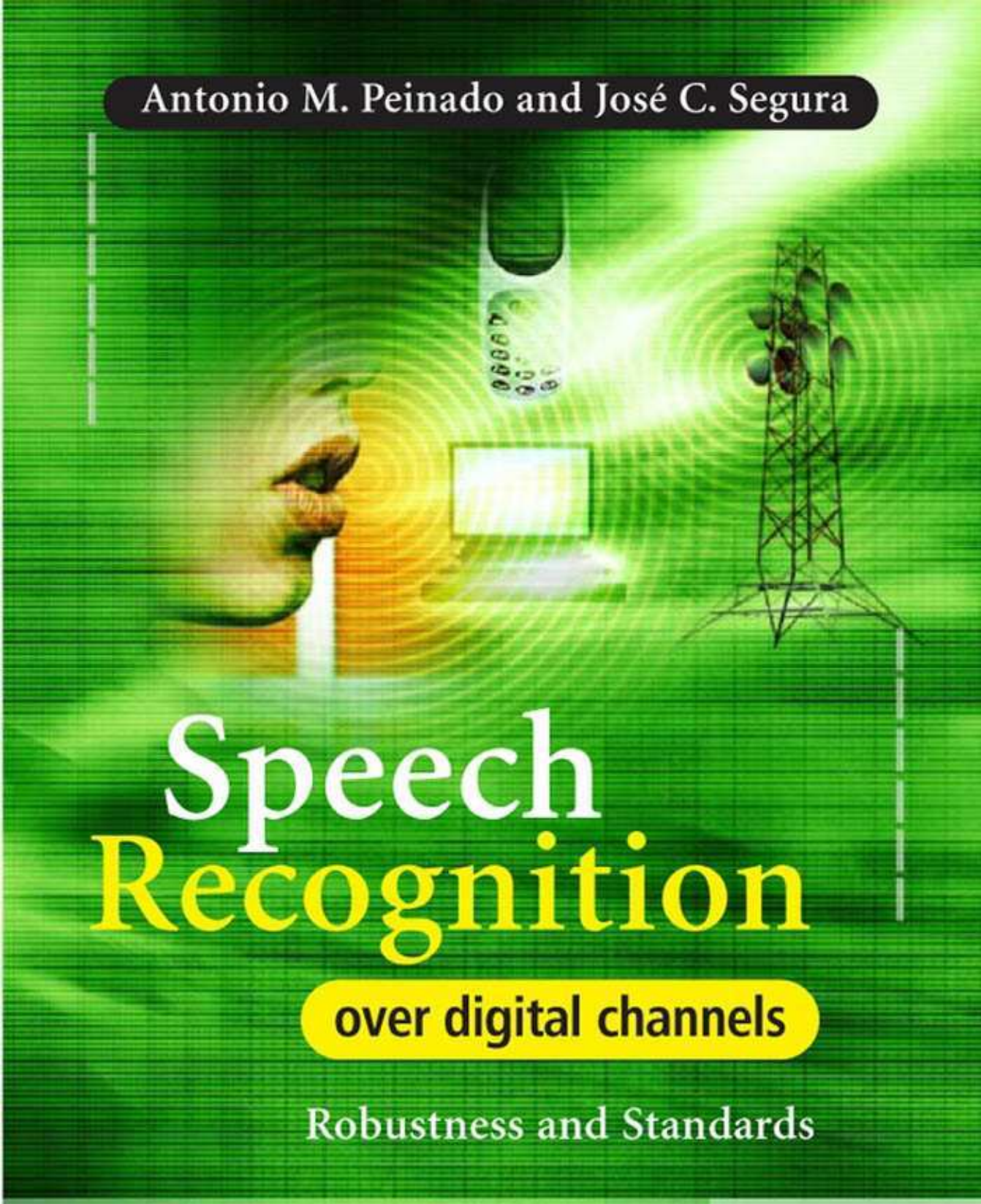
Antonio M. Peinado and José C. Segura



Speech Recognition over digital channels

Robustness and Standards

WILEY

SPEECH RECOGNITION OVER DIGITAL CHANNELS

SPEECH RECOGNITION OVER DIGITAL CHANNELS

Robustness and Standards

Antonio M. Peinado
University of Granada, Spain
José C. Segura
University of Granada, Spain

John Wiley & Sons, Ltd

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on www.wiley.com

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, ONT, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Peinado, Antonio.
Speech recognition over digital channels : robustness and standards /
Antonio Peinado, José C. Segura.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-470-02400-3 (cloth : alk. paper)
ISBN-10: 0-470-02400-3 (cloth : alk. paper)
1. Automatic speech recognition. 2. Signal processing. I. Segura, José C. II. Title.
TK7895.S65P45 2006
621.384 – dc22

2006014790

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN-13: 978-0-470-02400-3
ISBN-10: 0-470-02400-3

Typeset in 10/12 Times by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham, Wiltshire
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

To my Parents, José and María, and to the memory of Fausto . . .

. . . and to Victoria, Pilar and Enrique

To my Parents, José and Carmen, . . .

. . . and to Pepa, Juanjo and María

Contents

Foreword

The total number of mobile phone subscribers worldwide is expected to exceed two billion in 2006. While ordinary voice calling remains the dominant "application", mobile devices are becoming increasingly sophisticated, with features like multimedia messaging, cameras, web browsers, games, video, and music. The data capabilities of mobile networks are also improving, with 2.5G packet data networks widely available and the number of 3G network deployments growing. High speed data networks can deliver a wide range of audio and visual information to devices, and they enable access to remote content while on the move. The functionality of devices combined with the connectivity of the mobile network provides the opportunity for an exciting range of new services, including examples such as mobile search and location based services.

User interface technology, however, has not improved at the same rate, and small keypads and displays are still barriers to usability. This is where voice has the potential to provide a very powerful solution, since the use of reliable voice input can greatly improve data entry. Consider for example a navigation application and the difference in speed between speaking the destination address compared to typing it on a numeric keypad. In some situations pure speech input and output are appropriate, while if speech is combined with a visual modality the resulting *multimodal* user interface can greatly enhance the mobile user's experience and the effectiveness of the interaction.

Despite this strong motivation for a voice or multimodal interface to a mobile device, the downside of a bad experience resulting from misrecognition is high. Therefore the issues around achieving robustness and reliability become key technical challenges in enabling the capability. The user, of course, wants to access the services wherever they are located which includes many different background noise environments (the office, the home, a street corner, a car, a busy airport lounge. . . .), so noise robustness is a key issue.

For some applications it is best to use a local recognizer on the device itself. For example, interfacing to the phone functions and voice dialing using a personal address book. There are many applications where it is advantageous to use the resources of a remote server – especially when the content itself resides remotely and may be changing. The server speech recognition engine implementation is not constrained to the memory and processing power of the device and alternative specialized speech engines and data files can be used. In addition, new applications can also be more easily introduced, refined, extended and upgraded at the server. For recognition performed at a network server, the transmission channel becomes part of the recognition chain and needs to be

taken into consideration. For some channels reliable transmission can be obtained, while for others there will be transmission errors and techniques that enable robustness to those needed.

Distributed Speech Recognition is the main solution that the participants in the mobile industry (device manufactures & mobile operators) and speech recognition technology providers have developed to provide a robust recognition capability over mobile networks. Much of this work has been progressed in the telecommunications standards bodies of ETSI (European Telecommunications Standards Institute) and 3GPP (3rd Generation Partnership Project), while also supported and complemented by a body of wider research in academia and industry. These standards were needed to enable interoperability between the "front-end" feature extraction performed on the client device and the "back-end" recognition decoder at a remote server. The standards themselves were developed within ETSI with the subgroup formally called the "STQ-Aurora DSR Working Group" (more commonly referred to as "Aurora") and resulted in the publication of the standards for the DSR Advanced front-end (Oct 2002) and the Extended Advanced Front-end (Nov 2003). Following on from this the DSR standards were further extensively tested and evaluated within 3GPP to address their requirements for "Speech Enabled Services" (3GPP is the body that sets the standards for GSM and UMTS mobile communications). In June 2004 3GPP approved the DSR Extended Advanced Front-end as the recommended codec for "Speech Enabled Services". This selection was based on extensive evaluations undertaken by two of the leading ASR vendors (IBM & Scansoft) that confirmed the performance advantages of DSR compared to the normal voice codec. The significance of the selection by 3GPP in the release 6 specifications is that DSR will find widespread deployment in future GSM and 3G mobile handsets that will usher in a new wave of applications both for speech-only services and for distributed multimodal interfaces.

The exploration and explanation of the DSR standards, front-end noise robustness and robustness to transmission channel are at the heart of the material covered within this book. It draws together in one place the information to enable the reader to easily obtain a good understanding of the signal processing algorithms that make up the standards as well as providing the bigger picture of the technologies and components of a complete end-to-end system. It will also provide an appreciation of where DSR fits in the wider picture of remote speech recognition and the reasons that motivated the DSR solution.

In addition to the DSR standards, the subject of remote speech recognition and the techniques to achieve robustness to transmission errors has become a topic of active and continuing research. This book provides an introduction to previous and current research in this area and provides a perspective from which to view the alternative approaches and their combination. This, together with the background on the component system pieces that contribute to the overall performance, serves to make the area much more accessible to anyone wanting to enter this field or learn about recent developments.

Thus, this book will be of great value to anyone interested in implementing distributed speech and multimodal systems and who would like to get a broader understanding of the DSR standards and their background. It also gives a well presented and organized perspective for researchers coming into the field of noise and transmission robust recognition. It is a very timely publication in both of these respects. Antonio Peinado and José Segura are both experts in the field and leading contributors to the research on

robust speech recognition and particularly channel robustness. Their depth of understanding has enabled them to do an outstanding job in selecting, organizing and explaining the material clearly. I thoroughly recommend this book to you and hope you enjoy learning from it.

David Pearce
Motorola Labs &
Chairman of the ETSI STQ Aurora DSR working group.

Preface

The rapid development of digital networks during the last few years has opened a new field of expansion for speech technologies. In particular, automatic speech recognition (ASR) is a promising way for an easy and natural user access to network services, and encourages the concepts of speech enabled services (SES) and remote speech recognition (RSR). The main difference between ASR and RSR systems is that RSR involves a digital network placed between the user and the recognition engine. In this sense, we can consider that the network, which may not be fully reliable, becomes an additional stage of the ASR system. This new paradigm of RSR involves important consequences for the users, since they can interact with the service even with thin clients, such as mobile phones or PDAs, in mobile environments. The introduction of mobility in the system implies, in turn, that the user may access the system in an adverse environment where acoustic noise may severely degrade the system performance. All these issues have been recently addressed in a series of ETSI standards. These standards are only the tip of the iceberg of a huge amount of effort made in the design and development of RSR systems. Our main motivation for writing this book was to organize and present to the reader all these contributions, especially those related with system robustness and existing standards, as the subtitle of the book indicates.

This book provides a wide scope of the topics involved in the design and development of RSR systems, as well as a review of the recent and current research effort in this field. Our starting point of view is to consider an RSR system as a whole since the different system stages may be interdependent as shown in the course of the book. In order to ease this global conception, this book provides the background material needed in speech processing (analysis, recognition, coding and transmission) and communication systems. Furthermore, research work carried out during the last few years on RSR is dealt with. Our intention is to provide more than a list and summary of the relevant papers in the field. Although the given references are an indispensable source for a book such as this, we have tried to interpret and organize this material and present it under a joint formulation when possible. Therefore, the last chapter, devoted to the standards, is presented as a natural continuation and conclusion of the previous chapters, and not as an isolated list of techniques. We feel that this approach will enable an easier understanding of these standards. The result is a multidisciplinary book that may be useful for engineers developing RSR systems, researchers on speech technologies or communications, and PhD students of related fields.

We would finally like to thank all the people who have helped us in the process of writing this book and, especially, our colleagues of the research group on signal processing, networking and communications at the University of Granada.

Antonio M. Peinado
José C. Segura
Granada, Spain

1

Introduction

1.1 Introduction

One of the most fascinating characteristics of humans is their capability to communicate ideas by means of speech. This capability is undoubtedly one of the facts that has allowed the development of our society. Man has been always attracted by the possibility of creating machines capable of producing and recognizing speech, imitating ourselves. An automatic speech recognition (ASR) system can be defined as a mechanism capable of decoding the signal produced in the vocal and nasal tracts of a human speaker into the sequence of linguistic units contained in the message that the speaker wants to communicate.

The final goal of ASR is the man–machine communication. This natural way of interaction has found many applications because of the fast development of different hardware and software technologies. The most relevant are access to information systems, aid to handicapped, automatic translation or oral system control.

The present book is focused on a wide class of applications that involve access through speech to remote information systems or services. Among other applications, we could mention entertainment information, flight reservation, or help in street navigation. This type of applications has been clearly boosted by the fast development of the digital networks (cellular and Internet) during the last 10 years. These systems involve a client–server architecture in which the server contains the information, and the client picks the speech, which is transmitted to the server in a suitable form. This speech-based interface offers a natural interaction and can be particularly useful in the case of very small user interfaces. Also, it can be part of a multimodal and multidevice service.

The place at which the speech is processed (at the client, at the server or at both) and the way this processing is performed define the system architecture. The most direct solution is known as embedded speech recognition. In this case, all speech processing (including recognition) is carried out at the local device. The request is sent to the remote information server, which gives back the corresponding response. The obvious problem of this approach is that the task of installing, maintaining and upgrading the speech recognition system falls on the user. Moreover, if the recognition system has a certain degree of complexity (large vocabulary, flexibility, etc.), it can be an arduous task to embed it in a device like a PDA or a cellular phone. However, work in this direction is

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

being developed in order to provide efficient and robust embedded ASR (Deligne *et al*., 2002; Haeb-Umbach, 1997; Varga *et al*., 2002), since this is an attractive solution for some applications, such as voice dialing or control operation in portable devices or hands-free operation in cars.

A very flexible and powerful alternative is what we will call *remote speech recognition* (RSR). In this case, a local device (telephone, mobile phone, PDA, a speech analysis and encoding program running in a computer, etc.) sends the speech signal or parameters through a transmission channel to a remote server that includes the speech recognition engine. This approach has several advantages, such as the use of a simpler client (which involves lower costs for the user), language portability, centralized server upgrading and maintenance, or the use of more sophisticated recognition techniques, which can make it more attractive. Figure 1.1 shows a general diagram of a remote information system using the RSR approach. After recognition, the server provides and transmits the required information to the client. The information returned could be text, data or even a vocal response either generated by a text-to-speech system or previously recorded. The server may include other functions such as a call control block or a VoiceXML browser, which manages the dialog between the user and the service. Figure 1.2 shows a block diagram of a possible system architecture including these features. The (information) contents may be even out of the voice platform, and accessed by HTTP. An example of how RSR can be integrated in a multimodal service can be found in D. Pearce *et al*. (2005).

There have been several initiatives to promote and study RSR. First, we can mention the COST action 249, entitled "continuous speech recognition over the telephone" developed during the period 1994–2000 (Martens, 2000). The goal of this project was to gather the achievements of different research groups for the definition of a voice-activated information service. The scientific objectives covered different issues of speech recognition (specially those related to multilinguality) and dialog systems. This action had its extension in the COST project 278, called *spoken language interaction in telecommunication*. The most productive activity has been developed by the European Telecommunications Standards Institute (ETSI)- Speech, Transmission Planning and Quality of Service

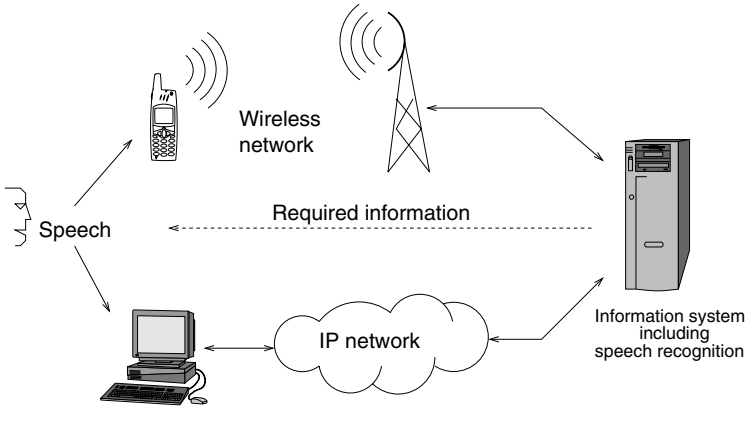


Figure 1.1 General scheme of a speech-driven remote information system

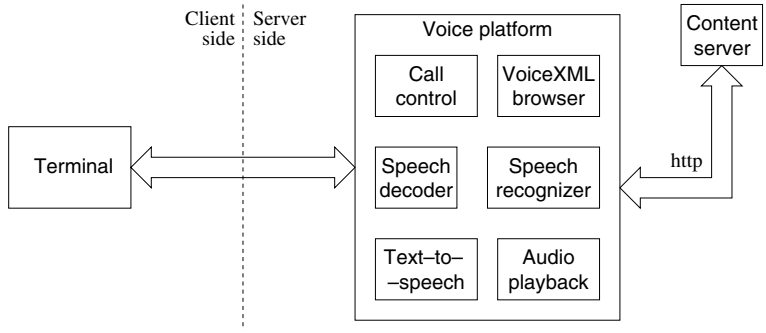


Figure 1.2 Block diagram of a possible RSR-based speech-enabled service architecture

(STQ) Aurora working group. This group is integrated in the STQ committee of the ETSI, and has developed several standards for RSR that are commented on later in this chapter.

The study of RSR systems involves a number of topics. However, the most characteristic issue that must be addressed and that differentiates this type of systems with respect to other ASR systems is robustness against degradation. There are two main sources of degradation that may affect a client–server ASR system. The first one is the noise introduced by the adverse acoustic environment. The acoustic noise is the unavoidable consequence of the noisy environment in which the speaker is usually placed (conference centers, airports, train stations, etc.) and can be treated at both the client and the server. The second one is the distortion introduced by the transmission channel. Typically, this distortion is the consequence of either a degraded wireless link (due to a bad coverage, fading, etc.) or of packet loss (in the case of transmission under the IP protocol), which can be treated by both, a suitable channel encoding at the client side or an error concealment technique applied in the server.

There are several possibilities for the implementation of an RSR system. The basic criterion to classify them is the way in which the speech is coded, transmitted and decoded. The first approaches to RSR were implemented over an analog (or a non-fully digital) transmission infrastructure, such as the public switched telephone network (PSTN). In this case, the speech signal is transmitted through the telephone line and sampled, analyzed and recognized at the server. This approach has the advantage of using a well-deployed network. Also, the whole speech signal is available at the server and can be fully processed there. However, its main drawback is the degradation introduced by the telephone channel, which has a narrow (250–3400 Hz) nonflat frequency response with unknown gain and phase, and this also introduces additive tones and stationary noise, impulse noise, amplitude and phase jitter, and so on. (Moreno and Stern, 1994). Besides, it does not benefit from the potential of a digital environment. The alternative is an RSR system implemented over a digital infrastructure. Researchers and developers have concentrated their attention on this possibility because of the multiple advantages it offers, such as robustness and the possibility of access to a wide range of services. In the following chapters, we will develop different issues regarding the implementation of RSR systems over digital channels, the problems that may arise, and some solutions to them.

1.2 RSR over Digital Channels

There are several possibilities for the implementation of an RSR system over a digital channel. In the first approach, usually known as *network speech recognition* (NSR), the recognition system resides in the network from the client's point of view. In this case, the speech is compressed by a speech codec in order to allow a low bitrate transmission and/or to use an existing speech traffic channel (as in the case of mobile telephony). The recognition is usually performed over the features extracted from the decoded signal, although it is also possible to extract the recognition features directly from the codec parameters. Figure 1.3 shows a scheme of this system architecture. In the case where implementation is over an IP network, a VoIP (Voice over IP) codec can be employed.

However, the approach that has received more attention during the last few years is the one known as *distributed speech recognition* (DSR). In this case, the client includes a local front end that processes the speech signal in order to directly obtain the specific features (usually cepstrum) used by the remote server (back end) to perform recognition, thus avoiding the coding/decoding process required by NSR. The conceptual scheme of DSR is shown in Figure 1.4. DSR has several advantages over NSR:

- It avoids the use of a speech codec, which can reduce the recognition performance because of compression.
- The bitrate involved in DSR is usually smaller than that of NSR.
- In mobile environments, DSR allows the increase of the system robustness. First, the front end located at the client can carry out some type of acoustic noise compensation. Also, the transmission can be carried out over a data channel instead of over a voice channel, so that the system is more robust against channel errors.
- It naturally enables multimodal interfaces by sending the speech features along with other information through a data channel.

On the other hand, the main advantage of NSR is that there is no need for modifying the existing clients in the case of mobile telephony networks.

In the same way as speech codecs are standardized for mobile telephony or VoIP, it is advised that a standardized feature extractor and encoder be used in DSR clients. The implementation of RSR systems over heterogeneous networks can also be eased by using DSR standards. Figure 1.5 shows two possible scenarios that mix mobile and IP

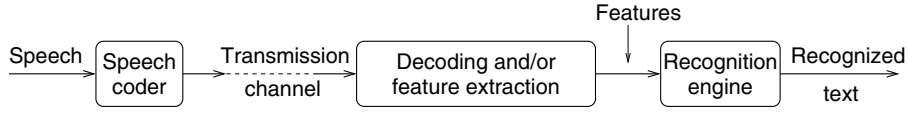


Figure 1.3 Scheme of a network speech recognition (NSR) system

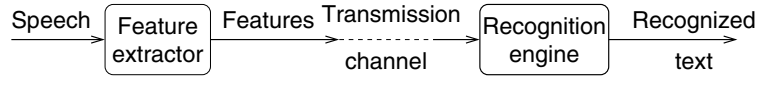


Figure 1.4 Scheme of a distributed speech recognition (DSR) system

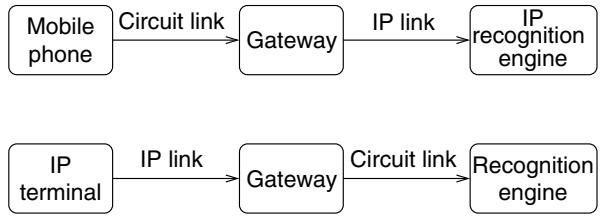


Figure 1.5 Two possible scenarios for a DSR system implemented over a heterogeneous network

networks. In both cases, the gateway must transcode the speech features, which can be straightforward if the mobile and IP payload are the same. On the contrary, NSR over heterogeneous networks may require the use of several codecs (tandeming), which can result in further speech degradation. The ETSI-STQ Aurora working group has been developing several DSR standards during the last few years: a basic standard for DSR (ETSI, 2003a) (front end FE), a DSR standard for working in noisy conditions (ETSI, 2003b) (advanced front end AFE), and two extensions of these two standards to enable tonal language recognition and speech reconstruction (ETSI, 2001, 2003c) (XFE and XAFE). In June 2004, 3GPP (3rd generation partnership project) approved XAFE as the recommended standard for speech-enabled services (SES).

1.3 Organization of the Book

The next chapter deals with the basic concepts required for the development and understanding of a state-of-the-art ASR system. Thus, the fundamentals of cepstral analysis and hidden Markov models (HMM) are introduced. Vector quantization (VQ) is also introduced in this chapter, since VQ is a useful tool in some HMM-based systems. A discussion is provided on how these tools can be used to build an ASR system. The chapter concludes with two specific topics, model adaptation and uncertainty treatment, that we consider specially useful for the development of RSR systems.

As mentioned earlier, an RSR system differs from a classical ASR system in that RSR systems are implemented over digital networks. The introduction of these networks (in particular, mobile and IP networks) in Chapter 3 has a double intention. The first is to provide the reader with those aspects that can be relevant for the implementation of RSR systems. On the other hand, as the subtitle of this book points out, the main topic of this book is robustness in RSR. This is the reason we are also interested in analyzing the degradation that may be introduced when RSR data is transmitted over these networks, which mainly results in data errors and losses. The chapter concludes with an in-depth study of the other source of degradation that may affect an RSR system: environmental noise. This includes the analysis and modeling of additive noise and linear channel distortion.

Chapter 4 deals with the two main architectures for RSR, NSR and DSR and their corresponding techniques for speech compression. First, some fundamentals of speech coding useful for both NSR and DSR are introduced. Then, the two variants of NSR (NSR from decoded speech and NSR from codec parameters) are studied, with special emphasis on degradation and robustness issues. The DSR approach is widely developed

next, since it has attracted the attention of a number of researchers during the last few years. They are classified and studied according to the compression scheme employed, although special attention is paid again to degradation and robustness.

The two subsequent chapters are devoted to the robustness techniques against both transmission channel and environmental degradation. Chapter 5 is devoted to the first type of degradation. The different techniques are classified into two groups: sender-driven (or channel coding) and receiver-based (or error concealment). The first group includes error detection and correction, interleaving and media-specific FEC. The second group includes classical techniques such as interpolation and estimation and also RSR-specific concealment techniques implemented in the speech recognizer. Chapter 6 is concerned with those techniques useful for the development of robust front ends, such as those included in the ETSI standards. In essence, this chapter deals with robust techniques against the environmental degradation, voice activity detection and feature normalization techniques. Robust back-end techniques are out of the scope of this book since they are not specific to RSR systems but apply to ASR systems in general.

The ETSI DSR standards are treated in Chapter 7. The goal of this chapter is to offer a comprehensive approach to these standards, which may facilitate their reading. The four standards are jointly developed so that it is easy to identify both common and differentiating elements. The study is carried out under the scope of the previous chapters, so the different elements contained in the standards can be easily identified and understood.

The book also includes three appendices on alternative representations of the linear prediction (LPC) coefficients, basic digital modulation concepts and a brief review of channel coding techniques. These appendices gather some procedures and concepts that may be useful for the understanding of the previous chapters, although we consider that they are neither the goal of this book nor essential. In particular, the last two appendices are specially developed for an appropriate comprehension of the communication concepts related with RSR and utilized in Chapters 3, 4, 5 and 7.

2

Speech Recognition with HMMs

2.1 Introduction

Speech recognition is possible thanks to the combined utilization of several types of information: acoustic, phonetic, syntactic and semantic. This multiplicity of information levels is a hint of the complexity of our problem. In addition, we have the difficulty introduced by the variability inherent to the speech production process. There are several sources of variability: First, we have intraspeaker variations (due to specific conditions of a given speaker) and interspeaker variations (due to different accents, gender, age, dialects, etc.). Besides, we must also consider the distortion (additive noise, channel distortion, etc.) that may affect the speech signal. Therefore, we need recognition tools capable of managing all this variability.

Speech recognition technologies are continuously studied and revised. The interested reader can find a recent survey on ASR in Lee (2003). However, there is a set of techniques that has made up a sort of standard during the last decade. This "standard" technology is mainly based on the use of HMMs and analysis techniques such as linear prediction or filter banks. We can find several free or open source packages, such as HTK or Sphinx-4, that implement it (HTK3, 2004; Sphinx4, 2004).

The main objective of this chapter is to describe the basic principles of speech recognition based on HMMs. First, we will review some general issues regarding speech recognition. After this, we will describe the different elements that make up an HMM-based speech recognition system. This system is depicted in Figure 2.1. It has the following stages:

- Analysis: The spectral information is extracted from the speech signal.
- Acoustic match: The spectral information is evaluated by the acoustic HMMs that cover all the phonetic varieties and words of the considered recognition task.
- Decision: The recognized sentence is finally obtained by combining the information obtained from the above stages with knowledge of the syntaxes and semantics of the recognition task.

The last two stages are usually integrated and their separation is merely conceptual. We will first review the speech analysis techniques based on linear prediction and filterbanks.

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

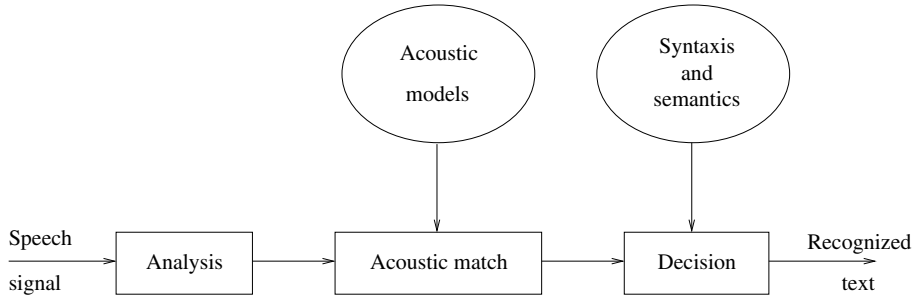


Figure 2.1 General scheme of an HMM-based speech recognition system

A review of VQ will also be introduced, since it is a useful tool in systems based on Markov models, as well as for parameter encoding. We will next introduce the statistical approach to speech recognition based on HMMs. We will define a discrete HMM (DHMM), study its application to speech recognition and present some of the main HMM variants. The chapter concludes with some practical implementation issues and other topics, such as model adaptation and the treatment of uncertainty by the recognition models, which will be useful in later chapters.

2.2 Some General Issues

As we can see in Figure 2.1, the recognition system requires to have stored a set of HMMs from which recognition can be carried out. These models are obtained during the *training* process. On the other hand, the system must be tested before its use. This is the *testing* stage. In order to perform both training and testing, there must be available a speech database, which must be divided into two different sets to perform separately training and testing. According to the speech data contained in the database, we can distinguish the following types of systems:

- Speaker-dependent: The system is only able to manage speech from a closed set of speakers, and, therefore, the training and testing is carried out with data from those speakers. This system is implemented when we are interested in recognizing only those speakers.
- Speaker-independent: The training database contains enough number of speakers and signals to perform the recognition of any new speaker with enough accuracy. Therefore, the testing is performed with a database containing different speakers than those used for training.

Speech recognition systems can also be differentiated by the linguistic *recognition unit* they use. Although we could consider the *word* as the most natural unit, it has the problem of a large set of units when we have to manage a large vocabulary. In this case, it is more suitable to use a *subword unit*. The basic subword units are the *context-independent phones*, which correspond to the basic phonemes of the language (Sugamura *et al*., 1983).

The problem of these units is that they cannot suitably model the coarticulatory effect between consecutive phonemes. In order to conceal this problem, other units such as diphonemes, syllables, demi-syllables or triphones have been proposed (Lee, 1990).

The recognition systems can also be classified according to the type of utterance to be recognized:

- Isolated word recognition (IWR): The words are isolated or can be easily isolated (i.e. when there are silences between consecutive words) and can be recognized one by one (normally using words as units).
- Continuous speech recognition (CSR): The utterances are whole sentences that do not necessarily have pauses between words. Although the recognition unit can be any one, subword units are commonly used. The purpose of CSR is to provide sentences adjusted to a set of rules known as *grammar*. The set of sentences generated by a given grammar is known as *language*. Thus, the uttered sentence must belong to this language for correct recognition. As we will see later in this chapter, both acoustics and language must be modeled in our statistical approach to recognition.

Finally, an important issue in speech recognition is how to measure the performance of the system. In the case of an IWR system, an obvious measure is the error rate (ER), measured as the rate between the number of words erroneously recognized and the total number of recognized words. However, in a CSR system, the ER is not suitable since there are different types of errors:

- Substitutions: A word of the original sentence appears substituted by a different word in the recognized sentence.
- Deletions: A word of the original sentence is missing in the recognized sentence.
- Insertions: The recognized sentence includes a new word between two words of the original sentence.

The performance measures most commonly used that consider these different errors are the percent correct (PC), the word error rate (WER) and the word accuracy (WAcc), defined as

$$PC = 100 \times \frac{C - I}{N} = \frac{N - (D + S)}{N} \tag{2.1}$$

$$WER = 100 \times \frac{S + D + I}{N} \tag{2.2}$$

$$WAcc = 100 - WER = 100 \times \frac{N - (S + D + I)}{N} \tag{2.3}$$

where C is the number of words correctly recognized, S is the total number of substitutions, D the total number of deletions, I the total number of insertions and N the total number of evaluated words. The evaluation of C, S, D and I requires the alignment of the recognized sentence with the original sentence, which is usually done by means of a dynamic programming algorithm (Deller *et al.*, 1993).

2.3 Analysis of Speech Signals

The goal of the analysis stage (or *front end*) is to provide a suitable parametric representation of the speech signal, appropriate for recognition. Figure 2.2 shows the general scheme of the analysis that is commonly applied. The short-term spectral representation is the one most used (Rabiner and Schafer, 1978). The signal is preprocessed and segmented into frames (20–40 ms), which are usually overlapped. If T_s is the shift between two consecutive frames, then $1/T_s$ is the frame rate (in Hertz). Inside a frame, the speech signal can be considered quasistationary, in order to obtain the short-term spectrum of each frame. The obtained spectral parameters are usually transformed to provide a frame representation more decorrelated and dimensionally reduced. The result of the analysis stage is that each frame is represented by a *feature vector* $\mathbf{x}$ containing the analysis parameters. The feature vector is usually enhanced by adding other features such as energy or dynamic features. The final result of the analysis stage is a sequence of feature vectors $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$.

There exist two main methods for spectral analysis in speech recognition: *filterbanks* and *linear prediction*. The linear prediction method has been classically used due to several reasons (Rabiner and Juang, 1993). For example, it is based on a powerful speech production model quite suitable for voiced sounds and still acceptable for unvoiced sounds. Also, when applied to recognition of nondistorted speech, it provides as good results or even better than the filterbank-based methods. However, filterbanks have been the main analysis tool during the past years since they have shown a better behavior in the presence of noise (Lockwood *et al*., 1991). In any case, the obtained *spectrum* is usually transformed into *cepstrum*. All these topics are developed in the following subsections.

2.3.1 Preprocessing of the Speech Signal

A first operation usually carried out on the speech signal is *pre-emphasis*, which consists of filtering the signal with a digital filter whose transfer function is (Markel and Gray, 1976)

$$P(z) = 1 - \mu z^{-1} \tag{2.4}$$

where $\mu \lesssim 1$ is a real factor. This filter approximately implements the derivative of the signal, and its objective is to remove possible DC components, as well as raising the high-frequency part of the spectrum, which has a 6 dB/decade decay for human speech on average.

After pre-emphasis, the signal is segmented into short segments (*frames*), typically 20–40 ms long and possibly overlapped. Within a frame, the signal is considered to be quasistationary, in such a way that the parameters that characterize the signal (*features*) can be considered constant within that frame.

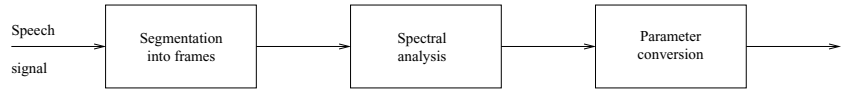


Figure 2.2 General scheme of the analysis stage

The sequence of frames is obtained by shifting an analysis window through the original signal. It is advisable to use a window other than the rectangular one to avoid *leakage*. A very common window in ASR systems is the *Hamming* window

$$w(n) = 0.54 - 0.46\cos\left[2\pi\frac{n-1}{L}\right] \tag{2.5}$$

where n is an integer index $0 < n \leq L$.

2.3.2 Linear Prediction Analysis

The LPC technique (from *linear prediction coding* (Makhoul, 1975)) provides an spectral description of short signal segments based on a speech production model which considers the speech signal $s(n)$ to be the response of an all-pole filter (representing the vocal tract) to an excitation $u(n)$. The transfer function of this filter is of the form

$$H(z) = \frac{\sigma}{1 + A(z)} \quad \text{with} \quad A(z) = \sum_{k=1}^{p} a_k z^{-k} \tag{2.6}$$

where a_k $(k = 1, \ldots, p)$ (usually referred to as *LPC coefficients*) and σ are the filter coefficients and the filter gain, respectively. The corresponding difference equation is

$$x(n) = \sigma u(n) - \sum_{k=1}^{p} a_k x(n-k) \tag{2.7}$$

This equation can also be interpreted as if we were using a linear predictor with a transfer function $(-A(z))$ and a prediction error $e(n) = \sigma u(n)$. More details about predictors and LPC coding can be found in Chapter 4.

Figure 2.3 shows the LPC speech production model. The excitation is modeled as follows:

1. If the speech signal corresponds to a voiced sound, the excitation consists of sequences of unit impulses separated by a constant number of sampling periods. This separation is called *fundamental period* or *pitch*, which corresponds to the inverse of the vibration frequency of the vocal cords.
2. If the signal corresponds to an unvoiced sound, the excitation is considered like a stationary white Gaussian noise with zero mean and variance equal to one.

The model parameters $(\sigma; a_k, k = 1, \ldots, p)$ can be obtained as those of an autoregressive (AR) model or, equivalently, as those of a linear predictor (only a_k's). The main resolution methods are the covariance and the autocorrelation methods. Both involve the resolution of a set of linear equations and are extensively treated in the speech processing literature ((Deller *et al*., 1993)). There also exist a number of algorithms for pitch estimation (Deller *et al*., 1993).

$H(e^{j\omega})$ is an AR representation of the spectrum of the signal, which is known as *LPC spectrum*. In Figure 2.4, the LPC spectrum of a segment corresponding to the Spanish vowel /e/ is presented along with its discrete Fourier transform (DFT) spectrum

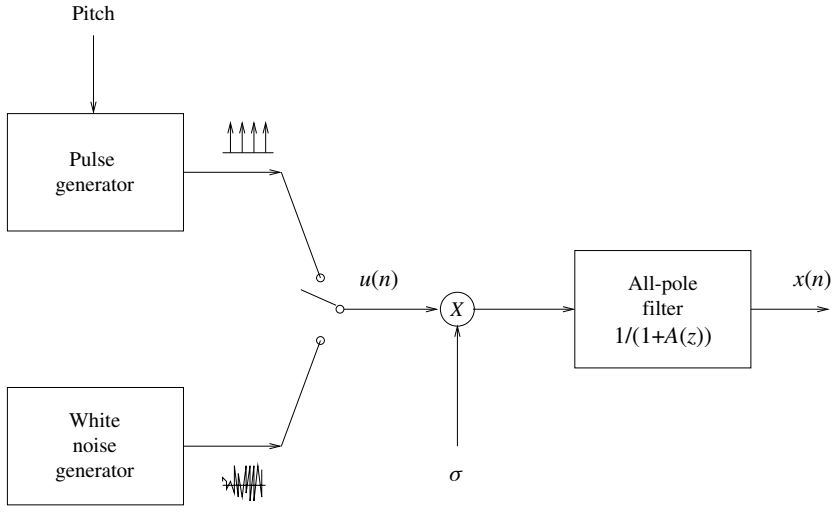


Figure 2.3 LPC model of speech production

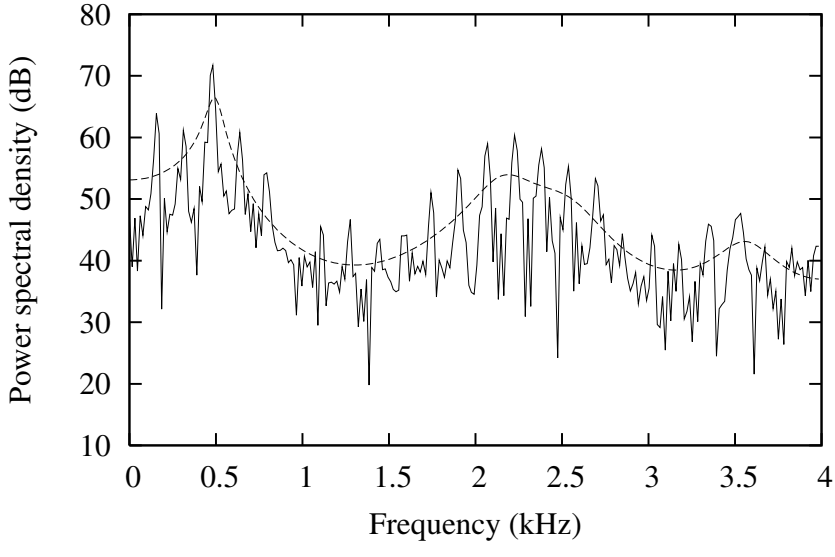


Figure 2.4 DFT (solid) and LPC (dashed) spectrum of a segment of the Spanish vowel /e/

(periodogram). The *formants* or resonance frequencies of the vocal tract are easily identified. The LPC order p is chosen high enough so that the formants and possible zeros can be clearly distinguished ($p = 10$ to 14 are typical values for a 4 kHz bandwidth; $p = 12$ in the figure). The figure also shows that the LPC spectrum provides an estimate of the short-term spectrum (envelope of the periodogram).

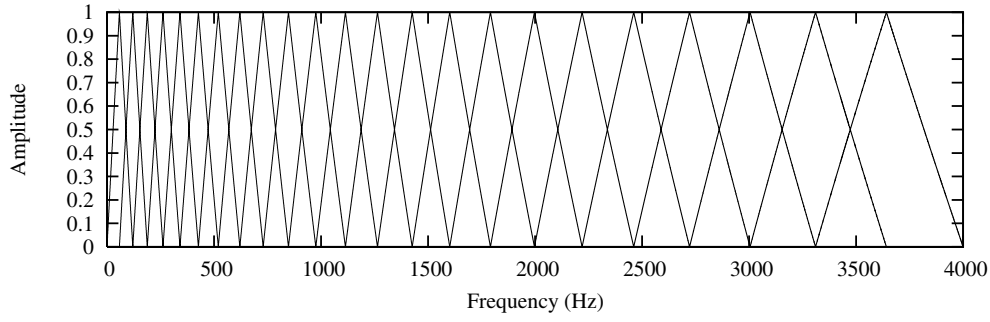


Figure 2.5 Frequency response of a typical mel-scaled triangular filterbank for signals sampled at 8 kHz

2.3.3 Mel-Frequency Filterbanks

The use of an analysis based on a mel-scaled triangular filterbank (Davis and Mermelstein, 1980; Rabiner and Juang, 1993) is extended in ASR nowadays. As previously mentioned, this success is mainly due to its better robustness than LPC against noise. Figure 2.5 shows the frequency response of such a filterbank. The mel scale is a nonlinear frequency transformation with the following expression:

$$g = 1127 \log \left[1 + f/700\right] \tag{2.8}$$

This nonlinear scaling provides a filterbank that imitates the critical band structure of the human ear. It has the effect of providing more resolution at low frequencies than at the higher ones (Zwicker and Fastl, 1990).

In order to apply this spectral analysis, it is necessary to first obtain the DFT $|X(i)|$ of the considered signal frame. Then, the filterbank outputs provide the following spectral representation:

$$H(k) = \sum_{i=ini(k)}^{end(k)} |X(i)|^{\alpha} W_k(i) \qquad (k = 1, \ldots, N) \tag{2.9}$$

where $H(k)$ and $W_k(i)$ are the output and the frequency response of the kth filter, respectively, $ini(k)$ and $end(k)$ are the frequency indices delimiting the pass band of that filter, N is the number of filters in the bank, and $\alpha = 1, 2$ (Young *et al.*, 2000).

2.3.4 Cepstral Coefficients

As indicated in Figure 2.2, the set of spectral coefficients is usually transformed to obtain an appropriate representation for recognition. The current analysis techniques tend to make use of a derivation known as *cepstrum* (Oppenheim and Schafer, 1975). By applying cepstral analysis, we obtain a representation more decorrelated (this is a desirable property when using HMMs, since it involves the use of diagonal covariance matrices) and with

a reduced dimensionality (which involves less computational burden). The cepstrum $c(n)$ is the time function obtained as the inverse transform of the logarithmic spectrum

$$\log X(\omega) = \sum_{n=-\infty}^{\infty} c(n) e^{j\omega n} \tag{2.10}$$

The samples of the cepstrum are usually known as *cepstral coefficients* and their domain is called *quefrency* (although they actually have the dimensionality of time). For a speech recognition task, only the first cepstral coefficients are kept. This is equivalent to applying a rectangular window to the cepstrum. In general, it is possible to apply other windows different from the rectangular one. This windowing operation is known as *liftering*, and its effect is a smoothing of the spectrum quite suitable for spectral comparison (Deller *et al*., 1993).

Depending on the spectrum representation we use, we will obtain different types of cepstra. The most direct way is to use the log-magnitude of the DFT transform, and then to obtain the cepstrum through the inverse DFT. The obtained features are known as *linear frequency cepstrum coefficients* (LFCC) (Davis and Mermelstein, 1980).

In the case of the LPC spectrum ($X(\omega) = H(e^{j\omega})$ in Equation (2.10)), the cepstral coefficients (LPCC) can be obtained (Davis and Mermelstein, 1980; Deller *et al*., 1993) by means of the following recursive expression

$$c(n) = \begin{cases} \log G & n = 0 \\ -a_1 & n = 1 \\ -a_n - \sum_{k=1}^{n-1} \dfrac{k}{n} c(k) a_{n-k} & 1 < n \leq L \end{cases} \tag{2.11}$$

In the case of using the mel-scaled filterbank previously described, we can obtain the so-called mel-frequency cepstral coefficients (MFCC) (Davis and Mermelstein, 1980). They are obtained by applying, as decorrelating transform, the discrete cosine transform (DCT) to the filterbank log outputs

$$c(n) = \sum_{k=1}^{N} \log H(k) \cos\left(\frac{\pi i}{N}(k - 0.5)\right) \qquad (n = 0, \ldots, L \leq N) \tag{2.12}$$

It is possible to define a *cepstral distance* between two spectra $X_r(\omega)$ and $X_t(\omega)$ as

$$d_C(X_t, X_r) = \int_{-\pi}^{\pi} \left|\log X_t(e^{j\omega}) - \log X_r(e^{j\omega})\right|^2 \frac{d\omega}{2\pi} = \sum_{n=-\infty}^{+\infty} (c_t(n) - c_r(n))^2 \tag{2.13}$$

The summation in the previous equation can be approximated by a finite sum over the L first cepstral coefficients (i.e. to apply a rectangular liftering), and thus the cepstral distance becomes a simple Euclidean distance in the space of the cepstral vectors $\mathbf{c} = (c(1), c(2), \ldots, c(L))$

$$d_C(\boldsymbol{c}_t, \boldsymbol{c}_r) \approx \sum_{n=1}^{L} (c_t(n) - c_r(n))^2 = \|\boldsymbol{c}_t - \boldsymbol{c}_r\|^2 \tag{2.14}$$

assuming that spectra are gain-normalized, that is, $c_r(0) = c_t(0) = 0$. In fact, the $0th$ order cepstral coefficient has a differentiated treatment since it is related to the frame energy and it is usually substituted by the log-energy (defined in the following subsection) in the feature vector. Therefore, the above cepstral distance is a measure of similarity between spectral shapes.

An alternative to the cepstral representation based on a logarithmic compression of the spectrum is to apply a root compression instead (Lim, 1979). Thus, the decorrelating transform is applied to $X(\omega)^\gamma$ (with $0 < \gamma \leq 1$). The result is known as *root-cepstrum*. Choosing an appropriate value for γ, it is possible to obtain a feature set more immune to noise (Alexandre and Lockwood, 1993; Sarikaya and Hansen, 2001). Applying root compression, it is possible to obtain root-LFCCs, root-LPCCs or root-MFCCs.

2.3.5 Other Features

The cepstral coefficients are usually referred to as *static* features, since they only contain information from a given frame. In order to enhance the frame representation, it is usual to introduce new features in the feature vector. Furui (Furui, 1981, 1986) suggested the use of *dynamic* features that introduce transitional information. In particular, Furui proposed the use of the first-order orthogonal polynomial coefficient obtained from the regression analysis of each nth order cepstral coefficient, considered like a time function ($c_t(n)$),

$$\Delta c_t(n) = \frac{\sum_{k=-K}^{k=K} w_k c_{t+k}(n)}{\sum_{k=-K}^{k=K} w_k^2} \qquad (n = 1, \ldots, L) \tag{2.15}$$

with $w_k = k$. These coefficients are an estimation of the time function derivative at time t. The vector $\Delta\mathbf{c} = (\Delta c_1(t), \Delta c_2(t), \ldots, \Delta c_L(t))$ is known as *delta cepstrum*. These new features also allow a Euclidean distance with the meaning of a distance measure between the log-spectral slopes $\partial \log X(\omega, t)/\partial t$.

Other possible features useful for recognition are the frame *log-energy* (Rabiner *et al.*, 1984)

$$E = 10 \log_{10} \left(\frac{1}{N} \sum_{n=0}^{N-1} x^2(n) \right) \tag{2.16}$$

which is usually normalized to the energy maximum in the utterance, and the *delta energy* (Furui, 1986; Peinado *et al.*, 1990), which can be computed using an expression similar to that of Equation (2.15).

The delta features are also known as *velocity* features. The second derivatives (*delta-delta* or *acceleration* features) can also help to improve the ASR system performance (Furui, 1981; Hanson and Applebaum, 1990). They can be obtained by either applying the derivative expression (2.15) over the velocity features or as the second-order orthogonal polynomial coefficients. In this last case, its employed expression is (2.15), again with

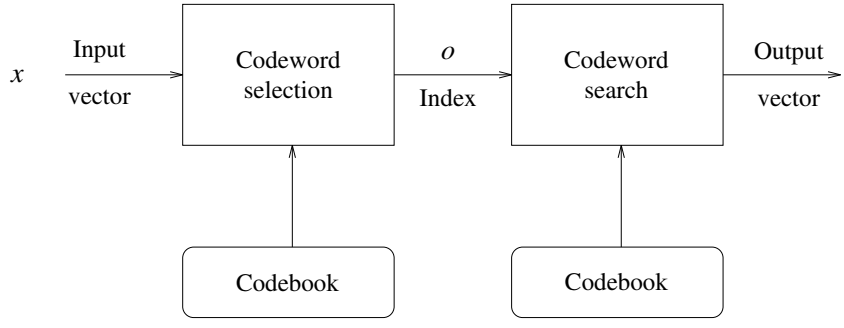


Figure 2.6 General scheme of a vector quantizer

the following weights (Furui, 1981)

$$w_k = (k + K + 1)^2 + 10(k + K + 1) + \frac{55}{3} \qquad (k = -K, \ldots, +K) \tag{2.17}$$

2.4 Vector Quantization

VQ was first applied as a technique for speech coding in 1980 (Buzo *et al.*, 1980). Later, it was applied to speech recognition in a number of ways. The basic idea is to replace a given input vector, obtained from a previous analysis of a signal frame, with a similar vector, which is called *codevector* and belongs to a finite set called *codebook*. Each codevector has an associated index (*codeword*) which is transmitted instead of the original vector, considerably reducing the amount of data to be transmitted (see Figure 2.6). From this point of view, VQ can be considered a form of pattern recognition, in which the "input object" is replaced with the "recognized object." Also, VQ can be considered a generalization of scalar quantization for multidimensional spaces. A thorough study of VQ applied to speech coding can be found in Gray (1984) and Gersho and Gray (1991).

A vector quantizer Q can be considered a mapping of the p-dimensional space $\mathbb{R}^p$ into a finite set of vectors C (codebook)

$$Q : \mathcal{R}^p \longrightarrow C \tag{2.18}$$

with $C = \{\mathbf{y}_i \in \mathbb{R}^p, i = 1, \ldots, N\}$. The quantizer Q has an associated partition in cells S_i, such that $S_i = Q^{-1}(\boldsymbol{y}_i)$. Actually, the quantizer function Q, as in Figure 2.6, can be broken down into two functions: $E : \mathbb{R}^p \longrightarrow J \subset \mathbb{N}$ for coding and $D : J \longrightarrow C$ for decoding, in such a way that $Q(\mathbf{x}) = D(E(\mathbf{x}))$.

The VQ process requires the definition of a distance measure between vectors, which will be generically noted as $d(\mathbf{x}, \mathbf{y})$, indicating the distortion introduced when replacing a vector $\mathbf{x}$ with $\mathbf{y}$. The accuracy of a quantizer can be measured by means of the expected distortion $E\,[d(\mathbf{x}, Q(\mathbf{x}))]$.

Given a codebook C, the optimal form of the function Q, in the sense of minimal average distortion, is given by the *nearest neighbor rule*

$$Q(\mathbf{x}) = \mathbf{y}_i \quad \text{if} \quad d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j) \quad \forall j \neq i \tag{2.19}$$

This rule involves the following partition:

$$S_i = \left\{ \mathbf{x} \in \mathbb{R}^p \ : \ d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j) \ \ \forall j \neq i \right\} \tag{2.20}$$

On the other hand, given a partition, the optimum codebook (in the sense of minimum average distortion) is the one in which the codevectors are the *centroids* of the cells S_i of the partition, defined as

$$\mathbf{y}_i \equiv cent(S_i) = \min_{\mathbf{y}}{}^{-1} E\left[d(\mathbf{x}, \mathbf{y}) | \mathbf{x} \in S_i \right] \tag{2.21}$$

This expression is known as *centroid condition* (Gersho and Gray, 1991). In particular, using Euclidean distances, each centroid $\mathbf{y}_i$ is the mean vector of cell S_i

$$\mathbf{y}_i = cent(S_i) = E\left[\mathbf{x} | \mathbf{x} \in S_i \right] \tag{2.22}$$

The most important problem in VQ is the design of the optimal codebook C given a set T of empirical data (set of training vectors). The most extended method is the *generalized Lloyd algorithm*, commonly known as *algorithm LBG* (Linde-Buzo-Gray (Linde *et al.*, 1980)) (also *k-means*), which consists of four basic steps:

1. An initial codebook C_1 is built. Set $m = 1$.
2. Given a codebook $C_m = \{\mathbf{y}_i\}$, a partition of the data set T is performed in clusters S_i, by using the nearest neighbor rule, in such a way that

$$S_i = \left\{ \mathbf{x} \in T \ : \ d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j) \ \ \forall j \neq i \right\}$$

3. Centroids are computed by applying the centroid condition (Equation (2.21)). Thus, a new codebook C_{m+1} is obtained. In case of an empty cell, an alternative assignment (instead of computing its centroid) is carried out.
4. Mean distortion for C_{m+1} is computed. If the change of distortion is too small the algorithm is concluded. Otherwise, we go to step 2 with $m \leftarrow m + 1$.

It can be easily proved that this algorithm continuously reduces the average distortion and that it converges in a finite number of iterations (Gersho and Gray, 1991). In the particular case of monodimensional vectors, LBG becomes the Lloyd algorithm.

There exist several alternatives for the selection of the initial codebook. For example, it can be randomly set or extracted from the training set. A more elaborate possibility is the iterative splitting of the training set (Gray, 1984).

The LBG algorithm only leads to a locally optimal solution. There exist methods for searching globally optimal solutions such as, for example, the *Simulated Annealing* algorithm (Vaisey and Gersho, 1988).

An important issue that we must to take into account is the computational cost involved by the quantization process. The nearest neighbor rule of Equation (2.19) involves a full search in the whole codebook. When the codebook size is large, the quantization can be very time consuming. The codebook search can be speeded up using a tree-structured VQ (TSVQ) search (Gersho and Gray, 1992). In an m-ary search, the quantization is carried out in several steps. At a given step, the input vector is compared with m different centroids. The chosen centroid defines which set of m different centroids is to going

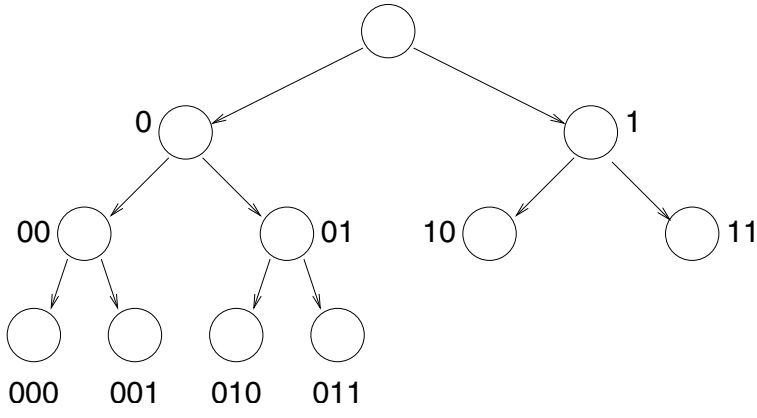


Figure 2.7 Structure of a variable length binary TSVQ quantizer

to be used at the following step. The search is much faster than a full search since an N-centroid codebook only requires the computation of only $m \log_m N$ distances. In Figure 2.7 is depicted the tree structure of a binary TSVQ quantizer with the corresponding bit allocation. An interesting property of TSVQ is that the resulting codewords can have a variable length. This allows TSVQ to provide an even better performance than a full search VQ since we can assign fewer bits to the less important centroids.

2.5 Approaches to ASR

The different techniques proposed to solve the recognition problem are usually grouped into three categories (in historic order): pattern matching, statistical models and neural networks. The first two groups are the ones that have been used more extensively. Pattern matching was the first approach to the ASR problem, and was preponderant during the sixties and seventies, and is sometimes still used to solve some specific issues (e.g. to compute the WER). However, the statistical approach, based on the use of HMMs, was progressively substituting pattern matching during the eighties owing to power and flexibility provided by the statistical tools. We will briefly summarize the pattern matching approach in the following subsection, although we will devote the rest of the chapter to the statistical approach and the implementation of ASR systems using HMMs.

2.5.1 Pattern Matching

Pattern matching is a geometrical approach based on measuring distances between a test input sequence and a reference sequence. Its main problem is how to measure distances between sequences of different lengths, as it is the normal case in speech recognition. To solve this problem, it is necessary to align the input object with the reference object. This alignment can be achieved through *dynamic programming* techniques (Bellman, 1957), which were first applied to speech recognition by Vintsjuk (Vintsjuk, 1968). The method is usually known as dynamic time warping (DTW) and works as follows: A matrix $d(i, j)$ $(i = 1, \ldots, I;\ j = 1, \ldots, J)$ with all possible distances between the vectors of the input

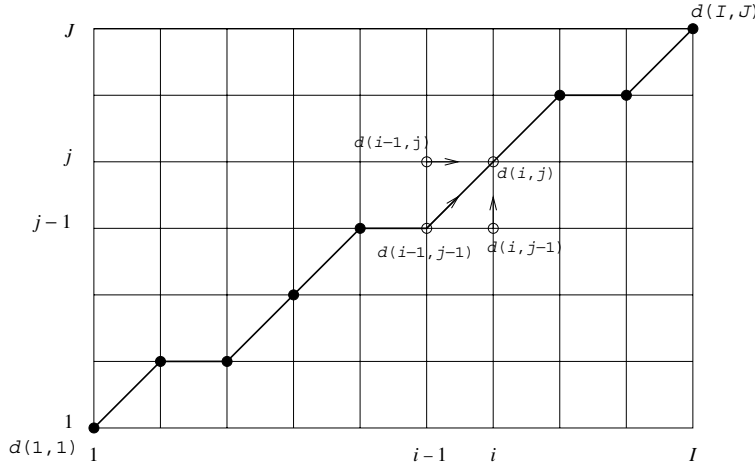


Figure 2.8 DTW algorithm

object (with I frames) and the reference object (with J frames) is built first. The matrix is depicted in Figure 2.8. Then, an alignment is carried out, that is, the search of the optimal path in matrix $d(i, j)$, by means of an accumulated distance computed as

$$g(i,j) = min \begin{cases} g(i-1,j) + d(i,j) & \text{frame insertion} \\ g(i-1,j-1) + 2d(i,j) & \text{normal progression} \\ g(i,j-1) + d(i,j) & \text{frame deletion} \end{cases} \tag{2.23}$$

Then, the distance between both the objects is

$$D = \frac{g(I,J)}{(I+J)} \tag{2.24}$$

This distance value can be used to classify an input object by comparing it with each one of the reference patterns representing the possible utterances, and selecting the one which provides the minimum distance. The process not only provides a distance between two objects, but also the optimal path (correspondence between particular frames), as shown in Figure 2.8. This can be used to train the system, that is, to obtain each one of the reference patterns from a set of training utterances by applying a clustering technique such as the k-means method (Tou and Gonzalez, 1974).

2.5.2 The Statistical Approach

A classical problem in physics and engineering is that of getting a *model* for a given real-world process. *Modeling* techniques are useful tools in prediction, recognition or identification tasks. The application of models to signals is important for a number of reasons. First, a signal model can help process that signal (e.g. to clean a noisy signal). Also, a model can help the understanding of the signal source and the signal generation process.

In particular, statistical models have been successfully applied to speech recognition because they allow setting the recognition problem in statistical terms. Let us suppose that

$\mathcal{W} = \{W_i\}$ is the set of possible sentences of a given language and that we wish to obtain the sentence $W(X)$ corresponding to an acoustic evidence X. By applying the maximum a posteriori (MAP) decision rule, the recognized sentence is obtained as

$$W(X) = \underset{j}{\operatorname{argmax}}\, P(W_j|X) \tag{2.25}$$

This maximization requires the computation of the probabilities $P(W|X)$. The classical approach is the decomposition of $P(W|X)$ by the Bayes rule

$$P(W|X) = \frac{P(X|W)P(W)}{P(X)} \tag{2.26}$$

Since $P(X)$ does not depend on X, the MAP rule can be written as

$$W(X) = \underset{j}{\operatorname{argmax}} \left[P(X|W_j)P(W_j)\right] \tag{2.27}$$

The conditional probability $P(X|W)$ is given by the *acoustic model* and $P(W)$ by the *language model*. In the next sections we will describe the acoustic modeling of signals by HMMs and its application to ASR. Although the basic theory of HMMs is well known since the sixties, it was not applied to ASR till the seventies and widely applied by the research community during the eighties. It is possible to find a large number of papers dealing with the fundamentals of HMMs and their application to ASR (Levinson *et al.*, 1983; Rabiner, 1989; Rabiner *et al.*, 1983). A brief description of language models is also given in this chapter.

2.6 Hidden Markov Models

2.6.1 Markov Processes and Hidden Markov Processes

The HMM is obtained as a generalization of a Markov process, which we define next. Let us assume a process (see Figure 2.9) described by a set of N states $\{s_1, s_2, \ldots, s_N\}$. Each state represents a certain event or observation. The system changes from one state to another (transition) in each time interval. We will call q_t the state at time t.

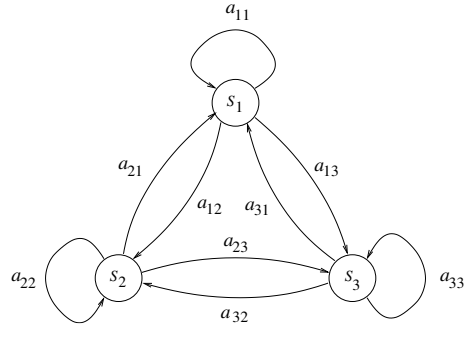


Figure 2.9 A Discrete Markov process

The Markov processes (or chains) are characterized by the dependence of the current state with respect to previous states. In other words, the process has "memory." In the case of a discrete stationary *first-order* Markov process, the current state only depends on the previous state, independently of the considered time. This process is fully described by the *transition probabilities* from one state to another:

$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i) \tag{2.28}$$

The following example can help understand the dynamics of such types of Markov processes. Let us suppose a model for weather forecast with only three possible states (observations): rainy (s_1), cloudy (s_2) and sunny (s_3). This model (see Figure 2.9) allows us to calculate the probability of observing any sequence of weather states. For example, we can calculate the probability of the "sunny/sunny/rainy/cloudy" sequence ($O = (s_3, s_3, s_1, s_2)$) as

$$P(O|\text{model}) = P(s_3 s_3 s_1 s_2|\text{model}) \tag{2.29}$$

$$= P(s_3)P(s_3|s_3)P(s_1|s_3)P(s_2|s_1) = P(s_3)a_{33}a_{31}a_{12} \tag{2.30}$$

Hidden Markov processes arise as a generalization of the Markov processes studied above. A classic example (Rabiner and Juang, 1993) to illustrate this is the coin tossing experiment: a certain person, who we call host, is going to perform the experiment of tossing coins, from which a sequence of heads (he) and tails (ta) is obtained. The host is hidden to another person (observer), and this one only knows the result of the experiment (i.e. the sequence of heads and tails) without knowledge of how it was performed. The host has three coins. Two of them are used to obtain the sequence of heads and tails. These two coins are rigged, so that the first coin has a 75 % probability of getting heads and the second one has a 75 % probability of getting tails. The third coin is not rigged and is used to decide at each time which of the other coins will be tossed. It is possible to build a model of the experiment (Figure 2.10) with two states (each one representing one of the coins to be tossed). In each state it is possible to generate "he" or "ta" (observations) according to the probabilities shown in the figure. We have just built an

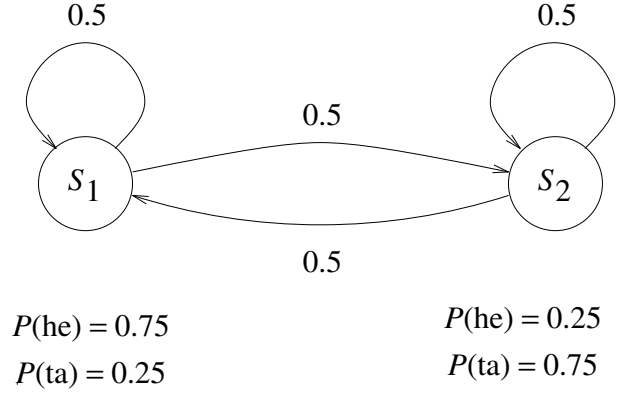


Figure 2.10 HMM for the coin toss experiment

HMM. The difference with a plain Markov model is that we have now two superposed processes. One of them is observable (the sequence of observations, heads and tails), but the other one is "hidden" (sequence of states, tossed coins). In the same way as for plain Markov processes, it is possible to calculate the probability of a sequence of observations $O = o_1, \ldots, o_T$ of length T generated by an HMM, although we will formally define them before the concept of HMM.

2.6.2 Definition of a Discrete HMM

A discrete HMM is characterized by the following elements:

1. A set S with N states, $S = \{s_1, s_2, \ldots, s_N\}$ interconnected by means of arcs, so that at each time t the model is in a certain state, which will be denoted as q_t;
2. A set V with M observation symbols, $V = \{v_1, v_2, \ldots, v_M\}$. At each time t, the model generates one symbol, that will be noted as o_t;
3. A transition matrix $A = \{a_{ij}\}$ containing the transition probabilities between states, which are defined as follows:

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i) \qquad (i, j = 1, \ldots, N) \tag{2.31}$$

   These probabilities must verify

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad (i = 1, \ldots, N) \tag{2.32}$$

4. An output or observation probability matrix $B = \{b_i(v_k)\}$, in which each element is the probability of generation of a certain symbol in a certain state

$$b_i(v_k) = P(o_t = v_k | q_t = s_i) \qquad (i = 1, \ldots, N; k = 1, \ldots, M) \tag{2.33}$$

   These probabilities must verify

$$\sum_{k=1}^{M} b_i(v_k) = 1 \qquad (i = 1, \ldots, N) \tag{2.34}$$

5. A matrix Π of initial states, in which each element is the probability of having a certain state as initial state

$$\Pi = \{\pi_i\} \quad \text{with} \quad \pi_i = P(q_1 = s_i) \qquad (i = 1, \ldots, N) \tag{2.35}$$

   These probabilities must verify

$$\sum_{i=1}^{N} \pi_i = 1 \tag{2.36}$$

As can be seen, the model is fully defined by matrices A, B y Π. Thus, from now on, the model will be noted as the set of parameters $\lambda = (A, B, \Pi)$.

As previously mentioned, the model generates a sequence of observations $O = (o_1, o_2, \ldots, o_T)$ and a hidden sequence of states $Q = (q_1, q_2, \ldots, q_T)$, which we will call *path*.

2.6.3 The Three Basic Problems

It is very common (Lee, 1989; Rabiner, 1989) to group the different problems associated with HMMs into three classes. These are:

1. Evaluation problem: Given an observation sequence O, find the probability $P(O|\lambda)$ of sequence O being generated by model λ.
2. Optimum path determination problem: Given a sequence O and model λ, find the optimal path Q.
3. Estimation problem: Given an observation sequence O, find the set of parameters λ that better fit that sequence.

The first problem is the basic problem of recognition. A given input sequence O must be evaluated by the recognition system for each of the models making up the system (each one representing a recognition class), so that the model λ for which the probability $P(O|\lambda)$ is maximum will correspond to the recognized class. The second problem, also known as the *decoding* problem, is the recovery of the hidden state sequence Q. Unfortunately, there is no unique answer, and, therefore, an optimization criterion must be chosen as we will see later in this section. The decoding problem has its main application in CSR, since their solutions help to alleviate the problem of recognizing an enormous number of classes (sentences). The last problem is the estimation of the model parameters. In order to build (or train) a speech recognition system, the parameters of the different models must be estimated from experimental data. Again, there is no unique answer to the problem of finding the optimal set of parameters, although in any case, the goal must be to obtain the maximum system performance.

2.6.3.1 Solution to the Evaluation Problem

It was previously established that a signal model allows the computation of the probability $P(O|\lambda)$ that a given input sequence $O = (o_1, o_2, \ldots, o_T)$ is generated by the model λ. The most direct way of doing this is to first evaluate the sequence for a given path

$$P(O|Q, \lambda) = b_{q_1}(o_1)b_{q_2}(o_2)\ldots b_{q_T}(o_T) \tag{2.37}$$

and the probability of such path

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \tag{2.38}$$

Summing the joint probabilities of sequence and path $P(O, Q|\lambda)$ (this is the product of the previous ones) for all the possible paths, we can finally obtain

$$P(O|\lambda) = \sum_{Q} P(O, Q|\lambda) = \sum_{Q} P(O|Q, \lambda)P(Q|\lambda) \tag{2.39}$$

$$= \sum_{Q} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) a_{q_2 q_3} \ldots a_{q_{T-1} q_T} b_{q_T}(o_T) \tag{2.40}$$

Since there are N^T possible paths, and each path requires $2T - 1$ multiplications, a total of $N^T - 1$ sums and $(2T - 1)N^T$ multiplications are required, that is, around

$2TN^T$ operations by using Equation (2.40). For a typical application to speech with $T = 50$ and $N = 5$, around 10^{37} operations would be required, which is almost impossible to perform. Fortunately, there exists an algorithm, known as *forward-backward*, which greatly simplifies the computation. It is based on the recursive computation of a forward probability $\alpha_t(i)$, defined as

$$\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = s_i|\lambda) \tag{2.41}$$

that is, the probability that the generated sequence up to time t is $o_1 o_2 \dots o_t$ and the state at that time is s_i, given the model λ. The procedure is as follows:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1) \qquad (1 \le i \le N) \tag{2.42}$$

2. Recursion:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \qquad (1 \le j \le N; 1 \le t \le T-1) \tag{2.43}$$

3. End:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.44}$$

For the computation of (2.44), $(N-1)N(T-1)$ sums and $N(N+1)(T-1)+N$ multiplications are required, that is, about $2N^2T$ operations in total. For the case $N = 5$ and $T = 50$, the total number of computations is 2500 operations. This means an approximate complexity reduction of 10^{33} with respect to the direct method.

It is also possible to solve the problem using a backward probability $\beta_t(i)$ defined as

$$\beta_t(i) = P(o_{t+1} o_{t+2} \dots o_T|q_t = s_i, \lambda) \tag{2.45}$$

that is, the probability of having sequence O from time $t+1$, with current state s_i for model λ. In the same way as for the forward probabilities, there is a recursive method for their computation:

1. Initialization:

$$\beta_T(i) = 1 \qquad (1 \le i \le N) \tag{2.46}$$

2. Recursion:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) b_j(o_{t+1}) \qquad (1 \le j \le N; t = T-1, T-2, \dots, 1) \tag{2.47}$$

3. End:

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i) \tag{2.48}$$

The computational complexity is similar to that obtained with forward probabilities. The forward–backward algorithm can easily generate underflow problems, so some type of scaling may be needed.

2.6.3.2 Solution to the Optimal Path Determination Problem

As was already pointed out, the difficulty is to define the criterion that the optimal path must accomplish. One possibility is to choose the most probable state at each time. In order to do this, the following variable can be defined:

$$\gamma_t(i) = P(q_t = s_i|O, \lambda) \tag{2.49}$$

This is the probability of model λ in state s_i at time t, during the generation of sequence O, and can be expressed as a function of the forward–backward probabilities as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \tag{2.50}$$

from which the optimal path Q^* can be determined as

$$q_t^* = \max_{1 \le i \le N}{}^{-1} \left[\gamma_t(i)\right] \qquad (1 \le t \le T) \tag{2.51}$$

A little thought on this method will show that we have a sequence of locally optimal states, that is, we determine q_t^* at each time, regardless of whether the resultant sequence is globally optimal or even whether it is a possible sequence. Thus, it seems more suitable to find the path Q^* that maximizes the probability $P(Q|O, \lambda)$ or, equivalently, $P(Q, O|\lambda)$, since

$$P(Q, O|\lambda) = P(Q|O, \lambda)P(O|\lambda) \tag{2.52}$$

and $P(O|\lambda)$ does not depend on the path. Therefore, we can write

$$Q^* = \operatorname*{argmax}_{Q} P(Q|O, \lambda) = \operatorname*{argmax}_{Q} P(Q, O|\lambda) \tag{2.53}$$

This can be solved by means of the well-known *Viterbi algorithm* (VA). VA is a recursive procedure, similar to the forward–backward procedure. Let us define the following function:

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_2 \ldots o_t|\lambda) \tag{2.54}$$

This function is given by the best sequence $q_1 q_2 \ldots q_{t-1}$ that make possible the generation of $o_1 o_2 \ldots o_t$, s_i being the current state ($q_t = s_i$). The algorithm uses an auxiliary function $\phi_t(j)$ that allows the recovery of the optimal path when the recursion finishes. The procedure is as follows:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(o_1) \qquad (1 \leq i \leq N) \tag{2.55}$$

$$\phi_1(i) = 0 \tag{2.56}$$

2. Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t) \qquad (1 \leq j \leq N; 2 \leq t \leq T) \tag{2.57}$$

$$\phi_t(j) = \max_{1 \leq i \leq N}{}^{-1} \left[\delta_{t-1}(i) a_{ij}\right] \qquad (1 \leq j \leq N; 2 \leq t \leq T) \tag{2.58}$$

3. End:

$$P^* = P(Q^*, O|\lambda) = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{2.59}$$

$$q_T^* = \max_{1 \leq i \leq N}{}^{-1} [\delta_T(i)] \tag{2.60}$$

4. Backtracking of Q^*:

$$q_t^* = \phi_{t+1}(q_{t+1}^*) \qquad (T-1, T-2, \ldots, 1) \tag{2.61}$$

The algorithm usually requires scaling to avoid underflows in the same way as the forward–backward algorithm. The similarity between the Viterbi and the forward–backward algorithms can also be observed. The computational complexities are also similar (substituting sums by maximum decisions). This number of computations can be unmanageable in the case of CSR systems with some degree of complexity. In this case, it is necessary to limit the number of candidate states when the algorithm progresses by means of the imposition of a threshold to the state probabilities (Schwartz *et al*., 1985).

Another solution for the decoding problem is the *stack decoding* algorithm (Bahl *et al*., 1983). Unlike the VA algorithm, this is not a time-synchronous search and it is based on a best-first strategy.

2.6.3.3 The Estimation Problem

We have to now obtain the model parameter set λ that better fits a given training utterance O. The maximum likelihood (ML) estimation criterion provides the following estimate:

$$\hat{\lambda} = \underset{\lambda}{\text{argmax}}\, P(O|\lambda) \tag{2.62}$$

This estimation is widely applied in HMM-based systems since there exists an efficient algorithm, the *Baum–Welch* algorithm, which implements it. It is based on the fact that the $P(O|\lambda)$ can be expressed as

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i) \tag{2.63}$$

Therefore, it requires the use of the forward–backward algorithm to compute the forward and backward probabilities. Let us define now the following probabilities:

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \tag{2.64}$$

$$\gamma_t(i) = P(q_t = s_i | O, \lambda) = \sum_{j=1}^{N} \xi_t(i,j) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{2.65}$$

Considering that $\sum_{t=1}^{T} \xi_t(i,j)$ is the expected number of transitions from s_i to s_j, and that $\sum_{t=1}^{T} \gamma_t(i)$ is the expected number of transitions from s_i, we can obtain the following reestimation formulae:

$$\hat{\pi}_i = \gamma_1(i) \tag{2.66}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.67}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^{T} \gamma_t(j) \delta_{o_t, v_k}}{\sum_{t=1}^{T} \gamma_t(j)} \tag{2.68}$$

where $\delta_{x,y}$ is the Kronecker delta function. These reestimation equations are straightforwardly extended to multiple training sequences by extending the sums in the above formulae to all those sequences. It is possible to show that if we apply equations (2.66-2.67) iteratively, the probability $P(O|\lambda)$ is increased at each step, leading, at least, to a local maximum of $P(O|\lambda)$. The success of this process mainly depends on the models used for the initialization of the reestimation process.

The ML estimation is widely applied in HMM-based systems because of its simplicity, the existence of an efficient algorithm that implements it and the excellent performance provided. Besides, Nadas (Nadas, 1983) showed that, under several assumptions, the ML estimation leads to an optimal classifier. However, these assumptions (i.e. that the true model generating the signal is known) are not satisfied in our problem. There are other estimation methods that try to minimize the system ER, which, in fact, is the goal of speech recognition. Among these methods, we find the maximum mutual information (MMI) estimation (Bahl *et al.*, 1986), the minimum discrimination information (MDI) estimation (Ephraim *et al.*, 1989) or the minimum classification error (MCE) estimation (Juang and Katagiri, 1992). In general, these methods are better than ML. As drawbacks, we can mention that they have a higher complexity and that their improvement is not always worthwhile, since more complex and accurate models can be used as the computational power is increased.

2.6.4 Generalization and Types of HMM Modeling

So far, we have assumed that the sequence to be recognized is a sequence of observations $O = o_1, \ldots, o_T$ that belong to a discrete set ($o_t \in V$). In the case of an ASR system, this observation sequence can be obtained from a sequence of feature vectors $X = \mathbf{x}_1, \ldots, \mathbf{x}_T$ submitted to a VQ process. However, the VQ process involves a loss of information that can reduce the system performance. This problem can be solved by forcing the HMM to work directly with the vectors $\mathbf{x} = (x(1), \ldots, x(p))$ of the continuous p-dimensional representation space $\mathbb{R}^p$. To do that, the output probabilities $b_i(v_k)$ must be substituted by probability density functions (pdf) $b_i(\mathbf{x}) = P(\mathbf{x}|s_i)$, which must accomplish the following normalization condition:

$$\int_{\mathbb{R}^p} b_i(\mathbf{x}) d\mathbf{x} = 1 \tag{2.69}$$

The most general representation of a pdf for which it is possible to find a suitable estimation procedure is a mixture

$$b_i(\mathbf{x}) = P(\mathbf{x}|s_i) = \sum_{v_k \in V(s_i)} c_{ik} P(\mathbf{x}|v_k, s_i) \tag{2.70}$$

where each $P(\mathbf{x}|v_k, s_i)$ is a log-concave or an elliptically symmetric pdf and $c_{ik} = P(v_k|s_i)$ is the pdf weight (the sum of the pdf weights over k must be 1). Each pdf of the mixture has been labeled with a symbol v_k ($k = 1, \ldots, M$) belonging to a finite set $V(s_i)$ defined for each state s_i. A multivariated Gaussian pdf $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{i,k}, \Sigma_{i,k})$, with mean vector $\boldsymbol{\mu}_{i,k}$ and covariance matrix $\Sigma_{i,k}$

$$b_i(\mathbf{x}) = P(\mathbf{x}|s_i) = \sum_{k=1}^{M} c_{ik} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{i,k}, \Sigma_{i,k}) \tag{2.71}$$

is frequently used to build the mixture. An AR Gaussian pdf has also been used (Juang *et al.*, 1985).

We have just defined a *continuous* HMM (CHMM). Details about this type of model can be found in the references (Juang *et al.*, 1985; Rabiner *et al.*, 1985a,b). The recognition task is now based on the computation of densities $P(X|\lambda)$. The answers to the three basic problems remain the same.

2.6.5 Simplifications to Continuous HMM Modeling

The computational complexity and the large number of parameters to estimate the continuous models justify the search for simplified models. First, let us modify our notation by introducing the specific model λ being considered

$$b_i(\mathbf{x}) = P(\mathbf{x}|s_i, \lambda) = \sum_{v_k \in V(s_i, \lambda)} P(\mathbf{x}|v_k, s_i, \lambda) P(v_k|s_i, \lambda) \tag{2.72}$$

This notation reflects that our recognition system consists of a set of different HMMs.

A first simplification is obtained forcing the different states of all the models to share the same set of pdfs, that is

$$V(s_i, \lambda) = V \qquad \text{for all} \qquad s_i, \lambda \tag{2.73}$$

and, therefore

$$b_i(\mathbf{x}) = P(\mathbf{x}|s_i, \lambda) = \sum_{v_k \in V} P(\mathbf{x}|v_k) P(v_k|s_i, \lambda) \tag{2.74}$$

This new type of modeling is known as *semicontinuous* HMM (SCHMM) (Huang *et al.*, 1990). The sum (2.74) is in practice only extended to the C most probable elements of V (best candidates for $\mathbf{x}$).

A special case of the SCHMM modeling is that in which only the best candidate o for an input vector $\mathbf{x}$ in (2.74) is kept. This is the same as considering that there is no overlapping among the different pdfs of V. This assumption yields the following approach:

$$P(\mathbf{x}|s_i, \lambda) = P(\mathbf{x}|o) P(o|s_i, \lambda) \tag{2.75}$$

$$o = \max_{v_j \in V}{}^{-1} \left[P(\mathbf{x}|v_j) \right] \tag{2.76}$$

The maximization of (2.76) is usually simplified to the search of the nearest neighbor center in a VQ codebook. It is easy to prove that the density $P(X|\lambda)$ can be decomposed as

$$P(X|\lambda) = P(X|O) P(O|\lambda) \tag{2.77}$$

Since $P(O|X)$ is independent of the model under consideration, it is useless to the recognition process and can be obviated. As a result, the SCHMM has become a DHMM (studied in previous subsections). The main feature of a DHMM-based system is the reduction of computational complexity in training and testing due to the above probability decomposition. The training has been divided into two separated stages: a VQ codebook generation and a DHMM training.

2.7 Application of HMMs to Speech Recognition

HMMs have been applied to ASR in multiple ways, as acoustic models for the recognition units (each HMM represents a different recognition unit). A simple example of an ASR system is shown in Figure 2.11. It shows a basic IWR system in which each word W_i is associated with an HMM λ_i.

An important topic when applying HMM to speech recognition is the topology of the model. The most general topology is the *ergodic* one, which ensures that there always exist a path between two different states. The HMM of Figure 2.9 is an ergodic model. However, the most extended one in speech recognition is the *left-to-right* topology, shown in Figure 2.12, since it suitably models the sequential nature of speech. The consecutive states represent the consecutive speech events in a given utterance. This model includes two null states (marked as I and F). A null state does emit an observation (i.e. it does not consume a time unit). Thus, null states provide a smart way of modeling the beginning

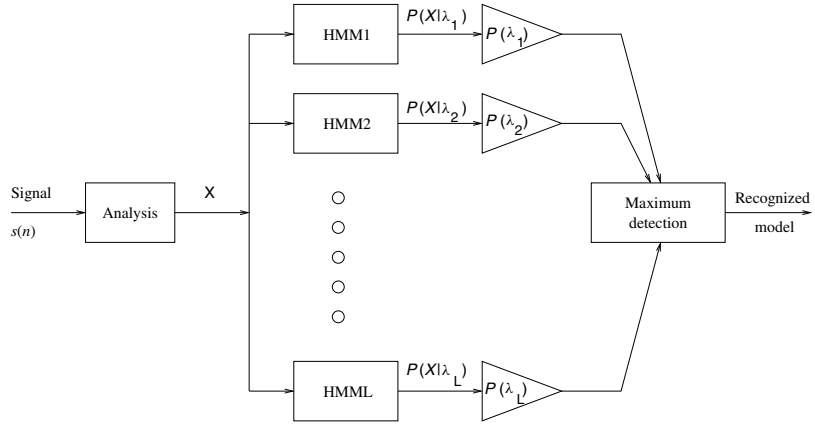


Figure 2.11 Isolated word recognition system based on HMMs

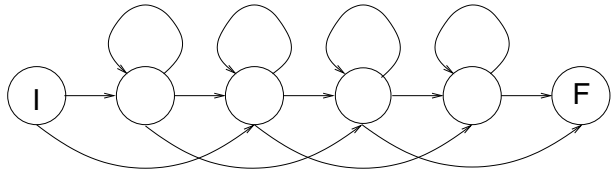


Figure 2.12 Left-to-right topology

and end of a sequence, avoid the need of the initial state probabilities π_i (it is now a transition probability) and are useful to concatenate HMMs in CSR systems as shown later. As mentioned previously in this chapter, an important issue for obtaining a good performance is the initialization of the reestimation procedure. In the case of left-to-right models, it is possible to perform a linear segmentation of the training data (each segment corresponding to a model state) to estimate the initial models (Rabiner and Juang, 1993).

When the recognition units are context-independent phones or triphones, it is quite common to use a left-to-right topology with three non-null states representing an initial transition, a stationary part and a final transition. Some researchers Lee (1989) prefer a phoneme model as that shown in Figure 2.13 in order to improve the phoneme duration modeling.

