ROBUST SPEECH SYNTHESIS

By

Christos C. Vosnidis

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
TECHNICAL UNIVERSITY OF CRETE
CHANIA, GREECE
JUNE 2004

TECHNICAL UNIVERSITY OF CRETE
DEPARTMENT OF
ELECTRONICS AND COMPUTER ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Robust Speech Synthesis**" by **Christos C. Vosnidis** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: June 2004

Supervisor: ______________________
Prof. Vassilis Digalakis

Readers: ______________________
Prof. Michael Paterakis

____
Assoc. Prof. Alexandros Potamianos

TECHNICAL UNIVERSITY OF CRETE

Date: **June 2004**

Author: **Christos C. Vosnidis**

Title: **Robust Speech Synthesis**

Department: **Electronics and Computer Engineering**

Degree: **M.Sc.** Convocation: **June** Year: **2004**

To Maria and Miles.

Table of Contents

List of Figures

Abstract

Research in Speech Synthesis has gone a long way since the time of MITalk during the mid-80s. Nowadays, synthesizers can generate speech in such quality that can hardly be distinguished from human. At least, most of the time...

The inconsistencies that characterize the quality of synthetic speech produced by current synthesizers is the subject of this work. We start by presenting the state–of–the art in Text–to–Speech synthesis, and we proceed to identify a series of issues specific to unit–selection based synthesis techniques that are responsible for this behavior. Aiming to produce a methodology that will enhance the robustness of these techniques, we propose several modifications and enhancements both to the unit selection process and the design of the speech segment database. Finally, in an effort to transfer knowledge from the field of Speech Recognition to Speech Synthesis, we propose the use of Linear Segmental Dynamic Models as a novel method for defining join cost functions.

Acknowledgements

I would like to thank Vassilis Digalakis, my supervisor, for his many suggestions and constant support during this research. His valuable experience in the field of Speech Recognition has been an inspiration to me. I am also thankful to Alexandros Potamianos for his guidance through the chaos and confusion of some aspects of my work.

I would also like to thank my colleagues at Rhetorical Systems Ltd. The experience that I've accumulated while working with them has been priceless; not to mention the good times that I've had with the people in Edinburgh while working on the Greek version of their TTS engine.

Of course, I am grateful to my parents for their patience and *love.* Without them this work would never have come into existence (literally).

Finally, I wish to thank the following: Dimitris Giakoumis, Vassilis Stoidis and Dimitris Keramidas (for their friendship); Maria (for changing my life from worse to bad and then back to worse again); Fanouris, Yiannis, Giorgos, Stavros, Dimitris, (for the mostly good times we had together during the past few months); Miles Davis and the U2 (and they know why); *and* my sister (because she asked me to).

Chania, Crete
June 7, 2004

Chris Vosnidis

Introduction

0.1 Background

Text-to-speech synthesis, i.e. the automatic production of speech from text, has seen tremendous progress during the past ten years. This can nevertheless be attributed to the fact that Text–To–Speech Synthesis (TTS), along with Automatic Speech Recognition, provide a pair of technologies that can lead us to the next stage in Human - Computer interaction. Ever since Stanley Kubrick filmed *2001: A Space Odyssey*, where HAL 2000, a speech-enabled supercomputer with a slightly demented AI personality, would listen to others and communicate its thoughts using speech, the fulfillment of the dream where a computer would properly understand human speech and respond with a high-quality, human-like voice has become the equivalent of the Holy Grail for researchers in the fields of AI and Speech Technologies.

Scientists in both fields have since tried to approach the problem quite aggressively, but unfortunately, at least until the mid-80s, with little success. Specifically, in the case of Speech Synthesis, the results that were achieved until that time were far even from being characterized as poor. Constraints imposed by the inadequacy of microprocessor speeds, volatile and non-volatile memory and storage sizes, could not allow research in the field to deliver the promised natural-sounding speech output. This outcome should also be blamed on the approach that was adopted by scientists working on Speech Synthesis; namely on the fact that they tried synthesizing speech by modeling their systems and algorithms after the human physiological speech generation mechanism. However, our understanding of the human speech generation system dynamics was, and still remains, quite limited. Thus, the systems that were developed in pursuit of this approach were quite deficient in terms of quality of the generated speech.

The breakthrough in the field of Speech Synthesis came with the realization of the fact that since we could not successfully imitate human speech, we could try copying it. That is, instead of trying to generate speech through the modification of elementary signals by functions that try to simulate the physiology of our speech production system, we would easily get the same effect by using pre-recorded, real-life samples of those sounds that we wish to generate. Thus, synthesizing a word or a sentence gets reduced to the comparatively simple task of selecting the proper sequence of sound samples and concatenating them into a single waveform.

This is of course an overly simplified description of the speech synthesis process, but it still represents the basic idea responsible for the remarkable progress witnessed in Speech Synthesis during the past few years. Nowadays, speech synthesizers have reached a quality that has proved sufficient to allow their use in applications such as call centers and on-line information systems, that were until recently dominated by human speakers. This does not mean, however, that research in the field is over. The quality of the synthesized speech might be good, but it still has a long way to go until it gets on par with human speech.

0.2 Scope of this Work

This work tries to assist in achieving the goal of creating natural synthetic speech. We are proposing a series of improvements in the architecture of a Text-to-Speech Synthesis system that utilizes the technique of Variable Unit Selection, and the methods that are used in identifying the proper sequence of speech samples that will be concatenated to produce the desired utterance. Our system will be able to operate on the input text, perform a series of text normalization and automatic phonetization tasks, extract the required series of phonemes that need to be used and specify the supra-segmental characteristics of the utterance that we wish to synthesize. Using these template specifications, the system will select the appropriate units from the unit database, concatenate them and perform the necessary signal processing to ensure smooth transitions within units.

In addition to the proposed modifications to the synthesizer's architecture, we are also suggesting an improved methodology on the design and realization of the Speech Unit Database, which contains the set of speech units that are available to the system at synthesis time, and which are concatenated in order to synthesize the desired utterance.

Furthermore, we are describing a series of optimizations that need to be implemented by the unit selection subsystem in order to achieve the goal of synthesizing natural speech in a consistent manner. Among others, we are proposing several improvements on subjects such as the definition of unit costs and the training of cost functions. We also suggest the introduction of an additional cost function that will aid in minimizing audible spectral discontinuities and the specification of objective metrics for performing off-line evaluations of the synthesized speech.

0.3 Outline

The rest of the proposal is organized as follows:

Chapter 1 serves as an overview of the speech synthesis process, first introducing the reader to the basic ideas behind speech synthesis, and then focusing in further detail on techniques used in Unit Selection synthesis. In doing so, we will identify the challenges that have to be dealt with when working on Unit Selection synthesis, namely text-normalization, prosodic contour generation, unit database construction and optimization and unit selection.

This overview of the field will allow us to demonstrate our approach to the problem at hand. This is done in Chapter 2, which describes the challenges that will be encountered in order to achieve the goal of *Robust Speech Synthesis*.

In Chapter 3, we present the reader with our proposal regarding a novel method for defining and computing join costs. Borrowing techniques widely used in Automatic Speech Recognition, we will be define join costs based on a dynamic segmental model of speech, trained on real-life speech data extracted from the voice talent that was used for recording the synthesizer's speech database. The learned underlying representation of the speech signal will allow us to evaluate prospective joins in a more consistent and user-oriented way, constraining the number of audible spectral discontinuities that will be present in the synthesized signal.

Finally, in Chapter 4 we are providing an overview of the design decisions that we have made, and an outline of the work that needs to be done in order to implement the proposed synthesizer.

Chapter 1

Background in Text-to-Speech

1.1 Introduction

It is tempting to think of the problem of converting written text into speech as "speech recognition in reverse". Current speech recognition systems are generally deemed successful if they can convert speech input into the sequence of words that was uttered by the speaker, so one might imagine that a TTS synthesizer would start with the words in the text, convert each word into speech (being careful to pronounce each word correctly) and concatenate that result together.

However, when one considers what literate native speakers of a language must do when they read a text aloud, it quickly becomes clear that things are much more complicated than this simplistic approach suggests. Pronouncing words correctly is only part of the problem faced by human readers: in order to sound natural and to sound as if they understand what they are reading, they must also appropriately emphasize (accent) some words, and de-emphasize others; they must "chunk" the sentence into meaningful (intonational) phrases; they must pick an appropriate F_0 (fundamental frequency) contour; they must control certain aspects of their voice quality; they must know that a word should be pronounced longer if it appears in some positions in the sentence, than if it appears in others, since segmental durations are affected by various factors, including phrasal positions.

What makes reading such a difficult task for a machine, is that all writing systems systematically fail to specify many kinds of information that are important in speech. While the written form of a sentence (usually) specifies completely the words that are present, it only specifies partly the intonational phrases – typically with some form of punctuation. The written form will usually not indicate which words to accent or de-accent, and hardly ever gives information on segmental duration, voice quality or intonation. One might think that a question mark '?' indicates that a sentence should be pronounced with a rising intonation: generally though a question mark merely indicates that a sentence is a question, leaving it up to the reader to judge whether this question should be rendered with a rising intonation. The orthographies of some languages – e.g. Chinese, Japanese, and Thai – fail to give information on where word boundaries are, so that even this needs to be figured out by the reader.

The task of a TTS system is thus a complex one, which involves mimicking what human readers do. But a machine is hobbled by the fact that it generally "knows" the grammatical facts of the language only imperfectly, and generally can be said to "understand" nothing of what it is reading. TTS algorithms thus have to do the best they can by making use, whenever possible, of purely grammatical information to decide on such things as accentuation, phrasing, and intonation, and by coming up with a reasonable "middle ground" analysis for aspects of the output that are more dependent on actual understanding.

It is natural to divide the TTS problem into two broad sub-problems. The first of these is the conversion of text – an imperfect representation of language,

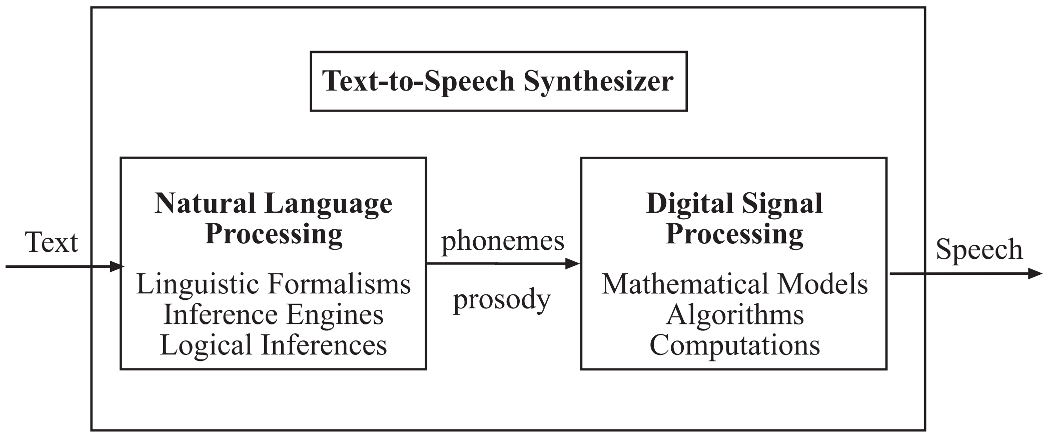

Figure 1.1: Functional Diagram of a TTS system

as we have seen – into some form of linguistic representation that includes information on the phonemes (sounds) to be produced, their duration, the locations and durations of any pauses and the F_0 contour to be used. The second – the actual synthesis of speech – takes this information and converts it into a speech waveform.

Figure 1.1 introduces the functional diagram of a very general TTS synthesizer. It comprises a *Natural Language Processing* module (NLP), capable of producing a phonetic transcription of the text read, together with the desired intonation and rhythm (often termed as *prosody*), and a *Digital Signal Processing* module (DSP), which transforms the symbolic information it receives into speech.

1.2 The NLP Component

The Natural Language Processing block of a synthesizer, is used to perform text and linguistic analysis on the input text, and can be broken down to the following parts:

Text Preprocessing Module Includes end-of-sentence detection, “text normalization” (expansion of numerals and abbreviations), and limited grammatical analysis, such as grammatical part-of-speech assignment.

Word Pronunciation Module Includes the pronunciation of names and the disambiguation of homographs.

Accent Assignment Module Assigns levels of prominence to various words in the sentence.

Intonational Phrasing Module Controls the breaking of (usually long) stretches of text into one or more intonational units.

Segmental Durations Module Determines, on the basis of linguistic information computed thus far, of appropriate durations for phonemes in the input.

F_0 **Contour computation Module**

Figure 1.2 shows the relation between the components of the synthesizer’s NLP subsystem. We will not provide any further details on the workings of the NLP subsystem, as it is beyond the scope of this work.

1.3 The DSP Component

Once the text has been transformed into phonemes, and their associated durations and a fundamental frequency contour have been computed, the system is ready to compute the speech parameters for synthesis.

Intuitively, the operations involved in the DSP module are the computer analogue of dynamically controlling the articulatory muscles and the vibratory

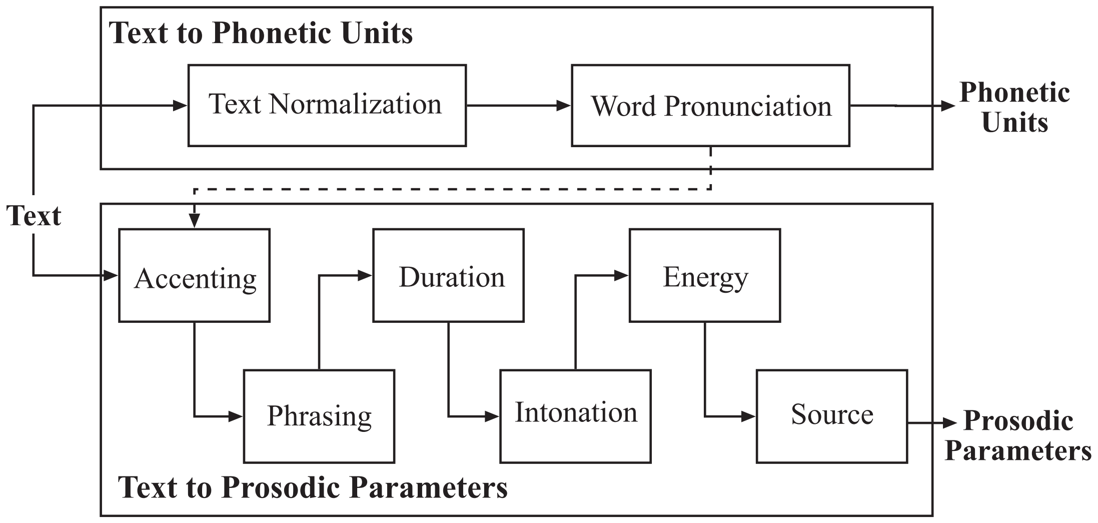

Figure 1.2: The components of the Natural Language Processing subsystem

frequency of the vocal chords so that the output signal matches the input requirements. The DSP module should obviously take articulatory constraints into account, since it has been known for a long time that phonetic transitions are more important than stable states for the understanding of speech. This, in turn, can be basically achieved in two ways:

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

Two main classes of TTS systems have emerged from this alternative, which quickly turned into synthesis philosophies given the divergences they present in their means and objectives: *synthesis-by-rule* and *synthesis-by-concatenation*.

1.3.1 Rule-Based Synthesizers

Rule-based synthesizers are mostly in favor with phoneticians and phonologists, as they constitute a cognitive, generative approach of the phonation mechanism. Rule-based approaches are space-efficient, since they eliminate the need to store speech segments, and they also make it easier in principle to implement new speaker characteristics for different voices, as well as different phone inventories for new dialects and languages. These systems are also restrictive regarding the choice of the parametric representation of speech, since such schemes rely both on our understanding of the relation between the parameters and the acoustic signals they represent, and on our ability to compute the dynamics of the parameters as they move from one sound to another. Thus far only articulation parameters and formants have been used in rule-based systems.

Most such systems describe speech as the dynamic evolution of up to 60 parameters, mostly related to formant and anti-formant frequencies and bandwidths together with glottal waveforms. Clearly, the large number of (coupled) parameters complicates the analysis stage and tends to produce analysis errors. Moreover, formant frequencies and bandwidths are inherently difficult to estimate from speech data. The need for intensive trials and errors, in order to cope with analysis errors, makes them time-consuming systems to develop – several years are commonplace. Yet, the synthesis quality achieved up to now reveals typical buzzyness problems, which originate from the rules themselves: introducing a high degree of naturalness is theoretically possible, but the rules to do so are still to be discovered.

Rule-based synthesizers remain, however, a potentially powerful approach to speech synthesis. They allow, for instance, to study speaker-dependent voice features so that switching from one synthetic voice into another can be achieved with the help of specialized rules in the rule database. Under the same reasoning, synthesis-by-rule seems to be a natural way of handling the articulatory aspects of changes in speaking styles, as opposed to their prosodic counterpart, which can be accounted for by concatenation-based synthesizers as well. No wonder then that it has been widely integrated into TTS systems (MITalk [1], DECtalk [25] and the JSRU synthesizer [26] for English).

1.3.2 Concatenative Synthesizers

As opposed to rule-based ones, *concatenative synthesizers* possess a very limited knowledge of the data they handle: most of it is embedded in the segments to be chained up. This clearly appears in Figure 1.3, where all the operations that could indifferently be used in the context of a music synthesizer (i.e. without any explicit reference to the inner nature of the sounds to be processed) have been grouped into a *sound processing* block, as opposed to the upper *speech processing* block whose design requires at least some understanding of phonetics.

Database Preparation

A series of preliminary stages have to be fulfilled before the synthesizer can produce its first utterance. At first, segments are chosen so as to minimize future concatenation problems. A combination of diphones, i.e. units that begin in the middle of the stable state of a phone and end in the middle of the following one, half-syllables, and triphones – which differ from diphones in that they include a complete central phone – are often chosen as speech units, since they involve most of the transitions and co-articulations while requiring an affordable amount of memory. When a complete list of segments has emerged, a corresponding list of words is carefully completed, in such a way that each segment appears at least

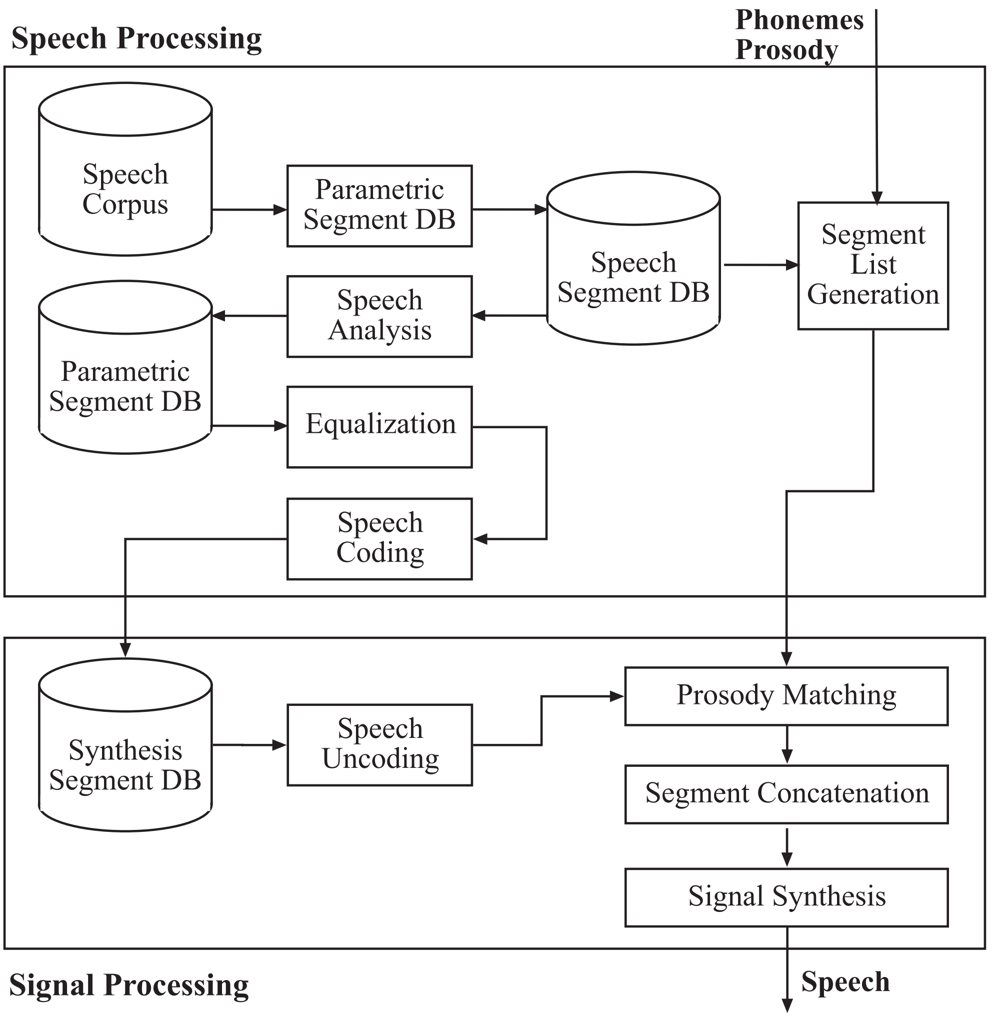

Figure 1.3: A general concatenation-based synthesizer. The upper left hatched block corresponds to the development of the synthesizer (i.e. it is processed once for all). Other blocks correspond to run-time operations.

once, although twice is better, for redundancy reasons. Unfavorable positions, like inside stressed syllables or in strongly reduced, (i.e. over-co-articulated) contexts, are excluded. A corpus is then digitally recorded and stored, and the elected segments are spotted, either manually with the help of signal visualization tools, or automatically, using segmentation methods borrowed from speech recognition, the decisions of which are checked and corrected interactively. A segment database finally centralizes the results, in the form of the segment names, waveforms, durations, and internal sub-splittings. In the case of diphones, for example, the position of the border between phones should be stored, to be able to modify the duration of one half-phone without affecting the length of the other one.

Segments are then often given a parametric form, in the form of a temporal sequence of vectors of parameters collected at the output of a speech analyzer and stored in a parametric segment database. Using a speech model has the following advantages:

- Well chosen speech models allow data size reduction, an advantage which is hardly negligible in the context of concatenation-based synthesis given the amount of data to be stored. Consequently, a parametric speech coder often follows the analyzer.
- A number of models explicitly separate the contributions of the source and the vocal tract, an operation that remains helpful for the pre-synthesis operations: prosody matching and segment concatenation.

Indeed, the actual task of the synthesizer is to produce, in real-time, an adequate sequence of concatenated segments, extracted from its parametric segment database. The prosodic characteristics of these segments, like intonation and duration, have been adjusted from the originally stored values, to the ones imposed by the language processing module. Consequently, the prosody matching and segment concatenation modules are considerably alleviated when input segments are presented in a form that allows easy modification of their pitch, duration, and spectral envelope, as is hardly the case with crude waveform samples.

Since segments to be chained up have generally been extracted from different words, that is in different phonetic contexts, they often present amplitude and timbre mismatches. Even in the case of stationary vocalic sounds, for instance, a rough sequencing of parameters typically leads to audible discontinuities. We can typically cope with this problem during the constitution of the synthesis segments database, using an equalization algorithm in which related endings of segments are imposed similar amplitude spectra, the difference being distributed on their neighborhood. In practice, however, this operation, is restricted to amplitude parameters: the equalization stage smoothly modifies the energy levels at the beginning and at the end of segments, eliminating amplitude mismatches by setting the energy of all the phones of a given phoneme to their average value. In contrast, timbre conflicts are better tackled at run-time, by smoothing individual couples of segments on demand rather than equalizing them at a preprocessing stage. This way, some of the phonetic variability naturally introduced by co-articulation is still maintained. In practice, amplitude equalization can be performed either before or after speech analysis, i.e. on crude samples or on speech parameters.

Once the parametric segment database has been completed, synthesis itself can begin.

Speech Synthesis

A sequence of segments is first deduced from the phonemic input of the synthesizer, in a block termed as *segment list generation* in Figure 1.3, which interfaces the NLP and DSP modules. Once prosodic events have been correctly assigned to individual segments, the prosody matching module queries the synthesis segment database for the actual parameters, of the elementary sounds to be used, and adapts them one by one to the required prosody. The segment concatenation block is then in charge of dynamically matching segments to one another, by smoothing discontinuities. Here again, an adequate modeling of speech is beneficial, provided that simple interpolation schemes performed on its parameters approximately correspond to smooth acoustical transitions between sounds. The resulting stream of parameters is finally presented at the input of a synthesis block, the exact counterpart of the analysis one. Its task is to produce speech.

Segmental Quality

The efficiency of concatenative synthesizers to produce high quality speech is mainly subordinated to:

1. The type of segments chosen.
2. The model of speech signal, to which the analysis and synthesis algorithms refer.

Segments should obviously exhibit some basic properties:

- They should account for as many co-articulatory effects as possible.
- Given the restricted smoothing capabilities of the concatenation block, they should be easily connectable.

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

On the other hand, longer units decrease the density of concatenation points, therefore providing better speech quality. Similarly, an obvious way of accounting for articulatory phenomena is to provide many variants for each phoneme. This is clearly in contradiction with the limited memory constraint, thus calling for some trade-off between quality and memory requirements.

Diphones are often chosen. The inventory size is reasonable (about 1200 for French, including lots of phoneme sequences that are only encountered at word boundaries, equivalent to 3 minutes of speech, i.e. approximately 5 Mbytes of 16 bits samples at 16 kHz) and they do incorporate most phonetic transitions. They imply, however, a high density of concatenation points (one per phoneme), which reinforces the importance of an efficient concatenation algorithm. Besides, they can only partially account for the many co-articulatory effects of a spoken language, since these often affect a whole phone rather than just its right or left halves independently. Such effects are especially prominent when somewhat transient phones, such as liquids and semi-vowels, are to be connected to each other. Hence, people have used some larger units as well, such as triphones.

The models used in the context of concatenative synthesis can be roughly classified into two groups, depending on their relationship with the actual phonation process.

- Production Models, and
- Phonological Models

Production models provide mathematical substitutes for the part respectively played by vocal chords, nasal and vocal tracts, and by the lips radiation. Their most representative members are Linear Prediction Coding (LPC) synthesizers

[34], and the formant synthesizers we mentioned in the Section about Rule-based synthesizers.

On the contrary, *phenomenological models* intentionally discard any reference to the human production mechanism. Among these pure digital signal processing tools, spectral and time-domain approaches are increasingly encountered in TTS systems. Two leading such models exist: the hybrid Harmonic/Stochastic (H/S) model of [35] and the Time-Domain Pitch-Synchronous-OverLap-Add (TD-PSOLA) of Charprentier and Moulines [16]. The latter is a time-domain algorithm: it virtually uses no explicit speech model. It exhibits very interesting practical features: the best currently available high speech quality combined with a very low computational cost – 7 operations per sample on the average. The hybrid Harmonic/stochastic model is intrinsically more powerful than the TD-PSOLA one, but it is also about ten times more computationally intensive. PSOLA synthesizers are now widely used in the speech synthesis community. The recently developed MBROLA algorithm [23, 24] even provides a time-domain algorithm which exhibits the very efficient smoothing capabilities of the H/S model (for the spectral envelope mismatches that cannot be avoided at concatenation points) as well as its very high data compression ratios (up to 10 with almost no additional computational cost) while keeping the computational complexity of PSOLA.

Chapter 2

Robust Unit Selection

Much of the complexity and indirectness of the relationship between the acoustic patterns of speech and the linguistic structures that they represent is caused by context-sensitivity.

> Speech does not consist simply of a string of articulations linked by simple movement between them. Instead, the articulation of individual segments is almost always influenced by the articulation of neighboring segments, often to the point of considerable overlapping of articulatory activities. As a consequence, the notional or "ideal" way of articulating a particular sound is subject to modifications in running speech.
>
> Context-sensitive variation has complex and interacting causes which are not yet completely understood. Two basic types can be distinguished: (1) the effects of the biomechanical performance of the vocal tract, and (2) the effects of the nature and organization of the neuromuscular control mechanisms which activate articulator movement. Both types may reflect the level of articulator performance that is sufficient to produce adequate phonetic distinctiveness in the language

in question.

The vocal tract, including the articulators within it, forms a biomechanical system which is subject to the laws governing all mechanical systems. The mass and size of the articulators constrain their movement in relation to the muscle systems that actuate them. Articulators have mass and are subject to inertia: they resist being set in motion.[17]

2.1 Introduction

Current unit–selection synthesizers are sufficiently effective in synthesizing high quality speech, most of the time [6]. However, quality is subject to notable fluctuations, ranging from speech which is almost indistinguishably human, to robot-like and jittery synthesized utterances that can immediately betray the fact that the speaker is a machine rather than a human.

This behavior can be attributed to several factors, the most prominent of which is the system's failure to select a series of acoustically appropriate units with the prosodic characteristics required by the utterance that needs to be synthesized. Unit–selection synthesizers might be producing much more natural sounding examples, but in the process of moving from diphone to unit–selection synthesis, we have given up a certain amount of control over what is synthesized. The quality of the synthesized speech is highly dependent on the quality of the speech database; the type of units stored in it, the texts that were used for recording the database, the genres, domains and the speaking styles covered during the recordings, all affect the quality of the produced speech. At the same time, little if any prosodic and spectral modification is done to the selected units, and this reduces the flexibility of unit–selection synthesizers to reach natural sounding speech.

In this chapter we are investigating the reasons that result to the aforementioned variations in speech quality observed in all contemporary unit-selection based synthesizers. We will focus our investigation mostly on the impact of the speech synthesis component, and ignore the text/prosodic analysis components that come before it. We will analyze each of the subsystems that make it up, and discuss the possible modifications that might be performed on each of them in order to improve the robustness of the unit selection process. Our goal is to modify the current framework for speech synthesis, in order to achieve consistently high quality in the synthesized speech.

2.2 Background

The speech synthesis process, as implemented by most unit selection synthesizers, consists of the following three main tasks:

- *Text processing*, which includes sentence parsing, text normalization, phrasing and pronunciation generation.
- *Linguistic processing*, during which the system applies prosodic models to predict intonation, accents and segmental timing.
- *Waveform synthesis*, which produces speech from the specifications that are imposed by the previous two stages.

Ng [36], in a survey of data-driven approaches used in speech synthesis, identifies the following components in concatenative synthesizers which implement the task of *Waveform Synthesis*:

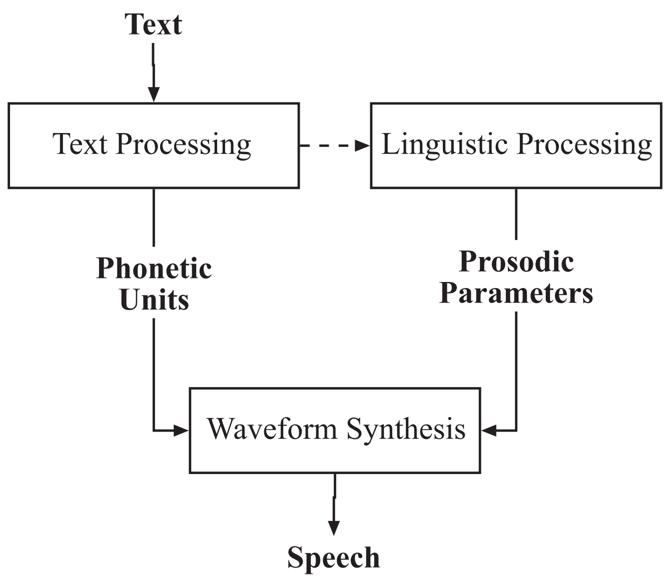

Figure 2.1: Information Flow through a Speech Synthesis System

Speech Corpus The speech corpus serves as the source of synthesis units and generally consists of speech recorded from a single speaker. It is made up of the speech waveforms and the corresponding orthographic transcriptions of these utterances.

Speech Segmentation and Labeling The speech corpus is segmented into speech units and each of them is labeled, both phonetically as well as prosodically.

Unit Inventory Design The goal of this component is to come up with a set of speech segments that covers all the phonetic variations in the language and at the same time accounts for as many coarticulatory effects as possible, has minimal discontinuities at the concatenation points and is as small as possible. The design of the unit inventory is the most important factor in a concatenative synthesis system, since it impacts almost any other processing component. The content of the unit inventory determines the type and size of the speech corpora needed, how the synthesis units are selected and concatenated, and what type of signal processing can and needs to be done.

Speech Coding and Storage The unit inventory is indexed and stored so that it can be accessed during synthesis. The system might either store the raw waveforms of perform some speech analysis on the segments, and store the resulting analysis parameters.

Unit Selection The goal of the unit selection process is the choice of the best units in terms of phonetic identity to match the sound to be produced, phonetic context to capture the coarticulation effects, and prosodic characteristics to match the prosody as closely as possible.

Unit Concatenation After selecting the best sequence of speech segments from the unit inventory, these segments have to be concatenated into a contiguous stream. The transition from one segment to the next has to be as smooth as possible, by minimizing the discontinuity at the boundaries. This can be achieved through signal modification, but it is preferred that such modifications are kept to a minimum, by selecting units in a manner that minimizes boundary discontinuities.

Prosodic Signal Processing In addition to the correct sequence of phonemes, the prosody needs to be appropriate. Many systems perform some type of signal processing to modify the duration, fundamental frequency and energy. However, such signal processing can degrade the naturalness of speech, and it can be kept to a minimum by providing a prosodically rich

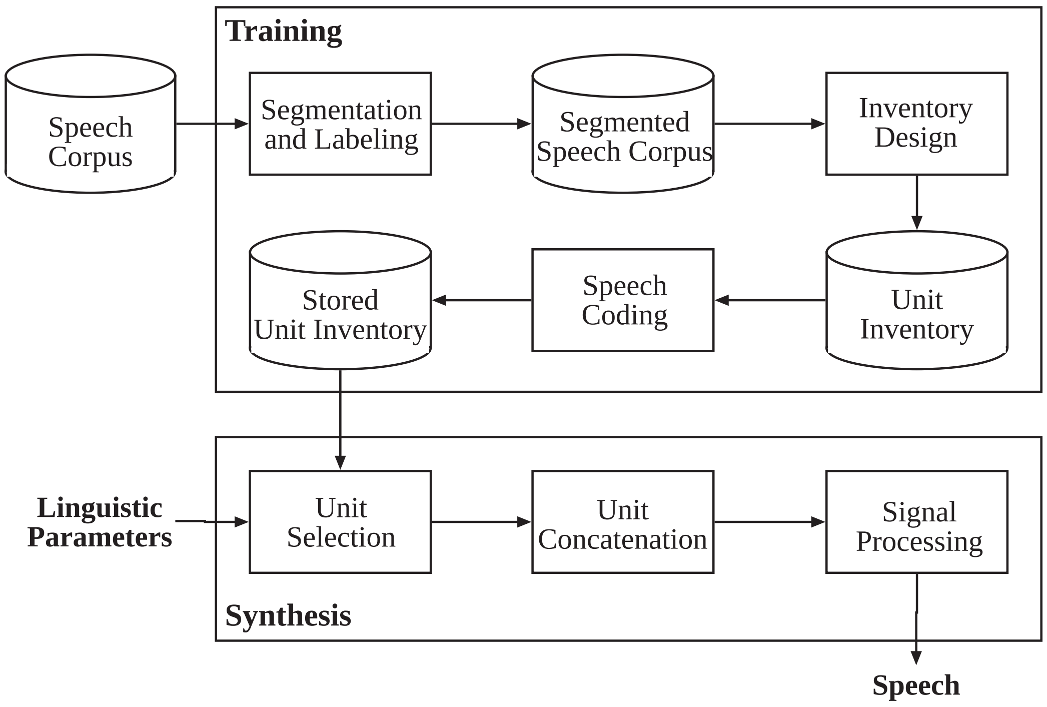

Figure 2.2: Components of the Waveform Synthesis Module of a Unit Selection Synthesizer

unit inventory, thus enabling well-matched units to be selected during synthesis.

As illustrated in Figure 2.2, there are two major subsystems that implement this process: one for *training* and one for *synthesis*.

In training, an inventory of synthesis units is created from a corpus of speech data. This is a preparation stage, taking place only once during the development of a new voice for the speech synthesizer.

At synthesis time, the combined output of the first two stages (text and linguistic processing) of the speech synthesis process is passed on to the Synthesis subsystem. The input to this subsystem is a series of feature vectors, specifying the predicted characteristics of the target units that should be used in order to synthesize the requested utterance. These features typically include information such as phoneme label, duration, power and F_0 [7]. The task of the *Unit Selection* component is to use this set of specifications in order to select an optimal sequence of units. These units are passed to the *Unit Concatenation* and *Prosodic Signal Processing* components in order to reconstitute a sequence of speech-describing symbols, aiming at synthesizing natural-sounding, artifact-free speech [46].

2.2.1 Unit Inventory Design

The Unit Selection subsystem has direct access to the synthesizer's *unit inventory.* This is the inventory of all unit instances that can be used by the *Synthesis* subsystem in synthesizing the requested utterance. A *unit instance* is a segment extracted from the collection of speech data, also referred to as the *speech database*, that contains the recorded speech of a single speaker (usually referred to as a voice talent) who was instructed to read a carefully selected assortment of texts. These texts are chosen with two goals in mind [18]:

- To contain sufficient examples of phoneme-phoneme pairs (also known as *diphone coverage*)
- To contain a wide variety of prosodic styles and contexts, achieved by including material from newspaper text and interactive prompt-style utterances

Unit Inventory Size

The size of the unit inventory is also a design decision. A relatively small unit inventory, which contains few or even a single unit instance per unit class, is much easier to manage, and does not impose any excessive computational requirements upon the system. However, almost all contemporary speech synthesizers are using relatively large unit inventories, equivalent to ten to twenty hours of speech. The primary motivation for the use of such large unit inventories is that with a large number of unit instances available per unit class which account for a variation of prosodic and spectral characteristics, it is possible to synthesize more natural–sounding speech than can be produced with a small set of controlled units [27]. In this way, it is no longer necessary to synthesize using only the most typical unit instance for all prosodic contexts. Instead, we can select from several unit instances and we can better match the requested prosodic specifications for the synthesized utterance [15].

Unit Instance Size

The size of these unit instances varies among different implementations of speech synthesizers. The actual type of the unit is a design decision in itself. In all cases, a unit must encode linguistic, intonational, and sometimes paralinguistic information. However, units are typically at the phone level (phones, half-phones, diphones) and recent implementations are moving towards variable-sized segments that can scale up to whole words [45].

The decision on the size of the unit instances for a given language is affected by the designer's decisions regarding the other characteristics of the unit inventory database. In making this decision, the designer is presented with the following challenges [39]:

- Units should cover all legal phone sequences for that language, including *inter-word combinations.*
- All the necessary phonemic and allophonic variations should be inflected in

the inventory structure.

- The concatenation of two or more inventory units should not produce audible discontinuities in the resulting synthetic speech.
- The inventory should have a manageable size.

Coverage is probably the most important aspect of this decision. In an analysis presented by Sproat in [39], the author discusses the advantages and disadvantages of introduced by the use of several types of units. Large–size units tend to produce better speech, but they also require huge unit inventories. Thus, although triphones might produce better quality synthetic speech, a triphone based synthesizer is almost impossible to construct. For instance, for a language like American English, assuming a 43-phone alphabet, we can have 79,507 possible triphones. However, some 9,000 such combinations are impossible according to the language's phonotactics, leaving us with almost 70,000 such triphones. However, each of them can occur in thousands of contexts, some of which might have strong coarticulatory effects. Even if we ignore the technical difficulties imposed by the requirements for tightly controlled speaking and recording conditions, a highly trained professional speaker could not produce more than $1,000 - 2,000$ successfully rendered and recorded units, making it really difficult to record the extensive material needed for the construction of the unit inventory.

The same problems are also present in inventories utilizing smaller–sized units, albeit they might be less severe. In the case of diphones, there are fewer combinations to be covered, but the problem of cross–unit coarticulation might be more becomes more important, thus demanding a more carefully–designed set of texts that will be used for the recordings. Phone–sized units also present the same problems with diphones.

The most prominent problem related with unit inventories comprising of phones or diphones is the audible spectral mismatch which might occur after joining two units during concatenation. As mentioned above, it is possible to use certain signal processing techniques in order to remove such artifacts; however, such processing will affect the speech signal in its totality, resulting in significant quality degradation. The only remedy to this problem is the inclusion of multiple units recorded in various contexts, in order to account for the coarticulatory phenomena and the different prosodic characteristics that might be required for the synthesis of the given utterance.

In any case, it seems that there is no single unit size that will serve as the panacea of the problems related to the design and implementation of the unit inventory. However, there is evidence that a unit inventory based on diphone or half-phone sized units, which is enriched with frequent, longer units, in order to include their intra-segment coarticulatory effects, can prove to be the best combination of quality and manageability.

Speech Coding and Storage

Originally, unit instances were stored in speech databases in some parametric format, encoded with techniques such as linear prediction coefficients [34], sinusoidal modeling and harmonic-noise modeling [40]. However, reduced storage costs have made the use of non–parametric models economically feasible. The move towards non-parametric speech representations has had positive impact on quality, allowing for the production of more clear and natural–sounding speech.

2.2.2 Unit Inventory Metadata

Independently of whether a parametric or non-parametric scheme is used in storing units in the speech database, each unit instance is associated with a predefined set of metadata. Typical features, as discussed in [7], are the following:

- *linguistic feature*, such as phoneme label, duration, power and F_0
- *acoustic features*, such as spectral tilt
- *context-related features*, such as the phoneme labels of neighboring units, position in phrase, direction of pitch/power change, etc.

Features can be *continuous* or *discrete.* They are stored along with the unit segment in the speech database in a hierarchical or flat format. Feature values are usually stored in a normalized form, so that objective comparisons between features can be made easily. Comparisons between features of the same type can be performed by using distance measures. These measures can be implicitly defined for continuous features using the actual difference between two features' values, but in the case of discrete features, distances have to be explicitly defined. In both cases, distances are normalized and take values in the range between 0 (good) and 1 (bad).

2.2.3 Target Features

As mentioned above, the first two stages of synthesis (text and linguistic processing) transform the input text into a target specification, expressed in terms of a subset of the features presented above. Any features which are available only from acoustic measures are excluded from this subset.

This target specification defines the string of phonemes that are required in order to synthesize the desired utterance, and the prosodic features (such as pitch, duration and power) that the selected units should have, explicitly specifying the intended utterance's segmental and prosodic characteristics.

2.3 Unit Selection

Speech synthesis algorithms attempt to generate the acoustic features of speech using as input only the linguistic features produced by the text/prosodic analysis components of the synthesizer. Unit–selection synthesizers achieve this by predicting the appropriateness of a particular unit instance extracted from the unit inventory, using only these linguistic targets.

A formal definition of the unit selection process is given by Macon et al. in [33]:

> Given a sequence of linguistic target features $l_1^t, l_2^t, \ldots, l_n^t$, find the sequence of candidate segments with linguistic features $l_1^c, l_2^c, \ldots, l_n^c$ that will minimize an acoustic feature distance between the target specifications and the candidate segments:
> $d(t_1, t_2, \ldots, t_n, u_1, u_2, \ldots, u_n)$

The linguistic target features l^t are the product of the text/prosodic analysis components of the synthesizer, and form the input to the unit selection subsystem. In contrast, the acoustic target features a^t are not known at synthesis time; these must be predicted automatically. However, not all possible combinations of linguistic feature contexts will be available in the unit inventory. Thus, some of the linguistic targets l^t will not have been seen previously in the database, and the algorithm will have to generalize and find outputs for these cases.

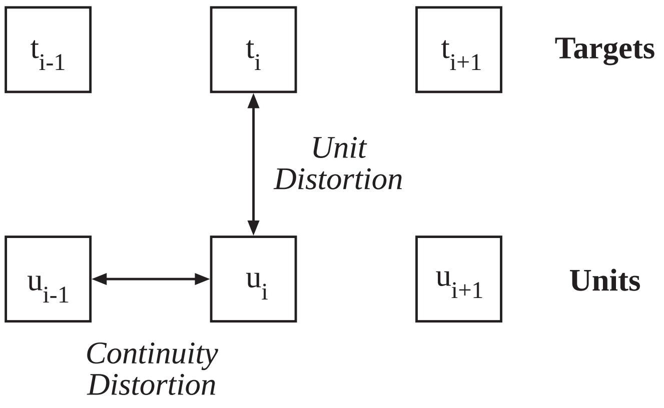

Figure 2.3: Distortion Types in Unit Selection

Hunt and Black [27] use the following two distortion measures that can be used in order to express the problems related with unit selection:

- **Unit Distortion** $D_u(u_i, t_i)$ This is the distance between a selected unit and a target segment, i.e. the difference between the selected unit's linguistic feature vector and the target segment's vector, multiplied by an appropriate weights vector.

- **Continuity Distortion** $D_c(u_i, u_{i-1})$ This is the distance between a selected unit and its immediately adjoining previous selected unit, i.e. the difference between the selected unit's acoustic feature vector and the acoustic feature vector of the previous unit, multiplied by an appropriate weights vector.

Figure 2.3 graphically shows how these two distortion types are defined among a series of target and candidate units. Qualitatively, unit distortion measures the suitability of a particular unit instance with regard to the target specifications. Continuity distortion measures the suitability of a particular unit instance with regard to the previously selected one. The function $d(\cdot, \cdot)$ presented above, can

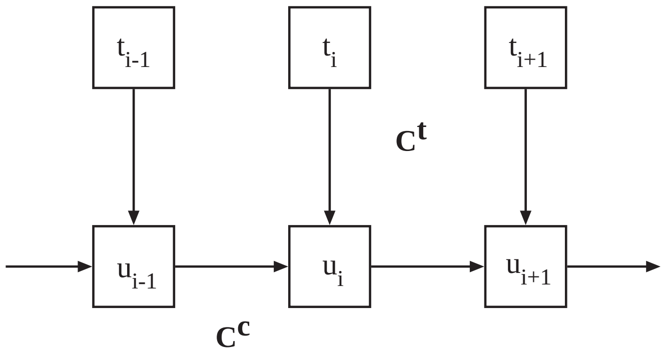

Figure 2.4: Unit Selection Costs

take into consideration both the unit distortion between a target and its associated candidate unit, and the continuity distortion between successive candidate units.

2.3.1 Cost Functions

Unit selection in most contemporary speech synthesis systems is based on the use of two cost functions that utilize the information quantified by the unit and continuity distortion measures. Figure 2.4 shows the relation between the two cost functions.

For the target unit (phone specification) t_i and the database (acoustic) unit u_i, which is a candidate to be used as a segment for realizing this part of the required utterance, we define the following two costs:

- **Target Cost** - $C^t(u_i, t_i)$

  The target cost is an estimate of the mismatch between a recorded unit and the requested target specification. We use this cost in order to choose "appropriate" units that are a good fit to the requested specification. Such units will not require any significant signal processing, and preferably they can be used without any signal processing at all.

- **Concatenation (or Join) Cost** - $C^c(u_{i-1}, u_i)$

  The concatenation cost is an estimate of the quality of the join between the two consecutive units u_{i-1} and u_i. It provides a measure for the acoustic mismatch, and we can use it to achieve smooth segmental joins between units. This cost is independent of the target specifications for the segment in review, and is only dependent upon the acoustic features of the selected unit and the previous unit.

2.3.2 Features

Both target and concatenation costs are actually defined as the weighted sums of sub-costs related with the effects of unit and continuity distortion, respectively. Each of these sub–costs is dependent upon one of the features used by the unit–selection model. Black and Campbell in [7] and Hunt and Black in [27] identify the following features:

- Unit Distortion Related Features

  phonetic context, log power, mean F_0

- Continuity Distortion Related Features

  phonetic context, prosodic context (duration, power and F_0 together), and acoustic join cost

In the case of target costs, sub–costs are defined as the distances (or differences) between the elements of the target and candidate feature vectors. Typical implementations use between 20 and 30 such features, with an appropriate set of weights that affect the impact of each of these sub–costs on the final target cost.

So, for a case of p target features, the target cost, given the weights w^t_j and the target sub-costs $C^t_j(t_i, u_i)$, is calculated as follows:

$$C^t(t_i, u_i) = \sum_{j=1}^{p} w^t_j C^t_j(t_i, u_i) \tag{2.3.1}$$

Concatenation sub–costs are typically fewer in number than target sub-costs. They can usually be determined from the unit characterizations of u_{i-1} and u_i, that is the previously selected acoustic unit and the current one. However, these sub-costs can also be derived from signal processing of the units. For the case of q concatenation/join features, the concatenation/join cost, given the weights w^c_j and the concatenation sub-costs $C^c_j(u_{i-1}, u_i)$, will be:

$$C^c(u_{i-1}, u_i) = \sum_{j=1}^{q} w^c_j C^c_j(u_{i-1}, u_i) \tag{2.3.2}$$

2.3.3 Unit Selection Model

Given the two cost functions presented above, the task of unit selection can be reduced to the parallel minimization of both target and concatenation costs, aiming at the production of an utterance which is made up of units that are as similar to the desired output as possible and which will present a minimum pair-wise perceptible acoustic mismatch when concatenated.

Thus, the total cost for a sequence of n units that can be used in order to form the requested utterance, will be the sum of these two costs, accumulated across all units in the sequence:

$$C(t^n_1, u^n_1) = \sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=2}^{n} C^c(u_{i-1}, u_i) + C^c(S, u_1) + C^t(u_n, S) \tag{2.3.3}$$

where S denotes silence (meaning either the beginning or the end of a phrase), $u^n_1 = (u_1, u_2, \ldots, u_n)$, $t^n_1 = (t_1, t_2, \ldots, t_n)$ and $C^c(S, u_1)$ and $C^t(u_n, S)$ define the start and end conditions defined by the concatenation of the first and last units to silence.

The weights used with the two different sub-costs (w_j^c and w_j^t) determine the relative importance of the different features in the calculation of each of these sub–costs. These weights are determined automatically, by using the data present in the speech database. There are two approaches to this automatic training: Hunt and Black in [27] suggested both limited weight–space search and regression training as methods that can be used in identifying a good set of weights for the cost functions. Both strategies use targets extracted from natural utterances held out from the segment database, and aim at determining the values of the weights that will minimize the difference between the original waveform, and the one produced by the synthesizer. The comparison between the natural and the synthesized segments is performed by using the mean cepstral distance. Evaluation tests performed using the automatically estimated weights have shown that they result to systems that produce consistently better results than that produced by hand-tuned weights.

More formally, we can use the cost function expression in Equation (2.3.3), and define the unit–selection procedure as the task of determining the set of units so that the total cost is minimized:

$$\overline{u}_1^n = \arg \min_{u_1,\ldots,u_n} C(t_1^n, u_1^n) \tag{2.3.4}$$

The selection of the set of units that minimize the overall cost is done by performing a Viterbi search over the network of all possible candidate units. Figure 2.5 shows the Viterbi search over a sample search network created for the utterance ”two”. A dynamic programming method, such as Viterbi search with additional beam–width constraints, is preferable to any exhaustive search method

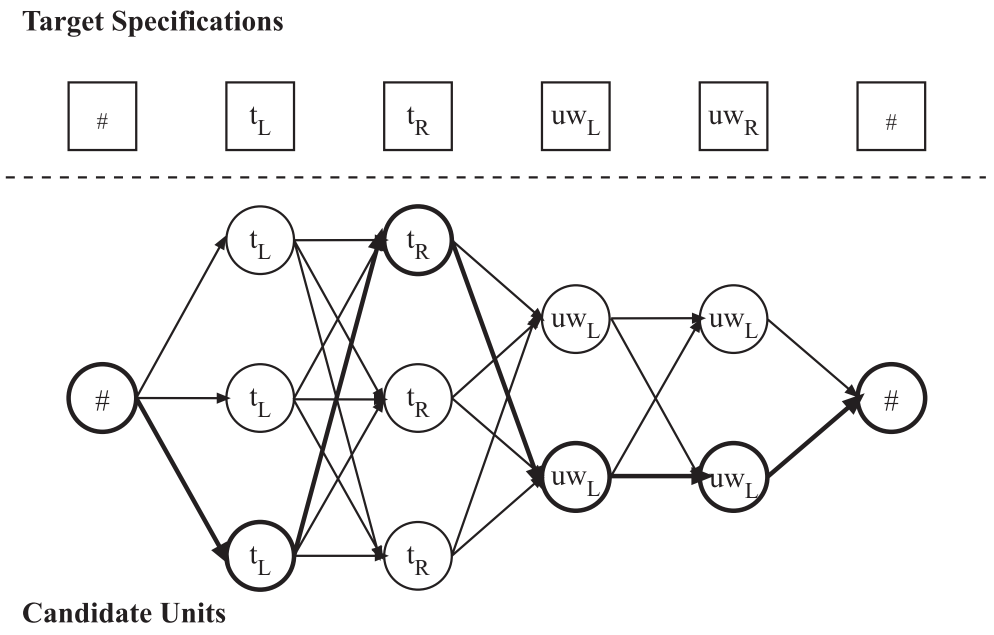

Figure 2.5: Viterbi search for the Utterance “two”, with best path highlighted

due to the computational load it would impose on the unit–selection process.

The process for creating the search network and for performing the search is described by Black and Campbell in [7]. Starting with the set of target specifications, then, for each target segment the algorithm takes the following steps:

- Finds all the units in the database with low phonetic distance from the targets
- Finds the unit distortion

This analysis was first formalized by Black and Campbel in [7], and was further refined by Hunt and Black in [27]. It was first put to use in ATR’s CHATR synthesizer [9]. Since then, it has been used in the implementations of most state–of–the–art TTS systems (for instance, Edinburgh University’s Festival system [5] and AT&T’s NexGen TTS system [2]). However, as noticed in [3] in practice the sub-cost computations are less elegant, employing heuristics at several points, as for example by treating stops in a special manner, or by predisposing mid-phone concatenations.

Optimization

Unit selection is the most demanding stage of the synthesis process in terms of processing power, memory and time. It imposes serious demands upon the synthesizer, especially if synthesis is to be performed in real-time, or even a fraction of real-time, as expected from most commercial synthesizers. Therefore, some kind of optimization in the unit–selection process is necessary.

The speeding up of the process of runtime unit selection can be achieved through the following strategies:

- By limiting the number of candidate synthesis units that can are considered in the unit selection process, the number of calculations required can be reduced.
- By pre-computing part of the needed calculations, the runtime complexity can be reduced.

Both approaches aim at lowering the runtime computational demands, while at the same time maintaining speech synthesis quality.

The most common optimization is the pruning of the unit database [21]. This technique employs a form of off-line pre-selection algorithm in order to determine the subset of development data which can be shipped with the runtime synthesis system and will introduce the minimum degradation in synthesis quality.

Another method that allows performance gains at run-time is the pruning of the search space during unit selection. This is done in multiple stages. As suggested by Black and Taylor in [10], only units belonging to the same cluster, i.e. units with phonetic contexts similar to the target, are considered for a given target. Next, these units are pruned using the target cost and finally the concatenation cost. Bulyko [13] suggested the use of an additional cost, namely *slicing cost*, which allows the selection of units with the aim of minimizing potential discontinuity that a given unit may incur when a splice is made at its boundary, irrespective of the adjoining unit. Pruning the search space based on this cost prior to the computation of the concatenation costs, further reduces the number of candidate units that need to be considered in the end.

Finally, computation caching can provide significant performance gains through the reduction in the computational load at synthesis time. By pre-computing and caching concatenation costs between the most frequently used pairs of units, as implemented by Beutnagel et al. [3], the reported complexity reduction in unit selection was of a factor of four, without any significant decrease in synthesis quality. This method was further improved in [19], where the unit space is vector quantized and a complete distance table between groups of units with closely related contexts is pre-computed and stored. The reported complexity reduction in this case was even higher, close to a factor of ten, and again without negative effects in the quality of the synthesized signal.

Chapter 3

Defining Join Costs using Linear Segment-Based Models

During the last decade there has been a fundamental paradigm shift in the approach used in designing and implementing Text–to–Speech systems. In fact, both experimental and commercial TTS platforms are now mostly based on data-driven techniques, and most of the development effort is now placed into implementing optimizations in engineering aspects of the unit–selection process, rather than the development of linguistic rules and the understanding of the speech generation.

This paradigm shift has led to the adoption of a large number of ideas and techniques borrowed from automatic speech recognition and their application in TTS. In this chapter we will show how yet another ASR technique, namely a Linear Segment-Based Acoustic Model, can be used in order to improve the unit-selection process.

3.1 Introduction

As we already mentioned in previous chapters, unit–selection synthesizers produce speech by selecting the optimal series of speech units from the unit database.

The database contains multiple instances of each unit class, and the decision regarding the segments that will be concatenated in order to produce the synthetic speech is based on a combination of two costs: target cost (how closely candidate units in the inventory match the required targets) and join cost (how well neighboring units can be concatenated). The optimal sequence is then determined after performing a Viterbi search on the network of all candidate units, identifying the path with the lowest cost.

However, the issue of defining these costs in such a way that will lead to perceptually optimal signal quality is still open. Join costs in particular, although based solely on measurable properties of the candidate units, such as amplitude, F_0 and spectral parameters, need to be defined in such a way that will lead to the selection of units that will minimize audible spectral discontinuities at their concatenation points. The high correlation between those features and the perception of the quality of the synthesized audio requires further research into the design of the unit cost function.

3.2 Motivation

In studies performed by Klabbers and Veldhuis [29, 30], Wouters and Macon [44], Stylianou and Syrdal [41] and Donovan [22], there was an effort to identify distance measures and criteria that can help in predicting audible discontinuities in an objective manner. However, most of these studies were focused on the reduction of such audible spectral discontinuities in isolated words generated by a concatenative synthesizer. Our aim is to define a distance measure that will be used in determining join costs and reduce audible spectral discontinuities on the utterance level.

In doing this, we will borrow a tool used in Speech Recognition, namely an *acoustic model* of speech. Speech can be viewed as a *discrete message source* with a *hierarchical structure*: phonemes are joined to form syllables, then words, phrases and finally continuous discourse. In modern speech recognition systems, stochastic models are postulated for certain units of speech, such as words or phonemes. The basic *acoustic models* are then combined to form models for larger units, with the aid of dictionaries, and/or probabilistic grammars.

We will use an implementation of such an acoustic models, namely a *linear dynamic model* for the definition of a distance measure that will be used for defining the join cost function. The acoustic model will be used to predict the parameters of the most likely observation.

Furthermore, we will use a segment-based model describing speech, as opposed to traditional, Hidden Markov Model (HMM) based approaches. Since such models use *segments*, defined as variable-duration parts of the speech waveform, usually corresponding to language units, they are able to represent higher-order phenomena and utilize features extracted from a longer time-scale processing of the signal than the typical 10-20 msec analysis window used in HMM acoustic models.

3.3 Linear Stochastic Systems

A linear dynamical system is described by:

$$x_{k+1} = Fx_k + w_k \tag{3.3.1}$$

$$y_k = Hx_k + v_k \tag{3.3.2}$$

where the hidden state x_k is a $(n \times 1)$ vector with initial value at $k = 0$, the observation y_k is $(m \times 1)$, and w_k, v_k are uncorrelated, zero-mean Gaussian vectors with covariances

$$E\{w_k w_l^T\} = Q\delta_{kl} \tag{3.3.3}$$

$$E\{v_k v_l^T\} = R\delta_{kl} \tag{3.3.4}$$

where δ_{kl} is the Kronecker delta and T denotes the transpose of a matrix. We further assume that the initial state x_0 is Gaussian, with known mean and covariance μ_0, Σ_0.

Maximum likelihood estimates of the unknown parameters θ in F, H, Q, R can be obtained by minimizing the negative log likelihood, or equivalently the quantity

$$J(Y,\theta) = -L(Y,\theta) = \sum_{k=0}^{N}\{\log|\Sigma_{e_k}| + e_k^T(\theta)\Sigma_{e_k}^{-1}(\theta)e_k(\theta)\} + const \tag{3.3.5}$$

where $e_k(\theta)$, $\Sigma_{e_k}(\theta)$ is the prediction error and its covariance, and can be obtained from the Kalman filter equations [28]

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k e_k \tag{3.3.6}$$

$$\hat{x}_{k+1|k} = F\hat{x}_{k|k} \tag{3.3.7}$$

$$e_k = y_k - H\hat{x}_{k|k-1} \tag{3.3.8}$$

where we have suppressed the parameterization on θ, and

$$K_k = \Sigma_{k|k-1}H^T\Sigma_{e_k}^{-1} \tag{3.3.9}$$

$$\Sigma_{e_k} = H\Sigma_{k|k-1}H^T + R \tag{3.3.10}$$

$$\Sigma_{k|k} = \Sigma_{k|k-1} - K_k\Sigma_{e_k}K_k^T \tag{3.3.11}$$

$$\Sigma_{k+1|k} = F\Sigma_{k|k}F^T + Q \tag{3.3.12}$$

3.4 Application to Speech Synthesis

In this section we shall see how the theory of linear stochastic systems can be applied to improve the unit selection module of speech synthesizers.

Basically we are describing an acoustic phonetic model which will be used to provide better control in determining the optimal series of acoustic segments that need to be concatenated in order to synthesize the requested utterance. In essence, the proposed method will operate on the series of vector-valued target characteristics associated with each requested phonetic segment, predict a smoothed trajectory of those characteristics over the sequence of targets by making smooth, continuous motion in a hidden state-space, and project those estimates onto the observation space. In this way, the model can be used both for inferring the most likely observation trajectories, and computing the probability of the actual, noisy observations.

Our work is based on the concept of Hidden Dynamic Models [12], an approach to acoustic-phonetic modeling, developed by the "Dynamic Segmental Models of Speech Coarticulation" workgroup during the 1998 NSF Workshop at JHU [11], that was inspired by the work on linear dynamic models of speech [20].

3.4.1 The LSDM as a Speech Synthesizer

The Linear Segmental Dynamic Model (LSDM) describes the way in which an acoustic pattern is produced from a sequence of phones with given durations. An LSDM can be trained from real-word data, in order to learn the underlying state space, which is specific to the speech features of the voice talent used to record the speech database, creating a customized model of the system's complex behavior in the observation space.

An LSDM consists of two separate components which accommodate separate sources of speech variabilities in the relationship between phone sequences and spectra.

The first component is a smooth dynamic one, linear but nonstationary. The nonstationarity is described by a sequence of segments, each corresponding to a phonological unit, i.e. *phone*. For each phone class, there is a single target vector which defines a point in the hidden dynamic space. For each phone segment in the sequence, the respective target applies for the duration of that particular segment, resulting in the target sequence t_j, as is shown in Figure 3.1. This is typically multidimensional, but a single dimension is shown here for clarity.

This target sequence is smoothed to produce a trajectory in hidden dynamic space, x_j. The filter used for this smoothing is a second order symmetrical (forward-backward) low-pass filter, whose single time-constant parameter, p_j, is also determined by the phone class. In the general multidimensional case, there is a different time-constant for each dimension of the hidden dynamic space.

The hidden dynamic trajectory can be mapped to the surface acoustic form, y_j, by a non-linear mapping function, here an extended Kalman filter (EKF). The EKF is an application of the standard Kalman filter (for linear systems) on non-linear systems with additive white noise [43]. It will be used in order

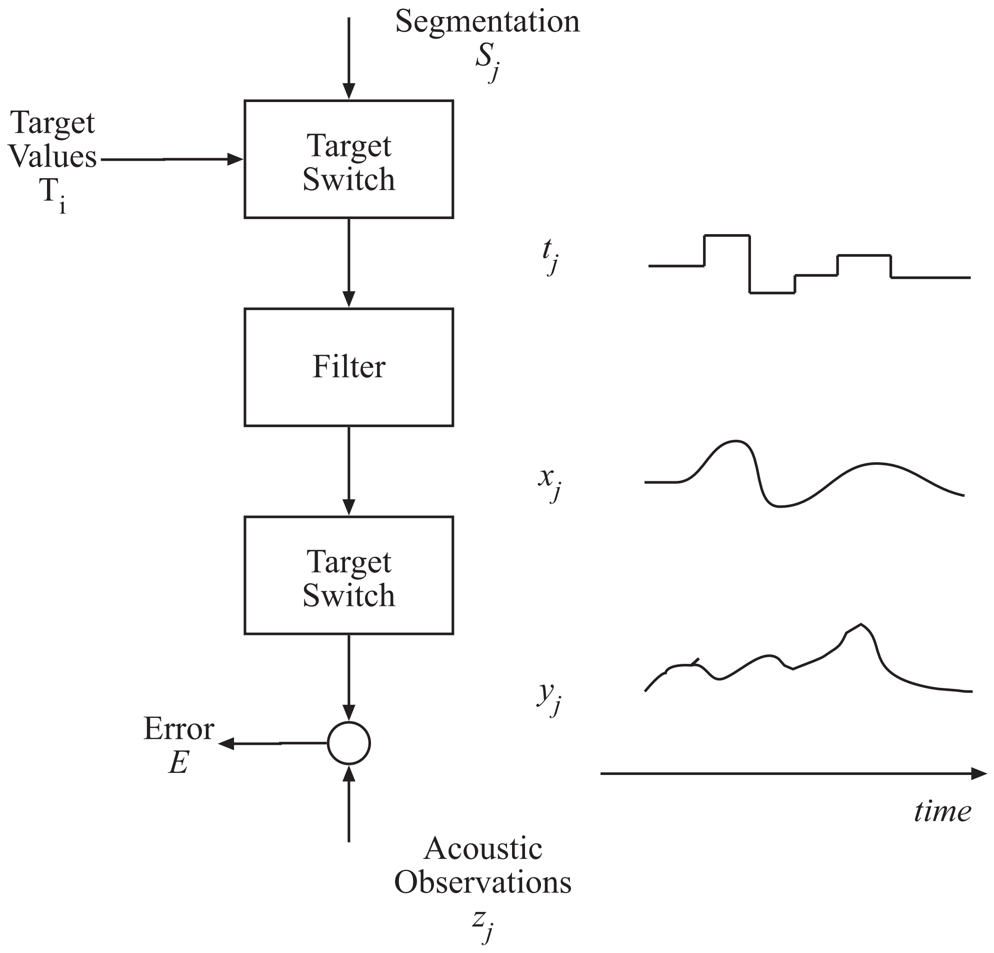

Figure 3.1: Calculating an Acoustic Error using the Linear Segmental Dynamic Model.

to compute the parameters of posterior probability distributions for the speech signal. A multi-layer perceptron (MLP) could also be used to perform this non-linear mapping function.

This mapping defines the hidden dynamic space, and a single EKF is used for all phones. This mapping can be considered analogous to the mapping between vocal tract shapes and speech sounds, although it will be learned only from the acoustic data, and will not be restricted to any predetermined from. The criterion is to model the structure of the acoustic speech pattern.

The two components combined form a nonstationary, non-linear dynamic system whose structure and properties are well understood in terms of the general process of human speech production.

As Richards and Bridle in [38] suggest, non-linearity is required for the model to work properly. This approach is superior to that utilizing linear mapping. According to [38], using a linear mapping scheme will be equivalent to using the simple transitions predicted by the dynamic system, but operating directly on the acoustic domain. As a result, the linear system will be unable to produce convincing formant transitions, like those expected from the non-linear system.

3.4.2 Applying the Model

The predictions of the LSDM can be used in order to compute the naturalness of the join between two phones. In this section, we are presenting a series of ideas on how this can be achieved.

Assuming that we are using phone specific models, with one set of parameters H, F, C, D, v, w and x_0 per phone, the LSDM can be used in order to track the observation trajectories.

There are two positions within a phone where a concatenation can occur: in the middle (which usually produces better results in the case of concatenation between vowels) and at the boundaries of a phone. We are proposing a series of metrics for evaluating the quality of the join. These metrics can serve as join sub-costs and will be used for defining the final cost function.

Mid-phone Concatenation

At the start of the phone, the model will follow these trajectories closely, since the speech is natural and closely resembles the data on which the model has been trained. As the model approaches the position of the join, where discontinuities will exist at a certain level, the model will treat these discontinuities as noise and predict a smooth path through the join. Finally, as we will be reaching the end of the phone segment, the model will follow the trajectories closely again, as this is natural speech.

The difference between the predicted and the actual trajectories at the point of concatenation, i.e. the error between the smooth path and the actual observation, can serve as the join cost.

The measure used for the computation of the join cost can be extracted from the log likelihood of the observation sequence Y, given the parameters of the model:

$$\log p(Y|m) = \sum_{k=0}^{N} \{\log|\Sigma_{e_k}| + e_k^T(\theta)\Sigma_{e_k}^{-1}(\theta)e_k(theta)\} + const \tag{3.4.1}$$

where e_k and Σ_{e_k} are the prediction error and its covariance for the model m, and they can be obtained from the standard Kalman filter recursions [20].

Suppose that a certain phone is synthesized by concatenating two half-phones, thus resulting to a concatenation point in the middle of the synthesized phone. In the case of a good join, the negative log likelihood estimate for this phone will

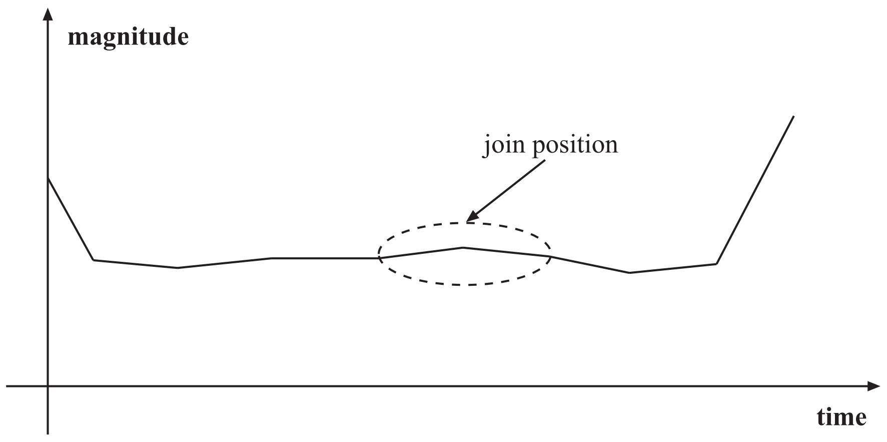

Figure 3.2: Negative log likelihood estimate for a good join

be almost constant, for the duration of the phone, just like in Figure 3.2.

However, in the case of a bad join, as the observed trajectories around the concatenation point will deviate from those predicted by the model, there will be an accumulation of error, as is graphically depicted in Figure 3.3.

Based on these, we can define a series of join cost measures. Some ideas include the following:

- An average of the negative log likelihood estimates over a number of frames centered on the join
- The relative increase (over an estimated baseline) in the negative log likelihood estimates, averaged over a number of frames centered on the frame with the highest estimate

A series of experiments which will involve the computation of these measures over various numbers of frames will help us determine the measure (or combination of measures) that will be used for determining the join cost.

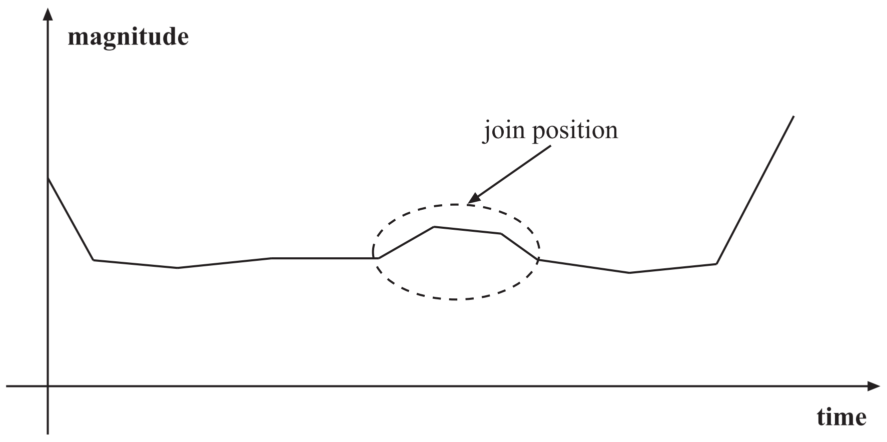

Figure 3.3: Negative log likelihood estimate for a good join

Phone-boundary Concatenation

Given the spectral differences between different types of phones, we can expect a considerable change in the Linear Spectral Frequency (LSF) trajectories across phone boundaries. However, coarticulatory effects tend to smooth the transition from one segment to the next. So, even if the LSF trajectories at the middle of the two phones are considerably different, a smooth transition from one phone segment to the next will be equivalent to a smooth change of the slope of these trajectories at the concatenation point.

The difference in the derivative of the trajectory from one segment to the next can also serve as a metric for determining the join cost.

3.4.3 Training the Model

In order to train the model we will use a series of automatically segmented and labeled acoustic data.

The LSDM will be used to synthesize an acoustic pattern y_i from a sequence of phone symbols and timings. These provide the target specifications for the utterance that needs to be synthesized. During the training phase, the following parameters of the model should be determined:

- Target Values (a vector in hidden parameter space for each phone class)
- Time-Constants (also a vector in hidden parameter space for each phone class)
- Non-Linear Mapping Parameters (MLP weights and/or EKF parameters)

The model parameters will be learned through the use of the Expectation-Maximization (EM) algorithm, which will produce maximum likelihood estimates for these parameters. During the E-step, statistics will be accumulated over the available training examples, using the previous set of model parameters. Then in the M-step, these statistics are used to update the model parameters [20].

Chapter 4

Research Plan

In the previous sections we discussed the open issues related to the Unit Selection subsystem of concatenative speech synthesizers. In this discussion we have attempted to present to the user the critical issues to be addressed for this method. These issues can be summarized as follows:

- the type and size of units to be stored in the speech database
- the design and recording of a speech corpus
- the cost functions that can be used in determining the optimal series of speech segments to be used in synthesizing an utterance
- the objective distance measure for evaluating qualitative differences between natural and synthesized utterances during the training of the cost function weights
- the optimizations in computational complexity for the unit selection component that can be incorporated into a speech synthesizer

4.1 Areas of Research

We present here the areas that will be investigated in the remainder of the research. The presentation is organized corresponding to the items in the list above.

4.1.1 Speech Units and the Unit Database

Although most contemporary concatenative synthesizers are using diphones or phones as their basic speech units, a wide variety of alternatives has been proposed in related bibliography. It seems though that the most promising alternative is the use of half phones, as proposed and implemented by the AT&T team in their Next-Gen TTS System [2].

Phones represent the actual units that make up speech. In the first successful unit–selection synthesizers, such as CHATR [4], phone-sized segments were used as synthesis units [27]. However, phone–sized segments fail to encapsulate the coarticulation effects that are present in speech, as these occur mostly at the boundaries between phones, thus requiring a large number of phone–sized units to provide sufficient coverage of those phenomena, on both boundaries.

On the other hand, a diphone is an artificial phonetic unit, which captures the transition information between two phones. A diphone segment contains the waveform starting in the middle of the first phone and ending in the middle of the second one. The reasons behind the popularity of diphone-based unit selection synthesizers are many. Firstly, diphones have the ability to preserve some of the coarticulation effects which are present at phoneme boundaries. Because of that, they minimize concatenation discontinuities, allowing for better quality speech. Finally, they are relatively few in number; for instance, with 40 or so phonemes in English, there can only be 1600 diphones, although, in practice, quite a few of them never occur due to the language's phonotactic constraints.

In an analysis of the advantages and disadvantages of the use of phone- and diphone-sized segments in unit selection synthesis, presented by Bulyko in [13], the author comes to the conclusion that neither of them is sufficient to provide consistently high quality synthesized speech. The problem lies in the fact that in some instances phone-boundary concatenation might be preferable to mid-phone concatenation, and vice versa. For instance, Sproat in [39] presents evidence that stops and fricatives have minimal coarticulation effects, and therefore are less likely to have perceived discontinuities at joins at the phone boundaries. Vowels, on the other hand, are found to have smoother concatenations in the middle of the phone. Furthermore, different vowels have different degrees of perceived discontinuity when spliced in the middle of the phone [30].

The evidence presented here motivate the following design decisions:

- Half-phones should be the acoustic units stored in the system's speech database, as they comprise the advantages associated with the use of phone- and diphone-sized speech segments for this function.

- We should use an efficient method for deciding in every particular instance whether a phone boundary of mid-phone concatenation is more desirable.

4.1.2 Speech Corpus Design

The speech corpus is the source of synthesis units that form the unit inventory. The corpus is constructed through the realization of a specially designed set of texts by a specially trained speaker, often referred to as the voice talent. Automatic segmentation methods, as those suggested by Ljolje and Riley [32] and van Santen and Sproat [42] are then utilized in order to create the unit inventory.

The construction of the speech corpus is thus made up of two tasks: i) the design of the texts that will be recorded, and ii) the recording of the speech data.

The design of the text corpus upon which the speech database is based is probably one of the most important tasks for the preparation of a speech synthesizer. During the text corpus design, the following goals should be met [8]:

- Provide full coverage for the required inventory units
- Provide sufficient coverage for domain-related contexts
- Include units in as many prosodic and coarticulatory environments as possible
- Keep the size of the unit inventory at a manageable level

Even if we might be using half–phones for synthesis, it will be impossible to get high quality synthetic speech if we made our design decisions for the text corpus (diphone coverage, coarticulatory contexts, etc.) with half–phones in mind. For the purposes of text–corpus design we should regard half–phones as nothing more than diphones that can also be split in half. Thus, in the case of phonetic coverage, we should be thinking in terms of diphones.

Several methods can be used for the selection of sentences that provide coverage for the requested diphones. Most of them are implemented using a greedy algorithm, or variations of it, using frequency–related weights. However, corpus selection is a complex problem, in which the full coverage of the unit space is indeed the most important, yet not the sole factor in deciding which of the candidate sentences to keep and which to discard. Thus, sentence selection should be treated as an optimization problem, in which all of the aforementioned goals should be pursued simultaneously.

The advantage of unit–selection synthesizers is that little or no signal processing is needed in order to synthesize natural-sounding speech. This is attributed to the fact that the unit inventory is made up of units extracted from real speech, thus encapsulating the prosodic characteristics of the utterances they were extracted from. In order to achieve natural-sounding results under most circumstances, we should include sentences extracted from a wide spectrum of text types, or genres. In addition to that, and in order to ensure proper coverage for the genre, even on the word and phrase level, we should also try to include the words which are characteristic to the vocabulary of each genre.

Regarding the recording of the data, there are also a few facts that need to be taken into account. Firstly, speech is highly variable, and even a highly–trained professional cannot produce speech with characteristics that do not alter with time. Practically, recording sessions should last at most two to three hours, after which the voice talent should be allowed to rest, and a new session be scheduled for the day after. Secondly, even during these three-hour long sessions, the number of successfully rendered and recorded utterances is limited, thus requiring a long series of sessions in order to record the whole corpus.

We can see that the design and recording of the speech corpus is a struggle between the extensive coverage, both phonetic and linguistic, of a language, and the constraints imposed by the size of the speech corpus. Our suggestions regarding this procedure are the following:

- Base our text corpus on real-life material. The material should cover as many genres as possible, allowing support both for genre-specific vocabulary, as well as for various prosodic styles

- Include all units in as many contexts as possible. Use context classes (phone classes) in order to provide coverage for as many contexts, without expanding the corpus to unmanageable sizes.
- Use an algorithm for selecting sentences from the candidate corpus based on the maximization of a series of criteria (phonetic coverage, genre coverage, vocabulary coverage) rather than a single one.
- Compile a text corpus of no more than 100k words. Even at this size, it will require ten days of 3-hour long studio sessions to record.
- Keep recording sessions short and allow the voice talent time to warm up, until his or her voice sounds similar to that of the previous session.

4.1.3 Costs and Cost Functions

Our decision to use half-phone sized synthesis units in our system introduces the requirement for a finer control over the decision on whether to perform phone boundary of mid-phone level concatenations in any particular instance.

Most concurrent systems use only two costs in determining the best sequence of segments that should be concatenated in order to synthesize the requested utterance, namely target and concatenation costs. Of them, only the concatenation cost can penalize the selection of a particular sequence of segments that will result in audible spectral discontinuities.

AT&T's NextGen TTS system [2], which also uses half-phones as its synthesis units, allows the scaling of concatenation costs at phone boundaries, relative to that at the mid-phone boundary. This feature has the effect of tuning the system's behavior, gradually shifting it between phone and mid-phone concatenation.

However, as mentioned above, there are certain instances where phone-boundary concatenation is preferable to mid-phone. The method used in AT&T's system, does not provide such a level of control over the unit selection process.

Bulyko in [13] suggested the introduction of an additional cost function that can be used as a measure of the potential discontinuity that a given unit my incur, if a splice is made at its boundary. This is called the *slicing cost*, it is defined for both left and right boundaries of a given unit boundaries and can be used in controlling the behavior of unit selection, separately from concatenation costs. Splicing costs are supposed to be inversely related to the spectral change at the boundaries, with [13] and [14] providing possible implementations for this measure.

In addition to the aforementioned work, Klabbers and Veldhuis in [29] and [30] also worked on a measure for assessing potential discontinuities for diphone-based synthesis. A series of experiments with a series of potential measure functions resulted in the adoption of the Kullback-Leibler distance [31] between two power normalized spectral envelopes as a metric for the severity of audible discontinuities that might occur from the joining of two diphones.

In a work by Richards et. al, first described in [11] and formalized in [38], Hidden Dynamic Models can be used in order to model coarticulation in speech. Although originally developed as an enhancement for the acoustic models used in speech recognizers, it could be applied to speech synthesis. We could use the HDM model in order to predict the spectral characteristics of the other side of the boundary, using the phonetic context and the spectrum on one side. If the prediction is accurate then this would mean that the two segments are good candidates for concatenation, whereas in the case of splicing costs, high predictability would be evident of strong coarticulatory effects, thus making this boundary a poor choice for making a splice. Donovan in [22] suggests modeling such costs as the probability of the prediction residual using a simple predictor such as $\hat{X}_{t+1} = X_t + \mu_q$, where q represents the context. We will investigate whether a more general linear predictor of the form $\hat{X}_{t+1} = B_q Z_t(q) + \mu_q$ might be more useful, so that X_t as well as elements from other time vectors, can be included in $Z_t(q)$, thus being able to consider a longer context in our decision.

Finally, we will also implement the ideas and methods presented in Chapter 3 with regard to the definition of join costs, based on the LSDM model.

Therefore, in order to achieve better control over the unit selection process, minimizing spectral discontinuities, we suggest the following:

- The adoption of a new cost function, namely *splicing cost* that, in addition to the join cost function, will offer us better control over both the selection of synthesis units, and the type of splicings (i.e. phone-boundary or mid-phone splicings)

- The evaluation of a series of metrics for measuring potential audible spectral discontinuities resulting from the join of two segments; these will include the Kullback-Leibler distance, Mahalanobis distance between successive vectors of spectral features, and an HDM-based prediction model, as described above.

- The implementation of the measures suggested in the LSDM approach to join costs, as presented in Chapter 3

4.1.4 Objective Evaluation for Cost Function Training

Each of the cost functions that we referred to in Chapter 2, combines a series of sub-costs in order to determine the overall cost for any given candidate unit. Each of these sub-costs, representing the various feature distances in both target and concatenation costs, participates in the cost function multiplied by a weighting factor, namely the respective *sub-cost weight*. As Black and Campbel acknowledge in [7], the quality of the result of unit selection depends heavily on the weights for the various sub-costs.

The initial approach to the tuning of these weights was manual. However, automatic training methods consistently resulted to better sounding speech [7] [27]. The method used for training the weights of the cost functions is still an open issue. The most common method is to use the system in order to synthesize utterances that were already recorded by the voice talent, yet they were kept out of the speech corpus. Then, through an iterative process, the weights are modified so that the synthesized utterances will match the pre-recorded ones.

This process requires an objective measure of the distance between the synthesized utterance and the pre-recorded one. Subjective listening tests with humans are prone to errors and quite impractical in this case, due to the large number of utterances that need to be evaluated. Initial approaches used the mean Euclidean cepstral distance [37] between time–aligned vectors from the selected segments and the target units, which are completely known, since we are trying to mimic natural speech utterances from the speaker of the source database.

However, as Ng points out in [36], comparisons of human perceptual measures and the cepstral distance measure show that people are more sensitive to continuity distortions, while cepstral distance measure gives more importance to unit distortions.

Clearly, a better distance measure, that is more indicative of perceived speech quality is needed. It is possible that a combination of measures, with the addition of a metric for spectral discontinuities will lead to better results. However, the actual choice of this metric and the methodology that will be used for the training remains to be investigated at a later stage of this research.

4.1.5 Computational Optimization

A successful speech synthesizer needs to be able to produce high quality speech in as little time and by requiring as little computational resources as possible. With unit selection being the most computationally demanding stage of the synthesis process, it is obvious that any optimization applied to this stage will have an immediate effect on the system's performance.

In Chapter 2 we referred to a series of techniques used in order to optimize the unit selection process. Unit pre-selection, cost caching and pruning can all be used in order to cut down on the resources needed by the speech synthesizer. However, it is obvious that the major improvement will come from the reduction of the number of candidate units that need to be examined for the synthesis of any given utterance.

A first optimization comes if we consider only the units with phonetic contexts similar to the target segments. With unit inventories containing tens of thousands of each unit class, even this step can considerably narrow the search space. Using these units we can start the unit selection process, but we can proceed in stages, pruning the search space initially using target and finally concatenation costs. The introduction of the proposed splicing cost, will add yet another pruning stage between target-cost and concatenation-cost pruning, minimizing the final number of units that will need to be considered for selection. The pre-computation of the costs between the most frequent combinations of units and the creation of caches holding this information will further reduce the requirements for complex computations at synthesis-time.

Bibliography

[1] J. Allen, S. Hunnicutt, and D. Klatt, *The MITalk System*, Cambridge University Press, Cambridge, 1987.

[2] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal, *The AT&T Next-Gen TTS system*, Joint Meeting of ASA, EAA, and DAGA (Berlin, Germany), March 1999.

[3] M. Beutnagel, M. Mohri, and M. Riley, *Rapid Unit Selection from a Large Speech Corpus for Concatenative Speech Synthesis*, Eurospeech '99 (Budapest, Hungary), 1999.

[4] A. Black, *Chatr version 0.8: A generic Speech Synthesizer*, System documentation, ATR Interpreting Telecomunications Laboratories, Kyoto, Japan, March 1996.

[5] A. Black, P. Taylor, and R. Caley, *The Festival speech synthesis system*, Technical report hcrc/tr-83, Human Computer Communication Research Centre, Edinburgh University, 1997.

[6] A. W. Black, *Perfect Synthesis for All of the People All of the Time*, IEEE TTS Workshop 2002, 2002.

[7] A. W. Black and N. Campbell, *Optimizing Selection of Units from Speech Databases for Concatenative Synthesis*, Eurospeech '95 (Madrid, Spain), vol. 1, 1995, pp. 581–584.

[8] A. W. Black and K. A. Lenzo, *Optimal Data Selection for Unit Selection Synthesis*, Proceedings 4th ESCA Workshop on Speech Synthesis (Edinburgh, Scotland), 2001.

[9] A. W. Black and P. Taylor, *CHATR: a generic speech synthesis system*, COLING94 (Kyoto, Japan), vol. 2, 1994, pp. 983–986.

[10] ———, *Automatically clustering similar units for unit selection in speech synthesis*, Proceedings Eurospeech '97 (Rhodes, Greece), 1997, pp. 601–604.

[11] J.S. Bridle, L. Deng, J. Picone, H. B. Richards, J. Ma, T. Kamm, M. Schuster, S. Pike, and R. Regan, *An Investigation of Segmental Hidden Dynamic Models of Speech Coarticulation for Automatic Speech Recognition*, Technical report of a projects at the jhu workshop on language engineering for students and professionals, Center for Language and Speech Processing, The Johns Hopkins University, Phoenix, AZ, 1998.

[12] J.S. Bridle and H. B. Richards, *Dynamic Segmental Models of Speech Coarticulation*, JHU Workshop on Language Engineering for Students and Professionals (Phoenix, AZ), 1999.

[13] I. Bulyko and M. Ostendorf, *Unit Selection for speech synthesis using Splicing Costs with Weighted Finite State Transducers*, Proceedings of Eurospeech'01, 2001.

[14] I. Bulyko, M. Ostendorf, and J. Bilmes, *Robust splicing costs and efficient search with bmm models for concatenative speech synthesis*, Proceedings of ICASSP'02, vol. 1, 2002, pp. 461–464.

[15] N. Campbell and A. W. Black, *Progress in Speech Synthesis*, ch. Prosody and the Selection of Source Units for Concatenative Speech Synthesis, Springer Verlag, New York, 1995.

[16] F. Charpentier and E. Moulines, *Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones*, Proc. Eurospeech '89 (Paris, France), vol. 2, September 1989, pp. 13–19.

[17] J. Clark and C. Yallop, *An Introduction to Phonetics and Phonology*, Blackwell, Oxford and Cambridge, 1990.

[18] A. Conkie, *A Robust Unit Selection System for Speech Synthesis*, 137th meeting of ASA/Forum Acusticum (Berlin, Germany), March 1999.

[19] A. Conkie, M. C. Beutnagel, A. K. Syrdal, and P. E. Brown, *Preselection opf Candidate Units in a Unit Selection-Based Text-To-Speech Synthesis System*, Proceedings ICSLP'00 (Beijing, China), vol. 3, October 2000, pp. 314–317.

[20] V. Digalakis, J. Rohlicek, and M. Ostendorf, *Ml estimation of a stochastic linear system with the em algorithm and its applications to speech recognition*, IEEE Transactions on Speech and Audio Processing **1** (1993), no. 4, 431–442.

[21] R. Donovan, *Segment preselection in decision-tree based speech synthesis systems*, Proceedings of ICASSP, vol. 2, 2000, pp. 937–940.

[22] ______, *A New Distance Measure for Costing Spectral Discontinuities in Concatenative Speech Synthesisers*, Proceedings 4th ESCA Workshop in Speech Synthesis (Atholl Palace Hotel, Scotland), 2001.

[23] T. Dutoit and H. Leich, *MBR-PSOLA: Text-to-Speech synthesis based on an MBE re-synthesis of the segments database*, Speech Communication **13** (1993), no. 3-4, 435–440.

[24] ______, *The MBROLA Project: Towards a Set of High-Quality Speech Synthesizers Free of Use for NonCommercial Purpose*, ICSLP '96 (Philadelphia, PA), vol. 3, October 1996, pp. 1993–1996.

[25] W.I. Hallahan, *Dectalk Software: Text-to-Speech Technology and Implementation*, DIGITAL Technical Journal **7** (1996), no. 4, 39–51.

[26] J.N. Holmes, R.D. Wright, J.W. Yates, and M.W. Judd, *Extension to the JSRU Synthesis by Rule System*, 9th Internaltional Congress of Acoustics (Madrid, Spain), 1977.

[27] A. J. Hunt and A. W. Black, *Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database*, ICASSP '96 (Atlanta, GA), vol. 1, May 1996, pp. 373–376.

[28] R. E. Kalman, *A New Approach to Linear Filtering and Prediction Problems*, Trans. ASME, Series D, J. Basic Eng. **82** (1960), 35–45.

[29] E. Klabbers and R. Veldhuis, *On the Reduction of Concatenation Artifacts in Diphone Synthesis*, Proceedings ICSLP'98 (Sydney, Australia), 1998, pp. 1983–1986.

[30] ———, *Reducing audible spectral discontinuities*, IEEE Transactions on Speech and Audio Processing **9** (2001), no. 1, 39–51.

[31] S. Kullback and R. Leibler, *On Information and Sufficiency*, Annals of Mathematical Statistics **22** (1951), 79–86.

[32] A. Ljolje and M. Riley, *Automatic Segmentation of Speech for TTS*, Proceedings 3rd ESCA (Berlin, Germany), vol. 2, 1993, pp. 1445–1448.

[33] M. W. Macon, A. E. Cronk, and J. Wouters, *Generalization and Discrimination in Tree-Structured Unit Selection*, Proceedings 3rd Synthesis Workshop, November 1998.

[34] J. D. Markel and A. Gray, *Linear Prediction of Speech*, Springer Verlag, Berlin, Germany, 1976.

[35] J. S. Marques and A. J. Abrantes, *Hybrid Harmonic Coding of Speech at low Bitrates*, Speech Communication **14** (1994), no. 3, 231–247.

[36] K. Ng, *Survey of Data-Driven Approached to Speech Synthesis*, Area survey, Spoken Language Systems Group, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[37] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[38] H. B. Richards and J.S. Bridle, *The HDM: A Segmental Hidden Dynamic Model of Coarticulation*, ICASSP (Phoenix, AZ), May 1999.

[39] R. Sproat, *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[40] Y. Stylianou, *Harmonic plus Noise Models for Speech, combined with Statistical Methods, for Speech and Speaker Modification*, Ph.D. thesis, Ecole Nationale Superieure des Telecommunications, Paris, France, January 1996.

[41] Y. Stylianou and A. K. Syrdal, *Perceptual and Objective Detection of Discontinuities in Concatenative Speech Synthesis*, ICASSP 2001, 2001.

[42] J. van Santen and R. Sproat, *High-accuracy Automatic Segmentation*, Proceedings of EuroSpeech99 (Budapest, Hungary), 1999.

[43] G. Welch and G. Bishop, *An introduction to the Kalman Filter*, Technical report, Computer Science Department, University of North Carolina, Chapel Hill, NC, April 2004.

[44] J. Wouters and M. Macon, *A Perceptual Evaluation of Distance Measures for Concatenative Speech Synthesis*, ICSLP 98 (Sydney, Australia), November 1998, pp. 2747–2750.

[45] J. R. W. Yi and J. R. Glass, *Natural Sounding Speech Synthesis using Variable-Length Units*, ICSLP 98 (Sydney, Australia), November 1998.

[46] R. W. Yi, *Corpus-Based Unit Selection for Ntural Sounding Speech Synthesis*, Ph.D. thesis, Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, Cambridge, MA, 2003.