Sproat, R. & Olive, J. “Text-to-Speech Synthesis”
Digital Signal Processing Handbook
Ed. Vijay K. Madisetti and Douglas B. Williams
Boca Raton: CRC Press LLC, 1999

46

Text-to-Speech Synthesis

Richard Sproat
Bell Laboratories
Lucent Technologies

Joseph Olive
Bell Laboratories
Lucent Technologies

46.1 Introduction

Text-to-speech synthesis has had a long history, one that can be traced back at least to Dudley's "Voder", developed at Bell Laboratories and demonstrated at the 1939 World's Fair [1]. Practical systems for automatically generating speech parameters from a linguistic representation (such as a phoneme string) were not available until the 1960s, and systems for converting from ordinary text into speech were first completed in the 1970s, with MITalk being the best-known such system [2]. Many projects in text-to-speech conversion have been initiated in the intervening years, and papers on many of these systems have been published.[1]

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 text-to-speech (TTS) synthesizer would start with the words in the text, convert each word one-by-one into speech (being careful to pronounce each word correctly), and concatenate the 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 view 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 deemphasize others; they must "chunk" the sentence into meaningful (intonational) phrases; they must pick an appropriate F0 (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 because *segmental durations* are affected by various factors, including phrasal position.

[1] For example, [3] gives an overview of recent Dutch efforts in this area. Audio examples of several current projects on TTS can be found at the WWW URL http://www.cs.bham.ac.uk/~jpi/synth/museum.html.

What makes reading such a difficult task 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) completely specifies the words that are present, it will only partly specify the intonational phrases (typically with some form of punctuation), will usually not indicate which words to accent or deaccent, and hardly ever give 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.[2] Humans are able to perform these tasks because, in addition to being knowledgeable about the grammar of their language, they also (usually) understand the content of the text that they are reading, and can thus appropriately manipulate various extragrammatical "affective" factors, such as appropriate use of intonation and voice quality.

The task of a TTS system is thus a complex one that 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 making use, where possible, of purely grammatical information to decide on such things as accentuation, phrasing, and intonation — and 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 subproblems. The first of these is the conversion of text — an imperfect representation of language, as we have seen — into some form of linguistic representation that includes information on the phonemes (sounds) to be produced, their duration, the locations of any pauses, and the F0 contour to be used. The second — the actual synthesis of speech — takes this information and converts it into a speech waveform. Each of these main tasks naturally breaks down into further subtasks, some of which have been alluded to. The first part, text and linguistic analysis, may be broken down as follows:

- *Text preprocessing*: including end-of-sentence detection, "text normalization" (expansion of numerals and abbreviations), and limited grammatical analysis, such as grammatical part-of-speech assignment.
- *Accent assignment*: the assignment of levels of prominence to various words in the sentence.
- *Word pronunciation*: including the pronunciation of names and the disambiguation of homographs.[3]
- *Intonational phrasing*: the breaking of (usually long) stretches of text into one or more intonational units.
- *Segmental durations*: the determination, on the basis of linguistic information computed thus far, of appropriate durations for phonemes in the input.
- *F0 contour computation*.

---

[2] Even in English, single *orthographic* words, e.g., *AT&T*, can actually represent multiple words — *A T and T*.

[3] A *homograph* is a single written word that represents two or more different lexical entries, often having different pronunciations: an example would be *bass*, which could be the word for a musical range — with pronunciation /be^{j}s/ — or a fish — with pronunciation /bæs/. We transcribe pronunciations using the International Phonetic Association's (IPA) symbol set. Symbols used in this chapter are defined in Table 46.1.

Speech synthesis breaks down into two parts:

- The selection and concatenation of appropriate concatenative units given the phoneme string.
- The synthesis of a speech waveform given the units, plus a model of the glottal source.

46.2 Text Analysis and Linguistic Analysis

46.2.1 Text Preprocessing

The input to TTS systems is text encoded using an electronic coding scheme appropriate for the language, such as ASCII, JIS (Japanese), or Big-5 (Chinese). One of the first tasks facing a TTS system is that of dividing the input into reasonable chunks, the most obvious chunk being the sentence. In some writing systems there is a designated symbol used for marking the end of a declarative sentence and for nothing else — in Chinese, for example, a small circle is used — and in such languages end-of-sentence detection is generally not a problem. For English and other languages we are not so fortunate because a period, in addition to its use as a sentence delimiter, is also used, for example, to mark abbreviations: if one sees the period in *Mr.*, one would not (normally) want to analyze this as an end-of-sentence marker. Thus, before one concludes that a period does in fact mark the end of a sentence, one needs to eliminate some other possible analyses. In a typical TTS system, text analysis would include an abbreviation-expansion module; this module is invoked to check for common abbreviations which might allow one to eliminate one or more possible periods from further consideration. For example, if a preprocessor for English encounters the string *Mr.* in an appropriate context (e.g., followed by a capitalized word), it will expand it as *mister* and remove the period.

Of course, abbreviation expansion itself is not trivial, since many abbreviations are ambiguous. For example, is *St.* to be expanded as *Street* or *Saint*? Is *Dr.*, *Doctor* or *Drive*? Such cases can be disambiguated via a series of heuristics. For *St.*, for example, the system might first check to see if the abbreviation is followed by a capitalized word (i.e., a potential name), in which case it would be expanded as *Saint*; otherwise, if it is preceded by a capitalized word, a number, or an alphanumeric (*49th*), it would be expanded as *Street*. Another problem that must be dealt with is the conversion of numbers into words: *232* should usually be expanded as *two hundred thirty two*, whereas if the same sequence occurs as part of *232-3142* — a likely telephone number — it would normally be read *two three two*.

In languages like English, tokenization into words can to a large extent be done on the basis of white space. In contrast, in many Asian languages, including Chinese, the situation is not so simple because spaces are never used to delimit words. For the purposes of text analysis it is therefore generally necessary to "reconstruct" word boundary information. A minimal requirement for word segmentation is an on-line dictionary that enumerates the wordforms of the language. This is not enough on its own, however, since there are many words that will not be found in the dictionary; among these are personal names, foreign names in transliteration, and morphological derivatives of words that do not occur in the dictionary. It is therefore necessary to build models of these non-dictionary words; see [4] for further discussion.

In addition to lexical analysis, the text-analysis portion of a TTS system will typically perform syntactic analysis of various kinds. One commonly performed analysis is grammatical part-of-speech assignment, as information on the part of speech of words can be useful for accentuation and phrasing, among other things. Thus, in a sentence like *they can can cans*, it is useful for accentuation purposes to know that the first *can* is a *function word* — an auxiliary verb, whereas the second and third are *content words* — respectively a verb and a noun. There are a number of part-of-speech algorithms available, perhaps the best known being the stochastic method of [5], which computes the

most likely analysis of a sequence of words, maximizing the product of the *lexical probabilities* of the parts-of-speech in the sentence (i.e., the possible parts of speech of each word and their probabilities), and the *n-gram probabilities* (probabilities of n-grams of parts of speech), which provide a model of the context.

46.2.2 Accentuation

In languages like English, various words in a sentence are associated with *accents*, which are usually manifested as upward or downward movements of fundamental frequency. Usually, not every word in the sentence bears an accent, however, and the decision on which words should be accented and which should be unaccented is one of the problems that must be addressed as part of text analysis. It is common in prosodic analysis to distinguish three levels of *prominence*. Two are accented and unaccented, as just described, and the third is *cliticized*. Cliticized words are unaccented but in addition have lost their word stress, so that they tend to be durationally short: in effect, they behave like unstressed affixes, even though they are written as separate words.

A good first step in assigning accents is to make the accentual determination on the basis of broad lexical categories or parts of speech. Content words — nouns, verbs, adjectives, and perhaps adverbs, tend in general to be accented; function words, including auxiliary verbs and prepositions tend to be deaccented; short function words tend to be cliticized. But accenting has a wider function than merely communicating lexical category distinctions between words. In English, one important set of constructions where accenting is more complicated than what might be inferred from the above discussion are complex noun phrases — basically, a noun preceded by one or more adjectival or nominal modifiers. In a "discourse-neutral" context, some constructions are accented on the final word (*Madison* **Avenue**), some on the penultimate (**Wall** *Street*, *kitchen* **towel** *rack*), and some on an even earlier word (**sump** *pump factory*). The assignment of accent to complex noun phrases depends on complex lexical and semantic factors; see [6].

Accenting is not only sensitive to syntactic structure and semantics, but also to properties of the discourse. One straightforward effect is *contrast*, as in the example *I didn't ask for* **cherry** *pie, I asked for* **apple** *pie*. For most speakers, the "discourse neutral" accent would be on *pie*, but in this example there is a clear intention to contrast the ingredients in the pies, and *pie* is thus deaccented to effect the contrast between *cherry* and *apple*. See [7] for a discussion of how these kind of effects are handled in a TTS system for English. Note, while humanlike accenting capabilities are possible in many cases, there are still some intractable problems. For example, just as one would often deaccent a word that had been previously mentioned, so would one often deaccent a word if a supercategory of that word had been mentioned: *My son wants a Labrador, but I'm* **allergic** *to dogs*. Handling such cases in any general way is beyond the capabilities of current TTS systems.

46.2.3 Word Pronunciation

The next stage of analysis involves computing pronunciations for the words in the input, given the orthographic representation of those words. The simplest approach is to have a set of "letter-to-sound" rules that simply map sequences of graphemes into sequences of phonemes, along with possible diacritic information, such as stress placement. This approach is naturally best suited to languages where there is a relatively simple relation between orthography and phonology: languages such as Spanish or Finnish fall into this category. However, languages like English manifestly do not, so it has generally been recognized that a highly accurate word pronunciation module must contain

a pronouncing dictionary that, at the very least, records words whose pronunciation could not be predicted on the basis of general rules. However, having a dictionary that is merely a list of words presents us with familiar problems of coverage: many text words occur that are not to be found in the dictionary, including morphological derivatives from known words, or previously unseen personal names.

For morphological derivatives, standard techniques for morphological analysis [2, 8] can be applied to achieve a morphological decomposition for a word. The pronunciation of the whole can then, in general, be computed from the (presumably known) pronunciation of the morphological parts, applying appropriate phonological rules of the language. For novel personal names, additional mechanisms may be necessary since novel names cannot always be related *morphologically* to previously seen ones. One such additional method involves computing the pronunciation of a new name by analogy with the pronunciation of a similar name [9, 10]. For example, imagine that we have the name *Califano* in our dictionary and that we know its pronunciation: then we could compute the pronunciation of a hypothetical name *Balifano* by noting that both names share the "suffix" *alifano*. The pronunciation of *Balifano* can then be computed by removing the phoneme /k/, corresponding to the letter *C* in *Califano*, and replacing it with the phoneme /b/.

There are some word forms that are inherently ambiguous in pronunciation, and for which a word pronunciation module as just described can only return a set of possible pronunciations, from which one must then be chosen. A straightforward example is the word *Chevy*, which is most commonly pronounced /$\int' \varepsilon vi$/, but is /t$\int' \varepsilon vi$/ in the name *Chevy Chase*, so in this case one could succeed by simply storing the bigram *Chevy Chase*. But n-gram models do not solve all cases of homograph

TABLE 46.1 IPA Symbols Used in this Chapter

IPA Symbol	Phonetic value
æ	**a** as in *c**a**t*
b	**b** as in **b***ass*
e^j	**a** as in *b**a**ke*
ɛ	**e** as in *b**e**t*
ə	**a** as in *d**a**ta*
i	**i** as in *l**i**nguini*
ɪ	**i** as in **i***n*
s	**s** as in **s***oggy*
ʃ	**sh** as in **sh***ip*
t	**t** as in **t***est*
tʃ	**ch** as in **ch***ase*
ð	**th** as in **th***e*
r	**r** as in *a**r**e*
v	**v** as in **v***oodoo*
w	**w** as in **w***e*
'	primary stress

disambiguation. So, the word *bass*, is most likely to be pronounced /bæs/ in a "fishy" context like *he was fishing for bass*, but /be^{j}s/ in a musical context like *he plays bass*. What defines the context as being musical or "fishy" is not characterizable in terms of n-grams, but rather relates to the occurrence of certain words (e.g., *fish*, *lake*, *boat* vs. *play*, *sing*, *orchestra*) in a wider context. A method proposed by

Yarowsky [11, 12] allows for both local (n-gram) context and wide context to be used in homograph disambiguation, and excellent results have been achieved using this approach.[4]

46.2.4 Intonational Phrasing

In reading a long sentence, speakers will typically break the sentence up into several phrases, each of which can be said to "stand alone" as an intonational unit. If punctuation is used liberally so that there are relatively few words between the commas, semicolons, or periods, then a reasonable guess at an appropriate phrasing would be simply to break the sentence at the punctuation marks (though this is not always appropriate [13]). The real problem comes when long stretches occur without punctuation; in such cases, human readers would normally break the string of words into phrases, and the problem then arises of where to place these breaks.

The simplest approach is to have a list of words, typically function words, that are likely indicators of good places to break [1]. One has to use some caution, however, because while a particular function word such as *and* may coincide with a plausible phrase break in some cases (*He got out of the car and walked towards the house*), in other examples it might coincide with a particularly *poor* place to break as in *I was forced to sit through a dog and pony show that lasted most of Wednesday afternoon*. Other approaches to intonational phrasing have been proposed in the literature, including methods that depend on syntactic parsers of various degrees of sophistication [13, 14]. An alternative approach, described in [15], uses a decision tree model [16, 17] that is trained on a corpus of text annotated with prosodic phrase-boundary information.

46.2.5 Segmental Durations

Having computed which phonemes are to be produced by the synthesizer, it is necessary to decide how long to make each one. In this section we briefly describe the methods used for computing segmental durations: the reader is referred to [18] for an extended discussion of this topic.

What duration to assign to a phonemic segment depends on many factors, including:

- The identity of the segment in question. For example, in many dialects of English, the vowel /æ/ has a longer *intrinsic duration* than the vowel /ɪ/.
- The stress of the syllable of which the segment is a member. For example, vowels in stressed syllables tend to be longer than vowels in unstressed syllables.
- Whether the syllable of which the segment is a member bears an accent. Accented syllables tend to be longer than otherwise identical unaccented syllables.
- The quality of the surrounding segments. For example, a vowel preceding a voiced consonant in the same syllable tends to be longer than the same vowel preceding a voiceless consonant.
- The position of the segment in the phrase: elements close to the ends of phrases tend to be longer than elements more internal to the phrase.

Various approaches have been taken to modeling segmental durations in TTS systems. One method involves *duration rules*, which are rules of the form "if the segment is *X* and it is in phrase-final position, then lengthen *X* by *n* milliseconds" [19, 20]. In rule-based systems of this kind, it is not unusual for the duration of a given segment to be rewritten several times as the conditions for the application of

[4] Clearly the above-described method for homograph disambiguation can also be applied to other formally similar problems in TTS, such as whether *St.* to be expanded as *Saint* or *Street*, or *747* is to be read as a number *seven hundred and forty seven* or the name of an aircraft *seven forty seven*.

the various rules are considered. The rule-based approach can be formalized explicitly in terms of the second approach — *duration models* — which are mathematical expressions that prescribe how the various conditioning factors are to be used in computing the duration of a segment [19]; the successive application of the rules can, in effect, be "compiled" into a single mathematical expression that implements the combined effect of the rules. As argued in [18], all extant duration models can be viewed as instances of a more general *sum-of-products* model, where the duration of a segment is predicted by a formula of the general form:

$$DUR(\mathbf{f}) = \sum_{i \in T} \prod_{j \in I_i} S_{i,j}(f_j) \tag{46.1}$$

Here the duration assigned to a feature vector — $DUR(\mathbf{f})$ — is computed by scaling each factor f_j in the ith product term by a factor scale $S_{i,j}$; computing the product of all scaled factors within each product term; and then summing over all i product terms. Rather than deciding *a priori* on a particular sums-of-products model (or set of such models) within the space of all possible models, one approach taken to segmental duration is to use exploratory data analysis to arrive at models whose predictions show a good fit to durations from a corpus of labeled speech [18].

More specifically, we start with a text corpus that (ideally) has a good coverage both of various phonemes and of the factors (and their combinations) that are deemed likely to be relevant for duration. A native speaker of the language reads this text and the speech is segmented and labeled. Using the text-analysis modules of TTS, with some possible hand correction, we automatically compute the sequence of phonemes, and the feature vectors (including features on stress, accent, phrasal position, etc.) associated with each phoneme. Given the feature vectors, various sums-of-products models are compared and their predictions of the values of the observed segmental durations are evaluated. In general, different specific duration models may be better suited to different sets of conditions than others: for example, in the English duration system, intervocalic consonants are associated with a different sums-of-products model than consonants that occur in clusters. In the actual implementation of segmental duration predictions, a decision tree is used to determine, on the basis of contextual factors appropriate to the segment at hand, what particular sums-of-products model to use; this model is then used to compute the duration of the segment.

Designing a corpus with good coverage of relevant factors is a non-trivial task in itself: the basic problem is to provide a set that has maximal coverage with the minimal amount of text to be read by a speaker, and analyzed. The method that we use involves starting with a large corpus of text in a language and automatically predicting the phonemic segments along with their features (again, using text analysis components for the language). A *greedy* algorithm is then applied to arrive at a minimal set of sentences that have good (ideally total) coverage of the desired feature vectors.

46.2.6 Intonation

Having computed linguistic information such as the sequence of segments to be produced, their duration, the prominence of the various words, and the locations of prosodic boundaries, the next thing that a TTS system needs to compute is an intonation contour. There are almost as many models of intonation implemented in TTS systems as there are TTS systems, and we do not have the space to review these different approaches here. Suffice it to say that most intonation models that have actually been incorporated into working TTS systems can be classified into one of three "schools":

- The *Fujisaki* school [21, 22, inter alia]. An intonation contour for a phrase is computed from a phrase impulse and some number of accent impulses. These impulses are convolved with a smoothing function to produce phrase and accent curves, which are then summed to produce the final contour.

- The *Dutch* school [23]. Intonation contours are represented as sequences of connected line segments which are chosen so as to perceptually closely approximate real (smooth) intonation contours.
- The *autosegmental/metrical* school [24, 25, inter alia]. Intonation contours are represented abstractly as sequences of high and low targets.

The computation of an intonation contour from a phonological representation can be illustrated by considering the Bell Labs English TTS system, which currently uses a version of the Pierrehumbert autosegmental model [26, 27, 28]. As the first stage in the computation of an intonation contour, a tone-timing function sets up nominal times for each accent in the sentence. Separate routines are called for initial boundary tones, final boundary tones, pitch accents and phrase accents. Roughly, initial boundary tones are aligned with the silence that is placed at the beginning of each minor phrase, whereas final boundary tones are aligned with the final vowel of each minor phrase. Phrase accents are aligned after the final word accent of the minor phrase, if there is one; otherwise at the end of the first vowel of the first word, or else at the end of the first phoneme. Finally, accents on words are aligned with their associated syllables using a complex set of contextual factors. These nominal accent times are then converted into actual F0/time pairs, by another function. F0 values are computed dependent on the prominence of the accent (either determined automatically, or else definable by the user), and various phrasal parameters from the intonation model, as well as the particular type of accent involved. Finally, an *F0 contour* is produced by interpolating the computed pitch/time pairs, and smoothing via convolution with a rectangular window.

46.3 Speech Synthesis

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.

There are two independent variables in the choice of parametric computation in a TTS system. One variable is the choice between a *rule-based* scheme for the computation of the parameters on the one hand, and a *concatenative scheme* involving concatenation of short segments of previously uttered speech on the other. The second variable is the actual parametric representation chosen: possible choices include articulatory parameters, formants, LPC (linear predictive coding), spectral parameters, or time domain parameters. In a concatenative scheme, any parametric representation that permits independent control of loudness, F0, voicing, timing, and possibly spectral manipulations is appropriate. Rule-based systems are more restrictive of the choice of parameters 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 articulatory parameters and formants have been used in rule-based systems. The best-known examples of a formant-based synthesizer are the Klatt synthesizer and its commercial offshoot DECtalk.

Rule-based systems are space-efficient because they eliminate the need to store speech segments. Rule-based approaches 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. However, since the dynamics of the parameters are very difficult to model it requires a great deal more effort to produce a rule-based system than it does to produce a concatenative system of comparable quality. Given the right choice of units, a concatenative scheme is able to store the dynamics of the speech signal and thus produce high quality synthetic sound. The choice of the exact parameters depends on what the designer values in such a system. Waveform representations — such as PSOLA [29] — have a high sound quality, but they are limiting in terms of the ability to alter the sound, and thus

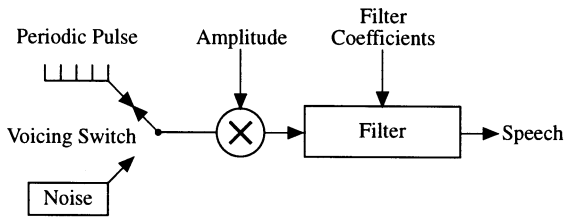

FIGURE 46.1: Source-filter model for speech synthesis.

far, no one has been able to change the spectral parameters in a time domain system. Articulatory parameters or formants, on the other hand, can be successfully manipulated. However, the speech quality produced by using these parameters is somewhat degraded because there are no reliable methods to extract these parameters and even in a plain coding application (analysis and resynthesis without manipulations) these methods produce degradation of the speech signal.

Other systems use a concatenative approach. In this approach, parametrized short speech segments of natural speech are connected to form a representation of the synthetic speech. The majority of the natural speech segments are merely transitions between pairs of phonemes. However, due to the large contextual variation of some phonemes, some segments consisting of three or more phoneme elements are often necessary; such elements consist of the transition from the first phoneme to the second, and a transition from the penultimate phoneme to the last, but the intermediate phonemes are stored completely. For the Bell Labs English system, there are approximately 2900 different speech elements — also called *dyads* — in the acoustic inventory, and these elements are sufficient to make up all the legal phoneme combinations for English.

The concatenative approach to speech synthesis requires that speech samples be stored in some parametric representation that will be suitable for connecting the segments and changing the signal's characteristics of loudness, F0, and spectrum. One method for changing the characteristics of natural speech is to analyze the speech in terms of a source/filter model, as diagramed in Fig. 46.1.

This model of speech synthesis has a variety of independent input controls. Starting at the left side of the figure, we show two possible source generators: a noise generator and a simple pulse generator. The noise generator has no controlling input whereas the pulse generator is controlled by the F0 parameter; the F0 parameter specifies the distance between any two pulses thus controlling the F0 of the periodic source. These inputs are selected by a switch which is controlled by a voicing flag. It is also possible to have a mixer to control the relative contribution of the noise and pulse source, and to insert a glottal pulse with additional controls for the shape of the glottal source in place of the simple pulse generator. To the right of the switch, we have a multiplier that multiplies the source by an amplitude parameter. This serves as the loudness control for the system. The signal from the multiplier is fed into a filter controlled by the filter coefficients which are varied slowly to shape the speech spectrum.

The source/filter model can be used to replicate naturally spoken speech when the parameters are obtained by analysis of natural speech. Speech can be parametrized by an amplitude control, voiced/voiceless flag, F0, and filter coefficients at a small interval (on the order of 5 msec. to 15 msec.) The loudness control is determined from the power of the speech at the time frame of the analysis. F0 extraction algorithms determine the voicing of the speech as well as the fundamental frequency. The filter parameters can be determined by various analysis techniques. The parameters obtained from the analysis can be used to drive the source filter model to reproduce the analyzed speech. However, these parameters can also be varied independently to change the speech. The ability to alter the analysis parameters is crucial to a concatenative approach, where the spectral parameters have to be smoothed and interpolated whenever two elements from different utterances are connected, or when

the duration of the speech has to be altered. Of course, the fundamental frequency of the original speech is completely discarded and replaced during synthesis by a rule-generated F0, as described earlier.

46.4 The Future of TTS

Using current methods such as those outlined in this chapter, it is possible to produce speech output that is of high intelligibility and reasonable naturalness, given unrestricted input text. (See [30] for a discussion of methods for evaluating TTS systems.) However, there is still much work to be done in all areas of the problem, including: improving voice quality, and allowing for greater user control over aspects of voice quality; producing better models of intonation to allow for more natural-sounding F0 contours; and improving linguistic analysis so that more accurate information on contextually appropriate word pronunciation, accenting, and phrasing can be computed automatically. This latter area — linguistic analysis — is particularly crucial: most high-quality TTS systems allow for user control of the output speech by means of various "escape sequences", which can be inserted into the input text. By use of such escape sequences, it is possible to produce highly appropriate and natural-sounding output. What is still lacking in many cases are natural-language analysis techniques that can mimic what a human annotator is able to do.

References

[1] Klatt, D., Review of text-to-speech conversion for English, *J. Acoustical Soc. Am.*, 82, 737–793, 1987.

[2] Allen, J., Hunnicutt, M.S. and Klatt, D., *From Text to Speech*, Cambridge University Press, Cambridge, 1987.

[3] van Heuven, V. and Pols, L., *Analysis and Synthesis of Speech: Strategic Research towards High-Quality Text-to-Speech Generation*, Mouton de Gruyter, Berlin, 1993.

[4] Sproat, R., Shih, C., Gale, W. and Chang, N., A stochastic finite-state word-segmentation algorithm for Chinese, in *Association for Computational Linguistics, Proceedings of 32nd Annual Meeting*, 66–73, 1994.

[5] Church, K., A stochastic parts program and noun phrase parser for unrestricted text, in *Proceedings of the Second Conference on Applied Natural Language Processing*, Association for Computational Linguistics, Morristown, NJ, 136-143, 1988.

[6] Sproat, R., English noun-phrase accent prediction for text-to-speech, *Computer Speech and Language*, 8, 79–94, 1994.

[7] Hirschberg, J., Pitch accent in context: Predicting intonational prominence from text, *Artificial Intelligence*, 63, 305–340, 1993.

[8] Koskenniemi, K., Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production, Ph.D. thesis, University of Helsinki, Helsinki, 1983.

[9] Coker, C., Church, K. and Liberman, M., Morphology and rhyming: Two powerful alternatives to letter-to-sound rules for speech synthesis, in *Proceedings of the ESCA Workshop on Speech Synthesis*, Bailly, G. and Benoit, C., Eds., 83–86, 1990.

[10] Golding, A., Pronouncing Names by a Combination of Case-Based and Rule-Based Reasoning, Ph.D. thesis, Stanford University, 1991.

[11] Yarowsky, D., Homograph disambiguation in speech synthesis, in *Proceedings of the Second ESCA/IEEE Workshop on Speech Synthesis*, 1994.

[12] Sproat, R., Hirschberg, J. and Yarowsky, D., A corpus-based synthesizer, in *Proceedings of the International Conference on Spoken Language Processing*, Banff, ICSLP, 563–566, Oct. 1992.

[13] O'Shaughnessy, D., Parsing with a small dictionary for applications such as text to speech, *Computational Linguistics*, 15, 97–108, 1989.
[14] Bachenko, J. and Fitzpatrick, E., A computational grammar of discourse-neutral prosodic phrasing in English, *Computational Linguistics*, 16, 155–170, 1990.
[15] Wang, M. and Hirschberg, J., Automatic classification of intonational phrase boundaries, *Computer Speech and Language*, 6, 175–196, 1992.
[16] Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J., *Classification and Regression Trees*, Wadsworth & Brooks, Pacific Grove, CA, 1984.
[17] Riley, M., Some applications of tree-based modelling to speech and language, in *Proceedings of the Speech and Natural Language Workshop*, DARPA, Morgan Kaufmann, Cape Cod, MA, 339–352, Oct. 1989.
[18] van Santen, J., Assignment of segmental duration in text-to-speech synthesis, *Computer Speech and Language*, 8, 95–128, 1994.
[19] Klatt, D., Linguistic uses of segmental duration in English: acoustic and perceptual evidence, *J. Acoustic Soc. Am.*, 59, 1209–1221, 1976.
[20] Syrdal, A.K., Improved duration rules for text-to-speech synthesis, *J. Acoustic Soc. Am.*, 85, S1(Q4), 1989.
[21] Fujisaki, H., Dynamic characteristics of voice fundamental frequency in speech and singing, in *The Production of Speech*, MacNeilage, P. Ed., Springer, New York, 39–55, 1983.
[22] Möbius, B., *Ein quantitatives Modell der deutschen Intonation*, Niemeyer, Tübingen, 1993.
[23] 't Hart, J., Collier, R. and Cohen, A., *A Perceptual Study of Intonation: An Experimental-Phonetic Approach to Speech Melody*, Cambridge University Press, Cambridge, 1990.
[24] Pierrehumbert, J.B., The Phonology and Phonetics of English Intonation, Ph.D. thesis, Massachusetts Institute of Technology, Sept. 1980.
[25] Ladd, D.R., *The Structure of Intonational Meaning*, Indiana University Press, Bloomington, Ind., 1980.
[26] Liberman, M. and Pierrehumbert, J., Intonational invariants under changes in pitch range and length, in *Language Sound Structure*, Aronoff, M. and Oehrle, R., Eds., MIT Press, Cambridge, 1984.
[27] Anderson, M., Pierrehumbert, J. and Liberman, M., Synthesis by rule of English intonation patterns, in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, vol.1, ICASSP, San Diego, 2.8.1–2.8.4, 1984.
[28] Silverman, K., Utterance-internal prosodic boundaries, in *Proceedings of the Second Australian International Conference on Speech Science and Technology*, Sydney, Australia, 86–91, 1988.
[29] Charpentier, F. and Moulines, E., Pitch-synchronous waverform processing techniques for text-to-speech synthesis using diphones, *Speech Commun.*, 9(5/6), 453–467, 1990.
[30] van Santen, J., Perceptual experiments for diagnostic testing of text-to-speech systems, *Computer Speech and Language*, 7, 49–100, 1993.