Foundations and Trends® in
Signal Processing
Vol. 1, Nos. 1–2 (2007) 1–194
© 2007 L. R. Rabiner and R. W. Schafer
DOI: 10.1561/2000000001

now
the essence of knowledge

Introduction to Digital Speech Processing

Lawrence R. Rabiner[1] and Ronald W. Schafer[2]

[1] *Rutgers University and University of California, Santa Barbara, USA, rabiner@ece.ucsb.edu*
[2] *Hewlett-Packard Laboratories, Palo Alto, CA, USA*

Abstract

Since even before the time of Alexander Graham Bell's revolutionary invention, engineers and scientists have studied the phenomenon of speech communication with an eye on creating more efficient and effective systems of human-to-human and human-to-machine communication. Starting in the 1960s, digital signal processing (DSP), assumed a central role in speech studies, and today DSP is the key to realizing the fruits of the knowledge that has been gained through decades of research. Concomitant advances in integrated circuit technology and computer architecture have aligned to create a technological environment with virtually limitless opportunities for innovation in speech communication applications. In this text, we highlight the central role of DSP techniques in modern speech communication research and applications. We present a comprehensive overview of digital speech processing that ranges from the basic nature of the speech signal, through a variety of methods of representing speech in digital form, to applications in voice communication and automatic synthesis and recognition of speech. The breadth of this subject does not allow us to discuss any

aspect of speech processing to great depth; hence our goal is to provide a useful introduction to the wide range of important concepts that comprise the field of digital speech processing. A more comprehensive treatment will appear in the forthcoming book, *Theory and Application of Digital Speech Processing* [101].

1

Introduction

The fundamental purpose of speech is communication, i.e., the transmission of messages. According to Shannon's information theory [116], a message represented as a sequence of discrete symbols can be quantified by its *information content* in bits, and the rate of transmission of information is measured in bits/second (bps). In speech production, as well as in many human-engineered electronic communication systems, the information to be transmitted is encoded in the form of a continuously varying (analog) waveform that can be transmitted, recorded, manipulated, and ultimately decoded by a human listener. In the case of speech, the fundamental analog form of the message is an acoustic waveform, which we call the *speech signal*. Speech signals, as illustrated in Figure 1.1, can be converted to an electrical waveform by a microphone, further manipulated by both analog and digital signal processing, and then converted back to acoustic form by a loudspeaker, a telephone handset or headphone, as desired. This form of speech processing is, of course, the basis for Bell's telephone invention as well as today's multitude of devices for recording, transmitting, and manipulating speech and audio signals. Although Bell made his invention without knowing the fundamentals of information theory, these ideas

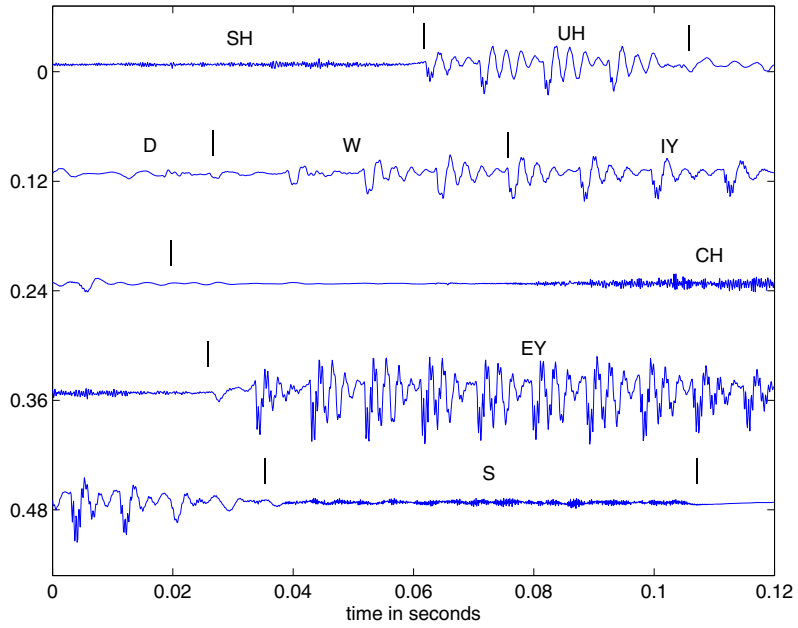

Fig. 1.1 A speech waveform with phonetic labels for the text message "Should we chase."

have assumed great importance in the design of sophisticated modern communications systems. Therefore, even though our main focus will be mostly on the speech waveform and its representation in the form of parametric models, it is nevertheless useful to begin with a discussion of how information is encoded in the speech waveform.

1.1 The Speech Chain

Figure 1.2 shows the complete process of producing and perceiving speech from the formulation of a message in the brain of a talker, to the creation of the speech signal, and finally to the understanding of the message by a listener. In their classic introduction to speech science, Denes and Pinson aptly referred to this process as the "speech chain" [29]. The process starts in the upper left as a message represented somehow in the brain of the speaker. The message information can be thought of as having a number of different representations during the process of speech production (the upper path in Figure 1.2).

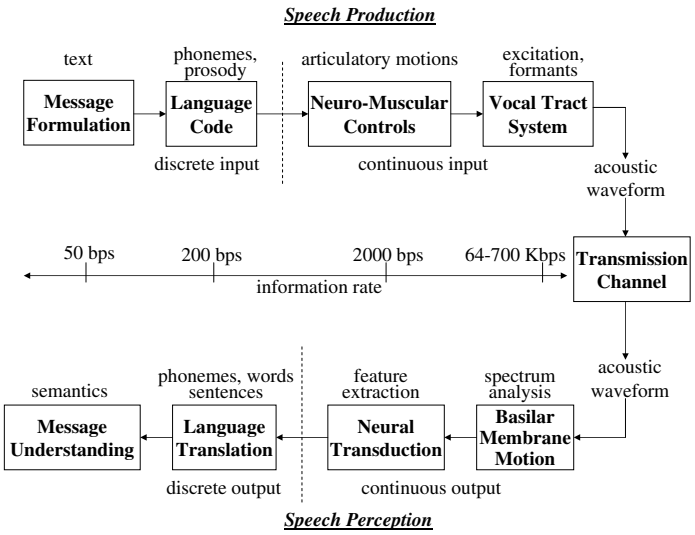

Fig. 1.2 The Speech Chain: from message, to speech signal, to understanding.

For example the message could be represented initially as English text. In order to "speak" the message, the talker implicitly converts the text into a symbolic representation of the sequence of sounds corresponding to the spoken version of the text. This step, called the language code generator in Figure 1.2, converts text symbols to phonetic symbols (along with stress and durational information) that describe the basic sounds of a spoken version of the message and the manner (i.e., the speed and emphasis) in which the sounds are intended to be produced. As an example, the segments of the waveform of Figure 1.1 are labeled with phonetic symbols using a computer-keyboard-friendly code called ARPAbet.[1] Thus, the text "should we chase" is represented phonetically (in ARPAbet symbols) as [SH UH D — W IY — CH EY S]. (See Chapter 2 for more discussion of phonetic transcription.) The third step in the speech production process is the conversion to "neuro-muscular controls," i.e., the set of control signals that direct the neuro-muscular system to move the speech articulators, namely the tongue, lips, teeth,

[1] The International Phonetic Association (IPA) provides a set of rules for phonetic transcription using an equivalent set of specialized symbols. The ARPAbet code does not require special fonts and is thus more convenient for computer applications.

jaw and velum, in a manner that is consistent with the sounds of the desired spoken message and with the desired degree of emphasis. The end result of the neuro-muscular controls step is a set of articulatory motions (continuous control) that cause the vocal tract articulators to move in a prescribed manner in order to create the desired sounds. Finally the last step in the Speech Production process is the "vocal tract system" that physically creates the necessary sound sources and the appropriate vocal tract shapes over time so as to create an acoustic waveform, such as the one shown in Figure 1.1, that encodes the information in the desired message into the speech signal.

To determine the rate of information flow during speech production, assume that there are about 32 symbols (letters) in the language (in English there are 26 letters, but if we include simple punctuation we get a count closer to $32 = 2^5$ symbols). Furthermore, the rate of speaking for most people is about 10 symbols per second (somewhat on the high side, but still acceptable for a rough information rate estimate). Hence, assuming independent letters as a simple approximation, we estimate the base information rate of the text message as about 50 bps (5 bits per symbol times 10 symbols per second). At the second stage of the process, where the text representation is converted into phonemes and prosody (e.g., pitch and stress) markers, the information rate is estimated to increase by a factor of 4 to about 200 bps. For example, the ARBAbet phonetic symbol set used to label the speech sounds in Figure 1.1 contains approximately $64 = 2^6$ symbols, or about 6 bits/phoneme (again a rough approximation assuming independence of phonemes). In Figure 1.1, there are 8 phonemes in approximately 600 ms. This leads to an estimate of $8 \times 6/0.6 = 80$ bps. Additional information required to describe prosodic features of the signal (e.g., duration, pitch, loudness) could easily add 100 bps to the total information rate for a message encoded as a speech signal.

The information representations for the first two stages in the speech chain are discrete so we can readily estimate the rate of information flow with some simple assumptions. For the next stage in the speech production part of the speech chain, the representation becomes continuous (in the form of control signals for articulatory motion). If they could be measured, we could estimate the spectral bandwidth of these

control signals and appropriately sample and quantize these signals to obtain equivalent digital signals for which the data rate could be estimated. The articulators move relatively slowly compared to the time variation of the resulting acoustic waveform. Estimates of bandwidth and required accuracy suggest that the total data rate of the sampled articulatory control signals is about 2000 bps [34]. Thus, the original text message is represented by a set of continuously varying signals whose digital representation requires a much higher data rate than the information rate that we estimated for transmission of the message as a speech signal.[2] Finally, as we will see later, the data rate of the digitized speech waveform at the end of the speech production part of the speech chain can be anywhere from 64,000 to more than 700,000 bps. We arrive at such numbers by examining the sampling rate and quantization required to represent the speech signal with a desired perceptual fidelity. For example, "telephone quality" requires that a bandwidth of 0–4 kHz be preserved, implying a sampling rate of 8000 samples/s. Each sample can be quantized with 8 bits on a log scale, resulting in a bit rate of 64,000 bps. This representation is highly intelligible (i.e., humans can readily extract the message from it) but to most listeners, it will sound different from the original speech signal uttered by the talker. On the other hand, the speech waveform can be represented with "CD quality" using a sampling rate of 44,100 samples/s with 16 bit samples, or a data rate of 705,600 bps. In this case, the reproduced acoustic signal will be virtually indistinguishable from the original speech signal.

As we move from text to speech waveform through the speech chain, the result is an encoding of the message that can be effectively transmitted by acoustic wave propagation and robustly decoded by the hearing mechanism of a listener. The above analysis of data rates shows that as we move from text to sampled speech waveform, the data rate can increase by a factor of 10,000. Part of this extra information represents characteristics of the talker such as emotional state, speech mannerisms, accent, etc., but much of it is due to the inefficiency

[2] Note that we introduce the term data rate for digital representations to distinguish from the inherent information content of the message represented by the speech signal.

of simply sampling and finely quantizing analog signals. Thus, motivated by an awareness of the low intrinsic information rate of speech, a central theme of much of digital speech processing is to obtain a digital representation with lower data rate than that of the sampled waveform.

The complete speech chain consists of a speech production/ generation model, of the type discussed above, as well as a speech perception/recognition model, as shown progressing to the left in the bottom half of Figure 1.2. The speech perception model shows the series of steps from capturing speech at the ear to understanding the message encoded in the speech signal. The first step is the effective conversion of the acoustic waveform to a spectral representation. This is done within the inner ear by the basilar membrane, which acts as a non-uniform spectrum analyzer by spatially separating the spectral components of the incoming speech signal and thereby analyzing them by what amounts to a non-uniform filter bank. The next step in the speech perception process is a neural transduction of the spectral features into a set of sound features (or distinctive features as they are referred to in the area of linguistics) that can be decoded and processed by the brain. The next step in the process is a conversion of the sound features into the set of phonemes, words, and sentences associated with the in-coming message by a language translation process in the human brain. Finally, the last step in the speech perception model is the conversion of the phonemes, words and sentences of the message into an understanding of the meaning of the basic message in order to be able to respond to or take some appropriate action. Our fundamental understanding of the processes in most of the speech perception modules in Figure 1.2 is rudimentary at best, but it is generally agreed that some physical correlate of each of the steps in the speech perception model occur within the human brain, and thus the entire model is useful for thinking about the processes that occur.

There is one additional process shown in the diagram of the complete speech chain in Figure 1.2 that we have not discussed — namely the transmission channel between the speech generation and speech perception parts of the model. In its simplest embodiment, this transmission channel consists of just the acoustic wave connection between

a speaker and a listener who are in a common space. It is essential to include this transmission channel in our model for the speech chain since it includes real world noise and channel distortions that make speech and message understanding more difficult in real communication environments. More interestingly for our purpose here — it is in this domain that we find the applications of digital speech processing.

1.2 Applications of Digital Speech Processing

The first step in most applications of digital speech processing is to convert the acoustic waveform to a sequence of numbers. Most modern A-to-D converters operate by sampling at a very high rate, applying a digital lowpass filter with cutoff set to preserve a prescribed bandwidth, and then reducing the sampling rate to the desired sampling rate, which can be as low as twice the cutoff frequency of the sharp-cutoff digital filter. This discrete-time representation is the starting point for most applications. From this point, other representations are obtained by digital processing. For the most part, these alternative representations are based on incorporating knowledge about the workings of the speech chain as depicted in Figure 1.2. As we will see, it is possible to incorporate aspects of both the speech production and speech perception process into the digital representation and processing. It is not an oversimplification to assert that digital speech processing is grounded in a set of techniques that have the goal of pushing the data rate of the speech representation to the left along either the upper or lower path in Figure 1.2.

The remainder of this chapter is devoted to a brief summary of the applications of digital speech processing, i.e., the systems that people interact with daily. Our discussion will confirm the importance of the digital representation in all application areas.

1.2.1 Speech Coding

Perhaps the most widespread applications of digital speech processing technology occur in the areas of digital transmission and storage

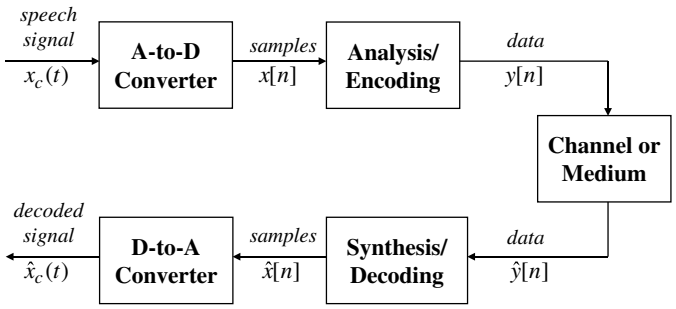

Fig. 1.3 Speech coding block diagram — encoder and decoder.

of speech signals. In these areas the centrality of the digital representation is obvious, since the goal is to compress the digital waveform representation of speech into a lower bit-rate representation. It is common to refer to this activity as "speech coding" or "speech compression."

Figure 1.3 shows a block diagram of a generic speech encoding/decoding (or compression) system. In the upper part of the figure, the A-to-D converter converts the analog speech signal $x_c(t)$ to a sampled waveform representation $x[n]$. The digital signal $x[n]$ is analyzed and coded by digital computation algorithms to produce a new digital signal $y[n]$ that can be transmitted over a digital communication channel or stored in a digital storage medium as $\hat{y}[n]$. As we will see, there are a myriad of ways to do the encoding so as to reduce the data rate over that of the sampled and quantized speech waveform $x[n]$. Because the digital representation at this point is often not directly related to the sampled speech waveform, $y[n]$ and $\hat{y}[n]$ are appropriately referred to as *data signals* that represent the speech signal. The lower path in Figure 1.3 shows the decoder associated with the speech coder. The received data signal $\hat{y}[n]$ is decoded using the inverse of the analysis processing, giving the sequence of samples $\hat{x}[n]$ which is then converted (using a D-to-A Converter) back to an analog signal $\hat{x}_c(t)$ for human listening. The decoder is often called a *synthesizer* because it must reconstitute the speech waveform from data that may bear no direct relationship to the waveform.

With carefully designed error protection coding of the digital representation, the transmitted ($y[n]$) and received ($\hat{y}[n]$) data can be essentially identical. This is the quintessential feature of digital coding. In theory, perfect transmission of the coded digital representation is possible even under very noisy channel conditions, and in the case of digital storage, it is possible to store a perfect copy of the digital representation in perpetuity if sufficient care is taken to update the storage medium as storage technology advances. This means that the speech signal can be reconstructed to within the accuracy of the original coding for as long as the digital representation is retained. In either case, the goal of the speech coder is to start with samples of the speech signal and reduce (compress) the data rate required to represent the speech signal while maintaining a desired perceptual fidelity. The compressed representation can be more efficiently transmitted or stored, or the bits saved can be devoted to error protection.

Speech coders enable a broad range of applications including narrowband and broadband wired telephony, cellular communications, voice over internet protocol (VoIP) (which utilizes the internet as a real-time communications medium), secure voice for privacy and encryption (for national security applications), extremely narrowband communications channels (such as battlefield applications using high frequency (HF) radio), and for storage of speech for telephone answering machines, interactive voice response (IVR) systems, and prerecorded messages. Speech coders often utilize many aspects of both the speech production and speech perception processes, and hence may not be useful for more general audio signals such as music. Coders that are based on incorporating only aspects of sound perception generally do not achieve as much compression as those based on speech production, but they are more general and can be used for all types of audio signals. These coders are widely deployed in MP3 and AAC players and for audio in digital television systems [120].

1.2.2 Text-to-Speech Synthesis

For many years, scientists and engineers have studied the speech production process with the goal of building a system that can start with

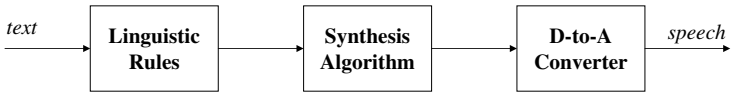

Fig. 1.4 Text-to-speech synthesis system block diagram.

text and produce speech automatically. In a sense, a text-to-speech synthesizer such as depicted in Figure 1.4 is a digital simulation of the entire upper part of the speech chain diagram. The input to the system is ordinary text such as an email message or an article from a newspaper or magazine. The first block in the text-to-speech synthesis system, labeled linguistic rules, has the job of converting the printed text input into a set of sounds that the machine must synthesize. The conversion from text to sounds involves a set of linguistic rules that must determine the appropriate set of sounds (perhaps including things like emphasis, pauses, rates of speaking, etc.) so that the resulting synthetic speech will express the words and intent of the text message in what passes for a natural voice that can be decoded accurately by human speech perception. This is more difficult than simply looking up the words in a pronouncing dictionary because the linguistic rules must determine how to pronounce acronyms, how to pronounce ambiguous words like *read*, *bass*, *object*, how to pronounce abbreviations like St. (street or Saint), Dr. (Doctor or drive), and how to properly pronounce proper names, specialized terms, etc. Once the proper pronunciation of the text has been determined, the role of the synthesis algorithm is to create the appropriate sound sequence to represent the text message in the form of speech. In essence, the synthesis algorithm must simulate the action of the vocal tract system in creating the sounds of speech. There are many procedures for assembling the speech sounds and compiling them into a proper sentence, but the most promising one today is called "unit selection and concatenation." In this method, the computer stores multiple versions of each of the basic units of speech (phones, half phones, syllables, etc.), and then decides which sequence of speech units sounds best for the particular text message that is being produced. The basic digital representation is not generally the sampled speech wave. Instead, some sort of compressed representation is normally used to

save memory and, more importantly, to allow convenient manipulation of durations and blending of adjacent sounds. Thus, the speech synthesis algorithm would include an appropriate decoder, as discussed in Section 1.2.1, whose output is converted to an analog representation via the D-to-A converter.

Text-to-speech synthesis systems are an essential component of modern human–machine communications systems and are used to do things like read email messages over a telephone, provide voice output from GPS systems in automobiles, provide the voices for talking agents for completion of transactions over the internet, handle call center help desks and customer care applications, serve as the voice for providing information from handheld devices such as foreign language phrasebooks, dictionaries, crossword puzzle helpers, and as the voice of announcement machines that provide information such as stock quotes, airline schedules, updates on arrivals and departures of flights, etc. Another important application is in reading machines for the blind, where an optical character recognition system provides the text input to a speech synthesis system.

1.2.3 Speech Recognition and Other Pattern Matching Problems

Another large class of digital speech processing applications is concerned with the automatic extraction of information from the speech signal. Most such systems involve some sort of pattern matching. Figure 1.5 shows a block diagram of a generic approach to pattern matching problems in speech processing. Such problems include the following: speech recognition, where the object is to extract the message from the speech signal; speaker recognition, where the goal is to identify who is speaking; speaker verification, where the goal is to verify a speaker's claimed identity from analysis of their speech

Fig. 1.5 Block diagram of general pattern matching system for speech signals.

signal; word spotting, which involves monitoring a speech signal for the occurrence of specified words or phrases; and automatic indexing of speech recordings based on recognition (or spotting) of spoken keywords.

The first block in the pattern matching system converts the analog speech waveform to digital form using an A-to-D converter. The feature analysis module converts the sampled speech signal to a set of feature vectors. Often, the same analysis techniques that are used in speech coding are also used to derive the feature vectors. The final block in the system, namely the pattern matching block, dynamically time aligns the set of feature vectors representing the speech signal with a concatenated set of stored patterns, and chooses the identity associated with the pattern which is the closest match to the time-aligned set of feature vectors of the speech signal. The symbolic output consists of a set of recognized words, in the case of speech recognition, or the identity of the best matching talker, in the case of speaker recognition, or a decision as to whether to accept or reject the identity claim of a speaker in the case of speaker verification.

Although the block diagram of Figure 1.5 represents a wide range of speech pattern matching problems, the biggest use has been in the area of recognition and understanding of speech in support of human–machine communication by voice. The major areas where such a system finds applications include command and control of computer software, voice dictation to create letters, memos, and other documents, natural language voice dialogues with machines to enable help desks and call centers, and for agent services such as calendar entry and update, address list modification and entry, etc.

Pattern recognition applications often occur in conjunction with other digital speech processing applications. For example, one of the preeminent uses of speech technology is in portable communication devices. Speech coding at bit rates on the order of 8 Kbps enables normal voice conversations in cell phones. Spoken name speech recognition in cellphones enables voice dialing capability that can automatically dial the number associated with the recognized name. Names from directories with upwards of several hundred names can readily be recognized and dialed using simple speech recognition technology.

Another major speech application that has long been a dream of speech researchers is *automatic language translation.* The goal of language translation systems is to convert spoken words in one language to spoken words in another language so as to facilitate natural language voice dialogues between people speaking different languages. Language translation technology requires speech synthesis systems that work in both languages, along with speech recognition (and generally natural language understanding) that also works for both languages; hence it is a very difficult task and one for which only limited progress has been made. When such systems exist, it will be possible for people speaking different languages to communicate at data rates on the order of that of printed text reading!

1.2.4 Other Speech Applications

The range of speech communication applications is illustrated in Figure 1.6. As seen in this figure, the techniques of digital speech processing are a key ingredient of a wide range of applications that include the three areas of transmission/storage, speech synthesis, and speech recognition as well as many others such as speaker identification, speech signal quality enhancement, and aids for the hearing- or visually-impaired.

The block diagram in Figure 1.7 represents any system where time signals such as speech are processed by the techniques of DSP. This figure simply depicts the notion that once the speech signal is sampled, it can be manipulated in virtually limitless ways by DSP techniques. Here again, manipulations and modifications of the speech signal are

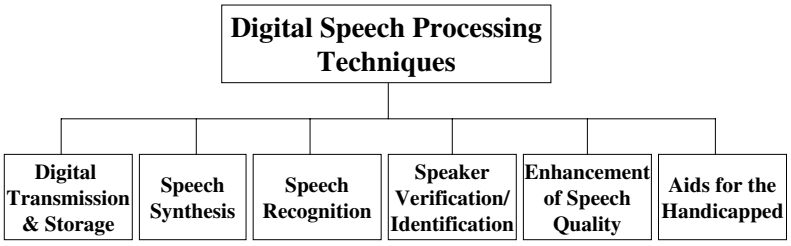

Fig. 1.6 Range of speech communication applications.

Fig. 1.7 General block diagram for application of digital signal processing to speech signals.

usually achieved by transforming the speech signal into an alternative representation (that is motivated by our understanding of speech production and speech perception), operating on that representation by further digital computation, and then transforming back to the waveform domain, using a D-to-A converter.

One important application area is *speech enhancement*, where the goal is to remove or suppress noise or echo or reverberation picked up by a microphone along with the desired speech signal. In human-to-human communication, the goal of speech enhancement systems is to make the speech more intelligible and more natural; however, in reality the best that has been achieved so far is less perceptually annoying speech that essentially maintains, but does not improve, the intelligibility of the noisy speech. Success *has* been achieved, however, in making distorted speech signals more useful for further processing as part of a speech coder, synthesizer, or recognizer. An excellent reference in this area is the recent textbook by Loizou [72].

Other examples of manipulation of the speech signal include timescale modification to align voices with video segments, to modify voice qualities, and to speed-up or slow-down prerecorded speech (e.g., for talking books, rapid review of voice mail messages, or careful scrutinizing of spoken material).

1.3 Our Goal for this Text

We have discussed the speech signal and how it encodes information for human communication. We have given a brief overview of the way in which digital speech processing is being applied today, and we have hinted at some of the possibilities that exist for the future. These and many more examples all rely on the basic principles of digital speech processing, which we will discuss in the remainder of this text. We make no pretense of exhaustive coverage. The subject is too broad and

too deep. Our goal is only to provide an up-to-date introduction to this fascinating field. We will not be able to go into great depth, and we will not be able to cover all the possible applications of digital speech processing techniques. Instead our focus is on the fundamentals of digital speech processing and their application to coding, synthesis, and recognition. This means that some of the latest algorithmic innovations and applications will not be discussed — not because they are not interesting, but simply because there are so many fundamental tried-and-true techniques that remain at the core of digital speech processing. We hope that this text will stimulate readers to investigate the subject in greater depth using the extensive set of references provided.

2

The Speech Signal

As the discussion in Chapter 1 shows, the goal in many applications of digital speech processing techniques is to move the digital representation of the speech signal from the waveform samples back up the speech chain toward the message. To gain a better idea of what this means, this chapter provides a brief overview of the phonetic representation of speech and an introduction to models for the production of the speech signal.

2.1 Phonetic Representation of Speech

Speech can be represented phonetically by a finite set of symbols called the *phonemes* of the language, the number of which depends upon the language and the refinement of the analysis. For most languages the number of phonemes is between 32 and 64. A condensed inventory of the sounds of speech in the English language is given in Table 2.1, where the phonemes are denoted by a set of ASCII symbols called the ARPAbet. Table 2.1 also includes some simple examples of ARPAbet transcriptions of words containing each of the phonemes of English. Additional phonemes can be added to Table 2.1 to account for allophonic variations and events such as glottal stops and pauses.

Table 2.1 Condensed list of ARPAbet phonetic symbols for North American English.

Class	ARPAbet	Example	Transcription
Vowels and	IY	b*ee*t	[B IY T]
diphthongs	IH	b*i*t	[B IH T]
	EY	b*ai*t	[B EY T]
	EH	b*e*t	[B EH T]
	AE	b*a*t	[B AE T]
	AA	b*o*b	[B AA B]
	AO	b*or*n	[B AO R N]
	UH	b*oo*k	[B UH K]
	OW	b*oa*t	[B OW T]
	UW	b*oo*t	[B UW T]
	AH	b*u*t	[B AH T]
	ER	b*ir*d	[B ER D]
	AY	b*uy*	[B AY]
	AW	d*ow*n	[D AW N]
	OY	b*oy*	[B OY]
Glides	Y	*y*ou	[Y UH]
	R	*r*ent	[R EH N T]
Liquids	W	*w*it	[W IH T]
	L	*l*et	[L EH T]
Nasals	M	*m*et	[M EH T]
	N	*n*et	[N EH T]
	NG	si*ng*	[S IH NG]
Stops	P	*p*at	[P AE T]
	B	*b*et	[B EH T]
	T	*t*en	[T EH N]
	D	*d*ebt	[D EH T]
	K	*k*it	[K IH T]
	G	*g*et	[G EH T]
Fricatives	HH	*h*at	[HH AE T]
	F	*f*at	[F AE T]
	V	*v*at	[V AE T]
	TH	*th*ing	[TH IH NG]
	DH	*th*at	[DH AE T]
	S	*s*at	[S AE T]
	Z	*z*oo	[Z UW]
	SH	*sh*ut	[SH AH T]
	ZH	a*z*ure	[AE ZH ER]
Affricates	CH	*ch*ase	[CH EY S]
	JH	*j*udge	[JH AH JH]

[a]This set of 39 phonemes is used in the CMU Pronouncing Dictionary available on-line at http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

Figure 1.1 on p. 4 shows how the sounds corresponding to the text "should we chase" are encoded into a speech waveform. We see that, for the most part, phonemes have a distinctive appearance in the speech waveform. Thus sounds like /SH/ and /S/ look like (spectrally shaped)

random noise, while the vowel sounds /UH/, /IY/, and /EY/ are highly structured and quasi-periodic. These differences result from the distinctively different ways that these sounds are produced.

2.2 Models for Speech Production

A schematic longitudinal cross-sectional drawing of the human vocal tract mechanism is given in Figure 2.1 [35]. This diagram highlights the essential physical features of human anatomy that enter into the final stages of the speech production process. It shows the vocal tract as a tube of nonuniform cross-sectional area that is bounded at one end by the vocal cords and at the other by the mouth opening. This tube serves as an acoustic transmission system for sounds generated inside the vocal tract. For creating nasal sounds like /M/, /N/, or /NG/, a side-branch tube, called the nasal tract, is connected to the main acoustic branch by the trapdoor action of the velum. This branch path radiates sound at the nostrils. The shape (variation of cross-section along the axis) of the vocal tract varies with time due to motions of the lips, jaw, tongue, and velum. Although the actual human vocal tract is not laid out along a straight line as in Figure 2.1, this type of model is a reasonable approximation for wavelengths of the sounds in speech.

The sounds of speech are generated in the system of Figure 2.1 in several ways. *Voiced sounds* (vowels, liquids, glides, nasals in Table 2.1)

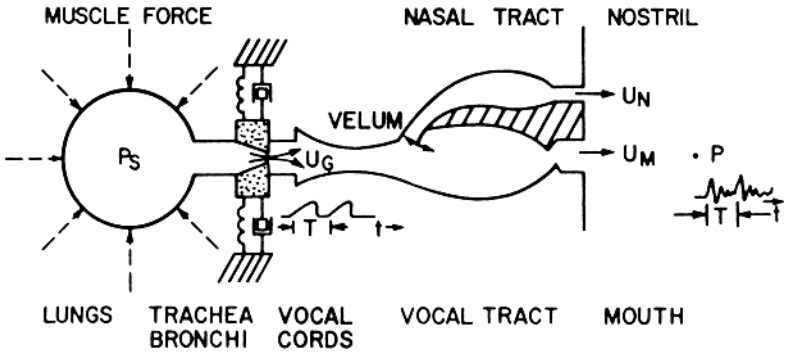

Fig. 2.1 Schematic model of the vocal tract system. (After Flanagan et al. [35].)

are produced when the vocal tract tube is excited by pulses of air pressure resulting from quasi-periodic opening and closing of the glottal orifice (opening between the vocal cords). Examples in Figure 1.1 are the vowels /UH/, /IY/, and /EY/, and the liquid consonant /W/. *Unvoiced sounds* are produced by creating a constriction somewhere in the vocal tract tube and forcing air through that constriction, thereby creating turbulent air flow, which acts as a random noise excitation of the vocal tract tube. Examples are the unvoiced fricative sounds such as /SH/ and /S/. A third sound production mechanism is when the vocal tract is partially closed off causing turbulent flow due to the constriction, at the same time allowing quasi-periodic flow due to vocal cord vibrations. Sounds produced in this manner include the *voiced fricatives* /V/, /DH/, /Z/, and /ZH/. Finally, *plosive sounds* such as /P/, /T/, and /K/ and affricates such as /CH/ are formed by momentarily closing off air flow, allowing pressure to build up behind the closure, and then abruptly releasing the pressure. All these excitation sources create a wide-band excitation signal to the vocal tract tube, which acts as an acoustic transmission line with certain vocal tract shape-dependent resonances that tend to emphasize some frequencies of the excitation relative to others.

As discussed in Chapter 1 and illustrated by the waveform in Figure 1.1, the general character of the speech signal varies at the phoneme rate, which is on the order of 10 phonemes per second, while the detailed time variations of the speech waveform are at a much higher rate. That is, the changes in vocal tract configuration occur relatively slowly compared to the detailed time variation of the speech signal. The sounds created in the vocal tract are shaped in the frequency domain by the frequency response of the vocal tract. The resonance frequencies resulting from a particular configuration of the articulators are instrumental in forming the sound corresponding to a given phoneme. These resonance frequencies are called the *formant frequencies* of the sound [32, 34]. In summary, the fine structure of the time waveform is created by the sound sources in the vocal tract, and the resonances of the vocal tract tube shape these sound sources into the phonemes.

The system of Figure 2.1 can be described by acoustic theory, and numerical techniques can be used to create a complete physical

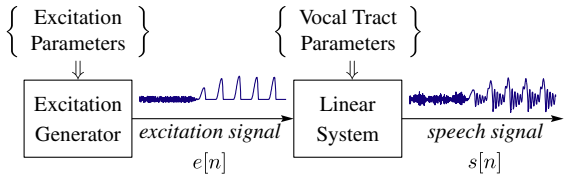

Fig. 2.2 Source/system model for a speech signal.

simulation of sound generation and transmission in the vocal tract [36, 93], but, for the most part, it is sufficient to model the production of a sampled speech signal by a discrete-time system model such as the one depicted in Figure 2.2. The discrete-time time-varying linear system on the right in Figure 2.2 simulates the frequency shaping of the vocal tract tube. The excitation generator on the left simulates the different modes of sound generation in the vocal tract. Samples of a speech signal are assumed to be the output of the time-varying linear system.

In general such a model is called a *source/system* model of speech production. The short-time frequency response of the linear system simulates the frequency shaping of the vocal tract system, and since the vocal tract changes shape relatively slowly, it is reasonable to assume that the linear system response does not vary over time intervals on the order of 10 ms or so. Thus, it is common to characterize the discrete-time linear system by a system function of the form:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} = \frac{b_0 \prod_{k=1}^{M} (1 - d_k z^{-1})}{\prod_{k=1}^{N} (1 - c_k z^{-1})}, \tag{2.1}$$

where the filter coefficients a_k and b_k (labeled as vocal tract parameters in Figure 2.2) change at a rate on the order of 50–100 times/s. Some of the poles (c_k) of the system function lie close to the unit circle and create resonances to model the formant frequencies. In detailed modeling of speech production [32, 34, 64], it is sometimes useful to employ zeros (d_k) of the system function to model nasal and fricative

sounds. However, as we discuss further in Chapter 4, many applications of the source/system model only include poles in the model because this simplifies the analysis required to estimate the parameters of the model from the speech signal.

The box labeled excitation generator in Figure 2.2 creates an appropriate excitation for the type of sound being produced. For voiced speech the excitation to the linear system is a quasi-periodic sequence of discrete (glottal) pulses that look very much like those shown in the righthand half of the excitation signal waveform in Figure 2.2. The fundamental frequency of the glottal excitation determines the perceived pitch of the voice. The individual finite-duration glottal pulses have a lowpass spectrum that depends on a number of factors [105]. Therefore, the periodic sequence of smooth glottal pulses has a harmonic line spectrum with components that decrease in amplitude with increasing frequency. Often it is convenient to absorb the glottal pulse spectrum contribution into the vocal tract system model of (2.1). This can be achieved by a small increase in the order of the denominator over what would be needed to represent the formant resonances. For unvoiced speech, the linear system is excited by a random number generator that produces a discrete-time noise signal with flat spectrum as shown in the left-hand half of the excitation signal. The excitation in Figure 2.2 switches from unvoiced to voiced leading to the speech signal output as shown in the figure. In either case, the linear system imposes its frequency response on the spectrum to create the speech sounds.

This model of speech as the output of a slowly time-varying digital filter with an excitation that captures the nature of the voiced/unvoiced distinction in speech production is the basis for thinking about the speech signal, and a wide variety of digital representations of the speech signal are based on it. That is, the speech signal is represented by the parameters of the model instead of the sampled waveform. By assuming that the properties of the speech signal (and the model) are constant over short time intervals, it is possible to compute/measure/estimate the parameters of the model by analyzing short blocks of samples of the speech signal. It is through such models and analysis techniques that we are able to build properties of the speech production process into digital representations of the speech signal.

2.3 More Refined Models

Source/system models as shown in Figure 2.2 with the system characterized by a time-sequence of time-invariant systems are quite sufficient for most applications in speech processing, and we shall rely on such models throughout this text. However, such models are based on many approximations including the assumption that the source and the system do not interact, the assumption of linearity, and the assumption that the distributed continuous-time vocal tract transmission system can be modeled by a discrete linear time-invariant system. Fluid mechanics and acoustic wave propagation theory are fundamental physical principles that must be applied for detailed modeling of speech production. Since the early work of Flanagan and Ishizaka [34, 36, 51] much work has been devoted to creating detailed simulations of glottal flow, the interaction of the glottal source and the vocal tract in speech production, and the nonlinearities that enter into sound generation and transmission in the vocal tract. Stevens [121] and Quatieri [94] provide useful discussions of these effects. For many years, researchers have sought to measure the physical dimensions of the human vocal tract during speech production. This information is essential for detailed simulations based on acoustic theory. Early efforts to measure vocal tract area functions involved hand tracing on X-ray pictures [32]. Recent advances in MRI imaging and computer image analysis have provided significant advances in this area of speech science [17].

3

Hearing and Auditory Perception

In Chapter 2, we introduced the speech production process and showed how we could model speech production using discrete-time systems. In this chapter we turn to the perception side of the speech chain to discuss properties of human sound perception that can be employed to create digital representations of the speech signal that are perceptually robust.

3.1 The Human Ear

Figure 3.1 shows a schematic view of the human ear showing the three distinct sound processing sections, namely: the *outer ear* consisting of the pinna, which gathers sound and conducts it through the external canal to the middle ear; the *middle ear* beginning at the tympanic membrane, or eardrum, and including three small bones, the malleus (also called the hammer), the incus (also called the anvil) and the stapes (also called the stirrup), which perform a transduction from acoustic waves to mechanical pressure waves; and finally, the *inner ear,* which consists of the cochlea and the set of neural connections to the auditory nerve, which conducts the neural signals to the brain.

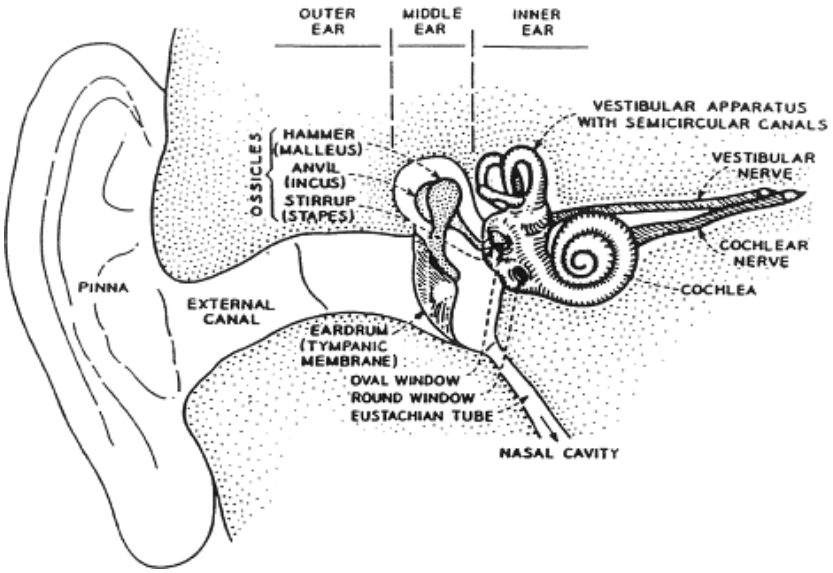

Fig. 3.1 Schematic view of the human ear (inner and middle structures enlarged). (After Flanagan [34].)

Figure 3.2 [107] depicts a block diagram abstraction of the auditory system. The acoustic wave is transmitted from the outer ear to the inner ear where the ear drum and bone structures convert the sound wave to mechanical vibrations which ultimately are transferred to the basilar membrane inside the cochlea. The basilar membrane vibrates in a frequency-selective manner along its extent and thereby performs a rough (non-uniform) spectral analysis of the sound. Distributed along

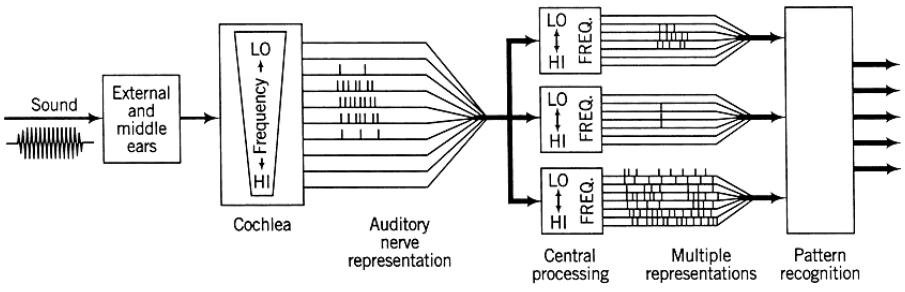

Fig. 3.2 Schematic model of the auditory mechanism. (After Sachs et al. [107].)

the basilar membrane are a set of inner hair cells that serve to convert motion along the basilar membrane to neural activity. This produces an auditory nerve representation in both time and frequency. The processing at higher levels in the brain, shown in Figure 3.2 as a sequence of central processing with multiple representations followed by some type of pattern recognition, is not well understood and we can only postulate the mechanisms used by the human brain to perceive sound or speech. Even so, a wealth of knowledge about how sounds are perceived has been discovered by careful experiments that use tones and noise signals to stimulate the auditory system of human observers in very specific and controlled ways. These experiments have yielded much valuable knowledge about the sensitivity of the human auditory system to acoustic properties such as intensity and frequency.

3.2 Perception of Loudness

A key factor in the perception of speech and other sounds is *loudness*. Loudness is a perceptual quality that is related to the physical property of sound pressure level. Loudness is quantified by relating the actual sound pressure level of a pure tone (in dB relative to a standard reference level) to the perceived loudness of the same tone (in a unit called phons) over the range of human hearing (20 Hz–20 kHz). This relationship is shown in Figure 3.3 [37, 103]. These loudness curves show that the perception of loudness is frequency-dependent. Specifically, the dotted curve at the bottom of the figure labeled "threshold of audibility" shows the sound pressure level that is required for a sound of a given frequency to be just audible (by a person with normal hearing). It can be seen that low frequencies must be significantly more intense than frequencies in the mid-range in order that they be perceived at all. The solid curves are equal-loudness-level contours measured by comparing sounds at various frequencies with a pure tone of frequency 1000 Hz and known sound pressure level. For example, the point at frequency 100 Hz on the curve labeled 50 (phons) is obtained by adjusting the power of the 100 Hz tone until it sounds as loud as a 1000 Hz tone having a sound pressure level of 50 dB. Careful measurements of this kind show that a 100 Hz tone must have a sound pressure level of about 60 dB in order

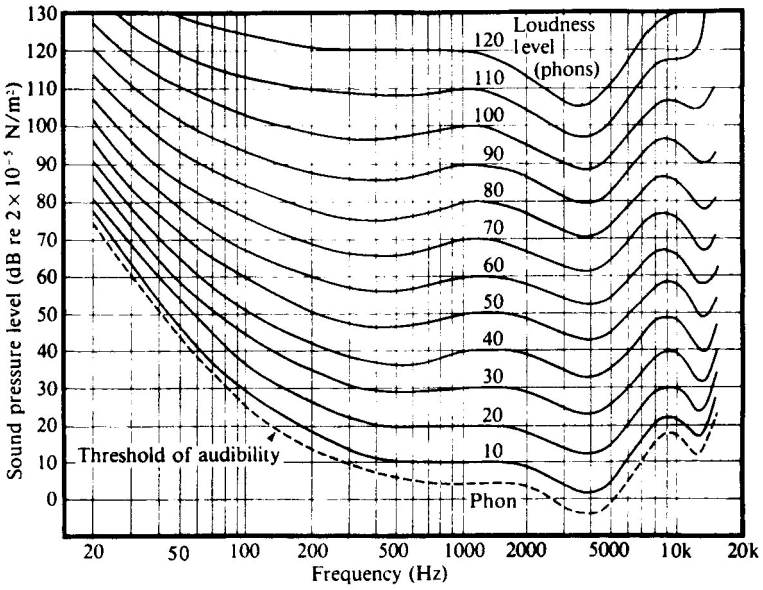

Fig. 3.3 Loudness level for human hearing. (After Fletcher and Munson [37].)

to be perceived to be equal in loudness to the 1000 Hz tone of sound pressure level 50 dB. By convention, both the 50 dB 1000 Hz tone and the 60 dB 100 Hz tone are said to have a *loudness level* of 50 phons (pronounced as /F OW N Z/).

The equal-loudness-level curves show that the auditory system is most sensitive for frequencies ranging from about 100 Hz up to about 6 kHz with the greatest sensitivity at around 3 to 4 kHz. This is almost precisely the range of frequencies occupied by most of the sounds of speech.

