Dipl.-Ing. Moheb Mekhaiel

Videounterstützte, robuste Merkmalsextraktion

für die Spracherkennung in Echtzeit

Videounterstützte, robuste Merkmalsextraktion für die Spracherkennung in Echtzeit

Zur Erlangung des akademischen Grades eines

DOKTOR - INGENIEURS

von der Fakultät für
Elektrotechnik
der Universität Fridericiana Karlsruhe

genehmigte

DISSERTATION

von

Dipl.-Ing. Moheb Mekhaiel
aus Giza, Ägypten

Tag der mündlichen Prüfung:	15.05.2000
Referent:	Prof. Dr.-Ing. K. Kroschel
Korreferent:	Prof. Dr.-Ing. H.-J. Jentschel
Karlsruhe:	25. Mai 2000

Danksagung

Mein Dank gilt in ganz besonderer Weise Herrn Prof. Dr.-Ing. K. Kroschel, der mir durch das entgegengebrachte Vertrauen und die gewährten Freiheiten das Entstehen der vorliegenden Arbeit überhaupt erst ermöglicht hat.

Mein Dank geht selbstverständlich auch an all diejenigen, die mir mit ihren Diplom- und Studienarbeiten wertvolle Dienste geleistet haben. Ebenso bedanken möchte ich mich bei meinen Kollegen, die mir mit ihren Anregungen und ihrer Kritik wertvolle Hinweise geschenkt haben. Insbesondere möchte ich mich bei Herrn Dipl.-Ing. Martin Heckmann für seine sorgfältige Durchsicht dieser Arbeit herzlich bedanken.

Ebenso bedanken möchte ich mich bei Herrn Prof. H.-J. Jentschel für die Übernahme des Korreferates.

Mein größtes Dankeschön geht an Elisabeth Hilz für ihre Hilfe bei der Eingabe und Korrektur der Arbeit. Meine Eltern, die mit ihrer Liebe und Unterstützung es überhaupt ermöglicht haben, dass ich soweit mit meinem Studium gekommen bin, haben einige dankende Worte in ihrer Sprache verdient.

كَلِمة شُكر

أَود أَن أَتقدم بجَزِيل الشُكرُ وَ خَالص العرفَان إِلَّى وَالدَّاى الأَحبَّاء علَى عنَايتهم و شَدِيد إِهتمَامهم بِى خِلَال كُلّ سَنَوَات درَاستَى، لولَا مُدَاوَمتهم علَّى الدُعَاء لَى وَ محبتهم الفَائِقة لِمَّا تمكنتُ مِن إِنهَاء هَذَا العمل.

ALLES REDEN IST SO VOLL MÜHE, DASS NIEMAND DAMIT ZU ENDE KOMMT. DAS AUGE SIEHT SICH NIEMALS SATT, UND DAS OHR HÖRT SICH NIEMALS SATT. WAS GESCHEHEN IST, EBEN DAS WIRD HERNACH SEIN. WAS MAN GETAN HAT, EBEN DAS TUT MAN HERNACH WIEDER, UND ES GESCHIEHT NICHTS NEUES UNTER DER SONNE. GESCHIEHT ETWAS, VON DEM MAN SAGEN KÖNNTE: „SIEH, DAS IST NEU“. ES IST LÄNGST VORHER AUCH GESCHEHEN IN DEN ZEITEN, DIE VOR UNS GEWESEN SIND.

[KOHELET, A21.1: 8-11]

Inhaltsverzeichnis

Verzeichnis der verwendeten Bezeichnungsregeln und Abkürzungen

Bezeichnungsregeln

x	skalare Größen
$\tilde{x}$	Schätzwert für x
$\mathbf{x}$	Spaltenvektor
$\mathbf{x}^T$	Zeilenvektor
$\mathbf{A}$	Matrix
$\bar{r}$	Mittelwert für r
$E\{x\}$	Erwartungswert für x
$\dot{x}$	erste Ableitung von x
$\ddot{x}$	zweite Ableitung von x
$x^{(i)}$	Iterationsschritt i
σ_{xx}	Standardabweichung für x

Abkürzungen

AKF	Autokorrelationsfunktion
ASE	automatische Spracherkennungssysteme
DTW	*Dynamic Time Warping*
ER	Erkennungsrate
FFT	schnelle Fourier-Transformation (engl.: *Fast Fourier Transformation*)
HMM	Hidden-Markov-Modell
KKF	Kreuzkorrelationsfunktion
LDA	*Linear Discriminant Analysis*
LPC	*Linear Predictive Code*
LTW	*Linear Time Warping*
MAP	Maximum A-posteriori-Kriterium
MFCK	Melfrequenz-Cepstral-Koeffizienten
NN	künstliche Neuronale Netze
PAL	*Phase Alternation Line* (Video-Farbsystem)
PARCOR	PARtial CORrelation-Koeffizienten
PCA	*Principal Component Analysis*
RNN	*Recurrent Neural Network*
ROI	*Region of Interest*
SAD	Summe absoluter Differenzen, (engl.: *sum of absolute differences*)
SNR	Signal-zu-Rauschleistungsverhältnis (engl.: *signal to noise ratio*)
TDNN	*Time Delay Neural Network*
VQ	Vektorquantisierung
WD	Wahrscheinlichkeitsdichtefunktion
WER	Worterkennungsrate

Zusammenfassung

Automatische Spracherkennungssysteme (ASE) bieten eine einfache und menschennahe Kommunikation zwischen Menschen und Maschinen. Klassische Erkennungssysteme segmentieren die fließende Sprache in kleinere Einheiten, die dann den vorgegebenen Klassen (z.B. Wörter, Phoneme oder Tri-Phoneme) zuzuordnen sind. Dabei wird ausschließlich das akustische Signal verwendet. Bei der Kommunikation unter den Menschen spielen optische Informationen eine zusätzliche Rolle. Deshalb erlangen optische Informationen bei der Mensch-Maschine-Interaktion immer größere Bedeutung, was eine neue Form der Kommunikation, nämlich das Lippenlesen, in den Blickpunkt des Interesses treten ließ.

Die vorliegende Arbeit präsentiert ein komplettes System zur Erkennung einzelner Wörter, bei dem die optischen Informationen des Sprechers zur Unterstützung der Interpretation des gesprochenen Vokals bzw. Wortes herangezogen werden. Die Fusion der Video- mit den Audiomerkmalen und die Berechnung von robusten Merkmalen aus dem Videosignal stellen dabei die Schwerpunkte der Arbeit dar, während für die Merkmalsextraktion aus der Sprache klassische Verfahren wie die LPC-Analyse angewandt werden. Die so genannte multimodale Erkennung anhand der Audio- und Videodaten führt vor allem bei stark störenden Hintergrundgeräuschen zu einer insgesamt besseren Erkennungsrate.

Kapitel 1

Einleitung

Bei der Kommunikation zwischen Mensch und Maschine haben sich einige Eingabegeräte wie z.B Schalter und Tastaturen durchgesetzt. Ausgabegeräte dienen der Kommunikation in die entgegengesetzte Richtung; so werden dem Menschen von der Maschine Informationen über Warn- und Lichtsignale oder Displays mitgeteilt.
Informationen unter den Menschen werden hauptsächlich über Sprache, aber auch visuell ausgetauscht. Um die Schnittstelle Mensch-Maschine „humaner“ zu gestalten, wird die Palette der Eingabegeräte bei manchen Maschinen um die Spracherkennung erweitert. Die Spracherkennung hat die Aufgabe, die akustischen Signale des Bedieners in maschinennahe Befehle umzusetzen.

Klassische Spracherkenner bieten nicht nur eine für den Menschen einfachere Kommunikationsart mit einer Maschine, sondern sie sind in manchen Situationen sogar kaum durch konventionelle Eingabegeräte zu ersetzen. Man denke an die Situation eines Arbeiters, der gerade mit seinen Händen eine Tätigkeit ausführt und dennoch eine Maschine mittels Sprachbefehl bedienen möchte. Der Vorschlag des Verkehrsministers Reinhard Klimmt, im Jahre 2000 den Gebrauch von Handys ohne Freisprechanlage während der Autofahrt zu verbieten [Taz99] - was in anderen Ländern schon längst Gesetz ist -, ist ein Beispiel für die zunehmende Bedeutung der Spracherkennung. In diesem Fall muss die Bedienung des Handys während der Fahrt per Sprachbefehl erfolgen.

Gerade in den Situationen, in denen die Spracherkennung eingesetzt wird, trifft man auf störende Umgebungsgeräusche im Hintergrund. Dabei handelt es sich um Umgebungen wie Büroräume, Fertigungshallen, Autoinnenräume, Züge, Straßen usw. In solchen Umgebungen beeinflussen die Hintergrundgeräusche die Leistung eines Erkenners sehr stark. So kann zum Beispiel eine Worterkennungsrate (WER) von über 90% bei einem Signal-zu-Rausch-Verhältnis (SNR) von ca. 20 dB auf weniger als 50% sinken, wenn nur noch ein SNR von 5 dB existiert [Sko98]. Eine so niedrige Erkennungsrate macht die Kommunikation mit einer Maschine kaum brauchbar.
Manche Erkennungssysteme versuchen daher, die Umgebungsgeräusche zunächst zu unterdrücken. Dabei wird das vom Mikrofon aufgenommene Signal digitalisiert und so verarbeitet, dass die Geräuschkomponenten im Signal unterdrückt werden. Die Spracherkennung arbeitet dann mit dem „verbesserten“ Signal. Dabei unterscheidet man fol-

gende Vorgehensweisen:

- Einkanalige Geräuschunterdrückung
- Geräuschkompensation mit einer Referenzquelle
- Mehrkanalige Systeme mit Hilfe eines Mikrofonarrays

Ausführliche Arbeiten über diese Verfahren sind in [Rei86], [Lin88], [Kro93], [Mek94], [Tro95] zu finden.

Jedes dieser Systeme hat leider seine Nachteile. So versuchen *einkanalige Systeme*, das Hintergrundgeräusch während der Sprachpausen zu schätzen, wobei ein stationäres Geräusch vorausgesetzt wird. Bei instationären Geräuschen scheitert deswegen dieses Verfahren und führt u.U. sogar zu einer Verschlechterung des Signals.
Bei der *Geräuschkompensation* muss eine Referenzquelle, also ein weiteres Mikrofon, das möglichst nur die Geräusche aufnehmen soll, vorhanden sein. In der Praxis gelangen aber auch Sprachanteile zum Referenzmikrofon und eine Entkoppelung des Referenzsignals vom Sprachsignal kann nicht mehr angenommen werden.
Die besten Unterdrückungsergebnisse liefern die *mehrkanaligen Systeme*, die das Signal mit mehreren Mikrofonen gleichzeitig aufnehmen. Sie sind jedoch für einen praktischen Einsatz relativ kompliziert bezüglich Installation und Verarbeitungsaufwand.
Die Leistung der Geräuschunterdrückung hält sich in Grenzen. In Situationen mit instationären Geräuschen sowie kurzanhaltendem, sehr starkem Lärm (z.B. Schlagen von Türen, Betätigen der Autohupe, Aufprall von Gegenständen auf den Boden) oder bei temporärem Ausfall des akustischen Signals scheitert dieses Konzept.

Es drängt sich nun die Frage auf: Wie meistert der Mensch solche Situationen? Die Interpretation des Gesprochenen beim Menschen ist hauptsächlich aus zwei Gründen besser:

1. Zum einen ist der Mensch in der Lage, mit Hilfe des Kontexts, der Situation und der Vorkenntnis der Person die akustisch nicht wahrgenommenen Anteile der Sprache zu „erahnen“. Es handelt sich dabei um eine sehr raffinierte und über sehr viele Jahre hinweg trainierte Fähigkeit des menschlichen Gehirns. In der Tat versuchen moderne Erkennungssysteme, diese Fähigkeit durch das Definieren von einfachen Grammatikregeln nachzubilden. So wird z.B. bei bestimmten Situationen entweder das Wort ‘ja’ oder ‘nein’ erwartet. Andere Wörter werden dann in diesem Zusammenhang ausgeschlossen.

2. Der andere Grund wurde bereits am Anfang dieses Abschnitts erwähnt: Der Mensch wertet nämlich auch die visuellen Informationen aus. Anhand des Gesichtsausdrucks des Sprechers wird sehr viel über den Zusammenhang vermittelt. Eine weitere wichtige Quelle ist die Lippenbewegung. Der Mensch kann bereits nur durch das Beobachten der Lippenbewegung viele Wörter erkennen. Diese Fähigkeit kann mittels Training sogar wesentlich verbessert werden.

Es liegt also nahe, dass Maschinen auch den visuellen Kanal auswerten. Die Zeiten, in denen sich nur einige Forschungsinstitutionen die teuren Ausrüstungen für Videoverarbeitung leisten konnten, sind schon längst vorbei. Rechner oder Maschinen mit Videoverarbeitungsgeräten auszustatten, ist heutzutage kein nennenswerter Kostenfaktor mehr.

Wie die Verarbeitung genau aufgebaut werden kann, um die Lippenbewegung interpretieren zu können, und wie man diese visuellen Informationen zusammen mit den akustischen Informationen fusionieren kann, sind zwei der Hauptthemen dir vorliegenden Arbeit.

1.1 Motivation der Arbeit

Die zu Beginn dieser Arbeit bereits vorgeschlagenen Modelle für die Lippenmodellierung [Pet84], [Fin86], [Sej90], [Pra93], [Gol93] waren unserer Ansicht nach nicht für den praktischen Einsatz geeignet. Viele dieser Modelle benötigen eine spezielle Laborausrüstung, besondere Markierungen an den Lippen oder eine sehr gute Beleuchtung. Ferner war bei keinem der uns bekannten Systeme ein Einsatz in Echtzeit vorgesehen. Solche Systeme sind daher für den praktischen Einsatz gar nicht oder nur bedingt geeignet. In dieser Arbeit haben wir daher zwei Ziele angestrebt. Das erste Ziel ist die Suche nach robusten Algorithmen zur Lippenmodellierung. Die Modellierung muss folgende Gesichtspunkte erfüllen:

1. Es wird eine normale Umgebung vorausgesetzt (s. Abschnitt 1.2 für eine ausführliche Beschreibung der vorausgesetzten Umgebung). Teure oder spezielle Aufnahmegeräte sind nicht zu verwenden.

2. Die Modellierung soll sprecherunabhängig sein.

3. Eine Markierung der Lippen ist nicht zulässig.

4. Die Modellierung muss echtzeitfähig sein.

5. Die Merkmale dieses Modells müssen sich für eine videobasierte Spracherkennung eignen.

6. Das gesamte System soll für einen normalen Benutzer leicht anwendbar sein. Das System muss sich, auch ohne lästige Parametereingabe vom Benutzer, automatisch einstellen können.

Das zweite Ziel ist die Untersuchung der Möglichkeiten, die Videomerkmale zusammen mit den Audiomerkmalen zu fusionieren. Man strebt eine insgesamt bessere Erkennungsrate beim gleichzeitigen Auswerten der beiden Kanäle an. Dabei ergeben sich viele Probleme:

1. Die Dimension des Merkmalsvektors aus dem Videokanal und die Dimension des Merkmalsvektors aus dem Audiokanal sind stark unterschiedlich. Eine direkte Erweiterung des Merkmalsvektors stellt daher keine ideale Lösung dar.

2. Die Datenrate im Videokanal, die von der Audioabtastrate abhängt, unterscheidet sich von der Bildrate des Videokanals, die im PAL-System mit 25 Bildern pro Sekunde gegeben wird.

3. Es müssen Zuverlässigkeitsparameter für jeden Kanal definiert werden, so dass die Merkmale, die aus dem zuverlässigeren Kanal stammen, dementsprechend gewichtet werden.

In diesem Zusammenhang wurden mehrere Fusionskonzepte wie z.B. *early fusion* oder *late fusion* untersucht.

Es gibt eine Fülle von weiteren Anwendungen (s. Abschnitt 1.3), bei denen die in unserer Arbeit vorgestellte Auswertung der Lippenkonstellationen hilfreich ist. Die Lokalisierung des Gesichts und der Mundregion kann z.B. bei der Bildübertragung ausgenutzt werden. Dabei kann man Teile des Bildes, die sich schneller verändern, wie z.B. Gesicht und Lippen, mit einer höheren Rate übertragen als den Hintergrund und spart damit Übertragungsbandbreite. Eine weitere interessante Anwendung ist die Hilfestellung für Gehörlose. Ein Lippenleser kann ihnen dabei helfen, die Bewegungen der Lippen zu deuten. Weitere Anwendungen, bei denen ein Lippenleser eingesetzt wird, werden in Abschnitt 1.3 näher betrachtet.

1.2 Systemaufbau

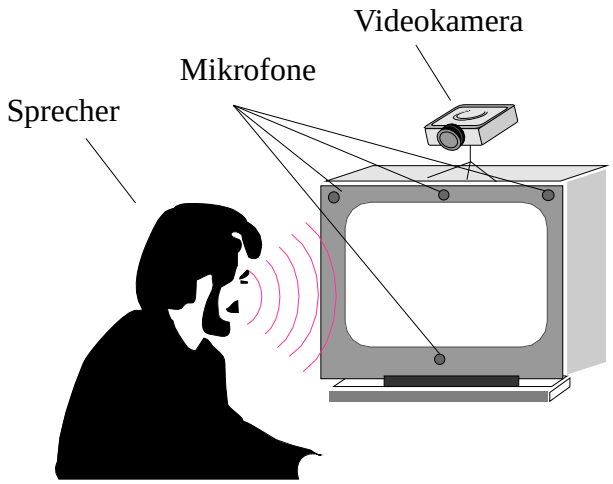


Abbildung 1.1: *Die Arbeitsumgebung des Systems*

Die Umgebung besteht aus einem Rechner, mindestens einem Mikrofon und einer Videokamera (s. Abb. 1.1). Einzelbilder werden durch eine Framegrabber-Karte digitalisiert und in Echtzeit in den Arbeitsspeicher des Rechners geschrieben. Der Benutzer darf seinen Körper und seinen Kopf frei bewegen. Die Videokamera muss dabei automatisch den Bewegungen des Sprechers folgen und dafür sorgen, dass sein Kopf mit ausreichender Auflösung erfasst wird.

Die Steuerung der Kamera erfolgt über eine RS232-Schnittstelle (serielle Schnittstelle). Dadurch lassen sich u.a. das Einzoomen und sowohl das horizontale als auch das vertikale Schwenken der Kamera softewaremässig vom Rechner aus steuern.
Bezüglich der Lichtverhältnisse und der Lichtverteilung gibt es keine besonderen Voraussetzungen. Eine künstliche Beleuchtung sowie z. B. Lichteinfall durch ein Seitenfenster sind zulässig. Es sind also keine besonderen Vorbedingungen im Bezug auf die Farbtemperatur des Lichtes, die Lichtstärke oder Schattenverteilung erforderlich. Die Lokalisierung des Gesichtes, die Verfolgung durch die Kamera, die Lippenmodellierung und die Auswertung der Lippenbewegung müssen sehr schnell (quasi in Echtzeit) abgewickelt werden.

1.3 Weitere Anwendungsgebiete des Lippenlesers

Lippenleser werden nicht nur auf dem Gebiet der audio-visuellen Spracherkennung eingesetzt. Zur Zeit gibt es viele Forschungsgebiete, in denen die Lippensuche sowie eine Modellierung und Auswertung der Lippenbewegung benötigt werden. Die wichtigsten Gebiete werden in diesem Abschnitt vorgestellt. Jedoch unterscheiden sich die Ziele der verschiedenen Forschungen zum Teil erheblich, so dass die Methodik, mit der man an das Problem des Lippenlesens herangeht, von dem jeweiligen Ziel abhängt. In diesem Abschnitt sollen die Anforderungen, die Ziele und die Randbedingungen der verschiedenen Gebiete erläutert werden.

1.3.1 Synchronisation von Audio- und Videoaufnahmen

Einige ältere Filmaufnahmen bestehen aus zwei verschiedenen Medien für jeweils die Tonspur und den Film. In diesem Fall muss beim Abspielen die Synchronisation zwischen Ton und Bild beachtet werden. Das Problem der Synchronisation tritt auch bei künstlich erzeugten Animationen auf, wo die Bewegung der Lippen an eine Sprachaufnahme angepasst oder synchronisiert werden soll. In diesem Fall werden die Zusammenhänge zwischen den gesprochenen Daten und den dazugehörigen Bildern gesucht. Eine eindeutige Abbildung zwischen der kleinsten Einheit der Sprache (Phonem) und der kleinsten Einheit aus dem visuellen Kanal (Visem) besteht nicht [Pah98]. Manche Phoneme, die akustisch ganz ähnlich sind, unterscheiden sich stark im visuellen Kanal, wie z.B. /m/ und /n/. Ein einfacher Ansatz baut auf der Detektion der Sprachpausen auf, die sowohl in der Bildsequenz als auch in der Sprache zu detektieren sind. Die Synchronisation findet dann immer am Anfang und Ende einer Sprachpause statt.
Ein weiterer Ansatz baut auf der Erkennung einiger Vokale bzw. Phoneme auf. Dabei ist es nicht unbedingt notwendig, alle Phoneme und Viseme an sich zu erkennen, sondern es würde genügen, nur solche Phoneme zu erkennen, deren Beziehungen zu den dazugehörigen Visemen eindeutig sind.
Dieser Ansatz wurde in [Pah98] erweitert. Mit einem Fuzzy-Modell, bei dem mehrere Phoneme zu einer einzigen Klasse zusammengefasst werden -gleiches gilt auch für die Visemklassen-, werden dann mit Hilfe von einfachen Regeln Zusammenhänge zwischen den Audio- und Videoklassen in einer Lernphase gefunden. Diese werden dann für die Synchronisation verwendet. In [Tam98] wurde ein *Hidden Markov Model*

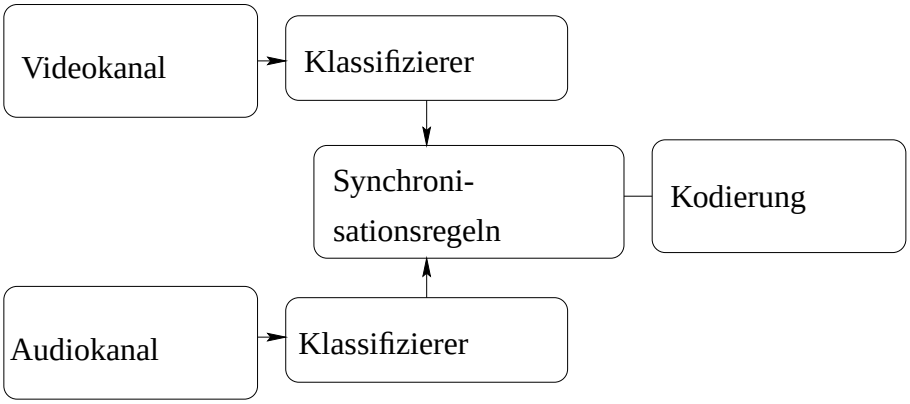


Abbildung 1.2: *Prinzip der Synchronisation von Audio- und Videodaten*

für die Erkennung von vordefinierten audio-visuellen Einheiten, die genau einen Laut umfassen, eingesetzt. Dabei besteht der Merkmalsvektor aus Merkmalen, die sowohl dem Audio- als auch dem Videokanal entnommen sind.

1.3.2 Abbildung von Lippenbewegungen

Bei einigen graphischen Anwendungen, wie z.B. *talking heads* (computeranimierte Gesichtsmodelle), ist man bestrebt, eine möglichst realistische Lippenbewegung zu erzeugen. Es wurden zahlreiche Arbeiten auf diesem Gebiet veröffentlicht. Das Interesse an sprechenden Köpfen scheint uralt zu sein. Der Philosoph Albertus Magnus soll bereits im 13. Jahrhundert ein mechanisches Gesicht zusammengebaut haben, das sprechen konnte [Rub98].
In unserer modernen Zeit werden hauptsächlich 3 verschiedene Wege beschritten:

- Sprache-zu-Bild-Abbildung (engl.: *speech-driven*),
- Bild-zu-Bild-Abbildung,
- rein synthetisch (engl.: *text-driven*)

Der erste Weg versucht, mit einer realistischen Sprachaufnahme auszukommen. Dabei werden die Sätze, die man auf das Gesichtsmodell abbilden will, von einer Person akustisch aufgenommen. Die Parameter des Gesichtsmodells, die die verschiedenen Mundbewegungen anregen sollen, werden direkt aus den Parametern des Sprachsignals gewonnen. Es wird also eine Abbildung zwischen den aus dem Sprachsignal gewonnenen Parametern und den Parametern des Gesichtsmodells benötigt. Im Englischen bezeichnet man diesen Weg mit *speech-driven.*
Der zweite Weg umgeht die audio-visuelle Beziehung. Die Parameter des Gesichtsmodells werden aus einer realistischen Videoaufnahme eines Sprechers gewonnen. Hier wird also ein Lippenleser benötigt. In [Rev98] wurden bei der Videoaufnahme die Lippen des Sprechers blau bemalt. Dies erleichtert das Extrahieren der Parameter (Breite und Höhe des Mundes von innen und außen). Diese 4 Parameter sind dann auf die Parameter des 3-dimensionalen Lippenmodells abzubilden.

(a)  (b)  (c) 

Abbildung 1.3: *Beispiele für einige Kopf- bzw. Lippenmodelle: (a) und (b) Royal Institute of Technology (KTH), Schweden, (c) ICP, Grenoble, France*

Der dritte Weg benötigt weder Sprach- noch Videoaufnamen, dabei werden ähnliche Verfahren wie bei der Sprachsynthese angewandt (engl.: *text-driven*) [Tam98].

1.3.3 Verbesserung der Verständlichkeit bei der Sprachübertragung

Bei der Sprachkommunikation zwischen Menschen kommt es häufig vor, dass einige Sprachanteile durch Umgebungsgeräusche oder Lärm verdeckt werden. Der Mensch besitzt aber die Fähigkeit, sowohl aus dem Zusammenhang als auch durch Beobachtung der Lippenbewegung des Sprechers auf die verdeckten Anteile Rückschlüsse zu ziehen. Die Zusatzhilfe, die auf der Interpretation der Lippenbewegung aufbaut, entfällt bei der Sprachübertragung.
Mit Hilfe der visuellen Daten des Sprechers können diese *fehlenden* Sprachanteile bereits vor der Übertragung „manipuliert“, also mit künstlich erzeugter Sprache ersetzt werden. Dabei unterscheidet man zwei Möglichkeiten:

- Die verdeckten Sprachanteile werden vom Geräusch völlig verdeckt. In diesem Fall werden solche Stellen durch eine aus den visuellen Informationen synthetisierte Sprache ersetzt.

- Die verdeckten Sprachanteile sind teilweise gestört. Man versucht in diesem Fall, bestimmte Parameter aus diesen Anteilen zu gewinnen; und die Parameter mit Hilfe der visuellen Daten so zu verändern, dass die Rücktransformation eine leichter verständliche Sprache liefert.

Es handelt sich also um eine Synthese bzw. Verbesserung des Sprachsignals, daher bezeichnet man solche Systeme auch als ‘*Video-zu-Sprache-Umwandler*’. Dies ist die Umkehrabbildung von 1.3.2.

Der Schwerpunkt solcher Arbeiten ist die Art, die aus dem visuellen Kanal gewonnenen Daten auf das Sprachsignal einwirken zu lassen.
Für solche Anwendungen ist es maßgebend, korrekte Beziehungen zwischen dem Video- und dem Audiokanal herzustellen. Es handelt sich hier um ein noch wenig erforschtes Gebiet. In der Arbeit von Girin [Gir98] wurde u.a. die Vokal-Vokal-Transformation untersucht. Aus der Bewegung der Lippen werden 3 Parameter extrahiert: der horizontale und der vertikale Abstand zwischen den Lippen sowie die Gesamtaußenfläche der Lippen.

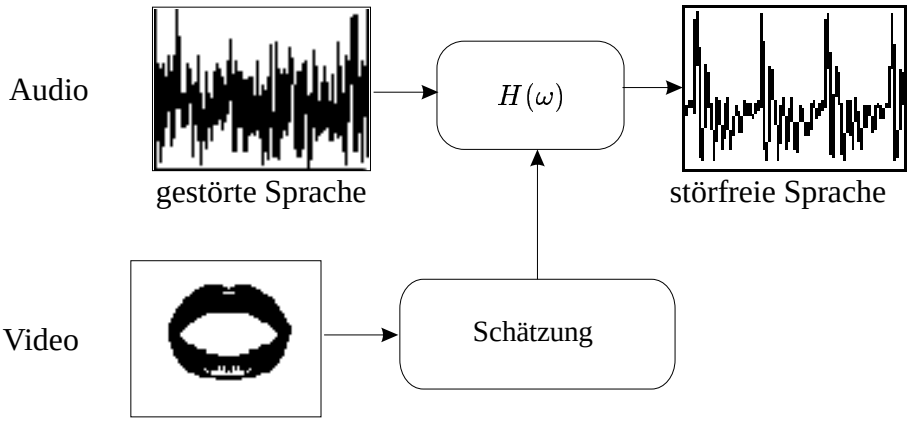


Abbildung 1.4: *Ein Beispiel für die Sprachverbesserung mittels Videoparameter*

Aus dem Audiokanal werden die PARCOR-Koeffizienten berechnet *(PARtial CORrelation).* Mit dem Filter $H(\omega)$, dessen Koeffizienten sich durch die Parameter des Videokanals berechnen lassen, werden dann die Parameter des Audiosignals so verändert, dass ein geräuschfreies Signal rekonstruiert werden kann (s. Abb. 1.4).

1.3.4 Verbesserung der Worterkennungsraten bei der Spracherkennung

Dies ist die Anwendung, die es in dieser Arbeit zu realisieren gilt. Forscher auf dem Gebiet der Spracherkennung sind sich schon längst einig, dass die Erkennungsrate bei einem schlechter werdenden Signal-zu-Rausch-Verhältnis (SNR) abnimmt. Der Mensch scheint aber im Gegensatz zur Maschine mit diesem Problem leichter zurecht zu kommen. Das liegt daran, dass der Mensch nicht nur den akustischen Kanal, sondern auch noch andere Kanäle zum Verstehen der Sprache nutzt.
Im Jahre 1976 hat McGurk [Mac76] mit einem Experiment die Wechselwirkung zwischen Akustik und Lippenbewegung beim Menschen demonstriert. Versuchspersonen wurde das akustische Signal von /ga/ und synchron dazu auf einem Monitor die Lippenbewegung eines /ba/ vorgespielt. Der Laut, für den sich die Versuchspersonen entschieden, war beiden Lauten ähnlich und lag sozusagen „zwischen“ den beiden vorgespielten, nämlich /da/. Dies ist natürlich eine künstliche Situation, die in der Realität nicht in dieser Form vorkommt. Im Alltag tritt vielmehr der Fall auf, dass durch Umgebungsgeräusche das akustische Signal verfälscht wird. Es liegt also nahe, bei künst-

lichen Spracherkennern zusätzlich die visuellen Informationen einzubeziehen, um die Erkennungsraten zu erhöhen (s. Abb. 1.5).

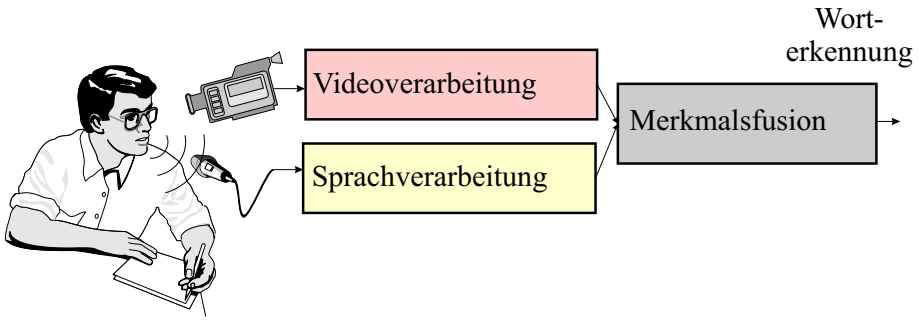


Abbildung 1.5: *Prinzip der Fusion von Audio- und Videodaten*

Einen der früheren Ansätze machte Ernie Nassimbene bei IBM im Jahr 1965. Mit den damaligen Mitteln beschränkte er sich auf die Information, ob die Zähne des Sprechers sichtbar sind oder nicht. Bei AT & T wurde diese Idee 1984 von Eric Petajan [Pet84] wieder aufgegriffen. Zu diesem Zeitpunkt stand schon eine wesentlich leistungsfähigere Hardware zur Verfügung. Er berechnete für jedes Bild Höhe, Breite, Umfang und Fläche der Lippenkontur. Damit erreichte er mittels eines Klassifikators, der nach dem Prinzip des „Dynamic Time Warping“ arbeitete, bei einem Buchstabierproblem eine Verbesserung der Erkennungsleistung um 2 Prozent. Neuere Ansätze arbeiten mit Neuronalen Netzen oder Hidden Markov Modellen (HMM).
Betrachtet man die gesamte Aufgabe des Lippenlesens, lässt sie sich in die folgenden Abschnitte unterteilen:

Schnelle Lokalisierung des Sprechers: Das Gesicht der Sprecher muss gefunden werden, erst dann lässt sich die Videokamera auf sein Gesicht einzoomen, so dass sein Gesicht in ausreichender Qualität aufgenommen werden kann.

Mundlokalisation: Nach der Gesichtslokalisation hat man einen groben Anhaltspunkt, wo man nach der Mundregion des Sprechers suchen kann. Die Mundregion muss in diesem Schritt genau bestimmt werden.

Modellierung der Lippen: Die Lippenkontur ist die Begrenzungslinie zwischen Ober- bzw. Unterlippe und der Gesichtshaut. Diese Konturen müssen modelliert werden. Die Parameter des Modells werden dann direkt oder nach einer Transformation als Merkmalsvektor für den nächsten Schritt verwendet.

Klassifikation: Diese Merkmale werden dann einem Erkenner zugeführt, der mit ihnen allein oder zusammen mit den Merkmalen, die aus dem akustischen Signal gewonnen wurden, eine Entscheidung für eine Klasse fällt.

Zur Zeit arbeiten weltweit viele Forschungsgruppen an diesem Problem, jedoch unterscheiden sich die Randbedingungen. Einige „Glieder“ der oben erläuterten Kette sind manchmal nicht implementiert. Tabelle 1.1 (Legende der Tabelle: 1.2) führt einige Beispiele der Arbeiten auf diesem Gebiet auf.

System	Gesichts- [1] suche	Mund- [2] suche	Merkmals- [3] extraktion	Erkennungs- [4] algorithmus
[Pet84]	-	Nase	Schwelle	Abstand
[Fin86]	-	-	Markierung	Abstand
[Sej90]	-	-	direkt	NN
[Pra93]	-	-	Markierung	TDNN
[Gol93]	-	-	Schwelle	HMM
[Sil93]	-	-	VQ	HMM
[Bre95]	-	-	Fläche	NN-HMM
[Cos95]	-	-	Markierung	RNN
[Lue97]	-	*template matching*	Form- & Intensitätsmodelle	HMM
[Mov97]	-	-	direkt	HMM
[Hen97]	Farbe	Farbe	*templates*	HMM
[Mei99]	Farbmodell	Nase, Pupille	direkt, LDA, PCA	MS-TDNN

Tabelle 1.1: *Entwicklung bei der visuellen Spracherkennung*

[1]Gesichtssuche	-	Bewegung des Sprechers nicht erlaubt
	Farbe	Farbmodell für die Haut.
[2]Mundsuche	Kante	Kantendetektion im Graubild
[3]Merkmals-extraktion	Schwelle	Schwelle im Graubild
	Markierung	spiegelnde Markierung am Gesicht
	VQ	Vektorquantizierung
	PCA	Haupt-Komponenten-Analyse
	LDA	*Linear Discriminant Analysis*
	templates	*deformable templates*
[4]Erkennungs-algorithmus	NN	Neuronale Netze
	RNN	*Recurrent NN*
	MS-TDNN	*Multiple State Time Delay NN*
	HMM	*Hidden Markov Model*

Tabelle 1.2: *Legende für Tabelle 1.1*

Es fällt auf, dass bei den früheren Arbeiten vielmehr die Evaluierung dieses Konzepts im Vordergrund stand. Es wurden beinahe ideale Bedingungen bezüglich der Beleuchtung, Kamerabildqualität und des Make-ups der Lippen vorausgesetzt. Es handelte sich meistens um eine bescheidene Anzahl von Schwarzweißbildern, die alle genau die Mundregion umfassten. Für die Merkmalsextraktion wurden meist klassische mathematische Modelle verwendet. Man hat mit den Graupixelwerten des Bildes bzw. deren Transformation (z.B. *Fast Fourier Transformation, FFT, Principal Component Analysis, PCA*) gearbeitet. Als Klassifikator wurden einfache Abstandsmetriken verwendet. In [Pra93] hat man sich langsam von den ganz mühsam vorbereiteten Bildern der Mundregion gelöst, indem man spezielle spiegelnde Markierungen an den Gesichtern angebracht hatte, die sich dann automatisch im Bild lokalisieren ließen.
Erst in den letzten Jahren wurden Begriffe wie „Freiheit der Bewegung des Kopfes“ und „variierende Lichtverhältnisse“ unterstrichen. Die zu erkennenden Einheiten sind entweder statischer Natur (Einzelbilder) oder dynamischer Natur (ganze Wörter, Sätze).
In dieser Arbeit wird von einer *real-life*-Situation ausgegangen. Das vorgestellte System eignet sich für übliche Rechnerarbeitsplätze, öffentliche Bedienungsterminals oder moderne Übertragungsterminals. Es wurde daher bewusst auf Hilfsmittel wie Markierungen am Gesicht, Bemalen der Lippen oder Spezialaufnahmesysteme verzichtet. Für die Gesichtssuche, Merkmalsextraktion und Klassifikation werden Algorithmen eingesetzt, die nicht rechenintensiv sind; dies ermöglicht eine Echtzeitrealisierung.

1.3.5 Vergleich der Anwendungen

In den Abschnitten 1.3.1 bis 1.3.4 wurden einige Anwendungen aufgeführt, für die ein Lippenleser notwendig ist. Da der Lippenleser ein wesentlicher Teil dieser Arbeit ist, ist es von großer Bedeutung, die unterschiedlichen Anforderungen an den Lippenleser bei den verschiedenen Anwendungen zu untersuchen. Tabelle 1.3 fasst die wichtigsten Unterschiede zusammen.

Anwendung	Echtzeit	A-V Zugehörigkeit	Gütemaß	Labor Bed.
(1.) Synchronisation von Audio und Video	nein	ja	objektiv	ja
(2.) Abbildung von Lippenbewegungen	nein	ja	objektiv	ja
(3.) Sprachverbesserung	ja	ja	objektiv	nein
(4.) visuelle Spracherkennung	ja	nein	Erkennungsrate	nein

Tabelle 1.3: *Vergleich der Anforderungen bei den verschiedenen Anwendungen*

Während nach Tabelle 1.3 bei den Anwendungen (1.) und (2.) auf eine Echtzeit-

Verarbeitung verzichtet werden kann, ist dies bei der Sprachverbesserung (3.) und der visuellen Spracherkennung (4.) nicht der Fall. Ein weiterer Gesichtspunkt ist die Notwendigkeit, eine Zugehörigkeit zwischen den Audio- und den Videomerkmalen zu finden. Die visuelle Spracherkennung ist das einzige Anwendungsgebiet, das ohne Kenntnis dieser Zuordnung auskommen kann. Es ist auch das einzige Gebiet, wo man die verschiedenen Algorithmen untereinander subjektiv vergleichen kann; dabei gilt die Worterkennungsrate (WER) als Maß für die Güte eines Algorithmus. Während sowohl die Synchronisation von Audio- und Videodaten (1.) als auch die Erzeugung von animierten Bildsequenzen (2.) in der Regel in speziell ausgestatteten Studios oder im Labor vorgenommen werden, ist dies bei den anderen Anwendungen nicht der Fall. Man kann also für die visuelle Spracherkennung zusammenfassen: Die Erkennung muss echtzeitfähig sein und unter normalen Umgebungen arbeiten können. Maßgebend für die Güte einer Erkennung ist die Worterkennungsrate.

1.4 Aufgabenstellung

In der vorliegenden Arbeit wird ein komplettes System zur Spracherkennung dargestellt. Das System wertet sowohl die akustischen als auch die visuellen Informationen des Sprechers aus. Folgende Anforderungen waren dabei zu beachten:

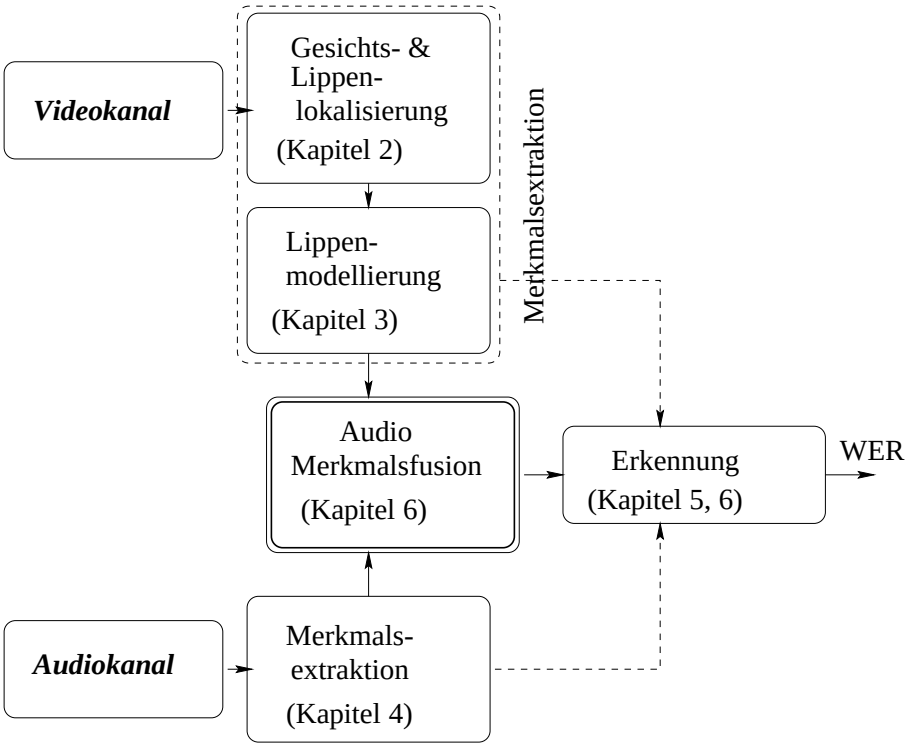


Abbildung 1.6: *Hauptteile des gesamten Systems des Erkenners*

Echtzeitfähigkeit: die Merkmalsextraktion aus dem Videokanal ist sehr rechenintensiv. Es müssen neue Verarbeitungsmethoden gesucht werden, mit denen eine

Echtzeitverarbeitung vorgenommen werden kann.

Reale Umgebungen: da dieses System Anwendung in normaler Umgebung finden soll, sind ideale Lichtverhältnisse, spezielle Geräte und unpraktische Gesichts- und Lippenmarkierungen nicht vorauszusetzen.

Modularität und Erweiterbarkeit: die Schnittstellen zwischen allen Teilsystemen müssen klar definiert werden. Die einfache Austauschbarkeit eines Teilsystems muss gewährleistet sein.

Ein klassischer Spracherkenner besteht im wesentlichen aus 2 Hauptteilen: der Merkmalsextraktion und dem Klassifikator. Bei unserem System kommen wegen der Betrachtung des Videokanals noch zwei weitere Bestandteile hinzu, denn die Merkmalsextraktion muss natürlich auch für die visuellen Informationen durchgeführt werden (s. Abb. 1.5):

1. Der erste neue Teil muss ganz am Anfang der Kette stehen, nämlich der Gesichtslokalisierer. Die Gesichtslokalisierung samt Kamerasteuerung ist in Kapitel 2 dargestellt. Die Merkmalsextraktion und die Klassifikation werden in den darauffolgenden Kapiteln vorgestellt.

