Interacting With Computers by Voice: Automatic Speech Recognition and Synthesis

DOUGLAS O'SHAUGHNESSY, SENIOR MEMBER, IEEE

Invited Paper

This paper examines how people communicate with computers using speech. Automatic speech recognition (ASR) transforms speech into text, while automatic speech synthesis [or text-to-speech (TTS)] performs the reverse task. ASR has largely developed based on speech coding theory, while simulating certain spectral analyses performed by the ear. Typically, a Fourier transform is employed, but following the auditory Bark scale and simplifying the spectral representation with a decorrelation into cepstral coefficients. Current ASR provides good accuracy and performance on limited practical tasks, but exploits only the most rudimentary knowledge about human production and perception phenomena. The popular mathematical model called the hidden Markov model (HMM) is examined; first-order HMMs are efficient but ignore long-range correlations in actual speech. Common language models use a time window of three successive words in their syntactic–semantic analysis.

Speech synthesis is the automatic generation of a speech waveform, typically from an input text. As with ASR, TTS starts from a database of information previously established by analysis of much training data, both speech and text. Previously analyzed speech is stored in small units in the database, for concatenation in the proper sequence at runtime. TTS systems first perform text processing, including "letter-to-sound" conversion, to generate the phonetic transcription. Intonation must be properly specified to approximate the naturalness of human speech. Modern synthesizers using large databases of stored spectral patterns or waveforms output highly intelligible synthetic speech, but naturalness remains to be improved.

Keywords*—*Continuous speech recognition, distance measures, hidden Markov models (HMMs), human–computer dialogues, language models (LMs), linear predictive coding (LPC), spectral analysis, speech synthesis, text-to-speech (TTS).

I. Introduction

People interact with their environment in many ways. We examine the means they use to communicate with computer-based machines. Transfer of information between human and machine is normally accomplished via one's senses. As humans, we receive information through many modalities: sight, audition, smell, and touch. To communicate with our environment, we send out signals or information visually, auditorily, and through gestures. The primary means of communication are visual and auditory. Human–computer interactions often use a mouse and keyboard as machine input, and a computer screen or printer as output. Speech, however, has always had a high priority in human communication, developed long before writing. In terms of efficiency of communication bandwidth, speech pales before images in any quantitative measure; i.e., one can read text and understand images much more quickly on a two- dimensional (2-D) computer screen than when listening to a [one-dimensional (1-D)] speech signal. However, most people can speak more quickly than they can type, and are much more comfortable speaking than typing. When multimedia is available [181], combining modalities can enhance accuracy, e.g., a visual talking face (animation) is more pleasant and intelligible than just a synthetic voice [154]. Similarly, automatic speech recognition (ASR) is enhanced if an image of the speaker's face is available to aid the recognition [261].

This paper deals with human–computer interactions via speech [67], [108], [158], [216]. We address the technical issues in the communication of information between humans and computers using speech. This field has two distinct areas, ASR [121], [233], [239] and text-to-speech (TTS) synthesis [251], [266], to handle the separate, but related, tasks of transforming speech to text and text to speech, respectively. Sometimes, the tasks refer to concepts, rather than text, at one end of the process (e.g., speech understanding, rather than recognition, where the output is an action taken by the computer, based on an understanding of the speech, instead of a literal textual translation of the speech). However, most research has assumed that speech is at one end of the communication process and text is at the other. This is truer of synthesis, where the input to the process is virtually always a text, than of recognition, where speech is increasingly used to control machines or to initiate dialogue with a database [206], rather than have ASR merely act as a replacement for a keyboard. This paper will focus

Manuscript received November 7, 2002; revised March 13, 2003.

The author is with INRS—Telecommunications, Montreal, QC, Canada H5A 1K6 (e-mail: dougo@inrs-telecom.uquebec.ca).

Digital Object Identifier 10.1109/JPROC.2003.817117

on ASR and TTS, omitting other aspects of human–machine dialogue (e.g., dialogue management, natural language understanding, and generation) [206], [132].

At first glance, ASR appears to be the inverse of speech synthesis. Indeed, ASR simulates a person who is listening [100], and TTS simulates a person speaking. However, an algorithm designed for ASR cannot be run in reverse to accomplish speech synthesis. The human auditory system is very similar to that of all mammals and other animals; it is likely to have evolved much earlier than the speech organs. While many mammals have similar vocal tracts (VTs), humans alone have the brainpower to use their vocal organs to produce speech.

ASR is the more difficult and useful of the two speech-processing tasks treated here. To appreciate the asymmetry of the recognition and synthesis of speech, consider two specific subtasks for either people or machines to process speech or text: *segmentation* and *adaptation*. Tasks requiring intelligence (including language processing) often can be simplified by subdivision into smaller tasks. To interpret an utterance of any length beyond that of a trivial, single sound (e.g., "oh"), a human (or machine) usually tries to segment the speech signal into smaller units, following some acoustic and/or linguistic criteria. Natural units for segmentation are words and phrases, but other linguistic units have also been proposed, e.g., phonemes (the fundamental sounds of speech). Despite their popularity in linguistics, such units have complex relationships to aspects of the acoustic speech signal. As a result, reliable segmentation, by either human or machine, is difficult [31]. There are few, if any, clear acoustic correlates for boundaries between phonemes or words, in general.

Synthesis, on the other hand, receives text as input, and text is easily divided into letters and words. While the orthography of many languages (e.g., Arabic) involves cursive script whose letters (and often words) run together (as does English handwriting) and modern printing presses run some letters together, the task of segmentation of text is generally much simpler than segmenting speech.

The other subtask mentioned previously, adaptation, leads to even greater differences in complexity between recognition and synthesis of speech. When a synthetic voice is heard, human listeners must "adapt" to its "accent," i.e., adjust their perceptual mechanisms (subconsciously) to properly interpret the speech, much as one does for a nonnative speaker. If speech synthesis is of high quality, the adaptation task is quite easy for humans. Even poor synthetic speech can be tolerated for short periods, or if the listener is well motivated (e.g., a visually handicapped person accessing a book via a reading machine).

Consider the corresponding problem in ASR: different people speak, and the machine must adapt to each new speaker. Humans learn such adaptation automatically, as they learn to speak and understand speech. Even a child raised in isolation by a single parent still achieves automatic adaptation, as there is great variability in each person's speech: people never say the same thing twice exactly. Variability, both within individual speakers and across speakers (and languages), requires proper adaptation for interpretation, whether easily (by humans) or with much greater difficulty (by machines). The clearest application for ASR is faced with much variability: spontaneous speech, as occurs in dialogues with computer databases. ASR accuracy on read speech (e.g., the *Resource Management* and *Wall Street Journal* tasks, popular with researchers) is often considerably higher than for the more variable spontaneous speech (e.g., the Switchboard and CallHome tasks) [84], [125].

In the following, we describe current methods for ASR and TTS. ASR may be viewed as a simulation of human audition, and TTS as a simulation of human speech generation [37]. This follows a view designing artificial intelligence (AI) devices by mimicking human behavior. For most AI applications, it is highly instructive to understand the mechanisms humans use to accomplish their intelligent tasks. Properly understood, these could furnish a clear plan for machine simulation. However, intelligent activities are based in the brain, perhaps the least understood organ of the human body. In practice, AI designers simulate how better-understood peripheral organs function (e.g., eyes and ears), for example, in terms of their filtering actions. AI design that relies heavily on human activity modeling is known as cognitive–theoretic. A different approach, largely ignoring how humans behave, views the AI task as a simple input–output transformation (e.g., build a "black box" that replicates what humans do, looking only at results), e.g., the stochastic approach to ASR.

The most successful ASR and TTS systems are examples of the latter approaches, rather than of human simulation. The extreme complexity of the human nervous system (e.g., immense parallel processing and redundancy) has prevented a direct simulation of most AI tasks. While we comprehend much of what happens in the VT for speech production and what the outer, middle, and inner ear do for speech perception, our understanding of what the brain does (converting linguistic concepts into VT muscle commands, or interpreting the parallel firings from 30 000 hair cells in the inner ear) is very limited.

II. Vocabulary

For either ASR or TTS, it is theoretically possible to establish a lookup table as the main algorithm. For synthesis, this approach is feasible, but not in its simplest form (except for trivial applications): one or more speakers (perhaps chosen for "high-quality" voices) utter all pertinent texts for a given TTS application. For talking toys, ovens, or cars, which only require small vocabularies (e.g., "door is open"), this is indeed adequate. However, most TTS needs a much larger vocabulary; in general, there are no limits on TTS input. While the number of words in any language is finite (but growing, except "dead" languages, e.g., Latin), many languages have hundreds of thousands of words, which can be combined to form trillions of sentences. Since sentence length is not directly limited in many languages, "unlimited TTS" allows as input any text in a language; it is, thus, impossible to record utterances for all sentences. Issues of speaker fatigue occur for most applications where the vocabulary is not severely limited, at least if utterances are stored as full units (e.g.,

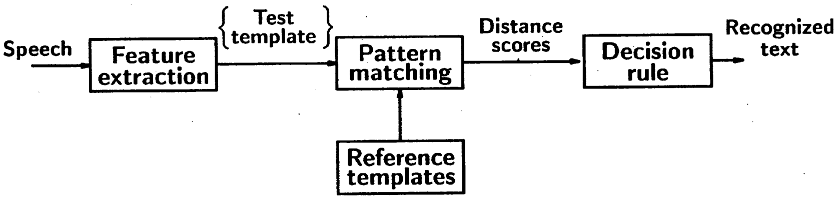

Fig. 1. Traditional pattern recognition model for speech recognition.

sentences); few speakers wish to record more than an hour of speech. Practical solutions usually require that the stored units be smaller than full sentences (e.g., TTS using units of words or smaller units). Such a compromise is typical of many made in achieving practical TTS and ASR, and often leads to greater efficiency, but reduces accuracy in ASR and quality in TTS.

A key aspect of any human–computer speech application is that of a *dictionary*: a list of all words in the vocabulary for that application, along with their pronunciations. As computers increase in power, there is less reason to restrain dictionary size, but it is necessarily finite. As such, the input to TTS may contain words not in the dictionary, and users speaking to an ASR system often do not obey system rules that limit speech to only permissible vocabulary words. [At the least, users find it difficult to avoid saying *uh* (a filled pause) or *you know*.]

TTS has a problem here only if the synthesizer has a limited vocabulary and the input derives from undisciplined users. This is usually overcome by allowing any input text and employing universal letter-to-phoneme rules to handle nondictionary words. Unlike TTS, ASR has no simple response for spoken words not in its dictionary: *out-of-vocabulary* (OOV) words. Designating portions of speech as “undecipherable” or outputting phonemes (rather than words) is usually unacceptable. If, each time an OOV word is uttered, ASR simply outputs the closest word from its vocabulary, it is in error. Even if ASR properly labels a section of speech as OOV, that is simply a cue to the user that he has used illegal speech (i.e., ASR rejects the utterance). If speech understanding is required, one could ignore OOV speech (if properly found) and hope that such speech is irrelevant (e.g., a filled pause, or other words of low utility).

In the future, the OOV problem will diminish as capacity to handle larger vocabularies increases; i.e., eventually all words may be in the vocabulary, although we would have to generalize the idea of “word” to include all mispronunciations, and develop ways to interpret all sounds. Currently, many ASR applications have very limited vocabularies. One often starts with an initial vocabulary of common words (e.g., articles, prepositions, pronouns) and assumed names and actions appropriate for a given task, and then trains on actual data for the application; e.g., starting from a 1000-word vocabulary and training on 10 000 words, we may find an OOV rate exceeding 10% [290]. As more training becomes available for a given task, the vocabulary is increased by including new words; OOV rates often fall in proportion to training size. A 20 000-word dictionary may require tens of millions of training words to achieve OOV rates below 1%.

III. Automatic Speech Recognition (ASR)

ASR is a task of pattern recognition. Of the major steps of ASR (normalization, parameterization, comparison, and decision) (see Fig. 1), the first two steps perform data reduction and constitute the ASR *front end* [120]. Each acoustic sequence of speech samples is classified as one of a set of linguistic categories or labels. If the utterance is a single word, we search for the best match among all words in the vocabulary. Consider a large acoustic space of N dimensions (i.e., a theoretical model for the speech signal), in which each utterance is a point in the space, using N features obtained from the speech signal via spectral analysis. The simplest feature representation would use the actual N speech signal samples, where the utterance has a total of N samples. This fares poorly because a direct set of time samples does not capture the essential information of the utterance. Speech is efficient for human-to-human communication, but must be processed for ASR. A direct comparison between speech waveforms, even for one word repeated by the same speaker, leads to a poor match with simple correlation measures, e.g., Euclidean distance, owing to the large variability (degrees of freedom) that speakers have. Small changes in pitch, duration, VT shape, and vocal dynamics occur even when speakers try to repeat a word exactly. Such changes are often imperceptible to listeners, but lead to large (objectively measured) distances. Instead, ASR first modifies (analyzes) the speech to better match the production and perception processes. This usually means spectral processing, as the ear is a good spectral analyzer. In particular, ASR tends to ignore phase in the speech signal, as the VT produces many (apparently random) variations of phase. Time-domain distance measures cannot ignore phase variations the way humans do.

A. Data Compression

First, the speech signal is compressed to a much smaller set of N spectral features, which not only makes ASR more efficient (less memory and computation), but also yields more accurate recognition. How big must N be? The number of features needed to characterize an utterance is roughly proportional to its duration, assuming that the information content of an utterance grows linearly in time. Consider first an ASR application for short words, and adopt a standard partitioning of speech into 10-ms units called “frames,” i.e., update the representation every 10 ms (i.e., frame rate of 100/s). This is common in many speech applications, and reflects a compromise between a slower rate (to reduce cost) and a higher one (to better model fast speech changes, e.g., stop bursts). A typical speaking rate is 12 phonemes/s—average phonemes lasting about 80 ms. Accounting for coarticulation between phonemes, sampling each about eight times is reasonable. While the representation is updated every 10 ms, spectral analysis for the frame usually extends over a larger time "window" (the set of successive speech samples analyzed by a spectral transform), where the samples are weighted (often nonuniformly, as with a Hamming window) during a longer interval, e.g., 20 to 30 ms, to better characterize the speech features.

Consider the size of an acoustic space to represent a short utterance. In basic ASR, N can easily exceed 500; e.g., 10 features for each of 50 frames in a brief 0.5-s utterance. If the features are well chosen, contiguous (ideally, convex) regions in the N-dimensional space would correspond to different words (a similar approach for phonemes might use $N = 10$ for a single frame). The ASR problem reduces to partitioning the space appropriately, so that each unknown input utterance (or frame) would be labeled with the text (or phoneme label) corresponding to the region in which its spectral representation occurred. The problem is simplified if the features are well chosen (i.e., have similar values for different renditions of the same phonetic segment, and have distinct values for segments that differ phonetically). As an example, using the first and second formants (and perhaps the third) as features could partition vowels along the "vowel triangle" in two (or three) dimensions [216].

Ideally, when speakers repeat the same speech unit, the resulting feature vector x should manifest little change. Thus, the feature values would be consistent and constitute small *clusters* or regions in the N-space. Also, speech units corresponding to differing linguistic entities should provide distinct measurements, i.e., widely separated points in the space (see Fig. 2). The best choice of features would show little intrasegment variance and large intersegment variances, and they would be independent of each other and with equal importance. In practice, however, ASR features share speech information (hence, are highly correlated) and some are much more relevant than others (hence, differing variances). If the similarity measuring procedure does not account for their interdependence and their unequal importance (as a result of cost-cutting measures), ASR accuracy is reduced.

This general approach is close to the methods currently used for ASR, but there are major problems: 1) how to determine the decision boundaries between regions; and 2) how to handle the obligatory larger values of N for longer utterances. The first of these problems is the more difficult. As vocabulary grows from a simple two-word (e.g., "yes–no") case to tens of thousands of words, the boundaries in acoustic space between words become increasingly complex. Indeed, the boundary just between "yes" and "no" is quite complicated, because of complex correlations across dimensions. Most ASR systems do not try to achieve a very low dimensionality, which could allow formation of simple space boundaries. Instead, using many correlated features and ignoring correlations between adjacent frames (as occurs often in ASR) hinders interpretation in acoustic space.

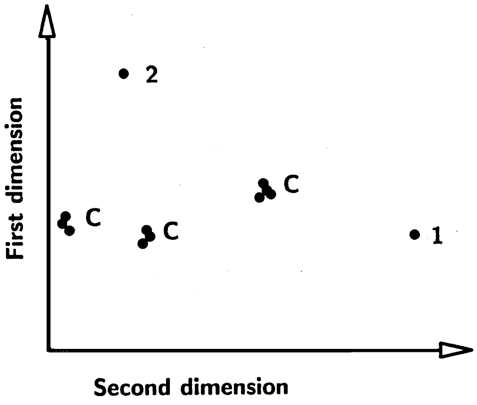

Fig. 2. Illustration of clustering of tokens in two dimensions. Three multitoken clusters C are shown, along with two unitoken clusters 1 and 2 (*outliers*).

To properly associate regions in an ASR acoustic space with their corresponding linguistic units (e.g., words) requires much training, involving the characterization of much speech. It is not feasible to have a lookup table for all possible utterances. A more feasible, but still daunting, task is to model the acoustic space well. Even for large-vocabulary applications (where the space is densely packed, with phonetically similar words near each other), we could expect clear (although complex) boundaries between the regions for different words if the ASR system is designed well.

A common practical problem is obtaining enough training data to characterize the regions in the acoustic space adequately (ASR models often have more than 10^7 parameters [26]). Given the infinite variability of speech and a finite amount of training data, a probabilistic approach is adopted: 1) averaging representations of repeated utterances of the same text; or 2) estimating a probability density function (pdf) for speech corresponding to each text based on such repetitions. In the first case, training yields, for each text, a representative average point in the N-dimensional space. During recognition, the space is searched for the closest match among these reference points for the point corresponding to the input utterance. For both training and testing, the same speech analysis (compression) is used. This first approach was typical around 1980 when dynamic time warping (DTW) dominated the ASR field [216].

In the second approach, the pdf for each reference model (e.g., one for each word or phoneme) is instead evaluated, using the point in acoustic space (from the analyzed input speech) to determine a likelihood that that model matches the speech. Thus, training creates a pdf to model each linguistic unit (or subunit, e.g., a frame of a phoneme), and each frame from the input (testing) speech is used to get a likelihood from the pdf. The combined (joint) probability of all frames from the test speech gives the likelihood that the corresponding sequence of linguistic units had been uttered in producing the test speech. The global joint likelihood that is highest among all possible model combinations specifies the estimated output text.

In both cases, we are faced with similar questions concerning the form of speech analysis, the choice of pdfs, the distance measure, etc. In particular, we should take into account the contributions of different dimensions in the acoustic space according to their utility in discriminating different linguistic units. We must exploit their varying utility by imposing different weights in any distance measure, or by weighting the pdf according to the standard deviations of the analysis parameters. Features that vary widely within a linguistic category, rather than across categories, should be deemphasized (e.g., weighted weakly in an ASR distance measure). On the other hand, dimensions that discriminate well (e.g., first and second resonances or "formants," for vowels, if such could be reliably obtained from speech) should be emphasized [103]. The importance of focusing attention (in the comparison or distance measure) on the most useful features increases as the number of dimensions N grows. As computers have become more powerful in recent years, the trend has been to use larger numbers of features in ASR, e.g., 40–50/frame. Major issues for ASR design are how to select an optimal speech spectral representation and the resulting reference memory and how to search for the best match or highest likelihood (in the set of reference models).

One goal of data reduction in ASR is the normalization of, or elimination of variation from, the speech signal. The many sources of variability in speech (randomness in human speech production, in signal transmission, or in noisy recording conditions) add undesired information to the speech signal; this extraneous data (often identified just as "noise") can theoretically be eliminated without loss of ASR accuracy. Reducing nonrelevant variation in speech moves us closer to the speech representations stored in an ASR system's reference models. When training and testing data are not matched well (e.g., a different speaker is used for a test, or the recording noise in the test is different from that observed in the training data), ASR using unnormalized data has decreased accuracy. These so-called mismatched conditions are the motivation of all normalization and adaptation algorithms, which try to create a closer match between the reference model and the test data.

B. ASR Origins

Both ASR and low-bit-rate coders have a common task in estimating the speech spectral envelope, and usually assume a standard model of human speech production: a filter representing the VT is excited by a source signal, either periodic (corresponding to a source at the vibrating vocal cords in the human larynx) or random (corresponding to a source at a narrow, nonvibrating constriction, usually in the upper third of the VT, near the mouth). In natural speech production, the quasi-periodic excitation corresponds to glottal puffs of air, at the fundamental frequency (abbreviated as $F0$) of the vocal cord vibration.

Many speech sounds have aperiodic excitation. Such noise sources (often called frication noise) are more diverse in nature than periodic excitation: typically, they can occur at the glottis (whispers and /h/ sounds), in the palatal region (/sh/ sounds), in the alveolar region (/s/), and at the teeth (/f, th/). [Some languages (but not English) allow noise excitation at other VT constriction locations, e.g., the velar, uvular, and pharyngeal consonants of German and Arabic, for example.] In general, all these noise speech sounds (at their source) have spectra that can be modeled as relatively flat. Thus, the spectra of fricative sounds (leaving a speaker's mouth) are largely modeled by the VT transfer function (i.e., the Fourier transform of the impulse response of the VT, between the source and the mouth). The excitation of voiced speech (i.e., the glottal pulses), on the other hand, has a discrete or line spectrum, owing to the harmonics of the periodic sound source at the vocal cords; i.e., the source spectrum consists of the fundamental ($F0$), typically in the range of 70 to 140 Hz for an adult male, and all its multiples ($2F0, 3F0, \ldots$).

ASR simulates the inversion of the speech production process; in the latter, a linguistic message in the speaker's brain formulates muscle commands for the VT organs, resulting in a sequence of VT shapes, and eventually the acoustic speech output. Thus, it is reasonable to ask whether there might exist an acoustic-to-articulatory (inversion) mapping, in which speech would be directly converted into a set of VT area functions, from which corresponding phonemes could be identified. (A camera focused on the speaker's mouth, and augmented by dynamic X-rays, could theoretically obtain such VT area displays, but of course this is neither feasible nor legal.) Attempts to determine VT areas directly from speech alone have had very limited success [256], [281].

Speech analyzers that aim for low-bit-rate representations usually try to separate excitation from VT response. Fewer bits are needed when these two major aspects of speech production are modeled separately. Features for ASR concentrate solely, in virtually all cases, on VT filter estimation. Besides total energy or amplitude, other excitation details (dealing with timing, phase, and periodicity) are simply ignored. (Even most reported ASR for tone languages, such as Chinese, forgoes use of pitch.) The reason for such disregard of the excitation of the speech signal, which human listeners indeed exploit, is that the role of pitch in speech communication is quite complex, and its effects are distributed in time in very different ways than VT-shape information about phonemes. The latter is often suitably handled on a frame-by-frame basis, but $F0$ requires interpretation over much longer time ranges. The effects of coarticulation (important for ASR) require examination well beyond 10-ms frames, but suitable approximations for frame-based analysis have been established in ASR for VT-shape information (and, so far, this has not been achieved for $F0$).

C. Feature Extraction

To convert speech to text requires an immense compression of data. For lossless speech coders, typical bit rates exceed 100 kb/s (e.g., compact discs: $705.6\ \text{kb/s} = 16\ \text{b/sample} \times 44\,100\ \text{samples/s}$). Text can be represented by tens of bits per second: assuming 32 phonemes and a typical speaking rate of 12 phonemes/s, we have 60 b/s ($12 \times \log_2 32$). (This ignores the phoneme-to-letter mapping needed to convert phoneme sequences into the final text, which, of course, is a major problem for ASR). For ASR we must discard the "irrelevant" or redundant parts of speech to arrive at the "essence" of the underlying (text) message. The compression ratio is three to four orders of magnitude. This oversimplifies the problem, since much more than pruning is needed, but suggests the large degree of redundancy found in speech. This redundancy is partly due to inefficiencies of the human system producing speech from an underlying text, but is also largely to increase communication robustness. Since speech is often subject to complications (e.g., background and communication-system noise), it must be comprehensible in adverse environments. A theoretically optimal speech generator, i.e., producing an acoustic signal of some sort at, say, 60 b/s would fail at communication when faced with the typical distortions that human speech communication tolerates easily.

For data reduction, ASR usually converts speech into a spectral representation, consisting of acoustic features (or equivalent parameters in a speech model). This "compression by analysis" often follows basic methods developed for speech coders, yielding features such as for linear predictive coding (LPC) coefficients [177], [183], amplitudes of filter bank outputs, or cepstra. Based on the early speech coding method of a differential waveform coder, LPC was used for ASR in the 1970s. The LPC predictor estimates each speech sample as a linear, weighted combination of P prior samples

$$\hat{s}(n) = \sum_{k=1}^{P} a_k s(n-k).$$

A smooth speech spectral envelope is directly modeled

$$\hat{S}(\omega) = 1/\left(1 - \sum_{k=1}^{P} a_k \exp(-j\omega k)\right)$$

as P poles simulate the formants. The automatically determined LPC coefficients a_k served as ASR features.

A further data reduction step toward high-level acoustic features is rarely done for coders, and few ASR systems employ them. Thus, we distinguish low-level and high-level acoustic features: the former are spectral coefficients directly obtained by a mathematical transformation, while the latter require nonlinear and coarse quantization, and risk labeling errors. (Low-level) acoustic features occupy more space or bandwidth (owing to less data reduction), but are free of decision errors. High-level feature estimation is more complex and less reliable, but has the potential of more relevant speech representations and more accurate ASR results.

We classify high-level features as either acoustic or phonological, depending on the degree of additional data reduction. *Acoustic features* (e.g., formants, $F0$) [81], [186], [219] represent an intermediate step between low-level and phonological features; they require an interpretation that is not always straightforward. *Phonological features*, on the other hand, have a very limited discrete range, assigning speech sounds to linguistic categories, e.g., "voiced" or "fricative"; they represent a more significant data reduction toward a phonemic decision than do acoustic features. In either case, high-level features can represent speech information much more compactly than can low-level features and, therefore, are potentially more efficient for ASR. They are used in expert-system approaches to ASR [81], but not in recent stochastic ASR [68], [163], [245], [289]. For low-level features, analysis "errors" derive directly from the precision of the model chosen (e.g., quantization error). The accuracy of high-level feature evaluation is much less predictable; e.g., relying on formant tracking for ASR leads to serious errors [283].

1) Desirable Properties for ASR Features: For low-rate speech coders (e.g., LPC), VT excitation in an analysis frame is typically represented by voicing, amplitude, and an $F0$ estimate. These excitation parameters are usually ignored (and, therefore, not computed) in ASR, on the assumption that sufficient information for ASR can be found in the spectral envelope. The voicing and $F0$ features, which require extra calculation beyond basic LPC or discrete Fourier transform (DFT) analysis, are subject to error (owing to the need for nonlinear decision rules) and are often difficult to interpret for the phonetic decisions needed in ASR. Amplitude and $F0$ tend to be more influenced by higher level linguistic phenomena (e.g., syntax and semantic structure) than by phonemics (e.g., $F0$ is only weakly related to phonemic identity). While all such linguistic information is relevant for ASR, most ASR systems concentrate their acoustic analysis toward phonemic decisions and leave higher level phenomena to the language model (LM) portion of the ASR system.

Spectral phase is rarely used for ASR because of its relative paucity of phonemic information. Phase is important for naturalness in coders, and often occupies the major portion of coding bandwidth for medium-rate coders [e.g., code-excited linear prediction (CELP) [277]]. However, reconstructed speech quality is not directly relevant for ASR. Listening to speech reconstructed using only the phase (i.e., artificially setting the spectral envelope to remain flat) allows much intelligibility, which indicates that relevant information is present in the phase. Nonetheless, it is commonly assumed that any "residual" information in the phase is redundant and can be ignored.

Thus, we turn to the spectral envelope, which provides the primary features for ASR. Most recognizers compute approximately 8 to 14 features for each speech frame. These are derived from a Fourier transform, an LPC analysis, or a bank of bandpass filters [45]. Recognizers use as few as five or as many as 80 features/frame [16]. The most common ASR features are the *mel-frequency cepstral coefficients* (MFCCs), but LPC coefficients, line-spectral frequencies (LSFs), energies from a channel vocoder (see Fig. 3), reduced forms of DFT, and zero-crossing rates in bandpass channels are other examples of feature sets used for ASR. They all attempt to capture, in about ten features, enough spectral information to identify spoken phonemes.

What information are we trying to represent by these spectral features? In other words, what is the most efficient parameterization of the VT response for ASR? It is well known in human speech production that a small number of low-frequency resonances, modeled via their center frequencies (and of less importance, their bandwidths), characterize well

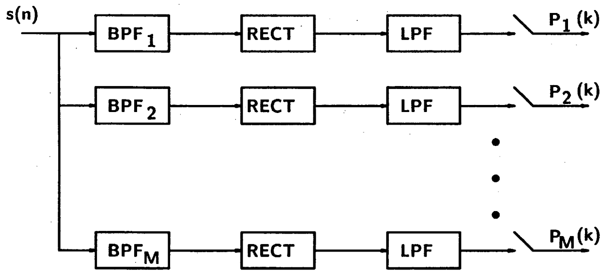

Fig. 3. Filter bank (channel vocoder) model for parameterization. M bandpass filters BPF$_i$ ($i = 1, 2, \ldots, M$) cover the input speech bandwidth range. Their outputs are usually converted to slowly varying estimates of channel energies via a nonlinear distortion (e.g., rectification), lowpass filtering, and decimation.

the spectral envelope for sonorant phonemes, which dominate most of speech. The other sounds (obstruents) can be well modeled by even fewer parameters, as their spectral envelope can often be approximated by two poles and a single zero. Thus, all speech sounds appear to require no more than five to eight real numbers (e.g., center frequencies for three to four resonances, and some of their bandwidths) to model the short-time speech spectral envelope [269]. Since formants are difficult to estimate precisely under all conditions, most ASR feature sets are more extensive than a mere 5 coefficients: 8 to 14 features are generally considered sufficient and efficient as a static set.

Formants exist on average every 1000 Hz for adult men (the spacing is about 15% higher for women, due to their shorter VTs). Thus, one might assume that the need for ASR parameters would increase with greater speech bandwidth. However, above 1 kHz, speech energy for sonorants falls off at about −6 dB/octave; thus, formants above 4 kHz (i.e., the fifth and higher formants) have very little energy. In addition, they appear to vary little with VT shape and, hence, are of little use for ASR. Thus, sonorant sounds appear to need little characterization of their spectra above $F4$ (the fourth resonance). Spectral detail at such high frequencies nonetheless contains phonemic information (e.g.,significant energy at high frequencies strongly suggests that the speech contains frication noise, and its spectral distribution helps identify the place of articulation). However, obstruent spectra can be very coarsely modeled; thus, there is no need for additional parameters for sounds with high-frequency energy. As a result, the set of ASR features need not change for applications at different speech bandwidths.

Many ASR applications concern speech over narrow channels, e.g., normal telephone links that heavily attenuate energy at frequencies outside the range of about 300–3300 Hz. When available to ASR systems, higher frequencies (up to perhaps 7 kHz) can yield useful information for ASR, primarily for consonants. Taking additional bandwidth into consideration, however, must be done properly, because merely extending the bandwidth range of ASR analysis treats all frequencies equally (whereas lower frequencies are more important for sound discrimination). As a result, simple bandwidth extension can actually deteriorate ASR performance [148].

2) Cepstrum: In the early days of ASR, simple filter banks or DFTs furnished the needed features. Later (from the early 1970s to the mid-1980s), LPC coefficients and their transformations (e.g., reflection coefficients, LSFs), which model the spectral envelope well and are still prevalent in speech coding, were widely used in ASR as well [223]. ASR tended to use the basic LPC coefficients (a_k), which have been rarely used in coding and synthesis applications, owing to quantization and stability issues (which are of much less concern to ASR).

Since the mid-1980s, the most widely used parameters for ASR have been the MFCCs. The cepstrum $c(n)$ is the inverse Fourier transform (IFT) of the log-amplitude speech spectrum $|S(\omega)|$. Since speech is usually modeled as an excitation $e(n)$ convolved with the VT response $h(n)$, the cepstrum effectively “deconvolves” the two

$$s(n) = e(n) * h(n) \tag{1}$$

$$|S(\omega)| = |E(\omega)||H(\omega)| \tag{2}$$

$$\log|S(\omega)| = \log|E(\omega)| + \log|H(\omega)| \tag{3}$$

$$c(n) = \hat{s}(n) = \hat{e}(n) + \hat{h}(n) \tag{4}$$

where the circumflex notation indicates the inverse transform of the log-amplitude. Such a deconvolution is desired because the spectral envelope, or more exactly the cepstral envelope, corresponds much more closely to the VT shape information needed for ASR than does the harmonic-laden original spectrum. The cepstrum separates approximately into a low-time component $\hat{h}(n)$ corresponding to the envelope and a high-time component $\hat{e}(n)$ for the harmonic structure, because the logarithm operation preserves the peaky nature of $E(\omega)$ (for voiced speech); thus, $\hat{e}(n)$ has major components at multiples of $F0$. The $\log|H(\omega)|$ operation, on the other hand, smooths the spectral envelope, leading to a more compressed $\hat{h}(n)$ VT component. It is this low-time component, the initial set of 8 to 16 values of $c(n)$, that is used as the set of ASR coefficients.

A major advantage of the cepstrum over an LPC representation is the option to modify or deform the frequency axis, to follow the auditory mel scale, as part of the cepstral calculation [44]. To minimize computation, the DFT is usually calculated via a fast Fourier transform (FFT) algorithm and, thus, has equal spacing of frequency samples. Similarly, LPC generally applies a minimum mean-square error criterion in which linear time and frequency scales are essential to simplify computation. Such linear scales do not correspond well to speech production or perception, which treat low frequencies more importantly than higher ones. In particular, a roughly logarithmic scale seems appropriate for many physical phenomena.

Thus, the MFCCs alter the basic procedure described previously, replacing $\log|H(\omega)|$ with a modified version, prior to the IFT. A small set of M filters (usually triangular in shape) first multiplies the log-amplitude spectrum to yield M output energies, which in turn furnish the input to the IFT step. The purpose of these filters is to deform frequency to follow the spatial relationship of hair cell distribution in the cochlea of the inner ear. In practice, the "mel" approximation uses a linear scaling (i.e., equally spaced filters) up to 1 kHz, and thereafter a logarithmic spacing (i.e., increasingly wider filters as frequency grows above 1 kHz). This spacing follows approximately that of the 24 "critical bands" of hearing research, although 20 triangular filters are common in ASR use. The actual shape of the weighting filters has little effect on ASR performance.

The log spectrum $\log|H(\omega)|$ may be obtained as part of an LPC analysis, or more often directly from an FFT. The mel-scale frequency warping improves performance versus ASR that linearly weights contributions from all frequencies [33], [51], [169], and reflects the nonuniform distribution in frequency of phonemic information. Auditory research has identified many other facets of human hearing (besides the mel scale), and ASR researchers have examined many ways to include such information into the ASR process. However, most current ASR does not exploit the nonlinear dynamics of the inner ear that affects human speech perception. Such efforts have demonstrated only limited success in improving recognition accuracy [80], [292], [83], although interest continues in exploring auditory models for ASR [268], [6], [253], e.g., *perceptual linear prediction* (PLP) [130].

3) Preemphasis: While the mel scale is an aspect of ASR that derives from speech perception, *preemphasis* is an analysis technique that reflects speech production. This option in the analysis phase raises the (generally weaker) amplitude of high frequencies prior to spectral analysis, so that frequencies are weighted more equally during analysis. Without preemphasis, LPC would model lower frequencies more precisely than higher formants. A simple operation replacing each speech sample $s(n)$ with its differenced version $s(n) - as(n-1)$ (where $a \approx 0.9$) accomplishes the desired spectral equalization (tilting the spectrum upward with increasing frequency). This procedure is similar to Dolby sound processing and to preemphasis in FM, except that for ASR there is no need for a corresponding deemphasis at the receiver, since we do not reconstruct the original speech signal (i.e., ASR performs lossy compression on the speech input).

Preemphasis is only helpful for voiced speech, which is subject to the -6 dB/octave falloff of sonorant phonemes (owing to the excitation of relatively smooth glottal puffs). Applying it universally to all speech (since unvoiced speech cannot be reliably identified *a priori*) unnecessarily boosts the noise of fricatives, which dominates such sounds at high frequencies. However, any negative effects on modeling unvoiced speech are outweighed by improved representations for the more prevalent voiced speech.

The use in ASR of preemphasis has a parallel in the acoustic emphasis, by the ear, of sound in the 2- to 4-kHz range. The outer ear, an acoustic tube of about 2.7 cm in length, has a natural resonance near 3 kHz, which boosts the weaker speech energy normally found in this frequency range that is critical for much speech perception [50]; e.g., most of the spectral detail distinguishing place of articulation for stop consonants occurs above 1 kHz.

4) Representing Speech Dynamics in a Spectral Vector: Usually ASR assumes an independent treatment of successive speech frames [227], despite clear evidence from human speech production that strong correlations exist across much longer time spans [17]. Instead, each frame of the input speech signal is analyzed separately, to greatly simplify computation. Separate *vectors* of frame-based spectral features are preferred, rather than a larger matrix of parameters (indexed on both time and frequency, whose column vectors would correspond to successive frames). The latter could better model timing correlations than frame-based methods, but are significantly more complex.

Because timing correlations are nonetheless important, the spectral vector for each speech frame often consists of the basic N *static* features (i.e., from an analysis of each frame) and dynamic (*delta*) features (i.e., the differences between corresponding static feature values over successive frames) [233]. Thus, if $f(i,n)$ represents the nth feature for the ith frame, the actual vector for frame i has $2N$ features

$$f(i,1), f(i,2), \ldots f(i,N), f(i,1) - f(i-1,1),\\ f(i,2) - f(i-1,2), \ldots, f(i,N) - f(i-1,N).$$

The second set of N features is included in each frame's vector to accommodate some dynamics, as well as the "static" information of the status of the VT during the actual frame. The static features reflect the position of the VT; the delta coefficients concern its velocity. Further, ASR may include acceleration (*delta–delta*) features, i.e., the differences of the delta features. Thus, a typical ASR frame might have 13 cepstral coefficients, 13 delta values, and 13 delta–delta values, forming a 39-dimensional vector.

The difference need not involve immediately adjacent frames, which restricts the temporal view to only two to three proximate frames. The delta features may be taken over a wider range, e.g., replacing each $f(i,n) - f(i-1,n)$ with $f(i,n) - f(i-k,n)$ [or with $f(i+k,n) - f(i-k,n)$, for symmetry] in the example vector described previously (where $k = 2,3$, typically). This reflects the notion that important aspects of coarticulation extend well beyond 20–30 ms. The common use of $f(i,n) - f(i-1,n)$ as the delta features is the simplest way to include timing variations in a frame-based ASR system.

Another way involves *linear discriminant analysis* (LDA), a more costly method to greatly reduce the number of dimensions in a pattern representation. L successive frames are first concatenated to form a high-dimension speech vector (e.g., $9 \times 13 = 117$ [254] for $L = 9$ and 13 MFCCs). A linear transformation (similar to principal components analysis; see discussion later) identifies the most relevant new dimensions.

Much of the information relevant for ASR is readily visible in a wideband spectrogram representation of the speech signal, although human "spectrogram reading" seems to be more art than science [289]. Some recognition approaches try to capture more of the correlations seen in such three-dimensional displays (energy versus time versus frequency) than is possible to exploit with standard frame-based hidden Markov models (HMMs), e.g., using a 2-D cepstrum rather than the usual 1-D version [165] or a "modulation spectrogram" [141].

5) Frame Rates: Whichever set of features represents each speech frame, the issue arises of how often to update, or equivalently what an appropriate duration is for a frame. There are two separate but related factors: frame size and update interval. The window length or frame size N is the number of speech samples examined for each frame. The update interval M is the time (in samples) between ASR updates, and reflects how often ASR evaluates a portion of speech. The simplest relationship would be $N = M$, i.e., successive sets of N speech samples with no overlap among adjacent frames. In general, however, most systems update every 10–20 ms, while window lengths are 25–30 ms; thus, frames overlap. Most analysis windows taper at their edges, thus heavily attenuating use of edge samples; if no overlap were to occur, analysis would overlook (or heavily underweight) arbitrary sets of speech samples.

The actual values chosen for both frames and updates are a compromise between cost (less frequent updates means less computation) and performance (overly long frames would smear out the dynamic effects of relevant VT transitions). As for the update rate, we could follow the Nyquist theory that requires samples at least at twice the highest frequency in the signal (i.e., each dynamic feature sequence). Typical rates for each sequence are 40 to 100 samples/s, which correspond to frequency cutoffs of 20 to 50 Hz. These frequencies should be interpreted as reflecting the velocities of VT movements relevant to phone identification. Higher rates are appropriate to better analyze speech transients (e.g., stop releases), where phonetic events are sometimes as brief as 5 to 10 ms. If the frame size is too small (e.g., under 20 ms), which could easily correspond to updates more often than 100/s, we risk increasingly important pitch interference in (VT) feature sequences. Traditional window lengths of 25 to 30 ms ensure having at least two pitch periods when analyzing voiced speech. With much shorter frames, the speech being analyzed in each frame can vary greatly from frame to frame.

A final issue here is the number of bits to allocate to represent each feature. Computer hardware concerns may dictate using 8 or 16 b/feature, but it appears that 3 to 4 b/feature may well be adequate for speech recognition [291], assuming reasonably chosen features.

D. Pattern Matching

A major concern of ASR design is establishing good reference patterns, models, or "templates," based on feature representations of the test (unknown) speech and reference (prior training) speech. The evaluation or comparison of pairs of such patterns involves finding the best match in terms of a weighted distance (or correlation) between patterns, or deciding which reference model is most likely. The reference models are derived during a prior training phase. At recognition time, the representation of the test utterance T (a set of features derived from the test speech) is compared with some or all of the stored reference models R_i ($i = 1, \ldots N$ for N units—words or phonemes). The reference memory may be partitioned for an efficient search; e.g., if the speaker is known [e.g., in speaker-dependent (SD) ASR], the search can be limited to that person's stored models. If the speaker's gender is estimated from an analysis of some of the input speech, then only the appropriate gender-specific models are examined.

The model evaluation determines how similar T and R_i are, or how well R_i models T. Based on T and the trained models, the most likely R_i is chosen, i.e., the R_i best modeling or matching T, yielding a text output corresponding to that reference. However, if the best match still yields a relatively big distance, small correlation, or low likelihood, or if other R_i provide similar (but slightly worse) results, then some ASR systems postpone making a decision. They ask the speaker to repeat the utterance, and use combined results from multiple repetitions.

A database of reference models, each characterized by an N-dimensional feature vector, must be established during training, in which one or more speakers usually utter a controlled vocabulary. Under such *supervised* training, acoustic segments are labeled with phonetic "codes" corresponding to the known training texts. For segments corresponding to short speech frames, under our previous assumption, the codes would involve phonetic segments (e.g., phonemes). If we now generalize to longer speech segments (syllables, words, or even phrases), this requires analysis of successive speech frames (not a single frame, which is adequate for some phonemes). Nonetheless, we can still use the analogy of an N-dimensional acoustic space, if we expand N to include the necessary parameters over several frames, to represent the temporal variation occurring during the longer speech units.

This description follows the common models for ASR: we first looked at the simplest approach for the shortest meaningful units—the phoneme, using a single frame; now we move to the other most commonly used unit for ASR—the word. For word-based ASR, we could use word templates that are M-dimensional vectors (or $L \times N$ matrices), where $M = LN$ and the L vectors, each of dimension N, are calculated at uniformly spaced time intervals for each word in the ASR vocabulary. We further assume a fixed value of L for all words, to continue discussion of the simplest approach for word-based ASR. We see later that this is impractical for words of varying durations. It would require a mapping of all words onto a word model of "typical" duration (i.e., one with L frames). (The significant problems of time normalization and aligning subunits in such models are examined later.) In discussing ASR comparison measures here, assume a single M-dimensional vector adequately represents each spoken word. Later sections will also deal with the nonuniform distribution of speech information over time and frequency.

1) Frame-Based Distance or Similarity Measures: The similarity between two patterns is often expressed via a *distance* or *distortion* [90], which measures how close the patterns are in N-space. A distance measure can account for the unequal contributions of different features to ASR and their correlations within the N vector [260]. Today's ASR is based largely on statistical models, where the reference models store pdfs and similarity is judged in terms of the likelihood of the test pattern. The discussion in this section accommodates both deterministic distance matching methods and stochastic approaches. To handle multiframe segments (e.g., words), local distance measures for each frame are typically summed to yield a global (word) distance, or probabilities are multiplied to yield a joint likelihood (we initially assume that successive frames are independent of each other). The reference pattern (model) with the smallest distance (greatest probability) yields the ASR output.

Ideally, the ASR distortion measure [179], [203] should represent perceptual similarity between the test (word) and each of the reference models. Among aspects of the feature vector representing the speech spectral envelope, some should be emphasized (and less relevant features ignored). For example, a good ASR measure would focus on similarities in spectral peak positions while largely ignoring spectral tilt, deemphasizing the higher frequencies (e.g., the Bark scale), and ignoring low-amplitude components.

In ASR that uses templates (e.g., DTW systems), each unknown utterance is represented by an N-feature test template, which is matched against each pertinent reference template to find the one with the closest match. (For a word with L frames, this process repeats periodically L times, summing L local distances; a single match using templates of $N \times L$ dimensions is usually impractical.) The similarity of two templates is inversely proportional to the distance in N-space between the points corresponding to the templates. A distance measure $d(x, y)$ between templates x and y is called a *metric* if it satisfies three conditions: positive definiteness ($d \geq 0$, and $d = 0$ only when $x = y$), symmetry ($d(x,y) = d(y,x)$), and the triangle inequality ($d(x,z) \leq d(x,y) + d(y,z)$). For ASR, the two most important aspects of a distance measure are its subjective significance (objectively measuring perceptual similarity) and mathematical tractability. Metrics ensure the latter, but many common ASR distance measures are not metrics; those satisfying only the first condition given previously are called *distortion measures*.

The most common metric is the Euclidean distance (or *L2*-norm)

$$d_2(x,y) = \sqrt{(x-y)^{\mathrm{T}}(x-y)} = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \quad (5)$$

which is the usual interpretation of a distance between two points in N-space. In the family of distances called L_p norms

$$d_p(x,y) = \left(\sum_{i=1}^{N} |x_i - y_i|^p\right)^{1/p} \quad (6)$$

d_2 is the most popular, but d_1 (the *Chebyshev* or *city-block* distance) is simpler to compute. Despite decreased recognition performance, the latter may be used for low-cost ASR, since no multiplications are needed; the distances along each dimension are summed

$$D_C(x,y) = \sum_{n=1}^{N} |x_n - y_n|.$$

As p increases, d_p concentrates more on large distortions, which are more prevalent near spectral peaks; this is similar to LPC distance metrics. Such weighting is justified perceptually, as listeners focus on differences in the strongest formant peaks. Nonetheless, the L2-norm is most common for ASR, owing to its low cost.

A disadvantage of the Euclidean distance is its equal treatment of features x_i: more important features are not allowed greater weight. Thus, we have another common speech metric called the Mahalanobis or *covariance-weighted* distance

$$d(x,y) = \sqrt{(x-y)^{\mathrm{T}} W^{-1} (x-y)} \quad (7)$$

where W is a positive-definite matrix that allows varied weighting for each feature depending on its ability to separate different speech units in N-space. (The Euclidean distance is a degenerate case of the Mahalanobis distance, where W and W^{-1} equal the identity matrix I.) The elements of the main diagonal of W are the intrafeature variances, where small values indicate more useful ASR features. Less useful features have greater variation for different versions of the phonetic unit they are modeling and, thus, large matrix values lower their weighting when calculating the distance.

Distance calculation can be greatly simplified if the features can be made orthogonal, which then allows a diagonal W^{-1}. A *principal components* or Karhunen–Loève (K–L) transformation on the parameters (needed to achieve orthogonality) is usually too expensive computationally for ASR. However, recent computer advances have allowed large increases in ASR modeling complexity. As distance or likelihood computations often tend to dominate ASR computation, restricting models to a small set of parameters is a way to reduce cost. Part of the popularity of the cepstrum is that a long-term average K–L transform of speech approximates a discrete-cosine transform (DCT), and DCT is the K–L transform of a first-order Gauss–Markov process.

