Speech synthesis: System design and applications

by JARED BERNSTEIN
SRI International
Menlo Park, California

ABSTRACT

This paper introduces speech synthesis. Included are: a review of current synthesis technologies, an examination of the component algorithms and control structures needed for text-to-speech synthesis, and a discussion of current and future research topics.
INTRODUCTION

The purpose of this tutorial is to introduce speech synthesis. Included are: a review of current synthesis technologies, an examination of the component algorithms and control structures needed for text-to-speech synthesis, and a discussion of current and future research topics.

Current Synthesis Systems

Speech synthesis refers to two kinds of processes: (1) encoding, transmission or storage, and decoding of natural speech that might better be called "re-synthesis"; and (2) synthesis of speech by rule from linguistic input such as written text. One common method of encoding/decoding speech at low bit rates is Linear Predictive Coding (LPC) which is an efficient method of de-convolving the fundamental frequency from the spectral envelope based on the assumption that the speech spectrum can be adequately modeled as an all-pole linear filter that is excited by an impulse train. Fundamental work on LPC is presented in Markel and Gray, 1 Makhoul, 2 and Atal and Hanauer. 3 Encoding speech for later resynthesis is reviewed by Flanagan. 4

The focus of this tutorial is speech synthesis by rule and, in particular, synthesis from text.

A paper by Kaplan and Lerner 5 reviews commercial offerings in synthesis, and includes an overview of the component processes in the Prose 2000 product. Other system-level descriptions of text-to-speech systems can be found in Allen, 6 Allen, Hunnicutt and Klatt, 7 Hertz, 8 and Umeda. 9 An early Bell Laboratories systems is clearly presented in the text of patent 3,704,345. 10

Text to Speech Components

A text-to-speech system can be implemented as a set of processes connected in series. The first linguistically interesting process of a text-to-speech converter is the system for translating words (as written in standard spellings) into phonemic forms that describe pronunciations. This is usually called letter-to-sound (LTS) conversion. The classic sources that provide complete descriptions with complete rule sets are McIlroy, 11 Hunnicutt, 12 and Elowitz, Johnson, McHugh, and Shore. 13 A comparison of Hunnicutt's and Elowitz's rules is presented by Bernstein and Nessly. 14 A more recent approach to LTS is explained in Hertz. 8 Related morphological and lexical issues are covered in Allen, Umeda, and Church. 15

The next process in series is allophonics. Allophones are contextually conditioned variants of phonemes. Rules for selecting the appropriate allophonic form for a phoneme in a given context are important in synthesizing natural-sounding and intelligible speech. Rule sets are given in Klatt 16 and Allen 9 for synthesis; computational allophonics are discussed with recognition applications in mind by Oshika, Weeks, Nue, and Auerbach; Woods et al. 18 and Church. 19

After the text to be spoken is in allophonic form, a subsequent prosodic process is required to assign a rhythm and melody to the string of allophones. Conventionally, the rhythm of the sentence is taken to be the result of segment-level durations. The best duration rules in the literature probably are Klatt's. 16,20 A general problem with the rhythm of synthetic speech used in text-to-speech systems results because the systems do not "understand" what they are saying, and conventional punctuation is less than complete (following the "when in doubt, leave it out" school of commas). One might gain some speech quality by implementing some of the processes studied by Cooper and Paesia-Cooper. 21

The melody of English sentences is encoded in the fundamental frequency (f0) of the voiced portions and the way that the f0 patterns are aligned in time. Most text to speech systems use some version of a "hat and declination" f0 pattern 22 as, for instance, parameterized by Maeda. 23 This f0 rule provides a humdrum rendition of most neutral declarative sentences of the kind on which the hat and declination studies were based, but it becomes wearing in connected text. Other sources on f0 patterns are Cooper and Sorensen 24 Bernstein's review 25 of Cooper and Sorensen, and a recent collection edited by Cutler and Ladd. 26 The most promising recent development is Pierrehumbert's PhD. thesis, 27 but her 1981 paper 28 suggests a meaning-blind application of her theory which reduces it almost to a variation of Maeda. 29

The next process used in text-to-speech systems converting a linguistic transcription of allophones with durations and f0 values into a parametric description suitable to drive a signal synthesizer. An excellent and clear introduction to this aspect of synthesis is Chapter 6 of Flanagan. 4 The somewhat standard approach now is Klatt's 30 synthesis by rule logic that is used in DECTalk, the Prose 2000, and new products from Texas Instruments and IBM. Klatt presented his approach in more detail in Allen, Hunnicutt, and Klatt. 7 Some alternative approaches to phonetic synthesis are discussed in the "Phonetic Synthesis by Concatenation" section of this paper.

FAST LETTER-TO-SOUND CONVERSION

Introduction

The usual practice in letter-to-sound conversion involves checking substrings of letters and right and left contexts for each of several rules, then replacing the letter string with...
a phoneme string. Movement is from left to right through the word, with stress assignment and vowel reduction handled in a second pass through the word (usually from right to left). Unfortunately, the resulting software has been Byzantine, or the phonemic output is inaccurate, or both. Within this general framework, there also have been several attempts to automatically train letter-to-sound rules for optimal accuracy or size. These optimizing approaches have not resulted in high accuracy systems for several reasons but at least in part because they assumed an overly restricted algorithm structure.

Linguistic Content

This section describes elements of an algorithm and its associated data structures that allow fast and accurate letter-to-sound conversion in English. The description ignores affix stripping. The structure of the algorithm is based on three generalizations about the linguistic content of the rules of English spelling and phonology:

1. The rules of English phonology (and the published rulesets for letter-to-sound conversion) may have more different left environments but many more rules have right environments than have left environments.
2. Stress assignment (inside the stress-neutral suffixes; e.g., -ed and -ing) is much more simply handled from the right end of the word than from the left end.
3. There are orthographic cues for certain letter-sound correspondences that are not phonological. Such cues are either morphological or reflect changes in spelling conventions.

Design Consequences

Taking advantage of the historical and linguistic nature of English spelling, a system can perform high accuracy letter-to-sound conversion, stress assignment, and vowel reduction in one right to left pass, starting inside the stress-neutral suffixes (e.g., -ing or -ly). Thus, a right-environment semaphore can be set by the operation of the previous rule and checked with a single instruction. Furthermore, stress assignment can be accomplished by using three bits of the semaphore and a pointer to the last vowel phoneme, if any. The principal advantage of this approach is increased speed. The increased speed results because the largest portion of the processing time in most implementations of context-sensitive rules is spent context matching (some 80 percent or more of which is trivialized in this approach). The rest of the summary describes the rule structure and semaphore flags used and gives an example of how they work.

Details

A rule example is:
(e.g., for (ian) as in "Armenian") :
Right environment semaphore:
(Boundary)
Letter string:
IAN
Left environment:
(or (N (sequence TH) (sequence PH)))
Phoneme string:
EaN
Semaphore Set:
(Tense Vowel & I-Short)

This example means check for the boundary bit in the semaphore. If set, check the letter string, and if it matches, check the left environment which could be N or TH or PH. If right, replace the “ian” with /EaN/. Then, the phonological flags (like voiced, vowel, high) are set from the /E/; and the tense vowel and I-short flags are set out of the rule. The other consequence of this rule would be to increment the syllable count by two.

The semaphore has three kinds of flags: (1) logical bits, like negative and disjunctive, that control the interpretation of the remaining flags; (2) the usual phonological flags such as voiced, nasal, stop, sonorant, and labial, that are set based on the leftmost phoneme inserted; and (3) morphological/orthographic flags such as soft, palatalize, irreducible, and latin, that are explicitly carried in the rule.

PHONETIC SYNTHESIS BY CONCATENATION

Designs for Phoneme-input Unlimited LPC Synthesis

Phonemic synthesis is the transformation of a transcribed pronunciation, like that found in a dictionary, into a speech signal. Sivertsen'9 is the classic reference on inventories for synthetic speech by concatenation. Units that have been tried with known results include phonemes, allophones, diphones, syllables, demisyllables, and words. Any of these units may be the internal unit in a phoneme-input synthesizer because the map from phonemic input into any of these units is straightforward. The issue is: Which unit can give the highest quality at a reasonable memory size? In the following sections we describe allophones, diphones, and demisyllables with reference to synthesis.

Allophones

Allophones are contextually determined variants of phonemes; so they are the same "size" as phonemes except there are more of them. Though linguists rarely identify more than 60 allophones in English, for reasonable synthesis one might need as many as 300 vowel allophones and 100 consonant allophones (assuming consonant voicing by rule.) For instance, a system might need a "labial-velar /e/" as in "beg", and also use it in any of the contexts with {p,b,f,v,m} on the left and {k,g,ng} on the right. A description of Texas Instruments' allophonic synthesizer can be found in Electronic Design, June 25, 1981. With 128 allophones, the resulting speech is poor. C. Harris’ early report10 on the possibility of splicing phoneme length units is discouraging. Some of the reasons for the failure of phoneme concatenation are discussed by Wang.24
**Diphones**

Diphones are stored lengths of speech that extend from near the target of one phoneme to near the target of the next. The diphone is an appropriate unit for synthesis because coarticulation is mainly restricted to the immediate phonemic context. In speech, the path between phoneme targets often is non-linear and even non-monotonic within any usual acoustic parameter space (e.g., track the formants or LPC coefficients in the word "joy"). Thus, the primary advantage of diphones over allophones is that they include exactly the transition from one target to the next. The first diphone system was described by Dixon and Maxey; more recent diphone work has been reported by Olive and by Schwartz, Klovdstad, Makhoul, Klatt, and Zue. Although there are only about 40 phonemes in English and thus, ideally, about 1600 diphones, Schwartz suggests that about 2000 diphones would be needed for high quality diphone synthesis because some diphones are not really context-free, and the vowel dipthongs in English should be treated as pseudo-diphones.

