FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Interaktive Analyse audiovisueller Medien

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Medieninformatik**

eingereicht von

**Alexander Fried**
Matrikelnummer 0625941

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer: Dr. Horst Eidenberger

Wien, 10.10.2012

(Unterschrift Verfasser) (Unterschrift Betreuer)

# Erklärung zur Verfassung der Arbeit

Alexander Fried
Mariahilfer Straße 56/18
1070 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, am 10.10.2012

## Abstract

Media understanding is the domain of creating human-like perception of media objects in computers. This thesis addresses three main topics within this area: similarity detection of human faces, object recognition (body part detection in images, in particular) and fast speech recognition. In order to illustrate the practical purpose and the benefit of media understanding for the target group (undergraduate students in computer science) exemplary applications are implemented and discussed. The first part of this thesis elaborates on the theory behind the methods applied. Based on existing systems the most suitable features and classifiers for a given problem statement are investigated. The second section of the thesis deals with the practical implementation. In the first application the measurement of the similarity of prominent faces is investigated. Template matching is used as a method for calculating similarity. In the second application different body parts, recorded with a webcam, are detected and classified. The software uses a local feature extraction method for this task. For the accurate classification, a probabilistic method is applied (among others). The third application facilitates verbal interaction with the computer. The user is required to simulate the sound of a given species. Then, the system confirms whether the user's interpretation matches the one of the animal. In the same way the sound can be presented to users, asking them to reply with the correct animal name. Here, the software uses spectral audio features, which are recognized by dynamic time warping. All three applications together prove that media understanding can be implemented successfully today.

## Kurzfassung

Medienverstehen ist der Versuch, Computern menschenartiges Erkennen von Medien unterschiedlicher Art zu ermöglichen. Diese Arbeit beschäftigt sich mit drei Themenbereichen aus diesem Gebiet: der Ähnlichkeitsberechnung von Gesichtern, der Objekterkennung (im Speziellen der Körperteilerkennung) und der Spracherkennung. Diese drei Themengebiete wurden in Applikationen praktisch umgesetzt. Die Absicht dieser Demoanwendungen besteht darin, die Praxisbezogenheit des Medienverstehens zu verdeutlichen. Die für die Umsetzung erforderlichen Methoden werden im theoretischen Teil dieser Diplomarbeit beschrieben. In diesem Abschnitt wird argumentiert, warum diese Methoden für die Implementierung geeignet sind, wobei Bezug auf verwandte Arbeiten genommen wird.
Die erste Demoanwendung verdeutlicht Ähnlichkeitsmessung zwischen Gesichtern. Als Methode der Ähnlichkeitsberechnung wird Template Matching verwendet. In der zweiten Anwendung werden die mit der Webcam aufgenommenen Körperteile erkannt. Hier wird eine lokale Featureextraktionsmethode verwendet. Für die Klassifizierung wird u. a. eine probabilistische Methode herangezogen. Die dritte Applikation ermöglicht sprachliche Interaktion mit dem Computer. Der Benutzer erhält die Aufforderung, den Laut eines vorgegebenen Tieres nachzumachen, das System soll überprüfen, ob es sich um das richtige Tier handelt. Genauso gut kann jedoch auch eine Frage gestellt werden, auf die der Benutzer mit dem Tiernamen antworten muss. In der Implementierung werden spektrale Features verwendet, die anhand eines geeigneten Klassifikators analysiert werden. Alle drei Applikationen beweisen, dass Medienverstehen erfolgreich implementiert werden kann.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

**DCT** diskrete Cosinustransformation

**DSIFT** Dense SIFT

**DTW** Dynamic Time Warping

**ETSI** European Telecommunications Standards Institute

**FFT** schnelle Fouriertransformation

**GUIDE** GUI design environment

**HMM** Hidden Markov Model

**JPEG** Joint Photographic Experts Group

**kNN** k-Nearest Neighbor

**LPC** Linear Predictive Coding

**MATLAB** Matrix Laboratory

**MEX** MATLAB Executables

**MFCC** Mel Frequency Cepstral Coefficients

**MP3** MPEG Audio Layer III

**PCA-SIFT** Principal Components Analysis SIFT

**PHOW** Pyramid Histogram of Visual Words

**RGB** Rot-Grün-Blau

**SIFT** Scale Invariant Feature Transform

**SMQT** Successive Mean Quantization Transform

**SNoW** Sparse Network of Winnows

**SURF** Speeded Up Robust Features

**SVM** Support Vector Machine

**VAD** Voice Activity Detection

KAPITEL 1

# Einführung

Zunächst wird in diesem Kapitel in Abschnitt 1.1 die Motivation dieser Arbeit erläutert. Die Zielsetzung und Vorgehensweise wird in Abschnitt 1.2 beschrieben. Abschließend wird in diesem Kapitel in Abschnitt 1.3 der Aufbau dieser Diplomarbeit vorgestellt.

## 1.1 Motivation

Die Motivation hinter technologischem Fortschritt war seit jeher eine Arbeitserleichterung für den Menschen und die Steigerung der Produktivität. Als bekanntestes Beispiel sei die Industrielle Revolution erwähnt, bei der die zuvor von der Agrarwirtschaft lebenden Nationen zu Industriestaaten wurden und die gesellschaftliche Struktur grundlegend verändert wurde. Auf ähnliche Weise revolutionierte der Computer im 20. Jahrhundert das Leben der Menschen. Heute ist er aus unserem Alltag nicht mehr wegzudenken. Durch die rasche Leistungssteigerung von Computern konnten immer komplexere Rechenaufgaben bewältigt werden, was durch die ebenfalls immer größer werdenden Speicherkapazitäten unterstützt wird. Die automatisierte Verarbeitung von Medien ist ein Beispiel der durch die sogenannte *digitale Revolution* verursachte Arbeitserleichterung. Der Begriff *Medienverstehen* fasst diese auf Medien spezialisierten Tätigkeiten zusammen, wobei anzumerken ist, dass diese Bezeichnung in der Wissenschaft selten verwendet wird. Meist wird oft nur auf einem Teilgebiet davon wie z. B. Bildverstehen geforscht, Wissenschaftler arbeiten selten mit echten Multimediainhalten [Eid11]. Medienverstehen bietet eine Vielzahl von Anwendungsmöglichkeiten, die unser tägliches Leben erleichtern. In dieser Arbeit soll auf den Praxisbezug aufmerksam gemacht werden und ein kleiner Einblick in dieses umfangreiche Themengebiet geboten werden. Insbesondere wird auf die Spracherkennung und auf das Bildverstehen anhand von praktischen Implementierungen näher eingegangen.
Die automatisierte Spracherkennung ist ein wichtiges Teilgebiet des Medienverstehens und ist vielseitig einsetzbar. Die gesprochene Sprache ist das primäre Interaktionsmedium der Menschen, insofern ist es ein vorrangiges Ziel, dass auch mit Computern in dieser Form interagiert werden kann. Die Einsatzbereiche der Spracherkennung sind sehr vielfältig. Einerseits erleichtern sie die Interaktionsmöglichkeiten mit technischen Geräten, sodass diese auch in Situationen

bedient werden können, in denen eine manuelle Bedienung nicht möglich ist. Andererseits ist die Sprachsteuerung technischer Geräte auch in Bezug auf Barrierefreiheit sehr wichtig. Spracherkennungssysteme stellen auch eine wesentliche Arbeitserleichterung für Personen ohne Technikaffinität dar und können die Bedienung wesentlich vereinfachen. Im Idealfall kann der Benutzer mit dem technischen Gerät wie mit einem Menschen interagieren. Eng verwandt mit der Spracherkennung ist auch die Sprechererkennung, welche beispielsweise als Authentifizierungsmöglichkeit einsetzbar ist.

Bei der Spracherkennung existieren potenzielle Problemfelder, die die Qualität der Identifikation des Inhalts stark beeinflussen können, die durch folgende Eigenschaften der gesprochenen Sprache verursacht werden: die Kontinuität, Komplexität und Variabilität. Bei Schriftsprache ist die Abgrenzung zwischen den Silben und Wörtern eindeutig durch die niedergeschriebene Form definiert, bei gesprochener Sprache hingegen verschmelzen die Laute zu einem kontinuierlichen Medium [PK08]. Im Deutschen existieren bis zu 500.000 Wörter, welche unter Berücksichtigung der grammatikalischen Regeln beliebig zu Sätzen kombiniert werden können, wodurch die Komplexität steigt. Ein zuverlässiges Spracherkennungssystem soll diese Sätze verarbeiten und in einem weiteren Schritt mitunter sogar semantisch verstehen können. Die Variabilität der Sprache macht sich durch die unterschiedliche Aussprache verschiedener Sprecher bemerkbar, die durch die individueller Anatomie der menschlichen Sprachproduktionsorgane bedingt ist. Jeder Sprecher hat eine eigene Sprechweise, die sich etwa durch den Ausdruck und Wortschatz auswirkt. Diese Individualität unter dem Begriff *Idiolekt* zusammengefasst. Auch unterschiedliche Dialekte sind eine Herausforderung für ein zuverlässige Spracherkennung.

Besonderes Interesse in dieser Diplomarbeit gilt außerdem dem Bildverstehen, dessen Ziel darin besteht, die in Bildern enthaltenen Informationen zu verstehen und zu verarbeiten. Der Sehsinn ist mit Abstand der wichtigste Sinn des Menschen. Mit ihm wird der Großteil der Informationen aus unserer Umgebung wahrgenommen. Insofern besteht hohes Interesse daran, die visuelle Wahrnehmung nachzubilden. Der praktische Nutzen dieses Teilgebiets ist sehr vielseitig und reicht von der Medizin über die Handschrifterkennung bis zu Anwendungsgebieten wie Lächelerkennung bei Digitalkameras. Die automatisierte Schrifterkennung wird z. B. bei der Post angewandt, um anhand der Postleitzahl die Briefe für die Zustellung zu sortieren. Ein zukunftsträchtiges Einsatzgebiet ist die biometrische Authentifizierung mittels Gesichtern. Große Herausforderungen bei solchen Systemen sind, dass nur reale Personen akzeptiert werden und die Authentifizierung anhand eines Fotos des Benutzers unterbunden wird, sodass nicht unrechtmäßig Zugriff verschafft wird.

Zusammenfassend kann gesagt werden, dass viele Verwendungszwecke des Medienverstehens sowohl in komplexen Anwendungsgebieten als auch in unserem Alltag existieren. Vielen Personen ist allerdings die Komplexität solcher Systeme mit dem damit verbundenen Aufwand gar nicht bewusst oder können sich unter der sich damit beschäftigenden Studienrichtung Medieninformatik wenig vorstellen. Aus diesem Grund soll diese Arbeit den starken Praxisbezug des Medienverstehens unterstreichen und die Theorie auf interessante Art umsetzen, sodass Interesse an diesem Studium geweckt und der Praxisbezug der Studienrichtung Medieninformatik verdeutlicht wird.

## 1.2 Zielsetzung und Vorgehensweise

In diesem Rahmen werden drei Demoanwendungen implementiert, die die Verwendung des Medienverstehens in der Praxis verdeutlichen sollen. Des Weiteren soll in dieser Diplomarbeit ein theoretischer Überblick über die dafür erforderlichen Verfahren geboten werden.

Die erste zu implementierende Anwendung stellt Benutzern Gesichtsdetektion und Ähnlichkeitsberechnung zwischen Gesichtern vor. Im Konkreten soll aus einem Pool von Gesichtern von prominenten Personen der dem Anwender am ähnlichsten sehende Prominente ermittelt werden. Die Gesichtserkennung ist eine Fähigkeit des Menschen, die sich in den frühen Jahren der Kindheit ausprägt und betrifft viele Bereiche unseres Lebens. Ohne diese Fähigkeit wäre zwischenmenschliche Interaktion in unserer bekannten Form nicht möglich. Des Weiteren spielte die Gesichtserkennung gemeinsam mit der Emotionserkennung eine wesentliche Rolle in der menschlichen Evolution [BP93].

In der Applikation wird zunächst der Benutzer dazu aufgefordert ein Foto seines Gesichts aufzunehmen. Dieses Bild wird nun mit den Gesichtern der Prominenten verglichen. Das ähnlichste davon wird ausgegeben. Zur Ähnlichkeitsberechnung wird Template Matching verwendet, wobei das Gesicht des Benutzers als Template verwendet wird. Die Ähnlichkeit soll auch für den Benutzer nachvollziehbar sein, indem ihm der Vergleich der Gesichter ausgegeben wird. Herausforderung in diesem Demo ist es, dass tatsächlich nur Bereiche verwendet werden, in denen auch das Gesicht des Prominenten vorkommt und somit keine falschen Übereinstimmungen zwischen dem Template und den Prominentenfotos auftreten.

Bei der zweiten Demoanwendung soll mit dem gesamten Körper ein an *Memory* angelehntes Spiel gespielt werden. Der Benutzer bekommt die Vorgabe, ein bestimmtes Körperteil in die Kamera zu halten und muss so schnell wie möglich reagieren. Bei einer Übereinstimmung erhält der Benutzer Punkte, die von der Schnelligkeit seiner Reaktion abhängig sind. Die größte Herausforderung bei der Implementierung ist hierbei, ähnlich aussehende Körperteile wie z. B. Ellenbogen und Knie auseinander zu halten. Wichtig ist außerdem, dass auch bei unterschiedlichen Lichtverhältnissen die Erkennung zuverlässig funktioniert. Die Orientierung der Körperteile soll so wenig wie möglich Einfluss auf die Erkennung haben. Diese lässt sich allerdings durch die Tatsache, dass die Webcam meist im Computer integriert ist bzw. oberhalb des Monitors platziert ist, erheblich einschränken, da die Perspektive und der Abstand zu verschiedenen Benutzern ungefähr gleich ist. Die Körperteilerkennung ist eine Spezialform der Objekterkennung, weshalb deren Methoden verwendet werden können. Objekterkennungsalgorithmen bestehen in der Regel aus einem Featureextraktionsverfahren, das die charakteristischen Merkmale eines Bildes ermittelt, und einem Klassifikationsverfahren, welches das Foto zu einer Klasse zuordnet. Für die Klassifizierung ist ein dementsprechendes Trainingsset erforderlich.

Ausgehend von einem von Vedaldi [VF08] implementierten Demoprogramm, das Bilder des Caltech-101-Datensets[1] verwendet, soll untersucht werden, mit welchem Klassifikationsalgorithmus Bilder anhand ihrer SIFT-Features klassifiziert werden können. Neben der bereits im Demoprogramm implementierten Support Vector Machine (SVM) sollen insbesondere der kNN-

[1] Das Datenset ist unter `http://www.vision.caltech.edu/Image_Datasets/Caltech101/` erhältlich (zuletzt abgerufen am 22.09.2012)

und der Naive-Bayes-Klassifikator betrachtet werden, da diese drei Klassifikationsalgorithmen oftmals mit SIFT-Features kombiniert werden.
Die dritte Demoanwendung ist ein auf Spracherkennung basierendes Spiel, bei dem der Benutzer mit dem Computer verbal kommunizieren soll. Der Benutzer bekommt vom Computer Tierlaute vorgegeben und muss mit dem entsprechenden Tiergattungsbegriff antworten. Genauso gut kann aber auch der Tiergattungsbegriff vom Computer vorgegeben werden, worauf der Benutzer mit dem entsprechenden Tierlaut antworten soll. Der Punktestand ergibt sich anhand der Anzahl der korrekten Antworten und der benötigten Zeit für die Antwort. Auch dieses Programm besteht aus einer Featureextraktionsmethode und der anschließenden Klassifizierung, für die ein Trainingsset erstellt werden muss. Diese Anwendung verlangt eine zuverlässige Detektion von unterschiedlichen Sprechern.
Zunächst wurde im Rahmen dieser Diplomarbeit der theoretische Hintergrund durch Literaturrecherche ermittelt. Dabei wurde der Stand der Technik recherchiert, um geeignete Methoden für die Implementierung zu finden. Anschließend wurde mit der Implementierung der Prototypen begonnen. Die Demos wurden mit diesem Wissen in MATLAB umgesetzt. MATLAB ermöglicht es laut Hersteller[2], rechenintensive Probleme aus der Mathematik einfacher und schneller zu lösen, als das in herkömmlichen Programmiersprachen möglich ist (wie z. B. C++ oder Java).

## 1.3 Struktur der Arbeit

Diese Arbeit geht zunächst in Kapitel 2 auf die zugrunde liegende Theorie ein, die für die praktische Umsetzung der Demoanwendungen erforderlich ist. Das Kapitel der Literaturrecherche ist unterteilt in die Featureextraktionsmethoden und in die Klassifikation der Medienbeschreibungen. Der Abschnitt, der die Featureextraktionsmethoden behandelt, ist wiederum unterteilt in audiospezifische und bildspezifische Verfahren. Bei den Features in Audiosignalen wird zunächst auf die Grundlagen der auditiven Wahrnehmung des Menschen und die Spracherzeugung des Menschen näher eingegangen. Darauf aufbauend werden die Mel Frequency Cepstral Coefficients (MFCC) vorgestellt. Bei diesen handelt es sich um Features, die sehr häufig bei der Spracherkennung verwendet werden. Des Weiteren wird ein Überblick über die Methoden der Featureextraktion in Bildern geschaffen. Kapitel 3 bietet einen Überblick über den praxisbezogenen Teil dieser Arbeit. Hier wird unter anderem auch argumentiert, weshalb MATLAB für die Umsetzung verwendet wird, und die Ergebnisse der Klassifikationsverfahren präsentiert. Insbesondere bei der Verwendung unterschiedlicher Klassifikatoren werden deren Vor- und Nachteile beschrieben. Auf die Handhabung der Bedienung wird in Kapitel 4 eingegangen. Der Fokus des Programms liegt auf Benutzerfreundlichkeit und intuitiver Benutzung. Abschließend werden die Ergebnisse in Kapitel 5 zusammengefasst, bei der die Performance der einzelnen Algorithmen beurteilt wird. Hier wird auf die in Kapitel 3 vorgestellten Gütemaße Bezug genommen. Da bei der Anwendung der Körperteilerkennung die Möglichkeit besteht, unterschiedliche Algorithmen zur Klassifikation zu verwenden, werden deren Ergebnisse untereinander verglichen, um den am besten abschneidenden Algorithmus zu ermitteln. Abschließend werden in Kapitel 6 die wichtigsten Erkenntnisse der Arbeit nochmals zusammengefasst und ein Ausblick auf zukünftige Verfahren geboten.

[2] `http://www.mathworks.de/products/matlab/index.html` (zuletzt abgerufen am 07.10.2012)

KAPITEL 2

# Literaturüberblick

Dieses Kapitel bietet einen Überblick über die Theorie, die für die Implementierung der drei Demoanwendungen erforderlich ist. Es unterteilt sich in Featureextraktionsmethoden und in die Klassifikation von Medienbeschreibungen. Zunächst müssen geeignete Merkmale gefunden werden, anschließend kann anhand dieser eine Klassifikation durchgeführt werden. Features sind in der Regel abhängig vom Medium und werden in Featurevektoren zusammengefasst. Klassifikatoren sind unabhängig vom Medium einsetzbar, da diese nur anhand der Features eine Entscheidung über die Klassenzugehörigkeit treffen und nicht über die dahinter liegende Information Bescheid wissen müssen. Somit werden die Featureextraktionsmethoden für Audiodateien und Bildern gesondert betrachtet. Diese Unterscheidung entfällt bei den Klassifikationsmethoden. In beiden Abschnitten wird zunächst jeweils auf die Grundlagen, aber auch auf konkrete Methoden eingegangen, die im Rahmen dieser Arbeit implementiert wurden. Des Weiteren wird in Abschnitt 2.3 anhand von Beispielen aus der Praxis argumentiert, warum diese Methoden für die Lösung der Problemstellungen geeignet sind.

## 2.1 Featureextraktionsmethoden

Für das Verständnis der Featureextraktionsmethoden werden die grundlegenden Eigenschaften von Features beschrieben und erklärt, wofür diese benötigt werden. Darauf aufbauend, wird auf die Features von Audiosignalen und Bildern näher eingegangen und konkrete Methoden vorgestellt, die im Kontext dieser Arbeit von Relevanz sind.

### 2.1.1 Was ist ein Feature?

Die gängigen Repräsentationsformen von Bildern oder Audiosignalen enthalten zu viel irrelevante Information, sodass eine rasche Weiterverarbeitung in dieser Form nicht möglich ist. So werden Graustufenbilder als eine $m \times n$ große Matrix mit verschiedenen Intensitätswerten pro Pixelkoordinate dargestellt, RGB-Farbbilder enthalten die dreifache Datenmenge, da pro Farbkanal eine Matrix gespeichert wird. Zeitabhängige Signale wie z. B. Audioaufnahmen werden

durch die Digitalisierung als Vektor repräsentiert, der an den Abtastpunkten die entsprechenden Amplitudenwerte des analogen Signals enthält. Für eine automatisierte Verarbeitung von Daten ist es unerlässlich, aus dieser Menge an Information diejenigen Eigenschaften zu extrahieren, die das Abgebildete möglichst gut beschreiben. Diese charakteristischen Merkmale werden Features genannt. Der Vorgang, der dafür erforderlich ist, ist die Featureextraktion. Bei der Featureextraktion ist es sowohl wichtig, die richtigen Merkmale zu verwenden, als auch die Anzahl der relevanten Merkmale gering zu halten, sodass die gewünschte Aufgabe (beispielsweise eine automatisierte Klassifizierung, siehe Abschnitt 2.2) rasch und zuverlässig bewältigt werden kann.
In den nun folgenden Abschnitten wird auf die beiden Featuretypen der in dieser Arbeit verwendeten Medien eingegangen: Audiosignale und Bilder. Insbesondere werden die Methoden vorgestellt, die für die Demoanwendungen von Relevanz sind.

### 2.1.2 Features in Audiosignalen

Um zu verstehen, welche Eigenschaften in Sprachaufnahmen relevante Informationen enthalten, ist die grundlegende Kenntnis der Spracherzeugung beim Menschen erforderlich. Außerdem wird der Aufbau des menschlichen Ohrs und die auditive Wahrnehmung erläutert, insbesondere auch die Psychoakustik. Letztere liefert für manche Featureextraktionsmethoden interessante und wichtige Rückschlüsse. Abschließend wird eine konkrete Methode der Featureextraktion vorgestellt, die MFCC.

#### 2.1.2.1 Spracherzeugung beim Menschen

Das vom Menschen erzeugte Sprachsignal besteht aus einem höherfrequenten Trägersignal und einer Vielzahl von niederfrequenten Signalen, die die eigentliche Information des Gesprochenen beinhalten. Das Trägersignal selbst enthält keine Information, sondern dient nur der Übertragung.
Die Erzeugung des akustischen Sprachsignals beim Menschen erfolgt in folgenden drei Bereichen: Der Abschnitt unterhalb des Kehlkopfes, in dem der Luftstrom erzeugt wird (Luftröhre, Bronchien und Lunge), dem Kehlkopf, der die Stimmlippen enthält, und dem Vokaltrakt, der den Mund- und Rachenraum umfasst [Pom97]. Während die Stimmlippen die Grundfrequenz des Trägers bestimmen (siehe Abschnitt *Schallproduktion*), kann der Mensch mit dem Vokaltrakt gewisse Frequenzen filtern. Dieser Vorgang ist auch unter dem Begriff *Klangformung* bekannt. Aufgrund dieser Art der Erzeugung spricht man auch von einem Quelle-Filter-Modell bei der Sprachproduktion [PM09]. Der Einfachheit halber wird in dieser Arbeit nur auf die Spracherzeugung von stimmhaften Lauten eingegangen, den Vokalen.

**Schallproduktion** Für die Erzeugung von Lauten wird Luft als Träger benötigt. Sobald zum Sprechen begonnen wird, bringt die aus der Lunge ausgestoßene Luft die Stimmlippen zur Bewegung [PK08]. In Abbildung 2.1 sieht man das Spektrum des Trägersignals und somit die Frequenzen, aus denen sich dieses zusammensetzt. Bei Audiosignalen bestimmt die Frequenz eines Signals die Tonhöhe – je höher die Frequenz, desto höher ist diese. Spitzen im Spektrum

deuten auf dominierende Frequenzen hin. Bei akustischen Signalen sind diese Spitzen jene Frequenzen, die das Signal charakterisieren. Die erzeugte Trägerfrequenz ist sprecherabhängig und bestimmt die Tonhöhe.

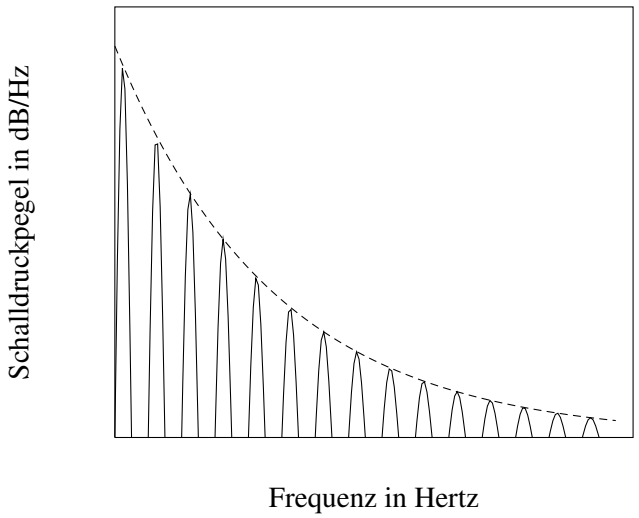



Abbildung 2.1: Das Spektrum der Frequenzen des Kehlkopfes (Trägersignal).

**Klangformung** Je nach Stellung des Mund- und Rachenraums werden unterschiedliche Laute erzeugt. Die dominierenden Frequenzen, die einen Vokal charakterisieren, werden Formanten genannt. Sie sind alters- und geschlechtsunabhängig, da der Rachenraum bei allen Menschen gleich aufgebaut ist. Im Frequenzspektrum sind sie als Spitzenwerte ersichtlich (siehe Abbildung 2.4). Laut einer Studie von Kremer [Kre04] sind jedoch auch die niederfrequenten Formanten teilweise sprecherabhängig. Sie werden aufsteigend durchnummeriert, je niedriger der Index, desto niedriger ist dessen Frequenz [BM76]. Zur Identifikation eines Vokals sind lediglich die ersten beiden Formanten notwendig (siehe Abbildung 2.2). Der erste liegt im Bereich von 200 bis 1000 Hz und wird durch den hinteren Rachenraum bestimmt, der zweite zwischen 800 und 3200 Hz und wird durch den vorderen Rachenraum geprägt. Durch die Bewegung der Zunge können die Frequenzen beeinflusst werden [Web04]. Der dritte und der vierte Formant charakterisieren eher sprecherabhängige Eigenschaften des gesprochenen Vokals. Laut anderen Quellen macht es jedoch durchaus auch Sinn, den dritten Formanten für die Spracherkennung miteinzubeziehen. Das wird vor allem dann gemacht, wenn der angrenzende Konsonant ebenfalls bestimmt werden soll [Mac96].
Abbildung 2.3 zeigt das Spektrum des Vokaltrakts und Abbildung 2.4 das des ausgesandten Signals. Die Frequenzen, die im Vokaltrakt entstehen, sind niederfrequenter als das Trägersignal und sind als Spitzenwerte im Spektrum ersichtlich. Ziel ist es, alle irrelevanten Informationen der Spektralkurve (auch bekannt als *Spectral Details*) zu eliminieren und nur die Formanten zu behalten. Grafisch kann man das durch die Verbindung der Spitzenwerte veranschaulichen (siehe Abbildung 2.4). Diese Hüllkurve wird auch *Spectral Envelope* genannt.

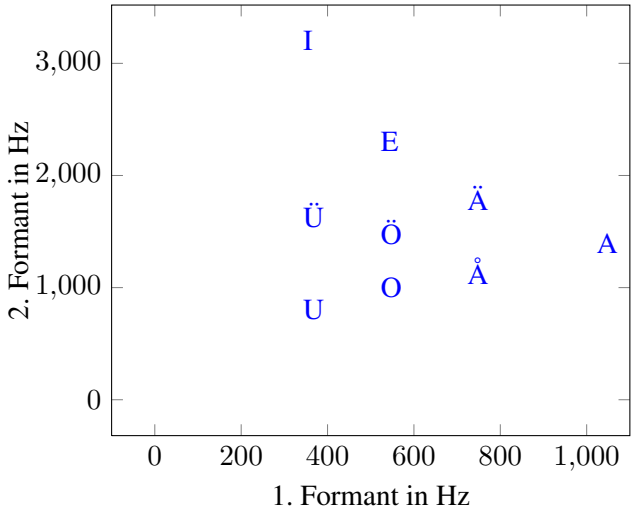



Abbildung 2.2: Der erste und zweite Formant der deutschen Vokale.

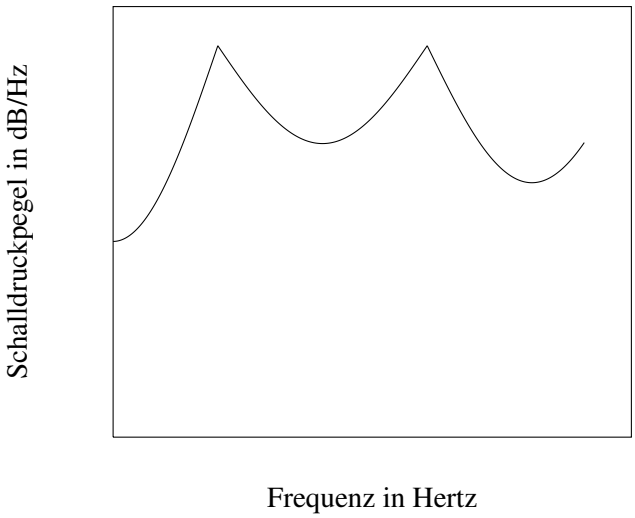



Abbildung 2.3: Die im Vokaltrakt erzeugten Frequenzen, die die Information enthalten.

Neben der Sprachproduktion ist auch die auditive Wahrnehmung des Menschen ein wichtiger Faktor für die Extraktion der relevanten Merkmale in einem Sprachsignal, worauf nun näher eingegangen wird.

#### 2.1.2.2 Die Funktionsweise des menschlichen Ohrs

Die Druckschwankungen in der Luft, die vom Menschen auditiv wahrgenommen werden, nennt man *Schall*. Für dessen Wahrnehmung ist das Ohr verantwortlich, welches sich in folgende drei

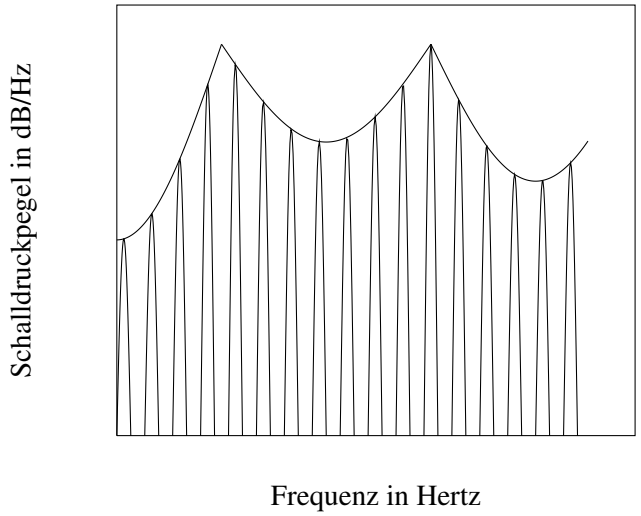



Abbildung 2.4: Die Hüllkurve der Funktion wird durch den Filter verursacht, die Grundfrequenz ist das Anregungssignal. Ziel ist es, die Informationen der Spitzen, der Formanten, zu extrahieren.

anatomischen Bereiche unterteilt: Dem Außen-, Mittel- und Innenohr. In diesen Abschnitten erfolgt die Umwandlung des Schalls in elektrische Signale, die vom Gehirn weiterverarbeitet werden.

**Außenohr** Das Außenohr besteht aus der Ohrmuschel und dem äußeren Gehörgang. Der auftreffende Schall wird durch die Ohrmuschel gesammelt und durch den Gehörgang zum Trommelfell geleitet. Im Gehörgang erfolgt eine Verstärkung der Frequenzen zwischen 2 und 4 kHz [Web04]. Die Ohrmuschel wirkt wie ein Schalltrichter. Durch seine spezielle Struktur mit seinen Erhebungen und Vertiefungen kann bestimmt werden, aus welcher Richtung der Schall kommt. Für die auditive Wahrnehmung hat die Ohrmuschel einen eher geringen Anteil, dennoch verbessert sie die Wahrnehmung vor allem von hohen Frequenzen [Web04].

**Mittelohr** Die im Trommelfell entstehenden Schwingungen werden im Mittelohr über die drei Gehörknöchelchen zum ovalen Fenster geleitet. Die drei Gehörknöchelchen heißen aufgrund ihrer Form *Hammer*, *Amboss* und *Steigbügel*. Ihre Aufgabe besteht darin, die Schwingungen des Trommelfells bei der Weiterleitung an das Innenohr zu verstärken. Diese Verstärkung ist erforderlich, da die Schwingung von einem luftgefüllten auf einen mit einer Flüssigkeit gefüllten Raum übergeben werden muss. Bei einem direkten Auftreffen des Schalls auf das Innenohr würden 99 Prozent der Energie verloren gehen.

**Innenohr** Im Innenohr befindet sich das Vestibularorgan und die Cochlea (auch *Hörschnecke* genannt). Das Vestibularorgan ist für den Gleichgewichtssinn verantwortlich, die Hörschnecke für die auditive Wahrnehmung. Letztere besteht aus drei übereinander liegenden Gängen, die mit

einer Flüssigkeit gefüllt sind. Die Flüssigkeit wird durch die Schwingungen des Steigbügelknöchelchens im Mittelohr in wellenförmige Schwingungen versetzt. In der Cochlea befindet sich das Cortische Organ, das die Wellen in ein akustisches Signal transformiert. Das Cortische Organ besteht aus Haarzellen, die auf die Schwingungen der Flüssigkeit reagieren. Am Beginn der Cochlea befinden sich kurze Haarzellen, die sehr empfindlich auf hohe Frequenzen reagieren. Je weiter innerhalb sich die Haarzellen befinden, desto länger werden diese. Die tiefsten Frequenzen werden demnach ganz innen in der Cochlea aufgefangen (Abbildung 2.5). Die Haarzellen wandeln die mechanischen Schwingungen in elektrische Signale um, diese stimulieren den Hörnerv. Die Erregung einer Haarzelle ist abhängig von der Dauer der auftretenden Schwingungen. Dauert die Anregung eine gewisse Zeit an, tritt Adaption auf: Die Haarzellen sind dann auf Anregung nicht mehr empfindlich.

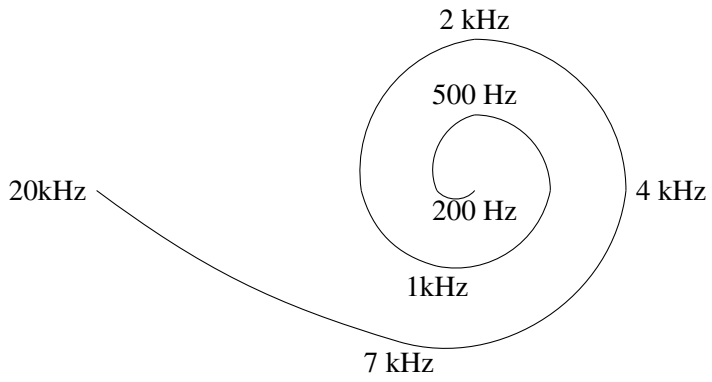



Abbildung 2.5: Schematischer Aufbau der Cochlea mit den in den jeweiligen Bereichen empfundenen Frequenzen.

Die Frequenzen, die wahrgenommen werden, liegen ca. zwischen 200 Hz bis zu 20 kHz. Im Laufe des Alters sinkt jedoch die Wahrnehmungsgrenze: Hohe Töne werden nur noch weniger gut gehört. Von den hörbaren Frequenzen werden allerdings nicht alle bei gleichem Lautstärkepegel gleich gut wahrgenommen. Diese und andere spezifischen Eigenschaften werden unter dem Begriff *Psychoakustik* zusammengefasst.

#### 2.1.2.3 Psychoakustik

Die Psychoakustik ist ein Teilgebiet der Psychophysik und beschreibt, wie das akustische Schallereignis vom menschlichen Ohr wahrgenommen wird. Die wichtigsten Teilbereiche der Psychoakustik sind die Lautheit, Tonheit, Schärfe, Rauhigkeit und Schwankungsstärke.

**Lautheit** Die Lautheit bildet das Lautstärkeempfinden des Menschen ab. Sie verdoppelt sich, wenn der auftreffende Schall als doppelt so laut empfunden wird. Die Lautheit wird in *Sone* gemessen, dabei entspricht 1 Sone einem Lautstärkepegel von 40 Phon. Der Zusammenhang zwischen den beiden Größen ist in Abbildung 2.6 dargestellt.

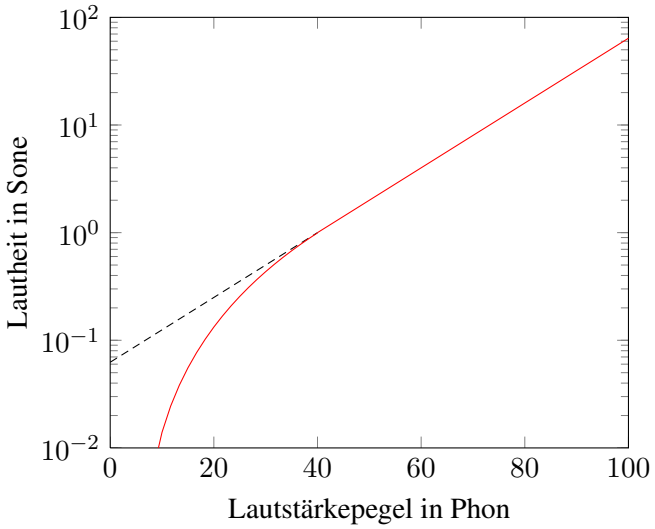



Abbildung 2.6: Zusammenhang zwischen Lautstärkepegel und Lautheit.

**Tonheit** Die Tonheit beschreibt die Wahrnehmung der Tonhöhe beim Menschen. Die Tonhöhe wird durch die Frequenz bestimmt. Bis zu 1 kHz steigt die wahrgenommene Tonhöhe linear an, darüber hinaus ist sie logarithmisch. Die Tonheit wird in *Mel* gemessen, dabei entspricht die wahrgenommene Tonhöhe bei 1 kHz 1000 Mel. Sie lässt sich approximativ durch die Formel

$$Mel(f) = 2595 \log_{10}(1 + f/700) \tag{2.1}$$

berechnen [MBE10][Nie83]. Die Funktion ist in Abbildung 2.7 abgebildet.

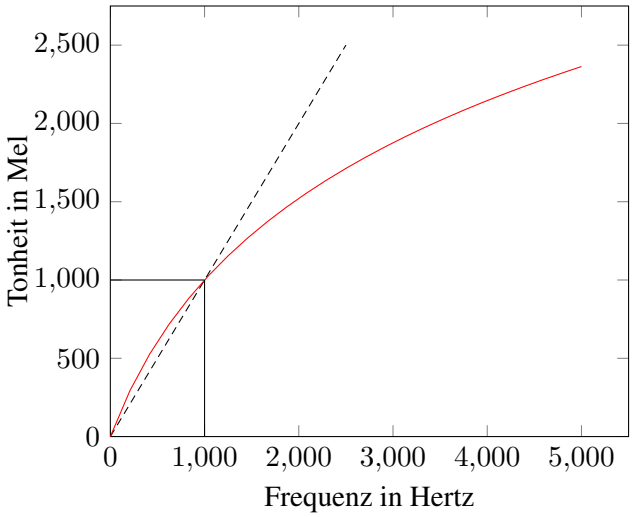



Abbildung 2.7: Zusammenhang zwischen Frequenz und Tonheit.

**Schärfe** Die Schärfe wird bei sehr schmalbandigen Ereignissen empfunden. Je höher die Frequenz, desto eher macht sich die Schärfe bemerkbar. Sie wird in der Einheit *Acum* gemessen.

**Rauhigkeit** Wenn sich bei tiefen Signalen Frequenzen schnell ändern, macht sich das durch einen rauen Eindruck bemerkbar. Die Rauhigkeit wird in *Asper* gemessen und ist beispielsweise bei einer Stimme im Stimmbruch hörbar [Her09].

**Schwankungsstärke** Die Schwankungsstärke bildet die wahrgenommene subjektive Empfindung der Schwankung der Lautstärke ab.
Die Psychoakustik bietet eine Grundlage für Datenkompressionsalgorithmen wie MP3 oder auch Featureextraktionsmethoden. Im Rahmen dieser Diplomarbeit sind allerdings die Lautheit und Tonheit hervorzuheben, die bei den MFCC von besonderem Interesse sind.

#### 2.1.2.4 Mel Frequency Cepstral Coefficients

Die MFCC sind Features von Audiosignalen, die zur Sprach- bzw. Sprechererkennung verwendet werden können. Wie bei anderen Featureextraktionsmethoden wird auch hier darauf abgezielt, die informationstragenden Anteile des Signals zu repräsentieren. In Abschnitt 2.1.2.1 wurde bereits erwähnt, dass die im Vokaltrakt geformten niederfrequenten Anteile eines Sprachsignals den Informationsgehalt über das Gesprochene enthalten. Auch die psychoaktustischen Eigenschaften des menschlichen Gehörs spielen hier eine Rolle. Insbesondere die in Mel gemessene Tonheit ist von speziellem Interesse, wie auch das *Mel* im Namen andeutet. Ein weiterer Vorteil der MFCC ist die kompakte Darstellungsform.
Die MFCC basieren auf den 1963 von Bogert, Healy und Tukey veröffentlichen Cepstralen Koeffizienten [BHT63], die jedoch nicht die psychoakustischen Eigenschaften des menschlichen Gehörs miteinbeziehen. Der Kunstbegriff *Cepstrum* entstand durch die Vertauschung der Buchstaben des Wortes Spektrum (engl. *spectrum*), da es sich hierbei um eine inverse Fouriertransformation des logarithmierten Spektrums handelt. Dementsprechend wird auch die Einheit des Cepstrums *Quefrenz* genannt (auf Basis der Einheit *Frequenz*), ein Filter der am Cepstrum angewendet wird, analog dazu auch *Lifter*.
Für die Berechnung der MFCC sind sechs Arbeitsschritte erforderlich, die im Folgenden näher erläutert werden [Tol10].

1. Framing
2. Fensterung
3. Diskrete Fouriertransformation
4. Abbildung auf die Mel-Skala
5. Logarithmierung des Betragsspektrums und Diskrete Cosinustransformation
6. Berechnung der Delta- und Delta-Delta-Koeffizienten

**Framing** Da Sprache ein sich zeitlich kontinuierlich änderndes Signal ist, muss dieses in Teilbereiche unterteilt werden, die jeweils separat analysiert werden. Im Idealfall würden einzelne Worte für die weitere Berechnung ausgeschnitten. Da die Detektion einzelner Worte aber sehr performancelastig ist, wird stattdessen das Audiosignal in gleich lange Teilbereiche unterteilt. Diese sind typischerweise zwischen 20 und 30 ms lang, wobei sich die Frames teilweise überlappen.

**Fensterung** Jeder Frame wird mit einer Funktion gewichtet, sodass die Randbereiche abgeschwächt werden. Häufig werden Hamming-Fenster für die Gewichtung verwendet (die grafische Darstellung des Hamming-Fensters ist in Abbildung 2.8 abgebildet). Zweck der Fensterung ist es, dass Kanten an den Rändern abgeschwächt werden, um Diskontinuitäten zu verhindern [Tol10] [L$^+$00]. Die Funktion des Hamming-Fensters ist:

$$w(n) = 0,54 + 0,46 \cos\left(\frac{2\pi n}{M}\right), n = -\frac{M}{2}, ...\frac{M}{2} - 1 \tag{2.2}$$

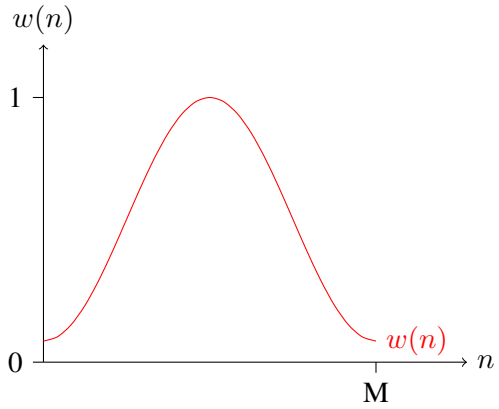



Abbildung 2.8: Hamming-Fenster-Funktion.

**Diskrete Fouriertransformation** Mit der diskreten Fouriertransformation werden die gefensterten Teilbereiche des Signals in den Frequenzbereich transformiert. Die Fouriertransformation bewirkt, dass die Faltung des Anregungssignals mit dem Filter in eine Multiplikation umgewandelt wird. In diesem Schritt wird die Phaseninformation verworfen, da sie bei der Spracherkennung im Gegensatz zur Amplitude weit weniger wichtig ist [L$^+$00]. Die schnelle Fouriertransformation (FFT) ermöglicht eine raschere Berechnung, indem bekannte Zwischenergebnisse wiederverwendet werden und somit viele Operationen nicht mehr berechnet werden müssen. Die Komplexität der FFT ist $N \cdot \log(N)$.

**Abbildung auf die Mel-Skala** Um die psychoakustischen Eigenschaften des menschlichen Gehörs abzubilden, werden die Frequenzen auf die Mel-Skala transformiert [Der09]. Wie schon

in Abschnitt 2.1.2.3 angesprochen, steigt die wahrgenommene Lautstärke bei einer Frequenzverdopplung eines Lautes unter 1 kHz linear an, sodass die doppelte Frequenz die doppelte Lautstärke verursacht. Darüber hinaus jedoch erfolgt die Steigung der Lautstärke nur noch logarithmisch [MBE10]. Das menschliche Ohr verfügt also über eine hohe Auflösung in niedrigen Frequenzen, die detaillierter wahrgenommen werden als höhere [FTKI92] [Fan73] [L$^+$00].
Die daraus resultierende Mel-Filterbank enthält ca. 30 dreieckförmige, sich teilweise überlappende Filter, welche auf das Signal angewendet werden. Diese Abstände der Filter zueinander sind im Mel-Bereich gleichmäßig.

**Logarithmierung des Betragsspektrums und diskrete Cosinustransformation** Die Mel-Frequenzen werden logarithmiert, um die logarithmische Wahrnehmung der Tonhöhe abzubilden. Differenzen bei hohen Frequenzen sind weniger leicht für den Menschen auszumachen als Unterschiede bei niederfrequenten Tönen.
Diese Werte werden zuletzt noch mittels einer diskreten Cosinustransformation (DCT) in den Zeitbereich zurückgewandelt [HJR04]. Die diskrete Cosinustransformation hat gegenüber der inversen Fouriertransformation den Vorteil, dass diese einfacher zu berechnen ist.
Die resultierenden MFCC sind nach Wichtigkeit geordnet. Für die Spracherkennung reicht es, die ersten zwölf Koeffizienten als Merkmale zur Klassifizierung zu verwenden. Eine Rücktransformierung vom Quefrenz- in den Frequenzbereich ist für die anschließende Klassifikation nicht nötig [LN10]. Die Koeffizienten mit niedrigem Index beschreiben die niederfrequenten, durch die Formanten geprägten Anteile des Audiosignals, die höheren Indizes beschreiben die sprecherspezifischen, höherfrequenten Details [Eis08].

**Berechnung der Delta- und Delta-Delta-Koeffizienten** Zusätzlich zu den MFCC können die Delta- und Delta-Delta-Koeffizienten ermittelt werden. Diese entsprechen der ersten bzw. zweiten Ableitung der Merkmale. Die Delta-Koeffizienten sind somit unempfindlich auf additive Störungen und bilden die zeitlichen Änderungen ab [BKB10]. Sie lassen sich durch

$$\Delta[n] = s[n] - s[n-1] \tag{2.3}$$

berechnen, die Delta-Delta-Koeffizienten analog dazu mittels

$$\Delta\Delta[n] = \Delta[\Delta[n]]. \tag{2.4}$$

Schlussendlich ergibt sich ein 39-dimensionaler Featurevektor, mit dem das Audiosignal beschrieben wird. Diese Zahl setzt sich zusammen aus den zwölf MFCC, der Energie (1), sowie jeweils zwölf Delta- und Delta-Delta-Koeffizienten mit den jeweiligen Energiewerten (jeweils 1). Allerdings ist die Anzahl der zu verwendeten Features je nach Einsatzgebiet unterschiedlich. Für die Musikgenre-Erkennung liefert die Verwendung der ersten fünf Koeffizienten die besten Resultate [TC02], für die Spracherkennung sind wie bereits angesprochen zwölf Koeffizienten erforderlich.
Für die Klassifikation der MFCC-Merkmalsvektoren können verschiedene Methoden herangezogen werden, eine davon ist der Vergleich mittels Dynamic Time Warping (DTW). Bei dieser Methode wird die Ähnlichkeit durch die Kosten, die beim Warping entstehen, ermittelt [MBE10]. Auf die Klassifizierung mittels DTW wird in Abschnitt 2.2.2 detailliert eingegangen.

### 2.1.3 Features in Bildern

Die Features, die aus Bildern extrahiert werden, basieren auf der Funktionsweise des menschlichen Gehirns. Der Sehsinn bzw. die Verarbeitung der visuellen Reize nimmt die relevanten Informationen des Umfelds auf, welche anschließend verarbeitet werden. Ziel bei der Computer Vision ist es, den Sehsinn nachzubilden. Aus diesem Grund wird im folgenden Abschnitt zunächst die visuelle Wahrnehmung beim Menschen beschrieben, worauf anschließend auf die Nachbildung bei der Computer Vision eingegangen werden kann.

#### 2.1.3.1 Die visuelle Wahrnehmung beim Menschen

Bei der visuellen Wahrnehmung werden relevante Merkmale aus den visuellen Reizen extrahiert, verarbeitet und interpretiert. Bestimmte Bereiche des menschlichen Gehirns extrahieren gewisse Merkmale des Gesehenen, diese sind im Speziellen die Tiefeninformation, Form, Bewegung oder Farbe [Ang06] [Mit02]. Beim Sehvorgang existieren parallele Algorithmen, die die unterschiedlichen Informationen aufnehmen. Die erfassten Informationen werden anschließend zusammengeführt. Beim menschlichen Sehverhalten ist es so, dass die Bereiche mit Diskontinuitäten verstärkt ins Interesse rücken. Unter Diskontinuität versteht man Intensitätsänderungen, welche wiederum durch unterschiedliche räumliche Beziehungen von Objekten zueinander oder durch unterschiedliche Materialien hervorgerufen werden können [Eis11]. Die Umgebung wird nicht komplett erfasst, sondern beschränkt sich auf einige wenige relevante Punkte. Diese Punkte werden auch *saliente Punkte* genannt. So sind beispielsweise Eckpunkte informationstragender als Kanten. In Abbildung 2.9 ist ersichtlich, wie wichtig Punkte an starken Krümmungen bzw. Eckpunkten sind. Sollten Eckpunkte aus einem Bild entfernt werden, ist die Rekonstruktion der Form des Objekts im Gehirn nicht möglich, bei Entfernung von Kanten allerdings schon. Biedermann [Bie87] verwendet in einem Experiment Linienzeichnungen von Objekten. Es werden jeweils 25, 45 und 65 Prozent der Linien im Bereich der Eckpunkte bzw. der Kanten entfernt. Die Erkennungsrate ist bei Entfernung der Kanten schlechter als bei Entfernung der Eckpunkte.

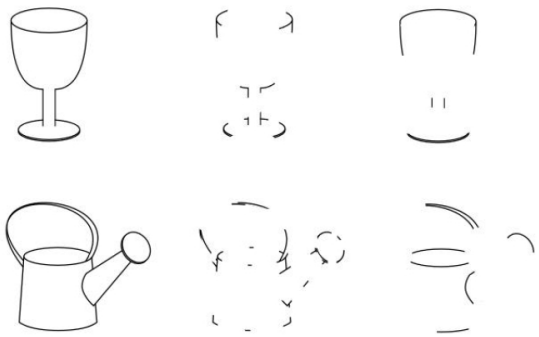


Abbildung 2.9: Die Form des abgebildeten Objektes lässt sich eher rekonstruieren, wenn Konturpunkte vorhanden sind, die an Eckpunkten liegen. Links ist die Originalzeichnung abgebildet, in der Mitte die Zeichnung mit Konturpunkten an Ecken, rechts die Zeichnung mit Konturpunkten an Kanten mit geringer Krümmung (Quelle: [Bie87]).

Die Funktionsweise der visuellen Wahrnehmung dient als Basis für die computerunterstützte Verarbeitung. Hier werden ebenso zunächst relevante Features extrahiert und anschließend verarbeitet [Mit02]. Die Hauptcharakteristika in einem Bild sind wie bei der visuellen Wahrnehmung Farbe, Textur und Kanteninformation. Die Farbinformation wird oft in der Form von Histogrammen repräsentiert, bei der unter Verlust der räumlichen Information die Anteile der vorkommenden Farben dargestellt werden. Ähnliche Histogramme deuten somit nicht unbedingt auf ähnliche Bildinformationen. Aus diesem Grund können allerdings räumliche Histogramme konstruiert werden, die das Bild in Teilbereiche unterteilen, für die jeweils ein Histogramm erstellt wird [Käs05]. Dadurch bleibt bis zu einem gewissen Grad die räumliche Information erhalten. Unter einer Textur versteht man ein regelmäßig auf der Oberfläche vorkommendes Muster, wodurch zusammengehörende Bereiche erkannt werden können. Im Gegensatz zur Farbinformation ist die Textur nicht auf einzelne Pixel beschränkt, sondern beschreibt eine räumliche Ausdehnung [Käs05]. Kanten und Eckpunkte sind ebenfalls beliebte Features in Bildern. Kanten in Bildern trennen Flächen mit unterschiedlichen Farb- bzw. Grauwerten mit ausreichend großen Unterschieden. Ein Graustufenbild lässt sich durch eine zweidimensionale Intensitätsmatrix repräsentieren, die Kanten liegen genau an den Stellen, an denen eine sprunghafte Änderung der Intensität erfolgt. Mathematisch lässt sich eine Kante also durch die Bestimmung des Gradienten ermitteln. Dieser ist ein Vektor, der immer in die größte Intensitätsänderung zeigt. Kantendetektoren werden üblicherweise durch Faltung mit einem Filter ermittelt, der den Gradienten abbildet. In Eckpunkten erfolgt im Vergleich zu Kanten eine Änderung in zwei Dimensionen. Eckpunkte sind als Features für die Objekterkennung von besonderem Interesse, da diese wie oben beschrieben einen viel höheren Informationsgehalt beinhalten als Kanten. Die Eckpunkt-Detektion ist ein Spezialfall der Interest-Point-Detektion und ist im Rahmen dieser Arbeit besonders relevant, wie dem folgenden Abschnitt zu entnehmen ist.

#### 2.1.3.2 Interest-Point-Detektion

Interest-Point-Detektoren ermitteln Punkte in einem Bild, die sich anhand unterschiedlicher Kriterien von deren Umgebung abheben und somit einen höheren Informationsgehalt beinhalten. Interest Points zählen zu den lokalen Features. Darunter versteht man diese Teile eines Bildes, die von ihrer unmittelbaren Umgebung abweichen [MB12]. Wichtige Eigenschaften lokaler Features sind laut Tuytelaars und Mikolajczyk [TM08]:

- **Wiederholbarkeit** *(engl. repeatability)*: Features sollten auch detektiert werden, wenn sich der Aufnahmestandpunkt oder die Lichtverhältnisse ändern. Bei zwei verschiedenen Bildern eines Objekts mit unterschiedlichem Standpunkt und Winkel der Aufnahme sollten die detektierten Features an derselben Stelle liegen.

- **Unterscheidbarkeit** *(engl. distinctiveness/informativeness)*: Die Features sollten untereinander gut unterscheidbar sein, sodass ein Matching möglich ist.

- **Lokalität** *(engl. locality)*: Durch die Lokalität wird gewährleistet, dass sich Features nicht überlappen und unterschiedliche Bereiche im Bild beschreiben.

- **Genauigkeit** *(engl. accuracy)*: Die Detektion soll positionsmäßig genau erfolgen.

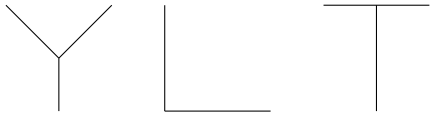


Abbildung 2.10: Verschiedene Arten von Eckpunkten laut Schmid et al. [SMB00], die Formen der Eckpunkte bestimmen die Bezeichnungen Y-, L- und T-Eckpunkt.

- **Quantität** *(engl. quantity)*: Auch bei kleinen Objekten soll der Featuredetektor eine ausreichende Anzahl an Features liefern, sodass das abgebildete Objekt gut beschrieben wird.

- **Effizienz** *(engl. efficiency)*: Die Dauer der Detektion darf nicht zu lange dauern, da sonst eine Verwendung bei zeitkritischen Applikationen nicht möglich ist.

Schmid et al. [SMB00] definieren als Interest Points diejenigen Punkte, in denen eine zweidimensionale Signaländerung der Intensität stattfindet. Diese Definition umfasst Y-, L- und T-Eckpunkte, einzelne schwarze Punkte auf einer weißen Fläche aber nicht (siehe Abbildung 2.10). Als Entscheidungskriterien werden Intensitätsunterschiede, unterschiedliche Konturen bzw. parametrische modellbasierte Methoden verwendet. Nachteil eines reinen kanten- oder eckpunktbasierten Interest-Point-Detektors ist, dass hauptsächlich Punkte an harten Kanten in texturreichen Regionen detektiert werden. Diese Punkte beinhalten allerdings nicht unbedingt eine Information über die Form des abgebildeten Objekts [Käs05].

Interest Points enthalten außerdem eine definierte Position innerhalb eines Bildes, dessen unmittelbare Umgebung relevante, bildbeschreibende Information liefert. Sie sind stabil bei veränderten Bedingungen, sodass diese auch bei anderen Lichtverhältnissen oder anderen geänderten Bedingungen erkannt werden und sind skalierungsinvariant [KS04].

Ein bekannter Interest-Point-Detektor ist der Harris Corner Detector, welcher eine Verbesserung des Moravec-Operators ist. Grundlegende Idee des Operators ist, dass auf einfärbigen Regionen sich die Intensitätswerte in alle Richtungen gleich verändern. Auf Kanten gibt es in eine Richtung eine starke Intensitätsänderung, bei Ecken in alle Richtungen starke Intensitätsänderungen. Nachteil des Harris Corner Detectors ist, dass er nicht skalierungsinvariant und nicht invariant gegenüber affinen Transformationen ist [MS04]. Diese Skalierungsinvarianz ist allerdings unerlässlich, da sonst Objekte in Bildern in verschiedenen Größen nicht zuverlässig beschrieben und klassifiziert werden können.

Interest Points alleine bieten allerdings keine Möglichkeit, Klassifikationen durchzuführen, da sie nur die Koordinaten interessanter Punkte ermitteln, aber nicht deren Umgebung beschreiben. Bei der Keypoint-Detektion wird die Umgebung der Interest Points beschrieben. Die Information der Umgebung kann von verschiedenen Faktoren abhängig sein, wie z. B. Gradienten oder Farbhistogramme. Ein wichtiger Keypoint-Detektor ist Scale Invariant Feature Transform (SIFT).

#### 2.1.3.3 Keypoint-Detektion am Beispiel SIFT

SIFT wurde 1999 von David Lowe [Low99] veröffentlicht und kombiniert einen Merkmalsdetektor mit einem Deskriptor. Dieses Verfahren ist illuminations-, rausch-, rotations- und skalierungsinvariant sowie unempfindlich auf perspektivische Verzerrungen [Gre05]. Das hat den Vorteil, dass Objekte trotz unterschiedlicher Bedingungen der oben aufgelisteten Eigenschaften zuverlässig erkannt werden. Mittels SIFT ist es möglich, korrespondierende Punkte in zwei verschiedenen Bildern zu finden. Somit eignet sich dieses Verfahren für die Objekt- bzw. Gestenerkennung oder für das automatische Stiching von Einzelbildern zu Panoramafotos, auch in der Medizin finden sich Anwendungsbeispiele [Gre05].
Der SIFT-Deskriptor besteht aus vier Arbeitsschritten [Low04]:

1. **Extremstellendetektion in verschiedenen Skalierungsvarianten**: Dieser Arbeitssschritt ermöglicht die Skalierungsinvarianz, da verschiedene Skalierungen betrachtet werden.

2. **Keypoint-Lokalisierung**: In diesem Schritt werden alle Interest Points eliminiert, die nicht aussagekräftig sind, weil sie zu wenig Kontrast aufweisen oder entlang von Kanten liegen.

3. **Orientierungszuweisung**: Um Rotationsinvarianz zu erreichen, wird der Gradient der stärksten Änderung ermittelt, anhand dem die Bildregionen dann ausgerichtet werden. Somit wird Rotationsinvarianz erreicht.

4. **Generierung der Keypoint-Deskriptoren**: Abschließend werden die Keypoints relativ zu deren Orientierung berechnet. Diese werden pro Keypoint zu 128 Werten zusammengefasst, die in einem Merkmalsvektor gespeichert werden.

Um die SIFT-Featurevektoren für ein Klassifikationsverfahren zu verwenden, können die Features mittels dem Bag-of-Features-Verfahren geclustert werden. Von den SIFT-Features existieren auch Abwandlungen, zu denen die DSIFT- und PHOW-Features gehören. Diese haben den Vorteil der schnelleren Berechenbarkeit. Bei DSIFT werden Punkte auf einem dichten Raster mit einer fixierten Skalierung und Orientierung beschrieben und werden oft bei der Objekterkennung verwendet. PHOW-Features wiederum sind eine spezielle Form der DSIFT-Features. Hier werden die Features bei verschiedenen Skalierungen extrahiert [VF08].
Sollten diese Features zur Beschreibung von Objektklassen dienen, wird ein Clusteringverfahren benötigt. Häufig werden dafür die SIFT-Features mit der Bag-of-Features-Methode gruppiert. Wie dieses Verfahren im Detail aussieht, ist dem folgenden Abschnitt zu entnehmen.

#### 2.1.3.4 Bag of Features

Die Bag-of-Features-Methode basiert auf der Bag-of-Words-Methode der Dokumentklassifizierung [CDF+04]. Das **Bag-of-Words-Verfahren** versucht, Texte anhand der darin vorkommenden Wörter für eine Klassifikation zu analysieren, wobei der Bezug der Wörter untereinander verloren geht. Jedes Dokument wird als *Bag* bezeichnet und besteht aus Wörtern eines Wörterbuchs. Eine prominente Anwendungsmöglichkeit davon ist beispielsweise automatisierte Spamerkennung bei E-Mails. Zunächst werden in der Trainingsphase Texte, bei denen die

Klassenzugehörigkeit bekannt ist, auf die vorkommenden Wörter und deren Häufigkeit analysiert. Eventuell können sehr häufig gebrauchte Wörter ignoriert werden, die keine Information über den eigentlichen Inhalt beinhalten (wie z. B. Artikel). Nachdem die Trainingstexte analysiert wurden, können andere Texte klassifiziert werden, indem die Testdaten ebenfalls auf die vorkommenden Wörter untersucht und mit den Wörtern der Trainingsdaten verglichen werden. Die größte Ähnlichkeit bestimmt dann die Klasse des Textes. Um die Funktionsweise besser zu verstehen wird nun anhand der Spamerkennung versucht, das Verfahren näher zu erläutern. Angenommen, es gibt zwei Klassen *Spam* und *Kein Spam*. Alle Wörter, die mit Spam assoziiert sind, wie z. B. Viagra, Sex etc., befinden sich in der ersten Klasse. Die Klasse *Kein Spam* enthält die Wörter, die mit normalen E-Mails assoziiert werden, wie Namen von Freunden, Kollegen etc. Mittels z. B. eines Bayes-Klassifikators wird dann der Text eines erhaltenen E-Mails anhand der darin vorkommenden Wörter analysiert und in eine der beiden Klassen eingeteilt. Die Funktionsweise eines solchen Klassifikators ist in Abschnitt 2.2.4 näher erörtert. Die Klassifizierung ist allerdings nicht nur auf zwei Klassen beschränkt, sondern ist mit beliebig vielen Klassen möglich. Anzumerken ist, dass die Relationen zwischen den Wörtern und somit die eigentliche Information der Texte verloren geht, aber trotzdem eine zuverlässige Klassifikation möglich ist.
Bei der **Bag-of-Features-Methode** wird hier statt mit Wörtern mit interessanten Merkmalen innerhalb des Bildes gearbeitet. Obwohl auch hier der Bezug der Features zueinander – also die räumliche Information der betrachteten Ausschnitte – verloren geht, funktioniert die Klassifizierung trotzdem zuverlässig. Analog zum Bag-of-Words-Verfahren wird hier das Bild als *Bag* betrachtet, das die visuellen Wörter (auch *Keypoints* genannt) beinhaltet.
Um die Features zu ermitteln, wird das Bild in kleine Ausschnitte zerteilt. Für jeden dieser Ausschnitte werden Interest Points ermittelt, die anschließend gruppiert werden. Diese Gruppen können zur anschließenden Klassifizierung verwendet werden [NJT06]. Ein Vorteil der Bag-of-Features-Methode ist, dass die Klassifizierung relativ resistent gegenüber Überlappungen von Objekten ist [NJT06].
Zunächst müssen die Features von Trainingsbildern ermittelt werden, dessen Klassenzugehörigkeit bekannt ist. Aus diesen Interest Points wird ein Vokabular (auch Wörterbuch oder Codebook genannt) erstellt, mit dem anschließend andere Bilder klassifiziert werden können [FFP05]. Das Bag-of-Features-Verfahren besteht aus folgenden Arbeitsschritten, auf die im Folgenden näher eingegangen wird:

1. Featuredetektion
2. Featurerepräsentation
3. Erstellung eines Vokabulars
4. Klassifizierung von unbekannten Bildern

**Featuredetektion** Um ein Bild klassifizieren zu können, muss dieses zunächst auf Merkmale (engl. *Features*) untersucht werden. Die einfachste Methode ist, das Bild in gleich große Patches zu unterteilen. Diese Ausschnitte können dann als Features für die weitere Analyse verwendet werden.

Ein besserer Ansatz ist die Verwendung der oben beschriebenen Interest-Point-Detektoren, da hier Regionen mit hohem Informationsgehalt stärker gewichtet werden als solche mit geringem Informationswert. Anforderungen an einem Interest-Point-Detektor sind somit die Reproduzierbarkeit der Interest Points (sie dürfen nicht willkürlich ausgewählt werden), dass für die schnelle Weiterverarbeitung nur wenige Interest Points detektiert werden damit eine schnellere Weiterverarbeitung möglich ist und dass die Information aussagekräftig ist (siehe Abschnitt 2.1.3.2).

**Featurerepräsentation** Nachdem die Features im Bild ermittelt wurden, müssen diese abstrahiert und in für eine schnelle Klassifikation in Merkmalsvektoren dargestellt werden. Die Features müssen rotations- und skalierungsinvariant sein und dürfen nicht auf Rauschen und Intensitätsschwankungen anfällig sein. Nur dann können bei unterschiedlichen Bedingungen zuverlässig Entscheidungen getroffen werden. Für die Featureextraktion und -repräsentation kann SIFT verwendet werden.

**Erstellung des Vokabulars** Für den Aufbau eines Vokabulars werden zunächst wie oben beschrieben die SIFT-Features jeder Region um einen Interest Point ermittelt und anschließend zu Cluster zusammengefasst, um für die Objekterkennung eingesetzt zu werden. Beim Clustering werden ähnliche Merkmale zu zusammengehörenden Klassen zusammengefasst. Für Klassenzuordnung wird häufig der k-Means-Algorithmus verwendet. Voraussetzung bei Clusteringverfahren ist, dass die Daten innerhalb einer Klasse so homogen wie möglich und die verschiedenen Klassen zueinander so heterogen wie möglich sind [WZ01].

Der k-Means-Clustering-Algorithmus ist ein iteratives Clusteringverfahren. Zunächst wird die Anzahl der verschiedenen Klassen festgelegt. Die Positionen der $k$ Mittelpunkte werden im ersten Schritt zufällig innerhalb der Daten ausgewählt. Die Ergebnisse sind allerdings besser, wenn die Mittelpunkte so gut wie möglich verstreut und die Abstände zwischen den Punkten so groß wie möglich sind [Mat]. Anschließend werden aus allen anderen Datenpunkten die dazu am nächsten liegenden Mittelpunkte zugeordnet. Die Schwerpunkte der zusammengehörenden Datenpunkte werden ermittelt und der Mittelpunkt an ebendiese Punkte verschoben. Dieser Vorgang wiederholt sich so lange, bis sich die Mittelpunkte von den Schwerpunkten nicht unterscheiden [TSK06]. Das Resultat ist ein in $k$ Klassen eingeteilter Datensatz. Die einzelnen Arbeitsschritte des Verfahrens sind in Abbildung 2.11 grafisch dargestellt. Pro Cluster ist mindestens ein Element vorhanden. Die Cluster überlappen sich nicht, jedes Element ist also genau einem Cluster zugeordnet.

Die ermittelten Cluster bilden nun das Wörterbuch der Bag-of-Features-Methode. Nachteil des k-Means-Clustering-Verfahrens ist, dass der Parameter $k$ nicht automatisch bestimmt werden kann [CDF+04]. Es muss also die Anzahl der Klassen schon im Vorhinein bekannt sein. Außerdem ist das Ergebnis der Cluster von der Wahl der Startpunkte abhängig [Loh10].

Beim k-Means-Clustering-Verfahren wird die Funktion

$$\sum_{j=1}^{k} \sum_{i=1}^{n} \| x_{i,j} - c_j \|^2 \tag{2.5}$$

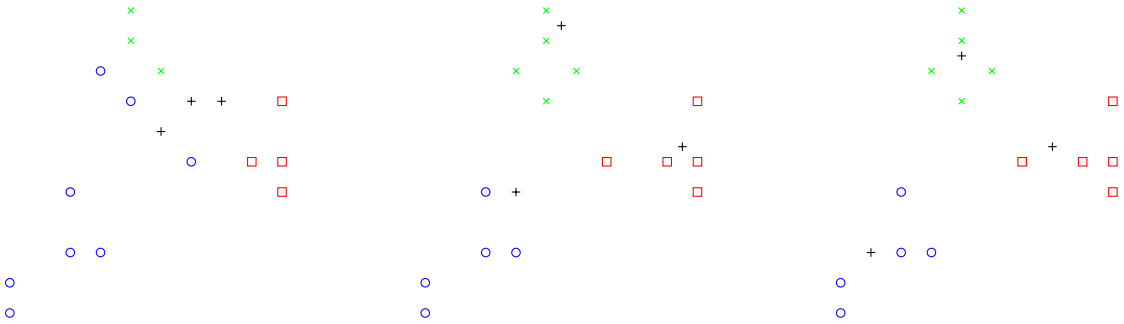


Abbildung 2.11: Verschiedene Iterationsschritte beim k-Means-Clustering (nach [TSK06]).

minimiert, wobei $x_{i,j} - c_j$ die Distanz zwischen zwischen dem Cluster-Mittelpunkt und dem jeweiligen Datenpunkt ist [Loh10]. Als Maß für die Distanz wird meist der euklidische Abstand verwendet, aber auch andere wie z. B. die Manhattan-Distanz sind möglich [Dav02].
Die einzelnen Arbeitsschritte des Algorithmus schauen laut Held [Hel08] und Lohninger [Loh10] folgendermaßen aus:

- **Initialisierung:** Auswahl zufälliger Cluster-Mittelpunkte. Die Anzahl der Mittelpunkte richtet sich an der Anzahl der gewünschten Cluster.
- **Klassifizierung:** Berechnung der Distanzen zwischen den Datenpunkten und den jeweiligen Mittelpunkten. Der nächste Mittelpunkt bestimmt die Clusterzugehörigkeit.
- **Berechnung der Cluster-Mittelpunkte:** Anhand dieser Distanzen werden die Mittelpunkte in die Schwerpunkte der Cluster verschoben.
- **Iteration:** Dieser Vorgang wird so lange wiederholt, bis sich die Cluster-Mittelpunkte nicht mehr bzw. nur noch kaum verschieben.

Jeder Cluster-Mittelpunkt beschreibt ein Wort im Codebook und jedes Feature ist einem dieser Cluster zugeordnet. Anhand dieser Zuordnung lässt sich das Bild als Histogramm repräsentieren. In jedem zu analysierenden Bild werden die Features mit dem Vokabular verglichen. Das Histogramm, das für die Klassifizierung verwendet wird, enthält die Häufigkeiten der visuellen Wörter aus dem Vokabular.
Die Anzahl der Wörter im Vokabular muss mit Bedacht gewählt werden: Ist sie zu klein, werden nicht alle Wörter im Vokabular abgebildet, ist die zu groß, werden zu viele Details mit einbezogen, was ebenfalls für die Klassifizierung nicht von Vorteil ist. In solch einem Fall spricht man auch von *Overfitting*.

**Klassifizierung von unbekannten Bildern** Anhand des erstellten Wörterbuchs können Bilder analysiert werden. Von den zu klassifizierenden Bildern werden deren SIFT-Features ermittelt und mit den Histogrammen des Trainingsdatensatzes verglichen. Für die Kategorisierung lassen sich verschiedene Klassifikationssalgorithmen verwenden. Weit verbreitet sind u. a. SVMs, die

Naive-Bayes-Klassifikation oder auch der kNN-Algorithmus, die in Abschnitt 2.2 näher erläutert werden.

#### 2.1.3.5 Template Matching

Das letzte Featureextraktionsverfahren, das vorgestellt wird, ist Template Matching, das bei der Ähnlichkeitsberechnung zu Prominentengesichtern verwendet wird.
Beim Template Matching wird in einem oder mehreren Bildern ein Ausschnitt, das sogenannte *Template* bzw. *Schablone*, gesucht. Hierbei wird das Template pixelweise über die Suchbilder gelegt und die Distanz dazwischen berechnet. Sei $R(i, j)$ das Template bzw. Referenzbild und $I(u, v)$ das Intensitätsbild, in dem das Vorkommen gesucht werden soll. Mittels Template Matching wird die Position im Bild $I(u, v)$ ermittelt, an der die größte Ähnlichkeit zwischen $R(i, j)$ und $I(u, v)$ vorhanden ist [BB06]. Template Matching ist ein einfach zu implementierendes Verfahren, das robust ist und hohe Erkennungsraten ermöglicht. Allerdings sollten die Klassen homogen sein und wenige Variationen liefern [Ste93].
Für die Berechnung der Übereinstimmung zweier Bildbereiche gibt es verschiedene Ähnlichkeitsmaße. Die am öftesten verwendeten Berechnungen für die Distanz sind die Summe der Differenzbeträge, der maximale Differenzbetrag, oder der euklidische Abstand [BB06]. Sollte letzterer als Ähnlichkeitsmaß verwendet werden, muss lediglich das Quadrat von $d_E(r, s)$ in Gleichung 2.6 minimiert werden um die größte Ähnlichkeit zu bestimmen.

$$d_E(r, s) = \sqrt{\sum_{i,j \in R} (I(r + i, s + j) - R(i, j))^2} \tag{2.6}$$

Allerdings kann auch der quadratische euklidische Abstand verwendet werden, wobei beim Ausmultiplizieren der jeweiligen Quadrate der Term $\sum R^2(i, j)$ weggelassen werden kann, da er von $r, s$ unabhängig und somit eine Konstante ist [BB06] [Lew95]. Weitere Vereinfachungen sind auch möglich, wenn der Term $\sum f^2(x, y)$ annähernd eine Konstante ist, dieser kann dann ebenfalls weggelassen werden [Lew95]. Der Ähnlichkeitswert zwischen Template und Referenzbild berechnet sich somit mittels dem sogenannten Kreuzkorrelationsterm $c(r, s)$ (siehe Gleichung 2.7 [Lew95]).

$$c(r, s) = \sum_{i,j \in R} (I(r + i, s + j)\ R(i, j)) \tag{2.7}$$

Sollte dieser Wert jedoch keine Konstante sein, sind diese Vereinfachungen nicht möglich. In solchen Fällen wird mittels normalisierter Kreuzkorrelationsberechnung vorgegangen.
Nach der Normalisierung enthält die Ergebnismatrix Werte zwischen $-1$ (der geringsten) und $+1$ (der höchsten Übereinstimmung).

$$\gamma(u, v) = \frac{\Sigma_{x,y}[f(x, y) - \overline{f}_{u,v}][t(x - u, y - v) - \overline{t}]}{\sqrt{\Sigma_{x,y}[f(x, y) - \overline{f}_{u,v}]^2[t(x - u, y - v) - \overline{t}]^2}} \text{ [Lew95].} \tag{2.8}$$

In Gleichung 2.8 entspricht $f$ dem Bild, $\overline{t}$ ist der Mittelwert des Templates und $\overline{f}_{u,v}$ ist er Mittelwert von $f(x, y)$ in der Region des Templates [Lew95].

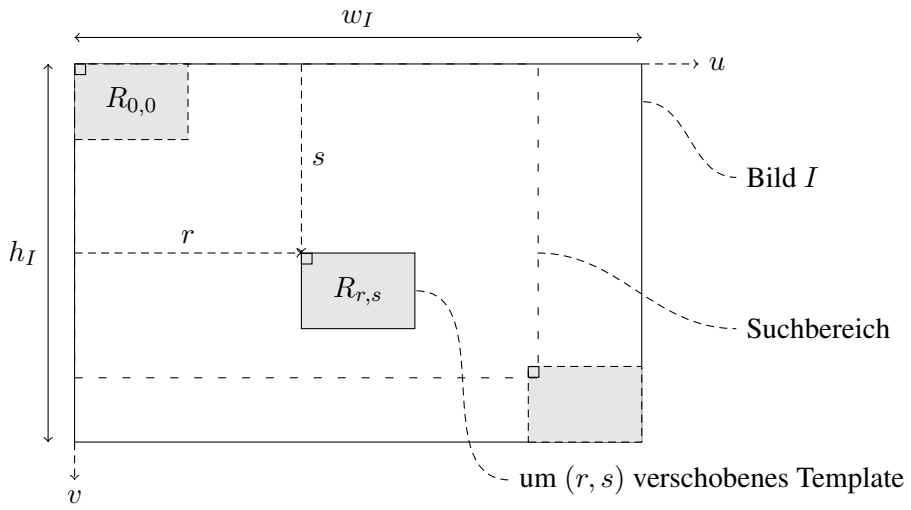



Abbildung 2.12: Template Matching (nach [BB06]).

Problematisch beim Template Matching ist, dass Unterschiede, die vom menschlichen Auge kaum wahrgenommen werden, zu starken Grauwertunterschieden zwischen den beiden Bildern und somit zu einer falschen Klassifikation führen können. Dazu gehören beispielsweise Helligkeitsschwankungen oder Beleuchtungsunterschiede. Das menschliche Auge nimmt oft aufgrund seiner Eigenschaften Bilder als ähnlich wahr, obwohl die Farbwerte der Pixel im direkten Vergleich doch sehr unterschiedlich sind [BB06]. Ein weiterer Nachteil des Template Matching ist, dass das Template theoretisch im gesamten Bild vorkommen kann und sodass überall danach gesucht werden muss, was wiederum sehr rechenaufwändig ist [Ste93].

Des Weiteren ist Template Matching nicht skalierungs- und rotationsinvariant. Das ist insofern dann problematisch, wenn man nicht weiß, in welcher Größe das Template im Bild vorkommt. Dem kann man entgegenwirken, indem die Suche mit verschiedenen Größen des Templates durchgeführt wird, was sich allerdings negativ auf den Rechenaufwand auswirkt, da das Template Maching für jede Größe separat ausgeführt werden muss. Nicht nur die Anzahl der Skalierungen, sondern auch die Anzahl der Bilder, in denen nach dem Template gesucht wird, erhöht logischerweise den Rechenaufwand.

Für die praktische Umsetzung ist es deshalb sinnvoll, Optimierungen durchzuführen. Durch die fehlende Skalierungsvarianz ist es erforderlich, dass die Suche des Templates in verschiedenen Größen erfolgen muss. Durch eine Einschränkung des Suchbereich lässt sich ebenfalls der Rechenaufwand reduzieren. In solchen Fällen ist allerdings Vorwissen über den abgebildeten Inhalt erforderlich. Da im Rahmen dieser Arbeit als Template ein Gesicht verwendet wird, kann der Suchbereich mittels Gesichtserkennungsalgorithmus auf dessen Umgebung eingeschränkt werden. Die detaillierte Beschreibung der Implementierung und den Optimierungsmaßnahmen des Algorithmus ist in Abschnitt 3.2 zu finden.

## 2.2 Klassifikation von Medienbeschreibungen

Die ermittelten Features dienen als Basis für die Analyse der Medienbeschreibungen. In diesem Kapitel wird zunächst auf allgemeine Eigenschaften von Klassifikatoren eingegangen, anschließend werden die für die Implementierung der Demos erforderlichen Verfahren vorgestellt. Die Reihenfolge der Klassifikation richtet sich nach den Demoanwendungen. Zunächst wird das Dynamic Time Warping beschrieben, das für die Klassifizierung von Sprachsignalen verwendet wird, anschließend folgt der kNN-Algorithmus, der sowohl bei der Spracherkennung, als auch bei der Körperteilerkennung verwendet wird. Der Naive-Bayes-Klassifikator ist das letzte vorgestellte Klassifikationsverfahren und wird ebenfalls in der Körperteilerkennung verwendet. Abschließend wird auf Evaluierungsmaße eingegangen, anhand derer die Güte der Ergebnisse beurteilt werden kann.

### 2.2.1 Grundlegendes zu Klassifikatoren

Klassifikatoren sind Methoden, die anhand gewisser Eigenschaften Objekte zu Klassen zusammenfassen. Diese Klassen haben Ähnlichkeiten, die üblicherweise durch extrahierte Features bestimmt werden. Der Vorgang der Einteilung wird Klassifizierung genannt, während das Endresultat die Klassifikation ist.

Klassifikationsverfahren lassen sich in überwachte und unüberwachte Verfahren unterteilen. Bei der überwachten Klassifikation existiert ein Trainingsset mit bekannter Klassenzugehörigkeit der enthaltenen Elemente. Durch dieses Vorwissen lassen sich nun unbekannte Elemente klassifizieren. Unüberwachte Klassifikationsmethoden haben im Gegensatz dazu jedoch kein Trainingsset, sondern sind selbstlernend.

Für die Klassifizierung gibt es verschiedene Methoden. Diejenigen, die für die vorgestellten Audio- und Bild-Features aus Abschnitt 2.1 von Interesse sind, werden im Folgenden noch näher erläutert. Bei der Klassifizierung ist der Inhalt der Featurevektoren irrelevant. Es spielt keine Rolle, ob es sich um Audio- oder Bilddaten die Grundlage bilden, es wird lediglich mit den daraus extrahierten Featurevektoren gearbeitet. Ein Klassifikator ist universal auf Features anwendbar, ohne dass deren nähere Bedeutung bekannt sein muss. Aus diesem Grund wird dieser Abschnitt nicht wie Abschnitt 2.1 in audio- und bildspezifische Methoden unterschieden.

Um die Übersicht zu bewahren, bestimmen die implementierten Demoanwendungen die Reihenfolge der Vorstellung der Klassifikationsmethoden. Zunächst wird auf das Dynamic Time Warping eingegangen, da dieses im ersten Demo mit der Tierlauterkennung verwendet wird, anschließend werden der kNN-Algorithmus und der Naive-Bayes-Klassifikator vorgestellt.

Aufgrund der automatisierten Vorgehensweise der Klassifizierung kommt es auch vor, dass diese fehlerhaft ist. Um die Qualität der Ergebnisse zu bestimmen, gibt es verschiedene Gütemaße, auf die in Abschnitt 2.2.5 noch näher eingegangen wird. Ebenfalls bieten diese Gütemaße Vergleichswerte, durch die sich die Ergebnisse unterschiedlicher Klassifikatoren untereinander vergleichen können.

### 2.2.2 Dynamic Time Warping

Zwei zeitlich unterschiedlich lange dauernde Signale können mittels verschiedener Methoden auf dieselbe Länge gebracht werden: Einerseits kann trivialerweise mittels Linear Time Warping ein Signal global gestreckt bzw. gestaucht werden oder andererseits, wie eben beim Dynamic Time Warping (DTW), Rücksicht auf die im Signal vorhandene Information genommen werden, wodurch ein besseres Resultat entsteht [RM03] [Mül07]. Obwohl ursprünglich von den Entwicklern Kruskal und Liberman 1983 für die Spracherkennung vorgesehen, sind die Anwendungsgebiete von DTW nicht ausschließlich auf zeitabhängige Signale beschränkt. So finden sich beispielsweise Anwendungsgebiete in der Gestenerkennung, Handschrifterkennung und Robotik [Nie04].
Auch Sprachsignale sind bei gleichem Informationsgehalt in der Länge nicht immer genau gleich. So werden beispielsweise Silben unterschiedlich lange betont oder von einem Sprecher zwischen zwei Wörtern Pausen eingehalten, von einem anderen wiederum nicht. DTW ist somit prädestiniert für einen Vergleich unterschiedlicher Sprachsignale. Abbildung 2.13 zeigt zwei stark vereinfachte zeitliche Signale die dieselbe Information beinhalten, die Pfeile dazwischen deuten auf die korrespondierenden Punkte. Mittels DTW werden diese Punkte auf denselben Zeitpunkt verschoben und die Signale dementsprechend gestaucht bzw. gestreckt.

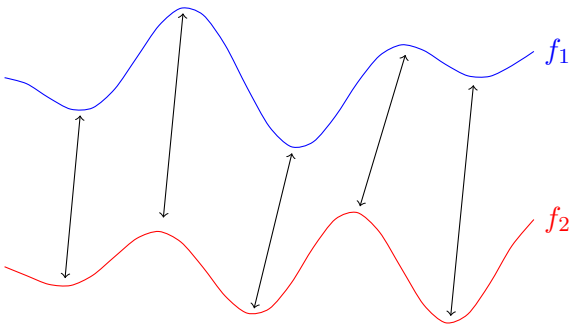



Abbildung 2.13: Zwei zeitabhängige Signale mit derselben Information und deren korrespondierenden Punkte.

Seien $X := (x_1, x_2, ..., x_N)$ bzw. $Y := (y_1, y_2, ..., y_M)$ zwei zeitlich abhängige Sequenzen unterschiedlicher Länge ($N, M \in \mathbb{N}$). Um Punkte dieser beiden Sequenzen zu vergleichen, muss ein Maß für die Kosten festgelegt werden, das auch lokale Distanz genannt wird [Mül07]. Seien den Sequenzen Werte aus dem Feature Space $\mathcal{F}$ zugeordnet. Somit entspricht die Berechnung der Kosten der Abbildung

$$\mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}_{\geq 0} \text{ [Mül07].} \tag{2.9}$$

Die Kosten, die beim DTW aufgewendet werden, können als Maß der Ähnlichkeit dieser beiden Signale gewertet werden. Je niedriger die Kosten, desto ähnlicher sind diese [Mül07]. Anhand der ähnlichsten Werte kann eine Klassenzugehörigkeit des untersuchten Elements bestimmt werden. Um die entstehenden Kosten zu ermitteln, werden zunächst die Werte der beiden Vektoren paarweise verglichen. Zur Distanzberechnung dieser Werte kann der euklidische Abstand verwendet werden [BKB10], aber auch andere Distanzmaße sind möglich [Nie04]. Der euklidische

Abstand berechnet sich durch

$$d(x, y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{2.10}$$

Oft werden diese Werte in einer Distanzmatrix erfasst, die an der Stelle $(i, j)$ die Distanz des i-ten Elements des ersten und des j-ten Element des zweiten Vektors beinhalten [HFMEA11]. Sie ist definiert als $C \in \mathbb{R}^{N \times M}$, wobei $C(n, m) := c(x_n, y_m)$ [Mül07]. Der Pfad mit den geringsten Kosten liefert somit die geringste Distanz dieser beiden Vektoren und dient als Maß für die Ähnlichkeit der beiden Sequenzen (siehe Abbildung 2.14) [RM03]. Eine geringe Distanz deutet auf eine große Übereinstimmung zwischen den beiden Signalen hin [Tol10].

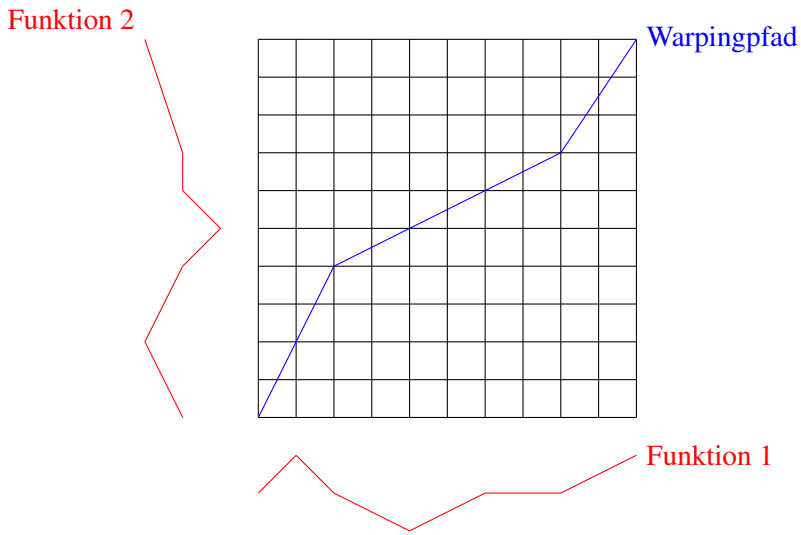



Abbildung 2.14: Der Warpingpfad zwischen zwei zeitabhängigen Funktionen.

In der Praxis wird die Berechnung des Pfades mit den geringsten Kosten meistens mittels dynamischer Programmierung gelöst [HFMEA11]. Darunter versteht man eine besondere Art von Lösungsfindung, bei der das Gesamtproblem aus einer Vielzahl sich wiederholender Teilprobleme besteht. Diese müssen nur einmal berechnet werden und können wiederverwendet werden. DTW kann mit dem kNN-Algorithmus (siehe Abschnitt 2.2.3) kombiniert werden, sodass die Klassifizierung nicht nur anhand der ähnlichsten Audioaufnahme, sondern anhand einer beliebigen Anzahl von Vergleichsaufnahmen bestimmt wird.

### 2.2.3 kNN-Algorithmus

Der k-Nearest-Neighbor-Algorithmus (kNN-Algorithmus) ist ein weit verbreiteter Klassifikator, der wegen seiner einfachen Implementierung sehr beliebt ist. Der Algorithmus ist ein sogenannter Lazy-Learning-Algorithmus. Das bedeutet, dass erst während dem Klassifizierungsvorgang das Modell gebildet wird [CD07]. Es gibt also keine Trainingsphase, bevor mit der Klassifizierung begonnen wird, was wiederum auch bedeutet, dass bei jedem Klassifizierungsvorgang alle Werte im Trainingsset miteinbezogen werden müssen.

Die Problemstellung beim kNN-Algorithmus ist folgende: Gegeben ist ein Datenset mit bekannter Klassenzuordnung und ein unbekannter Datenpunkt, gesucht ist dessen Klassenzugehörigkeit.

Es werden die $k$ ähnlichsten Werte für die Klassifizierung des unbekannten Elements verwendet. $k$ ist eine ganzzahlige, positive Zahl, bei einem $k = 3$ werden die drei nächsten Werte betrachtet, die Mehrheit der Klassen dieser Nachbarn bestimmt die Klassenzugehörigkeit des unbekannten Elements. Die Distanz der $k$ nächsten Werten zum unbekannten Punkt spielt keine Rolle. Es findet nur ein Mehrheitsvoting statt, alle Punkte sind hierbei gleichberechtigt. Allerdings ist es auch möglich den kNN-Algorithmus zu gewichten. Hier wird die Distanz zum Testdatenpunkt ebenfalls in die Klassifizierung einbezogen. Abbildung 2.15 zeigt die Klassifizierung zweier Punkte mit einem $k = 3$. Beim linken oberen Punkt sind von den nächsten drei Punkten zwei zur Klasse *Kreis* und einer zur Klasse *Quadrat* zugehörig, somit gehört dieser Punkt der Mehrheit entsprechend zur Klasse *Kreis*. Beim rechten unteren Punkt sind alle drei Nachbarn Mitglieder der Klasse *Quadrat*, weshalb dieser ebenfalls zu Klasse *Quadrat* gehört.

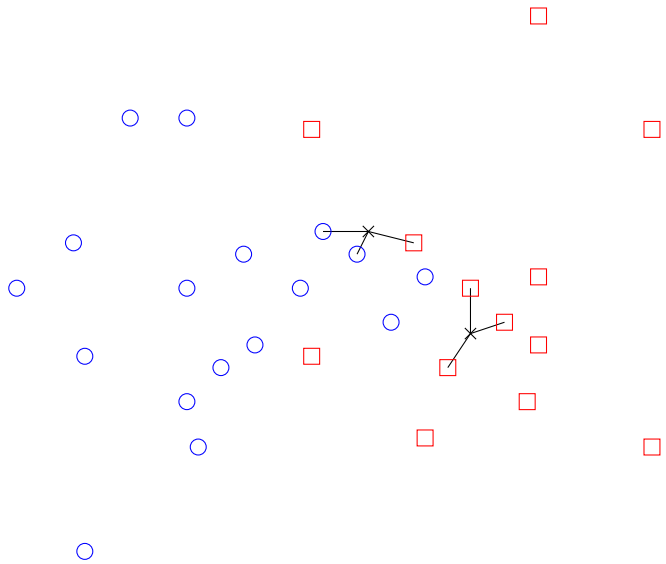


Abbildung 2.15: Bestimmung der Klasse mittels kNN-Algorithmus, $k = 3$.

Das unbekannte, der Klassifizierung unterzogene Element muss nicht nur ein einzelner Datenwert sein, sondern kann auch ein Featurevektor sein. Für die Ähnlichkeit werden verschiedene Berechnungen verwendet, so kann beispielsweise der euklidische Abstand zwischen den Vektoren die Distanz und somit die Ähnlichkeit bestimmen.

Die Komplexität des naiven Ansatzes, bei dem die Distanz zwischen Testdatensatz und allen im Trainingsset vorhandenen Datensätzen berechnet wird, ist $O(nd)$, und kann für große Datensätze problematisch werden [HAYSZ11].

### 2.2.4 Naive-Bayes-Klassifikator

Der Naive-Bayes-Klassifikator basiert auf dem Bayestheorem, mit welchem sich bedingte Wahrscheinlichkeiten berechnen lassen. Eine bedingte Wahrscheinlichkeit $P(A|B)$ ist die Wahrscheinlichkeit, dass ein Ereignis $A$ eintritt, wenn bereits ein anderes Ereignis $B$ eingetreten ist. Sie ist definiert als

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \tag{2.11}$$

falls $P(B) > 0$. Sind $A$ und $B$ unabhängig, also $P(A \cap B) = P(A)P(B)$, dann gilt

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{2.12}$$

Die sogenannte A-priori-Wahrscheinlichkeit $P(A)$ beschreibt die Wahrscheinlichkeit, dass Ereignis $A$ eintritt ohne dass ein Vorwissen vorhanden ist. $P(A|B)$ nennt sich A-posteriori-Wahrscheinlichkeit, und beschreibt nun die Wahrscheinlichkeit des Ereigniseintritts von $A$, nachdem Ereignis $B$ eingetreten ist.
Ein Bayes-Klassifikator ermittelt die Wahrscheinlichkeiten der Klassenzugehörigkeiten der einzelnen Elemente. Die größte Wahrscheinlichkeit bestimmt somit die Zugehörigkeit. Beim Naive-Bayes-Klassifikator wird zusätzlich angenommen, dass die Attribute von den Klassen unabhängig sind, weshalb der Algorithmus dieser Naivität den Namen verdankt [CDF+04] [DP97]. Obwohl in der Praxis selten zutreffend, funktioniert die Klassifizierung trotzdem zuverlässig [CDF+04] [DP97]. Aufgrund dieser Naivität verringert sich die Anzahl der Parameter von $2(2^n - 1)$ auf $2n$ für die Berechnung von $P(F_i|C_j)$ [Mit10]. Ein weiterer Vorteil ist die rasche Klassifizierung.
Beliebt ist die Kombination des Naive-Bayes-Klassifikators mit dem Bag-of-Words-Verfahren zur Textklassifikation sowie dem Bag-of-Features-Verfahren zur Bildklassifikation. Bei der Bildklassifikation werden zunächst die bildbeschreibenden Features eines Trainingssets ermittelt (siehe Abschnitt 2.1.3.4). Angenommen unser Trainingsvokabular $V = v_i$ besteht aus den Bildern $I = I_i$, von denen die Klassenzugehörigkeit bekannt ist. Aus diesem Vokabular werden die visuellen Wörter ermittelt, indem die bildbeschreibenden Features geclustert und die Bilder anhand dieser Cluster durch Histogramme der im Bild vorkommenden Wörter beschrieben werden. Auf dieselbe Art wird auch ein Histogramm des Bildes erstellt, dessen Klassenzugehörigkeit bestimmt werden muss. Angenommen, ein Bild oder ein Dokument, dessen Klassenzugehörigkeit unbekannt ist, besteht aus den Features $F_i$: Für jede Klasse $C_j$ wird nun die Wahrscheinlichkeit der Zugehörigkeit des Dokuments berechnet (siehe Gleichung 2.13).

$$P(C_j|F_i) = \frac{P(C_j)P(F_i|C_j)}{P(F_i)} \tag{2.13}$$

Die größte Wahrscheinlichkeit bestimmt nun die detektierte Klasse. Da der Nenner des Bayestheorems ($P(F_i)$) nicht von der Klasse abhängig ist, sondern für alle Klassen den gleichen Wert hat, ist dieser für die Klassifizierung nicht relevant und kann vernachlässigt werden [DP97] [Ris01]. Die A-posteriori-Wahrscheinlichkeit ist proportional zur A-priori-Wahrscheinlichkeit mal der Likelihood, der Maximalwert bestimmt die Klasse [CDF+04] (siehe Gleichung 2.14).

$$P(C_j|F_i) \propto P(C_j)P(F_i|C_j) \tag{2.14}$$

Durch die angenommene stochastische Unabhängigkeit der Attribute lässt sich die Wahrscheinlichkeit $P(F_i|C_j)$ durch das Produkt $\prod_{i=1}^{n} p(F_i = f_i|C = c)$ berechnen (siehe Gleichung 2.15).

$$P(C_j|F_i) \propto P(C_j)P(F_i|C_j) = \ p(C = c)\prod_{i=1}^{n} p(F_i = f_i|C = c) \tag{2.15}$$

Die größte Wahrscheinlichkeit bestimmt bekanntlich die korrespondierende Klasse. Durch das Weglassen des Nenners kann allerdings nicht mehr von Wahrscheinlichkeiten gesprochen werden – nun entscheidet der Maximalwert des folgenden Terms (siehe Gleichung 2.16).

$$\text{classify}(f_1, \ldots, f_n) = \underset{c}{\text{argmax}}\, p(C = c)\prod_{i=1}^{n} p(F_i = f_i|C = c) \tag{2.16}$$

In der Praxis jedoch lässt sich die Formel in Gleichung 2.16 zur Klassifizierung nicht verwenden. Aufgrund der Tatsache, dass Wahrscheinlichkeiten im Bereich zwischen 0 und 1 liegen und $n$ sehr hoch ist, ist das Produkt $\prod_{i=1}^{n} p(F_i = f_i|C = c)$ aufgrund der begrenzten Speichermenge von Gleitkommazahlen am Computer (dem sogenannten arithmetischen Unterlauf) fälschlicherweise immer 0. Dieses Problem kann durch Logarithmieren des Ausdrucks verhindert werden, die Werte des Intervalls 0 bis 1 werden nun auf den Bereich $-\infty$ bis 0 abgebildet. Da der Logarithmus eine streng monoton wachsende Funktion ist, bestimmt auch der Maximalwert des logarithmierten Wertes die Klasse. Durch die logarithmischen Rechenregeln lässt sich nun Gleichung 2.16 folgendermaßen umformen:

$$\text{classify}(f_1, \ldots, f_n) = \underset{c}{\text{argmax}} \log p(C = c)\prod_{i=1}^{n} p(F_i = f_i|C = c) \tag{2.17}$$

$$= \underset{c}{\text{argmax}} \log p(C = c) + \log\prod_{i=1}^{n} p(F_i = f_i|C = c) \tag{2.18}$$

$$= \underset{c}{\text{argmax}} \log p(C = c) + \sum_{i=1}^{n} \log p(F_i = f_i|C = c) \tag{2.19}$$

#### 2.2.4.1 Berechnung der A-priori-Wahrscheinlichkeit

Die Wahrscheinlichkeit, dass ein Feature eintritt, wenn die Klasse bekannt ist, lässt sich durch die Division der Anzahl der Features in der Klasse durch die Anzahl der Features in allen Klassen gesamt berechnen.

$$p(F_i = f_i|C = c) = \left(\frac{f_{iC}}{f_{itotal}}\right)^{f_i} \text{[Hal11]}. \tag{2.20}$$

Da Wörter mehrmals in einem Text vorkommen können, werden die Wahrscheinlichkeiten mit der Anzahl der Vorkommen des Wortes ($f_i$) potenziert.

#### 2.2.4.2 Additive Smoothing

Da ein Wert von $f_{iC} = 0$ das gesamte Produkt $0$ werden lässt und somit falsche Ergebnisse liefert, wird mittels Additive Smoothing (auch *Laplace Smoothing* genannt) geglättet, um die Klassifizierung durchzuführen. Somit wird der Ausdruck in den Klammern durch folgenden Term ersetzt:

$$\frac{f_{iC} + \alpha}{f_{itotal} + k\ \alpha} \tag{2.21}$$

$k$ entspricht dabei der Anzahl der Klassen, $\alpha$ ist der Smoothing Operator, mit dem die Stärke der Glättung eingestellt werden kann. Typischerweise wird für $\alpha$ ein Wert von $1$ oder kleiner gewählt. Schlussendlich ergibt sich für die Klassifizierung durch Naive Bayes folgende Formel:

$$\text{classify}(f_1, \ldots, f_n) = \underset{C}{\text{argmax}}\ \log p(C = c) + \sum_{i=1}^{n} f_i \log \frac{f_{iC} + \alpha}{f_{itotal} + k\ \alpha} \tag{2.22}$$

Nachdem die Klassifikationsmethoden vorgestellt wurden, wird nun auf Gütemaße eingegangen, mit denen die Qualität der Algorithmen beurteilt werden kann.

### 2.2.5 Evaluierungsmaße für die Klassifikation

Klassifikatoren unterscheiden sich in Bezug auf die Komplexität deutlich, weshalb verschiedene Klassifikatoren auch unterschiedliche Resultate liefern. Um diese untereinander zu vergleichen, sind Gütemaße erforderlich. Aufgrund der Qualität des Klassifikators können außerdem auch Rückschlüsse auf das Trainingsset geliefert werden: Kleine Trainingssets mit Daten unzureichender Qualität liefern schlechtere Resultate. Insofern ist es auch sinnvoll, die Qualitätsmaße in den Aufbau des Trainingssets einfließen zu lassen um zu sehen, ob das Trainingsset schon ausreichend ausgebaut bzw. die Qualität davon zufriedenstellend ist.
Die Qualität der Algorithmen wird mittels Konfusionsmatrizen bewertet. In diesen Matrizen wird pro Testklasse die erkannten Klassen in den Spalten eingetragen. Auf der Diagonale liegen die korrekt erkannten Werte, im Idealfall lägen somit alle Werte auf dieser und alle anderen Felder wären $0$.
Konfusionsmatrizen bieten eine übersichtliche Darstellung der Ergebnisse eines Klassifikators. Zunächst wird ein Testset mit bekannter Klassenzugehörigkeit der Elemente benötigt, welches dem Klassifkationsalgorithmus unterzogen wird. Jede Spalte der Konfusionsmatrix entspricht der tatsächlichen Klasse, jede Zeile entspricht der Klasse, die der Algorithmus für das jeweilige Element detektiert. Die Ergebnisse der Klassifizierung werden dementsprechend in den jeweiligen Zellen der Matrix eingetragen. Aus der Konfusionsmatrix lassen sich aussagekräftige Gütemaße über die Qualität des Klassifikators berechnen, die Overall Accuracy, Producer's Accuracy, User's Accuracy sowie der Kappa-Koeffizient.

#### 2.2.5.1 Overall Accuracy

Wie bereits erwähnt, liegen alle korrekt klassifizierten Werte auf der Hauptdiagonale. Die Overall Accuracy berechnet sich durch die Summe an korrekt klassifizierten Daten dividiert durch die

| | Hand | Fuß | Gesicht | Bauch | Knie | Faust | Ellenbogen | Ohr |
|---|---|---|---|---|---|---|---|---|
| Hand | 34 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| Fuß | 1 | 22 | 1 | 1 | 1 | 3 | 1 | 0 |
| Gesicht | 2 | 2 | 19 | 1 | 0 | 2 | 1 | 0 |
| Bauch | 0 | 2 | 1 | 17 | 3 | 5 | 1 | 1 |
| Knie | 0 | 0 | 0 | 1 | 32 | 8 | 1 | 2 |
| Faust | 3 | 2 | 2 | 1 | 5 | 31 | 1 | 0 |
| Ellenbogen | 4 | 0 | 2 | 1 | 4 | 6 | 14 | 1 |
| Ohr | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 12 |

Tabelle 2.1: Ein Beispiel einer Konfusionsmatrix. In den Spalten befinden sich die Referenzklassen, in den Zeilen die detektierten Klassen.

Anzahl aller Daten im Testset (siehe Gleichung 2.23). Dieser Wert sagt allerdings keine klassenspezifischen Details aus, da unabhängig von der Klasse alle Werte miteinbezogen werden. Das Ergebnis kann durch Klassen mit einer hohen Anzahl an Elementen das Ergebnis ziemlich beeinflussen, da diese viel stärker ins Gewicht fallen.

$$OA = \frac{\sum_{k=1}^{q} n_{kk}}{n} \cdot 100 \tag{2.23}$$

#### 2.2.5.2 Producer's Accuracy

Die Producer's Accuracy erfasst hingegen ein klassenabhängiges Qualitätskriterium. Hier wird die Berechnung spaltenweise durchgeführt. Sie sagt also aus, wie viele der Werte einer bekannten Klasse tatsächlich dann als solche erkannt werden und entspricht somit der Zuverlässigkeit „aus der Sicht des Produzenten".

$$PA_i = \frac{\sum_{k=1}^{q} n_{ki}}{n_i} \cdot 100 \tag{2.24}$$

Die Berechnung erfolgt ähnlich zur Overall Accuracy, nur dass spaltenweise gegliedert wird. $n_i$ entspricht dabei der Anzahl in Spalte $i$ (siehe Gleichung 2.24). Wenn die Producer's Accuracy einer bestimmten Klasse also niedrig ist, kann das darauf deuten, dass möglicherweise das Trainingsset in dieser Klasse nicht ausreichend ist.

#### 2.2.5.3 User's Accuracy

Die User's Accuracy hingegen berechnet die „Reinheit" des Resultats, also wie viele der detektierten Klassen auch tatsächlich Mitglied dieser Klasse sind. Hierfür wird zeilenweise dividiert. Bei der User's Accuracy wird also die Güte „aus der Sicht des Benutzers bzw. Abnehmers" betrachtet. Die Berechnung erfolgt wie die Producer's Accuracy, nur dass zeilenweise gruppiert wird, $n_i$ entspricht nun der Anzahl in Zeile $i$ (siehe Gleichung 2.25).

$$UA_i = \frac{\sum_{k=1}^{q} n_{ik}}{n_i} \cdot 100 \tag{2.25}$$

#### 2.2.5.4 Kappa-Koeffizient

Der Kappa-Koeffizient wurde 1960 von Cohen veröffentlicht und ist ein Gütemaß eines Klassifikationsalgorithmus. Die Werte des Koeffizienten schwanken zwischen $-1$ und $+1$, wobei $+1$ einer totalen Übereinstimmung entspricht. Bei einem Kappa von 0 spricht man von einer Übereinstimmung, die dem Zufall entspricht. Werte kleiner als 0 deuten auf eine Übereinstimmung hin, die noch kleiner ist als eine zufällige Übereinstimmung. Anhand des Kappa-Wertes lässt sich also einschätzen, wie gut der Klassifikator funktioniert. Werte größer als 0,6 deuten auf eine starke, Werte größer als 0,8 auf eine fast vollständige Übereinstimmung hin. Eine mittelmäßige Übereinstimmung ist bereits bei einem Kappa von 0,4 erreicht [Alt90] [GBZL07]. Er lässt sich durch folgende Formel berechnen:

$$\kappa = \frac{n \sum_{i=1}^{r} x_{ii} - \sum_{i=1}^{r} (x_{i+} \cdot x_{+i})}{n^2 - \sum_{i=1}^{r} (x_{i+} \cdot x_{+i})} \tag{2.26}$$

$n$ entspricht der Gesamtanzahl, $r$ die Anzahl der Zeilen, $x_{ii}$ die Anzahl in Zeile $i$ und Spalte $i$ (also der $i$-te Wert der Hauptdiagonale), $x_{i+}$ die Anzahl in Zeile $i$ und $x_{+i}$ die Anzahl in Spalte $i$. Nachdem nun verschiedene Featureextraktionsverfahren und Klassifikatoren samt deren Gütemaßen erläutert wurden, werden abschließend in diesem Kapitel verwandte Arbeiten vorgestellt.

## 2.3 Verwandte Arbeiten

In diesem Abschnitt wird auf Basis von bereits bestehenden Implementierungen und Forschungsarbeiten argumentiert, warum gerade diese Methoden für die Umsetzung des Programms interessant sind und verwendet werden. Insbesondere wird auch auf die Kombination der verwendeten Featureextraktions- und Klassifikationsmethoden eingegangen. Dementsprechend wird dieser Abschnitt anhand der jeweiligen Methoden pro Demoanwendung beschrieben, beginnend mit der Spracherkennung und gefolgt von der Körperteilerkennung und dem Gesichtsvergleich mittels Template Matching.

### 2.3.1 Spracherkennung mittels MFCC und Dynamic Time Warping

In den letzten Jahren haben sich bei der Audiosignal-Featureextraktion die MFCC durchgesetzt [WC05] [PK08]. Sie wurden vom European Telecommunications Standards Institute (ETSI) im Jahr 2003 standardisiert. Auch Jang [Jan] spricht bei MFCC von den meist verwendeten Audio-Features. Die MFCC werden allerdings nicht nur für die Spracherkennung eingesetzt, sondern auch für die Sprechererkennung und Musikgenre-Klassifikation. Laut [ZWLC00] reichen die ersten zwölf Koeffizienten aus, um Informationen über den Inhalt des Gesprochenen zu erhalten, für die Sprechererkennung reichen die Koeffizienten 2 bis 16. Der nullte und erste Koeffizient enthalten keine sprecherspezifische Information.

Während der 1980er-Jahre wurde der Fokus weg von den templatebasierten Ansätzen bei der Spracherkennung hin zu statistischen Ansätzen wie den Hidden Markov Models (HMM) und neuronalen Netzen gelenkt [LLT02]. Es wird argumentiert, dass die Klassifikation mittels DTW keine Klassifikation von kontinuierlicher Sprache erlaubt. Außerdem steigt der Berechnungsaufwand, wenn ein großes Vokabular abgedeckt werden muss, da die Features der Testaufnahme mit allen Features des Trainingssets verglichen werden müssen. Da es sich in unserem Fall allerdings um ein beschränktes Vokabular handelt, ist der Rechenaufwand nicht zu hoch. Für ein Modell, das auch unbekannte Wörter abbildet, ist es erforderlich, dass nicht nur Wörter ins Vokabular aufgenommen werden, sondern Silben bzw. Lautfolgen, aus denen sich Wörter zusammensetzen lassen. Da in dieser Arbeit allerdings die Erkennung von isolierten Wörtern vorgesehen ist, hat DTW als Klassifikator durchaus seine Berechtigung. Als Features können beispielsweise LPC- oder MFCC-Features dienen. Das Argument mit der Sprecherunabhängigkeit ist ebenfalls nicht greifend, da beim Aufbau des Trainingssets unterschiedliche Sprecher verwendet werden, und als Features außerdem MFCC verwendet werden, die eine gewisse Sprecherunabhängigkeit garantieren.
Gawali et al. [GGYM10] vergleichen für die Erkennung von isolierten Wörtern in *Marathi*[1] MFCC mit DTW. Zunächst werden die ersten 13 MFCC-Features der Aufnahme extrahiert und anschließend mittels mit einem Distanzmaß (z. B. euklidischer Abstand) verglichen. Bei der Implementierung mittels DTW werden nicht die MFCC-Features verglichen, sondern das Sprachsignal verwendet. Die Erkennungsgenauigkeit liegt bei MFCC bei 94,65 Prozent und bei DTW bei 73,25 Prozent.
Bala et al. [BKB10] hingegen vergleichen die MFCC-Features mittels DTW. Hier wird argumentiert, dass durch den die unterschiedliche zeitliche Dauer zweier Audiosignale die Performance eines solchen Featurevergleichs bessere Resultate liefert. Muda et al. [MBE10] verwenden MFCC als Featureextraktionsmethode und DTW als Feature-Matching-Methode für die Stimmerkennung. Die Implementierung ist in der Lage, den Sprecher anhand seiner Stimme zu identifizieren.

### 2.3.2 Kombination von SIFT-Features mit dem Bag-of-Features-Verfahren

SIFT wurde von Lowe 1999 veröffentlicht[Low99], das Patent auf diesen Algorithmus hält die University of British Columbia. Allerdings existieren unterschiedliche Implementierungen auf der Basis der Version von Lowe. Der große Vorteil von SIFT ist die Rotations- und Skalierungsinvarianz, welche eine größere Toleranz bezüglich der Objekte in den Bildern ermöglicht. Außerdem ist der Algorithmus nicht sehr empfindlich auf perspektivische Verzerrungen, Rauschen und Beleuchtungsänderungen, wodurch ein sehr flexibler Einsatz möglich ist. Durch die Schnelligkeit des Algorithmus wird dieses Verfahren in der Praxis oft verwendet. Ein beliebter Anwendungsbereich ist das Erstellen von Panoramabildern aus mehreren Einzelbildern [JG09] [YG08]. Auch die Objekt- und Szenerieerkennung in Bildern ist ein häufiges Anwendungsgebiet von SIFT. Lowe [Low04] schlägt die Objekterkennung als primäres Anwendungsgebiet vor, auch unter Beeinträchtigungen wie gegenseitiger Verdeckung. Die Objekterkennung weist durchaus viele Parallelen zur Körperteilerkennung auf. Auch hier muss anhand dem äußeren Er-

[1] Marathi ist eine indoarische Sprache, die hauptsächlich im indischen Bundesstaat Maharashtra gesprochen wird.

scheinungsbild und der Form eine Entscheidung über die Klassenzugehörigkeit getroffen werden, weshalb sie als Anhaltspunkt bei der Literaturrecherche und der Suche nach geeigneten Methoden dient.
Juan [JG09] vergleicht die Eigenschaften von Featureextraktionsverfahren wie SIFT, PCA-SIFT und SURF und kommt zu dem Ergebnis, dass die Verwendung der verschiedenen Methoden abhängig ist vom Einsatzgebiet. In seiner Studie zeichnet sich SIFT durch die beste Skalierungs- und Rotationsinvarianz aus, Nachteil ist die lange Berechnungsdauer des Algorithmus. Der Vorteil von SURF ist die schnellere Berechnung, da ein Mittelwertfilter anstelle eines Gauß-Filters bei der Berechnung verwendet wird.
Die Kombination von SIFT und der Bag-of-Features-Methode bietet viele Vorteile und ist deshalb auch sehr beliebt. Laut [JNY07] liefert Bag of Features überraschend gute Resultate. Die Kombination der Bag-of-Features-Methode mit einem SIFT-Keypoint-Detektor ist eine weit verbreitete Praxis. Wie bereits erwähnt, hat der Verlust der räumlichen Information keine Auswirkung auf die Qualität der Ergebnisse. Als Klassifikationsmethode wird die Bag-of-Features-Methode häufig mit dem kNN-Algorithmus, Support Vector Machines (SVM) oder eben auch dem Naive-Bayes-Klassifikator kombiniert. Beim Naive-Bayes-Klassifikator wirkt sich die angenommene Naivität nicht auf die Qualität der Ergebnisse aus [CDF+04]. Wallraven und Caputo [WCG03] kombinieren für die Objekterkennung lokale Features, die mit einer SVM klassifiziert werden. Die SVM war im Vergleich zu anderen Möglichkeiten performanter.
Der kNN-Algorithmus ist eine sehr beliebte Klassifikationsmethode, da er einfach zu implementieren ist. Chen und Hauptmann [CH09] verwenden für die Erkennung von menschlichen Gesten in Überwachungsvideos Features, die auf SIFT aufbauen. Zusätzlich zu den SIFT-Features wird auch die lokale Bewegung der Features abgebildet, die Implementierung wird MoSIFT genannt. Die Rate der korrekten Erkennung dieser Implementierung liegt bei 95,8 Prozent am KTH-Human-Action-Datenset.[2]

### 2.3.3 Template Matching

Template Matching ist ein sehr einfaches Verfahren für Vergleichszwecke, dessen Berechnungsaufwand allerdings sehr hoch ist. Eine in der Praxis verwendete Implementierung von Template Matching ist die Schrifterkennung. So sind Buchstaben nicht immer durchgängig verbunden, beim *i* ist z. B. der Punkt vom Stamm getrennt. Mu et al. [MAHW01] verwenden Template Matching beispielsweise als Basis für die Gesichtserkennung und erreicht bei einem Trainingsset von 600 Personen eine Erkennungsrate von über 99 Prozent. Smita et al. [SVS11] verwenden für die Gesichtserkennung eine Kombination von der Farbinformation mit Template Matching, welche bessere Resulate liefert als ein ausschließlicher Hautfarbendetektor, da anhand der Form die Bereiche, die keine Gesichter beinhalten, eliminiert werden. Andere Arbeiten, wie z. B. Jin [JLYS07] benutzen zusätzlich auch die Farbinformation der Haut um die Gesichtsregionen einzugrenzen. Bei der Gesichtserkennung müssen allerdings die relevanten Gesichtsbereiche (Nase, Augen, Mund, Gesichtsbereich allgemein) als Templates verwendet werden. Stengele [Ste93] benutzt für kartografische Mustererkennung Template Matching.

[2] Nähere Informationen zum Human-Action-Datenset der KTH (Kungliga Tekniska högskolan/Königlich Technische Hochschule Stockholm) unter `http://www.nada.kth.se/cvap/actions/` (zuletzt abgerufen am 24.06.2012)

Die in diesem Kapitel vorgestellten Methoden dienen als Basis für die Implementierung der Problemstellung. Aufgrund der Literaturrecherche erwiesen sich die oben genannten Featureextraktions- und Klassifikationsmethoden als gut geeignet bzw. als ausreichend, worauf u. a. auch bei der vorangegangenen Vorstellung der verwandten Arbeiten eingegangen wurde. Nachdem der theoretische Hintergrund bekannt ist, können im folgenden Abschnitt nun die Aufgabenstellung und die Umsetzung der beschriebenen Methoden erklärt werden.

KAPITEL 3

# Aufgabenstellung und Lösung

Die in den vorhergehenden Abschnitten beschriebenen Features und Klassifikationsmethoden wurden für die Implementierung der Demoanwendungen verwendet. Aufbauend auf den ermittelten theoretischen Erkenntnissen wurde die Problemstellung praktisch gelöst, worauf in diesem Kapitel näher eingegangen wird. Zunächst wird in Abschnitt 3.1 argumentiert, weshalb für die Implementierung MATLAB als Entwicklungsumgebung eine gute Wahl ist. Anschließend werden die drei Demoanwendungen in Unterabschnitten beschrieben, welche jeweils in die Aufgabenstellung und Lösung dieser unterteilt sind. Dabei werden verwendete Bibliotheken und sonstige erwähnenswerte Details näher erklärt. Da Geräte, auf denen die Demoanwendungen verwendet werden, mit Windows betrieben werden, ist Plattformunabhängigkeit nicht erforderlich.

## 3.1 Wahl der Arbeitsumgebung

Für die Entwicklung von Anwendungen des Medienverstehens eignet sich insbesondere die von MathWorks vertriebene Software MATLAB. Die Verwendung dieser Arbeitsumgebung war in den Anforderungen vorgegeben. Im Folgenden wird erklärt, warum gerade MATLAB für die Implementierung gut geeignet ist. Im Vergleich zu herkömmlichen Programmiersprachen wie Java oder C++ ist sie spezialisiert auf die Auswertung, Verarbeitung und Visualisierung von numerischen Daten in der Form von Matrizen oder Vektoren. Diese Matrizen sind auch zur Laufzeit beliebig erweiterbar und müssen somit nicht dimensioniert werden. Beliebte Anwendungsgebiete von MATLAB finden sich in der Mustererkennung und Signalverarbeitung. In MATLAB besteht des Weiteren die Möglichkeit, Daten schnell grafisch auszugeben. Die Entwicklung von Benutzeroberflächen ist mittels GUIDE möglich. Die auszuführenden Befehle können in MATLAB direkt in der Konsole eingegeben werden, ohne dass eine Kompilierung erforderlich ist. Dies hat den Vorteil, dass Befehle schnell ausprobiert werden können. Zusätzlich können mehrere Befehle als Befehlsfolge abgespeichert werden. Ein weiterer Vorteil ist die Kompaktheit des Codes. Durch die Mächtigkeit der Befehle können beispielsweise alle Elemente einer Matrix

in einem Schritt manipuliert werden. Der Programmcode bleibt somit übersichtlich. Komplexe Teile des Programms, deren Umsetzung in MATLAB z. B. zu langsam wäre, können in C++ implementiert und mittels MEX-Dateien eingebunden werden.
Für die Programmierung existieren eine Vielzahl von Erweiterungen, welche Toolboxen genannt werden. Die für die Implementierung dieser Demos erforderlichen Erweiterungen waren die *Image Acquisition Toolbox*, um auf die Webcam zugreifen zu können, sowie die *Signal Processing Toolbox*, die die Audiosignalverarbeitung ermöglicht. Die erstgenannte Toolbox ist nur für Windows verfügbar, die Signal Processing Toolbox ist plattformunabhängig. Da die Anforderung des Betreuers allerdings nur eine Verwendung des Programms auf Windows verlangt, stellt die mangelnde Plattformunabhängigkeit keine Einschränkung dar.
Aufgrund dieser Eigenschaften ist MATLAB am besten für die Implementierung geeignet. Mögliche Alternativen wie C++ mit der OpenCV-Bibliothek oder Java sind nicht sinnvoll, da alle drei Demoanwendungen in einer Programmiersprache implementiert werden sollen. OpenCV ist allerdings nur für Bildverarbeitung vorgesehen. MATLAB hat den Vorteil, dass es zusätzlich zur Bildverarbeitung auch andere Medien wie z. B. Audiosignale importieren, exportieren und verarbeiten kann [Eid11]. Auf Basis dieser Überlegungen werden im Folgenden nun die Details der Implementierung der drei Demoanwendungen beschrieben, beginnend mit der Ähnlichkeitsberechnung zu prominenten Personen, gefolgt von der Körperteilerkennung und der Sprachsignalerkennung.

## 3.2 Ähnlichkeitsberechnung zu prominenten Personen

Zunächst wird in Abschnitt 3.2.1 auf die Aufgabenstellung eingegangen. Hier wird spezifiziert, welche Anforderungen erfüllt werden müssen. Ein Anwendungsfalldiagramm zeigt die Abhängigkeiten zwischen den Anwendungsfällen und dem Benutzer. Der Ablauf des Programms wird in einem Flussdiagramm visualisiert. Im darauf aufbauenden Abschnitt 3.2.2 wird die Umsetzung der Problemstellung beschrieben.

### 3.2.1 Aufgabenstellung

Wie die anderen beiden Demoanwendungen auch, war in dieser Applikation ein spielerischer Charakter verlangt. Ziel ist es anhand eines Fotos des Benutzers den ihm am ähnlichsten sehenden Prominenten zu ermitteln. Der Benutzer soll mit wenigen Interaktionsschritten zum gewünschten Ziel erlangen, weshalb die Interaktionsschritte auf die wesentlichen Funktionen reduziert sind. Der Benutzer hat also nur die Möglichkeit, ein Foto von sich aufzunehmen, welches anschließend vom Computer analysiert wird. Andere Interaktionsschritte sind im Programm nicht vorgesehen. Wichtig ist, dass das Resultat rasch ermittelt wird und für den Benutzer eine Nachvollziehbarkeit der Ähnlichkeit gegeben ist. Beim Resultat wird deshalb der Ausschnitt des Benutzergesichts auf die korrespondierende Stelle des Prominenten gelegt. Durch eine Überblendung wird das Gesicht des ähnlichsten Prominenten ersichtlich. Abbildung 3.1 zeigt das Anwendungsfalldiagramm dieser Demoanwendung. In Abbildung 3.2 sind die Arbeitsschritte der Ähnlichkeitsberechnung dargestellt. Zunächst muss aus der Aufnahme das Gesicht des Benutzers ermittelt werden, welches anschließend mit den Gesichtern der Prominenten verglichen

wird. Das Gesicht, das am besten mit dem Benutzer übereinstimmt, wird ausgegeben. Wie die Problemstellung umgesetzt wurde, ist im Detail im nachfolgenden Unterabschnitt zu lesen. Hier wird auch die praktische Umsetzung der Verfahren aus Kapitel 2 beschrieben.

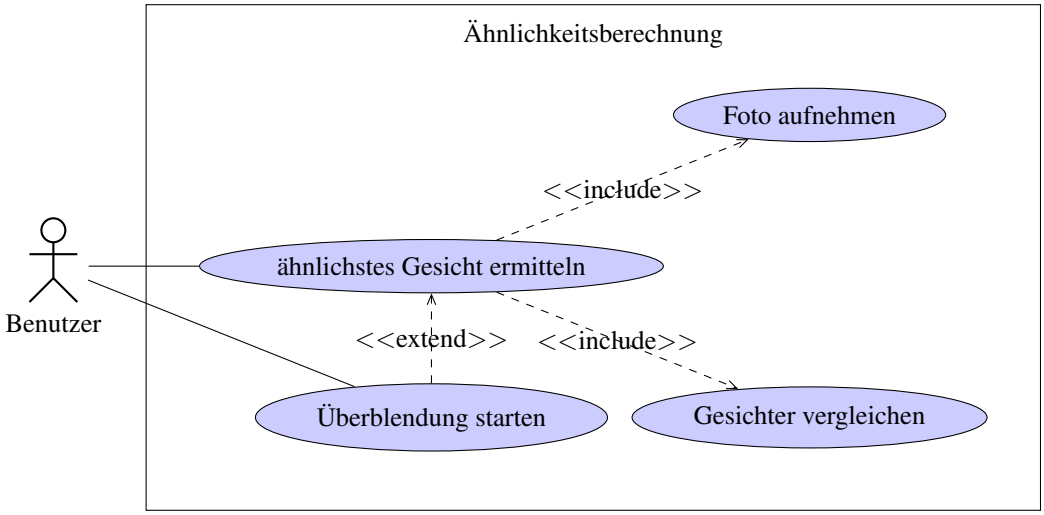



Abbildung 3.1: Anwendungsfalldiagramm der Ähnlichkeitsberechnung.

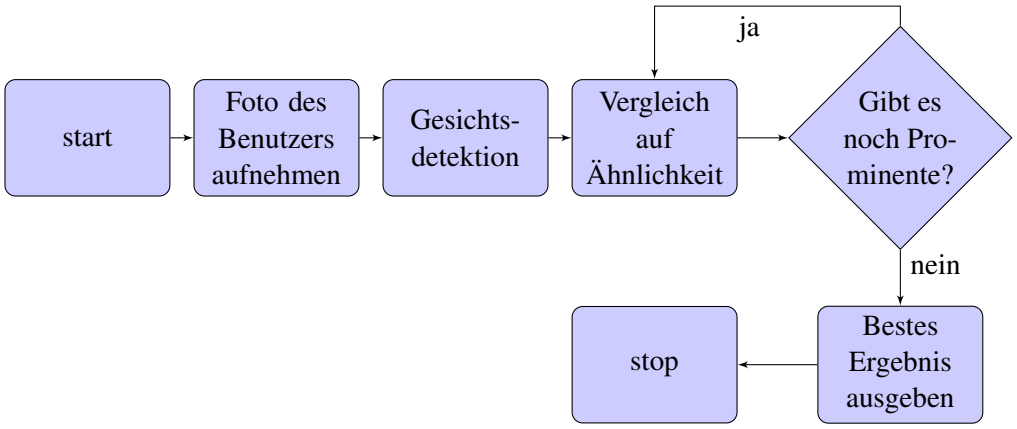



Abbildung 3.2: Ablauf bei der Ähnlichkeitsberechnung.

### 3.2.2 Lösung

Für die oben beschriebene Aufgabenstellung erwies sich Template Matching als die beste Möglichkeit der Ähnlichkeitsberechnung. Als Template dient der Gesichtsausschnitt des Benutzerfotos, welches mit den Bildern der Prominenten verglichen wird. Die Ähnlichkeit wird durch den geringsten Distanzwert zwischen Benutzergesicht und den Prominentengesichtern ermittelt.

Bevor allerdings das Template Matching durchgeführt wird, muss mittels Gesichtsdetektion die ungefähre Position des Gesichts sowohl im Benutzerbild als auch in den Prominentenbildern festgestellt werden, um das Template zu erstellen bzw. eine höhere Zuverlässigkeit zu erreichen.

#### 3.2.2.1 Gesichtsdetektion als Vorarbeit für das Template Matching

Die Verwendung eines Gesichtserkennungsalgorithmus hat wesentliche Vorteile: Einerseits gestaltet sich die Suche des Templates in den gesamten Prominentenbildern als sehr rechenintensiv und zu langsam für den Praxisgebrauch, weshalb zunächst mittels eines Gesichtserkennungsalgorithmus die Suchregion eingeschränkt werden kann. Andererseits erhöht sich somit auch die Zuverlässigkeit, denn so werden nur Regionen, in denen tatsächlich ein Gesicht vorkommt, für die Ähnlichkeitsberechnung in Betracht gezogen. Zusätzlich wird die Gesichtserkennung auch auf die Aufnahme des Benutzers angewendet, um das Template zu erstellen. Alternativen zur automatisierten Gesichtserkennung wäre eine manuelle Markierung des Benutzergesichtes in der Aufnahme oder ein vorgegebener Rahmen, in dem der Benutzer sein Gesicht formatfüllend halten solle. Beide Alternativen stehen allerdings im Widerspruch zur Vorgabe, die Interaktion einfach zu halten, weshalb diese zwei Methoden nicht verwendet werden.
Für die Gesichtsdetektion fiel die Wahl auf den von Mikael Nilsson implementierte Funktion [NNC07], da dieser schnell und zuverlässig funktioniert.[1] Er wird über folgende Funktion aufgerufen:

```
[output,count,m,svec] = facefind(x,minf,maxf,dx,dy,sens)
```

Der Funktion `facefind` wird ein Graustufenbild `x` übergeben, die anderen Parameter sind optional. Mittels `minf` und `maxf` kann die Gesichtsgröße bestimmt werden, `dx` und `dy` bestimmen die Schrittweite im Bild in x- und y-Richtung, mittels `sens` wird die Empfindlichkeit eingestellt. Für weitere – hier nicht genutzte – Konfigurationsmöglichkeiten dieser Funktion sei auf Nilsson [NNC07] hingewiesen. Zurückgegeben werden bei der Funktion die Position des Gesichtes, als auch die Anzahl der Gesichter, die im angegebenen Bild vorkommen. Diese Funktion verwendet lokale SMQT-Features und einem Split-up-SNoW-Klassifikator [NNC07]. Um die Detektion der Prominentengesichter nur einmal durchzuführen, werden deren Koordinaten nach der erstmaligen Detektion gesichert, wodurch eine weitere Performancesteigerung des Programms erreicht wird. Der Suchbereich in den Prominentenbildern wird mit einer zehnprozentigen Vergrößerung erweitert, da der Gesichtserkennungsalgorithmus nicht immer genau denselben Ausschnitt zurückgibt. Sollten sich bei der Aufnahme des Benutzerbildes mehrere Personen vor der Kamera befinden, fällt die Wahl auf das größte Gesicht, das im Bild vorkommt. Dieses entspricht dem Gesicht mit dem niedrigsten Index. Bei den Prominentenbildern ist eine Detektion mehrerer Gesichter nicht möglich, da diese anhand von Qualitätskriterien ausgesucht wurden. Zu diesen gehören u. a. eine frontale Ausrichtung des Gesichts und dass keine anderen Personen auf den Fotos vorhanden sind. Ein weiterer Vorteil der zusätzlichen Gesichtsdetektion ist die Reduzierung der Fehlerquote: Ohne Einschränkung des Suchbereichs kann das beste Resultat auch außerhalb des Prominentengesichts vorkommen. Diese Tatsache tritt beispielsweise

[1] Die Funktion ist unter `http://www.mathworks.com/matlabcentral/fileexchange/authors/26851` erhältlich (zuletzt abgerufen am 23.09.2012)

dann auf, wenn die Randbereiche des Benutzerbildes dunkel und Bereiche im Hintergrund der Prominentenaufnahmen ebenfalls dunkel sind. Somit ist die berechnete Distanz in diesen Bereichen am niedrigsten und stimmt nicht mit der Position der Gesichter überein. Der berechnete Distanzwert kann somit auch nicht für die Ermittlung des ähnlichsten Prominenten verwendet werden, da er keine Aussage über die Ähnlichkeit der Gesichter macht. Solche Fälle können durch die oben beschriebene Gesichtsdetektion vermieden werden, da die Suche auf den erweiterten Bereich eines Gesichts eingeschränkt wird und somit die Gesichter mit hoher Wahrscheinlichkeit übereinander liegen. Nachdem das Gesicht detektiert wurde, wird dieses in der Größe normalisiert, um den Arbeitsaufwand beim Template Matching zu reduzieren.

#### 3.2.2.2 Normalisierung der Größe des detektierten Gesichts

Das detektierte Gesicht auf eine vorgegebene Größe normalisiert, um den Vergleich mit den Prominentengesichtern zu beschleunigen. Template Matching ist nicht skalierungsinvariant und muss somit mit verschiedenen Größen des Templates durchgeführt werden (siehe Abschnitt 2.3.3). Durch die Normalisierung der Gesichtsgröße wird allerdings die Anzahl der erforderlichen Skalierungen eingeschränkt. Auch die Prominentengesichter werden in der Größe normalisiert, was ebenfalls die Anzahl der Vergleiche reduziert. Allerdings kann nicht vollständig auf den Vergleich mit verschiedenen Skalierungen verzichtet werden, da der Gesichtsdetektionsalgorithmus nicht immer denselben Ausschnitt eines Gesichtes liefert. Nachdem diese Optimierungen durchgeführt wurden, kann das Template Matching durchgeführt werden.

Abbildung 3.3: Das Template Matching wird mit zehn verschiedenen Größen des Templates durchgeführt. In dieser Abbildung ist das größte und das kleinste Template abgebildet.

#### 3.2.2.3 Durchführung des Template Matchings

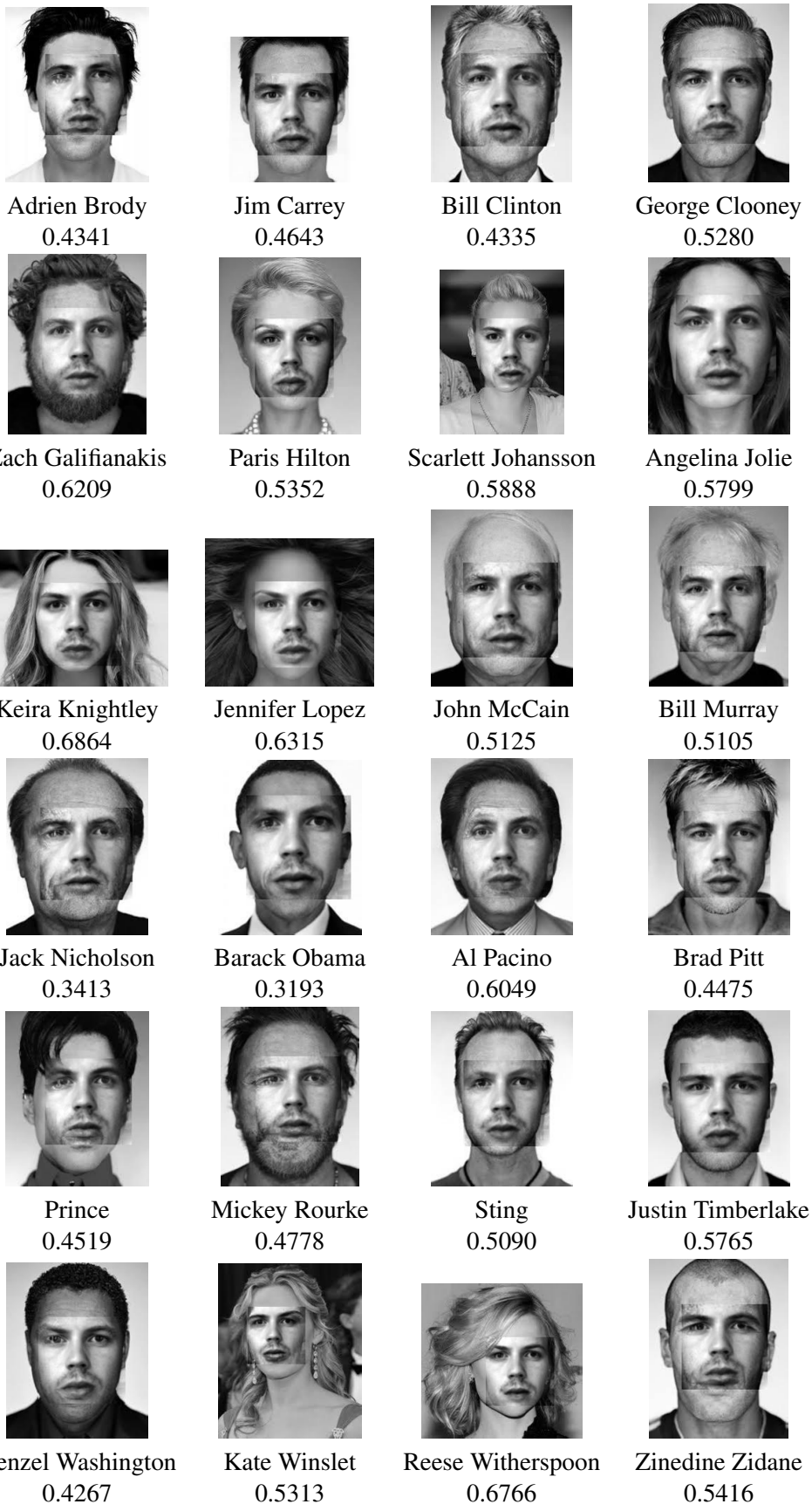
Das Template Matching wird mittels normalisierter Kreuzkorrelation berechnet, welche durch die MATLAB-Funktion

```
C = normxcorr2(template, A)
```

zur Verfügung gestellt wird. Mit dieser Funktion ist es möglich die Kreuzkorrelation zweier zweidimensionaler Signale zu berechnen, wie bereits in Abschnitt 2.1.3.5 beschrieben. Je nach Größe der Bilder wird die Kreuzkorrelationsberechnung entweder im Zeit- oder Bildbereich

durchgeführt.[2] Durch die Verwendung dieser Funktion werden Schleifen vermieden, was sich in MATLAB positiv auf die Performance auswirkt.
Um brauchbare Resultate zu erhalten, muss die `template`-Matrix kleiner als die Matrix `A` sein. `normxcorr2` liefert Werte im Bereich von $-1$ bis $+1$ zurück. Der Punkt mit dem größten Wert entspricht der Koordinate, an der die Übereinstimmung des Templates mit dem anderen Bild am größten ist. Dieser Wert wird für jedes Prominentenbild ermittelt und anschließend verglichen. Der größte Wert entspricht der größten Übereinstimmung.
Da Template Matching nicht skalierungsinvariant ist, muss das Template in verschiedenen Größen in den Bildern der Prominenten gesucht werden, wie bereits in Abschnitt 3.2.2.2 erwähnt. Die Rotationsabhängigkeit spielt bei der Implementierung keine besondere Rolle, da davon ausgegangen werden kann, dass vom Benutzer verlangt wird, Fotos seines Gesichts mit frontaler Ausrichtung zur Kamera aufzunehmen. Dieselbe Ausrichtung des Benutzergesichts zum Prominentengesicht ist für eine aussagekräftige Ähnlichkeitsberechnung erforderlich. Wie in Tabelle 3.1 ersichtlich, ist das jeweils beste Vergleichsresultat von der Proportion des Benutzergesichtes her meist korrekt. Sowohl die Augen als auch der Mund befinden sich an der richtigen Position. Des Weiteren ist der Übergang vom Kinn des Prominentengesichts zum Benutzergesicht ein weiteres Indiz für die korrekte Skalierung.
Bei der Darstellung der Resultate werden die Randbereiche des Templates abgeschwächt, sodass die Übergänge zwischen Benutzergesicht und Prominentenportrait nicht allzu stark sichtbar sind. Die Auswirkungen davon zeigt ebenfalls Tabelle 3.1. In dieser Abbildung sind die Endresultate samt den Korrelationswerten abgebildet, die vom Programm ausgegeben werden. Hier ist außerdem ersichtlich, dass die Benutzergesichter weitgehend korrekt skaliert werden (man vergleiche beispielsweise die Gesichtsgröße bei Scarlett Johannson [2. Reihe, 3. Spalte] und Angelina Jolie [2. Reihe, 4. Spalte]). Die beste Übereinstimmung ist zu Keira Knightley (3. Reihe, 1. Spalte) gegeben, da die Form der Augenbrauen und die Position der Augen und der Nase weitgehend übereinstimmt. Auch der Vergleich mit Reese Witherspoon (6. Reihe, 3. Spalte) zeigt ein gutes Ergebnis, welches sich durch die Positionsübereinstimmung der Augen und des Mundes begründen lässt. Die schlechtesten Resultate sind durch eine nicht exakte Skalierung und somit ungenaue Übereinanderlegung (siehe z. B. Barack Obama [4. Reihe, 2. Spalte] oder Adrien Brody [1. Reihe, 1. Spalte]) erklärbar. Die Qualität der Ergebnisse wird in Abschnitt 5.1 detaillierter beschrieben.

[2] Für nähere Informationen bezüglich der Funktion, siehe `http://www.mathworks.de/help/toolbox/images/ref/normxcorr2.html` (zuletzt abgerufen am 28.03.2012)



| | | | |
|---|---|---|---|
| Adrien Brody<br>0.4341 | Jim Carrey<br>0.4643 | Bill Clinton<br>0.4335 | George Clooney<br>0.5280 |
| Zach Galifianakis<br>0.6209 | Paris Hilton<br>0.5352 | Scarlett Johansson<br>0.5888 | Angelina Jolie<br>0.5799 |
| Keira Knightley<br>0.6864 | Jennifer Lopez<br>0.6315 | John McCain<br>0.5125 | Bill Murray<br>0.5105 |
| Jack Nicholson<br>0.3413 | Barack Obama<br>0.3193 | Al Pacino<br>0.6049 | Brad Pitt<br>0.4475 |
| Prince<br>0.4519 | Mickey Rourke<br>0.4778 | Sting<br>0.5090 | Justin Timberlake<br>0.5765 |
| Denzel Washington<br>0.4267 | Kate Winslet<br>0.5313 | Reese Witherspoon<br>0.6766 | Zinedine Zidane<br>0.5416 |

Tabelle 3.1: Die Zwischenschritte des Template Matchings.

## 3.3 Körperteilerkennung

Die zweite Demoanwendung behandelt das Problemfeld der Objekterkennung anhand der Körperteilerkennung. Wie auch der vorherige Abschnitt ist dieses Kapitel in die Aufgabenstellung und Lösung unterteilt.

### 3.3.1 Aufgabenstellung

In dieser Demoanwendung bekommt der Benutzer die Aufgabe, ein vorgegebenes Körperteil so schnell wie möglich in die Kamera zu halten. Je nach Reaktionszeit werden dementsprechend Punkte gutgeschrieben, sofern das richtige Körperteil erkannt wurde. Das dieses Spiel auf Zeit läuft, ist eine selbsterklärende Benutzeroberfläche und rasche Bedienungsmöglichkeit gewünscht. Anhand der erreichten Punkte wird eine Rangliste der besten Spieler erstellt. Dementsprechend soll auch die Punktevergabe gerecht sein, sodass eine Balance zwischen schneller Reaktionszeit und korrekter Lösung gefunden wird. Der Highscore wird nach jeder abgeschlossenen Runde angezeigt bzw. kann auch bei Verlangen des Benutzers eingeblendet werden. Abbildung 3.4 zeigt das Anwendungsfalldiagramm der Körperteilerkennung.

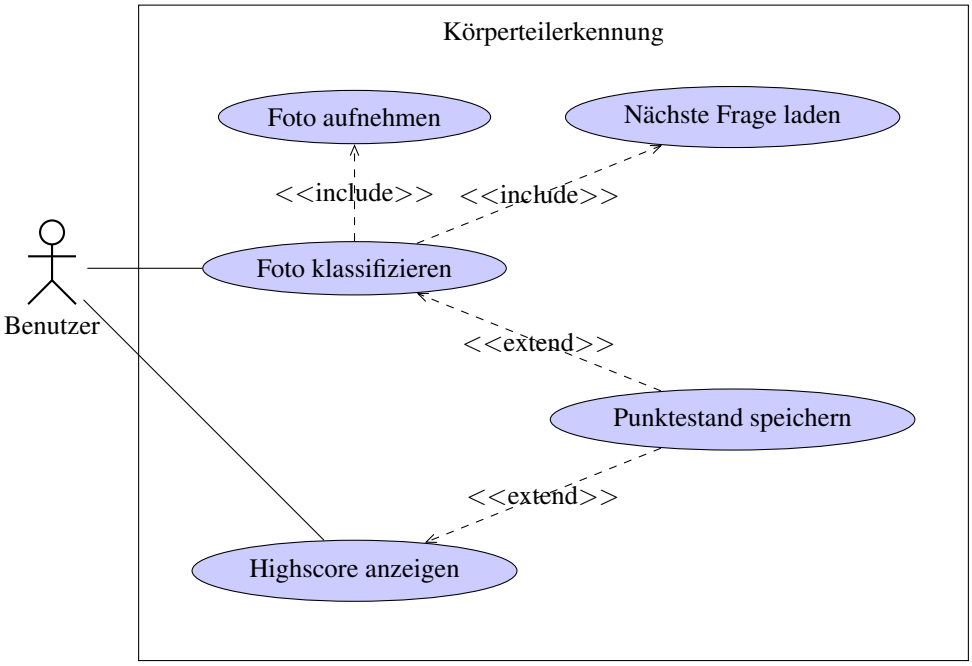



Abbildung 3.4: Anwendungsfalldiagramm der Körperteilerkennung.

Der schematische Ablauf der Körperteilerkennung ist in Abbildung 3.5 dargestellt. Wie die Problemstellung in MATLAB umgesetzt wurde, ist im nachfolgenden Kapitel nachzulesen.

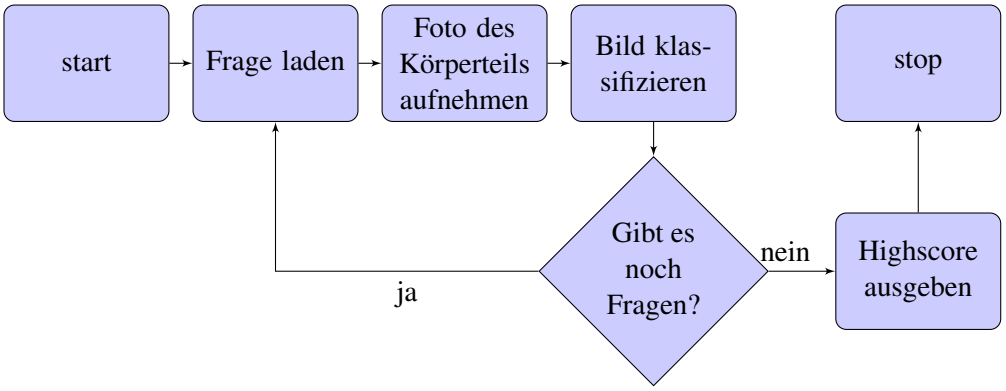



Abbildung 3.5: Ablauf bei der Körperteilerkennung.

### 3.3.2 Lösung

Für die Umsetzung der oben beschrieben Problemstellung wird eine Kombination des Keypoint-Detektors SIFT mit dem Bag-of-Features-Verfahren verwendet. Als Klassifikator dienen wahlweise der kNN-Algorithmus, das Naive-Bayes-Verfahren und eine SVM. Für die Keypoint-Detektion wurde als Basis die VLFeat-Bibliothek von Vedaldi et al. [VF08] herangezogen, welche viele Algorithmen aus dem Bereich der Computer Vision bereitstellt. So extistieren unter anderem der in dieser Arbeit vorgestellte SIFT samt dessen Abwandlungen DSIFT und PHOW, auch Clusteringverfahren wie der k-Means-Algorithmus sind implementiert. Die Bibliothek funktioniert auf den drei Plattformen Windows, Mac OS X und Linux [VF08].

VLFeat stellt außerdem Demoanwendungen der Bibliothek im Unterordner *VLROOT/apps* zur Verfügung. Für die Keypoint-Detektion wurde als Basis die Demonstrationsanwendung der Klassifikation des Caltech-101-Datensatzes herangezogen. Dieser Datensatz besteht aus 101 Objektkategorien, die aus jeweils 40 bis 800 Bildern bestehen [FFAR03]. Pro Klasse werden in diesem Demo 15 zufällig ausgewählte Bilder für den Aufbau des Trainingssets verwendet und 15 zufällig ausgewählte Bilder in das Testset aufgenommen, wobei die Anzahl der Bilder beliebig veränderbar ist. In dieser Konstellation liegt die Erkennungsrate bei 64 Prozent[3]. Für den Aufbau des Trainingssets werden die PHOW-Features jedes Bildes aus dem Trainingsset ermittelt, welche anschließend mittels k-Means-Verfahren geclustert werden. Wie bereits in Abschnitt 2.1.3.3 erwähnt, sind die PHOW-Features schneller zu berechnen als die SIFT-Features. Dieses Vokabular besteht aus beispielsweise 600 visuellen Wörtern (die Anzahl ist beliebig anpassbar, jedoch liefern 600 Wörter gute Ergebnisse), mit denen die Bilder des Trainingssets anhand von Histogrammen beschrieben werden. Anschließend wird die SVM trainiert. Nach der Trainingsphase erfolgt die Klassifizierung des Testdatensatzes. Auch hier werden Histogramme der Testbilder erstellt, die mittels der zuvor trainierten SVM klassifiziert werden. SVMs haben den Vorteil, dass auch nicht linear trennbare Daten durch den sogenannten Kernel-Trick klassifiziert werden können. Hierbei wird der Merkmalsraum in einen höherdimensionalen umgewandelt, in dem die

[3] Für weitere Informationen zur Demoanwendung siehe `http://www.vlfeat.org/applications/apps.html` (zuletzt abgerufen am 27.03.2012)

lineare Trennung möglich ist.
Diese Anwendung diente als erster Anhaltspunkt für die Körperteilerkennung. Zusätzlich zu dem bereits implementierten SVM-Klassifikators wurden auch die beiden in Abschnitt 2.2 beschriebenen Verfahren implementiert: der kNN-Algorithmus und der Naive-Bayes-Klassifikator. Die Ergebnisse der unterschiedlichen Klassifikationsalgorithmen können dadurch verglichen werden. Die ausgewählten Verfahren sind die gängigsten Klassifikationsverfahren, die in Kombination mit der Bag-of-Features-Methode verwendet werden. Die Ergebnisse der drei Algorithmen im Vergleich sind Abschnitt 5.2 zu entnehmen.
In den folgenden Unterabschnitten werden die einzelnen Arbeitsschritte detailliert erklärt. Zunächst wird der Aufbau des Trainingssets beschriebe, welches als Basis für eine zuverlässige Klassifizierung dient. Insbesondere ist die Auswahl der Bilder von großer Bedeutung, aus denen die Features extrahiert werden.

#### 3.3.2.1 Aufbau des Trainingssets

Der Aufbau des Trainingssets ist ein wichtiger Schritt bei der Umsetzung des Programms. Das Trainingsset ist maßgeblich für die Qualität der Klassifikationsergebnisse verantwortlich. Für die Körperteilerkennung müssen Fotos von den gewünschten Körperteilen aufgenommen werden, für die eine Klassifizierung vorgesehen ist. Als Klassen dienen die folgenden Körperteile: Fuß, Hand, Ellenbogen, Bauchnabel, Gesicht, Knie, Ohr und Faust. Da die SIFT-Features nicht spiegelungsinvariant sind, wird – sofern es zwei davon gibt – immer nur das linke Körperteil im Trainingsset abgebildet. Dementsprechend werden auch nur die linken Körperteile des Benutzers erkannt, weshalb dem Anwender kommuniziert werden muss, wie das Körperteil in die Kamera gehalten werden muss (siehe Abschnitt 4.2).
Somit ergeben sich für das Trainingsset acht Klassen zu je ca. 28 bis 124 Bildern (siehe Tabelle 3.2). Für eine zuverlässige Klassifikation ist es wichtig, dass das Trainingsset aus Bildern mit unterschiedlichen Lichtverhältnissen in unterschiedlichen Umgebungen besteht, sodass ein breites Spektrum abgebildet ist. Dadurch wird einer größere Erkennungsrate erreicht. In Abbildung 3.6 ist ein Auszug aus den Bildern der Faust aus dem Trainingsset abgebildet. In der ersten Version des Trainingssets wurden Aufnahmen der Webcam verwendet, bei denen die Körperteile nicht ausgeschnitten waren und dementsprechend viel Hintergrund sichtbar war. Die Größe der Bilder lag bei $640 \times 480$ Pixel. Obwohl die Bemühungen darauf abzielten, möglichst vielseitige Hintergründe zu verwenden, funktionierte die Klassifizierung nur zu ca. 60 Prozent korrekt. Diese Rate war für eine Verwendung in der Praxis nicht ausreichend. Ein weiteres Problem war, dass die Aufnahmen wegen der Entwicklung auf einem Stand-PC mit integrierter Webcam trotzdem immer ähnliche Hintergründe hatten. Der Monitor mit der Webcam konnte nur in einem begrenzten Umfang bewegt werden, weshalb sich Teile des Hintergrundes in mehreren Aufnahmen wiederfanden. Die immer wieder auftretenden ähnlichen Hintergründe beeinflussten die ins Trainingsset aufgenommenen Features. Dieses Problem konnte durch das Zuschneiden des Bildes auf die relevanten Bereiche und die zusätzliche Aufnahme von Bildern aus anderen Umgebungen signifikant verbessert werden. In Abbildung 3.6 ist exemplarisch ein Auszug des Trainingssets der Faust dargestellt. Wie man sehen kann, sind die Bilder des Trainingssets auf die Körperteilgröße angepasst, außerdem wird darauf geachtet, dass der Hintergrund auf den Bildern möglichst unterschiedlich ist, um sicherzugehen, dass dieser keinen Einfluss auf die Ergebnis-

| Klasse | Bilderanzahl |
|---|---|
| Ellenbogen | 28 |
| Faust | 34 |
| Fuß[4] | 124 |
| Gesicht | 44 |
| Hand | 33 |
| Knie | 28 |
| Nabel | 28 |
| Ohr | 28 |

Tabelle 3.2: Größe des Trainingsdatensets.

se hat. Für die Bilder eines linken Körperteils werden die Aufnahmen des rechten Äquivalents gespiegelt und ebenfalls in das Trainingsset aufgenommen, wodurch eine höhere Diversität der Hintergründe erreicht wird. Aus diesen Bildern werden die Features extrahiert und gespeichert, sodass eine rasche Wiederverwendung dieser möglich ist.

Abbildung 3.6: Auszug aus dem Trainingsset am Beispiel *Faust*. Es wurden verschiedene Belichtungen und Hintergründe verwendet um größtmögliche Flexibilität gegen unterschiedliche Benutzeraufnahmen zu erreichen.

[4]Durch die anfänglich schlechte Erkennungsrate der Füße wurde das Trainingsset ausgebaut und ist deshalb deutlich größer als das der anderen Klassen.

| Ellenbogen | Faust | Fuß | Gesicht |
|---|---|---|---|
| Hand | Knie | Nabel | Ohr |

Tabelle 3.3: Auszug aus dem Trainingsset (pro Klasse ein Bild). Die Körperteile des Benutzers müssen für die Klassifizierung in einer ähnlichen Position abgebildet werden.

**Berechnung der SIFT-Features** Zunächst müssen aus den Bildern des Trainingssets die SIFT-Features ermittelt werden, aus denen das Vokabular aufgebaut wird. Die VLFeat-Bibliothek[5] bietet drei verschiedene Verfahren an, um SIFT-Features zu berechnen. Die darin enthaltene Funktion

```
[F, D] = vl_sift(I)
```

berechnet die SIFT-Features `F` mit den dazugehörigen Deskriptoren `D` vom Bild `I`. Die Features enthalten die Koordinaten, die Skalierung sowie die Orientierung des Keypoints, die Deskriptoren sind die 128-dimensionalen Vektoren, die die Keypoints beschreiben. Die Funktion

```
[F, D] = vl_dsift(I)
```

berechnet die DSIFT-Deskriptoren. Diese Deskriptoren haben den Vorteil, dass sie schneller berechenbar sind. Auch hier werden die Frames mit den Koordinaten sowie die 128-dimensionalen Deskriptoren ermittelt. Die Funktion stellt keinen Gauß-Filter zur Verfügung sondern muss manuell durchgeführt werden. Als dritte Methode stellt VLFeat die Funktion

```
[F, D] = vl_phow(I)
```

zur Verfügung, welche einen Wrapper über die Funktionen `im_smooth` und `vl_dsift` bereit stellt. Die Merkmale werden in dieser Demoanwendung mittels `vl_phow` ermittelt, um den

[5] Für nähere Informationen siehe `http://www.vlfeat.org/mdoc/mdoc.html` (zuletzt abgerufen am 27.03.2012)

Vorteil der höheren Geschwindigkeit auszunützen. Aus den ermittelten Features des Trainingssets wird anschließend das Vokabular erstellt, mit dem die Klassen des Trainingssets beschrieben werden und mit dem die anschließende Klassifikation von Bildern unbekannter Körperteile möglich ist.

**Erstellung des Vokabulars** Die Erstellung eines guten Vokabulars ist sehr rechenaufwändig. Da allerdings das Trainingsset immer gleich bleibt und sich nur beim Hinzufügen oder Entfernen von Körperteilen ändert, braucht dieses nur einmal berechnet werden. Eine bei jedem Durchlauf benötigte Berechnung des Vokabulars wäre für den Praxisgebrauch zu zeitintensiv. Das Vokabular beschreibt den Wortschatz anhand dessen die Klassifizierung durchgeführt wird. Es ist auf 600 Wörter beschränkt. Dafür werden mittels k-Means-Clustering die SIFT-Deskriptoren des Trainingsdatensatzes in ähnliche Cluster zusammengefasst. Die Anzahl der Cluster wird durch die Anzahl der Wörter des Vokabulars bestimmt. Für das k-Means-Clustering gibt es in VLFeat eine Funktion namens

```
[C, A] = vl_kmeans(X, numcenters).
```

Dieser Algorithmus erlaubt es außerdem, optional bestimmte Zusatzparameter anzugeben, wie z. B. die Art des Algorithmus etc.
Anhand dieses Vokabulars werden räumliche Histogramme für die Bilder des Trainingssets erstellt. Der Vergleich zwischen den SIFT-Deskriptoren und dem Histogramm erfolgt mittels Vektorquantisierung. Hier wird der Merkmalsvektor dem Cluster zugeordnet, zu dem die Distanz des Deskriptors am geringsten ist. Nach der Berechnung des Modells kann die Klassifizierung durchgeführt werden.

#### 3.3.2.2 Klassifizierung

Die Klassifizierung wurde in der Funktion

```
[className, score] = classify(model, im, mode)
```

implementiert. Ihr wird das zuvor aufgebaute Modell des Trainingssets sowie das Bild des unbekannten Körperteils, das klassifiziert werden soll, übergeben. Der dritte Parameter `mode` bestimmt den verwendeten Algorithmus, als Übergabewerte sind `kNN`, `naiveBayes` sowie `SVM` möglich.
Für eine zuverlässige Klassifikation ist es erforderlich, die Körperteile soweit wie möglich freizustellen, sodass auf den Bildern wenig irrelevante Hintergrundinformation vorhanden ist. Aus diesem Grund wurde der Aufnahmebereich auf 200 Pixel Breite und 300 Pixel Höhe beschränkt. Mit einem hochformatigen Aufnahmefenster sind die gewünschten Körperteile gut abzubilden, da die meisten Körperteile in der Orientierung zur Webcam vertikal gestreckt sind. Um dem Benutzer die Interaktion zu erleichtern, wird der Bereich, der für die Klassifikation freigestellt wird, durch einen Rahmen hervorgehoben. Das verlangte Körperteil muss den Bereich so weit wie möglich ausfüllen.

Nach der betätigten Aufnahme werden die PHOW-Features berechnet. Wie auch schon beim Aufbau des Trainingssets wird auch hier die Funktion `vl_phow` der VLFeat-Bibliothek verwendet. Die Interest Points werden anhand von Histogrammen beschrieben. Die Klassifikation kann mit diesen Features durchgeführt werden.
Der kNN-Algorithmus vergleicht die Histogramme der Bilder. Als Maß für die Ähnlichkeit wird der euklidische Abstand verwendet. Die zehn ähnlichsten Bilder des Trainingssets bestimmen die Klasse des Bildes. Für den Naive-Bayes-Klassifikator dient als Ausgangslage die vorgestellte optimierte Formel der Naive-Bayes-Klassifikation (siehe Formel 2.22). Ohne Logarithmierung entsteht – wie bereits erwähnt – arithmetischer Unterlauf. Die Werte werden somit unbrauchbar. Ebenso wurde die Laplace-Glättung implementiert, um mit Klassenwahrscheinlichkeiten von 0 umzugehen. Die Ergebnisse des Klassifikators sind in Abschnitt 5.2 beschrieben. Hier werden die drei Klassifikationsalgorithmen anhand der vorgestellten Evaluierungsmaße verglichen. Nachdem in den vorherigen zwei Kapiteln die Demoanwendungen zum Bildverstehen beschrieben wurden, widmet sich der folgende Abschnitt nun der Audioanalyse.

## 3.4 Spracherkennung mit Tiergattungsbegriffen und -lauten

Die dritte Demoanwendung setzt Sprachsignalerkennung anhand der Nachahmung von Tierlauten bzw. Nennung von Tiergattungen um. Zunächst wird in Abschnitt 3.4.1 die Aufgabenstellung spezifiziert, auf die Umsetzung wird in Abschnitt 3.4.2 eingegangen. Hier wird auch die Umsetzung der Theorie aus Kapitel 2 beschrieben.

### 3.4.1 Aufgabenstellung

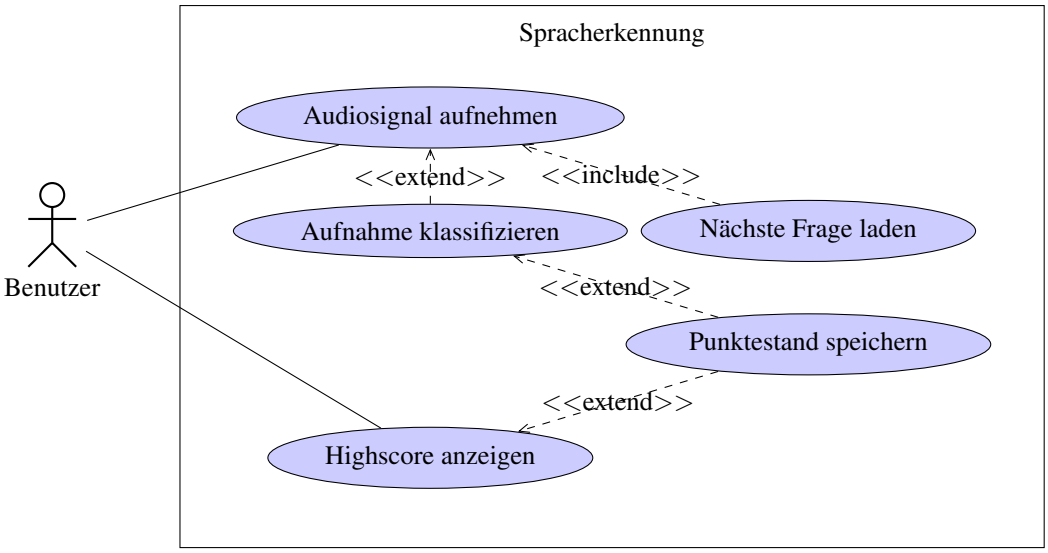



Abbildung 3.7: Anwendungsfalldiagramm der Sprachsignalerkennung.

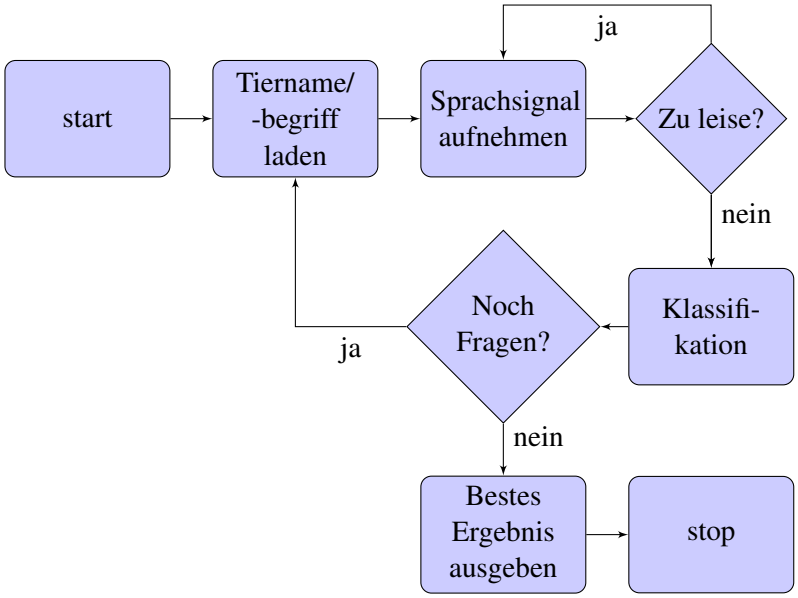



Abbildung 3.8: Ablauf bei der Sprachsignalerkennung.

In dieser Anwendung wird die automatisierte Spracherkennung demonstriert. Der Benutzer wird aufgefordert, Tierlaute nachzumachen bzw. gefragte Tiergattungen zu nennen. Zielsetzung dieses Spiels ist es, so schnell wie möglich dem Computer die korrekte Antwort zu geben. Die Fragen werden nacheinander eingeblendet. Der Benutzer muss mit dem korrekten Begriff antworten. Je schneller er antwortet, desto mehr Punkte werden vergeben (sofern der richtige Begriff erkannt wurde). Ziel ist es, so viele Punkte wie möglich zu erreichen. Die Nachahmung der Tierlaute muss allerdings nicht realistisch erfolgen, stattdessen ist eine Antwort mit dem gängigen deutschen Begriff des Lautes verlangt (wie z. B. das Wort *miau*). Andererseits können aber auch Fragen gestellt werden, die den Tiergattungsbegriff verlangen. Eine Frage dieser Art wäre z. B. *Welches Tier macht wuff?* Um den spielerischen Aspekt zu verdeutlichen, sind einige dieser Fragen in humorvoller Art formuliert. Sollte das aufgenommene Sprachsignal zu leise gewesen sein, hat der Benutzer die Möglichkeit die Antwort zu wiederholen. Hier wird allerdings – bedingt durch den entstandenen Vorteil des Wissens der Frage – die abgelaufene Zeit nicht zurückgesetzt. Sollte die Aufnahme laut genug gewesen sein, wird der Punktestand dementsprechend erhöht und per Zufallsprinzip die nächste Frage geladen. Nachdem alle Fragen beantwortet wurden, werden die Punkte der aktuellen Runde abgespeichert und der Highscore angezeigt. Der schematische Ablauf ist in Abbildung 3.8 ersichtlich. Auf die Umsetzung dieser Aufgabenstellung wird nun im folgenden Abschnitt näher eingegangen.

### 3.4.2 Lösung

Für MATLAB gibt es eine Toolbox namens *VOICEBOX*, die Sprachverarbeitung anbietet. Sie wurde von Mike Brookes et al. [Bro11] vom Department of Electrical and Electronic Engineering des Imperial College in London entwickelt und ist unter der GNU Public License verwend-

bar. Nachfolgend werden die Funktionen beschrieben, die bei der Spracherkennung verwendet werden. Für die Aufnahme der Audiodaten war eine Länge von zwei Sekunden mit einer Abtastrate von 11,025 kHz ausreichend. Laut Nyquist-Shannon-Theorem muss die Abtastfrequenz doppelt so hoch sein wie die höchste gewünschte abzubildende Frequenz, um alle gewünschten Frequenzen zu repräsentieren. Das menschliche Ohr nimmt beispielsweise Frequenzen bis ca. 20 kHz wahr, somit haben Audio-CDs beispielsweise eine Abtastrate von 44,1 kHz. Die maximal repräsentierte Frequenz liegt somit bei 22,05 kHz. Für die Spracherkennung ist allerdings keine verlustfreie Repräsentationsart erforderlich, die Inhalte können auch bei einer geringeren Abtastrate abgebildet werden. Für Sprachsignale ist eine Abtastrate von 8 kHz ausreichend [An09]. Durch die verringerte Abtastrate reduziert sich die Speichergröße der Audiosignale, wodurch der Klassifikationsvorgang schneller bewältigt werden kann.

#### 3.4.2.1 Aufbau des Trainingssets

Für eine zuverlässige Klassifizierung unterschiedlicher Sprecher ist es notwendig, ein Trainingsset von verschiedenen Sprechern aufzubauen. Für diese Applikation besteht es aus 17 männlichen und weiblichen Personen zwischen acht und 65 Jahren. Beim Aufbau machte sich rasch bemerkbar, dass eine hohe Diversität bei den Nachahmungen der Tierlaute vorhanden ist. Manche Personen versuchen diese realistisch nachzuahmen, andere wiederum antworten mit dem etablierten Lautwort, wie z. B. *miau* oder *kikeriki*. Diese Lautworte sind allerdings auch nicht eindeutig: Bei manchen Tiergattungen existieren mehrere davon, des Weiteren sind diese Unterschiede auch kulturell bedingt (vgl. *wau* bzw. *wuff*, oder das deutsche *kikeriki* im Gegensatz zum englischen *koo koo roo*). Da bei der realistischen Nachahmung der Tierlaute die Unterschiede zwischen verschiedenen Sprechern eklatant sind, liegt der Fokus auf die gängigen Tierlautbezeichnungen im deutschsprachigen Raum. Zusätzlich werden bei Tiergattungen, bei denen mehrere Lautworte existieren, alle gängigen Bezeichnungen ins Trainingsset aufgenommen, um eine höhere Erkennungsrate zu erzielen. Von der Aufnahme der realistischen Nachahmungen wurde abgesehen. Bei den Tierlauten wurden nach vollständiger Implementierung des Klassifikators und mehreren Testläufen verschiedene Klassen wieder entfernt, da sie nicht zuverlässig erkannt wurden. Die Tierlaute eines Pferdes wurden beispielsweise sehr unterschiedlich nachgemacht. Teilweise wurde mit den Worten *schnauf* geantwortet, oft jedoch auch mit dem Wort *wieher*. Auch die realistische Nachahmung war weit verbreitet und bereitete große Schwierigkeiten beim Aufbau eines zuverlässigen Trainingssets, weshalb schlussendlich davon abgesehen wurde, die Laute eines Pferdes in das Trainingsset aufzunehmen.

Der Aufbau des Trainingssets der Tiergattungsbegriffe war einfacher, da hier nur die Begriffe genannt werden müssen und die Fragen keinen großen Spielraum bei der Antwort lassen. Es werden ausschließlich Worte und keine Nachahmung von Lauten verlangt, weshalb der Aufbau eines Trainingssets nicht so aufwändig ist. Hier müssen lediglich genügend Sprecher aufgenommen werden, um eine möglichst breite Stimmvarietät abzubilden. Nachdem das Trainingsset aufgebaut ist, kann die Klassifizierung von Aufnahmen unbekannten Inhalts durchgeführt werden.

#### 3.4.2.2 Aufnahme des Audiosignals

Ein Problem bei den Audioaufnahmen in MATLAB ist, dass die Aufnahmedauer bekannt sein muss, bevor die Aufnahme gestartet wird. Ein durch den Benutzer manuell gestopptes Beenden der Aufnahme durch beispielsweise einen Buttondruck ist nicht möglich. Wenn die vorgegebene Aufnahmezeit knapp bemessen wird, besteht die Gefahr, dass das Audiosignal nicht vollständig aufgezeichnet wird. Bei einer zu langen Dauer muss das Audiosignal nachbearbeitet werden, indem Start- und Endpunkt des gesprochenen Wortes detektiert werden und das Signal dementsprechend gekürzt wird. Dieser Schritt kann durch eine manuelle Bedienung erfolgen, was allerdings zeitaufwändig und somit in diesem Kontext nicht umsetzbar ist. Das Aufsuchen des relevanten Teils des Audiosignals kann allerdings auch automatisiert erfolgen, was aber eine fehlerhafte Detektion nicht ausschließen kann (vor allem bei starken Nebengeräuschen).
Schlussendlich erwiesen sich zwei Sekunden aufgrund mehrerer Probeläufe mit Freiwilligen als Aufnahmedauer ausreichend. Eine kurze Aufnahmedauer hat den Vorteil, dass die Wahrscheinlichkeit auftretender Nebengeräusche vermindert und somit das Trimmen des Audiosignals erleichtert wird. Bei einer Aufnahmedauer von vier Sekunden gestaltet es sich schwierig, das Audiosignal automatisiert zurecht zu schneiden, da dann auftretende Nebengeräusche in der Aufnahme wahrscheinlicher sind. Diese Nebengeräusche erschweren eine zuverlässige Sprachsignalerkennung. Der Vorteil einer längeren Aufnahmedauer wäre jedoch, dass der Benutzer eine Toleranzzeit hat, bis er zu Sprechen beginnt, ohne dass das Audiosignal nicht komplett aufgenommen wird. Bei einer Aufnahmedauer von zwei Sekunden muss gleich mit dem Sprechen begonnen werden, da sonst das Ende des Wortes abgeschnitten wird. Trotzdem ist aus Sicht der Qualität der Ergebnisse die Aufnahme von zwei Sekunden sinnvoller.

**Bestimmung des Start- und Endpunkts des Audiosignals** Nach erfolgter Aufnahme wird mit einem Voice-Activity-Detection-Algorithmus (VAD-Algorithmus) das eigentliche Sprachsignal detektiert und ausgeschnitten. Im Gegensatz zu den Aufnahmen des Trainingsdatensatzes ist ein manuelles Zurechtschneiden des Audiosignals nicht möglich, da dies eine zeitaufwändige Interaktion des Benutzers erfordert, was nicht im Interesse des Anwenders liegt. Somit muss eine automatisierte Methode verwendet werden, um die Audiosignale zurechtzuschneiden. Auch wenn bei der VAD nicht immer das Sprachsignal zuverlässig erkannt wird, ist dessen Verwendung trotzdem sinnvoll. Die Detektion ist aufgrund der auftretenden Nebengeräusche immer etwas länger als das eigentliche gewünschte Audiosignal. Der umgekehrte Fall, dass relevante Information verworfen wird, tritt kaum auf. Somit ist ein zusätzliches automatisiertes Zurechtschneiden nur vom Vorteil für eine zuverlässige Klassifizierung, auch wenn diese nicht perfekt vonstatten geht. Die Bibliothek VOICEBOX hat mit der Funktion `vadsohn` einen VAD-Algorithmus implementiert:

```
[vs, zo] = vadsohn(si, fsz, m, pp)
```

Dieser Funktion werden das Sprachsignal `si` und dessen Abtastrate in Hertz (`fsz`) übergeben, die Parameter `m` und `pp` sind optional und geben Eigenschaften über den Rückgabewert zurück bzw. bieten zusätzliche Einstellungsmöglichkeiten für den Algorithmus. Für nähere Details sei auf die Dokumentation der Funktion in [Bro11] verwiesen. Je nachdem welche Werte für

`m` übergeben werden, kann das Sprachsignal anschließend getrimmt werden. Bei der Standardeinstellung `a` entscheidet der Algorithmus über die Zugehörigkeit zum Sprachsignal, bei der Einstellung `b` wird die Wahrscheinlichkeit der Zugehörigkeit zum Sprachsignal zurückgegeben. Der Rückgabevektor `zs` hat dieselbe Länge wie das Sprachsignal und enthält für jeden Abtastpunkt die Information der Zugehörigkeit zum Sprachsignal (von 0 bis 1). Bei bereits getroffener Entscheidung durch die Funktion gibt es keine Zwischenwerte.

Die Bereiche, die als Sprachsignal erkannt werden, müssen nicht unbedingt zusammenhängend sein. So sind zum Beispiel zwei Silben durch eine Pause getrennt, zum Sprachsignal gehören aber trotzdem beide Teile. Allerdings werden auch fälschlicherweise Nebengeräusche als Sprachsignal erkannt. In Abbildung 3.9 ist das Sprachsignal des Wortes *Biene* mit den Ergebnissen des VAD-Algorithmus abgebildet. In der oberen Abbildung ist die Wahrscheinlichkeit der Zugehörigkeit zum Sprachsignal und die endgültige Entscheidung eingezeichnet. Wie man erkennen kann, wird sowohl das Sprachsignal als auch die Nebengeräusche als Sprachsignal erkannt. Es wäre naheliegend, dass nur der längste zusammenhängende Bereich weiter verwendet wird, da in dieser Implementierung nur Worte und nicht zusammenhängende Sätze verarbeitet werden. Durch die Tatsache, dass Silben mitunter durch Pausen getrennt werden und somit das Sprachsignal aus Teilsequenzen besteht, muss die gesamte Sprachinformation verwendet werden. Nach der Bearbeitung werden die MFCC für die anschließende Klassifizierung berechnet.

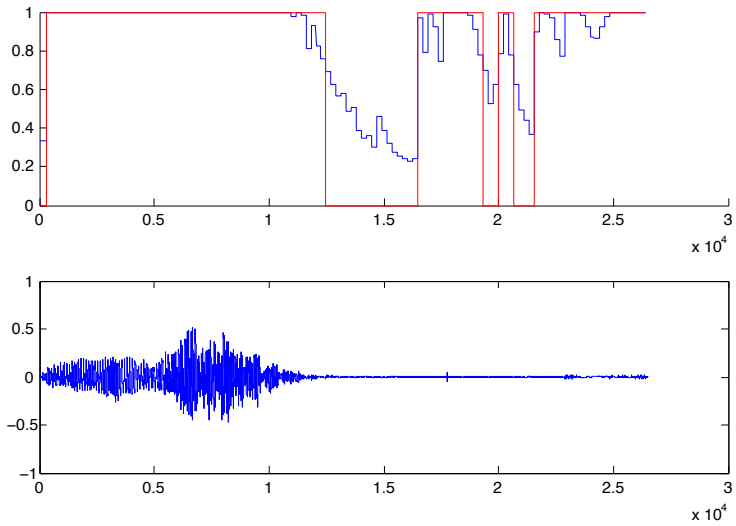



Abbildung 3.9: VAD beim Wort Biene. In der oberen Hälfte der Abbildung ist die Wahrscheinlichkeit (stufenförmige Funktion) und die Entscheidung (rechteckige Funktion) eingezeichnet. Der Schwellwert für die Detektion liegt standardmäßig bei einer Wahrscheinlichkeit von 0,7 was sich bei Probeläufen als der beste Wert herausgestellt hat. Die korrespondierende untere Abbildung zeigt die zeitliche Veränderung des Sprachsignals.

#### 3.4.2.3 Berechnung der MFCC

Die Berechnung der MFCC ist mit der Funktion `melcepst` aus der Bibliothek VOICEBOX möglich.

```
c = melcepst(s, fs, w, nc, p, n, inc, fl, fh)
```

Verpflichtende Übergabeparameter sind das Signal `s` und die Abtastrate in Hertz (`fs`), alle anderen Parameter sind optional und bieten einige Verfeinerungen (wie z. B. die Festlegung der Anzahl der Koeffizienten). Die Funktion gibt eine Matrix zurück, bei der jede Zeile einem Frame entspricht. Falls bei den Einstellungen die log-Energie gewünscht wurde, befindet sich diese in der ersten Zeile, ansonsten beginnt die Ausgabe mit dem ersten Koeffizienten.
Die MFCC beschreiben die Information des Audiosignals und müssen sowohl für das Trainingsset als auch für die Aufnahme des Benutzers, die gerade klassifiziert werden muss, berechnet werden. Um die Performance des Algorithmus zu verbessern, werden die Koeffizienten der Referenzdaten beim ersten Vergleich berechnet und anschließend gespeichert, sodass sie bei der nächsten Klassifizierung wiederverwendet werden können. Der Vergleich der Koeffizienten der Aufnahme mit denen des Trainingssets erfolgt mittels Dynamic Time Warping. In dieser Implementierung kann aus Performancegründen auf die Vergleiche mit den Tiergattungsbezeichnungen verzichtet werden (wenn Tierlaute verlangt werden). Genauso kann der Vergleich mit den Tierlautaufnahmen verzichtet werden, wenn ein Tiergattungsbegriff verlangt wird. Somit erfolgt die Klassifizierung bedeutend schneller. Wie der Vergleich mittels Dynamic Time Warping im Konkreten aussieht, wird im folgenden Abschnitt beschrieben.

#### 3.4.2.4 Ähnlichkeitsberechnung mittels Dynamic Time Warping

Jeweils zwei Audiosignale werden mittels Dynamic Time Warping (DTW) auf dieselbe Länge gebracht, die entstandenen Kosten des Pfades werden anschließend für die Messung der Übereinstimmung der beiden Signale verwendet. Je niedriger die ermittelten Kosten, desto ähnlicher sind sich die beiden Signale. Das DTW wird in der Funktion `DTW` in der gleichnamigen MATLAB-Datei berechnet, der die Featurematrix `F` und die Referenzmatrix `R` übergeben werden. Die Funktion samt Rückgabewerten und Übergabeaparametern schaut folgendermaßen aus:

```
[cost] = DTW(F, R).
```

Zunächst werden die Distanzen der beiden Eingangsmatrizen ermittelt und in einer Matrix abgespeichert. Diese Matrix enthält an der Stelle $(m, n)$ die euklidische Distanz zwischen der $m$-ten Zeile der Feature- und der $n$-ten Zeile der Referenzmatrix. Die Featurematrix ist diejenige, die die Koeffizienten des Sprachsignals des Benutzers beinhaltet. In der Referenzmatrix befinden sich die Koeffizienten des gerade betrachteten Tierlauts bzw. Tiernamens. Ein MATLAB-bedingtes Problem war, dass die Performance durch zwei ineinander verschachtelte Schleifen sehr schlecht war, weshalb versucht wurde, diese durch Vektoroperationen zu ersetzen.
Aus dieser Distanzmatrix werden nun die Kosten ermittelt. Es wird der Pfad in der Distanzmatrix gesucht, der die minimalen Kosten ergibt. Begonnen wird in der ersten Zelle. Je nachdem welche Nachbarzelle den niedrigeren Wert beinhaltet, wird der entsprechende Wert dazu addiert.

Die Nachbarn, die betrachtet werden, sind sowohl die beiden angrenzenden Zellen rechts und unterhalb der Zelle, als auch die rechts unten anschließende Zelle. Wenn das Ende der Matrix erreicht wird, werden die Kosten zurückgegeben.
Auch hier ist die Performance durch zwei ineinander verschachtelte Schleifen durch die Eigenschaften von MATLAB nicht zufriedenstellend. Solche Probleme lassen sich aber auch durch die Verwendung von MATLAB Executables (MEX) umgehen, bei denen unperformante Subroutinen in C++ ausgelagert werden können. Die Implementierung von Ellis [Ell03] arbeitet damit. Er verwendet für die Ermittlung der Lösung dynamische Programmierung. Sobald die Vergleiche mit allen Dateien durchgeführt wurden, kann mittels kNN-Algorithmus die Klassenzugehörigkeit bestimmt werden.

#### 3.4.2.5 Klassifizierung mittels kNN-Algorithmus

Beim kNN-Algorithmus zeigte sich, dass ein $k = 3$ bessere Ergebnisse lieferte als ein $k = 5$ oder $k = 7$, weshalb bei der finalen Implementierung des Programms dieser Wert verwendet wurde. Sollte unter den ersten $k$ Elementen keine Mehrheit vorhanden sein, wurden verschiedene Ansätze verfolgt und die Erfolgsrate verglichen. Einerseits wurde das $k$ so lange erhöht, bis eine Mehrheit vorhanden ist, welche dann für die Klassifizierung entscheidend ist. Andererseits wurde bei solchen Fällen die Klasse der ähnlichsten Datei entscheidungstreffend (somit ein kNN-Algorithmus mit $k = 1$). Die zweite Methode wurde schlussendlich verwendet. Zunächst werden die Kosten sortiert, die ersten drei bestimmen nun die Klasse des gesprochenen Wortes und somit, was der Benutzer gesagt haben könnte. Implementiert wurde das mit der Funktion `histc`, die die Anzahl der Vorkommen pro Wort ermittelt. Der gesamte Vorgang ist in der Funktion `getClassLabel` implementiert und hat folgende Übergabeparameter und Rückgabewerte:

```
[index, animalClass, value] = getClassLabel (trainingDir,
                    testFile, username)
```

Das Verzeichnis des Trainingssets wird über `trainingDir` übergeben. Dieser Parameter ist erforderlich, da bei verlangten Tierlauten nur mit dem Trainingsset, das die Tierlaute beinhaltet, verglichen wird. Genauso wird bei verlangten Tiergattungsbegriffen nur mit dem Trainingsset der Tiergattungsbegriffe verglichen. Die Aufnahme des Benutzers wird mittels `testFile` übergeben und ist der Name der Audiodatei, die getestet werden soll. Der `username` ist optional und muss nicht angeführt werden. Dieser Übergabeparameter war nur für Testdurchläufe erforderlich, um Vergleiche mit Aufnahmen desselben Sprechers zu vermeiden. Zurückgegeben werden drei Vektoren, die jeweils die Werte für $k = 3$, $k = 5$ und $k = 7$ beinhalten. `index` beinhaltet den Index der Tiergattungen bzw. Tierlaute, `animalClass` dessen Kurzbezeichnung wie z. B. *Schwein_grunz*. `value` speichert die Anzahl der Übereinstimmungen. Wenn z. B. bei einem $k = 5$ entschieden wird, dass es sich bei einer Aufnahme um eine Katze handelt, ist in `value` gespeichert, wieviele von den fünf nächsten Nachbarn Aufnahmen vom Wort *Katze* sind. Da – wie bereits erwähnt – diese Funktion auch für die Testphase verwendet wird, um die Qualität des Algorithmus festzustellen, sind einige Parameter nicht erforderlich in der Endversion. Aus diesem Grund gibt es die Funktion `testAudio`, die die oben genannte Funktion aufruft.

```
[index, animalClass, value] = testAudio(trainingDir)
```

Der Unterschied zur Funktion `getClassLabel` ist, dass weder `username` noch `testFile` angegeben werden müssen. Wie oben angeführt, dient die Angabe des Benutzernamens nur für den Ausschluss von Vergleichen eines Sprechers mit Aufnahmen desselben Sprechers aus dem Trainingsset. Ebenso ist der Name der Testdatei nun nicht mehr erforderlich, da nur noch die aktuelle Aufnahme getestet wird.
Nachdem in diesem Abschnitt die Umsetzung der Aufgabenstellung erläutert wurde, erfolgt im kommenden Abschnitt die Programmbeschreibung. Es wird insbesondere die Benutzerschnittstelle erklärt und auf die Usability eingegangen.

KAPITEL 4

# Programmbeschreibung

Die Demoanwendungen sollten einfach gehalten werden, sodass sie ohne Einschulung verwendet werden können. Da der spielerische Aspekt im Vordergrund steht, soll auf eine komplexe Oberfläche verzichtet und stattdessen intuitive Interaktion ermöglicht werden. Wichtige Anforderungen an die Programmoberfläche sind:

- **Intuitive Bedienung**: Die Oberfläche soll selbsterklärend sein.
- **Wenige Einstellungsmöglichkeiten**: Der Spielspaß soll im Vordergrund stehen.
- **Wenige Interaktionsschritte**: Einfache Bedienung ermöglicht das rasche Starten des Spiels.
- Übersichtliche, selbsterklärende **Darstellung der Ergebnisse**.
- **Überbrückung von Wartezeiten**: Bei längeren Berechnungszeiten soll die Wartezeit interessant für den Benutzer gestaltet werden.
- **Spielspaß** soll auch bei mehreren Durchläufen gegeben sein: Zufällige Reihenfolge der Fragen.
- Bei den Applikationen, die die Schnelligkeit des Benutzers beurteilen, sollen dementsprechend gerecht die Punkte vergeben werden, sodass eine **Highscoreliste** erstellt werden kann.
- **Erweiterungsmöglichkeiten und individuelle Anpassung**.

Wie die Anforderungen an die Benutzeroberfläche umgesetzt wurden, ist den folgenden Unterkapiteln pro Applikation separat zu entnehmen.

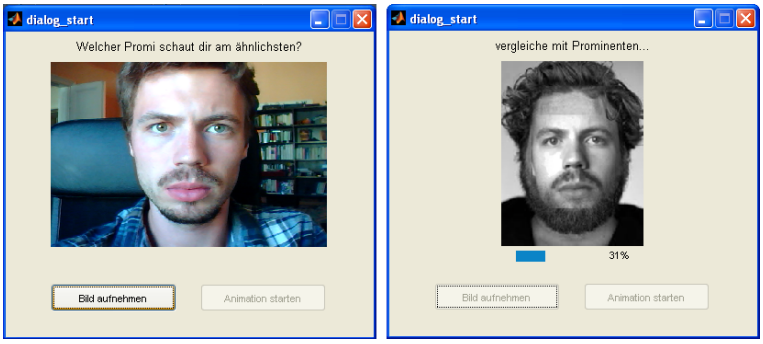



Abbildung 4.1: Benutzeroberfläche bei der Ähnlichkeitsberechnung zu Prominenten. Links ist die Oberfläche abgebildet, wenn der Benutzer das Foto seines Gesichts macht, rechts ist die Ansicht abgebildet, bei der die Prominentengesichter mit dem Benutzer verglichen werden.

## 4.1 Ähnlichkeitsberechnung zu prominenten Personen

Die Benutzeroberfläche öffnet sich durch die Funktion `dialog_start`. Aus einem Drop-Down-Menü wählt der Benutzer nun diejenige Webcam, die für die Aufnahme verwendet werden soll. Anschließend erscheint das Bild der Webcam auf dem Monitor. Der Benutzer muss sich nun so positionieren, dass das Gesicht vollständig in der Aufnahme sichtbar ist und kann zu einem beliebigen Zeitpunkt per Buttondruck die Kamera auslösen. Anschließend wird sein Gesicht erkannt und für den Vergleich mit den Prominentengesichtern verwendet. Der Gesichtserkennungsalgorithmus funktioniert so zuverlässig, dass eine manuelle Markierung des Gesichts vom Benutzer nicht erforderlich ist.
Da der Vergleich des Templates mit den Referenzbildern einige Sekunden pro Bild beansprucht, werden die Zwischenergebnisse laufend ausgegeben, um die Wartezeit bis zum Endergebnis interessanter zu gestalten. Zusätzlich wird eine Statusanzeige mit einem Fortschrittsbalken eingeblendet. Für die Darstellung der Resultate wird der Ausschnitt des Benutzergesichtes auf die Koordinaten im aufgenommenen Bild des Anwenders abgebildet, an dem das beste Ergebnis auftritt. Außerdem werden die Randbereiche mit Alpha-Blending abgeschwächt, sodass keine scharfen Kanten entstehen. Sobald alle Prominentengesichter durchlaufen wurden, wird die beste Übereinstimmung angezeigt. Hierfür wird eine Animation ausgegeben, die ein Bild des Prominenten mit darübergelegten Benutzergesicht auf das Prominentenfoto überblendet. Des Weiteren wird der Name des Prominenten ausgegeben. Der Überblendungsvorgang kann vom Benutzer durch Buttondruck wiederholt werden.

### 4.1.1 Erweiterungsmöglichkeit des Pools mit den Prominentengesichtern

Die Bilder der Prominentengesichter liegen alle im Unterverzeichnis *celebs* und müssen vom Dateiformat JPEG sein. Zusätzlich muss zu jeder Bilddatei auch eine gleichnamige Textdatei vorhanden sein, die den Namen des Prominenten beinhaltet, welcher in der Datei unter Anführungszeichen gesetzt werden muss. Das Programm ist somit leicht vom Benutzer erweiterbar. Der Ausschnitt des Prominentengesichtes wird beim ersten Aufruf des neuen Bildes berechnet.

Die Koordinaten werden für weitere Programmdurchläufe gespeichert und müssen nicht nochmals ermittelt werden.

## 4.2 Körperteilerkennung

Im Demo zur Erkennung der Körperteile ist für die Benutzerinteraktion lediglich ein Button nötig, der das Bild des Benutzers aufnimmt. Die Liveansicht der Webcam wird gespiegelt dargestellt, sodass dem Benutzer das Bewegen vor der Kamera leichter fällt (siehe Abbildung 4.2 b). Der Körperteil muss innerhalb des weißen Rechtecks liegen, damit er aufgenommen wird. Dieser Bereich hat die Größe $200 \times 300$ Pixel. Die gewählte Größe hat den Vorteil, dass trotz Beschränkung der Größe alle Körperteile bequem in die Kamera gehalten werden können, ohne dass zu viel vom Hintergrund in der Aufnahme sichtbar ist, wodurch die Qualität der Ergebnisse steigt. Der Bereich über der Liveansicht (Abbildung 4.2 a) zeigt an, welches Körperteil als nächstes in die Kamera gehalten werden soll. Neben der textuellen Beschreibung ist auch eine Darstellung des Körperteils mit der gewünschten Orientierung abgebildet. Somit können Fehler durch falsche Interaktion weitgehend vermieden werden. Nach Betätigung des Auslösers (Abbildung 4.2 c) wird die Aufnahme klassifiziert und der Zähler für die korrekten Körperteile erhöht, sollte das richtige Körperteil erkannt werden. Die Reihenfolge der Aufforderungen wird anhand eines Zufallsgenerators ausgewählt, sodass der Spielspaß auch nach mehreren Runden noch garantiert ist. Zusätzlich wird auch die Zeit der Interaktion gestoppt. Um möglichst gut bei diesem Spiel abzuschneiden, genügt es somit nicht, die richtigen Körperteile zu zeigen, es ist außerdem eine schnelle Interaktion erforderlich. Für jede korrekte Frage gibt es mindestens einen und ma-

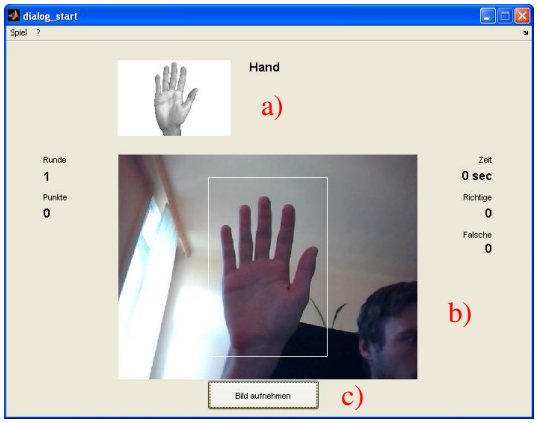



Abbildung 4.2: Benutzeroberfläche bei der Körperteilerkennung.

ximal 16 Punkte zu erreichen, wobei die Anzahl der vergebenen Punkte von der Reaktionszeit des Benutzers abhängig ist. Für jede verstrichene Sekunde wird ein Punkt weniger vergeben. Wenn also mehr als fünfzehn Sekunden benötigt werden, ändern sich die vergebenen Punkte nicht mehr, sondern bleiben konstant bei einem Punkt. Für falsche Antworten bekommt man keine Punkte, egal wie schnell die Interaktion abläuft. Nachdem alle Körperteile durchlaufen wurden, werden die Punkte samt Benutzernamen gespeichert und die Highscoreliste angezeigt. In dieser Anwendung ist keine Erweiterungsmöglichkeit vorgesehen.

## 4.3 Spracherkennung

Das Spiel beginnt, sobald der Benutzer seinen Namen eingegeben und auf den Start-Button gedrückt hat. Der Name wird für die Eintragung in die Highscoreliste benötigt. Die Fragen sind auf humorvolle Weise gestellt. Der Benutzer muss schnell interagieren, da die Reaktionszeit Auswirkungen auf die erzielten Punkte hat. Die vergangene Zeit wird nach Klick auf den Button *Aufnehmen* ermittelt, gleichzeitig wird mit der Aufnahme begonnen. Da die Berechnung der Ähnlichkeit zwischen der Aufnahme des Benutzers und dem gesamten Trainingsset jeweils ca. drei Sekunden dauert, wird die Reaktionszeit für die Punktevergabe verwendet. Die Berechnung dauert nämlich je nach Aufnahme und Benutzer unterschiedlich lange, weshalb es ungerecht wäre, die Gesamtzeit des Vergleichs dafür zu verwenden.
Insgesamt werden zwölf Fragen zu Tierlauten und vierzehn Fragen zu Tiergattungsbegriffen gestellt, die in zufälliger Reihenfolge ausgegeben werden. Der Benutzer startet die Aufnahme per Buttondruck, die Aufnahme wird automatisch nach zwei Sekunden beendet. Sollte das Signal zu leise sein, besteht die Möglichkeit, die Aufnahme zu wiederholen, da sonst keine zuverlässige Klassifikation möglich wäre. Sollte die Aufnahme wiederholt werden müssen, wird allerdings die gestoppte Zeit trotzdem nicht zurückgesetzt, um dem Informationsvorteil des Benutzers entgegenzuwirken. Die Dauer der erneuten Interaktion wird zur vorherigen Zeit addiert.
Der Benutzer wird laufend über den Status des Programms aufgeklärt, da der Vergleich mit dem Trainingsset etwas Zeit in Anspruch nimmt. Der Benutzer erfährt somit, ob die Aufnahme bereits läuft oder ob die Aufnahme abgeschlossen ist und der Vergleich mit den Dateien im Trainingsset gerade läuft. Sobald die Analyse abgeschlossen ist, erhöht sich die Anzahl der korrekten Antworten, die zusätzlich zur Zeit des Benutzers angezeigt wird. Der Punktestand des Benutzers ermittelt sich aus der Anzahl der korrekten Antworten und der Dauer, die der Benutzer benötigt. Sie wird in die Highscoreliste eingetragen. Sobald alle Fragen beantwortet wurden, öffnet sich der Dialog mit dem Highscore, die allerdings auch manuell vor oder während des Spiels per Buttondruck geöffnet werden kann. In dieser Liste werden die zehn besten Resultate angezeigt.

### 4.3.1 Benutzerdefinierte Anpassung der Fragen

Die Fragen zu den gewünschten Begriffen sind als Textdateien in den jeweiligen Ordnern gespeichert. Die Tiergattungsbegriffe befinden sich im Ordner *tiernamen*, die Tierlaute im Ordner *tierlaute*. Jeder dieser Ordner enthält pro Begriff einen Unterordner. Da es bei den Tierlauten allerdings mehrere verschiedene Laute gibt, gibt es pro Tier mitunter mehrere Ordner. Die Fra-

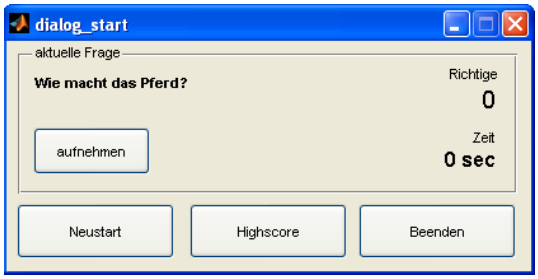



Abbildung 4.3: Benutzeroberfläche bei der Spracherkennung.

gen zu den Tiergattungsbegriffen sind in den Ordnern enthalten. Die Fragen zu den Tierlauten sind nicht im Ordner *tierlaute*, sondern im Ordner *tierlaute_questions* enthalten, die ebenfalls pro Tiergattungs-Unterordner eine Textdatei enthalten. Aufgrund der Tatsache, dass mehrere Begriffe pro Tier existieren, können diese nicht in diesen Ordnern gespeichert werden, da sonst zum selben Tier mehrmals eine Frage gestellt würde.
Die Audioaufnahmen der Tierlaute und Tiergattungsbegriffe befinden sich in den jeweiligen Tier-Unterordnern und müssen vom Dateityp *wav* sein. Da allerdings der Aufbau eines zuverlässigen Trainingssets viele Aufnahmen von unterschiedlichen Benutzern erforderlich sind, erfordert die Erweiterung des vorhandenen Sets Expertenkenntnisse und viel Zeit.

KAPITEL 5

# Evaluierung der Ergebnisse

In diesem Kapitel wird die Qualität der implementieren Demos überprüft. Als Grundlage werden die in Abschnitt 2.2.5 vorgestellten Gütemaße verwendet, mit denen z. B. bei der Körperteilerkennung die drei implementierten Klassifkationsalgorithmen verglichen werden können. Um die jeweiligen Algorithmen für die Klassifikation zu bewerten, wurden pro Demoanwendung ein Test- und ein Trainingsset von Bildern bzw. Audiosignalen erstellt. Das Testset wurde anschließend mit den implementierten Algorithmen klassifiziert und überprüft, ob die detektierte Klasse mit der tatsächlichen übereinstimmt. Im Folgenden werden aus Gründen der Übersichtlichkeit wiederum die drei Anwendungen separat behandelt.

## 5.1 Ähnlichkeitsberechnung zu prominenten Personen

Für die Gesichtserkennung wurde der von Mikael Nilsson implementierte Algorithmus verwendet [NNC07]. Um die Qualität dieses Klassifikators festzustellen und ihn somit auf die Praxisfähigkeit zu überprüfen, wurde ein Testset von 93 Bildern aufgebaut und überprüft, wie viele Gesichter auf diesen Bildern korrekt erkannt werden. Die Rate der korrekt erkannten Gesichtern lag bei 92 Prozent bzw. 86 Gesichtern. Der Gesichtserkennungsalgorithmus liefert somit zufriedenstellende Ergebnisse und ist für die Vorselektion des zu untersuchenden Bereichs und die Erstellung des Templates gut geeignet.
Außerdem wurde getestet, bei wie vielen Template-Matching-Vergleichen der jeweils besten Resultate das Gesicht des Benutzers mit der Position und Orientierung des Gesichtes des Prominenten übereinstimmt. Von 62 getesteten Bildern lagen bei zwölf Bildern die Gesichter der besten Übereinstimmungen nicht genau über der Position eines Prominentengesichtes bzw. stimmte die Mimik nicht überein, somit sind bei 80,65 Prozent eine korrekte Übereinstimmung gegeben. In Abbildung 5.1 ist exemplarisch ein korrektes und nicht korrektes Resultat abgebildet.
Für die Bestimmung der Ähnlichkeit wurden Methoden gesucht, die diese auf objektive Weise testen können. Zu Testzwecken wurden Portraits auf Ähnlichkeit untersucht, die auch im Set der Prominentenbilder vorhanden sind, wobei es sich um unterschiedliche Fotos handelt. Da es allerdings schwierig ist, mehrere unterschiedliche frontal aufgenommene Fotos von Prominenten

Abbildung 5.1: Beim linken Resultat liegt das Gesicht nicht genau über dem besten Prominenten, beim rechten schon.

| | Person 1 | Person 2 | Person 3 | Jolie | User's Acc. (%) |
|---|---|---|---|---|---|
| Person 1 | 8 | 1 | 0 | 0 | 89 |
| Person 2 | 0 | 4 | 5 | 0 | 44 |
| Person 3 | 0 | 0 | 7 | 0 | 100 |
| Jolie | 0 | 0 | 1 | 2 | 67 |
| Sting | 0 | 0 | 0 | 2 | - |
| Obama | 2 | 0 | 0 | 0 | - |
| Nicholson | 0 | 0 | 1 | 0 | - |
| Clinton | 1 | 0 | 0 | 0 | - |
| Faymann | 1 | 0 | 0 | 0 | - |
| Allen | 0 | 0 | 1 | 0 | - |
| Producer's Acc. (%) | 67 | 80 | 47 | 50 | |

Tabelle 5.1: Testdatenset bei der Ähnlichkeitsberechnung der Gesichter.

zu finden, wurden für die Testphase in das Trainings- und Testset Fotos von Personen aufgenommen, die nicht prominent sind. Im Idealfall wird ein Bild derselben Person als ähnlichste Person ausgegeben. Da allerdings auch die Orientierung, Mimik etc. Einfluss auf die Ergebnisse haben, ist davon auszugehen, dass nicht alle Personen erkannt werden. Da es sich bei diesem Verfahren allerdings um keinen zuverlässigen Klassifikator handelt, ist eine starke Übereinstimmung nicht erforderlich. Positiv anzumerken ist jedoch, dass sehr wohl eine Tendenz zu denselben Personen gegeben ist (siehe Tabelle 5.1). Die verwendeten Evaluierungsmaße (User's Accuracy sowie Producer's Accuracy) werden in Abschnitt 2.2.5 erklärt.

## 5.2 Körperteilerkennung

Für die Evaluierung der Qualität des Klassifikators wurde ein Testdatenset von mehreren hundert Bildern angelegt, deren Klassenzugehörigkeit bekannt ist. Die genaue Anzahl an Bildern pro Klasse ist in Tabelle 5.2 aufgelistet. Dieses Datenset besteht wie das Trainingsset aus Bil-

| Klasse | Testset | Trainingsset |
|---|---|---|
| Ellenbogen | 56 | 28 |
| Faust | 98 | 34 |
| Fuß | 96 | 124 |
| Gesicht | 115 | 44 |
| Hand | 80 | 33 |
| Knie | 87 | 28 |
| Nabel | 90 | 28 |
| Ohr | 55 | 28 |

Tabelle 5.2: Trainings- und Testdatensetgröße bei der Körperteilerkennung.

dern mit einer Größe von $200 \times 300$ Pixel. Das Testset besteht aus Bildern der vorgegebenen Klassen welche mittels der Klassifikationsalgorithmen in Klassen eingeteilt werden. Die Bilder wurden von unterschiedlichen Personen in einem realistischen Umfeld aufgenommen. Die detektierte Klasse wird mit der ausgegebenen Klasse verglichen und auf Korrektheit überprüft. Auf diese Weise lassen sich die Konfusionsmatrix und die Gütemaße berechnen. Anhand dieser Werte lassen sich die drei implementierten Algorithmen vergleichen und Rückschlüsse ziehen, welche dieser Methoden am besten geeignet sind.

### 5.2.1 Vergleich der Ergebnisse der implementierten Klassifikationsalgorithmen

Das Testset wurde mit den drei implementierten Algorithmen klassifiziert. Die Ergebnisse sind in Tabelle 5.3 dargestellt. Wie man sehen kann, liefert die SVM die besten Ergebnisse, knapp gefolgt vom Naive-Bayes-Verfahren. An letzter Stelle liegt die Klassifikation mittels kNN-Algorithmus. Nachfolgend sind in Tabelle 5.4 die Konfusionsmatrizen der drei Algorithmen abgebildet, die die Basis für die Berechnung der Gütemaße bilden. Die Ergebnisse der Klassifikationsalgorithmen sind sehr zufriedenstellend. Ein Kappa-Koeffizient zwischen 0,6 und 0,8 zeigt eine starke Übereinstimmung, bei Werten zwischen 0,8 und 1 ist eine (fast) vollständige Übereinstimmung gegeben. Der hohe Wert wurde insbesondere durch die Einschränkung des Aufnahmebereichs erreicht. Außerdem kommt auch die geringe Anzahl an Klassen einer hohen Erkennungsrate zugute.

| Algorithmus | Overall Acc. (%) | Kappa-Koeffizient |
|---|---|---|
| kNN ($k = 10$) | 75,48 | 0,7170 |
| Naive Bayes | 97,64 | 0,9728 |
| SVM | 98,67 | 0,9847 |

Tabelle 5.3: Ergebnisse der Klassifikatoren bei der Körperteilerkennung.

| | | Ellenbogen | Faust | Fuß | Gesicht | Hand | Knie | Nabel | Ohr | User's Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| kNN, $k = 10$ | Ellenbogen | 46 | 17 | 0 | 0 | 0 | 0 | 0 | 1 | 71,9 |
| | Faust | 1 | 46 | 0 | 3 | 1 | 0 | 0 | 0 | 90,2 |
| | Fuß | 0 | 26 | 96 | 2 | 57 | 19 | 0 | 1 | 47,8 |
| | Gesicht | 0 | 0 | 0 | 110 | 0 | 0 | 18 | 0 | 85,9 |
| | Hand | 0 | 8 | 0 | 0 | 22 | 2 | 0 | 0 | 68,8 |
| | Knie | 0 | 0 | 0 | 0 | 0 | 66 | 0 | 0 | 100 |
| | Nabel | 9 | 1 | 0 | 0 | 0 | 0 | 72 | 0 | 87,8 |
| | Ohr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 53 | 100 |
| | Producer's Acc. (%) | 82,1 | 46,9 | 100 | 95,7 | 27,5 | 75,9 | 80 | 96,4 | 75,5 |
| Naive Bayes | Ellenbogen | 54 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 |
| | Faust | 2 | 89 | 0 | 1 | 0 | 1 | 0 | 0 | 95,7 |
| | Fuß | 0 | 0 | 95 | 0 | 0 | 1 | 0 | 0 | 99 |
| | Gesicht | 0 | 0 | 0 | 113 | 0 | 0 | 0 | 0 | 100 |
| | Hand | 0 | 9 | 1 | 0 | 80 | 0 | 0 | 0 | 88,9 |
| | Knie | 0 | 0 | 0 | 0 | 0 | 85 | 0 | 0 | 100 |
| | Nabel | 0 | 0 | 0 | 0 | 0 | 0 | 90 | 0 | 100 |
| | Ohr | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 55 | 98,2 |
| | Producer's Acc. (%) | 96,4 | 90,8 | 99,0 | 98,3 | 100 | 97,7 | 100 | 100 | 97,6 |
| SVM | Ellenbogen | 54 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 |
| | Faust | 2 | 93 | 0 | 0 | 0 | 0 | 0 | 0 | 97,9 |
| | Fuß | 0 | 1 | 96 | 0 | 0 | 2 | 0 | 0 | 97 |
| | Gesicht | 0 | 0 | 0 | 115 | 0 | 0 | 0 | 0 | 100 |
| | Hand | 0 | 4 | 0 | 0 | 80 | 0 | 0 | 0 | 95,2 |
| | Knie | 0 | 0 | 0 | 0 | 0 | 85 | 0 | 0 | 100 |
| | Nabel | 0 | 0 | 0 | 0 | 0 | 0 | 90 | 0 | 100 |
| | Ohr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 55 | 100 |
| | Producer's Acc. (%) | 96,4 | 94,9 | 100 | 100 | 100 | 97,7 | 100 | 100 | 98,7 |

Tabelle 5.4: Konfusionsmatrizen bei der Körperteilerkennung.

## 5.3 Spracherkennung

Für die Spracherkennung wurden für die Tierlaute acht Klassen und für die Tiergattungsbegriffe neun Klassen von unterschiedlichen Sprechern verwendet. Die Trainingsklassen bestehen jeweils aus ca. 20 Audiodateien. Für das Testset wurden Aufnahmen von drei Personen verwendet. Wurden die Testpersonen auch für den Aufbau des Trainingssets verwendet, so wurden diese ausgeschlossen und nur anhand anderer Sprecher getestet. Logischerweise wären sonst die Ergebnisse besser und für eine Beurteilung des Algorithmus nicht aussagekräftig. Details zum Aufbau der Trainings- und Testsets sind den Tabellen 5.5 und 5.6 zu entnehmen.

| Tier | Trainingsset | Testset |
|---|---|---|
| Biene | 18 | 13 |
| Hahn | 22 | 16 |
| Hund | 23 | 24 |
| Katze | 20 | 15 |
| Kuh | 16 | 19 |
| Pferd | 12 | 16 |
| Rabe | 23 | 20 |
| Schaf | 18 | 20 |
| Schwein | 22 | 16 |

Tabelle 5.5: Bilderanzahl Tiergattungsbegriffe.

| Tier | Trainingsset | Testset |
|---|---|---|
| Biene | 10 | 70 |
| Biene (*summ*) | 23 | 30 |
| Hahn | 22 | 105 |
| Katze | 26 | 92 |
| Kuh | 20 | 71 |
| Schaf | 24 | 54 |
| Schwein | 19 | 45 |
| Schwein (*grunz*) | 23 | 26 |

Tabelle 5.6: Bilderanzahl bei Tierlauten.

Wie bereits erwähnt sind die Ergebnisse bei der Erkennung der Tiergattungsbegriffe besser als bei der Erkennung der Tierlaute, da bei diesen eine größere Vielfalt bei der Nachahmung gegeben ist. Die Ergebnisse sind Tabelle 5.7 zu entnehmen. Für die Demoanwendung wurde jeweils mittels DTW die Ähnlichkeit zwischen den Features der getesteten Aufnahme und den MFCC des Trainingssets berechnet. Für die endgültige Klassenzuordnung wurde ein kNN-Algorithmus mit einem $k = 3$ verwendet. Die Ergebnisse waren besser oder nahezu gleich gut als bei Verwendung eines $k = 5$ oder 7. Der Kappa-Koeffizient deutet auf eine sehr gute Klassifikation hin. Wie zuvor angeführt, deutet ein Kappa-Koeffizient zwischen 0,6 und 0,8 auf eine starke und ein Kappa-Koeffizient größer als 0,8 auf eine nahezu identische Übereinstimmung hin. Begründet werden kann das durch die geringe Größe des Vokabulars von neun bzw. acht verschiedenen Begriffen. Außerdem ist den Ergebnissen die geringere Erkennungsrate der Tierlauten zu entnehmen, welche durch die höhere Varianz der Sprecher verursacht wird.

Um die Zuverlässigkeit des Programms zu testen, wurde ein Testset mit unterschiedlichen Sprechern aufgebaut, welches mit den Trainingsset verglichen wird. Zur Evaluierung der Qualität

| | Tiergattungsbegriffe | | | Tierlaute | | |
|---|---|---|---|---|---|---|
| Gütewert | $k = 3$ | $k = 5$ | $k = 7$ | $k = 3$ | $k = 5$ | $k = 7$ |
| Overall Accuracy (in %) | 86,16 | 79,87 | 70,44 | 80,32 | 80,73 | 77,89 |
| Kappa-Koeffizient | 0,8436 | 0,7726 | 0,6661 | 0,7719 | 0,7761 | 0,7434 |

Tabelle 5.7: Ergebnisse der Spracherkennung.

| | Biene | Hahn | Hund | Katze | Kuh | Pferd | Rabe | Schaf | Schwein | User's Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|---|
| Biene | 11 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 91,7 |
| Hahn | 0 | 13 | 0 | 0 | 1 | 1 | 2 | 2 | 0 | 68,4 |
| Hund | 2 | 2 | 22 | 0 | 3 | 1 | 0 | 0 | 0 | 73,3 |
| Katze | 0 | 0 | 0 | 15 | 0 | 0 | 1 | 1 | 0 | 88,2 |
| Kuh | 0 | 0 | 0 | 0 | 15 | 0 | 0 | 0 | 0 | 100 |
| Pferd | 0 | 0 | 1 | 0 | 0 | 13 | 0 | 1 | 1 | 81,3 |
| Rabe | 0 | 1 | 0 | 0 | 0 | 0 | 17 | 0 | 0 | 94,4 |
| Schaf | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 100 |
| Schwein | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 15 | 93,8 |
| Producer's Acc. (%) | 84,6 | 81,3 | 91,7 | 100 | 79 | 81,3 | 85 | 80 | 93,8 | 86,2 |

Tabelle 5.8: Konfusionsmatrix bei der Erkennung der Tiergattungsbegriffe, $k = 3$.

| | Biene | Biene (*summ*) | Hahn | Katze | Kuh | Schaf | Schwein | Schwein (*grunz*) | User's Acc. (%) |
|---|---|---|---|---|---|---|---|---|---|
| Biene | 44 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 93,6 |
| Biene (*summ*) | 14 | 26 | 3 | 3 | 1 | 0 | 0 | 0 | 55,3 |
| Hahn | 0 | 0 | 87 | 0 | 0 | 0 | 0 | 0 | 100 |
| Katze | 0 | 0 | 2 | 70 | 2 | 2 | 0 | 0 | 92,1 |
| Kuh | 2 | 0 | 2 | 4 | 58 | 2 | 0 | 0 | 85,3 |
| Schaf | 8 | 1 | 5 | 12 | 7 | 43 | 2 | 1 | 54,4 |
| Schwein | 1 | 3 | 6 | 2 | 0 | 5 | 43 | 0 | 71,7 |
| Schwein (*grunz*) | 1 | 0 | 0 | 0 | 2 | 1 | 0 | 25 | 86,2 |
| Producer's Acc. (%) | 62,9 | 86,7 | 82,9 | 76,1 | 81,7 | 79,6 | 95,6 | 96,2 | 80,3 |

Tabelle 5.9: Konfusionsmatrix bei der Erkennung der Tierlaute, $k = 3$ .

wurde eine Konfusionsmatrix aufgestellt und die Gütemaße daraus abgeleitet. In dieser Arbeit sind in Tabelle 5.8 und 5.9 die Konfusionsmatrizen für $k = 3$ aufgelistet.
Zusammenfassend kann gesagt werden, dass die Resultate sehr erfreulich sind. Insbesondere die ermittelten Kappa-Koeffizienten bei der Körperteilerkennung und Spracherkennung beweisen eine sehr gute Klassifikationsrate der Algorithmen. Insbesondere ist dafür auch ein dementsprechendes Trainingsset erforderlich. Der akribische Aufbau des Trainingssets bewährte sich somit. Des Weiteren wird durch die Einschränkung der Klassen auch eine erhöhte Klassifikationsrate erreicht. Bei komplexeren Anwendungen, die mehrere Objekte bzw. Wörter klassifizieren, wäre die Erkennungsrate somit geringer.

KAPITEL 6

# Schlussbetrachtung

Ziel dieser Arbeit war es, drei Demoanwendungen zu implementieren, um exemplarisch Teilbereiche des Medienverstehens für den Durchschnittsbenutzer verständlich zu machen. Der große Fokus lag auf dem Praxisbezug, um zu verdeutlichen, dass Medienverstehen bereits Einzug in unseren Alltag genommen hat. Um dieses Themengebiet interessant zu gestalten, versuchen die drei Demoanwendungen mittels spielerischem Charakter Interesse des Benutzers zu wecken. In der ersten Applikation ist eine Ähnlichkeitsberechnung zwischen Gesichtern implementiert. Als Referenzdaten sind Bilder prominenter Personen hinterlegt. Dem Benutzer wird sein am ähnlichsten schauender Prominenter ausgegeben. Die zweite Demoanwendung ist ein Beispiel aus der Objekterkennung. Im konkreten Fall muss der Benutzer die am Computer dargestellten Körperteile in die Webcam halten. Die Interaktion soll so schnell wie möglich erfolgen, um einen möglichst hohen Punktestand zu erreichen. Die dritte Applikation ermöglicht die sprachliche Interaktion zwischen Benutzer und Computer. Alle Demoanwendungen wurden in MATLAB implementiert, da in dieser Entwicklungsumgebung unterschiedliche Medientypen verarbeitet werden können. Im Folgenden wird anhand der einzelnen Anwendungen auf die implementierten Algorithmen eingegangen.

## 6.1 Ähnlichkeitsberechnung zu prominenten Personen

Nach den ersten Testdurchläufen zeigte sich, dass die alleinige Verwendung von Template Matching nicht zufriedenstellend funktionierte und deshalb Optimierungen notwendig waren. So wird zunächst mittels Gesichtsdetektion die ungefähre Position des Prominentengesichts in den Aufnahmen ermittelt. Für die Ähnlichkeitsberechnung von Gesichtern wurde Template Matching eingesetzt. Für die Suche des Gesichts wurde die Mikael Nilssons Implementierung verwendet [NNC07]. Diese arbeitet mit SMQT-Features und einem Split-up-SNoW-Klassifikator und ist schnell und zuverlässig. Die Gesichtserkennung ist außerdem auch für die Erstellung des Templates erforderlich. Das Gesicht wird anschließend in der Größe normalisiert, sodass die Anzahl der verschiedenen Skalierungsstufen gesenkt werden kann. Template Matching ist

sehr rechenaufwändig und nicht skalierungsinvariant. Ein vollständiger Verzicht auf Skalierung ist allerdings nicht möglich, da der Algorithmus nicht immer genau denselben Ausschnitt eines Gesichts liefert. Das Template Matching wird in zehn verschiedenen Größen durchgeführt, wobei der Bereich in den Aufnahmen der Prominenten aus Performancegründen auf die Umgebung der Gesichter beschränkt ist. Auch hier wird für die Detektion des Gesichts Mikael Nilssons Methode verwendet. Für die Umsetzung des Template Matchings wurde die normalisierte Kreuzkorrelation benutzt. Schlussendlich war die Performance des Algorithmus zufriedenstellend. Bei 80,65 Prozent der Gesichter lag das erkannte Gesicht des Benutzers in der korrekten Region des Prominentenbildes. Sowohl Skalierung des Gesichts als auch die Position der Augen, des Mundes etc. stimmten überein.
Nochmals betont werden muss, dass diese Demoanwendung spielerischen Charakter hat und somit keine biometrische Ähnlichkeitsmessung verlangt wird. Für den sicherheitstechnischen Einsatz der Ähnlichkeitsberechnung oder Gesichtserkennung ist Template Matching beispielsweise nicht geeignet. In solchen Fällen sind komplexere Systeme erforderlich, die Charakteristika des Gesichts besser abbilden, wie z. B. Eigenfaces. Neuartige Verfahren betrachten Gesichter als Textur eines dreidimensionalen Objekts [CBF03].

## 6.2 Körperteilerkennung

Bei der Körperteilerkennung soll der Benutzer rasch den Anweisungen der Applikation folgen, die gewünschten Körperteile zu erkennen. Das Spiel läuft auf Zeit und der Benutzer bekommt, sollte das korrekte Körperteil erkannt werden, seiner Reaktionsdauer entsprechend Punkte gutgeschrieben. Für die Körperteilerkennung wurden die Aufnahmen der Körperteile mit einem Interest-Point-Detektor bzw. -Deskriptor (im Konkreten mit den SIFT-Features bzw. deren Abwandlung, den PHOW-Features) und anschließend mittels Bag-of-Features-Methode beschrieben. Für die Klassifizierung wurden drei verschiedene Verfahren verwendet. Neben einer aus dem Demo aus *VLFeat* bereits implementierten SVM wurde die Klassifizierung mit einem kNN-Algorithmus und der Naive-Bayes-Methode durchgeführt. Die Idee hinter der implementierten Methode ist, dass relevante Punkte in Bildern detektiert und mit einer geeigneten Methode beschrieben werden. Aussagekräftige Punkte in Bildern sind beispielsweise Eckpunkte. Der SIFT-Detektor erfüllt genau diese Funktion. Die ermittelten Punkte werden anschließend mittels der Bag-of-Features-Methode in Cluster eingeteilt. Für jede Klasse des Trainingssets wird hierbei ein eigener *Bag* erstellt. Anhand dieser Bags ist eine Klassifizierung von Bildern mit unbekanntem Inhalt möglich. Auch hier werden aus dem Bild SIFT-Features ermittelt, welche mit dem Vokabular verglichen werden. Für die Klassifizierung wurden drei Algorithmen verwendet. Beim kNN-Algorithmus entscheiden die $k$ nächsten Nachbarn über die Klassenzugehörigkeit, beim Naive-Bayes-Verfahren wird mit dem Bayestheorem entschieden, als dritte Methode wurde die SVM verwendet. Beim Aufbau des Trainingssets hatten die Bilder ähnliche Hintergründe, was sich schlecht auf die Ergebnisse auswirkte. Schlussendlich wurde ein zuverlässiges Trainingsset aufgebaut, sodass die Klassifikationsrate zufriedenstellend war. Von den drei implementierten Algorithmen schnitt die SVM mit einem Kappa von 0,9847 knapp vor dem Naive-Bayes-Verfahren mit einem Kappa von 0,9728 ab. Der kNN-Algorithmus hat eine deutlich schlechteren Wert mit einem Kappa von 0,717. Die gute Erkennungsrate vor allem bei der

SVM und dem Naive-Bayes-Verfahren ist besonders durch die geringe Anzahl an Klassen als auch an der Eingrenzung des Bildausschnitts erklärbar.

## 6.3 Spracherkennung

Bei der Spracherkennung muss der Benutzer mit dem Computer interagieren, indem er mit den Tierlauten eines vorgegebenen Tieres antwortet. Genauso gut kann auch der Tierlaut vorgegeben sein. Der Benutzer muss in solchen Fällen mit dem Namen der Tiergattung reagieren. Die Interaktion muss schnell erfolgen, da sich die Punktezahl sowohl aus der Schnelligkeit als auch der Korrektheit der Antwort zusammensetzt. Bei der Spracherkennung haben sich die MFCC-Features bewährt. Zunächst muss ein Trainingsset mit Aufnahmen von unterschiedlichen Sprechern aufgebaut werden. Für jede dieser Aufnahmen werden die Features ermittelt. Die MFCC sind spektrale Features und bestehen aus bis zu 45 Koeffizienten. Für die Spracherkennung sind allerdings nur die ersten zwölf erforderlich. Die niedrigen Indizes beschreiben die niederfrequenten Anteile des Audiosignals. Bekanntlich sind ja die hohen Frequenzen die Trägerfrequenzen und haben kaum Informationsgehalt über das Gesprochene sondern wirken sich hauptsächlich nur auf die Tonhöhe aus. Die niederen Frequenzen eines Audiosignals sind diejenigen, die informationstragend sind, wie auch bei der Produktion von Sprache beim Menschen. Die ausgestoßene Luft aus den Lungen bringt die Stimmlippen in Bewegung und ist hochfrequent, die eigentliche Information wird im Rachenraum erzeugt und ist niederfrequent. Die Aufnahme, die klassifiziert werden soll, wird ebenfalls mit den MFCC-Features beschrieben und mit dem Trainingsset verglichen. Die Klassifizierung erfolgt mit dem kNN-Algorithmus. Während der Entwicklung machte sich bemerkbar, dass der Aufbau eines für die Klassifikation zuverlässigen Trainingssets langwierig und schwierig war. Bei Testdurchläufen war die Erkennungsrate zunächst nicht ausreichend. Außerdem sind mehrere Sprecher verschiedener Altersklassen erforderlich. Ein weiteres Problem war die unterschiedliche Nachahmung von Tierlauten. Aus diesem Grund wird der Benutzer in der Applikation darauf hingewiesen, mit den gängigen Tierlautbegriffen zu antworten, und nicht zu versuchen, diese realistisch nachzuahmen. Außerdem wurden die Ergebnisse besser, nachdem die Aufnahmen aus dem Trainingsset zu Beginn und am Ende zurecht geschnitten wurden und keine Nebengeräusche in den Aufnahmen mehr vorhanden waren. Auch bei den Benutzeraufnahmen ist eine gute Qualität und möglichst gute zeitliche Anpassung von Vorteil. Die Sprachinformation im aufgenommenen Signal wird mittels VAD-Algorithmus detektiert und anschließend zurecht geschnitten. Für die implementierte Problemstellung sind MFCC-Features ausreichend, da die Wörter isoliert sind. Bei einem komplexen Spracherkennungssystem wäre es allerdings ratsam, Hidden-Markov-Models zu verwenden.
Zusammenfassend kann gesagt werden, dass Medienverstehen ein sehr weitreichendes Umfeld ist, was auch die drei implementierten Demoanwendungen bewiesen haben. Die Applikationen haben allesamt spielerischen Charakter, die dahinter liegenden Algorithmen und verwendeten Features werden jedoch auch in komplexeren Anwendungen verwendet. Somit konnte gut die praktische Anwendung dem Benutzer vermittelt werden.

# Tabellenverzeichnis

# Literaturverzeichnis


[Alt90] ALTMAN, D. G.: *Practical statistics for medical research*. Bd. 12. Chapman & Hall/CRC, 1990

[An09] AÑORGA, E.: *Development of the Feature Extractor for Speech Recognition*, Univerza v Mariboru, Diplomarbeit, 2009

[Ang06] ANGST, R.: Local Features and Shape for Recognition / Eidgenössische Technische Hochschule Zürich. 2006. – Forschungsbericht

[BB06] BURGER, W. ; BURGE, J. M. *Digitale Bildverarbeitung*. Springer, 2006. – ISBN 9783540309406

[BHT63] BOGERT, B. P. ; HEALY, M. J. R. ; TUKEY, J. W.: The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. In: *Symposium on Time Series Analysis*, 1963, S. 209–243

[Bie87] BIEDERMAN, I.: Recognition-by-components: A theory of human image understanding. In: *Psychological Review* 94 (1987), S. 115–147

[BKB10] BALA, A. ; KUMAR, A. ; BIRLA, N.: Voice Command Recognition System Based on MFCC and DTW. In: *International Journal of Engineering Science and Technology* 2 (2010), Nr. 12, S. 7335–7342

[BM76] BROSNAHAN, L. F. ; MALMBERG, B.: *Introduction to phonetics*. Cambridge University Press, 1976. – ISBN 9780521290425

[BP93] BRUNELLI, R. ; POGGIO, T.: Face recognition: Features versus templates. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (1993), Nr. 10, S. 1042–1052

[Bro11] BROOKES, M.: *VOICEBOX: Speech Processing Toolbox for MATLAB*. `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`. Version: September 2011. – zuletzt abgerufen am 07.10.2012

[CBF03] CHANG, K. ; BOWYER, K. ; FLYNN, P.: Face recognition using 2D and 3D facial data. In: *ACM Workshop on Multimodal User Authentication*, 2003, S. 25–32

[CD07] CUNNINGHAM, P. ; DELANY, S. J.: k-Nearest neighbour classifiers / University College Dublin. 2007. – Forschungsbericht

[CDF+04] CSURKA, G. ; DANCE, C. ; FAN, L. ; WILLAMOWSKI, J. ; BRAY, C.: Visual categorization with bags of keypoints. In: *Workshop on statistical learning in computer vision, ECCV* Bd. 1, 2004, S. 22

[CH09] CHEN, M. ; HAUPTMANN, A.: MoSIFT: Recognizing Human Actions in Surveillance Videos. (2009)

[Dav02] DAVIDSON, I.: Understanding K-means non-hierarchical clustering / Computer Science Department of State University of New York (SUNY), Albany. 2002. – Forschungsbericht

[Der09] DERAGISCH, F.: *Generierung von Sprachmustern für die Spracherkennung*. 2009

[DP97] DOMINGOS, P. ; PAZZANI, M.: On the optimality of the simple Bayesian classifier under zero-one loss. In: *Machine learning* 29 (1997), Nr. 2, S. 103–130

[Eid11] EIDENBERGER, H.: *Fundamental Media Understanding*. Books on Demand GmbH, 2011. – ISBN 9783842379176

[Eis08] EISENBERG, G.: *Identifikation und Klassifikation von Musikinstrumentenklängen in monophoner und polyphoner Musik*. Cuvillier, 2008. – ISBN 9783867278256

[Eis11] EISELE, P.: *Evaluierung und Visualisierung von Interest-Point-Detektoren*, Technische Universität Wien, Diplomarbeit, 2011

[Ell03] ELLIS, D.: *Dynamic Time Warp (DTW) in Matlab*. `http://labrosa.ee.columbia.edu/matlab/dtw/`. Version: September 2003. – zuletzt abgerufen am 07.10.2012

[Fan73] FANT, G.: Speech sound and features. In: *MIT Press, Cambridge* (1973)

[FFAR03] FEI-FEI, L. ; ANDREETTO, M. ; RANZATO, A. M. *Caltech 101*. `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`. Version: 2003. – zuletzt abgerufen am 07.10.2012

[FFP05] FEI-FEI, L. ; PERONA, P.: A bayesian hierarchical model for learning natural scene categories. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 2 IEEE, 2005, S. 524–531

[FTKI92] FUKADA, T. ; TOKUDA, K. ; KOBAYASHI, T. ; IMAI, S.: An adaptive algorithm for mel-cepstral analysis of speech. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing* Bd. 1 IEEE, 1992, S. 137–140

[GBZL07] GROUVEN, U. ; BENDER, R. ; ZIEGLER, A. ; LANGE, S.: Der Kappa-Koeffizient. In: *Deutsche Medizinische Wochenschrift* 132 (2007), Nr. 1, S. 65–68

[GGYM10] GAWALI, B. W. ; GAIKWAD, S. ; YANNAWAR, P. ; MEHROTRA, S. C.: Marathi Isolated Word Recognition System using MFCC and DTW Features ACEEE, 2010

[Gre05] GREMSE, F.: *Skaleninvariante Merkmalstransformation - SIFT Merkmale*. 2005

[Hal11] HALDEWANG, N.: *How to apply Naive Bayes Classifiers to document classification problems*. 2011

[HAYSZ11] HAJEBI, K. ; ABBASI-YADKORI, Y. ; SHAHBAZI, H. ; ZHANG, H.: Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph. In: *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011

[Hel08] HELD, A.: K-Means Clustering for Automatic Image Segmentation. (2008)

[Her09] HERRMANN, M.: *Psychoakustik und Sound-Engineering: Der neue Stellenwert des Hörens in der Corporate Identity*. GRIN Verlag, 2009

[HFMEA11] HACHKAR, Z. ; FARCHI, A. ; MOUNIR, B. ; EL ABBADI, J.: A Comparison of DHMM and DTW for Isolated Digits Recognition System of Arabic Language. In: *International Journal* (2011)

[HJR04] HASAN, M. R. ; JAMIL, M. ; RAHMAN, M. G. R. M. S.: Speaker identification using Mel frequency cepstral coefficients. In: *3rd International Conference on Electrical & Computer Engineering* Bd. 1, 2004, S. 4

[Jan] JANG, J.-S. R.: *Audio Signal Processing and Recognition*. `http://mirlab.org/jang/books/audioSignalProcessing/index.asp`. – zuletzt abgerufen am 07.10.2012

[JG09] JUAN, L. ; GWUN, O.: A comparison of SIFT, PCA-SIFT and SURF. In: *International Journal of Image Processing* 3 (2009), Nr. 4, S. 143–152

[JLYS07] JIN, Z. ; LOU, Z. ; YANG, J. ; SUN, Q.: Face detection using template matching and skin-color information. In: *Neurocomputing* 70 (2007), Nr. 4, S. 794–800

[JNY07] JIANG, Y. G. ; NGO, C. W. ; YANG, J.: Towards optimal bag-of-features for object categorization and semantic video retrieval. In: *Proceedings of the 6th ACM international conference on Image and video retrieval* ACM, 2007, S. 494–501

[Käs05] KÄSTER, T.: Intelligente Bildersuche durch den Einsatz inhaltsbasierter Techniken. (2005)

[Kre04] KREMER, G.: *Untersuchung der Sprecherindividualität höherer Formanten*. Azenbergstr. 12, 70174 Stuttgart, August 2004

[KS04] KE, Y. ; SUKTHANKAR, R.: PCA-SIFT: A more distinctive representation for local image descriptors. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 2 IEEE, 2004, S. II–506

[L+00] LOGAN, B. u. a.: Mel frequency cepstral coefficients for music modeling. In: *International Symposium on Music Information Retrieval* Bd. 28, 2000, S. 5

[Lew95] LEWIS, J. P.: Fast Template Matching. In: *Vision Interface* Bd. 95, 1995, S. 120–123

[LLT02] LIPEIKA, A. ; LIPEIKIENE, J. ; TELKSNYS, L.: Development of isolated word speech recognition system. In: *Informatica* Bd. 13, 2002, S. 37–46

[LN10] LAMA, P. ; NAMBURU, M.: *Speech Recognition with Dynamic Time Warping using MATLAB*. 2010

[Loh10] LOHNINGER, H.: *Fundamentals of Statistics*. `http://www.statistics4u.com/fundstat_eng/ee_kmeans_clustering.html`. Version: September 2010. – zuletzt abgerufen am 07.10.2012

[Low99] LOWE, D.: Object Recognition from Local Scale-Invariant Features. (1999)

[Low04] LOWE, D. G.: Distinctive image features from scale-invariant keypoints. In: *International journal of computer vision* 60 (2004), Nr. 2, S. 91–110

[Mac96] MACHELETT, K.: *Das Lesen von Sonagrammen V1.0 - Kapitel II*

[MAHW01] MU, X. ; ARTIKLAR, M. ; HASSOUN, M. H. ; WATTA, P.: Training algorithms for robust face recognition using a template-matching approach. In: *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on* Bd. 4 IEEE, 2001, S. 2877–2882

[Mat] MATTEUCCI, M.: *K-Means Clustering*. `http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html`. – zuletzt abgerufen am 01.03.2012

[MB12] MULANI, R. N. ; BARGE, S.: Local Feature Detectors. In: *Third Biennial National Conference on Nascent Technologies* (2012)

[MBE10] MUDA, L. ; BEGAM, M. ; ELAMVAZUTHI, I.: Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. In: *Journal of Computing* 2 (2010)

[Mit02] MITTELBACH, M.: Untersuchung und effiziente Implementierung von Methoden und Architekturen für integrierte aktive Sehsysteme / Technische Universität Dresden. 2002. – Forschungsbericht

[Mit10] MITCHELL, M. T. *Generative and discriminative classifiers: Naive bayes and logistic regression*. `http://www.cs.cmu.edu/~{}tom/mlbook/NBayesLogReg.pdf`. Version: 2010. – zuletzt abgerufen am 07.10.2012

[MS04] MIKOLAJCZYK, K. ; SCHMID, C.: Scale & affine invariant interest point detectors. In: *International journal of computer vision* 60 (2004), Nr. 1, S. 63–86

[Mül07] MÜLLER, M.: *Information retrieval for music and motion*. Springer, 2007. – ISBN 9783540740476

[Nie83] NIEMANN, H.: *Klassifikation von Mustern*. Springer, 1983. – ISBN 3540126422

[Nie04] NIELS, R.: Dynamic Time Warping: An intuitive way of handwriting recognition? In: *Artificial Intelligence* (2004)

[NJT06] NOWAK, E. ; JURIE, F. ; TRIGGS, B.: Sampling strategies for bag-of-features image classification. In: *Computer Vision–ECCV 2006* (2006), S. 490–503

[NNC07] NILSSON, M. ; NORDBERG, J. ; CLAESSON, I.: Face Detection using Local SMQT Features and Split Up SNoW Classifier. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* Bd. 2 IEEE, 2007, S. II–589

[PK08] PFISTER, B. ; KAUFMANN, T.: *Sprachverarbeitung: Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer, 2008

[PM09] POMPINO-MARSCHALL, B.: *Einführung in Die Phonetik*. Walter de Gruyter, 2009 (De Gruyter Studienbuch). – ISBN 9783110224801

[Pom97] POMPE, B.: *Die Spracherzeugung beim Menschen*. 1997

[Ris01] RISH, I.: An empirical study of the naive Bayes classifier. In: *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence* Bd. 3, 2001, S. 41–46

[RM03] RATH, M. T. ; MANMATHA, R.: Word Image Matching Using Dynamic Time Warping. 2 (2003), S. II–521

[SMB00] SCHMID, C. ; MOHR, R. ; BAUCKHAGE, C.: Evaluation of interest point detectors. In: *International Journal of computer vision* 37 (2000), Nr. 2, S. 151–172

[Ste93] STENGELE, R.: *Kartographische Mustererkennung durch template matching*. Eidgenössische Technische Hochschule Zürich, Inst. für Geodäsie und Photogrammetrie, 1993

[SVS11] SMITA, T. ; VARSHA, S. ; SANJEEV, S.: Face Detection using Combined Skin Color Detector and Template Matching Method. In: *International Journal of Computer Applications* 26 (2011), Nr. 7, S. 5–8

[TC02] TZANETAKIS, G. ; COOK, P.: Musical genre classification of audio signals. In: *IEEE transactions on Speech and Audio Processing* 10 (2002), Nr. 5, S. 293–302

[TM08] TUYTELAARS, T. ; MIKOLAJCZYK, K.: Local invariant feature detectors: a survey. In: *Foundations and Trends® in Computer Graphics and Vision* 3 (2008), Nr. 3, S. 177–280

[Tol10] TOLUNAY, A.: *Text-Dependent Speaker Verification Implemented in Matlab Using MFCC and DTW*, Linköping University, Diplomarbeit, 2010

[TSK06] TAN, N. P. ; STEINBACH, M. ; KUMAR, V.: Cluster analysis: Basic concepts and algorithms. In: *Introduction to data mining* (2006), S. 487–568

[VF08] VEDALDI, A. ; FULKERSON, B.: *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. 2008

[WC05] WU, Z. ; CAO, Z.: Improved MFCC-Based Feature for Robust Speaker Identification. In: *Tsinghua Science & Technology* 10 (2005), Nr. 2, S. 158–161

[WCG03] WALLRAVEN, C. ; CAPUTO, B. ; GRAF, A.: Recognition with local features: the kernel recipe. In: *Ninth IEEE International Conference on Computer Vision* IEEE, 2003, S. 257–264

[Web04] WEBER, B.: *Auditive Wahrnehmung und Sprachentwicklung*, Leopold-Franzens-Universität Innsbruck, Diplomarbeit, 2004

[WZ01] WIEDENBECK, M. ; ZÜLL, C.: Klassifikation mit Clusteranalyse: Grundlegende Techniken hierarchischer und K-means-Verfahren. In: *ZUMA how-to-Reihe* 10 (2001), S. 1–18

[YG08] YANG, Z. L. ; GUO, B. L.: Image mosaic based on SIFT. In: *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on* IEEE, 2008, S. 1422–1425

[ZWLC00] ZHEN, B. ; WU, X. ; LIU, Z. ; CHI, H.: On the importance of Components of the MFCC in speech and speaker recognition. In: *Sixth International Conference on Spoken Language Processing*, 2000