Technische Universität Berlin
Fakultät II: Mathematik und Naturwissenschaften

Bachelorarbeit

Zum Erwerb des akademischen Grades
Bachelor of Science

Mit Sprache programmieren - Grundlagen für ein interaktives Audiointerface

von:
Simon Untergasser
342688

Abgabedatum:
25. September 2017

1. Gutachter:
Prof. Dr. K.-R. Müller - Technische Universität Berlin
2. Gutachter und Betreuer:
Prof. Dr. M. Hild - Beuth Hochschule für Technik Berlin

Kurzfassung

Die Programmierung des humanoiden Roboters Myon soll in zunehmendem Maße über gesprochene Sprache erfolgen. Um dieses Ziel zu erreichen, werden in dieser Arbeit die Grundlagen für eine neuartige Herangehensweise an die Sprachverarbeitung vorgestellt. Dieser Ansatz besteht darin, gesprochene Wörter hinsichtlich ihrer Energie zu betrachten. Die Silben in Wörtern und Sätzen besitzen eine typische Energiehierarchie, die sich für den Vergleich von Wörtern nutzen lässt. Dazu werden Prozesse entwickelt, die Energiemaxima innerhalb eines Wortes identifizieren. Außerdem werden geeignete Datenstrukturen angelegt, um diese zu speichern. Durch die Repräsentation von Wörtern über ihre Energiemaxima werden Wörter im Speicher effizient gesucht und Sprachverarbeitung wird in einem autonomen System ermöglicht.

Abstract

The programming of the humanoid robot Myon should increasingly be carried out by spoken language. In order to achieve this goal, the basis for a novel approach to language processing is presented in this thesis. This includes to consider spoken words from the aspect of energy. The syllables in words and sentences have a typical energy hierarchy, which can be used to compare words. For this purpose, processes are developed that identify energy maxima within a word and create suitable data structures in order to store them. By representing words by their energy maxima, words in memory are efficiently searched for and language processing in an autonomous system is made possible.

Die selbstständige und eigenhändige Anfertigung versichere ich an Eides statt.

_____

Datum / Unterschrift

Inhaltsverzeichnis

1 Einleitung

Für viele Lebewesen stellen auditive Signale eine wichtige Informationsquelle dar. Sie warnen rechtzeitig vor Feinden und werden für die Kommunikation untereinander genutzt. Auch Menschen nehmen mit Hilfe des Hörsinns ihre Umwelt wahr, können auf Ereignisse reagieren und über Sprache mit ihr interagieren. In Verbindung mit der Fähigkeit komplexe Muster zu erkennen, zu interpretieren und mit Bekanntem zu vergleichen, haben sich im Laufe der menschlichen Evolution verschiedene Arten der Kommunikation entwickelt [**?**]. Heute sind Menschen in der Lage, über Sprache komplexe Sachverhalte zu kommunizieren und Dinge für andere vorstellbar zu machen. Durch die Entwicklung von Schriftformen wurde es dem Menschen möglich, sprachliche Informationen zu speichern und zu übermitteln. Auch im Zeitalter digitaler Technik spielt die Schrift als Kommunikationsmittel zwischen Mensch und Maschine eine entscheidende Rolle. Mittels einer Tastatur kann geschriebene Sprache in digitale Signale umgesetzt werden. Seit Beginn des Computerzeitalters versucht man außerdem, über gesprochene Sprache mit den Geräten zu kommunizieren. Wie in [**?**] zusammengefasst, werden schon seit der ersten Hälfte des 19. Jahrhunderts erste Erfolge in der Sprachverarbeitung erzielt. Über viele Meilensteine, wie die Entwicklung von Spektralanalysen oder Hidden Markov Modellen (Originalquelle: [**?**]), wurde die Anzahl der von Maschinen erkannten Wörter immer größer. In heutigen Systemen werden Fehlerraten von wenigen Prozent erreicht (siehe [**?**]). Mit hohem Daten- und Rechenaufwand werden Maschinen trainiert, Sprachsignale zu analysieren und zum Beispiel in Textausgabe umzuwandeln. Diese Systeme dienen hauptsächlich als Benutzerschnittstelle und übersetzen gesprochene Sprache in digitale Signale.

Auch in der Robotik ist die Verarbeitung von Sprache ein wichtiges Thema. Zum einen, um die Maschinen zu steuern, also die Kommunikation zwischen Mensch und Maschine zu vereinfachen. Zum anderen, kann Sprachverarbeitung dabei helfen, Roboter autonomer zu machen. So ist es vorstellbar, dass Roboter zukünftig in der Lage sein werden, gesprochene Worte zu verarbeiten, tatsächlich zu verstehen und in Handlungen umzusetzen.

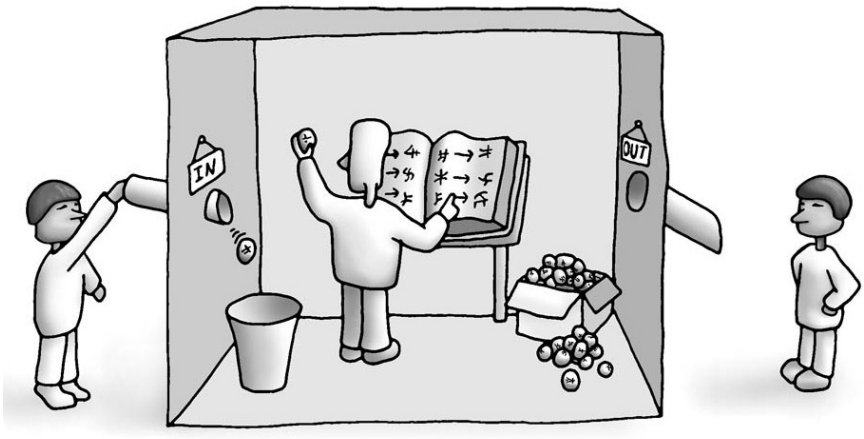


Abbildung 1.1: John Searles Gedankenexperiment „Chinese Room“. Eine Person in einem Raum kreiert eine Ausgabe aus chinesischen Schriftzeichen, die aus einer Eingabe folgt, ohne der Sprache mächtig zu sein. Sie folgt dabei nur den Regeln in einem Buch. Für Beobachter außerhalb des Raumes erscheint es, als beherrsche die Person im Raum Chinesisch. *Quelle: www.berto-meister.blogspot.com*

1.1 Sprachverarbeitung in einem autonomen System

Die im vorangehenden Abschnitt erwähnten Erfolge in der maschinellen Sprachverarbeitung spiegeln sich in Produkten wie Apples „Siri“ oder „Google Now“ wieder. Hier werden Maschinen und Algorithmen als Werkzeuge genutzt, um auditive Signale in digitale, schriftliche Form zu bringen. Das Audiosignal wird analysiert und nach bestimmten vordefinierten Regeln ein Ergebnis zurückgegeben. Der Ablauf erinnert an das „Chinesische Zimmer“ von John Searle [**?**]. In Abbildung 1.1 sieht man eine Person in einem geschlossenen Raum, die kein Chinesisch beherrscht. Diese erhält durch die „Eingabe“ auf der linken Seite einen Satz aus chinesischen Schriftzeichen. Sie erstellt darauf aufbauend die Ausgabe, indem sie Regeln befolgt, die in dem Buch festgelegt sind. Für Beobachter außerhalb des Raumes, die nur die Eingabe und die Ausgabe sehen, erscheint es, als beherrsche die Person innerhalb des Raumes Chinesisch. Ähnlich verhält es sich mit den vorgenannten Systemen. Der Input wird analysiert und anhand von teilweise gelernten und teilweise vorgegebenen Regeln in eine Ausgabe übersetzt. Um dies zu erreichen, werden unterschiedliche Ansätze benutzt: Google verwendet für die Spracherkennung neuronale Netze. Apples Siri verwendet ein auf „Hidden Markov Modellen“ basierendes System. Beide Modelle werden mit einer großen Anzahl von Beispielsätzen trainiert. Dadurch kann ein Sprachmodell entwickelt werden, das die Wahrscheinlichkeit eines Wortes anhand der spektralen Eigenschaften (Spektrum, siehe Kapitel 4)

und der Stellung im Satz vorhersagen kann. Trotz der Erfolge und des praktischen Nutzens dieser Systeme, sind für die vorliegende Arbeit andere Techniken vonnöten. Dies liegt zum einen am Embodiment Ansatz [1], der in dem Roboter Myon umgesetzt wird (näher erläutert in Abschnitt 2.1) und zum anderen an den wechselnden Kontexten der Kommunikationssituationen (Kapitel 3). Wie in Kapitel 2 genauer beschrieben wird, ist Myon ein humanoider Roboter und in seinen körperlichen Fähigkeiten an denen eines Menschen angelehnt. Myon unterscheidet sich in einigen seiner Charakteristika von etablierten Sprachsystemen wie Siri. Auf die Internetverbindung, die bei Systemen auf dem Smartphone benötigt wird (siehe [**?**] für eine Evaluation unter verschiedenen Netzwerkbedingungen), wird bewusst verzichtet. Eine Verarbeitungszeit von mehreren hundert Millisekunden, die für die Übertragung der Daten benötigt wird, ist für ein autonomes System zu lang. Der Roboter muss unter Umständen sehr schnell auf Reize reagieren. Ein fertiges, lokal operierendes Spracherkennungssystem hochzuladen widerspricht den Prinzipien des Forschungslabors Neurorobotik (siehe Abschnitt 2.1). Die Spracherkennung sollte zudem adaptiv sein um den Randbedingungen eines autonomen Systems in einer sich verändernden Umwelt gerecht zu werden. Des Weiteren bezieht sich der Kommunikationsinhalt oft auf eine bestimmte Umweltsituation, zum Beispiel Objekte, die beschrieben werden oder eine bestimmte Handlungsaufforderung. In anderen Situationen geht es um den Roboter selbst, seien es Winkelwerte der Motoren oder bestimmte Bewegungssabläufe. Hier spielen also andere perzeptive Teilsysteme mit in die Kommunikation ein. So kann das Wissen über die Umwelt, das zum Beispiel aus der visuellen Verarbeitung entstanden ist, die Spracherkennung ergänzen.

1.2 Stand der Forschung

Die humanoide Robotik ist ein aktives Forschungsfeld. [**?**] bietet eine Übersicht über Eigenschaften, die einen humanoiden Roboter ausmachen. Außerdem wird die vergangene, die aktuelle und die mögliche zukünftige Entwicklung in diesem Forschungsgebiet beleuchtet. Zusätzlich ist eine Auflistung von einigen historisch wichtigen humanoide Roboter geboten. Besonders hervorzuheben ist der Roboter „Nao“ von Aldebaran Robotics (entwickelt seit 2004). Seit 2011 ist er frei erhältlich und wird von vielen Forschungsinstituten als Plattform genutzt. Da er vier Mikrofone besitzt, eignet er sich auch für die Forschung zur Spracherkennung. Beispielsweise wurde in [**?**] ein System basierend auf Hidden Markov Modellen entwickelt. Der etwas neuere Roboter „Pepper“, ebenfalls von Aldebaran Robotics (vorgestellt 2014), wird als kommunikativer Roboter beschrieben (siehe www.softbankrobotics.com). Durch die Analyse von Stimme, Körpersprache und Ausdruck im Gesicht, kann er grundlegende Emotionen von Menschen erkennen und sein Verhalten dementsprechend anpassen. Ein weiterer inter-

[1] Um die reale Umwelt zu erfahren wird ein Körper benötigt. Durch das Wechselspiel zwischen Sensorik und Motorik und der Interaktion mit der Umwelt kann intelligentes Verhalten entstehen

essanter Ansatz wird in [**?**] beschrieben. Drei Designprinzipien werden vorgestellt. Diese Prinzipien sind für autonome Roboter wichtig sind, die sich in komplexen Umgebungen zurechtfinden müssen. Zunächst wird das Prinzip des Embodiment als essentiell gesehen. Des Weiteren wird hervorgehoben, dass Sensorinformationen vom System in geeigneter Weise organisiert und verarbeitet werden müssen. Die Datenfülle sei nur durch sich selbst organisierende Datenstrukturen zu meistern. Außerdem sollen Daten lokal verarbeitet werden. Ähnlich wie bei menschlichen Reflexen, müssen manche sensorischen Informationen nicht erst bis zur zentralen Recheneinheit geleitet werden, um motorische Reaktionen hervorzurufen. Diese Prinzipien sind auch für den Roboter Myon von Bedeutung und werden im Forschungslabor Neurorobotik angewendet.

1.3 Ziel der Arbeit

Ausgehend von der Fragestellung: *Welche Prozesse können einem autonomen Roboter als Grundlage für ein Kommunikationssystem dienen, was müssen diese leisten und wie können sie implementiert werden?* sollen in der vorliegenden Arbeit Werkzeuge für ein Kommunikationssystem entwickelt werden, die es dem Roboter ermöglichen, eine eigene Sprachfähigkeit aufzubauen. Dies ist ein mehrstufiger Prozess. Am Anfang steht die Analyse der auditiven Daten. Es müssen grundlegende perzeptive Systeme entwickelt werden, um diese Daten aufnehmen und analysieren zu können. Anschließend werden diese Analysen in eine vergleichbare Form gebracht (siehe Kapitel 4). Dazu werden auf Ebene der Laute immanente Merkmale extrahiert. Diese werden dem Roboter als perzeptive Erfahrungen zur Verfügung gestellt und er kann sie in sein Weltbild integrieren. Alle Mechanismen und Algorithmen werden im Hinblick darauf entwickelt, dass sie adaptiv und flexibel einsetzbar sind. Schlussendlich soll der Roboter in der Lage sein, einen Sprachschatz von Wörtern aufzubauen, die er mit Informationen aus anderen perzeptiven Kanälen verknüpfen kann. Damit kann der Roboter Bedeutungen von Wörtern lernen und beispielsweise Objekte oder Menschen, mit denen er interagiert, mit Gehörtem verknüpfen.

