Text to Speech Conversion for Odia Language

Diganta Jena
111CS0126

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela - 769008, India

Text to Speech Conversion for Odia Language

Project Report submitted on

May 2015

to the Department of

Computer Science and Engineering

of

National Institute of Technology, Rourkela

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

by

Diganta Jena

(Roll 111CS0126)

under the supervision of

Prof. Banshidhar Majhi

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela - 769008, India

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela - 769008, Odisha, India.
www.nitrkl.ac.in

Dr. Banshidhar Majhi
Professor

Date

Certificate

This is to certify that the work in the thesis entitled *Text-to-Speech Conversion for Odia Language* by *Diganta Jena*, bearing the roll number *111CS0126*, is a record of an original research work carried out by her during the period July 2014-April 2015 under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

Dr. Banshidhar Majhi

Acknowledgement

During the course of this project I faced many hurdles which could not have been overcome if not for the kind support of many individuals and our institute. I would like to express my heart-felt gratitude to each and everyone for their invaluable help. First and foremost, I would like to thank my esteemed supervisor Prof. Banshidhar Majhi, Department of Computer Science and Engineering, National Institute of Technology Rourkela, for his constant encouragement and well-advised guidance. He stimulated me to work and provided valuable information relating to my project. Without his direction, this project would not have progressed this far. I also thank my friend and batch mate Sandeep Panda, who has been working on this project previously, for all the brainstorming sessions we conducted for completing the work.

I am obliged to all the professors of the Department of Computer Science and Engineering, NIT Rourkela for instilling in me the basic knowledge about the field that greatly benefited me while carrying out the project and achieving the goal. I thank the admin and staff of National Institute of Technology Rourkela for extending their support whenever needed.

I would like to thanks my friends and family for providing me with all the random data I needed as input for my project. Their knowledge in Odia helped me create innumerable test cases for the project. Last but not the least I thank my parents explicitly for suggesting ideas and warning me about the various exceptions in Odia language which would never have come to my notice otherwise.

Diganta Jena

Abstract

Text-to-Speech(TTS) system is designed to generate natural and intelligible sounding speech from any arbitrary input Odia digital text. The arbitrary text can be generated from a corresponding image file that undergoes processing through an Optical Character Recognition (OCR) scheme. This project is a research work on various techniques viable to develop the segmentation phase of the OCR specifically and a corresponding TTS for Odia language. The pre-processing and segmentation phases of the OCR are thoroughly explored in our work. A scheme is contrived for extracting the atomic Odia characters from a given string of text consisting of both characters and *matras*. The concept of histograms has come handy in our research. An altogether new L-corner scheme is formulated to handle the exceptional cases. Extensive simulations are carried out to validate the efficacy of the proposed algorithm. Results show that the proposed scheme attains a high level of accuracy.

The methodology used in TTS is to exploit acoustic representations of speech for synthesis, together with linguistic analyses of text to extract correct pronunciations and prosody in context of Odia language. Phonetic analysis to convert grapheme to phoneme is achieved using an array numbering system for the audio files comprising of pre-recorded natural human speech. Concatenation of phones and diphones generates the speech. The phoneme database creation studying the prosody of Odia language is our primary focus in this project apart from accurate and intelligible speech generation.

Keywords: OCR, TTS, Odia, Histogram, L-corner, Connected Components, Speech, Prosody

Contents

List of Figures

Chapter 1

Introduction

Odia is one of the 22 official languages of India and has been awarded the distinction of classical language by the Govt. of India. Predominantly spoken in the Indian state of Odisha with over 80% of the population being native speakers, Odia is the second official language of Jharkhand too. Although having such an ancient origin, very little research has been done to adapt it into the new world of the Internet by digitalizing it. Lack of authentic linguistic resources like a word and sound database further prevents any kind of research work. In this project, we achieve the goal of digitalizing any printed Odia text and also generating speech for this very text.

Optical Character Recognition (OCR) refers to the task of recognizing written text and converting it into machine readable form. OCR for Odia language can have multiple applications like digitizing historical documents and the rich Odia literature for enthusiastic bibliophiles. The Text-to-Speech (TTS) conversion system can prove to be a vital aid for sight impaired Odia people to follow the content of any book, newspaper, magazine, et cetera.

1.1 Odia Language Script

1.1.1 Odia Basic Characters

Odia script is unique and not similar to the Devanagiri script, but the phones or syllables are quite alike only with a few discrepancies. Odia alphabet has 11 vowels and 36 consonants which constitute the basic simple character set.