2. Der zweite neue Teil, die Fusion der beiden Kanäle, wird in Kapitel 6 vorgestellt.

Abb. 1.6 ordnet den wesentlichen Teilen des Systems die entsprechenden Kapitel der Arbeit zu.

Kapitel 2

Robuste und schnelle Lippenregionsuche

Jedes Erkennungssystem besteht im wesentlichen aus zwei Modulen: einem Merkmalsextraktionsmodul und einem Klassifikator. Die Aufgabe des Merkmalsextrahierers besteht darin, einen '*geeigneten*' Merkmalsraum zu definieren, mit dem eine möglichst korrekte Trennung zwischen den verschiedenen Klassen erfolgen kann. Die Aufgabe der Trennung übernimmt der Klassifikator. '*Geeignet*' in dem Sinne bedeutet, dass Objekte unterschiedlicher Klasse möglichst fern voneinander, Objekte gleicher Klasse jedoch möglichst nah aneinander im Merkmalsraum platziert werden (engl.: *clustering*). Wenn der Merkmalsextrahierer nicht in der Lage ist, den Merkmalsraum so zu definieren, dass es zu keinerlei Überscheidungen der Klassenregionen kommt, dann kann auch ein idealer Klassifikator [1] keine Zuordnungsrate von 100% erreichen. Daher ist es für eine gute Erkennungsrate sehr wichtig, von vorne herein für eine gute Trennbarkeit zu sorgen. Dabei stellt die genaue Bestimmung der Lippenregion eine große Herausforderung dar. Für den Einsatz im '*wirklichen Leben*' muss die Bestimmung der Lippenregion auch in Echtzeit und unabhängig von der existierenden Beleuchtung und den Hintergrundszenen erfolgen.

In diesem Kapitel werden zunächst im Abschnitt 2.1 verschiedene Ansätze zur Lippenlokalisierung vorgestellt. Einige dieser Ansätze sind allgemein einsetzbar für die Suche nach beliebigen Objekten (nicht nur für Gesichter oder Lippen), andere sind speziell für dieses Problem konzipiert. Alle der hier vorgestellten Ansätze sind implementiert und genauer untersucht worden [Sei96], [Qui97], [Mar96], [Kor97]. Es ist zu erwarten, dass allgemeine Ansätze nicht besser abschneiden können als solche, die für dieses Problem maßgeschneidert sind.

Im Abschnitt 2.2 wird das tatsächlich eingesetzte Lokalisierungssystem vorgestellt. Es handelt sich dabei um einen top-down-Entwurf, bei dem die gesamte Aufgabe in 3 kleinere aufeinanderfolgende Schritte unterteilt wurde.

[1] Ein idealer Klassifikator kann beliebige Trennflächen zwischen den Klassen definieren.

2.1 Ansätze für die Lippenlokalisierung

2.1.1 Template Matching

Ein klassisches Aufgabengebiet der Bildverarbeitung ist das so genannte *Template Matching*. Es wird verwendet, um bestimmte Objekte in Bildern zu suchen. Für die gesuchten Objekte müssen Informationen über Struktur, Form oder Farbe vorliegen. Diese Informationen stammen häufig aus einem Muster. Um nun das gesuchte Objekt in einem Bild zu finden, wird die Region im Bild, die die größte Ähnlichkeit mit dem Muster aufweist, als dem Muster ähnlich selektiert. Die Kunst dieses Verfahrens liegt bei der genauen Definition der Begriffe: Muster, Ähnlichkeit und Selektionsschwellen. Am einfachsten werden die Bildpunkte , die in Form ihrer Helligkeitswerte vorliegen, miteinander verglichen.
Ein Bild besteht aus mehreren Bildpunkten (Pixel), deren Koordinaten (x, y) sind. Mit W_I und H_I wird die Breite bzw. die Höhe des Bildes und mit W_T bzw. H_T die des Templates bezeichnet. Die Matrix $\mathbf{I}(x, y)$ ordnet jedem Punkt $\mathbf{p} = (x, y)$ einen Skalar zu, der ein Maß für dessen Helligkeit ist. Sie besitzt die Dimension $W_I \cdot H_I$. Die Helligkeitsmatrix des Templates soll mit $\mathbf{T}(x, y)$ bezeichnet werden und besitzt die Dimension $W_T \cdot H_T$. Die Größe des Templates ist geringer als die des Bildes; es soll ja in ihm enthalten sein. Für den Vergleich kann man z.B. die Summe absoluter Differenzen (SAD) verwenden:

$$SAD(x, y) = \frac{1}{W_T H_T} \sum_{i=1}^{W_T} \sum_{j=1}^{H_T} |\mathbf{T}(i, j) - \mathbf{I}(i + x, j + y)| \qquad (2.1)$$
$$W_I > W_T, H_I > H_T$$

Es wird für jede mögliche Verschiebung x und y in Richtung der Koordinatenachsen die Summe über den Betrag der Helligkeitsdifferenz je Pixel gebildet. Diese Funktion hat folglich dort ein Minimum, wo sich Muster und Bild am ähnlichsten sind. Alternativ zur SAD kann auch die Kreuzkorrelationsfunktion (KKF) verwendet werden.

Häufig handelt es sich nicht nur um ein einziges Muster, sondern es sind vielmehr N verschiedene Muster, die sich in der Form ähnlich sind und die zusammen eine einzige Klasse bilden. Man müsste also Gl. 2.1 insgesamt N-mal durchlaufen.
Da aber in der Praxis diese Muster nicht direkt aus dem zu untersuchenden Bild stammen, unterscheiden sich diese Muster immer von dem Bildausschnitt, in dem das Objekt enthalten ist. Es kommt auch desöfteren vor, dass die Objekte im Bild verdreht, skaliert oder optisch verzerrt vorliegen. Würde man für jede mögliche Skalierung bzw. Verdrehung ein eigenes Muster ablegen, wäre die Anzahl der zu berechnenden SAD-Funktionen enorm hoch. Die gesamte Rechenzeit steigt dadurch quadratisch an.

Die variierenden Beleuchtungsverhältnisse und die Störungen im Bild machen diese Aufgabe nicht gerade leicht, selbst wenn es sich um sehr einfache Objekte handelt. Abhilfen, die man sich mit Histogramm-Normalisierung (um sich an die verschiedenen Lichtverhältnisse zu adaptieren) oder mit Gauß-Pyramiden (für eine schnellere Berechnung) zu verschaffen erhofft, halten sich in Grenzen [Sei96]. Daher wird das

Template Matching in der Regel auf Bilder mit einfachen Objekten und einem einfarbigen Hintergrund angewandt.
Für die Lokalisierung von Gesichtern oder von Gesichtsteilen wie Nase, Mund, usw. scheidet dieses Verfahren aus folgenden Gründen aus:

- die Form und Farbe der verschiedenen Gesichter bzw. Gesichtsteile variieren sehr stark von Mensch zu Mensch,
- bei realen Anwendungen ist die relative Gesichtsgröße und Orientierung im Bild nicht im voraus bekannt,
- das Verfahren ist sehr rechenintensiv,
- die Hintergrundszenen sind nicht einfarbig und
- die Lichtverhältnisse sind nicht bekannt.

2.1.2 Lokale Symmetrieachsen

Einen anderen Weg beschritt die Arbeit von [Mar96]. Sie stützt sich auf die Tatsache, dass Gesichter stets eine Symmetrie zwischen linker und rechter Gesichtshälfte aufweisen. Es sind Symmetrien einzelner Bereiche, wie des Mundes oder der Augen, vorhanden. Bereiche, bei denen eine solche Symmetrie vorliegt, müssen anschließend noch genauer betrachtet werden, denn schließlich kann es auch durchaus noch andere Objekte im Bild geben, die eine solche Symmetrieachse besitzen. Füllt das Gesicht das Bild aus, dann könnte eine solche Methode direkt zur Lokalisierung von Augen, Nase und Mund benutzt werden. Das Verfahren ist jedoch sehr schwerfällig im Falle von leicht zur Seite geneigten Gesichtern. Die Tatsache, dass dieses Verfahren sehr rechenintensiv ist, ließ es ausscheiden.

2.1.3 Bilddifferenzen

Eine einfache Möglichkeit, um das Gesicht zu suchen, ist die Bildung der Differenz zum Hintergrund. Mit der Lokalisierung des Gesichts liegt bereits eine erste Schätzung für die Lippenregion vor. Dies ist eine allgemeine Methode, mit der man sich bewegende Objekte in einer Bildersequenz finden kann. Dazu muss die Kamera auch bei der Abwesenheit des Sprechers die Bilder aufnehmen. Jedes Bild $\mathbf{I}(x, y)$ wird punktweise von dem vorherigen subtrahiert. Dadurch ist es möglich, bei einem Differenzensprung, den ersten Auftritt des Sprechers im Bildfeld zu detektieren.

$$\mathbf{D}(x, y) = \mathbf{I}^{(i)}(x, y) - \mathbf{I}^{(i-1)}(x, y) \tag{2.2}$$

Das letzte Bild vor diesem Sprung wird dann gespeichert und gilt als Referenz für das Hintergrundbild. Mit geschickten Adaptionsverfahren können die Teile des Hintergrundes, die nicht vom Sprecher verdeckt sind, dauernd aktualisiert werden. Die Region des Sprechers kann durch die Differenzbildung zwischen dem aktuellen Bild und dem gespeicherten Hintergrundbild gefunden werden.

Leider kann mit dieser Methode der Sprecher als bewegendes Objekt nur als Ganzes lokalisiert werden, d.h., es wird nicht nur sein Gesicht bestimmt, sondern der gesamte Teil seines Körpers, der von der Kamera erfasst wird. Die genauen Konturen seines Gesichtes bleiben daher unbestimmt. Außerdem würde das Verfahren scheitern, falls es sich um keinen statischen Hintergrund handelt oder falls es sich um eine bewegende (dynamische) Videokamera, wie es bei uns der Fall ist, handelt.

2.2 Das Lokalisierungssystem

Die Anforderungen an das Lokalisierungssystem der Lippen machen es unmöglich, ein Verfahren zu finden, mit dem man alle Probleme auf einen Schlag lösen kann.

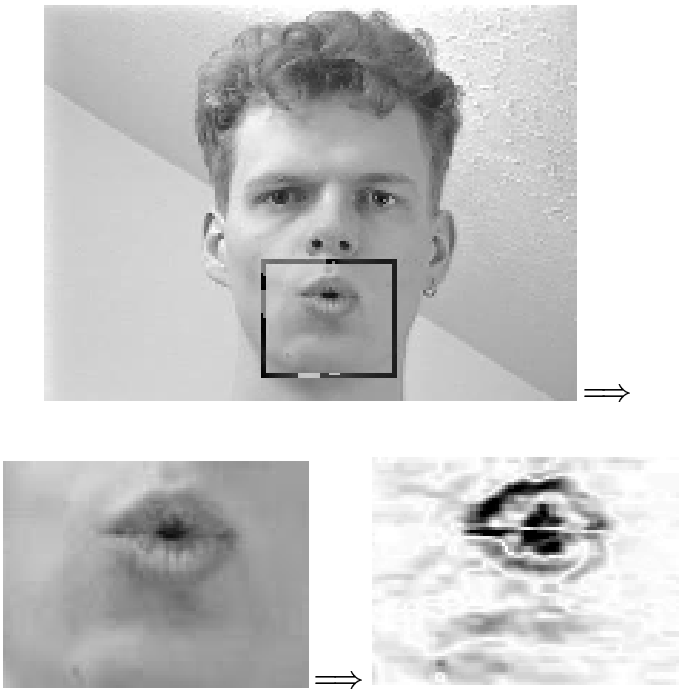


Abbildung 2.1: *Die Hierarchie bei der Lippenmodellierung*

Daher wurde das Problem hierarchisch in 3 Teilsysteme aufgeteilt (Abb. 2.1). Die Idee dabei ist, dass bei jeder Ebene die Lippenregion noch genauer als bei der vorangegangenen Ebene bestimmt wird.
Zunächst wird mit Hilfe eines Mikrofonarrays eine grobe Mundposition gefunden. Das Mikrofonarray besteht aus mindestens 4 Mikrofonen, die z.B. am Rand des Bildschirms befestigt werden können. Dabei werden nur die Sprachsignale des Sprechers (also ohne jegliche Auswertung des Bildes) verarbeitet. Die Lokalisierung mit dem Mikrofonarray kann als eine Initialisierungsphase für das darauffolgende Teilsystem betrachtet werden. Sie liefert eine grobe, aber sehr schnelle Schätzung für die Position des Mundes im Raum (s. Abschnitt 2.2.1).
Mit einem geeigneten Transformationsmodell kann die 3D-Schätzung der Position im Raum in Kamerakoordinaten transformiert werden. Die so erfolgte Initialisierung hat

zwei Vorteile:

- Die dafür benötigte Sprachverarbeitung erfordert wenig Rechenaufwand.
- Im Gegensatz zu einer Initialisierung, die nur auf Bildverarbeitungsalgorithmen aufbaut, kann hier das Gesicht auch dann gefunden werden, wenn der Sprecher nicht im Blickfeld der Kamera ist.

Mit dieser Initialisierung kann die Videokamera so gesteuert werden, dass sich der Sprecherkopf nun in ausreichender Auflösung etwa in der Mitte des Bildes befindet.
Es ist möglich, auf das Mikrofonarray zu verzichten. In diesem Fall ist das System nicht in der Lage, die Kamera während der Initialisierungsphase in Richtung Sprecher zu schwenken, falls sein Gesicht nicht im Blickfeld der Kamera ist. Die Suche nach dem Gesicht erfolgt in diesem Fall mit reinen Bildverarbeitungsroutinen (s. Abschnitt 2.2.2), wodurch sich die Initialisierungszeit verlängert.
Nach diesem ersten Schritt würde eine direkte Lippensuche im Bild sehr rechenintensiv sein, daher wird zunächst mit Hilfe des zweiten Teilsystems das Sprechergesicht gesucht. Dank des im Abschnitt 2.2.4 vorgestellten Farbmodells können die Gesichtskonturen schnell und genau gefunden werden. Im letzten Schritt wird die Lippenregion genau bestimmt (Abschnitt 2.2.5). Diese 3 Teile werden nun in den folgenden Abschnitten detaillierter erläutert.

2.2.1 Lokalisierung mit dem Mikrofonarray

Das Sprachsignal wird mit mehreren Mikrofonen gleichzeitig aufgenommen. Im allgemeinen erreicht die akustische Welle die verschiedenen Mikrofone zu unterschiedlichen Zeiten, die von der Entfernung des Mundes zum jeweiligen Mikrofon abhängen. Sind die Zeitverzögerungen zwischen den Mikrofonsignalen bekannt, so kann in Abhängigkeit der geometrischen Mikrofonanordnung die Position des Sprechers im Raum exakt berechnet werden. Abbildung 2.2 zeigt ein Beispiel für eine Mikrofonanordnung, bei der die Mikrofone an den Kanten des Bildschirms angebracht sind.

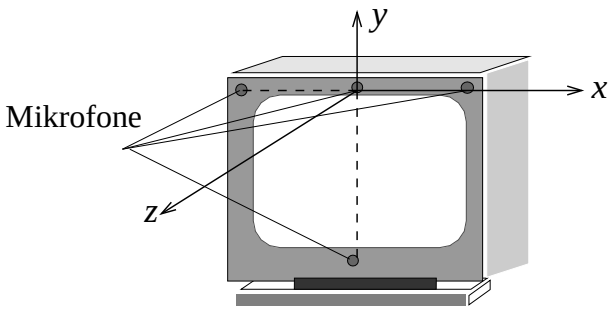


Abbildung 2.2: *Beispiel für ein Mikrofonarray mit einer T-Konfiguration*

Mathematisch gesehen benötigt man genau 4 Mikrofone (1 Mikrofon davon als Referenz), um die Position des Mundes im 3-dimensionalen Raum bestimmen zu können. Leider stehen die Zeitverzögerungen zwischen den Mikrofonsignalen nicht direkt zur

Verfügung, man muss sie mit Hilfe der empfangenen Signale schätzen. Nimmt man an, dass die empfangenen Signale bis auf eine Zeitverschiebung identisch sind und dass jedes Signal mit additivem Geräusch überlagert ist, so kann man die beiden Mikrofonsignale $x_i(t)$ und $x_j(t)$ mit 2.3 und 2.4 darstellen:

$$x_i(t) = s(t) + n_i(t) \tag{2.3}$$
$$x_j(t) = s(t + \tau_{ji}) + n_j(t), \tag{2.4}$$

wobei $s(t)$ und $s(t+\tau_{ji})$ die geräuschfreien Signale, $n_i(t)$ und $n_j(t)$ die Geräuschkomponenten sind. Für die Kreuzkorrelationsfunktion (KKF) zwischen $x_i(t)$ und $x_j(t)$ gilt unter der Annahme, dass die Geräusche beider Signale unkorreliert sind:

$$\begin{aligned} r_{x_i x_j}(\tau) &= E\left\{x_i(t)\, x_j(t-\tau)\right\} \\ &= E\left\{(s(t) + n_i(t)) \cdot (s(t + \tau_{ji} - \tau) + n_j(t-\tau))\right\} \\ &= E\left\{s(t) s\ (t - (\tau - \tau_{ji}))\right\} \\ &= r_{ss}(\tau - \tau_{ji}) \end{aligned} \tag{2.5}$$

Die KKF zwischen den beiden Signalen ist also identisch mit der um $\tau - \tau_{ji}$ verzögerten Autokorrelationsfunktion (AKF) des geräuschfreien Signals. Da das Maximum der AKF bei $\tau = 0$ liegt, muss das Maximum der KKF bei τ_{ji} liegen. Die Vorgehensweise ist also einfach: man berechne die KKF zwischen jedem Signal und einem Referenzsignal (das Signal des Mikrofons M_0), suche nach dem Maximum der KKF und berechne anhand der Mikrofonanordnung und der geschätzten Zeitverschiebungen die Position des Sprechers im Raum. Das bedarf der Kenntnis einer Relation $\mathcal{R}$ zwischen den Zeitverzögerungen $\boldsymbol{\tau} = (\tau_{10}, \tau_{20}, \ldots, \tau_{i0}, \ldots, \tau_{N-1,0})^T$, den geometrischen Abständen des Mikrofonarrays $\mathbf{a} = (a_1, a_2, \ldots)^T$ und den Koordinaten der Position des Sprechermundes $\mathbf{p}$:

$$\mathcal{R}\{\boldsymbol{\tau}, \mathbf{a}, \mathbf{p}\} \tag{2.6}$$

Die Formulierung einer solchen Relation für eine beliebige Konfiguration der Mikrofone ist sehr aufwendig. Stattdessen werden einige realistische Konfigurationen betrachtet. Im folgenden soll zunächst die Rechteck-Konfiguration studiert werden, bei der die Mikrofone an den Ecken des Bildschirms angebracht sind.
Abbildung 2.3 zeigt eine solche Anordnung. Der horizontale Abstand der Mikrofone soll a, der vertikale soll b betragen. Der Zeitunterschied zwischen der Ankunft der akustischen Welle an dem Referenzmikrofon M_0 und der Ankunft der Welle am Mikrofon M_i beträgt τ_{i0} und ist proportional zum Abstand $\Delta l_{i0} = C_{Schall}\, \tau_{i0}$, wobei C_{Schall} die Verbreitungsgeschwindigkeit der Welle ist. In diesem Fall gilt für die Vektoren $\mathbf{a}$ und $\boldsymbol{\tau}$:

$$\mathbf{a} = \begin{pmatrix} a \\ b \end{pmatrix}\ ,\ \boldsymbol{\tau} = \begin{pmatrix} \tau_{10} \\ \tau_{20} \\ \tau_{30} \end{pmatrix} \tag{2.7}$$

Der Abstand zwischen der akustischen Quelle (dem Mund) $\mathbf{p}(x_p, y_p, z_p)$ und dem Referenz-Mikrofon M_0 soll mit l_0 bezeichnet werden (die genaue Herleitung ist im Anhang A zu finden).

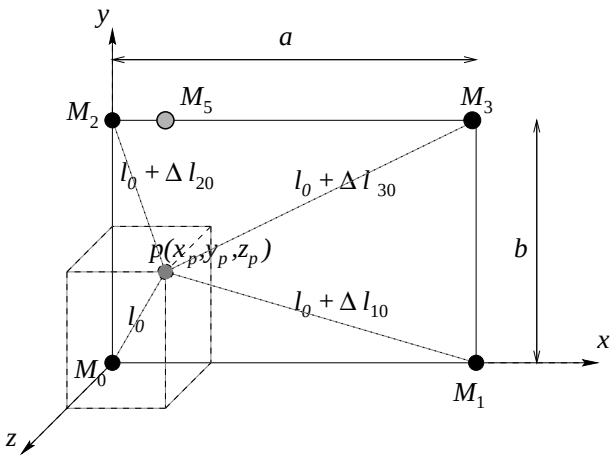


Abbildung 2.3: *Die Rechteck-Mikrofonkonfiguration*

Die 3 Koordinaten der geschätzten Position des Sprechers können dann mit

$$\begin{aligned} l_0 &= \sqrt{x_p^2 + y_p^2 + z_p^2} \\ &= \frac{\Delta l_{30}^2 - \Delta l_{20}^2 - \Delta l_{10}^2}{2\left(\Delta l_{10} + \Delta l_{20} - \Delta l_{30}\right)}, \end{aligned} \tag{2.8}$$

$$x_p = \frac{a^2 + 2l_0\left(\Delta l_{20} - \Delta l_{30}\right) - \Delta l_{30}^2 + \Delta l_{20}^2}{2a} \tag{2.9}$$

$$y_p = \frac{b^2 + 2l_0\left(\Delta l_{10} - \Delta l_{30}\right) - \Delta l_{30}^2 + \Delta l_{10}^2}{2b} \tag{2.10}$$

berechnet werden.

Die Gln. 2.8, 2.9, 2.10 stellen in diesem Fall die Relation $\mathcal{R}$ dar. Mit diesem Mikrofonarray kann die Position des Sprechers aus folgenden Gründen nur grob bestimmt werden:

- Die Schätzung der Zeitverschiebung weist einen großen Fehler auf. Da die Signale im Rechner alle digitalisiert vorliegen, ist die Auflösung der geschätzten Zeitverschiebung von der verwendeten Abtastfrequenz abhängig. Mit einer Abtastfrequenz von $f_A = 22.5$ kHz entspricht dies einer Positionsauflösung von $\frac{C_{Schall}}{f_A} \approx 1.5$ cm, bei einer Abtastfrequenz von $f_A = 8$ kHz sogar einer Auflösung von 4.25 cm.

- Die Geräusche sind nicht ganz unkorreliert wie angenommen wurde.

- Der Mund ist keine Punktquelle. Die Sprachorgane erstrecken sich vom Hals über den Gaumen hin bis zu den Lippen.

- Die Gleichungen 2.9 und 2.10 sind äußerst empfindlich gegenüber Fehlern bei der Bestimmung der Zeitverschiebungen. Die resultierenden Fehler bei der Koordinatenberechnung sind von der Position des Mundes abhängig und können sehr groß sein.

- Falls der Sprechermund an irgendeinem Punkt auf der Achse liegt, die senkrecht zur Arrayebene und genau durch den Mittelpunkt des Arrays verläuft, kann die z-Komponente des Punktes $\mathbf{p}$ nicht bestimmt werden. Dieses Problem hängt damit zusammen, dass alle Mikrofone auf einer Ebene liegen. In diesem Fall ist der Mund zu allen Mikrofonen gleich entfernt. Ferner sind die Lokalisierungsfehler sehr groß, wenn der Mund nahe dieser Achse liegt.

Um vor allem die letzten beiden Probleme (die Fehlerempfindlichkeit und das Problem der Bestimmung der z-Komponente) zu lösen, wurden auch andere Mikrofonarray-Konfigurationen untersucht, wie z.B. die T-Konfiguration [Leb96a] oder eine Kreuz-Konfiguration mit kleineren Abmessungen zwischen den Mikrofonen [Ber99].

Abbildung 2.4 zeigt die T-Konfiguration, deren Mikrofone auch am Bildschirmrand befestigt werden können. Bei dieser Konfiguration ist es komplizierter als im obigen Fall, eine geschlossene Lösung für $\mathcal{R}$ zu finden [Leb96a].

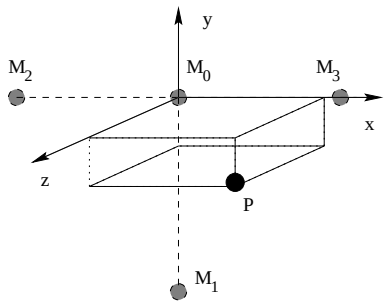


Abbildung 2.4: *T-Konfiguration der Mikrofone*

Die Fehlerempfindlichkeit einer solchen Struktur bleibt (obwohl besser als bei der Rechteck-Konfiguration) immer noch hoch. Dies ist anhand von Abb. 2.5 zu erkennen, wo eine Beispielposition des Mundes $\mathbf{p}(-30, -15, 50)$ dargestellt ist. Die schattierte Fläche stellt den Fall einer korrekten Bestimmung der Zeitverschiebung τ_{20} und τ_{30} dar. In diesem Fall ist der resultierende Fehler bei der Berechnung der x-Position gleich Null. Aus einer fehlerhaften Bestimmung der Zeitverschiebungen τ_{20} und τ_{30} resultiert aber ein Fehler für die Berechnung der x-Position, der überproportional mit den τ-Fehlern steigt.
Um diesen Fehler in Grenzen zu halten, wurden zwei Ansätze verfolgt, die hier nur kurz zusammengefasst werden. Für die genaue Beschreibung dieser Arbeit wird auf

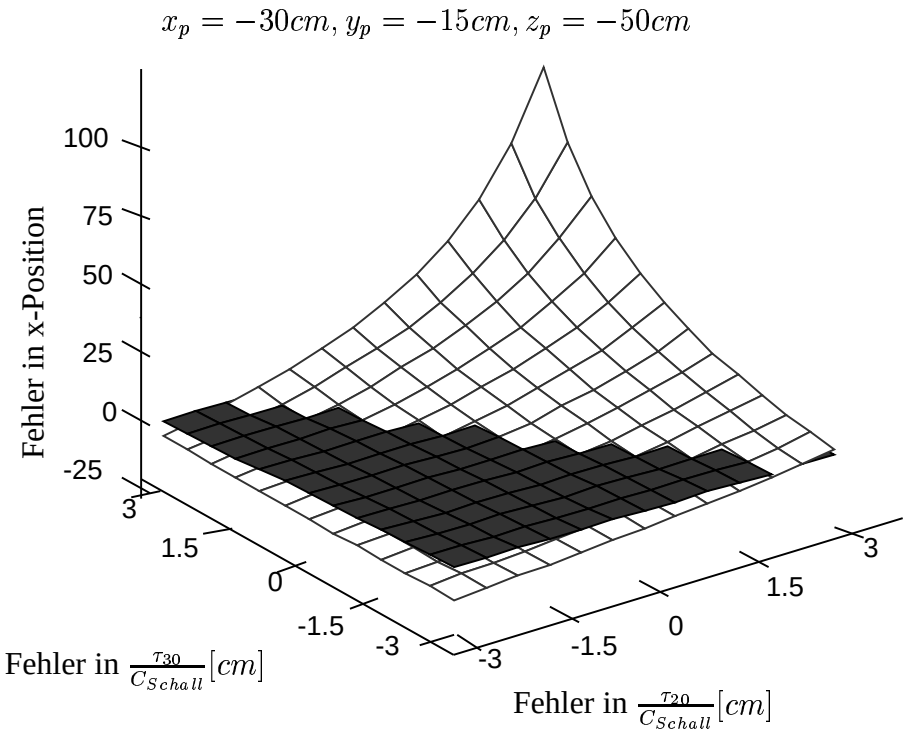


Abbildung 2.5: *Fehler in der Bestimmung der x-Position*

[Leb96a] und [Leb96b] verwiesen.
Beim ersten Ansatz wird $\mathcal{R}$ durch eine *look-up*-Tabelle ersetzt. Der gesamte Raum, in dem sich der Sprechermund befinden darf, wird vorgegeben und weiterhin in einer begrenzten Anzahl von Positionen quantisiert. Für jede dieser Positionen existiert ein Eintrag in der Tabelle mit den entsprechenden Zeitverzögerungen. Die Tabelle kann somit in eine Richtung bei einer Lernphase geschrieben werden und kann in die andere Richtung beim Bestimmen der Position gelesen werden.
Beim zweiten Ansatz wird $\mathcal{R}$ durch eine lineare Approximation ersetzt:

$$\begin{aligned} x_p &= a_1 \cdot \tau_{20} + a_2 \cdot \tau_{30} \\ y_p &= a_3 \cdot \tau_{10} + a_4 \\ z_p &= a_5 \cdot \tau_{10} + a_6 \cdot \tau_{20} + a_7 \cdot \tau_{30} + a_8 \end{aligned} \tag{2.11}$$

Die Parameter a_1 bis a_8 können mit Hilfe eines Lerndatensatzes bestimmt werden. Der aus dieser Approximation resultierende Fehler wird in Abb. 2.6 dargestellt.
Eine weitere Konfiguration, die sich besser für das Montieren an der bewegenden Videokamera eignet, ist die X-Konfiguration. Sie unterscheidet sich von den obigen Konfigurationen dadurch, dass die Abstände zwischen den Mikrofonen viel geringer sind (10 bis 20cm).
Eine solche Konfiguration lässt sich nicht mehr am Rand des Bildschirms anbringen, sondern wird am sich bewegenden Teil der Kamera befestigt, was dazu führt, dass das gesamte Array zusammen mit der Kamera bewegt wird. Eine detaillierte Beschreibung dieser Arbeit ist [Ber99] zu entnehmen.

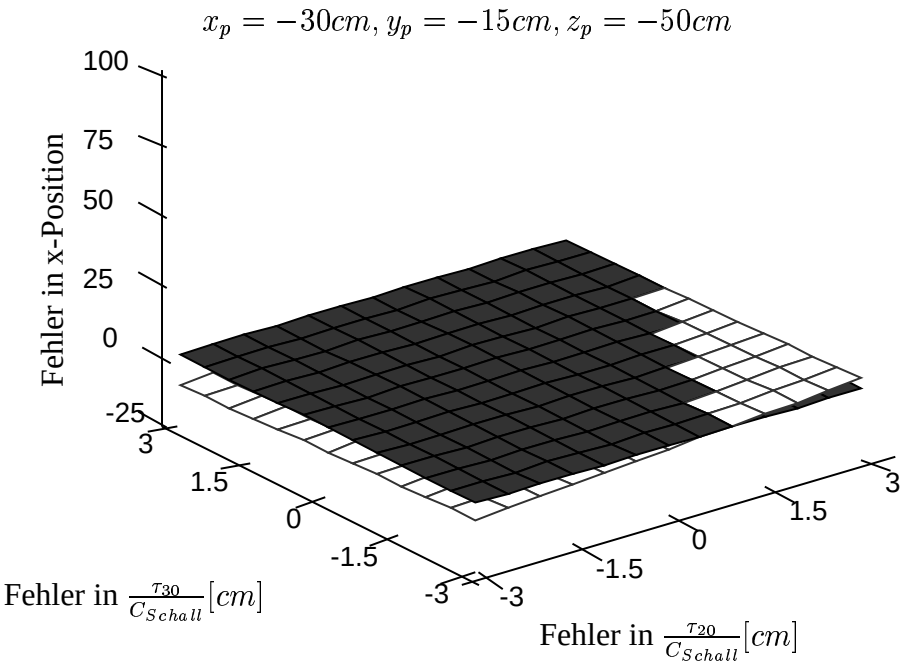


Abbildung 2.6: *Fehler in der x-Position mit linearer Approximation*

2.2.2 Gesichtslokalisierung mit einem Farbmodell

Im vorherigen Kapitel wurde gezeigt, dass das Mikrofonarray sich hervorragend für die Initialisierung der Gesichtssuche eignet. Ausgehend davon möchte man nun das Gesicht im Bild möglichst genau und schnell lokalisieren. Die hier verfolgte Idee basiert auf einer Analyse der Bildfarben.

Das menschliche Auge kann nur ein sehr schmales Spektrum der elektromagnetischen Wellen von ca. 400 bis 700 nm wahrnehmen. Das sichtbare Spektrum teilt sich grob in Violett (400-450 nm), Blau (450-500 nm), Grün (500-550 nm), Gelb (550-600 nm), Orange (600-650 nm) und Rot (650-700 nm) auf [Lev85].

Für die Farbdarstellung im Rechner kann man nur 3 Farben verwenden: Blau, Grün und Rot. Die Verhältnisse dieser drei Farbanteile untereinander ergeben den Farbeindruck. Beim Betrachten der Objekte sind neben dieser Farbzusammensetzung noch einige andere Punkte sehr wichtig, wie z.B. die Sättigung, die Helligkeit, die Intensität und der Farbton. Das menschliche Auge ist in der Lage, eine bestimmte Farbe selbst bei stark veränderten Lichtverhältnissen zu identifizieren. Dies ist nicht der Fall bei der digitalen Farbdarstellung im RGB-Raum, wo üblicherweise jedes Farbband (Rot, Grün und Blau) in 256 möglichen Werten quantisiert wird. Dabei wird z.B. eine bestimmte Farbe mit geringer Sättigung in diesem 3-dimensionalen Raum anders abgebildet wie bei einer höheren Sättigung der gleichen Farbe.

Die RGB-Darstellung stellt also keine geeignete Ausgangsbasis für die Farbanalyse dar. Es ist von großer Bedeutung, die menschliche Farbwahrnehmung näher zu betrachten. Nur so kann eine geeignete Farbdarstellung definiert werden, mit der Objekte durch ihre Farbe nach dem gleichen Prinzip wie beim Menschen bestimmt werden können. In Abschnitt 2.2.3 wird zunächst das biologische Farbmodell des Auges

sehr kurz zusammengefasst. In Abschnitt 2.2.4 wird dann ein dem menschlichen Auge nachempfundener Farbraum vorgestellt.

2.2.3 Die menschliche Farbwahrnehmung

Das Licht, das auf das menschliche Auge trifft, gelangt zur Netzhaut (Retina) durch die Hornhaut, die Öffnung der Iris, die Augenlinse und den retrolentalen Raum. Nur ca. 50% der Lichtenergie gelangen so zur Netzhaut, die die gesamte Innenfläche des Augapfels bis auf die Stelle, wo sich der Augennerv befindet, bedeckt [Lev85]. Die Menge des Lichts wird durch die Öffnung der Pupille reguliert. Um die Farbwahrnehmung beim Menschen zu verstehen, wird die Struktur der Netzhaut näher betrachtet. Sie besteht im wesentlichen aus 5 wichtigen Elementen: (1) die Stab- und Zapfen-Photorezeptoren, (2) die Horizontalzellen, (3) die Bipolarzellen, (4) die amakrinen Zellen und (5) die Ganglionzellen (Abb. 2.7).

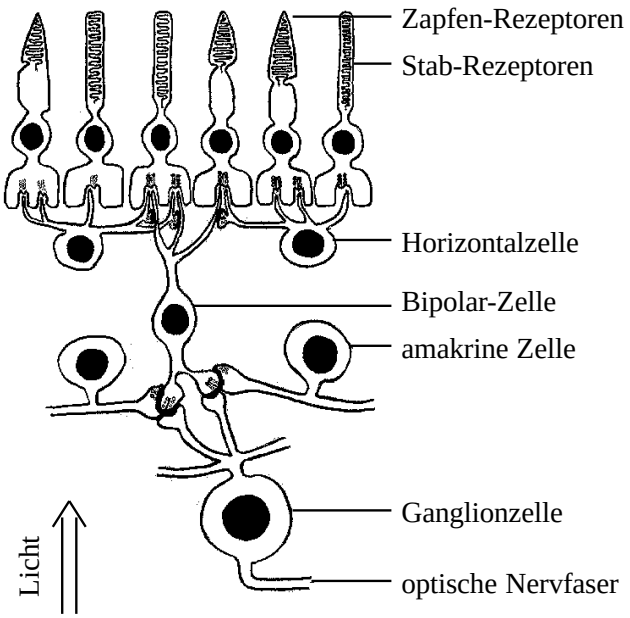
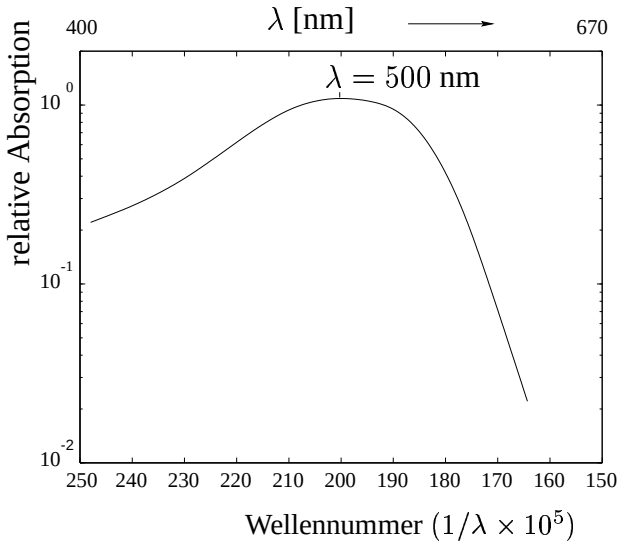


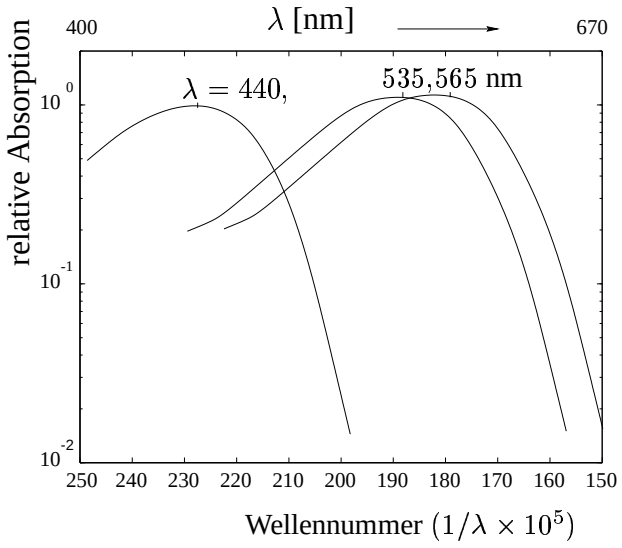
Abbildung 2.7: *Ein vereinfachtes Schemadiagramm der menschlichen Netzhaut (aus [Bla73])*

Das eintreffende Licht muss zunächst die relativ durchsichtige Sehnervfaser, die Blutgefäße und die verschiedenen Schichten der Zellen passieren, bevor es überhaupt zu den Empfangssensoren des Auges gelangt. Das Licht wird zusätzlich durch die Lage der Stab- und Zapfensensoren, deren lichtempfindliche Seite in die entgegengesetze Richtung des Lichteintritts gerichtet ist, gedämpft.
Das aus diesen Zellen erzeugte elektrische Ausgangssignal wird mit den Axonen der Ganglionzellen zum Sehnerv weitergeleitet. Die so genannten visuellen Pigmente der Stab- und Zapfenzellen sind unterschiedlich. Das äußere Segment einer Stabzelle wird als Rhodopsin bezeichnet, ist rosafarben und verfärbt sich bei Lichteinfall zu weiß. Die spektrale Sensitivität des Rhodopsins ist in Abb. 2.8(a) dargestellt. Die Zapfenzellen

dagegen besitzen drei verschiedene Photopigmente, deren Sensitivität in Abb. 2.8(b) dargestellt ist.



(a)



(b)

Abbildung 2.8: *Das Absorptionsspektrum der (a) Stab-Rezeptoren und (b) der Zapfen-Rezeptoren, aus [Abr74]*

In der Region um die Augenachse befindet sich eine kleine Netzhautgrube (die Fovea). An dieser Stelle fehlt die Gehirnschicht, so dass die Lichtstrahlen ungehinderten Zutritt zu den Sinnesepithelzellen haben. Dort ist die Dichte der Zapfen-Rezeptoren am größten und nimmt zum Rand hin sehr stark ab. Die Stab-Rezeptoren sind dagegen außerhalb der Fovea stärker vertreten, nehmen aber ebenfalls mit dem axialen Winkel ab.

Man glaubt, dass die Stab-Rezeptoren mehr für ein monochromatisches Sehen *(engl.: „scotopic vision“)* zuständig sind, während für die Farb- und Detailwahrnehmung

mehr die Zapfen-Rezeptoren zuständig sind *(engl.: „photopic vision“).* Man beobachtet, dass die maximale Empfindlichkeit der Zapfen-Rezeptoren etwa den Farben Blau (B), grün (G) und Rot (R) entspricht (s. Abb. 2.8(b)).
Es bleibt bis heute immer noch umstritten, ob das menschliche Gehirn diese drei Farben direkt verarbeitet (Young-Helmholtz, neurophysiologisches Modell) oder ob der Mensch eher die Farben als 2 chromatische Komponenten (rot-grün und gelb-blau) und eine Luminanz-Komponente (Hering, 1875, psychophysikalisches Modell) empfindet. Es konnte aber auf jeden Fall durch Farb-Matching-Experimente eindeutig belegt werden, dass der Mensch Farben mit dem gleichen Energieverhältnis R:G:B, bis auf eine unterschiedlich empfundene Luminanz, gleich farbig wahrnimmt. Dies führt zur Definition des chromatischen Farbraums, der in dem folgenden Abschnitt näher betrachtet wird.

2.2.4 Der chromatische Farbraum

In [Schie95], [Swa91], [Qui97], [Hen97] wurde berichtet, dass sich der chromatische Farbraum für die Erkennung von Objekten sehr gut eignet. Der 3-dimensionale RGB-Raum wird nach 2.12 und 2.13 in den 2-dimensionalen chromatischen Farbraum (auch als rg-Raum bezeichnet) abgebildet.

$$r = \frac{R}{(R+G+B)} \tag{2.12}$$

$$g = \frac{G}{(R+G+B)} \tag{2.13}$$

Aus 2.12 und 2.13 ist ersichtlich, dass der neue Raum durch die Normalisierung mit der Summe $(R+G+B)$ intensitätsunabhängig wird, weiterhin wird die Komponente $b = B/(R+G+B)$ überflüssig, da $r+g+b = 1$ ist (s. Abb. 2.9).