Für diese Prozesse sind zwei Gruppen von Wörtern relevant. Dies sind zum einen die eben beschriebenen Wörter, die der Erfahrungswelt des Roboters Bedeutung zuordnen und Informationen über die Umwelt liefern. Die zweite Gruppe beinhaltet Wörter, die bestimmte Prozesse innerhalb des Roboters aktivieren, für ihn selbst aber nicht unbedingt erfahrbar sind. Hier genügt ein begrenzter Wortschatz, der nicht gelernt werden muss und als Metasprache (siehe Abschnitt 3.1) für spätere neu erlernte Wörter und damit dem Sprachverständnis dienen kann. Diese Gruppe von Wörtern muss möglichst fehlerfrei und zuverlässig erkannt werden.

Aufbau der Arbeit

Die in der vorliegenden Arbeit beschriebenen Konzepte und Algorithmen wurden zum Einsatz auf dem humanoiden Roboter Myon entwickelt. Dieser wird im folgenden Kapitel mit allen relevanten Eckdaten und technischen Details vorgestellt. Anschließend werden im Kapitel 3 die wichtigsten Kommunikationssituationen vorgestellt. Außerdem wird die sprachliche Programmierung näher beleuchtet und die für ein Kommunikationssystem benötigten Prozesse werden herausgearbeitet. Die Anforderungen an das Kommunikationssystem bilden den Abschluss dieses Kapitels. Darauf aufbauend wird in Kapitel 4 die technische Umsetzung dieser Ideen im Roboter Myon dargestellt. Die Bewertungen des praktischen Einsatzes des Systems und die Resultate werden in Kapitel 5 zusammengefasst. Außerdem werden Ausblicke auf mögliche Weiterentwicklungen gegeben. Kapitel 6 schließt mit einer Zusammenfassung.

2 Roboterplattform Myon

Im Folgenden wird der humanoide Roboter Myon vorgestellt. Er dient als Forschungsplattform und als grundlegende Hardware für die in der vorliegenden Arbeit entwickelten Ideen und Konzepte. Es wird kurz auf die Entwicklung und den Aufbau des Roboters eingegangen, um dann etwas genauer die Begriffe Embodiment und Situatedness zu beschreiben. Danach folgt ein Überblick über die für die Perzeption wichtigen Teilsysteme. Und abschließend wird die Softwareentwicklungsumgebung vorgestellt.

2.1 Myon - Ein modularer Roboter

Myon (Abb. 2.1) ist im Rahmen des EU-Projekts ALEAR[1] [**?**] im Forschungslabor Neurorobotik (NRL) an der Beuth Hochschule für Technik Berlin entwickelt und gefertigt worden. Er ist modular aufgebaut und kann durch die verteilte Rechenleistung und Energieversorgung während der Laufzeit montiert und demontiert werden. Er besteht

Abbildung 2.1: Der humanoide Roboter Myon

[1] ALEAR: Artificial Language Evolution on Autonomous Robots

aus zwei Beinen, zwei Armen, zwei Händen, dem Torso und dem Kopf. Diese Modularität erlaubt es, die einzelnen Körperteilen eigenständig zu betreiben. So können die Mitarbeiter des Forschungslabors an den verschiedenen Körperteilen unabhängig voneinander arbeiten. Der Roboter ist $1,25$m hoch und etwa 16kg schwer. Im Kopf befinden sich eine zentrale Recheneinheit (genannt Brainmodule), eine Kamera und zwei Mikrofone, die dem Roboter als Ohren dienen. Auf Myon werden neuronale Regelschleifen und Algorithmen zur Verhaltenssteuerung entwickelt. Es gehört zum Konzept des NRL, dass alle Berechnungen des Roboters zur Laufzeit und lokal erfolgen. Da er außerdem autonom agieren soll und nur begrenzte Rechenleistung und Energie zur Verfügung hat, sind effiziente und robuste Algorithmen gefordert.

Embodiment und Situatedness

Laut Auffassung der Mitarbeiter des Forschungslabors Neurorobotik entsteht intelligentes Verhalten in der Interaktion mit der Umwelt. Dazu benötigt ein Roboter einen physischen Körper. Der Begriff Embodiment (dt. etwa Verkörperlichung) fasst zusammen, dass Erfahrungen in der realen Welt die Voraussetzung für intelligentes Verhalten und Bewusstsein sind. Er kommt aus der Psychologie und beschreibt Wahrnehmung als sensomotorischen Vorgang. Laut dieser Perspektive wirken Psyche und Körper aufeinander und beeinflussen sich gegenseitig. Die unterschiedlichen Auffassungen des Begriffs werden von Ziemke übersichtlich zusammengefasst und diskutiert [**?**]. Auf die Arbeit mit Robotern übertragen, gilt, dass Simulationen zwar hilfreiche Ergebnisse liefern können, diese aber nur in geringem Maße auf die Realität anwendbar sind. Deshalb ist es unerlässlich, an einem realen Roboter zu arbeiten, der potentiell in der Lage ist, seinen Körper wahrzunehmen. Situatedness (dt. etwa situiert, in einer Situation platziert) kann als Teilaspekt des Embodiment aufgefasst werden. Durch den physischen Körper ist der Roboter immer in eine Umgebung eingebettet. Dies bestimmt die Art der Interaktionen, die in bestimmten Situationen möglich sind. Lernen besteht im immer besser werdenden Umgang mit diesen Situationen. So wird kontextabhängiges Wissen angereichert und intelligentes Verhalten entsteht.

Myons Kopf

Für diese Arbeit ist ausschließlich der Kopf des Roboters von Bedeutung. Hier findet die auditive Wahrnehmung sowie die Verarbeitung auditiver Signale statt. Die Produktion von Lauten erfolgt durch einen Lautsprecher im Kopf. Hinsichtlich der Größe und der Proportionen ähnelt Myons Kopf dem eines Menschen (siehe Abb. 2.2). Auf Höhe der Ohren befindet sich rechts und links jeweils ein Mikrofon. Der Lautsprecher ist vorne, wo ein Mund zu erwarten wäre. Der Hals besteht aus drei Gelenken, die Roll-, Nick- und Gierachse repräsentieren und dem Roboter damit menschenähnliche Kopfbewegungen erlauben. Das Auge bildet hier eine Ausnahme, da es aus nur einer mittig platzierten

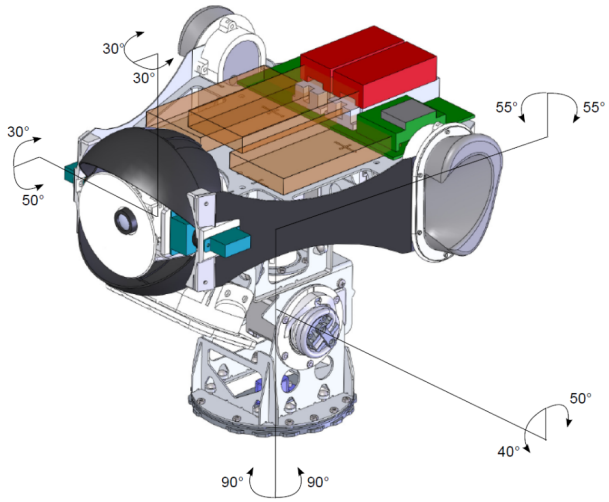


Abbildung 2.2: Myons Kopf, ohne Schalen. Dargestellt sind ie Roll-, Nick- und Gierachse sowie schematisch das Brainmodule, Auge und Ohren.

Kamera besteht. Im Inneren des Kopfes ist das Brainmodule verbaut. In Anlehnung an das menschliche Gehirn ist es eine zentrale Einheit mit erhöhter Rechenleistung. Es übernimmt die Audio- und Videoverarbeitung und ist für zentrale Aufgaben wie Entscheidungsfindung, Aufmerksamkeit und Lernen vorgesehen. Die Motorregelung in den Gelenken wird von lokalen Prozessoren übernommen. Die Vorbereitungen für die Motorregelungen und die Berechnung neuer Sollwerte übernimmt wiederum das Brainmodule. Dieses ist mit zwei Prozessoren und einem FPGA[2] ausgestattet. Es befinden sich außerdem zwei SD-Karten Slots, ein PS/2-Anschluss und Anschlüsse für Audio und Video auf dem Brainmodule. Die Anschlüsse sind gut erreichbar am Hinterkopf des Roboters platziert.

2.2 Für die Kommunikation genutzte Teilsysteme

Um erfolgreich mit dem Roboter kommunizieren zu können, sind dessen perzeptive Systeme entscheidend. Für die vorliegende Arbeit ist dafür hauptsächlich die auditive Wahrnehmung wichtig. Um Kommunikation in beide Richtungen zu erlauben, sind außerdem die Audioausgabe sowie die Bildschirmausgaben von Bedeutung.

Die zwei gegenüberliegenden Mikrofone bilden ein binaurales System, welches in seinen Dimensionen und der räumlichen Orientierung dem menschlichen sehr nahe kommt (Abb. 2.2). Sie sind in künstliche, nach menschlichem Vorbild geformte, 3D-

[2]FPGA: Field Programmable Gate Array

gedruckte Ohrmuscheln eingelassen und haben einen Abstand von 22 cm zueinander. Das serielle Signal aus den Mikrofonen wird in mehreren Stufen auf 16 kHz konvertiert. Die Audioausgabe erfolgt über einen Audio-Codec, der vorhandene Audiodaten am Lautsprecher und am Chinch-Ausgang am Hinterkopf ausgibt. Die Lautstärken der beiden Ausgänge lassen sich individuell anpassen.

2.3 Behavior Design Environment

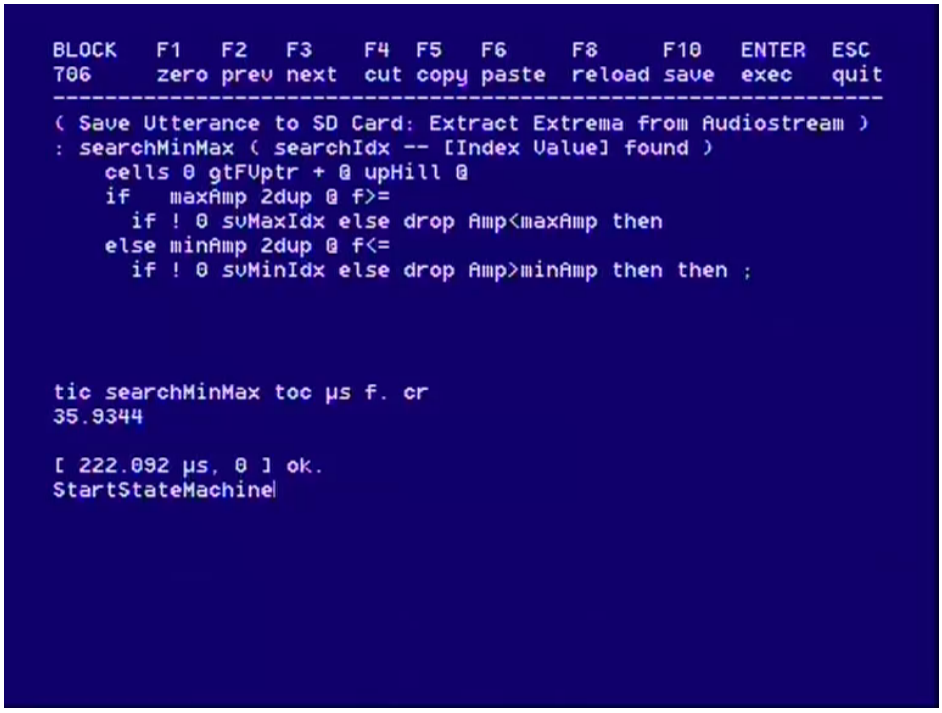


Abbildung 2.3: Ansicht eines typischen Fensters der Benutzeroberfläche der *BDE*.

Um den Roboter zu programmieren, schaltet man ihn ein und verbindet einen Bildschirm und eine Tastatur mit seinem Kopf. Die Entwicklungsumgebung *Behavior Design Environment (BDE)* wird direkt auf dem Brainmodul ausgeführt und bietet alle nötigen Funktionalitäten. Sie basiert auf *Forth*, einer imperativen, stackbasierten Programmiersprache (für nähere Informationen über *Forth* siehe [**?**] und [**?**]), die in der umgekehrten polnischen Notation formuliert wird. Die Benutzeroberfläche der *BDE* ist zum einen in Blöcken organisiert, in die der Programmcode geschrieben wird. Zum anderen können Bildschirmeingaben im interaktiven Modus direkt ausgeführt werden. Abbildung 2.3 zeigt eine typische Bildschirmansicht der *BDE*. Im oberen Bereich sieht

man Programmcode, der in einen Block geschrieben wurde. Im unteren Teil wurden einige Befehle eingegeben. Die *BDE* besteht aus etwa 140 integrierten Wörtern (Befehle und Routinen werden in *Forth* „Wörter“ genannt) und kann unbegrenzt mit eigenen Wörtern erweitert werden.