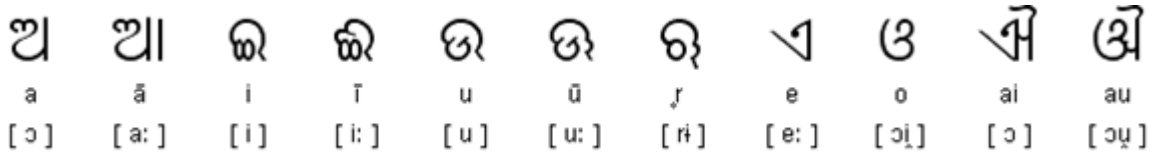


Figure 1.1: Odia Vowels

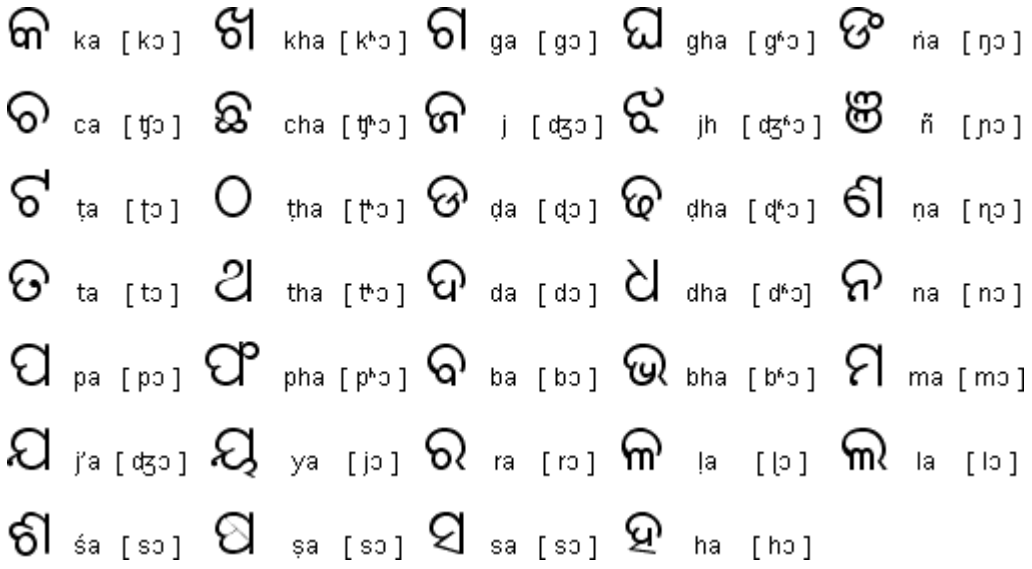


Figure 1.2: Odia Consonants

1.1.2 Odia Diacritics

Along with this basic character set there are a number of diacritics (*matras*), each for the independent vowels.

Figure 1.3: Independent Vowel *Matras*

There are certain dependent vowels *matras* called the *'chandrabindu'*, the *'anuswara'*, the *'bisarga'* and the *'halanta'* in Odia dialect.The combination of two consonants, a conjunct, is portrayed in Odia script using the *phalas*.

କଁ	କଂ	କଃ	କ୍
kã	kaṁ	kaḥ	k

(a) Dependent Vowel *Matras*

କ୍ର	କ୍ର	କ୍ୱ	କ୍ୟ
kra	rka	kwa	kya

(b) Odia *Phalas*

Figure 1.4: Other Diacritics

1.2 Segmentation

In the OCR, the pre-processing stage does the segmentation of the input image into sentences, words and finally characters. The segmentation is basically a dividing or demarcating process. Segementation is essential to properly recognize the characters. There are various techniques to segment an image. Depending of the script of the language used, the optimal technique is used. The speciality of Odia language is that it contains diacritics which can be placed to the left, right, below and above the basic character. According to the position of occurance they are separated as will be explained later. Special cases of joint *matras* is also solved. The words are separated by whitespaces which can be easily deteced.

1.3 Speech Synthesis

The artificial production of human speech is called speech synthesis. A Text-to-Speech (TTS) system converts normal language text into natural sounding speech. This text can be digital or hand-written depending on the capacity of the TTS system. Generally synthesized speech is generated by concatenating selected pieces of recorded speech that are stored in a database. The recorded speech can be in the form of phones(single sound or syllable), diphones(combination of two syllables) or triphones as per convenience and system specification. This categorization of the TTS systems in terms of the size of the stored speech units is the most common. A system that stores phones or diphones provides the largest output range, but may lack clarity. If the output of the TTS system is limited for example in telecom services, the best approach would be to save the exact word or sentence as the pre-recorded speech.

The TTS system architecture has two components: a front-end and a

back-end. The front-end comprises of the process of assigning phonetic transcriptions to words. Phonetic transcription is a kind of mapping done from characters in the language alphabet to a symbolic representation easily understandable. This process is called text-to-phoneme or grapheme-to-phoneme conversion. After phonetic transcription, prosody analysis is done to extract the information necessary for generating natural sounding speech. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation. The back-end finally converts this symbolic linguistic representation into the synthesized sound.

1.4 Motivation

Optical Character Recognition (OCR) is useful in many aspects starting from bank documents verification to number plate recognition. For Odia language it can be especially used to digitize historic documents, manuscripts and books. OCR has received significant attention from the pattern recognition community owing to its challenging nature and diverse impacts. Very little research has been done for Odia OCR because it is a regional language. The Text-to-Speech system for Odia too remains an unexplored field of research. Today, applications of TTS are in automated telecom services, as a part of a network voice server for e-mail, up-to-the-minute information to a telephone user, in computer games, and last but not least, in aids to the differently abled.

1.5 Problem Statement

The segmentation phase of the Optical Character Recognition (OCR) preprocesses the characters in an input line of Odia characters (image file) to extract them in atomic form. The focus is on the pre-processing of the string

of text combining a variety of methods to segment the characters. After the segmentation process, an existing recognition scheme is used and a string of characters mapped to numbers is fed as input to the Text-to-Speech (TTS) system that ultimately produces the natural sounding speech.

1.6 Thesis Organisation

Remaining part of the thesis is divided into the following chapters:

Chapter 2: Literature Survey

This chapter summarizes the existing work done in OCR and TTS for Odia language developed so far.

Chapter 3: Segmentation of Compound Odia Characters

This chapter outlines the proposed scheme for the segmentation process of the OCR and addresses all the exceptional cases arising in this endeavor.

Chapter 4: Speech Synthesis

This chapter describes the process of generating the speech from the input text.

Chapter 5: Simulation Results

In this chapter the result is shown in details with proper figures, tabulation and statistics for the proposed scheme.

Chapter 6: Conclusion and Future Scope

This chapter presents the analytic results of the overall achievement and future scope of the project.

Chapter 2

Literature Survey

2.1 Architecture of TTS System

The various steps of the TTS system shown in Figure 2.1. In the text analysis phase the OCR can be implemented. The text is recognized and the corresponding phonetic transcript is generated. From this transcript the prosodic analysis is conducted and the speech synthesis is done through concatenation.

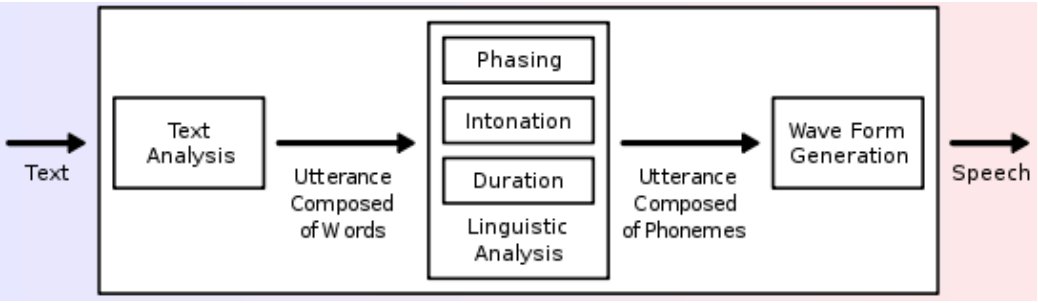


Figure 2.1: Overview of a typical TTS system adapted from Wikipedia

2.2 Segmentation

Prior to segmentation noise removal and binarization of text image is done. This enables to distinguish the background pixel and foreground pixel which represents the required text. The text is converted to black pixels while the background pixels are made white. This can be done by thresholding the image file after its converted to grayscale and then to black and white image. Now the image file contains only the text and the background. Taking advantage of this property, histograms can be utilized to analyse the image and segment the image.

2.2.1 Histogram

The histogram of an image is the range and distribution of pixels values in an image. In the segmentation process we utilize the histogram to count the number of foreground pixel in a row or column of an image. Studying the histogram we can separate the characters in the image.

2.2.2 Connected Component Analysis

Connected Component Analysis is the process of finding the sections of the image which are joint or connected to each other. Basically the connected component analysis follows a two pass algorithm over the image. In the first pass, temporary labels are assigned to image pixels or the image matrix elements. If there exists some connection between the current pixel to the adjoining neighbour pixels, then a equivalence set is created for these connected pixels. In the second pass this equivalence is over-ridden by assigning the smaller of the two labels. Connectivity is thus checked between neighbouring pixels, especially the North and West neighbour of the current pixel as shown in Figure 2.2

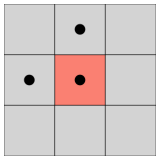


Figure 2.2: 4 connectivity of pixel with neighbours

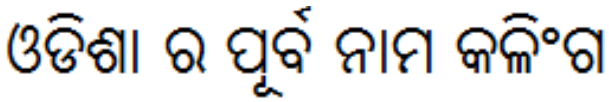
Chapter 3

Segmentation of Compound Odia Characters

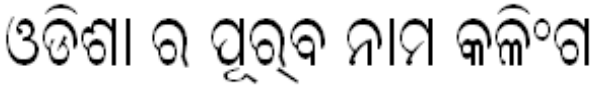
In this chapter, a novel approach is designed for segmenting an Odia sentence into words and then the corresponding characters in the word. The sentence is a printed script using fonts like the Arial Unicode MS, Kalinga script, Sarala script etc as shown in figure 3.1. The sentence might contain any one of the 11 vowels and 36 consonants along with any of the diacritics mentioned in chapter 1. The most important task is to preserve the character and the order of appearance of the characters in the word. The entire pre-processing scheme is shown in Figure 3.2.

3.1 Vertical Split

The individual characters can be split horizontally or vertically at the first step. In our implementation we have chosen the vertical splitting first. This is based on the fact that when a script is read, it must be done in order of the appearance of each character in the way it is written, that is from left to right. Thus to preserve that order vertical extraction is done first. In this



(a) Kalinga



(b) Arial Unicode MS

Figure 3.1: Sample sentence in different scripts

Figure 3.2: A block diagram showing the steps followed in the scheme

step, a vertical histogram is constructed for the input image. The histogram consists of the frequency of foreground pixels along a scan line which runs from the left end of the image to the right end. A sample histogram is shown in Figure 3.4 for the input sequence shown in Figure 3.3.

3.2 Horizontal Split

After the vertical splitting, horizontal splitting is done. In this case, the input are the extracted characters from the vertical split with or without

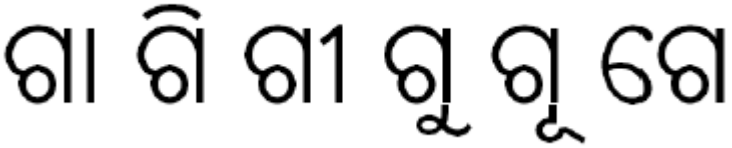


Figure 3.3: The input sequence

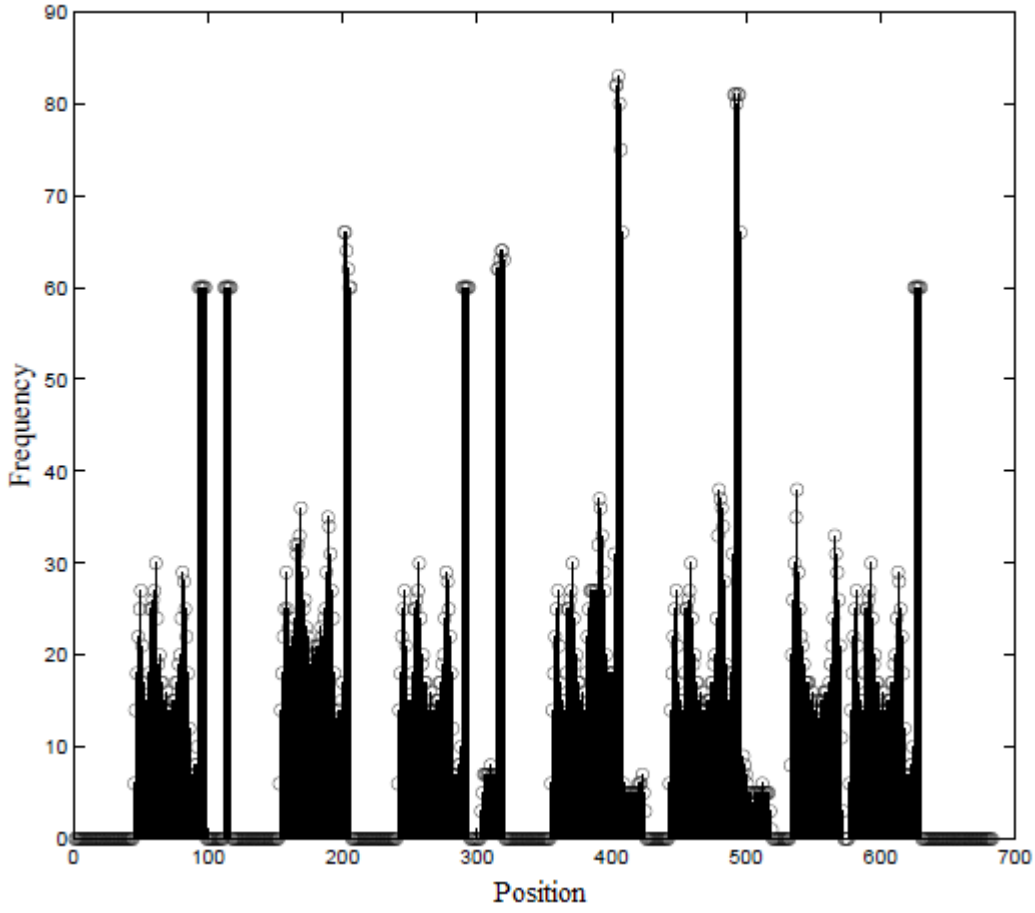


Figure 3.4: Vertical histogram for the above input sequence

Figure 3.5: An input to the horizontal split

some *matras*. In this process, a horizontal histogram is constructed in the same way as the vertical histogram except that the scan lines run from top to bottom and count the frequency of the foreground pixels. A sample of an horizontal histogram is shown in Figure 3.6 for the input shown in Figure 3.5.

In this step, a similar approach to the previous scheme is applied. The mid points of the white spaces are extracted and the intermediate characters are obtained. Thus at the end of each of these two shapes, maximum of the characters are separated out individually. However, for some cases, both due to the fonts and the properties of Odia characters, there arises cases which are not resolved by these two schemes. To handle those cases, first an L-Corner algorithm is applied which is followed by connected component analysis.

3.3 L-Corner Algorithm

After the separation of *matras* by the vertical and horizontal histograms, some cases remain unresolved. These cases arise due to two major reasons:

1. There are certain fonts that allow some *matras* to be joint to the characters thus making it impossible to be detected by the use of histograms.

2. Some *matras* are convoluted and occupy space along-side the characters

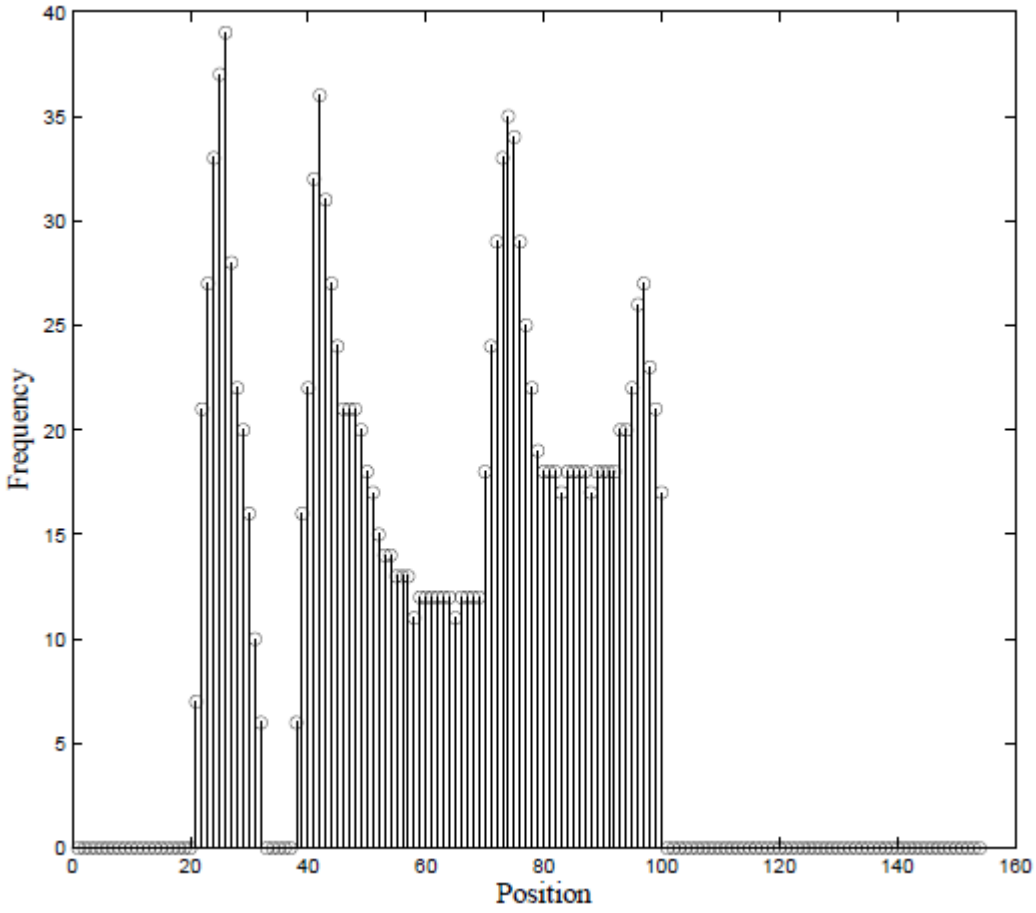
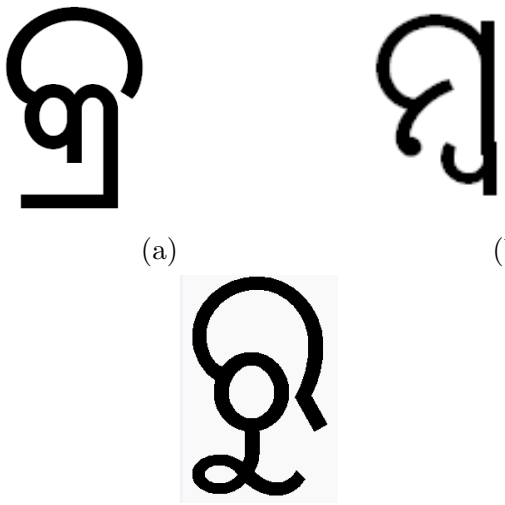


Figure 3.6: Horizontal histogram for above input sample



(a) (b)

(c)

Figure 3.7: Different cases of joint matras

and are interefere with the histograms for the basic character. In such a case also, the *matras* cannot be separated by using the histograms.

To address the first problem, we devise a L-Corner algorithm. The problem basically occurs as some fonts allow the *matras* to be connected to the characters. However, this appears only with the *matras* that are used underneath the character. Some of the fore mentioned cases are shown in Figure 3.7

To solve the above problem, a horizontal split is introduced at the joint location of the character and the *matra*. For this horizontal split, the line of pixels where the joint occurs, that is the line containing the end of the character and beginning of the *matra*, is replaced with the background pixel value. Detection of this line is done either by identifying the end of character or the joint itself which is in the form of the inverted L-corner shape. A sample of the different types of L-corners detected are shown in Figure 3.8. This results in a separation of one pixel width between the character and the

matra.

The detection of the required line is achieved by a single horizontal top-down scan of the character image. The steps are enumerated as follows:

1. The image is scanned left to right row-wise starting from the middle of the image as the occurrence of the joint is always in the lower-half of the image. This also reduces the time of computation.

2. While scanning a row, the first foreground pixel is detected. If a minimum threshold of consecutive foreground pixels occurs in the current row and subsequently a minimum set of background pixels are encountered in the row below it, then this row might be the required line and the algorithm proceeds to the next step. If not, the scanning proceeds to the next row.

3. On further scanning the row it can be checked whether the end of character occurs first or the joint occurs first as shown in cases (a) or (b) and (c) of Figure 3.8 respectively. Considering the first case, the scanning will check whether the current row contains foreground pixels and the row below it contains background pixels. The termination case occurs when foreground pixel occurs in both the current row and below it as shown in Figure 9. If this condition fails, the algorithm proceeds to the next step.

4. If the first case fails, the algorithm checks for the second case that is the L-corner. The row below the current line is scanned to obtain the first foreground pixel. Next it is checked if a minimum set of foreground pixels occurs in the vertical direction or column-wise. This is the L-corner. This condition check terminates when a set of background pixels are obtained along the column starting from the current row. Also it can be further checked to be the terminating point if a minimum

threshold of background pixels occur in the row below the current one.

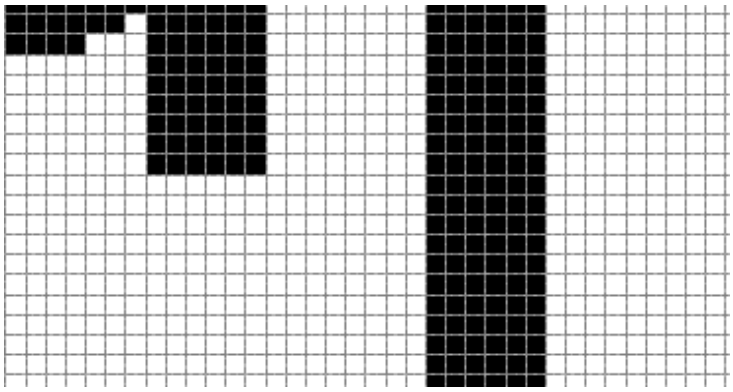
5. If all the conditions are satisfied then the row below the current row is the required line. All the pixels along this line are converted to background pixels.

A sample resolved case are shown in Figure 3.9

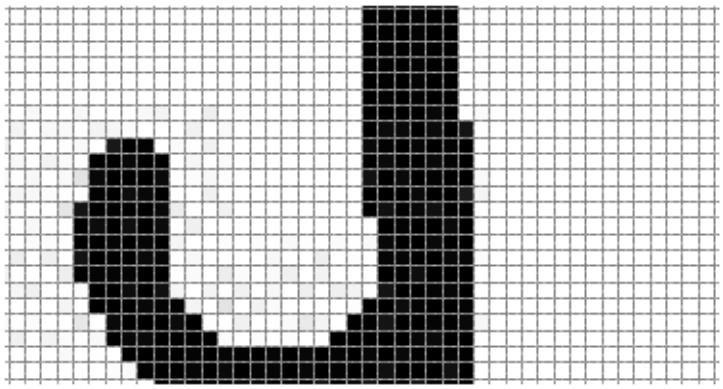
3.4 Connected Component Analysis

After the horizontal split is introduced, the final step involves application of connected component analysis. The use of connected components is essential as opposed to a horizontal histogram due the fact that some *matras* have a high curvature and occupy the same space as the character itself, thus making it impossible to be separated by histogram. An example of such *matras* is shown in Figure 3.10

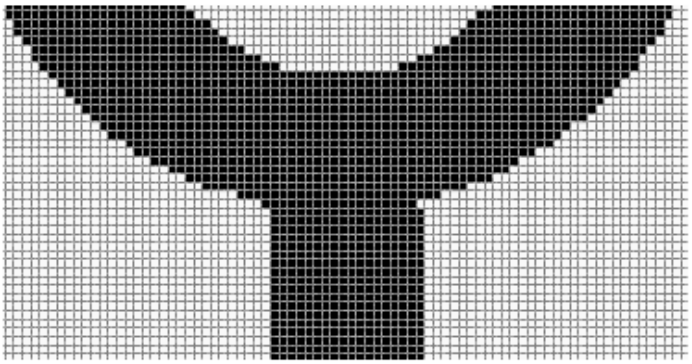
In the connected component analysis, a two pass approach is used. In the first pass , temporary labels are assigned to the different regions. In the second pass, the equivalence relation is used to reduce the different labels to a single label and form the various components. The components are separated out by the different labels assigned. Thus at the end of this step all the characters and the *matras* are separated.



(a)



(b)



(c)

Figure 3.8: Various types of L-corners detected

Figure 3.9: A sample solved case generated after running the L-corner algorithm

Figure 3.10: An example of the convoluted *matra*

Chapter 4

Speech Synthesis

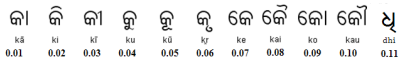
4.1 Text Analysis

In this step, the text input is recognized and a string array is generated corresponding to the input. The string array contains numbers as shown in Figure 4.1.

1. Vowels:1-12
2. Consonants:13-48
3. Whitespace:49
4. End of sentence: 50
5. Independent Vowel *Matra*: 0.01-0.11
6. Dependent Vowel *Matra*: 0.12-0.15
7. *Phalas*: 0.16-0.19

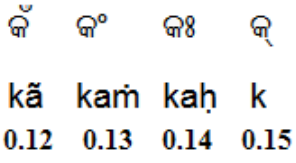
For compound characters which are a combination of two consonants, the consonants are extracted and using the *halant matra* they are re-written to generate the number string. Such an example is shown in Figure 4.2.

(a) Odia vowels



(b) Odia independent *matras*

(c) Odia consonants



(d) Odia dependent *matras*

(e) Phalas

Figure 4.1: Numbering system for various characters in Odia

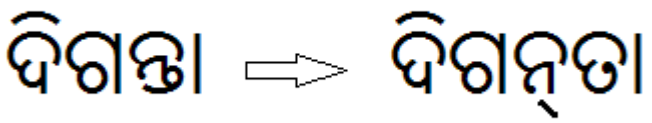


Figure 4.2: Expansion of compound characters for easy generation of number string

ମୋ। ନାମ ଦିଗନ୍ତ। । ମୋ। ଘର ଭୁବନେଶ୍ୱର ।

(a) A sample input text

[37 0.09 49 32 0.01 37 49 30 0.02 15 32 0.15 28 0.01 50 37
0.09 49 16 39 49 36 0.04 35 32 0.07 42 0.17 39 50]

(b) The corresponding phonetic number string

Figure 4.3: Text to phonetic transcript conversion

This number string is the phonetic transcription of the input text. This process is called the text to phonetic script conversion. An example is illustrated in Figure 4.3. Integers are selected for representing the vowels and consonants for ease of use. Decimal values ranging from 0.01 to 0.19 are selected for the rest of the characters to easily distinguish whether there is only the basic character or the basic character with a *matra*. Whitespace and full stop should be given a unique number because they signify silence of different durations in the actual speech synthesized.

4.2 Prosody Analysis

In Odia the stress for a particular syllable varies according to place of occurrence in the word. There is a more *'a'* sound prominent when the syllable or character occur at the end of the word. This can be observed from the Figure 4.4.

4.3 Grapheme to Phoneme Conversion

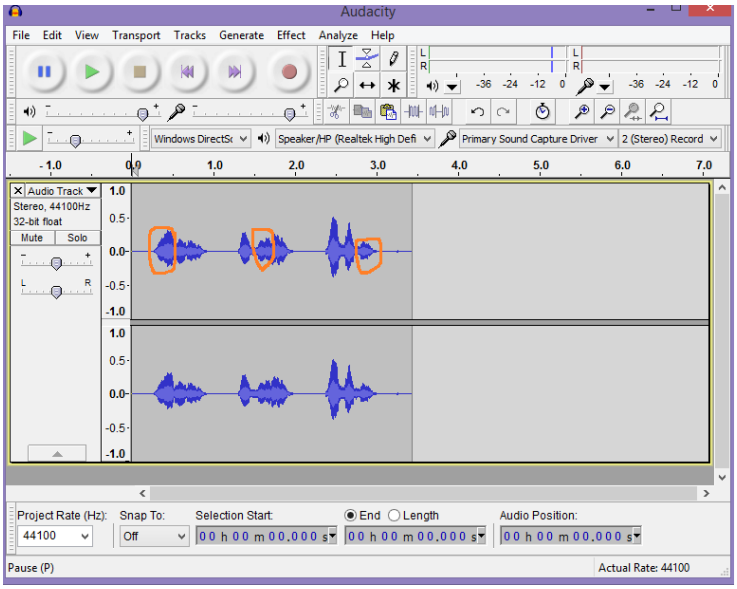
After the phonetic description is transcripted, the number string is parsed to read the corresponding sound file. There are 7 types of sound combination possible in Odia language:

1. V: only vowel
2. C: only consonant
3. HC: consonant with a *halant*
4. CV: consonant with vowel *matra*
5. CC: combination of two consonants
6. CCV: combination of two consonants with a *phala* or *matra*
7. CVV: consonant with *phala* and *matra*

These are the phones, diphones and triphones associated with Odia language. The phones are numbered as per the numbering system discussed in section 4.1. The diphones and triphones are created by combining various phones.

ମରଣ ତମର କଲମ

(a) Odia words containing the character *'m'* at the start, middle and end of the word



(b) The corresponding pitch or intonation of the words

Figure 4.4: Prosody analysis for different Odia words containing same character at different positions

4.4 Sound Database

The sound is recorded using the Audacity Sound Tool. The files are saved in .wav format for accessing the inbuilt functions of the MATLAB tool used for coding the project. Three types of audio files are saved for each of the phones and diphones. These 3 types are the start, middle and the end type. Three directories are made named "start", "middle" and "end", each storing sound according to the notation used for the basic characters or phones. The diphones CV and CC are stored under a sub-directory named C and the audio file named V. The triphone CCV is expanded as phone HC and diphone CV where HC is half consonant stored under C sub-directory with *halant* number, 15.wav file. The CVV case is a combination of CV an V and accessed in the respective sub-directories.This is shown in Figure 4.5.

4.5 Sound Concatenation

The concatenation is done directly after detecting the type of phone occurring in the number string. The detailed algorithm is described in the flowchart given in Figure 4.6

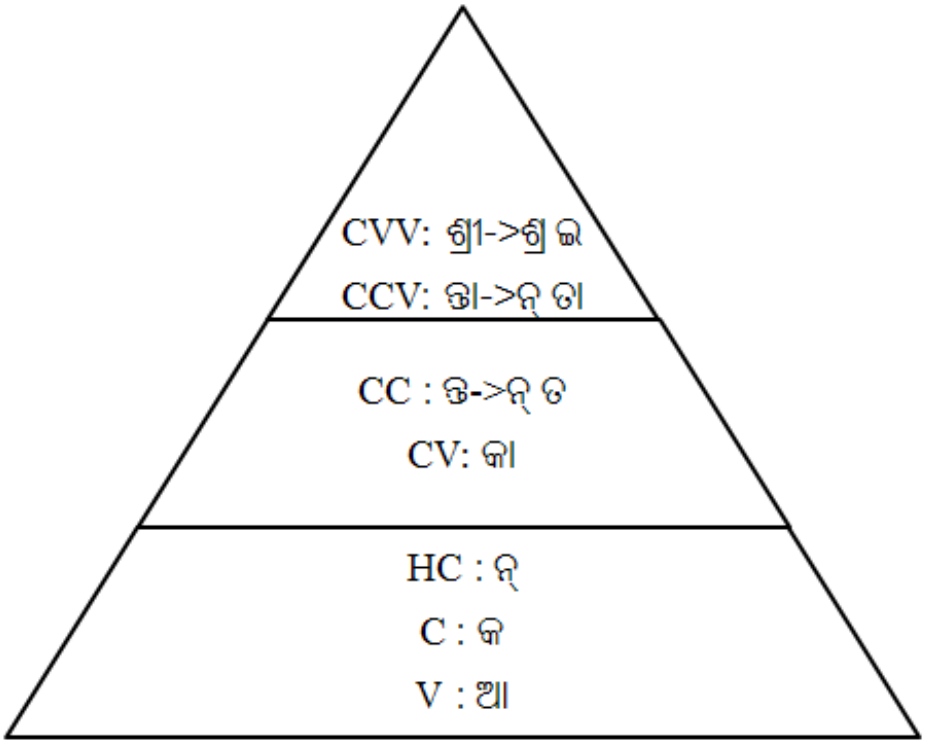


Figure 4.5: Phones, diphones and triphones creation shown in hierarchical fashion

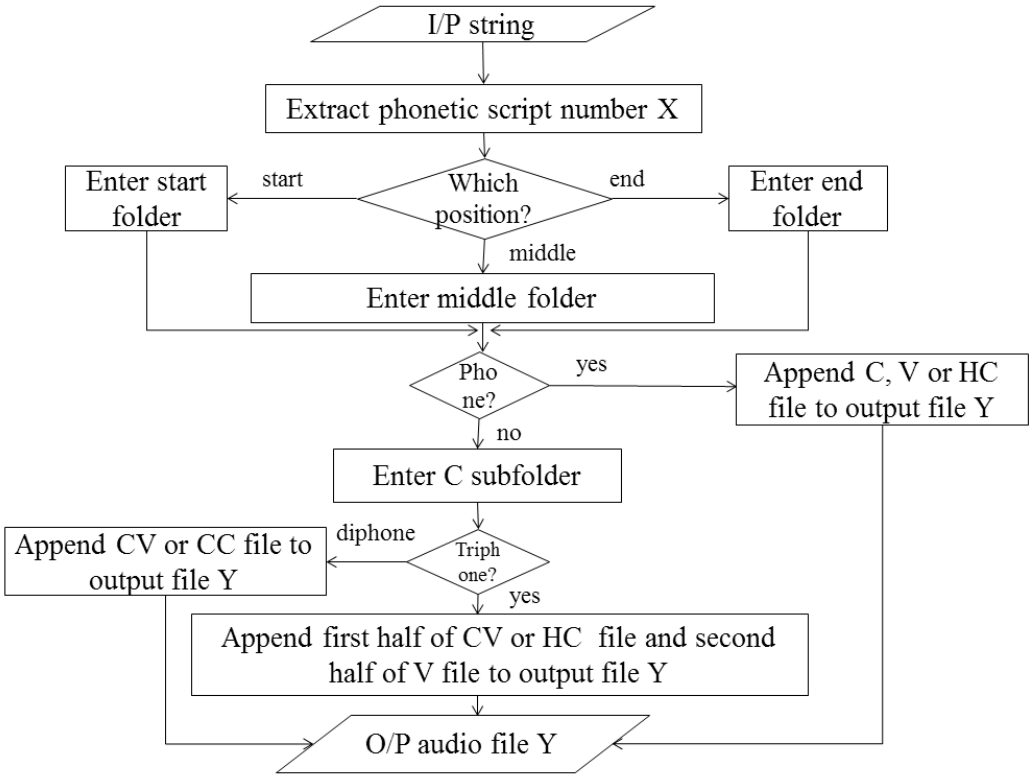


Figure 4.6: Algorithm for sound concatenation

Chapter 5

Simulation Results

5.1 Segmentation

The output generated for the L-corner case is shown in Figure 5.1.

5.2 Speech Synthesis

The output audio for the input number string given in Figure 4.3 is given in Figure 5.2.

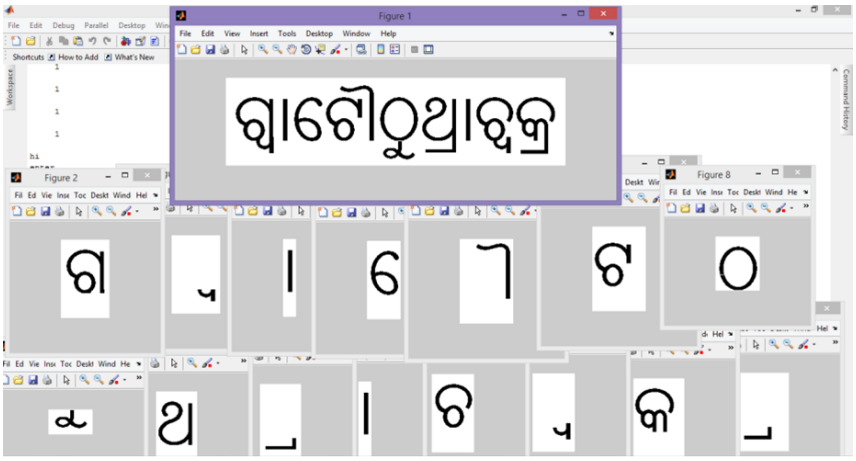


Figure 5.1: Simulation of segmentation process

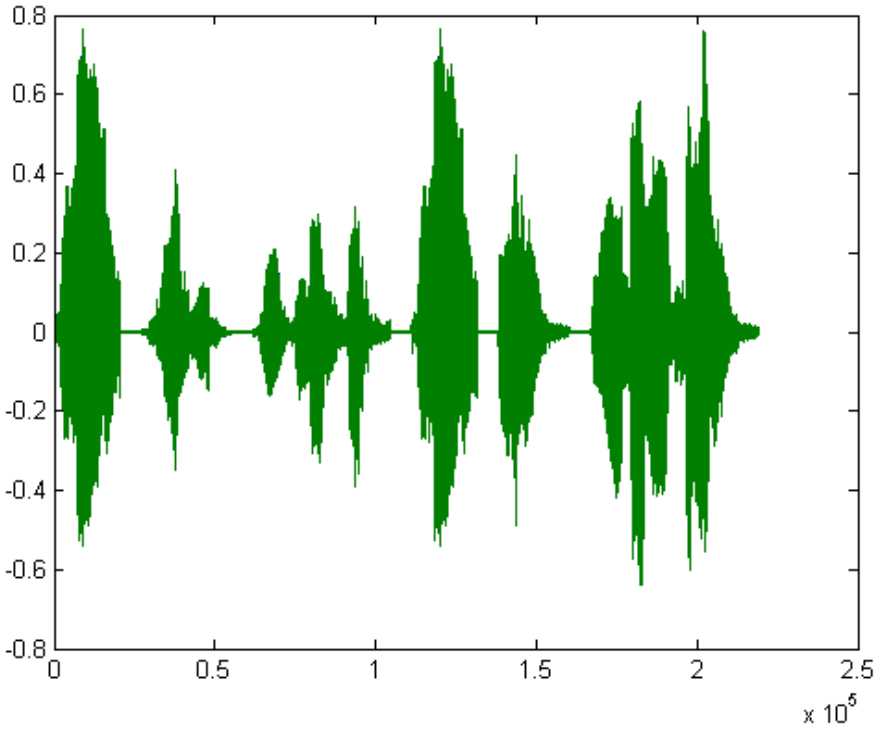


Figure 5.2: Simulation of speech wave

Chapter 6

Conclusion and Future Scope

In this project we process a text image by pre-processing and segmentation.The output of this OCR is set as the input for the TTS system. A roboust scheme for segmentation of Odia text is implemented.The future scope includes extending this scheme for the compound consonant characters called the *yuktashara* in the Odia script. The TTS system is quite accurate and further work can be done to reduce the size of the sound database which when fully expanded will be approximately 40Mb. The special cases of generating sound for Odia numericals and abbreviations can also be explored in future. Finally the OCR and TTS can be integrated using any existing feature extraction and classification system using neural networks.

Bibliography

[1] B. B. Chaudhuri, U. Pal and M. Mitra, "Automatic recognition of printed Oriya script," *Sadhana*, vol. 27, part 1, pp. 23–34, February, 2002.

[2] R. Mithe, S. Indalkar, N.Divekar, "Optical Character Recognition," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 2, pp. 2277-3878, March, 2013.

[3] S. Mishra, D. Nanda , S. Mohanty ,"Oriya Character Recognition using Neural Networks," *Special Issue of IJCCT* , vol. 2,issue 1, pp. 3- 4, December, 2010.

[4] T.K. Patra, B. Patra and P. Mohapatra,"Text to Speech Conversion with Phonematic Concatenation," *International Journal of Electronics Communication and Computer Technology*, vol. 2, Issue 5, pp. 223-226, Sept, 2012.

[5] A.A. Raj., T. Sarkar. and S.C. Pammi, "Text Processing for Text-to-Speech Systems in Indian Languages," in 6th ISCA Workshop on Speech Synthesis, Bonn, Germany, Aug. 22-24 2007.