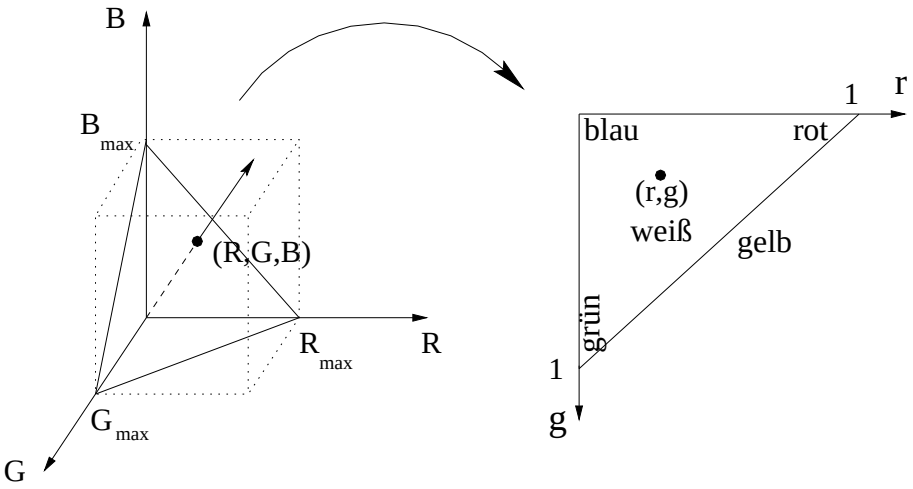


Abbildung 2.9: *Der chromatische Farbraum*

Untersuchungen der Farbverteilung (Histogramme) in diesem Farbraum ergaben, dass die menschliche Haut in einem engen Bereich liegt [Schie95]. Definiert man die Parameter einer Gaußverteilung so, dass sie die Verteilung von verschiedenen Hautfarben

umfasst, dann kann diese Gaußverteilung als Filter verwendet werden, um das Bild so zu filtern, dass nur die Farbanteile, die innerhalb eines bestimmten Bereiches dieser Gaußkurve sind, als hautzugehörig klassifiziert werden.
Die Gaußverteilung für den mehr-dimensionalen Fall ist durch

$$f_{\mathbf{x}}(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N \cdot |\mathbf{C_{xx}}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{C}_{\mathbf{xx}}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.14}$$

gegeben. Für unseren 2-dimensionalen Raum hat die Verteilung also genau 5 Parameter (da $\sigma_{rg} = \sigma_{gr}$): $\bar{r}, \bar{g}, \sigma_{rr}, \sigma_{rg}, \sigma_{gg}$

$$\mathbf{x} = \begin{bmatrix} r \\ g \end{bmatrix}, \boldsymbol{\mu} = \begin{bmatrix} \bar{r} \\ \bar{g} \end{bmatrix}, \tag{2.15}$$

$$\mathbf{C_{xx}} = \begin{bmatrix} \sigma_{rr} & \sigma_{rg} \\ \sigma_{gr} & \sigma_{gg} \end{bmatrix} \tag{2.16}$$

Diese Gaußverteilung wird im Laufe des Textes mit der '*globalen Verteilung*' bezeichnet. Um nun ein Gesicht an Hand seiner Farbverteilung im Bild bestimmen zu können, werden zwei Schritte benötigt. Der erste Schritt (Analyse) gilt als Lernphase und wird nur einmal benötigt. In diesem Schritt werden viele verschiedene Hautfarben analysiert, um die Gaußparameter der globalen Verteilung, die im zweiten Schritt benötigt wird, bestimmen zu können:

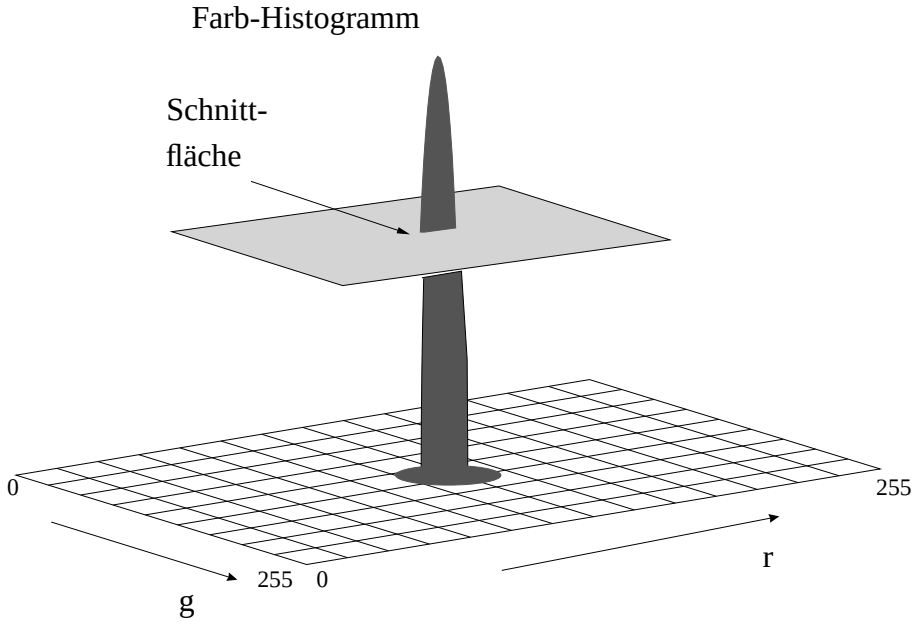


Abbildung 2.10: *Farbverteilung der menschlichen Haut im rg-Farbraum*

1. **Analyse:** Es wird die Hautverteilung von N Personen analysiert. Die Hautverteilung einer bestimmten Person i kann auch wiederum durch eine Gaußverteilung approximiert werden $\bar{r}^{(i)}, \bar{g}^{(i)}, \sigma_{rr}^{(i)}, \sigma_{rg}^{(i)}, \sigma_{gr}^{(i)}, \sigma_{gg}^{(i)}$ (im Laufe des Textes mit der '*spezifischen Verteilung*' bezeichnet).

   Für die Bestimmung der Mittelwerte $\bar{r}$ und $\bar{g}$ der globalen Verteilung werden die Mittelwerte der spezifischen Verteilungen gemittelt:

$$\bar{r} = \sum_{i=0}^{N} \bar{r}^{(i)} \tag{2.17}$$

$$\bar{g} = \sum_{i=0}^{N} \bar{g}^{(i)} \tag{2.18}$$

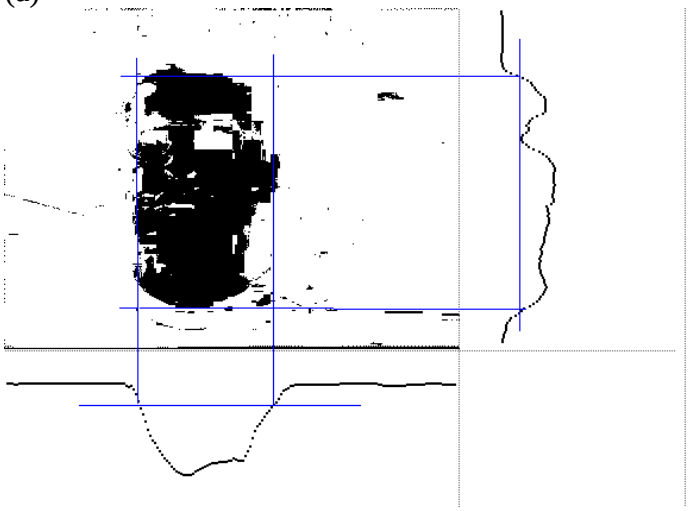
   Die Standardabweichungen der globalen Verteilung werden so gewählt, dass die globale Gauß-Kurve alle anderen spezifischen Gauß-Kurven umfasst. Mit anderen Worten: alle spezifischen Gauß-Kurven dürfen nur innerhalb der Kurve in Abb. 2.10 verlaufen.

Abbildung 2.11: *(a) Originalbild, (b) mit Gauß-Filterung bei 20%, (c) 45%, (d) 60% vom Maximum*

2. **Filterung:** Nachdem uns ein Vorwissen über die Verteilung der menschlichen Haut in Form von Parametern der globalen Verteilung vorliegt, kann nun ein neues Bild mit dieser Gauß-Verteilung folgendermaßen gefiltert werden: Die Verteilungskurve wird mit einer vorgegebenen Ebene, die senkrecht zur rg-Ebene verläuft, geschnitten. Die Schnittfläche stellt dann eine Begrenzung für die erlaubten rg-Werte dar. Es werden alle Bildpunkte eines Bildes kontrolliert: falls die Punktwerte innerhalb dieser Fläche liegen, dann werden sie als gesichtszugehörig klassifiziert. Alle anderen Bildpunkte gelten dann als nicht gesichtszugehörig. Es handelt sich hier also um einen Binärklassifikator, der auf der Ebene der Bildpunkte arbeitet. Abb. 2.11 zeigt ein Beispiel für eine solche Filterung bei unterschiedlicher Positionierung der Schnittflächen: 20%, 45% und 60% vom Maximum der Gaußkurve.

2.2.5 Grobe Bestimmung der Lippenregion

(a)



(b)

Abbildung 2.12: *(a) das Originalbild, (b) Projektion nach der Gauß-Filterung*

Wie in Abb. 2.11 zu erkennen ist, bildet das Gesicht nach der Filterung das größte zusammenhängende ‘Cluster’. Um das Gesicht in dem gefilterten Bild auf einfache und schnelle Weise bestimmen zu können, wird zunächst das Bild binär dargestellt:

alle Bildpunkte sind entweder ‘1’ (schwarz), falls sie als gesichtszugehörig klassifiziert wurden, sonst ‘0’ (weiß).
Mit Hilfe einer horizontalen und vertikalen Projektion können die Grenzen des Gesichts bestimmt werden (s. Abb. 2.12): alle Bildpunkte einer Zeile, die ‘1’ sind, werden aufaddiert. Das Ergebnis wird in den Vektor $\mathbf{v}$ geschrieben. Die Dimension dieses Vektor H = Anzahl der Zeilen im Bild. Die Elemente dieses Vektors nehmen an den Bildzeilen, in denen sich das Gesicht befindet, hohe Werte an.
Das erste Element des Vektors $\mathbf{v}$, das eine vordefinierte Schwelle überschreitet, begrenzt das Gesicht von oben, und das letzte Element, das diese Schwelle unterschreitet, begrenzt das Gesicht von unten.
Die rechte und linke Begrenzung verlaufen analog. Hier werden die Punkte spaltenweise aufaddiert. Auf diese Weise ist die Region des Mundes (ROI, *Region of Interest*) grob bekannt, da der Mund im unteren Bereich des Gesichtes liegt.

2.2.6 Beschleunigungsmaßnahmen

In diesem Abschnitt werden einige Konzepte vorgestellt, die dazu beigetragen haben, die Berechnungsroutinen für die Filterung und die Projektion zu beschleunigen.

Unterabtastung des Bildes

Da die Filterung und die Projektion pixelweise berechnet werden, ist die Rechenzeit direkt von der gesamten Anzahl der Bildpunkte abhängig. Theoretisch gesehen würde eine Halbierung der vertikalen und der horizontalen Auflösung dazu führen, dass die Rechenzeit auf nur $1/4$ reduziert wird. Mit einer Bildgröße von nur 192×144 Bildpunkten [2] konnte tatsächlich die Rechenzeit auf knapp $1/10$ reduziert werden (theoretisch wäre es $1/16$), ohne dass bei der Gesichtslokalisierung eine Qualitätseinbuße hingenommen werden musste.
Eine solche grobe Auflösung des Bildes beeinflusst leider die restlichen Teile des Systems (die genaue Lippenmodellierung und die Merkmalsextraktion) negativ. Es wurde daher ein Kompromiss gefunden: für die Filterung und die Projektion wird eine Bildgröße von 192×144 verwendet. Die so gefundene Mundregion wird in das hochauflösende Bild abgebildet, sodass die Lippenmodellierung mit der vollen Auflösung berechnet werden kann. Da die Mundregion ohnehin einen kleinen Bereich des Bildes ausmacht, bleibt die gesamte Anzahl der betrachteten Bildpunkte in der Mundregion trotz höherer Auflösung etwa gleich groß wie die des gesamten Bildes bei niedriger Auflösung.

Client-Server-Prinzip

Mit Hilfe des Mikrofonarrays konnte der Sprecher im Raum lokalisiert werden. Die Kamera muss dann in Richtung Sprecher schwenken. Da der Sprecher sich auch kontinuierlich bewegt, muss die Kamera gelegentlich nachgeführt werden. Die Steuerung der Kamera läuft über die serielle Schnittstelle (RS232). Die Kommunikation über die serielle Schnittstelle verläuft leider langsam. Es entstehen oft lange Reaktionszeiten,

[2] ein PAL-Signal hat eine Auflösung von 768×625 Bildpunkten

die man nicht im voraus erahnen kann. Schickt die Hauptanwendung einen Steuerungsbefehl an die Kamera, so muss die Hauptanwendung warten, bis die Kamera diesen Befehl ausführt und eine Rückmeldung an die Hauptanwendung sendet. Das führt dazu, dass die Hauptanwendung solange blockiert bleibt, bis die Kamera den Befehl ausgeführt hat; dies hat den Verlust von etlichen Bildern zur Folge.
Dieses Problem wurde mit Hilfe eines Client-Server-Prinzips gelöst. Die Kamerasteuerung wird von einem separaten Prozess (*Server*) abgewickelt. Die Hauptanwendung kommuniziert nun nicht mehr direkt mit der Kamera, sondern schickt die Befehle über das sehr schnelle Netz-Protokoll zum *Server* und wird dadurch nicht mehr blockiert. Es ist nun die Aufgabe des Servers, sich um die korrekte Ausführung der Kamerabefehle zu kümmern. Das Client-Server-Prinzip hat noch einen weiteren Vorteil: der Server kann auf einem anderen Prozessor oder sogar einem anderen Rechner im Netz laufen und somit wird der Prozessor, der sich um die Hauptanwendung kümmert, zusätzlich entlastet.

Shared Memory

Das X-Window-System unter UNIX läuft auch nach dem Client-Server-Prinzip. Alle Routinen zur Erstellung von Fenstern und Manipulationen von Bildern benutzen daher ein Netzwerkprotokoll. Mit Hilfe der *shared memory extension* kann aber der Speicher des Rechners, in dem die Anwendung läuft, direkt adressiert werden. In dem lokalen Speicher werden Adressbereiche für die Bilder reserviert, auf die die Anwendung direkt zugreifen kann. Damit erzielt man eine Beschleunigung der Bildmanipulation von bis zu 40%.

Regionenbegrenzung

Das PAL-Farbsystem hat eine Bildwiederholrate von 25 Bildern pro Sekunde. Selbst bei einem Sprecher, der seinen Kopf schnell bewegt, bleiben die Bildinhalte der aufeinanderfolgenden Bilder ähnlich. Man kann diese dynamische Ähnlichkeit ausnutzen, um die Gesichtslokalisierung zu beschleunigen.
In der Initialisierungsphase wird noch kein Gesicht gefunden und deshalb wird das Gesicht im gesamten Bild gesucht. Sobald das Gesicht mit einer gewissen Sicherheit gefunden ist, wird die Suchregion in den darauffolgenden Bildern nur auf einen kleineren Bereich um das bereits lokalisierte Gesicht herum beschränkt. Diese Suchregion muss so gewählt werden, dass selbst bei einer schnellen Bewegung des Kopfes jener beim nächsten Bild noch in dieser Suchregion liegt.
Die Beschleunigung hängt dabei direkt von der relativen Größe *(Suchregion/Bildfläche)* ab. Die Suchregion wird dann immer nach jedem Bild aktualisiert. Diese Maßnahme bringt aber Probleme mit sich, denn falls das Gesicht in einem Bild falsch lokalisiert wird, wird dementsprechend die Suchregion falsch bestimmt. Im schlimmsten Fall führt dies dazu, dass der „Faden“ verloren wird und dass die Gesichtssuche nur noch falsche Ergebnisse liefert. Auch eine falsche Wahl der Breite der Suchregion könnte dazu führen, dass die Suchregion immer kleiner wird. Es ist daher sehr wichtig, einen „Plausibilitätstest“ nach jeder Gesichtslokalisierung durchzuführen. Falls die

Gesichtslokalisierung ein „nicht plausibles“ Ergebnis liefert, muss die Suchregion neu auf das gesamte Bildgebiet initialisiert werden.
Dieser Test benutzt u.a. folgende Kriterien:

- Das Verhältnis a der gefundenen Gesichtsbreite zur Gesichtshöhe darf nur in einem bestimmten Bereich liegen ($0,2 < a < 1,0$).
- Die Differenz der ermittelten Gesichtsbreite zwischen zwei aufeinanderfolgenden Bildern darf einen vorgegebenen Wert nicht überschreiten.
- Der absolute Wert der ermittelten Breite B darf nur in einem vorgegebenen Bereich liegen ($0,1 \cdot Bildbreite < B < 0,9 \cdot Bildbreite$).

Die Werte der Schranken wurden anhand einer großen Datenbank heuristisch bestimmt.

Code-Optimierung

Der Quellcode wurde nach folgenden Gesichtpunkten bezüglich der Ausführungszeit optimiert:

- Alle Operatoren, die auf jeden Bildpunkt einwirken, sind in einem einzigen Operator zusammengefasst, sodass mit nur einer einzigen Schleife über alle Bildpunkte die Filterung, Begrenzung und Lippenmodellierung berechnet werden kann.
- Die Optimierungsoptionen vom Compiler waren so eingestellt, dass sich für die benutzte Systemarchitektur die schnellste Ausführungszeit ergibt.
- Multithread-Programmierung: Der Quellcode ist in zwei *threads* (dt.: Fäden) geschrieben. Dies erlaubt eine parallele Ausführung. Der eine „Faden“ ist zuständig für alle Sprachverarbeitungsroutinen, der andere kümmert sich um die Bildverarbeitungsroutinen. Eine Beschleunigung ist nur bei Multi-Prozessor-Architekturen zu erwarten.

2.3 Nachführung der Kamera

Wie bereits erwähnt, wird die Videokamera über die serielle Schnittstelle so gesteuert, dass das Gesicht des Sprechers immer in der Mitte des Bildes bleibt. Die verwendete Kamera (Canon VCC1) kann sowohl in die horizontale (engl.: *azimuth, pan*) als auch in die vertikale Richtung (engl.: *elevation, tilt*) geschwenkt werden. Dabei können für die Geschwindigkeit Werte zwischen 7,7 Grad/Minute bis 38 Grad/Minute eingestellt werden. Der Bereich des horizontalen Schwenkens ist ± 50 Grad, der des vertikalen ist ± 20 Grad.

Es handelt sich hier also um ein Regelungsproblem mit der geschätzten momentanen Position des Gesichts im Bild als Eingangsgröße und den Geschwindigkeiten der horizontalen und der vertikalen Bewegung als Einstellgrößen (s. Abb. 2.13).

Abbildung 2.13: *Regelungsproblem der Kameranachführung*

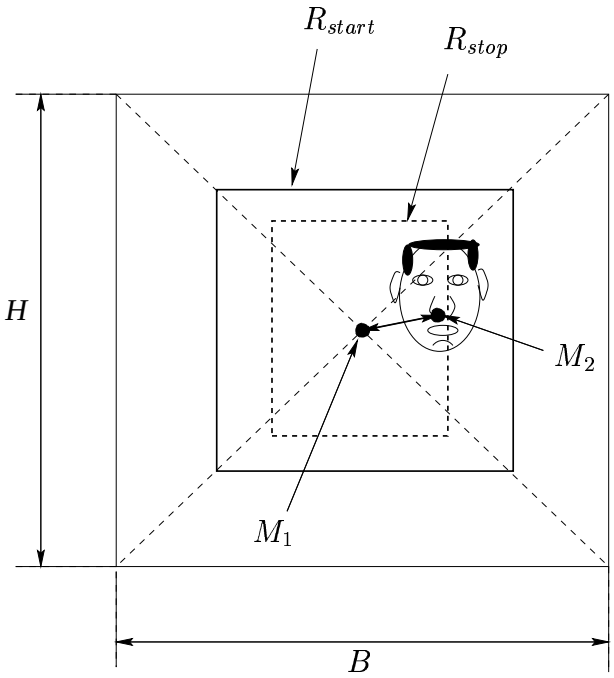


Abbildung 2.14: *Die Regionen R_{start} und R_{stop}*

In [Cro97] wurde für die Lösung dieses Problems eine PID-Steuerung vorgeschlagen. Hier soll aber auf eine solche verzichtet und statt dessen soll die Nachführung durch ein einfacheres Prinzip realisiert werden. Die Idee dabei ist, die Kamera nur dann in Bewegung zu setzen, wenn es nötig ist. Dabei werden zwei verschiedene Fälle unterschieden:

- Das Gesicht ist etwa in der Mitte des Bildes, es besteht kein Handlungsbedarf.
- Das Gesicht neigt dazu, das Bild zu „verlassen“, die Kamera muss in die richtige Richtung schwenken, bis das Gesicht in etwa in der Mitte des Bildes ist.

Dieses Prinzip ist also der Arbeitsweise eines Thermostats ähnlich.
Der Mittelpunkt $M_1(x_1, y_1)$ des Gesichts wird laufend in jedem Bild berechnet. Der Mittelpunkt $M_2(x_2, y_2)$ des Bildes ist konstant und errechnet sich aus der Höhe H und Breite B des Bildes zu:

$$M_2(x_2, y_2) = M_2(B/2, H/2) \quad . \tag{2.19}$$

Erfahrungsgemäß reicht die langsamste Geschwindigkeit der Kamera, um selbst

schnelle Bewegungen des Kopfes noch verfolgen zu können. Es wurde daher auf eine variierende Einstellung der Geschwindigkeit verzichtet.
Der Abstand zwischen M_1 und M_2 wird laufend in jedem Bild berechnet. Falls dieser Abstand eine vordefinierte Region R_{start} verlässt (s. Abb. 2.14), wird die Kamera in die entgegengesetzte Richtung in Bewegung gesetzt. Die Bewegung wird dann angehalten, sobald M_1 in der kleineren Region R_{stop} liegt.

2.4 Zusammenfassung

In diesem Kapitel wurde das gesamte Lokalisierungssystem der Lippen vorgestellt. Für die Initialisierungsphase dient ein Mikrofonarray, mit dem die Position des Gesichts grob, aber schnell bestimmt wird. Für die genauere Bestimmung der Gesichtsregion wird ein Farbmodell im chromatischen Farbraum eingesetzt. Durch vertikale und horizontale Projektion kann dann die Lippenregion gefunden werden. Das Problem der Kameranachführung wurde mit einem einfachen Einsatz gelöst. Für die schnelle Ausführung in Echtzeit wurden geeignete Maßnahmen getroffen.

Kapitel 3

Modellierung der Lippen

Wie in Kapitel 2 bereits erwähnt, wurde die Lippensuche in drei kleinere Teilaufgaben eingeteilt. Der letzte Teil, der in Abschnitt 2.2.5 beschrieben wurde, liefert eine grobe Bestimmung der Lippenregion. Diese Region stellt nun den Ausgangspunkt für die Modellierung der Lippen dar, die hier in diesem Kapitel vorgestellt wird. Grundsätzlich kann man die Methoden, die man für die Lippenmodellierung verwendet, in zwei Hauptgruppen einteilen. Diese werden in Abschnitt 3.1 vorgestellt.
In den Abschnitten 3.2 und 3.3 werden dann konkret zwei Modelle vorgestellt.

3.1 Arten der Lippenmodelle

Man kann die Lippenmodelle in zwei Hauptgruppen einteilen: **bildbezogene Modelle** und **geometrische Modelle**. Die bildbezogenen Modelle werden systematisch erstellt und basieren direkt auf den Bildpunkten bzw. deren Transformation. Der Nachteil solcher Modelle besteht darin, dass die Klassifikationsaufgabe erschwert wird.
Bei den geometrischen Modellen wird der Bildinhalt genau untersucht. Dabei werden bestimmte geometrische Eigenschaften des Gesichts, wie z.B Augenposition, Mundecken, Lippenbreite, Mundschwerpunkt oder Lippenkonturen, gesucht und mit einem geeigneten Merkmalsvektor beschrieben. Der Nachteil dieser Modelle liegt in der Schwierigkeit, diese Merkmale automatisch zu extrahieren.

3.1.1 Bildbezogene Modelle

Der einfachste Einsatz basiert direkt auf der Auswertung der Bildpunkte. Alle Bildpunkte der Lippenregion werden zunächst in Intensitätswerte umgewandelt ($I = \sqrt{R^2 + G^2 + B^2}$). Diese Punkte stellen dann den Merkmalsvektor dar. Der Merkmalsvektor besitzt in diesem Fall eine sehr große Dimension und weist eine hohe Redundanz auf. Die Aufgabe, Zusammenhänge zwischen dem Bild und der Mundform zu erkennen, verschiebt sich auf den Klassifikator. Es bleibt ihm überlassen, eine Generalisierung der Entscheidung trotz großer Variabilität (Lichtverhältnisse, Skalierung, Rotation, verschiedene Lippenformen) zu finden. In [Bre95] waren Neuronale Netze, in [Mov97] wurden *Hidden Markov Modelle* (HMM) zum Klassifizieren verwendet. Der Klassifikator kann keineswegs mit einer derart großen Variabilität zurecht

kommen, selbst wenn die Lerndaten alle möglichen Situationen repräsentieren. Daher kann diese Methode nur bei genau normalisierten Bildern mit bekannter Skalierung mit mäßigem Erfolg angewandt werden.

Um die Redundanz und die hohe Dimension in den Griff zu bekommen, können einige bekannte Methoden der Sprachverarbeitung hier auch angewandt werden. Mit der Hauptkomponentenanalyse (*engl. Principal Component Analysis, PCA*) [Chi96] oder der Karhunen-Loève-Entwicklung [Mur96] können die Dimension und die Redundanz minimiert werden. Als Erkenner kommen hier an erster Stelle Neuronale Netze (NN) zum Einsatz. Die entstehenden Klassenräume sind bei solchen Verfahren für einen Beobachter nicht direkt nachvollziehbar. Die bildbezogenen Modelle waren bei den früheren Systemen sehr beliebt. Viele Forscher wechselten aber inzwischen zu den vielversprechenden geometrischen Modellen.

3.1.2 Geometrische Modelle

Diese Modelle liefern einen aussagekräftigen Merkmalsvektor. Da das Extrahieren von geometrischen Merkmalen keine einfache Aufgabe ist, wird sehr oft eine manuelle Hilfe vorausgesetzt. In [Pet84] wurden durch einfache Schwellenwertbildung Weite, Breite und Fläche des Mundes aus manuell selektierten Mundbereichen extrahiert. Die bei [Gol93] verwendeten Merkmale wurden ebenfalls manuell extrahiert. Alternativ dazu wurden die Merkmale in [Fin86] zwar automatisch, aber mit Hilfe von reflektierenden Markierungen, die um den Mund befestigt waren, extrahiert. Andere Forscher verwenden spezielle Farben, mit denen die Lippen bemalt werden. Verfahren, die manuelle Hilfe oder spezielle Markierungen benötigen, eignen sich eher für Laborexperimente, bei denen die Tauglichkeit eines bestimmten Merkmals oder Klassifikators zu untersuchen ist, sind aber in der Praxis weniger geeignet.

In den folgenden Abschnitten werden zwei geometrische Modelle vorgestellt, die unter den alltäglichen Bedingungen die geometrischen Merkmale automatisch bestimmen können. Das erste Modell wird mit *active contour model*, oder auch *snakes*, bezeichnet und wurde in [Kas87] beschrieben. Mit diesem Modell können beliebige Objekte basierend auf dem Kontrast, der zwischen ihren Konturen und dem Hintergrund besteht, extrahiert werden. Bei den *active contour models* muss ein Kompromiss zwischen der Fähigkeit, feine Konturen noch gut approximieren zu können, und der Stabilität der verwendeten *snakes* gefunden werden.

Das zweite Modell ist ebenfalls ein geometrisches Modell und basiert auf dem Konzept des *deformable contour models*. Dabei wird eine einfache geometrische Struktur vorgegeben, die nur eine geringe Anzahl von Parametern besitzt. Je nachdem, wie diese Parameter gewählt sind, kann sich diese Struktur geometrisch verändern (z.B sich ziehen, skalieren), sodass sie an die vorgegebene Lippenform angepasst wird.

3.2 Active Contour Model (Snakes)

Snakes sind geschlossene, einfach zusammenhängende Kurven in der Ebene. Sie dienen dazu, einfache Objekte, deren Begrenzung zum Hintergrund (Konturen) eine geschlossene Kurve ist, zu trennen. Eine Snake benötigt also ein Bild, bei dem die Kanten bereits detektiert sind. Häufig, wie auch in dieser Arbeit, wird der Sobel-Operator für diese Kantendetektionsaufgabe eingesetzt. Es handelt sich um zwei Filter (Masken) für jede Komponente des Gradienten:

-1	0	1
-2	0	2
-1	0	1

horizontale Maske

-1	-2	-1
0	0	0
1	2	1

vertikale Maske

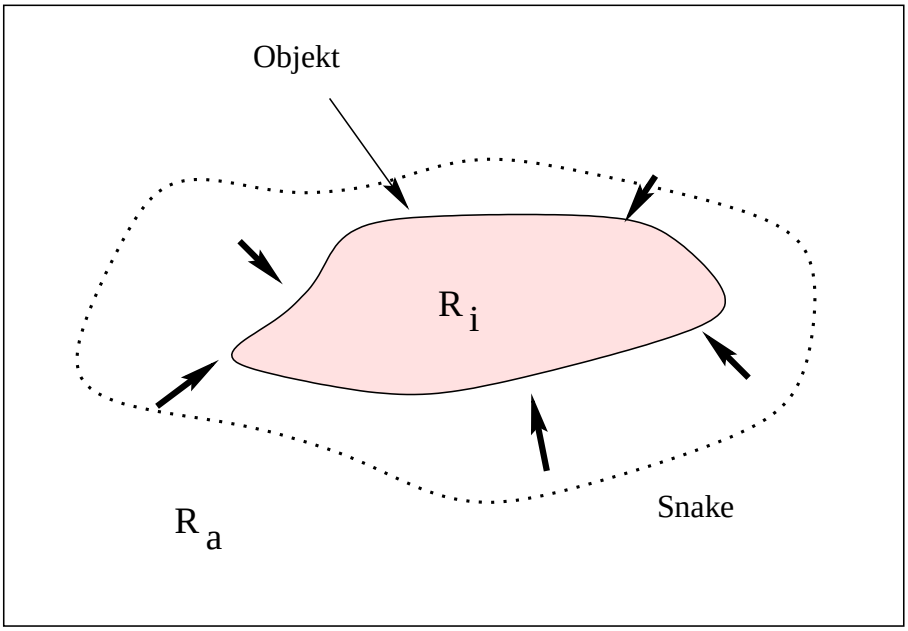


Abbildung 3.1: *Die Initialisierung einer Snake*

Die optimale *Snake* ist eine geschlossene Kurve, die auf den Konturen dieses Objektes liegt. Sie wird in mehreren Iterationsschritten bestimmt. Bei der ersten Iteration wird eine Snake um das Objekt R_i gelegt. Sie muss vollständig im Bereich R_a verlaufen (Abb. 3.1). Bei den folgenden Iterationsschritten wird sie schrittweise verkleinert, bis sie sich den Kanten des Objektes optimal angepasst hat.

Bei Bildern, wie sie in der Realität vorliegen, sind die Konturen des Objektes durch seitliche Beleuchtung, Schatten oder einen fließenden Helligkeitsverlauf zwischen R_a und R_i an mehreren Stellen unterbrochen und daher sehr schwach oder überhaupt

nicht zu erkennen. An solchen Bereichen darf die Snake nicht in R_i „hineinfließen“.

Ein weiteres Problem bezüglich solcher Bilder sind die über das gesamte Bild verteilten Kantenfragmente. Sie werden durch Rauschen, Schatten und andere Störungen hervorgerufen. Solche Stellen dürfen den Weg der Snake zu den Kanten nicht aufhalten; sie müssen übersprungen werden.

3.2.1 Die Bestimmung der optimalen Snake

In der Praxis wird die geschlossene Snake-Kurve $S^{(i)}$ in $N^{(i)}$ Stützpunkten diskretisiert (i steht für den Iterationsschritt, $S^{(0)}$ ist demnach die Snake beim erstem Iterationsschritt). Die Darstellung erfolgt durch eine Menge von N_i Punkten. Die Zahl der Punkte kann prinzipiell für jede Snake $S^{(i)}$ unterschiedlich sein. Im folgenden soll aber $N_i = N$ sein , d.h. die Anzahl der diskreten Punkte ist konstant für alle Snakes:

$$S^{(i)} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_j, \ldots, \mathbf{P}_N\} \tag{3.1}$$

Für jede Snake wird eine Energie definiert, und zwar in der Art, dass diejenige Snake, welche die gesuchte Kontur am besten wiedergibt, die minimale Energie E_o aller Snakes besitzt. Man kann sich diese Energie als ein Energiepotential vorstellen, das an jedem Punkt der Snake eine Kraft zur Folge hat, welche die Kurve in Richtung der Begrenzungslinie zieht. Diese Kraft sorgt dafür, dass die Snake am Ende der Iteration an der Kontur „haften“ bleibt.

Der Greedy-Algorithmus

Die Suche nach dem Energieminimum ist ein Optimierungsproblem einer skalaren Funktion in einem $2N$-dimensionalen Raum, da jeder der N Punkte eine $x-$ und eine y-Koordinate besitzt. Es ist praktisch unmöglich, das absolute Energieminimum analytisch zu finden. Auch die Berechnung der Energien aller möglichen Snakes (eine Gittersuche) ist natürlich kein effezienter Weg.
Es wurden mehrere Verfahren zur Lösung dieses Problems vorgeschlagen. Unter anderem Algorithmen, die auf Variationsrechnung [Kas87] und Dynamischer Programmierung [Ami88] beruhen. Untersuchungen von Williams und Shah [Wil92] haben jedoch gezeigt, dass das mit diesen aufwendigen Methoden gefundene Minimum sich nur unwesentlich von dem Minimum, das mit dem sog. Greedy-Algorithmus gefunden wurde, unterscheidet. Der Greedy-Algorithmus hat jedoch den Vorteil, dass er unkompliziert und schnell ist. Beim Greedy-Algorithmus wird **nicht** nach dem globalen Minimum der Snake-Energie gesucht, sondern nach dem Minimum jeder Punkt-Energie. D.h., jeder Snake-Punkt ist „selbständig“ und versucht, ohne Rücksichtnahme auf alle anderen Punkte, im nächsten Iterationsschritt die für ihn günstigste Position zu finden (daher die Bezeichnung „greedy“, zu dt.: gierig).

Dabei wird für einen Snake-Punkt der neue Punkt gewählt, an dem *seine* Energie in *seiner* Umgebung (z.B. 5×5 Pixel) minimal ist. Durch dieses schrittweise Herantasten

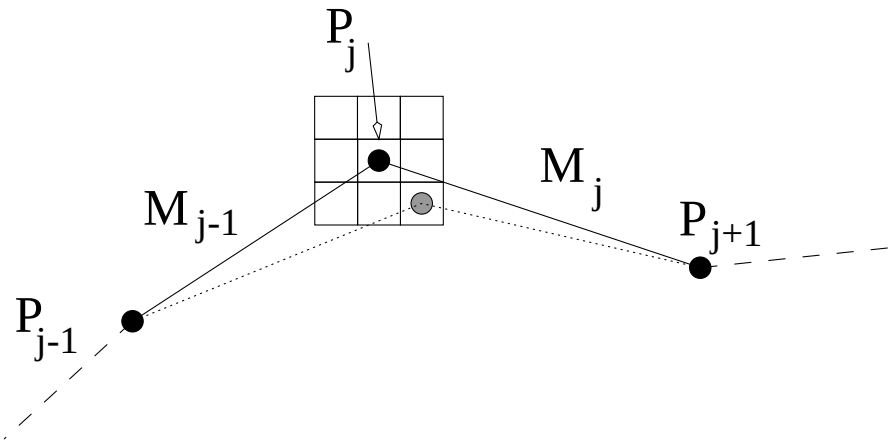


Abbildung 3.2: *Greedy-Algorithmus: In einer Umgebung (hier* 3×3 *Pixel) wird für Punkt* $\mathbf{P}_j^{(i)}$ *die neue Position gewählt, an der sein Energiebeitrag minimal ist.*

an ein Minimum wird nicht unbedingt das globale Minimum aller möglichen Snakes gefunden. In Abb. 3.2 ist der Greedy-Algorithmus mit einer Umgebung von 3×3 dargestellt.

3.2.2 Berechnung der Energieterme

Die gesamte Energie $E^{(i)}$ eines Snakepunktes $\mathbf{P}_j^{(i)}$ bei der Iteration i spaltet sich in zwei Teilenergien auf, die interne Energie $E_{j,int}^{(i)}$ und die externe Energie $E_{j,ext}^{(i)}$:

$$E^{(i)} \quad = \quad E_{j,int}^{(i)} + E_{j,ext}^{(i)} \tag{3.2}$$

Die interne Energie spaltet sich wiederum in zwei Teilenergien auf:

$$E_{j,int}^{(i)} \quad = \quad E_{j,krüm}^{(i)} + E_{j,kont}^{(i)} \tag{3.3}$$

Die Kontinuitätsenergie

Die erste Teilenergie ist die Kontinuitätsenergie oder Dehnungsenergie und hat die Aufgabe, dafür zu sorgen, dass sich die Snake zusammenzieht. Sie muss also umso kleiner sein, je geringer der Umfang der Snake ist. Es bietet sich folgende Definition über das Quadrat des Abstands zum Nachfolgepunkt an:

$$E_{j,kont}^{(i)} \quad = \quad \alpha_j \cdot \left| \mathbf{P}_j^{(i)} - \mathbf{P}_{j-1}^{(i)} \right|^2 \tag{3.4}$$

Bei der Positionierung von $\mathbf{P}_j^{(i)}$ wird also lediglich sein Nachfolger betrachtet. Das führt dazu, dass sich Punkte an den Stellen, wo eine hohe Krümmung der Kontur

vorliegt, ansammeln. Die hohe Krümmungsenergie wird durch Vermindern der Dehnungsenergie kompensiert. Das führt im Extremfall dazu, dass der Rest der Kontur mit nur sehr wenigen Punkten angenähert wird. Diesen Nachteil kann man durch folgende Definition umgehen:

$$E^{(i)}_{j,kont} \quad = \quad \alpha_j \cdot \left| \bar{d} - \left| \mathbf{P}^{(i)}_j - \mathbf{P}^{(i)}_{j-1} \right| \right|^2 \tag{3.5}$$

Es wird also das Quadrat der Abweichung des Abstands zweier benachbarter Punkte von der durchschnittlichen Entfernung $\bar{d}$ der Punkte untereinander berechnet. Bei der Anwendung führt dieser Term zu einer gleichmäßigen Verteilung der Punkte über die Snake.

Die Krümmungsenergie

Der zweite Term ist die Krümmungsenergie. Sie hat die Aufgabe, dafür zu sorgen, dass sich die Snake möglichst wenig krümmt, also einen glatten Verlauf bekommt.

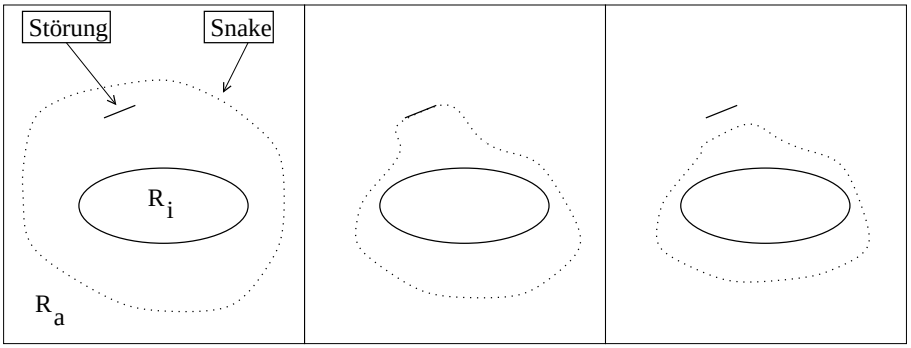


Abbildung 3.3: *Krümmungs- und Dehnungsenergien sorgen dafür, dass Störungen im Bild übersprungen werden*

Trifft die Snake während ihrer Kontraktion auf ein durch Störungen hervorgerufenes Kantenstückchen, dann „fließt“ sie neben ihm weiter, während ein Teil der Snake an der Kante stoppt. Das führt dazu, dass im Kantenbereich eine hohe Krümmung und Dehnung auftritt: die Krümmungsenergie und die Dehnungsenergie nehmen zu. Es wird für die Snake also energiemäßig günstiger, das Kantenfragment zu überspringen und ihrem Rest zu folgen (Abb. 3.3).
In [Kas87] wird für die Krümmungsenergie

$$E^{(i)}_{j,krüm} \quad = \quad \beta_j \cdot \left| \mathbf{P}^{(i)}_{j-1} - 2\mathbf{P}^{(i)}_j + \mathbf{P}^{(i)}_{j+1} \right|^2 \tag{3.6}$$

verwendet.

Diese Definition berücksichtigt jedoch nicht, dass die Punkte von Kurven, deren Krümmung miteinander verglichen werden soll, teilweise stark unterschiedliche Entfernungen zueinander haben. Dieser Effekt soll mit Abbildung 3.4 deutlich gemacht

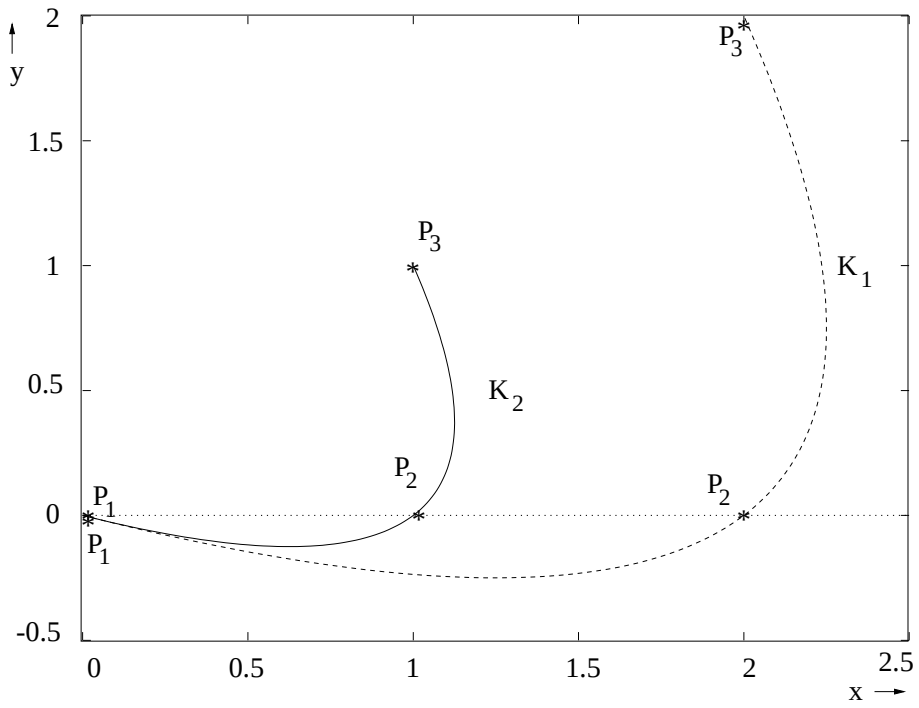


Abbildung 3.4: *Krümmungsberechnung mit finiten-Differenzen*

werden. Kurve K_2 ist stärker gekrümmt als Kurve K_1. Berechnet man die Krümmung der Kurven jeweils an dem mittleren Punkt $\mathbf{P_2}$ mit der finite-Differenzen-Methode (Gl. 3.6), dann erhält man für die weite Kurve einen Wert von 8 und für die enge Kurve einen Wert von 2. Also eine geringere Krümmung für die in Wirklichkeit stärker gekrümmte Kurve.
Als Ausweg wurden die Punkte $\mathbf{P}_{j-1}^{(i)}, \mathbf{P}_{j}^{(i)}, \mathbf{P}_{j+1}^{(i)}$ durch ein Polynom zweiten Grades $\mathbf{z}(t)$ ersetzt. Da dieses Polynom eine kontinuierliche Funktion ist, kann dann die Krümmung analytisch genau berechnet werden.

$$\begin{aligned} \mathbf{z}(t) &= \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \quad t = 0 \ldots 1 \\ &= \begin{pmatrix} a_0 + a_1 t + a_2 t^2 \\ b_0 + b_1 t + b_2 t^2 \end{pmatrix} \end{aligned} \tag{3.7}$$

Die Bestimmung der Koeffizienten a_0, a_1, a_2, b_0, b_1 und b_2 erfolgt nach der Vorschrift

$$\begin{aligned} \mathbf{z}(t=0) &\stackrel{!}{=} \mathbf{P}_{j-1} \\ \mathbf{z}(t=0,5) &\stackrel{!}{=} \mathbf{P}_{j} \\ \mathbf{z}(t=1) &\stackrel{!}{=} \mathbf{P}_{j+1}. \end{aligned} \tag{3.8}$$

Auf diese Art kann die Krümmung von $\mathbf{z}$ an der Stelle $t = 0,5$ $(= \mathbf{P}_j)$ berechnet werden. Die Berechnung erfolgt in Anhang B. Man erhält folgende Gleichung für die

Krümmung:

$$\kappa_j(t = 0,5) = \\ 8 \cdot \frac{x_{j-1}(y_j - y_{j+1}) - x_j(y_{j-1} - y_{j+1}) + x_{j+1}(y_{j-1} - y_j)}{((x_{j-1} - x_{j+1})^2 + (y_{j-1} - y_{j+1})^2)^{3/2}} \tag{3.9}$$

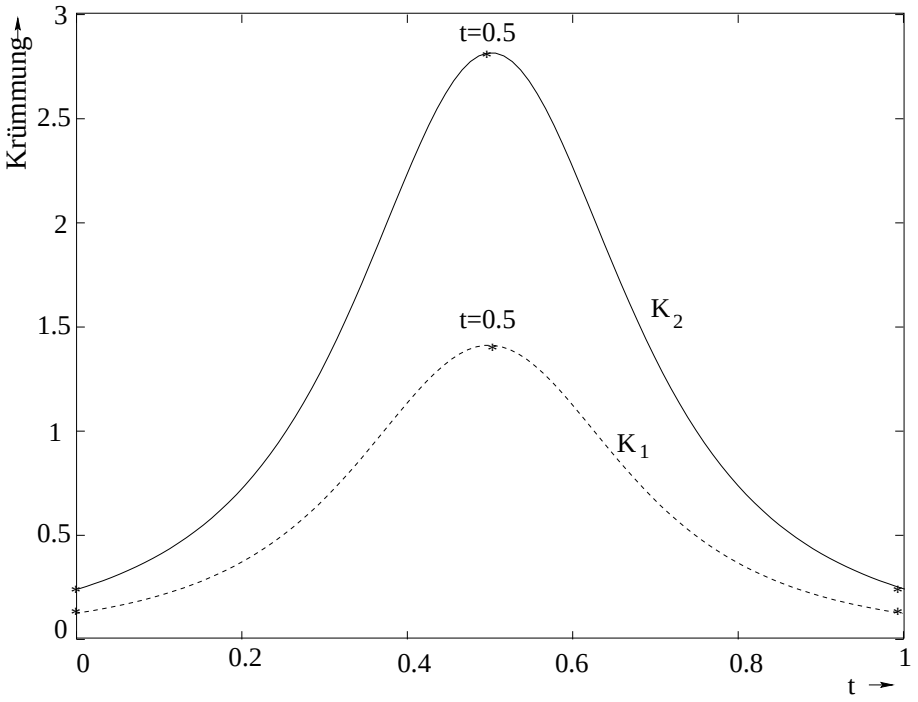
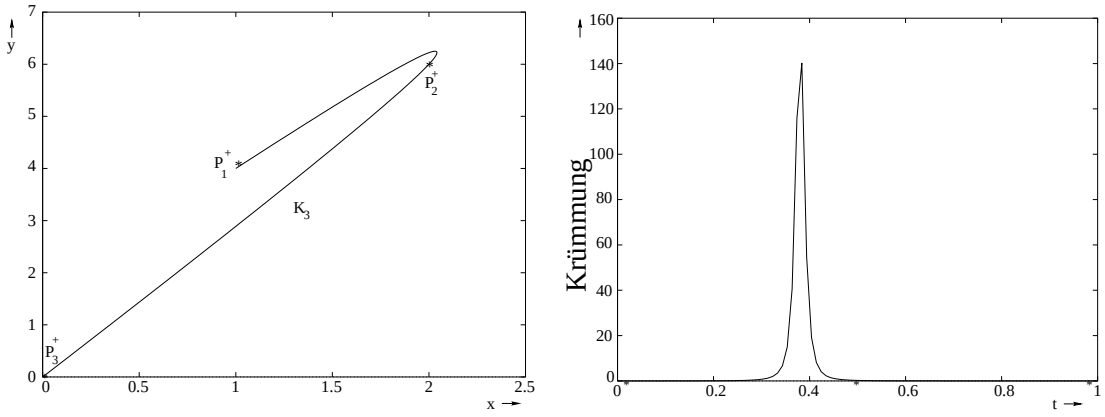


Abbildung 3.5: *Krümmungsverlauf bei Polynomnäherung*

Für die beiden Kurven aus Abb. 3.4 gibt Abb. 3.5 den Krümmungsverlauf im Bereich von $t = 0$ bis $t = 1$ wieder. Der Punkt $\mathbf{P_2}$ entspricht hier einem Wert von $t = 0,5$. Es ergibt sich ein Krümmungsverlauf, der den Kurven entspricht. Man könnte die Krümmungsenergie über den Wert von $\kappa_j(t = 0,5)$ definieren:

$$E^{(i)}_{j,krüm} = \beta_j \cdot |\kappa_j(t = 0,5)| \tag{3.10}$$

Es hat sich jedoch gezeigt, dass diese Definition immer noch Probleme mit sich bringt, denn durch die Annäherung mit Polynomen ergibt sich meist kein Kurvenverlauf wie er in Abb. 3.5 dargestellt ist. Dort liegt der Punkt $\mathbf{P}_2$ an der Stelle der Kurve, an der die größte Krümmung vorliegt. Dies ist in der Regel nicht so. Einen Extremfall zeigt Abb. 3.6(a). Die Kurve verläuft zwar durch alle drei Punkte, aber das Maximum der Krümmung befindet sich nicht bei $t = 0,5$, was dem Punkt $\mathbf{P}_2$ entspräche (Abb. 3.6(b)).
Da die Definition aus Gl. 3.10 nur den Wert von $\kappa_j(t)$ für $t = 0,5$ berücksichtigt, wird das Maximum der Krümmung schon bei $t \approx 0,38$ erreicht. Eine einfache



(a) (b)

Abbildung 3.6: *(a) Extremer Kurvenverlauf: K_3 verläuft zwar durch alle drei Punkte, hat aber einen „Überschwinger“; (b) Das Maximum der Krümmung liegt nicht bei $t = 0,5$ sondern bei $t \approx 0,38$.*

Lösungsmöglichkeit besteht darin, nicht nur den Wert $\kappa_j(t = 0,5)$ zu betrachten, sondern alle Werte dazwischen. Das wird von dieser Definition berücksichtigt:

$$E^{(i)}_{j,krüm} = \beta_j \int_{t=0}^{1} \kappa_j(t)\, dt \tag{3.11}$$

Die externe Energie

Die externe Energie sorgt dafür, dass sich die Snake bevorzugt zu den Stellen hin bewegt, an denen sich eine Grenze zwischen dem äußeren Bereich R_a und dem inneren Bereich R_i befindet. Der Gradient des Grauwertbildes (die Wirkung des Sobel-Operators) besitzt an solchen Stellen einen hohen Betrag. Deshalb liegt es nahe, als externe Energie die Summe der Intensitätsgradienten aller Bildpunkte entlang der Snake zu verwenden. Weil nur hier Information aus dem zu segmentierenden Bild einfließt, wird diese Energie auch Bildenergie genannt. Sie stellt die Verbindung zum vorliegenden Bild dar. Man könnte für diesen Energieterm

$$E^{(i)}_{j,ext} = -\gamma_j \cdot \left| \mathbf{G}(I)\big|_{\mathbf{P}^{(i)}_j} \right| \tag{3.12}$$

verwenden. Diese Definition führt aber dazu, dass die einzelnen Punkte oft an kleinen Kantenfragmenten im Bild, an denen der Gradient einen großen Betrag besitzt, „hängenbleiben“. Entlang der Verbindungsstrecken M_j [1] zwischen den Punkten der Snake nimmt der Gradient nur sehr kleine Beträge an. Es liegt also nahe, auch auf eine möglichst gute „Deckung“ der Verbindungsstrecken M_j mit den Kanten im Bild zu

[1] Die Strecke M_j ist die Menge aller Bildpunkte (Pixel), die auf der Verbindungslinie zwischen dem Snake-Punkt $\mathbf{P}_{j-1}$ und dem Snake-Punkt $\mathbf{P}_j$ liegen, s. Abb. 3.2.

achten. Erreicht wurde das mit dieser Definition:

$$E_{j,ext}^{(i)} = -\gamma_j \sum_{\mathbf{v} \in M_j, M_{j+1}} \left| \mathbf{G}(I)|_{\mathbf{v}} \right| \tag{3.13}$$

Abbruchkriterium

Das Abbruchkriterium wird vom Greedy-Algorithmus geliefert. Die beste Snake gilt als die, bei der die Summe aller Punktenergien bei einem Iterationsschritt nicht mehr wesentlich kleiner (besser) werden als beim vorherigen Iterationsschritt.

$$E_{summe}^{(i-1)} - E_{summe}^{(i)} < \Delta E_{Schwelle}, \quad \text{mit} \tag{3.14}$$

$$E_{summe}^{(i)} = \sum_{j=0}^{N} E_j^{(i)} \tag{3.15}$$

Falls dieses Abbruchkriterium nach einer vorgegebenen Anzahl von maximal erlaubten Iterationsschritten noch nicht erreicht ist, wird trotzdem abgebrochen (die Zahl der maximal erlaubten Iterationsschritte in unserer Arbeit war 100).

3.2.3 Die Leistung der Snake-Methode

Um die Kanten der Lippen mit einer akzeptablen Auflösung durch eine Snake annähern zu können, braucht man mindestens 30 bis 40 Snake-Punkte. Die Optimierung der Gewichtsfaktoren α_j, β_j und γ_j stellt ein kompliziertes Problem dar.

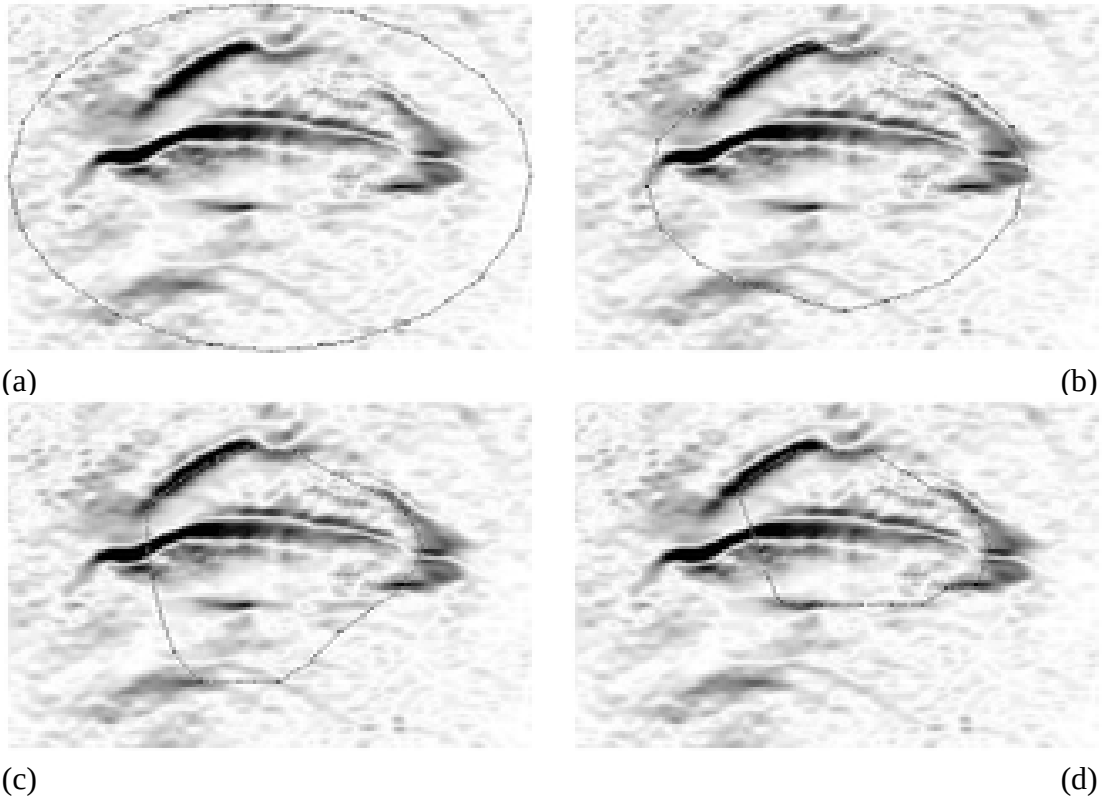
Größere Werte von α_j führen zu einer verstärkten Kontraktion der Snake, die optimale Kontur wird in der Regel dadurch kleiner. Wird dieser Wert zu groß gewählt, dann führt dies dazu, dass die Snake an der gesuchten Kontur nicht stoppt, sondern ihren Weg im Laufe der Iteration weiter fortsetzt.

Größere Werte von β_j lassen die Snake „steifer“ werden. Dadurch werden gekrümmte Bereiche einer Kontur nicht mehr so gut angenähert. Auf der anderen Seite allerdings sorgt gerade dieser Wert dafür, dass die Snake während ihres Weges Störungen, die nicht zur gesuchten Kontur gehören, überspringt. Dazu darf β_j jedoch nicht zu klein gewählt werden. Dieser Wert ist der kritischste bei der Optimierung der Gewichtsfaktoren und nimmt großen Einfluss auf das Endergebnis.

Größere Werte für γ_j sorgen schließlich dafür, dass die Information, die der Gradient liefert, stärker berücksichtigt werden. Die Form der Snake wird dann mehr vom Bild bestimmt als von der internen Energie. Allerdings wird es bei großem γ_j für die Snake schwieriger, Störungen im Bild zu überspringen und offene Bereiche in der Kontur zu überbrücken.

Eine weitere kritische Rolle spielt die Wahl der betrachteten Umgebung beim Greedy-Algorithmus. Wählt man die Umgebung klein (z.B. 3×3 Pixel), so erfolgt die Annäherung an die gesuchte Kontur nur sehr langsam und das Überspringen von Kantenfragmenten wird der Snake erschwert. Die Snake wird durch eine zu kleine Umgebung

leicht in einem lokalen Minimum festgehalten. Eine zu große Umgebung führt dazu, dass sich die Snake „verschlingt“. Das geschieht, wenn gegenüberliegende Teile der Kurve übereinander hinwegwandern. Außerdem führt eine Vergrößerung der Umgebung zu einem erheblich größeren Rechenaufwand. Bei N Punkten mit je einer $U \times U$ großen Umgebung müssen für einen Iterationsschritt NU^2 Teilenergien berechnet werden. Wie man sieht, wächst der Rechenaufwand quadratisch mit der Seitenlänge der Umgebung.



(a) (b)

(c) (d)

Abbildung 3.7: *(a) Initialisierung der Snake, (b) nach 10 Iterationen, (c) nach 20 Iterationen und (d) nach 45 Iterationen*

Abb. 3.7 zeigt ein Beispiel für das Verhalten der Snake, bei dem die Approximation der linken Hälfte des Mundes versagt.
Man kann die Nachteile der Snake-Methode wie folgt zusammenfassen:

- Die Leistung der Snake ist sehr stark von der Güte der detektierten Kanten abhängig.
- Es werden ca. 40 Snake-Punkte benötigt. In jedem Iterationsschritt werden alle Punkte unabhängig voneinander optimiert. Dies verlangt einen großen Rechenaufwand.
- Die Wahl der Gewichtsfaktoren ist sehr kritisch und beeinflusst das Verhalten der Snake sehr stark. Es müssen immer Kompromisse bezüglich der Fähigkeit einer Snake, Störungen zu überspringen, und der Güte der Lippenapproximation eingegangen werden.

- Der Greedy-Algorithmus bietet zwar eine schnelle Optimierung an, jedoch wird das globale Minimum nicht gefunden. Es besteht die Gefahr, bei lokalen Minima „stecken“ zu bleiben.

- Bei falscher Parameterwahl kann es vorkommen, dass die Snake mehrere Schleifen bildet oder durch eine unscharfe Kante „hineinfließt“.

- Die Anzahl der Iterationsschritte schwankt stark. Man kann nicht im voraus genau feststellen, wie lange die gesamte Rechenzeit dauert.

Aus diesen Gründen wurde ein anderer Algorithmus untersucht, bei dem die Anzahl der Parameter wesentlich geringer ist. Der Rechenaufwand ist zudem viel günstiger. Der Algorithmus verwendet ein *Deformable Contour Model* und wird im folgendem Abschnitt ausführlich erklärt.

3.3 Deformable Contour Model

Das Active Contour Model (Snakes) vom Abschnitt 3.2 ist in der Lage, Objekte unabhängig von ihrer Form aus Bildern zu segmentieren. Bei unserem Fall, nämlich dem Segmentieren der Lippenkonturen, ist aber eine ungefähre Form des Objektes schon im voraus bekannt. Um dieses *a-priori*-Wissen auszunutzen, kann man eine bestimmte Kurve bzw. Funktion vorgeben, die die Lippenform approximieren soll. Diese Funktion soll möglichst wenige Parameter besitzen. Durch Veränderung der Parameter lässt sich der Kurvenverlauf innerhalb gewisser Grenzen beeinflussen, bis eine optimale Form erreicht wird.

Die Vorteile gegenüber dem Snake-Algorithmus sind folgende:

- Der Rechenaufwand reduziert sich dramatisch gegenüber dem Snake-Algorithmus. Bei einer Snake mit 40 Punkten und einer durchschnittlichen Anzahl von Iterationsschritten $= 50$ muss die Energie bei einer Greedy-Optimierung mit 9×9-Umgebungsbetrachtung insgesamt $40 \cdot 50 \cdot (9 \cdot 9) = 162000$ Mal pro Bild berechnet werden. Die Energieberechnung ist mit ihren drei Teilenergien nicht gerade schnell. Hinzu kommt auch noch die Zeit für die Gradientenberechnung mit dem Sobel-Operator.

- Da die Kurve als Ganzes optimiert wird, ist die Gefahr, dass einige einzelne Punkte des Modells an Störungen im Bild „haften“, nicht mehr gegeben. Auch das Problem, dass einige Punkte an unscharfen Kanten in das Objekt „hineinfließen“, ist nicht mehr vorhanden.

- Kurvensymmetrie ist durch eine entsprechende Wahl des Modells gewährleistet.

- Es sind nur wenige Parameter zu optimieren. Die Wahl der Gewichtsfaktoren der Energieterme entfällt.

- Eine Snake war nur in der Lage, die Außenkonturen eines Objekts zu modellieren. Mit dem Deformable Contour Model können zusätzlich die inneren Konturen bzw. Strukturen des Objektes auch modelliert werden.

Ausgangspunkt für dieses Verfahren ist, wie auch bei der Snake, ein kantendetektiertes Bild. Alternativ zum Sobel-Operator wird hier eine andere Vorgehensweise vorgestellt, die, wie bei der Gesichtslokalisierung, mit einem Farbmodell arbeitet.

3.3.1 Kantendetektion der Lippen mit einem Farbmodell

Bei der Kantendetektion mit einem FIR-Filter (wie z.B. mit dem Sobel-Operator) wird zunächst das Bild in Graustufen (Intensität) umgewandelt. Dabei wird der 3-dimensionale Raum in einen 1-dimensionalen Raum transformiert, was notwendigerweise zu einem großen Informationsverlust führt. Die Intensität der Hautfarbe und die der Lippen liegen dann eng aneinander in dem Intensitätsraum. Aus diesem Grund ist die Kante zwischen den Lippen und der Haut unscharf. Man kann sicherlich den Bildausschnitt um die Lippen normalisieren, sodass der dynamische Bereich größer wird. Dabei werden aber die Bildstörungen um so mehr hervorgehoben und es entstehen viele kleine Störungen und Fragmente im Bild.

Es liegt also nahe, den Farbsprung zwischen den Lippen und der umgebenden Hautfarbe auszunutzen. Aufgrund der gleichen Argumente, die wir früher für die Verwendung des chromatischen Farbraums bei der Gesichtssuche (Abschnitt 2.2.4) aufgeführt haben, kommt auch hier dieser Farbraum zum Einsatz. Man kann hier auch eine Gaußfilterung durchführen. Diesmal sollen aber die Lippenpunkte (und nicht wie zuvor die Gesichtspunkte) segmentiert werden. Im Gegensatz zum Gesicht variiert die Farbverteilung der Lippen bei den verschiedenen Personen (gerade bei Frauen!) sehr stark. Hier fehlt also die Vorkenntnis über die Verteilung der Lippenfarben im chromatischen Farbraum. Für die Lösung dieses Problems bieten sich zwei Möglichkeiten an:

1. **manuell**: Für jeden Sprecher muss eine separate Kenntnis über die Farbverteilung seiner Lippen vorliegen.

2. **automatisch**: Die Farbverteilung der Lippen kann automatisch gefunden werden. Dabei wird das Histogramm des chromatischen Farbraums für die Lippenregion untersucht. Die Verteilung in diesem Histogramm kann durch zwei Gaußverteilungen approximiert werden. Die eine entspricht der der Lippen, die andere der der Hautfarbe.

In dieser Arbeit wurde eine „Mischlösung“ benutzt. Der Sprecher soll beim Start der Anwendung einen Teilbereich seiner Lippen manuell im Bild (das auf dem Bildschirm erscheint) mit der Maus markieren. Die Parameter der Gauß-Filterung werden dann automatisch berechnet und können in einer Datenbank abgelegt werden.

Abbildung 3.8 zeigt die Wirkung der Gaußfilterung der Lippen anhand eines Beispiels. Die Konturen der Lippen heben sich von der restlichen Hautfarbe so stark ab, dass für die Kantendetektion ein ganz simpler binärer Entscheider verwendet werden kann:

(a) (b)

Abbildung 3.8: *(a) Originalbild, (b) nach der Lippen-Gaußfilterung*

Das Bild, das nach der Filterung entsteht, ist ein binäres Bild. Alle Bildpunkte gehören entweder der Klasse '1' (Lippenregion) oder der Klasse '0' (keine Lippenregion) an. Daher werden die Kanten auch mit einer binären Maske detektiert: Jeder Bildpunkt gilt als Kantenpunkt, falls er in seiner 2×2- Umgebung mindestens einen benachbarten Punkt hat, der zu einer unterschiedlichen Klasse gehört (XOR-Funktion). Das Ergebnis dieser binären Kantendetektion ist in Abb. 3.11(a) dargestellt.

3.3.2 Approximation mit Parabelstücken

In [Kor97] wurden einige einfache Kurven zur Approximation der Lippen untersucht. Diese waren u. a. eine Ellipse für beide Lippen, zwei Parabelstücke (für jeweils Ober- und Unterlippe) und zwei biquadratische Kurven. In weiteren Untersuchungen hat sich die in Abb. 3.9 dargestellte Kurve als sehr stabil und einfach erwiesen.
Dabei wird die obere Lippe mit zwei Parabelkurven approximiert, während die untere Lippe nur mit einer Parabelkurve approximiert wird. Die Modellierung der oberen Lippe mit zwei Kurven ist vor allem bei leicht zur Seite geneigten Gesichtern vorteilhaft.

Die Parameter der drei Parabelkurven werden unabhängig voneinander optimiert. Ziel dieser Optimierung ist, die Parabel so zu parametrieren, dass sie möglichst exakt auf den Konturen der Lippen liegt. Da die Lippenkonturen nicht exakt den Verlauf einer Parabelkurve haben, wird es immer einen Fehler geben. Die 'beste' Parabel ist die, bei der dieser Fehler auf ein Minimum reduziert wird. Es ist also denkbar, die Summe der absoluten Abstände zwischen einer Parabel und der von ihr zu modellierenden Kante als Fehlerfunktion zu verwenden.

Die Parabel $A^{(i)}$ wird als eine Menge von M Punkten betrachtet:

$$A^{(i)} \quad = \quad \{(x_l^{(i)}, y_l^{(i)})\}, \quad \text{mit: } l = 1, \ldots, M \tag{3.16}$$

Die Beziehung zwischen $x_l^{(i)}$ und $y_l^{(i)}$ ist durch die Parabelgleichung gegeben:

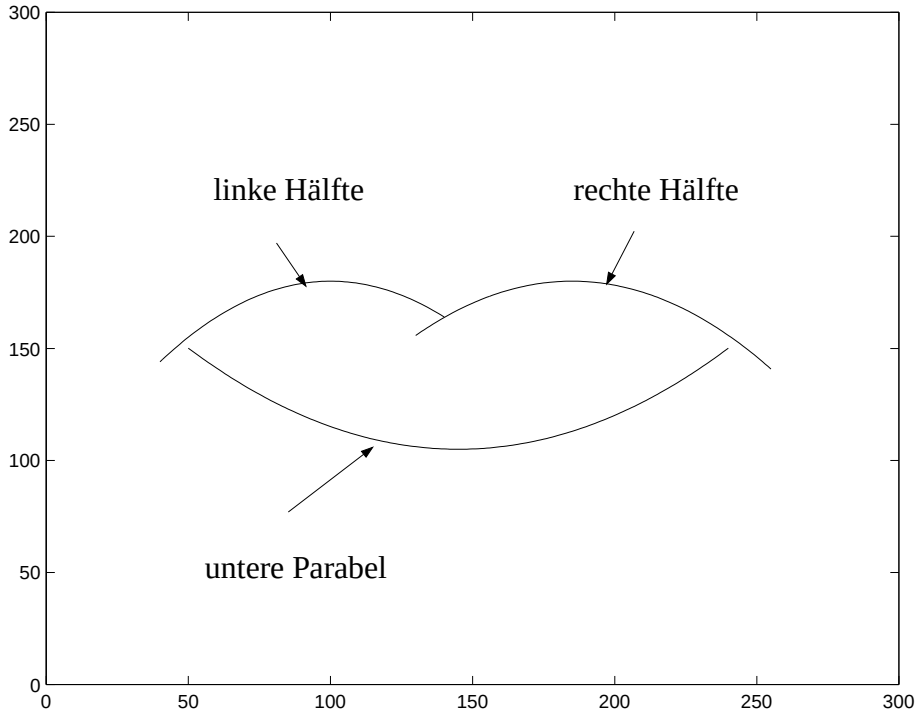


Abbildung 3.9: *Approximationsmodell mit 3 Parabelkurven*

$$y^{(i)} - y_0^{(i)} = c \cdot (x^{(i)} - x_0^{(i)})^2 \tag{3.17}$$

Die zu approximierende Kante wird durch die Menge der Punkte B gegeben:

$$B \quad = \quad \{(u_l, v_l)\}, \quad \text{mit: } l = 1, \ldots, M \tag{3.18}$$

Sie soll aus dem kantendetektierten Bild $\mathbf{G}$ bestimmt werden. Die Fehlerfunktion lautet dann (Abb. 3.10):

$$\epsilon^{(i)} = \sum_{l=1}^{M} |y_l^{(i)} - v_l| \tag{3.19}$$

Die beste Parabel $A^{(best)}$ realisiert dann:

$$\epsilon^{(i)}|_{min} = \min \Big\{ \sum_{l=1}^{M} |y_l^{(best)} - v_l| \Big\} \tag{3.20}$$

Die Bestimmung der Kantenpunkte v_l ist jedoch problematisch. In der Realität ist die Kante keine durchgehende Kurve, wie in Abb. 3.10 dargestellt, sondern es handelt sich vielmehr um mehrere unterbrochene Kurvenfragmente, die auch parallel zueinander laufen. Es wurde daher ein anderes Optimalitätskriterium verwendet. Für eine bestimmte Parabel werden alle Punkte der Parabel, die gleichzeitig auf einem Kantenpunkt liegen, aufgezählt. Die Anzahl der gemeinsamen Punkte (K) zwischen $A^{(i)}$ und

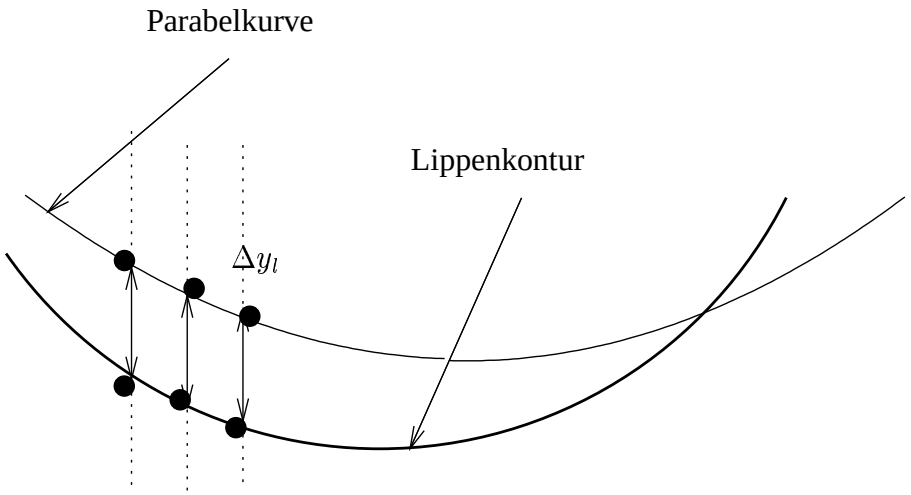


Abbildung 3.10: *Minimierung der absoluten Abstände*

B (die Schnittmenge) ist also in diesem Fall unser Optimalitätskriterium:

$$K = \sum_{l=1}^{M} g(x_l, y_l), \tag{3.21}$$

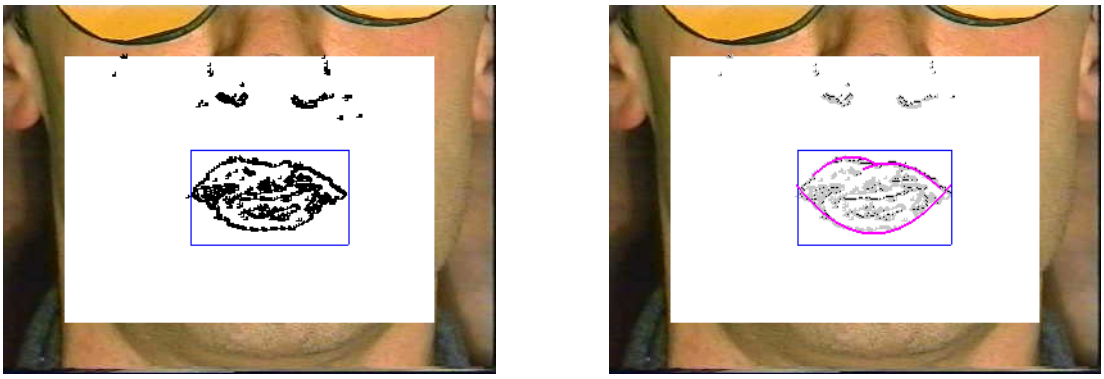
$$\mathbf{G} = \begin{bmatrix} g(x_1, y_l) & \cdots & g(x_W, y_1) \\ \vdots & \ddots & \vdots \\ g(x_1, y_H) & \cdots & g(x_W, y_H) \end{bmatrix}$$

$\mathbf{G}$ ist hier das kantendetektierte Bild: es handelt sich um eine Matrix mit der gleichen Dimension des Bildes $(W \times H)$. Sie besitzt an den Stellen, wo ein Kantenpunkt detektiert wurde, den Wert '1', sonst '0'.
Die optimale Parabel ist in diesem Fall diejenige, die Gl. 3.21 maximiert. Das Maximum der Gl. 3.21 wird mit einer Gittersuche gefunden. Für die 3 Parameter der Parabel (die Krümmung und die beiden Koordinaten des Extrempunkts) werden jeweils erlaubte Bereiche und Quantisierungsstufen definiert. Dabei entsteht ein 3-dimensionales Gitter. An jedem Punkt dieses Gitters wird K nach Gl. 3.21 berechnet. Der Gitterpunkt, der den maximalen Wert für K liefert, ist der optimale Parametersatz für die Parabel.

Das Ergebnis dieser Optimierung wird anhand eines Beispiels in Abb. 3.11 dargestellt.

3.4 Die Leistung der Kantendetektion und der Approximation

In Abschnitt 3.3 wurde das „*Deformable Contour Model*" vorgestellt. Dabei wurden zunächst die Lippen mit einem Farbmodell lokalisiert. Die Arbeitsweise dieses Modells setzt keine spezielle Beleuchtung und keine besondere Farbe der Lippen voraus,



(a) (b)

Abbildung 3.11: *(a) Kantendetektion, (b) Modellierung mit den Parabelkurven*

was der von uns geforderten „*real life*“-Situation entspricht. Mit Hilfe von speziellen Laborgeräten ließe sich die Position der Lippen sicherlich exakter bestimmen. Solche Laborgeräte eignen sich jedoch kaum für derartige „*real life*“-Situationen. Gründe dafür sind, neben der hohen Kosten, die Voraussetzungen bezüglich der Beleuchtung, Aufnahmegeräte und Markierung der Lippen.
Es ist dennoch wichtig, unser System zur Lokalisierung der Lippen mit einem solchen Laborsystem, das hier als ein *Referenzsystem* gilt, zu vergleichen, um seine Leistung beurteilen zu können. Für diesen Zweck kam das System von Ganymedia [Gany], das nach dem Chromakey-Prinzip arbeitet, zum Einsatz. Das System besteht aus einer Präzisionskamera, die an einem speziellen Helm befestigt ist, und die die Mundregion durch eine *Spotlight*-Quelle konstant beleuchtet (Abb. 3.12).

Abbildung 3.12: *Das Ganymedia-System*

Die Lippen des Sprechers müssen zuvor mit einer speziellen Blautinte bemalt werden. Mit Hilfe der mitgelieferten Hardware können dann diese blaumarkierten Flächen in Echtzeit vom Restbild herausgefiltert werden. Das Prinzip entspricht der Blauflächeneinblendung, die bei Fernsehstudios sehr bekannt ist (z.B. bei der Einblendung von verschiedenen Wetterkarten während des Wetterberichts). Das Referenzsystem ist so

in der Lage, die exakte Mundbreite und Mundhöhe in Echtzeit zu liefern.

Damit die von dem Referenzsystem gelieferten Daten (Mundbreite und Mundhöhe) mit den von unserem System ermittelten Daten verglichen werden können, wurden zwei Datensätze erstellt[2]. Der erste Datensatz stammt von einem Sprecher, der die Zahlen von Null bis Zehn aufgezählt hatte. Seine Lippen waren blau gefärbt und die Daten wurden mit dem Referenzsystem ermittelt. Der zweite Datensatz ist *identisch* mit dem ersten Datensatz, d.h., der gleiche Sprecher hat genau die gleiche Wortabfolge gesprochen, allerdings ohne dass seine Lippen bemalt waren. Die Aufnahme und Bestimmung der Mundbreite und Mundhöhe erfolgten aber diesmal mit unserem System. Natürlich bezieht sich hier das Wort *identisch* nur auf den Inhalt des Datensatzes.

3.4.1 Wie kann man vergleichen?

Die beiden Datensätze stammen zwar vom gleichen Sprecher, jedoch ist es unmöglich, dass der Sprecher, trotz großer Mühe, die Wörter jedesmal genauso schnell aussprechen kann. Außerdem kann man nicht vermeiden, dass sich sein Kopf während der Aufnahme minimal bewegt. Man kann also die Daten beider Verfahren nicht direkt miteinander vergleichen; es müssen zuerst die Datensätze aneinander angepasst werden. Es bieten sich Verfahren der dynamischen Programmierung, wie das in Abschnitt 5.1 beschriebene *Dynamic Time Warping*, an.
Nun stellt sich die Frage, wo genau die zeitliche Anpassung stattfinden soll. Abb. 3.13 zeigt die wesentlichen Schritte bei der Sprach- und Bildaufnahme. Die zeitliche Anpassung kann demnach an 3 verschiedenen Stellen stattfinden:

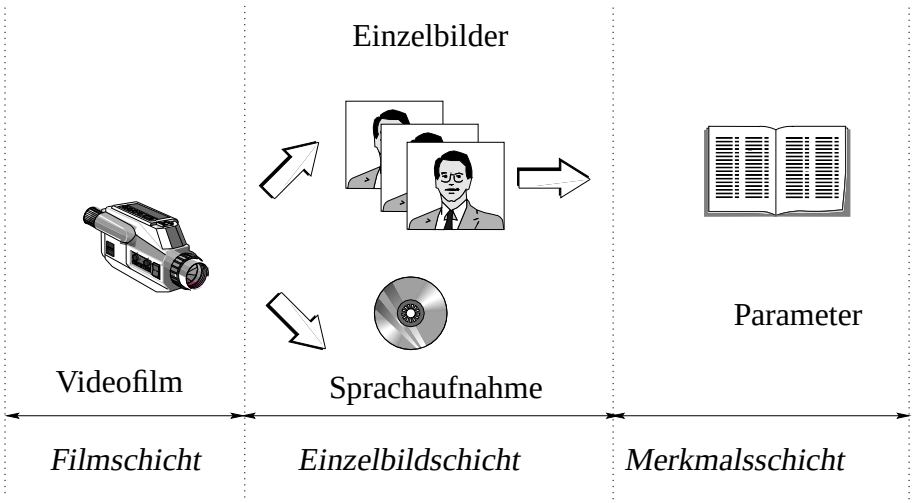


Abbildung 3.13: *Die Stellen eines möglichen Vergleichs*

1. **Auf der Ebene der Videoaufnahme**: Dafür muss die eine Videoaufnahme aus dem einem Datensatz an die entsprechende Videoaufnahme des zweiten Daten-

[2]Die Datensätze wurden uns freundlicherweise vom INP, Grenoble, Frankreich, zur Verfügung gestellt

satzes angepasst werden. Die Bildsequenz muss also interpoliert bzw. verkürzt werden, sodass sich eine gleiche Anzahl von Bildern in jeder Videoaufnahme ergibt. Zusätzlich zur zeitlichen Anpassung muss auch eine räumliche Anpassung stattfinden, da auch die Kopfbewegung beider Aufnahmen nicht identisch war. Eine solche Anpassung ist sehr aufwendig und übertrifft bezüglich der Komplexität das eigentliche Problem der Bestimmung der Lippenposition. Die Anpassung auf dieser Ebene ist deswegen nicht sinnvoll.

2. **Anpassung der Sprachdateien**: Zu jeder Videoaufnahme wurde synchron die Sprache aufgenommen. Es ist daher denkbar, die entsprechenden Sprachaufnahmen der jeweiligen Videoaufnahmen mit Hilfe des DTW-Algorithmus aneinander anzupassen und sich die Anpassungsinformationen zu „merken“, d.h. die Stellen, wo eine Verkürzung bzw. Interpolation nötig war; und dann schließlich die eine Videoaufnahme mit Hilfe von diesen Informationen an die andere anzupassen. Bei der Implementierung dieser Idee sind wir aber auf zwei Probleme gestoßen. Das erste Problem war die enorme Menge an Arbeitsspeicher, die der DTW-Algorithmus für das Anpassen von zwei Sprachsignalen benötigt (jede Sprachaufnahme ist zwischen 10 und 20 Sekunden lang). Das Anpassen von Segmentenergien statt Einzelabtastwerten ließ zwar den Speicherbedarf sinken, jedoch scheiterte die Idee letztendlich am zweiten Problem: während der Sprachpausen zwischen den Einzelwörtern lieferte der DTW-Algorithmus zufällige Ergebnisse.

3. **Anpassung der Daten**: Es blieb also nur noch übrig, die gewonnenen Daten an sich aneinander zeitlich anzupassen. Um das Konzept der Anpassung zu erklären, sei hier ein Beispiel erwähnt. Es wird angenommen, dass eine Videoaufnahme V_1 20 Sekunden lang ist. Diese soll an die Aufnahme V_2, die 16 Sekunden lang ist, angepasst werden. Da das PAL-System eine Bildrate von 25 Bildern/s hat, besitzt V_1 insgesamt $F_1 = 20 \cdot 25 = 500$ Bilder, V_2 dagegen besitzt $F_2 = 400$ Bilder. Das Referenzsystem liefert also für V_1 einen zweidimensionalen Vektor (Breite und Höhe des Mundes) für jedes Bild und das insgesamt 500 mal. Unser System liefert dagegen nur 400 mal einen solchen Vektor. Mit Hilfe des DTW-Algorithmus werden die 400 Vektoren „gezogen“, damit sie nun mit den Daten von V_2 verglichen werden können.

Die extrahierten Parameter zweier Videofilme werden also zeitlich aneinander angepasst. Der Vorteil des DTW-Algorithmus gegenüber einer linearen Zeitanpassung besteht darin, dass die Dynamik der Daten an sich (Positionen von Maxima und Minima) berücksichtigt wird.

3.4.2 Anpassungsergebnisse

Nachdem zwei Ergebnisdateien aus verschiedenen Datensätzen angepasst wurden, kann der *Fehler* in der Bestimmung der Höhe und Breite nach:

$$w_{Fehler} = \frac{1}{F_1} \sum_{F_1} |w_1 - w_2| \tag{3.22}$$

$$h_{Fehler} = \frac{1}{F_1} \sum_{F_1} |h_1 - h_2|,$$

berechnet werden. F_1 ist die Anzahl der Bilder nach der Anpassung, w_1, w_2 bzw. h_1, h_2 sind die ermittelten Mundbreiten bzw. Mundhöhen beider Videofilme. Die Bezeichnung *Fehler* mag hier etwas unglücklich gewählt sein; damit ist der Unterschied zwischen der vom Referenzsystem und der von unserem System gelieferten Höhe bzw. Breite gemeint. Der erste Datensatz besteht aus insgesamt 22 Aufnahmen von zwei verschiedenen Sprechern mit jeweils der Wortsequenz von Null bis Zehn. Der zweite Datensatz besteht ebenfalls aus 22 Aufnahmen mit den gleichen Sprechern und der gleichen Wortsequenz. Mit dem DTW-Algorithmus werden zwei dynamische Zeitfolgen aneinander angepasst; dies führt jedoch nicht zu einer 100%-Anpassung, d.h. selbst wenn die beiden Folgen aus dem gleichen Datensatz stammen (z.B. beide aus dem Datensatz mit den blaumarkierten Lippen), werden w_{Fehler} und h_{Fehler} nicht zu Null. Aus diesem Grund werden hier die Begriffe „inter-Fehler“ und „intra-Fehler“ eingeführt. Mit „inter-Fehler“ ist der Fehler gemeint, der nach dem Anpassen zweier Zeitfolgen **verschiedener** Datensätze entsteht. Mit „intra-Fehler“ ist der Fehler gemeint, der nach dem Anpassen zweier Zeitfolgen aus dem **gleichen** Datensatz entsteht. Die folgende Tabelle fasst die Ergebnisse der Fehler nach Gl. 3.22 zusammen. Die Mundhöhe schwankt zwischen 100 und 170 Pixel und weist somit eine größere Schwankung als die von der Mundbreite (zwischen 240 und 290 Pixel) auf.

	intra-Fehler	inter-Fehler
Breitenfehler	0.96%	3.25%
Höhenfehler	2.62%	6.23%

Um für eine genauere Statistik zu sorgen, wurde die *Round Robin* Methode verwendet. Die relativ geringe Abweichung zwischen unserem System und dem Referenzsystem beweist die gute Leistung des Systems.

3.5 Zusammenfassung

Mit dem *Snake*-Algorithmus kann eine genaue Beschreibung der Lippenkurven erzielt werden. Dabei müssen aber eine hohe Rechenzeit und Komplexität in Kauf genommen werden. Mit dem *Active Contour Model* können die Lippenformen dagegen mit einer geringeren Anzahl von Parametern repräsentiert werden. Für die Beurteilung der Kantendetektion der Lippen wurde eine kommerzielle Laborausrüstung eingesetzt.

