ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULITY OF INFORMATICS

DIPHONE BASED TEXT-TO-SPEECH SYNTHESIS SYSTEM FOR TIGRIGNA LANGUAGE

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT
FOR
THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

TESFAY YIHDEGO

JUNE 2004

DEDICATION

TO MY SON ABEL

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Eneyew Adugna for his guidance and encouragement to complete my thesis. Also I would like to thank Henock Luelseged for his help during the implementation phase.

I thank my caring wife Asqual Hailemariam for her support and patience during my thesis work.

I am also indebted to Ato Mesfine Fantaye for editing my draft thesis work.

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

The field of Text-To-Speech (TTS) synthesis is one in which much development and research has taken place over the last few decades. As a result of advances made, many laboratory and commercial systems of high quality exist today.

This thesis is an attempt made to develop a prototype TTS system for the Tigrigna language. It is based on the concatenation of diphones using TD-PSOLA (Time-Domain Pitch Synchronous Overlap and Add) technique. I used my voice to record an inventory of speech from which diphone units were extracted.

The Tigrigna text to speech system has two major distinct parts, which are text processing followed by speech synthesis. Visual C++ programming language is used to develop an interface and to handle text processing and MATLAB programming language is used to handle the signal processing.

Two major activities were made while preparing the diphone database; careful selection of corpus words and extraction of diphones from these words. To record corpus words and then to extract diphones from these words the free software called Praat is used.

For testing this system; the Mean Opinion Score (MOS) testing method was adopted. And the average result computed was found to be 3.05, which is closer to scale level good i.e. 3. The system is a good start to producing realistic speech from text, but there are several areas that can be improved. Inclusions of acronym converter to the text processing module and prosody control are some of the things that need further research.

OVERVIEW

This thesis is divided into five chapters:

Chapter 1: provides the definition of Text-to-speech synthesis, modules of TTS synthesis, statement of the problem and justification of the study, objectives of the study, methods employed, and scope and limitation of the study.

Chapter 2: outlines how human speech is produced, and briefly discusses types of speech synthesis, and speech units used for concatenation purposes.

Chapter 3: outlines the structure of Tigrigna TTS system, phonology of Tigrigna, and the processes involved in text processing as well as speech synthesis in this system. And also it briefly discusses the TD-PSOLA algorithm.

Chapter 4: In this chapter the text processing algorithm and speech synthesis algorithm and their flow charts are discussed in detail. And also experimentation results will be presented and discussed.

Chapter 5: draws conclusion from the test results obtained from experimentation. And it forwards recommendation for further research on the area.

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

Speech is the primary means of communication between people. Speech synthesis, automatic generation of speech waveforms, has been under development for several decades. Recent progress in speech synthesis has produced synthesizers with very high intelligibility but the sound quality and naturalness still remain a major problem. However, the quality of present products has reached an adequate level for several applications, such as multimedia and telecommunications (Lemmetty, S. 1999).

A Text-To-Speech (TTS) synthesizer is a computer based system that should be able to read any text audibly, whether it was directly introduced in the computer by an operator or scanned and submitted to an Optical Character Recognition (OCR) system.

Converting texts into synthetic speech encompasses, on the one side, a natural language processing (NLP) module able to transform the input text into an appropriate intermediate representation (includes information on the phoneme to be produced, their duration, locations and duration of any pauses, and the fundamental frequency), and on the other side, a digital signal processing (DSP) part capable of turning this representation into an output signal.

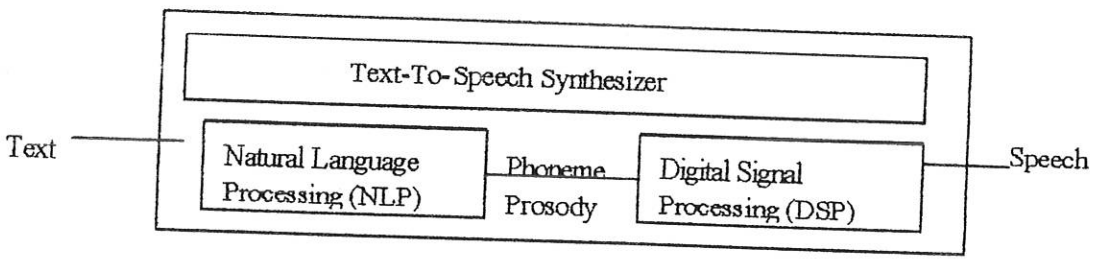


Figure 1.1 Text-To-Speech Synthesizer Modules

Figure1.1 introduces the functional diagram of a very general TTS synthesizer. It comprises a natural language processing module, which is capable of producing a phonetic transcription of the text read together with the desired intonation and rhythm (often termed as prosody), and a digital signal processing module that transforms the symbolic information it receives into speech.

1.1.1 NATURAL LANGUAGE PROCESSING MODULE

The natural language processing module of the synthesizer is used to perform text and linguistic analysis on the input text and can be broken to the following parts.

1.1.1.1 Text-to-Phoneme Conversion

In most text-to-speech systems; each input sentence is given as input to the text processing module of the system. The input is analyzed in such a way as to:

- Reformat everything encountered (e.g., abbreviations; acronyms) into words and punctuation
- Parse the sentence to establish the syntactic structure
- Find the semantically determined locations of contrastive stress
- Derive a phonemic representation from each word
- Assign a stress pattern to each word (Kaynar et al., 2001).

1.1.1.1.1 Text Formatting

There are some differences between how humans write and how they speak. Therefore the input text must be reconstructed from the written format into a format that is appropriate for the spoken. This process is called text formatting (normalization). This module identifies numbers, abbreviations, and

acronyms and transforms them into full alphabetic text when needed (e.g. $35.61, 35.61, 2000, the year 1971, 10:15 p.m.).

1.1.1.1.2 Letter-To-Phoneme Conversion

After the text has been properly normalized; the letter-to-phoneme module transforms this text into an intermediate linguistic representation and this representation has two components: the transformation of text into phonetic units and the conversion of text into prosodic parameters. The phonetic units specify what sounds to be produced while the prosodic parameters specify how they are to be produced (Kaynar et al., 2001).

Speech, which is produced by simple concatenation of segments, has good intelligibility but poor naturalness. Naturalness is an essential factor for user acceptance. It will be obtained by considering so-called prosodic parameters (Vosnidis, C., 2001).

- Phonemes are the smallest units of sound that help to represent semantics of a language systematically and unambiguously (Rodman, R., 2003). Changes of these units in a word or morpheme can bring a difference in meaning. Therefore to treat sounds that can bring meaning difference, the text has to be converted to these units of sound.
- The term prosody refers to the ensemble of properties of speech utterances that cannot be derived in a straightforward fashion from the identity of the phonemes constituting the speech utterances. Prosody comprises the melody of the speech, word and phrase boundaries, word stress, sentence accent, and changes in speaking rate, etc (Vosnidis, C., 2001).

1.1.1.1.3 Syntactic Analysis

Some pronunciation ambiguities can be resolved from syntactic information. And he only way to pronounce words correctly is to figure out the syntactic structure of an input sentence. Thus it would be highly desirable to include a parser in a text-to-speech system (Kaynar et al., 2001).

1.1.1.1.4 Semantic Analysis

Semantics is a subfield of linguistics that is traditionally defined as the study of meaning. Semantics deals with sense and reference, truth conditions and discourse analysis. Pragmatics is often considered a part of semantics (Webster-online-dictionary). Semantic and pragmatic knowledge is needed to disambiguate sentences. Contrastive stress may be applied to an important word depending on the meaning (Kaynar et al., 2001).

1.1.2 DIGITAL SIGNAL PROCESSING MODULE

Once the text has been transformed into phonemes and their associated durations and a fundamental frequency have been computed, the system is ready to compute the speech parameters for synthesis. Intuitively, the operations in the DSP module are the computer analogy of dynamically controlling the articulatory muscles and the vibratory frequency of the vocal cords so that the output signal matches the input requirements. In order to do it properly, the DSP module should obviously, in some way, take articulatory constraints into account, since it has been known for a long time that phonetic transitions are more important than stable states for understanding of speech. This, in turn, can be basically achieved in two ways (Vosnidis, C., 2001).

- Explicitly, in the form of a series of rules which formally describe the influence of phonemes on one another.
- Implicitly, by storing examples of phonetic transitions and co-articulations into speech segment database, and using them as they are, as ultimate acoustic units.

Two main classes of TTS systems have emerged from these alternatives to speech synthesis: concatenative synthesis and rule-based synthesis, the former relies heavily on the successful choice of concatenative units. Concatenative synthesis consists of concatenating segmental units (diphones, phonemes, etc); rule-based synthesis consists of the computation of control parameters based on pre-established rules (Vosnidis, C., 2001).

1.1.2.1 Rule-Based Synthesis

Rule-based approaches are memory efficient, since they eliminate the need to store speech segments, and they also make it easier in principle to implement new speaker characteristics for different voices than concatenative synthesis (Vosnidis, C., 2001).

These systems are restrictive regarding the choice of the parametric representation of speech, since such schemes rely both on our understanding of the relation between the parameters and the acoustic signals they represent, and on our ability to compute the dynamics of the parameters as they move from one sound to another. Thus far only formant-synthesizers and articulatory-synthesizers have been used in rule-based systems (Vosnidis, C., 2001).The formant-synthesizers try to describe speech elements by parameters related to formant frequencies, bandwidths and voicing while articulatory-synthesizers try to imitate the physical human mouth, wherein each speech element is described by parameters of the actual human mouth's position and movement (Lemmetty, S. 1999).

Rule-based synthesizers, however remain potentially powerful approaches to speech synthesis. They allow studying speaker-dependent voice features so that switching from one synthetic voice to another can be achieved with the help of specialized rules in the database. Following the same idea, synthesis by rule seems to be a natural way of handling the articulatory aspects of changes in speaking styles (as opposed to their prosodic counterparts, which can be accounted for by concatenative-based synthesizers as well). No wonder it has been widely integrated in TTS system (MITalk and JRSU synthesizers for English) (Vosnidis, C., 2001).

1.1.2.2 Concatenative Synthesis

Concatenative synthesis is based on the concatenation (or stringing together) of segments of prerecorded speech. As opposed to rule-based ones, concatenative synthesizers posses a very limited knowledge of the data they handle; most of it is embedded in the segments to be chained up. Three design decisions are particularly important in this type of synthesis; choice of unit, storage of units, and concatenation method (Kaynar et al., 2001).

1.1.2.3 Comparison of Rule-based and Concatenative Synthesis

Rule-based synthesis allows considerable freedom; a high number of adjustable parameters make high quality speech output possible. But this freedom is also the biggest disadvantage of rule-based synthesis; setting the parameters and devising rule sets such that the resulting speech is both intelligible and natural is very difficult, because as yet, we know little about the mechanisms of speech production (Wolter, M., 1997).

On the other hand, a concatenative approach only needs a reliable acoustic analysis of the language's phonetics, a good concatenation algorithm, a patient speaker, and enough time for segmenting the

- **Aid for Better Communication**: Redundancy in communication is the key to better communication. Redundancy in this context is the transmission of the same or closely related information to the receiver or learner through two sensory channels (usually aural and visual channels). Therefore TTS systems give chance to making errors in communication become minimized (Klatt, 1987).

In view of its application, it is clear that TTS systems must be developed for every language that has a need for it. Hence the conductor of this research selected the topic TTS synthesizer system for Tigrigna language, for the following reasons.

- As stated in Girma's (2001) thesis, according to the office of Population and Housing Census Commission of Ethiopia (1999) there are about 3,371,808 Tigrigna speakers of whom 3,224,875 speak as a mother tongue and 146,933 as a second language. However, before Eritrea became an independent country (since 1993), the total number of Tigrigna speakers in Tigray and Eritrea was 4,068,789 as stated in the 1984 census. At present, Tigrigna is the second most widely spoken Semitic language next to Amharic.
- Literatures, books, newspapers, and magazines published in Tigrigna have been increasing over the years. At the moment, Tigrigna is the medium of instruction in the primary and junior secondary schools in Tigray based on the new Education and Training Policy of Ethiopia (MOE, 1994, sub-article 3.5.1). Since the introduction of Geez word processing, electronic documents published in Tigrigna have been produced for different purposes. CD-ROM's publication and Web-page development are also emerging.
- Lots of documents have been prepared with this language. So for Proof reading of these documents a speech synthesizer connected with word processor is a helpful aid. Many users

find it easier to detect grammatical and stylistic problems when listening than reading. Normal misspellings are also easier to detect.

- A TTS system for other languages in the Semitic family can be developed with less effort.

1.3 Objective of the Study

1.3.1 General Objective

The main objective of this research is to develop a prototype TTS system for the Tigrigna language based on the concatenation of diphones using TD-PSOLA (Time-Domain Pitch Synchronous Overlap and Add) technique.

1.3.2 Specific Objectives

The specific objectives of the study are to:

- extract sample diphones from corpus words for database preparation
- Review the various literatures on Natural Language Processing (NLP) and Digital Speech Processing (DSP) concepts and techniques relevant to TTS synthesis.
- Build a prototype TTS synthesis system for Tigrigna language
- Test the system on how it performs for a limited number of words and sentences.
- Draw useful conclusion and forward recommendation for further study.

1.4 Methods

The following methods have been employed while developing a prototype.

1.4.1 Review of related literature

A number of resources such as books, research reports, articles in journals, and other published and unpublished documents have been used in order:

- To understand the phonology of Tigrigna language;
- To examine and select types of speech synthesis, acoustic units to be used, and concatenation method;
- To identify appropriate tools required to develop a prototype;
- To understand the processes involved in text processing;
- To understand the role of TD-PSOLA algorithm in speech synthesis.

1.4.2 Development tools and techniques

Since connecting prerecorded natural utterances is the easier way to produce intelligible and natural sounding speech, the concatenation synthesis method using TD-PSOLA algorithm is used for this TTS system development.

For the text analysis part Visual C++ programming language is used, because of its simplicity to develop an interface using it and in handling strings. And for the speech synthesis part Matlab programming language is used to handle the signal processing.

To record the corpus data and extract the desired acoustic units a tool called Praat was used.

1.4.3 Testing technique

For the evaluation of the system a number of native speakers of the language were invited to listen to some of the possible words and sentences that can be synthesized by the system (since we have few recorded diphones) and were asked to give their opinion on the overall quality of the synthesized speech with respect to scales used in MOS (Mean Opinion Score). MOS is preferred among others (Modified Rhyme Test (MRT)), Diagnostic Rhyme Test (DRT)) because it is the most widely used and the simplest method to evaluate the overall speech quality. It is a five level scale from bad (1) to excellent (5) (Lemmetty, S. 1999).

1.4.4 Scope and Limitation of the study

Due to shortage of time, only a limited number of corpus words were recorded, hence a limited number of diphone speech data was stored in the diphone database. Furthermore the text input could only extend up to paragraph level. But the words within the sentences are assumed to be separated only by blanks; punctuation marks like slash, semicolons were not considered and these words could contain abbreviations, acronyms and the like. And also only my voice was used to record the sample corpus.

CHAPTER TWO

SPEECH PRODUCTION

2.1 Human Speech Production

Sound is generated in several ways and at several locations in the human vocal tract. The most common sound generation sources are the quasi-periodic vibration of the vocal cords and turbulent noise generated by the passage of air through a narrow constriction, usually in the oral cavity (Eker, B., 2002). Vocal organs can be seen in figure 2.1.

The main components of the speech production organs are the lungs, trachea (windpipe), larynx, pharyngeal cavity (throat), oral or buccal cavity (mouth), and nasal cavity (nose). The pharyngeal and oral cavities are usually grouped into one unit referred to as the vocal tract, and the nasal cavity is often called the nasal tract. Accordingly, the vocal tract begins at the output of the larynx (vocal cords, or glottis) and terminates at the input to the lips. The nasal tract begins at the velum and ends at the nostrils. When the velum (a trapdoor-like mechanism at the back of the oral cavity) is lowered, the nasal tract is acoustically coupled to the vocal tract to produce the nasal sounds of speech. The pharynx is the tube-like organ extending from the back of the mouth to the larynx (Eker, B., 2002).

Air enters the lungs via the normal breathing mechanism. As air is expelled from the lungs through the trachea, the tensed vocal cords within the larynx are caused to vibrate by the airflow. The air flow is chopped into quasi-periodic pulses, which are then modulated in frequency in passing through the throat, the oral cavity, and possibly nasal cavity. Depending on the positions of the various articulators (i.e., jaw, tongue, velum, lips, mouth), different sounds are produced.

The vocal tract is bounded by hard and soft tissue structures. These structures are either essentially immobile (such as the hard palate and teeth), or are movable. The movable structures associated with speech production are also referred to as articulators. The tongue, lips, jaw, and velum are the primary articulators; movement of these articulators appears to account for most of the variation in vocal tract shape associated with speaking. However, additional structures are capable of motion as well. For instance, the glottis can be moved up or down to shorten or lengthen the vocal tract (Eker, B., 2002).

When the vocal cords are tensed, the airflow causes them to vibrate, producing so-called voiced speech sounds. When the vocal cords are relaxed, in order to produce a sound, the airflow passes through a constriction in the vocal tract and thereby become turbulent, producing so-called unvoiced sounds.

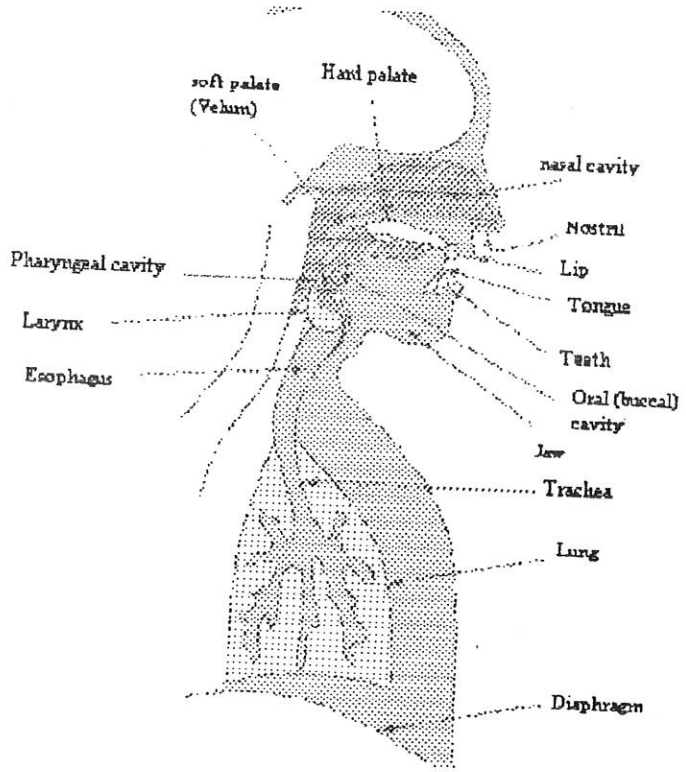


Figure 2.1 Vocal Organs (Eker, B., 2002).

2.2 Speech Synthesis Techniques

Synthesized speech can be produced by several methods (Lemmetty, S. 1999). Each of these methods has it own advantage and disadvantage. The methods are usually classified into three groups: Articulatory synthesis, Formant synthesis and concatenative synthesis. The first two are commonly referred as rule-based synthesis (Lemmetty, S., 1999).

2.2.1 Articulatory Synthesis

Articulatory synthesis tries to model the human vocal organs as perfectly as possible, so it is potentially the most satisfying method to produce high quality synthetic speech (Donovan, E., 1996).

On the other hand it is the most difficult method to implement and its computational load is also considerably higher than with the other common methods. Thus it has received less attention than other synthesis methods and has not achieved the same level of success (Donovan, E., 1996).

Articulatroy synthesis is quite rarely used in present systems but the synthesis methods are developing fast and the computational resources are increasing rapidly, it might be a potential synthesis method in the future (Lemmetty, S., 1999).

2.2.2 Formant Synthesis

Formant synthesis models pole frequencies of speech signal or transfer function of vocal tract based on source-filter model. Probably this is the most widely used synthesis method during the last decades (Lemmetty, S., 1999). There are two methods of combining the formants to make a model of the vocal tract. These are parallel and cascade, but for better performance some kind of combination is used (Donovan, E., 1996).

Formant synthesis provides infinite number of sounds, which makes it more flexible than for example concatentative synthesis. At least three formants are required to produce intelligible speech and up to five formants to produce high quality speech (Lemmetty, S., 1999).

Formant synthesis does not require a speech database; therefore they need less memory space. Also they are not dependent on speaker. These are the advantages of formant synthesis; however for modeling human speech some parameters should be extracted for the filter to be used in this model. Obtaining these parameters is a difficult task. Also these parameters should be used according to some formula and this requires some computational load in run-time. Although computational load of these systems are much less than articulatory synthesis systems, they are more than concatenative synthesis (Donovan, E., 1996).

2.2.3 Concatenative Synthesis

Concatenating prerecorded natural utterances is probably the easiest way to produce intelligible speech. However concatenative synthesizers are usually limited to one speaker and one voice and usually require more memory capacity than the other methods (Eker, B., 2002).

One of the most important aspects in concatenative synthesis is to find the correct unit length. The selection is usually a tradeoff between longer and shorter units. With longer units high naturalness and less concatenation points and good co-articulation are achieved but the amount of required units and memory is increased. With shorter units, less memory is needed but sample collecting and labeling procedures become more complex (Eker, B., 2002).

In the present systems units used are usually words, syllables, demisyllables, phonemes, and diphones, and sometimes triphones.

2.2.3.1 Word Concatenation

A word concatenation is perhaps the most natural unit for written text and some messaging systems with limited vocabulary (such as airline reservation, weather forecast reports). The concatenation of words is relatively easy to perform and co-articulation effects within a word are captured in the stored units (Vosnidis, C., 2001).

If the words are recorded separately, intonation will be lost. Moreover the system will be limited with the prerecorded words and this makes word concatenation unsuitable unit for unrestricted TTS systems. In addition since words start from zero level and end at zero level, concatenating without further processing produces a sentence that in every word you stop for a while. If words are taken from a sentence and there is an intonation in the sentence, that will also affect the system. However these systems require little computation in run time, since they simply concatenate prerecorded words. One other advantage of these systems is that quality within the output word is almost perfect since they are prerecorded, not created in run-time (Vosnidis, C., 2001).

2.2.3.2 Syllable Concatenation

Syllable units are smaller than words and larger than phonemes. The number of different syllables in each language is considerably smaller than the number of words, but the size of the database is still larger for TTS system. But unlike words; co-articulation between syllable units may not be so weak, and as a result smoothening across unit boundaries will not be as such easy (Donovan, E., 1996).

2.2.3.3 Phoneme Concatenation

Phonemes are the normal acoustic presentation of speech. However to obtain accurately the phonemes from input speech is difficult; since the start and the end of a phoneme in speech signal can't be determined certainly. So a lot of trial may be needed to get the final phoneme set. After obtaining the phoneme set, these phonemes should be concatenated smoothly however this is also a difficult task. On the other hand, these systems do not require much memory since few speech parts are prerecorded. For example for Tigrigna only 36 phoneme units are to be extracted from appropriate corpus data (Eker, B., 2002).

2.2.3.4 Diphone Concatenation

A diphone is roughly the last half of one phoneme followed by the first half of the next phoneme. So they contain the transition between adjacent phones. That means the concatenation point will be in the most steady state regions of the signal, which reduces the distortion from concatenation points (Traber, C., 2002).

In principle the number of diphones is the square of phonemes plus allophones, but not all combinations of phonemes are needed. For example there are no vowel-vowel phoneme combinations in Tigrigna. Diphone is a very suitable unit for sample based TTS synthesis as compared to others. Therefore this speech unit is used for concatenation purpose in this thesis.

Building the unit inventories consists of two main phases. First the natural speech must be recorded so that all units within all possible contexts (allophones) are included. After that, the units must be labeled and segmented. Gathering the sample units from natural speech is usually time consuming.

There are several problems in concatenative synthesis compared to other methods.

1. Discontinuities at concatenation points, which can be reduced using diphones or some special method for smoothing signal.
2. Memory requirements are usually very high especially with long concatenation units (such as words or syllables).
3. Data collecting and labeling of speech is usually time consuming. In theory all possible allophones should be included in the material, but tradeoff between quality and number of samples must be made.

Some of the problems may be solved with sophisticated DSP algorithms; for example algorithms that can reduce mismatch during concatenation, algorithms that can automatically label speech units from corpus data and the like (Donovan, E., 1996).

2.2.4 Linear predictive synthesis

Linear predictive (LP) synthesis is a source filter method of speech synthesis. The digital filter is estimated automatically from a frame of natural speech. Using a computationally efficient algorithm, LP has been used extensively in concatenation systems, since it enables the rapid coding of concatenation units. It is not really suited to rule-based systems since rules are more easily expressed in terms formants, and the relationship between the coefficients used to define the LP filter and the formants is not a simple one (Donovan, E., 1996).

The basis of linear prediction theory is the assumption that current speech sample y(n) can be estimated as a linear combination of the previous P samples , plus a small error term e(n). Thus, $e(n)=\sum_{i=0}^{P} a(i)y(n-i)$ where a(0)=1 and the a(i) are termed as the linear predictive coefficients , and P the linear predictive order. LP coefficients, a(i), are found by minimizing the sum of the squared

errors over the frame of speech under analysis. Two methods of performing this calculation are commonly used, termed the covariance method and the autocorrelation method, which differ in the range of over which the error is minimized (Donovan, E., 1996).

Synthetic speech produced using linear prediction synthesis is far from perfect. Klatt reports that autocorrelation of the LP synthesis does not produce formant frequencies and bandwidth correctly when speech is re-synthesized at different fundamental frequency to which it had originally. Even when re-synthesizing speech at he original pitch, the speech quality is considerably degraded compared to the original. The most noticeable result is that the synthetic speech is produced with characteristics of buzz sound (Klatt, D., 1987).

2.2.5 PSOLA synthesis

The Pitch Synchronous Overlap and Add (PSOLA) algorithm was developed by France Telecom at CNET. The technique does not synthesize speech itself, but merely enables prerecorded segments of speech to be smoothly concatenated while enabling the pitch and duration of the segments to be altered. It is therefore of use in concatenation synthesis in place of linear prediction, which was traditionally used to perform this role (Lemmetty, S., 1999). The advantage of PSOLA synthesis over LP synthesis is that the synthetic speech produced is of much higher quality (Donovan, E., 1996).

Time-Domain (TD)-PSOLA is the simplest, computationally efficient and the most widely used version of PSOLA (Lemmetty, S., 1999). The disadvantage of this method, compared to linear predictive synthesis is that it can't perform spectral smoothening at concatenation points. Consequently, one must choose synthesis unit very carefully to avoid formant discontinuities during

synthesis. The Frequency-Domain (FD)-PSOLA operates in the frequency domain and it can overcome this problem (Donovan, E., 1996).

All versions of PSOLA require a large waveform unit database for concatenation, but recent expansion of computer storage made this problem less critical. And also these work in essentially the same way (Lemmetty, S., 1999). A natural speech segment is broken into many short-term (ST) signals by Hanning windowing pitch-synchronously through regions of voiced speech and at a fixed interval through regions of unvoiced speech. The ST-signals are then recombined to produce the synthetic speech. The PSOLA algorithm is discussed in detail in section 3.5.3.1 (Traber, C., 2002).

CHAPTER THREE

PHONOLOGY OF TIGRIGNA AND TTS ALGORITHMS

3.1 The Phonology of Tigrigna

Tigrigna is one of the North-Ethio-Semitic languages namely Geez, Tigre and Tigrigna. It is mainly spoken in Tigray and Eritrea. As the name indicates the language is named after its home, the Tigray administrative region (Germay, B., 1983).

Phonology is the study of how speech sounds are organized in a language. The phonetic alphabet is usually divided in two main categories, vowels and consonants. Vowels are always voiced sounds and they are produced with vocal cords in vibration, while consonants may be either voiced or unvoiced. Vowels have considerably higher amplitude than consonants and they are more stable and easy to analyze and describe acoustically. Because consonants involve very rapid changes they are comparatively difficult to synthesize properly (Lemmetty, S., 1999).

3.1.1 Consonant Phonemes

Consonants are created when the airflow is directly restricted, or obstructed, so that air cannot escape without creating friction that can be heard (Rodman, R., 2003). There are twenty-nine consonant phonemes in Tigrigna. All consonants can be geminated except the pharyngeal and laryngeals (Germay, B., 1983). The term geminate consonant refers to a doubled or long consonant; for example the 'd' in the word gîddâf (ግደፍ) is a geminate consonant (Webster's Online Dictionary). Consonants are classified according to voicing, places of articulation, and manners of articulation (Olsen, S., 2003).

- **Voicing**: refers to whether the vocal cords vibrate or not. The level of vibration of the vocal cords determines whether a sound is voiced or unvoiced. If the vocal cords are apart, then air can escape unrestricted. Sounds produced in this way are said to be unvoiced. However, if the vocal cords are very close together, the air will blow them apart as it forces its way through. This makes the cords vibrate, producing a voiced sound. The sound [s (ሰ)] is called unvoiced because there is no vibration, and the sound [z (ዘ)] is called voiced because the vocal cords do vibrate.

- **Manner of Articulation**: The source of speech sounds is the air passing from the lungs through the larynx and into the cavities in the oral tract. These cavities can be modified in shape by the different positions of the articulators (the tongue, lips, velum etc.). The way the shape of the cavity changes influences the way the air in it vibrates giving rise to different sounds (Olsen, S., 2003). Tigrigna consonants may be classified by the manner of articulation as stops, fricatives, affricatives, nasals, liquids, and semi-vowels.

  - **The Stops**: are produced by a complete closure blocking the air momentarily and then releasing it abruptly (Olsen, S., 2003). Stops are also called Plosives. There are ten stop phonemes out of which three [b (በ) , d (ደ), g (ገ)] are voiced and the rest [p (ፐ), t (ተ), k (ከ), P (ጰ),T (ጠ),q (ቀ),ʔ (ዐ)] are unvoiced. [P (ጰ), T (ጠ), q (ቀ)] are the glottalized counterparts of [p (ፐ), t (ተ), k (ከ)].

  - **The Fricatives**: is made by an incomplete closure (or stricture) which produces friction as the air is forced through it (Olsen, S., 2003). Fricatives are also termed continuants. There are nine fricative phonemes out of which three

[z (ዘ), Z (ዠ), ʻe (ዐ)] are voiced and the other six [f (ፈ), s (ሰ), ʻS (ሠ), x (ኸ), h (ሀ), H (ሐ)] are unvoiced.

- **The Affricates** combines a stop with a following continuant but lasts only as long as a single fricative (Olsen, S., 2003). There are three affricate phonemes [J (ጀ), c (ቸ), C (ጨ)] which are voiced, unvoiced and ejective respectively. The (C) is the glottalized counter part of (c).

- **The Nasals**: is formed when air escapes through the nose. For this to happen, the soft palate is lowered to allow air to pass it, whilst a closure is made in the oral cavity to stop air escaping through the mouth (Olsen, S., 2003). There are three nasal phonemes [m (መ), n (ነ), N (ኘ)].

- **The Liquids:** is formed by partial blockage or obstruction of air in the mouth. Liquids consist of lateral [l (ለ)] sounds and flap [r (ረ)] sounds (Olsen, S., 2003). Both are voiced and dental phonemes and they occur in all positions.

- **The Semi Vowels**: are vowel-like consonants. They don't involve a blockage or obstruction of air in the mouth as is the case with stops and fricatives (Olsen, S., 2003).There are two semi-vowels [w (ወ), y (የ)] in Tigrigna and they occur in all positions.

➢ **Place of Articulation** The air stream used in producing speech sounds comes from the lungs and passes through the larynx and the vocal cords. Some of the components of the sound so produced are filtered out by certain configurations of the vocal tract, while other components are simultaneously amplified by them (Olsen, S., 2003). Tigrigna consonants may be classified

by the place of articulation as bilabial (lips together), labiodentals (lower lip against front teeth), dentals (teeth together), palatals (tongue on hard palate), velars (tongue near velum), and glottal (space between vocal folds).

All the twenty nine consonant phonemes of Tigrigna, which have been discussed above, are presented here in tabular form (Germay, B., 1983).

Manner of articulation \ Place of articulation		Bilabial	Labio-dental	Dental	Palatal	velar	pharyngeal	laryngeal
Stops	unvoiced	p (ፐ)		t (ፕ)		k (ክ)		ʔ (ዕ)
	voiced	B (ቢ)		d (ድ)		g (ግ)		
	ejective	P (ጵ)		T (ጥ)		K (ቅ)		
Fricatives	unvoiced		f (ፈ)	s (ስ)			ሐ (ሐ)	h (ሀ)
	voiced			z (ዝ)			ዐ (ዐ)	
	ejective			S (ፅ)				
Affricates	unvoiced			C (ች)				
	voiced			j (ጅ)				
	ejective			J (ጅ)				
Nasals		m (ም)		n (ን)	N (ኝ)			
Lateral				l (ል)				
Flap				r (ር)				
Semi-vowels		W (ው)				Y (ይ)		

Table 3.1 The consonant Phonemes

3.1.2 Vowel Phonemes

Vowels differ from consonants in that there is no noticeable obstruction in the vocal tract during their production. Air escapes in a relatively unrestricted way through the mouth and/or nose (Olsen, S., 2003).There are seven vowel phonemes in Tigrigna. These vowels can be divided into different categories depending on how they are formulated: Front, central or back positions of the tongue,

wideness/roundness of the constriction position, and place of the tongue (high, mid or low) (Rodman, R., 2003). Tigrigna vowels and their categorization are summarized below in table 3.2.

- **The Front Vowel:** these are two in number [ɨ (ኢ),e (ኤ)] both are unrounded. The mid front vowel (e) doesn't occur word finally while the high front (i) occurs medially as well as finally.
- **The Central Vowels:** there are three central vowels [ɨ̂ (ኧ),ə (ኧ),a (ኣ)] which are all unrounded. Among these (ɨ̂) never occurs word finally, where as the remaining two are found medially and finally.
- **The Back Vowels:** these are [u (ኡ), o (ኦ)]. Both are rounded and they occur in word medially and word final positions (Germay, B., 1983).

	Front	Central	Back
High	ɨ (ኢ)	ɨ̂ (ኧ)	u (ኡ)
Mid	ə̂ (ኧ)	e (ኤ)	o (ኦ)
Low		a (ኣ)	

Table 3.2 The Vowel Phoneme s

3.2 Intonation

The fundamental frequency or pitch is the frequency of voicing, that is, the frequency at which the vocal folds vibrate. The intonation of a sentence is the pattern of the pitch changes that occurs. In Tigrigna, three contrastive intonations are recognized.

- Falling intonation: the sentence final intonation has falling pitch
- Neutral intonation: when the sentence is simple sentence (declarative) the intonation is neutral
- Rising intonation: there is a relatively high rising intonation on the interrogative word or on the wh-questions (Germay, B., 1983).

Intonation variation can bring a difference in meaning. The statement ከይዱ (he left), for example, when spoken with falling intonation is a declarative, and a question when spoken with rising intonation, and with dynamically rising intonation interpreted as surprise.

ከይዱ.	Falling intonation	Declarative
ከይዱ?	Rising intonation	Question
ከይዱ!	Rising intonation	Surprise

Here are some more examples for each type of intonation discussed above

- (ናበይ ናብዝሸዶ ጨኒቁኒ.) Means 'I don't know where to go' →'Falling intonation'
- (ትማሊ ከይዱ.) tîmali kâyîdu. Means 'he went yesterday.' →'Neutral intonation'
- (መን ከይዱ?) mânîkâyîdu? Means 'who went?' →'Rising intonation'
- (ግደፍ!) gïdâf! Means 'don't do it!' →'Rising intonation'

3.3 The structure of Tigrigna TTS System

The efficiency of concatenative synthesizers to produce high quality speech mainly depends on the units chosen and algorithm used to concatenate the units smoothly. The units chosen should exhibit some of the following properties (Vosnidis, C., 2001).

I. They should account for as many co-articulation effects as possible

II. They should be easily concatenable

III. Their number and length should be kept as small as possible

On the other hand longer units decrease the density of concatenation points, therefore providing better speech quality. Similarly an obvious way of accounting for articulatory phenomena is to provide many variants of each phoneme. This is in contradiction with the limited memory constraint. Some tradeoff is necessary; diphones are often chosen. They are not too numerous and they

incorporate most transitions. They imply, however, a high density of concatenation points (one per phoneme), which reinforces the importance of efficient concatenation algorithm (Vosnidis, C., 2001). Therefore for Tigrigna TTS system diphones were chosen as basic speech units and TD-PSOLA algorithm was used to concatenate the appropriate diphone units smoothly.

The Tigrigna TTS system takes a written text as input from user. A sentence is read and is then passed to word separator. The word separator progresses word by word and words are assumed to be separated by blanks. When a word is obtained from the text it is passed to a unit that can process a word. This part separates the word into diphones; using diphone database where it gets speech data corresponding to diphone and finally concatenates the previously selected speech segments using PSOLA algorithm and produces sound. The general structure of the system can be seen in figure 3.1.

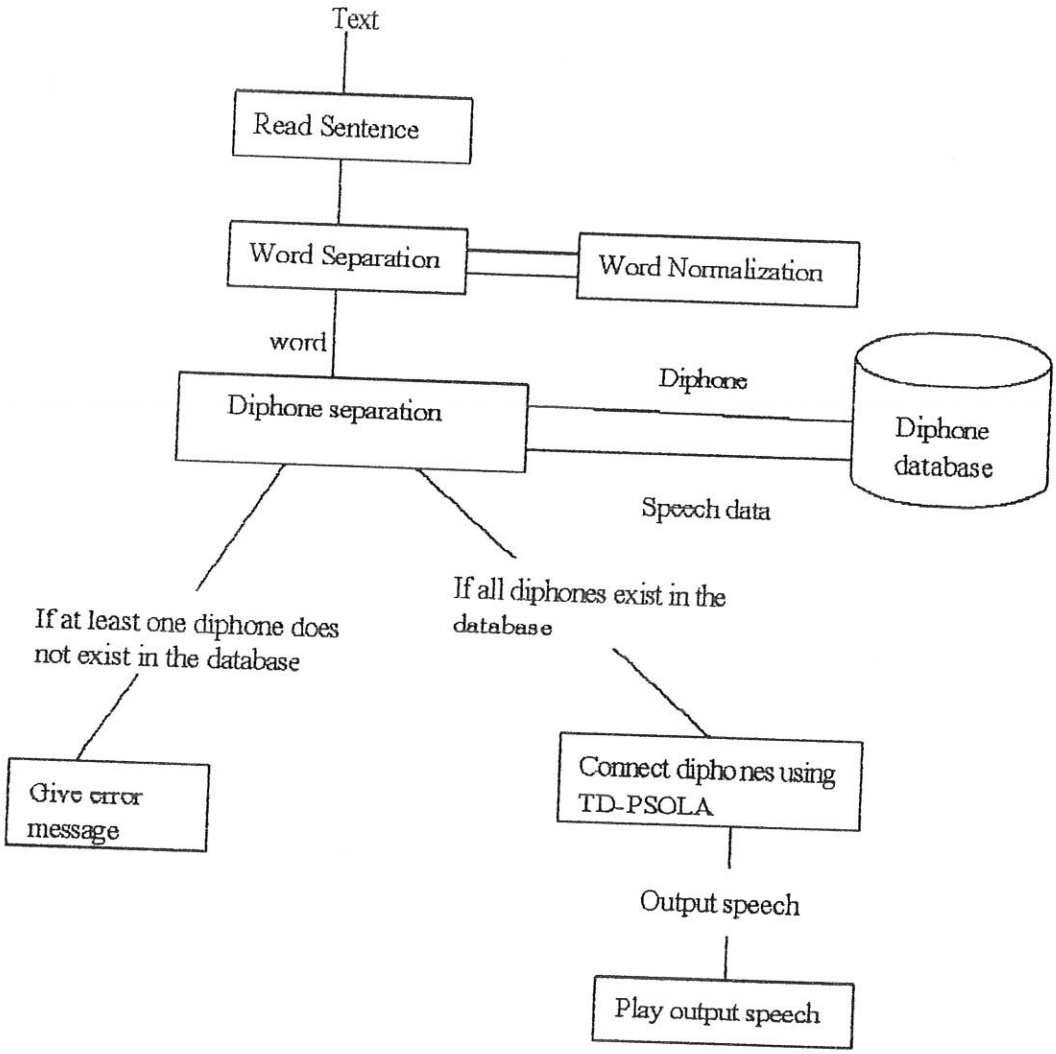


Figure 3.1 The General Structure of Tigrigna TTS System

3.4 Text processing

The text processing part in this system reads each line (sentence) one after the other until the whole text is read. The word separator separates the given sentence word by word and if a word contains an abbreviation or an acronym it is send to word normalization part where it gets expanded. And

further the words are segmented into units that the speech synthesis part can understand (i.e. Diphone units).

3.5 Speech synthesis

The speech synthesis part concatenates speech units (diphones) smoothly. PSOLA algorithm is used in concatenation of voiced parts while unvoiced parts are concatenated directly. In this system a sentence is read and played before the next line (sentence) is read.

3.5.1 Database preparation

Database preparation is one of the most vital parts in TTS systems that use a concatenation technique in speech synthesis part. The necessary speech units should be stored before the system starts running. In this system since diphones were used as basic concatenation units, they were stored in the diphone database. Data base preparation part can be divided into two parts; obtaining words to be recorded and extracting diphones from the recorded speech.

3.5.1.1 Obtaining corpus data

As mentioned before, to create diphone database, appropriate corpus data should be recorded and analyzed by sound editor in order to extract the corresponding diphone set. For better quality a careful selection of the corpus data is very important. In this system due to time limitation only up to 82 words were selected as sample corpus data and these are shown in appendix A.

3.5.1.2 Obtaining Diphones

There are two possible techniques that can be used for obtaining diphone units from recorded speech. These are automated and manual techniques. In the automated technique a text is read by a speaker and recorded. Then, the speech is separated into diphones by using some intelligent algorithms. Although this makes database preparation much easier, there are no efficient algorithms that are successful enough to be used in a TTS system. Hon et al stated that, the research on automating this process is still under construction (Eker, B., 2002). In this system 82 words were recorded using the PRAAT tool (Boersma, P. and Weenink, D., 2003), and about 139 diphones were extracted manually with the help of the visualization tools found in the PRAAT tool. And finally these were stored in the diphone database.

3.5.2 Diphone Concatenation

The technique used in the speech synthesis part of this system is diphone concatenation. Diphones are selected as basic speech segments because they minimize concatenation problems, as they include most of the transitions and co-articulations between phones, while requiring an affordable amount of memory. Moreover diphone concatenation is the easiest method to implement among others (Traber, C., 2002).

3.5.3 PSOLA Method

The PSOLA method, put simply, is a way of improving the quality of synthesized speech. There are several such methods LP-PSOLA (Linear Predictive PSOLA, MBR-PSOLA (Multi-Band Re-synthesis PSOLA) to list a few. The PSOLA can produce a speech of a high standard and is also relatively simple to implement (Lemmetty, S., 1999). It operates by providing a way of smoothly

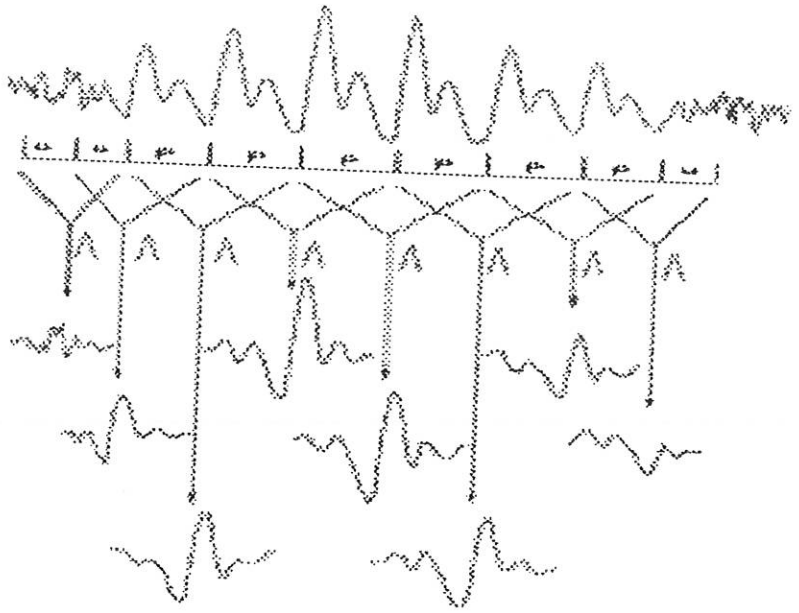


Figure 3.2: Schematic representation of the analysis of speech signals as a sequence of short-term signals by extracting double periods and multiplying them with window function. 'U' &'P' refer to unvoiced and voiced periods respectively (Traber, C., 2002).

The next four sections will describe how to obtain the pitch marks of a signal, how to identify the voiced and unvoiced part of a signal, how to generate fragments and finally how to concatenate these generated fragments. These are the major activities in speech analysis and synthesis in the TD-PSOLA algorithm.

3.5.3.1.1 Pitch-mark identification

Pitch-mark identification is perhaps the most complex part of PSOLA to implement. As stated previously, PSOLA smoothly concatenates speech segments, while at the same time adjusting their pitch to synthesize more realistic speech (White, S., 2003).

3.5.3.1.3 Fragment generation

As discussed in section 3.5.3.1.1, a voiced sound was assumed to have roughly constant pitch period. So a fragment is formed a round a pitch mark, extending one pitch period on each direction. Therefore each fragment has the size of twice the pitch period and every sample in the original segment has been lifted twice (in association with the two pitch-marks on either side of it). A fragment is then multiplied by Hanning window (a cosine wave such that the start and end are reduced to zero, but the amplitude climbs towards its original value as it tends towards the center of the fragment).

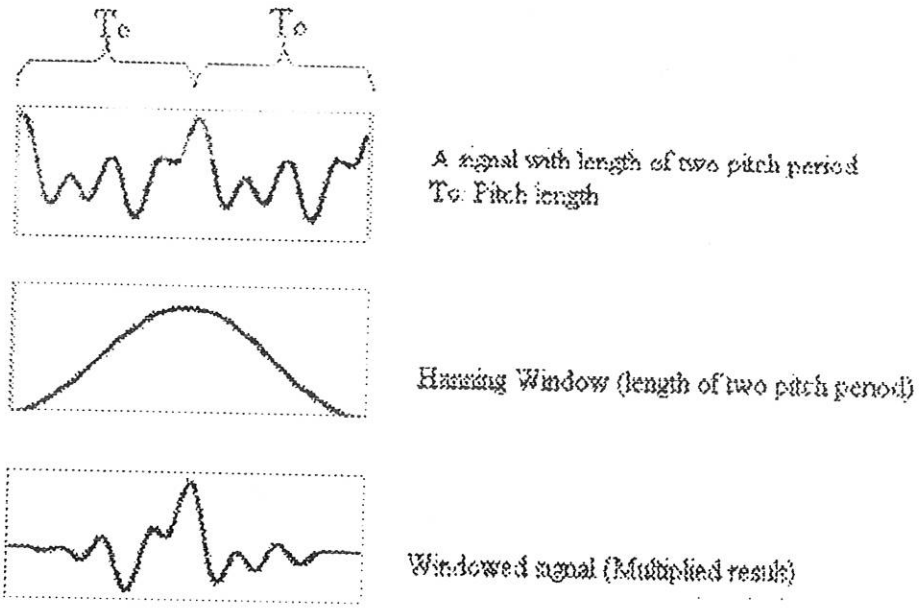


Figure 3.3 Multiplication of a signal by Hanning window

3.5.3.1.4 Fragment concatenation

Fragment concatenation involves the assembly of the final speech waveform from the fragments. This is the principal reason for Hanning window multiplication and also the reason why PSOLA has its

name. To create the final waveform, the fragments are merely added together at the point where they overlap. Here is where the voiced and unvoiced part of a signal treated differently.

- Unvoiced part of a speech segments are synthesized by direct concatenation.
- Voiced part of a speech segments are synthesized by overlapping the windowed signals with the proper spacing and adding them. (White, S., 2003).

Before speech segments are combined using overlap-add synthesis method; time and pitch scale modifications are performed as described below:

- **Time Scale Modification**: expanding the time scale of a signal causes compression in the frequency domain; so the output is a pitch scale compressed version of the original signal. On the other hand, when the time scale is compressed the pitch will be higher. The aim of time scale modification is to prevent these inherent modifications in the signal spectrum while modifying the time axis and obtain an output that has similar spectra as the original signal (Traber, C., 2002).

  TD-PSOLA modifies the temporal content by repeating or removing integer number of speech segments. Segment repetition produces a signal that is expanded in the time domain while the output using deletion is a time-compressed version of the original signal (Eker, B.,2002).

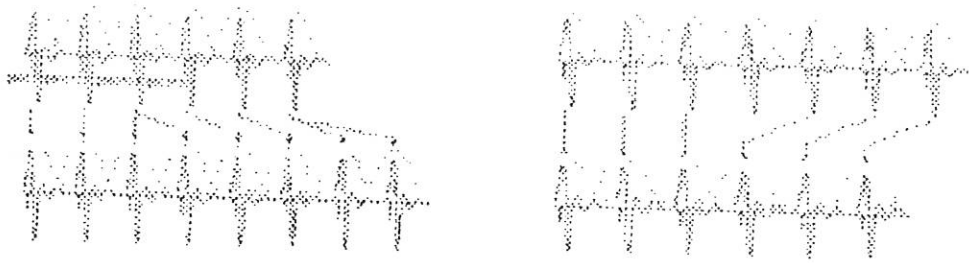


Figure 3.4 Time scale expansion (left) and Time scale compression (right)

- **Pitch Scale Modification** In this case, the aim is to modify the short-time spectral content of the signal without modifying its temporal characteristics. TD-PSOLA modifies the amount of overlap between successive pitch-synchronous Segments by changing pitch period (Traber, C., 2002). Since modification of pitch period means the modification of duration of speech; this should be compensated by time scale modification to avoid undesired result i.e. by deleting speech segments or by replicating speech segments (Eker, B., 2002).

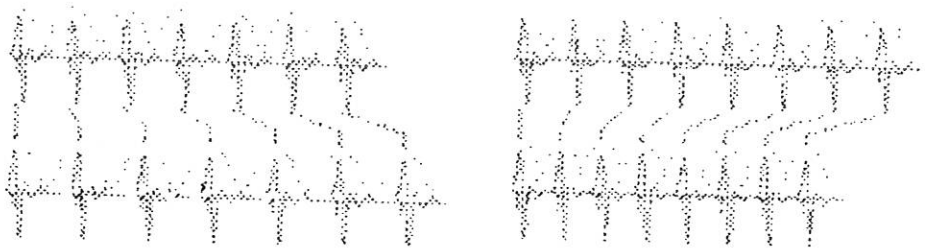


Figure 3.5 Pitch scale expansion (left) and Pitch scale compression (right)

- Pitch-marks are points in the speech signal, which correspond to the moments of principal excitation in the vocal tract. They are often identified visually when looking at the waveform since they usually appear at high peaks.
- Pitch-marks for a particular phone are usually roughly equally spaced, since the pitch of a phone is unlikely to change by any great amount during its utterance (White, S., 2003).
- Pitch-marks are not present in all speech. Unvoiced speech is composed of comparatively random noise. However, pitch-marks must still be assigned or the PSOLA method cannot deal with them. Since there is no pattern to this kind of speech, it is relatively arbitrary where pitch-marks should be placed and they are equally spaced (White, S., 2003).

3.5.3.1.2 Voiced/Unvoiced signal detection

Detection of the Voiced/Unvoiced is necessary because voiced signals are treated differently from unvoiced signals while modifying the pitch.

There are two approaches to detect the Voiced/Unvoiced parts of a signal (often referred as time domain parameters); the Root Mean Square amplitude (RMS) and Zero Crossing Rate (ZCR) (Cassidy, S., 2000).

- RMS is a measure of the energy in a speech signal. When applied to successive windows of a speech signal it gives a measure of the change in amplitude over time. RMS is high (i.e. greater than 0.25) in voiced signals while in unvoiced signals it is low (i.e. lower than 0.25).
- ZCR measures the number of times the signal crosses the zero line per unit of time. ZCR is useful in differentiating between voiced and unvoiced sounds since unvoiced sounds tend to have a large ZCR (i.e. greater than 0.75) while in voiced sounds it is smaller (i.e. smaller than 0.75) (Cassidy, S., 2000).

CHAPTER FOUR

IMPLEMENTATION AND EXPERIMENTATION

4.1 Database preparation

As mentioned in the section 3.5.1, database preparation is one of the most vital parts in TTS systems that use a concatenation technique in speech synthesis part. The necessary speech units should be stored before the system starts running. In this system since diphones were used as basic concatenation units, they were stored in the diphone database. Due to time constraint only up to 82 words were selected as corpus data, from which 139 diphones were extracted manually using the PRAAT tool. These selected corpus data and with their corresponding diphones, which were selected to construct the diphone database, are shown in the appendix A.

4.2 Implementation

The Tigrigna text to speech system has two major distinct parts, which are text processing followed by speech synthesis. For the first part, text processing, Visual C++ programming language was used, because of its simplicity to develop an interface using it and in handling strings. And for the second part, speech synthesis, MATLAB programming language was used to handle the signal processing. The algorithms used for this system will be briefly discussed in the following sections.

4.2.1 Text processing algorithm

Text processing is done very simply in this system. The system takes input text from user; then this text is processed one sentence at time. A sentence is read and it is passed to a word separator. Here the words

are assumed to be separated by blanks; slash and semicolon are not considered. Until end of line (sentence) is reached each word is processed one after the other. A diphone is then constructed from phonemes contained in a word. The result is then passed to speech synthesis module. The next line is read and it is passed to word separator and then to the diphone separator and finally the result is passed to speech synthesis module. This process continues iteratively until end of file is reached. Figure 4.1 and Figure 4.2 flow charts explain the text processing part. Here the underscore (_) sign is used to represent silence (short break) at the beginning and end of a word and backslash (/) is used to indicate end of file.

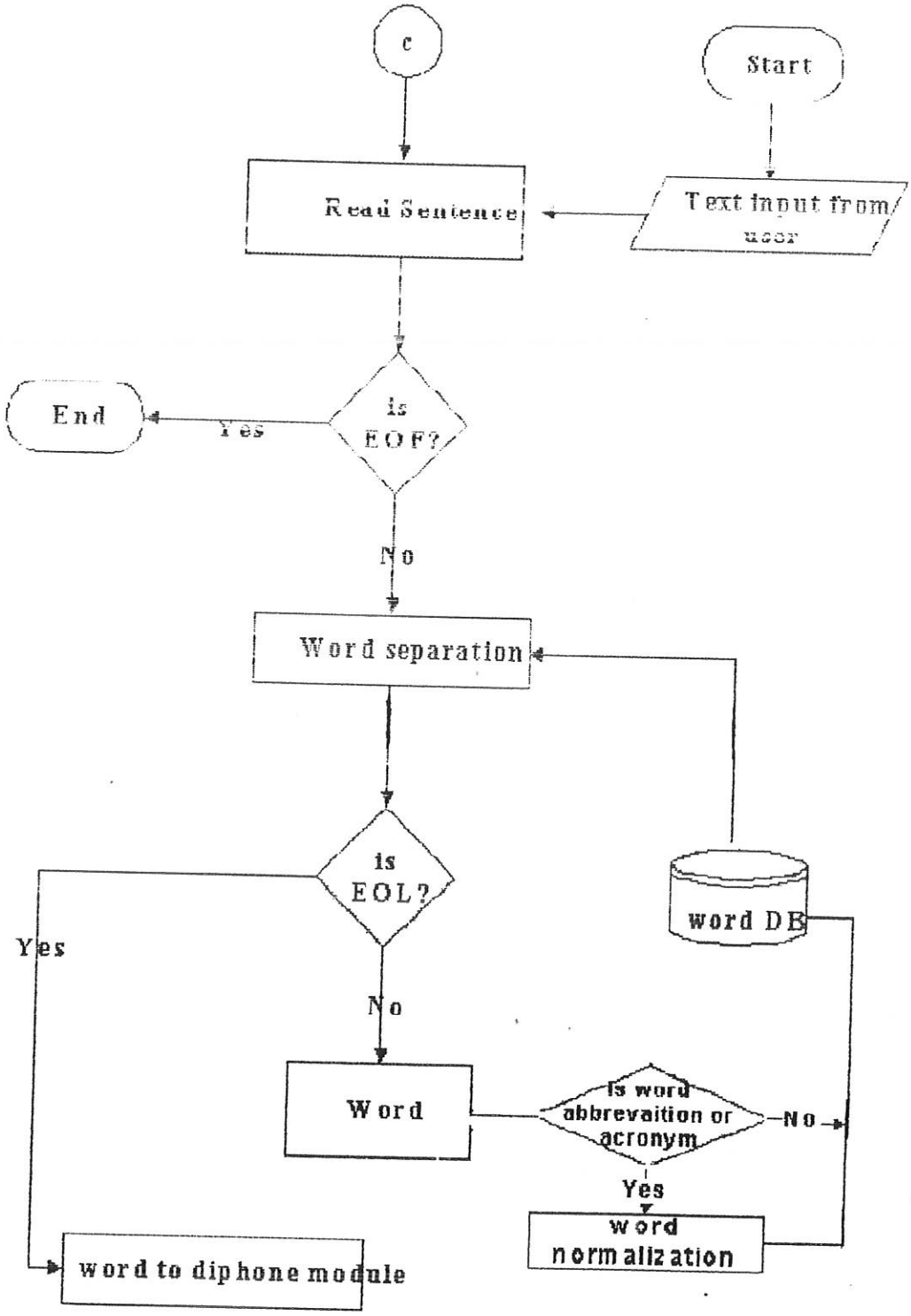


Figure 4.1: Flow chart for word separator module

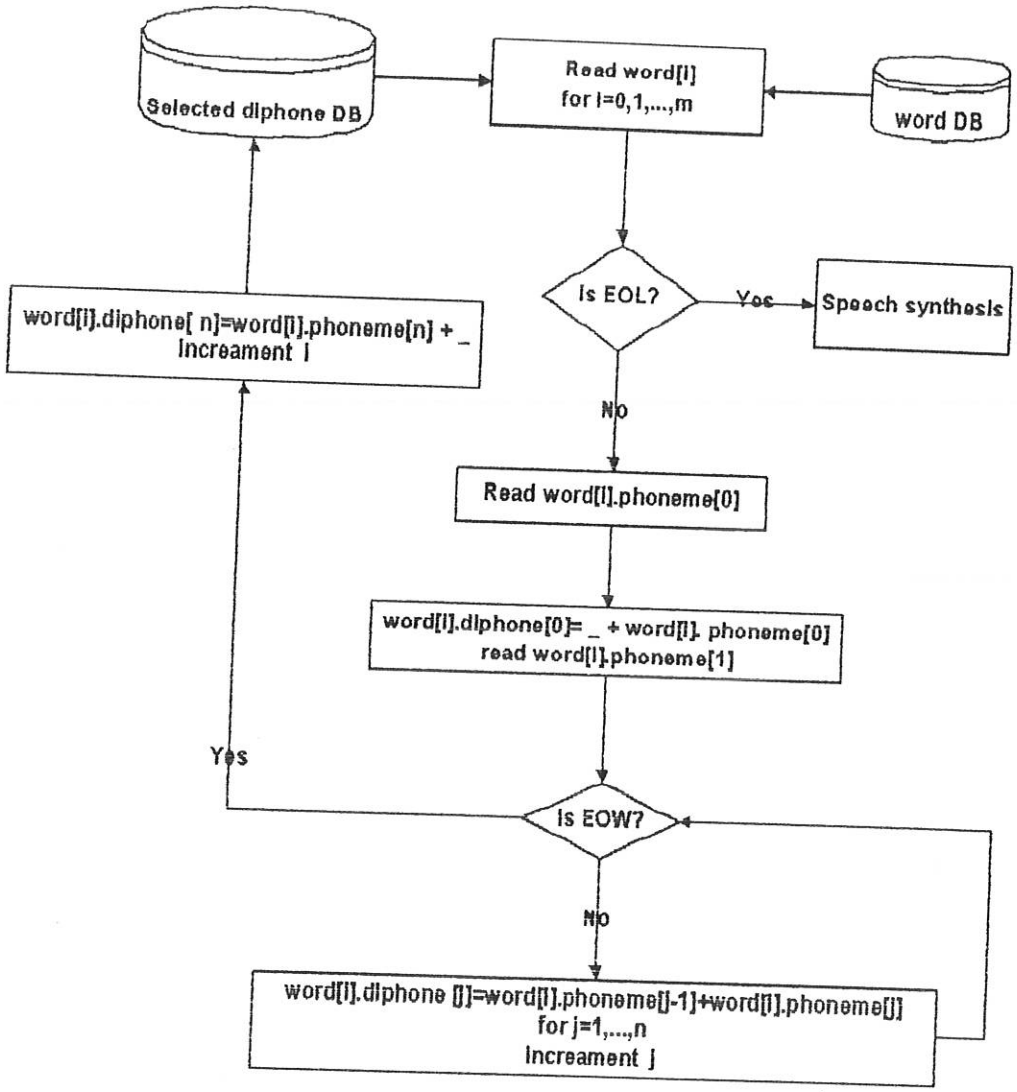


Figure 4.2 Flow chart for diphone separation module

Here is an example which shows how the text-processing module works to construct diphones.

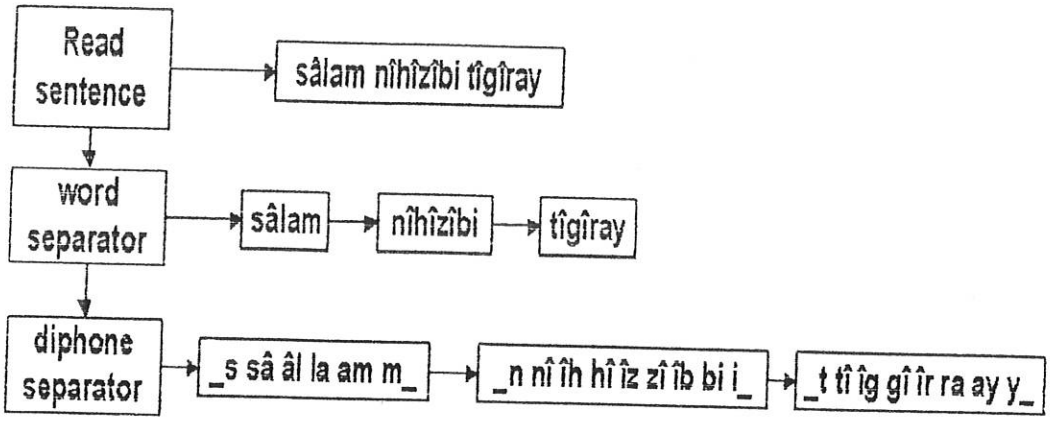


Figure 4.3 Example of text processing for diphone creation

4.2.2 Speech synthesis algorithm

Second part of the system deals mainly with concatenation of acoustical units. Here the text is not read until end of the file (EOF) in order to reduce the users waiting time and to use the memory efficiently. After decomposing input sentence in to diphones units, waveforms corresponding to these units are retrieved and concatenated. Concatenation is carried out using PSOLA for smooth concatenation. Finally the resulting waveform of the first sentence is played. The process continues iteratively until end of file is reached. In this module prosody generation is not considered.

In this module there are two alternatives while calculating the pitch marks and finding voiced/unvoiced parts of a speech. They can be calculated at run time for use or calculated offline and loaded into memory when the system is activated. Both of these approaches were implemented but for sake of simplicity the first approach was presented in the algorithm.

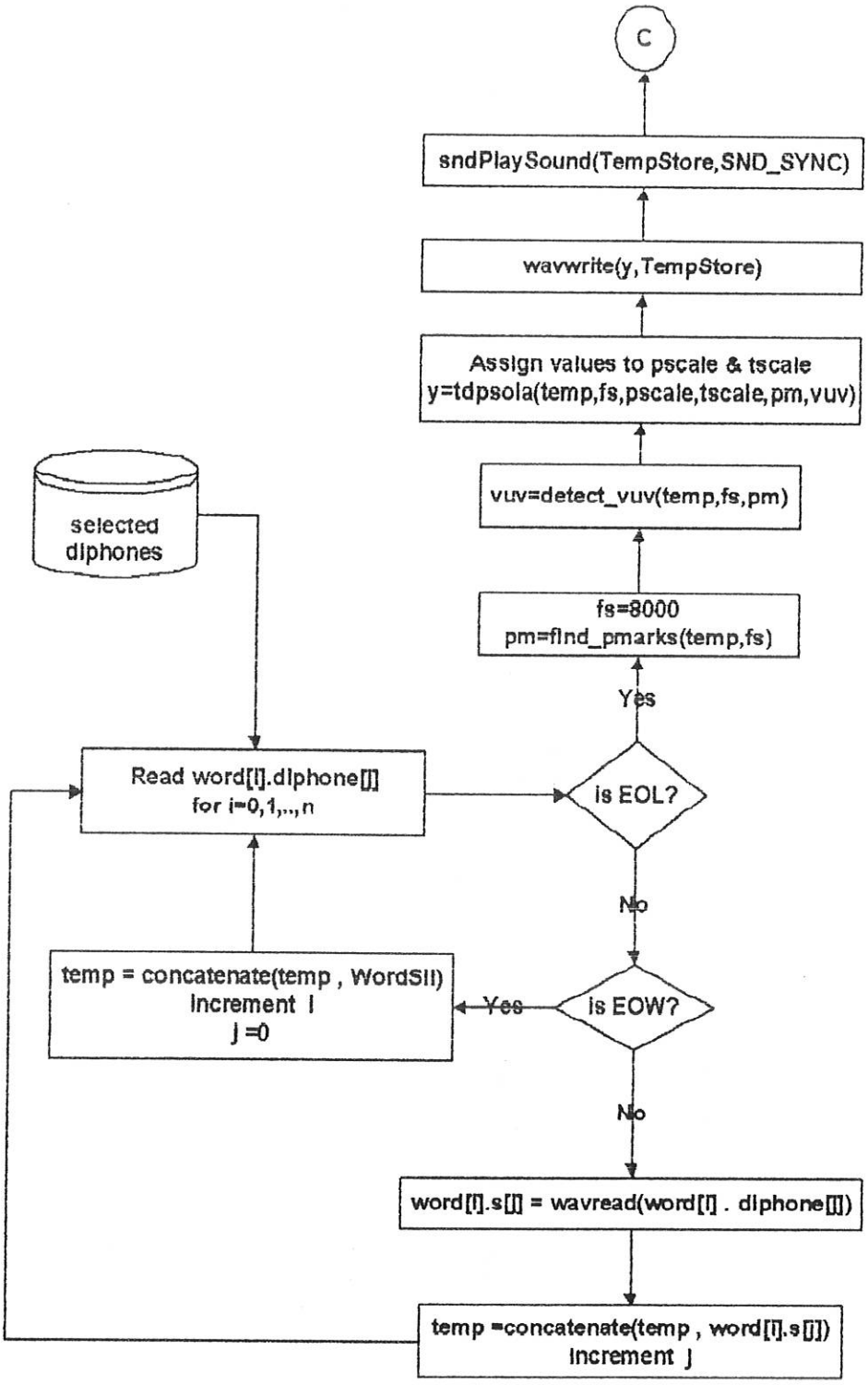


Figure 4.4 Flow chart for speech synthesis module

The following figure shows an interface developed for the Tigrigna TTS system prototype. A user writes Tigrigna text on the edit box using the keyboard and then submits the text to the system by pressing the speak button. Internally the text is decomposed into diphones by the text-processing module. The speech synthesis module will then play the sound if all the resulting diphone units are present in the database. Otherwise an error message will be displayed.

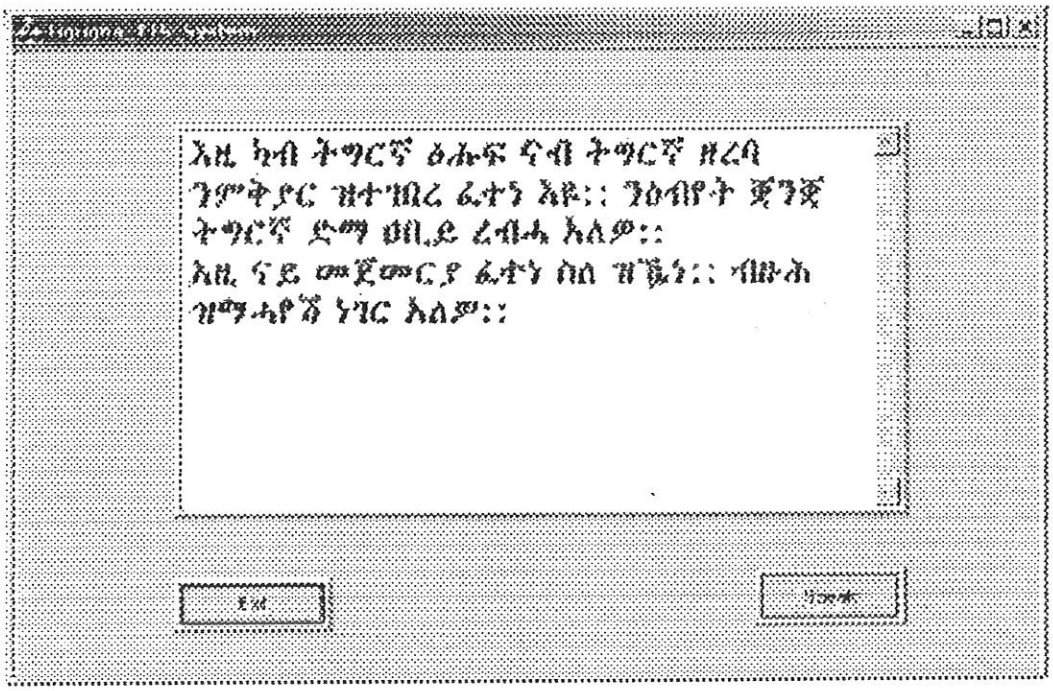


Figure 4.5: Tigrigna TTS System Interface for the Prototype.

4.3 Testing and Evaluation

Synthetic speech can be compared and evaluated with respect to intelligibility, naturalness, and suitability for used application. The evaluation can also be made at several levels, such as phoneme, word or sentence level, depending on what kind of information is needed. It is very difficult, almost impossible; to

say which test method provides the correct data. In a text-to-speech system not only the acoustic characteristics are important, but also text pre-processing and linguistic realization determine the final speech quality (Lemmetty, S., 1999).

Several individual test methods for synthetic speech have been developed during last decades. The most commonly used methods are Modified Rhyme Test (MRT), Diagnostic Rhyme Test (DRT), and Mean Opinion Score (MOS) to list a few. In this system the MOS test method was adopted (Lemmetty, S., 1999).

Mean Opinion Score is probably the most widely used and simplest method to evaluate speech quality in general. It is a five level scale from bad (1) to excellent (5). The listener's task is simply to evaluate the tested speech with scale described in Table 4.1 below (Lemmetty, S., 1999).

	MOS
5	Excellent
4	Very good
3	Good
2	Fair
1	Bad

Table 4.1 Scales used in MOS

To evaluate the performance of this system, six native speakers of Tigrigna language and three native speakers of Amharic language were invited. Among the six native speakers, three of them were asked to give their judgment on the Tigrigna test words and their test results were presented in table 4.2. The rest three were asked to give their judgment on the Tigrigna test sentences and their test results were presented in table 4.3. All the three native speakers of the Amharic language were asked to give their judgment on the Amharic test sentences and their test results were presented on the right most column of table 4.4.

	ቃል	Word	Person1	Person2	Person3
1	ትግራይ	tîgîray	3	4	3
2	ሰላም	sàlam	3	3	3
3	ንበብ	nîbab	3	4	4
4	ንበይነይ	nîbâyînây	3	3	3
5	መስተ	mâsîtâ	3	4	3
6	ፈነወ	fânâwâ	4	4	4
7	መንገዲ	mânîgâdi	2	3	3
8	ፋፋ	fafa	4	5	5
9	ስደደለይ	sîdâdâlây	3	4	4
10	ተረኛ	târâPa	2	2	2
11	ናበይ	nabây	3	4	3
12	ደንበር	dânîbâr	2	3	3
13	ብተበፋ	bîtâbîfa	3	3	3
14	ተበተነ	tâbâtânâ	2	3	3
15	ግራት	gîrat	3	4	3
16	ፈተወ	fâtâwâ	4	4	4
17	ገዛና	gâzana	4	4	4
18	ነባሪ	nâbari	3	3	3
19	ብትቶ	bîtîto	2	3	2
20	ሀዲግ	Yîdâg!	4	5	4
21	ፋንታ	fanîta	4	5	4
22	ፈታዊ	fâtawi	4	4	4
23	ነባራይ	nâbaray	3	3	3
24	ተመን	tâmân	2	3	3
25	ንህቢ	nîhîbi	3	4	4
26	ግንባር	gînîbar	2	2	2
27	ነገር	nâgâr	3	3	3
28	ቀረባ	qârâba	3	4	3
29	ስጋ	sîga	4	5	4
30	ሕኖ	Vîto	4	4	4
31	መዳለዊ	mâdalâwi	3	4	3
32	ትግርኛ	tîgîrîPa	2	2	2
33	ፈተነ	fâtânâ	3	4	4
34	ቀረበ	qârâbâ	3	3	3
35	ስፋይ	sîfay	3	4	3
36	ምግምጋም	nîgîmîgam	1	1	1
37	ዝቀረበ	zîqârâbâ	2	3	3

38	ገዳም	gâdam	4	4	3
39	መዳይ	mâday	3	3	3
40	ተዛረበ	tâzarâbâ	2	3	2
41	ወሰዶ	wâsâdo	3	3	3
42	ተሳትፎ	tâsatîfo	2	3	3
43	ዶክተር	dokûtâr	4	4	4
44	ጎዶሎ	godolo	3	4	4

Table 4.2 Test results of the Tigrigna words according to MOS test

ዐረፍተ ነገር	Sentences	P1	P2	P3
መደላዊ ተስፋይ ይህደን.	mâdalâwi tâsîfay yîhîdâgo	3	3	3
ብትግርኛ ዝቀረበ ንባብ.	bîtîgîrîPa zîqârâbâ nîbab	2	3	2
ንገምጋም ዝቀረበ ንባብ.	nîgâmîgam zîqârâbâ nîbab.	2	2	2
ርእይቶ ዶሎኩም?	rîyîto dolokum?	2	2	2
ሰላም ንህዝቢ ትግራይ!	sâlam nîhîzîbi tîgîray!	2	2	2
መንገዲ ትግራይ ቀረባ.	mânîgâdi tîgîray qârâba.	2	2	2
ብህዝቢ ትግራይ ዝቀርብ ፊነወ ትግርኛ.	bîhîzîbi tîgîray zîqârâb fânâwâ tîgîrîPa	2	2	2

Table 4.3 Test results of the Tigrigna sentences according to MOS test

Since the Praat tool doesn't discriminate between capital and small letters (case insensitive). Some changes were made. The letters V and P were used instead of H, N respectively to test the system.

ዐረፍተ ነገር	Sentence	Test result from Amharic TTS			Test Result from Tigrigna TTS		
		P1	P2	P3	P1	P2	P3
አበሰ በሶ በላ	abâbâ bâsa bâlla.	2	2	2	3	3	3
ወደቤት ተመለሰ	wâdâbet tâmâlâsâ.	2	3	3	2	3	2
አራት ኪሎ አካባቢ ደረሰ	arat killo akababi dârâsâ.	1	2	2	-	-	-
ነገ ይመለሳሉ?	nâgâ yîmâlâsalu ?	2	2	3	3	2	3
ሰባት ሰአት ሞላ?	sâbat sett mola.	2	2	2	1	1	1
አለሙ መኪና ነዳ	alâmu mâkina nâdda.	2	2	1	2	2	2
በሰላም ደረስክ?	bâsâlam dârâsîk ?	2	2	2	2	2	2

Table 4.4 Test results of the Amharic sentences according to MOS test

4.4 Discussion of the result

As seen from the test results, it was not easy to get high rate of intelligibility. Burst of sounds at the beginning and end of a word, elongation of concatenated utterances, noise that originate from recording and inexactness in extracting diphones while preparing diphones form sample corpus data were major problems that limit the intelligibility of synthesized utterances.

It was observed that words with few numbers of diphones were recognized easily when compared to words with more number of diphones. Also it was observed that the presence of geminate consonant in a word decreases the intelligibility of the word. Because a double consonant (eg. 'dd' as in the word addây (አደይ)) diphone is needed in the diphone database. One way of resolving the problem is that is associated with geminated consonants is to include corpus words, which contain the required geminate consonants and add these consonants to the diphone database.

It was observed that the system produces output in acceptable delay for short sentences by listeners, but it requires more time for long sentences. The part that consumes most of the time is the speech synthesis module in particular the PSOLA part.

Table 4.3 is a test data at a sentence level used for the Amharic language TTS system. The Amharic Text-To-Speech synthesis system was developed by Henock (Henock, L., 2003). This system was developed using Delphi for text processing and Matlab for speech synthesis. The sample sentences on this table were taken for testing to see if the same (closer result) could be obtained, because the Languages Tigrigna and Amharic are both Semitic. Furthermore both languages have the same alphabetic representation having the same sound except for few additional alphabets found in Tigrigna. On the top of these Amharic language is the official language of the government of Ethiopia.

The test result from Tigrigna TTS, at the right most column of Table 4.3, was obtained from the existing database in the Tigrigna TTS system. From the result; comparing the two columns, Test Result from Tigrigna TTS and Test Result from Amharic TTS; therefore a single system could be developed for these two languages possibly by preparing exception dictionary to manage their difference.

The average test result was found to be 3.05; which is closer to the scale level good i.e., 3. This result was encouraging and this in turn is an indication that diphone units are appropriate acoustic units for Tigrigna speech synthesis system from text input.

CHAPTER FIVE

CONCULSION

5.1 Conclusion

Text-To-Speech is a very promising technology, fulfilling a very real need in the rapidly growing communications market. Helping handicapped communicate, learning new language, and increasing productivity are just some of the benefits provided by text-to-speech synthesis.

The Tigrigna TTS system is a result of an attempt taken to understand the speech production and a realization to the possible benefits of the synthesized speech generation for Tigrigna. The thesis objectives were achieved to an acceptable performance in terms of understanding speech production in general, relationship to Tigrigna speech production, designing and implementing speech synthesizer prototype.

Minimizing noise during recording time, careful selection of corpus data, careful extraction of the required diphones for diphone database preparation, and a speaker whose voice is to be used to create the acoustical unit database are utmost important in improving the synthesized speech intelligibility.

This system can be used as a starting point to develop TTS systems for other languages in particular for Semitic languages with minor refinements on the linguistic analysis.

5.2 Recommendation

As this work is an initial attempt, there is a room for improvement

- Recording corpus data in sound laboratory has the advantage of avoiding any source of noise while recording the corpus data. So if possible, recording should be done in sound laboratory.
- The recording session should be one time (one session) if possible and the corpus data should be checked for constant energy. A great care should be taken while selecting words/phrases for the corpus data.
- The system is not concerned with numbers. So the incorporation of number converter for the word normalization module is important.
- Perceptually some sounds may be acoustically different in different places of a word, because phonemes are affected by their neighbor phonemes. Therefore there should be different diphones recorded for different parts of a word in order to have a quality speech. One way could be to divide into three: beginning, middle, and end diphone.
- To apply more experiment on the corpus data and hence to complete the diphone database.
- If possible to develop an intelligent algorithm that is able to separate a speech corpus data into diphone units automatically.
- This thesis work exposes a number of different disciplines such as speech production, phonology, and digital signal processing on the way reaching the objectives. The understanding of phonetic and phonological information for Tigrigna language gives better understanding of the relationship to letter to sound of the language.
- After obtaining a stable system for one voice, other voices could be added to the system.

- Having obtained an intelligible speech, it can be better to pay attention on naturalness. In this respect, a better text processor according to the need should be implemented and modules on intonation or prosody, etc should be developed.

REFERENCES

1. Boersma, P. and Weenink, D., (2003) "PRAAT: doing phonetics by computer" University of Amsterdam; Netherlands. http://www.fon.hum.uva.nl/praat/.
2. Cassidy, S., (2000) "Fundamentals of Speech Science Acoustics Module" Speech Hearing and Language Research Centre
   Department of Linguistics
3. Donovan, E.,(1996) "Trainable speech synthesis" Ph.D Dissertation , Cambridge University.
4. Dutoit T. (1996), "A Short Introduction to Text-to-Speech Synthesis". TTS research team, TCTS Lab., Mons, Belgium. http://tcts.fpms.ac.be/synthesis/introtts.html
5. Eker, B.,(2002). "Turkish Text-To-Speech system" M.Sc Thesis, Bilkent University
6. Ferencz A., Zaiu D., Ferencz M., Toderean G. (1989). "A Text-To-Speech System for the Romanian Language". http://www.racai.ro/books/awde/ferencz.html
7. Girma, B, (2001) "A Stemming Algorithm Development For Tigrigna Text Documents" M.Sc Thesis, AAU.
8. Girmay, B.,(1991) "Phonology of Tigrigna" B.Sc Thesis, Addis Ababa, AAU.
9. Henock, L.,(2003) "Concatenative Text-To-Speech (TTS) Synthesis for the Amharic Language" M.Sc Thesis, AAU.
10. Kaynar, I. and Gelgi, F., (2001) "Text-To-Speech Synthesis" project proposal report.
11. Klatt, D., (1987). "Review of Text-to-Speech Conversion for English". Washington, USA. http://www.mindspring.com/~dmaxey/ssshp/dk_737a.htm
12. Laine, B.,(1998) "Text-To-Speech System of The Amharic Language" M.Sc thesis ,Addis Ababa, AAU

ከይተነበበ	kâyItânâbâbâ	ân, nâ
ተበተነ	tâbâtânâ	â_
ዶቡብ	dobub	do, b_
ጎዶሎ	godolo	ol, _g
ዶክተር	dokitâr	it; kî
ሪኪኩም	rikum	ku, ri
ሓረስታይ	Harâsitay	_H
ርሑስ	rîHus	_r, s_
ርኢዩ	riyu	ri
ርእዩ	riyu	iy
ንእሽተይ	nixitây	y_
ሓፂባቶ	Ha'Sibato	to
ሰላም	sâlam	la ,sa
ዝካአለ	zikalâ	_z : ik
ዘለኩም	zâlâkum	ku; âk
ገዛኩም	gâzakum	um; ak
ገዳም	gâdam	äd, m_
ዝበሎ	zibâlo	âl ;lo
ውህብቶ	wihîbito	bî, w
ርእይቶ	riyito	ri
ዶቆም	doqom	_d, qo
ቀሪባቶ	qârîbato	_q
ይግደፎ	yigidâfo	dâ ,fo
ተሳትፎ	tâsatifo	tâ; at
መዳይ	mâday	mâ, y_
መዳናበሪ	mâdanabâri	da; an
ሓንጎል	Hanigol	go; l_
ጉልባብ	gulîbab	bî
ተፈነወ	tâfânâwâ	fâ, âf
ፋኑስ	fanus	_f ,fa
ግዜና	gizena	gi, en
ሓሳብ	Hasab	sa
ጋምባለ	gamîbalâ	ga
ካብዚ	kabîzi	ka, îz
ናቶም	natom	om, na
ሕራነ	HirîbâNa	Hî
ታኣዚዙ	ta'zizu	za, ta
ሓበነይ	Habânây	ây
አደይ	adây	_a ; ad
ድሙ	dîmu	_d, mu
ብክብሪ	bîkîbîri	îk ;kî
አለማሚድዎ	alâmamidîwo	al ; âm
ምስኪን	mîsîkin	ki ;in
በለስ	bâlâs	âl ; âs

APPENDIX B: VISUAL C++ SOURCE CODE

B-1: HEADER FILE OF THE IMPLEMENTATION OF THE PROTOTYPE

```
// Tigrigna_TTS_SystemDlg.h : header file

#if
!defined(AFX_TTSDRAFTDLG_H__DCA4D953_25DF_44D6_B836_8F1D740A86CF__INCLUDED_
)
#define AFX_TTSDRAFTDLG_H__DCA4D953_25DF_44D6_B836_8F1D740A86CF__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////////
//
// CTTSdraftDlg dialog

class CTTSdraftDlg : public CDialog
{
// Construction
public:

      void tdpsolaTerminate();
      void tdpsolaInitialize();
      void detect_vuvTerminate();
      void detect_vuvInitialize();
      void find_pmarksTerminate();
      void find_pmarksInitialize();
      void wavreadtTerminate();
      void wavreadtInitialize();
      void concatenateTerminate();
      void concatenateInitialize();
      void wavwritetTerminate();
      void wavwritetInitialize();

      struct x
      {
       CString phoneme[100];
       CString diphone[100];
      } word[15];

      int i;
      int j;
      int l;
      int count;

      CString filename;
    CStringArray Abbrev;
      CStringArray Acronym;

    void ParseToWord();
      void ParseToDiphone();
```

```
        void Play();

        void TextNormalization();
        void ConvertAbbreviations();
        void ConvertAcronyms();
      //void ConvertNumbers();

        CTTSdraftDlg(CWnd* pParent = NULL); // standard constructor

// Dialog Data
        //{{AFX_DATA(CTTSdraftDlg)
        enum { IDD = IDD_TTSDRAFT_DIALOG };
        CString     m_Text;
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CTTSdraftDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV
support
        //}}AFX_VIRTUAL

// Implementation
protected:
        HICON m_hIcon;

        // Generated message map functions
        //{{AFX_MSG(CTTSdraftDlg)
        virtual BOOL OnInitDialog();
        afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
        afx_msg void OnPaint();
        afx_msg HCURSOR OnQueryDragIcon();
        afx_msg void OnExit();
        afx_msg void OnSpeak();
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif //
!defined(AFX_TTSDRAFTDLG_H__DCA4D953_25DF_44D6_B836_8F1D740A86CF__INCLUDED_
)
```

```
CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//
// CTTSdraftDlg dialog

CTTSdraftDlg::CTTSdraftDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CTTSdraftDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CTTSdraftDlg)
	m_Text = _T("");
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in
Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CTTSdraftDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CTTSdraftDlg)
	DDX_Text(pDX, IDC_EDIT1, m_Text);
	DDV_MaxChars(pDX, m_Text, 300);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CTTSdraftDlg, CDialog)
	//{{AFX_MSG_MAP(CTTSdraftDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON1, OnExit)
	ON_BN_CLICKED(IDC_BUTTON2, OnSpeak)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//
// CTTSdraftDlg message handlers

BOOL CTTSdraftDlg::OnInitDialog()
```

```
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this
automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CTTSdraftDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code
below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CTTSdraftDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);
```

```
            // Center icon in client rectangle
            int cxIcon = GetSystemMetrics(SM_CXICON);
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(&rect);
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;

            // Draw the icon
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            CDialog::OnPaint();
      }
}

// The system calls this to obtain the cursor to display while the user
drags
//  the minimized window.
HCURSOR CTTSdraftDlg::OnQueryDragIcon()
{
      return (HCURSOR) m_hIcon;
}

void CTTSdraftDlg::OnExit()
{
OnOK();
}

void CTTSdraftDlg::OnSpeak()
{
UpdateData(TRUE);


      i=0;
      int s=0;
      count=0;

    while(CString(m_Text[i])!="\\")
      {

if(CString(m_Text[i])=="."||CString(m_Text[i])=="?"||CString(m_Text[i])=="!
")
        s++;//read the next line

            i++;
      };


    for(int t=0;t<s;t++)
       {
        ParseToWord();
        ParseToDiphone();
      Play();
      count +=(i+2);
       }
```

```
void CTTSdraftDlg::TextNormalization()
{

      ConvertAbbreviations();
      ConvertAcronyms();
      //ConvertNumbers();
}

void CTTSdraftDlg::ConvertAbbreviations()
{

 Abbrev.RemoveAll();
 Abbrev.SetSize(100);


//if(CString(word[j].phoneme[0])=="D"&&CString(word[j].phoneme[1])=="/"&&CS
tring(word[j].phoneme[2])=="R")
if(word[j].phoneme[0]=="D"&&word[j].phoneme[1]=="/"&&word[j].phoneme[2]=="R
")
 {
    char DR[]="dokîtâr";

    for(int n=0;CString(DR[n])!="\0";n++)
      {
       word[j].phoneme[n]=CString(DR[n]);
      }

 }


if(CString(word[j].phoneme[0])=="W"&&CString(word[j].phoneme[1])=="/"&&CStr
ing(word[j].phoneme[2])=="O")
 {
      char WO[]="wâyîzâro";

      for(int n=0;CString(WO[n])!="\0";n++)
      {
       word[j].phoneme[n]=CString(WO[n]);
      }

 }

}

void CTTSdraftDlg::ConvertAcronyms()
{

 Acronym.RemoveAll();
 Acronym.SetSize(100);


if(CString(word[j].phoneme[0])=="M"&&CString(word[j].phoneme[1])=="+"&&CStr
ing(word[j].phoneme[2])=="R")
 {
    char MRT[]="maHîbâr râdi?et tîgîray";

    for(int n=0;CString(MRT[n])!="\0";n++)
      {
```

```
        word[j].phoneme[n]=CString(MRT[n]);
       }

 }


if(CString(word[j].phoneme[0])=="M"&&CString(word[j].phoneme[1])=="."&&CStr
ing(word[j].phoneme[2])=="L")
 {
      char MLT[]="maHîbâr lîm'eat tîgîray";

      for(int n=0;CString(MLT[n])!="\0";n++)
      {
       word[j].phoneme[n]=CString(MLT[n]);
      }

 }

}


void CTTSdraftDlg::Play()
{
      i=0;
    int k=0;
      //int m=0;
   // int strglen=m_Text.GetLength();


      mxArray *temp;
      mxArray *InitialSil;
      mxArray *WordSil;
      mxArray *FileName;
      mxArray *s;
      mxArray *fs;
      mxArray *pm;
      mxArray *vuv;
      mxArray *pscale;
      mxArray *tscale;
      mxArray *y;
      mxArray *TStore;


      WordSil=mxCreateDoubleMatrix(800,1,mxREAL);
      InitialSil=mxCreateDoubleMatrix(100,1,mxREAL);
      fs=mxCreateDoubleScalar(8000);


    CString TempStore="D:\\aal0";
      TStore=mxCreateString(TempStore);


      wavreadtInitialize();
      concatenateInitialize();
      find_pmarksInitialize();
      detect_vuvInitialize();
      tdpsolaInitialize();
      wavwritetInitialize();
```

```
    temp=InitialSil;

   //do

while(CString(m_Text[count+i])!="."&&CString(m_Text[count+i])!="?"&&CString
(m_Text[count+i])!="!")
      {
            if(CString(m_Text[count+i])!=" ")
           {
         //filename="D:\\AmharicDB\\"+word[k].diphone[i]+".wav";
            //filename="D:\\tempDB\\"+diphone[i]+".wav";
            filename="D:\\diphoneDB\\"+word[k].diphone[i]+".wav";
         FileName=mxCreateString(filename);
            s=mlfWavreadt(FileName);
         temp=mlfConcatenate(temp,s);
            i++;
         //m++;
           }

           else
           {
         s=WordSil;
         temp=mlfConcatenate(temp,s);
            i++;
            //m=0;
            k++;

           }

      };

      //while(i<strglen-1);
      //int index = count+i;
      //identify the type of sentence and adjust pitch and duration
accordingly

   if(CString(m_Text[count+i])==".")
      {
      pscale=mxCreateDoubleScalar(1);
      tscale=mxCreateDoubleScalar(1);
      }

      else if(CString(m_Text[count+i])=="?")
      {
      pscale=mxCreateDoubleScalar(1.2);
      tscale=mxCreateDoubleScalar(0.9);
      }

    else
      {
      pscale=mxCreateDoubleScalar(1.4);
      tscale=mxCreateDoubleScalar(0.8);
      }


    pm=mlfFind_pmarks(temp,fs);
      vuv=mlfDetect_vuv(temp,fs,pm);
```

```
        y=mlfTdpsola(temp,fs,pscale,tscale,pm,vuv);
        mlfWavwritet(y,TStore);


    BOOL play=sndPlaySound(TempStore,SND_SYNC);


        wavwritetTerminate();
        tdpsolaTerminate();
        detect_vuvTerminate();
        find_pmarksTerminate();
        concatenateTerminate();
        wavreadtTerminate();


        mxDestroyArray(temp);
        mxDestroyArray(WordSil);
        mxDestroyArray(FileName);
        mxDestroyArray(s);
        mxDestroyArray(fs);
        mxDestroyArray(pm);
        mxDestroyArray(vuv);
        mxDestroyArray(y);
        mxDestroyArray(pscale);
        mxDestroyArray(tscale);
        mxDestroyArray(InitialSil);
        mxDestroyArray(TStore);

}



void CTTSdraftDlg::wavreadtInitialize()
{

}

void CTTSdraftDlg::wavreadtTerminate()
{

}


void CTTSdraftDlg::concatenateInitialize()
{

}

void CTTSdraftDlg::concatenateTerminate()
{

}


void CTTSdraftDlg::find_pmarksInitialize()
{
```

```
}

void CTTSdraftDlg::find_pmarksTerminate()
{

}

void CTTSdraftDlg::detect_vuvInitialize()
{

}

void CTTSdraftDlg::detect_vuvTerminate()
{

}

void CTTSdraftDlg::tdpsolaInitialize()
{

}

void CTTSdraftDlg::tdpsolaTerminate()
{

}

void CTTSdraftDlg::wavwritetInitialize()
{

}

void CTTSdraftDlg::wavwritetTerminate()
{

}
```

APPENDIX C: MATLAB SCRIPTS

C-1: SCRIPT FOR FINDING PITCHMARK OF A SPEECH DATA

```
function pitch_marks = find_pmarks(speech, fs_in)
%
% function pitch_marks = find_pmarks(speech)
% This MATLAB function calculates and returns the pitch marks (placed at
% peaks in the short-time energy function) for the input speech, that is
% assumed to be sampled at 8 KHz.
%       speech:    the input speech data


p1 = round(fs_in/400);
p2 = round(fs_in/60);

spch = speech(:)';
xsamp = length(spch);


%calculate the approximate pitch contour based on energy peaks:
wlen = round((p1+p2)/3);
ecurve = conv(hanning(wlen)',spch.^2);
ecurve = conv(hanning(wlen)',ecurve);
ecurve = ecurve((1:xsamp)+wlen);
peaks = ([0,diff(ecurve)]>0) & ([diff(ecurve),0]<0);
index = 1:xsamp;
index(~peaks) = [];
Npeaks = length(index);
pitch = diff(index);
pitch1 = [pitch, pitch(Npeaks-1)];
pitch2 = [pitch(1), pitch];
mat_row1 = max(1,min(p2-p1+1,pitch1-p1+1));
mat_row2 = max(1,min(p2-p1+1,pitch2-p1+1));
z1 = ecurve([index(2:Npeaks),index(Npeaks)]);
z2 = ecurve([index(1),index(1:(Npeaks-1))]);


step_size = round(fs_in/192);
Nbatch2 = ceil(xsamp/step_size);
mat_col = round(1+(index-1)*(Nbatch2-1)/(xsamp-1));
subset = zeros(p2-p1+1,Nbatch2);
for n = 1:length(index)
      subset(mat_row1(n),mat_col(n)) = z1(n);
      subset(mat_row2(n),mat_col(n)) = z2(n);
end
path = rridem(subset,3)+p1-1;
pitch = round(interp1(1:Nbatch2,path,linspace(1,Nbatch2,xsamp)));

array = zeros(1,2*xsamp);
```

```
n = 1;
array(1) = 1;
while n < xsamp
      n = n + pitch(n);
      array(n) = 1;
end
peaks = 1:length(array);
peaks(~array) = [];

Xres = 500;
xpts = round(linspace(1,xsamp,Xres));
M = length(peaks);
N2 = p2;
N = 2*N2;


pointers = max(1,min(xsamp,([1:N]'-N2)*ones(1,M)+ones(N,1)*peaks));
MAT = reshape(abs(spch(pointers)),N,M) .* [hanning(N)*ones(1,M)];
path = rridem(MAT,4);
peaks = round(peaks+path-N2);
pitch_marks = peaks([peaks>=1]&[peaks<=xsamp]);
if (pitch_marks(1)~=1)
  pitch_marks=[1 pitch_marks];
end
if (pitch_marks(length(pitch_marks)~=xsamp))
  pitch_marks=[pitch_marks xsamp];
end

return



function path = rridem(MAT,N)
%
% y = rridem(MAT)
%
% This function traces a path from the first to the last columns
% of MAT, one that does not exceed slope == N (N integer >0) when
% assuming that successive rows are separated by one unit, and that
% successive columns are separated by one unit) and has the maximum
% possible cumulative MAT values along the path.  The output
% path y adheres to the sample points of MAT.

% calculate best-path cumulative errors:
[mrows,mcols] = size(MAT);
sf = mean(mean(MAT));
MAT = [-Inf*ones(N,mcols); MAT; -Inf*ones(N,mcols)];
best_paths = zeros(size(MAT));
range = N + (1:mrows);
T = zeros(1+2*N,mrows);
B = zeros(1+2*N,mrows);
```

```
R = zeros(1,(1+2*N)*mrows);
for i = -N:N
        B(i+N+1,:) = ones(1,mrows) * sf/sqrt(1+i*i);
        R(mrows*(i+N)+[1:mrows]) = range + i;
end
for col = 2:mcols
        T = reshape(MAT(R,col-1),mrows,1+2*N)';
        [temp1,temp2] = max(T+B);
        MAT(range,col) = MAT(range,col) + temp1';
        best_paths(range,col) = temp2';
end


% trace the optimal path backwards through the cum. error matrix:
best_paths = best_paths - N - 1;
path = zeros(1,mcols);
[total_error,row] = max(MAT(:,mcols));
path(mcols) = row;
for col = mcols:-1:2
        row = row + best_paths(row,col);
        path(col-1) = row;
end

path = path - N;

return



function W = hanning(N)

    W1 = (1 + cos(pi*linspace(-1,1,N+2)))/2;
    W = W1(2:(N+1));

return
```

C-3: SCRIPT FOR TD-PSOLA ALGORITHM

```
function y = tdpsola(s,fs,pscale,tscale,pm,vuv);
% vuv: her pitch mark arasýndaki kýsmýn v/uv kararý
%      (length(vuv)=length(pm)-1)
%      uv ise pitch scaling yapýlmýyor

if (pscale ==1 & tscale==1)
  y=s;
else
  %Find pitch marks if necessary
  if (nargin==4)
    pm = find_pmarks(s,fs);
  end
  %Do v/uv detection if necessary here
  if (nargin<6)
    %vuv=ones(length(pm)-1,1);%detect v/uv here!!!
    vuv=detect_vuv(s,fs,pm);
  end

  %Apply pitch scaling on the pitch marks
  % and find new pitch marks (pm_ps)
  pm_ps=pm;
  if (pscale~=1)
    pshift=0;
    for i = 2:length(pm)
      T0=pm(i)-pm(i-1);
      if (vuv(i-1)>0)
        if (pscale>1)
          pshift=pshift-round(T0*(pscale-1)/pscale);
        else
          pshift=pshift+round(T0*(1/pscale-1));
        end
      end
      pm_ps(i)=pm(i)+pshift;
    end
  end

  %Find frames to be repeated/deleted for time scaling
  % and store this information in useds
  new_tscale=tscale*pm(length(pm))/pm_ps(length(pm_ps));
  avg=sum(diff(pm_ps))/(length(pm_ps)-1);
  if (new_tscale>1)
    useds=zeros(1,length(pm_ps)-2);
    tot=new_tscale;
    for i = 1 : length(useds)
      while(tot>1)
        useds(i)=useds(i)+1;
        %tot=tot-1;
```

```
      tot=tot-(pm_ps(i+1)-pm_ps(i))/avg;
    end
    tot=tot+new_tscale;
  end
elseif (new_tscale<1)
  useds=ones(1,length(pm_ps)-2);
  tot=new_tscale;
  for i = 1 : length(useds)
    while(tot<1)
      useds(i)=useds(i)-1;
      %tot=tot+1;
      tot=tot+(pm_ps(i+1)-pm_ps(i))/avg;
    end
    tot=tot-(1-new_tscale);
  end
end

%Synthesize the signal with overlap-add using pm_ps and useds
start=1;
count=1;
for i=1:length(useds)
  if (useds(i)>0)
    final(count,:)=[start pm(i) pm(i+2) 0];
    count=count+1;
    start=start+pm_ps(i+1)-pm_ps(i)+1;
  end
  for j=2:useds(i)
    final(count,:)=[start pm(i) pm(i+2) mod(j,2)];
    count=count+1;
    start=start+pm_ps(i+1)-pm_ps(i)+1;
  end
end
numfrm=size(final,1);
ylen=max(final(:,1)+(final(:,3)-final(:,2)+1));
y=zeros(ylen,1);
if (pscale>1)
  w=zeros(size(y));
end
for i = 1 : numfrm
  start=final(i,1);
  len=final(i,3)-final(i,2)+1;
  wgt=window(len,'han');
  frm=s(final(i,2):final(i,3));
  if (final(i,4))
    frm=wrev(frm);
  end
  y(start:start+len-1)=y(start:start+len-1)+frm.*wgt;
  if (pscale>1)
    w(start:start+len-1)=w(start:start+len-1)+wgt;
  end
end
```

```
  if (pscale>1)
    for i=1:ylen
      if w(i)==0
        w(i)=1;
      end
      y(i)=y(i)/w(i);
    end
  end
end
```

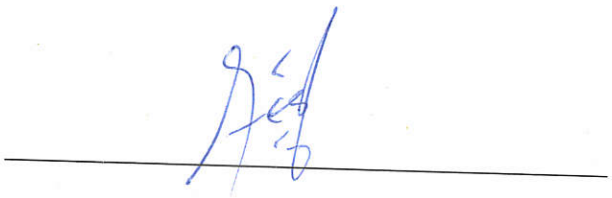
C-4: SCRIPT FOR WRITING PM, VUV, WAVE DATA INTO FILE

```
% The necessary information pm, vuv, wave data were placed
% into a file and this information is loaded into memory when the
%system is activated in order to reduce computation at run time.

function WriteDataToFile;
WavPath = 'D:\diphoneDB\';
PmkPath = 'D:\PMark';
VuVPath='D:\vuv';
AmpPath='D:\amp';

FileName=input('Enter a filename(end in " ") : ','s');
fid1 = fopen([PmkPath FileName '.pmk'], 'wt');
fid2 = fopen([VuVPath FileName '.vuv'], 'wt');
fid3= fopen([AmpPath FileName '.amp'], 'wt');
while (strcmp(FileName,'qt')~=1)

temp=[WavPath FileName '.wav'];
y = wavread(temp);
   fprintf(fid3,'%s ',FileName);
   fprintf(fid3, '%f', y);
   fprintf(fid3, '\n');
fs=8000;
pm = find_pmarks(y, fs);

   fprintf(fid1,'%s',FileName);
   fprintf(fid1, '%5d', pm);
   fprintf(fid1, '\n');

vuv = detect_vuv(y, fs, pm);
  fprintf(fid2,'%s',FileName);
  fprintf(fid2, '%5d', vuv);
  fprintf(fid2, '\n');
```

```
FileName=input('Enter a filename(end in " "): ','s');
end
fclose(fid1);
fclose(fid2);
fclose(fid3);
```

DECLARATION

I, the undersigned, hereby declare that this thesis is my original work and has not been submitted for a degree in any other university and that all sources of materials used for the thesis have been duly acknowledged.



Tesfay Yihdego

June 2004

This thesis has been submitted for examination with my approval as university advisor

Dr. Eneyew Adugna