**Demisyllables**

Demisyllables are initial or final portions of syllables that can be concatenated to form syllable sequences. Syllabic synthesis should be a very natural way to synthesize by concatenation, because (it is claimed) allophonic and coarticulatory variations rarely cross syllable boundaries. A method for cutting and joining demisyllables for synthesis by concatenation has been outlined by Fujimura, Macchi, and Lovins. They estimate between 1000 and 1200 demisyllables would be needed for a high quality unlimited synthesis system.

**Hybrid**

A hybrid concatenation system may be necessary to fix apparent problems with any of the three concatenation methods. In particular, LPC-based concatenative synthesis methods have trouble with sequences like (vowel) [l,r,y,w] (vowel). These "vowel-medial semi-vowels" can be handled by what Sivertsen calls "syllable dyads." to synthesize these sounds by concatenation, about 300 vowel-medial semi-vowels would have to be added to any of the inventories. A system that actually reaches production may need to include several types of units reflecting solutions to various detailed problems encountered in development.

**System Requirements for These Designs**

Assuming 42 bits per frame, an average speech rate of 150 words per minute, and a microprocessor that is fast enough to interpolate every other 20 msec frame, we can calculate a nominal memory size for each of the concatenation methods outlined. It probably is best to generate prosodic information by rule within the synthesizer system rather than try to store and adjust pitch, amplitudes, and gains as appropriate to current context. The “overhead” code (including the map from phonemes to internal units and the prosodics) would be less than 20k bytes.

1. **Allophones.** Full length vowels average 180 msec and consonants average 80 msec. Sampling every other 20 msec frame yields 4.5 frames per vowel and 2 frames per consonant. 300 vowels and 100 consonants, therefore, can be stored as 1550 frames at 6 bytes/frame for an allophone table of 10k bytes.
2. **Diphones.** Inter-phoneme transitions average about 100 msec. Sampling every other 20 msec frame of 2000 diphones that average 100 msec in length requires 5000 frames to be stored in 30k bytes.
3. **Demisyllables.** The average length of a demisyllable is 260 msec. 1200 demisyllables of 6.5 frames each would require storing 7800 6-byte frames in 48k bytes.

Reasonable quality synthetic speech should be possible with a total memory size between 40 kbytes and 70 kbytes, an LPC chip, and a microprocessor. Methods such as vector quantization for compressing LPC data could reduce the nominal table sizes given here by 10 percent to 40 percent.

**VOICE INSTRUMENTATION**

One could use vocal cues to direct user attention and to code the source and urgency of information within a voice interactive command/control environment. Voice output has several particular advantages in a user interface, especially when integrated with a voice recognition capability. A voice message reaches users regardless of their visual orientation and, like a flashing display or a red warning light, it can notify users that the system designers believe some aspect of the current situation requires user attention.

Also, voice response or verification are needed to maintain the advantages of voice input during “hands-busy eyes-busy” operation. Note, however, that just as an all-red instrument panel would decrease the advantage of bright red warning lights, routine or needless use of voice output could nullify its real advantages—especially if the same or a similar voice recites all the messages in a similar way. Therefore, there is a need to think carefully and design voice into complex displays in ways that use the distinctive communicative value of voice to best advantage.

In contrast to a written message, a spoken message carries several kinds of indexical information (for instance, the gender, size, and age of the speaker) as well as paralinguistic signals that express the speaker's attitude toward the message (for instance, the message is routine, or surprising, or urgent). By understanding and modeling the indexical properties of speech, we can develop separate vocal identities for different information sources. Through control of the paralinguistic aspects of a synthetic speech signal, we can use conventional modes of speaking to command a user's attention or reinforce the intent of the message by speaking it in a manner consistent with its content.

For example, differences in speech might be observed as message content changes from routine (e.g., "Fuel level at 80
From the collection of the Computer History Museum (www.computerhistory.org)

From the research literature we can guess that both voice pitch and amplitude may increase and that voice pitch may become more dynamic. Furthermore, the timing of the message and its enunciation may change, although these changes are less well understood. One course of action is to identify and test hypotheses about these changes by studying human speech; then decide the correct level at which to implement these changes in the message-generation subsystem of the command/control system.

The currently available text-to-speech devices (in particular, the Prose 2000 and DECTalk) are limited in the range of control that the host processor has over the indexical and paralinguistic properties of the synthetic speech. The Prose 2000 allows considerable control of speech parameters related to paralinguistic meaning, but supports a limited range of voice identities. DECTalk, in contrast, features six different voice identities, although host control of paralinguistic aspects is limited. Neither device offers the host a full set of appropriate controls for selecting voices and for encoding apparent attitude into the speech signal.

Steps needed for incorporating multiple voices into the command/control environment would involve: (1) understanding the acoustic-phonetic bases of indexical and paralinguistic information, (2) formulating that information in a way consistent with the message-generation logic of the command/control system, and (3) adapting (i.e. simplifying) the voice output control specification to the limitations of the available synthesis devices for demonstration and user evaluation, or (4) attempting to implement a special-purpose synthesis system to support the full range of indexical and paralinguistic cues identified in (1).

REFERENCES