3.3 Critical Bands

The non-uniform frequency analysis performed by the basilar membrane can be thought of as equivalent to that of a set of bandpass filters whose frequency responses become increasingly broad with increasing frequency. An idealized version of such a filter bank is depicted schematically in Figure 3.4. In reality, the bandpass filters are not ideal

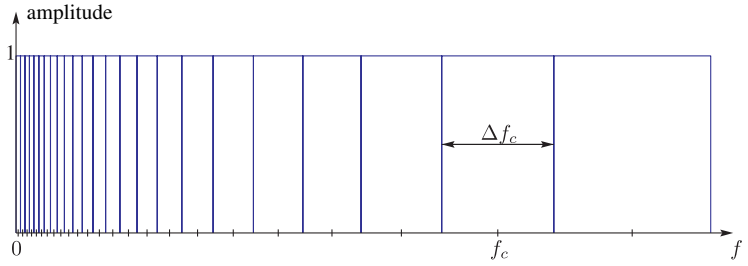

Fig. 3.4 Schematic representation of bandpass filters according to the critical band theory of hearing.

as shown in Figure 3.4, but their frequency responses overlap significantly since points on the basilar membrane cannot vibrate independently of each other. Even so, the concept of bandpass filter analysis in the cochlea is well established, and the critical bandwidths have been defined and measured using a variety of methods, showing that the effective bandwidths are constant at about 100 Hz for center frequencies below 500 Hz, and with a relative bandwidth of about 20% of the center frequency above 500 Hz. An equation that fits empirical measurements over the auditory range is

$$\Delta f_c = 25 + 75[1 + 1.4(f_c/1000)^2]^{0.69}, \tag{3.1}$$

where Δf_c is the critical bandwidth associated with center frequency f_c [134]. Approximately 25 critical band filters span the range from 0 to 20 kHz. The concept of critical bands is very important in understanding such phenomena as loudness perception, pitch perception, and masking, and it therefore provides motivation for digital representations of the speech signal that are based on a frequency decomposition.

3.4 Pitch Perception

Most musical sounds as well as voiced speech sounds have a periodic structure when viewed over short time intervals, and such sounds are perceived by the auditory system as having a quality known as *pitch*. Like loudness, pitch is a subjective attribute of sound that is related to the fundamental frequency of the sound, which is a physical attribute of the acoustic waveform [122]. The relationship between pitch (measured

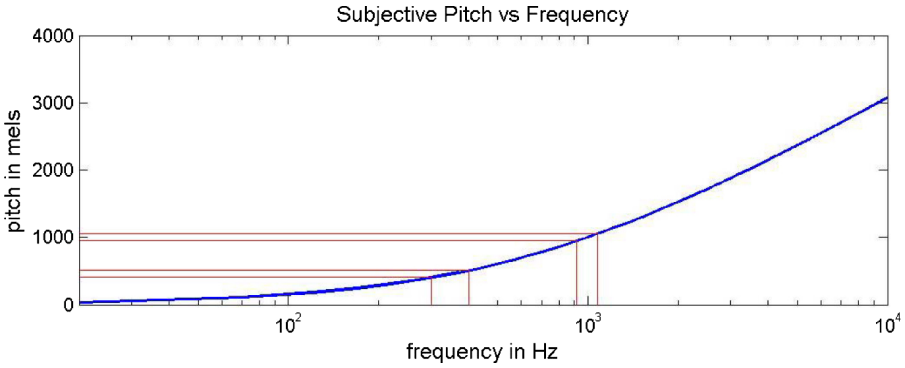

Fig. 3.5 Relation between subjective pitch and frequency of a pure tone.

on a nonlinear frequency scale called the mel-scale) and frequency of a pure tone is approximated by the equation [122]:

$$\text{Pitch in mels} = 1127 \log_e(1 + f/700), \tag{3.2}$$

which is plotted in Figure 3.5. This expression is calibrated so that a frequency of 1000 Hz corresponds to a pitch of 1000 mels. This empirical scale describes the results of experiments where subjects were asked to adjust the pitch of a measurement tone to half the pitch of a reference tone. To calibrate the scale, a tone of frequency 1000 Hz is given a pitch of 1000 mels. Below 1000 Hz, the relationship between pitch and frequency is nearly proportional. For higher frequencies, however, the relationship is nonlinear. For example, (3.2) shows that a frequency of $f = 5000$ Hz corresponds to a pitch of 2364 mels.

The psychophysical phenomenon of pitch, as quantified by the mel-scale, can be related to the concept of critical bands [134]. It turns out that more or less independently of the center frequency of the band, one critical bandwidth corresponds to about 100 mels on the pitch scale. This is shown in Figure 3.5, where a critical band of width $\Delta f_c = 160$ Hz centered on $f_c = 1000$ Hz maps into a band of width 106 mels and a critical band of width 100 Hz centered on 350 Hz maps into a band of width 107 mels. Thus, what we know about pitch perception reinforces the notion that the auditory system performs a frequency analysis that can be simulated with a bank of bandpass filters whose bandwidths increase as center frequency increases.

Voiced speech is quasi-periodic, but contains many frequencies. Nevertheless, many of the results obtained with pure tones are relevant to the perception of voice pitch as well. Often the term *pitch period* is used for the fundamental period of the voiced speech signal even though its usage in this way is somewhat imprecise.

3.5 Auditory Masking

The phenomenon of critical band auditory analysis can be explained intuitively in terms of vibrations of the basilar membrane. A related phenomenon, called masking, is also attributable to the mechanical vibrations of the basilar membrane. Masking occurs when one sound makes a second superimposed sound inaudible. Loud tones causing strong vibrations at a point on the basilar membrane can swamp out vibrations that occur nearby. Pure tones can mask other pure tones, and noise can mask pure tones as well. A detailed discussion of masking can be found in [134].

Figure 3.6 illustrates masking of tones by tones. The notion that a sound becomes inaudible can be quantified with respect to the threshold of audibility. As shown in Figure 3.6, an intense tone (called the masker) tends to raise the threshold of audibility around its location on the frequency axis as shown by the solid line. All spectral components whose level is below this raised threshold are masked and therefore do

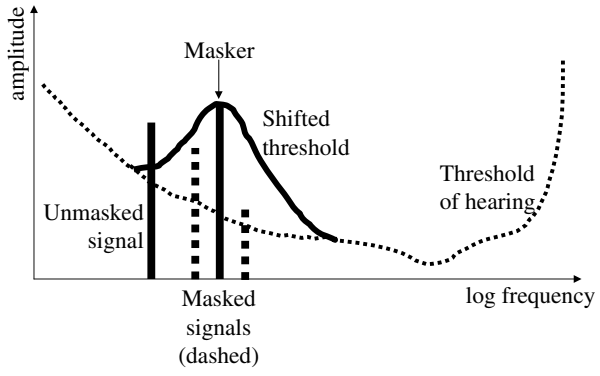

Fig. 3.6 Illustration of effects of masking.

not need to be reproduced in a speech (or audio) processing system because they would not be heard. Similarly, any spectral component whose level is above the raised threshold is not masked, and therefore will be heard. It has been shown that the masking effect is greater for frequencies above the masking frequency than below. This is shown in Figure 3.6 where the falloff of the shifted threshold is less abrupt above than below the masker.

Masking is widely employed in digital representations of speech (and audio) signals by "hiding" errors in the representation in areas where the threshold of hearing is elevated by strong frequency components in the signal [120]. In this way, it is possible to achieve lower data rate representations while maintaining a high degree of perceptual fidelity.

3.6 Complete Model of Auditory Processing

In Chapter 2, we described elements of a generative model of speech production which could, in theory, completely describe and model the ways in which speech is produced by humans. In this chapter, we described elements of a model of speech perception. However, the problem that arises is that our detailed knowledge of how speech is perceived and understood, beyond the basilar membrane processing of the inner ear, is rudimentary at best, and thus we rely on psychophysical experimentation to understand the role of loudness, critical bands, pitch perception, and auditory masking in speech perception in humans. Although some excellent auditory models have been proposed [41, 48, 73, 115] and used in a range of speech processing systems, all such models are incomplete representations of our knowledge about how speech is understood.

4

Short-Time Analysis of Speech

In Figure 2.2 of Chapter 2, we presented a model for speech production in which an excitation source provides the basic temporal fine structure while a slowly varying filter provides spectral shaping (often referred to as the spectrum envelope) to create the various sounds of speech. In Figure 2.2, the source/system separation was presented at an abstract level, and few details of the excitation or the linear system were given. Both the excitation and the linear system were defined *implicitly* by the assertion that the sampled speech signal was the output of the overall system. Clearly, this is not sufficient to uniquely specify either the excitation or the system. Since our goal is to extract parameters of the model by analysis of the speech signal, it is common to assume structures (or representations) for both the excitation generator and the linear system. One such model uses a more detailed representation of the excitation in terms of separate source generators for voiced and unvoiced speech as shown in Figure 4.1. In this model the unvoiced excitation is assumed to be a random noise sequence, and the voiced excitation is assumed to be a periodic impulse train with impulses spaced by the pitch period (P_0) rounded to the nearest

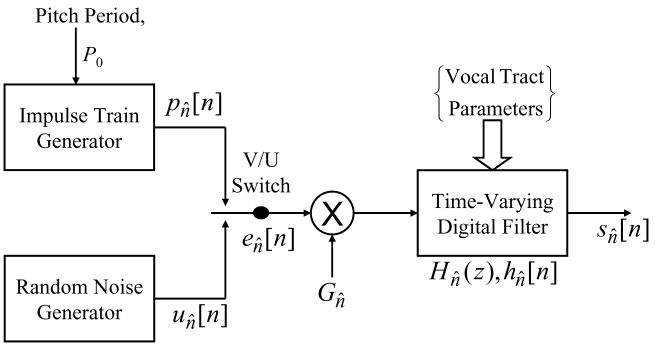

Fig. 4.1 Voiced/unvoiced/system model for a speech signal.

sample.[1] The pulses needed to model the glottal flow waveform during voiced speech are assumed to be combined (by convolution) with the impulse response of the linear system, which is assumed to be slowly-time-varying (changing every 50–100 ms or so). By this we mean that over the timescale of phonemes, the impulse response, frequency response, and system function of the system remains relatively constant. For example over time intervals of tens of milliseconds, the system can be described by the convolution expression

$$s_{\hat{n}}[n] = \sum_{m=0}^{\infty} h_{\hat{n}}[m] e_{\hat{n}}[n-m], \tag{4.1}$$

where the subscript $\hat{n}$ denotes the time index pointing to the block of samples of the entire speech signal $s[n]$ wherein the impulse response $h_{\hat{n}}[m]$ applies. We use n for the time index within that interval, and m is the index of summation in the convolution sum. In this model, the gain $G_{\hat{n}}$ is absorbed into $h_{\hat{n}}[m]$ for convenience. As discussed in Chapter 2, the most general linear time-invariant system would be characterized by a rational system function as given in (2.1). However, to simplify analysis, it is often assumed that the system is an all-pole system with

[1] As mentioned in Chapter 3, the period of voiced speech is related to the fundamental frequency (perceived as pitch) of the voice.

system function of the form:

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}. \tag{4.2}$$

The coefficients G and a_k in (4.2) change with time, and should therefore be indexed with the subscript $\hat{n}$ as in (4.1) and Figure 4.1, but this complication of notation is not generally necessary since it is usually clear that the system function or difference equation only applies over a short time interval.[2] Although the linear system is assumed to model the composite spectrum effects of radiation, vocal tract tube, and glottal excitation pulse shape (for voiced speech only) over a short time interval, the linear system in the model is commonly referred to as simply the "vocal tract" system and the corresponding impulse response is called the "vocal tract impulse response." For all-pole linear systems, as represented by (4.2), the input and output are related by a difference equation of the form:

$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + Ge[n], \tag{4.3}$$

where as discussed above, we have suppressed the indication of the time at which the difference equation applies.

With such a model as the basis, common practice is to partition the analysis of the speech signal into techniques for extracting the parameters of the excitation model, such as the pitch period and voiced/unvoiced classification, and techniques for extracting the linear system model (which imparts the spectrum envelope or spectrum shaping).

Because of the slowly varying nature of the speech signal, it is common to process speech in blocks (also called "frames") over which the properties of the speech waveform can be assumed to remain relatively constant. This leads to the basic principle of short-time analysis, which

[2] In general, we use n and m for discrete indices for sequences, but whenever we want to indicate a specific analysis time, we use $\hat{n}$.

is represented in a general form by the equation:

$$X_{\hat{n}} = \sum_{m=-\infty}^{\infty} T\{x[m]w[\hat{n}-m]\}, \tag{4.4}$$

where $X_{\hat{n}}$ represents the short-time analysis parameter (or vector of parameters) at analysis time $\hat{n}$.[3] The operator $T\{\ \}$ defines the nature of the short-time analysis function, and $w[\hat{n}-m]$ represents a time-shifted window sequence, whose purpose is to select a segment of the sequence $x[m]$ in the neighborhood of sample $m = \hat{n}$. We will see several examples of such operators that are designed to extract or highlight certain features of the speech signal. The infinite limits in (4.4) imply summation over all nonzero values of the windowed segment $x_{\hat{n}}[m] = x[m]w[\hat{n}-m]$; i.e., for all m in the region of support of the window. For example, a finite-duration[4] window might be a Hamming window defined by

$$w_H[m] = \begin{cases} 0.54 + 0.46\cos(\pi m/M) & -M \le m \le M \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Figure 4.2 shows a discrete-time Hamming window and its discrete-time Fourier transform, as used throughout this chapter.[5] It can be shown that a $(2M+1)$ sample Hamming window has a frequency main lobe (full) bandwidth of $4\pi/M$. Other windows will have similar properties, i.e., they will be concentrated in time and frequency, and the frequency width will be inversely proportional to the time width [89].

Figure 4.3 shows a 125 ms segment of a speech waveform that includes both unvoiced (0–50 ms) and voiced speech (50–125 ms). Also shown is a sequence of data windows of duration 40 ms and shifted by 15 ms (320 samples at 16 kHz sampling rate) between windows. This illustrates how short-time analysis is implemented.

[3] Sometimes (4.4) is normalized by dividing by the effective window length, i.e., $\sum_{m=-\infty}^{\infty} w[m]$, so that $X_{\hat{n}}$ is a weighted average.

[4] An example of an infinite-duration window is $w[n] = na^n$ for $n \ge 0$. Such a window can lead to recursive implementations of short-time analysis functions [9].

[5] We have assumed the time origin at the center of a symmetric interval of $2M+1$ samples. A causal window would be shifted to the right by M samples.

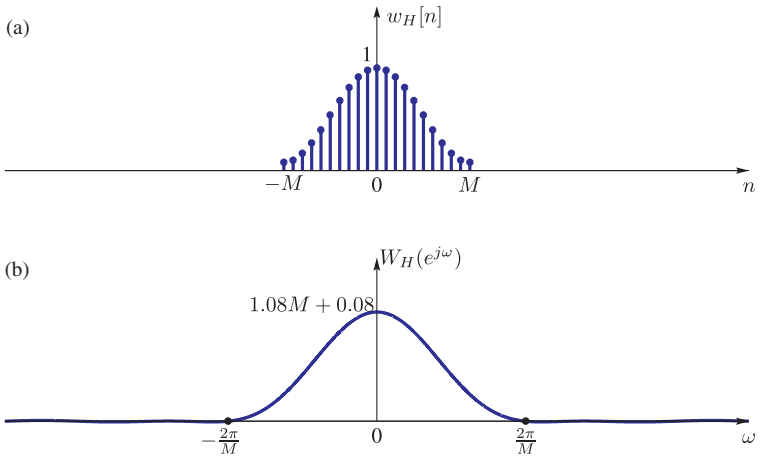

Fig. 4.2 Hamming window (a) and its discrete-time Fourier transform (b).

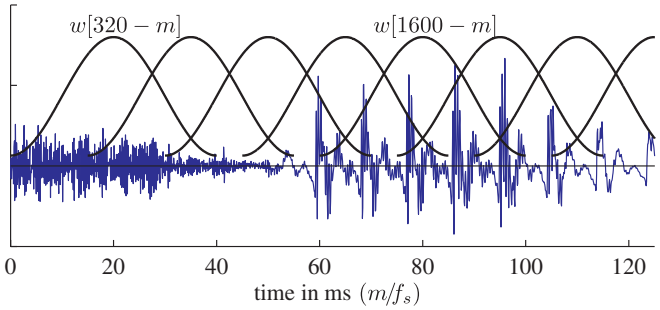

Fig. 4.3 Section of speech waveform with short-time analysis windows.

4.1 Short-Time Energy and Zero-Crossing Rate

Two basic short-time analysis functions useful for speech signals are the short-time energy and the short-time zero-crossing rate. These functions are simple to compute, and they are useful for estimating properties of the excitation function in the model.

The short-time energy is defined as

$$E_{\hat{n}} = \sum_{m=-\infty}^{\infty} (x[m]w[\hat{n} - m])^2 = \sum_{m=-\infty}^{\infty} x^2[m]w^2[\hat{n} - m]. \qquad (4.6)$$

In this case the operator $T\{\ \}$ is simply squaring the windowed samples. As shown in (4.6), it is often possible to express short-time analysis operators as a convolution or linear filtering operation. In this case, $E_{\hat{n}} = x^2[n] * h_e[n]\big|_{n=\hat{n}}$, where the impulse response of the linear filter is $h_e[n] = w^2[n]$.

Similarly, the short-time zero crossing rate is defined as the weighted average of the number of times the speech signal changes sign within the time window. Representing this operator in terms of linear filtering leads to

$$Z_{\hat{n}} = \sum_{m=-\infty}^{\infty} 0.5\,|\mathrm{sgn}\{x[m]\} - \mathrm{sgn}\{x[m-1]\}|\,w[\hat{n}-m], \tag{4.7}$$

where

$$\mathrm{sgn}\{x\} = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0. \end{cases} \tag{4.8}$$

Since $0.5|\mathrm{sgn}\{x[m]\} - \mathrm{sgn}\{x[m-1]\}|$ is equal to 1 if $x[m]$ and $x[m-1]$ have different algebraic signs and 0 if they have the same sign, it follows that $Z_{\hat{n}}$ in (4.7) is a weighted sum of all the instances of alternating sign (zero-crossing) that fall within the support region of the shifted window $w[\hat{n}-m]$. While this is a convenient representation that fits the general framework of (4.4), the computation of $Z_{\hat{n}}$ could be implemented in other ways.

Figure 4.4 shows an example of the short-time energy and zero-crossing rate for a segment of speech with a transition from unvoiced to voiced speech. In both cases, the window is a Hamming window (two examples shown) of duration 25 ms (equivalent to 401 samples at a 16 kHz sampling rate).[6] Thus, both the short-time energy and the short-time zero-crossing rate are output of a lowpass filter whose frequency response is as shown in Figure 4.2(b). For the 401-point Hamming window used in Figure 4.4, the frequency response is very small for discrete-time frequencies above $2\pi/200$ rad/s (equivalent to $16000/200 = 80$ Hz analog frequency). This means that the short-time

[6] In the case of the short-time energy, the window applied to the signal samples was the square-root of the Hamming window, so that $h_e[n] = w^2[n]$ is the Hamming window defined by (4.5).

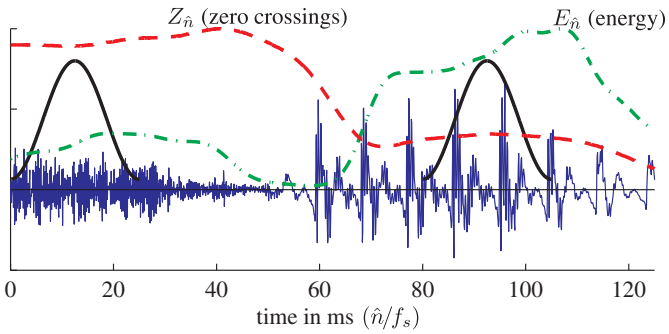

Fig. 4.4 Section of speech waveform with short-time energy and zero-crossing rate superimposed.

energy and zero-crossing rate functions are slowly varying compared to the time variations of the speech signal, and therefore, they can be sampled at a much lower rate than that of the original speech signal. For finite-length windows like the Hamming window, this reduction of the sampling rate is accomplished by moving the window position $\hat{n}$ in jumps of more than one sample as shown in Figure 4.3.

Note that during the unvoiced interval, the zero-crossing rate is relatively high compared to the zero-crossing rate in the voiced interval. Conversely, the energy is relatively low in the unvoiced region compared to the energy in the voiced region. Note also that there is a small shift of the two curves relative to events in the time waveform. This is due to the time delay of M samples (equivalent to 12.5 ms) added to make the analysis window filter causal.

The short-time energy and short-time zero-crossing rate are important because they abstract valuable information about the speech signal, and they are simple to compute. The short-time energy is an indication of the amplitude of the signal in the interval around time $\hat{n}$. From our model, we expect unvoiced regions to have lower short-time energy than voiced regions. Similarly, the short-time zero-crossing rate is a crude frequency analyzer. Voiced signals have a high frequency (HF) falloff due to the lowpass nature of the glottal pulses, while unvoiced sounds have much more HF energy. Thus, the short-time energy and short-time zero-crossing rate can be the basis for an algorithm for making a decision as to whether the speech signal is voiced or unvoiced at

any particular time $\hat{n}$. A complete algorithm would involve measurements of the statistical distributions of the energy and zero-crossing rate for both voiced and unvoiced speech segments (and also background noise distributions). These distributions can be used to derive thresholds used in voiced/unvoiced decision [100].

4.2 Short-Time Autocorrelation Function (STACF)

The autocorrelation function is often used as a means of detecting periodicity in signals, and it is also the basis for many spectrum analysis methods. This makes it a useful tool for short-time speech analysis. The STACF is defined as the deterministic autocorrelation function of the sequence $x_{\hat{n}}[m] = x[m]w[\hat{n} - m]$ that is selected by the window shifted to time $\hat{n}$, i.e.,

$$\begin{aligned}\phi_{\hat{n}}[\ell] &= \sum_{m=-\infty}^{\infty} x_{\hat{n}}[m]x_{\hat{n}}[m+\ell] \\ &= \sum_{m=-\infty}^{\infty} x[m]w[\hat{n}-m]x[m+\ell]w[\hat{n}-m-\ell]. \end{aligned} \tag{4.9}$$

Using the familiar even-symmetric property of the autocorrelation, $\phi_{\hat{n}}[-\ell] = \phi_{\hat{n}}[\ell]$, (4.9) can be expressed in terms of linear time-invariant (LTI) filtering as

$$\phi_{\hat{n}}[\ell] = \sum_{m=-\infty}^{\infty} x[m]x[m-\ell]\tilde{w}_{\ell}[\hat{n}-m], \tag{4.10}$$

where $\tilde{w}_{\ell}[m] = w[m]w[m+\ell]$. Note that the STACF is a two-dimensional function of the discrete-time index $\hat{n}$ (the window position) and the discrete-lag index ℓ. If the window has finite duration, (4.9) can be evaluated directly or using FFT techniques (see Section 4.6). For infinite-duration decaying exponential windows, the short-time autocorrelation of (4.10) can be computed recursively at time $\hat{n}$ by using a different filter $\tilde{w}_{\ell}[m]$ for each lag value ℓ [8, 9].

To see how the short-time autocorrelation can be used in speech analysis, assume that a segment of the sampled speech signal is a segment of the output of the discrete-time model shown in Figure 4.1 where

the system is characterized at a particular analysis time by an impulse response $h[n]$, and the input is either a periodic impulse train or random white noise. (Note that we have suppressed the indication of analysis time.) Different segments of the speech signal will have the same form of model with different excitation and system impulse response. That is, assume that $s[n] = e[n] * h[n]$, where $e[n]$ is the excitation to the linear system with impulse response $h[n]$. A well known, and easily proved, property of the autocorrelation function is that

$$\phi^{(s)}[\ell] = \phi^{(e)}[\ell] * \phi^{(h)}[\ell], \tag{4.11}$$

i.e., the autocorrelation function of $s[n] = e[n] * h[n]$ is the convolution of the autocorrelation functions of $e[n]$ and $h[n]$. In the case of the speech signal, $h[n]$ represents the combined (by convolution) effects of the glottal pulse shape (for voiced speech), vocal tract shape, and radiation at the lips. For voiced speech, the autocorrelation of a periodic impulse train excitation with period P_0 is a periodic impulse train sequence with the same period. In this case, therefore, the autocorrelation of voiced speech is the periodic autocorrelation function

$$\phi^{(s)}[\ell] = \sum_{m=-\infty}^{\infty} \phi^{(h)}[\ell - mP_0]. \tag{4.12}$$

In the case of unvoiced speech, the excitation can be assumed to be random white noise, whose stochastic autocorrelation function would be an impulse sequence at $\ell = 0$. Therefore, the autocorrelation function of unvoiced speech computed using averaging would be simply

$$\phi^{(s)}[\ell] = \phi^{(h)}[\ell]. \tag{4.13}$$

Equation (4.12) assumes periodic computation of an infinite periodic signal, and (4.13) assumes probability average or averaging over an infinite time interval (for stationary random signals). However, the deterministic autocorrelation function of a finite-length segment of the speech waveform will have properties similar to those of (4.12) and (4.13) except that the correlation values will taper off with lag ℓ due to the tapering of the window and the fact that less and less data is involved in the computation of the short-time autocorrelation for longer

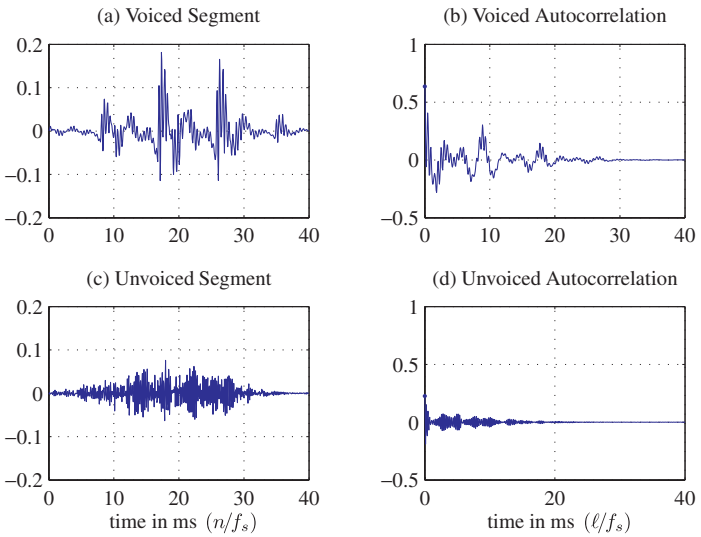

Fig. 4.5 Voiced and unvoiced segments of speech and their corresponding STACF.

lag values. This tapering off in level is depicted in Figure 4.5 for both a voiced and an unvoiced speech segment. Note the peak in the autocorrelation function for the voiced segment at the pitch period and twice the pitch period, and note the absence of such peaks in the autocorrelation function for the unvoiced segment. This suggests that the STACF could be the basis for an algorithm for estimating/detecting the pitch period of speech. Usually such algorithms involve the autocorrelation function and other short-time measurements such as zero-crossings and energy to aid in making the voiced/unvoiced decision.

Finally, observe that the STACF implicitly contains the short-time energy since

$$E_{\hat{n}} = \sum_{m=-\infty}^{\infty} (x[m]w[\hat{n}-m])^2 = \phi_{\hat{n}}[0]. \tag{4.14}$$

4.3 Short-Time Fourier Transform (STFT)

The short-time analysis functions discussed so far are examples of the general short-time analysis principle that is the basis for most

algorithms for speech processing. We now turn our attention to what is perhaps the most important basic concept in digital speech processing. In subsequent chapters, we will find that the STFT defined as

$$X_{\hat{n}}(e^{j\hat{\omega}}) = \sum_{m=-\infty}^{\infty} x[m]w[\hat{n}-m]e^{-j\hat{\omega}m}, \tag{4.15}$$

is the basis for a wide range of speech analysis, coding and synthesis systems. By definition, for fixed analysis time $\hat{n}$, the STFT is the discrete-time Fourier transform (DTFT) of the signal $x_{\hat{n}}[m] = x[m]w[\hat{n}-m]$, i.e., the DTFT of the (usually finite-duration) signal selected and amplitude-weighted by the sliding window $w[\hat{n}-m]$ [24, 89, 129]. Thus, the STFT is a function of two variables; $\hat{n}$ the discrete-time index denoting the window position, and $\hat{\omega}$ representing the analysis frequency.[7] Since (4.15) is a sequence of DTFTs, the two-dimensional function $X_{\hat{n}}(e^{j\hat{\omega}})$ at discrete-time $\hat{n}$ is a periodic function of continuous radian frequency $\hat{\omega}$ with period 2π [89].

As in the case of the other short-time analysis functions discussed in this chapter, the STFT can be expressed in terms of a linear filtering operation. For example, (4.15) can be expressed as the discrete convolution

$$X_{\hat{n}}(e^{j\hat{\omega}}) = (x[n]e^{-j\hat{\omega}n}) * w[n]\Big|_{n=\hat{n}}, \tag{4.16}$$

or, alternatively,

$$X_{\hat{n}}(e^{j\hat{\omega}}) = \big(x[n] * (w[n]e^{j\hat{\omega}n})\big)e^{-j\hat{\omega}n}\Big|_{n=\hat{n}}. \tag{4.17}$$

Recall that a typical window like a Hamming window, when viewed as a linear filter impulse response, has a lowpass frequency response with the cutoff frequency varying inversely with the window length. (See Figure 4.2(b).) This means that for a fixed value of $\hat{\omega}$, $X_{\hat{n}}(e^{j\hat{\omega}})$ is slowly varying as $\hat{n}$ varies. Equation (4.16) can be interpreted as follows: the amplitude modulation $x[n]e^{-j\hat{\omega}n}$ shifts the spectrum of

[7] As before, we use $\hat{n}$ to specify the analysis time, and $\hat{\omega}$ is used to distinguish the STFT analysis frequency from the frequency variable of the non-time-dependent Fourier transform frequency variable ω.

$x[n]$ down by $\hat{\omega}$, and the window (lowpass) filter selects the resulting band of frequencies around zero frequency. This is, of course, the band of frequencies of $x[n]$ that were originally centered on analysis frequency $\hat{\omega}$. An identical conclusion follows from (4.17): $x[n]$ is the input to a bandpass filter with impulse response $w[n]e^{j\hat{\omega}n}$, which selects the band of frequencies centered on $\hat{\omega}$. Then that band of frequencies is shifted down by the amplitude modulation with $e^{-j\hat{\omega}n}$, resulting again in the same lowpass signal [89].

In summary, the STFT has three interpretations: (1) It is a sequence of discrete-time Fourier transforms of windowed signal segments, i.e., a periodic function of $\hat{\omega}$ at each window position $\hat{n}$. (2) For each frequency $\hat{\omega}$ with $\hat{n}$ varying, it is the time sequence output of a lowpass filter that follows frequency down-shifting by $\hat{\omega}$. (3) For each frequency $\hat{\omega}$, it is the time sequence output resulting from frequency down-shifting the output of a bandpass filter.

4.4 Sampling the STFT in Time and Frequency

As defined in (4.15), the STFT is a function of a continuous analysis frequency $\hat{\omega}$. The STFT becomes a practical tool for both analysis and applications when it is implemented with a finite-duration window moved in steps of $R > 1$ samples in time and computed at a discrete set of frequencies as in

$$X_{rR}[k] = \sum_{m=rR-L+1}^{rR} x[m]w[rR-m]e^{-j(2\pi k/N)m} \quad k = 0, 1, \ldots, N-1, \tag{4.18}$$

where N is the number of uniformly spaced frequencies across the interval $0 \leq \hat{\omega} < 2\pi$, and L is the window length (in samples). Note that we have assumed that $w[m]$ is causal and nonzero only in the range $0 \leq m \leq L-1$ so that the windowed segment $x[m]w[rR-m]$ is nonzero over $rR - L + 1 \leq m \leq rR$. To aid in interpretation, it is helpful to write (4.18) in the equivalent form:

$$X_{rR}[k] = \tilde{X}_{rR}[k]e^{-j(2\pi k/N)rR} \quad k = 0, 1, \ldots, N-1, \tag{4.19}$$

where

$$\tilde{X}_{rR}[k] = \sum_{m=0}^{L-1} x[rR - m]w[m]e^{j(2\pi k/N)m} \quad k = 0, 1, \ldots, N-1. \quad (4.20)$$

Since we have assumed, for specificity, that $w[m] \neq 0$ only in the range $0 \leq m \leq L-1$, the alternative form, $\tilde{X}_{rR}[k]$, has the interpretation of an N-point DFT of the sequence $x[rR - m]w[m]$, which, due to the definition of the window, is nonzero in the interval $0 \leq m \leq L-1$.[8] In (4.20), the analysis time rR is shifted to the time origin of the DFT computation, and the segment of the speech signal is the time-reversed sequence of L samples that precedes the analysis time. The complex exponential factor $e^{-j(2\pi k/N)rR}$ in (4.19) results from the shift of the time origin.

For a discrete-time Fourier transform interpretation, it follows that $\tilde{X}_{rR}[k]$ can be computed by the following process:

(1) Form the sequence $x_{rR}[m] = x[rR - m]w[m]$, for $m = 0, 1, \ldots, L-1$.
(2) Compute the complex conjugate of the N-point DFT of the sequence $x_{rR}[m]$. (This can be done efficiently with an N-point FFT algorithm.)
(3) The multiplication by $e^{-j(2\pi k/N)rR}$ can be done if necessary, but often can be omitted (as in computing a sound spectrogram or spectrographic display).
(4) Move the time origin by R samples (i.e., $r \rightarrow r+1$) and repeat steps (1), (2), and (3), etc.

The remaining issue for complete specification of the sampled STFT is specification of the temporal sampling period, R, and the number of uniform frequencies, N. It can easily be shown that both R and N are determined entirely by the time width and frequency bandwidth of the lowpass window, $w[m]$, used to compute the STFT [1], giving the

[8] The DFT is normally defined with a negative exponent. Thus, since $x[rR - m]w[m]$ is real, (4.20) is the complex conjugate of the DFT of the windowed sequence [89].

following constraints on R and N:

(1) $R \leq L/(2C)$ where C is a constant that is dependent on the window frequency bandwidth; $C = 2$ for a Hamming window, $C = 1$ for a rectangular window.
(2) $N \geq L$, where L is the window length in samples.

Constraint (1) above is related to sampling the STFT in time at a rate of twice the window bandwidth in frequency in order to eliminate aliasing in frequency of the STFT, and constraint (2) above is related to sampling in frequency at a rate of twice the equivalent time width of the window to ensure that there is no aliasing in time of the STFT.

4.5 The Speech Spectrogram

Since the 1940s, the *sound spectrogram* has been a basic tool for gaining understanding of how the sounds of speech are produced and how phonetic information is encoded in the speech signal. Up until the 1970s, spectrograms were made by an ingenious device comprised of an audio tape loop, variable analog bandpass filter, and electrically sensitive paper [66]. Today spectrograms like those in Figure 4.6 are made by DSP techniques [87] and displayed as either pseudo-color or gray-scale images on computer screens.

Sound spectrograms like those in Figure 4.6 are simply a display of the magnitude of the STFT. Specifically, the images in Figure 4.6 are plots of

$$S(t_r, f_k) = 20\log_{10}|\tilde{X}_{rR}[k]| = 20\log_{10}|X_{rR}[k]|, \tag{4.21}$$

where the plot axes are labeled in terms of analog time and frequency through the relations $t_r = rRT$ and $f_k = k/(NT)$, where T is the sampling period of the discrete-time signal $x[n] = x_a(nT)$. In order to make smooth looking plots like those in Figure 4.6, R is usually quite small compared to both the window length L and the number of samples in the frequency dimension, N, which may be much larger than the window length L. Such a function of two variables can be plotted on a two dimensional surface (such as this text) as either a gray-scale or a color-mapped image. Figure 4.6 shows the time

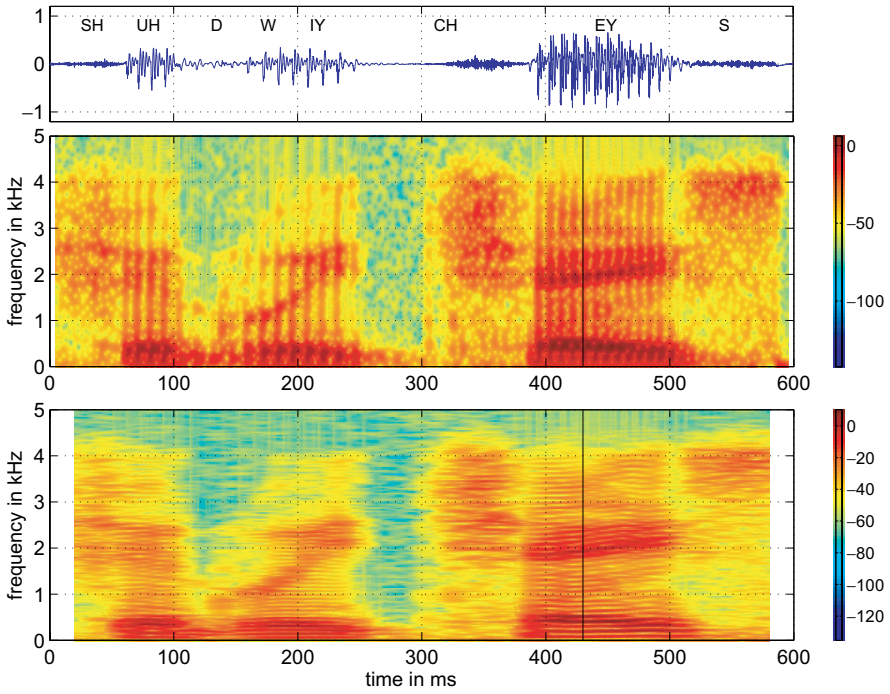

Fig. 4.6 Spectrogram for speech signal of Figure 1.1.

waveform at the top and two spectrograms computed with different length analysis windows. The bars on the right calibrate the color map (in dB).

A careful interpretation of (4.20) and the corresponding spectrogram images leads to valuable insight into the nature of the speech signal. First note that the window sequence $w[m]$ is nonzero only over the interval $0 \leq m \leq L-1$. The length of the window has a major effect on the spectrogram image. The upper spectrogram in Figure 4.6 was computed with a window length of 101 samples, corresponding to 10 ms time duration. This window length is on the order of the length of a pitch period of the waveform during voiced intervals. As a result, in voiced intervals, the spectrogram displays vertically oriented striations corresponding to the fact that the sliding window sometimes includes mostly large amplitude samples, then mostly small amplitude

samples, etc. As a result of the short analysis window, each individual pitch period is resolved in the time dimension, but the resolution in the frequency dimension is poor. For this reason, if the analysis window is short, the spectrogram is called a *wide-band spectrogram*. This is consistent with the linear filtering interpretation of the STFT, since a short analysis filter has a wide passband. Conversely, when the window length is long, the spectrogram is a *narrow-band spectrogram*, which is characterized by good frequency resolution and poor time resolution.

The upper plot in Figure 4.7, for example, shows $S(t_r, f_k)$ as a function of f_k at time $t_r = 430\,\text{ms}$. This vertical slice through the spectrogram is at the position of the black vertical line in the upper spectrogram of Figure 4.6. Note the three broad peaks in the spectrum slice at time $t_r = 430\,\text{ms}$, and observe that similar slices would

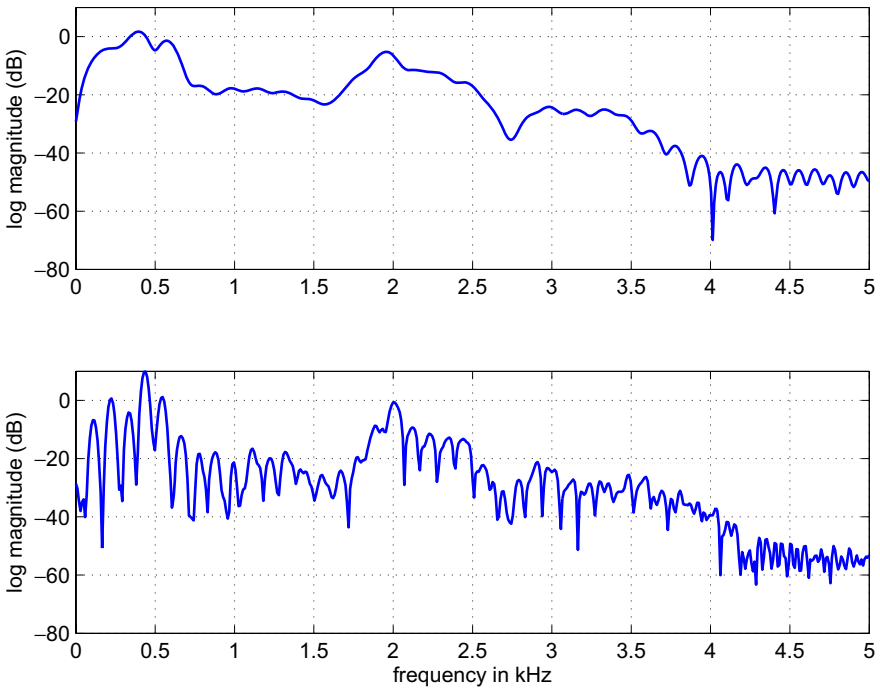

Fig. 4.7 Short-time spectrum at time 430 ms (dark vertical line in Figure 4.6) with Hamming window of length $M = 101$ in upper plot and $M = 401$ in lower plot.

be obtained at other times around $t_r = 430\,\text{ms}$. These large peaks are representative of the underlying resonances of the vocal tract at the corresponding time in the production of the speech signal.

The lower spectrogram in Figure 4.6 was computed with a window length of 401 samples, corresponding to 40 ms time duration. This window length is on the order of several pitch periods of the waveform during voiced intervals. As a result, the spectrogram no longer displays vertically oriented striations since several periods are included in the window no matter where it is placed on the waveform in the vicinity of the analysis time t_r. As a result, the spectrogram is not as sensitive to rapid time variations, but the resolution in the frequency dimension is much better. Therefore, the striations tend to be horizontally oriented in the narrow-band case since the fundamental frequency and its harmonics are all resolved. The lower plot in Figure 4.7, for example, shows $S(t_r, f_k)$ as a function of f_k at time $t_r = 430\,\text{ms}$. In this case, the signal is voiced at the position of the window, so over the analysis window interval it acts very much like a periodic signal. Periodic signals have Fourier spectra that are composed of impulses at the fundamental frequency, f_0, and at integer multiples (harmonics) of the fundamental frequency [89]. Multiplying by the analysis window in the time-domain results in convolution of the Fourier transform of the window with the impulses in the spectrum of the periodic signal [89]. This is evident in the lower plot of Figure 4.7, where the local maxima of the curve are spaced at multiples of the fundamental frequency $f_0 = 1/T_0$, where T_0 is the fundamental period (pitch period) of the signal.[9]

4.6 Relation of STFT to STACF

A basic property of Fourier transforms is that the inverse Fourier transform of the magnitude-squared of the Fourier transform of a signal is the autocorrelation function for that signal [89]. Since the STFT defined in (4.15) is a discrete-time Fourier transform for fixed window position, it follows that $\phi_{\hat{n}}[\ell]$ given by (4.9) is related to the STFT given by

[9] Each "ripple" in the lower plot is essentially a frequency-shifted copy of the Fourier transform of the Hamming window used in the analysis.

(4.15) as

$$\phi_{\hat{n}}[\ell] = \frac{1}{2\pi}\int_{-\pi}^{\pi} |X_{\hat{n}}(e^{j\hat{\omega}})|^2 e^{j\hat{\omega}\ell} d\hat{\omega}. \tag{4.22}$$

The STACF can also be computed from the sampled STFT. In particular, an inverse DFT can be used to compute

$$\tilde{\phi}_{rR}[\ell] = \frac{1}{N}\sum_{k=0}^{N-1} |\tilde{X}_{rR}(e^{j(2\pi k/N)})|^2 e^{j(2\pi k/N)\ell} \tag{4.23}$$

and $\tilde{\phi}_{rR}[\ell] = \phi_{rR}[\ell]$ for $\ell = 0, 1, \ldots, L-1$ if $N \geq 2L$. If $L < N < 2L$ time aliasing will occur, but $\tilde{\phi}_{rR}[\ell] = \phi_{rR}[\ell]$ for $\ell = 0, 1, \ldots, N-L$. Note that (4.14) shows that the short-time energy can also be obtained from either (4.22) or (4.23) by setting $\ell = 0$.

