Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Spielerkoordination in RoboCup-Fußballspielen mittels gesprochener Sprache

Bachelorarbeit
im Arbeitsbereich Natürlichsprachliche Systeme, NATS
Prof. Dr.-Ing. Wolfgang Menzel

Department Informatik
MIN-Fakultät
Universität Hamburg

vorgelegt von
Maike Paetzel
am
10. Juni 2013

Gutachter: Prof. Dr.-Ing. Wolfgang Menzel
Dr. Werner Hansmann

Maike Paetzel
Matrikelnummer: 6087233
Wurmsweg 1
20535 Hamburg

Zusammenfassung

In der RoboCup Humanoid Kid Size League wird das gesamte Datenvolumen zwischen den Robotern über ein drahtloses Netzwerk ausgetauscht, obwohl diese Technik insbesondere während Turnieren sehr störanfällig ist und keine menschenähnliche Kommunikationsform darstellt.

Die vorliegende Arbeit zeigt einen Ansatz auf, die teaminterne Roboter-Roboter-Kommunikation auf ein natürlichsprachliches System umzustellen. Es wird dabei zunächst die Hardware des DARwInOP-Roboters vorgestellt und die Eignung für Kommunikation durch Sprache analysiert. Dazu werden auch Möglichkeiten der Hardwareerweiterung aufgezeigt, welche die Ergebnisse in diesem Bereich deutlich verbessern können. Weiterhin werden eine Strategie für die Dialogmodellierung und den Nachrichtenaustausch entwickelt und verschiedene Ansätze der Sprachsynthese sowie Sprachanalyse aufgezeigt. Es wird dabei auch berücksichtigt, in wie weit sich eine solche Architektur in bestehende Softwaresysteme eingliedern lässt, ohne dass eine große Umstrukturierung des Gesamtsystems nötig wird.

Das zentrale Element der Arbeit ist die Implementation eines Systems zur Roboter-Roboter-Kommunikation basierend auf den vorgestellten Modellen zur Synthese und Analyse von Sprache. Die im Verlauf der Bearbeitung entstandene Software bietet ein System, das eigenständig Sounddateien einliest, mittels einer Methode zur Detektion von Sprachbeginn und Sprachende Wörter findet und diese auf ihren Inhalt hin analysiert. An Hand von Unterschieden in den Frequenzen kann eine eingelesene Sounddatei automatisch einem Roboter zugeordnet werden. Die Qualität der Implementation wird im Rahmen dieser Arbeit durch Vergleich mit einem zuvor vorgestellten Anforderungskatalog untersucht.

Den Schluss bildet ein Ausblick auf weitere mögliche Forschungsansätze, die sich aus dieser Arbeit ergeben können.

Inhaltsverzeichnis

Kapitel 1

Motivation

Die Spielerkoordination in RoboCup-Fußballspielen ist eine sehr komplexe Aufgabe: Jeder Roboter erstellt sein eigenes lokales Weltbild, in dem er seine eigene Position, die Position des Balls und der Tore auf dem Spielfeld kennt. Daraus kann ein Verhalten abgeleitet werden, das den Roboter bis zum Ball laufen und ein Tor schießen lässt. Der Roboter ist jedoch nicht allein auf dem Feld, da zu einem Team bis zu drei Roboter gehören können. Sobald ein zweiter Roboter des eigenen Teams mit seinem eigenen lokalen Weltbild hinzukommt, ist eine Kommunikation zwischen den beiden wünschenswert. So kann verhindert werden, dass sich die Roboter gegenseitig behindern, während sie zum Ball laufen, es können aber auch Unsicherheiten in der Balllokalisation ausgeglichen werden. Sogar komplizierte Spielzüge wie ein Pass um einen gegnerischen Roboter herum sind denkbar und auch bereits in vielen Ligen realisiert worden. Voraussetzung hierfür ist ein störungsfreies Kommunikationssystem. Im RoboCup wird den Regeln gemäß W-LAN eingesetzt. In der Praxis zeigt sich aber, dass während des Turniers keinesfalls davon auszugehen ist, dass aktuelle Daten ohne Verzögerung versendet werden können. In Ermangelung einer anderen Kommunikationsmöglichkeit setzen die Teams daher auf Einzelentscheidungen der Roboter, obwohl dies meist zu keinem optimalen Ergebnis führt.

Da das Fußballspielen im RoboCup nicht Selbstzweck ist, sondern als Grundlage für die Forschung in anderen Bereichen dienen soll, ist auch ein Blick über die Domäne RoboCup Soccer hinaus lohnenswert [3]. Schon heute erhalten Roboter einen ersten Einzug in den Alltag der Menschen, beispielsweise in Form von intelligenten Staubsaugern. Damit die Roboter weitere Aufgaben im Haushalt übernehmen und auch einer informatikfernen Gruppe von Menschen zur Verfügung gestellt werden können, ist eine Kommunikation über natürliche Sprache unerlässlich. Roboter sollen den Menschen unterstützen und das können sie nur, wenn der Nachrichtenaustausch für den Menschen so natürlich wie möglich abläuft. Auch in einer häuslichen Umgebung ist der Einsatz einer Roboter-Roboter-Kommunikation denkbar, denn der Mensch fürchtet sich grundsätzlich vor einer Aktion von Maschinen, die für ihn nicht nachvollziehbar ist [18][25]. Kommunizieren nun verschiedene intelligente Systeme im unmittelbaren Umfeld der Menschen über einen Kommunikationsweg, der sich nur schwer überwachen lässt, stärkt dies das Vertrauen der

Menschen in die Maschinen nicht. Können sie die Interaktion jedoch nachvollziehen, da diese sich über natürliche Sprache abspielt, ist das Verhalten der Roboter leichter verständlich und wird weniger abgelehnt.

Das Ziel dieser Arbeit ist daher, eine alternative Spielerkoordination in RoboCup-Fußballspielen basierend auf natürlicher Sprache zu entwickeln, um eine Unabhängigkeit vom W-LAN-Netzwerk zu erreichen und einen weiteren Schritt in Richtung menschenähnlichem Fußballspiel zu gehen. Darüber hinaus soll ein erster Ansatz für eine Maschine-Maschine-Kommunikation und -Kooperation geliefert werden, die für den Menschen nachvollziehbar ist und so eine größere Akzeptanz von Servicerobotern ermöglicht.

Kapitel 2

Die Domäne RoboCup

Der RoboCup ist ein internationaler Wettbewerb mit dem Ziel, den Austausch und die Vergleichbarkeit in der Forschung an humanoiden Robotern zu fördern. In unterschiedlichen Ligen mit klar definiertem Regelwerk treten die Roboter gegeneinander an.

Die Domäne dieser Bachelorarbeit ist die Humanoid Soccer League, in der bis zu drei Roboter, ein Torwart und zwei Feldspieler pro Team, gegeneinander Fußball spielen. Sowohl die Hardware als auch die Software werden von den Forschungsgruppen selbst entwickelt, wobei nur menschliche oder menschenähnliche Sensoren und Aktoren eingesetzt werden dürfen. Eine Ausnahme bildet hier nur die Kommunikation, die noch über ein drahtloses Netzwerk abläuft. Per Wireless Local Area Network (W-LAN) unterhalten sich nicht nur die Roboter untereinander, sondern der Schiedsrichter kann so auch ein Tor oder ein Foul den Spielern mitteilen.

Jedes Jahr wird das Regelwerk in den einzelnen Ligen angepasst und so immer mehr Elemente des menschlichen Fußballs eingeführt. Das Ziel, im Jahr 2050 in einem fairen Spiel gegen den amtierenden Fußballweltmeister der Menschen anzutreten und zu gewinnen gibt den Weg vor. Voraussetzung für ein Spiel unter fairen Bedingungen ist es, dass sowohl die Regeln als auch die Ausstattung der Hardware im Roboterfußball an die Menschen und ihre Fußballregeln angepasst sind. Die Kommunikation wird daher im Laufe der Zeit ohne eine Netzwerkverbindung auskommen müssen, da diese Kommunikation auf einem Spielfeld in einer lauten Umgebung einen großen Wettbewerbsvorteil bringen würde, der für die Menschen so nicht zu nutzen ist.

Eine Motivation, den Austausch der Roboter schon heute auf die Synthese und Verarbeitung natürlicher Sprache umzustellen, ergibt sich aus den beständigen Problemen mit der Netzwerkinfrastruktur auf den Turnieren. In jeder Liga haben sowohl das Spielfeld als auch die Arena ihr eigenes Netzwerk. Um dies zu realisieren, werden in jeder Halle mehrere Dutzend W-LAN-Router parallel betrieben. Doch auch viele Teams nutzen zu Testzwecken ein eigenes W-LAN, Roboter werden nach einem Spiel nicht wieder aus dem offiziellen Netzwerk genommen und Besucher in der Halle haben immer öfter Smartphones dabei, die sich mit den offiziellen Netzwerken verbinden wollen. Die Folge sind Interferenzen und Ausfälle der Router, die wie bei einer Distributed Denial of Service Attacke (DDoS) den

ständigen Anfragen nicht mehr Stand halten können. Häufig müssen Spiele daher komplett ohne Netzwerk stattfinden oder die Signale kommen nur mit einer großen Verzögerung an.

Auf eine Kommunikation zwischen den Robotern kann dennoch nur schwerlich verzichtet werden: Nicht jeder Roboter hat ein Modell des gesamten Spielfeldes, oft ist nur die direkte Umgebung bekannt. Daher kann ein Roboter allein nicht abschätzen, ob er selbst derjenige Spieler ist, der am nächsten zum Ball steht und daher zum Ball laufen sollte, oder ob ein anderer bereits auf dem Weg ist. Findet keine Kommunikation statt, bewegen sich mehrere Roboter zum Ball und bringen sich dabei häufig gegenseitig zu Fall. Auch kann es passieren, dass ein Roboter die Orientierung auf dem Spielfeld verliert oder den Ball nicht mehr finden kann. Eine erneute Lokalisation ist aufwendig und kann zu Fehlern führen. Dabei sehen Mitspieler vielleicht noch den Ball oder wissen, wo sie sich auf dem Feld befinden. Ein Austausch solcher Informationen kann wertvolle Zeit und Rechenressourcen sparen.

Da bisher noch kein Team seine Kommunikation auf natürliche Sprache umgestellt hat, ist der Ansatz der natürlichsprachlichen Kommunikation nicht nur ein großer Schritt hin zu menschenähnlicherem Fußball, sondern könnte auch einen deutlichen Vorteil bei Turnieren bieten, weil die Roboter nicht mehr auf die fehleranfälligen W-LAN-Netzwerke angewiesen sind.

2.1 Spezifikation der Roboterhardware

An der Universität Hamburg werden DARwIn-OP Roboter (Dynamic Anthropomorphic Robot with Intelligence – Open Platform) im Team *Hamburg Bit-Bots* genutzt. Diese wurden von der koreanischen Firma ROBOTIS in Zusammenarbeit mit der Virginia Tech, Purdue University, und der University of Pennsylvania entwickelt und werden inzwischen von vielen Teams in der Humanoid Kid Size League verwendet [14]. Große Teile der Hardware sind für die Anforderungen des Fußballspiels optimiert. Da eine Kommunikation über natürliche Sprache nicht zu diesem Anforderungskatalog zählt, wird im folgenden Kapitel die Hardware dieser Roboter im Hinblick auf die neuen Ansprüche untersucht.

2.1.1 Anforderungen an das Lautsprechersystem

Nach offizieller Herstellerspezifikation ist im DARwIn-OP ein Lautsprecher mit einem Schalldruckpegel von 93 ± 3 dB [0,1 m/0,1 W] mit einer Impedanz von 8 Ω ± 15 % bei 1,0 V verbaut.

Ein erster Test unter Realbedingungen bei einem Testspiel mit Publikum hat gezeigt, dass das Mikrophon klar und deutlich Umgebungsgeräusche auch über größere Distanzen aufnimmt. Aufgrund der Größe des Lautsprechers ist seine Ausgangsleistung jedoch begrenzt, so dass die Sprache der Roboter auch mit größter Aussteuerung nur bis zu einer Distanz von drei Metern aufgezeichnet werden konnte.

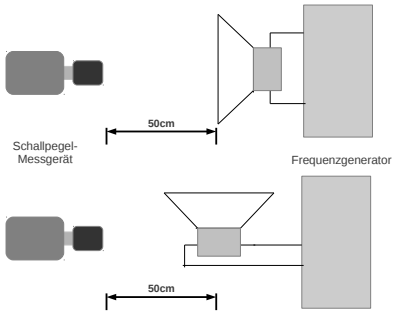

Abbildung 2.1: Versuchsaufbau Leistungstest der Lautsprecher des DARwIn-OP

Um genauere Anforderungen für eine Hardwaremodifikation zu evaluieren, wurde eine Testreihe mit dem Lautsprecher aus dem DARwIn-OP und einem vergleichbaren 3-Watt-Breitbandlautsprecher mit 8 Ω Impedanz und einer Membrangröße von 9 cm x 5 cm durchgeführt, im Folgenden mit *Referenzl.* abgekürzt. Dazu wurden ein Schallpegel-Messgerät der Firma Rhode&Schwartz sowie ein Frequenzgenerator von Grundig verwendet. Der exakte Versuchsaufbau ist in Abbildung 2.1 dargestellt.

Dabei sind Messungen bei 0,5 V und 1,0 V effektiver Spannung jeweils bei frontaler Ausrichtung der Lautsprecher sowie bei einer Drehung um 90° erfolgt. Gemessen wurde bei Frequenzen im Bereich von 600 Hz bis 1500 Hz.

