Text-to-Speech Synthesis Techniques

Eduardo M. B. de A. Tenório and Tsang Ing Ren

Centro de Informática, Universidade Federal de Pernambuco
Recife, PE, Brasil – www.cin.ufpe.br
{embat,tir}@cin.ufpe.br

June, 2014

ABSTRACT

This paper provides a short and comprehensive overview of Text-to-Speech (TTS) synthesis by highlighting its digital signal processing component. Initially it is shown how an input text is analysed and transformed into a sequence of control signals to be converted in speech by a TTS synthesis technique. The first two rule-based approaches (formant and articulatory) are briefly explained, then data-driven methods are explored. Concatenative synthesis is simpler than the rule-based ones since there is no need in determine rules for speech production. However, it introduces discontinuities at unit boundaries. This discontinuities are smoothed by signal processing, modifying the prosody of speech and making it sounds unnatural. An improvement to the concatenative synthesis is the unit selection technique, that partially solves this problem by storing numerous instances for each unit with differences in prosody. The best match is selected and concatenated. To resolve mismatches, speech synthesis systems combines the unit-selection method with Harmonic plus Noise Model (HNM) synthesis. This model represents speech as a sum of harmonic and noise parts, enabling more natural sound modifications. Last, but no less important, Hidden Markov Model (HMM) synthesis combined with an HNM is introduced in order to obtain a TTS synthesis system that requires smaller development time and cost.

keywords*: speech synthesis, formant, articulatory, concatenative, unit selection, Harmonic plus Noise Model, Hidden Markov Model*

1 INTRODUCTION

In the present time speech synthesized by a computer is more real than it was to Captain Kirk in the 1960s. A computer that understands a conversation and speaks as natural as a human may still take a decade or two to arise, but what exists today enables vocal commands to be sent to a machine and have simple sentences understood, as well as to receive a (close to) natural vocal response (with robotic accent and prosodic distortions). This interaction needs two complementary fields of speech processing: recognition and synthesis. The first allows a machine to extract information from a speech such as the message (but not its meaning), the speaker, the speaker's health and emotional state and many others. The second transforms an input text into acoustic waves that are understood by humans as speech. The aim of a TTS synthesis system is to generate a speech similar to one produced by a human. In other words, to create a system that equals the human performance. To achieve this, the speech synthesizer requires two qualities: intelligibility and naturalness [1].

A TTS system can be used in a variety of applications: screen reader for a blind person, a new voice for people with speech disorders, interface for a personal assistant, more interactive email messages, virtual actors whose speech is generated directly from the screenplay (still in the uncanny valley) and other uses to be thought.

This paper is proposed to explain how a TTS engine works, since the text served as input until the speech synthesized, following the coverage made by [1]. The next section shows the common modules to all speech synthesizers, turning a plain text into a set of control signals that represents the desired speech to be synthesized. It is based on the explanation provided by [2, 3]. Section 3 gives an overview of the rule-based techniques (formant and articulatory synthesis) and section 4 explains in more details the data-driven approaches (concatenative, unit selection, HNM and HMM synthesis). Finally, the conclusions are given in section 5.

2 TTS ENGINE

The TTS system simulates part of the speech chain shown in [2]. The function of the sequence of blocks from the text input until the acoustic waveform output is mimicked by the TTS engine .

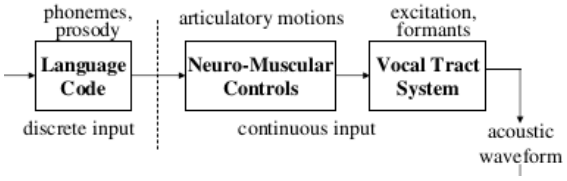

Figure 1: *Blocks simulated by a TTS in the speech chain.*

This section shows the common pieces used by all synthesis techniques to read a text and prepare it to be spo-

ken. The engine is divided in four blocks, each one composed of specialized modules:

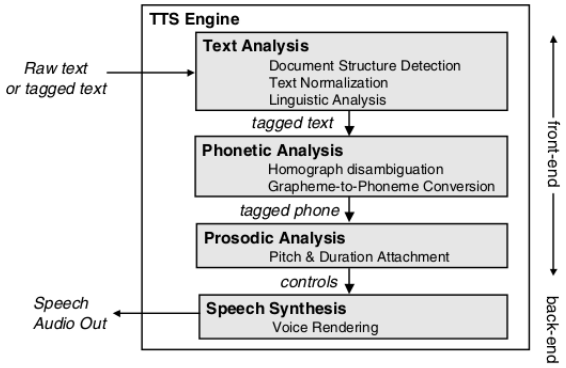

Figure 2: *Text turned into speech by a TTS engine.*

Text Analysis Transforms an input text into a tagged text. It contemplates the basic aspects of text reading.

The **document structure detection** determines the location of all punctuation marks in the text and decides their significance with regard to the sentence and paragraph structure of the input text [2].

The **text normalization** handles text problems that occur in TTS systems, such as abbreviations and acronyms (Mr., Dr., etc.), dates, numbers, unusual symbols and foreign words (as in *"I read Focault"*).

The **linguistic analysis** classifies the words, *aka* Part-of-Speech (POS), in gramatical types, determines the pauses between phrases, emphasis on right words inside a sentence and sets the style of speacking (sad, angry and etc.). Uses a dictionary and usually have to deal with unknown or ambiguous words. Parsers may be used to give correct output (tagged text), but tend to be very slow.

Phonetic Analysis Ultimately, the tagged text obtained from the basic text processing block of a TTS system has to be converted to a sequence of tagged phones which describe both the sounds to be produced as well as the manner of speaking, both locally (emphasis) and globally (speaking style) [2].

The **homograph disambiguation** operation resolves the correct pronunciation of each word in the input text (as "read" conjugated in the present and in the past).

The **grapheme-to-phoneme conversion** assigns the correct phonetic set to the token stream. It must be stated that this is a continuous language dependent process since the phonetic transcriptions of the token boundaries are influenced by the transcriptions of the neighboring token boundaries [3].

Prosodic Analysis Is the last step before the proper speech synthesis. It provides the synthesizer with the complete set of control signals that represents the desired sound (sequence of speech sounds, their durations, and an associated pitch contour). The assignment of duration and pitch contours is done by a set of pitch and duration rules, along with a set of rules for assigning stress and determining where appropriate pauses should be inserted so that the local and global speaking rates appear to be natural [2].

Speech Synthesis It is where the control signals generated from the input text are converted into acoustic waves. There is numerous techniques, which are described in both sections 3 and 4.

3 RULE-BASED TECHNIQUES

Rule-based synthesis techniques are characterized for presenting a set of well defined and rigid rules to produce speech. They are built using the vocal system as inspiration, but may differ in the way it is modeled.

3.1 FORMANT SYNTHESIS

The formant synthesis technique models the vocal tract with a satisfactory precision by simulating formant frequencies and formant amplitudes. Each phoneme has a group of pitches in the voiced signal. The frequencies where these pitches occur and their amplitudes defines the phoneme. The speech waveform is approximated using a set of rules according to the acoustic theory presented in [4].

The speech is generated when the voiced and unvoiced sources are stimulated, making the formants ressonate in the vocal tract and thus producing the speech. The use of a compensator when both signals are joined creates fricative, plosive and nasal effects.

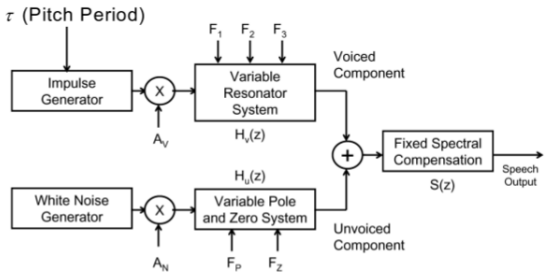

Figure 3: *Speech synthesizer based on a formant synthesis model.*

This method produces highly intelligible speech with few resources (ideal for embedded systems). However, the voice sounds very unnatural. Although the resonator's parameters being very correlated, they are too difficult to be set automatically [1]. The formant synthesis was very popular in the 1980s with the DECTalk and the Klatt Synthesis Interface. Nowadays was replaced by more sofisticated techniques.

3.2 ARTICULATORY SYNTHESIS

Articulatory synthesis generates speech by direct modeling of the human articulator behavior, 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 [1].

This method possess two major disadvantages. The first is the difficult to acquire data to the model. The second is to find a balance between accuracy and ease of implementation. More refinated systems tend to be harder to implement. Articulatory synthesis is the technique with the worst results according to [1].

4 DATA-DRIVEN TECHNIQUES

While the rule-based TTS systems tend to define a set of rules and produce speech easy to understand, but difficult to enjoy, the data-driven approach is based on a large set of data composed of hours of recording. The naturalness is better than the previous approach (it is composed of real voices after all), but the intelligibility may be compromised (not so much). However, in the practical world, both the naturalness and intelligibility are higher than the rule-based methods (which parameters are difficult to find automatically, making the models not viable for real applications).

4.1 CONCATENATIVE SYNTHESIS

The most common data-driven technique is the concatenative synthesis. As the name says, this method concatenate (or connect, join, etc.) various units in sequence to form a viable utterance. The units are prerecorded pieces of speech in one of the types (from longer to shorter): words, syllables, half-syllables, phonemes, diphones or triphones. Longer units leads to better quality, but needs more memory. Shorter units possess more concatenation points, making the naturalness worse, but asks for less memory [1].

The most widely used units in concatenative synthesis are diphones. A diphone is a unit that starts at the middle of one phone and extends to the middle of the following one. Diphones have the advantage of modeling coarticulation by including the transition to the next phone inside the diphone itself [1]. A diphone inventory is usually built from the dictation of sentences containing phonemes of all possible situations (allphones). These sentences are then cut in various diphones to populate the base. The phonemes are then labelled and segmented.

Even being made of real voices, the concatenative synthesis is still very unnatural (but highly intelligible), due to the breaks in natural prosody expected to be smooth as the real speech. There are some signal processing approaches to make the naturalness sounds better, but they tend to corrupt the speech when these modifications get "greed". One simple way to partially resolve this problem (or at least to smooth the unit boundaries) is by the technique shown next.

4.2 UNIT SELECTION SYNTHESIS

The unit selection synthesis is a kind of concatenative synthesis improved. It keeps the same structure, but now an unit is composed of various instances, each representing one prosodic variation. The instance that matches closest to the *target* (the desired piece of speech) is selected and concatenated to the speech in formation. Since an unit contains multiple instances, the probability of an instance be the best match available is higher than in the default concatenative synthesis (that could be viewed as a particular case of unit selection with one instance). This improvement in finding the desired piece leads to a lower need to apply prosody modifications. Less mismatches, less prosody modifications, higher quality of speech.

Finding the best match needs to be done logically. To quantize the best choice, a cost function is used. The cost is divided in two types: *target cost* and *concatenative cost*. The target cost is an estimate of the difference between a database unit and the target, and the concatenative cost is an estimate of the quality of a join between consecutive units in that the speech [5].

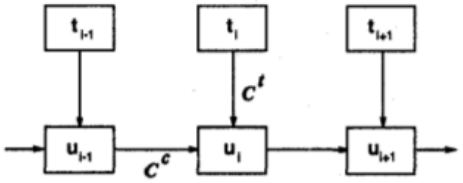

Figure 4: *Costs of a target t_i matches a unit u_i and a unit u_{i-1} join with unit u_i.*

The selection of good units for synthesis requires an appropriate definition of the target and concatenation costs and effective training of them. Each target and unit in the database has a pitch, power and duration. The closest they are, the best the match. Getting a wider view, the same can be said about a sequence of targets, denoted by $t^n = (t_1, ..., t_n)$, and units, $u^n = (u_1, ..., u_n)$, where each t_i matches a u_i, for $i = 1, ..., n$. Now it is possible to properly define the target cost and concatenation cost.

The target cost is defined as a weighted sum of p *target sub-costs* of the target t_i to the instance u_{ij} of the unit u_i

$$C^t(t_i, u_i) = \sum_{j=1}^{p} w_j^t C_j^t(t_i, u_i) \tag{1}$$

and the concatenation cost is a weighted sum of q *concatenative sub-costs* of the j^{th} component of the join between u_i and u_{ij}. As a special case, if u_{i-1} and u_i are consecutive units in the synthesis database, then their concatenation is natural and therefore has a cost of zero [5].

$$C^c(u_{i-1}, u_i) = \sum_{j=1}^{q} w_j^c C_j^c(u_{i-1}, u_i) \tag{2}$$

The total cost is given by the sum of both costs. This means that even a unit with a very small target cost is not the best match if the concatenative cost is too high (this means, the unit u_i rarely joins with the unit u_{i-1}), and another not so extremely, but similar match may be chosen due to the total cost.

The total cost is given by

$$C(t^n, u^n) = \sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=2}^{n} C^t(u_{i-1}, u_i) + C^n(S, u_1) + C^n(u_n, S) \quad (3)$$

where S denotes silence, and $C^n(S, u_1)$ and $C^n(u_n, S)$ define the start and end conditions given by the concatenation of the first and last units to silence [5]. The unit selection is the procedure to find the sequence u^n that minimizes the cost defined by *Equation 3*.

This better sequence u^n can also be chosen using a Viterbi Algorithm, that optimizes the search for the best match finding the best path in a directed graph composed of all instances of units, where each instance of u_{i-1} has departure edges aimed to all instances of u_i. The figure below shows an example of a Viterbi search for the word "two".

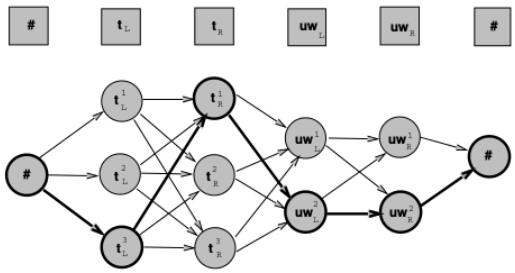

Figure 5: *The lowest cost path for diphones of the word "two".*

The details of Viterbi Algorithm and weight training are out of the scope of this paper. A deeper understanding may be found in [6] and [5], respectively.

Unit selection is the technique used nowadays to produce synthetic speech from text. However, it introduces the challenge to deal with a large and increasing database. Furthermore, there is always a considerable error rate, due to incorrect labeling of words and to fail in detect the context [1]. Implementations of this method needs a minimum recording base of 30 minutes to be feasible, but in practical applications is used between 1 and 10 hours of recording.

4.3 HNM SYNTHESIS

Prosodic modifications of speech are needed for high quality speech synthesis [1]. There are some techniques to execute this signal processing, such as PSOLA, TD-PSOLA and HNM. The HNM method, introduced in [7], when combined with the unit selection is the *state-of-the-art* of TTS synthesis. It assumes that the speech is composed of a harmonic (quasi-periodic) and a noise (non-periodic) parts, divided by a time-varying parameter, the maximum voiced frequency Fm. The lower band of the spectrum (below Fm) is represented solely by harmonics

$$s_h(t) = \sum_{k=-L(t)}^{L(t)} A_k(t) e^{jk\omega_0(t)t} \quad (4)$$

while the upper band (above Fm) is represented by a modulated noise component.

$$s_n(t) = e(t)[h(\tau, t) * b(t)] \quad (5)$$

In *Equation 4*, $L(t)$ denotes the number of harmonics included in the harmonic part, ω_0 denotes the fundamental frequency while $A_k(t)$ can take on one of the following forms shown in [8]. *Equation 5* is composed by the filtering of the white Gaussian noise, $b(t)$, which time domain structure is imposed by the envelope.

The discontinuities at concatenation boundaries are smoothed only for the harmonic part (noise is always noise), shown with great details in [7, 8].

4.4 HIDDEN MARKOV MODEL SYNTHESIS

The HMM approach comes to resolve some problems found in the concatenative methods: time consuming database creation and impossibility to create something beyond what was recorded. Using a stochastic approach two advantages appear: less memory to store the parameter of the model and more variations allowed for example [1] (as to convert the original voice to another).

The method is divided in two phases: the training and the synthesis. At each frame, a group of features (usually MFCC and its first and second derivatives) is extracted and fed to the Baum-Welch algorithm, that produces models for each phone. A model usually consists of three states that represent the beginning, the middle and the end of the phone [1]. The synthesis phase consists of two steps: firstly, the feature vectors for a given phone sequence have to be estimated. Secondly, a filter is implemented to transform those feature vectors into audio signals [1].

The quality of the speech generated by the HMM method is inferior to one generated by the state-of-the-art technique, unit selection. However, when combined with HNM (called HTS), it overtakes the unit selection method. This quality enhancement is achieved by replacing the source filter modeling approach typically used in HTS with the HNM model, which is known for being able to synthesize natural sounding speech under various prosodic modifications [1].

5 CONCLUSIONS

This paper has presented the major TTS synthesis techniques. The formant and articulatory methods are not very used today, but still have a place in applications that demands low power consumption and memory. The paper focused on unit selection combined with HNM, the state-of-the-art, which allows the synthesizer to produce more natural speech. The HMM method is an alternative to the usual method when looking to memory usage and variation possibilities. The combination of HMM with HNM has been proved to perform better than the unit selection, while is easier and cheaper to build. In the next few years this method will become dominant and the TTS systems will experience a leap quality.

REFERENCES

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

[2] Lawrence R. Rabiner, Ronald W. Schafer, *"Introduction to Digital Speech Processing"*, Foundations and Trends in Signal Processing, v.1 n.1, p.1-194, January 2007.

[3] Alexandre Trilla, *"Natural Language Processing techniques in Text-To-Speech synthesis and Automatic Speech Recognition"*, Working Paper, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, 2009, Barcelona, Spain.

[4] Dennis H. Klatt, "Software for a cascade/parallel formant synthesizer", *Journal of the Acoustical Society of America*, vol. 67, pp. 971–995, 1980.

[5] Andrew J. Hunt and Alan. W. Black, "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database", in *Proc. ICASSP-96*, pp. 373-376, 1996.

[6] George D. Forney, "The Viterbi algorithm", *Proc. of IEEE*, vol. 61, pp. 268-278, 1973

[7] Yannis Stylianou, "Harmonic plus Noise Models for Speech, combined with Statistical Methods, for Speech and Speaker Modification", PhD thesis, Ecole Nationale Supérieure des Telecommunications, Paris, January 1996

[8] Yannis Stylianou, "Applying the harmonic plus noise model in concatenative speech synthesis", *IEEE-Trans. Speech Audio Process*, vol. 9, no. 1, pp.21 -29, 2001