The issue of interparameter correlation (within a spectral vector representing a speech frame) is a significant problem for ASR. All the elements in full covariance matrices are needed to properly account for such correlation, and this leads to a large number of matrix weights to estimate. Most often, an ASR system must estimate many parameters but has a small amount of training data (hence, *undertraining*); thus, robust estimation is difficult. Great computational savings occur if diagonal covariance matrices are assumed, i.e., all parameters in a frame's spectral vector are assumed uncorrelated with each other—the parameter variances (along the main diagonal) are the only nonzero matrix elements. Though ASR parameters in a vector are indeed correlated (except for K–L), this assumption greatly simplifies computation of ASR distances or probabilities.

2) Stochastic Similarity Measures: The Mahalanobis distance is justified when forming distances where the dimensions are related and have varied value; it is also easily motivated by statistical decision theory. Following our image that each utterance of a word (or other phonetic unit) represents a point in N-space, then the many possible pronunciations of that word describe a multivariate pdf in N-space. Assume the ASR task must choose among equally likely words in a given vocabulary and that *maximum likelihood* (ML) is used as the decision criterion [16]. Then ASR should choose the word whose model pdf gives the highest probability when evaluated with the feature vector from the test word. The ML approach declares that this word model is most likely to match the test utterance.

The pdf models used in ASR are often coarse approximations of the true models, which could only be obtained by training on all (infinite) speech. In practice, the amounts available are often not even adequate to achieve good models. The most direct training method simply constructs a histogram from relevant training vectors. Assume that the domain axis for each parameter is quantized with L bins; e.g., divide its extent ($E =$ maximum value $-$ minimum value) by L, so that each bin has width E/L. If we have M training "tokens" (i.e., examples of each word) for the phonetic unit being modeled, then each bin, on average, has $\alpha = M/L$ values. For good histogram modeling, one would typically need $\alpha > 10$. Further, to allow reasonable discrimination among phonemes, for example, $L > 25$, which suggests that hundreds of tokens are needed in general. When dealing with a SD, word-based ASR system, for example, it is difficult to ask each speaker to repeat each word hundreds of times as training. There are several ways to ameliorate this training problem, but it suffices to note here that in general ASR suffers undertraining (hence, the ASR motto: "There is no data like more data").

Owing to the difficulty of estimating precise pdfs from little training, many ASR systems assume that each model's pdf can be reasonably characterized parametrically, i.e., a simple mathematical function specified by a small number of parameters. The most common such pdf for ASR is the Gaussian distribution, which is completely described by a mean vector μ and a covariance matrix W. (For a single dimension, we need only a mean value and a variance; for an N-dimensional spectral vector for each frame of speech, a multivariate, joint Gaussian pdf is needed.)

The Gaussian, or bell curve, is common in many stochastic systems. It is well justified by the Law of Large Numbers (a sum of K independent and identically distributed random variables tends to a bell curve as $K \rightarrow \infty$). Many actual physical measurements derive from such sums, and when modeling well-controlled phonetic conditions (e.g., a single speaker repeatedly uttering a single sound), a Gaussian curve is an excellent match to the true pdf. Unfortunately, training conditions are usually less controlled, e.g., a single model may be trained on many speakers in many phonetic environments, and the true pdf then deviates from a simple Gaussian model.

A simpler alternative, the Laplacian or symmetric exponential, is occasionally used [267]; it corresponds to the city-block distance (i.e., Gaussians follow the L2 norm or distance measure, which emphasizes the importance of large distance components, while Laplacians follow the L1 norm). As such, Laplacians are sometimes better at approximating the tails of speech pdfs.

The Gaussian pdf of an N-dimensional vector x for a phonetic unit (e.g., indexed among a vocabulary of M units by k, $1 \leq k \leq M$) is

$$P_k(x) = (2\pi)^{-N/2} |W_k|^{-1/2} \exp - \left[\frac{(x-\mu_k)^{\mathrm{T}} W_k^{-1} (x-\mu_k)}{2} \right] \tag{8}$$

where $|W_k|$ is the determinant of W_k and μ_k is the mean vector for unit k (e.g., word or phoneme). For complete specification, the Gaussian pdf for each word needs an estimate for the mean of each parameter (thus, N values) and the N^2 matrix elements. These latter elements are defined as $w_{i,j} = E(x_i x_j)$ (i.e., the i,jth element of an autocorrelation matrix W, where E signifies stochastic average or expectation). Since $E(x_i x_j) = E(x_j x_i)$, the upper right half of the matrix is the same as the lower left half (i.e., mirror image about the principal diagonal); thus, there are only $(N^2 + N)/2$ unique elements to estimate.

Many ASR systems model each state pdf as a sum or weighted *mixture* (combination) of several Gaussian pdfs, because the actual density (e.g., as verified by histograms of training data results) is much more complex than a simple Gaussian. While direct use of histograms (L values) is feasible instead of the parameters of multiple Gaussians, the latter have traditionally been retained, based on earlier successes with single Gaussian pdfs. To decrease computation, ASR sometimes uses the maximum of a set of Gaussian pdfs, rather than a weighted sum [267], or a subset of the pdfs [75]. As usual, each simplification often decreases accuracy, while reducing cost.

The most useful information component in each word's pdf is the mean vector μ_k. The W_k matrix gives secondary information, details about the interrelationships of the individual parameters (i.e., for $i \neq j$, $w_{i,j} \neq 0$ indicates a correlation between x_i and x_j) and their reliability (i.e., for good ASR parameters, the variances $w_{i,i}$ are small). Thus, the latter is used to judge how much weight to assign to each of the parameters in the former. It is difficult to obtain accurate estimates for W_k from small amounts of training data. When a full matrix is used, it may be split into two components, with one being a diagonal matrix, where the elements are "tied" or shared at different levels. Typically the full elements are extensively shared across many HMMs, requiring only a small increase in the number of parameters compared to the diagonal case [74].

Besides the availability of training data, their reliability can be an issue, especially when the speech is obtained via unsupervised training, or when the speech is poorly segmented or transcribed (human errors in deciding how to label speech sounds lead to degraded ASR models). A common problem is how to deal with "outliers," training data tokens that yield spectral vectors x that are far from others for the model of a given unit. Usually, all vectors for a model are averaged to yield means and variances, i.e., a linear combination is applied. With such standard averaging, each outlier can greatly affect the model statistics. Some ASR systems exclude such outliers, assuming that they represent procedural, recording, or labeling errors [10].

On the other hand, if outliers represent reliable measures, they can be exploited in discriminative training. LDA is a common method to improve the discrimination between classes (e.g., spoken digits) in a high-dimensional vector space [113]. This focuses recognition on the specific aspects of phonetic units that differ from each other, rather than a general ML approach, and is efficient in small-vocabulary ASR; e.g., basic ML estimation is often inadequate for confusable vocabularies (e.g., the alphabet [174], where the focus is best placed on the consonant in a spoken letter; identifying the vowel is less relevant, as three large classes (B,C,D,E,G,P,T,V,Z; A,H,J,K; F,L,M,N,S,X) all use high, front vowels).

ASR performance measures other than ML are usually more costly. Training systems to directly minimize error rates [*minimum classification error* (MCE)] instead has been often difficult [33], [40], [41], [128], [135], in part because we have no metric to measure the ordering of output hypotheses [271]. In *maximum mutual information estimation* [47], [204], for which *expectation–maximization* (E–M) algorithms exist (see discussion later), we seek to emphasize mutual information between recognition classes and features. *Deterministic annealing* is a method that minimizes a randomized MCE cost subject to a constraint of a gradually relaxing entropy [237], [221]. *Bayesian predictive classification* is another recent training method [115], [124].

E. Stochastic Automata for Speech Recognition

Almost all ASR uses some form of finite-state machine (FSM) to handle the great variability found in speech via stochastic modeling [15], [17]. For example, knowledge about syntactic and semantic constraints on allowable sequences of words may be efficiently coded in terms of an FSM whose states are the vocabulary words. Transitions between states are allowed only if the resulting string of words produces a legal sentence following the system's grammar. Such automata can be generalized to other ASR applications if the states represent acoustic segments or frames of speech data.

Automata in ASR employ a statistical, rather than rule-based, representation of acoustic information. Apply this first to isolated word recognition (IWR): model each word with a succession of phonetic states i (corresponding roughly to phonemes), linked by transitions specified by likelihoods a_{ij}. This probability of a phonetic segment j following segment i governs the transition between the states representing those two sounds. Consider *pass* as an example word, where states for /p/ closure (silence), /p/ burst, /pæ/ aspiration, /æ/, and /s/ might be chosen via coarse segmentation. To allow for the chance that the /p/ burst and/or aspiration may be missing (or overlooked in the analysis of some training samples), the a_{ij} may vary considerably; they typically correspond to the frequency of actual transitions in the training data.

For IWR, each input word utterance is evaluated by each word FSM, to find the FSM most likely to have generated the word. Instead of searching a DTW space for a path of minimal distance, each FSM is searched for the path maximizing the product of all transition probabilities between states corresponding to the test utterance. In practice, log likelihoods are used so that probability values may be summed, instead of multiplied, for faster computation.

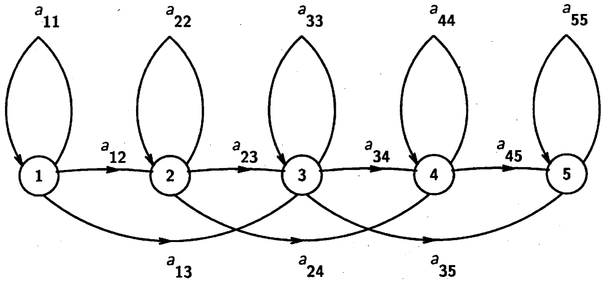

Fig. 4. Five-state Markov model with Bakis restrictions on transitions. With each state j is associated an observation vector b_j, whose elements indicate the likelihood of observing different speech spectra in that state and a set of transitions, each denoted by an arc with probability a_{ij} of arriving in that state from state i.

1) Hidden Markov Models: The most commonly used FSM for ASR is the *first-order Markov process* or *chain* [64], [110], [159], [182], [227], [242] (see Fig. 4). (Neural networks have been used as full ASR systems [194], but now they mostly contribute as components in ASR that uses HMMs as the main architecture.) The popularity of this method lies in its model simplicity, ease of training, and its recognition precision on certain practical ASR tasks. The Markov model imposes that the likelihood of being in a given state depends only on the immediately prior state (and not on earlier states); thus, the transition probabilities are conditional only on the immediate state, and past history is ignored. This assumption is key to the simplification of the model (and related computational needs), but represents inaccurate modeling, since both speech production and perception are conditional on much longer time ranges; human short-term memory lasts several seconds and speech planning exploits similar time scales, whereas the Markov ASR model views only the past speech frame (extended by delta–delta coefficients, to a few frames past history).

States are informally associated with phonetic events in the speech signal, in the sense that each model usually has a number of states that approximates the number of distinct events in the unit being modeled. Further, when training, the probabilities associated with each state are affected by those parts of the speech signal that roughly align with individual events in the signal. Each model's states are ordered in time, from the early phonetic events in the unit modeled to the later events, although states are rarely explicitly labeled. In both training and testing, state transitions are only allowed from left to right (since speech always proceeds forward in time).

The Markov machines for ASR are usually called HMMs [230] because details of the models' operation must be inferred through observations of speech, not from any actual internal representation of speech production (e.g., VT positions). The most common HMMs are those for phonemes and for words. Phonemic HMMs are more general, since word HMMs can be constructed from them. Assume a phoneme X is to be modeled in context WXY (i.e., phoneme W precedes and Y follows). A phonemic HMM for X might use three states to represent (in order) an initial transition spectrum from W, then a spectrum from the middle of X (a presumed steady-state portion), and a final transition spectrum to Y (four to five states are also widely used for phonemic HMMs). To account for acoustic variability, each state is represented by a joint pdf of spectral envelope information (call this B) rather than a fixed spectrum [the transition likelihoods are traditionally labeled as a_{ij}, or with an A matrix; the observation likelihoods reside in B, which is as simple as a matrix if vector quantization (VQ) is used]. When evaluating frames of an unknown phoneme with this HMM, the likelihood of a match appears in the B values. Formally, the correspondence between model states and acoustic segments is described by an observation probability matrix B whose elements $b_j(k)$ are the likelihoods of observing "symbol" k (columns in the matrix) in state j (rows). Symbols normally are associated with spectra of acoustic segments, as represented by vectors of 8 to 14 parameters; thus, B is a matrix of multivariate probability distributions. These column distributions may be discrete, e.g., 64 to 256 spectra chosen by VQ (as in common speech coding applications) [82]. Continuous B functions, on the other hand, e.g., a Gaussian pdf [127], however, provide better ASR performance than discrete functions [231].

In the basic *Bakis model* of Markov chains [121] (see Fig. 4), three transitions are allowed from each state: 1) a *self-loop* transition back to the same state i (representing the *insertion* or continuation of an acoustic segment); 2) a transition to the next state $i+1$ (a *substitution* or new segment); and 3) a *skip* transition to the following state $i+2$ (corresponding to the *deletion* of the acoustic segment for the skipped state). (More general HMMs allowing other transitions typically increase computation but not ASR accuracy.) The *state transition* matrix A of probabilities a_{ij} is, thus, zero below the main diagonal, with only three diagonals being nonzero. As such, the major storage and computation for HMMs lies in the observation pdf, and not in the transitions. This also reflects the value of the two types of information: ASR discrimination resides primarily in the B densities. The A probabilities essentially allow the proper functioning of the model, giving it temporal flexibility in aligning states with phonetic events. In practice, different HMMs often have very similar A matrices.

2) Training Speech Markov Models: The major challenges for ASR when using HMMs are to train them properly and to design an efficient evaluation procedure. In practice, the *training* of the parameters in each HMM means making an initial estimation of the A and B matrices and then doing an iterative *reestimation* whereby their parameters evolve toward a good model. A standard method is the E–M algorithm [192], a simple procedure similar to that of designing VQ codebooks in speech coding, where a good initial estimate accelerates the model evolution and raises the chances of achieving a good model. The training iterations normally use a steepest-gradient or hill-climbing method, which is only locally optimum. An exhaustive search for a global optimal model is rarely feasible. Due to the complexity of the models (large numbers of interrelated parameters) used in most of ASR, training techniques are often simple. Since training is usually *off line* or nonreal time, several HMMs can be developed for each unit, starting from different initial estimates, eventually choosing the one with highest accuracy [231]. Such training methods can accomplish adaptation, as well as initial model creation, to different speakers and to handle various noisy or stressed speech conditions [94].

In reestimation, an initial prototype acoustic vector for each state often provides the first estimate of the mean of the pdf. An initial estimate for the W matrix (if it is allowed to vary) may just be an average matrix from general speech. Both the mean and matrix evolve iteratively through an averaging or clustering procedure [76], exploiting the rest of the training data. Similarly, each transition probability may initially be assigned a likelihood of $1/L$ (where L is the number of allowed transitions from each state), and then evolves in response to how often that transition appears in the frames of the training set. (The self-loop (a_{ii}) probabilities are usually set higher than for other transitions, since the typical HMM has significantly fewer states than the number of frames that it models.) The most common training procedures are variations of the *Viterbi algorithm* [70], [121], [175] (e.g., the gradient method) and the *Baum–Welch algorithm* [18], [167], also called the *forward–backward* (F–B) algorithm [16]. These are a specific case of the E–M algorithm [52], [233].

Consider an observation matrix O of spectral vectors from U training utterances; the elements O_{kj} are typically MFCC vectors, where $k = 1, 2, \ldots, F$ (F frames/utterance) and $j = 1, 2, \ldots, U$. The probability of O being generated by an N-state HMM with parameter matrices A and B is

$$P = \prod_{j=1}^{U} \sum_{i_1, i_2, \ldots, i_F} b_{i_1}(O_{1j}) a_{i_1 i_2} b_{i_2}(O_{2j}) \ldots a_{i_{(F-1)} i_F} b_{i_F}(O_{Fj}) \tag{9}$$

i.e., the joint probability (over U training tokens) of the sum of the path probabilities for all possible paths through the model (each index i ranges over all N states). Each path probability is the product of F probabilities b (functions of the observation O and corresponding to the F frames of each token) and $F-1$ transition probabilities a (representing the $F-1$ transitions among states for F frames). HMM training determines the A and B matrix parameters to maximize P, which is a classical problem in constrained optimization ("constrained" because each matrix row or column is a set of probabilities and must sum to one).

In the F–B method, a forward joint probability $\alpha_t(i)$ for each N-state HMM is defined for being in state j ($1 \le j \le N$) and observing the data up to time t (i.e., $O_1, O_2, \ldots O_t$). The final probability for each model over U frames is

$$P = \sum_{i=1}^{N} \alpha_U(i)$$

calculated inductively starting initially with $\alpha_1(j) = \pi_j b_j(O_1)$, with induction step

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(O_{t+1}).$$

While theoretically N^U paths must be examined, this method notes that, in calculating P, the local probability $\alpha_t(i)$ at each state i needs no more than the product of the next observation likelihood $b_j(O_{t+1})$ and the N weighted likelihoods from the previous time t [from the previous N $\alpha_t(i)$]. A similar backward probability $\beta_t(i)$ for the remaining speech data (from t to U frames), starting from state i, is also used. The total calculation is reduced from a theoretical and extremely large $2UN^U$ calculations to a feasible N^2U operations. As with most iterative procedures, the reestimation cycle continues until an optimality measure (in our case, P) no longer increases between iterations, but the iteration is often truncated after a few cycles (to minimize computation).

One difficulty with this procedure, when dealing with relatively small amounts of training data, is that some parameters are assigned values of zero (e.g., no state transitions observed between states i and j, or the kth spectrum was not observed during a given state in the training data). If we could be sure that zero estimates were appropriate (i.e., with very extensive training, such values would remain unobserved), then these zeros would be correct. However, with undertraining, we cannot be sure, and usually we revise each null estimate (except for forbidden transitions) to have an arbitrary small probability (while rescaling the other probabilities accordingly, since they must sum to unity) [231]. Sometimes, observation $b_j(k)$ can be assigned interpolated estimates from neighboring values [16]. If we have used L frames to train a model, a minimum probability of $1/L$ can be assigned to each b parameter (for a VQ codebook with $M \ll L$ entries). The assumption here is that the next training token (the $L+1$th frame, not actually available) might exhibit the missing spectrum.

Given an HMM structure and a choice of acoustic parameters, training methods such as the Baum–Welch algorithm estimate good values for the models. However, the structure and parameters are chosen first, and it is difficult to make an optimal selection, since we do not know what model or parameters are best.

3) Classification via HMMs: Given a test utterance to recognize, how can each HMM be evaluated efficiently? Equation (9) is also used for testing, by calculating P for the corresponding test observation O, for all HMMs that may be candidates. The ASR output is the text that corresponds to the model yielding the highest P. (Since in testing there is only a single observation matrix O, as opposed to multiple training tokens, the second subscript for O in (9) is not used here.) A version of the Viterbi algorithm [231], [232] yields results similar to more exhaustive testing procedures, while considerably reducing calculation. The simpler Viterbi search is much faster than the F–B method, as it locates only the single best path in the HMM, whereas theory requires calculating the sum of probabilities of all paths. When log probabilities are used, no multiplications are needed; e.g., if $d(t,i)$ is the minimum "distance" (in terms of summed log likelihoods) to state i at time t, then in a recursion

$$d(t,i) = \min_j\{d(t-1,j) - \log a_{ji} - \log b_i(O_i)\} \qquad (10)$$

where $d(1,i) = -\log \pi_i - \log b_i(O_1)$.

The Viterbi method also yields backtracking, to discover which states were visited (and when) along the best path. This is especially useful in training because the best path provides a direct segmentation of each utterance, specifying which frames correspond to each of the states of the HMM. There is no segmentation possible in the F–B method, as all paths are examined and contribute equally.

4) Subword Models and Context-Dependent Acoustic Models: Phonemes and words are typical units for ASR. The choice of unit size reflects ability to characterize coarticulation, to handle different vocabularies, and to be efficient. We see the same tradeoffs in ASR and speech synthesis. In the latter, there is more vocabulary flexibility and smaller memory requirements when using smaller stored speech units (e.g., phonemes). However, quality decreases with small units in TTS, as does accuracy in ASR. For large-vocabulary ASR, one way to increase accuracy, compared to a phoneme-based system, is to use diphone or tied-phone models (see Fig. 5). These represent a compromise that avoids both the huge memories needed for word models and the poor ASR accuracy that occurs with *context-independent* (CI) phonemic HMMs.

Diphones are speech segments having the size of phonemes, but derived from phoneme sequences divided in the middle of each phoneme. Thus, each diphone retains the transition between adjacent phonemes (e.g., from the sequence /ua/, the second half of /u/ followed immediately by the first half of /a/). For TTS, memory needs increase significantly for diphones (versus phoneme units), since, assuming a language with N distinct phonemes (e.g., approximately 32 for English), there are N^2 diphones. Furthermore, because diphones correspond to dynamic speech segments (phonemes can be steady-state units), each diphone typically must store more data (e.g., frames) per unit.

To quantify, note that English has more than 50 000 common words. (Most native speakers may only use about 5000, but these word sets vary greatly.) The average English word has fewer than eight phonemes, and the typical word lasts less than a second. With efficient coding methods using about 4 kb/s, a 200-Mb storage is feasible with word units, with the possibility of significant reductions if universal vocabulary (i.e., unlimited TTS) is sacrificed. However, it is quite fatiguing for a single speaker to utter thousands of words in a uniform fashion, as would be needed for word-based TTS or SD ASR. (This issue is a significant problem for synthesizers storing large amounts of speech from individual speakers, and for large-vocabulary SD ASR.)

Assume that approximately 1000 diphones can represent a language and that each diphone model needs about 20 frames. Such a subsyllable template memory could handle unlimited vocabulary (for either TTS or ASR) and still only require memory comparable to that of 500 word templates at 40 frames/word [246]. Using subword speech units, the reference memory must also have a dictionary that maps each word into its sequence of these smaller units; this dictionary is, however, small compared with the model memory, e.g., the former needs only about 10 b/diphone for

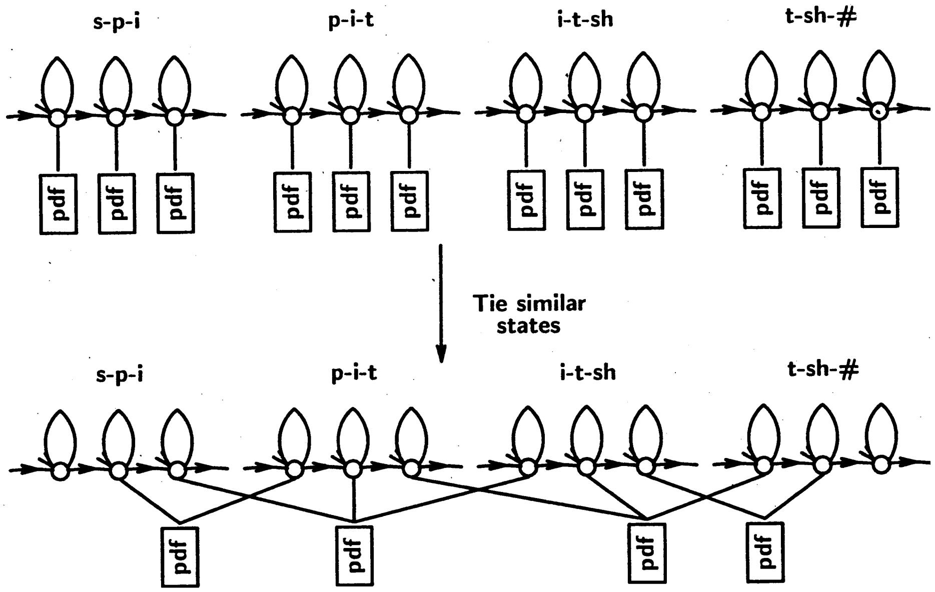

Fig. 5. View of state tying for HMMs.

an average of 6 diphones/word. Diphone ASR may require a hand-segmented database, however; syllables have also been used as ASR units [79], [140].

The major disadvantage of replacing word models with subword models (as with most ASR simplifications) is that ASR accuracy is reduced [246]: the acoustic effects of coarticulation and stress extend well beyond adjacent phonemes (the range of most subword units) and construction of word models by concatenating smaller units requires significant modifications to the merged units (e.g., at each unit boundary, smoothing template frames or merging HMM states). In addition to modeling spectral smoothing at unit boundaries, some units should be shortened when forming polysyllabic models, because word duration does not increase linearly with the number of syllables in a word.

A basic ASR system may use CI acoustic models, where each model unit is trained independently of its neighbors, ignoring coarticulation and stress effects. This approach minimizes undertraining problems, in which model parameters are often poorly estimated when using more numerous *context-dependent* (CD) models. The latter are typically *triphone* HMMs that represent N phonemes each with N^2 models (hence, N^3 models). For example, /d/ would have a separate model for each combination of all left and right neighbors (/idi,idu,uda…/). CD models capture most local coarticulation, but require developing and searching N^3 models (e.g., with $N \approx 30$—40 phonemes). Many ASR applications do not have enough memory and training data to properly train a complete set of such models. (A compromise is to use N^2 *diphone* models, conditioned on either right or left context alone.)

5) Tying Subword Speech Models: As large numbers of models are generally preferred (to increase ASR accuracy), we seek to reduce the deleterious effects of undertraining. One common way is to *share* or *tie* parameters across ASR models, where the same parameter values are used in all models that relate to a given phonetic context [30], [240]. Having a separate triphone HMM for each sequence of three phonemes is inefficient, because many contexts have very similar coarticulatory effects; e.g., the effects of labial phonemes (i.e., /f,v,p,b,m,u,o/) on an adjacent sonorant are very similar.

Tied models share the same parameter values, which reduces the number of parameters to train. Tying can be automatic (e.g., with unsupervised, data-driven decision trees [28], [153]) or guided by linguistic properties (e.g., clustering contexts with phonetic labels such as labial, velar, fricative, etc.). When tying involves Gaussian pdfs, the models are called *semicontinuous* HMMs. Related *allophone clustering* also reduces the number of models, while retaining the power of CD models and improving their accuracy [170], [244]. With tying, both ASR computation and memory for the reference models are reduced. Sharing has been applied at many levels: word, syllable, diphone, phone, frame, and parameter [12], [14].

6) Comparing the Two Most Recently Popular ASR Methods: While HMMs have dominated ASR in the last two decades, DTW was popular in the 1970s. In DTW, one or more templates represent each word (or phoneme) unit, where the template consists of a sequence of spectral feature vectors. There is no direct stochastic modeling; so the template has as many vectors as the unit has frames. In ASR evaluation via DTW, a global distance measure is accumulated as a sum of local distances, via a succession of comparisons between corresponding frames in the reference and test utterances. To accommodate timing variations (e.g., compare versions of the same word having, say, 48 and 53 frames each, and still achieve a small global distance despite temporal variability), each frame in the test model is compared to several in the reference, obeying both global and local constraints on timing. The result is better than with a linear interpolation of frames (e.g., mapping the 53 frames onto 48 linearly), while also avoiding the excessive computation of a wider set of comparisons.

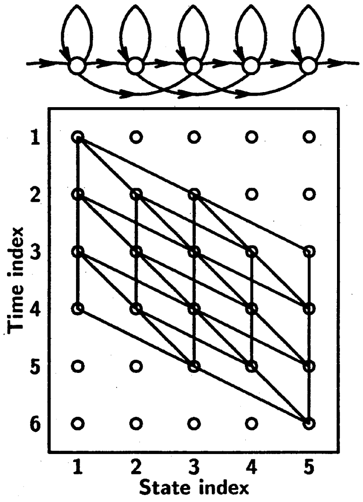

Fig. 6. Interpretation of an HMM as a trellis.

Both HMMs and DTW use dynamic programming [200], [212] to find the best match between a test utterance and different reference models. Viterbi HMM determines the sequence of states $i_1, i_2, \ldots, i_F$ in the HMM that maximizes likelihood, while DTW finds a warp path to minimize an accumulated distance (see Fig. 6). A major advantage for HMMs over DTW is that DTW requires expensive Mahalanobis distance calculations, while HMMs often use table lookup for probability calculations. DTW is a nonparametric speech model, which codes temporal and speaker variability in a "brute force" approach of storing many templates, each having many frames of features. Warping constraints and the use of a distance measure attempt to account for some variability, but DTW requires much computation.

Compared to DTW, HMMs incorporate more structure into their models, capturing much of the variability of speech in their A and B matrices during training. Unlike training, the recognition phase of ASR must usually be rapid, as it often serves a customer on-line. HMMs sum path probabilities for an FSM whose size (in states) is perhaps an order of magnitude less than the number of frames used in DTW templates (e.g., three to eight states versus 25 to 40 frames for word models). Where DTW applies a fixed penalty to temporal distortions (e.g., a constraint accommodating at most a change in speaking rate of 2 : 1), HMMs are more flexible in modeling temporal and spectral variability.

HMMs require much data to obtain a good set of model parameters, and are inflexible in terms of exploiting some acoustic–phonetic cues useful in ASR (e.g., durational and spectral transient cues). DTW works better with small amounts of training, but is poor at modeling spectral variability. Both DTW and HMMs, while yielding good performance in limited applications, are too rigidly formulated by their structures, which exclude many acoustic factors known to affect human speech perception. By themselves, neither DTW nor HMMs are likely to solve the general problem of ASR. In the future, significantly improved accuracy will likely come from research that combines the best aspects of different ASR methods.

7) Durations and Unit Size: In an HMM, durational information resides in the probability of the self-loop transition a_{ii} for state i. The likelihood of remaining in state i for a total of n frames is $a_{ii}^{n-1}(1 - a_{ii})$. This is a *geometric* or exponential pdf, and is a poor model for speech durations, which tend to follow a Poisson or gamma pdf. In particular, the geometric pdf notes a most likely state exit after only one frame ($n = 1$), whereas the average stay in an HMM state is usually several frames. Attempts to restructure HMM architecture to allow nongeometric models for duration [111], [189] have raised ASR accuracy, but cost more [127], [164]. Most ASR systems assume that their transition probabilities are of secondary importance and, thus, concentrate instead on modeling the state probabilities (i.e., the B matrix) [26].

Recognition of isolated words usually employs word HMMs, unless the vocabulary must be flexible. In continuous speech recognition, the HMMs are usually phonemic and, hence, fewer and smaller in size than for large-vocabulary IWR using word HMMs [27]. To model phonemes, HMMs likely need only about three states (roughly corresponding to the initial transition, steady state, and final transition of the phoneme). At one per phoneme, a set of 30 to 40 CI HMMs can be used to handle all words in a given language. This represents a great memory and computation savings when compared with IWR.

8) Improving HMMs: ASR usually processes speech with frame updates of 10 ms, and assumes independence of successive HMM states. This first-order assumption of HMMs is both a strength and weakness. It allows much model simplification, both in training and testing. However, ignoring past history forces use of less precise pdfs (i.e., with larger variances) and lowers accuracy. The use of delta coefficients as an indirect attempt to include timing information over several frames (up to 50 ms or so) improves accuracy, but is inefficient in exploiting the dynamic spectral information in the speech signal. Future ASR must be better at exploiting timing information. Recent attempts in this direction include *stochastic trajectory models* [217], [262], *trended* HMMs [53], [54], [238], *linear trajectory* HMMs [250], and other segmental models [104], [105]. Attempts to use Markov models of higher order have shown increased ASR accuracy, but the computational complexity of such models has hindered their application [180].

F. Adapting to Speaker Variability

The ASR methods discussed so far all try to accommodate variability in speech. We now examine more closely the sources of variability (intraspeaker, interspeaker, and environment) and how ASR handles them. No two utterances are exactly alike, as humans are incapable of precise repetitions (indeed, speech synthesizers that produce exact periodicities sound too mechanical). While the largest variations in speech signals (assuming a fixed text) are due to different speakers or recording conditions, large variations occur with each speaker's speech due to changes in speaking rate [252] and emotion [197]. The Gaussian pdfs in HMMs are often appropriate models for intraspeaker variations, and DTW is specifically designed to handle speaking rate changes. Given the large differences in speech that occur for different speakers, the simplest way to handle interspeaker variability is to use custom models for each speaker, in SD ASR systems: each user trains a personal set of HMMs to "learn" the voice, and only the reference models for that speaker are examined at recognition time.

Accommodating different speakers is straightforward, but expensive with DTW: separate templates are stored for individual repetitions of words, on the reasonable assumption that one utterance/word is inadequate to characterize all the intraspeaker variation. With HMMs, accommodating different speakers is more efficient. A single model per speech unit (e.g., phone or word) can be used, although using context-dependent HMMs often raises this number. For speaker-independent (SI) ASR, where the system does not know the identity of the speaker, the models are trained on a large number of speakers. A single model would average training data across all speakers. More often, a model is trained from each set of people characterized by some trait relevant acoustically (e.g., gender, size, age, dialect). The most common division into multiple HMMs uses gender, i.e., separate male and female models. This doubles runtime computation, since both models must be tested, with the final ASR output being chosen from the model with the highest score. The alternative of using automatic gender estimation risks ASR errors caused by gender mistakes (e.g., while women's $F0$ is usually about double that of men's, simply basing a gender estimate on $F0$ is imperfect).

When diverse conditions (e.g., different speakers) occur in the training of any HMM, the state pdfs of the model suffer increased variances, which in turn causes reduced discrimination between models. A model trained on tightly controlled data (i.e., one speaker uttering a specific phoneme in a given linguistic and recording context) will often have a clear low-variance Gaussian pdf. Averaging across diverse conditions broadens the pdfs, at times to the point where the resulting pdfs have great amounts of overlap across the phonetic classes they are meant to discriminate.

ASR accuracy is much higher with well-trained SD models than with SI models, because the former are specific to each individual's voice. However, many applications cannot use SD models (e.g., when serving the general public, rather than specific customers who train the system for repeated use). Most users of SI systems, however, speak for periods long enough to allow the system to gradually adapt to their voice. Thus, a common approach is to use an SI system to recognize the initial speech of an unknown person, and to continually adapt the HMM parameters based on the incoming recognized speech [109], [150], [288]. Thus, they transform from SI models to *speaker-adaptive* (SA) models. (With enough on-line training, SA models can approach SD models in ASR accuracy.) This is most feasible when the system is aware of any speaker changes (e.g., in a telephone call, a single speaker is assumed); in multispeaker audio conferences, speaker changes are often abrupt and hard to track. Even short amounts of adaptation data are useful (e.g., 9% improvement in one study with a single 3-s utterance [5]). This adaptation is especially useful for foreign speakers (whose error rates with SI models are up to three times as high as for native speakers) (e.g., improvements up to 50% [56]).

As HMMs are founded on maximizing probabilities, the most common adaptation method is the *maximum a posteriori* (MAP) technique [38], [86]. MAP augments the SI model with information about the incremental adaptation data. Often several minutes of training data are required, because only models corresponding to actual phonemes in the adaptation speech are updated. A related method to handle widely varying speakers, transmission channels and background conditions is called *ML linear regression* (MLLR) [77], [161], which calculates transformations of the acoustic space (from a general speaker orientation toward a speaker-specific one) using unsupervised adaptation data, which are grouped automatically. A relatively new sort of adaptation is *vector-field smoothing*, which adapts parameters for many models incrementally and rapidly, needing little data [276], as can *tree-structure speaker clustering* [149] and the method of principal components analysis. The latter extracts a small number $K \approx 5$ of "eigenvoices," which then allows adapting to each new speaker in the spanning K-space [152]. Other forms of speaker normalization [126] include frequency warping [89].

The methods just described are all founded on mathematics, with little reference to speech production. Since a major reason for the differing acoustics from different speakers is the size of their VT, it is reasonable to try to estimate VT size, especially length, which changes important spectral details of speech more than other aspects of VT size. In the adaptation procedure called *VT-length normalization* (VTLN), such an estimate can be made by estimating an average F3 [36], which tends to vary less then $\mathrm{F}1$ or $\mathrm{F}2$ [87], [166], [278], [282]. Children's speech is particularly difficult to recognize [49].

As a rule, the increase in ASR accuracy due to adaptation is proportional to the amount of training data. In the case of foreign speakers, however, compact models allow faster adaptation; SI systems are mostly trained on native voices, and small amounts of speech from a nonnative cause large changes in the resulting SA model [56].

G. Environmental Variability (Robustness Against Noise)

Pattern recognition of speech analyzes the signal, trying to extract textual information (and ignoring other aspects). What is critical information for ASR may not be useful for another task such as speaker identification. One often says (simplistically) that what is "noise" to ASR is "signal" to speaker identification. In ASR, we try to normalize the speech processing to eliminate acoustic differences unrelated to the identity of the underlying text (e.g., speaker-specific characteristics, but communication channel distortions too). For speaker identification, the text is irrelevant, and processing focusses instead on what ASR normalizes away. The information content of speech is not neatly divided between signal and noise, whatever the application, since certain parts of the speech signal provide information simultaneously for ASR, speaker identification, language identification, etc, while others (e.g., very weak portions) may furnish little useful information for any practical application (other than speech versus nonspeech detection, i.e., voice activity detection).

Among the aspects of variability affecting speech signals are distortions caused by the communication channels (e.g., telephone line, radio transmission) and the recording environment (background noise and other unwanted sounds). For some ASR, the speech signal can be partly "cleaned" or normalized in a preprocessing stage (before analysis). If the environmental conditions are stationary or their variations can be determined, such effects may be attenuated in the signal. The most common variation is simply due to intensity. An amplitude adjustment changes the maximum signal amplitude in the test utterance to a standard level, to help account for variations in recording level, distance from the microphone [264], original speech intensity, and general energy loss in transmission. Such variations are assumed to be slowly changing, which allows the system to update the amplitude scaling factor at relatively long intervals, e.g., at each pause in the utterance. Automatic gain control, as in radio receivers, is sometimes used, but gain adjustments must not be done so rapidly as to obscure linguistically relevant speech dynamics [48], [134].

Speech variation owing to background and channel noise is the most difficult variability to handle, as it is the least predictable [2], [157]. Extraneous sounds near the recording microphone appear in the resulting speech signal as additive distortion, whereas transmission channels often impose convolutional noise. Basic spectral subtraction techniques may help against additive noise, and cepstral methods (which convert multiplication in the spectral domain to cepstral addition) help suppress convolutional noise.

There are four basic approaches to assist with distorted speech: trying to remove the noise, altering ASR features, modifying ASR models, or changing the distortion measure. Many methods used for speech enhancement can be applied as *preprocessors* prior to recognition of noisy speech. However, there should be differences in the two applications of ASR and enhancement (as speech analysis may differ for speech coding and ASR). Among the ways to improve speech for listening, only those that improve the spectral envelope should apply for noisy ASR. Enhancement generally makes speech sound less annoying, but does little to improve intelligibility (the sole interest of ASR). ASR for noisy speech should likely highlight the frequency ranges with most energy, because such frequencies correspond to the strongest formants, and are often the least corrupted by noise [169]. The strongest resonances are the last spectral components to succumb to a rising level of background noise.

ASR accuracy is reduced by any signal distortion [1], [60], [88], [95], [131], [191], including changes in speech production as a result of noise (e.g., the Lombard effect, where people shout to be heard above noise [129]). Accuracy degrades much more rapidly for machines than for humans. One study noted an error rate increase from 7% to 13% as signal-to-noise ratio (SNR) dropped to 10 dB even with noise compensation enhancement (without compensation, it was 40%), whereas human listeners maintained a 1% error rate [168]. The large difference between human and computer speech recognition in degraded conditions suggests strongly that much research remains before solving the general ASR task.

1) Speech Enhancement: Transmission channels are often assumed to add noise and to filter the speech, with both effects being quasi-stationary. If the spectrum of the noise and spectral details of the channel filtering can be estimated, then some noise may be removed by spectral subtraction and a filter with inverse channel characteristics may enhance the distorted speech. The noise estimation is typically done by analyzing the distorted speech during portions of the signal where the energy is weak (i.e., during presumed pauses). This yields a spectral estimate of the combined background and transmission noise. Since noise is random, however, the benefits of a spectral subtraction of a noise estimate are limited, because phase is ignored in the process. A much greater improvement occurs if more than one microphone (or, in general, an array of microphones) is available to simultaneously capture different versions of the speech and noise (e.g., one microphone inside a pilot's helmet and another outside) [66], [199], [209]. Such multimicrophone applications, while rarer and more expensive than standard single-microphone cases, allow direct waveform subtraction in the time domain of an estimated version of the distortion (via adaptive sound cancellation methods) [273]. This is similar to echo cancellation on telephone lines; such methods have large SNR gains. However, most applications of ASR are limited to use of a single microphone.

2) Robust Features: To improve ASR in distorted conditions, we often revise the basic acoustic features, focussing on those that better resist the distortions. The most common adjustment is amplitude normalization: basic energy is often discarded, as amplitude can vary greatly due to transmission and environment; i.e., simple energy may be unreliable for phonetic decisions. Human listeners exploit energy to help distinguish many aspects of speech, but if energy is not properly normalized, its utility in ASR is limited. In practice, the difference in energy between successive frames (delta energy) is more commonly used in ASR than energy itself. When ASR need not be real time, energy may be used [117].

The cepstrum (e.g., the MFCCs) is more robust against noise than LPC [65]. In addition, simple techniques can be applied to increase ASR accuracy; e.g., the low-order MFCCs can be deemphasized (via a weighting window, or "lifter"), because these coefficients are too much affected by the transmission channel (the lowest-order coefficients represent the smoothest aspects of the spectral envelope and, thus, have less phonetic information and are more subject to changes in spectral tilt, which is largely irrelevant for ASR). As a result, a "cepstral lifter" that emphasizes the mid-range MFCCs seems optimal.

The RelAtive SpecTrA (RASTA) analysis method [99] tries to accommodate noisy speech channels by filtering time trajectories of speech parameters; its very broad bandpass filtering eliminates very slowly varying distortions. This is similar to another popular way to reduce the effects of channel mismatch between testing and training: *cepstral*

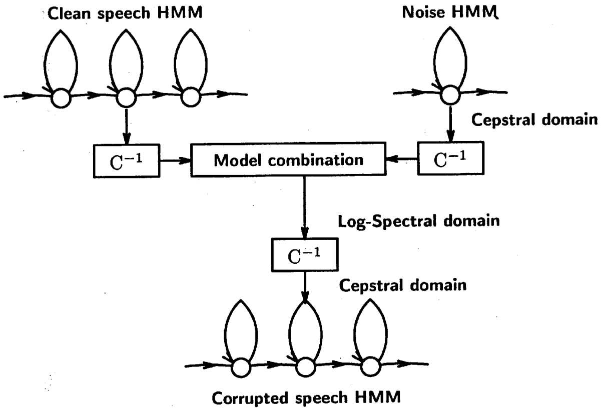

Fig. 7. Operation of parallel model combination.

mean subtraction (CMS). As with RASTA, CMS eliminates very slowly varying aspects of the speech signal, which are presumed to derive mostly from channel distortions. The average value for each ASR parameter, integrated over time periods that include several phonemes, is subtracted from each frame's parameter value. Similarly, *blind equalization* removes channel effects (e.g., *signal bias removal* [235]). Channel noise is often assumed to remain constant during a speaker's utterance; however, the fading effects that portable telephones suffer require frequent estimations to combat such nonstationary noise [234].

In CMS, the removal of the long-term spectrum from the representation of each speech frame significantly alters the parameters, and forces interpretation in terms of relative changes. This form of adaptation, thus, minimizes environmental and intraspeaker effects, and removes convolutional channel effects (since the cepstrum converts convolution to multiplication), but does not help with additive channel noise and nonlinear distortions. Long-term average subtraction cannot be applied to very brief utterances, as that would remove too much phonetic information (the subtracted values would be phoneme specific, rather than channel specific).

3) Model Compensation: An example of how ASR models can be transformed to improve accuracy under conditions of either additive or convolutional noise is *parallel model combination* (PMC) [78] (see Fig. 7). HMMs are first built for both clean speech and typical noise signals, using standard MFCCs. Then state parameters are converted back into linear spectra and added at different levels of noise. Finally, the speech-plus-noise spectra are converted back into MFCCs. In cases of stationary noise, a simple one-state HMM suffices as a noise model; nonstationary noise requires more complicated models. In practice, speech received by an ASR system may suffer any type of noise and noise level varies greatly. Thus, PMC must try to approximate the most common types of noise for a given ASR application. In general, PMC is expensive.

This approach to handling speech variability via increasing numbers of models for explicitly identifiable sources of variation can be extended to background environments and transmission channels. One current ASR task (Broadcast News) consists of music, speech, and other sounds on commercial radio; a common approach is to classify the sound input by type first and then to apply the model trained on the appropriate data [255].

4) Robust Distortion Measures: The effects of distortion often vary in different frequency bands, which suggests treating bands according to their contributions to ASR. In speech coding, subband coders process each band according to its perceptual importance (e.g., more bits at low frequency). For multiband ASR in noisy conditions, accuracy can be raised by combining likelihoods from different frequency bands [207]. Partitioning the speech spectrum does not seem to lose phonetic information, and can exploit differences in timing across bands [188]. If speech is subject to bandlimited distortions (e.g., rings, tones, horn sounds), discarding a corrupted subband is helpful [187] (the so-called missing feature approach [46]).

H. Language Models

The initial speech recognizers of the 1960s and 1970s assumed that the acoustic information in the speech signal was all that was needed to produce a text output. There was little exploitation of the fact that most spoken utterances follow rules of grammar and semantics. In the last two decades, in ASR systems, LMs have become equal partners with acoustic models. The term LM refers to constraints on word sequences imposed by the syntactic, semantic, and pragmatic rules of the language being spoken. If speakers chose words randomly, LMs would be useless. Indeed, in applications such as recognizing credit card or telephone numbers, where the utterances are simply digit sequences with little predictability, no LM is used.

In most applications, however, exploiting the great redundancy in word order is crucial for ASR accuracy [123], [201]. Given prior recognized words in an utterance, the average number of words P that a recognizer needs to consider as candidates for the immediately ensuing word is much smaller than the actual vocabulary size V. (For nonredundant digit sequences, $V = P = 10$.) P is called the *perplexity* or *average branching factor* (ABF) of an LM. Many practical applications use V on the order of 1000 to 10 000 words, whereas typical values for P for normal texts rarely exceed a few hundred [287]. Thus, word predictability reduces the range of word choices often by more than an order of magnitude [145]. P is a measure of an ASR grammar's restrictiveness. (In terms of a word FSM, P is the mean number of transitions leaving the node for each current word.) One difficulty of fully exploiting this lies in the fact that the relevant prior word history often exceeds a dozen words (i.e., many preceding words influence future word choices).

Perplexity is related to the concept of *entropy* H, as a way to characterize the difficulty of a given ASR task [16], [247]: $P = 2^H$; thus, H is the effective number of bits to characterize the average number of words to consider. Given a pdf $p(t)$ for a text t of n words from a vocabulary of V words, the entropy per word is

$$H = \lim_{n\to\infty} -\left(\frac{1}{n}\right)\sum_t p(t)\log_2 p(t). \qquad (11)$$

Thus, $H \leq \log_2 V$ (reaching the maximum in cases where word sequences are completely random). To not use a grammar in ASR is equivalent to assuming that word order is random (i.e., conditional probabilities for words equal their marginal distributions). In one study, where both humans and machines recognized utterances that obeyed no grammar (vocabulary of 1000 words), there were 17% ASR errors and 2% human perception errors [168], which illustrates both the importance of LMs and the continuing gap between human and machine performance in recognizing speech.

LMs are stochastic descriptions of text, usually involving the likelihoods of local sequences of N consecutive words in training texts (typically $N = 1, 2, 3$). Integrating an LM with acoustic HMMs is now common in most ASR systems. The assumption is usually made that statistics obtained from text that does not correspond to actual speech (e.g., most forms of text available on computers) can be applied to help ASR. Ideally, LMs should be developed from transcriptions of actual spontaneous speech (the largest application for ASR). In practice, reliably transcribed speech data is very limited, since human transcription is costly and machine transcription (i.e., using ASR output to train its LMs) often contains many errors. Thus, in yet another compromise, the LMs used for most ASR come from text lacking origins in speech.

1) Grammars in ASR: Absent an LM, ASR could output a sequence of symbols representing phonemes (or other phonetic segments) that correspond to the estimated text. Reflecting the uncertainty of ASR decisions, this output sequence may itself be stochastic, involving possibilities, e.g., a list of likely boundary locations and phoneme candidates for each segment, often in the form of a grid or *lattice*. (With word-based HMMs, the lattice's elements are words, instead of phonemes, and boundaries for different word combinations often overlap.) Owing to the difficulty of segmentation, specifying the ASR output word sequence, which requires assigning words to the lattice sequence, may be left to a postprocessor that compares the symbol string or lattice against all permissible vocabulary words [190]. We apply a *grammar* or structure of permitted phoneme or word sequences, which raises ASR accuracy by eliminating candidate sequences that are illegal (as per the grammar) [258].

Traditional grammars have syntactic rules by which sentences are parsed into component words and phrases [35]. Natural language (e.g., normal English), however, has a very complex grammar; computational linguists have been unable to design an algorithm that distinguishes all grammatical sentences from ungrammatical ones. Since ASR dictionaries usually have part-of-speech information, in addition to the phoneme sequence for each vocabulary word, we can impose simple grammar rules how words from different syntax classes (e.g., nouns, verbs, prepositions) may combine to form acceptable sentences, thus, not considering ungrammatical sentences as ASR output. Some systems use a *slot-frame* grammar, where speakers are presumed to utter only sentences that follow a very specific format. Positions or slots in such frame sentences must be filled from specific sets of words; e.g., a date-time grammar could impose the frame <month-date-*at*-hour-minutes-(*P.M.* or *A.M.*)>, where the *month* slot is restricted to one of 12 words, the *date* slot to one of 31 numbers, etc.

Markov models are a form of FSM grammar, but not adequate to generate all and only legal sentences [155]. *Context-free* and *context-sensitive* grammars are sometimes used in ASR. Some systems even employ a *pragmatic* component, which aids ASR by keeping track of discourse structure [132]: as words are often repeated in a discourse (e.g., a dialogue between a person and a database), previously recognized words, stored in a cache memory, have their likelihoods elevated when evaluating ensuing words [151].

2) Details of LMs for ASR: ASR systems typically use FSMs to represent the statistics of both language and speech acoustics [122]. For acoustics, HMMs account for speech variability. For text, much simpler LMs attempt to capture the redundancy of word sequences in terms of the probabilities of occurrence for each word in the allowed ASR vocabulary. Most often, N*-gram* LMs estimate the likelihood of each word in the context of its preceding $N - 1$ words. Thus, *bigram* models employ statistics of word pairs and *trigrams* model word triplets [29], [247]. *Unigrams* are just *a priori* likelihoods for each word (ignoring context). Usually trained without supervision, LMs integrate all redundancies (e.g., both syntactic and semantic) in text. Although FSMs, LMs are not HMMs, as each text word is directly observable (words in training and output text are not "hidden," as is the underlying speech articulation for the acoustic models), and LMs use only transition probabilities (not state pdfs). Thus, basic LMs are quite simple. Nonetheless, LMs are often extensive in memory and computation, when they use second-order (e.g., trigrams) or higher statistics (unlike the first-order HMMs), and can require transitions among thousands of word states.

As vocabulary size V grows for practical ASR, in cases where speakers are not heavily constrained in what they can say, the size of the required LM (V^N states) grows exponentially. This is most serious for highly inflected languages (e.g., Estonian, Finnish), where each dictionary word has many derived forms with different suffixes. An English word such as *dog* has *dogs* as its only inflected form, and the (irregular) verb *eat* has only *eats*, *ate*, *eaten*, and *eating*, whereas French has typically 39 different derived words for each lexical verb. Thus, simple LMs that work well for English often need modifications to handle other languages [24] (e.g., German compound words, French homophones) [286]. Even for low-inflection languages (e.g., English), the large lexicons needed for general speech lead to seriously undertrained LMs, not enough appropriate texts for training, large memory requirements, and a lack of computation power to search all textual possibilities (for $N > 2$). Thus, most ASR has employed unigram and bigram statistics, with trigrams used if resources permit [26]. Text normalization and handling of capital letters is critical in many cases [3].

Properly trained high-order N-grams can be very informative for ASR. Long word sequences that appear regularly in text should be part of a general LM. However, the vast majority of N-word sequences for $N > 2$ are rarely found in available training texts, yet many are common enough to appear when testing ASR; e.g., fully 60% of trigrams noted when testing the Switchboard task [84] had not occurred in 2 million training words (150 hours of speech). Thus, one cannot simply set to zero the likelihoods for N-grams that are unobserved in training; such a poor LM would cause many ASR errors.

The likelihood of unseen N-grams can be estimated by *back-off* methods that rely on lower order statistics, i.e., use $N-1$-grams whenever N-grams have no observed cases [136], [32]. In an *interpolated* LM, all statistics combine to yield a likelihood

$$\alpha_0+\alpha_1 P(W_1)+\alpha_2 P(W_1|W_2)+\alpha_3 P(W_1|W_2,W_3) \quad (12)$$

where α_i are i-gram weights for the likelihood of current word W_1, given preceding words $W_2,\ldots,W_i$. If the trigram $W_3W_2W_1$ does not occur in training, $\alpha_3 = 0$ and the other weights are elevated (similarly, if no W_2W_1, then $\alpha_2 = 0$). (α_0 is assigned a small value to account for vocabulary words that do not appear in training.) The α_is must sum to unity, being adjusted in back-off methods to handle unseen N-grams. Many systems smooth LM distributions after their initial training, by assigning small probabilities to all unseen N-grams (and slightly lowering all other likelihoods) [32], e.g., *deleted interpolation* [116].

3) Integrating LMs Into ASR: How to combine an LM with an acoustic HMM is not immediately obvious. Probability theory suggests a simple joint likelihood $P(S,T) = P(S|T)P(T)$, where text T is a word sequence $W_N,\ldots,W_1$, $P(S|T)$ is the acoustic result (e.g., if T is a word, the likelihood of a word HMM), and $P(T)$ is the *a priori* likelihood from the LM for T. Since HMMs are designed for specific linguistic units, it makes sense to use conditional acoustic models. However, the detailed probabilities for the two cases are quite different in range. For an L-frame word, $P(S|T)$ is a joint likelihood of L transition probabilities and L state observation probabilities, whereas $P(T)$ is a much simpler function of a few N-gram probabilities. Despite the significant difference in nature of the two likelihoods (very low values for the acoustical joint probability of $2L$ likelihoods and much higher LM values), most ASR systems simply add the two log-likelihoods (recall that log-probabilities are often used to avoid underflow and minimize multiplications) to obtain a total probability: $\beta \log P(T) + \log P(S|T)$, where β is a weighting factor to balance the two contributions and is empirically determined to maximize ASR accuracy.

Like acoustic models, LMs can also be improved by adaptation. Just as SD ASR usually has much higher accuracy than is possible with SI models, accuracy increases if we use domain-specific LMs, i.e., trained only on text for a specific task, rather than a general LM to handle all situations. As HMMs can be formed for different speakers and recording conditions, LMs can be trained for specific vocabularies and their related applications (e.g., for doctors, lawyers, business). Since it is easier to categorize tasks than people, it is common to train HMMs across many speakers, but to employ specialized LMs. Specialized LMs can be combined in different ways to handle other applications, such as when the topic of an ASR dialogue changes; this usually involves back-off methods [119].

A problem for basic LMs is the exponential growth of the models as vocabulary increases, because such models only indirectly exploit redundancy in the words. Syntactic correlations in word sequences can largely be explained via grammar rules and parts of speech. Semantic correlations are more difficult to analyze; however, e.g., certain classes of adjectives normally occur prior to certain nouns (while other adjective-plus-noun sequences have much lower likelihoods): thus, “big chair” is common, but “fast chair” is not. It is easy to assign adjective-plus-noun sequences higher bigram likelihoods in general than noun-plus-adjective sequences, but much harder to classify all semantic relationships in terms of efficient classes. For example, consider grouping all size adjectives into a class S_1 (big, small, tiny, enormous, $\ldots$), all color adjectives in S_2 (blue, red, yellow, $\ldots$), etc., so as to increase the power of an LM. We could then substitute classes for individual words in the LM, as an alternative or complement to N-gram LMs. Training and then searching statistics for an LM with L classes (e.g., nouns, verbs) is much easier than for $V \gg L$ words. Such classes need not be explicitly identified (nor have traditional linguistic labels), but can be automatically determined via clustering methods [184]. The corresponding great reduction in LM size makes such an approach more efficient than N-gram LMs; however, class-based LMs assist less in ASR than word-based LMs, because they are less explicit and, thus, less powerful [119]. For example, a “color” class S_2 (based either on unsupervised training text, or on explicit semantic rules) would treat (likely) “brown cat” and (unlikely) “green cat” equally.

LMs exploit various linguistic constraints found in texts. Local constraints on immediately adjacent words are modeled well (but at significant cost) with N-gram LMs, but global constraints (beyond an immediate time window of N words) are much more difficult to capture efficiently [118]. Future LMs must remain efficient while integrating global contexts, e.g., phrase-based LMs [98] and latent semantic analysis [20]. For example, languages such as German have correlations between verbs that are widely spaced in a sentence, and even English allows many intervening words in phrases, such that words prior to and following a given phrase are often correlated, yet beyond a short window of a few words. Thus, we need wide-range LMs [20], [247].

I. ASR Search Techniques

At each stage of our discussion on ASR, we have noted principles as well as methods. Both are driven by the desire to improve recognition accuracy, but ASR creators often make compromises (relegating certain principles to have lower priority than cost) when designing algorithms. While noting that current ASR is suboptimal (given the large variance between human and ASR performances), we distinguish between weaknesses in ASR methods due to our lack of knowing how to do ASR better versus intentional compromises to accelerate recognition or to lower cost. In some cases, ever-improving computers will eliminate the need to make certain compromises. In others, we cannot await such future efficiencies, and must make some categorical decisions which go against how we would ideally like to do ASR (e.g., using first-order HMMs, tied states, diagonal covariances, and only low-order N-grams). The discussion in this section on search methods falls into the area of compromises usually done to accelerate ASR decisions, i.e., to make practical the search for the optimal output text in an extremely large set of possibilities.

The traditional way to find the best path in an HMM for an utterance is a *breadth-first* approach, which examines all possibilities before rejecting any. The popular Viterbi method is much more efficient than the F–B method for such searches, but the large ASR FSMs required for large-vocabulary, continuous-speech recognition still place large demands on computers, preventing their use in simple hand-held devices and raising cost in general. Thus, we have several alternative faster search methods.

In *best-first* and *depth-first* approaches (e.g., *stack* or *A** searches), the most likely transitions in an HMM from state to state are examined, and one backtracks to examine alternative paths after a path spanning the complete sentence is found. A *beam-search* approach [93] is similar to the DTW method, as it examines a narrow "beam" of likely alternatives around the locally best path [202]. A *beam width* δ is imposed on the probability P of the most likely partial hypothesis in an FSM, removing (*pruning*) all hypotheses whose probability lies below $L - \delta$ [91], [114], [241] (where $L =$ current best P). A technique called "phone look-ahead" is a common method to prune ASR hypotheses. Each time a hypothesis crosses a phone boundary (in the frame-by-frame processing), the HMM is examined to see if the next few frames give high enough likelihoods (if not, that path is abandoned).

A multipass strategy may be employed in which the first pass uses simpler models (less costly in memory and computation) to do an initial coarse recognition, while retaining multiple possibilities (i.e., pruning). Thus, the number of paths to consider during a more detailed second analysis pass is much reduced [133], [213]. Some correct hypotheses are inadvertently discarded prematurely, while avoiding a detailed examination of many useless paths [257]. The second pass is viewed as a *rescoring* of the N hypotheses furnished by the first pass (e.g., selecting from the initial N-best hypotheses) [265].

J. Speech Understanding

ASR has two major applications: transcription and understanding. In the first, the objective is to map speech to text, i.e., convert the speech signal to the text intended by the speaker. Such data entry may eventually replace the keyboard and mouse with a microphone (and related software), as speech is simpler and faster and possibly a less costly way to input data to machines.

Speech is usually assumed to have been articulated properly, and that unbiased observers of the speech would unanimously agree on what text is intended and indeed uttered. (Thus, when evaluating the success of ASR, it is unreasonable to expect a machine to divine the intention of a speaker that is improperly uttering speech, although it is reasonable to expect ASR to handle normal utterances that have typical disfluencies such as hesitations and frequent occurrences of "you know.") In this application of ASR, performance is simply evaluated by some measure of the percentage of words correctly recognized, although other factors are important (e.g., cost, efficiency).

Many ASR applications need a transcription of speech into text (e.g., dictating a business letter), yet others only require understanding of the speaker's intention. The latter does not need to recognize all words uttered, but only a minority of the words spoken, specifically the "key words" [107]. Many human–computer dialogues are designed to keep user responses very short and, thus, pertinent. Users are typically asked a series of very specific questions, whose responses should be only a few words, in an attempt to minimize use of extraneous words (e.g., *please*), which add pleasantries to human conversations, but lower performance in ASR. Nonetheless, dialogue users often employ many extraneous words [284].

Use of *word spotting* in ASR allows users to speak more naturally (i.e., with extra words and less restrictions on phrasing). ASR can, thus, employ a small vocabulary containing only keywords, although it is often useful to augment the small task-specific vocabulary with some general words (e.g., common short words found in most dialogues). It is assumed that, for a given ASR task, the spoken message can be deciphered based on recognition of the keywords alone. Spotting, thus, treats all nonkeyword speech as (expected) OOV words.

The simplest approach to word-spotting is to apply a word HMM (or chain of phonemic HMMs) for each keyword to the entire test utterance, shifting a few frames for each successive analysis in time [284]. If a section of speech matches a keyword (in terms of exceeding a threshold of probability), that word is declared "spotted" there. The size of the frame shift between evaluations should be small enough not to miss the start of a keyword, yet big enough to minimize the search computation.

Spotting can be accomplished with general ASR systems, if we employ *filler* or *garbage* models to handle the nonkeyword speech [101]. We formulate one or more HMMs to handle all OOV words, rather than design large numbers of complex HMMs for the many nonkeywords. The filler models are generally simpler than keyword models (e.g., fewer HMM states) [69], [131], [178].

For general ASR, the success criterion is usually the percentage of words correctly identified [i.e., minimal word error rate (WER)]. When the system instead makes a binary decision (e.g., to accept a hypothesis, or to decide whether a word is a keyword), the evaluation criterion has a second component. There are two posible errors: a *false alarm*—speech incorrectly identified as a keyword—and a *false reject*—missing a keyword. We minimize both, but applications assign different costs to the two types of error.

One performance measure is the keyword detection rate for a given false alarm rate (the latter being typically 10/hour). Another measure, *figure-of-merit*, is the average percentage of keywords found before each false alarm [131].

If we extend the idea of keyword spotting to handle more words or longer phrases, we have the general problem of *utterance verification* [173], i.e., to identify a section of speech as a string of accepted (key) words or as a rejected OOV word (or words) [236], [272]. A *confidence measure* can be associated with the likelihood that a given word is OOV [137]; with such a measure, the speaker can be notified that results are less reliable, allowing him to repeat an utterance. Subword units are often used to decipher OOV speech, as well as to efficiently spot words [55].

IV. Speech Synthesis

Speech synthesis is the automatic generation of a speech signal, starting typically from a phonetic transcription and its associated prosody. It is founded on a database generated from previously analyzed digital speech data. Such waveform synthesis is usually the final step of a larger TTS system, which accepts a normal textual input. TTS first does linguistic processing, including "letter-to-sound" conversion to generate the phonetic transcription that corresponds to the input text, and a prosody or intonation generation module. These act as a *front-end* processor for the TTS, as acoustical analysis forms the first part of ASR [21].

The simplest type of TTS is a basic table lookup approach, which is impossible for ASR, but which works for many TTS applications, owing to the much reduced dimensionality of text (versus speech) input. Previously recorded (and coded) speech is just played back, from a corpus where each waveform is labeled with the text used by the speaker who recorded the stored speech. For many applications, the spoken messages are small in number: a person with a pleasant voice records a few dozen messages. To synthesize long digit strings, one exploits a standard synthesis technique of concatenating short sections of speech (from a small database, e.g., of ten single digits) to form longer numbers (e.g., 16-digit credit card numbers or 10-digit phone numbers). The resulting quality is lower than with simple playback, however. So we must try to maintain speech quality as the task grows from uttering a few dozen phrases to handling any given text.

It is sometimes said that TTS simulates the VT, while ASR models the ear. On the other hand, TTS design should account for auditory precision, since TTS output is destined for the ear. Similarly, ASR should not use speech features so detailed as to exceed humans' ability to control their VTs. Hence, the tasks of TTS and ASR are not simple inverses of one another. Both should exploit knowledge about human speech production and perception [9], but computer simulations need not simulate these processes directly. As is often said, an airplane need not flap its wings like a bird to fly (it uses superior engine power for the needed thrust); it must nonetheless follow the basic aerodynamics of wing design. In an example closer to the AI tasks at hand, a chess-playing program need not follow the thought processes of a grandmaster, and instead take advantage of superior memory and computation power; it must nonetheless "understand" the basic rules (of chess, or in our case, speech production) in order to structure its search algorithm for success.

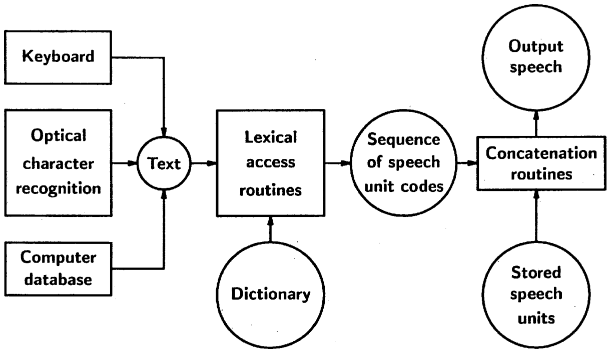

Fig. 8. Steps in speech synthesis.

Unlike ASR, evaluation of TTS requires subjective tests. ASR relies on WER and related objective measures of success, sometimes including cost weighting. Since TTS output enters the human ear in all applications, we must find efficient ways to measure success. It is tempting to use the evaluation methods for speech coders, as their output too is for human listening. However, speech coding has access to the original speech signal (as input), thus allowing a direct comparison between input and output; i.e., high bit-rate coders generally measure success in terms of a faithful reproduction (e.g., SNR, where the "noise" is the simple difference between the input and output). For TTS, the input is usually a text. Thus, lacking a target utterance for comparison, objective measures are rare. On the other hand, low-bit-rate speech coders often cannot use objective measures, and indeed both they and TTS usually rely on human listeners for subjective evaluations. As in coding, the quality of the output synthetic speech is measured in terms of intelligibility and naturalness [172].

Modern TTS is less concerned with intelligibility, as it often closely approaches human speech in comprehension. Synthetic speech from unlimited text, however, usually differs significantly from natural speech, making it easy for human listeners to identify the lower quality synthetic voice. The major challenge for TTS research is to better approximate the naturalness of human speech while minimizing costs (computer memory, computation, human training, delays) [39].

A. Basic Concatenation Approaches

Synthetic speech is generated by a concatenation of speech units stored in a reference database [8] (see Fig. 8). The challenge lies in the choice of units (in a training stage) and how they merge together to form a coherent utterance (at runtime). Until recently, TTS was principally done by storing a spectral template (spectral features, as a function of time, as in ASR, but with different features) for each short sound (usually a phoneme), retrieving templates as needed to produce an utterance, adjusting their values dynamically in time to simulate coarticulation, and finally driving a dynamic digital filter (which simulates the spectral effects of the VT) with an excitation. Recently, a simpler approach has become popular, where stored speech waveforms (not spectral patterns) of various durations are concatenated as needed. This latter method eliminates the need for filtering; its use of waveforms also renders any coarticulation modeling difficult. Signal adjustments at the unit boundaries are made in both cases, but adjustments in the second approach are more pragmatic (largely done to accomplish perceptual smoothness), rather than the spectral adjustments of the former method (founded in speech production theory).

The waveform method for TTS has origins decades ago, but was rejected when computers became powerful enough to support spectral smoothing in real time. In the 1970s and 1980s, computers could generate good quality TTS, but limited memories led to speech unit corpora too small to support waveform concatenation. The latter requires storage of very large numbers of speech units, much more so than with the spectral method. Since smoothing waveforms is generally simpler than spectral smoothing, the resulting speech was very discontinuous. It was, thus, natural to revisit such methods as computer costs declined significantly in the last decade.

Nonetheless, to see the compromises of TTS, consider first a system that minimizes cost. Such a TTS system forms synthetic speech from phonemic patterns. A choice of phonemes as units gives great flexibility and economy for TTS, since most languages have fewer than 40. The major disadvantage of concatenating such brief sounds is coarticulation, which causes large changes in the articulation and acoustics of phonemes, usually on a scale of one to three phonemes (i.e., coarticulation effects rarely extend beyond three phonemes; e.g., rounded lips during /s/ in "strew"). Coarticulation is an issue for units of any size, but with units of words or phrases, "joins" are much fewer than with phonemic units. Many intonational changes also accompany concatenations, and they too become more complicated as speech unit size shrinks.

Thus, small-unit TTS requires minimal storage, but is generally complicated and provides less natural output speech than systems using larger units. On the other hand, extremely long units, while having excellent quality, are not feasible due to excessive storage requirements. TTS quality is often proportional to the size of its database or memory, because it includes a greater diversity of stored human speech (with natural intonation and coarticulation). As unit size increases, storage often grows exponentially because the number of units increases as well.

B. *Choice of Units*

For maximal flexibility and economy, small units are preferred. In addition to phoneme units, words and diphones are also popular TTS units [196]. While diphones have the same duration as phonemes, their storage is much greater: a language with N phonemes has N^2 diphones and diphones have inherently more dynamic behavior, requiring storage of more frames/unit than phonemes (many of the latter are steady-state units, needing only one frame to represent each). A database of diphone units is still very feasible, needing only a few thousand frames of spectral data.

Syllables have occasionally been suggested as TTS units [73], but their disadvantages normally outweigh advantages. English needs about 10 000 syllables to be able to form all words. Assuming about 10 frames/syllable, it multiplies the storage substantially, yet a large percentage of phonemes still appear at unit boundaries, where coarticulation needs accurate simulation.

Consider now a word database for TTS, supposing that English has 50 000 common words (most speakers use only a few thousand, but vocabularies vary greatly for different speakers), with the average word having several phonemes, and the typical word lasting less than a second. With efficient coding methods of about 4 kb/s (e.g., CELP), a 200-Mb storage is required for unlimited vocabulary TTS when using word units. Of course, TTS limited to a small number of words or phrases could easily tolerate the storage needed here.

When the TTS unit is smaller than a syllable, an unlimited vocabulary is easily handled (i.e., all words can be generated from a universal phoneme or diphone database). For larger units, storage requirements increase quickly, and compromises are often made on vocabulary size. For TTS using word or larger units, storage is not the only TTS cost, however. Just as large-vocabulary SD ASR has training difficulties related to speaker fatigue, word-based TTS is difficult whenever the training speech exceeds a few thousands of words. Unlike ASR, however, fatigue is a more serious problem for TTS. One actually prefers a diversity of recording conditions for ASR, in order to capture and model variations. For TTS, on the other hand, all units must be pronounced uniformly; otherwise, the resulting synthetic speech (after segmentation and reconcatenation of the units) sounds very disjointed and discontinuous (e.g., as if two different people were uttering alternate sounds).

C. *Spectral Method for Speech Synthesis*

As in ASR, modern methods for TTS have not closely simulated human speech performance. Indeed, the trend in TTS is toward using less information about speech production. In theory, an elegant way to generate speech would be an articulatory approach [43], [220], [249]: an input text would incur muscle commands, which in turn create a temporal sequence of VT shapes, which are converted into digital filters, then excited by either noise or periodic pulses to form the speech. However, articulatory TTS has never produced good speech quality. Thus, commercial devices use either a spectral or waveform concatenation approach [176]. These latter systems are called *terminal-analog* synthesizers, because the synthetic speech is modeled on the terminal output of the VT, rather than on how speech is produced naturally.

1) Formant Synthesis: For most of the history of TTS (i.e., 1965–1995), the approach using a spectral envelope filter, driven by an excitation, has dominated. A set of parameters characterizing the short-time spectral envelope (VT filter) is stored for each of a number of units, e.g., for all phonemes or diphones. A simplified excitation is convolved with the impulse response of the VT filter, to yield the synthetic voice. The excitation is often either a periodic train of pulses (simulating glottal vibration) or white noise (as would

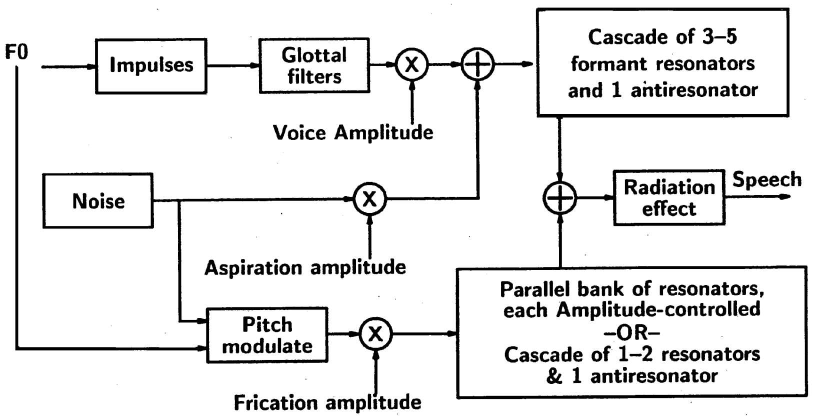
F0
Impulses
Glottal filters
X
Voice Amplitude
Cascade of 3–5 formant resonators and 1 antiresonator
Noise
X
Aspiration amplitude
Radiation effect
Speech
Pitch modulate
X
Frication amplitude
Parallel bank of resonators, each Amplitude-controlled –OR– Cascade of 1–2 resonators & 1 antiresonator

Fig. 9. Simplified organization of a formant synthesizer.

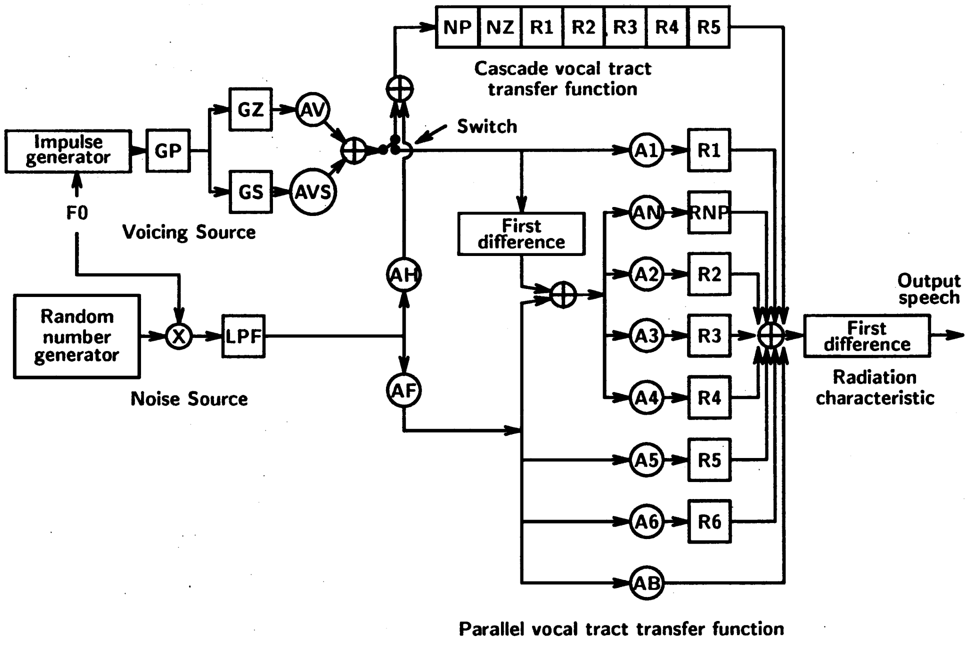

Fig. 10. Block diagram of a cascade-parallel formant synthesizer. Digital resonators are indicated by the prefix R and amplitude controls by the prefix A. Each resonator R_n has associated resonant frequency and bandwidth control parameters (after Klatt [146]).

be produced at a constriction in the VT). Typically, the filter is specified in terms of resonance (formant) center frequencies and bandwidths, over a frequency range up to approximately 5 kHz. (The TTS filter can be easily extended to higher frequencies, but these add little to the intelligibility of synthetic speech, and are only present if there is little cost to transmitting the digital synthetic speech at sample rates higher than 10 kHz.)

In formant-based speech synthesis, the four lowest center frequencies of the formants (i.e., $F1$ to $F4$) vary dynamically from frame-to-frame (to model coarticulation), along with the lowest three bandwidths ($B1$ to $B3$). The higher order parameters are usually kept fixed, as their variation has little perceptual effect (see Fig. 9). The classic formant approach to TTS is due to Klatt [147], where the VT filter has both a cascade and parallel structure of second-order filters, each of which simulates one resonance. A series or cascade structure is best for sonorant sounds, approximating their spectral envelopes well and allowing use of a single amplitude control. Fig. 10 displays one specific implementation that has been widely used, in part owing to the publication of actual program code [146]. This approach has both a cascade structure (for easy control of sonorants) and a parallel structure (for frication sounds). Second-order digital resonators are used in formant synthesis because higher order filters require additional bits (in their multiplier coefficients) to achieve the same spectral accuracy [211]. The synthesizer of Fig. 10 has 19 parameters that are updated for each 10-ms frame.

The sonorant phonemes that dominate speech, both in time and in energy, are excited at the glottis; thus, their filter models the entire VT. Most obstruents, on the other hand, excite a smaller portion of the VT, i.e., that anterior to the constriction where the obstruent noise is generated. Thus, for obstruents, the VT has higher frequency resonances and much less low-frequency energy. Using a cascade of resonators is awkward in such circumstances, as their parameters would have to abruptly change whenever speech shifted between sonorants and obstruents. It is more convenient to use a parallel set of second-order filters, with the same parameters as the formant filters of the cascade set, except for the amplitude controls. The cascade set has a single amplitude control, whereas each resonance in the parallel set is varied separately, allowing inclusion or not as needed. Fronted obstruents (e.g., /f, th/), with their short excited VT, raise the amplitude controls on only the highest-frequency resonators; rear fricatives (e.g., /sh/) excite lower resonators as well.

Switching between the cascade and parallel filter structures for voiced and unvoiced portions of TTS sometimes sounds as if two people were uttering the different sections in turn; i.e., like hissing noises (frication) superimposed on sonorants, rather than one integrated voice [138]. A parallel bank of filters could be used for sonorants as well, but then each formant amplitude must be individually specified [102]. Such amplitude control is essential for obstruents, but is unnecessary for sonorants. Another disadvantage of a parallel-only approach is that unintentional spectral zeros appear in such systems. This can be easily seen by considering the simple case of combining two single-pole systems $1/(z-a)$ and $1/(z-b)$: $1/(z-a)+1/(z-b)=(2z-a-b)/((z-a)(z-b))$. The resulting second-order system has the desired poles at the original locations a and b, but also an unwanted zero at $(a+b)/2$, which does not occur in a cascade of the two subfilters.

Among sonorant sounds, all but nasals have a similar formant structure: one formant per kilohertz on average (for an adult male). When the nasal tract is involved, the VT extends and adds an acoustic side-branch. As a result of the longer VT, nasals have one or two more resonances than other sonorants: a 21-cm VT length (versus 17 cm) leads to about 15% ($\approx 4/21$) reduced spacing between formants.

In addition to an extra formant or two (depending on the desired TTS bandwidth), nasals have spectral zeros (owing to the closed oral cavity), which other sonorants do not have. Thus, for 10-kHz synthesis, a basic cascade of five second-order resonators is used, with an extra resonator and an antiresonator in the cascade. Velar nasals, which have a relatively long closed oral cavity, have more than one spectral zero within the typical TTS bandwidth, but synthesizers usually model only one, as additional zeros have little perceptual importance. The retained zero and the extra pole (choosing a pole close in frequency to the zero) are often combined in a pole/zero second-order filter; the feature coefficients are set so that the zero and extra pole cancel for nonnasal sounds.

2) Noise Excitation Source: In LPC synthesizers (to be discussed next), the excitation is often binary: either periodic or noise, reflecting LPC's origins in speech coding, where determining further detail about the residual error signal is difficult. In contrast, formant TTS generally allows more types of excitation, exploiting its more complex architecture (versus a LPC synthesizer); e.g., excitation can be mixed (voiced and unvoiced) or from a codebook. In both formant and LPC TTS, the excitation is usually a periodic train of pulses to simulate voiced speech, and pseudorandom noise for unvoiced speech. Formant TTS allows noise to be modulated at the $F0$ rate, which is a better model for natural voiced frication.

Whereas the glottal pulses for voiced excitation decrease in intensity with frequency, the noise excitation for unvoiced sounds is better modeled by a flat spectrum. Another point to note is that the intensities of natural noise approximate a Gaussian distribution. TTS excitation noise samples usually come from a pseudorandom number generator that yields a flat spectrum with a uniform (non-Gaussian) distribution. However, summing several such random numbers can approximate a Gaussian noise sample (via the central limit theorem of probability) [146].

3) LPC Synthesis: Formant synthesis provides a flexible architecture for TTS, but requires that experts develop algorithms to manipulate all the synthesizer parameters as a function of the phoneme sequence to synthesize [143], [224]. Designers have never published details on how the parameters could be automatically extracted from natural speech. Formant estimators have had limited success, and formant TTS requires specification of many other values to model coarticulation.

Owing to this significant complexity and to the general preference for speech systems to be automatically trained, designers often prefer the automatic method of LPC. Another advantage for LPC is that its synthesis filter has a simpler structure than for formant TTS, because all details of the speech being modeled (except for intensity and periodicity, which present equal challenges to all TTS methods—see later) are included in the LPC filter coefficients. Unlike the second-order filters of formant TTS, a lattice filter is typically used for LPC because the multiplier (reflection) coefficients, which correlate with reflection percentages at boundaries between adjacent cylindrical sections in a model of the VT, can be linearly interpolated between successive frames. (Such parameter interpolation is common for TTS, to yield smoother speech, rather than simply change all parameters once per frame when the updated set is calculated.) The lattice method guarantees a stable filter. Direct-form digital filters are avoided in TTS, due to their sensitivity to quantization noise and risk of instability.

As with the waveform synthesizers discussed later, LPC is generally capable of only synthesizing the voice that trained the database. Thus, they are both inflexible in not allowing simple transformations of the stored data to simulate different voices. It is much easier to modify the parameters in formant TTS to simulate several synthetic voices [97], [142] than in other TTS methods; unfortunately, it is also harder to determine suitable parameters.

D. Waveform Methods for Speech Synthesis

Spectral methods dominated TTS until recently [205], [285], but the simplicity of directly concatenating units consisting of speech waveforms [87] forced a recent rejuvenation of this method, as computers became more able to store vast amounts of units. The spectral methods (principally formant and LPC) had the benefit of simulating VT behavior to a certain extent, but they too were just modeling the VT output, rather than its internal behavior. Rather than any theoretical objection, the reason that waveform approaches were abandoned earlier was the apparent need to store many units.

While modern waveform TTS requires much more computational resources than spectral approaches, the basic principles for the former are much simpler. Brief portions of actual speech waveforms are chained together. However, adjustments must be made to: 1) amplitude (scaling to levels appropriate for each unit in its broader sentential context); 2) pitch period (modifying $F0$ from its original stored value); and 3) other finer aspects of the stored unit. In spectral methods, the third category refers to coarticulation, but that requires more direct links to VT behavior than possible in the waveform method. Adjustments made to waveform units in lieu of coarticulation are less well justified by speech production theory than is smoothing in spectral methods.

The objective for all speech unit modifications is to transform each stored unit to correspond more closely to what they would be in the desired phonetic context in ordinary speech. Since humans normally control amplitude more globally than on phonemes (i.e., at higher linguistic levels such as syllables and words), intensity must be adjusted when concatenating subsyllable units. Phonemic units can be stored with an average amplitude, but level in normal speech varies significantly in complex ways with stress, speaking rate, and syntactic word type. Such amplitude adjustments need to be made for brief units in spectral TTS as well. $F0$ adjustments, on the other hand, are very different in the spectral and waveform approaches, since the spectral method explicitly controls $F0$ via the excitation. The waveform approach explicitly stores actual pitch periods. Thus, the durations of stored periods must be changed to accommodate varying $F0$, according to the needs of the synthetic utterance (e.g., to cue syntactic and semantic phenomena). The simplest way to change $F0$ is to truncate each period, removing its final few samples, if period shortening is desired (i.e., to raise $F0$), and to extend each period, if $F0$ must fall. The latter can be done by extrapolation of the final few samples, or by some other method interpolating with adjacent periods. Sometimes zero-amplitude speech samples are simply inserted at the end of each period to lengthen it.

When stored speech units are concatenated, the succession of units must sound continuous. Since the units are extracted (during training) from different speech signals, continuity (for both spectral amplitude and phase) at unit boundaries is not guaranteed when concatenating. The units are often chosen on spectral amplitude criteria, thus reducing continuity problems in that domain. Spectral phase is more difficult. Units often consist of, or terminate in, a complete pitch period. However, precise location of glottal closure instants (vocal cord closure causes the major VT excitation, and defines the start of a pitch period) is difficult, and small inaccuracies can lead to TTS that is perceived as harsh [270]. Manually labeled pitch periods (as in the original PSOLA method) are impractical for modern large TTS corpora [87]. Recent work with a *harmonic-plus-noise* model has explored simple ways to minimize phase mismatch [274].

1) TTS Databases or Corpora: The quality of synthetic speech is often proportional to the size and number of the speech units in the TTS database. If the units are complete phrases or sentences, the output quality resembles that of natural speech. However, such systems are very inflexible: each possible output phrase must be uttered during training by a single speaker with appropriate intonation. To accommodate large or unlimited vocabularies, TTS databases are stocked with a large number of units (corresponding typically to phonemes, allophones, diphones, and other subsyllable units). The recent trend for TTS has been to store many thousands of units, all automatically extracted from training speech [57], [61].

Database design has been a major area of recent TTS research. For decades, a database of a thousand diphone units appeared to be a limit for TTS, since it was difficult for any speaker to maintain uniform speaking style for more than a thousand units. One can, however, compensate for undesired intraspeaker variation in training speech by being selective in what gets stored as units, i.e., record the chosen speaker in many recording sessions, but select only desirable units. To make this feasible for large databases, the selection process must be automatic; earlier systems required human subjective evaluation.

The other key area of current TTS research is how to select the appropriate unit from a large corpus and what adjustments to perform at "join" boundaries. Joins are evaluated for optimality by a similarity or distance measure, as in ASR, but the objective for TTS is to minimize any discontinuities at the joins. When successive units for TTS come from diverse sources or recording times, the units have various degrees of mismatch at the boundaries (even for diphone or similar units that should theoretically align well). Unlike for ASR, TTS distance calculations must consider not only the spectral envelope, but phase continuity as well. Spectral measures may be simple distances using MFCCs as parameters, as well as measures more auditorily motivated, e.g., Klatt's weighted spectral slope measure [96], [229].

2) Automatic Unit Selection for Waveform TTS: In training, units are chosen for the database in terms of two costs to be minimized: that of choosing the proper speech unit at runtime (for a given phonetic context) and that of joining successive units. If poor units are retained in the database or if the units are too redundant to allow fits in varied environments, then poor TTS results. If the units are good, but the transitions are poor, the synthetic speech is excessively discontinuous [58].

The first task is more difficult, including both corpus design (which sentences to utter and which training speaker to use) and database design [106], [112]. When the synthetic speech is needed, the TTS algorithm examines many different sequences of eligible units from its database, calculating a cost for each proposed unit, in terms of the degree of mismatch from the desired characteristics for that unit, and a cost to make the join of the adjacent proposed units. There will generally be many combinations to test. As in ASR, the many possible combinations can require substantial computation in the search for the best sequence of units and the best way to join them.

As in ASR (reestimation), the database typically evolves during training. The cost of selecting units ("intrasegmental distortion") can be estimated by comparing each chosen unit from an initial version of the database against the actual unit observed in additional speech from the training speaker. The distance measure can involve both a spectral and prosodic match. If the distortion is excessive, the database is updated, e.g., by adding the new unit or by modifying the current set of units, so that the average distortion is reduced (this procedure is also similar to VQ codebook training for speech coders). The procedure is repeated for "intersegmental distortion," the acoustic mismatch caused by the required smoothing of units in the time frames adjacent to the join.

While we generally try to minimize all costs (i.e., acoustic distances), it is not clear which costs should predominate, i.e., whether an average cost should be the goal or one which minimizes a maximum error. Such questions are similar to the difference between SNR and segmental SNR, and to the minimization of total error energy in LPC analysis. A continual low distortion may be tolerable in TTS, whereas a few frames with large distortions could be unacceptable.

A recent example of such TTS training is *decision tree clustering*, where phonetic contexts for proposed units are chosen for their similar effects on the acoustic parameters of individual phonemes [25]. Decision trees are constructed without human intervention to maximize acoustic similarity within the selected classes. For example, in [59], the database was formed from a single speaker reading a popular novel for 1 h. Three-state (no-skip) CI HMMs were established for an extended set of a few hundred allophones (extending the usual allophone definition to include syllabic and stress effects). In comparison to the 23 million possible triphone HMMs, only 13 000 appeared in the hour of speech. This set was further reduced to 5700, by eliminating triphones with a low number of occurrences. All HMM state occurrences were discarded that either had duration or energy below 80% of average. Thus, in synthesis, time expansion of units would be limited and energy would not be excessively scaled upwards. The stretching issue is important because many units from read speech are short and need to be lengthened when synthesizing other texts. In addition, TTS is generally produced at rates slower than normal reading, to enhance intelligibility [225], [226], [279].

Precomputing and caching all possible join costs is impractical for large TTS corpora [22]. However, only a small subset of the theoretical search space is used in actual TTS. Thus, systems tend to synthesize large amounts of speech during training, to discover which units and joins are the most frequently examined; the most successful are cached for fast access at runtime. Dynamic programming makes the synthesis run efficiently.

E. Intonation

The automatic specification of an appropriate intonation for a given text is a challenge for any TTS system. ASR makes little use of intonation due to its complex functions, but TTS must attempt to produce appropriate intonation, as TTS output is judged by humans. Three dynamic *prosodic* (or *suprasegmental*) parameters contribute to intonation: pitch (or $F0$), duration, and amplitude [162], [215]. At the segmental (phoneme) level, speech amplitude varies greatly with manner of articulation (e.g., vowels being more intense than consonants), and stressed syllables are more intense than unstressed ones. Duration is also strongly linked to phonemes; e.g., vowels are longer than consonants, stressed syllables longer than unstressed ones, and consonants shorter in clusters [71], [72], [144], [195], [214]. When speaking rate is varied, vowels tend to change more in duration than consonants. $F0$ is the most complex intonation parameter. Lexically stressed syllables in emphasized words have large $F0$ changes, which also cue the boundaries of syntactic phrases.

Specifying a natural intonation is difficult. There is a lack of reliable indicators in input text to help specify most intonational effects. TTS inserts a pause for every clause-final punctuation (. ? ! : ;), but many sentences have no internal punctuation other than commas, and commas do not correspond well to intonation (e.g., "a large, round, red object" normally has no pauses). In many languages, an intonational "break" tends to occur after a (high-information) *content word* (e.g., noun, verb, adjective, or adverb) and preceding a *function word* (prepositions, articles, pronouns, etc.). Speakers highlight the final word in a sequence of content words (with lengthening and an $F0$ rise).

Traditionally, intonation is specified by application of rules (an expert-system approach), based on syntactic and semantic information that a TTS natural language processor (NLP) determines from the input sentences. However, deriving intonation directly and automatically from a training corpus of speech may become feasible [248], [280]. As in ASR, the advantage of such a data-driven procedure is that much more speech data can contribute than when phonetic experts interpret the data manually.

F. Text Processing for TTS

We examine the initial step of TTS last because it is the most distant from speech processing, just as language modeling is for ASR. Of course, as TTS is roughly a reverse procedure of ASR (with both differences and similarities, as we have seen), "front-end" text processing must be done initially in TTS, whereas language modeling may be viewed as a "back end" in ASR. The TTS input text is transformed to a representation allowing access to the stored units in the speech database, with additional controls for intonation. We must know the sequence of phonemes (or diphones or words) to pronounce, which syllables to stress [34], where to have intonational breaks, etc.

Until now, we have made little reference to variations in processing for different languages [266]. Obviously, language modeling in ASR must use words from the desired language, but most ASR techniques apply across languages with little change [7], [4], [23], [63], [85], [92], [156], [243], [139], [198], [171], [210], [208], [228]. TTS text processing is quite different in requiring language-dependent procedures, as the texts of different languages vary considerably, although certain NLP principles appear to be universal. Languages use different alphabets, and each language has its set of phonemes. Phoneticians have established a universal set of phonemes (e.g., as characterized by the International

Phonetic Alphabet), from which each language selects a subset. (There are however minor articulatory and acoustic differences across languages, e.g., in what constitutes an /i/ or an /s/.) Formant-based TTS can be relatively easily modified to a new language, by adjusting phoneme parameter targets, but LPC and waveform concatenation systems are less flexible.

The initial NLP step in TTS is the conversion of an input text into codes to access the database units. In the common case of phonemic units, this is called a text-to-phoneme (or letter-to-sound) mapping, usually via a table lookup. A TTS dictionary for a desired language, thus, has entries for all words, containing their pronunciation (including noting which syllables to stress), their syntactic category (part of speech), and sometimes semantic information. In addition, many systems have rules that predict the pronunciation of each letter in context; e.g., the letter “p” in English is pronounced /p/, except when preceding “h” (e.g., “pot” versus “telephone,” but note the exception “haphazard”). In certain languages (e.g., Korean, Spanish), this mapping is very easy because their written languages come directly from speech (one-to-one correspondence of letters to phonemes). Others (e.g., Italian, Finnish, German) are described by a very small set of rules. Yet other languages are more complex; e.g., English needs hundreds of letter-to-sound rules [62], and the thousands of symbols of Chinese need more. Nonetheless, the task is readily solved by phoneticians [222]. Errors in modern TTS systems seem to be limited to names and foreign words [11], [19].

It is common to search a dictionary for each word in an input text, and then apply rules for any word not found. The initial table lookup can provide data for intonation, which pronunciation rules often do not furnish. The rules are generally necessary owing to mistyped, new, and foreign words, which are normally not found in dictionaries.

V. Conclusion

This paper has examined the basic aspects of human–computer interactions via voice. The basic approaches and methods of ASR and synthesis have been discussed. Commercial synthesizers are widely available, but only for some of the world’s major languages. They provide speech whose intelligibility approaches 100% and whose naturalness is improving. Microprocessors can easily handle the computation speeds needed for many synthesizers, and many function entirely in software. However, the memory requirements of modern waveform concatenation systems strain the capacities of some practical systems [259].

Speech synthesis, in various forms, has existed for centuries (e.g., airflow mechanisms simulating the human VT, similar to musical instruments), but the development of computers made realistic synthetic speech possible. Increasingly inexpensive memory has led to use of large inventories of speech units, to overcome coarticulation problems. These trends follow similar ones in ASR, where stochastic methods involve simple automata, but large amounts of training data and the memory to accommodate great variability in the way speakers talk. In the future, a better understanding of how humans produce and perceive speech will yield yet more efficient speech synthesis and recognition, where a combination of stochastic- and knowledge-based methods will approach the performance of humans in automatic recognition and synthesis.

References

[1] A. Acero, *Acoustical and Environmental Robustness in Automatic Speech Recognition*. Boston, MA: Kluwer, 1993.

[2] A. Acero, L. Deng, T. Kristjansson, and J. Zhang, “HMM adaptation using vector Taylor series for noisy speech recognition,” in *Proc. ICSLP*, vol. 3, 2000, pp. 869–872.

[3] G. Adda, M. Adda-Decker, J.-L. Gauvin, and L. Lamel, “Text normalization and speech recognition in French,” in *Proc. Eurospeech*, 1997, pp. 2711–2714.

[4] S. Agrawal and K. Stevens, “Toward synthesis of Hindi consonants using Klsyn88,” in *Proc. ICSLP*, 1992, pp. 177–180.

[5] S. Ahadi and P. Woodland, “Combined Bayesian and predictive techniques for rapid speaker adaptation of continuous density hidden Markov models,” *Comput. Speech Lang.*, vol. 11, pp. 187–206, 1997.

[6] K. Aikawa, H. Singer, H. Kawahara, and Y. Tokhura, “Cepstral representation of speech motivated by time-frequency masking: An application to speech recognition,” *J. Acoust. Soc. Amer.*, vol. 100, pp. 603–614, 1996.

[7] E. Albano and A. Moreira, “Archisegment-based letter-to-phone conversion for concatenative speech synthesis in Portuguese,” in *Proc. ICSLP*, 1996, pp. 1708–1711.

[8] J. Allen, “Overview of text-to-speech systems,” in *Advances in Speech Signal Processing*, S. Furui and M. Sondhi, Eds. New York: Marcel Dekker, 1992, pp. 741–790.

[9] ——, “How do humans process and recognize speech?,” *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 567–577, 1994.

[10] L. Arslan and J. Hansen, “Selective training for hidden Markov models with applications to speech coding,” *IEEE Trans. Speech Audio Processing*, vol. 7, pp. 46–54, Oct. 1999.

[11] P. Bagshaw, “Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression,” *Comput. Speech Lang.*, vol. 12, pp. 119–142, 1998.

[12] L. Bahl, P. Brown, P. de Souza, R. Mercer, and M. Picheny, “A method for the construction of acoustic Markov models for words,” *IEEE Trans. Speech Audio Processing*, vol. 1, pp. 443–452, Oct. 1993.

[13] L. Bahl, P. Brown, P. de Souza, and R. Mercer, “Maximum mutual information estimation of hidden Markov model parameters for speech recognition,” *Proc. IEEE ICASSP*, pp. 49–52, 1986.

[14] L. Bahl, J. Bellegarda, P. de Souza, P. Gopalakrishnan, D. Nahamoo, and M. Picheny, “Multonic Markov word models for large vocabulary continuous speech recognition,” *IEEE Trans. Speech Audio Processing*, vol. 1, pp. 334–344, July 1993.

[15] L. Bahl and F. Jelinek, “Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition,” *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 404–411, July 1975.

[16] L. Bahl, F. Jelinek, and R. Mercer, “A maximum likelihood approach to continuous speech recognition,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 179–190, Mar. 1983.

[17] J. Baker, “The DRAGON system—an overview,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 24–29, Feb. 1975.

[18] L. E. Baum, “An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes,” *Inequalities*, vol. 3, pp. 1–8, 1972.

[19] K. Belhoula, “Rule-based grapheme-to-phoneme conversion of names,” in *Proc. Eurospeech*, 1993, pp. 881–884.

[20] J. Bellegarda, “Exploiting both local and global constraints for multi-span statistical language modeling,” in *Proc. IEEE ICASSP*, 1998, pp. 677–680.

[21] C. Benoit, G. Bailly, and T. Sawallis, Eds., *Talking Machines: Theories, Models and Applications*. Amsterdam, The Netherlands: North-Holland, 1992.

[22] M. Beutnagel, M. Mohri, and M. Riley, “Rapid unit selection from a large speech corpus for concatenative speech synthesis,” in *Proc. Eurospeech*, 1999, pp. 607–610.

[23] D. Bigorgne *et al.*, “Multilingual PSOLA text-to-speech system,” in *Proc. IEEE ICASSP*, vol. 2, 1993, pp. 187–190.

[24] J. Billa, K. Ma, J. McDonough, G. Zavaliagkos, and D. Miller, “Multilingual speech recognition: The 1996 Byblos Callhome system,” in *Proc. Eurospeech*, 1997, pp. 363–366.

[25] A. Black and P. Taylor, “Automatic clustering similar units for unit selection in speech synthesis,” in *Proc. Eurospeech*, 1997, pp. 601–604.

[26] H. Bourlard, H. Hermansky, and N. Morgan, “Toward increasing speech recognition error rates,” *Speech Commun.*, vol. 18, pp. 205–231, 1996.

[27] H. Bourlard, Y. Kamp, and C. Wellekens, “Speaker dependent connected speech recognition via phonemic Markov models,” in *Proc. IEEE ICASSP*, 1985, pp. 1213–1216.

[28] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Pacific Grove, CA: Wadsworth & Brooks, 1984.

[29] P. Brown, V. D. Pietra, P. deSouza, J. Lai, and R. Mercer, “Class-based n-gram models of natural language,” *Comput. Linguist.*, vol. 18, pp. 467–479, 1992.

[30] O. Cappé, C. Mokbel, D. Jouvet, and E. Moulines, “An algorithm for maximum likelihood estimation of hidden Markov models with unknown state-tying,” *IEEE Trans. Speech Audio Processing*, vol. 6, pp. 61–70, Jan. 1998.

[31] J. Chang and J. Glass, “Segmentation and modeling in segment-based recognition,” in *Proc. Eurospeech*, 1997, pp. 1199–1202.

[32] J. Chen and J. Goodman, “An empirical study of smoothing techniques for language modeling,” *Comput. Speech Lang.*, vol. 13, pp. 359–394, 1999.

[33] R. Chengalvarayan and L. Deng, “HMM-based speech recognition using state-dependent, discriminatively derived transforms on mel-warped DFT features,” *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 243–256, May 1997.

[34] K. Church, “Stress assignment in letter-to-sound rules for speech synthesis,” in *Proc. IEEE ICASSP*, 1986, pp. 2423–2426.

[35] ——, *Parsing in Speech Recognition*. Dordrecht, The Netherlands: Kluwer, 1987.

[36] T. Claes, J. Dologlou, L. ten Bosch, and D. van Compernolle, “A novel feature transformation for vocal tract length normalization in automatic speech recognition,” *IEEE Trans. Speech Audio Processing*, vol. 6, pp. 549–557, Nov. 1998.

[37] R. Cole *et al.*, “The challenge of spoken language systems: Research directions for the nineties,” *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 1–21, Jan. 1995.

[38] J.-T. Chien, C.-H. Lee, and H.-C. Wang, “A hybrid algorithm for speaker adaptation using MAP transformation and adaptation,” *IEEE Signal Processing Lett.*, vol. 4, pp. 167–169, June 1997.

[39] D. Childers and C. Lee, “Voice quality factors: Analysis, synthesis & perception,” *J. Acoust. Soc. Amer.*, vol. 90, pp. 2394–2410, 1991.

[40] W. Chou, C.-H. Lee, B.-H. Juang, and F. Soong, “A minimum error rate pattern recognition approach to speech recognition,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 8, pp. 5–31, 1994.

[41] W. Chou, B. Juang, and C.-H. Lee, “Discriminant-function-based minimum recognition error rate pattern-recognition approach to speech recognition,” *Proc. IEEE*, vol. 88, pp. 1201–1223, Aug. 2000.

[42] R. Christiansen and C. Rushforth, “Detecting and locating key words in continuous speech using linear predictive coding,” *IEEE Trans. Speech Audio Processing*, vol. SAP-25, pp. 361–367, Oct. 1977.

[43] C. Coker, “A model of articulatory dynamics and control,” *Proc. IEEE*, vol. 64, pp. 452–460, Apr. 1976.

[44] J. R. Cohen, “Application of an auditory model to speech recognition,” *J. Acoust. Soc. Amer.*, vol. 85, no. 6, pp. 2623–2629, 1989.

[45] L. Cohen, *Time-Frequency Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[46] M. Cooke, A. Morris, and P. Green, “Missing data techniques for robust speech recognition,” in *Proc. IEEE ICASSP*, 1997, pp. 863–866.

[47] H. Cung and Y. Normandin, “MMIE training of large vocabulary recognition systems,” *Speech Commun.*, vol. 22, pp. 303–314, 1997.

[48] S. Das, “Some experiments in discrete utterance recognition,” *IEEE Trans. Speech Audio Processing*, vol. SAP-30, pp. 766–770, Oct. 1982.

[49] S. Das, D. Nix, and M. Picheny, “Improvements in children’s speech recognition performance,” in *Proc. IEEE ICASSP*, 1998, pp. 433–436.

[50] B. Dautrich, L. Rabiner, and T. Martin, “On the effects of varying filter bank parameters on isolated word recognition,” *IEEE Trans. Speech Audio Processing*, vol. SAP-31, pp. 793–807, Aug. 1983.

[51] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition incontinuously spoken sentences,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 357–366, Aug. 1980.

[52] A. Dempster, N. Laird, and D. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *J. Royal Statist. Soc.*, vol. 39, pp. 1–88, 1977.

[53] V. Deng and M. Aksmanovik, “Speaker-independent phonetic classification using hidden Markov models with mixtures of trend functions,” *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 319–324, July 1997.

[54] L. Deng and R. Chengalvarayan, “A Markov model containing state-conditioned second-order nonstationarity: Application to speech recognition,” *Comput. Speech Lang.*, vol. 9, pp. 63–86, 1995.

[55] S. Dharanipragada and S. Roukos, “A new multistage algorithm for spotting new words in speech,” *IEEE Trans. Speech Audio Processing*, vol. 10, pp. 542–550, Nov. 2002.

[56] V. Digalakis and G. Neumeyer, “Speaker adaptation using combined transformation and Bayesian methods,” *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 294–300, July 1996.

[57] W. Ding and N. Campbell, “Optimizing unit selection with voice source and formants in the CHATR speech synthesis system,” in *Proc. Eurospeech*, 1997, pp. 537–540.

[58] R. Donovan, “A component by component listening test analysis of the IBM trainable speech synthesis system,” in *Proc. Eurospeech*, 2001, pp. 329–332.

[59] R. Donovan and P. Woodland, “A hidden Markov-model-based trainable speech synthesizer,” *Comput. Speech Lang.*, vol. 13, pp. 223–241, 1999.

[60] J. Droppo, L. Deng, and A. Acero, “Evaluation of the SPLICE algorithm on the Aurora2 database,” in *Proc. Eurospeech*, 2001, pp. 217–220.

[61] T. Dutoit, *From Text to Speech: A Concatenative Approach*. Boston, MA: Kluwer, 1997.

[62] H. S. Elovitz, R. Johnson, A. McHugh, and J. E. Shore, “Letter-to-sound rules for automatic translation of English text to phonetics,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 446–459, Dec. 1976.

[63] Y. El-Imam, “An unrestricted vocabulary Arabic speech synthesis system,” *IEEE Trans. Speech Audio Processing*, vol. 37, pp. 1829–1845, Dec. 1989.

[64] R. Elliott, L. Aggoun, and J. Moore, *Hidden Markov Models—Estimation and Control*. New York: Springer-Verlag, 1995.

[65] A. Erell and M. Weintraub, “Filterbank-energy estimation using mixture and Markov models for recognition of noisy speech,” *IEEE Trans. Speech Audio Processing*, vol. 1, pp. 68–76, Jan. 1993.

[66] G. Faucon and R. Le Bouquin-Jeannes, “Echo and noise reduction for hands-free terminals—state of the art,” in *Proc. Eurospeech*, 1997, pp. 2423–2426.

[67] J. Flanagan, *Speech Analysis, Synthesis and Perception*, 2nd ed. New York: Springer-Verlag, 1972.

[68] D. Fohr, J.-P. Haton, and Y. Laprie, “Knowledge-based techniques in acoustic-phonetic decoding of speech: Interest and limitations,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 8, pp. 133–153, 1994.

[69] J. Foote, S. Young, G. Jones, and K. Jones, “Unconstrained keyword spotting using phone lattices with application to spoken document retrieval,” *Comput. Speech Lang.*, vol. 11, pp. 207–224, 1997.

[70] G. D. Forney, “The Viterbi algorithm,” *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[71] D. Fry, “Duration and intensity as physical correlates of linguistic stress,” *J. Acoust. Soc. Amer.*, vol. 27, pp. 765–768, 1955.

[72] ——, “Experiments in the perception of stress,” *Lang. Speech*, vol. 1, pp. 126–152, 1958.

[73] O. Fujimura and J. Lovins, “Syllables as concatenative phonetic units,” in *Syllables and Segments*, A. Bell and J. Hooper, Eds. Amsterdam, The Netherlands: North-Holland, 1978, pp. 107–120.

[74] M. Gales, “Semi-tied covariance matrices,” in *Proc. IEEE ICASSP*, 1998, pp. 657–660.

[75] M. Gales, K. Knill, and S. Young, “State-based Gaussian selection in large vocabulary continuous speech recognition using HMMs,” *IEEE Trans. Speech Audio Processing*, vol. 7, pp. 152–161, Mar. 1999.

[76] M. Gales, “Cluster adaptive training of hidden Markov models,” *IEEE Trans. Speech Audio Processing*, vol. 8, no. 4, pp. 417–428, July 2000.

[77] ——, “Maximum likelihood linear transformations for HMM-based speech recognition,” *Comput. Speech Lang.*, vol. 12, pp. 75–98, 1998.

[78] ——, “Predictive model-based compensation schemes for robust speech recognition,” *Speech Commun.*, vol. 25, pp. 49–74, 1998.

[79] A. Ganapathiraju *et al.*, “Syllable—a promising recognition unit for LVCSR,” in *IEEE Workshop Speech Recognition*, 1997, pp. 207–213.

[80] Y. Gao and J.-P. Haton, “Noise reduction and speech recognition in noise conditions tested on LPNN-based continuous speech recognition system,” in *Proc. Eurospeech*, 1993, pp. 1035–1038.

[81] P. Garner and W. Holmes, “On the robust incorporation of formant features into hidden Markov models for automatic speech recognition,” in *Proc. IEEE ICASSP*, 1998, pp. 1–4.

[82] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.

[83] O. Ghitza, “Auditory nerve representation as a basis for speech processing,” in *Advances in Speech Signal Processing*, S. Furui and M. Sondhi, Eds. New York: Marcel Dekker, 1992, pp. 453–485.

[84] J. Godfrey, E. Holliman, and J. McDaniel, “Switchboard: Telephone speech corpus for research and development,” in *Proc. IEEE ICASSP*, vol. 1, 1992, pp. 517–520.

[85] B. Granstrom, P. Helgason, and H. Thráinsson, “The interaction of phonetics, phonology and morphology in an Icelandic text-to-speech system,” in *Proc. ICSLP*, 1992, pp. 185–188.

[86] J.-L. Gauvin and C.-H. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 291–298, Apr. 1994.

[87] E. George and M. Smith, “Speech analysis/synthesis and modification using an analysis-by-synthesis/overlap-add sinusoidal model,” *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 389–406, Sept. 1997.

[88] Y. Gong, “Speech recognition in noisy environments,” *Speech Commun.*, vol. 16, pp. 261–291, 1995.

[89] E. Gouvêa and R. Stern, “Speaker normalization through formant-based warping of the frequency scale,” in *Proc. Eurospeech*, 1997, pp. 1139–1142.

[90] R. M. Gray, A. Buzo, A. H. Gray, and Y. Matsuyama, “Distortion measures for speech processing,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 367–376, Aug. 1980.

[91] K. Greer, B. Lowerre, and L. Wilcox, “Acoustic pattern matching and beam searching,” in *Proc. IEEE ICASSP*, 1982, pp. 1251–1254.

[92] J. Gros, N. Pavesik, and F. Mihelic, “Speech timing in Slovenian TTS,” in *Proc. Eurospeech*, 1997, pp. 323–326.

[93] R. Haeb-Umbach and H. Ney, “Improvements in beam search for 10 000-word continuous speech recognition,” *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 353–356, Apr. 1994.

[94] J. Hansen and B. Womack, “Feature analysis and neural network-based classification of speech under stress,” *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 307–313, July 1996.

[95] J. Hansen and L. Arslan, “Robust feature-estimation and objective quality assessment for noisy speech recognition using the Credit Card Corpus,” *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 169–184, May 1995.

[96] J. Hansen and D. Chappell, “An auditory-based distortion measure with application to concatenative speech synthesis,” *IEEE Trans. Speech Audio Processing*, vol. 6, no. 5, pp. 489–495, Sept. 1998.

[97] H. Hanson, “Glottal characteristics of female speakers: Acoustic correlates,” *J. Acoust. Soc. Amer.*, vol. 101, pp. 466–481, 1997.

[98] P. Heeman and G. Damnati, “Deriving phrase-based language models,” in *IEEE Workshop Speech Recognition*, 1997, pp. 41–48.

[99] H. Hermansky and N. Morgan, “RASTA processing of speech,” *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 578–589, Oct. 1994.

[100] H. Hermansky, “Should recognizers have ears?,” *Speech Commun.*, vol. 25, pp. 3–27, 1998.

[101] A. Higgins and R. Wohlford, “Keyword recognition using template concatenation,” in *Proc. IEEE ICASSP*, 1985, pp. 1233–1236.

[102] J. Holmes, “Formant synthesizers—cascade or parallel?,” *Speech Comm.*, vol. 2, pp. 251–273, 1983.

[103] J. Holmes, W. Holmes, and P. Garner, “Using formant frequencies in speech recognition,” in *Proc. Eurospeech*, vol. 3, 1997, pp. 2083–2086.

[104] W. Holmes and M. Russell, “Probabilistic-trajectory segmental HMM’s,” *Comput. Speech Lang.*, vol. 13, pp. 3–27, 1999.

[105] H. Hon and K. Wang, “Unified frame and segment based models for automatic speech recognition,” in *Proc. IEEE ICASSP*, vol. 2, 2000, pp. 1017–1020.

[106] H. Hon, A. Acero, X. Huang, J. Liu, and M. Plumpe, “Automatic generation of synthesis units for trainable text-to-speech systems,” in *Proc. IEEE ICASSP*, 1998, pp. 273–276.

[107] E.-F. Huang, H.-C. Wang, and F. Soong, “A fast algorithm for large vocabulary keyword spotting application,” *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 449–452, July 1994.

[108] X. Huang, A. Acero, and S. Hon, *Spoken Language Processing*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[109] X. Huang and K.-F. Lee, “On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition,” *IEEE Trans. Speech Audio Processing*, vol. 1, pp. 150–157, Apr. 1993.

[110] X. Huang, Y. Ariki, and M. Jack, *Hidden Markov Models for Speech Recognition*. Edinburgh, U.K.: Edinburgh Univ. Press, 1990.

[111] X. Huang, H. Hon, M. Hwang, and K. Lee, “A comparative study of discrete, semicontinuous, and continuous hidden Markov models,” *Comput. Speech Lang.*, vol. 7, pp. 359–368, 1993.

[112] A. Hunt and W. Black, “Unit selection in a concatenative speech synthesis system using a large speech database,” in *Proc. IEEE ICASSP*, 1996, pp. 373–376.

[113] M. Hunt and C. Lefèbvre, “A comparison of several acoustic representations for speech recognition with degraded and undegraded speech,” in *Proc. IEEE ICASSP*, 1989, pp. 262–265.

[114] M. Hunt, M. Lennig, and P. Mermelstein, “Use of dynamic programming in a syllable-based continuous speech recognition system,” in *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, D. Sankoff and J. Kruskall, Eds. Reading, MA: Addison-Wesley, 1983, pp. 163–187.

[115] Q. Huo and C.-H. Lee, “A Bayesian predictive classification approach to robust speech recognition,” *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 200–204, Mar. 2000.

[116] M.-Y. Hwang, X. Huang, and F. Alleva, “Predicting unseen triphones with senones,” *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 412–419, Nov. 1996.

[117] S. Young, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[118] R. Iyer and M. Ostendorf, “Modeling long distance dependence in language: Topic mixtures versus dynamic cache models,” *IEEE Trans. Speech Audio Processing*, vol. 7, pp. 30–39, Jan. 1999.

[119] R. Iyer, M. Ostendorf, and H. Gish, “Using out-of-domain data to improve in-domain language models,” *IEEE Signal Processing Lett.*, vol. 4, pp. 221–223, 1997.

[120] C. Jankowski, H.-D. Vo, and R. Lippmann, “A comparison of signal processing front ends for automatic word recognition,” *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 286–293, July 1995.

[121] F. Jelinek, “Continuous speech recognition by statistical methods,” *Proc. IEEE*, vol. 64, pp. 532–556, Apr. 1976.

[122] ——, “The development of an experimental discrete dictation recognizer,” *Proc. IEEE*, vol. 73, pp. 1616–1620, Nov. 1985.

[123] F. Jelinek, R. Mercer, and S. Roucos, “Principles of lexical language-modeling for speech recognition,” in *Advances in Speech Signal Processing*, S. Furui and M. Sondhi, Eds. New York: Marcel Dekker, 1992, pp. 651–699.

[124] H. Jiang, K. Hirose, and Q. Huo, “Robust speech recognition based on Bayesian prediction approach,” *IEEE Trans. Speech Audio Processing*, vol. 7, pp. 426–440, July 1999.

[125] H. Jiang and L. Deng, “A robust compensation strategy against extraneous acoustic variations in spontaneous speech recognition,” *IEEE Trans. Speech Audio Processing*, vol. 10, pp. 9–17, Jan. 2002.

[126] H. Jin, S. Matsoukas, R. Schwartz, and F. Kubala, “Fast robust inverse transform speaker adapted training using diagonal transformations,” in *Proc. IEEE ICASSP*, 1998, pp. 785–788.

[127] B.-H. Juang and L. Rabiner, “Mixture autoregressive hidden Markov models for speech signals,” *IEEE Trans. Speech Audio Processing*, vol. SAP-33, pp. 1404–1413, Dec. 1985.

[128] B.-H. Juang, W. Chou, and C.-H. Lee, “Minimum classification error rate methods for speech recognition,” *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 257–265, May 1997.

[129] J.-C. Junqua, “The Lombard reflex and its role on human listeners and automatic speech recognizers,” *J. Acoust. Soc. Amer.*, vol. 93, pp. 510–524, 1993.

[130] J.-C. Junqua, H. Wakita, and H. Hermansky, “Evaluation and optimization of perceptually-based ASR front end,” *IEEE Trans. Speech Audio Processing*, vol. 1, pp. 39–48, Jan. 1993.

[131] J.-C. Junqua and J.-P. Haton, *Robustness in Automatic Speech Recognition*. Boston, MA: Kluwer, 1996.

[132] D. Jurafsky and F. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 2000.

[133] T. Kaneko and N. R. Dixon, "A hierarchical decision approach to large-vocabulary discrete utterance recognition," *IEEE Trans. Speech Audio Processing*, vol. SAP-31, pp. 1061–1072, Oct. 1983.

[134] G. Kaplan and E. Lerner, "Realism in synthetic speech," *IEEE Spectrum*, vol. 22, pp. 32–37, Apr. 1985.

[135] S. Katagiri, B. Juang, and C. Lee, "Pattern recognition using a family of design algorithms based upon the generalized probabilistic descent method," *Proc. IEEE*, vol. 86, pp. 2345–2373, Nov. 1998.

[136] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 400–401, Mar. 1987.

[137] T. Kawahara, C.-H. Lee, and B.-H. Juang, "Flexible speech understanding based on combined key-phrase detection and verification," *IEEE Trans. Speech Audio Processing*, vol. 6, pp. 558–568, Nov. 1998.

[138] J. Kerkhoff and L. Boves, "Designing control rules for a serial pole-zero vocal tract model," in *Proc. Eurospeech*, 1993, pp. 893–896.

[139] S.-H. Kim and J.-Y. Kim, "Efficient method of establishing words tone dictionary for Korean TTS system," in *Proc. Eurospeech*, 1997, pp. 247–250.

[140] S. King, T. Stephenson, S. Isard, P. Taylor, and A. Strachan, "Speech recognition via phonetically featured syllables," in *Proc. ICSLP*, vol. 1, 1998, pp. 1031–1034.

[141] B. Kingsbury, N. Morgan, and S. Greenberg, "Robust speech recognition using the modulation spectrogram," *Speech Commun.*, pp. 25, 117–132, 1998.

[142] D. Klatt and L. Klatt, "Analysis, synthesis, and perception of voice quality variations among female and male talkers," *J. Acoust. Soc. Amer.*, vol. 87, pp. 820–857, 1990.

[143] D. Klatt, "Structure of a phonological rule component for a synthesis-by-rule program," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 391–398, Oct. 1976.

[144] ——, "Linguistic uses of segmental duration in English: Acoustic and perceptual evidence," *J. Acoust. Soc. Amer.*, vol. 59, pp. 1208–1221, 1976.

[145] ——, "Review of the ARPA speech understanding project," *J. Acoust. Soc. Amer.*, vol. 62, pp. 1345–1366, 1977.

[146] ——, "Software for a cascade/parallel formant synthesizer," *J. Acoust. Soc. Amer.*, vol. 67, pp. 971–995, 1980.

[147] ——, "Review of text-to-speech conversion for English," *J. Acoust. Soc. Amer.*, vol. 82, pp. 737–793, 1987.

[148] G. Kopec and M. Bush, "Network-based isolated digit recognition using vector quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 850–867, Aug. 1985.

[149] T. Kosaka, S. Matsunaga, and S. Sagayama, "Speaker-independent speech recognition based on tree-structured speaker clustering," *Comput. Speech Lang.*, vol. 10, pp. 55–74, 1996.

[150] J. Kreiman, "Speaker modeling for speaker adaptation in automatic speech recognition," in *Talker Variability in Speech Processing*, K. Johnson and J. Mullennix, Eds. San Diego, CA: Academic, 1997, pp. 167–189.

[151] R. Kuhn and R. de Mori, "A cache-based natural language model for speech recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 570–583, June 1990.

[152] R. Kuhn *et al.*, "Eigenvoices for speaker adaptation," in *Proc. ICSLP*, 1998, pp. 1771–1774.

[153] A. Lazaridès, Y. Normandin, and R. Kuhn, "Improving decision trees for acoustic modeling," in *Proc. ICSLP*, 1996, pp. 1053–1056.

[154] B. Le Goff, T. Guiard-Marigny, M. Cohen, and C. Benoit, "Real-time analysis-synthesis and intelligibility of talking faces," in *ESCA Workshop*, 1994, pp. 53–56.

[155] W. Lea, "Speech recognition: Past, present, and future," in *Trends in Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1980, pp. 39–98.

[156] L. Lee, C. Tseng, and C.-J. Hsieh, "Improved tone concatenation rules in a formant-based Chinese text-to-speech system," *IEEE Trans. Speech Audio Processing*, vol. 1, pp. 287–294, July 1993.

[157] C.-H. Lee, "On stochastic feature and model compensation approaches for robust speech recognition," *Speech Commun.*, vol. 25, pp. 29–47, 1998.

[158] C.-H. Lee, F. Soong, and K. Paliwal, Eds., *Automatic Speech and Speaker Recognition-Advanced Topics*. Boston, MA: Kluwer , 1996.

[159] K.-F. Lee, *Automatic Speech Recognition: The Development of the SPHINX System*. Boston, MA: Kluwer, 1989.

[160] C. Leggetter and P. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Comput. Speech Lang.*, pp. 9, 171–185, 1995.

[161] ——, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Comput. Speech Lang.*, vol. 9, pp. 171–185, 1995.

[162] I. Lehiste, *Suprasegmentals*. Cambridge, MA: MIT Press, 1970.

[163] S. Levinson, "Structural methods in automatic speech recognition," *Proc. IEEE*, vol. 73, pp. 1625–1650, Nov. 1985.

[164] ——, "Continuously variable duration hidden Markov models for speech analysis," in *Proc. IEEE ICASSP*, 1986, pp. 1241–1244.

[165] C.-T. Lin, H.-W. Nein, and J.-Y. Hwu, "GA-based noisy speech recognition using two-dimensional cepstrum," *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 664–675, Nov. 2000.

[166] Q. Lin and C. Che, "Normalizing the vocal tract length for speaker independent speech recognition," *IEEE Signal Processing Lett.*, vol. 2, pp. 201–203, 1995.

[167] L. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 729–734, Sept. 1982.

[168] R. Lippmann, "Speech recognition by humans and machines," *Speech Commun.*, vol. 22, pp. 1–15, 1997.

[169] R. Lippmann and B. Carlson, "A Robust speech recognition with time-varying filtering, interruptions, and noise," in *IEEE Workshop Speech Recognition*, 1997, pp. 365–372.

[170] A. Ljolje, "High accuracy phone recognition using context-clustering and quasitriphonic models," *Comput. Speech Lang.*, vol. 8, pp. 129–151, 1994.

[171] M. Ljungqvist, A. Lindström, and K. Gustafson, "A new system for text-to-speech conversion, and its application to Swedish," in *Proc. ICSLP*, 1994, pp. 1779–1782.

[172] J. Logan, B. Greene, and D. Pisoni, "Segmental intelligibility of synthetic speech produced by rule," *J. Acoust. Soc. Amer.*, vol. 86, pp. 566–581, 1989.

[173] E. Lleida and P. Green, "Utterance verification in continuous speech recognition: Decoding and training procedures," *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 126–139, Mar. 2000.

[174] P. Loizou and A. Spanias, "High-performance alphabet recognition," *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 430–445, Nov. 1996.

[175] H.-L. Lou, "Implementing the Viterbi algorithm," *IEEE Signal Processing Mag.*, vol. 12, pp. 42–52, Sept. 1995.

[176] M. Macon and M. Clements, "Speech concatenation and synthesis using an overlap-add sinusoidal model," in *Proc. IEEE ICASSP*, 1996, pp. 361–364.

[177] J. Makhoul, "Linear prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561–580, Apr. 1975.

[178] A. Manos and V. Zue, "A segment-based word-spotter using phonetic filler models," in *Proc. IEEE ICASSP*, 1997, pp. 899–902.

[179] D. Mansour and B.-H. Juang, "A family of distortion measures based upon projection operation for robust speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1659–1671, Nov. 1989.

[180] J.-F. Mari, J.-P. Haton, and A. Kriouile, "Automatic word recognition based on second-order hidden Markov models," *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 22–25, Jan. 1997.

[181] J. Mariani, "Spoken language processing in multimodal communication," in *Proc. Int. Conf. Speech Processing*, 1997, pp. 3–12.

[182] ——, "Recent advances in speech processing," in *Proc. IEEE ICASSP*, 1989, pp. 429–440.

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

[184] S. Martin, J. Liermann, and H. Ney, "Algorithms for bigram and trigram word clustering," *Speech Commun.*, vol. 24, pp. 19–37, 1998.

[185] D. Massaro, *Perceiving Talking Faces*. Cambridge, MA: MIT Press, 1997.

[186] S. McCandless, "An algorithm for automatic formant extraction using linear prediction spectra," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 135–141, Apr. 1974.

[187] J. Ming, P. Jancovic, and F. J. Smith, "Robust speech recognition using probabilistic union models," *IEEE Trans. Speech Audio Processing*, vol. 10, pp. 403–414, Sept. 2002.

[188] N. Mirghafori and N. Morgan, "Transmissions and transitions: A study of two common assumptions in multi-band ASR," in *Proc. IEEE ICASSP*, 1998, pp. 713–716.

[189] C. Mitchell, M. Harper, L. Jamieson, and R. Helzerman, "A parallel implementation of a hidden Markov model with duration modeling for speech recognition," in *Dig. Signal Process.*, vol. 5, 1995, pp. 43–57.

[190] M. Mohri, "Finite-state transduceers in language and speech processing," *Comput. Linguist.*, vol. 23, pp. 269–312, 1997.

[191] C. Mokbel and G. Chollet, "Automatic word recognition in cars," *IEEE Trans. Speech Audio Processing*, vol. 3, pp. 346–356, Sept. 1995.
[192] T. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Mag.*, vol. 13, pp. 47–60, Nov. 1996.
[193] N. Morgan and H. Bourlard, "Continuous speech recognition," *IEEE Signal Processing Mag.*, vol. 12, pp. 25–42, May 1995.
[194] ——, "Neural networks for statistical recognition of continuous speech," *Proc. IEEE*, vol. 83, pp. 742–770, May 1995.
[195] J. Morton and W. Jassem, "Acoustic correlates of stress," *Lang. Speech*, vol. 8, pp. 159–181, 1965.
[196] E. Moulines and F. Charpentier, "Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Commun.*, vol. 9, pp. 453–467, 1990.
[197] I. Murray and J. Arnott, "Toward the simulation of emotion in synthetic speech: A review of the literature on human vocal emotion," *J. Acoust. Soc. Amer.*, vol. 93, pp. 1097–1108, 1993.
[198] I. Murray and M. Black, "A prototype text-to-speech system for Scottish Gaelic," in *Proc. Eurospeech*, 1993, pp. 885–887.
[199] S. Nakamura and K. Shikano, "Room acoustics and reverberation: Impact on hands-free recognition," in *Proc. Eurospeech*, 1997, pp. 2423–2426.
[200] H. Ney and S. Ortmanns, "Progress in dynamic programming search for LVCSR," *Proc. IEEE*, vol. 88, pp. 1224–1240, Aug. 2000.
[201] H. Ney, V. Steinbiss, R. Haeb-Umbach, B.-H. Tran, and U. Essen, "An overview of the Philips research system for large-vocabulary continuous-speech recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 8, pp. 33–70, 1994.
[202] N. Nilsson, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1980.
[203] N. Nocerino, F. Soong, L. Rabiner, and D. Klatt, "Comparative study of several distortion measures for speech recognition," in *IEEE Int. Conf. ASSP*, 1985, pp. 25–28.
[204] Y. Normandin, R. Cardin, and R. de Mori, "High-performance connected digit recognition using maximum mutual information estimation," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 299–311, Apr. 1994.
[205] D. O'Brien and A. I. C. Monaghan, "Concatenative synthesis based on a harmonic model," *IEEE Trans. Speech Audio Processing*, vol. 9, pp. 11–20, Jan. 2001.
[206] A. Oh and A. Rudnicky, "Stochastic natural language generation for spoken dialog systems," *Comput. Speech Lang.*, vol. 16, pp. 387–407, 2002.
[207] S. Okawa, E. Bocchieri, and A. Potamianos, "Multi-band speech recognition in noisy environments," in *Proc. IEEE ICASSP*, 1998, pp. 641–644.
[208] L. Oliveira, C. Viana, and I. Trancoso, "A rule-based text-to-speech system for Portuguese," in *Proc. IEEE ICASSP*, vol. 2, 1992, pp. 73–76.
[209] M. Omologo, P. Svaizer, and M. Matassoni, "Environmental conditions and acoustic transduction in hands-free robust speech recognition," *Speech Commun.*, vol. 25, pp. 75–95, 1998.
[210] Y. Ooyama, H. Asano, and K. Matsuoka, "Spoken style explanation generator for Japanese Kanji using a text-to-speech system," in *Proc. ICSLP*, 1996, pp. 1369–1372.
[211] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[212] S. Ortmanns and H. Ney, "The time-conditioned approach in dynamic programming search for LVCSR," *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 676–687, Nov. 2000.
[213] ——, "A word graph algorithm for large vocabulary continuous speech recognition," *Comput. Speech Lang.*, vol. 11, pp. 43–72, 1997.
[214] D. O'Shaughnessy, "Consonant durations in clusters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 282–295, Aug. 1974.
[215] ——, "Linguistic features in fundamental frequency patterns," *J. Phonetics*, vol. 7, pp. 119–145, 1979.
[216] ——, *Speech Communications: Human and Machine*. Piscataway, NJ: IEEE Press, 2000.
[217] M. Ostendorf, V. Digalakis, and O. Kimball, "From HMM's to segment models: A unified view of stochastic modeling for speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 360–378, Sept. 1996.
[218] M. Padmanabhan, L. Bahl, D. Nahamoo, and M. Picheny, "Speaker clustering and transformation for speaker adaptation in speech recognition systems," *IEEE Trans. Speech Audio Processing*, vol. 6, pp. 71–77, Jan. 1998.
[219] K. Paliwal, "Spectral subband centroids as features for speech recognition," in *IEEE Workshop Speech Recognition*, 1997, pp. 124–131.
[220] S. Parthasarathy and C. Coker, "On automatic estimation of articulatory parameters in a text-to-speech system," *Comput. Speech Lang.*, vol. 6, pp. 37–76, 1992.
[221] D. Paul, "Training of HMM recognizers by simulated annealing," in *Proc. IEEE ICASSP*, 1985, pp. 13–16.
[222] E. Pavlova, Y. Pavlov, R. Sproat, C. Shih, and P. van Santen, "Bell Laboratories Russian text-to-speech system," *Comput. Speech Lang.*, pp. 6, 37–76, 1997.
[223] J. Picone, "Signal modeling techniques in speech recognition," *Proc. IEEE*, vol. 81, pp. 1215–1247, Sept. 1993.
[224] N. Pinto, D. Childers, and A. Lalwani, "Formant speech synthesis: Improving production quality," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1870–1887, Dec. 1989.
[225] D. Pisoni, H. Nusbaum, and B. Greene, "Perception of synthetic speech generated by rule," *Proc. IEEE*, vol. 73, pp. 1665–1676, Nov. 1985.
[226] L. Pols, "Quality assessment of text-to-speech synthesis by rule," in *Advances in Speech Signal Processing*, S. Furui and M. Sondhi, Eds. New York: Marcel Dekker, 1992, pp. 387–416.
[227] A. Poritz, "Hidden Markov models: A guided tour," in *Proc. IEEE ICASSP*, 1988, pp. 7–13.
[228] T. Portele, F. Höfer, and W. Hess, "Structure and representation of an inventory for German speech synthesis," in *Proc. ICSLP*, 1994, pp. 1759–1762.
[229] S. Quackenbush, T. Barnwell, and M. Clements, *Objective Measures for Speech Quality*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[230] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, pp. 257–286, Feb. 1989.
[231] L. Rabiner, S. Levinson, and M. Sondhi, "On the application of vector quantization and hidden Markov models to speaker-independent, isolated word recognition," *Bell Syst. Tech. J.*, vol. 62, pp. 1075–1105, 1983.
[232] ——, "On the use of hidden Markov models for speaker-independent recognition of isolated words from a medium-size vocabulary," *AT&T Bell Labs Tech. J.*, vol. 63, pp. 627–641, 1984.
[233] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[234] M. Rahim, B.-H. Juang, W. Chou, and E. Buhrke, "Signal conditioning techniques for robust speech recognition," *IEEE Signal Processing Lett.*, vol. 3, pp. 107–109, Apr. 1996.
[235] M. Rahim and B.-H. Juang, "Signal bias removal by maximum likelihood estimation for robust telephone speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 19–30, Jan. 1996.
[236] M. Rahim, C.-H. Lee, and B.-H. Juang, "A study on robust utterance verification for connected digits recognition," *J. Acoust. Soc. Amer.*, vol. 101, pp. 2892–2902, 1997.
[237] A. Rao and K. Rose, "Deterministically annealed design of hidden Markov model speech recognizers," *IEEE Trans. Speech Audio Processing*, vol. 9, pp. 111–126, Feb. 2001.
[238] C. Rathinavelu and L. Deng, "A maximum a posteriori approach to speaker adaptation using the trended hidden Markov model," *IEEE Trans. Speech Audio Processing*, vol. 9, pp. 549–557, July 2001.
[239] R. Reddy, "Speech recognition by machine: A review," *Proc. IEEE*, vol. 64, pp. 501–531, Apr. 1976.
[240] W. Reichl and W. Chou, "Robust decision tree state tying for continuous speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 8, pp. 555–566, Sept. 2000.
[241] S. Renals, "Phone deactivation pruning in large vocabulary continuous speech recognition," *IEEE Signal Processing Lett.*, vol. 3, pp. 4–6, Jan. 1996.
[242] H. B. Richards and J. S. Bridle, "The HDM: A segmental hidden dynamic model of coarticulation," in *Proc. IEEE ICASSP*, vol. 1, 1999, pp. 357–360.
[243] T. Rietveld *et al.*, "Evaluation of speech synthesis systems for Dutch in telecommunication applications in GSM and PSTN networks," in *Proc. Eurospeech*, 1997, pp. 577–580.
[244] R. Rose, "Keyword detection in conversational speech utterances using hidden Markov model based continuous speech recognition," *Comput. Speech Lang.*, vol. 9, pp. 309–333, 1995.
[245] R. Rose, J. Schroeter, and M. Sondhi, "The potential role of speech production models in automatic speech recognition," *J. Acoust. Soc. Amer.*, vol. 99, no. 3, pp. 1699–1709, 1996.
[246] A. Rosenberg, L. Rabiner, J. Wilpon, and D. Kahn, "Demisyllable-based isolated word recognition systems," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 713–726, June 1983.
[247] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Comput. Speech Lang.*, vol. 10, pp. 187–228, 1996.

[248] K. Ross and M. Ostendorf, "A dynamical system model for generating fundamental frequency for speech synthesis," *IEEE Trans. Speech Audio Processing*, vol. 7, pp. 295–309, May 1999.

[249] P. Rubin, T. Baer, and P. Mermelstein, "An articulatory synthesizer for perceptual research," *J. Acoust. Soc. Amer.*, vol. 70, pp. 321–328, 1981.

[250] M. Russell and W. Holmes, "Linear trajectory segmental HMMs," *IEEE Signal Processing Lett.*, vol. 4, pp. 72–74, Mar. 1997.

[251] Y. Sagisaka, "Speech synthesis from text," *IEEE Commun. Mag.*, vol. 28, pp. 35–41, Jan. 1990.

[252] K. Samudravijaya, S. Singh, and P. Rao, "Pre-recognition measures of speaking rate," *Speech Commun.*, vol. 24, pp. 73–84, 1998.

[253] S. Sandhu and O. Ghitza, "A comparative study of mel cepstra and EIH for phone classification under adverse conditions," in *Proc. IEEE ICASSP*, 1995, pp. 409–412.

[254] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *Proc. IEEE ICASSP*, 2000, pp. 1129–1132.

[255] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," in *Proc. IEEE ICASSP*, 1997, pp. 1331–1334.

[256] J. Schroeter and M. Sondhi, "Techniques for estimating vocal tract shapes from the speech signal," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 133–150, Jan. 1994.

[257] R. Schwartz *et al.*, "Multipass search strategies," in *Automatic Speech and Speaker Recognition*, C.-H. Lee *et al.*, Eds. Boston, MA: Kluwer, 1995, ch. 18.

[258] S. Seneff, "TINA: A natural language system for spoken language applications," *Comput. Linguist.*, vol. 18, pp. 61–86, 1992.

[259] H. Sheikhzadeh, E. Cornu, R. Brennan, and T. Schneider, "Real-time speech synthesis on an ultra-low resource, programmable DSP system," in *Proc. IEEE ICASSP*, vol. 1, 2002, pp. 433–436.

[260] K. Shikano and F. Itakura, "Spectrum distance measures for speech recognition," in *Advances in Speech Signal Processing*, S. Furui and M. Sondhi, Eds. New York: Marcel Dekker, 1992, pp. 419–452.

[261] P. Silsbee and A. Bovik, "Computer lipreading for improved accuracy in automatic speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 337–351, Sept. 1996.

[262] O. Siohan, Y. Gong, and J.-P. Haton, "Comparative experiments of several adaptation approaches to noisy speech recognition using stochastic trajectory models," *Speech Commun.*, vol. 18, pp. 335–352, 1996.

[263] R. Sitaram and T. Sreenivas, "Incorporating phonetic properties in hidden Markov models for speech recognition," *J. Acoust. Soc. Amer.*, vol. 102, pp. 1149–1158, 1997.

[264] J. Smolders, T. Claes, G. Sablon, and D. van Campernolle, "On the importance of the microphone position for speech recognition in the car," in *Proc. IEEE ICASSP*, vol. 1, 1994, pp. 429–432.

[265] F. Soong and E.-F. Huang, "A tree-trellis based search for finding the N best sentence hypotheses in continuous speech recognition," in *Proc. IEEE ICASSP*, 1991, pp. 705–708.

[266] R. Sproat, *Multi-Lingual Text-to-Speech Synthesis: The Bell Labs Approach*. Boston, MA: Kluwer, 1998.

[267] V. Steinbiss *et al.*, "Continuous speech dictation—from theory to practice," *Speech Commun.*, vol. 17, pp. 19–38, 1995.

[268] B. Strope and A. Alwan, "A model of dynamic auditory perception and its application to robust word recognition," *IEEE Trans. Speech Audio Processing*, vol. 5, pp. 451–464, Sept. 1997.

[269] ——, "Robust word recognition using threaded spectral peaks," in *Proc. IEEE ICASSP*, 1998, pp. 625–628.

[270] Y. Stylianou, "Removing linear phase mismatches in concatenative speech synthesis," *IEEE Trans. Speech Audio Processing*, vol. 9, pp. 232–239, Mar. 2001.

[271] K.-Y. Su and C.-H. Lee, "Speech recognition using weighted HMM and subspace projection approaches," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 69–79, Jan. 1994.

[272] R. Sukkar and C.-H. Lee, "Vocabulary-independent discriminative utterance verification for nonkeyword rejection in subword based speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 4, pp. 420–429, Nov. 1996.

[273] T. Sullivan and R. Stern, "Multi-microphone correlation-based processing for robust speech recognition," in *Proc. IEEE ICASSP*, vol. 2, 1993, pp. 91–94.

[274] A. Syrdal, Y. Stylianou, A. Conkie, and J. Schroeter, "TD-PSOLA versus harmonic plus noise model in diphone based speech synthesis," in *Proc. IEEE ICASSP*, vol. 1, 1998, pp. 273–276.

[275] ——, "TD-PSOLA versus harmonic plus noise model in diphone based speech synthesis," in *Proc. IEEE ICASSP*, 1998, pp. 273–276.

[276] J. Takahashi and S. Sagayama, "Vector-field-smoothed Bayesian learning for fast and incremental speaker/telephone-channel adaptation," *Comput. Speech Lang.*, vol. 11, pp. 127–146, 1997.

[277] I. Trancoso, "An overview of different trends on CELP coding," in *Speech Recognition and Coding: New Advances and Trends*, A. J. Ayuso and J. M. Soler, Eds. New York: Springer-Verlag, 1995, pp. 351–368.

[278] S. Umesh, L. Cohen, N. Marinovic, and D. Nelson, "Scale transform in speech analysis," *IEEE Trans. Speech Audio Processing*, vol. 7, pp. 40–45, Jan. 1998.

[279] J. van Santen, "Perceptual experiments for diagnostic testing of text-to-speech systems," *Comput. Speech Lang.*, vol. 7, pp. 49–100, 1993.

[280] J. Véronis, P. di Cristo, F. Courtois, and C. Chaumette, "A stochastic model of intonation for text-to-speech synthesis," *Speech Commun.*, vol. 26, pp. 233–244, 1998.

[281] H. Wakita, "Normalization of vowels by vocal-tract length and its application to vowel identification," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 183–192, Apr. 1977.

[282] L. Welling, H. Ney, and S. Kanthak, "Speaker adaptive modeling by vocal tract normalization," *IEEE Trans. Speech Audio Processing*, vol. 10, pp. 415–426, Sept. 2002.

[283] L. Welling and H. Ney, "Formant estimation for speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 6, pp. 36–48, Jan. 1998.

[284] J. Wilpon, L. Rabiner, C.-H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 1870–1878, Nov. 1990.

[285] J. Wouters and M. Macon, "Control of spectral dynamics in concatenative speech synthesis," *IEEE Trans. Speech Audio Processing*, vol. 9, pp. 30–38, Jan. 2001.

[286] S. Young *et al.*, "Multilingual large vocabulary speech recognition: The European SQUALE project," *Comput. Speech Lang.*, vol. 11, pp. 73–89, 1997.

[287] S. Young and L. Chase, "Speech recognition evaluation: A review of the U.S. CSR and LVCSR programmes," *Comput. Speech Lang.*, vol. 12, pp. 263–279, 1998.

[288] Y. Zhao, "An acoustic-phonetic-based speaker-adaptation technique for improving speaker-independent continuous speech recognition," *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 380–394, July 1994.

[289] V. Zue, "The use of speech knowledge in automatic speech recognition," *Proc. IEEE*, vol. 73, pp. 1602–1615, Nov. 1985.

[290] ——, "Conversational interfaces: Advances and challenges," in *Proc. Eurospeech*, 1997, pp. KN-9–18.

[291] E. Zwicker, "Peripheral preprocessing in hearing and psychoacoustics as guidelines for speech recognition," in *Proc. Montreal Symp. Speech Recognition*, 1986, pp. 1–4.

[292] E. Zwicker, E. Terhardt, and E. Paulus, "Automatic speech recognition using psychoacoustic models," *J. Acoust. Soc. Amer.*, vol. 65, pp. 487–498, 1979.

Douglas O'Shaughnessy (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1972, 1972, and 1976, respectively.

Since 1977, he has been a Professor at INRS-Telecommunications, University of Quebec, Montreal, QC, Canada. For this same period, he has also taught as Adjunct Professor in the Department of Electrical Engineering, McGill University, Montreal, QC, Canada. He has worked as a teacher and researcher in the speech communication field for more than 20 years. He is the author of the textbook *Speech Communications: Human and Machine* (Piscataway, NJ: IEEE, 2000). Since 1998, he has been an Associate Editor of the *Journal of the Acoustical Society of America*. His research interests include automatic speech synthesis, analysis, coding, and recognition.

Dr. O'Shaughnessy is a Fellow of the Acoustical Society of America. He has been selected as the General Chair of the 2004 International Conference on Acoustics, Speech, and Signal Processing (ICASSP) in Montreal. From 1995 to 1999, he was an Associate Editor for the IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING.