Quality Preserving Compression of a Concatenative Text-To-Speech Acoustic Database

Tamar Shoham, David Malah, and Slava Shechtman

CCIT Report #799
December 2011

CCIT Report #799 December 2011

Quality Preserving Compression of a Concatenative Text-To-Speech Acoustic Database

Tamar Shoham, David Malah, and Slava Shechtman

Abstract

A Concatenative Text-To-Speech (CTTS) synthesizer requires a large acoustic database for high quality speech synthesis. This database consists of many acoustic leaves, each containing a number of short, compressed, speech segments. In this paper we propose two algorithms for re-compression of the acoustic database, by re-compressing the data in each acoustic leaf, without compromising the perceptual quality of the obtained synthesized speech. This is achieved by exploiting the redundancy between speech frames and speech segments in the acoustic leaf. The first approach is based on a vector polynomial Temporal Decomposition. The second is based on 3D Shape-Adaptive DCT, followed by optimized quantization. In addition we propose a segment ordering algorithm in an attempt to improve overall performance. The developed algorithms are generic and may be applied to a variety of compression challenges. When applied to compressed spectral amplitude parameters of a specific IBM small footprint CTTS database, we obtain a re-compression factor of 2 without any perceived degradation in the quality of the synthesized speech.

Index Terms

Concatenative Text-To-Speech (CTTS), Acoustic Leaf Compression, Temporal Decomposition (TD), Discrete-Cosine-Transform (DCT), Shape-Adaptive-DCT (SADCT).

Tamar Shoham and David Malah are with the Department of Electrical Engineering, Technion - Israel. Institute of Technology, Haifa 32000, Israel. Slava Shechtman is with the IBM Haifa Research Labs. [e-mails: tshoham3@gmail.com; malah@ee.technion.ac.il; SLAVA@il.ibm.com]

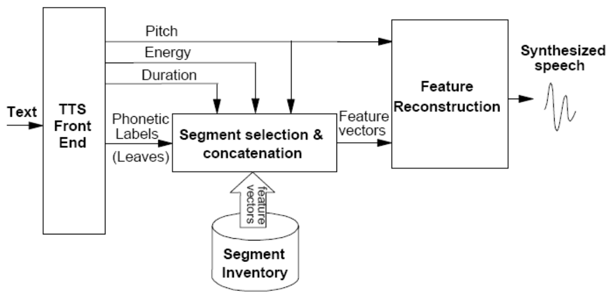

Fig. 1. Illustration of a Concatenative Text-To-Speech synthesizer

I. Introduction

A standard Concatenative TTS (CTTS) system is illustrated in Figure 1. To obtain high quality speech a large number of short speech segments must be stored in the segment inventory. These segments are organized in acoustic leaves, where all speech segments in a leaf belong to the same sub-phoneme in the same context. In small footprint CTTS systems, i.e., CTTS systems with low memory consumption, each speech segment is usually represented by a parametric model. Specifically, in IBM's system [1], on which the proposed algorithms were evaluated, each acoustic leaf contains 5-10 speech segments, with each speech segment consisting of 1-35 frames, with a median number of frames of 2. As detailed in [1], for each frame of 10msec duration, a vector of 32 amplitude parameters and a variable length vector of phase parameters, represent the spectral envelope of the frame, sampled in Mel-scale to match perceptual characteristics of the auditory system. During speech synthesis an appropriate acoustic leaf is selected for each sub-phoneme. A speech segment within a leaf is selected based on spectral and pitch smoothness criteria, combined with target prosody similarity metrics. The stored model parameters are used to synthesize the speech, possibly after modifications such as changing pitch or duration. The synthesized speech segment is then concatenated to the previously synthesized speech, as detailed in [2]. We wish to further compress the parameters stored in these acoustic leaves, thus further reducing the CTTS footprint, without perceptually reducing the obtained synthesized speech quality. We refer to the compression algorithm also as *recompression*, to indicate that the compression is performed on the acoustic leaf data which has already undergone a compression stage, rather than operating on raw data, which is sometimes, as in our case, no longer available.

A similar issue was addressed in [3], where Kain and Santen propose to compress acoustic inventories by performing asynchronous interpolation of templates representing the beginning and end of each diphone (an acoustic unit comprising of two phonemes). While this approach achieves a high compression ratio it comes at the price of poor perceptual quality and low flexibility.

In the system we used for algorithm validation the amplitude parameters footprint is much larger than that of the phase parameters, therefore we focused on the amplitude parameters. Thus, we are faced with the following recompression challenge: Further compression of amplitude spectral parameters arranged in acoustic leaves comprising of 3D arrays of varying dimensions. An example of this acoustic leaf structure is shown in Fig. 2.

The algorithms were designed without tailoring to the specifics of the setup used for their evaluation. Thus, the obtained algorithms are generic and may be applied to a wide range of recompression challenges. Examples include recompression of sign language databases, compression of databases used for image classification in Bag-Of-Words methods, or any other database that consists of many data 'buckets' containing data organized in 2D or 3D structures that exhibit redundancy. An additional algorithmic design requirement was low complexity decoding, since decompression, or in our case the speech synthesis, is generally performed on end devices with limited resources.

In this paper, we propose two compression approaches. In the first, which we have briefly described in [4], we use a Vectorial form of Polynomial Temporal Decomposition (TD) to obtain compression. In this approach the speech segments are concatenated so that the compression is applied to a set of 2D data units. In the second approach, we propose to apply the 3D Shape-Adaptive Discrete Cosine Transform (3D SADCT) to each acoustic leaf. By working on the original 3D structure, 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 hence expected to be similar. This incurs the price that reconstruction of a single speech segment requires performing the inverse SADCT for the entire leaf. In addition, we will also present a segment ordering algorithm which may be applied prior to compression in an attempt to improve overall performance. The proposed algorithms were evaluated by applying them to the amplitude parameters of an IBM CTTS voice (US English, female). In our evaluations we focused on the amplitude parameters, whose original footprint in the system we used is 5.7 MB. The phase parameters, with a footprint of only 1.6 MB, remain unaltered. We will show that we achieve a recompression factor of 2 of the amplitude parameters, in both approaches, without compromising perceptual quality relative to the original synthesized speech of [1], as measured by PESQ [5] - a differential perceptual quality measure, and confirmed in comparative listening tests.

The paper is structured as follows. In Section II we present the polynomial Temporal Decomposition based compression algorithm. In Section III we present the 3D SADCT based compression including a

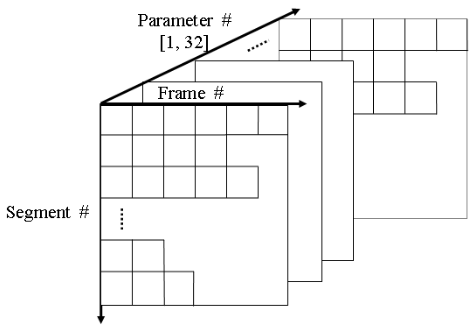

Fig. 2. 3D structure of an acoustic leaf. The vertical dimension corresponds to the segment number, the horizontal to the frame number within the segment, and the 'depth' dimension corresponds to the vector of parameters that represent the spectrum of each frame.

novel approach to a joint bit-allocation and splitting scheme for split Vector Quantization (split-VQ) design. The segment reordering algorithm is described in Section IV. In Section V we present the experimental setups and results of a quality evaluation of the proposed algorithms, followed by the conclusion in Section VI.

II. Compression using vector Polynomial Temporal Decomposition

In this section we describe one of the proposed algorithms for acoustic leaf (re)compression - Vector Polynomial 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 of either a scalar parameter (scalar TD) or a parameter vector (vector TD) over time. The parameter trajectory is represented with a reduced order model, thus achieving compression. The Vector TD method was first introduced in [6] and has been used for low rate speech coding in many previous works including [7], [8], [9], [10], [11] and [12]. In these methods the original parameter vectors are represented by a set of event times or locations, event vectors and interpolation functions. These algorithms generally require at least 10 consecutive frames per segment to be effective, which is not the case for our database which mainly consists of very short speech segments. Another limitation of these vector TD approaches is that the interpolation functions are identical 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. A DCT based Scalar TD approach proposed in [13], was shown there to be beneficial mainly for the low harmonics of voiced speech, of 10-20 consecutive speech frames. This approach, which includes adaptive

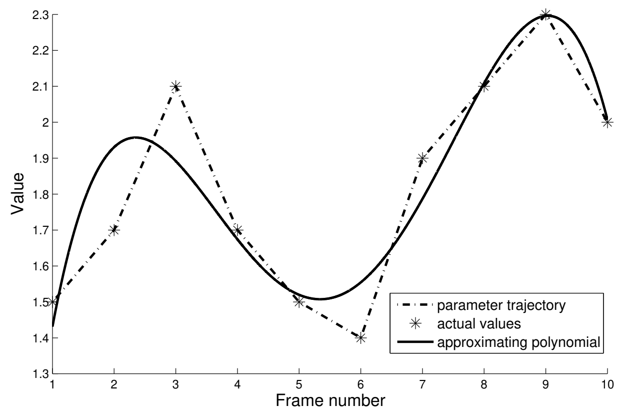

Fig. 3. Illustration of **scalar** Polynomial-based TD for a single TD segment with N=10 frames and a polynomial order P=4. The asterisks mark the actual parameter values at each of the 10 frames, the dashed line is a piecewise linear approximation of their trajectory and the solid line shows the approximating polynomial trajectory.

model order selection, has been extended to cepstrum parameters and LSF parameters in [14] and [15] respectively. In all these works the underlying speech is assumed to be continuous, and categorized as voiced or unvoiced, rather than consisting of short disconnected segments, containing mixed voiced and unvoiced frames, as in our case.

In [16] and [17] Dusan *et* al. present an alternative scalar TD approach, polynomial scalar TD. In this approach, the trajectory of each scalar (vector element) value along the N frame segment is 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. 3. Since polynomial coefficients are sensitive to quantization, the polynomial is sampled at $P+1$ points, and the obtained values, that lie in the original feature space, are coded and transmitted. For synthesis, the P+1 features are used to uniquely find the P^{th} order polynomial, which is re-sampled at all the original N points. A similar approach was presented by Nygaard *et* al. in [18], [19], for the compression of ECG signals. These works deal with a scalar setup only, using a signal which is approximately piecewise linear, and propose to use either linear interpolation (in [19]) or fitting with second order polynomials (in [18]). It is therefore, in a way, a subset of the polynomial TD approach, though a different application of it.

A. Proposed Vector Polynomial TD Algorithm

When attempting to apply Temporal Decomposition to CTTS speech segments, we must keep in mind that the data consists mainly of very short segments, often including transitions between voiced and unvoiced frames. 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 readily adapts to very short segments by decreasing polynomial orders.

We wish to benefit also from the advantages of vector TD, as it 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 shall also incorporate adaptive model order selection, as proposed by Girin *et* al. in [13], [14] and [15] for their DCT based approach, in order to obtain a consistent error, rather than using a constant model order as in the scalar polynomial TD proposed in [17].

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 perform 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 (interpolation) 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 address the selection of N, the number of frames in the TD segment, and the polynomial orders, P, which are both adaptive, as this constitutes the central challenge in applying the vector polynomial TD algorithm to the CTTS acoustic database.

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 Section IV. 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 vector 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 limit 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 calculated, per-frame, distortion value, D_f, (discussed in V-A) measured between each pair of stored (quantized) and reconstructed speech frames, the global distortion D_g is defined as:

$$D_g = \max_{leaves} \left\{ \max_{TDsegments} \left[\max_{frames}(D_f) \right] \right\} \tag{1}$$

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

$$D_g^* = \min(D_g) \;\; s.t. \;\; R(D_g^*) \leq R_t \tag{2}$$

Since the rate is a monotonic non-increasing function of the distortion, we can adopt an iterative solution, similar to the one proposed in [20], Ch. 4. 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 'pseudo-code' in Algorithm 1. Note that the proposed algorithm allows for automatic adaptation to any target rate or compression ratio.

The need for step 2 in Algorithm 1 stems from the fact that on one hand, we do not wish to initialize our interval with a value of D_U (highest distortion in examined interval) that is too high and will incur unnecessary iterations. For instance we could set the initial distortion to its maximum by transmitting no data, but then we would need 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 2 does.

Note that we perform simple bi-section, as proposed in [20]. We also evaluated the option of performing a weighted bi-section search, 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, for each TD segment, the bits incurred by coding the parameters, and also the required overhead bits. We use the IBM split-VQ quantization

Algorithm 1 Vector Polynomial TD: Algorithm for converging to target bit-rate with minmax distortion

```
1) Initializations:
    a) Set target rate, $R_t$ and tolerance range.
    b) $D_L$=0, $D_U$=default maximum distortion value.
2) Search-range determination:
    a) Perform polynomial TD as described in II-B for each leaf in the database, limiting maximum allowed distortion to $D_U$, and set $rate$ to the obtained overall rate.
    b) Verify $rate \leq R_t$. If not, double $D_U$ value and GOTO 2a.
3) Calculate value for $D_g$: $D_g = \frac{1}{2}(D_L + D_U)$.
4) Perform polynomial TD as described in II-B for each leaf in the database, limiting maximum allowed distortion to $D_g$, and set $rate$ to the obtained overall rate.
5) IF $rate$ is within the tolerance range of the target rate, $R_t$,: GOTO 7.
6) IF $(rate < R_t)$: $D_U = D_g$, ELSE: $D_L = D_g$ ; GOTO 3.
7) $D_g^* = D_g$ ; END.
```

(described in [21]) taken from the test CTTS system, which uses 86 bits to represent each amplitude parameter vector of 32 coefficients. Therefore, for each TD segment of length N frames, 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.

B. Jointly Optimal Segmentation and Polynomial Order Selection

We cannot presume that smoothness assumptions will hold for the entire super-segment, and as we also wish to use low order polynomials, to limit complexity and increase stability, 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 could concatenate speech segments that join smoothly, while we 'break-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 [22] and [23]. In these works Prandoni and Vetterli propose joint segmentation of speech samples with selection of an 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 and Vetterli, 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. The cost function used is described in Section V-A. Then we "flatten" the structure as follows: At each trellis point, defined by i, j, the value of $k(i, j)$ is set to $k^*(i, j)$ - 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 obtain a 2D generalized structure, as shown in Fig. 4 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 state $S_{i,j}$ that has the lowest accumulated cost, and mark it with a star. The accumulated cost calculated for $S^*_{i,j}$ is added to the cost of any path that starts at the following 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. 4, for a five column trellis.

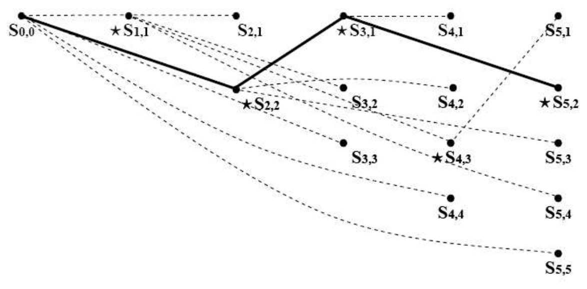

Fig. 4. 2D structure for optimal segmentation. Each point in this generalized trellis may connect to any point in the columns to its left. Dotted lines illustrate all the paths considered, solid line shows optimal path.

III. Compression using 3D Shape-Adaptive Discrete Cosine Transform

A. 3D SADCT Overview

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. Speech coding using 2D-DCT has been proposed in the past, for instance in [24] and [25]. Due to the variability in the segment lengths, in order to apply DCT to our 3D structure (Fig. 2), we must use a variation of the DCT, known as Shape-Adaptive DCT (SADCT) [26], which we now describe. The regular 3D DCT is defined by the following equations:

$$F(u,v,w) = \sqrt{\frac{8}{NMP}} C_u C_v C_w \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} \sum_{p=0}^{P-1} f(n,m,p) \cdot$$
$$\cos\left[\frac{(2n+1)u\pi}{2N}\right] \cos\left[\frac{(2m+1)v\pi}{2M}\right] \cos\left[\frac{(2p+1)w\pi}{2P}\right] \tag{3}$$

Where,

$$C_x = \begin{cases} \frac{1}{\sqrt{2}}, & \text{x=0;} \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

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

Note that the 3D DCT transform is separable, i.e., 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 a 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 [26], was originally developed for efficient coding of arbitrary shaped image segments in video. In [26] 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 [27] Markman and Malah extended this concept to 3D SADCT and applied it to hyperspectral image coding.

SADCT exploits the separability of the regular DCT transform, and performs the DCT transforms in each dimension consecutively. For 2D SADCT, as defined in [26], first a varying dimension 1D vertical transform is applied to each column. After these are completed, a varying dimension 1D horizontal transform is applied to the obtained coefficients of each row. For 3D SADCT this is then repeated for the 3rd dimension. The 3D SADCT applies to general contours, however our case is simpler, as in each acoustic leaf the variability 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 parameter vector length, which defines the third dimension, is fixed.

We apply the 3D SADCT to each acoustic leaf as illustrated in Fig. 5, by performing required shifts (to eliminate "holes" when segments are not stored according to length) 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.

B. 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 redundancies 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 property of energy compaction is illustrated in Fig. 6, for 300 sample leafs, by displaying the ratio between the number of coefficients we must retain in order to represent 95% of the total energy, and the original number of features in the leaf. This is shown for the original features and for DCT and

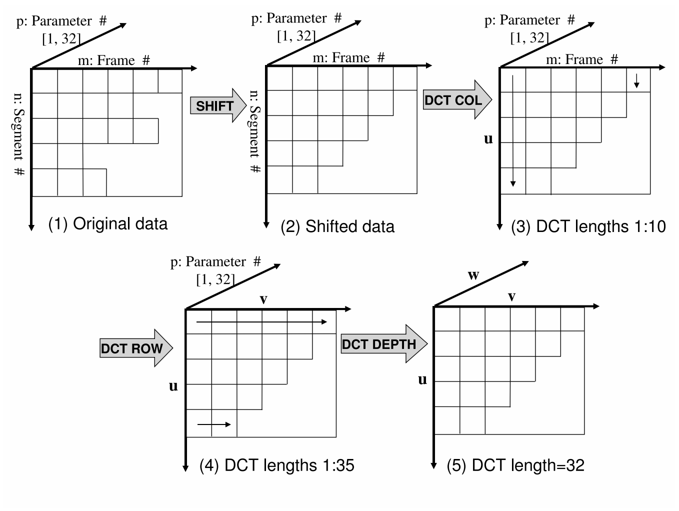

Fig. 5. 3D SADCT applied to a sample acoustic leaf. Shifting is required only along the vertical (segment number) dimension. The DCT Basis function lengths are adapted to the length of each column (segment number within leaf) or row (frame number within segment), and are fixed at 32 for the depth (feature vector) dimension.

SADCT coefficients. The DCT is performed on the bounding cube, with zero padding. From Fig. 6, it is clear the SADCT substantially and consistently outperforms the other two representations in terms of energy compaction. 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 95% of the total energy. It is also apparent that when the features require keeping of relatively more values (top line goes up), the SADCT can manage with relatively less values (bottom line in the right graph goes down).

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.

C. 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. As opposed to the case of polynomial based TD, where we obtain vectors for coding which can reuse the existing quantizer, in the case of 3D SADCT the parameters for coding lie in a different space, which requires design of

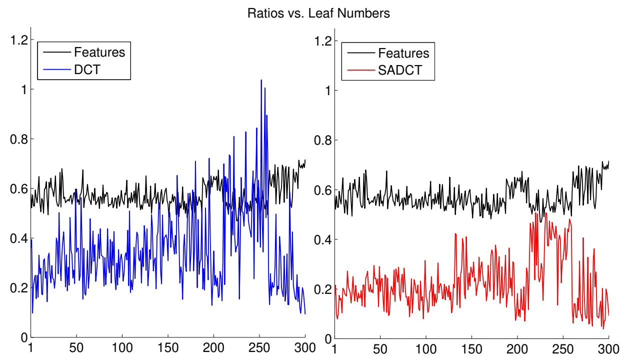

Fig. 6. Energy compaction of original features, DCT and SADCT coefficients, shown via the ratio between the number of coefficients that hold 95% of the total energy and the total number of elements in the leaf, for a set of 300 leaves (horizontal axis). The top (black) line corresponds to the original features (avg ratio: 0.57) The bottom (blue) line in the left graph corresponds to DCT (0.33), the bottom (red) line in the right graph to the SADCT (0.22).

an appropriate quantizer.

We evaluated the algorithm performance for a target recompression factor of 2 compared to the IBM system we used, which applies an 86 bit vector quantizer to each 32 element vector. 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 also pursue a Vector Quantization (VQ) approach. Previous works in the area propose a variety of approaches. In [28], a framework for bit allocation in a multi quantizer setup is proposed. Assuming a pre-known split of the data between quantizers, as in sub-band coding for instance, 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, and also since we do not have a closed form distortion function, required for this algorithm, as we wish to minimize perceptual error in the reconstructed speech. In [29] an optimum transform domain split Vector Quantization (VQ) method is developed, and both an optimal approach and a faster, sub-optimal, approach to finding the vector split points are presented. However, the algorithm is not easily adapted to the varying dimensionality of our data. Also, the source data distribution is assumed to be Gaussian, whereas our DCT coefficients are closer to a Laplacian distribution. In [27], 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 [30], 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:

- Finding a method for splitting the 3D structures into ”manageable” sub-units.
- Performing allocation of bits to the various units.
- Designing a Vector Quantizer (VQ) 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 VQ design.

1) Joint Bit Allocation and Splitting Algorithm: The data we wish to code is a set of 3D coefficient arrays, one for each of the I acoustic leaves, obtained after performing 3D SADCT on each leaf. The i^{th} leaf is represented by a set indices u, v, w, where u corresponds to the segment index, $u = 1, ..., U$ and U is the number of segments in the leaf, v corresponds to the frame index within the segment (after SADCT shifting), $v = 1, ..., V_u$ where V_u corresponds to the original segment lengths, and w corresponds to the parameter vector index, $w = 1, ..., W$, where W is the number of parameters per frame, which in our setup is constant at 32. An example of such a 3D array is shown in Fig. 5.

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.

STAGE I

In the first stage we begin with a 2D array with indices u, v. Each 32 element vector in each acoustic leaf is associated with the appropriate index combination $\{u_x, v_y\}$. For each $\{u_x, v_y\}$ the corresponding set of vectors is the collection of all the vectors in the acoustic databases with the indices $\{u = u_x; v = v_y\}$. We wish to divide these $\{u, v\}$ index pairs into M groups, so that each group comprises a specific set of $\{u, v\}$ pairs, and also determine an appropriate bit allocation for the groups of corresponding vectors. We found that $M = 5$, i.e., one DC group containing the $\{u = 1; v = 1\}$ vector from each acoustic leaf, in addition to 4 AC groups, provided a good working point. The DC group receives an empirical allocation of 50 bits, for each 32 element vector.

In order to use known bit allocation approaches, such as the one described in [31] Ch. 8, 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 $\{u = 10, v = 4\}$.

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 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 for a given target bit-rate (43 bits), proposes an average bit allocation value, per vector: $(R_{avg})_t$, 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 an 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 the following equation (based on [31]):

$$\begin{aligned}(R^{opt}_{u,v})_t = (R_{avg})_t + \frac{1}{2}\log_2 \frac{\hat{\sigma}^2_{u,v}}{\left(\prod_{p,q}\hat{\sigma}^2_{p,q}\right)^{\frac{1}{Q}}} + ...\\ ... + \frac{1}{2}\log_2 \frac{W^2_{u,v}}{\left(\prod_{p,q}W^2_{p,q}\right)^{\frac{1}{Q}}}\end{aligned} \qquad (5)$$

Where: $W_{u,v} = \frac{1}{u*v}$ are weights, selected this way to 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, with a simple clustering algorithm using 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, typically after about a dozen iterations.

This algorithm results in M groups of vectors, and a target bit allocation for each group, $R(m)$, $m = 2, ..., M$ ($R(1) = 50$). The obtained bit allocation for our database is provided in Fig. 8 in Section V-B.

STAGE II

Now, for each group, we require a splitting scheme and bit allocation along the w dimension, where $w = 1, ..., 32$, using the same underlying algorithm. Since the data is now 1D and with constant length, the algorithm is more straightforward. Again, we begin by allocating bits to the first element (DC). The DC element F(1,1,1), i.e, DC in all dimensions, receives an empirical bit allocation of $R_{DC}(1) = 8$ bits. The bit allocation for the DC element of the m^{th} group 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, i.e., $m = 1, ..., M$. For the m^{th} group, the standard deviation of each vector element is calculated, using only the elements of the vectors 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 a 1D bit allocation formulation, along the lines of (5), with $Q = 31$, $R_{avg}(m) = \frac{R(m) - R_{DC}(m)}{31}$, $W_u = \frac{1}{u}$ and $u = 2, .., 32$.

These 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 codebooks so that their footprint is no larger than the footprint of the codebooks in the IBM system we used (about 2kB). 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, in each group, is the rounded sum of the allocations of its elements. The results of this process for our database are provided in Fig. 9 in Section V-B.

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 2-D data with variable lengths, i.e., where each data unit, such as our acoustic leaf, has a different number of rows and a different number of columns per row. It may thus be applied to a variety of quantizer 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.

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 [32], where using an iterative approach we find the best clustering of the sub-vectors to be coded - assuming known cluster centroids, and then find the updated centroids for the resulting clusters. In each iteration the number of clusters is doubled until we reach the target number of clusters according to the bit allocation. Thus, for each sub-vector of each group, we use as a training set all the associated sub vectors, and create the corresponding codebook. As mentioned above, we limit the size of the largest cluster, (longest sub-vector), to no more than 8 elements, in order to verify that the collective footprint of the proposed codebooks do not exceed that of the IBM codebooks.

IV. Metropolis Based Ordering of segments

In order to improve recompression performance, we wish to sort the speech segments in each acoustic leaf in the ”best” order. Our aim is to order the segments so that we maximize smoothness of the *super-segment* for the vector polynomial TD approach, or, compact as much of the leaf energy as possible into G_2 (the lowest non-DC frequency group) for the 3D-SADCT based approach.

This ordering problem is a classic combinatorial optimization problem. 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’, assuming a known cost function. While this is possible for the smaller leaves, containing seven segments or less, it becomes too cumbersome for the larger leaves. Two previously proposed approaches that can be applied here are the Binary Switching Algorithm (BSA) originally proposed in [33] and Simulated annealing (SA) proposed in [34], which is performed using the Metropolis algorithm [35]. In BSA, random order changes are made and then kept if they cause a decrease in the obtained cost function. In SA a cooling schedule is applied. Again, random order changes are performed and the changes that cause a reduction in the cost function are kept, but, also changes that cause the cost function to increase are kept at a certain probability, which is reduced as the temperature is lowered. Since our ordering algorithm is applied offline, and we can ’keep’ the order that we found to have the lowest cost, it is sufficient that we ’visit’ the best order rather than converge to it. Therefore, the cooling process of the SA is not required here, however its pseudo-random behavior, which allows escaping local minimums, yields better results than the BSA algorithm. We therefore propose an algorithm that combines the BSA and SA properties. Algorithm 2 provides the outline of the proposed algorithm, given an initial order (the original order in the acoustic leaf), a defined cost function to be minimized and a ”Perturbation Generator” (a unit which enables a random segment ”move”).

The termination condition in our case is simply defined by reaching the maximum number of iterations

Algorithm 2 Metropolis Based Ordering: Algorithm Outline

```
1) Initialize: set initial order, and set T to a 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 and the temperature keep "bad moves").
6) If termination condition is reached: STOP, else: GOTO 2.
```

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. 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 simulation result, i.e., the lowest cost on average over 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. In Section V-C ((6), (7)), we will present the cost functions used for each of the two recompression schemes.

To summarize, we have proposed a segment reordering algorithm which may be applied prior to the proposed compression approaches. For leaves with seven segments or less an exhaustive search over all possible orders is performed to find the order with the lowest cost. For larger leaves, the proposed

Metropolis based algorithm is applied. As we will show in Sec. V, the reordering algorithm slightly improved the obtained speech quality as measured by PESQ.

V. Experimental Results

The proposed algorithms were applied to the spectral model amplitude parameters of an IBM small footprint CTTS acoustic leaf database, comprising of 23,263 leaves. The results were evaluated on 10 sample sentences (created using 1661 leaves). The quality of the reconstructed speech, created using the recompressed leaves, was evaluated by calculating the PESQ: Perceptual Evaluation of Speech Quality score ([5]), using the original CTTS output of the system described in [1] as a reference signal. Using this environment, the performance was evaluated for recompression of the amplitude parameters by a factor of 2, for all algorithms and setups.

A. Polynomial TD based recompression

In this section we will present the experimental results for recompression of a CTTS acoustic database using the proposed Vector Polynomial TD Algorithm. We will first address the issue of selecting a cost function to use in the optimization process, then describe some variants or different setups of the algorithm that we examined, followed by the obtained perceptual quality, as measured by PESQ, for recompression by a factor of x2.

Cost Function Selection

In the polynomial TD rate-distortion optimization process described in II-A, we must constantly evaluate D_f, the distortion between the speech frame synthesized using the original, stored segments, and the speech frame synthesized using the reconstructed segments in a specific TD 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. We evaluated distortion functions based on MSE and on Log Spectral Distortion (LSD).

The LSD measure, as defined in [36], is considered a reliable estimator of perceived speech quality. It is calculated based on the spectra of the original and reconstructed signals using a logarithmic frequency scale. Since the original signal spectrum is represented by the set of amplitude parameters we are attempting to compress, we may calculate the LSD directly in the parameter space, as described in Appendix B of [21]. The proposed measure can be easily computed during the optimization process

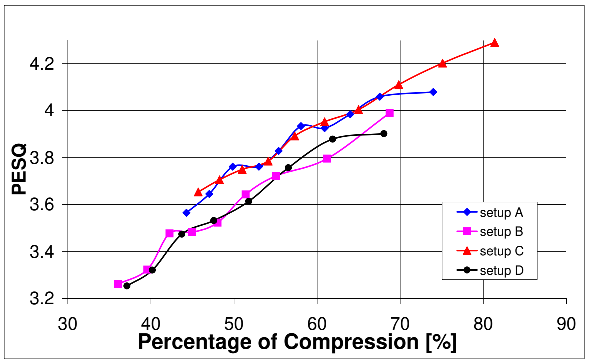

Fig. 7. PESQ vs. percent of compression for various cost functions all using full frequency band; setup A: LSD maximum; setup B: LSD mean; setup C: MSE maximum; setup D: MSE mean

thus striving to minimize of the actual LSD rather than just an Euclidean distance measure between the parameter vectors, in hope of improving overall performance.

We compared the performance when calculating the distortion measure over the entire frequency range used in the speech model: [0-11,050]Hz, and when calculating over an active range of [100-8,000]Hz only. Another parameter in the cost function determination, is whether to take the average distortion over the frames in the TD segment, or the maximum distortion. Thus, for each of the MSE and LSD based cost functions, we have four possible combinations of either full or active frequency range and mean or maximum distortion.

The performance of the cost functions was compared by integrating each possible cost function setting into a simplified TD algorithm, where we simply measured the obtained rates, or extent of compression, and the quality as indicated by the PESQ scores of the reconstructed speech for a number of target distortion values. In addition, in this evaluation phase, the speech was reconstructed from the obtained TD vectors, without quantization. Results for four selected cases, LSD and MSE using the maximum and mean over frames and the full frequency band, are shown in Fig. 7. Using the LSD measure did not improve perceptual performance. This appears to be because the spectral parameters were already obtained with perceptual considerations in mind, as described in [1]. Using only the active frequency range slightly improved the quality, on average, but caused the dependence of quality on the allowed distortion to be less monotone and, therefore, was not adopted. We found that using the **MSE** measure, calculated over the **full frequency** range, and limiting the **maximal distortion** over the TD segment, (Setup C in Fig. 7), provided the best and most consistent results.

EVALUATED SETUPS

The proposed algorithm is to be used for small footprint CTTS synthesizers. Since the host devices often have both CPU and memory constraints, we examined some reduced complexity setups. A number of different algorithm setups were examined, and are now described. Most of these setups aim to minimize complexity, while others also attempt to improve the obtained speech quality.

1) Full Setup: The full setup of the algorithm refers to the algorithm as described in section II, i.e., performing optimal segmentation and polynomial order selection. Polynomial orders are restricted to the range [0,4], to limit the effect of the quantization error, as described in [21]. Maximum TD segment length is 16 frames, as longer segments 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) Reduced Polynomial Order Setup: Since the end-devices performing speech synthesis often have limited CPU power, we may wish to avoid the high complexity incurred by the polynomial fitting required to reconstruct the N data points from the P+1 polynomial samples. Therefore, we evaluated the performance of the algorithm when allowing polynomials of orders 0 and 1 only, which require at most linear interpolation for reconstruction. 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 further 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.

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, the overhead of the 4 bits required to represent the selected TD length is excessive. We therefore evaluated also a setup which limits the maximum TD segment length to 8 frames, thus reducing 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 to the same TD segment. The results for this setup, shown in Table I, are for the case of reduced polynomial order.

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. Furthermore, since in this setup TD segments do not contain frames from more than one speech segment, the reordering algorithm is not relevant. 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 maximum TD segment lengths of 8 and 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 (as analyzed in [21]) and decoding complexity. The overhead size in this setup is adaptive in the range 0-3 bits, and depends on the speech segment length (which is stored as part of the original side information).

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.

RESULTS

The results obtained over the 10 test sentences, for the various polynomial TD algorithm setups, at a recompression factor of x2, are presented in Table I. These setups were described (and numbered) in Section V-A. The PESQ scores were calculated using corresponding speech signals, synthesized according to [1], as the reference signals. The results provided are using the proposed Metropolis Based Ordering algorithm for setups #1,2,3 and 5, with the current IBM Vector Quantization. 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. As shown, this simple approach provides much lower PESQ scores.

Note, that using embedded quantization, i.e., the setup where the distortion is evaluated using values that have undergone quantization, caused the PESQ score to decrease, contrary to expectations. This is due to the fact that the quantization error of the IBM VQ we used is perceptually shaped, whereas the proposed *minmax* optimization is not. Since the quantizer introduces quite large errors in frequency bands that are perceptually less important, the value of our target distortion is increased, which may allow for larger distortions in the frequencies that have higher perceptual importance. We therefore evaluated an optimization that applies a perceptually weighted MSE, which reduced the degradation in performance caused by the embedded quantization setup, but still did not improve overall performance. This is due to the fact that the IBM VQ error function is heuristic and cannot be accurately represented analytically, thus optimal corresponding perceptual weights are not available.

TABLE I

PESQ RESULTS FOR THE PROPOSED POLYNOMIAL TD SETUPS - SETUPS ARE DESCRIBED IN V-A.

	Avg. PESQ	Min. PESQ	PESQ STD.
Full (Setup #1)	3.67	3.49	0.167
Reduced Poly Order (#2)	3.69	3.51	0.155
Short TD segments (#3)	3.68	3.48	0.142
Naive segmentation (#4) (segment length: 8)	3.70	3.50	0.102
Naive seg b (#4) (segment length: 4)	3.63	3.50	0.078
Embedded Quant.(#5)	3.61	3.23	0.178
Naive downsampling	2.84	2.48	0.137

Combining complexity considerations with the results shown above we can choose the best performing algorithm for various target applications. While the Naive segmentation algorithm (#4a), with maximum segment length of 8 provides the highest PESQ score, it requires use of high order polynomials, (the highest order in our setup was 6), which adds unacceptable complexity to the decoding, which must be performed in real time. The reduced polynomial order setup (#2) and the short TD segment setup (#3) both provide good quality, with the reduced polynomial order resulting in a slightly higher worst-case PESQ score, but also a very slight decrease in the 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 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 reduced polynomial order TD setup, is recommended for recompression of the acoustic leaves.

B. 3D SADCT based recompression

BIT ALLOCATIONS

In Section III-C1 we described the two stage quantizer design. We will now detail the bit allocations obtained at each stage when applied to our database. The bit allocations obtained in Stage I, which

	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	

Fig. 8. Splitting of 3D SADCT coefficient vectors into $M = 5$ groups and corresponding bit allocations

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	-	-	-

Fig. 9. Vector split and bit allocation for each SADCT coefficient group

finds the appropriate bit allocation for the 32 element vector for each pair of $\{u, v\}$ values, (where u corresponds to the segment index and v corresponds to the frame index within the segment), are shown in Fig. 8. These results were obtained using the ordering algorithm described in IV. For Stage II, which allocates bits along the third dimension of the leaf, which corresponds to the 32 elements of the parameter vector, we obtain a bit-allocation for each of the groups found previously, in Stage I. This entire process results in 5 sets of split locations and bit allocations. Fig. 9 describes for each of the 5 groups, which elements belong to each of the quantizer 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). As explained in Section III-C1, we then use these allocations to design corresponding vector quantizers, with the classic LBG approach.

Results

Using 3D-SADCT and the quantizers we designed, we compressed the leaves in the database. Then the quality of the reconstructed speech was evaluated for the 10 sample sentences. The obtained PESQ scores, with the speech synthesized using [1] as reference, are provided in the last two rows of Table II. These results were obtained for a re-compression factor of x2 (43 bits on average per 32 element vectors) and are shown for both cases - with and without reordering. Note that while the reordering did not really improve the average performance, it did improve the worst-case, which is also reflected in a lower PESQ standard deviation for the test sentences: 0.14 with reordering vs. 0.23 without.

C. Metropolis based ordering

Cost Functions

In order to apply the Metropolis based ordering algorithm, described in Section IV, we must define the corresponding cost functions for each of the two proposed recompression algorithms.

The cost function used to determine the best segment order for the polynomial TD compression is calculated over the entire super-segment, consisting of N speech frames. This cost function attempts to maximize the 'super-segment' smoothness by minimizing the distance to a second order polynomial (smooth function). The cost is defined as follows:

$$C_{TD} = \sum_{i=1}^{32} w_i \sum_{n=1}^{N} (V_{n,i} - P_i(n))^2 \tag{6}$$

Where the weights $w_i = \frac{1}{i}$, $V_{n,i}$ is the value of the i^{th} feature vector element at the n^{th} frame and P_i is a second order polynomial fitted to the trajectory of the i^{th} vector element, along the entire super-segment. The weighting scheme prioritizes the lower frequency parameters, as they are perceptually more important.

As described in III-C1, for the 3D SADCT based compression the leaf data is split into M groups, $\{G_m\}_{m=1}^{M}$, of varying bit-rates. Our goal is to contain as much of the leaf energy as possible in G_2, the lowest non-DC frequency group. Therefore we define the cost function as:

$$C_{SADCT} = 1 - \frac{\sum_{\{u,v\}\in G_2} \sum_{w=1}^{32} F_{u,v,w}^2}{\sum_{\{u,v\}\in G_{2,..,M}} \sum_{w=1}^{32} F_{u,v,w}^2} \tag{7}$$

Where $F_{u,v,w}$ are the 3D SADCT coefficient values. In order to obtain an initial group splitting to use in this equation, the joint bit allocation and splitting algorithm is performed once on the leaves in their original order. Then these initial groups are used to calculate the cost of each proposed order, as given by (7), and the splitting and bit allocation is performed again using the reordered leaves.

Reordering for increased Energy compaction

We examined the contribution of the reordering algorithm to leaf energy compaction, for the SADCT based recompression. Fig. 10 demonstrates the improved energy compaction obtained when using the

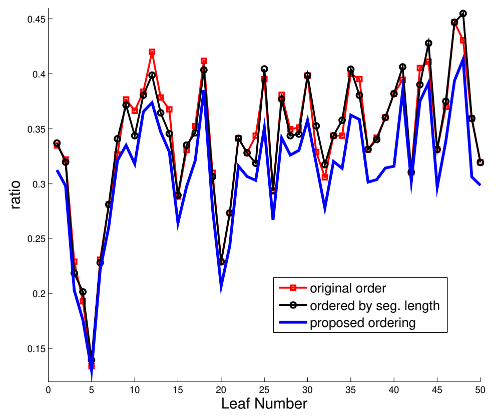

Fig. 10. Ordering performance: Ratio of SADCT coefficients that hold 95% of the total leaf energy, shown when ordering speech segments according to their original order, sorted according to length and with the proposed Metropolis based ordering

proposed ordering algorithm for 50 sample leaves. The figure shows the ratio of coefficients that hold 95% of the total energy when: (i) Performing SADCT on the leaves with the segments in their original order (average ratio: 0.341) (ii) Performing naive ordering - according to segment length (average: 0.340) (iii) Applying the proposed Metropolis based ordering (average: 0.312). As seen here the Metropolis based ordering increases energy compactness.

Results

The PESQ results for the two proposed recompression algorithms (TD and 3D SADCT), with and without reordering, are shown in Table II. Regarding the contribution of the segment reordering to overall performance, for the selected TD setup the reordering had little effect, since the selected TD segments tend to be very short and rarely extend across speech segment boundaries. For the 'Full' setup however, which allows polynomial of orders up to 4, the average PESQ score increases from 3.55, without reordering, to the aforementioned 3.67 (Table I), with the proposed reordering. For the proposed 3D SADCT algorithm, while segment reordering does not contribute much for the overall or average performance, it does improve the worst case performance and increases the stability of the solution across sentences. Since the ordering is performed offline its slight contribution to overall performance may still justify its usage.

D. *Summary of results*

Table II summarizes the results of the proposed algorithms for x2 recompression compared to naive downsampling (using appropriate pre-filtering and linear interpolation). As seen there, the 3D SADCT obtains the highest PESQ score. As both the reference and test signals consist of synthesized speech, no degradation was perceived in informal listening tests for the obtained average PESQ scores of 3.7-3.8.

Regarding added decoding complexity of the proposed solutions: For the reduced polynomial order TD approaches, only linear interpolation is required which has negligible complexity compared to the speech synthesis process. The setups using higher order polynomials may however not be suitable for end devices with limited resources. For the 3D SADCT approach, complexity is slightly higher, but, assuming usage of highly optimized DCT implementations which exist for most embedded devices, the complexity overhead is still very low. For comparison, the low complexity speech synthesis described in [1], requires applying a 512 tap FFT as part of the synthesis process of each speech frame, which has complexity of $O(256 \cdot log(512)) = O(2304)$ operations. The inverse 3D SADCT on the other hand, for a leaf with M segments, of average length N frames, requires M*N IDCTs of length 32, N*32 IDCTs of length M and M*32 IDCTs of length N. For the median values in our database, M=6 and N=2, this results in a total of $O(1648)$ operations, in addition to inverse quantization. Thus reconstructing the parameters of all the speech frames in the leaf, generally has lower complexity than the FFT part of the speech synthesis of a single speech frame. The encoding complexity of the proposed algorithms is significantly higher than the decoding complexity, and includes iterative processes. However for CTTS systems this in not of particular interest, as the effort invested in the initial creation and compression of such a database is very high and is performed only once, off-line.

VI. Conclusion

Faced with a high footprint acoustic database for a Concatenative Text-To-Speech synthesizer, we addressed the problem of reducing its footprint without compromising the perceptual quality of the synthesized speech.

We presented two algorithmic approaches for the recompression. The first is a Vector Polynomial TD approach, which we applied to the super-segments created by concatenating the speech segments in each acoustic leaf. For each leaf, given a target distortion value, we perform optimal segmentation and polynomial order selection. 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 form of vectorial TD that is well suited for short segments of discontinuous speech data. As the polynomials are represented by

TABLE II

PESQ PERFORMANCE OF PROPOSED ALGORITHMS, WITH AND WITHOUT PROPOSED SEGMENT REORDERING VS. NAIVE DOWNSAMPLING, EVALUATED FOR 10 TEST SENTENCES

Algorithm	Average PESQ	Minimum PESQ
Naive downsampling	2.84	2.48
Pol. TD	3.66	3.50
Pol. TD + reord.	3.69	3.51
3D SADCT	3.84	3.53
3D SADCT + reord.	3.85	3.65

their samples, we perform the sampling jointly along the vectors and hence are able to use the systems original vector coding tools. The second algorithmic approach is based on 3D SADCT. We reviewed the underlying theory and showed that good energy compaction can be obtained for our data using this approach. We addressed the issue of coefficient quantization and presented a methodical bit allocation and splitting approach, used along with LBG, to obtain a split-VQ algorithm for the coefficient quantization. We also presented an accompanying segment ordering algorithm. Although it only had a small effect on the overall performance of the two compression algorithms, it may be of interest in other applications that require offline ordering.

We have shown that by applying both proposed algorithms, Polynomial TD and 3D SADCT, to a specific, small footprint CTTS database, we can provide perceptually equivalent speech with a x2 compression ratio. The SADCT based approach provides higher PESQ scores than the Polynomial TD based approach. However, it cannot be easily adjusted to any desired recompression ratio, as new quantizers must be designed for each target ratio as opposed to the polynomial TD based approach which can provide any desired compression ratio in a fully automated process.

In this paper the proposed algorithms were applied to a specific acoustic leaf database for the sake of performance evaluation, however these algorithms have has a much wider scope, and may be used for compression of any CTTS database or any other database consisting of 3D ”buckets” of values that contain some extent of redundancy, such as sign language databases, databases used for image classification and others.

ACKNOWLEDGMENT

This research was performed at the Signal and Image Processing Lab (SIPL), Technion I.I.T, in collaboration with IBM's Haifa Research Lab (HRL). The authors would like to thank Ron Hoory, head of the Speech Technologies Group in HRL, Zvi Kons and the entire group for their co-operation and ongoing support. We thank the devoted SIPL staff: Nimrod Peleg, Yair Moshe, Ziva Avni and Avi Rosen for creating the productive and pleasant environment that enabled and encouraged this research.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] T. Shoham, D. Malah, and S. Shechtman, "Footprint reduction of concatenative text-to-speech synthesizers using polynomial temporal decomposition," *ICSSCP 2010*, March 2010.

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

[6] B.S. Atal, "Efficient coding of LPC parameters by temporal decomposition," in *ICASSP*, Boston, Mass, USA, April 1983, vol. 1, pp. 81–84.

[7] 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*, pp. 1016–1026, Oct. 2003.

[8] F. Bimbot, G. Chollet, P. Deleglise, and C. Montacie, "Temporal decomposition and acoustic-phonetic decoding of speech," in *ICASSP*, New-York, NY, USA, April 1988, vol. 1, pp. 445–448.

[9] Y.M. Cheng and D. O'Shaughnessy, "On 450-600 b/s natural sounding speech coding," *IEEE Trans. on Speech and Audio Proc.*, vol. 1, no. 2, pp. 207–220, April 1993.

[10] G. Ahlbom, F. Bimbot, and G. Chollet, "Modeling spectral speech transitions using temporal decomposition techniques," in *ICASSP*, April 1987, vol. 12, pp. 13–16.

[11] 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.

[12] 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*, vol. 49, no. 4, pp. 292–304, Apr. 2007.

[13] L. Girin, M. Firouzmand, and S. Marchand, "Perceptual long-term variable-rate sinusoidal modeling of speech," *IEEE Transactions on Audio Speech and Language Processing*, vol. 15, no. 3, pp. 851–861, Mar. 2007.

[14] M. Firouzmand and L. Girin, "Long-term flexible 2D cepstral modeling of speech spectral amplitudes," in *ICASSP*, Las-Vegas, NV, USA, March 2008, pp. 3937–3940.

[15] L. Girin, “Adaptive long-term coding of LSF parameters trajectories for large-delay/very- to ultra-low bit-rate speech coding,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2010, 2010, Article ID 597039.

[16] 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*, Jeju Island, Korea, Oct. 2004, pp. 1993–1996.

[17] S. Dusan, J.L. Flanagan, A. Karve, and M. Balaraman, “Speech compression by polynomial approximation,” *IEEE Transactions on Audio Speech and Language Processing*, vol. 15, no. 2, pp. 387–395, Feb. 2007.

[18] R. Nygaard and D. Haugland, “Compressing ECG signals by piecewise polynomial approximation,” in *ICASSP*, Seattle, May 1998, vol. 3, pp. 1809–1812.

[19] R. Nygaard, G. Melnikov, and A. K. Katsaggelos, “A rate distortion optimal ecg coding algorithm,” *IEEE Transactions on biomedical engineering*, vol. 48, no. 1, pp. 28–40, Jan. 2001.

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

[21] T. Shoham, “Quality-preserving footprint-reduction of concatenative text-to-speech synthesizers,” M.Sc. thesis, Faculty of EE, Technion, 2010, http://sipl.technion.ac.il/siglib/FP/Tamar-Shoham-thesis_all.pdf.

[22] P. Prandoni and M. Vetterli, “R/d optimal linear prediction,” *IEEE Trans. on Speech and Audio Proc.*, vol. 8, no. 6, pp. 646–655, Nov. 2000.

[23] P. Prandoni and M. Vetterli, “Approximation and compression of piecewise smooth functions,” *Philosophical Transactions: Mathematical, Physical and Engineering Sciences*, vol. 357, no. 1760, pp. 2573–2591, Sep. 1999.

[24] N. Farvardin and R. Laroia, “Efficient encoding of speech LSP parameters using the discrete cosine transformation,” in *ICASSP*, Glasgow, UK, May 1989, vol. 1, pp. 168–171.

[25] D. J. Mudugamuwa and A. B. Bradley, “Optimal transform for segmented parametric speech coding,” in *ICASSP*, Seattle, WA, USA, May 1998, vol. 1, pp. 53–56.

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

[27] D. Markman and D. Malah, “Hyperspectral image coding using 3D transforms,” in *International Conference on Image Processing (ICIP)*, Thessaloniki, Greece, Oct. 2001, vol. 1, pp. 114–117.

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

[29] S. Chatterjeet and T. V. Sreenivas, “Optimum transform domain split VQ,” *IEEE Signal Proc. Letters*, vol. 15, pp. 285–288, 2008.

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

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

[32] Y. Linde, A. Buzo, and R. Gray, “An algorithm for vector quantizer design,” *IEEE Trans. on Communications*, vol. COM-28, no. 1, pp. 84–95, Jan. 1980.

[33] K. Zeger and A. Gersho, “Pseudo-gray coding,” *IEEE Trans. on Communications*, vol. 38, no. 12, pp. 2147–2158, Dec. 1990.

[34] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, “Optimization by simulated annealing,” *Science, New Series*, vol. 220, no. 4598, pp. 671–680, May 1983.

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

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