

# 16

## Text to-Speech (TTS) Synthesis

---

Juergen Schroeter  
*AT&T Laboratories*

16.1	Introduction .....	16-1
16.2	Overview of TTS .....	16-1
16.3	Front-End Issues.....	16-2
16.4	Back-End Issues.....	16-3
	Formant Synthesis • Concatenative Synthesis	
16.5	TTS Evaluation .....	16-10
16.6	Conclusions .....	16-11

### 16.1 Introduction

---

The goal of Text-to-Speech (TTS) synthesis is to convert arbitrary input text to intelligible and natural sounding speech so as to transmit information from a machine to a person. Therefore, TTS goes beyond simple “cut-and-paste” systems used, for example, in some telecom applications to read back a phone number. Such systems string together words spoken in isolation and the artifacts of such a scheme are often perceptible. The methodology used in TTS is to exploit acoustic representations of speech for synthesis (see Speech Signal Processing chapter in this handbook), together with linguistic analyses of text to extract correct pronunciations (“content”, *what* is being said) and prosody in context (“melody” of a sentence; *how* it is being said). Synthesis systems are commonly evaluated in terms of three characteristics: *accuracy* of rendering the input text (does the TTS system pronounce, e.g., acronyms, names, URLs, email addresses, a knowledgeable human would?), *intelligibility* of the resulting voice message (measured as a percentage of a test set that is understood), and perceived *naturalness* of the resulting speech (does the TTS sound like a recording of a live human?). Today, applications of TTS are in automated telecom services (e.g., name and address rendering), as a part of a network voice server for e-mail (E-mail by Phone), in directory assistance, as an aid in providing up-to-the-minute information to a telephone user (e.g., business locator services, banking services, helplines), in computer games, and last but not least, in aids to the handicapped (e.g., cosmologist Steven Hawking). For a much more detailed overview of TTS and its applications, see Ref. 1 and Ref. 2.

### 16.2 Overview of TTS

---

A block diagram of a general TTS engine is depicted in Figure 16.1. We distinguish a TTS “front-end” (i.e., the part of the system closer to the text input) from a TTS “back-end” (i.e., the part of the system that is closer to the speech output). Input text, optionally enriched by tags that control prosody or other characteristics, enters the front-end where a text analysis module detects the document structure (in terms of, e.g., lists vs. running text, paragraph breaks, sentence breaks, etc.), followed by text normalization (expansion to literal word tokens,

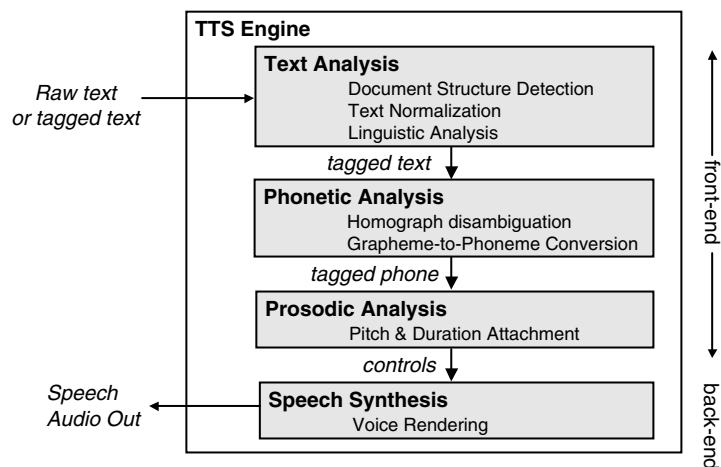


FIGURE 16.1 Block diagram of a general text-to-speech system.

encompassing transcription of acronyms, abbreviations, currency, dates, times, URLs, etc.), and further linguistic analysis that enables other tasks down the line. The tagged text then enters a phonetic analysis module that performs *homograph* disambiguation, and *grapheme-to-phoneme* conversion. The latter process is also called “letter-to-sound” conversion. The string of tagged *phones* enters a prosodic analysis module that determines *pitch*, duration (and amplitude) targets for each phone. Finally, the string of symbols that was derived from a given input sentence is passed on to the speech synthesis module where it controls the voice rendering that corresponds to the input text.

### 16.3 Front-End Issues

The text analysis and normalization module in the front-end determines to a large extent the “what” and “how” of the resulting synthetic speech. Note that punctuation (e.g., “;”, “?”) is not infallible. For example, the TTS system should not misinterpret the dot after “in” in the example “The table is 36.5 in. long” as the end of the sentence. In addition, punctuation and other special characters can be part of a time (e.g., 7:30 pm), or date (e.g., 5/25/2004), or currency expression (\$10 = “ten dollars”). Text normalization is difficult because it is context sensitive (e.g., \$1.5 million = “one point five million dollars”).

Abbreviations and acronyms fall in either of two categories. The first category contains a finite set of known “mappings” such as “Dr.” in the sentence “Dr. Smith lives on Smith Dr.” Note that a mapping may be ambiguous (Dr. can be “doctor” or “drive”). More difficult to handle, however, is the open category of abbreviations that people invent on the fly. COMM could mean “communications,” “committee,” etc. Other possible short forms are CMNCTNS, or COMMS. Also, a specific abbreviation can have different expansions depending on the task or topic domain. For example, DC could mean “direct current” in this book, but could mean “District of Columbia” as part of an address. Therefore, it may be necessary to handle certain text normalization tasks in form of a domain-specific text “filter” that would alter the raw text before it is passed on to the TTS system depicted in Figure 16.1. Applications like e-mail reading or web page reading, for example, also require text filters to strip out mundane header or formatting information. Even the “simple” reading of numbers can be difficult, such as “370,” where the 370 can be part of a phone number (370–1111, read as “three-seven-zero. . .”) or part of a name (e.g., IBM370, read as “i-b-m-three-seventy”). Note that the performance of the text analysis and normalization module affects the *accuracy* rating of a TTS system.

Linguistic analysis in the front-end encompasses the determination of parts-of-speech (POS), word sense, emphasis, appropriate speaking style, and speech acts (such as, e.g., greetings, apologies, etc.). A linguistic parser could be used, but typically only a shallow analysis is done for reasons of computational speed.

Grapheme-to-Phoneme conversion involves word pronunciation. The mapping from orthography to phonemes can be difficult because of context dependence. This is usually handled by trained Classification and Regression (decision) Trees (CARTs) that capture the probabilities of specific conversions given the context or the fact that a specific word is a homomorph. Grammatical (POS) analysis helps (“*Wind* down and let the *wind* blow”, “*Lead* the way down by following the way *lead* drops.”) At the word level, pronunciation dictionaries, combined with *morphological* decomposition are used. Finally, letter-to-sound rules are employed as a fall-back in most TTS systems. However, names are a difficult problem since even people with the same name may pronounce it differently and the identification of a name in the text is also difficult (e.g., “Begin met the president” vs. “Begin the work now!”). Finally, note that pronunciation is very much speaker and accent-dependent, even within the same language. Because of these difficulties, automation of creating speaker-dependent pronunciation dictionaries currently is an active area of research.

Prosody determines *how* a sentence is spoken in terms of melody, phrasing, rhythm, accent locations, and emotions. Prosody may even carry meaning, even in non-tonal languages (e.g., “I like the Eggs Benedict” vs. “I like the eggs, Benedict”). Therefore, prosody affects naturalness *and* intelligibility ratings of a TTS system. Prosody is difficult to annotate automatically. Mainly for use in manual annotation efforts, so-called Tone and Break Indices (ToBI) have emerged as the standard [3]. Speech signal correlates of prosody are the presence and duration of pauses, the pitch (fundamental frequency value, in particular, its dynamic behavior), as well as phone durations and amplitudes. As in the case of pronunciation, prosody can be dependent on the speaker gender, on the specific speech act or application task, and even on the individual speaker. This may be one reason why researchers now seem to move to using data-driven prosody approaches that employ large speech databases of a single speaker over rule-based systems [4].

## 16.4 Back-End Issues

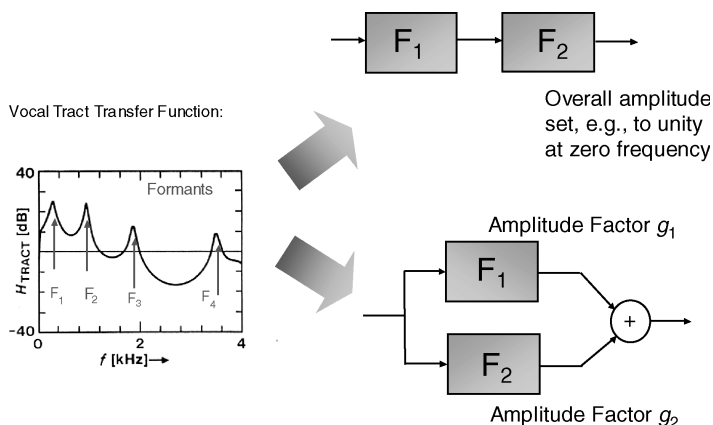
---

An important decision a designer of a TTS system needs to make is which synthesis method to use. In the literature [5] we find two basic categories of methods: *synthesis by rule* and *concatenative synthesis*. Synthesis by rule exploits the expert knowledge of speech scientists in speech production and perception by putting the human expert in the design (and quality) loop. Concatenative synthesis employs recordings of a human speaker, inherently putting more emphasis on data. Experts generally disagree on which is the more promising approach although, today, the concatenative approach clearly produces higher quality synthetic speech.

*Articulatory synthesis* uses computational models of the articulators (e.g., tongue, lips, etc.) and the glottis (voice box) to synthesize speech. Instead of describing the speech signal itself, it employs control parameters such as tongue position and movement, lip and glottal opening etc., that are meaningful in speech production. Therefore, articulatory synthesis has appeal to speech scientists that explore speech production theories and related topics. Mathematically, articulatory synthesizers can be as simple as describing the vocal tract as a straight tube of variable cross section [6], or as complicated as solving the full blown Navier-Stokes equations [7]. Early articulatory synthesizers clearly followed the synthesis by rule approach [8,9]; recently, researchers have adopted some data-driven concepts [10]. Although the method of choice for a few specific uses, articulatory synthesis has so far not delivered synthesis that can be mistaken as a recording of a specific speaker. In addition, deriving rules or dynamic system control parameters for it is computationally expensive and/or requires speech production measurements (e.g., dynamic MRI [11], tracing of x-ray pellets [12]) that may be considered as somewhat “invasive”. However, a highly accurate articulatory synthesizer still carries the promise of producing completely “tunable” (e.g., made to sound like different speakers, speaking styles, etc.) high-quality speech that then could be used to “train” other, more practical, synthesizers.

### Formant Synthesis

Within the rule-based synthesis category, *formant synthesis* is the most prominent example. Well known examples include efforts by Dennis Klatt [13,14], whose TTS system later was productized as “MITalk” and “DECTalk”, and is now maintained and sold commercially by Fonix. In its simplest form, formant synthesis employs second-order filter sections either in cascade/series, or in a parallel structure. Figure 16.2 depicts these



**FIGURE 16.2** The vocal-tract transfer function (left) shows four formants in the frequency range from 0 to 4 kHz. It can be approximated in two different ways using second-order filter sections: serial (top right), or parallel (bottom right).

two options. Starting from the vocal tract transfer function on the left that relates volume flow at the lips (output) to volume flow at the glottis (input), the task is to approximate all vocal tract resonances (peaks in the transfer function, “formants”) by a network of second-order filters, depicted on the right. It can be shown that the series representation of filters (top right) approximates a non-nasal (no nasal side branch) vocal tract reasonably well. For example, given about three to five filter sections (each matching one formant), we are able to approximate vocal tract transfer functions for vowels even between the formant frequencies (peaks). In this approach, we only need to specify formant frequencies and bandwidths, plus an overall gain factor. However, for nasal sounds, as well as for any fricative or mixed (voiced/unvoiced) sounds, a series representation of second-order filters may not be good enough. Parallel filters (shown in the bottom right of Figure 2) create the flexibility to approximate *any* speech spectrum (see, e.g. Ref. 15), but require individual gain factors, in addition to formant frequencies and bandwidths. As a side effect, parallel filters also introduce spectral zeros between the formant frequencies that may be cancelled by special correction filters.

The source excitation can be either voice pulses (approximating glottal pulses) or noise (approximating fricative or aspirative noise sources in the vocal tract), or mixtures of the two. For example, for the word “two” the /t/phoneme pattern might specify a sudden (burst) onset of noise excitation of a filter with a flat characteristic followed by the /uw/pattern of voice excitation of three formant filters with peak frequencies at 300, 850, and 2250 Hz, representing formants F1, F2, and F3. Note that in this case, one usually also sees a transition region between the burst and the vowel that is filled with aspiration noise but already shows the formant structure of the following vowel.

Deriving rules for synthesizing running speech is the main problem for formant synthesis. These rules specify the timings of source (voiced/unvoiced) and the dynamic values of all filter parameters. This is difficult to do by hand, even for simple words, let alone full sentences at high intelligibility. Automatic derivation of these rules can be achieved by analysis-by-synthesis approaches that try to mimic input speech [16]. A commercial system that makes use of articulatory and acoustic-phonetic knowledge and drives the Klatt synthesizer, is HLSyn [17].

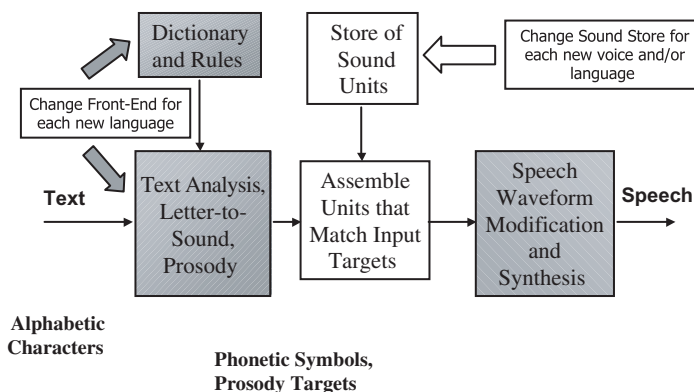
Formant synthesizers have moderate computational requirements. Some implementations allow a full TTS system to run within 2 MB of memory or less. Therefore, embedded applications, such as in handheld devices, for example, talking dictionaries, calendars, etc., or even in cellphones (e.g., for reading back names and key presses for car drivers that should not take their eyes off the road) are enabled by formant synthesis. Voice quality can be controlled to a high degree, but it is usually impossible to match the voice quality of a given target speaker. Intelligibility, however, is usually high. Finally, formant synthesis is highly appropriate for creating speech stimuli for research in speech perception, given the high level of control such experiments require.

## Concatenative Synthesis

Concatenative synthesis uses actual short segments of recorded speech that were cut from recordings and stored in an inventory (“voice database”), either as “waveforms” (uncoded), or encoded by a suitable speech coding method (see section on Speech Signal Processing in this handbook and Section 4.3 below). A block diagram of a typical concatenative TTS system is shown in Figure 16.3. As described in more detail above, and previously depicted in Figure 1, the front-end on the left converts a given input text string into a string of phonetic symbols and prosody (fundamental frequency, duration, and amplitude) targets. The front-end employs a set of rules and/or a pronunciation dictionary. Together with a string of phonetic symbols, it produces target values for fundamental frequency (pitch), phoneme durations, and amplitudes. The center block in Figure 16.3 assembles the units according to the list of targets set by the front-end. These units are selected from a store (top center in Figure 16.3) that holds the inventory of available sound units.

Different types of speech units may be stored in the inventory of a concatenative TTS system. Storing whole word units is impractical for general TTS because of the tremendous demands on a voice talent that would have to read a few hundreds of thousands of words in a consistent voice and manner. Even if recorded successfully in multiple sessions spread over several weeks, a lack of coarticulation and phonetic recoding at word boundaries may result in unnatural sounding speech. On the other extreme, using phones (e.g., about 50 for English) is also unsatisfactory because of the large coarticulatory effects that exist between adjacent phones. Therefore, transitions from one unit to the next may be audible as “glitches” that introduce perceptually disruptive discontinuities. Intuitively, longer units are more likely to result in higher quality synthesis, given that the rate of concatenations (how many unit-to-unit transitions occur per second of speech) is lower than in the case of shorter units. On the other hand, we need a larger set of longer units to “cover” any application domain, for example, travel (with TTS-generated prompts such as “From which airport do you want to leave?”), because of the tremendous multiplicity of possible unit variants [18]. Given these contradictory requirements, most practical TTS implementations until the mid-1990s compromised by using one of two types of inventory units, the diphone and the demissyllable.

A diphone is the snippet of speech from the middle of one phone to the middle of the next phone. (Note that the average length of a diphone is identical to that of a phone!) The middle of a phone tends to be its acoustically most stable region. Therefore, diphones represent acoustic transitions from the stable midsection of one phone to the next. A minimum inventory of about 1,000 diphones is required to synthesize unrestricted English text [19]. Because concatenative synthesis preserves the acoustic detail of natural speech, diphone synthesis is generally highly intelligible. A disadvantage of strict diphone synthesis is that coarticulation is only



**FIGURE 16.3** Block diagram of a concatenative text-to-speech system showing the front-end (left), and back-end (right), plus the sound store in the center that holds the voice inventory.

provided with the immediately preceding and following phonemes, whereas some phonemes can affect the articulation over several phonemes. Demisyllables are alternative units for concatenative synthesis [20]. A demisyllable encompasses half a syllable; that is, either the syllable-initial portion up to the first half of the syllable nucleus, or the syllable-final portion starting from the second half of the syllable nucleus. The number of demisyllables in English is roughly the same as the number of diphones. Because demisyllable units are usually longer than diphones, and allow for better capture of coarticulation effects compared to diphones, they should pose fewer concatenation problems.

