REAL TIME SPEECH DRIVEN FACE ANIMATION

Master Thesis at The Image Coding Group, Dept. of Electrical Engineering
at Linköping University
by

Andreas Axelsson
Erik Björhäll

Reg nr: LiTH-ISY-EX-3389-2003
Linköping 2003
REAL TIME SPEECH DRIVEN FACE ANIMATION

Master Thesis at The Image Coding Group, Dept. of Electrical Engineering at Linköping University by

Andreas Axelsson
Erik Björhäll

Reg nr: LiTH-ISY-EX-3389-2003

Supervisor: Jörgen Ahlberg
Examiner: Robert Forcheimer

Linköping 9th October 2003.
Sammanfattning
Abstract

The goal of this project is to implement a system to analyse an audio signal containing speech, and produce a classification of lip shape categories (visemes) in order to synchronize the lips of a computer generated face with the speech.

The thesis describes the work to derive a method that maps speech to lip movements, on an animated face model, in real time. The method is implemented in C++ on the PC/Windows platform. The program reads speech from pre-recorded audio files and continuously performs spectral analysis of the speech. Neural networks are used to classify the speech into a sequence of phonemes, and the corresponding visemes are shown on the screen.

Some time delay between input speech and the visualization could not be avoided, but the overall visual impression is that sound and animation are synchronized.

Nyckelord
Keyword
phonemes, visemes, real-time, neural networks
Acknowledgements

We would like to thank:

Our supervisor Jörgen Ahlberg for help and support.

Piotr Rudol and Mariusz Wzorek for great help with the Visage Technologies software.

Andreas Bergström, Jonas Biteus, Stefan Carlén, Sara Cerne, Stina Hellstrandh, Maria Holmström and Bertil Lyberg for helping out with the recording of the speech database.

Andreas Axelsson and Erik Björhäll
Linköping, September 2003
Abstract

The goal of this project is to implement a system to analyse an audio signal containing speech, and produce a classification of lip shape categories (visemes) in order to synchronize the lips of a computer generated face with the speech.

The thesis describes the work to derive a method that maps speech to lip movements, on an animated face model, in real time. The method is implemented in C++ on the PC/Windows platform. The program reads speech from pre-recorded audio files and continuously performs spectral analysis of the speech. Neural networks are used to classify the speech into a sequence of phonemes, and the corresponding visemes are shown on the screen.

Some time delay between input speech and the visualization could not be avoided, but the overall visual impression is that sound and animation are synchronized.

Keywords: phonemes, visemes, real-time, neural networks
Notation

Abbreviations

MU  Motion Unit
MUP  Motion Unit Parameter
NN  Neural Network
FAP  Facial Animation Parameter
FP  Feature Point
PCA  Principal Component Analysis
AV  Audio Visual
CC  Cepstrum Coefficients
MFCC  Mel-frequency Cepstrum Coefficients
MLP  Multi-layer Perceptron
LP  Linear Predictive
LPC  Linear Predictive Coding
HMM  Hidden Markov Model
FFT  Fast Fourier Transform
FLDT  Fisher Linear Discriminant Transformation
GMM  Gaussian Mixture Model
Contents

1 Introduction 1
   1.1 Thesis outline 1
   1.2 Target group 2

2 Existing Techniques 3
   2.1 Using Motion Units and Neural Networks 3
      Obtaining the MUs 3
      Real Time Audio-to-MUP Mapping 4
      Training Phase 4
      Estimation Phase 4
      A Similar Approach 4
   2.2 Combining Hidden Markov Models and
      Sequence Searching Method 5
      HMM-based Method 5
      Sequence Searching Method 5
      Combination 6
   2.3 Lip Synchronization Using Linear Predictive Analysis 6
      System Overview 6
      Energy Analysis 6
      Zero Crossing 7
      Facial Animation 7

3 Our Real Time Speech Driven Face Animation 9
   3.1 Initial Experiments 10
   3.2 Constructing a phoneme database 11
   3.3 Signal Processing 12
      Mel Frequency Cepstral Coefficients 12
      Fisher Linear Discriminant Transformation 17
   3.4 Recognition with Neural Networks 19
      The Structure of a Neural Network 19
      Training The Neural Networks 22
      Validation Of The Neural Networks 24
   3.5 Classification using a Gaussian Mixture Model 27
4 Implementation

5 Discussion

5.1 Results .................................................. 33
5.2 Limitations and Future Work .................................. 33

A The Visemes

B User’s Guide to the Program 47

List of Figures

3.1 Concatenated tube approximation of the vocal system ........... 10
3.2 Mel-scaled hamming and triangular filterbanks ................. 14
3.3 The Mel Frequency scale ........................................ 15
3.4 The MFCCs of one frame of the the letters \a and \e ................ 16
3.5 The scatter between the MFCCs of the letters \a end \e .......... 16
3.6 The scatter between the MFCCs of the letters \a and \e and also between \e and \i, before (left side) and after (right side) the FLDT 18
3.7 A nonlinear model of a neuron ..................................... 19
3.8 A three layer Neural Network with 12 inputs, 2 hidden layers consisting of 12 and 6 neurons respectively and a single neuron output . 20
3.9 The two activation functions tansig(x) and logsig(x) ........... 21
3.10 An overview of the training process ............................. 22
3.11 The overlapping between frames. The frame length is 256 samples . 23
4.1 The Graphical User Interface of the SpeechToFace application ... 32

List of Tables

3.1 Test results in percent of neural network classification of vowels ... 11
3.2 The words used to extract the different Swedish phonemes ........ 12
3.3 Predefined visemes and related phonemes in the MPEG-4 standard . 25
3.4 Validation results in percent of the neural networks, evaluated four frames at the time. 75 percent overlapping between frames has been used .................................................. 26
3.5 Validation results in percent of the neural networks, evaluated four frames at the time. No overlapping has been used ...................... 27
3.6 Validation results in percent of the Gaussian Mixture Model, evaluated four frames at the time ........................................ 28
Chapter 1

Introduction

The goal of this project is to construct and implement a real time speech to face animation system. The program is based on the Visage Technologies [2] software. Neural networks are used to classify the incoming speech, and the program shows an animated face which mimics the sound. The animation is already implemented, so the work done in this thesis is focused on signal processing of an audio signal, and the implementation of speech to lip mapping and synchronization.

It is very important that the facial animation and sound are synchronized, which makes demands on the program considering the time delay. Some time delay must be accepted, since speech has to be spoken before it can be classified. The goal set for this thesis is 100 ms as the upper limit of delay from input speech to visualization.

Both Matlab and Visual C++ are used to implement the work into a Windows application.

1.1 Thesis outline

The thesis is divided into the following chapters:

- **Chapter 2** describes earlier work on different speech driven facial animation techniques.

- **Chapter 3** describes our approach.

- **Chapter 4** is about the implementation

- **Chapter 5** contains results and discussions about future work.
1.2 Target group

This thesis applies to engineers and engineer-students with general knowledge in signal processing.
Chapter 2

Existing Techniques

Many different techniques have been proposed to map speech to lip movements in real time. In this chapter, some of them are described briefly.

2.1 Using Motion Units and Neural Networks

This technique developed by P Hong, Z Wen and T. S. Huang is presented in [8],[12]. It uses so called Motion Units (MUs) together with Neural Networks (NNs) in order to visualize the mapping of speech to lip movements in real time. As input it takes a speech stream and generates MPEG-4 compatible Facial Animation Parameters (FAPs) as output.

The MUs are a representation of the facial deformations during speech, and are learned from a set of real face deformations. It is assumed that any facial deformation can be described by a linear combination of MUs weighted by the corresponding MU parameters (MUPs).

Obtaining the MUs

First, a set of MUs are learned from real face deformations of a test subject. A number of markers are placed in the lower part of the subject’s face. Those markers cover the facial Feature Points (FPs) that are defined by the MPEG-4 Face Animation (FA) standard, to describe the movements of the lips and cheeks. Since the movements of the upper part of the face are not so closely correlated with speech, no markers are put there. A mesh is then created on the basis of the markers.

The subject is video taped while pronouncing all English phonemes while keeping the head as fixed as possible. Meanwhile the markers are tracked. The selection of facial shapes can now be done manually from the image samples so that each viseme and transition between each pair of visemes are evenly represented. The
deformation of the face are calculated with respect to the positions of the markers in the neutral face.

**Real Time Audio-to-MUP Mapping**

In order to train the real-time audio to MUP mapping, audio-visual (AV) training data is required. An AV database is collected by letting a subject read a text while being video taped. A MU-based facial motion tracking algorithm is used to analyse the facial image sequence of the video. The results are represented as MUP sequences, which are used as the visual feature vectors of the AV database. The corresponding audio feature vectors are chosen by calculating ten Mel-Frequency Cepstrum Coefficients (MFCCs) of each audio frame. The AV training data is then divided into 44 subsets, since a phoneme symbol set consisting of 44 phonemes were used.

**Training Phase**

When the AV database is built, Multilayer Perceptrons (MLPs), in this case three-layer perceptrons, are trained to estimate MUPs from audio features using each AV training subset.

The inputs of an MLP are the audio feature vectors of seven consecutive speech frames, 3 backward, current and 3 forward frames. This will result in a time delay between input and output of about 100 ms, since the video is digitalized at 30 frames per second. Hence, this method is only in near real time. However, depending on the application, some time delay can be acceptable. The output of the MLP is the visual feature vector, i.e. MUP sequences, of the current frame. The error backpropagation algorithm is used to train the MLPs.

**Estimation Phase**

In the estimation phase, an audio feature vector is first classified into one of the 44 groups. The corresponding MLP is then selected to estimate the MUPs for the audio feature vector which will give the correct facial expression. Some smoothing may be necessary in case of jerky mapping.

**A Similar Approach**

A similar technique is presented in [11] by the same authors. It also uses MUs together with Neural Networks to accomplish the auditory-visual speech recognition in real time.

Instead of MFCCs, as described earlier, twelve LPC coefficient are calculated and used as the audio feature vector. Previously the AV training data was divided into 44 subsets, but now the phonemes are instead divided into 12 classes according to their lip shapes (5 vowel groups, 6 consonant groups, and silence). A MLP is
trained to estimate the visual feature given an audio feature and then used for each audio group. As before, each input vector to the MLPs is the audio feature vector taken at 7 consecutive time frames and the output vector is the visual feature vector, i.e, the MUPs, at the current time frame.

2.2 Combining Hidden Markov Models and Sequence Searching Method

This technique is presented by Y. Huang, X. Ding, B. Guo and H. Y. Shum [13]. It combines Hidden Markov Models (HMMs) with sequence searching, in order to animate lip movements from speech in real time. Acoustic feature vectors are calculated from input voice and are directly used to drive the system without considering phonetic representation. The minimal distance between the acoustic vectors and the vocal data of sequences in a predefined database is found. If the distance is larger than a certain threshold, the face is synthesized using an HMM-based method. Otherwise the face in the corresponding sequence is exported.

In the training phase, a sequence of training video including synchronized speech and video is prepared. With the vocal and facial data gotten from the training video, HMMs are trained and a database of sequences are created.

The HMM-based Method

The HMM-based method maps from an input speech signal to lip parameters through HMM states determined by the Viterbi algorithm. Considering the efficiency requirement of real-time synthesis algorithms, a database is established containing only 16 representative faces selected from 2000 different ones. The acoustic feature vectors are calculated and probabilities for each face shape are calculated using the Viterbi algorithm. The face shape with the largest probability is chosen as the correct face.

The Sequence Searching Method

The HMM-based method has some limitations. The most prominent problem is that the number of face shapes in the database is limited. This makes the animation look a bit simplistic. A different approach is to reuse the training video as much as possible when the input voice is similar to the corresponding voice sequence in the database. Each sequence has a number of face shapes and acoustic feature vectors.

The vocal frames are compared with the feature vectors of all the sequences in the database and the shape of the sequence with the minimal distance is exported. The following vocal frames are then compared to the following feature
vectors in the sequence $n$. If the distance is larger than a threshold the whole sequence database must be searched again, otherwise the next face shape of sequence $n$ is chosen.

**Combination**

The sequence searching method also has some limitations. The size of the sequence database cannot be too big, because the entire database must be searched frequently to find the minimum distance. If the input voice is different from the optimal sequence some distortion will occur. Therefore the HMM-based method is used when minimum distance is larger than a threshold.

### 2.3 Lip Synchronization Using Linear Predictive Analysis

Another technique is presented by Sumedha Kshirsagar and Nadia Magnenat-Thalmann [9]. Using filter coefficients from LP analysis of the speech, a set of reflection coefficients can be calculated. These coefficients are closely related to the vocal tract shape. Since the vocal tract shape can be correlated with the phoneme being spoken, LP analysis can be directly applied to phoneme extraction. Neural networks are used to train and classify the reflection coefficients into a set of vowels. In addition, average energy is used to take care of vowel-vowel and vowel-consonant transitions. Zero-crossing information is used to detect the presence of fricatives.

**System Overview**

12 reflection coefficients are calculated as a result of LP analysis. The coefficients are obtained from sustained vowel data and are used to train the neural network. As a result, one of 5 chosen vowels (/a/, /e/, /i/, /o/, /u/) is obtained for the frame. The average energy is calculated and is used to decide the intensity of the detected vowel. Zero crossings are calculated to decide the presence of unvoiced fricatives and affricates (/sh/, /ch/, /zh/ etc.). Unvoiced speech contains a larger number of zero crossings per time frame than voiced speech. Finally the FAPs, included in the MPEG-4 standard, are generated depending on the phoneme input.

**Energy Analysis**

The application of the vowels alone for speech animation is not sufficient. The vowel-to-vowel transition and the consonant information is missing, both very important for speech animation. The consonants are typically produced by creating a constriction at some point along the length of the vocal tract. During such constrictions/closures, the energy in the speech signal diminishes. Therefore the average energy of a speech frame is used to modulate the recognized vowel.
2.3 Lip Synchronization Using Linear Predictive Analysis

Zero Crossing
Using the energy content of the signal may result in false closure of the mouth, especially in the case of affricates and unvoiced fricatives. For such cases, the average zero crossing rate of the frame can be used. In case of the presence of low energy in the speech frame, the zero crossing rate decides the phoneme.

Facial Animation
Since this method generates FAPs, the phonemes extracted can be applied to any parameterized face model for speech animation.
Existing Techniques
Chapter 3

Our Real Time Speech Driven Face Animation

In order to map speech to lip movements in realtime, some sort of classification of the speech is necessary. First, a representation of the speech must be done. Two different ways to do this were tested:

- LP analysis resulting in Reflection Coefficients.
- Cepstrum Coefficients.

As mentioned earlier, LP analysis of the speech results in a set of reflection coefficients that are closely related to the vocal tract shape [9]. As can be seen in Figure 3.1, the vocal system can be represented by a concatenated tube approximation. The comparison between the acoustic tube model and the LP derived model has led to the conclusion that the reflection coefficients $r_i$, are directly related to the
Our Real Time Speech Driven Face Animation

Figure 3.1. Concatenated tube approximation of the vocal system

vocal tract area by the following equation:

\[ r_i = \frac{A_{i-1} - A_i}{A_{i-1} + A_i} \]

Cepstrum Coefficients (CCs) are commonly used to represent speech and are calculated as the real part of:

\[ \mathcal{F}^{-1}\{\log(\text{abs}(\mathcal{F}\{x\}))\} \]

where \( x \) is the input speech and \( \mathcal{F} \) denotes the Fourier Transform. An input consisting of \( N \) samples will result in \( N \) Cepstrum Coefficients.

Calculations of these two different kinds of coefficients were made on some test speech and then used as inputs to a Neural Network. The results showed that the network trained with Cepstrum Coefficients classified input speech with fewer classification errors than the network trained with Reflection Coefficients. Therefore, it was decided to test CCs together with a Neural Network as an initial approach to classify and recognize speech.

3.1 Initial Experiments

Recordings of silence and five vowels (a,o,u,e,i) from two different speakers (the authors) were made and stored as digital audio files. Both short and long vowels were pronounced and the sampling frequency was 44100 Hz with 16 bits accuracy.

The files are first read in Matlab, which generates vectors with sample values in the range \([-1,1]\). The audio vectors are then divided into time frames of 20 ms which corresponds to 882 samples. About 2.5-3 seconds of each vowel were recorded which gives us approximately 125-150 frames per vowel.

In order to achieve recognition that is independent of the intensity of the speech, the samples in each frame are first normalized with respect to the largest value in the frame. CCs are then calculated on the 882 samples. The first 20 of the 882 coefficients are considered to be a good representation of the speech in each frame. The set of CCs are then used as a training set in order to train a three layer back
propagating Neural Network with 20 inputs, 10 hidden nodes and 3 outputs. The choice of 3 outputs are made since three bits are needed to represent the five vowels binary at the output.

The result shows good recognition of vowels, however some vowels are better recognized than others. Table 3.1 shows the classification results on an evaluation data set from one of the two speakers in the training data set.

<table>
<thead>
<tr>
<th>Expected</th>
<th>a</th>
<th>o</th>
<th>u</th>
<th>e</th>
<th>i</th>
<th>silence</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>93</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>o</td>
<td>0</td>
<td>90</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>u</td>
<td>21</td>
<td>0</td>
<td>72</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>83</td>
<td>17</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>82</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>silence</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>98</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1. Test results in percent of neural network classification of vowels

As can be seen in table 3.1 the results are overall good. However, since the training data set is based on speech from only two different persons and the evaluation data is produced by one of these persons, it is difficult to draw any major conclusions from these results. Nevertheless, it seems like a good idea to use neural networks in order to classify speech into phonemes.

### 3.2 Constructing a phoneme database

Our approach to represent the speech with Cepstrum Coefficients, and then train a Neural Network with these CCs as input, seems promising. This is done to be able to recognize speech and classify it into different phoneme classes. In order to obtain training data for the neural network, it is necessary to collect a training set with phonemes. This was done in the speech lab at the Dept. of Computer and Information Science, at Linköping University. The model should be speaker independent, i.e. the results should be satisfactory regardless of the speaker’s age, sex, accent etc. In an attempt to achieve this independence, nine different speakers, six male and three female, were asked to read the words in Table 3.2. Colon after the vowel means long vowel.

The audio software used for recording and editing is ‘wavesurfer’ [3] and the sampling frequency is 16000 Hz with 16 bits accuracy. Each Swedish phoneme is extracted from each of the three words corresponding to the phoneme. The phonemes are manually cut out from the words and stored as wav files. Hence, we have a set of phonemes consisting of three samples of each Swedish phoneme from each one of the nine test subjects. This gives us 27 versions of each phoneme in our database.
Our Real Time Speech Driven Face Animation

Since there are 37 Swedish phonemes plus recordings of silence in the database, there are now 38 subsets of training data available for the Neural Network.

### 3.3 Signal Processing

**Mel Frequency Cepstral Coefficients**

As mentioned above, Cepstrum Coefficients in their simplest form are the inverse Fourier Transform of the frequency spectrum in logarithmic amplitudes. However, when representing the speech with CCs, the classification results using neural networks were not satisfactory. The reason for the poor results is that we now have 38 different classes to separate instead of 6, as in the initial experiments. This is apparently difficult with the quite simplistic CCs leading to overlapping decision boundaries as a consequence.

Instead, the speech is represented with the more complex Mel Frequency Cepstral Coefficients (MFCCs). These coefficients are commonly used as a technique for automatic speech recognition and takes the characteristics of the human auditory system under consideration [10].

### Table 3.2. The words used to extract the different Swedish phonemes

<table>
<thead>
<tr>
<th>Phoneme</th>
<th>Word</th>
<th>Phoneme</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>arm</td>
<td>b</td>
<td>boll</td>
</tr>
<tr>
<td>a:</td>
<td>apa</td>
<td>d</td>
<td>data</td>
</tr>
<tr>
<td>o</td>
<td>orm</td>
<td>f</td>
<td>fisk</td>
</tr>
<tr>
<td>o:</td>
<td>osa</td>
<td>g</td>
<td>gala</td>
</tr>
<tr>
<td>u</td>
<td>unna</td>
<td>h</td>
<td>halsa</td>
</tr>
<tr>
<td>u:</td>
<td>utan</td>
<td>j</td>
<td>gilla</td>
</tr>
<tr>
<td>e</td>
<td>ett</td>
<td>k</td>
<td>kall</td>
</tr>
<tr>
<td>e:</td>
<td>el</td>
<td>l</td>
<td>laga</td>
</tr>
<tr>
<td>i</td>
<td>inner</td>
<td>m</td>
<td>mamma</td>
</tr>
<tr>
<td>i:</td>
<td>ivrig</td>
<td>n</td>
<td>nummer</td>
</tr>
<tr>
<td>y</td>
<td>ylle</td>
<td>p</td>
<td>pulka</td>
</tr>
<tr>
<td>y:</td>
<td>yla</td>
<td>r</td>
<td>rista</td>
</tr>
<tr>
<td>å</td>
<td>åtta</td>
<td>s</td>
<td>samla</td>
</tr>
<tr>
<td>å:</td>
<td>åt</td>
<td>t</td>
<td>tala</td>
</tr>
<tr>
<td>ä</td>
<td>äkta</td>
<td>v</td>
<td>vika</td>
</tr>
<tr>
<td>ä:</td>
<td>äta</td>
<td>rt</td>
<td>bort</td>
</tr>
<tr>
<td>ö</td>
<td>öster</td>
<td>ng</td>
<td>samling</td>
</tr>
<tr>
<td>ö:</td>
<td>öl</td>
<td>sh</td>
<td>kännra</td>
</tr>
<tr>
<td>ch</td>
<td>chans</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phoneme</th>
<th>Word</th>
<th>Phoneme</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch</td>
<td>choklad</td>
<td>ch</td>
<td>chark</td>
</tr>
</tbody>
</table>
Calculation of the MFCCs is done in the following steps:

- Frame-blocking and windowing.
- Fourier Transformation.
- Mel-Scaling by applying a filterbank.
- Cosine Transformation.

The speech is first divided into time frames consisting of an arbitrary number of samples. By introducing overlapping of the frames, the transition from frame to frame is smoothed. All time-frames are then windowed with a Hamming window to eliminate discontinuities at the edges for subsequent Fourier Transforms. After the windowing, the spectrum of each frame is calculated using Fast Fourier Transformation (FFT). For a signal $x(n)$, the spectrum of $N$ samples is defined as:

$$X(k) = \sum_{n=1}^{N} x(n) \cdot e^{-j2\pi(k-1)(n-1)/N}$$

Choosing the frame length in powers of 2 is optimal for the calculation of the FFT-algorithm by computers.

The fourier transformed signal is then passed through one of the sets of band-pass filters illustrated in Figure 3.2 on the next page.
The triangular filters are more suitable if the framesize is small. The bandwidths and center frequencies of the filters in the filterbanks have been determined by experimental results on human hearing. A good approximation of the non-linear frequency scale of the human auditory system is the mel-frequency scale, which is approximately determined by the following equation:

\[ m = 1127 \cdot \ln(1 + \frac{f}{700}) \]

The mel-scale is approximately linear below 1kHz and logarithmic above 1kHz (see Figure 3.3).

Each mel-scaled filter in the filterbank is multiplied by the spectrum and summed so that the result is one magnitude value per filter. This can be done by a simple matrix operation. These magnitude value represents the sum of amplitudes of the spectrum in each filter frequency band. The outputs are now mel-scaled and the number of outputs equals the number of filters in the filterbank.
Unlike the calculation of the CCs, the inverse Fourier Transform is instead replaced with Cosine Transformation of the outputs from the filterbank. The advantage is that the Cosine transform gives a result that approximates the results from a Principle Component Analysis (PCA). Consequently, after the DCT, the coefficients are ranked according to significance. The Discrete Cosine Transform is defined as:

\[
y(k) = w(k) \sum_{n=1}^{N} x(n) \cdot \cos \left( \frac{\pi(2n-1)(k-1)}{2N} \right), \quad k = 1, ..., N
\]

where \( w(k) = \begin{cases} \sqrt{1/N} & k = 1 \\ \sqrt{2/N} & 2 \leq k \leq N \end{cases} \)

The 0th coefficient is excluded, although it indicates energy, since it is generally considered somewhat unreliable. Thus, a filterbank of \( L \) filters generates \( L - 1 \) coefficients. The calculation of the MFCCs is now complete and the coefficients for the letters \( a \) and \( e \) and also the scatter between the two classes are shown in Figure 3.4 and Figure 3.5. The framesize is 256 samples and the sample frequency is 16 kHz. There are 29 triangular filters in the filterbank and the windowing in the time domain is done with a Hamming window. The MFCCs are 28-dimensional, but for simplicity, the scatter in the first two dimensions is plotted. Note that even if there are no separation in the first two dimensions between two phonemes, they may still be separated in some of the other dimensions.
The MFCCs were calculated using a speech processing toolbox for matlab [4]. Some modifications of the functions were made to improve the speed.
3.3 Signal Processing

Fisher Linear Discriminant Transformation

Since there are 38 different classes, it is difficult for the neural networks to separate some classes from each other with incorrect classification of phonemes as a result. It may thus be necessary to separate the classes from each other even more than the separation accomplished with the MFCCs. Therefore, a Fisher Linear Discriminant Transformation (FLDT) [5] is done on our MFCC-vectors. The FLDT reshapes the scatter of a data set to maximize class separability, and is defined as the ratio of a between-class matrix to a within-class matrix. The method is defined as follows:

Let \( \{ \vec{x}_i \}_{i=1}^N \) be a set of \( N \) column vectors of dimension \( D \). In our case, \( N \) equals the total number of \( D \)-dimensional MFCC vectors of the phoneme classes. The mean of the dataset is:

\[
\vec{m}_x = \frac{1}{N} \sum_{i=1}^{N} \vec{x}_i
\]

There are \( K=38 \) classes \( \{ C_1, C_2, \ldots, C_K \} \) and the mean of class \( k \) with \( N_k \) members is:

\[
\vec{m}_{xk} = \frac{1}{N_k} \sum_{\vec{x}_i \in C_k} \vec{x}_i
\]

The between class scatter matrix is defined as:

\[
S_B = \sum_{k=1}^{K} N_k (\vec{m}_{xk} - \vec{m}_x)(\vec{m}_{xk} - \vec{m}_x)^T
\]

while the within class scatter matrix is defined as:

\[
S_W = \sum_{k=1}^{K} \sum_{\vec{x}_i \in C_k} (\vec{x}_i - \vec{m}_{xk})(\vec{x}_i - \vec{m}_{xk})^T
\]

Let \( W = \{ \vec{w}_1, \vec{w}_2, \ldots, \vec{w}_D \} \) be the eigenvectors of \( S_B S_W^{-1} \) and \( W_d = \{ \vec{w}_1, \vec{w}_2, \ldots, \vec{w}_d \} \) the eigenvectors corresponding to the \( d \) largest eigenvalues. Then, the projection of a vector \( \vec{x} \) into a subspace of dimension \( d \leq D \) is given by:

\[
\vec{y} = W_d^T \vec{x}
\]

If no reduction of dimensions is wanted, \( d=D \).

Note that if there is no separation at all between two (or more) classes before the FLDT, there will not be any noticeable separation after the transformation either. However, if there is a only a slight confusion between classes, the FLDT will separate them satisfactorily. Figure 3.6 shows the scatter between the MFCCs of \( \backslash a \) and \( \backslash e \) and also between \( \backslash e \) and \( \backslash i \) before and after the FLDT.
Figure 3.6. The scatter between the MFCCs of the letters \( a \) and \( e \) and also between \( e \) and \( i \), before (left side) and after (right side) the FLDT.

As can be seen, the separation between \( a \) and \( e \) has improved, while the separation between \( e \) and \( i \) still is poor. The first two dimensions of the MFCCs are displayed.

The overall procedure of the coefficient calculation can be summarized as:

\[
X(t) \rightarrow \text{Frame-blocking} \quad + \quad \text{Windowing} \rightarrow \text{DFT} \rightarrow \text{Mel-scale} \rightarrow \text{DCT} \rightarrow \text{FLDT} \rightarrow c(n)
\]
3.4 Recognition with Neural Networks

The Structure of a Neural Network

In our case, Neural Networks [6] are used to recognize incoming speech and divide it into the correct phoneme class. A neuron is an information-processing unit that is essential for the operation of a NN. Figure 3.7 shows a model of the neuron.

![Figure 3.7. A nonlinear model of a neuron.](image)

Basically, the model have three important properties:

- A set of input weights. For every input $x_i$, there exists a weight $w_i$. Each input is multiplied with its corresponding weight, i.e the weight can be seen as the strength of a certain input.

- An adder for summing the weighted input signals.

- An activation function, $f(\cdot)$, which purpose is to limit and determine the output signal of the neuron to some finite value. Typically, the amplitude range of the final output, $y$ is in the range of [0,1] or [-1,1].

The neuron model also includes an externally applied bias, $b$, which have the effect of lowering or increasing the input to the activation function, depending on whether it is positive or negative.

In other words, the output $y$, can be written as:

$$y = f(u + b) \quad \text{where} \quad u = \sum_{j=1}^{n} w_j x_j$$
A Neural Network can now be built with a number of neurons. A *multilayer feed-forward net* is a network that has an *input layer* of source nodes that through a number of *hidden layers* of neurons projects onto an *output layer* of neurons. These are the kind of networks that have been used for the speech recognition in this project. Figure 3.8 shows a three layer neural network consisting of an input layer of 12 inputs, two hidden layers consisting of 12 and 6 neurons respectively and a single neuron output layer.

*Figure 3.8.* A three layer Neural Network with 12 inputs, 2 hidden layers consisting of 12 and 6 neurons respectively and a single neuron output.

The network takes a 12-dimensional vector as input and produces a single value as output. The purpose with the network is to recognize a pattern within a set of 12-dimensional vectors that are fed to the net. In order to achieve the ability to separate inputs from each other, the network needs to be *trained* to *learn* from its environment.

For example, if a network should be able to separate the letter 'a' from the letter 'e', it is trained in the following way:

Assuming that there is a set of 12-dimensional vectors representing the letter 'a
and a corresponding set representing the letter \( \vec{e} \). Then, the NN in figure 3.8 can be used for the separation. Let us denote the set of \( a \)-vectors as \( \{ \vec{a}_i \}_{i=1}^N \) and the set of \( e \)-vectors as \( \{ \vec{e}_i \}_{i=1}^M \). In the training phase, the user first specifies the desired response from the network. In this example, the desired response would be a \( 1 \times (M + N) \) dimensional vector with \( N \) ones and \( M \) zeros. In other words, we want the network to produce 1 as output when \( \vec{a} \) is the input vector and 0 if \( \vec{e} \) is the input vector.

Two different kinds of activation functions are shown in Figure 3.9. For the neurons in the first two layers, the \( \text{tansig} \) function is used, defined as

\[
\text{tansig}(x) = \frac{2}{1 + e^{-2x}}
\]

For the output layer, the \( \text{logsig} \) function is used, defined as

\[
\text{logsig}(x) = \frac{1}{1 + e^{-x}}
\]

The reason for choosing the \( \text{logsig} \) function in the output layer is that it produces outputs within the range of \([0,1]\), which is desired here.

![Figure 3.9. The two activation functions \( \text{tansig}(x) \) and \( \text{logsig}(x) \).](image_url)
Figure 3.10 describes the training process. \( y(n) \) is the output, \( d(n) \) the desired response, and \( e(n) = y(n) - d(n) \) is the error signal, i.e. the difference between the output and the desired output.

The error signal \( e(n) \) is used in a control mechanism which purpose is to adjust the weights of the neurons in the network. The adjustments are designed to make the output signal \( y(n) \) come closer to the desired response \( d(n) \). This is achieved by minimizing a cost function \( \xi(n) \), defined as:

\[
\xi(n) = \frac{1}{2} e^2(n)
\]

The training process is repeated a number of times. In every iteration, the vectors in the training set are processed through the network and the corresponding outputs are compared with the desired outputs, followed by adjustments of the weights. The training algorithm used for adjusting the weights in this work is the Levenberg-Marquardt backpropagation algorithm [1]. The step-by-step adjustments are carried on either until the system reaches a steady state (the weights are stabilized) or until the process has completed a number of chosen iterations. At that point, the learning process is terminated.

**Training Neural Networks**

The phoneme database is now used as a training set in order to train Neural Networks. When the MFCCs are extracted from the speech, the frame length must be chosen so that the frame contains enough information of the speech in order to achieve MFCCs that are distinguishing for the characteristics of the different phonemes. The frame length is set to 256 samples. Some phonemes are quite short...
and a longer frame length, for example 512 samples, could exceed the actual length of certain phonemes. The frame would then also cover the transition between two phonemes, which is not preferable for training data. As mentioned before, the optimal length of the frame is in powers of 2. With a higher sample frequency, longer frames would be possible but due to the fairly low sample frequency of 16 kHz, the length is set to 256. A frame length of 128 samples was also tested, but did not appear to contain enough information to separate the phonemes from each other.

The dimension of the MFCC vectors is another parameter that affects the separability of the phoneme classes. As mentioned earlier, the coefficients in MFCC vectors are ranked according to significance, and the information content diminishes towards the end of the vector. The choice of 12 dimensions was made and therefore the 12 first coefficients in each MFCC vector was picked. In the following FLDT, no reduction of dimensions is made \((d = 12)\). In order to smooth the transition from one phoneme to another, overlapping between frames is introduced. As a first approach the overlap is chosen so that 75 percent of the samples in the current frame is reused. Figure 3.11 illustrates the overlapping.

![Figure 3.11](image)

After the calculation of MFCCs followed by FLDT, we now have 38 subsets of 12-dimensional MFCC vectors as training data. A single network with 6 outputs was first tested on the entire training set. The output targets are the binary representation of the numbers 1-38. The results turned out to be poor. The computation time was long due to the complexity of the large network and the recognition was not satisfactory. Naturally, it will be more difficult to separate classes from each other when the number of classes increases, and it seems that a single network with 6 outputs could not manage the complexity of the decision boundaries.

The 12-dimensional MFCC vectors are instead used as inputs to 38 different networks. For each phoneme class, a Neural Network with 12 inputs, a number of hidden nodes and 1 output, is trained with the training data in the corresponding MFCC vector subset (see Figure 3.8). The number of hidden nodes for each network is tuned by running a training session several times on the data and studying the classification results. The network for each subset is expected to give 1 as output when the corresponding phoneme is present at the input and 0 otherwise.
Validation of the Neural Networks

The recognition part is straightforward: Each of the 38 networks takes the speech as input and the recognized phoneme is simply the one that corresponds to the network that produces 1 as output.

As soon as a frame of the incoming speech is classified into the correct phoneme class, the corresponding viseme is to be decided and sent to the animated face model. Ideally, the network corresponding to the correct phoneme should return 1 and the rest 0, but that is seldom the case. Instead, the outputs lies within the interval [0,1] and the network that produces the largest output is picked as the correct phoneme. Errors may occur, i.e. an incorrect phoneme can be identified as the correct one by the neural networks. If a viseme should be picked in each frame, an error would cause a sudden discontinuous facial expression. Therefore, four consecutive frames, i.e 1024 samples are considered, and the four outputs from each network are summarized. The correct viseme is the one that corresponds to the network with largest output sum. This will result in a time delay from input to output. Since the frame length and sample frequency is 256 samples respectively 16 kHz, the delay will be \( t_s = (4 \times 256)/16000 = 64\text{ms} \). Furthermore, the computations will result in additional time delay, \( t_{comp} \). Consequently, the model is only in near real time with a total time delay of:

\[
t_{delay} = t_s + t_{comp}
\]

It is not necessary to have a unique viseme for every phoneme, since there are several phonemes that have the same or similar facial expression. There are 15 predefined viseme Facial Animation Parameters (FAP group 1) in the MPEG-4 standard [7]. Since the visemes used are not developed for a specific language, the Swedish phonemes are divided into the viseme class that best describes the phoneme. The 15 viseme classes with related phonemes are described in Table 3.3. The visemes are also shown in Appendix A.
### Table 3.3. Predefined visemes and related phonemes in the MPEG-4 standard

<table>
<thead>
<tr>
<th>Viseme number</th>
<th>Phonemes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>silence</td>
</tr>
<tr>
<td>1</td>
<td>p, b, m</td>
</tr>
<tr>
<td>2</td>
<td>f, v</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>t, d, g</td>
</tr>
<tr>
<td>5</td>
<td>k, h, ng, rt</td>
</tr>
<tr>
<td>6</td>
<td>sh, ch</td>
</tr>
<tr>
<td>7</td>
<td>s</td>
</tr>
<tr>
<td>8</td>
<td>n, l</td>
</tr>
<tr>
<td>9</td>
<td>r</td>
</tr>
<tr>
<td>10</td>
<td>a, a:</td>
</tr>
<tr>
<td>11</td>
<td>e, e:, ä, ä:</td>
</tr>
<tr>
<td>12</td>
<td>i, i:, j</td>
</tr>
<tr>
<td>13</td>
<td>o, o:, å, å:</td>
</tr>
<tr>
<td>14</td>
<td>u, u:, ö, ö:, y, y:</td>
</tr>
</tbody>
</table>

The confusion matrices below shows the validation results of the classification with the NNs. Results are shown both with overlapping between frames (Table 3.4) and without overlapping (Table 3.5).
Table 3.4. Validation results in percent of the neural networks, evaluated four frames at the time. 75 percent overlapping between frames has been used.
3.5 Classification using a Gaussian Mixture Model

Table 3.5. Validation results in percent of the neural networks, evaluated four frames at the time. No overlapping has been used

<table>
<thead>
<tr>
<th>Exp. Class</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>93</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>53</td>
<td>7</td>
<td>0</td>
<td>20</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>9</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>69</td>
<td>31</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>-</td>
<td>69</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>-</td>
<td>14</td>
<td>82</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>69</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>77</td>
<td>7</td>
<td>0</td>
<td>17</td>
<td>0</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>59</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>94</td>
<td>0</td>
<td>94</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>84</td>
<td>0</td>
</tr>
</tbody>
</table>

As can be seen when comparing the two tables, the method with overlapping between frames gives an overall better recognition. However, when working without overlapping, the results are still satisfactory. As mentioned before, some phonemes are better recognized than others as a result of overlapping decision boundaries between the phonemes.

3.5 Classification using a Gaussian Mixture Model

To get an estimation of how good the results from the Neural Network classifier are, they are compared with a Gaussian Mixture Model (GMM) classifier. In a GMM, a mixture of Gaussian distributions are fit to a set of training data. To fit the distributions to the data we use Matlab functions (written by Jörgen Ahlberg). The starting points are randomly picked from the single-Gaussian distribution of the training data. The distributions are then adapted to the training data by iterating the Expectation-Maximization (EM) algorithm. The algorithm operates in the following two steps:

- **E-step:** Estimate the distribution given the data and current value of the parameters.
- **M-step:** Finds the new parameter set that maximizes the probability distribution.
The result is, for each class, a mixture model with the probability density function (pdf):

\[
f(\bar{x}) = \sum_{k=1}^{K} \frac{w_k}{\sqrt{(2\pi)^D | C_k |}} e^{-\frac{1}{2}(\bar{x} - \bar{m}_k)^T C_k^{-1}(\bar{x} - \bar{m}_k)},
\]

i.e., a weighted sum of \( K \) Gaussian distributions with means and covariances \( \bar{m}_k \) and \( C_k \). \( \{w_k, \bar{m}_k, C_k\}_{k=1}^{K} \) are estimated by the EM algorithm.

Below you can see a confusion matrix of the results from the GMM-classifier.

<table>
<thead>
<tr>
<th>Exp. Class</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>97</td>
<td>0</td>
<td>3</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>40</td>
<td>0</td>
<td>-</td>
<td>27</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>0</td>
<td>36</td>
<td>-</td>
<td>27</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>-</td>
<td>69</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>-</td>
<td>6</td>
<td>69</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>5</td>
<td>14</td>
<td>81</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>106</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td>14</td>
<td>0</td>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>4</td>
<td>41</td>
<td>5</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>5</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>8</td>
<td>10</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>71</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.6. Validation results in percent of the Gaussian Mixture Model, evaluated four frames at the time.

When comparing the GMM confusion matrix and the Neural Network confusion matrix, one can see some differences. For most phonemes the method using neural networks gives the best result. This is expected because the GMM method does not consider other phonemes. When a mixture of gaussian distributions are fitted to training data for a specific phoneme, it does not consider the fact that other phonemes may be similar. Therefore two similar phonemes will have a similar mixture of distributions and will be difficult to separate.

With neural networks it is different. When training a neural network, the other phonemes are also a part of the training data. If for example a network is trained to recognize the phoneme \( \text{\textbackslash a} \), it is also trained not to recognize the other phonemes. In our case the network is trained to give the output 1 for recognized phonemes...
and 0 for unrecognized phonemes.

When, on the other hand, the phonemes are well separated, the GMM method often is as good as the NN method.

The differences described above is well in the tables 3.4 -3.6 illustrated by comparing the results for the phonemes \(s\) (viseme class 7) and \(n, l\) (viseme class 8) for the two different methods. Apparently \(s\) stands out so much that even a very simple classifier could handle it, while more complex methods are needed for, for example, \(n, l\).
Our Real Time Speech Driven Face Animation
Chapter 4

Implementation

The implementation is accomplished by integrating Matlab functions with C/C++ using Microsoft Visual C++ 6.0. The integration is possible by the use of Matlab Compiler, which can be used as a plug-in in Visual Studio and generates C++ code from m-functions. The functions can now be used in a similar manner as in Matlab, with the limitation that arrays and matrixes used by the functions must be of the type \textit{MwArray}.

The 38 Neural Networks are created and trained with the training database in Matlab. Their biases and weight matrixes are extracted and saved as Matlab files. These are loaded together with the Fisher matrix $W$ (also calculated in Matlab) from the C-program.

In order to play the sound we use source code from Microsoft’s DirectX 9.0 Software Development Kit (SDK). Since the calculations of the MFCCs require frames of 256 samples of raw sound data, the sound is segmented into that size. When a frame has been played, the played data is stored and calculations are made during the playback of the next frame. These calculations consists of MFCC extraction and simulation of the 38 Neural Networks. The outputs are added to the outputs from the previous frame. It is necessary that the calculation time does not exceed 16 ms, which is the time for the playback of a frame. Every fourth frame, the viseme class that has the largest sum of output values from the Neural Networks is presented on the screen.

The program is based on the Visage Technologies software [2] to which it adds a new feature. The feature is called SpeechToFace and its Graphical User Interface (GUI) is shown in Figure 4.1.
When the user have chosen a wav file to play, he must also choose between "Real Time Mode" or "Preprocessing Mode". If "Real Time Mode" is selected, the calculation of visemes are done as the sound plays, as described above. No overlapping between frames is done in this mode. If "Preprocessing Mode" is selected, calculation are made on the entire file before the sound starts playing. This means that the real time ability is lost, but more complex and heavy calculations can be done, since the computational time is unlimited. In this mode, 75 percent overlapping between frames are introduced, which results in more accurate recognition.
Chapter 5

Discussion

5.1 Results

The work resulted in a Windows application with an animated face model whose lips are synchronized with incoming speech. Our goal of achieving a time delay of less than 100 ms is met. Since $t_{comp} < 16 ms$, the total time delay will be $t_{delay} < 80ms$. Therefore, we consider the real time demand to be fulfilled. The visual impression also supports this, no noticeable time delay between the sound and the animations lip movements can be seen.

The recognition is also considered to be good. However, as can be seen in the confusion matrix in table 3.4, some phonemes are better recognized than others.

5.2 Limitations and Future Work

The current limitations of the work can be summarized with the following list:

1. The system supports sound input via wav files only.
2. The wav file should not contain any background noise.
3. The sample frequency is bound to 16 kHz
4. The system is trained to classify Swedish phonemes.
5. The phoneme database is small.
6. Reduced calculations due to limitations in the computational time.

The limitation of reading the sound from a wav file does not effect the real time aspect. If ”Real Time Mode” is selected, a streaming buffer is used and the computations on a frame are made when the frame has been played. The program
is consequently a simulation of a real time situation when someone speaks into a microphone. With a few adjustments, it should not be too hard to implement the ability to read directly from the microphone, or that the incoming speech builds up a streaming buffer, much like the situation we have now.

The sound that the user chooses for the SpeechToFace animation should be recorded in a noise free environment. The reason for this is that the training data for the Neural Networks is recorded in an isolated noise free speech lab. If a noisy wav file would be played, the recognition would probably not be so good. To improve the application, both when it comes to playing a wav file and speaking directly into a microphone, the sound should be filtered in order to get rid of interfering noise.

The recorded words used for extracting phonemes for the phoneme database are sampled with 16 kHz. In our program, the frame size is locked to 256 samples, which means that each frame contains 16 ms of speech. If the sample frequency would be higher, the frame would contain less information of the speech. Therefore, it would be harder to find the distinguishing characteristics of a certain phoneme and the number of classification errors would increase. This has been confirmed in a test where the sample frequency was 44.1 kHz, corresponding to about 6 ms speech per frame, with very poor classification results as a consequence. When working with prerecorded wav files, the problem could be solved by introducing variable frame lengths with respect to the number of samples, but constant with respect to the actual time (in this case 16 ms). Then, an arbitrary sampling frequency may be used.

The predefined MPEG-4 viseme parameters that we use are adapted to the English language. Therefore, when a phoneme is recognized and the corresponding viseme is picked, it can look as the viseme is incorrect although the recognition is correct. This is a natural consequence, since there are phonemes that exist in Swedish but not in English, and vice versa. When it was decided which phonemes would belong to the different viseme classes, we simply picked the ones that seemed to match the best.

Of course, the classification is not always correct, and some times wrong decisions are made. The phoneme database used as a training set for the Neural Networks is based on only 7 persons. If the database was expanded in order to get a larger and wider set of training data, the recognition would not be as sensitive to variations in accents and deviations as it is with a smaller training set. Also, the phonemes are manually cut out from the words, which is another source of error.

Since it is necessary that the computational time does not exceed the time it takes to play a frame of 256 samples, i.e 16 ms, the algorithms must be time efficient. The method of 75 percent overlapping between frames, i.e to reuse 256-64=192 samples together with 64 new values, did not work. In order to do this, it is necessary to calculate a phoneme every 64/16000=4ms, i.e the computational time has to be less than that. This could not be achieved with the computer the program is developed
5.2 Limitations and Future Work

on, a machine with 1.5 GHz clock frequency and 768 MB RAM. If Table 3.4 and 3.5 are compared, it is clear that the recognition would be better with overlapping. A phoneme decision is made every fourth frame, i.e. every $4 \times 256 = 1024$ samples. If overlapping is used, the decision is then based on $1024/64 = 16$ overlapped frames in comparison with 4 frames with no overlap. Hence, an error will not have the same impact when overlapping is used and it is no surprise that the recognition is better.

Another improvement to the program would be blending of visemes, which is supported by MPEG-4. Right now, the animation jumps from one viseme to another without considering if there is a big visual difference between the two visemes. If a blending factor is introduced, the transition between one viseme to another would be smoother. The program would then need to be able to remember the last shown viseme in order to know which viseme to blend with the new one.

The dimension of the MFCC vectors are reduced from 28 to 12 before the FLDT. It would be a better idea to let the FLDT handle the reduction of dimensions. This, because the algorithm finds and ranks the dimensions with the best separation. The Fisher matrix, $W$, would then be a $12 \times 28$ matrix instead of $12 \times 12$. The following multiplication with the 28 dimensional MFCCs would then result in 12 dimensional MFCCs.
Bibliography


Appendix A

The Visemes
Viseme 0

Viseme 1
Viseme 2

Viseme 4
Viseme 5

Viseme 6
Viseme 7

Viseme 8
Viseme 9

Viseme 10
Appendix B

User’s Guide to the Program
Step 1: This is the start up screen of the program. To initialize the animation, press the "open" button marked with the white arrow.

Step 2: Choose the .wra model that is to be used as the facial animation. Then repeat step 1 and choose a .fba file.
**Step3:** Press the button "open sound".

**Step4:** Choose the .wav file you wish to use.
Step 5: Choose "Real Time mode" or "Preprocessing mode".

Step 6: Press "play sound" to start the Speech To Face animation. If you wish to stop the animation, press "stop sound".
The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© [Andreas Axelsson and Erik Björhåll]