Kapitel 4

Merkmalsextraktion aus dem Audiokanal

In diesem Kapitel werden mehrere Algorithmen zur Merkmalsextraktion aus dem Sprachkanal vorgestellt. Die erste Datenbank, mit der gearbeitet wurde, besteht aus 5 deutschen Vokalen, die von mehreren Sprechern sowohl akustisch als auch visuell aufgenommen wurden. Dabei eignet sich die in Abschnitt 4.1 vorgestellte LPC-Analyse, um die Formanten der Vokale zu schätzen. Die geschätzten Formanten können dann als Merkmale verwendet werden.
Die Melfrequenz-Cepstral-Koeffizienten (MFCK) (Abschnitt 4.3) sind vor allem in Kombination mit einem HMM-Erkenner weitverbreitet und liefern robuste Merkmale für die Einzelworterkennung aus dem zweiten Datensatz. Vor der Merkmalsextraktion wird das Sprachsignal der in Abb. 4.1 dargestellten Signalvorverarbeitung unterzogen. Nach der 16-Bit-Digitalisierung mit einer Abtastfrequenz von $f_A = 8000$Hz durchläuft

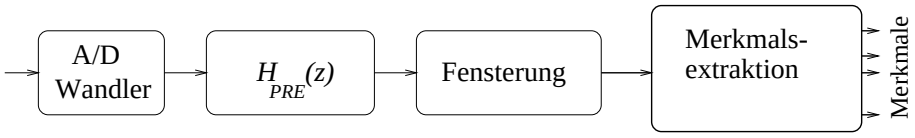


Abbildung 4.1: *Signalvorverarbeitung*

das Signal eine Preemphase, Segmentierung, Fensterung und Merkmalsextraktion. Das Preemphase-Filter $H^{PRE}(z)$ soll die Tiefpasscharakteristik der Sprache ausgleichen, indem die hohen Frequenzanteile angehoben werden:

$$H_{PRE}(z) = 1 - \alpha z^{-1} \tag{4.1}$$

Der Wert α liegt typischerweise zwischen 0,8 bis 0,95. Nach der Preemphase wird das Signal in überlappende Segmente zu jeweils N Abtastwerten zusammengefasst. Die Abtastwerte eines Segments werden schließlich mit einer Hamming-Fensterfunktion gewichtet [Kro89]:

$$Fen(i) = 0,54 - 0,46 \cos\left(\frac{2\pi i}{N-1}\right) \qquad 0 \leq i \leq N-1 \quad . \tag{4.2}$$

i ist der auf das Segment bezogene Index der Abtastwerte.

4.1 LPC-Analyse

LPC steht für *Linear Predictive Coding* (zu dt.: „Lineare Vorhersage“). Es handelt sich um eine Analyse, bei der ein Abtastwert mit Hilfe von vergangenen Werten geschätzt wird. Die Vokaltraktübertragungsfunktion $H(z)$ erzeugt das Sprachsignal $s(n)$ durch eine Anregung $x(n)$. Ganz allgemein gilt:

$$H(z) = \frac{S(z)}{X(z)} = \frac{\sum_{j=0}^{q} b(j) z^{-j}}{\sum_{i=0}^{p} a(i) z^{-i}} \quad . \tag{4.3}$$

Die lineare Vorhersage lautet:

$$\hat{s}(n) = \sum_{j=0}^{q} b(j) x(n-j) - \sum_{i=1}^{p} a(i) s(n-i) \quad . \tag{4.4}$$

Man kann aber $H(z)$ vereinfachen und durch ein Allpol-Modell (autoregressives Modell) approximieren. Diese Vereinfachung hat zur Folge, dass manche Nasale nicht mehr mit diesem Modell erzeugt werden können. Gl. (4.4) reduziert sich zu:

$$\hat{s}(n) = -\sum_{i=1}^{p} a(i) s(n-i) \tag{4.5}$$

und für den Prädiktionsfehler $e(n)$ gilt:

$$e(n) = s(n) - \hat{s}(n) = \sum_{i=0}^{p} a(i) s(n-i) \tag{4.6}$$

mit $a(0) = 1$.

Die besten Koeffizienten $a(i)$ gewinnt man aus der Minimierung der Leistung des Prädiktionsfehlers $E\{\|e(n)\|^2\}$. Hier wird das Orthogonalitätsprinzip verwendet. Die gesuchten Koeffizienten sind demnach solche, bei denen der Fehler und die Abtastwerte $s(n-1), s(n-2), \cdots, s(n-p)$ orthogonal zueinander sind:

$$E\{s(n-j)e(n)\} = 0, \quad \forall j = 1, 2, ...p \tag{4.7}$$

$$E\{s(n-j)\sum_{i=0}^{p} a(i) s(n-i)\} = 0 \tag{4.8}$$

Vertauscht man die Erwartungsbildung und die Summe, und ersetzt man die Erwartungsbildung durch die Summe über n, erhält man:

$$\sum_{i=0}^{p} a(i) \sum_{n} s(n-i)s(n-j) = 0, \quad j = 1, ..., p \tag{4.9}$$

Die Prädiktionskoeffizienten können mit Gl. 4.9 bestimmt werden.
Das Orthogonalitätsprinzip besagt auch, dass das resultierende Fehlerminimum durch

$$s_{ee} = E\{e^2(n)\} = \{s(n)e(n)\} \tag{4.10}$$

gegeben wird.

Analog zur Herleitung von Gl. 4.9 erhält man für s_{ee}:

$$\sum_{i=0}^{p} a(i) \sum_{n} s(n-i)s(n) = s_{ee} \quad . \tag{4.11}$$

Mit der Minimierung des Fehlers über die gesamte Zeit erhält man:

$$\sum_{i=0}^{p} a(i)s_{ss}(i-j) = 0, \quad j = 1, 2, ..., p \quad ; \tag{4.12}$$

$s_{ss}(i)$ ist die Autokorrelationsfunktion des Signals $s(n)$, die als

$$s_{ss}(i) = \sum_{n=-\infty}^{\infty} s(n)s(n-i) \tag{4.13}$$

definiert ist.

Aus Gl. 4.9 erhält man mit $s_{ss}(-i) = s_{ss}(i)$ und $a(0) = 1$:

$$\begin{bmatrix} s_{ss}(1) & s_{ss}(0) & s_{ss}(1) & \cdots & s_{ss}(p-1) \\ s_{ss}(2) & s_{ss}(1) & s_{ss}(0) & \cdots & s_{ss}(p-2) \\ s_{ss}(3) & s_{ss}(2) & s_{ss}(1) & \cdots & s_{ss}(p-3) \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_{ss}(p) & s_{ss}(p-1) & s_{ss}(p-2) & \cdots & s_{ss}(0) \end{bmatrix} \begin{bmatrix} 1 \\ a(1) \\ a(2) \\ \vdots \\ a(p) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.14}$$

mit s_{ss}= Autokorrelationsfolge des Signals.

Zur Lösung des LPC-Problems bieten sich prinzipiell drei Möglichkeiten an:

- *Autokorrelations-Methode*: Dabei wird das Sprachsignal $s(n)$ außerhalb des Messintervalls $0 \leq n \leq N-1$ zu Null gesetzt, wodurch man das wirkliche Signal und somit auch dessen Autokorrelationsfunktion verfälscht.

- *Kovarianz-Methode*: Das Signal wird außerhalb des Messintervalls als unbestimmt betrachtet, was zwar den Fehler der Autokorrelations-Methode umgeht, aber dafür zu instabilen Synthesefiltern führen kann.

- *Burg-Algorithmus*: Die Nachteile der beiden oben genannten Methoden vermag der Burg-Algorithmus zu umgehen, indem er zuerst die PARCOR-Koeffizienten [Kro89] durch gleichzeitige Minimierung des „Vorwärts“- und des „Rückwertsprädiktionsfehlers“ schätzt. Man erhält so zunächst eine „Lattice-Filterstruktur“, aus welcher die „Levinson-Durbin-Rekursion“ auf die gewünschten Koeffizienten des Nennerpolynoms im autoregressiven Modell führt. Auf die Herleitung der Rekursionsformeln der Burg-Analyse [Kro89] wird hier verzichtet.

4.2 Die Bestimmung der Formantenfrequenzen

Vokale sind stimmhafte Phoneme. Typisch für die stimmhaften Phoneme sind die so genannten Formantenfrequenzen. Diese sind Resonanzstellen der Übertragungsfunktion des Vokaltraktes. Ihre Bestimmung ist für die Erkennung der Vokale sehr hilfreich. Die Bezeichnung der Formantenordnung erfolgt in aufsteigender Reihenfolge nach der Frequenz (s. Abb. 4.2). Am besten zieht man für die Extraktion der Formanten

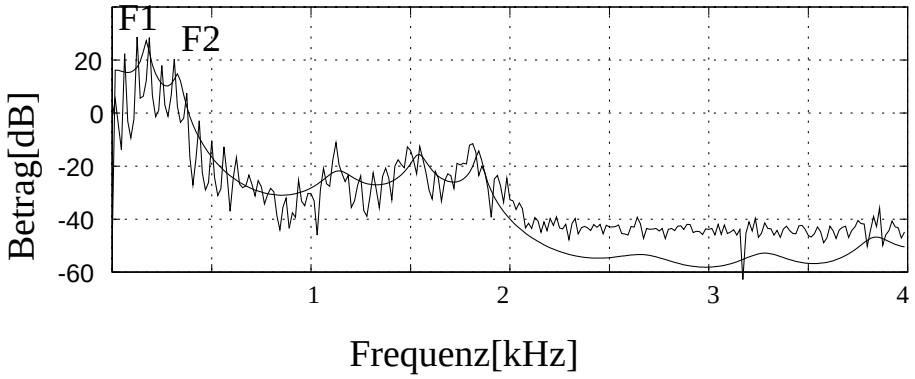


Abbildung 4.2: *Vergleich FFT-Betragsspektrum mit LPC-Spektrum (Burg-Analyse, LPC-Ordnung=20) anhand des Vokals /o/. Der glatte Kurvenverlauf resultiert aus dem LPC-Spektrum.*

das LPC-Spektrum heran, da es die Vokaltraktübertragungsfunktion mit einem Allpol modelliert, was erstens für die meisten Vokale den realen Gegebenheiten sehr nahe kommt und zweitens schärfer ausgeprägte Maxima bietet.

Aus 4.3 gilt mit der Annahme des AR-Modells (*allpole model*) für die Vokaltraktübertragungsfunktion:

$$H(z) = \frac{1}{\sum_{i=0}^{p} a(i) z^{-i}} = \frac{1}{A(z)} \quad . \tag{4.15}$$

Das Spektrum ist der Betrag von $H(z)$ und kann durch

$$|H(\omega)| = \left| \frac{1}{A(z)} \right|_{z=e^{j\omega}} \tag{4.16}$$

bestimmt werden. $A(z)$ wurde durch p-LP-Koeffizienten approximiert:

$$A(z) = \sum_{i=0}^{p} a(i) z^{-i} \tag{4.17}$$

Aus 4.16 und 4.17 kann der kontinuierliche Betrag des Spektrums $|H(\omega)|$ für alle möglichen Werte von $\omega = 0 \ldots 2\pi$ bestimmt werden. Da das LPC-Spektrum einen glatten Verlauf im Vergleich zum FFT-Spektrum aufweist (s. Abb. 4.2), kann man

das Betragsspektrum mit wenigen N-Stützstellen diskretisieren. In unserer Arbeit war $N = 128$. Daraus folgt für das diskrete Betragsspektrum $|H(k)|$:

$$|H(k)| = \left| \frac{1}{A(e^{-j2\pi k/N})} \right| \quad , \; k = 0, 1, \ldots, N \tag{4.18}$$
$$= \frac{1}{\left| \sum_{i=0}^{p} a(i) e^{-j2\pi ik/N} \right|}$$

Da $z = e^{j\omega}$ einen Einheitskreis in der komplexen z-Ebene darstellt, befinden sich die N-Stützstellen auf diesem Kreis.
Aus dem LPC-Spektrum werden die ersten drei Maxima gesucht und den ersten drei Formanten zugeordnet.

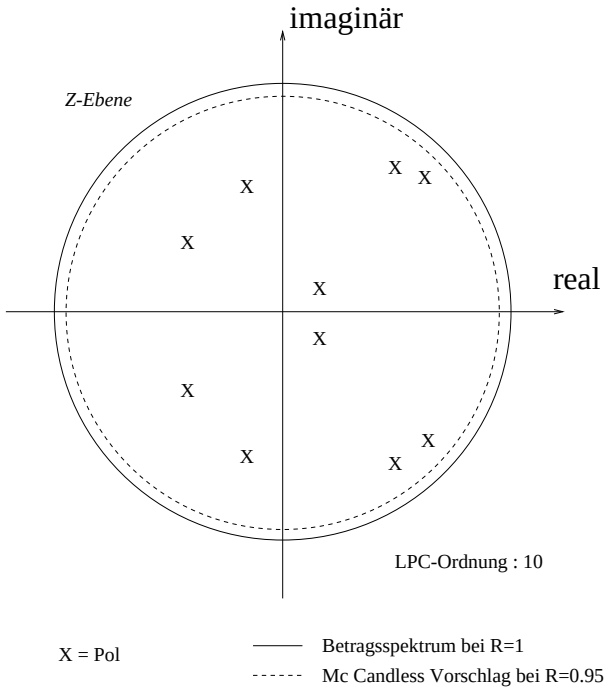


Abbildung 4.3: *Pole des Betragsspektrums*

Die ersten beiden Formanten sind als Merkmale hervorragend geeignet, da sie unmittelbar mit der Zungenlage korrelieren [Schmi89, Fri92, Fur89], wobei sich die besondere Bedeutung des zweiten Formanten in seiner großen Varianz von Vokal zu Vokal offenbart. Auch der dritte Formant spielt für die Unterscheidung von Vokalen eine Rolle, während höhere Formantanordnungen sprecherabhängig stark schwanken, aber sprecherspezifisch stabil sind [Schmi89, Hil95]. Bei der Zuordnung der Maxima aus dem LPC-Spektrum zu den Formantenfrequenzen kann es vorkommen, dass zwei benachbarte Formanten miteinander verschmelzen. Die Ursache dafür sind zwei eng aneinander liegende Pole. Eine Erhöhung der Auflösung des Spektrums (die Anzahl der Stützstellen) hebt dieses Problem nur in seltenen Fällen auf. Eine heuristische Abhilfe wurde in [McC74] vorgeschlagen. Demnach wird das Betragsspektrum auf einem kleineren Kreis als dem Einheitskreis berechnet (s. Abb. 4.3).

4.3 Melfrequenz-Cepstral-Koeffizienten (MFCK)

Zur Bestimmung der Melfrequenz-Cepstral-Koeffizienten (engl.: MFCC) [Dav78, Kel91, Kob94] wird das Audiosignal zunächst in den Frequenzbereich transformiert. Dabei kommt die schnelle Fourier-Transformation (FFT) zum Einsatz:

$$S(k) = \mathcal{FFT}\{s(n)\} \tag{4.19}$$

Aus den $N = 256$ Abtastwerten erhält man im Frequenzbereich ein Spektrum, das sich von $-N/2$ bis $N/2$ erstreckt. Wegen der Symmetrie hat $S(k)$ effektiv eine Länge von $N/2$ Werten (von 0 bis $(N/2) - 1$). Das Signal wird dann über eine Filterbank mit 20 sogenannten Mel-Filtern geleitet. Die Mel-Filter sind in Abb. 4.4 dargestellt.

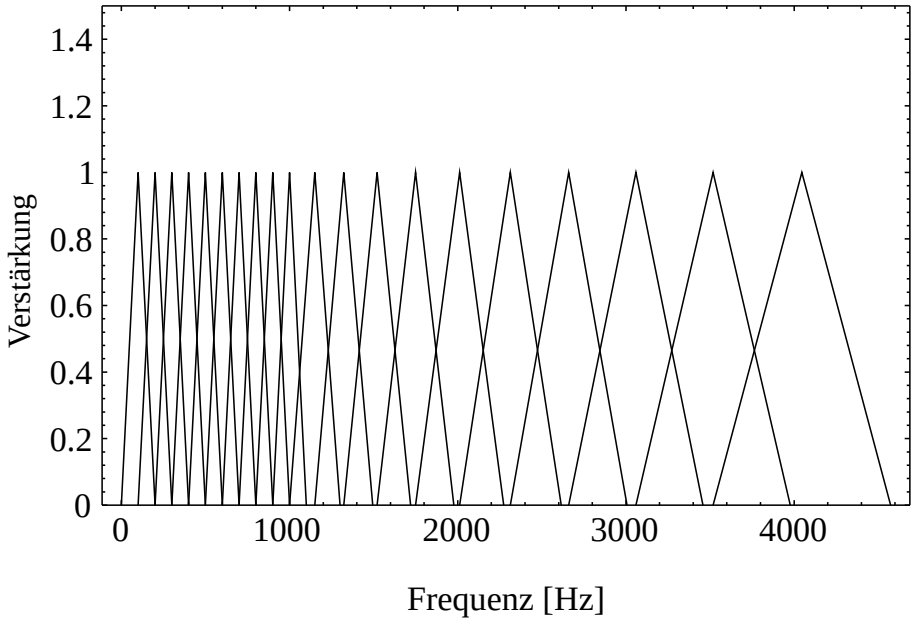


Abbildung 4.4: *Die Mel-Filter*

Dabei handelt es sich um Dreiecks-Filter, die mit ihren Mittenfrequenzen bis zu einem Bereich von 1kHz in linear zunehmendem und darüber in logarithmisch zunehmendem Abstand angeordnet sind. Benachbarte Filter überlappen sich in ihrem Frequenzgang (in Abbildung 4.4 ist die Überlappung = 50%). Diese Anordnung berücksichtigt die Physiologie des Gehörs.

Beschreibt man den Frequenzgang des j-ten Filters mit $H^{(j)}_{MEL}(k)$, wobei k die diskrete Frequenz darstellt, so erhält man an jedem Filterausgang jeweils eine sog. Mel-Energie:

$$E^{(j)}_{MEL}(k) = \sum_{k=0}^{N/2-1} |H^{(j)}_{MEL}(k)|^2 |S(k)|^2, \quad 1 \leq j \leq 20. \tag{4.20}$$

Die M MFC-Koeffizienten $c^{(u)}_{MFC}$ erhält man mittels einer modifizierten Cosinus-

Transformation aus den logarithmierten Mel-Energien:

$$c_{MFC}^{(u)} = \sum_{j=1}^{20} \log E_{MEL}^{(j)}(k) \cos\left[u(j-0,5)\frac{\pi}{20}\right], \qquad 1 \leq u \leq M. \tag{4.21}$$

Diese Koeffizienten werden als Merkmale für den Audiokanal bei der Einzelworterkennung verwendet. In der Praxis sind die ersten 8 bis 12 Koeffizienten völlig ausreichend für eine gute Erkennung.

4.4 Zusammenfassung

Dieses Kapitel wurde der Audioverarbeitung gewidmet. Da die Merkmalsextraktion aus dem Audiokanal an sich kein Schwerpunkt dieser Arbeit ist, wurden hier nur zwei auf dem Gebiet der Sprachverarbeitung bekannte Verfahren kurz dargestellt. Auf eine genaue Beschreibung der Sprachproduktion beim Menschen und die Hintergründe der vorgestellten Algorithmen wurde daher verzichtet. Für den interessierten Leser kann an dieser Stelle auf richtungweisende Literatur auf diesem Gebiet, wie [Fur89, Kro89, Rab78], verwiesen werden.
Für die Realisierung der LPC-Analyse wurde der Burg-Algorithmus realisiert. Die LP-Koeffizienten können direkt als Merkmale für die Erkennung der Vokale verwendet werden. Alternativ dazu können die Formantenfrequenzen, die man im LPC-Spektrum lokalisieren kann, verwendet werden.
Die Melfrequenz-Cepstral-Koeffizienten eignen sich ebenfalls als Merkmale. Die dabei verwendeten Mel-Filter sind dem psychoakustischen Gehörsystem des Menschen nachempfunden.

Kapitel 5

Erkennung anhand der Audio- oder Videodaten

In diesem Kapitel werden zwei voneinander unabhängige Erkennungssysteme vorgestellt. Das erste System setzt nur die aus dem Audiokanal berechneten Merkmale zur Erkennung der gesprochenen Vokale bzw. Wörter ein. Das zweite System verwendet dagegen nur die Videomerkmale für diese Erkennungsaufgabe. Es wird dann später in Kapitel 6 ein Erkennungssystem vorgeschlagen, das sowohl die Audio- als auch die Videomerkmale zur Erkennung der Vokale verwendet.

Es werden in diesem Kapitel zunächst zwei Algorithmen kurz erläutert, die bei der Erkennung eingesetzt werden:

- **Der DTW-Algorithmus** (Abschnitt 5.1): Dieser Algorithmus wird in unserem System beim audiobasierten Erkenner von Vokalen eingesetzt.
- **Die HM-Modelle** (Abschnitt 5.2): Diese werden in unserem Fall sowohl für die videobasierte als auch für die audiobasierte Erkennung von Einzelwörtern benötigt.

5.1 Dynamic Time Warping (DTW)

DTW ist ein Verfahren, das häufig bei Spracherkennung eingesetzt wird. Mit diesem Verfahren lässt sich ein Testvektor mit einer Reihe von Referenzvektoren vergleichen [Sak78]. In der Regel handelt es sich bei diesen Vektoren um zeitliche Merkmale. Es sind also Merkmale wie z.B. LPC oder Cepstrum, die zeitlich gewonnen werden. Der Testvektor wird der Wortklasse zugeordnet, deren Referenzvektor dem Testvektor am *ähnlichsten* ist. Die Tatsache, dass die Länge der Referenzvektoren untereinander sowie die des Testvektors unterschiedlich ist, stellt ein bekanntes Problem bei der Spracherkennung dar. Eine einfache Lösung wäre, die Längen aller Vektoren linear zu normalisieren, sodass Referenzvektor und Testvektor die gleiche *zeitliche* Länge besitzen. Dies führt aber zu schlechten Ergebnissen. Mit dem DTW-Algorithmus wird dagegen beim „Dehnen“ bzw. „Stauchen“ des Testvektors der dynamische Verlauf berücksichtigt.

Nehmen wir an, ein Referenzvektor $\mathbf{x} = (x_1, x_2, \cdots, x_i, \cdots, x_I)$ ist mit einem Testvektor $\mathbf{y} = (y_1, y_2, \cdots, y_j, \cdots, y_J)$ zu vergleichen. Es wird eine Abstandsmatrix $d(\mathbf{x}_i, \mathbf{y}_j)$ der Dimension $I \cdot J$ berechnet. Der Punkt (i, j) dieser Matrix repräsentiert den Abstand zwischen x_i und y_j. Verwendet man z.B. den Euklidischen Abstand, dann kann man die Elemente der Matrix wie folgt definieren:

$$d(i, j) = (x_i - y_j)^2 \tag{5.1}$$

Das Ziel des DTW-Algorithmus ist es, den besten Pfad L zwischen $i = 1, j = 1$ und $i = I, j = J$ zu bestimmen. Dies entspricht einer zeitlichen Anpassung der Zeitfolge $\mathbf{y}$ an die Zeitfolge $\mathbf{x}$. Abb. 5.1 zeigt ein Beispiel für eine solche Anpassung.

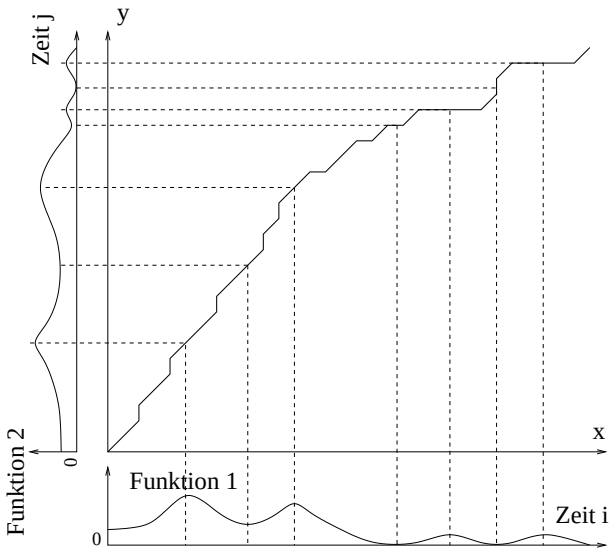


Abbildung 5.1: *Beispiel für den DTW-Algorithmus*

Hierbei wird nicht jeder Punkt i des Referenzvektors mit dem Punkt $j = i$ des Testvektors verglichen, sondern einem Punkt i können auch mehrere Punkte j des Testvektors zugeordnet werden. Somit entsteht eine nichtlineare Abbildung $j = (wr(i))$ mit einer Zuordnung ähnlicher Punkte beider Vektoren.

Die Aufgabe besteht nun darin, aus der Vielzahl der möglichen Pfade unterschiedlicher Länge L den optimalen Pfad zu bestimmen. Dazu wird für jeden möglichen Pfad die Summe aller Abstände entlang dieses Pfads $wr(l)$ berechnet:

$$D = \sum_{i=1}^{I} d(\mathbf{x}_i, \mathbf{y}_{wr(i)}) \tag{5.2}$$

Für den optimalen Pfad gilt:

$$D_{min} = \min_{wr} \{ \sum_{i=1}^{I} d(\mathbf{x}_i, \mathbf{y}_{wr(i)}) \} \tag{5.3}$$

Die beste Pfadsuche arbeitet nach dem Prinzip der dynamischen Programmierung und macht Gebrauch von der Tatsache, dass der Pfad nicht monoton fallend verläuft

und keine Sprünge aufweist. Das Entscheidungsproblem wird in eine rekursive Folge einfacherer Entscheidungen aufgeteilt. Schrittweise wird rückwärts ein „Stück“ des besten Pfads bis zum Punkt (i, j) bestimmt. Dabei wird Gebrauch davon gemacht, dass bis dahin auch die an vorhergehenden Punkten getroffenen Entscheidungen optimal sein müssen.

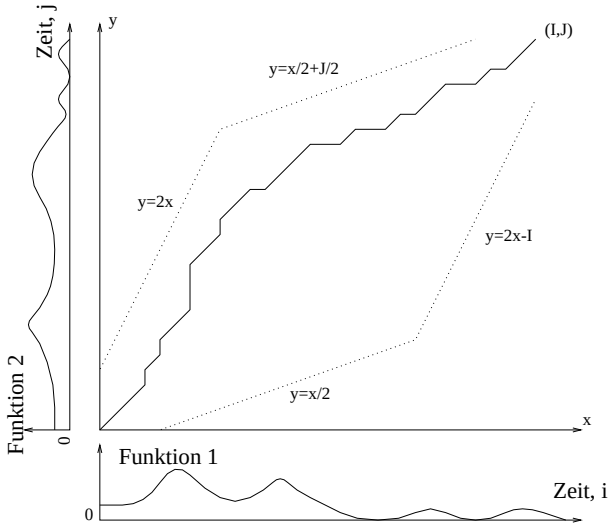


Abbildung 5.2: *Die Beschränkung der Suchregion durch ein Parallelogramm*

In [Sak78] wurde ein einfacher Algorithmus skizziert. Man beginnt beim Punkt (I, J) und versucht, den optimalen Pfad zurückzuverfolgen. Von jedem Punkt (i, j) aus sind nur folgende vorhergehende Punkte erlaubt:

- Punkt $(i - 1, j)$; entspricht einem horizontalen Schritt,
- Punkt $(i - 1, j - 1)$; entspricht einem diagonalen Schritt,
- und Punkt $(i, j - 1)$; entspricht einem vertikalen Schritt.

Gl. 5.3 wird schrittweise berechnet. Zusätzlich werden horizontale bzw. vertikale Entscheidungen gegenüber diagonalen Entscheidungen mit einem zusätzlichen Faktor w „bestraft“:

$$D_{min,(i,j)} = \min\{d(\mathbf{x}_{i-1}, \mathbf{y}_j) + w, d(\mathbf{x}_{i-1}, \mathbf{y}_{j-1}), d(\mathbf{x}_i, \mathbf{y}_{j+1}) + w\} \qquad (5.4)$$

Bei jedem Punkt wird der beste vorhergehende Punkt anhand der Abstandsdifferenz bestimmt. Man „merkt“ sich diese Entscheidung, da sie einen Teil des optimalen Pfads darstellt.

Bei der Suche nach dem optimalen Pfad wird nicht nach dem optimalen Pfad in der gesamten Abstandsmatrix gesucht, sondern man gibt sich einen Bereich vor, in dem dieser Pfad verlaufen muss (Abb. 5.2).

Dies hat mehrere Vorteile. Zum einen wird die Berechnung beschleunigt und der Speicherbedarf wird verringert. Zum anderen beschränkt man so die maximale Dehnung, die bei der Anpassung entsteht, denn die zeitliche Verzerrung kann nicht beliebig groß sein.

Bei einem Erkennungsproblem wird D_{min} zwischen dem Testvektor und allen Referenzvektoren bestimmt. Der Testvektor wird dann der Klasse zugeordnet, zu der er die kleinste D_{min} hatte. Der DTW-Algorithmus kann auch dazu verwendet werden, um eine zeitliche Folge an eine andere anzupassen. Von einer solchen Anpassung wurde bereits in Abschnitt 3.4 Gebrauch gemacht.

5.2 Hidden-Markov-Modelle

Bei der Spracherkennung stehen Lerndaten zur Verfügung, mit denen man Referenzvektoren erstellen kann. Mit Hilfe von Verfahren wie z.B. DTW kann dann der Referenzvektor bestimmt werden, der einem neuen Testvektor am ähnlichsten ist.
Das Hidden-Markov-Modell (HMM) umgeht diese zeitliche Anpassung. Es handelt sich um ein stochastisches Modell, das ein Wort mit zwei Zufallsprozessen beschreibt. Der erste Zufallsprozess beschreibt den sog. Zustand des Modells. Ein Wort befindet sich zu einem Taktzeitpunkt l in einem Zustand S_i von insgesamt N Zuständen $\{S_1, S_2, S_3, \ldots, S_N\}$. Die Zustände umfassen Zeitabschnitte mit einem oder mehreren aufeinanderfolgenden Merkmalsvektoren.
In dieser Arbeit wird angenommen, dass der Übergang vom Zustand $q_{l-1} = S_i$ nach $q_l = S_j$ nur vom momentanen und vorherigen Zustand abhängig ist und wird durch die Wahrscheinlichkeit

$$a_{ij} = P(q_l = S_j | q_{l-1} = S_i) \quad 1 \leq i \leq N, \quad 1 \leq j \leq N \tag{5.5}$$

festgelegt. Der Anfangszustand des Systems zum Zeitpunkt $l = 1$ wird durch die Anfangswahrscheinlichkeiten

$$\pi_i = P(q_1 = S_i), \quad i = 1, \ldots, N \tag{5.6}$$

bestimmt. Die Zuordnung eines Zustandes zu einer Gruppe von Merkmalsvektoren wird vom Modell beschrieben und bleibt „versteckt“. Offen ist dagegen der zweite Zufallsprozess, der auch daher Beobachtungsfolge (oder auch O-Folge) genannt wird: Zu jedem Taktzeitpunkt l wird –nachdem der aktuelle Zustand S_i festliegt– ein Symbolvektor o_l erzeugt.

Die Symbole stammen hier aus einer kontinuierlichen Menge V von Symbolvektoren $\mathbf{v}$ der Dimension m. Das Auftreten eines bestimmten Wertes $\mathbf{v}$ wird durch eine mehrdimensionale Wahrscheinlichkeitsdichtefunktion

$$f_i(\mathbf{v}) = f(\mathbf{o}_l = \mathbf{v} | q_l = S_i) \tag{5.7}$$

charakterisiert, die vom aktuellen Zustand S_i abhängt.

Die durch Merkmalsextraktion gefundene Merkmalsvektorfolge stellt hier die zu beschreibende Beobachtungsfolge dar. Zur Beschreibung der statistischen Eigenschaften der Zustandsfolge sind sowohl alle Wahrscheinlichkeiten a_{ij} nach Gl. 5.5 als auch die Anfangswahrscheinlichkeiten π_i nach Gl. 5.6 notwendig. Zur kompakten Notation werden alle a_{ij} in einer Matrix $\mathbf{A} = (a_{ij})$ der Dimension $N \times N$ zusammengefasst, die als Übergangsmatrix bezeichnet wird. Alle Anfangswahrscheinlichkeiten werden in dem $N \times 1$ Vektor $\boldsymbol{\pi} = (\pi_i)$ zusammengefasst.
Nimmt die Übergangsmatrix $\mathbf{A}$ die Form

$$\mathbf{A} = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & a_{23} & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & 0 & a_{N-2,N-2} & a_{N-2,N-1} & 0 \\ \vdots & \ddots & 0 & 0 & a_{N-1,N-1} & a_{N-1,N} \\ 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.8}$$

an, so handelt es sich um den in der Praxis häufig verwendeten Spezialfall des Links-zu-Rechts-HMMs, bei dem die Zustände von einem Anfangs- bis zu einem Endzustand nur in einer vorher festgelegten Reihenfolge durchlaufen werden können. Von einem Zustand S_i aus kann nur S_i selbst oder der nächste Zustand S_{i+1} erreicht werden. Ist das Modell im letzten Zustand S_N angelangt, kann es diesen nicht mehr verlassen. Der Vektor $\boldsymbol{\pi}$ nimmt in diesem Fall die Form

$$\boldsymbol{\pi} = (\pi_i) = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{5.9}$$

an.

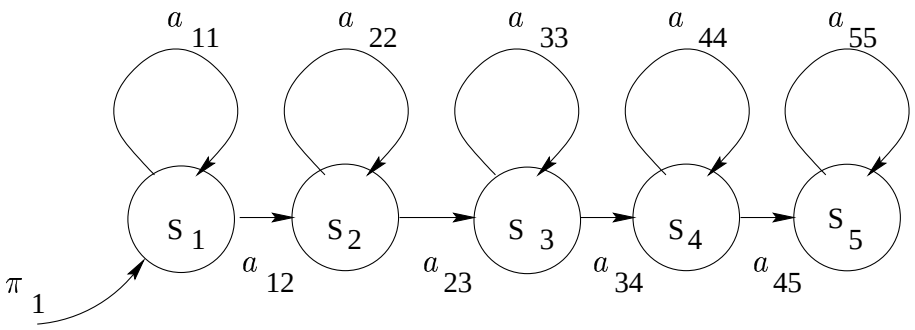


Abbildung 5.3: *Ein Links-zu-Rechts-HMM mit 5 Zuständen*

Abb. 5.3 zeigt ein Beispiel für ein solches Links-zu-Rechts-HMM mit $N = 5$ Zuständen.
Die Wahrscheinlichkeitsdichten (WD) in den Zuständen werden häufig mit einer Linearkombination von Gaußdichten approximiert, mit der sich beliebige Dichtefunktionen ausreichend genau darstellen lassen [Rab89].

Die Symboldichten $f_i(\mathbf{v})$ für den Zustand S_i werden aus jeweils M Gaußdichten $\mathcal{N}$ der Dimension m gebildet, wobei die k-te Dichte des Zustandes S_i mit einem Faktor c_{ik} gewichtet wird. Jede Gaußdichte wird durch einen eigenen $N \times 1$ Mittelwertsvektor $\boldsymbol{\mu}_{ik}$ und eine eigene $N \times m$ Kovarianzmatrix $\boldsymbol{\Sigma}_{ik}$ festgelegt:

$$\begin{aligned} f_i(\mathbf{v}) &= f(\mathbf{o}_l = \mathbf{v}|q_l = S_i) = \sum_{k=1}^{M} c_{ik}\mathcal{N}(\mathbf{v}, \boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}) \\ &= \sum_{k=1}^{M} c_{ik} \frac{1}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}_{ik}|}} \cdot \exp\left[-\frac{1}{2}(\mathbf{v} - \boldsymbol{\mu}_{ik})^T \boldsymbol{\Sigma}_{ik}^{-1}(\mathbf{v} - \boldsymbol{\mu}_{ik})\right] \end{aligned} \tag{5.10}$$

Ein kontinuierliches Hidden-Markov-Modell mit N Zuständen wird somit mit dem Parametersatz $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{C}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})$ vollständig beschrieben.

5.2.1 HMM-Lernphase

Die Aufgabe bei der Lernphase besteht darin, die O-Folgen, also die zeitlich gewonnenen Lernmerkmale, zu verwenden, um den Parametersatz $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{C}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})$ zu bestimmen. Der Baum-Welch-Algorithmus (BWA) [Rab89] passt iterativ die Parameter eines kontinuierlichen HMM an eine Menge von L O-Folgen $\mathcal{O} = \{\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \ldots, \mathbf{O}^{(L)}\}$ an. Ausgehend von einem Satz von Anfangsparametern $\boldsymbol{\lambda}_{Anfang}$ findet der BWA ein Modell $\boldsymbol{\lambda}$, indem er iterativ gegen ein lokales Maximum der WD $f(\mathcal{O}|\boldsymbol{\lambda})$ konvergiert, wobei zusätzlich die Bedingung

$$f(\mathcal{O}|\boldsymbol{\lambda}_{alt}) \leq f(\mathcal{O}|\boldsymbol{\lambda}_{neu}) \tag{5.11}$$

eingehalten wird. Für eine ausführliche Beschreibung des BWA wird auf [Rab89] und [Kob94] verwiesen.

5.2.2 HMM-Erkennungsphase

Nachdem in einer Lernphase zu jedem Wort ein HMM $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{C}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi})$ als Referenz abgelegt wurde, ist nun in der Erkennungsphase folgendes Problem zu lösen: Welche Zustandsfolge Q^* kann am besten an die gegebene Menge $\mathcal{O}$ der Beobachtungsfolgen (die Sequenz der Merkmalsvektoren aus dem Testwort) angepasst werden? Dabei gilt, dass die Zustandsfolge Q^* die WD $f(\mathcal{O}|\boldsymbol{\lambda})$ maximieren muss. Die Lösung dieser Optimierungsaufgabe leistet der Viterbi-Algorithmus mit Methoden der dynamischen Programmierung. Eine ausführliche Beschreibung dieses Algorithmus kann man [Hua90] , [Rab89] entnehmen.

5.3 Aufbau der Datensätze

Die Erkennungsleistung wurde anhand zweier verschiedener Datensätze bestimmt, die sowohl akustisch als auch mit einer Videokamera aufgenommen wurden:

1. Der erste Datensatz besteht aus den 5 deutschen Vokalen **/a/, /e/, /i/, /o/, /u/**. Diese Vokale sind von insgesamt 4 Sprechern (3 männlich, 1 weiblich) aufgenommen worden. Jede(r) Sprecher(in) hat jeden Vokal 6 mal wiederholt. Die Aufnahme erfolgte mit 3 Mikrofonen, sodass von jedem Vokal insgesamt $4 \cdot 6 \cdot 3 = 72$ Aufnahmen existieren. Für die Berechnung der Referenzvektoren des Klassifikators (bzw. für die Lernphase) wurden 48 Aufnahmen (12 Aufnahmen je Sprecher) verwendet, so dass 24 Aufnahmen pro Vokal für die Evaluierung zur Verfügung standen. Dieser Datensatz wird im Laufe dieser Arbeit mit dem Begriff „*statischer Datensatz*“ bezeichnet, da jede Aufnahme durch einen einzigen Merkmalsvektor repräsentiert werden kann.

2. Im Gegensatz zum ersten Datensatz wird dieser mit dem „*dynamischen Datensatz*“ bezeichnet. Er besteht aus 11 französischen Wörtern *(zéro, un, deux, trois, quatre, cinq, six, sept, huit, neuf, dix)*, die alle von einem Sprecher stammen. Die Wörter wurden insgesamt 10 mal wiederholt. Somit besteht diese Datenbank aus insgesamt 110 Wörtern. Für die Trainingsphase wurden davon 88 verwendet (die ersten 8 Wiederholungen). Die restlichen 22 Aufnahmen wurden für die Evaluierung der Erkennungsraten verwendet.

Die Erkennungsstrategie der beiden Datensätze unterscheidet sich. Beim statischen Datensatz wird angenommen, dass die aufeinanderfolgenden Segmente der Sprache sehr ähnliche Parameter besitzen. Idealerweise sind diese Parameter (z.B. die Formantenfrequenzen oder die LP-Koeffizienten) in jedem Segment identisch. Da dies in der Wirklichkeit nicht der Fall ist, wird mit Hilfe des DTW-Algorithmus versucht, aus den Lernaufnahmen einen einzigen Mustervektor pro Vokal zu bilden, der als Referenzvektor verwendet wird (eine nähere Beschreibung erfolgt in Abschnitt 5.4.1). Die dazugehörige Videoaufnahme kann auch als statisch angenommen werden. Daher kann theoretisch ein beliebiges Bild der Videosequenz als repräsentativ für die gesamte Aufnahme betrachtet werden.
Beim dynamischen Datensatz ist dies natürlich nicht möglich. Für die audiobasierte Erkennung wurden daher die Melfrequenz-Cepstral-Koeffizienten für die Merkmalsextraktion verwendet (s. 4.3). Für jedes Segment eines Wortes ensteht auf diese Weise eine Merkmalsmatrix der Dimension $N \times M$, mit $N =$ Anzahl der MFCK und $M =$ Anzahl der Segmente pro Wort. Da die Wörter unterschiedlich lang sind, ist eine Referenzvektorberechnung wie beim statischen Datensatz nicht möglich. Daher wird hier das in Abschnitt 5.2 beschriebene HM-Modell verwendet. Dementsprechend sind die Videosequenzen der einzelnen Wörter auch unterschiedlich lang. Für die videobasierte Erkennung müssen also alle Bilder einer Sequenz betrachtet werden. Hier wurden ebenfalls zur Erkennung HM-Modelle verwendet.

5.4 Erkennung anhand der Audiodaten

In diesem Abschnitt werden die Erkennungsergebnisse dargestellt, die man bei der Auswertung der Audiodaten allein erhält. Der Videokanal wird also nicht berücksichtigt. Im Abschnitt 5.5 wird dagegen nur der Videokanal betrachtet. Im nächsten Kapitel

sollen dann die Erkennungsergebnisse vorgestellt werden, die man bei der gleichzeitigen Auswertung des Video- und Audiokanals erhält. Auf diese Weise kann die Verbesserung der Erkennungsrate demonstriert werden.

5.4.1 Ergebnisse des statischen Datensatzes

Die Sprachsegmente durchlaufen eine LPC-Analyse der 12. Ordnung. Zusätzlich werden die Energieanteile in drei Bändern, die Tabelle 5.1 zu entnehmen sind, berechnet. Die Energiebereiche dieser Bänder entsprechen in etwa den Bereichen, in denen die

	von	bis
Tiefband	200 Hz	800 Hz
Mittelband	800 Hz	2250 Hz
Hochband	2250 Hz	3500 Hz

Tabelle 5.1: *Aufteilung der Energiebänder entlang der Frequenzachse*

ersten 3 Formanten liegen. Aus den 12 LP-Koeffizienten wird das Betragsspektrum, wie es in Abschnitt 4.2 beschrieben ist, berechnet. Aus dem Spektrum werden die ersten 3 Formantenfrequenzen lokalisiert. Somit ergeben sich insgesamt 6 Merkmale: 3 Formantenfrequenzen und 3 Energieterme. Man steht nun vor der Aufgabe, aus dem gesamten Lerndatensatz nur genau so viele Merkmalsvektoren zu erstellen wie die Anzahl der Klassen (in unserem Fall 5), die als Referenz bei der Vokalerkennung zu verwenden sind. Der einfachste Ansatz wäre, alle 6 Merkmale, die einem bestimmten Vokal entstammen, über alle Segmente und über alle Aufnahmen des Lerndatensatzes zu mitteln. Die auf diese Weise gewonnenen Referenzvektoren weisen aber hohe Abweichungen auf. Bessere Referenzvektoren erhält man durch folgende Vorgehensweise:

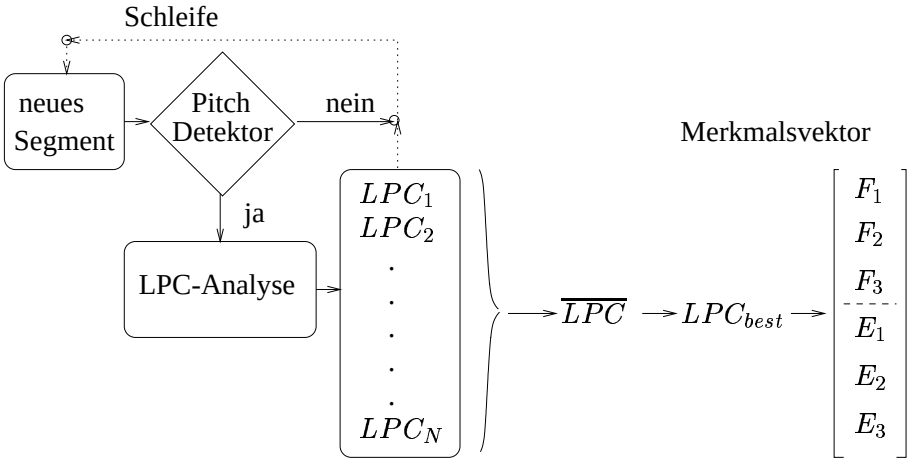


Abbildung 5.4: *Blockschaltbild für die Bestimmung der Mustervektoren*

1. Für alle Segmente aller Vokale des Lerndatensatzes werden die LP-Koeffizienten bestimmt.

2. Jedes Segment, das keine ausgeprägte Grundfrequenz besitzt, wird aussortiert und nicht mehr betrachtet.

3. Aus allen übriggebliebenen Segmenten werden pro Vokal die LP-Koeffizienten gemittelt. Dadurch erhält man für jede Klasse (=Vokal) i einen LPC-Parametersatz $\overline{LPC}_i$.

4. Aus allen Segmenten der Klasse i wird mit Hilfe des DTW-Algorithmus das beste Segment gesucht, dessen LP-Koeffizienten dem $\overline{LPC}_i$ Parametersatz am ähnlichsten sind.

5. Aus dem besten Segment werden die Merkmale (die 3 Formanten und die 3 Energieterme) berechnet und in einem Referenzvektor abgelegt.

	Formantenfrequenzen			Teilbandenergien		
	$\bar{F}_1$ / σ_{F_1}	$\bar{F}_2$ / σ_{F_2}	$\bar{F}_3$ / σ_{F_3}	$\bar{E}_1$ / σ_{E_1}	$\bar{E}_2$ / σ_{E_2}	$\bar{E}_3$ / σ_{E_3}
/a/	677 / 178	1381 / 564	2637 / 477	101 / 30	161 / 38	7 / 3
/e/	354 / 117	2224 / 334	1882 / 379	81 / 54	94 / 25	11 / 5
/i/	259 / 48	2171 / 614	3049 / 314	78 / 51	69 / 18	18 / 13
/o/	392 / 53	1379 / 828	2812 / 586	352 / 177	120 / 30	6 / 3
/u/	274 / 36	938 / 624	2412 / 514	374 / 146	109 /38	8 /5

Tabelle 5.2: *Referenzvektoren für alle Vokale*

Die so erhaltenen Referenzvektoren sind in Tabelle 5.2 dargestellt. In dieser Tabelle sind auch die Standardabweichungen jedes einzelnen Merkmals aufgeführt.
Für die Erkennungsaufgabe muss der unbekannte Vokal eine identische Analyse wie die der Lernphase durchlaufen: alle Segmente, die keine ausgeprägte Grundfrequenz aufweisen, werden ignoriert. Sonst werden für jedes Segment die 6 Merkmale bestimmt. Die Merkmale jedes Segments werden dann, unabhängig von den anderen Segmenten, auf ihre Ähnlichkeit zu allen Referenzvektoren untersucht. Für diesen Vergleich kann man z.B. das Euklidische Abstandsmaß oder den DTW-Algorithmus verwenden. Somit wird die Zugehörigkeit jedes Segments des unbekannten Vokals einer bestimmten Klasse zugeordnet. Der gesamte Vokal wird dann der Klasse zugeordnet, zu der die Mehrheit seiner Segmente zugeordnet worden sind. Die Erkennungsraten, die man durch diese Vorgehensweise erhält, sind in Tabelle 5.3 dargestellt. Zeilenweise sind die „tatsächlichen“ Klassen der Testvokale aufgelistet. Spaltenweise sind die Klassen, die vom Erkenner zugeordnet sind, eingetragen. Diese Darstellung der Erkennungsergebnisse ist sehr weit verbreitet und lässt uns auf einen Blick feststellen, welche Vokale vom Erkenner verwechselt werden. Daher werden auch solche Tabellen mit „Verwechslungstabellen“ bezeichnet.

	/a/	/e/	/i/	/o/	/u/
/a/	**95,83**	0,00	0,00	0,00	4,17
/e/	0,00	**100**	0,00	0,00	0,00
/i/	0,00	16,67	**79,17**	0,00	4,17
/o/	12,50	0,00	0,00	**87,50**	0,00
/u/	0,00	0,00	0,00	12,50	**87,50**

Tabelle 5.3: *Verwechslungstabelle beim DTW-Erkenner. Die Zahlen in der Tabelle stellen die Erkennungsrate in [%] dar*

So kann Tabelle 5.3 entnommen werden, dass 16,67% der getesteten /i/-Vokale fälschlicherweise als /e/ und 4,17% als /u/ erkannt, während alle anderen richtig als /i/ erkannt wurden. Die durchschnittliche Erkennungsrate aller Vokale beträgt 90,0%.
Alternativ zum DTW-Erkenner wurde auch ein einfaches Neuronales Netz mit einer Feedforward-Struktur verwendet. Neuronale Netze werden in Kapitel 6 näher betrachtet, da sie den Kernpunkt bei der Fusion von Audio- und Videodaten beim statischen Datensatz bilden. Bei der Erkennung mit Neuronalen Netzen wurde der unbekannte Testvokal durch einen einzigen Merkmalsvektor repräsentiert, der auf die gleiche Weise wie bei der Erstellung der Referenzvektoren bestimmt wurde. Tabelle 5.4 stellt die Verwechslungstabelle dar. Das verwendete Netz (Abbildung 5.5)besaß eine verdeckte Schicht mit 10 Einheiten.

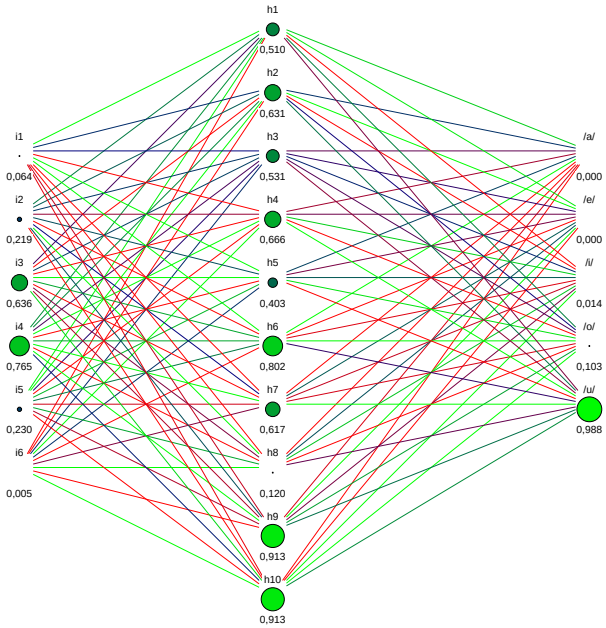


Abbildung 5.5: *Ein Beispiel für das trainierte Netz mit den 3 Formanten (i1 bis i3) und den 3 Teilenergien (i4 bis i6) als Merkmale während der Erkennung eines /u/-Vokals. Die Höhe der Aktivierung einer Einheit ist durch ihre Flächengröße dargestellt.*

	/a/	/e/	/i/	/o/	/u/
/a/	**100**	0,0	0,0	0,0	0,0
/e/	0,0	**91,7**	8,3	0,0	0,0
/i/	0,0	4,2	**95,8**	0,0	0,0
/o/	0,0	0,0	0,0	**100**	0,0
/u/	0,0	0,0	0,0	8,3	**91,7**

Tabelle 5.4: *Verwechslungstabelle beim Neuronalen Netz mit den 3 Formanten und den 3 Teilenergien als Merkmalsvektor*

Die durchschnittliche Erkennungsrate von 95,83% ist somit besser als die des DTW-Klassifikators.
Die Fähigkeit des Neuronalen Netzes, auch komplizierte Klassenregionen definieren zu können, macht es möglich, direkt die LPC-Parameter als Merkmale zu verwenden. In diesem Fall entfällt die Verarbeitung für die Berechnung der Formantenfrequenzen und der Teilenergiebeträge. Nach der Verwechslungstabelle 5.5 fällt sogar die gesamte Erkennungsrate (96,67%) geringfügig besser aus.

	/a/	/e/	/i/	/o/	/u/
/a/	**100**	0,0	0,0	0,0	0,0
/e/	0,0	**100**	0,0	0,0	0,0
/i/	0,0	0,0	**100**	0,0	0,0
/o/	0,0	0,0	4,2	**91,7**	4,2
/u/	0,0	0,0	0,0	8,3	**91,7**

Tabelle 5.5: *Verwechslungstabelle beim Neuronalen Netz mit den LPC-Parametern als Merkmalsvektor*

Man darf aber nicht zum Schluss kommen, dass die direkte Verwendung der LPC-Parameter grundsätzlich zu besseren Ergebnissen führt; dies ist nämlich nicht bei geräuschbehafteten Signalen der Fall (wie später gezeigt werden wird).
Für die in Tabelle 5.5 vorgestellten Ergebnisse wurde ein Feedforward-Netz mit 15 Einheiten in der verdeckten Schicht eingesetzt. Sowohl hier als auch bei 5.4 wurde das Netz mit dem *Backpropagation*-Algorithmus (s. Abschnitt 6.1.2) trainiert.

Abbildung 5.6 zeigt, dass die Erkennungsergebnisse für niedrige Signal-zu-Rausch-Verhältnisse nicht mehr so hoch sind wie beim störfreien Signal. Bei diesen Ergebnissen wurde ein in einem Fahrzeug aufgenommenes Geräusch zu unterschiedlichen Verhältnissen (-5 bis 15dB in 5dB-Stufen) auf alle Originalsignale addiert.
Die Ergebnisse bei den geräuschbehafteten Signalen werden nicht mehr durch die detaillierten Verwechslungstabellen repräsentiert. Für eine bessere Übersichtlichkeit wird nur noch die durchschnittliche Erkennungsrate aller Vokale betrachtet.

Man stößt bei den geräuschbehafteten Signalen auf folgendes Problem: Bei vielen Testaufnahmen, vor allem bei -5 und 0dB, kann nicht einmal die Grundfrequenz be-

stimmt werden (Tabelle 5.6), da sie vom Geräusch verdeckt wird. In diesem Fall wird eine solche Aufnahme nicht berücksichtigt und es werden keine Merkmale (LPC-Parameter) berechnet, da der Klassifikator (das Neuronale Netz) in diesem Fall nur noch Zufallsklassifizierungen liefern würde. Solche Fälle werden zusätzlich zu den „nicht erkannten“ Vokalen mitgezählt. Das bedeutet, dass z.B. bei den -5dB-Aufnahmen, bereits $120 - 74 = 46$ Aufnahmen als „falsch erkannt“ angenommen werden, noch bevor sie überhaupt zum Erkenner gelangen. Vom Erkenner werden nur die restlichen 74 Aufnahmen (genauer gesagt ihre Merkmale) zum Testen herangezogen.

	gesamte Anzahl	Pitch erkannt	[%]
-5dB	120	74	61,2
0dB	120	101	84,2
5dB	120	106	88,3
10dB	120	118	98,3
15dB	120	120	100
>25dB	120	120	100

Tabelle 5.6: *Bei niedrigeren Signal-zu-Rausch-Verhältnissen konnte bei einer großen Anzahl der Vokale kein Pitch gefunden werden.*

Man stößt noch auf ein weiteres Problem: würde man das Neuronale Netz nur mit den Originalsignalen trainieren, so würde die Erkennungsrate bei niedrigeren SNRs sehr schlecht ausfallen (Kurve (c) in Abb. 5.6).

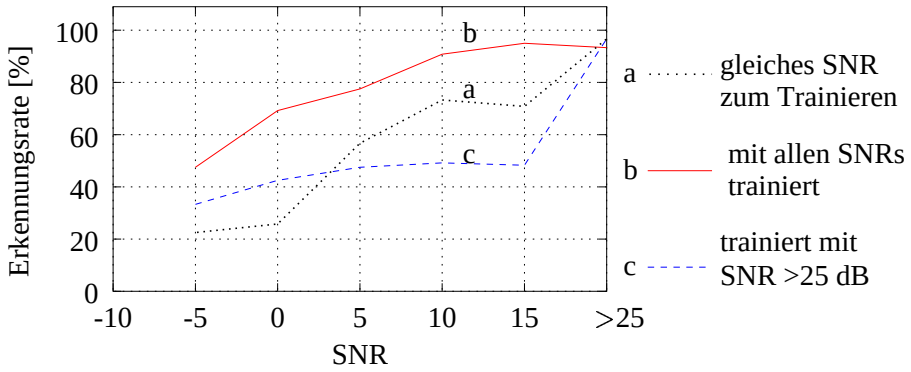


Abbildung 5.6: *Die Erkennungsrate der Vokale mit einem Feedforward-NN und den LPC-Parametern: (a) Lerndaten und Testdaten besaßen genau das gleiche SNR, (b) ein großer Lerndatensatz wurde aus allen SNR-Stufen erstellt, (c) Lerndaten nur aus störfreien Signalen.*

Würde man das Netz auch mit den geräuschbehafteten Signalen trainieren, erhält man wesentlich bessere Ergebnisse für alle SNRs (Kurve (b) in Abb. 5.6 bzw. Tabelle 5.7). Eine Verschlechterung der durchschnittlichen Erkennungsrate bei den Originalsignalen auf 93,3% muss in Kauf genommen werden.

Kurve (a) (erste Zeile in Tabelle 5.7) stellt eher einen theoretischen Fall dar, bei dem sowohl die Trainingsdaten als auch die Testdaten genau das gleiche SNR besaßen. Alle drei Fälle sind zwecks genauerer Untersuchung nocheinmal in Tabelle 5.7 eingetragen.

	-5dB	0dB	5dB	10dB	15dB	Orig.
(a) ER[%]	22,5	25,8	56,7	73,3	70,8	96,7
(b) ER[%]	47,5	69,2	77,5	90,8	95,0	93,3
(c) ER[%]	33,3	42,5	47,5	49,2	48,3	96,7

Tabelle 5.7: *Die Erkennungsrate (ER) bei den 3 Fällen der Abb. 5.6*

Der Fall, der durch Kurve (c) dargestellt ist, ist von allen drei Fällen der realistischste. In der Praxis werden die Neuronalen Netze mit störfreien Signalen trainiert. Informationen über den Geräuschtyp, seine Dauer und sein SNR sind im allgemeinen nicht bekannt. Selbst wenn man sich nur auf bestimmte vordefinierte Geräuschtypen beschränken will, müsste man das Netz mit diesen Typen mit jeweils mehreren unterschiedlichen SNRs trainieren, was aber eine bessere ER nicht garantiert. Zudem sind die Geräusche in der Wirklichkeit überwiegend instationär, sodass die Eigenschaften des Geräusches am Ende eines Wortes nicht einmal mit denen am Anfang des gleichen Wortes übereinstimmen. Die Erkennungraten nach der Kurve (c) entsprechen daher der Realität. Zum Vergleich zwischen den verschiedenen Ergebnissen soll deshalb nur noch dieser Fall betrachtet werden.

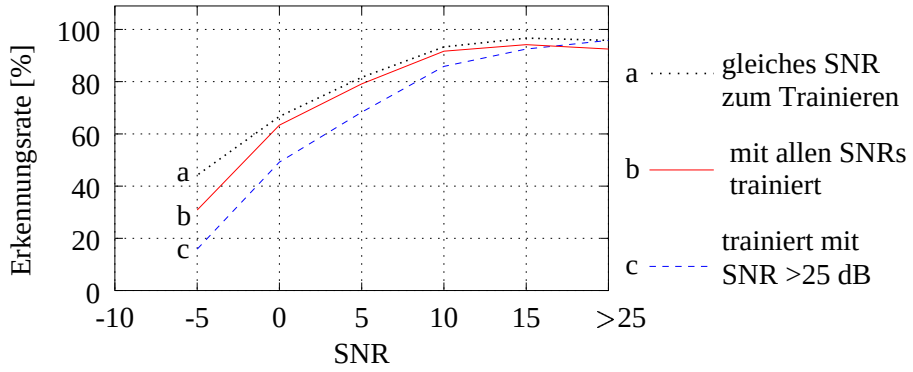


Abbildung 5.7: *Die Erkennungsrate der Vokale mit einem Feedforward-NN und die Formantenfrequenzen: (a) Lerndaten und Testdaten besaßen genau das gleiche SNR, (b) ein großer Lerndatensatz wurde aus allen SNR-Stufen erstellt, (c) Lerndaten nur aus störfreien Signalen.*

Man beobachtet, dass die Erkennungskurve (c) in Abb. 5.6 bereits ab einem SNR von 15 dB einbricht. Würde man statt der LPC-Parameter die 3 Formantenfrequenzen und die 3 Teilenergien als Merkmale verwenden, erhielte man wesentlich bessere Ergebnisse für den Fall (c) (Tabelle 5.8 bzw. Abb. 5.7).
Die weiteren Fälle (a) und (b) sind der Vollständigkeit wegen auch dargestellt. Man kann also zusammenfassen: die besten Erkennungsergebnisse für den einzigen reali-

	-5dB	0dB	5dB	10dB	15dB	Orig.
(a) ER[%]	44,2	66,7	81,7	93,3	96,7	95,8
(b) ER[%]	30,8	63,3	79,2	91,7	94,2	92,5
(c) ER[%]	15,8	49,2	68,3	85,8	92,5	95,8

Tabelle 5.8: *Die Erkennungsrate (ER) bei den 3 Fällen der Abb. 5.7*

stischen Fall erhält man beim Verwenden der 3 Formantenfrequenzen und der 3 Teilenergien zusammen mit einem Neuronalen Netz als Erkenner.

5.4.2 Ergebnisse des dynamischen Datensatzes

Diagramm 5.8 zeigt die Worterkennungsraten (WER) des dynamischen Datensatzes (die französischen Ziffern). Für die Merkmalsextraktion wurden die ersten 8 MFC-Koeffizienten verwendet. Die gleichen WER-Werte von Diagramm 5.8 sind zwecks genauerer Untersuchung noch einmal in Tabelle 5.9 eingetragen. Die Aufnahmen des Lerndatensatzes wurden mit einem kontinuierlichen *left-right*-HMM mit 5 Zuständen modelliert. Die Klassifikation selbst (die Zuordnung eines Testkandidaten zu einem HMM) erfolgte mit dem Viterbi-Algorithmus (s. 5.2.2).

Um die Einwirkung des Geräusches auf die Erkennungsrate zu demonstrieren, wurde hier ebenfalls das Autofahrgeräusch verwendet. Die Originalaufnahmen wurden mit diesem Geräusch zu Verhältnissen zwischen -5 bis 15dB in 5dB-Stufen gemischt.

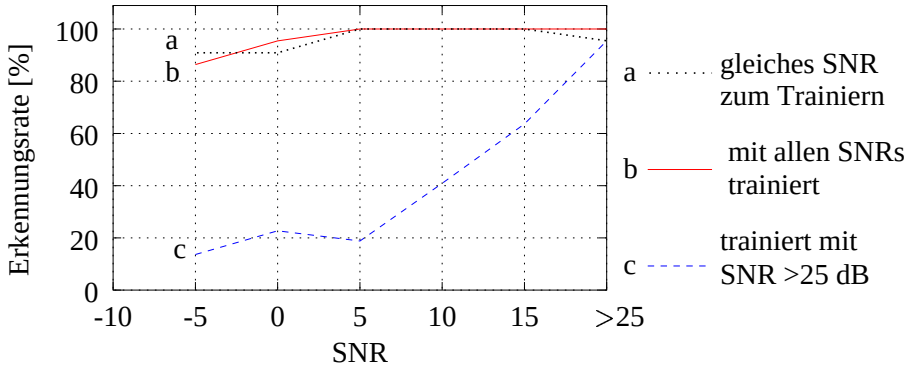


Abbildung 5.8: *Die Erkennungsrate der Wörter mit einem L-R-HMM mit 5 Zuständen: (a) Lerndaten und Testdaten besaßen genau das gleiche SNR, (b) ein großer Lerndatensatz wurde aus allen möglichen SNRs zusammengestellt, (c) Lerndaten nur aus störfreien Signalen.*

Wie man in Abb. 5.8 erkennen kann, hängt die Erkennungsrate sehr stark davon ab, welche Aufnahmen für das Erstellen der HMM-Parameter (die Lernphase) verwendet werden. Die WER-Werte aus Kurve (a) erhält man, wenn man die Modell-Parameter aus einem Lerndatensatz bestimmt, der genau zu dem gleichen SNR mit dem Geräusch gemischt war wie der Testdatensatz. D.h., dass z.B. die WER von 90,9%

	-5dB	0dB	5dB	10dB	15dB	Orig.
(a) WER[%]	90,9	90,9	100	100	95,5	95,5
(b) WER[%]	86,4	95,5	100	100	100	100
(c) WER[%]	13,6	22,7	18,8	40,9	63,6	95,5

Tabelle 5.9: *Die WER bei den 3 Fällen des Diagramms 5.8*

beim SNR=0dB zustande kommt, indem sowohl die 88 Aufnahmen des Lerndatensatzes als auch die 22 Aufnahmen des Testdatensatzes allesamt mit dem Geräusch zu 0dB gemischt werden. Dieser Fall ist nicht realistisch, da man in der Praxis das SNR des Testdatensatzes nicht im voraus kennt; es sei denn, man versucht, es zu schätzen. Außerdem müsste man viele Modelle für alle möglichen SNRs[1] ablegen.

Es scheint aber, zumindest bei diesem Datensatz und diesem Geräusch, der Fall zu sein, dass das HMM in der Lage ist, alle möglichen SNRs gleichzeitig zu modellieren (Kurve (b)). In diesem Fall wurden alle 6×88 Lernaufnahmen (-5 bis 15dB zusammen mit dem Original) zum Erstellen der Modelle verwendet. Man erkennt, dass alle Erkennungsraten (bis auf die bei -5dB) besser als in Fall (a) sind. Dies mag auf den ersten Blick seltsam erscheinen, ist aber damit zu erklären, dass der viel größere Lerndatensatz eine wesentlich bessere Statistik liefert.
Würde man die HM-Modelle nur aus den Original- (störungsfreien) Signalen erstellen (Kurve c), so würde die WER bei den mit dem Geräusch gemischten Signalen rapide abnehmen. Bei SNRs ab etwa 0dB und schlechter ist die WER mehr oder weniger eine zufällige Zahl. Dieser Fall gilt, wie es auch bei der Erkennung der Vokale der Fall war, als der realistischste von allen drei Fällen und wird auch hier als Referenz für den Vergleich mit dem Audio-Video-basierten Erkenner benutzt.

5.5 Erkennung anhand der Videodaten

In diesem Abschnitt werden die Erkennungsergebnisse dargestellt, die man nur bei der Auswertung der Videodaten erhält. Im Gegensatz zum Audiokanal werden hier keine Bildstörungen betrachtet und es wird angenommen, dass die Zuverlässigkeit der extrahierten Videoparameter konstant ist.

5.5.1 Ergebnisse des statischen Datensatzes

Synchron zur Audioaufnahme der Vokale wurden die Sprecher mit einer Videokamera aufgenommen. Für jede Äußerung wurde eine eigene Bildfolge gespeichert. Da jeder der 4 Sprecher die 5 Vokale 6 mal wiederholte, sind somit insgesamt $4 \times 5 \times 6 = 120$ Bildfolgen entstanden. Aus jeder Bildfolge wurden dann manuell Einzelbilder selektiert, die für einen Vokal repräsentativ sind. Aus einer Bildsequenz eines Vokals wurden genau 3 Einzelbilder ausgesucht. Auf diese Weise ist eine identische Anzahl von

[1] hier ist nur eine beschränkte Anzahl von SNRs gemeint, die etwa den Bereich -5 bis 20dB abdeckt, z.B. in 5dB-Stufen, wie dies auch in dieser Arbeit der Fall ist

Bildern wie die der dazugehörigen Audiodateien entstanden. Wie beim akustischen Datensatz wurden auch hier die ersten 4 Wiederholungen aller Sprecher zum Training und die letzten 2 Wiederholungen zum Testen verwendet.

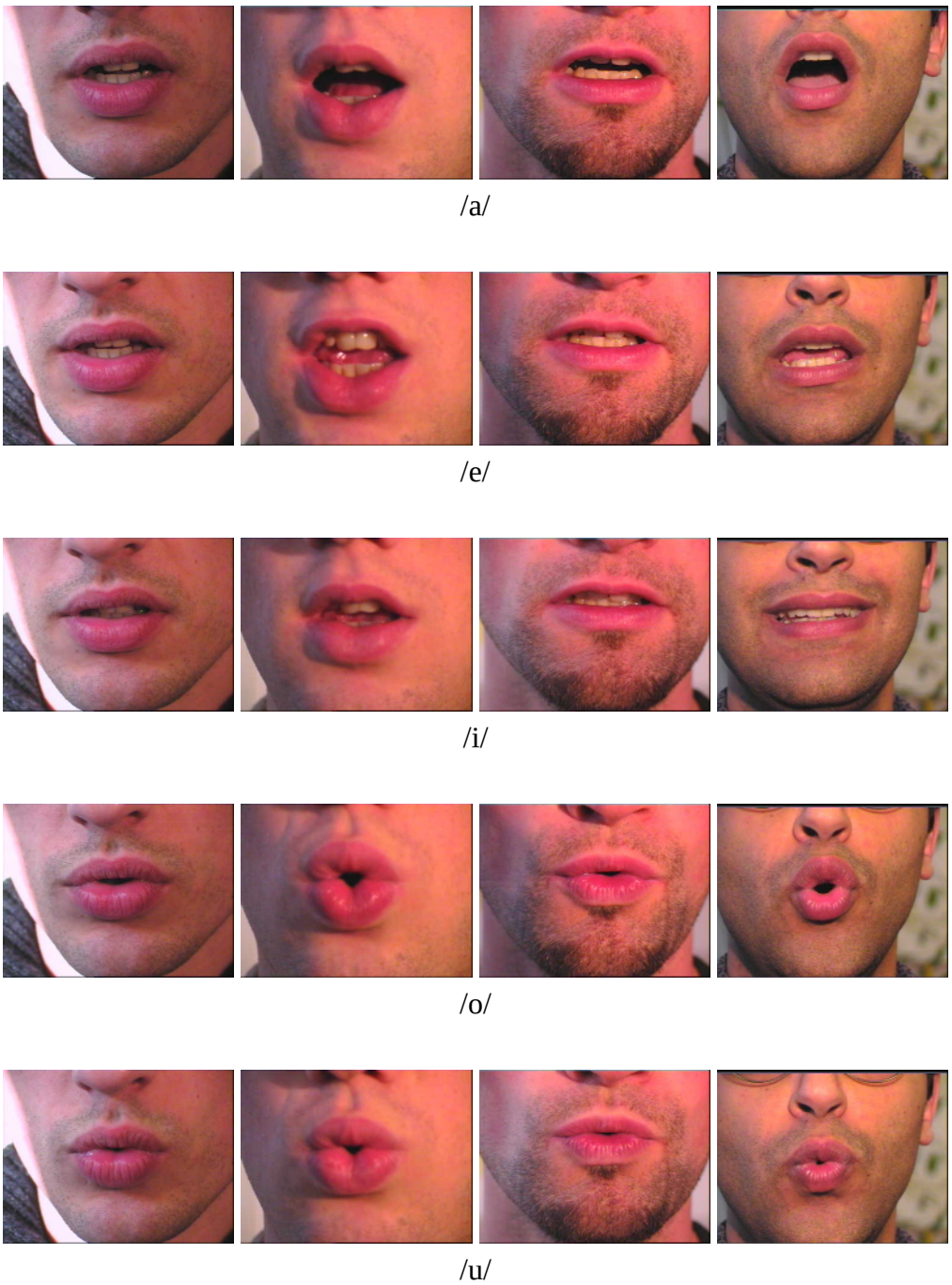
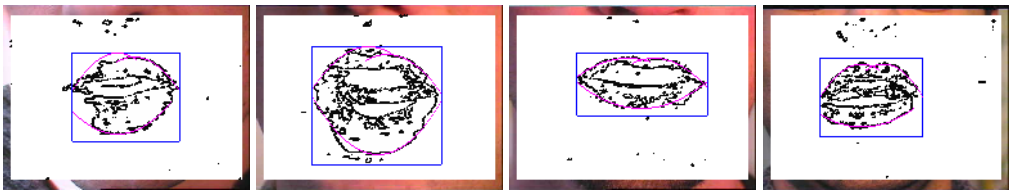


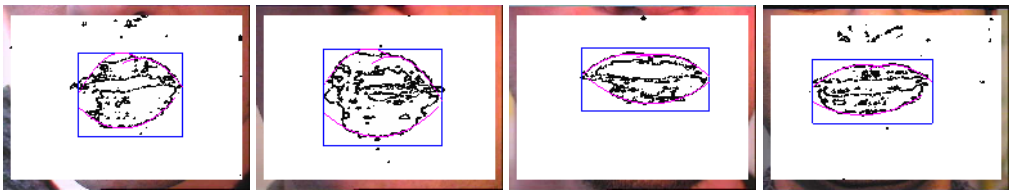
Abbildung 5.9: *Einige Beispiele aus der Vokaldatenbank*

Abbildung 5.9 zeigt einen Auszug aus der Vokalbild-Datenbank. Es fällt auf, dass die Lippenkonstellationen des gleichen Vokals von Sprecher zu Sprecher sehr stark ab-

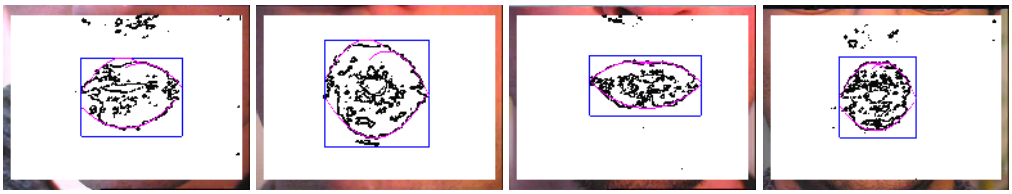
/a/



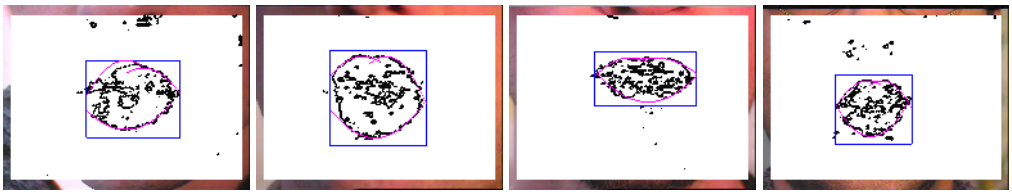
/e/



/i/



/o/



/u/

Abbildung 5.10: *Die Vokale nach der Modellierung*

weichen. Bei manchen Sprechern sind die Lippen bei den Vokalen /o/ und /u/ nur sehr wenig gewölbt.

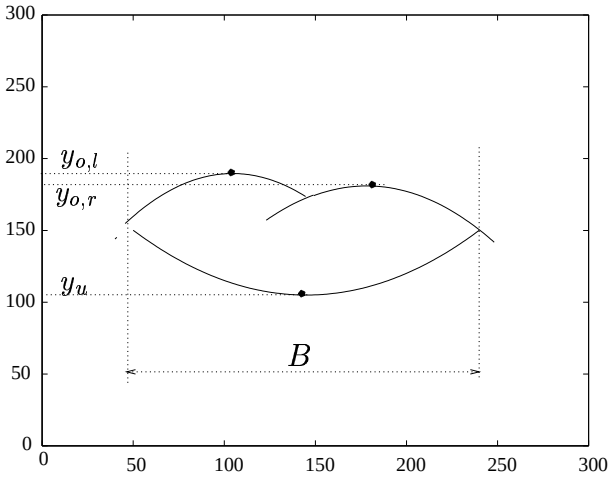


Abbildung 5.11: *Das Lippenmodell*

Die Höhe der Mundöffnung ist auch sehr unterschiedlich. Auf Grund der sehr ähnlichen Lippenform beim Vokalpaar /i/ und /e/ bzw. dem Vokalpaar /o/ und /u/ ist eine große Verwechslung innerhalb dieser Paare zu erwarten. Abbildung 5.10 zeigt die Bilder aus 5.9 nach der Merkmalsextraktion. Nach der Gauß-Filterung im chromatischen Farbraum werden alle Punkte, die nicht zu den Lippen gehören, ausgefiltert (die weiße Fläche). Die schwarzen Punkte stellen die lippnenähnlichen Regionen nach der Kantendetektion dar. Das blaue Rechteck ist die gefundene Mundregion, die mit Hilfe der vertikalen und horizontalen Projektion bestimmt wurde. Die Parabeln selbst sind durch die roten Kurven dargestellt.

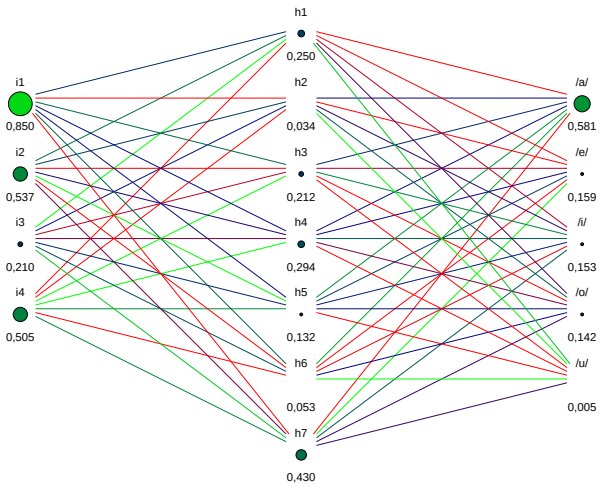


Abbildung 5.12: *Das trainierte Netz während der Erkennung eines /a/-Testvokals*

Für die Vokalerkennung wurde hier auch ein Neuronales Netz eingesetzt (Abb. 5.12). Die einzige verborgene Schicht besteht aus 7 Einheiten. Die Anzahl der Eingangseinheiten gleicht der Anzahl der Merkmale. Die Merkmale (s. Abb. 5.11) wurden von dem Parabel-Lippenmodell geliefert (Abschnitt 5.11). Jeder Merkmalsvektor besteht aus diesen 4 Größen:

1. Verhältnis Mundhöhe zu Mundbreite: $(y_{o,r} + y_{o,l} - 2 \cdot y_u)/B$,
2. die durchschnittliche Krümmung der beiden oberen Parabelstücke K_o,
3. die Krümmung der unteren Parabel K_u und
4. das Verhältnis Mundbreite zu Gesichtsbreite.

Das trainierte Netz während der Erkennung eines /a/-Vokals ist in 5.12 dargestellt. Je höher die Aktivierung einer Einheit, desto größer ist ihr Durchmesser.

	/a/	/e/	/i/	/o/	/u/
/a/	**54,2**	16,7	12,5	4,2	12,5
/e/	8,3	**54,2**	20,8	0,0	16,7
/i/	0,0	41,7	**41,7**	4,2	12,5
/o/	8,3	8,3	0,0	**29,2**	54,2
/u/	8,3	0,0	0,0	4,2	**87,5**

Tabelle 5.10: *Die durchschnittliche Erkennungsrate des statischen Datensatzes beträgt 53,33 %*

Tabelle 5.10 fasst die Ergebnisse in Form einer Verwechslungsmatrix zusammen.

	/a/	/e/	/i/	/o/	/u/
/a/	**100**	0,0	0,0	0,0	0,0
/e/	0,0	**75,0**	25,0	0,0	0,0
/i/	0,0	0,0	**100**	0,0	0,0
/o/	0,0	0,0	0,0	**87,5**	12,5
/u/	12,5	0,0	0,0	25,0	**62,5**

Tabelle 5.11: *sprecherabhängige Erkennung des ersten Sprechers, durchschnittliche Erkennungsrate = 85 %*

Die befürchtete große Verwechslung zwischen den Vokalen /o/ und /u/ wird bestätigt. Die durchschnittliche Erkennungsrate beträgt 53,33 %. Diese niedrige Rate ist auf die große Variabilität der Lippenformen zurückzuführen, kompliziertere Modelle zur Lippenapproximation würden diese Rate nur unwesentlich erhöhen.
Dies kann durch die Betrachtung eines einzigen Sprechers demonstriert werden, da in diesem Fall die Variabilität deutlich sinkt. Für diesen Zweck wurde die Datenbank

um zusätzliche Aufnahmen des ersten Sprechers erweitert, so dass für diesen Sprecher insgesamt 12 Aufnahmen pro Vokal für die Lernphase und 8 weitere Aufnahmen pro Vokal für die Testphase zur Verfügung standen. Die Erkennung ist somit sprecherabhängig. Die resultierenden Ergebnisse sind Tabelle 5.11 zu entnehmen. In diesem Fall erhöht sich die durchschnittliche Erkennungsrate auf 85 %.

5.5.2 Ergebnisse des dynamischen Datensatzes

Dieser Datensatz besteht, im Gegensatz zum statischen Datensatz, nicht aus Einzelbildern, sondern aus ganzen Bildfolgen. Da aber eine Bildfolge nichts anderes ist als eine Sequenz von Einzelbildern, kann man auch hier eine identische Verarbeitung wie die bei dem statischen Datensatz auf jedes Bild der Sequenz anwenden.
Die Verarbeitung unterscheidet sich hier jedoch von der des statischen Datensatzes dadurch, dass bestimmte dynamische Eigenschaften ausgenutzt werden, um die Verarbeitung zu beschleunigen. Außerdem kann hier eine Zuverlässigkeit der Merkmale geschätzt werden. Die Beschleunigungsmaßnahmen wurden bereits in 2.2.6 ausführlich erklärt. Die Zuverlässigkeit der Merkmale kann durch die Änderung der Merkmale von einem Bild zum nächsten bestimmt werden. Falls diese Änderung „sehr groß“ ist, weist dies auf eine fehlerhafte Modellierung hin, dies ist z.B. der Fall, wenn bei einem Bild die Lippenposition plötzlich an einen anderen, sehr entfernten Bildpunkt springt. Für den Fall, dass die Merkmale eines Bildes als unzuverlässig gelten, kann man z.B. die Merkmale des vorherigen Bildes übernehmen, oder die des vorherigen mit dem des folgenden Bildes mitteln. Falls aber die Merkmale als zuverlässig gelten, besteht die Möglichkeit, die aktuellen Merkmale mit den vorangegangenen exponentiell zu filtern, um einen glatten Verlauf der Merkmale zu erhalten. Ein Merkmal m_i aus einem Bild i wird in diesem Fall mit dem Merkmal m_{i-1} aus dem Bild $i-1$ nach:

$$\tilde{m_i} = \alpha.m_i + (1-\alpha).m_{i-1} \tag{5.12}$$

geglättet. Der Glättungsfaktor α wird typischerweise zwischen 0,8 und 0,99 gewählt.
Die Merkmale aus jedem Bild sind identisch mit denen aus dem statischen Datensatz. Für die Erkennung werden HM-Modelle mit 5 Zuständen aus den Lerndaten erstellt. Die Erkennungsergebnisse sind in der Verwechslungstabelle 5.12 dargestellt. Die durchschnittliche WER beträgt 54,6%.
Der Testdatensatz ist zu klein, um allgemeine Aussagen machen zu können; man kann jedoch, wenn man die Verwechslungstabelle 5.12 mit der vom Audioerkenner bei SNR=10dB (Tabelle 5.13) vergleicht, einige Schlussfolgerungen ziehen.
Es fällt auf, dass die Erkennung der Wörter *zéro, un* und *neuf* anhand des audiobasierten Erkenners sehr erfolgreich ist, während die restlichen Wörter mit niedrigerem Erfolg oder gar nicht erkannt worden sind. Beim videobasierten Erkenner sind die Wörter *quatre, sept* und *dix* zu 100% erkannt worden. Das Wort *trois* wurde von beiden Erkennern immer der richtigen Klasse zugeordnet.
Die Klassifizierung mit Hilfe der Audiomerkmale ist also bei einer Gruppe von Wörtern erfolgreicher gegenüber der Klassifizierung der gleichen Wörter anhand der Videomerkmale. Andere Wörter dagegen, die im Audio-Merkmalsraum unscharfe Klassentrennflächen haben, können im Video-Merkmalsraum besser klassifiziert wer-

Wort	0	1	2	3	4	5	6	7	8	9	10
0	**1**	0	0	0	0	1	0	0	0	0	0
1	0	**0**	0	0	2	0	0	0	0	0	0
2	0	0	**1**	0	0	0	0	0	0	1	0
3	0	0	0	**2**	0	0	0	0	0	0	0
4	0	0	0	0	**2**	0	0	0	0	0	0
5	1	0	0	0	0	**0**	0	0	0	0	1
6	0	0	0	0	0	1	**0**	0	0	0	1
7	0	0	0	0	0	0	0	**2**	0	0	0
8	0	0	0	1	0	0	0	0	**1**	0	0
9	0	0	0	0	0	1	0	0	0	**1**	0
10	0	0	0	0	0	0	0	0	0	0	**2**

Tabelle 5.12: *Verwechslungstabelle des videobasierten Erkenners. Die Zahlen stellen die Anzahl der richtig erkannten Wörter dar.*

Wort	0	1	2	3	4	5	6	7	8	9	10
0	**2**	0	0	0	0	0	0	0	0	0	0
1	0	**2**	0	0	0	0	0	0	0	0	0
2	2	0	**0**	0	0	0	0	0	0	0	0
3	0	0	0	**2**	0	0	0	0	0	0	0
4	2	0	0	0	**0**	0	0	0	0	0	0
5	0	0	1	0	0	**1**	0	0	0	0	0
6	0	0	2	0	0	0	**0**	0	0	0	0
7	2	0	0	0	0	0	0	**0**	0	0	0
8	2	0	0	0	0	0	0	0	**0**	0	0
9	0	0	0	0	0	0	0	0	0	**2**	0
10	0	0	2	0	0	0	0	0	0	0	**0**

Tabelle 5.13: *Verwechslungstabelle des audiobasierten Erkenners bei einem SNR von 5dB, die durchschnittliche WER beträgt 40,9%.*

den. Diese sehr niedrige Korrelation der beiden Merkmalsvektoren deutet auf den Sinn der Datenfusion beider Kanäle hin.

5.6 Zusammenfassung

In diesem Kapitel wurden die Erkennungsergebnisse repräsentiert, die man nur unter Berücksichtigung des Audio- bzw. Videokanals erhält. Alleine mit den Merkmalen, die aus dem Audiokanal stammen, sind Erkennungsraten nahe 100% erreichbar. Diese hohen Erkennungsraten sinken sehr stark, sobald das Sprachsignal mit einem Geräusch überlagert ist. Anhand des Videokanals ist die Erkennungsrate zwar niedriger im Vergleich zu einem störfreien Audiokanal, aber sie ist über alle SNR-Werte konstant. Ab

etwa 0dB und schlechter beim statischen Datensatz bzw. unter 10dB beim dynamischen Datensatz ist die Erkennung anhand der Videodaten besser als die des Audiokanals. Auf die Frage, wie man nun beide Kanäle gleichzeitig auswerten kann, damit die Erkennungsrate über alle SNRs besser als die bei der Auswertung eines einzelnen Kanals wird, wird im nächsten Kapitel eingegangen.

Kapitel 6

Fusion von Audio- und Videomerkmalen

In diesem Kapitel werden verschiedene Konzepte zur Fusion der aus dem Video- und Audiokanal gewonnenen Daten vorgestellt. Dabei kommt ein künstliches Neuronales Netz zum Einsatz. Es wird gezeigt, dass die Erkennung der 5 deutschen Vokale /a/, /e/, /i/, /o/ und /u/ in einer Geräuschumgebung durch die Fusion deutlich bessere Ergebnisse erreicht als nur bei der Auswertung des Audiokanals. Auch die Erkennung der einzelnen Wörter (der dynamische Datensatz) konnte durch eine einfache HMM-Modellierung deutlich verbessert werden.