Note that a typical database (a database that covers all possible diphone units in a minimum amount of sentences) might contain as little as 30 minutes of speech of a single speaker (voice), given that the units must be modified by signal processing to match the front-end predictions and to smooth over the concatenation points. In the following, we will look into some of the signal representations used in TTS. Note, however, that the latest high-quality TTS systems all employ voice databases containing many hours of recorded speech, because *not* having to modify a segment of speech (because you have the right one in the inventory) will always produce higher quality speech synthesis.

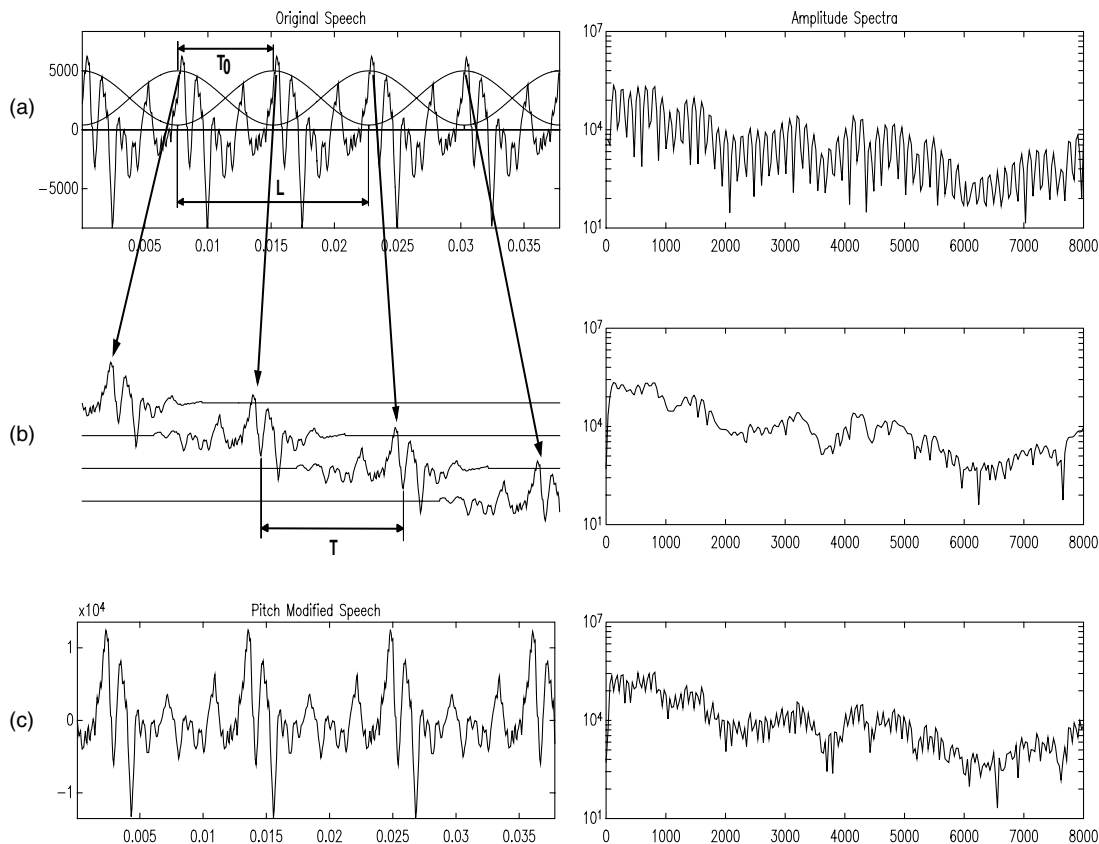
### Speech Signal Representation for Concatenative Synthesis

A good speech signal representation for concatenative synthesis approximates the following set of requirements:

1. The speech signal can be stored in a highly compressed (i.e., coded) form so that a large voice database can be used even under tight memory limitations. Coder and decoder are of low computational complexity.
2. Coding/decoding is perceptually transparent. Since we would like to mimic all the voice characteristics of a real person, subjecting the speech signal to “vocoder”-like degradations will not lead to speech synthesis of high naturalness.
3. Coding algorithms (see chapter on Speech Signal Processing) have to allow for “random access.” Since most speech coders contain some sort of autoregressive memory, all state variables of the coder have to be made available at concatenation points since the decoder will have to switch between units of speech that are very unlikely to have been recorded consecutively in time.
4. An ideal speech representation must allow for natural-sounding modifications of pitch, duration, and amplitude. This is particularly important for small inventories with one, or just a few, “typical” examples for each unit. Unfortunately, experience shows that, for most signal processing algorithms, modifying pitch more than a few percent may destroy perceived naturalness; that is, a pitch-modified speech signal is likely to be perceptually much different from a speech signal that has been recorded without modifications from the speaker producing the desired pitch value directly. (This is the reason why “singing TTS” does not sound like an opera star.)
5. For some advanced applications, it even might be desirable to allow for fine-tuning of the voice, for example, to add more aspiration, mellowness, or let the voice “scream” when needed. Instead of recording different voice inventories for different speaking “styles,” advanced “voice conversion” might be used to approximate an “angry” voice using a “happy” (or “neutral”) voice as a starting point. Today, algorithms for voice conversion (usually concerned with converting the speech of one speaker to sound like speech from another speaker) still do not produce consistently good enough results for sounding like the “real thing,” but might be sufficient for applications such as computer games where even the original voice does not sound “human.”

Given this set of requirements, in the following, we will briefly touch on three classes of speech signal processing algorithms with their (native) speech representations: low-complexity time-domain algorithms such as TD-PSOLA and its variants, LPC-based algorithms, and frequency domain-based speech representations.

Time Domain Pitch-Synchronous Overlap Add (TD-PSOLA [21]) consists of cutting exactly two pitch periods from a voiced speech signal (a vowel), windowing each segment with a Hanning window centered on



**FIGURE 16.4** Pitch modification using TD-PSOLA. The lengths of individual pitch epochs is modified by adding up neighboring segments to form the pitch-modified output speech. (Source: T. Dutoit, (1997). *An Introduction to Text-to-Speech Synthesis*. Kluwer Academic Publishers, Dordrecht/Boston/London; Figure 10.1. With permission.)

one glottal closure (maximum excitation) point and weighing down the signal in the vicinity of the previous and next glottal closure points. This process is depicted in Figure 16.4 (a) and (b). For all panels (a) to (c), on the left, we see time-domain signals, and on the right, spectra. The extracted and tapered signal traces (b) are recombined (via “overlap-add”) in (c), after shortening (for increased pitch) or padding with zero amplitude signal samples (for lowering pitch). The resulting reconstructed signal of a different pitch value is shown in panel (c). TD-PSOLA, although extremely efficient and widely used, can introduce audible glitches at concatenation points because it has no inherent way of smoothing the transition, which happens abruptly. A variant of TD-PSOLA may use the LPC filter excitation (“PSREL” [22]) instead of the speech waveform directly. This allows for smoothing of the spectral envelope at concatenation points, but still leaves the task of smoothing the LPC excitation itself.

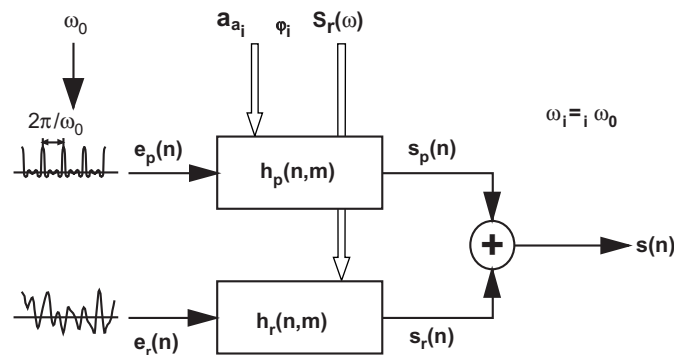
Some TTS systems employ modified LPC-based (see chapter on Speech Signal Processing) coder/decoders such as CELP or multi-pulse [23], and/or coders that employ glottal model excitation pulses. The latter approximate true glottal flow waveforms, but can be synthesized given a small set of parameters. Since such parameters can be smoothed over a few pitch periods, the concatenation problem can be addressed. However, it is very difficult to achieve a “transparent” coder/decoder system using this approach. Results usually sound a bit “buzzy.”

Finally, so-called “hybrid” speech representations such as the Harmonic-plus-Noise Model (HNM) [24] make use of the fact that the speech spectrum of a voiced sound tends to be composed of two distinct

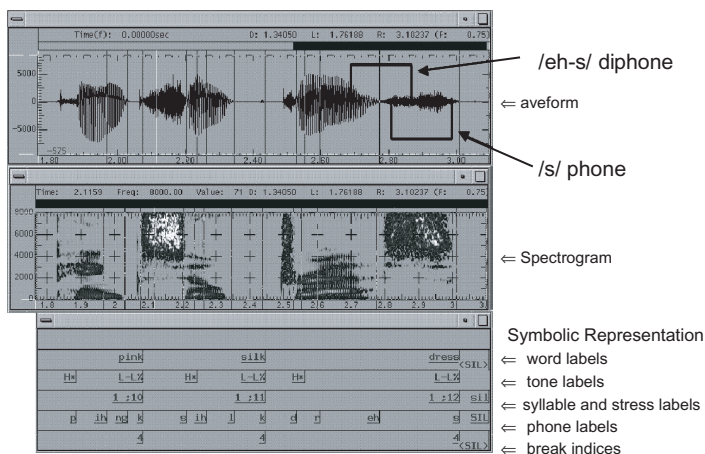
components: a harmonic (i.e., periodic) part, mostly at lower frequencies and highly relevant for representing a specific speaker, and a stochastic (noise-like) part, mostly at higher frequencies. Consequently, *two* separate synthesizers might be used as depicted in Figure 16.5: a harmonic (or sinusoidal, [25]) synthesizer, shown at the top, and an LPC-based synthesizer using a high-pass filtered stochastic (noise) excitation at the bottom of the figure. Note that the harmonic synthesizer is controlled by parameters like the fundamental frequency  $\omega_0$ , the amplitudes  $\alpha_i$  and phases  $\phi_i$  for the  $i$ -th harmonic, and the parameters of an optional time-varying filter with the impulse response  $h_p(n, m)$ . It synthesizes the harmonic speech waveform  $s_p(n)$ . The stochastic synthesizer (bottom) consists of a time-varying filter with the impulse response  $h_r(n, m)$ , excitation signal  $e_r(n)$ , and creates the output speech waveform  $s_r(n)$ . Both speech components are then added to form the full-band speech signal  $s(n)$ . Note that the HNM and similar approaches allow for sophisticated spectral and excitation smoothing at concatenation points. Another advantage is that hybrid synthesizers, such as HNM, can exploit any relevant knowledge of speech perception. However, one drawback of the hybrid approach is their relatively high computational complexity.

### Voice Creation for Concatenative Synthesis

As stated above, concatenative speech synthesis exploits recorded speech that forms the content of the speech inventory. An example fragment from such a database is shown in Figure 16.6. The top panel shows the time waveform of the recorded speech signal for the words “pink silk dress,” the middle panel shows the spectrogram (“voice print”), and the bottom panel shows the annotations that are needed to make the recorded speech useful for concatenative synthesis. For the word “dress,” we have highlighted the phone /s/ and the diphone /eh-s/ that encompasses the latter half of the /eh/ and the first half of the /s/ of the word “dress.” Until recently, expert labelers were employed to examine waveform and spectrogram, as well as using their sophisticated listening skills, in order to produce annotations (“labels”) such as those shown in the bottom panel of the figure. The set consists of word labels (time markings for the end of words), tone labels (symbolic representations of the “melody” of the utterance and syllable and stress labels, all labeled in the ToBI standard [3]), phone labels, and break indices (that distinguish between breaks between words, subphrases, and sentences, for example). Experience shows that expert labelers need approximately 100 to 250 seconds of work time to label one second of speech with the set depicted in Figure 6. For a diphone-based synthesizer, this might be a reasonable investment, given that a “diphone-rich” database (a database that covers all possible diphones in a minimum amount of sentences) might be as short as 30 minutes. Clearly, manual labeling would be impractical for much larger databases (dozens of hours), and/or if we were interested in creating



**FIGURE 16.5** A hybrid harmonic/stochastic synthesizer. (Source: T. Dutoit, (1997). *An Introduction to Text-to-Speech Synthesis*. Dordrecht/Boston/London: Kluwer Academic Publishers, Figure 9.2. With permission.)



“..p-ink s-ilk dre--s--s.”

**FIGURE 16.6** Screenshot of a speech labeler’s screen showing waveform (top), spectrogram (middle), and edited labels (bottom) for the speech segment corresponding to the text “pink silk dress.”

many voices, factory-style. In such cases, we would require fully automatic labeling, using speech recognition tools. Fortunately, these tools have become so good, that speech synthesized from an automatically labeled speech database may be of higher quality than speech synthesized from the same database that has been labeled manually [26].

Automatic labeling tools fall into two categories: automatic phonetic labeling tools to create the necessary phone labels and automatic prosodic labeling tools to create the necessary tone and stress labels, as well as break indices. Automatic phonetic labeling is adequate, provided it is done with a speech recognizer in “forced alignment mode” (i.e., with the help of the known text message so that the recognizer is only allowed to choose the proper phone boundaries but not the phoneme identities). The speech recognizer also needs to be speaker-dependent (i.e., be trained on the given voice), and has to be properly bootstrapped from a small manually labeled corpus for best results. Finally, it should be obvious that both, the labeling tool and the TTS system that is the target for the voice database to be labeled, should use identical phone labels and unit definition conventions. Automatic prosodic labeling tools work from a set of linguistically motivated acoustic features (e.g., normalized durations, maximum/average pitch ratios) plus some binary features looked up in the lexicon (e.g., word-final vs. word-initial stress) [27], plus the output from the phonetic labeling.

Recording large voice databases can bring with it daunting organizational tasks that should not be underestimated. Selecting the right voice talent, choosing the optimal material to record, providing a consistent recording environment (low background noise, negligible acoustic reflections from walls, tables, manuscript, constant microphone position relative to mouth, etc.), and ensuring a correct and consistent speaking style are all part of the effort.

### Unit Selection Synthesis

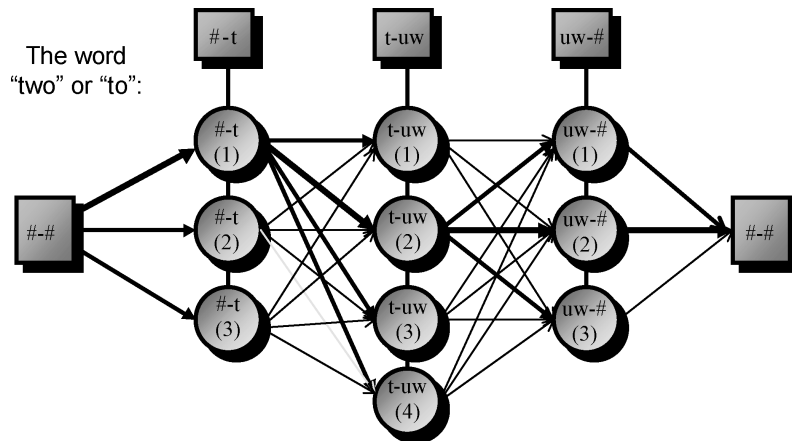
With the availability of good automatic speech labeling tools, concatenative speech synthesis has now embraced the use of multi-hour voice databases. With the availability of several (potentially hundreds of thousand) instances of a specific type unit (only different in pitch, duration, linguistic context at recording time), so-called *Unit-Selection Synthesis* has become viable for obtaining high-quality TTS. Based on early research done at ATR in Japan [28–30], this method enables the use of large speech databases recorded using specific, carefully crafted and controlled, speaking styles (e.g., angry, happy, apologetic, etc.). In addition, a given database may be focused on narrow-domain applications (such as “travel reservations” or “telephone

number synthesis”) that commonly allow for the use of smaller databases for a preset level of quality, or it may be used for general applications like e-mail or news reading (requiring a larger voice database). In the latter case, unit-selection synthesis can require on the order of ten hours of recording of spoken general material to achieve a desired level of quality, and several dozen hours for “natural quality” (that can be mistaken for direct recordings). In contrast with earlier concatenative synthesizers, unit-selection synthesis automatically picks the optimal synthesis units (on the fly) from an inventory that may contain thousands of tokens of a specific unit, and concatenates the selected units to produce the synthetic speech. This is in stark contrast with the fact that as late as the mid-1990s voice inventories for concatenative TTS were always carefully crafted by hand, trying to find the one or few units of each type that lead to optimum results in all possible synthesis scenarios (contexts). In more than one sense, the unit-selection approach has succeeded in putting “the expert into the box,” that is, has automated the process of finding the optimal sequence of inventory units given a “search query” of tagged phoneme strings. One important difference between “old” and “new” TTS is that unit-selection synthesis “knows” the text to be synthesized at selection time, while previous methods tried to satisfy more global selection criteria *without* explicit knowledge of the text to be synthesized.

The unit selection “database query” process is outlined in Figure 16.7, which shows how the method must dynamically find the best path through the unit-selection network corresponding to the diphones for the word ‘two.’ (Note that, in practice, a better choice would be to use half-phones instead of diphones because half-phones allow the search algorithm to create diphones on the fly from two half-phones that were not recorded in sequence.) For any query, the optimal choice of units selected from the database depends on factors such as spectral similarity at unit boundaries (components of the “join cost” between two units) and on matching prosodic targets set by the front-end (components of the “target cost” of each unit).

## 16.5 TTS Evaluation

Evaluation of TTS systems is currently a much discussed research topic. Here, we can only give broad guidelines. The three quality criteria mentioned in the beginning of this section, accuracy, intelligibility, and naturalness, overlap with respect to which part of a TTS system impacts which criteria. Accuracy, that is, the ability to read a given input text the way a knowledgeable human reader would, is the one criterion solely



**FIGURE 16.7** Viterbi search of diphone inventory to select units for synthesizing the word “two” or “to” in isolation (i.e., between silence /#/ segments). Arrow line width symbolizes appropriateness of a given transition (thicker is better), evaluated, e.g., in terms of spectral match across unit boundaries. Candidates in each of the three columns are assumed to be ordered according to their suitability in terms of pitch, context match between recording and synthesis, etc.

homed in the TTS front-end. Contrary to this, unsatisfactory intelligibility or naturalness is much harder to pinpoint. Between the latter two, designers of formant synthesizers should aim at maximizing intelligibility, accepting the fact that they are unlikely to deliver high naturalness. On the other hand, some of the newer concatenative synthesis systems may overemphasize naturalness with the undesired result of less than stellar intelligibility.

Accuracy can be evaluated by running a test corpus of task-relevant acronyms, abbreviations, embedded in context-defining input text through the TTS front-end and judging the generated output *text* (or transcribing the output speech back to text). Evaluating intelligibility and/or naturalness requires conducting elaborate listening tests. Intelligibility tests, known from testing speech coders, present word or sentence lists and let subjects transcribe words they heard [31] on TTS-related testing. Unfortunately, TTS intelligibility has many more aspects to it than are covered by standard word-list driven intelligibility tests. On a sentence level, for example, a wrong prosody can destroy intelligibility/comprehension. So far, a generally accepted standard intelligibility test for TTS systems is lacking. For overall quality evaluation, the International Telecommunication Union (ITU) recommends a specific method [32] that, in this author's opinion, are suitable for also testing "naturalness." Such tests involve five (or more) point rating scales for characteristics such as "overall impression," "listening effort," "comprehension," etc. Alternatively, listeners might be asked to state their preference (A/B tests) of which one of two systems sounds "better." Selection of the test material is also very relevant. For example, some systems sound very good for short sentences spoken in isolation, but show weaknesses when longer paragraphs of text are being rendered. Generally, a rule-of-thumb is to use material from the intended application. For more details on quality testing, see Ref, 33.

## 16.6 Conclusions

---

This section has highlighted some aspects of Text-to-Speech (TTS) synthesis with a slant towards catering to electrical engineers. Many aspects, such as, for example, prosody generation, natural language processing, and others, have been skimmed only for space reasons. It is clear that TTS systems have come a long way towards delivering high-quality output to listeners that sometimes fools them to believe that they are listening to recordings. This said, it is also clear that we are still far from delivering the perfect synthesis for all possible applications. Shorter-term, the best way towards high-quality synthesis seems to be to tailor TTS specifically for a given application. Both the front-end and the back-end of a TTS system can be optimized for this purpose. For example, including and maintaining a pronunciation dictionary of names of all prescription drugs on the market could be essential for using TTS in a healthcare application. Eliciting the desired voice characteristics from a voice talent that is being recorded for a unit-selection synthesis voice database could be essential for customers accepting an automated dialog system that speaks with a TTS voice. Last, not least, typical engineering choices, such as trading off memory vs. speed, quality vs. complexity, and time in development vs. market pressures, are very relevant also for TTS systems.

### Defining Terms

**Homographs** are words that are spelled the same but pronounced differently like, e.g., the present and past tense form of the verb "to read."

**Graphemes** are "functional spelling units" encompassing one or more letters of the text input; a grapheme in the text input corresponds to a single phoneme.

**Morphology** deals with the units of meaning.

**Pitch** is the fundamental frequency of vibration of the vocal cords that produce voiced sounds such as vowels.

**Phones** characterize any sound that can be produced by a human vocal tract; if a phone is part of a specific language, it becomes a phoneme of that language.

**Phonemes** are the elementary sounds of a language, such as /ow/ in the word "boat."

## References

1. C. Bickley, A. Syrdal, and J. Schroeter, "Speech Synthesis," in *The Acoustics of Speech Communication*, J.M. Picket, Ed., Boston, NY: Allyn and Bacon, 1998.
2. A. Syrdal, R. Bennett, and S. Greenspan, Eds., *Applied Speech Technology*, Boca Raton: CRC Press, 1995.
3. K. Silverman, M. Beckman, J. Pierrehumbert, M. Ostendorf, C. Wightman, P. Price, and J. Hirschberg, *ToBI: A Standard Scheme for Labeling Prosody*, ICSLP, 1992, pp. 867–879, Banff.
4. Y. Sagisaka, N. Campbell, and N. Higuchi, Eds., *Computing Prosody – Computational Models for Processing Spontaneous Speech*, Berlin: Springer, 1997.
5. T. Dutoit, *An Introduction to Text-to-Speech Synthesis*, Dordrecht/Boston/London: Kluwer Academic Publishers, 1997.
6. J. Schroeter, and M.M. Sondhi, "Speech coding based on physiological models of speech production," in *Advances in Speech Signal Processing*, S. Furui and M. Mohan Sondhi, Eds., New York: Marcel Dekker, 1991, pp. 231–268.
7. G. Richard, M. Liu, D. Sinder, H. Duncan, Q. Lin, J. Flanagan, S. Levinson, D. Davis, and S. Slimon, "Numerical simulations of fluid flow in the vocal tract," in *Proc. of Eurospeech*, Madrid, Spain, September 18–21, 1995.
8. K. Ishizaka and J.L. Flanagan, "Synthesis of voiced sounds from a two-mass model of the vocal cords," *Bell Syst. Tech. J.*, vol. 51, no. 6, pp. 133–1268, 1972.
9. C.H. Coker, "A model of articulatory dynamics and control," *Proc. IEEE*, vol. 64, no. 4, pp. 452–460, 1976.
10. M.M. Sondhi and D.J. Sinder, "Articulatory modeling: a role in concatenative text-to-speech synthesis," in *Text to Speech Synthesis: New Paradigms and Advances*, A. Alwan and S. Narayanan, Eds., Englewood Cliffs, NJ: Prentice Hall, 2003.
11. S. Narayanan, K. Nayak, S. Lee, A. Sethy, and D. Byrd, "An approach to real-time magnetic resonance imaging for speech production," *J. Acoust. Soc. Am.*, in press, 2004.
12. S. Kiritani, S.K. Itoh, and O. Fujimura, "Tongue-pellet tracking by a computer controlled X-ray microbeam system," *J. Acoust. Soc. Am.*, 57, 1516–1520, 1975.
13. D.H. Klatt, "Software for a cascade/parallel formant synthesizer," *J. Acoust. Soc. Am.*, vol. 67, no. 3, 971–995, 1980.
14. J. Allen, M.S. Hunnicutt, and D. Klatt, *From Text to Speech, The MITTalk System*, Cambridge: Cambridge University Press, 1987.
15. J.N. Holmes, "Formant Synthesizers: Cascade or Parallelm" *Speech Commun.*, 2, 251–273, 1983.
16. S. Parthasarathy and C. Coker, "On automatic estimation of articulatory parameters in a text-to-speech system," *Comput. Speech Language*, 6, 37–75, 1992.
17. H.M. Hanson, and K.N. Stevens, "A quasiarticulatory approach to controlling acoustic source parameters in a Klatt-type formant synthesizer using Hlsyn," *J. Acoust. Soc. Am.*, 112, 1158–1182, 2002.
18. J.P.H. van Santen, "Combinatorial issues in text-to-speech synthesis," in: *EuroSpeech'97: 5th Eur. Conf. Speech Commun. Technol.*, 5, 2511–2514, 1997.
19. J.P. Olive, "Rule synthesis of speech from diadic units," in *Proc. ICASSP-77*, 1977, pp. 568–570.
20. O. Fujimura and J. Lovins, "Syllables as concatenative phonetic elements," in *Syllables and Segments*, A. Bell and J.B. Hooper, Eds., New York: North-Holland, 107–120, 1978.
21. E. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Commun.*, vol. 9, no. 5–6, pp. 453–467, 1990.
22. M. Macchi, M.J. Altom, D. Kahn, S. Singhal, and M. Spiegel, "Intelligibility as a function of speech coding method for template-based speech synthesis," in *Proc. Eurospeech'93*, Berlin, Germany, 1993, pp. 893–896.
23. W. Kleijn and K. Paliwal, Eds., *Speech Coding and Synthesis*, Amsterdam: Elsevier, 1995.
24. Y. Stylianou, "Applying the harmonic plus noise model in concatenative speech synthesis," *IEEE Trans. Speech Audio Process.*, vol. 9, no. 1, pp. 21–29, 2001.

25. T.F. Quatieri, R.J. McAulay, "Shape Invariant time-scale and pitch modification of speech," *IEEE Trans. Signal Process.*, vol. 40, no. 3, pp. 497–510, 1992.
26. M.J. Makashay, C.W. Wightman, A.K. Syrdal, and A.D. Conkie, "Perceptual evaluation of automatic segmentation in text-to-speech synthesis," in *Proc. ICSLP 2000*, Beijing, China, October, 2000.
27. C.W. Wightman, A.K. Syrdal, G. Stemmer, A.D. Conkie, and M.C. Beutnagel, "Perceptually based automatic prosody labeling and prosodically enriched unit selection improve concatenative text-to-speech synthesis," in *Proc. ICSLP 2000*, Beijing, China, October, 2000.
28. Y. Sagisaka, N. Kaiki, N. Iwahashi, and K. Mimura, K., "ATR – v-TALK speech synthesis system," in *Proc. Int. Conf. Speech Language Process. 92*, Banff, Canada, vol. 1, 1992, pp. 483–486.
29. A. Hunt and A.W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *Proc. ICASSP-96*, 1996, pp. 373–376.
30. M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal, "The AT&T Next-Gen TTS System," in *Proc. Joint Meet. ASA EAA DEGA*, Berlin, Germany, March 1999, available on-line at <http://www.research.att.com/projects/tts/pubs.html>.
31. M.F. Spiegel, M.J. Altom, and M.J. Macchi, "Comprehensive assessment of the telephone intelligibility of synthesized and natural speech," *Speech Commun.* 9, 279–291, 1990.
32. ITU-T Recommendation P.85, "A Method for Subjective Performance Assessment of the Quality of Speech Output Devices," International Telecommunications Union publication, 1994.
33. Y.V. Alvarez and M. Huckvale, "The reliability of the ITU-T P.85 standard for the evaluation of text-to-speech systems," in *Proc. ICSLP2002*, Denver, Colorado, Session TuB9p.8, 329-332, 16–20 September 2002.

## Further Information

There is a treasure of information on speech synthesis available on the web.

For information on how TTS fits in and what it contributes to Language Technology, see, for example: <http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html> (Chapter 5, spoken output technologies).

To get started on speech synthesis, readers may want to explore the Festival project: <http://www.cstr.ed.ac.uk/projects/festival/>, and the follow-up project at <http://www.festvox.org>.

Finally, people interested in the earlier efforts in speech synthesis find historic examples at <http://www.cs.indiana.edu/rhythmsp/ASA/Contents.html> and at [http://www.mindspring.com/~ssshp/ssshp\\_cd/ss\\_home.htm](http://www.mindspring.com/~ssshp/ssshp_cd/ss_home.htm).

Finally, readers new to the topic might enjoy the interactive demos available at <http://www.research.att.com/projects/tts>.