A CSR system could be built in the same manner as an IWR system, using a specific HMM for each possible sentence allowed by grammar. These models could be obtained either by training them from samples of their corresponding sentences or by concatenating word or subword models previously trained. Thus, recognition would be based on the maximization of the posterior probability of the sentence W given the acoustic observations X, as established in Equation (2.25). This maximization could be carried out as indicated in Equation (2.27), where $P(W)$ is provided by a language model and $P(X|W)$ by an acoustic model. However, the number of possible sentences is normally so large that this solution is impractical. In practice, the approach commonly applied is based on building a single "macromodel" that includes all the sentences

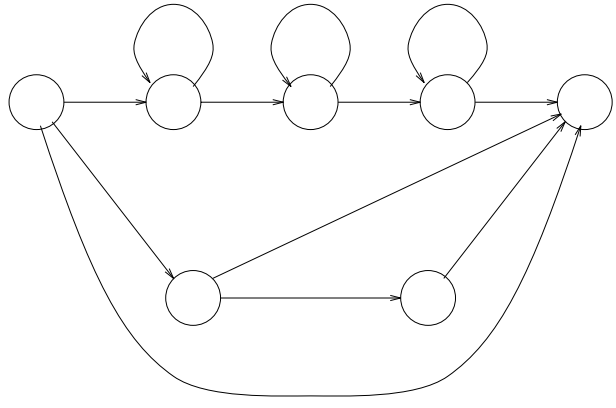


Figure 2.13 Phoneme HMM

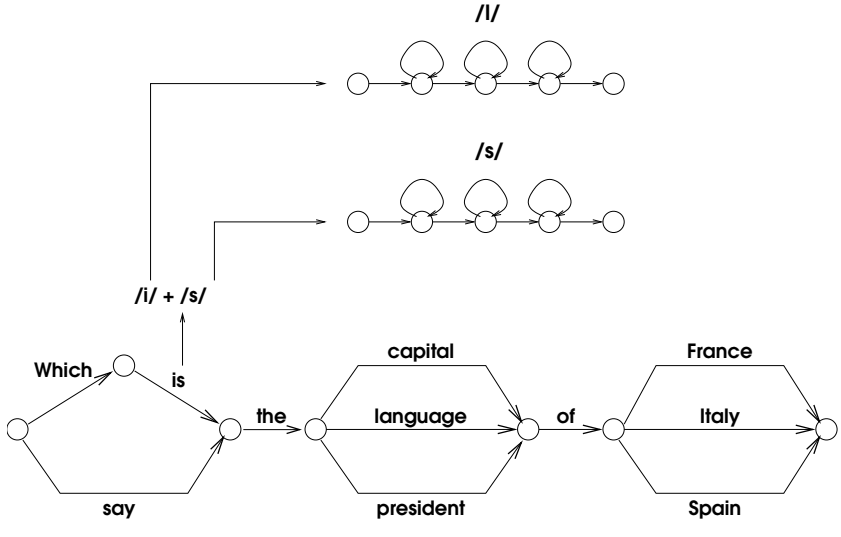


Figure 2.14 Continuous speech recognition system based on HMMs

allowed by grammar. This is illustrated in Figure 2.14 with a very simple grammar using context-independent left-to-right phoneme models. This macromodel was obtained by concatenating the corresponding phoneme models (according to grammar) through new null states. The macromodel can also be obtained using other units such as words or triphonemes (deciding this depends mainly on the particular recognition task). The recognized sentence is the one that corresponds to the optimal path obtained by means of the VA algorithm.

A CSR system can be trained using the Baum–Welch algorithm in a manner very similar to an IWR system by building for each training sentence a sentence model and concatenating the corresponding sequence of recognition units. In order to do this, it is only required to have knowledge of the sequence of words of the sequence (Lee,

1989). Then, the different models are trained by accumulating the partial results from each training sentence. Other training methods, such as the *segmental k-means* (Juang *et al*., 1985), introduce a VA segmentation to provide more accurate initial models to the Baum–Welch reestimation.

Finally, as previously mentioned, the statistical approach to CSR requires the application of a *grammar*. The obvious way to proceed is to build a finite state language, as we did in the example of Figure 2.14, for a given task. However, sometimes the task is not fully determined, but we only have a set of training sentences. Then, it is possible to have testing sentences that did not appear in training. This fact must be taken into account by using a flexible grammar. One possibility is that in which the probability of a word depends only on the $(n-1)$ previous words. They are known as *N-grams* (Huang *et al*., 2001). For example, when a bigram grammar is applied, the probability of a sequence of words (sentence) $W = (w_1, w_2, \ldots, w_M)$ is given by

$$P(W) = P(w_1) \prod_{m=2}^{m=M} P(w_m | w_{m-1}) \tag{2.78}$$

A special case of bigram is that in which all word pairs are equiprobable. In this case (known as *word pair grammar*), $P(W)$ is a constant and can be obviated (Lee, 1989). More information about language modeling can be found in Rosenfeld (2000).

2.8 Model Adaptation

Acoustic models are trained to accommodate a wide range of acoustic variability. Nevertheless, there always exists a mismatch between the acoustic conditions in which acoustic models are trained and those in which they are used. An obvious solution to this problem is to retrain the system in every new condition it is going to be used. Unfortunately, it is almost always impossible to collect sufficient training data for a complete retraining of the system.

One important factor in the usability of a speech recognition system is the possibility of adapting it to new operation conditions using a small amount of data. Adaptive techniques can be used to dynamically reduce the mismatch introduced by different microphones, transmission channels, background noises or speaker characteristics. The model adaptation goal is to modify the model parameters to reduce the mismatch of a new operation environment using only a small amount of adaptation data. One typical situation is the case of speaker variability. A speaker-dependent system can offer significant WER reductions in comparison to a speaker-independent system when sufficient data is available to retrain the system (Huang and Lee, 1993), but speaker adaptation can give almost the same results with a reduced amount of speaker- specific data.

Adaptation can be performed in several ways. One can perform continuous adaptation of the system during normal operation. That is, every new sentence decoded by the system is used to update the model parameters. This is an unsupervised adaptation scheme which uses the output of the recognizer to guide the adaptation process. The main advantage of this approach is that it can track nonstationary mismatches. However, the recognition results may be imperfect yielding incorrect transcriptions. When the ER is high, the adaptation process can diverge because incorrect transcriptions are used to guide it. One

alternative is to use a supervised scheme, in which the correct transcription of the adaptation speech is known in advance. The adaptation is usually performed by asking the user to pronounce a given set of sentences, which are used in turn to adapt the acoustic models.

2.8.1 Maximum Likelihood Linear Regression (MLLR)

In a CHMM-based speech recognition system, the output probability of each state is modeled as a Gaussian mixture. The parameters of these output densities (i.e. the mean vector and covariance matrix of each Gaussian in the mixture) are the most important parameters to adapt. The Maximum likelihood linear regression (MLLR) (Leggetter and Woodland, 1995) approach computes a set of transformations that will reduce the mismatch between the initial models and the adaptation data.

The transformations are constrained to be linear with the general form

$$\tilde{\boldsymbol{\mu}}_m = \mathbf{A}_m \boldsymbol{\mu}_m + \mathbf{b}_m \tag{2.79}$$

and represent the new mean vector $\tilde{\boldsymbol{\mu}}_m$ of Gaussian m as a rotation and translation of the original n-component mean vector $\boldsymbol{\mu}_m$, with $\mathbf{A}_m$ the rotation matrix of dimension $n \times n$ and $\mathbf{b}_m$ the bias vector. It is usual to write the transformation in a more compact form as

$$\tilde{\xi}_m = \mathbf{W}_m \xi_m \tag{2.80}$$

where ξ is the extended mean vector

$$\xi_m = [1 \ \mu_{m1} \ \mu_{m2} \ \ldots \ \mu_{mn}]^T \tag{2.81}$$

and $\mathbf{W}_m$ is a $n \times (n+1)$ transformation matrix that can be decomposed as

$$\mathbf{W}_m = [\mathbf{b}_m \ \mathbf{A}_m] \tag{2.82}$$

The parameters of the transformation matrix $\mathbf{W}_m$ are estimated by maximizing the likelihood of the adaptation data.

2.8.1.1 Regression Class Trees

The number of adaptation transforms must be selected as a trade-off between the number of Gaussians and the amount of adaptation data. If sufficient adaptation data is available, a transformation can be specifically designed for each Gaussian in the model set. In practice, the number of Gaussians is usually too large (of the order of 20,000) and this will require a huge amount of adaptation data. Instead, Gaussians can be grouped in broad phone classes (i.e. vowels, silence, fricatives, etc.) and a common transformation can be trained for each class, obtaining a more robust estimation of the parameters. Rather than using a static prior set of classes, a more robust estimation can be obtained by the use of *regression class trees* (Leggetter and Woodland, 1995).

The regression class tree is constructed by clustering together Gaussians that are close in the acoustic space, in such a way that similar components are transformed in a similar way (i.e. using the same transformation). Figure 2.15 shows a typical regression tree with four leaves. A binary regression tree can be constructed using a top-down splitting algorithm

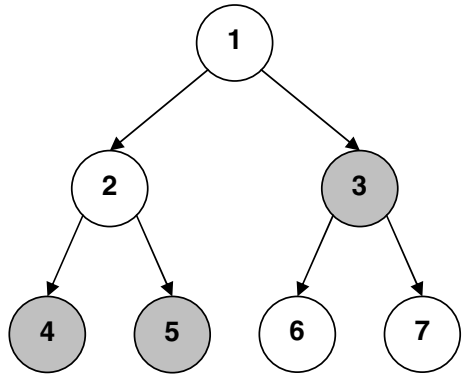


Figure 2.15 A binary regression tree

with a Euclidean distance. First, all Gaussians in the model set are assigned to the root node of the tree. At a given level of the tree, each node is split into two child nodes until a desired number of leaves is obtained (four in the example of Figure 2.15). The partition of each node is performed by distributing the Gaussians in the two child nodes in such a way that the sum of the Euclidean distances to the node centroid is minimized in each node. The terminal nodes (leaves) of the tree specify the base regression classes, and each Gaussian in the model set is assigned to one of these base classes.

During adaptation, the available data is aligned with the model set and occupancy counts (number of observation vectors aligned with a given Gaussian) are computed for each base regression class. If a base regression class has sufficient data, a transform matrix is then estimated. If there is no sufficient data in a given node, observations of child nodes are pooled in its parent node. This process is repeated until sufficient data is collected and then the transformation matrix is estimated. In the example of Figure 2.15, nodes 6 and 7 have insufficient data and therefore observations are pooled in parent node 3 and a common transformation matrix is estimated for this two nodes. Therefore, only three transformation matrices are estimated, one for node 4, one for node 5 and one common transformation for nodes 6 and 7. Finally, all Gaussians assigned to a base regression class are transformed using the same matrix. Details about regression class trees generation can be found in Gales (1996). Details about the estimation of the transformation matrices $\mathbf{W}$ and the adaptation of the covariance matrices can be found in Gales *et al*. (1996); Leggetter and Woodland (1995) and Woodland (1996).

2.8.1.2 Estimation Formulas for the Mean Transformation Matrix

As stated before, the transformation matrix is estimated by maximizing the likelihood of the adaptation data. Assuming that there are R Gaussians $m_1, m_2, \ldots, m_R$ sharing a common transformation matrix $\mathbf{W}_m$, this maximization leads to the following relation (Woodland, 1996):

$$\sum_{t=1}^{T}\sum_{r=1}^{R} L_{m_r}(t)\Sigma_{m_r}^{-1}o(t)\xi_{m_r}^{T} = \sum_{t=1}^{T}\sum_{r=1}^{R} L_{m_r}(t)\Sigma_{m_r}^{-1}\mathbf{W}_{m_r}\xi_{m_r}\xi_{m_r}^{T} \tag{2.83}$$

where L_{m_r} is the occupation likelihood defined as

$$L_{m_r}(t) = p(q_{m_r}(t)|\mathcal{M}, \mathbf{O}_T) \tag{2.84}$$

where $q_{m_r}(t)$, indicates the Gaussian component m_r at time t, $\mathcal{M}$ is the model set, and $\mathbf{O}_T = \{o(1), o(2), \ldots, o(T)\}$ the adaptation data. Defining the new variables $\mathbf{Z}$, $\mathbf{G}^{(i)}$ and $\mathbf{D}^{(r)}$

$$\mathbf{Z} = \sum_{t=1}^{T} \sum_{r=1}^{R} L_{m_r}(t) \Sigma_{m_r}^{-1} o(t) \xi_{m_r}^T \tag{2.85}$$

$$g_{jq}^{(i)} = \sum_{r=1}^{R} v_{ii}^{(r)} d_{jq}^{(r)} \tag{2.86}$$

$$\mathbf{V}^{(r)} = \sum_{t=1}^{T} L_{m_r}(t) \Sigma_{m_r}^{-1} \tag{2.87}$$

$$\mathbf{D}^{(r)} = \xi_{m_r} \xi_{m_r}^T \tag{2.88}$$

it can be shown that the solution for $\mathbf{W}_m$ is given by

$$\mathbf{w}_i^T = \mathbf{G}_i^{-1} \mathbf{z}_i^T \tag{2.89}$$

$\mathbf{w}_i$ being the i^{th} vector of $\mathbf{W}_m$ and $\mathbf{z}_i$ the i^{th} vector of $\mathbf{Z}$. An estimation formula has also been proposed for the covariance matrix transformation (see Woodland (1996) for details).

2.8.2 Maximum a Posteriori Linear Regression (MAPLR)

Maximum a posteriori (MAP) estimation of linear regression transformations (MAPLR) (Chesta *et al*., 1999) can also be used instead of ML. Application of MAP to this problem yields an estimation of the transformed means $\tilde{\boldsymbol{\mu}}_m$ as a linear interpolated value between the original means $\boldsymbol{\mu}_m$ and the means $\bar{\boldsymbol{\mu}}_m$ estimated using only the adaptation data

$$\tilde{\boldsymbol{\mu}}_m = \frac{N_m}{N_m + \tau} \bar{\boldsymbol{\mu}}_m + \frac{\tau}{N_m + \tau} \boldsymbol{\mu}_m \tag{2.90}$$

where τ is the weight of the a priori knowledge and N_m is the occupation likelihood of the adaptation data defined as

$$N_m = \sum_{t=1}^{T} L_m(t) \tag{2.91}$$

As a result, when the likelihood of occupation of a Gaussian component is small, the estimated mean is close to the model mean. A drawback of this approach is that more data is necessary for an effective adaptation. When a large amount of data is available, maximum a posteriori linear regression (MAPLR) begins to perform better than MLLR. In fact, the best results are obtained when both methods are combined by using MLLR-adapted means instead of model means in Equation (2.90) (Chesta *et al*., 1999).

2.9 Dealing with Uncertainty

It is commonly assumed that all the observations $\mathbf{x}_t$ of a given sequence are equally relevant for the decoding process, but there are situations in which we are not certain about the values of some of these observations. As an example, let us consider that the observations are transmitted over a noisy digital channel. In this situation some features can be affected by transmission errors, while others are received without errors. Another similar situation occurs when speech is contaminated by acoustic noise. Some features extracted from a noisy utterance (i.e. those corresponding to the low-energy parts of the utterance) are more affected than others (i.e. those extracted from the high-energy parts).

A common fact in these two examples is that there are situations in which we are not equally confident on the values of all features. If we can, in some way, quantify our level of confidence on each feature of each observation, it seems reasonable to modify the recognition procedures introduced in Section 2.6.3 in such a way that the most reliable observations have a higher weight than the least reliable ones. In particular, we will concentrate our attention on how the VA algorithm can be modified. We next develop two different approaches: exponential weighting and missing feature theory. We will see that both involve the modification of the observation probabilities during the VA decoding.

2.9.1 Exponential Weighting

A direct approach is to weight the observation probabilities of each feature vector reflecting our confidence on it like in Bernard and Alwan (2001). In this approach, Equation (2.57) is replaced by

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] [b_j(\mathbf{x}_t)]^{\gamma_t} \tag{2.92}$$

where γ_t is a measure of the confidence on the observation $\mathbf{x}_t$, and takes values in the interval [0, 1], with $\gamma_t = 0$ corresponding to full unreliable observations and the value $\gamma_t = 1$ to full reliable ones. It is clear that within this approach, fully unreliable observations do not contribute to the Viterbi step. The resulting VA algorithm is usually called the weighted Viterbi algorithm (WVA). This modified VA is obtained with a minimal modification of the normal VA (substituting $b_j(\mathbf{x}_t)$ by $[b_j(\mathbf{x}_t)]^{\gamma_t}$).

The WVA technique described by Equation (2.92) is independent of the specific form of the observation probabilities $b_i(\mathbf{x}_t)$, and is applicable to both discrete and continuous density HMMs. In the particular case of using the Gaussian mixture observation probabilities of Equation (2.71) with diagonal covariance matrices, it is also possible to apply a specific weight $\gamma_{j,t}$ for each feature $x_t(j)$ $(j = 1, \ldots, p)$ by modifying the observation probabilities as follows (Potamianos and Weerackody, 2001):

$$\hat{b}_i(\mathbf{x}_t) = \sum_{k=1}^{M} c_{ik} \prod_{j=1}^{p} \left[\mathcal{N}(x_t(j); \boldsymbol{\mu}_{i,k}(j), \Sigma_{i,k,j})\right]^{\gamma_{j,t}} \tag{2.93}$$

2.9.2 Bayesian Optimal Classification with Uncertain Data

If we do not consider uncertainty in the observations, the optimal Bayes decision about the class c^* to which an observation x belongs to is given by the MAP decision MAP rule

$$c^* = \underset{c}{\operatorname{argmax}}\, P(c|x) \tag{2.94}$$

In case of uncertain values we can consider that x is a random variable and, therefore, $P(c|x)$ is a random variable too. Then, the optimal Bayes decision rule is (Morris *et al.*, 1998)

$$c^* = \underset{c}{\operatorname{argmax}}\, E[P(c|x)|\mathcal{K}] \tag{2.95}$$

where $\mathcal{K}$ represents some knowledge we can apply to the expected value computation.

Let us see how we can apply the optimization of Equation (2.95) to ASR. As explained in Section 2.7, ASR is usually carried out by applying the VA to a single macromodel λ. The VA finds the state sequence $Q = (q_1, \ldots, q_T)$ that maximizes the probability $P(Q|X, \lambda)$ or, equivalently, $P(X, Q|\lambda)$ (Equation (2.53)). If the input data X is missing or uncertain, then X and $P(Q|X, \lambda)$ are random variables, and the Bayesian optimal classification rule becomes

$$Q^* = \underset{Q}{\operatorname{argmax}}\, E\left[P(Q|X, \lambda)|\mathcal{K}\right] \tag{2.96}$$

This optimization problem has been addressed from different points of view, yielding different solutions, which are generically known as *missing data* techniques. We develop two of these solutions in the following subsections.

For convenience, we will write $P(Q|X, \lambda)$ (implicitly defined in Equation (2.40)) as

$$P(Q|X, \lambda) = \frac{P(X, Q|\lambda)}{P(X|\lambda)} = C \prod_{t=1}^{T} a_{q_{t-1},q_t} P(\mathbf{x}_t|q_t) \tag{2.97}$$

where we have considered that q_0 is an initial null state (π_{q_1} is notated as a_{q_0,q_1}), that $P(X|\lambda) = 1/C$ is a constant (it does not depend on Q), and that $P(\mathbf{x}_t|q_t)$ is the observation probability of $\mathbf{x}_t$ in state q_t.

2.9.2.1 Viterbi Decoding with Missing Data

Let us consider that each observation $\mathbf{x}_t$ can be split into a *present* (or certain) subvector $\mathbf{x}_{pt}$ and a missing (or uncertain) subvector $\mathbf{x}_{mt}$, and let us note the sequences of present and missing subvectors as X_p and X_m respectively, so that $X = (X_p, X_m)$. Then, we can write that (Morris *et al.*, 1998)

$$P(Q|X, \lambda) = \frac{P(X|Q, \lambda)P(Q|\lambda)}{P(X|\lambda)} \tag{2.98}$$

$$= \frac{P(X_p|Q, \lambda)P(X_m|X_p, Q, \lambda)P(Q|\lambda)}{P(X_p|\lambda)P(X_m|X_p, \lambda)} \tag{2.99}$$

$$= \frac{P(X_p, Q|\lambda)}{P(X_p|\lambda)} \frac{P(X_m|X_p, Q, \lambda)}{P(X_m|X_p, \lambda)} \tag{2.100}$$

Using the common assumptions of statistical independence between feature vectors and Markovian dependence between states

$$P(Q|X,\lambda) = C\prod_{t=1}^{T} a_{q_{t-1},q_t} P(\mathbf{x}_{pt}|q_t)\frac{P(\mathbf{x}_{mt}|\mathbf{x}_{pt},q_t)}{P(\mathbf{x}_{mt}|\mathbf{x}_{pt})} \tag{2.101}$$

where $C = P(X_p|\lambda)$ is a normalization constant. Then, in order to obtain the optimal path, we have to maximize

$$E\left[P(Q|X,\lambda)|\mathcal{K}\right] = C\prod_{t=1}^{T} a_{q_{t-1},q_t} P(\mathbf{x}_{pt}|q_t) E\left[\frac{P(\mathbf{x}_{mt}|\mathbf{x}_{pt},q_t)}{P(\mathbf{x}_{mt}|\mathbf{x}_{pt})}|\mathcal{K}\right] \tag{2.102}$$

Therefore, the optimization can be carried out using the same VA algorithm but substituting the observation probabilities $P(\mathbf{x}|s_i)$ by

$$P(\mathbf{x}_p|s_i)E\left[\frac{P(\mathbf{x}_m|\mathbf{x}_p,s_i)}{P(\mathbf{x}_m|\mathbf{x}_p)}|\mathcal{K}\right] = P(\mathbf{x}_p|s_i)\int_{\mathbf{x}_m}\frac{P(\mathbf{x}_m|\mathbf{x}_p,s_i)}{P(\mathbf{x}_m|\mathbf{x}_p)}P(\mathbf{x}_m|\mathcal{K})d\mathbf{x}_m \tag{2.103}$$

For example, in the particular case that our only knowledge is $\mathcal{K} = \mathbf{x}_p$, the expected value in the above equation is 1 and the expression becomes $P(\mathbf{x}_p|s_i)$. That is, the missing features are simply ignored. This technique is commonly referred to as *marginalization*.

2.9.2.2 Viterbi Decoding with Soft Missing Data

An alternative approach consists of considering that the clean data X is (in general) uncertain and has a pdf $s(X)$. This means that we are using a probabilistic or "soft" data instead of the common deterministic data. By using a pdf instead of a single value to describe the observations, we can quantify the confidence or reliability of each observation (Morris *et al.*, 2001). In Figure 2.16 several possible alternatives are shown. The pdf associated with each observation models the uncertainty about its true value. For example, a Dirac delta distribution represents a full reliable observation, while other distributions represent observations with some level of uncertainty.

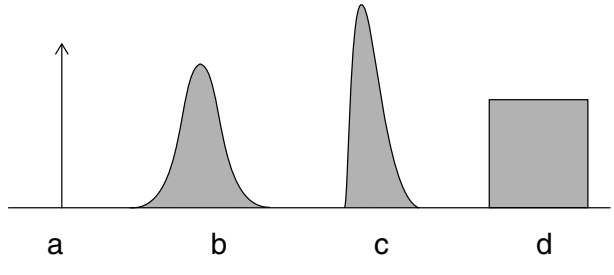


Figure 2.16 Probabilistic description of observations. (a) Delta, (b) Gaussian, (c) Beta, (d) Uniform

If we consider such soft data, the expected value to be maximized in Equation (2.96) is (Morris *et al.*, 2001)

$$E[P(Q|X,\lambda)|X \sim s(X)] = P(Q|\lambda)\int_X \frac{P(X|Q,\lambda)}{P(X|\lambda)} s(X)\,dX \tag{2.104}$$

We can consider that $s(X)$ is influenced by three types of knowledge: the clean training data X^{tr} (modeled by $P(X|X^{tr}) = P(X|\lambda)$), the observed uncertain data X^{obs} and any other knowledge $\mathcal{K}$. Then, it can be derived (assuming independence between X^{tr} and X^{obs}) that

$$s(X) = P(X|X^{tr}, X^{obs}, \mathcal{K}) = P(X|\lambda)P(X|X^{obs}, \mathcal{K})/P(X) \tag{2.105}$$

Assuming that the prior $P(X)$ is almost constant (equal to $1/C$), we can rewrite the above equation as

$$s(X) = CP(X|\lambda)s'(X) \tag{2.106}$$

where $s'(X) = P(X|X^{obs}, \mathcal{K})$ will be referred to as "evidence" pdf. Provided that C and $P(X|\lambda)$ are both nonzero, we can write

$$E[P(Q|X,\lambda)] = CP(Q|\lambda)\int_X P(X|Q,\lambda)s'(X)\,dX \tag{2.107}$$

Under the assumption

$$s'(X) = \prod_{t=1}^{T} s'(\mathbf{x}_t) \tag{2.108}$$

we finally obtain that the expected value to be maximized is,

$$E[P(X|Q,\lambda)] = C\prod_{t=1}^{T} a_{q_{t-1},q_t}\int_{\mathbf{x}_t} p(\mathbf{x}_t|q_t)s'(\mathbf{x}_t)d\mathbf{x}_t \tag{2.109}$$

where $s'(\mathbf{x}_t)$ is the evidence pdf of $\mathbf{x}_t$. This expression is similar to (2.97) and the decoding process the same, with the only difference being that the observation probability $b_{q_t}(\mathbf{x}_t) = p(\mathbf{x}_t|q_t)$ is replaced by

$$\int_{\mathbf{x}_t} p(\mathbf{x}_t|q_t)s'(\mathbf{x}_t)d\mathbf{x}_t \tag{2.110}$$

to take into account the evidence of each observation. Various evidence pdfs, like the ones shown in Figure 2.16, can be used depending on the observation process. For example, using a Dirac's delta, which represents certain data, leads us to the traditional Viterbi decoding with the observed values

$$s'(\mathbf{x}_t) = \delta(\mathbf{x}_t - \mathbf{x}_t^{obs}) \tag{2.111}$$

$$E[P(X|Q,\lambda)] = C\prod_{t=1}^{T} a_{q_{t-1},q_t}\int_{\mathbf{x}_t} p(\mathbf{x}_t|q_t)\delta(x_t - x_t^{obs})d\mathbf{x}_t$$
$$= C\prod_{t=1}^{T} a_{q_{t-1},q_t}\, p(\mathbf{x}_t^{obs}|q_t) \tag{2.112}$$

while other evidence pdfs provide averaged values.

Another interesting case is when the evidence pdf is a Gaussian:

$$s'(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{x}_t^{obs}, \Gamma_t) \tag{2.113}$$

and the observation probability of each HMM state is modeled as a Gaussian mixture as in (Equation (2.71)). In this case, for any state s_i

$$\int_{\mathbf{x}_t} p(\mathbf{x}_t|s_i)s'(\mathbf{x}_t)d\mathbf{x}_t = \sum_{k=1}^{M} c_{ik}\int_{\mathbf{x}_t}\mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{ik}, \Sigma_{ik})\mathcal{N}(\mathbf{x}_t; \mathbf{x}_t^{obs}, \Gamma_t)d\mathbf{x}_t$$
$$= \sum_{k=1}^{M} c_{ik}\mathcal{N}(\mathbf{x}_t^{obs}; \boldsymbol{\mu}_{ik}, \Sigma_{ik} + \Gamma_t) \tag{2.114}$$

The uncertainty of the observation (measured by the covariance matrix Γ_t) increases the covariance of the mixture components, deweighting the contribution of the observation to the final likelihood evaluation. Similar approaches can be found in Droppo *et al.* (2002) and Arrowood and Clements (2002), where Gaussian pdfs are used to model the observation uncertainty.

3

Networks and Degradation

3.1 Introduction

The RSR systems we are dealing with require a digital network for their deployment. Typically, this will be either a mobile telephony network or an IP-based network. In this chapter, we introduce the basic features of these networks that are relevant for RSR development. Special attention is given to the transmission channels used by these networks and the degradation that they usually introduce. In particular, a useful topic in RSR is how to establish a channel model suitable for the specific network in which we plan to implement the RSR system. This will allow the simulation and testing of the conditions under which this system has to work. Thus, some of the more utilized channels models are introduced in the sections devoted to their corresponding networks.

However, transmission channel distortion is not the only important degradation that can affect an RSR system. We must also consider the degradation introduced by the speech coding process and the acoustic environment, that is, the acoustic channel distortion. The acoustic environment is a crucial source of degradation, specially for systems deployed over mobile networks, in which the users can access the system in very aggressive acoustic environments (airports, train and bus stations, etc.). The last part of the chapter is devoted to the introduction and modeling of this degradation. Acoustic channel models will be helpful tools for system simulation and testing. Besides, an analysis of how this acoustic distortion affects the recognition features is provided. This will also be useful in Chapter 6, devoted to robust recognition FEs. The degradation introduced by speech codecs is analyzed in the next chapter.

3.2 Mobile and Wireless Networks

The digital mobile telephony was mainly developed and deployed during the eighties and nineties, impelled by the strong demand for such technology and the need for unified systems for it in order to lower costs and offer a minimum quality of service (QoS). In Europe, the group special mobile (GSM) was created in 1982 by the CEPT (*conference europeen des postes et telecommunications*) for the standardization of a unified radiotelecommunication system of second generation (2G) in the band of 900 MHz. Versions of GSM were

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

also created in the bands of 1800 and 1900 MHz (DCS1800 and DCS1900 systems). The work of the GSM group took about 10 years, during which period the acronym GSM acronym acquired the meaning *global system for mobile*, and its results were published as a set of ETSI standards. Currently, this standard is implemented in Europe and 120 countries of other continents. Other 2G systems are the Digital Advanced Mobile Phone System(DAMPS, also International Standard IS-136, formerly IS-54) deployed in Unites States of America, the Interim Standard 95 (IS-95) deployed in United States of America, South Korea and South America, and the Personal Digital Cellular (PDC) in Japan.

The main features of some of these 2G mobile networks are summarized in the following:

- GSM: Frequency bands 900, 1800 and 1900 MHz. Radio resource sharing method: TDMA (time division multiple access). Services: digital telephony (speech codecs at 13, 12.2, 12.2–4.75 or 6.5 kbps), data services up to 9.6 kbps, short message service (SMS), facsimile (group 3), and so on. International roaming (change of company providing service).
- DAMPS: Frequency bands 800 and 1900 MHz. TDMA access. High degree of compatibility with the previous system advanced mobile phone system (AMPS). Services: digital telephony (speech codec at 8 kbps), data services (up to 9.6 kbps, extended to 384 with IS-136HS), SMS.
- IS-95: Frequency bands 800 and 1900 MHz. CDMA (code division multiple access). Services: digital telephony (speech codec at 8 and 13 kbps), data services up to 9.6 kbps. International roaming.

The third generation (3G) of mobile communications is a concept identified with IMT-2000 (International Mobile Communications 2000). This is a set of recommendations that, in fact, has yielded several 3G systems such as CDMA-2000 and the *universal mobile telephone system* (UMTS). UMTS was started in 1998 by a consortium of several organizations called 3GPP. This new system has been developed from a GSM core (compatible with it) as an answer to the need for a universal coverage and to introduce new multimedia services. The data rates ranges from 144 kbps to 2 Mbps. The radio interface (UTRA (UMTS Terrestrial Radio Access)) uses two different modes for the radio link: TDD (time division duplex) for the bands 1900–1920 and 2010–2025 MHz, and FDD (frequency division duplex) for the bands 1920–1980 and 2110–2170 MHz. The access scheme is CDMA with two variants: W-CDMA (wideband CDMA) for FDD, and a combination TDMA/CDMA for TDD.

In 2004, 3GPP approved the ETSI Aurora DSR standard extended advanced front end (XAFE) (ES 202 212) as the recommended codec for speech-enabled services for UMTS release 6. After this, 3GPP have published a new standard (TS 26.243) (3GPP, 2004b) that specifies the fixed-point software implementation of the XAFE.

Finally, we also mention here two standards for wireless LAN (local area network), IEEE 802.11 (also wireless Ethernet or WiFi, from Wireless Fidelity) and Bluetooth. IEEE 802.11 comprises three standards: 802.11a (54 Mbps), 802.11b (11 Mbps) and 802.11g (more than 20 Mbps). For example, IEEE 802.11b is a high-power access technology that uses *direct sequence spread spectrum* (DSSS, similar to CDMA), at a frequency of 2.4 GHz (without license), and provides up to 11 Mbps. Bluetooth is essentially a

low-power cable replacing technology that uses TDD-TDMA and works in the band of 2.45 GHz (without license) and provides a data channel (up to 721 kbps) and three speech channels (64 kbps). IEEE 802.11 and Bluetooth networks can be integrated in IP networks, which are reviewed later in this chapter.

3.2.1 Cellular Structure of a Mobile Network

One of the most serious obstacles for the deployment of the mobile networks and telephony was the saturation of the radioelectric spectrum. The concept of cellular system meant a great advance for the resolution of this problem. The underlying idea is to divide the total coverage area into smaller areas called cells. Each cell is covered by a base station (BS), which has a certain number of frequency channels assigned to it. This structure allows reuse of frequencies (cochannels) in different cells and distribute the resources depending on the traffic. In order to reduce interferences, different groups of channels are assigned to close BSs. Also, the number of BSs can be increased when the service demand grows, thus avoiding the need for more radio channels. This results in a nonuniform cell size, which is larger in remote and sparsely populated areas than in urban areas. Thus, it is possible to distinguish macrocells (radius from several hundred meters to 10 kilometers), microcells (crowed and small areas like shopping centers, airports, etc.) or picocells (industrial and office buildings). The lower the emitted power from the BS antenna, the smaller the cell size. Figure 3.1 shows a diagram of an ideal uniform cell distribution and an example of channel reutilization (neighboring cells use different groups of channels, numbered from 1 to 7). The hexagonal shape of the cells is merely conceptual, and allows an easy analysis of the cellular system. The real coverage area of a cell does not have a hexagonal shape and is known as *footprint*.

3.2.2 Example of a Mobile Network Architecture: GSM/GPRS

The GSM network has the hierarchical structure shown in Figure 3.2. It consists of the following elements (Garg and Wilkes, 1999):

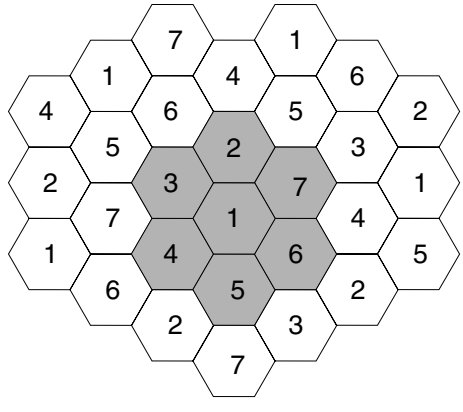


Figure 3.1 Scheme of an ideal uniform cell distribution (clusters of seven cells)

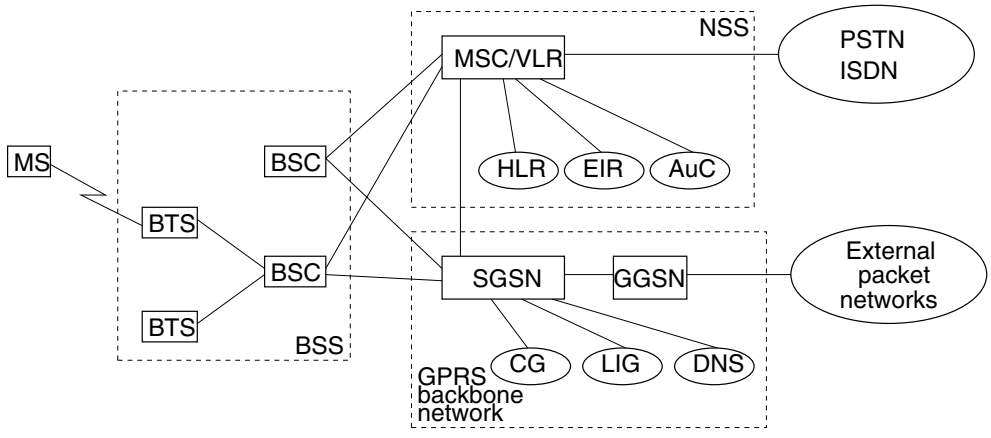


Figure 3.2 Architecture of the GSM/GPRS network

- MS (Mobile Station): It is the terminal from which the user accesses the network. There are several types of MS devices such as handset cellular phones or hands-free cellular phones for use in cars. The MS includes the speech codec and the functions necessary to access the network.
- BSS (Base Station System): It includes the following:
  - BTS (Base Transceiver Station): This is the GSM name for the BS. It transfers signals between the MS and the network by providing radio channels in its cell. It performs the necessary signal processing for the radio interface (known as *Um*).
  - BSC (Base Station Controller): It manages the radio interface. It controls the emission power of the MS and handles the channel assignment, frequency hopping (frequency and time slots changes to combat and randomize the multipath effect and the cochannel interference) and handovers (change of the cell assigned to the user).
- NSS (Network SubSystem): It includes the main switching functions and necessary databases. The main element of the NSS is the *Mobile services Switching Center* (MSC), which contains the switching functions for voice and data exchange within the network and with other external networks (PSTN, ISDN, etc.). The MSC also handles other functions such as registration, authentication, location updating, (interBSC) handovers and call routing for roaming. The NSS includes several databases:
  - HLR (Home Location Register): It contains the administrative information and current location of the subscribers.
  - VLR (Visitor Location Register): It temporarily contains the information of visitor subscribers.
  - AuC (Authentication Center): It stores the information necessary for user authentication and encryption.
  - EIR (Equipment Identity Register): It contains information on the identity of mobile equipment to prevent calls from stolen, unauthorized or defective MSs.

In addition to Um radio interface, it is also common to consider the interfaces *Abis* and *A*, which establish the functional borders (BTS/BSC and BSS/NSS) depicted in Figure 3.2.

Another important element is the TRAU (Transcoding Rate and Adaptation Unit). It compresses the speech from 64 to 16 kbps (or vice versa). Its function is commonly assigned to the BTS, although, in practice, it is usually located at the MSC site in order to use a 16-kbps channel between the BSC and the MSC, instead of one of 64 kbps.

Data and voice use *traffic channels* (TCH) at full rate (TCH/F, 22.8 kbps) or half rate (TCH/H, 11.4 kbps). For example, the TCH/F channel allows bitrates of 2.4, 4.8 or 9.6 kbps for data, and from 4.75 to 13 kbps for voice (depending on the speech codec applied). The remaining bits up to the total bitrate are used for channel error protection. There are also a set of control channels for frequency/time synchronization, broadcast system information, access control, establishment of calls, measures of physical channel quality, power control, and so on. Physically, the TDMA access scheme provides 124 frequency channels of 200 kHz (in the 900-MHz band) divided into eight logic channels (each occupies a different time slot with duration 0.577 ms), which form a *TDMA frame* (duration 4.615 ms).

Although GSM is basically a circuit-switching service, it also offers a packet-switching service called general packet radio service (GPRS). Although packet-switched networks are studied in Section 3.3, let us now see some specific features of GPRS as a wireless/mobile network. GPRS is often referred to as *2.5G* since it can be considered to be an evolution of GSM toward 3G (IS-136 also supports GPRS). Its architecture is also shown in Figure 3.2. It shares the BSS subsystem with the circuit-switched GSM network. Then, the BSS is connected to a GPRS backbone network, which contains several subsystems. The role of the MSC is provided by the serving GPRS support node (SGSN). It provides mobility management, packet routing and transfer, and so on. The gateway GPRS support node (GGSN) is the interface with other external packet-switched networks. Other subsystems are the CG (charge gateway, to simplify billing), the LIG (lawful interception, to monitor traffic), the DNS (domain name system, to resolve the access point name required by the user), and the BG (border gateway, to interconnect different GPRS operators).

The GPRS data traffic is delivered over the packet data traffic channel (PDTCH), which can operate at different bitrates, into data units called *radio blocks*. A radio or RLC (radio link control) block consists of the four data bursts that correspond to the same time slot of four consecutive frames, transports 456 (114×4) bits and have an assigned duration of 20 ms. These blocks can be used by different users, so that we can consider this sharing as a type of TDMA within TDMA. The bitrates provided by GPRS depend on the channel coding scheme (CS) applied. There are four CSs that provide four different levels of protection against channel errors. These are CS-1, CS-2, CS-3 and CS-4 (no error correction), which may transport 20, 30, 36 and 50 bytes, respectively. Considering the 20-ms duration of an RLC block, the corresponding bitrates are 8, 12, 14.4 and 20 kbps, respectively. The CS may be dynamically assigned by the network, depending on the channel conditions. It must be taken into account that the payload of an RLC block may include any higher-level protocol headers. If more than one time slot is available, the bitrate is increased proportionally. Therefore, bitrates of 64–160 kbps are theoretically possible using eight time slots, although a maximum of 115 kbps is more realistic. However, the real bitrate depends on several factors such as the number of GSM users (sharing the same air interface), the number of GPRS users and the number of time slots the MS can manage. Enhanced GPRS (EGPRS) will allow higher

bitrates (theoretically, up to 473.6 kbps using eight time slots, and, realistically, up to 384 kbps) by applying advanced modulation techniques. In particular, EGPRS introduces nine modulation and coding schemes, MCS-1 to MCS-9, providing bitrates from 8.8 to 59.2 kbps (one time slot). MCS-1 to MCS-4 are quite similar to CS-1 to CS-4 and use the same modulation (Gaussian-filtered minimum shift keying (GMSK)) as the former ones, while MCS-5 to MCS-9 use advanced modulation (8-phase shift keying (8PSK)) (see Appendix B for details about modulation).

The GPRS/GSM MS terminals are of three types: class A (support simultaneous circuit- and packet-switched traffic), class B (support both types of traffic, but not simultaneously) and class C (connected to either GSM or GPRS; service selection must be carried out manually).

3.2.3 Degradation in Wireless Networks

When speech information is transmitted over a wireless network there are two potential sources of degradation that can affect an RSR system (Garg and Wilkes, 1999), speech coding and speech transmission. We will analyze the effect of speech coding in the following chapter and will concentrate now on the second type of distortion, introduced by the wireless transmission channel.

In first place, the transmitted signal is commonly degraded by an additive noise inherent in this transmission medium. The noise can be quite destructive when the receiver moves away from the transmitter, since the signal strength diminishes. This is known as *path loss*. However, the most destructive degradation of the radio channel is due to the *multipath* phenomenon. Figure 3.3 illustrates this aspect. The radio waves reach the receiver antenna by different paths, which are combined there. These different signals have different amplitudes and phases. The result is that the received signal has a *fading* appearance, since it can vary its strength and phase from one location to another that is placed quite close.

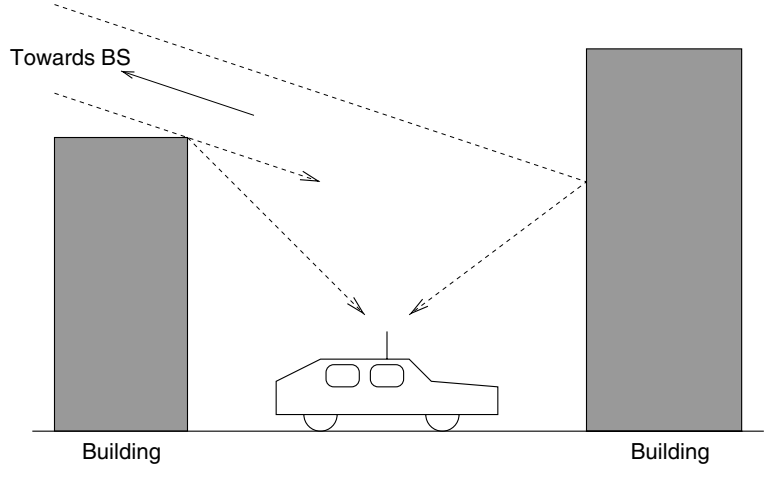


Figure 3.3 Illustration of the multipath phenomenon

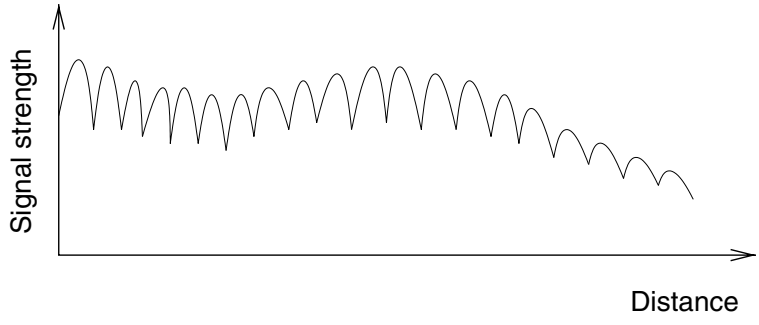


Figure 3.4 An example of signal fading versus distance

Fading is basically a spatial phenomenon, although it manifests in the time domain when the MS is moving. It must be taken into account that when the MS or the scatterers of the radio waves are moving (vegetation, trucks), the received signal also suffers from a *Doppler* shift in frequency.

Owing to multipath, the different paths can contribute constructively or destructively to the received signal. These rapid fluctuations (several fades per second) in amplitude and phase are known as *fast fading* or *Rayleigh fading*. Also, it can appear as *slow fading* owing to the partial path loss produced by large obstacles. Thus, the received signal envelope can be considered to be the product of a fast-fading component and a slow-fading component as shown in Figure 3.4. Fading can also be classified as *flat*, when it equally affects all frequency components, and as *selective* if it unequally affects these components. Deep fades can produce bursts of errors, which could be quite damaging for an application such as RSR.

Most of the work carried out on the effect of degraded wireless channels over RSR systems has dealt with the above mentioned types of degradation. However, with the emergence of new wireless networks using CDMA as access scheme, the degradation due to multiple access interference (MAI) must also be considered. MAI appears when the number of different users sharing the same transmission medium in CDMA grows. For a given user, the signals from other users appear as interferences that deteriorate the transmission. The effect of MAI on RSR has been studied in Han *et al*. (2004). RSR systems working over WLAN networks are also subject to interferences. For example, WiFi and Bluetooth networks share the same 2.4-GHz ISM band and thus interfere each other (Nour-Eldin *et al*., 2004).

Channel errors can be prevented by different techniques such us diversity, adaptive equalization, channel coding or interleaving, while the resilient errors can be treated at the receiver by applying concealment techniques. Some of these topics will be treated in Chapter 5 and Appendix C.

3.2.3.1 Characterization of Additive Noise and Fading

For an efficient transmission in a wireless network, the information is transmitted using digital modulation. The modulation process involves the change of some parameter of

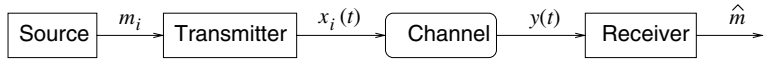


Figure 3.5 General diagram of wireless transmission

a carrier wave, thus obtaining a set of signals suitable for wireless transmission. The basic concepts regarding modulation and demodulation useful for the implementation and evaluation of RSR systems are given in Appendix B.

Let us now assume that we want to transmit symbols (one every T seconds) from an alphabet $\{m_i; i = 1, \ldots, M\}$. The binary case (0, 1) is a particular case with $M = 2$. A signal $x_i(t)$ of duration T, suitable for transmission, is assigned to each symbol m_i. Figure 3.5 shows a transmission process. After crossing the transmission channel, we do not receive the original signal $x_i(t)$, but a distorted version $y(t)$ is received, so that the received symbol is $\hat{m}$, which could be different from the transmitted one m_i if the channel is distorted enough. A channel model frequently used in the development and testing of communication systems is the additive white Gaussian noise (AWGN) channel, which modifies the transmitted signal as

$$y(t) = x_i(t) + n(t) \tag{3.1}$$

where $n(t)$ is a white Gaussian-distributed noise of zero mean and variance $\sigma_n^2 = N_0/2$. This noise is quite common in different communications systems and widely used for system analysis owing to its mathematical tractability.

Fading can be introduced by applying a random signal envelope a and a random phase θ to the transmitted signal (Haykin, 2000):

$$\tilde{y}(t) = ae^{-j\theta}\tilde{x}_i(t) + \tilde{n}(t) \tag{3.2}$$

where the tilde indicates the use of a complex notation ($f(t) = Re[\tilde{f}(t)]$). A background AWGN noise has also been considered by introducing $\tilde{n}(t)$. When there is no dominant received component, the envelope is Rayleigh-distributed:

$$p(a) = \frac{a}{\sigma^2}\exp\left(\frac{-a^2}{2\sigma^2}\right) \tag{3.3}$$

where $2\sigma^2 = E[a^2]$ is the mean power of the fading, and the phase has a uniform distribution. In this case, we have the *Rayleigh fading* channel, which has been shown to be a realistic one for the treatment of fading channels. When there is a dominant component (line-of-sight (LOS)), the envelope follows a Ricean distribution, and we obtain the *Ricean fading* channel.

Although we have just characterized noise and fading by employing time signals, it is more useful for their understanding and simulation to consider the modulation and demodulation processes through a different representation domain known as *signal space*, where we use vectors instead of time signals (Arthurs and Dym, 1962). This representation is also introduced in Appendix B.

3.2.3.2 Channel Condition Evaluation

A general measure of the channel condition is the signal-to-noise ratio (SNR), which is defined as the ratio of the average signal power to the average noise power. In the case of degradation caused by interferences from adjacent channels or cochannels, the carrier-to-interference (C/I) ratio is used instead.

An interesting measure of performance of a digital communication system is the average probability p_e of symbol error. Assuming an ML decoder, it is easy to prove, for the AWGN channel, that

$$p_e = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{3.4}$$

for binary phase shift keying (BPSK) modulation and

$$p_e = 1 - \left[1 - \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)\right]^2 \approx erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{3.5}$$

for quadriphase shift keying (QPSK) modulation. E_b is the energy per transmitted bit (according to the notation introduced in Appendix B, $E_b = E$ for BPSK, and $E_b = E/2$ for QPSK, since each symbol has two bits). Expression (3.4) is also a good approximation to the probability of error for the GMSK modulation used in GSM (Haykin, 2000).

The ratio E_b/N_0 is closely related to the channel SNR, which can be computed as

$$SNR = \frac{E_b}{N_0}\frac{R_s}{B} \tag{3.6}$$

where R_s is the bitrate and B is the signal bandwidth. Another figure of merit is the bit error rate (BER), which, in general, differs from p_e since each symbol can contain several bits. Thus, BER and p_e coincide in the BPSK case, while for QPSK modulation it can be obtained that

$$BER = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{3.7}$$

We see that QPSK transmits twice as much information as BPSK for the same BER and, therefore, for the same E_b/N_0 ratio. Thus, in this sense, QPSK has a better performance than BPSK.

In the case of a Rayleigh channel and BPSK, it can be shown that if the fading is slow enough so that the phase can be exactly determined (coherent reception), the probability of error is

$$p_e = \frac{1}{2}\left(1 - \sqrt{\frac{\gamma_0}{1+\gamma_0}}\right) \tag{3.8}$$

where $\gamma_0 = E[a^2]E_b/N_0$. It is also possible to define an *instantaneous bit error probability* for a specific received bit. In the case of BPSK over a fading channel, it can be shown

that the probability of error $p_e(y)$ of a received signal y (see signal space representation in Appendix B) can be computed as (Hagenauer, 1980)

$$p_e(y) = \frac{1}{1 + \exp|L_c y|} \qquad \text{with} \qquad L_c = 4a\frac{E_b}{N_0} \tag{3.9}$$

for a fading factor a assumed as known at the receiver ($a = 1$ corresponds to the AWGN channel). $|L_c y|$ can be considered as a reliability measure of the received bit.

3.2.4 Wireless Channel Models for RSR

In order to characterize the performance of RSR systems working over degraded wireless channels, the most direct way is to use bit error masks. An error mask $\mathbf{e}$ is a stream obtained for an specific transmission system in a specific channel condition by transmitting a known bitstream $\mathbf{x}$ and comparing it with the received bitstream $\mathbf{y}$ by applying the XOR operation $\mathbf{e} = \mathbf{x} \oplus \mathbf{y}$. The bitstream length must be large enough in order to be statistically representative. Once the error mask is available, it is possible to obtain the channel output bitstream corresponding to an arbitrary transmitted bitstream $\mathbf{x}$, as $\mathbf{y} = \mathbf{x} \oplus \mathbf{e}$. In Pearce (2000), the ETSI FE standard for DSR was tested using channel error masks obtained for the terrestrial trunked radio (TETRA) and GSM systems, conveniently adapted to the FE bitrate (4.8 kbps). For example, the GSM EP error patterns (ETSI, 2000c) EP1, EP2 and EP3, represent a GSM TCH channel (assuming ideal frequency hopping and flat fading) at 10, 7 and 4 dB of C/I, respectively. Each pattern yields several masks depending on the mode (speech or data) owing to the different error protection level of each mode. Table 3.1 shows the WAcc results obtained with the ETSI DSR FE standard (data mode) and a NSR system using the GSM-enhanced full rate (EFR) codec (speech mode) when the transmission is simulated by means of the EP patterns. These results are obtained with the Aurora-2 recognition task using only its clean sentences for testing. EFR offers worse results in clean conditions (due to speech compression) and in degraded conditions (due to the smaller level of error protection of the speech mode).

However, for the development of error protection and mitigation techniques, it is convenient to use analytical channel models. They are also a useful simulation tool that allows testing of the robustness of RSR system under a specific degradation. There are two types of analytical channel models (Bai and Atiquzzaman, 2003). First, we find the *physical-layer–oriented* models, which work directly over either transmitted signals or signal vectors. Secondly, we must consider that the applications using wireless transmission usually transfer the data in blocks or packets. This yields a second type of models known as *higher-layer–oriented* models.

Table 3.1 Word accuracies obtained with DSR/FE and NSR/EFR under the GSM EP channel conditions

RSR system	Channel condition			
	Clean	EP1	EP2	EP3
DSR	99.04	99.04	98.95	93.41
EFR	98.70	98.44	96.91	84.48

3.2.4.1 Physical-layer–oriented Channel Models

The simple and smart AWGN channel described in the previous section can be applied to study the robustness of RSR systems against channel errors. For its simulation, it is convenient to use the signal space representation introduced in Appendix B. In particular, Equation (B.2) offers a direct and simple way to simulate it. For example, for a BPSK modulation with coherent reception, the AWGN channel of Equation (3.1) can be simulated (using the signal space representation of Appendix B, (Equation (B.2)) as

$$y_k = x_k + n_k \tag{3.10}$$

where k is a time index indicating the time interval $[(k-1)T, kT]$ and x_k ($x_k \in [+\sqrt{E}, -\sqrt{E}]$), n_k and y_k are the transmitted bit, the noise (a discrete AWGN random process) and the received signal at that time interval.

The AWGN channel tends to introduce errors randomly distributed in the transmitted bitstream. However, in a mobile wireless environment, the main degradation of an RSR system is due to the error bursts caused by fading. The effects of error bursts and random errors are compared later in this chapter. As we pointed out earlier, fading is a spatial phenomenon, which manifests in time owing to the movement of the MS or the different signal scatterers. In order to simulate these time variations, we can consider that the envelope is a complex correlated Gaussian process $\alpha(t) = a(t)e^{j\theta(t)}$ (thus, $a(t)$ is Rayleigh-distributed for each time t) with the following autocorrelation function (Jakes, 1974):

$$R_\alpha(\tau) = E[\alpha(t)\alpha(t+\tau)] = \sigma^2 J_0(2\pi \frac{v}{\lambda}\tau) \tag{3.11}$$

where $2\sigma^2$ is the mean square value of the fading, $J_0(x)$ is the zeroth-order Bessel function, v is the MS speed and λ is the carrier wavelength. The level-crossing rate N_A (average number of times per second the signal envelope crosses an specified level $a = A$ in the positive direction) and the average fade duration τ_A (below level A) can be obtained as (Jakes, 1974)

$$N_A = \sqrt{2\pi}\frac{v}{\lambda}\rho e^{-\rho^2} \approx \sqrt{2\pi}\frac{v}{\lambda}\rho \tag{3.12}$$

$$\tau_A = \frac{e^{\rho^2} - 1}{\sqrt{2\pi}\frac{v}{\lambda}\rho} \approx \frac{\lambda}{v}\frac{\rho}{\sqrt{2\pi}} \tag{3.13}$$

where $\rho = A/(\sqrt{2}\sigma)$. For example, for a BPSK modulation with coherent reception, the Rayleigh fading channel of Equation (3.2) can be simulated (using the signal space representation of Appendix B) as

$$y_k = a_k x_k + n_k \tag{3.14}$$

where a_k is the fading factor at the time interval k (assumed as constant during that interval). The autocorrelation function (Equation (3.11)) must be computed for $\tau = kT$. An example of Rayleigh fading channel applied to wireless DSR can be found in Weerackody *et al.* (2002). Table 3.2 shows the WER results reported in this paper for an IWR task and for AWGN and Rayleigh channels. The system has a DSR architecture that works at

Table 3.2 WER results of a DSR/IWR system operating over AWGN and Rayleigh (for several MS speeds) channels (after Weerackody *et al*., 2002)

AWGN channel	
SNR (dB)	WER
Clean	7.1
4	7.4
3	7.6
2	10.3
1	49.6

Rayleigh channel			
SNR (dB)	10 km/h	50 km/h	100 km/h
15	7.4	7.1	7.3
10	10.2	7.7	7.3
7	17.3	10.6	8.7
5	28.3	21.0	16.7

9.6 kbps with an error protection scheme based on convolutional codes plus interleaving (developed in Section 5.2.2.1) and uses differential phase shift keying (DPSK) modulation. We can observe that the DSR system is more robust against random errors (AWGN channel) than against error bursts (Rayleigh channel). Also, low MS speeds are more damaging since they involve longer error bursts.

A simpler alternative to the Rayleigh channel for testing the effects of fading over speech recognition is the bursty channel model applied in Peinado *et al*. (2003). The error burst duration is fixed, which allows a specific test of the RSR system performance for that duration. It is a variation of the AWGN channel (Equations (B.2) or (3.10) for BPSK), for which the channel noise is obtained as a superposition of a background AWGN noise (variance $N_g/2$) plus a sequence of AWGN noise bursts (variance $N_b/2 \gg N_g/2$) of fixed duration d (in number of bits), with a separation given by a Poisson variable of mean T_b. The average variance of the channel noise is

$$\frac{N_0}{2} = \frac{N_g}{2} + \frac{N_b}{2}\frac{d}{T_b} \tag{3.15}$$

which allows to compute an average value for E_b/N_0.

The previous model reflects that the channel can be in two different states: a "good" state with a bit error probability and a "bad" state with a higher error probability. This consideration leads us directly to the well-known Gilbert–Elliot channel model (Elliot, 1963, 1965) depicted in Figure 3.6. This is a two-state (hidden) Markov model. State s_0 represents the good state, and s_1 is the bad state. Each state can emit at each time one of two possible symbols (0 and 1) that indicate a correct or an erroneous reception of the transmitted symbol, respectively. Considering a binary channel and following the notation introduced in Chapter 2 for HMMs, if p_{e0} and p_{e1} are the bit error probabilities in each state ($p_{e1} > p_{e0}$), the observation probabilities are

$$b_0(0) = 1 - p_{e0} \qquad b_0(1) = p_{e0}$$
$$b_1(0) = 1 - p_{e1} \qquad b_1(1) = p_{e1}$$

The model is completed with the transition probabilities

$$a_{00} = 1 - p \qquad a_{01} = p$$
$$a_{10} = q \qquad a_{11} = 1 - q$$

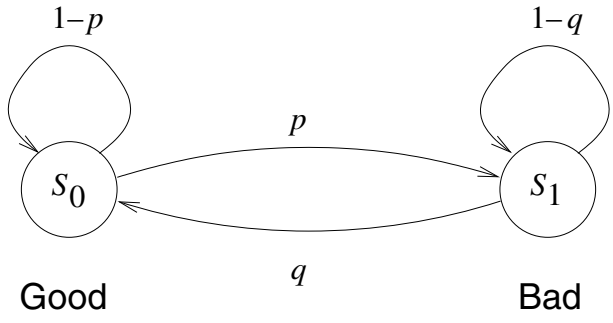


Figure 3.6 Gilbert–Elliot channel model

The *a priori* probability π_i and the mean duration $\bar{d}_i$ of each state can be obtained as

$$\pi_0 = \frac{q}{p+q} \qquad \bar{d}_0 = 1/p \tag{3.16}$$

$$\pi_1 = \frac{p}{p+q} \qquad \bar{d}_1 = 1/q \tag{3.17}$$

The average bit error probability is

$$p_e = \pi_0 p_{e0} + \pi_1 p_{e1} \tag{3.18}$$

The Gilbert–Elliot model is a special case of a more general model with N states, where each state represents a different range of channel SNR and is assigned a given value of bit error probability (Wang and Moayeri, 1995).

We must note that there is an important difference between the Gilbert–Elliot model and the earlier ones. In the case of the AWGN, Rayleigh or bursty channel models, we can carry out both hard decision and soft decision (see Appendix B). In the first case, the decision about the received symbols is carried out at the same time as the received signal is demodulated by applying, for example, ML decoding as shown in Appendix B. For soft decision, the decision about the received symbol is left to later stages. For example, for BPSK, the received bit may be simply obtained as the sign of y_k in Equations (3.10) or (3.14) (hard decision), although the soft value y_k may also be used by the subsequent stages of the receiver (soft decision). However, the Gilbert–Elliot model only informs about the correctness of the received symbols. Therefore, soft values are not produced at any stage, so it is not applicable in soft-decision receivers.

3.2.4.2 Higher-layer–oriented Channel Models

As mentioned earlier, in the case of higher-layer channel models, the transmission units are data blocks or packets instead of bits. The type and amount of data contained in such blocks depends on how those data are organized for an efficient transmission. Data organization in RSR is treated in the next section, devoted to IP networks, but it is also dealt with in Chapters 5 and 7 (for the Aurora standards).

We are now interested in assessing whether a whole data block is correct or erroneous. A first approach is to use error patterns as in the physical-layer models. For example, in 3GPP (2004a) the performance of the XAFE standard is tested under three channel conditions represented by three EGPRS radio block error patterns EG_EP1, EG_EP2 and EG_EP3, with block error rates (BLER) of 1, 3 and 10 %. However, channel models are again a useful tool for analysis and simulation. The Markov models studied above can also be used as higher-layer models. They provide a natural way of representing the bursty characteristic of fading channels. The details of these channel models will be introduced in the next section, since they are extensively used to model packet losses in IP networks.

3.2.5 Implementation of RSR Systems over Mobile Networks

The concept of RSR was developed in parallel with the deployment of circuit-switched mobile networks and employing an NSR architecture (Euler and Zinke, 1994), that is, utilizing the network codecs and speech channels. As shown in the next chapter, the performance of this architecture may be quite vulnerable to speech coding and channel degradations. DSR appeared as an alternative to palliate these degradations over circuit-switched channels. Thus, the ETSI DSR standards include a multiframe format (described in Chapter 7) suitable for the transmission of the speech features over circuit-switched data channels. However, with the evolution of mobile networks toward 3G, packet-switched mobile networks (such as GPRS/EGPRS) seem more suitable for the integration of RSR, and particularly DSR, with other types of data and for the development of multimodal interfaces under the same data connection (Pearce, 2004). The implementation of RSR over packet-switched networks is treated in Section 3.3.4.

3.3 IP Networks

During the last few years, we have witnessed the rapid extension of Internet and its related IP, which is also used in other networks. A multiplicity of services, including speech-related ones, have benefited from this rapid and enormous deployment. Regarding RSR, IP networks appear as a natural platform for its implementation, owing to their client/server architecture, which is identical to that of RSR.

The origin of Internet and IP is in the packet-switching network ARPANET, developed during the seventies. Packet switching arose as an alternative to traditional circuit switching and was considered a promising technique to allow resource sharing among computers. The transmission unit in a packet-switching network is a data block called *packet*. A message can be transmitted in a single packet or broken up into several packets, which are transmitted independently. Although packets can follow a preplanned route from their source (virtual circuit approach), we will only pay attention to the datagram approach, for which the different packets of a given message can follow different routes in the network up to their destination. In this case, each packet consists of a header (containing the destination address) and the data or *payload* to be transmitted. Unlike circuit switching, there is no a prefixed route between transmitter and receiver. Therefore, packets can be transmitted without waiting for a connection. On the other hand, it is not possible to ensure when a given packet is going to reach its destination (or if it is going to arrive). ARPANET utilized switchers that were connected to a minimum of two

computers. Thus, packets had alternative routes to reach the destination. Each switcher had a routing table, specifying which way a packet should follow, and stored in memory the incoming packets until they were retransmitted.

Following the success of ARPANET, the need arose for connecting different computer networks, for which it was necessary to develop protocols that hid the physical network and allowed compatibility. The connection is possible through a routing device that suits the information format besides containing the needed routing information for both networks. This problem is only apparently easy, since it turns more and more complicated as the number of interconnected networks grows. The protocol suite transmission control protocol (TCP)/IP solved the problem, so that the information exchange among networks was transparent and the user could see them as a single virtual network. TCP/IP uses the same idea of packet switching as ARPANET. Therefore, the routing devices need not store information about user states or information flows. This simplicity is the factor that allows IP networks to grow in a big scale. The specifications of TCP/IP have been developed by the Internet Engineering Task Force (IETF) (IETF, n.d.) and are known as RFC (request for comment).

3.3.1 The TCP/IP Protocol Suite

TCP/IP involves a whole family of protocols designed to perform different tasks and provide services. These protocols can be conceptually grouped into different levels or layers as normally done in computer networks (Kurose and Ross, 2003). TCP/IP involves a four-layer model. The layers are as follows(from top to bottom):

1. Application layer: It provides specific support for each type of application.
2. Transport layer: It enables the communication between applications running in different terminals. In order to do so, this layer provides port numbers (from 0 to 65535) that are assigned to the different applications. It can be concurrently used by different applications. The TCP protocol is placed here, although there exist other alternatives to TCP, which are analyzed later in this section.
3. Internet layer: It makes the communication among different networks transparent. It allows the transport layer to see a unique virtual network. The main protocol of this layer is IP, although there are other associated protocols.
4. Network access layer: It manages the real underlying network, so that this layer is network dependent. There is no TCP/IP protocol for this layer. The only TCP/IP specification for this layer is the one relative to the access from the upper layer.

Figure 3.7 shows this layer structure and its related protocols. The figure indicates how the different TCP/IP protocols are implemented one on top of the other. The TCP/IP layers rest over a physical layer (specifying signals, data rate and related issues), which is also depicted in the figure.

In contrast to packet-switching networks, which are used for the transmission of data that do not have special timing requirements, circuit-switching networks have been traditionally used for data with real-time requirements. However, the improvements in computer processing speed, the development of powerful data compression techniques and the increase of available bandwidth are facts that enable the implementation of real-time applications over IP networks such as, for example, IP telephony (known as *voice*

Layer	Protocols	
Application	FTP, HTTP, Telnet, VoIP, etc.	
Transport	TCP	RTP
		UDP
Interconnection	IP	
Network interface	Ethernet, WiFi, ATM, etc.	
Physical	Characteristics of the transmission	

Figure 3.7 TCP/IP layer structure and common protocols

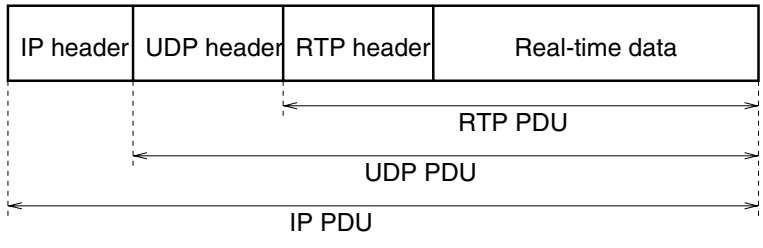


Figure 3.8 Format of a packet using RTP

over IP, VoIP) and RSR. There is a special protocol for data transmission with real-time requirements called real-time protocol (RTP). As shown in Figure 3.7, it relies on the user datagram protocol (UDP), and this one on the IP protocol. These protocols are briefly described in the following subsections. A packet using RTP has the structure shown in Figure 3.8. It contains three *embedded* PDUs (protocol data unit), one for each of the above mentioned protocols.

3.3.1.1 Interconnection Layer: IP Protocol

As mentioned earlier, the IP protocol (Postel, 1981a) is the main protocol of the Internet layer, and provides a single virtual network. In order to do this, this protocol utilizes IP addresses included in the IP header. Each connected entity has a single IP address, which consists of four bytes hierarchically organized: the first part of the address identifies the specific network, and the second part the specific host in that network. The IP address can be substituted by a given name. The association between a name and an IP address is carried out by a DNS server. The specific characteristics of the data transmission are specified by a protocol of the upper layer. The IP protocol is implemented not only at the end terminals but also in the routers. IP provides a *best effort* delivery service, that

is, tries (as best as it can) to deliver packets but it does not guarantee that packets are delivered, or that the delivery is made in a ordered manner, or that the payload data are error free (the IP header only contains a checksum for detection of errors in the header itself).

3.3.1.2 Transport Layer: TCP and UDP Protocols

TCP/IP networks offer two types of services: connection oriented and connectionless. The first one uses the TCP protocol (Postel, 1981b). Before transmitting any real data, sender and receiver exchange control packets to prepare the connection. TCP offers a reliable service over IP. This means that packets are received error free (there is a checksum for the whole TCP protocol data unit (PDU)) and ordered. In order to ensure this, when a packet is correctly received, the receiver sends a confirmation. If the source terminal does not receive any confirmation, it assumes that the packet was not received and transmits it again. TCP also performs flow control tasks to ensure that the data rate is suitable to the network congestion state and the transmitter and receiver processing capabilities.

The connectionless service is provided by the UDP protocol (Postel, 1980). In this case, there is no control packet exchange as in TCP. Packets are just sent when they are available. Thus, UDP provides a quite simple service with only two functions: demultiplexing and error check. The first function allows the delivery of a given packet to its corresponding application. In order to do this, the UDP header contains a destination port assigned to the specific application running in the destination computer and an origin port assigned to the application, in the source terminal, which will receive the answers. Second, while IP only checks whether its header is error free, UDP can optionally do it for the whole UDP datagram (i.e. including the payload). The applications that require real time usually employ UDP instead of TCP, since the latter protocol, although ensures the reception of all transmitted packets (by applying retransmission if necessary), does not ensure that packets are received in a reasonable time. For certain applications such as those transmitting video or audio (including RSR), it is more important to delay the incoming packets than receive all of them. UDP does not ensure a reasonable delay either, although it will be less than that involved in TCP. Besides, it can be considered that an outdated packet will not be necessary in a real-time application, so that no retransmission is required in order to avoid an unnecessary use of the network resources. Thus, UDP seems more appropriate for real-time applications since late or lost packets can be usually treated by different techniques that can anticipate or mitigate their negative effect.

There are other properties of UDP that can be useful in certain applications. For example, since it does not carry out any connection tracking, a server devoted to a certain application can support more active clients. Also, the TCP header introduces 20 overhead bytes, while UDP only adds 8 bytes, so it does contribute less to the network congestion.

3.3.1.3 Transport Layer: RTP

UDP does not have the mechanisms needed to solve difficulties such as packet time spread (jitter), packet losses, clock signal recovery or the synchronization of the different transmitted media (i.e. audio and video in a videoconference). The RTP protocol (Schulzrinne

et al., 1996), implemented on top of UDP, can partially solve these problems. The RTP header has the following fields:

- Payload type identifier: It specifies the CS used for the payload, in order to allow its decoding. The CS can be changed even in the middle of a session.
- Sequential number: It is used at the receiver to reconstruct the original order of the transmitted packets, since they can arrive disordered. It also allows the detection of lost packets.
- Time stamp: This field contains the sampling time of the first byte included in the payload. They allow the synchronization of packets from different sources.
- Synchronization source identifier (SSRC): It identifies the different sources participating in a RTP session.
- Contributing source list: It contains identifiers for the different sources that are mixed in an RTP packet, even though the mixed packet has a unique SSRC. This is applicable while using an RTP mixer, which can combine different sources.

We must highlight that RTP does not guarantee that packets will arrive on time and in order and does not offer any other guarantee of QoS. However, there is an additional protocol, the real-time control protocol (RTCP), which allows the monitoring of the QoS at the receiver and can inform the transmitter about it. This can be useful when the transmitter can adapt to the network conditions (i.e. available bandwidth, delay or time spread). The transmitter can also generate RTCP packets for source synchronization or including complementary data.

RTP is not a complete protocol, since it is designed for maximum flexibility. If RTP is to be used for a specific application, a *profile* document that defines the necessary attributes, modifications or extensions of RTP has to be followed. Also, it is required to specify a *payload format* for the real-time data. Regarding RSR, the IETF working group on audio and video transport has already issued an RFC recommendation for the *payload format* corresponding to the ETSI FE standard (Xie, 2003). This recommendation is treated in the last chapter of this book.

3.3.2 Degradation in IP Networks

Unlike wireless networks, bit errors in IP networks can be neglected for a number of reasons. The payload can include error detection and/or correction mechanisms. Additionally, the UDP header can include an error checking mechanism for the payload. Finally, it must taken into account that the underlying networks are usually reliable enough. The typical network has cable links with high channel SNR values. Therefore, although we can consider an IP network as a noiseless transmission medium, degradation can appear owing to the drawbacks inherent in its packet-switching structure, which are mainly latency, time spread and packet loss. The first two can be treated by means of introduction of decoding delays, which can absorb them. Late packets are considered as lost. Latency and time spread are important in applications such as VoIP, where it is intended to maintain the interactivity of a conversation. For an RSR application, an immediate answer from the server, although not crucial, is also desirable. Therefore, packet losses appear as the main source of degradation that can affect the recognition performance.

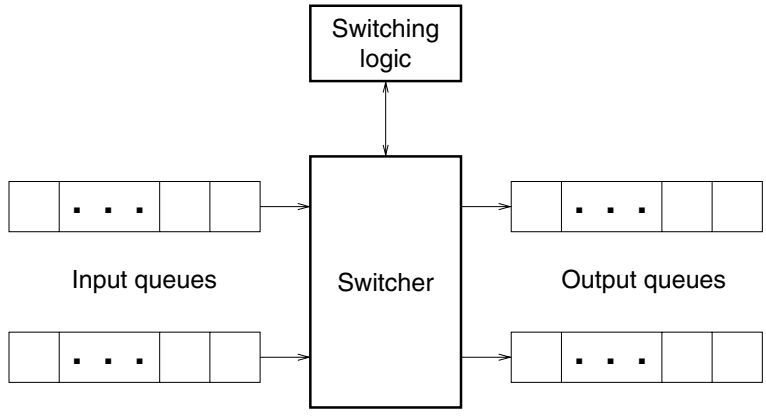


Figure 3.9 Diagram of a router device

In addition to late packets, packet losses may take place at the router devices. Figure 3.9 shows how this device works: it has several input and output queues, a packet-switcher and a switching logic that decides (utilizing the IP address) in which output queue a given packet must be placed. The function of the queues is to adapt the different transmission speeds of the interconnected networks. They can follow simple policies such as first-come-first-serve (FCFS) or others more sophisticated such as weighted fair queuing (WFQ), which involves an equitable sharing of the output. Packet losses can appear in the following circumstances (Kurose and Ross, 2003):

1. If the input flow is higher than the processing capacity of the switching logic, the packets are accumulated in the input queues. If this situation is prolonged for sufficient time, the input queues will overflow and some packets may be discarded. There are different policies to treat input overflow. For example, it is possible to eliminate the last input packet (drop-tail) or to remove (or mark for a possible removing) a given packet before an overflow occurs.
2. If the processing capacity of the switching logic is higher than the output flow speed, the packets could accumulate in the output queues. An overflow of the output queues can appear if the situation lasts for sufficient time. A typical situation at the output queues is as follows: let us consider a router with N input queues and N output queues. The switching rate is N times greater than the input and output rates. Thus, we can ensure that there will not be overflow at the input queues. But, in the worst case, the N input packets could be transferred to the same output queue, but only one of them can be transmitted. In the meantime, N new packets are transferred to output queues, and the previous situation could be repeated. This could exhaust the queue memory and some packets would be lost.

As mentioned earlier, the TCP protocol implements algorithms for packet retransmission in order to avoid packet losses. However, in a real-time application, retransmission is not suitable since a delayed packet can be considered to be the same as a lost packet. Besides, retransmission has the negative effect of unnecessarily consuming additional resources.

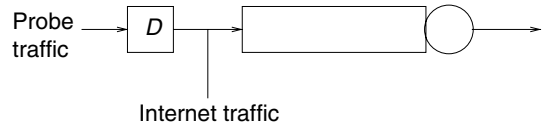


Figure 3.10 Bolot's model for packet loss and delay simulation

This is the reason RTP is usually implemented over UDP (instead of TCP), where there is no retransmission, and, therefore, a packet discarded by a router is a lost packet. Obviously, the problem of saturation that causes packet loss could be fixed by increasing the capacity of the networks in order to avoid router saturation. However, these network improvements usually involve the introduction of new applications that consume more resources and tend to produce new saturations. Several alternatives, such as dynamic bandwidth reservation (Shenker and Wroklawski, 1997), differentiated services (Blake *et al*., 1998) or private virtual networks (Fox and Gleeson, 1999; Gleeson *et al*., 2000) have been proposed, although their use is not currently extended to the Internet.

In the case of wireless networks or satellite links, the degradation factors (path loss, multipath, interferences, etc.) mentioned in the previous section, which may also produce packet losses, must also be considered. The specifications for wireless networks (network access and physical layers) usually include mechanisms to protect the transmitted data. Thus, transmission errors are treated at the lowest layers, thus avoiding the delay that would involve retransmission by TCP. For example, in an IEEE 802.11 LAN, when a station receives a packet, it sends back an acknowledgement packet to the source station. If the acknowledgement is not received before a given time interval, the packet is retransmitted. The 802.11 mechanism can optionally involve an exchange of four (instead of two) packets in order to obtain more reliability (Stallings, 2002). In the case of GPRS, acknowledged and unacknowledged modes (with and without retransmission) are possible (Bannister *et al*., 2004). On the other hand, a corrupted Bluetooth voice packet is simply discarded (SIG, 2001).

Packet losses can be treated by several techniques that, in fact, are not so different from those applied to channel errors in wireless channels. Thus, we have techniques oriented to anticipate the losses such as FEC or interleaving. Also, the resilient losses can be treated with mitigation techniques. These topics are treated in the following chapters.

Finally, we must point out that, in the same way as in RSR over wireless channels, the performance of an RSR system over IP can also be reduced by the codec employed for efficient transmission. Again, the interested reader may refer to the following chapter.

3.3.3 Lossy Packet Channel Models

The study of packet delays and losses began during the ARPANET project. This study was later used for the adjustment of the retransmission parameters of the TCP protocol. The implementation of real-time applications over IP has renewed this interest in the evaluation and modeling of these types of degradation.

An exhaustive analysis of packet delay and loss was carried out by Bolot (Bolot, 1993; Bolot *et al*., 1995). In (Bolot, 1993), Bolot examined a connection between France and United States of America with the objective of analyzing packet delays and losses. He

measured the round trip time (RTT) that an UDP packet required to make a round trip travel. Bolot concluded that the Internet traffic could be modeled as shown in Figure 3.10. The model consists of a server that processes packets stored in a queue, which has two different input flows. The first flow produces fixed size packets at a constant rate, every δ time units, that suffer a mean RTT delay D, and represents the application we want to test (probe traffic). The second flow represents the rest of the Internet traffic and generates variable size packets according to the distribution of the RTT time. Other conclusions of that work were as follows: (a) probe packets tend to accumulate (compression phenomenon), (b) queue delays fluctuate rapidly and (c) packet losses usually have a random nature, except in the case in which the probe traffic consumes a considerable part of the available bandwidth.

A first measure of the channel condition is the *a priori* probability of packet loss (*ulp*, *unconditional loss probability* or *packet loss rate*):

$$ulp = P(rtt_n = \infty) \tag{3.19}$$

where rtt_n is the RTT time of packet number n. This probability is increased when the packet generation rate is also increased (smaller δ). Another important conclusion of Bolot's work and other subsequent work is that packet losses tend to appear in bursts, that is, if packet number n is lost because of a network congestion, it is also quite likely that the subsequent packet $n + 1$ will be lost. Thus, it is also useful to know the *conditional loss probability* (*clp*), defined as

$$clp = P(rtt_{n+1} = \infty | rtt_n = \infty) \tag{3.20}$$

This probability is also higher for higher packet generation rates. This is a logical result, since it can be expected that a saturated network at time t will also be saturated at time $t + d$. More in-depth studies about the time correlation and burstiness exhibited by the IP channel can be found in Yajnik *et al*. (1999) and Jiang and Schulzrinne (2000). Packet loss bursts are more prejudicial than isolated losses. This is due to the fact that the protection and mitigation techniques for packet losses do not work well as the burst duration increases (Bolot *et al*., 1995).

In the following text, we will review some models for packet losses. Their suitability is commonly measured by their capacity to fit network packet traces (complete information of a network session in certain conditions). Again, a channel model can easily provide us with a whole set of channel conditions that allow us to perform a complete and controlled testing of our RSR system.

3.3.3.1 Bernoulli Model

This is the simplest loss model. In this model, packet loss is considered to be an i.i.d. random process $\{X_t\}_{t=1}^{\infty}$ where each random variable X_t is binary and can take values 0 (packet received) or 1 (packet lost). This model is defined by a loss probability $r = P(X_t = 1)$, which coincides with the *ulp* probability. The distributions of received (P_d) and loss ($\bar{P}_d$) runs with respect to length d (in number of packets) can be obtained as

$$P_d = r(1 - r)^{d-1} \tag{3.21}$$

$$\bar{P}_d = (1 - r)r^{d-1} \tag{3.22}$$

It is easily derived that the average loss burst duration is $d_{\text{loss}} = 1$ regardless of the particular value of r. Therefore, the model cannot properly model loss bursts. In Jiang and Schulzrinne (2000), it is shown that the Bernoulli model overestimates single loss probability and underestimates probabilities of longer loss bursts.

3.3.3.2 Gilbert–Elliot Models

A smart alternative to the Bernoulli model is to use the Gilbert–Elliot model already introduced for fading channel modeling (Figure 3.6). Error probabilities now mean loss probabilities. In IP environments, it is also common to use one of the following simplified versions:

1. Gilbert model (Gilbert, 1960): $p_{e0} = 0,\ p_e = \pi_1 p_{e1}$. Now, s_0 is a no-loss state.
2. Two-state Markov chain (Jiang and Schulzrinne, 2000): $p_{e0} = 0,\ p_{e1} = 1,\ p_e = \pi_1$. In this case, the Gilbert–Elliot model completely loses its "hidden" characteristic and becomes a simple Markov chain, where s_0 represents a no-loss (good) state and s_1 a lossy (bad) state. The two-state Markov chain model is also often referred to as *Gilbert model*, although we will reserve this name for the earlier "semihidden" model.

The two-state Markov chain is controlled by the transition probabilities between the two states, $p = P(X_t = 1|X_{t-1} = 0)$ and $q = P(X_t = 0|X_{t-1} = 1)$. The model is more flexible than the Bernoulli model since it has two parameters instead of only one. The probabilities of received packet bursts and lost packet bursts of length d are now

$$P_d = p(1-p)^{d-1} \tag{3.23}$$

$$\bar{P}_d = q(1-q)^{d-1} \tag{3.24}$$

The average loss burst duration is $d_{\text{loss}} = 1/q$. It can be observed that the two-state model becomes the Bernoulli model when $p + q = 1$. The *ulp* and *clp* probabilities are obtained from the model as

$$ulp = \frac{p}{p+q} \tag{3.25}$$

$$clp = 1 - q \tag{3.26}$$

The transition probabilities are usually computed to fit either given values of d_{loss} and *ulp* (using the above expressions) or a given network trace (Jiang and Schulzrinne, 2000).

Table 3.3 shows the WAcc results obtained with the ETSI DSR FE standard over an IP channel simulated with a two-state Markov chain for different channel conditions. The payload format is the one defined in Xie (2003) with two frames per packet. Frame losses are concealed by the mitigation algorithm included in the standard, based on frame repetition (this algorithm is detailed in Chapters 5 and 7). Under a no-loss transmission, the system yields WAcc = 99.04 %. Again, the result illustrates that random losses (small d_{loss}) are less damaging than bursty losses (larger d_{loss}). Thus, with $d_{\text{loss}} = 1$ and $ulp =$ 50 %, we can still obtain WAcc = 98.90 %, while with $d_{\text{loss}} = 16$ and $ulp = 10$ % (the opposite corner of the table), the WAcc is reduced to 90.77 %.

The two-state model is quite accurate for predicting short loss bursts (1–3 packets). However, it fails to model longer loss bursts, loss periods with lower loss density or the

Table 3.3 Word accuracies obtained with the DSR/FE standard over a two-state Markov chain channel (Aurora-2 task and testing only with clean sentences)

ulp (%)	d_{loss} (in number of lost packets)				
	1	2	4	8	16
10	98.98	98.61	96.50	93.42	90.77
20	98.96	98.08	94.02	87.28	83.03
30	98.88	97.56	90.92	81.43	74.87
40	98.92	96.81	88.18	76.61	66.89
50	98.90	96.21	84.56	70.36	59.63

distribution of loss-free periods (Jiang and Schulzrinne, 2000; Milner and James, 2004). This lack of accuracy is due to the simplicity of the two-state model, which only has two parameters. In order to obtain a more accurate fit of the channel statistics, a more complex model is required. For example, the Gilbert model adds a loss probability (or loss density) p_{e1} to the lossy state, which allows a better modeling of very short loss-free periods. The loss rate and average loss burst duration are now computed as

$$ulp = \frac{p}{p+q} p_{e1} \tag{3.27}$$

$$d_{\text{loss}} = \frac{1}{1-(1-q)p_{e1}} \tag{3.28}$$

3.3.3.3 Other First-order Markov Models

Instead of enhancing the two-state Markov chain by introducing a "hidden" characteristic to the lossy state (Gilbert model) or to both states (Gilbert–Elliot model), it is also possible to make the modeling more flexible by using Markov chains with more than two states. Thus, a solution that also allows a correct modeling of no-losses inside loss periods is the three-state Markov chain illustrated in Figure 3.11 (Milner and James, 2004). This model adds to the two-state model a third state s_2 that represents those no-losses inside loss periods. The loss rate and the average loss burst length can be obtained from the transition probabilities as

$$d_{\text{loss}} = \frac{1}{q+s} \tag{3.29}$$

$$ulp = \frac{rp}{r(p+q)+ps} \tag{3.30}$$

The model transition probabilities can be computed from the loss rate (*ulp*), the average loss burst length (d_{loss}), the average length of the loss-free periods (N_1), and the average length of the no-loss periods inside loss periods (N_3):

$$\begin{aligned} p &= \frac{1}{N_1} & r &= \frac{1}{N_3} \\ q &= \frac{p}{r-p}\left[\frac{r}{ulp} - \left(r + \frac{1}{d_{\text{loss}}}\right)\right] & s &= \frac{1}{d_{\text{loss}}} - q \end{aligned} \tag{3.31}$$

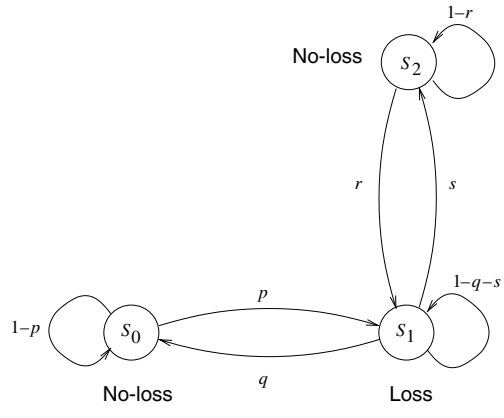


Figure 3.11 Three-state Markov chain

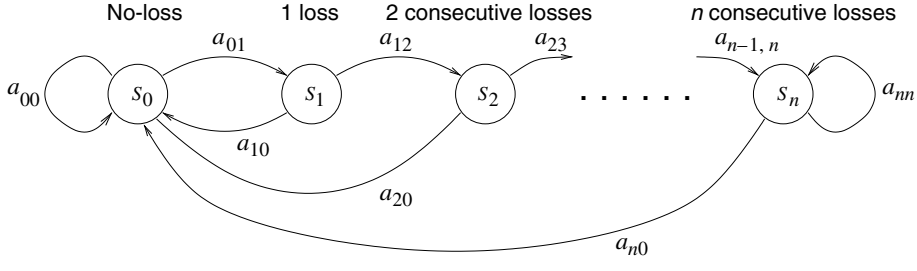


Figure 3.12 Extended Gilbert model

Sanneck and Carle (1996) proposes the use of an *extended Gilbert model* that requires $n + 1$ states. The model is depicted in Figure 3.12. As the basic two-state model, it has a "good" state s_0 corresponding to no-loss. The remaining states s_k ($k = 1, \ldots, n$) represent the k consecutive packet losses that have occurred. If the current state is s_k, there is a transition to state s_0 if the next packet is received. Otherwise, if the packet has been lost, the transition is made to s_{k+1}. If the loss burst is longer than n, there is a self-transition to s_n. This model becomes the two-state model for $n = 1$. The model can completely fit a given network trace (with loss bursts no longer than n packets) (Sanneck and Carle, 1996).

The other possibility that can capture both short duration consecutive losses and lower density loss events is the model shown in Figure 3.13 (ETSI, 2000b). It connects two two-state models representing burst (loss) and gap (no-loss) periods, respectively. The gap period is usually defined by a maximum loss rate or a minimum number of packets received consecutively. The burst period must start and end with a lost packet and the number of consecutive received packets must be less than a given value. The states shown in Figure 3.13 are explained below:

- State 0: Packet received within a gap.

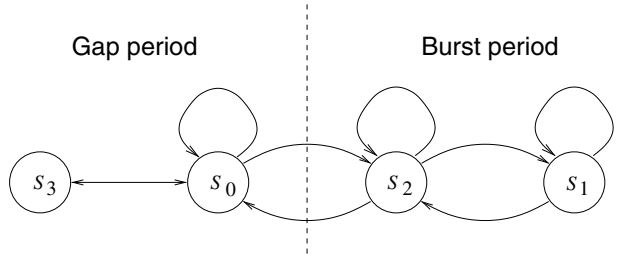


Figure 3.13 Four-state Markov channel modeling gap and burst periods

- State 1: Packet received within a loss burst.
- State 2: Packet loss within a loss burst.
- State 3: Isolated packet loss within a gap.

3.3.3.4 Higher-order Markov Models

It is also possible to use an nth-order Markov model for a more accurate representation of the correlations present in the packet loss process. In this case, the random variable X_t depends not only on the previous one, but also on the n previous ones. Thus, we must consider transition probabilities of the form $P(X_t|X_{t-1}, X_{t-2}, \ldots, X_{t-n})$. A state is defined by the set of n previous variables $(X_{t-1}, X_{t-2}, \ldots, X_{t-n})$, thus resulting in a total of 2^n states. Yajnik *et al*. (1999) showed that an order $n = 6$ was enough to correctly model most of the network traces they used.

3.3.4 Implementation of RSR Systems over Packet Networks

As mentioned earlier, the most suitable transmission scheme for an RSR application implemented over an IP network is the one based on UDP/RTP protocols. The main reason is that, although the latency is not crucial, we are usually interested in providing a service as interactive as possible. We have also seen that the price we pay for interactivity is the risk of packet loss. However, the speech signals and features are quite redundant, so it is possible to implement efficient loss concealment techniques as shown in Chapter 5. Other protocols required to establish an RSR session are SDP (session description protocol), which includes the type of media, the specific transmission protocol (i.e. RTP/UDP/IP), the media format and other information for the receiver, and the SIP (session initiation protocol), which allows to initiate, modify and terminate the session. Figure 3.14 shows how an RSR system can be integrated with the IP protocol stack. Both client and server have a metainformation block that contains data (keypad events, user equipment status, etc.) that may be useful for recognition or for dialog management (with the server application).

Another important issue is packetization, that is, what and how much data must be placed in each packet. The usual approach in to put an integer number of frames in each packet. In the previous example of Table 3.3 we chose two frames per packet, since the Aurora standards use frame pairs as transmission units protected by a 4-bit cyclic

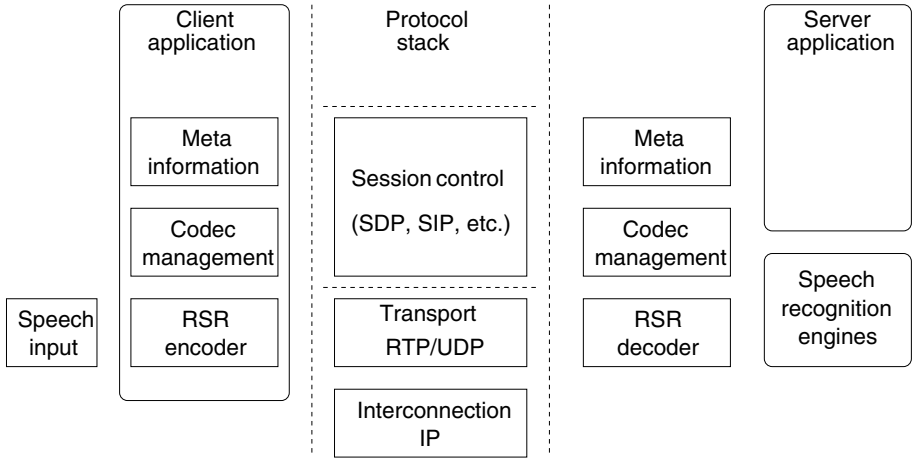


Figure 3.14 Integration of an RSR-based service in the IP protocol stack

redundancy check (CRC). The use of large packets is supported by the fact that more frames per packet involve less overhead due to headers, which may result in a smaller packet loss rate (Demichelis *et al*., 2005). However, large packets can increase end-to-end delay and generate long bursts of lost frames, which decrease the QoS and the recognition performance (see Table 3.3), respectively. This is the reason packets containing as small a number of speech frames as possible are usually employed and recommended (Xie, 2003; Xie and Pearce, 2005). In other cases, the underlaying network forces the packet size. This is the case of the DSR system described in Bawab *et al*. (2003), which is developed over a Bluetooth network with packet sizes of around 20 frames. As an alternative to the use of speech frames as data transmission units, it has also been proposed in Boulis *et al*. (2002) to distribute the speech frame data into several consecutive packets in order to provide a transmission scheme robust against packet losses by applying a suitable channel CS (for more details see Chapter 5).

3.4 The Acoustic Environment

ASR and, in particular, RSR systems trained using clean speech, acquired in a clean environment, may degrade significantly when used in real-word conditions because the clean speech models do not match speech acquired in real conditions. There are various reasons why real-word speech may differ from clean speech, and in this section we focus on those commonly referred to as acoustic environment.

The acoustic environment can be defined as the set of transformations that alter the speech signal from the time it leaves the speaker's mouth until it is recorded in digital form. Speech recorded in different acoustic environments has different characteristics, and the mismatch introduced by variations of the acoustic environment is the main source of speech recognition systems degradation. In this section, we focus on the two main sources of distortion: additive noise and channel distortion.

3.4.1 Additive Noise

The term *additive noise* refers to any unwanted signal that is added to the desired signal. The most common source of noise is the background noise. This is commonly referred as *acoustic noise* and is caused by air-conditioners, computer fans, moving cars, other background conversations, and so on. A signal captured with a close-talking microphone has little background noise. However, if a distant microphone is used instead, a great amount of background noise can be recorded along with the speech signal. Other types of additive noise are as follows:

1. Electromagnetic noise: caused by electric devices like radio and television emitters and receivers.
2. Electrostatic noise: generated by the presence of a voltage. Fluorescent lights are the main source of this type of noise.
3. Processing noise: resulting from the analog/digital processing of the speech signal. A common example is the quantization error in digital coding of speech signals.

3.4.1.1 White Gaussian Noise

The term white noise refers to a signal $x(t)$ with a flat power spectral density $S_{xx}(f)$. This means that $x(t)$ is an uncorrelated signal

$$R_{xx}(\tau) = E[x(t)x(t+\tau)] = \sigma^2\delta(\tau) \tag{3.32}$$

from which it is obtained

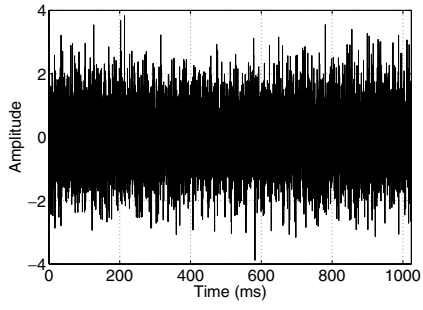
$$S_{xx}(f) = \int_{-\infty}^{\infty} R_{xx}(t)e^{-j2\pi ft}\,dt = \int_{-\infty}^{\infty} \sigma^2\delta(t)e^{-j2\pi ft}\,dt = \sigma^2 \tag{3.33}$$

being σ^2 the noise process variance. Such kind of noise can be generated by drawing samples from a given distribution $p(x)$, and therefore we can have different types of white noise depending on the selected distribution. When $p(x)$ is uniform, we have a uniform white noise. White Gaussian noise is obtained using a Gaussian probability distribution.

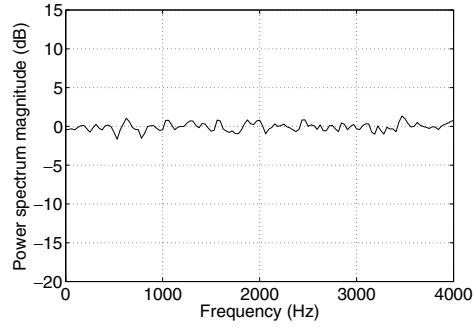
This definition can also be extended to discrete signals, and Figure 3.15 shows 8192 samples of a discrete white Gaussian noise $x(n)$ along with its power spectral density estimate $S_{xx}(k)$ and its autocorrelation function $R_{xx}(m)$.

3.4.1.2 Colored Noise

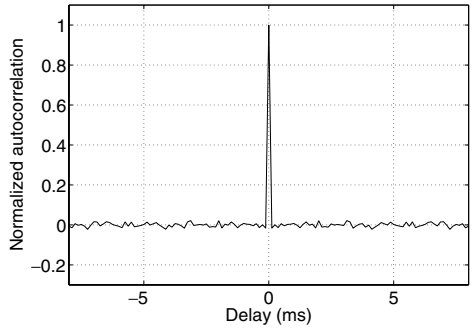
White noise seldom appears in practice. It is much more common to have noises with nonflat spectral shapes or *colored* noises. *Pink* noise is a particular class of colored noise with a low-pass nature, having most of its energy concentrated in the low-frequency region of the power spectrum. This kind of noise can be easily generated by passing a white noise through a linear filter with the desired spectral shape. Figure 3.16 shows a colored noise along with its autocorrelation and power spectral density. Now, the power



(a) Time domain



(b) Power Spectrum domain



(c) Autocorrelation domain

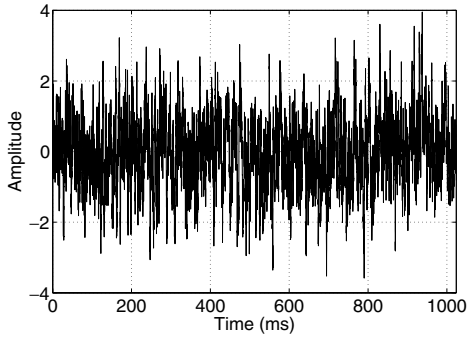
Figure 3.15 White Gaussian noise

spectrum is not flat, and samples are correlated in time, as can be observed in its autocorrelation function.

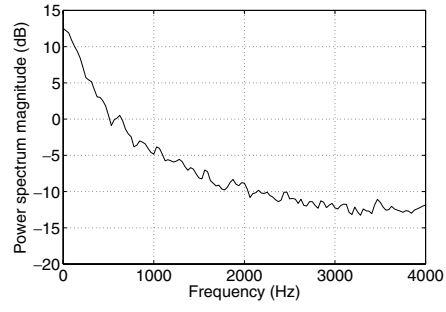
3.4.1.3 Stationary and Nonstationary Noises

A stationary noise is a signal whose characteristics are constant over time. Noises described earlier are stationary ones. If the characteristics of the noise vary over time, the noise is called *nonstationary*. In a strict sense there are no perfect stationary noises, but some noises are near-stationary like the aerodynamic noise produced in cars or noises from fans or air-conditioners. Figure 3.17 shows a segment of noise recorded inside a moving car along with its average power spectral density.

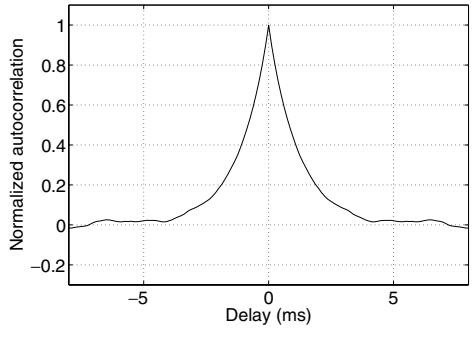
Noises caused by vehicles moving around are often nonstationary, for example, the noise recorded in a subway train station shown in Figure 3.18.



(a) Time domain

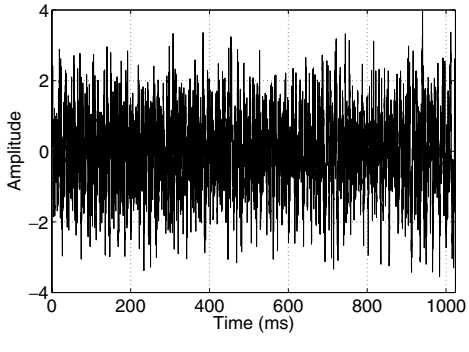


(b) Power spectrum domain

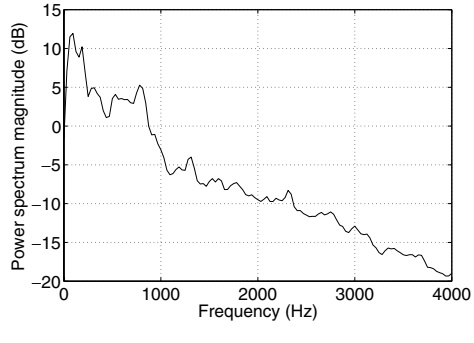


(c) Autocorrelation domain

Figure 3.16 Pink Gaussian noise

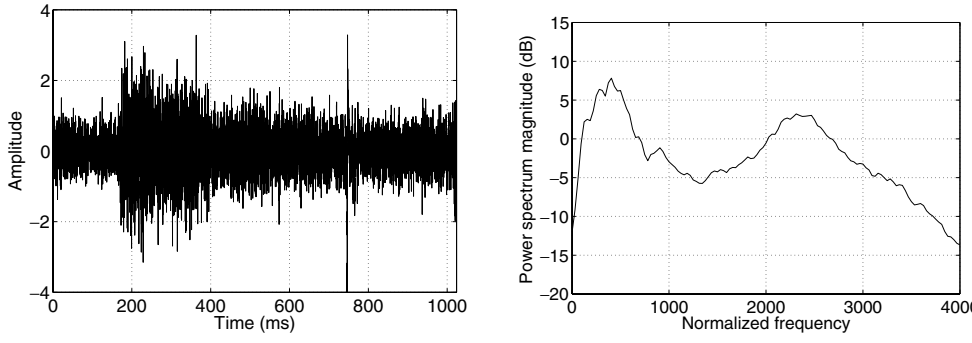


(a) Time domain



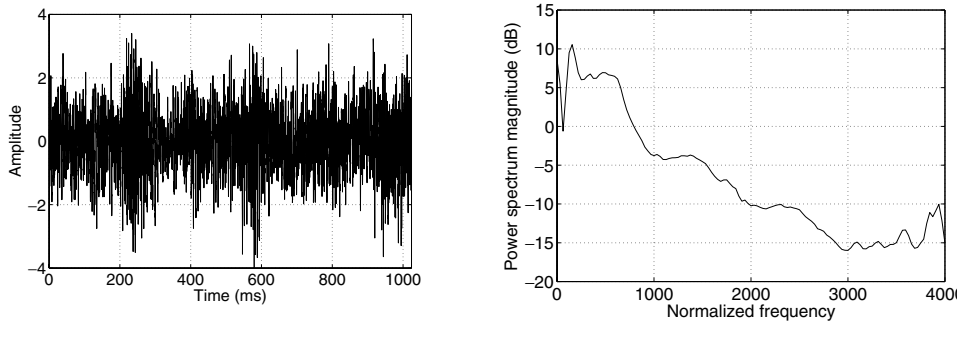
(b) Power spectrum domain

Figure 3.17 Car noise



(a) Time domain (b) Power spectrum domain

Figure 3.18 Subway noise



(a) Time domain (b) Power spectrum domain

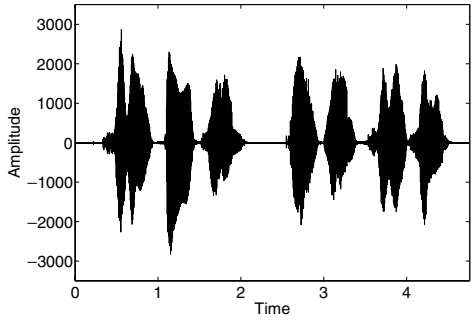
Figure 3.19 Babble noise

3.4.1.4 *Babble* Noise

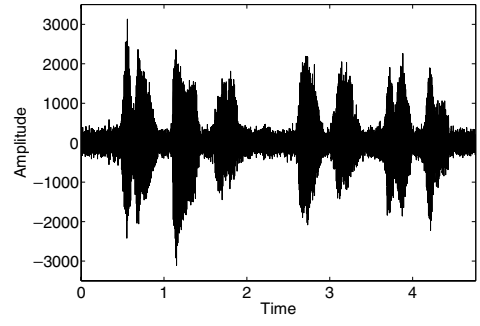
This is a particular type of noise formed by a mixture of speech signals. In the so-called cocktail party effect, a human listener can focus on a particular conversation out of many other simultaneous ones. The noise resulting from the sum of all the other conversations is commonly called *babble* noise. It is nonstationary and has a spectral shape close to the mean spectral shape of the human voice. An example is shown in Figure 3.19.

3.4.1.5 Noisy Speech

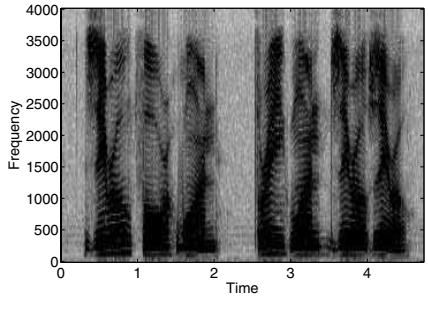
Figure 3.20 shows an illustrative example of how noise modifies the speech characteristics in both time and frequency domains. Plots (a) and (b) correspond to the clean speech signal and an artificially contaminated version obtained by adding a car noise at an average SNR of 10 dB; plots (c) and (d) show the corresponding spectrograms. Plot (e) shows a detail



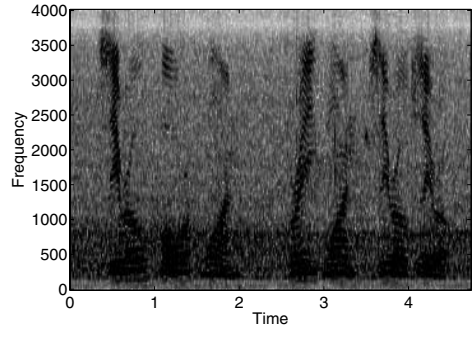
(a) Clean signal



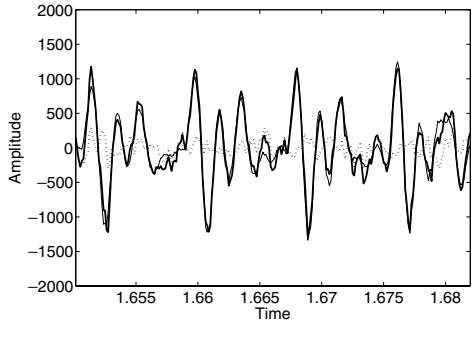
(b) Noisy signal



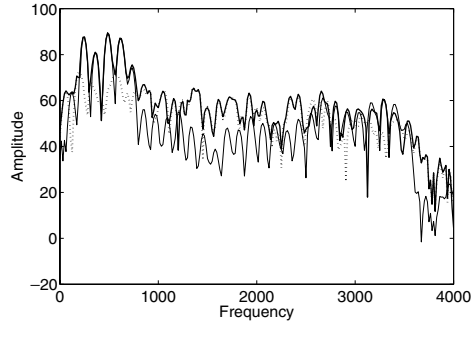
(c) Clean speech spectrogram



(d) Noisy speech spectrogram



(e) Speech and noise combination in time domain



(f) Speech and noise combination in frequency domain

Figure 3.20 The effects of noise in time and frequency domains. (a) Clean speech. (b) Speech with car noise added at an average SNR of 10 dB. (c) Spectrogram of clean speech. (d) Spectrogram of noisy speech. (e) A detail of the combination of speech and noise (dotted line) in time domain. (f) Power spectral density showing the combination of speech (bottom solid line) and noise (dotted line) to give the noise signal (top solid line) in frequency domain

of the clean signal, the noisy signal and the added noise in time domain, and plot (f) shows the corresponding power spectral density estimates.

From these plots, it is evident that not all speech values are equally affected by noise. In the time domain plots, the low amplitude parts of the signal are almost completely hidden by noise, while high amplitude values are less affected. This effect is also noticeable in the spectral domain. Consider for example the segment between 1 and 2 seconds in the spectrograms; the 1500–3000 Hz band is hidden by noise, while the lower band up to 1500 Hz is still clearly visible. This is because, in the first case, the noise power is greater than the speech power, while the inverse occurs in the second case.

3.4.2 Channel Distortion

Channel distortion is the second source of acoustic distortion and is caused by a change in the spectral shape of the speech signal due to the frequency response of the acoustic transmission channel. Some sources of channel distortion are as follows:

1. Analog transmission channel. This situation occurs for example in analog telephony, where different subscriber loops have different frequency responses.
2. Microphone characteristics. The frequency response of the microphone used to acquire the speech signal is another source of channel distortion, as different types of microphones have different frequency responses.
3. Signal conditioning. Digital codecs usually have an input filter to condition the signal before it is processed. The difference in frequency responses of these filters is another source of channel distortion. An example is shown in Figure 3.21 where the frequency response of G712 and motorola integrated radio system (MIRS) input filters are shown.

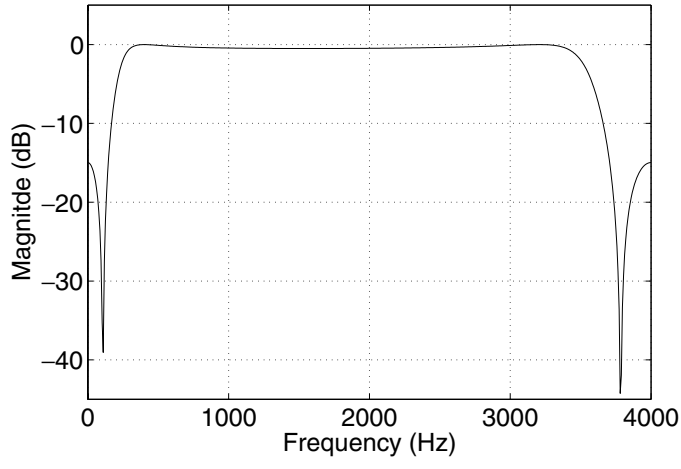
3.4.2.1 Reverberation

Unless the microphone and the speaker are located in free space or in an anechoic chamber, the microphone picks, along with the signal from the direct acoustic path, signals reflected in the nearby obstacles (walls, or other objects in the room). An example of this situation is depicted in Figure 3.22. This situation is similar to the multipath effect in radio transmission described in Section 3.2.3.

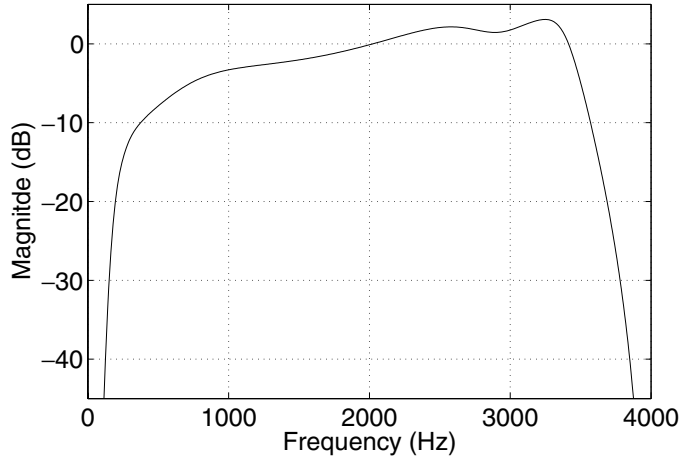
The received signal at the microphone is the sum of the signals received through the direct path and all indirect paths. Let us denote the signal traveling along the direct path of length d_0 as $x_0(t)$. This signal is received with attenuation inversely proportional to the distance, and with a delay given by $\tau_0 = d_0/c$, c being the speed of sound. The signal received through an indirect path $x_k(t)$ is received with a grater delay and lower amplitude due to the longer length of the path. The attenuation in this case is due to not only propagation but also the partial absorption that occurs in every reflection. Taking into account the combined effect of delay and attenuation, we can write down the following expressions for the received signal along the direct path ($x_0(t)$) and any of the indirect paths ($x_k(t)$):

$$x_0(t) = (A/d_0)x(t - \tau_0) \tag{3.34}$$

$$x_k(t) = r_k(A/d_k)x(t - \tau_k) \tag{3.35}$$



(a) G712 filter



(b) MIRS filter

Figure 3.21 Frequency response of G712 and MIRS input filters

with r_k being the combined attenuation due to reflections along the kth path. The received signal is a combination of attenuated and delayed versions of the original signal:

$$x(t) = x_0(t) + \sum_{k=1}^{\infty} x_k(t) = (A/d_0)x(t - \tau_0) + \sum_{k=1}^{\infty} r_k(A/d_k)x(t - \tau_k) \tag{3.36}$$

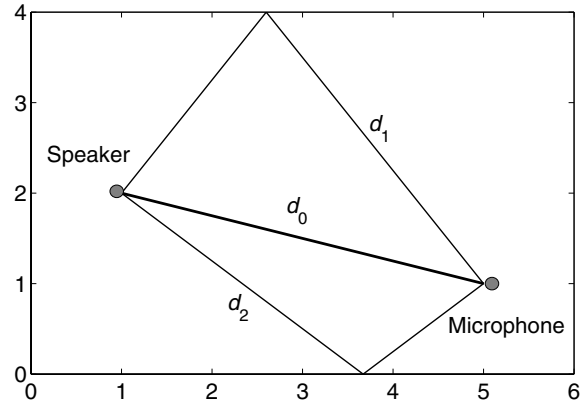


Figure 3.22 Schematic situation causing reverberation in a room showing the direct path d_0 and two indirect paths d_1 and d_2

In terms of the signal received along the direct path $x_0(t)$, we can write

$$x(t) = x_0(t) + \sum_{k=1}^{\infty} r_k(d_0/d_k)x_0(t - (\tau_k - \tau_0)) = h(t) * x_0(t) \tag{3.37}$$

The impulse response of the reverberating system $h(t)$ can therefore be expressed as

$$h(t) = 1 + \sum_{k=1}^{\infty} r_k(d_0/d_k)\,\delta(t - (\tau_k - \tau_0)) \tag{3.38}$$

and the frequency response of the system is, therefore,

$$H(f) = 1 + \int_{-\infty}^{\infty} h(t)e^{-j2\pi ft}\,dt = 1 + \sum_{k=1}^{\infty} r_k(d_0/d_k)e^{-j2\pi f(\tau_k - \tau_0)} \tag{3.39}$$

In the example of Figure 3.22, considering the dimensions in meters we can compute the direct and indirect path lengths as $d_0 = 4.12$ m, $d_1 = 6.4$ m, $d_2 = 5$ m, which leads us to delays of $\tau_0 = 12.45$ ms, $\tau_1 = 19.33$ ms, $\tau_2 = 15.11$ ms. If we consider an attenuation factor $r_1 = r_2 = 0.75$, we have finally the following impulse response for this simple system:

$$h(t) = 1 + 0.48\,\delta(t - 6.88 \times 10^{-3}) + 0.62\,\delta(t - 2.66 \times 10^{-3}) \tag{3.40}$$

The corresponding frequency response is plotted in Figure 3.23, which illustrates the frequency domain effect of reverberation.

3.4.3 A Model of Environment

As described in the previous subsections, the effect of the environment on the speech signal can be described by two main contributions: the additive noise and the channel

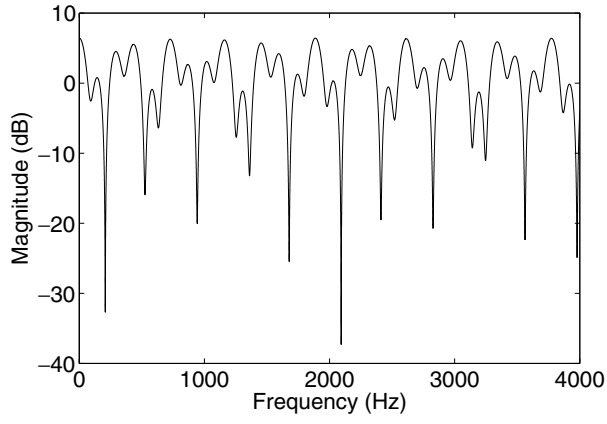


Figure 3.23 Frequency response of the reverberating system of Figure 3.22

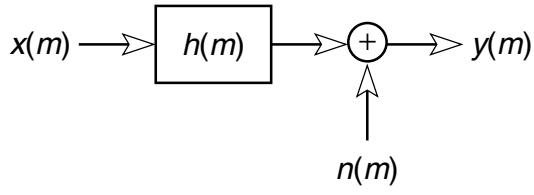


Figure 3.24 Model of acoustic environment

distortion. Therefore, we can establish the following model of the distortion suffered by the speech signal due to the environment:

$$y(m) = x(m) * h(m) + n(m) \tag{3.41}$$

where $y(m)$ are the distorted samples, $h(m)$ is the impulse response of the channel and $n(m)$ is the additive noise (Figure 3.24). In the time domain, the distorted speech signal is obtained by the convolution of the original signal with the impulse response of the channel plus an additive noise term. In the power spectrum domain, Equation (3.41) becomes

$$|Y(k)|^2 = |X(k)|^2|H(k)|^2 + |N(k)|^2 + 2Re\{X(k)H(k)N^*(k)\} \tag{3.42}$$

The last term in this equation is small and has an expected value of zero. If we use a filterbank approach for the analysis of the speech signal, the averaged value of the cross-product is small and can be neglected. Therefore, the output energy of filter b is

$$|Y_b|^2 \approx |X_b|^2|H_b|^2 + |N_b|^2 \tag{3.43}$$

In this approach, we have also made the implicit assumption that the impulse response of the channel is shorter than the window length used for the estimation of the spectra.

As discussed earlier, the most frequently used features in speech recognition systems are based on the MFCC cepstrum, which is defined as the DCT transformation of the

log-energies at the output of a mel-scaled filterbank. In order to apply the environment model of Equation (3.41) to these features, we first define the following vectors:

$$\mathbf{x} = [\log|X_1|^2 \ \log|X_2|^2 \ldots \log|X_M|^2] \tag{3.44}$$

$$\mathbf{y} = [\log|Y_1|^2 \ \log|Y_2|^2 \ldots \log|Y_M|^2] \tag{3.45}$$

$$\mathbf{h} = [\log|H_1|^2 \ \log|H_2|^2 \ldots \log|H_M|^2] \tag{3.46}$$

$$\mathbf{n} = [\log|N_1|^2 \ \log|N_2|^2 \ldots \log|N_M|^2] \tag{3.47}$$

where M is the number of filters in the bank. The relations for the log-energies at the output of the filterbank are

$$\mathbf{y} = \log(\exp(\mathbf{x} + \mathbf{h}) + \exp(\mathbf{n})) \tag{3.48}$$

$$= \mathbf{x} + \mathbf{h} + \log(1 + \exp(\mathbf{n} - \mathbf{x} - \mathbf{h})) \tag{3.49}$$

The noisy speech values $\mathbf{y}$ are obtained as a nonlinear combination of the clean speech $\mathbf{x}$, the noise $\mathbf{n}$ and the channel $\mathbf{h}$. This expression can be used to accurately predict the effect of the environment on the speech signal. As an example, Figure 3.25 shows the time evolution of the log-energy at the output of the fifth filter of a bank of 23 mel-spaced filters in the range 0–4000 Hz for the same signal as used in the example of Figure 3.20. The continuous lines correspond to the clean and noisy versions of the utterance, and the dots are the predicted values computed using Equation (3.49). The main effect of the nonlinear combination of noise in the logarithmic filterbank energies (log-FBE) domain is the range reduction and a shift in the mean value.

The MFCC coefficients are obtained as linear combinations of log-energies, and therefore they also suffer a nonlinear distortion. The main effects are again a range reduction and a shift in the mean values. Figure 3.26 shows an example of the time evolutions of the first two cepstral coefficients of clean and noisy speech. Using (3.49) and denoting

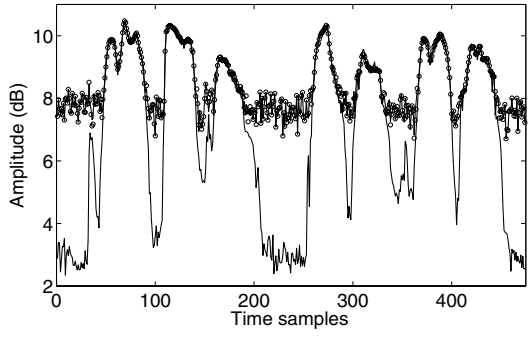
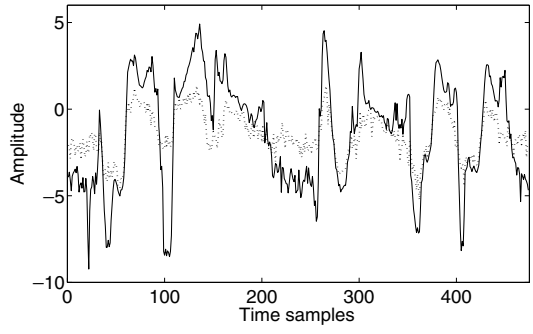
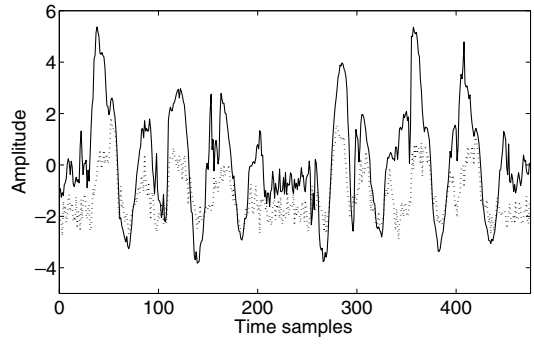


Figure 3.25 Example of using the environment model to predict the effect of an additive noise in the log-FBE domain. Bottom and top solid lines correspond to the time evolutions of clean and noisy speech log-FBEs, respectively, while dots are the predicted values



(a) Cepstral coefficient c_1



(b) Cepstral coefficient c_2

Figure 3.26 Example of the noise effect on the MFCC domain. The plots correspond to the first and second cepstral coefficients of the clean speech (solid line) and the noisy speech (dotted line)

the DCT transformation matrix by $\mathbf{C}$, we can write the following relation for the MFCC coefficients:

$$\mathbf{x}^c = \mathbf{C}\mathbf{x} \quad \mathbf{y}^c = \mathbf{C}\mathbf{y} \quad \mathbf{h}^c = \mathbf{C}\mathbf{h} \quad \mathbf{n}^c = \mathbf{C}\mathbf{n} \tag{3.50}$$

$$\mathbf{y}^c = \mathbf{C}\,\log(\exp(\mathbf{C}^{-1}\mathbf{x}^c + \mathbf{C}^{-1}\mathbf{h}^c) + \exp(\mathbf{C}^{-1}\mathbf{n}^c)) \tag{3.51}$$

$$\mathbf{y}^c = \mathbf{x}^c + \mathbf{h}^c + \mathbf{C}\,\log(1 + \exp(\mathbf{C}^{-1}(\mathbf{n}^c - \mathbf{x}^c - \mathbf{h}^c)) \tag{3.52}$$

3.4.4 Probability Distributions of Noisy Speech Features

Most speech recognition systems are based on a statistical description of MFCC-based speech features. Therefore, it is interesting to understand how the probability distributions

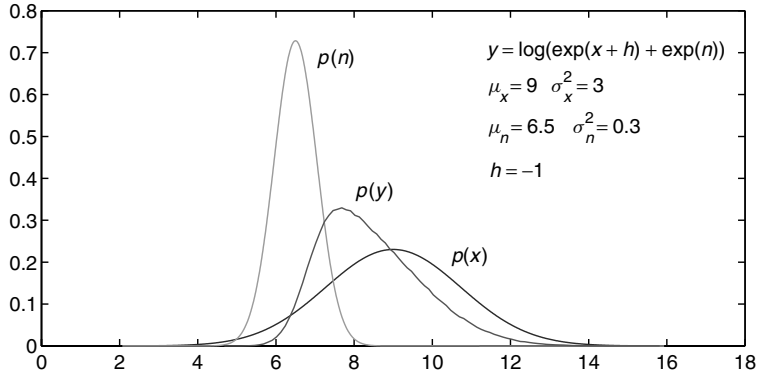


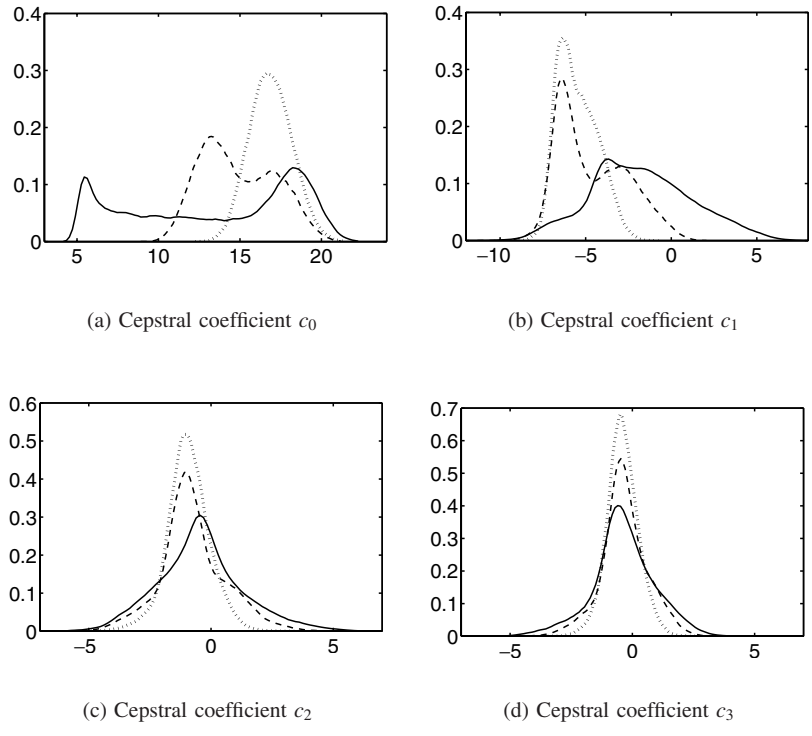
Figure 3.27 Probability density transformation for a simulated environment distortion with a fixed channel and random noise

of this features are modified by the effect of the environment. In previous sections, we have qualitatively shown that the main effects of noise in the log-FBE and MFCC domains are range compression and a shift in the mean values of the features. In this section, we present a more detailed description of how the feature probability distributions are affected by the acoustic environment. We will apply the environment model described by Equations (3.49) and (3.52).

The key point to note here is that the environment model is a nonlinear transformation in the log-FBE domain, which modifies not only the location (mean value) and scale (variance) but also the shape of the probability distributions of features. To illustrate this effect, let us consider that both clean speech **x** and noise **n** (log-FBE features) have Gaussian distributions at the output of a particular filter in a filterbank and that a constant channel **h** is also present. Figure 3.27 shows this situation where the means for the clean speech and noise are $\mu_x = 9$ and $\mu_n = 6.5$ (giving a signal-to-noise ratio of $SNR \equiv (\mu_x - \mu_n) = 2.5$), with variances $\sigma_x = 3$ and $\sigma_n = 0.3$. This is a common situation when Gaussian mixtures are used to model speech.

Note that although both clean speech and noise have Gaussian distributions ($p(x)$ and $p(n)$), the resulting probability distribution (simulated by Monte Carlo method) is no longer Gaussian. The predominant effects are the shift of the mean value and the reduction of the variance, which affect the two first moments of the probability distributions. Nevertheless, the nonlinear transformation also modifies higher-order statistical moments affecting the shape of the distributions. These modifications of the probability distributions of the speech are the main cause of the degradation suffered by speech recognition systems in noise conditions, because the probability distributions estimated using clean speech do not accurately represent noisy speech because of these effects.

Figure 3.28 shows the probability distributions of the first four MFCC coefficients in different noise conditions. The probability distributions have been approximated with histograms using data from 1001 utterances in different SNR conditions. The mean shift and the variance reduction are clearly noticeable, as also the change in the shape of the distributions caused by the modifications of higher-order moments of the probability



(a) Cepstral coefficient c_0

(b) Cepstral coefficient c_1

(c) Cepstral coefficient c_2

(d) Cepstral coefficient c_3

Figure 3.28 Probability distributions of the first four cepstral coefficients of clean speech (solid lines) and speech plus noise at an SNR of 20 dB (dotted lines) and 5 dB (dashed lines). Histograms obtained from 1001 utterances

distributions due to the nonlinear transformation. The noise effects are more noticeable for the low-order coefficients.

3.4.4.1 Piecewise Linear Approximation of the Environment Model

Knowing the probability distributions of speech and noise and the channel term, we can, in theory, obtain the exact probability distribution of the noisy speech using the log-FBE model Equation (3.49). In the one-dimensional case, the derivation is as follows. First consider that speech and noise are statistically independent and therefore we can write

$$p(x, n) = p_n(n)p_x(x) \tag{3.53}$$

Next, we can express the joint probability of y and n as

$$p(y, n) = p_n(n)\frac{p_x(\log(e^y - e^n) - h)}{|\partial y/\partial x|} \tag{3.54}$$

If we introduce the partial derivative

$$\frac{\partial y}{\partial x} = \frac{\partial(\log(e^{(x+h)} + e^n))}{\partial x} = \frac{e^{(x+h)}}{e^{(x+h)} + e^n} = 1 - e^{(n-y)} \tag{3.55}$$

in the previous equation, we obtain

$$p(y,n) = (1 - e^{(n-y)})\, p_n(n)\, p_x(\log(e^y - e^n) - h) \qquad \forall y \geq n \tag{3.56}$$

Finally, integrating on n, we obtain the following probability distribution for the noisy speech:

$$p(y) = \int_{-\infty}^{y} (1 - e^{(n-y)})\, p_n(n)\, p_x(\log(e^y - e^n) - h)\, dn \tag{3.57}$$

which has no closed form even in the case of Gaussian probability distributions for both speech and noise. Although the above relation gives us an exact solution for the probability distribution of the noisy speech in the log-FBE domain, its mathematical complexity limits its practical utility.

Most speech recognition systems use Gaussian mixtures to model speech observations, and therefore it will be of great interest to develop the relations between Gaussian mixtures models of clean and noisy speech. In the following text, we will restrict the discussion to the one-dimensional case for simplicity. Consider that the clean speech model is modeled by a Gaussian mixture of the form

$$\mathcal{N}(x;\, \mu, \sigma^2) = \frac{e^{-(x-\mu)^2/2\sigma^2}}{\sqrt{(2\pi\sigma^2)}} \tag{3.58}$$

$$p(x) = \sum_{m=1}^{M} a_k\, p_{xk}(x) = \sum_{m=1}^{M} a_k \mathcal{N}(x;\, \mu_{xk}, \sigma_{xk}^2) \tag{3.59}$$

and that noise is also modeled using a single Gaussian

$$p(n) = \mathcal{N}(n;\, \mu_n, \sigma_n^2) \tag{3.60}$$

The noisy speech distribution will therefore result in a mixture of probability density functions:

$$p(y) = \sum_{m=1}^{M} a_k\, p_{yk}(y) \tag{3.61}$$

where each density $p_{yk}(y)$ is the result of the combination of the noise density $p(n)$ with the corresponding speech density $p_{xk}(x)$. If we consider that both σ_{xk} and σ_n are small, we can use a linear approximation of Equation (3.49) around the mean values μ_{xk} and μ_n. Since a linear combination of two Gaussian random variables is also Gaussian, this approach will allow us to compute a Gaussian approximation for $p_{yk}(y)$, in such a way that we will finally have a Gaussian mixture approximation for the probability distribution of the noisy speech:

$$p(y) = \sum_{m=1}^{M} a_k\, p_{yk}(y) \approx \sum_{m=1}^{M} a_k \mathcal{N}(y;\, \mu_{yk}, \sigma_{yk}^2) \tag{3.62}$$

To obtain the values of the means μ_{yk} and variances σ^2_{yk} of the noisy speech, we first need to develop the linear approximation of the environment model. We will again restrict the development to the one-dimensional case for simplicity.

Let us define the following two functions:

$$y = x + g(x, n, h) \tag{3.63}$$

$$g(x, n, h) = h + \log(1 + e^{(n-x-h)}) \tag{3.64}$$

$$f(x, n, h) = \frac{1}{1 + e^{(x+h-n)}} \tag{3.65}$$

With these definitions, the partial derivatives of function $g(x, n, h)$ can be expressed as follows:

$$\frac{\partial g(x, n, h)}{\partial n} = f(x, n, h) \tag{3.66}$$

$$\frac{\partial g(x, n, h)}{\partial x} = -f(x, n, h) \tag{3.67}$$

The first terms in the Taylor expansion of Equation (3.64) around the mean values of the clean speech and noise are

$$\begin{aligned} y = x &+ g(\mu_{xk}, \mu_n, h) \\ &- f(\mu_{xk}, \mu_n, h)(x - \mu_{xk}) + f(\mu_{xk}, \mu_n, h)(n - \mu_n) \\ &+ \cdots \end{aligned} \tag{3.68}$$

Using this relation, it is easy to obtain the mean and variance values of the noisy speech Gaussian in terms of the mean and variances of the clean speech and noise Gaussians:

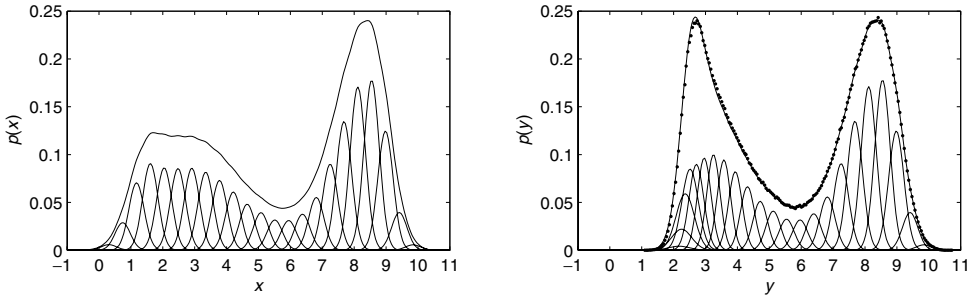
$$\mu_{yk} = E[y] \approx \mu_{xk} + g(\mu_{xk}, \mu_n, h) \tag{3.69}$$

$$\sigma^2_{yk} = E[(y - \mu_{yk})^2] \approx [1 - f(\mu_{xk}, \mu_n, h)]^2 \sigma^2_{xk} + [f(\mu_{xk}, \mu_n, h)]^2 \sigma^2_n \tag{3.70}$$

These relations explain both the shift of the mean and the scaling of the variance described in the preceding sections.

As an example to show the accuracy of this approximation, consider a Gaussian mixture like the one shown in Figure 3.29(a), and consider that noise is modeled as a single Gaussian of mean $\mu_n = 2.0$ and standard deviation $\sigma_n = 0.4$, with no channel distortion $h = 0$. Figure 3.29(b) shows the Gaussian mixture approximation for the noisy speech distribution obtained using Equations (3.69) and (3.70). The dots in this figure are values obtained using Monte Carlo simulation of the full environment model.

As a conclusion, if we can build an accurate model of the clean speech (i.e. a Gaussian mixture) and if we know or can estimate the statistics of the noise distribution (i.e. its mean an variance) and the channel term h, we can compute an accurate approximation of the distribution of the noisy speech using the procedure described earlier to transform



(a) Clean speech distribution

(b) Noisy speech distribution

Figure 3.29 Example of using the piecewise approximation of the environment function. (a) The clean speech probability density modeled as 20 Gaussians mixture. (b) The transformed mixture components and the corresponding probability density after adding a Gaussian noise of mean $\mu_n = 2.0$ and standard deviation $\sigma_n = 0.4$; the dots are values obtained using Monte Carlo simulation

the distribution of the clean speech, obtaining the corresponding distribution for the noisy speech.

This procedure has been formulated in the log-FBE domain, but it can be easily modified to work in the MFCC domain. Suppose we have a Gaussian mixture modeling the clean speech distribution in the MFCC domain and we also know the noise statistics in this domain. We can use the inverse DCT to transform means and variances to the log-FBE domain, combine the Gaussians in this domain and use a direct DCT to have the final values back in the MFCC domain. The transformation of mean values is not a problem, but to transform the variances, it must be taken into account that if the MFCC Gaussians have diagonal covariance matrices the inverse DCT results in nondiagonal covariance matrices in the log-FBE domain. One simple solution is to discard the nondiagonal elements in the covariance matrices after applying the inverse DCT. A more rigorous approach is to extend the earlier method to deal with full covariance matrices.

3.4.4.2 Information Loss Due to Noise

In the previous section, we have discussed a functional form of the environment distortion useful to predict the transformation of the probability distributions of speech. It has been shown that this is an accurate model of the environment but, despite this fact, even when we know the parameters of the transformation, it is not possible to recover the clean speech features from the noisy ones because of the random nature of the transformation.

The environment model can be used to predict the shift of the mean values and the scaling of variances, but it cannot be inverted because of the random behavior of the noise. The plot in Figure 3.30 shows the nonlinear random transformation due to a Gaussian noise in the log-energy domain of mean $\mu_n = 3$ and standard deviation $\sigma_n = 0.4$. The continuous line is the mean nonlinear transformation, while the dots show the transformed data. This mean transformation can be inverted to obtain the clean speech expected value given the noisy observation, but this estimation has a degree of uncertainty which depends

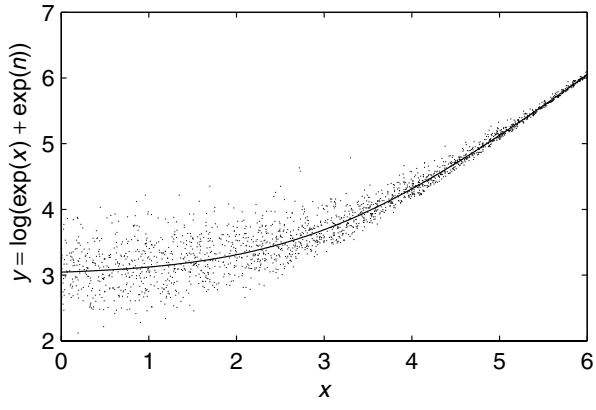


Figure 3.30 Random transformation due to noise

on the SNR. For values of y much greater than the noise mean value, there is a low level of uncertainty about the corresponding x value. On the other hand, there is a great uncertainty for values of y near the noise mean level. Consequently, a random noise always causes an unrecoverable information loss and the clean speech values can only be recovered from noisy observations with a certain degree of uncertainty.

4

Speech Compression and Architectures for RSR

4.1 Introduction

We have already seen in the Chapter 1 that there are two main architectures for RSR over digital channels: NSR and DSR. The basic schemes of both architectures are shown in Figures 1.3 and 1.4. These architectures differ in both client and server, because of the different formats used to transmit speech. In NSR, a speech coder is used at the client, and the received bitstream is usually decoded in order to reconstruct the speech signal, which is analyzed to provide a suitable parametrization for the speech recognizer. It is also possible to obtain this parametrization directly from the received bitstream. On the other hand, in DSR that parametrization is directly obtained at the client, so that no decoding and analysis is required at the server. This format difference also means that speech is transmitted with different bitrates and different channel error protection levels. An example of the DSR/NSR dichotomy arises with the implementation of an RSR system over GSM. We could implement it either as an NSR system using the speech TCH channel at a raw bitrate of 12.8 kbps (22.8 kbps including channel coding), or as a DSR system using the data TCH channel at bitrates of 4.8 or 9.6 kbps (22.8 kbps after channel coding). The NSR option has the advantage that it is possible to use a mobile phone without any special feature. The DSR option requires a mobile device with DSR capabilities, but it avoids the distortion introduced by the codec operation and uses a transmission mode more robust against channel errors.

In this chapter, we will review the main issues of NSR and DSR. In both cases, we study the different encoding (and decoding) techniques and the performance of the RSR system that uses them. We will pay special attention to the degradation introduced by the coding process. The degradation level depends on the speech coding technique, on the bitrate and, in the case of NSR, on the number of times speech is coded and decoded (e.g. a conversation between two mobile phones can involve two coding/decoding operations). The NSR approach is introduced under the two different variants mentioned above. The first one performs recognition from the decoded speech, while the second one avoids the reconstruction step. The speech coding concepts involved by NSR are introduced in

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

the following section and will also be useful for feature compression in DSR. We will conclude the chapter with a comparison of the NSR and DSR architectures.

4.2 Speech Coding

The goal of speech coding is to represent the signal employing as few bits as possible but maintaining a certain level of quality in the decoded speech signal. In general, coding techniques are based on the reduction of the signal redundancies, so that the residual signal (after redundancy reduction) can be encoded with fewer bits than the original one. In order to evaluate a speech coder, we must take into account several issues:

- Resulting bitrate
- Quality of the decoded speech
- Computational cost
- Introduced delay
- Robustness against acoustic and channel degradation

The objective quality measures such as the SNR can be helpful, but they cannot reflect how a decoded signal is perceived. A common measure of subjective quality of the decoded speech is the mean opinion score (MOS) that ranges from 1 (unacceptable) to 5 (excellent). . It is also possible to apply an objective measure such as the *perceptual estimation of speech quality* (PESQ, ITU-T recommendation P.862 (ITU-T, 2001)), which takes into account how we perceive speech.

Speech coders can be classified in waveform, parametric and hybrid coders. They are summarized in the following subsections.

4.2.1 Waveform Coders

Waveform coders try to obtain a decoded signal that reproduces the original input signal. They operate at medium bitrates (around 2 bits per sample) and show an acceptable robustness against acoustic and channel noise. They can be implemented in both time and frequency domains.

4.2.1.1 Waveform Coders in the Time Domain

A well-known example is the *pulse code modulation* (PCM) coder used for digital telephony (ITU-T standard G.711) (ITU-T, 1988a). The speech samples are represented by binary codes. The bandwidth is 200–3400 Hz and the sampling frequency, 8 kHz. It also uses a logarithmic quantization (μ-law in USA and A-law in Europe) with 8 bits. MOS is over 4. The final bitrate is 64 kbps.

An alternative technique is the *differential pulse code modulation* (DPCM) depicted in Figure 4.1. The DPCM coder transmits a quantized version $\hat{e}(n) = Q[e(n)]$ of the prediction error or *residual* obtained as

$$e(n) = s(n) - \bar{s}(n) \tag{4.1}$$

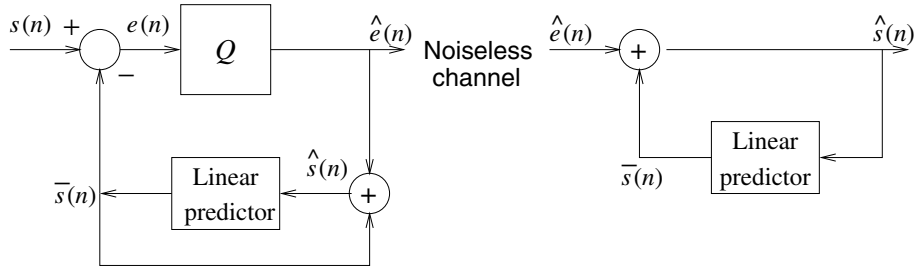


Figure 4.1 DPCM coder and decoder

where $\bar{s}(n)$ is a linear prediction of the reconstructed signal $\hat{s}(n)$

$$\bar{s}(n) = \sum_{k=1}^{p} a_k \hat{s}(n-k) \tag{4.2}$$

As deduced from the Figure, the reconstructed signal is

$$\hat{s}(n) = \bar{s}(n) + \hat{e}(n) \tag{4.3}$$

The operation of Equation (4.3) is also carried out at the decoder, as is shown in Figure 4.1. If the input signal is sufficiently correlated, the prediction error will have a random aspect (little correlation) and a smaller dynamic range than the input signal, so fewer bits will be required for its quantization. This is an example of how we try to eliminate redundancies from the speech signal for more efficient transmission. In order to obtain acceptable results, a prediction order not higher than 3 is commonly used. The prediction coefficients are fixed and can be obtained by different methods as was already mentioned in Chapter 2.

A special case of DPCM is *delta modulation* (DM). It is obtained for a 1-bit quantizer ($\hat{e}(n) = \pm\delta$) and a prediction order $p = 1$ with a prediction coefficient $a_1 = 1$. For example, a Bluetooth wireless LAN uses PCM and a variant of DM called *continuously variable slope delta* (CVSD) modulation for speech transmission (Stallings, 2002) (both at 64 kbps). In the case of CVSD, the prediction order is $p = 1$ again, the prediction coefficient is now $a_1 = 0.968$ and the quantization step size is dynamically computed.

The DPCM codec can be made more flexible by introducing an adaptive quantizer and/or an adaptive linear predictor. This yields the *adaptive differential pulse code modulation* (ADPCM). There are two ways of introducing adaptation:

- Forward adaptation: The adaptation is carried out at transmitter end. This scheme requires the transmission of the adaptation parameters (signal level or linear prediction coefficients) as side information. This involves the introduction of a delay.
- Backward adaptation: The adaptation parameters are obtained from the decoded signal itself, so that it is not necessary to send side information.

Figure 4.2 shows the ADPCM coders corresponding to forward and backward adaptation of the linear prediction coefficients.

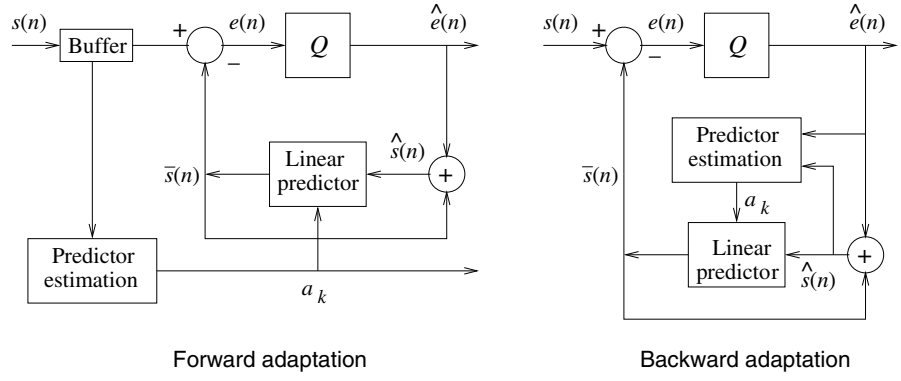


Figure 4.2 Forward and backward ADPCM coders

The standards ITU-T G.726 and G.727 (ITU-T, 1990a,b) convert 64 kbps μ/A-law PCM speech to ADPCM with backward predictor adaptation with bitrates of 40 kbps (data modem signals), 32 kbps (primary voice mode, known as G.721), and 24 and 16 kbps (low bitrate modes). The 40 and 24 kbps codecs form the standard G.723. The adaptive predictor has 2 poles and 6 zeros. The quantizers are individually optimized in G.726, and are embedded in G.727 (switch of bitrate; suitable for packet networks applications). They offer high-quality speech (MOS over 4).

4.2.1.2 Waveform Coders in the Frequency Domain

Frequency domain coders are based on the decomposition of the signal into frequency components that can be independently quantized and coded. The main types are *subband coders* and *transform coders*.

Subband coders (Figure 4.3) use analysis filterbanks to decompose the input signal $x(n)$ into frequency bands. The decoder uses a synthesis filterbank to provide a reconstructed signal $y(n)$.

The increment of the number of signals is compensated by decimation. For example, a filterbank providing M bands with the same bandwidth can use a decimation factor up to M as shown in Figure 4.3. A subband coder provides perfect reconstruction when $Y(z) = cz^{-k}X(z)$. Perfect reconstruction could be achieved by using filters with sharp transitions in the band limits. However, this could yield gaps or overlappings in those limits. Also, sharp filters usually involve a higher computational cost. In the case of $M = 2$, a smart solution is the use of *quadrature mirror filters* (QMF). Two filters $h_0(n)$ and $h_1(n)$ form a QMF pair if

$$h_1(n) = (-1)^n h_0(n) \tag{4.4}$$

and, therefore

$$H_1(\omega) = H_0(\omega + \pi) \tag{4.5}$$

A QMF filter pair reaches perfect reconstruction if

1. the filter order is even;

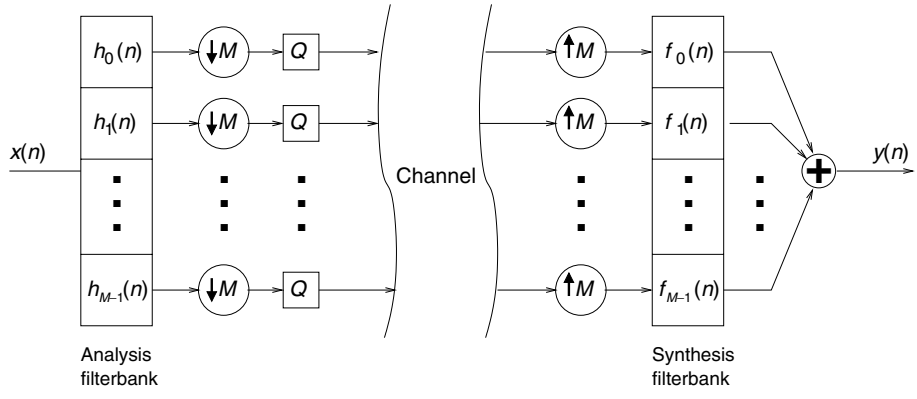


Figure 4.3 General scheme of a subband coder

2. they accomplish the following condition: $|H_0(\omega)|^2 + |H_1(\omega)|^2 = 1$ (this condition is not possible with linear phase FIR filters, although a good approximation can be reached);
3. The synthesis filterbank must satisfy:

$$F_0(z) = H_0(z) \tag{4.6}$$

$$F_1(z) = -H_0(-z) \tag{4.7}$$

Figure 4.4 shows a QMF filter pair. They are linear phase FIR filters, used in the ITU-T G.722 subband coder (ITU-T, 1988b). The bandwidth of G.722 is 50–7000 Hz, and the

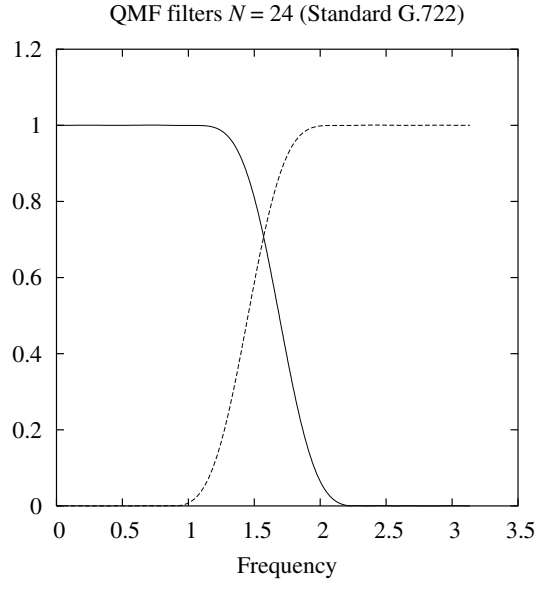


Figure 4.4 ITU-T G.722 QMF filters

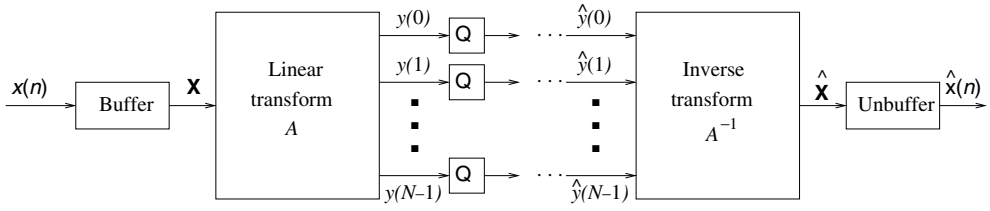


Figure 4.5 General diagram of a transform coder/decoder

bitrate is 64 kbps (56 and 48 kbps are also available). It uses two ADPCM quantizers (low passband 48 kbps; high passband 16 kbps). G.722 is appropriate for videoconference and ISDN applications.

Another important subband coder is the (masking pattern adapted universal subband integrated coding and multiplexing) MUSICAM algorithm used for MPEG-Audio coding (Dehery *et al*., 1991). It uses a DFT-based filterbank with 32 equal-bandwidth filters and a polyphase structure. The bit allocation is performed dynamically, according to a perceptual criterion, in order to minimize the quantization noise.

An alternative to subband coding is *transform coding* (TC). The frequency components are now obtained by applying an orthogonal transform to a signal input vector ($\mathbf{y} = A\mathbf{x}$), obtained by buffering the last N samples. A general diagram of such coding scheme is shown in Figure 4.5. The transform is commonly chosen so that the signal information (energy) is mainly concentrated in as few frequency components as possible. Thus, the quantization effort can be concentrated in those few components. An alternative approach to select a "good" transform is to make the new vector space as decorrelated as possible. This task is optimally performed by the Karhunen–Loewe transform (KLT), which diagonalizes the autocorrelation matrix of the input signal. However, it is common to apply a suboptimal transform instead of the KLT, since this one is signal-dependent so that it does not have any fast transform algorithm, and it should be transmitted along with the transformed signal every time the signal statistics change.

There are several important issues that must be considered when implementing a TC encoder:

- Choice of the transform: Well-known suboptimal transforms are the DFT, the discrete cosine transform DCT or the discrete Walsh–Hadamard transform (DWHT) (Jayant, 1984). The selection of a particular suboptimal transform depends on the statistics of the signal being encoded. A very commonly used transform is the DCT, which is defined as

$$y(k) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)k\pi}{2N} \tag{4.8}$$

$$x(n) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \alpha(k) y(k) \cos \frac{(2n+1)k\pi}{2N} \tag{4.9}$$

$$\alpha(0) = \frac{1}{\sqrt{2}} \qquad \alpha(k) = 1 \quad (k > 0) \tag{4.10}$$

The DCT transform is optimal for processes with autocorrelation matrices that are approximately Toeplitz (Sanchez *et al.*, 1995). Audio and video signals are usually approximated as AR processes, which have Toeplitz autocorrelation matrices. The DCT is widely employed in image and video coding, although it is also used in MPEG-Audio layer-3.

- Choice of the block size N: In the case of stationary processes, the distortion introduced by the TC coder is smaller for higher block sizes. However, for nonstationary signals the optimal block size must be determined in each case.
- Bit allocation: Given a certain transform and assuming that a total of NR bits are available (R is the average number of bits per transformed coefficient), it can be proved that the optimal bit allocation among the different component quantizers is (Jayant, 1984)

$$R_k = R + \frac{1}{2}\log_2 \frac{\sigma_k 2}{\left[\prod_{j=0}^{N-1} \sigma_j^2\right]^{1/2}} \tag{4.11}$$

  where R_k and σ_K^2 ($k = 0, \ldots, N-1$) are the number of bits assigned to the k-transformed component and the variance of this component, respectively.
- Zonal sampling: Since a good transform concentrates the signal energy in a few components, those components that are less relevant (with the lowest energy) can be removed (considered to be zero, so that it is not necessary to transmit them) without introducing any significant degradation. The problem is at which component the transform vector should be truncated. For example, for low-pass signals, this information corresponds to the lowest frequency components provided by a suitable transform such as the DCT.
- Choice of the quantizers.

These issues will be important for the implementation of TC-based compression for DSR systems (section 4.5.4).

4.2.2 Parametric Coders

These type of coders are usually known as *vocoders* and are based on a speech production model. They work on a frame-by-frame basis, so that it is required only to transmit the model parameters corresponding to the current signal frame. At the receiver, the model is used to synthesize the speech signal from the received parameters. Unlike waveform coders, parametric coders do not try to reproduce the input signal waveform, but to obtain a codec output signal that is perceptually similar to the original one.

The most commonly used model is the LPC model already seen in Chapter 2 (Figure 2.3). This model assumed that the speech signal is the output of a digital filter (see Figure 4.6), which represents the vocal tract. For convenience, we will consider now that the vocal tract filter has the following transfer function:

$$H(z) = \frac{\sigma}{1 - A(z)} \quad \text{with} \quad A(z) = \sum_{k=1}^{p} a_k z^{-k} \tag{4.12}$$

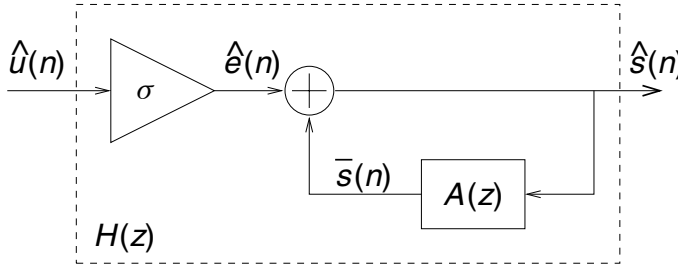


Figure 4.6 LPC filter representing the vocal tract

where we have changed the sign of the LPC coefficients with respect to Equation (2.6). As mentioned in Chapter 2, $H(e^{j\omega})$ is an estimate of the short-term spectrum of the signal. In the LPC model, the excitation is modeled as a signal $\hat{u}(n)$ that consists of a unit impulse train in the case of voiced sounds, and a white noise for unvoiced sounds. $A(z)$ can be considered to be the transfer function of a linear predictor (see Equation (4.2)), usually referred to as short-term predictor (STP). Therefore, $\hat{e}(n) = \hat{s}(n) - \bar{s}(n)$ can be considered an approximation to the residual $e(n)$ of Equation (4.1). The synthesized signal is $\hat{s}(n)$. Obviously, if we use the residual $e(n)$ as excitation (instead of $\hat{e}(n)$), then $\hat{s}(n)$ would coincide with the original signal $s(n)$. This parametric coding scheme requires the transmission of the filter parameters (LPC coefficients and gain), a voicing/unvoicing decision and an estimation of the pitch in the case of voiced sounds.

The U.S. Federal Standard 1015 LPC10e (Tremain, 1982) is a parametric LPC coder with 8 kHz sampling rate and 22.5 ms frame length. It uses an LPC order $p = 10$ and 54 bits/frame, providing a final bitrate of 2.4 kbps. Before transmission, the LPC coefficients are transformed into log-area ratios (a_1 and a_2) and reflection coefficients (a_3 to a_{10}). In the case of a voiced sound, one bit is used for voicing/unvoicing decision, six bits for pitch, five bits for gain, five bits for each of the first four vocal tract coefficients, four bits for coefficients 5 through 8, three bits for the ninth and two for the tenth and one bit for synchronization. In the case of a unvoiced sound, only the first four coefficients are transmitted, and the free bits are use for error channel protection. Thus, the LPC10e can be considered to be a variable bitrate coder. Although its intelligibility is acceptable, it provides an MOS score as low as 2.2.

The LPC10e coder has been replaced by the Federal Standard (Mixed Excitation Linear Prediction MELP) vocoder at 2.4 kbps, which reaches a MOS of 3.3 (McCree and Barnwell, 1995). The key to this improvement is the use of a better excitation model (which includes mixed pulse and noise excitation and periodic and aperiodic pulses) along with an adaptive spectral enhancement. In Section 4.2.4, we will see that the use of an improved excitation is crucial in order to develop high-quality coders.

4.2.3 Pitch Estimation

The previous LPC vocoders and other speech coders as those studied in the next section require an estimation of the pitch frequency. For example, the LPC10 standard uses an

average magnitude difference function, defined as

$$AMDF(m) = \sum_{n=0}^{N-1} |s(n) - s(n+m)| \tag{4.13}$$

In the case of a voiced signal, the AMDF function presents a valley for values of m close to the pitch period. The pitch is finally obtained by limiting the possible pitch estimates to the range of 50–400 Hz and by applying a dynamic programing smoothing.

There exist numerous pitch estimation algorithms, although all of them try to seek for peak values in a "correlation" function, such as the autocorrelation, the cross-correlation or the cepstrum. Detailed information about these algorithms can be found in Deller *et al*. (1993). An interesting method used in the extended DSR standards is the spectral comb method (Chazan *et al*., 2001; Martin, 1982). It is based on the maximization of the correlation between the signal spectrum $S(f)$ and a *comb* function $C(f; f_0)$,

$$U(f_0) = \int_0^{\infty} C(f; f_0) S(f)\, df \tag{4.14}$$

where $U(f_0)$ is called *utility* function. The comb function is non-negative and has a periodic pulse shape with peaks at $f_0, 2f_0, 3f_0, \ldots$ (identified with the harmonics of the pitch candidate f_0). The estimated pitch F_0 is obtained as,

$$F_0 = \underset{f_0}{\operatorname{argmax}}\, U(f_0) \tag{4.15}$$

The utility function can also be used to determine whether a speech signal is voiced or not by imposing a minimum threshold to $U(F_0)$.

4.2.4 Hybrid Coders

Hybrid coders can be considered as a mixture of waveform and parametric coders. They are parametric in the sense that they use a parametric model, but also try to preserve the signal waveform. If an LPC model is considered, this objective can be achieved by improving the excitation $\hat{u}(n)$ (or, equivalently, $\hat{e}(n) = \sigma\hat{u}(n)$) employed by the basic LPC model. The key point now is how to determine the optimal parameters of the new excitation $\hat{e}(n)$ once a reasonable model has been chosen for it. This is usually done employing by the *analysis-by-synthesis* (AbS) procedure depicted in Figure 4.7. It consists of obtaining the parameters that achieve the best fit between the original ($s(n)$) and synthesized ($\hat{s}(n)$) signals. This can be done by applying a *minimum mean square error* (MMSE) criterion. The mean square error (MSE) to be minimized is computed as

$$E = \sum_{n=0}^{N-1} \big(s(n) - \hat{s}(n)\big)^2 \tag{4.16}$$

The most characteristic feature of the AbS scheme is that the coder contains the decoder, which provides the required knowledge of the decoded signal $\hat{s}(n)$. The loop structure of the AbS scheme indicates that the optimization is performed iteratively. In the same way as for the parametric coder, the information finally transmitted includes the vocal tract and excitation parameters.

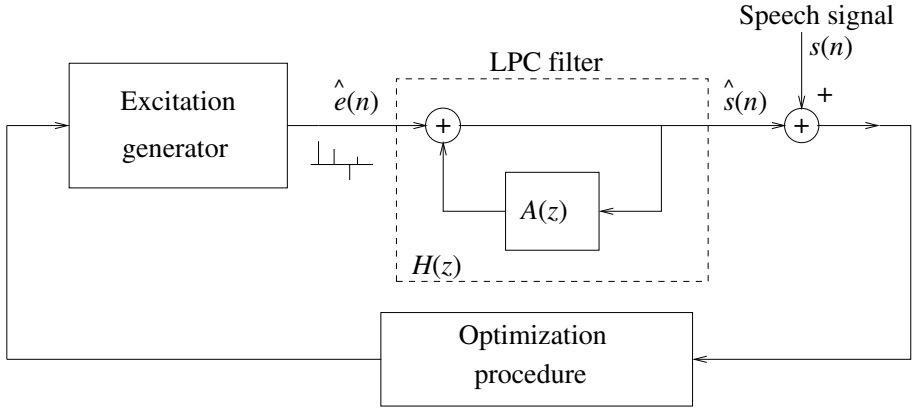


Figure 4.7 Generalized analysis-by-synthesis coder

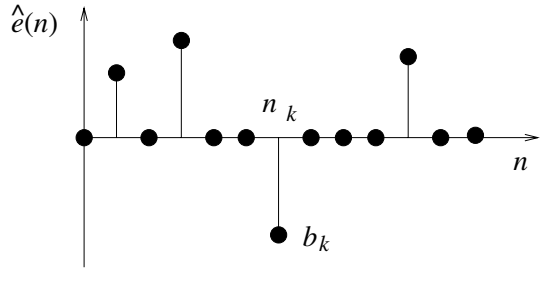


Figure 4.8 Excitation of a multipulse coder

4.2.4.1 Multipulse Coders

In multipulse coding (Atal and Remde, 1982), an excitation $\hat{e}(n)$ is built that consists of a series of l pulses with amplitudes b_k and positions n_k $(k = 1, \ldots, l)$

$$\hat{e}(n) = \sum_{k=0}^{l-1} b_k \delta(n - n_k) \tag{4.17}$$

where $\delta(n)$ is the unit impulse function. This excitation is depicted in Figure 4.8.

As previously mentioned, once the excitation model has been chosen, the excitation parameters (amplitudes and positions) have to be computed. They are obtained by minimizing the expression

$$E = \sum_{n=0}^{N-1} \big(s(n) - h(n) * \hat{e}(n)\big)^2 \tag{4.18}$$

with respect to b_k and n_k $(k = 0, \ldots, l-1)$.

The above multipulse coder can be improved by introducing two modifications in the basic AbS scheme. This is shown in Figure 4.9. First, a *pitch filter* $H_l(z)$ has been

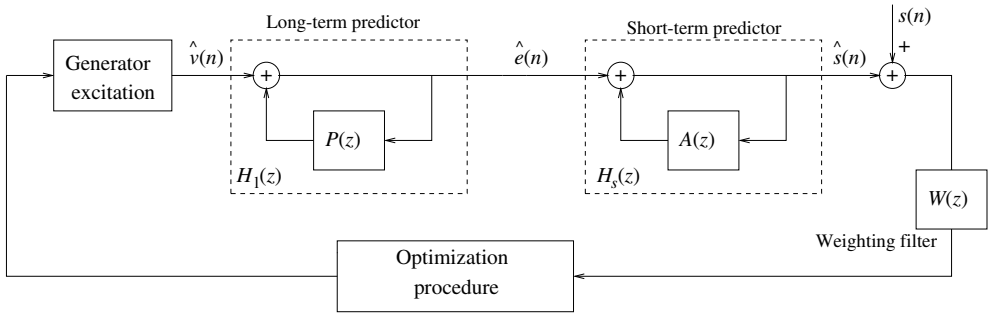


Figure 4.9 Improved AbS scheme

introduced in addition to the LPC filter $H_s(z)$. In the same way as the LPC filter accounts for the short-term spectrum, the pitch filter contains a *long-term predictor* (LTP) that accounts for pitch correlations. This predictor tries to eliminate the remaining redundancies contained in the residual $\hat{e}(n)$, so that the new residual $\hat{v}(n)$ can be correctly represented with less impulses. A possible expression for the LTP is,

$$P(z) = b_1 z^{-(M-1)} + b_2 z^{-M} + b_3 z^{-(M+1)} \tag{4.19}$$

where M coincides with the pitch period in the case of voiced sounds, and is a random value in the case of unvoiced sounds. The predictor parameters can also be obtained during the AbS optimization.

The second improvement to the AbS scheme is the introduction of a frequency weighting filter. This filter modifies the error signal $(\hat{s}(n) - s(n))$, with the aim that this error, which in fact is a coding noise, is perceptually masked by the speech signal itself. In order to do this, the filter emphasizes the error energy in the frequency valleys, where the SNR is lower, and does the opposite in the formant regions, where the SNR is higher. Thus, the optimization procedure applies more MSE minimization effort in the valley regions, providing thus an improved subjective quality (although the SNR gets worse). A possible transfer function for this filter is (Atal and Schroeder, 1979)

$$W(z) = \frac{1 - A(z)}{1 - A(z/\gamma)} \tag{4.20}$$

where γ is a heuristic factor usually chosen around 0.8.

A particular case of multipulse coding is the *regular pulse excitation* (RPE) coder, which has been extensively used for speech transmission on the TCH full rate (FR) channel of GSM. This coder is commonly known as GSM-FR (specification ETSI GSM 06.10) (ETSI, 1995) and uses an RPE-LTP (RPE with long-term prediction) technique (Kroon *et al.*, 1896). Its main feature is that the pulses are uniformly or regularly spaced. Therefore, it is not necessary to transmit the position of each pulse, but only the position of the first pulse. The GSM-FR coder uses a 8 kHz sampling rate and a frame length of 20 ms. It performs an LPC analysis of order $p = 8$. The LPC coefficients are transformed into LARs (see appendix A), which are interpolated between consecutive frames in order to avoid abrupt transitions. It divides each frame into 4 subframes of 5 ms. A single-coefficient LTP predictor and 13 regularly spaced pulses (grid) are estimated for each

subframe. The grid position (shift of the first pulse) is also required. The final bitrate is 13 kbps, 22.8 kbps after channel coding, and the MOS score is 3.7.

MPEG-4 has also incorporated two speech coders for narrowband (3.85–12.2 kbps) and wideband (10.9–23.8 kbps) speech, which use RPE and multipulse excitations, respectively (ISO, 1999).

4.2.4.2 CELP Coders

We can see now that hybrid coding has a double objective. First, it tries to substitute the original speech signal $s(n)$ by an excitation signal $\hat{v}(n)$ with as few redundancies as possible. This is the reason why we have introduced two prediction stages (including short-term and long-term predictors). If this excitation would equal the residual $v(n)$, then the synthesized signal would match the original speech signal. Thus, our second goal is to search for an excitation that represents the residual as accurate as possible. In this sense, the innovation of CELP (*code excited linear prediction*) coders is to provide a codebook containing a large and fixed set of excitations $c_k(n)$ $(k = 0, \ldots, M-1)$ (Schroeder and Atal, 1985). The basic formulation of the CELP coder assumes that the M excitations are random Gaussian (unit variance) sequences. The CELP idea is depicted in Figure 4.10. The excitation is built as $\hat{v}(n) = \gamma_k c_k(n)$, where γ_k is obtained during the optimization procedure along with $c_k(n)$. The CELP scheme requires only the index corresponding to the best codebook entry and the gain to encode the long-term residual $\hat{v}(n)$.

An example of implementation of a CELP coder is the Federal Standard 1016 (CELP) at 4.8 kbps (Campbell *et al*., 1989). It uses a frame length of 30 ms and subframes of 7.5 ms. The codebook contains 512 ternary (values $-1, 0, 1$) codewords 60 samples long. It obtains a 3.1 MOS score.

Another example is the ITU-T G.728 *low delay celp* at 16 kbps (Chen, 1990). While the previous coders can reach a high level of compression by introducing a coding delay of 50–100 ms, this coder reaches 16 kbps with a bitrate of only 2 ms. This can be accomplished by using an STP predictor with backward adaptation and avoiding the use of a LTP predictor, which is palliated by using an STP order of 50. The frame is 2.5 ms long (20 samples) and includes 4 subframes of 5 samples. The codebook uses 7-bit codewords shaped by a 3-bit gain. The reported MOS score is slightly less than 4.

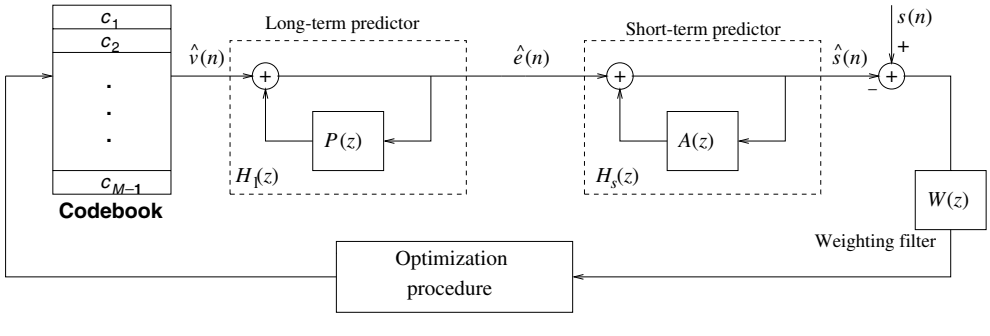


Figure 4.10 General scheme of a CELP coder

The CELP technique has also been derived in a number of variants such as the VSELP (vector sum excited linear prediction) adopted by the IS-54/136 (7.95 kbps) and GSM (for the half rate (HR)) channel, 5.6 kbps) mobile systems, (qualcomm code excited linear prediction) QCELP adopted by IS-95 (8.5-4-2-0.8 kbps, variable depending on voice activity) and the (*algebraic CELP ACELP*). This last variant has been extensively applied during the last years in mobile and IP networks. The following subsection is devoted to it.

4.2.4.3 ACELP Coders

The *algebraic CELP* (ACELP) is a modification of the CELP coder in which the codewords are sparse algebraic codes (Laflamme *et al*., 1990). These codes contain mainly zeros, which allows a fast codebook search. Sparse codes are obtained from permutation codes, which are obtained by permuting a series of pulses in a series of prefixed positions. Figure 4.11 shows an example of 4 monopulse codes for which a single pulse with amplitude ± 1 can be placed at 4 different positions. Each code requires 3 bits (1 bit for ± 1 decision and 2 bits for the position), obtaining thus a total of 12 bits and 2^{12} excitations of length 16 samples. The excitation is obtained by combining the 4 codes. The ACELP excitation resembles more that of the multipulse coder than that of the CELP coder previously seen. However, the codebook search, typical of CELP coders, is maintained.