In der vorliegenden Arbeit wurde das SNNS[1]-Paket verwendet. Es handelt sich um einen Software-Simulator für Neuronale Netze, der an der Fakultät für Informatik, Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR), an der Universität Stuttgart entwickelt wurde.

6.1 Neuronale Netze (NN)

Künstliche Neuronale Netze eignen sich vor allem für komplizierte Klassifikationsaufgaben. Ihre Funktionsweise ist dem Nervensystem der Lebewesen nachempfunden. Im folgendem Text werden die künstlichen Neuronalen Modelle kurz mit Neuronalen Netzen (NN) bezeichnet. Komplizierte Entscheidungsräume können mit NN während der Lernphase systematisch definiert werden.
Für reine Klassifikationsaufgaben, bei denen jede Beobachtung aus genau einem einzigen Merkmalsvektor besteht, können NN ohne weitere Hilfe eingesetzt werden. Bei der Worterkennung wird aber jedes Wort durch eine variierende Anzahl von Mustervektoren dargestellt. Im Gegensatz zu den HM-Modellen oder zum DTW-Algorithmus sind daher die meisten einfacheren NN-Strukturen nicht direkt für Erkennenunsaufgaben geeignet. Sie sind also nicht in der Lage, mit zeitlichen Merkmalsfolgen unterschiedlicher Länge umzugehen. Daher wird sehr häufig der DTW-Algorithmus zusammen mit einem NN kombiniert, um die Anzahl der Merkmalsvektoren an einander

[1]SNNS steht für *Stuttgart Neural Network Simulator*

anzupassen.
Abb. 6.1 stellt die Kerneinheit eines Neuronalen Netzes, nämlich ein Neuron j samt Aktivierungen, dar.

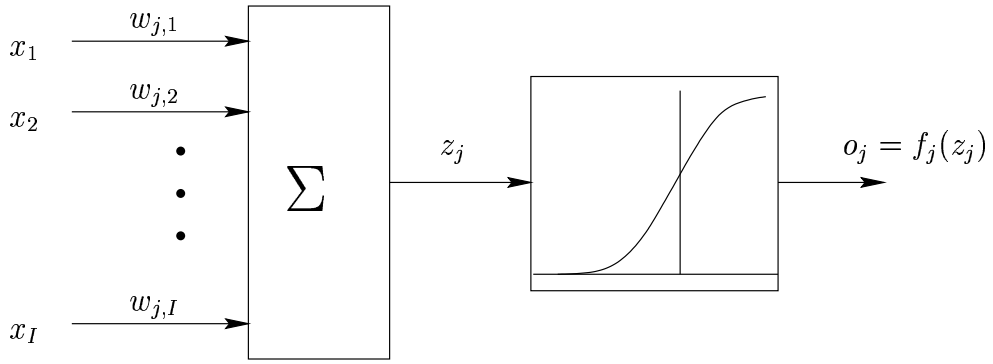


Abbildung 6.1: *Modell eines Neurons*

Die Reize oder Aktivierungen $x_1, x_2, \ldots, x_I$, die entweder aus Ausgängen anderer Neuronen oder direkt von dem Merkmalsvektor stammen, werden jeweils mit $w_{j,1}, w_{j,2}, \ldots, w_{j,I}$ unterschiedlich gewichtet und aufsummiert:

$$z_j = \sum_{i=1}^{I} w_{j,i} \cdot x_i \tag{6.1}$$

Oder in Vektorschreibweise:

$$\begin{aligned} z_j &= \mathbf{w}_j^T \mathbf{x}_j = \mathbf{x}_j^T \mathbf{w}_j \\ \text{mit: } \mathbf{x}_j &= (x_1, x_2, \ldots, x_I)^T \ , \ \mathbf{w}_j = (w_{j,1}, w_{j,2}, \ldots, w_{j,I})^T \end{aligned} \tag{6.2}$$

Die in Gl. 6.1 verwendete additive Verknüpfung kann im allgemeinen Fall durch eine beliebige Funktion $g(\mathbf{x}, \mathbf{w})$ ersetzt werden.

Der Ausgang o_j, den dieses Neuron „feuert“, dient seinerseits als Eingang für andere, mit ihm verbundene Neuronen. Die Aktivierungsfunktion $f_j(z_j)$ bestimmt diesen Ausgang o_j in Abhängigkeit von z_j. Häufig wird eine Sigmoid-Funktion dafür verwendet, da ihre Ableitung, die bei vielen Lernverfahren benötigt wird, stetig ist. Man verwendet die gleiche Aktivierungsfunktion für alle Neuronen, deswegen entfällt der Index j, also $f_j(z_j) = f(z_j)$.

$$o_j = f(z_j) = \frac{1}{1 + e^{-z_j}} \tag{6.3}$$

6.1.1 Die Feed-forward Netzstruktur

Abb. 6.2 stellt die sehr häufig verwendete Feed-forward Netzstruktur dar. Die Neuronen sind durch kleine Kreise dargestellt. Das Netz besteht aus Hierarchien von aufeinanderfolgenden Schichten. Jede Schicht hat nur Verbindungen zur nächsten Schicht,

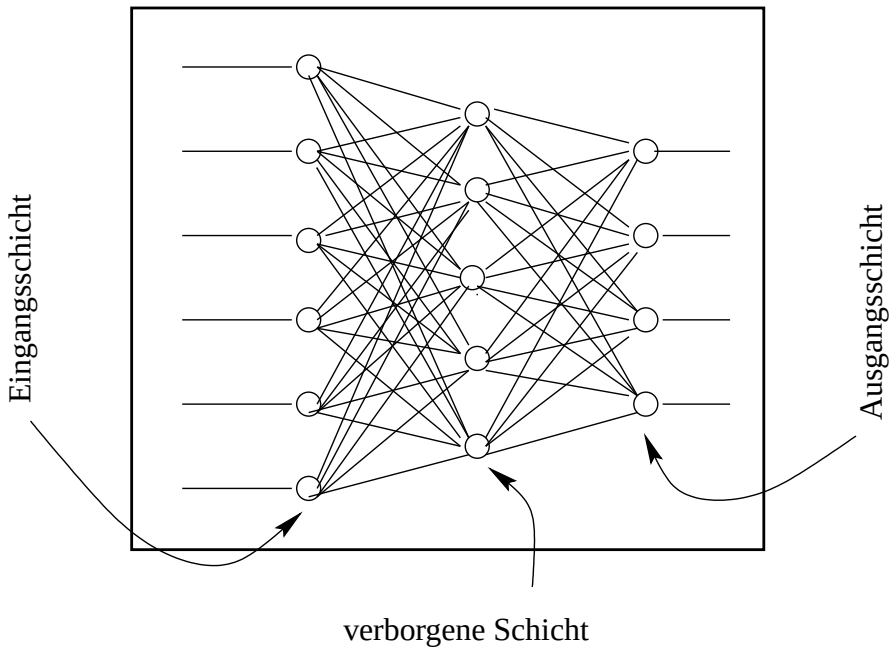


Abbildung 6.2: *Feed-forward-Struktur mit einer verborgenen Schicht*

und zwar eine vollständige Verbindung, d.h **jedes** Neuron einer Schicht ist mit **jedem** Neuron der nächsten Schicht verbunden.

Die Neuronen der ersten linken Reihe bilden zusammen die Eingangsschicht. Der Signalfluss verläuft von links nach rechts. Jedes Neuron der Eingangsschicht bezieht seinen Eingang aus einem Merkmal. Die mittlere Reihe von Neuronen bildet eine verborgene Schicht. Die rechte Reihe ist die Ausgangsschicht. Es können auch beliebig viele verborgene Schichten existieren.

Am einfachsten werden so viele Ausgangsneuronen gewählt, wie die Anzahl der Klassen beträgt. Jedes Ausgangsneuron repräsentiert somit genau eine Klasse. Das Ausgangsneuron mit der größten Aktivierung bestimmt dann die Klassenzugehörigkeit. In der Lernphase müssen also alle Gewichtsfaktoren so eingestellt werden, dass tatsächlich das Ausgangsneuron, das die Klasse eines bestimmten Lernvektors repräsentiert, den größten Ausgang unter allen Ausgangsneuronen erzeugt. Diese Einstellungsphase wird auch mit Trainingsphase bezeichnet.

6.1.2 Der Backpropagation-Algorithmus

Mit dem Backpropagation-Algorithmus [Bra91] kann eine Feed-forward-Struktur mit Hilfe eines Lernsatzes trainiert werden. Während des Trainings sind die Klassenzugehörigkeiten bekannt, so dass man die Gewichtsfaktoren einstellen kann.

Wir betrachten die Gewichtsfaktoren zwischen zwei benachbarten Schichten mit jeweils $I = \{1, 2, \ldots, i, \ldots, I\}$ und $J = \{1, 2, \ldots, j, \ldots, J\}$ Neuronen. Alle Gewichts-

faktoren zwischen diesen beiden Schichten werden durch die Matrix

$$\mathbf{W} = (w_{i,j}) = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,j} & \cdots & w_{1,J} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,j} & \cdots & w_{2,J} \\ \vdots & \vdots & & \ddots & & \vdots \\ w_{i,1} & w_{i,2} & \cdots & w_{i,j} & \cdots & w_{i,J} \\ \vdots & \vdots & & \ddots & & \vdots \\ w_{I,1} & w_{I,2} & \cdots & w_{I,j} & \cdots & w_{I,J} \end{pmatrix} \tag{6.4}$$

dargestellt. Nehmen wir an, dass eine Menge von $\mathbf{P} = \{1, 2, \ldots, p, \ldots, P\}$ Lernmustern zur Verfügung steht. Bei einem bestimmten Lernmuster p ensteht am Ausgang des Neurons j ein Fehler $e_j^{(p)}$ zwischen der tatsächlichen Aktivierung $o_j^{(p)}$ und der gewünschten Aktivierung $t_j^{(p)}$. Dieser Fehler wird durch den quadratischen Abstand zwischen $o_j^{(p)}$ und $t_j^{(p)}$ definiert:

$$e_j^{(p)} = \frac{1}{2}\left(t_j^{(p)} - o_j^{(p)}\right)^2 \tag{6.5}$$

Die Summe dieses Fehlers über alle Neuronen der zweiten Schicht $E^{(p)}$ ist dann eine Funktion der Gewichtsmatrix $\mathbf{W}$, da o_j seinerseits eine Funktion des Gewichtsvektors $\mathbf{w}_j$ ist:

$$E^{(p)} = E^{(p)}(\mathbf{W}) = \frac{1}{2}\sum_{j=1}^{J}(t_j^{(p)} - o_j^{(p)})^2 \ . \tag{6.6}$$

Die Gewichtsfaktoren $w_{i,j}$ sind so zu wählen, dass der Fehler $E^{(p)}$ zum Minimum wird. Die Grundidee des Backpropagation-Algorithmus besteht darin, die Gewichtsfaktoren $w_{i,j}$ in Richtung des negativen Gradienten des Fehlers $E^{(p)}$ zu verändern, sodass mit jedem Lernschritt p der Fehler $E^{(p)}$ kleiner wird:

$$w_{i,j}^{(p)} = w_{i,j}^{(p-1)} - \eta \cdot \frac{\partial E^{(p)}}{\partial w_{i,j}^{(p)}} = w_{i,j}^{(p-1)} + \Delta w_{i,j}^{(p)} \ . \tag{6.7}$$

Der Parameter η stellt die Schrittweite dar, mit der die Änderungen an den Gewichtsfaktoren stattfinden. Als Startwerte für $w_{i,j}^{(0)}$ verwendet man z.B. Zufallsvariable. Aus 6.7 und mit Hilfe der Kettenregel folgt für die Änderung $\Delta w_{i,j}^{(p)}$:

$$\Delta w_{i,j}^{(p)} = -\eta \cdot \frac{\partial E^{(p)}}{\partial w_{i,j}^{(p)}} = -\eta \cdot \frac{\partial E^{(p)}}{\partial z_j^{(p)}} \cdot \frac{\partial z_j^{(p)}}{\partial w_{i,j}^{(p)}}. \tag{6.8}$$

Der zweite Differentialquotient in 6.8 lässt sich mit 6.1 bestimmen:

$$\frac{\partial z_j^{(p)}}{\partial w_{i,j}^{(p)}} = x_i^{(p)} = o_i^{(p)} \ . \tag{6.9}$$

Dabei gilt $x_i^{(p)} = o_i^{(p)}$, da die Eingangswerte $x_i^{(p)}$ für die zweite Schicht nichts anderes sind als die Ausgangswerte der ersten Schicht.

Der erste Differentialquotient in 6.8 wird als das Fehlersignal $\delta_j^{(p)}$ bezüglich des Neurons j definiert und lässt sich wiederum mit der Kettenregel gemäß

$$-\delta_j^{(p)} = \frac{\partial E^{(p)}}{\partial z_j^{(p)}} \tag{6.10}$$

$$= \frac{\partial E^{(p)}}{\partial o_j^{(p)}} \cdot \frac{\partial o_j^{(p)}}{\partial z_j^{(p)}} \tag{6.11}$$

umformen. Setzt man nun diese beiden Differentialquotienten aus den Gleichungen 6.9 und 6.10 in die Gleichung 6.8 ein, erhält man:

$$\begin{aligned} \Delta w_{i,j}^{(p)} &= -\eta \cdot \underbrace{\frac{\partial E^{(p)}}{\partial z_j^{(p)}}}_{-\delta_j^{(p)}} \cdot \underbrace{\frac{\partial z_j^{(p)}}{\partial w_{i,j}^{(p)}}}_{o_i^{(p)}} \\ &= \eta \cdot \delta_j^{(p)} \cdot o_i^{(p)} \end{aligned} \tag{6.12}$$

Für eine Sigmoid-Aktivierungsfunktion kann man beweisen:

$$\frac{df(z_j)}{dz_j} = f(z_j)(1 - f(z_j)) = o_j(1 - o_j) \tag{6.13}$$

Daraus folgt für 6.10:

$$\frac{\partial E^{(p)}}{\partial z_j^{(p)}} = \frac{\partial E^{(p)}}{\partial o_j^{(p)}} \cdot o_j^{(p)} \cdot (1 - o_j^{(p)}) \; . \tag{6.14}$$

Soll $E^{(p)}$ direkt nach einem Ausgangsneuron j abgeleitet werden, dann gilt für die Ableitung von $E^{(p)}$ mit Hilfe der Definition von $E^{(p)}$ aus Gleichung 6.6:

$$-\frac{\partial E^{(p)}}{\partial o_j^{(p)}} = -\frac{\partial}{\partial o_j^{(p)}} \sum_{j=1}^{J} \frac{1}{2} \left(t_j^{(p)} - o_j^{(p)}\right)^2 = t_j^{(p)} - o_j^{(p)} \; . \tag{6.15}$$

Soll $E^{(p)}$ aber nach der Aktivierung eines Neurons i der verborgenen Schicht abgeleitet werden, so lässt es sich nicht direkt berechnen. Da aber die Aktivierungen $o_j^{(p)}$ von $o_i^{(p)}$ abhängig sind, gilt für 6.6:

$$\begin{aligned} -\frac{\partial E^{(p)}}{\partial o_i^{(p)}} &= -\frac{\partial}{\partial o_i^{(p)}} \sum_{j=1}^{J} \frac{1}{2} \left(t_j^{(p)} - o_j^{(p)}\right)^2 \\ &= -\sum_{j=1}^{J} \underbrace{2 \cdot \frac{-1}{2} \left(t_j^{(p)} - o_j^{(p)}\right)}_{\text{benutze 6.15}} \cdot \frac{\partial o_j^{(p)}}{\partial o_i^{(p)}} \\ &= -\sum_{j=1}^{J} \frac{\partial E^{(p)}}{\partial o_j^{(p)}} \cdot \frac{\partial o_j^{(p)}}{\partial o_i^{(p)}} \end{aligned} \tag{6.16}$$

$$
\begin{aligned}
&= -\sum_{j=1}^{J} \underbrace{\frac{\partial E^{(p)}}{\partial o_j^{(p)}} \cdot \frac{\partial o_j^{(p)}}{\partial z_j^{(p)}}}_{\text{benutze die Kettenregel}} \cdot \frac{\partial z_j^{(p)}}{\partial o_i^{(p)}} \\
&= -\sum_{j=1}^{J} \frac{\partial E^{(p)}}{\partial z_j^{(p)}} \cdot \frac{\partial z_j^{(p)}}{\partial o_i^{(p)}} \quad \text{, benutze 6.10} \\
&= \sum_{j=1}^{J} \delta_j^{(p)} \cdot \frac{\partial}{\partial o_i^{(p)}} \sum_k o_k^{(p)} w_{k,j} = \sum_{j=1}^{J} \delta_j^{(p)} w_{i,j} \; .
\end{aligned}
$$

D.h., der Fehler eines verborgenen Neurons i lässt sich durch die mit $w_{i,j}$ gewichtete Summe der Fehler $\delta_j^{(p)}$ der nachfolgenden Neuronen j ausdrücken, so dass Gleichung 6.16 rekursiv auf das ganze verbleibende Netz angewendet werden kann.
Die Berechnung der Gewichtsänderungen für den Fall einer Sigmoid- Aktivierungsfunktion lässt sich durch folgende Gleichungen übersichtlich darstellen:

$$\Delta w_{i,j} = \eta \, o_i^{(p)} \, \delta_j^{(p)} \tag{6.17}$$

$$
\delta_j^{(p)} = \begin{cases}
o_j^{(p)}(1 - o_j^{(p)}) \cdot (t_j^{(p)} - o_j^{(p)}) \\
\quad \text{falls } j \text{ Ausgabeneuron ist (aus 6.13 und 6.15)} \\
o_j^{(p)}(1 - o_j^{(p)}) \cdot \sum_k \delta_k^{(p)} w_{j,k} \\
\quad \text{falls } j \text{ verborgenes Neuron ist; } w_{j,k} \text{ sind die Gewichte} \\
\quad \text{zur nächsten Schicht (aus 6.13 und 6.16)}
\end{cases} \tag{6.18}
$$

6.1.3 Das Lernverhalten

Die Wahl der in 6.1.2 beschriebene Schrittweite η beim Backpropagation-Algorithmus ist für das Lernverhalten des Netzes sehr entscheidend. Wird sie groß gewählt, dann ist die Adaption der Gewichtsfaktoren zwar schnell, aber es besteht die Gefahr, dass ein Minimum des Fehlergradienten übersprungen wird. Wählt man dagegen einen kleinen Wert für η, dann werden beim Training sehr viele Iterationsschritte benötigt, bis der gewünschte Fehler erreicht wird. Dies bedeutet aber, dass eine große Menge von Trainingsmustern $\mathbf{P}$ zur Verfügung stehen muss.

Andere Möglichkeiten ergeben sich durch Modifikation des Lernverfahrens, dazu gehören Backpropagation mit Momentum, Quickprop, Counterpropagation und Cascade Correlation. Solche Lernverfahren sollen aber hier nicht weiter betrachtet werden. In der Praxis wird als Kompromiss oft ein kleiner Wert für η verwendet, dafür durchlaufen beim Training die P Trainingsmuster das Netz M-mal. M wird mit der Anzahl der Trainingszyklen bezeichnet. Es bleibt noch die Frage offen, wieviele Zyklen beim Training zu verwenden sind. Eine Möglichkeit besteht darin, das Netz solange zu trainieren, bis ein vorgegebener Fehler erreicht wird. Die Bestimmung dieser Fehlerschranke ist aber sehr kritisch. Ist sie zu hoch gewählt, ist das Netz noch nicht ausreichend trainiert und es enstehen viele Klassifikationsfehler. Ist die Fehlerschranke zu

niedrig, dann besteht die Gefahr, dass das Netz die Fähigkeit zur Generalisierung verliert, es „spezialisiert“ sich auf den Trainingsdatensatz, d.h., dass die Klassengrenzen so bestimmt werden, dass sie eng um die vorhandenen Trainingsmuster liegen.

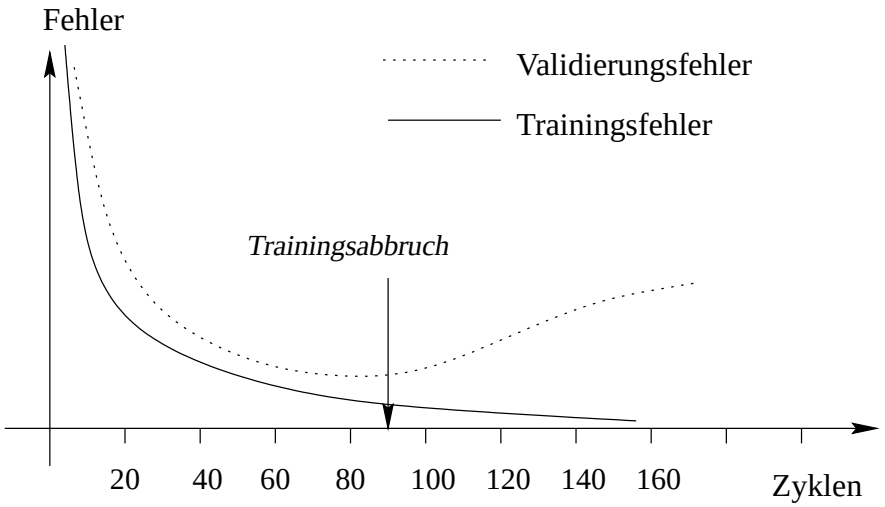


Abbildung 6.3: *Typischer Verlauf von Trainings- und Validierungsfehler*

Folgendes Abbruchkriterium wird daher verwendet: Eine Teilmenge der Trainingsdaten wird zur Validierung und nicht mehr zum Training benutzt. Nach wenigen Trainingszyklen wird der Fehler sowohl für die Trainingsdaten als auch für die Validierungsdaten bestimmt. Solange beide Fehler abklingen fährt man mit dem Training fort. Das Training ist dann genau an der Stelle abzubrechen, wo der Validierungsfehler nicht mehr abklingt. Abb. 6.3 stellt einen typischen Verlauf beider Fehler dar.

6.2 Fusionsergebnisse beim statischen Datensatz

Im letzten Kapitel wurden der Audio- und der Videokanal separat betrachtet. Es wurde gezeigt, dass eine Erkennung, die auf den Audiomerkmalen basiert, nur dann hohe Erkennungsraten erreicht, wenn das Audiosignal ein hohes SNR besitzt. In diesem Abschnitt werden einige Konzepte zur Fusion der Audio- und Videomerkmale vorgestellt, die dazu führen, dass die durchschnittlichen Erkennungsraten bei geräuschbehafteten Signalen höher werden, ohne dass die Erkennungsrate des störfreien Signals wesentlich schlechter wird.

6.2.1 Frühfusion (*early fusion*)

Die einfachste Möglichkeit, um Audio- und Videomerkmale gleichzeitig bei der Erkennung zu betrachten, besteht darin, einen neuen Vektor $\mathbf{F}$ zu definieren, dessen Elemente aus allen Elementen des Audiomerkmalsvektors $\mathbf{A}$ und des Videomerkmalsvektors $\mathbf{V}$ bestehen. Der neue Merkmalsvektor besitzt demzufolge eine Dimension L, die die Summe der Dimension M von $\mathbf{A}$ und N von $\mathbf{V}$ ist:

$$\begin{aligned} \mathbf{F} = \mathbf{A} \cup \mathbf{V} &= \{F_1, F_2, \ldots, F_L\} \\ &= \{A_1, \ldots, A_M, V_1, \ldots, V_N\}, \quad L = M + N \quad . \end{aligned} \tag{6.19}$$

Da die Fusion am Anfang *vor* der Erkennung stattfindet, handelt es sich hier um eine einfache Variante der so genannten Frühfusion (engl.: *early fusion*). In unserem konkreten Fall besitzt der neue Merkmalsvektor die Dimension 10 (6 Audiomerkmale die aus den Formantenfrequenzen und den Teilenergien bestehen sowie die 4 Videomerkmale, die in Abschnitt 5.5.1 beschrieben worden). Abbildung 6.4 zeigt die Struktur des Neuronalen Netzes in diesem Fall. Würde man sowohl die Trainings- als auch die Testvektoren nach 6.19 fusionieren, dann kann man das Netz in 6.4 direkt mit den neuen Vektoren trainieren bzw. testen.
Die Erkennungsraten, die man nach dieser simplen Fusion erhält, sind Abb. 6.5 bzw. Tabelle 6.1 zu entnehmen.

	-5dB	0dB	5dB	10dB	15dB	Orig.
Audio ER [%]	15,8	49,2	68,3	85,8	92,5	95,8
Video ER [%]	53,3	53,3	53,3	53,3	53,3	53,3
Frühfusion ER[%]	40,0	60,8	75,8	86,7	93,3	93,3

Tabelle 6.1: *Ergebnisse bei der Frühfusion*

Wir erinnern uns, dass die Grundfrequenz bei einigen der störbehafteten Signale nicht bestimmt werden konnte. Bei diesen Aufnahmen verwenden wir nur das Video-Netz. Auch hier wird nur der Fall betrachtet, bei dem das Netz ausschließlich mit den störfreien Signalen trainiert wird. Aus Abb. 6.5 wird deutlich, dass die Erkennungsraten nach der Fusion im Durchschnitt besser sind als die Erkennung anhand der Audiodaten. Lediglich die Erkennung beim störfreien Signal ist von 95,8% auf 93,3% gesunken. Bei den -5dB Signalen erhöht sich zwar die ER von 15,8% auf 40,0%, aber man hätte ein besseres Ergebnis erzielt, wenn nur der Videokanal verwendet worden wäre.

An dieser Stelle ist es sinnvoll, die **ideale Fusion** zu definieren: eine **ideale Fusion** ist diejenige, bei der die Erkennungsrate größer oder gleich der Erkennungsraten der einzelnen Kanäle ist, wenn man diese alleine betrachten würde. In einigen Literaturquellen wird eine nicht-ideale Fusion mit dem Begriff *catastrophic fusion* bezeichnet [Mov97].
Nach Abb. 6.5 ist also die Fusion sowohl bei dem störfreien Signal als auch bei einem SNR von -5dB nicht ideal. Im ersten Fall ist die Audioerkennung und im zweiten Fall die Videoerkennung besser als die nach der Fusion.

Es ist möglich, eine ideale Fusion zu zerwingen. Man kann z.B. bei Signalen mit bestimmten SNRs, bei denen man weiß, dass die Erkennung anhand eines bestimmten Kanals die besseren Ergebnisse liefert, nur noch anhand dieses Kanals auswerten. Dies ist aber keine systematische Lösung. Außerdem müsste man einen SNR-Schätzer

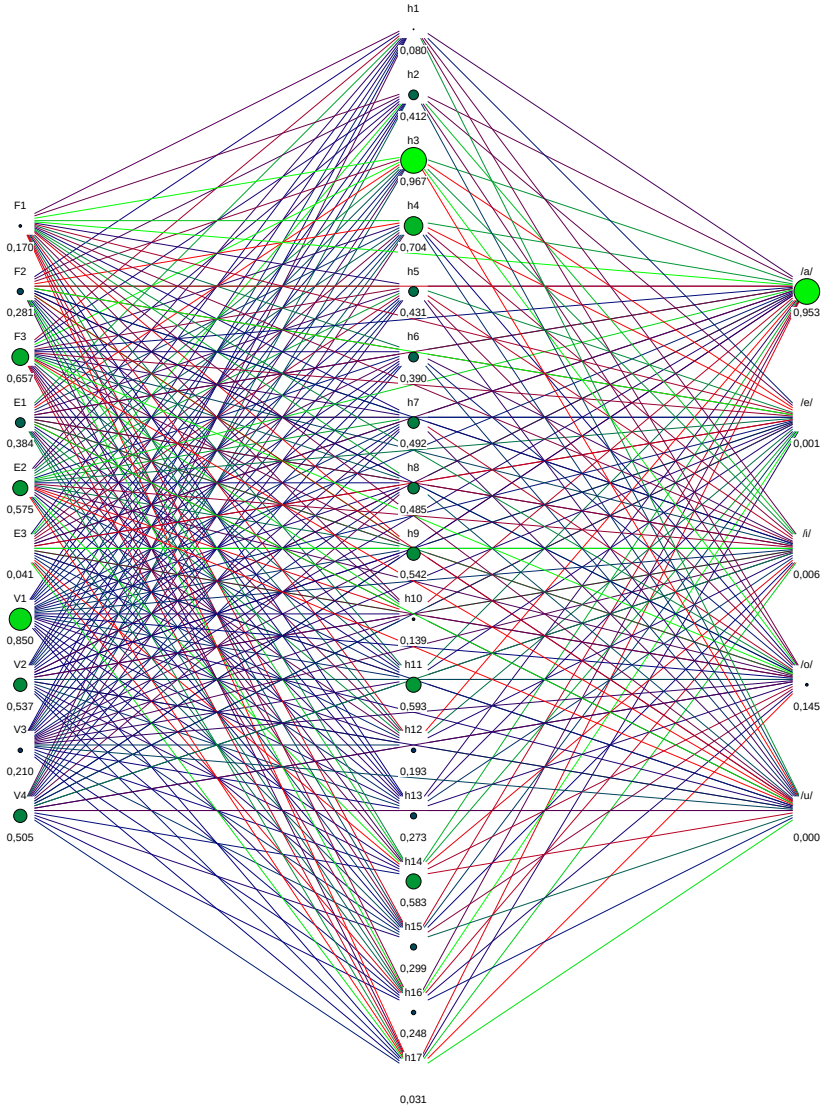


Abbildung 6.4: *Beispiel für die Frühfusion der Video- und Audiomerkmale während der Erkennung eines /a/-Vokals*

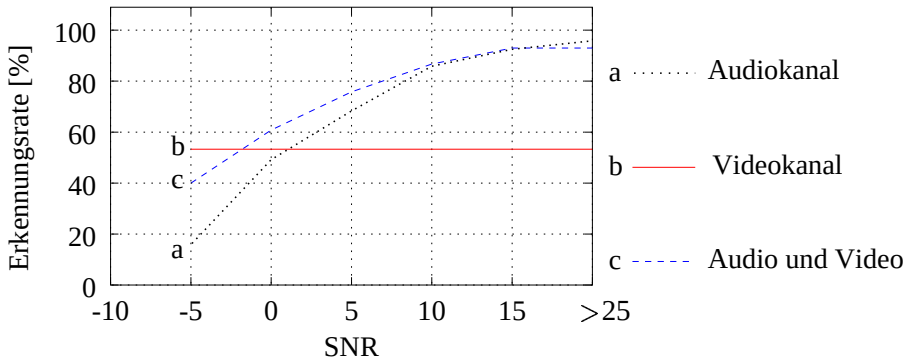


Abbildung 6.5: *Die Erkennungsrate der 5 Vokale vor und nach der Frühfusion: (a) anhand der Audiomerkmale (Abschnitt 5.4.1), (b) anhand der Videomerkmale (Abschnitt 5.5.1), (c) mit der Frühfusion*

haben, der das SNR des Testsignals dem Erkenner mitteilt. Dies ist bei dem hier verwendeten Netz nicht der Fall. Das Netz wurde nur mit den störfreien Signalen trainiert und dem Netz war es „gleichgültig“, was für ein SNR ein Testsignal besitzt.

Im folgenden Abschnitt wird in der Tat das geschätzte SNR bei der mehrversprechenden Spätfusion herangezogen. Bei der hier vorgestellten Frühfusion wurde darauf verzichtet, denn es gibt noch eine Reihe von Gründen, die eine Frühfusion weniger brauchbar machen:

- Die Frühfusion bietet keinen Mechanismus an, mit dem der Zuverlässigkeitsfaktor eines Kanals (in unserem Fall das SNR beim Audiokanal) in die Erkennerstruktur einfließen kann.
- Die Anzahl der Gewichte und somit die Netzkomplexität ist viel größer im Vergleich zu den einkanaligen Netzstrukturen.

Es existieren noch weitere Gründe, warum eine Spätfusion bevorzugt wird. Diese Gründe werden am Ende des nächsten Abschnitts, nachdem die Spätfusion vorgestellt worden ist, genauer betrachtet.

6.2.2 Spätfusion (*late fusion*)

Bei der Spätfusion setzt man für jeden Kanal einen eigenen Erkenner ein und wartet, bis alle Erkenner eine Entscheidung treffen. Diese Entscheidungen werden dann *zusammenkoordiniert*. Für die beste Fusion kann das Maximum-A-posteriori-Kriterium (MAP) verwendet werden. Demnach ist die A-posteriori-Wahrscheinlichkeit $P(M_i|\mathbf{A},\mathbf{V})$ bezüglich des Ereignisses M_i, $i = 1, 2, \ldots, N$ (in unserem Fall $N = 5 =$ Anzahl der Vokale) zu maximieren. Hier ist $\mathbf{A}$ der Audiomerkmalsvektor und $\mathbf{V}$ der Videomerkmalsvektor. Mit Hilfe der Bayes-Regel gilt für $P(M_i|\mathbf{A},\mathbf{V})$:

$$P(M_i|\mathbf{A},\mathbf{V}) = \frac{P(\mathbf{A}|\mathbf{V},M_i)\cdot P(\mathbf{V}|M_i)\cdot P(M_i)}{\sum_{j=1}^{N} P(\mathbf{A}|\mathbf{V},M_j)\cdot P(\mathbf{V}|M_j)\cdot P(M_j)} \tag{6.20}$$

Mit der Annahme, dass alle Ereignisse gleich häufig auftreten, gilt für alle Auftrittswahrscheinlichkeiten: $P(M_i) = 1/N$. Nehmen wir zusätzlich an, dass der akustische Kanal und der Videokanal voneinander statistisch unabhängig sind, dann gilt:

$$P(\mathbf{A}|\mathbf{V}, M_i) = P(\mathbf{A}|M_i) \tag{6.21}$$

Mit diesen beiden Annahmen und 6.20 vereinfacht sich das MAP-Kriterium und es ist nun folgende Größe zu maximieren:

$$\frac{P(\mathbf{A}|M_i) \cdot P(\mathbf{V}|M_i)}{\sum_{j=1}^{N} P(\mathbf{A}|M_j) \cdot P(\mathbf{V}|M_j)} \quad . \tag{6.22}$$

Für die Wahrscheinlichkeiten $P(\mathbf{A}|M_i)$ und $P(\mathbf{V}|M_i)$ können die Ausgangswerte des jeweiligen Netzes verwendet werden. Bei der Erkennung anhand eines einzigen Kanals wurde nach der Ausgangseinheit mit der maximalen Aktivierung gesucht. Hier müssen wir aber die Ausgangsaktivierungen des Audiokanals $\mathbf{h}_A$ und die des Videokanals $\mathbf{h}_V$ gleichzeitig betrachten. Man kann also $\mathbf{h}_A$ als $P(\mathbf{A}|M_i)$ und $\mathbf{h}_V$ als $P(\mathbf{V}|M_i)$ verwenden. In diesem Fall ist der Nenner gemeinsam bei allen Entscheidungen und kann wegfallen. Damit reduziert sich die Entscheidung auf die Maximierung des Produktes: $h_{A,i} \cdot h_{V,i}$.

Da aber eine Addition bei einem Neuronalen Netz einfach durch eine weitere Schicht (die so genannte Kombinationsschicht) implementiert werden kann (Gleichung 6.1), bietet es sich an, stattdessen die Summe $h_{A,i} + h_{V,i}$ zu maximieren. Es ist von großer Bedeutung zu beachten, dass es sich dabei um *keine* übliche Schicht handelt. Die Kombinationsschicht wird nämlich *nicht* während der Trainingsphase verwendet. In der Trainingsphase wird jedes Netz alleine trainiert. Nur während der eigentlichen Erkennung (der Testphase) werden dann beide Netze durch die Kombinationsschicht fusioniert.

Bei der Spätfusion bietet sich nun ein sehr einfaches Konzept zur Berücksichtigung des geschätzten SNRs an: man kann die Ausgänge $\mathbf{h}_A$ und $\mathbf{h}_V$ unterschiedlich gewichten, in diesem Fall ist die Summe

$$g_{AV,i} = \alpha.h_{A,i} + (1 - \alpha)h_{V,i} \tag{6.23}$$

zu maximieren. Der Gewichtsfaktor α wird von dem geschätzten SNR abhängig gemacht und besitzt einen Wert zwischen 0 und 1. Bei einem sehr zuverlässigen Audiokanal mit hohem SNR nimmt α hohe Werte an, bei niedrigem SNR nimmt α dagegen kleinere Werte an. Der Verlauf der Funktion $\alpha = f(\text{SNR})$ wurde in dieser Arbeit heuristisch ermittelt. Abbildung 6.6 zeigt eine mögliche Struktur für die Spätfusion bei der Vokalerkennung. Man erkennt, dass das gesamte Netz aus zwei einfachen Feed-forward-Netzen besteht, deren Ausgänge die Kombinationsschicht aktivieren. Das obere Netz ist im Grunde identisch mit dem Audionetz, das bereits in Abschnitt 5.4.1 vorgestellt wurde. Genauso ist das untere Netz identisch mit dem in Abschnitt 5.5.1 vorgestellten Videonetz.

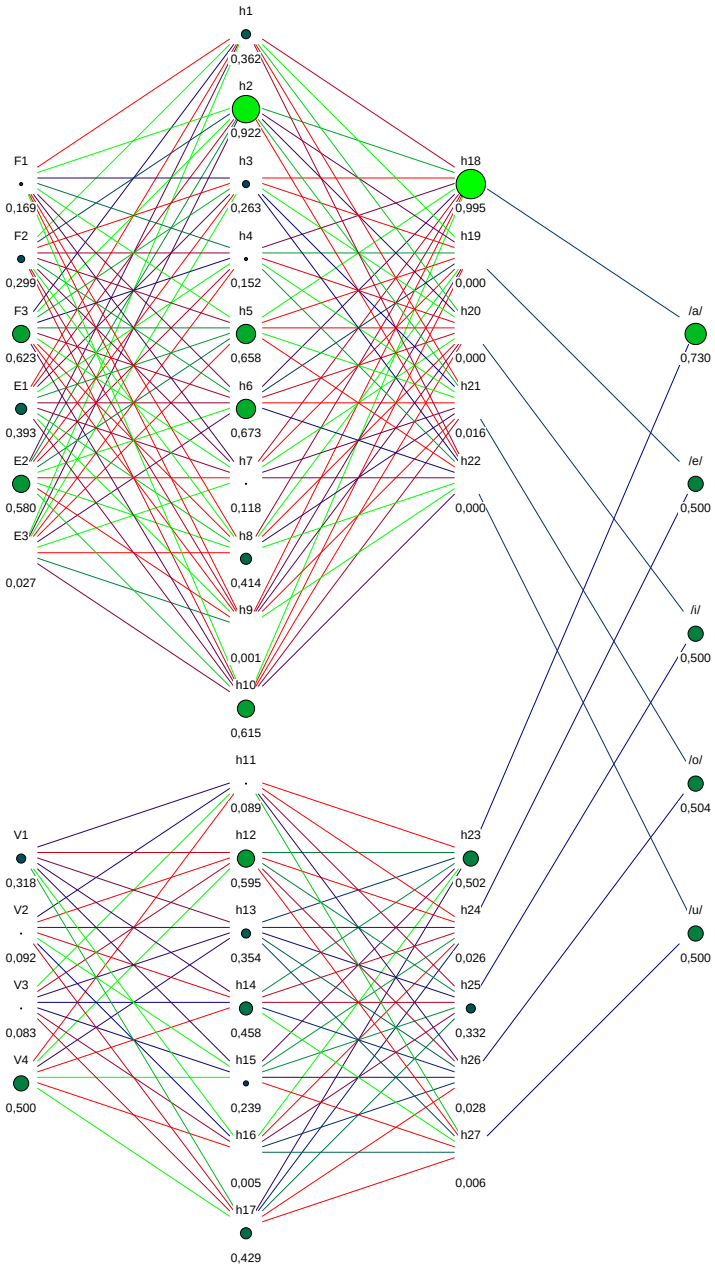


Abbildung 6.6: *Beispiel für die Spätfusion des Video- und Audionetzes*

Würde man sich z.B. das untere Netz und die letzte Schicht „wegdenken“, erhielte man genau die gleiche Netzstruktur und die gleichen Erkennungsergebnisse von Abschnitt 5.4.1.

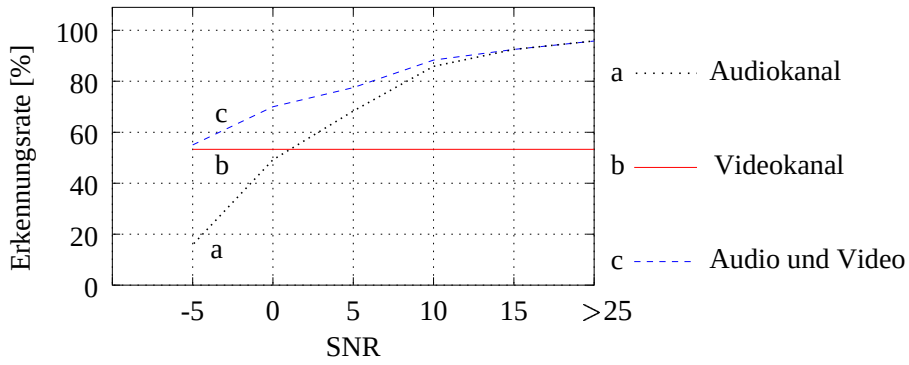


Abbildung 6.7: *Die Erkennungsrate der 5 Vokale vor und nach der Spätfusion: (a) anhand der Audiomerkmale (Abschnitt 5.4.1), (b) anhand der Videomerkmale (Abschnitt 5.5.1), (c) mit der Spätfusion*

Die Erkennungsraten, die man mit diesem Netz erhält, sind in Diagramm 6.7 dargestellt. Die genauen Zahlen sind in Tabelle 6.2 eingetragen. In der letzten Zeile ist das Gewichtungsverhältnis zwischen den Audio- und Videoausgängen eingetragen. Bei einem SNR von z.B. 0dB beträgt dieses Verhältnis 1:2, dies entspricht einen Wert von $\alpha = 1/(1+2) \approx 0,333$. Diese Werte waren experimentiell ermittelt worden.

	-5dB	0dB	5dB	10dB	15dB	Orig.
Audio ER [%]	15,8	49,2	68,3	85,8	92,5	95,8
Video ER [%]	53,3	53,3	53,3	53,3	53,3	53,3
Spätfusion ER[%]	55,0	70,0	77,5	88,3	92,5	95,8
Verhältnis A:V	1:100	1:2	1:1	1:1	2:1	∞:1

Tabelle 6.2: *Ergebnisse bei der Spätfusion*

In dieser Tabelle fällt auf, dass bei keinem SNR eine *katastrophale Fusion* aufgetreten ist. Die multimodale Erkennungsrate ist also durch eine entsprechende Wahl von α immer besser als die des Audio- und des Videokanals.
Soweit sind die Vorteile der Spätfusion noch nicht erschöpft. Man kann die Verbindungen zwischen den Ausgangsneuronen und der Kombinationsschicht von der Erfolgsquote jeder einzelnen Klasse abhängig machen. Nach der Verwechslungstabelle des Audioerkenners bei einem SNR von 5dB liegt z.B. die Erfolgsquote des Vokals /u/ bei 6,25% und ist somit deutlich niedriger als die durchschnittliche Erkennungsrate aller Vokale von 68,33%. Die Erfolgsquote des Videokanals beim Vokal /u/ liegt dagegen bei 87,5% und ist deutlich höher als der Durchschnitt dort von 55,33%. Es ist also sinnvoll, die entsprechende Videoverbindung zur Kombinationsschicht gegenüber der

Audioverbindung stärker zu gewichten. Durch diese Maßnahme konnten die Erkennungsraten der Tabelle 6.2 um weitere 1% bis 3% erhöht werden. Hier muss man aber noch erwähnen, dass eine solche Gewichtung nur bei größeren Datensätzen einen Sinn hat, denn nur dort kann man eine sichere Statistik der Erfolgsquoten erhalten.

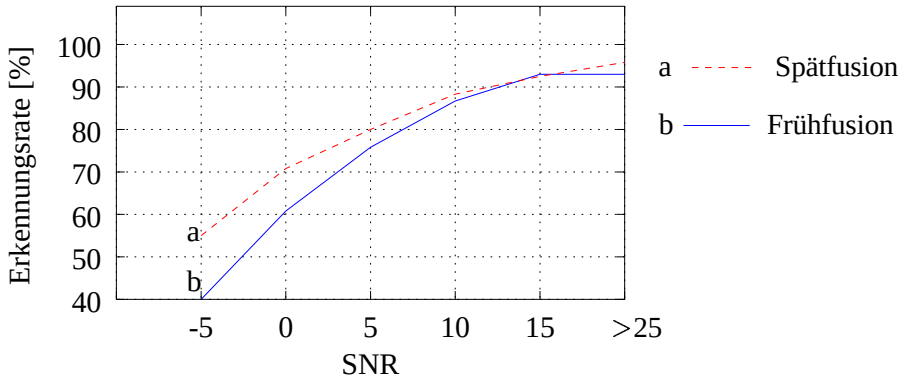


Abbildung 6.8: *Die Erkennungsrate der 5 Vokale: (a) mit der Frühfusion, (b) mit der Spätfusion*

Abbildung 6.8 zeigt die Ergebnisse der Spätfusion nach dieser Maßnahme im Vergleich zu den Ergebnissen bei der Frühfusion. Man kann die Vorzüge der Spätfusion gegenüber der Frühfusion wie folgt zusammenfassen:

- Jedes einzelne Netz kann unabhängig von dem anderen dimensioniert und trainiert werden. Dies ist ein nicht zu vernachlässigender Aspekt, denn man benötigt in der Tat eine ganz unterschiedliche Anzahl von Iterationsschritten beim Training der beiden Netze. Außerdem kann man für jeden Kanal eine eigene Struktur wählen: dies beinhaltet nicht nur die Wahl der Anzahl der verdeckten Schichten, sondern auch die Definition des Lernalgorithmus und die Art der Verbindungen zwischen den Einheiten. Man kann sogar ganz verschiedene Erkenner einsetzen, z.B. ein Neuronales Netz für die audiobasierte Erkennung und ein Nächster-Nachbar-Klassifikator für die Videoerkennung.

- Die Zuverlässigkeitsfaktoren für jeden Kanal können auf einfache Weise bei der Erkennung berücksichtigt werden.

- Die gesamte Komplexität eines spätfusionierten Netzes ist in der Regel geringer als die des frühfusionierten Netzes.

- Die Erkennungsraten einer Spätfusion sind durch korrekte Wahl der Zuverlässigkeitsfaktoren höher als die der Frühfusion. Man kann immer eine ideale Fusion erzwingen.

Neben der Spätfusion und der in Abschnitt 6.2.1 vorgestellten Frühfusion existieren noch weitere Mischformen. Ein Neuronales Netz, bei dem die Fusion an einer verborgenen Schicht stattfindet, ist denkbar. Auf die Untersuchung solcher Formen wurde verzichtet.

6.3 Fusionsergebnisse beim dynamischen Datensatz

Auch bei der Fusion der Audio- und Videodaten bei der Erkennung der französischen Wörter bieten sich ähnliche Fusionskonzepte wie beim statischen Datensatz an. Abbildung 6.9 zeigt drei solche Möglichkeiten. Bei der Frühfusion „klebt“ man beide Merkmale aneinander und modelliert dann die so entstandene neue Merkmalsfolge. Ein so genannter *Boltzmann Zipper* [Sto96] stellt eine Zwischenform der Früh- und Spätfusion dar. Dieser besteht aus zwei linearen Einheitsketten mit internen Kreuzverbindungen. Jede Kette darf eine unterschiedliche Anzahl von Zuständen besitzen. Ganz rechts in der Abbildung wird ein Beispiel für eine Spätfusion gezeigt, bei dem zwei unabhängige HM-Modelle für jeweils die Audio- und die Videoerkennung erst zum Schluss durch die Funktion $g_{AV,i}$ (s. Gl. 6.23) fusioniert werden.

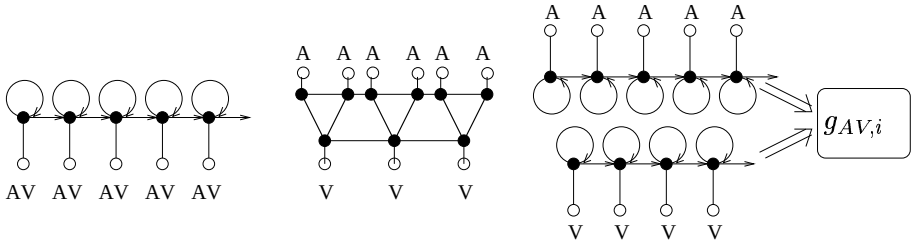


Abbildung 6.9: *Mögliche Fusionsstufen beim HMM: Frühfusion (links), Boltzmann-Zipper (Mitte) und Spätfusion (rechts)*

6.3.1 Frühfusion (*early fusion*)

Bei der Frühfusion stößt man hier auf ein Problem, das uns bei der Fusion des statischen Datensatzes erspart geblieben ist. Wir betrachten hier nämlich dynamische Ereignisse, die eine bestimmte Taktverarbeitung verlangen. Für die Sprachverarbeitung hängt dieser Takt von der Abtastfrequenz f_A, der verwendeten Fensterlänge N und der Verschiebung dieses Fensters S ab. Beim Videokanal ist der Takt durch die Anzahl der Bilder pro Sekunde festgelegt. Bei einer Frühfusion muss also für einen einheitlichen Takt gesorgt werden, ansonsten müsste man beide Merkmale durch aufwendige Interpolationsmethoden zeitlich aneinander anpassen.

	-5dB	0dB	5dB	10dB	15dB	Orig.
Audio WER [%]	13,6	22,7	18,8	40,9	63,6	95,5
Video WER [%]	54,6	54,6	54,6	54,6	54,6	54,6
Fusion WER[%]	22,7	31,8	45,5	63,6	100	100

Tabelle 6.3: *Ergebnisse bei der Frühfusion*

Dieses Problem entfällt bei der Spätfusion, was für diese Methode ein weiterer Vorteil ist. Da die Bildrate beim Videokanal mit 25 Bildern/s beim PAL-System vorgegeben

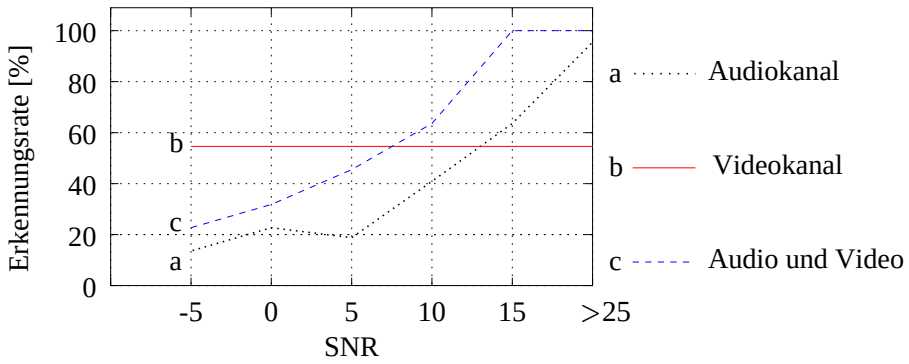


Abbildung 6.10: *Erkennungergebnisse des dynamischen Datensatzes: (a) anhand der Audiomerkmale (Abschnitt 5.4.2), (b) anhand der Videomerkmale (Abschnitt 5.5.2), (c) mit der Frühfusion*

ist, bleibt uns nichts anderes übrig, als die Fensterlänge N und die Fensterverschiebung S der Sprachverarbeitung so zu wählen, dass ein gleicher Takt erzwungen wird. Die Möglichkeiten dafür sind sehr beschränkt, da nur bestimmte Werte für N üblich sind. Eine mögliche Lösung ist $N = 512$ und $S = 192$. Somit ist die effektive Fensterlänge $= 512 - 192 = 320$, bei einer Abtastrate von $f_A = 8$ kHz ergibt sich eine effektive Fensterlänge von $320/8000Hz = 40$ ms, was genau der Dauer eines Bildes entspricht. Abbildung 6.10 und Tabelle 6.3 zeigen die Ergebnisse der Frühfusion. Für die SNR-Werte 10dB, 15dB und für das Originalsignal verbessert sich die WER deutlich. Es handelt sich bei diesen Fällen um eine ideale Fusion. Bei -5dB, 0dB und 5dB ist zwar die WER besser als die des Audiokanals, aber eine videobasierte Erkennung hätte besser abgeschnitten. Hier wird dem Erkenner nichts über das SNR des Testsignals mitgeteilt, d.h., für ihn fehlt diese Information und kann dementsprechend nicht berücksichtigt werden.

6.3.2 Spätfusion (*late fusion*)

Das Konzept der Spätfusion wurde in Zusammenhang mit den Neuronalen Netzen bei der Erkennung der Vokale bereits vorgestellt (Abschnitt 6.2.2). Für die Wahrscheinlichkeiten $P(\mathbf{A}|M_i)$ und $P(\mathbf{V}|M_i)$ wurden die Ausgangswerte des jeweiligen Netzes verwendet.
Bei der Spätfusion des dynamischen Datensatzes werden HM-Modelle statt der Neuronalen Netze verwendet, daher gibt es keine Ausgangswerte, die man bei der Fusion nach Gl. 6.23 verwenden könnte. Zum Glück liefert aber der in Abschnitt 5.2.2 erwähnte Viterbi-Algorithmus für jedes Testwort die Wahrscheinlichkeit $f(\mathcal{O}|\boldsymbol{\lambda}_i)$, dass ein bestimmtes Wort zu einem bestimmten HM-Model $\boldsymbol{\lambda}_i$ gehört. Hier kann man also für $P(\mathbf{A}|M_i)$ die Wahrscheinlichkeit $f(\mathcal{O}_A|\boldsymbol{\lambda}_i)$ und für $P(\mathbf{V}|M_i)$ die Wahrscheinlichkeit $f(\mathcal{O}_V|\boldsymbol{\lambda}_i)$ verwenden.

Diese beiden Wahrscheinlichkeiten werden nach Gl. 6.23 in Abhängigkeit vom SNR unterschiedlich gewichtet. Über die Zuordnung eines Wortes zu einer bestimmten

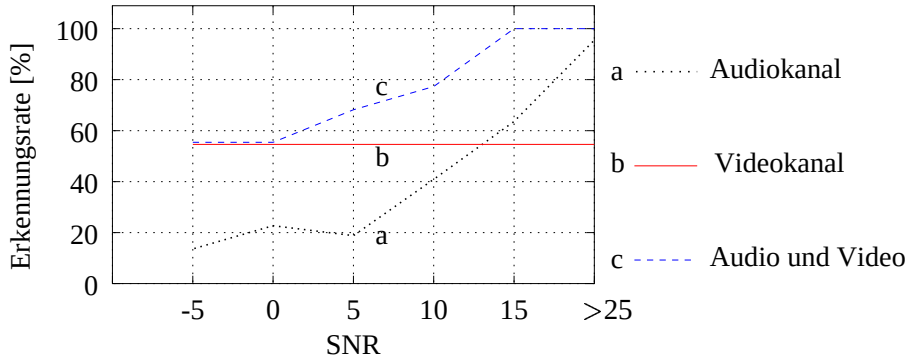


Abbildung 6.11: *Die Erkennungsrate der 11 Wörter vor und nach der Spätfusion: (a) anhand der Audiomerkmale (Abschnitt 5.4.2), (b) anhand der Videomerkmale (Abschnitt 5.5.2), (c) mit der Spätfusion*

	-5dB	0dB	5dB	10dB	15dB	Orig.
Audio WER [%]	13,6	22,7	18,8	40,9	63,6	95,5
Video WER [%]	54,6	54,6	54,6	54,6	54,6	54,6
Fusion WER[%]	54,6	54,6	68,2	77,3	100	100
Verhältnis A:V	1:∞	1:50	1:50	1:20	1:3	10:1

Tabelle 6.4: *Ergebnisse bei der Spätfusion*

Klasse entscheidet $g_{AV,i}$.
Der Gewichtsfaktor α ist hier ebenfalls vom SNR abhängig. Seine Werte wurden experimentell bestimmt. Die Erkennungsraten der Spätfusion im Vergleich zu den WER anhand des Audio- bzw. Videokanals sind in Abbildung 6.11 und Tabelle 6.4 dargestellt. Bei allen SNRs ist die WER nach der Fusion besser als die der einzelnen Kanäle. Es handelt sich hier also um eine ideale Fusion. In der letzten Zeile von Tabelle 6.4 sind die α-Werte aufgelistet.

Abbildung 6.12 vergleicht die Frühfusion mit der Spätfusion.

6.4 Zusammenfassung

In diesem Kapitel wurde zunächst das Konzept der Erkennung mit Hilfe von Feedforward-Netzstrukturen erläutert. Für das Training des Neuronalen Netzes kam der Backpropagation-Algorithmus zum Einsatz. Auf kompliziertere Lernalgorithmen wurde verzichtet, da solche an erster Stelle nur der Beschleunigung der Lernphase dienen. Man kann zwar durch solche Algorithmen die Anzahl der Iterationen (Lernzyklen) verringern, ein kleinerer Lernfehler als der der Backpropagation wird dadurch allerdings nicht unbedingt erreicht, dafür werden aber mehr Lernparameter benötigt, deren Einstellung viel Erfahrung benötigt.

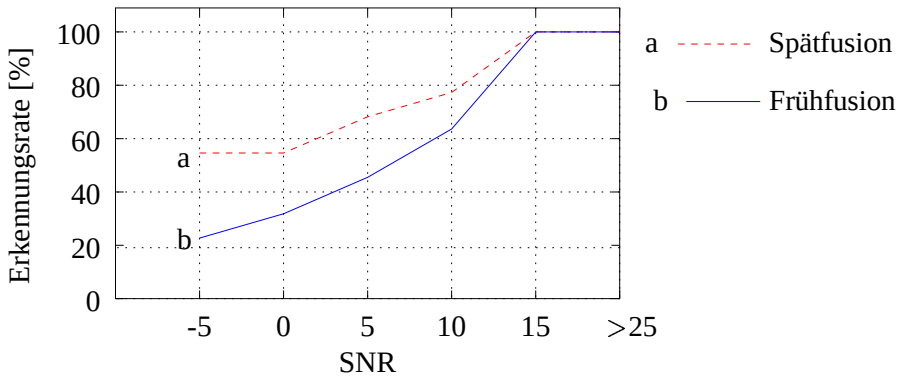


Abbildung 6.12: *Vergleich zwischen der (a) Frühfusion und (b) Spätfusion bei der Erkennung der 11 Wörter.*

Erkennungsergebnisse anhand eines einzigen Kanals, bei dem Neuronale Netze eingesetzt werden, sind bereits in Kapitel 5 vorgestellt worden. In diesem Kapitel wurden einige Konzepte zur Fusion der beiden Kanäle dargestellt. Es wurde gezeigt, dass die multimodale Erkennung der audiobasierten Erkennung überlegen ist.
Bei der Merkmalsextraktion aus dem Videokanal wurden nur 4 Merkmale verwendet, die die Außenkonturen der Lippen modellieren. Die Verarbeitung war so konzipiert worden, dass eine Echtzeitextraktion dieser Merkmale möglich war. Was zu Beginn und während der Arbeit als sehr rechenintensiv galt, ist mittlerweile nicht mehr der Fall. Damit ist es möglich, die Bildverarbeitung in kommenden Arbeiten um weitere Merkmale zu erweitern, ohne dass die Echtzeitbedingungen beeinträchtigt werden. Denkbar ist z.B. die Betrachtung der Fläche innerhalb der Lippenkonturen.

Anhang A

Die Rechteck-Konfiguration

Bei der in Abbildung 2.3 gezeigten Arraygeometrie haben die vier Mikrofone folgende verallgemeinerte Koordinaten: $M_0(0,0,0)$, $M_1(a_1,b_1,0)$, $M_2(a_2,b_2,0)$, $M_3(a_3,b_3,0)$. Mit den Koordinaten der Mikrofone und der gesuchten Position $P(x,y,z)$ des Sprechers lassen sich die drei Entfernungsdifferenzen wie folgt berechnen:

$$\begin{aligned}
\Delta l_{10} &= \sqrt{x^2+y^2+z^2} - \sqrt{(x-a_1)^2+(y-b_1)^2+z^2} \\
\Delta l_{20} &= \sqrt{x^2+y^2+z^2} - \sqrt{(x-a_2)^2+(y-b_2)^2+z^2} \\
\Delta l_{30} &= \sqrt{x^2+y^2+z^2} - \sqrt{(x-a_3)^2+(y-b_3)^2+z^2} \quad .
\end{aligned}$$

Sie werden durch Laufzeitdifferenzen geschätzt.
Nach der Umformung der obigen Gleichungen ergeben sich drei Gleichungen für $l_0 = \sqrt{x^2+y^2+z^2}$:

$$l_0 = \frac{\Delta l_{10}^2 + 2a_1x + 2b_1y - (a_1^2+b_1^2)}{2\Delta l_{10}} \tag{A.1}$$

$$l_0 = \frac{\Delta l_{20}^2 + 2a_2x + 2b_2y - (a_2^2+b_2^2)}{2\Delta l_{20}} \tag{A.2}$$

$$l_0 = \frac{\Delta l_{30}^2 + 2a_3x + 2b_3y - (a_3^2+b_3^2)}{2\Delta l_{30}} \quad . \tag{A.3}$$

Durch Subtraktion (A.2 - A.3) und (A.2 - A.3) erhält man:

$$y = \frac{w_{12}x + v_{12}}{u_{12}} \tag{A.4}$$

$$y = \frac{w_{13}x + v_{13}}{u_{13}} \quad , \tag{A.5}$$

die einen Zusammenhang zwischen x und y darstellen. Die Variablen u_{12}, u_{13}, w_{12}, w_{13}, v_{12} und v_{13} ergeben sich dabei zu:

$$
\begin{aligned}
u_{12} &= 2b_1 \cdot \frac{\Delta l_{20}}{\Delta l_{10}} - 2b_2 \\
u_{13} &= 2b_1 \cdot \frac{\Delta l_{30}}{\Delta l_{10}} - 2b_3 \\
w_{12} &= 2a_2 - 2a_1 \cdot \frac{\Delta l_{20}}{\Delta l_{10}} \\
w_{13} &= 2a_3 - 2a_1 \cdot \frac{\Delta l_{30}}{\Delta l_{10}}
\end{aligned}
$$

$$
\begin{aligned}
v_{12} &= \Delta l_{20}^2 - \Delta l_{10}\Delta l_{20} + (a_1^2 + b_1^2) \cdot \frac{\Delta l_{20}}{\Delta l_{10}} - (a_2^2 + b_2^2) \\
v_{13} &= \Delta l_{30}^2 - \Delta l_{10}\Delta l_{30} + (a_1^2 + b_1^2) \cdot \frac{\Delta l_{30}}{\Delta l_{10}} - (a_3^2 + b_3^2)
\end{aligned}
$$

und sind nur von den jeweiligen Entfernungsdifferenzen abhängig. Die Gleichungen A.4 und A.5 lassen sich nach den Sprecherkoordinaten auflösen. Sie ergeben sich zu:

$$x = \frac{u_{12}v_{13} - u_{13}v_{12}}{w_{12}u_{13} - w_{13}u_{12}} \tag{A.6}$$

$$y = \frac{w_{12}}{u_{12}} \cdot \frac{u_{12}v_{13} - u_{13}v_{12}}{w_{12}u_{13} - w_{13}u_{12}} + \frac{v_{12}}{u_{12}} \tag{A.7}$$

$$z = \sqrt{\frac{\Delta r_{10}^2 + 2a_1x + 2b_1y - (a_1^2 + b_1^2)^2}{2\Delta r_{10}} - x^2 - y^2} \quad . \tag{A.8}$$

Alle drei Gleichungen für x, y, z haben den Faktor

$$(w_{12}u_{13} - w_{13}u_{12}) \quad , \tag{A.9}$$

im Nenner gemeinsam.

Anhang B

Polynomnäherung der Snake-Krümmungsenergie

B.1 Berechnung des Annährungspolynoms

Die Polynomnäherung dient dazu, eine Kurve $\mathbf{v}(s)$, von der nur N Punkte $\mathbf{P}_j \quad , j = 1, \ldots, N$ bekannt sind, durch ein Polynom des Grades $N-1$ anzunähern. In unserem Fall lassen sich drei aufeinanderfolgende Punkte der Snake-Kurve mit einem Polynom zweiten Grades jeweils für x- und y-Komponete beschreiben. Mit $\mathbf{z}_j(t)$ wird im folgenden die Kurve bezeichnet, die man durch die Approximation erhält.
Es wird die Kurve

$$\mathbf{z}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} a_0 + a_1 t + a_2 t^2 \\ b_0 + b_1 t + b_2 t^2 \end{pmatrix} \tag{B.1}$$

durch die drei Punkte

$$\begin{aligned} \mathbf{P}_1 &= \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \\ \mathbf{P}_2 &= \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} \\ \mathbf{P}_3 &= \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} \end{aligned} \tag{B.2}$$

gelegt. t stellt hier den Kurvenparameter dar und nimmt die Werte 0 (bei $\mathbf{P}_1$) bis 1 (bei $\mathbf{P}_3$) an. Dabei müssen folgende Bedingungen erfüllt werden:

$$\mathbf{z}(t=0) \stackrel{!}{=} \mathbf{P}_1 \tag{B.3}$$

$$\mathbf{z}(t=0,5) \stackrel{!}{=} \mathbf{P}_2 \tag{B.4}$$

$$\mathbf{z}(t=1) \stackrel{!}{=} \mathbf{P}_3 \quad . \tag{B.5}$$

Hieraus erhält man folgende Gleichungen für die x-Komponente von $\mathbf{z}(t)$:

$$a_0 = x_1 \tag{B.6}$$
$$a_0 + 0,5a_1 + 0,25a_2 = x_2 \tag{B.7}$$
$$a_0 + a_1 + a_2 = x_3 \quad . \tag{B.8}$$

Oder in Matrixschreibweise:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0,5 & 0,25 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad . \tag{B.9}$$

Durch wenige Zeilenoperationen erhält man das Gleichungssystem:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0,5 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ -3x_1 + 4x_2 - x_3 \\ x_1 - 2x_2 + x_3 \end{bmatrix} \tag{B.10}$$

Hieraus ergeben sich die Koeffizienten zu:

$$a_0 = x_1 \tag{B.11}$$
$$a_1 = -3x_1 + 4x_2 - x_3 \tag{B.12}$$
$$a_2 = 2x_1 - 4x_2 + 2x_3 \tag{B.13}$$

Analog dazu ergeben sich die Koeffizienten für die Y-Komponente von $\mathbf{z}(t)$:

$$b_0 = y_1 \tag{B.14}$$
$$b_1 = -3y_1 + 4y_2 - y_3 \tag{B.15}$$
$$b_2 = 2y_1 - 4y_2 + 2y_3 \tag{B.16}$$

Insgesamt kommt man zu diesem Ergebnis für $\mathbf{z}(t)$:

$$\mathbf{z}(t) = \begin{pmatrix} x_1 + (-3x_1 + 4x_2 - x_3)t + (2x_1 - 4x_2 + 2x_3)t^2 \\ y_1 + (-3y_1 + 4y_2 - y_3)t + (2y_1 - 4y_2 + 2y_3)t^2 \end{pmatrix} \tag{B.17}$$

B.2 Berechnung der Krümmung

Die Krümmung von beliebig parametrisierten Kurven $\mathbf{z}(t)$ mit $x(t)$ als x-Komponente und $y(t)$ als y-Komponente berechnet man durch

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{\left(\dot{x}^2(t) + \dot{y}^2(t)\right)^{3/2}}. \tag{B.18}$$

Man benötigt also die erste und zweite Ableitung von $\mathbf{z}(t)$. Sie ergeben sich aus dem vorherigen Abschnitt zu:

$$\dot{x}(t) = -3x_1 + 4x_2 - x_3 + (4x_1 - 8x_2 + 4x_3)t \quad \text{(B.19)}$$
$$\ddot{x}(t) = 4x_1 - 8x_2 + 4x_3 \quad \text{(B.20)}$$
$$\dot{y}(t) = -3y_1 + 4y_2 - y_3 + (4y_1 - 8y_2 + 4y_3)t \quad \text{(B.21)}$$
$$\ddot{y}(t) = 4y_1 - 8y_2 + 4y_3 \quad \text{(B.22)}$$

Durch Einsetzen in Gl. B.18 kann man die Krümmung für ein beliebiges t berechnen. Das Ergebnis soll hier nur für $t = 0,5$ angegeben werden:

$$\kappa(t = 0,5) = 8 \cdot \frac{x_1(y_2 - y_3) - x_2(y_1 - y_3) + x_3(y_1 - y_2)}{\left((x_1 - x_3)^2 + (y_1 - y_3)^2\right)^{3/2}} \quad \text{(B.23)}$$

Literaturverzeichnis

[Abr74] Abramov, I., Gordon, J., *"Vision"*, in Carterette, E.C., Friedman, M.P. (eds), *"Handbook of Perception: Biology of Perceptual Systems"*, vol.3, Academic, S. 327-406, New York, 1974.

[Adj95] Adjondani, A. und Benoit, C., *"Audio-visual speech recognition compared across two architectures"*, Proceedings of the Eurospeech'95 Conference, volume 2, S. 1563-1566, Madrid, Spain, 1995.

[Ami88] Amini, A. A.; Tehrani, S. und Weymouth, T .E. *"Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints "*, Proc. Second International Conference on Computer Vision, S. 95-99, 1988.

[Bal99] Baldwin, J.F., Martin, T.P., Saeed, M., *"Automatic computer lip-reading using fuzzy set theory"*, Proceedings of Auditory-Visual Speech Processing (AVSP) Aug. 1999.

[Bla73] Blakmore, C., *"The Baffled Brain"*, in R.L. Gregory and E.H. Gombrich (eds.), *"Illusions in Nature and Art"*, Duckworth, London, S.9-48, 1973.

[Ber99] Berthommier, F., Mekhaiel M.S., Kroschel K., *"Speaker localisation with 4 microphones: study of the cross configuration "*, IAR-Jahresbericht, ENSAIS, Strasbourg 1999

[Bra91] Brause, R., *"Neuronale Netze"*, Teubner, Stuttgart, 1991.

[Bre95] Bregler C., Omohundro S. M., *"Nonlinear Manifold Learning for Visual Speech Recognition"*, IEEE Proceedings of International Conference on Computer Vision, 1995.

[Chi96] Chiou, G.I., Hwang, J-N., *"Lipreading by using snakes, principal component analysis, and hidden Markov models to recognize color motion video"*, IEEE Transactions on Image Processing, 1996.

[Coi95] Coianiz T, Torresani L., Caprile B., *"2D Deformable Models for Visual Speech Analysis"*, Speechreading by Humans and Machines, Springer-Verlag 1995; proceedings of the NATO Advanced Study Institute: Speechreading by Man and Machine, 1995.

[Cos95] Cosi, P., Dugatto, M., Ferrero, F.E., Magno, E., Vagges, K., *"Bimodal Recognition of Italian Plosives"*, Proceedings of XIII International Congress of Phonetic Sciences, ICPhS-95, Stochkolm, Vol. 4, S. 260-263, 1995.

[Cro97] Crowley J. und Bérard F., *"Multi-Modal Tracking of Faces for Video Communications"*, IEEE Conference on Computer Vision und Pattern Recognition, CVPR '97, Puerto Rico, June 17-19, 1997.

[Dav78] Davis, S.B. und Mermelstein, P., *"Evaluation of Acoustic Parameters for Monosyllabic Word Identification"*, JASA, Vol. 64, Suppl. 1, S. 180-181, 1978.

[Fin86] Finn, K. E., *"An Investigation of Visible Lip Information to be Used in Automated Speech Recognition"*, Ph.D. Thesis, Georgetown University, 1986.

[Fou98] Foucher, E., Girin, L., Feng, G., *"Audiovisual speech coder : using vector quantization to exploit the audio/video correlation"*, Proceedings of Auditory-Visual Speech Processing (AVSP) 1998.

[Fri92] Fries, G., *"Lautspezifische Synthese von Sprache im Zeit- und Frequenzbereich"*, VDI-Fortschrittsberichte, Reihe 10, Nr.213, VDI-Verlag, 1992.

[Fur89] Furui, S., *"Digital Speech Processing, Synthesis and Recognition"*, Marcel Dekker, New York, Basel, 1989.

[Hal98] Hallgren, A., Lyberg, B., *"Lip movements in non-focal and focal position for visual speech synthesis"*, Proceedings of Auditory-Visual Speech Processing (AVSP) 1998.

[Hen97] Hennecke, M.E., *"Audio-visual speech recognition: preprocessing, learning and sensory integration"*, Ph. D. Dissertation, Dept. of Electrical Eng., Stanford Univerity, Sept. 1997.

[Hil95] Hillenbrand, J., et.al., *"Acoustic characteristics of American english vowels"*, Journal Acoust. Soc. Am., Vol. 97/5, S. 3099-3111, 1995.

[Hua90] Huang, X.D.;Ariki, Y.;Jack, M.A., *" Hidden Markov Models for Speech Recognition"*, Edinburgh: Edinburgh University Press, 1990.

[Gany] *"http://www.hypervision.co.uk/motion/index.htm, http://www.ganymedia.com/ "*.

[Gir97] Girin, L., *"Noisy speech enhancement with a filtering process using lip shape information"*, Ph.D. Thesis, Institut de la Communication Parlée, INPG/ENSERG/Université Stendhal, Grenoble, France

[Gir98] Girin, L., Feng G., Schwartz J.-L., *"Fusion of auditory and visual information for noisy speech enhancement: a preliminary study of vowel transitions"*, Proc. of the 23rd IEEE International Conference on Acoustics, Speech, and Signal Processing, Seattle, USA, 1998.

[Gol93] Goldschen, A.J., *"Continuous Automatic Speech Recognition by Lipreading"*, Ph.D. Dissertation, George Washington University, Washington, D.C., September 1993.

[Gra97] Gray, M., Movellan, J. R., Sejnowski, T., *"Dynamic features for visual speechreading: A systematic comparison"*, in: Mozer, Jordan, Petsche (Eds.), Advances in Neural Information Processing Systems 9, 1997.

[Gün96] Günter, Armin, *"Zeitliche Begrenzung von Vokalen in fließender Sprache"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Mai 1996.

[Kas87] Kass M., Witkin A. und Terzopoulos D., *"Snakes: Active Contour Models"*, IEEE Proceedings of International Conference on Computer Vision, S. 259-268, London, 1987.

[Kel91] Keller, M., *"Einzelworterkennung für Sprache in gestörter Umgebung"*, Dissertation, Universität Karlsruhe, 1991.

[Kob94] Kobler, J., *"Minimum-Varianz-Schätzer zur Geräusch-Reduktion bei der Einzelworterkennung"*, Dissertation, Universität Karlsruhe, 1994.

[Kor97] Kornmeier, Klaus, *"Modellieren der Lippenkonturen in Bildsequenzen durch Deformable Contour Models (DCM)"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, März 1997.

[Kro89] Kroschel K., Kammeyer K., *"Digitale Signalverarbeitung"*, Teubner Verlag, Stuttgart, 1989.

[Kro93] Kroschel, K.; Lange, K., *"Ein Echtzeitsystem zur Geräuschreduktion auf der Basis von Subarrays"*, Nachrichtentechnik, Elektronik 5/1993, S. 231-235.

[Kro95] Kroschel, K., M. Mekhaiel, J. Crowley und F. Bérard *"Localization of Acoustic Sources using Microphone Arrays"*, Grenoble, IAR-Jahresbericht 1995.

[Kro96a] Kroschel K., Mekhaiel M., und H. Kabre *"Modeling the Lip-contour in noisy Images for Lip-reading Applications"*, IAR-Jahresbericht, Karlsruhe, 1996.

[Kro96b] Kroschel K., *"Sprachkommunikation akustisch und optisch"*, Funkschau 21/96, S.62-65, 1996.

[Kro97] Kroschel K., Mekhaiel M.S., *"A comparison of lip contour models used for automatic lipreading"*, Proc. of 9. Aachener Kolloquium "Signaltheorie" Bild- und Sprachsignale, Aachen 1997.

[Kro99a] Kroschel K., Mekhaiel M.S., *"Fusion of audio and video data by Neural Networks for robust vowel recognition"*, European Control Conference (ECC), Karlsruhe 1999.

[Kro99b] Kroschel K., Mekhaiel M.S., Berthommier, F., *"Fast lip modeling for lipreading applications using statistical color model"*, IAR-Jahresbericht, ENSAIS, Strasbourg Nov. 1999.

[Laj97] Lajoix, Nicolas, *"Recognition of German Vowels using DTW"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Juni 1997.

[Lar95] Larsen, O. V.; Radeva, P und Marti, E. , *"Calculating the Bounds on the Optimal Parameters of Elasticity for a snake"*, Computer Analysis of Images und Patterns: 6th International Conference, proceedings CAIP 1995, S. 106-113, Prague, 1995.

[Leb96a] Leber, J., Diplomarbeit, INPG, Grenoble, France, 1996.

[Leb96b] Leber, J., Crowley, J.L., Kroschel, K, *"Estimation of the position of a speaker using a microphone array"*, Universität Karlsruhe, IAR-Jahresbericht 1996.

[Lev85] Levine, M.D., *"Vision in man and machine"*, McGraw-Hill, 1985.

[Lia95] Liang, Litong, *"Schätzung der Sprecherposition durch Laufzeitschätzung bei Multimedia- Anwendungen"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Dez. 1995.

[Lin88] Linhardt, K., *"Adaptive Geräuschreduktion im Frequenzbereich bei Sprachübertragung"*, Dissertation, Universität Karlsruhe, 3. Mai 1988.

[Lue97] Luettin J., *"Visual speech and speaker recognition"*, Dissertation, Dept. of Computer Science, University of Sheffield, Mai 1997.

[Lue96] Luettin J., Thacker N.A., und Beet S.W., *"Speachreading using shape and intensity information"*, Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP) 1996.

[Mar96] Cereza Maria del Mar, *"Mouth Localization for Visual Speech Recognition"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Nov. 1996.

[Mat98] Matthews I.,Cootes T., Cox S.,Harvey R., Bangham, J. A., *"Lipreading using shape, shading and scale"*. Proceedings of Auditory-Visual Speech Processing (AVSP) 1998.

[Mac76] McGurk, H., MacDonald, *"Hearing lips and seeing voices"*, Nature, 264, S. 746-748, 1976.

[McC74] McCandless, S., *"An algorithm for automatic formant extraction using linear prediction spectra"*, IEEE ASSP-22, No.2, S. 135-141, 1974.

[Mei99] Meier, U., Stiefelhagen, R., Yang, J., Waibel, A., *"Towards unrestricted lipreading"*, Second International Conference on Multimodal Interfaces (ICMI99), Hong Kong

[Mei97] Meier U., Stiefelhagen R. und Yang J., *"Preprocessing of visual speech under real world conditions"*, Proceedings of European Tutorial & Research Workshop on Audio-Visual Speech Processing, 1997.

[Mek94] Mekhaiel, M., *"Untersuchung und Vergleich nichtlinearer Geräuschreduktionsverfahren für robuste Spracherkennung"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, 1994.

[Mov97] Movellan, J.R., Mineiro, P., *"Bayesian Robustification for Audio-visual Fusion"*, Advances in Neural Information Processing Systems 10, MIT Press, 1997.

[Mov99] Movellan, J.R., Mineiro, P., *"A diffusion network approach to visual speech recognition"*, Proceedings of Auditory-Visual Speech Processing (AVSP) Aug. 1999.

[Mur96] Murase, H., Saki, R., *"Moving object recognition in eigenspace representation: Gait analysis and lip reading"*, Pattern Recognition letters, 17(2), S. 155-162, Feb. 1996.

[Niy99] Niyogi, P., Petajan, E., Zhong, J., *"Feature based representation for audio-visual speech recognition"*, Proceedings of Auditory-Visual Speech Processing (AVSP) Aug. 1999.

[Oli97] Oliver N., Pentlund A., Bérard F. und Coutaz J., *"LAFTER: Lips und Face Real Time Tracker"*, IEEE Conference on Computer Vision und Pattern Recognition, Puerto Rico, 1997.

[Pah98] Pahor, V., Carrato, S., *"A fuzzy model of the audio-visual speech relations"*, Proc. of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT), volume 2, S. 1345-1349, Aachen, 1998.

[Pet84] Petajan, E.D., *"Automatic lipreading to enhance speech recognition"*, Proceedings of the IEEE Communication Society Global Telecommunications Conference, Atlanta, Nov. 1984.

[Pra93] Prasad, K.V., Stork D.G., Wolff G., *"Preprocessing video images for neural learning of lipreading"*, Ricoh California Research Center, Technical Report CRC-TR-93-26.

[Qui97] Quinchon, Boris, *"Lokalisierung des Gesichts mit Hilfe eines Farbmodells"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Mai 1997.

[Rab78] Rabiner, L.R., Schafer, R.W., *"Digital Processing of speech signals"*, Prentice-Hall, 1978.

[Rab89] Rabiner, R.R., *" A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition"*, Proceedings of the IEEE, vol. 77, no. 2, S. 257-285, 1989.

[Rao95] Rao R. und Mersereau R., *"State-Embedded Deformable Templates"*, ICIP 1995, Washington, D.C., 1995.

[Rei86] Reich, W., *"Adaptive Systeme zur Reduktion von Umgebungsgeräuschen bei Sprachübertragung"*, Dissertation, Universität Karlsruhe, 1986.

[Rev98] Reveret, L., Benoit, C., *"A new 3D lip model for analysis and synthesis of lip motion in speech production"*, Proceedings of Auditory-Visual Speech Processing (AVSP) 1998.

[Row95] Rowley H., Baluja S. und Kanade T., *"Human face detection in visual scenes"*, Technical Report CMU-CS-95-158, CMU, 1995.

[Rub98] Rubin, P., Vatikiotis-Bateson, E., *"Talking heads"*, Proceedings of Auditory-Visual Speech Processing (AVSP), 1998.

[Sak78] Sakoe, H., Chiba, C., *"Dynamic programming algorithm optimization for spoken word recognition"*, IEEE Trans., vol. ASSP-26, No.1, 1978.

[Schie95] Schiele B. und Waibel A., *"Gaze Tracking based on Face Color"*, IWAGFR 95-International Workshop on Face und Gesture Recognition, Zurich, 1995.

[Schmi89] Schmidbauer, O., *"Ein System zur Lauterkennung in fliessender Sprache auf der Basis artikulatorischer Merkmale"*, Dissertation an der Fakultät für Elektrotechnik und Informationstechnik der technischen Universität München, 1989.

[Sei96] Seifried, Andreas, *"Merkmalsextraktion in Bildsequenzen der Lippen zur Vokallokalisierung in Einzelwörtern"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Mai 1996.

[Sej90] Sejnowski, T.j., Yuhas, B.P., Goldstein, M.H. und Jenkins, R.E. , *" Combining Visual und Acoustic Speech Signals with a Neural Network Improves Intelligibility"*, in D. S. Touretzky (Ed.),Advances in Neural Information Processing Systems, volume 2, S. 232-239,Morgan Kaufman,1990.

[Sen99] Senior, A., Neti, C.V., Maison, B., *"On the use of visual information for improving audio-based speaker recognition"*, Proceedings of Auditory-Visual Speech Processing (AVSP) Aug. 1999.

[Sil93] Silsbee, P.L., *"Computer Lipreading for Improved Accuracy in Automatic Speech Recognition"*, Ph.D. Dissertation, University of Texas at Austin, Mai 1993.

[Sko98] Skobowsky, Tilman, *"Robuste Vokalerkennung mit Hilfe von Audio- und Videodaten"*, Diplomarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Aug. 1998.

[Sto96] Stork, D.G., Lu, H.-L., *"Speechreading by Boltzmann zippers"*, in *Machines that Learn*, Snowbird, UT, 1996.

[Sun94] Sung K. und Poggio T., *"Example-based learning for view-based human face detection"*, Technical Report A.I. Memo 1521, CBLC Paper 112, MIT, 1994.

[Swa91] Swain M. und Ballard D., *"Color Indexing"*, International Journal of Computer Vision, volume 7, No. 1, 1991.

[Tal99] Talle, B., Wichert, A., *"Audio-visual sensorfusion with neural architectures"*, Proceedings of Auditory-Visual Speech Processing (AVSP) Aug. 1999.

[Tam98] Tamura, M., Masuko, T., Kobayashi, T., Tokuda, K., *"Visual speech synthesis based on parameter generation from HMM: speech-driven and text-and-speech-driven approaches"*, Proceedings of Auditory-Visual Speech Processing (AVSP) 1998.

[Taz99] TAZ Nr. 5979 vom 1.11.1999 Seite 7 Wirtschaft und Umwelt.

[Tro95] Trompf, M., *"Künstliche neuronale Netzwerke zur adaptiven Geräuschreduktion für robuste Spracherkennung"*, Dissertation, Universität Karlsruhe, 1995.

[Vog96] Vogt, M., *"Fast Matching of a Dynamik Lip Model to Color Video Sequences under Regular Illumination Conditions"*, In: D.G. Stork, M.E. Hennecke (eds.): *"Speechreading by Humans and Machines, NATO ASI Series F, Vol. 150, Springer Verlag"*, S. 399-408, 1996.

[Vog97] Vogt, M., *"Interpreted Multi-State Lip Models for Audio-Visual Speech Recognition"*, In: C. Benoit, R. Campbell (eds.): *"Proceedings of the ESCA Workshop on Audio-Visual Speech Processing, Cognitive and Computational Approaches"*, ESCA-ESCOP, Rhodes (Greece), 1997.

[Wag96] Wagner, Sandrine, *"Schätzung der Sprecherposition durch Laufzeitschätzung"*, Studienarbeit, Institut für Nachrichtentechnik, Universität Karlsruhe, Juni 1996.

[Wil92] Williams D. und Shah M., *"A fast algorithm for active contour und curvature estimation"*, CVGIP: Image Understunding, S. 14-26, 1991.

[Yan94] Yang G. und Huang T., *"Human Face detection in a complex background"*, Pattern Recognition, 27(1), S. 53-63, 1994.

[Yan95] Yang J. und Waibel A., *"Tracking Human Faces in Real-Time"*, CMU Technical Report CMU-CS-95-210, 1995.

[Yan98] Yang, J., Stiefelhagen, R., Meier, U., Waibel, A., *"Real-time face and facial feature tracking and applications"*, Proceedings of Auditory-Visual Speech Processing (AVSP) 1998.

[Yuh90] Yuhas, B.P., Goldstein, M.H., Sejnowski, T.J., Jenkins, R.E., *"Neural network models of sensory integration for improved vowel recognition"*, Proceedings of the IEEE, 78(10) S. 1658-1668, 1990.

[Yul92] Yuille A., Hallinan P. und Cohen D. *"Feature extraction from faces using deformable templates"*, Journal of Computer Vision. S. 99-111, 1992.

Lebenslauf

Persönliche Daten

Name	Moheb Mekhaiel
Geburtsdatum	28.05.1968
Geburtsort	Giza, Ägypten
Staatsangehörigkeit	ägyptisch

Schulausbildung

1973-1977	Grundschule in Giza
1977-1985	Deutsche Evangelische Oberschule, Giza

Studium und Berufsweg

1985-1990	Abschluss B.Sc. in “Electrical Engineering“, Kairoer Universität
1990	Wissenschaftlicher Assistent, FRCU, Kairoer Universität
1990-1991	Planungs- und Montage-Ingenieur für Telefonvermittlungsanlagen, Philips, Giza
1991-1994	Diplom der Elektrotechnik an der Universität Karlsruhe (Diplomarbeit: Alcatel/SEL, Stuttgart)
1995-2000	Wissenschaftlicher Angestellter am Institut für Automation und Robotik der Universität Karlsruhe