Figure 4.8 illustrates the equivalence between the STACF and the STFT. Figures 4.8(a) and 4.8(c) show the voiced and unvoiced autocorrelation functions that were shown in Figures 4.5(b) and 4.5(d). On the right are the corresponding STFTs. The peaks around 9 and 18 ms in

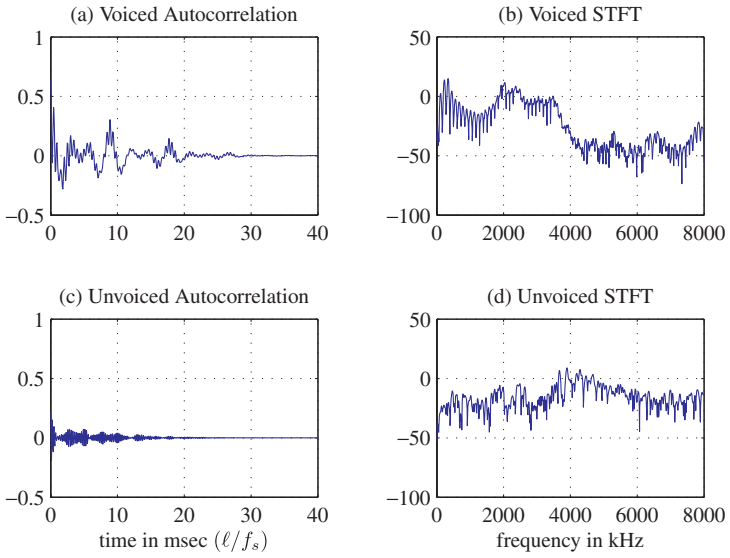

Fig. 4.8 STACF and corresponding STFT.

the autocorrelation function of the voiced segment imply a fundamental frequency at this time of approximately $1000/9 = 111$ Hz. Similarly, note in the STFT on the right in Figure 4.8(b) that there are approximately 18 local regularly spaced peaks in the range 0–2000 Hz. Thus, we can estimate the fundamental frequency to be $2000/18 = 111$ Hz as before.

4.7 Short-Time Fourier Synthesis

From the linear filtering point of view of the STFT, (4.19) and (4.20) represent the downsampled (by the factor R) output of the process of bandpass filtering with impulse response $w[n]e^{j(2\pi k/N)n}$ followed by frequency-down-shifting by $(2\pi k/N)$. Alternatively, (4.18) shows that $X_{rR}[k]$ is a downsampled output of the lowpass window filter with frequency-down-shifted input $x[n]e^{-j(2\pi k/N)n}$. The left half of Figure 4.9 shows a block diagram representation of the STFT as a combination of modulation, followed by lowpass filtering, followed by a downsampler by R. This structure is often called a *filter bank*, and the outputs of the individual filters are called the *channel signals*. The

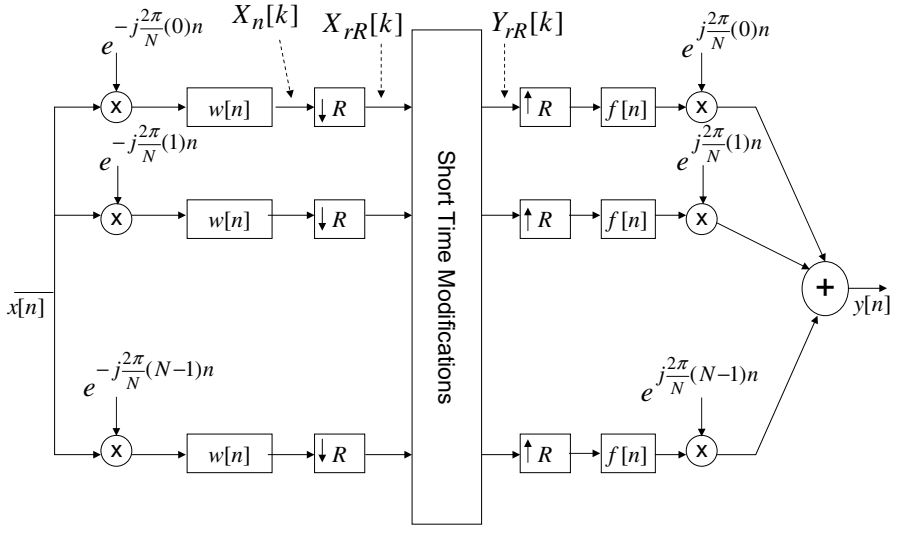

Fig. 4.9 Filterbank interpretation of short-time Fourier analysis and synthesis.

spectrogram is comprised of the set of outputs of the filter bank with each channel signal corresponding to a horizontal line in the spectrogram. If we want to use the STFT for other types of speech processing, we may need to consider if and how it is possible to reconstruct the speech signal from the STFT, i.e., from the channel signals. The remainder of Figure 4.9 shows how this can be done.

First, the diagram shows the possibility that the STFT might be modified by some sort of processing. An example might be quantization of the channel signals for data compression. The modified STFT is denoted as $Y_{rR}[k]$. The remaining parts of the structure on the right in Figure 4.9 implement the synthesis of a new time sequence $y[n]$ from the STFT. This part of the diagram represents the synthesis equation

$$y[n] = \sum_{k=0}^{N-1} \left(\sum_{r=-\infty}^{\infty} Y_{rR}[k] f[n - rR] \right) e^{j(2\pi k/N)n}. \tag{4.24}$$

The steps represented by (4.24) and the right half of Figure 4.9 involve first upsampling by R followed by linear filtering with $f[n]$ (called the synthesis window). This operation, defined by the part within the parenthesis in (4.24), interpolates the STFT $Y_{rR}[k]$ to the time sampling rate of the speech signal for each channel k.[10] Then, synthesis is achieved by modulation with $e^{j(2\pi k/N)n}$, which up-shifts the lowpass interpolated channel signals back to their original frequency bands centered on frequencies $(2\pi k/N)$. The sum of the up-shifted signals is the synthesized output.

There are many variations on the short-time Fourier analysis/synthesis paradigm. The FFT algorithm can be used to implement both analysis and synthesis; special efficiencies result when $L > R = N$, and the analysis and synthesis can be accomplished using only real filtering and with modulation operations being implicitly achieved by the downsampling [24, 129]. However, it is most important to note that with careful choice of the parameters L, R, and N and careful design of the analysis window $w[m]$ together with the synthesis window $f[n]$, it is possible to reconstruct the signal $x[n]$ with negligible

[10] The sequence $f[n]$ is the impulse response of a LTI filter with gain R/N and normalized cutoff frequency π/R. It is often a scaled version of $w[n]$.

error ($y[n] = x[n]$) from the unmodified STFT $Y_{rR}[k] = X_{rR}[k]$. One condition that guarantees exact reconstruction is [24, 129].

$$\sum_{r=-\infty}^{\infty} w[rR - n + qN]f[n - rR] = \begin{cases} 1 & q = 0 \\ 0 & q \neq 0. \end{cases} \tag{4.25}$$

Short-time Fourier analysis and synthesis is generally formulated with equally spaced channels with equal bandwidths. However, we have seen that models for auditory processing involve nonuniform filter banks. Such filter banks can be implemented as tree structures where frequency bands are successively divided into low- and HF bands. Such structures are essentially the same as wavelet decompositions, a topic beyond the scope of this text, but one for which a large body of literature exists. See, for example [18, 125, 129].

4.8 Short-Time Analysis is Fundamental to our Thinking

It can be argued that the short-time analysis principle, and particularly the short-time Fourier representation of speech, is fundamental to our thinking about the speech signal and it leads us to a wide variety of techniques for achieving our goal of moving from the sampled time waveform back along the speech chain toward the implicit message. The fact that almost perfect reconstruction can be achieved from the filter bank channel signals gives the short-time Fourier representation major credibility in the digital speech processing tool kit. This importance is strengthened by the fact that, as discussed briefly in Chapter 3, models for auditory processing are based on a filter bank as the first stage of processing. Much of our knowledge of perceptual effects is framed in terms of frequency analysis, and thus, the STFT representation provides a natural framework within which this knowledge can be represented and exploited to obtain efficient representations of speech and more general audio signals. We will have more to say about this in Chapter 7, but first we will consider (in Chapter 5) the technique of cepstrum analysis, which is based directly on the STFT, and the technique of linear predictive analysis (in Chapter 6), which is based on the STACF and thus equivalently on the STFT.

5

Homomorphic Speech Analysis

The STFT provides a useful framework for thinking about almost all the important techniques for analysis of speech that have been developed so far. An important concept that flows directly from the STFT is the *cepstrum*, more specifically, the *short-time cepstrum*, of speech. In this chapter, we explore the use of the short-time cepstrum as a representation of speech and as a basis for estimating the parameters of the speech generation model. A more detailed discussion of the uses of the cepstrum in speech processing can be found in [110].

5.1 Definition of the Cepstrum and Complex Cepstrum

The cepstrum was defined by Bogert, Healy, and Tukey to be the inverse Fourier transform of the log magnitude spectrum of a signal [16]. Their original definition, loosely framed in terms of spectrum analysis of analog signals, was motivated by the fact that the logarithm of the Fourier spectrum of a signal containing an echo has an additive periodic component depending only on the echo size and delay, and that further Fourier analysis of the log spectrum can aid in detecting the presence

of that echo. Oppenheim, Schafer, and Stockham showed that the cepstrum is related to the more general concept of homomorphic filtering of signals that are combined by convolution [85, 90, 109]. They gave a definition of the cepstrum of a discrete-time signal as

$$c[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log|X(e^{j\omega})|e^{j\omega n}d\omega, \tag{5.1a}$$

where $\log|X(e^{j\omega})|$ is the logarithm of the magnitude of the DTFT of the signal, and they extended the concept by defining the *complex cepstrum* as

$$\hat{x}[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\{X(e^{j\omega})\}e^{j\omega n}d\omega, \tag{5.1b}$$

where $\log\{X(e^{j\omega})\}$ is the complex logarithm of $X(e^{j\omega})$ defined as

$$\hat{X}(e^{j\omega}) = \log\{X(e^{j\omega})\} = \log|X(e^{j\omega})| + j\arg[X(e^{j\omega})]. \tag{5.1c}$$

The transformation implied by (5.1b) is depicted as the block diagram in Figure 5.1. The same diagram represents the cepstrum if the complex logarithm is replaced by the logarithm of the magnitude of the DTFT. Since we restrict our attention to real sequences, $x[n]$, it follows from the symmetry properties of Fourier transforms that the cepstrum is the even part of the complex cepstrum, i.e., $c[n] = (\hat{x}[n] + \hat{x}[-n])/2$ [89].

As shown in Figure 5.1, the operation of computing the complex cepstrum from the input can be denoted as $\hat{x}[n] = D_*\{x[n]\}$. In the theory of homomorphic systems $D_*\{\ \}$ is called the *characteristic system for convolution*. The connection between the cepstrum concept and homomorphic filtering of convolved signals is that the complex cepstrum has the property that if $x[n] = x_1[n] * x_2[n]$, then

$$\hat{x}[n] = D_*\{x_1[n] * x_2[n]\} = \hat{x}_1[n] + \hat{x}_2[n]. \tag{5.2}$$

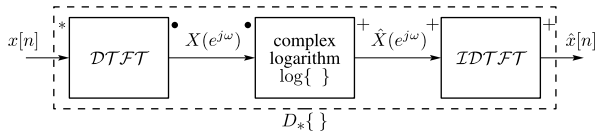

Fig. 5.1 Computing the complex cepstrum using the DTFT.

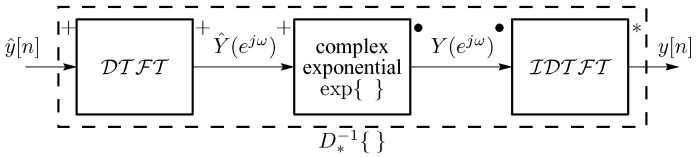

Fig. 5.2 The inverse of the characteristic system for convolution (inverse complex cepstrum).

That is, the complex cepstrum operator transforms convolution into addition. This property, which is true for both the cepstrum and the complex cepstrum, is what makes the cepstrum and the complex cepstrum useful for speech analysis, since our model for speech production involves convolution of the excitation with the vocal tract impulse response, and our goal is often to separate the excitation signal from the vocal tract signal. In the case of the complex cepstrum, the inverse of the characteristic system exists as in Figure 5.2, which shows the reverse cascade of the inverses of the operators in Figure 5.1. Homomorphic filtering of convolved signals is achieved by forming a modified complex cepstrum

$$\hat{y}[n] = g[n]\hat{x}[n], \tag{5.3}$$

where $g[n]$ is a window (a "lifter" in the terminology of Bogert et al.) which selects a portion of the complex cepstrum for inverse processing. A modified output signal $y[n]$ can then be obtained as the output of Figure 5.2 with $\hat{y}[n]$ given by (5.3) as input. Observe that (5.3) defines a linear operator in the conventional sense, i.e., if $\hat{x}[n] = \hat{x}_1[n] + \hat{x}_2[n]$ then $\hat{y}[n] = g[n]\hat{x}_1[n] + g[n]\hat{x}_2[n]$. Therefore, the output of the inverse characteristic system will have the form $y[n] = y_1[n] * y_2[n]$, where $\hat{y}_1[n] = g[n]\hat{x}_1[n]$ is the complex cepstrum of $y_1[n]$, etc. Examples of lifters used in homomorphic filtering of speech are given in Sections 5.4 and 5.6.2.

The key issue in the definition and computation of the complex cepstrum is the computation of the complex logarithm; more specifically, the computation of the phase angle $\arg[X(e^{j\omega})]$, which must be done so as to preserve an additive combination of phases for two signals combined by convolution [90, 109].

The independent variable of the cepstrum and complex cepstrum is nominally *time*. The crucial observation leading to the cepstrum concept is that the log spectrum can be treated as a waveform to be subjected to further Fourier analysis. To emphasize this interchanging of domains of reference, Bogert et al. [16] coined the word *cepstrum* by transposing some of the letters in the word *spectrum*. They created many other special terms in this way including *quefrency* as the name for the independent variable of the cepstrum and *liftering* for the operation of linear filtering the log magnitude spectrum by the operation of (5.3). Only the terms cepstrum, quefrency, and liftering are widely used today.

5.2 The Short-Time Cepstrum

The application of these definitions to speech requires that the DTFT be replaced by the STFT. Thus the short-time cepstrum is defined as

$$c_{\hat{n}}[m] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log|X_{\hat{n}}(e^{j\hat{\omega}})|e^{j\hat{\omega}m}d\hat{\omega}, \tag{5.4}$$

where $X_{\hat{n}}(e^{j\hat{\omega}})$ is the STFT defined in (4.15), and the short-time complex cepstrum is likewise defined by replacing $X(e^{j\hat{\omega}})$ by $X_{\hat{n}}(e^{j\hat{\omega}})$ in (5.1b).[1] The similarity to the STACF defined by (4.9) should be clear. The short-time cepstrum is a sequence of cepstra of windowed finite-duration segments of the speech waveform. By analogy, a "cepstrogram" would be an image obtained by plotting the magnitude of the short-time cepstrum as a function of quefrency m and analysis time $\hat{n}$.

5.3 Computation of the Cepstrum

In (5.1a) and (5.1b), the cepstrum and complex cepstrum are defined in terms of the DTFT. This is useful in the basic definitions, but not

[1] In cases where we wish to explicitly indicate analysis-time dependence of the short-time cepstrum, we will use $\hat{n}$ for the analysis time and m for quefrency as in (5.4), but as in other instances, we often suppress the subscript $\hat{n}$.

for use in processing sampled speech signals. Fortunately, several computational options exist.

5.3.1 Computation Using the DFT

Since the DFT (computed with an FFT algorithm) is a sampled (in frequency) version of the DTFT of a finite-length sequence (i.e., $X[k] = X(e^{j2\pi k/N})$ [89]), the DFT and inverse DFT can be substituted for the DTFT and its inverse in (5.1a) and (5.1b) as shown in Figure 5.3, which shows that the complex cepstrum can be computed approximately using the equations

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi k/N)n} \tag{5.5a}$$

$$\hat{X}[k] = \log|X[k]| + j \arg\{X[k]\} \tag{5.5b}$$

$$\tilde{\hat{x}}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \hat{X}[k] e^{j(2\pi k/N)n}. \tag{5.5c}$$

Note the "tilde" ˜ symbol above $\hat{x}[n]$ in (5.5c) and in Figure 5.3. Its purpose is to emphasize that using the DFT instead of the DTFT results in an approximation to (5.1b) due to the time-domain aliasing resulting from the sampling of the log of the DTFT [89]. That is,

$$\tilde{\hat{x}}[n] = \sum_{r=-\infty}^{\infty} \hat{x}[n + rN], \tag{5.6}$$

where $\hat{x}[n]$ is the complex cepstrum defined by (5.1b). An identical equation holds for the time-aliased cepstrum $\tilde{c}[n]$.

The effect of time-domain aliasing can be made negligible by using a large value for N. A more serious problem in computation of the complex cepstrum is the computation of the complex logarithm. This is

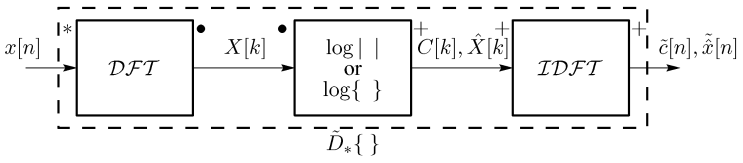

Fig. 5.3 Computing the cepstrum or complex cepstrum using the DFT.

because the angle of a complex number is usually specified modulo 2π, i.e., by the *principal value*. In order for the complex cepstrum to be evaluated properly, the phase of the sampled DTFT must be evaluated as samples of a continuous function of frequency. If the principal value phase is first computed at discrete frequencies, then it must be "unwrapped" modulo 2π in order to ensure that convolutions are transformed into additions. A variety of algorithms have been developed for phase unwrapping [90, 109, 127]. While accurate phase unwrapping presents a challenge in computing the complex cepstrum, it is not a problem in computing the cepstrum, since the phase is not used. Furthermore, phase unwrapping can be avoided by using numerical computation of the poles and zeros of the z-transform.

5.3.2 z-Transform Analysis

The characteristic system for convolution can also be represented by the two-sided z-transform as depicted in Figure 5.4. This is very useful for theoretical investigations, and recent developments in polynomial root finding [117] have made the z-transform representation a viable computational basis as well. For this purpose, we assume that the input signal $x[n]$ has a rational z-transform of the form:

$$X(z) = X_{\max}(z)X_{uc}(z)X_{\min}(z), \tag{5.7}$$

where

$$X_{\max}(z) = z^{M_o}\prod_{k=1}^{M_o}(1 - a_k z^{-1}) = \prod_{k=1}^{M_o}(-a_k)\prod_{k=1}^{M_o}(1 - a_k^{-1}z), \tag{5.8a}$$

$$X_{uc}(z) = \prod_{k=1}^{M_{uc}}(1 - e^{j\theta_k}z^{-1}), \tag{5.8b}$$

$$X_{\min}(z) = A\frac{\prod_{k=1}^{M_i}(1 - b_k z^{-1})}{\prod_{k=1}^{N_i}(1 - c_k z^{-1})}. \tag{5.8c}$$

The zeros of $X_{\max}(z)$, i.e., $z_k = a_k$, are zeros of $X(z)$ outside of the unit circle ($|a_k| > 1$). $X_{\max}(z)$ is thus the *maximum-phase* part of $X(z)$.

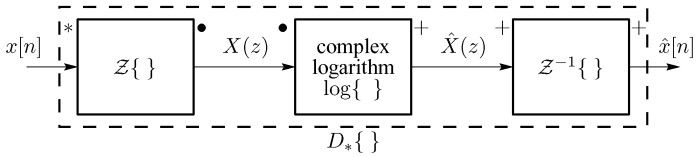

Fig. 5.4 z-transform representation of characteristic system for convolution.

$X_{uc}(z)$ contains all the zeros (with angles θ_k) *on* the unit circle. The *minimum-phase* part is $X_{\min}(z)$, where b_k and c_k are zeros and poles, respectively, that are inside the unit circle ($|b_k| < 1$ and $|c_k| < 1$). The factor z^{Mo} implies a shift of M_o samples to the left. It is included to simplify the results in (5.11).

The complex cepstrum of $x[n]$ is determined by assuming that the complex logarithm $\log\{X(z)\}$ results in the sum of logarithms of each of the product terms, i.e.,

$$\hat{X}(z) = \log\left|\prod_{k=1}^{M_o}(-a_k)\right| + \sum_{k=1}^{M_o}\log(1-a_k^{-1}z) + \sum_{k=1}^{M_{uc}}\log(1-e^{j\theta_k}z^{-1}) + \log|A| + \sum_{k=1}^{M_i}\log(1-b_kz^{-1}) - \sum_{k=1}^{N_i}\log(1-c_kz^{-1}). \quad (5.9)$$

Applying the power series expansion

$$\log(1-a) = -\sum_{n=1}^{\infty}\frac{a^n}{n} \quad |a| < 1 \quad (5.10)$$

to each of the terms in (5.9) and collecting the coefficients of the positive and negative powers of z gives

$$\hat{x}[n] = \begin{cases} \displaystyle\sum_{k=1}^{M_o}\frac{a_k^n}{n} & n < 0 \\ \displaystyle\log|A| + \log\left|\prod_{k=1}^{m_o}(-a_k)\right| & n = 0 \\ \displaystyle\left(-\sum_{k=1}^{M_{uc}}\frac{e^{j\theta_k n}}{n} - \sum_{k=1}^{M_i}\frac{b_k^n}{n} + \sum_{k=1}^{N_i}\frac{c_k^n}{n}\right) & n > 0. \end{cases} \quad (5.11)$$

Given all the poles and zeros of a z-transform $X(z)$, (5.11) allows us to compute the complex cepstrum with no approximation. This is

the case in theoretical analysis where the poles and zeros are specified. However, (5.11) is also useful as the basis for computation. All that is needed is a process for obtaining the z-transform as a rational function and a process for finding the zeros of the numerator and denominator. This has become more feasible with increasing computational power and with new advances in finding roots of large polynomials [117].

One method of obtaining a z-transform is simply to select a finite-length sequence of samples of a signal. The z-transform is then simply a polynomial with the samples $x[n]$ as coefficients, i.e.,

$$X(z) = \sum_{n=0}^{M} x[n]z^{-n} = A\prod_{k=1}^{M_o}(1 - a_k z^{-1})\prod_{k=1}^{M_i}(1 - b_k z^{-1}). \quad (5.12)$$

A second method that yields a z-transform is the method of linear predictive analysis, to be discussed in Chapter 6.

5.3.3 Recursive Computation of the Complex Cepstrum

Another approach to computing the complex cepstrum applies only to minimum-phase signals, i.e., signals having a z-transform whose poles and zeros are inside the unit circle. An example would be the impulse response of an all-pole vocal tract model with system function

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p}\alpha_k z^{-k}} = \frac{G}{\prod_{k=1}^{p}(1 - c_k z^{-1})}. \quad (5.13)$$

Such models are implicit in the use of linear predictive analysis of speech (Chapter 6). In this case, all the poles c_k must be inside the unit circle for stability of the system. From (5.11) it follows that the complex cepstrum of the impulse response $h[n]$ corresponding to $H(z)$ is

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ \log|G| & n = 0 \\ \sum_{k=1}^{p}\frac{c_k^n}{n} & n > 0. \end{cases} \quad (5.14)$$

It can be shown [90] that the impulse response and its complex cepstrum are related by the recursion formula:

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ \log G & n = 0 \\ \dfrac{h[n]}{h[0]} - \displaystyle\sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \frac{\hat{h}[k]h[n-k]}{h[0]} & n \geq 1. \end{cases} \tag{5.15}$$

Furthermore, working with the reciprocal (negative logarithm) of (5.13), it can be shown that there is a direct recursive relationship between the coefficients of the denominator polynomial in (5.13) and the complex cepstrum of the impulse response of the model filter, i.e.,

$$\hat{h}[n] = \begin{cases} 0 & n < 0 \\ \log G & n = 0 \\ \alpha_n + \displaystyle\sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \hat{h}[k]\alpha_{n-k} & n > 0. \end{cases} \tag{5.16}$$

From (5.16) it follows that the coefficients of the denominator polynomial can be obtained from the complex cepstrum through

$$\alpha_n = \hat{h}[n] - \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \hat{h}[k]\alpha_{n-k} \quad 1 \leq n \leq p. \tag{5.17}$$

From (5.17), it follows that $p+1$ values of the complex cepstrum are sufficient to fully determine the speech model system in (5.13) since all the denominator coefficients and G can be computed from $\hat{h}[n]$ for $n = 0, 1, \ldots, p$ using (5.17). This fact is the basis for the use of the cepstrum in speech coding and speech recognition as a vector representation of the vocal tract properties of a frame of speech.

5.4 Short-Time Homomorphic Filtering of Speech

Figure 5.5 shows an example of the short-time cepstrum of speech for the segments of voiced and unvoiced speech in Figures 4.5(a) and 4.5(c). The low quefrency part of the cepstrum is expected to be representative of the slow variations (with frequency) in the log spectrum, while the

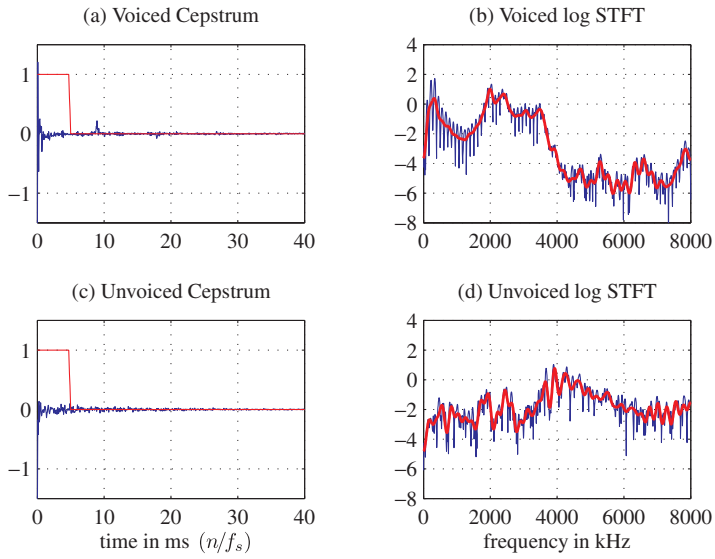

Fig. 5.5 Short-time cepstra and corresponding STFTs and homomorphically-smoothed spectra.

high quefrency components would correspond to the more rapid fluctuations of the log spectrum. This is illustrated by the plots in Figure 5.5. The log magnitudes of the corresponding short-time spectra are shown on the right as Figures 5.5(b) and 5.5(d). Note that the spectrum for the voiced segment in Figure 5.5(b) has a structure of periodic ripples due to the harmonic structure of the quasi-periodic segment of voiced speech. This periodic structure in the log spectrum of Figure 5.5(b), manifests itself in the cepstrum peak at a quefrency of about 9 ms in Figure 5.5(a). The existence of this peak in the quefrency range of expected pitch periods strongly signals voiced speech. Furthermore, the quefrency of the peak is an accurate estimate of the pitch period during the corresponding speech interval. As shown in Figure 4.5(b), the autocorrelation function also displays an indication of periodicity, but not nearly as unambiguously as does the cepstrum. On the other hand, note that the rapid variations of the unvoiced spectra appear random with no periodic structure. This is typical of Fourier transforms (periodograms) of short segments of random signals. As a result, there is no strong peak indicating periodicity as in the voiced case.

To illustrate the effect of liftering, the quefrencies above 5 ms are multiplied by zero and the quefrencies below 5 ms are multiplied by 1 (with a short transition taper as shown in Figures 5.5(a) and 5.5(c)). The DFT of the resulting modified cepstrum is plotted as the smooth curve that is superimposed on the short-time spectra in Figures 5.5(b) and 5.5(d), respectively. These slowly varying log spectra clearly retain the general spectral shape with peaks corresponding to the formant resonance structure for the segment of speech under analysis. Therefore, a useful perspective is that by liftering the cepstrum, it is possible to separate information about the vocal tract contributions from the short-time speech spectrum [88].

5.5 Application to Pitch Detection

The cepstrum was first applied in speech processing to determine the excitation parameters for the discrete-time speech model of Figure 7.10. Noll [83] applied the short-time cepstrum to detect local periodicity (voiced speech) or the lack thereof (unvoiced speech). This is illustrated in Figure 5.6, which shows a plot that is very similar to the plot first published by Noll [83]. On the left is a sequence of log short-time spectra (rapidly varying curves) and on the right is the corresponding sequence of cepstra computed from the log spectra on the left. The successive spectra and cepstra are for 50 ms segments obtained by moving the window in steps of 12.5 ms (100 samples at a sampling rate of 8000 samples/sec). From the discussion of Section 5.4, it is apparent that for the positions 1 through 5, the window includes only unvoiced speech, while for positions 6 and 7 the signal within the window is partly voiced and partly unvoiced. For positions 8 through 15 the window only includes voiced speech. Note again that the rapid variations of the unvoiced spectra appear random with no periodic structure. On the other hand, the spectra for voiced segments have a structure of periodic ripples due to the harmonic structure of the quasi-periodic segment of voiced speech. As can be seen from the plots on the right, the cepstrum peak at a quefrency of about 11–12 ms strongly signals voiced speech, and the quefrency of the peak is an accurate estimate of the pitch period during the corresponding speech interval.

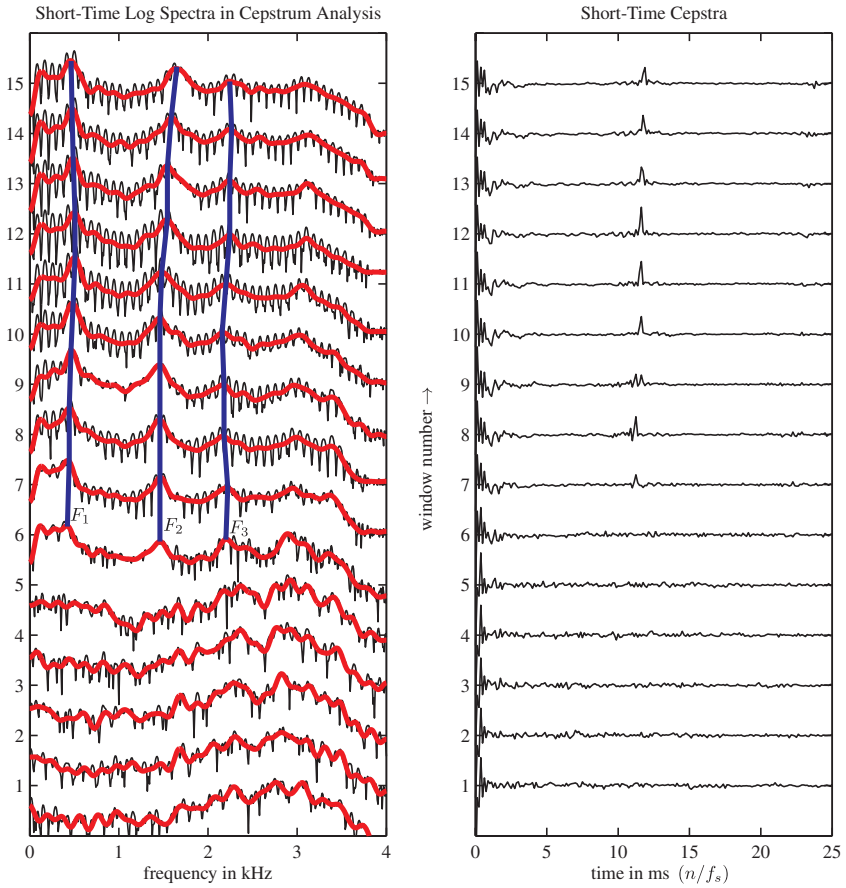

Fig. 5.6 Short-time cepstra and corresponding STFTs and homomorphically-smoothed spectra.

The essence of the pitch detection algorithm proposed by Noll is to compute a sequence of short-time cepstra and search each successive cepstrum for a peak in the quefrency region of the expected pitch period. Presence of a strong peak implies voiced speech, and the quefrency location of the peak gives the estimate of the pitch period. As in most model-based signal processing applications of concepts such as the cepstrum, the pitch detection algorithm includes many features designed to handle cases that do not fit the underlying model very well. For example, for frames 6 and 7 the cepstrum peak is weak

corresponding to the transition from unvoiced to voiced speech. In other problematic cases, the peak at twice the pitch period may be stronger than the peak at the quefrency of the pitch period. Noll applied temporal continuity constraints to prevent such errors.

5.6 Applications to Pattern Recognition

Perhaps the most pervasive application of the cepstrum in speech processing is its use in pattern recognition systems such as design of vector quantizers (VQ) and automatic speech recognizers (ASR). In such applications, a speech signal is represented on a frame-by-frame basis by a sequence of short-time cepstra. As we have shown, cepstra can be computed either by z-transform analysis or by DFT implementation of the characteristic system. In either case, we can assume that the cepstrum vector corresponds to a gain-normalized ($c[0] = 0$) minimum-phase vocal tract impulse response that is defined by the complex cepstrum[2]

$$\hat{h}[n] = \begin{cases} 2c[n] & 1 \leq n \leq n_{co} \\ 0 & n < 0. \end{cases} \tag{5.18}$$

In problems such as VQ or ASR, a test pattern $c[n]$ (vector of cepstrum values $n = 1, 2, \ldots, n_{co}$) is compared against a similarly defined reference pattern $\bar{c}[n]$. Such comparisons require a distance measure. For example, the Euclidean distance applied to the cepstrum would give

$$D = \sum_{n=1}^{n_{co}} |c[n] - \bar{c}[n]|^2 . \tag{5.19a}$$

Equivalently in the frequency domain,

$$D = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \log|H(e^{j\hat{\omega}})| - \log|\bar{H}(e^{j\hat{\omega}})| \right|^2 d\hat{\omega}, \tag{5.19b}$$

where $\log|H(e^{j\hat{\omega}})|$ is the log magnitude of the DTFT of $h[n]$ corresponding to the complex cepstrum in (5.18) or the real part of the

[2] For minimum-phase signals, the complex cepstrum satisfies $h[n] = 0$ for $n < 0$. Since the cepstrum is always the even part of the complex cepstrum, it follows that $\hat{h}[n] = 2c[n]$ for $n > 0$.

DTFT of $\hat{h}[n]$ in (5.18). Thus, cepstrum-based comparisons are strongly related to comparisons of smoothed short-time spectra. Therefore, the cepstrum offers an effective and flexible representation of speech for pattern recognition problems, and its interpretation as a difference of log spectra suggests a match to auditory perception mechanisms.

5.6.1 Compensation for Linear Filtering

Suppose that we have only a linearly filtered version of the speech signal, $y[n] = h[n] * x[n]$, instead of $x[n]$. If the analysis window is long compared to the length of $h[n]$, the short-time cepstrum of one frame of the filtered speech signal $y[n]$ will be[3]

$$c_{\hat{n}}^{(y)}[m] = c_{\hat{n}}^{(x)}[m] + c^{(h)}[m], \tag{5.20}$$

where $c^{(h)}[m]$ will appear essentially the same in each frame. Therefore, if we can estimate $c^{(h)}[n]$,[4] which we assume is non-time-varying, we can obtain $c_{\hat{n}}^{(x)}[m]$ at each frame from $c_{\hat{n}}^{(y)}[m]$ by subtraction, i.e., $c_n^{(x)}[m] = c_{\hat{n}}^{(y)}[m] - c^{(h)}[m]$. This property is extremely attractive in situations where the reference pattern $\bar{c}[m]$ has been obtained under different recording or transmission conditions from those used to acquire the test vector. In these circumstances, the test vector can be compensated for the effects of the linear filtering prior to computing the distance measures used for comparison of patterns.

Another approach to removing the effects of linear distortions is to observe that the cepstrum component due to the distortion is the same in each frame. Therefore, it can be removed by a simple first difference operation of the form:

$$\Delta c_{\hat{n}}^{(y)}[m] = c_{\hat{n}}^{(y)}[m] - c_{\hat{n}-1}^{(y)}[m]. \tag{5.21}$$

It is clear that if $c_{\hat{n}}^{(y)}[m] = c_{\hat{n}}^{(x)}[m] + c^{(h)}[m]$ with $c^{(h)}[m]$ being independent of $\hat{n}$, then $\Delta c_{\hat{n}}^{(y)}[m] = \Delta c_{\hat{n}}^{(x)}[m]$, i.e., the linear distortion effects are removed.

[3] In this section, it will be useful to use somewhat more complicated notation. Specifically, we denote the cepstrum at analysis time $\hat{n}$ of a signal $x[n]$ as $c_{\hat{n}}^{(x)}[m]$, where m denotes the quefrency index of the cepstrum.

[4] Stockham [124] showed how $c^{(h)}[m]$ for such linear distortions can be estimated from the signal $y[n]$ by time averaging the log of the STFT.

Furui [39, 40] first noted that the sequence of cepstrum values has temporal information that could be of value for a speaker verification system. He used polynomial fits to cepstrum sequences to extract simple representations of the temporal variation. The *delta cepstrum* as defined in (5.21) is simply the slope of a first-order polynomial fit to the cepstrum time evolution.

5.6.2 Liftered Cepstrum Distance Measures

In using linear predictive analysis (discussed in Chapter 6) to obtain cepstrum feature vectors for pattern recognition problems, it is observed that there is significant statistical variability due to a variety of factors including short-time analysis window position, bias toward harmonic peaks, and additive noise [63, 126]. A solution to this problem is to use weighted distance measures of the form:

$$D = \sum_{n=1}^{n_{co}} g^2[n] \, |c[n] - \bar{c}[n]|^2 \,, \tag{5.22a}$$

which can be written as the Euclidean distance of liftered cepstra

$$D = \sum_{n=1}^{n_{co}} |g[n]c[n] - g[n]\bar{c}[n]|^2 \,. \tag{5.22b}$$

Tohkura [126] found, for example, that when averaged over many frames of speech and speakers, cepstrum values $c[n]$ have zero means and variances on the order of $1/n^2$. This suggests that $g[n] = n$ for $n = 1, 2, \ldots, n_{co}$ could be used to equalize the contributions for each term to the cepstrum difference.

Juang et al. [63] observed that the variability due to the vagaries of LPC analysis could be lessened by using a lifter of the form:

$$g[n] = 1 + 0.5 n_{co} \sin(\pi n / n_{co}) \quad n = 1, 2, \ldots, n_{co}. \tag{5.23}$$

Tests of weighted distance measures showed consistent improvements in automatic speech recognition tasks.

Itakura and Umezaki [55] used the group delay function to derive a different cepstrum weighting function. Instead of $g[n] = n$ for all n, or

the lifter of (5.23), Itakura proposed the lifter

$$g[n] = n^s e^{-n^2/2\tau^2}. \tag{5.24}$$

This lifter has great flexibility. For example, if $s = 0$ we have simply lowpass liftering of the cepstrum. If $s = 1$ and τ is large, we have essentially $g[n] = n$ for small n with high quefrency tapering. Itakura and Umezaki [55] tested the group delay spectrum distance measure in an automatic speech recognition system. They found that for clean test utterances, the difference in recognition rate was small for different values of s when $\tau \approx 5$ although performance suffered with increasing s for larger values of τ. This was attributed to the fact that for larger s the group delay spectrum becomes very sharply peaked and thus more sensitive to small differences in formant locations. However, in test conditions with additive white noise and also with linear filtering distortions, recognition rates improved significantly with $\tau = 5$ and increasing values of the parameter s.

5.6.3 Mel-Frequency Cepstrum Coefficients

As we have seen, weighted cepstrum distance measures have a directly equivalent interpretation in terms of distance in the frequency domain. This is significant in light of models for human perception of sound, which, as noted in Chapter 3, are based upon a frequency analysis performed in the inner ear. With this in mind, Davis and Mermelstein [27] formulated a new type of cepstrum representation that has come to be widely used and known as the mel-frequency cepstrum coefficients (mfcc).

The basic idea is to compute a frequency analysis based upon a filter bank with approximately critical band spacing of the filters and bandwidths. For 4 kHz bandwidth, approximately 20 filters are used. In most implementations, a short-time Fourier analysis is done first, resulting in a DFT $X_{\hat{n}}[k]$ for analysis time $\hat{n}$. Then the DFT values are grouped together in critical bands and weighted by a triangular weighting function as depicted in Figure 5.7. Note that the bandwidths in Figure 5.7 are constant for center frequencies below 1 kHz and then increase exponentially up to half the sampling rate of 4 kHz resulting

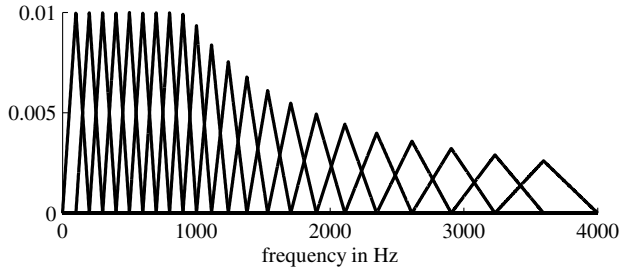

Fig. 5.7 Weighting functions for Mel-frequency filter bank.

in a total of 22 "filters." The mel-frequency spectrum at analysis time $\hat{n}$ is defined for $r = 1, 2, \ldots, R$ as

$$\mathrm{MF}_{\hat{n}}[r] = \frac{1}{A_r} \sum_{k=L_r}^{U_r} |V_r[k] X_{\hat{n}}[k]|^2, \tag{5.25a}$$

where $V_r[k]$ is the triangular weighting function for the rth filter ranging from DFT index L_r to U_r, where

$$A_r = \sum_{k=L_r}^{U_r} |V_r[k]|^2 \tag{5.25b}$$

is a normalizing factor for the rth mel-filter. This normalization is built into the weighting functions of Figure 5.7. It is needed so that a perfectly flat input Fourier spectrum will produce a flat mel-spectrum. For each frame, a discrete cosine transform of the log of the magnitude of the filter outputs is computed to form the function $\mathrm{mfcc}_{\hat{n}}[m]$, i.e.,

$$\mathrm{mfcc}_{\hat{n}}[m] = \frac{1}{R} \sum_{r=1}^{R} \log\left(\mathrm{MF}_{\hat{n}}[r]\right) \cos\left[\frac{2\pi}{R}\left(r + \frac{1}{2}\right) m\right]. \tag{5.26}$$

Typically, $\mathrm{mfcc}_{\hat{n}}[m]$ is evaluated for a number of coefficients, N_{mfcc}, that is less than the number of mel-filters, e.g., $N_{\mathrm{mfcc}} = 13$ and $R = 22$. Figure 5.8 shows the result of mfcc analysis of a frame of voiced speech in comparison with the short-time Fourier spectrum, LPC spectrum (discussed in Chapter 6), and a homomorphically smoothed spectrum. The large dots are the values of $\log\left(\mathrm{MF}_{\hat{n}}[r]\right)$ and the line interpolated

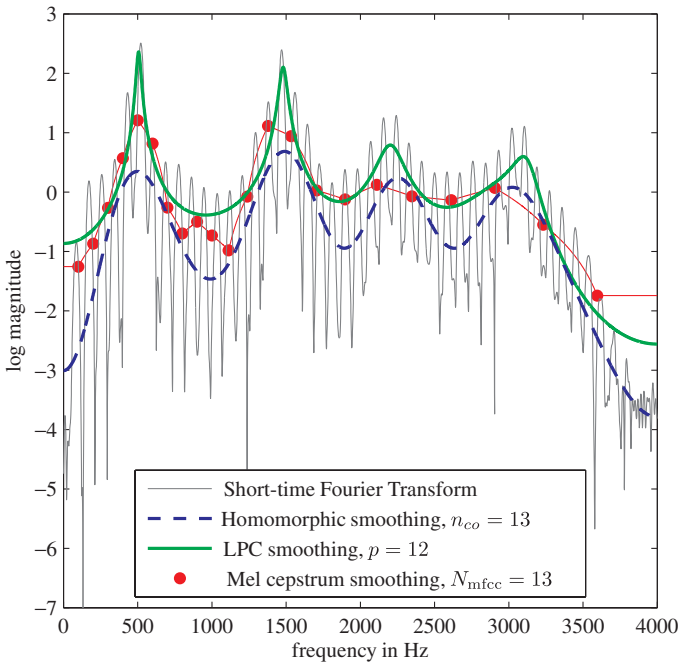

Fig. 5.8 Comparison of spectral smoothing methods.

between them is a spectrum reconstructed at the original DFT frequencies. Note that all these spectra are different, but they have in common that they have peaks at the formant resonances. At higher frequencies, the reconstructed mel-spectrum of course has more smoothing due to the structure of the filter bank.

Note that the delta cepstrum idea expressed by (5.21) can be applied to mfcc to remove the effects of linear filtering as long as the frequency response of the distorting linear filter does not vary much across each of the mel-frequency bands.

5.7 The Role of the Cepstrum

As discussed in this chapter, the cepstrum entered the realm of speech processing as a basis for pitch detection. It remains one of the most effective indicators of voice pitch that have been devised. Because the

vocal tract and excitation components are well separated in the cepstrum, it was natural to consider analysis techniques for estimation of the vocal tract system as well [86, 88, 111]. While separation techniques based on the cepstrum can be very effective, the linear predictive analysis methods to be discussed in the next chapter have proven to be more effective for a variety of reasons. Nevertheless, the cepstrum concept has demonstrated its value when applied to vocal tract system estimates obtained by linear predictive analysis.

6

Linear Predictive Analysis

Linear predictive analysis is one of the most powerful and widely used speech analysis techniques. The importance of this method lies both in its ability to provide accurate estimates of the speech parameters and in its relative speed of computation. In this chapter, we present a formulation of the ideas behind linear prediction, and discuss some of the issues that are involved in using it in practical speech applications.

6.1 Linear Prediction and the Speech Model

We come to the idea of linear prediction of speech by recalling the source/system model that was introduced in Chapter 2, where the sampled speech signal was modeled as the output of a linear, slowly time-varying system excited by either quasi-periodic impulses (during voiced speech), or random noise (during unvoiced speech). The particular form of the source/system model implied by linear predictive analysis is depicted in Figure 6.1, where the speech model is the part inside the dashed box. Over short time intervals, the linear system is

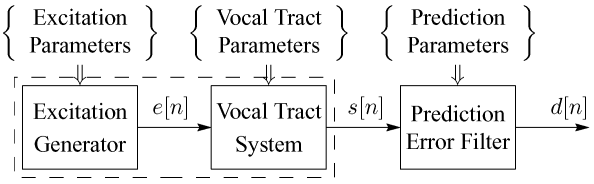

Fig. 6.1 Model for linear predictive analysis of speech signals.

described by an all-pole system function of the form:

$$H(z) = \frac{S(z)}{E(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}. \tag{6.1}$$

In linear predictive analysis, the excitation is defined implicitly by the vocal tract system model, i.e., the excitation is whatever is needed to produce $s[n]$ at the output of the system. The major advantage of this model is that the gain parameter, G, and the filter coefficients $\{a_k\}$ can be estimated in a very straightforward and computationally efficient manner by the method of linear predictive analysis.

For the system of Figure 6.1 with the vocal tract model of (6.1), the speech samples $s[n]$ are related to the excitation $e[n]$ by the difference equation

$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + Ge[n]. \tag{6.2}$$

A linear predictor with prediction coefficients, α_k, is defined as a system whose output is

$$\tilde{s}[n] = \sum_{k=1}^{p} \alpha_k s[n-k], \tag{6.3}$$

and the prediction error, defined as the amount by which $\tilde{s}[n]$ fails to exactly predict sample $s[n]$, is

$$d[n] = s[n] - \tilde{s}[n] = s[n] - \sum_{k=1}^{p} \alpha_k s[n-k]. \tag{6.4}$$

From (6.4) it follows that the prediction error sequence is the output of an FIR linear system whose system function is

$$A(z) = 1 - \sum_{k=1}^{p} \alpha_k z^{-k} = \frac{D(z)}{S(z)}. \tag{6.5}$$

It can be seen by comparing (6.2) and (6.4) that if the speech signal obeys the model of (6.2) exactly, and if $\alpha_k = a_k$, then $d[n] = Ge[n]$. Thus, the *prediction error filter*, $A(z)$, will be an *inverse filter* for the system, $H(z)$, of (6.1), i.e.,

$$H(z) = \frac{G}{A(z)}. \tag{6.6}$$

The basic problem of linear prediction analysis is to determine the set of predictor coefficients $\{\alpha_k\}$ directly from the speech signal in order to obtain a useful estimate of the time-varying vocal tract system through the use of (6.6). The basic approach is to find a set of predictor coefficients that will minimize the mean-squared prediction error over a short segment of the speech waveform. The resulting parameters are then *assumed* to be the parameters of the system function $H(z)$ in the model for production of the given segment of the speech waveform. This process is repeated periodically at a rate appropriate to track the phonetic variation of speech (i.e., order of 50–100 times per second).

That this approach will lead to useful results may not be immediately obvious, but it can be justified in several ways. First, recall that if $\alpha_k = a_k$, then $d[n] = Ge[n]$. For voiced speech this means that $d[n]$ would consist of a train of impulses, i.e., $d[n]$ would be small except at isolated samples spaced by the current pitch period, P_0. Thus, finding α_ks that minimize the mean-squared prediction error seems consistent with this observation. A second motivation for this approach follows from the fact that if a signal is generated by (6.2) with non-time-varying coefficients and excited either by a single impulse or by a stationary white noise input, then it can be shown that the predictor coefficients that result from minimizing the mean-squared prediction error (over all time) are identical to the coefficients of (6.2). A third pragmatic justification for using the minimum mean-squared prediction error as a

basis for estimating the model parameters is that this approach leads to an exceedingly useful and accurate representation of the speech signal that can be obtained by efficient solution of a set of linear equations.

The short-time average prediction error is defined as

$$E_{\hat{n}} = \left\langle d^2_{\hat{n}}[m] \right\rangle = \left\langle \left(s_{\hat{n}}[m] - \sum_{k=1}^{p} \alpha_k s_{\hat{n}}[m-k] \right)^2 \right\rangle, \tag{6.7}$$

where $s_{\hat{n}}[m]$ is a segment of speech that has been selected in a neighborhood of the analysis time $\hat{n}$, i.e.,

$$s_{\hat{n}}[m] = s[m + \hat{n}] \quad -M_1 \leq m \leq M_2. \tag{6.8}$$

That is, the time origin of the analysis segment is shifted to sample $\hat{n}$ of the entire signal. The notation $\langle\ \rangle$ denotes averaging over a finite number of samples. The details of specific definitions of the averaging operation will be discussed below.

We can find the values of α_k that minimize $E_{\hat{n}}$ in (6.7) by setting $\partial E_{\hat{n}}/\partial \alpha_i = 0$, for $i = 1, 2, \ldots, p$, thereby obtaining the equations[1]

$$\sum_{k=1}^{p} \tilde{\alpha}_k \langle s_{\hat{n}}[m-i] s_{\hat{n}}[m-k] \rangle = \langle s_{\hat{n}}[m-i] s_{\hat{n}}[m] \rangle \quad 1 \leq i \leq p, \tag{6.9}$$

where the $\tilde{\alpha}_k$ are the values of α_k that minimize $E_{\hat{n}}$ in (6.7). (Since the $\tilde{\alpha}_k$ are unique, we will drop the tilde and use the notation α_k to denote the values that minimize $E_{\hat{n}}$.) If we define

$$\varphi_{\hat{n}}[i,k] = \langle s_{\hat{n}}[m-i] s_{\hat{n}}[m-k] \rangle, \tag{6.10}$$

then (6.9) can be written more compactly as[2]

$$\sum_{k=1}^{p} \alpha_k \varphi_{\hat{n}}[i,k] = \varphi_{\hat{n}}[i,0] \quad i = 1, 2, \ldots, p. \tag{6.11}$$

[1] More accurately, the solutions of (6.7) only provide a stationary point which can be shown to be a minimum of $E_{\hat{n}}$ since $E_{\hat{n}}$ is a convex function of the α_is.

[2] The quantities $\varphi_{\hat{n}}[i,k]$ are in the form of a correlation function for the speech segment $s_{\hat{n}}[m]$. The details of the definition of the averaging operation used in (6.10) have a significant effect on the properties of the prediction coefficients that are obtained by solving (6.11).

If we know $\varphi_{\hat{n}}[i,k]$ for $1 \leq i \leq p$ and $0 \leq k \leq p$, this set of p equations in p unknowns, which can be represented by the matrix equation:

$$\mathbf{\Phi}\alpha = \psi, \tag{6.12}$$

can be solved for the vector $\alpha = \{\alpha_k\}$ of unknown predictor coefficients that minimize the average squared prediction error for the segment $s_{\hat{n}}[m]$.[3] Using (6.7) and (6.9), the minimum mean-squared prediction error can be shown to be [3, 5]

$$E_{\hat{n}} = \varphi_{\hat{n}}[0,0] - \sum_{k=1}^{p} \alpha_k \varphi_{\hat{n}}[0,k]. \tag{6.13}$$

Thus, the total minimum mean-squared error consists of a fixed component equal to the mean-squared value of the signal segment minus a term that depends on the predictor coefficients that satisfy (6.11), i.e., the optimum coefficients reduce $E_{\hat{n}}$ in (6.13) the most.

To solve for the optimum predictor coefficients, we must first compute the quantities $\varphi_{\hat{n}}[i,k]$ for $1 \leq i \leq p$ and $0 \leq k \leq p$. Once this is done we only have to solve (6.11) to obtain the α_ks. Thus, in principle, linear prediction analysis is very straightforward. However, the details of the computation of $\varphi_{\hat{n}}[i,k]$ and the subsequent solution of the equations are somewhat intricate and further discussion is required.

6.2 Computing the Prediction Coefficients

So far we have not been explicit about the meaning of the averaging notation $\langle\ \rangle$ used to define the mean-squared prediction error in (6.7). As we have stated, in a short-time analysis procedure, the averaging must be over a finite interval. We shall see below that two methods for linear predictive analysis emerge out of a consideration of the limits of summation and the definition of the waveform segment $s_{\hat{n}}[m]$.[4]

[3] Although the α_ks are functions of $\hat{n}$ (the time index at which they are estimated) this dependence will not be explicitly shown. We shall also find it advantageous to drop the subscripts $\hat{n}$ on $E_{\hat{n}}$, $s_{\hat{n}}[m]$, and $\varphi_{\hat{n}}[i,k]$ when no confusion will result.

[4] These two methods, applied to the same speech signal yield slightly different optimum predictor coefficients.

6.2.1 The Covariance Method

One approach to computing the prediction coefficients is based on the definition

$$E_{\hat{n}} = \sum_{m=-M_1}^{M_2} (d_{\hat{n}}[m])^2 = \sum_{m=-M_1}^{M_2} \left(s_{\hat{n}}[m] - \sum_{k=1}^{p} \alpha_k s_{\hat{n}}[m-k] \right)^2, \quad (6.14)$$

where $-M_1 \leq n \leq M_2$. The quantities $\varphi_{\hat{n}}[i,k]$ needed in (6.11) inherit the same definition of the averaging operator, i.e.,

$$\varphi_{\hat{n}}[i,k] = \sum_{m=-M_1}^{M_2} s_{\hat{n}}[m-i] s_{\hat{n}}[m-k] \quad \begin{cases} 1 \leq i \leq p \\ 0 \leq k \leq p. \end{cases} \quad (6.15)$$

Both (6.14) and (6.15) require values of $s_{\hat{n}}[m] = s[m+\hat{n}]$ over the range $-M_1 - p \leq m \leq M_2$. By changes of index of summation, (6.15) can be expressed in the equivalent forms:

$$\varphi_{\hat{n}}[i,k] = \sum_{m=-M_1-i}^{M_2-i} s_{\hat{n}}[m] s_{\hat{n}}[m+i-k] \quad (6.16a)$$

$$= \sum_{m=-M_1-k}^{M_2-k} s_{\hat{n}}[m] s_{\hat{n}}[m+k-i], \quad (6.16b)$$

from which it follows that $\varphi_{\hat{n}}[i,k] = \varphi_{\hat{n}}[k,i]$.

Figure 6.2 shows the sequences that are involved in computing the mean-squared prediction error as defined by (6.14). The top part of this figure shows a sampled speech signal $s[m]$, and the box denotes a segment of that waveform selected around some time index $\hat{n}$. The second plot shows that segment extracted as a finite-length sequence $s_{\hat{n}}[m] = s[m+\hat{n}]$ for $-M_1 - p \leq m \leq M_2$. Note the p "extra" samples (light shading) at the beginning that are needed to start the prediction error filter at time $-M_1$. This method does not require any assumption about the signal outside the interval $-M_1 - p \leq m \leq M_2$ since the samples $s_{\hat{n}}[m]$ for $-M_1 - p \leq m \leq M_2$ are sufficient to evaluate $\varphi[i,k]$ for all required values of i and k. The third plot shows the prediction error computed with the optimum predictor coefficients. Note that in solving for the optimum prediction coefficients using (6.11), the

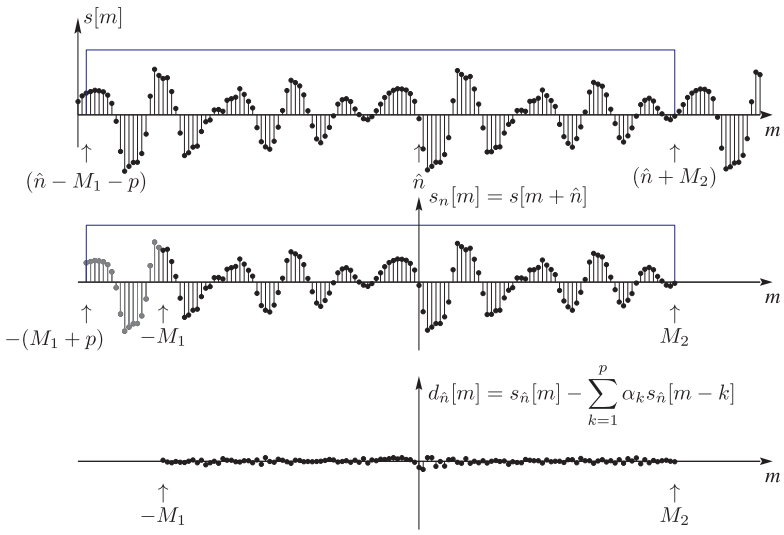

Fig. 6.2 Illustration of windowing and prediction error for the covariance method.

prediction error is implicitly computed over the range $-M_1 \leq m \leq M_2$ as required in (6.14) and (6.15). The minimum mean-squared prediction error would be simply the sum of squares of all the samples shown. In most cases of linear prediction, the prediction error sequence is not explicitly computed since solution of (6.11) does not require it.

The mathematical structure that defines the covariance method of linear predictive analysis implies a number of useful properties of the solution. It is worthwhile to summarize them as follows:

(C.1) The mean-squared prediction error satisfies $E_{\hat{n}} \geq 0$. With this method, it is theoretically possible for the average error to be exactly zero.

(C.2) The matrix $\mathbf{\Phi}$ in (6.12) is a symmetric positive-semi-definite matrix.

(C.3) The roots of the prediction error filter $A(z)$ in (6.5) are *not* guaranteed to lie within the unit circle of the z-plane. This implies that the vocal tract model filter (6.6) is *not* guaranteed to be stable.

(C.4) As a result of (C.2), the equations (6.11) can be solved efficiently using, for example, the well known Cholesky decomposition of the covariance matrix $\mathbf{\Phi}$ [3].

6.2.2 The Autocorrelation Method

Perhaps the most widely used method of linear predictive analysis is called the *autocorrelation method* because the covariance function $\varphi_{\hat{n}}[i,k]$ needed in (6.11) reduces to the STACF $\phi_{\hat{n}}[|i-k|]$ that we discussed in Chapter 4 [53, 54, 74, 78]. In the autocorrelation method, the analysis segment $s_{\hat{n}}[m]$ is defined as

$$s_{\hat{n}}[m] = \begin{cases} s[n+m]w[m] & -M_1 \leq m \leq M_2 \\ 0 & \text{otherwise,} \end{cases} \tag{6.17}$$

where the analysis window $w[m]$ is used to taper the edges of the segment to zero. Since the analysis segment is defined by the windowing of (6.17) to be zero outside the interval $-M_1 \leq m \leq M_2$, it follows that the prediction error sequence $d_{\hat{n}}[m]$ can be nonzero only in the range $-M_1 \leq m \leq M_2 + p$. Therefore, $E_{\hat{n}}$ is defined as

$$E_{\hat{n}} = \sum_{m=-M_1}^{M_2+p} (d_{\hat{n}}[m])^2 = \sum_{m=-\infty}^{\infty} (d_{\hat{n}}[m])^2. \tag{6.18}$$

The windowing of (6.17) allows us to use the infinite limits to signify that the sum is over all nonzero values of $d_{\hat{n}}[m]$. Applying this notion to (6.16a) and (6.16b) leads to the conclusion that

$$\varphi_{\hat{n}}[i,k] = \sum_{m=-\infty}^{\infty} s_{\hat{n}}[m]s_{\hat{n}}[m+|i-k|] = \phi_{\hat{n}}[|i-k|]. \tag{6.19}$$

Thus, $\varphi[i,k]$ is a function only of $|i-k|$. Therefore, we can replace $\varphi_{\hat{n}}[i,k]$ by $\phi_{\hat{n}}[|i-k|]$, which is the STACF defined in Chapter 4 as

$$\phi_{\hat{n}}[k] = \sum_{m=-\infty}^{\infty} s_{\hat{n}}[m]s_{\hat{n}}[m+k] = \phi_{\hat{n}}[-k]. \tag{6.20}$$

The resulting set of equations for the optimum predictor coefficients is therefore

$$\sum_{k=1}^{p} \alpha_k \phi_{\hat{n}}[|i-k|] = \phi_{\hat{n}}[i] \quad i = 1, 2, \ldots, p. \tag{6.21}$$

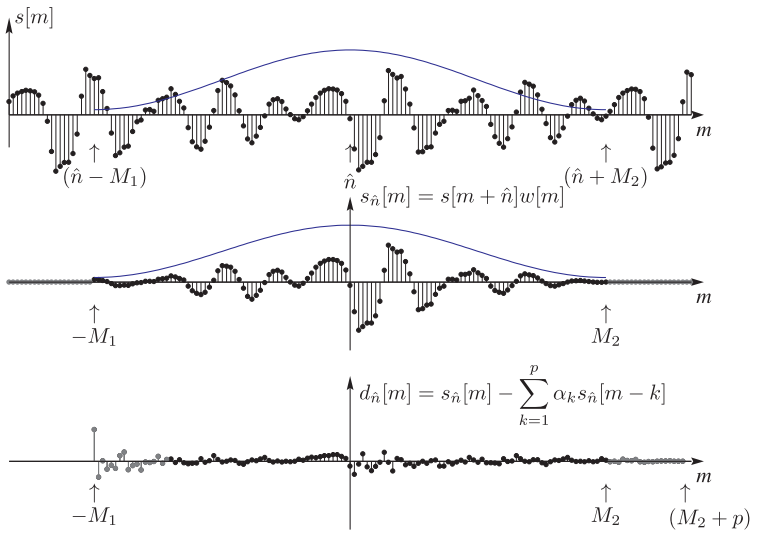

Fig. 6.3 Illustration of windowing and prediction error for the autocorrelation method.

Figure 6.3 shows the sequences that are involved in computing the optimum prediction coefficients using the autocorrelation method. The upper plot shows the same sampled speech signal $s[m]$ as in Figure 6.2 with a Hamming window centered at time index $\hat{n}$. The middle plot shows the result of multiplying the signal $s[\hat{n}+m]$ by the window $w[m]$ and redefining the time origin to obtain $s_{\hat{n}}[m]$. Note that the zero-valued samples outside the window are shown with light shading. The third plot shows the prediction error computed using the optimum coefficients. Note that for this segment, the prediction error (which is implicit in the solution of (6.21)) is nonzero over the range $-M_1 \leq m \leq M_2 + p$. Also note the lightly shaded p samples at the beginning. These samples can be large due to the fact that the predictor must predict these samples from the zero-valued samples that precede the windowed segment $s_{\hat{n}}[m]$. It is easy to see that at least one of these first p samples of the prediction error must be nonzero. Similarly, the last p samples of the prediction error can be large due to the fact that the predictor must predict zero-valued samples from windowed speech samples. It can also be seen that at least one of these last p samples of the prediction error must be nonzero. For this reason, it follows that $E_{\hat{n}}$, being the sum of

squares of the prediction error samples, must always be strictly greater than zero.[5]

As in the case of the covariance method, the mathematical structure of the autocorrelation method implies a number of properties of the solution, including the following:

(A.1) The mean-squared prediction error satisfies $E_{\hat{n}} > 0$. With this method, it is theoretically impossible for the error to be exactly zero because there will always be at least one sample at the beginning and one at the end of the prediction error sequence that will be nonzero.

(A.2) The matrix $\mathbf{\Phi}$ in (6.12) is a symmetric positive-definite Toeplitz matrix [46].

(A.3) The roots of the prediction error filter $A(z)$ in (6.5) are guaranteed to lie within the unit circle of the z-plane so that the vocal tract model filter of (6.6) is guaranteed to be stable.

(A.4) As a result of (A.2), the equations (6.11) can be solved efficiently using the Levinson–Durbin algorithm, which, because of its many implications, we discuss in more detail in Section 6.3.

6.3 The Levinson–Durbin Recursion

As stated in (A.2) above, the matrix $\mathbf{\Phi}$ in (6.12) is a symmetric positive-definite Toeplitz matrix, which means that all the elements on a given diagonal in the matrix are equal. Equation (6.22) shows the detailed structure of the matrix equation $\mathbf{\Phi}\alpha = \psi$ for the autocorrelation method.

$$\begin{bmatrix} \phi[0] & \phi[1] & \cdots & \phi[p-1] \\ \phi[1] & \phi[0] & \cdots & \phi[p-2] \\ \cdots & \cdots & \cdots & \cdots \\ \phi[p-1] & \phi[p-2] & \cdots & \phi[0] \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} \phi[1] \\ \phi[2] \\ \cdots \\ \phi[p] \end{bmatrix} \tag{6.22}$$

[5] For this reason, a tapering window is generally used in the autocorrelation method.

Note that the vector ψ is composed of almost the same autocorrelation values as comprise $\mathbf{\Phi}$. Because of the special structure of (6.22) it is possible to derive a recursive algorithm for inverting the matrix $\mathbf{\Phi}$. That algorithm, known as the Levinson–Durbin algorithm, is specified by the following steps:

Levinson–Durbin Algorithm

$$E^0 = \phi[0] \tag{D.1}$$

for $i = 1, 2, \ldots, p$

$$k_i = \left(\phi[i] - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} \phi[i-j] \right) \Big/ E^{(i-1)} \tag{D.2}$$

$$\alpha_i^{(i)} = k_i \tag{D.3}$$

if $i > 1$ then for $j = 1, 2, \ldots, i-1$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)} \tag{D.4}$$

end

$$E^{(i)} = (1 - k_i^2) E^{(i-1)} \tag{D.5}$$

end

$$\alpha_j = \alpha_j^{(p)} \quad j = 1, 2, \ldots, p \tag{D.6}$$

An important feature of the Levinson–Durbin algorithm is that it determines by recursion the optimum ith-order predictor from the optimum $(i-1)$th-order predictor, and as part of the process, all predictors from order 0 (no prediction) to order p are computed along with the corresponding mean-squared prediction errors $E^{(i)}$. Specifically, the equations labeled (D.3) and (D.4) can be used to show that the prediction error system function satisfies

$$A^{(i)}(z) = A^{(i-1)}(z) - k_i z^{-i} A^{(i-1)}(z^{-1}). \tag{6.23}$$

Defining the ith-order forward prediction error $e^{(i)}[n]$ as the output of the prediction error filter with system function $A^{(i)}(z)$ and $b^{(i)}[n]$ as the output of the ith-order backward prediction error filter $B^{(i)}(z) = z^{-i} A^{(i)}(z^{-1})$, (6.23) leads (after some manipulations) to an interpretation of the Levinson–Durbin algorithm in terms of a lattice filter structure as in Figure 6.4(a).

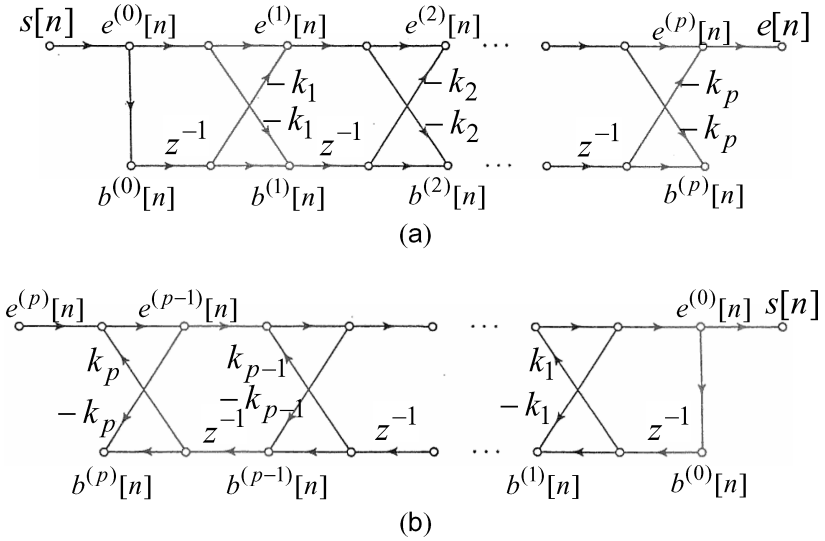

Fig. 6.4 Lattice structures derived from the Levinson–Durbin recursion. (a) Prediction error filter $A(z)$. (b) Vocal tract filter $H(z) = 1/A(z)$.

Also, by solving two equations in two unknowns recursively, it is possible to start at the output of Figure 6.4(a) and work to the left eventually computing $s[n]$ in terms of $e[n]$. The lattice structure corresponding to this is shown in Figure 6.4(b). Lattice structures like this can be derived from acoustic principles applied to a physical model composed of concatenated lossless tubes [101]. If the input and output signals in such physical models are sampled at just the right sampling rate, the sampled signals are related by a transfer function identical to (6.6). In this case, the coefficients k_i behave as reflection coefficients at the tube boundaries [3, 78, 101].

Note that the parameters k_i for $i = 1, 2, \ldots, p$ play a key role in the Levinson–Durbin recursion and also in the lattice filter interpretation. Itakura and Saito [53, 54], showed that the parameters k_i in the Levinson–Durbin recursion and the lattice filter interpretation obtained from it also could be derived by looking at linear predictive analysis from a statistical perspective. They called the k_i parameters, PARCOR (for partial correlation) coefficients [54], because they can be computed directly as a ratio of cross-correlation values between the forward and

backward prediction errors at the output of the $(i-1)$th stage of prediction in Figure 6.4(a), i.e.,

$$k_i = \frac{\sum_{m=-\infty}^{\infty} e^{(i-1)}[m]b^{(i-1)}[m-1]}{\left(\sum_{m=-\infty}^{\infty}\left(e^{(i-1)}[m]\right)^2 \sum_{m=-\infty}^{\infty}\left(b^{(i-1)}[m-1]\right)^2\right)^{1/2}}. \tag{6.24}$$

In the PARCOR interpretation, each stage of Figure 6.4(a) removes *part* of the correlation in the input signal. The PARCOR coefficients computed using (6.24) are identical to the k_i obtained as a result of the Levinson–Durbin algorithm. Indeed, Equation (D.2) in the Levinson–Durbin algorithm can be replaced by (6.24), and the result is an algorithm for transforming the PARCOR representation into the linear predictor coefficient representation.

The Levinson–Durbin formulation provides one more piece of useful information about the PARCOR coefficients. Specifically, from Equation (D.5) of the algorithm, it follows that, since $E^{(i)} = (1-k_i^2)E^{(i-1)}$ is strictly greater than zero for predictors of all orders, it must be true that $-1 < k_i < 1$ for all i. It can be shown that this condition on the PARCORs also guarantees that all the zeros of a prediction error filter $A^{(i)}(z)$ of any order must be strictly *inside* the unit circle of the z-plane [74, 78].

6.4 LPC Spectrum

The frequency-domain interpretation of linear predictive analysis provides an informative link to our earlier discussions of the STFT and cepstrum analysis. The autocorrelation method is based on the short-time autocorrelation function, $\phi_{\hat{n}}[m]$, which is the inverse discrete Fourier transform of the magnitude-squared of the STFT, $|S_{\hat{n}}(e^{j\hat{\omega}})|^2$, of the windowed speech signal $s_{\hat{n}}[m] = s[n+m]w[m]$. The values $\phi_{\hat{n}}[m]$ for $m = 0, 1, \ldots, p$ are used to compute the prediction coefficients and gain, which in turn define the vocal tract system function $H(z)$ in (6.6). Therefore, the magnitude-squared of the frequency response of this system, obtained by evaluating $H(z)$ on the unit circle at angles $2\pi f/f_s$,

is of the form:

$$|H(e^{j2\pi f/f_s})|^2 = \left| \frac{G}{1 - \sum_{k=1}^{p} \alpha_k e^{-j2\pi f/f_s}} \right|^2, \tag{6.25}$$

and can be thought of as an alternative short-time spectral representation. Figure 6.5 shows a comparison between short-time Fourier analysis and linear predictive spectrum analysis for segments of voiced and unvoiced speech. Figures 6.5(a) and 6.5(c) show the STACF, with the first 23 values plotted with a heavy line. These values are used to compute the predictor coefficients and gain for an LPC model with $p = 22$. The frequency responses of the corresponding vocal tract system models are computed using (6.25) where the sampling frequency is $f_s = 16\,\text{kHz}$. These frequency responses are superimposed on the corresponding STFTs (shown in gray). As we have observed before, the rapid variations with frequency in the STFT are due primarily to the excitation, while the overall shape is assumed to be determined by the effects of glottal pulse, vocal tract transfer function, and radiation. In

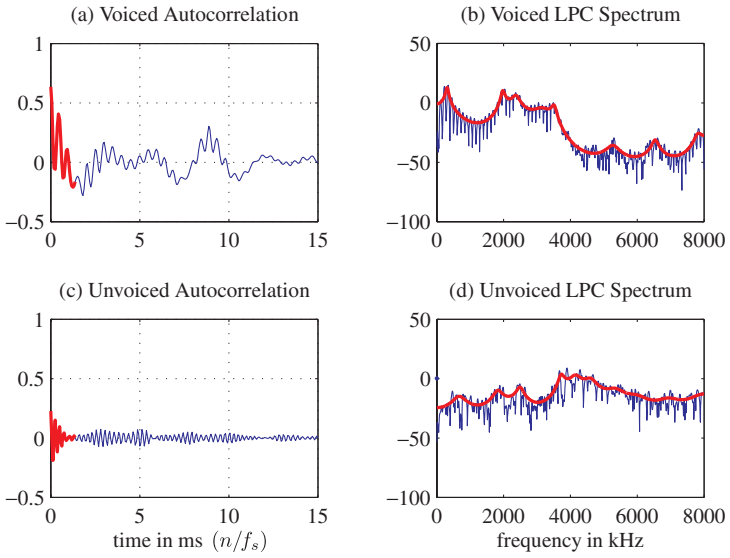

Fig. 6.5 Comparison of short-time Fourier analysis with linear predictive analysis.

cepstrum analysis, the excitation effects are removed by lowpass liftering the cepstrum. In linear predictive spectrum analysis, the excitation effects are removed by focusing on the low-time autocorrelation coefficients. The amount of smoothing of the spectrum is controlled by the choice of p. Figure 6.5 shows that a linear prediction model with $p = 22$ matches the general shape of the short-time spectrum, but does not represent all its local peaks and valleys, and this is exactly what is desired.

The question naturally arises as to how p should be chosen. Figure 6.6 offers a suggestion. In Figure 6.6(a) are shown the STFT (in gray) and the frequency responses of a 12th-order model (heavy dark line) and a 40th-order model (thin dark line). Evidently, the linear predictive spectra tend to favor the peaks of the short-time Fourier transform. That this is true in general can be argued using the Parseval theorem of Fourier analysis [74]. This is in contrast to homomorphic

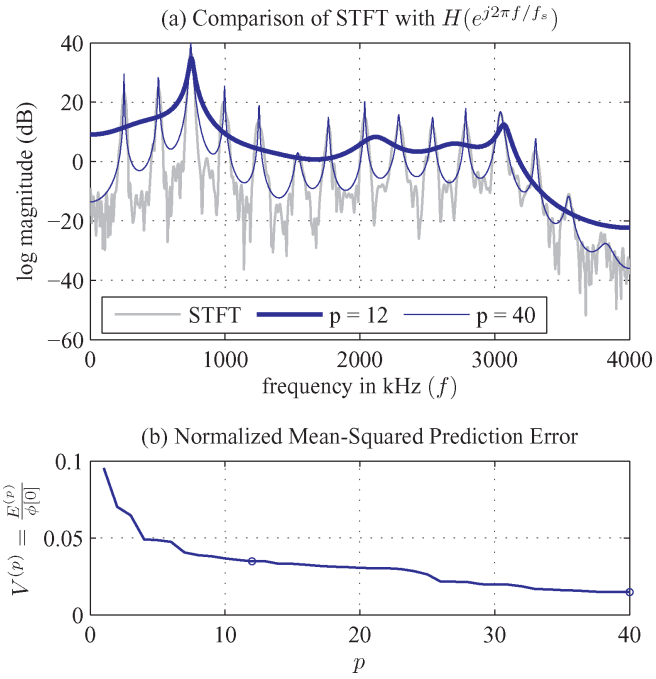

Fig. 6.6 Effect of predictor order on: (a) estimating the spectral envelope of a 4-kHz bandwidth signal; and (b) the normalized mean-squared prediction error for the same signal.

smoothing of the STFT, which tends toward an average of the peaks and valleys of the STFT. As an example, see Figure 5.8, which shows a comparison between the short-time Fourier spectrum, the LPC spectrum with $p = 12$, a homomorphically smoothed spectrum using 13 cepstrum values, and a mel-frequency spectrum.

Figure 6.6(b) shows the normalized mean-squared prediction error $V^{(p)} = E^{(p)}/\phi[0]$ as a function of p for the segment of speech used to produce Figure 6.6(a). Note the sharp decrease from $p = 0$ (no prediction implies $V^{(0)} = 1$) to $p = 1$ and the less abrupt decrease thereafter. Furthermore, notice that the mean-squared error curve flattens out above about $p = 12$ and then decreases modestly thereafter. The choice $p = 12$ gives a good match to the general shape of the STFT, highlighting the formant structure imposed by the vocal tract filter while ignoring the periodic pitch structure. Observe that $p = 40$ gives a much different result. In this case, the vocal tract model is highly influenced by the pitch harmonics in the short-time spectrum. It can be shown that if p is increased to the length of the windowed speech segment, that $|H(e^{j2\pi f/f_s})|^2 \rightarrow |S_{\hat{n}}(e^{j2\pi f/f_s})|^2$, but as pointed out in (A.1) above, $E^{(2M)}$ does not go to zero [75].

In a particular application, the prediction order is generally fixed at a value that captures the general spectral shape due to the glottal pulse, vocal tract resonances, and radiation. From the acoustic theory of speech production, it follows that the glottal pulse spectrum is lowpass, the radiation filtering is highpass, and the vocal tract imposes a resonance structure that, for adult speakers, is comprised of about one resonance per kilohertz of frequency [101]. For the sampled speech signal, the combination of the lowpass glottal pulse spectrum and the highpass filtering of radiation are usually adequately represented by one or two additional complex pole pairs. When coupled with an estimate of one resonance per kilohertz, this leads to a rule of thumb of $p = 4 + f_s/1000$. For example, for a sampling rate of $f_s = 8000\,\text{Hz}$, it is common to use a predictor order of 10–12. Note that in Figure 6.5 where the sampling rate was $f_s = 16000\,\text{Hz}$, a predictor order of $p = 22$ gave a good representation of the overall shape and resonance structure of the speech segment over the band from 0 to 8000 Hz.

The example of Figure 6.6 illustrates an important point about linear predictive analysis. In that example, the speaker was a high-pitched female, and the wide spacing between harmonics causes the peaks of the linear predictive vocal tract model to be biased toward those harmonics. This effect is a serious limitation for high-pitched voices, but is less problematic for male voices where the spacing between harmonics is generally much smaller.

6.5 Equivalent Representations

The basic parameters obtained by linear predictive analysis are the gain G and the prediction coefficients $\{\alpha_k\}$. From these, a variety of different equivalent representations can be obtained. These different representations are important, particularly when they are used in speech coding, because we often wish to quantize the model parameters for efficiency in storage or transmission of coded speech.

In this section, we give only a brief summary of the most important equivalent representations.

6.5.1 Roots of Prediction Error System Function

Equation (6.5) shows that the system function of the prediction error filter is a polynomial in z^{-1} and therefore it can be represented in terms of its zeros as

$$A(z) = 1 - \sum_{k=1}^{p} \alpha_k z^{-k} = \prod_{k=1}^{p} (1 - z_k z^{-1}). \tag{6.26}$$

According to (6.6), the zeros of $A(z)$ are the poles of $H(z)$. Therefore, if the model order is chosen judiciously as discussed in the previous section, then it can be expected that roughly $f_s/1000$ of the roots will be close in frequency (angle in the z-plane) to the formant frequencies. Figure 6.7 shows an example of the roots (marked with $\times$) of a 12th-order predictor. Note that eight (four complex conjugate pairs) of the roots are close to the unit circle. These are the poles of $H(z)$ that model the formant resonances. The remaining four roots lie well within the unit circle, which means that they only provide for the overall spectral shaping resulting from the glottal and radiation spectral shaping.

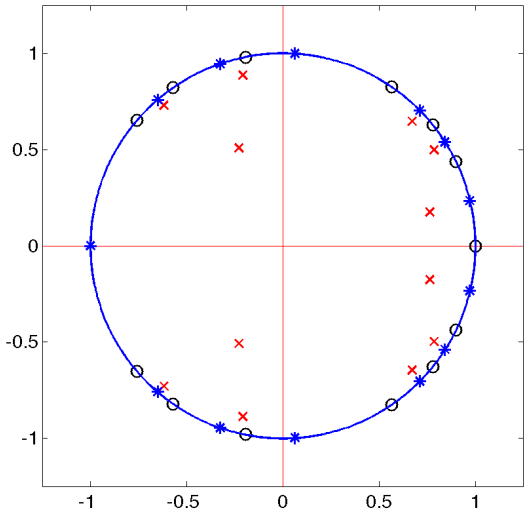

Fig. 6.7 Poles of $H(z)$ (zeros of $A(z)$) marked with $\times$ and LSP roots marked with $*$ and o.

The prediction coefficients are perhaps the most susceptible to quantization errors of all the equivalent representations. It is well-known that the roots of a polynomial are highly sensitive to errors in its coefficients — all the roots being a function of all the coefficients [89]. Since the pole locations are crucial to accurate representation of the spectrum, it is important to maintain high accuracy in the location of the zeros of $A(z)$. One possibility, not often invoked, would be to factor the polynomial as in (6.26) and then quantize each root (magnitude and angle) individually.

6.5.2 LSP Coefficients

A much more desirable alternative to quantization of the roots of $A(z)$ was introduced by Itakura [52], who defined the *line spectrum pair* (LSP) polynomials[6]

$$P(z) = A(z) + z^{-(p+1)}A(z^{-1}) \tag{6.27a}$$

$$Q(z) = A(z) - z^{-(p+1)}A(z^{-1}). \tag{6.27b}$$

[6] Note the similarity to (6.23), from which it follows that $P(z)$ and $Q(z)$ are system functions of lattice filters obtained by extending Figure 6.4(a) with an additional section with $k_{p+1} = \mp 1$, respectively.

This transformation of the linear prediction parameters is invertible: to recover $A(z)$ from $P(z)$ and $Q(z)$ simply add the two equations to obtain $A(z) = (P(z) + Q(z))/2$.

An illustration of the LSP representation is given in Figure 6.7, which shows the roots of $A(z)$, $P(z)$, and $Q(z)$ for a 12th-order predictor. This new representation has some very interesting and useful properties that are confirmed by Figure 6.7 and summarized below [52, 119].

(LSP.1) All the roots of $P(z)$ and $Q(z)$ are on the unit circle, i.e., they can be represented as

$$P(z) = \prod_{k=1}^{p+1} (1 - e^{j\Omega_k} z^{-1}) \tag{6.28a}$$

$$Q(z) = \prod_{k=1}^{p+1} (1 - e^{j\Theta_k} z^{-1}). \tag{6.28b}$$

The normalized discrete-time frequencies (angles in the z-plane), Ω_k and Θ_k, are called the *line spectrum frequencies* or LSFs. Knowing the p LSFs that lie in the range $0 < \omega < \pi$ is sufficient to completely define the LSP polynomials and therefore $A(z)$.

(LSP.2) If p is an even integer, then $P(-1) = 0$ and $Q(1) = 0$.

(LSP.3) The PARCOR coefficients corresponding to $A(z)$ satisfy $|k_i| < 1$ if and only if the roots of $P(z)$ and $Q(z)$ alternate on the unit circle, i.e., the LSFs are interlaced over the range of angles 0 to π.

(LSP.4) The LSFs are close together when the roots of $A(z)$ are close to the unit circle.

These properties make it possible to represent the linear predictor by quantized differences between the successive LSFs [119]. If property (LSP.3) is maintained through the quantization, the reconstructed polynomial $A(z)$ will still have its zeros inside the unit circle.

6.5.3 Cepstrum of Vocal Tract Impulse Response

One of the most useful alternative representations of the linear predictor is the cepstrum of the impulse response, $h[n]$, of the vocal tract filter. The impulse response can be computed recursively through the difference equation:

$$h[n] = \sum_{k=1}^{p} \alpha_k h[n-k] + G\delta[n], \tag{6.29}$$

or a closed form expression for the impulse response can be obtained by making a partial fraction expansion of the system function $H(z)$.

As discussed in Section 5.3.3, since $H(z)$ is a minimum-phase system (all its poles inside the unit circle), it follows that the complex cepstrum of $h[n]$ can be computed using (5.14), which would require that the poles of $H(z)$ be found by polynomial rooting. Then, any desired number of cepstrum values can be computed. However, because of the minimum phase condition, a more direct recursive computation is possible [101]. Equation (5.16) gives the recursion for computing $\hat{h}[n]$ from the predictor coefficients and gain, and (5.17) gives the inverse recursion for computing the predictor coefficients from the complex cepstrum of the vocal tract impulse response. These relationships are particularly useful in speech recognition applications where a small number of cepstrum values is used as a feature vector.

6.5.4 PARCOR Coefficients

We have seen that the PARCOR coefficients are bounded by ± 1. This makes them an attractive parameter for quantization.

We have mentioned that if we have a set of PARCOR coefficients, we can simply use them at step (D.2) of the Levinson–Durbin algorithm thereby obtaining an algorithm for converting PARCORs to predictor coefficients. By working backward through the Levinson–Durbin algorithm, we can compute the PARCOR coefficients from a given set of predictor coefficients. The resulting algorithm is given below.

Predictor-to-PARCOR Algorithm

$\alpha_j^{(p)} = \alpha_j \quad j = 1, 2, \ldots, p$

$k_p = \alpha_p^{(p)}$ (P.1)

for $i = p, p-1, \ldots, 2$

  for $j = 1, 2, \ldots, i-1$

$$\alpha_j^{(i-1)} = \frac{\alpha_j^{(i)} + k_i \alpha_{i-j}^{(i)}}{1 - k_i^2} \quad \text{(P.2)}$$

  end

  $k_{i-1} = \alpha_{i-1}^{(i-1)}$ (P.3)

end

6.5.5 Log Area Coefficients

As mentioned above, the lattice filter representation of the vocal tract system is strongly related to concatenated acoustic tube models of the physics of sound propagation in the vocal tract. Such models are characterized by a set of tube cross-sectional areas denoted A_i. These "area function" parameters are useful alternative representations of the vocal tract model obtained from linear predictive analysis. Specifically, the log area ratio parameters are defined as

$$g_i = \log\left[\frac{A_{i+1}}{A_i}\right] = \log\left[\frac{1 - k_i}{1 + k_i}\right], \tag{6.30}$$

where A_i and A_{i+1} are the areas of two successive tubes and the PARCOR coefficient $-k_i$ is the reflection coefficient for sound waves impinging on the junction between the two tubes [3, 78, 101]. The inverse transformation (from g_i to k_i) is

$$k_i = \frac{1 - e^{g_i}}{1 + e^{g_i}}. \tag{6.31}$$

The PARCOR coefficients can be converted to predictor coefficients if required using the technique discussed in Section 6.5.4.

Viswanathan and Makhoul [131] showed that the frequency response of a vocal tract filter represented by quantized log area ratio coefficients is relatively insensitive to quantization errors.

6.6 The Role of Linear Prediction

In this chapter, we have discussed many of the fundamentals of linear predictive analysis of speech. The value of linear predictive analysis stems from its ability to represent in a compact form the part of the speech model associated with the vocal tract, which in turn is closely related to the phonemic representation of the speech signal. Over 40 years of research on linear predictive methods have yielded a wealth of knowledge that has been widely and effectively applied in almost every area of speech processing, but especially in speech coding, speech synthesis, and speech recognition. The present chapter and Chapters 3–5 provide the basis for the remaining chapters of this text, which focus on these particular application areas.

7

Digital Speech Coding

This chapter, the first of three on digital speech processing applications, focuses on specific techniques that are used in digital speech coding. We begin by describing the basic operation of sampling a speech signal and directly quantizing and encoding of the samples. The remainder of the chapter discusses a wide variety of techniques that represent the speech signal in terms of parametric models of speech production and perception.

7.1 Sampling and Quantization of Speech (PCM)

In any application of digital signal processing (DSP), the first step is sampling and quantization of the resulting samples into digital form. These operations, which comprise the process of A-to-D conversion, are depicted for convenience in analysis and discussion in the block diagram of Figure 7.1.[1]

[1] Current A-to-D and D-to-A converters use oversampling techniques to implement the functions depicted in Figure 7.1.

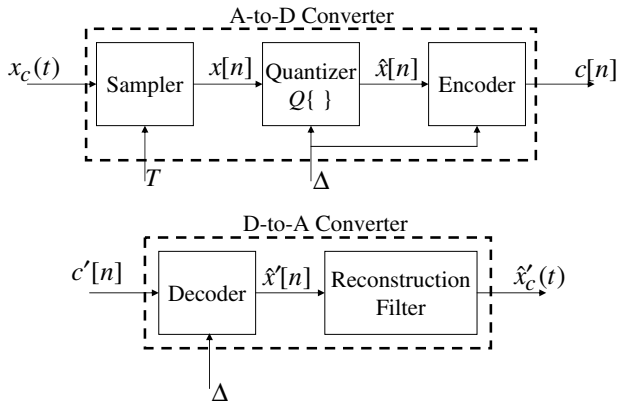

Fig. 7.1 The operations of sampling and quantization.

The sampler produces a sequence of numbers $x[n] = x_c(nT)$, where T is the sampling period and $f_s = 1/T$ is the sampling frequency. Theoretically, the samples $x[n]$ are real numbers that are useful for theoretical analysis, but never available for computations. The well known Shannon sampling theorem states that a bandlimited signal can be reconstructed exactly from samples taken at twice the highest frequency in the input signal spectrum (typically between 7 and 20 kHz for speech and audio signals) [89]. Often, we use a lowpass filter to remove spectral information above some frequency of interest (e.g., 4 kHz) and then use a sampling rate such as 8000 samples/s to avoid aliasing distortion [89].

Figure 7.2 depicts a typical quantizer definition. The quantizer simply takes the real number inputs $x[n]$ and assigns an output $\hat{x}[n]$ according to the nonlinear discrete-output mapping $Q\{\ \}$.[2] In the case of the example of Figure 7.2, the output samples are mapped to one of eight possible values, with samples within the peak-to-peak range being rounded and samples outside the range being "clipped" to either the maximum positive or negative level. For samples within range, the quantization error, defined as

$$e[n] = \hat{x}[n] - x[n] \tag{7.1}$$

[2] Note that in this chapter, the notation $\hat{x}[n]$ denotes the quantized version of $x[n]$, not the complex cepstrum of $x[n]$ as in Chapter 5.

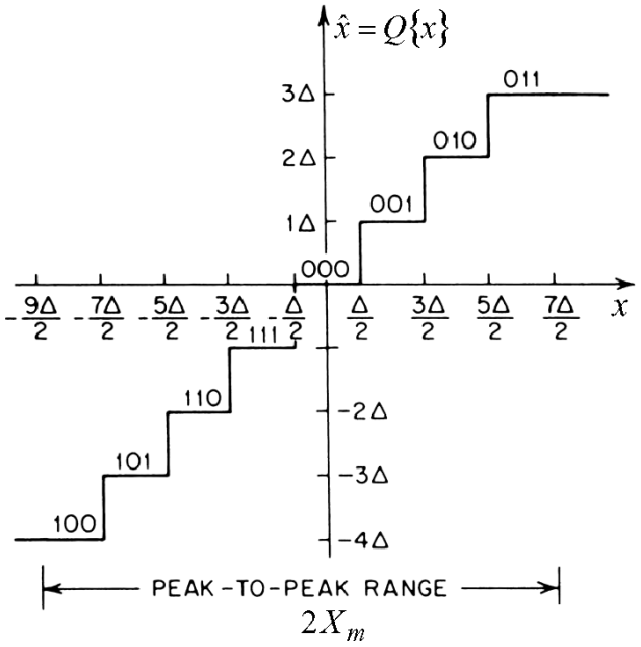

Fig. 7.2 8-level mid-tread quantizer.

satisfies the condition

$$-\Delta/2 < e[n] \leq \Delta/2, \tag{7.2}$$

where Δ is the quantizer step size. A B-bit quantizer such as the one shown in Figure 7.2 has 2^B levels (1 bit usually signals the sign). Therefore, if the peak-to-peak range is $2X_m$, the step size will be $\Delta = 2X_m/2^B$. Because the quantization levels are uniformly spaced by Δ, such quantizers are called *uniform quantizers*. Traditionally, representation of a speech signal by binary-coded quantized samples is called *pulse-code modulation* (or just PCM) because binary numbers can be represented for transmission as on/off pulse amplitude modulation.

The block marked "encoder" in Figure 7.1 represents the assigning of a binary code word to each quantization level. These code words represent the quantized signal amplitudes, and generally, as in Figure 7.2, these code words are chosen to correspond to some convenient binary number system such that arithmetic can be done on the code words as

if they were proportional to the signal samples.[3] In cases where accurate amplitude calibration is desired, the step size could be applied as depicted in the block labeled "Decoder" at the bottom of Figure 7.1.[4] Generally, we do not need to be concerned with the fine distinction between the quantized samples and the coded samples when we have a fixed step size, but this is not the case if the step size is adapted from sample-to-sample. Note that the combination of the decoder and lowpass reconstruction filter represents the operation of D-to-A conversion [89].

The operation of quantization confronts us with a dilemma. Most signals, speech especially, have a wide dynamic range, i.e., their amplitudes vary greatly between voiced and unvoiced sounds and between speakers. This means that we need a large peak-to-peak range so as to avoid clipping of the loudest sounds. However, for a given number of levels (bits), the step size $\Delta = 2X_m/2^B$ is proportional to the peak-to-peak range. Therefore, as the peak-to-peak range increases, (7.2) states that the size of the maximum quantization error grows. Furthermore, for a uniform quantizer, the maximum size of the quantization error is the same whether the signal sample is large or small. For a given peak-to-peak range for a quantizer such as the one in Figure 7.2 we can decrease the quantization error only by adding more levels (bits). This is the fundamental problem of quantization.

The data rate (measured in bits/second) of sampled and quantized speech signals is $I = B \cdot f_s$. The standard values for sampling and quantizing sound signals (speech, singing, instrumental music) are $B = 16$ and $f_s = 44.1$ or 48 kHz. This leads to a bitrate of $I = 16 \cdot 44100 = 705,600$ bits/s.[5] This value is more than adequate and much more than desired for most speech communication applications. The bit rate can be lowered by using fewer bits/sample and/or using a lower sampling rate; however, both these simple approaches degrade the perceived quality of the speech signal. This chapter deals with a wide range of

[3] In a real A-to-D converter, the sampler, quantizer, and coder are all integrated into one system, and only the binary code words $c[n]$ are available.

[4] The $'$ denotes the possibility of errors in the codewords. Symbol errors would cause additional error in the reconstructed samples.

[5] Even higher rates (24 bits and 96 kHz sampling rate) are used for high-quality audio.

techniques for significantly reducing the bit rate while maintaining an adequate level of speech quality.

7.1.1 Uniform Quantization Noise Analysis

A more quantitative description of the effect of quantization can be obtained using random signal analysis applied to the quantization error. If the number of bits in the quantizer is reasonably high and no clipping occurs, the quantization error sequence, although it is completely determined by the signal amplitudes, nevertheless behaves as if it is a random signal with the following properties [12]:

(Q.1) The noise samples appear to be[6] uncorrelated with the signal samples.

(Q.2) Under certain assumptions, notably smooth input probability density functions and high rate quantizers, the noise samples appear to be uncorrelated from sample-to-sample, i.e., $e[n]$ acts like a white noise sequence.

(Q.3) The amplitudes of noise samples are uniformly distributed across the range $-\Delta/2 < e[n] \leq \Delta/2$, resulting in average power $\sigma_e^2 = \Delta^2/12$.

These simplifying assumptions allow a linear analysis that yields accurate results if the signal is not too coarsely quantized. Under these conditions, it can be shown that if the output levels of the quantizer are optimized, then the quantizer error will be uncorrelated with the quantizer output (however not the quantizer input, as commonly stated). These results can be easily shown to hold in the simple case of a uniformly distributed memoryless input and Bennett has shown how the result can be extended to inputs with smooth densities if the bit rate is assumed high [12].

With these assumptions, it is possible to derive the following formula for the signal-to-quantizing-noise ratio (in dB) of a B-bit

[6] By "appear to be" we mean that measured correlations are small. Bennett has shown that the correlations are only small because the error is small and, under suitable conditions, the correlations are equal to the negative of the error variance [12]. The condition "uncorrelated" often implies independence, but in this case the error is a deterministic function of the input and hence it cannot be independent of the input.

uniform quantizer:

$$\mathrm{SNR}_Q = 10\log\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 6.02B + 4.78 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right), \qquad (7.3)$$

where σ_x and σ_e are the rms values of the input signal and quantization noise samples, respectively. The formula of Equation (7.3) is an increasingly good approximation as the bit rate (or the number of quantization levels) gets large. It can be way off, however, if the bit rate is not large [12]. Figure 7.3 shows a comparison of (7.3) with signal-to-quantization-noise ratios measured for speech signals. The measurements were done by quantizing 16-bit samples to 8, 9, and 10 bits. The faint dashed lines are from (7.3) and the dark dashed lines are measured values for uniform quantization. There is good agreement between these graphs indicating that (7.3) is a reasonable estimate of SNR.

Note that X_m is a fixed parameter of the quantizer, while σ_x depends on the input signal level. As signal level increases, the ratio X_m/σ_x decreases moving to the left in Figure 7.3. When σ_x gets close to X_m, many samples are clipped, and the assumptions underlying (7.3) no longer hold. This accounts for the precipitous fall in SNR for $1 < X_m/\sigma_x < 8$.

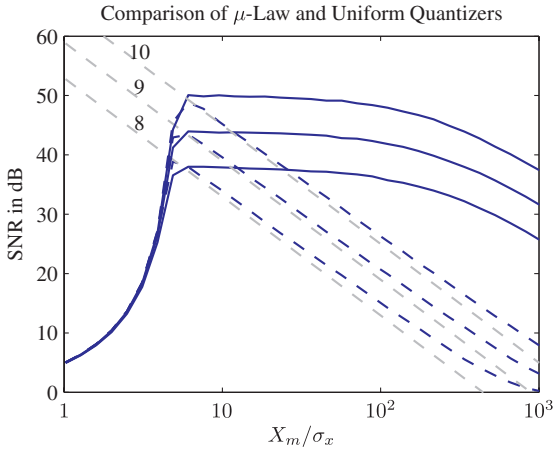

Fig. 7.3 Comparison of μ-law and linear quantization for $B = 8, 9, 10$: Equation (7.3) — light dashed lines. Measured uniform quantization SNR — dark dashed lines. μ-law ($\mu = 100$) compression — solid lines.

Also, it should be noted that (7.3) and Figure 7.3 show that with all other parameters being fixed, increasing B by 1 bit (doubling the number of quantization levels) increases the SNR by 6 dB. On the other hand, it is also important to note that halving σ_x decreases the SNR by 6 dB. In other words, cutting the signal level in half is like throwing away one bit (half of the levels) of the quantizer. Thus, it is exceedingly important to keep input signal levels as high as possible without clipping.

7.1.2 μ-Law Quantization

Also note that the SNR curves in Figure 7.3 decrease linearly with increasing values of $\log_{10}(X_m/\sigma_x)$. This is because the size of the quantization noise remains the same as the signal level decreases. If the quantization error were proportional to the signal amplitude, the SNR would be constant regardless of the size of the signal. This could be achieved if $\log(x[n])$ were quantized instead of $x[n]$, but this is not possible because the log function is ill-behaved for small values of $x[n]$. As a compromise, μ-law compression (of the dynamic range) of the signal, defined as

$$y[n] = X_m \frac{\log\left(1 + \mu \frac{|x[n]|}{X_m}\right)}{\log(1+\mu)} \cdot \text{sign}(x[n]) \tag{7.4}$$

can be used prior to uniform quantization [118]. Quantization of μ-law compressed signals is often termed *log-PCM*. If μ is large (> 100) this nonlinear transformation has the effect of distributing the effective quantization levels uniformly for small samples and logarithmically over the remaining range of the input samples. The flat curves in Figure 7.3 show that the signal-to-noise ratio with μ-law compression remains relatively constant over a wide range of input levels.

μ-law quantization for speech is standardized in the CCITT G.711 standard. In particular, 8-bit, $\mu = 255$, log-PCM with $f_s = 8000$ samples/s is widely used for digital telephony. This is an example of where speech quality is deliberately compromised in order to achieve a much lower bit rate. Once the bandwidth restriction has been imposed, 8-bit log-PCM introduces little or no perceptible distortion.

This configuration is often referred to as "toll quality" because when it was introduced at the beginning of the digital telephony era, it was perceived to render speech equivalent to speech transmitted over the best long distance lines. Nowadays, readily available hardware devices convert analog signals directly into binary-coded μ-law samples and also expand compressed samples back to uniform scale for conversion back to analog signals. Such an A-to-D/D-to-A device is called a encoder/decoder or "codec." If the digital output of a codec is used as input to a speech processing algorithm, it is generally necessary to restore the linearity of the amplitudes through the inverse of (7.4).

7.1.3 Non-Uniform and Adaptive Quantization

μ-law compression is an example of non-uniform quantization. It is based on the intuitive notion of constant percentage error. A more rigorous approach is to design a non-uniform quantizer that minimizes the mean-squared quantization error. To do this analytically, it is necessary to know the probability distribution of the signal sample values so that the most probable samples, which for speech are the low amplitude samples, will incur less error than the least probable samples. To apply this idea to the design of a non-uniform quantizer for speech requires an assumption of an analytical form for the probability distribution or some algorithmic approach based on measured distributions. The fundamentals of optimum quantization were established by Lloyd [71] and Max [79]. Paez and Glisson [91] gave an algorithm for designing optimum quantizers for assumed Laplace and Gamma probability densities, which are useful approximations to measured distributions for speech. Lloyd [71] gave an algorithm for designing optimum non-uniform quantizers based on sampled speech signals.[7] Optimum non-uniform quantizers can improve the signal-to-noise ratio by as much as 6 dB over μ-law quantizers with the same number of bits, but with little or no improvement in perceived quality of reproduction, however.

[7] Lloyd's work was initially published in a Bell Laboratories Technical Note with portions of the material having been presented at the Institute of Mathematical Statistics Meeting in Atlantic City, New Jersey in September 1957. Subsequently this pioneering work was published in the open literature in March 1982 [71].

Another way to deal with the wide dynamic range of speech is to let the quantizer step size vary with time. When the quantizer in a PCM system is adaptive, the system is called an adaptive PCM (APCM) system. For example, the step size could satisfy

$$\Delta[n] = \Delta_0 (E_{\hat{n}})^{1/2}, \tag{7.5}$$

where Δ_0 is a constant and $E_{\hat{n}}$ is the short-time energy of the speech signal defined, for example, as in (4.6). Such adaption of the step size is equivalent to fixed quantization of the signal after division by the rms signal amplitude $(E_{\hat{n}})^{1/2}$. With this definition, the step size will go up and down with the local rms amplitude of the speech signal. The adaptation speed can be adjusted by varying the analysis window size. If the step size control is based on the unquantized signal samples, the quantizer is called a *feed-forward* adaptive quantizer, and the step size information must be transmitted or stored along with the quantized sample, thereby adding to the information rate. To amortize this overhead, feedforward quantization is usually applied to blocks of speech samples. On the other hand, if the step size control is based on past quantized samples, the quantizer is a *feedback* adaptive quantizer. Jayant [57] studied a class of feedback adaptive quantizers where the step size $\Delta[n]$ is a function of the step size for the previous sample, $\Delta[n-1]$. In this approach, if the previous sample is quantized using step size $\Delta[n-1]$ to one of the low quantization levels, then the step size is decreased for the next sample. On the other hand, the step size is increased if the previous sample was quantized in one of the highest levels. By basing the adaptation on the previous sample, it is not necessary to transmit the step size. It can be derived at the decoder by the same algorithm as used for encoding.

Adaptive quantizers can achieve about 6 dB improvement (equivalent to adding one bit) over fixed quantizers [57, 58].

7.2 Digital Speech Coding

As we have seen, the data rate of sampled and quantized speech is $B \cdot f_s$. Virtually perfect perceptual quality is achievable with high data

rate.[8] By reducing B or f_s, the bit rate can be reduced, but perceptual quality may suffer. Reducing f_s requires bandwidth reduction, and reducing B too much will introduce audible distortion that may resemble random noise. The main objective in digital speech coding (sometimes called speech compression) is to lower the bit rate while maintaining an adequate level of perceptual fidelity. In addition to the two dimensions of quality and bit rate, the complexity (in terms of digital computation) is often of equal concern.

Since the 1930s, engineers and speech scientists have worked toward the ultimate goal of achieving more efficient representations. This work has led to a basic approach in speech coding that is based on the use of DSP techniques and the incorporation of knowledge of speech production and perception into the quantization process. There are many ways that this can be done. Traditionally, speech coding methods have been classified according to whether they attempt to preserve the waveform of the speech signal or whether they only seek to maintain an acceptable level of perceptual quality and intelligibility. The former are generally called *waveform coders*, and the latter *model-based coders*. Model-based coders are designed for obtaining efficient digital representations of speech and only speech. Straightforward sampling and uniform quantization (PCM) is perhaps the only pure waveform coder. Non-uniform quantization and adaptive quantization are simple attempts to build in properties of the speech signal (time-varying amplitude distribution) and speech perception (quantization noise is masked by loud sounds), but these extensions still are aimed at preserving the waveform.

Most modern model-based coding methods are based on the notion that the speech signal can be represented in terms of an excitation signal and a time-varying vocal tract system. These two components are quantized, and then a "quantized" speech signal can be reconstructed by exciting the quantized filter with the quantized excitation. Figure 7.4 illustrates why linear predictive analysis can be fruitfully applied for model-based coding. The upper plot is a segment of a speech signal.

[8] If the signal that is reconstructed from the digital representation is not perceptibly different from the original analog speech signal, the digital representation is often referred to as being "transparent."

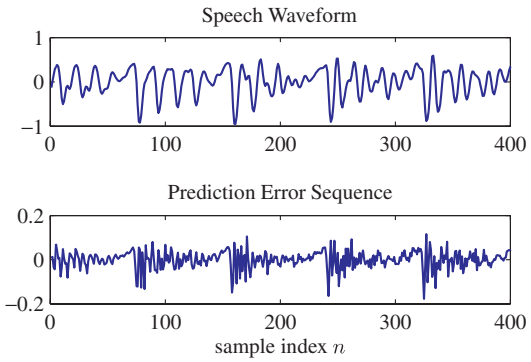

Fig. 7.4 Speech signal and corresponding prediction error signal for a 12th-order predictor.

The lower plot is the output of a linear prediction error filter $A(z)$ (with $p = 12$) that was derived from the given segment by the techniques of Chapter 6. Note that the prediction error sequence amplitude of Figure 7.4 is a factor of five lower than that of the signal itself, which means that for a fixed number of bits, this segment of the prediction error could be quantized with a smaller step size than the waveform itself. If the quantized prediction error (residual) segment is used as input to the corresponding vocal tract filter $H(z) = 1/A(z)$, an approximation to the original segment of speech would be obtained.[9] While direct implementation of this idea does not lead to a practical method of speech coding, it is nevertheless suggestive of more practical schemes which we will discuss.

Today, the line between waveform coders and model-based coders is not distinct. A more useful classification of speech coders focusses on how the speech production and perception models are incorporated into the quantization process. One class of systems simply attempts to extract an excitation signal and a vocal tract system from the speech signal without any attempt to preserve a relationship between the waveforms of the original and the quantized speech. Such systems are attractive because they can be implemented with modest computation.

[9] To implement this time-varying inverse filtering/reconstruction requires care in fitting the segments of the signals together at the block boundaries. Overlap-add methods [89] can be used effectively for this purpose.

These coders, which we designate as *open-loop coders*, are also called *vocoders* (*vo*ice *coder*) since they are based on the principles established by H. Dudley early in the history of speech processing research [30]. A second class of coders employs the source/system model for speech production inside a feedback loop, and thus are called *closed-loop coders*. These compare the quantized output to the original input and attempt to minimize the difference between the two in some prescribed sense. Differential PCM systems are simple examples of this class, but increased availability of computational power has made it possible to implement much more sophisticated closed-loop systems called *analysis-by-synthesis coders*. Closed loop systems, since they explicitly attempt to minimize a time-domain distortion measure, often do a good job of preserving the speech waveform while employing many of the same techniques used in open-loop systems.

7.3 Closed-Loop Coders

7.3.1 Predictive Coding

The essential features of predictive coding of speech were set forth in a classic paper by Atal and Schroeder [5], although the basic principles of predictive quantization were introduced by Cutler [26]. Figure 7.5 shows a general block diagram of a large class of speech coding systems that are called adaptive differential PCM (ADPCM) systems. These systems are generally classified as waveform coders, but we prefer to emphasize that they are closed-loop, model-based systems that also preserve the waveform. The reader should ignore initially the blocks concerned with adaptation and all the dotted control lines and focus instead on the core DPCM system, which is comprised of a feedback structure that includes the blocks labeled Q and P. The quantizer Q can have a number of levels ranging from 2 to much higher, it can be uniform or non-uniform, and it can be adaptive or not, but irrespective of the type of quantizer, the quantized output can be expressed as $\hat{d}[n] = d[n] + e[n]$, where $e[n]$ is the quantization error. The block labeled P is a linear predictor, so

$$\tilde{x}[n] = \sum_{k=1}^{p} \alpha_k \hat{x}[n-k], \tag{7.6}$$

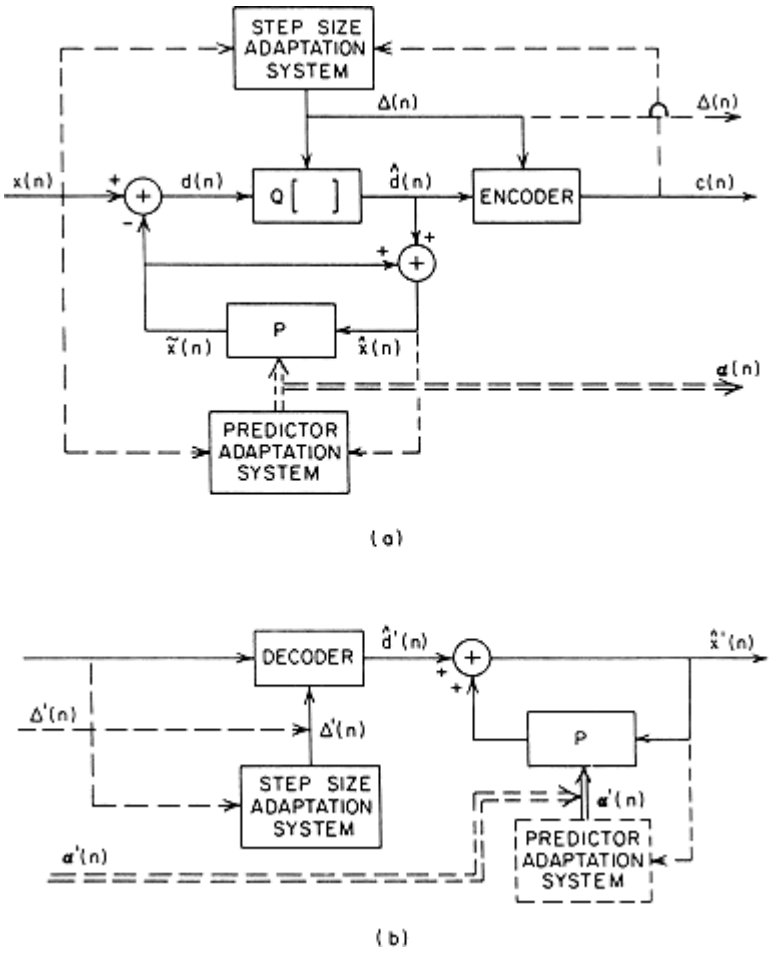

Fig. 7.5 General block diagram for adaptive differential PCM (ADPCM).

i.e., the signal $\tilde{x}[n]$ is predicted based on p past samples of the signal $\hat{x}[n]$.[10] The signal $d[n] = x[n] - \tilde{x}[n]$, the difference between the input and the predicted signal, is the input to the quantizer. Finally, the

[10] In a feed-forward adaptive predictor, the predictor parameters are estimated from the input signal $x[n]$, although, as shown in Figure 7.5, they can also be estimated from past samples of the reconstructed signal $\hat{x}[n]$. The latter would be termed a feedback adaptive predictor.

relationship between $\hat{x}[n]$ and $\hat{d}[n]$ is

$$\hat{x}[n] = \sum_{k=1}^{p} \alpha_k \hat{x}[n-k] + \hat{d}[n], \tag{7.7}$$

i.e., the signal $\hat{x}[n]$ is the output of what we have called the "vocal tract filter," and $\hat{d}[n]$ is the quantized excitation signal. Finally, since $\hat{x}[n] = \tilde{x}[n] + \hat{d}[n]$ it follows that

$$\hat{x}[n] = x[n] + e[n]. \tag{7.8}$$

This is the key result for DPCM systems. It says that no matter what predictor is used in Figure 7.5, the quantization error in $\hat{x}[n]$ is identical to the quantization error in the quantized excitation signal $\hat{d}[n]$. If the prediction is good, then the variance of $d[n]$ will be less than the variance of $x[n]$ so it will be possible to use a smaller step size and therefore to reconstruct $\hat{x}[n]$ with less error than if $x[n]$ were quantized directly. The feedback structure of Figure 7.5 contains within it a source/system speech model, which, as shown in Figure 7.5(b), is the system needed to reconstruct the quantized speech signal $\hat{x}[n]$ from the coded difference signal input.

From (7.8), the signal-to-noise ratio of the ADPCM system is $\mathrm{SNR} = \sigma_x^2/\sigma_e^2$. A simple, but informative modification leads to

$$\mathrm{SNR} = \frac{\sigma_x^2}{\sigma_d^2} \cdot \frac{\sigma_d^2}{\sigma_e^2} = G_P \cdot \mathrm{SNR}_Q, \tag{7.9}$$

where the obvious definitions apply.

SNR_Q is the signal-to-noise ratio of the quantizer, which, for fine quantization (i.e., large number of quantization levels with small step size), is given by (7.3). The number of bits B determines the bit-rate of the coded difference signal. As shown in Figure 7.5, the quantizer can be either feedforward- or feedback-adaptive. As indicated, if the quantizer is feedforward adaptive, step size information must be part of the digital representation.

The quantity G_P is called the prediction gain, i.e., if $G_P > 1$ it represents an improvement gained by placing a predictor based on the speech model inside the feedback loop around the quantizer. Neglecting

the correlation between the signal and the quantization error, it can be shown that

$$G_P = \frac{\sigma_x^2}{\sigma_d^2} = \frac{1}{1 - \sum_{k=1}^{p} \alpha_k \rho[k]}, \tag{7.10}$$

where $\rho[k] = \phi[k]/\phi[0]$ is the normalized autocorrelation function used to compute the optimum predictor coefficients. The predictor can be either fixed or adaptive, and adaptive predictors can be either feedforward or feedback (derived by analyzing past samples of the quantized speech signal reconstructed as part of the coder). Fixed predictors can be designed based on long-term average correlation functions. The lower figure in Figure 7.6 shows an estimated long-term autocorrelation function for speech sampled at an 8 kHz sampling rate. The upper plot shows $10\log_{10} G_P$ computed from (7.10) as a function of predictor order p. Note that a first-order fixed predictor can yield about 6 dB prediction gain so that either the quantizer can have 1 bit less or the

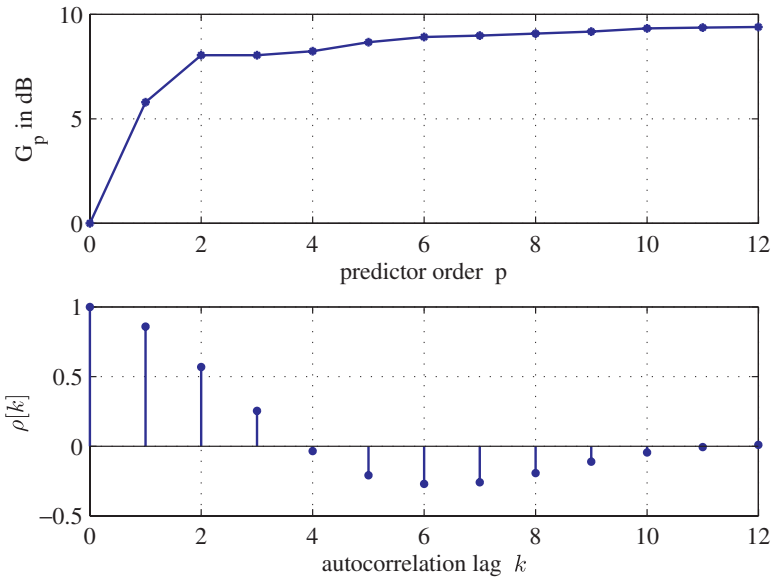

Fig. 7.6 Long-term autocorrelation function $\rho[m]$ (lower plot) and corresponding prediction gain G_P.

reconstructed signal can have a 6 dB higher SNR. Higher-order fixed predictors can achieve about 4 dB more prediction gain, and adapting the predictor at a phoneme rate can produce an additional 4 dB of gain [84]. Great flexibility is inherent in the block diagram of Figure 7.5, and not surprisingly, many systems based upon the basic principle of differential quantization have been proposed, studied, and implemented as standard systems. Here we can only mention a few of the most important.

7.3.2 Delta Modulation

Delta modulation (DM) systems are the simplest differential coding systems since they use a 1-bit quantizer and usually only a first-order predictor. DM systems originated in the classic work of Cutler [26] and deJager [28]. While DM systems evolved somewhat independently of the more general differential coding methods, they nevertheless fit neatly into the general theory of predictive coding. To see how such systems can work, note that the optimum predictor coefficient for a first-order predictor is $\alpha_1 = \rho[1]$, so from (7.10) it follows that the prediction gain for a first-order predictor is

$$G_P = \frac{1}{1 - \rho^2[1]}. \tag{7.11}$$

Furthermore, note the nature of the long-term autocorrelation function in Figure 7.6. This correlation function is for $f_s = 8000$ samples/s. If the same bandwidth speech signal were over-sampled at a sampling rate higher than 8000 samples/s, we could expect that the correlation value $\rho[1]$ would lie on a smooth curve interpolating between samples 0 and 1 in Figure 7.6, i.e., as f_s gets large, both $\rho[1]$ and α_1 approach unity and G_P gets large. Thus, even though a 1-bit quantizer has a very low SNR, this can be compensated by the prediction gain due to oversampling.

Delta modulation systems can be implemented with very simple hardware, and in contrast to more general predictive coding systems, they are not generally implemented with block processing. Instead, adaptation algorithms are usually based on the bit stream at the output of the 1-bit quantizer. The simplest delta modulators use a very high

sampling rate with a fixed predictor. Their bit rate, being equal to f_s, must be very high to achieve good quality reproduction. This limitation can be mitigated with only modest increase in complexity by using an adaptive 1-bit quantizer [47, 56]. For example, Jayant [56] showed that for bandlimited (3 kHz) speech, adaptive delta modulation (ADM) with bit rate (sampling rate) 40 kbits/s has about the same SNR as 6-bit log PCM sampled at 6.6 kHz. Furthermore, he showed that doubling the sampling rate of his ADM system improved the SNR by about 10 dB.

Adaptive delta modulation has the following advantages: it is very simple to implement, only bit synchronization is required in transmission, it has very low coding delay, and it can be adjusted to be robust to channel errors. For these reasons, adaptive delta modulation (ADM) is still used for terminal-cost-sensitive digital transmission applications.

7.3.3 Adaptive Differential PCM Systems

Differential quantization with a multi-bit quantizer is called differential PCM (DPCM). As depicted in Figure 7.5, adaptive differential PCM systems can have any combination of adaptive or fixed quantizers and/or predictors. Generally, the operations depicted in Figure 7.5 are implemented on short blocks of input signal samples, which introduces delay. A careful comparison of a variety of ADPCM systems by Noll [84] gives a valuable perspective on the relative contributions of the quantizer and predictor. Noll compared log PCM, adaptive PCM, fixed DPCM, and three ADPCM systems all using quantizers of 2, 3, 4, and 5-bits with 8 kHz sampling rate. For all bit rates (16, 24, 32, and 40 kbps), the results were as follows[11]:

(1) Log PCM had the lowest SNR.
(2) Adapting the quantizer (APCM) improved SNR by 6 dB.
(3) Adding first-order fixed or adaptive prediction improved the SNR by about 4 dB over APCM.

[11] The bit rates mentioned do not include overhead information for the quantized prediction coefficients and quantizer step sizes. See Section 7.3.3.1 for a discussion of this issue.

(4) A fourth-order adaptive predictor added about 4 dB, and increasing the predictor order to 12 added only 2 dB more.

The superior performance of ADPCM relative to log-PCM and its relatively low computational demands have led to several standard versions (ITU G.721, G.726, G.727), which are often operated at 32 kbps where quality is superior to log-PCM (ITU G.711) at 64 kbps.

The classic paper by Atal and Schroeder [5] contained a number of ideas that have since been applied with great success in adaptive predictive coding of speech. One of these concerns the quantization noise introduced by ADPCM, which we have seen is simply added to the input signal in the reconstruction process. If we invoke the white noise approximation for quantization error, it is clear that the noise will be most prominent in the speech at high frequencies where speech spectrum amplitudes are low. A simple solution to this problem is to pre-emphasize the speech signal before coding, i.e., filter the speech with a linear filter that boosts the high-frequency (HF) part of the spectrum. After reconstruction of the pre-emphasized speech, it can be filtered with the inverse system to restore the spectral balance, and in the process, the noise spectrum will take on the shape of the de-emphasis filter spectrum. A simple pre-emphasis system of this type has system function $(1 - \gamma z^{-1})$ where γ is less than one.[12] A more sophisticated application of this idea is to replace the simple fixed pre-emphasis filter with a time-varying filter designed to shape the quantization noise spectrum. An equivalent approach is to define a new feedback coding system where the quantization noise is computed as the difference between the input and output of the quantizer and then shaped before adding to the prediction residual. All these approaches raise the noise spectrum at low frequencies and lower it at high frequencies, thus taking advantage of the masking effects of the prominent low frequencies in speech [2].

Another idea that has been applied effectively in ADPCM systems as well as analysis-by-synthesis systems is to include a long-delay predictor to capture the periodicity as well as the short-delay correlation

[12] Values around $\gamma = 0.4$ for $f_s = 8$ kHz work best. If γ is too close to one, the low frequency noise is emphasized too much.

inherent in voiced speech. Including the simplest long-delay predictor would change (7.6) to

$$\tilde{x}[n] = \beta\hat{x}[n-M] + \sum_{k=1}^{p}\alpha_k(\hat{x}[n] - \beta\hat{x}[n-M]). \tag{7.12}$$

The parameter M is essentially the pitch period (in samples) of voiced speech, and the parameter β accounts for amplitude variations between periods. The predictor parameters in (7.12) cannot be jointly optimized. One sub-optimal approach is to estimate β and M first by determining M that maximizes the correlation around the expected pitch period and setting β equal to the normalized correlation at M. Then the short-delay predictor parameters α_k are estimated from the output of the long-delay prediction error filter.[13] When this type of predictor is used, the "vocal tract system" used in the decoder would have system function

$$H(z) = \left(\frac{1}{1 - \sum_{k=1}^{p}\alpha_k z^{-k}}\right)\left(\frac{1}{1-\beta z^{-M}}\right). \tag{7.13}$$

Adding long-delay prediction takes more information out of the speech signal and encodes it in the predictor. Even more complicated predictors with multiple delays and gains have been found to improve the performance of ADPCM systems significantly [2].

7.3.3.1 Coding the ADPCM Parameters

A glance at Figure 7.5 shows that if the quantizer and predictor are fixed or feedback-adaptive, then only the coded difference signal is needed to reconstruct the speech signal (assuming that the decoder has the same coefficients and feedback-adaptation algorithms). Otherwise, the predictor coefficients and quantizer step size must be included as auxiliary data (side information) along with the quantized difference

[13] Normally, this analysis would be performed on the input speech signal with the resulting predictor coefficients applied to the reconstructed signal as in (7.12).

signal. The difference signal will have the same sampling rate as the input signal. The step size and predictor parameters will be estimated and changed at a much lower sampling rate, e.g., 50–100 times/s. The total bit rate will be the sum of all the bit rates.

Predictor Quantization The predictor parameters must be quantized for efficient digital representation of the speech signal. As discussed in Chapter 6, the short-delay predictors can be represented in many equivalent ways, almost all of which are preferable to direct quantization of the predictor coefficients themselves. The PARCOR coefficients can be quantized with a non-uniform quantizer or transformed with an inverse sine or hyperbolic tangent function to flatten their statistical distribution and then quantized with a fixed uniform quantizer. Each coefficient can be allocated a number of bits contingent on its importance in accurately representing the speech spectrum [131].

Atal [2] reported a system which used a 20th-order predictor and quantized the resulting set of PARCOR coefficients after transformation with an inverse sine function. The number of bits per coefficient ranged from 5 for each of the lowest two PARCORs down to 1 each for the six highest-order PARCORs, yielding a total of 40 bits per frame. If the parameters are up-dated 100 times/s, then a total of 4000 bps for the short-delay predictor information is required. To save bit rate, it is possible to up-date the short-delay predictor 50 times/s for a total bit rate contribution of 2000 bps [2]. The long delay predictor has a delay parameter M, which requires 7 bits to cover the range of pitch periods to be expected with 8 kHz sampling rate of the input. The long-delay predictor in the system mentioned above used delays of $M-1, M, M+1$ and three gains each quantized to 4 or 5 bit accuracy. This gave a total of 20 bits/frame and added 2000 bps to the overall bit rate.

Vector Quantization for Predictor Quantization Another approach to quantizing the predictor parameters is called vector quantization, or VQ [45]. The basic principle of vector quantization is depicted in Figure 7.7. VQ can be applied in any context where a set of parameters naturally groups together into a vector (in the sense of

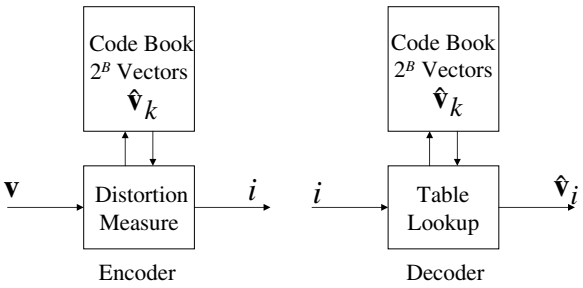

Fig. 7.7 Vector quantization.

linear algebra).[14] For example the predictor coefficients can be represented by vectors of predictor coefficients, PARCOR coefficients, cepstrum values, log area ratios, or line spectrum frequencies. In VQ, a vector, $\mathbf{v}$, to be quantized is compared exhaustively to a "codebook" populated with representative vectors of that type. The index $\boldsymbol{i}$ of the closest vector $\hat{\mathbf{v}}_{\boldsymbol{i}}$ to $\mathbf{v}$, according to a prescribed distortion measure, is returned from the exhaustive search. This index $\boldsymbol{i}$ then represents the quantized vector in the sense that if the codebook is known, the corresponding quantized vector $\hat{\mathbf{v}}_{\boldsymbol{i}}$ can be looked up. If the codebook has 2^B entries, the index $\boldsymbol{i}$ can be represented by a B bit number.

The design of a vector quantizer requires a training set consisting of a large number (L) of examples of vectors that are drawn from the same distribution as vectors to be quantized later, along with a distortion measure for comparing two vectors. Using an iterative process, 2^B codebook vectors are formed from the training set of vectors. The "training phase" is computationally intense, but need only be done once, and it is facilitated by the LBG algorithm [70]. One advantage of VQ is that the distortion measure can be designed to sort the vectors into perceptually relevant prototype vectors. The resulting codebook is then used to quantize general test vectors. Typically a VQ training set consists of at least 10 times the ultimate number of codebook vectors, i.e., $L \geq 10 \cdot 2^B$.

[14] Independent quantization of individual speech samples or individual predictor coefficients is called scalar quantization.

The quantization process is also time consuming because a given vector must be systematically compared with all the codebook vectors to determine which one it is closest to. This has led to numerous innovations in codebook structure that speed up the look up process [45].

Difference Signal Quantization It is necessary to code the difference signal in ADPCM. Since the difference signal has the same sampling rate as the input signal, it is desirable to use as few bits as possible for the quantizer. If straightforward B-bit uniform quantization is used, the additional bit rate would be $B \cdot f_s$. This was the approach used in earlier studies by Noll [84] as mentioned in Section 7.3.3. In order to reduce the bit rate for coding the difference samples, some sort of block coding must be applied. One approach is to design a multi-bit quantizer that only operates on the largest samples of the difference signal. Atal and Schroeder [2, 7] proposed to precede the quantizer by a "center clipper," which is a system that sets to zero value all samples whose amplitudes are within a threshold band and passes those samples above threshold unchanged. By adjusting the threshold, the number of zero samples can be controlled. For high thresholds, long runs of zero samples result, and the entropy of the quantized difference signal becomes less than one. The blocks of zero samples can be encoded efficiently using variable-length-to-block codes yielding average bits/sample on the order of 0.7 [2]. For a sampling rate of 8 kHz, this coding method needs a total of 5600 bps for the difference signal. When this bit rate of the differential coder is added to the bit rate for quantizing the predictor information (≈4000 bps) the total comes to approximately 10,000 bps.

7.3.3.2 Quality vs. Bit Rate for ADPCM Coders

ADPCM systems normally operate with an 8 kHz sampling rate, which is a compromise aimed at providing adequate intelligibility and moderate bit rate for voice communication. With this sampling rate, the speech signal must be bandlimited to somewhat less than 4 kHz by a lowpass filter. Adaptive prediction and quantization can easily lower the bit rate to 32 kbps with no degradation in perceived quality (with

64 kbps log PCM toll quality as a reference). Of course, raising the bit rate for ADPCM above 64 kbps cannot improve the quality over log-PCM because of the inherent frequency limitation. However, the bit rate can be lowered below 32 kbps with only modest distortion until about 10 kbps. Below this value, quality degrades significantly. In order to achieve near-toll-quality at low bit rates, all the techniques discussed above and more must be brought into play. This increases the system complexity, although not unreasonably so for current DSP hardware. Thus, ADPCM is an attractive coding method when toll quality is required at modest cost, and where adequate transmission and/or storage capacity is available to support bit rates on the order of 10 kbps.

7.3.4 Analysis-by-Synthesis Coding

While ADPCM coding can produce excellent results at moderate bit rates, its performance is fundamentally constrained by the fact that the difference signal has the same sampling rate as the input signal. The center clipping quantizer produces a difference signal that can be coded efficiently. However, this approach is clearly not optimal since the center clipper throws away information in order to obtain a sparse sequence. What is needed is a way of creating an excitation signal for the vocal tract filter that is both efficient to code and also produces decoded speech of high quality. This can be done within the same closed-loop framework as ADPCM, but with some significant modifications.

7.3.4.1 Basic Analysis-by-Synthesis Coding System

Figure 7.8 shows the block diagram of another class of closed-loop digital speech coders. These systems are called "analysis-by-synthesis coders" because the excitation is built up using an iterative process to produce a "synthetic" vocal tract filter output $\hat{x}[n]$ that matches the input speech signal according to a perceptually weighted error criterion. As in the case of the most sophisticated ADPCM systems, the operations of Figure 7.8 are carried out on blocks of speech samples. In particular, the difference, $d[n]$, between the input, $x[n]$, and the output of the vocal tract filter, $\hat{x}[n]$, is filtered with a linear filter called the

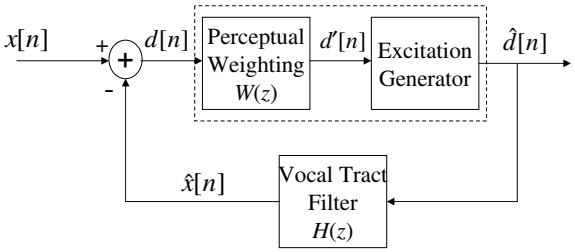

Fig. 7.8 Structure of analysis-by-synthesis speech coders.

perceptual weighting filter, $W(z)$. As the first step in coding a block of speech samples, both the vocal tract filter and the perceptual weighting filter are derived from a linear predictive analysis of the block. Then, the excitation signal is determined from the perceptually weighted difference signal $d'[n]$ by an algorithm represented by the block labeled "Excitation Generator."

Note the similarity of Figure 7.8 to the core ADPCM diagram of Figure 7.5. The perceptual weighting and excitation generator inside the dotted box play the role played by the quantizer in ADPCM, where an adaptive quantization algorithm operates on $d[n]$ to produce a quantized difference signal $\hat{d}[n]$, which is the input to the vocal tract system. In ADPCM, the vocal tract model is in the same position in the closed-loop system, but instead of the synthetic output $\hat{x}[n]$, a signal $\tilde{x}[n]$ predicted from $\hat{x}[n]$ is subtracted from the input to form the difference signal. This is a key difference. In ADPCM, the synthetic output differs from the input $x[n]$ by the quantization error. In analysis-by-synthesis, $\hat{x}[n] = x[n] - d[n]$, i.e., the reconstruction error is $-d[n]$, and a perceptually weighted version of that error is minimized in the mean-squared sense by the selection of the excitation $\hat{d}[n]$.

7.3.4.2 Perceptual Weighting of the Difference Signal

Since $-d[n]$ is the error in the reconstructed signal, it is desirable to shape its spectrum to take advantage of perceptual masking effects. In ADPCM, this is accomplished by quantization noise feedback or preemphasis/deemphasis filtering. In analysis-by-synthesis coding, the input

to the vocal tract filter is determined so as to minimize the perceptually-weighted error $d'[n]$. The weighting is implemented by linear filtering, i.e., $d'[n] = d[n] * w[n]$ with the weighting filter usually defined in terms of the vocal tract filter as the linear system with system function

$$W(z) = \frac{A(z/\alpha_1)}{A(z/\alpha_2)} = \frac{H(z/\alpha_2)}{H(z/\alpha_1)}. \tag{7.14}$$

The poles of $W(z)$ lie at the same angle but at α_2 times the radii of the poles of $H(z)$, and the zeros of $W(z)$ are at the same angles but at radii of α_1 times the radii of the poles of $H(z)$. If $\alpha_1 > \alpha_2$ the frequency response is like a "controlled" inverse filter for $H(z)$, which is the shape desired. Figure 7.9 shows the frequency response of such a filter, where typical values of $\alpha_1 = 0.9$ and $\alpha_2 = 0.4$ are used in (7.14). Clearly, this filter tends to emphasize the high frequencies (where the vocal tract filter gain is low) and it deemphasizes the low frequencies in the error signal. Thus, it follows that the error will be distributed in frequency so that relatively more error occurs at low frequencies, where, in this case, such errors would be masked by the high amplitude low frequencies. By varying α_1 and α_2 in (7.14) the relative distribution of error can be adjusted.

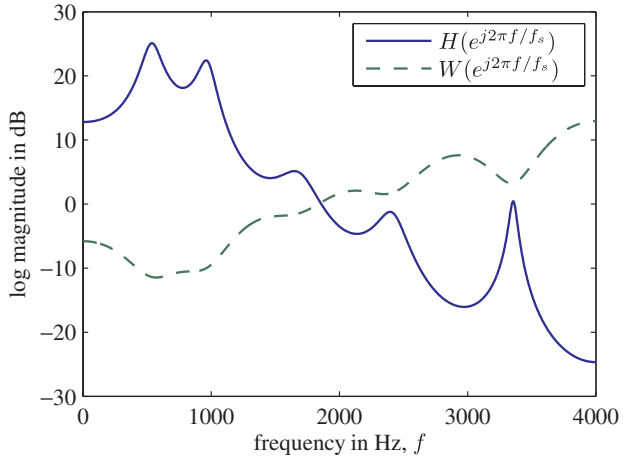

Fig. 7.9 Comparison of frequency responses of vocal tract filter and perceptual weighting filter in analysis-by-synthesis coding.

7.3.4.3 Generating the Excitation Signal

Most analysis-by-synthesis systems generate the excitation from a finite fixed collection of input components, which we designate here as $f_\gamma[n]$ for $0 \leq n \leq L-1$, where L is the excitation frame length and γ ranges over the finite set of components. The input is composed of a finite sum of scaled components selected from the given collection, i.e.,

$$\hat{d}[n] = \sum_{k=1}^{N} \beta_k f_{\gamma_k}[n]. \tag{7.15}$$

The β_ks and the sequences $f_{\gamma_k}[n]$ are chosen to minimize

$$E = \sum_{n=0}^{L-1} ((x[n] - h[n] * \hat{d}[n]) * w[n])^2, \tag{7.16}$$

where $h[n]$ is the vocal tract impulse response and $w[n]$ is the impulse response of the perceptual weighting filter with system function (7.14). Since the component sequences are assumed to be known at both the coder and decoder, the βs and γs are all that is needed to represent the input $\hat{d}[n]$. It is very difficult to solve simultaneously for the optimum βs and γs that minimize (7.16). However, satisfactory results are obtained by solving for the component signals one at a time.

Starting with the assumption that the excitation signal is zero during the current excitation[15] analysis frame (indexed $0 \leq n \leq L-1$), the output in the current frame due to the excitation determined for previous frames is computed and denoted $\hat{x}_0[n]$ for $0 \leq n \leq L-1$. Normally this would be a decaying signal that could be truncated after L samples. The error signal in the current frame at this initial stage of the iterative process would be $d_0[n] = x[n] - \hat{x}_0[n]$, and the perceptually weighted error would be $d'_0[n] = d_0[n] * w[n]$. Now at the first iteration stage, assume that we have determined which of the collection of input components $f_{\gamma_1}[n]$ will reduce the weighted mean-squared error the most and we have also determined the required gain β_1. By superposition, $\hat{x}_1[n] = \hat{x}_0[n] + \beta_1 f_{\gamma_1}[n] * h[n]$. The weighted error at the first iteration is $d'_1[n] = (d_0[n] - \beta_1 f_{\gamma_1}[n] * h[n] * w[n])$, or if we define the perceptually weighted vocal tract impulse response as $h'[n] = h[n] * w[n]$

[15] Several analysis frames are usually included in one linear predictive analysis frame.

and invoke the distributive property of convolution, then the weighted error sequence is $d'_1[n] = d'_0[n] - \beta_1 f_{\gamma_1}[n] * h'[n]$. This process is continued by finding the next component, subtracting it from the previously computed residual error, and so forth. Generalizing to the kth iteration, the corresponding equation takes the form:

$$d'_k[n] = d'_{k-1}[n] - \beta_k y'_k[n], \tag{7.17}$$

where $y'[n] = f_{\gamma_k}[n] * h'[n]$ is the output of the perceptually weighted vocal tract impulse response due to the input component $f_{\gamma_k}[n]$. The mean-squared error at stage k of the iteration is defined as

$$E_k = \sum_{n=0}^{L-1} (d'_k[n])^2 = \sum_{n=0}^{L-1} (d'_{k-1}[n] - \beta_k y'_k[n])^2, \tag{7.18}$$

where it is assumed that $d'_{k-1}[n]$ is the weighted difference signal that remains after $k-1$ steps of the process.

Assuming that γ_k is known, the value of β_k that minimizes E_k in (7.18) is

$$\beta_k = \frac{\sum_{n=0}^{L-1} d'_{k-1}[n] y'_k[n]}{\sum_{n=0}^{L-1} (y'[n])^2}, \tag{7.19}$$

and the corresponding minimum mean-squared error is

$$E_k^{(\min)} = \sum_{n=0}^{L-1} (d'_{k-1}[n])^2 - \beta_k^2 \sum_{n=0}^{L-1} (y'_k[n])^2. \tag{7.20}$$

While we have assumed in the above discussion that $f_{\gamma_k}[n]$ and β_k are known, we have not discussed how they can be found. Equation (7.20) suggests that the mean-squared error can be minimized by maximizing

$$\beta_k^2 \sum_{n=0}^{L-1} (y'_k[n])^2 = \frac{\left(\sum_{n=0}^{L-1} d'_{k-1}[n] y'_k[n] \right)^2}{\sum_{n=0}^{L-1} (y'_k[n])^2}, \tag{7.21}$$

which is essentially the normalized cross-correlation between the new input component and the weighted residual error $d'_{k-1}[n]$. An exhaustive search through the collection of input components will determine which component $f_{\gamma_k}[n]$ will maximize the quantity in (7.21) and thus reduce the mean-squared error the most. Once this component is found, the corresponding β_k can be found from (7.19), and the new error sequence computed from (7.17).

After N iterations of this process, the complete set of component input sequences and corresponding coefficients will have been determined.[16] If desired, the set of input sequences so determined can be assumed and a new set of βs can be found jointly by solving a set of N linear equations.

7.3.4.4 Multi-Pulse Excitation Linear Prediction (MPLP)

The procedure described in Section 7.3.4.3 is quite general since the only constraint was that the collection of component input sequences be finite. The first analysis-by-synthesis coder was called *multi-pulse linear predictive coding* [4]. In this system, the component input sequences are simply isolated unit impulse sequences, i.e., $f_\gamma[n] = \delta[n - \gamma]$, where γ is an integer such that $0 \leq \gamma \leq L - 1$. The excitation sequence derived by the process described in Section 7.3.4.3 is therefore of the form:

$$\hat{d}[n] = \sum_{k=1}^{N} \beta_k \delta[n - \gamma_k], \tag{7.22}$$

where N is usually on the order of 4–5 impulses in a 5 ms (40 samples for $f_s = 8$ kHz) excitation analysis frame.[17] In this case, an exhaustive search at each stage can be achieved by computing just one cross-correlation function and locating the impulse at the maximum of the cross-correlation. This is because the components $y'_k[n]$ are all just shifted versions of $h'[n]$.

[16] Alternatively, N need not be fixed in advance. The error reduction process can be stopped when $E_k^{(\min)}$ falls below a prescribed threshold.

[17] It is advantageous to include two or more excitation analysis frames in one linear predictive analysis frame, which normally is of 20 ms (160 samples) duration.

In a speech coder based on multi-pulse analysis-by-synthesis, the speech signal is represented by quantized versions of the prediction coefficients, the pulse locations, and the pulse amplitudes. The predictor coefficients can be coded as discussed for ADPCM by encoding any of the alternative representations either by scalar or vector quantization. A number of different approaches have been devised for coding the excitation parameters. These usually involve some sort of differential coding scheme. At bit rates on the order of 10–16 kbps multipulse coding can approach toll quality; however, below about 10 kbps, quality degrades rapidly due to the many parameters that must be coded [13].

Regular-pulse excitation (RPE) is a special case of multipulse excitation where it is easier to encode the impulse locations. Specifically after determining the location γ_1 of the first impulse, all other impulses are located at integer multiples of a fixed spacing on either side of γ_1. Thus, only γ_1, the integer multiples, and the gain constants must be encoded. Using a pre-set spacing of 4 and an excitation analysis window of 40 samples yields high quality at about 10 kbps [67]. Because the impulse locations are fixed after the first iteration, RPE requires significantly less computation than full-blown multipulse excitation analysis. One version of RPE is the basis for the 13 kbps digital coder in the GSM mobile communications system.

7.3.4.5 Code-Excited Linear Prediction (CELP)

Since the major portion of the bit rate of an analysis-by-synthesis coder lies in the excitation signal, it is not surprising that a great deal of effort has gone into schemes for finding excitation signals that are easier to encode than multipulse excitation, but yet maintain high quality of reproduction of the speech signal. The next major innovation in the history of analysis-by-synthesis systems was called code-excited linear predictive coding or CELP [112].[18] In this scheme, the excitation signal components are Gaussian random sequences stored in a

[18] Earlier work by Stewart [123] used a residual codebook (populated by the Lloyd algorithm [71]) with a low complexity trellis search.

"codebook."[19] If the codebook contains 2^M sequences, then a given sequence can be specified by an M-bit number. Typical codebook sizes are 256, 512 or 1024. Within the codebook, the sequence lengths are typically $L = 40$–60 samples (5–7.5 ms at 8 kHz sampling rate). If N sequences are used to form the excitation, then a total of $M \cdot N$ bits will be required to code the sequences. Additional bits are still required to code the β_ks. With this method, the decoder must also have a copy of the analysis codebook so that the excitation can be regenerated at the decoder.

The main disadvantage of CELP is the high computational cost of exhaustively searching the codebook at each stage of the error minimization. This is because each codebook sequence must be filtered with the perceptually weighted impulse response before computing the cross correlation with the residual error sequence. Efficient searching schemes and structured codebooks have eased the computational burden, and modern DSP hardware can easily implement the computations in real time.

The basic CELP framework has been applied in the development of numerous speech coders that operate with bit rates in the range from 4800 to 16000 bps. The Federal Standard 1016 (FS1016) was adopted by the Department of Defense for use in secure voice transmission at 4800 bps. Another system called VSELP (vector sum excited linear prediction), which uses multiple codebooks to achieve a bit rate of 8000 bps, was adopted in 1989 for North American Cellular Systems. The GSM half rate coder operating at 5600 bps is based on the IS-54 VSELP coder. Due to the block processing structure, CELP coders can introduce more than 40 ms delay into communication systems. The ITU G.728 low-delay CELP coder uses very short excitation analysis frames and backward linear prediction to achieve high quality at a bit rate of 16,000 bps with delays less than 2 ms. Still another standardized CELP system is the ITU G.729 conjugate-structure algebraic CELP (CS-ACELP) coder that is used in some mobile communication systems.

[19] While it may seem counter-intuitive that the excitation could be comprised of white noise sequence, remember that, in fact, the object of linear prediction is to produce just such a signal.

7.3.4.6 Long-Delay Predictors in Analysis-by-Synthesis Coders

Long-delay predictors have an interesting interpretation in analysis-by-synthesis coding. If we maintain a memory of one or more frames of previous values of the excitation signal, we can add a term $\beta_0 \hat{d}[n - \gamma_0]$ to (7.15). The long segment of the past excitation acts as a sort of codebook where the L-sample codebook sequences overlap by $L - 1$ samples. The first step of the excitation computation would be to compute γ_0 using (7.21) and then β_0 using (7.19). For each value of γ_0 to be tested, this requires that the weighted vocal tract impulse response be convolved with L-sample segments of the past excitation starting with sample γ_0. This can be done recursively to save computation. Then the component $\beta_0 \hat{d}[n - \gamma_0]$ can be subtracted from the initial error to start the iteration process in either the multipulse or the CELP framework. The additional bit rate for coding β_0 and γ_0 is often well worthwhile, and long-delay predictors are used in many of the standardized coders mentioned above. This incorporation of components of the past history of the excitation has been referred to as "self-excitation" [104] or the use of an "adaptive codebook" [19].

7.4 Open-Loop Coders

ADPCM coding has not been useful below about 9600 bps, multipulse coding has similar limitations, and CELP coding has not been used at bit rates below about 4800 bits/s. While these closed-loop systems have many attractive features, it has not been possible to generate excitation signals that can be coded at low bit rates and also produce high quality synthetic speech. For bit rates below 4800 bps, engineers have turned to the vocoder principles that were established decades ago. We call these systems open-loop systems because they do not determine the excitation by a feedback process.

7.4.1 The Two-State Excitation Model

Figure 7.10 shows a source/system model for speech that is closely related to physical models for speech production. As we have discussed

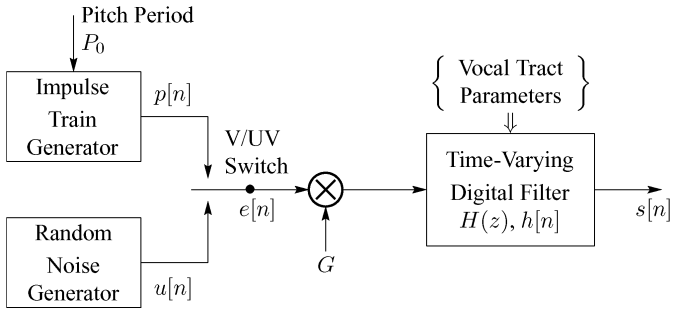

Fig. 7.10 Two-state-excitation model for speech synthesis.

in the previous chapters, the excitation model can be very simple. Unvoiced sounds are produced by exciting the system with white noise, and voiced sounds are produced by a periodic impulse train excitation, where the spacing between impulses is the pitch period, P_0. We shall refer to this as the *two-state excitation model*. The slowly-time-varying linear system models the combined effects of vocal tract transmission, radiation at the lips, and, in the case of voiced speech, the lowpass frequency shaping of the glottal pulse. The V/UV (voiced/unvoiced excitation) switch produces the alternating voiced and unvoiced segments of speech, and the gain parameter G controls the level of the filter output. When values for V/UV decision, G, P_0, and the parameters of the linear system are supplied at periodic intervals (frames) then the model becomes a speech synthesizer, as discussed in Chapter 8. When the parameters of the model are estimated directly from a speech signal, the combination of estimator and synthesizer becomes a vocoder or, as we prefer, an *open-loop analysis/synthesis speech coder*.

7.4.1.1 Pitch, Gain, and V/UV Detection

The fundamental frequency of voiced speech can range from well below 100 Hz for low-pitched male speakers to over 250 Hz for high-pitched voices of women and children. The fundamental frequency varies slowly, with time, more or less at the same rate as the vocal tract motions. It is common to estimate the fundamental frequency F_0 or equivalently,

the pitch period $P_0 = 1/F_0$ at a frame rate of about 50–100 times/s. To do this, short segments of speech are analyzed to detect periodicity (signaling voiced speech) or aperiodicity (signaling unvoiced speech).

One of the simplest, yet most effective approaches to pitch detection, operates directly on the time waveform by locating corresponding peaks and valleys in the waveform, and measuring the times between the peaks [43]. The STACF is also useful for this purpose as illustrated in Figure 4.8, which shows autocorrelation functions for both voiced speech, evincing a peak at the pitch period, and unvoiced speech, which shows no evidence of periodicity. In pitch detection applications of the short-time autocorrelation, it is common to preprocess the speech by a spectrum flattening operation such as center clipping [96] or inverse filtering [77]. This preprocessing tends to enhance the peak at the pitch period for voiced speech while suppressing the local correlation due to formant resonances.

Another approach to pitch detection is suggested by Figure 5.5, which shows the cepstra of segments of voiced and unvoiced speech. In this case, a strong peak in the expected pitch period range signals voiced speech, and the location of the peak is the pitch period. Similarly, lack of a peak in the expected range signals unvoiced speech [83]. Figure 5.6 shows a sequence of short-time cepstra which moves from unvoiced to voiced speech in going from the bottom to the top of the figure. The time variation of the pitch period is evident in the upper plots.

The gain parameter G is also found by analysis of short segments of speech. It should be chosen so that the short-time energy of the synthetic output matches the short-time energy of the input speech signal. For this purpose, the autocorrelation function value $\phi_{\hat{n}}[0]$ at lag 0 or the cepstrum value $c_{\hat{n}}[0]$ at quefrency 0 can be used to determine the energy of the segment of the input signal.

For digital coding applications, the pitch period, V/UV, and gain must be quantized. Typical values are 7 bits for pitch period ($P_0 = 0$ signals UV) and 5 bits for G. For a frame rate of 50 frames/s, this totals 600 bps, which is well below the bit rate used to encode the excitation signal in closed-loop coders such as ADPCM or CELP. Since the vocal tract filter can be coded as in ADPCM or CELP, much lower total bit

rates are common in open-loop systems. This comes at a large cost in quality of the synthetic speech output, however.

7.4.1.2 Vocal Tract System Estimation

The vocal tract system in the synthesizer of Figure 7.10 can take many forms. The primary methods that have been used have been homomorphic filtering and linear predictive analysis as discussed in Chapters 5 and 6, respectively.

The Homomorphic Vocoder Homomorphic filtering can be used to extract a sequence of impulse responses from the sequence of cepstra that result from short-time cepstrum analysis. Thus, one cepstrum computation can yield both an estimate of pitch and the vocal tract impulse response. In the original homomorphic vocoder, the impulse response was digitally coded by quantizing each cepstrum value individually (scalar quantization) [86]. The impulse response, reconstructed from the quantized cepstrum at the synthesizer, is simply convolved with the excitation created from the quantized pitch, voicing, and gain information, i.e., $s[n] = Ge[n] * h[n]$.[20] In a more recent application of homomorphic analysis in an analysis-by-synthesis framework [21], the cepstrum values were coded using vector quantization, and the excitation derived by analysis-by-synthesis as described in Section 7.3.4.3. In still another approach to digital coding, homomorphic filtering was used to remove excitation effects in the short-time spectrum and then three formant frequencies were estimated from the smoothed spectra. Figure 5.6 shows an example of the formants estimated for voiced speech. These formant frequencies were used to control the resonance frequencies of a synthesizer comprised of a cascade of second-order section IIR digital filters [111]. Such a speech coder is called a formant vocoder.

LPC Vocoder Linear predictive analysis can also be used to estimate the vocal tract system for an open-loop coder with two-state

[20] Care must be taken at frame boundaries. For example, the impulse response can be changed at the time a new pitch impulse occurs, and the resulting output can overlap into the next frame.

excitation [6]. In this case, the prediction coefficients can be coded in one of the many ways that we have already discussed. Such analysis/synthesis coders are called LPC vocoders. A vocoder of this type was standardized by the Department of Defense as the Federal Standard FS1015. This system is also called LPC-10 (or LPC-10e) because a 10th-order covariance linear predictive analysis is used to estimate the vocal tract system. The LPC-10 system has a bit rate of 2400 bps using a frame size of 22.5 ms with 12 bits/frame allocated to pitch, voicing and gain and the remaining bits allocated to the vocal tract filter coded as PARCOR coefficients.

7.4.2 Residual-Excited Linear Predictive Coding

In Section 7.2, we presented Figure 7.4 as motivation for the use of the source/system model in digital speech coding. This figure shows an example of inverse filtering of the speech signal using a prediction error filter, where the prediction error (or residual) is significantly smaller and less lowpass in nature. The speech signal can be reconstructed from the residual by passing it through the vocal tract system $H(z) = 1/A(z)$. None of the methods discussed so far attempt to directly code the prediction error signal in an open-loop manner. ADPCM and analysis-by-synthesis systems derive the excitation to the synthesis filter by a feedback process. The two-state model attempts to construct the excitation signal by direct analysis and measurement of the input speech signal. Systems that attempt to code the residual signal directly are called residual-excited linear predictive (RELP) coders.

Direct coding of the residual faces the same problem faced in ADPCM or CELP: the sampling rate is the same as that of the input, and accurate coding could require several bits per sample. Figure 7.11 shows a block diagram of a RELP coder [128]. In this system, which is quite similar to voice-excited vocoders (VEV) [113, 130], the problem of reducing the bit rate of the residual signal is attacked by reducing its bandwidth to about 800 Hz, lowering the sampling rate, and coding the samples with adaptive quantization. Adaptive delta modulation was used in [128], but APCM could be used if the sampling rate is

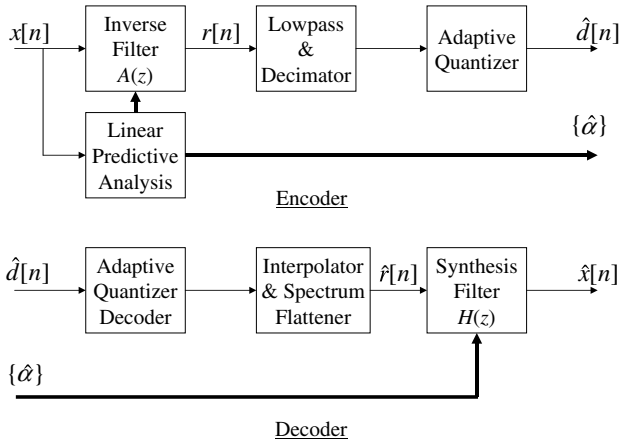

Fig. 7.11 Residual-excited linear predictive (RELP) coder and decoder.

lowered to 1600 Hz. The 800 Hz band is wide enough to contain several harmonics of the highest pitched voices. The reduced bandwidth residual is restored to full bandwidth prior to its use as an excitation signal by a nonlinear spectrum flattening operation, which restores higher harmonics of voiced speech. White noise is also added according to an empirically derived recipe. In the implementation of [128], the sampling rate of the input was 6.8 kHz and the total bit rate was 9600 kbps with 6800 bps devoted to the residual signal. The quality achieved at this rate was not significantly better than the LPC vocoder with a two-state excitation model. The principle advantage of this system is that no hard V/UV decision must be made, and no pitch detection is required. While this system did not become widely used, its basic principles can be found in subsequent open-loop coders that have produced much better speech quality at bit rates around 2400 bps.

7.4.3 Mixed Excitation Systems

While two-state excitation allows the bit rate to be quite low, the quality of the synthetic speech output leaves much to be desired. The output of such systems is often described as "buzzy," and in many cases, errors in estimating pitch period or voicing decision cause the speech to sound unnatural if not unintelligible. The weaknesses of the two-state model

for excitation spurred interest in a mixed excitation model where a hard decision between V and UV is not required. Such a model was first proposed by Makhoul et al. [75] and greatly refined by McCree and Barnwell [80].

Figure 7.12 depicts the essential features of the mixed-excitation linear predictive coder (MELP) proposed by McCree and Barnwell [80]. This configuration was developed as the result of careful experimentation, which focused one-by-one on the sources of distortions manifest in the two-state excitation coder such as buzziness and tonal distortions. The main feature is that impulse train excitation and noise excitation are added instead of switched. Prior to their addition, they each pass through a multi-band spectral shaping filter. The gains in each of five bands are coordinated between the two filters so that the spectrum of $e[n]$ is flat. This mixed excitation helps to model short-time spectral effects such as "devoicing" of certain bands during voiced speech.[21] In some situations, a "jitter" parameter ΔP is invoked to better model voicing transitions. Other important features of the MELP system are lumped into the block labeled "Enhancing Filters." This represents adaptive spectrum enhancement filters used to enhance formant regions,[22] and a spectrally flat "pulse dispersion filter" whose

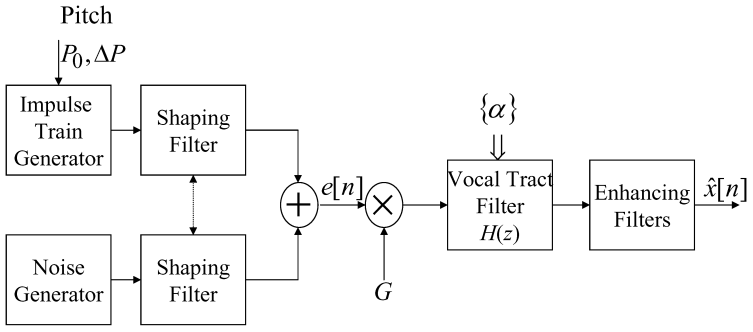

Fig. 7.12 Mixed-excitation linear predictive (MELP) decoder.

[21] Devoicing is evident in frame 9 in Figure 5.6.

[22] Such filters are also used routinely in CELP coders and are often referred to as post filters.

purpose is to reduce "peakiness" due to the minimum-phase nature of the linear predictive vocal tract system.

These modifications to the basic two-state excitation LPC vocoder produce marked improvements in the quality of reproduction of the speech signal. Several new parameters of the excitation must be estimated at analysis time and coded for transmission, but these add only slightly to either the analysis computation or the bit rate [80]. The MELP coder is said to produce speech quality at 2400 bps that is comparable to CELP coding at 4800 bps. In fact, its superior performance led to a new Department of Defense standard in 1996 and subsequently to MIL-STD-3005 and NATO STANAG 4591, which operates at 2400, 1200, and 600 bps.

7.5 Frequency-Domain Coders

Although we have discussed a wide variety of digital speech coding systems, we have really only focused on a few of the most important examples that have current relevance. There is much room for variation within the general frameworks that we have identified. There is one more class of coders that should be discussed. These are called frequency-domain coders because they are based on the principle of decomposing the speech signal into individual frequency bands. Figure 7.13 depicts the general nature of this large class of coding systems. On the far left of the diagram is a set (bank) of analysis bandpass filters. Each of these is followed by a down-sampler by a factor appropriate for the bandwidth of its bandpass input. On the far right in the diagram is a set of "bandpass interpolators" where each interpolator is composed of an up-sampler (i.e., raising the sampling rate by the same factor as the corresponding down-sampling box) followed by a bandpass filter similar to the analysis filter. If the filters are carefully designed, and the outputs of the down-samplers are connected to the corresponding inputs of the bandpass interpolators, then it is possible for the output $\hat{x}[n]$ to be virtually identical to the input $x[n]$ [24]. Such perfect reconstruction filter bank systems are the basis for a wide-ranging class of speech and audio coders called *subband coders* [25].

When the outputs of the filter bank are quantized by some sort of quantizer, the output $\hat{x}[n]$ is not equal to the input, but by careful

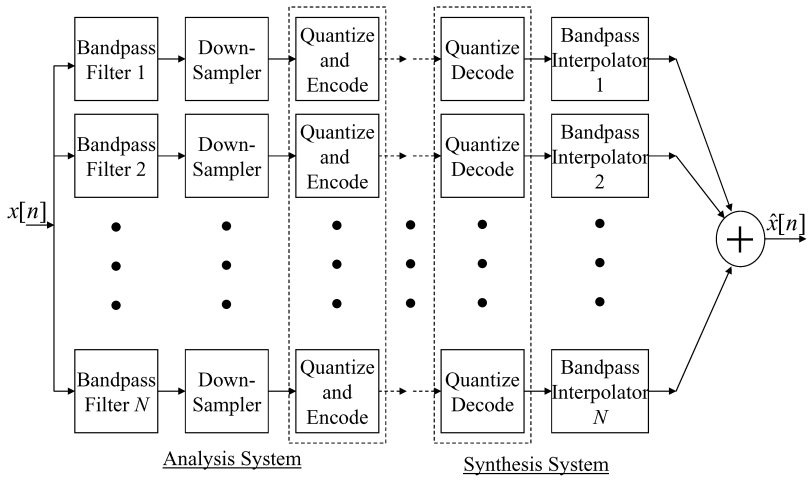

Fig. 7.13 Subband coder and decoder for speech audio.

design of the quantizer the output can be indistinguishable from the input perceptually. The goal with such coders is, of course, for the total composite bit rate[23] to be as low as possible while maintaining high quality.

In contrast to the coders that we have already discussed, which incorporated the speech production model into the quantization process, the filter bank structure incorporates important features of the speech perception model. As discussed in Chapter 3, the basilar membrane effectively performs a frequency analysis of the sound impinging on the eardrum. The coupling between points on the basilar membrane results in the masking effects that we mentioned in Chapter 3. This masking was incorporated in some of the coding systems discussed so far, but only in a rudimentary manner.

To see how subband coders work, suppose first that the individual channels are quantized independently. The quantizers can be any quantization operator that preserves the waveform of the signal. Typically, adaptive PCM is used, but ADPCM could be used in principle. Because the down-sampled channel signals are full band at the lower sampling rate, the quantization error affects all frequencies in

[23] The total bit rate will be the sum of the products of the sampling rates of the down-sampled channel signals times the number of bits allocated to each channel.

that band, but the important point is that no other bands are affected by the errors in a given band. Just as important, a particular band can be quantized according to its perceptual importance. Furthermore, bands with low energy can be encoded with correspondingly low absolute error.

The simplest approach is to pre-assign a fixed number of bits to each channel. In an early non-uniformly spaced five channel system, Crochiere et al. [25] found that subband coders operating at 16 kbps and 9.6 kbps were preferred by listeners to ADPCM coders operating at 22 and 19 kbps, respectively.

Much more sophisticated quantization schemes are possible in the configuration shown in Figure 7.13 if, as suggested by the dotted boxes, the quantization of the channels is done jointly. For example, if the channel bandwidths are all the same so that the output samples of the down-samplers can be treated as a vector, vector quantization can be used effectively [23].

Another approach is to allocate bits among the channels dynamically according to a perceptual criterion. This is the basis for modern audio coding standards such as the various MPEG standards. Such systems are based on the principles depicted in Figure 7.13. What is not shown in that figure is additional analysis processing that is done to determine how to allocate the bits among the channels. This typically involves the computation of a fine-grained spectrum analysis using the DFT. From this spectrum it is possible to determine which frequencies will mask other frequencies, and on the basis of this a threshold of audibility is determined. Using this audibility threshold, bits are allocated among the channels in an iterative process so that as much of the quantization error is inaudible as possible for the total bit budget. The details of perceptual audio coding are presented in the recent textbook by Spanias [120].

7.6 Evaluation of Coders

In this chapter, we have discussed a wide range of options for digital coding of speech signals. These systems rely heavily on the techniques of linear prediction, filter banks, and cepstrum analysis and also on models

for speech production and speech perception. All this knowledge and technique is brought to bear on the problem of reducing the bit rate of the speech representation while maintaining high quality reproduction of the speech signal.

Throughout this chapter we have mentioned bit rates, complexity of computation, processing delay, and quality as important practical dimensions of the speech coding problem. Most coding schemes have many operations and parameters that can be chosen to tradeoff among the important factors. In general, increasing the bit rate will improve quality of reproduction; however, it is important to note that for many systems, increasing the bit rate indefinitely does not necessarily continue to improve quality. For example, increasing the bit rate of an open-loop two-state-excitation LPC vocoder above about 2400 bps does not improve quality very much, but lowering the bit rate causes noticeable degradation. On the other hand, improved pitch detection and source modeling can improve quality in an LPC vocoder as witnessed by the success of the MELP system, but this generally would come with an increase in computation and processing delay.

In the final analysis, the choice of a speech coder will depend on constraints imposed by the application such as cost of coder/decoder, available transmission capacity, robustness to transmission errors, and quality requirements. Fortunately, there are many possibilities to choose from.

8

Text-to-Speech Synthesis Methods

In this chapter, we discuss systems whose goal is to convert ordinary text messages into *intelligible* and *natural sounding* synthetic speech so as to transmit information from a machine to a human user. In other words, we will be concerned with computer simulation of the upper part of the speech chain in Figure 1.2. Such systems are often referred to as text-to-speech synthesis (or TTS) systems, and their general structure is illustrated in Figure 8.1. The input to the TTS system is text and the output is synthetic speech. The two fundamental processes performed by all TTS systems are text analysis (to determine the abstract underlying linguistic description of the speech) and speech synthesis (to produce the speech sounds corresponding to the text input).

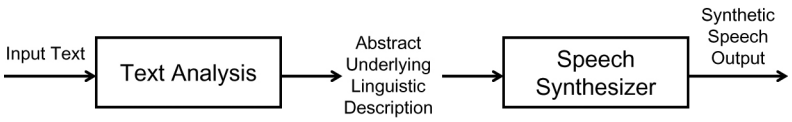

Fig. 8.1 Block diagram of general TTS system.

8.1 Text Analysis

The text analysis module of Figure 8.1 must determine three things from the input text string, namely:

(1) **pronunciation of the text string:** the text analysis process must decide on the set of phonemes that is to be spoken, the degree of stress at various points in speaking, the intonation of the speech, and the duration of each of the sounds in the utterance
(2) **syntactic structure of the sentence to be spoken:** the text analysis process must determine where to place pauses, what rate of speaking is most appropriate for the material being spoken, and how much emphasis should be given to individual words and phrases within the final spoken output speech
(3) **semantic focus and ambiguity resolution:** the text analysis process must resolve homographs (words that are spelled alike but can be pronounced differently, depending on context), and also must use rules to determine word etymology to decide on how best to pronounce names and foreign words and phrases.

Figure 8.2 shows more detail on how text analysis is performed. The input to the analysis is plain English text. The first stage of processing does some basic text processing operations, including detecting the structure of the document containing the text (e.g., email message versus paragraph of text from an encyclopedia article), normalizing the text (so as to determine how to pronounce words like proper names or homographs with multiple pronunciations), and finally performing a linguistic analysis to determine grammatical information about words and phrases within the text. The basic text processing benefits from an online dictionary of word pronunciations along with rules for determining word etymology. The output of the basic text processing step is tagged text, where the tags denote the linguistic properties of the words of the input text string.

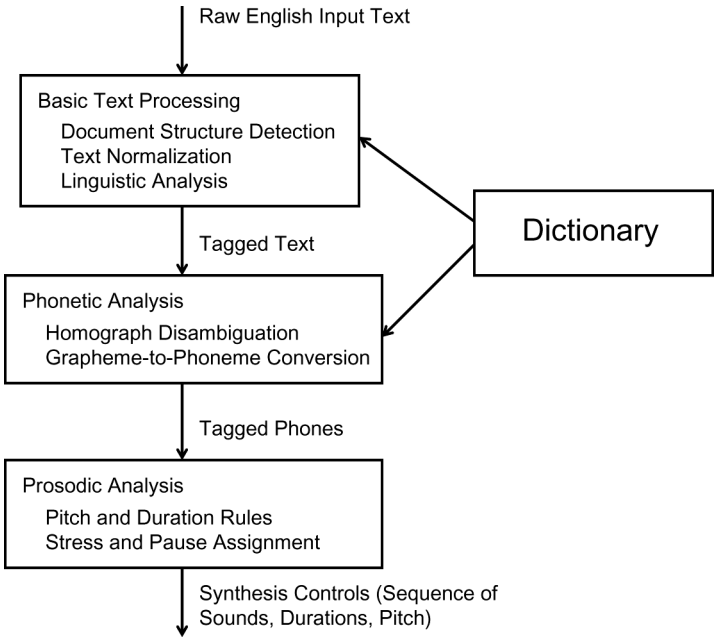

Fig. 8.2 Components of the text analysis process.

8.1.1 Document Structure Detection

The document structure detection module seeks to determine the location of all punctuation marks in the text, and to decide their significance with regard to the sentence and paragraph structure of the input text. For example, an *end of sentence* marker is usually a period, ., a question mark, ?, or an exclamation point, !. However this is not always the case as in the sentence, "This car is 72.5 in. long" where there are two periods, neither of which denote the end of the sentence.

8.1.2 Text Normalization

Text normalization methods handle a range of text problems that occur in real applications of TTS systems, including how to handle abbreviations and acronyms as in the following sentences:

Example 1: "I live on Bourbon St. in St. Louis"
Example 2: "She worked for DEC in Maynard MA"

where, in Example 1, the text "St." is pronounced as street or saint, depending on the context, and in Example 2 the acronym DEC can be pronounced as either the word "deck" (the spoken acronym) or the name of the company, i.e., Digital Equipment Corporation, but is virtually never pronounced as the letter sequence "D E C."

Other examples of text normalization include number strings like "1920" which can be pronounced as the year "nineteen twenty" or the number "one thousand, nine hundred, and twenty," and dates, times currency, account numbers, etc. Thus the string "$10.50" should be pronounced as "ten dollars and fifty cents" rather than as a sequence of characters.

One other important text normalization problem concerns the pronunciation of proper names, especially those from languages other than English.

8.1.3 Linguistic Analysis

The third step in the basic text processing block of Figure 8.2 is a linguistic analysis of the input text, with the goal of determining, for each word in the printed string, the following linguistic properties:

- the part of speech (POS) of the word
- the sense in which each word is used in the current context
- the location of phrases (or phrase groups) with a sentence (or paragraph), i.e., where a pause in speaking might be appropriate
- the presence of anaphora (e.g., the use of a pronoun to refer back to another word unit)
- the word (or words) on which emphasis are to be placed, for prominence in the sentence
- the style of speaking, e.g., irate, emotional, relaxed, etc.

A conventional parser could be used as the basis of the linguistic analysis of the printed text, but typically a simple, shallow analysis is performed since most linguistic parsers are very slow.

8.1.4 Phonetic Analysis

Ultimately, the tagged text obtained from the basic text processing block of a TTS system has to be converted to a sequence of tagged phones which describe both the sounds to be produced as well as the manner of speaking, both locally (emphasis) and globally (speaking style). The phonetic analysis block of Figure 8.2 provides the processing that enables the TTS system to perform this conversion, with the help of a pronunciation dictionary. The way in which these steps are performed is as follows.

8.1.5 Homograph Disambiguation

The homograph disambiguation operation must resolve the correct pronunciation of each word in the input string that has more than one pronunciation. The basis for this is the context in which the word occurs. One simple example of homograph disambiguation is seen in the phrases "an *absent* boy" versus the sentence "do you choose to *absent* yourself." In the first phrase the word "absent" is an adjective and the accent is on the first syllable; in the second phrase the word "absent" is a verb and the accent is on the second syllable.

8.1.6 Letter-to-Sound (LTS) Conversion

The second step of phonetic analysis is the process of grapheme-to-phoneme conversion, namely conversion from the text to (marked) speech sounds. Although there are a variety of ways of performing this analysis, perhaps the most straightforward method is to rely on a standard pronunciation dictionary, along with a set of letter-to-sound rules for words outside the dictionary.

Figure 8.3 shows the processing for a simple dictionary search for word pronunciation. Each individual word in the text string is searched independently. First a "whole word" search is initiated to see if the printed word exists, in its entirety, in the word dictionary. If so, the conversion to sounds is straightforward and the dictionary search begins on the next word. If not, as is the case most often, the dictionary search attempts to find affixes (both prefixes and suffixes) and strips them

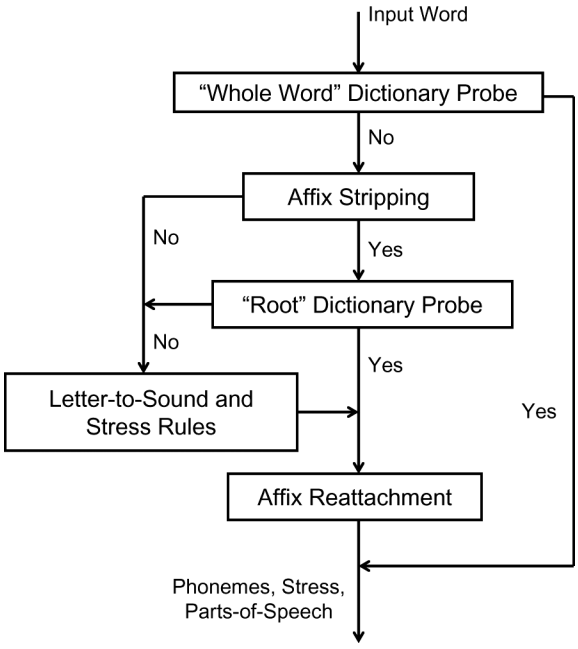

Fig. 8.3 Block diagram of dictionary search for proper word pronunciation.

from the word attempting to find the "root form" of the word, and then does another "whole word" search. If the root form is not present in the dictionary, a set of letter-to-sound rules is used to determine the best pronunciation (usually based on etymology of the word) of the root form of the word, again followed by the reattachment of stripped out affixes (including the case of no stripped out affixes).

8.1.7 Prosodic Analysis

The last step in the text analysis system of Figure 8.2 is prosodic analysis which provides the speech synthesizer with the complete set of synthesis controls, namely the sequence of speech sounds, their durations, and an associated pitch contour (variation of fundamental frequency with time). The determination of the sequence of speech sounds is mainly performed by the phonetic analysis step as outlined above. The assignment of duration and pitch contours is done by a set of pitch

and duration rules, along with a set of rules for assigning stress and determining where appropriate pauses should be inserted so that the local and global speaking rates appear to be natural.

8.2 Evolution of Speech Synthesis Systems

A summary of the progress in speech synthesis, over the period 1962–1997, is given in Figure 8.4. This figure shows that there have been 3 generations of speech synthesis systems. During the first generation (between 1962 and 1977) formant synthesis of phonemes using a terminal analog synthesizer was the dominant technology using rules which related the phonetic decomposition of the sentence to formant frequency contours. The synthesis suffered from poor intelligibility and poor naturalness. The second generation of speech synthesis methods (from 1977 to 1992) was based primarily on an LPC representation of sub-word units such as diphones (half phones). By carefully modeling and representing diphone units via LPC parameters, it was shown that good intelligibility synthetic speech could be reliably obtained from text input by concatenating the appropriate diphone units. Although the intelligibility improved dramatically

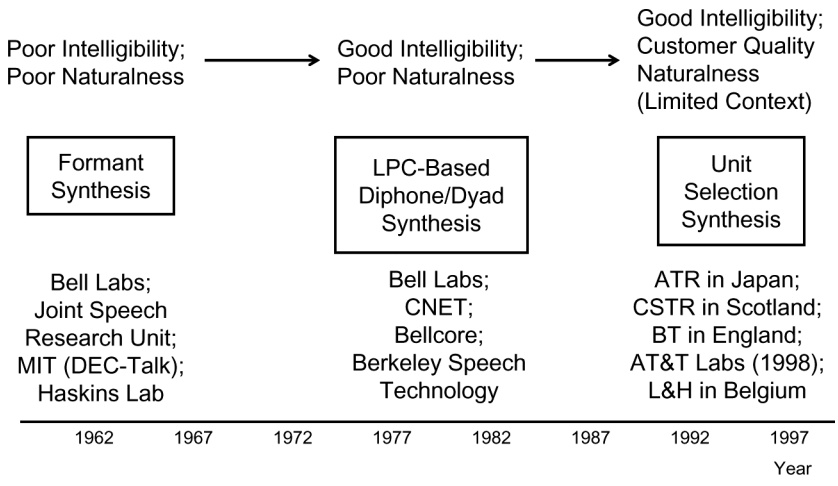

Fig. 8.4 Time line of progress in speech synthesis and TTS systems.

over first generation formant synthesis, the naturalness of the synthetic speech remained low due to the inability of single diphone units to represent all possible combinations of sound using that diphone unit. A detailed survey of progress in text-to-speech conversion up to 1987 is given in the review paper by Klatt [65]. The synthesis examples that accompanied that paper are available for listening at http://www.cs.indiana.edu/rhythmsp/ASA/Contents.html.

The third generation of speech synthesis technology was the period from 1992 to the present, in which the method of "unit selection synthesis" was introduced and perfected, primarily by Sagisaka at ATR Labs in Kyoto [108]. The resulting synthetic speech from this third generation technology had good intelligibility and naturalness that approached that of human-generated speech. We begin our discussion of speech synthesis approaches with a review of early systems, and then discuss unit selection methods of speech synthesis later in this chapter.

8.2.1 Early Speech Synthesis Approaches

Once the abstract underlying linguistic description of the text input has been determined via the steps of Figure 8.2, the remaining (major) task of TTS systems is to synthesize a speech waveform whose intelligibility is very high (to make the speech useful as a means of communication between a machine and a human), and whose naturalness is as close to real speech as possible. Both tasks, namely attaining high intelligibility along with reasonable naturalness, are difficult to achieve and depend critically on three issues in the processing of the speech synthesizer "backend" of Figure 8.1, namely:

(1) **choice of synthesis units:** including whole words, phones, diphones, dyads, or syllables
(2) **choice of synthesis parameters:** including LPC features, formants, waveform templates, articulatory parameters, sinusoidal parameters, etc.
(3) **method of computation:** including rule-based systems or systems which rely on the concatenation of stored speech units.

8.2.2 Word Concatenation Synthesis

Perhaps the simplest approach to creating a speech utterance corresponding to a given text string is to literally splice together prerecorded words corresponding to the desired utterance. For greatest simplicity, the words can be stored as sampled waveforms and simply concatenated in the correct sequence. This approach generally produces intelligible, but unnatural sounding speech, since it does not take into account the "co-articulation" effects of producing phonemes in continuous speech, and it does not provide either for the adjustment of phoneme durations or the imposition of a desired pitch variation across the utterance. Words spoken in continuous speech sentences are generally much shorter in duration than when spoken in isolation (often up to 50% shorter) as illustrated in Figure 8.5. This figure shows wideband spectrograms for the sentence "This shirt is red" spoken as a sequence of isolated words (with short, distinct pauses between words) as shown at the top of Figure 8.5, and as a continuous utterance, as shown at the bottom of Figure 8.5. It can be seen that, even for this trivial

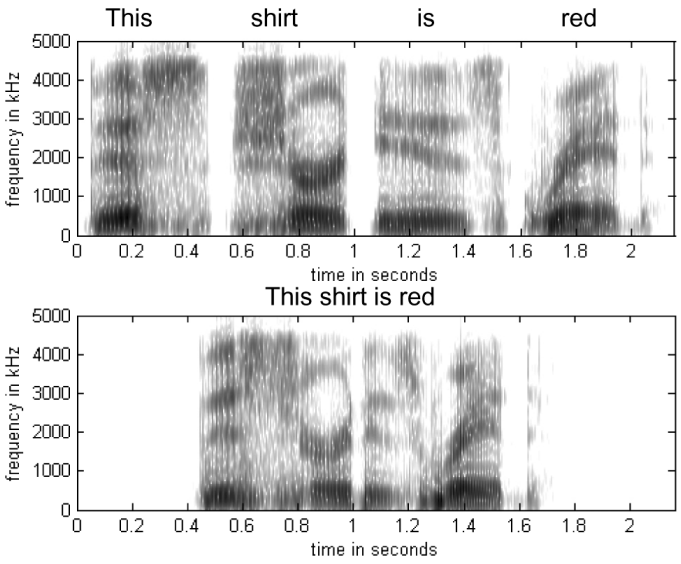

Fig. 8.5 Wideband spectrograms of a sentence spoken as a sequence of isolated words (top panel) and as a continuous speech utterance (bottom panel).

example, the duration of the continuous sentence is on the order of half that of the isolated word version, and further the formant tracks of the continuous sentence do not look like a set of uniformly compressed formant tracks from the individual words. Furthermore, the boundaries between words in the upper plot are sharply defined, while they are merged in the lower plot.

The word concatenation approach can be made more sophisticated by storing the vocabulary words in a parametric form (formants, LPC parameters, etc.) such as employed in the speech coders discussed in Chapter 7 [102]. The rational for this is that the parametric representations, being more closely related to the model for speech production, can be manipulated so as to blend the words together, shorten them, and impose a desired pitch variation. This requires that the control parameters for all the words in the task vocabulary (as obtained from a training set of words) be stored as representations of the words. A special set of word concatenation rules is then used to create the control signals for the synthesizer. Although such a synthesis system would appear to be an attractive alternative for general purpose synthesis of speech, in reality this type of synthesis is not a practical approach.[1]

There are many reasons for this, but a major problem is that there are far too many words to store in a word catalog for word concatenation synthesis to be practical except in highly restricted situations. For example, there are about 1.7 million distinct surnames in the United States and each of them would have to be spoken and stored for a general word concatenation synthesis method. A second, equally important limitation is that word-length segments of speech are simply the wrong sub-units. As we will see, shorter units such as phonemes or *diphones* are more suitable for synthesis.

Efforts to overcome the limitations of word concatenation followed two separate paths. One approach was based on controlling the motions of a physical model of the speech articulators based on the sequence of phonemes from the text analysis. This requires sophisticated control

[1] It should be obvious that whole word concatenation synthesis (from stored waveforms) is also impractical for general purpose synthesis of speech.

rules that are mostly derived empirically. From the vocal tract shapes and sources produced by the articulatory model, control parameters (e.g., formants and pitch) can be derived by applying the acoustic theory of speech production and then used to control a synthesizer such those used for speech coding. An alternative approach eschews the articulatory model and proceeds directly to the computation of the control signals for a source/system model (e.g., formant parameters, LPC parameters, pitch period, etc.). Again, the rules (algorithm) for computing the control parameters are mostly derived by empirical means.

8.2.3 Articulatory Methods of Synthesis

Articulatory models of human speech are based upon the application of acoustic theory to physical models such as depicted in highly stylized form in Figure 2.1 [22]. Such models were thought to be inherently more natural than vocal tract analog models since:

- we could impose known and fairly well understood physical constraints on articular movements to create realistic motions of the tongue, jaw, teeth, velum, etc.
- we could use X-ray data (MRI data today) to study the motion of the articulators in the production of individual speech sounds, thereby increasing our understanding of the dynamics of speech production
- we could model smooth articulatory parameter motions between sounds, either via direct methods (namely solving the wave equation), or indirectly by converting articulatory shapes to formants or LPC parameters
- we could highly constrain the motions of the articulatory parameters so that only natural motions would occur, thereby potentially making the speech more natural sounding.

What we have learned about articulatory modeling of speech is that it requires a highly accurate model of the vocal cords and of the vocal tract for the resulting synthetic speech quality to be considered acceptable. It further requires rules for handling the dynamics of the

articulator motion in the context of the sounds being produced. So far we have been unable to learn all such rules, and thus, articulatory speech synthesis methods have not been found to be practical for synthesizing speech of acceptable quality.

8.2.4 Terminal Analog Synthesis of Speech

The alternative to articulatory synthesis is called *terminal analog speech synthesis*. In this approach, each sound of the language (phoneme) is characterized by a source excitation function and an ideal vocal tract model. Speech is produced by varying (in time) the excitation and the vocal tract model control parameters at a rate commensurate with the sounds being produced. This synthesis process has been called *terminal analog synthesis* because it is based on a model (analog) of the human vocal tract production of speech that seeks to produce a signal at its output terminals that is equivalent to the signal produced by a human talker.[2]

The basis for terminal analog synthesis of speech is the source/system model of speech production that we have used many times in this text; namely an excitation source, $e[n]$, and a transfer function of the human vocal tract in the form of a rational system function, i.e.,

$$H(z) = \frac{S(z)}{E(z)} = \frac{B(z)}{A(z)} = \frac{b_0 + \sum_{k=1}^{q} b_k z^{-k}}{1 - \sum_{k=1}^{p} a_k z^{-k}}, \tag{8.1}$$

where $S(z)$ is the z-transform of the output speech signal, $s[n]$, $E(z)$ is the z-transform of the vocal tract excitation signal, $e[n]$, and $\{b_k\} = \{b_0, b_1, b_2, \ldots, b_q\}$ and $\{a_k\} = \{a_1, a_2, \ldots, a_p\}$ are the (time-varying) coefficients of the vocal tract filter. (See Figures 2.2 and 4.1.)

[2] In the designation "terminal analog synthesis", the terms *analog* and *terminal* result from the historical context of early speech synthesis studies. This can be confusing since today, "analog" implies "not digital" as well as an analogous thing. "Terminal" originally implied the "output terminals" of an electronic analog (not digital) circuit or system that was an analog of the human speech production system.

For most practical speech synthesis systems, both the excitation signal properties (P_0 and V/UV) and the filter coefficients of (8.1) change periodically so as to synthesize different phonemes.

The vocal tract representation of (8.1) can be implemented as a speech synthesis system using a direct form implementation. However, it has been shown that it is preferable to factor the numerator and denominator polynomials into either a series of cascade (serial) resonances, or into a parallel combination of resonances. It has also been shown that an all-pole model ($B(z) = \text{constant}$) is most appropriate for voiced (non-nasal) speech. For unvoiced speech a simpler model (with one complex pole and one complex zero), implemented via a parallel branch is adequate. Finally a fixed spectral compensation network can be used to model the combined effects of glottal pulse shape and radiation characteristics from the lips and mouth, based on two real poles in the z-plane. A complete serial terminal analog speech synthesis model, based on the above discussion, is shown in Figure 8.6 [95, 111].

The voiced speech (upper) branch includes an impulse generator (controlled by a time-varying pitch period, P_0), a time-varying voiced signal gain, A_V, and an all-pole discrete-time system that consists of a cascade of 3 time-varying resonances (the first three formants, F_1, F_2, F_3), and one fixed resonance F_4.

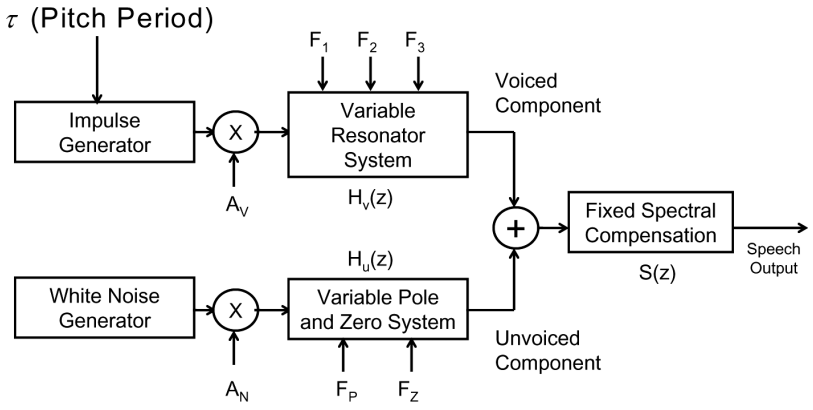

Fig. 8.6 Speech synthesizer based on a cascade/serial (formant) synthesis model.

The unvoiced speech (lower) branch includes a white noise generator, a time-varying unvoiced signal gain, A_N, and a resonance/antiresonance system consisting of a time-varying pole (F_P) and a time-varying zero (F_Z).

The voiced and unvoiced components are added and processed by the fixed spectral compensation network to provide the final synthetic speech output.

The resulting quality of the synthetic speech produced using a terminal analog synthesizer of the type shown in Figure 8.6 is highly variable with explicit model shortcomings due to the following:

- voiced fricatives are not handled properly since their mixed excitation is not part of the model of Figure 8.6
- nasal sounds are not handled properly since nasal zeros are not included in the model
- stop consonants are not handled properly since there is no precise timing and control of the complex excitation signal
- use of a fixed pitch pulse shape, independent of the pitch period, is inadequate and produces buzzy sounding voiced speech
- the spectral compensation model is inaccurate and does not work well for unvoiced sounds.

Many of the short comings of the model of Figure 8.6 are alleviated by a more complex model proposed by Klatt [64]. However, even with a more sophisticated synthesis model, it remains a very challenging task to compute the synthesis parameters. Nevertheless Klatt's Klattalk system achieved adequate quality by 1983 to justify commercialization by the Digital Equipment Corporation as the DECtalk system. Some DECtalk systems are still in operation today as legacy systems, although the unit selection methods to be discussed in Section 8.3 now provide superior quality in current applications.

8.3 Unit Selection Methods

The key idea of a concatenative TTS system, using unit selection methods, is to use synthesis segments that are sections of prerecorded

natural speech [31, 50, 108]. The word concatenation method discussed in Section 8.2.2 is perhaps the simplest embodiment of this idea; however, as we discussed, shorter segments are required to achieve better quality synthesis. The basic idea is that the more segments recorded, annotated, and saved in the database, the better the potential quality of the resulting speech synthesis. Ultimately, if an infinite number of segments were recorded and saved, the resulting synthetic speech would sound natural for virtually all possible synthesis tasks. Concatenative speech synthesis systems, based on unit selection methods, are what is conventionally known as "data driven" approaches since their performance tends to get better the more data that is used for training the system and selecting appropriate units.

In order to design and build a unit selection system based on using recorded speech segments, several issues have to be resolved, including the following:

(1) What speech units should be used as the basic synthesis building blocks?
(2) How the synthesis units are selected (extracted) from natural speech utterances?
(3) How the units are labeled for retrieval from a large database of units?
(4) What signal representation should be used to represent the units for storage and reproduction purposes?
(5) What signal processing methods can be used to spectrally smooth the units (at unit junctures) and for prosody modification (pitch, duration, amplitude)?

We now attempt to answer each of these questions.

8.3.1 Choice of Concatenation Units

The units for unit selection can (in theory) be as large as words and as small as phoneme units. Words are prohibitive since there are essentially an infinite number of words in English. Subword units include syllables (about 10,000 in English), phonemes (about 45 in English, but they are highly context dependent), demi-syllables (about 2500

in English), and diphones (about 1500–2500 in English). The ideal synthesis unit is context independent and easily concatenates with other (appropriate) subword units [15]. Based on this criterion, the most reasonable choice for unit selection synthesis is the set of diphone units.[3]

Before any synthesis can be done, it is necessary to prepare an inventory of units (diphones). This requires significant effort if done manually. In any case, a large corpus of speech must be obtained from which to extract the diphone units as waveform snippets. These units are then represented in some compressed form for efficient storage. High quality coding such as MPLP or CELP is used to limit compression artifacts. At the final synthesis stage, the diphones are decoded into waveforms for final merging, duration adjustment and pitch modification, all of which take place on the time waveform.

8.3.2 From Text to Diphones

The synthesis procedure from subword units is straightforward, but far from trivial [14]. Following the process of text analysis into phonemes and prosody, the phoneme sequence is first converted to the appropriate sequence of units from the inventory. For example, the phrase "I want" would be converted to diphone units as follows:

Text Input: I want.
Phonemes: /#/ AY/ /W/ /AA/ /N/ /T/ /#/
Diphones: /# AY/ /AY-W/ /W-AA/ /AA-N/ /N-T/ /T- #/,

where the symbol # is used to represent silence (at the beginning and end of each sentence or phrase).

The second step in online synthesis is to select the most appropriate sequence of diphone units from the stored inventory. Since each diphone unit occurs many times in the stored inventory, the selection

[3] A diphone is simply the concatenation of two phonemes that are allowed by the language constraints to be sequentially contiguous in natural speech. For example, a diphone based upon the phonemes AY and W would be denoted AY−W where the − denotes the joining of the two phonemes.

of the best sequence of diphone units involves solving a dynamic programming search for the sequence of units that minimizes a specified cost function. The cost function generally is based on diphone matches at each of the boundaries between diphones, where the diphone match is defined in terms of spectral matching characteristics, pitch matching characteristics, and possibly phase matching characteristics.

8.3.3 Unit Selection Synthesis

The "Unit Selection" problem is basically one of having a given set of target features corresponding to the spoken text, and then automatically finding the sequence of units in the database that most closely match these features. This problem is illustrated in Figure 8.7 which shows a target feature set corresponding to the sequence of sounds (phonemes for this trivial example) /HH/ /EH/ /L/ /OW/ from the word /hello/.[4] As shown, each of the phonemes in this word have multiple representations (units) in the inventory of sounds, having been extracted from different phonemic environments. Hence there are many versions of /HH/ and many versions of /EH/ etc., as illustrated in Figure 8.7. The task of the unit selection module is to choose one of each of the multiple representations of the sounds, with the goal being to minimize the total perceptual distance between segments of the chosen sequence, based on spectral, pitch and phase differences throughout the sounds and especially at the boundary between sounds. By specifying costs associated with each unit, both globally across the unit,

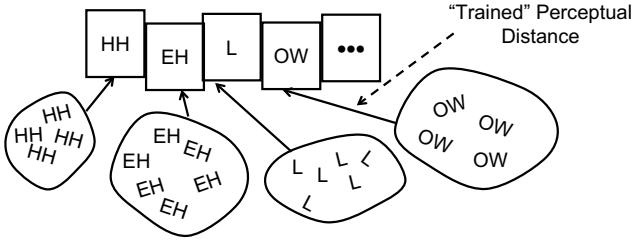

Fig. 8.7 Illustration of basic process of unit selection. (After Dutoit [31].)

[4] If diphones were used, the sequence would be /HH-EH/ /EH-L/ /L-OW/.

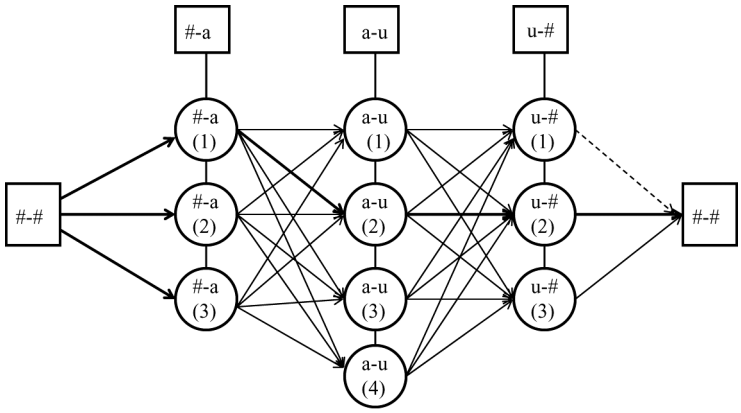

Fig. 8.8 Online unit selection based on a Viterbi search through a lattice of alternatives.

and locally at the unit boundaries with adjacent units, we can find the sequence of units that best "join each other" in the sense of minimizing the accumulated distance across the sequence of units. This optimal sequence (or equivalently the optimal path through the combination of all possible versions of each unit in the string) can be found using a Viterbi search (dynamic programming) [38, 99]. This dynamic programming search process is illustrated in Figure 8.8 for a 3 unit search. The Viterbi search effectively computes the cost of every possible path through the lattice and determines the path with the lowest total cost, where the transitional costs (the arcs) reflect the cost of concatenation of a pair of units based on acoustic distances, and the nodes represent the target costs based on the linguistic identity of the unit.

Thus there are two costs associated with the Viterbi search, namely a nodal cost based on the unit segmental distortion (USD) which is defined as the difference between the desired spectral pattern of the target (suitably defined) and that of the candidate unit, throughout the unit, and the unit concatenative distortion (UCD) which is defined as the spectral (and/or pitch and/or phase) discontinuity across the boundaries of the concatenated units. By way of example, consider a target context of the word "cart" with target phonemes /K/ /AH/ /R/ /T/, and a source context of the phoneme /AH/ obtained from the

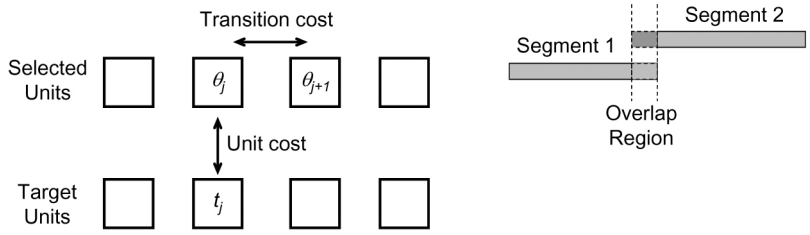

Fig. 8.9 Illustration of unit selection costs associated with a string of target units and a presumptive string of selected units.

source word "want" with source phonemes /W/ /AH/ /N/ /T/. The USD distance would be the cost (specified analytically) between the sound /AH/ in the context /W/ /AH/ /N/ versus the desired sound in the context /K/ /AH/ /R/.

Figure 8.9 illustrates concatenative synthesis for a given string of target units (at the bottom of the figure) and a string of selected units from the unit inventory (shown at the top of the figure). We focus our attention on Target Unit t_j and Selected Units θ_j and θ_{j+1}. Associated with the match between units t_j and θ_j is a USD Unit Cost and associated with the sequence of selected units, θ_j and θ_{j+1}, is a UCD concatenation cost. The total cost of an arbitrary string of N selected units, $\Theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$, and the string of N target units, $T = \{t_1, t_2, \ldots, t_N\}$ is defined as:

$$d(\Theta, T) = \sum_{j=1}^{N} d_u(\theta_j, t_j) + \sum_{j=1}^{N-1} d_t(\theta_j, \theta_{j+1}), \tag{8.2}$$

where $d_u(\theta_j, t_j)$ is the USD cost associated with matching target unit t_j with selected unit θ_j and $d_t(\theta_j, \theta_{j+1})$ is the UCD cost associated with concatenating the units θ_j and θ_{j+1}. It should be noted that when selected units θ_j and θ_{j+1} come from the same source sentence and are adjacent units, the UCD cost goes to zero, as this is as natural a concatenation as can exist in the database. Further, it is noted that generally there is a small overlap region between concatenated units. The optimal path (corresponding to the optimal sequence of units) can be efficiently computed using a standard Viterbi search ([38, 99]) in

which the computational demand scales linearly with both the number of target and the number of concatenation units.

The concatenation cost between two units is essentially the spectral (and/or phase and/or pitch) discontinuity across the boundary between the units, and is defined as:

$$d_t(\theta_j, \theta_{j+1}) = \sum_{k=1}^{p+2} w_k C_k(\theta_j, \theta_{j+1}), \tag{8.3}$$

where p is the size of the spectral feature vector (typically $p = 12$ mfcc coefficients (mel-frequency cepstral coefficients as explained in Section 5.6.3), often represented as a VQ codebook vector), and the extra two features are log power and pitch. The weights, w_k are chosen during the unit selection inventory creation phase and are optimized using a trial-and-error procedure. The concatenation cost essentially measures a spectral plus log energy plus pitch difference between the two concatenated units at the boundary frames. Clearly the definition of concatenation cost could be extended to more than a single boundary frame. Also, as stated earlier, the concatenation cost is defined to be zero ($d_t(\theta_j, \theta_{j+1}) = 0$) whenever units θ_j and θ_{j+1} are consecutive (in the database) since, by definition, there is no discontinuity in either spectrum or pitch in this case. Although there are a variety of choices for measuring the spectral/log energy/pitch discontinuity at the boundary, a common cost function is the normalized mean-squared error in the feature parameters, namely:

$$C_k(\theta_j, \theta_{j+1}) = \frac{[f_k^{\theta_j}(m) - f_k^{\theta_{j+1}}(1)]^2}{\sigma_k^2}, \tag{8.4}$$

where $f_k^{\theta_j}(l)$ is the kth feature parameter of the lth frame of segment θ_j, m is the (normalized) duration of each segment, and σ_k^2 is the variance of the kth feature vector component.

The USD or target costs are conceptually more difficult to understand, and, in practice, more difficult to instantiate. The USD cost associated with units θ_j and t_j is of the form:

$$d_u(\theta_j, t_j) = \sum_{i=1}^{q} w_i^t \phi_i \left\{ T_i(f_i^{\theta_j}), T_i(f_i^{t_j}) \right\}, \tag{8.5}$$

where q is the number of features that specify the unit θ_j or t_j, $w_i^t, i = 1, 2, \ldots, q$ is a trained set of target weights, and $T_i(\cdot)$ can be either a continuous function (for a set of features, f_i, such as segmental pitch, power or duration), or a set of integers (in the case of categorical features, f_i, such as unit identity, phonetic class, position in the syllable from which the unit was extracted, etc.). In the latter case, ϕ_i can be looked up in a table of distances. Otherwise, the local distance function can be expressed as a simple quadratic distance of the form

$$\phi_i \left\{ T_i(f_i^{\theta_j}), T_i(f_i^{t_j}) \right\} = \left[T_i(f_i^{\theta_j}) - T_i(f_i^{t_j}) \right]^2. \tag{8.6}$$

The training of the weights, w_i^t, is done off-line. For each phoneme in each phonetic class in the training speech database (which might be the entire recorded inventory), each exemplar of each unit is treated as a target and all others are treated as candidate units. Using this training set, a least-squares system of linear equations can be derived from which the weight vector can be solved. The details of the weight training methods are described by Schroeter [114].

The final step in the unit selection synthesis, having chosen the optimal sequence of units to match the target sentence, is to smooth/modify the selected units at each of the boundary frames to better match the spectra, pitch and phase at each unit boundary. Various smoothing methods based on the concepts of time domain harmonic scaling (TDHS) [76] and pitch synchronous overlap add (PSOLA) [20, 31, 82] have been proposed and optimized for such smoothing/modification at the boundaries between adjacent diphone units.

8.4 TTS Applications

Speech technology serves as a way of intelligently and efficiently enabling humans to interact with machines with the goal of bringing down cost of service for an existing service capability, or providing new products and services that would be prohibitively expensive without the automation provided by a viable speech processing interactive

system. Examples of existing services where TTS enables significant cost reduction are as follows:

- a dialog component for customer care applications
- a means for delivering text messages over an audio connection (e.g., cellphone)
- a means of replacing expensive recorded Interactive Voice Response system prompts (a service which is particularly valuable when the prompts change often during the course of the day, e.g., stock price quotations).

Similarly some examples of new products and services which are enabled by a viable TTS technology are the following:

- location-based services, e.g., alerting you to locations of restaurants, gas stations, stores in the vicinity of your current location
- providing information in cars (e.g., driving directions, traffic reports)
- unified messaging (e.g., reading e-mail and fax messages)
- voice portal providing voice access to web-based services
- e-commerce agents
- customized News, Stock Reports, Sports scores, etc.
- giving voice to small and embedded devices for reporting information and alerts.

8.5 TTS Future Needs

Modern TTS systems have the capability of producing highly intelligible, and surprisingly natural speech utterances, so long as the utterance is not too long, or too syntactically complicated, or too technical. The biggest problem with most TTS systems is that they have no idea as to *how* things should be said, but instead rely on the text analysis for emphasis, prosody, phrasing and all so-called suprasegmental features of the spoken utterance. The more TTS systems learn how to produce context-sensitive pronunciations of words (and phrases), the more natural sounding these systems will become. Hence, by way of example,

the utterance "I gave the book to John" has at least three different semantic interpretations, each with different emphasis on words in the utterance, i.e.,

> I gave the book to **John**, i.e., not to Mary or Bob.
> I gave the **book** to John, i.e., not the photos or the apple.
> **I** gave the book to John, i.e., I did it, not someone else.

The second future need of TTS systems is improvements in the unit selection process so as to better capture the target cost for mismatch between *predicted* unit specification (i.e., phoneme name, duration, pitch, spectral properties) and *actual* features of a candidate recorded unit. Also needed in the unit selection process is a better spectral distance measure that incorporates human perception measures so as to find the best sequence of units for a given utterance.

Finally, better signal processing would enable improved compression of the units database, thereby making the footprint of TTS systems small enough to be usable in handheld and mobile devices.

9

Automatic Speech Recognition (ASR)

In this chapter, we examine the process of speech recognition by machine, which is in essence the inverse of the text-to-speech problem. The driving factor behind research in machine recognition of speech has been the potentially huge payoff of providing services where humans interact solely with machines, thereby eliminating the cost of live agents and significantly reducing the cost of providing services. Interestingly, as a side benefit, this process often provides users with a natural and convenient way of accessing information and services.

9.1 The Problem of Automatic Speech Recognition

The goal of an ASR system is to accurately and efficiently convert a speech signal into a text message transcription of the spoken words, independent of the device used to record the speech (i.e., the transducer or microphone), the speaker's accent, or the acoustic environment in which the speaker is located (e.g., quiet office, noisy room, outdoors). That is, the ultimate goal, which has not yet been achieved, is to perform as well as a human listener.

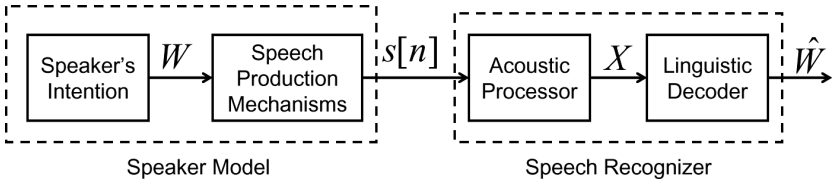

Fig. 9.1 Conceptual model of speech production and speech recognition processes.

A simple conceptual model of the speech generation and speech recognition processes is given in Figure 9.1, which is a simplified version of the speech chain shown in Figure 1.2. It is assumed that the speaker intends to express some thought as part of a process of conversing with another human or with a machine. To express that thought, the speaker must compose a linguistically meaningful sentence, W, in the form of a sequence of words (possibly with pauses and other acoustic events such as uh's, um's, er's etc.). Once the words are chosen, the speaker sends appropriate control signals to the articulatory speech organs which form a speech utterance whose sounds are those required to speak the desired sentence, resulting in the speech waveform $s[n]$. We refer to the process of creating the speech waveform from the speaker's intention as the *Speaker Model* since it reflects the speaker's accent and choice of words to express a given thought or request. The processing steps of the Speech Recognizer are shown at the right side of Figure 9.1 and consist of an acoustic processor which analyzes the speech signal and converts it into a set of acoustic (spectral, temporal) features, X, which efficiently characterize the speech sounds, followed by a linguistic decoding process which makes a best (maximum likelihood) estimate of the words of the spoken sentence, resulting in the recognized sentence $\hat{W}$. This is in essence a digital simulation of the lower part of the speech chain diagram in Figure 1.2.

Figure 9.2 shows a more detailed block diagram of the overall speech recognition system. The input speech signal, $s[n]$, is converted to the sequence of feature vectors, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, by the feature analysis block (also denoted spectral analysis). The feature vectors are computed on a frame-by-frame basis using the techniques discussed in the earlier chapters. In particular, the mel frequency cepstrum coefficients

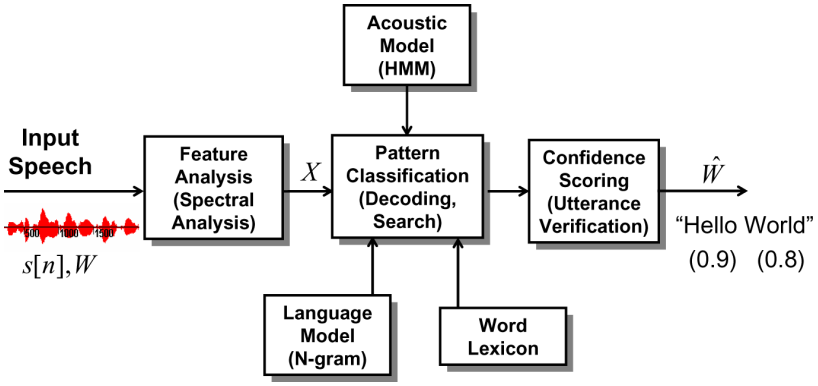

Fig. 9.2 Block diagram of an overall speech recognition system.

are widely used to represent the short-time spectral characteristics. The pattern classification block (also denoted as the decoding and search block) decodes the sequence of feature vectors into a symbolic representation that is the maximum likelihood string, $\hat{W}$ that could have produced the input sequence of feature vectors. The pattern recognition system uses a set of acoustic models (represented as hidden Markov models) and a word lexicon to provide the acoustic match score for each proposed string. Also, an N-gram language model is used to compute a language model score for each proposed word string. The final block in the process is a confidence scoring process (also denoted as an utterance verification block), which is used to provide a confidence score for each individual word in the recognized string. Each of the operations in Figure 9.2 involves many details and, in some cases, extensive digital computation. The remainder of this chapter is an attempt to give the flavor of what is involved in each part of Figure 9.2.

9.2 Building a Speech Recognition System

The steps in building and evaluating a speech recognition system are the following:

(1) choose the feature set and the associated signal processing for representing the properties of the speech signal over time

(2) choose the recognition task, including the recognition word vocabulary (the lexicon), the basic speech sounds to represent the vocabulary (the speech units), the task syntax or language model, and the task semantics (if any)
(3) train the set of speech acoustic and language models
(4) evaluate performance of the resulting speech recognition system.

Each of these steps may involve many choices and can involve significant research and development effort. Some of the important issues are summarized in this section.

9.2.1 Recognition Feature Set

There is no "standard" set of features for speech recognition. Instead, various combinations of acoustic, articulatory, and auditory features have been utilized in a range of speech recognition systems. The most popular acoustic features have been the (LPC-derived) mel-frequency cepstrum coefficients and their derivatives.

A block diagram of the signal processing used in most modern large vocabulary speech recognition systems is shown in Figure 9.3. The analog speech signal is sampled and quantized at rates between 8000 up to 20,000 samples/s. A first order (highpass) pre-emphasis network $(1 - \alpha z^{-1})$ is used to compensate for the speech spectral falloff at higher frequencies and approximates the inverse to the mouth transmission frequency response. The pre-emphasized signal is next blocked into frames of N samples, with adjacent frames spaced M samples apart. Typical values for N and M correspond to frames of duration 15–40 ms, with frame shifts of 10 ms being most common; hence adjacent frames overlap by 5–30 ms depending on the chosen values of N and M. A Hamming window is applied to each frame prior to spectral analysis using either standard spectral analysis or LPC methods. Following (optionally used) simple noise removal methods, the spectral coefficients are normalized and converted to mel-frequency cepstral coefficients via standard analysis methods of the type discussed in Chapter 5 [27]. Some type of cepstral bias removal is often used prior to calculation of the first and second order cepstral derivatives.

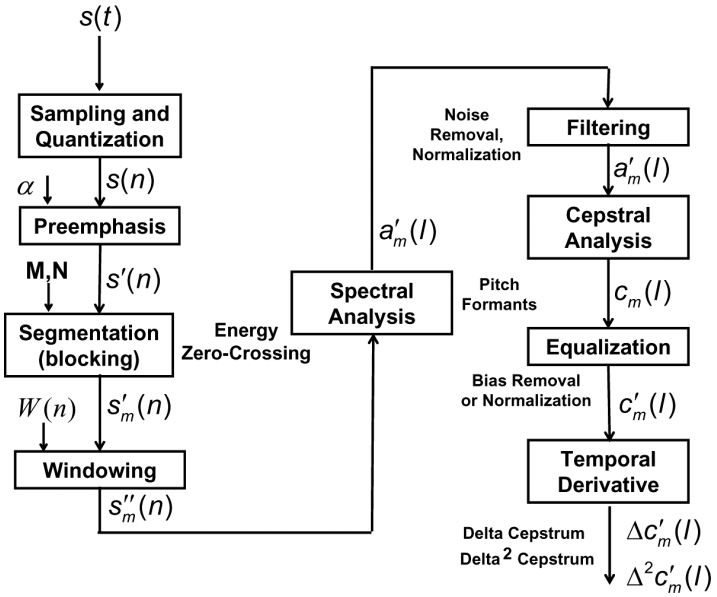

Fig. 9.3 Block diagram of feature extraction process for feature vector consisting of mfcc coefficients and their first and second derivatives.

Typically the resulting feature vector is the set of cepstral coefficients, and their first- and second-order derivatives. It is typical to use about 13 mfcc coefficients, 13 first-order cepstral derivative coefficients, and 13 second-order derivative cepstral coefficients, making a $D = 39$ size feature vector.

9.2.2 The Recognition Task

Recognition tasks vary from simple word and phrase recognition systems to large vocabulary conversational interfaces to machines. For example, using a digits vocabulary, the task could be recognition of a string of digits that forms a telephone number, or identification code, or a highly constrained sequence of digits that form a password.

9.2.3 Recognition Training

There are two aspects to training models for speech recognition, namely acoustic model training and language model training. Acoustic model

training requires recording each of the model units (whole words, phonemes) in as many contexts as possible so that the statistical learning method can create accurate distributions for each of the model states. Acoustic training relies on accurately labeled sequences of speech utterances which are segmented according to the transcription, so that training of the acoustic models first involves segmenting the spoken strings into recognition model units (via either a Baum–Welch [10, 11] or Viterbi alignment method), and then using the segmented utterances to simultaneously build model distributions for each state of the vocabulary unit models. The resulting statistical models form the basis for the pattern recognition operations at the heart of the ASR system. As discussed in Section 9.3, concepts such as Viterbi search are employed in the pattern recognition process as well as in training.

Language model training requires a sequence of text strings that reflect the syntax of spoken utterances for the task at hand. Generally such text training sets are created automatically (based on a model of grammar for the recognition task) or by using existing text sources, such as magazine and newspaper articles, or closed caption transcripts of television news broadcasts, etc. Other times, training sets for language models can be created from databases, e.g., valid strings of telephone numbers can be created from existing telephone directories.

9.2.4 Testing and Performance Evaluation

In order to improve the performance of any speech recognition system, there must be a reliable and statistically significant way of evaluating recognition system performance based on an independent test set of labeled utterances. Typically we measure word error rate and sentence (or task) error rate as a measure of recognizer performance. A brief summary of performance evaluations across a range of ASR applications is given in Section 9.4.

9.3 The Decision Processes in ASR

The heart of any automatic speech recognition system is the pattern classification and decision operations. In this section, we shall give a brief introduction to these important topics.

9.3.1 Mathematical Formulation of the ASR Problem

The problem of automatic speech recognition is represented as a statistical decision problem. Specifically it is formulated as a Bayes maximum *a posteriori* probability (MAP) decision process where we seek to find the word string $\hat{W}$ (in the task language) that maximizes the *a posteriori* probability $P(W|X)$ of that string, given the measured feature vector, X, i.e.,

$$\hat{W} = \arg\max_{W} P(W|X). \tag{9.1}$$

Using Bayes' rule we can rewrite (9.1) in the form:

$$\hat{W} = \arg\max_{W} \frac{P(X|W)P(W)}{P(X)}. \tag{9.2}$$

Equation (9.2) shows that the calculation of the *a posteriori* probability is decomposed into two terms, one that defines the *a priori* probability of the word sequence, W, namely $P(W)$, and the other that defines the likelihood that the word string, W, produced the feature vector, X, namely $P(X|W)$. For all future calculations we disregard the denominator term, $P(X)$, since it is independent of the word sequence W which is being optimized. The term $P(X|W)$ is known as the "acoustic model" and is generally denoted as $P_A(X|W)$ to emphasize the acoustic nature of this term. The term $P(W)$ is known as the "language model" and is generally denoted as $P_L(W)$ to emphasize the linguistic nature of this term. The probabilities associated with $P_A(X|W)$ and $P_L(W)$ are estimated or learned from a set of training data that have been labeled by a knowledge source, usually a human expert, where the training set is as large as reasonably possible. The recognition decoding process of (9.2) is often written in the form of a 3-step process, i.e.,

$$\hat{W} = \underbrace{\arg\max_{W}}_{\text{Step 3}} \underbrace{P_A(X|W)}_{\text{Step 1}} \underbrace{P_L(W)}_{\text{Step 2}}, \tag{9.3}$$

where Step 1 is the computation of the probability associated with the acoustic model of the speech sounds in the sentence W, Step 2 is the computation of the probability associated with the linguistic model of the words in the utterance, and Step 3 is the computation associated

with the search through all valid sentences in the task language for the maximum likelihood sentence.

In order to be more explicit about the signal processing and computations associated with each of the three steps of (9.3) we need to more explicit about the relationship between the feature vector, X, and the word sequence W. As discussed above, the feature vector, X, is a sequence of acoustic observations corresponding to each of T frames of the speech, of the form:

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}, \tag{9.4}$$

where the speech signal duration is T frames (i.e., T times the frame shift in ms) and each frame, $\mathbf{x}_t, t = 1, 2, \ldots, T$ is an acoustic feature vector of the form:

$$\mathbf{x}_t = (x_{t1}, x_{t2}, \ldots, x_{tD}) \tag{9.5}$$

that characterizes the spectral/temporal properties of the speech signal at time t and D is the number of acoustic features in each frame. Similarly we can express the optimally decoded word sequence, W, as:

$$W = w_1, w_2, \ldots, w_M, \tag{9.6}$$

where there are assumed to be exactly M words in the decoded string.

9.3.2 The Hidden Markov Model

The most widely used method of building acoustic models (for both phonemes and words) is the use of a statistical characterization known as Hidden Markov Models (HMMs) [33, 69, 97, 98]. Figure 9.4 shows a simple $Q = 5$-state HMM for modeling a whole word. Each HMM state is characterized by a mixture density Gaussian distribution that characterizes the statistical behavior of the feature vectors within the states of the model [61, 62]. In addition to the statistical feature densities within states, the HMM is also characterized by an explicit set of state transitions, a_{ij}, which specify the probability of making a transition from state i to state j at each frame, thereby defining the time sequence of the feature vectors over the duration of the word. Usually the self-transitions, a_{ii} are large (close to 1.0), and the jump transitions, $a_{12}, a_{23}, a_{34}, a_{45}$, in the model, are small (close to 0).

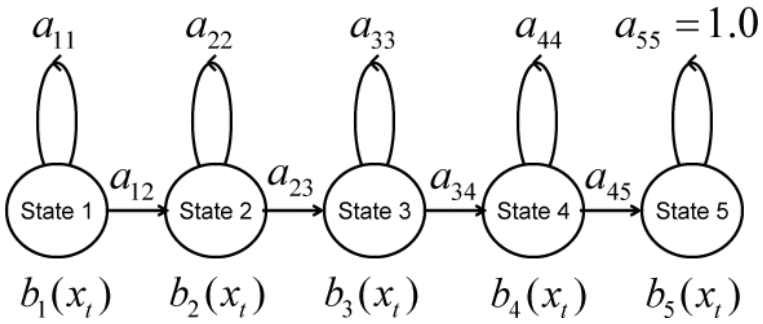

Fig. 9.4 Word-based, left-to-right, HMM with 5 states.

The complete HMM characterization of a Q-state word model (or a sub-word unit like a phoneme model) is generally written as $\lambda(A, B, \pi)$ with state transition matrix $A = \{a_{ij}, 1 \leq i, j \leq Q\}$, state observation probability density, $B = \{b_j(\mathbf{x}_t), 1 \leq j \leq Q\}$, and initial state distribution, $\pi = \{\pi_i, 1 \leq i \leq Q\}$ with π_1 set to 1 for the "left-to-right" models of the type shown in Figure 9.4.

In order to train the HMM (i.e., learn the optimal model parameters) for each word (or sub-word) unit, a labeled training set of sentences (transcribed into words and sub-word units) is used to guide an efficient training procedure known as the Baum–Welch algorithm [10, 11].[1] This algorithm aligns each of the various words (or sub-word units) with the spoken inputs and then estimates the appropriate means, covariances and mixture gains for the distributions in each model state. The Baum–Welch method is a hill-climbing algorithm and is iterated until a stable alignment of models and speech is obtained. The details of the Baum–Welch procedure are beyond the scope of this chapter but can be found in several references on Speech Recognition methods [49, 99]. The heart of the training procedure for re-estimating HMM model parameters using the Baum–Welch procedure is shown in Figure 9.5. An initial HMM model is used to begin the training process. The initial model can be randomly chosen or selected based on *a priori* knowledge of the model

[1] The Baum–Welch algorithm is also widely referred to as the forward–backward method.

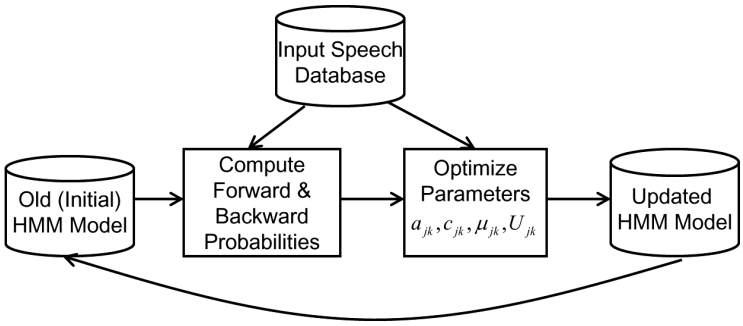

Fig. 9.5 The Baum–Welch training procedure based on a given training set of utterances.

parameters. The iteration loop is a simple updating procedure for computing the forward and backward model probabilities based on an input speech database (the training set of utterances) and then optimizing the model parameters to give an updated HMM. This process is iterated until no further improvement in probabilities occurs with each new iteration.

It is a simple matter to go from the HMM for a whole word, as shown in Figure 9.4, to an HMM for a sub-word unit (such as a phoneme) as shown in Figure 9.6. This simple 3-state HMM is a basic sub-word unit model with an initial state representing the statistical characteristics at the beginning of a sound, a middle state representing the heart of the sound, and an ending state representing the spectral characteristics at the end of the sound. A word model is made by concatenating the appropriate sub-word HMMs, as illustrated in

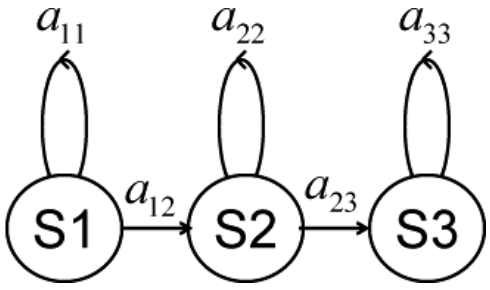

Fig. 9.6 Sub-word-based HMM with 3 states.

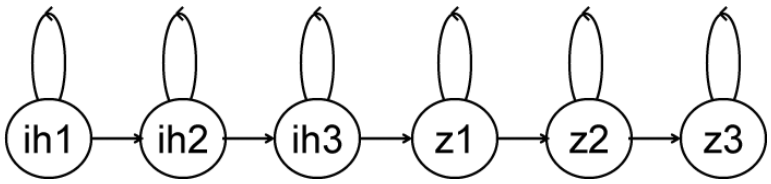

Fig. 9.7 Word-based HMM for the word /is/ created by concatenating 3-state subword models for the sub-word units /ih/ and /z/.

Figure 9.7, which concatenates the 3-state HMM for the sound /IH/ with the 3-state HMM for the sound /Z/, giving the word model for the word "is" (pronounced as /IH Z/). In general the composition of a word (from sub-word units) is specified in a word lexicon or dictionary; however once the word model has been built it can be used much the same as whole word models for training and for evaluating word strings for maximizing the likelihood as part of the speech recognition process.

We are now ready to define the procedure for aligning a sequence of M word models, $w_1, w_w, \ldots, w_M$ with a sequence of feature vectors, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$. The resulting alignment procedure is illustrated in Figure 9.8. We see the sequence of feature vectors along the horizontal axis and the concatenated sequence of word states along the vertical axis. An optimal alignment procedure determines the exact best matching sequence between word model states and feature vectors such that the first feature vector, $\mathbf{x}_1$, aligns with the first state in the first word model, and the last feature vector, $\mathbf{x}_T$, aligns with the last state in the Mth word model. (For simplicity we show each word model as a 5-state HMM in Figure 9.8, but clearly the alignment procedure works for any size model for any word, subject to the constraint that the total number of feature vectors, T, exceeds the total number of model states, so that every state has at least a single feature vector associated with that state.) The procedure for obtaining the best alignment between feature vectors and model states is based on either using the Baum–Welch statistical alignment procedure (in which we evaluate the probability of every alignment path and add them up to determine the probability of the word string), or a Viterbi alignment procedure [38, 132] for which

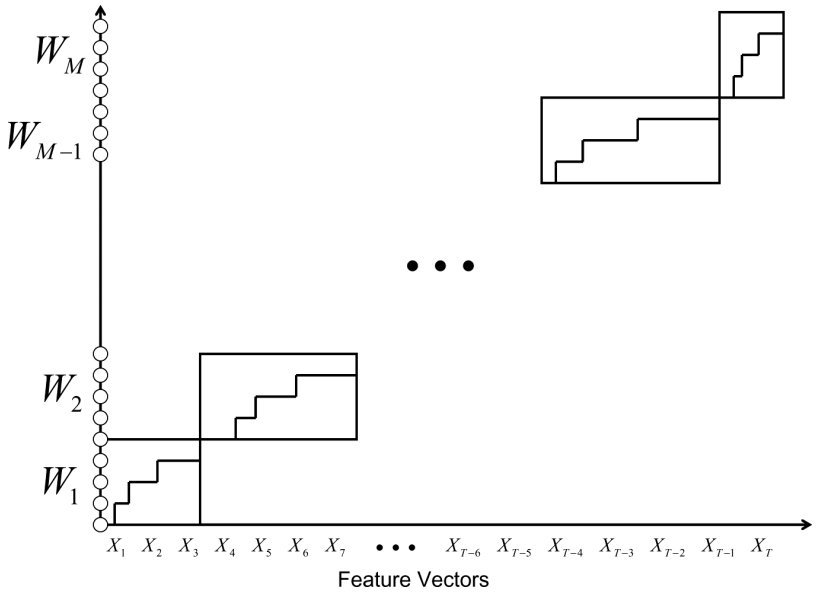

Fig. 9.8 Alignment of concatenated HMM word models with acoustic feature vectors based on either a Baum–Welch or Viterbi alignment procedure.

we determine the single best alignment path and use the probability score along that path as the probability measure for the current word string. The utility of the alignment procedure of Figure 9.8 is based on the ease of evaluating the probability of any alignment path using the Baum–Welch or Viterbi procedures.

We now return to the mathematical formulation of the ASR problem and examine in more detail the three steps in the decoding Equation (9.3).

9.3.3 Step 1 — Acoustic Modeling

The function of the acoustic modeling step (Step 1) is to assign probabilities to the acoustic realizations of a sequence of words, given the observed acoustic vectors, i.e., we need to compute the probability that the acoustic vector sequence $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$ came from the word sequence $W = w_1, w_2, \ldots, w_M$ (assuming each word is represented as an HMM) and perform this computation for all possible word sequences.

This calculation can be expressed as:

$$P_A(X|W) = P_A(\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}|w_1, w_2, \ldots, w_M). \tag{9.7}$$

If we make the assumption that each frame, x_t, is aligned with word $i(t)$ and HMM model state $j(t)$ via the function $w_{j(t)}^{i(t)}$ and if we assume that each frame is independent of every other frame, we can express (9.7) as the product

$$P_A(X|W) = \prod_{t=1}^{T} P_A\Big(\mathbf{x}_t|w_{j(t)}^{i(t)}\Big), \tag{9.8}$$

where we associate each frame of X with a unique word and state, $w_{j(t)}^{i(t)}$, in the word sequence. Further, we calculate the local probability $P_A\big(\mathbf{x}_t|w_{j(t)}^{i(t)}\big)$ given that we know the word from which frame t came.

The process of assigning individual speech frames to the appropriate word model in an utterance is based on an optimal alignment process between the concatenated sequence of word models and the sequence of feature vectors of the spoken input utterance being recognized. This alignment process is illustrated in Figure 9.9 which shows the set of T feature vectors (frames) along the horizontal axis, and the set of M words (and word model states) along the vertical axis. The optimal segmentation of these feature vectors (frames) into the M words is shown by the sequence of boxes, each of which corresponds to one of the words in the utterance and its set of optimally matching feature vectors.

We assume that each word model is further decomposed into a set of states which reflect the changing statistical properties of the feature vectors over time for the duration of the word. We assume that each word is represented by an N-state HMM model, and we denote the states as $S_j, j = 1, 2, \ldots, N$. Within each state of each word there is a probability density that characterizes the statistical properties of the feature vectors in that state. We have seen in the previous section that the probability density of each state, and for each word, is learned during a training phase of the recognizer. Using a mixture of Gaussian densities to characterize the statistical distribution of the feature vectors in each state, j, of the word model, which we denote as $b_j(\mathbf{x}_t)$,

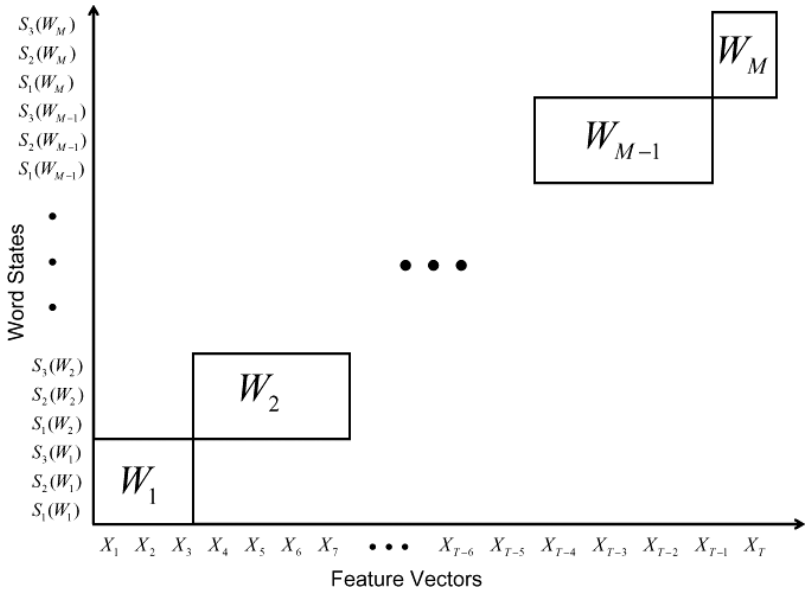

Fig. 9.9 Illustration of time alignment process between unknown utterance feature vectors and set of M concatenated word models.

we get a state-based probability density of the form:

$$b_j(\mathbf{x}_t) = \sum_{k=1}^{K} c_{jk} \mathbb{N}[\mathbf{x}_t, \mu_{jk}, U_{jk}], \tag{9.9}$$

where K is the number of mixture components in the density function, c_{jk} is the weight of the kth mixture component in state j, with the constraint $c_{jk} \geq 0$, and $\mathbb{N}$ is a Gaussian density function with mean vector, μ_{jk}, for mixture k for state j, and covariance matrix, U_{jk}, for mixture k in state j. The density constraints are:

$$\sum_{k=1}^{K} c_{jk} = 1, \quad 1 \leq j \leq N \tag{9.10}$$

$$\int_{-\infty}^{\infty} b_j(\mathbf{x}_t) d\mathbf{x}_t = 1, \quad 1 \leq j \leq N. \tag{9.11}$$

We now return to the issue of the calculation of the probability of frame x_t being associated with the $j(t)$th state of the $i(t)$th word in

the utterance, $P_A(\mathbf{x}_t|w_{j(t)}^{i(t)})$, which is calculated as

$$P_A\Big(\mathbf{x}_t|w_{j(t)}^{i(t)}\Big) = b_{j(t)}^{i(t)}(\mathbf{x}_t), \tag{9.12}$$

The computation of (9.12) is incomplete since we have ignored the computation of the probability associated with the links between word states, and we have also not specified how to determine the within-word state, j, in the alignment between a given word and a set of feature vectors corresponding to that word. We come back to these issues later in this section.

The key point is that we assign probabilities to acoustic realizations of a sequence of words by using hidden Markov models of the acoustic feature vectors within words. Using an independent (and orthographically labeled) set of training data, we "train the system" and learn the parameters of the best acoustic models for each word (or more specifically for each sound that comprises each word). The parameters, according to the mixture model of (9.9) are, for each state of the model, the mixture weights, the mean vectors, and the covariance matrix.

Although we have been discussing acoustic models for whole words, it should be clear that for any reasonable size speech recognition task, it is impractical to create a separate acoustic model for every possible word in the vocabulary since each word would have to be spoken in every possible context in order to build a statistically reliable model of the density functions of (9.9). Even for modest size vocabularies of about 1000 words, the amount of training data required for word models is excessive.

The alternative to word models is to build acoustic-phonetic models for the 40 or so phonemes in the English language and construct the model for a word by concatenating (stringing together sequentially) the models for the constituent phones in the word (as represented in a word dictionary or lexicon). The use of such sub-word acoustic-phonetic models poses no real difficulties in either training or when used to build up word models and hence is the most widely used representation for building word models in a speech recognition system. State-of-the-art systems use context-dependent phone models as the basic units of recognition [99].

9.3.4 Step 2 — The Language Model

The language model assigns probabilities to sequences of words, based on the likelihood of that sequence of words occurring in the context of the task being performed by the speech recognition system. Hence the probability of the text string $W =$"Call home" for a telephone number identification task is zero since that string makes no sense for the specified task. There are many ways of building Language Models for specific tasks, including:

(1) statistical training from text databases transcribed from task-specific dialogs (a learning procedure)
(2) rule-based learning of the formal grammar associated with the task
(3) enumerating, by hand, all valid text strings in the language and assigning appropriate probability scores to each string.

The purpose of the language model, or grammar, is to enable the computation of the *a priori* probability, $P_L(W)$, of a word string, W, consistent with the recognition task [59, 60, 106]. Perhaps the most popular way of constructing the language model is through the use of a statistical N-gram word grammar that is estimated from a large training set of text utterances, either from the task at hand or from a generic database with applicability to a wide range of tasks. We now describe the way in which such a language model is built.

Assume we have a large text training set of word-labeled utterances. (Such databases could include millions or even tens of millions of text sentences.) For every sentence in the training set, we have a text file that identifies the words in that sentence. If we make the assumption that the probability of a word in a sentence is conditioned on *only* the previous $N-1$ words, we have the basis for an N-gram language model. Thus we assume we can write the probability of the sentence W, according to an N-gram language model, as

$$P_L(W) = P_L(w_1, w_2, \ldots, w_M) \tag{9.13}$$

$$= \prod_{n=1}^{M} P_L(w_n | w_{n-1}, w_{n-2}, \ldots, w_{n-N+1}), \tag{9.14}$$

where the probability of a word occurring in the sentence only depends on the previous $N-1$ words and we estimate this probability by counting the relative frequencies of N-tuples of words in the training set. Thus, for example, to estimate word "trigram" probabilities (i.e., the probability that a word w_n was preceded by the pair of words (w_{n-1}, w_{n-2})), we compute this quantity as

$$P(w_n|w_{n-1}, w_{n-2}) = \frac{C(w_{n-2}, w_{n-1}, w_n)}{C(w_{n-2}, w_{n-1})}, \tag{9.15}$$

where $C(w_{n-2}, w_{n-1}, w_n)$ is the frequency count of the word triplet (i.e., the trigram of words) consisting of (w_{n-2}, w_{n-1}, w_n) as it occurs in the text training set, and $C(w_{n-2}, w_{n-1})$ is the frequency count of the word doublet (i.e., bigram of words) (w_{n-2}, w_{n-1}) as it occurs in the text training set.

9.3.5 Step 3 — The Search Problem

The third step in the Bayesian approach to automatic speech recognition is to search the space of all valid word sequences from the language model, to find the one with the maximum likelihood of having been spoken. The key problem is that the potential size of the search space can be astronomically large (for large vocabularies and high average word branching factor language models), thereby taking inordinate amounts of computing power to solve by heuristic methods. Fortunately, through the use of methods from the field of Finite State Automata Theory, Finite State Network (FSN) methods have evolved that reduce the computational burden by orders of magnitude, thereby enabling exact maximum likelihood solutions in computationally feasible times, even for very large speech recognition problems [81].

The basic concept of a finite state network transducer is illustrated in Figure 9.10 which shows a word pronunciation network for the word /data/. Each arc in the state diagram corresponds to a phoneme in the word pronunciation network, and the weight is an estimate of the probability that the arc is utilized in the pronunciation of the word in context. We see that for the word /data/ there are four total pronunciations,

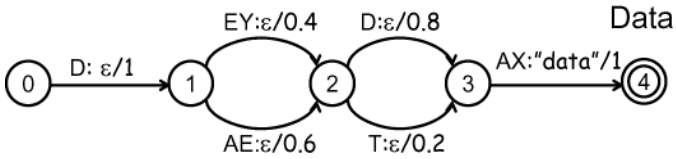

Fig. 9.10 Word pronunciation transducer for four pronunciations of the word /data/. (After Mohri [81].)

namely (along with their (estimated) pronunciation probabilities):

(1) /D/ /EY/ /D/ /AX/ — probability of 0.32
(2) /D/ /EY/ /T/ /AX/ — probability of 0.08
(3) /D/ /AE/ /D/ /AX/ — probability of 0.48
(4) /D/ /AE/ /T/ /AX/ — probability of 0.12.

The combined FSN of the 4 pronunciations is a lot more efficient than using 4 separate enumerations of the word since all the arcs are shared among the 4 pronunciations and the total computation for the full FSN for the word /data/ is close to 1/4 the computation of the 4 variants of the same word.

We can continue the process of creating efficient FSNs for each word in the task vocabulary (the speech dictionary or lexicon), and then combine word FSNs into sentence FSNs using the appropriate language model. Further, we can carry the process down to the level of HMM phones and HMM states, making the process even more efficient. Ultimately we can compile a very large network of model states, model phones, model words, and even model phrases, into a much smaller network via the method of weighted finite state transducers (WFST), which combine the various representations of speech and language and optimize the resulting network to minimize the number of search states (and, equivalently, thereby minimize the amount of duplicate computation). A simple example of such a WFST network optimization is given in Figure 9.11 [81].

Using the techniques of network combination (which include network composition, determinization, minimization, and weight pushing) and network optimization, the WFST uses a unified mathematical

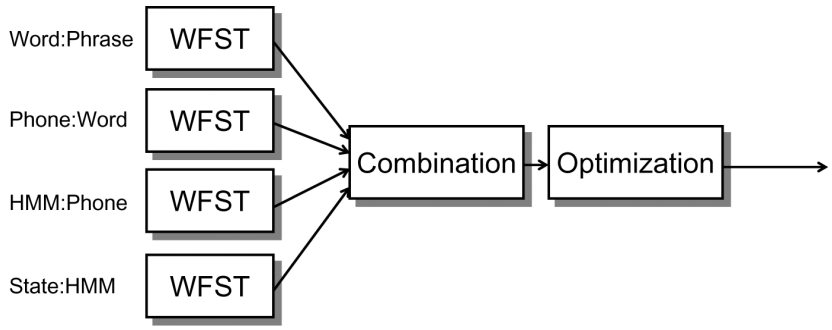

Fig. 9.11 Use of WFSTs to compile a set of FSNs into a single optimized network to minimize redundancy in the network. (After Mohri [81].)

framework to efficiently compile a large network into a minimal representation that is readily searched using standard Viterbi decoding methods [38]. Using these methods, an unoptimized network with 10^{22} states (the result of the cross product of model states, model phones, model words, and model phrases) was able to be compiled down to a mathematically equivalent model with 10^8 states that was readily searched for the optimum word string with no loss of performance or word accuracy.

9.4 Representative Recognition Performance

The block diagram of Figure 9.1 represents a wide range of possibilities for the implementation of automatic speech recognition. In Section 9.2.1, we suggest how the techniques of digital speech analysis discussed in Chapters 4–6 can be applied to extract a sequence of feature vectors from the speech signal, and in Section 9.3 we describe the most widely used statistical pattern recognition techniques that are employed for mapping the sequence of feature vectors into a sequence of symbols or words.

Many variations on this general theme have been investigated over the past 30 years or more, and as we mentioned before, testing and evaluation is a major part of speech recognition research and development. Table 9.1 gives a summary of a range of the performance of some speech recognition and natural language understanding systems

Table 9.1 Word error rates for a range of speech recognition systems.

Corpus	Type of speech	Vocabulary size	Word error rate (%)
Connected digit strings (TI Database)	Spontaneous	11 (0–9, oh)	0.3
Connected digit strings (AT&T Mall Recordings)	Spontaneous	11 (0–9, oh)	2.0
Connected digit strings (AT&T HMIHY©)	Conversational	11 (0–9, oh)	5.0
Resource management (RM)	Read speech	1000	2.0
Airline travel information system (ATIS)	Spontaneous	2500	2.5
North American business (NAB & WSJ)	Read text	64,000	6.6
Broadcast News	Narrated News	210,000	≈15
Switchboard	Telephone conversation	45,000	≈27
Call-home	Telephone conversation	28,000	≈35

that have been developed so far. This table covers a range of vocabulary sizes, speaking styles, and application contexts [92].

It can be seen that for a vocabulary of 11 digits, the word error rates are very low (0.3% for a very clean recording environment for the TI (Texas Instruments) connected digits database [68]), but when the digit strings are spoken in a noisy shopping mall environment the word error rate rises to 2.0% and when embedded within conversational speech (the AT&T HMIHY (How May I Help You) © system) the word error rate increases significantly to 5.0%, showing the lack of robustness of the recognition system to noise and other background disturbances [44]. Table 9.1 also shows the word error rates for a range of DARPA tasks ranging from

- read speech of commands and informational requests about a naval ships database (the resource management system or RM) with a 1000 word vocabulary and a word error rate of 2.0%
- spontaneous speech input for booking airlines travel (the airline travel information system, or ATIS [133]) with a 2500 word vocabulary and a word error rate of 2.5%

- read text from a range of business magazines and newspapers (the North American business task, or NAB) with a vocabulary of 64,000 words and a word error rate of 6.6%
- narrated news broadcasts from a range of TV news providers like CNBC, (the broadcast news task) with a 210,000 word vocabulary and a word error rate of about 15%
- recorded live telephone conversations between two unrelated individuals (the switchboard task [42]) with a vocabulary of 45,000 words and a word error rate of about 27%, and a separate task for live telephone conversations between two family members (the call home task) with a vocabulary of 28,000 words and a word error rate of about 35%.

9.5 Challenges in ASR Technology

So far, ASR systems fall far short of human speech perception in all but the simplest, most constrained tasks. Before ASR systems become ubiquitous in society, many improvements will be required in both system performance and operational performance. In the system area we need large improvements in accuracy, efficiency, and robustness in order to utilize the technology for a wide range of tasks, on a wide range of processors, and under a wide range of operating conditions. In the operational area we need better methods of detecting when a person is speaking to a machine and isolating the spoken input from the background, we need to be able to handle users talking over the voice prompts (so-called barge-in conditions), we need more reliable and accurate utterance rejection methods so we can be sure that a word needs to be repeated when poorly recognized the first time, and finally we need better methods of confidence scoring of words, phrases, and even sentences so as to maintain an intelligent dialog with a customer.

Conclusion

We have attempted to provide the reader with a broad overview of the field of digital speech processing and to give some idea as to the remarkable progress that has been achieved over the past 4–5 decades. Digital speech processing systems have permeated society in the form of cellular speech coders, synthesized speech response systems, and speech recognition and understanding systems that handle a wide range of requests about airline flights, stock price quotations, specialized help desks etc.

There remain many difficult problems yet to be solved before digital speech processing will be considered a mature science and technology. Our basic understanding of the human articulatory system and how the various muscular controls come together in the production of speech is rudimentary at best, and our understanding of the processing of speech in the human brain is at an even lower level.

In spite of our shortfalls in understanding, we have been able to create remarkable speech processing systems whose performance increases at a steady pace. A firm understanding of the fundamentals of acoustics, linguistics, signal processing, and perception provide the tools for

building systems that work and can be used by the general public. As we increase our basic understanding of speech, the application systems will only improve and the pervasiveness of speech processing in our daily lives will increase dramatically, with the end result of improving the productivity in our work and home environments.

Acknowledgments

We wish to thank Professor Robert Gray, Editor-in-Chief of Now Publisher's *Foundations and Trends in Signal Processing*, for inviting us to prepare this text. His patience, technical advice, and editorial skill were crucial at every stage of the writing. We also wish to thank Abeer Alwan, Yariv Ephraim, Luciana Ferrer, Sadaoki Furui, and Tom Quatieri for their detailed and perceptive comments, which greatly improved the final result. Of course, we are responsible for any weaknesses or inaccuracies that remain in this text.

References

[1] J. B. Allen and L. R. Rabiner, "A unified theory of short-time spectrum analysis and synthesis," *Proceedings of IEEE*, vol. 65, no. 11, pp. 1558–1564, November 1977.

[2] B. S. Atal, "Predictive coding of speech at low bit rates," *IEEE Transactions on Communications*, vol. COM-30, no. 4, pp. 600–614, April 1982.

[3] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *Journal of the Acoustical Society of America*, vol. 50, pp. 561–580, 1971.

[4] B. S. Atal and J. Remde, "A new model of LPC exitation for producing natural-sounding speech at low bit rates," *Proceedings of IEEE ICASSP*, pp. 614–617, 1982.

[5] B. S. Atal and M. R. Schroeder, "Adaptive predictive coding of speech signals," *Bell System Technical Journal*, vol. 49, pp. 1973–1986, October 1970.

[6] B. S. Atal and M. R. Schroeder, "Predictive coding of speech signals and subjective error criterion," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-27, pp. 247–254, June 1979.

[7] B. S. Atal and M. R. Schroeder, "Improved quantizer for adaptive predictive coding of speech signals at low bit rates," *Proceedings of ICASSP*, pp. 535–538, April 1980.

[8] T. B. Barnwell III, "Recursive windowing for generating autocorrelation analysis for LPC analysis," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-29, no. 5, pp. 1062–1066, October 1981.

[9] T. B. Barnwell III, K. Nayebi, and C. H. Richardson, *Speech Coding, A Computer Laboratory Textbook*. John Wiley and Sons, 1996.

[10] 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.

[11] L. E. Baum, T. Petri, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Annals in Mathematical Statistics*, vol. 41, pp. 164–171, 1970.

[12] W. R. Bennett, "Spectra of quantized signals," *Bell System Technical Journal*, vol. 27, pp. 446–472, July 1948.

[13] M. Berouti, H. Garten, P. Kabal, and P. Mermelstein, "Efficient computation and encoding of the multipulse excitation for LPC," *Proceedings of ICASSP*, pp. 384–387, March 1984.

[14] M. Beutnagel, A. Conkie, and A. K. Syrdal, "Diphone synthesis using unit selection," Third Speech Synthesis Workshop, Jenolan Caes, Australia, November 1998.

[15] M. Beutnatel and A. Conkie, "Interaction of units in a unit selection database," *Proceedings of Eurospeech '99,* Budapest, Hungary, September 1999.

[16] B. P. Bogert, M. J. R. Healy, and J. W. Tukey, "The quefrency alanysis of times series for echos: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking," in *Proceedings of the Symposium on Time Series Analysis*, (M. Rosenblatt, ed.), New York: John Wiley and Sons, Inc., 1963.

[17] E. Bresch, J. Nielsen, K. Nayak, and S. Narayanan, "Synchornized and noise-robust audio recordings during realtime MRI scans," *Journal of the Acoustical Society of America*, vol. 120, no. 4, pp. 1791–1794, October 2006.

[18] C. S. Burrus and R. A. Gopinath, *Introduction to Wavelets and Wavelet Transforms*. Prentice-Hall Inc., 1998.

[19] J. P. Campbell Jr., V. C. Welch, and T. E. Tremain, "An expandable error-pretected 4800 bps CELP coder," *Proceedings of ICASSP*, vol. 2, pp. 735–738, May 1989.

[20] F. Charpentier and M. G. Stella, "Diphone synthesis using an overlap-add technique for speech waveform concatenation," *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pp. 2015–2018, 1986.

[21] J. H. Chung and R. W. Schafer, "Performance evaluation of analysis-by-synthesis homomorphic vocoders," *Proceedings of IEEE ICASSP*, vol. 2, pp. 117–120, March 1992.

[22] C. H. Coker, "A model of articulatory dynamics and control," *Proceedings of IEEE*, vol. 64, pp. 452–459, 1976.

[23] R. V. Cox, S. L. Gay, Y. Shoham, S. Quackenbush, N. Seshadri, and N. Jayant, "New directions in subband coding," *IEEE Journal of Selected Areas in Communications*, vol. 6, no. 2, pp. 391–409, February 1988.

[24] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Prentice-Hall Inc., 1983.

[25] R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital coding of speech in subbands," *Bell System Technical Journal*, vol. 55, no. 8, pp. 1069–1085, October 1976.

[26] C. C. Cutler, "Differential quantization of communication signals," U.S. Patent 2,605,361, July 29, 1952.

[27] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, pp. 357–366, August 1980.

[28] F. deJager, "Delta modulation — a new method of PCM transmission using the 1-unit code," *Philips Research Reports*, pp. 442–466, December 1952.

[29] P. B. Denes and E. N. Pinson, *The speech chain*. W. H. Freeman Company, 2nd Edition, 1993.

[30] H. Dudley, "The vocoder," *Bell Labs Record*, vol. 17, pp. 122–126, 1939.

[31] T. Dutoit, *An Introduction to Text-to-Speech Synthesis*. Netherlands: Kluwer Academic Publishers, 1997.

[32] G. Fant, *Acoustic Theory of Speech Production*. The Hague: Mouton & Co., 1960; Walter de Gruyter, 1970.

[33] J. D. Ferguson, "Hidden Markov Analysis: An Introduction," *Hidden Markov Models for Speech*, Princeton: Institute for Defense Analyses, 1980.

[34] J. L. Flanagan, *Speech Analysis, Synthesis and Perception*. Springer-Verlag, 1972.

[35] J. L. Flanagan, C. H. Coker, L. R. Rabiner, R. W. Schafer, and N. Umeda, "Synthetic voices for computers," *IEEE Spectrum*, vol. 7, pp. 22–45, October 1970.

[36] J. L. Flanagan, K. Ishizaka, and K. L. Shipley, "Synthesis of speech from a dynamic model of the vocal cords and vocal tract," *Bell System Technical Journal*, vol. 54, no. 3, pp. 485–506, March 1975.

[37] H. Fletcher and W. J. Munson, "Loudness, its definition, measurement and calculation," *Journal of Acoustical Society of America*, vol. 5, no. 2, pp. 82–108, October 1933.

[38] G. D. Forney, "The Viterbi algorithm," *IEEE Proceedings*, vol. 61, pp. 268–278, March 1973.

[39] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Transactions on Acoustics Speech, and Signal Processing*, vol. ASSP-29, no. 2, pp. 254–272, April 1981.

[40] S. Furui, "Speaker independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. ASSP-26, no. 1, pp. 52–59, February 1986.

[41] O. Ghitza, "Audiotry nerve representation as a basis for speech processing," in *Advances in Speech Signal Processing*, (S. Furui and M. Sondhi, eds.), pp. 453–485, NY: Marcel Dekker, 1991.

[42] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone Speech corpus for research and development," *Proceedings of ICASSP 1992*, pp. 517–520, 1992.

[43] B. Gold and L. R. Rabiner, "Parallel processing techniques for estimating pitch period of speech in the time domain," *Journal of Acoustical Society of America*, vol. 46, no. 2, pt. 2, pp. 442–448, August 1969.

[44] A. L. Gorin, B. A. Parker, R. M. Sachs, and J. G. Wilpon, "How may I help you?," *Proceedings of the Interactive Voice Technology for Telecommunications Applications (IVTTA)*, pp. 57–60, 1996.

[45] R. M. Gray, "Vector quantization," *IEEE Signal Processing Magazine*, pp. 4–28, April 1984.

[46] R. M. Gray, "Toeplitz and circulant matrices: A review," *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 3, pp. 155–239, 2006.

[47] J. A. Greefkes and K. Riemens, "Code modulation with digitally controlled companding for speech transmission," *Philips Technical Review*, pp. 335–353, 1970.

[48] H. Hermansky, "Auditory modeling in automatic recognition of speech," in *Proceedings of First European Conference on Signal Analysis and Prediction*, pp. 17–21, Prague, Czech Republic, 1997.

[49] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*. Prentice-Hall Inc., 2001.

[50] A. Hunt and A. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," *Proceedings of ICASSP-96,* Atlanta, vol. 1, pp. 373–376, 1996.

[51] K. Ishizaka and J. L. Flanagan, "Synthesis of voiced sounds from a two-mass model of the vocal cords," *Bell System Technical Journal*, vol. 51, no. 6, pp. 1233–1268, 1972.

[52] F. Itakura, "Line spectrum representation of linear predictive coefficients of speech signals," *Journal of Acoustical Society of America*, vol. 57, pp. 535(a), p. s35(A).

[53] F. Itakura and S. Saito, "Analysis-synthesis telephony based upon the maximum likelihood method," *Proceedings of 6th International of Congress on Acoustics*, pp. C17–C20, 1968.

[54] F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Electronics and Communications in Japan*, vol. 53-A, no. 1, pp. 36–43, 1970.

[55] F. Itakura and T. Umezaki, "Distance measure for speech recognition based on the smoothed group delay spectrum," in *Proceedings of ICASSP87*, pp. 1257–1260, Dallas TX, April 1987.

[56] N. S. Jayant, "Adaptive delta modulation with a one-bit memory," *Bell System Technical Journal*, pp. 321–342, March 1970.

[57] N. S. Jayant, "Adaptive quantization with one word memory," *Bell System Technical Journal*, pp. 1119–1144, September 1973.

[58] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Prentice-Hall, 1984.

[59] F. Jelinek, *Statistical Methods for Speech Recognition*. Cambridge: MIT Press, 1997.

[60] F. Jelinek, R. L. Mercer, and S. Roucos, "Principles of lexical language modeling for speech recognition," in *Advances in Speech Signal Processing*, (S. Furui and M. M. Sondhi, eds.), pp. 651–699, Marcel Dekker, 1991.

[61] B. H. Juang, "Maximum likelihood estimation for mixture multivariate stochastic observations of Markov chains," *AT&T Technology Journal*, vol. 64, no. 6, pp. 1235–1249, 1985.

[62] B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Transactions in Information Theory*, vol. 32, no. 2, pp. 307–309, 1986.

[63] B. H. Juang, L. R. Rabiner, and J. G. Wilpon, "On the use of bandpass liftering in speehc recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-35, no. 7, pp. 947–954, July 1987.

[64] D. H. Klatt, "Software for a cascade/parallel formant synthesizer," *Journal of the Acoustical Society of America*, vol. 67, pp. 971–995, 1980.

[65] D. H. Klatt, "Review of text-to-speech conversion for English," *Journal of the Acoustical Society of America*, vol. 82, pp. 737–793, September 1987.

[66] W. Koenig, H. K. Dunn, and L. Y. Lacey, "The sound spectrograph," *Journal of the Acoustical Society of America*, vol. 18, pp. 19–49, 1946.

[67] P. Kroon, E. F. Deprettere, and R. J. Sluyter, "Regular-pulse excitation: A nove approach to effective and efficient multipulse coding of speech," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 1054–1063, October 1986.

[68] R. G. Leonard, "A database for speaker-independent digit recognition," *Proceedings of ICASSP 1984*, pp. 42.11.1–42.11.4, 1984.

[69] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell System Technical Journal*, vol. 62, no. 4, pp. 1035–1074, 1983.

[70] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, January 1980.

[71] S. P. Lloyd, "Least square quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, March 1982.

[72] P. C. Loizou, *Speech Enhancement, Theory and Practice*. CRC Press, 2007.

[73] R. F. Lyon, "A computational model of filtering, detection and compression in the cochlea," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Paris, France, May 1982.

[74] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of IEEE*, vol. 63, pp. 561–580, 1975.

[75] J. Makhoul, V. Viswanathan, R. Schwarz, and A. W. F. Huggins, "A mixed source model for speech compression and synthesis," *Journal of the Acoustical Society of America*, vol. 64, pp. 1577–1581, December 1978.

[76] D. Malah, "Time-domain algorithms for harmonic bandwidth reduction and time-scaling of pitch signals," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 27, no. 2, pp. 121–133, 1979.

[77] J. D. Markel, "The SIFT algorithm for fundamental frequency estimation," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, no. 5, pp. 367–377, December 1972.

[78] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*. New York: Springer-Verlag, 1976.

[79] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. IT-6, pp. 7–12, March 1960.
[80] A. V. McCree and T. P. Barnwell III, "A mixed excitation LPC vocoder model for low bit rate speech coding," *IEEE Transactions on Speech and Audio Processing*, vol. 3, no. 4, pp. 242–250, July 1995.
[81] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 2, pp. 269–312, 1997.
[82] E. Moulines and F. Charpentier, "Pitch synchronous waveform processing techniques for text-to-speech synthesis using diphones," *Speech Communication*, vol. 9, no. 5–6, 1990.
[83] A. M. Noll, "Cepstrum pitch determination," *Journal of the Acoustical Society of America*, vol. 41, no. 2, pp. 293–309, February 1967.
[84] P. Noll, "A comparative study of various schemes for speech encoding," *Bell System Technical Journal*, vol. 54, no. 9, pp. 1597–1614, November 1975.
[85] A. V. Oppenehim, "Superposition in a class of nonlinear systems," PhD dissertation, MIT, 1964. Also: MIT Research Lab. of Electronics, Cambridge, Massachusetts, Technical Report No. 432, 1965.
[86] A. V. Oppenheim, "A speech analysis-synthesis system based on homomorphic filtering," *Journal of the Acoustical Society of America*, vol. 45, no. 2, pp. 293–309, February 1969.
[87] A. V. Oppenheim, "Speech spectrograms using the fast Fourier transform," *IEEE Spectrum*, vol. 7, pp. 57–62, August 1970.
[88] A. V. Oppenheim and R. W. Schafer, "Homomorphic analysis of speech," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-16, pp. 221–228, June 1968.
[89] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Prentice-Hall Inc., 1999.
[90] A. V. Oppenheim, R. W. Schafer, and T. G. Stockham Jr., "Nonlinear filtering of multiplied and convolved signals," *Proceedings of IEEE*, vol. 56, no. 8, pp. 1264–1291, August 1968.
[91] M. D. Paez and T. H. Glisson, "Minimum mean-squared error quantization in speech," *IEEE Transactions on Communications*, vol. Com-20, pp. 225–230, April 1972.
[92] D. S. Pallett et al., "The 1994 benchmark tests for the ARPA spoken language program," *Proceedings of 1995 ARPA Human Language Technology Workshop*, pp. 5–36, 1995.
[93] M. R. Portnoff, "A quasi-one-dimensional simulation for the time-varying vocal tract," MS Thesis, MIT, Department of Electrical Engineering, 1973.
[94] T. F. Quatieri, *Discrete-time speech signal processing*. Prentice Hall, 2002.
[95] L. R. Rabiner, "A model for synthesizing speech by rule," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-17, no. 1, pp. 7–13, March 1969.
[96] L. R. Rabiner, "On the use of autocorrelation analysis for pitch detection," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-25, no. 1, pp. 24–33, February 1977.
[97] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *IEEE Proceedings*, vol. 77, no. 2, pp. 257–286, 1989.

[98] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE Signal Processing Magazine*, 1985.

[99] L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall Inc., 1993.

[100] L. R. Rabiner and M. R. Sambur, "An algorithm for determining the endpoints of isolated utterances," *Bell System Technical Journal*, vol. 54, no. 2, pp. 297–315, February 1975.

[101] L. R. Rabiner and R. W. Schafer, *Theory and Application of Digital Speech Processing*. Prentice-Hall Inc., 2009. (In preparation).

[102] L. R. Rabiner, R. W. Schafer, and J. L. Flanagan, "Computer synthesis of speech by concatenation of formant coded words," *Bell System Technical Journal*, vol. 50, no. 5, pp. 1541–1558, May–June 1971.

[103] D. W. Robinson and R. S. Dadson, "A re-determination of the equal-loudness contours for pure tones," *British Journal of Applied Physics*, vol. 7, pp. 166–181, 1956.

[104] R. C. Rose and T. P. Barnwell III, "The self excited vocoder — an alternate approach to toll quality at 4800 bps," *Proceedings of ICASSP '86*, vol. 11, pp. 453–456, April 1986.

[105] A. E. Rosenberg, "Effect of glottal pulse shape on the quality of natural vowels," *Journal of the Acoustical Society of America*, vol. 43, no. 4, pp. 822–828, February 1971.

[106] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," *IEEE Proceedings*, vol. 88, no. 8, pp. 1270–1278, 2000.

[107] M. B. Sachs, C. C. Blackburn, and E. D. Young, "Rate-place and temporal-place representations of vowels in the auditory nerve and anteroventral cochlear nucleus," *Journal of Phonetics*, vol. 16, pp. 37–53, 1988.

[108] Y. Sagisaka, "Speech synthesis by rule using an optimal selection of non-uniform synthesis units," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 679–682, 1988.

[109] R. W. Schafer, "Echo removal by discrete generalized linear filtering," PhD dissertation, MIT, 1968. Also: MIT Research Laboratory of Electronics, Cambridge, Massachusetts, Technical Report No. 466, 1969.

[110] R. W. Schafer, "Homomorphic systems and cepstrum analysis of speech," *Springer Handbook of Speech Processing and Communication,* Springer, 2007.

[111] R. W. Schafer and L. R. Rabiner, "System for automatic formant analysis of voiced speech," *Journal of the Acoustical Society of America*, vol. 47, no. 2, pp. 458–465, February 1970.

[112] M. R. Schroeder and B. S. Atal, "Code-excited linear prediction (CELP): High-quality speech at very low bit rates," *Proceedings of IEEE ICASSP*, pp. 937–940, 1985.

[113] M. R. Schroeder and E. E. David, "A vocoder for transmitting 10 kc/s speech over a 3.5 kc/s channel," *Acustica*, vol. 10, pp. 35–43, 1960.

[114] J. H. Schroeter, "Basic principles of speech synthesis," *Springer Handbook of Speech Processing,* Springer-Verlag, 2006.

[115] S. Seneff, "A joint synchrony/mean-rate model of auditory speech processing," *Journal of Phonetics*, vol. 16, pp. 55–76, 1988.

[116] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, 1949.

[117] G. A. Sitton, C. S. Burrus, J. W. Fox, and S. Treitel, "Factoring very-high-degree polynomials," *IEEE Signal Processing Magazine*, vol. 20, no. 6, pp. 27–42, November 2003.

[118] B. Smith, "Instantaneous companding of quantized signals," *Bell System Technical Journal*, vol. 36, no. 3, pp. 653–709, May 1957.

[119] F. K. Soong and B.-H. Juang, "Optimal quantization of LSP parameters," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 1, pp. 15–24, January 1993.

[120] A. Spanias, T. Painter, and V. Atti, *Audio Signal Processing and Coding*. Wiley Interscience, 2007.

[121] K. N. Stevens, *Acoustic Phonetics*. MIT Press, 1998.

[122] S. S. Stevens and J. Volkman, "The relation of pitch to frequency," *American Journal of Psychology*, vol. 53, p. 329, 1940.

[123] L. C. Stewart, R. M. Gray, and Y. Linde, "The design of trellis waveform coders," *IEEE transactions on Communications*, vol. COM-30, pp. 702–710, April 1982.

[124] T. G. Stockham Jr., T. M. Cannon, and R. B. Ingebretsen, "Blind deconvolution through digital signal processing," *Proceedings of IEEE*, vol. 63, pp. 678–692, April 1975.

[125] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[126] Y. Tohkura, "A weighted cepstral distance measure for speech recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, pp. 1414–1422, October 1987.

[127] J. M. Tribolet, "A new phase unwrapping algorithm," *IEEE Transactions on Acoustical, Speech, and Signal Processing*, vol. ASSP-25, no. 2, pp. 170–177, April 1977.

[128] C. K. Un and D. T. Magill, "The residual-excited linear prediction vocoder with transmission rate below 9.6 kbits/s," *IEEE Transactions on Communications*, vol. COM-23, no. 12, pp. 1466–1474, December 1975.

[129] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Prentice-Hall Inc., 1993.

[130] R. Viswanathan, W. Russell, and J. Makhoul, "Voice-excited LPC coders for 9.6 kbps speech transmission," vol. 4, pp. 558–561, April 1979.

[131] V. Viswanathan and J. Makhoul, "Quantization properties of transmission parameters in linear predictive systems," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-23, no. 3, pp. 309–321, June 1975.

[132] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding Aalgorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.

[133] W. Ward, "Evaluation of the CMU ATIS system," *Proceedings of DARPA Speech and Natural Language Workshop*, pp. 101–105, February 1991.

[134] E. Zwicker and H. Fastl, *Psycho-acoustics*. Springer-Verlag, 2nd Edition, 1990.

Supplemental References

The specific references that comprise the Bibliography of this text are representative of the literature of the field of digital speech processing. In addition, we provide the following list of journals and books as a guide for further study. Listing the books in chronological order of publication provides some perspective on the evolution of the field.

Speech Processing Journals

- *IEEE Transactions on Signal Processing.* Main publication of IEEE Signal Processing Society.
- *IEEE Transactions on Speech and Audio.* Publication of IEEE Signal Processing Society that is focused on speech and audio processing.
- *Journal of the Acoustical Society of America.* General publication of the American Institute of Physics. Papers on speech and hearing as well as other areas of acoustics.
- *Speech Communication.* Published by Elsevier. A publication of the European Association for Signal Processing (EURASIP) and of the International Speech Communication Association (ISCA).

General Speech Processing References

- *Speech Analysis, Synthesis and Perception*, J. L. Flanagan, Springer-Verlag, Second Edition, Berlin, 1972.
- *Linear Prediction of Speech*, J. D. Markel and A. H. Gray, Jr., Springer Verlag, Berlin, 1976.
- *Digital Processing of Speech Signals*, L. R. Rabiner and R. W. Schafer, Prentice-Hall Inc., 1978.
- *Speech Analysis*, R. W. Schafer and J. D. Markel (eds.), IEEE Press Selected Reprint Series, 1979.
- *Speech Communication, Human and Machine*, D. O'Shaughnessy, Addison-Wesley, 1987.
- *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi, Marcel Dekker Inc., New York, 1991.
- *Discrete-Time Processing of Speech Signals*, J. Deller, Jr., J. H. L. Hansen, and J. G. Proakis, Wiley-IEEE Press, Classic Reissue, 1999.
- *Acoustic Phonetics*, K. N. Stevens, MIT Press, 1998.
- *Speech and Audio Signal Processing*, B. Gold and N. Morgan, John Wiley and Sons, 2000.
- *Digital Speech Processing, Synthesis and Recognition*, S. Furui, Second Edition, Marcel Dekker Inc., New York, 2001.
- *Discrete-Time Speech Signal Processing*, T. F. Quatieri, Prentice Hall Inc., 2002.
- *Speech Processing, A Dynamic and Optimization-Oriented Approach*, L. Deng and D. O'Shaughnessy, Marcel Dekker, 2003.
- *Springer Handbook of Speech Processing and Speech Communication*, J. Benesty, M. M. Sondhi and Y Huang (eds.), Springer, 2008.
- *Theory and Application of Digital Speech Processing*, L. R. Rabiner and R. W. Schafer, Prentice Hall Inc., 2009.

Speech Coding References

- *Digital Coding of Waveforms*, N. S. Jayant and P. Noll, Prentice Hall Inc., 1984.
- *Practical Approaches to Speech Coding*, P. E. Papamichalis, Prentice Hall Inc., 1987.
- *Vector Quantization and Signal Compression*, A. Gersho and R. M. Gray, Kluwer Academic Publishers, 1992.
- *Speech Coding and Synthesis*, W. B. Kleijn and K. K. Paliwal, Elsevier, 1995.
- *Speech Coding, A Computer Laboratory Textbook*, T. P. Barnwell and K. Nayebi, John Wiley and Sons, 1996.

- *A Practical Handbook of Speech Coders*, R. Goldberg and L. Riek, CRC Press, 2000.
- *Speech Coding Algorithms*, W. C. Chu, John Wiley and Sons, 2003.
- *Digital Speech: Coding for Low Bit Rate Communication Systems*, Second Edition, A. M. Kondoz, John Wiley and Sons, 2004.

Speech Synthesis

- *From Text to Speech*, J. Allen, S. Hunnicutt and D. Klatt, Cambridge University Press, 1987.
- *Acoustics of American English*, J. P. Olive, A. Greenwood and J. Coleman, Springer-Verlag, 1993.
- *Computing Prosody*, Y. Sagisaka, N. Campbell and N. Higuchi, Springer-Verlag, 1996.
- *Progress in Speech Synthesis*, J. VanSanten, R. W. Sproat, J. P. Olive and J. Hirschberg (eds.), Springer-Verlag, 1996.
- *An Introduction to Text-to-Speech Synthesis*, T. Dutoit, Kluwer Academic Publishers, 1997.
- *Speech Processing and Synthesis Toolboxes*, D. Childers, John Wiley and Sons, 1999.
- *Text To Speech Synthesis: New Paradigms and Advances*, S. Narayanan and A. Alwan (eds.), Prentice Hall Inc., 2004.
- *Text-to-Speech Synthesis*, P. Taylor, Cambridge University Press, 2008.

Speech Recognition and Natural Language Processing

- *Fundamentals of Speech Recognition*, L. R. Rabiner and B. H. Juang, Prentice Hall Inc., 1993.
- *Connectionist Speech Recognition-A Hybrid Approach*, H. A. Bourlard and N. Morgan, Kluwer Academic Publishers, 1994.
- *Automatic Speech and Speaker Recognition*, C. H. Lee, F. K. Soong and K. K. Paliwal (eds.), Kluwer Academic Publisher, 1996.
- *Statistical Methods for Speech Recognition*, F. Jelinek, MIT Press, 1998.
- *Foundations of Statistical Natural Language Processing*, C. D. Manning and H. Schutze, MIT Press, 1999.
- *Spoken Language Processing*, X. Huang, A. Acero and H.-W. Hon, Prentice Hall Inc., 2000.
- *Speech and Language Processing*, D. Jurafsky and J. H. Martin, Prentice Hall Inc., 2000.
- *Mathematical Models for Speech Technology*, S. E. Levinson, John Wiley and Sons, 2005.

Speech Enhancement

- *Digital Speech Transmission, Enhancement, Coding and Error Concealment*, P. Vary and R. Martin, John Wiley and Sons, Ltd., 2006.
- *Speech Enhancement, Theory and Practice*, P. C. Loizou, CRC Press, 2007.

Audio Processing

- *Applications of Digital Signal Processing to Audio and Acoustics*, H. Kahrs and K. Brandenburg (eds.), Kluwer Academic Publishers, 1998.
- *Audio Signal Processing and Coding*, A. Spanias, T. Painter and V. Atti, John Wiley and Sons, 2007.