Hierbei werden Wörter wie folgt definiert:

```
: identifier ( stack[before] -- stack[after] )  Programmcode ;
```

Der Doppelpunkt startet die Wortdefinition. Darauf folgt der Name (identifier) des neuen Wortes, mit dem es später aufgerufen wird. Nach *Forth*-Konvention folgt dann zunächst ein Kommentar mit den Operanden, die das Wort vom Stack nimmt (linke Seite des Doppelstrichs) und den Ergebnissen, die es auf dem Stack hinterlässt (rechte Seite). Nun kommt der Programmcode, der wiederum aus Wörtern besteht. Ein gut durchdachter Programmcode lässt sich auf höheren Ebenen fast wie normale englische Sprache lesen. Ein Beispiel sieht folgendermaßen aus:

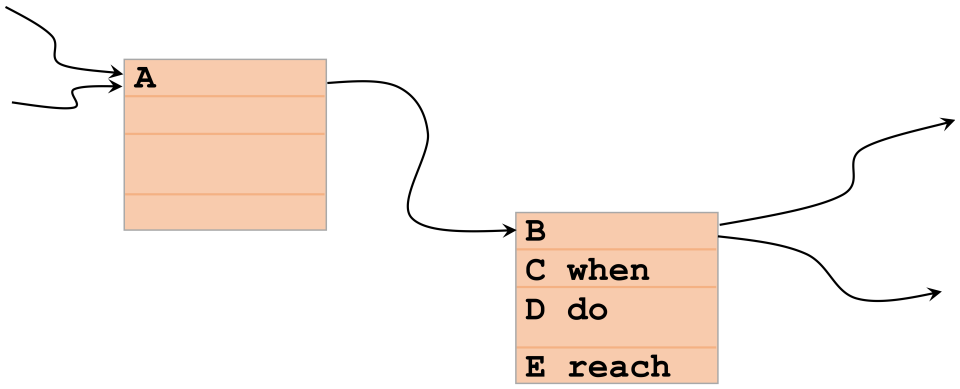
```
: svStart (  --  ) WordAdr @ svWordNr svlast2FV updList 96 svCnt ! ;
```

Ein Kernbaustein der *BDE* bildet die 10ms-Schleife. Jedes Wort in der *BDE* kann in die Schleife geschickt werden und wird dann alle 10ms ausgeführt. Um beispielsweise Audiosamples mit 16kHz zu verarbeiten, wird das Wort `getSample` 160 mal pro Schleifendurchlauf ausgeführt. Dieses Wort ist wie folgt aufgebaut:

```
: getSample ( -- ) AudioBuffer SampleReadPtr @ + @ dup calcSampleR
                   SampleR ! calcSampleL SampleL ! incSamplePtr ;
```

Hier wird die Adresse des Audiobuffers und der Pointer auf das aktuelle Sample auf den Stack gelegt und dann das Sample geladen. Da das Sample aus zwei 16 Bit Zahlen, kombiniert in einer 32 Bit Zahl, besteht, wird es nun in rechts und links aufgespalten und gespeichert. Zum Schluss wird der Pointer eine Speicherstelle weiter geschoben. Eine weitere wichtige Funktionalität der *BDE* sind Zustandsautomaten. Dazu gibt es spezielle Wörter, mit denen Zustände definiert und verbunden werden können (siehe Abbildung 2.4).

Einen Zustand definiert man nicht mit einem Doppelpunkt, sondern mit einem Gatter. Es folgt der Name des Zustands. Anschließend kommt eine Bedingung `C` (optional), die erfüllt sein muss, damit der Zustand aktiv werden kann. `D` steht für die Wörter, die ausgeführt werden sollen, wenn in diesen Zustand gewechselt wird (optional). Den Abschluss bildet eine Bedingung, die erreicht werden muss, bevor der Zustand verlassen werden kann (optional). Über das Schlüsselwort `link` können Zustände verknüpft werden. Ein Beispiel einer umfangreicheren Zustandsmaschine ist in Abschnitt 4.1 zu




```
#A do ;
#B C when D do E reach ;
A B link
```

Abbildung 2.4: Zustandsmaschinen innerhalb der *BDE*. Zustand B folgt auf Zustand A, wenn Bedingung C erfüllt ist. Wenn B aktiv wird, wird D ausgeführt. Erst wenn E erfüllt ist, kann ein Folgezustand von B aktiv werden.

finden.

3 Design einer multimodalen Kommunikation

Ein langfristiges Ziel des Forschungslabors Neurorobotik ist es, mit dem Roboter Myon auf sprachlicher Ebene kommunizieren zu können. Dabei soll Myon in der Lage sein, Werkzeuge der Sprachverarbeitung und anderer perzeptiver Domänen nach eigenem Ermessen zu nutzen, um sich in seiner Umwelt zurecht zu finden. Ein Meilenstein auf dem Weg zu diesem Ziel ist die Möglichkeit, Myon vollständig ohne Tastatur und Bildschirm programmieren zu können. Der Roboter soll dann unter anderem mit Sprache programmiert werden. Hierbei sind nicht nur gesprochene Wörter von Bedeutung, sondern auch Gesten, Bewegungen und Zeichnungen.

Da der Begriff der Programmierung viele Aspekte beinhaltet und von Vertretern der verschiedenen Domänen ganz unterschiedlich verstanden wird, definiert der nächste Abschnitt die sprachliche Programmierung, wie sie für die vorliegende Arbeit gilt.

3.1 Sprachliche Programmierung

Die Programmierung des Roboters Myon zielt darauf ab, ihm neue Fähigkeiten zu verleihen und ihm den Erwerb von neuem Wissen zu ermöglichen. Dies soll in zunehmendem Maße ohne Tastatur und Bildschirm geschehen. Die Programmierung findet dann nicht mehr im klassischen Sinne durch das Tippen von Programmcode statt, sondern durch Zeigen, Erklären und Bewegen des Roboters und seiner Körperteile. Zum Beispiel werden Sachverhalte anhand von Graphen, Tafelabbildungen oder ähnlichen visuellen Quellen dargestellt. Durch visuelle Wahrnehmung und interne Übersetzung in geeignete Datenstrukturen erweitert der Roboter damit sein Weltwissen. Beispielsweise bekommt er verschiedenfarbige Objekte gezeigt, die er dann als Datenstrukturen speichert und bei erneuter Präsentation wiedererkennen kann. In anderen Beispielsituationen werden die Gelenke der Gliedmaßen des Roboters bewegt und ihm daran bestimmte Körperhaltungen erklärt. Diese Vorgänge werden durch Worte und Sätze kommentiert, sodass sprachliche Referenzen möglich sind. Durch den Abgleich interner Strukturen untereinander entstehen Verknüpfungen und Assoziationen, basierend auf Gemeinsamkeiten. Der Roboter kann schließlich durch Abstraktion und logische Schlüsse den Bedeutungsinhalt verschiedener Wörter und Sätze lernen.

Es lassen sich bei dieser Form der Programmierung zwei Ebenen unterscheiden. Auf

der höheren Ebene bezeichnen gesprochene Worte zum Beispiel bestimmte Objekte, die bereits im Wissensschatz des Roboters vorhanden sind. Diese können dann um weitere Eigenschaften erweitert oder in Zusammenhang mit anderen Objekten gebracht werden. Die niedrigere Ebene bildet die direkte Programmierung mit Worten. Dabei dienen bestimmte Worte als Auslöser bestimmter Funktionen in der Software des Roboters. Zum einen wird damit ermöglicht, Programmcode mit sprachlichen Mitteln zu erstellen. Zum anderen können Abläufe im System des Roboters gesteuert werden. Dazu zählen auch Bewertungen innerhalb von Lernsituationen, die über eine Metasprache kommuniziert werden.

Metasprache

Eine Metasprache wird benötigt, um über eine Sprache zu sprechen. Sie besteht aus Symbolen, die eine klare Bedeutung für alle Kommunikationsteilnehmer haben. Diese werden vorher als Vokabular für eine gemeinsame Kommunikationsgrundlage definiert. Da der Roboter anfänglich über kein Sprachverständnis verfügt und auch noch keine Assoziationen bestehen, werden Schlüsselwörter benötigt, die zum Beispiel als Zustimmung oder Ablehnung in Lernsituationen dienen können. Diese Wörter müssen fest in das System eingebunden sein und zuverlässig erkannt werden. Die Metasprache muss nicht in gesprochener Form realisiert werden. Beispielsweise sind auch visuelle Symbole möglich. Jedoch ist die gesprochene Sprache für die Interaktion mit dem Roboter vorteilhaft. Dies ist zunächst in der Einfachheit begründet, mit der Menschen Sprache produzieren können. Dabei wird eine Fülle an Informationen in kurzer Zeit übermittelt. Des Weiteren werden keine Hilfsmittel benötigt. Die mit dem Roboter interagierende Person hat beide Hände frei und kann sie in der Situation anderweitig nutzen. Hilfreich ist es vor allem dann, wenn die Situation physische Interaktion erfordert.

Im Folgenden werden Situationen beschrieben, die für sprachliche Interaktion relevant sind. Im Anschluss daran werden die Anforderungen an das Kommunikationssystem des Roboters anhand der sprachlichen Erstellung von Programmcode beschrieben.

3.2 Darstellung relevanter Kommunikationssituationen

Die Kommunikationssituationen, die in dieser Arbeit behandelt werden, sind als Dialoge aufgebaut. In Anlehnung an den „Semiotic Cycle“ aus [**?**] ist eine solche Situation in Abbildung 3.1 dargestellt. Hier wird die linke Seite als „Sprecher“ und die rechte als „Zuhörer“ beschrieben. Der Sprecher ist der aktive Teilnehmer, der dem Zuhörer Informationen übermittelt oder eine Handlungsaufforderung gibt. Der Zuhörer führt daraufhin eine Aktion aus. Dies kann eine interne Abfolge von Aktionen oder auch eine

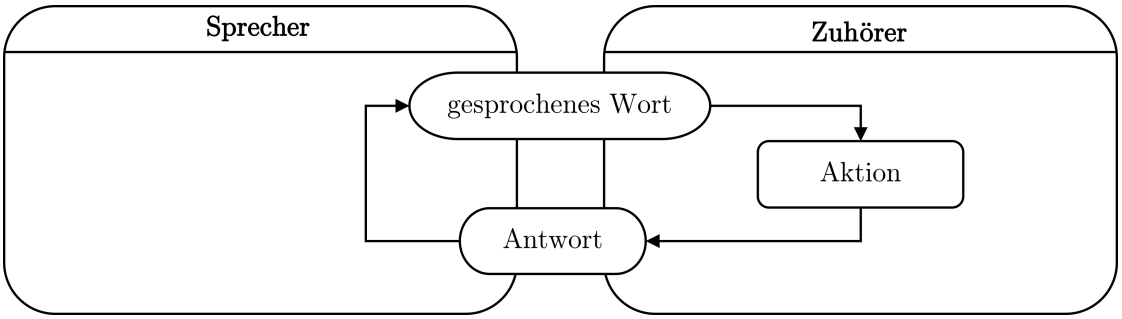


Abbildung 3.1: Darstellung des Dialogs zwischen Sprecher und Zuhörer. Beide Teilnehmer sind als eigene Entitäten (große Kästen) mit inneren Vorgängen (kleine Kästen) dargestellt. Ovale zwischen Sprecher und Zuhörer stellen die Kommunikation zwischen beiden dar.

körperliche Handlung sein. Es findet aber auch durchaus eine Kommunikation in Richtung des Sprechers statt. Beispielsweise kann der Zuhörer Verständnis kommunizieren oder bei Fehlen von ebendiesem nachfragen.

Diese Darstellung eines Dialoges ist die Grundlage für verschiedene Kommunikationssituationen. Ebenfalls in Anlehnung an [**?**] werden die verschiedenen Situationen im Folgenden „Sprachspiele“ genannt. Multimodale Kommunikation wird auf verschiedenen Ebenen durchgeführt. Nicht nur die gesprochene Sprache bietet ein Mittel zum Informationsaustausch, sondern auch geschriebene Sprache, Gesten des Sprechers sowie Zeichnungen und andere visuelle Mittel. Des Weiteren kann der Körper des Zuhörers ebenfalls als Kommunikationsmittel dienen, indem der Sprecher zum Beispiel die Begriffe „links“ und „rechts“ an Bewegungen des linken und rechten Arms des Zuhörers knüpft.
Für die folgenden Sprachspiele ist der Sprecher ein Mensch und der Zuhörer ein Roboter.

Sprachspiel Wörter lernen: Bei diesem Spiel erlernt der Zuhörer neue Worte. Der Sprecher kündigt ein neues Wort über die Metasprache an (zum Beispiel durch die Schlüsselworte: "Myon, neues Wort") und nennt das Wort anschließend. Der Zuhörer speichert das neue Wort. Im Optimalfall spricht er das neue Wort nach, sodass der Sprecher gegebenenfalls korrigieren kann.

Sprachspiel Worterkennung: Der Sprecher sagt Worte, die der Zuhörer bereits im Speicher hat. Der Zuhörer vergleicht das Gehörte mit dem Speicher und gibt an, welches Wort er erkannt hat. Der Sprecher signalisiert Erfolg oder Misserfolg über die Metasprache. Dieses Spiel dient als Maß für die Qualität der Erkennung.

Sprachspiel Wörter sprechen: Der Sprecher spricht mehrere Male einen Satz. Anschließend lässt der Sprecher genau ein Wort im Satz weg. Der Zuhörer kann nun durch die vorangegangenen Beispiele erschließen, dass ein Wort fehlt und

dieses ergänzen. Der Sprecher bewertet die Ausgabe des Wortes und korrigiert gegebenenfalls. Beispielhafter Ablauf:
Sprecher: „Heute ist es kalt draußen.“ (Drei mal.)
Sprecher: „Heute ist es ... draußen.“
Zuhörer: „Alt.“
Sprecher: „Nicht alt. *Kalt*!“
Der Zuhörer kann nun die Differenz des Gesagten und des Gehörten nutzen um die eigene Ausgabe zu verbessern.

Sprachspiel Objektbenennung: Der Sprecher zeigt dem Zuhörer ein Objekt und sagt dazu einen Satz mit dem Namen oder einer Eigenschaft des Objekts. In Verbindung mit visueller Verarbeitung können so Schritt für Schritt neue Objekte und Objekteigenschaften gelernt werden. Hier können aber auch Konzepte wie „links“, „rechts“, „oben“ oder „unten“ mit eingebaut werden.

Sprachspiel Programmierung: In dieser Situation geht es nicht um das Lernen neuer Wörter, sonder um die Programmierung des Zuhörers (des Roboters) mit Hilfe von gesprochener Sprache. Dabei geht es nicht um die Erweiterung des Weltwissens des Roboters, sondern um die Erstellung von Programmcode.

Das Sprachspiel Programmierung benötigt alle grundlegenden Mechanismen, die auch im Kontext der anderen Spiele benötigt werden. Außerdem ist es von besonderem praktischen Nutzen und dient deshalb im Folgenden als beispielhafte Kommunikationssituation.

Sprachspiel Programmierung

Die Situation kann folgendermaßen zusammengefasst werden: Der Sprecher programmiert den Zuhörer (Roboter), indem er eine Wortdefinition spricht. Diese Definition wird vom Roboter gespeichert und auf dem Bildschirm ausgegeben. Wortdefinitionen beginnen, wie in Abschnitt 2.3 beschrieben, mit einem Doppelpunkt und enden mit einem Semikolon. Diese beiden Schlüsselwörter dienen dem Roboter als Start- und Endpunkt der sprachlichen Programmierung und werden außerdem im Programmcode übernommen. Dazwischen werden Wörter der BDE in einer sinnvollen Reihenfolge deutlich gesprochen. Der Roboter fügt diese Wörtern nun in den Programmcode ein, beendet die Wortdefinition mit dem Semikolon und speichert. Bei Unsicherheit in der Erkennung soll der Roboter auf diese aufmerksam machen und gegebenenfalls nachfragen. Diesen gesamten Dialog kann man in einzelne Stationen unterteilen. Für jedes neu gesprochene Wort wird eine eigene Kommunikationssituation definiert. Abbildung 3.2 zeigt den Ablauf für die Erkennung eines einzelnen Wortes im Sprachspiel Programmierung. Hier ist genau diese Dialogsituation zwischen dem Programmierenden und dem Roboter dargestellt. Der Programmierer spricht ein Wort, das der Roboter dann

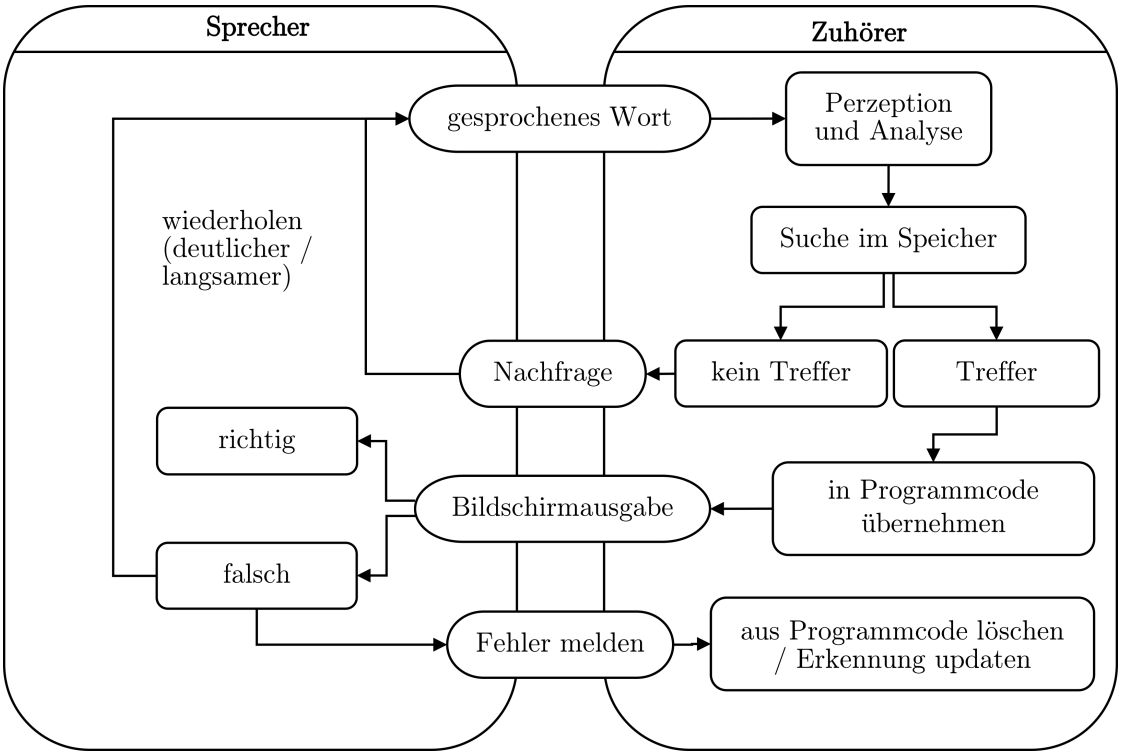


Abbildung 3.2: Darstellung des Dialogs zwischen Programmierer und Roboter. Ablauf zur Erkennung eines Wortes und Einfügen von diesem in aktuelle Wortdefinition.

intern verarbeitet. Es wird entweder erkannt oder nicht erkannt. Falls das Wort nicht erkannt wurde, kann der Roboter nachfragen. Dies geschieht entweder durch eine Verbalisierung, zum Beispiel ein Fragewort der vereinbarten Metasprache oder durch eine Ausgabe am Bildschirm. Bei Übereinstimmung mit einem Wort im Speicher wird dieses im Programmcode übernommen. Das Ergebnis wird auf dem Bildschirm ausgegeben, sodass der Programmierer es kontrollieren kann. Falls der Roboter das richtige Wort erkannt hat, ist die Dialogsituation abgeschlossen und der Kreislauf für das nächste Wort kann beginnen. Andernfalls wird die fehlerhafte Erkennung dem Roboter gemeldet, damit dieser gegebenenfalls Maßnahmen ergreifen kann. Des Weiteren wird das letzte, in den Programmcode eingefügte Wort gelöscht. Nun kann der Programmierer das Wort erneut sprechen.

Vereinfachung des Systems

Für die vorgestellte Dialogsituation benötigt der Roboter verschiedene Mechanismen. Den Kernbaustein bildet die Fähigkeit, Wörter zu erkennen und in den Programmcode zu schreiben. Um diesen Teilprozess gezielt untersuchen zu können, wird die Situation entsprechend der Abbildung 3.3 weiter vereinfacht. Falls das Wort nicht erkannt

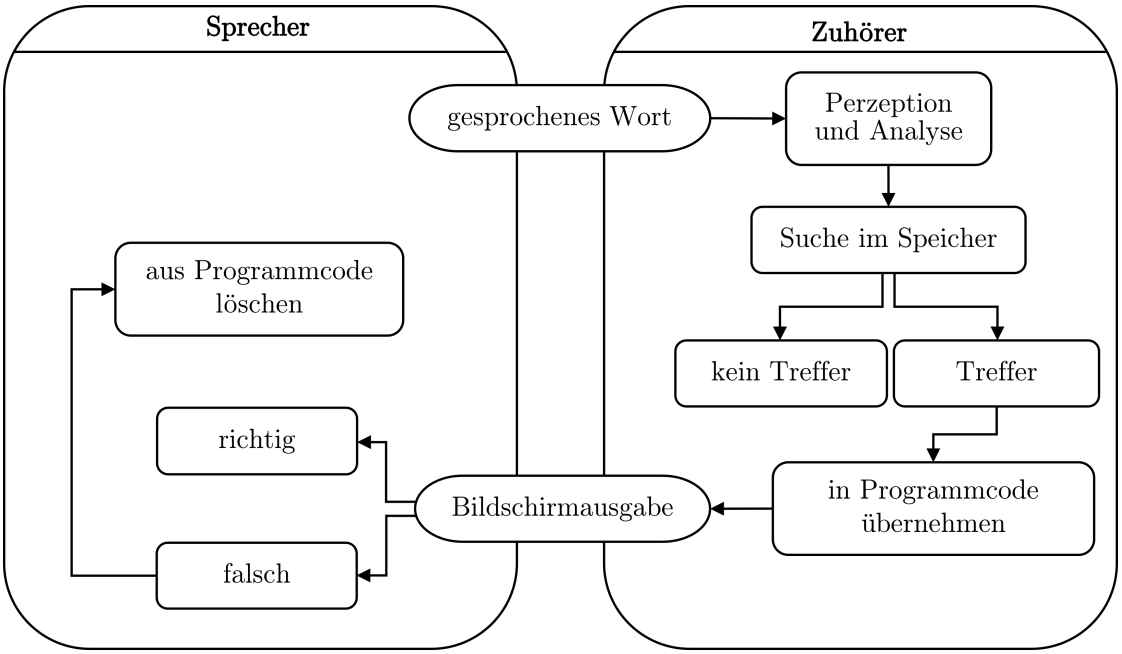


Abbildung 3.3: Reduzierte Dialogsituation mit Kernbausteinen: Perzeption, Worterkennung und Erweiterung von Programmcode.

wurde, wird dem Programmcode nichts hinzugefügt. Falls es doch erkannt wurde, erscheint das neue Wort auf dem Bildschirm. In beiden Fällen ist die Dialogsituation damit abgeschlossen. Der Programmierer sieht das Ergebnis sofort und kann ein nicht erkanntes Wort in einer neuen Dialogsituation wiederholen. Andernfalls kann er mit dem nächsten Wort fortfahren oder ein falsches Wort mit der Tastatur wieder löschen.

Diese vereinfachte Situation wird nun als Grundlage für die folgenden Abschnitte verwendet.

3.3 Anforderungen an ein Kommunikationssystem

In Abbildung 3.3 sind die einzelnen Stationen des Dialogs zwischen Programmierer und Roboter dargestellt. Daraus lassen sich folgende vom Roboter benötigte Prozesse ableiten:

1. **Perzeption und Analyse** Der Roboter muss über Mechanismen verfügen, um Schallwellen in eine für ihn verwertbare Form zu bringen. Dazu werden die auditiven Signale auf verschiedene Weisen analysiert und gespeichert. Die Details hierzu finden sich in Kapitel 4.

2. **Suche im Speicher** Die analysierten auditiven Signale werden mit bereits gespeicherten Wörtern verglichen. Auf diese Weise wird eine beste Übereinstimmung

gefunden.

3. **Bewertung des Vergleichs** Der Vergleich ergibt ein Maß für die Übereinstimmung des gehörten Wortes mit Worten im Speicher. Anhand dieses Maßes wird entschieden, ob ein Wort erkannt wurde, oder ob nachgefragt werden muss.

4. **Übernahme des Wortes** Das erkannte Wort wird in den Programmcode übernommen und auf dem Bildschirm ausgegeben. (In anderen Sprachspielen und für ein vollständiges Kommunikationssystem notwendig, ist hier die Sprachsynthese zu erwähnen).

Perzeption und Analyse

Für die Wiedererkennung von Wörtern ist die Reproduzierbarkeit der Ergebnisse von zentraler Bedeutung. Das selbe gesprochene Wort sollte immer wieder als dasselbe erkannt werden. Dazu muss die Perzeption und die Analyse robust gegen Variationen sein. Variationen in der Lautstärke (Amplitude), Abschwächung der hohen Frequenzen durch Entfernung, Hintergrundgeräusche und Änderungen der Tonhöhe dürfen nicht dazu führen, dass sich die Bedeutung des Wortes verändert. Zusätzlich muss die Erkennung sprecherunabhängig sein. Damit ergibt sich:

> **Anforderung 1:** Die Prozesse der Perzeption und Analyse müssen ein auditives Signal auf eine Repräsentation der Lautqualität abbilden, die robust gegen Störungen ist und verlässlich wiedererkannt wird.

Trotz der benötigten Invarianz gegenüber Tonhöhe und Amplitude für den Vergleich der Merkmalsvektoren, sind diese Eigenschaften wichtig.

Suche im Speicher

Silben bestehen aus dem sogenannten Nukleus (Silbenkern), der oft, aber nicht immer, von der Silbenschale umschlossen ist. Der linke Rand wird Silbenansatz und der rechte Rand Silbenkoda genannt. Die beiden Ränder bestehen meist aus Lauten niedriger Sonorität (Klangfülle) und der Silbenkern aus Lauten hoher Sonorität. Zur ersten Klasse gehören Plosive und Konsonanten, zur zweiten stimmhafte Laute (nachzulesen unter anderem in [**?**]). Dies ist in Abbildung 3.4 schematisch dargestellt. Die Schallenergie verhält sich proportional zur Klangfülle. Damit ist die meiste Energie im Kern der Silbe konzentriert. Diese Tatsache kann ausgenutzt werden, um die Worterkennung über Energiehierarchien robust und effizient zu gestalten. Dazu kann die Amplitude genutzt werden.

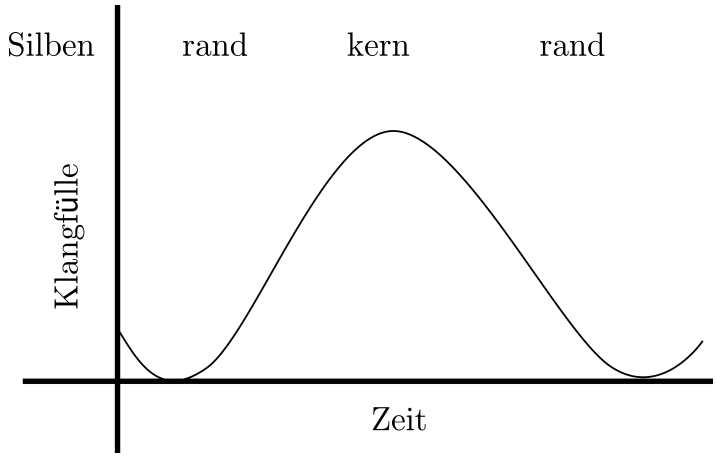


Abbildung 3.4: Schema des akustischen Verlaufs einer Silbe. Linker und rechter Rand sind optional. Der Kern ist am energiereichsten.

Die Amplitudenspitzen in den Silben eines Wortes bieten einen guten Einstiegspunkt, um eine Suche im Wortspeicher durchzuführen. Dadurch sind im ersten Schritt nur wenige Vergleiche pro Wort nötig. Alle Wörter, deren Vokalqualitäten an den entsprechenden Stellen nicht übereinstimmen, werden aus der Suche ausgeschlossen. Anschließend wird die Auflösung der Suche erhöht. In Abschnitt 4.3 wird dieses Vorgehen genauer erläutert. Dabei sollen einige Parameter der Suche variabel bleiben und potentiell vom Roboter veränderbar sein. Auch hierauf wird in Abschnitt 4.3 eingegangen. Daraus folgt die zweite Anforderung:

> **Anforderung 2:** Benötigt wird ein Prozess zur Silbenseparation durch die Suche nach Energiemaxima. Die Ergebnisse müssen in geeigneten Datenstrukturen abgelegt werden. Zusätzlich muss ein weiterer Prozess diese Maxima mit dem Speicher vergleichen.

Bewertung des Vergleichs

Die Suche im Speicher und die Bewertung der Vergleiche sind trotz der Darstellung als unterschiedliche Prozesse nicht voneinander trennbar. Die Suche besteht aus Vergleichen mit den Wörtern im Speicher auf unterschiedlichen Hierarchieebenen. Zunächst wird die Zahl der Maxima (und damit die ungefähre Silbenanzahl) und deren Metrum verglichen. In Folge der Bewertung dieser Vergleiche wird die Liste der potentiellen Treffer angepasst und (hoffentlich) verkleinert. Im Anschluss daran wird, von den Maxima ausgehend, nach vorne und hinten (zeitlich auf der Ebene der Merkmalsvektoren, siehe Abschnitt 4.3) gesucht. Damit werden nun auch die Ränder der Silben betrachtet. Die Anforderung an diese Stufe der Prozesse lautet:

> **Anforderung 3:** Um die Merkmalsvektoren vergleichen zu können, wird ein Ähnlichkeitsmaß benötigt.

Übernahme des Wortes

Die Übernahme des Wortes in den Programmcode ist in der betrachteten Kommunikationssituation das Beispiel für die Reaktion des Roboters. In diesem speziellen Fall werden dazu nur geeignete Funktionalitäten benötigt, um das gefundene Wort entsprechend umzusetzen. Für andere Situationen sind an dieser Stelle sehr unterschiedliche Reaktionen möglich. Dies kann im Sprachspiel *Objekterkennung* zum Beispiel das Aktualisieren des Gedächtnisses sein. Im Sprachspiel *Wörter Sprechen* ist es die Sprachsynthese eines Wortes oder Satzes. Hieraus ergeben sich ganz unterschiedliche Anforderungen. Je nach Sprachspiel oder je nach anderer denkbarer Situation müssen neue Anforderungen definiert werden. Als zunächst wichtigste erscheint:

> **Anforderung 4:** Ein funktionales Feedbacksystem. Wünschenswert wäre eine Sprachsynthese. Als Metasprache würde eine Ausgabe von Sinustönen verschiedener Frequenzen zunächst genügen.

In der vorliegenden Arbeit wird die letzte Anforderung noch nicht umgesetzt.

4 Implementierung

Die im vorhergehenden Kapitel erarbeiteten Anforderungen an das System wurden im Roboter Myon umgesetzt. In diesem Kapitel wird zunächst das System, wie es derzeit arbeitet, vorgestellt. Im Anschluss wird die Verarbeitung der auditiven Signale im Detail beschrieben. Den Anfang bildet die Perzeption. Es wird kurz auf die einzelnen Stationen dieses Prozesses eingegangen. Danach wird die Analyse der Daten und die Erstellung der Merkmalsvektoren beschrieben. Den Abschluss bildet die Beschreibung der Suche nach Wörtern im Speicher durch die Analyse der Energieverteilung.

4.1 Das Kommunikationssystem

Um das Sprachspiel Programmierung durchführen zu können, wird ein Zustandsautomat benötigt. Im Folgenden wird der Zustandsautomat zunächst grob beschrieben. Anschließend wird genauer auf die Funktion der einzelnen Zustände eingegangen. Dabei werden die Speicherstruktur und die einzelnen Verarbeitungsschritte beschrieben.

Wie in Abbildung 4.1 zu sehen, besteht der Zustandsautomat aus 5 Zuständen. Der Zustand `Init` ist ein Startzustand, der nur als Einstiegspunkt in den Zustandsautomaten dient. Von `Init` wird sofort zu `WaitforWord` gesprungen. Dieser Zustand bleibt solange aktiv, bis die Amplitude einen festgelegten Schwellenwert überschreitet. Der Amplitudenanstieg wird als Wortanfang gewertet und der Zustand `getAmplitude` wird aktiv. In diesem Zustand wird das aktuelle Audiosignal mit im Speicher liegenden Wörtern verglichen. Wenn eine Übereinstimmung gefunden wurde, wird in den Zustand `WordKnown` gewechselt. Dort wird das Wort in den Programmcode eingefügt und das Ergebnis auf dem Bildschirm ausgegeben. Wenn kein Wort erkannt wurde, wird der Zustand `WordUnknown` aktiv und der Roboter bittet um eine Wiederholung des Wortes. In allen Zuständen ist das *BDE*-Wort `Rec` in der 10ms-Schleife. `Rec` wurde folgendermaßen definiert:

```
: Rec ( -- ) rdSamplePtr rstResults
             16 0 for wr2PitchBuffer
                  10 0 for getSample GoertzelLoop
                           calcAmplitude calcNoise
             next next PitchDetect wrFeatureVector ;
```

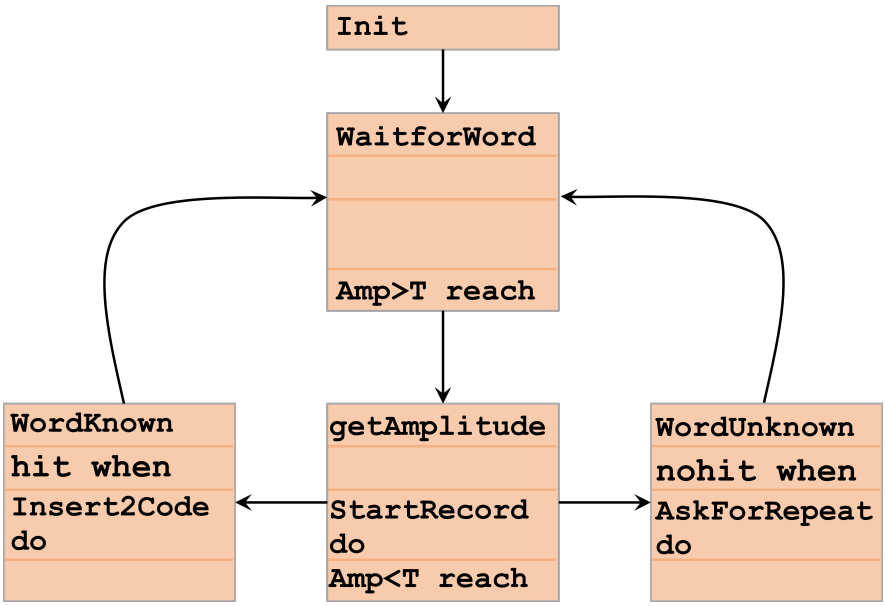


Abbildung 4.1: Der Zustandsautomat für das Sprachspiel Programmierung. Sobald die Amplitude einen Grenzwert überschreitet, wird aus dem Zustand „WaitforWord“ in den Zustand „getAmplitude“ gewechselt. Hier wird mit Wörtern im Speicher verglichen. Je nach Ergebnis werden dann die Zustände „WordKnown“ oder „WordUnknown“ ausgeführt.

Mit Hilfe des Wortes `Rec` werden 160 Samples aus dem Audiobuffer geholt und verarbeitet. Dabei wird jedes zehnte Sample in den Buffer für die Tonhöhenanalyse (siehe Abschnitt 4.2: Tonhöhe) geschrieben. Es werden außerdem alle Berechnungen für die Goertzelfilter (siehe Abschnitt 4.2: Frequenzspektrum) und die Amplitudenberechnung (siehe Abschnitt 4.2: Amplitude) durchgeführt. Den Abschluss bildet das Wort `wrFeatureVector`, das alle berechneten Werte in einen Merkmalsvektor schreibt und diesen im `FeatureVecorBuffer` speichert. Das Wort `Rec` ist der zentrale Prozess der Sprachverarbeitung. Es beinhaltet alle perzeptiven und analytischen Teilprozesse.

Eine Übersicht über die verschiedenen Buffer bietet Tabelle 4.1. Der Audiobuffer

Tabelle 4.1: Die unterschiedlichen Buffer

Buffer	Größe	Inhalt	Lese-,Schreibfrequenz
Audiobuffer	1024	Sample (int32)	16kHz
Speakerbuffer	1024	Sample (int32)	16kHz
Pitchbuffer	2048	Sample (float32)	1600Hz
Featurevectorbuffer	6000	Merkmalsvektor (32 Bytes)	100Hz

ist als Ringbuffer realisiert und wird vom zweiten Prozessor des Brainmodules mit Audiosamples gefüllt. Hieraus holt `getSample` (siehe Abschnitt 2.3) das Sample und

verarbeitet es weiter. Der Speakerbuffer ist das Pendant des Audiobuffers für die Audioausgabe. Hier werden Samples hineingeschrieben, die dann an den Audiocodec und dann an den Lautsprecher geschickt werden. Der Pitchbuffer enthält die Samples, die von `PitchDetect` genutzt werden, um die Tonhöhe zu berechnen (siehe Abschnitt 4.2). Der Featurevectorbuffer enthält Merkmalsvektoren und ist der größte Buffer. Er muss eine große Kapazität besitzen, da der Roboter das gerade Gehörte auf unterschiedliche Weisen verarbeiten können soll. Dazu muss er auf die letzten Sekunden des Audiosignals zugreifen. Daher bietet dieser Buffer Platz für die Merkmalsvektoren der letzten 60 Sekunden.

Auf ein Wort warten

Der Zustandsautomat wird gestartet, indem der Zustand `Init` aktiviert wird. Da dieser Zustand leer ist und keine Bedingungen vorhanden sind, wird sofort in den Zustand `WaitforWord` gewechselt. Dieser Zustand führt keinen Programmcode aus, sondern prüft alle 10ms, ob die Bedingung für das Verlassen des Zustandes schon erreicht ist. Diese Bedingung ist in `Amp>T` formuliert:

```
: Amp>T ( -- ) Amplitude @ 0.003 f> ;
```

Es wird also nur geprüft, ob die Amplitude den Schwellenwert von 0.003 überschreitet.

Amplituden aufnehmen und im Speicher suchen

Sobald die Amplitude den Schwellenwert überschreitet, wird in den Zustand `getAmplitude` gewechselt. Nun wird `StartRecord` einmal ausgeführt und dann die Bedingung `Amp<T` in der Schleife geprüft. `StartRecord` schickt zwei Wörter in die Schleife. Das erste, `fillAmplitudeList`, registriert Amplitudenmaxima im Audiosignal und fügt die entsprechenden Merkmalsvektoren in eine Liste ein. Das Word `searchforWord` nutzt diese Liste und die Listen der im Speicher liegenden Wörter um Übereinstimmungen zu finden (für nähere Erläuterung dieser beiden Wörter, siehe Abschnitt 4.3). Der Zustand ist so lange aktiv, bis die Amplitude wieder unter einen Schwellenwert fällt.

(K)ein bekanntes Wort gefunden

Nach dem Zustand `getAmplitude` gibt es zwei mögliche Folgezustände. Entweder war die Suche im Speicher erfolgreich. Dann gibt das Wort `hit` eine 1 zurück und der Zustand `WordKnown` wird aktiv. Oder das Gegenteil ist der Fall und der Zustand `WordUnknown` wird aktiv. Ist der Zustand `WordKnown` aktiviert worden, wird das erkannte Wort in den Programmcode eingefügt. Außerdem wird die Bildschirmanzeige

um das erkannte Wort aktualisiert, damit das Wort vom Programmierer überprüft werden kann. War die Suche im Speicher erfolglos, wird der Zustand `WordUnknown` aktiv. In diesem Fall soll der Roboter nachfragen oder auf eine andere Weise signalisieren, dass keine Übereinstimmung gefunden wurde. Das Wort `AskForRepeat` dient aktuell als Platzhalter für solch eine Reaktion.

4.2 Einlesen und Analysieren der Audiodaten

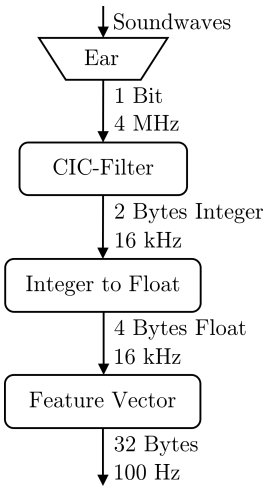


Abbildung 4.2: Flussdiagramm des Datenstroms (Pfeile) vom Mikrofon im „Ohr“ des Roboters bis zur Erstellung der Merkmalsvektoren. Einzelne Verarbeitungsstationen sind als Kästchen dargestellt. Die Bit-Tiefe und die zeitliche Auflösung sind neben den Datenströmen vermerkt.

Die einzelnen Verarbeitungsschritte der auditiven Signale sind in Abbildung 4.2 zu sehen. Wie bereits in Abschnitt 2.2 beschrieben wurde, werden Schallwellen im „Ohr“ des Roboters empfangen und anschließend durch einen Cascaded-Integrator-Comb-Filter (siehe [**?**] für nähere Informationen über CIC-Filter) in ein digitales Signal mit einer Frequenz von 16 kHz und einer Bit-Tiefe von 16 Bit pro Seite umgewandelt. Die nun vorhandenen Werte sind Integer-Zahlen im Zweierkomplement. Diese werden innerhalb der BDE mit wenigen einfachen Befehlen in 32 Bit Float-Zahlen umgewandelt, die nun für alle folgenden Operationen in den Variablen `SampleR` und `SampleL` verfügbar sind. Anschließend wird der Datenstrom verschiedenen Analysen unterzogen und die extrahierten Merkmale in einem Merkmalsvektor gespeichert. Diese Vektoren fassen jeweils 160 Samples zusammen und repräsentieren damit 10ms analysiertes Signal. Der Merkmalsvektor enthält folgende Informationen:

- Frequenzspektrum, bestehend aus 12 Frequenzbändern

- Amplitude als gleitender Durchschnitt
- Informationen über vorhandene Grundfrequenz und deren Wert
- Normierungsfaktoren

Für den Wortvergleich werden unter anderem die Amplitude und das Frequenzspektrum herangezogen (siehe Abschnitt 4.3). Die anderen Informationen werden für das Wortwiederholungsspiel und die damit verbundenen Sprachausgaben genutzt. Außerdem sind diese Informationen für zukünftige Prosodie-Analysen vorgesehen. Im Folgenden wird kurz auf die Erstellung der Merkmale und deren Bedeutung eingegangen.

Frequenzspektrum

Das Frequenzspektrum (siehe Abbildung 4.3) wird durch eine Filterbank, bestehend aus 12 Goertzelfiltern erstellt. Diese Filter werden über den Goertzelalgorithmus berechnet, der 1958 von Gerhard Goertzel entwickelt wurde (Originalquelle: [**?**]). Er dient dazu, einzelne Komponenten einer Diskreten Fourier Transformation eines Signals zu analysieren. Dabei ist die zentrale Struktur des Algorithmus die eines rekursiven, digitalen Filters mit zwei Verzögerungsgliedern. Zur Berechnung der Filterausgabe wird die Formel

$$q_n = s_n + 2\,cos(\omega_0)\,q_{n-1} - q_{n-2}$$

mit jedem neuen Sample s_n aktualisiert. Nach N Berechnungsschritten, wird das Ergebnis ausgewertet und der Filter zurückgesetzt. Dazu wird die Energie, die in dem jeweiligen Frequenzband vorhanden ist über

$$Power = q_{n-1}^2 + q_{n-2}^2 - 2\,cos(\omega_0)\,q_{n-1}\,q_{n-2}$$

berechnet. Dabei gilt für ω_0:

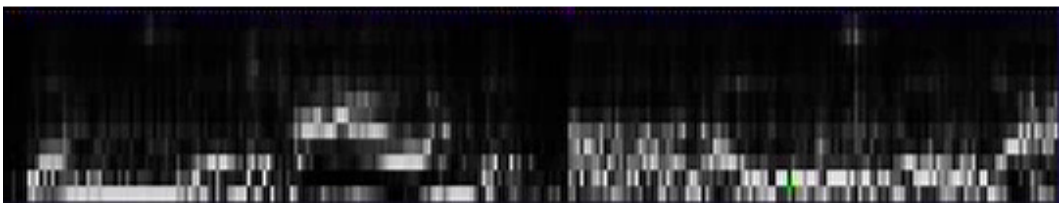


Abbildung 4.3: Auschnitt aus der *BDE*. Gezeigt ist ein Spektogramm, erstellt aus den 12 Frequenzbändern über die Zeit aufgetragen.

$$\omega_0 = 2\pi\frac{k}{N} \quad \text{mit } k \in \{0, 1, 2, \ldots, N-1\}\,.$$

N bestimmt die Bandbreite des Filters. Je größer N gewählt wird, desto kleiner die Bandbreite. Für jede Spektralkomponente wird eine eigene Filterstruktur benötigt. Wenn nur wenige Frequenzen ausgewertet werden sollen, ist die Nutzung von Goertzelfiltern sehr effizient. Für die Auswertung aller Komponenten eines Spektrums ist die Fast Fourier Transformation (FFT) effizienter.

Dennoch werden Goertzelfilter bei der Analyse der Spektralkomponenten verwendet, da sie mehrere Vorteile bieten. Im Gegensatz zur FFT können die Goertzelfilter sehr flexibel eingesetzt werden. Durch geeignete Wahl der Koeffizienten lässt sich der Frequenzbereich annähernd logarithmisch abdecken. In Abbildung 4.4 sind die 12 verwendeten Goertzelfilter dargestellt.

Der Bereich zwischen 100Hz und 1000Hz wird mit schmalbandigen Filtern abgedeckt, während im oberen Bereich zunehmend breitbandige Filter eingesetzt werden. Da eine höhere Bandbreite durch häufigere Auswertung erreicht wird, haben die Filter für die hohen Frequenzen eine bessere Zeitauflösung. Das ist von Vorteil für die Spracherkennung, da Frikative und Plosive oft sehr kurz sind und einen großen Anteil von hochfrequenten Schwingungen besitzen. Diese hochfrequenten Anteile sind oft über einen größeren Frequenzbereich verteilt und müssen deshalb nicht genau lokalisiert werden. Der Bereich niedriger Frequenzen beinhaltet die Formanten der stimmhaften Laute und benötigt deshalb eine bessere Frequenzauflösung. Somit kann mit dieser Anordnung der gesamte, für die Sprache wichtige Frequenzbereich analysiert werden. Ein weiterer Vorteil der Goertzelfilter ist die einfache Struktur. Mit wenigen Änderungen im Programmcode können weitere Filter ergänzt oder Bandbreiten und Zielfrequenzen geändert werden. Diese Einstellungen kann der Roboter später potentiell selbst übernehmen.

Die Spektralkomponenten werden mit jeweils 8 Bit in einen 12-dimensionalen Vektor kodiert. Dieser wird über die euklidische Norm auf Einheitslänge normiert. Die Norm wird auch in den Merkmalsvektor geschrieben, sodass die Originalresultate der Goertzelfilter wiederhergestellt werden können. Damit wurde eine Transformation des Signals aus dem Zeitbereich in den Frequenzbereich durchgeführt. Das Signal wurde in seine spektralen Komponenten zerlegt und in einer neuen Repräsentation gespeichert. Die zeitliche Struktur ist von 16kHz durch die diskrete Schrittweite von 10ms pro Merkmalsvektor auf 100Hz reduziert worden. Damit ist eine gute und vergleichbare Repräsentation der Lautqualitäten entstanden.

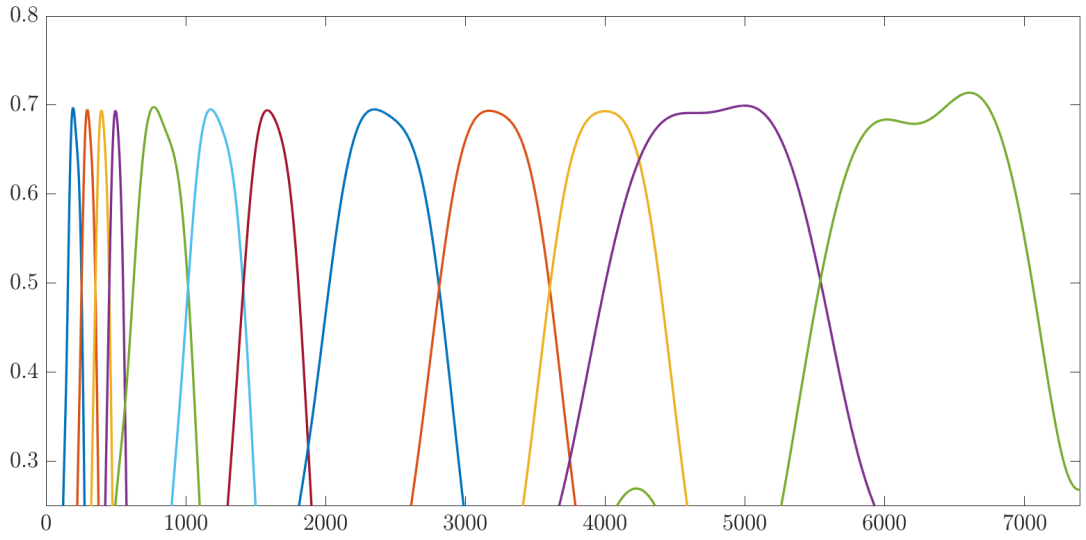


Abbildung 4.4: Filterbank aus 12 Goertzelfiltern. Die Bandbreite im unteren Frequenzbereich ist schmaler um genauere Analysen zuzulassen. Im oberen Frequenzbereich genügen breitbandige Filter, die dafür eine bessere zeitliche Auflösung haben.

Tonhöhe

Eine weitere wichtige Eigenschaft der Sprache ist die Tonhöhe (Grundfrequenz) des Sprechers. Dies ist die Frequenz, mit der die Stimmlippen im Kehlkopf vibrieren, wenn ein stimmhafter Laut (zum Beispiel ein Vokal) produziert wird. Die Tonhöhe wird innerhalb eines Satzes moduliert, um beispielsweise Fragen und Aussagen zu betonen. Im Deutschen steigt sie am Ende einer Frage und sinkt am Ende einer Aussage (siehe Abbildung 4.5). Über längere Zeit gemittelt, kann die Tonhöhe Aufschluss über den Sprecher geben. Des Weiteren ist die Tonhöhe auch für die Synthese von Sprache wichtig, da stimmhafte und stimmlose Laute auf unterschiedliche Weise produziert werden. Um die Tonhöhe eines Sprachsignals zu bestimmen, gibt es eine Fülle von Ansätzen, die in [**?**] umfassend dokumentiert wurden. Häufig genutzte Methoden basieren auf der FFT oder auf Autokorrelation. Einige Beispiele und Evaluationen sind in [**?**] beschrieben. Für den Roboter Myon wurde der in [**?**] vorgestellte Algorithmus implementiert. Auch dieser Algorithmus kann, ähnlich wie die Goertzelfilter, im Zeitbereich sampleweise berechnet und flexibel eingesetzt werden. Er basiert auf der Detektion von sinusförmigen Anteilen in einem Signal, auch bekannt als Pronys Methode (bereits 1795 von Gaspard Riche de Prony entwickelt). Demnach gilt für Sinusoide, dass jedes Sample s_n proportional zum Durchschnitt seiner Nachbarn ist. Damit wird über die Methode der kleinsten Quadrate abgeschätzt, ob ein Signal x_n annäherungsweise sinusförmig ist. Dazu wird zunächst die Fehlerfunktion

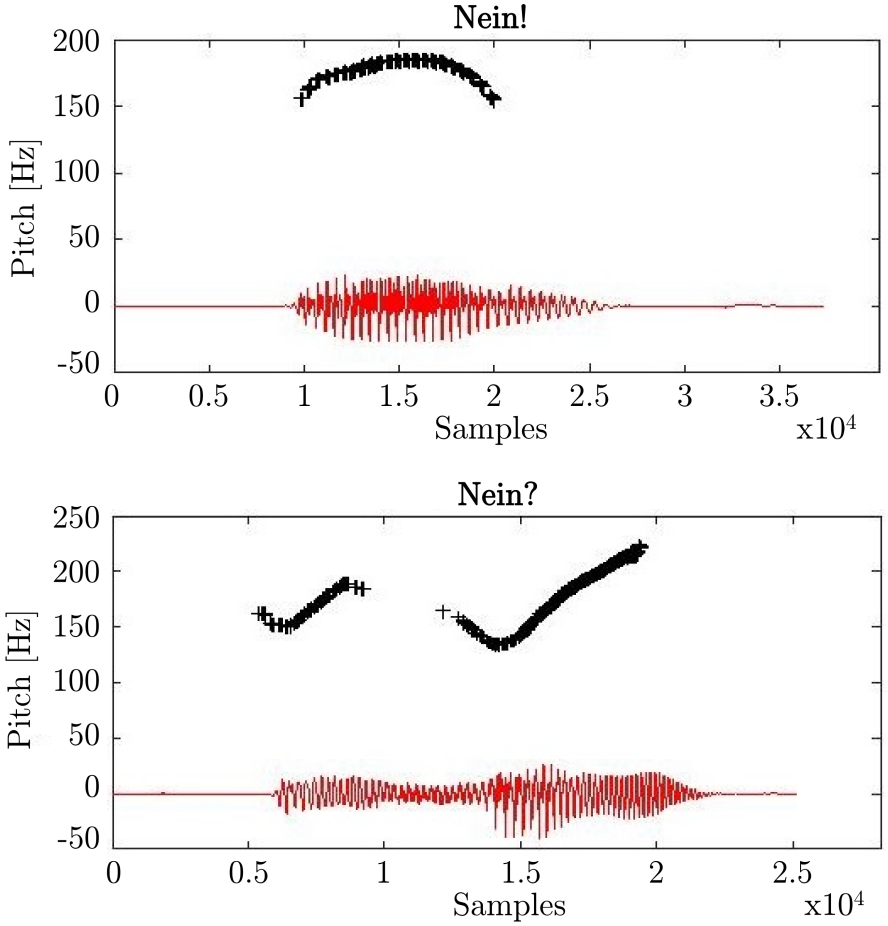


Abbildung 4.5: Am Ende einer Aussage nimmt die Tonhöhe ab (oberer Ausschnitt), bei einer Frage hingegen steigt die Tonhöhe an (unterer Ausschnitt).

$$\varepsilon(\alpha) = \sum_n \left[x_n - \alpha \left(\frac{s_{n-1} + s_{n+1}}{2} \right) \right]^2$$

gebildet. Nun wird der Wert für α gesucht, der den Fehler minimiert. Dazu werden n vergangene Audiosamples in die Analyse einbezogen. Die zu berechnende Funktion lautet

$$\alpha^* = \frac{2 \sum_n x_n \left(x_{n-1} + x_{n+1} \right)}{\sum_n \left(x_{n-1} + x_{n+1} \right)^2}$$

und ist innerhalb der *BDE* im Wort `calcSums` realisiert. Die Fehlerabschätzung und die Berechnung der Werte für die Tonhöhe übernehmen die Worte `calcCost` und

`calcF0s`. Die idealisierte Filterbank aus [**?**] ist mit 3 Bandpassfiltern angenähert worden. Damit wird der Frequenzbereich zwischen 90Hz und 400Hz abgedeckt. Die gesamte Tonhöhenerkennung ist folgendermaßen definiert:

```
: PitchDetect ( -- ) 16 0 for wrSums calcSums calcF0s calcCost next
                     calcFinalF0 ;
```

Die Samplingfrequenz kann für den vorliegenden Algorithmus stark reduziert werden und wurde hier auf 1600Hz festgelegt. Die Anzahl der Samples, die in die Summen mit einfließen, kann variabel gestaltet werden. Durch weniger Samples (kürzerer Zeitabschnitt) kann eine schnellere Schätzung auf Kosten der Genauigkeit erstellt werden. Das Betrachten eines längeren Zeitabschnitts liefert eine genauere Schätzung. Am nützlichsten ist jedoch eine adaptive Gestaltung des Algorithmus. Die Zahl der Samples bleibt variabel, sodass eine größere Zahl Samples analysiert wird, wenn ein leises Signal vorhanden ist. Wenn ein lautes Signal auftaucht, wird die Zahl der Samples reduziert, sodass die Zeitauflösung erhöht wird. Dieser Ansatz ist über einfache Maximalwerte realisiert.

Amplitude

Die Amplitude wird durch einen gleitenden Mittelwert berechnet, der durch eine bedingte Verknüpfung zweier Verzögerungsglieder realisiert ist. Dabei handelt es sich um einfache lineare Tiefpassfilter. Der absolute Betrag des aktuellen Audiosamples wird als Input genommen. Dabei wird unterschieden, ob die Lautstärke ansteigt oder abfällt. Ist der Wert des aktuellen Samples größer als die Amplitude im letzten Zeitschritt, dann ist ein Onset (ansteigende Amplitude, Einschwingvorgang) vorhanden, andernfalls handelt es sich um einen Decay (abfallende Amplitude, Ausschwingvorgang). Um einen möglichst guten gleitenden Mittelwert zu erhalten, der aber trotzdem schnell auf einen Lautstärkeanstieg reagieren kann, werden diese beiden Fälle gesondert berechnet. In Abbildung 4.6 sieht man auf der linken Seite die Filterstruktur des Filters, das die Amplitudenberechnung übernimmt. Je nach Einstellung des Koeffizienten b wird das Signal mehr oder weniger stark gemittelt. Für den Onset gilt $b = 0.1$, für den Decay gilt: $b = 0.0008$. Damit erreicht das Onset-Filter bei gleich bleibender Lautstärke nach etwa 25 Samples 90% des tatsächlichen Wertes (entspricht etwa 1.5 Millisekunden), wohingegen das Decay-Filter etwa 2880 Samples benötigt (180 Millisekunden), um auf 10% des vorherigen Maximums zu fallen. Das Ergebnis dieser Art der Amplitudenberechnung ist in Abbildung 4.6 rechts dargestellt. Im Beispiel in der Abbildung ist ein starker Anstieg am Anfang des Wortes zu erkennen. Der Abfall der Amplitude verhält sich eher gedämpft. Ein drittes Filter mit einem sehr viel kleineren Koeffizienten dient als Langzeitdurchschnitt und wird als Maß für den Pegel der Hintergrundgeräusche genutzt. Dieses Filter benötigt 2.87 Sekunden um 90% der Lautstärke zu erreichen.

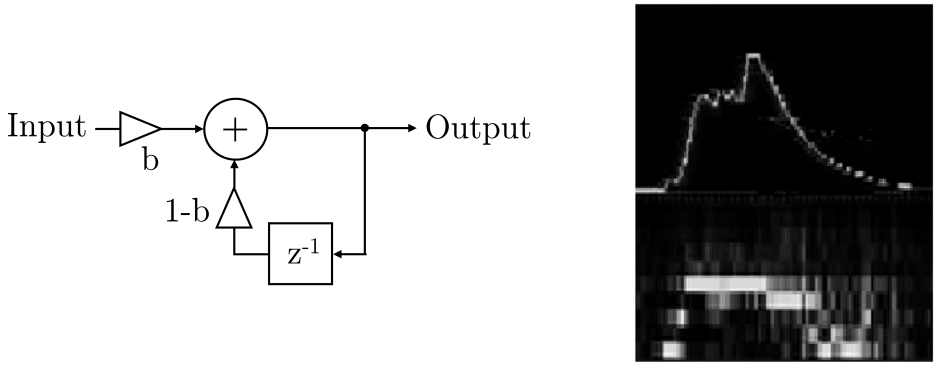


Abbildung 4.6: Tiefpassfilter für die Amplitudenberechnung. Links befindet sich das Filter zur Berechnung der Amplitude des Signals. Der Koeffizient beträgt für den Onset: $b = 0.1$ und für den Decay: $b = 0.0008$. Rechts ist ein beispielhafter Amplitudenverlauf dargestellt, wie er in der *BDE* angezeigt wird.

Der Mittelwert der Amplitude über eine bestimmte Zeit liefert Informationen über die Entfernung einer Geräuschquelle, aber auch über den Gemütszustand eines Sprechers. Innerhalb eines Wortes gibt die Amplitude Auskunft über die Energie des aktuellen Lautes. Wie bereits in Abschnitt 3.3 beschrieben wurde, kann durch die Lokalisierung der Amplitudenmaxima eine grobe Silbenseparation durchgeführt werden. Dies lässt sich nutzen um den Wortspeicher effizient zu durchsuchen.

4.3 Wortsuche durch energiehierarchisches Parsing

Um eine effiziente Suche nach Wörtern im Speicher zu ermöglichen, müssen die Wörter in geeigneten Datenstrukturen gespeichert werden. Dazu wird das auditive Signal auf Amplitudenmaxima hin untersucht und die Maxima werden mit dem zugehörigen Merkmalsvektor in eine sortierte Liste eingefügt. Zusätzlich wird das ganze Wort, repräsentiert durch eine Liste von Merkmalsvektoren, gespeichert. Diese Merkmalvektoren werden hinsichtlich der Frequenzspektren verglichen. Gegebenenfalls muss innerhalb des Wortes, von den Energiemaxima ausgehend, vorwärts und rückwärts gesucht werden.

Amplitudenmaxima

Um nicht eventuell auftretende lokale Maxima fälschlicherweise mit in die Liste aufzunehmen, arbeitet der Prozess der Maximumerkennung mit einer Hysterese. In Abbildung 4.7 ist ein beispielhafter Amplitudenverlauf dargestellt. Die beiden Amplitudenspitzen des Worts „Hallo“ sind mit senkrechten Pfeilen markiert. Die waagerechten

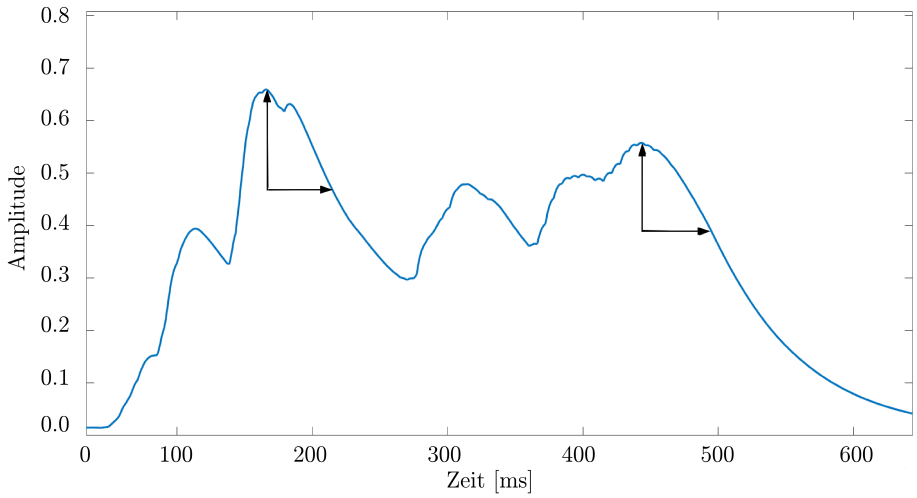


Abbildung 4.7: Amplitudenverlauf des Wortes „Hallo“. Markiert sind die beiden Amplitudenspitzen und der auf die Spitze folgende Wert, an dem das Maximum in die Liste aufgenommen wird.

Pfeile markieren die Amplitudenwerte, die 70% des letzten Maximums betragen. An dieser Stelle wird das vorhergehende Maximum in die Liste aufgenommen. Diese Zahl repräsentiert die Sensitivität des Algorithmus. Sie soll später vom Roboter variiert werden. Wenn er sich in einer Situation befindet, in der erhöhte Aufmerksamkeit gefordert ist, kann er die Sensitivität erhöhen. Mit einer Sensitivität von 90% würden auch die Maxima bei 110ms und bei 320ms in die Liste aufgenommen werden.

Diese Liste der Amplitudenmaxima beinhaltet nur wenige Einträge, je nach Silbenstruktur des entsprechenden Wortes. Das Wort „Hallo“ in Abbildung 4.7 wird mit etwa 60 Merkmalsvektoren gespeichert; die Amplitudenliste besteht jedoch nur aus zwei Vektoren. Diese Vektoren zeigen zusätzlich auf die Position desselben Vektors in der Liste der Merkmalsvektoren des ganzen Wortes. Der aktuelle auditive Strom wird nun nach Amplitudenmaxima durchsucht. Sobald ein Maximum auftritt, wird dieses mit den jeweils ersten Maxima der gespeicherten Wörter verglichen. Die laufenden Vergleiche werden in einer seperaten Liste festgehalten und als derzeitige Hypothesen gehandhabt. Wenn ein Vergleich ergibt, dass eine Hypothese falsch war, dann wird der Eintrag aus der Liste entfernt.

Ähnlichkeitsmaße

Der Vergleich wird zwischen den Frequenzspektren der Merkmalsvektoren durchgeführt. Dazu können verschiedene Methoden angewendet werden. Für diese Arbeit wurden die L1-Norm, die L2-Norm und das Skalarprodukt getestet. Die L1-Norm besteht aus der

Summe der absoluten Differenzen der einzelnen Einträge der Vektoren. Um die größeren Differenzen zwischen den Vektoren stärker zu gewichten, kann die L2-Norm genutzt werden. Hierbei werden die Differenzen der Spektralkomponenten jeweils quadriert, bevor summiert wird.

Beide Normen liefern bei größerer Ähnlichkeit der Vektoren niedrigere Werte. Üblicherweise wird der errechnete Wert daher „Fehler“ genannt. In Abbildung 4.8 sieht man einen Wortvergleich des Wortes „Banane“ mit 15 im Speicher liegenden Wörtern. Hier wurden nur 2 Maxima erkannt. Dies ist dem Umstand geschuldet, dass im Deut-

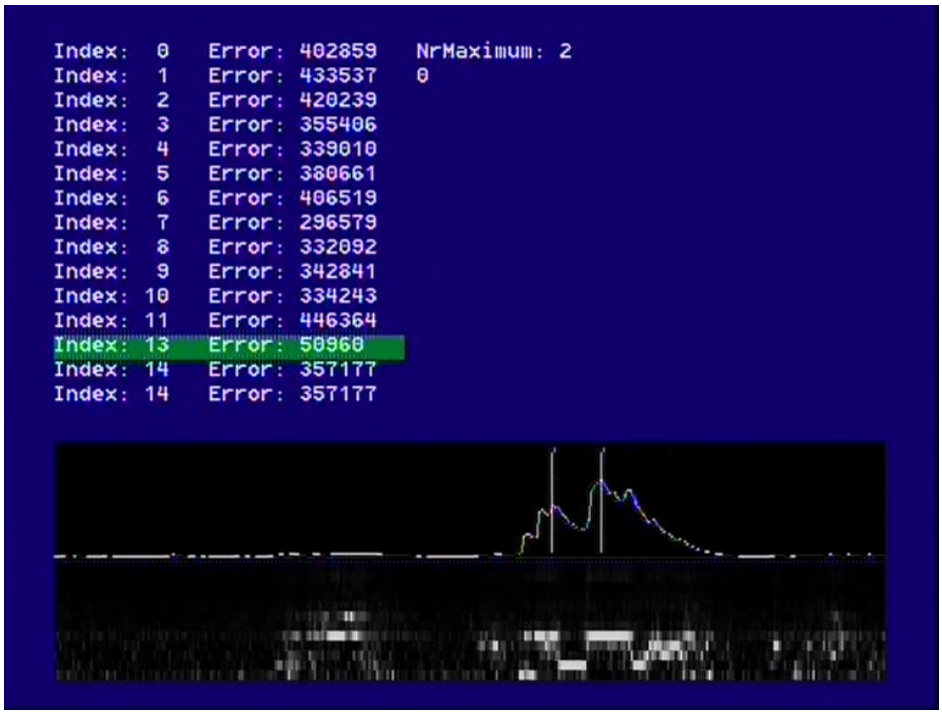


Abbildung 4.8: Wortvergleich über die L2-Norm. Das Wort „Banane“ wird mit 15 im Speicher liegenden Wörtern verglichen.

schen die letzte Silbe oft unbetont ausgesprochen wird. Dadurch hat das „e“ in Banane sehr viel weniger Energie als die vorhergehenden „a“. Der Vergleich wurde mit der L2-Norm durchgeführt. Als drittes Ähnlichkeitsmaß wurde das Skalarprodukt verwendet. Hierbei werden die einzelnen Einträge des Spektrums miteinander multipliziert. Dementsprechend stehen höhere Werte für größere Übereinstimmung. Der Vergleich der Merkmalsvektoren an den Amplitudenmaxima liefert mehrere Informationen. Zunächst wird die Übereinstimmung der Lautqualität am Silbenkern überprüft. Damit können alle Wörter aussortiert werden, die an den jeweiligen Stellen beispielsweise einen anderen Vokal haben. Des Weiteren kann die Zahl der Maxima ein Ausschlusskriterium sein. Für ein eindeutig zweisilbiges Wort können alle einsilbigen, aber auch alle mit mehr

als zwei Silben aussortiert werden. Da aber wie im Beispiel in Abbildung 4.8 manche Silben nicht eindeutig erkannt werden, ist hier etwas Spielraum nötig. Nun enthält die Vergleichsliste nur noch Wörter, deren Silbenanzahl und Vokalqualitäten mit dem aktuellen, zum Vergleich stehenden Wort übereinstimmt. Beispielsweise könnte die Liste die Wörter „Hallo“, „Salto“ und „Karo“ enthalten.

Erhöhung der Auflösung

Um diese Ambiguität aufzulösen, werden ausgehend von den Amplitudenmaxima, die Merkmalsvektoren vor und hinter den Maxima verglichen. Dazu wird jeweils ein Zeitschritt nach vorne gegangen und die entsprechenden Vektoren verglichen. Der Fehler wird akkumuliert bis entweder ein Grenzwert oder das Wortende erreicht wurde. Wenn also auch vor und nach den Maxima genügend Übereinstimmung gefunden wird, bleibt das Wort in der Liste der potentiellen Treffer. Damit können auch die stimmlosen Laute an den Rändern der Silben verglichen werden. Hierbei ist aber zu beachten, dass nun die starken Variationen innerhalb der menschlichen Sprache zum Tragen kommen. Dies sind Variationen in der Betonung, in der zeitlichen Streckung oder Stauchung und andere. Dies muss kompensiert werden, indem beispielsweise nur nach bestimmten Mustern gesucht wird. Als Beispiel sei wieder das Wort „Hallo“ aufgeführt. Wenn nach vorne (in der Zeit zurück) gesucht wird, sollte hauptsächlich Vokalqualität gefunden werden. Im Wort „Salto“ würden aber die hohen Frequenzanteile des „s“ gefunden werden und es damit als potentiellen Treffer ausschließen.

5 Bewertung des Gesamtsystems und Ausblick

In diesem Kapitel wird dargestellt, ob die implementierten Prozesse die in Abschnitt 3.3 definierten Anforderungen erfüllen. Zusätzlich wird zu den jeweiligen Themengebieten ein Ausblick gegeben, mit welchen Methoden und Ideen die Prozesse zukünftig weiterentwickelt werden können.

5.1 Repräsentation der auditiven Signale

Für die Berechnung der Energie wurden mehrere Verfahren getestet. Zur Auswahl standen:

1. Gleitender Mittelwert über Absolutwerte mit unterschiedlicher Bewertung von Onset und Decay (Implementierte Version aus Abschnitt 4.2)
2. Summe über Quadrate der Audiosamples
3. Vektornorm der Spektralkomponenten

Der gleitende Mittelwert über die Absolutwerte ist eine einfache Methode, um die Amplitude eines Signals abzuschätzen. Erweitert mit einer besseren Onset-Erkennung hat sich diese Methode gegenüber Methode 2 und 3 durchgesetzt. Die Energieberechnung über die Summe der Quadrate der Audiosamples liefert weitestgehend ähnliche Ergebnisse wie der gleitende Mittelwert. Die Onsets und auch die Amplitudenmaxima werden an den selben Stellen erkannt. Wenn aber die Amplitude als Schwellenwert für Wortgrenzen genutzt wird, hat es sich als hilfreich herausgestellt, wenn der Amplitudenverlauf „schwerfälliger“ bei der abfallenden Flanke ist. Vor allem Plosive, die sich durch Stille von bis zu 100ms vor dem eigentlichen Laut auszeichnen, sollen nicht als Wortgrenzen erkannt werden. Dies kann durch eine Filterung der Amplitude mit einem Tiefpass geschehen, wie im Ansatz des gleitenden Mittelwerts umgesetzt.

Ein gänzlich anderer Ansatz ist die Nutzung der Vektornorm der Spektralkomponenten. Die Spektralkomponenten werden mit der euklidischen Norm auf die Länge 1 normiert. Der dabei berechnete Faktor repräsentiert die in den Spektralkomponenten enthaltene Energie. Dieses Ergebnis müsste auch mit einem Tiefpass gefiltert werden, um für die Erkennung von Wortgrenzen genutzt werden zu können.

Für die Berechnung der Tonhöhe wird über eine bestimmte Menge von Audiosamples eine Summe gebildet. Die Tonhöhenerkennung wurde für verschiedene Größen dieser Menge getestet. Ein Zeitfenster von 20ms-40ms (entspricht 320-640 Samples) hat sich als sinnvoll herausgestellt. Wenn das Fenster zu groß gewählt wird, ist die Schätzung der Tonhöhe oft erst am Ende des stimmhaften Lautes vorhanden. Bei kurzen Vokalen kommt es vor, dass gar keine Stimmhaftigkeit erkannt wird. Ein sehr kleines Fenster hat den Effekt, dass die erkannte Tonhöhe stark variiert, die Erkennung also unpräzise ist. Deshalb wurde die Größe des Zeitfensters variabel gestaltet. Damit ist eine erste Schätzung der Tonhöhe oft schon nach wenigen Samples vorhanden und in den folgenden Zeitschritten wird diese präzisiert, indem die Größe des Zeitfensters entsprechend angepasst wird.

Die in Abschnitt 4.2 beschriebene Berechnung des Frequenzspektrums wurde effizienzorientiert implementiert. Hierbei werden 12 Goertzelfilter genutzt, über die die Energie in bestimmten Frequenzbändern berechnet werden. Obwohl bei dieser Methode alle Frequenzbereiche zuverlässig erkannt werden (getestet mit reinen Sinustönen), treten Leck-Effekte auf. Dies ist dadurch zu erklären, dass das Analysefenster durch eine Rechteckfunktion erstellt wird. Diese störenden Effekte können durch die Nutzung anderer, in der Signalverarbeitung weit verbreiteter Fenster, gemindert werden.

Um die Berechnung des Frequenzspektrums zu präzisieren, sind beispielsweise Hanning - und Blackmann-Harris-Fenster geeignet. Diese Methoden sind allerdings rechenintensiver, da jedes Sample zusätzlich mit einem Faktor multipliziert werden muss. Da außerdem die verschiedenen Bandbreiten der Goertzelfilter unterschiedlich große Fenster erfordern, ist hier mit zusätzlichem Aufwand im Algorithmus zu rechnen. Dennoch ist für zukünftige Arbeiten vorgesehen, diese Verbesserungen zu implementieren und zu testen.

Ein weiterer Ansatz, dessen Implementierung und Test vorgesehen ist, ist die „Intrinsic time-scale decomposition“ (ITD) von Frei und Osario [**?**]. Bei dieser Methode wird ein Signal in einzelne Schwingungen der im Originalsignal enthaltenen Frequenzen zerlegt. Dazu wird aus den lokalen Maxima und Minima über Mittelwertberechnung die jeweils höchste Frequenz herausgefiltert. Diese Methode bietet eine Zeitauflösung, die nur durch die enthaltene Frequenz begrenzt wird (im resultierenden Signal muss mindestens eine komplette Schwingung vorhanden sein). Außerdem ist mit der ITD eine einstellbare Frequenzauflösung möglich. Ein weiterer Vorteil dieser Methode ist, dass die Amplitudenberechnung durch die benutzten Maxima schon implizit enthalten ist. Die Implementierung und Testung dieses Ansatzes ist auf Grund der beschriebenen Vorteile zukünftig vorzunehmen.

5.2 Suche nach Wörtern und Wortvergleich

Das vorgestellte Verfahren der Speicherung von gesprochenen Wörtern als Liste von Energiemaxima liefert vielversprechende Ergebnisse. Der Vergleich der Merkmalsvektoren an diesen Stellen zeigt, dass Vokalqualitäten zuverlässig wiedererkannt werden können. Hierbei wurden zwei unterschiedliche Methoden getestet. Zum einen werden mehrere Repräsentationen desselben Wortes im Speicher des Roboters hinterlegt. Dies geschieht durch mehrmaliges Sprechen und Speichern (Sprachspiel Wörter lernen), vorzugsweise durch unterschiedliche Sprecher. Zum anderen werden die Vektoren „verschmiert“, indem ein Mittel aus zeitlich hintereinanderliegenden Vektoren gebildet wird (beispielsweise wird ein Mittel aus einem Vektor vor und einem nach dem Energiemaximum gebildet). Der Merkmalsvektor des aktuell zu vergleichenden Wortes wird dann mit diesem Vektor verglichen. Beide Methoden verbessern die Worterkennung. Die Repräsentation eines Wortes durch mehrere gesprochene Beispiele erhöht die Wahrscheinlichkeit, dass ein Wort mit der passenden Tonlage im Speicher gefunden wird. Das Verschmieren des Vektors dient vor allem der Kompensation der Leck-Effekte, beschrieben im vorangehenden Abschnitt.

Da der Roboter in Zukunft mit seiner Umwelt kommunizieren soll, ist es nötig, dass zwischen bekannten und unbekannten Worten unterschieden werden kann. Ein wichtiger Teil, der in den Prozess des Wortvergleichs zukünftig eingebaut werden muss, ist daher die Möglichkeit der Speicherung von Wörtern, wenn die Suche im Speicher erfolglos war. Dies ermöglicht dem Roboter, Wörter, die er noch nicht kennt, zu lernen.

6 Zusammenfassung

In der vorliegenden Arbeit wurde eine neuartige Herangehensweise an die Sprachverarbeitung in der humanoiden Robotik vorgestellt. Dazu wurden einige wichtige Situationen beschrieben, in denen ein Roboter ein verlässlich funktionierendes Kommunikationssystem benötigt. Dazu zählen die Sprachspiele: *Wörter lernen*, *Worterkennung*, *Wörter sprechen*, *Objekterkennung* und *Programmierung*.

Das Sprachspiel Programmierung war für die vorliegende Arbeit von besonderer Bedeutung, da es als grundlegendes Beispiel für die vorgestellten Ideen diente. Die Methoden und Algorithmen wurden im Hinblick darauf entwickelt, den Roboter über gesprochene Sprache zu programmieren. Mit diesem Ziel wurden folgende Anforderungen an ein Kommunikationssystem formuliert:

- Invarianz der Perzeption gegenüber Variationen des auditiven Signals, die den Inhalt nicht verändern.

- Suche von Wörtern im Speicher über Energiehierarchien. Dazu Speicherung dieser Wörter in geeigneter Datenstruktur.

- Vergleich der Merkmalsvektoren durch ein geeignetes Ähnlichkeitsmaß.

Die auditiven Signale werden analysiert und die Ergebnisse im 10ms Rhythmus in einen Merkmalsvektor gespeichert. Dieser enthält das Frequenzspektrum mit 12 Frequenzbändern, die zugehörige Vektornorm, den Amplitudenwert und die Grundfrequenz. Diese Merkmalsvektoren werden im Speicher hinterlegt. Zusätzlich wird eine sortierte Liste angelegt, in die die Merkmalsvektoren mit Amplitudenmaxima gespeichert werden. Diese Vektoren repräsentieren den Kern jeweils einer Silbe. Dort treten die Laute höchster Sonorität und damit höchster Energie auf. Dieser Umstand wird genutzt, um Wortvergleiche energiehierarchich durchzuführen. Zunächst werden die Frequenzspektren an den Stellen höchster Energie verglichen. Damit kann die Liste potentieller Treffer schon stark dezimiert werden. Erst wenn nach diesem Schritt noch mehrere potentielle Treffer vorhanden sind, wird an Stellen niedrigerer Energie verglichen, wobei dann auch Laute geringerer Klangfülle betrachtet werden. Als Ähnlichkeitsmaß dienen Vektornormen wie die L1-Norm (auch Manhatten-Norm), die L2-Norm (auch euklidische Norm) oder das Skalarprodukt.

Diese neuartige Herangehensweise an die Sprachverarbeitung bietet eine effiziente und robuste Lösung, dem humanoiden Roboter Myon die Nutzung von Sprache zu

ermöglichen. Die Konzentration auf die Energiezentren innerhalb von Wörtern reduziert die Auflösung zunächst drastisch. Dadurch kann mit minimalem Rechenaufwand eine große Menge an Wörtern durchsucht werden. Wenn es anschließend noch kein eindeutiges Ergebnis gibt, wird die Auflösung schrittweise erhöht.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis