Gib gas !“

usw.

Die andere Möglichkeit wäre einen bestimmten Sprachumfang festzulegen, aus dem die gesprochene Sätze gebildet werden dürfen um so die Interaktion zwischen dem Mensch und Maschine zu implementieren.
Diese könnte man durch folgende Beispiele darstellen:

- ➔ „Fahre“
- ➔ „Fahre vorwärts“
- ➔ „vorwärtsfahren“

usw.

Diese Implementierung hat natürlich keine Definitionsfreiheit bei der Formulierung der Sprachkommandos und bietet keine menschenähnliche Kommunikation.
Andererseits ist die menschliche Mentalität jedoch noch nicht fähig die Maschinen, die ganz anders aussehen und von denen man weiß, dass diese „das Werk des Menschen“ sind, als gleichgestellte Kommunikationspartner zu sehen. Wobei hier wohl das Gefühl der „Fremdheit“ eher auf das „Andersaussehen“ zurückzuführen ist. Dieser Umstand wird sich wahrscheinlich erst dann ändern wenn die Manschinen und speziell Roboter, weil diese hauptsächlich die sprachliche Kommunkation Mensch-Maschine fördern, menschenähnlicher aussehen werden.

Wir haben uns im nachhinein für die zweite Möglichkeit entschieden, denn selbst wenn die „semantische Lösung“ der Idee einer „ungezwungenen“ Kommunikation mit einem Roboter zwar näher als die Befehlsähnliche kommt, mussten wir jedoch den Zusatzaufwand in der Implementierung berücksichtigen, denn der Umfang der Arbeit würde sich zusätzlich auf das Gebiet der „semantische Spachanalyse“ ausdehnen. Da jedoch die Bearbeitungszeit limitiert war, konnten wir in dem vorhandenen Zeitraum nur die zweite Möglichkeit vollständig implementieren.

Hierfür haben wir uns eine für unsere Zwecke in Absprache mit Chi Cuong Tat [Chi Cuong Tat] notwendigen Sprache [Sprachumfang] entwickelt.

In diesem Punkt hat sich zum ersten Mal die Komponentenimplementierung als positiv herausgestellt. Da die Sprachverarbeitung und Ausführung getrennt betrachtet werden wird es auch später immer noch möglich sein die „semantische Sprachanalyse“ in die Implementierung einzufügen.

Als letzte Komponente sollte die Ausführung betrachtet werden, hier sollten die Möglichkeiten der Hardwareansteuerung ausgelotet werden.
Da der Pioneer - Roboter mit einem Entwicklungspaket ausgeliefert wird, war die Schnittstelle zu der Hardware bereits gegeben und wir hatten auch vor diese zu benutzen.
Diese Entwicklungsumgebung heißt „Saphira“ [Saphira] und ist eine fertige Schnittstelle zu dem Pioneer - Roboter. In „Saphira“ besitzt man die Möglichkeit mittels bestimmter Kommandos den Roboter zu steuern, wobei diese in eine Kommandozeile eingegeben werden müssen. Ausserdem bietet die Umgebung die Möglichkeit, eigene Programme mittels einer internen Programmiersprache „Colbert“ [Saphira] zu erstellen.

Projektbezogen sollten speziell die Möglichkeiten und der Umfang der Verwendung der Entwicklungsumgebung in der Implementierung erarbeitet werden.
Da wir keine zu große Abhängigkeit von der Saphira - Umgebung haben wollten, haben wir uns entschieden nach Möglichkeit nur die Steuerkommandos für den Roboter umzusetzen und auf weitere Einbindung von z.B. der Darstellungsmöglichkeiten oder der Kontrolle über den grafischen UI zu verzichten.

Bereits am Anfang unserer Überlegungen hatten wir die Idee, die Steuerung auch ortsunabhängig zu implementieren, d.h. man sollte den Roboter von jedem Standort kontrollieren zu können.
Es gab zwei mögliche Wege die dies erreichen könnten.

- Der Roboter besitzt eigene Möglichkeiten der Spracherfassung und -verarbeitung und kann folglich an jedem Ort befehligt werden.
- Die Implementierung einer Netzwerkfähigkeit in der Anwendung z.B. Client-Server-Struktur.

Wir haben uns klar für die zweite Möglichkeit entschieden, die einige Vorteile mit sich brachte.

Für die erste Möglichkeit, nennen wir sie mal „Roboter-Implementierung“ musste vorausgesetzt werden, daß es bereits Hardware vorhanden ist auf der die gesamte Umgebung laufen kann. Da der Pioneer-Roboter keinen integrierten PC besitzt, ist diese Lösung nur dann möglich wenn z.B. einen Notebook auf dem Roboter montiert wird wo die gesamte Software läuft. Diese Lösung hatte zwar einen Vorteil, daß die gesamte Software nur einmal installiert werden müsste, jedoch überwog, in unseren Augen, der Nachteil einer geringeren Kontrolle über den Roboter. Hierzu ein Beispiel:

Nehmen wir mal an, man würde den Roboter zu einem Ziel schicken, nach einiger Zeit jedoch hat man es sich anders überlegt und möchte dem Roboter eine andere Aufgabe geben, so hat man in der „Roboter -

Implementierung“ keine Eingriffsmöglichkeiten solange der Roboter sich nicht in dem Radius der Spracherfassung und Erkennung befindet, einfach gesagt er muss den Benutzer hören können. Das bedeutet um das Kommando trotzdem abzugeben ist man gezwungen sich zu dem Roboter zu begeben, dieses kann jedoch nicht immer erwünscht sein.

Die zweite Möglichkeit, wir nennen es mal „Netzwerk - Implementierung“ hat den Vorteil der ständigen Kontrolle über den Roboter. Der Nachteil ist jedoch dabei die Software, die an jedem Standort installiert werden müsste, was manchmal mit der Anschaffung von Zusatzlizenzen verbunden sein kann und folglich Kosten nach sich zieht.

Eine andere „Netzwerk - Implementierung“ die am optimalsten das Problem lösen könnte, wäre die Möglichkeit der Zentralisierung der Software auf einem Steuercomputer, d.h. die Sprache wird an jedem Standort erfasst und dann direkt an das zentral installierte Spracherkennungsprogramm übergeben wo die weitere Verarbeitung läuft.
Dies könnte man sich am folgenden Beispiel vorstellen:

In letzter Zeit wird immer öfter die Rede von „intelligenten Häusern“ (z.B. [Heitmann]) geführt, darunter versteht man im groben ein Haus, das unterschiedliche Aufgaben selbständig übernimmt, z.B. das an/ausmachen von Licht, wenn man in das Zimmer rein oder raus geht. Oder das selbständige kontrollieren der vorhandenen Ressourcen ob z.B. der Heizölbestand ausreichend ist oder der Inhalt des Kühlschranks noch für ein Abendessen reicht.

Die bereits vorhandenen Implementierungen basieren öfter auf einem Zentralen System, daß die gesamte Informationsverarbeitung durchführt.
So könnte man sich in naher Zukunft Haus-Roboter vorstellen, die nun auch etwas komplexere Aufgaben übernehmen können.
Man würde natürlich den Roboter von jedem Zimmer aus steuern wollen. Wenn man z.B. den Roboter in ein Zimmer holen wollte, würde der Hausherr höchstwahrscheinlich ungerne zu dem Roboter gehen, der vielleicht mit der Wäsche im Keller beschäftigt ist, so wäre die Lösung in diesem Falle, man integriert in jedem Zimmer Mikrophone oder andere Eingabemedien, die die Information erfassen und an ein zentrales System weiterleiten, von wo aus das Kommando an den Roboter geschickt wird, dieser möge sich zu dem Hausherren begeben und auf dem Weg ein Bier mitnehmen.
Diese Möglichkeit wurde jedoch unsererseits nicht weiter betrachtet, da der Umfang der Arbeit wiederum die Zeitgrenzen überstieg.

Letztendlich konnte man alle Komponenten zusammenfassen und in einem Diagramm darstellen, das in der letzten Fassung folgendes Aussehen hat:

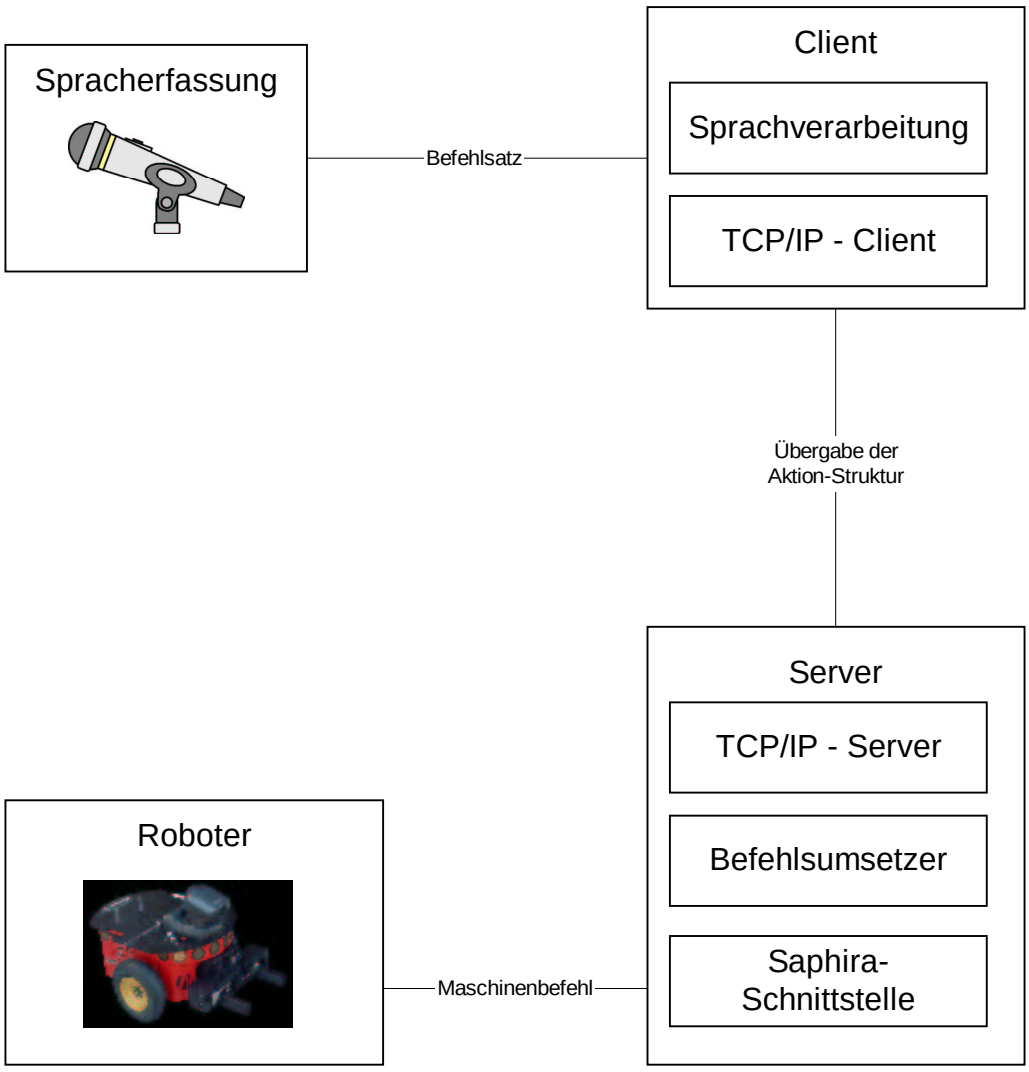

Abbildung 3: Komponentendiagramm

4 Realisierung

4.1 Auswahl der Spracherkennungssoftware

Da das System ja darauf ausgelegt sein soll, die Befehle per Sprache aufzunehmen haben wir uns erst einmal über verschiedene Möglichkeiten informiert, die Sprache in den Computer zu bekommen.

Zum einen wäre es möglich gewesen, die Sprache über die SDKs von verschiedenen Herstellern von Spracherkennungssoftware direkt einzubinden. Damit wären wir aber zu sehr an ein bestimmtes Produkt gebunden. Wir haben uns deshalb dafür entschieden, dass wir eine Spracherkennungssoftware in eine von uns geschriebene Sprachzeile schreiben lassen und diese dann auswerten. Somit kann jede beliebige Software genommen werden, die dann ihren Fokus auf diese Eingabezeile setzen kann. Für unsere Versuche haben wir uns für Dragon Naturally Speaking in der Version 5 entschieden.

4.2 Auswertung der gesprochenen Sätze

Um die gesprochenen Sätze auszuwerten, gibt es nun verschiedene Methoden. Zum einen kann man den Satz auf die richtige Semantik und Syntax untersuchen, was aber den Programmieraufwand für unsere Studienarbeit bei weitem überschritten hätte[1]. Zum anderen gibt es noch das Verfahren des „Keyword Matching“. Hierbei werden dann alle relevanten Worte über einfaches vergleichen aus dem Eingabesatz

[1] Einen Ansatz für geschriebene Sprache stellt Chi Cuong Tat vor. [Chi Cuong Tat]

herausgefiltert. Diese Methode ist sehr einfach zu realisieren, birgt aber auch einen sehr großen Nachteil in sich. Die beiden folgenden Sätze könnten z.B. nicht unterschieden werden:

fahre langsam vorwärts
fahre nicht langsam vorwärts

In beiden Fällen würden die drei Schlüsselwörter „fahre, langsam und vorwärts“ sofort erkannt werden und das Wort „nicht“ wäre nicht mehr relevant.
Sobald nun ein von uns erlaubter Satz in der Eingabe als gültig erkannt wird, wird ein vordefinierter Aktionsstring in ein File geschrieben und die Eingabezeile wird wieder gelöscht. Dieses File wird nun vom Client wieder eingelesen und weiterverarbeitet.
Um die Verwechslung von Wörtern möglichst gering zu halten, haben wir die Bibliothek von Naturally Speaking an unsere Arbeit angepasst [Sprachumfang], so dass nur noch die für uns relevanten Wörter erkannt werden können. Dieses ist auch sehr hilfreich gewesen, wenn die Umgebungslautstärke etwas höher war. So konnten wir unsere Sätze auch verarbeiten, wenn sich nebenbei andere Leute unterhalten haben oder die Demonstration in einer Halle mit Echoeffekten stattgefunden hat.