Folgende Messwerte wurden bei einer Durchführung mit 0,5 V Spannung ermittelt:

Typ	**600 Hz**	**800 Hz**	**1000 Hz**	**1200 Hz**	**1500 Hz**
DARwIn-OP (frontal)	23 dB	40 dB	38 dB	32 dB	27 dB
DARwIn-OP (gedreht)	24dB	39dB	27dB	27dB	20dB
Referenzl. (frontal)	44 dB	41 dB	50 dB	53 dB	50 dB
Referenzl. (gedreht)	35 dB	39 dB	42 dB	46 dB	48 dB

Folgende Messwerte wurden bei einer Durchführung mit 1,0 V Spannung ermittelt:

Typ	**600 Hz**	**800 Hz**	**1000 Hz**	**1200 Hz**	**1500 Hz**
DARwIn-OP (frontal)	30 dB	50 dB	42 dB	39 dB	34 dB
DARwIn-OP (gedreht)	27 dB	43 dB	31 dB	35 dB	26 dB
Referenzl. (frontal)	52 dB	52 dB	58 dB	61 dB	62 dB
Referenzl. (gedreht)	42 dB	45 dB	53 dB	51 dB	57 dB

Die Auswertung zeigt, dass der bereits im Roboter verbaute Lautsprecher in einem Bereich von 800 Hz eine sehr gute Leistung auch im Vergleich zu Lautsprechern mit einer höheren Wattzahl und einer größeren Membran erbringen kann. Jedoch nimmt diese Leistung bereits im Bereich von 600 Hz und 1000 Hz stark

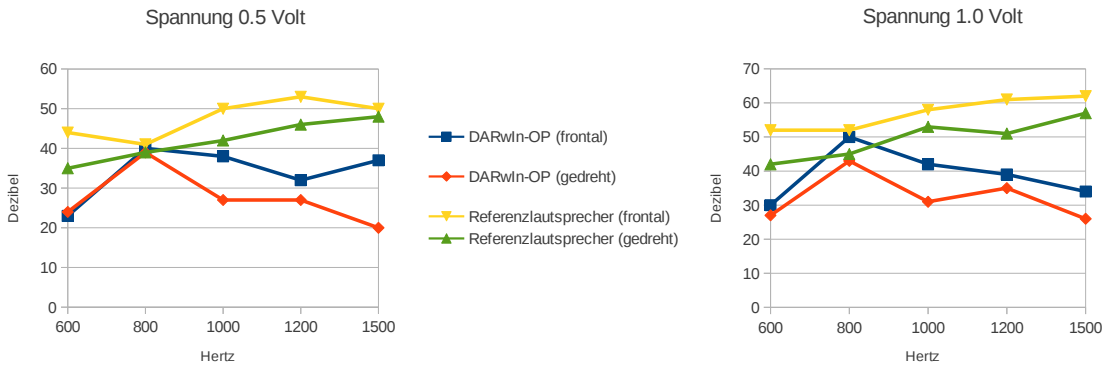

Abbildung 2.2: Messreihe der Lautsprecherleistung für den DARwIn-OP und einen Referenzlautsprecher, links bei 0,5 V Spannung, rechts bei 1,0 V Spannung

ab (vergleiche Abbildung 2.2). Da die vom Roboter synthetisch erzeugte Sprache jedoch zum großen Teil Frequenzen von unter 500 Hz produziert (vergleiche Abbildung 2.3), ist eine Veränderung des Lautsprechers sinnvoll, so dass auch im unteren Frequenzbereich lautere Töne erzeugt werden können. Der Frequenzbereich über 10000 Hz kann dagegen vollständig ignoriert werden, da die vom Roboter synthetisch erzeugte Sprache keine so hohen Frequenzen erzeugt. Werden Frequenzen in einem höheren Bereich aufgezeichnet, können diese als Umgebungsgeräusche oder Rauschen identifiziert und herausgefiltert werden. Zusätzlich ist eine weitere Signalverstärkung ratsam – diese könnte durch einen der USB-Anschlüsse auf dem Mainboard oder ein externes, kleines Akkusystem mit Energie versorgt werden. Eine Vergrößerung der Lautsprechermembran ist dagegen nur in einem sehr geringen Umfang möglich, da weder in der Brust noch im Kopf des Roboters zwischen dem Metallgerüst und der äußeren Plastikverkleidung genügend Platz zur Verfügung

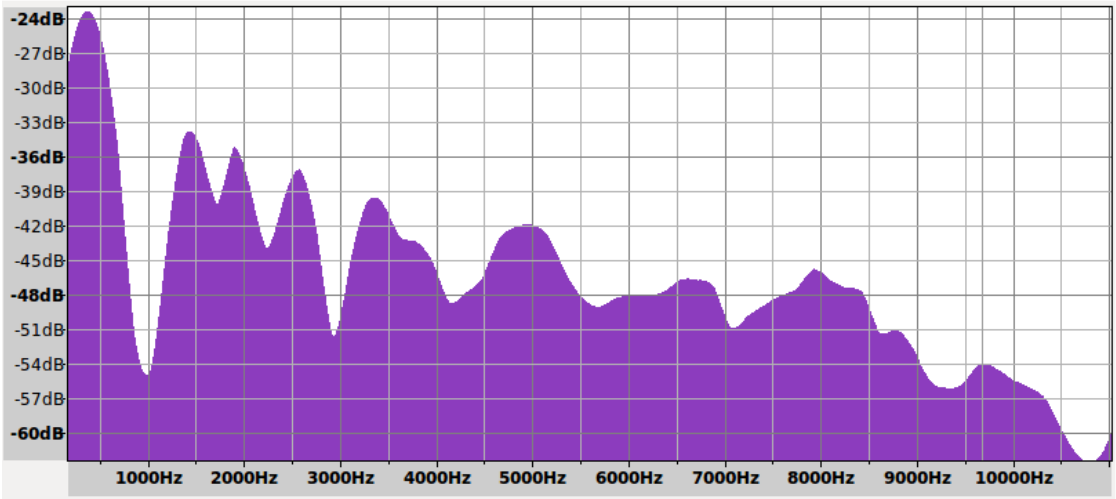

Abbildung 2.3: Frequenzanalyse des synthetisch erzeugten Wortes *„Distance“*

steht. An einer anderen Stelle darf der Lautsprecher dem Regelwerk entsprechend nicht angebracht werden.

Beim Einbau des Lautsprechersystems ist darauf zu achten, dass ein Roboter einem anderen Roboter nur selten frontal gegenübersteht. Wie in den Vergleichskurven (Abb. 2.2) zu sehen ist, nimmt die Lautstärke bei einer Drehung um 90° in den meisten Frequenzbereichen bereits enorm ab. Dies ist dem Aufbau des Lautsprechers geschuldet, der nur auf eine Abstrahlung nach vorn und nicht zur Seite ausgelegt ist. Hinzu kommt die Abschirmung durch das Plastikgehäuse, das um den Lautsprecher herum angebracht ist. Beides führt zu einem erhöhten Leistungsanspruch an den Lautsprecher, damit diese Einschränkungen kompensiert werden können.

2.1.2 Anforderungen an die Mikrophone

Es sind serienmäßig zwei Mikrophone im Kopf des DARwIn-OP verbaut, die jeweils eine Sensitivität von -62 ± 2 dB, eine Impedanz von 2,2 kΩ und ein Frequenzspektrum von 50 bis 16000 Hz haben. Diese Mikrophone sind jedoch, wie in Abbildung 2.4 gezeigt, nur über Analogports an das CM-730-Board im Roboter angeschlossen. Daher können die Mikrophone mit der aktueller Firmware nur mit einer maximalen Abtastrate von 100 Hz ausgelesen werden. Nach dem Abtasttheorem von Shannon muss jedoch ein Signal mit einer Bandbreite von $f_{\min}$ bis $f_{\max}$ mit $2 \cdot f_{\max}$ abgetastet werden, damit eine vollständige Rekonstruktion des Signals möglich ist [16]. Menschliche Sprache hat eine Minimalfrequenz von 50 Hz und eine Maximalfrequenz von ca. 6000 Hz [4], der DARwIN-OP Roboter dagegen erzeugt seine Sprachausgabe zwar hauptsächlich im Bereich bis 1000 Hz, trotzdem erstreckt sich das Frequenzspektrum, wie in Abbildung 2.3 zu sehen ist, bis 10000 Hz. Nach dem Abtasttheorem muss also eine Abtastung mit mindestens 2000 Hz, besser jedoch sogar 20000 Hz möglich sein. Die beschriebenen Mikrophone sind daher ungeeignet.

Im Kopf des DARwIN-OP ist noch ein weiteres Mikrophon in der Logitech-C905-Webcam verbaut. Da es sich bei der Webcam nicht um ein Bauteil von ROBOTIS handelt, ist auch keine Open-Source-Spezifikation dazu veröffentlicht. Auch bei einer Recherche beim Hersteller konnte keine nähere Spezifikation der

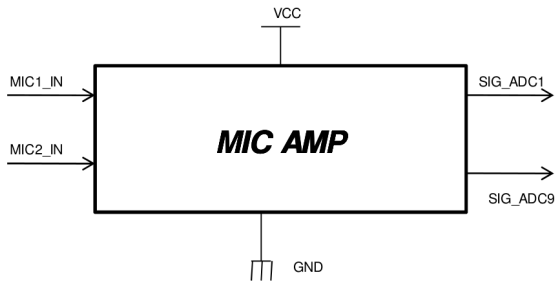

Abbildung 2.4: Schaltplan des *Microphone Amplifier* auf dem CM-730-Board [19, S. 22].

Mikrophone gefunden werden. Da es jedoch über die Webcam mit USB 2.0 an den im DARwInOP verbauten FitPC2 angeschlossen ist, lassen sich problemlos Abtastraten von über 20000 Hz realisieren.

Um ein Sprachanalysesystem langfristig auf dem DARwInOP zu etablieren, ist es jedoch notwendig, mehr als ein Mikrophon zu benutzen, um stärkere Signalwerte in einem größeren Bereich um den Roboter herum zu erhalten. Lässt sich durch ein Firmware-Update des CM-730-Boards noch keine signifikante Verbesserung in der Abtastrate erzielen, ist auch die Installation von zwei eigenen Mikrophonen im Kopf denkbar. Diese könnten am Kopfplastik befestigt werden und über USB oder den ebenfalls auf dem FitPC2 verbauten Klinkenanschluss angeschlossen werden. Eine Nutzung von mehr als zwei Mikrophonen gleichzeitig ist nach Regelwerk nicht gestattet.

2.1.3 Einschränkungen durch den Prozessor

Im DARwInOP ist ein FitPC2 bzw. FitPC2i verbaut. Dieser besitzt eine Intel Atom Z530-CPU mit 1,6 GHz Taktung und einem Gigabyte RAM. Die Sprachverarbeitung muss unter diesen gegebenen Bedingungen trotzdem noch in einer solchen Geschwindigkeit berechnet werden, dass ein dynamisches Spiel möglich ist. Dabei ist zu beachten, dass sich daraus keine Rechenzeitnachteile für die Bilderkennung oder die Berechnung der Roboterbewegungen ergeben dürfen, die ebenfalls dieselbe Hardware nutzen.

2.2 Analyse der Spielabläufe im Hinblick auf den Nachrichtenaustausch

In der bisherigen Architektur werden bereits in den folgenden Bereichen Daten über das W-LAN-Netzwerk unter den Robotern ausgetauscht bzw. ein Austausch der Daten ist geplant:

- **Spielstatus:** Bisher erfahren die Roboter über das Netzwerk, wenn ein anderer Roboter ein Foul begangen und daher das Spielfeld verlassen hat. Diese Information kann unmittelbar in der Spielstrategie verwendet werden, da so beispielsweise ein anderer Roboter die Aufgaben des Torwarts übernehmen kann, wenn dieser das Spielfeld verlassen muss. Ist die Strafe vorüber und hat der Roboter seine Orientierung auf dem Spielfeld wiedergefunden, sollte er dieses ebenfalls mitteilen.

- **Balldistanz:** Damit nicht alle Roboter zum Ball laufen, sobald sie diesen sehen, wird die geschätzte Distanz zum Ball kommuniziert. Derjenige Roboter, der die kürzeste Distanz gemeldet hat, versucht dann den Ball zu erreichen, während die anderen Roboter abwarten bzw. sich in eine günstige Position zu bringen versuchen. Sinnvoll ist es, die Balldistanz und die bestmögliche Strecke direkt in eine geschätzte Dauer bis zur Erreichung des Balles umzurechnen, denn potentielle Hindernisse auf dem Weg können dazu führen,

dass ein Roboter zwar nach euklidischer Distanz näher zum Ball steht, den direkten Weg jedoch nicht nehmen kann und daher länger als die anderen Roboter braucht.

- **Position auf dem Spielfeld:** Um sich ein globales Bild vom Spielfeld zu machen, braucht der Roboter nicht nur die globale eigene Position und die Ballposition, sondern auch die Position der Mitspieler. Zwar kann er deren Position aus seinen Kamerabildern heraus berechnen, um jedoch ein globales gemeinsames Weltbild zu schaffen, müssen die Ergebnisse der Berechnung regelmäßig mit den anderen Spielern abgeglichen werden.

- **Einsatzgebiet des Roboters:** Die Roboter können kommunizieren, ob sie gerade als Feldspieler oder Torwart spielen. Diese Information kann sowohl für die Teamstrategie, als auch für eine Vereinfachung der Lokalisation genutzt werden. Weiß der einzelne Roboter zum Beispiel, dass ein Ball weniger als zwei Meter vom Torwart entfernt und dieser im eigenen Tor ist, kann angenommen werden, dass sich der Ball in der eigenen Spielfeldhälfte befindet.

2.3 Einbindung der Kommunikation in das bestehende Softwaresystem

Bei einem funktionsfähigen W-LAN, über das die Roboter Daten austauschen können, bietet diese Art der Kommunikation heute noch entscheidende Vorteile, die durch die Kommunikation über natürliche Sprache noch nicht zu leisten sind. So kann jeder Roboter mehrmals pro Sekunde aktuelle Daten aller anderen Roboter in seine Berechnungen einbeziehen. Es ist unter diesen Voraussetzungen nicht notwendig zu prüfen, welche Daten dringend notwendig sind, da ohnehin alle Daten zur Verfügung stehen. Für die Kommunikation über natürliche Sprache muss sowohl die Anzahl der zu kommunizierenden Nachrichten als auch die Kommunikationshäufigkeit deutlich reduziert werden. Eine Umstellung auf eine Softwarearchitektur, die entscheidet, welche Informationen in einer bestimmten Situation benötigt werden, und diese dann über Sprache erfragt, ist notwendig, wenn natürlichsprachliche Kommunikation genutzt werden soll.

Um die Vorteile beider Systeme zu nutzen, ist es empfehlenswert, die Softwarearchitektur auf ein hybrides System umzustellen: Unter Normalbedingungen nutzen die Roboter weiterhin das W-LAN zur Kommunikation. Dabei merkt sich jedoch jeder Roboter, wie viel Zeit seit dem letzten Update durch das Netzwerk vergangen ist. Wird diese Zeit zu groß, wird die Kommunikation auf natürliche Sprache umgestellt. Sobald das W-LAN wieder aktuelle Daten liefert, kann der Nachrichtenaustausch wieder darauf umgestellt werden.

Kapitel 3

Der Nachrichtenaustausch

Die Roboter auf dem Spielfeld benötigen für ihr Spielverhalten regelmäßig Informationen von anderen Robotern. Dazu müssen sie Nachrichten austauschen bzw. kommunizieren, um an diese Informationen zu gelangen. Die bisherige Kommunikation wurde mittels Versendung von Paketen über das Netzwerk realisiert. Eine Umstellung auf direkte lautsprachliche Kommunikation erfordert die Entwicklung neuer Nachrichtenaustauschstrategien, die im Folgenden beschrieben werden [16].

3.1 Die Dialogmanagementstrategie

„Wenn an ... [einer] Kommunikation mindestens zwei Personen beteiligt sind, die sich sowohl selbst äußern, als auch die Äußerung der anderen unmittelbar wahrnehmen, spricht man von einem Dialog, einer Wechselrede zwischen mindestens zwei Sprechern.“ [4, S. 395]

Ein Dialog erfordert, dass jeder Kommunikationspartner bereits erhaltene Informationen speichern und dahingehend verarbeiten kann, dass er daraus Handlungen oder weitere Äußerungen bzw. Nachfragen generieren kann.

Innerhalb eines Dialogmanagementsystems ist die Dialogkontrolle dafür zuständig, auf Basis eines gegebenen Inputs eine solche Entscheidung zu treffen. Dazu benötigt es ggf. Kontextinformationen, die durch das Kontextmodellierungssystem des Dialogmanagers aufbereitet werden [9].

Die Dialogkontrolle lässt sich nach Jokinen und McTear basierend auf Graphen oder Frames realisieren. Ein graphenbasiertes Modell besteht aus einer endlichen Menge an Zuständen und lässt sich durch eine Menge von Knoten, welche die eigenen Fragen oder Aussagen repräsentieren, und eine Menge von Übergängen zwischen den Knoten beschreiben, welche die möglichen Antworten des Dialogpartners enthalten. Der Vorteil eines solchen Systems liegt in der einfachen Modellierung, da nur eine endliche Menge an vorher bekannten Zustandsübergängen implementiert werden muss. Dies wird dann zu einem Nachteil, wenn das System an Komplexität zunimmt.

