Quality-Preserving Footprint-Reduction of Concatenative Text-To-Speech Synthesizers

Tamar Shoham

Quality-Preserving Footprint-Reduction of Concatenative Text-To-Speech Synthesizers

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering

Tamar Shoham

Submitted to the Senate of
the Technion — Israel Institute of Technology
Iyar 5770 Haifa May 2010

The research thesis was done under the supervision of Prof. David Malah in the Electrical Engineering Department.

I would like to express my deepest gratitude to my supervisor, Prof. David Malah, who has been a true mentor. I am privileged to have had his devoted guidance and to have benefited from his wealth of knowledge.
Many thanks to the devoted Signal and Image Processing Lab (SIPL) staff: Nimrod Peleg, Yair Moshe, Ziva Avni and Avi Rozen who jointly create the environment which enabled and encouraged this research.
I would also like to thank the speech technology group at IBM-HRL who instigated and supported this research. Ron Hoory the group leader, Slava Shechtman who invested many hours of help, Ariel Sagi, Zvi Kons, and the entire team for their ongoing interest and support.

On a more personal note, I would like to thank my husband Shlomo for never giving up, even when I would. Many thanks to my sister, parents and in-laws for their endless help and support. And finally, to my beloved children Asael-Neryah, Na'ama-Sara and Elyassaf-Benayah who make it all worthwhile.

Contents

List of Figures

Abstract

High quality low footprint Concatenative Text-To-Speech (CTTS) synthesizers provide a persistent challenge in the field of speech processing. The spectral parameters representing the short speech segments, used in the concatenation process, constitute a large portion of the required memory. In this work we propose algorithms for (re)compression of this previously compressed data, stored in 3D acoustic leaves. We require that the algorithms be generic, so that they may be applied for (re)compression of other data that is stored in 3D units or structures that exhibit some redundancy, primarily due to temporal evolution, but possibly along all three axes. Since the requirement for small footprint, i.e., small memory consumption, often corresponds to devices with limited resources, we also limit the algorithms to have low decoding complexity.

We propose two (re)compression approaches. The first is an algorithm from the family of Temporal Decomposition, which aims to minimize temporal redundancy between consecutive frames. We use Polynomial TD and perform data segmentation and polynomial order selection adaptively using a generalized trellis scheme. The second approach is based on 3D-Shape Adaptive DCT that removes redundancies in all three dimensions. This approach requires design of quantizers for 3D data, which we address by developing a methodical bit-allocation and splitting algorithm. We also propose a segment reordering algorithm, which may be applied before the (re)compression, to order the speech segments in a manner that will maximize overall performance. The proposed algorithms were evaluated on an IBM small footprint CTTS system, and enabled reduction of the stored amplitude spectral parameters (the main contributor to the footprint) by a factor of 2, without compromising the perceptual quality of the obtained speech. While we tested the proposed algorithms on a specific setup, they are expected to apply to a variety of 3D (re)compression challenges.

Abbreviations and Notations

BSA	—	Binary Switching Algorithm, used for segment ordering
$CELP$	—	Code Excited Linear Prediction (speech coder)
$CTTS$	—	Concatenative Text To Speech
DCT	—	Discrete Cosine Transform
LSD	—	Log Spectral Distortion (a speech quality measure)
LBG	—	Linde - Buzo - Gray algorithm for quantizer design
LSF	—	Line Spectral Frequencies
$MBOrd$	—	A proposed Metropolis Based Ordering algorithm
$MELP$	—	Mixed-Excitation Linear Predictive (speech coder)
$MFCC$	—	Mel-frequency Cepstral Coefficients used to represent speech spectrum
MSE	—	Mean Square Error
$PESQ$	—	Perceptual Evaluation of Speech Quality (ITU-T P.862.2)
		[By default we refer to the wide-band version]
$SADCT$	—	Shape Adaptive Discrete Cosine Transform
TD	—	Temporal Decomposition
TTS	—	Text To Speech
VQ	—	Vector Quantizer or Quantization
SA	—	Simulated Annealing
$SADCT$	—	Shape Adaptive Discrete Cosine Transform
STD	—	Standard Deviation

$A(\omega), A(f)$	—	Spectral amplitude of a speech frame
B_n	—	Basis functions used to derive the representing speech parameters
C	—	A cost associated with a specific segment
C_n	—	Set of normalized parameters representing speech frame spectrum
D_F	—	Distortion of a single speech frame
D_g	—	Global distortion used in the polynomial TD optimiaztions
f	—	Frequency of the speech spectral representation
f'	—	Warped frequency using the Mel-scale
$F_{u,v,w}$	—	3D SADCT coefficient values
G_i	—	Group of SADCT coefficient vectors (i=1,..,5)
N	—	Number of consecutive frames used for TD (TD segment length)
P	—	Polynomial order
R	—	Rate
R_t	—	Target rate
$S_{i,j}$	—	A state in the generalized trellis used for polynomial TD optimization
T	—	Temperature, used in the reordering algorithm
V_P	—	Vandermonde matrix f order P
$W_i, W_{i,j}$	—	Weights used to prioritize low frequency data

Chapter 1

Introduction

The goal of this research was to further reduce the footprint, a.k.a. memory consumption, of a small footprint Concatenative-Text-To-Speech synthesizer, without perceptually degrading the quality of the synthesized speech. Most of the system footprint is due to the segment inventory which stores the short speech segments used for the speech synthesis. Therefore, we propose to take these compressed speech segments, and re-compress them further, by removing redundancies between speech frames, and possibly also redundancies between speech segments and among the parameters representing each speech frame. We refer to this as re-compression, since it applies to data that has already undergone one level of compression, rather than to raw data. We add two constraints to our (re)compression scheme. The first is that we would like to develop a generic approach, which will be applicable to a variety of recompression challenges. We target databases that consist of data sets stored in 2D or 3D units, or structures, that exhibit redundancy. The redundancy may be due to slowly evolving data that has temporal redundancy, or redundancy due to correlation between data elements in the structure. For a 3D data structure we may have redundancies along all three axis. The second constraint is that the (re)compression algorithms should have low decoding complexity, so as to enable real-time decoding on devices with limited resources.

In order to evaluate the proposed algorithms, we will take a specific small footprint Concatenative Text-To-Speech (CTTS) synthesizer, and reduce its footprint further, without compromising the perceptual quality of the obtained speech.

To enable an understanding of the platform on which the proposed al-

gorithms were evaluated, we will first describe the structure of a CTTS synthesizer. Then we will provide some details on the specific IBM system we used. This is followed by an overview of previous research in this field, and then an outline of the structure of this thesis.

1.1 Concatenative Text-To-Speech Synthesizers

The goal of a Text-To-Speech (TTS) synthesizer, is to create synthesized speech based on an input text sequence. TTS synthesizers generally belong to one of two categories, rule based or corpus based as explained in detail in [6]. Corpus based synthesizers usually use a concatenative approach, where short speech segments, stored in the segment inventory or database, are selected, processed and combined to create the synthesized speech. A standard Concatenative TTS (CTTS) system is illustrated in Figure 1.1 (taken from [9]). The main components of this system are the front end which performs the text analysis, the speech segment inventory or database, and various speech processing modules.

The speech segment inventory is created by taking many hours of recorded speech, decomposing it into short speech segments, i.e. sub-phonemes, and clustering these sub-phonemes into acoustic leaves. Each acoustic leaf holds a number of speech segments, all corresponding to the same sub-phoneme in the same context.

The speech synthesis is performed with the following steps: The input text enters the 'front-end', which performs phonetic analysis to determine the sub-phonemes that make up the target speech, and determines the appropriate pitch, energy, and duration of each sub-phoneme as well as providing various context parameters that help select the most appropriate speech segment. Then, for each sub-phoneme in the target sequence, a speech segment is selected from the segment inventory, or database. The speech segment selection is performed in two stages: First, according to the required sub-phoneme and its context the appropriate acoustic leaf is selected. Then out of the multiple speech segments stored in the acoustic leaf, the one that provides the lowest target and concatenation costs is selected. A detailed description of how the segment selection may be performed, using dynamic programming, can be found in [13]. Various signal processing modules perform reconstruction of the speech segments from their stored parametric representation, various morphing operations and the actual concatenation

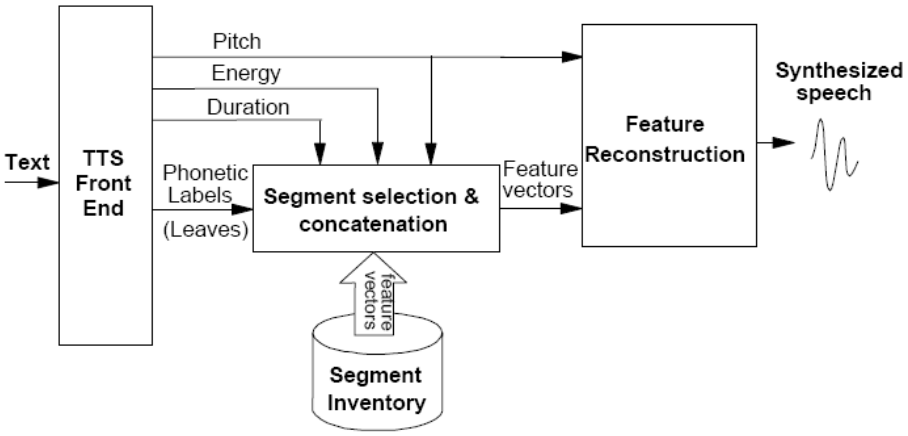

Figure 1.1: Illustration of a Concatenative Text-To-Speech synthesizer (as provided in [9]).

(usually using an OverLap and Add approach).

1.2 IBM Small Footprint CTTS Synthesizer

In order to implement and evaluate the proposed (re)compression algorithms, we applied them to the database of a small footprint IBM CTTS system. A description of this system can be found in [13], with the description of the efforts towards reducing the footprint in [9], [8]. (The system we used also combines improved high band synthesis via frequency data manipulations as described in [10], which is less relevant to our compression scheme since we work on the amplitude spectral parameters only). The CTTS synthesizer achieves low footprint using two complementing tools:

1. Careful selection of the speech segments to be stored in the acoustic leaf database: only the most frequently used speech segments are kept.

2. Compact representation of each speech frame using a parametric spectral model.

Each acoustic leaf in the database consists of the 5-10 "pre-selected" speech segments, that correspond to the sub-phoneme and context of the

leaf. The segments are selected by applying a statistical pruning process, where the frequency of each speech segment in synthesizing a large text corpus is evaluated, and the most frequently used speech segments are kept. Each speech segment comprises of 1-35 speech frames, with a median frame number of 2. For each frame of 10msec duration, the following spectral modeling is performed, resulting in a set of 32 amplitude parameters and a varying length set of phase parameters.

The speech analysis process examines the polar form of the complex spectral envelope of the speech frame:

$$S(f) = A(f)e^{j\varphi(f)} \tag{1.1}$$

Amplitude and phase are modeled separately using similar algorithms - we describe the amplitude modeling since these are the parameters we focused on, as they are responsible for most of the system footprint. The phase modeling uses similar concepts, though the length of the phase parameter vector depends on the speech frame type - voiced, unvoiced or *clicks*- such as plosives or fricatives. Further details can be found in [8].

To obtain high perceptual quality while maintaining simplicity, the spectrum amplitude is analyzed on a warped frequency scale, using the Mel-scale mapping [48]:

$$f' = 2595 \cdot log_{10}\left(1 + \frac{f}{700}\right) \tag{1.2}$$

The log-amplitude spectrum of each frame is modeled by a linear combination of L basis functions:

$$\log\left(A(f')\right) = \sum_{n=1}^{L} c_n \cdot B_n(f') \tag{1.3}$$

The basis functions used, B_n, are triangular and in the warped Mel-scale have equal widths and half overlap each other. A schematic example of four such functions, shown on the normalized (non warped) frequency scale is provided in Figure 1.2. In practice 32 such functions are used.

During speech analysis, we calculate the parameters c_n that minimize the squared error between a frequency-warped and re-sampled version of the amplitude line spectrum, and its parameter based reconstruction. In order to preserve the original frame energy, G, without adding another parameter,

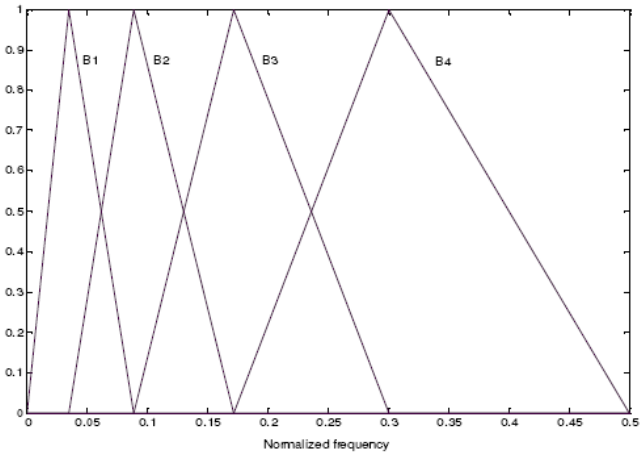

Figure 1.2: Illustration of the basis functions used for IBM small footprint CTTS spectral modeling, illustrated for L=4, as shown in [8]. In practice, for the system we used, L=32

it is embedded into the optimal c_n values, defined above, as follows:

$$C_n = c_n - \frac{1}{L}\left(\sum_{i=1}^{L} c_i - \log G\right) \tag{1.4}$$

The log-amplitude spectrum of each frame is therefore represented by the set of values C_n. These are the final model parameters or features, which are the input to our (re)compression algorithm.

In the IBM system we used, the vectors C_n undergo quantization, with an 86 bit quantizer. Differential coding is performed, which, since these parameters are on a logarithmic scale, is equivalent to coding the parameter ratios. The differences are calculated in multiple stages, starting with the vector of 32 amplitude parameters (V_0). At each stage the input vector is transformed to an output vector, according to the following: First stage:

$$V_1[2n] = V_0[2n+1] + V_0[2n];\ V_1[2n+1] = V_0[2n+1] - V_0[2n];\ n = 0, 1, ..., 15. \tag{1.5}$$

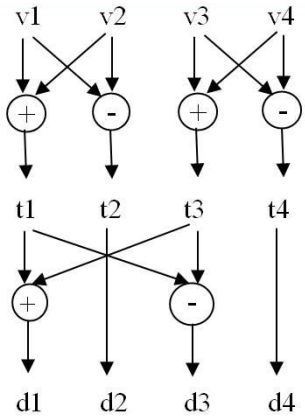

Figure 1.3: Illustration of the differential coding scheme used for quantization of the spectral amplitude parameters in the IBM small footprint CTTS system. Input vector: [v1, v2, v3, v4], Output vector :[d1, d2, d3, d4]. (Note d2 and d4 are passed through from the second stage).

Second stage:

$$V_2[4n] = V_1[4n+2] + V_1[4n];\ V_2[4n+1] = V_1[4n+2] - V_1[4n];\ n = 0, 1, ..., 7. \tag{1.6}$$

This continues until $V[0]$ holds the sum of all the vector values, and the differences from each stage are stored and inserted into the final vector. Note that at each stage we work on less and less elements, and since the calculations are performed in place, at the last stage we have the complete vector ready for coding, though its order is somewhat un-intuitive. This process is demonstrated in Figure 1.3 for a vector of length 4.

The obtained difference vector is then coded using a Split Vector Quantization (VQ) approach. The system we used for evaluation, which codes each 32 feature vector with 86 bits, uses the setup shown in Table 1.1. Note that the sum of coefficients, which contains the vector energy, is coded separately.

To summarize: Each speech frame is represented by a vector of 32 quantized parameters. Each of these vectors represents the spectral amplitude of a single speech frame. Each frame belongs to a speech segment, which in turns belongs to a specific acoustic leaf. The database we used for the eval-

Table 1.1: Bit allocation in IBM small footprint CTTS quantizer. The indices refer to the indices in the obtained difference vector.

Vector length	Num. bits	indices
4	10	1, 3, 5, 7
4	10	9, 11, 13, 15
4	8	17, 19, 21, 23
4	7	25, 27, 29, 31
3	10	2, 4, 6
3	9	10, 12, 14
2	10	8, 16
4	7	18, 22, 26, 30
3	7	20, 24, 28
1	8	0

uation of the proposed algorithm consists of 23,263 such leaves, with 1661 of them used to synthesize the 10 test sentences evaluated in our simulations. This concludes the description of our test data.

1.3 Prior Art

There is an abundance of previous work that can be applied to the problem of data compression, where the data is stored in 3D structures that exhibit redundancy in at least one of the three dimensions, or more specifically, (re)compression of CTTS acoustic leaf data. If we were to apply the compression directly to the speech frames or segments stored in the acoustic leaf, rather than the to the parameters of a selected spectral model as described in the previous section, many possible speech coding algorithms could be applied. These include, but are not limited to, sinusoidal based models as first introduced by McAulay and Quatiere in [36] and [37], or one of its various expansions to Harmonic plus Noise models such as the one proposed by Stylianou in [56]. An adaptation of the sinusoidal model specifically for TTS applications was proposed by Macon and Clements in [32] and [33]. Other appropriate models are a decomposition of the spectra into periodic and a-periodic components was described by d'Alessandro *et* al. in [12] and enhanced by Ahn and Holmes in [2]. Iterative signal subtraction for analysis by synthesis using the sinusoidal model was proposed by George and

Smith in [17] with a minor enhancement by Bailly in [4]. The advantage of these approaches is that pitch modification or speech morphing, which is often required for CTTS, is quite easily integrated within a sinusoidal speech model. A slightly different approach is to use MFCC, Mel-frequency Cepstral Coefficients, to model the speech spectrum as proposed in [9]. Other small footprint CTTS systems, such as the one described in [27], propose to use standard speech coding techniques, such as Code Excited Linear Prediction (CELP). A disadvantage of this approach is that since we store short, discontinuous, speech segments the prediction efficiency of this algorithm is limited. Using any of these speech models may enable better compression of the speech frames, but will probably leave inter-frame and/or inter-segment redundancies. Therefore, these models could serve as an alternative starting point for the proposed algorithms, which attempt to remove the remaining redundancies.

A somewhat different approach to the compression of an acoustic leaf database, or acoustic unit inventory compression, was presented by Kain and van Santen in [24], [25] and [26]. They propose to approximate each diphone, an adjacent pair of phones, by interpolating between a left and right phoneme template. These templates are acquired by collecting all the relevant representing frames of a specific diphone onset or end, i.e., all the left frames of diphones that begin with a specific phoneme, or alternatively, all the right frames of diphones that end with a specific phoneme. The corresponding template is set to the complex spectrum of the frame that is identified as the centroid of these frames in the log-magnitude spectrum domain [24]. For diphone synthesis two asynchronous, non-linear, interpolation operations are applied to the left and right templates, using two sets of evolving transition weights, which are estimated and stored for each frequency component of each speech frame of each diphone in the acoustic inventory. Using this approach provides high compression ratios, but this comes at the price of poor perceptual quality and low flexibility, as the algorithm provides a single possible working point. The process of moving from frame by frame representation of phones and diphones to the template based approach incurs non-reversible loss which causes a perceptible quality degradation. Creating the template database is also a complex process, which means that changing speakers for instance has a high algorithmic overhead.

These approaches were not appropriate for our case, where we wish to

use the parameters stored in the acoustic leaves of a specific CTTS synthesizer. We will therefore examine approaches that can fit seamlessly into an existing system, by using pre-determined spectral model parameters.

Regarding possible algorithms for recompression of a set of per-frame spectral parameters, a variety of works exist on coding spectral parameters using Gaussian Mixture Model based quantization. The earlier works by Hedelin [21] and Subramaniam [58], [57], have been expanded by Linblom and Stylianou in [23] and [1], accordingly, for usage in sinusoidal based spectral modeling, dealing with the varying vector dimensionality. These approaches propose improved quantization of each speech frame, but do not take advantage of slow temporal volution or remove temporal redundancies.

We chose to concentrate on alternatives that allow reuse of the existing parameter vectors, and focus on re-compressing these by removing inter-frame redundancies. We propose two different approaches for this problem. The first uses Temporal Decomposition to remove long-term redundancy, by modeling the trajectory of the parameters along frames. This approach enables reuse of the existing quantizer. The second uses the 3D Shape Adaptive Discrete Cosine Transform to remove redundancies along all three axis of the acoustic leaf. For this algorithm we will need to design a new quantizer, that is suitable for the new type of data (DCT coefficients). We will present the algorithms details and the relevant prior art in the following chapters.

1.4 Thesis Structure

This thesis is structured as follows. Having introduced the research goal, and the basics of the CTTS synthesizer database on which we evaluate the proposed algorithms, we will next present the two proposed (re)compression algorithms.

Chapter 2 elaborates on a Temporal Decomposition approach for long-term (re)compression, presenting first the previous works in this field, followed by the polynomial TD proposed in [14], [15]. We then present our expansion of this algorithm to a vectorial form where the segment length and model order are adapted to local data behavior rather than being selected in advance.

Chapter 3 presents an alternative approach, based on 3D SADCT, (used previously in 3D in the context of hyperspectral image coding in [35]), to

remove redundancies along all 3 axis of the acoustic leaf.

Chapter 4 presents an optimal ordering algorithm, which may be applied to the speech segments prior to applying either of the two proposed recompression algorithms.

Chapter 5 presents experimental results of the algorithms presented in this thesis. We show that we can recompress the spectral amplitude data stored in the acoustic leaf database of our test system, by a factor of 2, without introducing perceptual quality loss.

In Chapter 6 we conclude and suggest some future research directions.

Chapter 2

Compression using Polynomial Temporal Decomposition

2.1 Overview

In this chapter we describe one of the proposed algorithms for acoustic leaf compression - Polynomial Temporal Decomposition. We begin by introducing a range of Temporal Decomposition algorithms, then describe the proposed algorithm and its components, including an algorithm to perform joint segmentation and polynomial order selection. We proceed to describe a number of different proposed algorithm configurations and then conclude. We will also refer to the compression algorithm as *recompression*, to indicate that the compression is performed on the acoustic leaf data which has already undergone compression as described in Chapter 1, rather than operating on raw data.

2.2 Temporal Decomposition

Temporal Decomposition (TD) describes a set of compression methods which attempt to exploit the temporal redundancy in the data by representing the evolution over time, or between frames, of either a scalar parameter (scalar TD) or a parameter vector (vector TD). The representation of the parameter trajectory is achieved with a reduced order model, thus achieving

compression.

In the remainder of this section, we briefly describe some existing vector and scalar TD approaches, wrapping up with the Polynomial based Temporal Decomposition which we adopted in our research.

2.2.1 Vector TD

When applying vector TD for speech compression, each speech segment, consisting of a number of subsequent speech frames, is jointly represented by a set of events. The set consists of event locations, event targets and event functions. The locations describe the time index of the events, the targets hold the event data or feature vector and the event functions describe the interpolation functions for time indices that lie between event locations.

Vector TD has been used for low rate speech coding in a number of previous works. In [3], Athaudage *et* al. present a dynamic programming based optimization strategy for optimal event localization. All possible locations for the k_{th} event are evaluated, assuming the optimal locations for all other events are known. The suggested TD model is incorporated into the standard MELP speech coder for evaluation of compression efficiency. Shechtman and Malah in [52] and [51] propose a computationally efficient sub-optimal approach to the TD modeling, as well as a perceptually weighted error criterion for improved perceptual performance. They also attempted to apply the TD model not only to the spectral parameters, but also to the MELP excitation parameters. Their work was based partly on the LEBEL-TD algorithm presented by Nguyen in 2002 and re-presented with more detailed evaluations in [41].

These works report about a 50% compression gain, when applied to spectral parameters, with equivalent quality. They also conclude that spectral parameters are more suited to TD than the excitation parameters. The efficient sub-optimal approach of [51] displays a low penalty in quality and the perceptual weighting improves perceptual quality of the reconstructed speech. Note that to obtain this performance, these algorithms generally require at least 10 consecutive frames per segment, which is not the case for our database. Another limitation of these vector TD approaches is that the interpolation functions are uniform for all vector elements. Therefore they do not apply well to cases where different elements in the vector have different trajectories, which is often the case for the CTTS acoustic leaf

data.

2.2.2 DCT Based Scalar TD

In [19] Girin et al. present a DCT based, long-term model for the trajectories of the amplitude and phase parameters of sinusoidal coding parameters. The DCT model order is found iteratively, by minimizing a perceptually weighted cost function. The algorithm was applied and evaluated on the first 10 partials of voiced frames only. When using short, pitch synchronized windows, the authors showed a reduction factor of 7 in the amount of data required for the first ten harmonics. When using fixed length windows of 10-20ms, a reduction factor of 2-4 was obtained. While this work offers an interesting approach, it doesn't seem to extend well to parameters that exhibit higher randomness such as unvoiced speech or the higher partials. It also requires reasonably large segments - the authors used 20 consecutive frames in their evaluations.

2.2.3 Polynomial Based Scalar TD

In [14] and [15] Dusan et al. present an alternative scalar TD approach. The trajectories of various speech features (i.e., spectral coefficients, pitch and gain) along the segment are each modeled by an approximating polynomial. The trajectory is then represented by the approximating polynomial samples. These papers successfully apply the proposed algorithm also to noisy speech signals. In polynomial scalar TD, a segment consisting of N consecutive analysis frames is chosen. For each scalar parameter, for instance the i^{th} Line Spectral Frequency (LSF) coefficient at each frame, a trajectory of length N is created, consisting of the parameter value in each of the N frames in the TD segment. This N point, scalar, trajectory is then approximated by a P^{th} order polynomial, with $P + 1 < N$. The approximation is done using the least squares method, so as to minimize the distance between the actual trajectory and the polynomial. This is illustrated in Fig. 2.1. Since polynomial coefficients are sensitive to quantization, the polynomial is sampled at $P + 1$ points, and the obtained values, that lie in the original features space, are coded and transmitted. For synthesis, the P+1 features are used to fit the (unique) P^{th} order polynomial which is re-sampled at all the original N points. The authors show that the algorithm is better suited for compression of spectral coefficients than pitch and gain, whose

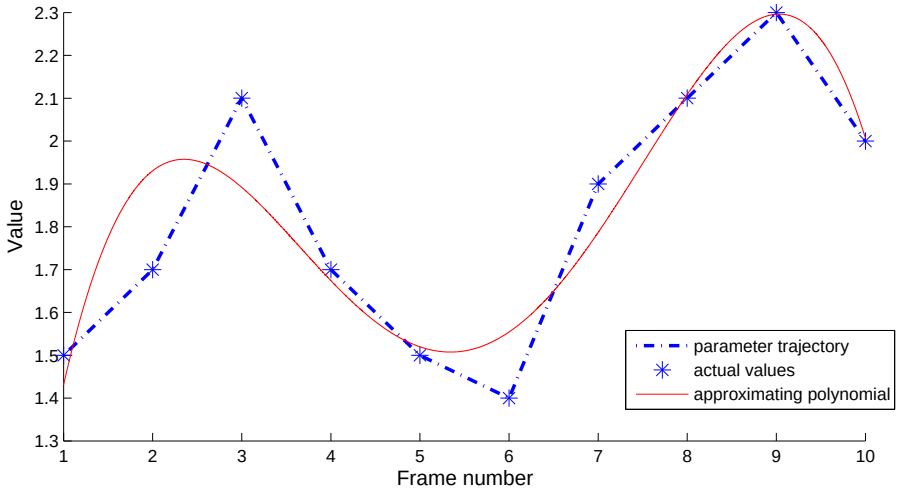

Figure 2.1: Illustration of Polynomial based scalar TD for a single TD segment and a single parameter with N=10 and P=4. The asterisks mark the actual parameter values at each of the 10 frames, the dashed line is their actual trajectory and the solid red line shows the approximating polynomial of order 4.

time trajectories are less stable. Evaluation of the algorithm was performed by embedding it into the MELP 2400 b/s codec, and using various quality measures including spectral distortion (SD), Itakura-Saito distance and subjective testing. Their tests show equivalent quality at 1533 b/s (N=10, P=4), and slightly superior quality at that rate for noisy signals.

A similar approach was presented by Nygaard et. al. in [42], [43], for the compression of ECG signals. This work deals with a scalar setup only, using a signal which is known to be piecewise linear, and proposes to use either linear interpolation (in [43]) or fitting with second order polynomials (in [42]) consistently, and is therefore in a way a subset of the polynomial TD approach, though it is an interesting application of it.

2.2.4 Proposed Vectorial Polynomial TD

When attempting to apply Temporal Decomposition to CTTS speech segments, we must keep in mind that the data consists mainly of very short segments. Therefore, it seems that of the above approaches, the most appropriate model is the scalar polynomial TD, which applies well to short segments, as it contains little model overhead and can be easily adapted to short segments by decreasing polynomial orders.

However, we wish to also benefit from the advantages of vector TD. Vector TD incurs reduced overhead due to joint modeling of all parameters in the vector. Another advantage of vector TD is that the event targets can be quantized and coded with the same tools used to code the original feature vectors, as they lie in the same space.

We also wish to incorporate the adaptive model order selection as proposed by Girin *et* al. in their DCT TD approach, in order to obtain a consistent error, rather than using a constant model order as in the original scalar polynomial TD.

Combining all of the above, we propose the following. For each segment of length N, we perform scalar polynomial TD for each of the amplitude parameters, however we select an optimal polynomial order, jointly for all the trajectories. Then we sample these polynomials at the same points, and thus we are essentially performing a form of vector TD. The polynomial sampling points are our event locations, event targets are the sample values recombined into vector form and event functions are implicitly given by the interpolation polynomials. An advantage of this approach over classic

vector TD is that each vector element may have a different interpolation function, which is well matched to its trajectory - the only common factor between the interpolation functions being that they belong to the class of P^{th} order polynomials.

We will also address the selection of N, the number of frames in the TD segment, which in our proposed algorithm is no longer constant, as we perform adaptive segmentation.

Another point that is not addressed by Dusan *et* al. is an analysis of the resulting error, including the quantization error manifestation, which affects selection of sampling points. We discuss this in detail in Appendix A.

The central challenge in applying the vectorial polynomial TD algorithm to the CTTS acoustic leaves revolves around performing the segmentation and polynomial order selection wisely using a suitable rate-distortion model and a good cost function, which are all described in this chapter.

2.3 Algorithm Outline

To apply our TD-based algorithm to each acoustic leaf, we begin by concatenating the speech segments in the leaf to obtain one long super-segment. The order of the speech segments is either according to their original order in the leaf, or else determined according to the proposed segment ordering algorithm described later, in Ch. 4. Since we do not expect smoothness to hold for the entire super-segment, we perform sub-segmentation into TD segments in an optimal manner, as described in the following section. Then, the vectorial polynomial TD, is applied to the parameters of each TD segment. Our goal is to reach a target rate R_t while minimizing the obtained distortion. We found that the **minmax** approach, i.e., minimizing the maximum distortion, provides better perceptual quality than minimizing mean distortion. Also, we bound the allowed distortion jointly over the entire database, to obtain consistent quality, while achieving the overall target rate or compression ratio, allowing variation of compression ratio among leaves. This approach outperforms enforcing the target compression ratio on each leaf.

Thus, our constrained optimization problem can be described as follows: Assuming a known per-frame distortion value, D_f, (discussed in 2.4.1) calculated between the original and reconstructed speech frames, the global

distortion D_g is defined as:

$$D_g = \max_{leaves} \{\max_{TDsegments} [\max_{frames}(D_f)]\} \tag{2.1}$$

We wish to find the smallest global distortion, D_g^*, for which the target rate R_t is obtained. i.e.:

$$D_g^* = \min(D_g) \;\; s.t. \;\; R(D_g^*) \leq R_t \tag{2.2}$$

Since the rate is a monotonic non-increasing function of the distortion, we can adopt an iterative solution, similar to the one proposed in [49]. We define a Rate-Distortion structure, RD_s, that holds the distortions obtained in the previous iterations, and enables an efficient bi-section search for the optimal distortion value. This structure also holds the target rate and tolerance range. The tolerance is necessary due to the step-wise nature of the rate distortion function, which does not guarantee the possibility of convergence to an exact target value. At each iteration, RD_s holds the current lower and upper distortion values, D_L and D_U, which define the ends of the 'active' interval, within which we are trying to pinpoint our target working point. The TD algorithm steps are described in the following 'pseudo-code':

1. Initialize: D_L=0, D_U=maximum distortion value.
2. Perform Polynomial TD with $D_g = D_U$, set $rate$ to obtained rate.
3. Verify $rate < R_t$. If not, double D_U value and GOTO 2.
4. Calculate next value for D_g: $D_g = \frac{1}{2}(D_L + D_U)$.
5. Perform polynomial TD for each leaf in the database, limiting maximum allowed distortion to D_g, and set $rate$ to the obtained overall rate.
6. IF $rate$ is within the tolerance range of the target rate, R_t,: GOTO 8.
7. IF ($rate < R_t$): $D_U = D_g$, ELSE: $D_L = D_g$; GOTO 5.
8. $D_g^* = D_g$; END.

The need for step 3 stems from the fact that on one hand, we do not wish to initialize our interval with a value of D_U that is too high and will incur unnecessary iterations. For instance we could set initial distortion to its maximum by transmitting no data, but then we would require quite a few iterations to narrow our interval to the relevant values. On the other hand, we must make sure that our initial D_U is high enough to assure that the working point we seek lies within the designated interval - which is exactly what step 3 does.

This algorithm is illustrated in Fig. 2.2. Note that we perform simple bi-section, as proposed in [49]. We also evaluated the option of performing weighted bi-section, but found that in many cases the convergence was actually slower due to the non-linearity of the rate-distortion function.

When calculating the obtained rate we take into account both the bits incurred by coding the parameters, and, for each TD segment, the overhead bits required to hold the selected segment length and polynomial order. We use the current IBM split-VQ quantization, described in Chapter 1, which represents each amplitude parameter vector with 86 bits. Therefore, for each TD segment of length N represented with a P^{th} order polynomial, the full rate is $86*N$, and the obtained rate is $86*(P+1)+overhead$, where the overhead consists of the bits required to represent the TD segment length and selected polynomial order.

Note that the proposed algorithm allows for automatic adaptation to any target rate or compression ratio.

2.4 Jointly Optimal Segmentation and Polynomial Order Selection

Since we wish to use low order polynomials, (in order to limit decoding complexity and quantization error), and also cannot presume smoothness assumptions will hold for the entire acoustic leaf, we must perform segmentation of the super-segment into a number of TD segments. The advantage of this approach, as opposed to just using the original speech segments, is that we may concatenate speech segments that join smoothly, while 'breaking-up' speech segments with discontinuities. We will now describe the proposed algorithm for jointly optimal segmentation and polynomial order selection, which is based on the algorithm presented in [47] and [46]. In these publications Prandoni proposes joint segmentation of speech samples with selection

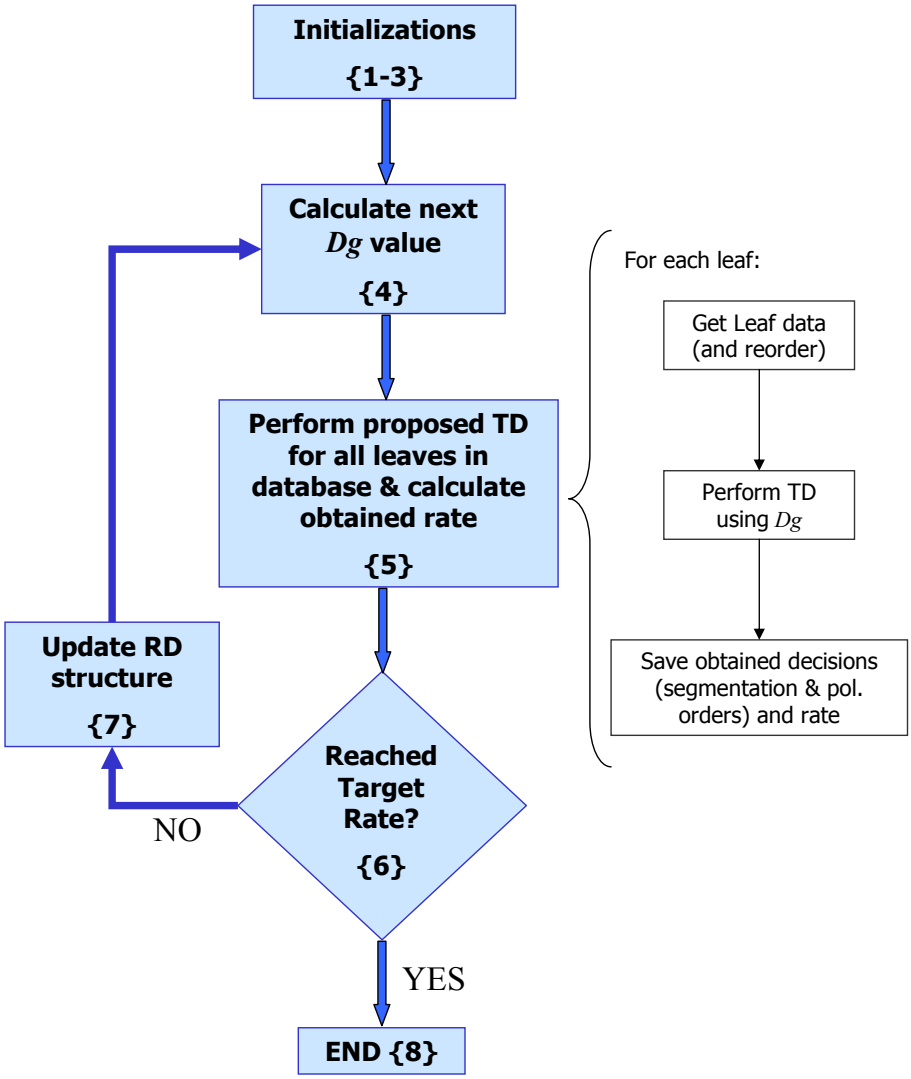

Figure 2.2: Outline of the proposed algorithm

of optimal LPC model order, and joint segmentation and polynomial order selection for piecewise smooth continuous functions, respectively. The main innovation in our algorithm is that we have an additional dimension. Our 'input parameter', for which we seek to perform segmentation and model order selection, is actually a vector, holding the frames' spectral features. Therefore we expand the 2D solution proposed by Prandoni, to a 3D solution, as described next.

We define a generalized 3D trellis structure. The need for a generalized trellis stems from the dependencies between the segmentation decisions, which invalidates the assumption of independency used in a regular trellis. The horizontal dimension of the 3D generalized trellis, corresponds to the candidate TD segment termination points (segment ends), the vertical dimension corresponds to TD segment length, and the depth dimension corresponds to candidate polynomial orders. Each point $S_{i,j,k}$ in this structure is assigned a cost, based on the distortion calculated for a TD segment that ends after frame i, consists of j frames and uses a polynomial of order k. Then we "flatten" the structure as follows: At each trellis point, the value of k for that point is set to the lowest polynomial order for which the resulting distortion in the corresponding TD segment does not exceed the current value of D_g. Thus we have a 2D generalized structure, containing the points $S_{i,j,k^*} \equiv S_{i,j}$, through which we wish to find the optimal path. The initial state $S_{0,0}$, is a virtual state that provides a 'root' for the trellis, i.e., a point where all paths must begin. Once costs have been assigned to each state in the trellis, we step through the trellis and calculate the accumulated costs at each point. For each column, or value of i, we find the j that has the lowest accumulated cost, and mark that state with a star. The accumulated cost of $S^*_{i,j}$ will be added to the total cost of any path that starts at frame $i+1$. When the end of the trellis is reached, we perform backtracking from the state with the lowest cost in the last column, back through the trellis, until we reach the root. During the backtracking we store the selected segmentation points and their pre-selected corresponding polynomial orders. This is illustrated in Fig. 2.3, for a five column trellis.

2.4.1 Cost Function Selection

In the optimization process we must constantly evaluate the distortion between the original speech and the reconstructed speech with a specific TD

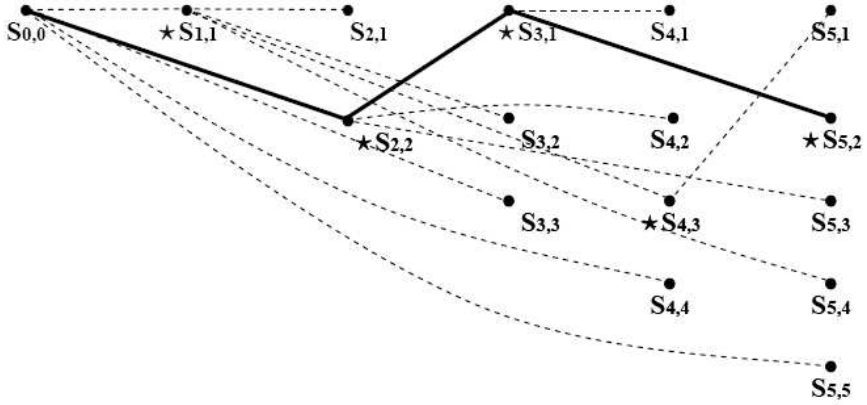

Figure 2.3: 2D structure for optimal segmentation and polynomial order selection. The horizontal dimension corresponds to the candidate TD segment termination points (segment ends), the vertical dimension corresponds to TD segment length. Dotted lines illustrate all the paths considered while proceeding through the generalized trellis, solid line shows optimal path. A star marks the state with the lowest accumulated cost in each column

setup. However, we cannot afford to transform back into the speech domain for each evaluation point, in order to actually measure the obtained distortion. Therefore, we must find a function that when applied to the original and reconstructed parameter sets, predicts the perceptual distortion reliably. In this section we will describe various evaluated distortion functions, and the one we eventually selected. The two types of candidate distortion functions were MSE and an LSD measure.

1. Mean Squared Error (MSE):
   Given an original parameter vector V of length L, and a reconstructed vector $\hat{V}$, the Mean Squared Error, or MSE, between them is simply:

$$MSE = \frac{1}{L}\sum_{l=1}^{l=L}(V_l - \hat{V}_l)^2$$

   .

2. Log Spectral Distortion (LSD):

The LSD measure is considered a reliable estimator of perceived speech quality. It is calculated based on the spectrums of the original and reconstructed signals: $A(\omega)$ and $\widehat{A(\omega)}$ using a logarithmic scale. The LSD, as defined in [29], is given by:

$$LSD = \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left[10\log_{10}\frac{|A(\omega)|^2}{|\widehat{A(\omega)}|^2}\right]^2 d\omega} \tag{2.3}$$

Since our parameters represent the sampled signal spectrum, we may calculate the LSD directly in the parameter space. The required steps and calculations are detailed in Appendix B. The proposed measure can be easily computed during the optimization process so that the distortion we are minimizing is the actual LSD rather than just an Euclidean distance measure between the parameter vectors, in hope of improving overall performance.

For each of the candidate measures, MSE and LSD, we compared the performance obtained when the measure was calculated over the entire frequency range, [0-11050]Hz, to the performance obtained when calculating over an active range of [100-8000]Hz only.

Another parameter in the cost function determination, is wether to take the average distortion over the frames in the TD segment, or the maximum distortion.

Thus we are faced with two types of cost functions, each having one of four possible setups:

1. Full frequency range and mean over frames.
2. Full frequency range and max over frames.
3. Active frequency range and mean over frames.
4. Active frequency range and max over frames.

All candidate cost functions were combined into a simple compression setup (without quantization). The obtained quality and compression ratio were evaluated for each proposed cost, and a performance comparison was performed. Results for four selected cases, LSD and MSE using the maximum and mean over frames and the full frequency band, are shown in Fig. 2.4. Using the LSD measure did not improve perceptual performance.

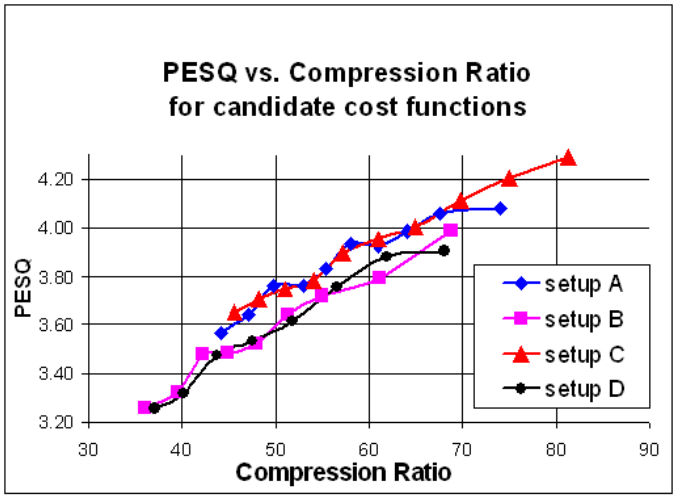

Figure 2.4: PSNR vs. Rate for various cost functions. Results shown for full frequency band; A: LSD maximum; B: LSD mean; C: MSE maximum (selected); D: MSE mean

This appears to be because the spectral parameters were already obtained with perceptual considerations in mind. We found that using the MSE measure, calculated over the entire frequency range, and limiting the maximal distortion over the TD segment, (Setup C in Fig. 2.4), provided the best and most consistent results.

2.5 Evaluated Setups

The proposed algorithm is to be used in small footprint CTTS synthesizers. Since the host devices often have both CPU and memory constraints, we examined some reduced complexity setups.

A number of algorithm setups were examined, and are described in the following subsections. Most of these setups aim to minimize complexity, while others also attempt to improve the obtained speech quality.

2.5.1 Full Setup

The full setup of the algorithm refers to the algorithm as described above, i.e., performing optimal segmentation and order selection. Polynomial orders are restricted to the range [0,4] in order to limit the effect of the quantization error, as described in Appendix A. Maximum TD segment length is 16 frames, as longer frames were very rarely selected, and the longer the allowed segments the higher the algorithm complexity and resulting overhead. The required overhead for each TD segment in this setup is 7 bits: 3 to represent the selected polynomial order and 4 to represent the selected TD segment length.

2.5.2 Reduced Polynomial Order Setup

To avoid the need to perform complex polynomial fitting when decoding (using least-squares), we evaluated the performance allowing polynomials of orders 0 and 1 only, which require at most linear interpolation. The optimization procedure is carried out in the same manner, but it has lower complexity. This approach was also justified based on the fact that in the Full Setup, approximately 70% of the TD segments used polynomial orders of 0 and 1. We will explain this in detail when we present the results below. The overhead in this setup is 5 bits per TD segment: a single bit for polynomial order and 4 for selected TD segment length.

2.5.3 Short TD Segment Setup

We found that in both the full and reduced polynomial orders setups, most of the TD segments are quite short. Thus, we are not using the overhead of the 4 bits required to represent the selected TD length efficiently. We therefore also evaluated a setup which limits the maximum TD segment length to 8 frames, which reduces the overhead per TD segment by 1 bit. An additional advantage to this is the decreased probability that in order to reconstruct a specific speech segment we reconstruct many unneeded frames that belong to a different speech segment but the same TD segment. The results for this setup shown in Table 2.1, are for the case of reduced polynomial order.

2.5.4 Naive Segmentation Setup

In this approach, we do not extend the TD segments beyond original speech segment boundaries. This substantially reduces encoding complexity, and also reduces decoding complexity since there is no need for decoding of additional frames in neighboring speech segments that are part of the same TD segment. Long speech segments are split into TD sub-segments, and the polynomial order for each TD segment is found s.t. the maximum distortion along the segment is bounded by D_{max}, which is found using the iterative algorithm used in the full setup. We evaluated this setup with a maximum TD segment length of 8 and of 4. The longer segments enable more efficient compression, but in order to obtain the target distortion may require polynomials up to order 7, which in turn may increase quantization error and decoding complexity. The overhead size in this setup is adaptive in the range 0-3, and depends on the speech segment length (which is stored as part of the original side information).

2.5.5 Embedded Quantization Setup

In all the setups described above, the quantization is performed as a final step, after optimization is completed. In the Embedded Quantization setup, we perform the quantization in-loop, i.e., prior to evaluating the distortion resulting from each segmentation and polynomial order selection. The optimization is performed using the overall error, consisting of both the model error and the quantization error.

2.5.6 Experimental Results of the Proposed Setups

The proposed algorithm setups were evaluated by applying them to our leaf database, and then using the reconstructed leaves of each setup to create 10 sample sentences. The PESQ scores [22], were calculated between the original sentences created using the original leaves for the TTS, and the reconstructed sentences using the recompressed leaves. A more detailed description of the experimental environment is described in Chapter 5. Using this environment the performance was evaluated for recompression with a factor of 2, for each setup. The obtained PESQ scores are provided in Table 2.1, and their statistics are presented in Table 2.2. All results are shown using the proposed Metropolis Based Ordering algorithm (where applicable) and using the current IBM VQ. For comparison, we also bring the results

obtained by performing simple downsampling each feature trajectory by a factor of 2, with appropriate pre-filtering, and reconstruction via linear interpolation. This simple approach provides much lower PESQ scores.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Full	3.74	3.76	3.60	3.49	3.49	3.91	3.95	3.56	3.63	3.53
Red. Pol Ord	3.72	3.54	3.70	3.55	3.64	3.72	4.04	3.51	3.81	3.65
Short TD seg	3.73	3.64	3.71	3.57	3.65	3.65	4.02	3.48	3.78	3.63
Naive seg a (8)	3.73	3.63	3.50	3.83	3.79	3.67	3.72	3.62	3.73	3.82
Naive seg b (4)	3.59	3.71	3.50	3.66	3.70	3.60	3.77	3.59	3.59	3.66
Embedded Q	3.72	3.49	3.23	3.56	3.64	3.87	3.80	3.59	3.63	3.53
Downsamplex2	2.81	2.93	2.89	2.88	2.82	2.90	2.48	2.86	2.99	2.86

Table 2.1: PESQ results for the proposed Polynomial TD setups for 10 test sentences

Note, that the embedded quantization caused the PESQ score to decrease, although we initially expected this setup to provide the best perceptual quality. This is due to the fact that the quantization error is perceptually shaped, whereas the proposed *minmax* optimization is not. We will first explain why this causes the drop in performance and will then describe a possible solution and its limitations. The quantizer introduces quite large errors in frequency bands that are perceptually less important, but this causes a rise in the value of our target distortion, which may allow for larger distortions in the frequencies that have a higher perceptual importance. This is demonstrated in the following example: Assuming

	Avg. PESQ	Min. PESQ	PESQ STD.
Full	3.67	3.49	0.167
Red. Pol Ord	3.688	3.51	0.155
Short TD seg	3.685	3.48	0.142
Naive seg a (8)	3.70	3.50	0.102
Naive seg b (4)	3.63	3.50	0.078
Embedded Q	3.61	3.23	0.178
Downsamplex2	2.84	2.48	0.137

Table 2.2: PESQ results for the proposed Polynomial TD setups - summary

two frequencies ω_1, ω_2, and a quantizer that introduces corresponding errors $e(\omega_1) = E$, $e(\omega_2) = 2E$, that results in imperceptible distortion, due to lower perceptual sensitivity at frequency ω_2. If we use MSE to combine these and calculate the target distortion, i.e., we allow for an error up to $1.5 * E$, thus a vector that has the errors $e(\omega_1) = 2 * E$, $e(\omega_2) = E$, is equally 'valid', yet will cause a greater perceptual distortion. This causes the Embedded setup to minimize the obtained MSE but it does not maximize the obtained perceptual quality. To solve this we need to perform optimization with a perceptually weighted cost function, tuned to the quantizer error. If the quantizer had an analytical error shaping function, we could use this rather than the MSE to obtain a shaped error to use in our distortion calculation. But since the Q function is purely heuristic this is not possible. Therefore, we attempted to apply some appropriate perceptually shaped weights to the distortion function, which becomes a Weighted MSE (WMSE). Inspired by the perceptual distortion measures proposed in [59] and [11] we propose the following weights, which are derived from the same Mel-scale used in creating the 32 feature vectors:

$$w_i = \frac{1}{\sum_{i=1}^{32} \tilde{w_i}}; \tilde{w_i} = \frac{1}{BW_i}, i = 1, ..., 32. \tag{2.4}$$

BW_i is the bandwidth of the i^{th} feature, i.e. the segment of the spectrum which it represents. By providing weights that are proportionally inverse to these bandwidths we give decreasing importance to higher frequency using the well accepted Mel-scale as the weighting factor. When performing the TD using this WMSE distortion measure, we obtained average PESQ scores for the 10 test sentences, of 3.66 without embedded quantization and 3.62 with embedded quantization. The worst case PESQ score was 3.29 without embedded quantization and 3.47 with embedded quantization. This shows that by using a perceptually weighted distortion measure the performance of the embedded quantization approaches that of the non embedded setup, but since we there is no analytical formulation for the weighting achieved by the IBM quantizer we cannot find a WMSE solution that perceptually outperforms the results when using MSE without embedded quantization.

In Table 2.3, the advantages and disadvantages of each proposed approach are provided. Combining these considerations with the results shown above we can choose the best performing algorithm for various target applications. While the Naive segmentation algorithm, with maximum segment

length of 8 provides the highest PESQ score, it requires use of high order polynomials, the highest order used being 6, which adds unacceptable complexity to decoding which must be performed in real time. The reduced polynomial order setup and short TD segment setup (also using reduced polynomial orders - as described in 2.5.3) both provide good quality, with the reduced polynomial order offering slightly better worst case PESQ score, but very slightly lower average PESQ score. The short TD segment setup has the added advantage of using shorter TD segments, which decreases the probability of having to reconstruct many unneeded frames from neighboring speech segments that lie in a joint TD segment, when reconstruction of a single speech segment is required. Assuming the criteria for choosing the setup to use are the obtained perceptual speech quality (as measured by PESQ), and the decoding complexity, which is the case in the IBM TTS application, the **short TD segments setup**, which also uses reduced polynomial orders, is recommended for recompression of the acoustic leaves. Some more results and further discussion can be found in Chapter 5.

	PROs	CONs
Full	Optimal under defined constraints	High complexity encoding
Red. Pol Ord	Low complexity decoding	Medium-High complexity encoding
Short TD seg	Low complexity decoding	Medium-High complexity encoding
Naive seg a (8)	Highest quality	Uses some high order polynmials (up to order 6)
Naive seg b (4)	Lowest encoding complexity	lower obtained quality than alternative setups
Embedded Q	Reduces feature distortion; more methodically complete	Very high complexity encoding; reduced perceptual quality

Table 2.3: Pros and Cons of the various Polynomial TD setups

2.6 Summary

In this chapter we presented the Vectorial Polynomial TD approach to acoustic leaf compression. We applied our algorithm to the super-segment created by concatenating the speech segments in each acoustic leaf. For each leaf, given a target distortion value, we performed optimal segmentation and polynomial order selection. Thus the trajectory of each parameter in the vector, is modeled by a polynomial of an order that assures the obtained distortion meets the target distortion. Although each parameter is modeled by a separate trajectory, we enforce the same polynomial model for all parameters in each TD segment, thus reverting to a vectorial TD scheme. As the polynomials are represented by their samples, we perform the sampling jointly along the vectors and thus are able to use the systems original vector coding tools. We also presented an iterative algorithm that enables automatic convergence to a target bit-rate or compression ratio, by iterating over distortion values.

A number of candidate distortion functions were presented. We found that using the MSE for each frame, calculated over the entire frequency range, and limiting the maximum distortion obtained among all frames in the TD segment, provided the best and most stable results, and was therefore selected.

We described a number of possible algorithmic and implementation setups, under various constraints. Results for the proposed setups were also provided as well as the advantages and disadvantages of each setup. Some more detailed results and further discussion will be provided in Chapter 5, however from the above results it is already clear that the proposed algorithm enables perceptually equivalent quality speech with a recompression factor of 2.

Chapter 3

Compression using 3D Shape Adaptive DCT

3.1 Motivation

In order to reduce the CTTS footprint, we wish to compress the amplitude parameters stored in the acoustic leaves. In the setup we used, each acoustic leaf consists of 5-10 speech segments, with each segment containing 1-35 frames (with a median value of 2 frames). Each speech frame is represented by 32 amplitude parameters. Thus, we have a 3D data structure, as illustrated in Fig. 3.1. In the previously proposed approach, based on Polynomial Temporal Decomposition, the speech segments were concatenated into one long super-segment, reducing the dimensionality to 2D.

In this chapter we explore an alternative approach, where the compression is performed directly on the 3D acoustic leaf. Thus, we hope to remove not only the temporal redundancy between adjacent speech frames, but also the redundancy between speech segments that belong to the same leaf, and are expected to be similar. The Discrete Cosine Transform (DCT), an energy preserving reversible transform, is widely adopted for redundancy removal due to it's energy compaction property and the fact that it is separable, real valued and easy to compute. Due to the variability in segment lengths, in order to apply DCT to our 3D structure we must use a variation, known as Shape Adaptive DCT (SADCT) which we will describe shortly. After applying the transform we must perform quantization of the obtained values in order to achieve compression. Since the obtained parameters do

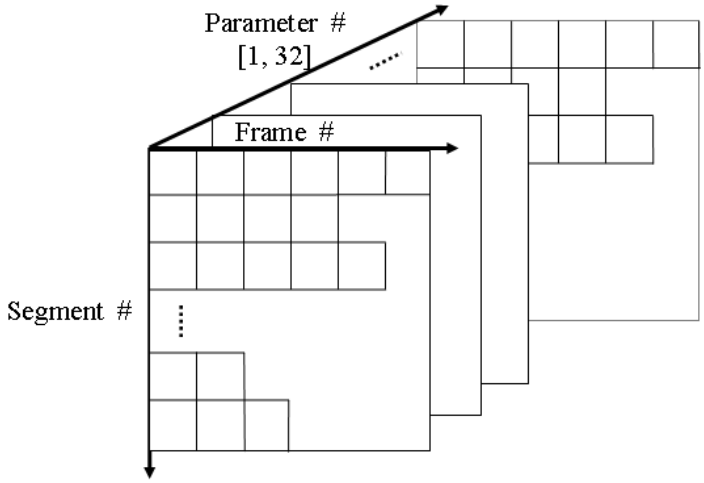

Figure 3.1: 3D structure of acoustic leaf

not lie in the same space as the original parameters, we cannot reuse the existing quantizer. We will describe the quantizer design in the last part of this chapter.

3.2 3D Shape Adaptive Discrete Cosine Transform

The 3D Discrete Cosine Transform, or DCT, is defined by the following equations:

$$F(u,v,w) = \sqrt{\frac{8}{NMP}} C_u C_v C_w \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \sum_{p=0}^{P-1} f(n,m,p) \cos\left[\frac{(2n+1)u\pi}{2N}\right] ... \\ \cos\left[\frac{(2m+1)v\pi}{2M}\right] \cos\left[\frac{(2p+1)w\pi}{2P}\right] \quad (3.1)$$

Where,

$$C_x = \begin{cases} \frac{1}{\sqrt{2}}, & \text{x=0;} \\ 1, & \text{otherwise.} \end{cases} \quad (3.2)$$

$F(u,v,w)$ are the transform coefficients whose squared values represent the energy present in the acoustic leaf at the corresponding 3D frequency. The

inverse transform is defined as:

$$f(n,m,p) = \sqrt{\frac{8}{NMP}} C_u C_v C_w \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \sum_{w=0}^{P-1} F(u,v,w) \cos\left[\frac{(2n+1)u\pi}{2N}\right] ... \\ \cos\left[\frac{(2m+1)v\pi}{2M}\right] \cos\left[\frac{(2p+1)w\pi}{2P}\right] \quad (3.3)$$

Note that the these transforms are separable. The same result can be obtained by first applying a 1D DCT along the first dimension, to obtain $F_1(u,m,p)$, then applying 1D transform to these values along the second dimension to obtain $F_2(u,v,p)$, and finally applying a 1D DCT along the third dimension to obtain $F(u,v,w)$.

The 2D SADCT, first proposed in [54], was originally developed for efficient coding of arbitrary shaped image segments in video. In [54] Sikora and Makai also provide a statistical analysis showing that the energy compaction property of the DCT is retained in SADCT with a 'reasonable' contour. In [35] Markman and Malah extended this concept to 3D SADCT and applied it to hyperspectral image coding.

SADCT exploits the separability of the DCT transform, and applies variable length basis functions to each row and column. The transform applies to general contours, however our case is simpler as the variability, in each leaf, is only in one dimension, which corresponds to the number of frames per segment. Also, as opposed to the general case which requires additional storage for the contour, our 'contour' is well defined by the number of speech segments in the leaf (vertical dimension), and their lengths (horizontal dimension), which are already stored in the leaf header.

The 2D SADCT, for an example block, is demonstrated in Fig. 3.2. First, the data is aligned along the y axis and the appropriate length 1D DCT is applied to each column. Then the results are aligned along the x axis and the appropriate 1D DCT is applied to each row. The original contour data is also required for reconstruction.

We apply the 3D SADCT to each acoustic leaf as illustrated in Fig. 3.3, by performing required shifts (to eliminate "holes") and subsequent 1D DCT transforms. The length of the basis functions used is adapted to the length of the data in each column (segment number) or row (frame number) and fixed to 32 for the depth dimension. This provides our 3D variable dimension array of SADCT coefficients.

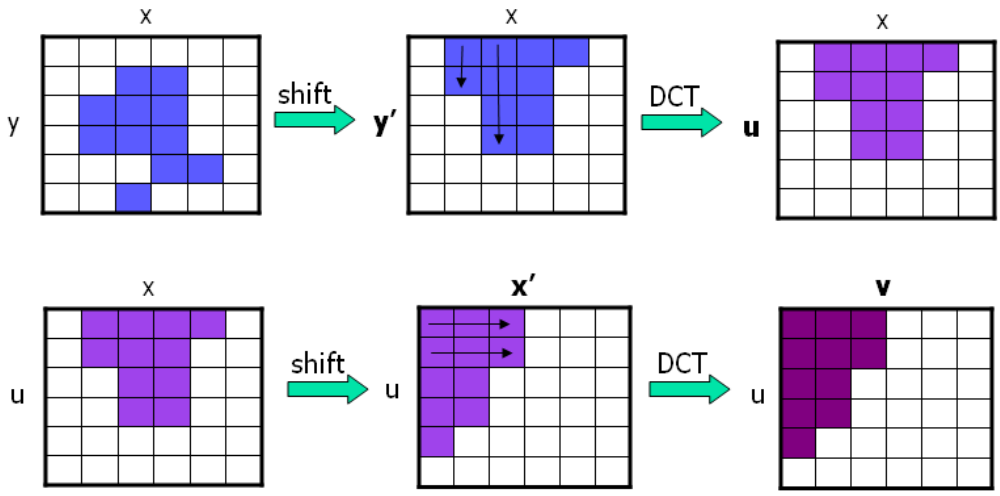

Figure 3.2: 2D SADCT: Top illustrates DCT along columns, bottom illustrates DCT along rows. Basis function lengths are adapted to data length of each column or row.

3.3 Compacting Acoustic Leaf Energy using 3D-SADCT

As explained above, the 3D SADCT is readily applied to the 3D acoustic leaf structure. In order to evaluate the contribution of this transform in removing redundances within the leaf, we evaluated the energy compaction of the obtained coefficients. If we manage to concentrate the energy into few, low frequency, coefficients, this indicates that we have done a good job at removing redundancies and will be able to obtain compression.

The obtained energy compaction is illustrated in Fig. 3.4, for 670 sample leafs, by displaying the ratio between the number of coefficients we must retain in order to represent 90, 95 and 98 percent of the total energy, and the original number of features in the leaf. This is shown for the original features along with DCT and SADCT coefficients. The DCT is performed on the bounding cube, with zero padding. Note that the higher the compaction the lower this percent, or ratio, should be. From Fig. 3.4, it is clear the SADCT substantially and consistently outperforms the other two representations

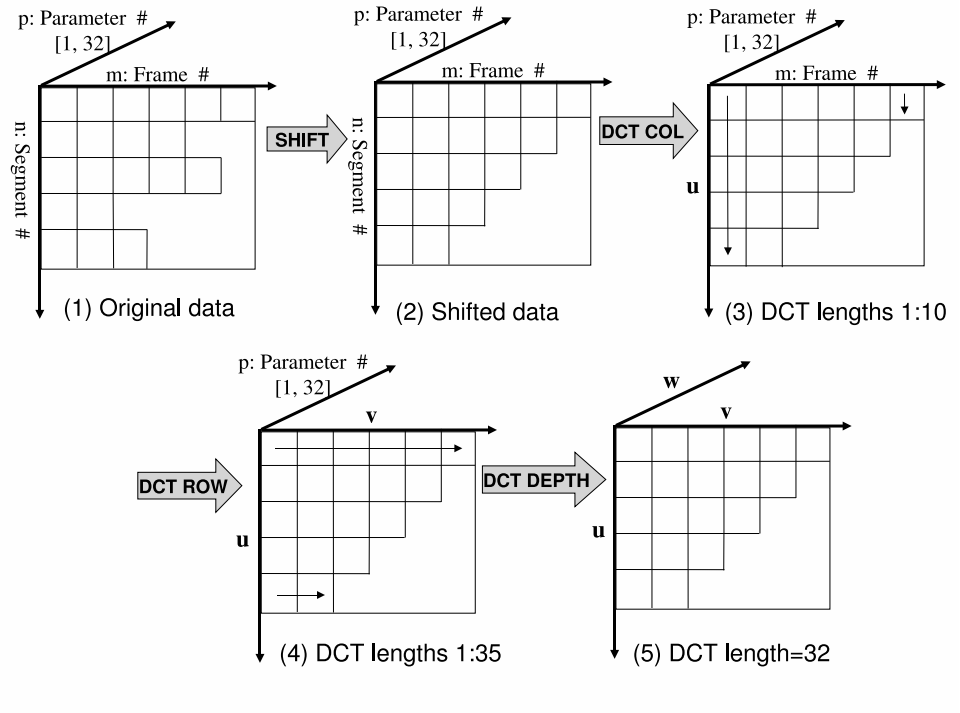

Figure 3.3: 3D SADCT applied to a sample acoustic leaf. Shifting is required only along the segment number dimension. The DCT Basis function lengths are adapted to the length of each column (segment number within leaf) or row (frame number within segment), and are fixed at 32 for the depth (feature vector) dimension.

in terms of energy compaction. Also, not surprisingly, we have an uphill battle - the higher percent we wish to retain the more difficult it becomes. It is interesting to note the points where the DCT graph rises above 1. This is due to the fact that the artificial edge between the feature values and the zeros in the bounding cube adds some high frequency elements to the DCT coefficients. Since the DCT coefficients are not zero outside the feature support area, we may actually require more DCT coefficients than the original number of features to retain 90% or more of the total energy.

One last interesting point: It is apparent that when the features require keeping of relatively more values (blue line goes up), the SADCT can manage with relatively less values (red line goes down). This is in accordance with the theory (flat distribution means the energy is very spread in the feature domain but highly concentrated in the DCT domain), and easy to demonstrate on the extreme case: if all the feature values are identical we require 95% of the values to keep 95% of the energy, while a single DC coefficient holds all their energy in the DCT domain.

To summarize, it is apparent that 3D SADCT provides good energy compaction when applied to our acoustic leaves, which justifies further pursuit of this approach.

3.4 Quantizer Design for 3D SADCT Coefficients

We have shown that 3D SADCT applies well to 3D acoustic leaves, and assists in redundancy removal. However, to obtain compression, we must develop an appropriate quantization scheme.

Our goal is to reach a recompression factor of 2 compared to the IBM system we used, which applies an 86 bit vector quantizer to each 32 element vector, i.e., about 2.6 bits per coefficient. This means our target is to obtain a rate of about 1.3 bits per coefficient, which is unattainable with scalar quantization, even when using run-length encoding techniques. Thus we pursue a vector quantization approach.

Previous works in the area propose a variety of approaches. In [53], a framework for bit allocation in a multi quantizer setup is proposed. Assuming a pre-known split of the data between quantizers, for instance in sub-band coding, they propose a method of allocating the bits between the quantizers finding the allocation that yields minimum distortion. We cannot use this approach since we do not know how to split the data in advance,

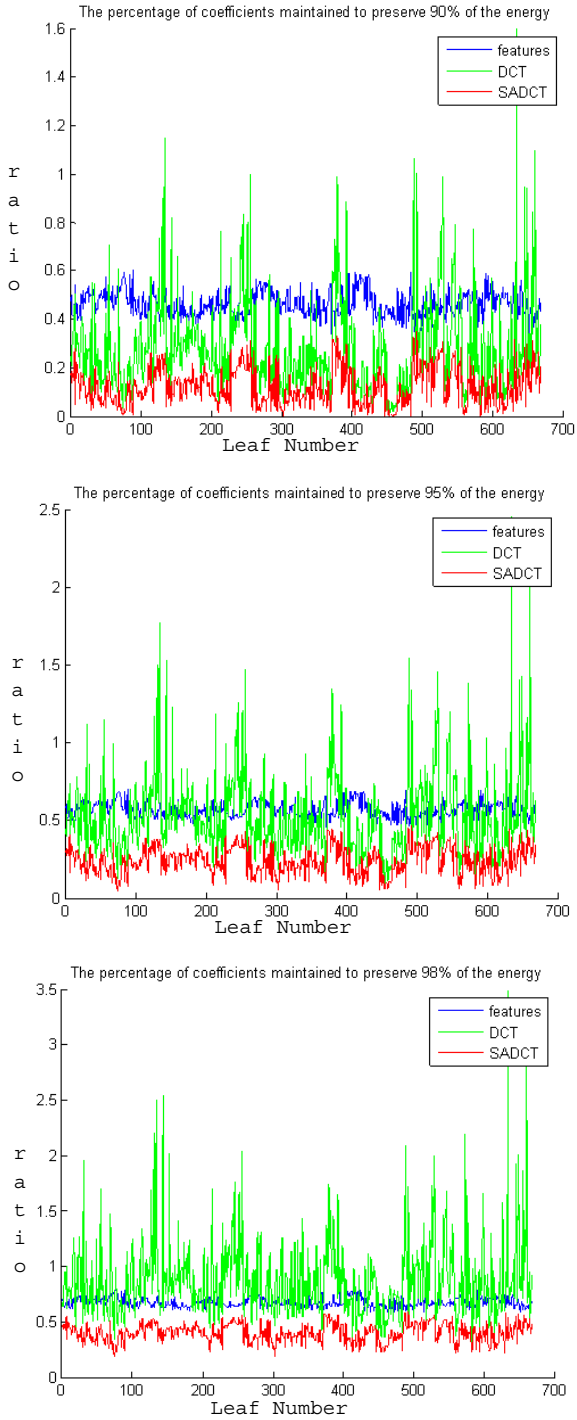

Figure 3.4: Energy compaction of original features, DCT and SADCT coefficients, shown via the ratio between the coefficients that hold 90%(top) 95% (middle) and 98%(bottom) of the total energy and the total number of elements in the leaf. The sporadic green line corresponds to DCT, the bottom red line to SADCT and the blue line to the original features.

and also do not have a closed form distortion function, required for this algorithm, since we wish to minimize perceptual error in the reconstructed speech and not just the DCT coefficient MSE or a similar distortion. In [7] an optimum transform domain split vector quantization method is developed, and both an optimal and sub-optimal fast approach to finding the optimal vector split points are presented. However, assumptions are made regarding the source statistics that do not apply to our case, and the algorithm is also not easily adapted to the varying dimensionality of our data. In [35], where quantization of 3D DCT coefficient arrays is required, two approaches are presented. In the first a 3D quantization matrix is applied and then the coefficients are coded in a run-length scheme. Alternatively, the 3D coefficients are scanned into a single vector of increasing frequency coefficients and quantized with a split VQ approach using heuristically determined splitting points. A Matrix quantization approach, such as the one used in [60], is difficult to apply here due to the varying dimensions.

We opted to pursue a methodical split VQ approach, which requires solving the following problems:

1. Finding a method for splitting the 3D structures into "manageable" sub-units.
2. Performing Allocation of bits to the various units.
3. Designing a vector quantizer for each unit.

We found that the best results were obtained by performing the splitting and bit allocation jointly, as we describe in the following sub-section. We will then address the issue of the Vector Quantizer design.

All the quantizer training procedures were performed on the full leaf database which was provided by IBM, containing 23,263 acoustic leaves. This was done to avoid over-fitting to the 1,661 leaves used to create the 10 evaluation sentences.

3.4.1 Bit Allocation and Splitting Algorithm

The data we wish to code is a set of 3D coefficient arrays, one for each acoustic leaf, with indices u, v, w, where $u = 1, ..., U$ and U is the number of segments in the current leaf; $v = 1, ..., V$ and V is a given segment length, and $w = 1, ..., W$ where W is the number of parameters per frame, which in our setup is constant at 32, as demonstrated for a sample leaf in Fig. 3.3.

We perform the splitting and bit allocation in two stages. First, we ignore the w index, and split the $\{u, v\}$ indices into M groups, and determine the bit allocation for each group. Then for each of the M groups, we find an optimal splitting and bit allocation along the $\{w\}$ index. In both stages, the DC coefficient is treated separately from the remaining AC coefficients, as its behavior under quantization is different. We now present a methodical splitting and bit allocation algorithm, which will be applied twice - with slight variations, once for each stage.

In both stages, the bit allocation will be performed using a variant of following, well known, formula ([18] Chapt. 8):

$$R_i^{opt} = R_{avg} + \frac{1}{2}\log_2 \frac{\sigma_i^2}{\left(\prod_{j=1}^{I} \sigma_j^2\right)^{\frac{1}{I}}} + \frac{1}{2}\log_2 \frac{W_i^2}{\left(\prod_{j=2}^{I} W_j^2\right)^{\frac{1}{I}}} \qquad (3.4)$$

Where: R_{avg} is the average bit allocation per element, σ_i is the (pre-known) standard deviation of the i^{th} data element and W_i are weights used for quantization error shaping.

In the first stage we have a 2D array with indices u, v, which we wish to divide into M groups, so that each group comprises a set of $\{u, v\}$ pairs. We found that $M = 5$, i.e., one DC and 4 AC groups, provided a good working point. The first group consists of the DC element only, i.e. $u = 1; v = 1$, and receives an empirical allocation of 50 bits, for each vector of 32 elements. In order to use the above bit allocation formula, some data statistics must be gathered. To obtain robust statistics, despite the varying dimensionality of each leaf, i.e., varying segment lengths and varying number of segments per leaf, we compose two 2D arrays: $STD_{u,v}$ holding the standard deviation of the coefficient vectors with indices $\{u, v\}$, and $N_{u,v}$ which holds the total number of vectors that exist in the transformed database with indices $\{u, v\}$. For instance, for $\{u = 1, v = 1\}$, every acoustic leaf has a representative vector so $N_{1,1} = 23263$, the total number of leaves in the database used; whereas, for instance, $N_{10,4} = 326$, since only 326 leaves have non zero vectors with $\{u = 10, v = 4\}$. The full N matrix for our database is provided in Fig. 3.5.

Next, we need to calculate an average bit rate that will result in our target bit-rate of 43 bits per vector. Due to the varying dimensionality between leafs, i.e., the varying number of segments and speech segment lengths, this becomes non-trivial. To understand this, imagine the following

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	23263	22007	16983	10703	6653	4312	2966	2197	1618	1192	848	598	418	286
2	23263	20684	13691	7383	4103	2544	1779	1244	850	574	379	244	168	118
3	23263	19389	11215	5484	2981	1833	1262	846	551	356	221	161	105	72
4	23263	18095	9150	4159	2267	1425	962	620	375	240	153	103	67	48
5	23263	15777	6846	3045	1681	1063	688	425	249	158	104	67	43	30
6	8759	6080	3081	1593	925	596	370	231	131	87	55	36	24	14
7	6028	4104	2119	1156	704	427	252	148	95	61	34	20	12	5
8	4590	2978	1430	814	506	299	171	104	63	40	23	13	5	2
9	3645	2157	964	545	332	176	104	63	33	20	15	8	5	2
10	2822	1406	573	326	176	91	44	22	11	6	5	1	1	0

...	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
1	198	145	114	88	60	44	35	27	22	18	14	10	8	6	6	6	4	4	2	2	1
2	76	52	34	27	22	16	13	9	9	4	3	2	2	2	1	0	0	0	0	0	0
3	47	33	24	17	15	11	9	6	4	4	3	2	0	0	0	0	0	0	0	0	0
4	33	23	16	10	8	6	3	3	2	1	0	0	0	0	0	0	0	0	0	0	0
5	22	13	6	6	4	3	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	7	3	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.5: The number of elements for each {i,j} in the full database

example. Say that with our allocation of 43 bits per vector we use 53 bits for the 10 lowest frequency coefficient vectors and 33 bits for the remaining coefficient vectors. This will not result in an average of 43 bits per vector since there are more low frequency coefficients in the database than high frequency ones. Therefore, we use an iterative process, which given a target bit-rate (43), proposes an average bit allocation value: $(R_{avg})_t$, where t is the current iteration number, and finds the corresponding bit allocation and splitting scheme as described next. Then, the actual average obtained bit-rate (per vector) is calculated, using $N_{u,v}$, and the average bit allocation is modified appropriately. The iterations continue until the obtained average bit-rate is within the allowed tolerance of the target bit-rate, i.e., 43 bits per vector.

Each iteration of the algorithm, used for the first stage splitting and bit allocation, is as follows:

Using $(R_{avg})_t$ perform bit allocation for each $\{u, v\}$ using 3.4:

$$(R_{u,v}^{opt})_t = (R_{avg})_t + \frac{1}{2}\log_2 \frac{\hat{\sigma}_{u,v}^2}{\left(\prod_{p,q} \hat{\sigma}_{p,q}^2\right)^{\frac{1}{Q}}} + \frac{1}{2}\log_2 \frac{W_{u,v}^2}{\left(\prod_{p,q} W_{p,q}^2\right)^{\frac{1}{Q}}} \tag{3.5}$$

Where: $W_{u,v} = \frac{1}{u*v}$ are weights that prioritize low frequency coefficients, Q is the number of non DC combinations of $\{p, q\}$ for which $N_{p,q} > 0$, and $\hat{\sigma}_{u,v} = STD_{u,v} \cdot N_{u,v}$.

The DC component, $F_{1,1}$, is classified as the first group, and as previously mentioned, receives 50 bits for its 32 elements. Then the remaining bit-allocation values, one for each $\{u, v\}$ pair, are clustered into $M - 1$ groups, using a simple clustering algorithm with Euclidean distance. The bit allocation for each group is then the centroid of the corresponding cluster. The obtained bit-rate is calculated, and if needed R_{avg} is corrected and the next iteration begins. This continues until the target bit-rate is reached.

If the segment reordering proposed in Chapter 4 (MBOrd) is used, this process is performed once on the leaves in their original order. Then the groups found are used to calculate the cost of each proposed order as given by Equation 4.3. Then the splitting and bit allocation is performed again using the reordered leaves.

This algorithm results in M groups of vectors, and a target bit allocation for each group, $R(m)$. The bit allocation per $\{u, v\}$ pair, obtained on our database, with and without the proposed reordering are shown in Fig. 3.6.

Now, for each group, we find a splitting scheme and bit allocation along the w dimension, where $w = 1, ..., 32$, using the same algorithm with slight variations. Since the data is now 1D and with constant length, the algorithm is more straightforward. Again, we begin by allocating bits to the first, DC element. The DC element F(1,1,1), i.e, DC in all dimensions, receives a bit allocation of $R_{DC}(1) = 8$ bits. The bit allocation for the DC element of group m for $m = 2, ..., M$ is: $R_{DC}(m) = 8 \cdot \frac{R(m)}{R(1)}$. Once the DC elements have an allocation we proceed to perform the splitting and bit allocation for the remaining 31 AC coefficients in each of the M groups. For each group, the standard deviation of each vector element is calculated, using only the elements that belong to that group. Then, for each group we perform an

allocation of bits among the 31 non DC elements in the vector using:

$$R_w^{opt} = R_{avg} + \frac{1}{2}\log_2 \frac{\sigma_w^2}{\left(\prod_{l=2}^{32}\sigma_l^2\right)^{\frac{1}{31}}} + \frac{1}{2}\log_2 \frac{W_w^2}{\left(\prod_{l=2}^{32}W_l^2\right)^{\frac{1}{31}}} \tag{3.6}$$

with $R_{avg} = \frac{R(m)-R_{DC}(m)}{31}$, $W_w = \frac{1}{w}$, $w = 2, .., 32$.

The 31 AC bit allocation values are clustered using an iterative clustering algorithm, under the constraint that no sub-vector shall have more than 8 elements. This constraint is required to limit the size of the resulting code books so that their total footprint is no larger than the footprint of the code books in the IBM system we used. The iterative clustering starts with two clusters, then in each iteration one cluster is added and the values are re-clustered. This continues until the largest cluster has no more than 8 elements. Then the bit allocation for each sub-vector is the rounded sum of the allocations of its elements.

This entire process results in 5 sets of split locations and bit allocations, one for each group. Fig. 3.7 describes for each of the 5 groups, the elements belonging to each of the obtained sub-vectors, the sub-vector lengths and bit allocations. (Sub vectors with more than 8 elements are allowed only if they are represented by 0 bits, i.e., not coded). Results are provided both with and without applying the proposed segment ordering - and are very similar.

As we have shown, the proposed methodical splitting and bit allocation algorithm can be used for both fixed length data (vectors of length 32) and 2D data with variable lengths. It may thus be applied to a variety of qunatizer design problems, including but not limited to split VQ quantizer design.

The next stage is to design the VQ codebooks for each sub-vector, as described in the following sub-section.

3.4.2 VQ Design

Once the data is divided into sub units and each unit, i.e., a sub-set of coefficients, has a known bit-allocation, finding the optimal quantizer becomes a well defined problem with known solutions. We chose to adopt the classic LBG approach, presented in [31]. Using an iterative approach we find the best clustering given the cluster centroids, and find the updated centroids

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14....end*
1	50	46	46	42	42	42	39	39	39	39	39	32	32	
2	46	46	42	42	39	39	39	39	32	32	32	32	32	
3	46	46	42	39	39	39	32	32	32	32	32	32	32	If N(I,j)>0:
4	46	42	42	39	39	32	32	32	32	32	32	32	32	32
5	46	42	39	39	32	32	32	32	32	32	32	32	32	Otherwise:
6	42	42	39	32	32	32	32	32	32	32	32	32	32	0
7	42	39	39	32	32	32	32	32	32	32	32	32	32	
8	42	39	32	32	32	32	32	32	32	32	32	32	32	
9	42	39	32	32	32	32	32	32	32	32	32	32	32	
10	42	39	32	32	32	32	32	32	32	32	32	0	0	

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14... end*
1	50	46	46	42	42	42	39	39	39	39	39	33	33	
2	46	46	42	42	39	39	39	33	33	33	33	33	33	
3	46	46	42	39	39	39	33	33	33	33	33	33	33	If N(I,j)>0:
4	46	42	42	39	39	33	33	33	33	33	33	33	33	33
5	46	42	39	39	33	33	33	33	33	33	33	33	33	Otherwise:
6	42	42	39	33	33	33	33	33	33	33	33	33	33	0
7	42	39	39	33	33	33	33	33	33	33	33	33	33	
8	42	39	33	33	33	33	33	33	33	33	33	33	33	
9	42	39	33	33	33	33	33	33	33	33	33	33	33	
10	39	39	33	33	33	33	33	33	33	33	33	0	0	

Figure 3.6: Splitting of 3D SADCT coefficients into $M = 5$ groups and corresponding bit allocations for the case without reordering (Top) and with reordering algorithm described in Chapter 4 - MBOrd (Bottom).

Group #1 Tot: 50	Elements	1	2	3	4	5:6	7:8	9:10	11:14	15:21	22:32
	length	1	1	1	1	2	2	2	4	7	11
	Alloc.	8	5	4	4	6	5	4	6	6	2
Group #2 Tot: 46	Elements	1	2	3	4	5:6	7:8	9:13	14:21	22:32	-
	length	1	1	1	1	2	2	5	8	11	-
	Alloc.	7	5	4	4	6	5	8	6	1	-
Group #3 Tot: 42	Elements	1	2	3:4	5:6	7:8	9:13	14:21	22:32	-	-
	length	1	1	2	2	2	5	8	11	-	-
	Alloc.	7	5	8	6	4	7	5	0	-	-
Group #4 Tot: 39	Elements	1	2	3:4	5:6	7:8	9:13	14:21	22:32	-	-
	length	1	1	2	2	2	5	8	11	-	-
	Alloc.	6	5	7	5	4	7	5	0	-	-
Group #5 Tot: 32	Elements	1	2	3:4	5:6	7:10	11:18	19:32	-	-	-
	length	1	1	2	2	4	8	14	-	-	-
	Alloc.	5	5	7	4	6	5	0	-	-	-

Group #1 Tot: 50	Elements	1	2	3	4	5:6	7:8	9:10	11:14	15:21	22:32
	length	1	1	1	1	2	2	2	4	7	11
	Alloc.	8	5	4	4	6	5	4	6	6	2
Group #2 Tot: 46	Elements	1	2	3	4	5:6	7:8	9:13	14:21	22:32	-
	length	1	1	1	1	2	2	5	8	11	-
	Alloc.	7	5	4	4	6	5	8	6	1	-
Group #3 Tot: 42	Elements	1	2	3:4	5:6	7:8	9:13	14:21	22:32	-	-
	length	1	1	2	2	2	5	8	11	-	-
	Alloc.	7	5	8	6	4	7	5	0	-	-
Group #4 Tot: 39	Elements	1	2	3:4	5:6	7:8	9:13	14:21	22:32	-	-
	length	1	1	2	2	2	5	8	11	-	-
	Alloc.	6	5	7	5	4	7	5	0	-	-
Group #5 Tot: 33	Elements	1	2	3:4	5:6	7:10	11:18	19:32	-	-	-
	length	1	1	2	2	4	8	14	-	-	-
	Alloc.	5	5	7	5	6	5	0	-	-	-

Figure 3.7: Vector split and bit allocation for each SADCT coefficient group for the case without reordering (Top) and with MBOrd (Bottom).

for the clusters. In each iteration the number of clusters is doubled until we reach the target number of clusters according to the bit allocation. For each sub-vector of each group, we use as a training set all the associated sub vectors, and create the corresponding code-book. As mentioned above, we limit the size of the largest cluster, or longest sub-vector, to no more than 8 elements, in order to verify that the collective footprint of the proposed codebooks does not exceed that of the IBM codebooks. Thus, we obtained a set of codebooks, as described by the tables in 3.6, with a total footprint of 6.44 Kbits for the case without reordering, and 6.47 Kbits when using MBord, compared to 17.1 Kbits in the IBM system we used.

3.5 Summary

In this chapter we presented a SADCT based approach to compression of acoustic leaf parameters. The underlying theory of SADCT was presented, and demonstrated for the both the 2D and 3D cases (Fig. 3.2 and 3.3). We showed that good energy compaction of the acoustic leaf data can be obtained using 3D SADCT. We addressed the issue of coefficient quantization and presented a methodical bit allocation and splitting approach. The bit allocation scheme uses two variants of the bit allocation in Equation 3.4. First we perform the bit allocation for each $\{u, v\}$ pair, which represent the frame and segment numbers within the acoustic leaf, using Equation 3.5. Then these allocation values are clustered, providing a split of $\{u, v\}$ pairs into M groups. Then for each group, bit allocation using Equation 3.5 is performed along the 32 element feature vectors of each of the M groups, and these values are clustered to obtain a split-VQ scheme for the vectors in each group. A VQ is then designed for each sub-vector in each group, using the well-known LBG algorithm [31]. This entire procedure is performed using the complete database of the IBM CTTS system provided, to avoid over-fitting, and then evaluated on the 10 test sentences. As we will show in Chapter 5, this algorithm provides perceptually equivalent speech with a x2 re-compression ratio.

Chapter 4

Segment Reordering

4.1 Motivation

As previously explained, the CTTS pre-selected database contains many acoustic leaves, each containing up to ten segments with one or more frames of spectral parameters. The original order of the segments in the leaf has no significance. Since we wish to compress the segments jointly, their order is relevant to the overall performance. In the vectorial polynomial TD approach, we concatenate the speech segments into one super-segment which is then fed to the algorithm. The order of the segment will affect the smoothness of the super-segment. In the 3D SADCT approach, the order of the segments with affect the results of the transform in the vertical direction. Therefore, prior to applying either compression algorithm, we must address the challenge of finding an optimal segment ordering.

First we must define a cost function, assigning a cost to each possible order, such as the degree of discontinuity at adjacent segment joints. Then, we need to seek a method to find the order that minimizes this cost.

This is actually a form of the renown Traveling Salesperson Problem (TSP), from the realm of Combinatorial Optimization [20]. We will describe some possible solutions to this problem and their applicability to optimal ordering.

We note that a trellis, or Viterbi, based approach is not applicable here. The requirement that the optimal path through the trellis contain the optimal sub-paths doesn't hold. Since each segment can be selected only once, when advancing through the trellis previous decisions affect the available options, which is inconsistent with Viterbi optimization requirements.

4.2 Cost Function

In order to determine the best order, we must first define a cost function that can be evaluated for each proposed order. The ordering algorithm will then seek to minimize this function. We will suggest a corresponding cost function for each of the (re)compression approaches: Polynomial TD which requires smoothness at segment joints, and SADCT which requires maximum energy compaction after applying the transform.

4.2.1 Polynomial TD Cost Function

A simple measure of ensuring smoothness of the re-ordered leaf could be obtained by minimizing the sum of Euclidean distances at each of the segment joints. However, since the algorithm's goal is to pass a polynomial through the parameter trajectories, our goal is actually to find a segment order so that when they are concatenated they can be easily fitted with a low order polynomial. Therefore we must also take into account the discontinuity in the derivative at the joints. We evaluated a number of such costs based on the degree of discontinuity and derivative discontinuities as segment joints.

We also examined an alternative smoothness measure that was found by fitting the re-ordered data with a N^{th} order polynomial (N=1,2,3 or 4 were checked) and defining the cost as the mean square distance between the polynomial and the actual data points.

These costs were all evaluated in two setups: calculated only for the first feature of the 32 element parameter vector - for simplicity, or calculated and averaged over all 32 features.

All proposed cost function were evaluated within a simple compression setup. The different approaches gave very similar results, providing a minor improvement in overall algorithm performance when comparing to the case with no reordering. The similarity in the results of the different approaches is probably due to the fact that for the most part they all provided a "sensible" ordering solution - with many of the abrupt discontinuities appearing within the segments.

The selected cost was the distance of the actual values from a second order polynomial passed through them, evaluated over all 32 elements of

the vector, for each of the N frames in the leaf:

$$cost_{TD} = \sum_{i=1}^{32} w_i \sum_{n=1}^{N} (actualVal_{n,i} - polynomialVal_{n,i})^2 \tag{4.1}$$

$$w_i = \frac{const}{i} \tag{4.2}$$

The weighting scheme prioritizes the lower frequency parameters, as they are perpetually more important. The constant is selected so that the obtained cost values of the various approaches are of the same order of magnitude.

Simulations showed this cost function provided a very slight improvement over the other methods. Note that the complexity is not of great concern as the ordering is performed once, off-line, and then can be kept per leaf, for use in evaluating various coding setups.

4.2.2 SADCT Cost Function

For our second compression approach we wish to obtain maximum energy compaction after performing the transform. As a first stage, where we evaluated the contribution of optimal reordering for SADCT coefficient energy compaction, we used a cost function that estimates the compaction obtained for a given order. Since the transform complexity is relatively low, we perform the SADCT for each evaluated order and calculate the cost based on the number of coefficients that contain 95% of the total leaf energy. The results provided later in this chapter were obtained using this cost function.

In our proposed SADCT based algorithm, the leaf data is split into M groups, $\{G_m\}_{m=1}^{M}$, of varying bit-rates, as we have described in Chapter 3. Our goal is to contain as much of the leaf energy as possible in G_2, the lowest non-DC frequency group. Therefore we defined the reordering cost function, used for the proposed recompression algorithm as:

$$C = 1 - \frac{\sum_{\{u,v\} \in G_2} \sum_{w=1}^{32} F_{u,v,w}^2}{\sum_{\{u,v\} \in G_{2,..,M}} \sum_{w=1}^{32} F_{u,v,w}^2} \tag{4.3}$$

Where $F_{u,v,w}$ are the 3D SADCT coefficient values. All the results of the proposed SADCT based algorithm with reordering, provided in the Chapters 4 and 6, were obtained using this cost function.

4.3 Examined Reordering Approaches

The most straightforward method of finding the optimal order, is to perform an exhaustive search over all possible orders and select the order resulting in the lowest cost. While this is possible for the smaller leaves, containing seven segments or less, it becomes too cumbersome for the larger leaves. Leaves with 10 segments would require evaluation of over 3.6 million candidate orderings. Therefore we require a faster algorithm for finding the segment ordering with the lowest cost, for large leaves.

We begin by examining a number of previously proposed approaches for finding the optimal order among data units.

In [61] Zeger and Gersho seek an optimal ordering of a vector quantization codebook for transmission of its indices over a noisy channel. The underlying assumption is that the bits representing the codebook entry suffer from noise and the codebook vectors should be ordered in such a way that the distortion added by the erroneous reconstruction is minimal. An appropriate cost function is defined and the Binary Switching Algorithm (BSA) is introduced to find the optimal order. In BSA we begin with a random ordering and define C as a cost function for a given ordering. We randomly switch two entries and check if C is reduced. If so, keep the switch - otherwise, discard it. This continues until C reaches the required minimum or no more 'successful' switches can be found. The BSA algorithm outline is as follows:

1. Select an initial order, and set C_{min} to its cost.
2. Perform a random switch between two elements.
3. Calculate the new cost C_n.
4. IF $C_n < C_{min}$ then $C_{min} = C_n$ and the switch is accepted.
5. IF not converged AND maximum number of iterations not reached: GOTO 2.
6. END.

Zeger and Gersho also propose to enhance this basic BSA by selecting a 'clever' non-random starting point, or choosing various random starting points and selecting the best result. Also, upon convergence to the minimum

cost (local minima) they propose to perform some random perturbations and check if the cost is reduced. If so the iterative process is resumed.

In [39] and [38] Memon *et* al. solve a similar problem of optimal color ordering for improved predictive coding. Three approaches are examined:

1. Sorting color indices in lexicographic order.

2. A pair-wise merging algorithm. This algorithm attempts to find an optimal ordering by sequentially merging pairs of ordered color sets until all colors have been merged.

3. A Simulated Annealing approach - a detailed description of which is provided below.

In [55] Spira addressed the problem of adapting images to color limited displays. One of the stages to achieve this goal was to reorder the color indices, so that correlation between adjacent pixels in the image is maximized and quantization error resulting in selecting a neighboring color is minimized. This required finding an optimal color-ordering. Several approaches examined, that do not extend to our problem setup, include projections of three dimension color space to the one-dimensional color index space where the distances are then minimized, selection of color orders according to the LBG trees used for their compression, and partitioning of the 3D color space into sub-cubes using Hilbert scan. In addition classic TSP solution methods - Farthest Insertion Algorithm (FIA) and the dual Nearest Insertion Algorithm (NIA) [30] were examined. However, FIA and NIA require a cost function for which the triangle inequality holds, which isn't the case in our problem setup. Spira also examined the use of BSA for optimal ordering and offered to improve on it by allowing for reordering by performing a random 'move' rather than a random 'switch'. This allows for greater flexibility in obtained orders, and is illustrated in Fig. 4.1. This method achieved good results, but had very high complexity when implemented on the 256 color indices of the color table.

Thus, from the above prior works, the two methods that appear relevant to our ordering problem are the improved BSA, proposed by Spira, and the Simulated Annealing approach which is described next.

Simulated annealing (SA) [28] is a generic probabilistic meta-algorithm for global optimization problems, namely locating a good approximation to the global optimum of a given function in a large search space.

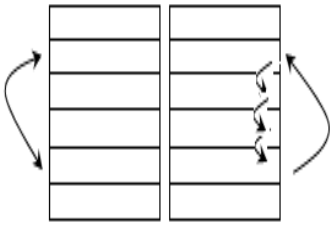

Figure 4.1: Illustration of random perturbations for the Binary Switching Algorithm. Left: "switch" ; Right: "move"

Our problem, optimal segment ordering, is a classic combinatorial optimization problem often solved by the SA approach. The idea of SA is to mimic the behavior of atoms in a slow cooling process which brings them to organize themselves in a structure with minimal energy. To apply SA, an 'energy' function must be defined as well as a cooling schedule. The idea is that, in a fashion similar to the atomic behavior, pseudo-random moves are made such that overall, if the cooling is performed correctly, the system converges into its lowest energy state.

In SA a random change is accepted not only if the cost function is reduced, but at a certain probability even if it increases. This probability is dependent on the cost increase and the current temperature. If the cooling is instantaneous, i.e., zero temperature, the SA is essentially identical to the BSA algorithm, where only "good" changes are accepted.

The SA may be performed using a Metropolis Monte Carlo algorithm, originally proposed in [40], to investigate equations of state for substances consisting of interacting individual molecules. The Metropolis algorithm has since been widely adopted as a solution to a range of probabilistic problems, such as obtaining a sequence of random samples from a probability distribution.

In addition to applying the Metropolis algorithm, Kirkpatrick, in [28], describes methods to perform the random changes and select and apply a cooling schedule. Application of the proposed SA to a variety of problems is then described, as well of the importance of selecting a successful cooling schedule to obtain convergence.

We chose to examine the BSA, improved BSA and SA algorithms for our segment reordering problem. We performed a performance comparison when adjusting several parameters such as using 'switch' vs. 'move' for the random perturbations (as illustrated in 4.1), using different initial setups (random or sorted by length) and for SA - various candidate cooling schedules.

When comparing the 'move' vs. 'switch' perturbations, we found that for both BSA and SA the 'move' approach provided slightly better results, and thus was adopted.

Since we perform the ordering offline, we may chose the global minima over all iterations, rather than the last obtained value, as done in Real-Time applications. Thus, the cooling schedule or initial setup did not really affect our result, as long as enough iterations were performed (5000). Therefore, rather than concerning ourselves with finding appropriate cooling schedules, we simply apply a Metropolis based algorithm, using a single pre-selected temperature. The proposed Metropolis Based Ordering Algorithm, which we name MBOrd, can be viewed as a single level of the SA algorithm, at a single point of the cooling schedule. Alternatively, it can be viewed as an extended BSA algorithm. In MBOrd, as in BSA and SA, "good moves", i.e., moves that cause a decrease in the cost function are always accepted. As in SA, "bad moves" are accepted at a probability which depends on the amount of increase in the cost function. However, this probability does not converge to zero over time as in SA, due to the usage of a constant 'temperature'.

4.4 Proposed Ordering Algorithm

For small leaves, i.e. containing 7 segments or less, a full search over all possible orders is performed, and the ordering with the lowest cost is selected. For 7 segments 5,040 orderings must be checked, however for 8 segment this grows to an unacceptable value of over 40,000 candidate orderings, which is why we require a sub-optimal, non-exhaustive, approach. As explained above, after examining possible approaches we opted for a Metropolis Based Algorithm, using a "move" random change generator. We will now describe the proposed algorithm.

Given an initial order (the original order in the acoustic leaf), a cost function which we wish to minimize and a "Perturbation Generator", a

unit which enables random order changes (either a random segment move or a random switch between two segments), the algorithm outline is:

1. Initialize: set initial order, and set T to desired temperature.
2. Calculate current cost, C.
3. Perform a random move, and calculate the new cost $Cnew$, and the difference $\Delta C = Cnew - C$.
4. If $\Delta C < 0$ keep the move.
5. If $\Delta C > 0$ and $e^{\frac{-\Delta C}{T}} > rand(0,1)$ also keep the move. [i.e. at a probability that depends on the increase in the cost function, the temperature and the random variable rand(0,1), keep "bad moves"].
6. If termination condition reached: STOP, else: GOTO 2.

Termination condition in our case is simply defined by reaching the maximum number of iterations allowed (set to 5000, which is about the same number of iterations required for optimal ordering of leaves with 7 segments). The appropriate temperature was found for the TD and SADCT setups separately. Note, that the lower the temperature used, the closer we are to a BSA solution since increases in the cost function are accepted with very low probability. On the other hand, using a high temperature causes the algorithm to be pseudo random, accepting almost any new ordering and evaluating a large number of random orders to find the best ordering among them. For each the proposed algorithms, we performed temperature tuning by performing ordering with a range of temperatures, and selecting the temperature that provided the best overall result, i.e the lowest cost on average among all leaves. For the TD algorithm T=0.01 gave the best results. For the SADCT algorithm T=10 provided the lowest overall cost. This high temperature essentially indicates that we examine a large number of pseudo-random setups and choose the one with the lowest cost.

To demonstrate the search for the cost global minima using MBOrd, we show in Fig. 4.2, the obtained TD cost for each "accepted" ordering during the iterative process, for both MBOrd and BSA, for one example acoustic leaf. MBOrd was performed using a constant temperature of T=0.01. Note that BSA performance is monotone, since only "good" changes are accepted, while the MBOrd accepts both good and bad changes, which results in many

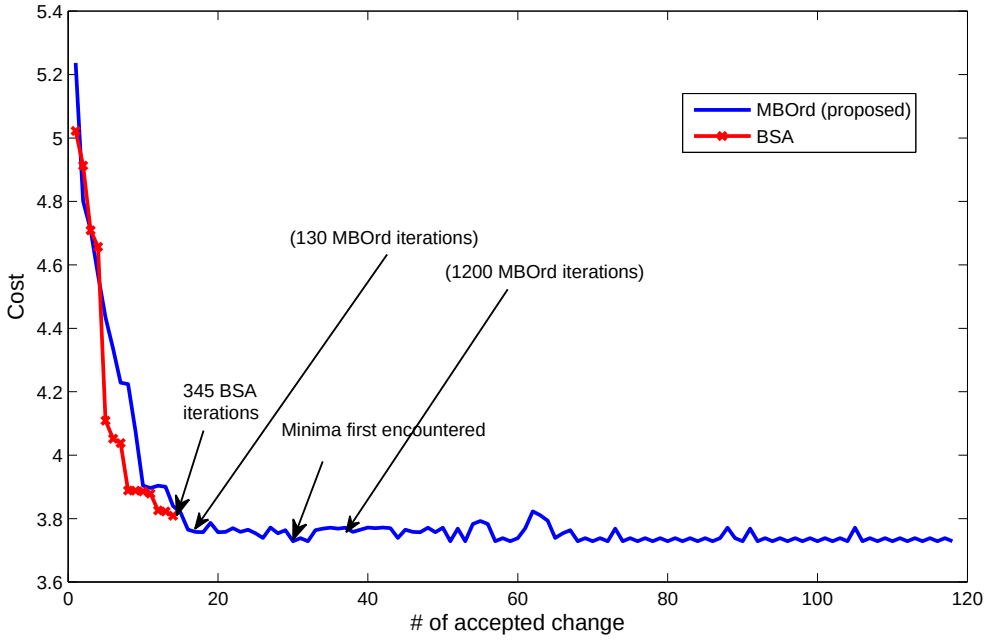

Figure 4.2: TD Costs obtained by MBOrd and BSA: Comparison between the proposed Metropolis Based Sorting Algorithm (MBOrd) and BSA for a single leaf with 5000 iterations and T=0.01. Note the global minimum of the MBOrd is 3.728, vs. 3.809 for BSA. The BSA line is shorter since less changes are accepted.

more "accepted" points. Since we don't use a cooling schedule, but rather a constant temperature, we do not obtain a gradual convergence, but do obtain a better global minima - for this example: 3.728 with MBOrd, vs. 3.809 using BSA.

Using MBOrd and the selected cost functions for TD and SADCT respectively, Figures 4.3 and 4.4 demonstrate the performance obtained for 50 examined leaves. For TD we show the decrease in Mean Squared Error of the reconstructed signal, when compressed to 50% using second order polynomials to represent every six frames (i.e. Polynomial TD with fixed segmentation and polynomial order selection). For SADCT we show the decrease in percent of coefficients that hold 95% of the total energy, as a result

of naive ordering - according to segment length, and MBOrd based ordering.

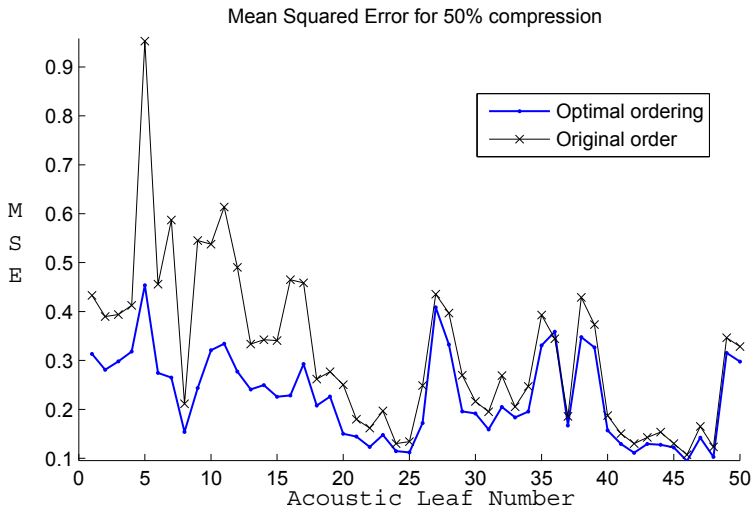

Figure 4.3: TD - Optimal Ordering performance: Mean Squared Error of the reconstructed parameters, after performing Temporal Decomposition, with and without MBOrd reordering.

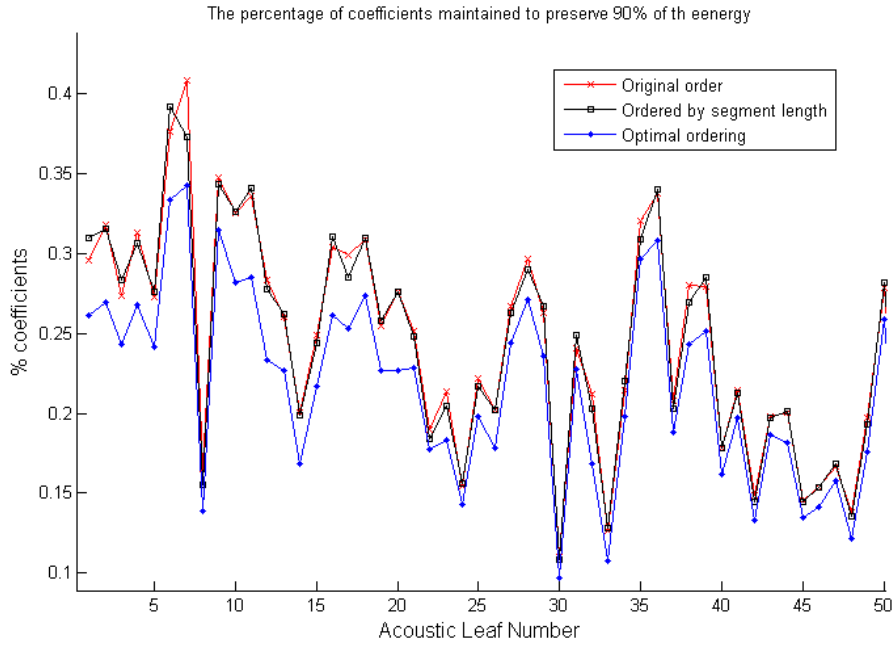

Figure 4.4: SADCT - Optimal Ordering performance: Percentage of SADCT coefficients that hold 90% of the total leaf energy, shown when ordering speech segments according to their original order, sorted according to length and with MBOrd.

4.5 Conclusions

In this chapter we addressed the issue of finding an optimal segment ordering, prior to applying one of the two proposed compression algorithms, polynomial TD or SADCT. We defined appropriate cost functions for each of the algorithms, which indicate how good a given order is. Since the reordering is performed once, offline, we allow for exhaustive search for the optimal order in leaves containing 7 segments or less. For larger leaves we proposed a Metropolis Based Ordering Algorithm (MBOrd), i.e., performing random moves and accepting good moves, but also bad moves at a certain probability. The selected order is the one that provided the lowest cost out of all the orders examined in the 5000 MBOrd iterations. We showed that using this approach we obtained orderings that provided a minor gain in

evaluated criteria such as Mean Squared Error of the reconstructed parameters of the polynomial TD and coefficient energy concentration for SADCT, compared to the case of no reordering. As we will show in Chapter 5 , the optimal ordering had less of an impact on the obtained speech quality than expected. We will elaborate on this and the underlying reasons when we present the experimental results.

Chapter 5

Experimental Results

In this chapter we will present the experimental results of compression of the IBM TTS acoustic leaves, using the algorithms described in the preceding three chapters, namely polynomial Temporal decomposition, 3D Shape adaptive DCT and Metropolis Based Sorting Algorithm.

5.1 Testing setup

All evaluations were performed on a database of acoustic leaves, used in the IBM low footprint TTS [8]. These leaves have undergone pre-selection so that each acoustic leaf in the test database consists of 5-10 speech segments. 10 sample sentences were used, along with the 1661 leaves that are used to create them. The statistics of these leaves are illustrated in Fig. 5.1. Note the majority of very short speech segments which required the use of an acoustic leaf compression approach rather than a segment by segment approach.

The quantizer training for the SADCT quantizers, was performed on the full acoustic leaf database, comprising 23263 acoustic leaves with the statistical characteristics described by Fig. 5.2.

Since our target was reducing the footprint or required storage of the acoustic leaf database, without perceptually affecting obtained speech quality, we use the standardized PESQ [22] - Perceptual Evaluation of Speech Quality, to quantify the perceptual distortion. Given the imperfect quality if the 'original' TTS output, a PESQ score of 3.7 and above has been found to represent speech which is perceptually indistinguishable in listening tests.

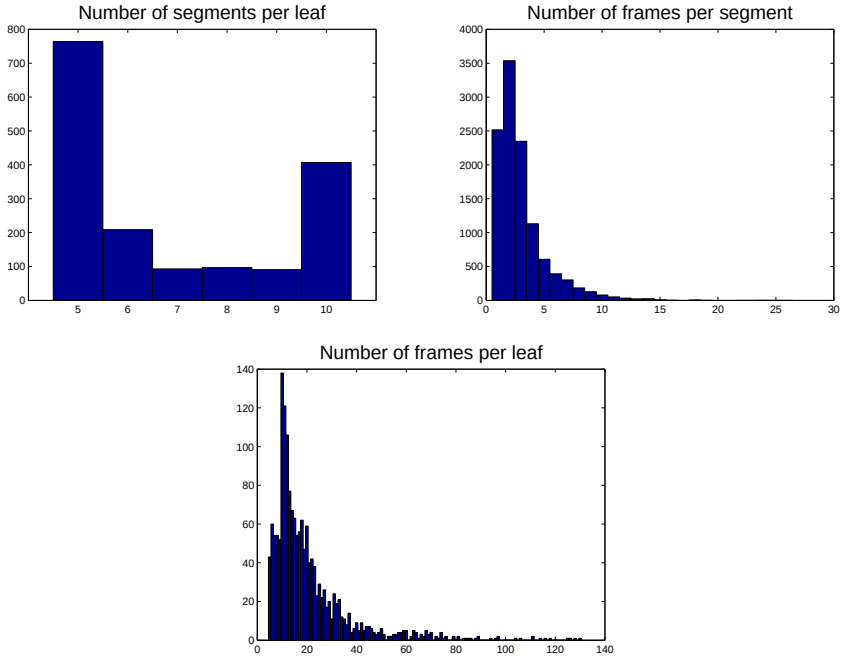

Figure 5.1: Statistics of acoustic leaf database used for testing. Top left: Number of speech segments per acoustic leaf (average: 6.86, median:6). Top right: Number of frames per speech segment (average: 3.01, median: 2). Bottom: Number of frames per acoustic leaf(average: 20.6, median: 15)

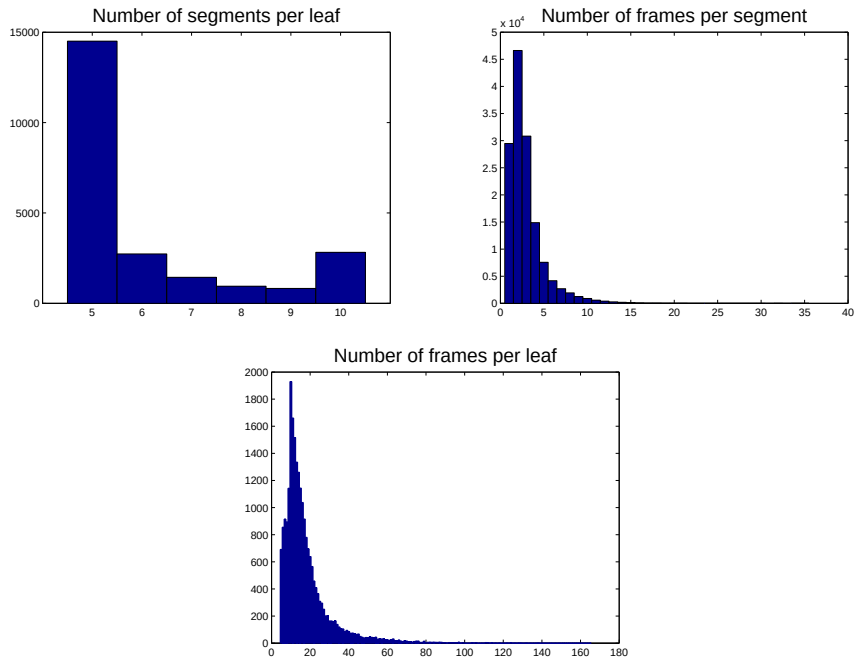

Figure 5.2: Statistics of the full acoustic leaf database, used for quantizer design. Top left: Number of speech segments per acoustic leaf (average: 6.11, median:5). Top right: Number of frames per speech segment (average: 2.93, median: 2). Bottom: Number of frames per acoustic leaf(average: 17.9, median: 14)

5.2 Polynomial Temporal Decomposition

As described in Chapt. 2, a number of polynomial TD setups were evaluated and a performance comparison was performed. The conclusion was that using jointly optimal segmentation and polynomial order selection along with optimal reordering provided the best overall results.

In Tab. 5.1 we compare the results obtained for the proposed polynomial TD with and without applying optimal ordering, for maximum polynomial orders of 4 and 1 (reduced polynomial order setup). As can be seen, optimal ordering offers a slight average improvement only for the low polynomial order setup, where most TD segments are short and therefore rarely span across speech segments, and a slightly more significant improvement in the 'full' setup. Note that the worst-case performance (in bold) is consistently improved with use of reordering. Since the ordering increases only encoding complexity, which is performed offline, it may be used despite the minor performance gain.

The reason that the reduced polynomial order slightly outperforms the 'full setup', although we might have expected the opposite, lies in the fact that the overhead required to represent the TD segment information is decreased by 2 bits. Add to this the fact fact that in the 'full' setup, even though there is an inherent preference for long TD segments with high order polynomials, as this reduces relative overhead, the vast majority of TD segments are chosen to be short with low polynomial orders, as demonstrated in Fig, 5.3 and Tab. 5.2. This means our data is well matched with low order polynomials, if the segmentation is performed wisely, thus the reason for the improved performance of the reduced polynomial order, with lower overhead, becomes apparent.

To summarize, the Vectorial Polynomial TD approach, using optimal reordering performed with the proposed MBOrd, and limiting polynomial orders to 0,1, provides perceptually equivalent duality synthesized speech while reducing the storage required for the amplitudes of the acoustic leaf parameters by a factor of 2.

5.3 Shape Adaptive DCT

The PESQ results of the proposed 3D SADCT based algorithm on the 10 sample sentences as well a a summary of these results are provided

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	Avg.
Full no REO	3.56	3.46	3.61	3.51	3.62	3.63	3.52	**3.45**	3.64	3.46	3.55
Reduced PO no REO	3.60	**3.39**	3.82	3.56	3.66	3.71	3.89	3.57	3.68	3.70	3.66
Full w. REO	3.74	3.76	3.60	**3.49**	3.49	3.91	3.95	3.56	3.63	3.53	3.67
Reduced PO w. REO	3.72	3.54	3.70	3.55	3.64	3.72	4.04	**3.51**	3.81	3.65	3.69

Table 5.1: PESQ results for the Full and Reduced Polynomial Order setups of the Polynomial TD based compression - with and without reordering (REO). Worst case score for each setup is emphasized.

Selected pol. order	0	1	2	3	4
Full setup	3581	1934	1195	658	539
Reduced PO. setup	5180	5655	-	-	-

Table 5.2: Polynomial orders selected by the Polynomial TD algorithm for the 'full' and reduced polinomial order setups

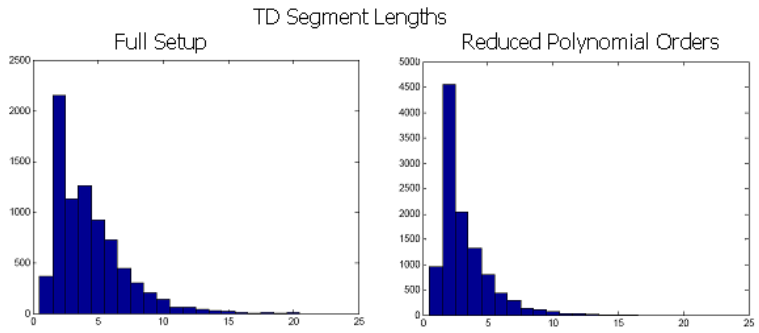

Figure 5.3: Distribution of selected TD segment lengths. Left: Full setup, Right: Reduced polynomial order setup.)

in Table 5.3 and 5.4 respectivley. These results were obtained for a re-compression factor of x2 (43 bits on average per 32 element vectors) and are shown for the case with and without reordering.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
3D SADCT	4.17	4.09	3.59	4.07	3.72	3.53	3.99	3.80	3.81	3.63
3D SADCT+REO	3.73	3.93	3.75	4.04	3.78	3.75	4.03	4.00	3.88	3.65

Table 5.3: PESQ results for the proposed 3D SADCT algorithm for the 10 test sentences

	Avg. PESQ	Min. PESQ	PESQ STD.
3D SADCT	3.84	3.53	0.23
3D SADCT + REO	3.85	3.65	0.14

Table 5.4: PESQ results for the proposed 3D SADCT algorithm - summary

5.4 Discussion

The contribution of the segment reordering to overall performance, for the polynomial TD algorithm depends on the setup. For the 'recommended' setup the reordering did not offer much of an improvement, however some of the other evaluated polynomial TD setups benefited more from the segment reordering as demonstrated in Table 5.1. For the 3D SADCT algorithm, while segment reordering does not contribute much for the overall or average performance, it does improve the worst case performance, and since it is performed offline this may justify its usage. Regarding the two proposed approaches: The SADCT achieves slightly higher PESQ scores and has lower complexity, as fast DCT implementations are widely available. The improved performance may be due to the fact that while the polynomial TD approach removes redundancies between consecutive frames only, the SADCT approach can also remove the redundancies between the speech segments in the same leaf. Note that both approaches remove some of the redundancy along the feature vectors, the SADCT via the transform and the polynomial TD via IBMs differential quantization of the feature vectors.

The polynomial TD approach provided slightly lower PESQ scores but has two significant advantages:

1. Polynomial TD can easily adapted to any desired compression ratio, simply by providing the functions with a different target ratio value, whereas for the case of the SADCT this requires design of new quantizers.

2. Polynomial TD can be merged into an "active" current system in a much more seamless way as the compressed stored data has the same characteristics of the original stored data.

Using either algorithm to extract a single speech segment from an acoustic leaf, incurs some overhead. For the 3D SADCT algorithm the inverse quantization and transform must be applied to the entire leaf, though these are quite low complexity operations. For the Polynomial TD algorithm, any other speech segment frames in the acoustic leaf, that lie in the same TD segment as the frames of the desired speech segment must also be 'decoded'. Using short TD segments decreases this overhead.

Overall, the choice of which algorithm to use depends very much on the specifics of the target application. Both algorithms however reached the goal of recompression by x2 of the amplitude spectral parameters, without any noticeable degradation in the synthesized speech quality.

Chapter 6

Conclusion

6.1 Summary of Proposed Algorithms

In this work we proposed two approaches for recompression of spectral amplitude parameters held in a CTTS acoustic leaf database. Our aim was to develop generic recompression approaches that enable removal of redundancy between parameter vectors that are grouped into a structure with a shared context, and are therefore expected to have a reasonable amount of redundancy. As a test case for the proposed approaches, we aim to recompress the amplitude parameter vectors of a the acoustic leaves in a small footprint CTTS synthesizer, without compromising the perceptual quality of the obtained synthesized speech.

In the first approach, Vectorial Polynomial TD, we remove the long-term redundancy between the amplitude parameter vectors of consecutive speech frames. We begin by concatenating the speech segments in each acoustic leaf into a single 'super-segment'. We define the recompression goal as obtaining a target bit rate with minimal distortion. We can then choose an appropriate working point, where no perceptual loss is present in the synthesized speech. We use an iterative rate-distortion optimization approach, where in each iteration we have a maximal allowed distortion value, D_g. This limits the maximal distortion in each frame of each segment of each leaf. Using this bound, for each acoustic leaf or corresponding 'super-segment', we determine optimal TD segmentation points along with optimal polynomial order per TD segment. Each TD segment, represented by a polynomial of order P, is sampled at P+1 points, and these samples, along with the segment length and value of P, represent the reduced data

set. The obtained rate is calculated and, if necessary, D_g is adjusted and the procedure is repeated. We examined a number of distortion functions and found that the MSE between the actual parameters and the reconstructed parameters - prior to quantization, provided the highest PESQ scores for a given rate. Limiting the maximum distortion provided better performance than limiting the mean distortion. We examined various variants of this approach, allowing different TD segment lengths, polynomial orders and without optimal segmentation. We showed that when performing recompression by a factor of 2, we obtained a PESQ score of about 3.7, relative to the original synthesized speech generated from the original compressed database. This score indicates equivalent perceptual quality. Listening to the sentences shows they are indistinguishable.

In the second proposed approach, 3D SADCT, we aim to remove redundancies not only between consecutive speech frames but also between speech segments and along the parameter vectors. This approach enables redundancy removal along all three axis of the 3D data structure, thus improving recompression performance. 3D SADCT was applied to each acoustic leaf and the obtained coefficients were quantized. The energy compaction property of the DCT helps remove the temporal redundancy within speech segments, as well as the redundancy between segments in the same acoustic leaf and redundancies in the spectral amplitudes of each frames. (Note that the IBM quantizer we used for the polynomial TD approach also removes some of the spectral redundancy by performing differential coding of these values). The main challenge in this approach was the quantizer design. A methodical bit allocation and splitting algorithm was proposed to first divide the vectors into different groups, each with a corresponding bit allocation. Then, for the vectors of each group, a split-VQ was designed using the same methodical bit allocation and split algorithm. We also made sure that the footprint of each of the obtained codebooks was small enough, since we require more codebooks than in the original approach, due to the split into groups. The performance evaluation showed that, for a recompression factor of 2, we obtained PESQ scores of about 3.85. Again, when listening to the original and recompressed sentences, they are indistinguishable.

For both re-compression approaches we suggested performing optimal ordering of the speech segments prior to re-compression. The proposed algorithm, MBord, enables offline minimization of a predefined cost function, which aims to maximize the recompression gain. For the case of polynomial

TD, the cost function reflects the smoothness at speech segment joints. For 3D SADCT it aims to pack as much energy as possible into the first non-DC coefficient group, which has the highest bit allocation and thus are expected to introduce the lowest quantization error. The cost function is minimized by performing random order changes and keeling any changes that cause a decrease in the cost function, but also at a certain probability those that cause an increase in the cost function - thus avoiding convergence to a local minima.

Regarding complexity of the proposed algorithms, the goal of low complexity decoding was achieved. The proposed reordering algorithm adds complexity only to the encoding process. The recompression based on Polynomial TD provides the option of using low complexity setups where the decoding requires at most linear interpolation for reconstruction of the parameters. For the 3D SADCT based recompression, decoding requires inverse SADCT of each leaf, which is slightly more complex, however due to the popularity of the DCT in a variety of applications, many highly optimized implementations for embedded devices are available, substantially reducing the expected run-time of this algorithm. Thus the decoding complexity of the proposed algorithms is indeed small in comparison with the other blocks required for speech synthesis.

6.2 Main Contributions

- Extension of the fixed segment length, scalar, polynomial modeling proposed in [15], by Dusan *et* al., to a variable segment length, vectorial form of polynomial modeling. In [15] polynomial modeling of a trajectory of parameters such as an LSF coefficient is proposed using a pre-determined segment length N, and a pre-determined polynomial order P. We extended this to allow adaptation of both N and P to the local data. We enforce the same local N and P values for all vector elements, to allow for Vector Quantization (VQ) of the obtained P+1 sample vectors. Our extension may be used for compression of any vector parameter set that has a degree of temporal redundancy. The algorithm can be easily adjusted to any desired target rate and can seamlessly reuse the quantization scheme developed for the original data set as we do not alter the behavior of the 'samples'.

- Application of the segmentation and model order selection proposed with various variations in [45], [47], [46] and [44], by Prandoni *et* al., to the problem of segmentation and polynomial order selection for the acoustic leaf re-compression, in combination with vectorial polynomial TD. For this purpose we extended the 2D scheme proposed previously, to a 3D scheme, in order to enable application to our vectorial data. We then combined this per-leaf optimization with an iterative Rate-Distortion optimization over all the acoustic leaves to obtain the target bit-rate over the database, with minimal distortion.

- Application of the 3D SADCT, previously used for hyperspectral image compression, to a new challenge of acoustic leaf (re)compression. Whereas the use of DCT, in general, and SADCT, in particular, is common in image and video coding, it is not often applied to speech coding. We showed that for acoustic leaf data where redundancy exists between speech frames in the same segment, between segments in the same leaf and between parameters in each feature vector, the energy compaction property of the DCT allows for high quality re-compression. The algorithm is useful for any 3D compression problem, as long as the data exhibits a degree of redundancy.

- We proposed a methodical bit allocation and splitting algorithm for split VQ design. While previous algorithms usually assume a known split along the vector, for instance using pre-determined sub-bands, or else the split is determined heuristically, we proposed an algorithm that provides a methodical split into element groups. This is achieved by allocating bits to each of the elements, and then clustering these allocation values into groups, either under a constraint of the number of element groups, or under a constraint on the maximum number of elements per group.

- We proposed an algorithm for optimal segment ordering, MBord, based on the Metropolis algorithm [40], that enables minimization of a target energy or cost function when an exhaustive search over all segment orders is not possible. The proposed algorithm utilizes concepts from the Binary Switching Algorithm in [61] and from the Simulated Annealing proposed in [28]. We used the algorithm with appropriate cost functions for both re-compression approaches, and

obtained a slight gain in overall performance. For other applications, where offline ordering has a greater impact on overall performance, MBord may provide a more significant advantage.

6.3 Future Research Directions

This research can be further developed in a number of directions, as described in the following subsections.

6.3.1 Further Work on Proposed Algorithms

The algorithms we proposed could be further tuned or extended in the following ways:

- Further work on the proposed polynomial TD recompression algorithm may include further investigation of error weighting functions that may provide better results for the embedded quantization setup. Specifically, replacing the current IBM quantizer with a quantizer that has an analytically shaped error could improve the performance of this setup, but was out of scope of the current research.
- We used polynomial based TD since the speech segments are short, however once we concatenate all the segments in the leaf into a single 'super-segment' other TD approaches may be applied and evaluated. This opens a wide range of options for further investigation. Note, that other TD approaches will have to find a way of dealing with the discontinuities in and between speech segments, which we dealt with via adaptive segmentation.
- Developing quantizers for additional working points of the SADCT scheme may prove an interesting challenge. While we proposed a methodical split and bit allocation scheme, a number of parameters such as the DC bit allocation and the weights used were still found empirically, and would need to be tuned for different working points.
- The MBOrd, our proposed Metropolis Based segment Ordering algorithm, could be further investigated. It's possible that cost functions could be found that will cause the reordering to have more of an effect on overall results. Application of this algorithm to other offline ordering problems that originally use BSA could also be considered.

6.3.2 Alternative Signal Models

Further research could attempt to apply the proposed compression schemes to different spectral parameters. The spectral parameters used in this version of the small footprint IBM CTTS system limit the obtained speech quality. In an earlier stage of our research we investigated candidate signal models for the system. These included the sinusoidal model presented in [36] and [37], Harmonic plus Noise models such as the one proposed by Stylianou in [56]. A decomposition of the spectra into periodic and a-periodic components was described by d'Alessandro *et* al. in [12] and enhanced by Ahn and Holmes in [2]. Iterative signal subtraction for analysis by synthesis using the sinusoidal model was proposed by George and Smith in [17] with a minor enhancement by Bailly in [4]. An adaptation of the sinusoidal model specifically for TTS applications was proposed by Macon and Clements in [32] and [33]. Incorporating any of these as an alternative spectral representation, could allow for higher quality of the synthesized speech. It would then be interesting to tune and apply the proposed compression schemes to the new acoustic leaf feature sets and evaluate the obtained compression vs. quality behavior.

6.3.3 Phase Compression

The scope of our research was limited to the amplitude parameters, for two reasons. The first is that the amplitude parameters in the system we worked on have a footprint of 5.7 MB, compared to only 1.6 MB of the phase parameters. The other reason is that most of the loss of quality in the synthesized speech of this system stems from the poor phase representation. Therefore, in parallel to this research some alternative phase representations were investigated. GMM based quantization of sinusoidal model phase parameters was examined. The algorithm described in [23] was partially implemented and evaluated, as an undergraduate project in SIPL. It shows promise as a potential method to improve speech quality while maintaining low bit-rates. If this approach were to be used, further recompression would become unnecessary, as the extent of quantization would be the best method to determine the rate vs. quality tradeoff.

Another examined phase representation was the modified group delay originally presented in [5] and implemented and evaluated in another SIPL undergraduate student project. We found that the values representing the

phase information using this model showed quite a bit of temporal redundancy in consecutive frames in voiced areas. Therefore, it would be very interesting to apply the proposed polynomial TD approach to these values.

6.3.4 Applying the Proposed Algorithms to Other Re-Compression Challenges

As we have stated, although we focused on a specific challenge of acoustic leaf compression, the algorithms developed are generic and may be applied to a variety of re-compression setups. A few such examples are:

1. Sign language databases: each sign is represented by a number of instances, each instance by a set of temporal frames, and each frame by a set of parameters that model the hand movements.(An example can be found at: http://archive.ics.uci.edu/ml/datasets/Libras+Movement).

2. Image classification database: For each image class, either a set of sample images, or their parametric representation are stored. (Bag of Words codeword dictionary).

3. Personalized content recommendation system databases, where data is stored in clusters. Each cluster holds a set of data per user, per movie, and is expected to have redundancies.

4. Medical data sets, where each patient may have a set of 2D parametric data (or images) collected over time, for tracking disease or treatment progress.

Appendix A

Polynomial TD Error Analysis

In the vectorial polynomial TD scheme, the error in the reconstructed amplitude parameters, originates from two sources: The model error and the quantization error. In order to evaluate the effect of the quantization of the polynomial samples on the final reconstructed values let's examine a sample segment, which consists of N frames and is modeled with a polynomial of order P. In Fig. A.1, N=10 and P=3.

The original feature set consists of: $\{x_n\}_{n=0}^{N-1}$, these are represented by red Xs in Fig. A.1. After the TD process, a corresponding P^{th} order polynomial is found: $Pol(n) = \sum_{k=0}^{k=P} a_k n^k$. If we were to sample this polynomial at N points, we would obtain reconstructed values $\{\hat{x_n}\}_{n=0}^{N-1}$, with $\hat{x_n} = x_n + d_n$, where d_n represents the model distortion at each sample, which is the distortion we measure when performing optimization. To proceed, we now sample this polynomial at $P+1$ points: $\{n_p\}_{p=0}^{P}$, to obtain $\{s_p\}_{p=0}^{P}$, represented by green circles. These are the features we actually quantize to obtain $\{\tilde{s_p}\}_{p=0}^{P} = Q[\{s_p\}_{p=0}^{P}]$, where $\tilde{s_p} = s_p + e_p$, shown by the black squares. Next, we match the quantized samples with the appropriate polynomial, using least-squares, and obtain the reconstructed polynomial, $\widetilde{Pol}(n) = \sum_{p=0}^{p=P} \tilde{a_p} n^p$. Finally, we sample the reconstructed polynomial at all N sampling points to obtain the reconstructed features: $\{\tilde{x_n}\}_{n=0}^{N-1}$ - shown by the cyan asterisks, and we have: $\tilde{x_n} = x_n + d_n + f(\{e_p\}_{p=0}^{P})$, where the first error component is due to model error, and the second is a function of the quantization error.

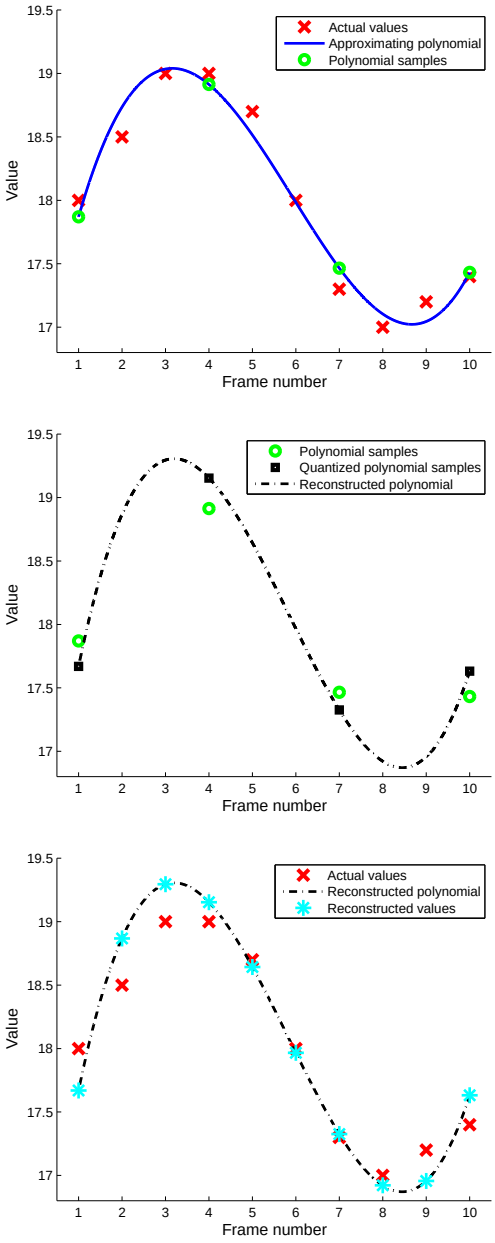

Figure A.1: Polynomial TD reconstruction error. Top shows the model error. Middle shows the added quantization error. Bottom demonstrates the total error

The reconstructed polynomial, defined by $\tilde{a_p}$, is related to the quantized samples, $\tilde{s_p}$, as follows:

$$\begin{bmatrix} \tilde{s_0} \\ \tilde{s_1} \\ \tilde{s_2} \\ \vdots \\ \tilde{s_P} \end{bmatrix} = \begin{bmatrix} 1 & n_0 & n_0^2 & \cdots & n_0^P \\ 1 & n_1 & n_1^2 & \cdots & n_1^P \\ 1 & n_2 & n_2^2 & \cdots & n_2^P \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & n_P & n_P^2 & \cdots & n_P^P \end{bmatrix} \begin{bmatrix} \tilde{a_0} \\ \tilde{a_1} \\ \tilde{a_2} \\ \vdots \\ \tilde{a_P} \end{bmatrix} \tag{A.1}$$

Where n_p are the the sampling points. In compact form we have: $\tilde{S} = V_p \cdot \tilde{A}$, where V_P is a form of the well known Vandermonde matrix. The polynomial coefficients are then found from the quantized samples by:

$$\tilde{A} = V_p^{-1}\tilde{S} = V_p^{-1}(S + E) = V_p^{-1}S + V_p^{-1}E \tag{A.2}$$

where, S is the vector of the un-quantized samples, and E is the vector of the quantization errors. These coefficients are then used to find the reconstructed features by:

$$\hat{x_n} = \sum_{p=0}^{P} \tilde{a_k} n^k \tag{A.3}$$

Thus the final error is related to the original quantization error via the inverse of the Vandermonde matrix. The final error also increases with polynomial order.

The inverse Vanderomnde matrix was examined by Macon and Spitzbart in [34] and Gautschi in [16], and results were later extended and used by Seidner and Feder to evaluate noise amplification in nonuniform sampling in [50]. In these papers the following bound for the norm of the inverse Vandermonde matrix was found:

$$\|V_P^{-1}\| \leq max_{0 \leq j \leq P} \prod_{i=0 i \neq j}^{P} \frac{1 + |n_i|}{|n_i - n_j|} \tag{A.4}$$

As apparent here, the norm of the inverse Vandermonde matrix grows as a function of the absolute values of the sampling points, $\{n_p\}_{p=0}^{P}$. On the other hand, the norm is also inversely dependent on the spacing between sampling points. On examining this norm for the case of $P = 2$, which is easy to compute, we found that the optimal sampling points always include

the two end points, and an additional midpoint whose value changes with segment length. For instance for $N = 5$ the sampling points that minimize the norm are: [1,3,5], for $N = 7$ they are [1,5,7].

To summarize, when evaluating the total quantization based error in the reconstructed feature values we must differentiate between two cases:

1. For sample points, the quantization based error is equal to the actual quantization error.

2. For interpolated points, the quantization based error is related to the actual quantizaton error via the inverse Vandermonde matrix, whose norm depends on the location of sampling points. This error also increases with polynomial order.

From this formulation we deduce two facts:

1. The higher the polynomial order the larger the quantization based error becomes. Therefore, we will limit the allowed polynomial order in our application. The maximum allowed polynomial order is set to 4.

2. In order to reduce errors, we must take care in the selection of sampling points. After examining a number of candidate sampling schemes, we selected a scheme that takes first the values at the segment end points, and then adds equally distributed points within the segment. This is in accordance with eq. (A.4) and its analysis.

Appendix B

LSD Calculation

During the optimization process, we wish to calculate the Log-spectral-Distortion, or LSD, between the original speech spectrum, $A(\omega)$, and the spectrum of the reconstructed speech, $\widehat{A(\omega)}$, which is defined in [29] as:

$$LSD = \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left[10\log_{10}\frac{|A(\omega)|^2}{|\widehat{A(\omega)}|^2}\right]^2 d\omega}$$
$$= 20 * \sqrt{\frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\log_{10}|A(\omega)| - \log_{10}|\widehat{A(\omega)}|\right]^2 d\omega} \tag{B.1}$$

In order to reduce complexity, we wish to calculate the LSD directly in the parameter space. As explained in Chapter 1, the parameters are essentially the Mel-scale sampling of the spectral envelope. Thus by performing the appropriate interpolation we can recreate the spectral envelope and directly calculate the LSD. First we provide a reminder of how the model parameters we wish to compress are obtained, and then show how the LSD measure is calculated using these parameters.

In the analysis process parameters $\tilde{c_n}$ are found, s.t. they minimize the squared error between a frequency warped and re-sampled version of the amplitude line spectrum, and its estimation. The relationship between these intermediate model parameters, $\tilde{c_n}$ and the spectral amplitude $A(\omega)$ is given by:

$$log_2 A(\omega) = \sum_{n=1}^{n=32} \tilde{c_n} \cdot B_n(\omega) \tag{B.2}$$

The triangular basis functions, B_n, are warped using the Mel-frequency

scale. Then, the original frame energy, G, is embedded into the $\tilde{c_n}$ values by:

$$C_n = \tilde{c_n} - \frac{1}{32}\sum_{i=1}^{i=32}\tilde{c_i} - logG \tag{B.3}$$

These C_n are the model parameters which we are working with.

Note, that the intermediate and final model parameters are the same up to a constant (determined per frame). However when calculating the LSD, "DC" or constant differences between the two spectrums, while possibly contributing to enlarge the LSD value, are not really of interest. Therefore we would rather calculate the difference between normalized spectra. Thus rather than attempting to estimate this unknown constant, we simply normalize the reconstructed parameters so that their sum equals the sum of the original parameters, and use these values, $\hat{C_n}$ to calculate the LSD. We note that:

$$log_{10}A(\omega) = \frac{\log_2 A(\omega)}{\log_2 10} \equiv L_1 \log_2 A(\omega) \tag{B.4}$$

Combining the three above equations we obtain:

$$LSD = 20 * \sqrt{\frac{1 \cdot L_1^2}{2\pi}\int_{-\pi}^{\pi}\left[\sum_{n=1}^{3} 2C_nB_n(\omega) - \sum_{n=1}^{3} 2\hat{C_n}B_n(\omega)\right]^2 d\omega} =$$
$$L_2 \cdot \sqrt{\sum_{n=1}^{3} 2\sum_{m=1}^{3} 2(C_n - \hat{C_n})(C_m - \hat{C_m})\int_{-\pi}^{\pi} B_n(\omega)B_m(\omega)d\omega} \tag{B.5}$$

Where: $L_2 = \frac{20L_1}{\sqrt{2\pi}} = \frac{20}{\log_2 10} \cdot \frac{1}{2\pi}$ Due to the structure of the basis functions, each function overlaps only with the preceding and following functions. Thus the integral in the above expression is non-zero only when $m = \{n-1, n, n+1\}$. A simple variable substitute of $k = m - n + 2$ gives us:

$$LSD = L_2 \cdot \sqrt{\sum_{n=1}^{3} 2\sum_{k=1}^{3} (C_n - \hat{C_n})(C_{k+n-2} - \hat{C_{k+n-2}})\int_{-\pi}^{\pi} B_n(\omega)B_{k-n+2}(\omega)d\omega} \tag{B.6}$$

Note that the integral value, $B_{int}(n,k)$ depends solely on n and k and thus

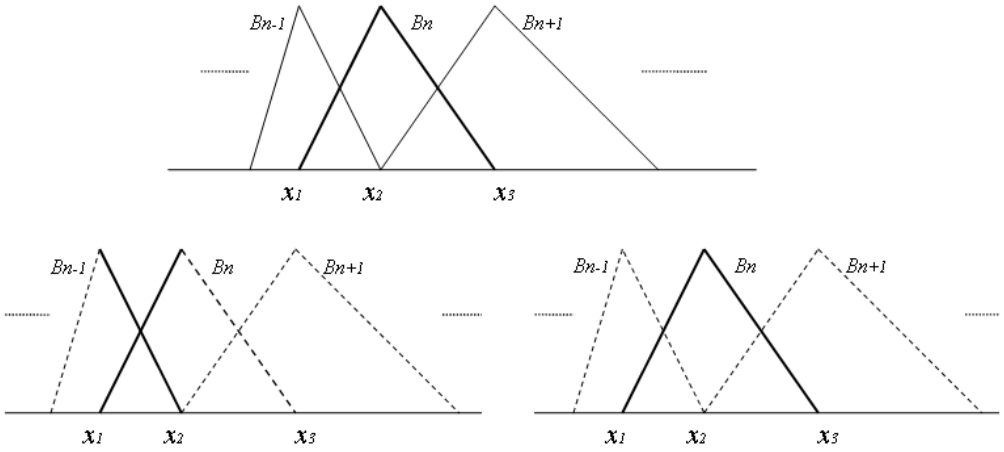

Figure B.1: Basis functions used in LSD calculation. Top: central basis function B_n and its two neighbors. Bottom left: The slopes used to calculate for $k = 1$. Bottom right: The slopes used to calculate for $k = 2$.

can be pre-computed. We have:

$$LSD = L_2 \cdot \sqrt{\sum_{n=1}^{3} 2 \sum_{k=1}^{3} (C_n - \hat{C}_n)(C_{k+n-2} - \hat{C_{k+n-2}}) B_{int}(n,k)} \qquad \text{(B.7)}$$

It remains to examine the values of $B_{int}(n,k)$: Figure B.1 demonstrates the structure for a central basis function and its two neighbors. We need to calculate two types of integrals: For $k = 1$ or 3 (Fig. B.1 illustrates the $k = 1$ case, the $k = 3$ case is equivalent):

$$B_{int}(n,1) = \int_{x_1}^{x_2} \frac{x_2 - x}{x_2 - x_1} \cdot \frac{x - x_1}{x_2 - x_1} dx = \frac{1}{6}(x_2 - x_1) \qquad \text{(B.8)}$$

$$B_{int}(n,3) = \frac{1}{6}(x_3 - x_2) \qquad \text{(B.9)}$$

For $k = 2$ we need the integral of $B_n \times B_n$:

$$B_{int}(n,2) = \int_{x_1}^{x_2} (\frac{x - x_1}{x_2 - x_1})^2 dx + \int_{x_2}^{x_3} (\frac{x_3 - x}{x_3 - x_2})^2 dx = \frac{1}{3}(x_3 - x_1) \qquad \text{(B.10)}$$

Combining these expressions provides a straightforward and simple way

to calculate the LSD using our model parameters. The x_i values are simply the peaks of the B_n functions, i.e. the 32 frequency points equally spaced in the Mel-scale domain. Their values in Hz for our setup are:

[62.4, 130.5, 204.5, 285.2, 373.1, 468.9, 573.1, 686.7, 810.4, 945.2, 1091.9, 1251.8, 1425.9, 1615.6, 1822.1, 2047.1, 2292.2, 2559.2, 2849.9, 3166.6, 3511.5, 3887.2, 4296.5, 4742.2, 5227.7, 5756.5, 6332.5, 6959.95, 7643.2, 8387.5, 9198.2, 10081.2];

The proposed measure can be easily computed during the optimization process so that the distortion we are minimizing is the actual LSD rather than just an Euclidean distance measure between the parameter vectors, in hope of improving overall performance.

Bibliography

[1] Y. Agiomyrgiannakis and Y. Stylianou. Stochastic modeling and quantization of harmonic phases in speech using wrapped gaussian mixture models. *ICASSP*, 4(2):IV–1121–4, May 2007.

[2] R. Ahn and W.H. Holmes. An improved harmonic-plus-noise decomposition method and its application in pitch determination-effectiveness of a periodic and aperiodic decomposition method for analysis of voice sources. *Proc. Of IEEE Workshop on Speech Coding For Telecommunications*, pages 41–42, Sep. 1997.

[3] C. Athaudage, A. Bradley, and M. Lech. Model-based speech signal coding using optimized temporal decomposition for storage and broadcasting applications. *EURASIP Journal On Applied Signal Processing*, pages 1016–1026, Oct. 2003.

[4] G. Bailly. Accurate estimation of sinusoidal parameters in an harmonic+noise model for speech synthesis. *Proc. of the European Conference on Speech Communication and Technology*, pages 1051–1054, Sep. 1999.

[5] H. Banno, J. Lu, S. Nakamura, K. Shikano, and H. Kawahara. Efficient representation of short-time phase based on time-domain smoothed group delay. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 86(10):56–64, 2003.

[6] J. Benesty, M. M. Sondhi, and Y. Huang. *Springer Handbook on Speech Processing*, chapter Part D: Text - to - Speech Synthesis, Chapters 19-21, pages 421–455. Springer Berlin Heidelberg, 2008.

[7] S. Chatterjeet and T. V. Sreenivas. Optimum transform domain split VQ. *IEEE Signal Proc. Letters*, 15:285–288, 2008.

[8] D. Chazan, R. Hoory, Z. Kons, A. Sagi, S. Shechtman, and A.Sorin. Small footprint concatenative text-to-speech synthesis system using complex spectral envelope modeling. In *Eurospeech*, Lisbon, Portugal, Sep. 2005.

[9] D. Chazan, R. Hoory, Z. Kons, D. Silberstein, and A. Sorin. Reducing the footprint of the IBM trainable speech synthesis system. In *7th Int. Conf. Spoken Language Processing (ICSLP)*, Denver, USA, Sep. 2002.

[10] D. Chazan, R. Hoory, A. Sagi, S. Shechtman, A.Sorin, Z. Shuang, and R. Bakis. High quality sinusoidal modeling of wideband speech for the purpose of speech synthesis and modification. In *ICASSP*, Toulouse, France, May 2006.

[11] J.S. Collura, A. McCree, and T.E. Tremain. Perceptually based distortion measurements for spectrum quantization. *IEEE Workshopon Speech Coding for Telecommunications*, pages 49–50, Sep. 1995.

[12] C. d'Alessandro, V. Darsinos, and B. Yegnanarayana. An iterative algorithm for decomposition of speech signals into periodic and aperiodic components. *IEEE Transactions on Speech and Audio Processing*, 6:1–11, Jan. 1998.

[13] R.E. Donovan, A. Ittycheriah, M. Franz, B. Ramabhadran, E. Eide, M. Viswanathan, R. Bakis, W. Hamza, M. Picheny, P. Gleason, T. Rutherfoord, P. Cox, D. Green, E. Janke, S. Revelinand C. Waastand B. Zeller, C. Guenther, and J. Kunzmann. Current status of the IBM trainable speech synthesis system. In *4th ISCA Tutorial and Research Workshop on Speech Synthesis*, Scotland, UK, Aug. 2001.

[14] S. Dusan, J.L. Flanagan, A. Karve, and M. Balaraman. Speech coding using trajectory compression and multiple sensors. In *International Conference on Spoken Language Processing*, pages 1993–1996, Jeju Island, Korea, Oct. 2004.

[15] S. Dusan, J.L. Flanagan, A. Karve, and M. Balaraman. Speech compression by polynomial approximation. *IEEE Transactions*

on Audio Speech and Language Processing, 15(2):387–395, Feb. 2007.

[16] W. Gautschi. On the inverse of vandermonde and confluent vandermonde matrices. *Numerische Mathematik*, 4:117–123, Dec. 1962.

[17] E.B. George and M.J.T. Smith. Speech analysis/synthesis and modification using an analysis-by-synthesis/overlap-add sinusoidal model. *IEEE Transactions on Speech and Audio Processing*, 5(5):389–406, Sep. 1997.

[18] A. Gersho and R.M. Gray. *Vector Quantization and signal compression.* Kluwer Academic Publishers, Boston, 1991.

[19] L. Girin, M. Firouzmand, and S. Marchand. Perceptual long-term variable-rate sinusoidal modeling of speech. *IEEE Transactions on Audio Speech and Language Processing*, 15(3):851–861, Mar. 2007.

[20] B. Golden, L. Bodin, T. Doyle, and W. Stewart Jr. Approximate traveling salesman algorithms. *Operations Research*, 28(3):694–711, May-June 1980.

[21] P. Hedelin and J. Skoglund. Vector quantization based on gaussian mixture models. *IEEE Transactions on Speech and Audio Processing*, 8(4):385–401, Jul. 2000.

[22] ITU-T Rec. P.862.2. Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs, Nov. 2005.

[23] J.Lindblom. A sinusoidal voice over packet coder tailored for the frame-erasure channel. *IEEE Transactions on Speech and Audio Processing*, 13(5):787–798, Sep. 2005.

[24] A.B. Kain and J.P.H. van Santen. Compression of acoustic inventories using asynchronous interpolation. In *IEEE Workshop on Speech Synthesis*, pages 83–86, Sep. 2002.

[25] A.B. Kain and J.P.H. van Santen. A speech model of acoustic inventories based on asynchronous interpolation. In *Eurospeech*, pages 329–332, Geneva, Switzerland, Sep. 2003.

[26] A.B. Kain and J.P.H. van Santen. Unit-selection text-to-speech synthesis using an asynchronous interpolation model. In *6th ISCA Workshop on Speech Synthesis*, Bonn, Aug. 2007.

[27] S. Karabetsos, P. Tsiakoulis, A. Chalamandaris, and S.Raptis. Embedded unit selection text-to-speech synthesis for mobile devices. *IEEE Trans. on Consumer Electronics*, 55:613–621, 2009.

[28] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science, New Series*, 220(4598):671–680, May 1983.

[29] W.B. Kleijn and K.K. Paliwal, editors. *Speech Coding and Synthesis*, chapter 12. Elsevier Science Inc, NY, USA, 1995.

[30] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Interscience Series in Discrete Mathematics. Wiley, New York, 1985.

[31] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28(1):84–95, Jan. 1980.

[32] M. W. Macon and M. A. Clements. Sinusoidal modeling and modification of unvoiced speech. *IEEE Transactions on Speech and Audio Processing*, pages 557–560, Nov. 1997.

[33] M. W. Macon and M. A. Clements. An enhanced ABS/OLA sinusoidal model for waveform synthesis in TTS. *Proc. of the European Conference on Speech Communication and Technology*, pages 2327–2330, Sep. 1999.

[34] N. Macon and A. Spitzbart. Inverses of vandermonde matrices. *Amer. Math Monthly*, 65:95–100, Feb. 1958.

[35] D. Markman and D. Malah. Hyperspectral image coding using 3D transforms. In *International Conference on Image Processing (ICIP)*, volume 1, pages 114–117, Thessaloniki, Greece, Oct. 2001.

[36] R.J. McAulay and T.F. Quatiere. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustic, Speech and Signal Processing*, ASSP-34:744–754, Aug. 1986.

[37] R.J. McAulay and T.F. Quatiere. *Speech Coding and Synthesis*, chapter 4. Elsevier Science Inc., New York, 1995.

[38] N.D. Memon and R. Rodila. Transcoding GIF images to JPEG-LS. *IEEE Trans. Consumer Electronics*, 43(3):423–429, Aug. 1997.

[39] N.D. Memon and A. Venkateswaran. On ordering color maps for lossless predictive coding. *IEEE Trans. on Image Processing*, xx(11):1522–1527, Nov. 1996.

[40] N. Metropolis, A. Rosenbluthand M. Rosenbluth., A. Teller., and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, June 1953.

[41] P. C. Nguyen, M. Akagi, and B. P. Nguyen. Limited error based event localizing temporal decomposition and its application to variable-rate speech coding. *Speech Communication*, 49(4):292–304, Apr. 2007.

[42] R. Nygaard and D. Haugland. Compressing ECG signals by piecewise polynomial approximation. In *ICASSP*, volume 3, pages 1809–1812, Seattle, May 1998.

[43] R. Nygaard, G. Melnikov, and A. K. Katsaggelos. A rate distortion optimal ecg coding algorithm. *IEEE Transactions on biomedical engineering*, 48(1):28–40, Jan. 2001.

[44] P. Prandoni. *Optimal Segmentation Techniques for Piecewise Stationary Signals*. PhD thesis, EPFL, 1999.

[45] P. Prandoni, M. Goodwin, and M. Vetterli. Optimal time segmentation for signal modeling and compression. In *ICASSP*, volume III, pages 2029–2032, Munich, Apr. 1997.

[46] P. Prandoni and M. Vetterli. Approximation and compression of piecewise smooth functions. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, 357(1760):2573–2591, Sep. 1999.

[47] P. Prandoni and M. Vetterli. R/d optimal linear prediction. *IEEE Trans. on Speech and Audio Proc.*, 8(6):646–655, Nov. 2000.

[48] J. Volkman S. S. Stevens and E. B. Newman. A scale for the measurement of the psychological magnitude of pitch. *Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[49] G.M. Schuster and A.K. Katsaggelos. *Rate-Distortion Based Video Compression.* Kluwer Academic Publishers, Dordrecht, 1997.

[50] D. Seidner and M. Feder. Noise amplification of periodic nonuniform sampling. *IEEE Trans. on Signal Processing*, 48-1:275–277, Jan. 2000.

[51] S. Shechtman. Very low bit-rate speech coding based on temporal decomposition. Master's thesis, Faculty of EE, Technion, 2004.

[52] S. Shechtman and D. Malah. Efficient sub-optimal temporal decomposition with dynamic weighting of speech signals for coding applications. In *Interspeech 2004 - ICSLP*, Korea, Oct. 2004.

[53] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 36:1445–1453, Sept. 1998.

[54] T. Sikora and B. Makai. Shape-Adaptve DCT for generic coding of video. *IEEE Trans. on Circuits, Systems and Video Technologies*, 5-1:59–62, Feb. 1995.

[55] A. Spira. Image palletization and compression for color-limited displays. Master's thesis, Faculty of EE, Technion, 2000.

[56] Y. Stylianou. Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing*, 9(1):21–29, Jan. 2001.

[57] A.D. Subramaniam. Gaussian mixture models in compression and communication. Master's thesis, University of California, 2003.

[58] A.D. Subramaniam and B.D. Rao. Pdf optimized parametric vector quantization of speech line spectral frequencies. *IEEE Transactions on Speech and Audio Processing*, 11(2):130–142, Mar. 2003.

[59] S. Wang, A. Sekey, and A. Gersho. An objective measure for predicting subjective quality of speech coders. *IEEE Journal on Selected Areas in Communications*, 10(5):819–829, June 1992.

[60] C.S. Xydeas and C. Papanastasiou. Split matrix quantization of LPC parameters. *IEEE Trans. on Speech and Audio Processing*, 7:113–125, Mar. 1999.

[61] K. Zeger and A. Gersho. Pseudo-gray coding. *IEEE Trans. on Communications*, 38(12):2147–2158, Dec. 1990.

צמצום עקבה משמר איכות

של

מסנתזי תדבור משורשר

תמר שהם

צמצום עקבה משמר איכות

של

מסנתזי תדבור משורשר

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת חשמל

תמר שהם

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
אייר התש"ע חיפה מאי 2010

המחקר נעשה בהנחיית פרופ' דוד מלאך בפקולטה להנדסת חשמל

ברצוני להודות מעומק ליבי לפרופ' דוד מלאך על הנחייתו המסורה. היה לי לזכות ללמוד ממנו, ולעונג ליהנות מהכוונתו ותמיכתו.

תודה לצוות המעבדה לעיבוד אותות ותמונות (SIPL), נמרוד פלג, יאיר משה, זיוה אבני ואבי רוזן אשר יחדיו מייצרים סביבת עבודה מפרה ונעימה.

תודות גם לקבוצת טכנולוגיות דיבור במעבדות המחקר של IBM בחיפה. לראש הקבוצה, רון חורי, לסלבה שכטמן שהשקיע רבות, ולכל הקבוצה על תמיכתם והעניין שגילו במחקר.

ובנימה אישית, ברצוני להודות בראש ובראשונה לבעלי שלמה, העזר כנגדי, שסייע וחיזק ולא נתן לי להישבר. תודה מיוחדת לאחותי, אבי, אימי, חמי וחמותי על העזרה והתמיכה המתמדת. ואחרונים חביבים, ילדי האהובים, עשהאל-נריה, נעמה-שרה ואליסף-בניה, תודה על הסבלנות, הפרגון, ובעיקר על התוכן שיצקתם לחיי.

תקציר

צמצום עקבה (footprint) למסנתזי תדבור משורשר (Concatenative Text-To-Speech Syhnthesizers), לצורך הטמעתם במגוון מערכות קצה, מהווה אתגר מתמשך. במסנתזים אלו נוצר הדיבור ע"י שרשור מקטעי דיבור קצרים מאוד. מקטעי הדיבור שמורים בעץ אקוסטי, כאשר כל עלה בעץ משויך לתת פונמה מסוימת בהקשר מסוים. בסיס הנתונים האוגר את אוסף העלים האקוסטי הוא התורם המרכזי לעקבה של המערכת. בכדי לצמצם את העקבה, דהיינו, צריכת הזיכרון של המערכת, נרצה לבצע דחיסה נוספת למקטעי הדיבור, על גבי הדחיסה שבוצעה במעבר ממקטעי הדיבור לפרמטרים המייצגים אותם. אנו נרצה שהדחיסה הנוספת תבוצע כך שלא נפגע באיכות התפיסתית של הדיבור המסונתז. נחפש אלגוריתמי דחיסה שאינם מצומדים לבעיה שלפנינו כך שניתן יהיה לעשות בהם שימוש לצורך דחיסה של בסיסי נתונים אחרים, בעלי מאפיינים דומים לבסיס הנתונים שלנו. היות וצמצום העקבה נועד, לרוב, לאפשר שימוש במסנתז במערכות דלות משאבים, נדרוש גם להגביל את סיבוכיות הפענוח של אלגוריתמי הדחיסה.

בעבודה זו אנו מתמקדים בדחיסת הפרמטרים המייצגים את האמפליטודה של המעטפת הספקטראלית בכל מסגרת דיבור, היות והם בעלי העקבה הגדולה ביותר במערכת. במערכת איתה עבדנו מקטעי הדיבור מורכבים ממסגרת דיבור אחת או יותר, (בממוצע 2, אך ישנם גם מקטעים המורכבים מ-35 מסגרות). כל מסגרת דיבור מיוצגת ע"י מודל ספקטראלי המכילה 32 פרמטרי אמפליטודה. בין חמישה לעשרה מקטעים כאלו שמורים בכל עלה אקוסטי. לצורך דחיסה נרצה להוריד את היתירות הקיימת בין מסגרות דיבור ולצורך כך נבצע דחיסה של העלה האקוסטי כיחידה. אנו מציעים שתי שיטות לדחיסת העלים האקוסטיים, אשר יאפשרו צמצום העקבה ללא פגיעה באיכות הדיבור המסונתז. בנוסף נציע אלגוריתם סידור שיאפשר סידור מקטעי הדיבור בעלה האקוסטי לפני ביצוע הדחיסה, בניסיון לשפר את הביצועים הכוללים.

אלגוריתם הדחיסה הראשון מבוסס על Temporal Decomposition (TD). זו משפחת גישות המציעה להתאים מודל להתפתחות הזמנית של אוסף פרמטרים וע"י כך להשיג דחיסה. התבססנו על גישת TD סקלרית, שמציעה לעקוב אחרי איבר מסוים בווקטור לאורך N מסגרות, ולייצגו ע"י פולינום מסדר P. היות ומקדמי הפולינום רגישים לקוונטיזציה מתבצעת דגימה של הפולינום ב-(P+1) נקודות, המאפשרות את שחזורו. אנחנו מציעים הרחבה לאלגוריתם זה לעבודה עם וקטורים. בשיטת דחיסה זו מתקבלים P+1 וקטורים מייצגים (במימד 32 כל אחד) עבור כל מקטע באורך N, אשר בשילוב עם מידע על אורך המקטע וסדר הפולינום שנבחרו מהווים את המידע הדחוס. היות והוקטורים נמצאים באותו מרחב כמו וקטורי המערכת המקורית, ניתן להשתמש בקוונטייזר הקיים לצורך דחיסתם. היתרון של גישה זו על פני גישות TD וקטוריות קיימות נעוץ בכך שבגישה המוצעת פונקציות האינטרפולציה משתנות לאורך הוקטור (פולינומים שונים מסדר P), וכך מתאפשר מידול טוב יותר של וקטורים בהם לאלמנטים שונים ישנה התפתחות זמנית שונה. בנוסף, במקום לעבוד עם מספרי מסגרות וסדרי פולינום קבועים כפי שמוצע במאמר עליו התבססנו, אנחנו מציעים לחפש אדפטיבית ערכי N ו-P אופטימאליים, ע"י שימוש במבנה סבכה (trellis) מוכלל. עבור כל עלה אקוסטי, מוצאים חלוקה לסגמנטים וסדרי פולינומים אופטימאליים תחת אילוצי עיוות וסיבוכיות. בעבודה מוצגות מספר פונקציות עיוות ונמצא שהגבלת השגיאה הריבועית הממוצעת בין הפרמטרים המקוריים והמשוחזרים (ללא שגיאת הקוונטיזציה) המחושב על פני כל ציר התדר, מביא לתוצאות הטובות ביותר. כמו כן נבחנו נקודות עבודה שונות - הגבלות שונות של סדרי הפולינומים המאופשרים (לצורך הגבלת סיבוכיות מפענח), שימוש בגבולות מקטעי הדיבור לסגמנטציה וכיו"ב. לצורך השגת יחס הדחיסה הרצוי, אנו מציעים אלגוריתם איטרטיבי אשר בהינתן קצב מטרה מביא לקצב זה עם מינימום עיוות במובן של minmax.

אלגוריתם הדחיסה השני שמוצע בעבודה זו מבוסס על Shape Adaptive Discrete Cosine Transform (SADCT). התמרה זו הינה הרחבה של ה-DCT המוכר, עבור מידע בעל מתאר שרירותי ולא מלבני. ההרחבה של ה- SADCT לתלת מימד מאפשרת הפעלתו על הפרמטרים של העלה האקוסטי. היות וההתמרה מגבירה את ריכוזיות האנרגיה היא מאפשרת דחיסה ללא פגיעה באיכות. לשם ביצוע הדחיסה יש לתכנן קוונטייזרים מתאימים. אנו מציעים גישה מתודית להקצאת סיביות ופיצול הווקטורים לקבוצות שונות, לצורך תכנון הקוונטייזרים. האלגוריתם המוצע מופעל פעמיים עם שינויים קלים. בפעם הראשונה, מחולקים הוקטורים למספר קבוצות שונות, כאשר הוקטורים בכל אחת מהקבוצות מקבלים הקצאת סיביות שונה. בשלב שני, עבור הוקטורים בכל

קבוצה מתבצעת חלוקה לתתי וקטורים עם הקצאות מתאימות. לאחר שלב זה ניתן לתכנן קוונטייזר וקטורי לכל תת-וקטור בכל קבוצה תוך שימוש בשיטות מוכרות כדוגמת ה-LBG.

שני האלגוריתמים המוצעים פועלים על העלה האקוסטי, המכיל מספר מקטעי דיבור אשר מסודרים בסדר שרירותי. מטרת האלגוריתם לסידור הסגמנטים הינה למצוא סידור שיביא למינימום פונקצית מחיר המתארת את טיב הסידור עבור אלגוריתם הדחיסה המיועד. בדחיסה המבוססת על TD מתבצע שרשור של מסגרות הדיבור בעלה לפני דחיסתם, לכן נרצה לסדרם באופן שיגרום לכך שתוצאת השרשור תהיה חלקה ככל האפשר. עבור אלגוריתם הדחיסה המבוסס על SADCT, הסידור ישפיע על תוצאות ההתמרה לאורך עמודות, ונרצה להביא לריכוזיות אנרגיה גבוהה ככל האפשר. בחנו מספר שיטות מתאימות והצענו שיטה המבוססת על אלגוריתם מטרופוליס ומשלבת רעיונות של Binary Switching Algorithm (BSA) ושל חישול מדומה (Simulated Annealing) לקבלת אלגוריתם שאנו מכנים MBOrd. באלגוריתם זה מבצעים שינוי אקראי בסידור הקיים. אם הסידור החדש הוא טוב יותר השינוי נשמר, אך בהסתברות מסוימת, התלויה בפרמטר טמפרטורה ובגודל העלייה בפונקצית המחיר, גם שינוי לרעה יישמר. בניגוד לחישול מדומה אנו לא מבצעים תהליך קירור ולכן אין התכנסות לסידור הרצוי. במקום זאת אנו לוקחים את הסידור בעל המחיר הנמוך ביותר מכל אלו שנבחנו, וע"י כך משיגים שיפור בביצועים לעומת ה-BSA.

מימוש האלגוריתמים המוצעים, בסביבה של מסנתז תדבור משורשר בעל עקבה קטנה שהתקבל ממעבדות המחקר ב-IBM, הראה כי ניתן להקטין את העקבה של פרמטרי האמפליטודה של מסגרות הדיבור פי 2, מבלי לפגוע באיכות התפיסתית של הדיבור המתקבל. האיכות נמדדה ע"י ציון PESQ, תקן של ITU למדידה אובייקטיבית של איכות דיבור (מול דיבור מקור). השתמשנו בנקודת העבודה המיועדת לדיבור רחב סרט המספקת ציונים בתחום 0-5, כאשר איכות של 3.7 ומעלה, נחשבת לרוב לכמעט בלתי ניתנת לאבחנה לעומת המקור. בחנו את האלגוריתמים המוצעים על 10 משפטים, ולא נשמעו הבדלים. ציוני ה-PESQ שהתקבלו היו 3.7 לשיטה המבוססת על TD, ו-3.85 לשיטה המבוססת על SADCT. (לצורך השוואה האיכות המתקבלת ע"י דחיסה פי 2 ע"י דחיסה פשטנית, כלומר סינון, דגימה ואינטרפולציה ליניארית, הינו 2.8). לשיטה הראשונה (TD) ישנו יתרון של גמישות גבוהה יותר בבחירת יחס הדחיסה הרצוי, כאשר בשיטה השנייה (SADCT) שינוי יחס הדחיסה מצריך תכנון מחדש של הקוונטייזר. הוספת האלגוריתם לסידור הסגמנטים כשלב מקדים כמעט לא השפיעה על התוצאות הממוצעות, אך שיפרה בכ-0.1 את ציון ה-PESQ הנמוך ביותר במשפטים שנבדקו.

האלגוריתמים המוצעים עומדים בדרישות התכנון. הם מאפשרים צמצום העקבה של מסנתז תדבור משורשר, ע"י דחיסה פי 2 של פרמטרי האמפליטודה המייצגים את הספקטרום של מסגרות הדיבור השמורות, המהוות את רוב העקבה של המערכת. כמו-כן הראינו שאיכות הדיבור המסונתז נשמרה. נדרש מחקר נוסף לבחון את הרחבת האלגוריתם למודלים ספקטראליים חליפיים וכן להפעלתו על פרמטרי פאזה. היות שהאלגוריתמים הם כלליים, ניתן ליישמם במגוון בעיות דחיסה מחדש של בסיסי נתונים המורכבים מיחידות דו- או תלת- ממדיות בעלות יתירות, למשל: בסיס נתונים לשפת הסימנים, בסיסי נתונים המשמשים לסיווג תמונות, בסיסי נתונים גדולים המשמשים למערכות להמלצות צפייה אישיות וכיו"ב.