Abbildung 4: Sprachverarbeitung „Talk“

4.3 Schnittstelle zwischen Sprachauswertung und Roboter

In unserer Studienarbeit gehen wir ja davon aus, daß wir die gesprochenen Sätze von jedem beliebigen Rechner aus absetzen können. Für diese Zwecke haben wir uns dazu entschieden, eine einfache TCP Client – Server Architektur zu verwenden.

Unser Client ließt dabei das von der Spracherkennung angelegte File aus, und schickt es ohne es weiter zu anlysieren an den Server weiter. Die Struktur des Files hat dabei folgendes Format:

commandclass
action
speed
quantity
direction
distance
target
degree
object
do
color
room

Sobald nun ein neuer Befehl ausgelesen ist, wird dieser im linken Fenster des Clients angezeigt. Gleichzeitig wird dieser Befehl über die bestehende Verbindung zum Server geschickt, wo dann die Verarbeitung des Strings übernommen wird.

Im rechten Fenster wird der aktuelle Status angezeigt, der vom Server aus übermittelt wurde. Hier sind unter anderem folgende Meldungen vorzufinden:

FREE, Batteriestand, Zugriffsrecht, Position usw.

Über den Button Setup lassen sich die wichtigsten Dinge einstellen, wie Port-Nummer, IP des Servers und der Name des Files, welches von der Spracherkennung erzeugt wurde.

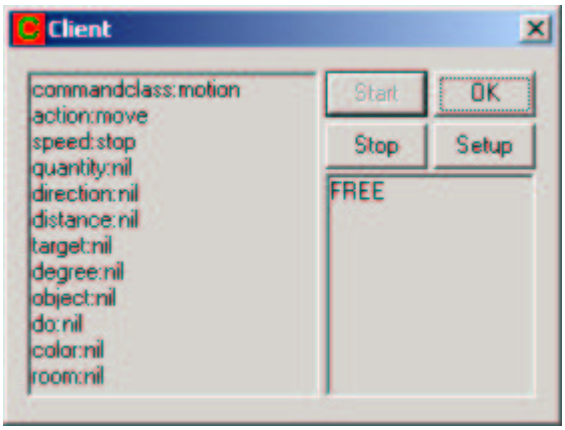

Abbildung 5: Client

4.4 Schnittstelle zum Roboter

Die von der Spracherkennung erzeugten und von den Clients verschickten Sätze werden an einer zentralen Stelle im Netzwerk verarbeitet.
Dieser Server basiert auf einem Web-Server, der auf einem beliebig einstellbaren Port auf die Sätze der Clients wartet. Die Antworten des Servers an die Clients werden ebenso an einen eigenen Webserver bei den Clients geschickt. Der Port ist hier ebenso beliebig einzustellen. In unserem Aufbau haben wir für den Server den Port 10000 und für die Clients den Port 10001 benutzt.

Zuerst wird der Server auf dem Rechner gestartet, an dem der Pioneer 1 [ActivMedia] per seriellem Funkmodem angeschlossen ist. Dieser Vorgang dauert ca. 3 Sekunden, bis alle Funktionen initialisiert sind. Beim Connecten des Robots werden hierbei die Sonars eingeschaltet und weitere Behaviors gestartet.

Die Clients müssen sich nun zuerst an dem zentralen Server anmelden, bevor Sie eine Befehlsstruktur an den Server absetzen können. Sobald sich der Client anmeldet, erscheint die IP-Adresse in dem rechten oberen Fenster des Servers. Diese beleibt nun solange hier stehen, bis sich dieser Client wieder abmeldet.

Der Server wartet nun auf einen String von einem beliebigen Client, der eine bestimmte Befehlsstruktur enthält. Sobald eine neue Befehls-

struktur ankommt, wird diese in dem linken Fenster des Servers angezeigt.

Das rechte untere Fenster gibt den aktuellen Status des Servers wieder. Der Status FREE bedeutet z.B. dass der Server zur Zeit auf einen Befehl wartet und nicht mit der Abarbeitung eines Befehls beschäftigt ist. Weitere Meldungen sind Batteriestand, Zugriffsrecht erteilt, aktuelle Position usw.

Die Abarbeitung des Befehlsstrings erfolgt über einen selbstgeschriebenen Parser, der in einem eigenen Task läuft, der weiter unten genau beschrieben wird.
Auch hier lassen sich über den Setup Button wieder die wichtigsten Funktionen einstellen, wie Port Nummer, Saphira [ActivMedia] einblenden und Multiuser Betrieb.

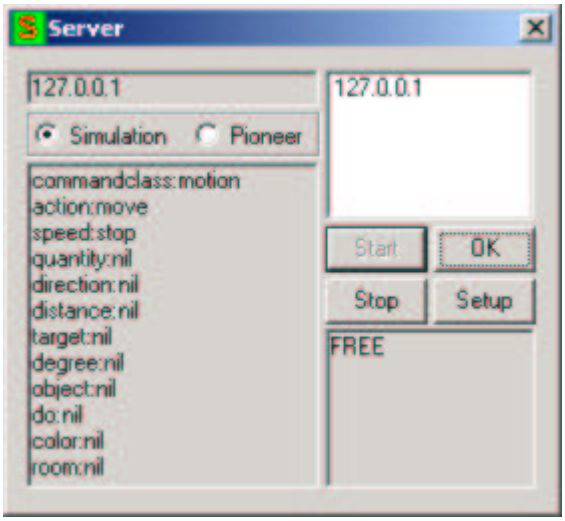

Abbildung 6: Server

4.5 Parser – der Denker –

Wie steuert man einen Roboter ?
Auf welche Art und Weise kann man es machen ?
Warum überhaupt eine Steuerung ?
Wie komplex soll die Steuerung sein ?
Was bringt eine neue Steuerung im Vergleich zu der alten ?

Das wären die ersten wichtigen Fragen, wenn es darum ginge eine neue Schnittstelle auf der Basis einer alten implementieren zu müssen.
Um diese Fragen zu beantworten und die Erkenntnis und Erfahrung dabei sammeln zu können, wurde ein Projekt durchgeführt, der sich mit der Aufgabe der Implementierung einer Schnittstelle auf der Basis der Pioneer-Roboter von ActiveMedia beschäftigte.
Die an der HAW-Hamburg eingesetzten Pioneer-Roboter bieten mit der Hardware (die eigentlichen Roboter) ebenso eine Softwareschnittstelle, die die Implementierung von komplexen Aufgabenstellungen erlaubt.
Diese Plattform hat den Vorteil sehr vielseitig zu sein und vielfältige Möglichkeiten bieten zu können, die für größere Projektaufgaben von Vorteil sind.

4.5.1 Implementierung

Durch die klar gegebene Punkte der Aktionsmöglichkeiten des Roboters, wurde, durch die Festlegung der jeweiligen Befehle, eine Struk-

tur zusammengestellt, die eine möglichst universelle Schnittstelle bilden sollte, in der alle festgelegte Befehle codiert werden könnten. Unter anderem wurden folgende Befehle berücksichtigt:

- Einfaches fahren mit Richtungsangabe,
- Fahren einer bestimmten Distanz mit Richtungsangabe,
- Drehen um einen bestimmten Winkel mit Richtungsangabe,
- Anhalten der Fahrtbewegung,
- Greifer anheben,
- Greifer runter lassen,
- Greifer schließen,
- Greifer öffnen,
- Festlegung eines Objektes, das z.B. zu transportieren ist,
- Festlegung der Farbe des zu transportierenden Objektes,
- Festlegung der Startposition des Roboters,
- Fahren innerhalb einer Umgebungswelt, die Mittels einer Karte festgelegt ist,
- Abholen eines Objektes von einer bestimmten Position,
- Hinbringen eines Objektes zu einer bestimmten Position,

In der nachfolgenden Tabelle sind die gesamten Möglichkeiten dargestellt. (Erstellt gemeinsam mit Chi Cuong Tat [Chi Cuong Tat])

Kommando-klasse	Aktion	Parameter	Attribute
MOTION	move	speed	fast, slow, default, stop
		quantity	more, less
		direction	forward, backward
		target	number (4 ziffrige Nummer)
		distance	number (in Zentimetern)
MOTION	turn	direction	left, right,
		degree	number (0 – 360)
MOTION	gripper	direction	up, down
		do	close, open
MOTION	transfer	object	cup(z.Z. nur Becher)
		target	number
MOTION	get	object	cup (z.Z. nur Becher)
		target	number
		color	red, yellow, blue
SET	location	room	number
REQUEST	status	-------	--------

Abbildung 7: Aktionstabelle

Als Endresultat nahm die Schnittstelle folgende Struktur an:

commandclass
action
speed
quantity
direction
distance
target
degree
object
do
color
room

Wobei jedes einzelne Teil davon mit fest definierten Attributen besetzt werden kann.

Im einzelnen sind es:

➔ **commandclass**:

Hier wird die Kommandoklasse festgelegt, die für eine bestimmte Aktionsart steht.
Bei dieser Implementierung sind 3 Kommandoklassen möglich:

- **motion**
- **set**
- **request**

Kommandoklasse **‚motion**' steht für eine allgemeine Bewegung des Roboters oder seiner Komponenten und schließt jede mögliche Art davon ein.

Kommandoklasse ‚**set**' steht für eine Art Initialisierung des Roboters in der Umgebung, mit dieser Kommandoklasse wird dem Roboter eine Startposition in der „Welt" zugeordnet. Dies hat den Vorteil, mit dem Roboter von jeder möglichen Position starten zu können.
Kommandoklasse ‚**request**'ist für die Statusabfrage des Roboters zuständig. Diese findet Anwendung z.B. bei der Abfrage des aktuellen Akkustatus oder der aktuellen Position des Roboters.

➔ **action:**

Hiermit werden die jeweilige Aktionen des Roboters festgelegt.
Folgende Aktionen sind möglich:

move:
hiermit ist Bewegung des Roboters gekennzeichnet, wobei hierdurch nur die Fahrtbewegung des Roboters angegeben wird.

turn:
hiermit wird die Drehbewegung des Roboters gekennzeichnet.

gripper:
hierdurch werden die Greifer des Roboters angesprochen.

transfer:
hiermit wird das Ziel einer Transportaufgabe festgelegt;
bei der Befehlsart „bringe Objekt zum Punkt A"

get:
hiermit wird die Position des Transportobjektes, von der es abzuholen ist, festgelegt.
Ein Befehlsbeispiel wäre in diesem Falle: „Hole Objekt von Punkt B ab“

room:
diese Aktionskennzeichnung ist der set-Kommandoklasse zugeordnet und kennzeichnet die Startposition „in der Welt“ von der der Roboter startet. Die Bezeichnung ‚room’ wurde entsprechend der Konvention in der Saphira-Umgebung gewählt, in der die Umgebung durch bestimmte Artefakte beschrieben werden kann, z.B. Türen, Korridore, Durchgänge.

Im weiteren wurden für die unterschiedliche Aktionen die möglichen Parameter, die eine festdefinierte Aktion angeben, festgelegt.

Im einzelnen wären es dann:

move

Aktion ‘move’ mit Parametern:

speed:
Angabe der Geschwindigkeit, wobei hier nur drei Mögliche Werte definiert wurden:

- normal
- fast

- slow

Diesen Angaben sind bestimmte Geschwindigkeitswerte zugeordnet und sind somit fest definiert.

quantity:
mit diesem Parameter sollte die Verstärkung erzeugt werden,
wurde in der eigenen Implementierung jedoch nicht benutzt, da die Geschwindigkeitsangabe sich nur auf 3 Parameter beschränkt.

direction:
dieser Parameter definiert die Fahrtrichtung des Roboters hier sind nur 2 Angaben möglich:

- forward (Vorwärtsfahrt)
- backward (Rückwärtsfahrt)

target:
hiermit wird das Fahrziel des Roboters festgelegt. Dieses Ziel kennzeichnet ein Artefakt, das sich in der aktuell geladener Umgebungswelt befindet. Dies kann z.B. der Raum 1181 im 11-ten Stock des HAW – Hochhauses, Berliner Tor 3 sein. (Siehe Studienarbeit „Bürobote“ von Florian Boldt und Hauke Löhler [Löhler / Boldt])

distance:
mit diesem Parameter wird eine Fahrtdistanz angegeben, die Angabe wird in cm akzeptiert.

turn

Turn besitzt folgende Parameter:

direction:
hiermit wird die Drehrichtung des Roboters angegeben, es sind nur zwei Werte möglich

- left (Linksdrehen)
- right (Rechtsdrehen)

degree:
dieser Parameter gibt eine Gradanzahl an, die gedreht werden soll. Möglich sind hier Werte von 0 bis 360.

gripper

gripper besitzt folgende Parameter:

direction:
hiermit wird die vertikale Bewegungsrichtung der Greifer angegeben, möglich sind hier:

- up (hochfahren)
- down (runterfahren)

do:
dieser Parameter gibt an, ob die Greifer geöffnet oder geschlossen werden sollten und hat folgende Werte:

- open (öffnen)

- close (schließen)

transfer

transfer besitzt folgende Parameter

object:
mit ‚object' wird ein Objekt festgelegt, das z.B. transportiert werden soll. Dies kann z.B. ein Becher sein, ein typisches Transportobjekt, das bereits bei vielen Projekten benutzt wurde.

target:
mit Target gibt man das Ziel des Transports an, bei dieser Arbeit wurde festgelegt, dass die Angabe eine 4-Stellige Nummer, die einen Raum definiert, sein sollte.

get

‚get' besitzt folgende Parameter

object:
mit ‚object' wird ein Objekt festgelegt, das von einer Position abgeholt werden soll.

target:

mit Target gibt man das Ziel das Objekt abzuholen ist, bei dieser Arbeit wurde festgelegt, dass die Angabe ebenso wie auch bei ‚transfer' eine 4-Stellige Nummer, die einen Raum definiert, sein sollte.

room

Diese Aktion besitzt ein Parameter:

target:
mit Target gibt man die Startposition des Roboters in der Umgebungswelt an. z.B. Raum 1180, würde hier bedeuten, dass der Roboter von der Tür des Raumes 1180 startet.

4.5.2 Programmierung

Die bisherige Beschreibung wurde programmtechnisch in einer Struktur definiert und ist auf folgende Art aufgebaut:

```
struct ActionStruct
{
        int commandclass;
        int action;
        int speed;
        int quantity;
        int direction;
        int distance;
        int target;
        int degree;
        int object;
        int do_gripper;
        int color;
        int room;
}PioAction;
```

Jeder Parameter gibt je nach dem entweder eine Kodierte oder eine echte Information wieder. Die Kodierten Parameter sind hier:

```
int commandclass;
int action;
int speed;
int quantity;
int direction;
int object;
int do_gripper;
int color;
```

diese kodieren die bisher beschriebenen Parameter mittels Integer-Werte.
Hier ein kleiner Auszug aus der Headerdatei:

```
// COMMANDCLASS-Constants
#define P_MOTION 100
…
// ACTION-Constants
#define P_MOVE 1000
…
// SPEED-Constants
#define P_FAST 2000
….
```

Es ist somit jeder wörtlichen Angabe ein bestimmter Integer Wert zugeordnet.
Die Wahl der Codierung wurde aufgrund der Einfachheit bei der Programmierung und besonders aufgrund der Wert - Vergleiche, die aus Laufzeit technischen Aspekten auch ein Vorteil haben könnten, ausgewählt. (Stringvergleich gegenüber dem Vergleich der Integerwerte)

Die nachfolgenden Parameter geben echte Informationen wieder, da diese mittels Integer-Zahlen dargestellt werden können:

```
int distance;
int target;
int degree;
int room;
```

Beispielsweise würde die Distanz hier direkt angegeben

```
PioAction.distance = 100;  // hier wird die Distanz auf
                           // 100 cm festgesetzt
```

4.5.3 Grundsätzlicher Ablauf

Der grundsätzlicher Ablauf des Programms, gliedert sich im groben in zwei Teile:

- Bestimmung aller Parameter und damit die Zusammenstellung der Struktur, hier wird jeder Parameter, unabhängig von seinem logischen Zusammenhang mit der gewünschter Aktion, übernommen.
- Die Zusammenstellung des/der Befehls/Befehle und deren Ausführung

Der erste Teil „die Vorbereitung“ wird mittels Funktion initStructActionPio()

umgesetzt hier wird der Aktionsstring nach den markanten Elementen der Struktur durchsucht und je nach dem entweder mit einem entsprechenden kodierten oder mit dem echten Wert belegt.

Nach der Initialisierung der Struktur wird die Kommandoklasse abgefragt, die die festgelegte Aktionsart angibt.
Entsprechend der Funktionsart wird dann die entsprechende Aktion aufgerufen, die die weitere Unterscheidung durchführt.
Am Ende der Bearbeitung befindet man sich an der Stelle, die einem bestimmten Befehl zugeordnet ist und folglich dem interpretiertem Befehl entspricht, das dann ausgeführt wird.

Grundsätzlich, kann die Befehlsinterpretation wie ein „Baumabstieg“ bezeichnet werden.
Der Aufbau der entsprechenden Befehle wurde aus der Saphira – Dokumentation ermittelt.
So z.B. der Befehl für eine einfache Bewegung auf bestimmte Distanz

....

```
sfSetPosition(PioAction.distance*10);       // Sende den Befehl an den Roboter
sfPause(1000);                              // warte bis gesendet
while(!sfDonePosition(-PioAction.distance*10))   // solange die Distanz nicht
                                                 // erreicht
{
        sfPause(100);                       // warte 100 ms
};
```

....

Für die weitere Implementierung sei hier auf den Programmcode verwiesen.

5 Zusammenfassung / Resümee

Die bisher geplante Aufgabenstellung konnte im ganzen komplett gelöst werden. Da jedoch für das Projekt der „Pioneer 1“ Roboter [ActiveMedia] verwendet wurde, der keine Kamera vorweist, konnte der Teil mit der automatischen Kamerasuche nicht implementiert werden.
Diese Aufgabenstellung kann im Rahmen eines WPP - Projektes oder einer Studienarbeit durchgeführt werden, wobei in diesem Fall zusätzlich der Kompatibilitätsaspekt der implementierten Schnittstelle mit dem „Pioneer 2“ Roboter [ActiveMedia] ebenso von Bedeutung wäre, da dieses unsererseits nicht getestet wurde.
Weiterhin ist es nicht ausgeschlossen, dass die bisherige Implementierung nicht universell genug ist und weiterer Erweiterungen bedarf, da unsererseits die Festlegung der Aktionen, wie bereits erwähnt, nur Aufgrund älterer Projekte durchgeführt wurde.
Ebenso wurde die objektorientierte Implementierung der Aufgabenstellung aufgrund der Einfachheit der Möglichkeiten nicht in Betracht gezogen, kann jedoch durchgeführt werden, wenn dies im weiteren von essentieller Bedeutung werden würde.

Wir sind von unserer Implementierung überzeugt, selbst unter der Annahme, dass man im nachhinein die Probleme anders lösen würde, oder den einen oder anderen Punkt auf andere Weise betrachtet hätte. Den Vorteil sehen wir mehr in der globalen Nützlichkeit, weil dieses Projekt zum ersten Mal an der HAW - Hamburg durchgeführt wurde.

Im weiteren wäre es in unseren Augen vom großen Vorteil, die Spracherfassung universeller zu machen, da die kommerziellen Produkte unterschiedlichen Benutzern noch nicht die Möglichkeit bieten, den Roboter zu befehligen. Diese Feststellung beruht auf der Tatsache, dass nur bestimmte Benutzer den Roboter steuern können, die das Spracherfassungsprogramm auf sich abgestimmt haben. Dieses war der Grund, der die Durchführung weiterer Ergonomiestudien verhinderte, da die Lernphase mehr als 15 Minuten in Anspruch nimmt. Das war aus unserer Sicht keinem aussen stehenden Benutzer zuzumuten.

Abschließend möchten wir sagen, dass wir an diesem Projekt sehr gern gearbeitet und dabei viel gelernt haben. Hierbei möchten wir auch nochmal unseren Dank an Prof. Dr. rer. nat. Kai von Luck, Dipl.-Ing. Gerhard Oelker und Chi Cuong Tat aussprechen.

Anhang

A Sprachumfang

Fahrbefehle

- fahre langsam/normal/schnell vorwärts/rückwärts
- fahre vorwärts/rückwärts
- fahre *distanz* langsam/normal/schnell vorwärts/rückwärts
  (distanz: Entfernung in cm)
- fahre *raumnummer* raum
  (raumnummer: Nummer eines Raumes, zweistellig z.B. 1180 → 80)
- stop

Drehbefehle

- dreh *grad* links/rechts
  (grad: Drehwinkel in Grad)

Greiferbefehle

- greifer hoch/runter

Sonstige

- status
- setze *raumnummer* raum
  (raumnummer: Nummer eines Raumes, zweistellig z.B. 1180 → 80)
- kontrolle an/aus

B Literatur- und Quellenverzeichnis

- [ActiveMedia] ActivMedia Robotics
  www.activemedia.com
- [Löhler / Boldt] Studienarbeit: „Bürobote“
  www.informatik.haw-hamburg.de/~pioneer/Projekte/projekte.html
- [WPP Sommersemester 2001] Kommunikationsserver für Pioneerroboter
  www.informatik.haw-hamburg.de/~pioneer/Projekte/WPP-SS01
- [Saphira] Beginner’s Guide to Programming the Pioneer Robot in Saphira
  www.cs.smith.edu/~ksinclai/Murphy/Saphira_tut.html
  Saphira Robot Control System
  www.ai.sri.com/~konolige/saphira/
- [Chi Cuong Tat] Entwicklung und Aufbau einer Sprachsteuerungsschnittstelle für den Pioneer Roboter
  www.informatik.haw-hamburg.de/~robots/papers.html
- [Wheelesley, MIT] Wheelesley: Development of a Robotic Wheelchair System
  www.ai.mit.edu/people/holly/wheelesley/
- [Heitmann] Das Internet Haus
  www.cpt.haw-hamburg.de/~heitmann

C Abbildungsverzeichnis

Abbildung 1: Pioneer 1 4
Abbildung 3: Komponentendiagramm 19
Abbildung 4: Sprachverarbeitung „Talk“ 21
Abbildung 5: Client 23
Abbildung 6: Server 25
Abbildung 7: Aktionstabelle 28

D Lizenzbedingungen

Um die hier verwendete „Saphira“ - Software zu benutzen, bitte unbedingt die bei ActivMedia ® erhältlichen Lizenzbedingungen beachten.

E Systemanforderungen

Aus Kompatibilitätsgründen ist die von uns entwickelte Software nur unter dem Betriebssystem „Microsft Windows 2000“ ® lauffähig.