Eine andere Möglichkeit für die Dialogkontrolle liegt in der Benutzung von Frames. Ein Frame besteht dabei aus einer Reihe von Variablen, die als freie Platz-

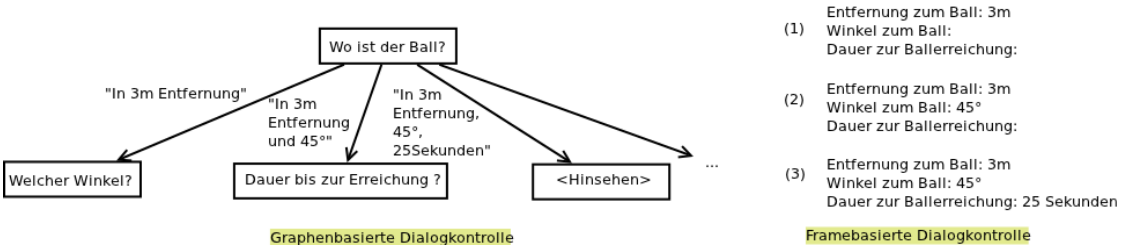

Abbildung 3.1: Gegenüberstellung von graphenbasiertem und framebasiertem Dialogkontrollsystem

halter für bestimmte Daten dienen, die während des Dialogs gesammelt werden sollen. Immer wenn eine neue Information aufgenommen wurde, wird der entsprechende Platz im Frame dadurch besetzt. Zunächst kann nach irgendeiner Information gefragt werden, denn alle Variablen sind noch unbelegt. Ist eine Information verarbeitet, kann abhängig von der Dialogstrategie ein anderer Wert abgefragt werden. Frames bieten eine wesentlich flexiblere Möglichkeit der Dialogsteuerung, denn insbesondere bei unspezifischen Fragen können in der Antwort unterschiedlich viele Informationen enthalten sein. Bei einem graphenbasierten Dialogmodell müsste für jede Anzahl an Informationen ein eigener Übergang geschaffen werden, während bei einem framebasierten Kontrollsystem einfach die entsprechenden Felder belegt werden. Die beiden unterschiedlichen Verfahren sind in Abbildung 3.1 dargestellt.

Für den Nachrichtenaustausch zwischen den Robotern ist ein framebasiertes Kontrollsystem in diesem Anwendungskontext besser geeignet als ein graphenbasiertes Verfahren. Die Frames lassen sich einfach als Dictionary oder Map implementieren, wobei die Schlüssel oder Keys der Map den Feldern entsprechen und die Werte mit den aktuellen Informationen belegt werden. Ein weiterer Vorteil ist, dass die Frames leicht zu erweitern sind, wenn Dialoge ausgebaut werden. Außerdem kann ein Feld angelegt werden, das dafür genutzt wird, den Zeitpunkt der letzten Änderung im Frame zu speichern. Dadurch lässt sich leicht entscheiden, ob eine Information noch aktuell genug ist. Ein weiteres Entscheidungskriterium für ein framebasiertes Kontrollsystem ist, dass das jetzige Kommunikationsmodell bereits auf Frames beruht, die in jedem Updatezyklus mit Daten gefüllt werden. Es ist daher keine Umstellung der Architektur notwendig, sondern lediglich eine Erweiterung, da mit der Kommunikation über natürliche Sprache nicht mehr in jedem Updatezyklus alle Informationen bereitgestellt werden. Stattdessen muss berechnet werden, welche Informationen im aktuellen Spielstatus relevant sind, um diese gegebenenfalls abzufragen.

Die relevanten Kontextinformationen lassen sich ebenfalls in Frames abspeichern bzw. aus diesen gewinnen. Jokinen und McTear beschreiben fünf relevante Punkte, die eine Kontextmodellierung liefern sollte:

- Dialogverlauf: In beschriebenen Kontext wird kein kompletter Dialogverlauf benötigt, da bestimmte Daten wie z.B. eine Balldistanz vor mehreren Minuten für das aktuelle Verhalten nicht mehr relevant sind. Alle zukünftig

weiterhin relevanten Daten aus dem Verlauf bleiben in den Frames gespeichert.

- Aufgabenbeschreibung: Durch die unbesetzten Felder, die in den Frames vorgehalten werden, weiß der Roboter stets, welche relevanten Daten er noch sammeln muss.

- Modell der Domäne: Wie bereits in Abschnitt 2.3 beschrieben, sollte die Softwarearchitektur so gestaltet werden, dass die Kommunikation und Dialogmodellierung ohne Kenntnis der Domäne auskommt. An anderer Stelle sollte berechnet werden, welche Informationen zum aktuellen Zeitpunkt nützlich sind. Das Kommunikationsmodul benutzt dieses Wissen, um daraus einen Dialog zu generieren.

- Modell der Konversationskompetenz: Der Roboter benötigt ein Modell davon, wann er in einem Dialog selbst die Dialogkontrolle übernehmen darf und an welcher Stelle er lediglich auf Fragen antworten sollte (siehe Kapitel 3.2).

- Präferenzmodell der Dialogpartner: Der Roboter benötigt keine erweiterten Informationen über die anderen Roboter, da sich ihre Präferenzen untereinander nicht unterscheiden.

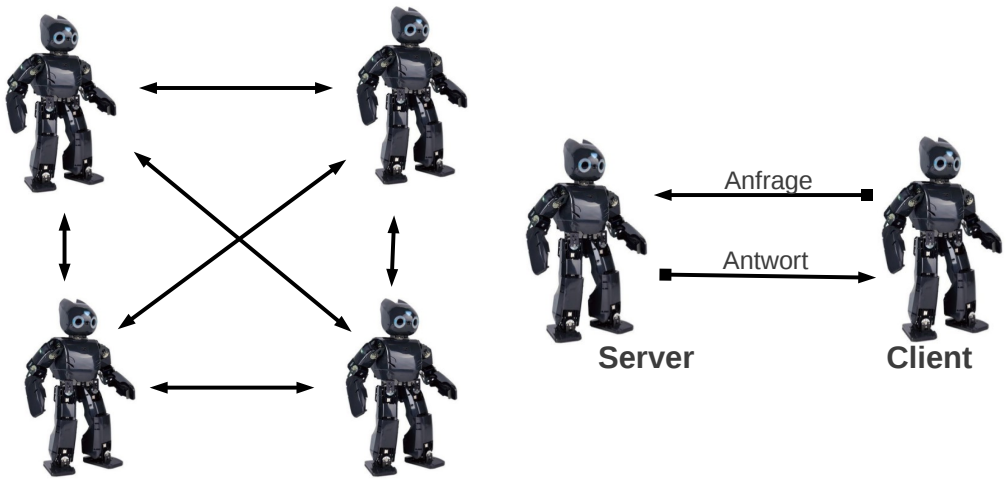

Abbildung 3.2: Links das globale Dialogmodell als Peer-to-Peer-Struktur, rechts der einzelne Dialog als Client-Server-Modell.
DARwIn-OP Grafik von *http://darwinop.sourceforge.net/*

3.2 Die Konversationsmodellierung

Grundsätzlich gibt es bei jeder Modellierung des Nachrichtenaustauschs zwei mögliche Beziehungen, in denen die Dialogpartner zueinander stehen können: Eine symmetrische Beziehung und eine asymmetrische Beziehung.

In einer asymmetrischen Beziehung, oft auch als Client-Server-Modell bezeichnet, stellt einer der Kommunikationspartner, der Client, eine Anfrage an den Server. Der Server selbst kann keine Anfragen stellen, sondern nur Anfragen von den Clients beantworten und so Informationen bereitstellen. Im Kontrast dazu steht das symmetrische Modell, auch Peer-to-Peer-Verbindung genannt, in dem jeder Kommunikationspartner sowohl Anfragen stellen als auch beantworten kann [10].

Bezogen auf die Domäne RoboCup bieten sich so zwei Möglichkeiten: Eine zentrale Instanz, zum Beispiel der Torwart, stellt Anfragen an die anderen Roboter. Alle anderen Roboter fungieren als Server, die lediglich Informationen bereitstellen. Der Torwart errechnet aus den Informationen ein Spielverhalten für die anderen Roboter und gibt diese in Form von Anweisungen an die Mitspieler weiter. Dieses Modell bietet allerdings keinerlei Ausfallsicherheit – wenn der Torwart wegen eines Defekts vom Platz genommen werden muss oder durch eine Überlastung keine Anfragen mehr beantworten kann, findet keine Kommunikation mehr statt. Das Problem der Ausfallsicherheit durch ein zentrales Kommunikationssystem kann durch die Implementation eines Client-Server-Systems nur verschoben werden.

Aus diesen Überlegungen ergibt sich, dass jeder Roboter in der Lage sein muss, selbst einen Dialog durch Anfragen an andere Roboter zu beginnen, aber auch auf Anfragen zu reagieren. Um die Konversation für die Roboter zu vereinfachen, bietet es sich jedoch an, jeden einzelnen Dialog als Client-Server-Modell zu modellieren. Es kann also global betrachtet jeder Roboter eine Anfrage an andere Roboter stellen und eine solche auch beantworten. Innerhalb eines Dialogs fungiert aber immer ein Roboter als Client und einer als Server. Derjenige Roboter, der den Dialog beginnt, nimmt die Rolle des Clients ein, und der antwortende Roboter ist der Server. Die antwortenden Roboter können jedoch ihrerseits im Anschluss einen eigenen Dialog beginnen, um selbst eine Frage stellen zu können. Abbildung 3.2 zeigt das Zusammenspiel der beiden Modelle.

3.3 Die Vermittlungstechnik

Ein Dialog kann durch die direkte Ansprache eines Dialogpartners initiiert werden (Punkt-zu-Punkt-Verbindung) oder durch eine generelle Frage, auf die jeder antworten kann, der die Frage gehört hat (Broadcast).

Die erste beschriebene Möglichkeit einer Punkt-zu-Punkt-Dialogvermittlung zeichnet sich durch eine eindeutige Kennzeichnung aus, welcher mögliche Kommunikationspartner angesprochen werden soll. Dazu wird entweder eine direkte Ansprache per Namen benötigt oder ein Verbindungsaufbau mittels Augenkontakt oder Gesten. Da die Roboter bisher nicht präzise andere Roboter erkennen können, bleibt nur ein Kommunikationsaufbauversuch mittels direkter Ansprache.

Geeignet ist diese Vermittlungstechnik immer dann, wenn einen Roboter die Information von genau einem anderen Roboter interessiert. Es wird im Quelltext festgelegt, dass in einem solchen Fall der Satz immer mit dem Namen des angesprochenen Roboters beginnen soll.

Oft wird es in der Kommunikation unter den Robotern notwendig sein, eine Information von allen Robotern zu erhalten. In diesem Fall verlängert sich die Nachrichtenaustauschzeit stark, wenn ein Roboter zunächst jeden einzelnen Roboter ansprechen muss. Es bietet sich daher an, eine allgemeine Frage ohne konkreten Adressaten zu stellen. Das erste Wort einer allgemeinen Frage ist dann nicht der Name eines Roboters. Haben alle Roboter die Frage vernommen, ist nun noch zu regeln, in welcher Reihenfolge sie antworten dürfen, damit nicht alle Roboter gleichzeitig mit dem Sprechen beginnen. Hier bietet es sich an, die Nummer der Roboter im Team zu nutzen, so dass zunächst der Roboter mit der kleinsten Nummer spricht. Hat dieser seinen Satz beendet oder in einem definierten Zeitraum nicht mit dem Sprechen begonnen, kann der nächste Roboter beginnen. Da die Reihenfolge klar definiert ist, kann auch jeder Roboter erkennen, welcher Roboter gerade spricht und sich diese Information im entsprechenden Frame abspeichern. Jeder Roboter, der Frage und Antworten der anderen Roboter vernommen hat, kann so seine eigenen Frames aktualisieren.

Auf den Aufbau einer Verbindung durch eine Synchronisation und eine Bestätigung des Nachrichteneingangs durch den Synchronisationspartner wird verzichtet, da der Nachrichtenaustausch so wenig Zeit wie möglich in Anspruch nehmen darf. So können die Roboter auf Basis der erhaltenen Nachrichten schnell ihre Spielzüge planen.

Kapitel 4

Konzepte der Spracherzeugung und Sprachverarbeitung

4.1 Sprachsynthese

Unter einer Sprachsynthese verstehen wir „die Umsetzung von geschriebener Sprache ... in Lautsprache, also in Sprachsignale“ [16, S. 194], im Folgenden auch als Text-to-Speech-System (TTS) bezeichnet. Als Bewertungskriterium für die Qualität von Sprachsynthese wird heute vor allem die Natürlichkeit der Aussprache gesehen [8]. Einige Autoren unterscheiden zwischen High-Level- und Low-Level-Synthese, wobei sich erstere mit der Phonetik (also der Produktion und physikalischen Beschaffenheit von Sprache) und letztere mit der Funktion von Sprachlauten im Sprachsystem (Phonologie) beschäftigt [11]. Im Folgenden werden drei unterschiedliche Ansätze im Bereich der Low-Level-Sprachsynthese vorgestellt. Die Phonologie wird hierbei nicht betrachtet, da sie für unseren Anwendungskontext nicht unmittelbar relevant ist.

4.1.1 Artikulationssynthese

Der Ansatz der Artikulationssynthese ist es, die neurophysiologischen und biomechanischen Abläufe bei der Lautbildung des Menschen möglichst exakt zu kopieren. Dabei werden mit Hilfe von Parametern die Kieferstellung und die Zungenbewegung angepasst und nachgebildet. Dieses Verfahren erfordert nicht nur eine langwierige Anpassung der Parameter, sondern vor allem einen mechanischen Nachbau des menschlichen Sprechapparats bzw. eine entsprechende Softwaresimulation. Beides ist sehr aufwendig, die Ergebnisse aber dennoch nicht so humanoid wie erhofft. Daher wird in neueren Forschungen daran gearbeitet, die dynamischen Parameter direkt aus dem Sprachsignal zu gewinnen und damit einen direkten Zusammenhang zwischen Modell und Sprachsignal herzustellen [16][23].

4.1.2 Formantsynthese

Als Formanten bezeichnet man einen Frequenzbereich, „in dem die Obertöne stärker erscheinen als in anderen Frequenzbereichen“[4, S. 201]. Grundlage der Formantsynthese ist es, dass für die Lautwahrnehmung in erster Linie Lage, Höhe und Güte des Formanten ausschlaggebend sind. In der Regel werden pro Formant mehrere Filter eingesetzt, die das Sprachsignal erzeugen sollen. Daraus ergibt sich vor allem die Möglichkeit, Lautübergänge zu formen, so dass kein störendes Stocken zwischen den Lauten wahrgenommen wird [16].

4.1.3 Verkettungssynthese

Die Modellierung von Sprache wird hier als eine Konkatenation von einzelnen Spracheinheiten betrachtet, die passend aneinandergereiht werden. Spracheinheiten können zum Beispiel Phone, „die kleinste diskrete Lauteinheit im Lautkontinuum“[7, S. 299] oder Diphone sein. Diphone werden entweder diejenigen Lauteinheiten genannt, die von der Mitte eines Phons bis zur Mitte des nächsten Phons reichen, oder solche, die vom Punkt der geringsten Änderung im ersten Phon bis zum gleichen Punkt im nächsten Phon reichen. Im Gegensatz zur Phon-Synthese werden bei der Diphon-Synthese die Lautübergänge mitbetrachtet, so dass die erzeugte Sprache flüssiger klingt.

Der Diphon-Ansatz wird immer dann gewählt, wenn nur wenige Grundelemente zur Verfügung stehen sollen. Erweitert man die Datenbank der Grundelemente, so dass variantenreichere Worte zur Verfügung stehen, dann spricht man von Korpussynthese. Hier versucht das System die Worte entweder direkt aus der Datenbank zu entnehmen oder möglichst große und ähnlich klingende Wortteile miteinander zu verketten. Nur wenn das nicht gelingt, wird auf die Diphon-Synthese zurückgegriffen. Um den Preis höheren Speicherbedarfs bietet einem die Verkettungssynthese eine natürlichere Lautausgabe.

4.2 Spracherkennung

Als Spracherkennung wird „das Umsetzen eines Sprachsignals in eine textliche Form“ bezeichnet [16, S. 28]. In den nachfolgenden Abschnitten werden zunächst einige Vorverarbeitungsschritte beschrieben, die eine Analyse überhaupt erst ermöglichen oder erleichtern. Anschließend werden zwei verschiedene Ansätze zur Spracherkennung vorgestellt und basierend darauf eine Verfahrensauswahl für den Anwendungskontext getroffen.

4.2.1 Vorverarbeitung

Um das aufgezeichnete Signal einer Sprachanalyse zuzuführen, muss es zunächst in eine dafür geeignete Form gebracht werden. Dies beinhaltet sowohl Störgeräusche frühzeitig zu eliminieren als diese auch von Intensität und Grundfrequenz zu ab-

strahieren. Die folgenden Abschnitte beschreiben Ansätze und Algorithmen einer solchen Vorverarbeitung.

Diskrete Fouriertransformation: Mit Hilfe einer Fouriertransformation (DTF) lässt sich eine Signalfunktion x(n), also der Verlauf eines Signalparameters über die Zeit, in eine Spektralfunktion X(k), also die Energie im Frequenzspektrum, überführen:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)kn}, \qquad 0 \leq k \leq N-1 \tag{4.1}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j(2\pi/N)kn}, \qquad 0 \leq k \leq N-1 \tag{4.2}$$

N entspricht hierbei der Anzahl der Abtastwerte.

Nimmt man am Anfang einer Aufnahme stets mehrere Sekunden ohne ein Sprachsignal von Robotern auf, kann mit Hilfe des dort erhaltenen Frequenzspektrums eine Filterung der späteren Aufnahmen durchgeführt werden. Unerwünschte Störsignale wie Rauschen und Umgebungsgeräusche in den Aufnahmen werden dadurch erheblich reduziert.

Mel-Cepstrum: In der Sprachverarbeitung wird eine Fouriertransformation häufig nur als Teil eines komplexeren Vorverarbeitungsverfahrens benutzt, bei der

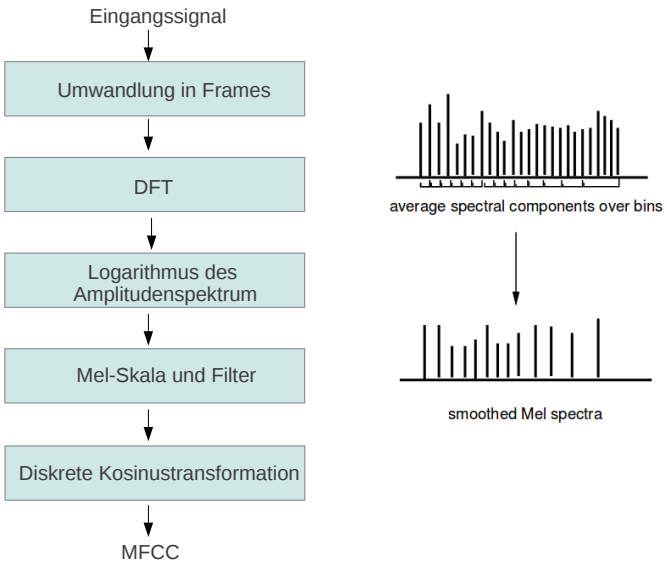

Abbildung 4.1: Algorithmus der Transformation eines Sprachsignals in das Mel-Cepstrum von [12]

unter anderem das Spektrum geglättet, eine Grundperiode erkannt oder ein Übertragungskanal kompensiert werden kann. Einen Algorithmus für eine solche Faltung, die eine Eingangsfolge x(n) im Zeitbereich in eine Ausgangssequenz c(m) überführt, nennt man Cepstrum. Statt einer reinen Abbildung auf das Frequenzspektrum, wie im Abschnitt über die Diskrete Fouriertransformation beschrieben, wird in der Sprachverarbeitung eine Abbildung auf die so genannte Mel-Skala bevorzugt, denn diese abstrahiert von Intensität und Grundfrequenz des Signals. Dazu wird folgende Skalentransformation durchgeführt:

$$h(f) = 2595 \cdot \log_{10}(1 + \frac{f}{700Hz}) \tag{4.3}$$

f beschreibt hier die Frequenz in Hz [16].

Nach Beth Logan lässt sich eine Überführung in ein Mel-Cepstrum durch die in Abbildung 4.1 abgebildeten Schritte durchführen [12]. Zunächst wird das gesamte Signal in kleinere Frames, auch als Fenster bezeichnet, aufgeteilt. Diese sind typischerweise 20 ms lang und dienen dazu, Kanteneffekte zu vermeiden. Ab jetzt wird jede der folgenden Operationen auf die Frames getrennt angewendet. Zunächst wird nach dem beschriebenen Verfahren eine Fouriertransformation angewendet. Da die wahrgenommene Lautstärke näherungsweise logarithmisch beschrieben werden kann, wird von dem Ergebnis der DFT der Logarithmus gebildet. Die Frequenzbänder werden nun erneut in so genannte Behälter unterteilt, wobei diese nicht in jedem Frequenzbereich gleich groß sind. Je höher die Frequenz, desto größer ist der Frequenzbereich pro Behälter, d.h. es finden sich mehr Frequenzbereiche in einem Behälter wieder. Da tiefere Frequenzen in der Sprachverarbeitung eine größere Bedeutung haben als hohe, entsteht so eine größere Skalierbarkeit in den Behältern mit den niedrigen Frequenzbereichen. Die Abbildung des Frequenzspektrums auf das Mel-Cepstrum, die pro Behälter vorgenommen wird, ist in Abbildung 4.1 rechts dargestellt. Zuletzt wird durch eine diskrete Kosinustransformation eine Dekorrelation der Komponenten bewirkt:

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right], \qquad k = 0, \ldots, N-1 \tag{4.4}$$

Formel 4.4 ist jedoch nur eine Anlehnung an die eigentliche Formel, die sich aus der DFT ergibt:

$$c(m) = \frac{1}{J}\sum_{n=1}^{J}(\log S_n)\cos\left[m \cdot (n - \frac{1}{2})\frac{\pi}{J}\right], \qquad 0 \le m \le D \tag{4.5}$$

$$S_n = \sum_k X(k) \cdot H_n(k), \qquad 1 \le j \le J \tag{4.6}$$

$H_n(k)$ entspricht der Anwendung des n-ten Filters der Mel-Filterbank, X(k) ist das Ergebnis der Fouriertransformation und D die Anzahl cepstraler Koeffizienten [16].

Hoch- und Tiefpassfilter, Rauschfilterung: Um Realdaten aus einem Mikrophon zu nutzen, sind weitere Vorverarbeitungsschritte wie eine Rauschfilterung oder auch eine Hoch- bzw. Tiefpassfilterung wünschenswert, um die zu analysierenden Daten mit so wenig irrelevanten Signalwerten wie möglich zu belasten. Insbesondere zur Rauschfilterung gibt es bereits zahlreiche verschiedene Ansätze und Algorithmen, die nicht Gegenstand dieser Arbeit sind. Daher wird – sofern nötig – auf eine händische Vorverarbeitung durch die freie Software Audacity zurückgegriffen [13].

4.2.2 Analyse von Beginn und Ende einer Sprachaufzeichnung

Während eines Spielszenarios nimmt der Roboter mit seinen Mikrophonen ständig Umgebungsgeräusche wahr. Damit Rechenzeit gespart werden kann, ist es nützlich zu definieren, ab wann ein relevantes Sprachsignal vorliegt und wann lediglich Grundrauschen aufgenommen wird.

Pfister und Kaufmann schlagen hier ein einfaches automatisiertes Verfahren vor, dass einen Intensitätsschwellwert S, eine Minimaldauer einer Äußerung t_a und die Maximaldauer einer Pause t_b benötigt [16]. Der Roboter soll sich danach stets in einem von fünf Zuständen befinden, wobei bei mehreren Aufnahmen hintereinander auch der fünfte Zustand durch einen Übergang vom vierten Zustand zurück in den ersten ersetzt werden kann. Der fünfte Zustand, der den Endzustand beschreibt, fällt dadurch weg. Abbildung 4.2 zeigt, unter welchen Umständen ein Zustandsübergang erfolgt. Initial befindet sich das System dabei in Zustand *S1*, in welchem noch kein Sprachsignal wahrgenommen wurde. Wird die Intensität des aktuellen Signalparameters größer als der Schwellwert S, so findet ein Wechsel in Zustand *S2* statt. Sinkt die Intensität des Signalparameters wieder, bevor eine definierte Zeitdauer t_a überschritten wurde, wird der Anstieg der Intensität als Da-

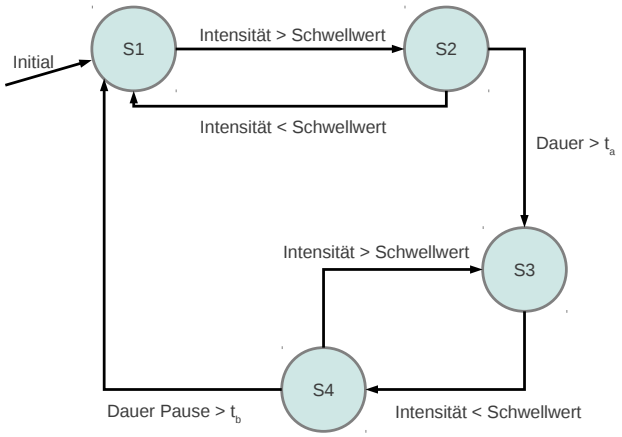

Abbildung 4.2: Zustandsübergänge bei der Detektion von relevanten Sprachsignalen in Anlehnung an [16]

tenausreißer gewertet und daher wieder verworfen. Das System kehrt in Zustand *S1* zurück. Bleibt die Intensität dagegen konstant über dem Schwellwert für länger als t_a, folgt ein Zustandsübergang zu Zustand *S3*. In diesem Zustand wurde der Anfang eines gesprochenen Textes detektiert. Solange sich das System in diesem Zustand befindet, werden alle eingehenden Signalparameter als zur Spracheingabe gehörig eingestuft. Sinkt in diesem Zustand die Intensität des aktuellen Signalwerts wieder unter den Schwellwert, geht das System in Zustand *S4*, dem Ende der Sprachaufzeichnung über. Nur wenn für länger als t_b die Intensität niedriger als der Schwellwert verbleibt, wird das Ende der Sprachaufzeichnung bestätigt und das System geht in Zustand *S1* über, so dass eine neue Spracheingabe detektiert werden kann. Steigt die Intensität in *S4* wieder über den Schwellwert, so geht das System in Zustand *S3* zurück und die Signalwerte werden der Spracheingabe zugerechnet. Alle nicht explizit beschriebenen Ereignisse führen stets zu einem reflexiven Zustandsübergang.

Da dieses Verfahren hauptsächlich für ruhige Umgebungen gedacht ist, gilt es zu prüfen, in wie weit das Verfahren trotzdem im RoboCup Anwendung finden kann.

4.2.3 Spracherkennung mit Hilfe eines Mustervergleichs

Beim Sprachmustervergleich wird ein aufgezeichnetes Sprachsignal mit allen Sprachsignalen verglichen, die vorher vorgegeben wurden. Die Signalfunktion mit der geringsten Distanz zum Mikrophon-Input wird für die weiteren Analysen verwendet (vgl. [16]).

Vorbedingungen: Der Mustervergleich erfordert neben der aufgezeichneten Signalfunktion von jeder möglichen Eingabe ein Vergleichsmuster. Da alle Roboter die gleiche Sprachsynthese benutzen, reicht als Vergleichsmuster ein aufgezeichnetes Signal aus. Zu jedem Vergleichsmuster muss der gesprochene Text in Schriftsprache gespeichert sein, da während des eigentlichen Analyseschrittes das am besten passendste Referenzmuster berechnet und von diesem der schriftlich zugeordnete Text zu Weiterverarbeitung genutzt wird.

Durchführung: Ein Vergleich zwischen zwei Signalen lässt sich durch eine Distanzfunktion realisieren. Dabei sollen Funktionen, die sich sehr ähnlich sind, eine geringe Distanz aufweisen. Da das Signal in Form einer Funktion des Signalparameters über die Zeit vorliegt, kann für jeden Zeitpunkt ein Vektor angeben werden, der den Signalparameter und den Zeitpunkt enthält. Zum selben Zeitpunkt ist der Unterschied der beiden Signalfunktionen also nur die Differenz im Signalparameter.

Das Problem des Sprachmustervergleichs liegt in den Unterschieden vom aufgezeichneten Signal und den Datenbanksignalen auf Ebene der Prosodie, also den lautlichen Strukturen der Sprache [2]. Es liegen Differenzen im Bereich der Lautstärke, der Länge, des Sprechrhythmus und der Sprachmelodie vor. Die Unterschiede in Lautstärke und Grundfrequenz werden bereits in der Vorverarbei-

tung durch Überführung der Sprachsignale und der Vergleichsfunktionen in das Mel-Cepstrum aufgelöst. Das Signal liegt nun als Vektor von Frames vor. Für den eigentlichen Mustervergleich muss nun die Distanz zu jedem Zeitpunkt zwischen dem Referenzsignal und dem zu analysierenden Signal berechnet werden.

Durch eine Variation in der Länge der ausgesprochenen Silben und einer Änderung des Sprechrhythmus kann jedoch nicht davon ausgegangen werden, dass zum gleichen Zeitpunkt im Referenzsignal und dem aufgezeichneten Signal die gleiche Silbe gesprochen wird. Es muss also für jeden Zeitpunkt des zu analysierenden Signals derjenige Punkt gefunden werden, dessen Abstand zum Referenzsignal am geringsten ist. Dabei müssen bestimmte Bedingungen erfüllt werden, beispielsweise die Einhaltung der zeitlichen Reihenfolge der Silben. Dies wird mit Hilfe des Dynamic-Time-Warping-Algorithmus (DTW) realisiert, welcher eine minimale Distanz zwischen zwei Sprachsignalen unter bestimmten Vorbedingungen berechnen kann.

4.2.4 Stochastische Sprachverarbeitung

Vorbedingungen: In der stochastischen Sprachverarbeitung werden keine Referenzsignalkurven benötigt, dafür jedoch eine Zuordnung von Signalmerkmalen zu Lauten, Silben oder Wörtern sowie ein Lexikon, das sämtliche zu erkennende Einheitensequenzen enthält.

Durchführung: Nach Rabiner [17] und Ebert und Ebert [4] läuft eine stochastische Spracherkennung in fünf Schritten ab, wobei die letzten beiden Schritte lediglich einer Suchraumeingrenzung dienen und nicht von elementarer Bedeutung sind.

Die Verarbeitung beginnt, wie auch bei der Spachanalyse durch Mustervergleich, mit einer Digitalisierung des Sprachsignals, sowie mit einer Vorverarbeitung durch Filterung und Frequenzanalyse. Zur Vorverarbeitung zählt jedoch hier als elementarer Schritt die Merkmalsextraktion aus dem Sprachsignal. In diesem Schritt findet eine erhebliche Datenreduktion statt. Im Anschluss werden diese Merkmale in Laute, Silben oder Wörter überführt, je nachdem für welche Granularität von Untereinheiten man sich entschieden hat. Die Erkennung einer Einheit beruht nun auf einer Wahrscheinlichkeitsberechnung nach dem Hidden-Markov-Modell. Im Hidden-Markov Modell wird davon ausgegangen, dass die Wahrschein-

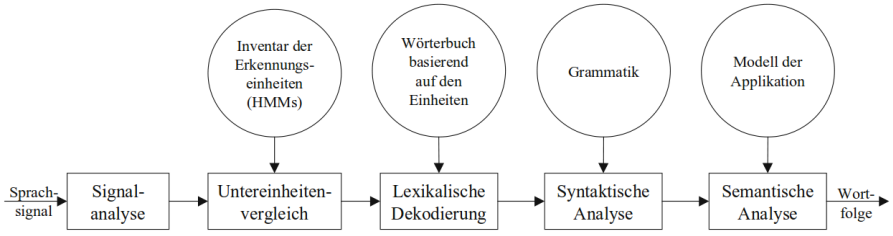

Abbildung 4.3: Spracherkenner nach Rabiner [4]

lichkeit für einen Zustandsübergang, in unserem Fall also die Erkennung einer Einheit, nur vom aktuellen Zustand abhängt und die Wahrscheinlichkeiten für den Zustandsübergang konstant sind. Die Wahrscheinlichkeit für die Erkennung einer Untereinheit hängt also deterministisch von den vorher erkannten Einheiten ab. Die Zustände selbst sind dabei verborgen und können nur durch die Ausgabe der wahrscheinlichsten Einheit beobachtet werden. Aus diesen Untereinheiten werden in der Weiterverarbeitung Wörter und Sätze nach einem vorliegenden Lexikon gebildet. Weitere Einschränkungen bei der Eliminierung von möglichen Einheitensequenzen bieten syntaktische und semantische Analysen, wobei diese ebenfalls mit umfangreichen Wörterbüchern bzw. Verarbeitungsverfahren implementiert werden müssen. Am Ende der Verarbeitung ist die Wortfolge ermittelt, die mit größter Wahrscheinlichkeit in dem Sprachsignal enthalten war.

4.3 Verfahrensauswahl

Da sich diese Arbeit ausschließlich auf eine Roboter-Roboter-Kommunikation beschränkt, können durch die Auswahl von geeigneten Verfahren in der Sprachsynthese Vereinfachungen in der Sprachanalyse erwirkt werden: Fällt die Entscheidung auf eine Syntheseart, bei der keine Änderungen im Sprachrhythmus und der Sprachgeschwindigkeit von Worten auftreten können, sondern das gleiche Wort an jeder Stelle im Satz und in jeder Satzumgebung immer gleich klingt, so ist beim Mustervergleich bereits eine sehr gute Erkennungsrate zu erwarten. Es kann sogar geprüft werden, in wie weit von dem DTW-Algorithmus abstrahiert werden kann und ob dieser durch eine rein lineare Signalverschiebung entlang der Zeitachse ersetzt werden kann. Dies wird möglich, da zwar der exakte Anfangszeitpunkt des Sprachsignals nicht bekannt ist, aber immer die gleiche Silben- und Pausenlänge garantiert werden kann. Dies lässt eine Reduktion der Zeit und Rechenkomplexität erwarten, die unter den gegebenen Hardwareeinschränkungen und im Hinblick auf einen dynamischen Spielfluss sehr relevant sind.

Eine stochastische Sprachverarbeitung ist unter diesen Voraussetzungen nicht nur sehr komplex in der Berechnung, sondern kann kaum eine Verbesserung in der Erkennungsrate bringen, da diese schon im Mustervergleich nahezu optimal sein sollte.

Die Wahl für das zu implementierende System fällt daher auf eine Verkettungssynthese, die zur Erzeugung eines Satzes immer gleich klingende Worte aus einer Datenbank verbindet, in Zusammenhang mit einem (ggf. vereinfachten) Mustervergleich zur Sprachanalyse.

Sprachanalysen mittels Mustervergleich sind nur bedingt auf die Erkennung von menschlicher Sprache, die nicht auf einen bestimmten Sprecher trainiert ist, zu übertragen. Es wäre nicht nur eine große Anzahl von Referenzmustern nötig, sondern auch eine größere Variabilität im Signal, das vom DTW-Algorithmus nicht gewährleistet werden kann. Daher bedeutet diese Wahl der Sprachanalyse eine starke Einschränkung für die mögliche Erweiterung auf die Interaktion der Roboter mit Menschen und die damit verbundene Analyse menschlicher Sprache.

Kapitel 5

Implementation

Nachfolgend wird ein beispielhafter Ansatz zur Implementation einer Roboter-Roboter-Kommunikation vorgestellt. Dabei wird von der konkreten Anwendung auf dem DARwIn-OP-Roboter abstrahiert und sowohl für die Sprachsynthese als auch für die Sprachanlyse ein allgemeiner Ansatz vorgestellt, der sich in beliebige Programmiersprachen und auf beliebige Systeme übertragen lässt.

5.1 Sprachsynthese

In Kapitel 4.3 ist die Verkettungssynthese als optimaler Ansatz ermittelt worden, wobei ganze Worte als Einheiten verwendet werden sollen. Im Laufe der Recherche gelang es jedoch nicht, ein bereits existierendes Programm ausfindig zu machen, das eine solche Synthese realisiert und sich einfach in das bestehende Softwarearchitektursystem auf dem Roboter integrieren lässt.

Die Wahl ist daher auf das Programm eSpeak gefallen, welches die Formantsynthese anwendet und in vielen Sprachen erhältlich ist [1]. Es läuft unter Linux als Konsolen-Tool und kann daher einfach in Programmiersprachen integriert werden, auch wenn es keine Bibliothek für diese Sprache gibt. Eine Nachinstallation auf den Robotern ist nicht nötig, da es in der Linuxversion, die auf den Robotern läuft, bereits vorhanden ist. Ein großer Vorteil des Konsolenprogramms ist, dass sich Sprechgeschwindigkeit und Pitch des Sprachsignals einfach über Parameter anpassen lassen. So kann leicht jedem Roboter eine unterscheidbare Stimme gegeben werden, ohne große Änderungen im Programmtext vornehmen zu müssen.

Das Programm eSpeak ist nicht darauf ausgelegt, eine menschenähnliche Stimme zu erzeugen, so dass das Ergebnis der Synthese eindeutig als mechanisch erzeugtes Signal zu erkennen ist. Es stellte sich jedoch als problematisch heraus, dass, bedingt durch die Formantsynthese, das gleiche Wort am Satzanfang anders ausgesprochen wird als am Satzende oder in der Satzmitte. Damit dies nicht zu einem Problem für die Sprachverarbeitung führt, wird die Formantsynthese so verwendet, dass die Ausgabe wie eine Verkettungssynthese erscheint: Innerhalb eines Satzes steht nach jedem Wort ein Punkt, so dass das System jedes Wort als eigenen Satz behandelt und somit immer gleich ausspricht. Diese Art der Synthese

bietet als weiteren Vorteil längere Pausen zwischen den einzelnen Wörtern, die eine Trennung einzelner Wörter im Satz möglich machen.

Eine Trennung zwischen zwei Sätzen kann auf zwei Arten realisiert werden: Eine Möglichkeit besteht darin, dass ein Satzende durch ein definiertes letztes Wort bestimmt wird. Problematisch ist hier jedoch, dass die Satzseparierung nicht korrekt möglich ist, wenn das letzte Wort fehlerhaft erkannt wurde. Daher bietet es sich auch hier an, die Satztrennung an Hand einer Pause zu detektieren, wobei diese Pause mindestens doppelt so lang wie die Wortpause sein muss, damit eine klare Trennung der beiden Pausen möglich wird.

5.2 Sprachsynthese

5.2.1 Auswahl der Programmiersprache

Die Softwarearchitektur der Roboter des Hamburger RoboCup-Teams setzt sich aus einem komplexen Klassenkonstrukt zusammen, das zum einen Teil in der Interpretersprache Python und zum anderen in der Compilersprache C++ verfasst ist. Dieses hybride System ermöglicht durch kurze Compile-Zeiten schnelle Softwareänderungen auch während eines Spiels, da nur wenige Sekunden gebraucht werden, um die aktuelle Software zu kompilieren.

Damit sich die Sprachanalyse optimal in die bestehende Softwarearchitektur einpflegen lässt, ist es notwendig, Teile der Software in C++ und Teile in Python zu implementieren, um einerseits die Compile-Zeit gering zu halten und andererseits aufwendige Rechenoperationen wie die Distanzberechnung von Vektoren so zeitsparend wie möglich ausführen zu können. Um jedoch zunächst das Verfahren an sich unabhängig von den Robotern zu testen, ist die Wahl auf eine Sprache gefallen, die Audioanalyse durch eigene Bibliotheken gut unterstützt. Daher ist das Testsystem, das im Folgenden vorgestellt wird, in Java verfasst.

5.2.2 Die Softwarearchitektur

Herzstück der Software ist ein Zustandsautomat entsprechend der Abbildung 4.2.2. Im Initialzustand werden dafür die Referenzdaten eingelesen, außerdem wird der zu prüfende Audiostream initialisiert. Jeder danach eingelesene Signalwert wird darauf überprüft, ob er bloßes Grundrauschen darstellt oder Sprache zugeordnet werden kann. Sobald ein Wert als von sprachlicher Herkunft klassifiziert wurde, wird dieser als Wortanfang gespeichert. Folgt allerdings innerhalb der nächsten zehn Samples wieder ein Datenwert, der nicht zur Sprache zählt, wird dieser Anfangswert wieder verworfen. Folgen erst im weiteren Verlauf Signalwerte, die dem Grundrauschen zugeordnet werden, wird ein potentielles Wortende markiert. Dieses wird jedoch erst bestätigt, wenn die folgenden Samples ebenfalls dem Grundrauschen zugeordnet werden. Wie groß die Anzahl dieser Pausensamples ist, hängt davon ab, ob es sich bei den Analysedaten um Idealdaten oder Originaldaten aus Mikrophonaufnahmen handelt. Im ersten Fall muss die Pause mindestens eine Länge von

5000 Samples aufweisen, im zweiten Fall ist der Minimalwert der Pause doppelt so lang. Dieser hohe Wert wurde gewählt, damit die einzelnen Worte bei der Spracherzeugung als eigenständiger Satz generiert werden können (vgl. 5.1). So folgt auf jedes Wort eine lange Pause, die gut für eine eine Segmentierung geeignet ist. Sind Wortanfang und Wortende gefunden, wird die Länge des Wortes geprüft. Ist dieses kleiner als 500 Samples, kann es sich nicht um ein Wort handeln, sondern ein Störgeräusch wurde irrtümlich detektiert. Die Aufnahme wird im Ergebnis verworfen und es wird nach einem neuen Wortanfang gesucht. Anderenfalls wird das gefundene Wort der eigentlichen Analyse übergeben. Der Zyklus der Worterkennung beginnt gleichzeitig von vorn.

5.2.3 Die Energiewertbestimmung

Die Energiewertbestimmung soll für einen gegebenen Signalwert entscheiden, ob dieser zu einer Spracheingabe gehört oder nicht. Zu diesem Zweck wird über die letzten im Buffer gespeicherten Werte des Signalparameters ein gleitender Durchschnittswert berechnet. Ist das eingehende Signal mindestens doppelt so laut wie dieser Durchschnittswert und wird dabei zusätzlich ein bestimmter Schwellwert überschritten, so wird ein Signalwert einer Spracheingabe zugeordnet. Der gleitende Durchschnitt D berechnet sich dabei wie folgt:

$$D = \sum_{i=0}^{n} \frac{value(i)}{n} \tag{5.1}$$

value(i) entspricht dabei dem Signalwert mit dem Index i und n der Anzahl der betrachteten Samples. Die Zahl n bestimmt sich nach der aktuellen Anzahl der Werte, die im Buffer gehalten werden. Durchschnittlich sind dies 6000 Samples, Abweichungen davon gibt es sowohl bei der Speicherung der ersten Signalwerte als auch bei längeren Worten. Ein einmal als Sprachanfang markierter Wert bleibt so lang im Buffer, bis er revidiert wurde oder ein komplettes Wort erkannt wurde.

Der Vorteil der gleitenden Durchschnittsberechnung über den Werten im Streambuffer liegt auf der einen Seite darin, dass der Durchschnittswert durch einen Ausreißer-Signalwert nicht zu stark beeinflusst wird. Andererseits fließen auch nicht zu viele Werte in die Berechnung ein, denn dies würde dazu führen, dass auch korrekte Spracheingaben nicht mehr detektiert werden, da sich der Durchschnittswert zu langsam anpasst.

Für die Wortanalyse wird die Zuordnung eines Signalparameters zu einer Spracheingabe vor der Weiterverarbeitung gefiltert. Der Filter hält dazu eine Liste der letzten zehn Signalparameter bereit. Sind unter diesen mindestens zwei, die nach der Energiewertbestimmung einem Sprachsignal zugeordnet werden können, wird jeder eingehende Wert ebenfalls Sprache zugeordnet. Im Filter werden die ursprünglichen Zuordnungen nach dem gleitenden Durchschnitt gespeichert, die gefilterten Werte werden lediglich für die Weiterverarbeitung genutzt. So werden die Signalwerte um ein tatsächliches Sprachsignal herum geglättet.

5.2.4 Der Streambuffer

Im Streambuffer ist die temporäre Speicherung der eingelesenen Signalwerte realisiert. Dabei werden neue Werte aus dem Audiostream abgefragt und in einem Array gespeichert. Bei jedem Einlesen eines neuen Wertes wird außerdem überprüft, ob Daten aus dem Buffer wieder entfernt werden können. Dies ist dann der Fall, wenn kein Wortanfang markiert und die Länge des Buffers von 6000 Samples überschritten ist.

Der Streambuffer bietet Methoden an, einen neuen Wortanfang und ein neues Wortende zu setzen und gesetzte Werte wieder zurück zu setzen. Mit Hilfe dieser Methoden kann ein Array zurückgegeben werden, das die Signalwerte für ein detektiertes Wort plus den Pufferbereich für die Verschiebung enthält (vgl. Quellcode 5.1).

```
1  public double[] getInterestingData()
2    {
3      //Berechnet die Laenge fuer das zu erzeugende Array
4      int length = end_interesting-start_interesting+2*config.OVERHEAD;
5
6      double[] temp = new double[length];
7      //Fuegt die Werte in das Array zur Ausgabe ein
8      for(int i = 0; i<temp.length; i++)
9      {
10       //Hier wird der Fall abgefangen, dass vor dem Wortanfang
11       //nicht genuegend Bufferwerte zur Verfuegung stehen
12       if(config.OVERHEAD < start_interesting)
13       {
14       temp[i] = buffer.get(i+start_interesting-config.OVERHEAD);
15       }
16       else
17       {
18         temp[i] = buffer.get(i);
19       }
20     }
21     return temp;
22   }
```

Quellcode 5.1: Erzeugung des Arrays mit dem relevanten Sprachbereich für die weitere Analyse

5.2.5 Die Wortanalyse

Wenn eine Gruppe von Samples als interessante Region gekennzeichnet wurde, gilt es als nächstes, die größte Übereinstimmung mit einer der Sounddateien, die als Referenzdaten angegeben sind, zu suchen. Wegen der Störgeräusche kann dabei nicht gewährleistet werden, dass der Anfang und das Ende des Wortes exakt erkannt worden sind. Daher werden für jedes zu analysierende Wort 256 lineare Verschiebungen getestet und das globale Minimum über alle daraus resultierenden Distanzberechnungen gebildet. Dazu werden im Buffer immer 2048 Samples vorgehalten, die vor dem markierten Wortanfang und nach dem markierten Wortende liegen. Es wird nun in jedem Analysedurchgang ein neuer Anfangswert a für die Analyse bestimmt und von diesem bis zum Signalwert a + Wortlänge L geprüft. Hat also die Wortanalyse den Wortanfang an der Stelle x markiert, so wird im ersten Analysedurchlauf von x-1024 bis x-1024+L geprüft. Im nächsten Durchlauf wird sowohl auf den Wortanfang als auch auf das Ende vom vorherigen Durchgang ein Indexwert von acht Samples addiert und daraus ergibt sich der nächste zu prüfende Bereich. Es ergibt sich also, dass jede achte Verschiebung im Bereich

1024 + detektiertes Wort + 1024 getestet wird.

Jeder so erhaltene Wert wird gegen alle möglichen Referenzdaten geprüft. Dazu wird zunächst der in jedem Schritt zu überprüfende Bereich in das Mel-Cepstrum umgewandelt. Zur Umwandlung der Sounddateien wird eine Bibliothek von Klaus Seyerlehner verwendet [21][22]. Die Bibliothek liefert ein Array von Vektoren zurück, in denen pro Cepstrum-Behälter die Werte gespeichert sind. Jeder Vektor in dem Array der zu prüfenden Sounddateien wird nun gegen den Vektor mit dem gleichen Index in den Referenzdaten geprüft. Der Quelltextausschnitt 5.2 beschreibt dieses Verfahren. Von allen berechneten Distanzwerten wird nur das globale Minimum gespeichert, alle anderen Vergleichswerte werden verworfen.

Für die Referenzdatei, deren berechnete Distanz am geringsten war, wird der Inhalt durch Nachschlagen in einer Datei ermittelt, in welcher die Zuordnung von Dateiname und gesprochenem Inhalt vermerkt ist.

```
1  private void computeMin(double[] temp, int i)
2  {
3    // Fuehrt die Umwandlung in das Mel-Cepstrum durch
4    Vector<double[]> toCheck = generateVector(temp);
5
6    // Prueft Uebereinstimmung mit jeder moeglichen Referenzdatei
7    for(String s : referenzen.keySet())
8    {
9      //Vektordistanz berechnen
10     double distance = getVectorDistance(toCheck, referenzen.get(s));
11     //Globales Minimum speichern
12     if(distance < minDistance)
13     {
14       minDistance = distance;
15       valueOfMinDist = s;
16     }
17   }
18 }
```

Quellcode 5.2: Berechnung der besten Übereinstimmung durch Umwandlung der Sounddatei in das Mel-Cepstrum und anschließende Vektordistanzberechnung

5.2.6 Speicherung der Referenzdaten

Da jeder Roboter, der eine Sprachanalyse im Spiel durchführen soll, Referenzdaten benötigt und diese auf allen Robotern identisch sind, bietet es sich nicht an, eine Menge von Sounddateien als Referenz zu verwenden. Der Speicherplatzbedarf für die erforderliche Datenmenge wäre unnötig groß, außerdem müsste jeder Roboter zu Beginn der Sprachanalyse die Referenzdaten in ein Mel-Cepstrum umwandeln, wodurch eine hohe Redundanz entsteht und Rechenzeit verschwendet wird. Es bietet sich daher an, den Referenzdatensatz vor dem Überspielen auf die Roboter in das Mel-Cepstrum umzuwandeln und lediglich die so entstandene Array- und Vektorstruktur zu speichern. Als Speicherform wurde das XML-Format gewählt, wobei die Spezifikation der Sounddatei das höchste Element bildet. Diesem Element sind die einzelnen Arrays untergeordnet, welche wiederum die Vektoren enthalten. Auf unterster Ebene werden in jedem Vektor die einzelnen Mel-Cepstrum-Werte als Element eingefügt (vgl. Abb. 5.1). Beim Starten der Sprachanalyse müssen nur noch die XML-Files wieder eingelesen und in die erwartete Array-Vektor-Struktur gebracht werden.

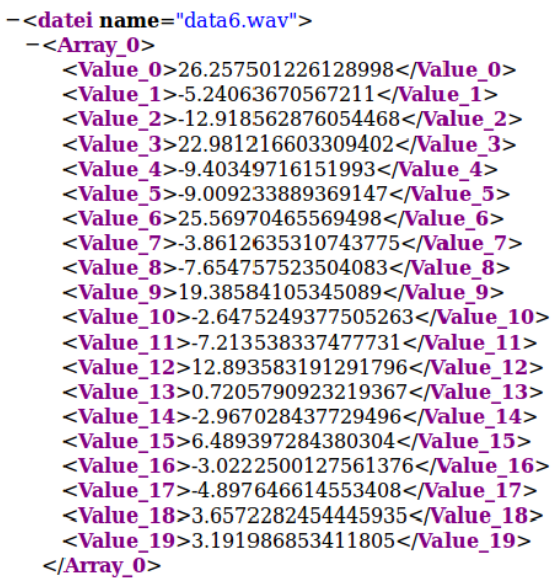

Abbildung 5.1: Struktur der erzeugten XML-Dateien.

5.2.7 Test- und Analysestruktur

Um das System auch ohne Roboter testbar zu machen, wurde eine Architektur entwickelt, die einen Mikrophoninput mit einer Audiodatei simulieren kann. Diese Klasse liest die Datei zwar komplett ein, stellt die Signalwerte aber nur nach und nach zur Verfügung. Mit dieser Hilfsklasse ist es sowohl möglich auf dem Rechner selbst erzeugte Idealdaten zu testen, als auch auf dem Roboter aufgenommene Audiodateien zu verwenden.

Für die Analyse der Erkennungsraten wird ein Ordner mit zu prüfenden Daten eingelesen. In diesem ist eine Datei hinterlegt, die zu jeder Sounddatei die tatsächlich synthetisch erzeugten Worte speichert. Für jede Analyse werden die vom Programm errechneten Wörter mit den tatsächlich erzeugten verglichen. Richtig erkannte Worte werden sowohl nach Position als auch nach Inhalt gespeichert, so dass eine prozentuale Auswertung der Erkennungsraten nach beiden Mustern möglich ist.

Kapitel 6

Evaluation und Ergebnisse

In diesem Kapitel werden verschiedene Aspekte der im vorherigen Kapitel beschriebenen Implementation kritisch beleuchtet, außerdem werden die Ergebnisse vorgestellt, die damit erzielt werden konnten.

6.1 Evaluation

6.1.1 Einteilung der Sprachdateien in Sektionen

In der ersten Version des Programms wurde die Spracheingabe durch den endlichen Zustandsautomaten (State-Machine) lediglich in Sätze unterteilt. Dies führte dazu, dass nur für die Daten zur Distanzberechnung von 0,00 bis 6,00 Meter 600 Referenzdatensätze pro eingegebenem Satz geprüft werden mussten. Um eine zukunftsfähige Software zu schaffen, die auch für größere Distanzen und weitere Datensätze wie die Angabe einer Position auf dem Spielfeld nutzbar ist, muss die Menge an Referenzdaten möglichst klein gehalten werden.

Die Idee war daher, den Algorithmus so umzustellen, dass die Spracheingabe nach einzelnen Worten getrennt wird und diese einzeln analysiert werden. Grundvoraussetzung dafür ist, dass die einzelnen Worte untereinander in ihrer Abbildung im Mel-Cepstrum einen genügend großen Abstand voneinander haben, so dass sie weiterhin klar voneinander abgegrenzt werden können. In einem Test wurden daher zunächst die Referenzdaten gegen sich selbst geprüft, um die Distanz der Worte zueinander zu errechnen. Tabelle 6.1 zeigt das Ergebnis, wobei als Referenzdaten auf dem Computer erzeugte eSpeak-Sounddateien verwendet wurden.

Die Matrix zeigt, dass es zwar einzelne Distanzen wie die von *Nine* zu *Five* gibt, die mit 7,5 recht gering sind, im Durchschnitt jedoch ein Abstand von 16,02 der einzelnen Soundfiles zueinander gegeben ist. Da diese Werte hoch genug sind, um einen Mustervergleich darauf aufzubauen, wurde die Sprachanalyse ab der zweiten Version auf die Detektion von Wörtern umgestellt.

	D	M	P	0	1	2	3	4	5	6	7	8	9
D	**0.0**	17.7	17.1	17.4	20.5	20.4	22.7	20.3	16.9	20.0	17.5	22.1	17.8
M	17.7	**0.0**	18.6	15.5	19.7	18.3	20.0	18.2	19.1	17.8	16.8	19.4	19.1
P	17.1	18.6	**0.0**	15.1	13.4	14.3	15.9	14.2	11.5	16.3	13.3	20.6	13.2
0	17.4	15.5	15.1	**0.0**	21.2	18.1	21.2	18.1	16.4	17.7	11.2	23.7	17.7
1	20.5	19.7	13.4	21.2	**0.0**	17.2	16.6	12.3	13.3	19.7	19.3	19.1	13.3
2	20.4	18.3	14.3	18.1	17.2	**0.0**	10.5	16.4	16.5	14.2	17.0	18.6	17.9
3	22.7	20.0	15.9	21.2	16.6	10.5	**0.0**	17.0	17.2	15.4	19.6	19.5	19.0
4	20.3	18.2	14.2	18.1	12.3	16.4	17.0	**0.0**	12.9	17.8	16.2	21.3	14.5
5	16.9	19.1	11.5	16.4	13.3	16.5	17.2	12.9	**0.0**	18.3	13.4	22.6	7.5
6	20.0	17.8	16.3	17.7	19.7	14.2	15.4	17.8	18.3	**0.0**	16.7	16.1	19.9
7	17.5	16.8	13.3	11.2	19.3	17.0	19.6	16.2	13.4	16.7	**0.0**	23.3	15.4
8	22.1	19.4	20.6	23.7	19.1	18.6	19.5	21.3	22.6	16.1	23.3	**0.0**	22.3
9	17.8	19.1	13.2	17.7	13.3	17.9	19.0	14.5	7.5	19.9	15.4	22.3	**0.0**

Abbildung 6.1: Distanzen der Referenzdatensätze zueinander, wobei D für das Wort „Distance“, M für „Meter“ und P für „Point“ steht

6.1.2 Bewertung der Detektion von Sprachbeginn und Sprachende

Die Detektion von Wortbeginn und Wortende beruht insbesondere auf der Sprachpause, die zwischen den Wörtern gemacht wird. Dank der Spracherzeugung durch eSpeak, die darauf beruht, dass jedes Wort als einzelner Satz erzeugt wird, lassen sich die Wörter sehr gut an Hand der Ruhephase zwischen den Wörtern segmentieren. Des Weiteren wird jeder Frame als relevante Sprachinformation betrachtet,

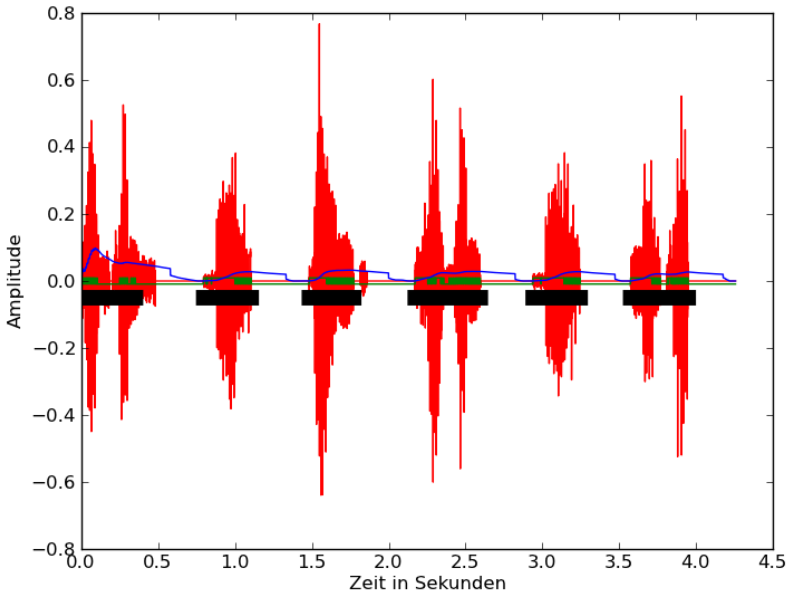

Abbildung 6.2: Kennzeichnung der Einzelworte in einem Idealdatensatz

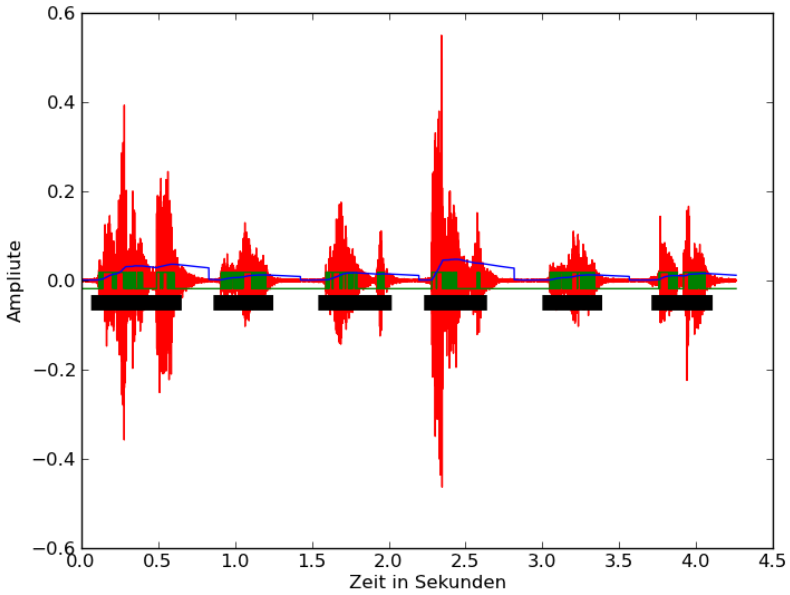

Abbildung 6.3: Kennzeichnung der Einzelworte in einer Originalaufnahme

dessen Signalwert mehr als doppelt so hoch wie der durchschnittliche Signalwert ist. Ebenfalls spielt die Durchschnittsberechnung eine Rolle, welche nur die letzten 6000 Signalwerte betrachtet, und die Aussortierung von einzelnen Peaks, auf die direkt eine längere Pause folgt.

Das Zusammenspiel der benannten Parameter ist sehr komplex und die daraus resultierenden kombinatorischen Möglichkeiten ließen im Rahmen dieser Arbeit nur eine experimentelle Bestimmung ihrer Belegung zu.

Die Grafiken 6.2 und 6.3 zeigen das Ergebnis der Wortdetektion. In rot ist jeweils die Signalfunktion eingezeichnet, grün sind die Bereiche, die von der Signalstärke her einem Wort zugeordnet werden können, und in schwarz sind die durch die State Machine tatsächlich detektierten Worte markiert. Abbildung 6.2 ist dabei mit Idealdaten erstellt, die auf dem eigenen Computer erzeugt und danach analysiert wurden. Abbildung 6.3 zeigt Realdaten, die mit dem Mikrophon im Roboter aufgenommen wurden. Beiden Abbildungen liegt ein Sprachinput von *„Distance. 3. Point. 7. 3. Meter.“* zu Grunde.

6.1.3 Evaluation der Signalwertverschiebung für die Sprachauswertung

Statt des DTW-Algorithmus wird in der vorgestellten Software eine lineare Verschiebung der Signalwerte verwendet. Die Abbildungen 6.4 und 6.5 zeigen die sich daraus ergebenden Abstände für alle verwendeten Verschiebungen und alle Referenzdaten. Abbildung 6.4 zeigt dabei die Analyse des Wortes *Distance*, Abbildung 6.5 die Analyse des Wortes *Seven*. Beide Analysen beruhen auf Idealdaten. In rot

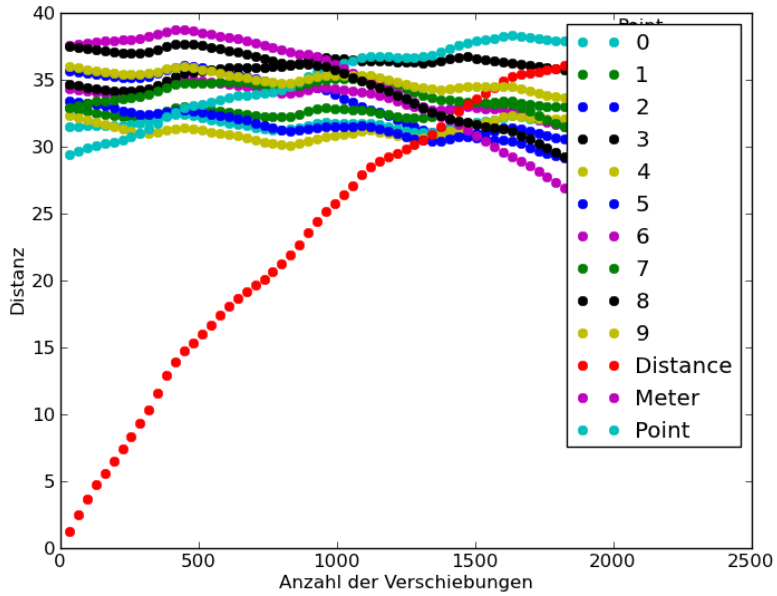

Abbildung 6.4: Distanzberechnungen für das Wort *„Distance“*

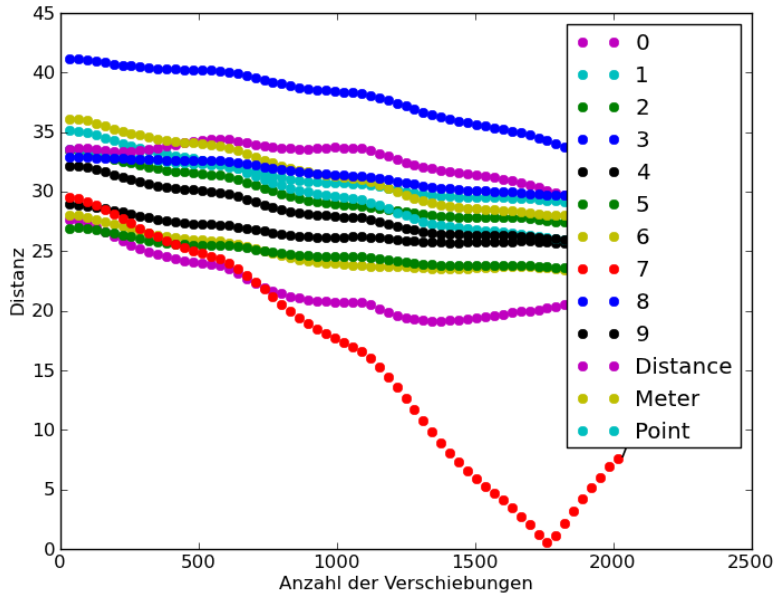

Abbildung 6.5: Distanzberechnung für das Wort *„Seven“*

ist der Distanzverlauf des jeweils korrekten Wortes eingezeichnet, in den anderen Farben die Distanzen aller anderen Referenzdaten.

Es zeigt sich, dass es jeweils in den abgebildeten Sektionen Bereiche gibt, in denen das korrekte Wort in der Distanz deutlich unter allen anderen liegt. Es gibt

aber auch solche Bereiche, in denen andere, nicht korrekte Worte eine geringere Distanz haben. Es wird deutlich, dass eine Verschiebung der Referenzdaten sinnvoll ist, weil abhängig von der Güte der Worterkennung nicht immer im selben Verschiebungsbereich das korrekte Referenzdatum die geringste Distanz aufweist. Somit stellt die Verschiebung sicher, dass trotz einer ungenauen Erkennung des Wortanfangs stets das Referenzdatum zum tatsächlich gesagten Wort die geringste Distanz aufweist, wenn das globale Minimum betrachtet wird.

6.1.4 Einfluss verschiedener Parameter auf die Worterkennung

Da das Verfahren zur Worterkennung auf einer Distanzberechnung der aufgenommenen Daten zu Referenzdaten beruht, ist die Erkennung um so besser, je größer die Distanz der Referenzdaten zueinander ist. Dieser Abstand wird vor allem durch die Größe der Fenster beeinflusst, in die das Signal aufgeteilt wird, sowie durch die Samplingrate, mit der die Sounddatei abgetastet wird. Die in 6.1.1 beschriebene Matrix ist mit einer Fenstergröße von 1024 Samples und einer Samplingrate von 44100 Hz entstanden.

Lässt man die Samplingrate fix bei 44.1 kHz und ändert lediglich die Fenstergröße, so hat dies die in Tabelle 6.6 gezeigten Auswirkungen auf den mittleren Abstand der Worte zueinander. Es zeigt sich deutlich, dass der mittlere Wortabstand mit steigender Fenstergröße ebenfalls vergrößert wird. Allerdings hat sich in der Praxis eine Fenstergröße ab einschließlich 2048 Samples nicht mehr als praktikabel erwiesen, da es Worte gibt, die kürzer sind, und damit die Worterkennungsrate wieder massiv sinkt.

Im zweiten Versuch wird die Fenstergröße unverändert bei 1024 Samples belassen und die Samplingrate verändert. In Tabelle 6.7 ist das Ergebnis des Versuchs dargestellt. Es zeigt sich, dass der mittlere Wortabstand bis zu einer bestimmten Samplingrate zusammen mit dieser deutlich ansteigt. Dies ist der Punkt der eigenen Samplingrate der Soundaufnahmen, der hier bei 22050 Hz liegt. Danach ist nur noch ein sehr schwacher Anstieg und zuletzt sogar ein Abfall der Abstände zu

Fenstergröße	**Mittlerer Wortabstand**
128 Samples	4,95
256 Samples	8,02
512 Samples	11,50
1024 Samples	16,02
2048 Samples	21,88
4096 Samples	29,74

Abbildung 6.6: Mittlerer Wortabstand bei einer Samplingrate von 44 kHz und variabler Fenstergröße

Samplingrate	Mittlerer Wortabstand
8000 Hz	12,72
11025 Hz	13,57
16000 Hz	14,48
22050 Hz	15,16
32000 Hz	15,79
44100 Hz	16,02
48000 Hz	16,04
96000 Hz	15,99

Abbildung 6.7: Mittlerer Wortabstand bei einer Fenstergröße von 1024 Samples und variabler Samplingrate

verzeichnen. Daraus lässt sich schließen, dass die optimale Samplingrate diejenige ist, die in der Sounddatei verwendet wurde. Wird eine wesentlich niedrigere Rate benutzt, wirkt sich dies negativ auf die Erkennungsrate der Wörter aus. Eine höhere Rate führt dagegen nicht zu einer nachhaltigen Verbesserung der Untersuchungsergebnisse.

6.1.5 Implementation einer Robotererkennung durch Frequenzanpassungen

Elementar bei den in dieser Arbeit präsentierten Dialogablaufstrategien ist, dass die Roboter bei der Analyse der Daten erkennen, welcher Roboter diese Daten gesprochen hat. Als Versuchsreihe wurden dazu als Referenzdaten pro Wort drei verschiedene Sounddateien erzeugt, jeweils mit einer unterschiedlichen Sprachfrequenz. Es zeigte sich, dass die Frequenzwerte so gewählt werden können, dass bei der Analyse von 600 Idealdatensätzen (200 pro Frequenzwert) zu 100 % der richtige Roboter als Sprecher erkannt werden konnte. In der Testreihe wurden die eSpeak-Pitches 30, 50 und 70 verwendet. Wie groß die Erkennungsrate der einzelnen Wortinhalte ist, ist nicht Gegenstand der Testreihe gewesen.

Im Ergebnis ist es möglich, den verschiedenen Roboter eine unterschiedliche Stimme zu geben und sich diese bei der Sprachanalyse zu Nutze zu machen.

6.2 Ergebnisse

6.2.1 Auswertung des gewählten Verfahrens für Idealdaten

Zu Testzwecken wurden mit Hilfe eines Python-Skripts 600 voneinander verschiedene Sounddateien der Art *„Distance. X. Point. Y. Z. Meter.“* erzeugt, wobei es sich bei X, Y und Z um ganzzahlige Werte zwischen 0 und 6 für X und zwischen 0

und 9 für Y und Z handelt. Das Testprogramm analysiert die eingegebenen Sounddateien, errechnet das Ergebnis und vergleicht dieses mit dem tatsächlichen Wort, das aus einer Textdatei ausgelesen wird.

Die Erkennungsrate für diese Sounddateien lag bei 100 %, es wurde kein einziges Wort falsch erkannt.

6.2.2 Übertragung des gewählten Verfahrens auf Originaldaten

Für die Übertragung des Verfahrens auf Originaldaten wurden zunächst neue Referenzdaten benötigt. Die Originaldaten sollen durch die Mikrophone des Roboters aufgenommen werden. Dazu wurden zwei Roboter in einer Entfernung von einem Meter zueinander aufgestellt, so dass der Lautsprecher des einen und das Mikrophon des anderen Roboters zueinander zeigten (siehe Abb. 6.8). Die auf die beschriebene Weise aufgenommenen Referenzdaten wurden mit Hilfe der Detektion von Wortanfang und Wortende und dem Programm Audacity so geschnitten, dass keine Samples außer denen zum Wort zugehörigen gespeichert wurden.

Es wurden weitere Aufnahmen mit dem gleichen Versuchsaufbau gemacht, wobei der eine Roboter nun über mehrere Minuten komplette Testsätze erzeugte und ausgab. Der andere Roboter speicherte diese ab, so dass sie später zur Analyse herangezogen werden konnten. Auch die Testdatensätze wurden mit Hilfe des Programms Audacity manuell voneinander separiert, wobei hier jedoch absichtlich zu Beginn und Ende unterschiedlich viel Pause belassen wurde. Sowohl die Referenzdaten als auch die Testdatensätze wurden per Rauschreduktion in Audacity vorverarbeitet.

Von den 100 Originaldatensätzen lag die Erkennungsrate bei 100 %. Dabei wurde nicht mit Robotern gearbeitet, die sich beim Sprechen und Zuhören bewegen, um die Umgebungsgeräusche gering zu halten. Allerdings wurden die Aufzeichnungen mit stehenden Robotern gemacht, so dass die Motoren- und Lüftergeräusche deutlich auf den Aufnahmen zu hören sind. Unter den momentan auf den Robo-

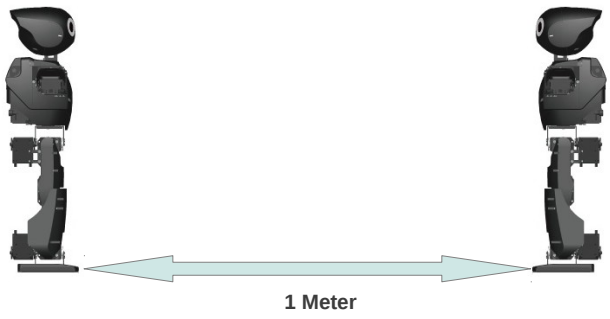

Abbildung 6.8: Versuchsaufbau bei der Aufnahme der Originaldatensätze und Referenzdaten.
DARwIn-OP Grafik von *http://www.robotis.com/img/img_ko/ sub/DARwIn_img_05.jpg*

tern gegebenen Hardwarebedingungen sind andere Testszenarien nicht erfolgversprechend (vgl. Kapitel 2.1).

6.2.3 Erforderlichkeit des DTW-Algorithmus

In Kapitel 4.3 wurde die These aufgestellt, dass eine Implementation des DTW-Algorithmus für die Roboter-Roboter-Kommunikation durch eine einfache lineare Verschiebung der Vergleichsmuster ersetzt werden kann.

Wie in den vorhergehenden Kapiteln 6.2.1 und 6.2.2 beschrieben, liefert der gewählte Algorithmus Erkennungsraten von 100 %. Eine lineare Verschiebung der Sprachsignale ist somit tatsächlich ausreichend.

Die Verwendung einer linearen Verschiebung hat ihren Vorteil gegenüber dem DTW-Algorithmus in der Laufzeit begründet: Der Dynamic-Time-Warping-Algorithmus hat eine Rechenkomplexität im Bereich $O(N^2)$, also eine quadratische Laufzeit. Zwar gibt es Ansätze, einen Algorithmus basierend auf dem DTW in linearer Laufzeit zu schaffen (Fast-DTW), allerdings ist in diesen Ansätzen bisher nur eine im Optimalfall näherungsweise lineare Rechenkomplexität erzielt worden. Im Gegensatz zum DTW-Algorithmus findet der Fast-DTW auch nicht in jedem Fall das optimale Ergebnis [20]. Die lineare Verschiebung demgegenüber kommt mit einer Laufzeit von $O(N)$ aus, da in einer einfachen Schleife nur jeder achte Wert einmalig verglichen wird. Gerade unter den gegebenen Hardwareeinschränkungen der Roboter lässt sich hier eine wertvolle Einsparung an Rechenzeit erreichen.

6.2.4 Laufzeitanalyse

Auf dem Laptop, auf dem die Analysen gemacht wurden (DELL INSPIRON N5110, Intel Core i7-2670QM CPU, 2.20GHz x 8), wurde pro Satzanalyse eine Zeit zwischen einer und drei Sekunden benötigt. Dies liegt noch unter der Laufzeit der Sounddateien, die etwa vier Sekunden lang sind.

Eine so geringe Laufzeit kann auf dem Roboter durch die Hardwareeinschränkungen aktuell nicht erreicht werden. Das Programm bietet jedoch reichlich Optimierungspotential, nicht zuletzt durch die Verwendung einer hardwarenäheren Programmiersprache für die Vektorberechnungen wie C++. Ein großer Zeitfaktor in der Analyse ist die häufige Verschiebung der Signalwerte, nach der jeweils wieder das Mel-Cepstrum gebildet wird und eine Distanzberechnung zu allen Referenzdaten erfolgen muss. Hier ist mit deutlich weniger Verschiebungen auszukommen, wenn man die Struktur der sich ergebenen Distanzwerte, wie in Abschnitt 6.1.3 beschrieben, mitbetrachtet. Es wäre möglich, mit wenigen Datenpunkten das jeweilige Minimum pro Referenzdatum zu bestimmen und nur die jeweiligen Minima miteinander zu vergleichen. Welche weiteren Laufzeitverbesserungen erzielt werden können und welche minimale Laufzeit erzielt werden kann, ist in weitergehenden Arbeiten zu untersuchen.

6.3 Zusammenfassung

Diese Arbeit stellt sowohl theoretische Ansätze für die Hardwareanpassung der DarwinOP-Roboter für die Nutzung von Kommunikation durch natürliche Sprache, Dialogstrategien und eine Eingliederungsstrategie in das bestehende Softwaresystem des RoboCup-Teams *Hamburg Bit-Bots* dar, als auch eine praktische Implementation vor, die die folgenden Anforderungen erfüllt:

- Abstraktion vom DTW-Algorithmus durch lineare Verschiebung der Signalwerte.
- Leichte Erweiterbarkeit auf neue Anwendungsszenarien durch Ergänzung der Referenzdatensätze.
- Analyse der Mel-Cepstrum-Parameter.
- Verwendbarkeit des Systems mit Realdaten des Roboters.
- Leichte Test- und Analysierbarkeit durch Simulation des Audiostreams.
- Ressourcensparende Speicherung der Referenzdaten in XML-Dateien.
- Unterstützung einer Sprechererkennung durch Erzeugung von Sprache in verschiedenen Frequenzbereichen.

Kapitel 7

Ausblick

Aufbauend auf der vorliegenden Arbeit ergeben sich einige Forschungsansätze, die in weiterführenden Arbeiten zu betrachten sind: Die hier vorgestellte Erkennung von Sprachanfang und Sprachende setzt eine relativ ruhige Umgebung voraus, die in einer realen Spielsituation bei RoboCup-Turnieren nicht gegeben ist. Denkbar wäre hier beispielsweise die Erweiterung des vorgestellten Ansatzes, wie in [6] beschrieben. Andere Ansätze, wie die Verwendung einer *Spektralmaske*, einer *Geometric Source Separation (GSS)* oder einer *Computational Auditory Scene Analysis (CASA)* sind bereits in Zusammenhang mit mobilen Robotern getestet worden, so dass eine Umsetzung für das Fußballspiel denkbar wäre [5][15][24].

Darüber hinaus ist eine Laufzeitoptimierung notwendig, um das Verfahren im Spiel unter den Hardwareeinschränkungen der Roboter einsetzen zu können. Ansätze dafür sind bereits in Kapitel 6.2.4 beschrieben. In den Tests mit Originaldaten wurde stets eine manuelle Rauschreduktion verwendet. Um eine autonome Erkennung auf dem Roboter zu ermöglichen, muss eine solche Rauschfilterung unter den gegebenen Anforderungen an Hardware und Laufzeit automatisiert werden.

Diese Arbeit bietet weiterhin globalere Ansatzpunkte, das Spielverhalten im RoboCup zu verbessern. Zunächst liegt natürlich eine Erweiterung auf die Analyse von menschlicher Sprache nahe. Die W-LAN-Netzwerkarchitektur würde so nicht nur bei der Roboter-Roboter-Kommunikation überflüssig, auch der elektronische Schiedsrichter könnte so ersetzt werden. Die Schiedsrichteranweisungen sind deutlich durch ihre Verbindung mit der Benutzung der Pfeife zu erkennen. Durch das klar definierte Regelwerk mit eindeutigen Bezeichnungen für Fouls und andere Spielsituationen ist eine gemeinsame Sprachbasis bereits gegeben und muss nur noch in die Roboter implementiert werden.

Eine Mensch-Roboter-Kommunikation im RoboCup bringt weitere Vorteile mit sich: Wie im menschlichen Fußball auch könnte es einen Trainer am Spielfeldrand geben, der taktische Anweisungen gibt. Um hier jedoch einen Missbrauch durch gegnerische Teams oder das Publikum zu unterbinden, sind weitere Vorkehrungen wie die Sprecheridentifikation nötig. Eine andere Möglichkeit bietet die Analyse der Stimmen im Publikum. Schon heute bringen sich insbesondere die Kinder am Spielfeldrand in das Spiel ein, in dem sie Hinweise wie: „Schieß!“ oder „Da liegt der Ball!“ geben. Könnte der Roboter diese Hinweise analysieren, könnte er beispiels-

weise den Bereich, in dem nach dem Ball zu suchen ist, viel genauer eingrenzen. Problematisch ist auch hier wieder, dass Potential für Missbrauch eröffnet wird.

Doch auch ohne die Implementation einer Mensch-Roboter-Kommunikation können neue Wege in einigen Verfahren beschritten werden. Beispielhaft sei hier die Lokalisation auf dem Spielfeld angeführt. Zwar ist dem Roboter eine exakte Lokalisation an Hand der Feldlinien und den Vorinformationen möglich, jedoch kann es im Spielverlauf dazu kommen, dass ein Roboter neu oder nach einer Pause wieder in das Spiel eingesetzt wird und keinerlei Orientierung hat. Durch die Symmetrie des Spielfelds ist es ihm nun nicht möglich zu bestimmen, auf welcher der beiden Spielfeldhälften sein eigenes Tor steht. Bisherige Ansätze beschäftigen sich vor allem mit einer Analyse des Horizonts um das Spielfeld herum. Die Roboter-Roboter-Kommunikation im Zusammenhang mit der Geräuschquellenlokalisation eröffnet hier noch einen anderen Verfahrensansatz: Der orientierungslose Roboter gibt einen Hilferuf ab, auf den beispielsweise der Torwart im eigenen Team reagiert. Durch die Lokalisation des Sounds zusammen mit der Verifikation des Torwarts durch die Sprachanalyse kann so leicht festgestellt werden, in welchem Tor der eigene Torwart steht.

Diese Arbeit soll die Grundlage für eine kontextübergreifende Forschung im Bereich Roboter-Roboter-Kommunikation bilden, so dass – wie bereits in Kapitel 1 dargelegt – auch weitere Forschungsansätze über das Fußballspiel hinaus entstehen, beispielsweise im Bereich der mobilen Serviceroboter.

Literaturverzeichnis

[1] eSpeak.
http://espeak.sourceforge.net/.
GNU General Public License Version 3, Last Access 07.06.2013 23:51.

[2] Was versteht man unter Prosodie?
http://www.uni-bielefeld.de/Universitaet/Einrichtungen/Zentrale%20Institute/IWT/FWG/Sprache/Prosodie.html.
Last Access 08.06.2013 22:35.

[3] Hans-Dieter Burkhard and Hans-Arthur Marsiske. *Endspiel 2050: Wie Roboter Fußball spielen lernen*. Heise, 2003. ISBN 9783936931020.

[4] Kai-Uwe Carstensen, Christian Ebert, Susanne Jekat, Cornelia Ebert, Hagen Langer, and Ralf Klabunde. *Computerlinguistik und Sprachtechnologie: Eine Einführung*. Spektrum Lehrbuch. Spektrum Akademischer Verlag GmbH, 2010. ISBN 9783827422248.

[5] Antoine Deleforge and Radu Horaud. The cocktail party robot: Sound source separation and localisation with an active binaural head. *IEEE/ACM International Conference on Human Robot Interaction*, 2012.

[6] Masrur Doostdar, Stefan Schiffer, and Gerhard Lakemeyer. RoboCup 2008: Robot Soccer World Cup XII. chapter *A Robust Speech Recognition System for Service-Robotics Applications*, pages 1–12. Springer-Verlag, Berlin, Heidelberg, 2009.

[7] Michael Dürr and Peter Schlobinski. *Deskriptive Linguistik*. Studienbücher zur Linguistik. Vandenhoeck + Ruprecht Gm, 2006. ISBN 9783525265185.

[8] L. Dybkjaer, H. Hemsen, and W. Minker. *Evaluation of text and speech systems*. Text, speech, and language technology. Springer London, Limited, 2007. ISBN 9781402058172.

[9] Kristiina Jokinen and Michael McTear. *Spoken Dialogue Systems*. Synthesis Lectures on Human Language Technologies Series. Morgan & Claypool, 2010. ISBN 9781598295993.

[10] Hartmut König. *Protocol Engineering*. Springer, 2012. ISBN 9783642291449.

[11] Angelika Linke, Markus Nussbaumer, and Paul R. Portmann. *Studienbuch Linguistik*. Reihe Germanistische Linguistik, 121. Niemeyer Max Verlag GmbH, 2004.

[12] Beth Logan. Mel frequency cepstral coefficients for music modeling. *International Symposium on Music Information Retrieval*, 2000.

[13] Dominic Mazzoni and Roger Dannenberg. Audacity.
http://audacity.sourceforge.net/?lang=de.
GNU General Public License Version 2, Last Access 07.06.2013 23:54.

[14] Karl J. Muecke and Dennis W. Hong. Darwin's evolution: development of a humanoid robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2574–2575, 2007.

[15] Hiroshi G. Okuno, Tetsuya Ogata, and Kazunori Komatani. Computational auditory scene analysis and its application to robot audition: Five years experience. In *Second International Conference on Informatics Research for Development of Knowledge Society Infrastructure*, pages 69–76. 2007.

[16] Beate Pfister and Tobias Kaufmann. *Sprachverarbeitung: Grundlagen und Methoden Der Sprachsynthese und Spracherkennung*. Springer, 2008. ISBN 9783540759102.

[17] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1989.

[18] Christian Rentrop. Roboter-Angst im europäischen Kulturkreis.
http://www.netzwelt.de/news/73191_3-verkehrte-netzwelt-honda-asimo-terminator-alpha-version.html, 12/2005.
Last Access 07.06.2013 23:34.

[19] Robotis. DARwIn-OP Subcontroler.
http://sourceforge.net/projects/darwinop/files/Hardware/Electronics/Sub%20Controller/.
Last Access 07.06.2013 23:58.

[20] S. Salvadore and P. Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *3rd Workshop on Mining Temporal and Sequential Data*, 2004.

[21] Klaus Seyerlehner. Mel Frequency Cepstrum.
SVN *http://mirlastfm.googlecode.com/svn/trunk/ mirlastfm-read-only*.
GNU General Public License Version 3, Last Access 07.06.2013 23:53.

[22] Klaus Seyerlehner. *Content-Based Music Recommender Systems: Beyond simple Frame-Level Audio Similarity*. 2010.

[23] Mark Tatham and Katherine Morton. *Developments in speech synthesis.* John Wiley, 2005. ISBN 9780470855386.

[24] Jean-Marc Valin, Seiichi Yamamoto, Jean Rouat, François Michaud, Kazuhiro Nakadai, and Hiroshi G. Okuno. Robust recognition of simultaneous speech by a mobile robot. *IEEE Transactions on Robotics (4)*, pages 742–752.

[25] Jutta Weber. Der Roboter als Menschenfreund. *c't*, (Heft 2):144 – 149, 2006.

Abbildungsverzeichnis

Anhang

Programmbeispiele ausführen

Der Quelltext als Maven-Projekt kann unter
https://github.com/MaikePaetzel/RobotToRobotCommunication
heruntergeladen werden und steht unter der CC BY-NC-SA 3.0 DE Lizenz.

Konsolenbenutzung (unter Linux und Mac)

Voraussetzung: Eine Installation von *Java JDK 1.7* und *Maven 3* ist Voraussetzung für die Ausführung des Programms.

Kompilierung: Unter Linux und Mac kann der Quelltext über die Konsole durch die Eingabe der Zeile „*mvn clean install*“ im Git-Verzeichnis kompiliert werden.

Ausführung: Im Hauptordner liegt nach der erfolgreichen Kompilierung ein zusätzlicher Ordner *target*. Die Ausführung des Programms geschieht im Hauptordner des Git-Verzeichnisses, in dem die Datei *BA.jar* liegt, durch Eingabe der Zeile „*java -jar BA.jar -s #nummer*“, wobei #nummer durch eine natürliche Zahl zwischen Null und Sieben ersetzt werden muss. Die Bedeutung der einzelnen Szenariennummern ist im Abschnitt „Ausführbare Klassen“ erklärt.

Einbindung in eine IDE am Beispiel Eclipse (für alle Betriebssysteme)

Voraussetzung: Es wird vorausgesetzt, dass die *Maven Integration für Eclipse* installiert ist. Diese kann jederzeit über den *Eclipse-Marketplace* kostenlos hinzugefügt werden.

Import:

1. Unter Eclipse das Import-Fenster öffnen.
2. Import-Quelle ist ein *existierendes Maven-Projekt* aus dem übergeordneten Ordner *Maven*.

3. Das Wurzelverzeichnis ist der Ordner, in dem die Datei *pom.xml* liegt. Diese muss auch als Projekt ausgewählt werden.

Ausführbare Klassen

StartUp: In der Klasse StartUp können alle in der Bachelorarbeit beschriebenen statistischen Auswertungen gestartet werden. Dazu sind folgende Parameter bei der Eingabe erlaubt:

- **-s 0** analysiert eine einzelne Originalaufnahme auf den Sprachinhalt. Welche Datei verwendet wird, kann in der entsprechenden Klasse manuell geändert werden. Beim initialen auschecken des Repositories wird stets eine Datei mit dem Inhalt *„Distance. 3. Point. 7. 3. Meter.“* gewählt.
- **-s 1** analysiert den Ordner *Micro/Microphondaten* mit 100 Originalaufnahmen auf den Sprachinhalt.
- **-s 2** analysiert eine einzelne eSpeak-Idealdatei auf den Sprachinhalt. Welche Datei verwendet wird, kann in der entsprechenden Klasse manuell geändert werden. Beim initialen auschecken des Repositories wird stets eine Datei mit dem Inhalt *„Distance. 3. Point. 7. 3. Meter.“* gewählt.
- **-s 3** analysiert den Ordner *eSpeak/eSpeakdaten* mit 600 Idealdaten auf den Sprachinhalt.
- **-s 4** analysiert den Ordner *Pitch/RoboterErkennungsdaten* mit 600 Idealdaten verschiedner Pitches darauf, mit welchem Pitch die Aufnahme erzeugt wurde.
- **-s 5** gibt eine Matrix mit den Distanzwerten zurück, die jede Datei im Referenzdatensatz auf Idealdaten im Vergleich zu jedem anderen Referenzdatensatz hat.
- **-s 6** gibt eine Matrix mit den Distanzwerten zurück, die jede Datei im Referenzdatensatz auf Originaldaten im Vergleich zu jedem anderen Referenzdatensatz hat.
- **-s 7** gibt eine Matrix mit den Distanzwerten zurück, die jede Datei im Referenzdatensatz auf Idealdaten im Vergleich zu jedem anderen Referenzdatensatz auf Originaldaten hat.

Bei den Szenarien 1, 3 und 4 werden am Ende der Analyse die Testergebnisse in die Datei *Testergebnis/ausgabe.txt* geschrieben. Alle Pfade zu Sound- oder Textdateien liegen immer im Ordner *resources*.

CreateXMLFile: Diese Klasse nimmt als Parameter den Pfad zu einem Ordner mit Referenzdaten und den Namen für die zu erzeugende XML-Datei inklusive relativ zum Projektverzeichnis liegender Pfadangabe entgegen und erzeugt eine XML-Datei aus dem Referenzdatenordner.

Erklärung der Urheberschaft

Ich versichere, dass ich die Bachelorarbeit im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift