TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Jani Nurminen

A Parametric Approach for Efficient Speech Storage, Flexible Synthesis and Voice Conversion

Julkaisu 1156 • Publication 1156

Tampere 2013

Tampereen teknillinen yliopisto. Julkaisu 1156
Tampere University of Technology. Publication 1156

Jani Nurminen

A Parametric Approach for Efficient Speech Storage, Flexible Synthesis and Voice Conversion

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB111, at Tampere University of Technology, on the 4th of October 2013, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2013

ISBN 978-952-15-3136-1 (printed)
ISBN 978-952-15-3157-6 (PDF)
ISSN 1459-2045

Abstract

During the past decades, many areas of speech processing have benefited from the vast increases in the available memory sizes and processing power. For example, speech recognizers can be trained with enormous speech databases and high-quality speech synthesizers can generate new speech sentences by concatenating speech units retrieved from a large inventory of speech data. However, even in today's world of ever-increasing memory sizes and computational resources, there are still lots of embedded application scenarios for speech processing techniques where the memory capacities and the processor speeds are very limited. Thus, there is still a clear demand for solutions that can operate with limited resources, e.g., on low-end mobile devices.

This thesis introduces a new segmental parametric speech codec referred to as the VLBR codec. The novel proprietary sinusoidal speech codec designed for efficient speech storage is capable of achieving relatively good speech quality at compression ratios beyond the ones offered by the standardized speech coding solutions, i.e., at bitrates of approximately 1 kbps and below. The efficiency of the proposed coding approach is based on model simplifications, mode-based segmental processing, and the method of adaptive downsampling and quantization. The coding efficiency is also further improved using a novel flexible multi-mode matrix quantizer structure and enhanced dynamic codebook reordering. The compression is also facilitated using a new perceptual irrelevancy removal method.

The VLBR codec is also applied to text-to-speech synthesis. In particular, the codec is utilized for the compression of unit selection databases and for the parametric concatenation of speech units. It is also shown that the efficiency of the database compression can be further enhanced using speaker-specific retraining of the codec. Moreover, the computational load is significantly decreased using a new compression-motivated scheme for very fast and memory-efficient calculation of concatenation costs, based on techniques and implementations used in the VLBR codec.

Finally, the VLBR codec and the related speech synthesis techniques are complemented with voice conversion methods that allow modifying the perceived speaker identity which in turn enables, e.g., cost-efficient creation of new text-to-speech voices. The VLBR-based voice conversion system combines compression

with the popular Gaussian mixture model based conversion approach. Furthermore, a novel method is proposed for converting the prosodic aspects of speech. The performance of the VLBR-based voice conversion system is also enhanced using a new approach for mode selection and through explicit control of the degree of voicing.

The solutions proposed in the thesis together form a complete system that can be utilized in different ways and configurations. The VLBR codec itself can be utilized, e.g., for efficient compression of audio books, and the speech synthesis related methods can be used for reducing the footprint and the computational load of concatenative text-to-speech synthesizers to levels required in some embedded applications. The VLBR-based voice conversion techniques can be used to complement the codec both in storage applications and in connection with speech synthesis. It is also possible to only utilize the voice conversion functionality, e.g., in games or other entertainment applications.

Preface

The majority of the research work presented in this thesis was carried out in 2002–2006, mostly while working at Nokia Research Center, and the thesis was finalized in 2012–2013 while working at the Department of Signal Processing, Tampere University of Technology. The work at Nokia Research Center was partially funded by the European Union under the integrated project TC-STAR – Technology and Corpora for Speech-to-Speech Translation – (IST-2002-FP6-506738, http://www.tc-star.org), and the work at Tampere University of Technology was in part funded by Academy of Finland.

First and foremost, I would like to express my deep and sincere gratitude to my supervisor Prof. Moncef Gabbouj for all his support and guidance, and for giving me the opportunity to finalize my thesis in his team. I would also like to thank the pre-examiners of my thesis, Prof. Paul Micallef and Dr. Aki Härmä for their efforts in reviewing the manuscript. I am also grateful to Prof. Mikko Kurimo for agreeing to serve as the opponent in the public defense of this thesis.

Next, I would like to warmly thank MSc. Hanna Silén for carefully peer reviewing this thesis, as well as for our fruitful and pleasant collaboration, reaching well beyond the scope of this thesis. I also owe special thanks to the other coauthors of the publications and patents forming the basis of this thesis: Dr. Feng Ding, Dr. Ari Heikkinen, Dr. Elina Helander, Mr. Sakari Himanen, Dr. Imre Kiss, Dr. Marja Mettänen (Lähdekorpi), Dr. Victor Popa, MSc. Anssi Rämö, Dr. Jukka Saarinen, Dr. Yuezhong Tang, Dr. Jilei Tian, and MSc. Janne Vainio – it has always been a real pleasure to work with all of you.

Moreover, I am thankful to all of my former and current colleagues both at Nokia and at Tampere University of Technology for always creating a friendly and fun working environment regardless of the organizational details. I would also like to thank everybody that I have collaborated with in my research over the years, both in Finland and abroad. To me, different kinds of collaborations always not only make the work more productive but also infinitely more enjoyable.

The financial support provided by Emil Aaltonen Foundation, Nokia Foundation, and TTY:n tukisäätiö is gratefully acknowledged.

Finally, I would like to thank my family and friends for making the world such an exciting place. Especially, I want to express my gratitude to my parents Irma

and Erkki for their continued support throughout my life. I also wish to thank my children, Laura and Mika, for all the moments we have shared together, and for their patience when I was finalizing my thesis. Last but not least, I am deeply grateful to Katja for her love and support during the writing process, as well as for proofreading this thesis.

Tampere, September 2013

Jani Nurminen

Contents

Abstract i

Preface iii

List of figures ix

List of tables xi

List of abbreviations xiii

1 Introduction 1
- 1.1 Scope of the thesis and the main objectives . . . 2
- 1.2 Main contributions . . . 3
- 1.3 Thesis outline . . . 5

2 Overview of speech processing 7
- 2.1 Speech production, perception and processing . . . 7
  - 2.1.1 Speech production . . . 8
  - 2.1.2 Speech perception . . . 9
  - 2.1.3 Processing of discrete-time speech signals . . . 11
- 2.2 Speech coding . . . 12
  - 2.2.1 Linear prediction and line spectral frequencies . . . 12
  - 2.2.2 Speech coding at low bitrates . . . 14
- 2.3 Quantization . . . 17
  - 2.3.1 Vector quantization . . . 17
  - 2.3.2 Multistage vector quantization . . . 20
  - 2.3.3 Predictive vector quantization . . . 23
- 2.4 Text-to-speech synthesis . . . 26
  - 2.4.1 Overview of the text-to-speech process . . . 27
  - 2.4.2 Acoustic synthesis . . . 28
  - 2.4.3 Concatenative synthesis and unit selection . . . 29
- 2.5 Voice conversion . . . 31
  - 2.5.1 Requirements for the training data . . . 32

2.5.2 Domain of conversion 33
2.5.3 Conversion methods 33

3 VLBR – segmental speech coding for efficient storage 35
3.1 Parametric representation 36
3.1.1 Excitation modeling 36
3.1.2 Parameter estimation 38
3.1.3 Speech signal reconstruction 40
3.2 VLBR speech codec 42
3.2.1 Overview of the proposed coder structure 42
3.2.2 Segmentation 44
3.2.3 Adaptive downsampling and quantization 46
3.2.4 Update rate and vector dimension conversions 47
3.2.5 Performance evaluation 49
3.3 Vector-predictive multi-mode matrix quantization 52
3.3.1 Proposed quantizer structure 52
3.3.2 Quantizer training 55
3.3.3 Performance evaluation 58
3.4 Enhanced dynamic codebook reordering 60
3.4.1 Conventional dynamic codebook reordering 61
3.4.2 Enhanced dynamic codebook reordering 64
3.4.3 Experimental results 65
3.5 Improvement of coding efficiency via preprocessing 68
3.5.1 Proposed approach for perceptual irrelevancy removal . . 69
3.5.2 Performance evaluation in isolation 71
3.5.3 Performance evaluation with speech codecs 73
3.5.4 Discussion 75
3.6 Conclusions 76

4 VLBR-based concatenative speech synthesis 77
4.1 Use of VLBR for the compression of TTS databases 78
4.1.1 Overview of VLBR-based concatenative synthesis 79
4.1.2 Database compression 83
4.1.3 Experimental findings and discussion 84
4.2 Dynamic quantizers and codec retraining 86
4.2.1 Making the quantizers dynamic 86
4.2.2 Practical experiments: Is speaker-specific retraining useful? 88
4.3 Compression-motivated method for computing concatenation costs 91
4.3.1 Concatenation cost calculation in unit selection TTS . . . 92
4.3.2 Computational load of concatenation cost calculation . . . 93
4.3.3 Proposed concatenation cost calculation technique 94
4.3.4 Experiments 98
4.4 Conclusions 100

5 VLBR-based voice conversion **103**
5.1 VLBR-based voice conversion system 104
5.1.1 Training data alignment 104
5.1.2 Model training and the conversion function 105
5.1.3 Conversion of the VLBR parameters 106
5.1.4 Performance evaluation 107
5.2 Prosody conversion 110
5.2.1 Conventional methods for prosody conversion 111
5.2.2 Overview of the proposed method 112
5.2.3 Model training and usage 113
5.2.4 Performance evaluation 117
5.3 Data clustering and mode selection 120
5.3.1 Proposed approach for data clustering and mode selection 120
5.3.2 Experimental results 123
5.4 Voicing level control 125
5.4.1 Unwanted changes in voicing 125
5.4.2 Voicing control 127
5.4.3 Experiments on voicing control 128
5.5 Conclusions 130

6 Conclusions and future work **131**

Bibliography **135**

List of figures

2.1 Some of the organs involved in speech production. (From [Wik13].) 8
2.2 Block diagram demonstrating parametric encoding and decoding of speech. (From [Nur01a].) 15
2.3 Block diagram of a multistage vector quantizer using sequential search. (From [Nur01a].) 21
2.4 Example of M-L tree search procedure with $M = 4$ in a 4-stage VQ. (From [Nur01a].) 22
2.5 Predictive vector quantizer. (From [Nur01a].) 24
2.6 Functional diagram of a TTS system. 27
2.7 Block diagram illustrating stand-alone voice conversion. (From [Nur12].) 32

3.1 Three examples illustrating the use of different playback speeds. (From [Nur06b].) 41
3.2 Proposed speech coder structure. (From [Räm04].) 42
3.3 Segmental nature of speech (the frame length used in these plots is 10 ms). (From [Räm04].) 44
3.4 Practical example illustrating the segmentation process. (From [Räm04].) 46
3.5 Algorithm for searching the optimal downsampling ratio. (From [Räm04].) 48
3.6 Average spectral distortion obtained with the proposed multi-mode quantizer and with the basic matrix quantizer at different bit error rates. Both quantizers operated at the bitrate of 20 bits/vector. (From [Nur03b].) 60
3.7 Index probabilities at the last stage of the 4-stage quantizer. (From [Nur06a].) 68
3.8 Block diagram of the preprocessing function. (From [Läh03b].) . 69
3.9 Example of masking threshold calculation. (From [Läh03b].) . . . 70
3.10 Combined MOS results with 95% confidence intervals. The conditions are listed in Table 3.12. (From [Läh03b].) 75

4.1 Simplified block diagram demonstrating the use of the VLBR codec in a concatenative TTS system. The database is compressed using the principles described in this section. 80
4.2 Concrete memory savings for databases of different size (for the speakers *slt* and *rms*). 90

5.1 K-means based clustering of target data vs. voiced/unvoiced clustering. The line illustrates the division between the two K-means based clusters while o and x denote voiced and unvoiced data, respectively. It is easy to see that there is significantly less variability within each cluster when the clustering is performed using target data instead of voicing decisions. (From [Nur06e].) 123
5.2 Level of voicing before (dashed line) and after conversion (solid line). (From [Nur07c].) 129

List of tables

3.1 Update rates used for the speech parameters during different segment types. The symbol - indicates that the corresponding information is not needed. In the ~1.0 kbps mode, the amplitudes were set to a fixed value, whereas in the ~2.4 kbps mode they were coded for all active frames (100 Hz). 47

3.2 Bit allocations for the sinusoidal coders evaluated in the listening test. For the proposed coder, the bit allocations depend on the input and thus the numbers given in the table are averages obtained using an exemplary speech file (the duration of this speech sample was 10 minutes and the speech activity level was 90%). 50

3.3 Listening test results (MOS scale 1–5). The absolute numbers do not carry any inherent meaning but the relative differences are meaningful. 51

3.4 Performance of the proposed vector-predictive multi-mode quantizer, a conventional matrix quantizer and a vector quantizer in an error-free environment. All the quantizers operate at the fixed bitrate of 20 bits/vector. 59

3.5 Theoretical bitrates achievable using a 4-stage MSVQ for LSFs (originally 2200 bps). 66

3.6 Theoretical bitrates achievable using a 3-stage MSVQ for LSFs (originally 2200 bps). 66

3.7 Theoretical bitrates achievable using a 5-stage MSVQ for LSFs (originally 2200 bps). 66

3.8 Theoretical bitrates achievable using a 5-stage PMSVQ for LSFs (originally 2200 bps). 67

3.9 Average scores of the reference samples. 72

3.10 Results of the CCR test. The table lists the average scores for the preprocessed speech with respect to the original unprocessed speech $\pm$ the 95% confidence interval. 72

3.11 Segmental SNR (in dB) between the input and output of the AMR codec with original and preprocessed signals. The improvement percentages are also shown. 73

3.12 Results of the ACR test with the two standardized codecs. 74

4.1 Performance of multistage LSF vector quantizers of different sizes (using at most 6 bits per stage) for the databases *slt* and *rms*, measured using spectral distortion in dB. The left column for each speaker presents the results obtained using database-specific retraining whereas the right column contains the results obtained using quantizers trained with generic multi-speaker data. Gray background is used for highlighting some cases where roughly similar or slightly better performance was achieved using the proposed retraining than with the generic quantizers despite the drop in the bitrate. 89
4.2 Memory usage in kilobytes using the conventional uncompressed approach and the proposed approach. 99
4.3 Pair-wise comparison between the baseline and the proposed approach: the average score and the 95% confidence interval. 99
4.4 MOS evaluation between the baseline and the proposed scheme, including 95% confidence intervals. 100

5.1 Scale used for evaluation of speaker identity. The listeners were asked to evaluate whether the two samples in the given pair were spoken by the same person or not. The real target speaker was used as the reference speaker. 108
5.2 Scale used in the evaluation of speech quality 108
5.3 Results from the first part of the evaluation (speaker identity, with the target speaker used as the reference in every sample pair). F denotes a female and M a male speaker. The column *Average* shows the combined score for all the directions. 108
5.4 Results achieved from the second part of the evaluation (speech quality). 109
5.5 Preference votes given to the proposed approach and to the GMM-based approach, and the "no preference" votes (equal). 119
5.6 Comparison between the conversion MSE achieved using the conventional voiced/unvoiced clustering and the proposed data-driven clustering schemes. 124
5.7 Direction of the change in the overall level of voicing after voice conversion in the test material (percentage of frames). The voicing values are not changed in the conversion but the effective degree of voicing changes due to the spectral modifications. 129

List of abbreviations

ACL	Asymptotic closed-loop
ACR	Absolute category rating
AMR	Adaptive multi-rate
AR	Auto-regressive
CART	Classification and regression tree
CELP	Code excited linear prediction
CW	Characteristic waveform
DCR	Dynamic codebook reordering
DCT	Discrete cosine transform
DTW	Dynamic time warping
EM	Expectation maximization
FFT	Fast Fourier transform
GLA	Generalized Lloyd algorithm
GMM	Gaussian mixture model
GSM	Global system for mobile communications
HMM	Hidden Markov model
IRS	Intermediate reference system
kbps	Kilobits per second
LP	Linear prediction
LPC	Linear predictive coding

LSF	Line spectral frequency
MA	Moving average
MBE	Multi-band excitation
MCC	Mel-cepstral coefficient
MELP	Mixed excitation linear prediction
MFCC	Mel-frequency cepstral coefficient
MNRU	Modulated noise reference unit
MOS	Mean opinion score
MQ	Matrix quantization
MSVQ	Multistage vector quantization
MV	Mean-variance
NSTVQ	Non-square transform vector quantization
PCM	Pulse code modulation
PSMVQ	Predictive multistage vector quantization
PVQ	Predictive vector quantization
REW	Rapidly evolving waveform
SD	Spectral distortion
SEW	Slowly evolving waveform
SNR	Signal-to-noise ratio
TTS	Text-to-speech
VLBR	Very low bitrate (name of the codec)
VQ	Vector quantization
WI	Waveform interpolation
WMOPS	Weighted million operations per second
WMSE	Weighted mean squared error
XOR	Exclusive or

Chapter 1

Introduction

Speech is generally regarded as the most natural and intuitive form of communication between humans. In human-machine interaction, other means of communication have been dominant and the role of speech has traditionally been rather small. However, thanks to the recent advances in the related fields of technology, such as speech synthesis, speech recognition and dialogue management, as well as the recent trend of minimizing the costs related to manual labor, e.g., in customer service, it seems likely that voice-based user interface solutions are going to become increasingly popular in the future.

In mobile devices, speech technology has other uses than just those related to the user interfaces. The first and the most obvious example is the usage of speech coding solutions to enable real-time mobile phone calls. In addition, there are also many other potential applications for speech technology. Examples of such applications include storage and listening of audiobooks, recording and storage of discussions during a meeting or a lecture (assuming that the local legislation allows this), personal voice memos, voice dialing, as well as dictation applications.

This thesis deals firstly with the topic of efficient speech storage. In the work described in this thesis, the goal was to go well beyond the compression ratios offered by the standardized speech coding solutions designed for real-time processing of conversational speech, e.g., during mobile phone calls. The main outcome of this work was the very low bitrate codec capable of achieving relatively good speech quality at bitrates of about 1.0 kbps (*kilobits per second*), referred to as the VLBR codec.

In addition to introducing the VLBR codec and some of the main techniques contributing to its efficiency, this thesis also discusses certain issues related to the topic of speech synthesis. Particular emphasis is placed on the compression of speech databases needed in concatenative unit selection based speech synthesis using the VLBR codec. Furthermore, the topics of VLBR-based concatenation and signal generation are also discussed, and some methods for further reducing the footprint and the complexity are introduced as well.

The above parts of the work were carried out mostly in 2002–2006, i.e., during a period when the memory capacities of mobile devices were typically rather small and when the memory requirements related to unit selection based synthesis were severely limiting the usefulness of this otherwise successful synthesis approach. Regardless of the recent increases in the memory sizes of mobile devices and smart phones, the ability to store speech data efficiently is still beneficial today: It is a common experience of many users of personal computers that no matter how big the memory or the hard drive is, it is likely that the space will run out sooner rather than later.

In addition to the speech coding and speech synthesis related issues, this thesis also deals with the topic of voice conversion. The research field of voice conversion relates to the conversion of the perceived speaker identity. In the context of speech synthesis, voice conversion techniques can be used for creating new synthetic voices. In this thesis, the topic is approached from the perspectives of the VLBR codec and VLBR-based speech synthesis.

1.1 Scope of the thesis and the main objectives

Since the content of this thesis covers many areas of speech processing, such as speech coding, speech synthesis and voice conversion, the scope is narrowed down heavily to keep the size of the thesis manageable. As the first rule, only those parts of the author's work on speech coding, speech synthesis and voice conversion that are directly related to the VLBR codec are included in this thesis. Furthermore, since a full and detailed description of the VLBR codec alone, together with all of the relevant background information, would already most likely require hundreds of pages of text, the scope is further narrowed down by focusing only on the narrowband version of the codec and on items that contain the strongest personal contributions from the author and at the same time contain important novelties compared to other similar solutions proposed in the literature. Also, since most of the work was carried out while working in the industry, further balancing acts were needed to keep all the confidential aspects of the work outside the thesis while still providing all the pieces of information necessary for keeping the discussions academically relevant.

In the discussions related to speech synthesis, only the acoustic synthesis part of the synthesis process is considered. The focus is also further placed only on unit selection based concatenative speech synthesis, leaving out closely related work done on statistical speech synthesis and hybrid synthesis. Moreover, all the other criteria explained above are also used for limiting the scope. In particular, only those parts of the work that deal with VLBR-based synthesis are included. A similar approach is chosen with the topic of voice conversion, i.e., only the work carried out particularly in the context of the VLBR-based voice conversion system is included. Furthermore, in the discussions related to spectral voice conversion,

only the work based on the use of Gaussian mixture model based conversion is included to further limit the scope.

The first primary goal in the work described in this thesis was to develop high-quality methods for efficient speech storage. More precisely, the aim was to achieve roughly the speech quality level of traditional solutions operating at 2.4 kbps and above but with much lower bitrates (in the neighborhood of 1.0 kbps). In addition to the design goals related to the compression efficiency and speech quality, the leading design constraints concerned the run-time memory usage and the computational complexity of the decoder. The second main objective was successful application of the developed speech storage solutions for efficient compression of text-to-speech unit selection databases. The main driver behind this objective was the inhibitively large memory consumption of the unit selection based text-to-speech systems. The third main objective was the development of compatible voice conversion solutions that would allow creation of new voices in VLBR-based synthesis. All of these main objectives were commercially motivated but the work also produced results with academic value.

1.2 Main contributions

This thesis introduces a novel parametric framework that is suitable for highly efficient speech storage, flexible speech synthesis and voice conversion. At a more detailed level, the main contributions of this thesis work are:

- Development of a new segmental parametric speech coding approach capable of achieving relatively good speech quality at very low bitrates (at about 1 kbit/s or below). The resulting very low bitrate speech codec referred to as the VLBR codec and the related novel techniques are outlined in Chapter 3. In particular, the first developmental version of the codec and an overview of the overall segmental coding approach are introduced in Section 3.2. (The segmental coding approach used already in this first version of the VLBR codec was originally first patented [Räm05] and then academically published in [Räm04].)

- In addition to the significant amounts of general development, implementation and team leading work, the most important of author's contributions to the first version of the VLBR codec described in Section 3.2 was the idea to utilize adaptive parameter downsampling and quantization techniques, together with mode-based segmental operation and variable bitrates. (These parts of the work were also included in the descriptions presented originally in [Räm05] and [Räm04].)

- A novel method for efficient compression based on multi-mode matrix quantization of adjacent parameters using a low-complexity vector-based predic-

tion scheme. The proposed quantizer structure described in Section 3.3 is very flexible and it can be used more widely than just in the VLBR codec. In addition to the introduction of the quantizer structure, an algorithm for training quantizers having the proposed structure is introduced as well. (Originally published in [Nur03b].)

- Enhanced dynamic codebook reordering for advanced quantizer structures, introduced in Section 3.4. The proposed enhancements extend the applicability of the dynamic codebook reordering method from basic vector quantizers to more complicated quantizer structures. In the context of the VLBR codec, the proposed approach enables significant further bitrate reductions through lossless compression of the reordered codebook indices. (Originally first patented [Nur07a] and then academically published in [Nur06a].)

- A preprocessing method for improving the compression efficiency of narrowband speech codecs. This novel preprocessing approach described in Section 3.5 is based on the author's ideas related to perceptual irrelevancy removal. Even though the method can be used for further enhancing the efficiency of the VLBR codec, it is not tied to any particular speech codec and it can even be used to enhance the coding efficiency of standardized speech codecs. (Originally published in [Läh03b], and also made public in a Master's thesis [Läh03a] supervised by the author.)

- Development of a VLBR-based concatenative text-to-speech back end allowing highly efficient speech database compression and high-quality concatenation. The synthesis approach described in Section 4.1 also facilitates flexible parametric modifications. In addition, the simple playback speed alteration technique, discussed in Section 3.1.3, can be used for modifying the timing-related prosodic aspects of the output speech. (Originally published in the patent application [Nur07b].)

- Introduction of the simple but effective concept of dynamic quantizer structures. As discussed in Section 4.2, the proposed idea enables flexible codec retraining and run-time quantizer updates, which in turn enhance the coding efficiency, e.g., in the case of text-to-speech database compression. (Originally published in the patent [Nur11b] and then partially published in [Nur13a].)

- A novel compression-motivated method for very fast computation of concatenation costs. The proposed method is based on the author's ideas and implementations related to the use of multistage vector quantization and pseudo-gray coding for computationally-efficient approximation of concatenation costs. Even though originally developed for VLBR-based synthesis, using building blocks also used in the VLBR codec, the method is more

widely applicable for concatenative speech synthesis as demonstrated by the results provided in Section 4.3. (Originally published in [Din08].)

- Development of a VLBR-based voice conversion system. The first version of this voice conversion system, introduced in Section 5.1, utilizes the classic Gaussian mixture model based conversion functions. (Originally first patented [Nur06d] and then academically published in [Nur06c].)

- Development of an enhanced version of the VLBR-based voice conversion system that not only converts spectral information but prosody as well. The novel prosody conversion method discussed in Section 5.2 can be applied to practically any voice conversion system, in addition to the evident use in VLBR-based voice conversion. (Originally first patented [Nur11a] and then academically published in [Hel07a].)

- A novel method for data clustering and mode selection. The method described in Section 5.3 was originally designed for enhancing the conversion accuracy in VLBR-based voice conversion but the same approach could be applied in other types of voice conversions systems, too. (Originally first patented [Tia10] and then academically published in [Nur06e].)

- New thinking related to unwanted changes in the effective degree of voicing, and a new approach for explicit control of voicing in voice conversion to avoid this problem. The use of the approach described in Section 5.4 reduces the conversion-induced noise and enhances the speech quality in VLBR-based voice conversion. Similar benefits could be enjoyed in other voice conversion systems that allow the required control of the effective degree of voicing. (Originally first patented [Nur08a] and then academically published in [Nur07c].)

A separate section is dedicated to each of these main contributions.

1.3 Thesis outline

This thesis is organized as follows. After the introduction provided in Chapter 1, Chapter 2 provides background information on the aspects of speech processing that are the most relevant from the viewpoint of the work described in this thesis. The main topics covered in the chapter include fundamental issues such as speech production and perception, as well as introductions to the main areas of speech processing covered in this thesis, i.e., speech coding, speech synthesis and voice conversion. A separate section is also dedicated for quantization related topics due to their importance in this thesis.

The detailed description of the main contributions of this thesis begins in Chapter 3 where the VLBR codec is introduced. In particular, the first two sections describe the parametric representation used in the codec and the main aspects of the segmental coding approach, and provide an overview of the first version of the VLBR codec and its performance. The rest of the sections of Chapter 3 cover a set of additional coding-related solutions that can be used for further enhancing the performance of the VLBR codec.

Chapter 4 discusses the topic of VLBR-based speech synthesis. The first section of the chapter focuses on the integration of the VLBR codec into concatenative unit selection based text-to-speech systems while the second section discusses codec retraining. Section 4.3 introduces a compression-motivated method for very efficient calculation of concatenation costs. While originally developed for VLBR-based synthesis, the method is applicable to practically any unit selection based text-to-speech system.

The next chapter, Chapter 5 deals with the topic of VLBR-based voice conversion. First, the initial version of the VLBR-based voice conversion system is introduced in Section 5.1. Then, the next three sections are dedicated to the introduction of three additional techniques that enhance the performance of the initial VLBR-based voice conversion system.

Finally, conclusions are drawn in Chapter 6. This last chapter both provides a very brief summary of the work introduced in the thesis and indicates the most attractive directions for future research.

Chapter 2

Overview of speech processing

Speech has a central role in human interaction. Thus, it is not surprising that different aspects of speech processing have received a lot of research attention. In addition to the extensive work on digital signal processing based methods, the areas of human speech production, speech perception and spoken language understanding are constantly under rigorous study.

The first section of this chapter summarizes the most important aspects of speech production and speech perception, from the viewpoint of this thesis. In addition, some general issues related to the processing of discrete-time speech signals are discussed. Next, the research area of speech coding is briefly introduced in Section 2.2. The topic is approached from the perspective of low or very low bitrates. The basic tool of linear prediction is introduced as well.

Section 2.3 discusses the topic of quantization, mainly from the perspective of vector quantization that is one of the basic compression tools used in this thesis. Next, Section 2.4 introduces the topic of text-to-speech synthesis. The emphasis is placed on the acoustic synthesis part, and especially on unit selection based concatenative synthesis. Finally, the research area of voice conversion is briefly introduced in Section 2.5.

2.1 Speech production, perception and processing

Children usually learn to produce speech at a very young age, and babies can perceive speech sounds at even younger ages. Even though exact modeling of the related natural mechanisms is not required in speech processing, it is beneficial to discuss, and later on utilize, some of the most relevant aspects. In addition to providing very brief introductions on speech production and perception, this section also covers some basic issues related to discrete-time speech signal processing.

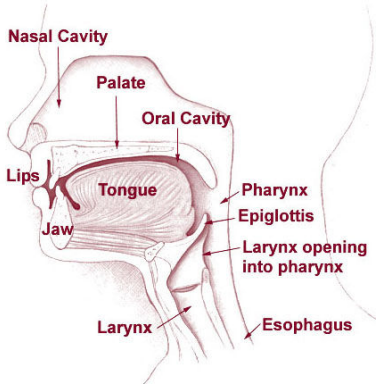

Figure 2.1: Some of the organs involved in speech production. (From [Wik13].)

2.1.1 Speech production

The production of speech is a complex process requiring a coordinated action of a number of muscles [Dut97, Chapter 1]. The process also involves many organs, some of which are illustrated in Figure 2.1. As described in many text books (including, e.g., [Par87], [Qua02], and [Dut97]), the human speech production system can be considered to consist of three main parts, the respiratory organs, the larynx, and the vocal tract. The respiratory organs act as a power supply, i.e., the lungs create an airflow that is forced into the system via the trachea. The air flows into the larynx and through the vocal cords (located in the larynx). The larynx effectively modulates the airflow to provide either a periodic pulse-like or a noisy airflow source into the vocal tract. The vocal tract is usually considered to be comprised of the pharyngeal, oral, and nasal cavities, and it gives the modulated airflow its "color" or timbre by spectrally shaping it. Constrictions along the vocal tract itself can also be used for generating impulsive sound sources. The variation of air pressure at the lips results in an audible speech sound wave.

The perspective of human speech production can be conveniently used for introducing the important concepts of voicing, pitch, and the fundamental frequency F_0 needed in this thesis. Roughly speaking, voiced sounds are produced by forcing air through the opening between the vocal cords, referred to as glottis, with the tension of the vocal cords adjusted so that they vibrate in oscillation. Unvoiced sounds, on the other hand, are produced without the oscillation of the vocal cords. In human speech, vowels can generally be considered voiced sounds and some of the consonants can be considered unvoiced but often voiced and unvoiced speech characteristics co-exist, leading to the need to be able to deal with different degrees of voicing between the fully voiced and fully unvoiced extremes.

The term pitch relates to the fundamental frequency of the vibration in the vocal cords during voiced sounds. Sometimes the terms pitch and the fundamental frequency, also referred to as F_0, are used as synonyms in the literature, sometimes

F_0 refers to the speech production related fundamental frequency and pitch to the perceived frequency [Hua01]. Both terms, pitch and F_0, can be used for referring to the fundamental frequency estimated from the speech signal based on its quasi-periodic nature, even though the estimate may not accurately correspond to neither the actual rate of the vibration in vocal cords nor the perceived pitch. In this thesis, this last approach is taken, and the terms pitch and F_0 are used interchangeably because the term pitch is most commonly used in the field of speech coding and the term F_0 in speech synthesis and voice conversion.

Detailed modeling of the human speech production is extremely difficult due to the many physical components involved and the complex relationships between the different components. However, a simplified model of the human speech production system can be obtained using the source-filter theory [Fan60], in which the sound sources can be thought to form a source signal and the vocal tract acts as a filter. The filter can be separated into different subparts, e.g., the effect of lip radiation can be modeled separately from the vocal tract contribution, but often it is more convenient to consider only a single filter and its transfer function.

The different speech sounds produced by humans can be studied, labeled, and categorized in many different ways. Links can also be created between different levels of description, e.g., between the acoustic and the phonological levels, as discussed, e.g., in [Dut97]. For example, phonemes can be considered basic units of a language that can be combined together to form words. A single phoneme of a language can be regarded to represent a class of speech sounds called phones that are acoustically close enough to each other so that variations within the class do not cause a change of meaning.

More detailed information on speech production related issues can be found, e.g., from [Mac87] as well as from the other references mentioned above.

2.1.2 Speech perception

The task of speech perception involves highly complicated mechanisms and not all details of the human auditory system and the processing of speech sounds are known or fully understood despite active research on the topic. Nevertheless, a lot of information on the related processes has been gathered and some of the findings are particularly useful in digital speech processing.

The human auditory system is typically considered to consist of two main parts, the peripheral part and the part containing the nervous system and certain areas of the brain. From the viewpoint of this thesis, the former can be considered more important. The peripheral part of the auditory system, i.e., the human ear, can be considered to be a preprocessor of sounds [Zwi90], and its structure can be considered to consist of the outer ear, the middle ear and the inner ear. The outer ear captures the sound energy and conveys it to the middle ear via the tympanic membrane, also referred to as the eardrum. The main functions of the middle ear

are amplitude limiting and impedance transformation to ensure efficient transfer of the acoustical energy [Par87]. The most complicated part of the ear is the inner ear comprised of the cochlea and the vestibular organ.

The details of physical structure of the ear are clearly outside the scope of this thesis but the physical properties of the ear cause interesting phenomena, discussed more closely, e.g., in [Moo95a] and [Moo95b], that can be exploited in speech processing. The first of these phenomena is the uneven sensitivity across the audible frequency range (from about 20 Hz to about 16 kHz). The studies on this topic have resulted in the introduction of the concept referred to as the absolute threshold of hearing. The absolute threshold of hearing at a given frequency indicates the minimum sound pressure level that a tone having that frequency needs to have to be audible in an otherwise quiet environment. Extensive measurements have indicated that the threshold varies tremendously over the audible frequency range, and also between different individuals. A useful approximation for the absolute threshold of hearing has been given in [Ter79].

Masking is another interesting phenomenon related to human hearing. Masking is caused by the finite accuracy of the human auditory system and, basically, it means the process by which the perception of one sound is suppressed by another, louder sound. The overall masking effect is mainly determined by the relative levels and frequencies of the maskee and the masker, as well as by their temporal characteristics. The nature of a sound also has a prominent impact on its masking capability. An approximate measure of the amount of masking can be obtained by evaluating a masking threshold that indicates the sound pressure level at which a test sound is just audible in the presence of a masker. A speech signal typically contains multiple maskers and maskees at any given time and there are two distinct types of masking, simultaneous masking and non-simultaneous masking, that can be also present at the same time. Thus, when considering complex signals such as speech, the exact evaluation of the masking threshold is extremely difficult and coarse simplifications must be made.

Yet another interesting phenomenon is a result of the physical structure of the inner ear and entails the concepts of auditory filters and critical bands, as well as the power spectrum model. The power spectrum model approximates the peripheral part of the auditory system using a bank of linear overlapping bandpass filters referred to as auditory filters. Furthermore, the frequency-dependent critical bandwidth is considered to be the noise bandwidth limit at which the detection threshold of a sinusoidal signal located at the center of the noise band does not increase anymore. The experimentally determined critical bands, in turn, can be used for deriving the Bark scale that takes into account the frequency resolution of the auditory system at different frequencies. In practice, the frequency resolution gets less accurate as the frequency increases, i.e., the widths of the critical bands increase with increasing frequency. The same phenomenon has also motivated the development of other perceptual scales such as the mel scale.

In speech coding, the masking effects, as well as other main properties of the human hearing, are typically very roughly taken into account through the use of perceptual weighting schemes and/or postprocessing techniques. It is also possible but not very common to use more sophisticated psychoacoustic models. One such model, the psychoacoustic model proposed by Johnston [Joh88], is used in Section 3.5.

More information on the human auditory system and its properties can be found, e.g., from [Zwi90], [Par87], [Moo95a], and [Moo95b].

2.1.3 Processing of discrete-time speech signals

Speech signal is fundamentally a continuously varying acoustic pressure wave. In this thesis, and more widely in digital signal processing, the term speech signal refers to a discrete-time speech signal, i.e., a measurement of the speech signal sampled at a regular interval. The number of samples per second corresponds to the sampling frequency in Hz. For example, a speech signal containing 8000 samples per second is said to have a sampling rate or a sampling frequency of 8 kHz. Speech signals with an 8-kHz sampling rate are referred to as narrowband speech signals.

Discrete speech signals are typically processed in a frame-based manner. The frame rate depends on the application. For example, many narrowband speech codecs operate using a frame rate of 50 or 100 frames per second and consequently each new frame of speech contains 160 or 80 samples of new speech signal data, respectively. Signal analysis is typically performed on windowed speech. Any signal processing windows, such as rectangular windows, Hann/Hanning windows or Blackman windows, can be used. The length of the window does not have to match the length of the frame. In fact, it is common to use longer than frame length windows but to use a step size equal to the frame length.

In addition to time-domain processing, speech signals are often analyzed and processed in frequency domain. The conversion from time domain to frequency domain is typically performed using Fourier transformation. In addition to spectral representations, cepstral representations and in particular *mel-frequency cepstral coefficients* (MFCCs) [Dav80] are sometimes used as well, also in this thesis. The perceptually motivated MFCC parameters are commonly obtained by mapping the spectral power spectrum onto the mel scale using triangular overlapping windows, taking logarithms of the resulting filterbank energies, and by taking the *discrete cosine transform* (DCT) of the mel-scale logarithmic energies.

More information can be found from speech processing related literature. For example, an excellent introduction to discrete-time processing of speech signals is provided by Quatieri in his book [Qua02].

2.2 Speech coding

All speech coding systems consist of an encoder and a decoder. The encoder converts a speech signal into a bitstream that is conveyed to the decoder via a digital channel. The decoder then reconstructs the speech signal based on the bitstream. The digital channel between the encoder and the decoder can be a communication channel or a storage medium.

The simple and straightforward *Pulse Code Modulation* (PCM) [Oli48] is usually considered to be the first method developed for digital speech coding. In PCM, the encoder only performs the basic operations of sampling of the input signal, quantization of the samples, and coding of the quantized sample values using their binary representations. Similarly, the decoder merely restores the samples by decoding the received binary information and reconstructs the signal based on the restored sequence of sample values.

The sampling and the reconstruction phases used in PCM, and in all digital speech processing, allow perfect reconstruction of the original signal assuming that it does not contain information above the Nyquist frequency, i.e., above half of the sampling frequency. For this kind of band-limited input signals, the only source of possible quality loss is the quantization process, assuming that the digital channel is error-free. The quantization process also determines the bitrate of the PCM codec, together with the sampling frequency. For example, with the quantization accuracy of 16 bits per sample, a narrowband speech signal sampled at 8 kHz would require a PCM bitrate of 128 kbps.

Reducing the bitrate by simply adjusting the quantization accuracy used in PCM is possible but the resulting quantization noise quickly deteriorates the output speech quality. This is where the different speech coding strategies developed since the introduction of PCM step in. For example, the GSM (*Global System for Mobile communications*) full rate speech codec [ETS94] operates at the bitrate of 13 kbps and the standardized *adaptive multi-rate* (AMR) coder [ETS00][Eku99] operates at eight bitrates from 4.75 to 12.2 kbps.

This thesis deals with speech coding at very low bitrates. There is no exact definition for the term very low bitrate but in practice the bitrates considered in this thesis are below 3 kbps, and often in the neighborhood of 1 kbps or even below that. The remaining parts of this section introduce the important concept of linear prediction (Section 2.2.1) and give an overview of the task of coding speech at low or very low bitrates (Section 2.2.2). More detailed and extensive general background information on speech coding can be found, e.g., in [Spa94] and [Kon04].

2.2.1 Linear prediction and line spectral frequencies

The vast majority of modern speech coders are based the *linear prediction* (LP) technique [Mak72] [Mak75] that is also one of the basic tools in all speech pro-

cessing. This source-filter model can be used for separating a discrete speech signal $s(t)$ into an excitation signal and into linear prediction coefficients that roughly model the vocal tract contribution. More precisely, the excitation signal, $r(t)$, also referred to as the residual signal, can be obtained through LP analysis filtering,

$$r(t) = s(t) - \sum_{j=1}^{g} a_j s(t-j), \tag{2.1}$$

where $\{a_j\}$ are the linear prediction coefficients and g is the order of the LP analysis filter that has the form

$$A(z) = 1 - \sum_{j=1}^{g} a_j z^{-j}. \tag{2.2}$$

Similarly, $1/A(z)$ is the corresponding LP synthesis filter that can be used for filtering the excitation signal to generate speech.

In linear prediction, as can be seen from the above equations, each predicted signal value is calculated as a weighted sum of a predetermined number of recent signal values in accordance with the auto-regressive predictor model [Par87]. The linear prediction coefficients $\{a_j\}$ can be estimated using either the autocorrelation method or the covariance method [Mak72]. Slightly different variants of the two methods also exist, as mentioned already in [Mak72]. The autocorrelation based methods possess the important property of ensuring that the resulting linear prediction filters are stable. With all variants, the aim in coefficient estimation is to minimize the short-term correlation, in practice by minimizing the mean energy of the resulting excitation signal $r(t)$.

Because the properties of speech signals vary in time, a new set of linear prediction coefficients is estimated at certain time intervals. Typically, the speech signal is processed using 5-25 ms frames and the estimation of LP coefficients is performed once per frame. A typical analysis window length is roughly 20-30 ms [Kon04] and is usually larger than the frame length. Windowing can be done using, e.g., a Hamming window.

To obtain smooth changes in the filter, the LP coefficient set can be updated more often than once per frame. The coefficient values between the estimates can be calculated using interpolation. However, direct interpolation of the LP coefficients is rather troublesome since the stability of the resulting filters cannot generally be ensured. To overcome this problem the LP coefficients are usually converted to the *line spectral frequency* (LSF) representation [Ita75] before the interpolation.

The conversion to the line spectral frequencies can be established by first calculating the roots of the polynomials

$$\begin{aligned} P(z) &= A(z) + z^{-(g+1)} A(z^{-1}) \\ Q(z) &= A(z) - z^{-(g+1)} A(z^{-1}) \end{aligned}, \tag{2.3}$$

using, e.g., the discrete cosine transform as in [Soo93] or Chebyshev polynomials as proposed in [Kab86]. Now the angular positions $\{\omega_j\}$ of the complex roots of the polynomials, that lie on the unit circle between 0 and π, form the LSF representation. Moreover, the line spectral frequencies are defined to be in ascending order,

$$0 < \omega_1 < \omega_2 < ... < \omega_g < \pi \tag{2.4}$$

that ensures the stability of the filters after the interpolation. The filter $A(z)$ can be calculated using

$$A(z) = \frac{P(z) + Q(z)}{2} \tag{2.5}$$

and thus the conversion to the LSF representation is fully reversible.

From the viewpoint of speech coding, one of the advantages of the linear prediction comes from the fact that it lowers the energy and the average entropy of the signal to be coded by removing redundant and predictable information from the original signal. Because the predicted part of the speech signal can also be efficiently compressed by coding the linear prediction coefficients using, e.g., the LSF representation, linear prediction facilitates efficient compression of the signal.

2.2.2 Speech coding at low bitrates

Speech coders can generally be classified as waveform coders and parametric coders [Kle95b][Spa94]. The waveform coders attempt to model and transmit the shape of the speech waveform as accurately as possible. This approach attains good or even excellent speech quality provided that the bitrate is high enough. However, at lower bitrates, the quality of the reconstructed speech deteriorates quickly and thus for example the wide and popular family of speech codecs based on the idea of *code excited linear prediction* (CELP) [Sch85] are not generally applicable at very low bitrates. On the other hand, the parametric coders, also referred to as voice coders, transmitting a set of parameters that describe the perceptually most important features of the speech signal have the potential to produce intelligible and relatively good-quality speech at very low bitrates. Thus, the focus in this thesis is on parametric speech coding.

In low bitrate speech coding, the most evident design goal is to achieve a low bitrate. Alternatively, the goal could be to achieve a given speech quality level with the lowest possible bitrate. In addition to the goals related to the bitrate and speech quality, there are typically several other design goals or constraints that have to be taken into account. Traditionally, the design of speech coders has been heavily affected by the design constraints related to conversational speech. The most common constraints relate to encoding delay and complexity, sensitivity to channel errors and background noise conditions, frame size, bitrate and bit allocation, decoding complexity, and memory requirements. In the storage-related

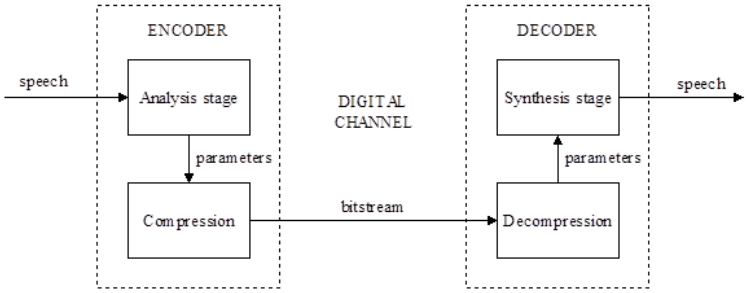

Figure 2.2: Block diagram demonstrating parametric encoding and decoding of speech. (From [Nur01a].)

scenarios studied in this thesis, some of these traditional design constraints can be relaxed which in turn can lead to enhancements in the compression efficiency, as will be discussed in Chapter 3.

Figure 2.2 depicts the basic operation of a parametric speech codec. Similarly as in the case of the basic PCM method discussed in the beginning of this section, the main parts of the speech coding system are the encoder and the decoder. In parametric speech coding, the encoder can be further split into two parts, a speech signal analysis stage that estimates a set of parameters for each frame of speech and a compression stage that compresses the set of parameters into a compact bitstream. Similarly, the decoder can be considered to consist of a decompression stage that restores the parametric representation based on the bitstream and a synthesis stage that produces the reconstructed speech signal. Typical errors caused by the encoding and decoding processes in low bitrate speech codec include both modeling errors caused by imperfect analysis/synthesis and quantization errors caused by the compression and decompression.

Various different approaches have been proposed for parametric speech coding. [Kle95b] categorizes the different approaches into three classes: linear prediction based coders, sinusoidal coders and waveform interpolation coders. Without further explanations, this division could be a bit misleading since actually almost all speech coders designed for low bitrates are based on linear prediction, including many codecs based on the sinusoidal and waveform interpolation approaches. It is therefore important to note that the first class of linear prediction based coders does not include all the speech coders that utilize linear prediction. Rather, this class includes linear prediction based codecs that utilize very simplistic excitation models. A typical example of a codec belonging to this class is the classical *linear predictive coding* (LPC) based LPC-10 vocoder [Tre82] that regards all segments of speech as either fully voiced or fully unvoiced. The voiced excitation is modeled as periodic pulses while the unvoiced excitation is represented using random noise. Even though this approach is quite well in line with a simplified view of the human speech production, the model has been found

inadequate for producing high quality speech, regardless of the accuracy of quantization.

The second and the third class of parametric speech coders, the sinusoidal coders and the waveform interpolation coders, offer more sophisticated approaches for modeling the excitation signal and can be considered more relevant from the viewpoint of this thesis. Both approaches can also be used for direct modeling of speech signals but it is more common to model the LP residual signals. The approaches also share another important property: with specific design choices, both approaches can obtain perfect reconstruction, i.e., the modeling errors caused by the analysis and synthesis stages in Figure 2.2 can be completely avoided. Such designs are discussed, e.g., in the case of the sinusoidal modeling in [Fer02] and in the case of waveform interpolation in [Yan98] and [Ruo00]. At very low bitrates, however, obtaining perfect reconstruction is not mandatory and typically better results can be achieved using more approximative versions of the models because those tend to result in parameter sets that are easier to compress efficiently.

Most sinusoidal speech coders are based on the model presented in [McA86]. The main idea of this model is to represent the signal using sinusoidal components of arbitrary amplitudes, frequencies and phases. Each sinewave is represented by a time-varying envelope and a phase equal to the integral of a time-varying frequency track [Qua02, Chapter 9]. During voiced speech, the frequency tracks of the sinewaves are roughly harmonically related. Even though the model is more intuitive for voiced speech, noise-like and transient speech sounds can also be approximated by a sum of sinewaves but now, in contrast to the case of voiced speech, the frequencies have arbitrary values and their tracks come and go randomly in time over shorter durations (see, e.g., [Qua02, Chapter 9]). More information on the sinusoidal model is given in Section 3.1.

In *waveform interpolation* (WI) coding [Kle95a], a *characteristic waveform* (CW) is extracted at regular intervals. These pitch-cycle waveforms are placed along an axis perpendicular to the time axis to obtain a two-dimensional signal that represents the evolution of the characteristic waveform in time. After alignment, the two-dimensional representation can be further separated via filtering into a low-pass *slowly evolving waveform* (SEW) component that corresponds to the periodic component of speech and a high-pass *rapidly evolving waveform* (REW) component that represents the noise-like component. Even though both components are typically present (non-zero) most of the time, the SEW component is dominant during voiced speech while unvoiced speech is mainly modeled by the REW component. At the decoder, the characteristic waveforms can be recovered by summing up the SEW and REW components. The successive CWs can be thought to form a surface that can be upsampled to the sampling rate of one CW per output speech sample using interpolation between the CWs. The speech signal can be reconstructed by sampling the waveform surface along a phase track. The phase at each time instant is equal to the integral of the fundamental frequency.

2.3 Quantization

The term quantization was briefly touched upon in Section 2.2 as a part of the high-level description of the PCM coding approach. In that context, the term simply refers to the representation of the exact sample values with a limited accuracy in a discrete manner. With adequate scaling, this can be thought to correspond to rounding off that is generally considered to be the simplest and oldest example of quantization [Gra98].

In the examples of classical PCM coding and rounding off, the quantizers are uniform, i.e., the possible quantization output values are equally spaced. In general this is not the case, and it is also very common to quantize multiple values simultaneously, i.e., to quantize vectors instead of scalars. In addition, it is possible to use special quantizer structures and/or prediction. This section provides a brief introduction to these topics, in a manner that meaningfully supports the description of the main contributions of this thesis. More complete introductions to the topic of quantization can be found, for example, in [Ger92] and [Gra98].

2.3.1 Vector quantization

Vector quantization (VQ) is one of the most efficient and powerful tools that can be used in data compression. A fundamental result of Shannon's rate-distortion theory [Sha59] shows that better performance can always be achieved by coding vectors instead of scalars, even for uncorrelated or independent data, as discussed in [Gra84]. The basic idea in vector quantization is to compress the input vectors by representing them using a predefined set of symbols. The symbols can be decompressed and converted into vectors using a reproduction codebook. Optimally, the process is performed in a way that minimizes the resulting distortion.

A k-dimensional vector quantizer consists of two mappings [Gra84]. The first mapping is an encoder γ that assigns to each input vector $\mathbf{x} = [x_1, x_2, ..., x_k]^\top$ a channel symbol $\gamma(\mathbf{x})$ in a channel symbol set ζ. The symbol is then conveyed to a decoder β that performs the second mapping by assigning to each channel symbol h in ζ a code vector in a reproduction set $\mathbf{C}$. This finite set is usually referred to as the codebook of the quantizer and is defined as

$$\mathbf{C} = \{\mathbf{c}_h | h \in \zeta\}, \tag{2.6}$$

where $\mathbf{c}_h = \beta(h)$. Consequently, the quantized vector $\hat{\mathbf{x}}$ can be obtained through the two mappings as

$$\hat{\mathbf{x}} = \beta(\gamma(\mathbf{x})) = \mathbf{c}_{\gamma(\mathbf{x})}. \tag{2.7}$$

The accuracy achievable using a quantizer is dependent on the size of the reproduction codebook. The resolution, or rate, of a k-dimensional VQ is $\frac{\log_2 N}{k}$, where N denotes the number of elements in the channel symbol set [Ger92]. The rate of

a quantizer measures the number of bits needed for representing one vector component [Ger92]. Another, often even more popular way for describing the rate of a quantizer is to define the number of bits needed for representing the channel symbol for the whole vector. E.g. if the rate of the quantizer is 3 and $k = 2$, the quantizer can also be referred to as a 2-dimensional 6-bit vector quantizer.

Optimality conditions and distortion measures

A vector quantizer is considered optimal if two conditions are fulfilled. First, the encoder must always select a mapping that minimizes the resulting distortion

$$\gamma(\mathbf{x}) = \arg\min_{h \in \zeta} d\left(\mathbf{x}, \beta(h)\right), \tag{2.8}$$

where $d(\cdot)$ is a distortion measure such as, for example, the squared error

$$d(\mathbf{x}, \mathbf{c}) = (\mathbf{x} - \mathbf{c})^\top (\mathbf{x} - \mathbf{c}). \tag{2.9}$$

In low bitrate speech coding, it is typical to complement the distortion measure in Equation (2.9) with perceptually motivated weighting. The resulting weighted squared error can be expressed as

$$d(\mathbf{x}, \mathbf{c}) = (\mathbf{x} - \mathbf{c})^\top \mathbf{W} (\mathbf{x} - \mathbf{c}), \tag{2.10}$$

where the weighting matrix $\mathbf{W}$ is typically diagonal. The second condition for the optimality of a vector quantizer states that the decoder must assign to each channel symbol h the generalized centroid of all vectors mapped into h,

$$\beta(h) = \text{cent}(h) = \arg\min_{\hat{\mathbf{x}}} E\left(d(\mathbf{x}, \hat{\mathbf{x}} | \gamma(\mathbf{x}) = h)\right). \tag{2.11}$$

In other words, the average distortion caused by the two mappings in the quantization process should be minimized [Gra84].

From the optimality condition in Equation 2.8, it follows that full search should be employed in the encoder, meaning that the distortion is measured for every code vector in the codebook and the channel symbol corresponding to the code vector leading to minimum distortion is selected. The condition in Equation 2.11 implies that the reproduction codebook $\mathbf{C}$ must be optimal. The two optimality conditions together imply that an optimal vector quantizer can be fully described by the distortion measure d and the reproduction codebook $\mathbf{C}$ (and a mapping rule for cases where more than one mapping leads to equal minimum distortion).

Codebook training

Since the reproduction codebook, along with the distortion measure, determines the performance of a vector quantizer, it is essential that the codebook is well designed. As stated in Equation (2.11), a reproduction codebook is considered optimal if it consists of the distinct centroids of the source vectors mapped into each channel symbol. However, since such codebooks can be constructed in many ways, it is obvious that the optimality conditions alone only ensure that the codebook is locally optimal. A reproduction codebook that minimizes the overall distortion of the quantizer among every possible codebook is considered a globally optimal codebook.

The objective in codebook design is to find a space partitioning that minimizes the expected overall distortion between the input and the reproduction. The overall distortion is usually approximated using the long-term sample average [Gra84], i.e., the empirical average distortion for all the vectors in the training set. Since the source distribution is estimated using a training sequence consisting of a finite number of training vectors, one of the most fundamental problems in codebook training is the selection of the training sequence. There are no strict rules or solutions to this problem. However, the training data should always consist of representative pieces of the typical input data. Moreover, it is recommended that the training set should consist of at least 50 vectors per available channel symbol [Mak85].

Once the training data is selected, the actual codebook can be constructed in several ways. The most commonly used basic approach is to employ the *generalized Lloyd algorithm* (GLA) [Lin80], also referred to as the Linde-Buzo-Gray algorithm (the algorithm is also essentially similar to the well-known K-means algorithm). The main idea is to begin with an initial codebook, and then to alternately encode the training sequence using the minimum distortion rule in Equation (2.8) and to replace the old reproduction codebook by the centroids of the training vectors mapped into each channel symbol according to Equation (2.11). The iteration is carried on until the overall distortion or the change in the overall distortion is considered low enough, or a predetermined maximum number of iterations has been reached.

The generalized Lloyd algorithm can be shown to converge to a local optimum [Gra84]. However, an inherent problem with the GLA approach is that it often gets greedily attracted to a nearby local minimum instead of finding the global minimum. Finding a globally optimal codebook is possible but only if the process is started with an initial codebook that converges to the global minimum. Thus, the selection of the initial codebook can be considered the most crucial step in the GLA method.

Many techniques have been proposed for constructing the initial codebook (several alternatives were already introduced in [Gra84]), but the method for generating an initial codebook always yielding a globally optimal codebook is yet to

be found. The most simple technique that provides reasonably good results is to use GLA with a set of different random initial codebooks and to select the codebook that results in the lowest distortion. More refined approaches have also been proposed in the literature. For example, deterministic annealing [Ros93] has been reported to achieve promising results [Ros98a], and particle swarm optimization has also been found a valid approach [Sun10]. Despite the often slightly improved quality, the related additional complexity makes many of these improved methods less appealing. In practice, satisfactory performance can be usually achieved by using the simple technique of repeated random initializations.

2.3.2 Multistage vector quantization

Vector quantization can be considered the best possible memoryless compression tool in the sense that no other memoryless coding scheme that maps a signal vector into one of N binary words can outperform vector quantization as there always exists a vector quantizer with codebook size N that provides at least the same accuracy [Ger92]. However, in many application scenarios, the related memory consumption and the computational complexity of the codebook search can often make direct use of the basic vector quantization approach impractical. Consequently, many alternative quantizer structures and search strategies have been proposed in the literature. Examples of such alternative approaches include split vector quantization, gain-shape quantization, binary search codebooks, and lattice vector quantization (see, e.g., [Ger92], [Gra84], and [Kon04] for more information on the different alternative quantizer structures and search strategies). From the viewpoint of this thesis, *multistage vector quantization* (MSVQ) [Jua82] is of particular interest due to the excellent tradeoff between the performance and the resource needs in terms of the computational load and memory usage that it offers. MSVQ can also be considered an excellent choice because it can regarded as a generalization that also represents many of the other alternatives. For example, split vector quantizers and gain-shape quantizers can be realized as special cases of the multistage vector quantization approach.

A multistage VQ [Jua82] quantizes the vectors in two or more additive stages. The objective is to find a vector combination, in other words a sum of the selected vectors at different stages, that minimizes the resulting distortion. The quantized vector can be defined as

$$\hat{\mathbf{x}} = \sum_{j=1}^{K} \mathbf{c}_{l_j}^{(j)}, \tag{2.12}$$

where $\mathbf{c}_m^{(j)}$ denotes the mth reproduction vector from the jth stage , K is the total number of stages, and l_j is the index of the vector selected at the jth stage.

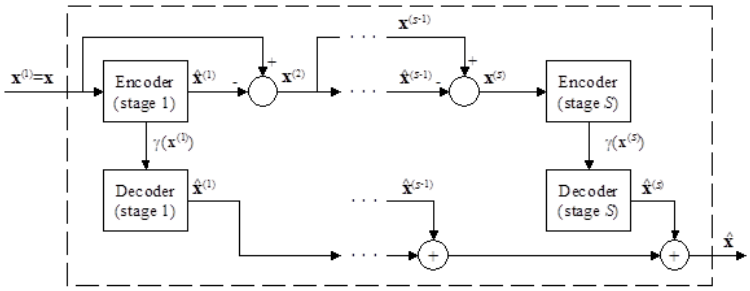

Figure 2.3: Block diagram of a multistage vector quantizer using sequential search. (From [Nur01a].)

Codebook search in MSVQ

The use of multistage codebooks can drastically reduce the memory consumption but the computational complexity of a multistage vector quantizer depends on the applied search algorithm. If full search is used, i.e., the distortion measure is calculated for every possible vector combination, the computational complexity is higher than with the normal unconstrained vector quantizer, due to the extra additions needed to sum up the codevectors from the different stages. However, an efficient search algorithm can significantly reduce the complexity.

The simplest search algorithm for multistage quantization is the sequential search. The process begins with a full search quantization using only the codebook of the first stage. Then, the quantization error is calculated as the error between the original vector and the quantized vector. After that, the error vector is quantized using only the second stage codebook and the resulting error vector is computed. This is carried on until the quantization for every stage has been performed. Finally, the quantized vector is the sum of the quantized vectors at different stages. This procedure is illustrated in Figure 2.3.

The sequential search algorithm is simple, but the resulting quantization performance is rather poor. A better choice is to use the M-L tree search depicted in Figure 2.4, in which the M best vector combinations are searched at each stage. That is, at the first stage, the M vectors that result in the lowest distortion are selected. Then at the second stage each reproduction vector is combined with the M vectors selected at the first stage and again the M paths that achieve the lowest overall distortion are selected. This is carried out for all stages. Finally, the full path with the lowest distortion determines the channel symbols for each stage.

It is easy to see that setting $M = 1$ corresponds to a sequential search. Naturally, it is beneficial to use a larger M in the search algorithm since usually larger values of M lead to smaller overall distortion. However, it has been found that the M-L tree search achieves performance close to that of the full search with a relatively small M [LeB93].

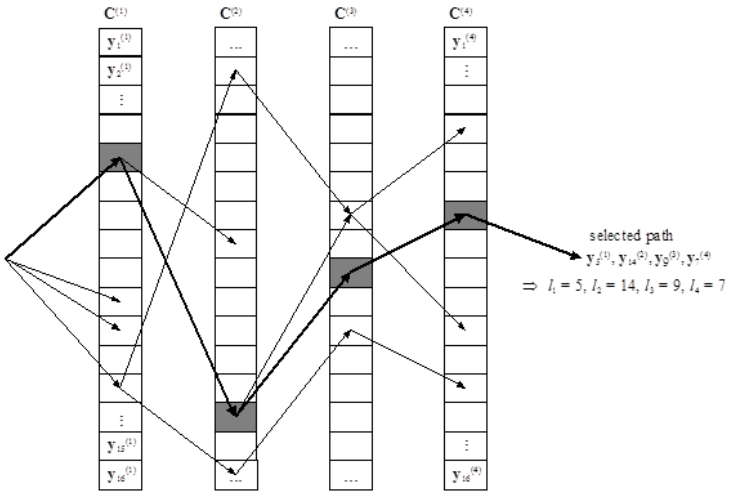

Figure 2.4: Example of M-L tree search procedure with $M = 4$ in a 4-stage VQ. (From [Nur01a].)

Codebook training for MSVQ

Training the codebooks is more complicated for a multistage VQ than for a conventional VQ because the final reproduction vector depends on the codebooks of all the stages. The simplest applicable training method is to train the codebooks sequentially. In this approach, the codebook for the first stage is computed in a traditional manner using, e.g., the generalized Lloyd algorithm. Next, the training data is quantized with this one-stage VQ and the quantization error vectors are calculated. Then, the codebook for the second stage is trained using these error vectors as the training data. This is repeated for all the stages, with each new codebook calculated using the error between the original vector and the reconstruction vector including all the previous stages as the training data. The training procedure is terminated when the codebooks for all the stages have been computed.

The basic sequential training method is simple, but unfortunately the resulting codebooks are only sub-optimal with respect to the overall performance [Cha92]. The algorithm fails to efficiently exploit the inter-stage dependencies in the codebook optimization. The performance of the sequential training can be improved by making two modifications to the algorithm. Firstly, the error vectors can be calculated as the error between the original vector and the multistage reproduction vector including all the stages except the current stage (that the codebook is currently trained for). Secondly, the algorithm is repeated until the relative change in the distortion is low enough or the total number of repetitions has reached a certain predetermined limit. The resulting algorithm is referred to as the joint design of the stage codebooks [Cha92].

The joint codebook design algorithm offers a performance improvement over the traditional sequential codebook training. However, the improvement is rather modest [Cha92] and the codebook optimization is still performed for one codebook at a time. Furthermore, the convergence of the algorithm is quite slow. The simultaneous joint design algorithm proposed in [LeB93] offers yet another step towards better performance and faster convergence. The basic idea in this method is to jointly optimize the codebooks after each pass over the training sequence. The resulting multistage VQ simultaneous joint design algorithm [LeB93] will be used extensively in this thesis.

The simultaneous joint design algorithm is usually initialized with sequentially designed random codebooks. In theory, it is assumed that the quantization is performed using full search. However, it has been experimentally found that good performance can be achieved by employing the M-L tree search with a moderate value of M [LeB93]. This is partially enabled by the fact the codebooks are reordered at each training iteration in such a manner that the energy at any given stage after subtracting the codebook mean is less than the corresponding energies at all the previous stages. See [LeB93] for more detailed information on the algorithm.

2.3.3 Predictive vector quantization

The vector quantization methods described in this section can achieve very good coding quality for a given bitrate. However, the performance can still be improved by incorporating memory into the vector quantizer, provided that there is some correlation between successive vectors. The memory can be used to store information about one or more previously quantized vectors. Based on this previously obtained information, a prediction of the current vector to be quantized is calculated. Then, instead of quantizing the vector itself, only the error between the original vector and the prediction is quantized. This approach is referred to as *predictive vector quantization* (PVQ) [Cup85]. (It should be noted that while PVQ best fits the needs in this thesis, there are also other quantization approaches that utilize memory. Finite-state vector quantization [Fos85] is one example of such an approach. See, e.g., [Ger92] for more information.)

The basic idea in predictive vector quantization is close to that in the linear prediction approach introduced in Section 2.2.1. There are two major differences between these methods. Firstly, the PVQ method operates on vectors instead of scalars. Secondly, the "prediction coefficients" or the predictor matrices are often constant in predictive vector quantization. Thus, the prediction cannot usually adapt to changes in the input data like in the case of linear prediction described earlier.

Mathematically, in predictive vector quantization the prediction $\tilde{\mathbf{x}}_n$ of the input vector $\mathbf{x}_n$ is calculated using information about a finite number of previously

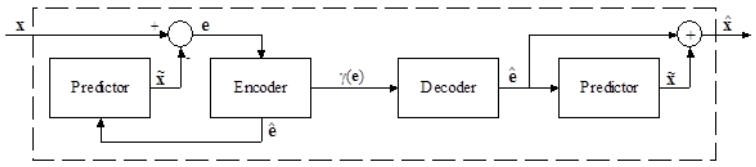

Figure 2.5: Predictive vector quantizer. (From [Nur01a].)

quantized vectors. Then, the prediction error

$$\mathbf{e}_n = \mathbf{x}_n - \tilde{\mathbf{x}}_n \tag{2.13}$$

is quantized instead of the original input vector. Finally, the output of the PVQ is computed by adding together the prediction and the quantized prediction error,

$$\hat{\mathbf{x}}_n = \tilde{\mathbf{x}}_n + \hat{\mathbf{e}}_n. \tag{2.14}$$

Consequently, a predictive vector quantizer can be seen simply as a normal vector quantizer operating with the prediction errors. The resulting quantizer structure is depicted in Figure 2.5. There are alternative methods for obtaining the prediction errors, or predictions, and in principle there are no limitations on how to compute them, as long as the same predictions can be made both at the encoder side and at the decoder side. Usually, the predictions are calculated in a linear manner using either the *auto-regressive* (AR) or the *moving average* (MA) approach.

Common predictor types

In the auto-regressive method, the prediction is established using the quantized values of previous input vectors. Each involved vector is multiplied with the corresponding predictor and the prediction is computed as a sum,

$$\tilde{\mathbf{x}}_n = \sum_{j=1}^{m_A} \mathbf{A}_j \hat{\mathbf{x}}_{n-j}. \tag{2.15}$$

By plugging Equation (2.14) into Equation (2.15), the prediction can also be expressed in terms of earlier predictions and quantized prediction errors as

$$\tilde{\mathbf{x}}_n = \sum_{j=1}^{m_A} \mathbf{A}_j \left(\hat{\mathbf{e}}_{n-j} + \tilde{\mathbf{x}}_{n-j}\right). \tag{2.16}$$

In both Equation (2.15) and Equation (2.16), m_A is the predictor order and $\mathbf{A}_j$ is the jth AR predictor matrix.

The prediction in the moving average approach is based only on the preceding quantized prediction errors. The prediction vector can be conveniently expressed as

$$\tilde{\mathbf{x}}_n = \sum_{q=1}^{m_B} \mathbf{B}_q \hat{\mathbf{e}}_{n-q}, \tag{2.17}$$

where m_B is the MA predictor order and $\mathbf{B}_q$ is the qth predictor matrix. It is also possible to combine the moving average method with the auto-regressive approach as

$$\tilde{\mathbf{x}}_n = \sum_{j=1}^{m_A} \mathbf{A}_j \hat{\mathbf{x}}_{n-j} + \sum_{q=1}^{m_B} \mathbf{B}_q \hat{\mathbf{e}}_{n-q} \tag{2.18}$$

to form an ARMA predictor.

The difference between the two introduced methods is that the MA approach uses only the earlier quantized prediction errors while AR takes advantage of the previous predictions as well. It has been reported that potentially lower distortion with a lower predictor order can be achieved by employing the auto-regressive prediction [Sko97]. However, this advantage is only true for an error-free environment because there is no mechanism to limit the propagation of the effect of the occurring bit errors. The error propagation in moving average prediction is limited by the predictor degree and thus the performance for noisy channels is better [Ohm93]. This makes MA the most popular predictor type in speech coding applications despite its slightly inferior performance in error-free situations.

Training of predictive quantizers

The coding performance in predictive vector quantization depends on both the predictor and the reproduction codebook. Optimally, the codebook of a predictive vector quantizer and the predictor matrices should be jointly optimized to obtain the best possible quantization quality. However, this approach is rarely applied in practice since it leads to somewhat complicated and computationally burdensome optimization techniques such as the stochastic gradient and coordinate descent methods (see for example [Cha86] and [Zeg91]). Furthermore, it has been reported that joint optimization is not usually worth the effort and much simpler techniques can achieve nearly identical performance [Ger92][Chapter 13]. In particular, it has been found that good overall performance can be achieved if the codebook is optimized for the predictor even when the predictor is not optimized for the codebook [Cha86].

In [Cup85], two popular basic techniques were introduced for the training of predictive quantizers. The first technique, referred to as the open-loop approach, is the simplest. In this approach, the predictor is designed first and then a training set of prediction error vectors is obtained directly using the predictor and the original, unquantized source vectors. Finally, the codebook is trained for this training set using conventional training methods.

The second basic approach proposed in [Cup85] is the closed-loop approach. Both the original version presented in [Cup85] and the later version described in [Ger92] utilize a closed-loop system for generating the prediction errors in an iterative fashion. At each iteration, a new training set is generated by computing the prediction error vectors using the quantizer of the previous iteration, alternating between the computation of $\hat{\mathbf{x}}_n$ and $\mathbf{e}_n$ for all the vectors in the training sequence. The initial quantizer is typically generated using the open-loop method. Even though both of these simple basic techniques fulfill the criterion of [Cha86] by optimizing the codebook for the predictor, better performance can be obtained by modifying the training algorithm.

The *asymptotic closed-loop* (ACL) design algorithm, proposed in [Ros98b] and further studied and developed, for example, in [Kha01b], is one of the most appealing methods developed to tackle the problems related to the basic approaches. The ACL technique has been shown to provide a stable design process and to produce high-quality predictive quantizers. In the asymptotic closed-loop design algorithm, when implemented as described in [Kha01b], the predictive quantizers are trained using alternate optimizations of the predictor and the codebook. The stability of the training procedure is improved using a simple trick: the predictions are always based on a fixed set of vectors obtained during the previous iteration. Thus, the training is effectively carried out in an open-loop fashion and the instability problems associated with the closed-loop approach can be avoided. However, the optimization is ultimately performed for closed-loop operation [Kha01b].

Due to the good performance and simplicity, the predictive quantizers used in this thesis work are trained using the asymptotic closed-loop design technique. A practical example demonstrating effective use of the ideas behind the ACL design method is provided in Section 3.3.

2.4 Text-to-speech synthesis

The term *text-to-speech* (TTS) synthesis refers to technology that produces artificially-generated speech based on an input text, in essence "reading aloud" the input text. TTS synthesis can be applied, e.g., whenever a computerized application needs to deliver information to a human user via voice-based interaction. Typical examples of potential applications include applications designed for reading e-mails or SMS messages, as well as entertainment applications such as games. The possibility to use TTS synthesis is especially useful for visually-impaired people but people with normal visual acuity can benefit from speech synthesis, too. For example, on embedded devices speech synthesis can provide a meaningful additional means for delivering information to the user, e.g., while driving a car. TTS technology can be regarded as one of the key enablers that make the current and the future voice-based interaction solutions feasible.

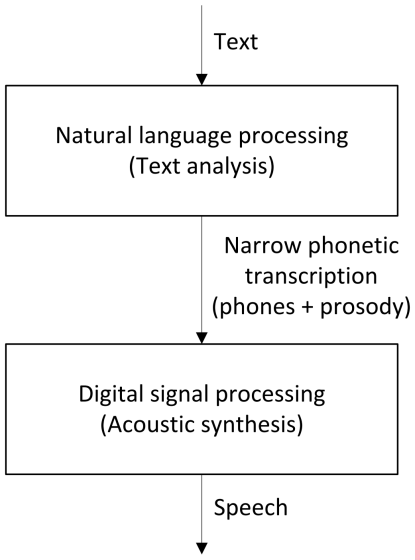

Figure 2.6: Functional diagram of a TTS system.

2.4.1 Overview of the text-to-speech process

While most humans learn how to read text aloud at a young age, even the most sophisticated modern text-to-speech (TTS) systems sometimes fail at the same task. The overall process of converting text into artificially-generated speech is a complex task, and the development of text-to-speech systems requires multi-disciplinary insights from numerous areas of research, and the expertise of both engineers and linguists is needed extensively. Detailed introduction of the different aspects of TTS and the processes needed in text-to-speech systems is clearly outside the scope of this thesis but at a very high level, the text-to-speech synthesis process can be thought to consist of two main parts, a first part dealing with natural language processing and a second part dealing with digital signal processing. This division depicted in Figure 2.6, and also used by Dutoit in his excellent introduction to text-to-speech synthesis [Dut97], offers a meaningful level of abstraction from the viewpoint of this thesis.

The first part of the TTS system in Figure 2.6 deals with the analysis of the input text. A wide range of natural language processing techniques and often a lot of language-specific knowledge is required for turning the text to be read into its phonetic transcription and the related prosodic information (describing the desired intonation and rhythm). The exact form of this narrow phonetic transcription is system-specific. From the viewpoint of this thesis, it is sufficient to know that the front-end of the TTS system, i.e., the natural language processing part, produces some kind of narrow phonetic transcription that consists of a list of phone or speech sound names (typically phoneme names) that together form the desired

output speech, along with side information that describes the desired prosody of the output speech.

Chapter 4 of this thesis deals with the second part of the TTS system of Figure 2.6. This second part converts the narrow phonetic transcription into audible speech. Many digital signal processing techniques have been proposed for this acoustic synthesis part in the literature. The following subsections discuss the most prominent approaches and the particular aspects that are important for the work described in this thesis.

2.4.2 Acoustic synthesis

In [Dut97], the different acoustic synthesis strategies are categorized into two main classes, synthesis by rule and synthesis by concatenation. Another general view, that better serves the purpose of this brief introduction, is to consider four main categories of techniques that can be used for handling the acoustic synthesis, i.e., the conversion of the narrow phonetic transcription into synthetic speech. These main synthesis technique categories are formant synthesis, articulatory synthesis, concatenative synthesis, and statistical synthesis.

Formant synthesizers are generally rule-based systems that represent speech as the dynamic evolution of different parameters mostly related to formants, antiformants, and glottal waveforms, controlled by a series of rules. The Klatt synthesizer [Kla80] is the most well-known example of a formant synthesizer, and its wide-spread use can be partly explained by the fact that phoneticians and phonologists favor rule-based synthesizers because they constitute a cognitive generative approach of the phonation mechanisms [Dut97]. Articulatory synthesis takes even one step further into this direction by trying to directly model the human speech production system, in the hope that a more realistic articulatory model might lead to simpler, more elegant rules [Kla87].

From the viewpoint of this thesis, concatenative synthesis is the most relevant of the synthesis technique categories since Chapter 4 deals with VLBR-based concatenative unit selection based synthesis. Consequently, this approach based on the concatenation of pre-recorded units of speech is introduced in a more detailed manner in the following subsection.

The remaining fourth main category of speech synthesis techniques is statistical speech synthesis in which the synthesis is typically based on the use of context-dependent *hidden Markov models* (HMMs) and many of the core techniques originate from HMM-based speech recognition summarized in [Rab89]. In HMM-based speech synthesis [Tok02], a parametric speech model is employed, and all the speech parameters are typically modeled simultaneously within a single multi-stream HMM. The model parameters, including the means and the covariances of the state output probability distributions and the probabilities of the state transitions are determined based on a training data set. Detailed operation of

statistical HMM-based speech synthesizers is beyond the scope of this thesis but the author acknowledges the fact that HMM-based synthesis is currently the most actively studied synthesis method. Hybrid synthesizers, based on the combination of concatenative synthesis and HMM-based synthesis, have also recently received an increasing amount of attention, e.g., in [Lin07], [Pol08], [Sil10], and [Tio11].

2.4.3 Concatenative synthesis and unit selection

Partially due to historical reasons, the group of concatenative speech synthesizers is typically considered to consist of two types of synthesizers, diphone synthesizers and unit selection based synthesizers. Diphone synthesizers, such as the early diphone synthesizer called Sparte [Cou82], utilize a database that contains one instance of each possible diphone in the given language (diphone is a unit consisting of two half-phones, covering a transition from one phone to another). The number of diphones to be stored in the database is language-dependent, for example, the French language contains about 1200 diphones, corresponding to about three minutes of speech [Dut97]. Thus, a small speech database is sufficient provided that it covers all the possible diphones. As a consequence, a lot of signal processing is needed to make the concatenation smooth and the output to have the desired prosody, and consequently the resulting speech quality tends to be compromised. The recent increases in the available memory sizes and in the processing power have made diphone synthesis largely an obsolete solution.

Unit selection based synthesis [Hun96], on the other hand, is currently the predominant technology behind most of the existing commercial TTS systems and applications. The main difference between diphone synthesizers and unit selection based synthesizers is the size of the database. Whereas only small databases are used in diphone synthesis, the unit selection based systems typically operate using large or sometimes even very large speech databases. As the name implies, there is now a need to select appropriate units, instead of just performing concatenation, because the large database typically contains numerous possible instances for each of the units (e.g., each of the diphones).

To clarify the process of unit selection, let us assume that the natural language processing front-end of a TTS system converts the input text into a target sequence, $\{t_1, t_2, \ldots, t_T\}$, where T denotes the total number of units to be concatenated to form the synthesized waveform. The units are picked from the entire speech database, and one candidate sequence could be expressed as $U = \{u_1, u_2, \ldots, u_T\}$. This brings us to a first central issue of the unit selection based concatenative synthesis approach: How to find an appropriate sequence of units from a large speech database in which multiple instances of each of the speech units would be available for concatenation? The classical solution proposed in [Hun96] is to define two costs for the evaluation of the distances between the candidate sequence and the target sequence: a target cost C_t and a concatenation cost

C_c [Hun96]. The target cost is an estimate of the difference between a database unit and the desired target unit. It is typically calculated as a weighted sum of sub-distances computed based on different pieces of phonetic and prosodic information. The concatenation cost estimates the quality of the concatenation of consecutive units, and it typically considers the acoustic characteristics at both sides of the concatenation boundary.

The total cost of using a candidate sequence to represent the desired target sequence can be calculated as the sum of the two costs, the target cost and the concatenation cost,

$$C = C_t + C_c. \tag{2.19}$$

The total cost is computed over the entire sequence, and the overall objective in unit selection is to find the optimal sequence $\overline{U}$ that satisfies

$$\overline{U} = \arg\min_{u} C, \tag{2.20}$$

i.e., the sequence having the lowest total cost over all the possible sequences should be selected.

Another central issue of unit selection based synthesis is the concatenation of the units. Compared to the case of diphone synthesis, the requirements for the signal processing techniques used for handling the concatenation are less stringent. If the database is large enough, there may not be any need to change the prosody of the concatenated speech. Nevertheless, it is still important to avoid audible discontinuities at the concatenation boundaries.

In theory, there is no upper limit for the quality of speech generated using a unit selection based system, besides the quality of the database recordings. Provided that particularly suitable units are found, the resulting speech quality can be excellent. In the extreme case, for example, an entire input sentence could be available in the database as such. In practice, however, it is very hard to achieve consistently excellent speech quality because inconveniently large database would be needed to be able to synthesize any input text with high quality. Thus, the output quality of a typical unit selection based TTS system is inconsistent and sometimes the system fails at finding a good unit from the database. The problems become even more pronounced if the aim is to produce highly natural and expressive speech.

As a rule of thumb, it can be concluded that the better the unit selection database, the better the resulting speech quality, provided that the unit selection algorithm works sensibly and that adequate signal processing is applied to treat the concatenation boundaries (and naturally also assuming that the front-end does not cause any problems). In addition to the coverage provided by the database design and the selected unit sizes, another crucial issue is the annotation of the database. In high quality TTS systems, the annotation, including the placement of the unit boundaries, is in practice performed manually, or at the very least the possible automatic annotations are manually verified and adjusted.

The final major issue in unit selection based synthesis is the compression and/or the representation of the speech databases. Since large databases are preferred, practical memory limitations often raise a need for the use of compression, especially in embedded applications. Moreover, even in cases where memory is not an issue, alternative parametric representations may be considered to facilitate the concatenation of the units. Chapter 4 addresses both of these issues. In addition, Section 4.3 presents a novel compression-motivated method for computationally efficient calculation of the concatenation cost. All the other parts of the unit selection based TTS systems are considered to be outside the scope of the thesis, and the discussions provided in the remaining chapters assume that the main TTS system and the required databases, including high-quality annotations, are readily available.

2.5 Voice conversion

The term voice conversion refers to the modification of the perceived speaker identity by modifying the speech signal uttered by a source speaker to sound as if it was spoken by a second speaker called the target speaker, without changing the lexical content of speech. In general, a voice conversion system is first trained using a relatively small amount of speech data from both the source and the target speakers, and then the trained models can be used for performing the actual conversion. This is demonstrated in Figure 2.7 that depicts a block diagram illustrating a typical stand-alone voice conversion system, i.e., a voice conversion system that can be used for modifying speech signals without support from any other systems. In addition to such stand-alone voice conversion systems, another main class of voice conversion systems can be thought to involve the use of adaptation techniques for modifying the speaker identity in statistical speech synthesis.

Commercial usage of voice conversion techniques has not been popular yet but potential applications for voice conversion include security related usage (hiding the identity of the speaker), entertainment applications, and text-to-speech synthesis in which voice conversion techniques can be used for creating new and personalized voices in a cost-efficient way. The topic of voice conversion has been an active area of research for more than two decades, and a large number of different voice conversion approaches have been proposed in the literature. Roughly speaking, one way to categorize the voice conversion techniques is to consider three different aspects: the requirements concerning the training material (and the processing of this material), the domain of the conversion, and the techniques used for the actual conversion. These issues are briefly discussed in the remaining parts of this section. A more thorough introduction to voice conversion is provided, e.g., in [Nur12].

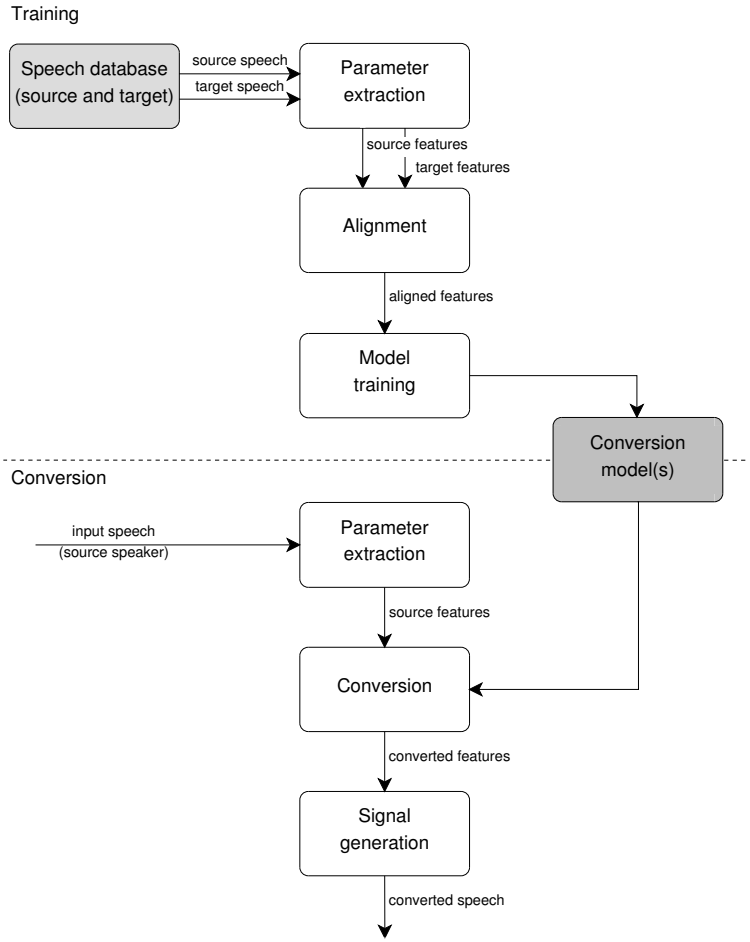

Figure 2.7: Block diagram illustrating stand-alone voice conversion. (From [Nur12].)

2.5.1 Requirements for the training data

For the training, the most common approach is to require parallel speech materials from the speakers, i.e., the same training sentences are spoken by both speakers. The training data is then typically further aligned as a preprocessing step, as depicted in Figure 2.7. Voice conversion systems that use parallel training materials are also sometimes referred to as text-dependent systems.

In some voice conversion approaches, such as in the one proposed in [Sün04b], the requirement to have parallel training materials is relaxed by allowing the usage of different speech content from the two speakers. The ultimate goal in this type of work is to establish a high-quality solution for cross-lingual voice conversion, i.e., for cases in which even the phoneme sets can be different for the source and

the target speakers. Voice conversion systems that can cope with non-parallel data can also be referred to as text-independent voice conversion systems.

2.5.2 Domain of conversion

Concerning the domain of conversion, many different, mostly parametric, approaches have been proposed in the literature. Most voice conversion systems utilize the source-filter model in some way, and the vocal tract contribution and the excitation are typically converted separately. However, other kinds of approaches have also been studied. For example, the direct frequency warping can be used for handling the conversion, either using a simple warping function formed based on a spectrum representing a single voiced frame [Shu06] or using vocal tract length normalization [Sün03].

The use of linear prediction, or in particular the LSF representation has been relatively popular in voice conversion systems. This approach has been used, e.g., in [Ars99], [Tur06], and [Err10]. In addition to the linear prediction based methods, cepstrum-based parameterization has been widely used, for example in the form of MFCCs [Sty98]. Yet another popular approach, used widely both in voice conversion and in statistical speech synthesis (e.g., in [Tod07], [Des10], [Hel10], and [Tok02]), is to utilize *mel-cepstral coefficients* (MCCs) computed using the generalized mel-cepstral analysis method [Tok94] that provides a flexible method controlled via two parameters that adjust the balance between the cepstral and linear prediction representations and the frequency resolution of the spectrum.

There are also several different proposals for the modeling and conversion of the excitation signal in voice conversion. For example, in [Abe88] the excitation was modeled in a very simple manner using only pitch and energy parameters, and in [Kai98] the source excitation was used with only pitch modification. In contrast, a more sophisticated harmonics + noise model was used in [Sty98]. Naturally, the selection of the excitation model and the related conversion requirements depend also largely on the parameterization used for modeling the vocal tract contribution.

Even though the aim of this section is not to provide a complete list of voice conversion approaches or parameterizations proposed in the literature, there is still one more alternative that needs to be mentioned: the STRAIGHT vocoder [Kaw99]. The use of STRAIGHT has recently been widely popular in HMM-based speech synthesis, and consequently also in voice conversion research, especially in studies dealing with adaptation based voice conversion techniques used in HMM synthesis.

2.5.3 Conversion methods

The most popular approach for the actual conversion task has been *Gaussian mixture model* (GMM) based conversion, proposed in slightly different ways in

[Kai98] and [Sty98]. The former approach uses GMMs for modeling the density of the source features while the latter models the joint density of both source and target features. The GMM-based conversion approach, implemented as proposed in [Kai98] is also used in Chapter 5. More information on this simple but effective approach is provided in Section 5.1.

In addition to the GMM-based approach, a wide variety of different conversion techniques have been proposed in the literature. Examples of different approaches include neural network based conversion studied in [Nar95], [Wat02], and more recently in [Des10], hidden Markov model based conversion [Kim97], codebook based conversion studied, e.g., in [Abe88], [Ars97], and [Esl11], and non-linear conversion techniques such as [Son11] and [Hel12]. In addition, there have been proposals that combine these different approaches. For example, a hybrid approach combining GMM-based conversion and codebook based conversion has been proposed in [Kan05]. More information on the different conversion methods can be found, e.g., in [Nur12], and in the other references mentioned in this section.

Chapter 3

VLBR – segmental speech coding for efficient storage

In speech storage applications, many of the traditional speech codec design constraints discussed in Section 2.2 can be relaxed in order to achieve higher quality and/or lower bitrate [Won92][Mud98]. For example, the limitations regarding encoding delay can be relaxed or omitted and the lack of bit errors in most storage applications enables the use of lossless coding and/or all kinds of predictors and memory-based solutions. In addition, variable bitrate can be conveniently used to adaptively adjust the parameter update rate [Rou82][Lee01] and the quantization accuracy based on the short-time properties of the input speech. All of these aspects are considered in the development of the VLBR codec and the related compression techniques presented in this chapter.

Since the aim was to achieve relatively good speech quality, speech coding solutions such as the ones presented, e.g., in [Rou82], [Rou83], [Shi88], [Pic89], and [Cer98], aiming at producing intelligible speech at extremely low bitrates as low as 0.15 kbps, were considered too coarse, and the usual limitation to only one speaker was also seen undesirable. Furthermore, since the memory consumption of the decoder had to be kept as small as possible, solutions based on the use of a large speech database (see, e.g., [Lee01] and [Lee02]) were also readily out of the question. Thus, the development approach chosen was to start with the models and techniques typically used at bitrates between 1.2 and 4.0 kbps and to develop further solutions for making the overall process more efficient.

The first section of this chapter introduces the parametric representation used in the VLBR codec. The parametric representation itself or the parameter estimations are not considered to be core contributions of this thesis but short descriptions are provided due to their important role in the VLBR codec and consequently in the whole thesis. Section 3.2 can be regarded as the central section of this chapter since it provides an overview of the first developmental version of the VLBR codec and introduces the related mode-based segmental processing and quantiza-

tion solutions. The remaining parts of the chapter discuss additional techniques that can be used to further enhance the efficiency of the VLBR codec. In Section 3.3, a specific general-purpose quantizer structure, based on the multi-mode matrix quantization of adjacent parameter vectors using low-complexity vector-based predictions, is introduced. In Section 3.4, further bitrate reductions are sought for by considering the possibility to compress the quantizer index data using lossless compression. In particular, an enhanced version of the conventional dynamic codebook reordering technique [DeN96] is proposed and evaluated. The final core contribution of this chapter, a novel preprocessing method based on perceptual irrelevancy removal, is presented in Section 3.5.

The discussions provided in the main parts of this chapter, i.e., Section 3.2, Section 3.3, Section 3.4, and Section 3.5, are largely based on the publications [Räm04], [Nur03b], [Nur07a], [Nur06a], and [Läh03b].

3.1 Parametric representation

The selection of the parametric model is one of the crucial issues when designing a parametric speech codec. The main reason for this is that the chosen parametric representation directly sets an upper limit on the achievable speech quality. The parametric model also indirectly affects the bitrate of the codec because the perceptual importances of different parameters are different, and there are also differences in the achievable compression efficiencies, i.e., the number of bits needed for perceptually accurate representation of a parameter value or a set of parameter values varies a lot.

As discussed in Section 2.2, most of the modern speech codecs utilize the idea of linear prediction due to its beneficial properties. Thus, it was an easy decision to base the operation of the new storage codec on linear prediction. The selection between the excitation models based on waveform interpolation and sinusoidal modeling was not as straightforward but the sinusoidal modeling approach was eventually chosen. One of the reasons for this selection was the fact that the resulting parameter set is slightly more compact but it is worth noting that a similar codec could be built around the waveform interpolation approach as well.

3.1.1 Excitation modeling

The sinusoidal modeling approach used in this thesis is based on the fact that a vocal tract excitation signal, or alternatively a speech signal, can be represented as a sum of sine waves of arbitrary amplitudes, frequencies and phases [McA86] [McA95]. To obtain a frame-wise representation, the parameters are assumed to be constant over each analysis frame. Consequently, the discrete excitation signal

$r(t)$ in a given frame can be approximated as

$$r(t) = \sum_{m=1}^{L} A_m cos(t\omega_m + \theta_m), \tag{3.1}$$

where A_m and θ_m represent the amplitude and the phase offset of each sine-wave component associated with the frequency track ω_m, and L denotes the number of sine-wave components.

One way to utilize the above model would be to model only the voiced contribution using Equation (3.1) and the unvoiced contribution could be modeled separately as a spectrally shaped noise. However, it is also possible to generate both voiced and unvoiced content using Equation (3.1). One alternative is to assume that the sinusoids can be classified as continuous or random-phase sinusoids. The continuous sinusoids represent voiced speech and at low bitrates they can be modeled using linearly evolving phases. Due to this crude modeling, the original phase information is lost but the resulting quality loss is not severe (the human ear is generally not considered to be very sensitive to phases, and the model works relatively well). The random-phase sinusoids, on the other hand, are used for modeling unvoiced noise-like speech. A separate voicing decision can be made for each sinusoid separately or for different frequency bands. With split-band voicing, the voicing information can be represented using a single value that represents a voicing cut off frequency: below this frequency all the sinusoids are deemed voiced whereas the content above this frequency is considered unvoiced.

It is also possible to allow voiced and unvoiced content to co-exist at the same frequency. In this scenario, the model can be rephrased as

$$r(t) = \sum_{m=1}^{L} A_m(v_m cos(t\omega_m + \theta_m^V) + (1 - v_m)cos(t\omega_m + \theta_m^U)), \tag{3.2}$$

where v_m is the degree of voicing for the mth sinusoidal component ranging from 0 to 1, while θ_m^V and θ_m^U denote the phase of the voiced and unvoiced portions of the mth sine-wave component, respectively. This model, together with the above-mentioned approach of using linearly evolving phases for the voiced content and random phases for the unvoiced content, allows for a reasonably accurate approximation of the original speech signal from the perceptual point of view, despite the fact that the waveform-level modeling will not be accurate anymore.

To further simplify the representation with only a very small loss in modeling capability, it is also possible assume that the sinusoids used in the modeling are always harmonically related, i.e., that the frequencies of the sinusoids are integer multiples of the fundamental frequency ω_0. With this assumption, the model becomes

$$r(t) = \sum_{m=1}^{L} A_m(v_m cos(tm\omega_0 + \theta_m^V) + (1 - v_m)cos(tm\omega_0 + \theta_m^U)). \tag{3.3}$$

During voiced speech, ω_0 corresponds directly to the F_0 associated with the analysis frame, i.e.,

$$\omega_0 = 2\pi F_0 / F_s, \tag{3.4}$$

where F_s denotes the sampling frequency. During unvoiced speech, there is no physically meaningful F_0 available but it is possible to use a fixed value for ω_0 because the random phases used for generating unvoiced signal ensure that the resulting signal will not be periodic. The use of Equation (3.3) simplifies the selection of the parameter L as it can always be set to the number of harmonics that fit into the available frequency band.

3.1.2 Parameter estimation

The speech representation used in this thesis consists of three elements: i) the vocal tract contribution modeled using linear prediction and more specifically line spectral frequencies, ii) overall gain/energy, iii) normalized excitation spectrum. The third of these elements, i.e. the residual spectrum, is further parameterized using the fundamental frequency F_0, the amplitudes of the sinusoids, and voicing information, as described above. Each of these parameters is estimated at 10-ms intervals from 8-kHz input speech signals.

The detailed parameter estimation methods are not considered to be a core contribution of this thesis. Nevertheless, a high-level description of the main steps of the parameter estimation process is provided in this subsection. It should be noted that the topic of parameter estimation for sinusoidal modeling is a widely studied and challenging research topic on its own, and the estimation methods and design choices briefly summarized below are definitely not the only alternatives and almost certainly they are not the most accurate and robust alternatives either. Discussions on parameter estimation related issues can be found in a plethora of publications, for example, in [McA86], [McA90], [McA95], [Sty01], [Vil01], [Qua02, Chapter 9], [Hei02], [Kon04, Chapter 8], [Yan09], just to name a few examples.

Line spectral frequencies

The coefficients of the linear prediction filter are estimated using the autocorrelation method and the well-known Levinson-Durbin algorithm, together with mild bandwidth expansion. This approach ensures that the resulting filters are stable and helps in preventing unnatural spectral peaks due to interactions of pitch and the formants. Each analysis frame consists of a 25-ms speech segment, windowed using a Hamming window. The degree of the linear prediction filter is set to 10 for 8-kHz speech. For further processing, the linear prediction coefficients are converted into the line spectral frequency representation. This widely-used representation is very convenient since it has a close relation to formant locations and

bandwidths, and it offers favorable properties for different types of processing and guarantees filter stability.

Pitch / fundamental frequency

The first step in F_0 estimation is frequency-domain pitch estimation. A Fourier transform is computed using a variable-length window, with the window length determined by the previous F_0 value. For all integer-length pitch periods, a frequency-domain metric is computed using a sinusoidal speech model matching based approach [McA90]. A time-domain metric measuring the similarity of successive pitch cycles is computed for a fixed number of pitch candidates with the highest frequency-domain scores. The actual pitch estimate for each analysis frame is obtained by applying a pitch tracking algorithm for a fixed number of potential candidates that received the highest combined scores based on the two above-mentioned metrics. The pitch tracking algorithm utilizes a fixed number of previous pitch values and favors continuous evolvement of pitch tracks. As the last step, the pitch estimate is further refined using a sinusoidal speech model matching based technique to provide an accuracy higher than one-sample that is essential for perceptually accurate F_0 modeling.

Voicing

The voicing information is estimated based on the refined F_0 value. Again, the estimation is performed using the frequency domain representation generated by Fourier transform with variable-length windowing. The voicing information is derived for the residual spectrum and the analysis of voicing-specific spectral properties is carried out separately at each harmonic frequency. A voicing likelihood in the range from 0 to 1 is determined for each F_0 harmonic by evaluating the normalized correlation between the spectral shape of the frequency band surrounding the harmonic frequency and the spectral shape of the variable-length window applied on the signal. The voicing likelihoods can be used as v_m in Equation (3.3).

To enable efficient compression at the lowest bitrates and easy segmentation (to be discussed in Section 3.2.2), the harmonic likelihoods are also simplified to a single voicing parameter, a voicing cut-off frequency. The voicing cut-off frequency divides the residual spectrum into a voiced lower part and an unvoiced higher part, i.e., the voicing values for the harmonics below the cut-off frequency are set to unity and the voicing values above it are set to zero. The determination of the voicing cut-off frequency is performed already during parameter estimation as opposed to performing it as a separate quantization step to be able to also utilize waveform level information: the cut-off frequency leading to the lowest possible modeling error is selected. In the implementation discussed in this chapter, the voicing cut-off frequency is represented as an integer in the range from 0 to 7.

The value of 0 corresponds to a fully unvoiced frame and the value 7 represents a fully voiced frame.

Harmonic amplitudes

Once the F_0, or equivalently ω_0, and the voicing parameters have been estimated, the next step is to compute the spectral harmonic amplitude values from the *fast Fourier transform* (FFT) spectrum. Each FFT bin is associated with the harmonic frequency closest to it. This results in estimated harmonic amplitude values for each integer multiple of F_0. It should be noted, however, that for unvoiced segments where there is no true periodicity, a constant forced F_0 is used.

Energy

The overall energy is estimated from the time-domain speech signal, using the root mean square energy. Since the frame-wise energy varies significantly depending on how many pitch peaks are located inside the analysis frame, the estimation computes the energy of a pitch-cycle length signal instead. The pitch cycle that is located closest to the center of the analysis frame is used for representing the energy level of the given frame.

3.1.3 Speech signal reconstruction

At the decoder, a speech signal can be reconstructed simply using Equation (3.3), together with linear prediction synthesis filtering. The signal generation could be realized frame by frame, e.g., using overlap-add but in the implementation discussed in this thesis, pitch-synchronous processing is used, partially to obtain low computational load. Thus, Equation (3.3) is applied one pitch cycle at a time. The parameter values for the pitch cycle are obtained using interpolation. Once the residual signal is reconstructed, it is scaled using the energy parameter and filtered using the corresponding LP analysis filter to generate the speech signal. Finally, the energy level is readjusted to take into account the changes caused by the LP filtering.

Due to the parameter interpolation, the parametric representation can be considered to consist of piece-wise continuous parameter tracks. This interpretation enables simple high-quality playback speed alteration as explained in [Nur06b]. Due to the simplified model, the process is much more straightforward than, e.g., in [Qua92]. The playback speed can be taken into account simply by requiring at the decoder that the center of the pitch cycle, t_{center}, must satisfy the condition

$$2\left(t_{\text{center}} - t_{\text{boundary}}\right) = \nu p(t_{\text{center}}), \tag{3.5}$$

where ν is the relative playback speed, $p(t)$ denotes the time-domain pitch parameter at the time instant t, and t_{boundary} denotes the "virtual boundary" of the

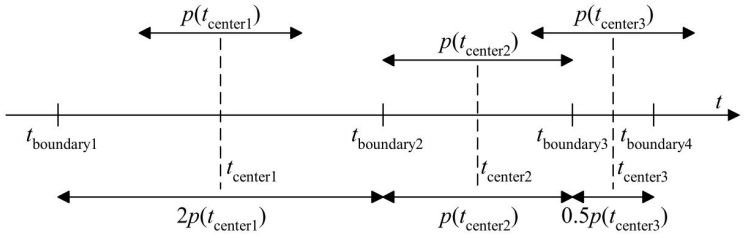

Figure 3.1: Three examples illustrating the use of different playback speeds. (From [Nur06b].)

previous pitch cycle (at the beginning of the synthesis process t_{boundary} is set to 0). After a center that satisfies the above condition is found, t_{boundary} is updated using

$$t_{\text{boundary}} = t_{\text{boundary}} + 2\left(t_{\text{center}} - t_{\text{boundary}}\right), \tag{3.6}$$

before finding the next pitch cycle center. The pitch cycle centers obtained in this manner are the time instants for which the parameter values are computed during pitch-synchronous signal generation.

It is easy to see that if the pitch contour is continuous, there always exists a center satisfying the condition in Equation (3.5). If the contour is only piece-wise continuous, as is the case in the model described in this section due to the use of constant pitch values during unvoiced speech, the discontinuity points have to be regarded as special cases. With the simple piece-wise linear representation used in [Nur06b] and in the practical implementation discussed in this chapter, the search for the next pitch cycle center should be done one linear piece at a time. Furthermore, t_{center} can be found by first finding the candidate

$$t_{\text{candidate}} = \frac{\left(\frac{2t_{\text{boundary}}}{\nu} + p(t_{n-1})\right)(t_n - t_{n-1}) - t_{n-1}\left(p(t_n) - p(t_{n-1})\right)}{\frac{2(t_n - t_{n-1})}{\nu} - \left(p(t_n) - p(t_{n-1})\right)}, \tag{3.7}$$

and by checking whether the obtained $t_{\text{candidate}}$ is in the current interval between t_{n-1} and t_n. If it is, then $t_{\text{center}} = t_{\text{candidate}}$. Otherwise, n is increased by 1 and the equation is solved for the next interval, etc.

The process of finding the center of the next pitch cycle using different playback speeds is demonstrated in Figure 3.1. The figure illustrates the center locations found in three different situations:

1) Playback using double speed ($\nu = 2$)

2) Normal playback speed ($\nu = 1$)

3) Half speed ($\nu = 0.5$)

In the figure, these situations are denoted with the corresponding subscript. For example, $t_{\text{boundary1}}$ denotes the value of t_{boundary} when searching for the center t_{center1}.

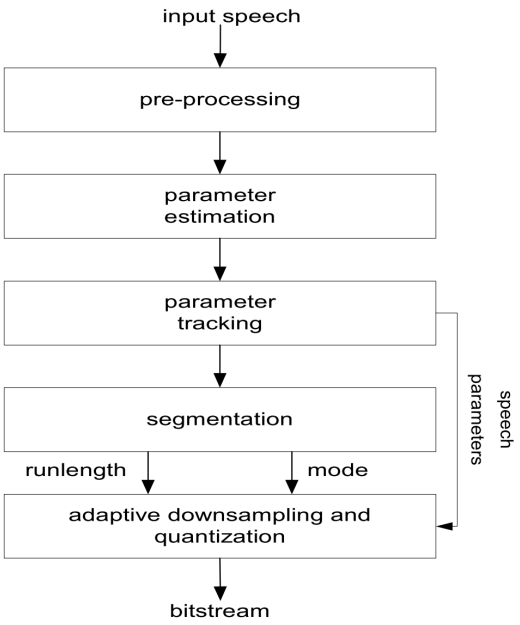

Figure 3.2: Proposed speech coder structure. (From [Räm04].)

3.2 VLBR speech codec

As discussed in the beginning of this chapter, it is possible to relax many of the traditional design constraints when designing a speech codec for storage applications. In this section, this is exploited in the development of a new parametric speech codec structure for low-bit-rate storage applications. In the proposed segmental speech coding technique, each speech parameter is estimated, tracked, downsampled, and quantized at a variable bitrate based on its time-varying perceptual relevance. The performance improvement achieved using the proposed structure is confirmed in a formal listening test. The results presented in this section have been published in [Räm04].

3.2.1 Overview of the proposed coder structure

The high-level encoder structure of the proposed speech coder is illustrated in Figure 3.2. In this structure, the encoding starts with the estimation of the parameter values from a preprocessed speech signal. Then, a tracking procedure is performed on the parameter values to remove possible outliers that might negatively affect the compression efficiency or speech quality. The tracked parameter values are grouped into variable-length segments that are coded using adaptive downsampling and quantization. As can be seen from Figure 3.2, this encoding procedure can be illustrated using five high-level blocks that take care of pre-

processing, parameter extraction, parameter tracking, segmentation, and adaptive downsampling and quantization. This subsection will briefly go through the first three blocks. The last two blocks are described in the subsequent subsections. In addition to the blocks listed here, lossless compression could be applied as the last block as well, to make the bitstream even more compact. In this section, lossless compression is not considered but this topic will be discussed in Section 3.4.

Preprocessing

In the proposed speech coding approach, the input speech signals can be preprocessed to enhance the speech quality and to increase the coding efficiency since exact reproduction of the original speech signal is not required. The optimal operation of the preprocessing step depends on the target application and on the properties of the input signals but the most obvious goals are to make the input signals possibly recorded in slightly different conditions more uniform and to attenuate possible background noise. In the practical codec implementation discussed and evaluated in this section, as well as in all the other parts of this thesis, the input signals were expected to be clean speech signals and consequently the preprocessing block only included a modified *intermediate reference system* (IRS) [ITU00] filter and a block for adjusting the signal energy to a specified level. If the codec were to be used for noisy speech, a noise suppressor could also be included as an additional preprocessing block.

Another, yet a bit less obvious, option is to modify the input signal in such a manner that more efficient compression becomes possible, while still keeping the modifications perceptually inaudible or insignificant. In this section, this option is not exploited but Section 3.5 will introduce a preprocessing method that can effectively remove perceptually irrelevant content from speech input. This reduces the amount of information stored in the signal which in turn enables more efficient compression. While the method is very suitable for the VLBR codec, it is fully codec-independent and can also improve the efficiency of standardized speech codecs, as will be shown in Section 3.5.

Parameter estimation

The time-varying speech parameters are estimated from the preprocessed input speech. In principle, the proposed codec structure could be utilized with different parameterizations but in this thesis and in the implementation evaluated in Section 3.2.5 the parameterization described in Section 3.1 was used. Consequently, the parameter set consisted of linear prediction coefficients represented as LSFs, gain, pitch, voicing, and harmonic excitation amplitudes. As in Section 3.1, input signals with a sampling rate of 8 kHz were used and the parameter estimation interval was 10 ms for all the parameters. The degree of the LP filter was set to 10.

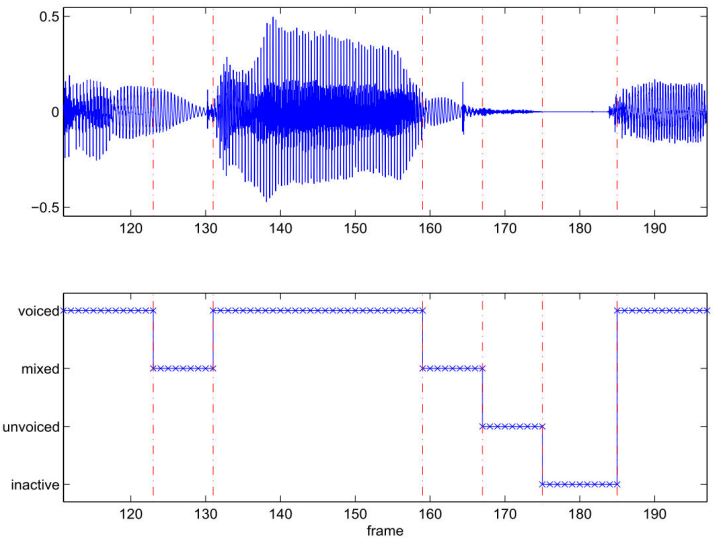

Figure 3.3: Segmental nature of speech (the frame length used in these plots is 10 ms). (From [Räm04].)

Parameter tracking

The coding efficiency can be improved with additional processing of the speech parameter tracks, i.e., using parameter tracking. The main target of the parameter tracking step is to remove possible perceptual artifacts caused by incorrectly estimated parameter values. Another target is to remove possible parameter outliers that might unnecessarily effect the segmentation decisions to be discussed in Section 3.2.2. These targets were achieved in the implementation discussed in this section first by searching for possible gross pitch detection errors and by replacing clear outliers with values fitting a continuous pitch track. Then, the second step was to find and manipulate voicing and gain values that would result in very short segments that could also be compressed using a neighboring mode. For example, very short unvoiced segments with low energy can be filtered out without significantly decreasing the speech quality.

3.2.2 Segmentation

Different properties of different types of speech signals are exploited in mode-based speech coding solutions. The mode selection can be based on the minimization of the coding error (see, e.g., [Tan88], [Jay89], and [Kro88]) or on signal analysis, typically through the estimation of a parameter or a set of parameters that can be used for determining the signal type, as proposed, e.g., in [Tre82], [Oza94], and [Pak97]. Typically, the mode selection is performed frame by frame

but the relaxation of the requirements for the encoding delay allows the segmental nature of the speech to be further exploited in the coding system. Consequently, in the proposed approach, the input speech signal is divided into segments of voiced speech, unvoiced speech, transitions (mixed voiced speech) and pauses (silence). The motivation behind this four-mode approach is that these four types of speech segments have clearly different physical and perceptual properties. Support for this approach was also found in [Pak97] that also proposed to use four modes that were roughly similar except for the case of the silence mode (a noise mode was used instead of the silent mode). Furthermore, the division into these four classes is consistent with the approximately segmental nature of speech illustrated in Figure 3.3. The upper half of the figure shows a short part of a speech signal and the lower half illustrates the segmentation. The segment boundaries are denoted with dash-dotted vertical lines.

The segmentation of the speech signal can be based either on the parametric representation used in the speech modeling or on additional information derived from the speech signal itself, as long as the segments are chosen so that the intra-segment similarity of the speech parameters is high. In the implementation evaluated in this section, the segmentation was based on the gain and voicing parameters. First, inactive segments were detected by classifying all the 10-ms frames having a gain value below a predetermined threshold as inactive and by combining adjacent inactive frames into longer segments. Second, the remaining frames were classified into three distinct classes based on the voicing parameter values. The classes were *voiced*, *unvoiced*, and *mixed*. The maximum segment length was set to 640 ms.

To further improve the coding efficiency, the initial segmentation achieved using the procedure discussed above was modified using backward and forward tracking. The aim of this tracking approach is to improve the segmentation by reducing the effect of single voicing outliers. For example, very short mixed segments between two voiced segments are eliminated by merging them with the neighboring segments. This tracking scheme, along with the whole segmentation process, is illustrated in Figure 3.4 where it can be seen how the tracking smoothes single voicing outlier peaks. As a consequence, the average segment length is increased which in turn improves the coding efficiency.

After successful segmentation, each segment can be efficiently quantized using a coding scheme designed specifically for the corresponding segment type. These coding schemes include quantizer designs, bit allocations for the quantizers and information on the perceptually sufficient update rates. The bit allocations for the quantizers depend directly on the segment type: more bits are allocated for quantizing the parameter values inside voiced or mixed segments than for values inside unvoiced or silent segments. In the implementation evaluated in this section, the update rates were determined as presented in Table 3.1. As can be seen from the table, the update rates are adaptively adjusted for most of the parameters.

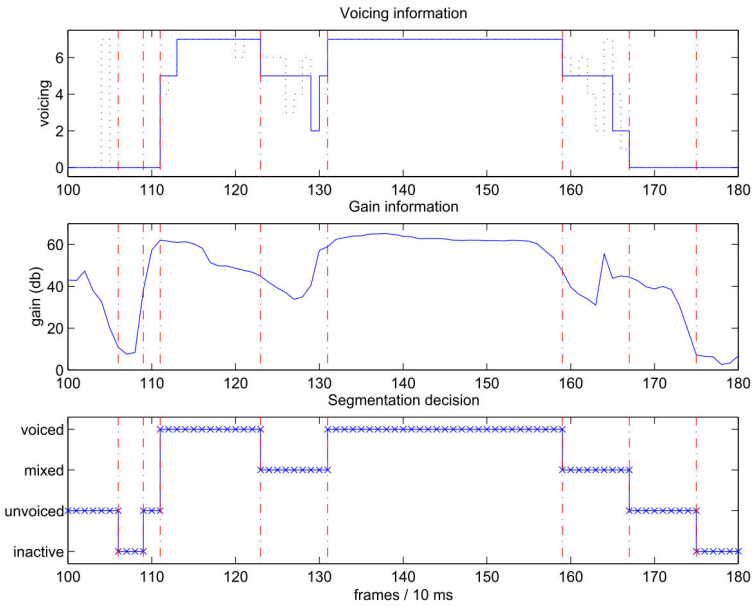

Figure 3.4: Practical example illustrating the segmentation process. (From [Räm04].)

3.2.3 Adaptive downsampling and quantization

Adaptive downsampling and quantization is performed for one speech segment at a time and within each segment the process is gone through in two phases. First, the target accuracy for the coded parametric representation is adaptively selected based on the segment type. The selected accuracy level also determines the number of bits used in the quantization of a single parameter value. Then, a downsampling ratio and quantization that just meets the accuracy requirement is searched and used. At the decoder, the reduced update rate used in the quantization is upsampled back to the original fixed update rate of 100 Hz. The down- and upsampling steps can be performed using any appropriate method. In the practical implementation tested in this section, these update rate conversions were computed using discrete cosine transform. This approach is described more closely in Section 3.2.4.

The search for the optimal downsampling rate is illustrated in Figure 3.5. Let k denote the number of parameter values (for one parameter) inside the current segment. The iteration begins by downsampling the k parameter values to i values (during first iteration $i = 1$). Then, the i values are quantized and upsampled back to k parameter values. After this, the distortion caused by these processing steps is measured using one or more distortion measures (in the practical implementation discussed in this section, the distortion was measured using weighted mean squared error and maximum absolute error). If the distortion is low enough, ac-

Table 3.1: Update rates used for the speech parameters during different segment types. The symbol - indicates that the corresponding information is not needed. In the ~1.0 kbps mode, the amplitudes were set to a fixed value, whereas in the ~2.4 kbps mode they were coded for all active frames (100 Hz).

	Voiced	Mixed	Unvoiced	Silent
LSFs	adaptive	adaptive	50 Hz	-
Gain	adaptive	adaptive	adaptive	adaptive
Pitch	adaptive	adaptive	-	-
Voicing	once	100 Hz	once	-
Amplitudes	-/100Hz	-/100Hz	-/100Hz	-

cording to the target accuracy defined for the segment, the iteration can be finished and the found k:i is the optimal downsampling ratio. Otherwise, i is incremented by one and compared to a limit determined in such a manner that a downsampling ratio k:limit provides perceptually satisfactory quality. If i is smaller than this limit, the iterative process is continued; otherwise, the process is terminated and the downsampling ratio k:limit is used.

The adaptive downsampling and quantization scheme significantly increases the coding efficiency when compared to conventional approaches with fixed bit allocations and parameter update rates. The improvement is achieved because both the parameter update rate and the bitrate are locally optimized for each speech segment, individually for each parameter. Consequently, the update rate and the bitrate can always be kept as low as possible while still maintaining an adequate perceptual quality. During perceptually sensitive portions of speech, a sufficiently high update rate and/or bitrate can be temporarily used without significantly increasing the average bitrate.

3.2.4 Update rate and vector dimension conversions

As mention above, in the VLBR codec implementation discussed in this section, the update rate conversions were carried out using discrete cosine transform. DCT-based conversion was also used in the quantization of the variable-dimension harmonic amplitude vectors.

The decision to use DCT for the update rate conversions and vector dimension conversions was based firstly on the beneficial properties of the method, supported by concrete vector dimension conversion related results in numerous publications, e.g., in [Cup95], [Lup95], [Lup96], and [Li98]. The DCT based approach used in these publications can be regarded as a special case of *non-square transform vector quantization* (NSTVQ) [Lup94] that offers a generalized tool for describing vector quantizers that utilize some linear dimension conversion scheme. In this technique, a non-square transform is employed to map variable-dimension in-

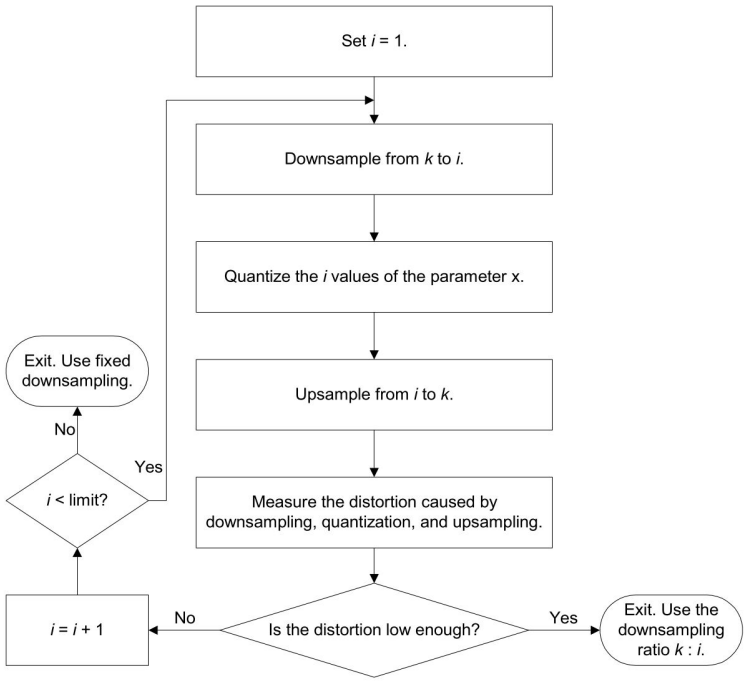

Figure 3.5: Algorithm for searching the optimal downsampling ratio. (From [Räm04].)

put vectors into fixed-dimension vectors prior to quantization. At the decoder, the fixed-dimension vectors are converted into their original dimensions using the corresponding inverse transform. With appropriate selections of the transforms, all linear techniques for dimension conversion can be represented using the NSTVQ approach [Shl98].

The selection of the DCT based approach for the update rate and vector dimension conversions was also further supported by the results obtained in comparative experiments [Nur01b] that evaluated the performance and complexity of five different dimension conversion techniques in the context of quantizing the variable dimension spectral vectors in waveform interpolation speech coding, as well as the good performance observed later, e.g., in the practical use cases documented in [Nur01c] and [Nur02]. The five methods included in the comparison [Nur01b] were the simple truncation and zero-padding, the use of frequency bins implemented using the variable-dimension vector quantization technique proposed in [Das96], band-limited interpolation based conversions implemented using cubic B-splines [Uns99], DCT-based NSTVQ, and polynomial approximation based conversion implemented using Chebyshev polynomials as described in [Pre92]. The main outcome of the comparison was that DCT-based NSTVQ provided the most accurate quantization results in all the tested conditions.

The use of DCT for update rate conversions, in addition to the use in vector dimension conversions, was further supported by additional experimentations related to this particular task. More specifically, as a part of this thesis work, the performance of the update rate conversion was informally evaluated by expert listeners, and it was confirmed that the method is successful as long as the resulting distortion is kept low enough, using the approach of adaptive downsampling and quantization described above. In other words, the perceptual performance was found to be correlated with the distortion level, regardless of whether the distortion was caused by the combination of adaptive downsampling and quantization or the quantization alone.

DCT-based dimension conversion can be implemented simply as follows. First, the discrete cosine transform is applied to the original vector. Then, the DCT-domain vector is truncated or zero-padded to the desired length. Due to the truncation or zero-padding, the DCT coefficients also need to be normalized by the factor $\sqrt{L}$, where L denotes the original variable length. In the case of vector quantization, the quantization can be performed for the resulting fixed-dimension DCT-domain vector. After quantization, the vector can be converted back to its original dimension using inverse DCT. If desired, the process can be implemented using multiplications with the corresponding transformation matrix and its inverse transformation matrix, as described in [Lup94].

In the case of update rate conversion, the process is similar but slightly different. First, the vector is formed using the parameter values at different time instants. Another difference is that instead of performing all the further processing in the DCT domain, the vector is returned into the original domain for further processing (quantization and accuracy measurement) but to another dimension. For example, in case the original update interval was 10 ms and the segment contained 8 update instants (frames), corresponding to a segment of 80 ms, initial vectors of dimension 8 can be used. Then, when the update is, e.g., halved, inverse DCT is used for obtaining 4-dimensional vectors in the original domain, corresponding now to the update interval of 20 ms. Once the desired update rate is found, the final quantization is carried out, and the decoder can then perform a similar update rate conversion but now back to the original update rate.

3.2.5 Performance evaluation

A preliminary version of the VLBR codec utilizing the proposed segmental processing techniques was evaluated in a listening test. The average bitrate of the codec was adjusted to a level of about 1 kbps or below. Another codec bitrate of about 2.4 kbps was also tested. The higher bitrate was obtained in a very simple manner by using a 15-bit multistage vector quantizer for the harmonic amplitude information in active frames. More specifically, the variable-dimension amplitude vectors were quantized as DCT-domain vectors of fixed dimension using the DCT-

Table 3.2: Bit allocations for the sinusoidal coders evaluated in the listening test. For the proposed coder, the bit allocations depend on the input and thus the numbers given in the table are averages obtained using an exemplary speech file (the duration of this speech sample was 10 minutes and the speech activity level was 90%).

	Proposed	Conventional
LSFs	660	800
Pitch	105	170
Voicing	10	50
Energy	151	180
Residual amplitudes	0 / 1346	0
Segmentation	71	0
TOTAL	997 / 2343	1200

based dimension conversion approach described earlier in this section. The simple squared error was used as the distortion measure, i.e., no perceptual weighting was used for the amplitude vectors in this first implementation.

For all the other parameters, the VLBR codec versions for the two bitrates were similar. The quantizer designs in the evaluated implementation were fairly simple: The segmentation output was stored without compression, the integer voicing values were stored as such but only for the mixed mode, and the energy values were quantized in the logarithmic domain. LSFs were quantized using multistage vector quantization and the perceptually-motivated weighting scheme [Pal93], also described in Section 3.3.3. Predictive quantization was also utilized in the voiced and mixed modes. The pitch values were quantized using the simple approach presented in [Nur06b]. Even though these parameter quantizations were quite efficient, most of the bitrate savings were offered by the segmentation process and the method of adaptive downsampling and quantization.

A conventional state-of-the-art proprietary sinusoidal coder operating at 1.2 kbps (using six-frame superframes for efficient quantization) was also included in the evaluation as a benchmark. The bit allocations for all of these sinusoidal coders are given in Table 3.2. In addition to these coders, the standardized *mixed excitation linear prediction* (MELP) codec [Sup97], the unquantized sinusoidal coding model described in Section 3.1, and clean direct unprocessed speech signals were also included in the evaluation.

The listening test was carried out using the *absolute category rating* (ACR) [ITU96] procedure with a five-level *mean opinion score* (MOS) scale. The test material was taken from a high quality Finnish language database and downsampled to 8 kHz. The test sentences were spoken by four male and four female speakers. Four sentence pairs were selected from each speaker.

Table 3.3: Listening test results (MOS scale 1–5). The absolute numbers do not carry any inherent meaning but the relative differences are meaningful.

	Female	Male	Total
Direct	4.16	4.48	4.32
Sinusoidal model	3.63	4.00	3.82
MELP (2.4 kbps)	2.95	3.04	2.99
Proposed (2.4 kbps)	3.09	3.55	3.32
Proposed (1 kbps)	2.51	3.11	2.81
Conventional (1.2 kbps)	2.28	2.83	2.55

The test results obtained from 24 naive listeners are presented in Table 3.3. As can be seen from the MOS values, the proposed segmental speech coding approach outperformed the traditional approach by a wide margin even though many parts of the implementation were not optimized to perfection. When compared to the standard 2.4 kbps MELP codec, the proposed codec was already quite near with the bitrate of 1.0 kbps. With similar bitrates, the proposed codec was significantly better.

The simplified sinusoidal modeling approach can also be considered successful: without quantization, the speech quality is relatively close to that of the direct reference. It should be noted that even though there was no quantization for the parameters in general, the integer-based representation of voicing values described earlier in Section 3.1.1 was used instead of harmonic voicing likelihoods to evaluate the actual model used at the lowest bitrates. In addition, before parameter estimation, the speech samples were first processed using a modified IRS [ITU00] filter, and a low-complexity fixed-point implementation of the signal generation was used in the evaluation instead of the higher-complexity floating point version, to obtain results that realistically demonstrate the performance of the model in a practical application.

It should be noted that in the tested implementation, the footprint of the decoder was very small: the worst-case complexity of the fixed-point decoder was only about 7 WMOPS (*weighted million operations per second*) and the amount of memory occupied by the codebooks was only about 7 kilobytes. For comparison, the decoder of the conventional sinusoidal coder used as a benchmark was more complex and contained codebooks of more than 200 kilobytes in size. The encoding complexity is naturally higher with the proposed structure but it is still reasonable with approximately 70 WMOPS.

3.3 Vector-predictive multi-mode matrix quantization

As discussed in [Gra84], and briefly mentioned in Section 2.3.1, a fundamental result of Shannon's rate-distortion theory [Sha59] is that joint quantization of elements is always more efficient, in terms of bitrate and/or coding accuracy, than quantization of single elements. The result is valid even for uncorrelated or independent data [Gra84]. In parametric speech coding, this information can be exploited in two ways: by quantizing vectors instead of scalars and by employing joint quantization of two or more adjacent speech parameter values or vectors whenever this is possible (the joint quantization of two or more vectors is referred to as *matrix quantization* (MQ) [Tsa85]). In the VLBR codec, and in storage coding in general, the freedom to use longer encoding delays enables meaningful use of matrix quantization. As a consequence, either better quantization accuracy can be achieved while maintaining the bitrate or a lower bitrate can be achieved while maintaining the quantization accuracy.

In this work originally presented in [Nur03b], the efficiency of the basic matrix quantization scheme [Tsa85] for joint quantization of adjacent speech parameter vectors is improved in two ways. First, a low-complexity prediction scheme is used for exploiting the correlations between successive parameter values or vectors. Second, the quantizer structure is enhanced by allowing the quantizer to operate in two or more modes, with separate codebooks and predictors for each mode. Following the safety-net idea presented in [Eri99], a memoryless mode is included in the quantizer structure. This mainly enhances the performance in noisy channels but also helps in avoiding coarse quantization errors in applications where the channel can be assumed to be error-free. Due to the flexibility of the proposed approach, it can be used, e.g., for realizing all the quantizers of the VLBR codec. In addition to presenting a quantizer structure providing high quantization accuracy, this section also introduces an efficient algorithm for training quantizers having the proposed structure.

3.3.1 Proposed quantizer structure

The first step in the proposed approach is to utilize the idea of joint quantization of J adjacent parameter vectors (for scalar parameters the vector dimension k equals unity). This is achieved using the matrix quantization approach introduced in [Tsa85]. Let $\mathbf{x}_l$ be the lth k-dimensional vector to be quantized. Now, the input matrices are of the form

$$\mathbf{X}_n = [\mathbf{x}_{nJ+1} \dots \mathbf{x}_{nJ+J}], \tag{3.8}$$

where the index n is a non-negative integer. To simplify the implementation, the matrix quantizer can be realized using an kJ-dimensional vector quantizer for coding the vectors $\mathrm{col}(\mathbf{X}_n)$, where col is a column operator that column-wise

reshapes a matrix into a column vector. Furthermore, to make the proposed quantizer structure applicable to all the quantization tasks in the VLBR codec, the selection of $J = 1$ is allowed to enable the compression of isolated parameter values or vectors. Nevertheless, to show the full benefits, this section focuses on cases where $J > 1$.

Low-complexity prediction scheme

A typical approach in predictive vector quantization [Cup85] discussed in Section 2.3.3 is to utilize the auto-regressive and/or the moving average prediction with either scalar predictors or diagonal predictor matrices. Consequently, the lth prediction vector, $\widetilde{\mathbf{x}}_l$, can be computed according to Equation (2.18). The actual quantization is performed on the prediction residual vector $\mathbf{e}_l = \mathbf{x}_l - \tilde{\mathbf{x}}_l$. For notational convenience, the rest of this subsection discusses the case of the first-order AR predictor (with $m_A = 1$ and $m_B = 0$). However, the same principles are applicable to other predictor types as well. When vector quantization of $\mathbf{x}_l$ is extended to quantization of col($\mathbf{X}_n$), the straightforward extension would be to use either scalar predictors or diagonal predictor matrices of size kJ-by-kJ. However, the prediction achieved with these approaches is rather weak because the time difference between the source of prediction and the target vector is rather large. Better predictions would be achieved using full-matrix predictors of size kJ-by-kJ but this approach would result in a significant increase in the computational complexity, especially at the decoder.

An efficient prediction scheme can be formulated by taking into account the fact that the kJ-dimensional block to be coded consists of adjacent values of a k-dimensional speech parameter. The correlation between successive values is typically quite strong, especially during voiced sounds, but the correlation decreases when the time distance between the values is higher. Thus, intuitively, an efficient predictor can be obtained by designing a predictor that emulates the predictor operating with the k-dimensional vectors. Let $\mathbf{A}_1$ be the diagonal predictor matrix optimized for the k-dimensional vectors. Provided that the same predictor is used for the J vectors inside the longer vector, the k-dimensional predictions can be computed in a closed-loop manner as

$$\tilde{\mathbf{x}}_l = \mathbf{A}_1(\hat{\mathbf{e}}_{l-1} + \tilde{\mathbf{x}}_{l-1}). \tag{3.9}$$

Direct usage of this equation would result in excessive computations since the prediction inside the kJ-dimensional vector depends on the quantization of the same vector. Thus, the prediction should be recomputed for all the code vectors in the codebook. To derive a more useful form, Equation (3.9) can be rewritten for the nth joint vector as

$$\tilde{\mathbf{x}}_{nJ+q} = \mathbf{f}_{nJ+q} + \mathbf{g}_{nJ+q}, \tag{3.10}$$

where q runs from 1 to J and

$$\mathbf{f}_{nJ+q} = \sum_{j=q}^{1} \mathbf{A}_1 \hat{\mathbf{x}}_{nJ+q-j} + \sum_{j=1}^{\min(q-1,1)} \mathbf{A}_1 \mathbf{f}_{nJ+q-j}, \tag{3.11}$$

and

$$\mathbf{g}_{nJ+q} = \mathbf{0}_k + \sum_{j=1}^{\min(q-1,1)} \mathbf{A}_1 (\hat{\mathbf{e}}_{nJ+q-j} + \mathbf{g}_{nJ+q-j}), \tag{3.12}$$

where $\mathbf{0}_k$ is a k-dimensional vector containing zeros. Because now the intra-matrix prediction in Equation (3.12) can be directly included in the codebook, the prediction reduces to the inter-matrix prediction in Equation (3.11). Based on the above discussion, the computation of the prediction residual to be quantized, $\mathbf{E}_n = [\mathbf{e}_{nJ+1} \dots \mathbf{e}_{nJ+J}]$, can be reformulated for the nth joint vector as

$$\text{col}(\mathbf{E}_n) = \text{col}(\mathbf{X}_n) - \mathbf{A}\text{col}(\hat{\mathbf{X}}_{n-1}), \tag{3.13}$$

where col is again the column operator that stacks the columns of a matrix into a column vector and the kJ-by-kJ predictor matrix $\mathbf{A}$ has the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \dots & \mathbf{0} & \mathbf{A}_1 \\ \mathbf{0} & \ddots & \vdots & \mathbf{A}_1^2 \\ \vdots & \ddots & \mathbf{0} & \vdots \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{A}_1^J \end{bmatrix}, \tag{3.14}$$

where $\mathbf{0}$ denotes a k-by-k matrix containing zeros. In the above equation, the raise to the power 1...J is computed element-wise. This form allows very efficient predictor optimization since only the k diagonal elements have to be optimized. It should be noted, however, that it is also possible to optimize all kJ non-zero elements in $\mathbf{A}$ to achieve better prediction accuracy especially when designing a predictive quantizer for transitions. Even in this case, the predictor optimization is still computationally efficient when compared to optimization of a full kJ-by-kJ predictor matrix.

Multi-mode operation

There are some inherent disadvantages in predictive vector quantization. First, the predictor cannot be optimal for the whole data to be quantized if the statistical properties of the data change over time. Because of this fact, relatively poor average prediction accuracy is often achieved in practical speech coding applications. Second, the effects of bit errors may propagate to several vectors. While there is no need to prepare for bit errors in the applications discussed in this thesis, in most applications this is a real concern that needs to be addressed to make the

method proposed in this section applicable more widely. The situation is especially problematic in AR prediction since there is no mechanism to limit the bit error propagation.

In [Eri99], the problems with predictive quantization were addressed using a safety-net approach in which part of the vectors are coded with a memoryless quantizer. This effectively reduces the propagation of bit errors as the occasional use of the memoryless quantizer limits the effects of single bit errors to short segments. In addition, the safety-net approach allows the predictor to be optimized only for the parts that contain significant correlation between successive vectors.

In the proposed quantizer structure, a similar technique is used. However, the number of quantizer modes is not limited to two; the only limitation is that one of the modes should be a memoryless mode. Furthermore, the mode selection step can be implemented in two ways: either the mode resulting in the smallest distortion can be selected as in [Eri99] or a mode information determined prior to the actual quantization, e.g., a voicing-based mode information, can be used. The latter approach has a lower encoding complexity whereas the former selection technique always yields as good or better results in terms of quantization accuracy, at the expense of spending an additional bit or bits for storing the mode selection. The selection method can be taken into account in the training phase as will be discussed in the following subsection.

3.3.2 Quantizer training

Training of a predictive multi-mode quantizer is a challenging task because of the complex relationships between the different modes, and between the codebooks and the predictors. Although direct joint optimization is computationally impractical and there are no known solutions for finding the global optimum, good results can be achieved using the fairly simple design algorithm introduced in this section. The design procedure is partly based on the asymptotic closed-loop design technique presented in [Kha01b].

Initialization

The initialization step begins with establishing a training data including initial mode information. If the mode information is readily available, for example from a voicing classifier, this information should naturally be used for the initial modes. In other cases, there are no strict rules on how the initial modes should be determined. One solution is to divide the training data into sets of approximately equal sizes based on the information obtained by measuring the difference of each vector relative to its predecessor. For example, in the case of a two-mode quantizer, the memoryless mode could be selected for vectors with a difference larger than the median of the measured differences, while the rest of the vectors could be classified into the predictive mode.

After the training data and the initial modes have been determined, the initialization continues with obtaining a set of vectors that will be used as the basis for the predictions during the first iteration of the actual training algorithm. In the case of first-order AR prediction, the set to be computed is the initial set of reconstructed vectors, $\{\hat{\mathbf{X}}\}$. In practice, the reconstructed vectors can be computed by quantizing the training data with the initial codebooks and predictors designed using the basic open-loop design technique discussed in [Gra84]. Each initial codebook and predictor should be designed using only the training data classified into that mode. An exception to this rule is that the whole training data can be used for training the initial codebook for the memoryless mode.

Training algorithm

The most important steps in the actual training algorithm can be summarized as follows:
Step 1. Compute an initial set of prediction residuals using Equation (3.13). Base the predictions on the fixed set of reconstructed vectors computed during initialization.
Step 2. Optimize new predictors for all predictive modes by minimizing, within each mode, the distortion

$$D_m = \sum_n d\left(\text{col}\left(\mathbf{E}_n\right), \mathcal{Q}\left(\text{col}\left(\mathbf{E}_n\right)\right)\right), \tag{3.15}$$

where D_m denotes the distortion for the mth mode, $d\left(\cdot,\cdot\right)$ is a vector-based distortion measure, $\mathcal{Q}\left(\cdot\right)$ denotes quantization with the latest codebook, and the sum is computed over all the vectors in the current mode.
Step 3. Compute an updated set of prediction residuals using Equation (3.13). Employ the latest predictor within each mode and base the predictions on the fixed reconstructed vectors. Using these prediction residuals as the training data, design new codebooks for all the modes. For each predictive mode, use only the training vectors within that mode.
Step 4. Repeat the two distortion lowering steps (Step 2 and Step 3) until a predetermined convergence criterion is satisfied.
Step 5. Recalculate the set of reconstructed vectors as

$$\text{col}\left(\hat{\mathbf{X}}_n\right) = \mathbf{A}\text{col}\left(\hat{\mathbf{X}}_{n-1}^{\text{old}}\right) + \mathcal{Q}\left(\text{col}\left(\mathbf{E}_n\right)\right), \tag{3.16}$$

where $\mathbf{A}$ is the latest predictor for the current mode. The prediction is based on the old, non-updated set of reconstructed vectors and the latest codebooks are used for coding exactly the same data used for training them.
Step 6. If the mode selection is to be performed during quantization, encode the entire training data using the most recent quantizer and update the mode information accordingly. Otherwise, proceed to Step 7.

Step 7. Check whether the termination criterion for the algorithm is satisfied. If not, continue from Step 2 with the new fixed set of reconstructed vectors. Otherwise, terminate the training procedure.

Remarks on the training procedure

To allow simple analytic optimization in Step 2, it can be assumed that the changes in the predictors do not affect the quantization in Equation (3.15). Now, a set of equations can be formed, separately for each mode, by taking partial derivatives of Equation (3.15) with respect to the predictor coefficients to be optimized and by setting them to zero. This set of linear equations can then be solved using Gaussian elimination. As discussed in this section, the predictor optimization step can be further simplified by optimizing only the k non-zero coefficients of the diagonal predictor $\mathbf{A}_1$ and by constructing the predictor matrix $\mathbf{A}$ according to Equation (3.14).

The codebook optimization is performed on a set of prediction residuals in a memoryless fashion. Thus, practically any optimization algorithm designed for training memoryless quantizers, such as the generalized Lloyd algorithm described in Section 2.3.1, can directly be applied. Furthermore, if the codebook size or the search complexity would be too high with a full-search codebook, structurally constrained codebooks, e.g., multistage codebooks described in Section 2.3.2, can be used.

If the quantizer is to be used in noisy channels, it is beneficial to add some index assignment procedure as an additional step after the training algorithm has terminated. This optional step ensures more robust performance when bit errors are encountered. Practically any index assignment algorithm can be used to supplement the proposed training procedure. For example, a binary switching algorithm for achieving a locally optimal index assignment can be found in [Zeg90].

The proposed training procedure inherits the design stability of the asymptotic closed-loop algorithm because the predictions are always based on the reconstructed vectors from the previous iteration. The original ACL technique achieves monotonic improvement throughout the training process under the assumption that smaller prediction errors lead to smaller quantization errors and vice versa [Kha01b]. The same is true for the proposed training algorithm, with the additional assumption that the new mode selections do not disturb the process. Although neither of the assumptions is strictly valid in all situations, the proposed training technique is very stable and provides significant improvements over the basic open-loop design approach.

Finally, it should be noted that even though the discussions and the experiments provided in this section focus on the case of auto-regressive prediction, both the proposed quantization approach and the proposed training algorithm can be applied to cases with other types of prediction, such as MA or ARMA prediction, as well. The same is true for the original ACL technique, despite the fact that

all the early publications on the ACL approach ([Ros98b], [Kha01b], [Kha00], [Kha01c], and [Kha01a]) only dealt with AR prediction. In particular, the author demonstrated in [Nur03a] that even though the basic open-loop design technique [Ger92] cannot be directly used in the case of MA prediction, the asymptotic closed-loop design methodology can be successfully applied to the training of MA predictive quantizers.

3.3.3 Performance evaluation

To demonstrate the performance advantage gained by using the proposed quantization scheme, it was tested in a practical quantization task that is present not only in the VLBR codec discussed in this thesis but also in the vast majority of other parametric speech coders, i.e., in the quantization of the linear prediction coefficients. In the test set-up, 10th-order linear prediction analysis was performed at 20 ms intervals on speech sentences randomly selected from a multilingual database sampled at 8 kHz. The coefficients were converted into the line spectral frequency representation to obtain 10-dimensional parameter vectors. A data set of $250\,000$ vectors was collected for the training phase while a distinct set of $150\,000$ vectors was used for testing. The weighted squared error,

$$d\left(\mathbf{x}_n, \hat{\mathbf{x}}_n\right) = \left(\mathbf{x}_n - \hat{\mathbf{x}}_n\right)^\top \mathbf{W}_n \left(\mathbf{x}_n - \hat{\mathbf{x}}_n\right), \tag{3.17}$$

was used as the distortion measure both in training and in quantization. Diagonal weighting matrices were used and the weights were derived using the weighting scheme presented in [Pal93], i.e., the distortion of each LSF was weighted proportionally to the value of the spectrum at the corresponding frequency. More precisely, the weight for the jth LSF vector element was computed using

$$w_j = \left| H(e^{i2\pi F_j/F_s}) \right|^{0.3}, \tag{3.18}$$

where F_j denotes the frequency of the jth LSF and F_s is the sampling frequency. In addition, as described in [Pal93], the distortions of the ninth and the tenth LSF are multiplied by the fixed weights of 0.64 and 0.16, respectively.

Quantizer design

The predictive multi-mode quantizer was designed for joint quantization of two LSF vectors. Two modes were included: a memoryless mode and a mode with a first order AR predictor. The codebooks in both modes were 39-bit multistage codebooks with 7 stages of sizes 6, 6, 6, 6, 6, 6, and 3 bits. The quantizer design was performed using the procedure described in Section 3.3.2. In the quantizer, the mode was selected by finding the mode resulting in the smallest distortion and thus new mode selections were also allowed during training. The simultaneous joint design algorithm [LeB93] discussed in Section 2.3.2 was used in the

Table 3.4: Performance of the proposed vector-predictive multi-mode quantizer, a conventional matrix quantizer and a vector quantizer in an error-free environment. All the quantizers operate at the fixed bitrate of 20 bits/vector.

	Mean SD	Max SD	WMSE
Multi-mode	0.9613	5.3223	4.66×10^{-3}
Matrix quantizer	1.0975	5.6322	5.94×10^{-3}
Vector quantizer	1.2581	5.8965	7.79×10^{-3}

codebook optimization step. As an additional step, the codebook indices were reordered using the index assignment algorithm introduced in [Zeg90].

In addition to the proposed quantizer, a basic matrix quantizer and a basic vector quantizer operating at the same bitrate were also designed to allow comparisons. The matrix quantizer was designed for joint quantization of two LSF vectors and was realized using a 40-bit multistage structure with 7 stages (6, 6, 6, 6, 6, 6, and 4 bits). The 20-bit vector quantizer was trained for quantization of single vectors and the codebook contained 4 stages (6, 6, 5, and 3 bits). The training for both quantizers was performed using the simultaneous joint design algorithm. The index assignment algorithm [Zeg90] was used as the last design step for both quantizers.

Test results

The three quantizers, all operating at the bitrate of 1.0 kbit/s, were first tested in an error-free environment. *Spectral distortion* (SD) was used as the primary distortion metric in the evaluation. SD is defined in dB as

$$SD^2 = \frac{1}{F_u - F_l} \int_{F_l}^{F_u} \left(20 \log_{10} \frac{\left|H(e^{i2\pi F/F_s})\right|}{\left|\hat{H}(e^{i2\pi F/F_s})\right|} \right)^2 dF, \qquad (3.19)$$

where $H(e^{i\omega})$ and $\hat{H}(e^{i\omega})$ denote the frequency responses of the original and quantized versions of the synthesis filter $H(z) = 1/A(z)$, respectively, while F_l and F_u define the lower and upper frequency limits of the integration, and F_s is the sampling frequency. In addition, the *weighted mean squared error* (WMSE) was also computed using Equation (3.17). The spectral distortion was computed over the frequency band from 0 Hz to 3000 Hz because of the weighting scheme used in the test.

The results of the first test are presented in Table 3.4. The proposed multi-mode quantization approach clearly yielded the best quantization accuracy with both distortion metrics. Furthermore, as expected, the matrix quantizer that used joint quantization of two vectors outperformed the conventional vector quantizer.

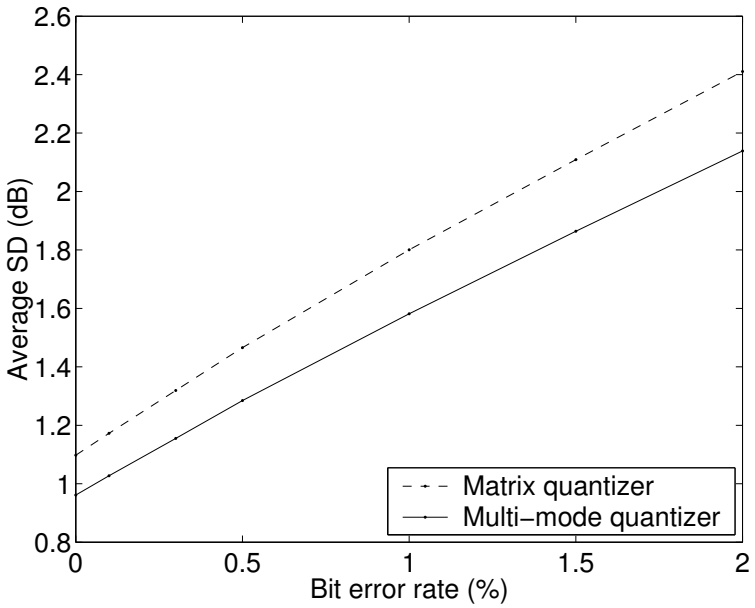

Figure 3.6: Average spectral distortion obtained with the proposed multi-mode quantizer and with the basic matrix quantizer at different bit error rates. Both quantizers operated at the bitrate of 20 bits/vector. (From [Nur03b].)

The differences in the performance of the quantizers were quite clear with both metrics.

In the second test, the two quantizers that yielded the best results in the first test (the proposed predictive multi-mode quantizer and the matrix quantizer) were evaluated in the presence of bit errors. Bit error rates from 0% to 2% were used in the evaluation. The average spectral distortions measured during this test are depicted in Figure 3.6. As can be seen from the figure, the proposed approach maintained the performance advantage in erroneous transmission conditions, despite the fact that the quantizer included an AR predictive mode. Even though there are practically no bit errors in the application scenarios discussed in this thesis, this result is important for highlighting the general usefulness of the proposed approach.

3.4 Enhanced dynamic codebook reordering

As discussed in Section 2.3.1, vector quantization is one of the most efficient and powerful tools in the field of data compression, and in fact, it can be shown that no memoryless quantization technique can outperform vector quantization as explained in [Ger92]. When the quantizer is allowed to utilize memory, the compression performance can be further improved using many different techniques.

Examples of such techniques include predictive vector quantization [Cup85] discussed in Section 2.3.3, finite-state vector quantization [Fos85], and *dynamic codebook reordering* (DCR) [DeN96] [Sri98] [Kri05].

In dynamic codebook reordering, when implemented as in [Sri98] and in [Kri05], the main idea is to dynamically reorder the codebook at each compression pass based on the selected code vector. Without reordering, the probabilities for the different indices are quite similar, making lossless coding relatively inefficient. The reordering exploits the correlation between successive vectors to make the probability distribution less flat, which in turn reduces the entropy and makes the lossless compression of the indices more efficient. The conventional dynamic codebook reordering approach works usually well in the case of simple memoryless vector quantization, provided that the synchrony between the encoder and the decoder can be assured, but the technique becomes less efficient when applied to more advanced quantizer structures.

In the work presented in this section, originally presented in [Nur06a], the aim is to improve the performance of the dynamic codebook reordering technique in advanced quantizer structures. Here, the term advanced structure refers to cases where the structure also includes other parts in addition to a single full-search vector quantizer. Multistage vector quantization [Jua82] discussed in Section 2.3.2 offers a typical example of such a structure. The proposed enhancements are described mainly from the viewpoint of MSVQ and *predictive multistage vector quantization* (PMSVQ) but the same ideas can be extended to other quantizer structures as well. The original target application for the proposed technique was the speech codec introduced in this chapter but the method can also be used in other data compression related applications, especially in systems that operate on error-free bit streams. To obtain the benefits of the proposed approach, the lossless compression of the quantizer indices needs to be performed separately for the different quantizers.

3.4.1 Conventional dynamic codebook reordering

To lay the ground for the description of the proposed method, the basic principles and the practical usage of the conventional dynamic codebook reordering technique are discussed first. The first part of this subsection deals with general background information while the second part provides a basic description of the conventional DCR approach. The problems related to the use of DCR with advanced quantizer structures are discussed in the last part of this subsection.

Background information

Let $\mathbf{x}$ be the input vector to be quantized and let $\mathbf{C}$ be the codebook that is available both at the encoder and at the decoder. As discussed in Section 2.3.1, in conventional vector quantization the encoder finds from the codebook $\mathbf{C}$ the code

vector that best represents the input vector, i.e., the code vector that is closest to the vector $\mathbf{x}$ according to some distortion measure (e.g., weighted squared error). The compressed representation is then given by a digital symbol h that represents the index of the code vector that was selected by the encoder. During decoding, the dequantized version of the vector can be reconstructed simply by taking from $\mathbf{C}$ the code vector that corresponds to the received codebook index.

Since the number of code vectors in the codebook is always limited, the symbol h belongs to a limited set of symbols, $h \in \zeta$. If the number of code vectors, or equivalently the number of symbols in set ζ, is denoted as N, the number of bits needed to represent a single symbol h can be computed simply as $\lceil \log_2 (N) \rceil$. Through the use of lossless compression techniques, it may be possible to decrease the average number of bits per symbol and thus to achieve lower rate for the same distortion level. If it is possible to know or to estimate the probability $p(h)$ for every $h \in \zeta$, the theoretical entropy bound for lossless compression efficiency can be computed as

$$\mathsf{H} = -\sum_{h} p(h) \log_2 (p(h)), \tag{3.20}$$

where the sum is computed over all symbols $h \in \zeta$. Without any reordering, the probabilities for the different symbols are usually quite similar, resulting in high entropy. Consequently, in this case, the lossless compression of the indices can only bring very limited benefits.

Description of the technique

The conventional dynamic codebook reordering technique was originally designed to facilitate the compression of the codebook indices [DeN96]. It reduces the entropy of the index data by exploiting the correlation between adjacent vectors. Following the detailed description presented in [Kri05], the encoding at time index n is performed as follows. First, the codebook search is performed by finding the code vector $\mathbf{c}_{j,n}$ that best represents the input vector $\mathbf{x}_n$ similarly as described earlier in this subsection. Then, the symbol $h_n = \psi(j, n)$ corresponding to the code vector $\mathbf{c}_{j,n}$ is stored or transmitted to the decoder. $\psi(j, n)$ is a simple dynamic index map [Kri05] that relates the physical code vector index to its reordered index at time instant n. The dynamic index map is initialized as $\psi(l, 0) = l$ for $l = 0, 1, \ldots, N-1$, and it is updated during each encoding pass after storing or transmitting the selected index. The updated order is obtained by sorting the values

$$\delta(l, n) = d(\mathbf{c}_l, \mathbf{c}_{j,n}) \quad \text{for } l = 0, 1, \ldots, N-1, \tag{3.21}$$

in increasing order so that

$$\delta(l_0, n) \leq \delta(l_1, n) \leq \cdots \leq \delta(l_{N-1}, n), \tag{3.22}$$

where $l_0, l_1, \ldots, l_{N-1}$ belong to the set of integers between 0 and $N-1$. The distortion measure or dissimilarity measure $d(\cdot)$ in Equation (3.21) can be chosen freely as long as the computation can be performed both at the encoder and at the decoder. After the reordering, the updated dynamic index map for the next encoding pass can be obtained simply as

$$\psi(j, n+1) = l_j \quad \text{for } l = 0, 1, \ldots, N-1. \tag{3.23}$$

The decoder performs the reordering similarly as the encoder. The reordering is carried out during each decompression pass after accessing the codebook. The mapping between the received symbol and the physical code vector index is obtained using an inverse dynamic index map $j = \psi^{-1}(h_n, n)$ [Kri05]. As all the information is readily available both at the encoder and at the decoder, the dynamic index maps can be kept in synchrony without any side information, provided that the channel used for conveying the indices is error-free. Since usually the data to be compressed is not completely uncorrelated, the dynamic reordering makes the probability distribution less flat, which in turn reduces the entropy.

Usage in advanced quantizer structures

In multistage vector quantization, the input vector is quantized in two or more additive stages. The objective in encoding is to find a code vector combination, in other words a sum of the selected code vectors at different stages, that minimizes the resulting distortion. The selections are performed using some search algorithm. The simplest technique is to use sequential search, i.e., the selection is first performed for the first stage and then for the second stage, and so on. A better approach is to utilize joint selection, using e.g. the M-L tree search algorithm [LeB93]. According to the descriptions given in [Kri05], the correct way to use the DCR algorithm in the case of MSVQ is to apply it separately to each stage codebook. Each reordering is performed in the order of increasing dissimilarity to the code vector selected from that codebook. This approach is very simple and straightforward but the improvements gained over the case without reordering are usually rather small. This problem was discussed in [Kri05], where the authors correctly stated that after the first stage the vectors are more decorrelated, which makes the technique less efficient. The proposed enhancements described in this section allow clearly better reordering in the case of multistage vector quantization, which significantly improves the performance especially at latter stages.

In predictive vector quantization, a prediction of the input vector is calculated using information about a finite number of previously quantized vectors, using, e.g., auto-regressive prediction and/or moving average prediction. Then, the prediction error vector (also referred to as the prediction residual) is quantized instead of the original input vector. Finally, in decoding, the output is computed by adding together the prediction and the quantized prediction error. Since both PVQ and

DCR exploit the correlation between adjacent vectors, the benefit achieved using both of them at the same time cannot be expected to be very high. Nevertheless, it is possible to apply dynamic codebook reordering on the indices of PVQ, using a direct approach similar to the one proposed for MSVQ in [Kri05]. In this straightforward application of the conventional DCR technique, the reordering is performed simply using the dissimilarity to the selected codebook entry as the sorting criterion.

3.4.2 Enhanced dynamic codebook reordering

The proposed enhanced dynamic codebook reordering approach improves the efficiency of DCR by taking into account the whole quantizer structure and all the pieces of information that are available from the other parts of the structure. To maximize the amount of available information, each reordering is delayed to the last possible moment, in contrast to the conventional DCR where the reordering is performed immediately after each quantization. In this section, the enhanced approach is presented from the viewpoint of MSVQ and predictive quantization. However, using the same ideas, it is possible to generalize the proposed approach to other quantizer structures as well. For example, it would be possible to extend the technique for the predictive multi-mode matrix quantizer structure presented in Section 3.3 and in [Nur03b].

In the proposed approach, the dynamic index map for each stage is first initialized similarly as described in Section 3.4.1. Then, the encoding at time instant n is performed as follows:
Step 1. Prediction (optional); As before in this thesis, let $\tilde{\mathbf{x}}_n$ denote the predicted vector and $\mathbf{e}_n$ denote the prediction residual vector, i.e., $\mathbf{e}_n = \mathbf{x}_n - \tilde{\mathbf{x}}_n$.
Step 2. Perform the codebook search for the vector $\mathbf{e}_n$ for all the stages of the K-stage quantizer structure. Let $\mathbf{c}_{j,n}^{(q)}$ denote the code vector selected at stage q for $q = 1, 2, \ldots, K$.
Step 3. Set $q = 1$.
Step 4. Dynamic codebook reordering for stage q using for $l = 0, 1, ..., N^{(q)} - 1$ the measure

$$\delta^{(q)}(l, n) = d\left(\sum_{m=1}^{q-1} \mathbf{c}_{j,n}^{(m)} + \mathbf{c}_{l}^{(q)} + \tilde{\mathbf{x}}_n, \hat{\mathbf{x}}_{n-1}\right), \tag{3.24}$$

where $\hat{\mathbf{x}}_{n-1}$ can be computed using

$$\hat{\mathbf{x}}_n = \sum_{m=1}^{K} \mathbf{c}_{j,n}^{(m)} + \tilde{\mathbf{x}}_n, \tag{3.25}$$

together with the sorting strategy similar to that given in Equations (3.21), (3.22), and (3.23). The outcome is the updated dynamic index map $\psi^{(q)}\left(j^{(q)}, n\right)$ for

stage q. Here, $j^{(q)}$ denotes the physical index of the code vector selected at stage q.
Step 5. Dynamic index mapping for stage q: $h_n^{(q)} = \psi^{(q)}\left(j^{(q)}, n\right)$.
Step 6. Transmit or store the digital symbol $h_n^{(q)}$. The lossless compression can be performed during this step or later jointly for several symbols at the same time.
Step 7. Set $q = q + 1$. If $q \leq K$, go to Step 4. Otherwise, exit.
If the lossless compression is not carried out in Step 6, it must be handled later, e.g. after processing the whole vector or after processing a block of vectors, or even after processing all the input vectors. The optimal timing of the lossless compression depends on the selected compression algorithm. For example, with Huffman coding [Huf52] the compression can be done at Step 6 but with arithmetic coding [Ris76] the compression should be done for a larger block of symbols at the same time.

The operation of the decoder is modified in a similar manner. The decoding at time instant n is performed as follows:
Step 1. Prediction (optional); Set $q = 1$.
Step 2. Dynamic codebook reordering and dynamic index map update for stage q similarly as in Step 4 of encoding.
Step 3. Inverse mapping for stage q: $j^{(q)} = \psi^{-1}\left(h_n^{(q)}, n\right)$.
Step 4. Set $q = q + 1$. If $q \leq K$, go to Step 2. Otherwise, continue to Step 5.
Step 5. Reconstruction: compute $\hat{\mathbf{x}}_n$ using Equation (3.25). Exit.
As in the case of conventional DCR, all the necessary information is readily available both at the encoder and at the decoder. Thus, there is no need to include any side information, unless the channel is erroneous. However, an occasional reset of the dynamic index maps can be included to allow more convenient access to the bit stream.

The above descriptions are applicable to both MSVQ and PVQ, and to the combination of them. If the quantizer structure does not include prediction, $\tilde{\mathbf{x}}_n$ can be set to zero vector for all n. If there is only one stage ($K = 1$) and no prediction, the proposed technique reduces to the conventional DCR approach (with different timing of the reordering). As will be shown in Section 3.4.3, the proposed modifications make favorable changes to the probability distributions, which in turn reduces the entropy and significantly improves the compression efficiency. This performance advantage can be achieved at a very low cost: with careful implementation, the additional complexity compared to the conventional DCR is only one add operation per stage.

3.4.3 Experimental results

To demonstrate the benefits of the proposed approach, some experiments were carried out in the context of a speech compression related task, more specifically in the quantization of the line spectral frequency (LSF) vectors derived from the

Table 3.5: Theoretical bitrates achievable using a 4-stage MSVQ for LSFs (originally 2200 bps).

	No reorder	DCR	Proposed
Stage 1	566	368	325
Stage 2	582	531	407
Stage 3	497	482	397
Stage 4	497	491	426
Total	2142	1873	1555

Table 3.6: Theoretical bitrates achievable using a 3-stage MSVQ for LSFs (originally 2200 bps).

	No reorder	DCR	Proposed
Stage 1	774	509	457
Stage 2	687	642	519
Stage 3	694	680	593
Total	2155	1830	1570

time-varying linear prediction coefficients. Since many speech coders utilize linear prediction, this quantization task is one of the most common tasks in speech coding, and hence a good example application for the proposed technique. In the experiments, the training set contained $180\,000$ 10-dimensional LSF vectors and the test set included a separate set of another $180\,000$ vectors, all estimated at 10 ms intervals from a speech database containing 8-kHz speech signals. The speech database consisted of active speech containing full sentences from many female and male speakers in several languages.

Table 3.5 illustrates the results from an experiment involving the application of the proposed approach in MSVQ. The studied quantizer structure included 4

Table 3.7: Theoretical bitrates achievable using a 5-stage MSVQ for LSFs (originally 2200 bps).

	No reorder	DCR	Proposed
Stage 1	464	300	266
Stage 2	482	440	328
Stage 3	397	379	301
Stage 4	399	390	327
Stage 5	400	395	342
Total	2142	1904	1564

Table 3.8: Theoretical bitrates achievable using a 5-stage PMSVQ for LSFs (originally 2200 bps).

	No reorder	DCR	Proposed
Stage 1	448	431	419
Stage 2	476	471	434
Stage 3	398	398	376
Stage 4	399	398	383
Stage 5	400	400	387
Total	2120	2099	1999

stages with 64, 64, 32 and 32 code vectors in their codebooks. The training was performed using the multistage VQ simultaneous joint design algorithm [LeB93]. Weighted squared error was used both as the dissimilarity measure in the reordering and as the distortion measure in the M-L tree search. The tree search was run so that 8 best paths were stored at each stage (i.e., the parameter M in [LeB93] was set to 8). The weights were computed similarly as in [Pal93] in such a manner that during the codebook search the weights were derived based on the corresponding unquantized LSF vector whereas in the reordering the weights were computed based on the latest quantized LSF vector. Table 3.5 contains a comparison of the theoretical bit rates achievable using the same codebooks in three cases: using lossless compression without reordering, using conventional DCR, and using the proposed enhanced approach. As can be seen from these theoretical bounds obtained through empirical entropy (Equation (3.20)), the proposed approach clearly outperforms conventional DCR. The reason for the improved performance can be seen from Figure 3.7 that depicts the empirical index probabilities for the fourth stage, using the test data and the three different approaches (the same as in Table 3.5). The proposed technique produces the least flat distribution and consequently the lowest entropy. The difference compared to the conventional DCR is significant, especially in the latter stages.

To illustrate the effect that the number of stages has on the performance, results from two alternative designs are presented in Table 3.6 and Table 3.7. More specifically, Table 3.6 contains results obtained using a 3-stage quantizer structure. The performance advantage gained using the proposed approach is a bit smaller than in the case of the 4-stage quantizer but the gap to the conventional DCR is still a clear 260 bps. Table 3.7 depicts the theoretical bitrates achievable in the case of a 5-stage quantizer. Here, the increased number of stages makes the difference between the conventional DCR and the enhanced DCR larger (now 340 bps).

In Table 3.8, the 5-stage quantizer structure of the previous experiment was complemented with a predictor. The prediction was performed using a first or-

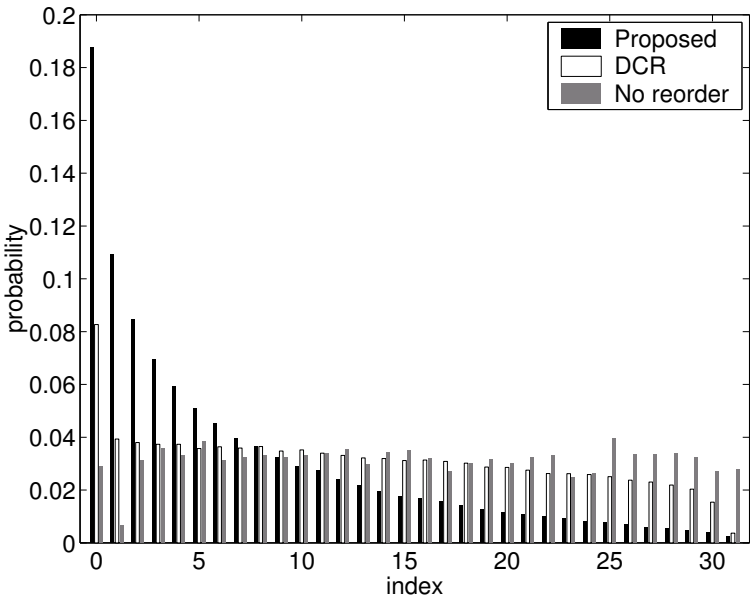

Figure 3.7: Index probabilities at the last stage of the 4-stage quantizer. (From [Nur06a].)

der auto-regressive predictor with a diagonal predictor matrix. The predictor and the codebooks were jointly optimized using a combination of the multistage VQ simultaneous joint design algorithm [LeB93] and the asymptotic closed loop design approach [Kha01b]. The PMSVQ structure contains properties that make it very challenging from the viewpoint of dynamic codebook reordering. Nevertheless, the proposed approach still clearly outperforms the conventional DCR but the improvement of 100 bps is smaller than in the structure that did not contain prediction (see Table 3.7).

3.5 Improvement of coding efficiency via preprocessing

The exploitation of the psychoacoustic principles discussed briefly in Section 2.1.2 can ultimately lead a speech coding process into a perceptually optimal state in which a considerable reduction in the bitrate can be obtained without impairing the speech quality. Especially at the very low bitrates discussed in this thesis, it is very advantageous to avoid spending scarce resources for coding perceptually irrelevant information. The work presented in this section [Läh03b] is aimed at processing narrowband speech signals in such a manner that the modified signal contains less perceptually unimportant information than the original, yet keeping the speech quality essentially unaltered.

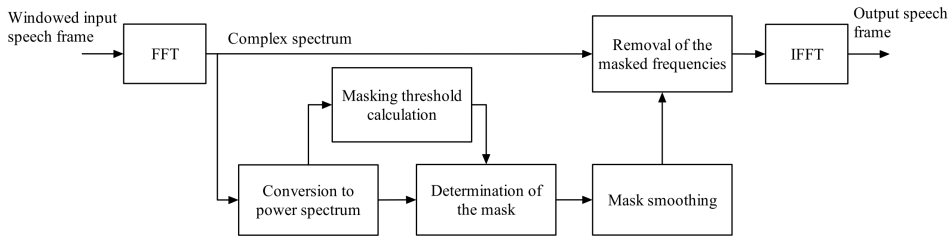

Figure 3.8: Block diagram of the preprocessing function. (From [Läh03b].)

The basic idea behind the inherent existence of perceptual irrelevancy is the masking phenomenon that is present in all real-world auditory signals. In this work, the determination of the irrelevant components is based on a psychoacoustic model for audio signals proposed by Johnston [Joh88] that offers a simple approach for modeling simultaneous masking. The performance of the proposed preprocessing approach is evaluated objectively and using listening tests, both on its own and when used as a separate preprocessor for standardized speech codecs.

Even though the novel preprocessing method presented in this subsection approaches the problem from a fresh angle, utilizing the results of psychoacoustic research in signal compression is not by any means a new idea. Schroeder reported already in 1979 his method of exploiting the auditory masking effects in speech coders [Sch79] and a part of his work is utilized by Johnston in the masking threshold calculation. Applications of masking models have been reported mainly from the field of audio coding, but some interesting experiments have also been made with speech signals. In the work published in [Luk01a] and [Luk01b], simultaneous masking was incorporated in the linear prediction, resulting in improved perceptual quality of coded speech. Examples of psychoacoustically assisted speech enhancement methods can be found in [Tso97] and [Vir99].

3.5.1 Proposed approach for perceptual irrelevancy removal

A block diagram of the proposed preprocessing method is shown in Figure 3.8. As the first step, each input speech frame is multiplied with a Hamming window and transformed into its frequency domain representation via a fast Fourier transform. The Johnston's model [Joh88], now modified to operate on a sampling rate of 8 kHz instead of the original 32 kHz, is used to determine the masking threshold for each frame of speech. In the experiments presented in this section, the frame length was set to 320 samples (40 ms) and an overlap of 50% between the frames was used, and an FFT of 512 samples was found to provide an adequate frequency resolution for narrowband 8-kHz speech.

After computing FFT, the power spectrum is divided into 18 critical bands according to [Zwi90] and a masking threshold is evaluated for each band. The final masking threshold is determined as the maximum of the calculated threshold

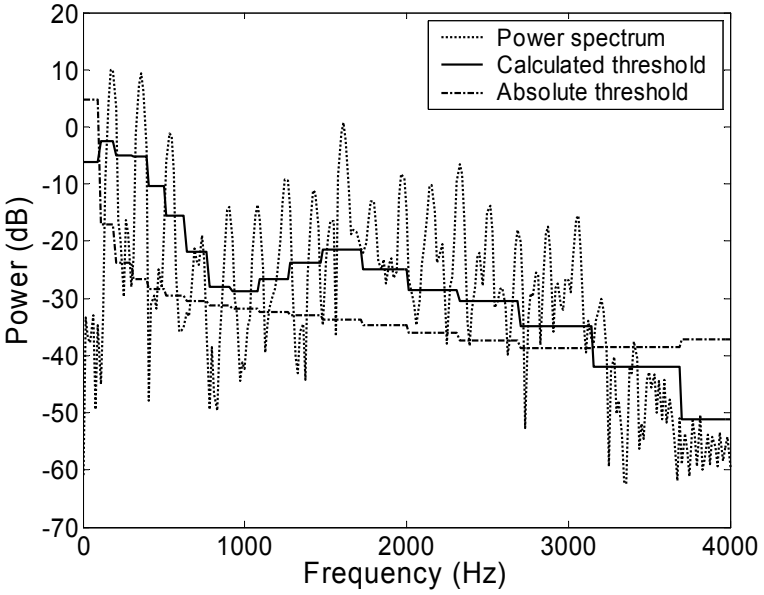

Figure 3.9: Example of masking threshold calculation. (From [Läh03b].)

and the absolute threshold of hearing at each critical band. An example of the thresholds for a female speech frame is shown in Figure 3.9. The masking thresholds of the frequency bands are compared with the power spectrum components to produce a binary mask. The mask is set to a value of zero at those frequencies where the power spectrum is below the masking threshold and a value of one is used elsewhere.

A straightforward means to remove the masked frequencies would be the multiplication of the complex spectrum of the input frame by the mask at each frequency. This corresponds to adaptive filtering of the input speech. However, due to the variation of the filter's frequency response (i.e., the mask) from frame to frame, the direct use of the mask would result in an output speech containing nonharmonic distortion caused by time domain aliasing [Ras99]. Thereby, the mask is smoothed by convolving it with an appropriate window in the frequency domain. This procedure is detailed in [Ras99]. The digital prolate spheroidal window [Ver96] was chosen for this purpose because it is optimal in the sense that it concentrates most of its energy in the mainlobe and attenuates the aliasing components at its sidelobes, leading to a maximized ratio of the desired signal power to the distortion signal power.

After the mask smoothing, the output segment is adaptively filtered and transformed into time domain using the inverse Fourier transform. The overlapping sections are normalized so that the effect of the windowing is cancelled. The computational complexity of the proposed preprocessing technique is affordable

for many applications: approximately 3.4 million floating point operations per second are needed, the FFT and its inverse transform being the most complicated parts.

3.5.2 Performance evaluation in isolation

The performance of the proposed technique was first evaluated in isolation in order to get information about the performance of the preprocessing technique itself, in such a manner that the speech codec does not affect the results. Both objective measurements and a listening test were carried out. Due to the key position of linear prediction in almost all modern speech coding systems, the objective measurements were computed for LP residuals. In particular, the effect that the preprocessing has on the entropy and the energy of the residual signal was studied.

The role of the listening test was to verify that the preprocessing does not cause substantial deterioration of the speech quality. The test was performed using the comparison category rating (CCR) procedure [ITU96]. In the listening test, the settings of the preprocessor were maintained as they were during the objective measurements in which an average of 43% of the spectral components were deemed masked in each frame.

Objective measurements

As the first step of the objective evaluation, LP analysis was performed on both the original and the preprocessed speech and the resulting LP residuals were examined. The LP coefficients were excluded from the tests since it has already been shown in [Luk01b] that the quantization of the LP coefficients calculated from the modified speech signal requires no extra bits compared to the LP coefficients calculated from the original speech.

Using a testing speech signal with the total duration of about 70 minutes, the energies and entropies of the different LP residuals were computed. The energy of the residual calculated from the preprocessed speech was, on average, 20.3% smaller than that of the residual obtained by analyzing the original speech. The decrease in the energy causes modest loudness differences between the original and the preprocessed speech, but this effect can be compensated at the decoder using an additional level control.

The entropy of the residual signal was reduced by 7.2% by using the modified speech instead of the original speech in the LP analysis and filtering. This result, together with the evident energy reduction, confirms that the usage of the preprocessing technique in the front end of a speech coder produces a residual signal that can be compressed more efficiently than the original residual signal.

Table 3.9: Average scores of the reference samples.

Sample	Female 1	Female 2	Male 1	Male 2
Score	-2.50	-2.50	-1.92	-1.75

Table 3.10: Results of the CCR test. The table lists the average scores for the preprocessed speech with respect to the original unprocessed speech $\pm$ the 95% confidence interval.

	No level adjustment	With level adjustment
Female	-0.25 ± 0.27	0.10 ± 0.21
Male	-0.10 ± 0.26	-0.08 ± 0.26
Total	-0.18 ± 0.18	0.01 ± 0.17

Perceptual evaluation

In the objective measurements described above, the preprocessing function was kept at constant settings, adjusted so that the preprocessing would be quite aggressive but the output speech would not suffer from substantial deterioration. This was verified through a CCR listening test. In the CCR test, listeners were presented with pairs of speech samples and for each pair, they were asked to grade the quality of the latter sample with respect to the former. The grading was done using the seven point scale ranging from -3 (much worse) to $+3$ (much better).

Each sample included in the evaluation consisted of a single sentence. The test material consisted of Finnish speech filtered using an intermediate reference system filter. Six female and six male speakers were chosen from a database and one sentence from each speaker was processed using the proposed technique. Furthermore, another version of each test sample was generated using an additional level adjustment in order to compensate for the slight loudness decrement caused by the preprocessing stage.

Four reference sample pairs were also provided by choosing one male and one female sentence and processing them deliberately in such a manner that they had much lower quality than the original sentences. This was done with the preprocessing function using a random mask with 30–50% of the coefficients set to zero. Furthermore, in samples Male 1 and Female 1 (see Table 3.9) the smoothing window was not in use.

Each processed sentence in the test had the corresponding original unprocessed version as the quality reference. The pairs were played in random order through high-quality headphones.

Altogether 24 naive listeners participated in the test. The listeners and the 24 actual test sample pairs were divided into three groups. The four reference pairs were common to all groups. Thus, each sample pair was listened by eight persons

Table 3.11: Segmental SNR (in dB) between the input and output of the AMR codec with original and preprocessed signals. The improvement percentages are also shown.

	female			male		
kb/s	orig	prep	imp-%	orig	prep	imp-%
4.75	3.17	3.45	8.71	2.75	3.01	9.54
5.15	3.31	3.59	8.56	2.91	3.19	9.55
5.9	3.58	3.91	9.25	3.34	3.66	9.78
6.7	3.72	4.07	9.64	3.52	3.88	10.11
7.4	4.21	4.60	9.21	4.43	4.82	8.80
7.95	3.99	4.38	9.72	4.08	4.46	9.37
10.2	4.67	5.10	9.23	5.21	5.70	9.47
12.2	4.85	5.28	8.76	5.52	5.98	8.43

except for the reference pairs that were evaluated by all the listeners. In Table 3.10, the three listener groups are combined and the average grades for the processed samples with and without the additional level adjustment are shown. Table 3.9 presents the average scores given to the four references. Further discussion on the results is provided in Section 3.5.4.

3.5.3 Performance evaluation with speech codecs

In order to test how actual speech codecs perform with the preprocessed speech generated using the proposed approach, another set of tests was carried out. Again, the performance was first evaluated using objective measurements and then in a listening test. The objective part of the evaluation utilized the standardized adaptive multi-rate algebraic CELP codec [Eku99][ETS00] that is also referred to as the AMR codec. In the listening test, the standardized 2.4 kbps MELP codec [Sup97] was used as the second codec. Different types of well-known standardized speech codecs were used to demonstrate that the proposed method can be used with practically any speech codec, without changing the codec implementation at all.

Objective evaluation using the AMR codec

In the objective part of the evaluation, the standardized AMR codec was used to code Finnish speech and segmental *signal-to-noise ratio* (SNR) was calculated between the input and the output of the codec. The test material consisted of about 39 minutes of female and 31 minutes of male speech. Both the original and the preprocessed speech was coded with each of the eight modes of the codec and the segmental SNRs were determined.

Table 3.12: Results of the ACR test with the two standardized codecs.

	Condition	Female	Male
01	MNRU Q = 8 dB	1.23	1.25
02	MNRU Q=14 dB	2.02	2.14
03	MNRU Q=20 dB	2.98	3.04
04	MNRU Q=26 dB	3.70	3.50
05	Direct	4.16	3.66
06	Preprocessed	4.41	4.09
07	AMR 5.15 kb/s original	3.54	3.32
08	AMR 5.15 kb/s preproc.	3.54	3.54
09	AMR 6.70 kb/s original	3.93	3.48
10	AMR 6.70 kb/s preproc.	4.21	3.73
11	AMR 7.95 kb/s preproc.	4.18	3.96
13	AMR 12.2 kb/s original	4.02	3.68
14	AMR 12.2 kb/s preproc.	4.27	3.96
15	MELP 2.4 kb/s original	2.48	2.64
16	MELP 2.4 kb/s preproc.	2.61	2.71

Table 3.11 presents the SNR values and the improvement percentages when using the preprocessed speech instead of the original as the coder input. It can be clearly seen that the codec has performed better with the preprocessed than with the original speech even though no changes or optimizations were made to the codec that would support the handling of preprocessed speech.

Listening test with speech codecs

To confirm the promising SNR figures, another listening test was arranged, this time using absolute category rating test [ITU96] and the mean opinion scores on a five-point scale. The test contained 16 different conditions, including four *modulated noise reference unit* (MNRU) conditions with the noise level ranging from 8 to 26 dB. The remaining conditions tested the quality of the preprocessed speech and the performance of the AMR codec and the 2.4 kbps MELP codec with both original and preprocessed speech as the input signal.

The test material was spoken by two male and two female speakers and two sentences were chosen from each. These eight samples were normalized to -26 dBov and processed through each of the 16 conditions. Thus, each listener assessed the quality of 128 samples in random order. Altogether 14 naive listeners participated in this test. The MOS values for the male and female speakers for all the conditions are shown in Table 3.12 and the combined MOS values together with their 95% confidence intervals in Figure 3.10.

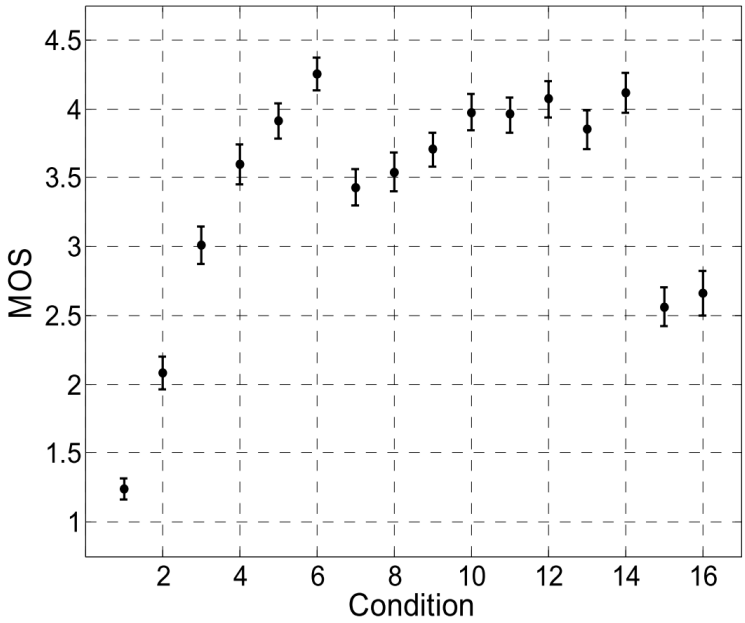

Figure 3.10: Combined MOS results with 95% confidence intervals. The conditions are listed in Table 3.12. (From [Läh03b].)

3.5.4 Discussion

The masking model often judges even more than 40% of the frequency components of a frame to be zeroed. Nevertheless, the results of the perceptual evaluation indicate that the preprocessing causes very little or no perceptual degradation. Even without the extra speech level amplification, the quality deterioration is hardly perceivable. When combined with the additional level adjustment, the speech quality remains essentially unaltered in the preprocessing, as can be seen from Table 3.10. The MOS difference between conditions 5 and 6 in the ACR test even implies a slight enhancement in the speech quality due to the preprocessing, but the magnitude of this particular difference in Table 3.12 should not be compared with the results of the earlier listening test because of the fundamental difference between the test methods.

The results of the ACR test demonstrate that the usage of the proposed preprocessing technique in the front end of a speech coder systematically improves the speech quality, indicating that the preprocessing allows the coding process to better focus on the perceptually important content. Both waveform-approximating and parametric codecs have been tested and the direction of the change has remained the same. It should be noted that these results have been obtained without any modifications to the standardized speech codecs. In the case of the VLBR codec, since there is no need to follow any standard, the quantizers of the codec

can be trained specifically for parameter data estimated from the altered speech signals.

3.6 Conclusions

This chapter has introduced the VLBR codec and the main techniques behind its efficiency. The first cornerstone of the codec is its parametric representation based on simplified sinusoidal excitation modeling. Second, the speech codec exploits the segmental nature of speech and uses adaptive downsampling and quantization schemes to maximize the speech quality with respect to the average bitrate. The quantization efficiency is further enhanced using a predictive multi-mode quantization approach for joint coding of adjacent speech parameter values or vectors. The predictive modes of the quantizer structure utilize an efficient low-complexity prediction scheme whereas one memoryless mode is included to enhance the performance. Enhancements to the conventional dynamic codebook reordering technique are also proposed to enable further bitrate savings through enhanced lossless compression of the quantizer indices. Finally, a novel preprocessing method based on perceptual irrelevancy removal is proposed for making the input signal easier to compress.

The final bitrate achievable using the complete VLBR codec containing all of the components described in this chapter depends a lot on the application scenario, on the target speech quality, on the speech data to be compressed, and whether or not the codec is trained specifically for the particular data (this issue will be discussed more closely in Section 4.2). Based on practical experimentations carried out by expert listeners, relatively good and generally "acceptable" speech quality can often be achieved at rates of about 0.7-1.0 kbps for active speech (due to the large amounts of silence in typical speech signals, the average bitrate for the whole data can be even lower than this). The bitrate can be increased from this very low level to obtain higher speech quality but the simplified parametric model quickly becomes a quality bottleneck. In practice, the point of saturation is reached at latest at bitrates of about 3.0 kbps. After this rate, only very slight improvements of speech quality can be obtained without changing the parametric model.

After the completion of the research work introduced this chapter, there have been some partially related advancements in the field of speech coding at bitrates below 1.0 kbps. For example, the publications [Ram06], [McC08], [Jah08], [Ram09], [Unv10], and [Ram12] present results that can be seen interesting. However, direct comparison to the VLBR codec presented in this thesis is not possible or meaningful because of the differences in the design constraints, bitrates and in the overall application scenarios. Thus, the work presented here can be considered to represent the state of the art and to offer a unique tradeoff between the speech quality and the bitrate, memory usage, and computational load.

Chapter 4

VLBR-based concatenative speech synthesis

Even though there are some network based text-to-speech systems available, on-device integration is by far the most common approach used when introducing TTS technology into new devices. As discussed in Section 2.4.3, concatenative unit selection based synthesis is currently the predominant technology in the commercial TTS systems and applications, despite the recent popularity of statistical speech synthesis techniques on the research side. As evidenced most concretely by the transition from diphone synthesis to unit selection synthesis, an increase in the size of the database generally increases the synthesis quality. Unfortunately, the size of the database also directly affects the memory consumption and the computational load. This is problematic in certain environments, such as in the current low-end mobile devices, where the available memory sizes and computational resources are still rather limited despite the continuous improvements in both of these areas. In addition, even on high-end devices, extra computations may shorten the battery life, and the users may also wish to preserve more memory for their own purposes instead of having to store large unit selection TTS databases. This is especially relevant when TTS systems operating on different languages and perhaps databases from different speakers are pre-installed on a device. Thus, there is still a clear demand for solutions that reduce the footprint and the computational complexity of unit selection based TTS systems.

In the literature related to the use of unit selection synthesis on embedded devices, or more generally to the reduction of the memory footprint of unit selection databases, the most typical approach is to use different methods for pruning seldomly-used or redundant units from the database. Examples of such database reduction methods can be found, e.g., in [Rut02], [Nuk06], and [Tsi08]. These approaches are valid but the database should be kept large enough to maintain the naturalness. In the work reported in this thesis, the practical constraints dictated that the unit selection databases would have to be compressed using bitrates of

about 1–2 kbps to fit them in the available memory space, even after database pruning.

Speech coding techniques have been used for compressing the databases in the literature but the bitrates are typically rather high. For example, bitrates of 25.6–51.2 kbps were reported in [Sch02], [Fék06] talked about the bitrate of 32 kbps, and [Kar09] used CELP-based codecs with compression ratios between 7 and 10, corresponding to 12.8–18.3 kbps with 8-kHz signals. In [Sar08], standardized speech codecs with bitrates from 4.75 kbps to 13 kbps were used for compressing the databases. The lowest bitrates are reported in [Cha02b] that used MFCC and sinusoidal model based compression at 3.8–7.3 kbps, and in [Kai07] that compressed the databases using an asynchronous interpolation model at 3.4 kbps. Another potentially low-bit-rate solution was reported in [Hua97] but the total bitrate was not mentioned. However, the mixed excitation model used at least 1.6 kbps for the LP coefficients alone, and pitch, voicing and excitation data needs to be stored in addition to that. Thus, it can be concluded that the other reported solutions, even at the time of writing this thesis, do not fulfill the memory requirements that were faced in the work.

Based on the concrete need to obtain low footprints for the unit selection based text-to-speech systems, and the good results obtained in the VLBR codec development discussed in Chapter 3, the logical next step was to integrate the VLBR codec into concatenative unit selection-based TTS systems. The first section of this chapter discusses the different practical issues that need to be considered when performing this integration, as well as some of the beneficial properties of the resulting VLBR-based synthesis solution. Also, the most important experimental findings observed during the work are discussed. Section 4.2 introduces the concept of dynamic quantizers and the related possibility to retrain the codec or some of its quantizers specifically for a given TTS database to obtain further reductions in the memory consumption. Finally, in Section 4.3, a novel and highly efficient solution for computing the concatenation costs in unit selection based TTS is presented.

The discussions in the first two sections of this chapter are loosely based on the patent applications [Nur07b] and [Nur11b], as well as on the recent publication [Nur13a], but a significant amount of new information is provided as well. The third section is based on [Din08].

4.1 Use of VLBR for the compression of TTS databases

The task of compressing text-to-speech databases bears a lot of similarities to the task of compressing speech data for storage purposes. In both cases, efficient compression is desired but excessive degradation of speech quality needs to be avoided. Thus, the VLBR codec described in Chapter 3 is readily a potential candidate for handling efficient compression of TTS databases.

This section deals with the topic of integrating the VLBR codec into concatenative TTS systems in a seamless manner. The discussion is not tied to any particular TTS system but instead different implementation alternatives are considered and different practical aspects of the integration are discussed. The experimental findings presented in this section confirm that the VLBR codec can be successfully used for efficient compression of TTS databases and for generating speech output based on the compressed speech units. The topic of database pruning is considered to be outside the scope of this thesis, and is left outside the discussion. It should nevertheless be noted that in practical applications, best results can be obtained using a combination of database pruning and compression.

Before the work presented in this section was carried out, the use of sinusoidal model based parameterizations in concatenative synthesis had already been confirmed to be a good solution in a number of studies, e.g., in [Cre97], [Sty00], [Sty01], and [O'B01]. However, as mentioned in the beginning of this chapter, extremely high compression ratios had not been studied before in the context of unit selection synthesis. Considering the database compression ratios and the bitrates reported in the literature, significant memory savings can be obtained by compressing the databases and handling the synthesis using the proposed VLBR-based approach.

4.1.1 Overview of VLBR-based concatenative synthesis

A simplified block diagram demonstrating the use of the VLBR codec in a TTS system is shown in Figure 4.1. The complete TTS process starts with the processing of an input text by a TTS front-end, continues with the selection of the units and proper handling of the unit database compressed using the VLBR codec. Finally, the process ends with parametric concatenation and the actual generation of the synthesized speech signal. The database related issues are discussed in Section 4.1.2 whereas the other parts of the process are covered in the following subsections.

TTS front-end

The details on the operation of the TTS front-end and on its output are out of the scope of this thesis but on a general level, a text-to-speech front-end is used for analyzing the input text and for generating a description of how the input text should be spoken. As discussed in Section 2.4.1, the output of the TTS front-end contains a narrow phonetic transcription, i.e., the phonetic representation of the input text and some additional information related to the desired prosody. Various types of narrow phonetic transcriptions could be used, depending on the implementation of the TTS system. Typically, some kind of linguistic analysis is used for deriving features that describe the context in which each phoneme appears. These features

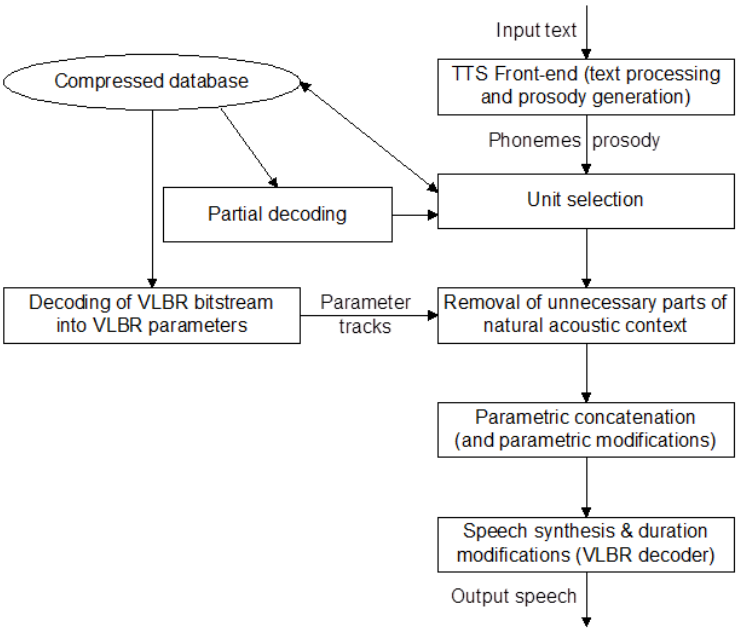

Figure 4.1: Simplified block diagram demonstrating the use of the VLBR codec in a concatenative TTS system. The database is compressed using the principles described in this section.

can then be used in the computation of the target cost in the unit selection process to roughly obtain the desired prosody. The TTS front-end can also contain a specific module for generating a target prosody that is used for guiding the unit selection and/or the signal generation/modification.

Unit selection and optional partial decoding

The VLBR parameters can be used as additional information in the unit selection. Also, if the codec parameters are used in the selection, it may no longer be necessary to store side information such as pitch contours or spectral content at the unit boundaries for the purpose of unit selection since this information is already available in the bitstream, leading to further memory savings. Naturally, these potential memory savings come at the expense of additional computation load caused by the need to decode the bitstream. However, due to the parametric coding approach, it is straightforward to decode only the relevant parameters, e.g., a pitch contour without a need to decode the whole parametric representation or the actual speech segment. In the block diagram presented in Figure 4.1 this is referred to as partial decoding. It is worth noting that bitstream decoding is not computationally intensive since typically only table lookups and simple computations are needed.

The use of the VLBR parameters can be particularly convenient in the computation of the concatenation costs. Depending on TTS system and the requirements of the application, it could be possible to compute the concatenation costs directly based on the LSFs (and possibly some other parameters if desired). Furthermore, it is possible to approximate concatenation costs using directly the quantizer indices as will be shown in Section 4.3. This makes the calculation of the concatenation costs computationally very efficient.

Removal of unnecessary acoustic context

As will be discussed in Section 4.1.2, the segmental parametric data retrieved from the database may contain parameters outside the actual TTS unit, either due to the segment boundary locations or due to the desire to include certain amount of acoustic context for each unit. If present, these extra parameters need to be removed before or during the concatenation and parametric synthesis. The removal can be performed already when processing the bitstream or after the bitstream has been decoded into parameter tracks, assuming that the acoustic context is not used in parameter smoothing.

Parametric concatenation and modifications

The concatenation is performed in the parametric domain using the VLBR parameters simply by combining the parameters representing different units into continuous parameter streams. At the unit boundaries, the parameter values can be smoothed as an optional step to achieve better continuity. The smoothing can be applied to all or to some of the parameters, or even using some alternative parameters derived from the VLBR parameters. The smoothing algorithm itself can also be realized in many ways. A relatively thorough discussion covering several smoothing methods can be found in [Cha02a]. From the viewpoint of this thesis and parameter stream concatenation, it is enough to note that the smoothing methods can be considered to belong to three distinct categories:

1) Smoothing that happens inside the unit boundary based on the parameter values in the neighboring concatenated unit, i.e., on the other side of the concatenation boundary. An example of this type of smoothing approach is provided in [Mac96] where weighted versions of the feature values/vectors from one side of the boundary are added to the feature values/vectors on the other side of the boundary, and vice versa. With the weights used in [Mac96], this corresponds to simple linear interpolation of the features in the boundary regions. In this scenario, no additional acoustic context is needed for the smoothing.

2) Smoothing that uses the method of optimal coupling [Con97]. In optimal coupling, the unit boundaries are allowed to move within a small region in order to find the best fit with the adjacent units. Performing additional parametric smoothing together with optimal coupling is not mandatory since the continuity

can be improved already by the optimization of the boundary locations. When optimal coupling is used, a certain amount of acoustic context needs to be decoded into the parametric form and kept in memory until the final boundary locations are found. Another alternative would be to perform optimal coupling after generating the excitation signal or even in speech domain, i.e., the acoustic context would be synthesized, too, and the extra portions from the excitation or the speech signal would be removed once the final boundary locations are known. This approach, however, would result in heavier signal processing needs and increased complexity, and the resulting speech quality could be compromised as well.

3) Smoothing that utilizes the acoustic contexts. In this alternative, a certain amount of acoustic context is decoded in addition to each unit. At each boundary, weighted versions of the feature values or vectors of the neighboring unit's context are added to the feature values of the current unit. For good results, it is required that the contexts have a good match, i.e., the units on the different side of the boundary had fairly similar neighboring units as their original context.

The methods 1 and 3 can be combined with the method 2. The selection of the smoothing method is not trivial as different types of smoothing would typically be optimal in different kinds of scenarios. The effects that parameter smoothing has on the resulting speech quality in VLBR-based concatenative synthesis are discussed in Section 4.1.3.

In addition to the optional smoothing at the unit boundaries, it is also possible to perform other kinds of parametric processing during the concatenation. For example, voice conversion could be performed on VLBR parameters at this stage. This option will be the main topic of Chapter 5. In addition to voice conversion, parametric processing could also be used, for example, to make the synthesis result match a desired modeled prosody. Duration modifications should be treated separately during the signal generation as will be discussed in the next subsection but other prosodic changes can be handled by manipulating the parameter values, e.g., the pitch values.

Regardless of the type of parametric modifications, it is important to note that for the time instants at which the pitch values are modified, the residual amplitude spectra need to be modified as well to take into account the changed number of pitch harmonics. This involves, together with possible additional spectral modifications, resampling of the amplitude spectra at the new fundamental frequency and its integer multiples using interpolation.

Speech synthesis and duration modifications

The actual speech synthesis is performed using the VLBR decoder. The concatenated parametric representation is used as the input and the decoder produces the speech output. The use of the VLBR decoder for generating synthesized speech signals does not require any large modifications to the decoder itself: the only

requirement is the separation of the bitstream processing, dequantizations and the speech signal generation, according to the considerations discussed in this chapter.

The actual signal generation is performed similarly as in the case of speech coding for storage applications. As discussed earlier, this approach described in Section 3.1.3 also allows straightforward high-quality duration modification during speech synthesis. Thus, the durations in the speech output can be easily modified to fit the desired durations, if necessary.

4.1.2 Database compression

The VLBR codec can be used for the compression of basically any kind of TTS-related speech databases. However, to make the compression as efficient as possible, the organization of the database needs to be taken into account during the compression and retrieval of speech units.

Database organization and compression

In general, TTS databases are usually organized using one of the following two alternatives: 1) similar units are grouped together or 2) the database is organized as sentences. This leads to two different main approaches for the implementation:

1) Compression of non-continuous units: In this case, the compression is performed using the original recorded sentences in such a way that the natural acoustic context is also given as input into the encoder to enhance the quality and continuity of the speech output. Parts of the natural context can also be included in the bitstream of each unit, if desired due to the possible reasons described in the previous section. The VLBR segment boundaries are forced into the locations of the (possibly context-extended) unit boundaries. After the compression, the compressed database is organized into its final form (e.g., the bitstreams generated from the different instances of the same diphone are grouped together for storage, etc).

2) Compression of continuous speech data: In this case, complete sentences are given as input into the encoder. The VLBR segment boundaries can either be forced into the locations of the (possibly context-extended) unit boundaries or they can be placed according to the results of the speech analysis and segmentation performed by the VLBR encoder. In the former alternative, the retrieval time and complexity is minimized while the latter alternative maximizes the compression efficiency.

Since the bitrate of the VLBR codec is variable it is necessary to also store detailed location information for each unit, in addition to the actual bitstreams, but the most efficient storage and organization of this side information depends on the implementation of the TTS system and its unit selection algorithm, as well as on the speech database. The durations of the units and the durations of the possible extra context portions also need to be stored either explicitly or implicitly. Other

kinds of side information data can be stored as well, for example to facilitate efficient unit selection. The detailed content and format of the side information data is for the most part not specifically linked to the VLBR codec, and can be considered to be outside the scope of this thesis.

Unit retrieval and decoding

The unit retrieval can be implemented in two different ways, depending on the database compression method:

1) Retrieval of non-continuous units: In this alternative, the non-continuous unit is retrieved directly based on the location information. The retrieved unit is decoded as such. The possible additional acoustic context can be deleted after decoding, at latest during the concatenation as described earlier in this section.

2) Retrieval of units from continuous speech data: In this alternative, it may be necessary to decode additional data if the segment boundaries are not forced to match with the (possibly context-extended) unit boundaries. After decoding, the possible additional parameter values or vectors at the beginning of the first segment and at the end of the last segment of the unit should be deleted prior to the concatenation.

Codec retraining

Since the total size of the quantizer codebooks used in the VLBR codec is much smaller than the total size of a typical TTS database (less than 100 kilobytes vs. at least several megabytes), the codec can be retrained specifically for every TTS database in order to achieve the best compression ratio and speech quality. The retraining can be performed using conventional training methods with the uncompressed TTS database as the training material. The speaker-specific quantizer designs and codebooks can be stored as one part of the compressed TTS database, as will be discussed more thoroughly in Section 4.2.

4.1.3 Experimental findings and discussion

The VLBR-based synthesis approach has been implemented and integrated into several different concatenative TTS systems operating on different languages and with different types of speech units ranging from half phonemes to full syllables. A published example of such a system is the Mandarin Chinese TTS system [Tia06]. The codec integrations and the related database compressions, not discussed before in any publications, followed the guidelines given in this section and were always relatively straightforward.

One of the main findings in the practical experiments carried out as a part of the thesis work was that for the best results, the VLBR codec should be tuned specifically for the different speakers, database sizes and languages, as will be

discussed more closely in Section 4.2. This does not only involve quantizer codebook trainings but the bit allocations could be changed as well: To achieve a perceptually similar speech quality level, a slightly higher or a slightly lower average bitrate might be needed for some particular database compared to another database. Also, as always in speech coding and synthesis, special attention might be needed in the quantization of certain parameters with certain languages. For example, with tonal languages such as Mandarin Chinese, the pitch parameter should be stored quite accurately. The use of dynamic quantizer structures presented in Section 4.2 offers a convenient mechanism for enabling such speaker, database and language specific optimizations. The optimizations can be carried out by experts manually but it would also be possible to use automatic optimizations with pre-specified objective distortion limits.

Another main finding concerns the concatenation of the speech units and the optional parameter smoothing. Different variations of the three smoothing methods discussed earlier in this section were implemented and evaluated by expert listeners but the main outcome of these experiments was that the improvements offered by the smoothing were inconsistent at best: Sometimes the smoothing improved the speech quality but in some other cases the same method resulted in a clear quality degradation, even with the same database. This result was expected and in line with the findings presented in [Cha98b] and [Cha02a].

It should be noted that the use of VLBR for synthesis offers inherent mechanisms for avoiding discontinuities and for enhancing the smoothness at the concatenation boundaries, even without any additional smoothing, and this might be sufficient for taking care of the most critical smoothing needs. The first inherent continuity-enhancing mechanism is the use of modeled linearly evolving and/or random phases instead of estimated phases. While this necessarily causes small modeling errors, it also effectively avoids phase discontinuities at the unit boundaries. The second inherent mechanism for enhancing the smoothness at the concatenation boundaries is offered by the use of interpolation for obtaining parameter values in a pitch-synchronous manner during the signal generation. This interpolation in a way provides automatic smoothing in regions placed between the parameter values or vectors coming from different units.

Due to the above findings and considerations, no additional smoothing was used for treating the unit boundaries. The final major experimental finding concerning the basic VLBR-based concatenative synthesis also deals with the topic of parametric concatenation but from a slightly different angle. More concretely, expert listeners compared VLBR-based synthesis results against the corresponding synthesis results obtained using the same TTS systems, languages and databases but with concatenation of uncompressed waveforms. The focus was placed on studying the successfulness of the concatenation process. The conclusion of this part of the study was that VLBR-based concatenation itself does not seem to introduce new artifacts. In other words, the artifacts present in the synthesis results

were always either also present in the version generated using the uncompressed database or were caused by the compression, not by the concatenation (this was confirmed by listening to the original TTS database sentences after VLBR compression). Based on these findings, it is safe to conclude that the quality bottlenecks in VLBR-based synthesis are the parametric model, the bitrate used for the compression and the quality of the original TTS database, including its annotations (naturally together with the performance of the unit selection algorithm).

When used with large TTS databases and manually-tuned unit boundary annotations, together with database-specific retraining, VLBR-based concatenation performs very well and is capable of producing intelligible and relatively good overall speech quality even at database compression bitrates of about 1.0 kbps and even slightly below that. Higher bitrates can be used for obtaining higher speech quality but it should be noted that the simplified parametric model quickly becomes the limiting factor, and to obtain meaningful benefits from bitrates higher than about 3.0 kbps, the parametric speech model used in VLBR should be improved or changed, similarly as in the case of storage applications.

4.2 Dynamic quantizers and codec retraining

In the area of speech coding, some research has been carried out related to speaker-specific compression (see, e.g., [Rou82], [Rou83], [Shi88], and [Cer98]) but the vector quantizers and their codebooks have traditionally been fixed in terms of their structure and size. For example, the codebooks of standardized speech coders used in mobile devices are typically stored in the read-only memory, with the obvious exception of adaptive codebooks. In addition, the bit allocations are also typically static, even in codecs that can operate on multiple bitrates.

In this section, the simple but effective concept of dynamic quantizer structures, originally described in the patent [Nur11b], is presented. The main idea in this work is to make the quantizer designs and the codebooks fully dynamic and configurable at run time. This enables easy reconfiguration of the codec as well as retraining and optimization of the quantizers specifically for different use cases. The use of dynamic quantizers and quantizer retrainings enhances the compression efficiency, for example in the compression of text-to-speech databases. The memory saving benefit achievable using quantizer retraining is demonstrated using LSF quantization as a practical example.

4.2.1 Making the quantizers dynamic

Flexible retraining of the VLBR codec is obtained through the use of dynamic quantizer structures. In practice, to enable full retraining of the VLBR codec in a wide sense, all the quantizers need to be fully reconfigurable. To realize this, the design of the VLBR codec and especially the quantizer designs needs to be

considered. To recap, the most important properties of the overall quantization approach used in the VLBR codec are the following:

1) The compression uses different processing for different types of input. As described in Section 3.2, different bit allocations and quantizer designs are used for the different segment types.

2) All the quantizers can be realized using the vector-predictive multi-mode matrix quantizer structure presented in Section 3.3.

3) According to the principles presented in Section 3.3.1, there may be several predictive and/or memoryless quantizer modes for a certain type of speech data (possibly with different types of predictors).

4) The sub-quantizers can contain codebooks with one or more stages.
The quantizers are considered dynamic if all of the above aspects can be reconfigured at run time.

The dynamic quantizer data can be represented in a compact manner as a bitstream consisting of two parts, a configuration header specifying the quantizer designs and the actual quantizer data including the codebooks and possible predictors. The bitstream is used as an additional input during the initialization of the codec. In addition, default settings and quantizers are stored as a part of the codec itself, to allow partial reconfiguration and even usage without separate reinitialization.

In the case of concatenative TTS, each compressed database is complemented with the above-mentioned dynamic quantizer bitstream. The bitstream begins with the configuration header that fully specifies which of the quantizers are reconfigured and what are the new quantizer structures. Then, the actual quantizer data is only included for the listed quantizers. This solution allows the greatest possible flexibility and compression efficiency: the quantizer structures can be freely selected based on the concrete needs and optimization (e.g., differently for different languages and speakers) but at the same time it is possible to include in the reconfiguration bitstream only the new quantizers that make the overall compression more efficient, considering the overhead needed for storing the quantizer data. A practical example of a configuration header structure is given in the patent [Nur11b].

The quantizer data and the configurations are allocated fully dynamically into the codec memory. Together with the above considerations, this allows each subquantizer to be updated individually. Furthermore, to save memory, the quantizer data can be shared between different quantizers. In particular, a compression scheme for a given parameter may use the same codebooks and/or predictors for different segment types (voiced, unvoiced, mixed) with single memory instance. The codebook and predictor data can also be shared between different speakers, for example to realize gender-specific quantizers.

4.2.2 Practical experiments: Is speaker-specific retraining useful?

With the above solution involving dynamic quantizer structures, it is possible to reconfigure only those quantizers or codebooks that can offer enhanced compression efficiency even when the additional memory needed for the quantizer data is also taken into account. Thus, it is safe to state that proper use of codec retraining never makes the compression less efficient. However, an important question remains: Can speaker-specific retraining lead to significant memory savings?

Experimental set-up

To study the issue of codec or quantizer retraining, several LSF quantizers were specifically retrained for two TTS databases and similar quantizers trained for a generic training set served as a reference. The two TTS databases used in the study were the publicly available CMU Arctic US English voices *slt* and *rms*, i.e., a female voice and a male voice. For the first part of the test, LSF data sets consisting of $145\,654$ and $177\,612$ vectors were obtained for *slt* and *rms*, respectively, using LP analysis with a 20-ms update interval and a voice activity detector for discarding silent frames. For the second part of the test, similar LSF sets were generated but with the update interval of 10 ms. The generic training set was generated similarly using speech from multiple speakers and languages. The total size of the set was $203\,250$ vectors, including 6000 vectors from *slt* and *rms*.

To make the experimental results relevant more widely than just for the case of the VLBR codec, no segmentation or adaptive downsampling was used and the same quantizer was used for quantizing the whole data instead of using different processing for different segment types. Furthermore, all the quantizers included in the experiment were multistage vector quantizers, trained using the simultaneous joint design algorithm described in Section 2.3.2. Different bitrates from 10 bits/vector to 26 bits/vector were included in the test.

In the first part of the experiment, the evaluation focused on comparing the performance of multistage vector quantizers with the maximum codebook size of 64 vectors/stage. The performance of the quantizers was evaluated using spectral distortion defined in Equation (3.19). Similarly as in Section 3.3.3, the weighting scheme defined in [Pal93] was used both during the training and the quantizations. The experiment was repeated with the maximum codebook size of 256 vectors/stage.

The second part of the experiment studied the effect that the size of the speech database has on the achievable memory saving. The 26 bits/vector MSVQ with the maximum codebook size of 64 vectors/stage and trained with the generic data set was selected as the benchmark, and now the task was to find with different database sizes a similar specifically-trained quantizer that achieves or exceeds the quantization accuracy of the benchmark quantizer. When such quantizer was found, the total memory saving was calculated, so that the memory needed for

Table 4.1: Performance of multistage LSF vector quantizers of different sizes (using at most 6 bits per stage) for the databases *slt* and *rms*, measured using spectral distortion in dB. The left column for each speaker presents the results obtained using database-specific retraining whereas the right column contains the results obtained using quantizers trained with generic multi-speaker data. Gray background is used for highlighting some cases where roughly similar or slightly better performance was achieved using the proposed retraining than with the generic quantizers despite the drop in the bitrate.

		slt		*rms*	
Bits	Stage sizes	*slt*	gen	*rms*	gen
10	{ 64, 16 }	2.21	2.73	2.29	2.83
11	{ 64, 32 }	2.06	2.54	2.12	2.62
12	{ 64, 64 }	1.92	2.37	1.98	2.45
13	{ 64, 64, 2 }	1.81	2.24	1.86	2.33
14	{ 64, 64, 4 }	1.70	2.11	1.75	2.19
15	{ 64, 64, 8 }	1.60	1.98	1.65	2.07
16	{ 64, 64, 16 }	1.50	1.86	1.54	1.93
17	{ 64, 64, 32 }	1.41	1.76	1.44	1.82
18	{ 64, 64, 64 }	1.32	1.65	1.36	1.69
19	{ 64, 64, 64, 2 }	1.24	1.57	1.28	1.62
20	{ 64, 64, 64, 4 }	1.18	1.48	1.21	1.53
21	{ 64, 64, 64, 8 }	1.11	1.40	1.14	1.43
22	{ 64, 64, 64, 16 }	1.04	1.32	1.07	1.35
23	{ 64, 64, 64, 32 }	0.98	1.25	1.01	1.27
24	{ 64, 64, 64, 64 }	0.93	1.18	0.95	1.20
25	{ 64, 64, 64, 64, 2 }	0.88	1.11	0.90	1.13
26	{ 64, 64, 64, 64, 4 }	0.83	1.06	0.85	1.08

storing the new codebooks was taken into account as well. The databases of different size were obtained by using different-sized subsets of the full databases.

Experimental results

The results given in Table 4.1 for the maximum codebook size of 64 vectors/stage indicate that training the quantizers specifically for a given TTS database improves the quantization accuracy significantly, compared to the case where the quantizers are trained using generic speech data, even when training material from the particular speaker is included in the training set. Due to the improved accuracy, the quantization bitrate can be reduced without causing any quality degradation. With higher bitrates, the performance advantage is about 15% or 4 bits/vector, i.e., an equal or lower SD can be obtained using a 22-bit quantizer trained specifically

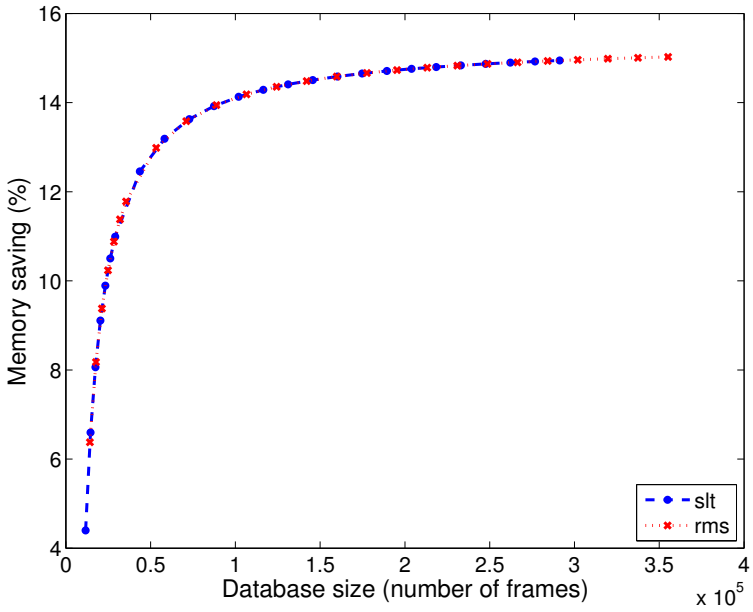

Figure 4.2: Concrete memory savings for databases of different size (for the speakers *slt* and *rms*).

for the database instead of using a 26-bit quantizer trained using generic data. At lower rates, the relative bitrate advantage is increased to about 23% but in terms of bits/vector, the difference is only 3 bits per vector. These results are highlighted in the table using gray background color for marking equal or slightly improved performance at higher and lower bitrates. Even though the numerical SD values are slightly different for *slt* and *rms*, the bitrate advantages are identical.

When the experiment was repeated with the maximum codebook size of 256, the results were perfectly in line with the results reported above. The numerical SD values obtained in this second run were slightly better than in the first run due to the less strict limitation on the maximum codebook size. Nevertheless, the relative and absolute bitrate advantages were identical in both cases, i.e., 4 bits/vector were saved at higher rates while the advantage at lower rates was 3 bits/vector. Due to the similarity with the results presented in Table 4.1, these results obtained with the larger maximum codebook sizes are not shown in detail.

The results obtained in the second part of the experiment are summarized for both speakers in Figure 4.2. The figure depicts the concrete memory saving obtained for the different database sizes using the retraining of the LSF quantizer, while matching or exceeding the quantization accuracy of the benchmark quantizer operating at 26 bits/vector. The memory savings are given as percentage values, relative to the total memory required for storing all the LSF related data for the different database sizes. The memory needed for storing the additional

LSF codebook data is taken into account. As expected, the relative benefit depends on the database size. With larger databases, the benefit is more significant because the memory required by the codebook still stays the same. In this experiment, the relative memory saving curves were basically identical, smooth, and continuous because the size of the database did not happen to affect the quantizer size in these cases but more generally changes in the quantizer bit usage could cause small discontinuities in the curve (i.e., sometimes a smaller database can be compressed using a smaller-sized quantizer).

It should be noted that even though the bitrate advantages achieved in these experiments were very consistent, the bitrate savings that can be achieved in the compression of TTS databases do depend on the database/speaker and on the quantizer design. Also, the performance advantage depends on the parameter to be quantized. In the case of LSFs, the difference is quite clear, as demonstrated in this section, but for example with the gain parameter, the performance advantage is in practice very small or even non-existent because the energy level is typically normalized as a part of preprocessing.

As a final note concerning the results, the possible changes in the recording conditions have their own effect on the results. In the experiments discussed here, the recordings were not carried out in fixed conditions, and there were differences, e.g., in room acoustics, microphones, and background noise levels. As a consequence, the performance advantage obtained using database specific training was slightly larger than it would have been if all the databases were recorded in identical conditions. However, based on separate comparisons performed on databases recorded in identical conditions vs. databases recorded in different conditions, it seems that the extra benefit in the case of LSF quantization is at most 1 bit per vector or even smaller. Also, when dealing with multilingual TTS systems and multiple voices, it is likely that the recording conditions in all databases are not strictly fixed. Thus, the results provided in this section can be considered valid and representative of a realistic scenario.

4.3 Compression-motivated method for computing concatenation costs

As discussed in Section 2.4.3, the concatenation costs play an important role in unit selection based synthesis [Hun96]. The resulting calculations also clearly form the most computationally intensive part of the entire synthesis process. Preselection techniques, such as [Nis07], reduce the complexity by reducing the size of the search space. Similarly, database reduction techniques, such as the one proposed in [Tsi08], also make the search space smaller and thus reduce the computational load.

In this section, a very efficient novel solution, originally presented in [Din08], is introduced for the calculation of the concatenation costs. In the proposed technique, the multistage vector quantization approach discussed in Section 2.3.2 is used for compressing the acoustic features describing the boundaries of each unit. The multistage codebooks of the quantizer are specially trained for the feature vector data, using the MSVQ simultaneous joint design algorithm [LeB93] presented in Section 2.3.2. In addition, the idea of pseudo-gray coding [Zeg90] is utilized for approximating Euclidean distances between features directly based on their multistage indices. Binary switching algorithm adapted for the multistage structure is used as a locally optimal solution to rearrange codebook indices.

The proposed scheme is evaluated in a TTS system. Based on the listening test results, the use of the proposed approximative solution for computing concatenation costs does not seem to degrade the speech quality despite the significant reduction of the computational complexity, even when used with uncompressed speech databases. In addition, the memory needed for storing the feature data at unit boundaries may be reduced as well. These properties make the proposed technique especially suitable for implementations designed for use on embedded devices such as mobile terminals. It is worth noting, however, that if the method is used together with the dynamic codebook reordering technique discussed in Section 3.4, the original indices need to be used in the computation instead of the reordered ones.

4.3.1 Concatenation cost calculation in unit selection TTS

As a reminder from Section 2.4.3, the front end of a TTS system converts the input text into a target sequence, $\{t_1, t_2, \ldots, t_T\}$, where T denotes the total number of units to be concatenated to form the output speech, and $U = \{u_1, u_2, \ldots, u_T\}$ denotes a candidate sequence selected from the entire speech database. The selection of the units is carried out using the target cost C_t and concatenation cost C_c [Hun96], and in particular the minimization of the sum of these two costs, $C = C_t + C_c$, computed over the entire sequence. Weighting can be used for adjusting the relative importances of the target cost and the concatenation cost.

This section focuses on the calculation of the concatenation cost part. The function of the concatenation cost is to measure the perceptual cost that is introduced when two speech units are concatenated together. The measurement uses a set of features that represent the perceptually essential aspects of speech to calculate the distance. Considering that U is a candidate sequence, the concatenation cost for the whole candidate sequence can be computed as

$$C_c = \sum_{j=2}^{T} C_c(u_{j-1}, u_j), \tag{4.1}$$

where $C_c(u_{j-1}, u_j)$ is the concatenation cost of adjoining units.

During the design of a unit selection based text-to-speech system, a set of acoustic features to be used for calculating the acoustic distance at the joined boundaries of two adjacent units is selected. Let $\mathbf{f}_j$ and $\underline{\mathbf{f}}_j$ denote the acoustic feature vector representing the first frame and the last frame of the unit u_j, respectively. The concatenation cost can be rewritten as

$$C_c(u_{j-1}, u_j) = d_c(\underline{\mathbf{f}}_{j-1}, \mathbf{f}_j). \tag{4.2}$$

The concatenation cost function is denoted as d_c. Optimally, the feature set and the concatenation cost function should be carefully designed by analyzing and maximizing the correlation between the perceptual concatenation quality, evaluated through listening tests, and the corresponding cost measurements carried out using the feature set and the cost function. A typical feature set could include energy, pitch, and/or other features, such as line spectral frequencies or cepstral coefficients, and a typical concatenation cost function is the squared error or the weighted squared error, computed similarly as in the case of vector quantization (see Equation (2.9) and Equation (2.10)). The different features can be combined into one feature vector and the relative importances or scaling differences can be taken into account using weights for the different vector components.

4.3.2 Computational load of concatenation cost calculation

The main aspects affecting the computational load related to the traditional calculation of the concatenation cost are the size of the database, the cost function, and typically also the feature set and its dimension. The size of the database affects the computational load through the number of possible instances for each unit. In principle, all the instances of a certain unit can be regarded as potential candidates for that unit. Thus, if full search is used, the concatenation cost calculation needs to be repeated for all the candidates and pairs. Due to this high number of repetitions, increases in the size of the database beyond the compact sizes of diphone synthesis databases quickly make the related calculations the dominant factor defining the overall computational load of the TTS system.

The second factor affecting the computational load is the cost function. Typically, the cost is computed separately for each vector element and then summed together over the vector length. Often, weighting is also applied. With this kind of cost functions, regardless of the details, the complexity is proportional to the dimension of the feature vector. As mentioned above, the dimension of the feature set or the feature vector used in the calculation is system-dependent. A typical example could be the feature set used in the practical experiments reported in this section that consisted of 16 elements, including 14 mel-frequency cepstral coefficients plus one value for the energy and one for the pitch. Thus, with this feature set and the cost function of weighted squared error, it can be concluded that the conventional concatenation cost computation for a single boundary between a single pair of candidate units requires 16 subtractions, 32 multiplications

and 16 additions. Since this needs to be repeated for all the concatenation boundaries of all or at least many of the potential candidate sequences, any reduction in the complexity can significantly lower the total computational load.

One additional aspect affecting the complexity is the format in which the feature vectors are stored and the cost of the related data access. With compressed databases, the speech data itself can be retrieved and decoded only after the unit sequence is decided. However, the feature data related to the boundaries of the units will be accessed frequently during the unit selection.

For each candidate unit, at least one acoustic feature vector has to be stored (one is usually enough since the same feature vector can represent the right boundary of the first unit and the left boundary of the second neighboring unit). To improve the efficiency of storage, it is possible to store this information in a compressed format. It is, for example, possible to use vector quantization techniques for compressing the feature vectors. The quantizer indices can then be used to represent the acoustic features. In the conventional concatenation cost calculation, these indices have to be decoded into the original feature domain at run time for measuring the cost. Even when the decoding is implemented using simple table-lookups, the decoding procedure requires additional processing effort. Thus, the basic compression or quantization based solutions reduce the memory needed for storing the feature vector data but increase the overall computational load.

In a typical TTS system, the speech database is fixed and does not change during run time. Consequently, it would be possible to pre-calculate the concatenation costs. Thus, one solution for reducing the run-time complexity would be to store a pre-calculated cost table covering the transitions between all the possible unit pairs. With large database sizes, this is not feasible due to the required table size. With compressed feature vectors, the size of the table could be smaller since the costs would only be stored for code vector pairs instead of feature vector pairs. However, considering the fact that the size of the codebook is typically not small, a large amount of memory would still be needed at run time. Thus, pre-calculating and storing a distance table for the whole codebook space is usually not feasible, unless the quantization is very coarse or the database is small.

4.3.3 Proposed concatenation cost calculation technique

The main idea of the proposed concatenation cost calculation technique is to compress the acoustic features and to approximate the distances based on the codebook indices. It is not possible to directly approximate concatenation costs based on the codebook indices but this is made feasible through the use of three separate ideas. In particular, the overall technique involves the combination of a specifically-trained multistage vector quantizer, pseudo-gray coding, and the use of weighted multistage Hamming distances. These aspects are discussed more thoroughly in the following.

Multistage vector quantization

As a part of the proposed approach, the feature vectors used for the calculation of the concatenation costs are compressed using multistage vector quantization. The concatenation cost function can be used as the distortion measure in the quantization. This makes the quantizer and the quantization results readily relevant from the viewpoint of concatenation cost calculation. As a further optional optimization step, if fixed weights are used for the different vector elements, the weights can be taken into account by scaling the corresponding feature values accordingly. This can enable, e.g., the use of the squared error instead of the weighted squared error as the cost function and distortion measure.

The multistage codebooks are trained on the feature vector data using the MSVQ simultaneous joint design algorithm [LeB93]. This design algorithm described in Section 2.3.2 jointly optimizes all the code vectors at all the stages in an iterative manner. A special aspect of the training is that the training can be performed specifically for a given TTS database, and thus the training set and the data set to be compressed are identical. Thus, it is possible to let the training run as long as the overall distortion decreases without any fear of over-training, and the overall distortion directly measures the accuracy of the compressed feature data representation.

There are two main reasons for the use of MSVQ and the particular training algorithm. Firstly, the important energy ordering property, mentioned in Section 2.3.2 and guaranteed by the use of the simultaneous joint design algorithm, makes direct comparison of the multistage indices partially feasible. For example, a unit pair having identical indices at the first stage at their concatenation boundaries is more likely to be cost-wise closer to each other than a pair with no identical indices or a pair having identical indices at the last stage but different indices at the first stage. Secondly, MSVQ also offers high precision with reasonable search complexity and codebook sizes, i.e., it allows efficient storage of the feature vector data.

Pseudo-gray coding

Pseudo-gray codes [Zeg90] have been traditionally used as protection when dealing with noisy channels. As a practical example, the training procedure introduced in Section 3.3.2 utilizes index assignment based on pseudo-gray codes as an optional step. A pseudo-gray code is an assignment of n-bit binary indices to 2^n points in a Euclidean space so that the Hamming distance [Ham50] between two points corresponds closely to their Euclidean distance. In other words, the binary indices are assigned to code vectors in a way that reduces the average quantization distortion introduced in the reproduced vectors when a transmitted index is corrupted by channel noise [Zeg90].

In the unit selection scenario, a large database practically always contains several candidate units that would be available for representing a certain speech unit. Even for a specified context, there are usually several instances that would be suitable or at least good enough to be used. Thus, obtaining high speech quality is often possible even when the selection fails at finding the truly optimal candidate sequence, as long as the chosen units are close to the desired target. Selecting among a set of "good" candidates can be thought to be analogous to transmitting indices through a noisy channel. Thus, the use of pseudo-gray coding, together with the previously-mentioned considerations regarding the distortion measure, could make the Hamming distances between the feature vector codebook indices to roughly correspond to the concatenation costs.

Under pseudo-gray coding, the code vectors are arranged in such a manner that any pair of code vectors whose indices have a low Hamming distance are on average close to each other in the original feature vector domain, too. The search for the optimal codebook rearrangement would in theory involve trying every possible index assignment. With the quantizer structures and sizes used in practical implementations, this task requires enormous computational complexity. Thus, binary switching algorithm (BSA) [Zeg90] is often used as a sub-optimal solution to the pseudo-gray coding problem. BSA involves iteratively switching the position of two code vectors to reduce the distortion. The choice of which pair of vectors to switch in the codebook at each iteration is determined by a heuristic ordering process. Each code vector is assigned a cost, and the code vectors are sorted in a decreasing order of their cost values. The vector with the largest cost is selected as a candidate to be switched first. If no code vector can decrease the distortion when switching it with any other code vector, the algorithm has reached a local optimum. With the proposed multistage quantization approach, the binary switching algorithm can be run separately at each stage, as a final step after the actual quantizer training.

Weighted Hamming distance as concatenation cost

After the quantization, the feature vector data for all the instances can be regarded as being converted into indices, i.e., each original feature vector $\mathbf{f}_j$ is represented using the corresponding index I_j. Even for feature vectors having a high dimension, the corresponding index can be represented using a small number of bits. Consequently, during run time, much less information has to be read. Naturally, the same is true for the storage space requirement. The traditional solution involving compressed feature data is to decode the feature data from the index domain to the original feature domain first. Let $\hat{\mathbf{f}}_j$ denote the reconstructed feature vector decoded from I_j. Thus,

$$\hat{\mathbf{f}}_j \approx \beta(I_j), \tag{4.3}$$

where β denotes decoding using the corresponding codebook $\mathbf{C}$ as described in Section 2.3.1. Then in the case where the features at the boundaries of all the units are compressed using above method, the concatenation cost could be

$$C_c = d_c(\underline{\mathbf{f}}_{j-1}, \mathbf{f}_j) \approx d_c(\hat{\underline{\mathbf{f}}}_{j-1}, \hat{\mathbf{f}}_j) = d_c(\beta(\underline{I}_{j-1}), \beta(I_j)). \tag{4.4}$$

After using the pseudo-gray coding approach to rearrange the codebook, the distance measurement can be performed directly in the index domain in an approximate manner. Thus, the concatenation cost becomes

$$C_c \approx d_c(\hat{\underline{\mathbf{f}}}_{j-1}, \hat{\mathbf{f}}_j) = d_c(\beta(\underline{I}_{j-1}), \beta(I_j)) \approx d_I(\underline{I}_{j-1}), \beta(I_j)), \tag{4.5}$$

where d_I denotes the distance measure operating in the index domain. The actual decoding procedure is not needed, and it is not even necessary to store the codebooks, if the only purpose is to use the proposed approach for the fast calculation of the concatenation costs.

When the multistage VQ is used, the concatenation cost can be approximated using the sum of the index distances at the different stages. The distances at the different stages can be weighted differently, i.e.,

$$C_c \approx d_c(\hat{\underline{\mathbf{f}}}_{j-1}, \hat{\mathbf{f}}_j) \approx \sum_{m=1}^{K} w_c^{(m)} d_I(\underline{I}_{j-1}^{(m)}), \beta(I_j^{(m)})), \tag{4.6}$$

where m denotes the stage, K is the total number of stages, and $w_c^{(m)}$ is the weight for mth stage. Due to the use of pseudo-gray coding, the Hamming distances between stage indices can be used to approximately calculate the distance between two vectors,

$$C_c \approx d_c(\hat{\underline{\mathbf{f}}}_{j-1}, \hat{\mathbf{f}}_j) \approx \sum_{m=1}^{K} w_c^{(m)} \mathrm{Ham}(\underline{I}_{j-1}^{(m)}), \beta(I_j^{(m)})), \tag{4.7}$$

where the operator Ham denotes the Hamming distance and the stage weights $w_c^{(1)}, w_c^{(2)} \ldots, w_c^{(K)}$ can be used for adjusting the distance scales at the different stages of the MSVQ. For example, due to the energy ordering property of the training algorithm described in Section 2.3.2, the weight of the first stage should be bigger than the weights at the latter stages. The Hamming distances cannot directly take this into account and thus the weighting scheme is needed. The weights can be optimized using the actual feature vector data as the training data, to ensure that the approximate similarity between the proposed concatenation cost and the original concatenation cost is maximized.

In the unit selection algorithm, the weight between concatenation cost and target cost can be re-adjusted when this new concatenation cost module is adapted. The reason for this is the fact that the overall scaling might be different and after

the proposed change the concatenation costs are only approximations instead of the exact measurements.

Depending on the implementation, it may be possible to simplify the above cost calculation even further in the implementation phase. In a big database, many instances exist for a certain unit. Thus, it is reasonable to assume that an acceptable instance should have a similar first stage code. Consequently, if the Hamming distance at the first stage is high enough, no further action at the other stages is needed: it is certain that the instance is not a good one. It should be noted, however, that this simplification may not be needed in practice since it is also possible to compute the total cost over all the stages simultaneously using precalculation.

4.3.4 Experiments

Experimental conditions

The main purpose of the experiments presented in this section is to quantify the effect that the use of the proposed approximate technique has on the quality of the synthesized speech. The new concatenation cost technique was implemented under the publicly available Festival text-to-speech framework [Cla04]. An implementation based on the original Festival was taken as the baseline, and the new concatenation cost calculation technique was implemented as described earlier in this section.

A speech database with the total duration of about 9 hours was used in the experiments. In the baseline system, 16-dimensional features were used for the calculating the concatenation costs. As mentioned before, each feature vector included 14 MFCCs, one value for the pitch, and one value for the energy. The baseline system was built using the multisyn method presented in [Cla07]. The target costs were calculated using context information, and the concatenation cost were calculated based on the 16 features.

The proposed system was similar to the baseline reference, except for the new concatenation cost calculation. The original concatenation cost function was used as the distortion measure in the MSVQ codebook training. A three-stage codebook was trained using the simultaneous joint design algorithm. Altogether 16 bits were used to quantize each of the 16-dimensional features vectors. The bit allocations for the three stages were 7 bits, 5 bits, and 4 bits. The weights for the three stages in Equation (4.7) were set empirically. Larger weights were set for the early stages, as suggested both by the energy ordering property of the training algorithm and the bit allocations.

Memory saving and reduction in computational complexity

For the 9-hour voice, the total number of units was about $330\,000$. Originally, 4 bytes were used for representing each feature element. Thus, the total memory

Table 4.2: Memory usage in kilobytes using the conventional uncompressed approach and the proposed approach.

	Uncompressed	Proposed
Memory usage (in kilobytes)	20 625	645

Table 4.3: Pair-wise comparison between the baseline and the proposed approach: the average score and the 95% confidence interval.

	Average score	Confidence interval
Baseline vs. Proposed	-0.014	0.1157

usage for the concatenation cost related features was roughly $330\,000 \times 16 \times 4$, i.e., more than 20 megabytes in the original uncompressed form. With the proposed approach, each instance needs two bytes for representing the entire feature vector. Thus, the related memory usage is about 645 kilobytes, as summarized in Table 4.2.

The computational complexity is also reduced significantly. In the baseline system, the computation involves 16-dimensional vectors, and for each element, the computation includes a subtraction, a multiplication and an addition, altogether at least 48 arithmetic operations, in addition to the requirement of reading all the 16 vector elements from the memory for each vector. For the new concatenation cost calculation, mostly only simple bit operations are needed after reading for each vector the single two-byte word that includes the MSVQ indices. The computation of a single weighted Hamming distance at one stage can be performed using only one *exclusive or* (XOR) operation and a simple table lookup. The exact computational load for the entire cost calculation procedure depends on the implementational details but in the simplest case, with the aid of a precalculated table, only one XOR operation and one table lookup can take care of the entire computation of the concatenation cost, over all the stages. In any case, both the data access needs and the computational complexity are drastically reduced.

Quality comparison

In order to evaluate the quality achievable using the proposed approach, a simple listening test was carried out. The test sentences were selected from texts related to news and novels. Altogether 12 listeners participated in the test including two parts, a pair comparison test and a mean opinion score (MOS) evaluation.

The pair comparison test was carried out to identify any obvious quality degradations caused by the use of the new concatenation cost calculation. The listeners

Table 4.4: MOS evaluation between the baseline and the proposed scheme, including 95% confidence intervals.

	Baseline	Proposed
MOS	3.333	3.375
Confidence interval	0.120	0.119

were asked to judge the relative quality level of the samples in each sample pair. The order of the samples was randomized, but in the results, the score of 1 was used in the cases where the baseline was considered better than the new method, the score -1 for the opposite case, and the score of 0 for the case where the quality levels were considered equal. Based on the results presented in Table 4.3, it is easy to conclude that the proposed concatenation cost approximately preserves the speech quality of the original method, despite the significant reductions in the memory requirements and in the computational load.

To further study the performance of the proposed technique, a MOS test was also carried out. Table 4.4 shows the MOS scores and the 95% confidence intervals for two systems. Again, it is easy to see that the proposed approach offers similar performance as the original concatenation cost calculation.

4.4 Conclusions

The VLBR-based concatenative synthesis discussed in this chapter offers a very efficient method for compressing text-to-speech databases, enabling the implementation of low-footprint concatenative TTS systems. In addition to the efficiency of compression, the proposed approach also facilitates the treatment of the unit boundaries and enables efficient parametric modifications. This chapter has also discussed the concept of dynamic quantizer structures, and the related topic of codec or quantizer retraining. The proposed approach enables flexible run-time quantizer updates and can be used for minimizing the bitrate needed for obtaining a given quality level or for maximizing the speech coder output quality for a given bitrate. Finally, a highly efficient technique for the calculation of the concatenation costs in unit selection based synthesis has also been proposed. The novel method, implementable using the implementations readily available in the VLBR codec, is based on the use of multistage codebooks optimized using the simultaneous joint design algorithm, pseudo-gray coding and weighted multistage Hamming distances.

The findings obtained through experiments with different TTS systems, languages and databases confirm that the VLBR codec can be successfully used for the compression of TTS databases and for generating speech output in low-footprint concatenative TTS systems. The best compression ratios can be obtained

by retraining the codec specifically for each unit selection database. The memory footprint and/or the computational load can be further reduced using the proposed method for concatenation cost computation, also in concatenative synthesizers that do not use the VLBR codec for the compression of the databases.

Chapter 5

VLBR-based voice conversion

The work presented in Chapter 3 enabled highly efficient compression of all types of clean speech signals, including, e.g., audiobooks, and in particular, as discussed in Chapter 4, the speech databases needed in unit selection based text-to-speech systems. Even though the framework was readily capable of handling speech modifications such as high-quality playback speed alterations, the speaker identity was preserved during the playback, and the creation of new TTS voices required recording and annotation of additional large TTS databases. Thus, the idea of including voice conversion functionality for cost-efficient creation of new voices was seen highly attractive.

As mentioned in Section 2.5, most of the existing voice conversion systems operate on parametric representations of speech. Parameterizations based on the sinusoidal modeling approach had also been studied before (e.g., in the classical paper [Sty98]), so it seemed likely that the VLBR codec's parameterization would be at least moderately suitable for voice conversion. What was not known, however, was how the simplifications of the parametric model presented in Section 3.1 and the quantizations would affect the outcome. To the author's knowledge, the combination of a very-low-bit-rate speech codec and voice conversion had not been studied earlier.

Section 5.1 presents the first version of a VLBR-based voice conversion system. This first attempt deals mostly with direct application of the Gaussian mixture modeling based conversion approach [Kai98] to the VLBR framework, and the main focus is placed on spectral conversion. Short-term spectral conversion addressed in the first section has also in general gained a lot of research interest but the conversion of prosodic features such as F_0 movements and speaking rhythm has been studied less actively. Section 5.2 introduces a novel method for prosody conversion that also enhances the naturalness of the output speech. The third and the fourth sections of this chapter present further enhancements to the VLBR-based voice conversion system. In Section 5.3, a novel approach for data clustering and mode selection is introduced. The proposed approach is shown to

enhance the conversion accuracy compared to the direct usage of voicing-based modes. Section 5.4 demonstrates that the voice conversion process causes unwanted changes to the perceived level of voicing and proposes a method for explicitly controlling the voicing level. The use of this technique makes the output of the VLBR-based voice conversion system less noisy.

The discussions provided in this chapter are mostly based on the publications [Hel07a], [Nur06d], [Nur06c], [Nur06e], [Nur07c], [Nur08a], and [Nur11a].

5.1 VLBR-based voice conversion system

The initial version of a VLBR based voice conversion system, originally published in [Nur06c], directly utilizes the VLBR codec presented in Chapter 3 and in particular the parametric model described in Section 3.1, and operates on parallel aligned training materials from the source and the target speakers. The conversion of the parameters is performed using the widely popular GMM-based approach [Kai98] that is generally considered the reference approach that new techniques are often compared against in the voice conversion literature. Like the vast majority of the published voice conversion techniques, the initial conversion system presented in this section focuses on the spectral aspects of conversion, including instantaneous pitch, and it does not modify the duration and timing related prosodic features or intonation contours at all. The prosodic aspects, however, were studied as a part of further research, and the most promising results are presented in Section 5.2. The unique feature of the voice conversion scheme presented in this section is that it also enables very efficient speech coding seamlessly within a single framework.

5.1.1 Training data alignment

The training of the GMM-based models utilizes aligned parametric data from the source and target voices. In the system presented and evaluated in this section, the alignment is achieved in two steps. First, both the source and the target speech signals are segmented to obtain coarse information about the alignment and then a fine-level alignment of the speech frames is performed within each segment. This two-step approach was selected to be able to conveniently use manually labeled phoneme boundaries if such information is available. Even though phoneme boundary information is sometimes readily available, for example for unit selection databases, the use of the manual labels is kept optional because the requirement for any manual processing would be impractical in many real-world applications. Manual labeling of phoneme boundaries is also time-consuming and prone to human errors.

In principle, the coarse first-level segmentation of the training speech data could be conducted using very simple techniques, for example by measuring

spectral changes without taking into account any knowledge about the underlying phoneme sequence. Also, it would be possible to use the segmentation provided by the VLBR codec. However, to make the segmentation process fully compatible with the approach of manual phoneme boundary labeling, it is convenient to exploit the information about the phonetic content and to perform the segmentation using HMM based models. This not only enables the combination of manual and automatic labeling, i.e., cases in which either the source speech or the target speech is manually labeled and the other one is automatically segmented, but also helps in achieving better performance. Naturally, this approach requires that the phonetic content of the training material is known.

The first step in HMM-based segmentation is to extract a sequence of feature vectors from the speech signal. The extraction is performed frame by frame, using similar frames as in the parameter estimation procedure described in Section 3.1. The features could consist of, e.g., VLBR parameters such as the LSFs but in practice it is better to use additional features such as mel-frequency cepstral coefficients. As mentioned above, the phoneme sequence associated with the corresponding speech is assumed known, and given this phoneme sequence, a compound HMM model is built up by sequentially concatenating the phoneme HMM models. Next, the frame-based feature vectors are associated with the states of the compound HMM model using Viterbi search to find the best path [Rab93]. By keeping track of the states, a backtracking procedure can be used to decode the maximum likelihood state sequence. The phoneme boundaries in time are then recovered by following the transition change from one phoneme HMM to another.

The phoneme-level alignment obtained using either pre-defined labels or the procedure above is further refined by performing frame-level alignment. In the first implementation discussed in this section, simple interpolation based alignment was used. Furthermore, both the simultaneous and the non-simultaneous silent segments were discarded from the training data.

5.1.2 Model training and the conversion function

The alignment procedure described in Section 5.1.1 results in a combination of aligned source and target vectors $\mathbf{z} = \left[\mathbf{x}^\top \mathbf{y}^\top\right]^\top$ that can be used to train a conversion model. In the work described in this section, the training was implemented using the popular approach proposed in [Kai98] that makes use of the aligned data $\mathbf{z}$ to estimate the GMM parameters $(\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ of the joint distribution $p(\mathbf{x}, \mathbf{y})$ [Kai98]. This is accomplished iteratively through the well-known *expectation maximization* (EM) algorithm [Dem77]. In the above notation, $\mathbf{x}$ and $\mathbf{y}$ correspond to the source and target feature vectors, respectively.

The conversion of the speech parameters follows a scheme where the trained GMM parameterizes a linear function that minimizes the mean squared error

(MSE) between the converted source and the target vectors. The conversion function is, as shown in [Kai98],

$$F(\mathbf{x}) = E(\mathbf{y}|\mathbf{x}) = \sum_{m=1}^{G} p_m(\mathbf{x}) \left[\boldsymbol{\mu}_m^{(y)} + \boldsymbol{\Sigma}_m^{(yx)} \left(\boldsymbol{\Sigma}_m^{(xx)} \right)^{-1} \left(\mathbf{x} - \boldsymbol{\mu}_m^{(x)} \right) \right], \quad (5.1)$$

where

$$p_m(\mathbf{x}) = \frac{\alpha_m \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_m^{(x)}, \boldsymbol{\Sigma}_m^{(xx)} \right)}{\sum_{j=1}^{G} \alpha_j \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}_j^{(x)}, \boldsymbol{\Sigma}_j^{(xx)} \right)}. \quad (5.2)$$

The symbol G denotes the number of Gaussian components used in the modeling and α_m denotes the prior probability of the mth Gaussian component. The covariance matrices are formed as

$$\boldsymbol{\Sigma}_m = \begin{bmatrix} \boldsymbol{\Sigma}_m^{(xx)} & \boldsymbol{\Sigma}_m^{(xy)} \\ \boldsymbol{\Sigma}_m^{(yx)} & \boldsymbol{\Sigma}_m^{(yy)} \end{bmatrix}, \quad (5.3)$$

and

$$\boldsymbol{\mu}_m = \begin{bmatrix} \boldsymbol{\mu}_m^{(x)} \\ \boldsymbol{\mu}_m^{(y)} \end{bmatrix} \quad (5.4)$$

is the mean vector of the mth Gaussian component of the GMM.

5.1.3 Conversion of the VLBR parameters

In the development of the first version of a VLBR-based voice conversion system, the emphasis was placed on the conversion of the pitch and the LSFs because these parameters were found in the first experiments to be particularly important from the perception point of view. Other parameters such as voicing and the residual spectrum were partially used as complementary information and were exploited in the model training but no explicit conversion was performed for these parameters.

The conversion of the LSF vectors is performed using an extended vector that also contains the derivative of the LSF vector, to take some dynamic context information into account. This combined feature vector is transformed through GMM modeling, using Equation (5.1). Only the true LSF part is retained after conversion. The conversion utilizes several modes, each containing its own GMM model with 8 Gaussian components. The number of components was selected based on practical experimentation. In the first implementation described in this section, the modes were decided in a data-driven manner based on the voicing parameter, i.e., the LSF data is clustered during the model training into separate sets using the corresponding voicing information, and similar voicing-based mode selections are used during the conversion phase. The motivation for using voicing-based modes is similar as in the case of the segmental speech coding approach

presented in Section 3.2, i.e., different types of speech signals typically benefit from different type of processing. Also, because the VLBR codec already operates on different segment types, the same voicing-based segmentation decisions can be directly used in VLBR-based voice conversion.

The pitch parameter is transformed through the associated GMM in the frequency domain using Equation (5.1). During unvoiced parts, the fixed pitch value is left unchanged. The GMM with 8 Gaussian components used for the pitch conversion is trained on aligned data, with the additional requirement of having matched voicing between the source and the target data.

After the conversion of the pitch parameter, the residual amplitude spectrum is processed accordingly. The reason for this processing is the fact that the length of the amplitude spectrum vector depends on the pitch value at the corresponding time instant, as discussed earlier in this thesis. This means that the residual spectrum, although essentially unchanged, will be re-sampled to fit the dimension dictated by the converted pitch at that time.

Once the parameters have been converted as described above, they are used together to re-synthesize the transformed waveform. The signal generation part of the VLBR decoder can be used as such for synthesizing the waveform in a pitch-synchronous manner.

5.1.4 Performance evaluation

The initial VLBR-based voice conversion system described in this section was evaluated in listening tests in the context of the second TC-STAR [TC-13] evaluation campaign. The evaluation covered aspects related to both speaker identity and speech quality. The evaluation was carried out by an independent evaluation agency.

Test set-up

The data set used in the testing included UK English speech data from four different speakers (two female and two male speakers). The training set included 159 sentences per speaker and a distinct testing set consisted of 9 sentences per speaker. The same sentences were recorded from all the speakers.

Among the 12 possible conversion directions, 4 were chosen as the directions included in the test. For the selected directions, the test organizer provided the recorded source sentences used in the test. These source sentences were converted using the voice conversion system to the voices of the target speakers. The converted signals were evaluated by 20 native non-expert listeners.

The listening test included two parts. In the first part, the listeners were asked to evaluate the speaker identity without considering the speech quality using the 5-level scale summarized in Table 5.1. The true target signals recorded from the target speakers, available only for the test organizer, were used as the reference.

Table 5.1: Scale used for evaluation of speaker identity. The listeners were asked to evaluate whether the two samples in the given pair were spoken by the same person or not. The real target speaker was used as the reference speaker.

Grade	Meaning
5	Definitely identical
4	Probably identical
3	Not sure
2	Probably different
1	Definitely different

Table 5.2: Scale used in the evaluation of speech quality

Grade	Meaning
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

In the second part, the listeners evaluated the perceptual quality of the converted speech using the mean opinion score (MOS) grades shown in Table 5.2.

Results and discussion

The results are summarized in Table 5.3 and Table 5.4. Table 5.3 contains the results from the first part of the listening test, focusing on the evaluation of speaker identity. The results from the speech quality evaluation are summarized in Table 5.4.

When looking at the evaluation results, the first observation that can be made is that there were large differences between the different conversion directions. Moreover, despite the moderate average scores, the person identity conversion was

Table 5.3: Results from the first part of the evaluation (speaker identity, with the target speaker used as the reference in every sample pair). F denotes a female and M a male speaker. The column *Average* shows the combined score for all the directions.

Direction	F_1 to F_2	F_1 to M_2	M_1 to F_2	M_1 to M_2	Average
Score	3.10	3.05	2.20	1.77	2.53

Table 5.4: Results achieved from the second part of the evaluation (speech quality).

	MOS score
Achieved score	2.09
Reference 1 (source)	4.80
Reference 2 (target)	4.78

sometimes perceived very successful, as indicated by the more detailed sentence-level results not shown here. This can be regarded as a good result due to two main reasons. First, the initial system that participated in the evaluation was a rather elementary system that only converted the LSFs and the pitch parameter. Moreover, the conversion was performed in a frame-wise manner without considering the frame-to-frame evolvement of the parameters or the intonation contours.

As can be seen from Table 5.4, a rather low score was achieved in the speech quality evaluation. There are a couple of clear reasons for this. First, the system produced 8-kHz output signals while all the other signals included in the listening test (e.g., the reference samples and the samples from the other TC-STAR participants) had a sampling rate of 16 kHz. Second, the source signals also contained some non-speech elements such as audible breathing and the parametric speech and conversion models created many audible artifacts to the corresponding places in the output signals. Third, the frame-by-frame conversion made the converted parameter contours, including the pitch contours, a bit noisy and this was also audible in the output signals. Also, it is known that the GMM-based conversion approach has its shortcomings related to overfitting and oversmoothing, as discussed, e.g., in [Nur12]. Finally, the fact that not all the parameters were converted also had its impact on the quality.

It should also be noted that the use of the simple interpolation based alignment had a small negative impact on the output quality. Furthermore, it was also found later on, as a part of the research whose main findings were reported in [Hel08b], that the use of the two-step alignment procedure described in Section 5.1.1 is unnecessary and sometimes even counterproductive due to the sometimes erroneous phoneme boundary locations that limit the frame-level alignment. The main outcome of the study [Hel08b] was that even though the alignment of the training data significantly affects the voice conversion quality, it is possible to obtain the same quality as with hand-marked labels using only simple voice activity detection, *dynamic time warping* (DTW) and certain additional considerations. This effectively renders the phoneme boundary information unnecessary at best, and, as mentioned above, detrimental at worst.

Several techniques that enhance the performance of the initial VLBR-based system are presented in Sections 5.2–5.4. In those discussions, the alignment is

handled using the dynamic time warping based approach but otherwise the system presented in this section is used as the baseline voice conversion system.

5.2 Prosody conversion

Speech prosody is affected not only by the content of a given sentence but also by speaker-specific variations. For the same sentence, different people generally produce different prosody. This was studied as a preparatory step before starting the main work described in this section. The results, published in [Hel07b], revealed that even the "pure prosody", represented using a delexicalized signal consisting of only one sinusoid whose energy and frequency follow the energy and F_0 levels of the original speech signal, contains sufficient amounts of speaker-specific information that enables listeners to successfully identify people that they know. Moreover, it was clear that the identification did not happen based on average F_0 levels alone since the result was valid even when the identification was done between speakers of the same gender having almost identical average F_0 levels. Similar findings confirming the importance of the prosodic information were obtained in a recent study [Mor12]. Some earlier studies, such as [Rem97] and [Fel97], have also confirmed that listeners can identify people they know based on sinewave signals but in those studies the signals contained 3-4 sinewaves related to the formants.

Since speech prosody was confirmed to contain acoustic cues related to speaker identity but there were only a few published studies that addressed the topic of prosody conversion, the next step of the work was to develop a method for converting the prosodic aspects of speech. The outcome of the work was the novel prosody conversion method published originally in [Nur11a] and [Hel07a]. The work was carried out in the context of VLBR-based voice conversion but the solution is applicable to other types of voice conversion systems as well.

The goal in the work described in this section was to obtain high-quality prosody conversion results with only a relatively small set of training data. To make the goal tangible, a few clarifications are needed. First, even though the studies mentioned in the beginning of this section confirm that the prosody contains speaker-specific information, it is also true that the same person can utter the same sentence with quite different prosodies if her/his speech is recorded several times. Due to this reason, the goal was the generation of "believable" prosody, i.e., prosody that the target speaker could use in a certain context. The second remark concerns the fact that the prosody models used in TTS systems are usually trained using a very large database (consisting of, e.g., 10 hours of speech). In voice conversion, the size of the training set has to be much smaller, in the order of some minutes of speech, e.g., to enable online training of new conversion models. Consequently, the aim in this work is not to build a sophisticated prosody model for conversion purposes but only to capture some major tendencies from the data.

In the method introduced in this section, a syllable-level codebook containing paired source and target F_0 contours is built. The F_0 information is compressed using discrete cosine transform coefficients that enable fast comparison and make it possible to avoid the use of dynamic time warping based techniques for alignment. The codebook source contours that are close enough to the source contour under conversion are regarded as candidates and the final decision is made based on a *classification and regression tree* (CART) trained on linguistic and durational features. The performance of the method is evaluated in a listening test.

5.2.1 Conventional methods for prosody conversion

Most of the publications in the area of voice conversion (e.g., [Kai01] and [Err07], among many other publications) neglect detailed prosodic modeling and simply adjust the F_0 level and range of every source F_0 value $f^{(x)}$ to match those of the target using

$$f^{(y)} = \frac{f^{(x)} - \mu^{(x)}}{\sigma^{(x)}}\sigma^{(y)} + \mu^{(y)}, \tag{5.5}$$

where $\mu^{(x)}$, $\sigma^{(x)}$, $\mu^{(y)}$, and $\sigma^{(y)}$ represent mean and standard deviation of the F_0 values for the source and the target, respectively. Typically, this approach, referred to in this section as the *mean-variance* (MV) scaling method, is used in logarithmic domain. The MV method retains the shapes of source F_0 contours and cannot model local changes in F_0.

A more sophisticated mapping function can be obtained by representing pitch values with Gaussian mixture models as described in Section 5.1. This approach essentially utilizes a weighted sum of different mapping functions. The resulting GMM based pitch prediction approach serves as the reference approach in the listening test presented in this section.

There are not many proposals in the literature for converting the F_0 contours or other prosodic features in more detail. Nevertheless, [Cha98a] describes three F_0 prediction methods: the MV method presented in Equation (5.5), a similar approach but using a cubic fit to the data, and a third method based on an utterance level codebook. These three methods and a voiced contour codebook were evaluated in [Ina03]. A closely-related idea of voiced contour codebook was also presented in [Tur03] but instead of picking only one contour like in [Ina03], the resulting contour was formed by using a weighted average of all contours according to the distance between a codebook source contour and the voiced contour to be transformed. In [Gil03], sentence-initial and sentence-final F_0 values as well as hand-marked accents were modeled using separate means and standard deviations.

Since the publication of the proposed approach, two more recent prosody conversion techniques have been introduced, both designed for use with non-parallel training data. The first one presented in [Lol08] involves the use of maximum

likelihood linear regression for obtaining syllable-level conversion of F_0 and duration. The second technique introduced in [Wu10] uses the combination of text-independent alignment, GMM modeling of the F_0 values and the delta F_0 features, and histogram equalization to perform the F_0 conversion.

5.2.2 Overview of the proposed method

Firstly, a decision was made to model prosody at the syllable level. Syllables are in principle linguistically determined units but the use of syllable-level processing can also be considered prosodically justified. The reason for this is that prosodic events take place in synchrony with syllables or groups of syllables. For example, the tone sequence theory on intonation modeling studies prosodic events and tones that cover a single syllable or group of syllables [Dut97, Chapter 6]. Another benefit of syllable-level operation is the related robustness. According to the sonority principle in English [Rad99] (that is also applicable to many other languages), the syllable nucleus constitutes a sonority peak. In practice, the level of voicing behaves in a roughly similar manner and thus it is relatively easy to capture continuous syllable F_0 contours for the codebook using the VLBR codec's parametric model introduced in Section 3.1.

The prosodic codebook is generated by first collecting syllable-aligned F_0 contours using parallel training sentences from the source and the target speakers. The F_0 contours are transformed into DCT coefficients and this information is stored in the codebook as source-target vector pairs. Linguistic and durational information is also stored for each entry. As a second part of the training process, a CART is trained using the codebook as the training data. The role of the CART is to help in the selection process.

During conversion, the source contour to be converted is first transformed into DCT domain and compared to the codebook keys (the source contour sides of the contour pairs). The codebook entries whose source contours are close enough to the contour to be converted are chosen as candidates for the final selection that is performed using the CART. The target contour side of the selected entry is inverse discrete cosine transformed and given a new F_0 level value based on the mean level of the current syllable predicted with the MV method. These steps are described in more detail in the following subsections.

The F_0 contours stored in the codebook are actual contours estimated directly from the speech data without any clustering or averaging of contours. In addition, only one target contour is chosen as the output in order to avoid the risk of obtaining a flat contour as a result of weighted averaging. The linguistic context is taken into account in the selection to avoid producing linguistically incorrect contours.

5.2.3 Model training and usage

Similarly as in the case of the other voice conversion models discussed in this chapter, the proposed model has to be trained using a training data set before being able to actually use the model in conversion. This section addresses issues related both the training phase and the usage. The discussion of the model training is split into two parts. The first of them describes the codebook creation process and the second one deals with the topic of CART training. The last part of this subsection introduces the actual prosody conversion process.

Codebook creation

The first step in the codebook creation process is to obtain the syllable-length F_0 contours using a pitch estimation algorithm and parallel training materials from the source and the target speakers, including information on the boundary locations and the linguistic content. These pieces of information are typically readily available for the recorded sentences stored in TTS databases. In the main target application for the proposed approach, involving the conversion of TTS voices to cost-efficiently generate new voices, a set of unit selection database sentences can be used as the source side training material. Then, it is sufficient to only record the target material where the target speaker utters the same sentences, and to use alignment to be able to use the same boundary labels and linguistic information for the target materials. During conversion, similar linguistic information can be obtained from the TTS front end. Thus, in the target application, the use of the linguistic data does not typically require any additional annotation work or training of new models.

After the initial pitch/F_0 estimation, the resulting source and target contours of each syllable can be further smoothed, if necessary, and possible F_0 outliers at the syllable boundaries can be removed. The syllables containing voiced contours that have a duration too short for meaningful contour representation are discarded. For all the other syllables, the process is continued by applying DCT on the contours.

The main motivations behind the use of DCT are exactly the same as in the case of update rate and vector dimension conversions discussed in Section 3.2.4. The implementations developed for the vector dimension conversions can be directly reused in the generation of the pitch contour codebook. The DCT-domain truncations or zero-paddings and the related normalizations are exactly the same as in Section 3.2.4, and the contours are stored as fixed-dimension DCT vectors. One issue worth noticing is that the first DCT coefficient does not have to be stored in the codebook since it represents the mean F_0 level that is handled separately.

In addition to the DCT domain contours, simple linguistic information and durational features are stored in the codebook for each entry. The set of linguistic information can be decided based on the application scenario. When the proposed prosody conversion is used in a TTS system, many kinds of linguistic informa-

tion is readily available without training specific models. In the implementation evaluated in this section, the feature set consisted of features that can be easily obtained from the TTS system and that have also been popular data-driven prosody generation techniques. In particular, the following items were included: lexical stress, local position in the word {*initial*, *mid*, *final*, *monosyllabic*}, global position in the phrase {*initial*, *final*, *first in a prosodic phrase* (predicted using simple punctuation rules), *none*}, Van Santen-Hirschberg classification for onset as well as coda {*unvoiced*, *voiced but no sonorants*, *sonorant*}, and the type of the word the syllable belongs to {*content*, *function*}. In addition to the linguistic information related to a specific syllable, the information related to the previous and the next syllable can also be taken into account. As the duration related features, the total duration of the syllable for the source and the target, respectively, and the duration of the F_0 contour of the source and the target, respectively, were stored in the codebook for each entry. The duration of the syllable and the duration of the F_0 contour were typically different since the unvoiced frames for which the VLBR codec's pitch estimator gives a fixed pitch value were excluded from the pitch contours.

CART training

In the training of the CART, the design goal is to build a tree that can output an optimality score based on the linguistic and durational similarity. The process begins with the generation of the training data. As a preliminary step, two distance matrices are computed based on the codebook. The elements of a source-side distance matrix $\mathbf{R}^{(x)}$ are computed as

$$r_{jm}^{(x)} = (\mathbf{h}_j^{(x)} - \mathbf{h}_m^{(x)})^\top (\mathbf{h}_j^{(x)} - \mathbf{h}_m^{(x)}) \quad j, m = 1, 2, \ldots, S, \tag{5.6}$$

where $\mathbf{h}_j^{(x)}$ is the fixed-length DCT-domain vector corresponding to the jth source-side pitch contour stored in the codebook and S denotes the total number of syllable-sized contour pairs in the codebook. As can be seen from the equation, the element $r_{jm}^{(x)}$ gives the squared distance between the source contours j and m. A similar distance matrix $\mathbf{R}^{(y)}$ is computed using

$$r_{jm}^{(y)} = (\mathbf{h}_j^{(y)} - \mathbf{h}_m^{(y)})^\top (\mathbf{h}_j^{(y)} - \mathbf{h}_m^{(y)}) \quad j, m = 1, 2, \ldots, S, \tag{5.7}$$

for the DCT-domain target contours $\mathbf{h}_j^{(y)}$.

As the next step, the actual training data is formed from the codebook data as follows. All the entries in the codebook are taken into consideration, one by one. For the jth entry, this means that the source contour of this entry is compared against the source contours of the other entries based on the elements of matrix $\mathbf{R}^{(x)}$ from $r_{j1}^{(x)}$ to $r_{jS}^{(x)}$ except for $r_{jj}^{(x)}$. If $r_{jm}^{(x)}$ is below a certain threshold, i.e., a

$r_{jm}^{(x)} < \tau_j$, the corresponding entry m is considered a potential candidate for being a good substitute for the entry j. In the implementation evaluated in this section, the threshold τ_j was made adaptive on the source contour of the entry j in such a way that a certain percentage deviation from the closest match was allowed in terms of contour distance.

For each of the potential candidates, the corresponding target distance $r_{jm}^{(y)}$ is obtained. Based on $r_{jm}^{(y)}$, the entry m is considered either a *possibly optimal*, a *neutral* or a *non-optimal* candidate as an substitute for the entry j. The codebook entries having a distance below an experimentally tuned threshold κ_o are considered possibly optimal choices and the entries having a distance above a second experimentally set threshold κ_n represent the non-optimal case. The neutral cases having a distance between these thresholds are not used in the training since they fall into an uncertain region. For the possibly optimal and non-optimal entries, the linguistic information is compared against the linguistic information of the entry j, resulting in a binary vector. In the binary vector, each zero means that there was a match in the corresponding feature (for example both entries, m and j, were monosyllabic), while the value one means that the corresponding features were not the same. In addition to the binary distances, the absolute differences of the syllable durations and the F_0 contour durations are also computed and stored for usage as the training data. After repeating the above procedure for all the entries in the codebook, the generated training data consists of a reasonably large amount of data from the two classes (*possibly optimal* and *non-optimal*) with the corresponding linguistic and durational information.

The actual training of the classification and regression tree aims at finding which features are important in the final candidate selection. There can be many codebook entries that have quite similar source contours but clearly different target contours, and thus finding out how much the duration and the context affect the situation is important. In the training of the CART used in the prosody conversion model evaluated in this section, a CART with Gini impurity measure [Dud01] was used. The CART was pruned according to the results of 10-fold cross-validation in order to prevent over-fitting and the terminal nodes were pruned if they ended up having only a small number of observations.

Conversion of F_0 contours and durations

The conversion process starts with the detection of syllable boundaries. When used in a TTS system, this information is readily available from the TTS front end. Next, a syllable-length F_0 source contour to be converted is formed. Again, the unvoiced and silent portions are ignored. Once the F_0 contour is available for processing, the discrete cosine transform is applied and the resulting vector is zero-padded or truncated to a fixed length and normalized similarly as in the codebook generation phase.

For the syllables that do not contain sufficiently many F_0 values for obtaining a meaningful contour representation, the MV scaling method of Equation (5.5) is used for the F_0 prediction. Otherwise, the process starts similarly as in the training: Some codebook entries become potential candidates based on the small-enough difference between the source contours. The computation is performed similarly as in Equation (5.6) but now instead of calculating the squared distance between different entries stored in the codebook, the squared distance is computed between the source contour to be converted and the source-side of the different entries stored in the codebook.

The threshold for accepting candidates is determined based on the smallest difference, allowing again a certain percentage deviation, similarly as in the training phase. If the adaptively calculated threshold is above a pre-specified limit, indicating a poor match, the MV scaling method is used for converting the F_0 contour. In all other cases, the linguistic information between the syllable whose F_0 contour is to be converted and the candidates is matched, resulting in a binary vector similarly as in the training phase. In addition, the absolute differences in the syllable duration as well as in the F_0 contour duration are calculated. This information is used as an input to the CART, and the candidate leading to the tree node producing the highest probability for the possibly optimal class is chosen as the selected codebook entry. If there are two or more candidates producing the highest probability, the candidate whose source-side contour's difference to the contour to be converted is the smallest is selected.

After selecting the most appropriate entry from the codebook using the CART as described above, the final contour is produced by taking the inverse DCT of the corresponding target contour. The length in the DCT domain is zero-padded or truncated to match the length of the F_0 contour to be converted, together with appropriate scaling in order to obtain a contour having the correct length (the possible duration change is handled separately, in the example implementation using the playback speed alteration technique presented in Section 3.1.3). Next, the mean F_0 level is added to the contour.

If the original F_0 contour is continuous across the boundary of two syllables, the converted contours are also made continuous by adding a bias value to the second syllable. The bias is determined as the difference between the last point of the first syllable and the first point of the second syllable. Since this can result in major changes in the standard deviation of F_0 calculated over the two syllables, the standard deviation is scaled back to the level where it was before the change. In addition, the F_0 level is also set again for both of the syllables, now calculated jointly.

Conventionally, the durations are either left unconverted or they are modeled using simple utterance-level scaling. In the proposed conversion technique, the durations are converted through syllable-level scaling using regression coefficients calculated from all the source and target syllable durations. This results in more

detailed modifications than the simple utterance-level scaling. Alternatively, the duration scaling ratios could be predicted by building a CART using the linguistic features. A third alternative would be to use directly the target syllable duration that corresponds to the chosen index. As mentioned above, in the implementation evaluated in this section, the durations are modified using the playback speed alteration technique presented in Section 3.1.3

5.2.4 Performance evaluation

The proposed prosody conversion technique was implemented and integrated into the VLBR-based voice conversion system presented in Section 5.1. The performance of the technique was also evaluated in this context. Even though the method is mainly designed for use in unit selection based text-to-speech systems, the testing was carried out with recorded sentences instead of synthesized sentences to ensure that the synthesis process does not affect the results.

How to evaluate prosody conversion?

It is not straightforward to evaluate a prosody conversion technique. There are no generally accepted objective measures for evaluating prosody conversion so the only choice is to organize a listening test but what should be evaluated and how? In the literature, no evaluations were carried in [Cey02] nor in [Cha98a]. In [Ina03], the converted pitch was transplanted to the real target utterance using dynamic time warping. Although the intention to prevent the spectral conversion from affecting the result is logical, tentative experiments with this approach indicated that it is not easy for the listeners to notice the prosodic differences. In addition to the F_0 contours, there are many other prosodic aspects (e.g., durations and prosodic voice quality) that remain unchanged with this approach and the real differences can be difficult to hear.

In [Tur03], better prosodic modeling improved the similarity to the target in a real voice conversion system but the confidence score and the quality score decreased. A sophisticated voice conversion system should retain its quality regardless of whether the conventional MV scaling method or some more advanced approach is used, and thus it was decided that it is the best to evaluate the proposed prosody conversion in connection with the spectral conversion, i.e., in the VLBR-based voice conversion system. Moreover, as discussed in the beginning of this section, there are no strictly right or wrong F_0 contours for the target speaker, the goal should be to achieve acceptable and believable prosody, and this should be reflected in the listening test.

Experimental set-up

As mentioned above, the experiments were carried out using the VLBR-based voice conversion system described in Section 5.1. The language used in the experiments was US English. A female voice recorded for TTS purposes served as the source database and several matching sentences were collected from a male speaker. This target speaker was allowed to speak more freely from the prosodic point of view.

An interesting observation related to the voices used in the test is that the mean F_0 level for the source (female) was 176 Hz and for the target (male) 118 Hz and standard deviations were 18.1 Hz and 15.5 Hz, respectively. However, the mean syllable-wise standard deviation of the syllables used in the codebook were 6.7 Hz and 7.1 Hz for the source and the target, respectively. Thus, it is straightforward to see that simple global modifications of standard deviation do not produce optimal results.

The performance of the proposed approach was compared against the performance of the GMM-based pitch conversion model used in Section 5.1. Since the GMM-based model and cubic conversion functions were reported in [Ina03] to result in quite similar performance as the MV scaling method, the most sophisticated approach of these, i.e. the GMM-based technique, was chosen for the experiment. This conventional pitch conversion model was implemented using GMM-based modeling with 8 Gaussian components.

A training set of 90 parallel sentences was used for the training of both the conversion models of the VLBR-based conversion system and the proposed prosody conversion approach. A set of 25 sentences, not included in the training set, was used for testing. F_0 was measured at 10-ms intervals and 8 DCT coefficients were used to represent the contour in the transformed domain.

The converted F_0 values mimicking the target F_0 were generated using the two techniques, the GMM-based modeling and the proposed approach. The spectral part of the conversion was handled in both cases using identical models and techniques. With the GMM-based method, the durations were not modified as the utterance-level scaling factors were extremely close to 1 for all the test sentences. With the proposed method, the durations were modified using the proposed syllable-level scaling. An interesting observation was that at the syllable level, 22% of the syllable instances had a scaling ratio falling outside of the range from 0.9 to 1.15.

Test arrangement

Altogether 19 listeners participated in the test. Nativeness was not required as the test was designed in such a way that also non-native listeners with good English skills can easily judge the relevant issues from the speech samples. The experiment contained two parts, referred to as Test 1 and Test 2. In addition, at the

Table 5.5: Preference votes given to the proposed approach and to the GMM-based approach, and the "no preference" votes (equal).

Method	Proposed	GMM	Equal
Test 1	67.0% (318)	22.7% (108)	10.3% (49)
Test 2	70.3% (334)	17.1% (81)	12.6% (60)

beginning of the test, the subjects were asked to listen to several speech samples from the real target speaker (not including the test sentences) and to pay special attention to the speaking style.

In the first part of the test, the listeners heard two versions of the sentences, in which the prosody was converted using the two different techniques, the GMM approach and the proposed approach. The listeners were asked to choose the sample that best mimicked the target speaker's speaking style. They were guided to choose the sample whose prosody could be closer to the prosody that the target speaker could use. They were asked not to care about quality of the spectral conversion. The subjects could also choose "equal" and it was possible to listen to the samples as many times as necessary.

The VLBR-based voice conversion system was found to lead to somewhat robotic voice quality in the experiments described in Section 5.1. The impact that the prosody may have to this phenomenon was studied in the second part of the listening test. The same sentences were played again and the listeners were asked to indicate which sample sounded less robotic. Again, it was possible to respond that the samples were equally robotic.

Results

The percentages of preference votes that the two methods received as well as the total number of votes are shown in Table 5.5 for both Test 1 and Test 2. In the first part of the test (Test 1), the results clearly indicate that the proposed approach was found to achieve better prosody conversion than the GMM-based approach. In the second part (Test 2), the proposed technique was found to contribute to the voice quality by making it less robotic. According to a two-tailed t-test, there was a significant difference between the performances of the proposed method and the GMM method ($p = 2.9 \times 10^{-14}$) for Test 1. Since there was also the third alternative of samples being equally good, the performance of the proposed method was also compared against the summed votes of both the equal choice and the GMM method votes. The results were still very clearly statistically significant ($p = 9.8 \times 10^{-10}$). For Test 2, a similar analysis was performed and the results were also clearly significant ($p = 7.3 \times 10^{-19}$ and $p = 2.3 \times 10^{-13}$) for the proposed approach to sound less robotic.

5.3 Data clustering and mode selection

One of the common challenges in several areas of speech processing research is caused by the complexity of speech signals. The signals are generated through complicated speech production mechanisms and consequently the signals possess highly variable statistical properties. Moreover, each person has their own unique physical properties related to speech production. As a result, it is very challenging to develop speech processing techniques that would perform consistently on all input speech signals. For example, nearly stationary voiced regions should usually be treated differently than plosives. Numerous different approaches have been proposed to tackle this problem in different areas of speech processing research, usually based on some heuristic solutions. Nevertheless, no universal solution exists for the problem.

Voice conversion is one of the speech related research topics in which the signal processing techniques have to operate with different kinds of speech signals and speech signal regions. Different techniques for the conversion itself are discussed elsewhere in this chapter as well as in Section 2.5. For the particular problem of handling different types of speech segments, the solutions proposed in the literature include the use of acoustic similarity based classification and regression trees [Dux04], phoneme-tied codebooks [Kan05], K-means based clustering [Sün04a], and phoneme-based modeling [Kum03].

This section introduces a novel solution for handling different types of speech segments in a completely data-driven way using different modes, originally published in [Nur06e]. When applied in voice conversion, the proposed approach for data clustering and mode selection is based on the idea that for the training of the multiple processing schemes the data is split into different clusters using clustering on the target data. In this way, the intra-cluster behavior of the data becomes easy to model. For the mode selection, a different approach has to be used since the target data itself is not available. The solution proposed in this section is to train a classifier that aims to recognize the correct target based cluster using only source-related data features.

5.3.1 Proposed approach for data clustering and mode selection

The proposed approach for data clustering and mode selection starts from the idea that the data should preferably be clustered into different operating modes in the most relevant manner from the viewpoint of effective processing in the particular application. In the case of voice conversion, the most relevant clustering would be based on data that is not yet available during usage, i.e., since the target is to minimize the potential conversion error, the most effective approach would be to cluster the combined data from the source and the target side into different processing modes based on the target data. This choice ensures a minimized potential conversion error within each mode or cluster. However, the corresponding mode

selections would not be possible during usage since the target data is not available at that time. Nevertheless, in the proposed approach, the training data is initially clustered using the most relevant clustering approach. Then, the next step is to train a mode selector that aims at finding the correct cluster based on the data that is available during usage. This data can include in addition to the conventionally available data any auxiliary features that can be made available. Finally, a separate processing scheme is trained and used for each mode.

Training

When applied in the voice conversion task, the proposed approach first finds B clusters solely based on the target data features $\mathbf{y}$. For example, if the aim is to convert vectors containing line spectral frequency data, the initial clustering is performed based on target LSF vectors only. The clustering can be performed, e.g., using the well-known K-means algorithm to obtain the clusters $\mathbf{y}^{(1)}$, $\mathbf{y}^{(2)}$, $\ldots$, $\mathbf{y}^{(B)}$.

After obtaining the initial grouping, the next step is to train a mode selector with the aim to recognize the target based clusters using only data from the source speaker. To facilitate the classification task, auxiliary features derived from the source data can be used in addition to the source vectors $\mathbf{x}$. In principle, this auxiliary data denoted as $\mathbf{o}$ can include any/all the features that one can extract from the source data. For example, the auxiliary feature set could include acoustic parameters such as pitch, voicing and energy as well as other parameters such as phoneme information, linguistic location, linguistic duration and part-of-speech. Given the initial target-based clusters, the extended aligned data set, denoted now as $\mathbf{z} = \left[\mathbf{x}^\top \mathbf{o}^\top \mathbf{y}^\top\right]^\top$, can be straightforwardly split into the same B groups, $\mathbf{z}^{(1)}$, $\mathbf{z}^{(2)}$, $\ldots$, $\mathbf{z}^{(B)}$. Based on this grouping, it is possible to train a classifier aiming to find the correct cluster using only the source related data vector $\left[\mathbf{x}^\top \mathbf{o}^\top\right]^\top$. In the experiments described in this section, the classifier was implemented using a simple linear discriminative function $D\left(\left[\mathbf{x}^\top \mathbf{o}^\top\right]^\top\right)$ but it would also be possible to use other techniques such as non-linear discriminative functions, neural networks or support vector machines. The exact selection of the auxiliary feature set is not a highly critical issue in the sense that the features with no additional discriminative information will receive a very low or even a zero weight in the training while the more relevant features will receive a larger weight.

Once the mode selector or the classifier is available, a separate conversion scheme is trained for every mode with a training data set belonging to that mode. It is possible to either use the training data sets based on the initial clustering that was made based only on the target data or to re-cluster the data using the trained classifier to obtain re-grouped training data sets. The latter approach provides enhanced robustness against classification errors, and thus it should preferably be followed in cases where the classification error rate is not very low.

Thus, the training algorithm can be summarized as follows:
Step 1. Define and extract an auxiliary feature set $\mathbf{o}$ from the source training data set.
Step 2. Align the source related data and the target data to form extended combined feature vectors $\mathbf{z} = \left[\mathbf{x}^\top \mathbf{o}^\top \mathbf{y}^\top\right]^\top$.
Step 3. Split the target data $\mathbf{y}$ into B clusters using, e.g., the K-means algorithm.
Step 4. Group the extended vectors $\mathbf{z}$ into the same B clusters based on the clustered target data $\mathbf{y}$.
Step 5. Train a mode classifier that aims at finding the correct target based cluster using only the source related features $\mathbf{x}$ and $\mathbf{o}$.
Step 6. Train B separate models for the different modes. Use as the training data the data classified to the corresponding cluster.

Conversion

During the usage of the multi-mode processing system, the conversion system must first obtain the source vector to be converted and the corresponding auxiliary vector. This data is used as an input to the classifier that selects the mode. Finally, the conversion of the vector is handled using the conversion scheme corresponding to the selected mode.

The following concrete steps summarize the conversion algorithm:
Step 1. Extract the auxiliary feature vector $\mathbf{o}$ from the source data.
Step 2. Select the correct mode using the source related vectors $\mathbf{o}$ and $\mathbf{x}$ as input;
Step 3. Use the selected model to convert the source feature vector $\mathbf{x}$;

Remarks on the properties of the proposed approach

The proposed approach summarized above has many beneficial properties. First, the approach is fully data-driven and there is no need to rely on any heuristic solutions. Second, the method is very flexible in the sense that it is for example very easy to change the number of modes/clusters. Third, there is no requirement to utilize any linguistic information but if such information is available it can very easily be used to support the mode selection. Finally, the proposed method offers very good performance. The performance advantage is demonstrated in the next subsection using practical experiments but the good performance can also be explained from another point of view. Figure 5.1 depicts the distribution of the first two LSFs in a small set of target LSF vectors, selected randomly from a larger voice conversion training set. The line illustrates the boundary between the two clusters obtained through K-means clustering of the target data, whereas the circles and crosses demonstrate the clustering decisions based on the widely used voiced/unvoiced classification. Provided that the mode selection is made correctly,

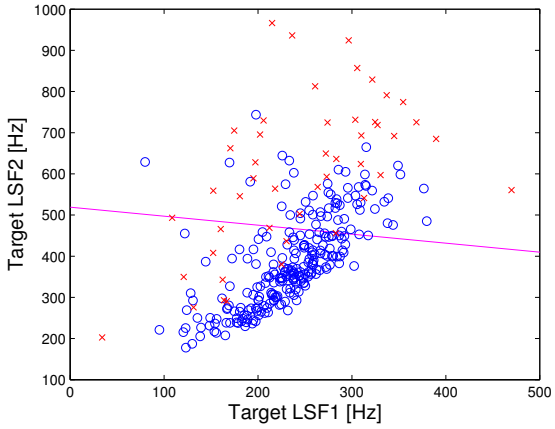

Figure 5.1: K-means based clustering of target data vs. voiced/unvoiced clustering. The line illustrates the division between the two K-means based clusters while o and x denote voiced and unvoiced data, respectively. It is easy to see that there is significantly less variability within each cluster when the clustering is performed using target data instead of voicing decisions. (From [Nur06e].)

it is evident that in the case of target data based clustering the distribution of any conversion errors will be much narrower than in the case of voiced/unvoiced clustering. While it is in general not possible to achieve a 100% mode classification rate, the proposed approach still successfully mimics this optimal case, leading to clear measurable improvements.

5.3.2 Experimental results

The proposed data clustering and mode selection approach was tested in the voice conversion system introduced in Section 5.1. To highlight the performance advantage achievable using the novel clustering method, it was compared against the common approach of voiced/unvoiced clustering that also offers good performance and that was used in Section 5.1. The comparison was carried out through the measurement of the average mean squared error between the converted and the target vectors.

Test set-up

The two different conversion schemes, the first based on the proposed approach and the second based on the traditional voiced/unvoiced clustering used in Section 5.1, were implemented for the conversion of LSF vectors. In the implementation of the proposed approach, several source-speech related features were used to form the auxiliary data vector $\mathbf{o}$. More specifically, the auxiliary data included the

Table 5.6: Comparison between the conversion MSE achieved using the conventional voiced/unvoiced clustering and the proposed data-driven clustering schemes.

	Training set	Testing set
Voiced/unvoiced clustering	23 058	23 559
Proposed	21 015	21 833
Proposed (perfect classifier)	15 295	15 770

first and second derivatives of the LSF vectors, the pitch parameter, the energy parameter, the residual amplitude spectrum and the voicing information for the spectrum. The number of modes was set to two to enable easy comparison with the voiced/unvoiced approach, and the mode selector was implemented using a simple linear discriminative function. In the case of the traditional voiced/unvoiced clustering, a single voicing decision for each frame was made based on the voicing threshold estimate given by the VLBR codec, obtained using the approach presented in Section 3.1.2.

Both conversion schemes were trained and tested using the same training and testing data sets. A data set containing 90 sentences (29 880 frames) from a source speaker and a target speaker was used for the training while a distinct set of 99 sentences (32 700 frames) was reserved for the testing phase. In both sets, the source and target vectors were aligned using dynamic time warping, supported with phoneme-level segmentation. All the conversions were handled using the Gaussian mixture modeling based approach. A separate GMM with 16 Gaussian components was trained for each mode.

Since the mode classifier was implemented in a very simple way, a third conversion scheme was also implemented that directly utilized the perfect classification based on the target data. In general, of course, the implementation of such a perfect classifier is not possible. Nevertheless, this third conversion scheme can be used for demonstrating the theoretical performance bound that cannot be exceeded with the proposed approach provided that the initial clustering and the conversion schemes are kept unchanged.

Results

The results achieved in the test are summarized in Table 5.6. For the scheme based on the conventional voiced/unvoiced clustering, the total mean squared error between the converted LSFs and the corresponding target LSFs was 23 058 for the training set and 23 559 for the testing set. For the proposed scheme, when implemented as described above, the total MSE scores of 21 015 and 21 833 for the training set and the testing set, respectively. In the ideal hypothetical case with 100% classification rate, providing the performance bound for the given initial

clusters, the total MSE figures were found to be 15 295 and 15 770 for the training and the testing set, respectively.

As is clearly evident from the results, the proposed method outperforms the conventional voicing based approach with a clear margin, despite the fact that the simple mode classifier only achieved a classification error rate of 12.4%. Moreover, the performance advantage was achieved even though the traditional voiced/unvoiced classification used as a reference also offered a very natural, theoretically appealing and efficient clustering scheme. For example, it was found earlier, while carrying out preliminary experiments during the preparation of the first version of the VLBR based conversion system presented in Section 5.1, that this voiced/unvoiced scheme already clearly outperforms an implementation with only one GMM model but twice the number of mixtures. If the proposed approach was compared against some arbitrary clustering scheme, the improvement might have been even larger.

5.4 Voicing level control

In voice conversion, and in many other speech processing applications, the speech signal is modified in a manner that causes changes in the spectrum. Typical examples of such cases outside the field of voice conversion include the coarse quantization of the linear prediction coefficients in very low bit rate speech coding and the spectral smoothing at the concatenation boundaries in concatenative text-to-speech synthesis. The modifications performed on speech signals or their spectra may also cause unwanted changes in the effective degree of voicing if there is no explicit control on voicing. The changes in voicing, in turn, may degrade the perceptual quality of the processed speech.

In this section, the problem of unwanted changes in the degree of voicing is studied and an explicit control of voicing is proposed to tackle this problem. The proposed approach for voicing control, previously published in [Nur07c], is implemented and experimented with in a VLBR-based voice conversion system that is based on the implementation presented in Section 5.1 but that includes the improvements presented in Section 5.2 and in Section 5.3. The voicing control is found to offer more natural and stable voicing levels and a clear improvement in speech quality.

5.4.1 Unwanted changes in voicing

As discussed earlier in this thesis, e.g., in Chapter 2, many speech processing techniques utilize linear prediction. For this reason, and to make the discussions easy to follow, it is assumed in this section that the spectral envelope of the vocal tract contribution is modeled using linear prediction. Moreover, it is assumed that the excitation is modeled using the parametric representation introduced in

Section 3.1. Nevertheless, it should be noted that this approach was chosen only for convenience and that the voicing related phenomenon discussed in this section is not only present in this particular speech model or in voice conversion.

To illustrate the unwanted changes in voicing, let us assume that the original LP coefficients are modified from $\{a_j\}$ to $\{a'_j\}$ as a result of some speech processing technique. The modification could happen, e.g., due to very coarse quantization in very low bit rate speech coding, due to spectral smoothing in concatenative TTS or due to a voice conversion related transformation. As a result of this modification, the spectral envelope changes accordingly. Assuming that the filter remains stable (that can be guaranteed, e.g., by performing the modification in the line spectral frequency domain), the old and the new spectral envelopes can be directly computed based on the gth order LP synthesis filter using

$$H\left(e^{i\omega}\right) = \frac{1}{1 - \sum_{j=1}^{g} a_j e^{-ij\omega}}, \tag{5.8}$$

and by using the same equation with the modified coefficients $\{a'_j\}$ to obtain $H'(e^{i\omega})$. In the experiments, g was set to 10.

The effect that the spectral modification has on voicing can be studied by measuring the energies of the voiced and unvoiced contributions in the spectrum before and after the modification. The average energy of the voiced part for a single frame, denoted as E_V, can be estimated by sampling the spectrum at the frequencies of the sinusoids, ω_m, as

$$E_V = \sum_{m=1}^{L} \left(\left|H\left(e^{i\omega_m}\right)\right| v_m A_m\right)^2. \tag{5.9}$$

Similarly, the energy of the unvoiced contribution, E_U , can be computed as

$$E_U = \sum_{m=1}^{L} \left(\left|H\left(e^{i\omega_m}\right)\right| (1 - v_m) A_m\right)^2. \tag{5.10}$$

The notations in both of the above equations are similar as in Section 3.1, i.e., v_m denotes the degree of voicing for the mth sinusoidal component ranging from 0 to 1, A_m is the amplitude of the mth sinusoid, and L denotes the number of harmonics. The corresponding energies after the spectral modifications, E'_V and E'_U, can be obtained using similar calculations as in Equation (5.9) and Equation (5.10) but by substituting $H(e^{i\omega})$ with $H'(e^{i\omega})$ to take into account the changes in the LP coefficients. It should also be noted that if the spectral modifications would cause changes in other parameters than the LP coefficients, these changes should also be taken into account when computing E'_V and E'_U.

It is usual in speech processing systems to carefully control the behavior of the overall energy but it is not common to explicitly control the relative contributions of the voiced and unvoiced components to the overall energy. However, if

there is no explicit control on voicing, the spectral modifications often change the perceived level of voicing in a clearly audible way. This is caused by the fact that the relative contribution of the voiced (or the unvoiced) component to the overall energy is often changed due to the spectral modification, i.e.,

$$\frac{E'_V}{E'_V + E'_U} \neq \frac{E_V}{E_V + E_U}. \tag{5.11}$$

The perceptual effect of the unwanted changes in voicing can in practice be observed as audible changes in the amplitude of the spectrally shaped noise generated to model the noise-like unvoiced contribution. This effect is discussed more closely and demonstrated using a practical example and experimental results in Section 5.4.3.

5.4.2 Voicing control

The unwanted changes in voicing can be corrected by controlling in an explicit way the overall level of voicing, calculated as the ratio between the energy of the voiced contribution and the total energy, $E_V/(E_V + E_U)$. The detailed implementation of the voicing control depends on the speech model used in the target application. In addition, even with a fixed speech model, there are different alternatives regarding the implementation.

In the case of the speech model discussed in this thesis, one possible solution could be to establish a frequency-dependent function for modifying the degree of voicing for the different sinusoids. For example, in the simplest case,

$$v'_m = f\left(v_m, \omega_m, E_V, E'_V\right). \tag{5.12}$$

The exact function can be designed in many ways and the parameters used in defining the modified degree of voicing, v'_m , could also be different than the ones given in Equation (5.12). Nevertheless, the aim is to modify the voicing of the sinusoids in such a manner that if computations similar to Equation (5.9) and Equation (5.10) would be applied again for calculating the energies after the voicing control, $\hat{E}'_V$ and $\hat{E}'_U$, we would now have

$$\frac{E'_V}{E'_V + E'_U} = \frac{E_V}{E_V + E_U}. \tag{5.13}$$

Alternatively, it may be desired to only go towards this goal without fully satisfying it, or the target level of voicing might be decided using other techniques. For example, in voice conversion, there could be a separate conversion model for finding out the target levels for the relative contributions of the voiced and unvoiced components based on the non-converted voicing values and possibly some other parameters, with the aim of modeling the speaker-dependencies in voicing.

The frequency-dependent operation sketched above can be used, for example, for focusing the increase in voicing more to low frequencies and/or the decrease in voicing more to high frequencies, which may be perceptually justified. However, a much simpler but still effective solution can be obtained by treating all sinusoids in the same manner. It is easy to see that the objective can be approximately achieved e.g. using the following simplified function,

$$v'_m = \min\left(v_m \sqrt{\frac{E_V}{E'_V}}, 1\right). \tag{5.14}$$

In cases where $E'_V = 0$, the voicing can be left unmodified.

Assuming that there is also a mechanism for ensuring that the overall energy stays unchanged, and that the voicing values v_m are continuous values in the range from 0 to 1, the simplified solution presented in Equation (5.14) effectively controls the level of voicing. If the voicing decisions are hard as, e.g., in the *multi-band excitation* (MBE) model [Gri88], i.e., v_m is always either 0 or 1, the best solution would be to change the voicing values of some sinusoids to approximately satisfy the condition in Equation (5.13). Similar approach but with continuous voicing values could be used to complement the simplified solution in Equation (5.14) to fully satisfy Equation (5.13). Another solution could be obtained by also modifying the amplitudes of the sinusoids in addition or instead of modifying the degree of voicing of the sinusoids.

5.4.3 Experiments on voicing control

In voice conversion, the spectral changes are coming from multiple sources because many, if not all, parameters are converted. However, the voicing control can still be implemented as proposed earlier in this section, provided that the changes in all parameters are taken into account when applying the equations introduced in this section. The voicing control can operate directly on the voicing values without changing any other parameter values.

A practical example case demonstrating the need for controlling the voicing is given in Figure 5.2. The figure depicts the overall level of voicing, $E_V/(E_V + E_U)$, for each 10-ms frame of an example sentence before and after voice conversion, obtained using the VLBR codec's speech model presented in Section 3.1 and the voice conversion techniques presented earlier in this chapter. As can be seen from the figure, the effective voicing level has clearly changed in the conversion even though the parameter values related to the degree of voicing have not been converted at all. The figure also shows that the effective level of voicing is often decreased in the conversion, leading to a higher contribution of the noise-like excitation that can be perceptually observed as increased noise. Moreover, since the difference in the level of voicing before and after the conversion is not constant,

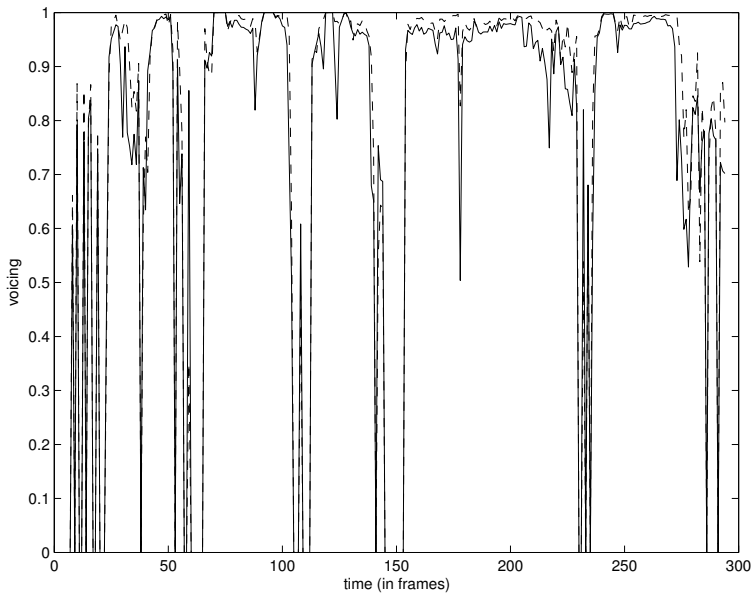

Figure 5.2: Level of voicing before (dashed line) and after conversion (solid line). (From [Nur07c].)

Table 5.7: Direction of the change in the overall level of voicing after voice conversion in the test material (percentage of frames). The voicing values are not changed in the conversion but the effective degree of voicing changes due to the spectral modifications.

	Increased voicing	Decreased voicing	No change
Voicing after conversion	26.4%	47.9%	25.7%

the increase in noise is also non-constant, leading to a pumping-like perceptual effect that is clearly audible with careful listening.

The unwanted changes in voicing were also studied objectively using a larger test set consisting of 42 sentences. The overall level of voicing was measured before and after voice conversion for all the 10-ms frames of these sentences. As summarized in Table 5.7, in 47.9% of the frames, the voice conversion decreased the level of voicing, while an increase in the level of voicing occurred in 26.4% of the frames. In the rest of the frames, the level of voicing did not change due to the fact that the whole spectrum was considered either fully voiced or fully unvoiced both before and after conversion. The average level of voicing in the whole training set including all the frames was decreased due to the conversion by about 2.8%. These experimental results on a larger test set support the findings

observed in the sentence illustrated in Figure 5.2: the voice conversion system on average decreases the level of voicing, which as a direct consequence of the properties of the parametric speech model leads to an increased level of noise in the output.

The proposed scheme for voicing control can efficiently correct the level of voicing in such a manner that the voicing change caused by the voice conversion system is always fully compensated. The perceptual effect of having the voicing control available was evaluated using four expert listeners. The listeners heard converted speech samples with and without voicing control and were asked to evaluate the quality differences between the two samples. Repeated listening was possible without any restrictions. The language used in the samples was English and the voice conversion was performed between genders.

The speech quality in the samples produced using voicing control was always observed better or equal to the quality of the conventional samples. The voicing control was found to remove part of the noise generated during the conversion. This is quite natural since the voicing control slightly reduces the contribution of the noise-like component, and it makes the behavior of that component more stable and ensures that it is better in line with the expected degree of voicing.

5.5 Conclusions

This chapter has introduced the VLBR-based voice conversion system and further enhancements to the baseline system. As the first enhancement to the baseline Gaussian mixture model based system, this chapter has introduced a novel technique for converting the prosody based on a prosodic codebook. The proposed approach not only improves the conversion of the prosodic features but was also found to help the overall system in achieving less robotic sound quality. Moreover, a novel approach for data clustering and mode selection has also been introduced and used for enhancing the performance of the VLBR-based voice conversion system. Finally, the effect that spectral modifications have on the degree of voicing has been discussed. The unwanted changes in voicing that can be corrected using the proposed simple but effective scheme based on explicit control of the voicing level.

One of the main benefits of the proposed VLBR-based voice conversion system is the obvious fact that the approach also offers the possibility for very efficient speech compression. At the same time, this main strength can also be regarded as the main weakness because the complete system contains more sources of errors than typical voice conversion systems, i.e., the complete system suffers also from quantization errors in addition to the modeling and conversion errors. This naturally limits the achievable speech quality and dictates that if voice conversion is used, e.g., in VLBR-based synthesis, the bitrate should be higher than usual to avoid excessive degradation of quality.

Chapter 6

Conclusions and future work

This thesis has introduced a new segmental parametric speech codec referred to as the VLBR codec, presented techniques for enhancing its compression efficiency, and extended the codec into a framework suitable for memory-efficient concatenative speech synthesis and voice conversion. The VLBR codec based on sinusoidal modeling of speech is capable of achieving relatively high speech quality at bitrates of about 1 kbps or below. The coding efficiency is firstly enhanced through the use of model simplifications, mode-based segmental processing, and the method of adaptive downsampling and quantization. The performance is further enhanced using a new multi-mode matrix quantizer structure that utilizes a low-complexity vector-based prediction scheme. Lossless compression of the quantizer indices is made more efficient by enhancements proposed to the conventional dynamic codebook reordering approach. Finally, the input signal is made easier to compress using a novel preprocessing technique based on perceptual irrelevancy removal.

The work on the VLBR codec was extended into the field of speech synthesis, focusing mainly on the compression of text-to-speech databases. A VLBR-based concatenative synthesizer, applicable to different types of TTS front ends and database designs, is presented. Moreover, the simple concept of dynamic quantizer structures is introduced. This approach further improves the compression efficiency of the VLBR codec in the context of TTS database compression by allowing convenient retraining and run-time updating of the codec and/or its quantizers. In addition, a compression-motivated method for highly efficient computation of concatenation costs in unit selection synthesis is introduced.

The VLBR codec and the VLBR based concatenative synthesizer are also complemented with voice conversion functionality. The main motivation in this work is to enable cost-efficient and flexible generation of new TTS voices in VLBR-based speech synthesis. First, a voice conversion system based on a combination of the VLBR codec and the basic GMM-based conversion approach is presented. Second, a new method for prosody conversion is introduced and inte-

grated into the VLBR-based voice conversion system. Third, the performance of the system is further enhanced using a novel method for data clustering and mode selection, as well as through explicit control of the perceived level of voicing.

When combined together, the methods presented in this thesis offer a powerful and complete framework for speech compression, speech synthesis and voice conversion. The combined system can be used in different ways and configurations, depending on the application scenario. For example, the VLBR codec itself can be utilized for efficient compression of audio books, and when supported with the speech synthesis related methods, the combined system can be used for reducing the footprint and the computational load of concatenative text-to-speech synthesizers to levels required in some embedded applications. The VLBR-based voice conversion techniques can be used to complement the codec both in storage applications and in concatenative speech synthesis. It is also possible to only utilize the voice conversion functionality, e.g., in games or other entertainment applications. Compared to the use of separate systems for compression, synthesis, and voice conversion, the integrated solution, for example, avoids unnecessary repeated analysis steps and the related cumulation of estimation inaccuracies. In addition, it can be ensured that the overall result is not compromised by isolated optimization of the different parts.

As for future studies, there are still a number of open issues that require further investigation. First, even though many parts of the work presented in this thesis have already been in commercial usage, finding the optimal complete set-up is still a very challenging task. Many of the techniques presented in this thesis can be included into or omitted from the overall system and there are numerous set-up parameters and design choices that can be modified, etc. Furthermore, yet another degree of difficulty is caused by the fact that the optimal set-up depends on the speech data to be processed. Better and more complete methods for automatic fine-tuning of the complete system, e.g., for different TTS voices could be developed. Psychoacoustic models could be used in these fine-tuning methods, as well as otherwise for further improving the different parts of the overall system.

Concerning the VLBR codec, there is still room for further improvement despite the high compression efficiency, especially when the complete framework is considered. The combination of the actual compression techniques used in the codec is already quite efficient, and further improvements would most likely require increases in the memory usage and/or computational load, but the parametric model could clearly be improved. The use of the simplified model was perfectly justified in the initial phases of the work when the target was to achieve reasonably good speech quality at the lowest possible bitrates but it also limits the achievable quality, especially from the viewpoint of speech synthesis and voice conversion. It could be possible to use a more accurate model without compromising the compression efficiency too much. Less simplified sinusoidal models or waveform interpolation based models could be studied. In addition to improv-

ing the parametric speech model, the implementation of the segmentation process could be improved as well. Also, the bandwidth of the signal could be extended from 8 kHz to 16 kHz and beyond.

The preliminary results related to the use of the VLBR framework in HMM-based statistical synthesis and in hybrid synthesis, left outside the scope of this thesis but reported in [Sil09], [Sil10], and [Nur13b], can be considered very promising and to also open up an especially interesting area of future research. The use of the same framework and parameterization for both the coded unit selection based parts and for the HMM-synthesized parts can be expected to yield good results but the final selection of the parameterization is still an open question. It is clear that the parametric representation should be modified so that it would best fit the needs of both speech synthesis and compression, and most likely making the vocal tract model more accurate would help but further research is needed to figure out a detailed solution. Another possible direction would be to go towards models that are better in line with the human speech production. The use of glottal inverse filtering could be considered one possible direction. Promising results have already been obtained using this approach in statistical synthesis [Rai10] but the method's suitability for efficient compression is not evident and modifications to the basic method might be needed. Yet another attractive area of future research relates to the development of a fully functional hybrid synthesizer: it would be important to study more closely whether hybrid synthesis really is the way to go, as it currently seems, or could further improvements in the area of statistical speech synthesis, and in the related parameterizations, make it the best choice for the TTS systems of tomorrow.

The VLBR-based voice conversion system presented in this thesis offers a relatively good approach for voice conversion, provided that the bitrate is high enough for reaching a meaningful speech quality level even after voice conversion. However, the approach also has its weaknesses. First, the conventional GMM-based voice conversion technique used in the system suffers from the well-known problems of overfitting, oversmoothing, and the limitation to time-independent framewise conversion, as described, e.g., in [Nur12]. Second, the size of the training set needs to be relatively large for the best results. Third, the training data needs to be parallel but still the accuracy of the alignment may be a bottleneck limiting the quality. These, as well as several other voice conversion related issues, have been studied in closely related research activities that are deemed to be outside the scope of this thesis despite containing several contributions from the author, due to the reasons mentioned in Chapter 1. Nevertheless, the work reported, e.g., in [Hel08a], [Hel10], [Nur08b], [Nur10b], [Pop09], [Pop11], [Pop12], [Sil13], [Tao10], [Tia08], and [Tia12] has resulted in ideas or findings that can be used for enhancing the performance of the proposed VLBR-based voice conversion system in certain usage scenarios or as potentially beneficial alternative methods for handling the conversion.

Despite the above out-of-the-scope work, there are still subtopics that require further research, not only in connection with VLBR-based voice conversion but also more widely in the field of voice conversion. Firstly, the topic of training data alignment would definitely deserve additional studying because any inaccuracies in the alignment directly affect the performance of the voice conversion system, regardless of the conversion techniques used in the system. Intuitively, more accurate alignment between the source and the target materials would allow building of better voice conversion models. In particular, the soft alignment method [Tia09] offers one possible area of future research, and better techniques for handling non-parallel and cross-lingual training materials should also be developed.

From the viewpoint of practical applications of voice conversion, another important area of research relates to the use of real-world recordings instead of voice data recorded in a quiet recording studio. For example, if the target is to allow the user of a mobile phone to create new TTS voices by providing a small amount training speech material from the intended target speaker (such as, e.g., the spouse of the user), the voice conversion system should be able to deal with any type of recorded speech signals, potentially recorded in different types of noisy environments. The collection of such data could be facilitated using the basic framework proposed in the patent [Nur10a].

The topic of parameterization has already been touched upon several times in this chapter but it should still be noted that improvements in the parameterization could potentially also offer enhancements in the voice conversion quality. Finally, deeper understanding on the underlying issues related to speaker identity perception could significantly advance the field. For example, it would be interesting to study how highly-skilled impersonators or imitators change their voice during their performance and how dependent their success is, for example, on the choice of words or on the use, and possibly even over-use, of certain mannerisms.

Bibliography

[Abe88] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, Voice conversion through vector quantization. In *Proc. of ICASSP*, pp. 565–568, 1988.

[Ars97] L. Arslan and D. Talkin, Voice conversion by codebook mapping of line spectral frequencies and excitation spectrum. In *Proc. of Eurospeech*, pp. 1347–1350, 1997.

[Ars99] L. Arslan, Speaker transformation algorithm using segmental codebooks (STASC). *Speech Communication*, 28(3):pp. 211–226, 1999.

[Cer98] J. Cernocky, G. Baudoin, and G. Chollet, Segmental vocoder-going beyond the phonetic approach. In *Proc. of ICASSP*, pp. 605–608, 1998.

[Cey02] T. Ceyssens, W. Verhelst, and P. Wambacq, On the construction of a pitch conversion system. In *Proc. of EUSIPCO*, 2002.

[Cha86] P.-C. Chang and R. Gray, Gradient algorithms for designing predictive vector quantizers. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):pp. 679–690, 1986.

[Cha92] W.-Y. Chan, S. Gupta, and A. Gersho, Enhanced multistage vector quantization by joint codebook design. *IEEE Transactions on Communications*, 40(11):pp. 1693–1697, 1992.

[Cha98a] D. Chappell and J. Hansen, Speaker-specific pitch contour modelling and modification. In *Proc. of ICASSP*, pp. 885–888, 1998.

[Cha98b] D. Chappell and J. Hansen, Spectral smoothing for concatenative speech synthesis. In *Proc. of ICSLP*, 1998.

[Cha02a] D. Chappell and J. Hansen, A comparison of spectral smoothing methods for segment concatenation based speech synthesis. *Speech Communication*, 36, 2002.

[Cha02b] D. Chazan, R. Hoory, Z. Kons, D. Silberstein, and A. Sorin, Reducing the footprint of the IBM trainable speech synthesis system. In *Proc. of Interspeech*, pp. 2381–2384, 2002.

[Cla04] R. Clark, K. Richmond, and S. King, Festival 2 - build your own general purpose unit selection speech synthesiser. In *Proc. of ISCA Workshop on Speech Synthesis*, 2004.

[Cla07] R. Clark, K. Richmond, and S. King, Multisyn: Open-domain unit selection for the Festival speech synthesis system. *Speech Communication*, 49(4):pp. 317–330, 2007.

[Con97] A. Conkie and S. Isard, Optimal coupling of diphones. In J. van Santen et al., (Ed.) *Progress in Speech Synthesis*, pp. 293–304, Springer-Verlag, 1997.

[Cou82] J.-L. Courbon and F. Emerard, Sparte: A text-to-speech machine using synthesis by diphones. In *Proc. of ICASSP*, pp. 1597–1600, 1982.

[Cre97] M. Crespo, P. Velesco, L. Serrano, and J. Sardina, On the use of a sinusoidal model for speech synthesis in text-to-speech. In J. van Santen et al., (Ed.) *Progress in Speech Synthesis*, pp. 57–70, Springer-Verlag, 1997.

[Cup85] V. Cuperman and A. Gersho, Vector predictive coding of speech at 16 kbits/s. *IEEE Transactions on Communications*, 33(7):pp. 685–696, 1985.

[Cup95] V. Cuperman, P. Lupini, and B. Bhattacharya, Spectral excitation coding of speech at 2.4 kb/s. In *Proc. of ICASSP*, pp. 496–499, 1995.

[Das96] A. Das, A. Rao, and A. Gersho, Variable-dimension vector quantization. *IEEE Signal Processing Letters*, 3(7):pp. 200–202, 1996.

[Dav80] S. Davis and P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):pp. 357–366, 1980.

[Dem77] A. Dempster, N. Laird, and D. Rubin, Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 39(1), 1977.

[DeN96] F. DeNatale, S. Fioravanti, and D. Giusto, DCRVQ: A new strategy for efficient entropy coding of vector-quantized images. *IEEE Transactions on Communications*, 44(6):pp. 696–706, 1996.

[Des10] S. Desai, A. Black, B. Yegnanarayana, and K. Prahallad, Spectral mapping using artificial neural networks for voice conversion. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):pp. 954–964, 2010.

[Din08] F. Ding, J. Nurminen, and J. Tian, Efficient join cost computation for unit selection based TTS systems. In *Proc. of Interspeech*, pp. 589–592, 2008.

[Dud01] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. Wiley and Sons, New York, 2001.

[Dut97] T. Dutoit, *An introduction to text-to-speech synthesis*. Kluwer Academic Publishers, The Netherlands, 1997.

[Dux04] H. Duxans, A. Bonafonte, A. Kain, and J. van Santen, Including dynamic and phonetic information in voice conversion systems. In *Proc. of Interspeech*, 2004.

[Eku99] E. Ekudden, R. Hagen, I. Johansson, and J. Svedberg, The adaptive multi-rate speech coder. In *Proc. of IEEE Workshop on Speech Coding*, pp. 117–119, 1999.

[Eri99] T. Eriksson, J. Lindén, and J. Skoglund, Interframe LSF quantization for noisy channels. *IEEE Transactions on Speech and Audio Processing*, 7(5):pp. 495–509, 1999.

[Err07] D. Erro and A. Moreno, Weighted frequency warping for voice conversion. In *Proc. of Interspeech*, pp. 1965–1968, 2007.

[Err10] D. Erro, A. Moreno, and A. Bonafonte, INCA algorithm for training voice conversion systems from nonparallel corpora. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):pp. 944–953, 2010.

[Esl11] M. Eslami, H. Sheikhzadeh, and A. Sayadiyan, Quality improvement of voice conversion systems based on trellis structured vector quantization. In *Proc. of Interspeech*, pp. 665–668, 2011.

[ETS94] ETSI, European digital cellular telecommunications system (phase 2); Full rate speech transcoding, GSM 06.10 (ETS 300 580-2). European Telecommunications Standards Institute (ETSI), 1994.

[ETS00] ETSI, Digital cellular telecommunications system (phase 2+); Adaptive multi-rate (AMR) speech transcoding (GSM 06.90 version 7.2.1 release 1998). European Telecommunications Standards Institute (ETSI), 2000.

[Fan60] G. Fant, *Acoustic Theory of Speech Production*. Mouton & Co., The Netherlands, 1960.

[Fék06] M. Fék, P. Pesti, G. Németh, C. Zainkó, and G. Olaszy, Corpus-based unit selection TTS for Hungarian. In *Proc. of Text, Speech and Dialogue*, pp. 367–373, 2006.

[Fel97] J. M. Fellowes, R. E. Remez, and P. E. Rubin, Perceiving the sex and identity of a talker without natural vocal timbre. *Perception & Psychophysics*, 59(6):pp. 839–849, 1997.

[Fer02] A. Ferreira, Perceptual coding using sinusoidal modeling in the MDCT domain. In *Proc. of Audio Engineering Society Convention*, 2002.

[Fos85] J. Foster, R. Gray, and M. Ostendorf Dunham, Finite-state vector quantization for waveform coding. *IEEE Transactions on Information Theory*, 31(3):pp. 348–359, 1985.

[Ger92] A. Gersho and R. Gray, *Vector quantization and signal compression*. Kluwer Academic Publishers, Boston, 1992.

[Gil03] B. Gillet and S. King, Transforming F0 contours. In *Proc. of Interspeech*, pp. 101–104, 2003.

[Gra84] R. Gray, Vector quantization. *IEEE ASSP Magazine*, 1(2):pp. 4–29, 1984.

[Gra98] R. Gray and D. Neuhoff, Quantization. *IEEE Transactions on Information Theory*, 44(6):pp. 2325–2383, 1998.

[Gri88] D. Griffin and J. Lim, Multi band excitation vocoder. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(8):pp. 664–678, 1988.

[Ham50] R. Hamming, Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2), 1950.

[Hei02] A. Heikkinen, *Development of a 4 kbps hybrid sinusoidal/CELP speech coder*. Ph.D. thesis, Tampere University of Technology, June 2002.

[Hel07a] E. Helander and J. Nurminen, A novel method for prosody prediction in voice conversion. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 509–512, 2007.

[Hel07b] E. Helander and J. Nurminen, On the importance of pure prosody in the perception of speaker identity. In *Proc. of Interspeech*, pp. 2665–2668, 2007.

[Hel08a] E. Helander, J. Nurminen, and M. Gabbouj, LSF mapping for voice conversion with very small training sets. In *Proc. of ICASSP*, pp. 4669–4672, 2008.

[Hel08b] E. Helander, J. Schwarz, J. Nurminen, H. Silén, and M. Gabbouj, On the impact of alignment on voice conversion performance. In *Proc. of Interspeech*, pp. 1453–1456, 2008.

[Hel10] E. Helander, T. Virtanen, J. Nurminen, and M. Gabbouj, Voice conversion using partial least squares regression. *IEEE Transactions on Speech and Audio Processing*, 18(5):pp. 912–921, 2010.

[Hel12] E. Helander, H. Silén, T. Virtanen, and M. Gabbouj, Voice conversion using dynamic kernel partial least squares regression. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3), 2012.

[Hua97] X. Huang, A. Acero, H. Hon, Y. Ju, J. Liu, S. Meredith, and M. Plumpe, Recent improvements on Microsoft's trainable text-to-speech system-whistler. In *Proc. of ICASSP*, pp. 959–962, 1997.

[Hua01] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, 2001.

[Huf52] D. Huffman, A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):pp. 1098–1101, 1952.

[Hun96] A. Hunt and A. Black, Unit selection in a concatenative speech synthesis system using a large speech database. In *Proc. of ICASSP*, pp. 373–376, 1996.

[Ina03] Z. Inanoglu, *Transforming a pitch in a voice conversion framework*. Master's thesis, University of Cambridge, July 2003.

[Ita75] F. Itakura, Line spectrum representation of linear predictor coefficients of speech signals. *Journal of the Acoustical Society of America*, 57(S1):p. S35, 1975.

[ITU96] ITU, Methods for subjective determination of transmission quality, ITU-T recommendation P.800. International Telecommunication Union (ITU), 1996.

[ITU00] ITU, ITU-T software tool library 2000 user's manual. International Telecommunication Union (ITU), 2000.

[Jah08] E. Jahangiri and S. Ghaemmaghami, Scalable speech coding at rates below 900 bps. In *Proc. of ICME*, pp. 85–88, 2008.

[Jay89] N. Jayant and J.-H. Chen, Speech coding with time-varying bit allocations to excitation and LPC parameters. In *Proc. of ICASSP*, pp. 65–69, 1989.

[Joh88] J. Johnston, Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas in Communications*, 6(2):pp. 314–323, 1988.

[Jua82] B.-H. Juang and J. Gray, A., Multiple stage vector quantization for speech coding. In *Proc. of ICASSP*, pp. 597–600, 1982.

[Kab86] P. Kabal and R. Ramachandran, The computation of line spectral frequencies using Chebyshev polynomials. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(6):pp. 1419–1426, 1986.

[Kai98] A. Kain and M. Macon, Spectral voice conversion for text-to-speech synthesis. In *Proc. of ICASSP*, pp. 285–288, 1998.

[Kai01] A. Kain and M. Macon, Design and evaluation of a voice conversion algorithm based on spectral envelope mapping and residual prediction. In *Proc. of ICASSP*, pp. 813–816, 2001.

[Kai07] A. Kain and J. van Santen, Unit-selection text-to-speech synthesis using an asynchronous interpolation model. In *Proc. of ISCA Workshop on Speech Synthesis*, pp. 172–177, 2007.

[Kan05] Y. Kang, Z. Shuang, J. Tao, W. Zhang, and B. Xu, A hybrid GMM and codebook mapping method for spectral conversion. In *Proc. of ACII*, pp. 303–310, 2005.

[Kar09] S. Karabetsos, P. Tsiakoulis, A. Chalamandaris, and S. Raptis, Embedded unit selection text-to-speech synthesis for mobile devices. *IEEE Transactions on Consumer Electronics*, 55(2):pp. 613–621, 2009.

[Kaw99] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigné, Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds. *Speech Communication*, 27(3–4), 1999.

[Kha00] H. Khalil and K. Rose, Asymptotic closed-loop design of predictive multi-stage vector quantizers. In *Proc. of ICIP*, pp. 199–202, 2000.

[Kha01a] H. Khalil and K. Rose, Predictive multistage vector quantizer design using asymptotic closed-loop optimization. *IEEE Transactions on Image Processing*, 10(11):pp. 1765–1770, 2001.

[Kha01b] H. Khalil and K. Rose, Robust predictive vector quantizer design. In *Proc. of Data Compression Conference*, pp. 33–42, 2001.

[Kha01c] H. Khalil, K. Rose, and S. Regunathan, The asymptotic closed-loop approach to predictive vector quantizer design with application in video coding. *IEEE Transactions on Image Processing*, 10(1):pp. 15–23, 2001.

[Kim97] E.-K. Kim, S. Lee, and Y.-H. Oh, Hidden Markov model based voice conversion using dynamic characteristics of speaker. In *Proc. of Eurospeech*, pp. 2519–2522, 1997.

[Kla80] D. Klatt, Software for a cascade/parallel formant synthesizer. *Journal of the Acoustical Society of America*, 67:pp. 971–995, 1980.

[Kla87] D. Klatt, Review of text-to-speech conversion for English. *Journal of the Acoustical Society of America*, 82(3):pp. 737–793, 1987.

[Kle95a] W. Kleijn and J. Haagen, Waveform interpolation for coding and synthesis. In W. Kleijn and K. Paliwal, (Eds.) *Speech Coding and Synthesis*, pp. 175–207, Elsevier Science B.V., 1995.

[Kle95b] W. Kleijn and K. Paliwal, An introduction to speech coding. In W. Kleijn and K. Paliwal, (Eds.) *Speech Coding and Synthesis*, pp. 1–47, Elsevier Science B.V., 1995.

[Kon04] A. Kondoz, *Digital Speech, 2nd edition*. Wiley and Sons, England, 2004.

[Kri05] V. Krishnan, T. Barnwell III, and D. Anderson, Using dynamic codebook re-ordering to exploit inter-frame correlation in MELP coders. In *Proc. of Interspeech*, pp. 2717–2720, 2005.

[Kro88] P. Kroon and B. Atal, Strategies for improving CELP coders. In *Proc. of ICASSP*, pp. 151–154, 1988.

[Kum03] A. Kumar and A. Verma, Using phone and diphone based acoustic models for voice conversion: a step towards creating voice fonts. In *Proc. of ICASSP*, 2003.

[Läh03a] M. Lähdekorpi, *Perceptual irrelevancy removal in narrowband speech coding*. Master's thesis, Tampere University of Technology, Tampere, Finland, 2003.

[Läh03b] M. Lähdekorpi, J. Nurminen, A. Heikkinen, and J. Saarinen, Perceptual irrelevancy removal in narrowband speech coding. In *Proc. of Interspeech*, pp. 1081–1084, 2003.

[LeB93] W. LeBlanc, B. Bhattacharya, S. Mahmoud, and V. Cuperman, Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding. *IEEE Transactions on Speech and Audio Processing*, 1(4):pp. 373–385, 1993.

[Lee01] K.-S. Lee and R. Cox, A very low bit rate speech coder based on a recognition/synthesis paradigm. *IEEE Transactions on Speech and Audio Processing*, 9(5):pp. 482–491, 2001.

[Lee02] K.-S. Lee and R. Cox, A segmental speech coder based on a concatenative TTS. *Speech Communication*, 38(1–2):pp. 89–100, 2002.

[Li98] C. Li, E. Shlomot, and V. Cuperman, Quantization of variable dimension spectral vectors. In *Proc. of Asilomar Conference on Signals, Systems & Computers*, pp. 352–356, 1998.

[Lin80] Y. Linde, A. Buzo, and R. Gray, An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):pp. 84–95, 1980.

[Lin07] Z.-H. Ling, L. Qin, H. Lu, Y. Gao, L.-R. Dai, R.-H. Wang, Y. Jiang, Z.-W. Zhao, J.-H. Yang, J. Chen, and G.-P. Hu, The USTC and iFlytek speech synthesis systems for Blizzard Challenge 2007. In *Proc. of Blizzard Challenge Workshop*, 2007.

[Lol08] D. Lolive, N. Barbot, and O. Boeffard, Pitch and duration transformation with nonparallel data. In *Proc. of Speech Prosody*, pp. 111–114, 2008.

[Luk01a] J. Lukasiak and I. Burnett, Source enhanced linear prediction of speech incorporating simultaneously masked spectral weighting. *Journal of Telecommunications and Information Technology*, 2:pp. 15–23, 2001.

[Luk01b] J. Lukasiak, I. Burnett, and C. Ritz, Low rate speech coding incorporating simultaneously masked spectrally weighted linear prediction. In *Proc. of Interspeech*, pp. 1989–1992, 2001.

[Lup94] P. Lupini and V. Cuperman, Vector quantization of harmonic magnitudes for low-rate speech coders. In *Proc. of IEEE Global Telecommunications Conference*, pp. 858–862, 1994.

[Lup95] P. Lupini and V. Cuperman, Non-square transform vector quantization for low-rate speech coding. In *Proc. of IEEE Workshop on Speech Coding for Telecommunications*, pp. 87–88, 1995.

[Lup96] P. Lupini and V. Cuperman, Nonsquare transform vector quantization. *IEEE Signal Processing Letters*, 3(1):pp. 1–3, 1996.

[Mac87] I. MacKay, *Phonetics: the science of speech production, second edition*. Allyn and Bacon, USA, 1987.

[Mac96] M. Macon, Speech concatenation and synthesis using an overlap-add sinusoidal model. In *Proc. of ICASSP*, 1996.

[Mak72] J. Makhoul and J. Wolf, Linear prediction and the spectral analysis of speech. *BBN Report No. 2304*, 1972.

[Mak75] J. Makhoul, Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):pp. 561–580, 1975.

[Mak85] J. Makhoul, S. Roucos, and H. Gish, Vector quantization in speech coding. *Proceedings of the IEEE*, 73(11):pp. 1551–1588, 1985.

[McA86] R. McAulay and T. Quatieri, Speech analysis-synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):pp. 744–754, 1986.

[McA90] R. McAulay and T. Quatieri, Pitch estimation and voicing detection based on a sinusoidal speech model. In *Proc. of ICASSP*, 1990.

[McA95] R. McAulay and T. Quatieri, Sinusoidal coding. In W. Kleijn and K. Paliwal, (Eds.) *Speech Coding and Synthesis*, pp. 121–174, Elsevier Science B.V., 1995.

[McC08] A. McCree, K. Brady, and T. Quatieri, Multisensor very lowbit rate speech coding using segment quantization. In *Proc. of ICASSP*, pp. 3997–4000, 2008.

[Moo95a] B. Moore, *Hearing, 2nd edition*. Academic press, San Diego, 1995.

[Moo95b] B. Moore, *An introduction to the psychology of hearing, 4th edition*. Academic press, London, 1995.

[Mor12] E. Morley, E. Klabbers, J. van Santen, A. Kain, and S. Mohammadi, Synthetic F0 can effectively convey speaker ID in delexicalized speech. In *Proc. of Interspeech*, 2012.

[Mud98] D. Mudugamuwa and A. Bradley, Optimal transform for segmented parametric speech coding. In *Proc. of ICASSP*, pp. 53–56, 1998.

[Nar95] M. Narendranath, H. Murthy, S. Rajendran, and B. Yegnanarayana, Transformation of formants for voice conversion using artificial neural networks. *Speech Communication*, 16(2):pp. 207–216, 1995.

[Nis07] N. Nishizawa and H. Kawai, A preselection method based on cost degradation from the optimal sequence for concatenative speech synthesis. In *Proc. of Interspeech*, pp. 2869–2872, 2007.

[Nuk06] N. Nukaga, R. Kamoshida, K. Nagamatsu, and Y. Kitahara, Scalable implementation of unit selection based text-to-speech system for embedded solutions. In *Proc. of ICASSP*, pp. 849–852, 2006.

[Nur01a] J. Nurminen, *Pitch-cycle waveform quantization in a 4.0 kbps WI speech coder*. Master's thesis, Tampere University of Technology, Tampere, Finland, 2001.

[Nur01b] J. Nurminen, A. Heikkinen, and J. Saarinen, Objective evaluation of methods for quantization of variable-dimension spectral vectors in WI speech coding. In *Proc. of Interspeech*, pp. 1969–1972, 2001.

[Nur01c] J. Nurminen, A. Heikkinen, and J. Saarinen, Quantization of magnitude spectra in waveform interpolation speech coding. In *Proc. of Nordic Signal Processing Symposium*, pp. 65–69, 2001.

[Nur02] J. Nurminen, A. Heikkinen, and J. Saarinen, A novel quantization scheme for the noise-like component in waveform interpolation speech coding. In *Proc. of ICASSP*, pp. 649–652, 2002.

[Nur03a] J. Nurminen, Asymptotic closed-loop design of MA predictive quantizers. In *Proc. of 7th World Multiconference on Systemics, Cybernetics and Informatics*, pp. 402–407, 2003.

[Nur03b] J. Nurminen, Multi-mode quantization of adjacent speech parameters using a low-complexity prediction scheme. In *Proc. of Interspeech*, pp. 1069–1072, 2003.

[Nur06a] J. Nurminen, Enhanced dynamic codebook reordering for advanced quantizer structures. In *Proc. of Interspeech*, pp. 209–212, 2006.

[Nur06b] J. Nurminen, S. Himanen, and A. Rämö, Efficient technique for quantization of pitch contours. In *Proc. of Speech Prosody*, 2006.

[Nur06c] J. Nurminen, V. Popa, J. Tian, Y. Tang, and I. Kiss, A parametric approach for voice conversion. In *Proc. of Workshop on Speech-To-Speech Translation*, pp. 225–229, 2006.

[Nur06d] J. Nurminen, J. Tian, and I. Kiss, Framework for voice conversion, US Patent application 20060235685. Oct. 2006.

[Nur06e] J. Nurminen, J. Tian, and V. Popa, Novel method for data clustering and mode selection with application in voice conversion. In *Proc. of Interspeech*, pp. 2258–2261, 2006.

[Nur07a] J. Nurminen, Compression and decompression of data vectors, US Patent application 20070094019. Apr. 2007.

[Nur07b] J. Nurminen, S. Himanen, A. Rämö, and J. Vainio, Supporting a concatenative text-to-speech synthesis, US Patent application 20070011009. Jan. 2007.

[Nur07c] J. Nurminen, J. Tian, and V. Popa, Voicing level control with application in voice conversion. In *Proc. of Interspeech*, pp. 1973–1976, 2007.

[Nur08a] J. Nurminen, Method, apparatus and computer program product for controlling voicing in processed speech, US Patent application 20080109217. May 2008.

[Nur08b] J. Nurminen, J. Tian, and V. Popa, Memory-efficient method for high-quality codebook based voice conversion, US Patent application 20080147385. Jun. 2008.

[Nur10a] J. Nurminen, V. Popa, E. Helander, and J. Tian, Voice conversion training and data collection, US Patent 7813924. Oct. 2010.

[Nur10b] J. Nurminen, V. Popa, and J. Tian, Method, apparatus and computer program product for providing voice conversion using temporal dynamic features, US Patent 7848924. Dec. 2010.

[Nur11a] J. Nurminen and E. Helander, Prosody conversion, US Patent 7996222. Aug. 2011.

[Nur11b] J. Nurminen and S. Himanen, Dynamic quantizer structures for efficient compression, US Patent 8086057. Dec. 2011.

[Nur12] J. Nurminen, H. Silén, V. Popa, E. Helander, and M. Gabbouj, Voice conversion. In S. Ramakrishnan, (Ed.) *Speech Enhancement, Modeling and Recognition–Algorithms and Applications*, InTech, 2012.

[Nur13a] J. Nurminen, H. Silén, and M. Gabbouj, Speaker-specific retraining for enhanced compression of unit selection text-to-speech databases. In *Proc. of Interspeech*, 2013.

[Nur13b] J. Nurminen, H. Silén, E. Helander, and M. Gabbouj, Evaluation of detailed modeling of the LP residual in statistical speech synthesis. In *Proc. of ISCAS*, 2013.

[O'B01] D. O'Brien and A. Monaghan, Concatenative synthesis based on a harmonic model. *IEEE Transactions on Speech and Audio Processing*, 9(1):pp. 11–20, 2001.

[Ohm93] H. Ohmuro, T. Moriya, K. Mano, and S. Miki, Coding of LSP parameters using interframe moving average prediction and multi-stage vector quantization. In *Proc. of IEEE Workshop on Speech Coding for Telecommunications*, pp. 63–64, 1993.

[Oli48] B. Oliver, J. Pierce, and C. Shannon, The philosophy of PCM. *Proceedings of the IRE*, 36(11):pp. 1324–1331, 1948.

[Oza94] K. Ozawa, M. Serizawa, T. Miyano, and T. Nomura, M-LCELP speech coding at 4 kbps. In *Proc. of ICASSP*, pp. 269–272, 1994.

[Pak97] E. Paksoy, A. McCree, and V. Viswanathan, A variable-rate multimodal speech coder with gain-matched analysis-by-synthesis. In *Proc. of ICASSP*, pp. 751–754, 1997.

[Pal93] K. Paliwal and B. Atal, Efficient vector quantization of LPC parameters at 24 bits/frame. *IEEE Transactions on Speech and Audio Processing*, 1(1):pp. 3–14, 1993.

[Par87] T. Parsons, *Voice and speech processing*. McGraw-Hill, New York, 1987.

[Pic89] J. Picone and G. Doddington, A phonetic vocoder. In *Proc. of ICASSP*, pp. 580–583, 1989.

[Pol08] V. Pollet and A. Breen, Synthesis by generation and concatenation of multiform segments. In *Proc. of Interspeech*, 2008.

[Pop09] V. Popa, J. Nurminen, and M. Gabbouj, A novel technique for voice conversion based on style and content decomposition with bilinear models. In *Proc. of Interspeech*, pp. 2655–2658, 2009.

[Pop11] V. Popa, J. Nurminen, and M. Gabbouj, A study of bilinear models in voice conversion. *Journal of Signal and Information Processing*, 2(2):pp. 125–139, 2011.

[Pop12] V. Popa, H. Silen, J. Nurminen, and M. Gabbouj, Local linear transformation for voice conversion. In *Proc. of ICASSP*, pp. 4517–4520, 2012.

[Pre92] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge University Press, Cambridge, 1992.

[Qua92] T. Quatieri and R. McAulay, Shape invariant time-scale and pitch modification of speech. *IEEE Transactions on Signal Processing*, 40(3):pp. 497–510, 1992.

[Qua02] T. Quatieri, *Discrete-time speech processing: principles and practice*. Prentice-Hall, NJ, 2002.

[Rab89] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):pp. 257–286, 1989.

[Rab93] L. Rabiner and B.-H. Juang, *Fundamentals of speech recognition*. Prentice-Hall, USA, 1993.

[Rad99] A. Radford, M. Atkinson, D. Britain, H. Clahsen, and A. Spencer, *Linguistics: An Introduction*. Cambridge University Press, Cambridge, 1999.

[Rai10] T. Raitio, A. Suni, J. Yamagishi, H. Pulakka, J. Nurminen, M. Vainio, and P. Alku, HMM-based speech synthesis utilizing glottal inverse filtering. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(1), 2010.

[Räm04] A. Rämö, J. Nurminen, S. Himanen, and A. Heikkinen, Segmental speech coding model for storage applications. In *Proc. of Interspeech*, pp. 2677–2680, 2004.

[Räm05] A. Rämö, J. Nurminen, S. Himanen, and A. Heikkinen, Method and system for speech coding, US Patent application 20050091041. Apr. 2005.

[Ram06] V. Ramasubramanian and D. Harish, An unified unit-selection framework for ultra low bit-rate speech coding. In *Proc. of Interspeech*, 2006.

[Ram09] V. Ramasubramanian and D. Harish, Ultra low bit-rate speech coding based on unit-selection with joint spectral-residual quantization: no transmission of any residual information. In *Proc. of Interspeech*, pp. 2615–2618, 2009.

[Ram12] V. Ramasubramanian, Ultra low bit-rate speech coding: An overview and recent results. In *Proc. of International Conference on Signal Processing and Communications*, pp. 1–5, 2012.

[Ras99] U. Rass and G. Steeger, Reducing time domain aliasing in adaptive overlap-add algorithms. In *Proc. of 138th Meeting of the Acoustical Society of America*, 1999.

[Rem97] R. E. Remez, J. M. Fellowes, and P. E. Rubin, Talker identification based on phonetic information. *Journal of Experimental Psychology: Human Perception and Performance*, 23(3):pp. 651–666, 1997.

[Ris76] J. Rissanen, Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):pp. 198–203, 1976.

[Ros93] K. Rose and D. Miller, A deterministic annealing algorithm for entropy-constrained vector quantizer design. In *Proc. of Asilomar Conference on Signals, Systems and Computers*, pp. 1651–1655, 1993.

[Ros98a] K. Rose, Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):pp. 2210–2239, 1998.

[Ros98b] K. Rose, H. Khalil, and S. Regunathan, Open-loop design of predictive vector quantizers for video coding. In *Proc. of ICIP*, pp. 953–957, 1998.

[Rou82] S. Roucos, R. Schwartz, and J. Makhoul, Segment quantization for very-low-rate speech coding. In *Proc. of ICASSP*, pp. 1565–1568, 1982.

[Rou83] S. Roucos, R. Schwartz, and J. Makhoul, A segment vocoder at 150 b/s. In *Proc. of ICASSP*, pp. 61–64, 1983.

[Ruo00] V. Ruoppila, M. Tammi, and J. Saarinen, Waveform extraction for perfect reconstruction in WI coding. In *Proc. of ICASSP*, pp. 1359–1362, 2000.

[Rut02] P. Rutten, M. Aylett, J. Fackrell, and P. Taylor, A statistically motivated database pruning technique for unit selection synthesis. In *Proc. of Interspeech*, pp. 125–128, 2002.

[Sar08] K. Sarathy and A. Ramakrishnan, A research bed for unit selection based text to speech synthesis. In *Proc. of IEEE Spoken Language Technology Workshop*, pp. 229–232, 2008.

[Sch79] M. Schroeder, B. Atal, and J. Hall, Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of the Acoustical Society of America*, 66(6):pp. 1647–1652, 1979.

[Sch85] M. Schroeder and B. Atal, Code-excited linear prediction(CELP): High-quality speech at very low bit rates. In *Proc. of ICASSP*, pp. 937–940, 1985.

[Sch02] M. Schnell, M. Kustner, O. Jokisch, and R. Hoffmann, Text-to-speech for low-resource systems. In *IEEE Workshop on Multimedia Signal Processing*, pp. 259–262, 2002.

[Sha59] C. Shannon, Coding theorems for a discrete source with a fidelity criterion. *IRE International Convention Records*, 7 (part 4), 1959.

[Shi88] Y. Shiraki and M. Honda, LPC speech coding based on variable-length segment quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(9):pp. 1437–1444, 1988.

[Shl98] E. Shlomot, V. Cuperman, and A. Gersho, Combined harmonic and waveform coding of speech at low bit rates. In *Proc. of ICASSP*, pp. 585–588, 1998.

[Shu06] Z. Shuang, R. Bakis, and Y. Qin, Voice conversion based on mapping formants. In *TC-STAR Workshop on Speech-to-Speech Translation*, pp. 219–223, 2006.

[Sil09] H. Silén, E. Helander, J. Nurminen, and M. Gabbouj, Parameterization of vocal fry in HMM-based speech synthesis. In *Proc. of Interspeech*, 2009.

[Sil10] H. Silén, E. Helander, J. Nurminen, K. Koppinen, and M. Gabbouj, Using robust Viterbi algorithm and HMM-modeling in unit selection TTS to replace units of poor quality. In *Proc. of Interspeech*, 2010.

[Sil13] H. Silén, J. Nurminen, E. Helander, and M. Gabbouj, Voice conversion for non-parallel datasets using dynamic kernel partial least squares regression. In *Proc. of Interspeech*, 2013.

[Sko97] J. Skoglund and J. Linden, Predictive VQ for noisy channel spectrum coding: AR or MA? In *Proc. of ICASSP*, pp. 1351–1354, 1997.

[Son11] P. Song, Y. Bao, L. Zhao, and C. Zou, Voice conversion using support vector regression. *Electronics Letters*, 47(18):pp. 1045–1046, 2011.

[Soo93] F. Soong and B. Juang, Optimal quantization of LSP parameters. *IEEE Transactions on Speech and Audio Processing*, 1(1):pp. 15–24, 1993.

[Spa94] A. Spanias, Speech coding: A tutorial review. *Proceedings of the IEEE*, 82(10):pp. 1541–1582, 1994.

[Sri98] S. Sridharan and J. Leis, Two novel lossless algorithms to exploit index redundancy in VQ speech compression. In *Proc. of ICASSP*, pp. 57–60, 1998.

[Sty98] Y. Stylianou, O. Cappe, and E. Moulines, Continuous probabilistic transform for voice conversion. *IEEE Transactions on Speech and Audio Processing*, 6(2):pp. 131–142, 1998.

[Sty00] Y. Stylianou, On the implementation of the harmonic plus noise model for concatenative speech synthesis. In *Proc. of ICASSP*, pp. 957–960, 2000.

[Sty01] Y. Stylianou, Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing*, 9(1):pp. 21–29, 2001.

[Sün03] D. Sündermann and H. Ney, VTLN-based voice conversion. In *Proc. of ISSPIT*, pp. 556–559, 2003.

[Sün04a] D. Sündermann, A. Bonafonte, H. Ney, and H. Höge, Voice conversion using exclusively unaligned training data. In *Proc. of ACL/EMNLP*, 2004.

[Sün04b] D. Sündermann, A. Bonafonte, H. Ney, and H. Höge, A first step towards text-independent voice conversion. In *Proc. of Interspeech*, 2004.

[Sun10] J. Sun and J. Ouyang, Codebook design of vector quantization based on improved particle swarm optimization. In *Proc. of ICACTE*, pp. 470–473, 2010.

[Sup97] L. Supplee, R. Cohn, J. Collura, and A. McCree, MELP: the new federal standard at 2400 bps. In *Proc. of ICASSP*, pp. 1591–1594, 1997.

[Tan88] T. Taniguchi, S. Unagami, and R. Gray, Multimode coding: A novel approach to narrow- and medium-band coding. *Journal of the Acoustical Society of America*, 84:p. S12, 1988.

[Tao10] J. Tao, M. Zhang, J. Nurminen, J. Tian, and X. Wang, Supervisory data alignment for text-independent voice conversion. *IEEE Transactions Audio, Speech, and Language Processing*, 18(5):pp. 932–943, 2010.

[TC-13] TC-STAR, TC-STAR–Technology and corpora for speech to speech translation. Online, accessed Apr. 12, 2013, available: http://www.tcstar.org/, 2013.

[Ter79] E. Terhardt, Calculating virtual pitch. *Hearing research*, 1:pp. 155–182, 1979.

[Tia06] J. Tian, J. Nurminen, F. Ding, and I. Kiss, Modular design for Mandarin text-to-speech synthesis. In *Proc. of Workshop on Speech-To-Speech Translation*, pp. 187–191, 2006.

[Tia08] J. Tian, V. Popa, and J. Nurminen, Efficient model re-estimation in voice conversion. In *Proc. of EUSIPCO*, 2008.

[Tia09] J. Tian, J. Nurminen, and V. Popa, Soft alignment based on a probability of time alignment, US Patent 7505950. Mar. 2009.

[Tia10] J. Tian, J. Nurminen, and V. Popa, Method, apparatus, mobile terminal and computer program product for providing data clustering and mode selection, US Patent 7725411. May 2010.

[Tia12] J. Tian, V. Popa, and J. Nurminen, Hybrid approach in voice conversion, US Patent 8224648. Jul. 2012.

[Tio11] S. Tiomkin, D. Malah, S. Shechtman, and Z. Kons, A hybrid text-to-speech system that combines concatenative and statistical synthesis units. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(5):pp. 1278–1288, 2011.

[Tod07] T. Toda, A. Black, and K. Tokuda, Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(8):pp. 2222–2235, 2007.

[Tok94] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai, Mel-generalized cepstral analysis - a unified approach to speech spectral estimation. In *Proc. of ICSLP*, 1994.

[Tok02] K. Tokuda, H. Zen, and A. Black, An HMM-based speech synthesis system applied to English. In *Proc. of 2002 IEEE Workshop on Speech Synthesis*, pp. 227–230, 2002.

[Tre82] T. Tremain, The government standard linear predictive coding algorithm. *Speech Technology Magazine*, pp. 40–49, 1982.

[Tsa85] C. Tsao and R. Gray, Matrix quantizer design for LPC speech using the generalized Lloyd algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-33(3):pp. 537–545, 1985.

[Tsi08] P. Tsiakoulis, A. Chalamandaris, S. Karabetsos, and S. Raptis, A statistical method for database reduction for embedded unit selection speech synthesis. In *Proc. of ICASSP*, pp. 4601–4604, 2008.

[Tso97] D. Tsoukalas, J. Mourjopoulos, and G. Kokkinakis, Speech enhancement based on audible noise suppression. *IEEE Transactions on Speech and Audio Processing*, 5(6):pp. 497–514, 1997.

[Tur03] O. Turk and L. Arslan, Voice conversion methods for vocal tract and pitch contour modification. In *Proc. of Interspeech*, pp. 2845–2848, 2003.

[Tur06] O. Turk and L. Arslan, Robust processing techniques for voice conversion. *Computer Speech and Language*, 4(20):pp. 441–467, 2006.

[Uns99] M. Unser, Splines: a perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):pp. 22–38, 1999.

[Unv10] E. Unver, S. Villette, and A. Kondoz, Joint quantisation strategies for low bit-rate sinusoidal coding. *Signal Processing, IET*, 4(5):pp. 548–559, 2010.

[Ver96] T. Verma, S. Bilbao, and T. Meng, The digital prolate spheroidal window. In *Proc. of ICASSP*, pp. 1351–1354, 1996.

[Vil01] S. Villette, *Sinusoidal speech coding for low and very low bit rate applications*. Ph.D. thesis, University of Surrey, October 2001.

[Vir99] N. Vira, Single channel speech enhancement based on masking properties of the human auditory system. *IEEE Transactions on Speech and Audio Processing*, 7(2):pp. 126–137, 1999.

[Wat02] T. Watanabe, T. Murakami, M. Namba, T. Hoya, and Y. Ishida, Transformation of spectral envelope for voice conversion based on radial basis function networks. In *Proc. of Interspeech*, pp. 285–288, 2002.

[Wik13] Wikipedia, Speech production. Online, accessed Apr. 12, 2013, available: http://en.wikipedia.org/wiki/Speech_production, 2013.

[Won92] D.-K. Wong, Issues on speech storage. In *Proc. of IEE Colloquium on Speech Coding - Techniques and Applications*, pp. 711–714, 1992.

[Wu10] Z.-Z. Wu, T. Kinnunen, E. Chng, and H. Li, Text-independent F0 transformation with non-parallel data for voice conversion. In *Proc. of Interspeech*, pp. 1732–1735, 2010.

[Yan98] H. Yang, W. Kleijn, E. Deprettere, and Y. Chen, Pitch synchronous modulated lapped transform of the linear prediction of residual speech. In *Proc. of ICSP*, pp. 591–594, 1998.

[Yan09] C. Yang and G. Wei, Sinusoidal parameters estimation in speech sinusoidal model. In *Proc. of International Conference on Machine Learning and Cybernetics*, 2009.

[Zeg90] K. Zeger and A. Gersho, Pseudo-gray coding. *IEEE Transactions on Communications*, 38(12):pp. 2147–2158, 1990.

[Zeg91] K. Zeger, Corrections to 'Gradient algorithms for designing predictive vector quantizers'. *IEEE Transactions on Signal Processing*, 39(3):pp. 764–765, 1991.

[Zwi90] E. Zwicker and H. Fastl, *Psychoacoustics*. Springer-Verlag, Berlin, Germany, 1990.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-3136-1
ISSN 1459-2045