Design of Syllable-based Speech Segmentation Methods for Text-to-Speech (TTS) Synthesis System for Gujarati

by

Swati Talesara

201111046

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY

in

INFORMATION AND COMMUNICATION TECHNOLOGY

to

Dhirubhai Ambani Institute of Information and Communication Technology

July, 2013

Declaration

I hereby declare that

(i) The thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at DA-IICT and has not been submitted elsewhere for a degree,

(ii) Due acknowledgement has been made in the text to all the reference material used.

Signature of Student

Certificate

This is to certify that the thesis work entitled "Design of Syllable-based Speech Segmentation Methods for Text-to-Speech (TTS) Synthesis System in Gujarati" has been carried out by Swati Talesara (201111046) for the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology (DA-IICT) under my supervision.

Prof. Hemant A. Patil

Thesis Supervisor

Acknowledgement

First and foremost I would like to extend my gratitude to Department of Information Technology (DEiTy), Govt. of India which, as the funding body, made the project on "Development of Text-to-Speech (TTS) System in Indian Language Phase II" possible. I owe equal gratitude to my guide Prof. Hemant Patil for his valuable guidance and insights, which was the major reason for making my contribution to the research community a reality. Without his motivation, I would not have been able to complete the transition from a student to a researcher.

I would also like to thank my expert member Prof. Manjunath Joshi for his critical feedback and valuable discussions during last one year. I am also thankful to my thesis reviewer, Prof. Suman K. Mitra.

Many other people deserve my whole hearted thanks for making this work possible. I am extremely grateful to my co-worker Ms. Tanvina B. Patel for helping me a lot throughout my thesis. Next, I would like to thank my other co-workers Ms. Janki Akhani, Mr. Nilesh Vaishnav, Mr. Bhavik Vachhani, Mr. Nirmesh Shah, Mr. Hardik Sailor, Mr. Maulik Madhavi and all other members of speech group at DA-IICT for their support throughout the project, the encouragement and motivation that they have always provided me and their mentoring which was very critical to my one year journey in the project.

Finally, I would like to thank my parents, husband, family members and friends who have always stood beside me in tough times without ever complaining about my workaholic nature. Without their affection, this journey would have been very difficult.

Table of Contents

Abstract

Text-to-speech (TTS) synthesizer has been proved to be an aiding tool for many visually challenged people for reading through hearing feedback. Although there are TTS synthesizers available in English and other languages as well, however, it has been observed that people feel more comfortable in learning in their own native language. Keeping this point in mind, Gujarati TTS synthesizer has been built and the building process has been described in this thesis. This TTS system has been built in Festival speech synthesis framework. Syllable is taken as the basic speech sound unit in building Gujarati TTS synthesizer as Indian languages are syllabic in nature. However, in building this unit-selection-based Gujarati TTS system, one requires large Gujarati labeled corpus. This task of labeling is the most time-consuming and tedious task. This task requires large manual efforts. In this thesis work, an attempt has been made to reduce these manual efforts by automatically generating labeled corpus at syllable-level. To that effect, a Gaussian-based segmentation method has been proposed for automatic segmentation of speech at syllable-level.

It has been observed that the percentage correctness of labeled data is around *80* % for both male and female voice as compared to *70* % for group delay-based labeling. In addition, the system built on the proposed approach shows better intelligibility when evaluated by a visually challenged subject. The word error rate is reduced by *5* % for Gaussian filter-based TTS system, compared to group delay-based TTS system. Furthermore, 5 % increment is observed in correctly synthesized words. The main focus of this thesis has been to reduce the manual efforts required in building TTS system (which are the manual efforts required in labeling speech data) for Gujarati language.

List of Principal Symbols and Acronyms

ω	Digital frequency
μ	Mean
σ	Standard deviation
α	*-3dB* bandwidth

Other minor symbols are defined at first occurrence; where necessary some symbols are redefined in the text.

TTS	Text-to-Speech
DSP	Digital Signal Processing
SPS	Statistical Parametric Synthesis
LPC	Linear Prediction Coefficients
WSF	Window Scale Factor
MOS	Mean Opinion Score
STM	Spectral Transition Measure
LTS	Letter-to-Sound
IVS	Inherent Vowel Suppression
CART	Classification and Regression Tree
STE	Short-Term Energy
pdf	Probability Density Function
MFCC	Mel Frequency Cepstral Coefficients
MFC	Mel Frequency Cepstrum
MSE	Mean Squared Error

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Text-to-Speech (TTS) Synthesis System

Speech is the most basic form of communication between human beings. Every human being cannot read and write, however, can communicate using speech. Probably, because of its high influence on the day-to-day life of human being, speech processing has been a subject of research from decades. The goal of TTS is the automatic conversion of written text into corresponding speech. The speech synthesis field has witnessed much advancement in the past few decades. Due to the increasing applications of TTS, research activity in this area is also increasing. TTS system is widely useful for the people with disabilities. It has been proved to be a very useful tool for visually challenged persons. TTS are also employed to aid people with speech impairment. It is also used for public announcement at the railway stations, in retrieving information over telephone services such as banking services, in call centers, etc.

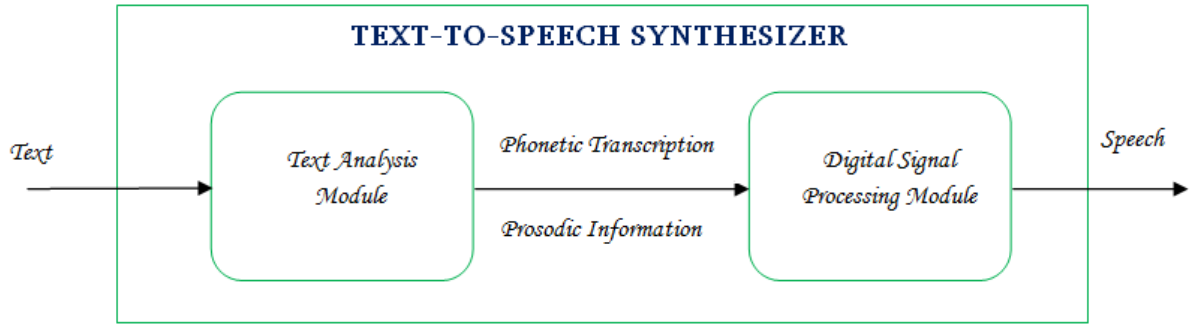


Figure 1.1: General TTS system (After [1])

A general TTS synthesizer comprises of a text analysis module and digital signal processing (DSP) module. Figure 1.1 shows the functional diagram of a very general TTS synthesizer [1]. The text analysis module produces a phonetic

transcription of the text read, together with the desired intonation and rhythm (i.e., *prosody*). The DSP module produces the synthetic speech corresponding to the transcription produced by text analysis module

1.1.1 Text Analysis Module

Figure 1.2 shows the skeleton of a general text analysis module. The text analysis stage is extremely difficult, as this stage needs to produce all the information required by DSP module (for producing speech) from merely text only. However, a mere text does not contain all of the information needed to produce speech. The first block of the text analysis module is Text-to-Phonetics block which converts the input text to phonetic transcription and the second block is Text-to-Prosody which produces *prosodic* information.

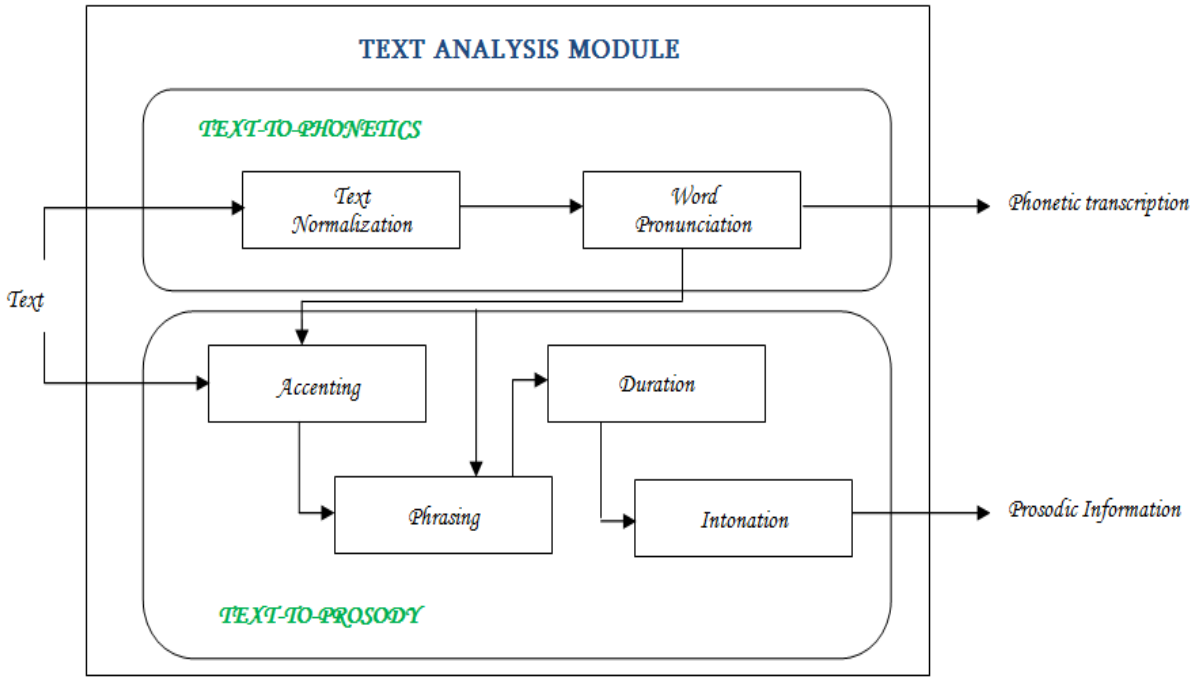


Figure 1.2: Text analysis module of a general Text-to-Speech (TTS) system (After [2])

a) Text-to-Phonetics

The Text-to-Phonetics block can be further be broken down into *text normalization* module and a *word pronunciation* module. Following is the brief description of these two modules:

- **Text Normalization:** A text normalization module organizes the input text into manageable lists of words. It identifies numbers, abbreviations, acronyms and idiomatic and expands them into full text. This is commonly done by using regular grammars.
- **Word Pronunciation:** Once the sequence of words has been generated using text normalization module, their pronunciation can be determined. A simple Letter-to-Sound (LTS) rule may be applied where words are pronounced as they are written. Where this is not the case, a morpho-syntactic analyzer may be used. A morpho-syntactic analyzer tags the speech with various identities such as prefixes, roots and suffixes, and organizes the sentences into syntactically related groups of words such as nouns, verbs, and adjectives. The pronunciation of these can then be determined using a *lexicon*.

b) Text-to-Prosody

The term *prosody* refers to certain properties of the speech signal such as audible changes in the pitch (i.e., intonation), loudness, tempo, duration, stress and rhythm. The *naturalness* of speech can be described mainly in terms of prosody. Prosodic events are also referred to as *suprasegmental* phenomena as these events appear to be time-aligned with syllables or group of syllables rather than with segments (i.e., sounds, phonemes) [1].

The pattern of prosody is used to communicate the meaning of sentences [2]. For example, the sentences "Open the door." and "Open the door?" have very different prosody. In terms of intonation contour (defined as rise and fall of the pitch throughout the utterance), the first sentence is *declarative* and has a relatively flat pitch contour, whereas the second is questioning and exhibits a *rise* in pitch at the end of the phrase.

The Text-to-Prosody block produces prosody information using the text and output of the word pronunciation module. This block can be further broken down into smaller processes that determine accenting, phrasing, duration, and intonation for each sentence. Following is the brief description of these four processes:

- **Accenting:** Accent or stress assignment is based on the category of the word. For example, context words (such as nouns, adjectives and verbs) are typically accented and function words (such as prepositions and auxiliary verbs) are usually not [2]. This information is used for predicting the *intonation* and *duration*.
- **Phrasing:** Sentences are broken down into phrasal units and phrase boundaries are assigned to the text. These boundaries indicate pauses and the resetting of intonation contours.
- **Intonation:** Intonation clarifies type and meaning of the sentence (neutral, imperative or question*).* In addition, intonation also conveys information about the speaker characteristics (such as his/her gender and age) and emotions [3]. The intonation module generates a pitch contour for the sentences.
- **Duration:** Segmental duration is an essential aspect of prosody [2]. It affects the overall rhythm of the speech, stress and emphasis, the syntactic structure of the sentence, and the speaking rate [2]. There are many factors which contribute to the duration of a speech segment. Some of them are the identity of the phone itself, the identity and characteristics of neighboring phones, the accent status of the syllable containing the phone, its phrase position and the speaking rate and dialect of the speaker.

1.1.2 Digital Signal Processing (DSP) Module

The DSP module uses the phonetic transcription and prosodic information produced by text analysis module to produce speech. This can be done in the following ways, *viz.*,

- By using the series of rules which formally describes the influence of one phoneme on to the other (i.e., coarticulation).
- By storing numerous instances of each speech sound unit and using them as they are, as ultimate acoustic units.

- One instance of this technique, called hidden Markov model (HMM)-based speech synthesis, has recently been demonstrated to be very effective in synthesizing acceptable speech. This review also contrasts these techniques with the more conventional technique of unit-selection synthesis that has dominated speech synthesis over the last decade.

Based on the above ways, the main classes of TTS systems that have emerged are, *synthesis-by-rule* and *synthesis-by-concatenation* [3] and *Statistical Parametric Synthesis.* Figure 1.3 shows a general DSP module.

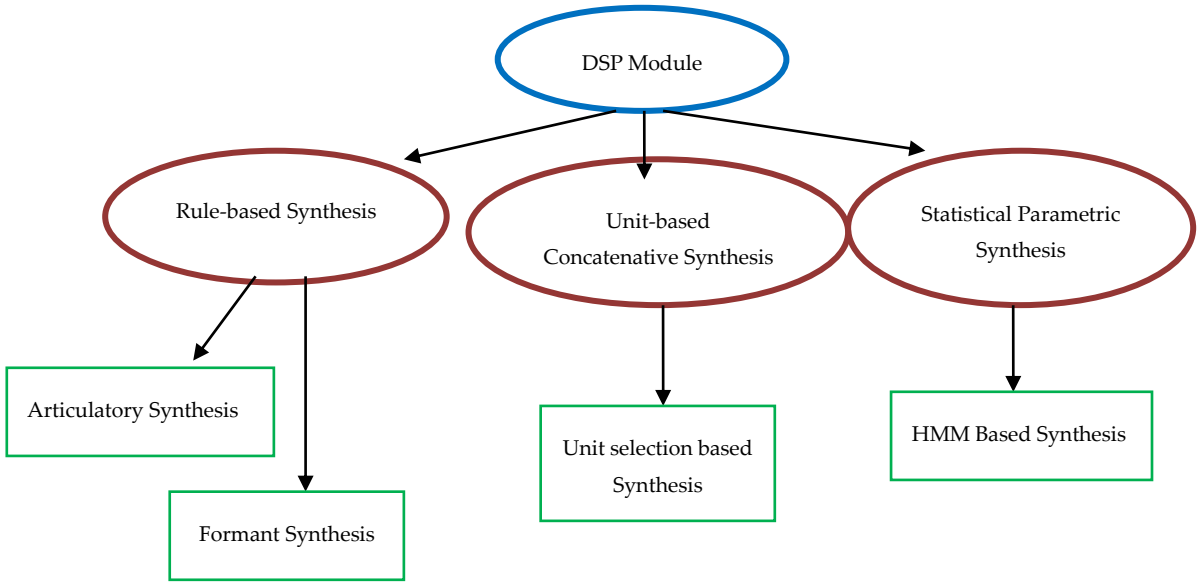


Figure 1.3: Digital Signal Processing (DSP) module

a) Rule-based synthesis

A rule-based synthesizer is a *knowledge*-based system. It is based on the explicit description of the control parameters over time representing the speech signal. It consists of a set of rules derived from an explicit knowledge of production phenomena, at least at the acoustic-level [3]. Following are the two rule-based synthesis approaches:

- *Articulatory Synthesis* - Articulatory synthesis generates speech by direct modeling of the human behavior of articulators, so in principle; it is the most

satisfying method to produce high quality speech. In practice, it is one of the most difficult methods to implement. The articulatory control parameters include lip aperture, lip protrusion, tongue tip position, tongue tip height, tongue position and tongue height [4].

- *Formant Synthesis*- In formant synthesis, the basic assumption is that the vocal tract transfer functions can be satisfactorily modeled by simulating formant frequencies and formant amplitudes. The synthesis thus consists of the artificial reconstruction of the formant characteristics to be produced [4].

b) Unit-based concatenative synthesis

In concatenative speech synthesis approach, there is no need to determine speech production rules. Hence, concatenative synthesis is simpler than rule-based synthesis [5]. Concatenative synthesis generates speech by connecting natural, prerecorded speech sound units. These sound units can be words, syllables, half-syllables, phonemes, half-phone, diphones or triphones [4]. The main challenges in this approach are the prosodic modification to speech units and resolving discontinuities at unit boundaries. Prosodic modification results in artifacts in the speech which in turn make the speech sound unnatural. Unit-selection-based speech synthesis technique, which is a kind of concatenative synthesis, solves this problem by storing numerous instances of each unit with varying prosodic information. The unit that best matches the target prosody is selected and concatenated. However, in unit-selection-based concatenative synthesis, large speech corpus is required to be labeled, which is very tedious and time-consuming. Thus, there is a genuine need to automate this labeling process.

c) Statistical Parametric Synthesis (SPS)

The idea here is to derive a set of parameters such as formant frequencies, linear prediction coefficients (LPC) for each phone in a language and develop coarticulation rules of joining of two phone segments. Statistical parametric speech synthesis based on hidden Markov model-like models, has become competitive with established unit-selection-based concatenative techniques over the last few years.

1.2 Motivation and Objective of the Thesis

While building unit-selection-based TTS synthesizer, after selecting the training corpus, next important task is to label the speech sound units. Manual segmentation and labeling of speech units is very tedious, time-consuming, expensive and error prone. Moreover, due to variability in human perception, large inconsistencies are observed. Because the cost and effort required for labeling are significant for large speech databases, the need to automate it is mainly motivated by the need for large speech databases to build unit-selection-based TTS systems. Moreover, manual segmentation is not real time whereas automatic segmentation can be real time. Automatic segmentation is not error free, however, it is inherently consistent and results are reproducible. Hence, it is necessary to automate identification and segmentation of appropriate speech sound units. Presently, there is a minimum-phase group delay-based labeling tool available for Indian languages, *viz.*, DONLabel, developed by [6]. However, this tool is not completely automatic. One needs to manually set a parameter, *viz.*, Window Scale Factor (WSF) for setting resolution of boundary detection at syllable-level for each speech utterance. There are two objectives of this thesis. First objective is to explore signal processing techniques to automate labeling of large speech corpus at syllable-level required while building TTS system in Gujarati. Second objective is to build TTS system for Gujarati using the **.lab* files generated using the available technique of segmentation and proposed technique of segmentation and furthermore, to compare these two systems in terms of word error rate and Mean Opinion Score (MOS) of TTS system. The main emphasis of the thesis is to reduce the manual labeling efforts and analyze trade-offs between manual labeling efforts *vs.* automatic labeling with corresponding effect on naturalness and intelligibility of TTS voice.

1.3 Organization of the Thesis

The rest of the thesis is organized into *5* chapters. Chapter *2* discusses the process of developing a unit-selection-based TTS system in Gujarati in Festival speech synthesis framework. Chapter 3 describes various speech segmentation methods at syllable-

level, *viz.,* minimum-phase group delay-based segmentation method, proposed Gaussian-based segmentation method and Spectral Transition Method (STM-based) segmentation method. In Chapter 4, algorithm for automatic generation of labeled corpus has been described. Experimental results have been presented in Chapter 5. Chapter 6 includes summary of the thesis and possible directions for future research work.

1.4 Contribution in the Thesis

In this thesis, a new Gaussian-filter based speech segmentation algorithm at syllable-level is proposed. This proposed algorithm is found to perform reasonably well as compared to other existing algorithms, *viz.,* group-delay based method and spectral-transition measure (STM) based methods.

1.5 Chapter Summary

In this chapter, an overview of general text-to-speech synthesizer has been presented. A general TTS synthesizer consists of text analysis and DSP module. The text analysis module generates the phonetic transcription and prosodic information from the given text input. The DSP module describes the approaches that can be used in generating speech from the given phonetic transcription and prosodic information. Concatenative waveform synthesis is used in state-of-the-art high quality synthesizer [7]. Unit-selection-based speech synthesis is a concatenative waveform synthesis approach. The Festival speech synthesis framework is used in development of unit-selection-based TTS. Festival unit-selection algorithm is used for this purpose. In the next chapter, how to develop a unit-selection based synthesizer in Gujarati in Festival framework have been described.

Chapter 2

Development of Unit-Selection-Based TTS system in Gujarati

2.1 Introduction

The main aim of this chapter is to describe how an unrestricted Gujarati TTS system can be developed using unit-selection-based synthesis technique. Unrestricted TTS system is capable of synthesizing speech from any domain preserving sufficient amount of speech intelligibility and naturalness in synthesized voice. Here, the festival framework is used for building TTS. Festival is a free, language-independent speech synthesis engine that can be used for developing text-to-speech synthesis systems [8]. It is highly *flexible* and has a *modular* architecture. Figure 2.1 shows the block diagram of unit-selection-based TTS system in Festival framework.

The input to the TTS system is the text in Gujarati language. The text normalization block used to expand the numbers, acronyms, abbreviations, etc. present in the input text. The prosodic phraser block used to chunk the input text based on punctuation. Lexicon contains list of all speech sound units, *viz.*, syllables and phones, present in the waveform repository. The pronunciation generation block uses lexicon and letter-to-sound (LTS) rules to generate the sequence of basic speech sound units. The lexicon is a list of all speech units, *viz.*, syllables and phones, present in the waveform repository. The LTS rules splits a word into syllables first and if syllables not present then split it into phones. The unit-selection algorithm block first generates a target specification for the identified speech sound units and then picks the best sequence of speech units which minimizes both the *join cost* and the *target cost*. The waveforms of these selected speech units are then concatenated to produce synthetic speech [9]. In this case, TTS system building consists of two phases, *viz.*, the voice building phase and the synthesis phase. In the synthesis phase, speech will be synthesized for the given input text. The following section describes training phase or voice building and synthesis phase required for building TTS system

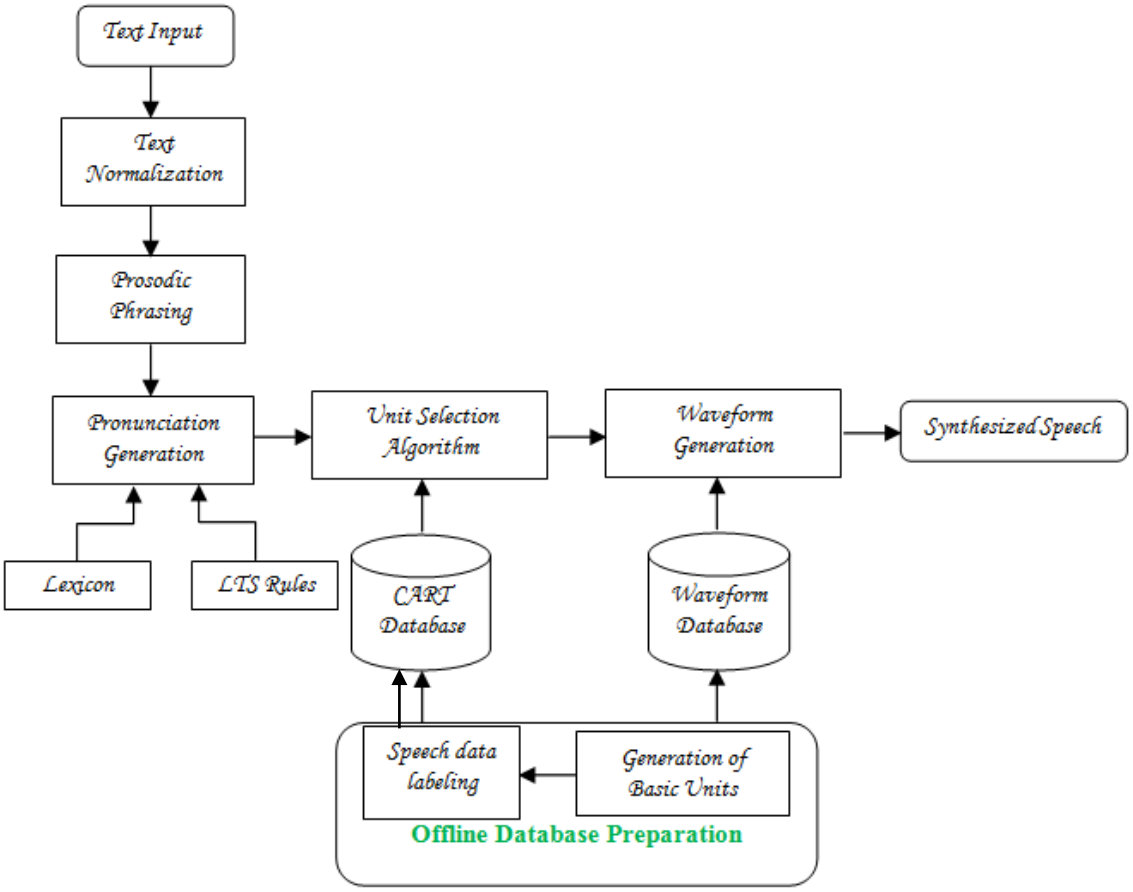


Figure 2.1: Block diagram of unit-selection-based TTS system in Festival speech synthesis framework (After [9]).

2.2 Voice Building Stage

Major issues to be addressed in the training phase are choosing basic speech sound units for synthesis, text corpus selection, optimal text selection for speech recordings, voice talent/speaker selection, recording the speech corpus, letter-to-sound (LTS) rules, labeling of speech corpus (most tedious, laborious, expensive, error prone and time- consuming task to build TTS voice) and organizing the training corpus based on the speech parameters and contextual information using *clustering* techniques.

2.2.1 Basic sound unit appropriate for TTS

The choice of the basic speech sound units highly affects the quality of unit-selection-based TTS synthesizer. The basic speech sound unit (in the sequence of increasing size) used for speech synthesis can be of various types such as half-phones, phones, diphones, triphones, polyphones, syllables and polysyllables [7]. The choice of basic speech sound unit is language-dependent. For example, for synthesizing American English, labeling is done at phoneme-level whereas for synthesizing Indian languages, syllable is found to be appropriate unit [10]. If the basic speech sound unit chosen is of smaller size such as half-phones or phones, then the quality of synthesized speech may not be good. This is because of more number of concatenation points, which will result in perceptual distortion due to lack of coarticulation at the joins (i.e., discontinuity in energy flow of various speech sound units). However, with basic speech sound units of smaller size, limited training corpus will be required. On the other hand, with larger size sound units such as diphones, triphones, polyphones, syllables and polysyllables, the quality of synthesized speech will be good as the number of concatenation points will be less. However, with larger size basic speech sound units, large training corpus will be required and hence storage requirement will also be large. Therefore, while choosing the basic sound unit, there is a trade-off between the size of the unit and the quality of synthesized speech. It has been shown that syllables are best suited as the basic speech sound unit for Indian languages because Indian languages are syllable centered [10] . Syllable is a speech unit having a vowel at the nucleus surrounded by none or more consonants. A syllable is typically of the form C*VC*, where C is a consonant, V is a vowel and C* indicates there may be none or more consonant present. However, the number of syllables present in a language can be large. There are many advantages of syllable units, *viz.*,

- Syllable units contain most of the coarticulation information.
- As syllable units are larger in size, number of concatenation points decreases when syllable is used as the basic speech sound unit.

- As syllable boundaries are characterized by low energy; spectral mismatches at the boundary are less perceived.

Therefore, syllables are chosen as the basic speech sound unit for building TTS system in Gujarati.

2.2.2 Text corpus collection

In order to synthesize a good quality voice, the TTS system should contain the training corpus (i.e., text material and corresponding speech recording) from different domains. This is needed to get better syllable coverage for Gujarati language. However, the entire collected corpus cannot be used as it is. Thus, the collected corpus requires pre-processing, as corpus may contain similar sentences and sentences with lots of repetitions of similar words with similar context. If the whole corpus is used as it is, then this redundant corpus results in unnecessary increase in storage space which results in reduction in efficiency of the TTS system. Therefore, in order to reduce the text corpus while keeping the syllable coverage as high as possible, optimal text needs to be selected which is described in next sub-section.

2.2.3 Optimal text selection

Optimal text is selected in order to cover as many syllables as possible and each syllable is having certain minimum number of repetitions. In order to select optimum text, following process is used.

- The text is divided into lines containing around *10* words.
- For each line, a score is calculated. This score would be based upon the number of "*new syllables*" in the line and the "*frequency*" of those syllables. The syllabification script is used for this purpose. "*New syllable*" means the syllable that has not already been selected by the optimization process in the optimized text.
- After each of the iteration, top *300* lines were appended to a file which would contain optimized text.

- The above process is repeated iteratively till all the lines are gathered in the optimized text.

The optimized text contains the line-by-line corpus, such that the lines having highest score are at the top and lowest scores are at the bottom. Figure 2.2 shows how the number of *unique* syllables (i.e., syllables without repetition) increases as the number of lines increase. It can also be seen in Figure 2.2 that number of syllables saturates after some lines. The decision about how many lines from this optimized corpus used for recording is decided dynamically based on the number of unique syllables covered and the number of hours of actual speech to be recorded.

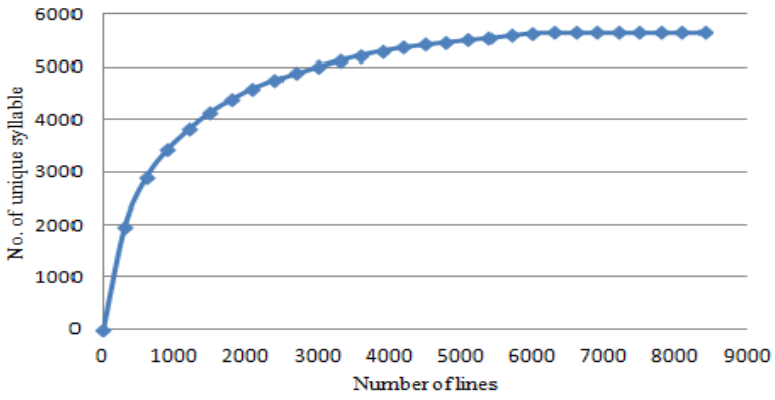


Figure 2.2: Accumulation of unique syllables for Gujarati language over the optimized corpus ('+' marks the data-points here)

2.2.4 Voice artist/speaker selection

The speaker selection should be done properly because the voice quality of speaker will directly affect the synthesized voice in concatenative speech synthesis approach. Voice samples of multiple speakers are collected. The collected samples are subjected to variations in *pitch*, *tempo* and *amplitude*. The level of variations done should be limited so that the quality of voice does not change drastically. A shrillness factor of voice can also be considered in order to evaluate the comfort (during long-time hearing). The speaker whose voice loses its quality least on applying these prosodic variations is selected.

2.2.5 Voice recording

After selecting the required text for recording and a speaker, the actual recording is scheduled to be performed in stages. The recording must be done in a high quality studio to ensure the collected speech corpus would be clean. While recording, the things that should be taken care are:

- The energy of the speaker remains almost same for whole recording session.
- Recording should be preferably done in morning session as the speaker will not be tired at that time.
- Speaker must be physically fit at the time of recording. For instance, if the speaker is suffering from cough and cold at the time of recording, this will directly affect the recorded voice and ultimately the TTS system.

The collected data is recorded in a professional setting, with the following technical specifications *48 KHz* sampling rate (later down-sampled to *16 KHz*), *16 bits* resolution, single-channel recording. The speech data is down-sampled to *16 KHz* because *16 KHz* sampled voice results in reasonably good quality synthesized voice and also memory requirement will be one-fourth time of that of *48 KHz* voice used.

2.2.6 Text normalization

This step defines set of rules for expansion of commonly used abbreviations, acronyms, numbers based on the context, etc.

2.2.7 Letter-to-Sound (LTS) rules

LTS rules play an important role while developing any TTS system. These rules indicate how the written text has to be spoken. The LTS rules for Gujarati are very much similar to LTS rules of Hindi [10]. The characters of Gujarati and its transcription are given in Appendix A. In case of Gujarati language, what is written and what is spoken has close correspondence. However, *inherent vowel* (i.e., short /a/) associated with each consonant is not spoken always. Depending on the context, inherent vowel associated with a consonant may not be spoken. This is referred to as

schwa deletion or Inherent Vowel Suppression (IVS). For example, the word *fala* [fruit] is mapped to sounds /*f*/ /*a*/ /*l*/, ignoring the vowel associated with /*l*/.

The rules to determine IVS of a consonant character have been derived with the help of Hindi IVS rules and observing few words of Gujarati. These rules may be incomplete. The rules are as follows [10]:

- The character present at the final position of a word always undergoes IVS. For instance, /*la*/ in *kamala* (કમલ) will undergo *schwa deletion*.
- The character present at the first position of a word never undergoes IVS. For instance, /*ka*/ in *kamala* will *not* undergo schwa deletion. However, there is an exception for this rule. When a word *jha* occurs in Gujarati then although it can be considered to be at the first position of a word, however it will undergo schwa deletion and schwa deleted *jh* get attached to the previous word. For instance, consider the phrase *jheeii jha* (જીઈ ઝ), although /*jha*/ is at first position but it will undergo schwa deletion here and get attached to the previous word, hence the resultant word for syllabification will be *jheeiijh.*
- Two *successive* characters of a word never undergo IVS. For instance, since /*la*/ in *kamala* will undergo schwa deletion, therefore /*ma*/ will not undergo schwa deletion.
- The character present at the *middle* position of a word undergoes IVS if either the next character has a vowel other than /*a*/ or the next character in the word is not just the last character. For instance, in *kamala,* /*ma*/ will not undergo schwa deletion since the next character has a vowel /*a*/ only and also it is the last character whereas in *kamalaa,* /*ma*/ will undergo schwa deletion since the next character has a vowel /*aa*/ (which is other than vowel /*a*/).
- If the nasal /*n'*/ (Gujarati anuswara 'ં') is present in the word then it can be mapped either to half pronounced /n/ or /m/ sound depending upon the category to which the following character belongs. For example,

- If *labial* category occurs after a nasal /*n'*/ then the sound of anuswara will be that of /m/. For example, in *an'baa* (અંબા), the anuswara is preceded by /*b*/ which is a *labial* category then that anuswara corresponds to /*m*/ *sound*.
- In *an'tar* (અંતર), the anuswara is preceded by /*t*/ which is in a *dental* category, therefore the anuswara sound will be that of /*n*/.

Similarly, there are other variations of /*n*/ which occurs in the velar, palatal and retroflex category. However, if the nasal /*n'*/ occurs at the end of a word, then it is interpreted as mild /*n*/ *sound* (i.e., sound of Gujarati *chandrabindu*). In this case, it is known as *nasalized vowel*. Anuswara issue is an example of one-to-many mapping. Here, it is the single anuswara that can be mapped to five different nasalized sounds.

2.2.8 Syllabification rules

The syllabification rules are required to break the words into syllables. By analyzing several words, it has been observed that the syllabification rules of Gujarati are similar to syllabification rules of Hindi [10]. These rules are as follows:

- When a vowel is immediately followed by nasals such as /n'/half pronounced /m/ or /n/ sound, then nasals would be considered as a part of the vowel and also the same syllable. For instance, /*n'*/ in *san'skaar* will be a part of syllable containing /*sa*/.
- When exactly two consonants are present between two consecutive vowels, the first consonant would be a part of *coda* of the previous syllable and the second consonant would be a part of *onset* of the next syllable. For instance, syllables of word *karta* would be /*kar*/ /*ta*/. However, there are few exceptions for this rule which are as follows:
  - When the second consonant is the member of the set {/*r*/ /*s*/ /*sh*/ /*shh*/}, then both the consonants would be a part of onset of the next syllable. For instance, /*putri*/ would be split as /*pu*/ and /*tri*/.

- When three or more consonants are present between two vowels, the first consonant would be a part of coda of the previous syllable and the remaining consonants would be a part of onset of the next syllable. For instance, syllables of word *san'skaar* would be /*san'*s/ and /*kaar*/.

Pronunciation rules are constructed which incorporate both LTS and syllabification rules.

2.2.9 Labeling of the speech corpus

Once the optimum text is selected, corresponding speech is recorded and pronunciation rules are created, the next important task is to label the speech corpus to prepare the speech database for building TTS voice. In the labeling processes, given the syllables (which are obtained using pronunciation rules) of the text and corresponding recorded speech, one has to align which part of speech corresponds to which syllable. This can be done manually either using the interface of Wave Surfer or Audacity or using labeling tool (such as DONLabel [11]). The accuracy of labeling greatly affects the quality of unit-selection-based speech synthesis.

2.2.10 Creation of Lexicon

Lexicon is list of all speech sound units, (here syllables and phones which has been used in fallback), that are present in the waveform repository. This is used in the synthesis phase to check whether the speech sound unit required for a text to be synthesized is available or not. Consider an instance when only syllables are present in lexicon and no mechanism of fallback is incorporated. As for a language, syllables are not fixed; there may be a case when the text to be synthesized may contain a syllable which is not present in waveform repository. In that case, if syllable is not found in lexicon, then that text would not be synthesized, and error will come indicating that syllable had no *duration* information.

2.2.11 Clustering of speech sound units

As the speech database contains many realizations of the same syllables, clustering techniques are used in the festival framework to organize the speech corpus in

systematic manner so that the search becomes easier. Classification and regression trees (CART) are used to represent the clusters in festival framework. A decision is created for each type of speech unit present in the speech corpus. The leaves of this decision tree are the list of candidate units. CART trees can be built based on different feature of speech unit such as the position of speech unit in word, phrase, sentence, context of speech sound unit, type of speech sound unit, etc.

It has been observed that syllables of the same type differ in its characteristics depending upon its position in a word. Syllable occurring at the start of a word is larger in duration as compared to the same syllable occurring at the middle of a word. Similarly, syllable occurring at the middle of a word is larger in duration as compared to the same syllable occurring at the end of a word. Same syllable occurring at different position in a word also vary in *energy* and *pitch*. As suggested in [12], pre-clustering has been done based on the position of the syllable in the word. Syllables are tagged appropriately with *begin*, *middle* and *end* tag depending on their position in the word. The first syllable of a word is always tagged with begin tag, last syllable with end tag and remaining syllables are tagged with middle tag. For instance, syllables of a word *pathshala* with tag will be */path_beg/*, */sha_mid/* and */la_end/*. Syllables of the same type (i.e., based on its position in the word) are clustered together. During speech synthesis, in case a syllable of the required position is not available, then syllable of same type occurring at an alternate position is picked, using an *order of preference*. If the syllable required at begin is not available then middle syllable is preferred. If the syllable required at middle is not available then begin syllable is preferred. If the syllable required at end is not available then middle syllable is preferred.

Clustering can further be fine tuned using the following phonological features of the syllable [13], [14]:

- Type of the syllable *viz.*, V, C*V, VC*, C*VC* ('*' indicates none or more consonant)
- Length of vowel *viz.*, long and short
- Type of vowel

- Height of vowel *viz.,* high, mid and low
- Vowel frontness *viz.,* front, mid and back
- Place of coarticulation of the first and last consonant *viz.,* labial, palatal, dental, labio-dental, alveolar and velar

2.3 Synthesis Stage

In order to synthesize speech in festival framework, the input text utterance is first parsed into sequence of largest speech sound units (i.e., first syllable then phone) present in the lexicon. Festival will make use of pronunciation rules for converting the text into *sequence* of speech sound units. After obtaining the sequence of speech units from the parser, linguistic analysis module will generate phonetic and contextual features associated with each unit. If prosodic model have also been built during voice building, then the target prosody is also generated for the input text by prosodic analysis module. For the given sequence of target units, the unit-selection algorithm selects an optimal set of speech units that best matches the target speech unit specification and with the other speech units in sequence without any *perceptual distortion*. For every target unit, the unit-selection algorithm first consider a list of candidate units, U_i, from the speech corpus. The candidate unit which minimizes the sum of two costs (*viz., target cost* and *join cost*) is then selected as the final choice. A function $Tdist(U)$ is defined as the distance of a unit U to its cluster center (which is synonymous to finding the target cost). Another function $Jdist(U_i, U_{i-1})$ is also defined to find the join cost between a candidate unit U_i and the previous candidate unit U_{i-1}. A Viterbi search is then used to find the *optimal path* through the candidate speech units that minimizes the following expression:

$$\sum_{i=1}^{n}\big(Tdist(U_i) + W \times Jdist(U_i, U_{i-1})\big), \quad (1)$$

where W is a weight that can be set to give more importance for the *join cost* over *target cost* [9]. The waveforms of these speech units are then concatenated to produce synthetic speech.

2.3.1 Fallback of speech sound units

In the unit-selection-based synthesis approach, only those sound units which are present in the speech corpus are available for synthesis. Thus, in order to synthesize everything (i.e., all possible text), at least one instance of each syllable must be present in the speech corpus. However, unlike number of phones, numbers of syllables are also not fixed. Thus, practically, it is impossible to cover all the syllables of a language. Therefore, there must be some strategy by which sound units that are not present in the speech corpus could be handled. As phones are fixed and small in number, so if a syllable is not present at the time of synthesis, then syllables are split into phones and phones are concatenated to produce synthesized speech. However, for this to happen, at the time of voice building stage, some lines are selected (such that all the phones are covered) and label at phone-level and corresponding phone-level clusters are built. For instance, syllables of *kamala* would be /*ka*/ and /*mal*/. If instance of /*mal*/ syllable is not present in the speech corpus, then it will be further split into phones /*ma*/ and /*l*/ and these phones are concatenated to produce /*mal*/ sound. Thus, fallback of speech sound units helps in making an unrestricted TTS system.

2.4 Chapter Summary

In this chapter, steps involved in developing a syllable-based Gujarati TTS in festival framework have been described which has primarily two stages, *viz.*, the voice building stage and the synthesis stage. Selection of optimum text, recording of text by a good speaker, deriving LTS rules and syllabification rules for Gujarati and labeling the corpus at syllable-level are the key steps in voice building TTS. The most time-consuming and tedious task is speech labeling. The quality of TTS voice (both *naturalness* and *intelligibility*) depends on the quality of labeling. However, the *manual* efforts require in labeling are very large. In this thesis, efforts have been made to reduce the efforts required in labeling by *automating* the labeling process. The following chapter describes the existing minimum-phase group delay-based speech segmentation algorithms at syllable-level.

Chapter 3

Speech Segmentation Approaches at Syllable-Level

3.1 Introduction

Segmentation of continuous speech into syllables find its potential application in Automatic Speech Recognition (ASR) systems and Text-to-Speech (TTS) Synthesis systems, more specifically in unit-selection-based TTS system for Indian languages, as it requires large amount of data to be labeled. Therefore, algorithms for automatic labeling of training corpus have been proposed by the researchers. In general, there are two broad categories of speech segmentation algorithms. One class of algorithms performs the speech segmentation when the underlying sequence of phonemes is assumed to be known. Another class of segmentation algorithms uses *no* knowledge of the underlying phoneme sequence contained within the speech waveform; instead the segment boundaries are identified at time instants, where there is a high degree of change in the acoustic properties of the waveform.

Syllable consists of three parts, *viz.*, the onset, rime and coda. The *onset* and *coda* can consist of consonants, while the *rime* consists of vowel. Figure 3.1 indicates syllables of a Gujarati word, *gareeba* (ગરીબ) (i.e., */ga/* (ગ) and */reeb/ (*રીબ*)*) obtained using LTS and syllabification rules of Gujarati language.

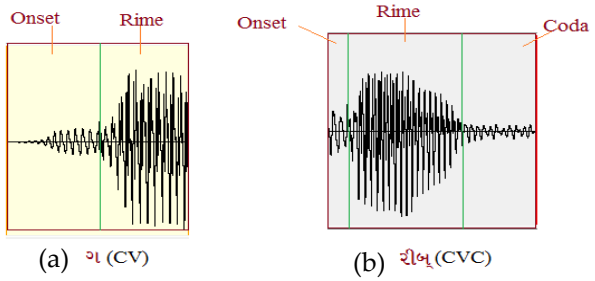


Figure 3.1: Syllables of a Gujarati word *gareeba* (ગરીબ) (a) syllable */ga/* (ગ) and (b) syllable */reeb/* (રીબ).

A syllable typically consists of a vowel surrounded by a consonant. The vowels are of high energy surrounded by consonants which are of low energy. In the other words, syllable boundary is a junction where *coarticulation* in speech reduces and again increases. The high energy regions in the short-term energy contour correspond to *syllable nuclei*, and the valleys at both ends approximately correspond to syllable boundaries [15]. However, the raw STE function cannot be used directly for speech segmentation task because of the significant local energy fluctuations due to the presence of consonants. Instead of choosing minima of raw STE directly as syllable boundary, fixed thresholding can be used such that minimas below a particular energy threshold (to avoid lower energy peaks due to surrounding consonants) are only considered as valid syllable boundary. However, this technique does not work well when there are high energy variations across the speech signal. In case of continuous speech signal, fixed thresholding cannot be used as energy is generally high at the start of a sentence and decreases towards the end of a sentence. In such a realistic scenario, an *adaptive* thresholding can be used. However, the threshold value to be used needs to be learnt continuously from the speech signal. Furthermore, the region over which the adaptive threshold is computed will become crucial, i.e., too large a region will miss correct boundaries, while too short a region will generate *spurious* boundaries [15] and hence there is a trade-off between location of the correct syllable boundaries and value of adaptive threshold.

If the local energy fluctuations in the STE contour are smoothed out, then the valleys at both ends of the nucleus can be considered as the boundaries of a syllable. The main aim of every segmentation algorithm described in this chapter is to remove the local variations in STE contour of a speech signal while retaining the *syllabic structure*.

3.2 Short-Term Energy (STE) Function

As speech is produced as a result of interaction of time-varying vocal tract system with time-varying excitation, therefore the speech signal is non-stationary in nature and the energy associated with speech signal is time varying in nature. Hence, the

interest for any automatic processing of speech is to know how the energy is varying with time and to be more specific, energy associated with short-term region of speech. The short-term energy (STE) of a signal is an estimate of the energy of a real signal $s(n)$ over a short window of N samples. STE is defined as

$$E(n) = \sum_{l=n-N+1}^{n} |s(l)|^2, \tag{2}$$

where $E(n)$ represents the short-term energy at sample 'n' which is the sum of the squares of N samples at $n - N + 1$ through n. There are several windowing functions present; however, the mostly used ones include rectangular, Hanning and Hamming. For all time-domain parameters estimation, generally rectangular window is used for its simplicity.

3.3 Minimum-Phase Group Delay-based Segmentation Approach

Minimum-phase group delay-based segmentation method is proposed in [16]**.** It is a *semi-automatic* segmentation method. In this method, STE alone is considered for segmentation of continuous speech at syllable-level. This method uses the result that the group delay function of the minimum-phase signal can be used in spectrum estimation [15]. Firstly, the minimum-phase signal corresponding to the STE is obtained and then its group delay function is computed. The minima in the resultant group delay correspond to the minima in the STE and hence, they are considered as syllable boundary. The following section briefly describes some of the important concepts of signal processing which are used in this method.

3.3.1 Group delay function and its properties

Group delay is defined as the *negative* derivative of the unwrapped Fourier transform phase. With the phase specified as continuous function of ω, the group delay of a system with the frequency response $G(e^{j\omega})$ is defined as

$$\tau_g(\omega) = grd\left[G\left(e^{j\omega}\right)\right] = -\frac{d}{d\omega}\left\{\arg\left[G\left(e^{j\omega}\right)\right]\right\}, \tag{3}$$

where $\arg[G(e^{j\omega})]$ is referred to as the *continuous* (i.e., unwrapped) *phase of* $G(e^{j\omega})$. The deviation of the group delay from a constant indicates the degree of nonlinearity of the phase and is expressed in seconds or samples. The *additive* property of the group delay function and high resolution property of the minimum-phase group delay function are used in this method.

a) Additive property of the group delay function

Consider the multiplication of two systems with frequency response $G(e^{j\omega})$ and $G_2(e^{j\omega})$ in spectral-domain. Let us consider,

$$G(e^{j\omega}) = G_1(e^{j\omega}).G_2(e^{j\omega}), \tag{4}$$

$$|G(e^{j\omega})|e^{j\arg(G(e^{j\omega}))} = |G_1(e^{j\omega})|e^{j\arg(G_1(e^{j\omega}))}.|G_2(e^{j\omega})|e^{j\arg(G_2(e^{j\omega}))}, \tag{5}$$

$$|G(e^{j\omega})|e^{j\arg(G(e^{j\omega}))} = |G_1(e^{j\omega})|.|G_2(e^{j\omega})|e^{j(\arg(G_1(e^{j\omega}))+\arg(G_2(e^{j\omega})))}, \tag{6}$$

From equation (6),

$$|G(e^{j\omega})| = |G_1(e^{j\omega})|.|G_2(e^{j\omega})|, \tag{7}$$

$$\arg\left(G(e^{j\omega})\right) = \arg\left(G_1(e^{j\omega})\right) + \arg\left(G_2(e^{j\omega})\right). \tag{8}$$

Then, the group delay function is given by

$$\tau_g(\omega) = -\frac{\partial\left(\arg\left(G(e^{j\omega})\right)\right)}{\partial\omega} \tag{9}$$

$$\tau_g(\omega) = -\frac{\partial\left(\arg\left(G_1(e^{j\omega})\right)\right)}{\partial\omega} - \frac{\partial\left(\arg\left(G_2(e^{j\omega})\right)\right)}{\partial\omega} \tag{10}$$

$$\tau_g(\omega) = \tau_{g_1}(\omega) + \tau_{g_2}(\omega), \tag{11}$$

where $\tau_{g_1}(\omega)$ and $\tau_{g_2}(\omega)$ correspond to the group delay function of $G_1(e^{j\omega})$ and $G_2(e^{j\omega})$, respectively. From Eqs. (4) and (11), we see that *multiplication* in the spectral-domain becomes an *addition* in the group delay-domain. In order to demonstrate the power of the additive property of the group delay spectrum, three different systems are chosen, *viz.,* (i) a system with one complex conjugate pole pair at an angular frequency $\pi/7$ *rad*, (ii) a system with one complex conjugate pole pair at an angular frequency $\pi/4$ *rad* and (iii) a system with two complex conjugate pole pairs at an angular frequency $\pi/7$ *rad and* $\pi/4$ *rad*. This is shown in figure 3.2.

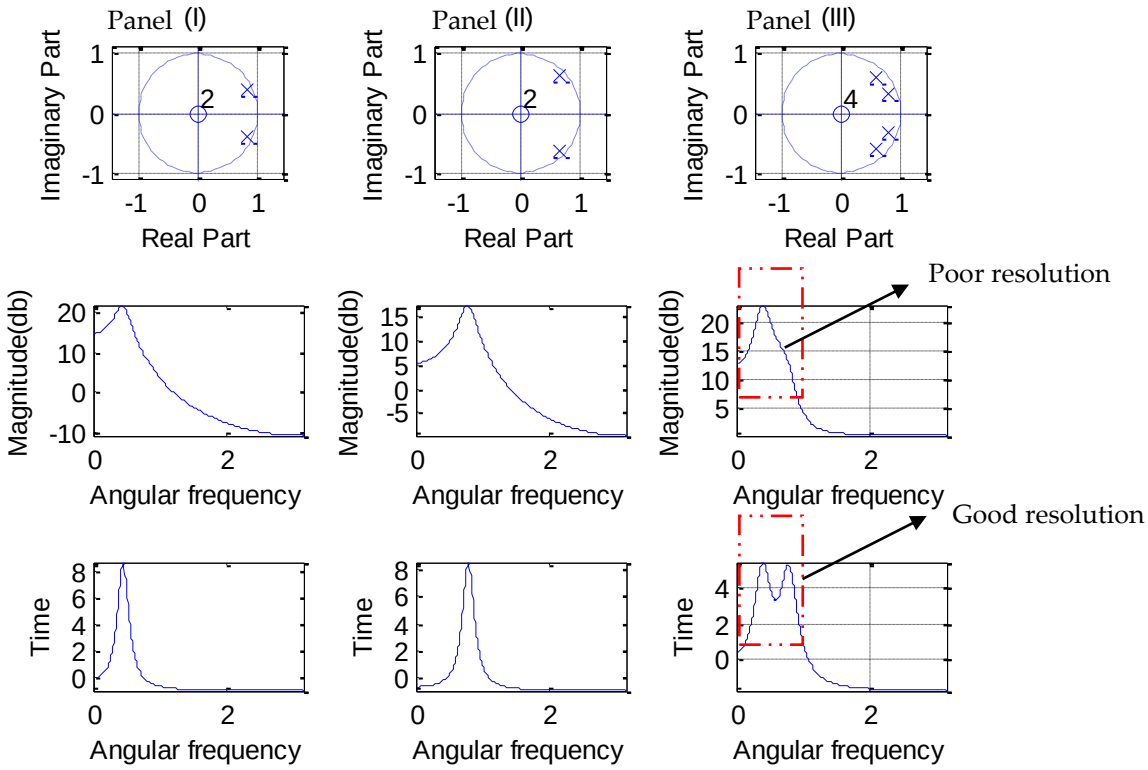


Figure 3.2: Resolving power of group delay spectrum: z-plane, magnitude spectrum and group delay spectrum (I) a pole pair inside the unit circle at $(0.9, \pi/7)$ and $(0.9, 13\pi/7)$, (II) a pole pair inside the unit circle at $(0.9, \pi/4)$ and $(0.9, 7\pi/4)$, and (III) a pole at $(0.9, \pi/7)$ and $(0.9, 13\pi/7)$ and another pole at $(0.9, \pi/4)$ and $(0.9, 7\pi/4)$, inside the unit circle.

Figure 3.2(b), (e) and (h) are the magnitude spectrum and Figure 3.2(c), (f) and (i) are the group delay spectrum of the systems shown in Figure 3.2(a), (d) and (g), respectively. It can be observed from the magnitude spectra that peaks are resolved well for the systems with single complex conjugate pole pair. However, in a system consisting of these two complex conjugate poles pairs, the peaks are *not* resolved well (as shown in Figure 3.2(h)) because of the multiplicative property of magnitude spectrum. However, from Figure 3.2(c), (f) and (i), it can be observed that the peaks are resolved *well* in the group delay-domain for all the three systems. This is due to the additive property of group delay function [17].

b) High resolution property of the minimum-phase group delay function

A signal $s(n)$ is called minimum-phase signal if and only if all the poles and zeros of the transform $S(z)$ lie inside the unit circle. Minimum-phase group delay function is the group delay function derived from the minimum-phase signal. Peaks correspond to poles while valleys correspond to zeros of the signal in the minimum-phase group delay function. However, this property is not possessed by non-minimum-phase signals.

It has been observed that in the group delay spectrum, for the minimum-phase system, both the peaks and valleys are resolved correctly (where peaks correspond to poles and valleys correspond to zeroes). In the case of non-minimum-phase systems, the zeros which are outside the unit circle are not resolved properly. The zeroes outside the unit circle, instead of showing up as valleys, appear as peaks at the corresponding angular frequencies. It is therefore, difficult to distinguish between poles and zeroes (when the zeroes are outside the unit circle) in the group delay spectrum [17]. This is illustrated in figure 3.3.

3.3.2 Property of STE function

Similarity between Fourier transform magnitude spectrum and STE function have been used in this method. Both the Fourier transform magnitude function and the STE function are positive, real-valued and non-zero in nature. However, while the magnitude spectrum of the short-term Fourier transform is *even-symmetric* about

the Y - axis, the STE function is *not* symmetric. The STE function can be reflected about the Y – axis in order to make it resemble as the Fourier transform magnitude spectrum. This can be done by concatenating STE function with its reflected function.

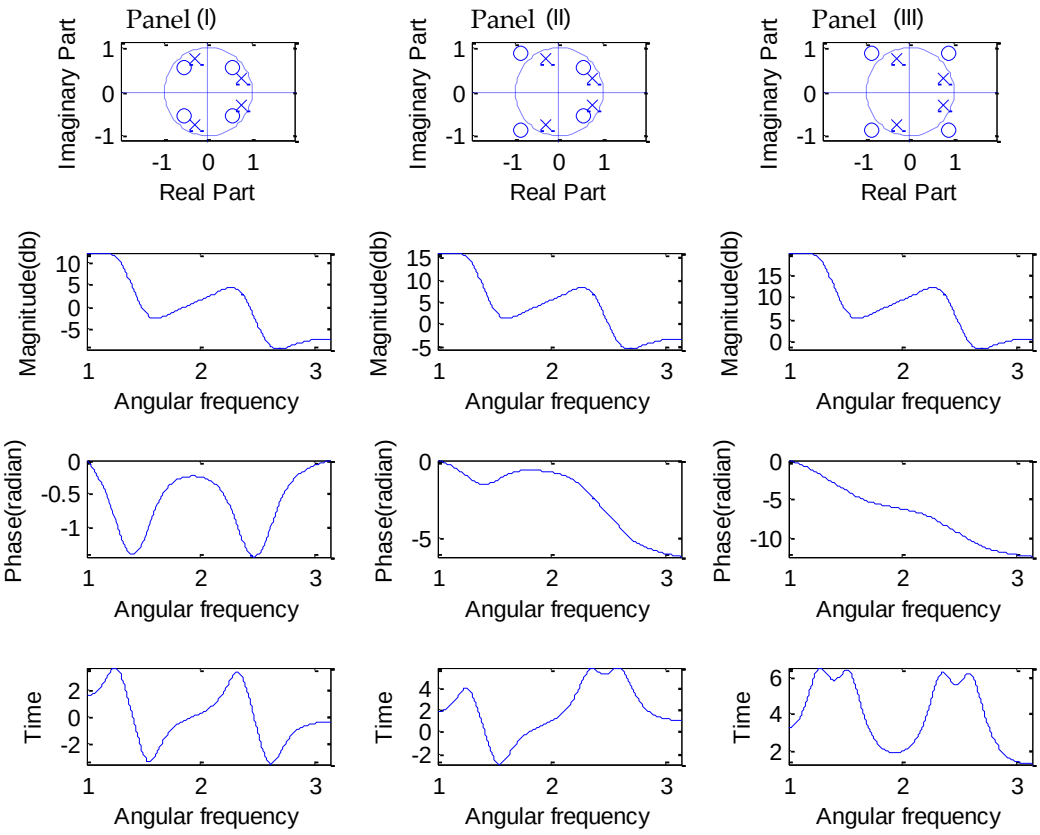


Figure 3.3: Group delay property of different types of signals: the z-plane, the magnitude spectrum, the phase spectrum, and the group delay spectrum for (I) minimum-phase, (II) non-minimum-phase with one zero pair outside the unit circle and (III) non-minimum-phase with both zero pairs outside the unit circle.

3.3.3 Root cepstrum approach [15], [16], [18]

It is used to derive a minimum-phase signal from any arbitrary magnitude spectrum $|S_{nm}(e^{jw})|$. Following steps are followed in this approach *viz.,*

1. Compute the squared magnitude spectrum $|S_{nm}(e^{jw})|^2$ from the given magnitude spectrum $|S_{nm}(e^{jw})|$.

2. Compute the IDFT of ($|S_{nm}(e^{jw})|^2$). Let this be $\tilde{c}(n)$.

3. Extract the *causal* portion of $\tilde{c}(n)$ which is a minimum-phase signal [19]. The poles of this resultant minimum-phase signal correspond to the peaks in the original magnitude spectrum $\left|S_{nm}\left(e^{jw}\right)\right|$.

Theoretical details of the root cepstrum (also known as spectral root homomorphing filtering) are given in [20].

3.3.4 Segmentation algorithm [15], [16]

Followings steps are involved in the minimum-phase group delay-based segmentation method:

1. Compute the short-term energy (STE) function $E(n)$ for a given digitized speech signal $s(n)$ using overlapped windows.
2. Symmetrize the sequence $E(n)$ by laterally inverting this sequence about Y-axis. This symmetrized sequence is now viewed as an arbitrary magnitude spectrum and denoted by $E(l)$.
3. Compute $(E(l))^{\gamma}$ where γ is $0 < \gamma \leq 2$ (Specifically, the value of γ has been optimized to *0.01*) to reduce the *dynamic range*.
4. Invert the sequence $(E(l))^{\gamma}$ in order to overcome problem of detecting *spurious valleys* which is due to the truncation of the signal in the root cepstral-domain. Let the resultant function be $\tilde{E}^{i}(l)$. Since the sequence is inverted in this step, it is the peaks of the group delay function which correspond to the segment boundaries now, not the valleys.
5. Compute the IDFT of the sequence $\tilde{E}^{i}(l)$. The resultant sequence $\tilde{c}(n)$, is the root cepstrum and the *causal* portion of it is a minimum-phase signal.
6. Apply a window to the resultant minimum-phase signal to obtain windowed causal sequence $c(n)$. The size of the window (N_c) applied is given by

$$N_c = \frac{\text{Short} - \text{term energy function size}}{\text{Window scale factor(WSF)}} \tag{12}$$

7. Compute the group delay function of the windowed causal sequence $c(n)$. Let it be denoted as $\widetilde{E_{gd}}(l)$.

8. The positive peaks in the group delay function $\widetilde{E_{gd}}(l)$ approximately correspond to syllable boundaries.

The frequency resolution in the magnitude spectrum as well as in the group delay spectrum depends on the size of the *cepstral lifter* (i.e., N_c). The window scale factor (WSF) takes integer values. If N_c is high, the resolution will also be high, i.e., closely spaced boundaries will be better resolved. In other words, the syllable-like units generated will have a shorter duration and will correspond to *monosyllables*. On the other hand, as N_c is reduced, the resolution decreases and larger units corresponding to polysyllables, bisyllables and trisyllables, start appearing. The MATLAB pseudo code for the above algorithm is included in box B.1 in Appendix B.

Figure 3.4 to figure 3.7 shows the output at various stages of a minimum-phase group delay-based segmentation algorithm for the speech signal of a Gujarati phrase, "*be ek athvaadxiyaa maate*". Figure 3.8 shows the speech signal of same Gujarati phrase, "*be ek athvaadxiyaa maate*" its short-term energy, and its group delay function along with the estimated boundaries (indicated using red color) and labels (i.e., speech sound units) corresponding to the estimated boundaries.

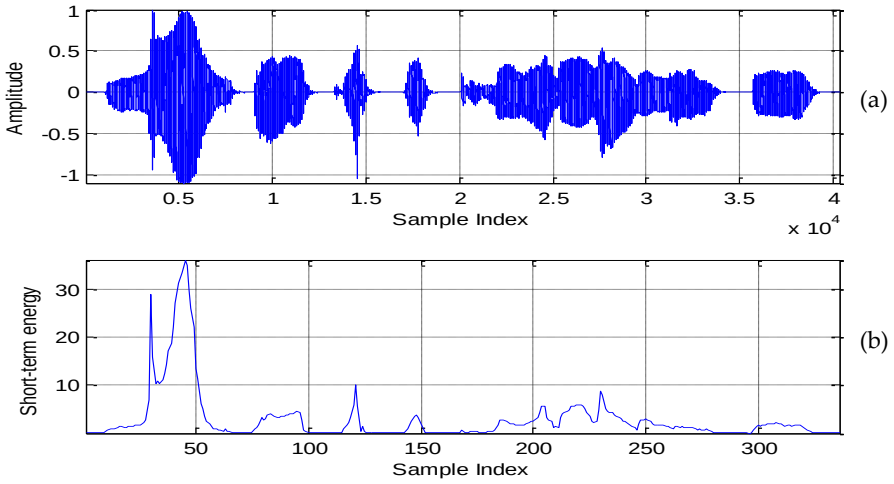


Figure 3.4: (a) Speech signal of a Gujarati phrase, "*be ek athvaadxiyaa maate*" and (b) short-term energy function of speech signal shown in (a) using rectangular window of length *12.5 msec* and overlap of *5 msec*.

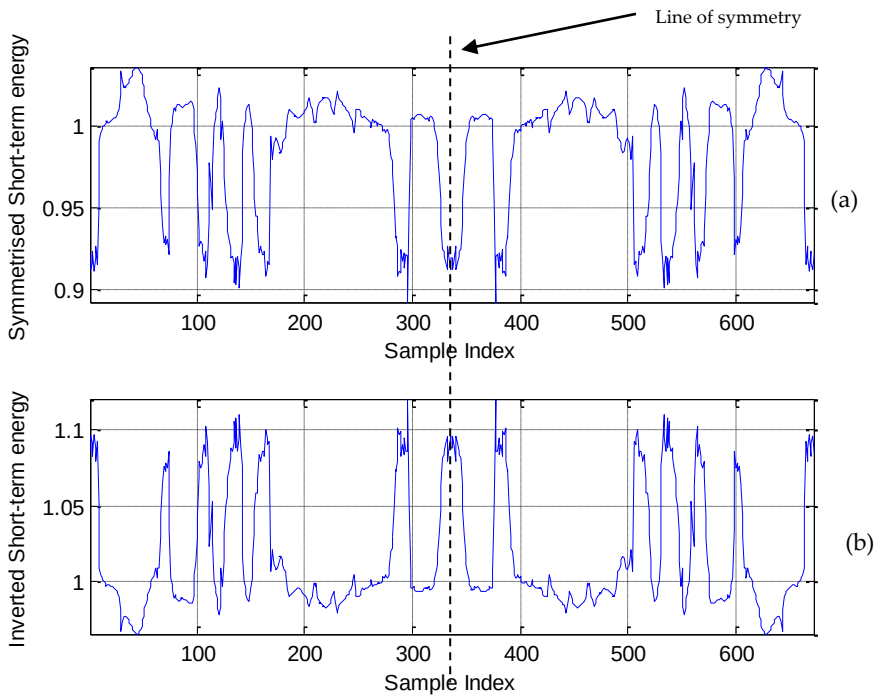


Figure 3.5: (a) Symmetrized short-term energy produced by lateral inversion of short-term energy sequence about the Y-axis (symmetry indicated by dotted lines) and (b) inverted short-term energy.

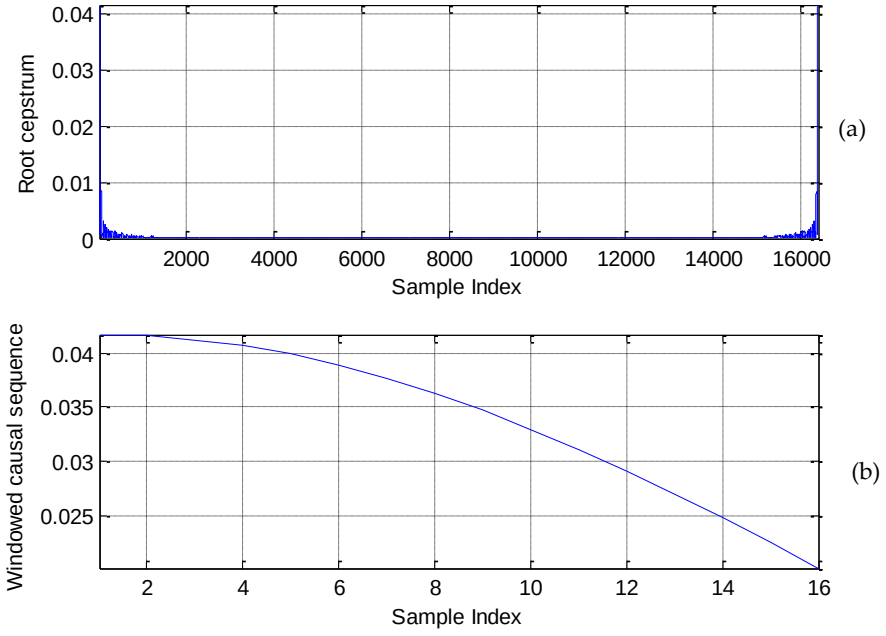


Figure 3.6: (a) Root cepstrum of speech signal and (b) windowed causal portion of root cepstrum having minimum-phase properties

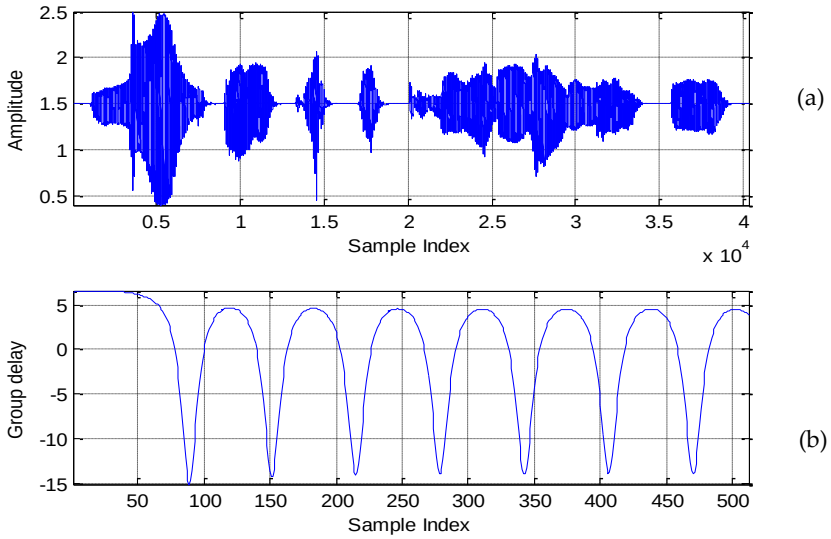


Figure 3.7: (a) Speech signal and (b) corresponding minimum-phase group delay function of windowed causal sequence.

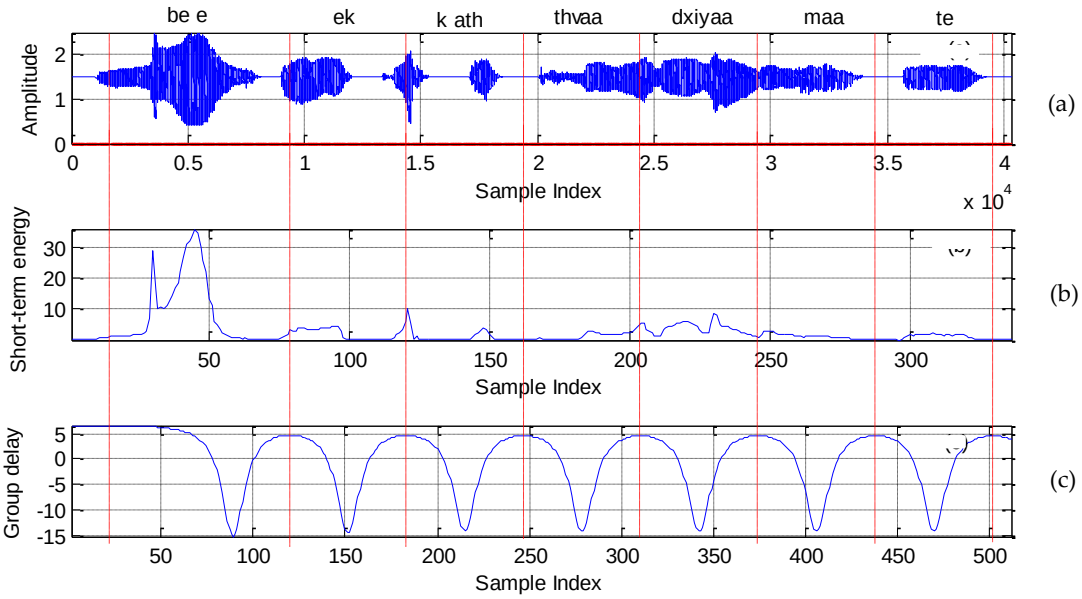


Figure 3.8: Syllable boundaries estimated (indicated in red color) using minimum-phase group delay method for the Gujarati phrase, "*be ek athvaadxiyaa maate*" (a) speech signal and (b) corresponding short–term energy and (c) corresponding minimum-phase signal group delay function.

The labels assigned to the estimated segment boundaries are written after listening to them in Audacity software. The estimated boundaries are */be e/ /ek/ /k ath/ /thvaa/*

/dxiyaa/ /maa/ /te/. The actual syllables of this phrase are */be/ /ek/ /ath/ /vaa/ /dxi/ /yaa/ /maa/ /te/*.

3.4 Proposed Gaussian Filter-Based Segmentation

With the objective of removing the local variations in STE contour of a speech signal while retaining the syllabic structure, the Gaussian-based segmentation approach has been attempted. It has been observed that if the short-term energy contour of a speech signal is low-pass filtered with the Gaussian filter (whose impulse response is shown in Figure 3.9 (a)), then the minimas of the filtered signal correspond to syllable boundaries.

3.4.1 Gaussian filter

The Gaussian filter has an impulse response given by

$$g(t) = e^{-\alpha^2 t^2}, \tag{13}$$

where α is related to *B*, i.e., *-3dB* bandwidth of the baseband Gaussian shaping filter [21]. Thus, frequency response of Gaussian filter is given by

$$G(f) = \frac{\sqrt{\pi}}{\alpha}\left[\exp\left(-\frac{\omega^2}{4\alpha^2}\right)\right], \tag{14}$$

Where $\omega = 2\pi f$. (15)

As probability density function (pdf) of Gaussian distribution is given as

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-(x-\mu)^2/2\sigma^2\right), \tag{16}$$

where μ is the mean and σ^2 is the variance of the distribution. Comparing equations (13) and (16), we can see

$$\alpha \propto \frac{1}{\sigma}. \tag{17}$$

i.e., *-3 dB* bandwidth and variance of the Gaussian filter are *inversely* proportional to each other.

It is well known that Gaussian filter can serve as excellent low-pass filter for one-dimensional (1-D) signals (e.g., speech). It is also well known that Gaussian filter has excellent time-frequency resolution property because the magnitude response of the Gaussian filter is also Gaussian in shape (This may be due to the fact that Fourier transform of Gaussian is Gaussian) [22]). Figure 3.9 (a) shows the impulse response of the Gaussian filter and (b) corresponding magnitude response.

If the *-3 dB* BW 'α' of the filter is increased (in other words variance 'σ' is decreased), the number of detected boundary increases and *vice-versa.* This has been shown in figure 3.10. Figure 3.10 (a), (c) and (e) shows impulse response of Gaussian filter with *-3 dB* BW of *70 Hz*, *90 Hz* and *110 Hz*, respectively. Estimated boundaries for the speech signal of a Gujarati phrase using these Gaussian filters of different *-3 dB* B.W are shown just below each filter's impulse response in Figure 3.10 (b), (d), (f) respectively.

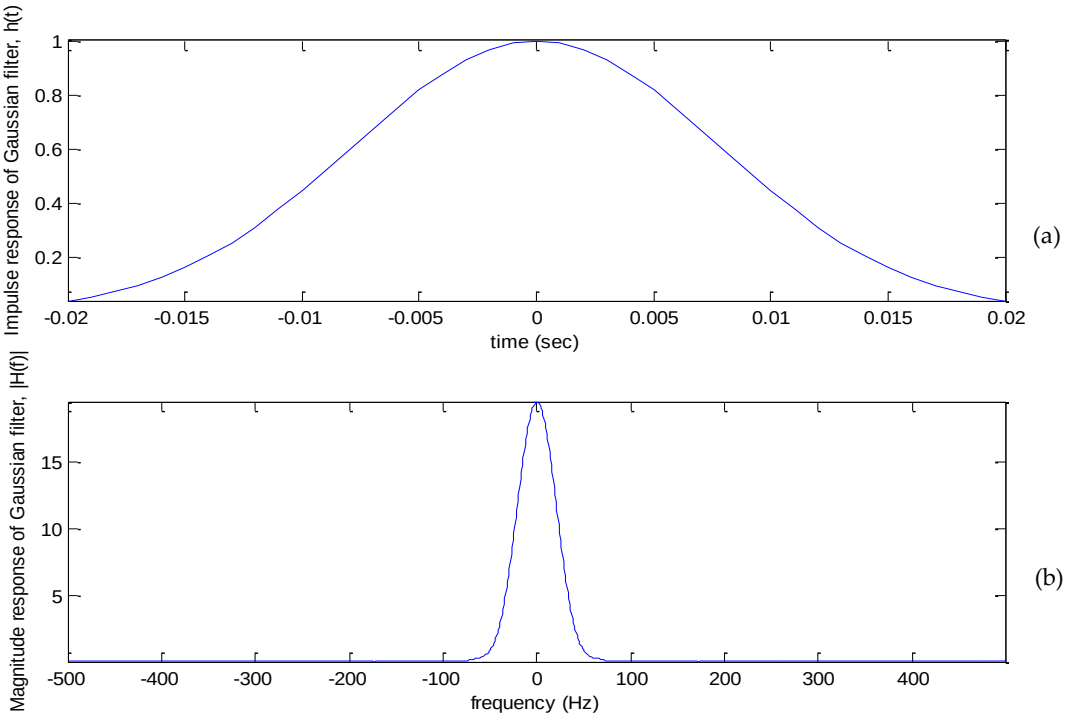


Figure 3.9: (a) Impulse response of the Gaussian filter and (b) corresponding magnitude response.

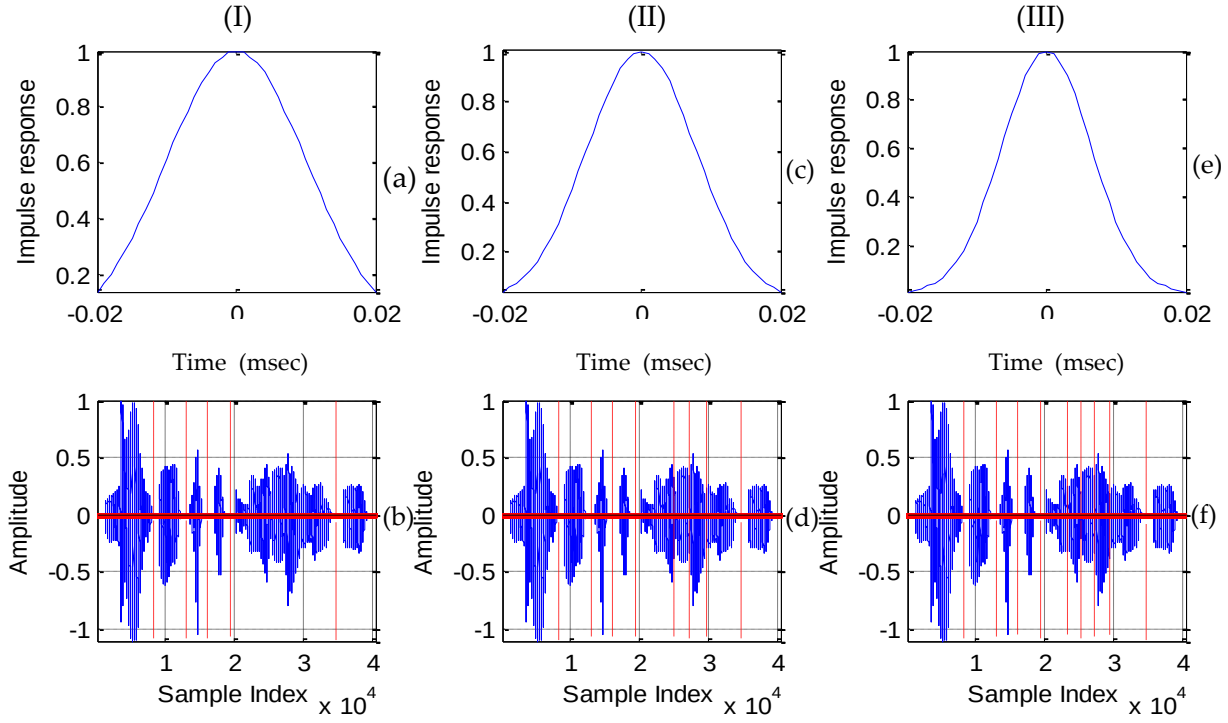


Figure 3.10: Impulse response of Gaussian filters with different *-3 dB* B.W. and estimated syllable boundaries (indicated in red color) for a Gujarati phrase using these Gaussian filters (I) *70 Hz -3 dB* B.W., (II) *90 Hz -3 dB* B.W. and (III) *110 Hz -3 dB* B.W.

3.4.2 Proposed segmentation algorithm

The steps involved in segmentation of continuous speech signal using Gaussian filter-based segmentation technique are as follows:

1. For a given digitized speech signal, compute its corresponding Short-Term Energy (STE) sequence $E(n)$, using overlapped window.
2. Low pass filter the *STE* sequence using Gaussian filter to get filtered *STE* sequence.
3. Locations of the minimas in the Gaussian filtered *STE* sequence gives estimated boundaries of syllable-like segments.

The MATLAB pseudo code for proposed segmentation algorithm is included in box B.2 in Appendix B.

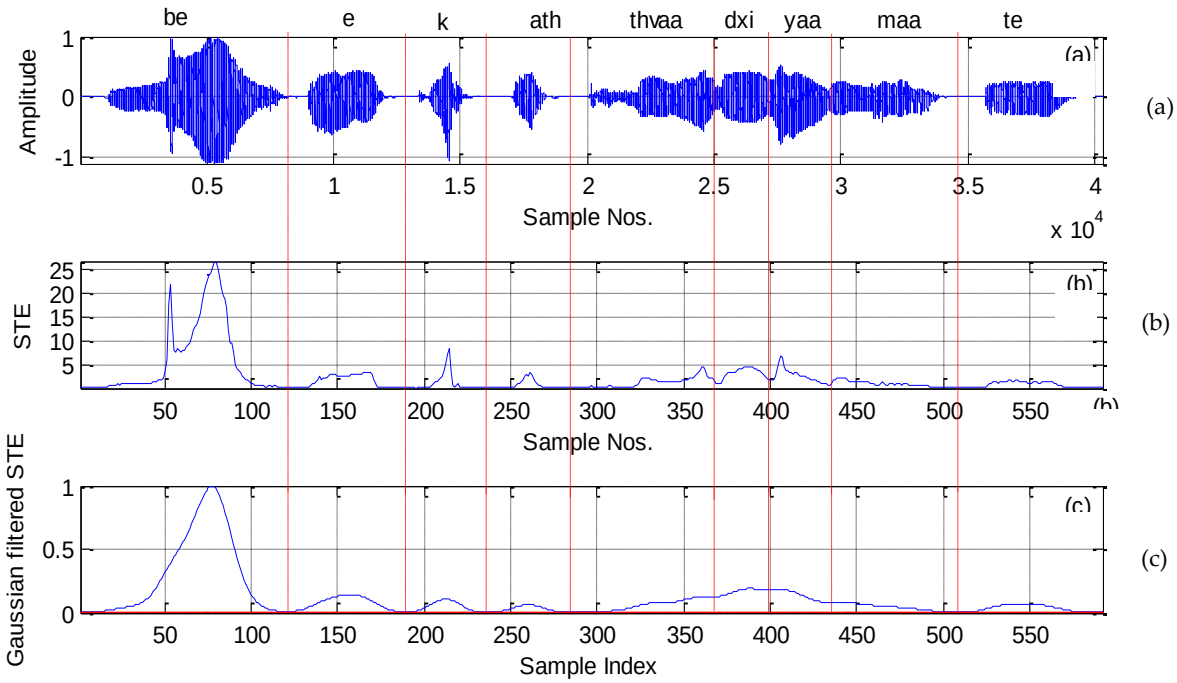


Figure 3.11: Syllable boundaries (in red color) estimated using Gaussian filter-based segmentation method for the Gujarati phrase, "*be ek athvaadxiyaa maate*" (a) speech signal, (b) corresponding short –term energy (STE) and (c) corresponding Gaussian smoothed STE.

Figure 3.11 shows the speech signal of a Gujarati phrase, "*be ek athvaadxiyaa maate*", its short-term energy, and its corresponding Gaussian filtered STE along with the estimated boundaries (indicated using red color) and labels (i.e., speech sound units at syllable-level) corresponding to the estimated boundaries. The labels assigned to the estimated segment boundaries are written after listening to them in Audacity software. The estimated boundaries are */be/ /e/ /k/ /ath/ /thvaa/ /dxi/ /yaa/ /maa/ /te/*. The actual syllables of this phrase are */be/ /ek/ /ath/ /vaa/ /dxi/ /yaa/ /maa/ /te/*.

3.5 Spectral Transition Measure (STM)-Based Segmentation

This method presents a quantitative analysis of the relation (in time) between the valleys of spectral transition measure (STM) and the syllable boundaries in fluent read speech. This method has been motivated from [23], where maximum spectral transition positions were considered as phone boundaries. In addition, STM was itself originally proposed for syllable boundary detection [23].

3.5.1 Spectral features

Spectral variation is very useful to detect syllable boundaries. This information is exploited in STM [9]. In addition, most neurons in *auditory cortex* have been found to respond poorly to *steady-state* stimuli and better for *transitional* stimuli (i.e., at spectral change) indicating key motivation for using STM to detect syllable boundaries [23]. The spectral feature used here for speech segmentation at syllable-level is Mel Frequency Cepstral Coefficients (MFCC) [24]. In speech processing, the Mel Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear discrete cosine transform of a log-power spectrum on a *nonlinear* Mel scale of frequency. The difference between the cepstrum and the Mel frequency cepstrum is that in *Mel warping*. The frequency bands are equally spaced on the Mel scale, which approximates the response of human auditory system more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of speech sound [25].

3.5.2 Segmentation algorithm using STM approach

The steps involved in segmentation are as follows:

1. Transform the speech signal into frames first using any window (such as rectangular, Hanning, Hamming etc.).
2. Then transform speech frames into a set of *10*-dimensional (i.e., *10-D*) MFCC (excluding the zero-order coefficient that represents the total signal energy).
3. Compute STM after computing MFCC. The STM, at frame, *i*, can be computed as a mean squared error (MSE) [24], i.e.,

$$\xi(m) = \frac{\sum_{i=1}^{D} a_i^2(m)}{D}, \tag{18}$$

where ξ(m) is STM at given frame m, D is the dimension of the spectral feature vector (*10* in this case) and $a_i(m)$ is the *regression coefficient* or the rate of change of the spectral feature $MFCC_i$ which is defined as [24]:

$$a_i(m) = \frac{\sum_{n=-I}^{I} MFCC_i(n+m) * n}{\sum_{n=-I}^{I} n^2}, \tag{19}$$

where n represents the frame index and I represents the number of frames (on either side of the current frame) used to compute these *regression* coefficients. We used $I=5$ corresponding to an interval of *200 ms* centered on the current frame at which the ξ value is computed for syllable boundary detection. No overlap is used for syllable boundary detection.

4. Locations of the minimas in the STM sequence gives estimated boundaries of syllable-like segments.

The MATLAB pseudo code for speech segmentation using STM approach is included in box B.3 in Appendix B. Figure 3.12 shows the speech signal of a Gujarati phrase, "*be ek athvaadxiyaa maate*", its short-term energy and its corresponding STM contour along with the estimated boundaries (indicated using red color) and labels (i.e., speech sound units at syllable-level) corresponding to the estimated boundaries. The labels assigned to the estimated segment boundaries are written after listening to them in Audacity software. The estimated boundaries are */be/ /e/ /ek/ /a/ /th/ /vaadxi/ /yaa/ /maa/ /aa/ /te/ /e/*. The actual syllables of this phrase are */be/ /ek/ /ath/ /vaa/ /dxi/ /yaa/ /maa/ /te/*.

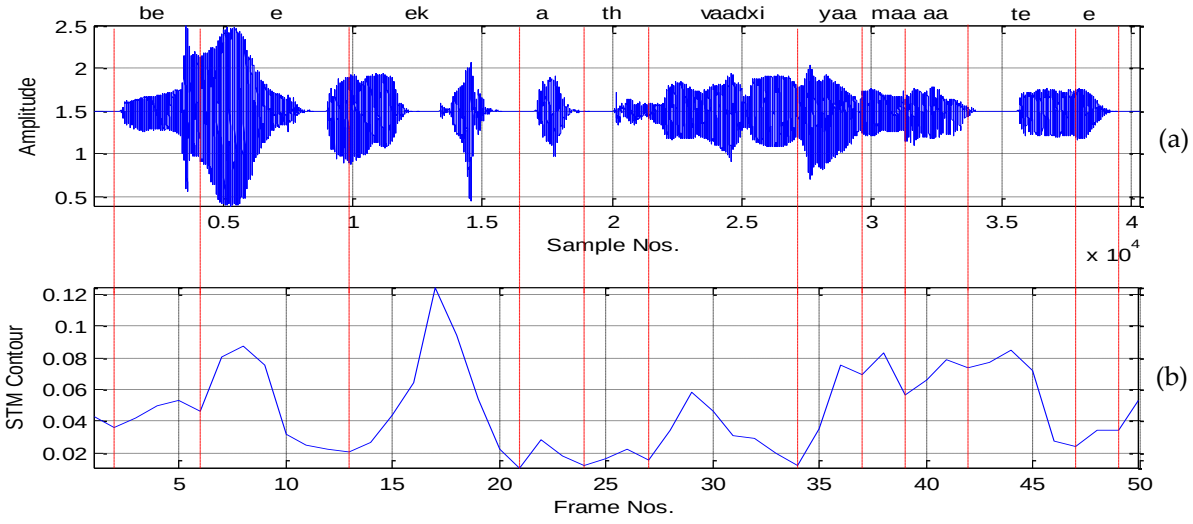


Figure 3.12: Syllable boundaries (in red color) estimated using STM-based segmentation method for the Gujarati phrase, "*be ek athvaadxiyaa maate*" (a) speech signal and (b) corresponding STM contour.

3.6 Chapter Summary

In this chapter, three different speech segmentation approaches at syllable-level has been described. Firstly, the relation between syllable boundary and short-term energy of speech signal has been described. Then the minimum-phase group delay-based segmentation approach proposed in [16] has been described. Then proposed Gaussian-based segmentation approach and spectral transition measure-based approach have been described. A Gujarati speech signal has been taken and estimated boundaries using each of these segmentation methods have been shown. In this chapter, any information regarding automatic generation of which syllable boundary correspond to which sound unit is *not* discussed. The following chapter deals with the algorithm proposed to generate the labeled data given the speech signal and corresponding text.

Chapter 4

Automatic Generation of Labeled Speech Data

4.1 Introduction

Development of unit-selection based TTS system in Gujarati (using syllable as basic speech sound unit) in festival framework requires large amount (i.e., around *8-10* hrs.) of labeled speech corpus. Basically, labeled file contains approximate information about what parts of speech correspond to which syllable or silence (i.e., SIL). Since manual labeling of speech corpus is very tedious, expensive and time-consuming, an attempt has been made to reduce the manual efforts by automatically generating the labeled files to certain accuracy and then correcting this lab file manually using either Audacity tool, WaveSurfer or any other labeling tool like DONLabel. Since automatic generation is not faultless, one has to manually check the labeled files and correct it, however, the time required in labeling and hence efforts are reduced.

4.2 Existing DONLabel Tool

Presently, there is a DONLabel labeling tool (used for syllable-based labeling tool for Indian languages) available based on minimum-phase group delay-based segmentation method (described in Chapter 3). In this tool, one needs to set a parameter *viz.*, WSF for each utterance such that the syllables get distributed to entire regions corresponding speech signal. And then manually check the labels and then save the **.lab* file. Screenshot of the tool at different stages are given in Figure 4.1, 4.2 and 4.3.

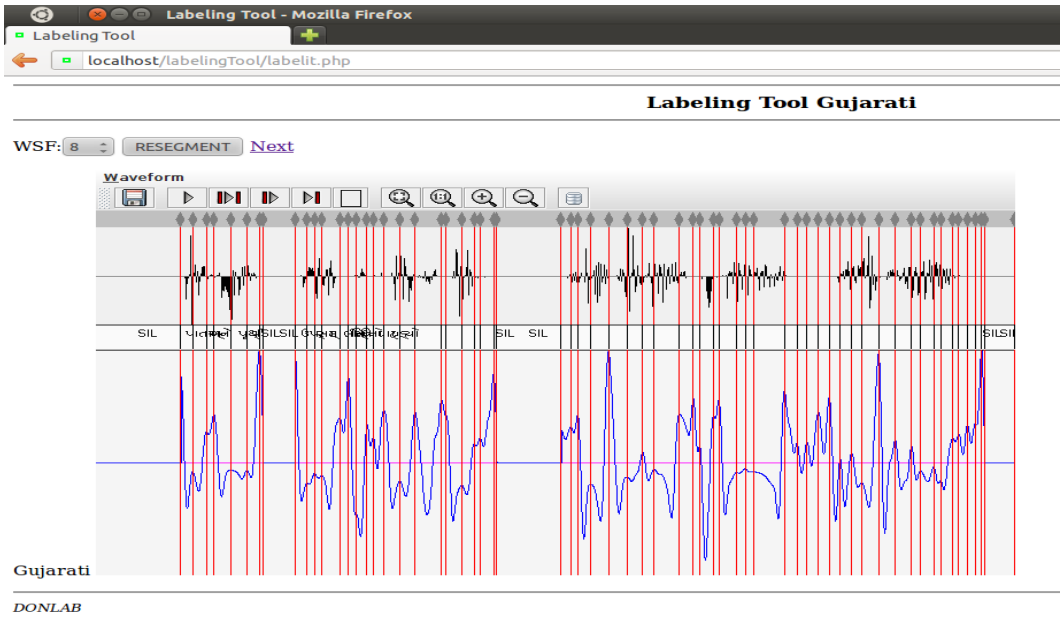


Figure 4.1: Screenshot of DONLabel labeling tool for a Gujarati utterance before setting WSF.

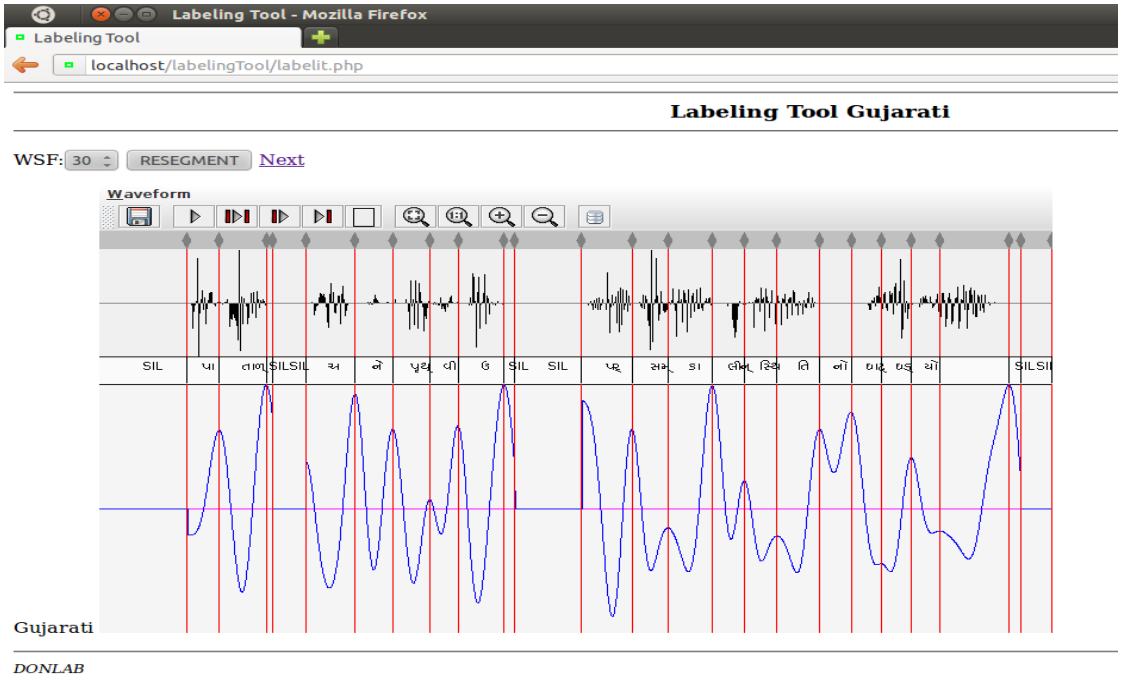


Figure 4.2: Screenshot of DONLabel labeling tool for a Gujarati utterance after setting WSF.

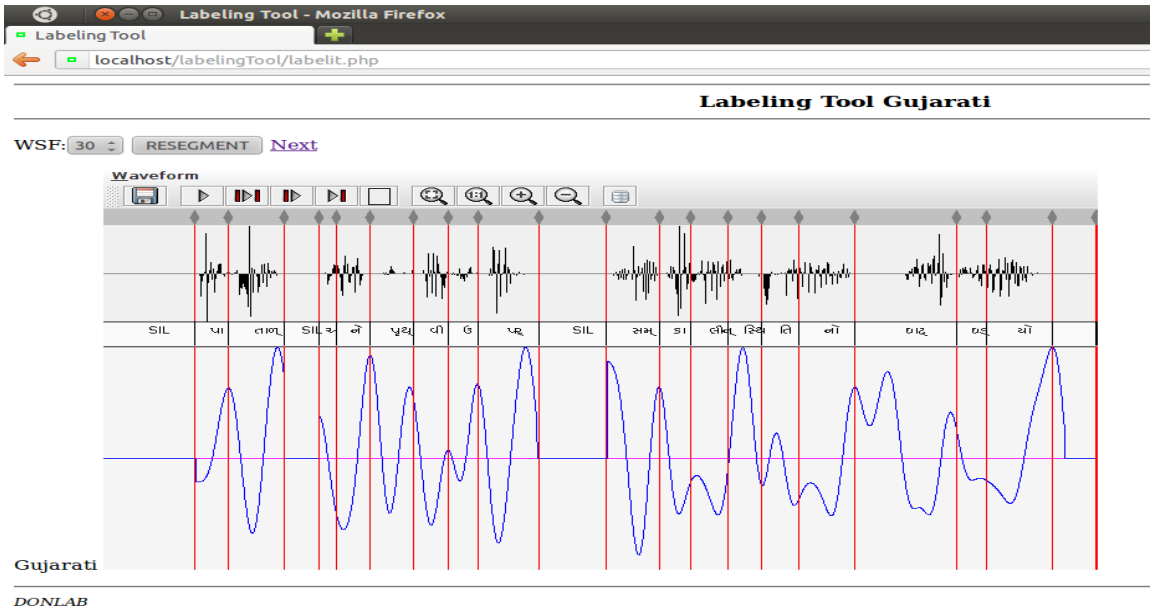


Figure 4.3: Screenshot of DONLabel labeling tool for a Gujarati utterance after manually correcting the labeling.

4.3 Proposed Labeling Algorithm

Algorithm for automatic generation of labeled files given the speech corpus and corresponding text has been developed. Given a file containing all text and a folder containing corresponding recorded speech, output of this algorithm will be the labeled files in a folder.

The key steps involved in the algorithm are silence and speech segments detection, detecting syllable boundaries (in *sec*) for each speech segments using any of the speech segmentation algorithms (such as group delay-based, STM-based or Gaussian filter-based as described in Chapter 3), making number of syllables detected equal to number of syllables present if not equal by inserting or deleting the syllable boundaries and then writing the data to labeled file.

Following is the flowchart showing the key steps required in labeled file generation:

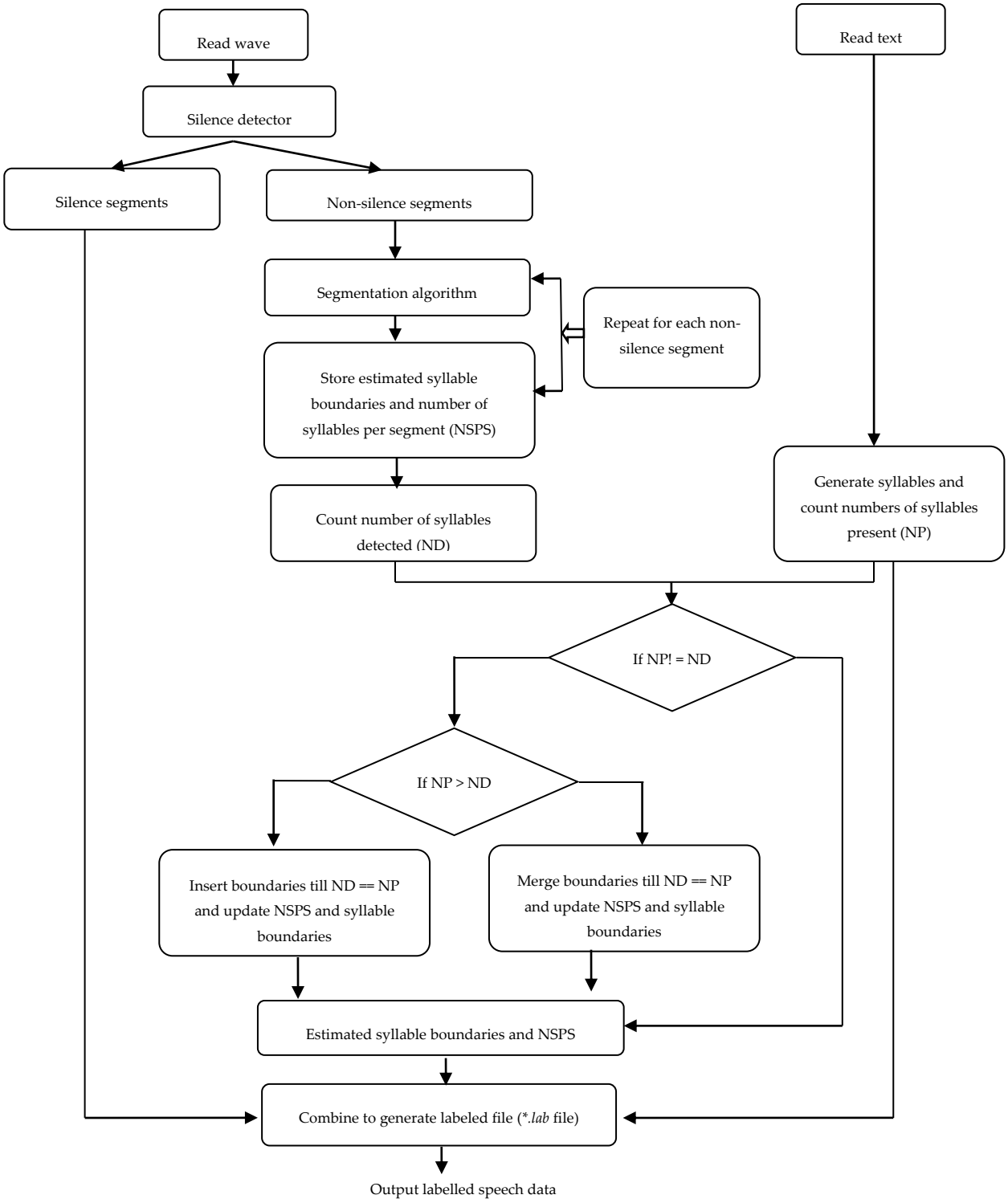


Figure 4.4: Flowchart showing the steps of labeled file generation algorithm

4.3.1 Silence Detection in Speech Signal

Recorded speech signal contains both *silence* and *non-silence* (speech) part. The word *silence* refers to any absence of information other than speech. Since labeled file also contains the information regarding which part of speech signal contains silence, therefore one need to first extract the information of silence segments from the speech signal. In order to achieve this, following algorithm [26] is used.

- Speech signal is first broken into non-overlapping frames of *30 ms* length. Then for each frame, the two features described below are calculated, leading to two feature sequences for the entire speech signal
  - Signal Energy – Let $s_i(n), n = 1\ to\ N$ the speech samples of the i^{th} frame of length N. Then, for each frame i the energy is defined as

$$E(i) = \frac{1}{N}\sum_{n=1}^{N}|s_i(n)|^2. \tag{20}$$

This simple feature can be used for detecting silent periods in speech signals, and also for discriminating between speech classes.

  - Spectral centroid - The spectral centroid, C_i, of the i^{th} frame is defined as

$$C_i = \frac{\sum_{n=1}^{N}(n+1)S_i(n)}{\sum_{n=1}^{N}S_i(n)}, \tag{21}$$

where $S_i(n), n = 1\ to\ N$, is the Discrete Fourier Transform (DFT) coefficients of the i^{th} short-term frame, where N is the frame length. This feature is a measure of the *spectral position*, with high values corresponding to *brighter* sounds.

The reasons for selection of these particular features are (i) for simple cases, the energy of the voiced segments is larger than the energy of the silent segments. (ii) if unvoiced segments simply contain environmental sounds, then the spectral centroid for the voiced segments is again larger, since these noisy sounds tend to have lower frequencies and therefore the spectral centroid values are lower.

- Two thresholds are dynamically estimated for each of the feature sequence.
- A simple thresholding criterion is applied on the sequences.

- Speech segments are detected based on the above criterion and the remaining segments will correspond to silence part.

Figure 4.5 shows the speech signal with detected non-silence speech segment indicated with red color and silence segment with gray color.

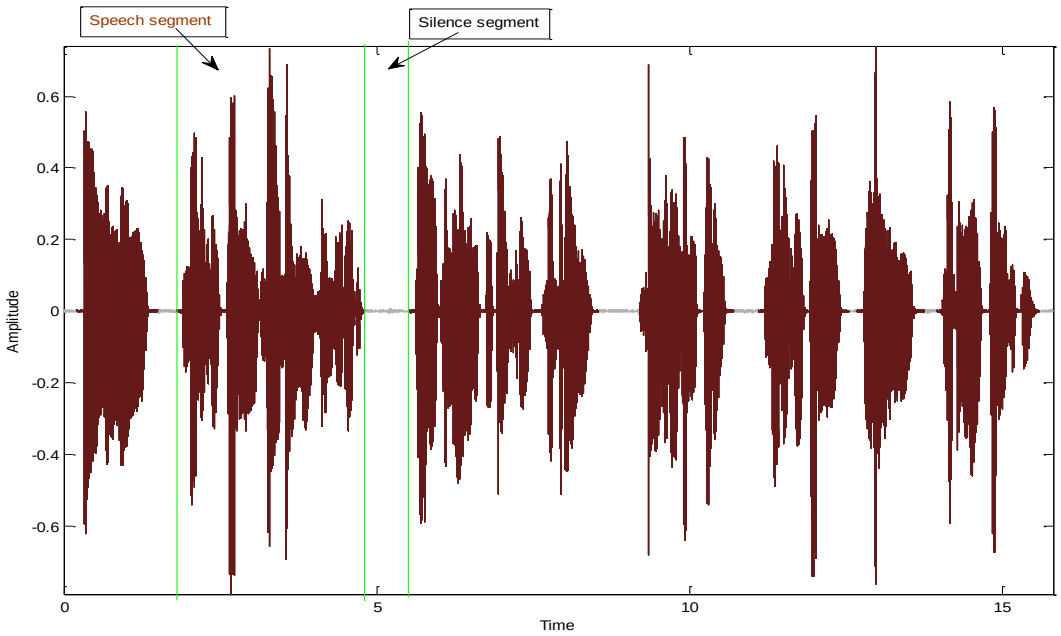


Figure 4.5: Speech signal with automatically detected non-silence speech segment indicated with red color.

4.4 Labeled File Generation Algorithm

The key steps followed in lab file generation are as follows:

1. Copy all the text to a file and a folder containing corresponding recorded speech. File containing text will be of the structure shown in figure 4.6 and the corresponding **.wav* file will be stored in a folder with name *text_00001.wav, text_00002.wav* and so on.
2. Syllables are generated one-by-one for all the text lines of the above text file using pronunciation rules and saved into another folder by the name *text_00001.tx*t, *text_00002.txt* and so on, with each file containing syllables of the corresponding text line.

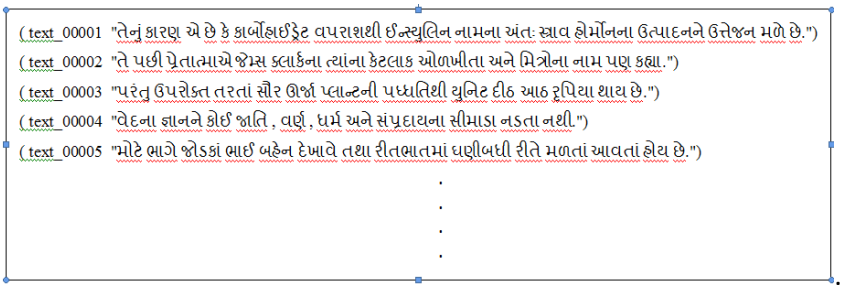


Figure 4.6: Text file structure

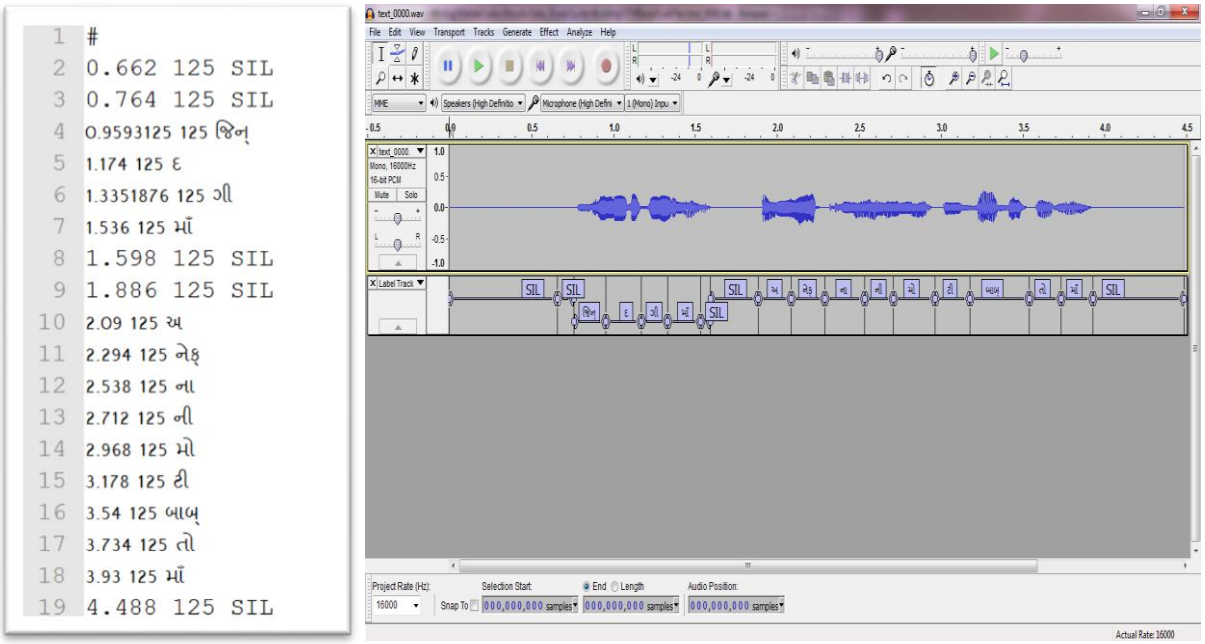
3. Repeat the step *4* to step *14* for all the files whose **.lab* files you want to generate.
4. Read **.wav* file and corresponding **.txt* file one-by-one.
5. Syllables from text file are read into an array which contains unicode of each syllable.
6. Each syllable is separated by space in the text file so space is used as a *delimiter* in identifying two syllables.
7. Number of syllables present in the text is calculated and space index is also stored.
8. Speech signal is passed to silence detection algorithm and segments containing non-silence part of speech is stored into an array.
9. These non-silence segments are further passed through the segmentation algorithm and finally number of syllables detected per segment is stored. This step is repeated for all the detected non-silence segments.
10. Total number of syllables detected for the given **.wav* file are calculated by adding number of syllables detected per segment.
11. If number of syllables detected is found to be less than number of syllables present, then maximum syllable duration boundary is detected and a boundary is inserted in between that speech segment. This process is repeated until number of syllables detected becomes equal to number of syllables present.
12. If number of syllables detected is found to be more than number of syllables present, then required number of boundaries are deleted by merging the segment which is of least duration with its immediate right or left segment

depending on which one is of smaller duration. This process is also repeated until number of syllables detected becomes equal to number of syllables present.

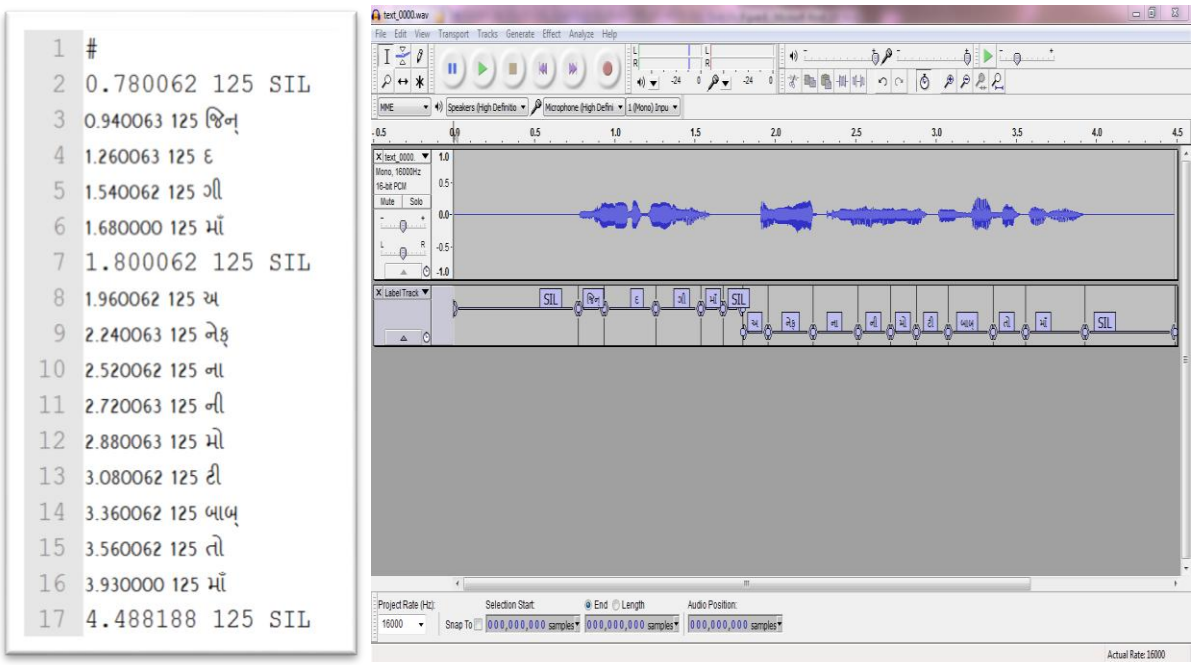
13. If number of syllables detected is found to be equal to number of syllables present obtained in step *5* above, then the **.lab* file generation steps were executed.

In the **.lab* file generation process, syllables, detected syllable boundary (in time), silence information obtained from silence detection algorithm, all are merged together to generate **.lab* file containing time information of presence of each syllable or silence in speech waveform. The pseudo code for the **.lab* file generation algorithm is included in box B.4 in Appendix.

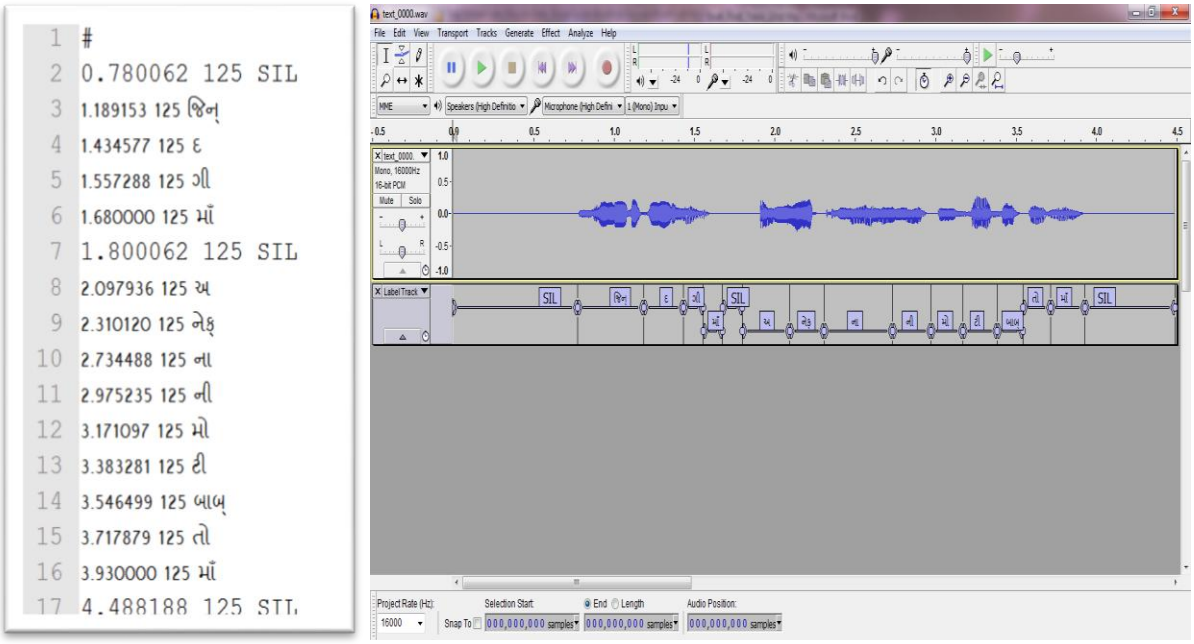
Screenshot of the *lab* file in text format and the *lab* file when opened with Audacity software is shown in Figure 4.7. Generated **.lab* file along with the wave file can be opened in the DONLabel tool (or Audacity or WaveSurfer tool) and can be corrected manually. The **.lab* files shown in Figure 4.7(a), (b) and (c) are obtained using group delay-based segmentation approach, STM-based segmentation approach and Gaussian-based segmentation approach, respectively.



(a)



(b)



(c)

Figure 4.7: Figure showing screenshot of the **.lab* **file in Festival format (on the left side) and screenshot of audacity with **.lab*** **file and corresponding **.wav*** **file (on the right side) obtained using (a) Group delay-based segmentation approach and (b) STM-based segmentation approach and (c) Gaussian-based segmentation approach.**

4.5 Chapter Summary

This chapter describes the complete process of **.lab* file generation. The process is mainly divided into three parts. First is getting syllables from text, second is estimating syllable boundaries for non-silence parts of corresponding speech utterance using any of the segmentation algorithm described in chapter *3* and third is to combine the estimated syllable boundaries, syllables and silence information and write **.lab* file. The following chapter deals with the experimental results and discussion for the performance evaluation of speech data labeling for building TTS system in Gujarati language.

Chapter 5

Experimental Results

5.1 Introduction

In this chapter, the performance evaluation of speech data labeling for Gujarati language has been carried out on group delay-based method, STM approach and proposed Gaussian- based method. Percentage of correctness is evaluated based on the total number of *insertions* and *deletions* made during the task of labeling the speech data. It has been observed that the percentage of correctness is more for the proposed Gaussian-based method. In addition to this, the intelligibility of the TTS system is evaluated for all these three approaches. The evaluation is done by a visually challenged subject in terms of the word error rate. It has been found that the there is a decrease in error rate for the proposed method.

5.2 Speech Corpus

The speech corpus used in this thesis for analysis was the studio recorded Gujarati database of a female speaker and a male speaker. All the experiments were conducted on the same Gujarati database of *38 minutes* covering *5350 syllables* for the voice of both female and male speaker separately (i.e., same text recorded by two speakers).

5.3 Optimal Determination of Parameters

In order to determine the optimum value of window length, window overlap and Gaussian filter cutoff frequency in Gaussian-based segmentation approach, segment boundaries were estimated for speech corpus with different parameters. Table 5.1 presents boundary counts and percentages for the automatic syllable boundary detection experiment with *20* % of syllable boundary agreement using Gaussian-

based segmentation approach. The manually derived boundaries are represented as (manual) and the automatically detected boundaries are represented as (automatic).

Table 5.1: Automatic syllable boundary detection results for various parameters using Gaussian-based segmentation approach

Window Length *(ms)*	Window Overlap *(ms)*	Filter B.W. *(Hz)*	Total (Manual)	Detected (Automatic)		Correctly Detected (Automatic)		% Accuracy		% Over segmentation	
				Female	Male	Female	Male	Female	Male	Female	Male
9.375	5	65	5350	4514	4472	2845	2901	53.17	54.22	-18.52	-19.63
9.375	5	70	5350	4729	4615	2988	3022	55.85	56.48	-13.13	-15.92
9.275	5	73	5350	4852	4676	3048	3064	56.97	57.27	-10.26	-14.41
8.75	5.3125	70	5350	5503	5154	3334	3265	62.31	61.02	2.78	-3.80

Observation – For a given window length, window overlap and filter B.W., percentage of detected and correctly detected boundaries are almost similar for both female and male voice. As the filter BW increase, the number of spurious boundaries also increases. As the number of correctly detected boundaries increase, number of wrongly detected boundaries also increases. Hence, considering the trade-off between the two, window length of *9.375 ms*, window overlap of *5 ms* and filter BW of *70 Hz* has been considered as optimum and used in generating **.lab* files for TTS system building.

In order to determine the optimum value of frame size and number of frames on each side of current frame (i.e., *I*) for STM-based segmentation approach, segment boundaries were estimated for speech corpus with different parameters. Table 5.2 presents boundary counts and percentages for the automatic syllable boundary detection experiment with *20 %* of syllable boundary agreement using STM-based segmentation approach. The manually derived boundaries are represented as (manual) and the automatically detected boundaries are represented as (automatic).

Table 5.2: Automatic syllable boundary detection results for various parameters using STM-based segmentation approach

Frame size (ms)	I	Total (Manual)	Detected (Automatic)		Correctly Detected (Automatic)		% Accuracy		% Over segmentation	
			Female	Male	Female	Male	Female	Male	Female	Male
50	3	5350	7180	6688	2377	2190	44.42	40.93	25.48	20.00
50	4	5350	6438	6034	2120	1998	39.62	37.34	16.89	11.33
50	5	5350	6032	5579	2092	1917	39.10	35.83	11.30	4.10
40	4	5350	7600	7095	2440	2254	45.60	42.13	29.60	24.59
40	3	5350	8533	7974	2706	2492	50.57	46.57	37.30	32.90

Observation: For a given frame size and *I*, percentage of detected and correctly detected boundaries are almost similar for both female and male voice. It has been observed as the frame size or *I* reduce percentage of detection accuracy increase. However, at the same time over-segmentation also increases. So considering the tradeoff between the two, frame size of *50 ms* and *I=4* has been considered as optimum and used in generating **.lab* files for TTS building.

It can be observed from Table 5.1 and Table 5.2 that there is negligible over-segmentation in case of Gaussian-based segmentation approach whereas there is high over-segmentation in case of STM-based approach. In the case of STM-based speech segmentation approach, for getting *50%* of correctly detected boundaries, there is around *70%* over-segmentation. Whereas, there is no over-segmentation in case of Gaussian-based speech segmentation approach for getting same percentage of result.

5.4 Performance Evaluation of Labeling for Gujarati Language

Testing has been conducted using a set of *55* test sentences of Gujarati speech corpus for each of the three labeling algorithms for both male as well as female recorded speech and the percentage of correctness has been calculated based on the following formula.

$$Percentage\ of\ Correctness = \left[1 - \frac{(No.of\ insertions + No.of\ deletions)}{Total\ no.of\ segments}\right] \times 100. \qquad (22)$$

The calculation for minimum-phase group-delay based segmentation has been done using the DONLabel tool with the optimal WSF.

The calculation for Gaussian-based segmentation has been done using the optimum window length (i.e., *9.375 ms*), window overlap (i.e., *5 ms*) for STE computation and optimum cut-off frequency (i.e., *70 Hz*) and filter length of Gaussian filter. The calculation for STM-based segmentation has been done using the optimum frame size (i.e., *50 ms*) and *I (*i.e., *4)*. Table 5.3 shows the percentage of correctness of different segmentation algorithms for Gujarati language.

Table 5.3: Percentage of correctness of different segmentation method for Gujarati language

Segmentation Method	Percentage of Correctness	
	Female	**Male**
Minimum-phase group-delay based segmentation	68.64 %	71.05 %
Proposed Gaussian-based segmentation	**81.24 %**	**84.31 %**
STM-based segmentation	77.25 %	80.20 %

It can be observed from Table 5.3 that the percentage of correctness is almost similar for both female and male voice as well as it is the highest in case of Gaussian-based segmentation algorithm. These results demonstrate the effectiveness of the proposed algorithm for speech segmentation at syllable level.

5.5 Intelligibility Evaluation of TTS Systems

Six different Gujarati TTS systems, three each for female and male voice corresponding to three different segmentation methods, have been built using Gujarati database of *38 min* consisting of *184 utterances* as speech corpus covering *5350 syllables*. TTS system 1 has been built using **.lab* files generated by already

existing DONLabel tool which gives lab files using minimum-phase group delay-based segmentation algorithm. For generating **.lab* files corresponding to each utterance using DONLabel, *WSF* needs to be set manually for each utterance. TTS system *2* have been built using **.lab* files generated by using lab file generation step described in Section 3 above and proposed STM-based segmentation approach described in Section 2 above. TTS system 3 have been built using **.lab* files generated by using lab file generation step described in Section 3 above and proposed Gaussian-based segmentation approach described in Section 2 above.

For evaluating and comparing the speech intelligibility of the three TTS systems, *65 words* have been synthesized using all the three TTS system. These *65 words* have been taken from different positions of utterances used in speech corpus. Three folders, *viz.,* folder1, folder2 and folder3 each for female and male voice, containing *65* synthesized words from TTS system *1*, *2* and *3*, respectively have been given to a visually challenged subject and she was asked to open each folder one-by-one and make *6* text files for each of the six folders in order to write the transcription of each of the synthesized word after hearing it in that text files.

Table C.1 and Table C.2 in Appendix C shows the word list with the corresponding transcription of each of the synthesized word written by the subject from each of the three systems corresponding to female voice and male voice, respectively. Remark indicates whether the word is Partially Correct (PC), Correct (C), Correct with extra component at front (CEF), Correct with extra component at back (CEB) and Wrong (W).

Table 5.4 indicates the intelligibility of six TTS systems (as evaluated by a visually challenged subject). It can be observed from Table 6 that the system built using Gaussian-based segmentation has highest intelligibility for both female and male voice as compared to the two TTS systems built using minimum-phase group delay-based segmentation and STM-based segmentation.

Table 5.4: Intelligibility Evaluation of different TTS system

Percent	Group delay-based		STM-based		Gaussian-based	
	Female	Male	Female	Male	Female	Male
Correct Words Synthesized	16.92 %	18.46 %	10.77 %	18.53 %	23.07%	21.53%
Correct Words Synthesize with extra component at front	3.07 %	4.61 %	4.61 %	8.69 %	0%	7.69%
Correct Words Synthesize with extra component at back	6.15 %	1.53 %	4.61 %	0 %	9.231%	0%
Correct Words Synthesize with extra component at both front and back	0 %	0 %	3.07 %	0 %	0%	0%
Partially correct words synthesized	46.15 %	50.76 %	49.23 %	47.15 %	50.77%	49.23%
Wrong words synthesized	27.69 %	24.46 %	27.69 %	25.61 %	16.92%	21.53%

5.6 Limitations

The possible cases where failure may occur are (i) at the *fricative* segments where the energy of the fricative is quite high and (ii) at the *trills* and *nasals* segments as they are highly transient sounds.

5.6.1 Presence of fricatives

If a fricative is present in a speech signal and we compute the energy function, we may get a peak at those segments (as fricatives have high frequency). This will be manifested in Gaussian filtered STE which will result in *spurious* syllable boundary. Table 5.5 shows some of the words consisting of fricatives in which spurious boundaries were detected during segmentation.

Table 5.5: Speech signal of words with spurious boundaries

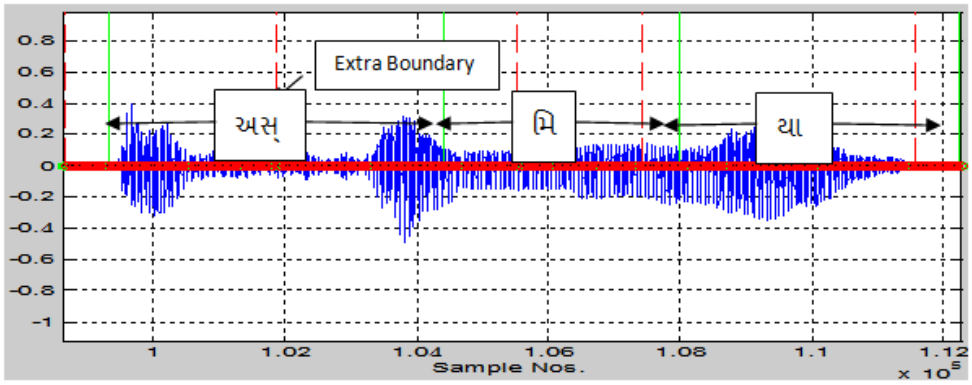
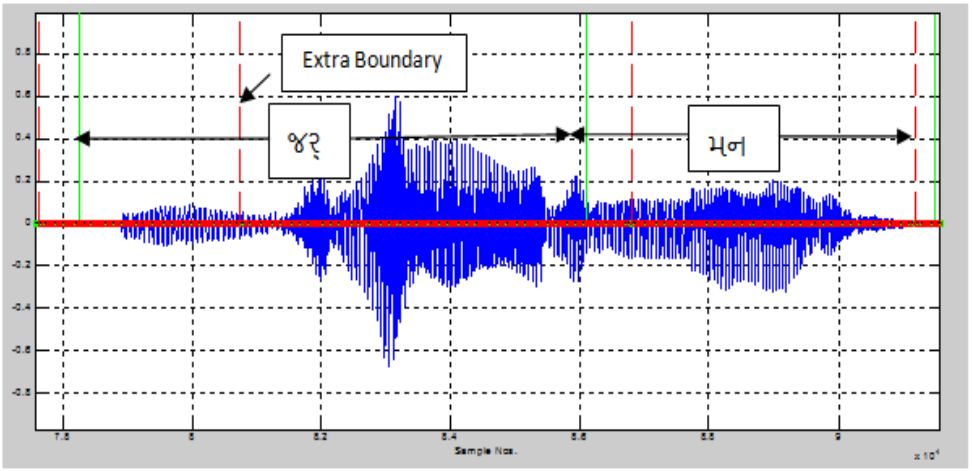
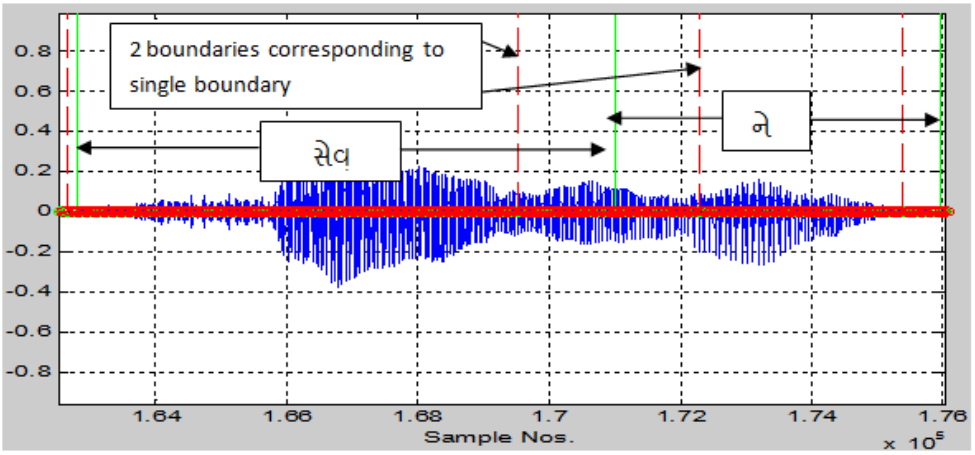
Word	Speech Signal with Expected (marked green) and Estimated (marked red) Boundaries
અસમિયા	
જર્મન	
સેવને	

Table 5.5: (Continued)

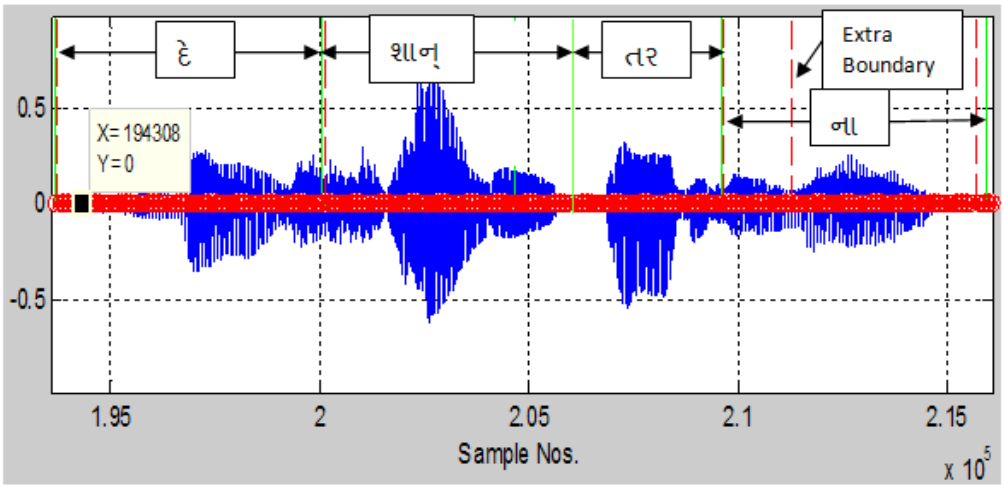
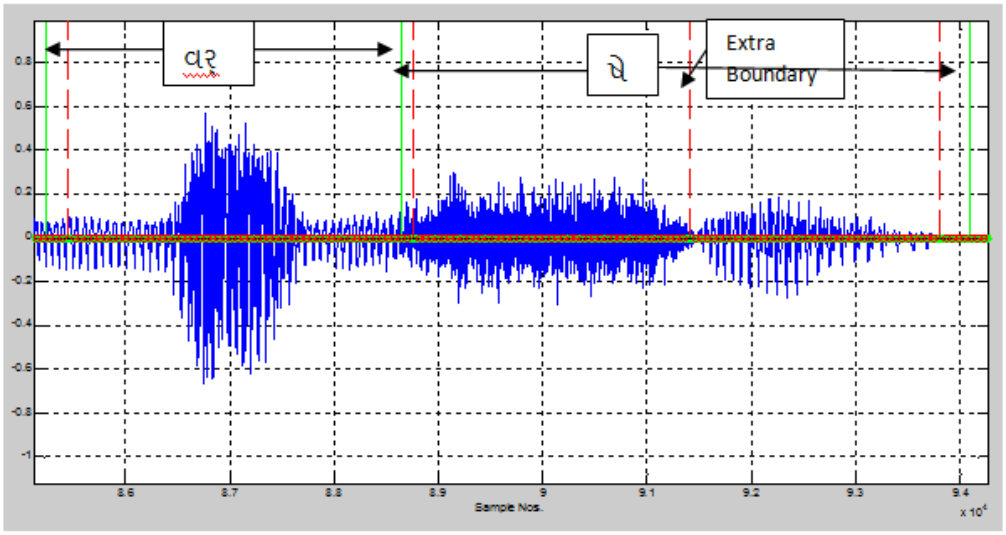
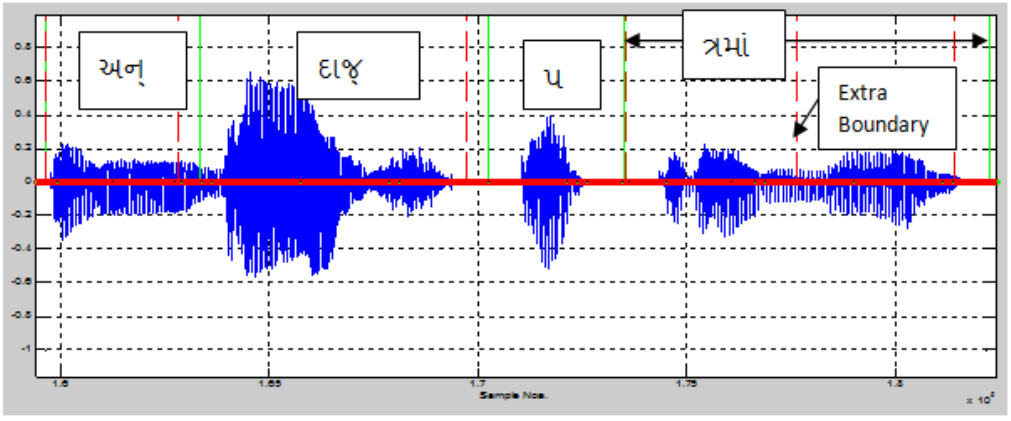
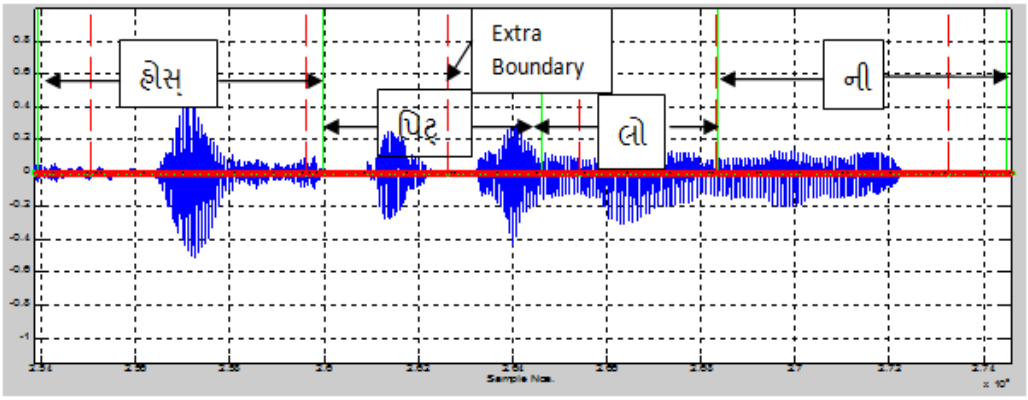
દેશાંતરના	
વર્ષે	
અંદાજપત્રમાં	

Table 5.5: (Continued)

હોસ્પિટલોની	

5.6.2 Presence of trills and nasal sounds

If a *trill* or *nasal* is present in a speech signal and we compute the energy function, we may miss a peak at those segments as these are *highly transient* sounds. This will be manifested in group delay domain as well as in Gaussian filtered STE which will result in missing of syllable boundaries. Table 5.6 shows some of the words consisting of trills and nasals in which boundaries were missed during segmentation.

Table 5.6: Speech signal of words with missed boundaries

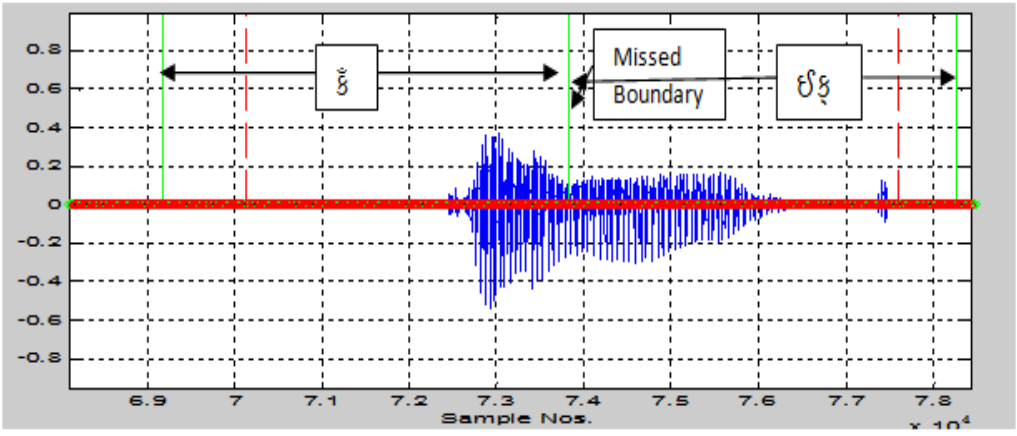
Word	**Speech Signal with Expected (marked green) and Estimated (marked red) Boundaries**
કંઈક	

Table 5.6: (Continued)

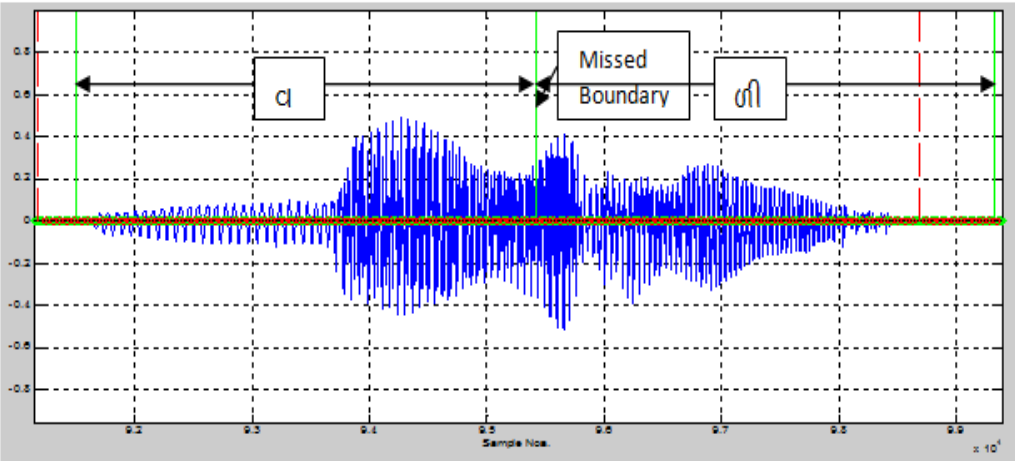
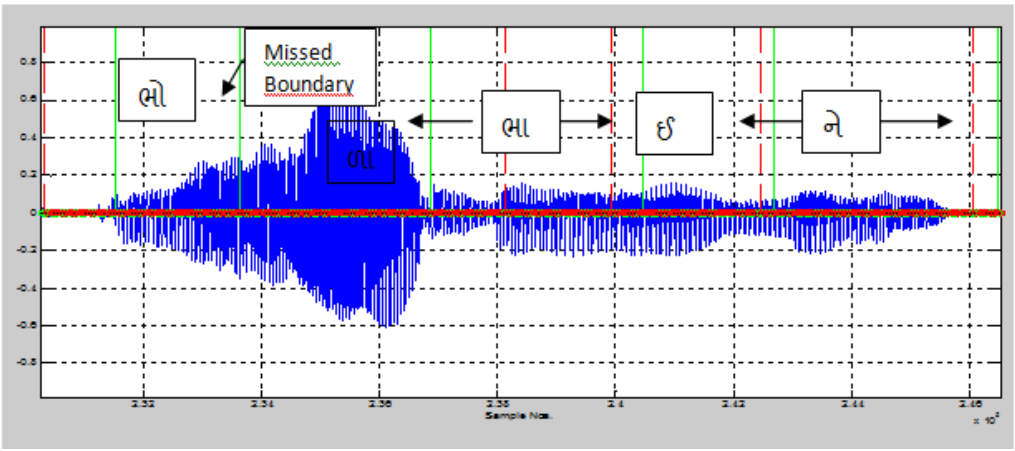
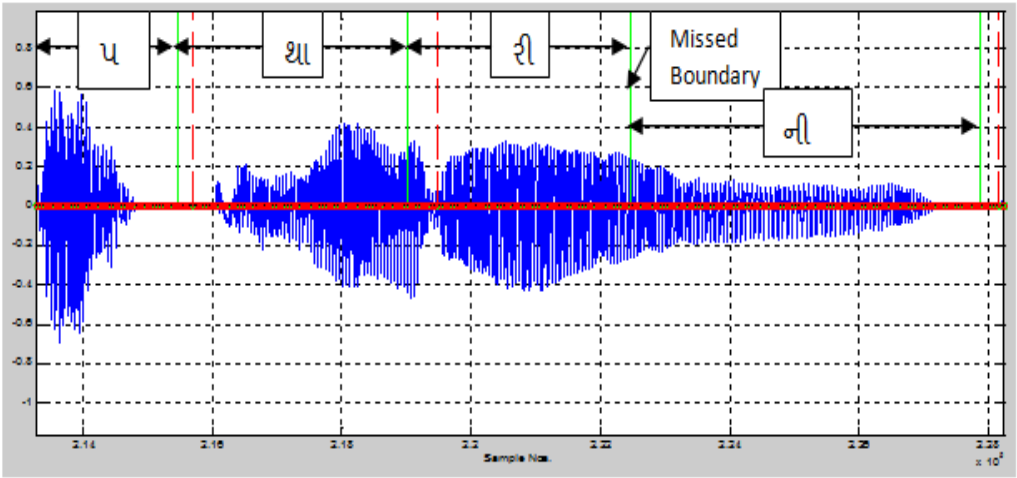
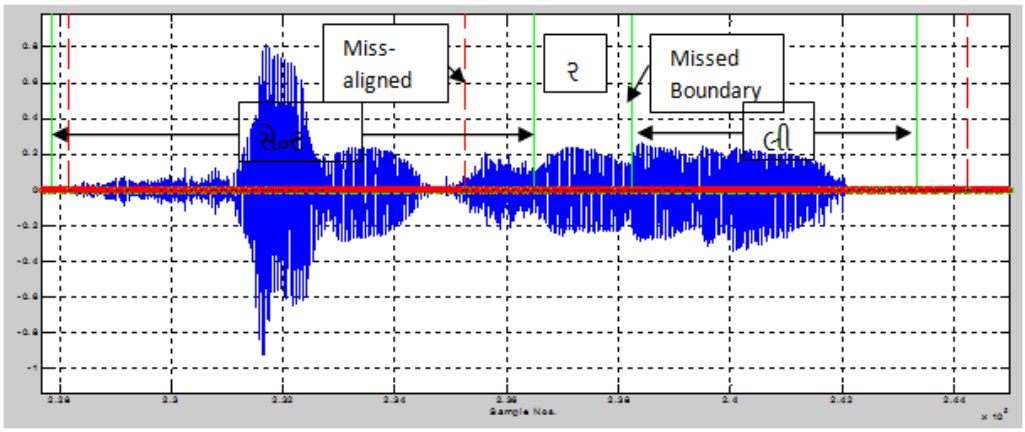
વળી	
ભોળાભાઈને	
પથારીની	

Table 5.6: (Continued)

સેન્ટ્રલી	

5.7 Chapter Summary

In this chapter, various experimental results have been presented. Firstly, results have been reported for optimum parameters selection for both Gaussian-based and STM-based segmentation. After selecting set of optimum parameters, performance of the speech data labeling using three segmentation algorithms (as discussed in Chapter 3) has been evaluated. TTS systems have also been built using **.lab* files generated using these three segmentation methods and intelligibility of TTS voice has been evaluated. It has been observed that Gaussian-based segmentation performs better than other two segmentation methods. Limitations of the Gaussian-based segmentation have also been discussed in this chapter. In the next chapter, summary and conclusion of the complete work has been presented along with the possible future directions in this work.

Chapter 6

Summary and Conclusions

6.1 Summary of the Work

There were two main objectives of this thesis. First objective was to explore signal processing techniques to *automate* labeling of speech corpus at syllable-level which is required while building unit-selection–based (by taking syllable as speech sound unit) TTS system in Gujarati. Second objective was to build TTS systems using the automatic generated **.lab* files using the three techniques of segmentation, *viz.,* minimum-phase group delay-based segmentation, Gaussian-based segmentation, STM-based segmentation and compare these three TTS systems in terms of word error rate. A speech corpus recorded in both male and female voice has been taken and various experiments have been conducted on the same. Firstly, optimum parameters for Gaussian-based and STM-based segmentation method for syllable boundary detection were determined. This was done by observing the number of boundaries correctly detected for different set of parameters and then optimum parameters were chosen. Then **.lab* files were generated using selected optimum set of parameters for the Gaussian and STM-based method and existing minimum-phase group delay-based method (with the help of DONLabel labeling tool) and corresponding TTS systems were built for both female and male voice. These TTS systems were then evaluated for their intelligibility. It has been observed that the intelligibility (i.e., percentage of words correctly synthesized) is highest in case of Gaussian-based segmentation approach and least in case of group delay-based segmentation approach. In addition, group delay-based segmentation algorithm is a semiautomatic segmentation algorithm as one needs to set the *WSF* parameter for each utterance manually. However, Gaussian-based segmentation approach is an automatic segmentation approach. It has also been observed that stops, fricatives, glides and nasals are difficult phonemes for segmentation task at syllable level. While

evaluating the TTS it has been observed that missing or one detecting one extra boundary at wrong location result in complete shift of the text in the **.lab* file. Gaussian-based TTS system results in less word error rate as compared to other two TTS systems. It has also been observed that as the training corpus for building TTS system is increased, the word error rate increase as the intelligibility of TTS system decreases because the probability of occurrence of wrong labeling increase.

6.2 Future Research Directions

In building TTS systems using Gaussian-based segmentation approach, since the text (and hence, the number of syllables present) is known for an utterance to be labeled, one can use this information to set the optimum parameters automatically for each of the utterance. This may improve the labeling performance. Also, since we know the case where boundaries are missed or spurious boundaries are detected Gaussian-based approach, so by any way if one can combine this text information and information about detected boundaries then this may further improve segmentation performance and hence the may improve the quality of synthesized voice. In this thesis, optimal parameter detection has been done considering ± 20 % of syllable boundary as the allowable error range. It is intuitive that as the allowable range will increase, TTS system performance will degrade. However, one can also experiment with the different allowable ranges and the resultant TTS performance built using these different allowable ranges.

Appendices

A: Gujarati characters with their corresponding transcription.

a	ax	aa	i	ii	u	uu	ee	ei	o	ae	ou
અ	ઑ	આ	ઇ	ઈ	ઉ	ઊ	એ	ઐ	ઓ	ઍ	ઔ
				n’	hq	mq					
				ં	ઃ	ઁ					
k	kh	g	gh	ng			c	ch	j	jh	nj
ક	ખ	ગ	ઘ	ઙ			ચ	છ	જ	ઝ	ઞ
tx	txh	dx	dxh	nx			t	th	d	dh	n
ટ	ઠ	ડ	ઢ	ણ			ત	થ	દ	ધ	ન
				p	ph	b	bh	m			
				પ	ફ	બ	ભ	મ			
		y	r	l	lx	w	sh	s	h		
		ય	ર	લ	ળ	વ	શ	સ	હ		

B: Matlab Pseudo codes

B.1: MATLAB Pseudo code for the Minimum phase group delay-based segmentation algorithm

Step I: Read the speech signal, *xn* of sampling frequency, *fs*.

Step II: Compute the STE sequence

Step a: Initialize window length, *winLen* and window overlap, *winOverlap*, type of window, *window*.

Step b: Frame and window the speech signal

```
sigFramed = buffer(xn, winLen, winOverlap, 'nodelay');
sigWindowed = diag(sparse(window))*sigFramed;
```

Step c: Calculate STE, En

```
En = sum(abs(sigWindowed).^2,1);
```

Step III: Symmetrize *En* and the resultant sequence is *El*

```
El = zeros(1,length(En)*2);
for j=1:length(En)
        El(j)= En(j);
        El(length(El)-j+1)= En(j);
end
```

Step IV: Initialize Gamma=0.01 and compute Eltilda

```
Eltilda=El.^Gamma;
```

Step V: Invert the sequence Eltilda

```
EltildaInv = 1./Eltilda;
```

Step VI: Compute root cepstrum, cntilda

```
cntilda = (ifft(EltildaInv,2048*8));
```

Step VII: Initialize WSF and compute minimum-phase signal, cn

```
n = length(En);
Nc = floor(n/WSF);
cn = emtilda(1,1:Nc);
```

Step VIII: Compute group delay function, grp_delay

```
[grp_delay , w] = grpdelay(cn,1);
```

Step IX: Obtain peaks of the group delay function.

```
t12 = linspace(0,signal_time,length(grp_delay));
maxima=zeros(size(grp_delay));
```

```
k=0;
for i= 2:length(grp_delay)-1
        if (grp_delay(i-1)<grp_delay(i) & grp_delay(i+1)<grp_delay(i))
                maxima(i)=1;
                k=k+1;
                SegmentTime(k) = t12(i);
        end
end
```

B.2: MATLAB Pseudo code for the proposed Gaussian-based segmentation algorithm

Step I: Read the speech signal, *x(n)* of sampling frequency, *fs*.

Step II: Compute the STE

Step a: Initialize window length, *winLen* and window overlap, *winOverlap*, type of window, *window*.

Step b: Frame and window the speech signal

```
sigFramed = buffer(x(n), winLen, winOverlap, 'nodelay');
sigWindowed = diag(sparse(window))*sigFramed;
```

Step c: Calculate STE, En

```
En = sum(abs(sigWindowed).^2,1);
```

Step V: Design a Gaussian–filter of *-3 dB B.W.*, *alpha* and length, *t*.

```
lpfcoeff = exp(-(alpha^2)*(t.^2);
```

Step VI: Filter the STE using Gaussian filter.

```
sigfiltered=conv(En,lpfcoeff,'same');
```

Step VII: Obtain minima's of the filtered STE.

```
t12 = linspace(0,length(x(n))/fs,length(En));
minima = zeros(size(sigfiltered));
k=0;
SegmentTime = [];
for i= 2:length(sigfiltered)-1
        if((sigfiltered(i-1)> sigfiltered(i)) & (sigfiltered(i+1)> sigfiltered(i)))
                minima(i)=1;
```

```
                    k=k+1;
                        SegmentTime(k) = t12(i);
                end
            end
```

B.3: MATLAB Pseudo code for STM-based segmentation algorithm

Step 1: Read the speech signal, *xn* of sampling frequency, *fs*.

Step 2: Compute the *10*-dimensional MFCC for the speech signal.

```
mc=melcepst(x,fs,'M',10,floor(3*log(fs)),sample,sample/shift,0,.5);
```

Step 3: Compute the regression coefficients, aim.

```
aim = [];
for m=I+1:size(mc,2)-I
                s1=0;   s2=0;
                for n=-I:I
                        s1=s1+((mc(:,n+m))*n);
                        s2=s2+n^2;
                end
        ai=s1/s2;
        aim=[aim,ai];
end
```

Step 4: Compute STM.

```
s=0;
for i=1:size(mc,1)
                s=s+aim(i,:).^2;
end
stm=s/size(mc,1);
```

Step VII: Obtain minima's of the STM.

```
        t12 = linspace(0,length(xn)/fs,length(stm));
minima = zeros(size(stm));
        k=0;      SegmentTime = [];
        for i= 2:length(stm)-1
                if((stm(i-1)> stm(i)) & (stm(i+1)> stm(i)))
```

```
                                        minima(i)=1;
                                k=k+1;
                                        SegmentTime(k) = t12(i);
                        end
                end
```

B.4: MATLAB Pseudo code for the **.lab* file generation algorithm

Step I: Read directories containing text and **.wav* files (D1 and D2 directories respectively) and also directory 3 (i.e., D3) containing output **.lab* files **.**

```
path1 = ('E:\thesis\ThesisWriting\MatlabCodes\Data\Data_9June\SyllabifiedText');
path2 = ('E:\thesis\ThesisWriting\MatlabCodes\Data\Data_9June\DONLabelData\wavfiles');
path3 = ('E:\thesis\ThesisWriting\MatlabCodes\Results\Data_9June\SystemBuilding\GaussainBased\LabFiles');

D1 = dir([path1, '\*.txt']);
D2 = dir([path2, '\*.wav']);
D3 = dir([path3, '\*.lab']);
```

Step II: Read Gujarati texts from '*SyllableText.txt'* file and write corresponding syllables (separated by space) in a text file (let it be *text_0001.txt)*

```
perl('Pronunciation_Rules.pl','SyllableText.txt');
```

Step III: Count number of syllables present (using above obtained text file)

```
%Reading syllables from text file into vector M
fileID=fopen([path1 '\' D1(MainFileIndex).name],'r', 'n', 'UTF-8');
M = fread(fileID);
fclose(fileID);
SpaceIndex=find(M==32);
NosofSyllablesPresent=length(SpaceIndex);
```

Step IV: Read corresponding wave file and store speech samples, *x* and sampling frequency, *fs*.

```
[x,fs]=wavread(filename);
```

Step V: Pass the speech samples to a silence detector and store non-silence speech segments indexes

```
[SpeechSegStartIndex, SpeechSegEndIndex] = detectVoiced(x,fs);
```

Step VI: Pass each non-silence speech segments to a segmentation algorithm and obtain syllable boundaries, NosofSyllablePerSegment and NosofSyllablesDetected.

```
for mainLoop=1:length(SpeechSegStartIndex)
    xn = x(SpeechSegStartIndex(mainLoop):SpeechSegEndIndex(mainLoop));
   TimeOffset = SpeechSegStartIndex(mainLoop)/fs;
    segmentTime = Segmentation_Algorithm_function(xn, fs, parameter1, parameter2 , ……) %% Obtain segment time by calling any of the three segmentation algorithms.
    SegmentTime1 = segmentTime + TimeOffset.*ones(size(segmentTime));
    SegmentTime2 = [SpeechSegStartIndex(mainLoop)/fs, SegmentTime1, SpeechSegEndIndex(mainLoop)/fs];
    NosofSyllablePerSegment(mainLoop) =  length(SegmentTime2)-1;
    SyllableBoundary = [ SyllableBoundary , SegmentTime2 ];
    if (mainLoop < length(SpeechSegStartIndex))
          SilenceDurations(mainLoop) = (SpeechSegStartIndex(mainLoop+1) - SpeechSegEndIndex(mainLoop))/fs;
    end
end
NosofSilences = length(SpeechSegStartIndex)+1;
NosofSyllablesDetected = length(SyllableBoundary)+1-NosofSilences;
```

Step VII: Check if NosofSyllablesPresent are not equal to NosofSyllablesDetected and if not equal then make it equal.

```
if ( NosofSyllablesPresent ~= NosofSyllablesDetected )
   EstimatedSyllableDuration = [];
   j=1;
   i=1;
    while (j<=length(NosofSyllablePerSegment) && (i+NosofSyllablePerSegment(j))<=length(SyllableBoundary))
          SyllDurSegment = diff(SyllableBoundary(i:(i+NosofSyllablePerSegment(j))));
```

```
        i=i+NosofSyllablePerSegment(j)+1;
        EstimatedSyllableDuration = [EstimatedSyllableDuration SyllDurSegment];
  end
```

Step VII(a): Insert boundaries if NosofSyllablesPresent > NosofSyllablesDetected

```
  if ( NosofSyllablesPresent > NosofSyllablesDetected )
        NosofInsertion=NosofSyllablesPresent - NosofSyllablesDetected;
        while(NosofInsertion>0)
            Error =  EstimatedSyllableDuration;
            [sortedError,sortIndex] = sort((Error),'descend');
            clear l;
            l=1;
  %%%% Search for the segment to which sortIndex(1) i.e. maximum error index belong...
   initial = 0;
   for k=1:length(NosofSyllablePerSegment)
      End(k) = NosofSyllablePerSegment(k)+initial;
      initial = End(k);
      if k==1
         Start(k) = 1;
      else
         Start(k) = End(k-1)+1;
      end
  end
  for (k=1:length(End))
      if(Start(k)<=sortIndex(l) && sortIndex(l)<=End(k))
         InsertionSegment = k;
         break;
      else
         k=k+1;
      end
   end
   silences=InsertionSegment;
   default = NosofSyllablePerSegment(1);
   k=InsertionSegment;
    j=InsertionSegment;
```

```
    InsertionIndexWithoutSil = sortIndex(l);
    InsertionIndexWithSil = sortIndex(l)+silences;
    NewSegmentDur = sortedError(l)/2;
    EstimatedSyllableDuration = [EstimatedSyllableDuration(1:sortIndex(l)-1);
NewSegmentDur;EstimatedSyllableDuration(sortIndex(l):end)];
    NosofSyllablePerSegment(j)=NosofSyllablePerSegment(j)+1;
    NewSyllableBoundary = (SyllableBoundary(sortIndex(l)+silences-1) +
SyllableBoundary(sortIndex(l)+silences))/2;
    SyllableBoundary = [SyllableBoundary(1:sortIndex(l)+silences-1) NewSyllableBoundary
SyllableBoundary(sortIndex(l)+silences:end)];
     NosofSyllablesDetected = NosofSyllablesDetected+1;
    NosofInsertion = NosofInsertion-1;
     l=l+1;
  end
    fileID = fopen('EstimatedSyllableDuration.txt','w');
   fileID = fopen('EstimatedSyllableDuration.txt','a');
   fprintf(fileID, '%f\n',EstimatedSyllableDuration);
   fclose(fileID);
 end
```

Step VII(b): Merge boundaries if NosofSyllablesDetected > NosofSyllablesPresent

```
if ( NosofSyllablesPresent < NosofSyllablesDetected )
  NosofDeletion=NosofSyllablesDetected - NosofSyllablesPresent;
  while(NosofDeletion>0)
     [MergeSegment,MergeIndex]=min(EstimatedSyllableDuration);
    %%%% Search for the segment to which Mergeindex belong...
    initial = 0;
    for k=1:length(NosofSyllablePerSegment)
       End(k) = NosofSyllablePerSegment(k)+initial;
       initial = End(k);
       if k==1
          Start(k) = 1;
       else
          Start(k) = End(k-1)+1;
       end
    end
```

```
    for (k=1:length(End))
      if(Start(k)<=MergeIndex && MergeIndex<=End(k))
        DeletionSegment = k;
        break;
      else
        k=k+1;
      end
    end
      silences=DeletionSegment;
      default = NosofSyllablePerSegment(1);
      j=DeletionSegment;
    %%% Search end
    %%% Merging the boundaries
  if (MergeIndex+1<=length(EstimatedSyllableDuration) && MergeIndex-1>=1)
    RightSeg =  EstimatedSyllableDuration(MergeIndex+1);
    LeftSeg = EstimatedSyllableDuration(MergeIndex-1);
     if (RightSeg > LeftSeg)
      %%%% Merging segment with the left boundary
      EstimatedSyllableDuration = [EstimatedSyllableDuration(1:MergeIndex-2);
EstimatedSyllableDuration(MergeIndex:end)];
      NosofSyllablePerSegment(j)=NosofSyllablePerSegment(j)-1;
      SyllableBoundary = [SyllableBoundary(1:MergeIndex-2+silences)
SyllableBoundary(MergeIndex+silences:end)];
      NosofSyllablesDetected = NosofSyllablesDetected-1;
      NosofDeletion = NosofDeletion-1;
      else
          %%%% Merging segment with the right boundary
      EstimatedSyllableDuration = [EstimatedSyllableDuration(1:MergeIndex-1);
EstimatedSyllableDuration(MergeIndex+1:end)];
      NosofSyllablePerSegment(j)=NosofSyllablePerSegment(j)-1;
      SyllableBoundary = [SyllableBoundary(1:MergeIndex-1+silences)
SyllableBoundary(MergeIndex+1+silences:end)];
      NosofSyllablesDetected = NosofSyllablesDetected-1;
      NosofDeletion = NosofDeletion-1;
     end
  elseif (MergeIndex+1>length(EstimatedSyllableDuration))
      %%% Merge with left boundary if no right boundary is present
```

```
        EstimatedSyllableDuration = [EstimatedSyllableDuration(1:MergeIndex-2);
EstimatedSyllableDuration(MergeIndex:end)];
        NosofSyllablePerSegment(j)=NosofSyllablePerSegment(j)-1;
        SyllableBoundary = [SyllableBoundary(1:MergeIndex-2+silences)
SyllableBoundary(MergeIndex+silences:end)];
        NosofSyllablesDetected = NosofSyllablesDetected-1;
        NosofDeletion = NosofDeletion-1;
   elseif (MergeIndex-1<1)
        %%% Merge with right boundary if no left boundary is present
         EstimatedSyllableDuration = [EstimatedSyllableDuration(1:MergeIndex-1);
EstimatedSyllableDuration(MergeIndex+1:end)];
        NosofSyllablePerSegment(j)=NosofSyllablePerSegment(j)-1;
        SyllableBoundary = [SyllableBoundary(1:MergeIndex-1+silences)
SyllableBoundary(MergeIndex+1+silences:end)];
        NosofSyllablesDetected = NosofSyllablesDetected-1;
        NosofDeletion = NosofDeletion-1;
       end       %%%% Merging of boundary end
   end %% while loop end


fileID = fopen('EstimatedSyllableDuration.txt','w');
fileID = fopen('EstimatedSyllableDuration.txt','a');
fprintf(fileID, '%f\n',EstimatedSyllableDuration);
fclose(fileID);
  end
end
```

Step VIII: Writing lab file in festival format portion

```
default = 0;
fID2=fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'],'w');
fID2 = fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'],'a');
fprintf(fID2, '#\n');
fprintf(fID2,'%4f 125 SIL\n', SpeechSegStartIndex(1)/fs
 fclose(fID2);
clear k;
clear l;
l=0;  k=1;
```

```
if(NosofSyllablesDetected == NosofSyllablesPresent)
   for i=1:length(SpeechSegStartIndex)
      StartIndex = find(SyllableBoundary==SpeechSegStartIndex(i)/fs);
      StartIndex = find(SyllableBoundary==SpeechSegStartIndex(i)/fs);
      StartIndex = find(SyllableBoundary==SpeechSegStartIndex(i)/fs);
      StartIndex = find(SyllableBoundary==SpeechSegStartIndex(i)/fs);
      c=1;
      StartIndex = find(SyllableBoundary==SpeechSegStartIndex(i)/fs);
      while (count~=-1)
      StartIndex = find(SyllableBoundary==SpeechSegStartIndex(i)/fs);
       if (count == 0 )
          l = l+1;
          if(l<=length(SpaceIndex) && SpaceIndex(l)-1 <=length(M) )
              Syll=M(k:SpaceIndex(l)-1);
             str1 = native2unicode(Syll,'UTF-8');
             fID2 = fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'], 'a', 'n', 'UTF-8');
             fprintf(fID2,'%4f 125 %s\n',SpeechSegEndIndex(i)/fs,str1 );
             fclose(fID2);
             k=SpaceIndex(l)+1;
          elseif (l==length(SpaceIndex)+1)
             Syll=M(SpaceIndex(l-1)+1:length(M));
             str1 = native2unicode(Syll,'UTF-8');
             fID2 = fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'], 'a', 'n', 'UTF-8');
             fprintf(fID2,'%4f 125 %s\n',SpeechSegEndIndex(i)/fs,str1);
             fclose(fID2);
       end
       count = count-1;
   else
       l=l+1;
       if(l<=length(SpaceIndex) && SpaceIndex(l)-1 <=length(M))
           Syll=M(k:SpaceIndex(l)-1);
           str1 = native2unicode(Syll,'UTF-8');
           fID2 = fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'], 'a', 'n', 'UTF-8');
           fprintf(fID2,'%4f 125 %s\n',SegmentTime(c),str1 );
           fclose(fID2);
           k=SpaceIndex(l)+1;
```

```
            elseif (l==length(SpaceIndex)+1)
                Syll=M(SpaceIndex(l-1)+1:length(M));
                str1 = native2unicode(Syll,'UTF-8');
                fID2 = fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'], 'a', 'n', 'UTF-8');
                fprintf(fID2,'%4f 125 %s\n',SpeechSegEndIndex(i)/fs,str1 );
                fclose(fID2);
            end
        count = count-1;
        default = SegmentTime(c);
        c=c+1;
    end
  end
       if(i<length(SpeechSegStartIndex))
            fID2 = fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'],'a');
            fprintf(fID2,'%4f 125 SIL\n',SpeechSegStartIndex(i+1)/fs);
           fclose(fID2);
        else
            fID2 = fopen([path3 '\' D1(MainFileIndex).name(1:end-3),'lab'],'a');
           fprintf(fID2,'%4f 125 SIL',(length(x))/fs);
          fclose(fID2);
      end
   end
 end
end
```

C: TTS Intelligibility Evaluation Results

Table C.1: Word list with the corresponding transcription of each of the synthesized words from the three TTS systems corresponding to the female voice.

Word List	TTS System1 Output Group Delay Based	Remark	TTS System2 Output STM Based	Remark	TTS System3 Output Gaussian Based	Remark
1. ન્યુક્લિયર	1. nyukli avarthi	PC	1. nyukli	PC	1. nyukliyar pa	CEB
2. બોસ્ટનમાં	2. schanmarna	W	2. bostanma	C	2. pyochis	W
3. સ્પષ્ટ	3. spashya	PC	3. aek	W	3. spash	PC
4. સત્યાવિસ	4. ael sa	W	4. sattar mail	PC	4. satyavi	PC
5. રહેવાની	5. joi j rahe	W	5. ahevani disha	PC	5. hevani	PC
6. અંધારાથી	6. aashiyone kalthi	PC	6. gherayel ra	W	6. parathi vavarayel	W
7. વિશ્વકર્મા	7. shri vishva	PC	7. vishvakarma	C	7. vishvakarma	C
8. ઉઠશે	8. udhashe	PC	8. could not understand	W	8. zalahali	W
9. આપતી	9. aavta rej	PC	9. blank, no word	W	9. aa	PC
10. જ્ઞાનેન્દ્રિય	10. pana panch g	W	10. nyakriy sathe	W	10. panch gnanya	PC
11. માર્કેટમાં	11. marketma	C	11. ketma ka	PC	11. marketma	C
12. ખૂબજ	12. tatu	W	12. khub j nana	CEB	12. khubjis	CEB
13. ઉંમરમાં	13. injena	W	13. imjem oma	PC	13. ujem aa	PC
14.નમ્રતા	14. namrata	C	14. namragarta	PC	14. namrata	C
15. દળદાર	15. naar	PC	15. tran	W	15. daldar	C
16. બ્રહ્માંડનું	16. brahmandnu	C	16. brahmandn	PC	16. brahmandnu sa	CEB
17. ઝીણાંઝીણાં	17. aazinano	PC	17. inazinano	PC	17. zina	PC
18. ટૂવૂમ્બામાં	18. umbamakma	PC	18. tuvumbama	C	18. tovoma	PC
19. ઈન્દ્રભૂમિ	19. indrabhumi	C	19. indrabhumi indrad	CEB	19. indrabhut gau	PC
20. મોહનકુમાર	20. hupreme	W	20. premi moh	PC	20. ae mohan kum	PC
21. ઇન્સ્ટિટયૂટ	21. tityu of	PC	21. nolo	W	21. tityud of	PC
22. તેથી	22. te to	PC	22. ate gna	W	22. tethi	C
23. વ્યક્તિને	23. vyaketine	C	23. nine karane	W	23. tene	PC
24. શ્રધ્ધાળુઓ	24. shradne poto	PC	24. shrahaluo	PC	24. shradhdhhauao aa	CEB
25. પિરિયડ્સથી	25. kari kakhata nathi	W	25. nakhata pir taraf	W	25. ginibuk kari nakhata thi	W
26. આશ્રમમાં	26. na aashara	PC	26. upaka	W	26. kshanma aa	PC
27. ઋગ્વેદમાં	27. unvina	W	27. mul urvama	W	27. rugma	PC
28. નેન્એન્ટેનાની	28. aanenentenani	CEF	28. aa nenentanani	CEF	28. nenet	PC
29. વાસ્તવમાં	29. vastavma	C	29. vastavama	C	29. vastavma	C
30. જાણવાની	30. chavane	PC	30. jadisha	PC	30. chho	W
31. વિદેશના	31. videshana	C	31. ish videshna	CEF	31. videshna	C
32. ગાંધીઆશ્રમ	32. seiskonmol	W	32. ghankru	PC	32. Gandhi aashram gu	CEB
33. સ્પર્ધા	33. spar	PC	33. spe	PC	33. dharamat	W
34. છુપાયેલ	34. tupai	PC	34. ta aayeli	PC	34. tama chhupa	PC
35. સ્થિતિને	35. thitine	PC	35. karane	PC	35. thabai	W
36. મંજૂરીની	36. mamanju	PC	36. mantrine	PC	36. pan manjuna	PC
37. ભૌતિકતાની	37. bhautiktani	C	37. bhautitanita	PC	37. bhautitan	PC
38. ચગાવવાનો	38. pag chagavavano	CEF	38. sangh chagao	W	38. pragavavano	PC
39. ટ્રાફિકવાળા	39. trafik jiyaras chhe	PC	39. aa trafikvala ras	CEF+CEB	39. trafivala	PC
40. હૃદયના	40. hadaynarna	PC	40. hadayna	PC	40. hadaynaaa	PC
41. થ્રિયરીને	41. gebengthiyarine	PC	41. thiae thiaerine	CEF	41. bigbengthi	W
42. ઇચ્છામાં	42. chhamakma	PC	42. aama otaa	W	42. tachchhama	PC
43. શ્રીમતી	43. shrimati j	CEB	43. shri	PC	43. shrim	PC
44. નાળિયેરનું	44. naliyernu pa	CEB	44. naliyernu	C	44. naliye	PC
45. પ્રમાણમાં	45. pramanma tuma	CEB	45. manma	PC	45. pra aanama	PC
46. મહાપંડિત	46. mahapan	PC	46. mahab	PC	46. mahapandit	C
47. સેનાનાયકોમાં	47. senanayakoma	C	47. seynayakoma	C	47. senanayakoma	C
48. ચૈતન્યશીલતાને	48. chaitanyashiltane katre	CEB	48. chaitanyashilatane mane	CEB	48. tanyashilatane panch	PC
49. ઉજવાય	49. saaay	W	49. ujava	PC	49. ujavay	C
50. જમીનમાંથી	50. ivadaaathi	PC	50. jaminmathi	C	50. jaminmathi	C
51. શ્રધ્ધાનો	51. dhardhano	PC	51. shradhdhhauo	PC	51. shradhdhhano	C
52. છબીનું	52. bane	W	52. chatpunah	PC	52. itheke	W
53. વિશ્વાસથીજ	53. vad vishe	W	53. d vis	W	53. visvasthi j	C
54. વસ્તુ	54. kimati	W	54. ti vastu kai	CEB+CEF	54. vastub	CEB
55. છતાં	55. ta	PC	55. hovay	W	55. chhata	C
56. શિક્ષણ	56. shikshan	C	56. shibhushan	PC	56. shichan	PC
57. ક્રિકેટને	57. kriketne	C	57. tane tarne	PC	57. ketne	PC
58. કેટલાંક	58. tilala	W	58. itala	PC	58. tala	W
59. વિષ	59. aeao	W	59. ayo	W	59. bi	W
60. હિસાબે	60. dhilk be	PC	60. hisai aa	PC	60. girinke be	PC
61. હરણીએ	61. ania	PC	61. tit bahenya	W	61. heaanine	PC
62. મહત્વની	62. aehanabchhe	W	62. mahatvama huta	PC	62. may vish	PC
63. નિલકંઠ	63. nibilto	PC	63. nilnil	PC	63. nilkanth	C
64. મહાપરાણે	64. khulbuchinoti vathi	W	64. mahabharati kar	PC	64. mahati re	PC
65. આધારસ્તંભ	65. aachama dharambi	PC	65. aatrane tram	PC	65. aavdha rasta	PC

Table C.2: Word list with the corresponding transcription of each of the synthesized words from the three TTS systems corresponding to the male voice.

Word List	TTS System1 Output Group Delay-Based	Remark	TTS System2 Output STM-Based	Remark	TTS System3 Output Gaussian-Based	Remark
1. ન્યુક્લિયર	1 nyukliy	PC	1. nyukliyar	C	1. nyukliyar pa	CEF
2. બોસ્ટનમાં	2 yor boste	PC	2. nyuyork bostak	PC	2. kuma	PC
3. સ્પષ્ટ	3 pash	PC	3. could not understand	W	3. pas	PC
4. સત્યાવિસ	4 sishru	W	4. satyavisvist	PC	4. satis lakh	PC
5. રહેવાની	5 aavi javani	PC	5. i	W	5. ketla ma vaat	W
6. અંધારાથી	6 anandhara	PC	6. arathithi	PC	6. anare dharaone	PC
7. વિશ્વકર્મા	7 vishvakarma	C	7. vishvakarma	C	7. pakrmada	PC
8. ઉઠશે	8 uthashe	C	8. ali uthashe	CEB	8. ae	PC
9. આપતી	9 aapvati	PC	9. aapdu	PC	9. aapi	PC
10. જ્ઞાનેન્દ્રિય	10 gnendriyasan	PC	10. gnchagnanedri	PC	10. gnachidnane	PC
11. માર્કેટમાં	11 martama	PC	11. marketyak	PC	11. marketma	C
12. ખૂબજ	12 chhva ab	W	12. vakhub j	CEB	12. khub	PC
13. ઉંમરમાં	13 umarma aam	CEF	13. u umarmukh	PC	13. umarma pan	CEF
14.નમ્રતા	14 namrata	C	14. namrata ane	CEF	14. namrata	C
15. દળદાર	15 aa dal	PC	15. artara	W	15. darata	PC
16. બ્રહ્માંડનું	16 bhramandu	PC	16. bhramandnu sarjan	CEF	16. bhraman	PC
17. ઝીણાંઝીણાં	17 zina	PC	17. a zina jam	PC	17. na zina	PC
18. ટૂવૂમ્બામાં	18 tuvombama pa	CEF	18. la tuoma	PC	18. tuombama pan	CEF
19. ઈન્દ્રભૂમિ	19 indrabhumi'	C	19. indrabhumi	C	19. indrabhumi gau	CEF
20. મોહનકુમાર	20 kahan kumarshe	PC	20. mohan	PC	20. kumar sharma	PC
21. ઇન્સ્ટિટયૂટ	21 institu	PC	21. institute	C	21. tud of te	PC
22. તેથી	22 tethi	C	22. toti	W	22. tethi	C
23. વ્યક્તિને	23 ti tine	PC	23. tkarmgat	W	23. vyaktine	C
24. શ્રધ્ધાળુઓ	24 chhek shradhdhhalu	PC	24. adhdhhaluo pano	PC	24. dhaduo pota	PC
25. પિરિયડ્સથી	25 pikhta piryayi	PC	25. piriyads thi kani	CEF	25. piriyad	PC
26. આશ્રમમાં	26 aashrama	C	26. na aashak	PC	26. ma je gan	W
27. ઋગ્વેદમાં	27 rugvedma	C	27. ularugma	PC	27. ulama	W
28. નેનએન્ટેનાની	28 aanenaentena	PC	28. nenaentenaninis	CEF	28. nen aentenani	C
29. વાસ્તવમાં	29 upma	W	29. vastavma m	CEF	29. vastma	C
30. જાણવાની	30 janvani	C	30. aejanavanst	PC	30. teni vaat	W
31. ઝિદેશના	31 jideshna	PC	31. shvideya	PC	31. zideshna	PC
32. ગાંધીઆશ્રમ	32 gangandhi aashram	CEB	32. gana Gandhiaashram	CEB	32. Gandhi aashram	C
33. સ્પર્ધા	33 spakad	PC	33. pardha hama	PC	33. sparde	PC
34. છુપાચેલ	34 paelch	W	34. aama chuksel chhe	W	34. aelchib	W
35. સ્થિતિને	35 tirtine	PC	35. atine	PC	35. aaro ne	PC
36. મંજૂરીની	36 manjurini	C	36. pan nithis	W	36. jurininunit	PC
37. ભૌતિકતાની	37 vanni bhoni	W	37. bhautiktani dod	CEF	37. hotari chat	W
38. ચગાવવાનો	38 chunchagavano	PC	38. chagavavano	C	38. chu chagavavu	PC
39. ટ્રાફિકવાળા	39 trafikmava	PC	39. aa trafikvala rasta	CEF+CEB	39. trafficvala	C
40. હ્રદયના	40 aay radaa	PC	40. radayna dhabakara	CEF	40. radayna	C
41. થિયરીને	41 thimadlthyana	PC	41. namu karine	PC	41. modal thyine	PC
42. ઇચ્છામાં	42 ikothma	PC	42. chhama yak	PC	42. aa othan	W
43. શ્રીમતી	43 shri	PC	43. srini	PC	43. shrimati	C
44. નાળિયેરનું	44 naliyernu	C	44. naliyernu	C	44. naliyer nu pan	CEF
45. પ્રમાણમાં	45 pravanma pan	PC	45. tiprama	PC	45. madmarpan	W
46. મહાપંડિત	46 kushanmaha	W	46. mahapandi	PC	46. mahapandi	PC
47. સેનાનાયકોમાં	47 tamara nayakoma	PC	47. sipsenalama	PC	47. senanayogan	PC
48. ચૈતન્યશીલતાને	48 chaitanyashiltane	C	48. chaitanyashilatane tale ae	CEF	48. tanyashilatane panch ma	PC
49. ઉજવાય	49 juvalva	W	49. ujva	PC	49. ujv	PC
50. જમીનમાંથી	50 jinnaminathi	PC	50. j jamathi	PC	50. jaminmathi	C
51. શ્રધ્ધાનો	51 chirdhdhho	PC	51. adhdhhano	PC	51. dhadhanov	PC
52. છબીનું	52 chhairman nu	W	52. chhinu dazan	PC	52. dhunu	PC
53. વિશ્વાસથીજ	53 rad vishe	W	53. al vishva	PC	53. vishvasthi j	C
54. વસ્તુ	54 tiva	W	54. kimb	W	54. vastu	C
55. છતાં	55 chhataj	CEF	55. chha aa chha	PC	55. chhachhaj	PC
56. શિક્ષણ	56 shikdhuti	PC	56. shibhusht	PC	56. shikshan	C
57. ક્રિકેટને	57 kriketbane	PC	57. kriune	PC	57. ne	PC
58. કેટલાંક	58 klu	W	58. to dhyan	W	58. thi dhyan	W
59. વિષ	59 oshe	W	59. ayo	W	59. chha	W
60. હિસાબે	60 aegramba	W	60. hi	PC	60. kleket aat	W
61. હરણીએ	61 aniyu	W	61. thay ogane	W	61. gain aek	W
62. મહત્વની	62 kut vishi	W	62. matvenis	PC	62. panghat karvil	W
63. નિલકંઠ	63 nilkanth	C	63. dinil	PC	63. nilkan	PC
64. મહાપરાણે	64 manmayane	PC	64. manmahapne	PC	64. trimakrane	PC
65. આધારસ્તંભ	65 aarpanvam	W	65. aa vadne	W	65. dhanomat	W

Bibliography

[1] Thierry Dutoit, *An Introduction to Text-to-Speech Synthesis*. Springer, 1997.

[2] Jennifer Ann Longster, "Concatenative speech synthesis: a Framework for Reducing Perceived Distortion when using the TD-PSOLA Algorithm," Bournemouth University, PhD Thesis 2003.

[3] D. Jurafsky and J.H. Martin, *Speech and Language Processing*.: Prentice Hall, 2000.

[4] Youcef Tabet and Mohamed Boughazi, "Speech Synthesis Technique. A Survey," in *7th International Workshop on Systems, Signal Processing and their Applications (WOSSPA)*, pp. 67-70, 2011.

[5] Paul Taylor, *Text-to-Speech Synthesis.:* Cambridge University Press, 2009.

[6] Mukund Jha, Rakesh Guttikonda, and Hema A. Murthy P. G. Deivapalan, "DONLabel: Automatic Labeling Tool for Indian Languages," in *The National Conference Communications (NCC)*, IIT Bombay, pp. 263-266, 2008.

[7] A. Narayan, K.B. Bellur, K.R. Krishnan, and H.A. Murthy, "Prosody modeling for syllable-based concatenative speech synthesis of Hindi and Tamil," in *The National Conference on Communications (NCC)*, IISc Banglore, pp. 1-5, 2011.

[8] A.W. Black, P. Taylor, and R. Caley. (2007) The Festival speech synthesis system. [Available Online]. http://festvox.org/festival/ {Last accessed: 17th June 2013}.

[9] Samuel Thomas, "Natural Sounding Text-to-Speech Synthesis Based on Syllable like Units," Indian Institute of Technology Madras, Madras, Master of Science Thesis 2007.

[10] S P Kishore and Alan W Black , "Unit Size in Unit-selection Speech Synthesis," in *EUROSPEECH*, Geneva, pp. 1317-1320, 2003.

[11] P. G. Deivapalan, Mukund Jha, Rakesh Guttikonda, and Hema A. Murthy, "DONLabel: An Automatic Labeling Tool for Indian Languages," in *The National Conference on*

Communcations (NCC), IIT Bombay, pp. 263-266, 2008.

[12] Samuel Thomas, T. Nagarajan, and Hema A. Murthy M. Nageshwara Rao, "Text-to-Speech Synthesis using syllable-like units," in *National Conference on Communication (NCC)*, pp. 277-280, 2005.

[13] M.V. Vinodh et al., "Using polysyllabic units for text to speech synthesis in Indian languages," in *National Conference on Communications (NCC)*, IIT Madras, Chennai, pp. 1-5, 2010.

[14] K. Sreenivasa Rao, Krishnendu Ghosh, Ramu Reddy Vempada, Sudhamay Maity N.P. Narendra, "Development of syllable-based text to speech synthesis system in Bengali," *International Journal of Speech Technology (IJST) Springer*, vol. 14, no. 3, pp. 167-181, September 2011.

[15] V. Kamakshi Prasad, T. Nagarajan, and Hema A. Murthy, "Automatic segmentation of continuous speech using minimum-phase group delay functions," *Speech Communication*, vol. 42, pp. 429-446, November 2004.

[16] Hema A. Murthy, Rajesh M. Hedge T. Nagarajan, "Segmentation of speech into syllabic-like units," in *EUROSPEECH* , GENEVA, pp. 2893-2896, 2003.

[17] Hema A. Murthy and B Yegnanarayana, "Group delay functions and its applications in speech technology," *Sadhana*, vol. 36, Part 5, pp. 745-782, October 2011.

[18] V Kamakshi Prasad and Hema A. Murthy T. Nagarajan, "Minimum-phase signal derived from root cepstrum," *IEE Electronic Letters*, vol. 39, no. 12, pp. 941-942, June 2004.

[19] Openheiem and Ronald W. Schafer Alan V, *Discrete-Time Signal Processing*.: Prentice Hall, p. 781, 2000.

[20] J. S. Lim, "Spectral Root Homomorphic Deconvolution System," in *IEEE International Conference on Acoustic, Speech and Signal Processing, ICASSP '79*, Washington, D.C, USA, pp. 409-414, 1979.

[21] Thomas F. Quatieri, "Nonlinear Measurements and Modeling Techniques," in *Discrete-Time Speech Signal Processing, Principles and Practice*. Delhi, India: Pearson Education

(Singapore) Pte. Ltd., Indian Branch, ch. 11, pp. 580-581, 2004.

[22] S. Mallat, *A Wavelet Tour of Signal Processing*.: Academic Press, 1999.

[23] Sadoki Furui, "On the role of spectral transition for speech perception," *J. Acoust. Soc. Am.*, vol. 80, no. 4, pp. 1016-1025, October 1986.

[24] Sorin Dusan and Lawrence Rabiner, "On the relation between maximum spectral transition positions and phone boundaries," in *INTERSPEECH'06*, Pittsburgh, USA, pp. 1-4, 2006.

[25] Hemant Patil, "Speaker Recognition in Indian Languages: A Feature Based Approach," Indian Institute of Technology Kharagpur (IIT-K), Ph.D. Thesis July 2006.

[26] Theodoros Giannakopoulos. (2013, February) www.mathwork.in. [Available Online]. http://www.mathworks.in/matlabcentral/fileexchange/28826-silence-removal-in-speech-signals/content/detectVoiced.m {Last accessed: 17th June 2013}.

Publications from the Thesis

1. Swati Talesara, Hemant A. Patil, Tanvina Patel, Hardik Sailor and Nirmesh Shah, "A Novel Gaussian Filter-based Automatic Labeling of Speech Data for TTS System in Gujarati Language" in *International Conference on Asian Language Processing (IALP)*, Urumqi, China, 17-19th August, 2013 [To Appear].