An important example of an ACELP coder is the GSM *EFR coder* (ETSI, 1999a). A basic diagram of the decoder is shown in Figure 4.12. The bitrate is 12.2 kbps and obtains an MOS score of about 4. Again, the frames are 20 ms long (160 samples), with subframes of 5 ms. A 10th order LPC analysis is performed twice per frame using 2 asymmetric analysis windows of 30 ms (they are overlapped with the previous frame). The two LPC coefficient sets are transformed into line spectrum pairs (LSP), so that subframes 2 and 4 use their corresponding LSPs and subframes 1 and 3 use the interpolated versions. Both LSP sets are quantized by *split matrix quantizers* (SMQ) with a total of 38 bits. An important difference with the GSM-FR coder is how the pitch correlations are taken into account. Instead of using an LTP predictor, there is a *closed-loop pitch analysis* module that accounts for those correlations. The new pitch analysis can be considered as if an adaptive codebook was used in addition to the fixed algebraic one. Thus, as indicated in Figure 4.12, the excitation signal for the current subframe is obtained as

$$u(n) = g_p v(n) + g_c c(n) \tag{4.21}$$

where g_p is the adaptive codebook gain, $v(n)$ is the adaptive codebook excitation, g_c is the algebraic codebook gain, and $c(n)$ is the algebraic codebook excitation. The adaptive

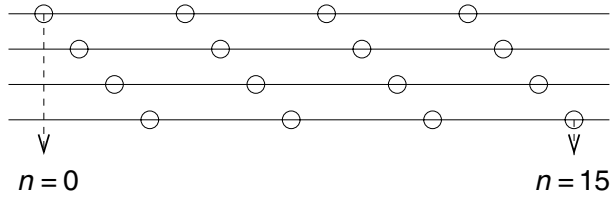


Figure 4.11 An example of a 12-bit permutation code

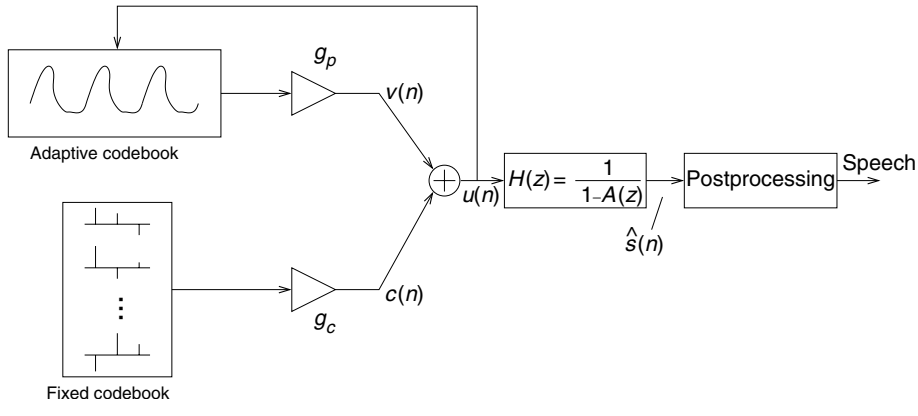


Figure 4.12 Diagram of an GSM-EFR decoder

Table 4.1 Allowed positions of the fixed codebook pulses in the EFR coder

Code	Pulses	Allowed Positions
1	p0,p1	0,5,10,15,20,25,30,35
2	p2,p3	1,6,11,16,21,26,31,36
3	p4,p5	2,7,12,17,22,27,32,37
4	p6,p7	3,8,13,18,23,28,33,38
5	p8,p9	4,9,14,19,24,29,34,39

codebook excitation is obtained as $v(n) = u(n - T)$ where T is the pitch period. EFR allows the use of a fractional pitch, so that $v(n)$ is built by applying an interpolation filter to the past samples of $u(n)$. The pitch is quantized with 9 bits for subframes 1 and 3, and differentially with 6 bits for subframes 2 and 4. The gains are quantized with 4 bits each.

The sparse algebraic codes in the EFR are built from 5 permutation codes, which allow the positioning of 10 pulses (2 pulses per code of values ± 1) in each subframe. The pulse positions are determined according to an AbS criterion to minimize the error perceptually weighted. Table 4.1 shows how the 10 pulses are distributed. Each code is quantized with 7 bits (total of 35 bits).

Another standard quite similar to EFR is the IS-641 coder employed by the IS-136 cellular system. Its bitrate is 7.4 kbps completed with 5.6 kbps of channel coding, yielding a total of 13 kbps (Honkanen *et al*., 1997). The IS-95 telephony system also admits the use of a RCELP (relaxed CELP) coder with an algebraic codebook search. This coder is known as *enhanced variable rate coder* (EVRC), and allows a variable bitrate operation (8.55, 4 and 0.8 kbps) (TR45, 1996).

The FR and EFR specifications include substitution and muting mechanisms for channel error mitigation. An alternative approach to avoid the annoying effect of channel errors consists in increasing the number of bits devoted to channel coding and reducing those devoted to source coding in order to maintain the total bit rate constant. The GSM *adaptive*

multirate (AMR) coder (ETSI, 1998b) implements this idea. In fact, AMR is a family of ACELP coders with the same structure as EFR, but working at different bitrates. At a given instant, the selection of a specific coder depends on the channel condition. The higher bitrate is chosen for a channel in a good condition, while the lowest one is selected when the channel is severely degraded. The remaining bits are devoted to channel coding up to the completion of the total bitrate (22.8 kbps for the TCH/F channel and 11.4 kbps for the TCH/H channel). The AMR bitrates are 12.2 (this is the EFR coder), 10.2, 7.95, 7.4, 6.7, 5.9, 5.15 and 4.75 for the TCH/F channel and 7.40, 6.7, 5.9, 5.15 and 4.75 kbps for the TCH/H channel. The AMR coders use a frame length of 20 ms, and an LSP coefficient set is computed once per frame (except at 12.2 kbps, EFR) by applying an asymmetric window that emphasizes the more recent samples. The AMR specifications have also been adopted by UMTS (specification 3GPP TS 26.090).

4.2.4.4 Hybrid Coders for Packet Network Applications

While the above coders were mainly developed for use in cellular systems, there are also other speech coding standards such as the ITU-T G.723.1 (part of the H.323 standard for videoconferencing) and G.729, which are commonly used for applications over packet networks. The ITU-T G.723.1 is a dual-rate coder for which the excitation can be represented either as CELP (bitrate of 5.3 kbps, MOS 3.6) or multipulse (bitrate of 6.3 kbps, MOS 3.9), although both coders share the same short-term analysis section. The frame length is 30 ms with 4 subframes and the algorithmic delay (look-ahead) is 7.5 ms. Its relatively low complexity combined with its low bitrates make G.723.1 suitable for IP networks. G.729 is a *conjugate structure ACELP* conjugate structure ACELP(CS-ACELP) coder (Salami *et al.*, 1998) that works at 8 kbps with frames 10 ms long and a look-ahead of 5 ms. Thus, G.729 adds a low delay feature to its low complexity, low bitrate and excellent performance (MOS close to 4). Low delay is desirable in order to avoid annoying echoes (a delay above 50 ms would require echo cancellation). G.729A is a variation with about half the complexity of G.729 with a slight performance degradation, and G.729B describes a *voice activity detector* (to be used with either G.729 or G.729A) that allows discontinuous speech transmission, that is, speech is not transmitted during silence periods (comfort noise is generated at the decoder). This is a way of reducing the bandwidth requirements. Both standards, G.723.1 and G.729, include algorithms for frame erasure concealment.

Although we see that CELP coders have also been applied to packet networks, it must be considered that they were originally developed for wireless and circuit-switched networks, where they should be resilient against bit errors. However, in packet networks the most important degradation comes from packet delays and losses. Thus, more recently the *internet low bit-rate codec* (iLBC) (Andersen *et al.*, 2004) has been proposed. Unlike CELP coders, iLBC does not have interframe dependencies, since prediction is carried out only within the current frame during encoding (frame-independent long term prediction). As a result, the effect of frame losses, due to channel impairments, can be reduced. The robustness of this coder against packet losses has the price of an increased bitrate. It has two possible bitrates: 15.2 kbps for a frame length of 20 ms and 13.33 for a frame of 30 ms (the sampling rate is 8 kHz). It reaches an MOS score close to 4 in clean channel conditions.

4.3 Recognition from Decoded Speech

We have seen in the previous section that there was intense activity in the speech coding field during the nineties. This activity arose from the interest in speech recognition from decoded speech and in how speech coders could affect speech recognition. In the first chapter, we already called this variant of RSR as NSR. In a basic NSR system, the received codec bitstream is decoded and the recognition is performed from the decoded speech. Thus, we will refer to this basic approach either as *speech-based* NSR (S-NSR) or, simply, NSR. Figure 4.13 shows a diagram of an S-NSR system. The received codec bitstream is decoded and the recognition is performed from the decoded speech. The availability of a speech signal of acceptable quality at the receiver is an important characteristic of NSR that DSR does not always allow and which can make NSR an attractive solution for RSR (Kim and Cox, 2000).

4.3.1 Effect of Coding

One would think that a good coder for hearing is a good coder for recognition. However, this does not have to be true since coders are designed with a criterion different from recognition accuracy. Thus, it makes sense to study how the different speech coders perform in a recognition task. An early work on the effect of speech coding over speech recognition can be found in Euler and Zinke (1994), where it is shown that there are two reasons why a speech codec can degrade the recognition performance. The first and more important is the degradation involved by the compression itself, which degrades the speech quality and, therefore, the recognition performance. The authors tested the performance of an HMM-based 23-word vocabulary-isolated word recognition system in which the HMM models were initially trained with 64 kbps A-law speech and a feature vector which consisted in (LPC) cepstrum, delta cepstrum and delta energy. The test set was previously coded and decoded with several codecs (64 kbps G.711 A-law, 16 kbps G.728 LD-CELP, 13 kbps GSM-FR and 4.8 kbps TETRA CELP codec) and the recognition experiments showed that the word accuracy diminished as the bitrate was diminished (98.52, 97.57, 96.96 and 96.04 %, respectively). The second reason for performance reduction was that if the recognition system accepted speech from different speech codecs, then a mismatch between training and testing conditions could appear. To solve this problem, Euler proposed a Gaussian classifier that allows the identification of the codec applied, and then the use of a suitable (trained for that codec) set of HMM models.

Lilly and Paliwal (1996) obtained a similar decrease in performance for a decreasing bitrate. They chose six coders that covered the range 40–4.8 kbps: 40 kbps G.726 ADPCM, 32 kbps G.726 ADPCM, 24 kbps G.726 ADPCM, 16 kbps G.728 LD-CELP, 13 kbps GSM-FR and 4.8 kbps CELP-1016. An important conclusion of this work is that

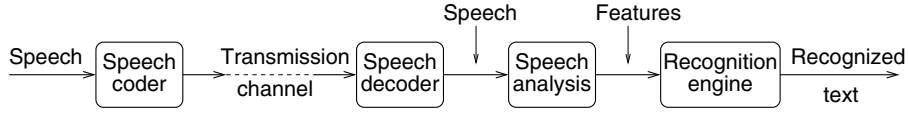


Figure 4.13 Scheme of an NSR system using decoded speech

the word accuracy does not necessarily decrease monotonically with the bitrate, since speech coders are commonly designed with a perceptual criterion, and their performance in speech recognition is not predictable. Thus, the performance of each coder varies depending on the recognition task. Another important conclusion was that MFCC cepstrum clearly outperformed LPC cepstrum. These results support the fact that MFCC is a more robust parametrization than linear prediction cepstrum coefficients LPCC, as was already mentioned in section 2.3.3.

A complete study of the effect of GSM/UMTS codecs (FR, HR, EFR and AMR) over speech recognition can be found in Hirsh (2002). The experiments were carried out on the Aurora-2 database (including clean and contaminated speech) by applying the ETSI FE and AFE FE (without compression). The results of these experiments, when training and testing were carried out using the same codec, are shown in Table 4.2. The training and testing databases contains clean and noisy speech. The word accuracy WAcc values in columns labeled as "almost-clean" correspond to the average of six different experiments performed over speech at SNRs ranging from 0 to 20 dB. Three of these experiments exclusively use clean data for training. The average of the other three experiments (noisy training data) are listed in columns labeled as "multicondition." The results of the FE-almost-clean experiment are quite random (WAcc does not monotonically decrease when the bitrate is decreased) as a consequence of the mismatch between training and testing. This does not happen in the FE-multicondition experiment. However, the results for the AFE FE look quite coherent for both almost-clean and multicondition experiments, which is a proof of the power of that FE under noisy conditions. In another set of experiments, the effect of mismatch due to the use of different codecs for training and testing is tested. The conclusion at this point is that the best performance is generally obtained using PCM speech for training (without additional coding). This result, also reported in other references Kim and Cox (2000, 2001a), is opposite to the one observed in Euler and Zinke (1994) or Pelaez *et al*. (2001), among others. In Kim and Cox (2001a), this variable behavior is explained as the consequence of the variable number of Gaussian mixtures employed by the HMM models.

Table 4.2 Recognition performance (WAcc) for PCM and different GSM codecs (Hirsch, 2002)

Codec and bitrate (kbps)	FE almost-clean WAcc (%)	FE multicondition WAcc (%)	AFE almost-clean WAcc (%)	AFE multicondition WAcc (%)
PCM 64	73.23	86.39	89.30	91.55
A-LAW 64	70.15	85.76	88.88	91.53
FR 13	68.31	85.28	87.31	90.28
HR 5.6	66.44	82.45	81.77	87.18
EFR 12.2	71.44	86.22	88.16	90.97
AMR 4.75	70.16	84.89	84.76	88.99
AMR 5.15	71.17	84.49	84.23	89.09
AMR 5.9	69.46	85.05	85.02	89.66
AMR 7.4	67.58	85.74	85.23	90.01
AMR 10.2	68.38	85.63	86.36	90.50

Another complete study on the recognition performance obtained using different GSM codecs, plus G.723.1 and G.729, under noisy conditions can be found in Villarrubia and Hernandez (2001). The best results are obtained for G.729, FR and AMR at high bitrates, although FR is less robust against acoustic noise. HR and G.723.1 are the worst options. The application of G.723.1 to S-NSR is also studied in Pelaez *et al*. (2001).

4.3.2 Effect of Tandeming

A possible scenario for RSR, especially in the case of long distance calls, is that in which the speech is submitted to several coding schemes as it is passed through different networks. This situation, known as *tandeming*, usually results in a degradation of the quality of the decoded speech or the performance of a recognizer placed at the tandem end.

The effect on tandeming over speech recognizers was also studied in Lilly and Paliwal (1996) by measuring the recognition performance on IWR and CSR systems when one of the utilized codecs was consecutively applied. The conclusion was that tandeming has a little effect when a high bitrate coder (40–16 kbps) was used, while the lowest bitrate coders (13 kbps or less) are more damaging. Thus, the GSM-FR coder decreased the word accuracy from 85.71 % (1 coding) to 70.26 % (5 codings), and the CELP-1016 from 81.86 % (1 coding) to 41.54 % (5 codings).

In Salonidis and Digalakis (1998) we can find a more realistic study, in which multiple topological scenarios for the GSM mobile network are considered. These scenarios are shown in Figure 4.14. The worst case corresponds to topology (1). The speech codecs utilized in the different topologies are G.711 (64 kbps), ADPCM G.721 (32 kbps) and G.723 (16 kbps), and GSM-FR (13 kbps). The ADPCM codecs were selected because they are used for digital circuit multiplication equipment (DCME) international connections. The recognition results in a large vocabulary task ranging from 13.30 % of WER for

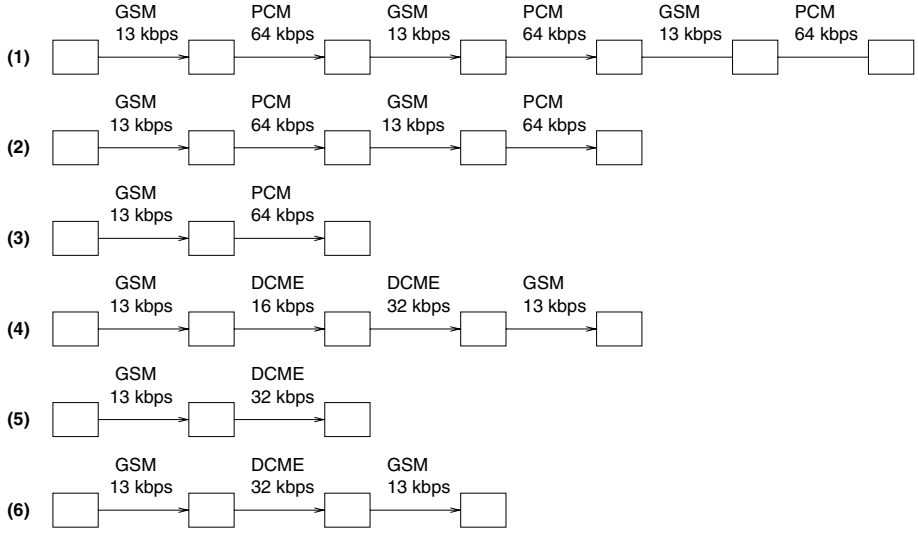


Figure 4.14 Several topologies of tandeming (after Salonidis, 1998)

clean speech to 23.75 % of WER for topology (1). The degradation introduced by the codec tandem can be partially compensated by applying model adaptation techniques (see Chapter 2). Thus, the WER of topology (1) can be reduced to 17.3 %. This adaptation requires the knowledge of the specific topology of the tandem. This *a priori* knowledge is difficult to obtain, although it can be obviated if a "cocktail" adaptation, which mixes the six considered topologies, is applied. The result obtained with this cocktail transformation for each topology is quite close to the case in which the adaptation is specifically trained for that topology.

The degradation involved by a tandem can also be avoided if the system operates in a *tandem free operation* (TFO) mode, available, for example, for GSM (ETSI, 2003d). TFO allows the transmission of the codec parameters corresponding to the first tandem coder up to the last decoder, which must be of the same type as the first one, without applying any new decoding and coding.

4.3.3 Treatment of the "Coding" Degradation

As previously mentioned, speech coding can introduce a mismatch (due to compression and decoding artifacts) between training and testing conditions. In order to reduce this mismatch, there are several possibilities:

- To "clean" the input speech by applying some enhancement algorithm to the decoded speech;
- To search for a feature extraction method more robust (or, equivalently, less sensitive) against the "coding" degradation. For example, it has been proven that the root-cepstrum introduced in Chapter 2 is more robust than the usual MFCC representation (Dufour *et al*., 1996);
- To apply feature processing or compensation techniques as the ones that will be developed in Chapter 6 (Mokbel *et al*., 1996);
- To transmit, along with the codec parameters, extra data containing compensation information;
- To adapt the recognition models to the input speech.

Let us analyze in detail the two last options. Regarding the transmission of extra compensation data, it has been proposed (Skogstad and Svendsen, 2005) that the transmission of the error vector $\boldsymbol{\epsilon}$ between the feature vector $\mathbf{x}$ extracted from the original signal and the one extracted from the decoded signal $\hat{\mathbf{x}}$ is

$$\boldsymbol{\epsilon} = \mathbf{x} - \hat{\mathbf{x}} \tag{4.22}$$

At the decoder, the speech signal is reconstructed, and the corresponding features $\hat{\mathbf{x}}$ are computed from it and compensated with $\boldsymbol{\epsilon}$. It has been shown that by compressing $\boldsymbol{\epsilon}$ with VQ (2 bits for energy information and 4 bits for the MFCCs), the same performance as with uncoded data can be obtained when using AMR-4.75 encoded speech with a total source bitrate (4.75 kbps plus the compensation information) of 5.35 kbps, which is close to the 5.4 kbps of the ETSI extended front-end XFE/XAFE standards. This approach has two problems. First is its high computational cost. Also, it may require the use of two different channels (for the codec parameters and the compensation data). However, both

problems could be overcome by adopting the trans-parametrization approach described in the next section (Skogstad and Svendsen, 2005).

The adaptation solution has been extensively proposed for this mismatch reduction. An example of the application of model adaptation techniques to mitigate the effect of tandeming was already seen in (Salonidis and Digalakis, 1998) (previous section). The model adaptation is performed through the MLLR techniques reviewed in Chapter 2, which transform the model means $\boldsymbol{\mu}$ and covariance matrices Σ, by means of a regression matrix A and a bias vector $\boldsymbol{b}$, as

$$\hat{\boldsymbol{\mu}} = A\boldsymbol{\mu} + \boldsymbol{b} \tag{4.23}$$

$$\hat{\Sigma} = A\Sigma A^t \tag{4.24}$$

where A can be either diagonal (then, means and variances are transformed) or block diagonal (then, only the means are transformed). The models are grouped into phonetic classes, and one transformation is computed for each class using some adaptation data (collected in the conditions we want to adapt to). The WER of 17.3 % mentioned in the previous subsection was obtained using 10 classes and diagonal regression matrices. As also mentioned, the problem of having *a priori* knowledge of the specific working condition (tandem configuration) is solved by obtaining a "cocktail" transformation from adaptation data (400 sentences) collected in the six different tandem topologies. Similar work can be found in Mokbel *et al*. (1997) and (Srinivasamurthy *et al*., 2001a), where Bayesian (MAP) and MLLR adaptations are applied to provide robust models for GSM-FR and MELP coders. The "cocktail" approach is also studied in Sciver *et al*. (2002), although in this case they obtain the best results when the mixed data from the different codecs (11.8 kbps G.729E, 8 kbps G.729, 6.4 kbps G.723.1, and 6.3 kbps G.729D) is used to retrain the models, instead of adapting them.

Huerta and Stern (2001) also deal with model adaptation to GSM-FR speech. This work starts by asserting that not all speech sounds are encoded with the same quality. For example, vowels are less sensitive to the encoding process than consonants or silences. This fact is reflected in the energy of the long-term residual (a large value indicates less predictability and, therefore, a worse encoding). In fact, the long-term residual is used to measure the similarity between two codec subframes by means of the relative log spectral distortion (RLSD), defined as

$$RLSD = \frac{1}{\pi}\int_0^{\pi} \left| \frac{\log S(\omega) - \log S_R(\omega)}{\log S(\omega)} \right| d\omega \tag{4.25}$$

where $S(\omega)$ and $S_R(\omega)$ are the power spectra of the subframe long-term residual and its corresponding quantized (after encoding) version, respectively. Two sets of HMM models are computed for coded (set 1) and clean (set 2) speech, respectively. The adaptation is carried out by merging the observation probabilities (defined in Equation (2.71) of the equivalent state pairs from both HMM sets

$$b_j(\mathbf{x}) = \lambda_j \sum_{k=1}^{M} c_1(j,k) N(\mathbf{x}; \boldsymbol{\mu}_{1,j,k}, \Sigma_{1,j,k}) + (1-\lambda_j) \sum_{k=1}^{M} c_2(j,k) N(\mathbf{x}; \boldsymbol{\mu}_{2,j,k}, \Sigma_{2,j,k}) \tag{4.26}$$

where λ_j and $(1 - \lambda_j)$ are the weights assigned to state s_j of both sets. In order to compute these weights, the phonetic recognition units are clustered into 15 phonetic categories by measuring MSE values between RLSD-normalized log-histograms of those phonetic units, and an averaged log-histogram is computed for each category. Finally, a single weight is computed for each category as the median of the corresponding log-histogram divided by the peak value of that histogram. Thus, the weight varies in the interval [0, 1]. The recognition experiments with the SPHINX-3 system and the TIMIT database provide WERs of 11.5 % for clean speech, 12.2 % for GSM speech (matched training/testing), 11.9 % for GSM speech and training with clean and GSM speech and 11.7 % with the proposed combination of acoustic models.

4.4 Recognition from Codec Parameters

The NSR architecture can be modified in order to obtain the recognition features directly from the codec parameters, which are encoded in the bitstream generated by the speech coder. This variant is depicted in Figure 4.15. As can be observed, it avoids the intermediate reconstruction of the speech signal by introducing a bitstream-based feature extraction that directly transforms the codec parameters into recognition features. We will refer to this architecture as *bitstream-based NSR* (B-NSR). The feature extraction can also be viewed as a *trans-parametrization*, or *transcoding* if we take into account that the speech signal could also be reconstructed from recognition features (as in the ETSI XFE or XAFE DSR standards). In fact, we can view the XFE/XAFE standards as a variant of B-NSR in which no trans-parametrization is required. Furthermore, some authors have proposed the development of speech coders using cepstrum as spectral representation in order to improve recognition while maintaining a good decoded speech quality (Zhong *et al.*, 2002a).

Figure 4.16 shows a typical B-NSR feature extractor which can obtain LPCC and MFCC cepstra from some of the LPC representations mentioned in appendix A. LPCCs are straightforwardly extracted by applying the recursion of Equation (2.11). In order to obtain the MFCCs, it is necessary to obtain a sampled version of the LPC spectrum. Then, the process is identical to the one described in Chapter 2 (Equations (2.9) and (2.12)).

There are several reasons why the B-NSR approach can be attractive:

- Speech coders are designed with the goal of providing a subjective quality as good as possible. For example, it is common to include some type of postprocessing at the decoder in order to obtain a decoded signal perceptually improved. However, this

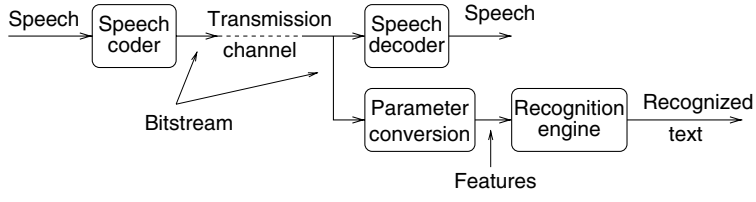


Figure 4.15 Scheme of speech recognition from codec parameters

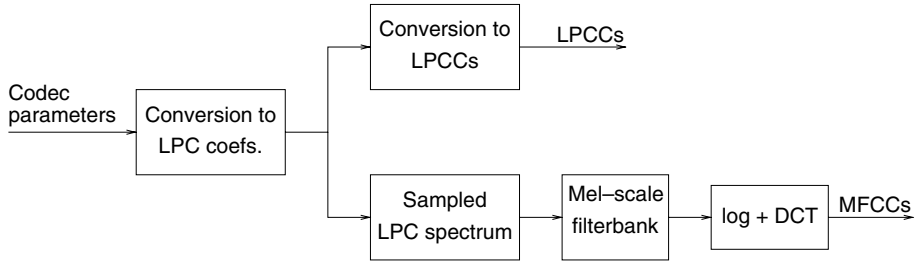


Figure 4.16 Block diagram for the extraction of LPCCs and MFCCs from the codec parameters

postprocessing is not optimized for an objective performance measure as in speech recognition (Turunen and Vlaj, 2001).

- By applying this approach, the short-term spectral parameters and the excitation parameters can be treated separately. This fact can provide more robustness against acoustic and transmission channel degradation, as we will see in the next subsection.
- It is not necessary to reconstruct the speech signal. This provides computational saving.
- In the case of a degraded transmission channel, a suitable mitigation algorithm for speech recognition can be applied directly on the codec parameters. For example, the mitigation algorithm can benefit from the fact that ASR allows more delay than in wireless or IP telephony.
- B-NSR can combine the advantages of S-NSR and DSR, that is, a decoded speech of good quality and a good recognition performance.

However, the B-NSR approach also has several drawbacks. First, we can mention that the FE is codec-dependent. Second, the frame rates of the codec and the recognition FE can be different, although this is a minor problem that can be easily overcome, as is shown in this section. Finally, although its implementation over IP networks can be straightforward, this is not true for the current mobile networks, since the coded speech is normally transcoded into PCM speech in the TRAU units as we saw in the previous chapter. However, TFO operation could also offer a solution for this problem.

4.4.1 Robustness of Bitstream-based NSR

The robustness of the bitstream-based features against acoustic noise is studied in Huerta and Stern (1998) for the GSM-FR coder. In this work, three different cepstra are derived: the cepstrum derived from decoded speech, the cepstrum derived from the quantized LAR (Q-LAR) coefficients, and the cepstrum derived from the residual signal. The MSE between the Q-LAR and original speech cepstra increases with respect to order of the cepstral coefficient. This is a behavior similar to that of the MSE between the decoded speech cepstrum and the original one (versus the coefficient order). However, MSE between the residual and original cepstra does the opposite, and has much greater values than the former one. The baseline system uses the cepstrum derived from the original speech and provides 89.7 % and 45.0 % of WAcc for clean and noisy speech, respectively. The recognition experiments (using the same type of cepstrum for training and testing) show that the cepstrum derived from GSM-decoded speech and the Q-LAR cepstrum provides

the best results (89.2 and 87.5 % WAcc, respectively, for clean speech, and 47.5 and 44.9 %, respectively, for noisy speech). However, the most surprising result is obtained for the GSM residual cepstrum (67.5 % for clean and 3.9 % for noisy), which indicates that the residual still contains information relevant for recognition. This result suggests that the GSM Q-LAR and residual cepstra could be combined to improve the performance. The obvious combination corresponds to the sum of both cepstra, and provides 89.1 % of WAcc for clean speech and 47.1 % for noisy speech. However, the MSE experiments suggest the use of an N-dimensional cepstral vector obtained by concatenating the first i Q-LAR cepstral coefficients with the last $N - i$ residual cepstral coefficients ($N = 13$, $i = 8$). This combination provides the best results, with 89.7 % of WAcc for clean speech and 49.4 % for noisy speech. These results are even better than those obtained with the original cepstrum (without coding).

A B-NSR approach and its performance against acoustic noise is also tested in (Yu and Wang, 1998), where LSPs, LPCCs and MFCCs extracted from the coders LPC10e, FS1016 and GSM-FR are tested in different acoustic conditions. The results show again the robustness of the MFCC parameters.

Gallardo *et al*. (1998) have studied the robustness of B-NSR against channel errors in a GSM environment. The GSM-FR LARs are directly converted into MFCC cepstrum in three stages: a frequency sampling of the LPC spectrum, application of a mel-distributed filterbank and DCT. The system, depicted in Figure 4.17, still requires the reconstruction of the speech signal in order to obtain the log-energy. The performance of this feature vector in an IWR task is only slightly worse (99.66 % of WAcc) than that obtained from the original speech (99.77 %) and that obtained from coded/decoded speech (99.89 %). The experiments were carried out for matched conditions (training and testing condition are the same) and a clean transmission channel. However, when a noisy channel (random errors or bursty errors simulated by a Gilbert–Elliot model) is considered, the system using trans-parametrization is much more robust (95.23 and 90.91 %, for random and bursty errors, respectively, BER = 1 %) than the system using decoded speech (90.34 and 80.80 %, for random and bursty errors, respectively, BER = 1 %). This result can be explained from the fact that a transmission error affecting only the encoding of the residual affects the whole decoded signal, but not the LARs. Besides, LARs are more protected against channel errors than the residual in the GSM-FR codec. Similar work ((Gallardo *et al*., 1999) and (Pelaez *et al*., 2001)) also support the idea that the use of bitstream-based feature extraction can obtain better and more reliable results than feature extraction from decoded speech for both clean and degraded transmission channels.

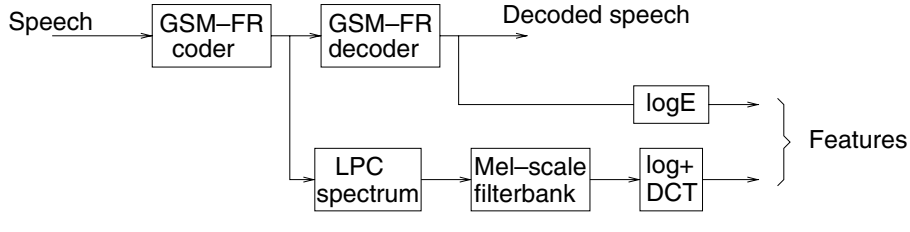


Figure 4.17 Diagram of a bitstream-based MFCC feature extractor with logE extracted from the decoded signal (after Gallardo *et al*., 1998)

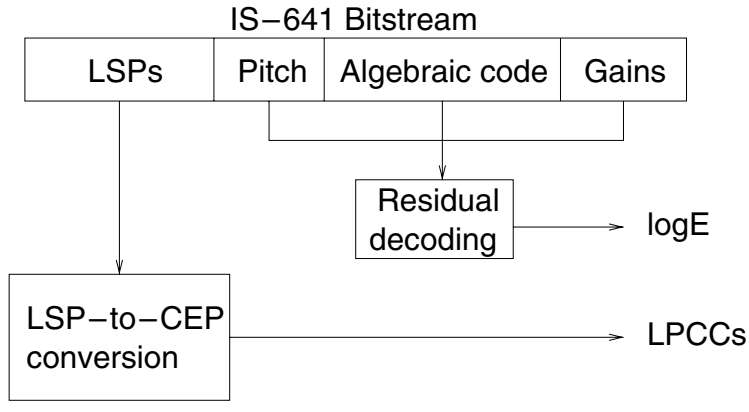


Figure 4.18 Diagram of a bitstream-based LPCC feature extractor with logE extracted from the decoded residual (after Kim *et al*., 2001)

The robustness of the B-NSR approach against both acoustic impairments and transmission errors is widely analyzed in Kim and Cox (2001a). This work develops a bitstream feature extractor for the IS-641 speech codec (see Figure 4.18). The feature vectors contain 12 LPCC cepstral coefficients (obtained from the LSPs) and the log-energy, and are obtained from speech segments of 30 ms every 20 ms (frame rate of 50 Hz, as the IS-641 codec). As baseline (without encoding), it is considered a similar feature analysis but with frames shifted by 10 ms (frame rate 100 Hz). The frame rate of the bitstream feature extractor is then increased to 100 Hz by linear interpolation of the LSPs. This work contributes two improvements with respect (Gallardo *et al*., 1998). First, it obtains the log-energy from the codec parameters instead of from the decoded speech. Specifically, it reconstructs the residual from the adaptive and fixed codebook parameters as illustrated in Figure 4.18, and then the log-energy is obtained as the logarithm of the square-sum of the residual samples. This feature extractor performs slightly worse (95.81 % of WAcc) than the baseline feature extractor (96.17 %) in matched training/testing conditions for a connected-digits task. The performance of a similar feature extraction from decoded speech is worse than both (94.75 %). The second improvement consists of substituting the two last cepstral coefficients (LPCC(11−12)) by two new features obtained from the fixed (algebraic) and adaptive codebook gains (ACG) (g_p and g_c of Equation (4.21)). Since there are four gain pairs per codec frame (20 ms) corresponding to four subframes (5 ms), and the recognition features are computed every 10 ms, the new features (ACG and fixed codebook gains (FCG)) are obtained by combining the gains of two consecutive subframes (indicated with indices 0 and 1)

$$ACG = g_p^2(0) + g_p^2(1) \tag{4.27}$$

$$FCG = \gamma \cdot 10\log_{10}\left(g_c^2(0) + g_c^2(1)\right) \tag{4.28}$$

This is a way to introduce voiced/unvoiced information into the feature vector. Median filtering is applied to these new features in order to avoid temporal fluctuations. Besides, the fixed codebook gain feature is weighted (factor γ) to equalize its effect with respect the

other features. With this second improvement the bitstream feature extractor can reach a WAcc of 96.24 %. This enhanced FE also provides quite excellent results when car noise is added to the testing database. However, this is not true when a babble noise is added. The reason for this is that this noise specially affects the voicing information, so that the new features are not useful yet. This situation is partially compensated by applying a speech enhancement algorithm based on MMSE log-spectral amplitude estimation, although there is a slight performance reduction in clean conditions. Finally, this work studies the effect of a degraded channel that causes frame erasures. The IS-641 codec mitigates frame erasures by extrapolation, and the decoder uses the extrapolated parameters to reconstruct the speech signal. In the case of the bitstream FE, the erased frames are mitigated by both extrapolation or deletion (see mitigation techniques in the following chapter). The bitstream FE obtains much better results than the decoded speech-based FE for both random and bursty erasures and for both mitigation methods. The deletion method provides additional computational saving, although it tends to introduce a high deletion error.

While all the robust B-NSR techniques described above use conventional speech codecs, it has also been argued that *multiple description coding* (MDC) (Zhong and Juang, 2002) can be a suitable alternative for robust ASR over erasure channels (Zhong *et al*., 2002b). The MDC concept is depicted in Figure 4.19. In this example, two different encoders transmit two different speech descriptions over two different channels. If both channels transmit correctly, decoder 0 can reconstruct a super-resolution signal S_0. Otherwise, we still have the coarser descriptions S_1 and S_2 of the signal. In Zhong *et al*. (2002b), MDC is successfully applied to RSR over degraded channels using repetitive coding, time diversity coding and residual compensation coding.

4.4.1.1 Feature Enhancement in B-NSR Systems

In mobile networks, the codec parameter domain is suitable for the application of some type of enhancement in order to compensate the effect of the acoustic noise, since it is implemented at the receiver side, discharging so the mobile phone (encoder side) of implementing additional processing.

In Kim and Cox (2001b); Kim *et al*. (2002), we can find a modification of bitstream-based FE of (Kim and Cox, 2001a) (Figure 4.18) that introduces a speech enhancement algorithm over the decoded speech in order to obtain an improved set of codec parameters.

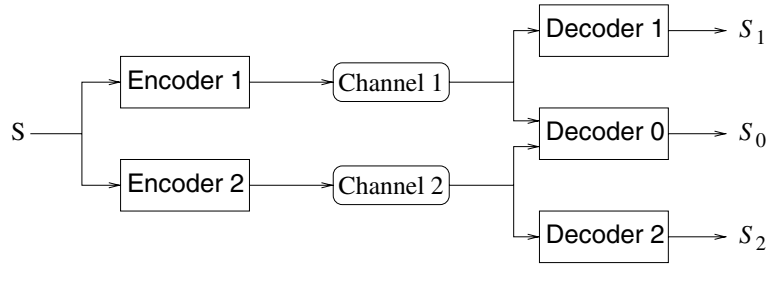


Figure 4.19 Diagram of a multiple description coder (after Zhong *et al*., 2002)

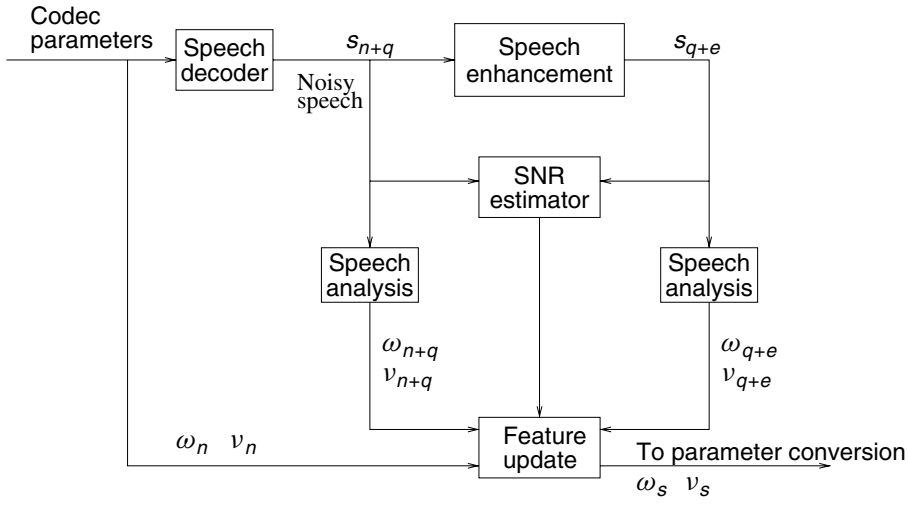


Figure 4.20 Enhancement of the codec parameters for a B-NSR system (after Kim *et al.*, 2002)

Figure 4.20 shows a diagram of the parameter enhancement algorithm. The spectral codec parameters (LSPs) are noted as ω_n, and the coder-specific parameters (adaptive codebook lags and gains, FCG and algebraic codes) as ν_n, where subscript n indicates that the parameters are corrupted by acoustic noise. The original parameter set (of the clean speech) is noted as (ω_s, ν_s). The proposed enhancement algorithm is the MMSE-LSA (minimum mean square error log-spectral amplitude) estimation (Ephraim and Malah, 1985), applied to some speech coders. As seen in the Figure, the speech is decoded and analyzed before and after enhancement, obtaining new sets of codec parameters $(\omega_{n+q}, \nu_{n+q})$ and $(\omega_{q+e}, \nu_{q+e})$, respectively, where subscripts q and e denote decoding and enhancement, respectively. An LMS (least mean square) estimate of the clean speech spectral parameters $\hat{\omega}_s$ can be obtained by means of a gradient descent, which tends to reduce the distortion with respect to the original clean parameters. The LMS updating formula is

$$\hat{\omega}_s = \omega_n + \mu_g(\omega_{q+e} - \omega_{n+q}) \tag{4.29}$$

where μ_g is a convergence factor that can be fixed to $2/p$ (p is the number of LSPs) to ensure convergence. Thereafter, the resulting LPC filter must be checked for stability (this is easily carried out by checking the order of the LSPs). If it is unstable, then $\hat{\omega}_s = \omega_n$.

The coder-specific parameters can directly be assigned as $\hat{\nu}_s = \nu_{q+e}$ or reestimated from an enhanced version of the residual, that is, by applying MMSE-LSA to the reconstructed noisy residual. The first method works better for car noise and the second one for babble noise. Finally, the improved bitstream FE estimates the SNR from the speech signals before and after enhancement in order to avoid that the enhancement algorithm can degrade speech with high SNR. In this last case, when the SNR is greater than a given threshold, the same assignment as for instability ($\hat{\omega}_s = \omega_n$) is used. After the whole enhancement process, LPCCs and energy are computed from the enhanced codec parameters.

As an example of performance, the baseline and the enhanced feature extractors yield WAcc performances of 96.2,95.5,92.0,71.6,22.3 % and 96.2,95.7,93.6,81.3,32.5 %, respectively, for SNRs (car and babble noise) of ∞,30,20,10,0 dB, and using direct assignment of the coder-specific parameters.

4.4.2 Log-Energy Computation in B-NSR Systems

As we have seen, the extraction of log-energy requires the decoding of the whole signal Gallardo *et al*. (1998) or a partial decoding (only the residual) Kim and Cox (2001a). However, this feature can be estimated from the codec bitstream. In general, the energy of a given frame in an LPC-type coder can be computed by integrating the LPC spectrum (see Equation (4.12))

$$E = \frac{\sigma^2}{2\pi} \int_{-\pi}^{\pi} \left| \frac{1}{1 - A(e^{j\omega})} \right|^2 d\omega \tag{4.30}$$

The computation of the residual energy σ^2 depends on the specific type of coder. For example, for an ACELP coder with the structure of Figure 4.12, σ^2 can be derived from Equation (4.21) as

$$E = g_p^2 E_v + g_c^2 E_c \tag{4.31}$$

assuming that the adaptive and algebraic codebook excitations are uncorrelated. The algebraic codebook contribution E_c is easily obtained from the algebraic code (summing the pulse square amplitudes). The computation of the adaptive codebook contribution E_v depends on how the codec builds the adaptive signal $v(n)$. For example, the computation of E_v for the G.723.1 codec can be found in Pelaez *et al*. (2001).

4.4.3 Alternative Parameter Conversion

We have also seen that most of the work on bitstream-based feature extraction deals with the conversion of the codec parameters into recognition features commonly used such as MFCC or LPCC. However, one could wonder whether it is possible to find a set of features more suitable to the codec parameters. Thus, in the case of a codec using LSPs to represent the short-time spectrum, it is possible to derive a *pseudo-cepstrum* that it is computed as (Choi *et al*., 2000)

$$\hat{c}_n = \frac{1}{n} \sum_{i=1}^{p} \cos(n\omega_i) \qquad (n \geq 1) \tag{4.32}$$

where $\{\omega_i;\ i = 1, \ldots, p\}$ are the LSP coefficients. This pseudo-cepstrum is an approximation to the LPC cepstrum, although with much less computation, since the LSP–LPC conversion is avoided. The experimental results on the QCELP codec in Choi *et al*. (2000) show that this new cepstrum yields the same performance as LPC cepstrum (86.3 % of WAcc) in a connected Korean digit task. Both cepstra are computed on a mel scale. The same parametrization is used for training and testing. For comparison, the same system yields 87.0 %, 83.6 % and 82.6 % of WAcc when recognizing from MFCCs

computed from the uncoded speech, MFCCs computed from reconstructed speech and LPCCs obtained from the quantized LSPs, respectively.

4.5 Distributed Speech Recognition

Unlike speech coding, for which several institutions have developed intense standardization activities as we could see in the previous sections, in the case of DSR these activities have been concentrated in the ETSI-STQ Aurora working group. The Aurora Project has its origin in the TELECOM-1995 meeting, in which several companies agreed to join efforts in order to establish a standard for wireless speech recognition. However, there have been other remarkable efforts in DSR before Aurora issued the FE, its first standard for DSR, in April 2000. Thus, although the Aurora group started its work oriented to mobile environments, DSR initially appeared more related to IP applications (Digalakis *et al*., 1998b; Stallard, 1997), for which its client-server architecture seems specially suitable.

As mentioned in the introduction section of this chapter, the information compressed and transmitted by the FE of a DSR system is the one corresponding to the speech features used by the recognition engine placed at the back end (see Figure 1.4). It is important to note that only the static features (computed from the current frame) are usually encoded, since the dynamic features (first and second derivatives) can be computed at the back end from the static ones.

In this section, we will summarize the main contributions on feature compression for DSR. The objectives are similar to those of speech coding, and can be summarized as follows:

- Low bitrate for a reduced transmission bandwidth
- Low computational complexity and memory requirements
- Negligible effect on the recognition performance
- Robustness against environmental variations and background noise

In principle, we are not concerned with the reconstruction of the speech signal. This is an important difference with respect to speech coding. While the compression techniques for speech coding (NSR) are oriented toward obtaining a good (subjective) quality of the synthesized speech, those applied in DSR must be directed to a good (objective) recognition performance. This is the main reason why NSR can lead us to a certain performance degradation. Moreover, in order to achieve good quality, speech coders conserve a certain degree of signal redundancy that is neither required nor desirable for recognition. As a result, a "good" speech coder will normally require a higher bitrate than a "good" encoder of recognition features. However, speech reconstruction, although not the goal in DSR, can be an added value to a DSR system. This is the reason why extended FE, including additional speech features to allow reconstruction, have also been developed (XFE and XAFE (ETSI, 2001, 2003c)).

A first in-depth study that undertakes some of the above-mentioned issues of DSR was carried out by Digalakis *et al*. (1998b). This work considers several possibilities of implementation. First, a natural scheme for DSR could be the use of a recognition system based on discrete HMMs (DHMMs). As seen in Chapter 2, those HMM models

are fed with discrete observations, which are obtained by a previous VQ process. Thus, one could transmit those observations and recognize from them. A typical DHMM-based system would work at about 3.8 kbps (Stallard, 1997). However, DHMMs perform worse and require more memory (to store observation probabilities) than SCHMMs or CHMMs. This is the reason why these authors are inclined to use recognition systems based on continuous pdfs along with a compression scheme independent of the recognizer, as we will see in the following subsections.

4.5.1 Scalar Quantizers

In Digalakis *et al*. (1998b) it is shown that it is possible to obtain good performance with a simple nonuniform scalar quantization of each feature. The quantization cells are determined from the empirical distribution of the feature, so that the feature values are uniformly distributed in those cells (Gersho and Gray, 1992). All the scalar quantizers use the same number of bits. The features are filterbank-based cepstrum coefficients. For telephone-quality speech (using 9 cepstral coefficients, MFCC(1–8) plus MFCC(0)), WERs of 12.7, 14.5 and 13.19 % were obtained when using G.711 (64 kbps), GSM-FR (13 kbps) and the proposed compression scheme at 3.6 kbps, respectively. Moreover, for high-quality speech (using 13 cepstral coefficients, MFCC(1–12) plus MFCC(0)), they obtained WERs of 6.55 and 6.88 % for uncompressed speech and the proposed feature compression at 3.9 kbps, respectively. Although both data sets, the telephone-quality one and the high-quality one, were not directly comparable, these results already pointed out the benefit of using the DSR architecture instead of NSR.

A very similar nonuniform quantization is applied in Weerackody *et al*. (2002) to a feature vector made up of 12 LPCCs and an energy coefficient, computed every 10 ms. Instead of a uniform bit allocation, 6 bits are devoted to the energy and the LPCCs 1–5, 4 bits to LPCCs 6–11, and 0 bits to LPCC 12 (which is equivalent to using its mean value). This bit allocation is based on an empirical analysis that showed that it is more important to put more allocation effort on the first cepstral coefficients, while the last ones can be quantized more coarsely. This fact will be analyzed in greater detail in the following subsections. With this bit allocation, the bitrate is 6 kbps, and provides a WER of 7.1 % on an IWR task (the performance without compression is 6.8 %).

Instead of quantizing the recognition features directly, it is also possible to apply DPCM encoding (Figure 4.1) to each feature, so that the scalar quantization is performed now on the prediction error. In Srinivasamurthy *et al*. (2000), first-order prediction is applied, and the prediction errors corresponding to the different MFCCs are uniformly quantized and Huffman-encoded. The same step size is used for all the MFCCs since they are normalized by their standard deviations prior to quantization. Different bit rates can be straightforwardly generated by changing the step size. This scheme results in a variable bitrate encoder. The reported results indicate that this predictive compression scheme outperforms the nonuniform quantization applied in Digalakis *et al*. (1998b).

4.5.2 Vector Quantizers and Product Codes

A higher level of compression and lower encoder complexity can be reached by using VQ and, in particular, *product code* VQ, instead of scalar quantization. This is an approach

that has been extensively applied in speech coding. Product code VQ involves the use of several codebooks to quantize parameter vectors. A simple example of product codes is *split vector quantization* (SVQ), which consists of partitioning the feature vector into several subvectors. This partitioning strategy involves a set of small codebooks instead of a single larger one (as in VQ), so that codebook search is faster and the memory requirements are lower. These properties make SVQ especially suitable for DSR, since feature compression takes place at the client. Scalar quantization is a degenerated case of SVQ in which each subvector contains a single feature.

In order to apply SVQ to DSR, there are two questions that must be answered: how to distribute the different features into subvectors and how to carry out the bit allocation for the different subvectors. Regarding the first question, an SVQ product code is optimal if the different subvectors are independent and the distance measure employed is separable (Makhoul *et al*., 1985). This problem is studied in Digalakis *et al*. (1999), where the authors propose two different approaches. The first one tries to follow the above optimality criterion by computing the correlation matrix of the considered feature vector and putting together those features that are more correlated. However, they also considered a simple knowledge-based approach for the case of a cepstral feature vector that consists of putting together consecutive cepstral coefficients. This last approach provided the best results and is, in fact, the most extended way of applying SVQ to DSR. Figure 4.21 illustrates this partitioning approach for a 13-dimensional feature vector (MFCC(1–12) + MFCC(0)) split into 5 subvectors. In comparison with the use of 3 subvectors, the 5-subvector setup converged faster to the baseline (unquantized features) performance as the bitrate was increased.

The remaining problem is how to allocate bits among the different codebooks. Unlike speech coding, where bit allocation is addressed with a minimum distortion criterion, in a DSR system bit allocation should be carried out on a minimum WER or maximum WAcc

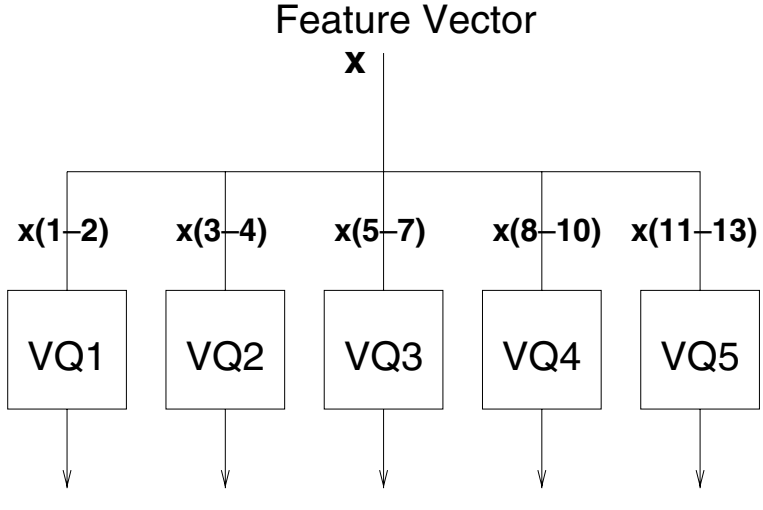


Figure 4.21 An example of feature compression based on split vector quantization

criterion. Of course, the optimal solution would be to obtain the performances corresponding to all possible bit allocations for a given bitrate and select the best one. However, this solution is clearly impractical even for medium complexity tasks. In Digalakis *et al*. (1998a, 1999), a heuristic iterative approach also is proposed that uses WER as performance measure. The procedure can be summarized as follows:

1. Initialization: choose an initial bit allocation and obtain the recognition performance.
2. For each subvector, increase by one the number of bits assigned to it, keeping the other subvectors the same, and obtain the WER for that configuration. The bit is finally assigned to the subvector whose corresponding configuration yields the best performance.
3. Stop the procedure if the desired performance has been reached or if there is no more available bits to allocate. Otherwise, go to the previous step.

In order to reduce the computational cost involved, the increment in the number of bits in step 2 can be greater than one. The different bit allocations consecutively obtained and their performances, for the 13-dimensional MFCC vector and the 5-subvector partition previously mentioned, are shown in Table 4.3. The experimental setup is the same as used in Digalakis *et al*. (1998b), described in the scalar quantization section 4.5.1. There are several interesting conclusions that can be extracted from the table. First, we can see that the number of bits assigned to the different cepstral subvectors is higher for the lowest order cepstral coefficients than for the highest order ones. It is also observed that the performance of the uncompressed speech features (6.55 %) is mostly obtained with the 2 kbps compression scheme, which means almost half the bitrate required by nonuniform scalar quantization. However, this work also shows that when the speech signal is corrupted by a 24 dB additive noise, the baseline WER increases from 6.55 to 8.51 %, while the 2 kbps scheme increases from 6.63 to 12.19 %. In order to maintain the recognition performance in this case, a bitrate of 2.7 kbps is required.

An alternative to the previous growing strategy is to apply a pruning one, where we delete bits one by one from an initial bit allocation and we finally keep the configuration that yields the lowest performance reduction (Boulis *et al*., 2002). As arguments to

Table 4.3 Recognition performance (WER) for different bit allocations for the SVQ compression scheme of Figure 4.21 (after Digalakis *et al*., 1998)

Subvector					Bitrate (kbps)	WER (%)
1	2	3	4	5		
Bits assigned						
3	3	2	2	2	1.2	16.79
5	3	2	2	2	1.4	11.71
5	3	4	2	2	1.6	9.30
5	3	4	4	2	1.8	8.10
5	4	4	4	2	1.9	6.99
5	5	4	4	2	2.0	6.63

support pruning instead of growing, Boulis *et al*. (2002) mention that this choice provides better results in the case of using the minimum distortion criterion with additive distance measures (although we are using a maximum performance criterion) and that pruning results in smaller approximation errors at high bitrates than at low ones (they assume that the DSR encoder will work at high rates more frequently). In order to obtain a relief from the computational burden, the authors propose the use of a distortion-based criterion to select the best M candidates to be pruned and then to choose from them the one yielding the lowest WER. With this pruning approach, it is possible to compress MFCC(1–8) plus the frame energy at 2.6 kbps without any significant performance reduction. The selected feature distribution is MFCC(1,2), MFCC(3,4), MFCC(5,6), MFCC(7,8) and energy alone, and the obtained bit allocation for the 2.6 kbps case is 6, 6, 4, 6, and 4 bits, respectively. In order to quantize each subvector, fixed-length TSVQ is employed. The experiments in Boulis *et al*. (2002) were carried out on a continuous speech recognition task associated to the Defense Advanced Research Project Agency (DARPA) communicator program with a 2647-word vocabulary.

As we will see later in this subsection (see section 4.5.2.1), there is also a third alternative for the bit-allocation problem based on the combination of the minimum distortion and the optimal classification criteria by means of using an information theory framework, avoiding thus the need of a heuristic procedure.

The Aurora standards use SVQ quantization for feature compression. They employ a 14-dimensional feature vector, containing 13 mel-frequency cepstral coefficients (MFCC(0–12)) and the log-energy (logE). These features are grouped into 7 pairs for SVQ. The first six pairs are MFCC(1,2), MFCC(3,4), MFCC(5,6), MFCC(7,8), MFCC(9,10) and MFCC(11,12), and the corresponding SVQ codebooks have 64 centroids (6 bits) for pairs 1 to 5 and 64 or 32 centroids (6/5 bits, depending on the specific standard) for pair number 6. These codebooks are built using Euclidean distance measures. The seventh pair consists of MFCC(0) and logE, that is, the energy information, which is encoded with a 256-centroid codebook (8 bits) constructed using a weighted distance measure. This separated treatment accounts for the larger variability of the energy parameters. The details of this quantization scheme are given in Chapter 7. Since a feature vector is computed every 10 ms, the raw bitrate is 4.4 kbps, although is increased to 4.8 kbps after including overhead and error protection bits. This relatively high bitrate is the reason why no special effort is required for bit allocation. The recognition performance of the Aurora compression is detailed in Pearce (2000).

A more sophisticated compression algorithm of speech features can be found in Ramaswamy and Gopalakrishnan (1998). This work proposed the use of a feature vector $\mathbf{x}$ that consists of 12 mel-cepstral features (C1–C12) plus the 0th order cepstral coefficient C0 or the frame energy. Features are computed every 10 ms. The compression scheme is depicted in Figure 4.22. It has the predictive structure already seen in Figure 4.1, with a first-order predictor (the prediction coefficient is $a_1 = 1$), and uses multistage vector quantization, which is another type of product code VQ that involves primary and secondary codebooks. The primary VQ codebook has 4096 centroids (12 bits) and is used to quantize the prediction error vector $\mathbf{e}$. The 13-dimensional residual vector $\mathbf{r}$ is divided into 3 subvectors corresponding to C1–C6, C7–C12 and C0 or the energy, respectively, for a secondary SVQ quantization. The energy parameter is treated separately, since it is quite sensitive to environmental variations. Thus, a higher level of robustness against

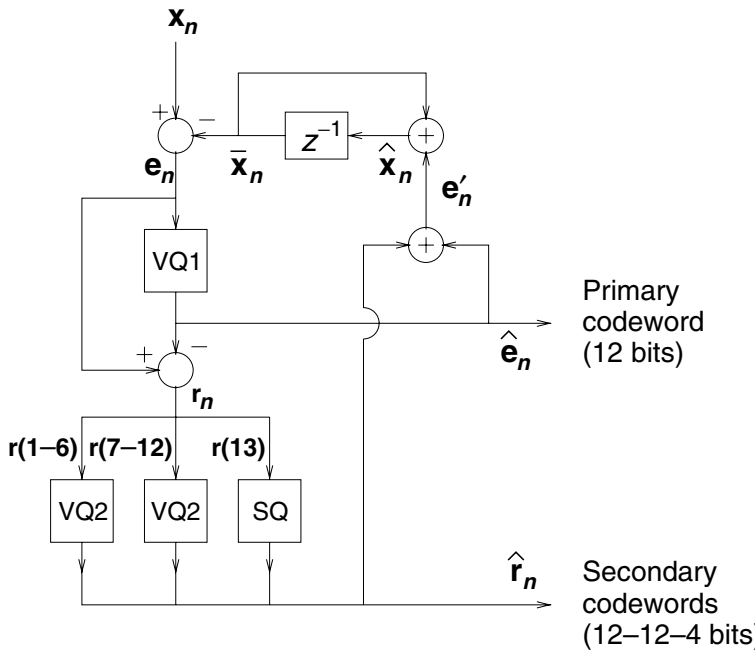


Figure 4.22 Feature compression based on predictive coding and multistage VQ

acoustic noise is obtained. The first two subvectors are quantized with 2 secondary SVQ codebooks using again 4096 centroids, while a scalar 4-bit quantizer is used for the energy (or C0) residual. The final encoded feature vector requires 40 bits (12 for the primary codeword, 12 for each secondary codeword and 4 for the energy residual), and the final bitrate is 4 kbps. At the decoder, a feature vector $\hat{\mathbf{x}}$ is reconstructed from the received vectors $\hat{\mathbf{e}}$ and $\hat{\mathbf{r}}$. All the VQ quantizers use a Euclidean distance measure and are designed by applying the k-means algorithm. The codebook search is speeded up using a fixed-length TSVQ search (reviewed in Chapter 2). Here a two-step search is used in which the 4096 centroids are group into 64 groups containing 64 centroids each. In the first step is determined the group, and in the second one the specific centroid, so that it is necessary only to compute 128 distances instead of 4096. Additionally, it proposes the use of a short-integer (2 bytes) representation of codebook entries (instead of floating-point) for memory saving, so that all the codebooks can be stored in 200 kbits approximately. The compression algorithm was tested using the Wall Street Journal (WSJ) task and excellent results were obtained: an average error rate of 11.4 % for the uncompressed features, and 11.2 and 11.3 % for the compressed features with and without using the integer representation, respectively. The algorithm also performed quite well when applied to different languages and non-clean test sets. A similar and simpler encoder for the MFCCs that uses first-order prediction and SVQ on the residual vector can be found in Bernard and Alwan (2002).

All the previous product code schemes split a given feature vector (or a vector derived from it) into subvectors and perform SVQ at some stage. This approach can be called

intraframe VQ since the different subvectors are built with features extracted from the same frame. An alternative is *interframe* VQ. In this case, the vectors to be quantized are built with features from different frames. The advantage of this approach is that consecutive values of the same feature are likely to be more correlated than adjacent features within the same frame as in intraframe VQ, and this leads us to a more efficient compression. As an example, in Boulis *et al*. (2002) applied interframe VQ to a 9-dimensional feature vector, containing MFCC(1–8) and the frame energy, by utilizing 9 codebooks (one for each feature) for two-dimensional subvectors which consists of the two consecutive values of the considered feature. This scheme provides a performance similar to the one obtained with the intraframe scheme also applied in that work (and explained previously), that is, no degradation with respect to the unquantized features, but with a bitrate of 1.2 kbps. The bit allocation obtained for this bitrate by the "driven by WER" method, also proposed in this reference (and also explained above), is 4,3,3,3,2,3,3,4,5 bits for MFCC(1–8) and the frame energy, respectively.

4.5.2.1 Classification-oriented Quantization

The compression schemes mentioned so far are based on well-known techniques that use traditional distance measures, such as Euclidean or weighted Euclidean, which are minimized during both quantization and quantizer design. Thus, if $d(\mathbf{x}, \hat{\mathbf{x}})$ is the distance between an input vector $\mathbf{x}$ and a codebook centroid $\hat{\mathbf{x}}$, then the codebook is designed by minimizing its expected value

$$E[d(\mathbf{x}, \hat{\mathbf{x}})] = \int_{\mathbf{x}} p(\mathbf{x}) d(\mathbf{x}, \hat{\mathbf{x}})\, d\mathbf{x} \tag{4.33}$$

However, these classical methods do not take into account the fact that the final goal of a recognition system is to minimize the classification error. Obviously, the problem of designing quantizers that provide a minimum error rate is the lack of mathematical coverage. This problem can be overcome by using the mutual information (MI) between the random variables $\mathbf{X}$ and C, representing feature vectors ($\mathbf{x}$) and classes (c), respectively (Srinivasamurthy *et al*., 2003). The MI is defined as

$$I(\mathbf{X}, C) = \int_{\mathbf{x}} p(\mathbf{x}) \sum_{c} p(c|\mathbf{x}) \log \left(\frac{p(c|\mathbf{x})}{p(c)} \right) d\mathbf{x} \tag{4.34}$$

where it must be considered that a vector $\mathbf{x}$ is part of a vector sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$ that belongs to a certain class c. Since

$$H(C|\mathbf{X}) = H(C) - I(\mathbf{X}, C) \tag{4.35}$$

the MI can be interpreted as the amount by which the uncertainty about C is reduced after observing $\mathbf{X}$. If $\hat{\mathbf{x}}$ represents a quantized version of $\mathbf{x}$, then we can consider that the optimal quantizer is the one that minimizes the MI loss,

$$I(\mathbf{X}, C) - I(\hat{\mathbf{X}}, C) = H(C|\hat{\mathbf{X}}) - H(C|\mathbf{X}) = \int_{\mathbf{x}} p(\mathbf{x}) \sum_{c} p(c|\mathbf{x}) \log \left(\frac{p(c|\mathbf{x})}{p(c|\hat{\mathbf{x}})} \right) d\mathbf{x} \tag{4.36}$$

Comparing this expression with Equation (4.33), we see that the minimization of the MI loss involves the following distance measure:

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{c} p(c|\mathbf{x}) \log \left(\frac{p(c|\mathbf{x})}{p(c|\hat{\mathbf{x}})} \right) \tag{4.37}$$

Once we have identified the distortion measure involved by the minimum MI loss criterion, we can use it for both the design of the VQ codebook, by using a well-known procedure such as k-means, and the quantization of incoming feature vectors. Furthermore, the feature vector used by the recognition system can be split into several subvectors, and the corresponding SVQ codebooks designed with the described technique. This distance measure can also be used by a bit-allocation procedure, such as the generalized BFOS algorithm (Riskin, 1991), to provide an optimal distribution of the available bits. This rate allocation scheme has been successfully applied to speech recognition in Srinivasamurthy *et al*. (2003), using uniform scalar quantizers (1-dimensional subvectors), and in Srinivasamurthy *et al*. (2004) to the Aurora compression FE scheme i.e., two-dimensional subvectors) as shown in Table 4.4. In this last case, it is necessary only to change the Aurora SVQ quantizers, while the rest of the standard is retained. The class labels c correspond to the 45 phonemes of the recognition task. The table reflects the same trend as the heuristic bit allocation performed in Digalakis *et al*. (1998a), that is, the lowest order cepstral coefficients and the energy features require more bits than the highest order cepstral coefficients.

A different approach to classification-oriented quantization is to use different SVQ codebooks for different phonetic classes (Deng *et al*., 2002). Using this approach, each input frame must be classified into a phonetic class (phonemes) and then quantized with the corresponding set of SVQ codebooks. In Deng *et al*. (2002) three different codebooks are used for subvectors, MFCC(0), MFCC(1–6) and MFCC(7–12), and the bit allocation is carried out on a minimum WER basis similar to the one commented on at the beginning of this section. This phone-dependent coder has yielded good performance on the WSJ

Table 4.4 Different bit allocations for the Aurora FE compression scheme: original of Aurora and those obtained with the generalized BFOS algorithm by minimizing both the MSE and MI loss (after Srinivasamurthy *et al*., 2004)

Subvector	Bits allocated		
	Aurora	MSE	MI loss
MFCC(0),logE	8	8	11
MFCC(1,2)	6	9	7
MFCC(3,4)	6	7	6
MFCC(5,6)	6	6	6
MFCC(7,8)	6	5	5
MFCC(9,10)	6	5	5
MFCC(11,12)	6	4	4
Total	44	44	44

database for bitrates between 4.8 and 1.6 kbps, by using a simple frame Mahalanobis distance classifier.

4.5.2.2 Mitigation of the Compression Degradation

As we have seen, the VQ-based compression schemes can degrade the system performance, especially when the bitrate is low. This is due to the "hard decision" involved by simple substitution of an input vector $\mathbf{x}$ by the nearest centroid μ_n (n is the corresponding quantized index). However, we are not considering any other information such as the covariance matrix Σ_n of the corresponding cell, which can be used to soften the decision. In order to do this, the observation probabilities of a continuous HMM recognizer (Equation (2.71)) can modified as (Arrowood and Clements, 2004)

$$b_i(\mu_n) = \sum_k c_{ik} \mathcal{N}(\mu_n; \boldsymbol{\mu}_{i,k}, \Sigma_{i,k} + \Sigma_n) \tag{4.38}$$

as it was already established in section 2.9.2.2 (Equation (2.114)). The HMM state covariance matrices are broadened in order to take into account the uncertainty introduced by the VQ quantization. This technique has been successfully applied in Arrowood and Clements (2004) to an Aurora-based encoder working at 3.3 kbps (the SVQ quantizers use two bits less than in the standard). The technique could also be useful to lower the bitrate by treating the degradation at the decoder.

Another possible solution to mitigate the degradation due to compression in DSR is to apply the model adaptation techniques of Chapter 2 (MLLR and MAP), which already were successfully applied for the NSR case (Srinivasamurthy *et al.*, 2001a). However, unlike NSR, where the mismatch was due to compression and decoding artifacts, we have only to deal now with compression. Therefore, these techniques will be mainly useful in the case of low bitrate feature encoders.

4.5.3 Very Low Bitrate PLP-based Compression for DSR

The above compression techniques do not put special emphasis in reaching a very low bitrate, with less than 1 kbps. Such a very low bitrate could be useful, for example, for transmitting the recognition features as side information along with the parameters of a given speech codec. This system architecture (depicted in Figure 4.23) can

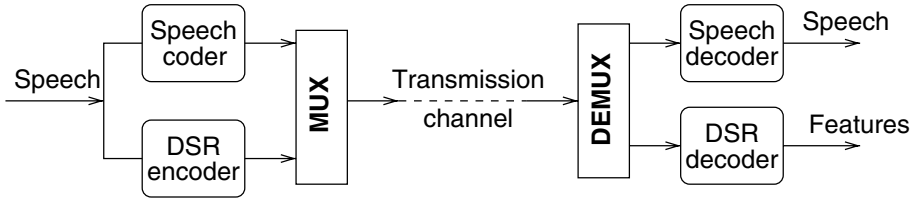


Figure 4.23 Multiplexed NSR/DSR architecture (after Gunawan, 2001)

provide both high-quality speech and high recognition accuracy with a small increase in the final bitrate. For example, if the dual-rate G.723.1 coder is employed, the system could operate at 6.3 kbps as normal operation mode, and at 5.3 kbps for a simultaneous coding/recognition mode without reaching the maximum bitrate of 6.3 kbps (Gunawan and Hasegawa-Johnson, 2001). In order to reach a very low bitrate, the PLP parametrization (Hermansky, 1990) (see Chapter 6 for details) is a good candidate, since it obtains its best performance with fewer parameters (typically, 5 linear prediction coefficients) than a conventional LPC analysis (typically, 10 linear prediction coefficients). Besides, this is a robust parametrization based on perceptual considerations.

In Gunawan and Hasegawa-Johnson (2001), a 30 ms asymmetrical window (the same as in the CS-ACELP coder of reference Salami *et al*. (1998)) is applied every 10 ms in order to obtain a set of 5 PLP coefficients, which are converted into LSPs for a suitable VQ quantization, since the LSP coefficients are quite correlated. These LSPs are quantized with LBG-trained codebooks with a number of centroids that vary from 64 (6 bits) to 256 (8 bits). In order to achieve a higher compression, the PLP analysis can be optionally carried out every 20 ms (downsampling), and then the LSPs are linearly interpolated at the decoder to restore the original frame rate of 10 ms, thereby taking advantage of the slow temporal variation of the LSPs. This involves a bitrate that ranges from 600 to 800 bps (10 ms of frame shift), or from 300 to 400 bps (20 ms of frame shift). At the decoder, the PLPs are restored from the LSPs, and converted into LPCCs. The recognition experiments in an IWR system showed that the 400 bps encoding scheme with 8 bits per frame and interpolation yielded a very acceptable performance.

As we have mentioned, the LSP vectors are a representation of the speech frames suitable for VQ because of their high intraframe correlation. Besides, as also mentioned, they also present a high temporal correlation. This interframe correlation can be used to improve their quantization by applying predictive coding (see Figure 4.1). This scheme was applied by Bernard and Alwan (2001) to the six LSPs extracted from six PLPs in three steps: 1) mean removal, 2) first-order prediction and 3) VQ quantization of the residual vector (using a weighting distance). Using a 6 bit codebook and applying the same downsampling and interpolation of the LSPs as explained above, this encoding scheme has a bitrate of 300 bps and provides an accuracy comparable to the use of unquantized features. However, the extra compression obtained with prediction and interpolation results in a higher sensitivity to transmission errors (Bernard and Alwan, 2002).

4.5.4 Transform Coders

4.5.4.1 DCT-based Coding

The slow varying characteristic with time of the features used for recognition means that they can be considered to be low-pass time functions, so that a transform that provides a frequency decomposition such as the DCT can be suitable to implement a TC encoder of those features. The most straightforward DCT encoding of DSR features consists of implementing an independent TC encoder, similar to that of Figure 4.5, for each speech feature, considered like a function of time. The incoming feature samples are buffered and the resulting blocks (vectors) DCT-transformed. The convenience of this approach is

supported by two facts, the overlapping nature of the feature extraction process and the slow varying characteristic of the speech production process, which are both responsible for the high degree of correlation present at the time feature function. However, in order to obtain an efficient compression, it is necessary to give an appropriate solution to the implementation issues set out in section 4.2.1. A DCT-based encoder using this basic scheme and dealing with some of these issues has been proposed by Milner and Shao (2003). It encodes the 14 features of the Aurora standard (12 MFCCs plus MFCC(0) and log-energy), which are computed at a frame rate of 100 Hz. Each feature x_n ($n = 0, \ldots, N-1 = 13$) is considered like a function of time ($x_n = x_n(t)$). Consecutive feature samples are then grouped into blocks of size M, as depicted in Figure 4.24. For a block starting at a given time t_0, we obtain a vector $\mathbf{x}_n = (x_n(t_0), x_n(t_0+1), \ldots, x_n(t_0+M-1))$. The DCT provides a transformed vector $\mathbf{y}_n$. The solutions to the implementation issues of section 4.2.1 provided by this work can be summarized as follows:

1. Block size M: It is selected as $M = 8$ by taking into account several facts such as delay or robustness to erasures in the transmission channel.
2. Zonal sampling: The most relevant frequencies of the feature time functions are typically between 1 and 16 Hz (Hermansky and Jain, 1999). Since the eight frequencies of this scheme range from 0 to 50 Hz, zonal sampling can help in selecting the relevant information to be quantized and transmitted. The selected truncation size M' ($M' < M$) depends on the bitrate, as we comment later.
3. Bit allocation: It is applied as a bit assignment similar to that of Equation (4.11) to every transformed coefficient $y_n(m)$ and which is given by

$$R_{n,m} = R + \frac{1}{2}\log_2 \frac{\sigma_{n,m}2}{\left[\prod_{n=0}^{N-1}\prod_{m=0}^{M'-1}\sigma_{n,m}^2\right]^{1/2}} \tag{4.39}$$

where RNM' is the total number of available bits (bitrate divided by the frame rate).

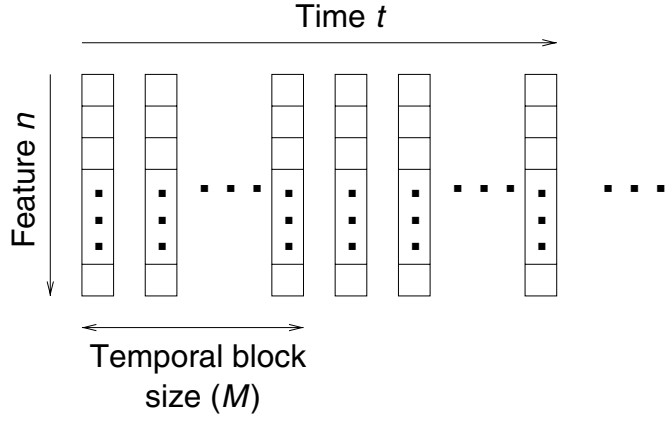


Figure 4.24 Blocking of static features

4. Choice of the quantizers: They are nonuniform quantizers designed by applying the Lloyd–Max algorithm and considering Laplacian pdfs for the coefficients being quantized.

The experimental results over the Aurora TI digits database (baseline WAcc's: 98.6 % and 92.9 % for clean and 10-dB noisy speech, respectively) show that it is possible to achieve a good recognition performance at 2400 bps (98.6 and 93.4 % with $M' = 4$), 1200 bps (98.0 and 90.1 % with $M' = 2$) and 800 bps (97.1 and 84.0 % with $M' = 2$). The most important conclusion of this work is that the selection of the truncation length M' is critical and strongly depends on the bitrate. This can be explained by considering that if the bitrate is high, there are enough bits to be distributed among the transformed coefficients, so that it is possible to assign bits to both the low and high frequency components. On the contrary, if the bitrate is low, it is better to perform a good quantization of the lower frequencies (the most relevant) and to truncate the higher ones, rather than to quantize these last ones. This basic TC encoder can be enhanced by using DPCM encoding/quantization of some transformed components and by applying a differentiated treatment to the energy features (Kiss and Kapanen, 1999). These modifications are useful to provide an encoding scheme robust against acoustic variations and noise.

4.5.4.2 2D-DCT Coding

When the recognition features are MFCCs, the basic DCT-based method studied above can be viewed as if a two-dimensional (2D) DCT was applied over the filterbank outputs, since MFCCs are obtained from them through a DCT over the frequency domain. This transform along with the additional transform applied over the time domain for each feature is equivalent to a 2D-DCT transform.

In general, we can consider the sequence of feature vectors as a 2-D signal $X = \{x(n, t)\}$ ($n = 0, \ldots, N-1$; $t = 0, \ldots, T-1$), where n is the feature number and t the time index, so that we can apply a 2D-DCT transform in order to exploit the interframe and intraframe correlations. The block size is $N \times M$, that is, a block consists of M consecutive feature vectors. Thus, block number j is built as $X_j = x(n, t)$ ($n = 0, \ldots, N-1$; $t = jM, \ldots, (j+1)M - 1$), and the corresponding transform is $Y_j = CX_jC^t$ (C stands for the DCT matrix). In the case where the matrix X contains MFCC coefficients, the energy of the features contained in X will be mainly concentrated in the first columns of the transformed matrix since the intraframe correlations (along index n) have been already exploited during the computation of the MFCCs (by applying the 1D-DCT to the filterbank log-outputs). In order to take into account the fact the higher-order MFCCs are less important that lower ones, each 2D block X_j can be split into two different subblocks that can be quantized with different accuracies. Thus, the first subblock is of size $N_1 \times M$ and contains the first MFCCs ($n = 0, \ldots, N_1 - 1$), and the second one is of size $N_2 \times M$ and contains the last ones ($n = N_1, \ldots, N_1 + N_2 - 1$). Then each subblock is independently 2D-DCT transformed.

In the same way as for 1D-DCT encoders, we can play with the block size, the truncation length, the bit allocation, and so on. For example, in Zhu and Alwan (2001), a 2D-DCT encoding is applied to MFCC features with $N = 12$ (MFCC(0) is excluded) and $M = 12$. The zonal sampling is dynamic, that is only the components of each block with the highest

energy are (scalarly) quantized and transmitted (the rest are set to zero), so that the chosen components can vary from block to block. The position of the nonzero components is specified by means of run-length encoding. A graceful degradation is obtained with bitrates as low as 624 bps in noisy conditions.

An example of the subblock approach can be found in Hsu and Lee (2004b), where two subblocks of MFCCs are used with $N_1 = 6$, $N_2 = 6$ and $M = 12$. An initial zonal sampling that retains only the first two columns of each transformed subblock is applied (24 components). From these 24 components, only 18 are finally retained (those that provide the minimum WER), 10 corresponding to the first subblock and 8 to the second one. The bit allocation for those 18 components is also carried out to minimize WER in a heuristic manner so that a WER similar to that of the baseline system (without compression) can be achieved at 1.45 kbps. A very interesting contribution of (Hsu and Lee, 2004b) is a study of how the 2D-DCT compression performs in the presence of acoustic noise. This study was carried out using the Aurora FE (extracting 12 MFCCs plus log-energy) and the Aurora-2 task. The conclusion was that the 2D-DCT scheme is quite robust in presence of nonstationary noise, but not so much for stationary noise. This can be explained from the fact that the 2D-DCT compression mainly tries to look for temporal correlations, as a stationary distortion could be, therefore, more damaging.

4.5.4.3 KLT-based Coding

A more sophisticated quantization based on TC can be developed by assuming that the source pdf can be modeled by a *Gaussian mixture model* (GMM), which is noted as M. This scheme has been proposed in Paliwal and So (2004a); Subramaniam and Rao (2003) and applied to DSR in Paliwal and So (2004b). The GMM pdf is a mixture of m Gaussians (clusters in the following)

$$p(\mathbf{x}|M) = \sum_{i=1}^{m} c_i N_i(\mathbf{x}; \mu_i, \Sigma_i) \tag{4.40}$$

where $\mathbf{x}$ is the input vector (p-dimensional), and c_i, μ_i and Σ_i are the weight, the mean vector and the covariance matrix of the ith cluster, respectively. The quantization of an input vector is carried out in a decorrelated space by applying the optimal transform, that is, the KLT. As previously mentioned in the speech coding section, the problem of the KLT is that it is a signal-dependent transform. In the GMM-based encoder, this problem is palliated by using a different KLT K_i for each cluster i, which diagonalizes Σ_i. The corresponding set of eigenvalues is noted as $\lambda_i = (\lambda_{i,1}, \ldots, \lambda_{i,p})$ and the new covariance matrix as Λ_i. An important characteristic of the GMM-based encoder is its bit allocation procedure. If b_{tot} is the total number of bits available to quantize an input vector, then the total number of codepoints $2^{b_{tot}}$ must accomplish,

$$2^{b_{tot}} = \sum_{i=1}^{m} 2^{b_i} \tag{4.41}$$

where b_i is the number of bits assigned to cluster i. Using a minimum distortion criterion, the following bit allocation among clusters is obtained:

$$2^{b_i} = 2^{b_{tot}} \frac{(c_i \bar{\lambda}_i)^{\frac{p}{p+2}}}{\sum_{i=1}^{m} (c_i \bar{\lambda}_i)^{\frac{p}{p+2}}} \tag{4.42}$$

where $\bar{\lambda}_i$ is the geometric mean of the eigenvalues of cluster i

$$\bar{\lambda}_i = \left(\prod_{j=1}^{p} \lambda_{i,j} \right)^{\frac{1}{p}} \tag{4.43}$$

The optimal bit allocation for the different components (j) of the vector in a cluster (i) is given by applying Equation (4.11) to this case

$$b_{i,j} = \frac{b_i}{p} + \frac{1}{2} \log_2 \frac{\lambda_{i,j}}{\hat{\lambda}_i} \tag{4.44}$$

The quantization of an input vector $\mathbf{x}$ in a given cluster i is performed as follows: first the corresponding mean is subtracted, and the result is transformed and normalized by the corresponding variance,

$$\mathbf{z}_i = \Lambda_i^{-1} K_i (\mathbf{x} - \mu_i) \qquad (i = 1, \ldots, m) \tag{4.45}$$

Then, every component of $\mathbf{z}_i$ is scalarly quantized, obtaining a new vector $\hat{\mathbf{z}}_i$. The vector is reconstructed by means of

$$\hat{\mathbf{x}}_i = K_i^t \Lambda_i \hat{\mathbf{z}}_i + \mu_i \tag{4.46}$$

The cluster $i = opt$ that provides the lowest distortion $d(\mathbf{x}, \hat{\mathbf{x}}_i)$ is selected. Finally, the information to be transmitted is the index corresponding to the codepoint of the selected cluster. This index belongs to the range $[0, 2^{b_{tot}} - 1]$. This interval must be divided into several subintervals corresponding to the different clusters and according to the bit allocation of Equation (4.42). The whole encoding process is illustrated in Figure 4.25. At the decoder the selected cluster and codepoint can be identified, so that the decoded vector is $\hat{\mathbf{x}}_{opt}$. In Paliwal and So (2004b) this compression technique is applied to a vector obtained by concatenating N consecutive feature vectors containing 12 MFCCs (MFCC(1–12); the energy features are excluded), in order to exploit both intraframe and interframe correlations, so that the total number of dimensions is $12N$. The scalar quantizers are designed through the Lloyd–Max algorithm and using the above bit allocation. The recognition was carried out on a clean subset of the Aurora-2 database. The experimental results show that the performance rapidly drops (below 90 % of WAcc) for a bitrate below 1200 bps for scalar quantization with Lloyd–Max quantizers, and below 800 bps for the GMM scheme using 1 frame per vector. The GMM-based technique with $N = 5$ frames always provides the best results and can even operate at 300 bps with 92.96 % of WAcc.

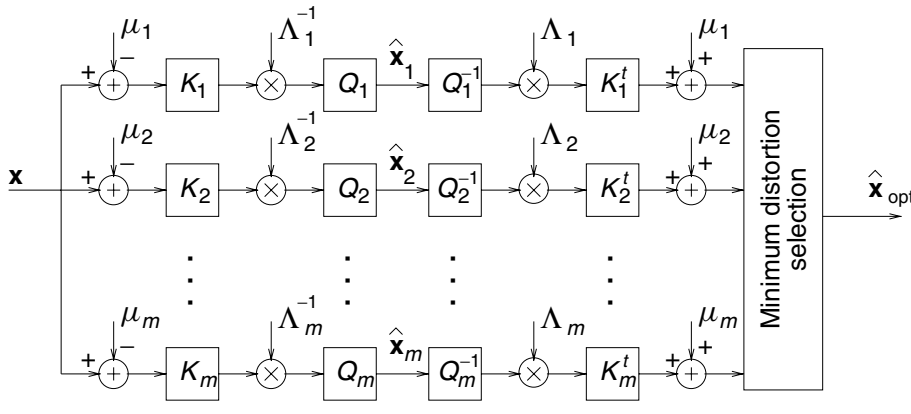


Figure 4.25 Block diagram of the GMM-based encoder (after Paliwal and So, 2004)

4.5.5 Scalability in DSR Systems

A system involving digital transmission is *scalable* if it can operate at different levels of accuracy in order to adapt to working conditions such as a variable transmission channel quality or a variable number of users accessing the system that could saturate it. Thus, a scalable encoder must be able to switch its bitrate to adapt to those conditions. A possible solution to the scalability problem is to have several encoders at different rates as, for example, we have seen in the case of the AMR speech coder. In general, any system using VQ as the quantization method must use this solution for adaptation, since a bitrate change involves a different VQ codebook. However, this introduces large storage costs and does not allow a continuous variation of the bitrate. Scalability is more natural for scalar quantization or TC encoders. A first step to scalability in DSR is the variable bitrate encoder of reference Srinivasamurthy *et al*. (2000), previously described in section 4.5.1. Also, the TC encoders are easily scalable by modifying the bit allocation and/or the number of transformed components as is proposed in (Zhu and Alwan, 2001). The GMM-based block quantization described in the previous subsection is also scalable, since, assuming that the bitrate is known, the corresponding bit allocations can be computed effortlessly at both client and server, by simply applying Equations (4.42) and (4.44).

A more sophisticated scalable encoder must provide multiresolution or an embedded bitstream that contains a basic layer and a refinement layer. Srinivasamurthy *et al*. (2001b) have extended their earlier work (Srinivasamurthy *et al*., 2000), where a DPCM-based scalable encoder (for each recognition feature) is obtained from two parallel DPCM loops. The first loop provides a coarse quantization $\hat{e}(n)$ (with quantization step size Δ_c) and the second a finer one $\hat{E}(n)$ (with quantization step size Δ_f) of the residual $e(n)$. A quantization bin of the coarser loop involves (intersects) several bins of the finer one ($\lceil \Delta_c/\Delta_f \rceil + 1$ at most) as shown in Figure 4.26. Thus, once the basic information corresponding to the coarser loop is transmitted, it is enough to transmit, as refinement information, the right bin among those intersected bins. Additional compression is achieved by applying context-dependent entropy coding plus run-length coding.

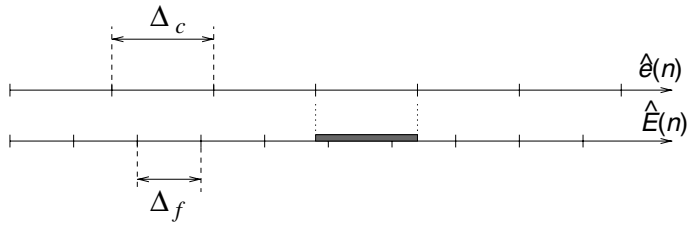


Figure 4.26 Coarse and fine uniform scalar quantization for scalable encoding (after Srinivasamurthy *et al*., 2001)

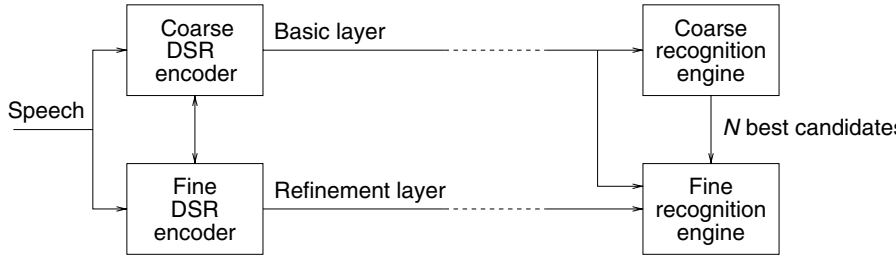


Figure 4.27 Fully scalable recognition system

Encoder scalability can be combined with recognizer scalability in order to obtain a fully scalable RSR system. This idea is depicted in Figure 4.27. A coarse recognition engine can be used to provide a list of the N best recognition candidates from the basic encoding layer. It must be noted that this list can contain a single candidate, so that no further refinement is required. Otherwise, a finer recognition engine can be applied to select the recognized sentence from the list by using, if this is possible, the refinement information. In Srinivasamurthy *et al*. (2001b), a scalable recognition system is proposed which uses DTW for the coarse recognition engine and HMM for the fine one.

4.6 Comparison between NSR and DSR

The dichotomy DSR/NSR was already discussed in the Chapter 1. Although the advantages of DSR over NSR were clear, we should also take into account the following issues:

- The bandwidth constraints: From the previous sections, it is clear that DSR requires a smaller bitrate than NSR. Thus, we have seen that acceptable performances can be obtained with speech coders between 5 and 10 kbps, while DSR requires 2 kbps or less. However, it must also be taken into account that the final bitrate (after channel coding) in some networks can be the same for DSR and NSR. An example of this is the case in which we want to implement an RSR system using a GSM traffic channel.

We could implement an NSR system over the voice channel or a DSR system over a data channel, but the final bitrate would be 22.8 kbps in both cases.

- The type of client device and the associated underlying network: NSR over a voice channel allows the use of the existing phones without modifications, while DSR would require handsets with feature extraction capabilities. On the other hand, DSR can be easily introduced (as new software) if other devices like personal digital assistants (PDAs) or notebooks are used.
- The type of interaction: Speech may be combined with other inputs (i.e. keypads) and the system response may include different audiovisual outputs (speech, audio, text, etc.). Under this multimodal scheme, DSR over a data channel seems more natural than NSR over both voice and data channels.
- Availability of the transmitted speech: If we want to have the possibility of reconstructing the speech from the user accessing the system, NSR is, in principle, more appropriated. However, we must take into account the fact that speech can also be reconstructed from speech recognition features, as is implemented in the Aurora XFE and XAFE.
- Robustness against channel errors or acoustic degradation: In general, DSR is more robust than NSR. However, it must also be taken into account that there are powerful techniques that allow both the mitigation of channel errors and the reduction of acoustic noise that can compensate the performance reduction in both cases, DSR and NSR. Chapters 5 and 6 will be devoted to them. Comparisons between NSR and DSR can be found in Fingscheidt *et al*. (2002); Ion and Haeb-Umbach (2005b); Kelleher *et al*. (2002); Kiss (2000); Pearce (2000). There are several important conclusions that can be extracted from those comparisons:
  - For clean speech, DSR provides better performance than NSR. The performance of both approaches is excellent for low complexity tasks, but it tends to diminish as the complexity is increased. This effect is much more noticeable in NSR, although it can be palliated by matched training/testing.
  - In general, DSR is more robust against channel errors, since its performance is not affected by small or moderate channel degradation, while the NSR performance decreases continuously as the channel degradation is increased. This effect was already shown in Table 3.1 for the GSM-EFR codec and ETSI DSR FE standard. However, if we use a variable source bitrate codec such as AMR, the corresponding NSR system can perform better than DSR for C/I less than 4 dB (Ion and Haeb-Umbach, 2005b). It must be considered that, for a fair comparison, NSR and DSR should be compared using the same final bitrate (after channel coding). Since DSR usually requires a lower source bitrate, it can include a higher channel bitrate, what results in a more protected encoding scheme. However, this may not be true for the AMR codecs with a lower bitrate.
  - In general, DSR shows more robustness against acoustic degradation. It is interesting to note the following particular aspects:
    - In the case of stationary noises NSR can perform better. A possible explanation for that is that speech codecs are optimized to encode and decode speech and not other types of sounds.

- Training/testing mismatch affects NSR more than DSR.
- The robust Aurora AFE FE has been shown to be more effective for DSR than for NSR (when applied to the decoded speech) (Kelleher *et al*., 2002).

The XAFE DSR standard was approved by 3GPP in June 2004 as the recommended codec for speech enabled services. This decision was based on the ETSI document (3GPP, 2004a), where XAFE was compared with AMR (narrowband and wideband) as speech codec for RSR over UTRAN/EGPRS/GPRS channels.

5

Robustness Against Transmission Channel Errors

5.1 Introduction

In Chapter 3 we studied that digital channels can introduce several types of degradation such as fading or packet loss. We also studied different channel models that can be useful to understand and simulate the effect of different channels over the transmitted information. This information was structured in several levels, so that the channel models could be classified into two groups depending on which level is affected by the degradation physical-layer–oriented models (the degradation may affect each transmitted symbol) and higher-layer–oriented models (the degradation may affect whole data blocks or packets). The first type is mainly useful for wireless networks, while the second type is used for both wireless and IP networks by considering that a degraded block is useless (and then dropped) or that a packet is lost, respectively.

A robust RSR system must be designed so that it can maintain an acceptable performance in the presence of channel degradation. Therefore, it is necessary to develop a set of techniques to prevent, correct and mitigate its effects. In general, we will refer to them as *recovery* techniques. It may be possible to classify and study the different recovery techniques according to the specific transmission framework, that is, the chosen system architecture, the type of channel considered (physical- or higher-layer) and the network (wireless or wireline, circuit- or packet-switched) over which the system is implemented. However, the techniques usually employed are commonly shared by the different transmission frameworks, so it is more appropriate to classify them by considering exclusively the type of technique itself, and to later specify how it is applied in a given framework. Such a classification can be established as follows (Perkins *et al.*, 1998; Tan *et al.*, 2005):

1. Sender-driven techniques: These techniques are characterized by an active participation of the sender and can be classified as active or passive. The first type refers to retransmission. As we mentioned in Chapter 3, retransmission is not suitable for an application such as RSR owing to the involved delay, so it will not be considered here. On the other hand, there are a number of passive or channel coding techniques that are suitable for RSR. Some of these are as follows:

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

(a) Forward error correction (FEC): The encoder introduces redundancy information in the bitstream in order to anticipate channel degradation effects. The redundancy can be either independent (media-independent FEC) or dependent (media-specific FEC) on the information being transmitted.
(b) Interleaving: The information is reordered before transmission. Thus, channel errors (specially those grouped into bursts) are randomized.

2. Receiver-based techniques: We will also refer to these techniques as *error concealment* (EC) or mitigation techniques. In this case, the receiver has to deal with channel errors without any participation of the sender. These techniques are useful when the sender-driven techniques fail to correct the erroneous or lost data and/or when the sender is not able to participate in the recovery process. We will consider the three groups of techniques: interpolation, estimation and recognizer-based techniques.
   Estimation and interpolation are reconstruction techniques, since they provide a replacement of the damaged or lost data. These replacements are entered into the speech recognizer as if they were fully reliable. On the other hand, in the recognizer-based techniques the recognizer is informed that it is being fed with unreliable data and decides how to deal with it.

The different recovery techniques are not usually exclusive, so they can be combined to increase the robustness. In particular, the information obtained during channel decoding can be used by the EC modules to provide a better performance, as we will see in this chapter. In addition to the above sender-driven and receiver-based techniques, channel degradation can also be treated in the network, although these techniques are out of the scope of this book.

Figure 5.1 shows a block diagram of the whole RSR transmission system considered in this chapter. It includes both channel coding and EC, which are developed throughout this chapter. The first stage of the encoding process is called *source coder* and contains the compression algorithms (see Chapter 4). The information being encoded depends on the RSR architecture, that is, speech features in the case of DSR and the speech signal itself in the case of NSR. The bitstream **d** produced by the source coder is submitted to the

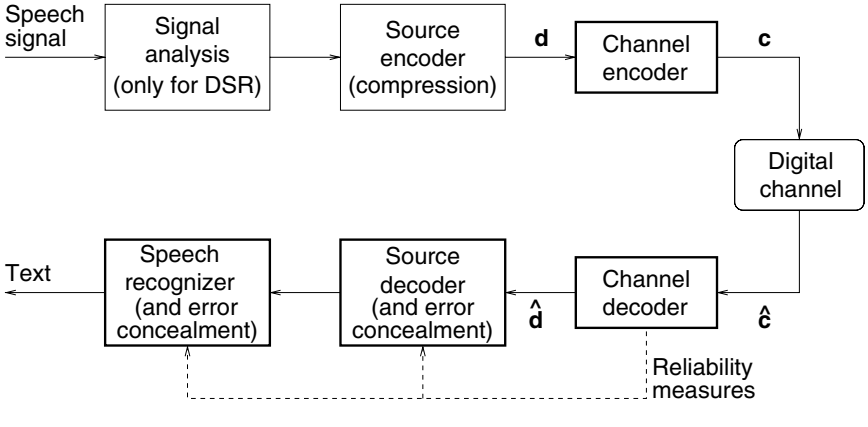


Figure 5.1 Block diagram of a whole RSR transmission system

channel coder, which introduces the error protection redundancies and generates $\mathbf{c}$. This bitstream can be degraded by the channel, so that we receive $\hat{\mathbf{c}}$. The receiver performs channel decoding and EC. This last operation can be performed by the source decoder, by the recognizer or by both. The recovery techniques developed in this chapter can operate at different data levels, ranging from bits to individual speech features, feature vectors or subvectors, which, in general, will be referred to as *speech units*.

An important issue regarding recovery techniques is the latency that we can allow. In a RSR system, an immediate response of the system is desirable, but not indispensable, so a small delay can be accepted if it favors a better recognition performance. This is an important difference with respect to other applications such as mobile telephony or VoIP, where latency can be a critical issue, which must be taken into account by the system designer.

In the previous transmission scheme, we have considered that modulators and demodulators are included in the digital channel block. We will usually assume that they perform *hard decision* so that the channel output $\hat{\mathbf{c}}$ is again a bitstream. This involves an information loss that can be avoided by performing *soft decision* instead, so that we can have the channel output represented in the signal space, which is introduced in Appendix B. Soft decision can be used to improve not only the performance of the channel decoder but also the subsequent stages of the receiver by means of reliability measures (see Figure 5.1), as we will also see in this chapter.

5.2 Channel Coding Techniques

In this section, we will see how the main types channel coding techniques, that is, FEC and interleaving, have been applied to RSR systems. This section is mainly devoted to DSR since, in this case, channel coding is developed to maximize the recognition performance, while in NSR the goal of channel coding is different. Besides, NSR is mainly based on existing speech coding standards, and their corresponding channel encoders are defined in the standards, so they are not normally accessible for the NSR system designer. Nevertheless, we include the GSM/EFR channel encoder as an example of channel coding in NSR. For those readers not familiar with channel coding concepts, it is recommended that they first study Appendix C, where the main media-independent FEC (linear block codes, cyclic codes and convolutional codes) and interleaving techniques are reviewed.

5.2.1 Error Detection

A speech unit affected by a degraded channel may be either erroneous (wireless transmission) or lost (packet-based transmission). Although both types of degradation reduce the recognition performance of an RSR system, the effect of a loss is much smaller than that of a channel error (Bernard and Alwan, 2002). A lost unit can be easily detected (just by checking the packet order number) and mitigated (e.g. by replicating the one last received) in DSR systems, so that recognition accuracy can be maintained at an acceptable level. Thus, a smart solution to the problem of dealing with erroneous units is to use media-independent FEC to detect them and to consider the affected units as if they were channel losses. When the channel decoder exclusively performs error detection, the responsibility of providing suitable data to replace the lost or erroneous data falls on the

EC algorithm, which must generate these replacements as if the source data was erased by the channel. In these cases, we will use the term *erasure channel*.

5.2.1.1 Block Codes for Error Detection

Linear block codes are an interesting option for error detection in RSR if we take into account their low delay, complexity and overhead. As shown in Appendix C, a (n, k) block encoder accepts datawords $\mathbf{d}$ of length k (in bits) and provides a codeword $\mathbf{c}$ with n bits, which is useful to detect up to $d_{min} - 1$ errors. Table 5.1 shows the performance of several linear block codes over a connected-digit recognition task as reported in Bernard and Alwan (2002). This work employs the (perceptual linear prediction) PLP/LSP-based feature vector already seen in Section 4.5.3, but without first-order prediction and interpolation to reduce sensitivity to transmission errors. Five LSP coefficients are computed every 10 ms and VQ quantized with $k = 7$ to 10 bits. The Rayleigh fading channel of Equation (3.14) is considered. The results are presented for both standard hard decision and soft decision. With hard decision, the decoding operation is equivalent to searching the codeword $\mathbf{c}_i$ that provides the smallest Hamming distance d_H between the received codeword $\hat{\mathbf{c}}$. The block code decides whether the received information is correct ($d_H = 0$) or erroneous ($d_H > 0$), but there may also be undetected errors for $d_H = 0$. With soft decision, the process is rather different since we now select the codevector $\mathbf{c}_i$ with the smallest Euclidean distance d_E between the input $\hat{\mathbf{c}}$ (for an AWGN channel and assuming a bipolar (± 1) bit representation). Then, for both types of decisions, the decoded stream can be either correct or incorrect (undetected error), but there is no error detection. However, it is easy to introduce an error detection mechanism for soft decoding. This is carried out by obtaining the factor

$$\lambda = \frac{d_E(\hat{\mathbf{c}}_{12}, \mathbf{c}_2) - d_E(\hat{\mathbf{c}}_{12}, \mathbf{c}_1)}{d_E(\mathbf{c}_1, \mathbf{c}_2)} \tag{5.1}$$

where $\mathbf{c}_1$ and $\mathbf{c}_2$ are the two closest codewords to input $\hat{\mathbf{c}}$, and $\hat{\mathbf{c}}_{12}$ is the projection of $\hat{\mathbf{c}}$ over the line joining $\mathbf{c}_1$ and $\mathbf{c}_2$, so $0 \leq \lambda \leq 1$ ($\lambda = 0$ means that both codevectors are equiprobable and $\lambda = 1$ almost ensures correct decision). An error is declared when $\lambda < \tau$, where τ is an heuristic threshold conveniently set to 0.16. This procedure is referred to as λ-*soft* decision. Finally, frames detected as erroneous are simply erased. We can see

Table 5.1 Performance evaluation of several block codes and different decision techniques over a Rayleigh fading channel (after Bernard *et al*., 2002)

Code (n, k)	d_{min}	Bitrate (kbps)	SNR (dB)	BER (%)	WAcc (%)		
					Hard	Soft	λ-Soft
(10,10)		1.0	19.96	0.25	94.71	94.71	98.32
(10,9)	2	1.0	13.87	1.00	97.31	96.35	98.12
(10,8)	2	1.0	10.69	2.00	94.47	95.03	97.82
(11,8)		1.1	8.80	3.00	87.24	95.62	97.43
(12,8)	3	1.2	6.29	5.00	67.25	93.17	97.04
(12,7)	5	1.2	4.53	7.00	40.48	91.31	95.88

in Table 5.1 that hard decision rapidly diminishes the WAcc when the error detection capability of the code is surpassed (SNR < 10 dB). Besides, an excessive number of erased frames can cause a significant WAcc reduction. Soft decision is, in general, more accurate, and λ-soft decision, in particular, provides the best results.

Another example of linear block codes for the detection of erroneous feature vectors is in the channel coding of ETSI Aurora standards. It uses 4-bit CRC parity bits to protect the 88 bits obtained from the SVQ quantization of a frame pair (44 bits for each frame). Along with the CRC check, the Aurora error detection algorithm also carries out a consistency test to prevent errors not detected by the CRC, which consists of checking the continuity of the features from two consecutive frames. The details of this error detection algorithm are given in Chapter 7 (Section 7.5.2). The frames detected as erroneous are replaced by a frame repetition concealment technique, which is studied later in this chapter. In Tan and Dalsgaard (2002), it has been shown that the use of a frame pair basis for error detection is quite damaging for the system performance, because when an error is declared the whole frame pair is ignored. The authors of this paper propose to use a 4-bit CRC for each frame to overcome this drawback in the Aurora FE standard. The bitrate is slightly increased (from 4800 to 5000 bps), but the WAcc can be raised from 47.1 to 85.6 % in a degraded channel with a BER of 2 %.

5.2.1.2 Error Detection with Convolutional Codes

As indicated in Appendix C, there are decoding algorithms for convolutional codes, such as the soft-output Viterbi algorithm (SOVA) or the Max-Log-MAP algorithms, which can provide a reliability measure $|L(k)|$ for each decoded bit $\hat{d}_k$. Therefore, a simple way of deciding if a given bit is correct or not is to impose a reliability threshold $L_T(>0)$ such that $\hat{d}_k$ is considered correct if $|L(k)| \geq L_T$ or incorrect otherwise (Potamianos and Weerackody, 2001).

5.2.1.3 Conclusion

As shown later in this chapter, the error detection strategy, implemented along with some EC technique, may result in good system performance with a slight bitrate increase in contrast to an error correction strategy, which involves a substantial bitrate increase. We will see that very simple EC techniques such as frame deletion or repetition are enough to provide acceptable results, although the use of more sophisticated concealment techniques may yield an extremely robust RSR system. Besides, we will see that under this detection-based strategy it is not necessary to "drop" the erroneous frames but they can be reused by the concealment algorithm to provide further improvements.

5.2.2 Error Correction and Unequal Error Protection

If we do not have the restriction of avoiding a meaningful bitrate increase when introducing media-independent FEC codes, then error correction is an interesting option for channel error recovery. An important issue that must be addressed is how to carry out the channel encoding of the different features or feature subvectors (we are assuming scalar or SVQ quantizers), that is, how to distribute the error protection and correction

bits among them. A first approach to solve this problem may be to consider that we have just a data stream and to protect all the bits of the stream equally (see e.g. (Hsu and Lee, 2004b)). However, we will obtain better results if we take into account our application. In this case, a straightforward solution is to adopt the same approach as for bit allocation in feature compression seen in the previous chapter: in the same way as we must allocate more bits for the most important features, more FEC must be devoted to them, and, given a certain feature, its most significant bits (MSB) must be more protected. This yields an unequal error protection (UEP) strategy for FEC assignment, which we illustrate in this section with two examples in wireless and lossy packet channels, respectively.

5.2.2.1 Example of UEP for a Wireless Channel

In Weerackody *et al*. (2002), two different UEP schemes for wireless channels have been developed. Table 5.2 summarizes the scheme that provides the best results. The source encoder has already been dealt with in Chapter 4 (Section 4.5.1) and uses scalar quantization with 6 bits for LPCC(1–6) ($c_1, \ldots, c_6$) and the energy (e) and 4 bits for LPCC(6–11) ($c_6, \ldots, c_{11}$), which results in a source rate of 6 kbps at a frame rate of 100 Hz. Bit number i of feature x is indicated as x^i ($x = e, c_1, \ldots, c_{11}$) and $i = 0$ corresponds to the MSB bit. The intended bitrate after channel coding is 9.6 kbps. In order to carry out the channel encoding, groups (multiframes) of eight speech frames (80 ms) are taken, which means that each group takes 768 bits. We can see in the table that the source bits are distributed into three different levels of error protection. Level L1 contains the $13(\times\ 8 = 104)$ more important bits, L2 the $24(\times\ 8 = 192)$ subsequent bits and L3 the $23(\times\ 8 = 184)$ least important bits. The 104 bits of L1 are protected with a convolutional code with rate 1/2 and constraint length 8, generating 208 output bits. The same code is continued with the 200 L2 bits (192 plus an 8-bit tail), generating 400 more bits. In order to accomplish this with the total of 768 bits, 24 of the 400 encoded L2 bits are punctured, resulting in 376 encoded bits. The puncturing affects the last bits of L1. This is the reason it is considered that the L1 level is divided into two different protection levels (L1_1 and L1_2). The last 184 bits are not protected. Finally, a 32×24 rectangular interleaver with column-wise writing is applied before modulation. The resulting channel coding scheme is quite robust

Table 5.2 UEP scheme for a feature vector containing LPCC(1–11) and energy (after Weerackody *et al*., 2002)

UEP level	Feature bits	Code
L1_1	$e^0, e^1, c_1^0, c_2^0, c_3^0, c_4^0, c_5^0$	Convolutional code 1/2
L1_2	$e^2, c_1^1, c_2^1, c_3^1, c_4^1, c_5^1$	Convolutional code 1/2
L2	$e^3, e^4, c_1^2, c_1^3, c_2^2, c_2^3, \ldots,$ $c_6^0, c_6^1, c_7^0, c_7^1, \ldots, c_{11}^0, c_{11}^1$	Convolutional code 1/2 with puncturing
L3	$e^5, c_1^4, c_1^5, c_2^4, \ldots, c_5^4, c_5^5,$ $c_6^2, c_6^3, c_7^2, c_7^3, \ldots, c_{11}^2, c_{11}^3$	No code

against random (AWGN channel) and bursty errors (Rayleigh channel), as already shown in Table 3.2.

5.2.2.2 Example of UEP for a Lossy Packet Channel

The next arrangement, proposed in Boulis *et al.* (2002), has been designed to protect a DSR system against packet losses. In this case, the feature vector contains MFCC(1-8) and energy, which are distributed into $N_{sub} = 5$ subvectors (pairs of adjacent MFCCs and energy alone). For subvector quantization, fixed-length TSVQ is employed using N_{bits} bits (6, 6, 4, 6 and 4 bits, respectively). This source encoding (already described in Section 4.5.2) has a bitrate of 2.6 kbps. As in the previous subsection, the speech frames are grouped into multiframes of size L for their transmission. Figure 5.2 shows an implementation example of the proposed packetization and UEP scheme for a given multiframe. It uses (N, k) Reed–Solomon (RS) codes, which are a special case of nonbinary block codes. Each codeword (RS stream) contains a total of N symbols ($N = 12$ in the figure), in which k are source symbols (notated as $\mathbf{x}_i^j$ in the table) and $N - k$ are FEC symbols (notated as F in the figure). The notation $\mathbf{x}_i^j$ for the source symbols indicates a bit vector of length L formed with the jth ($j = 1, 2, \ldots, N_{bits}$) bit from L consecutive ith feature subvectors ($i = 1, 2, \ldots, N_{sub}$). The source bits are distributed according to their importance, from top to bottom, so that more FEC is assigned to the most important bits. The importance of each bit must be predetermined during bit allocation (Section 4.5.2). An RS code optimized for channel erasures guarantees that if any k symbols (source or FEC symbols) of the RS stream are received then it is possible to recover the original k source symbols (Rizzo, 1997). Symbols belonging to a given RS stream are transmitted in consecutive packets. Each packet contains symbols from S RS streams ($S = 4$ in the figure). The whole content of a RS stream can be recovered whenever the number of lost packets of a multiframe is $N - k$ or less. In the example shown in the figure, the fourth RS stream loses information whenever a packet loss appears. On the contrary, we are guaranteeing that we can decode the MSB bits of the most important subvectors (first RS stream) even with a high loss rate. Therefore, this encoding scheme leads to a graceful degradation in the presence of packet losses since it allows intermediate feature vector reconstruction from the MSB bits due to the use of TSVQ quantization, which provides embedded encoding of the source.

Given the code length N and the symbol size (in bits), which, in this case, coincides with the number L of frames in the multiframe ensemble, and once the source bits have been ordered during bit allocation according to their importance for recognition, the remaining

RS stream	Packets (1–12)											
1	$\mathbf{x}_1^1$	$\mathbf{x}_2^1$	$\mathbf{x}_3^1$	$\mathbf{x}_4^1$	F	F	F	F	F	F	F	F
2	$\mathbf{x}_5^1$	$\mathbf{x}_1^2$	$\mathbf{x}_1^3$	$\mathbf{x}_3^2$	F	F	F	F	F	F	F	F
3	$\mathbf{x}_5^2$	$\mathbf{x}_1^4$	$\mathbf{x}_2^2$	$\mathbf{x}_2^3$	$\mathbf{x}_3^3$	$\mathbf{x}_4^2$	F	F	F	F	F	F
4	$\mathbf{x}_4^3$	$\mathbf{x}_5^3$	$\mathbf{x}_4^4$	$\mathbf{x}_1^5$	$\mathbf{x}_1^6$	$\mathbf{x}_4^5$	$\mathbf{x}_3^4$	$\mathbf{x}_4^6$	$\mathbf{x}_2^4$	$\mathbf{x}_2^5$	$\mathbf{x}_2^6$	$\mathbf{x}_5^4$

Figure 5.2 Example of a multiframe ensemble with UEP channel coding using RS codes (after Boulis *et al.*, 2002)

problem is to allocate them to the RS streams, that is, we have to find the dataword lengths (in number of symbols) $(k_1, k_2, \ldots, k_S)$. This is equivalent to obtaining the best partition of the ordered bit list. The optimal solution would be to carry out recognition tests in order to obtain the WER for each possible FEC assignment. However, this solution will lead us to a prohibitive amount of computation. An approximated solution, proposed in Boulis *et al*. (2002), can be implemented by using the WER table obtained during the pruning procedure for bit allocation. In fact, packet loss produces bit erasures in the same way as in bit allocation by pruning. Then, for each possible FEC assignment, the expected WER is computed as

$$\overline{WER} = \frac{\sum_{k=1}^{N} p(k)\epsilon(k)}{\sum_{k=1}^{N} p(k)} \tag{5.2}$$

where k is the number of lost packets (in the ensemble of N packets), $p(k)$ is the probability associated to that loss (selected according to a Poisson loss model), and $\epsilon(k)$ is the WER corresponding to the erased (lost) bits. It must be taken into account that, when RS codes are applied, the erased bits are exclusively determined by the number of lost packets, and not by the particular combination of these lost packets. The optimal FEC assignment is the one that provides minimum $\overline{WER}$. Of course, this approach is feasible only if the number of possible FEC assignments is not very large.

5.2.3 Example of Channel Coding for NSR: GSM-EFR

We have seen that channel coding can be used for either error detection, which is complemented later with EC, or for error correction. The GSM/EFR channel encoder follows both the strategies. At the bit level, it introduces error correction coding with UEP. However, when the channel degradation exceeds the error correction capacity of the encoding scheme for a given received frame, the channel decoder also provides a bad frame indicator (BFI) that indicates to the subsequent stages of the receiver that frame is useless and must be concealed.

The GSM/EFR channel encoder is based on that of the GSM/FR encoder, so we detail the latter one first. The FR speech encoder provides 260 source bits (bitrate 13 kbps), which are split into class 1 (182 protected bits) and class 2 (78 unprotected bits) for UEP. Class 1 is further divided into class 1a (50 most important bits) and class 1b (remaining 132 bits). Class 1 bits are encoded as follows:

1. Class 1a bits are encoded with a (53,50) cyclic code for error detection.
2. Class 1 convolutional encoding: Class 1a bits, plus their 3 parity bits, plus class 1b bits, plus 4 zero-padding bits (189 bits in total) are encoded with a convolutional code of rate 1/2 and constraint length $m = 5$, obtaining 378 encoded bits.

The resulting 456 bits $(378 + 78)$ are finally interleaved. The final bitrate, after channel coding, is 22.8 kbps (456 bits $\times$ 50 Hz). Since the FR generates 260 bits per frame, the

EFR encoder must first increase its 244 source bits up to 260, so that the FR channel encoding can be applied. In order to do this, the 65 most relevant class 1 bits (class 1a bits + 15 from class 1b) are protected with a (73,65) error-detection CRC code, which introduces 8 extra bits (added to class 1b). The remaining 8 bits are obtained by replicating twice the first 4 unprotected bits (corresponding to 4 subframes) from pulse 5, and are added to the unprotected bit set. The details of the GSM/EFR channel encoding scheme can be seen in ETSI (1998a).

5.2.4 Frame-Level Interleaving

We learnt in Chapter 3 (Tables 3.2 and 3.3) that error bursts are more damaging than random errors. The reason is that error correction and concealment techniques can be quite effective when the error bursts are short (more random) but are not so effective for long bursts. We can see in Appendix C and in the previous examples that interleaving is a technique commonly applied at the bit level to randomize the appearance of errors, thus reducing the effect of error bursts due to fading in wireless transmission.

Interleaving can also be applied when we consider a higher-layer transmission scheme, such as IP, where the transmission units are data blocks or packets. In particular, a useful approach for RSR is to interleave speech frames, although it can also be applied to subframes (Delaney, 2005). Figure 5.3 shows an example of a transmission scheme that includes a block interleaver. The 16 speech frames of this example are interleaved and distributed into two-frame packets. The details of this interleaver are given in Appendix C, Section C.2, and Figure C.4. A packet loss burst affects packets #3 to #6. Without interleaving, this loss burst would affect up to 8 consecutive speech frames. However, with interleaving, the loss bursts after deinterleaving are no longer than 2 frames. We can now recall the recognition results for a two-state Markov chain of Table 3.3. In Figure 5.3, the loss rate is 50 %. This means that we would be moving in Table 3.3 from $d_{loss} = 4$ (WAcc = 84.56 %) without interleaving to $d_{loss} = 1$ (WAcc = 98.90 %) with interleaving. This improvement is due to the fact that EC is much more effective for short loss bursts.

The main drawback of interleaving is the latency involved. However, as mentioned in the introduction section of this chapter, this is not so critical in RSR as in VoIP, for example. Besides, interleaving does not require any increment of the transmission bandwidth. Frame-level interleaving is specially attractive for IP networks, where each

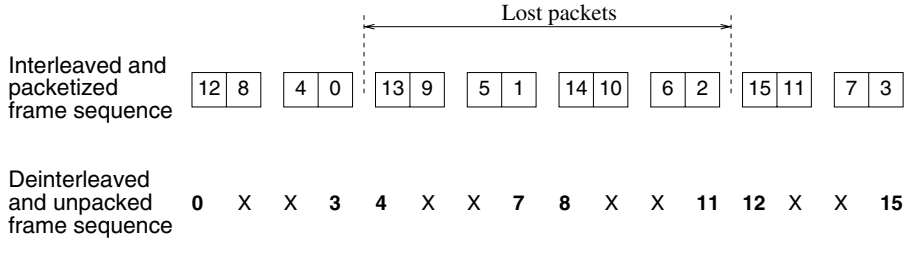


Figure 5.3 Example of a 4 × 4 block interleaving

packet usually contains one or more speech frames. It can be applied to both NSR (to the speech codec frames) and DSR (to the feature vectors representing frames).

A complete study of the effect of different interleavers on a DSR system has been carried out by James and Milner (2004). The authors compare block and convolutional interleavers (developed in Appendix C) with an EC algorithm based on a polynomial interpolation for the remaining loss bursts. The conclusion of this work is that both methods yield similar performances, although those of the block interleaver are slightly better. This work also proposes a method to determine a good block interleaver: if a block interleaver of degree d is defined by a permutation $\pi(i)$, then a good measure of its burst breaking capability is the following decorrelation measure:

$$D = \sum_{i=0}^{d^2-1} \sum_{j=0}^{d^2-1} \frac{|\pi(i) - \pi(j)|}{|i - j|} \tag{5.3}$$

Thus, the problem is to find a block interleaver that maximizes D, which is carried out by a greedy local search.

5.2.5 Media-specific FEC

An alternative to prevent channel errors or losses is to transmit information about every speech unit more than once. Typically, in addition to the main or primary encoding, there is a secondary encoding with a smaller bitrate that generates redundant FEC data useful to replace damaged or lost primary data. This is known as *media-specific FEC*. Its utility is clear in packet-based transmission: we can place in each packet the primary data corresponding to the current speech units plus secondary (FEC) data corresponding to previous and/or subsequent units. This strategy is depicted in Figure 5.4. Each packet k contains information about 4 frames placed at times $2k$ and $2k + 1$ (primary data) and $2k - T_{fec}$ and $2k + 1 + T_{fec}$ (secondary data) ($T_{fec} = 4$ in the figure). In the example of the figure, packets 5 to 9 are lost, which implies that primary frames 10–19 are lost. However, this FEC scheme allows the recovery of frames 11, 13, 16 and 18 from the secondary stream. Therefore, the effect of the media-specific FEC is similar to interleaving in the sense that it breaks long loss bursts into shorter ones (in fact, the secondary frames are interleaved).

As shown in Figure 5.4 and in the same way as for interleaving, media-specific FEC does not allow the recovery of all lost frames. Again, the EC algorithm applied after channel decoding must deal with these frames as if they are inevitably lost. Additionally, it must be taken into account that the secondary-encoded frames are usually degraded by

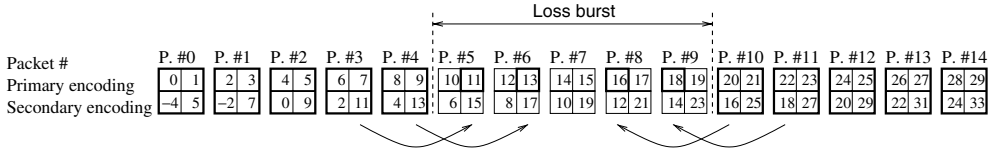


Figure 5.4 Example of a media-specific FEC. The secondary frames are delayed $T_{fec} = 4$ frames with respect to the nearest primary frames

a severe compression, which could also be managed by means of EC. The utility of a media-specific FEC similar to that of Figure 5.4 using 2 or 3 bits per feature vector for the secondary encoding and combined with EC based on MMSE estimation (developed in following section) has been shown in Gomez *et al*. (In Press) and Peinado *et al*. (2005a).

5.3 Error Concealment (EC)

There are several reasons EC is suitable and important in RSR systems:

- While the earlier channel coding techniques require the participation of the sender (client), EC is fully implemented at the receiver (server). Let us remember that a RSR has a client/server architecture in order to implement low-complexity clients and powerful servers that can carry out complex recognition tasks. Therefore, it seems logical to benefit from these powerful servers by introducing EC techniques whenever their computational cost is reasonably lower than that required for the recognition itself.
- EC is quite suitable for an application such as RSR since the speech signal is quite redundant. This redundancy allows the implementation of efficient EC techniques.
- The channel coding techniques of the previous section are usually required to be combined with some type of EC technique. Thus, EC is mandatory in the cases of error-detection FEC, interleaving or media-specific FEC and recommendable for error correction to mitigate the effect of the remaining errors.
- In packet networks, EC reduces the sensitivity of the recognition performance with respect to the packet size (in number of speech frames) (Mayorga *et al*., 2002).

In the following subsections, we study three different types of EC techniques that have been used for RSR: interpolation, estimation and recognizer-based techniques. Estimation and interpolation are reconstruction techniques, since they provide a replacement of the damaged or lost data, so we can consider them as part of the source decoding module of Figure 5.1. The replacements can enter the speech recognizer as if they were fully reliable. However, in the recognizer-based techniques, the recognizer also participates in the EC task, since it is informed by the decoder that it is being fed with unreliable data and hence it can treat those data in a different way. In some references (Perkins *et al*., 1998), another class of EC techniques under the generic name of *insertion-based techniques* (frame dropping, repetition, etc.) are considered. These techniques can be considered as particular cases of the three types previously mentioned, as we see later.

Similarly to the previous section, we are mainly concerned with EC for DSR, since EC algorithms for NSR are designed with a goal different from speech recognition, with the exception of B-NSR, which is considered here as a type of DSR for EC purposes. In the case of DSR, we should take into account that only the static features are usually encoded and transmitted, while the dynamic features are computed at the receiver from the static ones. Then, although EC is applied to the received parameters, that is, the static features, it must be taken into account that the dynamic features are also affected by the EC operation. In fact, in Milner and James (2003) it is shown how error bursts can be more damaging for dynamic features than for static ones.

Although EC techniques can use bit-level information provided by the channel decoder, basically they deal with speech units such as individual features, feature subvectors or

feature vectors, which, in general, are referred to as *vectors* in the following text. We will use the vector notation $\mathbf{x}_t$ for the vector transmitted at time t. We will also consider that error bursts start at time $t = 1$ and end at $t = T$. Therefore, $\mathbf{x}_0$ and $\mathbf{x}_{T+1}$ are the last and first correctly received vectors before and after the burst, respectively. The replacements provided by the EC algorithm are notated as $\hat{\mathbf{x}}_t$.

5.3.1 Interpolation

We studied in the previous section that a channel decoder can be exclusively implemented for error detection, so that, in the case of a channel error, it only informs that a channel erasure has occurred and does not provide any source data to the subsequent stages of the receiver. As a consequence, we will be missing an individual feature, a feature subvector or a whole feature vector (depending on the encoding scheme). In this case, interpolation seems a suitable and simple approach to provide the needed replacements for the missing parameters, since it only requires knowledge of some features correctly received (before and after the error interval) and of the selected interpolation function.

In general, the interpolate of a feature vector or subvector at time t is obtained as

$$\hat{\mathbf{x}}_t = F\left(t; \mathbf{x}_{-\mathcal{M}+1}, \mathbf{x}_{-\mathcal{M}+2}, \ldots, \mathbf{x}_0, \mathbf{x}_{T+1}, \mathbf{x}_{T+2}, \ldots, \mathbf{x}_{T+\mathcal{N}}\right) \quad (1 \leq t \leq T) \tag{5.4}$$

where F is the interpolation function and $\mathcal{M}$ and $\mathcal{N}$ are the maximum number of frames correctly received before and after the error burst, respectively, which are considered to have carried out the interpolation. When we say "maximum," we are taking into account the possibility that there are no $\mathcal{M}$ or $\mathcal{N}$ frames before or after the burst due to other (previous or subsequent) errors. Note that, when an erasure burst of length T is detected, a decoding delay of $T + \mathcal{N}$ is required when $\mathcal{N} > 0$.

A very common choice for function F is a polynomial. The minimum degree required for the polynomial is $\mathcal{M} + \mathcal{N} - 1$. The simplest choice is a zero-order interpolation with $\mathcal{M} = 1$ and $\mathcal{N} = 0$, so that $\hat{\mathbf{x}}_t = \mathbf{x}_0$. This is a forward *repetition* of the last correctly received vector. Backward repetition ($\mathcal{M} = 0$, $\mathcal{N} = 1$, $\hat{\mathbf{x}}_t = \mathbf{x}_{T+1}$) could also be chosen, although the former choice is preferred since it does not introduce any delay. *Linear interpolation* (Milner and Semnani, 2000) has been proposed for RSR in different contexts. It corresponds to the case $\mathcal{M} = \mathcal{N} = 1$, which yields the following interpolation formula:

$$\hat{\mathbf{x}}_t = \mathbf{x}_0 + \frac{t}{T+1}(\mathbf{x}_{T+1} - \mathbf{x}_0) \tag{5.5}$$

Its implementation requires the introduction of a decoding delay equal to the burst length. The performance of linear interpolation over an erasure channel is given in Table 5.3. The recognition system is exactly the same as the one used in the experiment of Table 3.3 for the Aurora EC algorithm.

An enhanced polynomial interpolation is used in James and Milner (2004), where *cubic Hermite polynomials* ($\mathcal{M} + \mathcal{N} - 1 = 3$) are proposed as interpolation functions. This is a nonlinear interpolation that can be expressed as

$$\hat{\mathbf{x}}_t = \mathbf{a}_0 + \bar{t}\mathbf{a}_1 + \bar{t}^2\mathbf{a}_2 + \bar{t}^3\mathbf{a}_3 \tag{5.6}$$

where $\bar{t} = t/(T+1)$ and coefficients $\mathbf{a}_i$ $(i = 0, 1, 2, 3)$ are the multivariate coefficients of the polynomial. This interpolation ensures a smooth trajectory by forcing the continuity of its first derivative at the beginning and end of the burst. After obtaining the coefficients, Equation (5.6) can be expressed as

$$\hat{\mathbf{x}}_t = \mathbf{x}_0(\bar{t} - 3\bar{t}^2 + 2\bar{t}^3) + \mathbf{x}_{T+1}(3\bar{t}^2 - 2\bar{t}^3) + \mathbf{x}'_0(\bar{t} - 2\bar{t}^2 + \bar{t}^3) + \mathbf{x}'_{T+1}(\bar{t}^3 - \bar{t}^2) \quad (5.7)$$

Although the derivatives could be approximated by $\mathbf{x}'_0 = T(\mathbf{x}_0 - \mathbf{x}_{-1})$ and $\mathbf{x}'_{T+1} = T(\mathbf{x}_{t+2} - \mathbf{x}_{T+1})$, in practice, rapid fluctuations of the feature values result in bad interpolates, so it is better to assume $\mathbf{x}'_0 = \mathbf{x}'_{T+1} = 0$.

Nonlinear interpolation can also be obtained by a joint forward and backward repetition. In this case, the interpolate can be obtained as

$$\hat{\mathbf{x}}_t = \begin{cases} \mathbf{x}_0 & \text{if } t \leq \lceil T/2 \rceil \\ \mathbf{x}_{T+1} & \text{if } t > \lceil T/2 \rceil \end{cases} \quad (5.8)$$

This is the case of the ETSI Aurora standard. Since the associated error detection procedure has a frame pair basis, the Aurora EC algorithm can be summarized as follows: once an error burst, containing $T = 2 \times B$ frames, is detected, the first B frames are substituted by the last correct frame before the burst ($\mathbf{x}_0$) and the last B ones by the first correct frame after the burst ($\mathbf{x}_{T+1}$). More details of the Aurora EC algorithm are given in Chapter 7.

The performance of the Aurora EC algorithm over several channel conditions was shown in Table 3.3 (Aurora-2 task, clean sentences). Since linear interpolation and the Aurora algorithm have been extensively proposed for RSR EC, it is interesting to compare both methods from the results of Tables 5.3 and 3.3. The results are clearly in favor of the Aurora algorithm. This behavior has also been found in Peinado *et al.* (2003) and Tan *et al.* (2004b). Figure 5.5 illustrates how linear interpolation and the Aurora EC algorithm generate interpolates over the trajectory of MFCC(1). Linear interpolation provides a smoother transition from $\mathbf{x}_0$ to $\mathbf{x}_{T+1}$ than the Aurora algorithm. Besides, it can be proved that linear interpolation provides a smaller MSE between the original and interpolated sequence than the Aurora algorithm. However, the original sequence contains

Table 5.3 Word accuracies obtained with the ETSI DSR/FE standard with linear interpolation EC over a two-state Markov chain erasure channel (Aurora-2 task and testing only with clean sentences)

ulp (%)	d_{loss} (in number of lost packets)				
	1	2	4	8	16
10	99.00	98.59	96.31	93.02	90.57
20	98.97	97.93	93.60	86.73	82.47
30	98.95	97.36	90.16	80.40	74.13
40	98.95	96.71	86.99	74.89	65.49
50	98.88	95.92	82.22	67.89	57.43

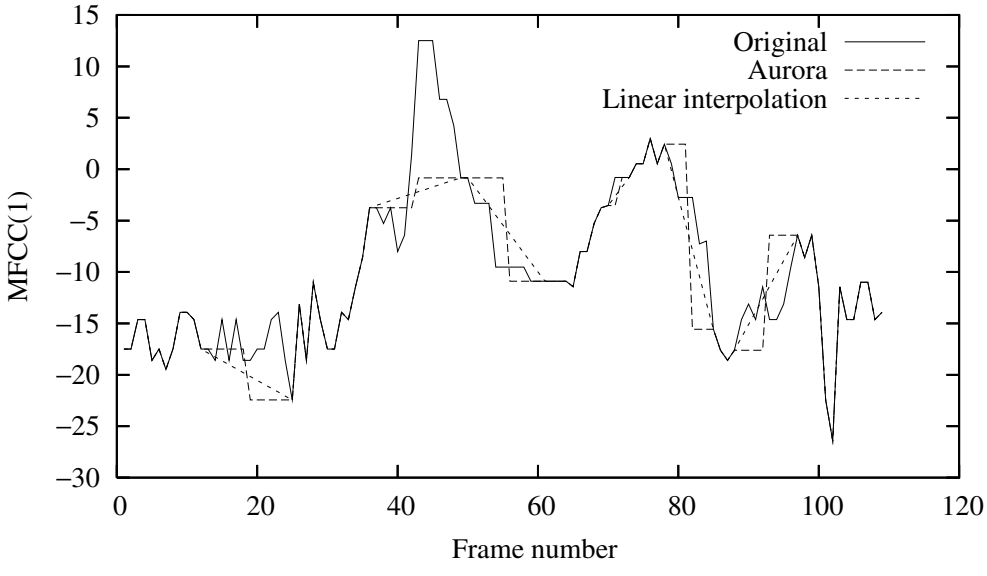


Figure 5.5 Trajectories of feature MFCC(1) transmitted over an erasure channel and mitigated with linear interpolation and the Aurora algorithm

rapid variations that do not justify the use of a smoothing technique such as linear interpolation. This problem of the linear interpolation technique can be better understood in terms of the alignment carried out by the VA for recognition: linear interpolation causes frequent transitions out of the current state. This effect has been proven in Tan *et al*. (2004b), where it is shown than linear interpolation yields an average state duration smaller than that obtained with the original features, while the Aurora algorithm approximates it better. The performance of this algorithm over the GSM EP error patterns is shown in Tables 3.1 and 5.4.

The Aurora EC algorithm operates at the frame level over frame pairs, since this is the speech unit managed by the Aurora error detection procedure (CRC check plus consistency test). According to recommendation RFC-3557 (Xie, 2003), this scheme is suitable for a packet-based transmission, since the smallest recommended payload is a frame pair, so this is the smallest unit that may be lost during transmission. However, in the case of a wireless transmission, the frame pair data, although corrupted, is available, and it may be expected that part of this data is error free. Therefore, we can argue that it is not necessary to drop the whole frame pair. In order to do this, we need an error detection procedure working over speech units smaller than frame pairs. This can be easily done for the Aurora standard by considering the pairs of consecutive SVQ subvectors contained in the frame pair as speech units (Tan *et al*., 2004a). Thus, an SVQ pair is declared as erroneous if it does not pass the partial consistency test of Equation (7.55), and it is concealed by applying the forward–backward repetition algorithm of Equation (5.8) by using the nearest correct SVQ vectors. The effectiveness of this approach, in comparison with the baseline Aurora EC algorithm, is shown in Table 5.4 (experiment Aurora-sub).

The forward–backward repetition interpolation of Equation (5.8) has also been modified for its application in a bluetooth network that generates very long bursts (18–24 frames) (Bawab *et al.*, 2003) by zeroing the interpolates $\hat{\mathbf{x}}_t$ in the middle of the burst by means of the following weighting factors:

$$w_t = \frac{1}{2}\left[1 + \cos\left(2\pi \frac{t-1}{T-1}\right)\right] \tag{5.9}$$

Extrapolation can be considered as a particular case of Equation (5.4) when $\mathcal{N} = 0$. Thus, the forward repetition technique can also be considered a type of extrapolation. Extrapolation based on repetition can be a quite simplistic approach, especially in the case of long error bursts. In Kim and Cox (2001a), it is proposed the extrapolation method used by the IS-641 speech codec is applied in a B-NSR system. This method is described by the following equation:

$$\hat{\mathbf{x}}_t = c\hat{\mathbf{x}}_{t-1} + (1-c)\mathbf{x}_{ave} \tag{5.10}$$

where $\mathbf{x}_{ave}$ is the average parameter vector, c is a forgetting factor (equal to 0.9 in the given reference) and $\hat{\mathbf{x}}_0 = \mathbf{x}_0$ is used for initialization.

5.3.2 Estimation

Similar to interpolation, the goal of the estimation techniques is to provide replacements for those parameters that have been affected by channel degradation. However, instead of using parametric functions, estimation techniques explicitly employ some statistical speech model in order to generate such replacements (or estimates). In fact, when we interpolate with a given function, we are implicitly using a model, although, unlike estimation, this model has not been trained from speech data, and its selection is based on considerations such as the feature trajectory shape, as discussed in the previous section. We will see in this section how different EC techniques can be derived by using different estimation methods (MMSE, MAP, etc.), speech models, and by considering different received data formats.

Speech models provide a probabilistic framework to compute estimates from *available data*, that is, data received just before, during and just after an error burst. In the case of a wireless channel, the received data are correct before and after the burst ($t < 1$ and $t > T$) and erroneous during the burst ($1 \leq t \leq T$). However, we may expect that these during-burst data are partially correct and, therefore, still useful for estimation. For IP channels, the data is also correct before and after the burst, but there is no data available from the channel during a loss burst, and, in most cases, the vectors of the interval $[1, T]$ are completely lost. However, we must take into account that under certain transmission schemes using FEC (Boulis *et al.*, 2002; Peinado *et al.*, 2005a) the data corresponding to a given feature vector (or subvector) can be distributed among several packets, so that the effect of a loss burst is either that the vector is definitively lost or that we still have a version of the vector degraded by a coarser quantization. In the last case, the situation is similar to a wireless transmission in the sense that there is available data in the interval $[1, T]$. In general, we will use the notation $\mathbf{y}_t$ for the decoded vectors (degraded or correct).

5.3.2.1 MMSE Estimation Fundamentals

MMSE estimation involves an expected value computation of the received vector at time t:

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|\text{available data, source model}] \qquad (1 \leq t \leq T) \tag{5.11}$$

In order to compute this expected value, we will consider that the transmitted feature vector $\mathbf{x}_t$ (corresponding to frame number t) belongs to a finite set $\{\mathbf{x}^{(i)}; i = 0, \ldots, N-1\}$ due to quantization. The effect of the transmission channel is that we receive a vector $\mathbf{y}_t$ that can differ from $\mathbf{x}_t$. Therefore, the MMSE estimate of the vector at time t can be obtained as

$$\hat{\mathbf{x}}_t = \sum_{i=0}^{N-1} \mathbf{x}^{(i)} P(\mathbf{x}_t^{(i)}|Y) \qquad (1 \leq t \leq T) \tag{5.12}$$

where we have expressed $\mathbf{x}_t = \mathbf{x}^{(i)}$ as $\mathbf{x}_t^{(i)}$ (indicating that vector i was transmitted at time t) for notational simplicity. Y represents the available data and probabilities $P(\mathbf{x}^i|Y)$ are determined by the speech model employed. In the most general case, $Y = (Y^-, Y_1^T, Y^+)$, where $Y^- = (\mathbf{y}_{-\mathcal{M}+1}, \ldots, \mathbf{y}_0)$ represents the vectors correctly received ($\mathbf{y}_t = \mathbf{x}_t$) before the error burst, $Y_1^T = (\mathbf{y}_1, \ldots, \mathbf{y}_T)$ are the degraded vectors received during the burst ($\mathbf{y}_t \neq \mathbf{x}_t$, in general), and $Y^+ = (\mathbf{y}_{T+1}, \ldots, \mathbf{y}_{T+\mathcal{N}})$ are the vectors correctly received ($\mathbf{y}_t = \mathbf{x}_t$) after the error burst. As mentioned earlier, the vector sequence Y_1^T may be fully or partially (only some vectors) available, or even completely lost. Similar to interpolation, MMSE estimation introduces a decoding delay of $T + \mathcal{N}$ if $\mathcal{N} > 0$.

A simple example of MMSE estimation can be obtained by considering $Y = \mathbf{y}_t$ ($\mathcal{M} = \mathcal{N} = 0$) (Skoglund and Hedelin, 1994):

$$\hat{\mathbf{x}}_t = \sum_{i=0}^{N-1} \mathbf{x}^{(i)} P(\mathbf{x}^{(i)}|\mathbf{y}_t) \tag{5.13}$$

$$P(\mathbf{x}^{(i)}|\mathbf{y}_t) = \frac{P(\mathbf{y}_t|\mathbf{x}^{(i)})P_i}{P(\mathbf{y}_t)} \tag{5.14}$$

where the source has been simply modeled by the *a priori* probabilities $P_i = P(\mathbf{x}^{(i)})$ of its symbols. We will refer to Equation (5.13) as *raw-MMSE* estimation.

The above MMSE estimation does not benefit from the temporal redundancies that are usually contained in speech parametrization. In order to account for these redundancies, it is necessary to use a richer source model. This can be achieved by introducing the HMM models studied in Chapter 2. A HMM model is described by transition probabilities $a_{ij} = P(\mathbf{x}_t^{(j)}|\mathbf{x}_{t-1}^{(i)})$ and observation probabilities $b_i(\mathbf{y}_t) = P(\mathbf{y}_t|\mathbf{x}^{(i)})$. The transition probabilities can be obtained from the training (speech) database, while for the observation probabilities we must specify the method for their computation which depends on the transmission scheme as it is shown in the next subsections. Now we can obtain the MMSE estimate of the received parameter vector at time t, given the available data $Y = (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{T+1})$ ($\mathcal{M} = \mathcal{N} = 1$), as (Peinado *et al.*, 2003)

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{T+1}] = \sum_{i=0}^{N-1} \mathbf{x}^{(i)} \gamma_t(i) \quad (1 \leq t \leq T) \tag{5.15}$$

with

$$\gamma_t(i) = P(\mathbf{x}_t^{(i)}|\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_{T+1}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=0}^{N-1} \alpha_t(j)\beta_t(j)} \tag{5.16}$$

$$\alpha_t(i) = P(\mathbf{x}_t^{(i)}|\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_t) \tag{5.17}$$

$$\beta_t(i) = P(\mathbf{y}_{t+1}, \ldots, \mathbf{y}_{T+1}|\mathbf{x}_t^{(i)}) \tag{5.18}$$

where $\alpha_t(i)$ and $\beta_t(i)$ are the forward and backward conditional probabilities, respectively, which can be computed by means of the forward–backward procedure already described in Section 2.6.3. We will refer to the resulting technique as forward–backward MMSE (FBMMSE) estimation.

It must be observed that, when an error burst is detected, FBMMSE forces the introduction of a decoding delay of $T + 1$ vectors. This delay can be avoided by substituting probabilities $\gamma_t(i)$ by $\alpha_t(i)$ in Equation (5.15). In this case $\mathcal{M} = 1$, $\mathcal{N} = 0$ and $Y = (\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_t)$, so no delay is required. This estimation method has been proposed in Fingscheidt and Vary (2001) and Peinado *et al.* (2003) and we will refer to it as forward MMSE (FMMSE) estimation. FMMSE involves half the computational burden of FBMMSE. Other efficient MMSE variants and an analysis of their computational complexity are studied in Peinado *et al.* (2005b).

The remaining problem for the application of the different MMSE-based techniques is how to compute the observation probabilities $b_i(\mathbf{y}_t)$. This is treated in the following subsections.

5.3.2.2 MMSE Estimation for Wireless Channels

In wireless transmission, we have to consider the correctness of the received vectors. At times $t = 0$ and $t = T + 1$, the received vectors are fully reliable. Therefore, the corresponding observation probabilities must be fixed as

$$b_i(\mathbf{y}_0), b_i(\mathbf{y}_{T+1}) = \begin{cases} 0 & \mathbf{x}^{(i)} \neq \mathbf{y}_0,\ \mathbf{c}^{(i)} \neq \mathbf{y}_{T+1} \\ 1 & \mathbf{x}^{(i)} = \mathbf{y}_0,\ \mathbf{x}^{(i)} = \mathbf{y}_{T+1} \end{cases} \tag{5.19}$$

A vector $\mathbf{y}$ received during an error burst is not fully reliable, and its observation probability depends on the level of degradation. This reliability information must be provided by the channel decoder. We will first consider a soft-decision decoder. The dataword $\mathbf{d}_y = (d_y(0), d_y(1), \ldots, d_y(M-1))$, corresponding to vector $\mathbf{y}$, is provided by the channel decoder along with a set of reliability measures $l_y(k)$ $(k = 0, 1, \ldots, M-1)$. We will notate pairs $(d_y(k), l_y(k))$ as $L_y(k)$ (usually called soft bits). Then, we can compute (assuming bit equiprobability and a memoryless channel)

$$b_i(\mathbf{y}) = C \prod_{k=0}^{M-1} P(d_x^{(i)}(k)|L_y(k)) \tag{5.20}$$

where C is a normalization constant, $M = \log_2 N$ is the dataword length (number of bits of the VQ codebook), $d_x^{(i)}(k)$ is the kth bit of the dataword $\mathbf{d}_x^{(i)}$ corresponding to vector

$\mathbf{x}^{(i)}$. Probabilities $P(d_x^{(i)}(k)|L_y(k))$ can be obtained as

$$P(d_x^{(i)}(k)|L_y(k)) = \begin{cases} 1 - p_e(k) & d_x^{(i)}(k) = d_y(k) \\ p_e(k) & d_x^{(i)}(k) \neq d_y(k) \end{cases} \tag{5.21}$$

where $p_e(k)$ is the instantaneous bit error probability. Equations (3.9) and (C.15) show two examples of how $p_e(k)$ can be computed from the reliability measures $l_y(k)$.

However, standard hard-decision decoders do not provide soft bits. Soft decision can be avoided if we use the average error probability p_e of the channel during error bursts instead of instantaneous values in Equation (5.21). Then, bit reliabilities are not necessary, and Equation (5.20) becomes

$$b_i(\mathbf{y}) \approx C(1 - p_e)^{M - d_{ham}} p_e^{d_{ham}} \tag{5.22}$$

where d_{ham} is the Hamming distance between datawords $\mathbf{d}_x^{(i)}$ and $\mathbf{d}_y$. Equation (5.22) still requires an estimation of the channel SNR (necessary to obtain p_e) during error bursts (see Section 3.2.3 for details). In Peinado *et al*. (2003), it has been shown that this hard-decision scheme does not involve any loss of accuracy with respect to soft decision and, furthermore, that the SNR estimation is unnecessary by assuming a small fixed SNR value during error bursts. This conclusion is very important since it involves that MMSE estimation can be applied to RSR without modifying the simple hard-decision schemes commonly applied (as in the ETSI DSR standards).

In Table 5.4, we compare the performances obtained with the hard-decision implementation of the different MMSE estimations (FBMMSE, FMMSE, raw-MMSE) described above, considering a fixed channel SNR of -2 dB during error bursts. The tested system has a DSR architecture and uses the ETSI FE standard. The resulting MMSE-based EC algorithms are applied to every SVQ subvector defined in the standard. The recognition task is Aurora-2 (only clean sentences) and the EPx (x = 1, 2, 3) error patterns have been used to simulate a GSM transmission. The results obtained with the Aurora EC algorithm and linear interpolation are also shown. We observe that the MMSE techniques, except raw-MMSE, are quite effective. The reason for this drastic improvement lies in the combination of two independent types of information. First, we are applying a powerful speech model. Second, unlike interpolation techniques, FBMMSE and FMMSE do not drop the data received from the channel, which, although contains some errors, is still usable.

Another solution for obtaining p_e without the need for soft decision consists of estimating the number of bit errors N_e occurring during an interval of duration N (bits), so that $p_e = N_e/N$. The resulting estimate is valid during the considered interval. This solution has been successfully applied in Ion and Haeb-Umbach (2005a) to the ETSI advanced front-end (AFE) standard in two different ways. First, the considered interval is a whole frame pair, and N_e is increased by one each time the consistency test of a feature pair fails. In order to obtain an estimate of p_e that is more localized in time, a smaller interval such as a feature pair can also be considered. However, the experimental results obtained in Ion and Haeb-Umbach (2005a) show that a shorter interval leads to a worse estimate of p_e.

Table 5.4 Performance (WAcc) of the Aurora EC algorithm (baseline and subvector versions), linear interpolation and different hard-decision HMM-based mitigation methods over GSM error patterns EP1, EP2 and EP3

EC method	Channel condition		
	EP1 BER $\simeq 0$ %	EP2 BER $= 1.76$ %	EP3 BER $= 3.48$ %
FBMMSE	99.04	99.02	98.83
FMMSE	99.04	98.99	98.18
Raw-MMSE	99.04	98.68	91.32
Viterbi	99.04	99.01	98.23
Linear	99.04	98.96	92.87
Aurora	99.04	98.94	93.40
Aurora-sub	99.04	99.00	97.74

5.3.2.3 MMSE Estimation for Erasure Channels

Finally, we must also consider the estimation of $b_i(\mathbf{y}_t)$ during loss bursts in packet networks. The most common situation is that there are no data available from the channel during the period affected by the loss burst, so we can only use the vectors received before and after the burst from Equations (5.11) and (5.12), which yields the following expected value computation:

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|Y^-, Y^+] = \sum_{i=0}^{N-1} \mathbf{x}^{(i)} P(\mathbf{x}_t^i|Y^-, Y^+) \quad (1 \leq t \leq T) \tag{5.23}$$

Note that this expression does not use any instantaneous information. Therefore, the estimates can be precomputed and stored, so that the corresponding EC algorithm reduces to a table look up. This estimation method has been studied in Gomez *et al*. (2003) for $\mathcal{M} = \mathcal{N}$ over the Aurora FE standard. Its main problem is the huge size that the estimate table can reach by using $\mathcal{N} > 1$, which can make this method useless. The table is first reduced by dividing the loss period into two halves and obtaining forward estimates for the first half and backward estimates for the second half:

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|Y^-] \quad (1 \leq t \leq T/2) \tag{5.24}$$

$$\hat{\mathbf{x}}_t = E[\mathbf{x}_t|Y^+] \quad (T/2 + 1 \leq t \leq T) \tag{5.25}$$

where a combination of $\mathcal{N}$ SVQ vectors (Y^- or Y^+) is called *register*. If the register does not have $\mathcal{N}$ available vectors, because of a previous (or subsequent) loss burst or because it corresponds to the beginning (or end) of the speech utterance, it is completed by repeating the first (last) available vector backward (forward). If a register appears less than μ times, then it is not stored (the Aurora EC algorithm is applied in these cases). Thus, for example, for $\mathcal{N} = 3$ and $\mu = 10$ the number of required registers is reduced from 18 million to

Table 5.5 Comparison of the word accuracies obtained with the Aurora EC algorithm and different statistical-based reconstruction methods over an erasure channel

EC method	Channel condition				
	$ulp = 10\ \%$ $d_{loss} = 1$	$ulp = 20\ \%$ $d_{loss} = 4$	$ulp = 30\ \%$ $d_{loss} = 8$	$ulp = 40\ \%$ $d_{loss} = 12$	$ulp = 50\ \%$ $d_{loss} = 20$
Aurora	99.05	93.45	82.05	69.48	57.97
MMSE	99.05	94.69	85.92	74.73	63.07
MAP	99.02	94.83	85.10	74.70	63.28
Hybrid	99.06	94.78	86.31	75.14	64.07

131,462 for the seven SVQ codebooks of the Aurora FE compression scheme. Finally, the estimates are SVQ quantized to further reduce the required storage memory. The table search can be speeded up by carrying out a binary search. The results obtained by this MMSE-based technique for the Aurora-2 task (clean sentences) over a three-state Markov chain channel, with several channel conditions (different packet loss rates and average loss durations), are shown in Table 5.5 (each packet contains two feature vectors). The MMSE-based technique can be improved by using the MAP technique developed in the next subsection for the nonavailable estimates (corresponding to the less frequent registers) (Gomez *et al*., 2004). This technique is referred to as *Hybrid* in Table 5.5.

As we have already mentioned, it is also possible that, under a FEC transmission scheme, the received vector $\mathbf{y}_t$ $(t = 1, \ldots, T)$ may be a coarser quantized version (with less bits) of the originally transmitted one. Then, the mapping between the transmitted vector $\mathbf{x}^{(i)}$ and the received vector $\mathbf{y}_t$ is not one-to-one and the observation probability $b_i(\mathbf{y}_t)$ must be estimated as the probability that transmitted vector $\mathbf{x}^{(i)}$ becomes $\mathbf{y}_t$. In the case that we do not have any received data of the vector at time t, we can still consider that we have a 0-bit quantized version available. These observation probability assignments have been successfully tested for media-specific FEC in reference Peinado *et al*. (2005a).

5.3.2.4 MAP Estimation

A MAP estimate of the vector at time t is simply obtained by selecting the centroid $\mathbf{x}^{(i)}$, which yields the maximum probability $P(\mathbf{x}_t^{(i)}|Y)$ instead of the expected value computations of Equations (5.11) and (5.12). However, this approach is clearly inferior to MMSE in our EC context. From a computational point of view, MAP estimation can be attractive to obtain an estimate of the optimal vector sequence

$$\hat{X}_1^T = \underset{X_1^T}{\operatorname{argmax}}\, P(X_1^T|Y) \tag{5.26}$$

where $X_1^T = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ represents a vector sequence in the interval $[1, T]$. One possible solution is to employ the same HMM modeling as in the FBMMSE estimation and to obtain the optimal vector sequence by means of the VA described in Section 2.6.3, which is applied from $t = 0$ until $t = T + 1$ (i.e. $\mathcal{M} = \mathcal{N} = 1$). The necessary observation probabilities can be computed in the same way as for FBMMSE or FMMSE.

The performance of this VA-based approach and a comparison with MMSE-based and interpolation techniques over the GSM EP error patterns is shown in Table 5.4 (experiment Viterbi).

In the context of erasure channels, it has also been proposed (Gomez *et al*., 2004; James *et al*., 2004) that a Gaussian distribution be assumed for vector streams. Then, the MAP estimation of a missing vector stream $\mathbf{X}_m = (\mathbf{x}_1^t, \ldots, \mathbf{x}_T^t)$ given the known vectors $\mathbf{X}_0$ (around the lost ones) is obtained as

$$\hat{\mathbf{X}}_m = \underset{\mathbf{X}_m}{\operatorname{argmax}}\, P(\mathbf{X}_m|\mathbf{X}_0) = \boldsymbol{\mu}_m + \Sigma_{m0}\Sigma_{00}^{-1}\left(\mathbf{X}_0 - \boldsymbol{\mu}_0\right) \tag{5.27}$$

where $\boldsymbol{\mu}_m$ and $\boldsymbol{\mu}_0$ are the mean vectors of $\mathbf{X}_m$ and $\mathbf{X}_0$, respectively, Σ_{00} is the covariance matrix of $\mathbf{X}_0$, and Σ_{m0} is the cross-covariance matrix of $\mathbf{X}_m$ and $\mathbf{X}_0$. Although $\boldsymbol{\mu}_m$ and Σ_{m0} are not known, we can consider them (assuming stationarity) as equal to the mean vector and cross-covariance matrix obtained from global statistics. This technique is suitable and efficient for isolated losses. In order to apply it to loss bursts, the isolated loss case can be extended through an iterative strategy where the estimates already obtained are reused for the estimation of the next vector (from the outer to the inner vectors of the burst). Furthermore, since temporal correlations are much greater than correlations among the different features, the estimation can be applied to each feature separately in order to reduce the size of matrix Σ_{00}, which must be inverted. The performance of this MAP technique, taking the ten previous and ten subsequent feature values for the feature being estimated, is also compared with the Aurora EC algorithm and MMSE in Table 5.5. Its performance is similar to MMSE. Although it is much more complex than MMSE, it involves much less amount of memory. The *hybrid* technique described in the previous section tries to reach a compromise between computational burden and memory requirements by combining MMSE and MAP.

5.3.3 Recognizer-based EC Techniques

The success of the earlier estimation techniques is based on the use of a speech model. We can now go further by arguing that in a RSR system we already have a powerful statistical speech model. This is the tool we are using for recognition. Therefore, the question now is, why this should not be used for EC. This is the base of the recognizer-based techniques. The RSR back end is not further divided into two different main stages, decoding and recognition, but the recognizer is now part of a decoder whose goal is to decode the received data into the original message created by the remote speaker. Thus, recognizer-based techniques realize, in a certain sense, the idea of considering an RSR system as a whole communication system in which the different stages are closely related.

In general, recognizer-based EC requires that the recognizer must be fed, as usual, with the feature vectors provided by the source decoder (which may include some type of reconstruction EC) together with a reliability measure for those features. This reliability measure can be used for not only recognition but also other tasks such as out-of-vocabulary (OOV) detection, as we will see in this section. In Chapter 2 (Section 2.9), we studied two different types of techniques to deal with unreliable data in the Viterbi decoding: exponential weighting and missing-data techniques. We see next how they can be applied to recognizer-based EC.

5.3.3.1 Missing-data Techniques

Let us first consider the case in which each feature vector contains both reliable and unreliable features, so that the unreliable features are dropped. In other words, we have an erasure channel that erases some features. Let us denote the subvector containing the reliable features as $\mathbf{x}_{rel}$ and the one containing the unreliable features as $\mathbf{x}_{unrel}$. The whole feature vector is $\mathbf{x} = (\mathbf{x}_{rel}, \mathbf{x}_{unrel})$. We can proceed according to the simple marginalization technique described in Section 2.9.2 (derived from Equation (2.103)). Then, we can use

$$P(\mathbf{x}_{rel}|s_i) = \int_{\mathbf{x}_{unrel}} P(\mathbf{x}|s_i) d\mathbf{x}_{unrel} \tag{5.28}$$

as observation probabilities in each state s_i for the VA decoding. This approach has been applied in (Potamianos and Weerackody, 2001). In this work, static features whose first and/or second MSB bits are detected as erroneous are considered unreliable. For dynamic features, the same error detection test is applied to the features included in the temporal window used for the delta computation, so if any of these features is erroneous, then the dynamic feature being computed is considered unreliable. The performance of this EC technique for AWGN and Rayleigh channels is shown in Table 5.6. The system has a DSR architecture with the encoding scheme of Table 5.2 and the decoded features are directly used by the recognizer without any other EC process. The results can be directly compared with those of Table 3.2, which were obtained without EC.

Let us now consider that whole feature vectors are completely lost by an erasure channel. Without any loss of generality, let us now assume that only one single loss takes place at time $t = l$. If we again apply the marginalization technique, then we do not have any reliable feature in the corrupted or lost feature vector and the probability of Equation (5.28) becomes 1. Therefore, according to Equation (2.102), the probability maximized by the VA is

$$E[P(Q|X, \lambda)] = C a_{q_0,q_1} b_{q_1}(\mathbf{x}_1) a_{q_1,q_2} b_{q_2}(\mathbf{x}_2) \cdots \tag{5.29}$$

$$b_{q_{l-1}}(\mathbf{x}_{l-1}) a_{q_{l-1},q_l} a_{q_l,q_{l+1}} b_{q_{l+1}} b_{q_{l+1}}(\mathbf{x}_{l+1}) \cdots \tag{5.30}$$

$$a_{q_{T-1},q_T} b_{q_T}(\mathbf{x}_T) \tag{5.31}$$

Table 5.6 WER results of a DSR/IWR system operating over AWGN and Rayleigh (for several MS speeds) channels, using missing-data EC at the feature level (after Weerackody *et al.*, 2002)

AWGN channel	
SNR (dB)	WER
Clean	7.1
4	7.4
3	7.6
2	8.8
1	21.6

Rayleigh channel			
SNR (dB)	10 km/h	50 km/h	100 km/h
15	7.3	7.1	7.3
10	8.7	7.4	7.2
7	12.9	8.7	7.7
5	19.9	13.9	11.1

where the time interval $[1, T]$ corresponds to the whole speech utterance. If we take into account that in the Viterbi search transition probabilities are less important than observation probabilities, the above equation can be approximately realized by simply removing vector $\mathbf{x}_l$ from the received vector sequence (Kim and Cox, 2001a). This leads us to a *frame dropping* (or deletion) EC technique. This method could be attractive because of its simplicity. However, it does alter the temporal structure of the vector sequence since it tends to introduce deletion errors when long loss bursts are present.

In the two previous cases, vectors or subvectors are considered to be as either, fully reliable or unreliable. However, the most general case corresponds to the use of soft data with different degrees of reliability, as explained in Section 2.9.2. For example, in the case of a wireless channel, the source and channel decoders must provide the observed features and their corresponding reliability measures, so that the evidence pdf for each feature vector can be built from these measures. A straightforward example is the case in which the observed feature vectors (or subvectors) are MMSE estimates (i.e. expected values $E[\mathbf{x}_t]$) provided by the source decoder. We can assume a Gaussian evidence pdf with a diagonal covariance matrix (no correlated features), whose diagonal elements (variances) are simply computed as second-order moments (Haeb-Umbach and Ion, 2004), that is,

$$\sigma_{x,t}^2 = E[x_t^2] - E[x_t]^2 \tag{5.32}$$

for a given feature x at time t. Finally, the observation probabilities used by the Viterbi decoder are computed as indicated in Equation (2.114).

The soft-data approach can also be applied to erasure channels. In this case, we can consider that each feature at time t is a Gaussian random variable x_t with a mean equal to the replacement value $\hat{x}_t$ (previously obtained by interpolation or estimation) and a variance $\sigma_{x,t}^2$, which can be previously obtained from the speech data. These variances are then applied to Equation (2.114) to account for the uncertainty inherent in the interpolation/estimation process. This approach has been successfully tested in Delaney (2005) in combination with a cubic spline interpolation. The use of interpolation to provide the necessary feature replacements entails that for each considered burst length T we must obtain a different set of variances for each feature (since the accuracy of the interpolation at each time t depends on the length of the erasure burst that must be reconstructed).

5.3.3.2 Weighted Viterbi Algorithm

We studied in Section 2.9.1 that the application of exponential weights to the observation probabilities in the VA algorithm yields the WVA. The exponential weights are a measure (from 0 to 1) of the observation reliability. It is interesting to note that WVA with binary (0/1) weights leads us to the same solution as marginalization, and this is the reason exponential weighting is also sometimes considered like a data-missing technique. The key point for the application of WVA to EC is the determination of these weights from information provided by the previous decoding stages or any useful *a priori* knowledge.

In the case of a frame erasure channel, we can simply apply the frame-level weighting of Equation (2.92) (all features have the same weight) in a binary manner ($\gamma_t = 0$ for a erased vector and $\gamma_t = 1$ for an available vector). In reference Bernard and Alwan (2002)

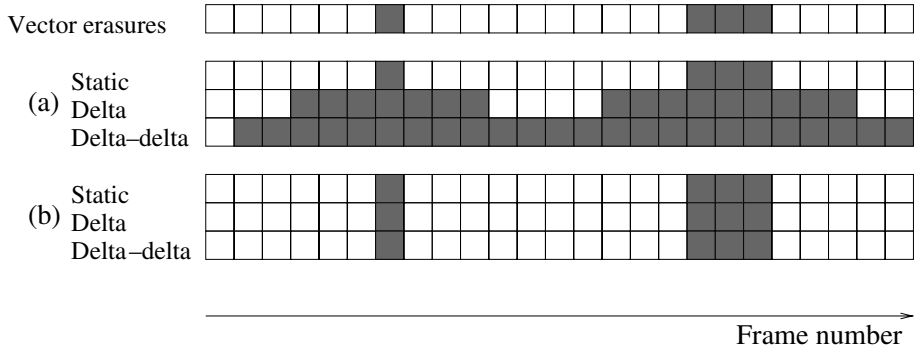


Figure 5.6 Two possible schemes for the weighting of static and dynamic features (after James *et al*., 2004). A gray square indicates a feature subvector with a less-than-one weight. The window radiuses in (a) are $W_v = 3$ and $W_a = 2$ for delta and delta–delta features, respectively

it has been shown that this binary weighting is more helpful than frame dropping for random and bursty erasures and frame repetition for long erasure bursts, although frame repetition is better in the case of short erasure bursts or random erasures. Instead of binary weighting, it is also possible to use weighting functions with linear ($\gamma_t = 1 - (1 - \alpha)t$) or exponential ($\gamma_t = \alpha^t$) (with $\alpha = 0.8, 0.7$, respectively) variation in combination with the Aurora EC algorithm, obtaining some improvement with respect to binary weighting (Cardenal *et al*., 2004) and (James and Milner, 2005).

The application of *frame weighting* assumes that the whole feature vector, including static and dynamic features, has been transmitted, so that we can apply the same weight to all the features in a given vector. However, only static features are usually transmitted and the dynamic ones are computed at the back end from the static features received. Therefore, if we consider that a delta feature is computed from the static values contained in a temporal window of radius W_v and that if any of these values is unreliable (or lost), then the corresponding delta feature is also unreliable. The same argument can be used for the computation of delta–delta features from delta features using a window of radius W_a. This effect forces a *feature weighting* scheme as shown in Figure 5.6(a), where the different features from the same feature vector can have different weights, as in Equation (2.93). In Milner and James (2004) it is shown that, for the case of binary weights, it is better to use frame weighting, with the temporal scheme of Figure 5.6(b), instead of a feature weighting as that of Figure 5.6(a). This is performed by interpolating the static features required to obtain the corresponding dynamic features (the interpolation is exclusively carried out for this purpose).

Although frame weighting can be useful, it has been shown that the exponential feature weighting of Equation (2.93) clearly outperforms it (Bernard and Alwan, 2002; Cardenal and Garcia, 2004). As we have just seen in Figure 5.6(a), feature weighting may be forced by considering the temporal windows employed for the computation of the dynamic features. We have also seen that this is not the best approach when using binary weighting. However, it can be useful when a more sophisticated weighting scheme is applied. An example of this can be found in James and Milner (2005), where the following weighting for features reconstructed by the Aurora EC algorithm is proposed:

1. Static features: exponential weighting.

$$\gamma_{S,t} = \begin{cases} \alpha^t & 1 \le t \le T/2 \\ 1 - \alpha^{(T-t+1)} & t/2 < t \le T \end{cases} \tag{5.33}$$

2. Delta and delta–delta features:

$$\gamma_{\Delta,t} = \frac{\sum_{k=1}^{W_v} k\gamma_{S,t+k}\gamma_{S,t-k}}{\sum_{k=1}^{W_v} k} \quad \gamma_{\Delta\Delta,t} = \frac{\sum_{k=1}^{W_a} k\gamma_{\Delta,t+k}\gamma_{\Delta,t-k}}{\sum_{k=1}^{W_a} k} \tag{5.34}$$

Figure 5.7 shows an example of this weighting scheme for $\alpha = 0.7$, $W_v = 3$ and $W_a = 2$.

In the most general case of feature weighting, it is possible to use a specific weight for each particular feature (instead of for groups of features). For example, in Bernard and Alwan (2002), the use of a basic EC based on a forward repetition $\hat{\mathbf{x}}_t = \mathbf{x}_0$ along with the following weighting function for the reconstructed feature $\hat{x}_t(j)$ at time t is proposed:

$$\gamma_{j,t} = \begin{cases} 1 & \text{correct frame} \\ \sqrt{\rho_j(t)} & \text{incorrect frame of an erasure burst starting at } t = 1 \end{cases} \tag{5.35}$$

where $\rho_j(t)$ is the (normalized) time autocorrelation function of feature $x_t(j)$. Note that $0 \le \rho_j(t) \le 1$ and that $\rho_j(t)$ will typically be a decreasing function of t with $\rho_j(0) = 1$. This autocorrelation-based weighting strategy can be straightforwardly modified to be used with the Aurora EC algorithm instead of with repetition, by simply dividing each erasure burst into two halves and proceeding backward on the second half Cardenal and Garcia (2004).

The previous frame or feature weighting techniques are straightforwardly applied to lossy packet channels by simply detecting the lost frames. However, for wireless channels, as usual, an error detection mechanism is required. For example, in reference Bernard and Alwan (2002), the λ-soft error detection already explained in Section 5.2.1 and based on the reliability measure λ defined in Equation (5.1), along with the weighting function of Equation (5.35) is applied. Furthermore, in the case of wireless channels and soft-decision decoding, we can use the reliability measures provided by the channel decoder to generate

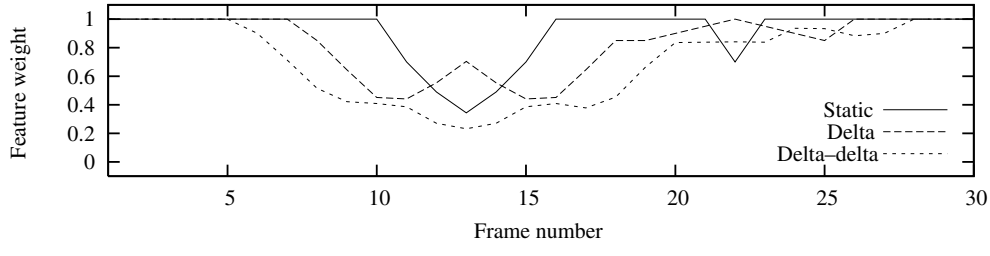


Figure 5.7 Example of feature weighting using different weights for static, delta and delta–delta features with $W_v = 3$ and $W_a = 2$ (after James *et al.*, 2005)

the weighting functions. In particular, $\gamma_t = \lambda_t^2$ has been tested in Bernard and Alwan (2002), although similar results as those of the feature weighting of Equation (5.35) were obtained. A similar approach was proposed in Potamianos and Weerackody (2001) for feature weighting and wireless transmission. In this work, the error detection mechanism is the one explained in Section 5.2.1 for convolutional codes and the feature weights are computed as

$$\gamma_{j,t} = f(C) = \frac{\alpha + C}{\alpha + 1} \tag{5.36}$$

where $C(0 \leq C \leq 1)$ is the confidence associated to decoded feature $y_t(j)$ at time t and α is a smoothing constant. The confidence measure C is obtained from the hard-decoded feature $y_t(j)$ as $C = 1 - E/\sigma_j^2$, where σ_j^2 is the variance of the considered feature and E is the MSE error between the transmitted feature $x_t(j)$ and the received feature $y_t(j)$,

$$E = E[(y_t(j) - x_t(j))^2 | y_t(j)] = \sum_{i=0}^{N-1} P(x^{(i)}(j)|y_t(j))(y_t(j) - x^{(i)}(j))^2 \tag{5.37}$$

where $\{x^{(i)}(j); i = 0, \ldots, N-1\}$ is the set of possible feature values. Probabilities $P(x^{(i)}(j)|y_t(j))$ can be computed from the received soft bits the same as in Equation (5.21).

5.3.3.3 Robust Out-of-vocabulary Detection

As studied in the preceding subsections, the recognizer can use channel reliability measures to improve its performance. These measures can also be used to improve the performance of other postrecognition tasks such as OOV detection. By means of OOV detection we try to detect OOV words contained in the input utterance X in order to improve the overall performance. This is carried out by imposing a threshold T to the following likelihood ratio:

$$LR(X) = \frac{P(X|H_0)}{P(X|H_1)} \tag{5.38}$$

where H_0 represents an in-vocabulary word and H_1 is the OOV word (modeled by a filler model). When $LR(X) < T$, the hypothesis H_0 is rejected and a OOV word is detected. However, a degraded channel modifies the optimal threshold value. In Tan *et al.* (2003), a function $T = T(FER)$ (FER is the frame error rate) to automatically adapt the threshold to the channel condition is proposed. This function is a fourth-order polynomial whose coefficients are computed so that the false rejection rate for several BER values is maintained approximately constant.

5.3.4 Error Concealment for NSR

The previous subsections have been devoted to EC applied to both DSR and B-NSR. Let us now consider the case of speech-based NSR (S-NSR). Speech codecs are usually implemented with some EC algorithm. The simplest algorithms are more oriented to avoid

generation of annoying sounds by the decoder than to try to reconstruct the damaged speech data. This is the case of some standards like the IS-641 (Equation (5.10)), the ETSI EFR and AMR codecs (ETSI, 1999c, 2000a), the ITU-T G.729 or the Internet low bitrate codec (iLBC). The damaged frames are substituted by repetition or extrapolation (of previous correct frames). When the error situation is prolonged, the output signal is usually muted to indicate the breakdown of the channel to the user. However, since EC is implemented at the decoder, system providers are allowed to implement their own more sophisticated solutions (Fingscheidt and Vary, 2001; Perkins *et al*., 1998). Obviously, these EC algorithms are not conceived for ASR applications, so it may be convenient to introduce some kind of S-NSR-specific concealment. We will see some of these specific techniques next.

5.3.4.1 Solutions for S-NSR

Since codec EC algorithms are not conceived for ASR applications, a S-NSR recognizer still sees the concealed and decoded signal as degraded. In particular, if a substitution and muting concealment is applied and the error burst affects more than one frame, we can observe a "hole" in the signal, which is a consequence of the muting mechanism. Then, we have a twofold problem. First we have to detect such holes and then mitigate them. In Karray *et al*. (1998), a statistical detection of holes based on amplitude, variance and segment length is proposed. Once the hole is detected, the word in which it appears is rejected to diminish substitution and false rejection rates.

A complete analysis of the errors that may affect recognition on a EFR-based system is carried out in Gomez *et al*. (2004). A first source of errors is due to the bits that are not protected by the channel encoder. This is a *background noise* with very little effect on the recognition performance. The most important source of errors is due to the frames marked as erroneous (BFI flag set to 1) by the channel decoder (*bad frame noise*). The effect of this degradation is not limited to frames marked with BFI $= 1$ (*bad frame isolated noise*) but is prolonged in time to subsequent frames with BFI $= 0$ because of the "memory" of the codec (*memory noise*) (typical of LPC-based codecs). This effect can affect up to 20 frames. The isolated effect of these different noises over recognition is shown in Table 5.7 for the GSM EPx (x $= 1, 2, 3$) patterns and for the Aurora-2 task (clean sentences). Gomez *et al*. (2004) propose a EC scheme in which bad frame isolated noise is mitigated by applying linear interpolation, while the memory noise is compensated

Table 5.7 Performance (WAcc) achieved by the combination of interpolation and compensation (Equation (5.39)) of feature vectors in a S-NSR system

Channel	DSR	EFR	Background noise	Bad frame noise	Memory noise	Bad frame isolated noise	Interpolation + Compensation
Clean	99.04	98.70	–	–	–	–	98.81
EP1	99.04	98.44	98.50	98.61	98.44	98.68	98.64
EP2	98.95	96.91	98.31	97.53	97.73	98.28	98.19
EP3	94.41	84.48	98.22	85.80	93.54	90.47	94.04

by means of the following expression:

$$\hat{\mathbf{x}}_t = \mathbf{x}_t + \mathbf{r}(t, \mathbf{x}_t, L) \qquad (t = 1, \ldots, 20) \tag{5.39}$$

where $t = 0$ is the time at which the last bad frame occurs, $\mathbf{x}_t$ is the feature vector corresponding to the decoded frame at time $t\,(> 0)$, and $\mathbf{r}(t, \mathbf{x}_t, L)$ is the correction applied when the previous error burst has a length of L frames. The correction vectors $\mathbf{r}$ are precomputed from the training database. A special correction vector for the case of a clean transmission between the original vectors and those after coding/decoding is also computed in an effort to compensate the coding artifacts. The performance achieved with this interpolation/compensation EC scheme is also shown in Table 5.7. The results of this table show that the EFR-based system can achieve a performance close to DSR. However, this EC technique requires the knowledge of the BFI flag (it could be known if TFO operation is possible). Also, we still have the problem of a small degradation in clean and almost-clean conditions due to the codec artifacts.

In Pelaez *et al*. (2002) it is postulated that transmission errors cause artificial high frequencies in the temporal trajectories of the spectral parameters (MFCCs or LSPs). Since these parameter trajectories have a limited bandwidth (up to 8–15 Hz for the LSPs and up to 10–30 Hz for the MFCCs), the effect of channel errors can be mitigated by low-pass filtering each spectral parameter with its corresponding cutoff frequency. The technique is more suitable for LSPs than for MFCCs (since they have smaller bandwidths), and this requires that the LSPs be first computed from the decoded signal and then transformed into MFCCs for the final recognition. The advantage of this technique, compared to the previous ones, is that it does not require any kind of previous error detection.

6

Front-end Processing for Robust Feature Extraction

6.1 Introduction

Speech recognition systems performance is severely degraded when evaluated in conditions that differ from those used to collect training data. The mismatch between training and testing data is due to differences in speaker characteristics and acoustic environment (microphone, transmission channel and noise conditions). It is commonly assumed that the best solution is to train the recognition system using data from the same acoustic environment where the system is going to be used. Nevertheless, this is not possible in most situations. The goal of the so-called robust speech recognition techniques is to reduce the impact of this mismatch over the speech recognition system performance.

The mismatch between training and test conditions is mainly due to differences in the acoustic domain (microphone, transmission channel, background noises, etc.). These differences can be modeled as distortions in the signal domain, feature domain and models domain; and robust speech recognition techniques can be classified depending on the domain they are used in (see Figure 6.1).

Preprocessing techniques are applied in the signal domain, before feature extraction. The goal of these techniques is to partially remove additive background noises. Typical examples of these techniques are spectral subtraction (Lockwood and Boudy, 1979; Vaseghi and Milner, 1997) and Wiener filtering (WF) (Bernstein and Shallom, 1991; Vaseghi and Milner, 1993). These techniques usually are based on an estimation of the background characteristics that are obtained during nonspeech periods. Selection of nonspeech periods is accomplished with a voice activity detector (VAD).

Feature compensation techniques are applied in the feature domain; The objective is to compensate for the environment effects in such a way that the processed features come as close as possible to those that would be obtained in the clean training environment. These approaches are usually based on an ML estimation of clean features given the distorted ones; and they use models for the clean features and for the environment effects. Vector Taylor series (VTS) (Moreno *et al.*, 1998) and stereo-piecewise linear

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

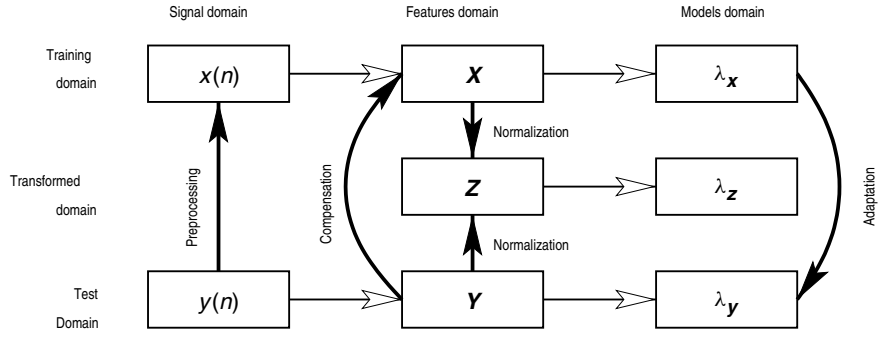


Figure 6.1 A possible classification of robust speech recognition techniques

compensation for environments (SPLICE) (Deng *et al*., 2001), for example, belong to this group of techniques.

As an alternative to feature compensation, the acoustic models trained with clean speech features can be adapted to every new acoustic environment. Recognition is performed on the original distorted features but using adapted models. Techniques belonging to this group are parallel model combination (PMC) or MLLR (Leggetter and Woodland, 1995).

It is also possible to transform the features to a new domain less affected by variations of the acoustic environments. In this situation, the transformation is applied for both training and test, and the acoustic models are therefore trained in this new domain. Cepstral mean normalization (CMN) (Atal, 1974) is perhaps the most successful feature normalization technique. Other techniques belonging to this group are cepstral mean and variance normalization (CMVN) (Viikki and Laurila, 1998), relative spectral processing (RASTA) (Hermansky and Morgan, 1994) and the recently proposed nonlinear feature transformation techniques like histogram equalization (HEQ) (de la Torre *et al*., 2003) and quantile-based HEQ (Hilgher and Ney, 2001).

In this chapter we will focus on a subset of the aforementioned techniques which are more suitable for their implementation at the front end of the speech recognition system. Preprocessing techniques work on the speech signal and, therefore, must be implemented at the front end of the recognition system, before feature extraction. On the other hand, model adaptation techniques cannot be implemented at the front end because acoustic models are stored at the back end. The other two groups of techniques work over the speech features, and can be implemented either at the front end or at the back end. Anyway, feature normalization techniques are less resource-demanding than the feature compensation ones. For this reason it is more common to have feature normalization techniques implemented at the front end.

In Section 6.2 we will present noise reduction preprocessing techniques based on both spectral subtraction and WF. Section 6.3 will be devoted to VAD. Finally, in Section 6.4 we will present classical and recently proposed feature normalization techniques.

6.2 Noise Reduction Techniques

Noise reduction belongs to the group of preprocessing techniques. For situations in which the speech signal is corrupted by additive uncorrelated noise, these techniques can be

used to suppress part of the noise, yielding a "cleaned" signal. When both the desired signal and the additive noise are stationary Gaussian, the optimum design (in an MMSE sense) is the WF, discussed in the following section.

Instead of trying to directly restore the time domain signal, short-time spectral attenuation techniques rely in the fact that the most important perceptual characteristic of an audio signal is the spectral magnitude. Therefore, these techniques are designed to obtain and estimate the spectral magnitude of the desired signal. In general, this estimation is obtained as the output of an SNR-dependent spectral attenuator, and different techniques differ in the attenuation rule. Spectral subtraction is derived as the square root of the optimal ML estimator of each speech spectral component variance, while WF is derived as the modulus of the optimal MMSE estimator of each signal spectral component (see (McAulay and Malpass, 1980)). A different approach was proposed by Ephraim and Malah (1984), where the suppression rule is derived as an optimal MMSE estimator of the spectral magnitude.

All the above cited techniques need an estimation of the spectral characteristics of the noise. In some situations it is possible to have a noise reference from a separate microphone. But in single microphone systems (like actual RSR systems), only the corrupted signal is observed. In these situation, the noise statistics can be obtained only during periods of speech absence. A VAD is then used to discriminate between speech and noise-only frames. VAD algorithms are discussed in Section 6.3.

6.2.1 Wiener Filters

The theory of linear least-square error filter formulated by Wiener plays a central role in a wide range of applications such as linear prediction, echo cancellation, signal restoration, channel equalization and system identification. In the WF design, the optimization criterion is to minimize the squared distance between the filter output and the desired signal. In its basic form, the Wiener theory assumes that signals are generated by stationary processes. For nonstationary signals, however, the filter coefficients can be periodically updated for blocks of N samples. Such block-adaptive filters can be used for signals like speech, which may be considered stationary within small blocks of samples. In this section we briefly review the foundations of WFs and its application to the problem of noise reduction in speech signals.

In many practical situations, we are given a signal $y(m)$ consisting of the sum of a desired signal $x(m)$ and an undesired interference $w(m)$, and we want to design a filter that suppresses the undesired interference. The objective in designing such a filter is to cancel the interfering signal while preserving the characteristics of the desired signal. The situation is depicted in Figure 6.2.

The criterion used to optimize the coefficients of the filter impulse response $h(m)$ is to minimize the MSE between the filter output $\hat{x}(m)$ and the desired signal $x(m)$. This criterion has the advantage of simplicity and mathematical tractability and can be proved to be optimal when the signals are assumed to be Gaussian-distributed.

The basic assumptions are that the temporal sequences $x(m)$ and $w(m)$ are zero-mean stationary processes. The filter can be assumed to be either a finite impulse response (FIR) filter or infinite impulse response (IIR) filter. In general, the formulation of IIR WFs results in a set of nonlinear equations, whereas the FIR WF formulation results in

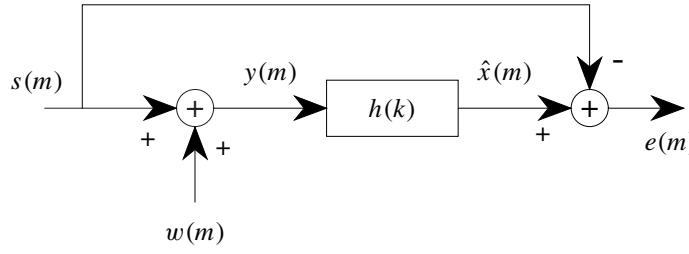


Figure 6.2 Model for the linear filtering problem

a set of linear equations and has a closed form. Therefore, we present here only the last formulation, as they are simple to compute and inherently stable.

6.2.1.1 FIR Wiener Filter

For an FIR filter with an impulse response of length M, the input–output relation is

$$\hat{x}(m) = \sum_{k=0}^{M-1} h(k)y(m-k) \tag{6.1}$$

The error signal is defined as the difference between the desired filter output and the desired signal

$$e(m) = \hat{x}(m) - x(m) = \sum_{k=0}^{M-1} h(k)y(m-k) - x(m) \tag{6.2}$$

which depends on the filter coefficients $h(k)$. And the MSE is

$$\mathcal{E}_M = E[e(m)^2] = E\left[\left(\sum_{k=0}^{M-1} h(k)y(m-k) - x(m)\right)^2\right] \tag{6.3}$$

where $E[\cdot]$ is the expectation operator. The derivatives of $\mathcal{E}_M$ with respect to the filter coefficients are

$$\frac{\partial \mathcal{E}_M}{\partial h(l)} = 2\sum_{k=0}^{M-1} h(k)r_{yy}(l-k) - 2r_{xy}(l) \quad \forall l = 0..M-1 \tag{6.4}$$

where $r_{yy}(l)$ and $r_{xy}(l)$ are defined as the autocorrelation of the input sequence and the cross-correlation between the input and desired sequences, respectively:

$$r_{yy}(l) = E[y(m)y(m-l)] \tag{6.5}$$

$$r_{xy}(l) = E[x(m)y(m-l)] \tag{6.6}$$

The minimum value for the MSE is found by simply setting to zero the derivatives in Equation (6.4), which yields a set of M linear equations with M unknowns (i.e. the filter coefficients $h(k)$)

$$\sum_{k=0}^{M-1} h(k) r_{yy}(l-k) = r_{xy}(l) \quad \forall l = 0..M-1 \tag{6.7}$$

This set of linear equations specifies the optimum filter coefficients and is known as *Wiener–Hopf* equation (also normal equations). This equation set can be expressed in the matrix form as,

$$\mathbf{R}_{yy}\mathbf{h} = \mathbf{r}_{xy} \tag{6.8}$$

where $\mathbf{R}_{yy}$ is an $M \times M$ Toeplitz matrix whose elements are the autocorrelations coefficients of the input signal[1]

$$\mathbf{R}_{yy} = \begin{pmatrix} r_{yy}(0) & r_{yy}(1) & r_{yy}(2) & \cdots & r_{yy}(M-1) \\ r_{yy}(1) & r_{yy}(0) & r_{yy}(1) & \cdots & r_{yy}(M-2) \\ r_{yy}(2) & r_{yy}(1) & r_{yy}(0) & \cdots & r_{yy}(M-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{yy}(M-1) & r_{yy}(M-2) & r_{yy}(M-3) & \cdots & r_{yy}(0) \end{pmatrix} \tag{6.9}$$

and $\mathbf{r}_{xy} = [r_{xy}(0) r_{xy}(1) r_{xy}(2) \ldots r_{xy}(M-1)]^\top$ is a $1 \times M$ column vector whose components are the cross-correlation coefficients of the input and the desired signals. Therefore, the optimal filter coefficients are given by

$$\mathbf{h} = \mathbf{R}_{yy}^{-1}\mathbf{r}_{xy} \tag{6.10}$$

and the resulting MMSE achieved by the WF is

$$\text{MMSE}_M = min\ \mathcal{E}_M = r_{xx}(0) - \sum_{k=0}^{M-1} h(k) r_{xy}(k) \tag{6.11}$$

where $r_{xx}(0) = E[x(m)^2]$ is the variance of the desired signal.

6.2.1.2 The Orthogonality Principle

The normal equations for the optimum filter coefficients given by Equation (6.7) can alternatively be derived by applying the orthogonality principle of linear mean-square estimation. Simply stated, the MSE $\mathcal{E}_M$ is minimum if the filter coefficients are selected in such a way that the error signal is orthogonal to each of the input data points in the estimate

$$E[e(m)y(m-l)] = 0 \quad \forall l = 0, \ldots M-1 \tag{6.12}$$

[1] Recall that $r_{yy}(k) = r_{yy}(-k)$ by definition.

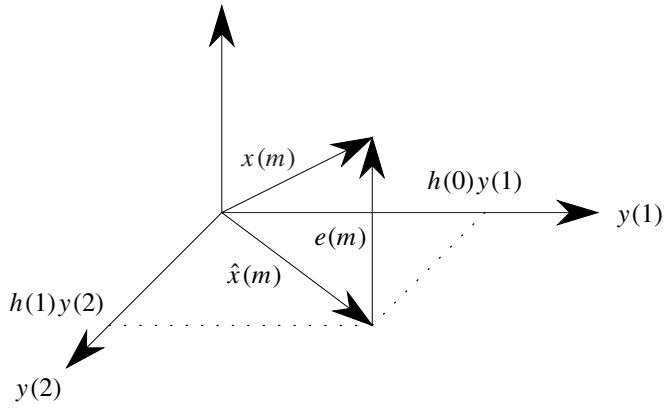


Figure 6.3 Geometric interpretation of linear MMSE

In a geometrical interpretation, the output $\hat{x}(m)$ of the WF is a vector in the subspace spanned by the input data $\{y(m), 0 \leq m \leq M-1\}$. The error $e(m)$ is a vector from $\hat{x}(m)$ to $x(m)$ (i.e. $x(m) = e(m) + \hat{x}(m)$). The orthogonality principle states that the error is minimum when $e(m)$ is perpendicular to the data subspace (i.e. $e(m)$ is perpendicular to each data point $y(m), 0 \leq m \leq M-1$). This is illustrated in Figure 6.3.

The solution obtained from the normal Equations (6.7) is unique if the data $y(m)$ are linear independent, in which case, the autocorrelation matrix $\mathbf{R}_{yy}$ is nonsingular. On the contrary, if the data are linear dependent, the rank of $\mathbf{R}_{yy}$ is less than M and the solution is not unique. There exist infinitely many solutions that can be expressed as linear combinations of the solutions for a reduced set of linear independent data points equal to the rank of $\mathbf{R}_{yy}$.

6.2.1.3 Frequency Domain Wiener Filter

In the previous derivation we have constrained the filter to be FIR and causal. By removing this condition, we can use infinite past and future samples of the input, so that the output is obtained as, estimate

$$\hat{x}(m) = \sum_{k=-\infty}^{\infty} h(k)y(m-k) \tag{6.13}$$

The resulting filter is noncausal and, therefore, physically unrealizable because it uses future unobserved samples.

Application of the orthogonality principle yields the Wiener–Hopf equation for the noncausal filter in the form

$$\sum_{k=-\infty}^{\infty} h(k)r_{yy}(l-k) = r_{xy}(l) \qquad -\infty \leq l \leq \infty \tag{6.14}$$

and the minimum error is

$$\text{MMSE}_{NC} = r_{xx}(0) - \sum_{k=-\infty}^{\infty} h(k) r_{xy}(k) \tag{6.15}$$

Taking the Fourier transform of both terms of Equation (6.14) yields to following solution for the optimum WF in the frequency domain[2]

$$H_{NC}(f) = \frac{S_{xy}(f)}{S_{yy}(f)} \tag{6.16}$$

where f is the frequency variable and $S_{yy}(f)$ and $S_{xy}(f)$ are the power spectral density of $y(m)$ and the cross-power spectral density of $x(m)$ and $y(m)$.

The solution can also be found by direct formulation of the optimization criterion in the frequency domain. The WF output can be written as

$$\hat{X}(f) = H(f)Y(f) \tag{6.17}$$

The expected value of the squared error is

$$E[|X(f) - \hat{X}(f)|^2] = E[(X(f) - H(f)Y(f))^*(X(f) - H(f)Y(f))] \tag{6.18}$$

where * denotes the complex conjugate. The optimal solution is obtained by setting the complex derivative of the above equation with respect to the filter $H(f)$ to zero,

$$\frac{\partial}{\partial H(f)} E[|X(f) - \hat{X}(f)|^2] = 2H(f)S_{yy}(f) - 2S_{xy}(f) = 0 \tag{6.19}$$

$$H(f) = \frac{S_{xy}(f)}{S_{yy}(f)} \tag{6.20}$$

where

$$S_{yy}(f) = E[Y(f)Y^*(f)] \tag{6.21}$$

$$S_{xy}(f) = E[X(f)Y^*(f)] \tag{6.22}$$

This is the same solution as in Equation (6.16).

6.2.1.4 Application of Wiener Filters to Additive Noise Reduction

Consider a signal $x(m)$ observed in additive noise $n(m)$

$$y(m) = x(m) + n(m) \tag{6.23}$$

[2] The derivation is based on the Wiener–Khinchine relation (i.e. the autocorrelation sequence and the power spectral density functions form a Fourier transform pair), and the Fourier transform property that the convolution in time is equivalent to multiplication in frequency.

If we assume that signal and noise are uncorrelated, it follows that the autocorrelation sequence of the observed signal is the sum of the autocorrelations sequences of the signal and noise processes $r_{yy}(k) = r_{xx}(k) + r_{nn}(k)$. Also, the cross-correlation of the input and desired signals equals the autocorrelation of the desired signal $r_{xy}(k) = r_{xx}(k)$, and therefore we can write

$$\mathbf{R}_{yy} = \mathbf{R}_{xx} + \mathbf{R}_{nn} \tag{6.24}$$

$$\mathbf{r}_{xy} = \mathbf{r}_{xx} \tag{6.25}$$

and the optimum filter coefficients are given by,

$$\mathbf{h} = (\mathbf{R}_{xx} + \mathbf{R}_{nn})^{-1}\mathbf{r}_{xx} \tag{6.26}$$

which is the optimum WF for additive uncorrelated noise suppression. To obtain the solution, the autocorrelation sequence of both the noise $r_{nn}(k)$ and the desired signal $r_{xx}(k)$ must be known.

Further insight can be obtained from the frequency domain formulation of the optimal filter. In this case, the observed signal is given by

$$Y(f) = X(f) + N(f) \tag{6.27}$$

where $X(f)$ and $N(f)$ are the signal and noise spectra. Under the assumption that they are uncorrelated processes, we can obtain the following relations for the power spectral densities:

$$\begin{aligned} S_{yy}(f) &= E[Y(f)Y^*(f)] = E[(X(f) + N(f))(X(f) + N(f))^*] \\ &= S_{xx}(f) + S_{nn}(f) \end{aligned} \tag{6.28}$$

$$\begin{aligned} S_{xy}(f) &= E[X(f)Y^*(f)] = E[X(f)(X(f) + N(f))^*] \\ &= S_{xx}(f) \end{aligned} \tag{6.29}$$

and therefore, the optimum frequency domain WF for additive noise suppression is

$$H(f) = \frac{S_{xx}(f)}{S_{xx}(f) + S_{nn}(f)} \tag{6.30}$$

By defining the a priori SNR $\xi(f) = S_{xx}(f)/S_{nn}(f)$ as the power of the desired signal relative to the noise power, we can write

$$H(f) = \frac{\xi(f)}{1 + \xi(f)} \tag{6.31}$$

From this relation we can view the WF as a frequency domain SNR-dependent attenuator. The attenuation is bounded in the interval $0 \leq H(f) \leq 1$. Those frequency components with $S_{xx}(f) \gg S_{nn}(f)$ receive little attenuation, while those with $S_{xx}(f) \ll S_{nn}(f)$ are heavily attenuated.

Noise-free frequency bands have $H(f) = 1$ and remain unaltered, while noise-only frequency bands have $H(f) = 0$ and are fully suppressed. Therefore, when the signal and noise spectra do not overlap, the WF fully suppresses the noise and leaves the signal unaltered. When both spectra overlap, the WF enhances those frequency bands with high SNR in relation to those with low SNR, which receive a greater attenuation.

6.2.2 Short-time Spectral Attenuation

Short-time spectral attenuation goal is to obtain an estimate of the spectral magnitude of a signal observed in additive uncorrelated noise. In general, this estimation is obtained as the output of an SNR-dependent spectral attenuator, and different techniques differ in the attenuation rule. Spectral subtraction attenuator is derived as the square root of the optimal ML estimator of each speech spectral component variance, while WF is derived as the modulus of the optimal MMSE estimator of each signal spectral component (see (McAulay and Malpass, 1980)). A different approach was proposed in Ephraim and Malah (1984), where the suppression rule is derived as an optimal MMSE estimator of the spectral magnitude. In this section we discuss different approaches of this group of techniques.

6.2.2.1 Power Spectral Subtraction

Spectral subtraction is a technique designed to restore the power or magnitude spectrum of a signal observed in additive, uncorrelated noise. The restored spectrum is obtained by the subtraction of the mean noise spectrum from the noisy signal spectrum. The noise spectrum is estimated during periods where the speech signal is absent, under the assumption that it is a stationary or slow varying random process. When the time domain speech signal is to be restored, an estimation of the magnitude spectrum is combined with the phase of the noisy signal, and transformed back to the time domain by means of the DFT.

For a signal observed in additive noise, the time domain relation among the noisy observations $y(m)$, the desired signal $x(m)$ and the additive noise $n(m)$ was established in Equation (6.23) and the corresponding frequency domain relation in Equation (6.27). The power spectrum of the observed signal can be obtained as

$$|Y(f)|^2 = |X(f)|^2 + |N(f)|^2 + \left[X(f)N^*(f) + X^*(f)N(f)\right] \tag{6.32}$$

where the cross-term on the right has a null expected value under the assumption that the desired signal and the noise are uncorrelated, and therefore Equation (6.32) can be approximated by

$$|Y(f)|^2 \approx |X(f)|^2 + |N(f)|^2 \tag{6.33}$$

The noise power is unknown, but we can estimate it using the average periodogram over M frames of the incoming signal known to contain only noise (i.e. when no speech signal is present), as long as the noise is stationary

$$|\hat{N}(f)|^2 = \frac{1}{M}\sum_{i=1}^{M}|Y_i(f)|^2 \tag{6.34}$$

For slow varying noises, an alternative estimation can be obtained using a first-order recursive filter to get an updated noise estimate during nonspeech periods

$$|\hat{N}(f)|^2 = \rho|\hat{N}(f)|^2 + (1-\rho)|Y_i(f)|^2 \tag{6.35}$$

An intuitive estimation of the desired signal power can then be obtained from Equation (6.33) by subtracting the average noise power from the incoming signal

$$|\hat{X}(f)|^2 = |Y(f)|^2 - |\hat{N}(f)|^2 \tag{6.36}$$

A problem with this approach is that $|X(f)|^2$ must be positive, but Equation (6.36) is by no means constrained to give positive estimates. It is therefore necessary to modify the estimation to map negative values into positive ones by means of

$$T[|\hat{X}(f)|^2] = \begin{cases} |\hat{X}(f)|^2 & |\hat{X}(f)|^2 > \beta|Y(f)|^2 \\ fn[|Y(f)|^2] & \text{otherwise} \end{cases} \tag{6.37}$$

where $fn[|Y(f)|^2]$ is the noise floor (i.e. the minimum value allowed by the estimator), which in its simple form can be fixed to a fraction of the input signal power yielding the following modified estimator:

$$|\hat{X}(f)|^2 = max\left\{|Y(f)|^2 - |\hat{N}(f)|^2, \beta|Y(f)|^2\right\} \tag{6.38}$$

where $\beta < 1$ is a positive constant.

Spectral subtraction can also be viewed as a linear filtering of the input signal simply rewriting Equation (6.38) as

$$|\hat{X}(f)| = |Y(f)|H_{PSS}(f) \tag{6.39}$$

$$H_{PSS}(f) = \sqrt{max\left\{1 - \frac{1}{SNR(f)}, \beta\right\}} \tag{6.40}$$

where we have defined $SNR(f)$ as the relative input signal power with respect to the estimated mean noise power.

$$SNR(f) = \frac{|Y(f)|^2}{|\hat{N}(f)|^2} \tag{6.41}$$

Equations (6.39) and (6.40) give an interpretation of spectral subtraction as an SNR-dependent attenuator. Frequency components with low SNR values are attenuated more than those for which the SNR is high.

6.2.2.2 Magnitude Spectral Subtraction

Spectral subtraction can be applied to the magnitude spectrum instead of the power spectrum. In this case (Boll, 1979), the estimator is formulated as

$$|\hat{X}(f)| = |Y(f)| - |\hat{N}(f)| \tag{6.42}$$

where now $|\hat{N}(f)|$ is an estimate of the mean magnitude of the noise spectrum. The same discussion about dealing with negative estimates applies in this case, and the same approach can be used to modify the estimator

$$|\hat{X}(f)| = max\left\{|Y(f)| - |\hat{N}(f)|, \beta|Y(f)|\right\} \tag{6.43}$$

which results in a similar SNR-dependent attenuation rule

$$|\hat{X}(f)| = |Y(f)|H_{MSS}(f) \tag{6.44}$$

$$H_{MSS}(f) = max\left\{1 - \frac{1}{\sqrt{SNR(f)}}, \beta\right\} \tag{6.45}$$

Both power and magnitude spectral subtraction can be considered as belonging to a more general class of noise reduction techniques, which are referred to as *short-time spectral attenuation*. All these techniques use a common approach in which the spectral magnitude of the desired signal is estimated by means of an SNR-dependent attenuation of the input signal spectral magnitude, and differ in the definition of the attenuation rule.

6.2.2.3 Implementation Details

Short-time spectral attenuation is implemented as depicted in Figure 6.4, using the overlap-add method (Proakis and Manolakis, 1996).

- First, the input signal is buffered and divided into overlapped temporal frames.
- Each frame is windowed using a temporal window (usually a Hanning window is used).
- The DFT is used to obtain the instantaneous magnitude and phase.
- A VAD decision is used to update the noise statistics.
- The spectral attenuation rule is applied to the input spectral magnitude.
- The estimated magnitude is combined with the input signal phase.
- The inverse DFT is used to transform the signal back to the time domain.
- Finally, the output overlapped frames are added together, and the output signal is unbuffered.

An example of spectral subtraction is illustrated in Figure 6.5.

6.2.2.4 Musical Noise

The implementation of spectral subtraction described in the previous sections results in an output speech signal with a much lower noise content but which exhibits what is called *musical noise* (Berouti *et al*., 1979). This effect is caused by random variations of the noise around its mean value, and appears in frequency bands for which the input signal and the noise have similar power values $|Y(f)|^2 \approx |\hat{N}(f)|^2$.

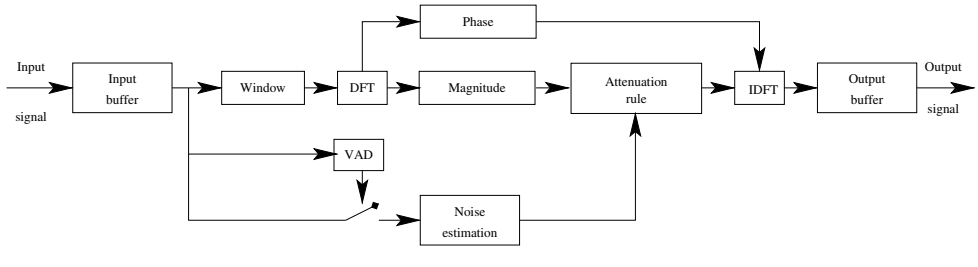


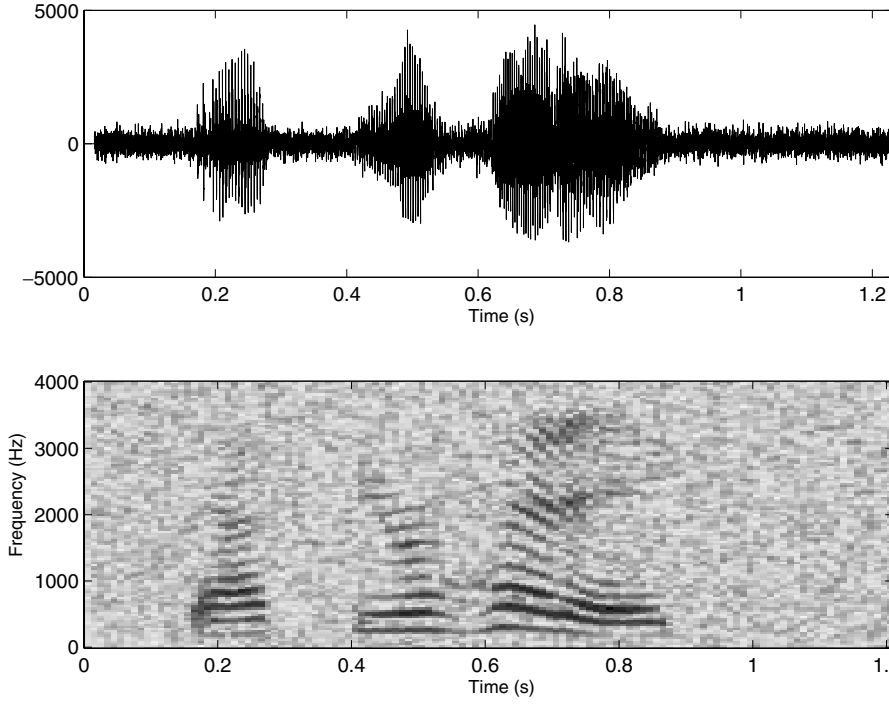
Figure 6.4 Schematic short-time spectral attenuation implementation

This situation is illustrated in Figure 6.6. A frequency band f_0 for which $|Y(f_0)|^2 < |\hat{N}(f_0)|^2$ is attenuated by a factor β, while another frequency band f_1 in its neighborhood with $|Y(f_1)|^2 > |\hat{N}(f_1)|^2$ is much less attenuated. These abrupt changes in the frequency introduce frequency components that appear and disappear rapidly, owing to the random behavior of the noise. This effect is also noticeable in the spectrogram of Figure 6.5(b), in which the random tones are visible in the nonspeech parts. This distortion is perceived as a set of short tones at random frequencies, and is the main source of degradation in the perceived signal and also in the accuracy of speech recognition systems using short-time spectral attenuation.

Musical noise is caused by random variations of the noise, which introduce random variations in the estimated SNR. One possibility is to smooth the estimated SNR over adjacent frequency bands. Alternatively, a time-smoothing recursive filter can be used

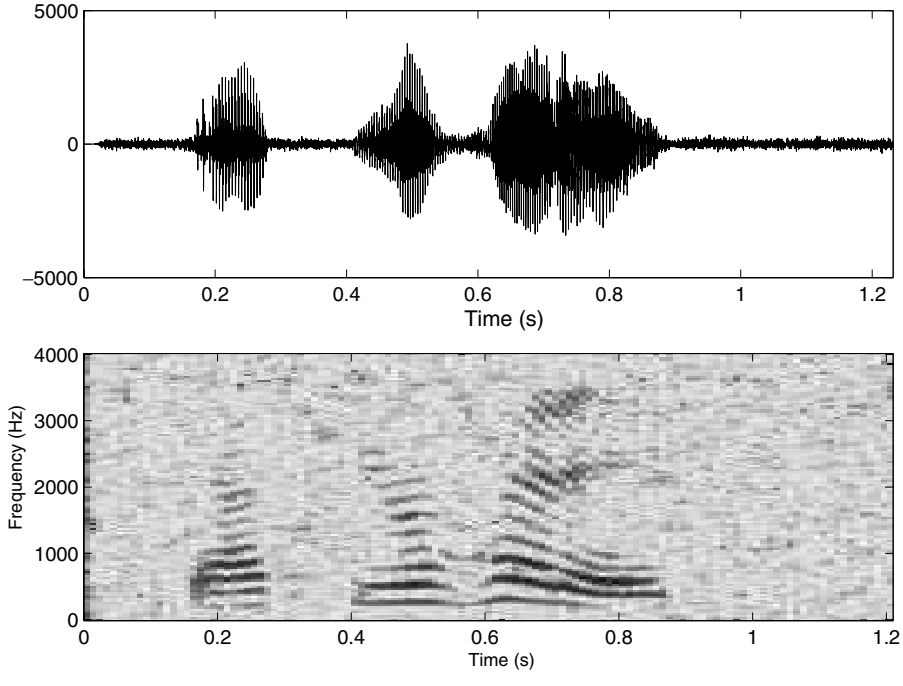
$$SNR(f,t) = \gamma SNR(f,t-1) + (1-\gamma)\frac{|Y(f,t)|^2}{|\hat{N}(f,t)|^2} \tag{6.46}$$

Both time and frequency smoothing can be used to obtain better SNR estimations at the expense of lower noise suppression.



(a) Noisy speech signal

Figure 6.5 Spectral subtraction example: (a) The speech signal and the spectrogram of noisy utterance; (b) The spectral subtracted signal and spectrogram



(b) Speech signal after spectral subtraction

Figure 6.5 (*continued*)

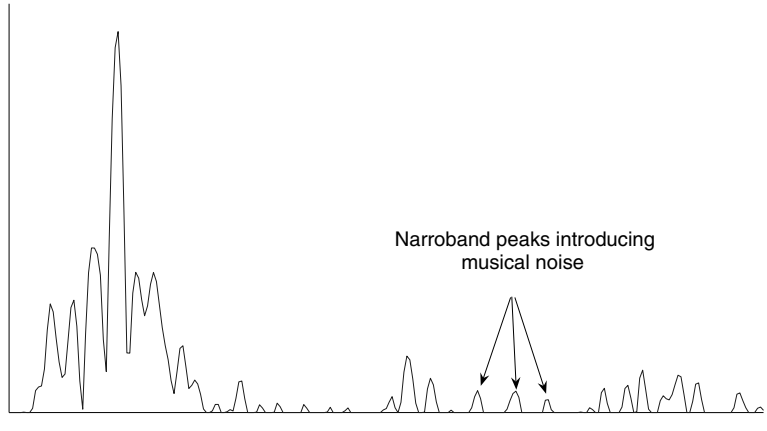


Figure 6.6 Illustration of musical noise

Oversubtraction can also be used to reduce the musical noise. In this variant, the average noise estimation is multiplied by a factor $\alpha > 1$ resulting in a modified suppression rule

$$|\hat{X}(f)|^a = max\left\{|Y(f)|^a - \alpha|\hat{N}(f)|^a, \beta|Y(f)|^a\right\} \tag{6.47}$$

$$H_{OS}(f) = \left[max\left\{1 - \alpha\frac{|\hat{N}(f)|^a}{|Y(f)|^a}, \beta\right\}\right]^{1/a} \tag{6.48}$$

where $a = 1$ is used for magnitude and $a = 2$ for power spectral subtraction. By subtracting more than the estimated noise mean, it is more probable to have $|Y(f)|^a < \alpha|\hat{N}(f)|^a$ than $|Y(f)|^a > \alpha|\hat{N}(f)|^a$ for frequency bands having $|Y(f)|^a \approx |\hat{N}(f)|^a$, and therefore the musical noise is reduced. Typical values are in the range $1 \leq \alpha \leq 2$. Typical values for β are in the range $0.001 \leq \beta \leq 0.01$ which correspond to attenuations of -30 dB and -20 dB, respectively, in the power subtraction case.

6.2.2.5 Nonlinear Spectral Subtraction

As stated before, oversubtraction can achieve a greater noise suppression at low SNRs, but at the expense of introducing distortion in the speech signal, because more than the estimated noise mean is subtracted.

Nonlinear spectral subtraction are heuristic methods that use the fact that at low SNR oversubtraction can produce better results. These methods use an oversubtraction factor that is a function of the local SNR, in such a way that the oversubtraction factor increases as the SNR decreases. One simple form of SNR-dependent oversubtraction factor is given by

$$\alpha(SNR(f)) = 1 + \frac{sd(|N(f)|}{|\hat{N}(f)|} \tag{6.49}$$

where $sd(|N(f)|)$ is the standard deviation of the noise at frequency f. In this approach the subtraction factor depends on both the noise mean and the variance. For deterministic noises there is no oversubtraction ($sd(|N(f)| = 0$). For a white Gaussian noise, $sd(|N(f)|) = |\hat{N}(f)|$ and $\alpha(SNR(f)) = 2$. In general, α takes values between 1 and 2.

In the nonlinear spectral subtraction proposed by Lockwood and Boudy (1979), the spectral subtraction filter is formulated as

$$H_{NSS}(f) = \frac{|Y(f)|^2 - |N(f)|^2_{NL}}{|Y(f)|^2} \tag{6.50}$$

where $|N(f)|^2_{NL}$ is a nonlinear estimation of the amount of noise to be subtracted, which is obtained as a function of the local SNR and the mean and maximum values of the noise

$$|N(f)|^2_{NL} = \frac{\underset{\text{over M frames}}{max}\left\{|N(f)|^2\right\}}{1 + \gamma SNR(f)} \tag{6.51}$$

γ being an experimental design parameter. With this definition, the amount of subtracted noise approaches the maximum estimated value as the SNR decreases. As the

SNR increases, the amount of subtracted noise approaches zero. The values are further limited to be in the interval

$$|\hat{N}(f)|^2 \leq |N(f)|^2_{NL} \leq 3|\hat{N}(f)|^2 \tag{6.52}$$

6.2.2.6 Ephraim and Malah Suppression Rule

Spectral subtraction is derived as the square root of the optimal ML estimator of each speech spectral component variance, while WF is derived as the modulus of the optimal MMSE estimator of each signal spectral component (see (McAulay and Malpass, 1980)). A different approach was proposed by Ephraim and Malah (1984), where the suppression rule is derived as an optimal MMSE estimator of the spectral magnitude. As shown by Cappe (1994), with this estimator, it is possible to obtain significant noise reduction while avoiding the musical noise phenomenon. The Ephraim and Malah (EPH) noise suppression filter can be expressed as follows:

$$H_{\text{EPH}}(f) = \frac{\sqrt{\pi}}{2}\frac{\sqrt{v(f)}}{\gamma(f)}M\left(v(f)\right) \tag{6.53}$$

$$v(f) = \frac{\xi(f)}{1+\xi(f)}\gamma(f) \tag{6.54}$$

$$M(\phi) = \exp\left(-\frac{\phi}{2}\right)\left[(1+\phi)I_o\left(\frac{\phi}{2}\right) + \phi I_1\left(\frac{\phi}{2}\right)\right] \tag{6.55}$$

where I_o and I_1 are the modified Bessel functions of zero and first order, respectively. And $\xi(f)$ and $\gamma(f)$ are identified as the a priori and a posteriori SNRs of each spectral component ((McAulay and Malpass, 1980)) defined by

$$\xi(f) = \frac{S_{xx}(f)}{S_{nn}(f)} \tag{6.56}$$

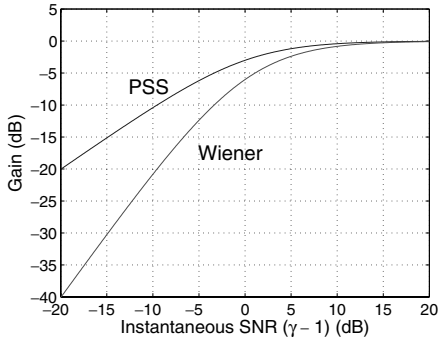
$$\gamma(f) = \frac{|Y(f)|^2}{S_{nn}(f)} \tag{6.57}$$

The a priori SNRs $\xi(f)$ represents the information of the unknown spectrum gathered from previous observations, and is estimated in a decision-directed (DD) approach given by

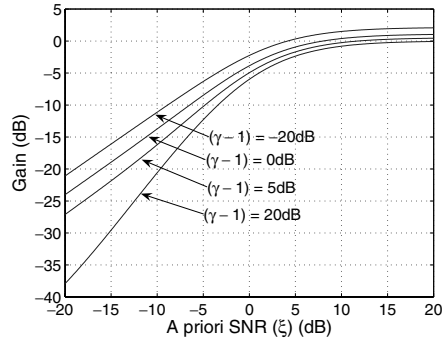
$$\xi(f) = (1-\alpha)\max\{\gamma(f)-1, 0\} + \alpha\frac{|H_{\text{EPH}}(f)Y(f)|^2}{S_{nn}(f)} \tag{6.58}$$

with $\alpha = 0.98$.

This estimator provides a noise suppression characteristic that depends on both the a priori and the a posteriori SNRs as shown in Figure 6.7. The attenuation is mainly controlled by the a priori SNR, while the a posteriori SNR acts as a control parameter whose influence is limited to the situations where the a priori SNR is low. In these situations (mostly when speech is not present), Equation (6.58) provides a highly smoothed version of ξ and, as a consequence, a low variance estimation of the a priori SNR that prevents rapid changes in the attenuation. This way, musical noise is greatly reduced. On the contrary, when the a posteriori SNR is high (when speech is present), Equation (6.58)



(a) Power spectral subtraction (PSS) and Wiener gains as a function of the instantaneous SNR



(b) EPH Gain as a function of the a priori SNR for various values of the instantaneous SNR

Figure 6.7 Ephraim and Malah gain in comparison with power spectral subtraction and WF gains

applies little smoothing to ξ, which is desirable to not distort the speech signal. Time smoothing was also proposed in Boll (1979). However, the superior performance of this approach relies on the automatic selection of the smoothing level. Smoothing is applied only when the signal level is near or below the noise level, while little smoothing is applied when the signal level is well above the noise level, which prevents the distortion of nonstationary signals like speech.

6.3 Voice Activity Detection

VAD is an important topic in speech processing with application in different fields such as robust speech recognition, discontinuous transmission, real-time speech transmission on the Internet or combined noise reduction and echo cancellation schemes in the context of telephony.

In robust speech recognition, VAD plays an important role in two principal applications. The first one is the estimation of the background noise statistics needed by single channel noise reduction algorithms like spectral subtraction or WF. Although some techniques have been proposed to continuously update the background noise estimation, it is usually computed during nonspeech periods and therefore a VAD algorithm is needed.

The second application of VAD is to discard nonspeech frames as a previous step in speech recognition. This is commonly referred to as *frame-dropping*. Removing nonspeech frames from the speech recognition system input stream effectively reduces the insertion error rate of the system.

For noise suppression algorithms such as WF or spectral subtraction, VAD is critical in attaining a high level of performance. These techniques estimate the noise spectrum during nonspeech periods in order to compensate for its harmful effect on the speech signal. Thus, VAD is more critical for nonstationary noise environments, since it is needed to update the constantly varying noise statistics, as a mismatch classification error strongly affects the system performance.

In order to palliate the importance of VAD in noise suppression systems Martin (1993) proposed an algorithm which continually updated the noise spectrum in order to prevent a misclassification of the speech signal. These techniques are faster in updating the noise but usually capture signal energy during speech periods, thereby degrading the quality of the compensated speech signal.

Several different approaches have been proposed for VAD algorithm design. Different features can be used to address the problem of VAD, the most commonly used being those based on the full-band or sub-band energy measures, but including also perceptually based measures, the zero-crossing rate or the pitch.

6.3.1 Full-band-energy-based VAD

The simplest approach to perform VAD is based on the detection of significant changes on the full-band energy of the speech signal. In this approach, the log-energy of the signal is computed for each frame. Given an estimation of the average background noise log-energy, the actual frame is classified as speech or nonspeech by simply computing the difference between the actual log-energy and the background level. The decision is made on the basis of a fixed threshold. This situation is illustrated in Figure 6.8. This approach can also be seen as a classification rule based on an estimation of the instantaneous full-band SNR. Improvements of this basic approach have been proposed ((Srinivasan and Gersho, 1993)) by extending the analysis to several spectral sub-bands and using adaptive thresholds.

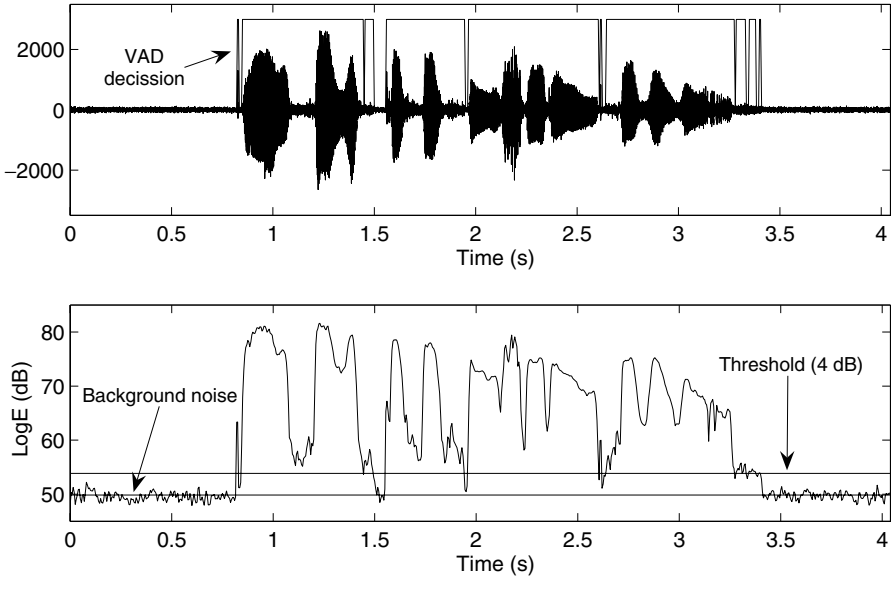


Figure 6.8 An example of full-band-energy-based VAD. Threshold fixed 4 dB above the background noise energy

6.3.1.1 Hangover

Transitions from nonspeech to speech portions of an utterance are normally characterized by a rapid increment of the signal energy. On the contrary, it is common to have slow energy decreases in the transitions from speech to nonspeech, which can cause misclassification. To avoid this problem, VAD algorithms usually delay the speech to nonspeech decision. This technique is commonly referred to as *hangover* and is implemented using a counter. Whenever the actual frame energy is above the threshold, the counter is reset to a fixed value. When the energy is below the threshold, the counter is decreased and the actual frame is classified as nonspeech only if the counter reaches a value of zero. The algorithm is summarized in Figure 6.9 and also illustrated in Figure 6.10.

```
if (frameEn-meanEn) > THRESHOLD
then
  VAD = 1
  nbSpeechFrame = nbSpeechFrame + 1
  if nbSpeechFrame > MIN_SPEECH_FRAME_HANGOVER
  then
    hangOver = HANGOVER
  end
else
  if (hangOver==0)
    VAD = 0
  else
    VAD = 1
    hangOver = hangOver - 1
  end
end
```

Figure 6.9 Pseudocode of a hangover algorithm for delaying the VAD decision

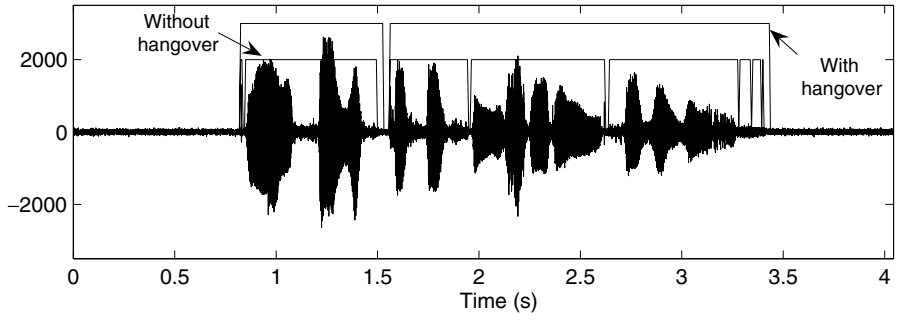


Figure 6.10 An example of hangover. Threshold fixed 4 dB above the background noise energy. Hangover period fixed at 5 frames

Initially, the `hangOver` counter is set to zero. When a fixed number of frames with energy above the threshold are observed, the variable is set to `HANGOVER` (the number of frames to delay the VAD decision). When a frame with energy below the threshold is observed, the counter is decreased and the VAD flag is set to 0 only if the counter has reached a value of 0 (indicating that `HANGOVER` previous frames have energy values below the threshold). In other cases, the VAD flag is set to 1.

6.3.1.2 Background Noise Level Estimation

The log-energy of the background noise can be estimated from an initial part of the utterance containing only noise (i.e. 10 or 20 frames). A better estimation can be obtained using a first-order recursive filter to track slow variations of the background noise energy. The updating is performed for frames labeled as nonspeech by the VAD.

$$\texttt{meanEn} = \alpha\texttt{meanEn} + (1-\alpha)\texttt{frameEn} \tag{6.59}$$

This is useful in case of nonstationary noises. Typical values for α are in the range 0.95–0.99.

6.3.2 Statistical VAD

A different approach to VAD design is to formulate the decision from a statistical point of view. Assuming a statistical model for the speech signal Sohn *et al*. (1999) proposed a VAD algorithm based on an LRT test.

For speech corrupted by additive uncorrelated noise, two hypotheses can be considered:

$$H_0: \text{ speech absent: } \mathbf{X} = \mathbf{N} \tag{6.60}$$

$$H_1: \text{ speech present: } \mathbf{X} = \mathbf{N} + \mathbf{S} \tag{6.61}$$

where $\mathbf{S}$, $\mathbf{N}$, and $\mathbf{X}$ are the L dimensional DFT coefficients of the speech, noise and noisy speech of a short temporal frame, S_k, N_k and X_k being the k-th components of the respective Fourier vectors.

A common statistical model for the DFT coefficients (Ephraim and Malah, 1984) is to consider that the Fourier coefficients of each process are asymptotically independent Gaussian random variables. Under this model, the probability density functions (PDFs) conditioned to the H_0 and H_1 hypothesis are given by

$$p(\mathbf{X}|H_0) = \prod_{k=0}^{L-1} \frac{1}{\pi\lambda_N(k)} \exp\left\{-\frac{|X_k|^2}{\lambda_N(k)}\right\} \tag{6.62}$$

$$p(\mathbf{X}|H_1) = \prod_{k=0}^{L-1} \frac{1}{\pi[\lambda_N(k)+\lambda_S(k)]} \exp\left\{-\frac{|X_k|^2}{[\lambda_N(k)+\lambda_S(k)]}\right\} \tag{6.63}$$

where $\lambda_N(k)$ and $\lambda_S(k)$ are the variances of N_k and S_k, respectively. The likelihood ratio for the k-th frequency band (DFT coefficient) is therefore

$$\Lambda = \frac{p(\mathbf{X}|H_1)}{p(\mathbf{X}|H_0)} = \prod_{k=0}^{L-1} \Lambda_k \tag{6.64}$$

$$\Lambda_k = \frac{1}{1+\xi_k} \exp\left\{ -\frac{\gamma_k \xi_k}{1+\xi_k} \right\} \tag{6.65}$$

$$\xi_k \triangleq \frac{\lambda_S(k)}{\lambda_N(k)} \tag{6.66}$$

$$\gamma_k \triangleq \frac{|X_k|^2}{\lambda_N(k)} \tag{6.67}$$

where ξ_k and γ_k are the a priori and a posteriori SNR of the k-th frequency band.

The decision rule is formulated in terms of the geometric mean of the likelihood ratios of the individual frequency bands, which is consistent with the assumption that Fourier coefficients are asymptotically independent random variables. It can be formulated, in logarithmic form, as

$$\log \Lambda = \frac{1}{L} \sum_{k=0}^{L-1} \log \Lambda_k = \frac{1}{L} \sum_{k=0}^{L-1} \left\{ \frac{\gamma_k \xi_k}{1+\xi_k} - \log(1+\xi_k) \right\} \underset{H_0}{\overset{H_1}{\gtrless}} \eta \tag{6.68}$$

where η is a fixed threshold. The noise variance $\lambda_N(k)$ in each frequency band can be estimated from nonspeech periods using the same approach previously described for the estimation of the background noise level.

For the estimation of the a priori SNR ξ_k, two approaches can be used. The simplest one is to use an ML estimator based on the a posteriori SNR

$$\hat{\xi}_k^{(ML)} = \gamma_k - 1 \tag{6.69}$$

Using this estimator, we get a decision rule directly related with the discrete form of the Itakura–Saito distortion

$$\log \hat{\Lambda}^{(ML)} = \frac{1}{L} \sum_{k=0}^{L-1} \{\gamma_k - \log \lambda_k - 1\} \underset{H_0}{\overset{H_1}{\gtrless}} \eta \tag{6.70}$$

It is well known that the Itakura–Saito distortion has a positive definite magnitude, which implies that this form of the decision rule is biased toward H_1.

To reduce this bias, Sohn *et al*. (1999) proposed the use of a DD approach ((Ephraim and Malah, 1984)) for the estimation of the a priori SNR

$$\hat{\xi}_k^{(DD)} = \alpha \frac{\hat{A}_k^2(n-1)}{\lambda_N(k)} + (1-\alpha) \max\{\gamma_k(n) - 1, 0\} \tag{6.71}$$

where n is the frame index, and $\hat{A}_k(n-1)$ is the MMSE estimation of the spectral magnitude of the signal in the previous frame ((Ephraim and Malah, 1984)). This approach yields a smoother estimation of the a priori SNR and reduces the fluctuations of the estimated likelihood during nonspeech periods.

6.3.3 Using Long-term Information

Most VAD perform the speech/nonspeech classification on the basis of features extracted from the frame under consideration. This is the case with the two previously described

approaches. The instantaneous observations are compared with an averaged estimation of the background noise process, and the decision is based on a fixed or adaptive threshold.

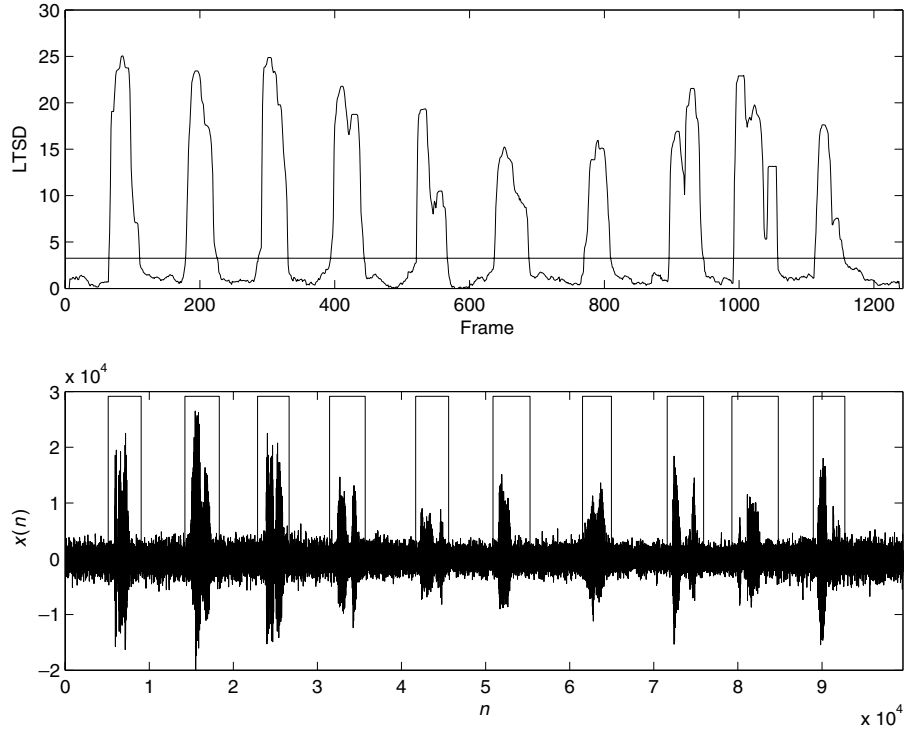
Recently, several approaches have been proposed (Ramirez *et al*. (2004); Ramirez *et al*. (2005)) which combine information extracted from multiple frames instead of using a single-frame-based observation. The main goal in this approach is to build a combined observation with less variance than the single-frame-based observations, yielding a more robust decision.

In Ramirez *et al*. (2004), the decision rule is based on the so-called long-term spectral divergence (LTSD), defined in terms of the long-term spectral envelope (LTSE). Let $X(k,l)$ be the k-th DFT coefficient of a given frame l, and the order N LTSE for the k-th band of frame l is defined as

$$LTSE_N(k,l) = \max_{-N \le j \le N} \{X(k,l+j)\} \tag{6.72}$$

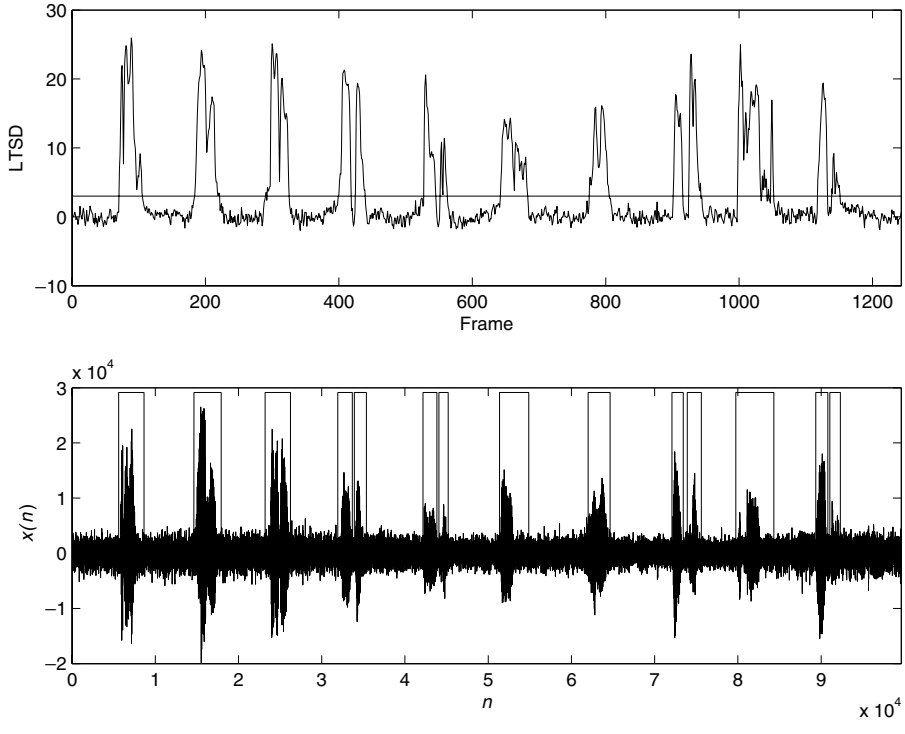
The combined observation is therefore obtained as the maximum value of a set of $2N+1$ frames around the actual frame. The LTSD for the l-th frame is then defined as

$$LTSD_N(l) = 10\log_{10}\left(\frac{1}{NFFT}\sum_{k=0}^{NFFT-1}\frac{LTSE^2(k,l)}{N^2(k)}\right) \tag{6.73}$$



(a) LTSD order N = 6

Figure 6.11 LTSD and VAD output of a noisy utterance for orders $N = 0$ (a) and $N = 6$ (b)



(b) LTSD order N = 0

Figure 6.11 (*continued*)

where $N^2(k)$ is an averaged estimation of the k-th band energy of the background noise. When N is set to zero, the long-term information is discarded, and the LTSE becomes an instantaneous measure of the spectral divergence between the actual frame and the background noise. Increasing the order N results in a smoother LTSD, which in turn gives a better VAD performance. As an example, Figure 6.11 shows the LTSD and VAD outputs for a noisy utterance for orders $N = 0$ and $N = 6$.

The robustness of using multiple-observations-based decision rules is also exploited in Ramirez *et al.* (2005), where the statistical VAD presented Section 6.3.2 is reformulated to use a multiple observation likelihood test (MO-LRT). Following the same notation as in the previous section, the single observation LRT for a given frame t is stated as

$$\Lambda_t \stackrel{\triangle}{=} \frac{p(\mathbf{X}_t|H_1)}{p(\mathbf{X}_t|H_0)} \tag{6.74}$$

If instead of a single frame, we consider a set of $2m + 1$ frames around the actual one $\{X_{t-m}, \ldots, X_t, \ldots, X_{t+m}\}$, a MO-LRT can be defined as follows

$$\Lambda_t^{(MO)} \stackrel{\triangle}{=} \frac{p(\mathbf{X}_{t-m}, \ldots, X_t, \ldots, X_{t+m}|H_1)}{p(\mathbf{X}_{t-m}, \ldots, X_t, \ldots, X_{t+m}|H_0)} \tag{6.75}$$

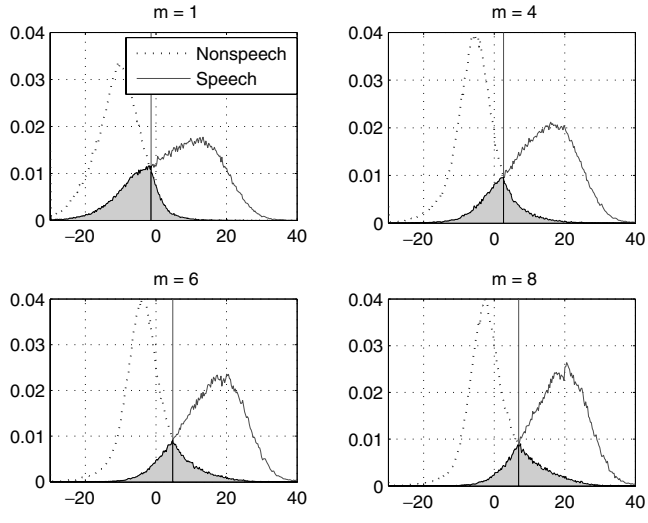


Figure 6.12 Probability distribution of the log-likelihood ratio for the MO-LRT VAD using different number of observations m. Histograms obtained for the distant microphone recordings of the Spanish SpeechDat-Car database

This expression involves the joint probability of the observations, which can be simplified under the assumption of statistical independence yielding

$$\Lambda_t^{(MO)} = \prod_{l=-m}^{m} \frac{p(\mathbf{X}_{t+l}|H_1)}{p(\mathbf{X}_{t+l}|H_0)} \tag{6.76}$$

Finally, under the same statistical model used by Sohn *et al.* (1999), the log-likelihood ratio can be found to be,

$$\log \Lambda_t^{(MO)} = \frac{1}{L} \sum_{l=-m}^{m} \sum_{k=0}^{L-1} \left\{ \frac{\gamma_{t+l,k}\xi_{t+l,k}}{1+\xi_{t+l,k}} - \log(1+\xi_{t+l,k}) \right\} \underset{H_0}{\overset{H_1}{\gtrless}} \eta \tag{6.77}$$

which involves not only the a priori $\xi_{t,k}$ and a posteriori $\gamma_{t,k}$ SNRs of the actual frame but also the values corresponding to the m preceding and m following frames.

The better discriminative behavior of the MO-LRT is illustrated in Figure 6.12. This figure shows the probability distributions of the log-likelihood ratio (Equation 6.77) of speech and nonspeech classes for different numbers of observations m. The probability distributions have been estimated using a hand-labeled version of the Spanish SpeechDat-Car database ((Moreno *et al.*, 2000)). It is clear that by increasing the number of observations m, a more discriminative test is obtained. The speech and nonspeech classes are better separated, and the classification error is therefore reduced.

6.4 Feature Normalization

The main goal of feature normalization techniques is to transform features in such a way that the variability induced by changes in the acoustic environment is minimized. Recalling the discussion presented in Chapter 3, a linear channel distortion can be modeled as a constant bias in the MFCC domain. CMN can be used to suppress the effects of this type of distortion by simply removing the mean value of each cepstral coefficient. For real-time implementations, CMN can be viewed as an FIR filter working over the time sequences of cepstral coefficients.

Filtering the time sequence of features has also been proposed in other approaches like the so-called RASTA ((Hermansky and Morgan, 1994)). In this approach, the time sequence of log-filterable energies is filtered with an IIR band-pass filter to enhance the most discriminative components of the modulation spectra.

When speech is observed in additive noise, other higher-order effects appear in the MFCC domain, including a reduction of the dynamic range of the coefficients. CMNV is an extension of CMN, which also takes into account this effect.

In general, the acoustic environment induces a nonlinear transformation in the cepstral domain, and these linear transformation techniques can only partially remove its effects. In this section we also discuss some recently proposed techniques that make use of more general approaches in which nonlinear feature transformations are used instead of linear ones.

6.4.1 Cepstral Mean Normalization

There are situations in which the predominant effect of the environment is due to a linear channel distortion. This occurs, for example, when using different microphones with different frequency responses. Even using the same microphone, the frequency response depends on the room acoustics and the distance from the speaker to the microphone. Another similar situation is the case of telephone speech recognition, where each call has a different frequency response.

CMN ((Atal, 1974)) is a simple but powerful technique to handle convolutional distortions, which increases the robustness of speech recognition systems to unknown linear filtering channels. Let us consider the temporal sequence of cepstral vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T\}$ from a given signal $x(n)$ (i.e. an utterance) whose sample mean is given by

$$\bar{\mathbf{x}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t \tag{6.78}$$

A normalized version of the cepstral sequence is defined by subtracting the sample mean

$$\hat{\mathbf{x}}_t = \mathbf{x}_t - \bar{\mathbf{x}} \tag{6.79}$$

Consider now that the signal $y(n)$ is the result of filtering the signal $x(n)$ through a linear channel whose impulse response is $h(n)$. In this situation (see Section 3.4), the sequence of cepstral vectors for $y(n)$ is of the form

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{h} \tag{6.80}$$

where $\mathbf{h}$ is the cepstral vector corresponding to the frequency response of the channel, which is assumed to be constant over time. The sample mean of this new cepstral sequence is

$$\bar{\mathbf{y}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{y}_t = \frac{1}{T}\sum_{t=1}^{T}(\mathbf{x}_t + \mathbf{h}) = \bar{\mathbf{x}} + \mathbf{h} \tag{6.81}$$

and the normalized cepstral sequence is given by

$$\hat{\mathbf{y}}_t = \mathbf{y}_t - \bar{\mathbf{y}} = (\mathbf{x}_t + \mathbf{h}) - (\bar{\mathbf{x}} + \mathbf{h}) = \mathbf{x}_t - \bar{\mathbf{x}} = \hat{\mathbf{x}}_t \tag{6.82}$$

which indicates that CMN is invariant against liner filtering operations.

CMN is performed sentence by sentence for both training and test utterances. For sufficiently long utterances ($T \to \infty$) it is expected that the cepstral mean vector $\bar{\mathbf{x}}$ will be equal for all the utterances recorded under the same environment conditions, and that it will mainly contain information about the environment. Therefore, subtracting the cepstral mean will remove cepstral variations due to the environment. On the contrary, for short utterances, the cepstral mean will also contain information about the phonemes present in a particular utterance. As an example, consider that a given utterance contains only one vowel (i.e. the /a/ phoneme). As vowels are quite stationary sounds, the cepstral values will be very similar to the cepstral mean, and after CMN they will be almost zero. But this will be the same situation for any utterance containing a single vowel and therefore it will be very difficult to distinguish such utterances, and the error rate will be very high. Empirically, it has been found that CMN does not degrade the recognition rate on matched conditions (training and test recordings from the same environment) as long as the utterances are longer than 2–4 seconds. When a channel mismatch is present (i.e. training and test utterances are recorded with different microphones) CMN provides significant error rate reductions.

CMN also provides robustness against speaker variations. The error rate is reduced when using CMN in a speaker-independent system even in matched conditions, when no channel mismatch is present. The explanation is that the cepstral mean not only characterizes the transmission channel but also the speaker average frequency response. Removing the cepstral mean, CMN performs some kind of speaker normalization.

6.4.1.1 Real-time Implementation of CMN

CMN requires the collection of all frames of a given utterance before the cepstral mean can be computed and removed from the cepstral sequence, and therefore it cannot be used in a real-time system.

Alternatively, CMN can be seen as the output of a linear filter operating over the time sequence of cepstral coefficients. Consider a discrete temporal sequence of a given cepstral coefficient of length N

$$\{x(0), x(T_s), \ldots, x(kT_s), \ldots, x((N-1)T_s)\} \tag{6.83}$$

where T_s is the time interval between successive samples. In the following, we will omit T_s for simplicity and write the time sequence as

$$\{x(0), x(1), \ldots, x(k), \ldots, x(N-1)\} \tag{6.84}$$

The mean value of the utterance is computed as

$$\bar{x} = \frac{1}{N}\sum_{k=0}^{N-1} x(k) \tag{6.85}$$

and the subtracted coefficients are computed as

$$\hat{x}(k) = x(k) - \bar{x} \tag{6.86}$$

Alternatively, the mean value of a cepstral coefficient can be estimated using only past values with a first-order recursive estimator

$$\bar{x}(k) = \alpha\bar{x}(k-1) + (1-\alpha)x(k) \tag{6.87}$$

which is removed from the actual cepstral value

$$\hat{x}(k) = x(k) - \bar{x}(k) \tag{6.88}$$

Using (6.87) and (6.88) we can write the recursive relation

$$\hat{x}(k) - \alpha\hat{x}(k-1) = \alpha\,(x(k) - x(k-1)) \tag{6.89}$$

which is equivalent to a high-pass IIR filter with a transfer function of the form

$$H(z) = \frac{\alpha(1-z^{-1})}{1-\alpha z^{-1}} \tag{6.90}$$

having a zero in $z = 1$ and a pole in $z = \alpha$. Note that now the average value $\bar{x}(k)$ is time variant. The value of α is selected around 0.998 to provide a sufficient smooth estimation of the mean.

Real-time CMN can also be implemented using a segmental approach, by means of a high-pass FIR filter. The subtracted mean is again time dependent, and estimated averaging over a set of M past and future observations of the cepstral coefficient

$$\hat{x}(k) = x(k) - \bar{x}(k) = x(k) - \frac{1}{2M+1}\sum_{m=-M}^{M} x(k+m) \tag{6.91}$$

which is equivalent to filter the cepstral sequence using a high-pass FIR filter with impulse response

$$h(k) = \begin{cases} 1/(2M+1) & -M \le k < 0 \\ 1 - 1/(2M+1) & 0 \\ 1/(2M+1) & 0 < k \le M \end{cases} \tag{6.92}$$

and whose transfer function is of the form

$$H(z) = 1 - \frac{1}{2M+1}\sum_{k=-M}^{M} z^{-k} \tag{6.93}$$

6.4.2 Frequency Analysis of Time-filtered Features

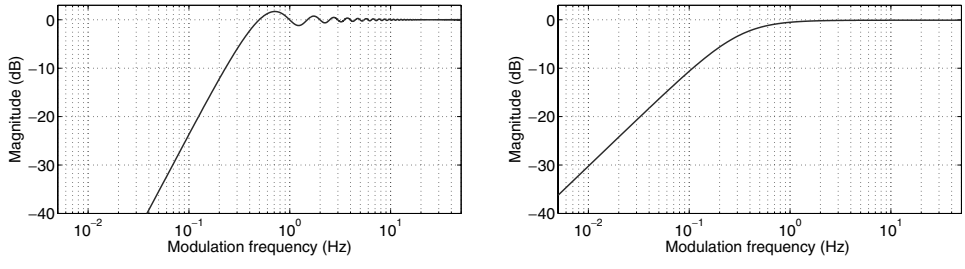
Further insight on the effect of filtering the time sequence of features can be gained by considering the frequency domain representation ((Nadeu *et al.*, 1997)). The Fourier transform of the time sequence of a given feature (i.e. a given cepstral coefficient) can be computed as

$$X(\theta) = \sum_{k} x(k) e^{-jk\theta} \tag{6.94}$$

where θ is usually referred to as the *modulation frequency* and is the frequency domain counterpart of the time index k. The frequency responses of the IIR and FIR implementations of real-time CMN shown in the previous section are depicted in Figure 6.13 for a frame rate of 100 Hz ($T_s = 0.01s$). The main effect of real-time CMN is the attenuation of the slow varying components in the modulation frequency domain. These slow frequency variations are mainly due to variations in the acoustic environment (i.e. speaker variations, background acoustic noise and channel characteristics). The attenuation of the slow varying components of features is also implicitly exploited when using dynamic features in addition to static ones. Augmenting the cepstral vector with the first and second time derivatives of the cepstral coefficients significantly improves the recognition performance by adding dynamic information to the feature vector. A common used definition for these dynamic parameters (see Section 2.3.5) is

$$\Delta c(kT_s) = \frac{\sum_{n=-LD}^{LD} nc((k+n)T_s)}{\sum_{n=-LD}^{LD} n^2} \tag{6.95}$$

$$\Delta\Delta c(kT_s) = \frac{\sum_{n=-LA}^{LA} n\Delta c((k+n)T_s)}{\sum_{n=-LA}^{LA} n^2} \tag{6.96}$$



(a) FIR (b) IIR

Figure 6.13 Modulation frequency response of real-time implementations of CMN. (a) FIR filter implementation with $M = 50$ frames and (b) IIR filter with $\alpha = 0.98$ for a frame rate of 100 Hz ($T_s = 0.01s$)

The computation of Δ coefficients can be rewritten as the operation of a FIR filter whose impulse response is

$$\Delta c = h_D * c \tag{6.97}$$

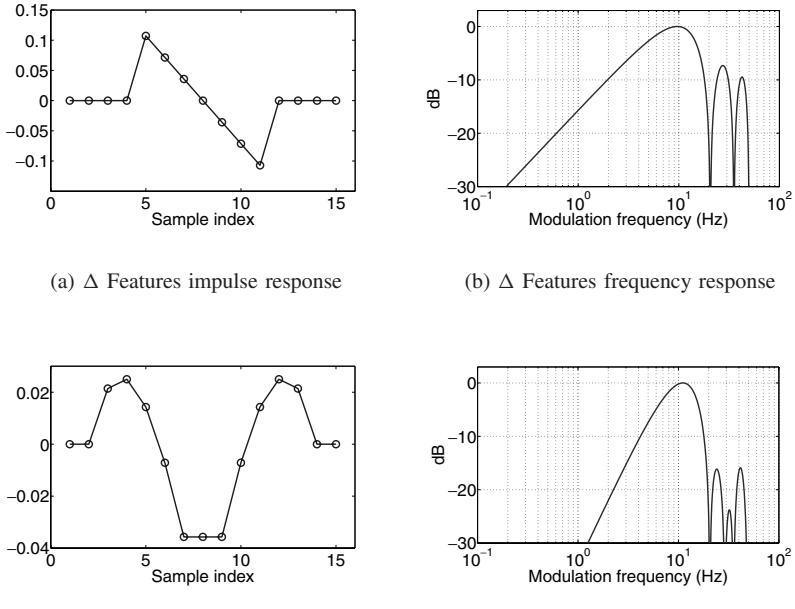
$$h_D(n) = \frac{-n}{\sum_{n=-LD}^{LD} n^2} \;\; ; \;\; -LD \leq n \leq LD \tag{6.98}$$

In a similar way, the $\Delta\Delta$ coefficients can be rewritten as

$$\Delta\Delta c = h_a * \Delta c = h_a * (h_D * c) = (h_a * h_D) * c = h_A * c \tag{6.99}$$

$$h_a(n) = \frac{-n}{\sum_{n=-LA}^{LA} n^2} \;\; ; \;\; -LA \leq n \leq LA \tag{6.100}$$

h_A being the discrete convolution of the impulse response of h_D and h_a. In Figure 6.14 an example is shown for the particular values of $LD = 3$ and $LA = 2$.



(a) Δ Features impulse response

(b) Δ Features frequency response

(c) $\Delta\Delta$ Features impulse response

(d) $\Delta\Delta$ Features frequency response

Figure 6.14 Δ (a) and $\Delta\Delta$ (c) impulse response and corresponding (b) and (d) normalized frequency response for a frame rate of 100 Hz and values $LD = 3$ and $LA = 2$

Dynamic features have a band-pass frequency characteristic. In addition to the attenuation of the low-frequency components of the modulation spectrum, high-frequency components are also attenuated.

6.4.3 RASTA Processing

Filtering the time sequence of features has also been proposed in other approaches like the so-called RASTA ((Hermansky and Morgan, 1994)). In this approach, the time sequence of log-filterbank energies is filtered with an IIR band-pass filter

$$H(z) = z^4 \frac{0.2 + 0.1z^{-1} - 0.1z^{-3} - 0.2z^{-4}}{1 - 0.98z^{-1}} \tag{6.101}$$

whose frequency response is shown in Figure 6.15.

Several other works have studied the influence of filtering the time evolution of features, and have proposed several forms of filters ((Nadeu *et al.*, 1997; Tyagi *et al.*, 2003)). Also linear discriminant analysis has been proposed to design RASTA-like FIR filters ((Avendano *et al.*, 1996)).

The main reason for the improvements come from the fact that the filters enhance the frequency components of time evolution of features (i.e. cepstral coefficients or log-filterbank energies) in the range from 1–10 Hz, which are most relevant to the speech recognition task (Equation 6.101). RASTA processing has been reported to improve recognition performance in presence of both additive and convolutional noises.

6.4.4 Cepstral Mean and Variance Normalization

As discussed in Section 3.4.3, additive noise introduces a nonlinear transformation of the features in the cepstral domain. The main effects of this distortion are a shift of the mean value and a reduction of the variance of each cepstral coefficient. The techniques discussed in the previous sections can deal with the shift of the mean, but are not able to compensate for the variance reduction.

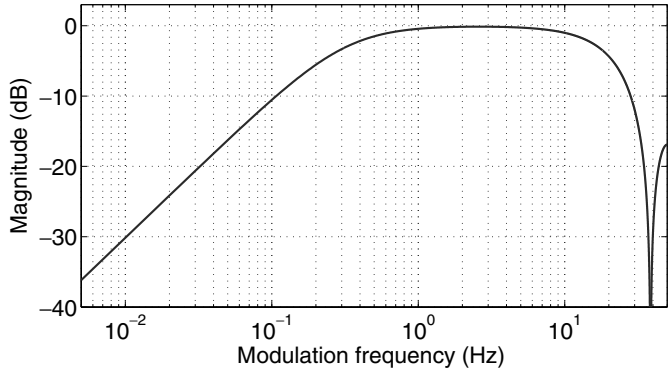


Figure 6.15 Frequency response of the RASTA band-pass filter of Equation (6.101)

A straightforward extension of CMN is to normalize also the variance of the features. In this approach, features are linearly transformed in such a way that the resulting features have zero mean and unity variance. For a given temporal sequence of features (i.e. the temporal sequence of a given cepstral coefficient of a particular utterance), the normalization can be performed in a sentence-by-sentence approach as follows:

$$X = \{x(1), x(2), \ldots, x(N)\} \tag{6.102}$$

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x(n) \qquad \sigma_x^2 = \frac{1}{N}\sum_{n=1}^{N}(x(n) - \bar{x})^2 \tag{6.103}$$

$$\hat{x}(n) = \frac{x(n) - \bar{x}}{\sigma_x} \tag{6.104}$$

Like for CMN, a segmental approach is also necessary for real-time systems. This approach was first proposed by Viikki and Laurila (1998). In this work, the mean and variance of each cepstral coefficient are computed over a short temporal segment around the frame to be normalized. Now again, the mean and variance estimates are time dependent:

$$\bar{x}(n) = \frac{1}{2M+1}\sum_{k=-M}^{M} x(n+k) \tag{6.105}$$

$$\sigma_x^2(n) = \frac{1}{2M+1}\sum_{n=-M}^{M} (x(n+k) - \bar{x}(n))^2 \tag{6.106}$$

This approach provides an automatic gain control (variance normalization) in addition to the low-pass filtering provided by the mean subtraction, which outperforms CMN in additive noise. The optimum length of the temporal segment has been found to be around 1 second for connected-digit speech recognition in noise. Figure 6.16 shows the

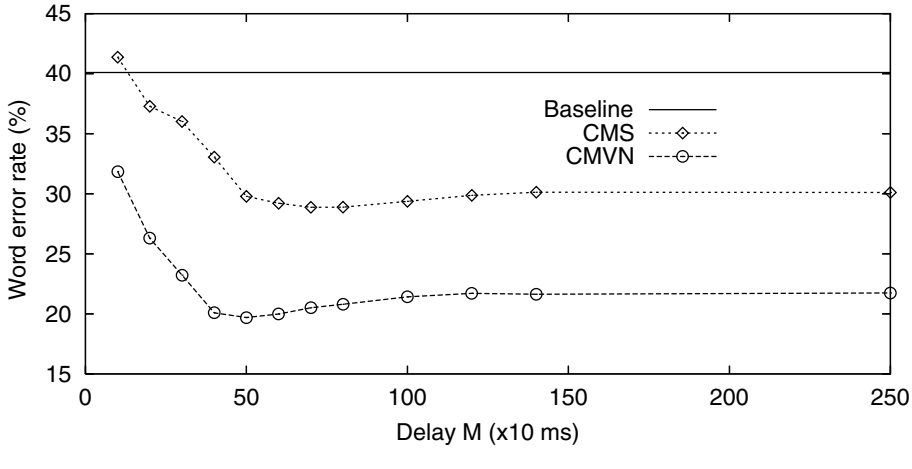


Figure 6.16 Averaged word error rates for the Aurora-2 task and for CMN and CMVN as a function of the computational delay M

performance comparison between CMN and CMVN as a function of the delay M used for the computation of the mean and the variance. These results have been obtained for Aurora-2 connected-digit recognition task published in Segura *et al*. (2004). Results are shown for the baseline system without normalization, and for CMN and CMVN as a function of the delay. The frame rate is 100 Hz, and WERs are averaged for the different noisy conditions specified in the Aurora-2 task.

6.4.5 Nonlinear Feature Normalization

Although CMVN improves CMN in additive noise conditions, the nonlinear effect of the additive noise (see Section 3.4.4) modifies not only the two first moments (mean and variance) of cepstral features but also higher-order moments (skewness, kurtosis, etc.) of the probability distributions. A general linear transformation is able to normalize only the two first moments of a probability distribution. Therefore, a nonlinear transformation is needed to remove higher-order effects of the nonlinear distortion caused by the additive noise in the cepstral domain.

6.4.5.1 Cepstrum Third-order Normalization (CTN)

An extension of the mean and variance normalization has been proposed in Suk *et al*. (1999), where the third-order moment of the probability distributions of cepstral coefficients is normalized in addition to the mean and variance. In this approach, a cubic form is used to formulate a nonlinear transformation

$$x_{CVN}(n) = \frac{x(n) - \mu_x}{\sigma_x} \tag{6.107}$$

$$x_{CTN}(n) = a x_{CVN}^2(n) + b x_{CVN} + c \tag{6.108}$$

where $x_{CVN}(n)$ is the mean- and variance-normalized sequence and a, b and c are selected to normalize the third moment. That is, x_{CTN} must have zero mean, unity variance and zero third-order moment. This implies that b should be nonzero; otherwise, a and c will be zero because of the zero mean and unity variance conditions. Therefore, we can divide Equation (6.108) by b yielding

$$\tilde{x}_{CTN}(n) = x_{CTN}(n)/b = \tilde{a} x_{CVN}^2(n) + \tilde{x}_{CVN} + \tilde{c} \tag{6.109}$$

where $\tilde{a} = a/b$ and $\tilde{c} = c/b$. Now, these values can be found by setting to zero the first and third moment of $\tilde{x}_{CTN}(n)$

$$E\{\tilde{x}(n)\} = E\{\tilde{a} x_{CVN}^2(n) + \tilde{x}_{CVN} + \tilde{c}\} = \tilde{a} + \tilde{c} = 0 \tag{6.110}$$

$$\begin{aligned} E\{\tilde{x}^3(n)\} &= E\{(\tilde{a} x_{CVN}^2(n) + \tilde{x}_{CVN} + \tilde{c})^3\} \\ &= \tilde{a}^3[E\{x_{CVN}^6(n)\} - 3E\{x_{CVN}^4(n)\} + 2] \\ &\quad + \tilde{a}^2[E\{x_{CVN}^5(n)\} - 6E\{x_{CVN}^3(n)\}] \\ &\quad + \tilde{a}[3E\{x_{CVN}^4(n)\} - 3] + E\{x_{CVN}^3(n)\}] = 0 \end{aligned} \tag{6.111}$$

Hence, $\tilde{c} = -\tilde{a}$ from Equation (6.110) and $\tilde{a}$ is obtained from Equation(6.111). Finally, setting the variance of $x_{CTN}(n)$ to unity, b is obtained as follows:

$$b = \frac{1}{\sqrt{Var\{\tilde{x}_{CTN}(n)\}}} \tag{6.112}$$

Hsu and Lee (2004a) have also proposed a variant in which one even moment and one odd moment can be normalized in addition to the mean. By using this kind of approaches, a parametric form is obtained for the nonlinear transformation, but only three moments can be simultaneously normalized.

6.4.5.2 CDF Matching Approach

A different approach to the problem is to transform the probability distribution of the features into a reference distribution. With this approach, not only the first two moments (mean and variance) but also all the higher-order moments of the probability distribution of the features are normalized. As an example, consider that the reference distribution is a normal Gaussian. Transforming the probability distribution of features into this target distribution can be seen as a natural extension of mean and variance normalization. The transformed data will have zero mean, unity variance and all the higher-order moments matching those of a normal Gaussian distribution (in particular the third moment is forced to be zero).

Finding a transformation that maps a given distribution into a desired one is a complex problem that does not have a unique solution in the multidimensional case. But in the one-dimensional case, there exists a unique solution that can be found by simply matching the cumulative distribution functions (CDFs) of the original and transformed random variables.

Consider a given random variable x with PDF $p_x(x)$, and a function $z = T_x(x)$ that transforms it into a new random variable z with PDF $C_z(z)$. If the function is invertible, it can be demonstrated that the CDF $C_x(x)$ and $C_z(z)$ are equal (León-García, 1989)

$$C_x(x) = \int_{-\infty}^{x} p_x(v)dv = \int_{-\infty}^{z=T_x(x)} p_z(v)dv = C_z(z) \tag{6.113}$$

From this relation, it is easy to find the functional form of the transformation in terms of the CDF of the original and transformed random variables

$$C_x(x) = C_z(z) = C_z(T_x(x)) \tag{6.114}$$

$$z = T_x(x) = C_z^{-1}(C_x(x)) \tag{6.115}$$

Note the subscript in the transformation that emphasizes the dependence of the transformation on the CDF of the original random variable. The relation between the PDF of the random variables can be obtained using Equation(6.114) as

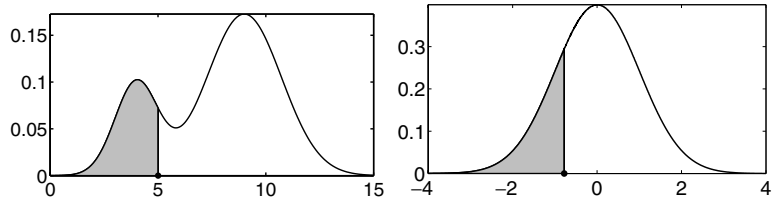
$$p_x(x) = \frac{dC_x(x)}{dx} = \frac{dC_z(T_x(x))}{dx} = p_z(T_x(x))\frac{dT_x(x)}{dx} = p_z(z)\frac{dT_x(x)}{dx} \tag{6.116}$$

As both $C_x(x)$ and $C_z(z)$ are CDFs, they are monotonically increasing and single-valued. Therefore, the resulting transformation is monotonically increasing and nonlinear in general.

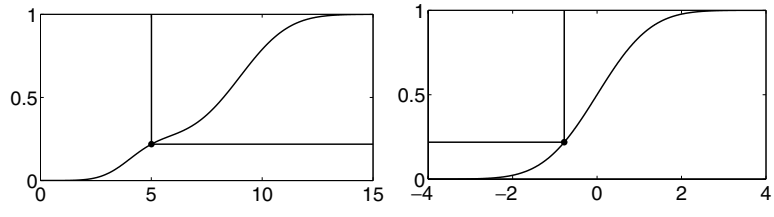
Figure (6.17) shows an example of such a transformation, in which a normal Gaussian is selected as the target distribution. Figures 6.17(a) and 6.17(b) correspond to the PDF of the original x and transformed variables z. The transformed value z_o for a given value x_o can be found in a two-step procedure. First, the CDF of the original value $C_x(x_o)$ is found (Figure 6.17(c)). Then, the value of the transformed variable z_o is selected as the one having this same CDF value of $C_z(z_o)$ (Figure 6.17(d)). Figure 6.17(e) also shows the transformation function. Every point of this function corresponds to a pair of values (x_o, z_o) having the same accumulated probability.

6.4.5.3 Histogram Equalization (HEQ)

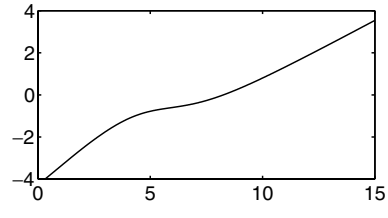
Introduced in Dharanipragada and Padmanabhan (2000) to reduce the mismatch between speaker phones and handset recordings, it was later successfully used for robust speech recognition in noise by de la Torre *et al*. (2005, 2003); de Wet *et al*. (2003); Molau *et al*.



(a) Original PDF $p_x(x)$ (b) Transformed PDF $p_z(z)$



(c) Original CDF $C_x(x)$ (d) Transformed CDF $C_z(z)$)



(e) Transformation $z = T_x(x)$

Figure 6.17 An example of transformation between two given probability distributions

(2002); Segura *et al*. (2004) and also for speaker identification by Pelecanos and Sridharan (2001), and Xiang *et al*. (2002).

In this approach, the CDF matching transformation is obtained by using histograms to model the CDF of both the original and target distributions. In Dharanipragada and Padmanabhan (2000), this technique is used in a feature adaptation approach to reduce the mismatch between handset data, used to train the speech recognition system, and hands-free data used for test. The CDF of the training data is approximated using the cumulative histogram of these training data. Using a set of adaptation utterances, an approximation of the CDF of the test data is obtained using the corresponding cumulative histogram. Finally, a piecewise linear transformation is built by mapping the bin centers of the two cumulative histograms having equal probability values. Relative WER reductions up to a 32.5 % are reported.

In de la Torre *et al*. (2005, 2003), the CDF matching approach is used as a normalization technique in which both training and test data distributions of each cepstral coefficient are transformed into a Gaussian reference. For each utterance, the CDF of each cepstral coefficient is approximated by its cumulative histogram. Histograms are built using 100 equally spaced bins in the range $[\mu - 4\sigma,\ \mu + 4\sigma]$ where μ and σ are the mean and standard deviation of the coefficient to be equalized. Given a set of N observations corresponding to the values of a cepstral coefficient in a given utterance, the PDF is approximated by its histogram as

$$p_x(x \in B_i) = \frac{n_i}{N} \tag{6.117}$$

and the CDF is approximated as

$$C_x(x_i) = C_x(x \in B_i) = \sum_{j=1}^{i} \frac{n_j}{N} \tag{6.118}$$

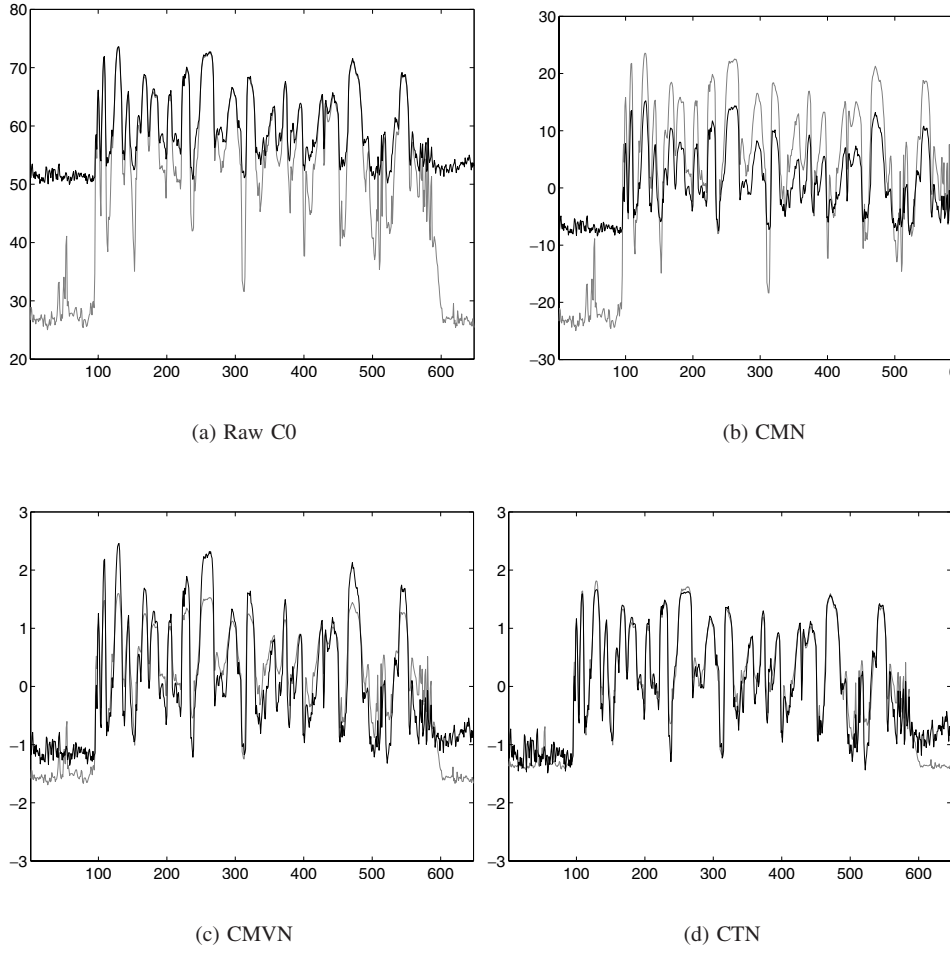
where n_i is the number of observations in the B_i bin. The center x_i of each bin is then transformed using the inverse of the reference CDF (a normal Gaussian in this case)

$$z_i = C_z^{-1}(x_i) \tag{6.119}$$

This way, the set of (x_i, z_i) values defines a piecewise linear approximation of the desired nonlinear transformation. Finally, the transformed values are obtained by linear interpolation between these tabulated values. This approach has been tested using the Aurora-2 evaluation setup. The baseline system uses a 39-component feature vector including the first 12-cepstral coefficients, the log-energy and the corresponding delta and acceleration parameters. HEQ performed better than the baseline system and both CMN and CMVN, giving averaged relative word error reductions of 53.42 % over the baseline, 38.0 % over CMN and 14.1 % over CMVN.

The effects of different feature normalization techniques are illustrated in Figure 6.18. Figure 6.18(a) shows the raw $C0$ cepstral coefficient of a typical utterance. The dashed line corresponds to data obtained in clean conditions; and the solid line is the same

utterance recorded in additive car noise. Figures 6.18(b), 6.18(c) and 6.18(d) correspond to the features obtained after CMN, CMVN and cepstrum third-order normalization (CTN) respectively. Figures 6.18(e) and 6.18(f) correspond to features transformed using HEQ: the first one using a Gaussian reference and the last one using a CDF reference obtained from clean training data.



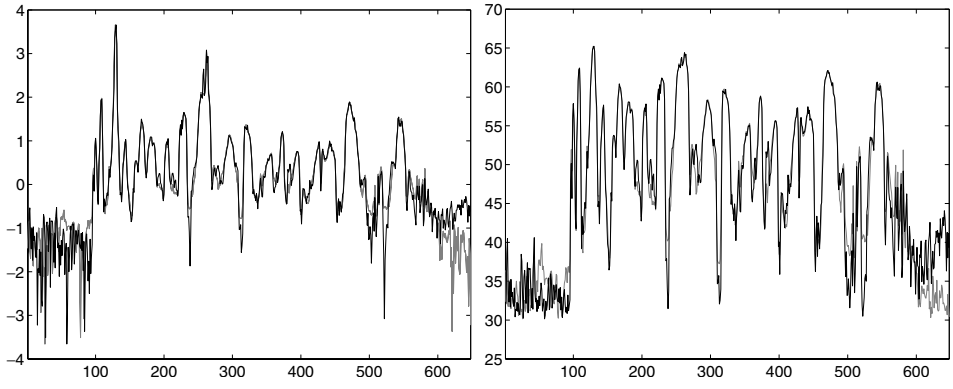
(a) Raw C0

(b) CMN

(c) CMVN

(d) CTN

Figure 6.18 Illustration of the effects of several feature normalization techniques over the C0 cepstral coefficient of a typical utterance. (a) raw C0, (b) mean subtraction, (c) variance normalization, (d) third-order moment normalization, (e) HEQ with a Gaussian reference, and (f) HEQ with a reference obtained from clean training data. Each plot shows the C0 obtained from clean data (gray line) and for the same utterance in additive car noise (black line)



(e) HEQ Gaussian

(f) HEQ clean reference

Figure 6.18 (*continued*)

6.4.5.4 Quantile-based Equalization

Instead of using histograms to model the cumulative probability distributions, the CDF matching transformation can be obtained from the sampling quantiles of the distributions. For a given random variable, the quantile function is defined as the inverse of its CDF

$$Q_x(p) = C_x^{-1}(p) \quad \forall\ p \in [0, 1] \tag{6.120}$$

Quantiles can be efficiently estimated from a set of observations using the order statistics of the data. Consider a set of N observations of a given random variable x

$$\{x_1, x_2, \ldots, x_N\} \tag{6.121}$$

The corresponding order statistics are obtained by simply sorting the data in increasing order

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)} \tag{6.122}$$

An estimation of the quantile $Q_x(p)$ for a given probability p can be obtained from the above order statistics as

$$\hat{Q}_x(p) = \begin{cases} (1-f)x_{(k)} + f x_{(k+1)}\ , & 1 \leq k \leq N-1 \\ x_{(k+1)}\ , & k = N \end{cases} \tag{6.123}$$

where k and f are the integer and fractional parts of pN.

An approximation of the transformation function can be obtained from a reduced set of quantiles as follows: Consider a set of N_Q probability values uniformly distributed in the interval [0, 1],

$$p_r = \left(\frac{r - 0.5}{N_Q}\right)\ , \qquad r = 1, 2, \ldots, N_Q \tag{6.124}$$

If we compute the quantile values for this set of probabilities, each pair of values $(\hat{Q}_x(p_r), \hat{Q}_y(p_r))$ obtained from two random variables x and y correspond to a point of the transformation that maps $C_x(x)$ into $C_y(y)$.

In Segura *et al*. (2004), this approach was used to transform features to match a normal Gaussian distribution. A fixed set of probability values are first defined as in Equation (6.124), and the quantiles of the reference distribution are computed using the quantile function of the Gaussian distribution. For both training and test utterances, the corresponding sampling quantiles are computed for each feature using Equation (6.123), and linear interpolation between pairs of quantiles is used to perform a piecewise linear transformation. This transformation is applied in the cepstral domain, and the reported performance for a reduced set of 30 quantiles is virtually the same as the one obtained for HEQ. This approach can also be used with non-Gaussian reference distributions. For example, the reference distribution can be obtained from a set of quantiles computed from training data.

In Hilgher and Ney (2001) a similar approach is used. Instead of using linear interpolation between sampling quantiles of cepstral coefficients, a power transformation is used to transform log-energies at the output of the analysis filterbank.

Finally, on-line versions for both HEQ and quantile-based equalization have been proposed in Segura *et al*. (2004) and Hilgher *et al*. (2002) using a segmental implementation similar to the one used for real-time CMN and CMVN (Sections 6.4.1 and 6.4.4).

7

Standards for Distributed Speech Recognition

7.1 Introduction

As mentioned in Chapter 1, the RSR standardization activities have been concentrated in DSR, and have been carried out by the STQ ETSI Aurora working group. This working group has developed four standards that specify the *feature extraction and compression* algorithms required for DSR. These are (in issuing order):

- ETSI ES 201 108 (ETSI, 2003a): basic MFCC-based front end (FE). First version in February 2000.
- ETSI ES 202 050 (ETSI, 2003b): advanced front end (AFE). It introduces noise reduction and feature normalization algorithms to increase the robustness against acoustic degradation. First version in October 2002.
- ETSI ES 202 211 (ETSI, 2001): extended front end (XFE). It adds pitch and voicing information to the basic MFCC-based parametrization in order to allow speech reconstruction or improved recognition of tonal languages such as Mandarin, Cantonese and Thai. First version in November 2003.
- ETSI ES 202 212 (ETSI, 2003c): extended advanced front end (XAFE). It gathers the improvements of the AFE and XFE FEs. First version in November 2003.

The implementation of the above FEs over mobile and IP networks have been treated in another two sets of public documents:

- **Implementation over mobile networks**. The FE standard was originally developed to be implemented over circuit-switched channels. The other three FEs still consider this possibility. However, 3GPP carried out in 2002 a study on the feasibility of SES, which considered their implementation over packet channels rather than circuit channels (3GPP, 2002). Therefore, the payload formats described in the documents of the next item (implementation over IP networks) can be applied. 3GPP also carried out a performance evaluation of the XAFE FE, which was published in 2004 (3GPP, 2004a).

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

As a result, the XAFE was selected as the default codec for SES (3GPP, 2005) and a fixed-point implementation of the XAFE was developed (3GPP, 2004b).

- **Implementation over IP networks**. As explained in Chapter 3, the appropriate transport-layer protocols for RSR are UDP/RTP. Thus, the IETF has published two documents specifying the RTP payload format for the FE (Xie, 2003) and for the other three FEs (Xie and Pearce, 2005).

Each Aurora standard extracts a different set of features. However, each feature set can be built from the following three subsets:

1. Basic feature vector: $\mathbf{x} = (c(1), \ldots, c(12), c(0), \log E)^t$. It includes MFCC(0–12) and the log-energy. Used by the four standards.
2. *VAD* flag. It is quantized and encoded using 1 bit (1/0 for speech/non-speech). Used only by the advanced standards.
3. Extension features: $\mathbf{x}_{ex}$ = (pitch,voicing class (VC)). Used only by the extended standards.

The features computed and transmitted by each standard are specified in Table 7.1.

A general and joint block diagram of the four front(back) ends is shown in Figure 7.1. The basic blocks shared by the four standards are indicated with thick lines. The thin

Table 7.1 Acronyms and feature vectors of the four Aurora standards

	Non-Extended		Extended	
Non-Advanced	FE	$\mathbf{x}$	XFE	$(\mathbf{x},\mathbf{x}_{ex})$
Advanced	AFE	$(\mathbf{x},VAD)$	XAFE	$(\mathbf{x},VAD,\mathbf{x}_{ex})$

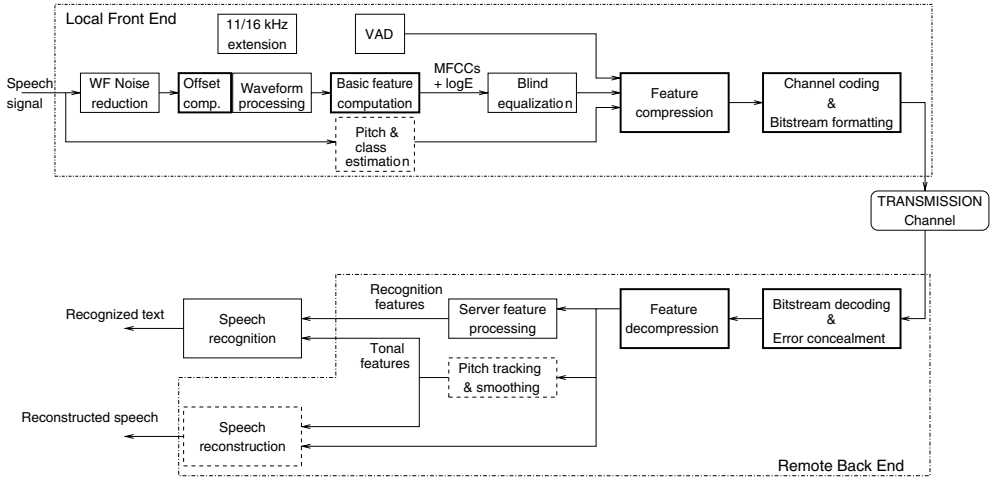


Figure 7.1 General block diagram of a DSR system using the FE, AFE, XFE or XAFE standards. Blocks shared by the four standards are indicated by thick lines. Thin and dashed lines indicate blocks employed only by the advanced and extended standards, respectively

and dashed lines indicate blocks exclusive to the (A)dvanced and e(X)tended standards, respectively. As we go along this chapter, we will study the functional blocks of the four FE standards (FE, AFE, XFE and XAFE). This is an appropriate way to approach them since they share many of these blocks. It must be pointed out that the goal of the next sections is not to reproduce the ETSI standards, but to provide a comprehensive approach to them on the basis of the material given in the previous chapters and the appendices at the end of the book. The exact implementation details can be extracted from the public ETSI documents. The block corresponding to speech reconstruction is omitted, since it is beyond the scope of this book. The recognition performance of the ETSI DSR standards has been evaluated in different publications (3GPP, 2004a; Hirsh, 2002; Hirsh and Pearce, 2000; Kelleher *et al*., 2002; Sorin *et al*., 2004).

This chapter is organized according to Figure 7.1. First, there is a section devoted to the preprocessing blocks, that is, noise reduction, offset compensation and waveform processing. Then, in Section 7.3 we deal with the feature extraction blocks, including basic feature vector computation, AFE/XAFE extension to 16 kHz, blind equalization, voice activity detection and pitch and voicing class estimation. Section 7.4 is devoted to the quantization of the different features and their corresponding channel encoding, and finishes with the description of the bitstream and payload formats. The chapter ends with a section describing the server operations (decoding, decompression, error detection and concealment, server feature processing and pitch tracking and smoothing (PTS).

7.2 Signal Preprocessing

The first element of a DSR system is the speech signal acquisition. Regarding the input audio parts, the four ETSI standards assume that the DSR terminals accomplish it with the specifications of ETSI EN 300 903 (ETSI, 1999b) and warn that the recognition performance could be reduced otherwise. The allowed sampling frequencies are $F_s =$ 8, 11, 16 kHz. The resulting digitized signal is $s_{in}(n)$.

As we can see in Figure 7.1, there is a big difference between the signal preprocessing carried out in the advanced standards and in the nonadvanced standards, since the first ones introduce noise reduction blocks that are not present in the second ones. In particular, the advanced standards introduce two noise reduction techniques based on a Wiener filtering and an SNR-dependent waveform processing, respectively, while the nonadvanced standards apply only an offset compensation (also present in the advanced ones). All these techniques are studied in the following subsections.

7.2.1 Two-stage Mel-warped Wiener Filtering

This block is exclusive for the advanced FEs. In principle, a sampling frequency of $F_s = 8$ KHz is assumed. The extension to 11 and 16 kHz is specified in the following section, devoted to feature extraction. The WF block is the main noise reduction technique applied in the Aurora standards and is also very time-consuming. An alternative and more efficient frequency domain implementation has been proposed in Li *et al.* (2004).

The original two-stage mel-warped WF technique was proposed by Argawal and Cheng (1999). A diagram of this technique is shown in Figure 7.2. Its basic principle is a

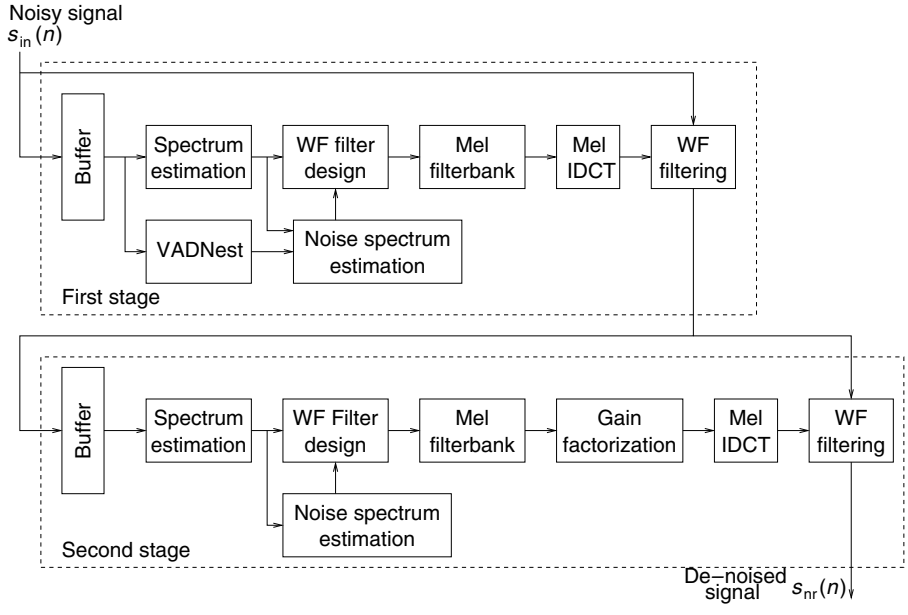


Figure 7.2 General block diagram of the two-stage mel-warped Wiener filtering for noise reduction

double WF filtering (the output of the first stage is the input to the second one). The first stage de-noises the input signal using a noise spectrum estimate obtained during the silence segments by means of a VAD detector VAD for noise estimation(VADNest). Then, the second stage tries to remove the residual noise, which is basically due to the inaccurate spectrum estimates employed to build the WF filter of the first stage. Both stages are almost identical except for the noise spectrum estimation applied and the gain factorization introduced in the second stage (described later). In every stage, the whole process of obtaining the WF filter is carried out in the frequency domain, although the final filtering is applied in the time domain since this is required by the subsequent waveform processing block. The next subsections are devoted to the description of the different blocks used in each stage.

7.2.1.1 Buffering and Spectrum Estimation

In order to be coherent with the notation introduced in Chapter 6, we will denote the input noisy signal of every stage as $y(k)$ ($y(k) = s_{in}(k)$ in the first stage), and consider that the original clean signal $x(k)$ was corrupted by an additive noise $n(k)$ (i.e. $y(k) = x(k) + n(k)$), which is obviously different in each stage. The WF filter is computed for every $M = 80$ samples, using frames of length $N_{in} = 200$ samples. In order to obtain the power spectrum of the input signal, an FFT of $N_{FFT} = 256$ points is applied to each frame of $y(k)$ (previously windowed by a Hanning window). The spectrum obtained for

frame number t ($|Y_t(i)|^2$; $i = 0, \ldots, N_{FFT} - 1$) is smoothed in frequency

$$|\tilde{Y}_t(i)|^2 = \frac{|Y_t(2i)|^2 + |Y_t(2i+1)|^2}{2} \qquad (0 \le i < N_{FFT}/4) \tag{7.1}$$

$$|\tilde{Y}_t(N_{FFT}/4)|^2 = |Y_t(N_{FFT}/2)|^2 \tag{7.2}$$

and also in time

$$\overline{|Y_t(i)|^2} = \frac{\tilde{Y}_t(i) - \tilde{Y}_{t-1}(i)}{2} \qquad (0 \le i < N_{SPEC} = N_{FFT}/4 + 1) \tag{7.3}$$

For the sake of simplicity and to be coherent with the notation used in Chapter 6, we use the frequency variable f ($0 \le f \le F_s/2$) instead of the frequency bin i ($0 \le i < N_{SPEC}$) in the following subsections.

7.2.1.2 Wiener Filter Design

The WF filter design of every stage is carried out in two iterations as depicted in Figure 7.3. In the first iteration, we obtain the first WF filter $H_{1,t}(f)$ which is employed to obtain a first estimate of the clean spectrum $|\hat{X}_{2,t}(f)|$, which is in turn used to obtain the final WF filter $H_{2,t}(f)$. On filtering $|\hat{X}_{2,t}(f)|$ with this final filter, we obtain the final estimate $|\hat{X}_{3,t}(f)|$ of the noiseless signal spectrum. In both iterations ($i = 1, 2$), the WF filter is computed as (see Equations (6.30) and (6.31)),

$$H_{i,t}(f) = \frac{|\hat{X}_{i,t}(f)|}{|\hat{X}_{i,t}(f)| + |\hat{N}_t(f)|} = \frac{\sqrt{\xi_{i,t}(f)}}{1 + \sqrt{\xi_{i,t}(f)}} \tag{7.4}$$

where $|\hat{N}_t(f)|$ is an estimate of the noise spectrum at time t (described in the next subsection) and $\xi_{i,t}(f) = |\hat{X}_{i,t}(f)|^2/|\hat{N}_t(f)|^2$ is the corresponding estimate of the a priori SNR.

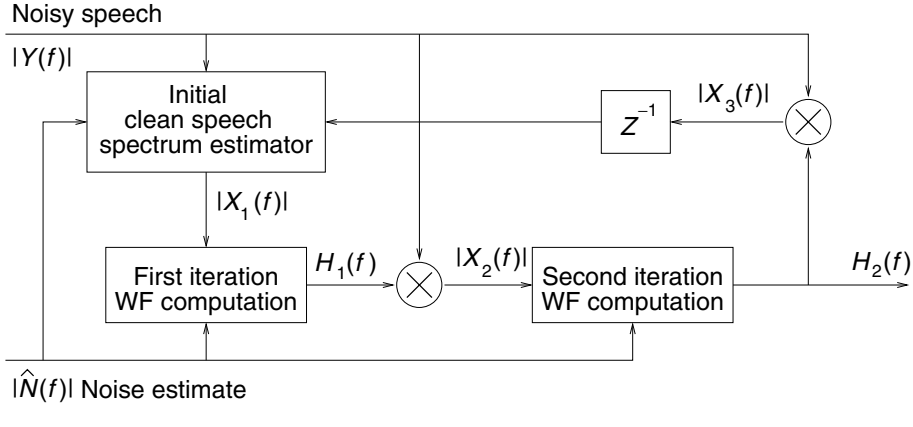


Figure 7.3 Simplified diagram of the Wiener filter design block

To obtain the first iteration WF filter, an initial estimate $|\hat{X}_{1,t}(f)|$ of the clean speech spectrum is required, which is obtained by spectral subtraction (see Equation (6.43)) and smoothing,

$$|\hat{X}_{1,t}(f)| = \beta|\hat{X}_{3,t-1}(f)| + (1-\beta)\max(\overline{|Y_t(f)|} - |\hat{N}_t(f)|, 0) \tag{7.5}$$

where $\beta = 0.98$. Then, we can obtain the first estimate of the clean speech spectrum as $|\hat{X}_{2,t}(f)| = H_{1,t}(f)\overline{|Y_t(f)|}$. The second iteration WF filter $H_{2,t}(f)$ is obtained in the same way (applying Equation (7.4) again), but by using $|\hat{X}_{2,t}(f)|$ as the clean speech spectrum estimate. Finally, we obtain $|\hat{X}_{3,t}(f)| = H_{2,t}(f)|\tilde{Y}_t(f)|$, which is used in the next frame. Note that, in this second iteration, the less smoothed estimate $|\tilde{Y}_t(f)|$ of the noisy signal spectrum is used(only decimated in frequency).

It is worthwhile highlighting that the WF filter computed in Equation (7.4) uses magnitude spectra rather than power spectra as the WF theory introduced in Chapter 6 indicates. Since Wiener filtering can be interpreted as a form of spectral subtraction, we can consider that the WF filtering developed here is somewhat a magnitude spectral subtraction technique (Vaseghi, 2000). This fact is exploited later for gain normalization.

7.2.1.3 Noise Spectrum Estimation and VAD for Noise Reduction (VADNest)

In order to compute the WF filters of both stages, we have assumed that an estimate of the noise spectrum is available. In the first stage, those frames marked as silence by the VADNest detector are used to compute this spectrum by means of a recursive smoothing filter (see Equation (6.35)),

$$|\hat{N}_t(f)| = \lambda_t|\hat{N}_t - 1(f)| + (1-\lambda_t)\overline{|Y_t(f)|} \tag{7.6}$$

The filtering is initialized with and thresholded by a small value $EPS = \exp(-10)$. During speech frames, $|\hat{N}_t(f)| = |\hat{N}_{t-1}(f)|$. The forgetting factor λ_t is $(1-1/t)$ for the first 100 frames and 0.99 for the subsequent ones. The VADNest flag is generated by a simple full-band-energy VAD like the one already described in Chapter 6 (section 6.3.1).

In the second stage, the noise spectrum estimate is updated every frame, independently of its type (noise/speech). While for the first 10 frames the same estimation is applied as in Equation (7.6) (although using power spectra), for the subsequent ones the following smoothing expression is used:

$$|\hat{N}_t(f)|^2 = \gamma|\hat{N}_{t-1}(f)|^2 + (1-\gamma)|\tilde{N}_t(f)|^2 \tag{7.7}$$

where $\gamma = 0.9$ and $|\tilde{N}_t(f)|^2$ is an initial estimate of the noise spectrum at time t obtained by filtering $\overline{|Y_t(f)|}^2$

$$|\tilde{N}_t(f)|^2 = B_t(f)\overline{|Y_t(f)|}^2 \tag{7.8}$$

with a filter $B_t(f)$ defined by the following Equations:

$$B_t(f) = W_t(f)F_t(f) \tag{7.9}$$

$$W_t(f) = \frac{1}{1+\gamma_t(f)} \tag{7.10}$$

$$F_t(f) = \left(1 + \frac{1}{1+0.1\gamma_t(f)}\right) \tag{7.11}$$

$$\gamma_t(f) = \frac{\overline{|Y_t(f)|}^2}{|\hat{N}_{t-1}(f)|^2} \tag{7.12}$$

where $\gamma_t(f)$ is somewhat a measure of the a posteriori SNR. The filter is obtained as the product of two factors. The first one ($W_t(f)$) has the form of a Wiener filter to estimate the noise spectrum, but substituting the a priori SNR $\xi_t(f)$ by the a posteriori SNR $\gamma_t(f)$. The second factor ($F_t(f)$) forces $B_t(f)$ to be in the range [0, 1]. When the noise is predominant, $\gamma_t(f)$ tends to 1 and $B_t(f)$ is also close to 1 so that there is an important contribution of the input signal spectrum to the noise spectrum update. In the opposite case, $\gamma_t(f)$ is large (and close to $\xi_t(f)$) and $B_t(f)$ tends to zero, so that there is no contribution of the input signal spectrum to the noise spectrum update. Figure 7.4 shows the magnitude of $B_t(f)$ versus the values of $\gamma_t(f)$. The same thresholding as for the first stage is applied to the estimate obtained from Equation (7.7).

7.2.1.4 Mel Filterbank

It is well known that the introduction of perceptual criteria improves both speech enhancement and recognition systems. This is the reason the Wiener filter previously obtained is transformed to a perceptual domain. This is carried out by applying a mel-scaled triangular filterbank as that of Figure 2.5 (also used in the feature extraction section) to the frequency response $H_{2,t}(f)$ of the WF filter obtained in each stage. The filterbank uses $K_{FB} = 23$ frequency bands, although the marginal bands corresponding to frequencies 0 and $F_s/2$ are also considered, obtaining, in total, $K_{FB} + 2 = 25$ filterbank outputs representing the WF filter frequency response. The filterbank outputs are computed through an expression similar to Equation(2.9), although introducing a normalization with respect to

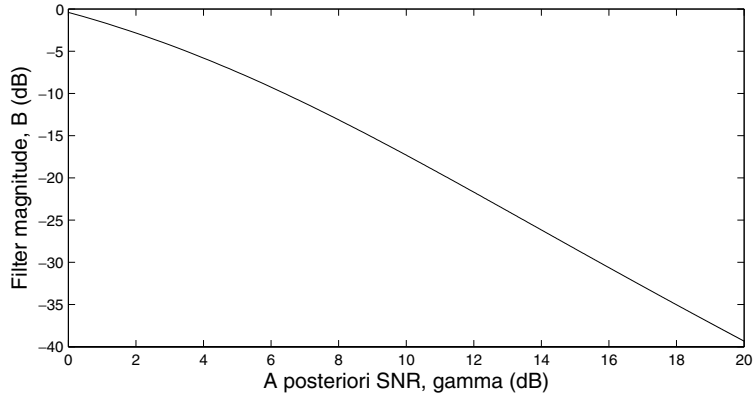


Figure 7.4 Magnitude of $B_t(f)$ versus a posteriori SNR ($\gamma_t(f)$)

the weight of each filter in the bank,

$$H_{2,t}^{(mel)}(f_k) = \frac{\sum_f w(f_k, f) H_{2,t}(f)}{\sum_f w(f_k, f)} \qquad (0 \le k \le K_{FB} + 1) \tag{7.13}$$

where $w(f_k, f)$ represents the kth filterbank channel (with central frequency f_k) and f takes N_{SPEC} frequency values in the linear-frequency domain.

7.2.1.5 Gain Factorization (Only in Second Stage)

As was previously mentioned, the WF filtering can also be interpreted as a magnitude spectral subtraction technique,

$$|\hat{X}(f)| = |Y(f)| - |\hat{N}(f)| = \left(1 - \frac{|\hat{N}(f)|}{|Y(f)|}\right)|Y(f)| \tag{7.14}$$

where the equivalent WF filter is (see Equation (6.45)),

$$H(f) = 1 - \frac{|\hat{N}(f)|}{|Y(f)|} = \frac{|Y(f)| - |\hat{N}(f)|}{|Y(f)|} \approx \frac{|\hat{X}(f)|}{|\hat{X}(f)| + |\hat{N}(f)|} \tag{7.15}$$

where we need an initial estimate of the clean speech spectrum $|\hat{X}(f)|$.

The amount of subtracted noise can be controlled by a factor α (see Equation (6.47)),

$$|\hat{X}(f)| = |Y(f)| - \alpha|\hat{N}(f)| \tag{7.16}$$

Now, the equivalent WF filter is,

$$H^{(GF)}(f) = 1 - \alpha\frac{|\hat{N}(f)|}{|Y(f)|} = (1 - \alpha) + \alpha\left(1 - \frac{|\hat{N}(f)|}{|Y(f)|}\right) = (1 - \alpha) + \alpha H(f) \tag{7.17}$$

This modification is applied to $H_{2,t}^{(mel)}(f_k)$, obtaining a modified filter $H_{2,t}^{(melGF)}(f_k)$. While in common spectral subtraction α is usually greater that 1 in order to introduce oversubtraction, in the Aurora standards it varies from 0.1 (during speech frames) to 0.8 (during pure noise frames). That is, the level of aggression applied by the WF filter is higher during pure noise frames.

Factor α is computed for every frame using two SNR measures: a smoothed SNR (SNR_{aver}) computed using $|\hat{X}_{3,t}(f)|$ and $|\hat{N}_{2,t}(f)|$ over the last three frames, and a measure $SNR_{\text{low\_track}}$ of the lowest values of SNR_{aver} tracked during the previous frames. Thus, when SNR_{aver} is small with respect to $SNR_{\text{low\_track}}$ then α is increased, and it is decreased otherwise (although always maintained between 0.1 and 0.8).

7.2.1.6 Mel-IDCT and Application of the WF Filter

The WF filters of both stages are applied in the time domain by convolution. Therefore, we must obtain their corresponding impulse responses. Let us describe now how the Aurora standards do this. Given a filter frequency response $H(f)$ real and even, we can obtain its impulse response as

$$h(n) = \frac{2}{F_s} \int_0^{F_s/2} H(f) \cos\left(\frac{2\pi f n}{F_s}\right) df \tag{7.18}$$

which can be approximated (except by a gain factor) by using a finite set of frequencies $\{f_k; k = 0, \ldots, L\}$ as

$$h(n) \approx \frac{1}{F_s} \sum_{k=0}^{L} H(f_k) \cos\left(\frac{2\pi f_k n}{F_s}\right) \Delta f_k \tag{7.19}$$

This expression is referred to as *mel-IDCT* in the standards and can directly be applied to our mel-warped WF filter (i.e. $H(f_k) = H_{2,t}^{(melGF)}(f_k)$) just by considering the set of central frequencies f_k corresponding to the filters of the mel-filterbank employed for mel warping. These frequencies can be obtained as the weighted average of every band

$$f_k = \frac{\sum_f w(f_k, f) f}{\sum_f w(f_k, f)} \qquad (0 < k < L = K_{FB} + 1) \tag{7.20}$$

Also, the frequencies for $k = 0$ and $k = L$ are assigned as $f_0 = 0$ and $f_L = F_s/2$, respectively. The frequency increments Δf_k are computed as

$$\Delta f_k = f_{k+1} - f_{k-1} \qquad (0 < k < L) \tag{7.21}$$

Additionally, $\Delta f_0 = (f_1 - f_0)$ and $\Delta f_L = (f_L - f_{L-1})$.

From Equation (7.19), we derive the noncausal FIR filter $h(n)$ $(-L \leq n \leq L)$ of length $2L + 1 = 2(K_{FB} + 1) + 1$ corresponding to the frequency response of the WF filter $(H_{2,t}^{(mel)}(f_k)$ in the first stage and $H_{2,t}^{(melGF)}(f_k)$ in the second one). Equation (7.19) is applied for $n = 0, \ldots, L = K_{FB} + 1$, and the filter coefficients corresponding to $n = -L, \ldots, -1$ are easily derived by taking into account the fact that the resulting impulse response $h(n)$ must be even (since its frequency response is real and even). The filter is further truncated using a Hanning window of length $FL = 17$ and centered at $n = 0$. The objective of this truncation is to obtain a smoother frequency response. Finally, the resulting filter $\tilde{h}(n)$ is convolved with the input signal $y(n)$ of the considered stage to obtain the final denoised signal,

$$\hat{x}(n) = \sum_{i=-(FL-1)/2}^{(FL-1)/2} \tilde{h}(i) y(n - i) \tag{7.22}$$

The filtering is applied only to the first $M = 80$ samples of the frame (those not overlapped with the following frame). Note that the input to the first stage is $y(n) = s_{in}(n)$ and the output from the second stage is $\hat{x}(n) = s_{nr}(n)$, which is the signal used by the subsequent stages.

7.2.2 Offset Compensation and Waveform Processing

The four standards share an offset compensation which removes the DC component by means of a notch filter

$$s_{of}(n) = s_{nr}(n) - s_{nr}(n-1) + F s_{of}(n-1) \tag{7.23}$$

where $F = 0.999$ for FE/XFE and $F = (1 - 1/1024)$ for AFE/XAFE, and $s_{nr}(n) = s_{in}(n)$ for FE/XFE.

While the offset-compensated signal $s_{of}(n)$ is directly passed to the feature extraction section in the FE/XFE standards, it is previously submitted to an extra noise reduction stage in the case of AFE/XAFE. This additional stage corresponds to the SNR-dependent waveform processing (SWP) proposed in Macho and Cheng (2001). This technique is applied in the time domain and used as a complementary processing of the WF filtering of the previous subsection. The underlying idea of the SWP processing is that the signal energy within a pitch period of a voiced sound is variable, so that it reaches its highest value while the glottis is closed, and then decreases very quickly. Since the noise energy is quite constant during a pitch period, we can consider that the SNR is variable within that interval. Therefore, if we can increase the energy of high-energy periods and decrease that of the low-energy ones, the overall SNR is increased, thereby obtaining an improved recognition performance.

The first step of the SWP block consists in obtaining a smoothed energy contour. The instantaneous signal energy can be obtained by applying the discrete version of the Teager operator proposed in (Kaiser, 1990),

$$E_{Teag}(n) = |s_{of}^2(n) - s_{of}(n-1)s_{of}(n+1)| \tag{7.24}$$

These instantaneous energies are obtained for every frame of length $N_{in} = 200$. In order to apply the above operator at the frame limits ($n = 0, N_{in} - 1$), the limit samples $s_{of}(0)$ and $s_{of}(N_{in} - 1)$ are repeated backward and forward. The smoothed energy contour $E_{Teag\_Smooth}(n)$ is finally obtained as the mean energy of the interval $[n-4, n+4]$ (the required energies outside the frame are obtained again by repetition of the energies at the frame limits). For voiced sounds, we can expect that the energy contour will have a quasiperiodic shape. On the other hand, in the case of unvoiced sound the energy contour will be more or less flat and random.

The SWP processing works over periods between consecutive energy peaks. Thus, the next step is to locate these energy peaks. This is carried out by a peak-picking strategy that finds the N_{MAX} maxima in the frame, which are expected to be separated between 25 and 80 samples (the peak search starts by locating the global maximum in the frame). The high-energy portions of the frame are located by means of a weighting function $w(n)$, which is a sequence of rectangular unit windows. Each window starts just before each peak (four samples) and has a length equal to 80 % of the period between the corresponding

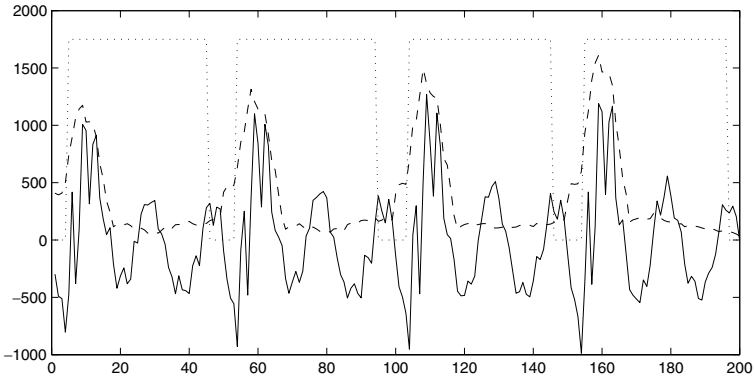


Figure 7.5 Example of a voiced signal (solid), its smoothed energy contour (dashed) and its corresponding SWP weighting function (dotted). The scales are modified for a suitable display

peaks. For the last high-energy period of the frame, the length of the previous window is repeated. The low-energy portions are selected by $1 - w(n)$. The whole process for the selection of high and low SNR portions is depicted in Figure 7.5. Finally, the high- and low-energy portions are amplified and attenuated as follows:

$$s_{swp}(n) = \gamma w(n) s_{of}(n) + \epsilon \left(1 - w(n)\right) s_{of}(n) \tag{7.25}$$

where $\gamma = 1.2$ and $\epsilon = 0.8$. In Macho and Cheng (2001), ϵ is 0.8 again and γ is taken so that the frame energy is preserved. Therefore, the SWP processing of the AFE/XAFE standards does not preserve that energy.

In the following, we will refer to the output signal of the whole preprocessing stage as $s_{pre}(n)$ $(n = 0, \ldots, N_{in} - 1)$, which coincides with $s_{swp}(n)$ in the case of the AFE/XAFE standards and with $s_{of}(n)$ in the case of the FE/XFE standards.

7.3 Feature Extraction

7.3.1 Computation of the Basic Features

The four standards extract the same basic features from the output signal $s_{pre}(n)$ of the previous preprocessing stage, that is, MFCC(1–12), MFCC(0) and the log-energy. The corresponding feature extraction procedures are basically the same, although they differ in how the different sampling frequencies (F_s) are taken into account. FE and XFE use different frame sizes: $N_{in} = 200, 256, 400$ samples (25, 23.27, 25 ms) for $F_s = 8, 11, 16$ kHz, respectively. The frame shift is always 10 ms ($M = 80, 110, 160$ samples for $F_s = 8, 11, 16$ kHz, respectively). However, in AFE and XAFE, a basic sampling frequency of 8 kHz is assumed with a fixed frame size of $N_{in} = 200$ samples and a fixed shift of $M = 80$. The processing of the corresponding 0–4 kHz band is similar to that of the FE/XFE standards for $F_s = 8$ kHz. In the case of $F_s = 11$ kHz, the signal is decimated to $F_s = 8$ kHz and the processing is the same as for this former sampling frequency. In the case of $F_s = 16$ kHz, an extension for the processing of the new band of 4–8 kHz is included, which is detailed in the following subsection. A summary of these

Table 7.2 Frame size (N_{in}), Frame shift (M) and FFT size (N_{FFT}) for the different sampling frequencies (F_s) and for the different standards

Standard	Frame size (N_{in})			Frame shift (M)			FFT size (N_{FFT})		
FE/XFE	200	256	400	80	110	160	256	256	512
(F_s kHz)	(8)	(11)	(16)	(8)	(11)	(16)	(8)	(11)	(16)
AFE/XAFE (0–4 kHz band)		200			80			256	

feature extraction parameters for the different sampling frequencies and for the different standards is given in Table 7.2.

The log-energy is always computed as

$$\log E = \log \sum_{n=0}^{N_{in}} s_{pre}(n)^2 \tag{7.26}$$

with a threshold of -50. To compute the MFCCs, the input signal $s_{pre}(n)$ is preemphasized (Equation (2.4), with $\mu = 0.97$ for FE/XFE and $\mu = 0.9$ for AFE/XAFE) and Hamming-windowed, obtaining a signal $s_w(n)$ $(n = 0, \ldots, N_{in} - 1)$.

In order to obtain the filterbank outputs, first the Fourier transform is computed $X_w(i) = FFT[s_w(n)]$ $(i = 0, \ldots, N_{FFT})$ of the windowed signal. The values of N_{FFT} are given in Table 7.2. Zero-padding is applied if $N_{in} < N_{FFT}$. A mel-scaled triangular filterbank, as the one depicted in Figure 2.5, with $K_{FB} = 23$ filters covering the range from 64 Hz (frequencies below 64 Hz are discarded) to $F_s/2$, is applied to $|X_w(i)|$ (FE/XFE) or to $|X_w(i)|^2$ (AFE/XAFE) $(i = 0, \ldots, N_{FFT}/2)$, according to Equation (2.9), obtaining the filterbank outputs. The MFCC coefficients $c(n)$ $(n = 0, 1, \ldots, 12)$ are finally obtained by applying the DCT to the log-outputs $(S_{FB}(k);\ 1 \leq k \leq K_{FB})$ of the filterbank (see Equation (2.12)).

7.3.2 AFE/XAFE Sampling Frequency Extension to 16 kHz

The extension to 16 kHz tries to exploit the new frequency components (4–8 kHz) to improve the recognition performance. The AFE/XAFE standards do this without modifying the processing of the 0–4 kHz frequency band already described for $F_s = 8, 11$ kHz. This is possible through the subband analysis depicted in Figure 7.6. It uses a 2-channel filterbank implemented with a pair of 118-tap quadrature mirror filter (QMF) filters, which split the 0–8 kHz band into two subbands (0–4 kHz and 4–8 kHz). Details of this type of subband analysis can be found in section 4.2.1.2. The processing of the LP band coincides with the one described in the previous subsections. The HP band is also processed through a mel scale triangular filterbank, with the idea of appending the resulting high-frequency band (HFB) filterbank outputs to those already obtained for the low-frequency band (LFB). The processing of the HFB band also includes noise reduction based on spectral subtraction (SS) and an HFB coding and decoding.

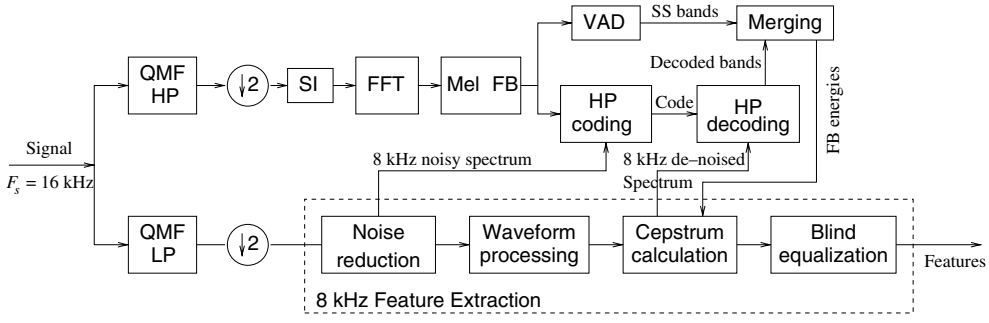


Figure 7.6 Block diagram of the AFE/XAFE extension to $F_s = 16$ kHz

7.3.2.1 Computation of the HFB Filterbank Outputs

The HP QMF filter and the sampling frequency compressor shift the higher band to 0–4 kHz. The resulting spectrum is inverted with respect to the original one. The spectrum inversion (SI) block turns it back by multiplying the signal by $(-1)^n$ (n is the sample index) (see (Crochiere and Rabiner, 1988) for details). The resulting signal is analyzed synchronously with that of the lower band by using the same framing with $N_{in} = 200$ and $M = 80$. Then, the processing carried out for the lower band (Hamming window, 256-points FFT and power of 2) is applied to the higher band, obtaining a spectral representation $|X_{HFB}(i)|^2$ ($i = 0, \ldots, N_{FFT}/2$) of the frame, which is smoothed and shortened to $N_{SPEC} = N_{FFT}/4 + 1$ samples in the same way as in Equations (7.1) and (7.2), obtaining finally a spectral estimate $\overline{|X_{HFB}(i)|^2}$ ($i = 0, \ldots, N_{SPEC} = N_{FFT}/4$).

A mel-scaled triangular filterbank with $K_{HFB} = 3$ filters covering the range from 80 to 4000 Hz, is then applied to $\overline{|X_{HFB}(i)|^2}$ according to Equation (2.9), so the three higher-band filterbank outputs $E_{HFB}(k)$ ($1 \leq k \leq K_{HFB}$) and their corresponding log-outputs $S_{HFB}(k)$ are obtained.

7.3.2.2 Noise Reduction for the HFB

The AFE/XAFE standards apply a very simple noise reduction mechanism to the HFB filterbank outputs based on spectral subtraction. First, an estimate of the noise $\hat{N}(k)$ that affects every filterbank output $E_{HFB}(k)$ is required. This is obtained by applying the same recursive smoothing as in Equation (7.6) and a full-band VAD detector (VADNestH) similar to the one applied to the LFB. Then, the following spectral subtraction (with oversubtraction and thresholding)is applied

$$E_{SS\_HFB}(k) = E_{HFB}(k) - \alpha \hat{N}(k) \tag{7.27}$$

with oversubtraction factor $\alpha = 1.5$ and threshold $\beta E_{HFB}(k)$ ($\beta = 0.1$). Then, the corresponding log-outputs including a rough preemphasis correction are,

$$S_{SS\_HFB}(k) = \log\left((1+\mu)E_{SS\_HFB}(k)\right) \tag{7.28}$$

with $\mu = 0.9$.

The noise reduction technique applied to the HFB filterbank outputs clearly differs from that applied to the LFB. This one, based on Wiener filtering, is much more powerful and complex. As a result, the LFB filterbank outputs are not completely compatible with those of the HFB. The "coding and decoding" technique applied to the HFB tries to reduce this mismatch. In the coding step, we must obtain first the log-energies $S_{LFB\_aux}(k)$ of three auxiliary LFBs between 2 and 4 kHz from the noisy signal spectrum computed in Equations (7.1) and (7.2)

$$S_{LFB\_aux}(1) = \log\left(\sum_{f=2062\ \text{Hz}}^{2375} |\tilde{Y}(f)|^2\right) \tag{7.29}$$

$$S_{LFB\_aux}(2) = \log\left(\sum_{f=2437\ \text{Hz}}^{3000} |\tilde{Y}(f)|^2\right) \tag{7.30}$$

$$S_{LFB\_aux}(3) = \log\left(\sum_{f=3062\ \text{Hz}}^{4000} |\tilde{Y}(f)|^2\right) \tag{7.31}$$

Then, the HFB coding is carried out

$$Code(l,k) = S_{LFB\_aux}(l) - S_{HFB}(k) \qquad (1 \le k, l \le K_{HFB}) \tag{7.32}$$

Decoding is performed from another set of low-frequency auxiliary bands ($S_{w\_LFB\_aux}(k)$; $1 \le k \le K_{HFB}$), although in this case they are obtained from the denoised spectrum $|X_w(f)|^2$ (corresponding to the denoised, preemphasized and windowed LFB signal described in section 7.3.1). They are computed using expressions similar to Equations (7.29)–(7.31), although considering the double density of FFT bins in $X_w(f)$ with respect to $\tilde{Y}(f)$ (which introduced a factor 1/2 inside each logarithm). Finally, the HFBs are decoded as

$$S_{cod\_HFB}(k) = \sum_{l=1}^{K_{HFB}} w_{cod}(l) Decode(l,k) \tag{7.33}$$

where weights $w_{cod}(l)$ are 0.1, 0.2 and 0.7 (for $l = 1, 2, 3$, respectively), and

$$Decode(l,k) = \left(S_{w\_LFB\_aux}(l) - Code(l,k)\right) \tag{7.34}$$

$$= S_{HFB}(k) - S_{LFB\_aux}(l) + S_{w\_LFB\_aux}(l) \tag{7.35}$$

The coding/decoding operation can be interpreted as follows: $(-Code(l,k))$ is the difference between the kth noisy HFB and the lth noisy LFB. Then, in Equation (7.34), this difference is added to the denoised LFB number l. Therefore, $Decode(l,k)$ is a new estimate of the kth HFB log-energy with a smooth transition with respect to the lth denoised LFB. Finally, in Equation (7.33) we carry out a weighted average of the three values of $Decode(l,k)$ ($l = 1, 2, 3$) to obtain a new estimate of the kth HFB log-energy with a smoothed transition from the three LFBs. This smoothing includes somehow a denoising, since $S_{w\_LFB\_aux}(l)$ (denoised) replaces $S_{LFB\_aux}(l)$ (noisy) in Equation (7.35).

The log-energies obtained by spectral subtraction are now combined with the ones obtained from coding and decoding to obtain a new smoothed estimate of the HFB log-energies,

$$S_{HFB}(k) = 0.7S_{cod\_HFB}(k) + 0.3S_{SS\_HFB}(k) \tag{7.36}$$

Before joining the LFB and HFB filterbank outputs, the transition between them is further smoothed by means of

$$S'_{FB}(K_{FB}) = 0.6S_{FB}(K_{FB}) + 0.4S_{\text{aver}} \tag{7.37}$$

$$S'_{HFB}(1) = 0.6S_{HFB}(1) + 0.4S_{\text{aver}} \tag{7.38}$$

where S_{aver} is the mean of $S_{FB}(K_{FB})$ and $S_{HFB}(1)$. The final vector of log-energies that must be DCT-transformed to obtain the cepstral coefficients ($c(n);\ 0 \leq n \leq 12$) is,

$$\left(S_{FB}(1), S_{FB}(2), \ldots, S_{FB}(K_{FB}-1), S'_{FB}(K_{FB}), S'_{HFB}(1), S_{HFB}(2), S_{HFB}(3)\right) \tag{7.39}$$

The log-energy feature computed for the LP signal (Equation (7.26)) is also corrected

$$\log E = \log\left(E_{LP} + E_{HFB}\right) \tag{7.40}$$

where E_{HFB} is estimated (undoing the preemphasis correction) as

$$E_{HFB} = \sum_{k=1}^{K_{HFB}} \exp\left(S_{HFB}(k) - \log(1+\mu)\right) \tag{7.41}$$

7.3.3 Blind Equalization

In order to introduce robustness against channel variations (due either to the use of different microphones or to different acoustic conditions), the advanced FEs include a blind equalization block, which operates directly in the cepstral domain (Mauuary, 1998). Equalization in the cepstral domain is an additive operation

$$c_{eq}(n) = c(n) + c_h(n) \tag{7.42}$$

where $c(n)$ $(n = 0, 1, \ldots, 12)$ are the MFCC coefficients previously obtained, $c_{eq}(n)$ represents the equalized cepstrum and $c_h(n)$ is the cepstrum of the equalization filter. The equalizer can be obtained by minimizing the following MSE function:

$$MSE(n) = E\left[\left(Ref(n) - c_{eq}(n)\right)^2\right] \tag{7.43}$$

where $Ref(n)$ represents the reference cepstrum, which in the advanced standards corresponds to a flat spectrum. We can see in Equation (7.42) that $c_h(n)$ tries to compensate the bias of $c(n)$ with respect to the reference cepstrum. An LMS solution for $c_h(n)$ is

$$c_h(n; t+1) = c_h(n; t) + \mu\left(Ref(n; t) - [c_h(n; t) + c(n; t)]\right) \tag{7.44}$$

where t has been introduced to represent the frame number and $\mu = 0.008789u$ $(0 \leq u \leq 1)$ is the step size, which is related to the frame energy ($u = 0$ for low energy frames and $u = 1$ for the high energy ones) in the standards.

This blind equalization algorithm outperforms RASTA filtering and obtains a performance comparable to that of CMN, but unlike CMN, it can operate on-line so that it avoids the need of any additional delay (Mauuary, 1998).

7.3.4 Voice Activity Detection

The advanced and extended Aurora standards include the computation of several VAD flags: VADNest and VADNestH in AFE/XAFE, and VAD for voicing classification (VADVC) in XFE/XAFE. They are internally required by the noise reduction algorithms (the first two) and by the pitch and class estimation block (the third one), but not transmitted. In order to allow frame-dropping in the recognition server, the advanced standards (AFE and XAFE) consider the inclusion of a VAD flag among the parameters to be transmitted, although its computation is not included as part of the standards, but as an informative annex, so that manufacturers can choose their own VAD.

The proposed VAD is based on detecting rapid energy changes associated with voice onsets. Rather than using energy measures, SNR values derived from the first stage Wiener filter of the noise reduction block are used, which makes the VAD decisions more robust against background noises.

Speech evidence is obtained from three different speech detectors: The first one is based on a whole spectrum measure. The second one is based on a subregion of the spectrum likely to contain the fundamental frequency. Finally, a third detector makes use of the spectral variance that is sensitive to the harmonic structure of voiced sounds. These detectors are described in the following subsections.

7.3.4.1 Whole Spectrum Detector

The whole spectrum detector uses the mel-warped Wiener filter coefficients obtained in the first stage of the 2-stage Wiener filter defined in the noise reduction block. A single input value is defined as the square of the averaged value of the mel-scaled Wiener filter coefficients.

The algorithm is initialized with the maximum value of the input over the 15 first frames of the utterance. Then, the long-term average of the input (`Tracker`) is obtained, and a final decision about speech presence is obtained as follows:

```
If (Frame < 15) and (Acceleration < 2.5)
   Tracker = MAX(Tracker,Input)

If (Input < Tracker x UpperBound)
and (Input > Tracker x LowerBound)
   Tracker = a x Tracker + (1-a) x Input

If (Input < Tracker x Floor)
   Tracker = b x Tracker + (1-b) x Input

If Input > Tracker x Threshold
   output = TRUE
else
   output = FALSE
```

```
a=0.8
b=0.97
UpperBound=1.5
LowerBound=0.75
Floor=0.5
Threshold=1.65
```

`Acceleration` is defined as the double-difference between successive input values, and is used to prevent `Tracker` to be updated if speech occurs during the initial 15 frames.

7.3.4.2 Subband Detector

The subband detector uses the average of the Wiener filter coefficients over the second, third and fourth mel-filter bands. This value is further time-averaged. The algorithm is initialized with the maximum value of the input over the 15 first frames of the utterance. Then, the long-term average of the input is obtained, and a final decision about speech presence is obtained as follows:

```
Input = p x CurrentInput + (1-p) x PreviousInput

If Frame < 15
   Tracker = MAX(Tracker,Input)

If (Input < Tracker x UpperBound)
and (Input > Tracker x LowerBound)
   Tracker = a x Tracker + (1-a) x Input

If (Input < Tracker x Floor)
   Tracker = b x Tracker + (1-b) x Input

If Input > Tracker x Threshold
   output = TRUE
else
   output = FALSE
```

where `p`=0.75 and `Threshold`=3.25. The other parameters have values equal to those used for the previous detector.

7.3.4.3 Spectral Variance Detector

This last detector is based on the spectral variance of the linear-frequency Wiener filter coefficients, and is computed as

$$\frac{1}{N_{SPEC}} \sum_{i=0}^{N_{SPEC}-1} (H_2(i))^2 - \left(\frac{1}{N_{SPEC}} \sum_{i=0}^{N_{SPEC}-1} H_2(i) \right)^2 \tag{7.45}$$

where $H_2(i)$ is the Wiener filter coefficient corresponding to the i-th frequency bin of the frequency-smoothed spectrum (Equation (7.4), second iteration).

The detector uses the same algorithm presented for the subband detector without the temporal smoothing step and with the same values for the parameters.

7.3.4.4 VAD Logic

An initial decision about speech presence is obtained as a logical OR combination of the three detector outputs. That is, the input to the VAD logic algorithm is TRUE whenever any of the three detector outputs is TRUE for the current frames. The final decision is formulated within a hangover scheme that uses a buffer containing the seven most recent raw VAD decisions as follows:

1. If the buffer contains three or more contiguous TRUE values, it is judged that the buffer contains "possible" speech, and a short timer T of five frames is activated if no hangover is already present ($T = 5$).
2. If the buffer contains four or more contiguous TRUE values, it is judged that the buffer contains "likely" speech. If the system is processing the initial part of the utterance (first 35 frames) a long timer is activated ($T = 40$). In other situations, a shorter timer is activated ($T = 23$). This is to prevent the initial noise estimate of the VAD to be too high.
3. If there are less than three contiguous TRUE values in the buffer, the timer is decremented ($T = T - 1$).
4. If the timer is greater than 0 ($T>0$), a TRUE decision is output, otherwise the final VAD decision is FALSE.

7.3.5 Pitch and Class Estimation

This part of the extended standards (XFE and XAFE) specifies how to compute the extension features, that is, pitch(P) and voicing class(VC). A general block diagram of the estimator is shown in Figure 7.7. The input signal $s_{in}(n)$ is the input signal to the whole FE in the case of the XAFE standard, and the signal after offset compensation in the case of the XFE standard. In addition to this input signal, the system uses spectral and energy information provided by the spectrum estimation, energy computation and

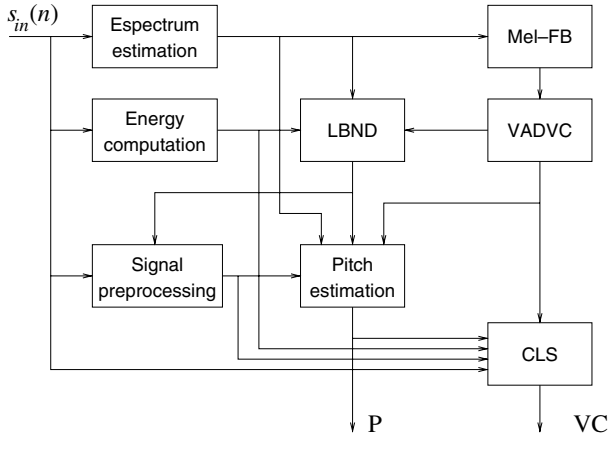


Figure 7.7 Block diagram of the pitch and voicing class estimation

(mel-filterbank) Mel-FB blocks. In the case of the XFE, these blocks and their outputs are the same as those of the basic feature extraction, providing the power spectrum, the frame energy and the filterbank outputs, respectively. In the case of the XAFE, the spectrum estimator is shared with the noise reduction block, although it also generates a preemphasized version of the power spectrum, and the energy computation block is specific (not shared with the basic feature extraction).

The pitch and class estimation system incorporates several blocks in order to make it more robust. Thus, the system has its own voice activity detector (VADVC, VAD for voicingclassification). It provides a speech presence flag used by the pitch estimator. This VAD is similar to the one used in the adaptive multirate speech traffic channels of the GSM Phase 2+ specification (ETSI, 1999d). It is based on an averaged voice metric measure (Vilmur *et al.*, 1989). First, for each of the 23 Mel-FB channels, a raw estimation of the SNR is obtained using a smooth estimate of the background noise in each channel. These raw estimates are then nonlinearly mapped to voice metric values using a tabulated transformation. A voice metric is a measure of the overall voicelike characteristics of each channel. The transformation is nonlinear to take into account the fact that higher SNRs are more indicative of voicelike characteristics in a channel. The tabulated nonlinear transformation is shown in Figure 7.8. Finally, a single measure is obtained by averaging the individual voice metrics of each of the 23 channels. When the averaged voice metric value reaches a given threshold, the current frame is declared to be speech; otherwise it is declared as noise.

Also, the fact that the pitch estimate can be affected by the presence of low-frequency noises, which may appear in some application environments such as cars or planes, must be taken into account. This is the reason the XFE/XAFE standards include a low-frequency-band noise detector (LBND), which provides a flag indicating the presence/absence of this type of noises. The LBND analyzes only nonspeech frames, as indicated by VADVC. The detection is based on the computation of a score R which is updated at every frame as

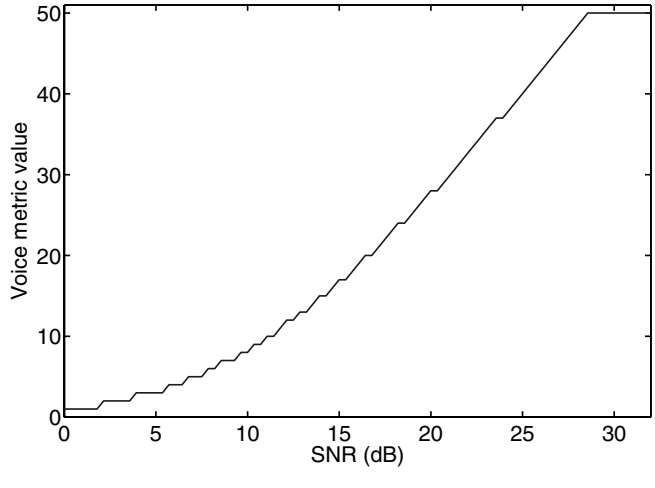


Figure 7.8 Non-linear mapping between SNR values and voice metric values

$R = 0.99R + 0.01R_{cur}$, where R_{cur} is the ratio of the maximum spectral peak in the range [0, 380] Hz to the maximum spectral peak for higher frequencies. The flag is activated when $R > 1.9$. Finally, the preprocessing block provides an LP-filtered and decimated (by a factor 4) signal to be used by the pitch estimator, and an HP-filtered signal to be used by the classification block (CLS). The LP filtering is carried out by a 6th order Butterworth, with a cutoff frequency of 800 Hz in the case of a true LBND flag. If LBND is false, the lowest frequencies are further emphasized with a first-order IIR filter. The XFE specifies different filters and decimation factors (4,5,8) for the different sampling frequencies (8,11,16 kHz), while the XAFE works only at 8 kHz.

The pitch estimation block (Chazan *et al*., 2001; Sorin *et al*., 2004) obtains the pitch for those frames labeled as speech by VADVC with a relatively low complexity. If the frame has been labeled as nonspeech or has an energy below a given threshold, the pitch frequency F_0 is set to zero. The algorithm models the signal as a finite sum of sine waves and extract the pitch using a spectral comb analysis (see Chapter 4, section 4.2.3) in three stages:

1. Generation of a list of pitch candidates. First, the N highest spectral peaks (up to 20), with amplitude A_i (their sum is normalized to one) and frequency f_i ($i = 1, \ldots, N$) are obtained (only peaks with frequency greater than 300 Hz are selected if the LBND flag is true). Assuming a sine-wave model, the utility function of Equation (4.14) becomes,

$$U(F_0) = \sum_{i=1}^{N} A_i I(f_i/F_0) \tag{7.46}$$

   where the comb function is a defined piecewise constant (in order to speed up the search) as (one period),

$$I(r) = \begin{cases} 1 & |r| \leq 65/512 \\ 0.5 & 65/512 < |r| \leq 100/512 \\ 0 & 100/512 < |r| < 0.5 \end{cases} \tag{7.47}$$

   Instead of an exhaustive maximization of the utility function, the search is first restricted to the $M = 7$ highest spectral peaks. Then, the four highest local maxima of the partial utility function built with the M peaks are obtained, and the total utility function (considering now the N peaks) is computed for them. The two maxima providing the highest utilities are considered pitch candidates. In order to obtain a list of candidates more spread, the described search is carried out independently on three intervals ([52,120], [100,210] and [200,420] Hz). If the time pitch function has had a stable behavior during the last six frames, only the intersection of these intervals is considered with the interval $[0.666F_{0prev}, 2.2F_{0prev}]$ (F_{0prev} is the pitch frequency of the previous frame), resulting, finally, in three search intervals SR_1, SR_2 and SR_3 (in increasing frequency order). Then, we finally have a list with (a maximum of) two pitch candidates per search interval. They are ordered according to a criterion that outweighs high F_0 values and values close to that of the previous frame.
2. Computation of correlation scores. A correlation score is computed for every pitch candidate. This is carried out from the LP-filtered and decimated signal provided by

the preprocessing block. Then, every pitch candidate is transformed into a (decimated) time lag and the corresponding correlation score (according to the procedure described in Medan *et al*. (1991)) is computed.

3. Final decision. Every candidate is now represented by three parameters ($F_{0,k}$, A_k and CS_k) (frequency, spectral score and correlation score). The final decision process is carried out over the three frequency intervals. The SR_3 candidate list is processed first. Then, the following lists are successively added (first SR_2, and then SR_1) till a pitch value is found. This decision process is as follows: if there is a candidate with both A_k and CS_k close to 1, this is declared as the frame pitch, and, otherwise, a heuristic decision-making process is applied. If all the intervals have been processed and no decision is finally made, the frame is considered as unvoiced ($F_0 = 0$). The final pitch value is fixed to $P = 8000/F_0$ for voiced frames, and to 0 for unvoiced frames.

The voicing classification block (CLS) labels the speech frame as belonging to one of these four voicing classes (VC): nonspeech, unvoiced, mixed-voiced and fully-voiced. The mixed-voiced class is introduced to enhance the quality of the reconstructed speech. While all the previous processing can provide the nonspeech, unvoiced and voiced labels, the differentiation between mixed- and fully-voiced speech requires further processing based on the analysis of the high-frequency band and the zero-crossings.

7.4 Feature Compression and Encoding

We will see next how the feature set of each standard (see Table 7.1) is encoded for transmission. In spite of the fact that a feature set is computed for every speech frame, the basic encoding and transmission unit of the Aurora standards is the *frame pair* (FP), which consists of two consecutive frames. This fact involves that both frames of the pair are required for channel and source encoding, as we will see in this section.

7.4.1 Basic Feature Vector Quantization

The basic feature vector $\mathbf{x}$ is quantized through the SVQ techniques explained in section 4.5.2. Each vector is split into $N_{cod} = 7$ pairs of consecutive features. For the quantization of a feature pair $\mathbf{x}_n = (x(2n), x(2n+1))$ $(n = 0, \ldots, N_{cod} - 1)$ and for the training of the corresponding codebook, a weighted distance measure is used

$$d(\mathbf{x}_n, \mathbf{x}_n^{(i)}) = \sum_{j=2n}^{2n+1} w_j \left(x(j) - x^{(i)}(j)\right)^2 \tag{7.48}$$

where $\mathbf{x}_n^{(i)} = (x^{(i)}(2n), x^{(i)}(2n+1))$ $(i = 0, \ldots, 2^M - 1$, for an M-bit codebook) represents an SVQ centroid of the n-th codebook. The details of this quantization scheme are summarized in Table 7.3. The bit allocation follows the same guidelines as established in section 4.5.2. The final SVQ indices to be transmitted will be noted as SVQ_n $(n = 0, \ldots, N_{cod} - 1)$.

Table 7.3 Description of SVQ quantization scheme applied to the basic MFCC + log E feature vector

SVQ Codebook	Features (x_{2n}, x_{2n+1})	# Bits FE/XFE	# Bits AFE/XAFE	(w_{2n}, w_{2n+1}) FE/XFE	(w_{2n}, w_{2n+1}) AFE/XAFE
$n = 0$	$(c(1), c(2))$	6	6	(1, 1)	(1, 1)
$n = 1$	$(c(3), c(4))$	6	6	(1, 1)	(1, 1)
$n = 2$	$(c(5), c(6))$	6	6	(1, 1)	(1, 1)
$n = 3$	$(c(7), c(8))$	6	6	(1, 1)	(1, 1)
$n = 4$	$(c(9), c(10))$	6	6	(1, 1)	(1, 1)
$n = 5$	$(c(11), c(12))$	6	5	(1, 1)	(1, 1)
$n = 6$	$(c(0), \log E)$	8	8	(1446.0, 14.7) ($F_s = 8/11$ kHz)	(10498.7, 15.1)
				(1248.9, 12, 7) ($F_s = 16$ kHz)	(10617.2, 21.8)

Table 7.4 Encoding of the extension features

Voicing class (VC)	Pitch index (P_{idx})	Class index (C_{idx})
Non-speech	0	0
Unvoiced speech	0	1
Mixed-voiced speech	>0	0
Fully-voiced speech	>0	1

7.4.2 Extension Features Quantization and Encoding

The extension features (*pitch* and VC) are encoded as indicated in Table 7.4 by using the (quantized) pitch period itself (P_{idx}) and a 1-bit *class index* (C_{idx}).

For pitch quantization, the following quantization rule is employed:

$$P_{\text{idx}} = \underset{1 \le j \le N_{\text{lev}}}{\text{argmin}} \left(P - q_j\right)^2 \tag{7.49}$$

where P is the pitch value to be quantized (in the range [19, 140]), q_j is the center of the jth quantization interval and N_{lev} is the number of quantization intervals. As we shall see below, several cases are considered, with different values for N_{lev} and different distributions of the quantization centers. In most of the cases the quantization centers are uniformly distributed in the log domain,

$$q_j = \exp\left(\log P_{\text{ini}} + (j-1)\frac{\log P_{\text{end}} - \log P_{\text{ini}}}{N_{\text{lev}} - 1}\right) \qquad (j = 1, \ldots, N_{\text{lev}}) \tag{7.50}$$

where $[P_{\text{ini}}, P_{\text{end}}]$ is the considered pitch range for the log-uniform center distribution. In any case, the pitch values are restricted to the interval [19, 140] (samples), and $P_{\text{idx}} = 0$ is reserved to nonspeech or unvoiced speech.

The quantization is applied in a different manner depending on whether the considered frame is the first of a feature pair (frame number m) or the second one (frame number $m + 1$). For the first frame (m), an absolute quantization is applied as described by

Equations (7.49) and (7.50) with 7 bits ($N_{\text{lev}} = 127$, since level 0 is reserved) and with $P_{\text{ini}} = 19$ and $P_{\text{end}} = 140$. For the second frame $(m + 1)$, an 5-bit nonuniform adaptive quantization is employed relative to a previous frame (m, $m - 1$ or $m - 2$), which is referred to as *reference* pitch (*Ref*) in the following. This quantization technique tries to take advantage of the fact that a given frame pitch value is likely to be close to the pitch value of the previous frame(s), and it can also be considered to be a form of differential quantization. This quantization scheme can be summarized as follows:

- If $P_{\text{idx}}(m) > 0$, the reference is $P_{\text{idx}}(m)$ and the quantization of Equation (7.49) is applied with the following 31 levels:
  - First 27 levels: Equation (7.50) is applied with $N_{\text{lev}} = 27$, $P_{\text{ini}} = 0.8163Ref$ and $P_{\text{end}} = 1.225Ref$.
  - Levels 28 to 31: computed as indicated in Table 7.5 (second column).
- If $P_{\text{idx}}(m) = 0$, it is necessary to take as reference a frame previous to m.
  - We can take frame $(m - 1)$ if $P_{\text{idx}}(m - 1) > 0$ and if it is *reliable*, which means that frame $m - 2$ was used as reference to compute $P_{\text{idx}}(m - 1)$ (*unreliable* means that its reference was frame $m - 3$). If some of this conditions are not accomplished, we can still use frame $m - 2$ as reference if $P_{\text{idx}}(m - 2) > 0$. The quantization applied in any of these cases is quite similar to that of the case $P_{\text{idx}}(m) > 0$ (31 levels using Equation (7.49)):
    - First 25 levels: Equation (7.50) is applied with $N_{\text{lev}} = 25$, $P_{\text{ini}} = 0.7781Ref$ and $P_{\text{end}} = 1.2852Ref$.
    - Levels 26 to 31: computed as indicated in Table 7.5 (third column).
  - Otherwise: frame $m - 1$ has $P_{\text{idx}}(m - 1) = 0$ or is unreliable, and $P_{\text{idx}}(m - 2) = 0$. It is not possible to apply the adaptive scheme since there is no reference, so an absolute scalar quantization with $N_{\text{lev}} = 31$, $P_{\text{ini}} = 19$ and $P_{\text{end}} = 140$ is applied.

7.4.3 Bitstream/Payload Format and Error Protection

As previously mentioned, the basic transmission unit of the Aurora standards is the FP. However, the final bitstream or payload format depends on the underlying network. In the case of a circuit-switched network, twelve FPs are grouped into a multiframe in order to reduce the header overhead. The multiframe formats of the Aurora standards are described in the their corresponding ETSI documents. In the case of packet networks, the payload formats are not described in the ETSI documents, but in the IETF documents (Xie, 2003) and (Xie and Pearce, 2005).

Table 7.5 Adaptive pitch quantization: computation of extra levels

$P(m + 1)$ range	Case $P_{\text{idx}}(m) > 0$ $q_{28}, \ldots, q_{31}$	Case $P_{\text{idx}}(m) = 0$ $q_{26}, \ldots, q_{31}$
$19 \leq Ref \leq 30$	$(2.00, 3.00, 4.00, 5.00) \times Ref$	$(1.50, 2.00, 2.50, 3.00, 4.00, 5.00) \times Ref$
$30 < Ref \leq 60$	$(1.50, 2.00, 2.50, 3.00) \times Ref$	$(0.67, 1.50, 2.00, 2.50, 3.00, 4.00) \times Ref$
$60 < Ref \leq 95$	$(0.50, 0.67, 1.50, 2.00) \times Ref$	$(0.33, 0.50, 0.67, 1.50, 1.75, 2.00) \times Ref$
$95 < Ref \leq 140$	$(0.25, 0.33, 0.50, 0.67) \times Ref$	$(0.20, 0.25, 0.33, 0.50, 0.67, 1.50) \times Ref$

7.4.3.1 Frame Pair Formats

Since the ETSI standards provide different feature sets, their corresponding FP payload formats are also different. In order to simplify and systematize the composition of these formats, we will define the following bit sequences:

$$\mathbf{b}_{FE} = (SVQ_0, SVQ_1, SVQ_2, SVQ_3, SVQ_4, SVQ_5, SVQ_6) \quad (44 \text{ bits}) \tag{7.51}$$

$$\mathbf{b}_{AFE} = (SVQ_0, SVQ_1, SVQ_2, SVQ_3, SVQ_4, VAD, SVQ_5, SVQ_6) \quad (44 \text{ bits}) \tag{7.52}$$

$$\mathbf{b}_{EX} = (P_{\text{idx}}(m), P_{\text{idx}}(m+1), C_{\text{idx}}(m), C_{\text{idx}}(m+1)) \quad (14 \text{ bits}) \tag{7.53}$$

It must be noted that both $\mathbf{b}_{FE}$ and $\mathbf{b}_{AFE}$ require 44 bits, but differ in index SVQ_5, which has 6 bits for $\mathbf{b}_{FE}$ and 5 for $\mathbf{b}_{AFE}$. The leftover bit is employed for VAD in $\mathbf{b}_{AFE}$. The final FP format of each ETSI standard is described in Table 7.6. The bitrates are obtained by taking into account the fact that an FP represents 20 ms of speech.

To build the final FP payloads, we must also consider two CRC codes for error protection, **CRC** (four bits) and $\mathbf{PC-CRC}$ (two bits), which protect the basic features (88 bits) and the extension features (14 bits), respectively. Their generator polynomials are $g(x) = 1 + x + x^4$ and $g(x) = 1 + x + x^2$, respectively (see section C.1.2 for details).

7.4.3.2 Multiframe Format for Circuit-switched Networks

In the case of implementation over a circuit-switched network, the standards specify a multiframe format. Each multiframe consists of (see Figure 7.9):

1. A 2-byte fixed synchronization sequence. For the case of a synchronous channel which requires rate adaptation.
2. A 4-byte header. It contains information about sampling rate, FE type (normal/advanced and extended/not-extended), a multiframe counter (starting at 1)and extension bits (not used). This information requires a total of 2 bytes. The header has the final form of a cyclic (31, 16) code plus an overall-parity-check bit. The generator polynomial of the cyclic code is $g(x) = 1 + x^8 + x^{12} + x^{14} + x^{15}$.
3. Frame packet stream. It contains 12 FPs. For the last multiframe of the utterance, the trailing frames are filled with zeros.

The least significant bits are transmitted first. The resulting bitrates taking into account these overhead bits are 4800 bps for FE/AFE and 5600 bps for XFE/XAFE.

Table 7.6 Frame pair formats of the FE/AFE/XFE/XAFE standards

Standard	Frame Pair Payload Format		# Bits (octets)	Bitrate (bps)
	Basic features bit sequence	Extension bit sequence		
FE	$\mathbf{b}_{FE}(m), \mathbf{b}_{FE}(m+1), \mathbf{CRC}$		92(11.5)	4600
AFE	$\mathbf{b}_{AFE}(m), \mathbf{b}_{AFE}(m+1), \mathbf{CRC}$		92(11.5)	4600
XFE	$\mathbf{b}_{FE}(m), \mathbf{b}_{FE}(m+1), \mathbf{CRC}$	$\mathbf{b}_{EX}, \mathbf{PC-CRC}$	108(13.5)	5400
XAFE	$\mathbf{b}_{AFE}(m), \mathbf{b}_{AFE}(m+1), \mathbf{CRC}$	$\mathbf{b}_{EX}, \mathbf{PC-CRC}$	108(13.5)	5400

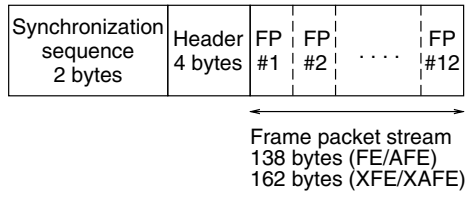


Figure 7.9 Multiframe format

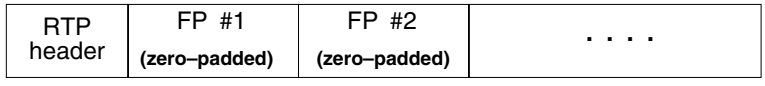


Figure 7.10 Format of a DSR RTP datagram

7.4.3.3 Payload Format for IP Networks

The RFC documents 3557 and 4060 (Xie, 2003; Xie and Pearce, 2005) specify the recommended RTP payload formats for the FE and AFE/XFE/XAFE DSR standards. In this case, the FPs previously described are zero-padded (4 zeros) in order to obtain an integral number of octets per FP (12 for FE/AFE and 14 for XFE/XAFE). The format of the DSR RTP PDU is depicted in Figure 7.10. The DSR RFC specifications allow a variable number of FPs per packet, which can be suitably selected according to the specific characteristics of the implementation. The number of FPs is limited only by the MTU (maximum transmission unit) to avoid packet fragmentation, and must be determined according to the available bandwidth, latency and possibility of packet loss. While small packets increase header overhead, large packets increase latency and the effect of packet loss. This is why RFC-3557 recommends to fix the number of FPs as small as possible (see Chapter 3, section 3.3.4 for more details about payload size).

The transmission may be carried out discontinuously(DTX). The end of a speech segment is indicated with one or more null FPs. A null FP ID created by filling the speech features bits with zeros (the CRC codes are computed in the same way as for a normal FP).

In the RTP header, the timestamp corresponds to the first sample of the first speech frame in the payload, and it is increased by 160, 220 or 320 (according to the particular standard and the sampling frequency) for every FP. The DSR session is described by SDP packets that contain information such as the multipurpose internet mail extensions MIME media type (audio), MIME media subtype (identifying the specific FE), sampling rate and the maximum amount of media contained in a packet.

7.5 Feature Decoding and Postprocessing

7.5.1 Bitstream Decoding and Decompression

In the case of a circuit-switched network, the use of the synchronization sequence and the decoding method of the header of a multiframe are not specified. The header cyclic code may be used for error correction and/or detection. However, the ETSI documents

specify that the multiframe header decoding is not started until two headers, containing coherent data, are received.

Regarding feature decompression, the decoding of the SVQ-quantized features consists of a simple table lookup. On the other hand, the decoding of the pitch and voicing class is straightforward following the rules given in Section 7.4.2.

7.5.2 Error Detection and Concealment

As described in Chapter 5, a speech recognition application allows efficient EC in the case of a degraded transmission channel. This is the reason the ETSI standards follow the approach based on error detection and concealment, which was introduced in Section 5.2.1. Next let us see both stages.

7.5.2.1 Error Detection

We have previously seen that the DSR features are protected by two CRC codes: **CRC** for the VAD and the basic features and **PC − CRC** for the extension features. At the decoder, these codes are reevaluated using the received data and then matched with the received ones in order to check for possible transmission errors. However, the small size of these codes involves a nonnegligible probability of accepting erroneous data as correct. This is the reason an additional *consistency check* is applied to the input data. This check tries to determine whether the basic features contained in an FP have a minimum degree of continuity. First, the consistency of each feature pair is checked

$$badindexflag_n = \begin{cases} 1 & \text{if } (dif(2n) > T_{2n}) \ OR \ (dif(2n+1) > T_{2n+1}) \\ 0 & \text{otherwise} \end{cases} \qquad (n = 0, \ldots, 6) \tag{7.54}$$

where

$$dif(i) = |x_{m+1}(i) - x_m(i)| \qquad (i = 0, \ldots, 13) \tag{7.55}$$

and n indicates the feature pair considered (following the notation used in Section 7.4.1), T_i $(i = 0, \ldots, 13)$ is a similarity threshold for feature i and m and $m + 1$ indicate the first and second frames of an FP. The thresholds have been obtained from error-free speech (their specific values can be found in the free C-programs that accompany the ETSI documents). For the final decision, a voting procedure is used so that if

$$\sum_{n=0}^{6} badindexflag_n \geq 2 \tag{7.56}$$

then the FP is considered inconsistent.

The basic error detection mechanism is the CRC code. When the CRC check fails, then the consistency of the current and previous FPs is checked to determine in which of the FPs we must start the error situation. Then, this situation is maintained until an FP passing both the CRC check and the consistency test correctly is found. The double test in case of error is justified by the fact that errors are expected to appear grouped in bursts.

7.5.2.2 Error Concealment

The EC algorithm of the Aurora standards for recognition is the nonlinear repetition-based interpolation described by Equation (5.8). The repetition is applied to the whole set of received parameters. Since the error detection algorithm has a FP basis, therefore, if it has detected B erroneous pairs, we must fix $T = 2B$ in Equation (5.8). In the case of an error at the beginning (or end) of the speech utterance, the first (or last) correctly received frame is repeated backward (or forward), respectively. The logE, pitch and class parameters are further refined for speech reconstruction.

7.5.3 Server Feature Processing

The FE/XFE do not include any specification about what to do with the received features, so that the processing applied to them in order to obtain the final feature vector employed for recognition (e.g. the computation of the corresponding dynamic features) is completely left to the recognition server. However, the advanced front ends (AFE/XAFE) specify the feature post-processing that must be applied after decoding to generate a 39-dimensional feature vector. This processing can be summarized as follows:

1. Combination of MFCC(0) and log-energy into a single energy feature:

$$\log E \;\&\; c(0) = 0.6c(0)/23 + 0.4 \log E \tag{7.57}$$

   The weights reflect the relevance of each feature. As a result, the basic (static) feature vector $\mathbf{x}$ contains now 13 features (the 13th one is the new combined energy) instead of 14.
2. Computation of 13 velocity and 13 acceleration features. They are computed using Equations (2.15) and (2.17) (except constants) with a length $K = 4$.

7.5.4 Pitch Tracking and Smoothing

According to the DSR concept, pitch and voicing class estimation at the client are aimed at maintaining low complexity. However, this constraint is not so strict at the server side, so that it is possible now to refine the pitch contour and the voicing decision. In order to do that, the pitch tracking and smoothing (PTS)block (in Figure 7.1) uses the received parameters P, VC and $\log E$. The PTS block has three stages and the refined parameters are passed from one stage to the following one:

1. Gross pitch error correction stage. In general, its function consists of recomputing P and VC of a given frame according to the surrounding frames. For example, it reclassifies a voiced frame but surrounded by two unvoiced frames as unvoiced. It does the same if two isolated consecutive voiced frames have very different pitch values. It is also possible that the client pitch estimator has generated a pitch value multiplied or divided by an integer. In this case, a reference pitch value is extracted from the surrounding frames, and the current frame pitch is multiplied or divided by the integer that makes it more similar to that reference value.

2. Voiced/unvoiced decision and other corrections. In a sequence of three frames starting and ending with voiced frames, the pitch of a middle voiced frame is reassigned to the average of the surrounding ones if these have similar pitch values and the middle one clearly differs. If the middle frame is unvoiced, it is directly redefined as voiced.
3. Smoothing. The final pitch value associated with a voiced frame is obtained as a weighted average of the surrounding pitch values. If a frame in the considered interval is unvoiced, it is considered that it has the pitch of the middle frame. On the other hand, if it is voiced, its pitch value is multiplied or divided by the integer, which makes it more similar to the one of the middle frame.

A

Alternative Representations of the LPC Coefficients

When it is required to quantize the LPC coefficients, it is possible that the LPC filter becomes unstable. This is the reason it is sometimes preferred to transform them into new representations more robust for coding purposes. We summarize three of these representations in the following text.

- Reflection coefficients k_i, $1 \leq i \leq p$: Their values are constrained as $|k_i| < 1$, so that it is easier to ensure the filter stability after quantization. They are recursively computed during the Levinson–Durbin recursion used in the autocorrelation method for computation of the LPC coefficients.

  Autocorrelation function of $s(n)$: $r(l) = \frac{1}{N} \sum_{m=0}^{N-1-l} s(m)s(m+l)$

  Levinson–Durbin recursion:

$$E^{(0)} = r(0) \tag{A.1}$$

$$k_i = \frac{r(i) - \sum_{j=1}^{i-1} r(i-j)}{E^{(i-1)}} \tag{A.2}$$

$$a_i^{(i)} = k_i \tag{A.3}$$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i)} \quad (1 \leq j \leq i-1) \tag{A.4}$$

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

  Termination: $a_i = a_i^{(p)}$

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

- Log-area ratios (LARs) L_i, $1 \leq i \leq p$: They are easily derived from the reflection coefficients as

$$L_i = \log \frac{1 + k_i}{1 - k_i} \quad k_i = \frac{1 + e^{L_i}}{1 - e^{L_i}} \tag{A.6}$$

  LARs are useful to avoid problems in the case of ki close to unity.
- Line spectrum pairs/frequencies (LSPs/LSFs): They are the roots of the following polynomials:

$$P(z) = B(z) + z^{-(p+1)} B(z^{-1}) \tag{A.7}$$

$$Q(z) = B(z) - z^{-(p+1)} B(z^{-1}) \tag{A.8}$$

  where $B(z) = 1/H(z) = 1 - A(z)$ is the inverse LPC filter (see Equation (4.12)). The roots of $P(z)$ and $Q(z)$ are interleaved and occur in complex-conjugate pairs so that only $p/2$ roots are retained for each of $P(z)$ and $Q(z)$ (p roots in total). Also, the root magnitudes are known to be unity and, therefore, only their angles (frequencies) are needed. The resulting LSP coefficient set allows a very efficient quantization with DPCM or SVQ.
  Each root of $B(z)$ corresponds to one root in each of $P(z)$ and $Q(z)$. Therefore, if the frequencies of this pair of roots are close, then the original root in $B(z)$ likely represents a formant, and, otherwise, this latter root represents a wide bandwidth feature of the spectrum. These correspondences provide us with an intuitive interpretation of the LSP coefficients.

B

Basic Digital Modulation Concepts

In Section 3.2.3, we considered the transmission of symbols (one every T seconds) from an alphabet $\{m_i; i = 1, \ldots, M\}$. By means of modulation, a signal $x_i(t)$ of duration T suitable for transmission is assigned to each symbol m_i. Instead of working directly with signals, it is quite common and useful in digital communication systems to use a vector representation of the signals. In order to do that, the signals of the set $\{x_i(t); i = 1, \ldots, M\}$ are expressed as a linear combination of time functions (defined in the interval $[0, T]$) from an orthonormal base $\{\phi_j(t); j = 1, \ldots, N\}$:

$$x_i(t) = \sum_{j=1}^{N} x_{ij}\phi_j(t) \qquad \text{with} \qquad x_{ij} = \int_0^T x_i(t)\phi_j(t)\,dt \tag{B.1}$$

Thus, we can represent each signal $x_i(t)$ by a vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iN})^t$, called *signal vector*, of an Euclidean space known as *signal space*. The set of signal vectors is known as *constellation*. The representation of signals using signal vectors is also especially suitable for computational purposes and provides an easy method for performing several signal operations (Arthurs and Dym, 1962).

Two easy and useful examples of modulation and signal space representation are BPSK and QPSK (used in UMTS and IS-95 for the downlink). They are summarized in Table B.1 and depicted in Figure B.1 for a carrier frequency f_c and a signal energy per symbol E. Other types of modulation are $\pi/4$-QPSK (used in IS-136), offset QPSK (OQPSK, used in IS-95 for the uplink), GMSK (used in GSM/GPRS), and 8PSK (used in EGPRS).

The AWGN degradation can be easily expressed in the signal space representation as

$$\mathbf{y} = \mathbf{x}_i + \mathbf{n} \tag{B.2}$$

where $\mathbf{y}$, $\mathbf{x}_i$ and $\mathbf{n}$ are the signal vectors corresponding to $y(t)$, $x_i(t)$ and $n(t)$ and are known as observation, observable and noise vectors, respectively. The components n_j $(j = 1, 2, \ldots, N)$ of the noise vector are samples of Gaussian variables with zero mean and variance $N_0/2$. The received signal $y(t)$ must be demodulated in order to determine its corresponding signal vector $\mathbf{y}$. The effect of the AWGN degradation is that, because of

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

Table B.1 Summary of BPSK and QPSK signals and signal vectors

BPSK	QPSK
$M = 2,\ N = 1$	$M = 4,\ N = 2$
$x_i(t) = \sqrt{\frac{2E}{T}} \cos(2\pi f_c t + (i-1)\pi)$	$x_i(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi f_c t + (2i-1)\frac{\pi}{4}\right)$
$\phi_1(t) = \sqrt{\frac{2}{T}} \cos(2\pi f_c t)$	$\phi_j(t) = \sqrt{\frac{2}{T}} \cos\left(2\pi f_c t - j\frac{\pi}{2}\right)$
$x_1 = +\sqrt{E}$	$\mathbf{x}_1 = \sqrt{E/2}(+1,-1)^t \quad \mathbf{x}_2 = \sqrt{E/2}(-1,-1)^t$
$x_2 = -\sqrt{E}$	$\mathbf{x}_3 = \sqrt{E/2}(-1,+1)^t \quad \mathbf{x}_4 = \sqrt{E/2}(+1,+1)^t$

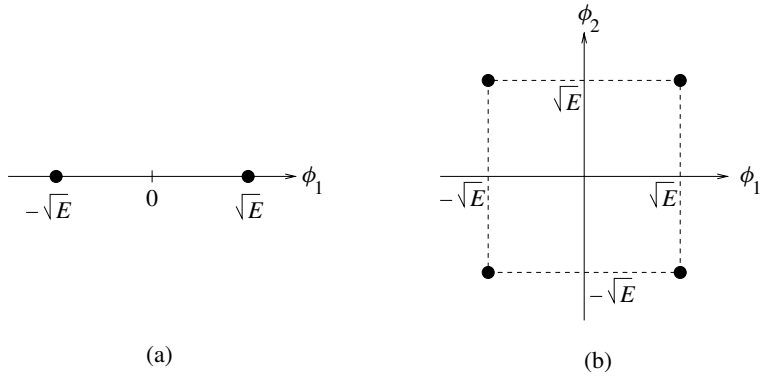


Figure B.1 (a) BPSK and (b) QPSK: signal space representation (constellations)

the random nature of the noise $\mathbf{n}$, it is possible to commit an error when deciding which one is the received symbol $\hat{m}$. This decision operation is called *decoding*, and there are several decision rules for it. For example, the ML decision rule is based on the application of an Euclidean distance:

$$\hat{m} = \underset{i}{\operatorname{argmin}} \, ||\mathbf{y} - \mathbf{x}_i|| \tag{B.3}$$

The ML rule allows us to visualize that the signal space is divided into regions with centroids $\mathbf{x}_i$. The decoded symbol is the one corresponding to the region where $\mathbf{y}$ belongs. For example, for BPSK the monodimensional space is divided into two regions with frontier in $y = 0$, and for QPSK there are four regions corresponding to the four quadrants (see Figure B.1).

The application of the ML rule implies that we are performing *hard decision*, that is, it provides a bitstream to the subsequent stages of the receiver. It must be taken into account that hard decision involves loss of information about the received signal. This is the reason some receivers do not apply decision just after demodulation and directly pass the received signal vector $\mathbf{y}$ to the subsequent stages of the receiver. This is known as *soft decision* and, although it yields a more complex receiver, it allows a better performance of the overall system as shown in Chapter 5.

C

Review of Channel Coding Techniques

C.1 Media-independent FEC

Media-independent FEC techniques are not concerned about the type of information being transmitted. When such techniques are used, the channel coder introduces some degree of redundancy in the bitstream in order to allow the receiver to detect and correct possible channel errors (see Figure 5.1 for a transmission diagram). In wireless channels, the effect of these FEC codes can be seen as an increase in the channel SNR and, therefore, as a reduction in the channel BER. This effect can be helpful to reduce the transmitted power and to employ smaller antennas. In the case of IP networks, the effect is a reduction in the packet loss rate.

The *channel coding theorem* guarantees that if the source rate is smaller than the channel capacity (assuming a discrete memoryless channel), it is possible to find a channel code with a symbol error probability arbitrarily low. However, the theorem does not say anything about how to obtain such a code. Therefore, it is necessary to develop methods for the design of good codes and, then, we will have to select a channel code suitable for our application from these codes. The main known types of media-independent FEC codes are briefly summarized in the following subsections.

C.1.1 Linear Block Codes

A block encoder accepts data blocks $\mathbf{d} = (d_0, \ldots, d_{k-1})$ (datawords), which contain k bits and generates new blocks $\mathbf{c} = (c_0, \ldots, c_{k-1})$ (codewords) of length n ($n > k$). This is a (n, k) block code with a code rate k/n. The code is said to be *systematic* if the dataword $\mathbf{d}$ is a subvector of $\mathbf{c}$. We will assume systematic codes with the structure depicted in Figure C.1, that is, $\mathbf{c} = (\mathbf{b}, \mathbf{d})$. Subvector $\mathbf{b}$ contains the first $(n - k)$ bits of the codeword $\mathbf{c}$. These bits are known as *parity-check bits*.

Codewords can be added and multiplied in modulo-2 arithmetic, which correspond to the logic operations XOR ($\oplus$) and ($\times$), respectively. A block code is *linear* if the addition of any two codewords is another codeword. Thus, the code must contain codeword $\mathbf{0}$,

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

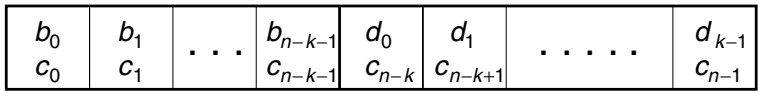


Figure C.1 Structure of a systematic block codeword

since $\mathbf{c} \oplus \mathbf{c} = \mathbf{0}$. Interesting parameters of the codewords are the weight and the hamming distance. The *weight* $w(\mathbf{c})$ of a codeword $\mathbf{c}$ is the number of ones contained in the codeword. The *hamming distance*, defined as the number of different bits between two given codewords, is a measure of how different these codewords are. The minimum distance $d_{\min}$ between any pair of codewords is a parameter usually used to characterize a linear block code. It is possible to show that a linear block code can detect up to t errors if and only if

$$d_{\min} \geq t + 1 \tag{C.1}$$

and correct up to t errors if and only if

$$d_{\min} \geq 2t + 1 \tag{C.2}$$

If the equality is accomplished in the above equations, the code is *perfect*.

The encoding operation can be implemented through the following expression:

$$\mathbf{c} = \mathbf{d}[P, I_k] = \mathbf{d}G \tag{C.3}$$

where I_k is the identity $(k \times k)$ matrix, matrix P defines how the parity bits are computed $(\mathbf{b} = \mathbf{d}P)$, and G is called *generator matrix*. The *parity-check matrix* of the code is defined as $H = [I_{n-k}, P^t]$, and HG^t yields a zero matrix.

The parity-check matrix is useful for decoding. Let us remember that, in section 3.2.4, we learnt that it is possible to introduce the effect of the channel by means of an error pattern $\mathbf{e}$, as $\hat{\mathbf{c}} = \mathbf{c} \oplus \mathbf{e}$, where $\hat{\mathbf{c}}$ is the received codeword. Decoding is equivalent to determining vector $\mathbf{e}$. It can be shown that the syndrome vector, defined as $\mathbf{s} = \hat{\mathbf{c}}H^t$, coincides with $\mathbf{e}H^t$, which only depends on the error pattern. Although the decoding problem is only partially solved, since $\mathbf{s} = \mathbf{e}H^t$ is a system of $(n - k)$ equations with n unknowns, the search space is considerably reduced. Thus, the following procedure (syndrome decoding) allows an efficient decoding:

1. Compute the syndrome $\mathbf{s} = \hat{\mathbf{c}}H^t$.
2. From the set of error patterns defined by $\mathbf{s}$, choose $\mathbf{e}_0$ with the largest probability of occurrence (e.g. in a BSC channel, $\mathbf{e}_0$ is the error pattern with the minimum number of errors).
3. Compute $\mathbf{c} = \hat{\mathbf{c}} \oplus \mathbf{e}_0$ and then extract the received dataword $\hat{\mathbf{d}}$.

An example of linear block codes are *Hamming codes*, characterized by $m = n - k$ and $n = 2^m - 1$ with $m \geq 3$. They have $d_{\min} = 3$, and can perfectly correct $t = 1$ errors.

C.1.2 Cyclic Codes

A linear block code is said to be *cyclic* if a cyclic shift of any codeword is also a codeword of the code. A convenient form to represent cyclic codes is by means of polynomials. Thus, a given codeword **c** is represented by

$$c(x) = c_0 + c_1 x + \cdots, c_{n-1} x^{n-1} \tag{C.4}$$

An interesting property of this representation is that it is possible to perform polynomial divisions with modulo-2 arithmetic. Thus, we can divide two polynomials, $f(x)$ and $h(x)$, so that

$$f(x) = q(x)h(x) + r(x) \tag{C.5}$$

where $q(x)$ is the *quotient* and $r(x)$ is the *remainder*, which can be expressed as

$$r(x) = f(x) \bmod h(x) \tag{C.6}$$

Following a parallelism with the previous subsection, we can say that if $g(x)$ is a polynomial of degree $(n-k)$ and a factor of $(1+x^n)$, then $g(x)$ is a *generator polynomial* of a cyclic code. Thus, each codeword in the code can be obtained as

$$c(x) = a(x)g(x) \tag{C.7}$$

where $a(x)$ is a polynomial of degree $k-1$. The code is uniquely determined by polynomial $g(x)$. It must be noted that, in general, the resulting code is not systematic. The code is also determined by the *parity-check polynomial*, which is defined as a polynomial of degree k that is a factor of $(1+x^n)$ and that is related to $g(x)$ by either

$$g(x)h(x) = 0 \bmod (1+x^n) \tag{C.8}$$

or

$$g(x)h(x) = 1 + x^n \tag{C.9}$$

For simplicity of decoding, it is interesting to have systematic cyclic codes. In this case, we can obtain $\mathbf{c} = (\mathbf{b}, \mathbf{d})$ by computing

$$c(x) = b(x) + x^{n-k} d(x) = \left(x^{n-k} d(x) \bmod g(x)\right) + x^{n-k} d(x) \tag{C.10}$$

with

$$b(x) = \left(x^{n-k} d(x) \bmod g(x)\right) \tag{C.11}$$

These equations specify how a given dataword **d** may be encoded.

A transmitted code $c(x)$ is affected by the channel as

$$\hat{c}(x) = c(x) + e(x) \tag{C.12}$$

where $\hat{c}(x)$ represents the received data and $e(x)$ is a certain error pattern. The *syndrome* polynomial is defined as $s(x) = \hat{c}(x) \bmod g(x)$. It can be easily derived that $s(x) =$

$e(x)$ mod $g(x)$ ($s(x)$ is also the syndrome of $e(x)$), so $s(x)$ only depends on the error pattern. It can be shown that if the weight of the error pattern is less than $d_{\min}/2$, then the syndrome of $e(x)$ is unique. Thus, the syndrome provides us again with a decoding (error-correcting) procedure: we have to select the error pattern with less weight and a syndrome polynomial $s(x)$.

Cyclic codes useful in RSR are the following:

Cyclic redundancy check (CRC) codes: A CRC is a cyclic code that is only used for error detection. CRCs can be quite useful for error detection since they can detect a great variety of error combinations and, besides, there are very efficient implementations for them. A binary (n, k) CRC code can detect (a) error bursts with length up to $n - k$, (b) part of the error bursts with length $n - k + 1$ or greater, (c) any combination of $d_{\min} - 1$ errors and (d) any error pattern with an odd weight if $g(x)$ has an even number of nonzero coefficients (Haykin, 2000).

Bose–Chaudhuri–Hocquenghem (BCH) codes: These are efficient error-correcting codes and provide a flexible selection of n and k. Very common BCH codes are the *primitive BCH codes*, which, for a given $m = n - k$ and a given $t < (2^m - 1)/2$ number of errors, are characterized by the following:

Block length:	$n = 2^m - 1$
Data length:	$k \geq n - mt$
Minimum distance:	$d_{\min} \geq 2t + 1$

Hamming single-error–correcting codes are BCH codes with $t = 1$.

Reed–Solomon (RS) codes: They are a subclass of nonbinary BCH codes, that is, they operate with m-bit symbols. An RS code (n, k) encodes a block of k data symbols into blocks of $n = 2^m - 1$ symbols. Then, the length (in bits) of a codeword is $m(2^m - 1)$ bits. A typical value for m is 8. The characterizing parameters of an RS code are as follows:

Block length:	$n = 2^m - 1$
Data length:	k symbols
Parity-check size:	$n - k = 2t$ symbols
Minimum distance:	$d_{\min} = 2t + 1$

RS codes are very useful when errors appear forming bursts rather than randomly. There also exist efficient decoding techniques that make them very attractive in multiple applications.

C.1.3 Convolutional Codes

A convolutional code with a code rate k/n also generates n output bits from every k input bits, as in the case of block codes. The difference is because the encoding of these k bits is not independent from the bits previously received but it has "memory." A general diagram of a convolutional encoder is shown in Figure C.2. At each time unit, the encoder takes a k-bit input sequence, shifts it through a set of m registers, and generates an n-bit output by performing a linear combination (or convolution), in modulo-2 arithmetic, of the data stored in the registers. The integer m is called *constraint length*. When $k = 1$, we have a special case for which the input bitstream is continuously processed.

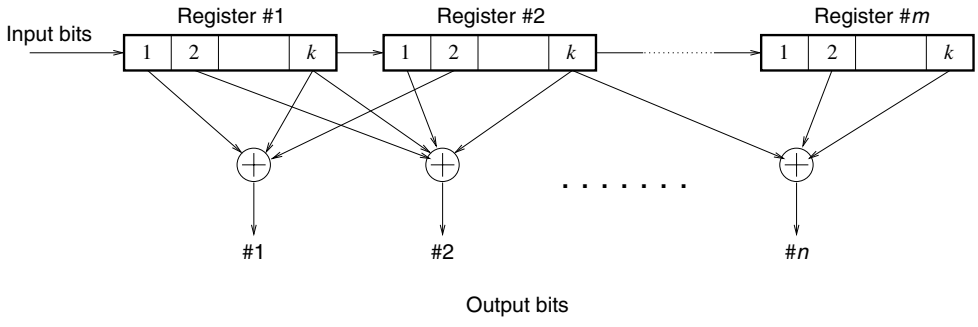


Figure C.2 Structure of a convolutional encoder

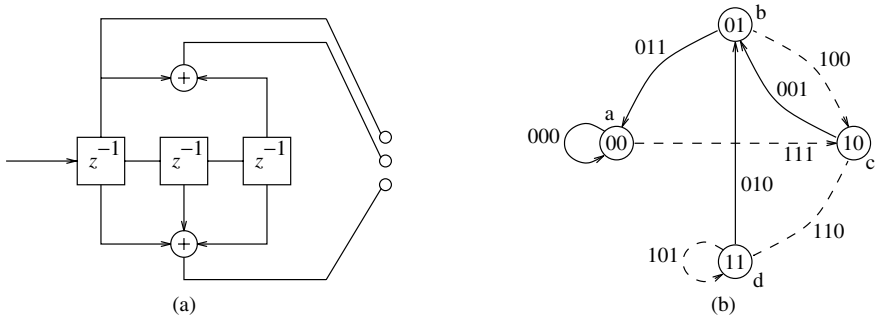


Figure C.3 (a) Example of a convolutional encoder with $k = 1$, $n = 3$ and $m = 3$. (b) State diagram: solid and dashed transition arrows represent input bits 0 and 1, respectively. The encoder outputs are shown close to their corresponding transitions

As mentioned earlier, the output at each time unit depends on the last k incoming bits and the $(m-1)k$ bits received earlier. Thus, we can also interpret the convolutional encoder as a finite-state machine with $2^{(m-1)k}$ states. As an example, Figure C.3 shows an encoder and its corresponding state diagram with $k = 1$, $n = 3$ and $m = 3$. Each state is defined by the $(m-1)k$ bits received earlier. When a new input arrives, there is a transition to the new corresponding state.

For decoding, we can apply a ML rule if we assume that all possible data sequences are equiprobable. Thus, if we have received a bitstream $\hat{\mathbf{c}}$, the corresponding ML estimate is

$$\mathbf{c}_{\text{ML}} = \underset{\text{all possible } \mathbf{c}_i}{\text{argmax}} \; P(\hat{\mathbf{c}}|\mathbf{c}_i) \tag{C.13}$$

This ML rule is equivalent to selecting the codeword that provides minimum hamming distance. However, this minimum distance search can be unfeasible when the length of the transmitted bitstream is very long. In this case, we can consider the encoder finite-state machine as a Markov model with transition probabilities 0.5 (assuming zeros and ones are equiprobable) and observation probabilities derived from the hamming distance. Thus, the VA developed in Chapter 2 can be applied to determine the best state sequence,

which is equivalent to estimating which was the input sequence. With this procedure, we are obtaining an ML decoding in the case of an AWGN channel.

In order to implement decoding, codes with rate $1/a$ are quite efficient. The problem is that if we choose a $1/a$ code, then there is little flexibility to set the final bitrate. This problem can be overcome if we delete or puncture some output bits according to a known pattern. This technique is known as *puncturing*. For example, we can puncture every fourth bit from the output of a $1/2$ code, so that the resulting rate is $2/3$.

So far, we have assumed that $\hat{\mathbf{c}}$ is a bit sequence, which implies that *hard decision* has been applied earlier to the channel outputs. However, the performance of the VA-based decoding procedure can be improved if it is directly fed with continuous values obtained by applying *soft decision* to the channel outputs. Moreover, we can even apply the SOVA or the Max-Log-MAP algorithm (Fossorier *et al*., 1998; Hagenauer, 1980), which provide soft outputs. Given a specific transmitted bit d_k, these algorithms provide

$$L(k) = \log \frac{P(d_k = 1|\hat{\mathbf{c}})}{P(d_k = 0|\hat{\mathbf{c}})} \tag{C.14}$$

Thus, each output bit $\hat{d}_k$ is given a reliability value $|L(k)|$ from which we can compute the instantaneous bit error probability as

$$p_e(k) = \frac{1}{1 + \exp|L(k)|} \tag{C.15}$$

This can be useful to perform a more accurate estimation of the received parameters, as shown in Chapter 5.

C.2 Interleaving

The goal of interleaving is to break long error bursts, which can be quite damaging, into smaller bursts by spreading them over time. Thus, the effect of an interleaver is that errors appear more randomly distributed, so that error protection codes have a chance to detect or correct the errors. The interleaver is usually the last stage of the channel encoder at the transmitter and the deinterleaver is the first stage of the channel decoder at the receiver. The interleaver reorders the symbol sequence a_t (t is the order index) to be transmitted and generates a new sequence, b_t, so that $b_t = a_{\pi(t)}$, where $\pi(t)$ is the reordering function. The deinterleaver reverses that operation by applying $a_t = b_{\pi^{-1}(t)}$, where the condition $\pi(\pi^{-1}(t)) = t$ must be accomplished. Thus, if the transmitted sequence b_t suffers an error burst during transmission, the deinterleaver breaks it into smaller length bursts. This burst breaking effect (it was already illustrated in Figure 5.3) eases the task of the posterior error correction or concealment algorithms.

An interleaver can be formally defined as a periodic permutation of integers $\pi : \mathbb{Z} \to \mathbb{Z}$. The period p is defined so that $\pi(t + p) = \pi(t) + p$, that is, the interleaver does the same operation every p symbols. In order to carry out the permutation, it is usually necessary to buffer incoming symbols. Therefore, the interleaver introduces a *delay*, which can be computed as

$$\delta^+ = \max_t [\pi(t) - t] \tag{C.16}$$

Similarly, the deinterleaver also introduces a delay that is obtained as

$$\delta^- = \max_t \left[\pi^{-1}(t) - t\right] \tag{C.17}$$

The sum $\delta = \delta^+ + \delta^-$ is the total *latency* of the interleaving/deinterleaving process.

Another important parameter of a interleaver is its *spread* S. It is defined by the following relation:

$$|\pi(t_1) - \pi(t_2)| \geq S \text{ whenever } |t_1 - t_2| < S \tag{C.18}$$

The spread S gives us an idea of how close symbols are spread by the interleaver, and, therefore, how effective it is against error bursts. Thus, a burst of length T symbols will be totally broken into errors of length 1 if $S \geq T$. This cannot be ensured in the case of $S < T$, although the interleaver is still beneficial since it breaks the error burst into smaller ones. We see that a large value of S is desirable, although it must be taken into account that the larger S is, the larger is the delay.

A way of implementing interleaving is to arrange the incoming symbols into a matrix C of dimension $d \times n$. Thus, the symbol arriving at time $t = i * n + j$ $(0 \leq t \leq nd$, $0 \leq i \leq d, 0 \leq j \leq n)$ is placed at position (i, j) of the matrix, that is, the matrix is filled in row by row. Usually, each row corresponds to a codeword of a (n, k) block code. Then, the symbols are read from the matrix column by column for transmission (instead of row by row), thus obtaining an interleaved code. This technique is referred to as *block* interleaving. The parameter d is the *degree* of interleaving. If the original block code can correct error bursts of length T, the resulting (dn, dk) interleaved block code can manage bursts of length dT. Figure C.4 shows the block interleaver with $n = d = 4$ employed in the example of Figure 5.3. The input symbols are placed in the rows of a 4×4 matrix as they arrive. The arrow indicates the order in which the symbols are read (transmitted) from the matrix. The permutation function of this interleaver can be expressed as

$$\pi(t = i * d + j) = (d - 1 - j)d + i \tag{C.19}$$

Its associated delays and spread are $\delta^+ = \delta^- = d^2 - d$ and $S = d$, respectively.

There also exist convolutional interleavers, which are more suitable for convolutional codes. These interleavers divide the input symbol sequence into subsequences of length d, which is defined as the interleaver degree. Each element of a given subsequence suffers

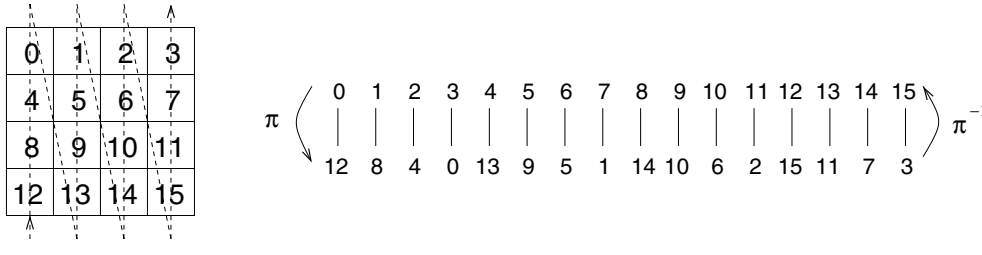


Figure C.4 Example of a 4×4 block interleaver

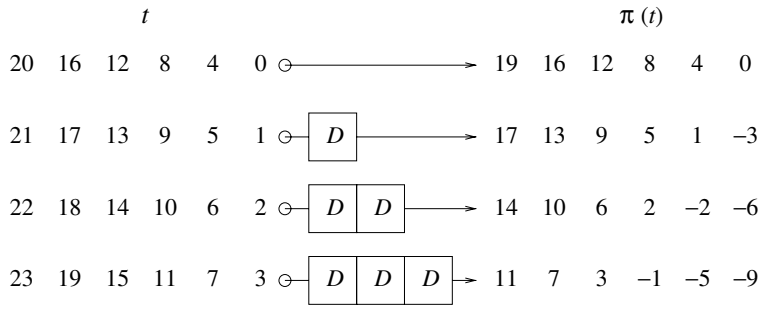


Figure C.5 Example of convolutional interleaver with degree $d = 4$. Unit delays are represented by D

a different delay. Figure C.5 shows an example of a convolutional interleaver of degree $d = 4$ with a permutation function

$$\pi(t) = t - d(t \bmod d) \tag{C.20}$$

Its associated delays and spread are $\delta^+ = 0$, $\delta^- = d^2 - d$ and $S = d - 1$, respectively.

Bibliography

3GPP 2002 ETSI TR 122 977 – UMTS; Feasibility study for speech-enabled services. Technical Report ETSI TR 122 977, ETSI.

3GPP 2004a ETSI TR 126 943 – Recognition performance evaluations of codecs for speech enabled services. Technical Report ETSI TR 126 943, ETSI.

3GPP 2004b ETSI TS 126 243 – UMTS; ANSI-C code for the fixed-point distributed speech recognition extended advanced front-end. Technical Report ETSI TS 126 243, ETSI.

3GPP 2005 ETSI TS 126 235 – UMTS; Packet switched conversational multimedia applications; default codecs. Technical Report ETSI TS 126 235, ETSI.

Alexandre P and Lockwood P 1993 Root cepstral analysis: an unified view. Application to speech processing in car noise environments. *Speech Communication* **12**, 277–288.

Andersen S, Duric A, Astrom H, Hagen R, Kleijn W and Linden J 2004 RFC 3951 – Internet Low Bit Rate Codec (iLBC). Technical Report RFC 3951, IETF.

Argawal A and Cheng Y 1999 Two-stage mel-warped Wiener filter for robust speech recognition *1999 Workshop on Automatic Speech Recognition and Understanding (ASRU'99)*. Keystone, Colorado, USA.

Arrowood J and Clements M 2004 Extended cluster information vector quantization (ECI-VQ) for robust classification *Proc. of ICASSP'2004*. Montreal, Canada.

Arrowood JA and Clements MA 2002 Using observation uncertainty in hmm decoding *Proc. of ICSLP'02*, Denver, Colorado.

Arthurs E and Dym H 1962 On the optimum detection of digital signals in the presence of white gaussian noise – a geometric interpretation and a study of the three basic data transmission systems. *IRE Trans. on Communications Systems* **10**, 336–372.

Atal B 1974 Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America* **55**(6), 1304–1312.

Atal B and Remde J 1982 A new model of lpc excitation for producing natural-sounding speech at low bit rates *Proc. of ICASSP'82*, pp. 614–617. Paris, France.

Atal B and Schroeder M 1979 Predictive coding of speech and subjective error criteria. *IEEE Trans. on Acoustics, Speech and Signal Processing* **27**, 247–254.

Avendano C, van Vuuren S and Hermansky H 1996 Data-based RASTA-like filter design for channel normalization in ASR. *Proc. ICSLP'96* **4**, 2087–2090. Philadelphia, PA, USA.

Bahl L, Brown P, de Souza P and Mercer R 1986 Maximum mutual information estimation of hidden markov models parameters for speech recognition *Proc. of ICASSP-86*, pp. 49–52, Tokyo.

Bahl L, Jelinek F and Mercer R 1983 A maximum likelihood approach to continuous speech recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **5**, 179–190.

Bai H and Atiquzzaman M 2003 Error modeling schemes for fading channels in wireless communications: a survey. *IEEE Communications Surveys* **5**(2), 2–9.

Bannister J, Mather P and Coope S 2004 *Convergence technologies for 3G networks*. Wiley.

Bawab Z, Locher I, Xue J and Alwaan A 2003 Speech recognition over bluetooth wireless channels *Proc. of Eurospeech'2003*. Geneva, Switzerland.

Bellman RE 1957 *Dynamic Programming*. Princeton Univ. Press.

Bernard A and Alwan A 2001 Source and channel coding for remote speech recognition over error-prone channels *Proc. of ICASSP'2001*, pp. 2613–2616. Salt Lake City, Utah, USA.

Bernard A and Alwan A 2002 Low-bitrate distributed speech recognition for packet-based and wireless communication. *IEEE Trans. on Speech and Audio Processing* **10**(8), 570–579.

Bernstein AD and Shallom ID 1991 An hypothesized Wiener filtering approach to noisy speech recognition. *Proc. of ICASSP'91* **2**, 913–916. Toronto, Canada.

Berouti M, Schwartz R and Makhoul J 1979 Enhancement of speech corrupted by acoustic noise *Proc. of ICASSP'1979*, pp. 208–211. Washington DC, USA.

Blake S, Black D, Carlson M, Davies E, Wang Z and Weiss W 1998 RFC 2475 – An architecture for differentiated services. Technical Report RFC 2475, IETF.

Boll SF 1979 Suppression of acoustic noise in speech using spectral subtraction. *IEEE Trans. on ASSP* **27**(2), 113–120.

Bolot J 1993 End-to-end packet delay and loss behavior in the internet *ACM Sigcomm*, pp. 289–298. San Francisco, CA, USA.

Bolot J, Crepin H and Vega A 1995 Analysis of Audio Packet Loss in the Internet *Proc. of Int. Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 163–174. Durham, New Hampshire, USA.

Boulis C, Ostendorf M, Riskin E and Otterson S 2002 Graceful degradation of speech recognition performance over packet-erasure networks. *IEEE Trans. on Speech and Audio Processing* **10**(8), 580–590.

Buzo A, Gray A, Gray R and Markel J 1980 Speech coding based upon vector quantization. *IEEE Trans. on ASSP* **28**, 562–574.

Campbell J, Welch V and Tremain T 1989 An expandable error-protected 4800 bps celp coder (US federal stadard 4800 bps voice coder) *Proc. of ICASSP-89*. Glasgow, Scotland.

Cappe O 1994 Elimination of the musical noise phenomenon with the ephraim and malah noise suppressor. *IEEE Trans. on Speech and Audio Processing* **2**(2), 345–349.

Cardenal A and Garcia C 2004 Correlation-based soft-decoding for distributed speech recognition over IP networks *Proc. of Robust'2004 (Cost 278 and ISCA-ITRW)*. Norwich, UK.

Cardenal A, Docio L and Garcia C 2004 Soft decoding strategies for distributed speech recognition over IP networks *Proc. of ICASSP'2004*. Montral, Canada.

Chazan D, Tzur M, Hoory R and Cohen G 2001 Efficient periodicity extraction based on sine-wave representations and its application to pitch determination of Speech Signals *Proc. of Eurospeech'2001*. Aalborg, Denmark.

Chen J 1990 High quality 16 kbit/s speech coding with a one-way delay less than 2 ms *Proc. of IEEE ICASSP-90*, pp. 453–456. Albuquerque, New Mexico, USA.

Chesta C, Siohan O and Lee CH 1999 Maximum a posteriori linear regression for hidden markov model adaptation *Proc. of EuroSpeech'99*, pp. 211–214. Budapest, Hungary.

Choi S, Kim H and Lee H 2000 Speech recognition using quantized LSP parameters and their transformations in digital communication. *Speech Communication* (30), 223–233.

Crochiere R and Rabiner L 1988 *Advanced Topics in Signal Processing* Prentice Hall chapter Multirate Signal Processing.

D. Pearce JE, Ferrans J and Johnson J 2005 An architecture for seamless access to distributed multimodal services *Proc. of EUROSPEECH'2005*. Lisbon, Portugal.

Davis S and Mermelstein P 1980 Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustics, Speech and Signal Processing* **28**(4), 357–366.

de la Torre A, Peinado AM, Segura JC, Pérez JL, Benítez MC and Rubio A 2005 Histogram equalization of the speech representation for robust speech recognition. *IEEE Trans. on SAP* **13**(3), 355–366.

de la Torre A, Segura JC, Benítez MC, Peinado AM and Rubio A 2003 Non-linear transformations of the feature space for robust speech recognition *Proc. of ICASSP'02*, vol. 1, pp. 401–404, Orlando.

de Wet F, de Veth J, Cranen B and Boves L 2003 The impact of spectral and energy mismatch on the Aurora-2 recognition task *Proc. of ICASSP'03*, vol. II, pp. 105–108, Hong Kong.

Dehery Y, Lever M and Urcun P 1991 A MUSICAM source codec for digital audio broadcasting and storage *Proc. of ICASSP 1991*, pp. 3605–3608. Toronto, Canada.

Delaney B 2005 Increased robustness against bit errors for distributed speech recognition in wireless environments *Proc. of ICASSP'2005*. Philadelphia, USA.

Deligne S, Dharanipragada S, Gopinath R, Maison B, Olsen P and Printz H 2002 A robust high accuracy speech recognition system for mobile applications. *IEEE Trans. on Speech and Audio Processing* **10**(8), 551–561.

Deller J, Proakis J and Manolakis J 1993 *Discrete-Time Processing of Speech Signal*. Prentice Hall.

Demichelis P, Rinitti A and Martin JD 2005 Performance analysis of distributed speech recognition over 802.11 wireless networks on the TIMIT database *Proc. of the 61st Simiannual Vehicular Tech. Conf.* Stockholm, Sweden.

Deng L, Acero A, Jiang L, Droppo J and Huang X 2001 High-performance robust speech recognition using stereo training data. *Proc. of ICASSP'01* **1**, 301–304. alt Lake City, Utah, USA.

Deng L, Wang K, Acero A, Hon H, Droppo J, Boulis C, Wang Y, Jacoby D, Mahahan M, Chelba C and Huang X 2002 Distributed speech processing in MiPads multimodal user interface. *IEEE Trans. on Speech and Audio Processing* **10**(8), 605–619.

Dharanipragada S and Padmanabhan M 2000 A non-linear unsupervised adaptation technique for speech recognition *Proc. of ICSLP'00*, pp. 556–559, Peking, China.

Digalakis V, Neumeyer L and Perakakis M 1998a Product-code vector quantization of cepstral parameters for speech recognition over the WWW *Proc. of ICSLP'98*. Sydney, Australia.

Digalakis V, Neumeyer L and Perakakis M 1998b Quantization of cepstral parameters for speech recognition over the world wide web *Proc. of ICASSP'98*, pp. 989–992. Seattle, Washington, USA.

Digalakis V, Neumeyer L and Perakakis M 1999 Quantization of cepstral parameters for speech recognition over the world wide web. *IEEE Journal on Selected Areas in Communications* **17**(1), 82–90.

Droppo J, Acero A and Deng L 2002 Uncertainty decodign with SPLICE for robust speech recognition *Proc. of ICASSP'02*. Orlando, USA.

Dufour S, Glorion C and Lockwood P 1996 Evaluation of the root-normalised front-end (RN_LFCC) for speech recognition in wireless GSM network environments *Proc. of ICASSP'96*. Atlanta, GA, USA.

Elliot E 1963 Estimates of error rates for codes on bursty noise channels. *Bell System Technical Journal* **42**(9), 1977–1997.

Elliot E 1965 A model of the switched telephone network for data communications. *Bell System Technical Journal* **44**(1), 89–109.

Ephraim Y and Malah D 1984 Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Trans. on Acoustics, Speech, and Signal Processing* **32**(6), 1109–1121.

Ephraim Y and Malah D 1985 Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Trans. on ASSP* **33**, 443–445.

Ephraim Y, Dembo A and Rabiner L 1989 A minimum discrimination information approach for hidden markov modeling. *IEEE Trans. on Information Theory* **35**(5), 1001–1013.

ETSI 1995 ETSI EN 300 961 – GSM Full Rate Speech Transcoding (GSM 06.10). Technical Report ETSI EN 300 961, ETSI.

ETSI 1998a ETSI EN 300 909 – Digital cellular telecommunications system (Phase 2+) (GSM);Channel coding (GSM 05.03 version 7.3.1 Release 1998). Technical Report ETSI EN 300 909, ETSI.

ETSI 1998b ETSI EN 301 704 – Adaptive Multi-Rate (AMR) Speech Transcoding (GSM 06.90). Technical Report ETSI EN 301 704, ETSI.

ETSI 1999a ETSI EN 300 726 – Enhanced Full Rate (EFR) Speech Transcoding (GSM 06.60). Technical Report ETSI EN 300 726, ETSI.

ETSI 1999b ETSI EN 300 903 – Digital cellular telecommunications system (Phase 2+); transmission planning aspects of the speech service in the GSM public land mobile network (PLMN) system (GSM 03.50). Technical Report ETSI EN 300 903, ETSI.

ETSI 1999c ETSI EN 301 705 – Substitution and muting of lost frames for adaptive multi rate (AMR) speech traffic channels. Technical Report ETSI EN 301 705, ETSI.

ETSI 1999d ETSI EN 301 708 – Digital cellular system (Phase 2+); Voice Activity Detector (VAD) for Adaptive Multi-Rate (AMR) speech traffic channels; General description. Technical Report ETSI EN 301 708, ETSI.

ETSI 2000a ETSI EN 300 727 – Substitution and muting of lost frames for enhanced full rate (EFR) speech traffic channels. Technical Report ETSI EN 300 727, ETSI.

ETSI 2000b ETSI TIPHON TS 101-329-5; QoS measurements for Voice over IP. Technical Report ETSI TIPHON TS 101-329-5, ETSI.

ETSI 2000c ETSI TR 101 085 v8.0.0, 2000. Digital cellular telecommunications system (Phase 2+) (GSM);Performance characterization of the GSM Enhanced Full Rate (EFR) speech. Technical Report ETSI TR 101 085, ESTI.

ETSI 2001 ETSI ES 202 211 v1.1.1. Distributed speech recognition; Extended front-end feature extraction algorithm; Compression algorithms; Back-end speech reconstruction algorithm. Technical Report ETSI ES 202 211, ETSI.

ETSI 2003a ETSI ES 201 108 v1.1.3. Distributed Speech Recognition; Front-end Feature Extraction Algorithm; Compression Algorithms. Technical Report ETSI ES 201 108, ETSI.

ETSI 2003b ETSI ES 202 050 v1.1.3. Distributed Speech Recognition; Advanced Front-end Feature Extraction Algorithm; Compression Algorithms. Technical Report ETSI ES 202 050, ETSI.

ETSI 2003c ETSI ES 202 212 v1.1.1. Distributed speech recognition; Extended advanced front-end feature extraction algorithm; Compression algorithms; Back-end speech reconstruction. Technical Report ETSI ES 202 212, ETSI.

ETSI 2003d ETSI TS 128 062 – Inband Tandem Free Operation (TFO) of speech codecs; Service description. Technical Report ETSI TS 128 062, ETSI.

Euler S and Zinke J 1994 The influence of speech coding algorithms on automatic speech recognition *Proc. of ICASSP'94*, pp. I–621–624. Adelaide, Australia.

Fingscheidt T, Aalburg S, Stan S and Beaugeant C 2002 Network-based vs. distributes speech recognition in adaptive multi-rate wireless systems *Proc. of ICSLP'2002*. Denver, USA.

Fingscheidt T and Vary P 2001 Softbit speech decoding: a new approach to error concealment. *IEEE Trans. on Speech and Audio Processing* **9**(3), 240–251.

Fossorier M, Burkert F and Hagenauer J 1998 On the equivalence between SOVA and Max-Log-MAP decodings. *IEEE Communications Letters* **2**(5), 137–139.

Fox B and Gleeson B 1999 RFC 2685 – Virtual private networks identifier. Technical Report RFC 2685, IETF.

Furui S 1981 Cepstral analysis technique for automatic speaker verification. *IEEE Trans. on ASSP* **29**, 254–272.

Furui S 1986 Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Trans. on ASSP* **34**, 52–59.

Gales M 1996 The generation and use of regression class trees for MLLR adaptation. Technical Report Technical Report CUED/F-INFENG/TR263, Cambridge University.

Gales M, Pye D and Woodland P 1996 Variance compensation within the MLLR framework for robust speech recognition and speaker adaptation *Proc. ICSLP'96*, vol. citeseer.ist.psu.edu/gales96variance.html, pp. 1832–1835, Philadelphia, PA. Philadelphia, PA.

Gallardo A, Diaz F and Valverde F 1998 Recognition from GSM digital speech *Proc. of ICSLP'98*. Sydney, Australia.

Gallardo A, Diaz F and Valverde F 1999 Avoiding distortions due to speech coding and transmission errors in GSM ASR tasks *Proc. of ICASSP'99*, pp. 277–280. Phoenix, Arizona, USA.

Ganawan W and Hasegawa-Johnson M 2001 PLP coefficients can be quantized at 400 bps *Proc. of ICASSP'2001*, pp. 77–80. Salt Lake City, Utah, USA.

Garg V and Wilkes J 1999 *Principles and Applications of GSM*. Prentice Hall.

Gersho A and Gray R 1991 *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Gersho A and Gray R 1992 *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Gilbert E 1960 Capacity of a burst noise channel. *Bell System Technical Journal* **39**(5), 1253–1265.

Gleeson B, Lin A, Heinanen J, Armitage G and Malis A 2000 RFC 2764 – A framework for IP based virtual private networks. Technical Report RFC 2764, IETF.

Gomez A, Peinado A, Sanchez V and Rubio A In Press Combining media-specific FEC and error concealment for robust distributed speech recognition over loss-prone packet channels. *IEEE Trans. on Multimedia (2006)*.

Gomez A, Peinado A, Sanchez V, Milner B and Rubio A 2004 Statistical-based reconstruction methods for speech recognition in IP networks *Proc. of Robust'2004 (Cost 278 and ISCA-ITRW)*. Norwich, UK.

Gomez A, Peinado AM, Sanchez V and Rubio A 2003 A source model mitigation techniques for distributed speech recognition over lossy packet channels *Proc. of Eurospeech'2003*. Geneva, Switzerland.

Gray R 1984 Vector quantization. *IEEE ASSP Magazine* **1**(2), 4–29.

Gunawan W and Hasegawa-Johnson M 2001 PLP coefficients can be quantized at 400bps *Proc. of ICASSP'2001*, pp. 77–80. Salt Lake City, Utah, USA.

Haeb-Umbach R 1997 Robust speech recognition for wireless networks and mobile telephony *Proc. of Eurospeech'97*. Rhodes, Greece.

Haeb-Umbach R and Ion V 2004 Soft features for improved distributed speech recognition over wireless networks *Proc. of ICSLP'2004*. Jeju Island, Korea.

Hagenauer J 1980 Viterbi decoding of convolutional codes for fading- burst-channels *Proc. of Zurich Seminar on Digital Communications*, pp. G2.1–G2.7. Zurich, Switzerland.

Han K, Srinivasamurthy N and Narayanan S 2004 A distributed speech recognition system in multi-user environments *Proc. of ICSLP'2004*. Jeju Island, Korea.

Hanson B and Applebaum T 1990 Robust speaker-independent word recognition using static, dynamic and acceleration features: experiments with lombard and noisy speech *Proc. of ICASSP'1990*. Albuquerque, New Mexico, USA.

Haykin S 2000 *Communication sytems*. Wiley.

Hermansky H 1990 Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America* **77**(4), 1738–52.

Hermansky H and Jain P 1999 Downsampling speech representation in ASR *Proc. of Eurospeech'99*. Budapest, Hungary.

Hermansky H and Morgan N 1994 RASTA processing of speech. *IEEE Trans. of Speech and Audio Processing* **2**(4), 578–589.

Hilgher F and Ney H 2001 Quantile based histogram equalization for noise robust speech recognition *Proc. of EUROSPEECH 2001*, pp. 1135–1138, Denver.

Hilgher F, Molau S and Ney H 2002 Quantile based histogram equalization for noise robust speech recognition *Proc. of ICSLP 2002*, pp. 237–240, Denver.

Hirsh H 2002 The influence of speech coding on recognition performance in telecommunication networks *Proc. of ICSLP'02*, pp. 1877–1880. Denver, USA.

Hirsh H and Pearce D 2000 The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions *Proc. of ISCA workshop ASR'2000*. Paris, France.

Honkanen T, Vainio J, Jarvinen K, Haavisto P, Salami R, Laflamme J and Adoul J 1997 Enhanced full rate speech codec for IS-136 digital cellular system *Proc. of ICASSP'1997*. Munich, Germany.

Hsu CW and Lee LS 2004a Higher order cepstral moment normalization (HOCMN) for robust speech recognition *Proc. of ICASSP'2004*, pp. 197–200. Montreal, Canada.

Hsu W and Lee L 2004b Efficient and robust distributed speech recognition (DSR) over wireless fading channels: 2D-DCT compression, iterative bit allocation, short BCH code and interleaving *Proc. of ICASSP'2004*. Montreal, Canada.

HTK3 2004 HTK3 – Hidden Markov Model Toolkit. Technical report, Cambridge University, http://htk.eng.cam.ac.uk/.

Huang X, Acero A and Hon H 2001 *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall.

Huang X and Lee KF 1993 On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition. *IEEE Trans. on Speech and Audio Processing* **1**(2), 150–157.

Huang X, Lee K and Hon H 1990 On semi-continuous hidden markov modeling *Proc. of ICASSP-90*, pp. 689–692. Albuquerque, New Mexico, USA.

Huerta J and Stern R 1998 Speech recognition from GSM codec parameters *Proc. of ICSLP'98*. Sydney, Australia.

Huerta J and Stern R 2001 Distortion-class modeling for robust speech recognition under GSM RPE-LTP coding. *Speech Communication* (34), 213–225.

IETF n.d. http://www.ietf.org.

Ion V and Haeb-Umbach R 2005a A comparison of soft-feature speech recognition with candidate codecs for speech enabled mobile services *Proc. of ICASSP'2005*. Philadelphia, USA.

Ion V and Haeb-Umbach R 2005b A unified probabilistic approach to error concealment for distributed speech recognition *Proc. of Eurospeech'2005*. Lisbon, Portugal.

ISO 1999 ISO/IEC 14496-3 – Coding of Audio-Visual Objects – Part 3. Technical Report ISO/IEC 14496-3, ISO.

ITU-T 1988a ITU-T Recommendation G.711. Pulse code modulation (PCM) of voice frequencies. Technical Report ITU-T G.711, ITU.

ITU-T 1988b ITU-T Recommendation G.722. 7 kHz audio-coding within 64 kbit/s. Technical Report ITU-T G.722, ITU.

ITU-T 1990a ITU-T Recommendation G.726. 40,32,24,16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM). Technical Report ITU-T G.726, ITU.

ITU-T 1990b ITU-T Recommendation G.727. 5-, 4-, 3- and 2-bit/sample embedded adaptive differential pulse code modulation (ADPCM). Technical Report ITU-T G.727, ITU.

ITU-T 2001 ITU-T Recommendation P.862. Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs. Technical Report ITU-T P.862, ITU.

Jakes W 1974 *Microwave mobiles communications*. IEEE Press.

James A and Milner B 2004 An analysis of interleavers for robust speech recognition in burst-like packet loss *Proc. of ICASSP'2004*. Montreal, Canada.

James A and Milner B 2005 Soft decoding of temporal derivatives for robust distributed speech recognition in packet loss *Proc. of ICASSP'2005*. Philadelphia, USA.

James A, Gomez A and Milner B 2004 A comparison of packet loss compensation methods and interleaving for speech recognition in burst-like packet loss *Proc. of ICSLP'2004*. Jeju Island, Korea.

Jayant N 1984 *Digital Coding of Waveforms*. Prentice-Hall.

Jiang W and Schulzrinne H 2000 Modeling of packet loss and delay and their effect on real time multimedia service quality *Proc. of NOSSDAV*. Chapel Hill, North Carolina, USA.

Juang B and Katagiri S 1992 Discriminative learning for minimum error classification. *IEEE Trans. on Signal Processing* **40**(12), 3043–3054.

Juang B, Rabiner L, Levinson S and Sondhi M 1985 Recent developments in the application of hidden Markov models to speaker-independent isolated word recognition *Proc. of ICASSP'1985*. Tampa, Florida, USA.

Kaiser J 1990 On a simple algorithm to calculate the 'energy' of a signal *Proc. of ICASSP'1990.* Albuquerque, New Mexico, USA.

Karray L, Jelloun A and Mokbel C 1998 Solutions for robust recognition over the GSM cellular network *Proc. of ICASSP'1998*. Seattle, Washington, USA.

Kelleher H, Pearce D, Ealy D and Mauuary L 2002 Speech recognition performance comparison between DSR and AMR transcoded speech *Proc. of ICSLP'2002*. Denver, USA.

Kim H and Cox R 2000 Bitstream-based feature extraction for wireless speech recognition *Proc. of ICASSP'2000*. Istanbul, Turkey.

Kim H and Cox R 2001a A bitstream-based front-end for wireless speech recognition on IS-136 communications system. *IEEE Trans. on Speech and Audio Processing* **9**(5), 558–568.

Kim H and Cox R 2001b Feature enhancement for bitstream-based front-end in wireless speech recognition *Proc. of ICASSP'2001*. Salt Lake City, Utah, USA.

Kim H, Cox R and Rose R 2002 Performance improvement of a bitstream-based front-end for wireless speech recognition in adverse environments. *IEEE Trans. on Speech and Audio Processing* **10**(8), 591–604.

Kiss I 2000 A comparison of distributed and network speech recognition for mobile communication systems *Proc. of ICSLP'2000*. Beijing, China.

Kiss I and Kapanen P 1999 Robust feature vector compression algorithm for distributed speech recognition *Proc. of Eurospeech'99*. Budapest, Hungary.

Kroon P, Deprettere E and Sluyter R 1896 Regular-pulse excitation – a novel approach to effective and efficient multipulse coding of speech. *IEEE Trans. on Acoustics Speech and Signal Processing* **34**, 1054–1063.

Kurose J and Ross K 2003 *Computer network: a top-down approach featuring the internet*. Addison-Wesley.

Laflamme C, Adoul J, Su H and Morisette S 1990 On reducing computational complexity of codebook search in CELP coders through the use of algebraic codes *Proc. of ICASSP'1990*. Albuquerque, New Mexico, USA.

Lee C 2003 On automatic speech recognition at the dawn of the 21st century. *IEICE Trans. on Infor. and Syst.* **E86-D**(3), 377–396.

Lee KF 1989 *Automatic Speech Recognition (The Development of the Sphinx System)*. Kluwer Academic Publishers.

Lee KF 1990 Context-dependent phonetic hidden markov models for speaker-independent continuous speech recognition. *IEEE Trans. on ASSP* **38**, 599–623.

Leggetter C and Woodland C 1995 Flexible speaker adaptation using maximum likelihood linear regression *Proc. Eurospeech'1995*, pp. 1155–1158, Madrid.

León-García A 1989 *Probability and Random Processes* Electrical and Computer Engineering. Addison-Wesley.

Levinson S, Rabiner L and Sondhi M 1983 An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *The Bell System Technical Journal* **62**(4), 1035–1074.

Li J, Liu B, Wang R and Dai L 2004 A complexity reduction of ETSI advanced front-end for DSR *Proc. of ICASSP'2004*. Montreal, Canada.

Lilly B and Paliwal K 1996 Effect of speech coders on speech recognition performance *Proc. of ICSLP'1996*. Philadelphia, PA, USA.

Lim J 1979 Spectral root homomorphic deconvolution system. *IEEE Trans. ASSP* **27**(3), 223–233.
Linde Y, Buzo A and Gray R 1980 An algorithm for vector quantizer design. *IEEE Trans. on Commun.* **28**(1), 84–95.
Lockwood P and Boudy J 1979 Experiments with a non-linear spectral subtractor (NSS) hidden markov models and the projection, for robust speech recognition in car. *Speech Communications* pp. 215–228.
Lockwood P, Baillargeat C, Gillot J, Boudy J and Faucon G 1991 Noise reduction for speech enhancement in cars: non linear spectral subtraction *Proc. Eurospeech'91*. Genova, Italia.
Macho D and Cheng Y 2001 SNR-dependent waveform processing for improving the robustness of ASR front-end *Proc. of ICASSP'2001*. Salt Lake City, Utah, USA.
Makhoul J 1975 Linear prediction: A tutorial review. *Proceedings of IEEE* **63**, 561–580.
Makhoul J, Roucos S and Gish H 1985 Vector quantization in speech coding. *Proc. of the IEEE* **73**, 1551–1588.
Markel J and Gray A 1976 *Linear Prediction of Speech*. Springer-Verlag.
Martens J 2000 Final report of COST action 249: speech recognition over the telephone. Technical report.
Martin P 1982 Comparison of pitch detection by cepstrum and spectral comb analysis *Proc. of ICASSP'1982*. Paris, France.
Martin R 1993 An efficient algorithm to estimate the instantaneous snr of speech signals. *Proc. EURSPEECH'93* **1**, 1093–1096.
Mauuary L 1998 Blind equalization in the cepstral domain for robust telephone based speech recognition *Proc. of EUSIPCO'1998*. Rhodes, Greece.
Mayorga P, Lamy R and Besacier L 2002 Recovering of packet loss for distributed speech recognition *Proc. of EUSIPCO'2002*. Toulouse, France.
McAulay R and Malpass M 1980 Speech enhancement using a soft-decision noise suppression filter. *IEEE Trans. on Acoustics, Speech, and Signal Processing* **28**(2), 137–145.
McCree A and Barnwell T 1995 A mixed excitation LPC vocoder model for low bit-rate speech coding. *IEEE Trans. on Speech and Audio Processing* **3**(4), 242–250.
Medan Y, Yair E and Chazan D 1991 Super resolution pitch determination of speech signals. *IEEE Trans. on Acoustics Speech and Siganal Processing* **39**, 40–48.
Milner B and James A 2003 Analysis and compensation of packet Loss in distributed speech recognition using interleaving *Proc. of Eurospeech'2003*, pp. 2693–2696. Geneva, Switzerland.
Milner B and James A 2004 An Analysis of Packet Loss Models for Distributed Speech Recognition *Proc. of ICSLP'2004*. Jeju Island, Korea.
Milner B and Semnani S 2000 Robust speech recognition over IP networks *Proc. of ICASSP'2000*. Istanbul, Turkey.
Milner B and Shao X 2003 Low bit-rate feature vector compression using transform coding and non-uniform bit allocation *Proc. of ICASSP'2003*. Hong-Kong, China.
Mokbel C, Mauuary L, Jouvet D and Monne J 1996 Comparison of several preprocessing techniques for robust speech recognition over both PSN and GSM networks *Proc. of EUSIPCO'96*. Triest, Italy.
Mokbel C, Mauuary L, Karray L, Jouvet D, Monne J, Simonin J and Bartkova K 1997 Towards improving ASR robustness for PSN and GSM telephone applications. *Speech Communication* (23), 141–159.
Molau S, Hikgher F, Keysers D and Ney H 2002 Enhanced histogram normalization in the acoustic feature space *Proc. of ICSPL'02*, vol. 1, pp. 1421–1424, Denver.
Moreno A, L.Borge, Christoph D, Gael R, Khalid C, Stephan E and Jeffrey A 2000 Speechdat-car: a large speech database for automotive environments *Proc. of LREC'2000*. Athens, Greece.
Moreno P and Stern R 1994 Sources of degradation of speech recognition in the telephone network *Proc. of ICASSP'94*, pp. I/109–I/112 IEEE. Adelaide, Australia.
Moreno PJ, Raj B and Stern RM 1998 Data-driven environment compensation for speech recognition: A unified approach. *Speech Communication* **24**(4), 267–288.
Morris A, Cooke M and Green P 1998 Some solutions to the missing feature problem in data classification, with application to noise robust ASR *Proc. of ICASSP'1998*. Seattle, Washington, USA.
Morris AC, Barker J and Bourlard H 2001 From missing data to maybe useful data: Soft data modelling for noise robust ASR *Proc. WISP'01*, Stratford-upon-Avon, UK.
Nadas A 1983 A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Trans. on ASSP* **31**(4), 814–817.
Nadeu C, Paches-Leal P and Juang BH 1997 Filtering the time sequence of spectral parameters for speech recognition. *Speech Communication* **22**, 315–332.

Nour-Eldin A, Tolba H and O'Shaugnessy D 2004 Automatic recognition of bluetooth speechin 802.11 interference and the effectiveness of insertion-based compensation techniques *Proc. of ICASSP'2004*. Montreal, Canada.

Oppenheim A and Schafer R 1975 *Digital Signal Processing*. Prentice-Hall, Inc.

Paliwal K and So S 2004a Multiple frame block quantisation of line spectral frequencies using Gaussian mixture models *Proc. of ICASSP 2004*. Montreal, Canada.

Paliwal K and So S 2004b Scalable distributed speech recognition using multi-frame GMM-based block quantization *Proc. of ICSLP 2004*. Jeju Island, Korea.

Pearce D 2000 Enabling New Speech Driven Services for Mobile Devices: An overview of the ETSI standards activities for Distributed Speech Recognition Front-ends *AVIOS 2000: The Speech Applications Conference*. San Jose, USA.

Pearce D 2004 Robustness to transmission channel – the DSR approach *Proc. of Robust'2004 (Cost 278 and ISCA-ITRW)*. Norwich, UK.

Peinado A, Gomez A, Sanchez V, Perez-Cordoba J and Rubio A 2005a Packet loss concealment based on VQ replicas and MMSE estimation applied to distributed speech recognition *Proc. of ICASSP'2005*. Philadelphia, USA.

Peinado A, Ramesh P and Roe D 1990 On the use of energy information for speech recognition using HMM *Proceedings of EUSIPCO-90*, vol. 2, pp. 1243–1246, Barcelona.

Peinado A, Sanchez V, Perez J and de la Torre A 2003 HMM-based channel error mitigation and its application to distributed speech recognition. *Speech Communication* **41**, 549–561.

Peinado A, Sanchez V, Perez J and Rubio A 2005b Efficient MMSE-based channel error mitigation techniques. Application to distributed speech recognition over wireless channels. *IEEE Trans. on Wireless Communications* **4**(1), 14–19.

Pelaez C, Gallardo A and Diaz F 2001 Recognizing voice over ip: a robust front-end for speech recognition on the world wide web. *IEEE Trans. on Multimedia* **3**(2), 209–218.

Pelaez C, Gallardo A, Vicente J and Dˆaz F 2002 Filtering the spectral parameters to mitigate the influence of transmission errors on ASR systems *Proc. of ICSLP'2002*. Denver, USA.

Pelecanos J and Sridharan S 2001 Feature warping for robust speaker verification *Proc. of Speaker Oddyssey 2001 conference*.

Perkins C, Hodson O and Hardman V 1998 A survey of packet loss recovery techniques for streaming audio. *IEEE Network* pp. 40–48.

Postel J 1980 RFC 768 – User Datagram Protocol. Technical Report RFC 768, IETF.

Postel J 1981a RFC 791 – Internet Protocol. Technical Report RFC 791, IETF.

Postel J 1981b RFC 793 – Transmission Control Protocol. Technical Report RFC 793, IETF.

Potamianos A and Weerackody V 2001 Soft-feature decoding for speech recognition over wireless channels *Proc. of ICASSP'2001*. Salt Lake City, Utah, USA.

Proakis J and Manolakis D 1996 *Digital Signal Processing* 3 edn. Prentice-Hall, Englewood Cliffs, NJ.

Rabiner L and Juang B 1993 *Fundamentals of Speech Recognition*. Prentice-Hall.

Rabiner L, Juang B, Levinson S and Sondhi M 1985a Recognition of isolated digits using hidden markov models with continuous mixture densities. *AT&T technical Journal* **64**(6), 1211–1233.

Rabiner L, Juang B, Levinson S and Sondhi M 1985b Some properties of continuous hidden markov model representations. *AT&T Technical Journal* **64**(6), 1251–1269.

Rabiner LR 1989 A tutorial on hidden markov models and selected applications in speech recognition *Proceedings of the IEEE*, vol. 77, pp. 257–285.

Rabiner LR and Schafer RW 1978 *Digital Processing of Speech Signals*. Prentice-Hall.

Rabiner LR, Levinson SE and Sondhi MM 1983 On the application of vector quantization and hidden markov models to speaker-independent, isolated word recognition. *The Bell System technical Journal* **62**(4), 1075–1105.

Rabiner LR, Sondhi MM and Levinson SE 1984 A vector quantizer incorporating both LPC shape and energy *Proc. of ICASSP 84*. San Diego, USA.

Ramaswamy G and Gopalakrishnan P 1998 Compression of acoustic features for speech recognition in network environments *Proc. of ICASSP'98*, pp. 977–980. Seattle, Washington, USA.

Ramirez J, Segura JC, Benitez C, de la Torre A and Rubio A 2004 Efficient voice activity detection algorithms using long-term speech information. *Speech Communication* **42**, 271–287.

Ramirez J, Segura JC, Benitez C, Garcia L and Rubio AJ 2005 Statistical voice activity detection using a multiple observation likelihood ratio test. *IEEE Signal Processing Letters* **12**(10), 689–692.

Riskin E 1991 Optimum bit allocation via generalized BFOS algorithm. *IEEE Trans. on Information Theory* **37**, 400–402.

Rizzo L 1997 Efective erasure codes for reliable computer communication protocols. *ACM Computer Commun. Rev* **27**, 24–36.

Rosenfeld R 2000 Two decades of statistical language modeling: where do we go from here?. *Proc. of the IEEE* **88**(8), 1270–1278.

Salami R, Laflamme C, Adoul J, Kataoka A, Hayashi S, Moriya T, Lamblin C, Massaloux D, Proust S, Kroon P and Shoham Y 1998 Design and description of CS-ACELP: a toll quality 8 kb/s speech coder. *IEEE Trans. on Speech and Audio Processing* **6**(2), 116–130.

Salonidis T and Digalakis V 1998 Robust Speech Recognition for Multiple Topological Scenarios of the GSM mobile phone system *Proc. of ICASSP'98*, pp. 101–104. Seattle, Washington, USA.

Sanchez V, Garcia P, Peinado A, Segura J and Rubio A 1995 Diagonalizing properties of the discrete cosine transforms. *IEEE Trans. on Signal Processing* **43**(22), 2631–2641.

Sanneck H and Carle G 1996 A framework model for packet loss metrics based on loss runlengths *Proc. of IEEE Global Internet*, pp. 554–557. London, England.

Sarikaya R and Hansen J 2001 Analysis of the root-cepstrum for acoustic modeling and fast decoding in speech recognition *Proc. of EUROSPEECH'2001*. Aalborg, Denmark.

Schroeder M and Atal B 1985 Code-excited linear prediction (CELP): high quality speech at very low bit rates *Proc. of ICASSP-85*, pp. 937–940. Tampa, USA.

Schulzrinne H, Casner S, Frederick R and Jacbson V 1996 RFC 1889 – RTP: a transport protocol for real-time applications. Technical Report RFC 1889, IETF.

Schwartz R, Chow Y, Kimball O, Roucos S, Krasner M and Makhoul J 1985 Context-dependent modeling for acoustic-phonetic recognition of continuous speech *Proc. of ICASSP-85*. Tampa, USA.

Sciver J, Ma J, Vanpoucke F and Hamme H 2002 Investigation on speech recognition over IP channels *Proc. of ICASSP'2002*, vol. 4, pp. 3812–15. Orlando, USA.

Segura J, Benitez M, de la Torre A, Rubio A and Ramirez J 2004 Cepstral domain segmental nonlinear feature transformations for robust speech recognition. *IEEE Signal Processing Letters* **11**(5), 517–520.

Shenker S and Wroklawski J 1997 RFC 2215 – General characterization parameters for integrated service network elements. Technical Report RFC 2215, IETF.

SIG 2001 Specification of the Bluetooth system – Core v1.1. Technical report, Bluetooth.

Skoglund M and Hedelin P 1994 Vector quantization over a noisy channel using soft decision *Proc. of ICASSP'1994*. Adelaide, Australia.

Skogstad T and Svendsen T 2005 Distributed ASR using speech coder data for efficient vector representation *Proc. of EUROSPEECH'2005*. Lisbon, Portugal.

Sohn J, Kim NS and Sung W 1999 A statistical model-based voice activity detection. *IEEE Signal processing letters,* **6**(1), 1–3.

Sorin A, Ramabadran T, Chazan D, Hoory R, McLaughlin M, Pearce D, Wang F and Zhang Y 2004 The ETSI extended distributed speech recognition (DSR) standards: client side processing and tonal language recognition evaluation *Proc. of ICASSP'2004*. Montreal, Canada.

Sphinx4 2004 A speech recognizer written entirely in the JavaTM programming language. Technical report, Carnegie Mellon University, http://cmusphinx.sourceforge.net/sphinx4/.

Srinivasamurthy N, Narayanan S and Ortega A 2001a Use of Model Transformations for Distributed Speech Recognition *Proc. of ISCA Tutorials and Research Workshops 2001*. Sophia-Antipolis, France.

Srinivasamurthy N, Ortega A and Narayanan S 2001b Efficient scalable compression for scalable speech recognition *Proc. of Eurospeech 2001*. Aalborg, Denmark.

Srinivasamurthy N, Ortega A and Narayanan S 2003 Towards optimal encoding for classification with applications to distributed speech recognition *Proc. of Eurospeech 2003*, pp. 1113–16. Geneva, Switzerland.

Srinivasamurthy N, Ortega A and Narayanan S 2004 Enhanced standard compliant distributed speech recognition (Aurora encoder) using rate allocation *Proc. of ICASSP 2004*, pp. I–485–488. Montreal, Canada.

Srinivasamurthy N, Ortega A, Zhu Q and Alwan A 2000 Towards efficient and scalable speech compression for robust speech recognition applications *Proc. of ICME'2000*. New York City, USA.

Srinivasan K and Gersho A 1993 Voice activity detection for cellular networks *Proc. of IEEE Workshop on Speech Coding for Telecomunications*, pp. 85–86. St. Jovite, Quebec, Canada.

Stallard D 1997 The BBN SPIN system *Proc. of Voice on the Net Conf.* Boston, USA.

Stallings W 2002 *Wireless communications and networks*. Prentice Hall.

Subramaniam A and Rao B 2003 PDF optimized parametric vector quantization of speech line spectral frequencies. *IEEE Trans. on Speech and Audio Processing* **11**(2), 130–142.

Sugamura N, Shikano K and Furui S 1983 Isolated word recognition using phoneme-like templates *Proc. of IEEE ICASSP'1983*, pp. 723–726. Boston, USA.

Suk YH, Choi SH and Lee HS 1999 Cepstrum third-order normalisation method for noisy speech recognition. *IEE Electronic Letters* **35**(7), 527–528.

Tan Z and Dalsgaard P 2002 Channel error protection scheme for distributed speech recognition *Proc. of ICSLP'2002*. Denver, USA.

Tan Z, Dalsgaard P and Lindberg B 2003 OOV-detection and channel error protection for distributed speech recognition over wireless networks *Proc. of ICASSP'2003*. Hong-Kong, China.

Tan Z, Dalsgaard P and Lindberg B 2004a A subvector-based error concealment algorithm for speech recognition over mobile networks *Proc. of ICASSP'2004*. Montreal, Canada.

Tan Z, Dalsgaard P and Lindberg B 2005 Automatic speech recognition over error-prone wireless networks. *Speech Communication* **47**, 220–242.

Tan Z, Lindberg B and Dalsgaard P 2004b A comparative study of feature-domain error concealment techniques for distributed speech recognition *Proc. of Robust'2004 (Cost 278 and ITRW workshop)*. Norwich, UK.

Tou JT and Gonzalez RC 1974 *Pattern Recognition Principles*. Addison-Wesley Publishing Company, Inc.

TR45 1996 Enhanced variable rate codec, speech service option 3 for wideband spread spectrum digital systems. Technical Report IS-127, TIA.

Tremain TE 1982 The Government Standard Linear Predictive Coding Algorithm: LPC-10. *Speech Technology Magazine* pp. 40–49.

Turunen J and Vlaj D 2001 A study of speech coding parameters in speech recognition *Proc. of EUROSPEECH'2001*. Aalborg, Denmark.

Tyagi V, McCowan I, Misra H and Bourlard H 2003 Mel-cepstrum modulation spectrum (MCMS) features for robust asr *Proc. of ASRU'2003*, pp. 399–404. US Virgin Islands, USA.

Vaisey J and Gersho A 1988 Simulated annealing and codebook design *Proc. of ICASSP'1988*, pp. 1176–1179. New York, USA.

Varga I, Aalburg S, Andrassy B, Astrov S, Bauer J, Beaugeant C, Geissler C and Hoge H 2002 ASR in mobile phones – an industrial approach. *IEEE Trans. on Speech and Audio Processing* **10**(8), 562–569.

Vaseghi S 2000 *Advanced Digital Signal Processing and Noise Reduction*. Wiley.

Vaseghi SV and Milner BP 1993 Noisy speech recognition based on HMM's, Wiener filters and re-evaluation of most likely candidates. *Proc. of ICASSP'93* **2**, 103–106.

Vaseghi SV and Milner BP 1997 Noise compensation methods for hidden markov models speech recognition in advese environments. *IEEE Trans. on Speech and Audio Processing* **5(1)**, 11–21.

Viikki O and Laurila K 1998 Cepstral domain segmental feature vector normalization for noise robust speech recognition. *Speech Communication* **25**, 133–147.

Villarrubia L and Hernandez L 2001 Reconocimiento de voz en el entorno de las nuevas redes de comunicacion UMTS e Internet. *Comunicaciones de Telefonica I+D* (23), 99–112.

Vilmur RJ, Barlo JJ, Gerson IA and Lindsey BL 1989 United States Patent Number 4,811,404: Noise Suppression System. Technical Report US Patent 4811404, Motorola, Inc.

Vintsjuk TK 1968 Recognition of words of oral speech by dynamic programming. *Kibernetica*.

Wang H and Moayeri N 1995 Finite-state markov channel: a useful model for radio communication channels. *IEEE Trans. on Vehic. Tech.* **44**(1), 163–171.

Weerackody V, Reichl W and Potamianos A 2002 An Error-Protected Speech Recognition System for Wireless Communications. *IEEE Trans. on Wireless Communications* **1**(2), 282–291.

Woodland G 1996 Mean and variance adaptation within the mllr framework. *Computer Speech & Language* **10**, 249–264.

Xiang B, Chaudhari UV, Navratil J, Ramaswamy GN and Gopinath RA 2002 Short-time gausssianization for robust speaker verification *Proc. of ICASSP'02*, pp. 681–684, Orlando, Florida.

Xie Q 2003 RFC 3557 – RTP Payload Format for European Telecommunications Standards Institute (ETSI) European Standard ES 201 108 Distributed Speech Recognition Encoding. Technical Report RFC 3557, IETF.

Xie Q and Pearce D 2005 RFC 4060 – RTP Payload Format for European Telecommunications Standards Institute (ETSI) European Standards ES 202 050, ES 202 211, and ES 202 212 Distributed Speech Recognition Encoding. Technical Report RFC 4060, IETF.

Yajnik M, Moon S, Kurose J and Towsley D 1999 Measurement and modelling of the temporal dependence in packet loss *Proc. of IEEE Infocom*, pp. 345–352. New York, USA.

Young S, Kershaw D, Odell J, Ollason D, Valtchev V and Woodland P 2000 *The HTK Book*. Microsoft Corporation.

Yu A and Wang H 1998 A study on the recognition of low bit-rate encoded speech *Proc. of ICSLP'98*. Sydney, Australia.

Zhong X and Juang B 2002 Multiple description speech coding with diversities *Proc. of ICASSP'2002*. Orlando, USA.

Zhong X, Arrowood J and Clements M 2002a Speech coding and transmission for improved automatic recognition *Proc. of ICSLP'2002*. Denver, USA.

Zhong X, Arrowood J, Moreno A and Clements M 2002b Multiple description coding for recognizing voice over IP *Proc. of the 10th Digital Signal Processing Workshop*. Pine Mountain, Georgia, USA.

Zhu Q and Alwan A 2001 An efficient and scalable 2D DCT-based feature coding scheme for remote speech recognition *Proc. of ICASSP'2001*. Salt Lake City, Utah, USA.

Zwicker E and Fastl H 1990 *Psychoacoustics*. Springer-Verlag.

List of Acronyms

ACELP	algebraic CELP
ADPCM	adaptive differential pulse code modulation
AFE	advanced front end
AMPS	advanced mobile phone system
AMR	adaptive multirate
ASR	automatic speech recognition
AWGN	additive white Gaussian noise
BER	bit error rate
BFI	bad frame indicator
B-NSR	bitstream-based NSR
bps	bits per second
BSC	base station controller
BSS	base station system
BTS	base transceiver station
CDF	cumulative density function
CDMA	code division multiple access
CELP	code excited linear prediction
CHMM	continuous HMM
C/I	carrier-to-interference ratio
CMN	cepstral mean normalization
CMVN	cepstral mean and variance normalization
CRC	cyclic redundancy check
CS-ACELP	conjugate structure ACELP
CSR	continuous speech recognition
CTN	cepstrum third-order normalization
DARPA	defence advanced research project agency
DCT	discrete cosine transform
DHMM	discrete HMM
DPCM	differential pulse code modulation
DPSK	differential phase shift keying
DSR	distributed speech recognition

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura

DTW	dynamic time warping
EC	error concealment
EFR	enhanced full rate
EGPRS	enhanced GPRS
ETSI	European telecommunications standards institute
EVRC	enhanced variable rate coder
FBMMSE	forward–backward MMSE estimation
FDD	frequency division duplex
FEC	forward error correction
FE	front end
FMMSE	forward MMSE estimation
FP	frame pair
FR	full rate
GMM	Gaussian mixture model
GSM	group special mobile
HEQ	histogram equalization
HMM	hidden Markov model
HR	half rate
IETF	internet engineering task force
iLBC	internet low bitrate codec
IMT	international mobile communications
IP	internet protocol
IWR	isolated word recognition
kbps	kilobit per second
KLT	Karhunen–Loeve transform
LAR	log-area ratio
LBND	low-frequency band noise detector
LD-CELP	low delay CELP
LFCC	linear frequency cepstrum coefficients
LMS	least mean squares
log-FBE	logarithmic filterbank energies
LPCC	linear prediction cepstrum coefficients
LPC	linear prediction
LSP	line spectrum pair
LTP	long-term prediction
LTSD	long-term spectral divergence
LTSE	long-term spectral envelope
MAI	multiple access interference
MAPLR	maximum a posteriori linear regression
MAP	maximum a posteriori
Mbps	megabit per second
MCE	minimum classification error
MDC	multiple description coding
MELP	mixed excitation linear prediction
MFCC	mel-frequency cepstral coefficients
MIRS	motorola integrated radio system

MIME	multipurpose internet mail extensions
MLLR	maximum likelihood linear regression
ML	maximum likelihood
MMI	maximum mutual information
MMSE-LSA	minimum mean square error log spectral estimation
MMSE	minimum mean square error
MOS	mean opinion score
MSC	mobile services switching center
MSE	mean square error
NSR	network-based speech recognition
NSS	network subsystem
OOV	out-of-vocabulary
PCM	pulse code modulation
PDA	personal digital assistant
PDC	personal digital cellular
PDF	probability density function
PLP	perceptual linear prediction
PSTN	public switched telephone network
PTS	pitch tracking and smoothing
QCELP	Qualcomm CELP
QMF	quadrature mirror filter
RASTA	relative spectral processing
RCELP	relaxed CELP
RFC	request for comment
RLC	radio link control
RSR	remote speech recognition
RTP	real-time protocol
SCHMM	semicontinuous HMM
SDP	session description protocol
SES	speech-enabled services
SGSN	serving GPRS support node
SIP	session initiation protocol
SMQ	split matrix quantization
SNR	signal-to-noise ratio
S-NSR	speech-based NSR
STP	short-term prediction
STQ	speech, transmission planning and quality of service
SVQ	split vector quantization
TCH	traffic channel
TCP	transmission control protocol
TC	tranform coding
TDD	time division duplex
TDMA	time division multiple access
TETRA	terrestrial trunked radio
TFO	tandem free operation
TRAU	transcoding rate and adaptation unit

TSVQ	tree-structured VQ
UDP	user datagram protocol
UEP	unequal error protection
UMTS	universal mobile telephone system
UTRA	UMTS Terrestrial Radio Access
VADNest	VAD for noise estimation
VADVC	VAD for voicing classification
VAD	voice activity detection
VA	Viterbi algorithm
VoIP	voice over IP
VQ	vector quantization
VSELP	vector sum excited linear prediction
WAcc	word accuracy
W-CDMA	wideband CDMA
WER	word error rate
WF	Wiener filtering
WVA	weighted Viterbi algorithm
XAFE	extended advanced front end
XFE	extended front end

Index

Speech Recognition Over Digital Channels Antonio Peinado and José C. Segura