AALBORG UNIVERSITY
INSTITUTE OF ELECTRONIC SYSTEMS
DEPARTMENT OF COMMUNICATION TECHONOLOGIES

Fredrik Bajersvej 7 ▪ DK-9220 Aalborg East Phone 96 35 80 80

TITLE:	Automated Lip Synchronization of Animated Characters
THEME:	Acquisition and Description of Information
PROJECT PERIOD:	6th Semester. From February 2000 to June 2000
PROJECT GROUP:	622

Participants:
Tonny Gregersen
Henrik Harsfort
Lars Chr. Hausmann
Peter Korsgaard
Michael Nielsen
Robert Stepien
Claus Thomsen

Supervisors:
Ove Andersen
Tom Brøndsted

Abstract

This report deals with development of a system for Automatic Lip Synchronization of Animated Characters (ALSAC). The development is done in cooperation with Interactive Television Entertainment Aps (ITE). After being trained the system is able to determine a sequence of visemes and corresponding time stamps, which matches a given speech signal. The system is based on visemes and not on phonemes which is the normal approach.
For training ALSAC utilizes a clustering mechanism, incorporating the Dynamic Time Warping (DTW) algorithm. The recognition is implemented using the One-Pass algorithm.
Modularity is obtained by use of the Strategy Design Pattern. This is done to ensure the possibility to change algorithms (strategy) of the various parts independently of the rest of the system.
The ALSAC-system is a C++ library which ITE can use as a building block in their products.
It is concluded that a system performing the intended task of classifying the contained sequence of visemes and time stamps in speech signals has been implemented correctly. However the test shows that ALSAC does not produce an accuracy score as high as required. This can partly be explained by the fact, that the training data used were not optimal. It is expected that with better training data and adjusted system parameters the performance of ALSAC can be increased. ALSAC is believed to be able to accelerate the currently manual animation process at ITE, even though the system's accuracy score was not found to be optimal in the system's present state.

PUBLICATIONS:	12
NUMBER OF PAGES:	122
FINISHED:	7th of June 2000

AALBORG UNIVERSITET
INSTITUT FOR ELEKTRONISKE SYSTEMER
AFDELING FOR KOMMUNIKATIONS TEKNOLOGI

Fredrik Bajersvej 7 ▪ DK-9220 Aalborg Øst Telefon 96 35 80 80

TITEL:	Automated Lip Synchronization of Animated Characters
TEMA:	Opsamling og beskrivelse af information
PROJEKTPERIODE:	6. semester, februar 2000 til juni 2000
PROJEKTGRUPPE:	622

Deltagere:
Tonny Gregersen
Henrik Harsfort
Lars Chr. Hausmann
Peter Korsgaard
Michael Nielsen
Robert Stepien
Claus Thomsen

Vejledere:
Ove Andersen
Tom Brøndsted

Synopsis

Denne rapport omhandler udviklingen af et system til automatisk synkronisering af animerede figures mundstillinger (ALSAC). Udviklingen er foregået i samarbejde med Interactive Televison Entertainment Aps (ITE). Efter optræning kan systemet labelere et talesignal med en sekvens af visemer og tilhørende tidspunkter. Systemet er baseret på genkendelse af visemer og ikke fonemer, hvilket er den mest almindelige tilgang.
Til træning anvender ALSAC clustering, der indkorporerer Dynamic Time Warping (DTW). Genkendelsen er implementeret vha. One-Pass algoritmen. Modulariteten opnås vha. et Strategi Design Pattern. Herved er det muligt at udskifte algoritme (strategi) for forskellige dele uafhængigt af resten af systemet.
Selve programmet er udformet som et C++ library, hvilket gør det muligt for ITE at benytte ALSAC som en byggesten i deres produkter.
Det konkluderes, at det udviklede system udfører den ønskede opgave, bestående i at udtrække visemsekvensen og tilhørende tider fra et talesignal, korrekt. Derimod har genkendelsesnøjagtigheden vist sig at være mindre end krævet. Dette kan delvist forklares udfra det faktum, at træningsdata ikke er optimale. Det forventes at ALSAC med bedre træningsdata og finjusterede systemparametre kan øge sin effektivitet. Det forventes at ALSAC er i stand til at accelerere animationsprocessen hos ITE, selvom systemets genkendelsesscore ikke er optimal i dets nuværende tilstand.

OPLAGSTAL:	12
SIDEANTAL:	122
AFSLUTTET:	7. juni 2000

Preface

This report is the result of a 6th semester Informatics project at the University of Aalborg. The project is made by group 622, spring 2000. The theme description for the semester is "Acquisition and Description of Information".

The report documents the construction of a system for Automated Lip Synchronization of Animated Characters. The report contains an outline of different theories used in speech recognition systems. Furthermore it documents the analysis, design and implementation of a speech recognition system, which is based on visemes. It should be noted that the theory outlined in chapter 2 is not meant as thorough explanation of the theoretical models used in speech recognition. The models used in this project are explained in depth in the appendices B, C and E.

The source code for the developed system is available at http://www.kom.auc.dk/~jacmet/inf6/ until the 1st of July 2000. Following this date the source code will only be available through contact with the project group via email at 00gr622@kom.auc.dk.

Throughout the report citations are in the form "[authors, year of publication, specific page]" when used in context and in the form "[authors, year of publication]" when otherwise used. Please note that "specific page" is only included when relevant. Footnotes are numbered in succession within each chapter and are only used when elaboration is needed. Figures and tables are likewise numbered in succession within each chapter and references are made in the form of X.Y, where X is the chapter number and Y is a successive number.

The project group wishes to thank the staff from Interactive Television Entertainment Aps for their interest and support. Furthermore the project group wishes to thank Søren Østergaard from Gekko Webdesign for designing the cover.

Aalborg University, 7th of June 2000.

Tonny Gregersen	Henrik Harsfort
Lars Chr. Hausmann	Peter Korsgaard
Michael Nielsen	Robert Stepien

Claus Thomsen

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Objective

According to the theme of this semester "Acquisition and Description of Information" the objective of the semester is according to [**?**]:

- to give the students an understanding of how physical measurable data can be transformed into abstract information.
- to enable the students to make a synthesis of theories, methods and techniques for acquisition of signals, signal to symbol transformation, and to generate an abstract representation of the relevant information.
- to enable the students to use theories, methods, and techniques to design and implement systems for information acquisition and processing.

In order to be able to evaluate the above mentioned objectives the Study Board has issued some requirements concerning the contents of the project:

> *On this semester the focus is to gather, represent and process abstract knowledge. With a starting point in a specific problem description the student should work with gathering the necessary information and represent it on abstract form. This involves that the student obtains knowledge on how the information is extracted from the information carrying signals, how this information can be represented as symbols and how these symbols can be processed. The interesting information would typically originate from physical signals but can also be available in another forms.*
> [**?**]

In order to satisfy the objectives of this semester the project group has decided to develop a system for automated lip synchronization together with Interactive Television Entertainment Aps (ITE). ITE wants to use this system for complete automated synchronization of their animated cartoon characters or at least be able to use it as a tool to help them in the current process. In essence the system should be able to give the matching mouth positions and time stamps as output to a given input speech signal.

The system to be developed is primarily to be used in the development of computer games all though ITE would also like to use the implementation for Real-Time purposes such as TV game shows.

The next section will describe how ITE currently handles the task of lip synchronization and what the system to be developed should be capable of.

1.2 The System

1.2.1 ITE's Current Method

The following is based on information on from a meeting with ITE the 15th of march 2000. At ITE lip synchronization of animated characters is presently done by hand. When a new computer game is being developed, a number of employees listen to the recorded speech and then manually decides which of the mouth positions best fits the given speech. Using this process it takes one person approximately one hour to process 1 MB of speech recorded in 22kHz 8 bit mono.

In order to aid their employees during the lip synchronization ITE has originally developed a tool called Out-Of-Sync (OOS) which is capable of visualizing the synchronization.

The OOS program uses a rather simple format to describe how the mouth positions should be during the animation. This format only changes the mouth positions 8.3 times per second, which means that the time a character has the same mouth position is at least 120 ms. With this low frame rate the resulting animation will some times look rather rigid. Furthermore the OOS format has a rather limited scheme for mouth positions - there is currently only about 12-15 mouth positions available. This can be used in ordinary 2-dimensional (2D) games, though the animation will be rough, but is not detailed enough for 3D-animation. Thus ITE wants to develop a greater set of mouth positions in order to use OOS for 3D-animation.

The greatest problem is however not with the OOS program, but primarily with the fact that all of the synchronization has to be done manually. It would cause a significant cut in expenses if this procedure could be automated. It need not be a perfect output, it would still be a big improvement if just some of the mouth positions would be correct using an automated tool, hereby making it easier for the animators.

1.2.2 The Desired System

ITE wants a system that is able to automatically generate the matching mouth positions from a given speech signal. The system should ideally be independent of speaker, language and animated character. Furthermore the system should be ready for integration with the a program that ITE is currently developing to replace the visualization part of OOS.

1.3 The Report Structure

The following report is made using a combination of Structured Analysis & Design as defined in [**?**] and Object Oriented Analysis & Design as defined in [**?**].

The report will begin with a brief introduction to the speech recognition theory. This will be followed by an outline of the requirements of the system to be developed. These requirements will be presented in a requirement specification as outlined in [**?**]. The structured approach has been chosen in order to ensure mutual understanding between ITE and the project group of the system to be developed. This is particularly important in this project as the developed system is to be incorporated in a greater system under development by ITE. Following the requirement specification the speech recognition theory and the requirements will come together to form a detailed view of how the system is to be developed. The discussion concerning how this is done will be outlined in the "Choice of method" chapter. After this chapter follows a test specification, which describes how the system is to be tested.

Once the system has been outlined the actual system development is done using Object Oriented Analysis & Design (OOAD). The object oriented approach has been chosen due to the fact that the system ITE is developing is to be designed and implemented in an object oriented environment (C++). The documentation of the system development process will be presented in the 3 chapters: Analysis, Design, and Implementation and as this project is not a project in Object Oriented Analysis & Design only the crucial parts of the process will be documented.

Finally the developed system will be tested as specified in the test specification. Following this the results will be discussed in the "Conclusion" chapter. It should be noted that in addition to the theory presented within the report some subjects will be elaborated in the appendices.

The actual division of the report into chapters is shown in figure 1.1. Each box in the figure illustrates a chapter in the report.

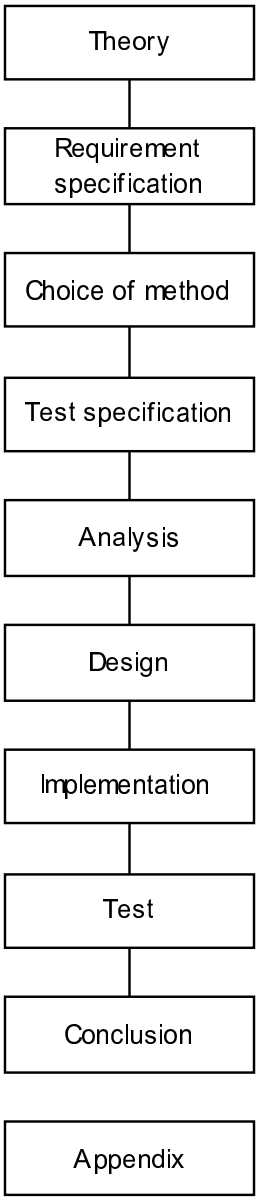

Figure 1.1: Report structure.

In the following the chapters and their objectives will be outlined:

Theory

The primary concern of the theory chapter is to bring the reader to a basic understanding

of the theories used in most speech recognition systems. Furthermore the objective is to give the foundation for the choice of solution that has to be made in order to implement the system.

Requirement Specification
The purpose of the requirement specification is to determine the requirements for the system based on the needs and wishes of ITE and the project group.

Choice of Method
The purpose of the choice of method chapter is to present a detailed view of the system to be developed and discuss the choices made based on the Theory and Requirement specification chapters.

Test Specification
The test specification specifies the different tests which will have to be made in order to test whether the developed system meets the requirements specified in the requirement specification.

Analysis
The purpose of the analysis chapter is to gain an insight in the elements, use patterns, function and interfaces of the desired system.

Design
The purpose of the design chapter is to gain an insight in the structure of the analyzed system and its processes, classes and interfaces.

Implementation
The purpose of the implementation chapter is to describe crucial parts of the implementation of the designed system.

Test
The objective of the test chapter is to describe the tests that the project group has performed to make sure the implemented system acts according to the requirements.

Conclusion
The objective of the conclusion chapter is to outline the results for this project. Furthermore it is evaluated whether the project group has met the requirements for the report set by the Study Board.

Appendix
The objective of the appendices is to elaborate on theoretical subjects presented in the theory chapter. Furthermore additional details on various aspects of the system are covered.

CHAPTER 2

Theory

This chapter will give the reader a brief introduction to the basic theories of speech recognition. Futhermore it will help determine which method or methods is to be used in this project.

2.1 Overview

The system to be developed will consist of the 2 parts: preprocessing and recognition. The input to the system is a speech stream and it produces a viseme stream as output. A block diagram of the system can be seen in figure 2.1.

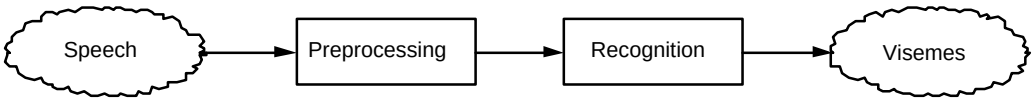

Figure 2.1: Block diagram of the system.

In the following sections the basic theory behind creating a system for speech recognition will be discussed. The subject of this discussion is as visualized in figure 2.1 and the general structure of the discussion is as follows:

Speech
Since the system to be developed is essentially a system for recognizing speech the fundamentals of human speech production and perception are presented in section 2.2 and 2.3. A general description of visemes is included in the speech perception section.

Preprocessing
In order to make a reliable comparison between two speech signals a number of characteristics should be extracted from the signals. The concept is to compare the characteristics (features) of each signal, instead of comparing the signals directly. Thus finding appropriate and robust features is extremely important at this stage. Two approaches which can be used to make the preprocessing is described in section 2.4.

Recognition
The features from the preprocessing are used to determine which of the visemes (mouth positions) that best matches the current speech input. Four different approaches to recognition are described in section 2.5.

2.2 Speech Production

In order to achieve a basic understanding of the theory used in systems for speech recognition it is useful to gain an understanding of the basic properties of speech production. The basic theory behind speech production is described in the following section.

2.2.1 The Vocal Tract

Human speech is produced by pressing air out of the lungs. The air is pressed through the vocal tract hereby producing sounds at the other end of the vocal tract.

The vocal tract begins at the vocal cords (glottis) in the throat and ends at the lips and nostrils. It consist of the pharynx (the throat cavity) and the mouth (oral cavity). The vocal tract is shown in figure 2.2.

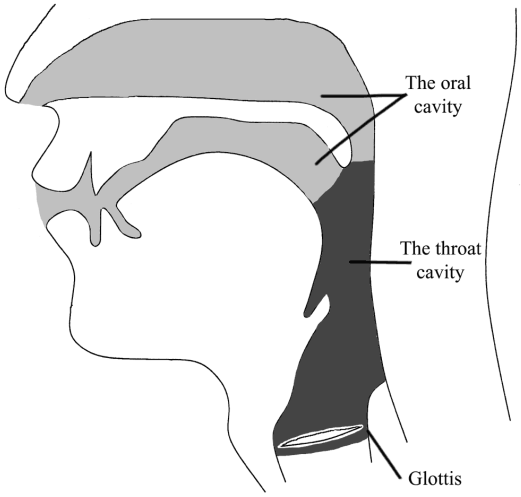

Figure 2.2: The vocal tract.

The Pharynx

The throat cavity (pharynx) is a tube leading from the wind pipe (trachea) to the oral cavity. The vocal cords (glottis) are placed at the bottom of the throat cavity (pharynx) (see figure 2.2). These are the source of the produced sounds in the speech. The muscle force pushes the air out of the lungs and through the bronchi (bronchiolus) and wind pipe (trachea). When the air flows by the vocal cords (glottis) a sound is made.

It is custom to divide speech into three categories according to the state of the vocal cords (glottis). The first speech-category is unvoiced speech in which the vocal cords (glottis) are relaxed. Unvoiced speech is either made by air flowing past the relaxed vocal cords (glottis) hereby creating turbulent unvoiced sounds. It can also be made by a build up of pressure behind a total closure in the vocal tract which causes a brief transient sound e.g. /p/ or /b/. This is so due to the pressure being suddenly and abruptly released when the closure is opened. The second speech-category is voiced speech where the vocal cords (glottis) are tensed. Voiced speech is created by air flowing past the tense vocal cords (glottis) thereby making them vibrate periodically. Sound produced is this way can be distinguished by the frequencies at which vocal cords (glottis) resonates. Their resonant frequencies are also known as formants. The final category is silence where no speech is produced.

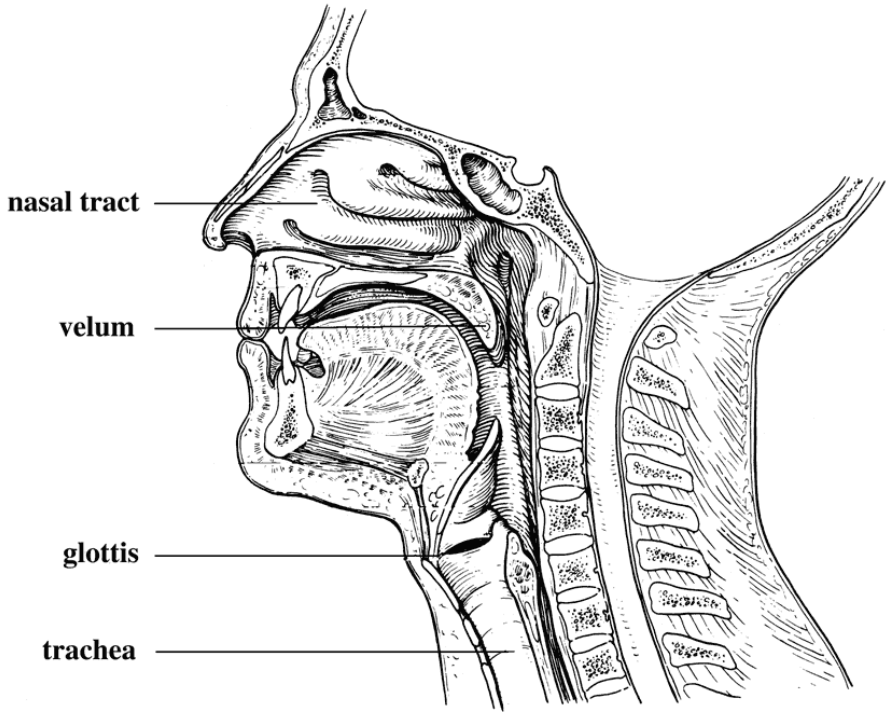

Figure 2.3: Mid-sagittal plane of the human vocal apparatus [after ?, p. 186].

The Mouth

The mouth consists of the mouth cavity and the nasal tract.

The mouth cavity begins at the upper part of the pharynx and ends at the lips. The position of the lips, jaw, and the tongue alters the sound coming from the pharynx.

The nasal tract is a cavity that begins at the velum (at the upper part of the pharynx and the back of the mouth cavity) and ends at the nostrils. When the velum is lowered the sounds produced in the pharynx are allowed to enter the nasal cavity adding nasal sounds in the final speech signal.

2.2.2 Phonemes

Speech is produced as a sequence of sounds. These sounds (or phones) are distinguished by the difference in their spectral contents. In vowels this difference is primarily detected by the spacing and amplitude of the resonance frequencies (or formants) created due to the voiced nature of these signals. Since consonants can be both voiced and unvoiced distinguishing between these are however not that simple. The only way distinguishing between different consonants is to look at changes to their spectral contents.

At a higher level the phones of a given speech signal can be combined to form a sequence of units that makes more perceptive sense. These units are called phonemes and are determined by the following conditions of the vocal tract:

- The state of the vocal cords
- The position of the lips
- The position of the jaw
- The position of the tongue
- The state of the velum

The length of each phoneme can differ in time. The shortest duration of phonemes consisting of one phone is between 5 and 100 ms. During this time the vocal tract parameters are considered approximately stationary. It should be noted that phonemes differ from language to language and so does the number of phonemes.

2.3 Speech Perception

When modeling a system for speech recognition it is useful to know how humans perceive speech. This can be done using only the speech signal (sound perception), but it can also be done by combining the speech with the visual image of the talking mouth (visemes). The objective of the following section is to present the fundamentals of both auditory and visual sound perception.

2.3.1 Sound Perception

In order to understand how humans perceive sounds it is useful to know how the human ear is constructed.

The ear is normally divided into three regions. These are called the outer ear (auris externa), the middle ear (auris media), and the inner ear (auris interna). The three regions are shown in figure 2.4.

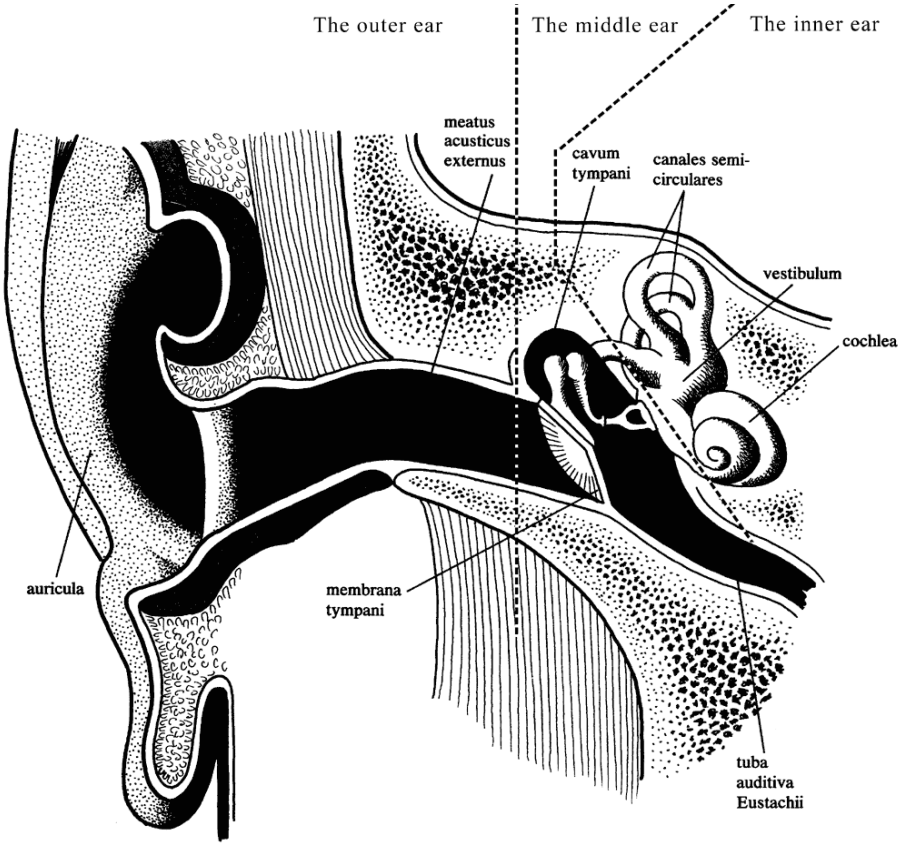

Figure 2.4: The three parts of the human ear [after ?, p. 409].

The Outer Ear

The outer ear consists of the auricula and the external canal (meatus acusticus externus). The auricula guides the soundwaves through the external canal to the ear drum in the inner ear.

The Middle Ear

The middle ear consists of the ear drum (membrana tympani) and the drum cave (cavum tympani) which contains the hammer (malleus), the anvil (incus), and the stirrup (stapes). The hammer is attached to the ear drum and touches the anvil which is attached to the stirrup. The stirrup is attached to fenestra ovalis in the inner ear.

The soundwaves reach the ear drum through the external canal. They cause the ear drum to move hereby moving the hammer, anvil, and the stirrup. The hammer and anvil amplifies the sound waves due to their construction and causes the stirrup and fenestra ovalis to move. Another amplification factor is caused by the scale between the ear drum (big surface) and the stirrups surface connected to fenestra ovalis (little surface) (see figure 2.4).

The Inner Ear

The inner ear consists of bone canals filled with liquid. These canals are often referred to as the labyrinth. It consists of three parts; the cochlea, canales semicirculares, and vestibulum. The cochlea is used for hearing and the two others are used for balance.

The cochlea looks like a snail shell (see figure 2.4 and 2.5). The spiral staircase twists 2.5 time around a bone pillar and contains three corridors. The upper and the lower corridors are connected at the end of the spiral staircase. The inner corridor on the other hand is closed. It contains about 25000 thin hair cells from the beginning of the inner corridor to the end.

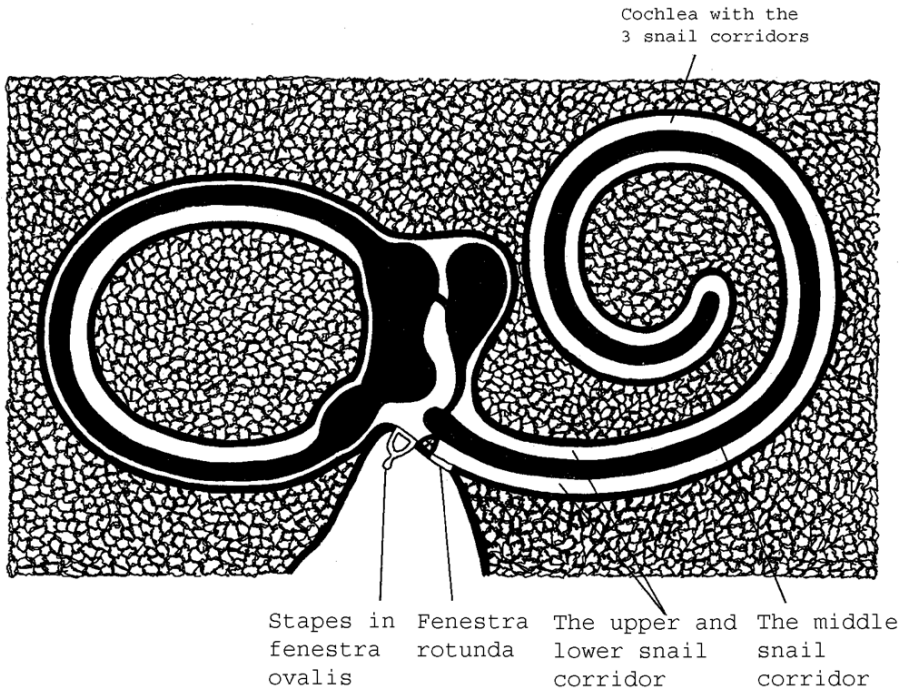

Figure 2.5: The architecture of the cochlea. Note that the corridors ought to twist 2.5 times [after ?, p. 411].

When fenestra ovalis is moved by the stirrup it produces a wave through the liquid in the cochlea. The wave spread from the bottom of the snail shell through the upper corridor and down through the lower corridor to fenestra rotunda. Fenestra rotunda is an elastic membrane that dampens the reflection of the wave. The wave transfer from the upper corridor to the lower corridor causes a wave motion in the middle corridor which reaches its maximal oscillation where the resonance frequency is identical to the sound frequency. The hair cells at this point will begin to swing and the nerve attached to the hair cells will send nerve impulses to the brain. The inner corridor is

shown in figure 2.6. An interesting remark is that low frequencies will cause the hair cells in the lower part to vibrate and high frequencies will cause hair cells in the upper part to vibrate.

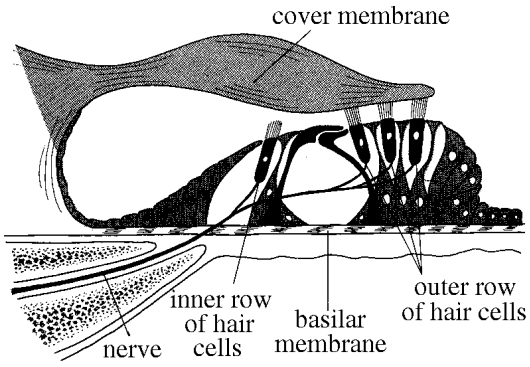

Figure 2.6: A part of the inside of the middle corridor [after ?, p. 412].

Modeling the Ear

The ear is used as a model for auditory systems, because it is assumed that doing so will cause the system to inherit some of the advantages of the ear (among other things immunity to noise and reverberation).

The human ear is able to hear sounds in the range from 20Hz-20kHz. Due to the construction of the ear it acts as a logarithmic filter. In effect this means that humans are better at detecting small frequency variations in the lower range of the spectrum, because it is as difficult to hear the difference between sounds at 200 and 300 Hz as it is to hear the difference between sounds at 2000 and 3000 Hz.

2.3.2 Visemes

Visemes refer to what can be seen when a person speaks. That is the lips, teeth, and possibly the tongue depending on how detailed the visemes are and how much the speaker articulates.

Visemes are important for human speech recognition, because seeing the movement of the mouth can help people understand what is being said. Due to the fact that humans have learned how the mouth should appear during speech it would seem unnatural if the mouth in animations does not follow the usual movements. This is the main reason for wanting a correct stream of visemes in animations.

Visemes in Animation

In general visemes can be described using mere two parameters. The roundness and the openness of the mouth. Using these two parameters one can describe the necessary visemes for an animation. The two parameters are illustrated in figure 2.7.

The visemes are affected by the phonemes, which are differ depending on how the vocal tract is shaped. Not all phonemes affect the visemes to the same degree. Basically, vowels affect both the roundness and openness of the lips, whereas consonants only affect the openness. Thus, the vowels affect the surrounding consonants in a syllable. The visemes for the consonants will be shaped as the vowel in the syllable. Table 2.1 shows examples of how vowels affect consonants.

The duration of the visemes is also different. When animating speech at normal rate the most apparent visemes are the closed ones compared to the rest. They are usually the shortest

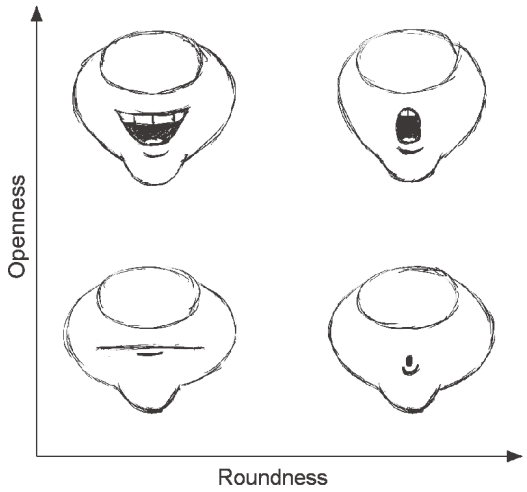

Figure 2.7: Visemes have two major parameters. The lips can be rounded or unrounded and closed or opened. Any low-detailed viseme can be placed in this map.

Phoneme	Example	Explanation
l	reel	Viseme for /l/ depends on the previous viseme.
t	tool	Viseme for /t/ depends on the next viseme.
h	high	The next phoneme defines the viseme for the /h/
s	soothsayer	The /oo/ affects the previous as well as the next viseme.
s	wise	Viseme for /s/ depends on the previous viseme.
z	zebra	Viseme for /z/ depends on the next viseme.
h	sahara	The next phoneme defines the viseme for the /h/
s	sigh	The /i/ affects the previous as well as the next viseme.

Table 2.1: Examples of context sensitive phonemes.

visemes, too. Thus, it is important that the closed lip visemes (such as /m/, /b/, and /p/) are synchronized correctly in order to create a convincing animation [**?**].

2.4 Preprocessing

This section will describe the preprocessing of a sampled speech signal, so that it may be used in a recognition scheme. More precisely this section will deal with the aspect of extracting features from a speech signal. It is presented through a brief summary of the two methods mostly used: the Bank-Of-Filter method and the Linear Predictive Coding (LPC) model (see [**?**] for more information).

It should be noted that it is not the aim of the following text to give a precise mathematical representation of the methods, but merely to serve as an introduction to the subject.

2.4.1 Feature Extraction

The general principle of feature extraction is to identify a simple measurable entity a so-called feature which accurately characterizes a given object. Considering that the object is a sampled speech signal of a given length, two measurable entities comes to mind:

- The spectral content of the signal
- The energy of the signal

However in order to use these measurements a number of issues will have to be considered. These issues are primarily regarding extraction of a feature which is both simple and robust.

The problem of creating a simple and parsimonious measurement can to a certain degree be solved by looking at the properties of the speech signal. In section 2.2.2 it was stated, that a speech signal could be considered approximately stationary over a short period of time (between 5 and 100 ms). By using this fact to divide the sampled speech signal into a sequence of frames of between 15 and 45 ms the number of feature necessary can be greatly reduced. This will create a significant simplification of the signal without loss of information. It should be noted that the limit on 45 ms is due to the fact it is phones and not phonemes that are to be considered approximately stationary spectrally speaking. This and the fact that phonemes can consist of a series of phones makes it highly unlikely that the signal would be approximately stationary for over 45 ms (see [**?**] for more information).

Finally there is the general problem of extracting robust features that accurately characterizes the speech signal. There are a number of issues that must be considered in order to solve this problem, but all of them are connected with the basic properties of the speech signal.

First and foremost there is the problem of characterizing the speech signal with the full signal spectrum at a given time. This approach will result in a huge amount of redundant data and only representing the actual frequency band of speech (the vocal spectrum) should be considered.

Another issue that must be considered in order to accurately represent the vocal spectrum is the distribution of information within a speech signal. In section 2.3.1 a model of the human ear was presented and it is worth noticing that the spectral envelope was perceived on an approximation to a log-scale meaning that the spectrum between 200-300Hz and 2000-3000Hz will be of equal importance.

Reflecting upon past discussions it seems evident that a meaningful measurement characterizing a given speech signal should consist of a sequence of frames with the vocal spectrum of each frame being represented on a log-scale. This feature is however not sufficient in a recognition scheme because of the need to know the energy in the signal at a given time in order to estimate

the start and endpoint of a given entity within the speech signal (e.g. a single word). Therefore the ideal feature should be a combination of both the energy and spectral content of the signal. The following section shows how this combination can be achieved.

2.4.2 Feature Extraction Methods

As mentioned in the beginning of this section, there are two commonly used approaches for extracting features from speech signals: the Bank-Of-Filters model and the LPC model.

The Bank-Of-Filters Model

The Bank-Of-Filters or filter-bank model is characterized by the use of Q bandpass-filters covering the frequency range of the speech signal as viewed in figure 2.8. The output of the ith bandpass filter $X_n(e^{j\omega_i})$ is the short-time spectral representation of the signal $s[n]$ at time n seen from the ith bandpass filter with center-frequency ω_i and the length M_i.

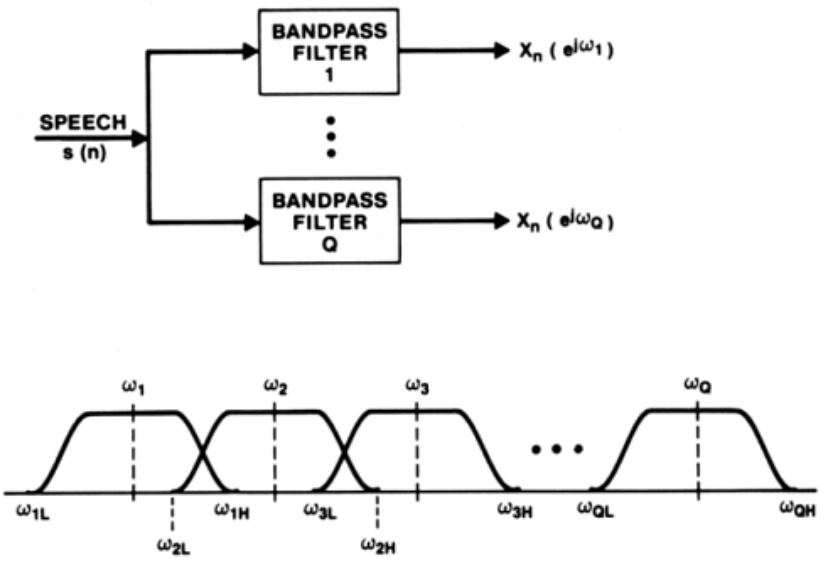

Figure 2.8: Bank-of-Filters analysis model [?, p. 72].

The filtering process can be described as:

$$s_i[n] = s[n] * h_i[n] \qquad \text{where } 1 \leq i \leq Q \tag{2.1}$$

$$= \sum_{m=0}^{M_i-1} h_i[m]s[n-m] \tag{2.2}$$

The purpose of dividing the signal in this way is to extract the spectral contents of the speech signal in a given frequency band for a duration of M_i samples. The energy of the signal in each of the frequency bands can then be found and the measurement thereby fulfills the requirements for a feature to represent both the energy of the signal and its spectral content.

It should be noted that the filters in general overlap to ensure full spectral coverage which otherwise would be impossible because of the frequency response associated with the use of bandpass-filters.

When choosing which filters to use there are two choices:

- IIR-filters
- FIR-filters

IIR can be implemented in simple and efficient structures, but they have a non-linear phase characteristic which is unfortunate as it would cause the signal to be distorted. FIR-filters can achieve linear phase without compromising the ability to approximate ideal magnitude. However FIR filters have the disadvantage of being computational demanding in implementation, although a Fast Fourier Transform (FFT) realization can be used to minimize the computational demands.

There are also two different kinds of implementations of the filter-bank:

Uniform: the filters have equal bandwidth.

Nonuniform: the bandwidth of the filters are not equal.

The nonuniform approach is usually implemented because of the ability to base the spacing of the filters on perceptual studies and thereby fulfilling the requirements for a feature which accurately represent the information within the speech signal.

When the filter type has been decided one has to choose the number of filters used. Generally it is important that the number of filters is not too small because that would reduce the filter-banks ability to resolve the speech spectrum. Likewise the number of filters cannot be too large since that would make the bandwidth of the filters too narrow for some talkers (i.e.. high pitch talkers like women or children) unless there is a considerable overlap in the filters. This is so because the energy spectrum of the high pitch talkers is narrower than low pitch talkers and hence there is a bigger chance that the energy spectrum might lie between two filters if the number of filters is too high. In practise it has been proven that the ideal number of filters to be used lies between 8 and 32 (taken from [**?**, p. 93]). The precise number of filters depends on the sample rate of the signal as this will directly influence the number of resonance peaks (formants) of a given voiced sound.

Characteristics of the Filter-Bank Model

One of the advantages of using filter-banks is that one gets a large reduction in the bit-rate of the signal, because the signal is sampled and convoluted with a bandpass filter. This will hopefully result in an improved representation of the significant information in the speech signal. In addition it is possible by adjusting the spacing of the bandpass-filters to generate a model which describe the distribution of the information in the speech signal.

The major disadvantage of using filter-banks is that they are computationally demanding when realized in software which is true for both the IIR and FIR implementation. It should also be noted that althrough the model will extract features in a straight forward manner it is done without much thought to the physiological properties of the source.

The Linear Predictive Coding Model

Like the Bank-of-Filters model the purpose of the LPC model is to extract features from the speech signal through spectral analysis. The way in which the LPC model is implemented is however significantly different from the Bank-of-Filters model. While in the Bank-of-filters model the spectral content of the signal is derived though an array of bandpass-filters the LPC model takes advantage of the way humans produce speech in order to derive the spectral content.

The basic idea behind the LPC model is, that a speech sample $s[n]$ at a given time can be expressed as a linear combination of the past p speech samples, such that

$$s[n] = \sum_{i=1}^{p} a_i s[n-i] + Gu[n], \tag{2.3}$$

where the coefficients $a_1, a_2, ..., a_p$ are assumed constant over a given time frame and the normalized excitation $u[n]$ and the gain of the excitation G represent the base of the signal at time $n = 0$. By expressing $s[n]$ in the Z-domain and extracting the transfer function

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-i}} \tag{2.4}$$

it can be shown that a signal at a given time can be synthesized by using the scaled excitation $Gu[n]$ as input for an all-pole system $H(z)$. Figure 2.9 shows how a given speech signal can be synthesized. As mentioned in section 2.2 the normalized excitation for human speech is either voiced speech (essentially a quasiperiodic pulse train) or unvoiced speech (essentially a random noise sequence). By scaling the human excitation by the gain G and using this scaled excitation as the input for the all-pole system $H(z)$, the full vocal range can be synthesized. From this synthesis model it becomes clear that the all-pole system $H(z)$ is a time-varying digital filter modeling the vocal tract with the coefficients $a_1, a_2, ..., a_p$ as the filter-coefficients.

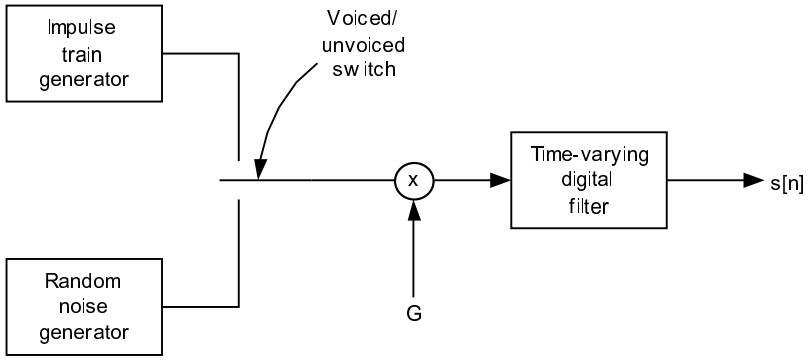

Figure 2.9: Speech synthesis model based on the LPC model [after ?, p. 101].

The method used to implement the LPC model can be divided into 3 parts, each of them will be described briefly. The individual parts and how they interconnect can be seen in figure 2.10.

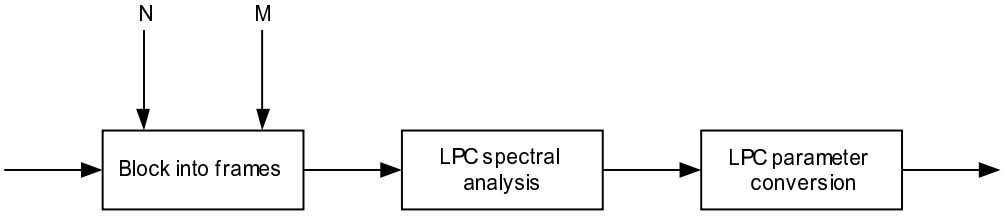

Figure 2.10: LPC analysis model [?, p.72]

The purpose of the first part "block into frames" is to divide the speech signal into a number of frames with a length of N samples in which the signal is considered approximately stable. As previously mentioned frame segmenting works because a speech signal can be considered approximately stationary over a short period of time. It should be noted that adjacent frames

will overlap with $N - M$ samples in order to ensure full coverage of the signal through the following spectral analysis. The process of frame blocking can be seen in figure 2.11.

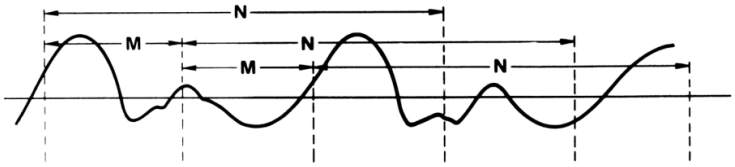

Figure 2.11: Blocking of speech into overlapping frames [**?**, p.114]

The next part is the actual process of determining LPC coefficients and this is mostly done through a method called autocorrelation. The autocorrelation method works by minimizing the average error between the actual and the approximated signal. The output of this process is p coefficients from where the first coefficients determines the energy of the signal within the given frame. The actual p LPC-coefficients are then determined by applying the Levinson-Durbin algorithm to the autocorrelated coefficients. The number of coefficients, p, are determined from previous implementation of the LPC-model and values between 8 and 16 are considered appropriate in a recognition scheme [**?**, p. 114]. The actual analysis-order is, as the number of filters in Bank-of-Filters, dependent upon the sample rate of the signal as all the resonance peaks (formants) will have to be represented.

The final part consists of converting the LPC-coefficients into a more robust representation of the signal called cepstral-coefficients. These coefficients are a spectral representation of the signal on a logarithmic scale.

Characteristics of the LPC Model

The LPC model has been widely used in speech recognition systems and there are several reasons for using the LPC method instead of the filter-bank method according to [**?**, p. 98]:

1. LPC models the speech signal. This is especially true for the quasi steady state voiced regions of speech in which the all-pole model of the LPC provides a good approximation to the vocal tract spectral envelope. It is less effective during unvoiced regions than for voiced regions but it still provides an acceptably useful model for speech recognition purposes.

2. LPC gives a parsimonious representation of the vocal tract characteristics.

3. LPC is mathematically precise and is simple and straightforward to implement in either software or hardware. The computation involved in LPC processing is considerably less than that required for an all-digital implementation of the bank-of-filters model.

4. Experience has shown that the performance of LPC used in speech recognizers is comparable or better than that of recognizers based on filter-bank [**?**, p. 98].

2.5 Recognition Approaches

In the following an overview of the different approaches available for doing the speech recognition task will be presented. The strengths and weaknesses of each of them will be examined and outlined to establish the basis for a proper choice of method.

The approaches can roughly be categorized as follows:

1. Pattern-Recognition approach
2. Artificial Intelligence (AI) approach
3. Artificial Neural Networks (ANN) approach
4. Acoustic-Phonetic approach

It should be noted that all through Artificial Neural Networks (ANN) are primarily used as an extention to the other 3 approaches mentioned it can also be used as a stand-alone recognizer and this is why it has been placed on the same level as the other approaches.

In the following the basic ideas behind each of the methods will be presented.

2.5.1 Pattern-Recognition Approach

The Pattern-Recognition approach can be split into four major steps as listed below:

1. Feature measurement
2. Pattern training
3. Pattern classification
4. Decision logic

A block diagram illustrating this approach is shown in figure 2.5.1.

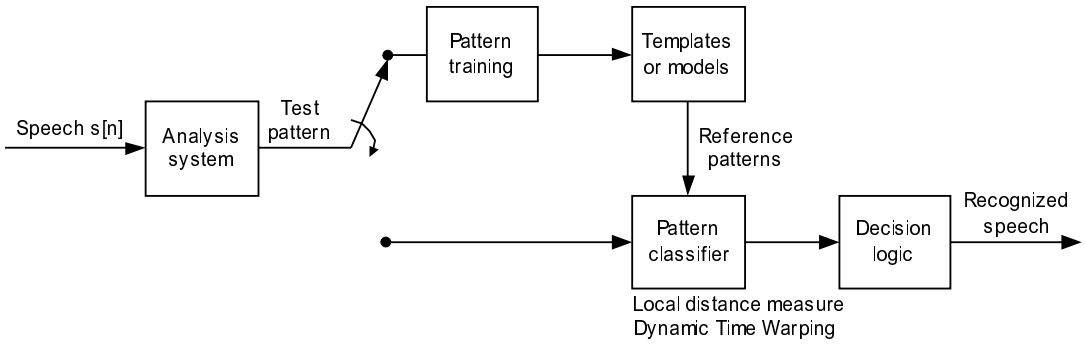

Figure 2.12: Block diagram for the pattern-recognition approach [after ?, p. 51].

Feature Measurement

At this common step a number of characteristics (features) are extracted from the input speech signal as described in section 2.4 in order to make a reliable comparison.

Pattern Training

In this step one or more test patterns corresponding to sounds of the same class are manipulated to determine a pattern representative for the features of the actual class. The pattern created from this process are called a reference pattern (or a template). It can either be derived with some averaging technique or constructed by a model that uses statistical information from the features in the test patterns. The most commonly used template training methods include: Causal training, Robust training and Clustering. Refer to appendix E for details on these methods.

Pattern Classification

In this step the test pattern (the feature vectors from the speech to be recognized) is compared to the reference patterns and a score measuring the similarity between the test pattern and each of the reference patterns is computed. In order to make these comparisons a local distance measure as well as a global time alignment mechanism is required. The local distance measure is needed because comparing speech patterns involves computation of local distance between each of the spectral vectors that constitute the speech patterns. Thus, an important decision at this stage is which distance measure the recognizer should implement. A wide number of measures are known, each with their strengths and weaknesses depending on the specific system to be implemented. A more detailed description of the considerations involved in finding an optimal distance measure is given in appendix C.1. Aligning and nonlinearly comparing two speech patterns is known as Dynamic Time Warping (DTW). A detailed description of this method can be found in appendix C.2.

The DTW time-aligns and compares two speech patterns X and Y consisting of the feature vectors $x_1, x_2, ..., x_{T_x}$ and $y_1, y_2, ..., y_{T_y}$. The result of the DTW is a global distance $d(X, Y)$, which allows a test-pattern X to be compared to a number of reference patterns Y by comparing the global distances. Because speech production is (almost always) realized at a different rate, the number of feature-vectors, T_x and T_y are (typically) not the same, and the indices i_x and i_y into the two patterns (mostly) does not line up linearly.

Figure 2.13 shows this nonlinear connection when comparing two patterns of the same spoken word. The DTW method "warps" the speech patterns being compared to a common time-axis by shrinking and stretching the patterns.

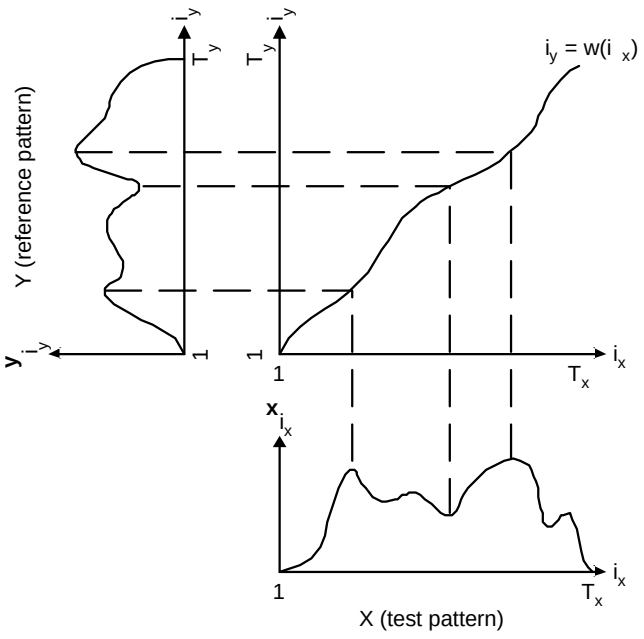

Figure 2.13: Overview of the DTW method [after ?, p. 126].

The warping is done by finding the optimal warping function $i_y = w(i_x)$ (which is generalized to $\phi = (\phi_x, \phi_y)$ in appendix C.2). The simplest solution for the warping function is when the two patterns compared are equal. Then $i_y = i_x$ and thus $w(i_x) = i_x$, giving a straight diagonal warping path. In the general case, the optimal warping function is found when the global distance is minimized:

$$d(X,Y) = \min_{w(i_x)} \sum_{i_x=1}^{T_x} d(i_x, w(i_x)) \tag{2.5}$$

where $d(i_x, w(i_x))$ is the local distance (between the feature vectors). The solution to the above equation can be obtained efficiently by using dynamic programming techniques, which is a widely used tool for solving sequential decision problems.

Decision Logic

Finally the last step is to decide which of the reference patterns match the test pattern best. Of course this decision is based on the computed scores in the pattern classification step.

When the units to be recognized are connected (with no silence between them), a connected word recognition algorithm must be used. [**?**] describe three algorithms for this task:

- The two-level dynamic programming approach
- The level building approach
- The one-pass (or one-stage or one-state) algorithm

The algorithms basically perform the same task of finding the best sequence of connected words (or visemes in our case). The main differences of the algorithms are in computational load, memory requirements, and the sequence in which the computations are carried out. In appendix C.3 the connected word recognition problem is described in general and the one-pass algorithm is described in detail.

It is possible to embed a decision support system (represented by e.g. AI) or adapt some grammar to improve the correctness of the decisions made by the recognizer. This is an improvement since it is not arbitary which viseme can follow the previous viseme.

Characteristics of the Approach

In the following the strengths and weaknesses that characterize the Pattern-Recognition approach will be outlined as stated in [**?**].

Advantages:

- The approach is insensitive to the choice of vocabulary words, task, syntax and task semantics, because no speech-specific knowledge is explicitly used.
- It is applicable to a wide range of speech units covering. phrases, words and sub-word units (visemes)
- It features an easy incorporation of syntactic and semantic constraints to improve the recognition accuracy and limit computations.

Disadvantages:

- The reference patterns are affected by the speaking environment and characteristics of the medium used to convey the speech.
- The recognition is speaker dependent.

2.5.2 Artificial Intelligence Approach

Analyzing sounds without thought can lead to errors when the sounds are somewhat similar. The words “come” and “run” or “hut” and “hot” sound almost the same, but change the actual

meaning, for example “Good ideas run when least expected” - “It sure is hut today”. It is easy for a person to realize the errors but the standard pattern recognizer cannot.

This is the motivation that leads to AI speech recognition. It is a study connected to cognitive psychology. The following will describe the human perception processes briefly in order to understand the intelligence, which is the foundation of speech perception.

Human Perception

The acoustic input signal is analyzed giving spectral information, which is stored in a sensory information store. This sensory information store holds other relevant info such as visual input and scents. Focusing on the auditory information, it is analyzed in various processes in neural nets in the brain. The analysis is controlled by the short- and long-term memory.

Lexical, syntactic, and semantic processes analyze the phonetic feature combinations. The context and other senses are included in order to fully understand the information at hand.

For example, if the next sentence would be “Then add flower and stir gently” the reader of this text would wonder right away, because it is out of context. The sentence “Power plants colorless happily old” makes no sense syntactically nor semantic.

Another interesting observation is the tendency for a person to manifest an image of the entire meaning of a message after the first few percepted words. The more previous knowledge and experience a person has regarding the topic, the easier it is to “guess” the correct meaning. The knowledge and experience is stored in a cognitive framework for later use.

These are the considerations that are the foundation of using artificial intelligence in speech recognition.

When using artificial intelligence in speech recognition it is mainly used for identifying and simulating the cognitive processes in the brain. Futhermore the acoustic, lexical, and semantic knowledge can be used to classify the speech.

Knowledge Based AI

There are two well-known types of knowledge based speech recognizer designs, bottom-up and top-down processors. The common problem is that they need huge knowledge sources regarding syntactical, lexical, acoustic, semantic, and language models. This knowledge must be categorized, structured, and modeled like the cognitive framework of the human brain.

The bottom-up processor analyses the low-level processes such as acoustic feature analysis. Then it sequentially proceeds to the high-level processes such as the language model. The bottom-up approach is illustrated in fig. 2.14.

The top-down processor is illustrated in fig. 2.15. The language model generates hypothetic meaningful sentences based on the syntactical, lexical, pragmatic and semantic knowledge. This is similar to the way a person predicts the meaning of a sentence based on the cognitive framework. The hypothetic sentences are matched against the speech signal word by word. When all units (words) match the sentence (utterance) is verified.

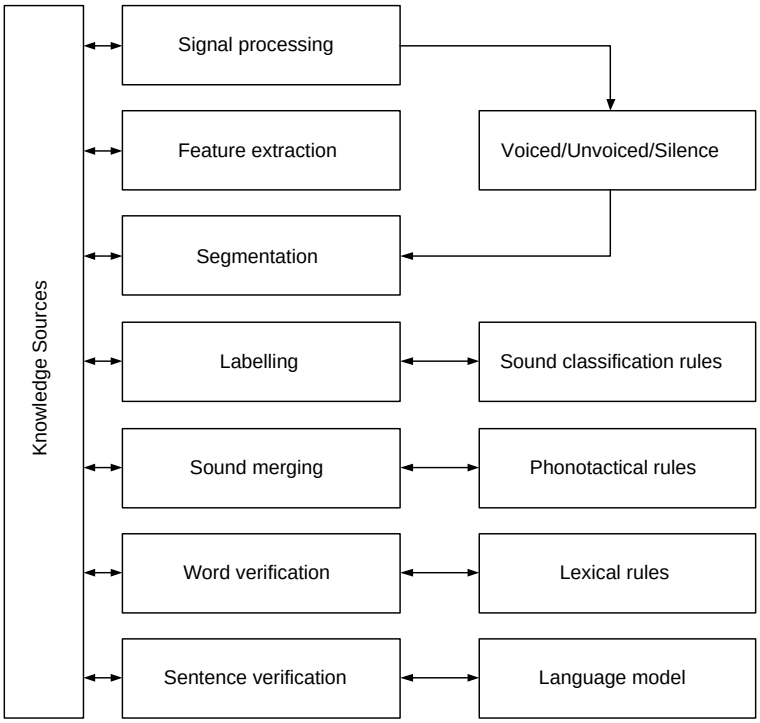

Figure 2.14: Bottom-up knowledge based recognition. The processes are matching the speech signal sequentially starting with the low-level units [after ?, p. 55].

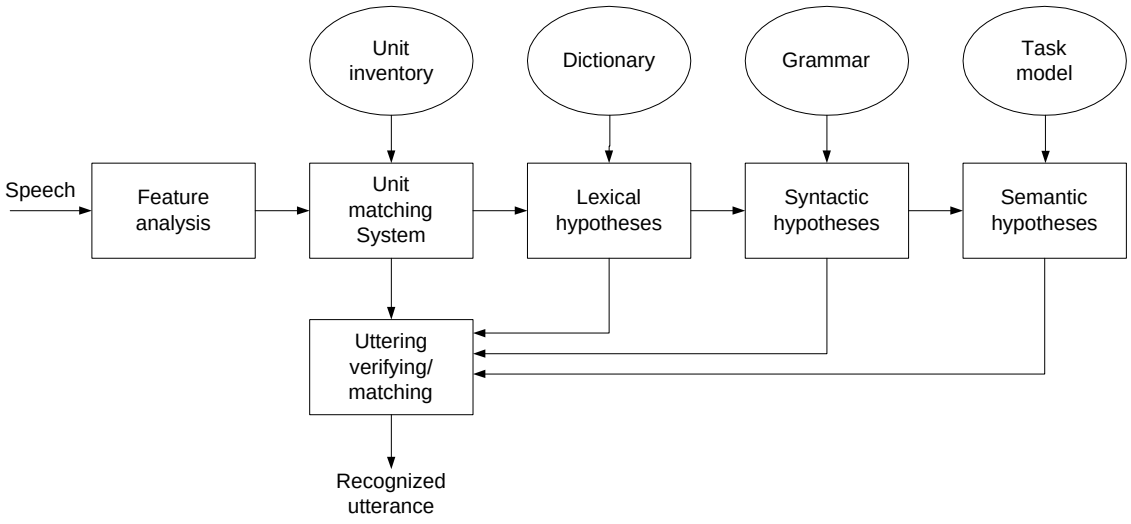

Figure 2.15: Top-down knowledge based recognition. The processor generates hypotheses based on the knowledge sources (inventory, grammar, etc) [after ?, p. 55].

Characteristics of the Approach

Advantages of knowledge based speech recognition:

- Automatic verification and correction of syntactic and semantic information in the recognized sentences.
- It is easy to locate a knowledge source and add new information or correct the old.

Disadvantages of knowledge based speech recognition:

- It needs huge data banks and frameworks. Advanced models have to be developed.
- The input must be sentences that make syntactical and semantic sense.

2.5.3 Artificial Neural Networks Approach

The neural networks can be used as an extension to a recognizer or it can be used solely as a recognizer.

The neural networks consist of neurons and weighting connectors. See fig. 2.16.

A neuron acts as a function that often is differentiable, continuous and nonlinear such as the sigmoid functions. The sigmoid function is as follows:

$$f(x) = tanh(\beta x), \qquad \beta > 0 \tag{2.6}$$

One neuron and its input connectors is called a simple computation element.

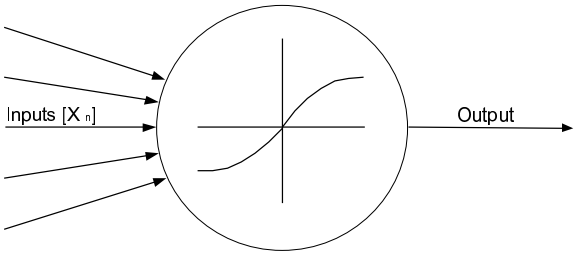

Figure 2.16: A computation element with only one neuron [after ?, p. 57].

Different types of neural networks are distinguished from each other by the network topology (how the neurons are interconnected). The most common network topology is the multi-layer perceptron network. All inputs are connected to each neuron in the first hidden layer. Each neuron in the first layer is connected to the next layer and so forth. A simple two-layer perceptron can recognize 10 steady state vowels using mere two acoustic formants. It has one output neuron per vowel as shown in fig. 2.17.

In order to include the dynamics that exist in speech signals the simple computational elements are expanded to include a number (N) of feature frames over time. This topology is called time-delayed neural network (TDNN).

Fig. 2.18 shows a TDNN designed to recognize three consonants.

There are more advanced network topologies such as the Hopfield network in which the outputs are included as inputs. This way it is recurrent and tends to stabilize at fixed points making it especially good at recognizing fixed sets of patterns such as printed number or letters. The neural network can be supported by the knowledge based AI to correct syntactical and lexical errors. The neural network can also be used as support for another pattern recognizer.

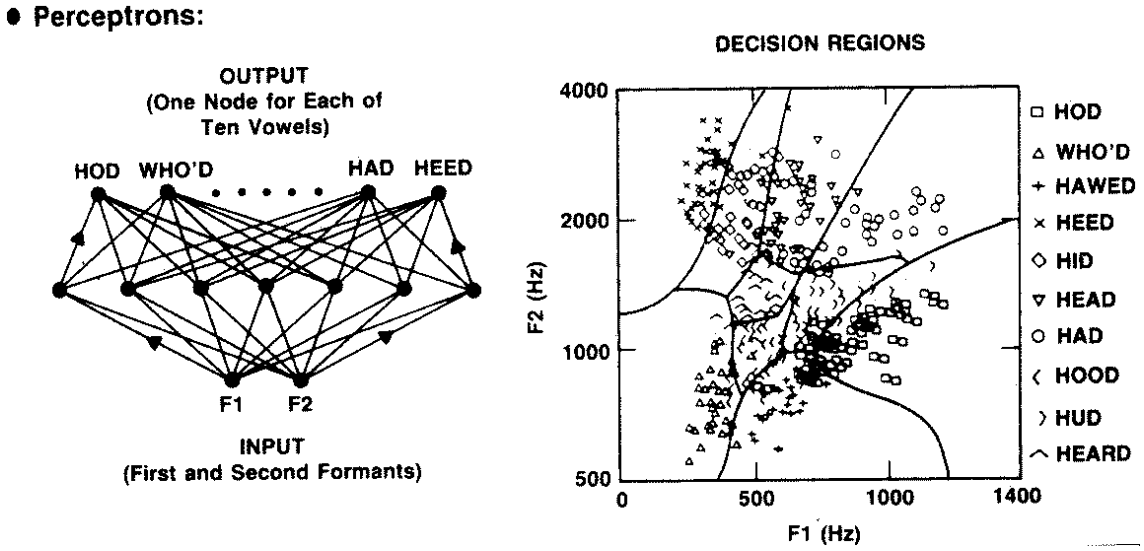

Figure 2.17: A multi-layer perceptron network designed to classify 10 vowels. Each output node represent a vowel. To the right the decision regions are plotted [?, p. 59].

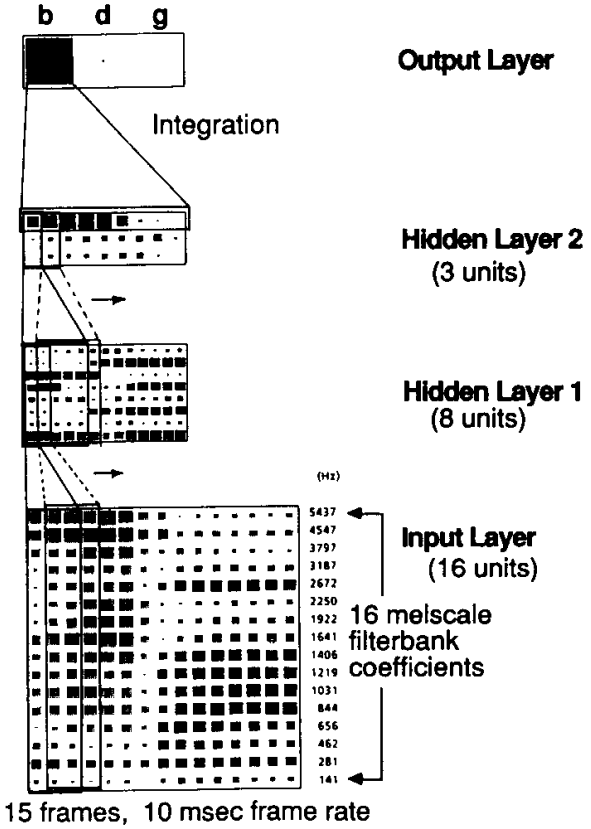

Figure 2.18: Time-delayed neural network. The first hidden layer joins 3 feature sets from the input. The second layer generate a feature set based on 5 sets from the first. Nine of those classify the consonants. Thus, the decision is based on 15 input frames (150 ms delay) [?, p. 55].

Learning

When the network is trained it is the weighting of the connectors and the offset within the neuron that are adjusted. This is done with back propagation. A labeled training set is used and the weights are randomized from the start. The output is calculated for a given input. The deviation from the desired output is calculated and has to be minimized by adjusting the weights and offsets. The gradients are derived for each computational element and the weights are adjusted accordingly. The exact procedure can be found in [**?**].

Characteristics of the Approach

Advantages of neural networks:

- The neural network is suitable for parallel computation because of their simple construction of computation elements.
- The performance of the system is not programmed or constrained; it can be adapted and improved real-time.
- The information is spread to every element of the network. If a defect in one element occurs the rest take over.
- It will learn the general information in a stochastic signal - if there is enough training material it will recognize the general pattern in a signal that would be almost impossible to identify and program manually.

Disadvantages of neural networks:

- In order to generalize it needs a lot of training material. The more hidden layers a multi-layer perceptron network has, the more training it needs.

2.5.4 Acoustic-Phonetic Approach

The Acoustic-Phonetic approach to speech recognition is based on the assumption that phonetic symbols can be recognized by using relations between phonemes and a set of acoustic properties of a speech signal.

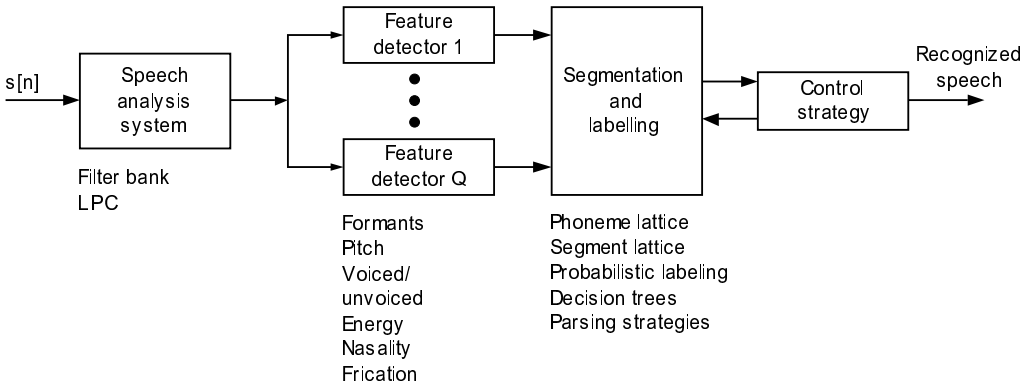

Figure 2.19: Block diagram of the acoustic-phonetic recognition process [**?**, p. 45].

The recognition process is outlined in figure 2.19 and consists of the following 4 parts:

- Preprocessing
- Feature detection
- Segmentation and labeling
- Lexical analysis

Preprocessing

The preprocessing is done using a technique such as LPC or Bank-of-Filters. For more information see section 2.4.

Feature Detection

The purpose of the feature detection part is to translate the spectral measurements from the preprocessing to a set of features that describe the acoustic properties of the different phonemes. Among these features are:

- Nasality (presence/absence of nasal resonance)
- Frication (random excitation in the speech)
- Formant locations
- Voiced/unvoiced
- Ratio of high- and low-frequency energy

The features are then used for the segmentation and labeling.

Segmentation and Labeling

The purpose of the segmentation and labeling part is to separate the speech signal into stable segments; that is regions in which the features are constant or change very little. Each of these segments represents a phoneme.

Furthermore each segment is labeled according to which phonetic unit best matches the acoustic features of the segment. To improve recognition accuracy often the N best matches are saved for the lexical analysis.

After segmentation and labeling a block of speech may look like figure 2.20, where the best matches are at the top of the lattice.

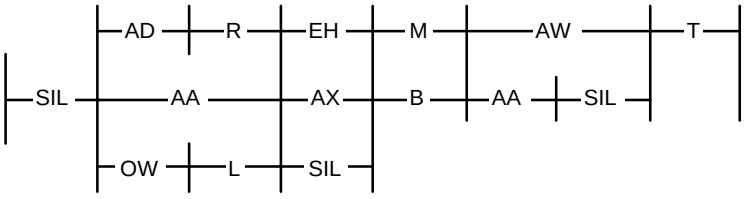

Figure 2.20: Phoneme lattice for the string "All about" [?, p. 43].

The labeling is usually done using binary trees.

It is the segmentation part that is the heart of the process and the most difficult part of the algorithm.

Lexical Analysis

The last part of the acoustic-phonetic recognition process is the conversion from the sequence of phonetic units to the spoken word(s). This process is usually done using a lexical search. Since we don't need to recognize which words are being said we don't need an advanced lexical analysis, instead the phoneme lattice from the segmentation and labeling part is converted into a viseme lattice (see appendix F for more information on how this can be done).

Characteristics of the Approach

In the following the disadvantages that characterize the Acoustics-Phonetic approach will be outlined.

- The approach requires extensive knowledge of the acoustic properties of phonetic units. This knowledge is language dependent.
- The choice of measurements does not have a solid theoretical base and are hence often selected based on intuition and ad hoc considerations.
- No well defined automatic procedure for optimizing or training the recognition process exist.

2.6 Summary

In this chapter an overview of the theory behind the system to be developed was provided. The fundamentals of how speech is produced and how it is interpreted by the human ear were presented. Next the concept of visemes was introduced. With this basic knowledge necessary for understanding the system to be developed the relevant analysis models and recognition approaches were finally presented. The characteristics of the analysis models for preprocessing and recognition approaches are repeated in summarized form in table 2.2 and table 2.3 respectively.

Finally the reader should note that state-of-art solution for the recognition task analyzed in this report includes Hidden Markov modeling, which has been omitted here, because it has been considered as being beyond the scope of this project.

Bank-of-filters	Linear Predictive Coding
1. Large bit-rate reduction, because the signal is sampled and convoluted with a bandpass filter. 2. By adjusting the spacing of the bandpass-filters the model can be made to describe the distribution of the information in the speech signal 3. Computationally demanding when realized in software. This is true for both the IIR and FIR implementation. 4. The model will extract features in a straight forward manner. However this is done without much thought to the physiological properties of the source.	1. Models the speech signal very precisely, this is especially true for the quasi steady state voiced regions of the speech. In these conditions the all-pole model of the LPC provides a good approximation to the vocal tract spectral envelope. 2. Parsimonious representation of the vocal tract characteristics. 3. Mathematically precise method, which is simple and straightforward to implement in either software or hardware. 4. Computation demands is considerably less than that required for an all-digital implementation of the bank-of-filters model. 5. Experience have shown that the performance of LPC used in speech recognizers is comparable or better than that of recognizers based on filter-bank.

Table 2.2: Characteristics of the preprocessing methods

Pattern recognition	Artificial Intelligence
1. Computation time for both training and classification are linearly proportional with the number of patterns used. 2. Insensitive to choice of vocabulary words, task, syntax and task semantics. 3. Applicable to a wide range of speech units. 4. Easy incorporation of syntactic and semantic constraints to improve the recognition accuracy. 5. Recognition is speaker dependent.	1. Needs huge data banks and frameworks. Advanced models have to be developed. 2. The input must be sentences that make sense if the knowledge based AI is to be used 3. Suitable for parallel computation. 4. Adaptive learning. 5. Information is spread to every element of the network making it a tolerant approach. 6. Able to recognize a general pattern in a signal that would be almost impossible to identify and program manually.
Acoustic-phonetic	**Artificial Neural Networks**
1. Requires extensive knowledge of the acoustic properties of phonetic units. This knowledge is language dependent. 2. The choice of measurements does not have a solid theoretical base. Therefore these are often based on intuition and/or ad hoc considerations. 3. No well defined automatic procedure for optimizing or training the recognition process exist.	1. Suitable for parallel computation. 2. Not programmed or constrained; it can be adapted and improved real-time. 3. The information is spread to every element of the network. If a defect in one element occurs the rest take over. 4. If there is enough training material it will recognize the general pattern in a signal that would be almost impossible to identify and program manually. 5. Needs a lot of training material. The more hidden layers a multi-layer perceptron network has, the more training it needs.

Table 2.3: Characteristics of the recognition approaches

CHAPTER 3

Requirement Specification

3.1 Preface

A part of a system for automatic mouth animation from a speech signal is being developed. The subsystem is named "Automatic Lip Synchronization of Animated Characters".

The project starting point is a project proposal made by Jakob Buck from Interactive Television Entertainment Aps (ITE). Even though the subsystem will be a component of the main system, it is being developed as an independent program, hereby ensuring a high level of reusability.

3.1.1 References

The project proposal [**?**]

3.1.2 Guidance for Reading

"Automatic Lip Synchronization of Animated Characters" is in the following abbreviated as ALSAC.

As this specification is based on [**?**], terms from there will appear in the following.

The subsystem being develop by the project group will be referred to as the system and the system ITE is developing will be referred to as the main system.

In the specification the person who's speech is recorded in the sound file is referred to as the speaker. The animated character is referred to as the character.

3.2 General Description

3.2.1 System Description

ALSAC has two main functions, training and recognition.

The training in ALSAC is for teaching the system which sounds corresponds to which visemes, so the system later can produce a stream of visemes from a speech signal. The training is done by comparing a speech stream with the corresponding ideal visemes for the speech stream. The

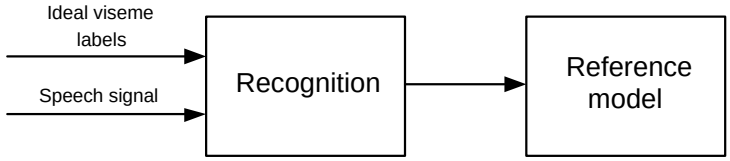

Figure 3.1: Training. The model is trained using speech signals and the corresponding ideal viseme labels

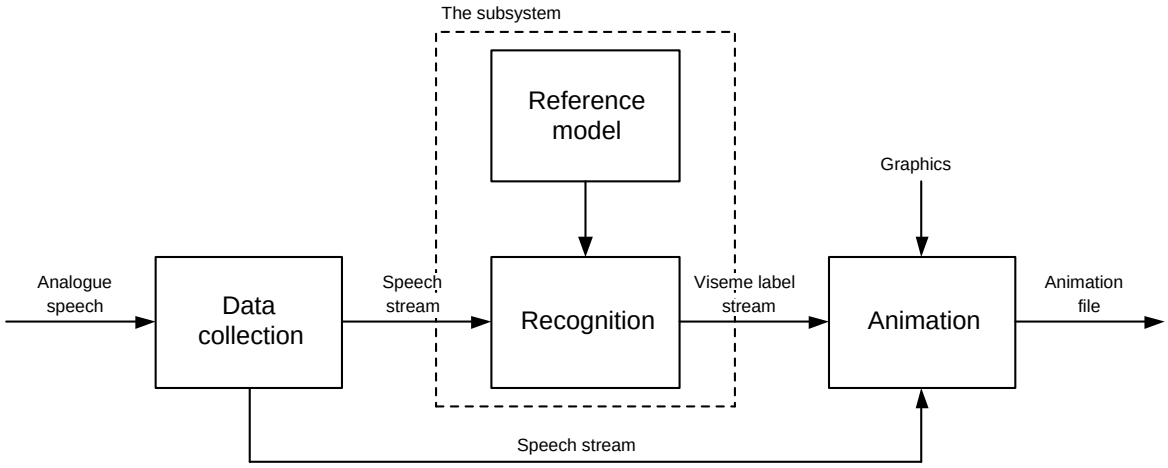

Figure 3.2: Recognition. The dashed lines represent the interfaces between the subsystem and the main system

training process creates a model of the mapping between speech and visemes. The training process is shown in figure 3.1.

The recognition is done by ALSAC which is a subsystem of the main system. In the main system a speech signal is sampled (data collection). The sampled speech is then streamed to ALSAC and the animation part of the system. ALSAC uses the model made during the training stage to generate a visemes stream which is streamed to the animation part of the system. Here the speech stream, the viseme stream and the animation graphics are combined to an animation file. The recognition process is shown in figure 3.2.

3.2.2 The Main Functionality of the System

ALSAC is to produce a stream of visemes from a sampled speech signal. The stream also contains corresponding time-stamps for the beginning of each viseme. The format of the output is to be defined in 3.3.1.

Since the recognition is speaker dependent, the system must include the ability to be configured and trained to different speakers using different languages. Consequently it is necessary to be able to load and save the templates belonging to a specific speaker. Furthermore it should be possible to update a profile if more training data becomes available.

3.2.3 The Limits of the System

In the following the limits of ALSAC will be outlined

- ALSAC is not to recreate the meaning of the speech.
- Phonemes is not to be recognized.
- The system is to run off-line (not real-time).
- The recognition is character and language dependent.
- ALSAC is not to perform noise reduction of the speech signal.

3.2.4 The Future of the System

The system has to be designed so that it can be expanded to include different kinds of recognition techniques. Likewise the user should be able to alter the complexity of the recognition process. Furthermore the system could be expanded to post-process the stream of visemes to limit the changes in the position of the mouth between two adjacent visemes (because too big changes in mouth position appears unnatural).

3.2.5 User Profile

The users of ALSAC will be the animators at ITE, who are to be considered as professional users.

The user interface is described in section 3.4.1.

3.2.6 Requirements to the Development Phase

The following analysis and design is to be done according to the object oriented method [**?**]. The implementation will be done i C++, using ANSI C++ where possible.

3.2.7 The Extend of the Delivery

ITE will be given a copy of the source-code and compiled program on a CD-ROM. Furthermore ITE will recieve a copy of the report as documentation. The rights to the system is as outlined in the NDA.

3.3 Specific Requirements

3.3.1 Definitions

- The specifications of the viseme file format will be specified by ITE and the project group during the design of the system.
- The sampled speech signal is defined as being 16 bit uncompressed mono wave files. The speech signal must be studio recording without any effect added to it. The sample-rate of the speech signal should be the identical during both training and recognition as this will produce the best result, but it is not a requirement.

3.3.2 Functional Requirements

- Training: The system should be able to be trained for different speakers using different languages (e.g. different profiles). The training function should have a graphical user interface.
- Recognition: The system should be configurable to work with different profiles.

3.4 Requirements for the External Interfaces

3.4.1 User Interface

The user interface will be implemented as a graphical interface only providing functionality necessary to demonstrate training and recognition. The recognition and training process should be able to run with minimum user interaction. The interface will be in English.

3.4.2 Software Interface

ALSAC will be running under Microsoft Windows 9x. The interface between ALSAC and the main system is yet to be determined, but will generally be as outlined in figure 3.2.

3.5 Performance Requirements

ALSAC should determine the same viseme as the handmade viseme files in at least 80% of the test-cases. Furthermore is should be at least as fast as the current method, which is defined as computing 1 MB of 22 kHz 8 bit mono samples (approx. 45 seconds of speech) per hour as described in section 1.2.1.

3.6 Quality Factors

The quality factors are given a priority according to the scale in [**?**]. 1 is unimportant and 5 is extremely important.

Reliability (4)
: It is very important that the system will work without errors. This is because the computation time could be long and the program should run with minimum user interaction.

Maintenability (2)
: It is less important that the system has a high level of maintenability, because debugging requires a high level of theoretical knowledge and therefore must be done by the project group or a person with equal level of knowledge about the theory.

Expandability (3)
: It is important that the system can be easily expanded, because of the terms outlined in section 3.2.4.

Userfriendliness (3)
: It is important that the system will be user friendly. The term user friendly is here more in the meaning 'fast and efficiently to use' instead of 'easy to use'.

Reusability (4)
: It is very important that the system has a high level of reusability because the system

should both be able to run as a stand-alone program and as a component of a greater system.

Integrity (1)
It is not important that the system maintain a high integrity, because it does not contain any secret data.

Efficiency (3)
It is important that the system is efficient, as outlined in section 3.5.

CHAPTER 4

Choice of Method

Unfortunately it is not possible in this project to make lip synchronization completely speaker and language independent. Furthermore to lower the burden on ITE it is preferable to choose a method that will not require a huge amount of preliminary work. Particularly labeling the speech signal would result in extra work for ITE in the training phase of the system. Considering this, the requirements from chapter 3 and the theories listed in chapter 2 under consideration the following choices regarding the system to be developed have been taken.

4.1 Speech Data

The EUROM1 database will be used for training and testing (see [**?**] for more information). This is done because it is possible to get a substantial amount of data that is labeled at the phonemic level. The fact that the data is labeled with phonemes makes it necessary to convert the phonemes to visemes. However this is a small price to pay considering the work required to manually label a clean speech signal with visemes. Furthermore the process of converting from phonemes to visemes can be automated as described in appendix F.

4.2 Viseme Set

ITE provided a simple viseme set that is found to be insufficient (cf. appendix F). The reason for this is mainly that the phonemes cannot be mapped to visemes in a ratio of 1 to 1. In the chosen viseme set the consonants will have 2 visemes. The decision of which viseme to use is based on the roundness of the vowel within the same syllable. Appendix F contains a thorough description of the selected viseme set and how the mapping from phonemes to visemes is done.

4.3 Preprocessing

As described in chapter 2 there are two commonly used methods for preprocessing a speech signal:

- Bank-Of-Filter
- Linear Predictive Coding

Considering that both the LPC approach and the Bank-Of-Filter approach fulfills the main objectives of the preprocessing, namely to find a way to accurately characterize a given speech signal without loosing any valuable information, both approaches should be considered. However in this project it has been chosen to implement the LPC approach since it gives some advantages over the Bank-Of-Filter approach, namely it is mathematically precise and computational less demanding. Furthermore the results obtained using the LPC approach have been proven to be at least equal to or greater than those achieved by the Bank-of-Filters approach (see section 2.4.2).

4.3.1 LPC Parameters

The actual algorithm used to implement the LPC approach is presented in appendix B and the parameters presented in the following are primarily based upon the considerations presented within this appendix.

As presented in section 2.4.2 there are three parameters which have to be determined in order to use the LPC approach: the frame length N, the frame slide M and the order of the LPC-analysis. In appendix B the optimal frame length and frame slide are determined to be 300 samples and 100 samples respectively. However in order to use these values in the program some problems will have to be solved first. Converting the values directly will give 15 ms and 5 ms respectively, which isn't sufficient to accurately estimate the filter. Considering this and the fact that the findings in appendix B assumes a relatively lower sample-rate than that of EUROM1 it has been chosen to use a frame length of 20 ms and a slide of 10 ms. These proposed values have previously been used successfully in similar projects using LPC on relatively high sample-rate signals [1].

Regarding the order of the LPC-analysis there is a problem similar to that of the frame length and slide. In appendix B it is stated that the range of the analysis order should be between 8 and 16 with 8 being the value of choice. However these values assume a relatively small sample-rate of $6,67kHz$ to $10kHz$. According to [**?**] the order of the LPC-analysis should be decided using following equation:

$$p = 2 + f_s/1000Hz \tag{4.1}$$

Considering that the sample rate f_s of the chosen speech data from EUROM1 is 20kHz the appropriate order p should be 22. Due to the fact that equation 4.1 provides a combination between the analysis-order and the signal sample-rate whereas [**?**] does not it has been chosen to perform the LPC-analysis with an analysis-order of 22.

4.4 Pattern Recognition

Regarding the method used to do the actual recognition there were four methods mentioned in chapter 2. The methods were as follows:

- Pattern-Recognition approach
- Artificial Intelligence approach
- Artificial Neural Networks approach
- Acoustic-Phonetic approach

The Acoustic-Phonetic approach is not in consideration in this system since it requires knowledge on phoneme-level and this system deals only with visemes. Since there is a problem getting enough data both to perform a decent training and still have enough data to do a good test, it

[1] Rule of thumb proposed by the supervisers.

is necessary to pick the one of the three remaining methods that requires the least amount of data. This leads to the choice of the Pattern Recognition approach because it needs less data to the training than both the Artificial Intelligence and the Artificial Network Network approach.

4.4.1 Pattern-Recognition

Within the pattern-recognizer DTW is chosen because of its ability to handle the problem of speech not always having a constant velocity when spoken.

The distance measure in the pattern recognizer is chosen be an Euclidean distance because of its low computational load, and the fact that it is much easier to use than some of the other distance measures (which also are considered out of scope for this project).

As for the connected word recognition problem mentioned in section C.3 it has been chosen to use the one-pass (one state) algorithm. This decision is based on the algorithms low computational load and low storage requirements compared to other algorithms that solve a similar task [**?**, Table 1]. It shoukd noted that a grammar will not be implemented as suggested in section 2.5.1.

4.4.2 Training

The Training method that will be incorporated is chosen to be the clustering method (Refer to Appendix E for a description). It was selected as the most reliable approach primarily because of its high recognition accuracy [**?**, p. 267]. The specific clustering algorithm that will be implemented is chosen to be a Modified K-Means (MKM) algorithm in favor of the Unsupervised Clustering Without Averaging (UWA). The reason for this decision is that it guarantees a 100% coverage of the training set [**?**, p. 273].

CHAPTER 5

Test Specification

In order to determine whether the requirements in the requirement specification has been fulfilled a set of tests have been developed. The training and recognition will be made using the speech streams and altered phoneme files from the EUROM1 database. The phoneme files are converted to viseme files using a conversion program made by the group. It is based on the simple algorithm described in appendix F and is called "lab2vis".

The available data will be divided into a training set and a test set. The ratio between the two data set will be varied as variable amounts of training data will be used in order to determine the optimal amount. The parameters used will likewise be varied in order to test whether the parameters given in section 4.3.1 are the optimal.

National Institute of Standards and Technology [**?**] has developed a benchmarking program for speech recognition systems. It is mostly used to compare systems, but it can also give single systems a quality score. The advantage of using this test system is that it implements time aligning and considers deletions, substitutions, insertions, and separations. Another advantage is that the score is a recognized evaluation which makes it comparable to other systems that uses the same benchmark. Another speech recognition tool called "The HTK Toolbox" [**?**] contains an evaluation tool called "HResult", which is compatible with NIST. "HResult" is used for comparing output label files with reference files. This tool will be used in the test, because it is accessible. When referring to "score" in the following it refers to the HResult score, which is an accuracy score, i.e. higher scores are better. This is opposed to the NIST score, which is an error score, i.e. lower scores are better. However the two scores are compatible as $NISTscore = 100 - HResultscore$.

The tests are outlined in table 5.1 to table 5.3.

Furthermore a check list is made. This is made in order to check that the statements about the system in the requirement specification have been fulfilled by the program. The following checklist shows each demand:

- The program is able to run under Windows 9x
- The program is able to load and save templates
- The program is able to update a profile if more training data becomes available
- The program output format is consistent with the format specified in 3.3.1

In the test, chapter 9, it will be marked whether each of the statements has been fulfilled. In

Basic system test
Purpose To show that the basic system is working, so that any further testing of ALSAC is reasonable. NOTE: The purpose is not to test the recognition performance of ALSAC. **Test method** The test is performed using ALSAC on a speech stream which were used during the training. The resulting viseme file is compared with the viseme file generated by lab2vis which was used during training. **Test passed criteria** ALSAC is working if the test scores above 90%

Table 5.1: Test of the basics in ALSAC

Recognition test
Purpose To test how ALSAC's recognition performs on a speech stream. **Test method** The test is performed using ALSAC on a speech stream which was not used during the training. The resulting viseme file is compared with the matching viseme file generated by lab2vis. **Test passed criteria** As stated in the requirement specification ALSAC passes the test if it scores above 80%.

Table 5.2: Test of the recognition done by ALSAC

Visual test
Purpose To test the visual aspect of ALSAC. It should be noted that this is the most important test.
Test method The test is performed by an impartial group. The group will evaluate the following set of viseme streams while listening to the matching audio stream: • A viseme stream generated by ALSAC • A viseme stream generated by lab2vis • A random generated viseme stream These are to be graded by the group on a scale from 1 to 5 where 1 is poor performance and 5 is good performance.
Test passed criteria ALSAC has passed the test if its average grade is at least 80% of the average grade of the viseme stream generated by lab2vis.

Table 5.3: Visual test of the recognition done by ALSAC

the following the analysis of the system to be developed is documented.

CHAPTER 6

Analysis

6.1 Introduction

In the following the program to be developed will be analysed.

The system has two user functions: Training and Recognition. Figure 6.1 shows a rich picture of the system. The input to the system is the speech data to be classified. When the system is in training mode, the viseme labels corresponding to the speech data is also given as input to the system, while this is not the case for recognition. First a preprocessing proces takes place, extracting the information which is used for the classification. Then the actual classification takes place and depending on the mode (training or recognizing) a stream of output visemes will be generated or the reference database updated.

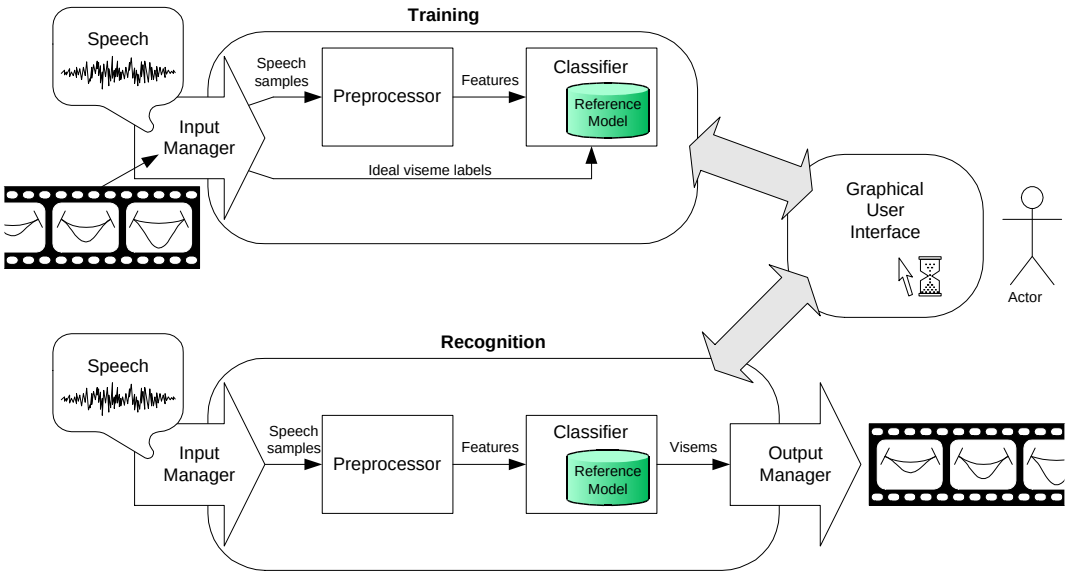

Figure 6.1: Rich picture of the system

The analysis is split into two parts; Analysis of the problem domain and analysis of the Application domain. In the first part the problem domain is defined and the object system corresponding to it is identified. A model is then developed by describing the classes of the object system, their

structure and behavior. In the second part the application domain is defined and the system's properties of use examined. That is, the use patterns and corresponding actors are identified. Finally also the interfaces of the system will be outlined in this part.

6.2 Analysis of the Problem Domain

The problem domain is by definition the part of the surroundings, which is to be administrated, monitored or controlled by a computer system. Thus we define the problem domain of this project as follows:

> *The proces of synchronizing the mouth positions of animated characters with corresponding speech files in a multimedia production.*

6.2.1 Structure

By observing figure 6.1 the class names listed below make candidates:

- Input Speech Stream
- Input Viseme Stream
- Output Viseme Stream
- Feature Stream
- Preprocessor
- Classifier
- Reference Database
- Profile

Futhermore the classes can almost immediately be logically grouped as shown in figure 6.2, which shows the class hiearchy. The cluster Input Manager is the grouping of the classes Input Speech Stream and Input Viseme Stream, while the Output Manager only consists of Output Viseme Stream. The Pattern Processor is a grouping of the classes Preprocessor, Feature Stream and Classifier. Finally The Profile and Reference Database classes groups into a cluster called User Manager. This structure logically divides the system into four major steps each handling a part of the proces described in the definition of the problem domain, namely:

- Input Manager: Collect the data to be processed
- Pattern Processor: Proces the data to determine the visemes and time labels the speech should be synchronized with
- User Manager: Provide a way for the user to save the result and the configuration
- Output Manager: Output the result of the proces in the desired format

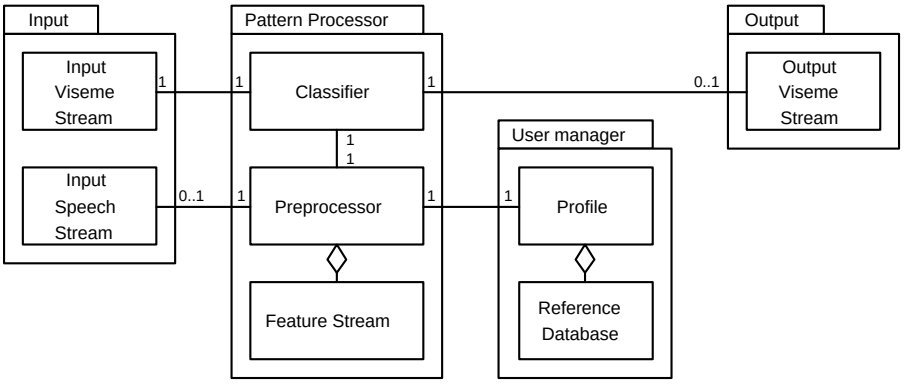

Figure 6.2: Class structure

6.2.2 Classes and Behavior

In the following each class and the corresponding object's behavior will be described in order to complete the model of the problem domain.

Input Speech Stream

This class embeds the speech data and provides funtionality for retrieving it on the fly during execution. The state diagram for it can be seen in figure 6.3.

Figure 6.3: State diagram of Input Speech Stream

Input Viseme Stream

This class is intended to hold the viseme sequence and time labels which belongs to a given training speech file. The state diagram for it can be seen in figure 6.4.

Figure 6.4: State diagram of Input Viseme Stream

Output Viseme Stream

The Output Viseme Stream class provides a mechanism for accumulating the determined visemes to a resulting viseme sequence. The state diagram for it can be seen in figure 6.5.

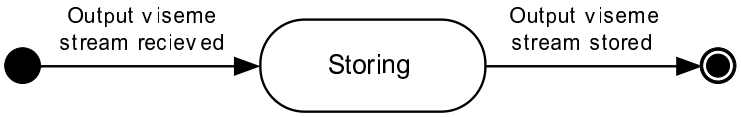

Figure 6.5: State diagram of Output Viseme Stream

Preprocessor

The Preprocessor processes the sound data into features for the recognition. The state diagram for it can be seen in figure 6.6.

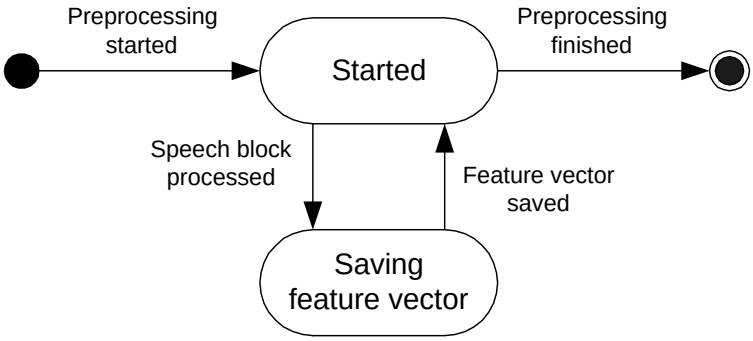

Figure 6.6: State diagram of Preprocessor

Feature Stream

The Feature Stream class should constitute an internal interface between the Preprocessor and the Classifier in the Pattern Processor. Thus it should contain functions allowing the Preprocessor to deliver extracted features and allowing the Classifier to retrieve features for pattern classification. The state diagram for it can be seen in figure 6.7.

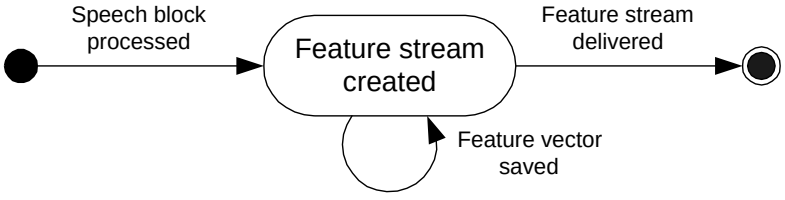

Figure 6.7: State diagram of Feature Stream

Classifier

The Classifier selects which viseme best matches the current set of features. The state diagram for it can be seen in figure 6.8.

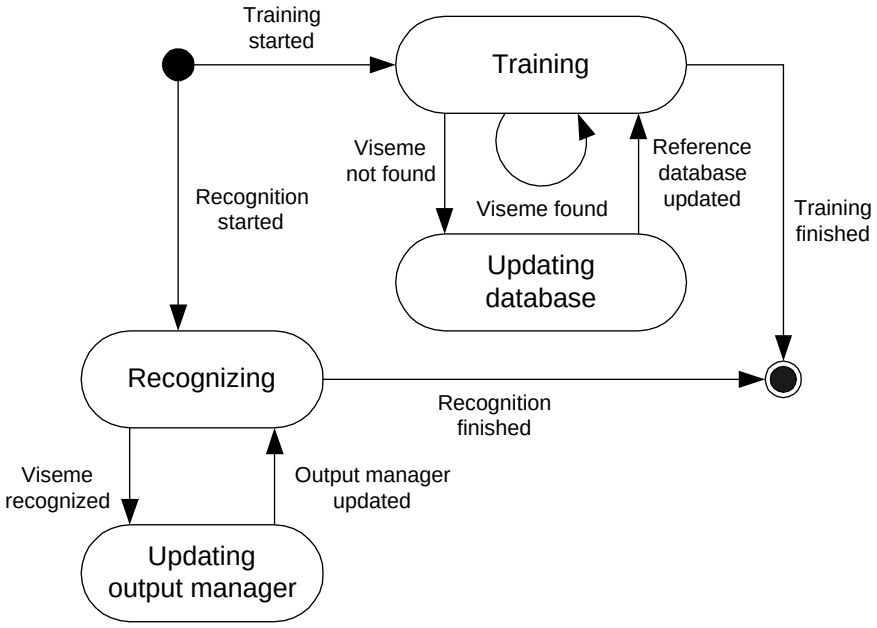

Figure 6.8: State diagram of Classifier

Profile

The Profile class is intended to keep track of the different users of the system. The state diagram for it can be seen in figure 6.9.

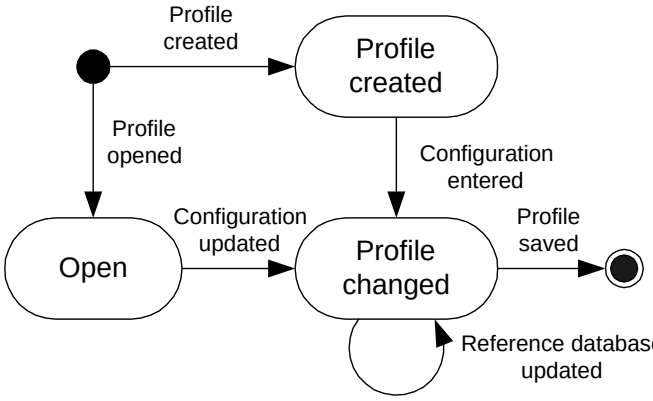

Figure 6.9: State diagram of Profile

Reference Database

The Reference Database class is the storage for the patterns identified by the Classifier. It is used as a dictionary in the pattern comparison proces. The state diagram for it can be seen in figure 6.10.

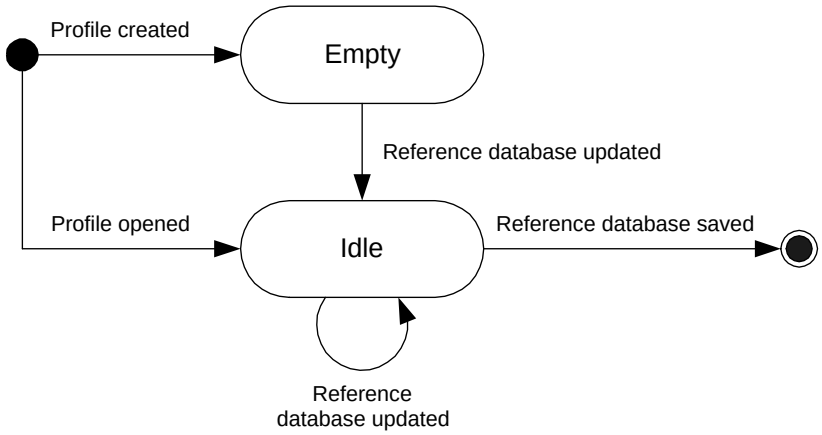

Figure 6.10: State diagram of Reference Database

From the state diagrams of the objects the events shown in table 6.1 can be generated. The following abbreviations for the object names have been used in the table:

- Input Viseme Stream(IVS)
- Input Speech Stream(ISS)
- Output Viseme Stream(OVS)
- Feature Stream(FES)
- Preprocessor(PRP)
- Classifier(CLS)
- Profile(PRF)
- Reference Database(RDB)

6.3 Application Domain

The application domain is outlined in the requirement specification.

6.3.1 Use

The system is to help the animators at ITE in their work. More precise it shall provide a tool for the animaters to (semi)automate the proces of synchronizing the speech with the appropiate visemes. However some postprocessing, such as interpolation of visemes and other visual effects, must be expected in other to create a satisfying final result. Futhermore it should be noted that recording of speech signals is not part of the system. It is expected and required that the speech has been recorded and digitized prior to the use in this system.

	IVS	ISS	OVS	PRP	FES	CLS	PRF	RDB
IVS requested	+							
IVS delivered	+							
ISS requested		+						
ISS delivered		+						
OVS recieved			+					
OVS stored			+					
Preprocessing started				+				
Preprocessing finished				+				
Speech block processed				+	+			
Feature vector saved				+	+			
FES delivered					+			
Training started						+		
Recognition started						+		
Viseme not found						+		
Viseme Found						+		
RDB updated						+		+
Training finished						+		+
Viseme recognized						+		
Output Manager updated			+			+		
Recognition finished						+		
Profile created							+	
Configuration entered							+	
Profile opened							+	
Configuration updated							+	
Profile saved							+	

Table 6.1: Events and affected objects

Actors

The actors of the system are animators working at ITE. More than one actor is expected, but the actors (animators) are all assumed to share a common set of use patterns, that is they are considered as actors of the same type.

Use Patterns

Two main use patterns can immediately (based on the requirement specification) be identified, namely

1. Training
2. Recognition

The use patterns are illustrated in figure 6.11 and figure 6.12 respectively.

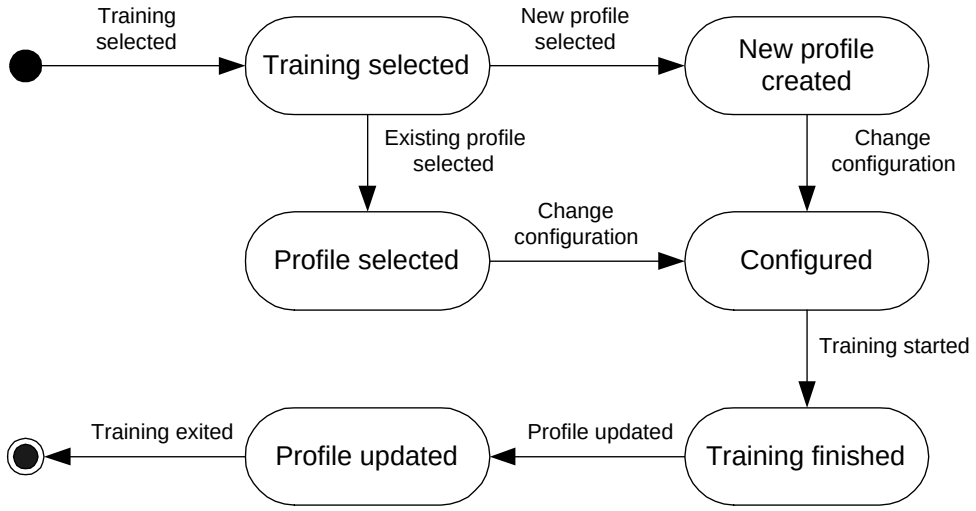

Figure 6.11: Use pattern for Training

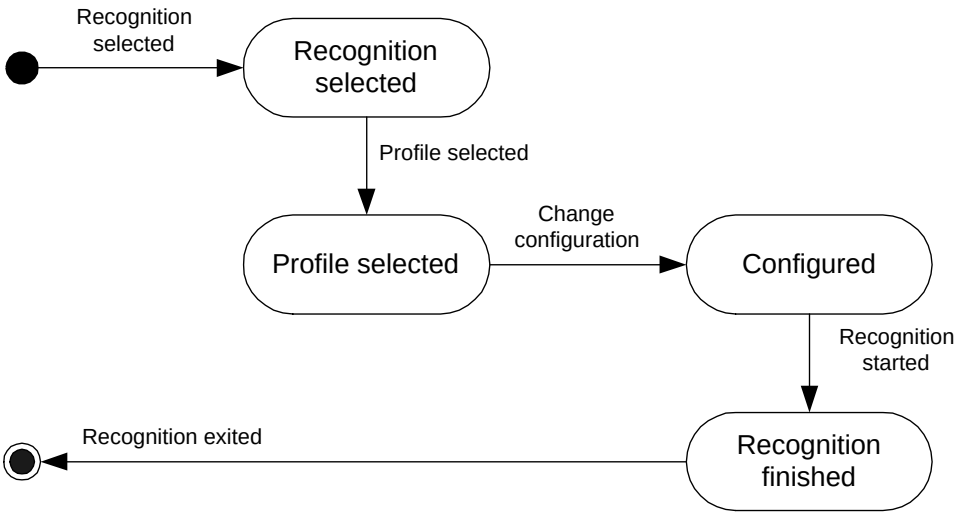

Figure 6.12: Use pattern for Recognition

6.3.2 Graphical User Interface

In the following section the classes, their structure and behavior will be identified in order to determine the objects the Graphical User Interface should be made of.

Structure, Classes and Behavior

The classes needed for the Graphical User Interface is determined partly by the requirement specification and the use patterns. The class diagram can be seen in figure 6.13.

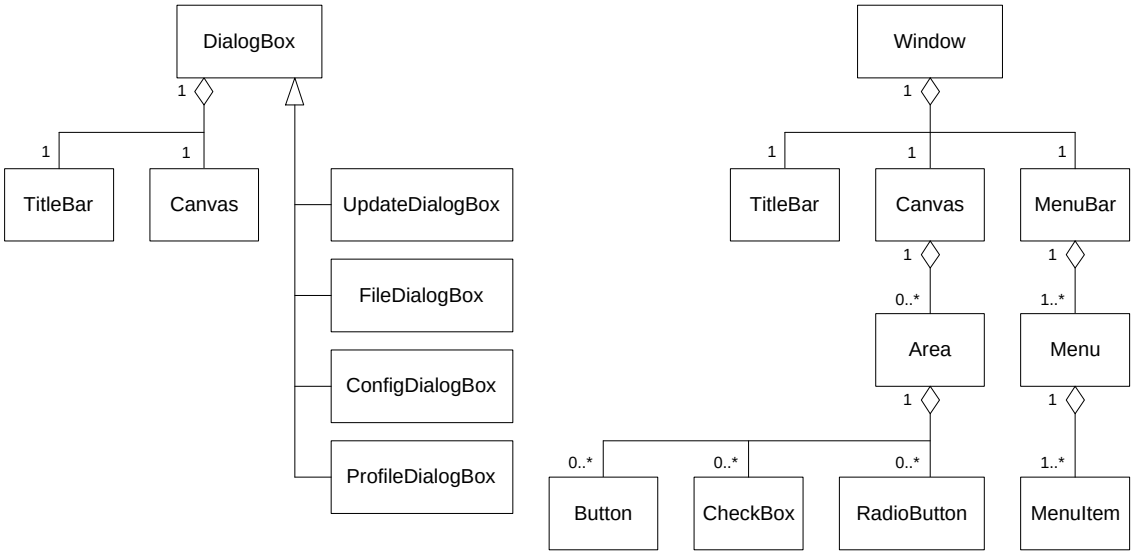

Figure 6.13: Class diagram for the Graphical User Interface

In order for an operating system to handle the actions performed on a window, the window needs to be subscribed to the operating system. Hereby the existence of the window will be known for it and thus enabling event handling. The involved objects must be associated with an event handler in order for the events to be catched and handled in a way different from the predefined action for a given event. Hence several of the classes shown in figure 6.13 are already defined and handled by MicrosoftWindows. The focus will therefore be on identifying the events which will not be handled by performing a predefined standard action. Thus events such as “start training” will be in focus, in opposition to standard events like “resize window”, because these are automatically handled. A description of these objects and their attributes are not of interest, since this already can be found in the Win32 API. The events of interest will be found by a relatively detailed investigation of the interactions between the user and the system illustrated in the use patterns - Training and Recognize.

The Training Scenario

Given the program has been started and initialized, the user can now begin a training session. The expected user actions and corresponding events are shown in figure 6.14, which partly is identified from the use pattern shown in figure 6.11. The training session can be exitted from the main window.

The Recognition Scenario

In a similar manner the events involved in a recognition session is created partly from it’s corresponding use pattern shown in figure 6.12. The event flowdiagram is shown in figure 6.15.

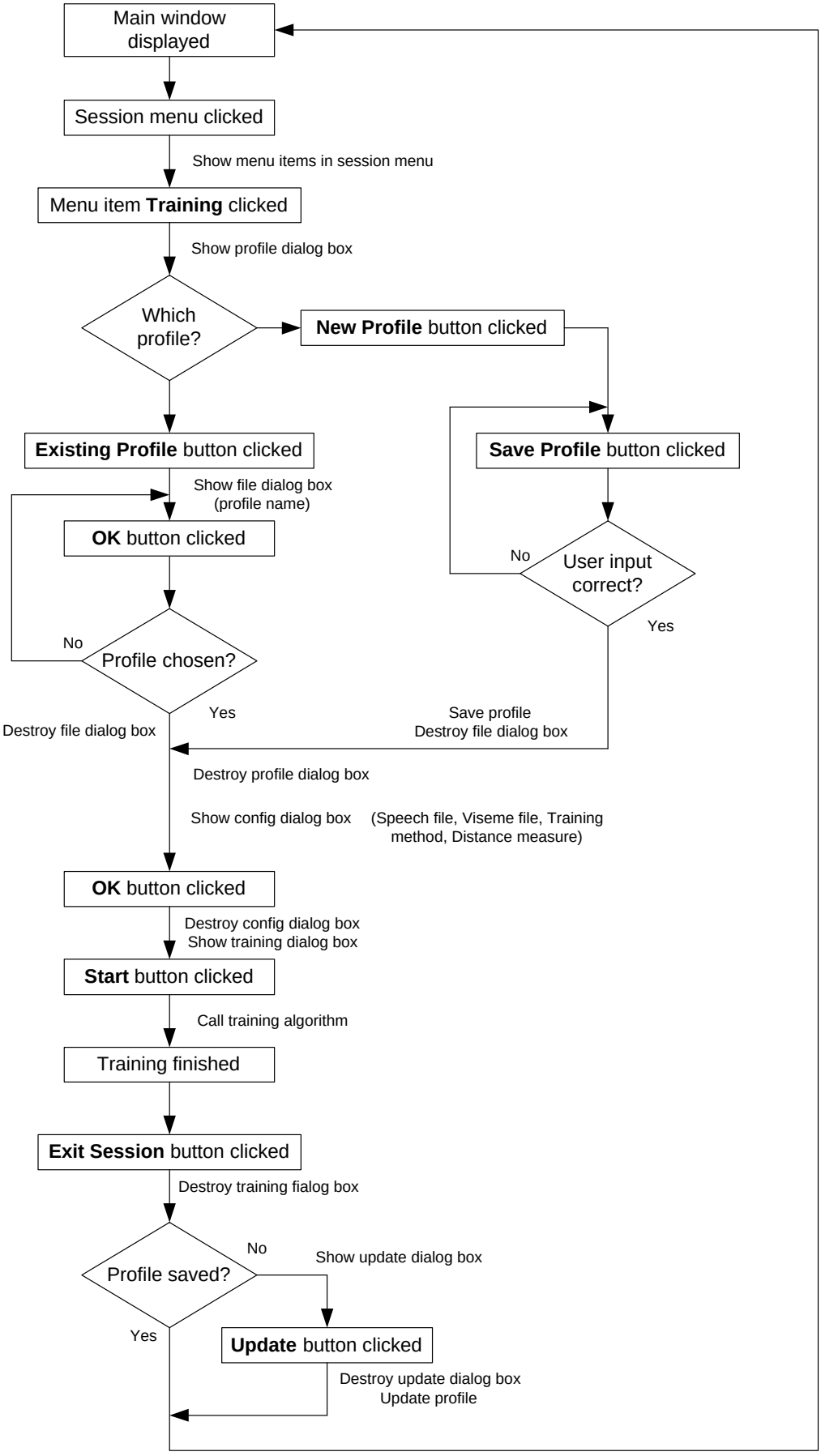

Figure 6.14: Event flowdiagram for Training

The recognition session can be exitted from the main window.

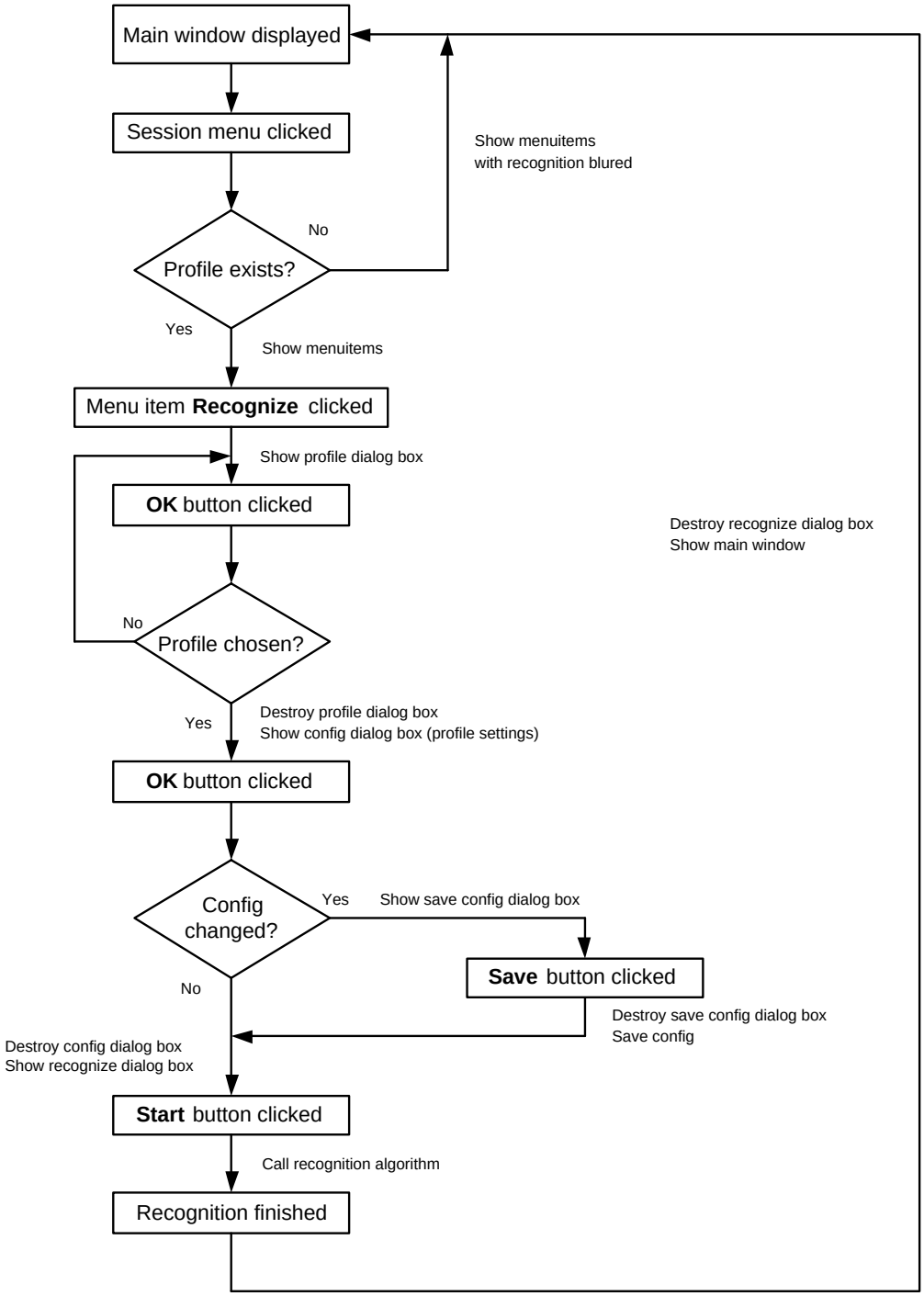

Figure 6.15: Event flowdiagram for Recognition

Functions

The user should be able to train and recognize as the two main actions. Examining the events illustrated in figure 6.14 and figure 6.15 it is however quite obvious, that these main actions is performed in a number of small steps. This is from the developers point of view equal to invoking a number of functions to constitute the complete action. The functions which immediately can be seen from the figures are:

- createWindow
- destroyWindow
- createDialogBox
- destroyDialogBox
- loadConfig
- saveConfig
- loadProfile
- saveProfile
- initialize
- validateUserInput

CHAPTER 7

Design

This chapter describes the design of the system to be developed based on the analysis described in chapter 6.

7.1 Architecture

7.1.1 Criteria

The system will be designed according to the criteria outlined in the requirement specification (chapter 3) and the choice of method (chapter 4). Furthermore it is to be designed with a high level of modularity. This is done in order to allow for modifications of for example the preprocessing or recognition strategy with only minor modifications to the rest of the system. In order to allow for easy change of strategy the design will make widespread use of the Strategy Design pattern (see appendix D for more detail).

7.1.2 Component Architecture

The system can be directly divided into two main components from the analysis: The Model component and the User Interface component. The Model component handles everything related to the actual recognition, and is almost directly described as the problem domain in the analysis. The component is further divided into the clusters given in the analysis document as shown in figure 6.2. The User Interface component handles all interaction with the user. This component is not directly part of the system wanted by ITE thus it will be implemented only with minimum functionality making it possible to demonstrate the system. Because the User Interface developed is of no interest to ITE, the interface between the GUI and the Model component must be loosely coupled, thereby providing an simple system interface for ITE.

The architecture can in general be viewed as a client/server architecture in which the Model component serves the User Interface component with functions for training and recognition.

7.1.3 Process Architecture

Because of the inherent serial structure of the algorithm the entire recognition process will be designed as one thread. The User Interface will likewise be running as an independent thread.

This is done to keep the system simple and still gain an responsive User Interface. Furthermore this architecture provides a low coupling between the two components.

7.2 Model Component

7.2.1 Structure

After the analysis it has become clear that the Model component needs two more classes, a SystemManager class and a Segmentator class. The purpose of the SystemManager class is to provide a simple interface to the Model component.

The purpose of the Segmentator class is to segment the speech signal into segments containing speech data. The purpose of this is threefold:

- To limit the amount of speech data held in memory at a time
- To minimize the latency of the system. This is important if the system is ever to run realtime
- To remove areas of silence which do not need to be recognized

The Segmentator class is located between the Preprocessor and the Classifier. This is done because the feature vectors contain information about the energy of the signal which is useful in determining areas of silence.

In addition to the Segmentator class two more classes are needed to handle the stream of visemes and feature vectors.

Figure 7.1 shows the class diagram of the Model component.

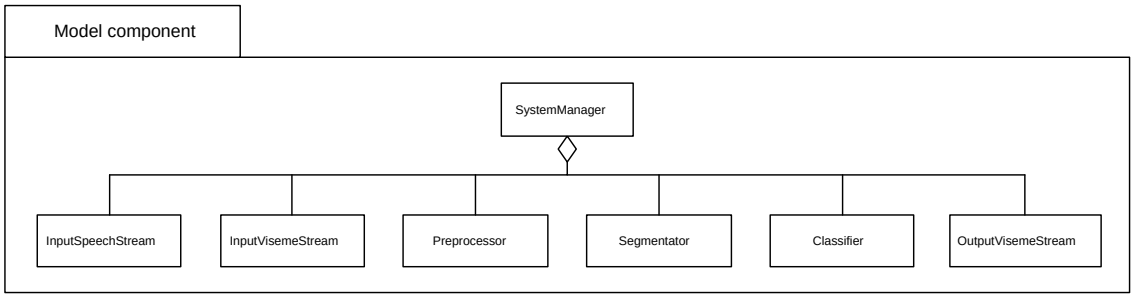

Figure 7.1: Class diagram of the Model component

7.2.2 Model Component Architecture

The Model component uses a stratified architecture. The SystemManager is located at the top level. This is the interface to the Model component. The SystemManager handles initialization and deinitialization of the needed objects and then transfers control to the Segmentator which is the controlling object for the signal processing proces.

In the following section the behavior of the Segmentator for the two cases will be described.

Training

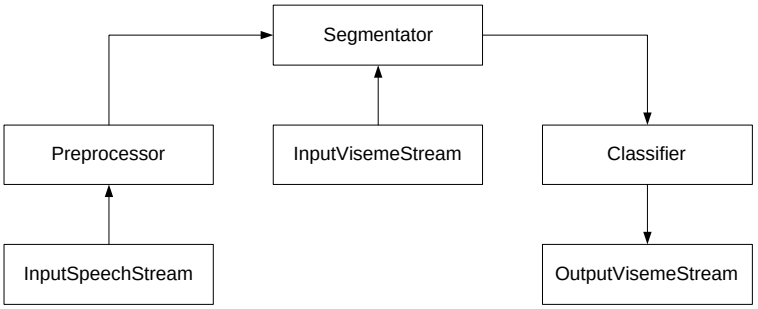

Figure 7.2: Active objects in the Model component during training.

Figure 7.2 shows the interaction between the active objects during training. The algorithm of the Segmentator consists of 5 steps.

1. A Viseme object is read from the InputVisemeStream object
2. The corresponding Feature objects are read from the Preprocessor
3. The Classifier's updateTrain method is called with the Viseme object and the sequence of Feature objects
4. Step 1 to 3 is repeated for all of the viseme stream
5. Finally the Classifier is notified by calling its train method

Recognize

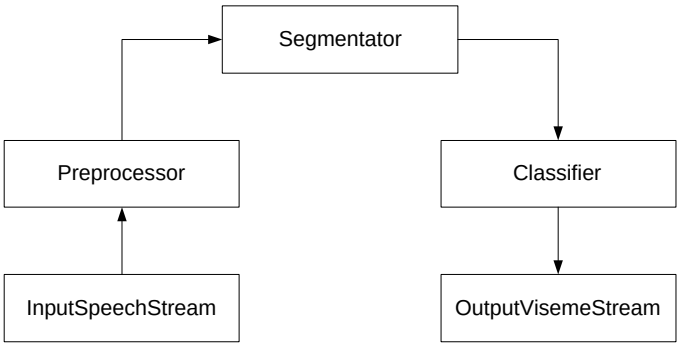

Figure 7.3: Active objects in the Model component during recognition.

Figure 7.3 shows the interaction between the active objects during recognition. During recognition the Segmentator acts like a gate which only opens for input if the energy level of the stream of Feature objects rises above a given threshold. This is done to shorten the length of the test sequence and thereby lower the memory consumption of the system. The system can be in two states: Opened or closed.

1. The system starts in the closed state
2. A Feature object is fetched from the Preprocessor
3. If the energy level of the Feature is above the threshold the Feature object is saved for later, otherwise it is discarded
4. If the energy levels have been above the threshold for a specified time the segmentator changes to the open state, else step 2 and 3 are repeated
5. A Feature object is fetched from the Preprocessor and saved for later
6. If the energy level of the Feature object is below the threshold a flag is set, else it is cleared
7. If the flag has been set for a specified time the last n Feature objects are discarded, the Segmentator calls the recognize method of the Classifier with the rest of the sequence and the state is changed to the closed state, else step 5 and 6 are repeated
8. Step 2 to 7 are repeated for all features

7.2.3 Configuration

A way of keeping configuration data like thresholds for the Segmentator or the refernce model for the Classifier is needed. Since the system is to support different strategies an extendable configuration system must be designed. For this purpose a new class called a Container is needed. A Container contains configuration data for an object of the Model component. Different specializations of the Container class is needed for each class containing the specific configuration data and methods for accessing it. In addition to the methods for accessing the specific configuration data two methods for setting and getting the configuration data inside the Container object are shared by all specializations.

- getData()
- getLength()
- setData(data)

These three functions are used for loading and saving configuration data to a disk file in a standardized manner.

Furthermore each container contains a getType() method so its type may be determined at runtime.

Furthermore an extra class is needed to contain all the Container objects to a given system. For this purpose a Profile class has been made. This class contains the following public methods for accessing the internal containers.

- getInputSpeechStream()
- getInputVisemeStream()
- getPreprocessor()
- getSegmentator()
- getClassifier()
- getOutputVisemeStream()

Information about the Profile file format can be found in appendix A.

7.2.4 Classifier

The Classifier consists of two parts; training and recognition. As the training part uses the MKM clustering algorithm and the DTW algorithm, these algorithms have been put in seperate classes, MKMTraining and DTW. This makes it possible to change the DTW algorithm to one with different path constraints. Similarly, the recognition part uses the One-Pass algorithm which has been put in a seperate class, OnePass. This also allows for changing to different path constraints. The class hierarchy of the classifier can be seen in figure 7.4.

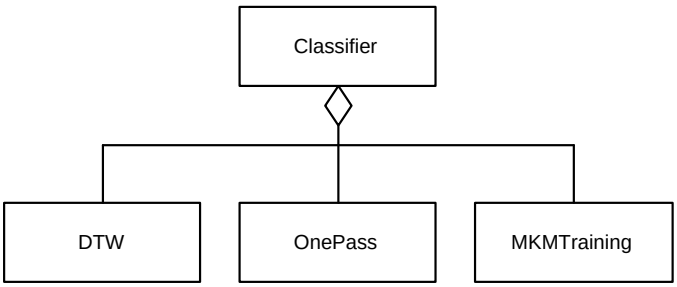

Figure 7.4: Classifier class hiearchy

7.2.5 Classes

InputSpeechStream

Objective: To read speech data from a given source and deliver a stream of speech when requested.

Methods: getSpeech(nr_samples), getSampleRate()

InputVisemeStream

Objective: To read viseme data from a given source and deliver a Viseme object when requested.

Methods: getViseme()

Viseme

Objective: To contain a viseme and the corresponding time stamp.

Methods: getViseme(), getTime()

Preprocessor

Objective: To convert a block of speech data to a feature object.

Methods: getFeature()

Feature

Objective: To contain a feature vector and offer operations on it.

Methods: data(), length(), energy(), distance(Feature), time()

Segmentator

Objective: To divide the stream of Feature objects from the Preprocessor into segments suitable for training or recognition.

Methods: recognize(), train()

Classifier

Objective: To classify which viseme a stream of Feature objects correspond to. Further more it should be able to be trained by giving it a stream of Feature objects and the corresponding viseme. The training is divided into to parts, updateTrain and train to support various kinds of classifier strategies.

Methods: recognize(Feature[]), updateTrain(Feature[],Viseme), train()

OutputVisemeStream

Objective: To save viseme data to a destination.

Methods: setViseme(Viseme)

SystemManager

Objective: To handle construction and destruction of the other objects of the Model component. Further more it provides a simple interface to the component.

Methods: recognize(Profile), train(Profile)

Profile

Objective: To contain configuration data. This includes the reference model of the classifier.

Methods: load(), save(), getInputSpeechStream(), setInputSpeechStream(), getInputVisemeStream(), setInputVisemeStream(), getPreprocessor(), getClassifier(), getOutputVisemeStream(), setOutputVisemeStream()

7.2.6 User Interface Component

The User Interface consists of a single main window. From this window the user can interact with the system as shown in the navigation diagram on figure 7.5. The window is build up from a menubar and an empty canvas (client area). The menubar contains the menus and menu items shown in figure 7.6 from where the user can choose the desired action to be performed. When a menu item is clicked an appropriate dialogbox is created to handle the user's dialog with the system and to collect user input.

Classes

A User Interface consists of the classes Window, Dialog, Menubar, Menu, MenuItem, Event, EventHandler, EventListener, Button etc., but since the User Interface component for this system will be build up from existing standard classes these will not be described here. Please refer to the C++ programming language and literature on windows programming for more information on the subject.

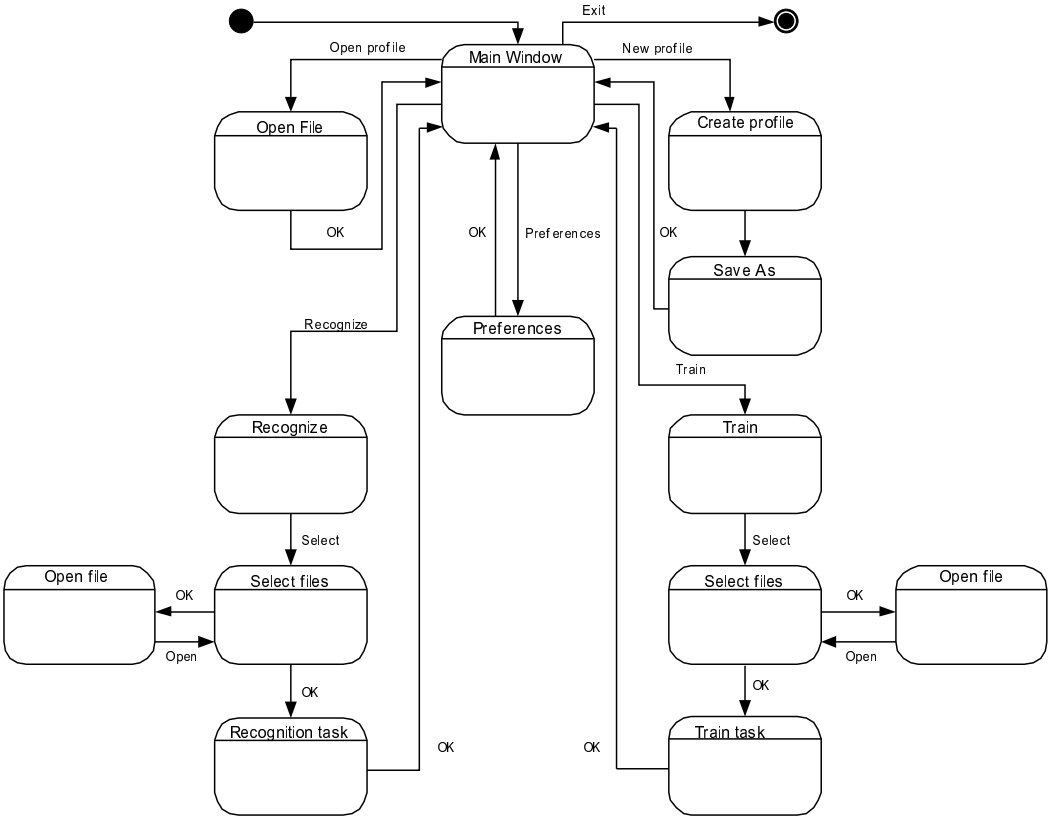

Figure 7.5: Navigation diagram for the program

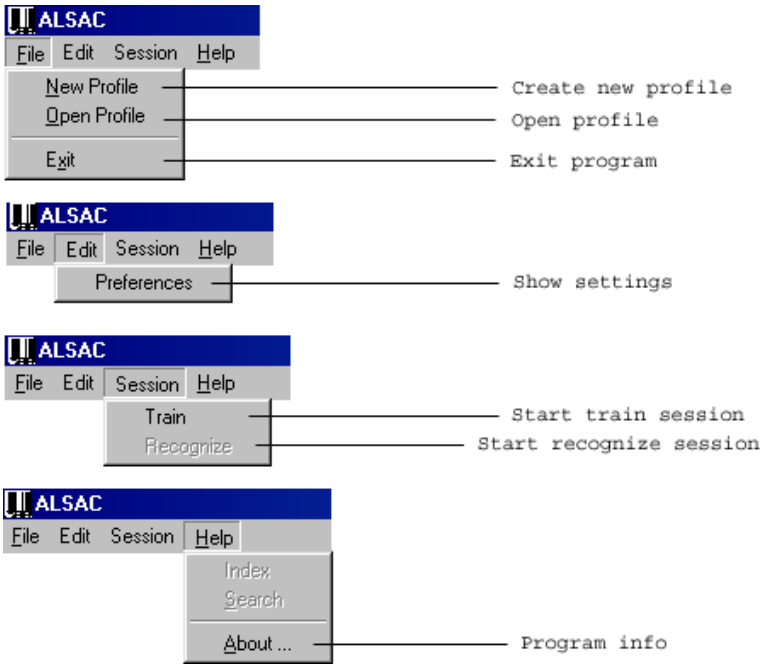

Figure 7.6: The menus of the system

Elements

The elements of the User Interface are:

- Main window
- New Profile Dialog Box
- Open Dialog Box
- Preferences Dialog Box
- Train Dialog Box
- Train Input Dialog Box
- Train Update Dialog Box
- Recognize Dialog Box
- Recognize Input Dialog Box

A description of each element and its purpose is presented in the following:

The Main Window

This is the initial displayed window that will be displayed when the user executes the program. All user performed actions is initialized from the menubar of this window.

New Profile Dialog Box

From this dialog box a new profile can be created. The user must specify a profile name and a character name in order to create a new profile.

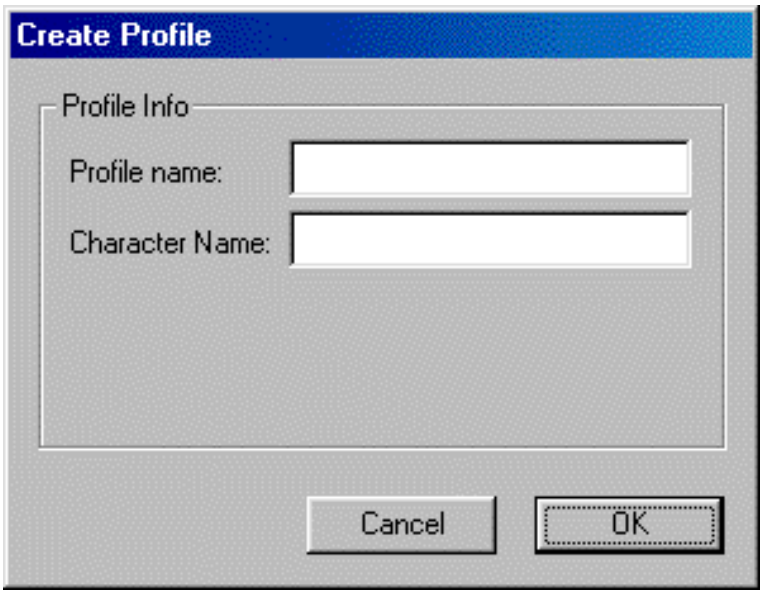

Figure 7.7: The New Profile Dialog Box

The Open Dialog Box

From this dialog box the user may load an existing profile into the system.

The Preferences Dialog Box

This dialog box let the user configure the settings for the loaded profile. These must be set prior to training and can't be changed for that specific profile afterwards.

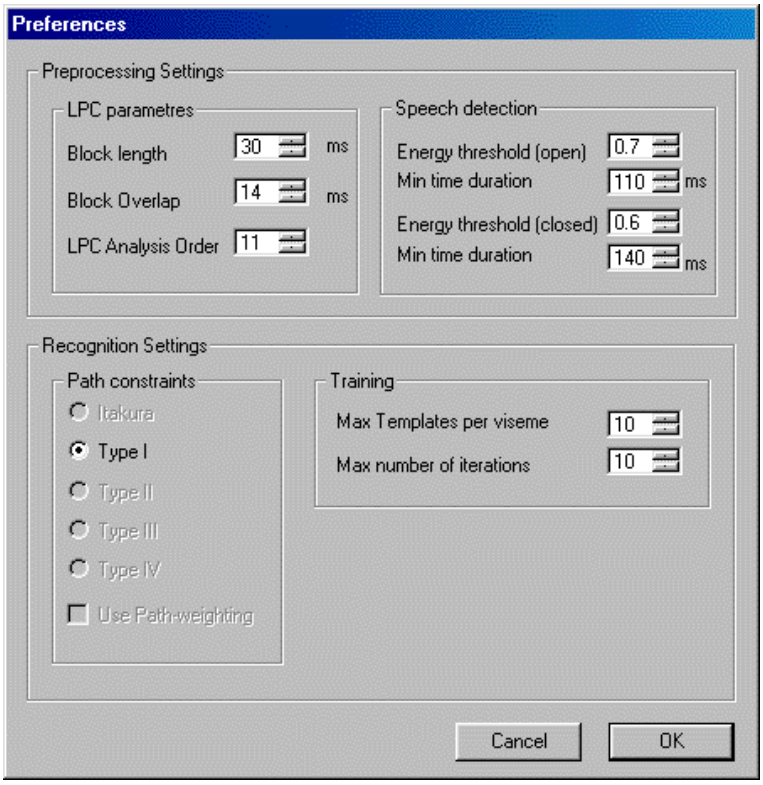

Figure 7.8: The Preferences Dialog Box

The Train Dialog Box

The Train Dialog Box appears when the user starts a training session. From here the user can start the actual training, when an input speech file and the input viseme file has been chosen.

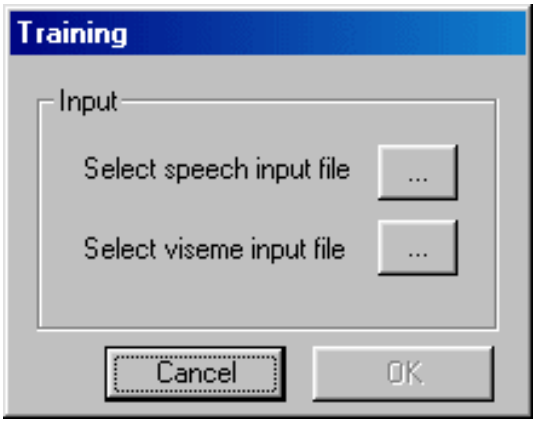

Figure 7.9: The Train Dialog Box

The Train Input Dialog Box

This dialog box lets the user specify the name of the speech input file and the viseme input file to be used for training.

The Train Update Dialog Box

This dialog box lets the user update the profile after training is completed.

The Recognition Dialog Box

The Recognition Dialog Box appears when the user starts a recognition session. From here the user can start the actual recognition, when an input file and output has been chosen.

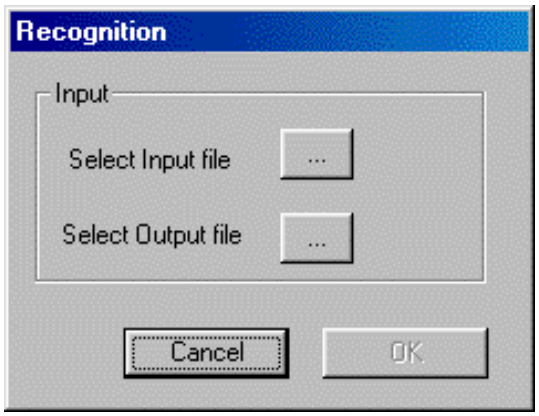

Figure 7.10: The Recognition Dialog Box

The Recognition Input Dialog Box

This dialog box lets the user specify the name of the input file and output file to be used for recognition.

CHAPTER 8

Implementation

This chapter contains the documentation of the implementation of the developed system. First some changes since the design will be discussed and the reason for these changes will be given. The last part of this chapter will be used for elaborating on some of the more crucial parts of the implementation.

8.1 Changes Since the Design

In the design it was specified that the Model component and the GUI component would run in different threads. During the implementation this was not done due to lack of time and resources.

Some things specified in the previous chapter have not been implemented. This was not done because of lack of time and resources towards the end of the project.

The following functions were not implemented:

- The possibility to update an already trained reference database has not been implemented.
- The use of cepstral coefficients in the preprocessor has not been implemented.
- Only one type of path weights and path constraints is implemented in the classifier.
- Error handling is not implemented.

Neither of the above mentioned items will lead to a drastic change of the program and is therefore considered of minor importance. The consequences of these choices is that it is not possible to add to an existing training nor is it possible to continue an ended training and the program will not be responsive while training or recognizing. Furthermore the system can only be tweaked in a limited manner due to missing preprocessor methods, constraint types and path constraints support.

8.1.1 Data format

It has been chosen to represent the speech data to be transferred between the InputSpeechStream and the Preprocessor as floats. This is done to avoid the problems normally associated with fixed point data types, namely:

- Different multiplier depending on the number of significant bits of the input
- Overflow and underflow

To keep the amplitude of the signal in control thoughout the system the input signal is scaled to lie within the range -1 to 1. This is preferable because it makes it easier to define the energy thresholds for the Segmentator.

8.2 Core Elements of the System

The core parts of the system is specified as being the Preprocessor, the Classifier and the Profile clusters. In the following sections issues specific for the implementation of these clusters will be discussed.

8.3 Profile

As the system has been developed using both Intel and Sparc machines portability between big and little endian machines has been a major concern. Since the file format is in little endian format (see appendix A for further information about the file format) it is nessecary to convert the container data before and after accessing the disk for the system to work on a big endian machine. This is done by the containers since only they know the structure of the data. The conversion is implemented using compiler preprocessor macros.

8.4 Preprocessor

Implementing the preprocessor was a two step process. Before actually designing and implementing the preprocessor the algorithm used where prototyped and tested thoroughly using MATLAB. As described in appendix B the algorithm used for feature extraction by Linear Predictive Coding (LPC) can be split into 7 parts all of which are relatively straight forward in nature. Since the LPC parameter conversion was not implemented for reasons explained in section 8.1 only the first five parts of the algorithm where prototyped using MATLAB.

In general there are certain properties of MATLAB which must be taken under consideration before using this program to prototype a given algorithm. The primary cause of problems when using MATLAB is the fact that it is matrix-based. This fact create an array of problems ranging from how to store variables and constants to how to use matrix-based indexes in equations. However after having solved these minor problems MATLAB has proved to be a helpful tool in evaluating algorithms.

Prototype

The following parts of the Linear Predictive Coding algorithm was prototyped in MATLAB:

- Preemphasis
- Frame blocking
- Windowing
- Autocorrelation analysis
- LPC analysis - Levinson-Durbin's algorithm

In general all the algorithms where implemented according to appendix B. In the following problems encountered during the implementation will be discussed.

The first problem occurred when trying to implement the frame blocking algorithm. The problem concerns what to do when it is not possible to block the signal into frames of equal length. The problem occurs when there is not enough data to fill the last blocks. The solution to this problem is to zero-pad the signal until there is enough data. Normally zero-padding will not change the properties of the approximated filter as this problem in general only occur after the last sentence when the signal is approximately zero. In the C++ implementation this is handled by the InputVisemeStream class.

The last two problems occurred in the implementation for the Levinson-Durbin algorithm. The first problem, which is not actually a problem but rather an observation, concerns computational load caused by the algorithm. It seems that the computation of the α variable is not optimal when done recursively as the computational complexity will be $O(p^2)$, where p is the analysis-order. In the light of this discovery and the fact that computation in MATLAB is relatively slow is was chosen not implement the computation of α as a recursion. Instead the computed values of α are stored in a matrix structure, which reduced the computational complexity to $O(p)$.

The final problem concerns the actual algorithm for Levinson-Durbin. It seems that the algorithm presented in appendix B and the algorithm used by MATLAB to compute the LPC coefficients does not produce an equal result. The difference is however not as severe as one might expect as the only difference is a change of sign. However since MATLAB offers insufficient documentation to determine the nature of the algorithm used it has been chosen not to question the algorithm presented by **?**, p. 115 and reproduced in appendix B.

8.5 Classifier

In the following it will be described how the classifier is implemented. The classifier consist of mainly two parts: The training part and the recognition part. The recognition part is a straightforward implementation of the one-pass algorithm described in appendix C and will not be covered here.

8.5.1 Training

The training is performed by collecting the templates corresponding to the different visemes, and afterwards reducing the number of templates by running the templates belonging to a viseme through the Modified K-Means clustering algorithm.

In appendix E.3.2 the MKM was described as :

1. Initialize: $j, i, k = 1$, $\omega_{1,1}^1 = \Omega$ and compute centriod $Y(\omega)$ of Ω
2. Optimal minimum distance classification: Each pattern X_l, for $l = 1, 2, ..., L$ in Ω is labeled by index i according to the minimum distance principle:

$$X_l \in \omega_{j,i}^k \qquad if \qquad \delta(X_l, Y(\omega_{j,i}^k)) = min_{l'} \delta(X_l, Y(\omega_{j,i'}^k)) \tag{8.1}$$

   Sum the total intracluster distance for each cluster $\omega_{j,i'}^k$, defined as:

$$\Delta_i^j = \sum \delta(X_l, Y\omega_{j,i}^k) \tag{8.2}$$

   The summation is overall $X_l \in \omega_{j,i}^k$.

3. Revision of clusters and centroids: Form $\omega_{j,i}^{k+1}$ by grouping all X_l's with label i retrieved from step 2. Compute new centroids for $\omega_{j,i}^{k+1}, i = 1, 2, ..., j$
4. Convergence test: Goto 5 if one of these is fulfilled:
   - $\omega_{j,i}^{k+1} = \omega_{j,i}^k$ for all $i = 1, 2, ..., j$
   - $k = k_{max}$: maximum iteration count reached
   - Change in average (or total accumulated) distance is below predefined threshold Δ_{th}

   else k++ and repeat step 2 - 4.
5. Record the j-cluster solution: Given convergence has been reached, the resultant clusters and centroids, $\omega_{j,i}^k$ and $Y(\omega_{j,i}^k), i = 1, 2, ..., j$ are the j-cluster solution for the training set Ω.

Under the implementation of the MKM clustering algorithm the algorithm was changed slightly from the one described in the appendix E.3.2. The order of some calculations have been changed. The condition deciding when to break out of the k-loop has been simplified; it does not check whether convergense is reached before k_{max} is reached, and it does not use the change in average distance threshold. This was done because these factors are only interresting with much larger amount of training data than the amount currently operated on.

The new algorithm is displayed in the flowchart shown in figure 8.1.

During the implementation it was not clear how to represent the clusters and cluster centers best. It was decided to operate with indexes pointing to templates instead of operating with the templates themselves, because this is much faster, and because the only thing the algorithm needs to work is the distance between two templates. This distance is found by a look-up in a precalculated distance matrix with DTW-scores for each combination of templates.

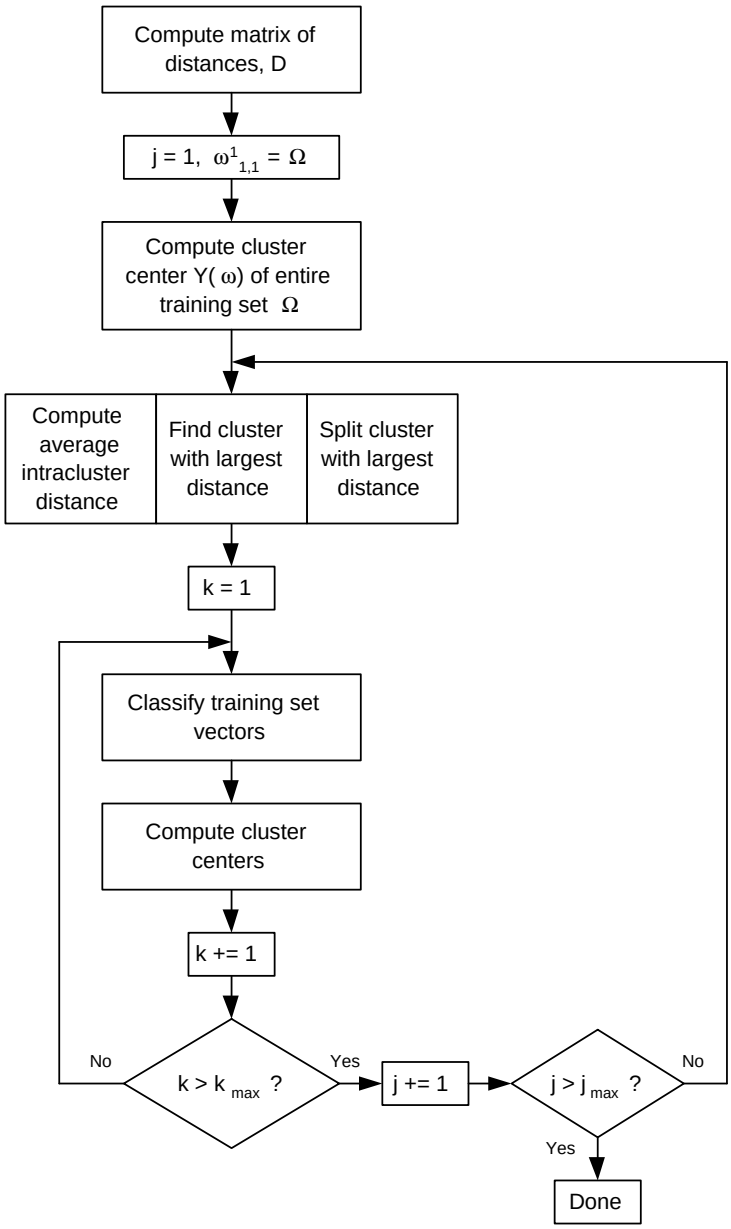

Figure 8.1: Flow chart for the revised Modified K-Means clustering algorithm.

Cluster centers are thus represented as indexes pointing to templates.

Two solutions were discussed for a representation of the clusters:

Solution 1: For each cluster there is a vector with indexes pointing to the templates belonging to it.

Solution 2: A vector with l elements exists, where l is the number of templates. Each element is an index, that tells which cluster the template belongs to.

Solution 1 has been chosen since it saves unnecessary searches when for instance the center of the cluster is to be calculated.

There have been a number of special cases in the algorithm that were hard to spot. These are listed below:

Problem: findGreatestCluster finds a cluster with only 1 template. This occurs because splitGreatestCluster makes two centers pointing to the same template.

Solution: Do not include clusters with less than 2 templates when finding greatest cluster.

Problem: No greatest cluster was found. This occurs because all clusters have max one template.

Solution: Do not subdivide further. This can be avoided by setting the number of wanted clusters to less than or equal to the number of templates - before the actual algorithm.

Problem: splitGreatestCluster chooses the same template for the new cluster centers. This happends because all the templates in the cluster are identical (that is, the DTW-distance is the same).

Solution: Do not assign the same template to two centers.

Problem: computeClusters creates empty clusters. This occurs because some of the input templates are identical. If these two templates are chosen as two new centers, the computation of the clusters would assign all of the surrounding templates to only one of the centers.

Solution: An extra test must be performed to test if the template being associated with a center is the center itself. If yes, associate the template with the center and not with any of the other centers. This guarantees that there will always be at least one template associated with each cluster.

CHAPTER 9

Test

The test of ALSAC is divided into two parts. The first is the component test, which is used to test whether the individual components are implemented correctly. This is done in order to ensure that all the components functions as specified, which is necessary if the second test is to be successful. The second test is the system test. This test is used to determine the performance of the system. Furthermore it is checked whether the requirements stated in the requirement specification have been fulfilled.

9.1 Component Test

The following parts are the most crucial in ALSAC:

- Preprocessor
- Segmentator
- Classifier
- GUI

The following contains a description of how each of these parts of ALSAC have been tested using stubs and drivers and/or prototypes.

9.1.1 Preprocessor

The Preprocessor cluster was primarily tested using the MATLAB prototype described in chapter 8. As previously mentioned the purpose of the prototype was to evaluate the algorithm used and by doing so successfully we gained a powerful tool to use in the implementation and test.

During the driver tests of the Preprocessor cluster the MATLAB prototype proved an invaluable help as it could provide actual values needed to test every step of the preprocessor. The MATLAB prototype was modified to display details for crucial variables before, during and after the five steps of the algorithm. The Preprocessor cluster was likewise modified to display debug information and the prototype values could then be used to detect potential bugs.

9.1.2 Segmentator

In order to test the segmentator a test application was created. Test stubs were also imposed to generate and print test data. As test data does not need to represent actual sound features pseudo random numbers were used instead.

In order to create a reproducible test scenario the random number generator is seeded with a fixed number each time (the default seed).

The following classes were simulated:

- Preprocessor::getFeature();
  - It will return random 1-dimensional features
- InputVisemeStream::getViseme();
  - A random viseme will be generated with a random viseme number and a random time added to the previous time stamp.
- Classifier::recognize(features);
  - This stub will print the feature stream on screen.
- Classifier::update_train(features);
  - The feature stream and the viseme plus its time-range will be printed on screen.
- Classifier::train();
  - This stub will do nothing in the test, but it is called by the Segmentator and therefore included here.

Results

The pseudo random numbers used to test the recognize function is presented in table 9.1. It should be noted that through the following 3 tables the pseudo random numbers are formatted as follows:

[Energy] [Time]

starting	24464 180	4827 360	18716 540
18467 20	5705 200	5436 380	19718 560
6334 40	28145 220	32391 400	19895 580
26500 60	23281 240	14604 420	5447 600
19169 80	16827 260	3902 440	21726 620
15724 100	9961 280	153 460	14771 640
11478 120	491 300	292 480	11538 660
29358 140	2995 320	12382 500	end
26962 160	11942 340	17421 520	

Table 9.1: Pseudo random numbers used to test the recognize function of the segmentator.

Testing the recognize function with an energy threshold of 10000 and time threshold of 2 gives the result presented in table 9.2. Noted that 'finished' marks the separation of segments.

starting	24464 180	14604 420	11538 660
18467 20	5705 200	12382 500	end
6334 40	28145 220	17421 520	
26500 60	23281 240	18716 540	
19169 80	16827 260	19718 560	
15724 100	finished	19895 580	
11478 120	11942 340	5447 600	
29358 140	finished	21726 620	
26962 160	32391 400	14771 640	

Table 9.2: Result of testing the recognize function of the segmentator.

In table 9.2 it should be noted that:

- A single feature lower than the threshold do not separate segments.
- The features before and after a separation are all above the threshold - Apparently the segments are successfully chopped without silence.

Consequently the test passed, because both item one and two are fulfilled.

Testing the train function gives the result presented in table 9.3. Noted that a different set of random numbers are used in this test as various other values also needs to be randomized in order to thoroughly test the train function. The values needed in addition to those already determined are: segment start time, segment end time, and viseme number. The generated values will appear in connection with the result of the test in table 9.3.

starting	23281 240	11942 340	segment start: 486
segment start: 132	16827 260	4827 360	segment end: 576
segment end: 269	segment finished	5436 380	viseme: 4
viseme: 0	segment start: 269	32391 400	12382 500
29358 140	segment end: 486	14604 420	17421 520
26962 160	viseme: 24	3902 440	18716 540
24464 180	9961 280	153 460	19718 560
5705 200	491 300	292 480	segment finished
28145 220	2995 320	segment finished	

Table 9.3: Result of testing the train function for the segmentator.

In the test of the train function it should be noted that each feature in a segment is within the range of the corresponding viseme, therefore it most be concluded that the test is passed.

9.1.3 Classifer

Training

The training mainly consists of an implementation of the MKM clustering algorithm. Since the MKM clustering algorithm uses the DTW algorithm, the implementation of the DTW also has to be tested.

DTW

At an early stage in the project period the DTW algorithm has been implemented and tested in MATLAB. This created great possibilities to plot the local distance matrix of the algorithm

to gain understanding of what is going on inside the algorithm.

The C++ DTW implementation has been tested with a test driver. The test driver is a console program, that takes two strings as parameters. The strings are converted to one-dimensional feature-vectors by their ASCII-value. The converted strings are then compared by the DTW algorithm which returns a dissimilarity score.

The implementation has been tested with various strings. When the two strings are equal the returned score is zero. Streching one of the strings by repeating some of the letters also results in a zero score (as it should when the path constraints are Type I). The less similar the strings are, the worse the score.

By the above observations and by logging the computations of the test it has been concluded (as a black box test) that the DTW implementation is working.

MKM Clustering

The MKM clustering implementation has been tested with a test driver. The test driver calls the clustering with a number of templates with one two-dimensional vector in each template. The two-dimensional vectors which allows for plotting the vectors on the screen for easy testing of different test cases. The screen-shot in figure 9.1 shows a test case with 300 templates and the 6 templates (centers) from the clustering representing them.

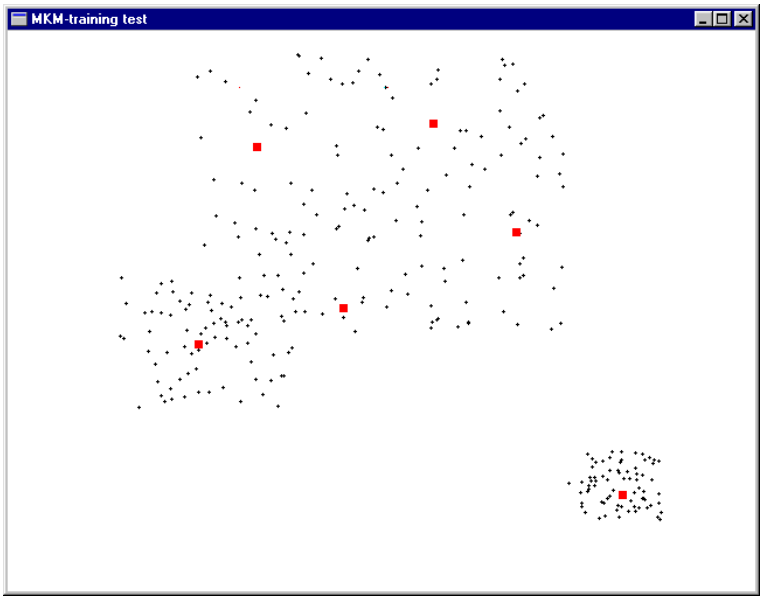

Figure 9.1: Screen shot of the test driver for the implementation of the MKM-clustering algorithm.

The training has been tested with various extremes. Under this test a number of bugs was found and fixed in the implementation of the algorithm, as described in 8.5.1.

With several extreme cases tested the implementation has been considered correct (as a black box test).

Recognizer

Since the recognizer is based on the One-Pass algorithm, this algorithm has been implemented in a separate class and tested with a test driver.

The driver is a stand-alone console program that recognizes text strings. The reference database (the templates) of the driver are the strings “zero”, “one”, “two”, “three”, “four”, “five”, “six”,

"seven", "eight", and "nine" which are converted into one-dimensional feature vectors by their ASCII-value.

The test driver takes a test string as input and tries to recognize the above strings in it. When using the test driver, the following was observed:

When typing perfect test strings, the test driver program performs as expected. When typing words from the reference database and stretching them (that is, repeating a letter one or more times), the program also recognizes the words correctly. When typing words from the reference database and changing some of the letters with letters with a close ASCII-value, the program mostly recognizes the words correctly.

With these observations together with a log-file with the computations of the algorithm it has been concluded as a black-box test that the implementation of the One-Pass algorithm is working.

9.1.4 GUI

The User Interface component was tested by imitating the various use patterns and comparing the hereby produced responses to the expected ones. Thus almost every possible combination of actions from the user has been observed. The test also included verifying the correctness of the states for all elements that the GUI is build up from. This means that the states of bottoms, menus, edit boxes etc. were observed in several different situations. The User Interface was found to behave as specified. The test was performed iteratively during the development process and completed by a final test for correct behavior after ended implementation. Some of the software objects used in the GUI did, though behaving correct, not perform as user friendly as intended. However due to lack of time and because the GUI component is not of great importance to the system wanted by ITE, these issues has not been addressed.

9.2 System Test

This section presents the three tests specified in the Test Specification, chapter 5. Furthermore it is checked whether the requirements specified in the requirement specification have been met. The tests will be described in detail and the results will be presented. Technical merit will be given by the HTK tool HResults. In order to use this tool, the viseme label files must be converted into a format HResults can read. For this purpose a utility called "vis2test" has been developed by the project group. The utility converts a reference viseme file and a recognized viseme file into files that can be used in HResults.

Throughout the system test the individual files from EUROM1 used in each test will mentioned in connection with the specific test. Refer to the documentation for the EUROM1 database for more detailed information concerning these files.

9.3 Checklist Test

The purpose of the Checklist Test is to check that the statements made about the system in the requirement specification have been fulfilled by the program.

The following tests are performed:

Run Under Window9x
: This demand is tested by running the training and recognize session in a Microsoft Windows 95 or 98 environment. If the system does as specified the test is to be considered passed.

Load And Save Profiles
This demand is tested by starting the program, creating a new profile, saving it, shutting the system down and then try to open the save profile in a new session. The test is passed if the previously saved profile matches the newly loaded one.

Update Profile
This demand is tested by creating a profile, starting a training session with the data specified in the profile and then try to train the profile with another data source. The test is passed if the system updates the profile successfully.

Consistent Output Format
This demand is tested by training the system, recognizing a speech file and then comparing the output file to the file format specified in appendix A. The test is passed if the output file respect the guide lines specified.

Test	Passed	Failed
ALSAC can run under Windows 9x	X	
ALSAC can load and save profiles	X	
ALSAC can update a profile		X
ALSAC give a consistent output file	X	

Table 9.4: Results of the checklist test

The results of the checklist test are displayed in table 9.4. From this it can be seen that the system fulfills three out of four demands specified in the requirement specification. The reason of ALSAC not being able to update a profile with new training data is explained in section 8.1 in the implementation chapter.

9.3.1 Base System Test

The purpose of the base system test is to show that the base system is working, so that any further testing of ALSAC is reasonable. The purpose is not to test the recognition performance of ALSAC.

The test is first and foremost done using one passage per speaker for both training and recognition. In addition to this test the effects of changing the number of passages used in training are also tested.

For the two above mentioned tests the following files from EUROM1 are used:

One passage
Training and recognition - anq00075.pds.

Four passages
Training - anq00075.pds, anq10076.pds, anq20077.pds and anq30078.pds.
Recognition - anq00075.pds.

The output test files will be compared to the reference label files by the HTK HResult test program. The output of HResult must be set to be compatible with NIST. By default it returns an accuracy while NIST returns an error percentage.

The maximum accuracy of the tests must be above 90% to pass.

The results of the base system is displayed in table 9.5. From these results it can be concluded at the system does not perform as well as expected. The reason for this can partly be explained by the lag of optimal training data. The data used in the training process was generated by the

Test	LPC Order	Frame length	Overlap	# Files	# Clusters	Correct	Accuracy
1	18	15 ms	1/2	4	8	40%	27%
2	18	15 ms	1/2	1	4	48%	39%
3	18	15 ms	1/2	1	16	74%	64%

Table 9.5: Results of the base system test.

'lab2vis' tool. This tool takes a phoneme-labeled file and translates it into a less than prefect viseme file. It is expected that the base system test will provide a better result if the training data were optimal.

9.3.2 Performance Test

The performance test is performed using ALSAC on a speech stream which was not used during the training. The resulting viseme file is compared with the matching viseme file generated by lab2vis. It is specified that a test table must be made using variable amount of training data and variable parameters. Only one passage will be recognized.

The following files from the EUROM1 database are use in the performance test:

One passage
Training - anq00075.pds.
Recognition - anq90084.pds

Four passages
Training - anq00075.pds, anq10076.pds, anq20077.pds and anq30078.pds.
Recognition - anq90084.pds.

Nine passages
Training - anq00075.pds, anq10076.pds, anq20077.pds, anq30078.pds, anq40079.pds, anq50080.pds, anq60081.pds, anq70082.pds, and anq80083.pds.
Recognition - anq90084.pds.

The output test file will be compared to the reference label files by the HTK HResult test program. The maximum accuracy must be above 80% to pass.

There are five parameters to tweak the recognition performance; LPC order, frame length, frame overlap, number of files to train on, and number of clusters for each viseme. To find the optimal settings various combinations are tested using HResult.

In addition to tweaking the basic parameters the parameters of the segmentator are also tweaked for better silence detection. This is done by comparing the energy output file and the viseme file of various speakers not used in the test. The optimal parameters were from these tests determined to be:

Opened:
Time threshold: 3 blocks
Minimum energy level: $6 \cdot 10^{-8}$

Closed:
Time threshold: 3 blocks
Maximum energy level: $2 \cdot 10^{-8}$

It should be noted that a bad combination of the parameters will cause many insertions (seen as flickers) in the output viseme file.

As many tests last for hours the number of tests are limited. Table 9.6 shows the tests divided into three phases:

A. Test of which ratio of overlap is best. Values of 1/2 and 1/3 will be tested. The LPC order is set at 14 and the frame length at 25 ms. These values were used because a higher order would mean unreasonably long computation times. The best overlap will be used in the remaining test phases.
B. Test of which combination of LPC order (14, 18, or 22) and frame length (15, 20, or 25 ms) performs best. The best combination will be used in the remaining test phase.
C. Test of which combination of number of files to be trained (1, 4, or 9) and number of clusters (4, 8, or 16) are best. The best combination is considered the best result.

By using the approach defined above the number of tests can be limited to (the parentheses mean the number of e.g. LPC orders to be tested):

$$\begin{aligned} tests &= (LPC\ orders) \cdot (frame\ lengths) + (files\ to\ train) \cdot (clusters) \\ &\quad +(overlaps) - (repeated\ tests) \\ tests &= 3 \cdot 3 + 3 \cdot 3 + 2 - 2 \\ tests &= 18 \end{aligned} \tag{9.1}$$

Otherwise the tests needed would have been:

$$\begin{aligned} tests &= (LPC\ orders) \cdot (frame\ lengths) \cdot (files\ to\ train) \cdot (clusters) \\ &\quad \cdot(overlaps) \\ tests &= 3 \cdot 3 \cdot 3 \cdot 3 \cdot 2 \\ tests &= 162 \end{aligned} \tag{9.2}$$

Phase	Test	LPC Order	Frame length	Overlap	# Files	# Clusters	Accuracy
A	1	14	25 ms	1/2	4	8	16%
	2	14	25 ms	1/3	4	8	5%
B	3	14	15 ms	1/2	4	8	18%
	4	14	20 ms	1/2	4	8	16%
	5	18	15 ms	1/2	4	8	22%
	6	18	20 ms	1/2	4	8	18%
	7	18	25 ms	1/2	4	8	18%
	8	22	15 ms	1/2	4	8	22%
	9	22	20 ms	1/2	4	8	22%
	10	22	25 ms	1/2	4	8	16%
C	11	18	15 ms	1/2	4	4	22%
	12	18	15 ms	1/2	4	16	19%
	13	18	15 ms	1/2	1	4	25%
	14	18	15 ms	1/2	1	8	20%
	15	18	15 ms	1/2	1	16	14%
	16	18	15 ms	1/2	9	4	23%
	17	18	15 ms	1/2	9	8	18%
	18	18	15 ms	1/2	9	16	27%

Table 9.6: Performance test matrix.

The result of the test is as displayed in table 9.6. From these results it can be seen that the system does not perform nearly as well as expected. There are various reasons why this is so. The results can primarily be explained by the criteria for success not being realistic.

When the criteria for recognition accuracy was determined in the requirement specification it was expected that a viseme based recognizer could perform approximately as well a phoneme based system. However phoneme based systems has the obvious advantage of knowing the exact number of sounds to associate with each template (1 to 1), which is not the case for the viseme based recognizer. Taking into account that the training mechanism implemented uses a fixed number of clusters to represent each viseme, there is a certain likelihood that the system is either over- or under-trained with respect to some of the visemes.

The inherited problems concerning over- and under-training constitute a problem when comparing the system with other systems for speech recognition, but it will not necessarily influence the visual aspects of the recognition. The following test will determine how the system performs visually.

9.3.3 Visual Test

Even though the visemes are not correctly recognized the viseme stream might look acceptable for a human being. Due to this a visual test will be made by an impartial group. This test has higher importance than the prior tests, since it is more important that the output looks good on the animated character than that the output is correctly recognized.

The group will evaluate the following set of viseme streams in random order while listening to the matching audio stream:

- A viseme stream generated by ALSAC
- A viseme stream generated by lab2vis
- A random generated viseme stream
  - When generating the random viseme stream the test program reads the silence boundaries in the lab2vis reference file. When the speech begins it fills a segment with random visemes of random durations until silence begins again. Thus the number of visemes will be random as well.

These are to be graded by the group on a scale from 1 to 5 where 1 is the worst performance and 5 is the best performance. 10 passages are available per speaker for the test.

ALSAC will be trained using 9 passages per speaker which are concatenated. The reference viseme files are concatenated using a tool developed by the project group called viseme_combiner. 3 speakers will be used. Each training set will be stored in profiles corresponding to each speaker. The last passage will be presented to the test group with each of the above mentioned three viseme files.

The speaker sets used for the test are:

- anq0*.pds-anq8*.pds for training and anq9*.pds for recognition
- blq0*.pds-blq8*.pds for training and blq9*.pds for recognition
- ebo0*.pds-ebo8*.pds for training and ebo9*.pds for recognition

The parameters of the preprocessor and the classifier were: 16 clusters, 9 files, 16 LPC, 1/2 overlap, and 20 ms frame.

Each of the test members will be given a test sheet which can be seen in appendix G. The correct order will be as shown in table 9.7.

	Speaker 1	Speaker 2	Speaker 3
A	Random	Recognized	Reference
B	Reference	Reference	Random
C	Recognized	Random	Recognized

Table 9.7: Order of test viseme streams.

The members of the test group will evaluate one by one not being able to discuss the results. They will be equipped with a test sheet, a pencil, and a set of headphones. When evaluating a speaker, they will view all three viseme streams without giving grades. Afterward they will view the streams again and give the grades.

The instructions are as follows:

> *There are 3 speakers in total. For each of these there are three recognized viseme streams. You must decide which viseme stream creates the best lip synchronized animation. First, you will see the three viseme streams without voting. Next you'll see them again, where you must give each a grade from 1 through 5, and 5 is the best.*

The results will be processed as follows:

- The average grade for each of the viseme streams are calculated.
- The standard deviation for each of the viseme streams are calculated.
- Evaluation of how ALSAC has performed in comparison to the reference and the random.

The standard deviation is calculated to see whether the test group has a mutual understanding on how good the different streams perform. It is clear that the smaller the standard deviation is the more unanimous the test group is on how well the stream performed.

The standard deviation is calculated as:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{N}(\mu - x_i)^2}{N-1}} \tag{9.3}$$

Where μ the average value and N is the number of elements, and x_i is the current grade.

Test Results

Test	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Average
Random	2	4	3	2	2	3	2.66
Reference	3	5	4	4	5	2	3.83
ALSAC	4	2	2	3	3	3	2.83

Table 9.8: Recognition performance test for stream 1

It can be seen from the standard deviation test results (tables 9.9, 9.11, and 9.13) that it is difficult for the test group to make an unanimous decision on how good the system performs because the standard deviation is rather big.

Inspecting the results all together it can be seen that the reference visemes perform best in all tests as expected. However looking at the tests individually two out of three tests shows a slightly better performance for ALSAC compared to the random generated visemes (see tables

Test	Standard Deviation	Correctness	Accuracy
Random	0.8165	32	4
Reference	1.1691	100	100
ALSAC	0.7653	40	19

Table 9.9: Recognition performance test for stream 1

Test	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Average
ALSAC	1	3	2	1	3	3	2.16
Reference	3	5	5	3	4	2	3.66
Random	4	3	2	3	2	3	2.83

Table 9.10: Recognition performance test stream 2

Test	Standard Deviation	Correctness	Accuracy
ALSAC	0.9832	32	22
Reference	1.2111	100	100
Random	0.7528	25	10

Table 9.11: Recognition performance test stream 2

Test	Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Average
Reference	3	4	5	4	2	4	3.66
Random	2	3	3	2	1	2	2.16
ALSAC	1	3	2	3	3	3	2.50

Table 9.12: Recognition performance test 3

Test	Standard Deviation	Correctness	Accuracy
Reference	1.0328	100	100
Random	0.8367	24	17
ALSAC	0.6056	28	19

Table 9.13: Recognition performance test 3

9.8, 9.10, and 9.12). Some explanation is in place here, because the word "random" is associated with implicit expectation to a poor performance. Thus the fact that the randomly generated visemes outperforms ALSAC in one of the tests may seem disappointing.

The word "random" is however misleading in this context, because actually the "random" visemes are generated with time indexes taken from the reference visemes. This means that the correct start and endpoints for silence is "known" by the "random" visemes, while these are computed and therefore unknown for ALSAC. Thus this "knowledge" gives the "random" generated visemes a considerable advantage which should be taken into consideration when judging the system's immediate poor performance.

Furthermore it became obvious that the test persons evaluate the performance from quite different criteria. This was discovered by interviewing the testers afterwards. However, one common characteristic (drawback) seemed present among the testers, namely that of flickering. Flickering occurs when inserting a viseme of short duration whose distance to the previous viseme is large (e.g. shifting from closed to open mouth to closed again swiftly).

When the number of such insertions is quite seldom (which is the case for ALSAC - as can be seen from the HResult scores) compared to the number of insertions made by the random generated visemes, these insertions are clearer noticed by the test persons, because of their more seldom occurrence. Thus it creates the paradox that the test persons interpret the visemes generated by ALSAC as being wrong, when many of them actually are correct recognized.

Finally a bug in the implementation of the segmentation algorithm (end point detection) was found. The bug causes the detection of starting and ending frames to be shifted in time by the value of the time duration threshold parameter (3 frames of 10 ms in the test case). Unfortunately this was first discovered after the tests.

It is important to note that the visual test was performed without tweaking the performance of the system. This is due to lack of time as the performance tests lasted for hours.

Debugging the system revealed that the training process stored lots of templates being mere 3 frames long. They could cause the recognition to accept many short visemes because one actual (large) viseme could be divided into a number of shorter visemes (of 3 frames). It is more likely for the one pass algorithm to be able to match a series of short templates to the input than to match long templates that are up to 10 times longer.

Another problem is the way the templates are created. As the templates originate from a connected stream of feature vectors, the endpoints of the templates can easily contain some information (partly) belonging to another viseme. This could also explain some of the flickering between some templates as the decisions at template boundaries become rather random and short templates are filled in because they fit best.

Another fact that could explain that the One-Pass algorithm prefers short templates is that warping slopes greater than 1 are not preferred by the standard algorithm (as described by [**?**]). A possible solution for this is slope weighting, which has not been used because of low time resources at the end of the project.

CHAPTER 10

Conclusion

This chapter is the conclusion of this report and the developed system. First a short summary of the report will be given and then it will be concluded if the objectives for the project has been met and if the developed system meets the requirements.

First an overview of the common theories used in speech recognition was given. This includes human speech production and perception. Then varies techniques both for preprocessing speech and classifying visemes were documented. Next the requirement specification for the system to be developed was introduced. After the requirement specification the decisions on which approach to adapt for the implementation was chosen based on the theories discussed in chapter 2 and the requirement specification. From this point a test specification was constructed. With this in place the analysis and design together with the complex parts of the implementation were described in the analysis, design, and implementation chapters accordingly. Finally, the system was tested against the requirements, which was documented in the test chapter.

10.1 Formal Requirements to the Project

The requirements from the Study Board for the 6th semesters project were as stated in chapter 1:

> *On this semester the focus is to gather, represent and process abstract knowledge. With a starting point in a specific problem description the student should work with gathering the necessary information and represent it on abstract form. This involves that the student obtains knowledge on how the information is extracted from the information carrying signals, how this information can be represented as symbols and how these symbols can be processed. The interesting information would typically originate from physical signals but can also be available in another forms.*

It is concluded that the project group has met the requirements stated above, because a system including these aspects of information processing and retrieval actually has been developed. The physical signals (speech signals) are available in sampled form and the visemes used can be interpreted as an abstract representation of speech signals. The abstract information is processed, classified and afterwards presented to the user of the system. Thus all aspects described are included in the developed system and consequently the formal requirements are fulfilled.

10.2 The Developed System

A system performing the desired task of processing a speech signal classifying the contained sequence of visemes and corresponding time stamps has been implemented and tested. The system has in general been found to implement the correct task, but lags the sufficient recognition accuracy stated in the requirement specification. In this context it is important to note that the relatively high accuracy demands were proposed by the project group and not by ITE. Thus lagging this requirement do not imply that the developed system is not applicable for ITE in their animation process. On the contrary the system is believed to contain the ability to accelerate the synchronization process in a usable degree, compared to the present manual approach. However when this is said, it must also be clearly stated that some of the more exclusive features lags implementation. It is by reason of these yet unexplored possibilities believed that the system persists potential for higher accuracy scores than achieved by the system in its present state. Thus the next section provides a description of the suggestions made by the project group to further improve the present system.

10.3 Future Improvements

As it clearly was stated in the previous section the achieved system holds place for numerous updates that yet remains to be implemented. In order to supply ITE with the best possible idea of issues that can increase the system's recognition accuracy, a list of suggested improvements are given here.

- Interpolation between visemes to improve the final visual impression
- Support for repeatedly training of profiles.
- Incorporation of various path constraints and slope weighting to improve distance calculations.
- Incorporation of a grammar to reduce the effect of flickering
- Incorporation of better distance measures, e.g. use of cepstral distance measure.
- Manual tuning and optimization of the system specific parameters are expected to improve the overall performance of the system.
- Template stretching in order to compensate for errors introduced by short templates.
- Removal (avoidance) of too small templates. Perhaps incorporation of a template time duration threshold.

Hopefully this will be helpful for the programmers at ITE to adapt the system to future requirements in the their continuous development.

APPENDIX A

File Format Documentation

This appendix contains information about the file formats used by ALSAC. Note that it is only the formats designed for this project and not the standard formats such as the EUROM1 files.

Two formats will be described:

- The viseme format
- Profile format

A.1 Viseme Format

When defining a viseme format the following issues must be considered:

- File type
- Identifiers
- Time unit
- Viseme representation

A.1.1 File Type

There are two major file types when implementing the viseme format:

Type	Advantage
1. Binary	Smaller files, protected data
2. ASCII	Easy to create test files and view test-results for debugging

Type 2 (ASCII) is chosen because the task at hand is testing and debugging.

A.1.2 Identifiers

It must be considered whether the viseme format require a header identifying the file type. As type 2 is chosen this header should be readable like a comment. Three lines are reserved as a header while the viseme stream is ended with a '*' so that it is possible to write comments afterwards.

Another identifier that must be defined is the separator between time stamps and visemes. Each set of a time stamp and a viseme is defined as: [time][space][viseme][newline]

A.1.3 Time Unit

The LPC frames are defined as being 15-30 milliseconds, which makes it not feasible to use a resolution lower than milliseconds. The resolution could be test-frames instead of millisecond, but then it would be less readable. Consequently, the time unit is chosen to be milliseconds.

A.1.4 Viseme Representation

The symbols used for visemes in appendix F table F.4 is chosen to represent the visemes in the viseme format. This way it is easy to read the files because it is consistent with the viseme documentation.

A viseme file could look like:

```
**********************
* Generated by ALSAC *
**********************
0 0
60 2a
200 13
520 1
600 10
840 4
960 0
************************
* End of example file *
************************
```

A.2 Profile Format

The profile file format must be able to store different kinds of data while also supporting future extensions of the program and the file format. This flexibility can be achieved by using a chunk-based file format which has the following form:

- A *chunk* usually consists of a *header* and some *data*
- The *header* usually consists of an *ID* of four chars and a *length* of the data stored in an unsigned int
- The *data* can be anything, for example a number of other chunks

As the strategy design pattern has been used extensively different kinds of input speech streams, input viseme streams, segmentators, preprocessors, classifiers, and output viseme streams have to be stupported in the file format. Therefore after the standard chunk header there is a *Container type* field (stored as an unsigned int) to tell which kind of the above is being used, and hence which kind of containers should be instantiated.

The structure of the profile file format can be seen in figure A.1. The chunks inside the *ALSC* chunk can lie in any order and eventual undefined chunks should be expected and ignored.

The contents of the chunks in the *ALSC* chunk are implementation specific to the different kinds of containers. The structure of the *CLAS* chunk, however, is essential if any other software is

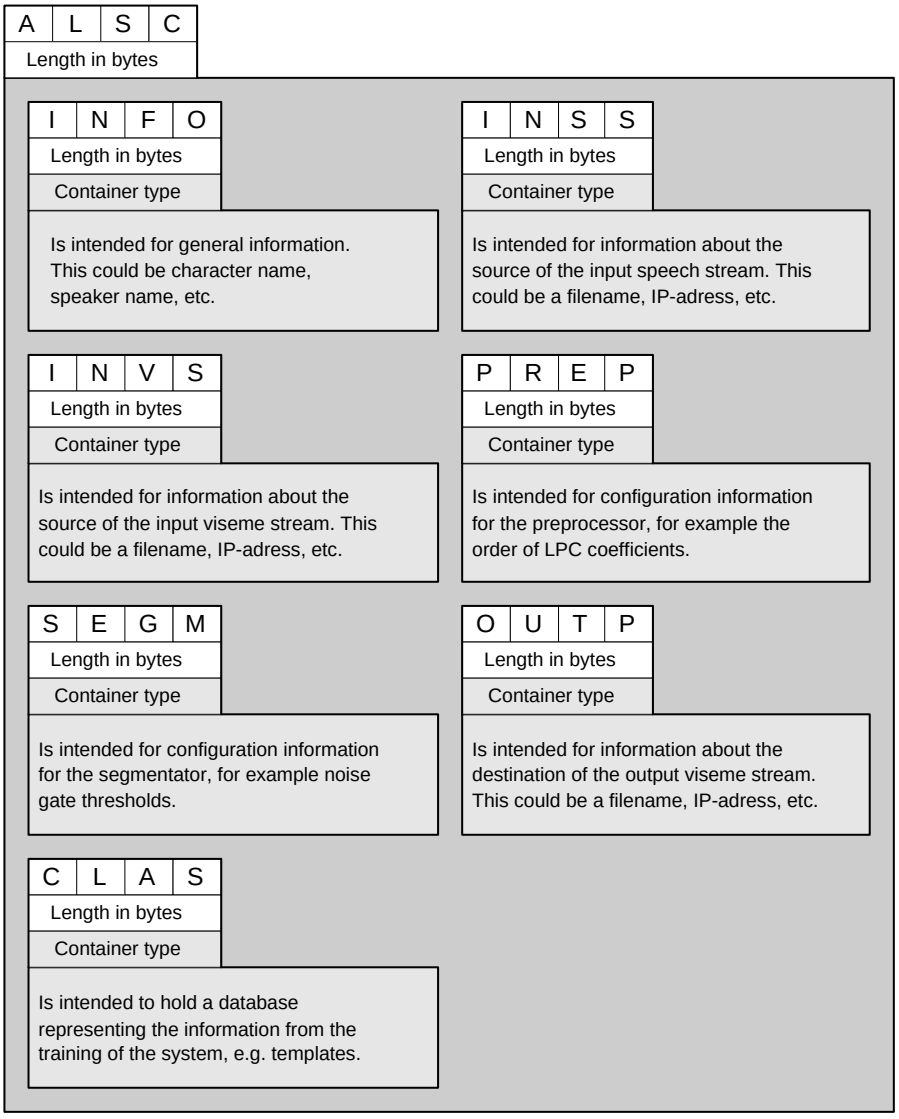

Figure A.1: The profile file format

to read the reference database of the current system. The structure of the *CLAS* chunk when the *Container type* is *CLASSIFIER_TEMPLATEBASED* (defined as en enum in the .h file for the BaseContainer) is explained in figure A.2.

It should be noted that the file format is little endian.

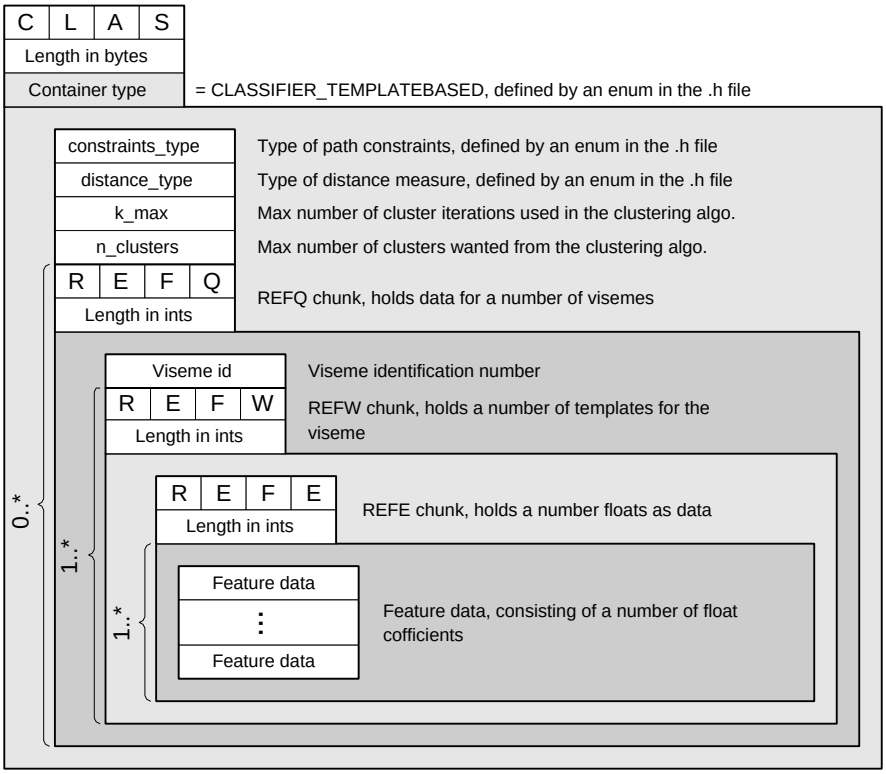

Figure A.2: The classifier chunk

APPENDIX B

Feature Extraction using Linear Predictive Coding

The purpose of this appendix is to explain and expand the theory of the Linear Predictive Coding (LPC) model for feature extraction presented in section 2.4.2. Furthermore an algorithm for implementing a LPC front-end processor for speech recognition will be described. Note that this document is primarily based on the theory presented in [**?**] unless otherwise stated.

B.1 The LPC Analysis

The strength of the LPC model is its ability to model a speech signal through the associated vocal tract characteristics thereby producing a robust and parsimonious representation of the given signal. It is done by approximating the all-pole filter that best fits the signal spectrum of a given frame of speech samples that can be considered relatively stable. The actual extraction of the coefficients a_k of the digital filter is based on the idea that a given speech sample at time n can be approximated by a linear combination of the p past speech samples.

$$s[n] \approx \tilde{s}[n] = \sum_{k=1}^{p} a_k s[n-k] \tag{B.1}$$

However, this interpretation leaves the possibility to make a prediction error, $e[n]$, which at a given time n is defined as

$$e[n] = s[n] - \tilde{s}[n] = s[n] - \sum_{k=1}^{p} a_k s[n-k] \tag{B.2}$$

The solution for extraction of a set of optimal coefficients for the digital filter is to minimize the prediction error of the entire speech frame consisting of m samples. This is represented by the mean squared error signal E:

$$E = \sum_{m} (s[m] - \sum_{k=1}^{p} a_k s[m-k])^2 \tag{B.3}$$

In order to find the optimal coefficients $\hat{a}_k$ the error signal E will have to be differentiated with respect to each a_k and these equations will have to be solved for each of the results equal to zero. Doing so gives the following result:

$$\sum_m s[m-i]s[m] = \sum_{k=1}^{p} \hat{a}_k \sum_m s[m-i]s[m-k] \tag{B.4}$$

This equation can be solved using autocorrelation and the Levinson-Durbin algorithm in order to get the actual LPC-coefficients. The way in which this is done is presented in the following algorithm for an LPC front-end processor for speech recognition.

B.2 Algorithm for the LPC Front-End Processor

In section 2.4.2 the generalized algorithm for implementing the LPC front-end processor presented in figure B.1 was briefly reviewed.

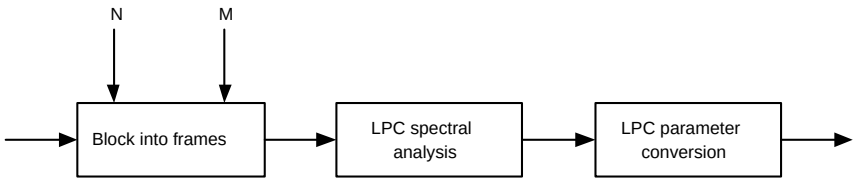

Figure B.1: LPC analysis model [?, p. 72]

This three-step algorithm can easily be expanded to a 7-step algorithm, which also encompasses the pre- and post-processing necessary in a speech recognition scheme. This algorithm will be the focus the following text and is illustrated in figure B.2.

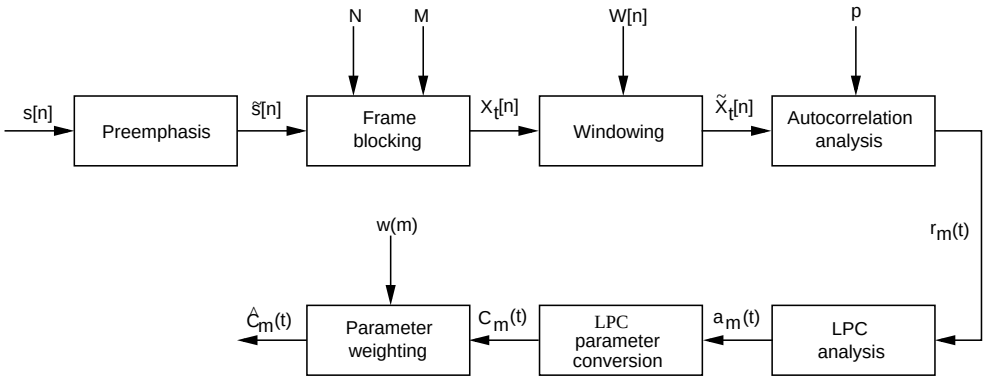

Figure B.2: Block diagram of the LPC processor for speech recognition [after ?, p. 113]

Pre-Emphasis

The purpose of this first step in the processor is to compensate for the uneven dispersion of signal amplitude within the given frequency spectrum. This is done by dampening the low frequency sounds and accentuating the high-frequency sounds thereby creating more flattened spectral characteristics of the signal $s[n]$. The actual pre-emphasizing is done by a fixed or adaptive first-order FIR filter. Since adaptive filtering is not relevant within the scheme of this project only the fixed value implementation of the filter will be used. The difference equation of the fixed first-order FIR filter is as follows:

$$\tilde{s}[n] = s[n] - \tilde{a}s[n-1] \qquad \text{for } 0.9 \geq a \geq 1.0 \tag{B.5}$$

In equation B.5 $\tilde{s}[n]$ represent the pre-emphasized speech signal and $\tilde{a}$ is the filter-coefficient which is commonly set to 0.95.

Frame Blocking

As mentioned in the generalized algorithm described in section 2.4.2 the speech signal is blocked into frames in which the spectral contents of the signal can be considered approximately stable. Note that in this algorithm it is the pre-emphasized speech signal that is divided into frames. The actual frame-blocking technique is the same as described in 2.4.2. As shown in B.3 the pre-emphasized speech signal is blocked into a finite number of adjacent frames with a length of N samples and overlaps by $N - M$ samples. The overlap by $N - M$ samples adjacent frames is used to ensure full signal coverage through the following spectral analysis.

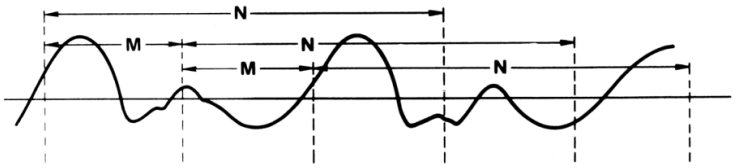

Figure B.3: Blocking of speech into overlapping frames [**?**, fig. 3.39]

The actual algorithm used for dividing a given pre-emphasized signal into L frames of length N is defined as follows:

$$x_l[n] = \tilde{s}[Ml + n], \qquad n = 0, 1, ..., N-1, \qquad l = 0, 1, ..., L-1, \tag{B.6}$$

with $x_l[n]$ denoting the l^{th} frame of speech. The frame blocking has been found to provide the best result with the frame length N and frame slide M set respectively at 45ms and 15ms at $6.67kHz$ [**?**, p. 284].

Windowing

In order to compensate for the signal discontinuities caused by the speech frames overlapping each of frames will have to be windowed. By windowing each speech frame the discontinuities at the beginning and end for each frame can be minimized thereby ensuring an approximately continuous signal. When using autocorrelation to perform the actual LPC-analysis a commonly used window $w[n]$ is the "Hamming" window. Doing so the algorithm will look as follows:

$$\tilde{x}_l[n] = x_l[n]w[n] = x_l[n](0,54 - 0,46cos\Big(\frac{2\pi n}{N-1}\Big)) \qquad 0 \geq n \geq N-1. \tag{B.7}$$

Autocorrelation Analysis

The first step in extracting the actual LPC-coefficients is to define each windowed speech frames through a series of m autocorrelations $r_l(m)$ ending at the desired order p for the LPC-analysis:

$$r_l(m) = \sum_{n=0}^{N-1-m} \tilde{x}_l[n]\tilde{x}_l[n+m], \qquad m = 0, 1, ..., p. \tag{B.8}$$

In speech recognition schemes the order p of the LPC-analysis is usually in the range of 8 to 16 with actual value depending on the sample-rate of the speech signal.

When using the autocorrelation in a speech recognition scheme the zeroth autocorrelation $r_l(0)$ of the l^{th} frame is notable as it denotes the energy of that frame. Knowing the energy of each frame will prove useful in future attempts of endpoint-estimation.

LPC Analysis

The final step in extracting the LPC-coefficients is to convert the $p+1$ autocorrelation-coefficients into a set of LPC parameters from which the LPC-coefficients can be read. This procedure is based on the relationship between the autocorrelation coefficients (r_l) and optimal LPC coefficients ($\hat{a}_k$):

$$\sum_{k=1}^{p} r_l(\mid i-k \mid)\hat{a}_k = r_l(i), \qquad 1 \geq i \geq p \tag{B.9}$$

Equation B.9 can also be expressed as a Toeplitz[1] matrix to which the Levinson-Durbin algorithm can provide a solution. This algorithm can best be defined by a series of recursive equation that must be solved for $i = 1, 2, ..., p$.

$$\begin{aligned}
E^{(0)} &= r_l(0) && \text{(B.10)}\\
k_i &= \Big\{r(i) - \sum_{j=1}^{L-1} \alpha_j^{i-1} r_l(\mid i-j \mid)\Big\} \Big/ E^{i-1} && \text{(B.11)}\\
\alpha_i^{(i)} &= k_i && \text{(B.12)}\\
\alpha_j^{(i)} &= \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} && \text{(B.13)}\\
E^{(i)} &= (1-k_i^2)E^{i-1} && \text{(B.14)}
\end{aligned}$$

The actual LPC-coefficients a_m for a given LPC-analysis order p is denoted as:

$$a_m = \alpha_m^{(p)} \tag{B.15}$$

Since the LPC-coefficients are the normalized filter coefficients of the digital filter modeling the vocal tract the gain σ of this filter can also be determined:

$$\sigma = \sqrt{\frac{r_l(0)}{p}} \tag{B.16}$$

LPC Parameter Conversion

One of the major weaknesses of the plain LPC-parameters is their inability to reflect the distribution of information within the spectral envelope. In order to create a more robust set of parameters that compensates for this loss the LPC-coefficients will have to be converted into LPC cepstral coefficients. The cepstral coefficients rely on representing the LPC parameter as Fourier transform of the log magnitude spectrum and has been proven to be more robust and reliable than the plain LPC coefficients when used in a recognition scheme. The LPC cepstral coefficients c can be derived directly from the LPC coefficients a using the following recursion:

$$\begin{aligned}
c_0 &= ln\sigma^2 && \text{(B.17)}\\
c_m &= a_m + \sum_{k=1}^{m-1} \Big(\frac{k}{m}\Big) c_k a_{m-k} \qquad 1 \geq m \geq p && \text{(B.18)}\\
c_m &= \sum_{k=1}^{m-1} \Big(\frac{k}{m}\Big) c_k a_{m-k} \qquad m > p && \text{(B.19)}
\end{aligned}$$

[1] Symmetric matrix with all diagonal element equal

The number of LPC cepstral coefficients Q are generally higher than the order of the LPC-analysis p and values where $Q = \frac{3}{2}p$ has previously been used successfully [**?**, p. 116].

Parameter Weighting

LPC cepstral coefficients has the advantage of an increased sensitivity to the lower part of the spectral envelope and opposed at the higher part. This change in sensitivity spawns a range of problems. The following two factors are the most severe; the sensitivity of the low-order coefficients to the overall spectral slope and the high-order coefficients sensitivity to noise. In order to compensate for these problems it is common practice to weight the cepstral coefficients thereby minimizing the influence of the sensitivities. These weighted cepstral coefficients $\hat{c}_m$ are achieved is by using a tapered window w_m:

$$\hat{c}_m = w_m c_m, \qquad 1 \geq m \geq Q \tag{B.20}$$

The purpose of the tapered window w_m is to de-emphasis the upper and lower parts of c_m. The following window has proven suitable:

$$w_m = 1 + \frac{Q}{2} \sin\left(\frac{\pi m}{Q}\right), \qquad 1 \geq m \geq Q. \tag{B.21}$$

APPENDIX C

Pattern Comparison

This appendix will describe the basics of template-based pattern recognition. It will contain considerations of distance measurement of speech patterns. This is followed by a description of a time warping recognizer. Finally connected word recognition is explained.

Finding a reasonable measure will enable an evaluation of the distance between test patterns and reference patterns and thereby making it possible for the system to make a decision based on this distance.

In the pattern-based approach to speech recognition, speech is represented by a time sequence of spectral vectors given by the feature extraction. A test pattern T can be defined as

$$T = \{t_1,\ t_2,\ \ldots\ ,\ t_I\} \tag{C.1}$$

Each t_i is the spectral vector from the input speech at time i and I is the number of frames in the input speech. Similar, a reference pattern is defined as

$$R_k = \{r_1,\ r_2,\ \ldots\ ,\ r_{J(k)}\} \tag{C.2}$$

where each r_j is a spectral vector from the k'th reference pattern at time j and $J(k)$ is the number of frames in the k'th reference pattern. The reference patterns are also referred to as templates.

The task is to identify the reference pattern with the minimum distance to the test pattern, and then associate this with the spoken input. In order to complete this task the following problems must be addressed:

1. The time duration of T and R_k may differ because the speaking rates may be different.
2. T and R_k may not line up in time in a simple way, because different sounds can not be varied in duration to the same degree. Vowels are easily lengthened or shortened, while most consonants can not change dramatically in duration.
3. Comparing pairs of spectral vectors (local distance) is required in order to evaluate the distortion (global distance) and to facilitate a line up between T and R_k .

Thus a local measure and a method for global time aligning T and R_k must be found, so that the accumulated local distance between spectral frames in the time-aligned patterns are minimized.

According to [**?**, p. 142] global time alignment can be addressed analytically and merged with the problem of determining a local distance measure, yielding optimal solutions for a wide range of spectral distance measures.

C.1 Distance Measures

The spectral distance function d between two feature vectors has the form:

$$d(v_i,\ v_j) = d_{ij} \begin{cases} = 0 & \text{if } v_i = v_j \\ > 0 & \text{otherwise} \end{cases} \tag{C.3}$$

where v_i and v_j denote the spectral (e.g. feature) vectors. A number of different measures of the distance between two feature vectors exist. However, usually the method used in the preprocessing, influences this choice. In speech recognition there is particularly two important issues to consider when defining a measurement:

1. High degree of mathematical tractability
2. Subjective meaningfulness

Given two feature vectors v_i and v_j defined on a vector space V, and distance function d defined as a real-valued function on V, the high degree of mathematical tractability is present, when the distance function d satisfy the following conditions:

1. $0 \leq d(v_i,\ v_j) < \infty$ for $v_i,\ v_j \in V$ and $d(v_i,\ v_j) = 0$ if and only if $v_i = v_j$
2. $d(v_i,\ v_j) = d(v_j,\ v_i)$ for $v_i,\ v_j \in V$
3. $d(v_i,\ v_j) \leq d(v_i,\ v_k) + d(v_j,\ v_k)$ for $v_i,\ v_j,\ v_k \in V$
4. $d(v_i + v_k,\ v_j + v_k) = d(v_i,\ v_j)$

With subjective meaningfulness the intention is to derive a measure, which perceptually makes sense. We're interested in defining a measure that yields a large distance, when the two sounds being compared is very different perceptually and a small distance for two sounds which are very alike. Some of the spectral changes that fundamentally don't change the perceived sound are listed below:

1. Spectral tilt: $S_1(\omega) = S_2(\omega) \cdot \omega^{\alpha},\ \alpha$: the tilt factor
2. Highpass filtering: $S_1(\omega) = S_2(\omega) \cdot (\|H_{HP}(e^{j\omega})\|)^2$
3. Lowpass filtering: $S_1(\omega) = S_2(\omega) \cdot (\|H_{LP}(e^{j\omega})\|)^2$
4. Notch filtering: $S_1(\omega) = S_2(\omega) \cdot XS \cdot (\|H_N(e^{j\omega})\|)^2$

A relatively large distance $d(S_1,\ S_2)$ is ideally expected for the following spectral changes:

1. Large differences in formant locations, meaning that the spectral resonances of $S_1(\omega)$ and $S_2(\omega)$ occur at different frequencies.
2. Large differences in formant bandwidths, resulting in different frequency widths of the spectral resonances of $S_1(\omega)$ and $S_2(\omega)$,

The distance would be relatively large in order for the subjective meaningfulness to be present. Unfortunately, both can not be accomplished at the same time and some compromise is needed. A number of such measures with the mentioned compromise are listed in [**?**, chapter 4]. It is considered beyond the scope of this report to give a detailed description of each measure and their individual differences and the reader should refer to the appropriate literature for further details.

Based on considerations concerning both the computational complexity and the requirements for the system to be developed, an Euclidean Distance measure seems a good candidate. The distance measure is defined as follows:

$$Euclidean\ Distance: \quad d_E = \sqrt{\sum_{n=0}^{N}(s_n - s'_n)^2} \tag{C.4}$$

The Euclidean Distance measure has a rather attractive complexity and holds the mathematical tractability. On the other hand it is not as desirable - perceptually speaking. The perceptive lag is considered acceptable because the exact spoken words is not to be recognized - only the information necessary to decide the correct visemes is needed.

Having found a suitable measure for the distance between speech signals, it must be determined which method to implement in the recognizer.

C.2 Dynamic Time Warping

The basic idea in DTW is that when comparing two speech patterns, the utterance may vary in time. The two speech patterns must be time-aligned by "warping" by stretching and shrinking them, see figure C.1. The optimal warping is the one giving the minimal dissimilarity between the two patterns.

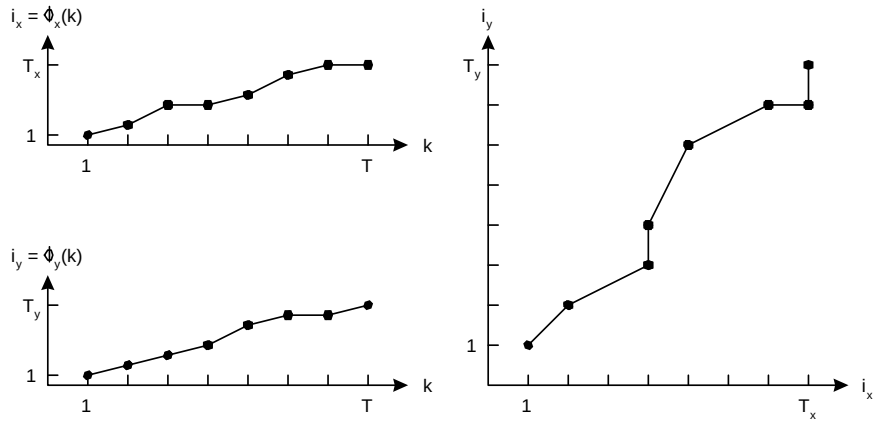

Figure C.1: Time normalization (or "warping") of two patterns into a common time axis [after **?**, fig. 4.37].

The warping functions used to map the patterns to a normalized time axis k, are denoted:

$$i_x = \phi_x(k), \qquad k = 1,\ 2,\ \ldots,\ T \tag{C.5}$$

$$i_y = \phi_y(k), \qquad k = 1,\ 2,\ \ldots,\ T \tag{C.6}$$

The DTW contains a method to find the optimal path through the distance matrix, and defines a dissimilarity measure $d_\phi(X, Y)$ for the path given by $\phi_x(k)$ and $\phi_y(k)$:

$$d_\phi(X, Y) = \sum_{k=1}^{T} d(\phi_x(k), \phi_y(k))m(k)/M_\phi \tag{C.7}$$

where $m(k)$ is a path weighting coefficient and M_ϕ is a path normalizing factor (explained later).

C.2.1 Input

The DTW compares two speech patterns X and Y, consisting of the feature vectors x_1, x_2, ... , x_{Tx} and y_1 , y_2, ... , y_{Ty}. The number of feature vectors, T_x and T_y, need not be identical. The feature vectors can be any vectors derived from the speech signal (e.g. through LPC analysis), if a (local) distance measure $d(i_x, i_y)$ is defined for comparing them, see C.1.

C.2.2 Output

The DTW returns a (global) dissimilarity value $d(X, Y) = \min_\phi d_\phi(X, Y)$ representing the optimal path. By comparing a test pattern to a number of reference patterns this value can be used to find the best matching reference pattern.

C.2.3 Distance Matrix

When calculating the (global) dissimilarity between two patterns X and Y, the DTW works on a distance matrix calculated by using the (local) distance measure $d(i_x, i_y)$ of the feature vectors. Figure C.2 is an example of a distance matrix calculated from two patterns X and Y, consisting of 5 and 4 one-dimensional feature vectors respectively.

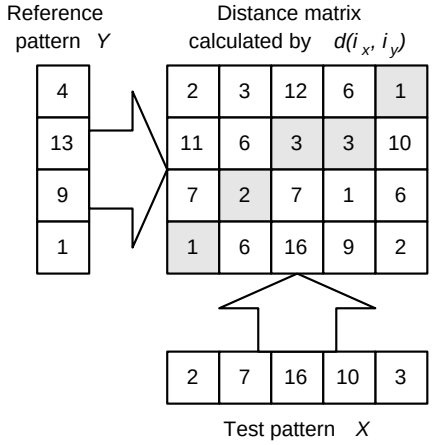

Figure C.2: Example of a distance matrix. The grey fields in the matrix represent the optimal warping path through the matrix, when certain warping constraints are used.

Finding the optimal warping is thus the same as finding the optimal path through the distance matrix.

C.2.4 Warping Constraints

As some paths through the distance matrix are meaningless (e.g. going round in circles), constraints for the warping can be defined.

This will limit computations as well. [**?**, p. 208] define the following constraints:

- endpoint constraints
- monotonicity conditions
- local continuity constraints
- global path constraints
- slope weighting

Endpoint Constraints

When the beginning and the end of the patterns to be compared are known, the endpoints are beginning at $\phi_x(1) = 1, \phi_y(1) = 1$ and ending at $\phi_x(T) = T_x, \phi_y(T) = T_y$.

When the endpoints are not known with such accuracy, relaxed endpoint constraints can be used, where a number of beginning and ending points are used.

In this system the patterns to be recognized (the visemes) are connected and only endpoints of whole sentences (or a number of words) are known. A method for solving this problem (known as the connected word recognition) is described later in C.3.

Monotonicity Conditions

The monotonicity condition means that it is not allowed to go back in time. This corresponds to not allowing going left or down in the distance matrix.

Local Continuity Constraints

These constraints define the legal increments in a path through the distance matrix. The increments are constrained in a way so that it does not skip any (or almost any) information (feature vectors) in the patterns. Figure C.3 illustrates eight sets of local constraints from [**?**, p. 211]. The P's in the figure are the allowed (local) paths. Note that a (local) path can consist of several jumps.

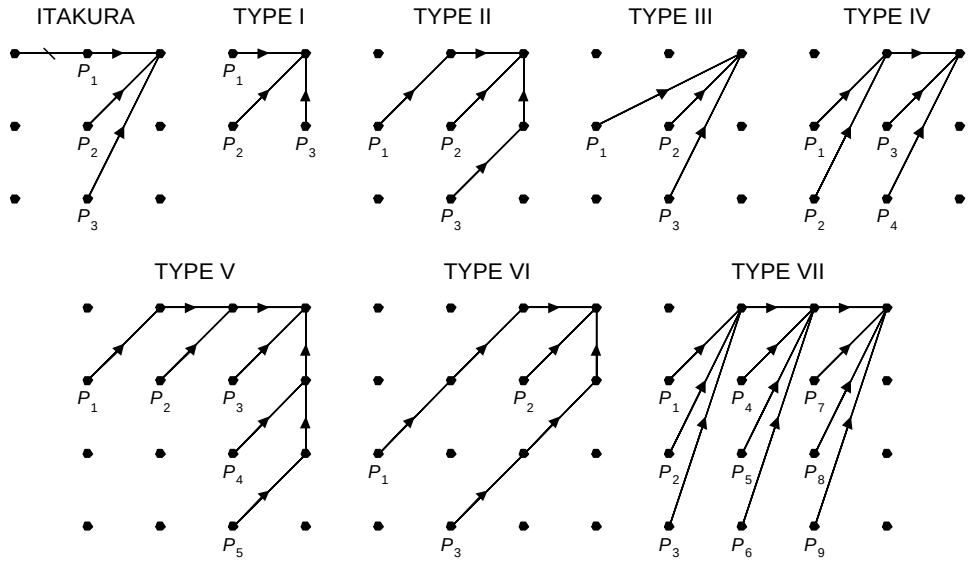

Figure C.3: Local continuity constraints. Read as "where came I from"not "where can I go"[after **?**, p. 211].

The Itakura local constraints include a special constraint that disallows two consecutive moves to the right.

The choice of a specific local constraint cannot be made analytically. The constraints perform different depending on the application. Therefore the choice must be based on experimental results.

Global Path Constraints

Because the local path has maximum and minimum slopes, the local continuity constraints limit which coordinates in the distance matrix that can be accessed when calculating the DTW. This can be used to save computations.

Table C.4 lists maximum and minimum slopes (Q_{max} and Q_{min}) for the eight different local path constraints.

Type	Q_{max}	Q_{min}
I	inf	0
II	2	1/2
III	2	1/2
IV	2	1/2
V	3	1/3
VI	3/2	2/3
VII	3	1/3
Itakura	2	1/2

Figure C.4: Maximum and minimum slopes for different types of local path constraints [after ?, p. 214].

Figure C.5 illustrates the global constraints. Additional lines with slope = 1 can be placed with some distance from the diagonal as range-limiting constraints to further save computations. However, these range-limiting constraints are only interesting when working with big distance matrices.

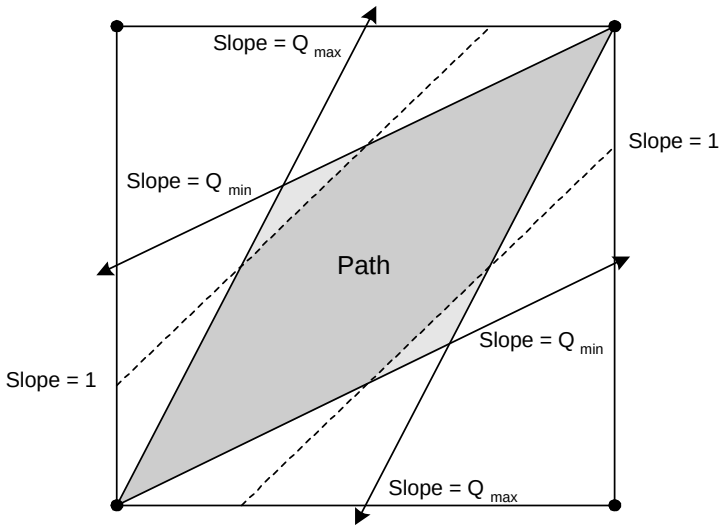

Figure C.5: Global path constraints [after ?, p. 215].

Slope Weighting

Slope weighting adds a weight to each allowed path in the local continuity constraints. This gives the option to prefer some paths compared to other paths. For example, in type 1 local constraints, the diagonal path could be given a weight of 2 while the remaining paths are given

the weight of 1. This would have the effect that one diagonal jump is equally expensive as a horizontal plus a vertical jump.

Four types of weighting given in [?, p. 216] are:

- Type (a): m(k) = min[jump width, jump height]
- Type (b): m(k) = max[jump width, jump height]
- Type (c): m(k) = jump width
- Type (d): m(k) = jump width + jump height

Jump width is calculated by $\phi_x(k) - \phi_x(k-1)$, and jump height by $\phi_y(k) - \phi_y(k-1)$, where $\phi_x(0) = \phi_y(0) = 0$ is assumed.

If a path in a local constraint consists of multiple jumps, the weights from each jump can be redistributed along the path by using their mean value. By doing this, 0 weights can be avoided (in most cases). This is illustrated in figure C.6.

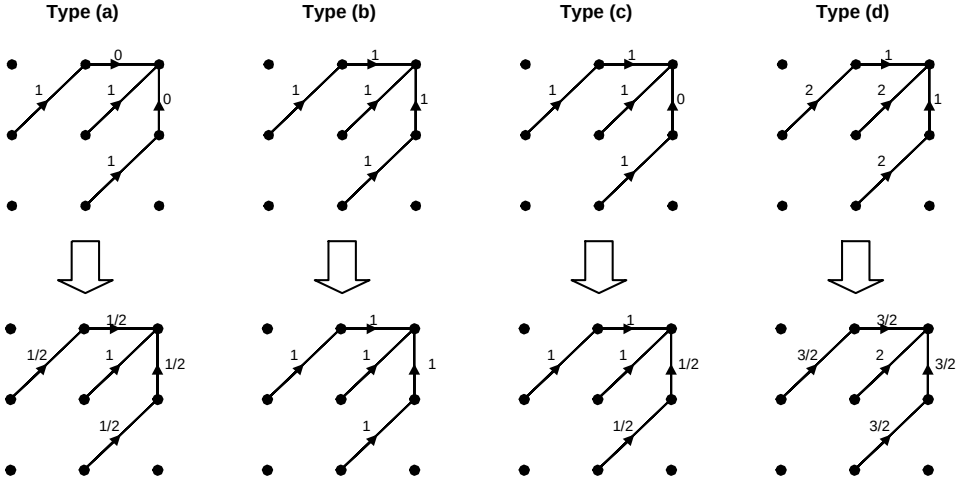

Figure C.6: Slope weighting for the four types of weighting. Weights are shown before and after redistribution along paths [after ?, p. 218].

C.2.5 Path Normalizing Factor

The purpose of the path normalizing factor M_ϕ is to make the calculated dissimilarity value $d(X, Y)$ independent of the lengths of the reference and test patterns. Since the dissimilarity value in equation. C.7 is a weighted sum, M_ϕ is chosen to be the sum of the weights:

$$M_\phi = \sum_{k=1}^{T} m(k) \tag{C.8}$$

This gives a normalizing factor that is independent of the warping path for type (c) and (d) slope weighting. For type (c) M_ϕ is the sum of the jump widths (Tx). For type (d) M_ϕ is the sum of the jump widths and heights ($Tx + Ty$).

For type (a) and (b) slope weighting, the normalizing factor depends on the chosen warping path. One solution to this is to choose a value of Tx that is only reasonable in some cases. Another solution could be to use path back-tracking to calculate M_ϕ based on the actual path.

When no slope weighting is used (that is, $m(k) = 1$), ideally M_ϕ is set to the number of steps T [?, p. 292], but this also depends on the chosen warping path, so the problem is the same as for type (a) and (b).

C.2.6 DTW Algorithm

The DTW algorithm is based on dynamic programming, which is a method for solving sequential decision problems. The general dynamic programming algorithm will not be shown here, but it is important to mention the principle of optimality:

> *An optimal policy has the property that, whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*
> [**?**, p.205]

For the DTW algorithm a new dissimilarity function, $D(T_x, T_y)$ is defined, where the path normalizing factor is removed:

$$M_\phi d(X,Y) = D(T_x, T_y) = \min_{\phi_x,\phi_y} \sum_{k=1}^{T} d(\phi_x(k), \phi_y(k)) m(k) \tag{C.9}$$

A partial dissimilarity function is defined for a path connecting $(1,1)$ and (i_x, i_y):

$$D(i_x, i_y) = \min_{\phi_x,\phi_y,T'} \sum_{k=1}^{T'} d(\phi_x(k), \phi_y(k)) m(k) \tag{C.10}$$

where $\phi_x(T') = i_x$ and $\phi_y(T') = i_y$.
A weighted accumulated distance ψ between point (i'_x, i'_y) and (i_x, i_y) is defined:

$$\psi((i'_x, i'_y), (i_x, i_y)) = \sum_{l=0}^{L_s} d(\phi_x(T'-l), \phi_y(T'-l)) m(T'-l) \tag{C.11}$$

where L_s is the number of jumps in the local path, $\phi_x(T' - L_s) = i'_x$ and $\phi_y(T' - L_s) = i'_y$.
The steps in the algorithm are as follows:

1. Initialization:
$$D(1,1) = d(1,1)m(1) \tag{C.12}$$

2. Recursion:
For $1 \leq i_x \leq T_x$, $1 \leq i_y \leq T_y$, and only within the allowable grid defined by the global constraints, calculate
$$D(i_x, i_y) = \min_{(i'_x, i'_y)} [D(i'_x, i'_y) + \psi((i'_x, i'_y), (i_x, i_y))] \tag{C.13}$$

3. Termination:
$$d(X,Y) = D(T_x, T_y)/M_\phi \tag{C.14}$$

C.3 Connected Word Recognition

When recognizing word (or viseme) units in speech, the units to be recognized are usually connected, i.e. there is no silence between them and thus there is no (simple) way to detect their boundaries. The standard DTW algorithm is only meant for isolated word recognition and can not be used alone for connected words. Several methods for solving the task of connected word recognition have been proposed. In the following the problem in general will be described and a specific solution known as the one-pass (or one-state or one-stage) algorithm will be presented.

C.3.1 The Super-Reference Pattern

The task of connected word recognition can be described as the task of finding the super-reference pattern,

$$R^S = \{R_{q(1)} \oplus R_{q(2)} \oplus ... \oplus R_{q(L)}\} \tag{C.15}$$

that is, finding the best matching sequence of reference patterns (represented by $q(1)...q(L)$).

This task can be solved in a brute-force way by evaluating the super-reference pattern using the standard DTW algorithm described in the previous section, and comparing the dissimilarity values for every possible combinations of reference patterns and every number of reference patterns L between L_{min} and L_{max} (where L_{min} and L_{max} is the minimum and maximum expected number of words to be recognized).

If K is the number of reference patterns, then the number of possible combinations of local reference patterns is K^L, which is a highly undesired complexity (except for very small values of K and L). This clearly shows the need for an alternative way of solving the connected word recognition problem.

C.3.2 The One-Pass Algorithm

The following is mainly based on [**?**]. The one-pass algorithm is easiest described as an extension of the DTW-algorithm. Instead of time-warping only one reference pattern, the warping is now done to all reference patterns simultaneously, as shown in figure C.7. As in the DTW, the path is not known during the warping. First when the end of the test pattern is reached, the path can be found by using path backtracking. Note that the goal of the algorithm, however, is different than the goal of the DTW. The DTW gave a dissimilarity score. In the one-pass algorithm the score itself is not interesting. It is the warping path, which is used to find the transitions between the reference patterns (which are words or visemes).

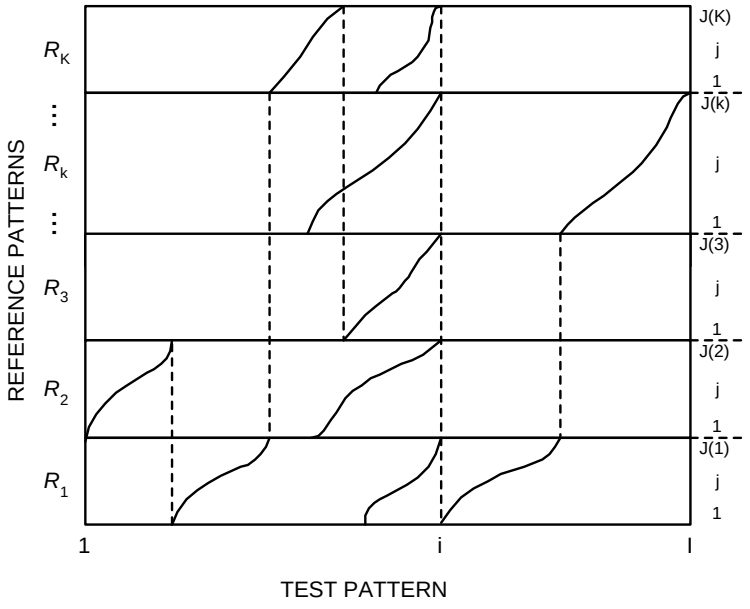

Figure C.7: The one-pass algorithm. The time-warping is done in all reference patterns simultaneously [after **?**, p. 43].

The local dissimilarity in the one-pass algorithm is denoted as:

$$d(i, j, k) \tag{C.16}$$

where the reference patterns are indexed by $k = 1, ..., K$, the time frames of the reference pattern k are denoted as $j = 1, ..., J(k)$, and the time frames of the test patterns are referenced by the index $i = 1, ..., I$ (The local dissimilarity is calculated with the chosen dissimilarity measure as in the DTW).

Similarly, the minimum accumulated distance to the point (i, j, k) is denoted as:

$$D(i, j, k) \tag{C.17}$$

When warping inside a reference pattern k, the operations of the one-pass algorithm are similar to the DTW. The minimum accumulated distance is calculated by choosing a predecessor with the minimum accumulated distance:

$$D(i, j, k) = d(i, j, k) + min[D(i-1, j, k), D(i-1, j-1, k), D(i, j-1, k)] \tag{C.18}$$

The path constraints of the above equation corresponds to the TYPE I local path constraints of the DTW.

At the boundary of the reference pattern (i.e. at $j = 1$), a special path constraint is used, as illustrated in figure C.8.

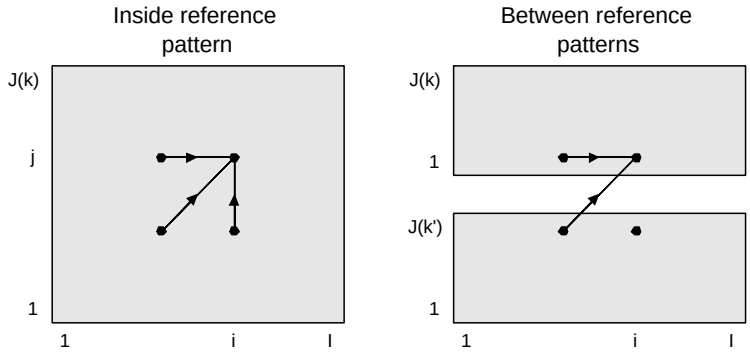

Figure C.8: Different path constraints are used when inside and when between the reference patterns [after **?**, p. 3].

Either a predecessor from the same reference pattern which is one test time frame to the left will be chosen, or a predecessor from the ending reference time frame one frame to the left of the reference pattern with the lowest accumulated dissimilarity will be chosen:

$$D(i, 1, k) = d(i, 1, k) + \min[D(i-1, 1, k), \min_{k'}[D(i-1, J(k'), k')]] \tag{C.19}$$

The steps of the one-pass algorithm with the path constraints described above are as follows:

1. Initialization:
$$D(1, j, k) = \sum_{n=1}^{j} d(1, n, k) \tag{C.20}$$

2. Warping:

For $2 \leq i \leq I$, calculate
For $1 \leq k \leq K$, calculate

$$\begin{aligned} D(i, 1, k) &= d(i, 1, k) + \min[D(i-1, 1, k), \\ &\quad \min_{k'}[D(i-1, J(k'), k')]] \end{aligned} \tag{C.21}$$

For $2 \leq j \leq J(k)$, calculate

$$\begin{aligned} D(i, j, k) &= d(i, j, k) + \min[D(i-1, j, k), \\ &\quad D(i-1, j-1, k), D(i, j-1, k)] \end{aligned} \tag{C.22}$$

End j
End k
End i

3. Path back-tracking:
Find ending reference pattern with minimum total distance
$$R_{end} = \min_{k'}[D(I, J(k'), k')] \tag{C.23}$$
Use back-pointers to do the back-tracking. At reference pattern transitions save the interesting information: the test frame (time) index i and the reference pattern number k.

The one-pass algorithm can be implemented with different path constraints than the ones shown above (e.g. the constraints shown in [**?**, p. 419]). Some memory requirements optimizations are possible, syntactic constraints can also be incorporated, see [**?**]. These topics will not be covered here.

APPENDIX D

The Strategy Design Pattern

This appendix will describe the strategy design pattern. For more in depth information see [**?**, p.315-324].

D.1 Motivation

The motivation for the use of this design pattern is the wish to have multiple algorithms in a program depending on the situation at hand. This is best illustrated through a small example.

When solving a problem there is normally several algorithms that can do the job. Each of the algorithms normally has their strengths and weaknesses. For instance there could be from 1 to n algorithms to solve the problem at hand. The goal of this design pattern is then to make all of the n algorithms available to the program through a simple interface. Hence not all of the algorithms needs to hardwire in to the program and it only needs one of the algorithms, but the program can then specify which one it needs. This is called a strategy.

The strategy design pattern should be used when:

- Many related classes differ only in their behavior. Strategies provide a way to configure a class with one of many behaviors.
- There is need for different variants of an algorithm. Strategies can be used when these variants are implemented as a class hierarchy of algorithms.
- An algorithm uses data that clients shouldn't know about. Use strategy pattern to avoid exposing complex, algorithm-specific data structures.
- A class defines many behaviors and these appear as multiple conditional statements in its operations. Instead of many conditionals move related conditional branches into their own strategy class.

The structure of the Strategy design pattern is shown in figure D.1

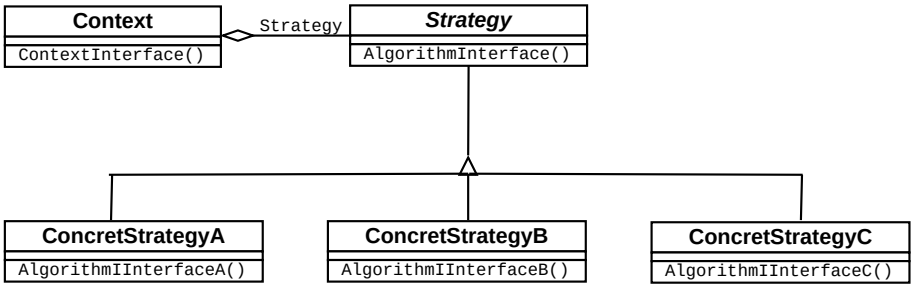

Figure D.1: Structure of the design.

There is the following participants in this design pattern:

Strategy: declares an interface common to all supported algorithms. Context uses this interface to call the algorithm defined by ConcreteStrategy.

ConcreteStrategy: implements the algorithm using the Strategy interface.

Context: is configured with a ConcreteStrategy object and it maintains a reference to a Strategy object. Furthermore it may define an interface that lets Strategy access its data.

There are the following interactions in this design pattern:

- Strategy and Context interact to implement the chosen algorithm. A context may pass all data required by the algorithm to the strategy when the algorithm is called. Alternatively the context can pass itself as an argument to Strategy operations. That lets the strategy call back on the context as required.
- A context forwards requests from its clients to its strategy. Clients usually create and pass a ConcreteStrategy object to the context. Thereafter clients interact with the context exclusively. There is often a family of ConcreteStrategy classes for a client to choose from.

APPENDIX E

Training Methods

In this appendix the various methods for training of pattern-recognition based recognizers will be explained.

E.1 Casual Training

In casual training each utterance class is represented by several reference patterns. Normally all spoken tokens during the training session are used as reference patterns. This leads to the following characteristics:

1. Simple template training procedure
2. The number of utterance classes in the vocabulary should not be too many
3. Works only with a speaker-trained system (speaker dependent)
4. No attempt to estimate the pattern variability
5. Errors committed in training (improper articulation, mispronounciation etc.) are accepted as valid reference patterns (no use of threshold)

E.2 Robust Training

In robust training the reference patterns are created from a concept called consistent pairs. The term "consistent pair" will be described in the following. The training is performed as described in these steps:

1. Let $X_1 = (x_{11},\ x_{12},\ \ldots\ ,\ x_{1T_1})$ and $X_2 = (x_{21},\ x_{22},\ \ldots\ ,\ x_{2T_2})$ be two patterns that are being compared via DTW.
2. The DTW distortion score $d(X_1,\ X_2)$ is compared to a threshold ϵ . If the score is smaller than ϵ, X_1 and X_2 is accepted as a consistent pair. Else a new pattern X_3 must be introduced (possibly also X_4 and X_5 and so on).
3. The reference pattern $Y = (y_1,\ y_2,\ \ldots\ ,\ y_{T_y})$ is computed as the warped average of X_1 and X_2, where $y_k = \frac{1}{2}(x_{1\phi_1(k)} + x_{2\phi_2(k)}),\ k = 1,\ 2,\ \ldots\ ,\ T_y$

The characteristics of the robust training procedure can be seen below:

1. The training could be cumbersome depending on the value of the threshold ϵ
2. Works only with a speaker-trained system (speaker dependent)
3. Given a set of pre-recorded and digitized training data it is critical that consistent pairs can be found from these data
4. Simple and efficient in term of computation and storage
5. Inadequate for words having more than one mode (i.e.. words with released stops)

E.3 Clustering

In this procedure the starting point is a set of L speech patterns, each a realization of the same utterance class. These L patterns is then clustered into N clusters, satisfying that within each cluster the patterns are highly similar with respect to the specific chosen dissimilarity measure. The result is N representative templates from a set of L training patterns for each utterance class. Two different clustering algorithms are commonly used, but before these will be examined some basic mathematic definitions should be introduced.

Let $\Omega = \{X_1,\ X_2,\ ...\ ,\ X_L\}$ denote a set of L training patterns, each X_i being a realization of a specific utterance class. An $L\ x\ L$ distance matrix D can now be defined with ij^{th} entry d_{ij} given by:

$$d_{ij} = \frac{1}{2}[d(X_i,\ X_j) + d(X_j,\ X_i)] = \delta(X_i,\ X_j) \tag{E.1}$$

In order to cluster the training set Ω into N disjoint clusters $\{\omega_i,\ i = 1,\ 2,\ ...\ ,\ N\}$ so that

$$\Omega = \bigcup_{i=1}^{N} \omega_i \tag{E.2}$$

and such that the speech patterns in corresponding clusters are very similar, the solutions for $\{\omega_i\}_{i=1}^{N}$ must be computed. Each cluster ω_i is represented by a representative pattern $Y(\omega_i) \triangleq Y_i$. Note however that Y_i is not necessarily a member of ω_i.

E.3.1 Unsupervised Clustering Without Averaging (UWA)

The concept of unsupervised clustering is given in the five steps below:

1. Identify the largest cluster
2. Determine all training patterns close to the "center" of this cluster
3. Exclude these patterns from the training set
4. Recluster the remaining patterns
5. Iterate

Let Ω_j represent the partial set that includes the training patterns up to the j^{th} cluster, then

$$\Omega_j = \bigcup_{i=1}^{j} \omega_i = \Omega_{j-1} + \omega_j \tag{E.3}$$

and hence the complement set

$$\overline{\Omega}_j = \Omega - \Omega_j \tag{E.4}$$

is then the set consisting of all remaining patterns after the j^{th} cluster. As it was stated in step 5 the UWA is iterative and in the following k is the iteration index. Let ω_j^k denote the set of patterns in the j^{th} cluster at the k^{th} iteration. Then for every cluster ω, a min-max "center" $Y(\omega) = X_{i_c} \in \omega$ is defined such that

$$\max_m d_{i_c,\, m} = \min_i \max_m d_{i,\, m} \tag{E.5}$$

for all $X_i \in \omega$. This means that the min-max "center" of a set is the pattern with the smallest maximum distance to all the remaining patterns. The UWA algorithm steps can be found in [**?**, p. 269-270]. It should be noted that the exclusion in step 3 requires a distance threshold which can not be optimized in any analytical manner. Furthermore the algorithm does not guarantee 100% coverage of the training.

E.3.2 Modified K-Means Algorithms (MKM)

In this algorithm the clusters and cluster centers (centriods) are iteratively refined until an optimality requirement is satisfied. In the following the algorithm is explained.

Let $\omega_{j,i}^k$ denote the i^{th} cluster of a j-cluster set at the k^{th} iteration. The range of the cluster summation index variable is $i = 1, 2, ..., j$ and the iteration index variable range is $k = 1, 2, ..., k_{max}$, where k_{max} is the maximum allowed iteration count. $Y(\omega)$ is defined as the centriod (the min-max center) of ω and denotes the representative pattern for the cluster ω. The algorithm determines j clusters incrementally from $j = 1$ to $j = j_{max}, j_{max}$ being the desired maximum number of clusters to be determined. Assuming the distance matrix D is to be precomputed, the algorithm is as follows:

1. Initialize: $j, i, k = 1$, $\omega_{1,1}^1 = \Omega$ and compute centriod $Y(\omega)$ of Ω
2. Optimal minimum distance classification: Each pattern $X_l, l = 1, 2, ..., L$ in Ω is labeled by index i according to the minimum distance principle:

   $$X_l \in \omega_{j,i}^k \qquad if \qquad \delta(X_l, Y(\omega_{j,i}^k)) = min_{l'} \delta(X_l, Y(\omega_{j,i'}^k)) \tag{E.6}$$

   Sum the total intracluster distance for each cluster $\omega_{j,i'}^k$, defined as:

   $$\Delta_i^j = \sum \delta(X_l, Y\omega_{j,i}^k) \tag{E.7}$$

   The summation is overall $X_l \in \omega_{j,i}^k$.
3. Revision of clusters and centroids: Form $\omega_{j,i}^{k+1}$ by grouping all X_l's with label i retrieved from step 2. Compute new centroids for $\omega_{j,i}^{k+1}, for i = 1, 2, ..., j$
4. Convergence test: Goto 5 if one of these is fulfilled:
   - $\omega_{j,i}^{k+1} = \omega_{j,i}^k$ for all $i = 1, 2, ..., j$
   - $k = k_{max}$: maximum iteration count reached
   - Change in average (or total accumulated) distance is below predefined threshold Δ_{th}

   else $k++$ and repeat step 2 - 4.
5. Record the j-cluster solution: Given convergence has been reached, the resultant clusters and centroids, $\omega_{j,i}^k$ and $Y(\omega_{j,i}^k), i = 1, 2, ..., j$ are the j-cluster solution for the training set Ω.

A flow diagram for the algorithm can be found in [**?**, p. 272]

E.3.3 Characteristics of the Clustering Method

Based on the previous the clustering method possesses the following characteristics:

1. Speaker-independent
2. Dependent in a higher degree on choice of spectral distance measure than other training procedures
3. The templates are statistical consistent
4. Robust to a wide range of individual speech variations in a speaker-independent environment
5. Higher recognition accuracy is achieved in practical tasks

APPENDIX F

Viseme Format

This chapter will discuss phonemes and visemes for practical use in an application. The result will be a set of visemes and a system for mapping a stream of phonemes into a stream of visemes. The reader is advised to consider the phoneme examples by uttering the phonemes while exaggerating the articulation.

F.1 Phonemes - From a Visual Point of View

SAMPA [**?**] is a well known and documented standard of phonemes. It is used in the EUROM1 database, which is used as training data for ALSAC. As discussed in chapter 2 each viseme can be mapped into a viseme-map based on their roundness and openness. Unfortunately, not all phonemes can be mapped into the same map because of the context and speaker dependence. Simple guidelines for mapping will be considered below.

The vowels are consistent in openness and roundness thereby giving a solid ground for construction of a visemes map. Phonology describes a vowel triangle in which the vowels are mapped according to roundness and openness. The vowels defined in SAMPA is shown in table F.1

	Rounded	Unrounded
Closed		
	y u	i
	2 o @	e
	9 O	E
	Q	{
		a
		A
Open		

Table F.1: Vowel in SAMPA ordered by openness and roundness.

However, the mapping of consonants are not that simple. The vowel affects the surrounding consonants within a given syllable. This is because the generation of many consonants are performed far behind the lips and further down in the pharynx tract. For example /g/ and /k/ are produced in the throat, while /l/ and /n/ is produced using the tongue. Thus, the roundness of the consonants does not affects the roundness of the syllable and are used only to smoothen the movements of the lips between two vowels. Regarding openness the consonants labials, such

as /b/ and /m/, have closed lips. The rest are mid-open and details such as teeth and tongue distinguish the different consonants. The consonants defined by SAMPA is shown in table F.2

Phoneme	Visualization
p m b	Bi-labials: lips shut
f v	Labio-dentals: bite lower lip
D	tongue behind teeth
j	teeth show, tongue hidden
s	teeth show, tongue hidden
t n d	teeth slightly open, tongue up
k N g R	upper teeth show, tongue drawn back
l	teeth open, tongue up

Table F.2: Identification of visemes for consonants in SAMPA.

F.2 Visemes in Animations

The MPEG4 standard includes a standard for face animation [**?**]. This includes animation of the mouth. In the MPEG4 standard each phoneme is mapped to a viseme. The phonemes used in MPEG4 are the most common English phonemes, which are shown in table F.3.

Viseme	Phonemes	Example
0	none	
1	p, b, m	put, bed, mill
2	f, v	far, voice
3	T,D	think, that
4	t, d	tip, doll
5	k, g	call, gas
6	tS, dZ, S	chair, join, she
7	s, z	sir, zeal
8	l, n	lot, not
9	r	red
10	A:	car
11	e	bed
12	I	tip
13	Q	top
14	U	book

Table F.3: MPEG4's phoneme to viseme mapping.

This ambiguous mapping is not the ideal way to do it, but it is an often-used method, because it is a simple way to generate a set of visemes. Each of the visemes in MPEG4 is a vector-based guideline for the final image. In addition expressions can be added to the viseme. The transitions from one viseme to the next are defined by blending the two visemes with a weighting factor. Context dependence makes this system inadequate for lip-reading and perfect natural animation.

The viseme map used by ALSAC is composed of fixed images. ITE provided a set of 16 basic images, which is found to be inadequate for the wide range of phonemes. The viseme set has been modified to fit the phonemes of MPEG4 in figure F.2.

A vowel affects the surrounding consonants within a syllable. If the viseme set only consists of one viseme for each consonant the consonants ought to be placed in the center of the viseme map. Anyhow, this would lead to a terrible animation if two subsequent syllables contain rounded

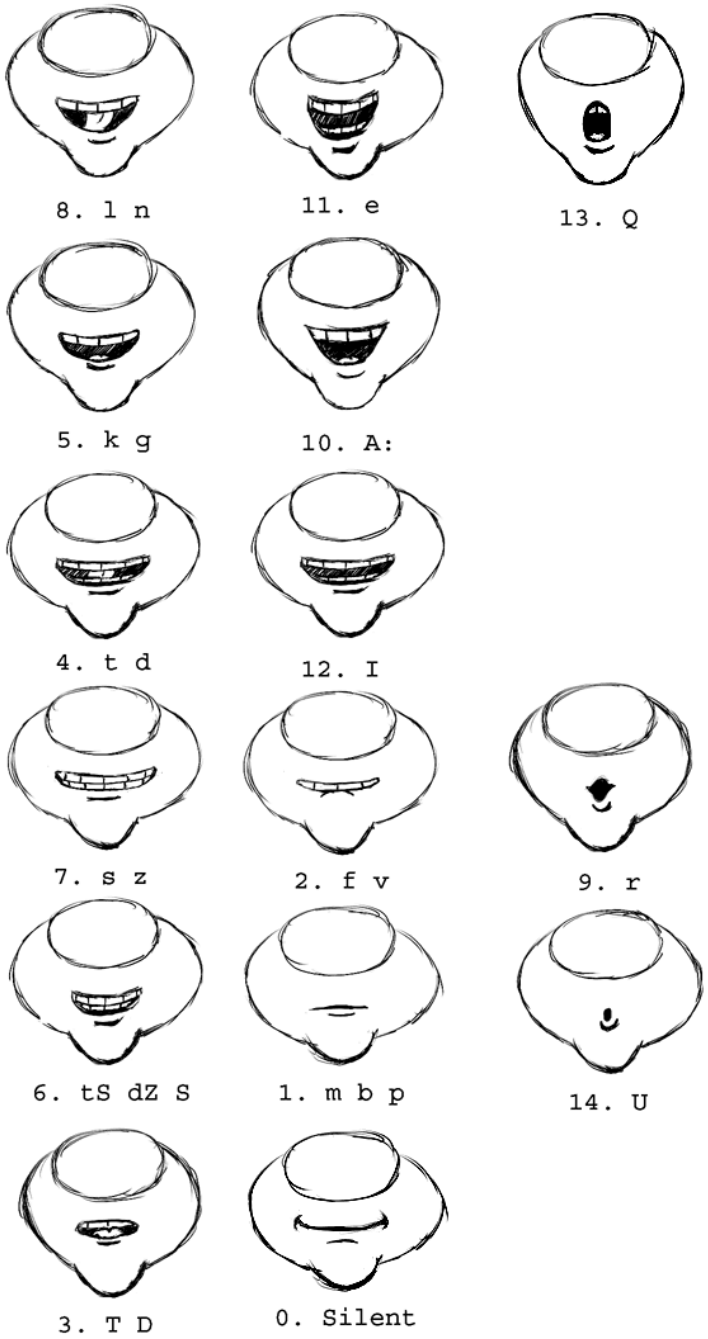

Figure F.1: Viseme set adapted for MPEG4.

vowels, while the consonants between those vowels are unrounded. In order to smoothen the animation one can use two visemes for each consonant - a rounded and an unrounded. This is the basis for the extended viseme set used by ALSAC. The set is shown in table F.4 where rounded consonants are represented by the prefix a. The new viseme map using the double consonant system is shown in figure F.2. Note that the phoneme /h/ does not have a corresponding viseme because the visualization of /h/ depends directly upon the following viseme. This means that the viseme for 'h' will be the same as the following viseme thereby in reality the stretching the duration of this viseme.

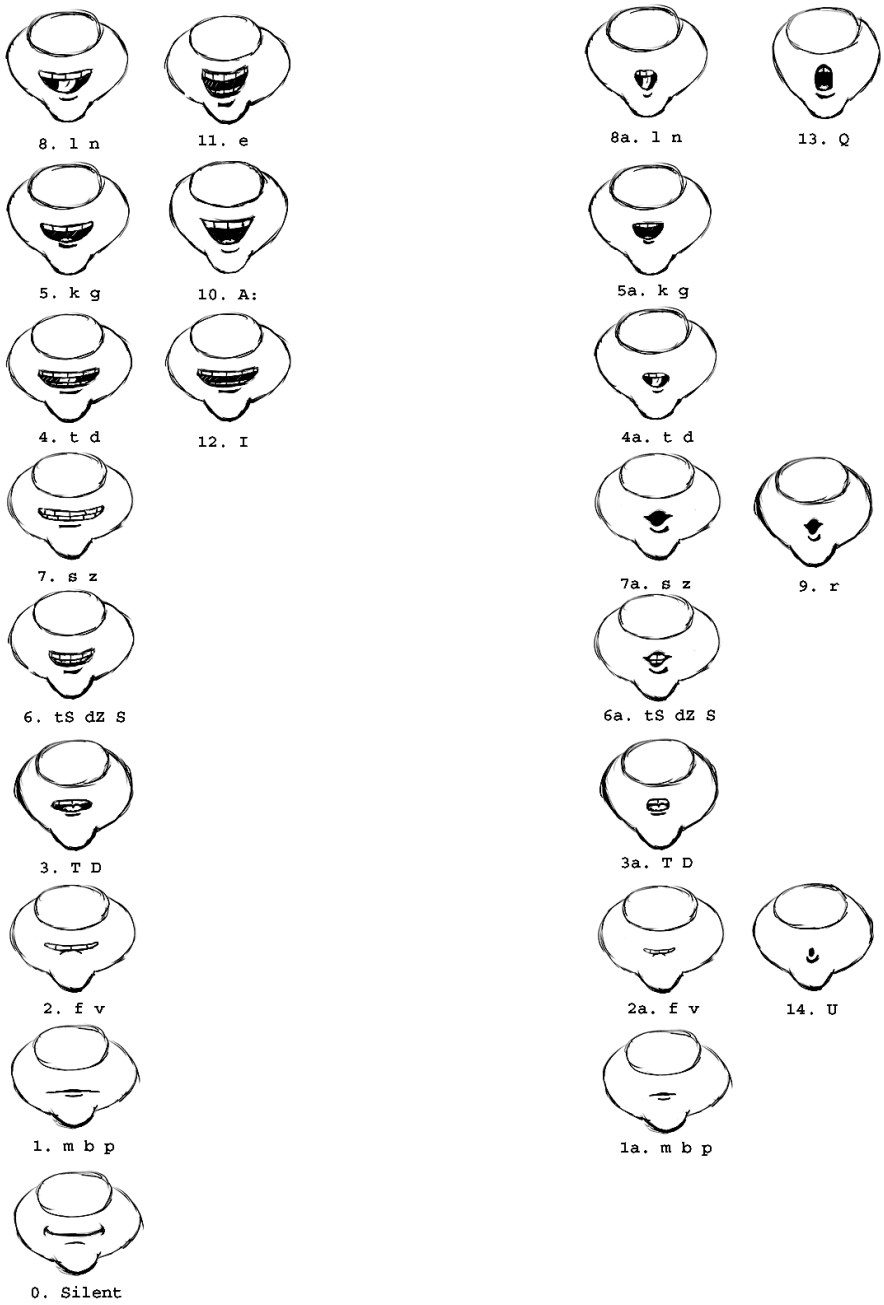

Figure F.2: Viseme set adapted to phonemes of ALSAC.

Viseme	Style	Phonemes
0	S	.
1	C u	p m b
2	C u	f v
3	C u	D
4	C u	t d
5	C u	k g R N
6	C u	j
7	C u	s
8	C u	l n
9	V r	Y y 2 9
10	V u	a {
11	V u	A E e
12	V u	i
13	V r	@ o O Q
14	V r	u
15	R	h
1a	C r	p m b
2a	C r	f v
3a	C r	D
4a	C r	t d
5a	C r	k g R N
6a	C r	j
7a	C r	s
8a	C r	l n

Table F.4: SAMPA phonemes mapped to visemes in F.2.
C: Consonant, V: Vowel
u: Unrounded mouth, r: Rounded mouth
R: repeat

F.3 Phoneme to Viseme Conversion

The training and test data used during the development of ALSAC is from the phoneme database EUROM1. ALSAC is supposed to be trained using viseme labeled data, but the data in EUROM1 is phoneme labeled. Thus, the phonemes in EUROM1 have to be converted to visemes.

A perfect conversion would require either a manual conversion by an expert or artificial intelligence that emulates the thought patterns used for manual conversion. Due to the objectives of this project and the lack of time a suboptimal solution will be designed.

The conversion is performed mapping phonemes to visemes according to table F.4. The fact that the viseme map has two visemes for each consonant complicates the conversion, because it has to be determined which version to use. It was stated previously that the consonant viseme depends on the vowel within the syllable. Thus, it is necessary to detect each syllable. This fact leads to a theory about how syllables are constructed. A syllable is composed as illustrated in figure F.3.

The essence is that a syllable must be constructed by selecting phonemes from left to right in the figure; e.g. unvoiced / nasal / vowel / semivowel / lateral (like “snail”). As illustrated syllables may not contain all groups from the figure but they must appear in that order.

For example an invalid syllable /s//t//A//t//r/ (unvoiced / unvoiced / vowel / unvoiced / semivowel). A correct syllable is /s//d//A://t/ that is (unvoiced / voiced / vowel / unvoiced). The most significant observation is that the vowels are centered between the consonants.

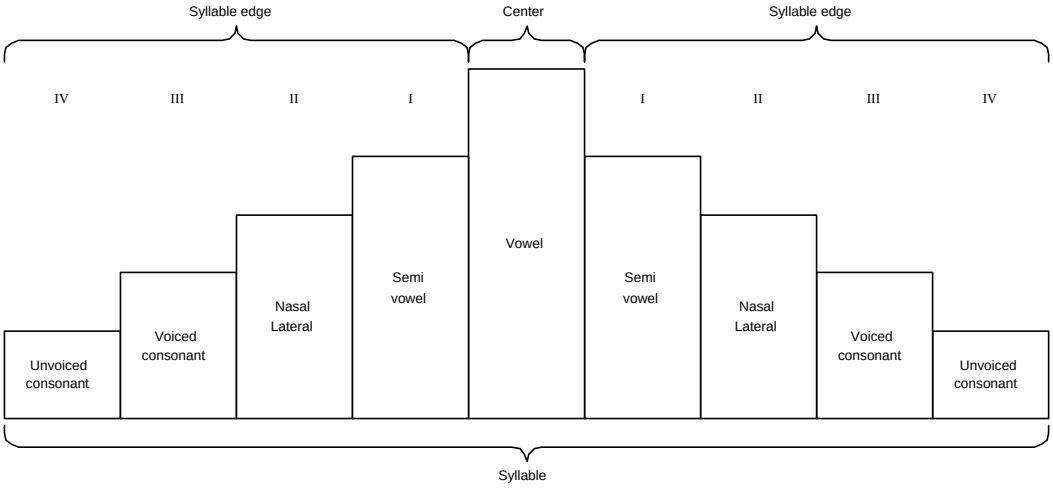

Figure F.3: Universal composition of a syllable [after ?, p. 138].

A major drawback of the method is that it is unable to determine whether a consonant between two syllables belongs to the first or the last syllable. If for instance the last consonant in the first syllable is voiced and the first two consonants in syllable two are an unvoiced and a voiced consonant, it is not possible to determine whether the unvoiced consonant belongs to the first or second syllable.

Since it is not possible to determine the above it has been chosen to make a simpler detection which is presented in the following.

F.3.1 Simple Method for Phoneme to Viseme Conversion

The simple method for phoneme to viseme conversion is based upon the assumption that the consonants are distributed equally over the syllables in an infinitely long text. This means that there are the same number of consonants on both sides of a separation of two syllables. Of course this assumption is false at least when considering single words such as “de-scribe” and “di-stri-bute”, where it is one phoneme that will be assigned wrongly. However, in many cases it will be true e.g. “num-ber” and “far-mer”.

The algorithm using the simple principle of conversion from phonemes to visemes does as follows:

Procedure (illustrated in figure F.4):

1. Read next phoneme
2. If phoneme equals /h/ convert to next viseme.
3. If phoneme is a vowel then count the number of consonants coming afterwards. Note the phoneme ending the series of consonants:
   a If last phoneme is silence then X=number of consonants
   b If last phoneme is a vowel then X=number of consonants/2 rounded down
   c If the vowel is rounded then set flag
4. If phoneme is a consonant then convert phoneme to viseme
   a If X is defined subtract 1 from X
      If flag is set convert viseme to rounded type (prefix a)
   b Else if the next vowel is rounded convert the viseme to rounded type (prefix a)

This method is almost as good as the one represented in figure F.3 all through it has one flaw compared to the other. As mentioned above it might assign one or more consonants to the wrong syllable. This would for instance happen if the first syllable has no ending consonants while the second syllable has three or four beginning consonants.

Whether this will cause a noticeable decrease in animation quality shall remain unsolved until the final visual test.

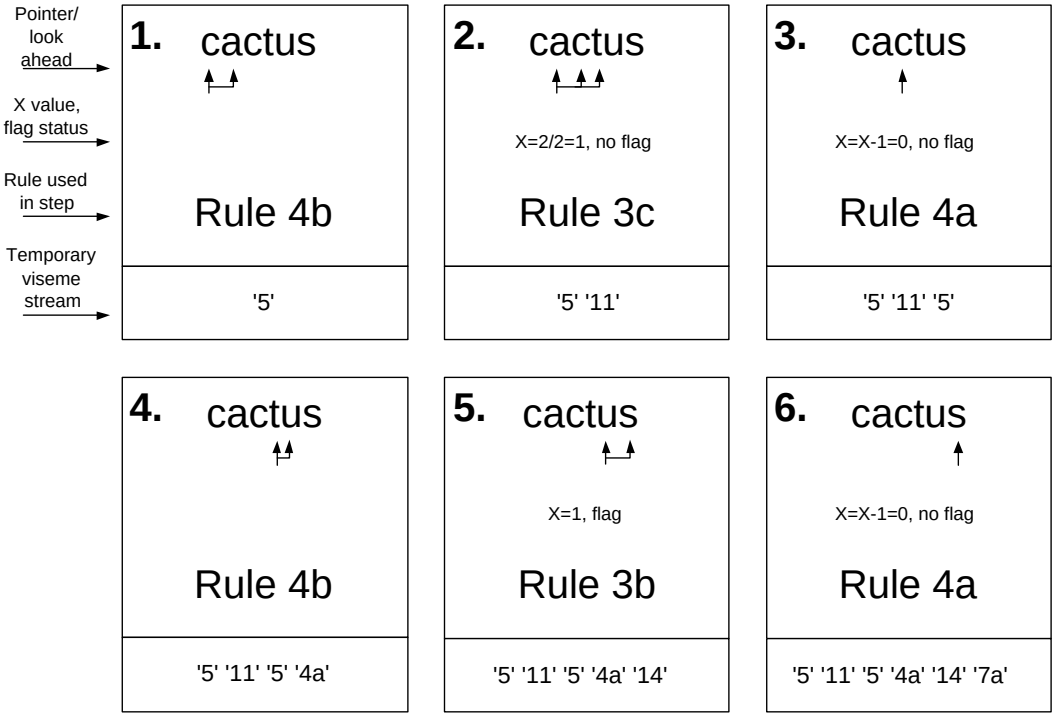

Figure F.4: Illustration of the algorithm used to convert an EUROM1 phoneme stream into an ALSAC viseme stream.

APPENDIX G

Test Sheets

The system should graded with the grades 1 to 5. Where 5 is the best score and 1 is the lowest score.

Speaker 1

Test part	Grade
A1	
B1	
C1	

Table G.1: Speaker 1

Speaker 2

Test part	Grade
A2	
B2	
C2	

Table G.2: Speaker 2

Speaker 3

Test part	Grade
A3	
B3	
C3	

Table G.3: Speaker 3