FAST LOW-LATENCY CONVOLUTION BY LOW-RANK TENSOR APPROXIMATION

Martin Jälmby[*], *Filip Elvander*[†],*Toon van Waterschoot*[*]

[*]Dept. of Electrical Engineering, ESAT-STADIUS, KU Leuven, Belgium
[†]Dept. of Information and Communications Engineering, Aalto University, Finland

ABSTRACT

In this paper we consider fast time-domain convolution, exploiting low-rank properties of an impulse response (IR). This reduces the computational complexity, speeding up the convolution, without introducing latency. Previous work has considered a truncated singular value decomposition (SVD) of a two-dimensional matricization, or reshaping, of the IR. We here build upon this idea, by providing an algorithm for convolution with a three-dimensional tensorization of the IR. We provide simulations using real-life acoustic room impulse responses (RIRs) of various lengths, convolving them with music, as well as speech signals. The proposed algorithm is shown to outperform the comparable existing algorithm in terms of signal quality degradation, for all considered scenarios, without increasing the computational complexity, or the memory usage.

Index Terms— Low-rank modeling, convolution, tensor decomposition

1. INTRODUCTION

Ever since the seminal works by Cooley and Tukey, [1], and Stokham, [2], a popular approach to efficiently compute the convolution of two time-domain signals has been to perform it in the discrete frequency domain. The convolution theorem states that convolution in the time-domain equals point-wise multiplication in the frequency domain, allowing for considerable reduction of the computational complexity in most cases, owing to the computational efficiency of the fast Fourier transform (FFT) algorithm. The frequency-domain convolution has been further improved by methods such as overlap-add (OLA) and overlap-save (OLS), and partitioned convolution (see e.g. [3]). Frequency-domain convolution is, however, block-based and inevitably introduces input-output latency [4]. Perceptual convolution, introduced by Lee *et al.* in [5], is another possible way to speed up the convolution. Here, a perceptual criterion is used in order to reduce the computational complexity, without considerable signal quality degradation. Yet another approach to accelerate computations is optimizing the implementation with respect to the processor architecture, and the use of graphics processing units (see e.g. [6] and the references therein).

This research work was carried out at the ESAT Laboratory of KU Leuven. The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program / ERC Consolidator Grant: SONORA (no. 773268). This paper reflects only the authors' views and the Union is not liable for any use that may be made of the contained information.

In this paper we look to exploit the (approximate) low-rank structure of an impulse response (IR) in order to carry out fast time-domain convolution. Low-rank modeling is used in a variety of applications, such as image analysis [7], matrix completion [8], and recently also acoustic signal processing [9, 10, 11], and has been shown to yield possibly more compact models. Utilizing low rank in time-domain convolution has been considered before by Atkins *et al.* in [4]. Therein, the authors consider a low-rank approximation of a two-dimensional matricization of the IR. In this work, we extend upon this idea and show how the concept of low-rank convolution can be generalized, and we provide a detailed algorithm for the three-dimensional case. This is motivated by the findings in [10], where it was shown that higher-order tensorization yields a lower IR approximation error, given a fixed compression rate. By performing the convolution in the time domain, low latency can be achieved. In [4], to further promote a compact representation of the IR, the resulting finite impulse response (FIR) filter is approximated by an infinite impulse response (IIR) filter. Herein, the proposed method is concerned with exploiting low-rank structure of IRs on FIR form, with possible extensions of the IIR methodology being a topic of future research. Accordingly, as to allow for a fair comparison between the algorithm in [4] and the herein proposed method, the IIR approximation is not considered in the numerical section.

This paper is organized as follows: first Section 1 is concluded with an introduction of the notation used in this paper. Then, in Section 2, we introduce the signal model. Low-rank convolution is explained in Section 3, where we first give an account of the algorithm from [4], then introduce the proposed algorithm, and lastly expand on its complexity. Simulation results are presented in Section 4, and in Section 5 we present our conclusions.

1.1. Notation

We denote scalars, vectors, matrices, and tensors by lower case (e.g., h), bold lowercase (e.g., $\mathbf{h}$), bold uppercase (e.g., $\mathbf{H}$), and calligraphic letters (e.g., $\mathcal{H}$), respectively. Linear operators are also denoted by calligraphic letters, but it will be clear from context what is considered. The selection of one or several elements from a vector, matrix, or tensor will be denoted by square brackets, e.g. $\mathbf{H}[m:n,j]$ is a vector containing the mth till the nth element of the jth column of $\mathbf{H}$. The symbol $\circ$ denotes the outer product, i.e., $(\mathbf{x}_1 \circ \mathbf{x}_2 \circ \cdots \circ \mathbf{x}_D)[j_1, j_2, \ldots, j_D] = \mathbf{x}_1[j_1]\mathbf{x}_2[j_2]\ldots\mathbf{x}_D[j_D]$.

2. SIGNAL MODEL

In this work we consider a discrete-time impulse response (IR) $h(k)$, for $k = 0, 1, \ldots, n_h - 1$, arranged in the vector $\mathbf{h} \in \mathbb{R}^{n_h}$, as well as a discrete-time signal $x(k)$, for $k = 1, 2, \ldots, n_x$, arranged in the vector $\mathbf{x} \in \mathbb{R}^{n_x}$. The convolution of these vectors yields the discrete-time output

$$y(k) = \sum_{n=0}^{n_h-1} h(n)x(k-n), \tag{1}$$

for $k = 1, 2, \ldots, n_y$, with corresponding vector $\mathbf{y} \in \mathbb{R}^{n_y}$, where $n_y = n_h + n_x - 1$. Generally, throughout this paper, an element is considered to be 0, if the index is out of its defined range, equivalent to appropriate zero-padding.

Many signals can be considered *low-rank*, in the sense that if the signal vector is reshaped into a matrix, the matrix will have low rank (see e.g., [12]). This persists for higher-order tensorization, where the rank of a tensor $\mathcal{H} \in \mathbb{R}^{n_{s_1} \times n_{s_2} \times \cdots \times n_{s_D}}$, with n_{s_d} denoting the size of the dth dimension, $d = 1, 2, \ldots, D$, is defined as the smallest number of rank-1 tensors needed to generate the tensor as their sum. As IRs used in real-world applications are often estimated from noisy measurements, a recorded (estimated) IR will rarely be low-rank in a strict sense. Therefore, in this paper, *low-rank* is used in a less strict sense, i.e., a low-rank approximation will render a small approximation error. An example of this are acoustic room impulse responses (RIRs), which we have explored in previous work [10].

3. LOW-RANK CONVOLUTION ALGORITHM

3.1. Partitioned Truncated SVD Filter

As to give background to the herein proposed method for fast convolution, we first give a brief description of the method in [4]. Assuming $n_h = n_{s_1} n_{s_2}$, for $n_{s_1}, n_{s_2} \in \mathbb{N}$, an output sample $y(k)$ of the convolution in (1) can instead be written as

$$y(k) = \sum_{j=1}^{n_{s_2}} \mathbf{x}_k^{(j)^T} \mathbf{h}^{(j)}, \tag{2}$$

where $\mathbf{h}^{(j)} \triangleq \begin{bmatrix} h((j-1)n_{s_1}) & \ldots & h(jn_{s_1}-1) \end{bmatrix}$ and $\mathbf{x}_k^{(j)} \triangleq \begin{bmatrix} x(k-(j-1)n_{s_1}) & \ldots & x(k-jn_{s_1}+1) \end{bmatrix}$, for $j = 1, 2, \ldots, n_{s_2}$. This is merely a rearrangement, as in (1), $y(k)$ is written as an inner product computed from vectors of length $n_h = n_{s_1} n_{s_2}$, whereas in (2), it is written as the sum of n_{s_2} inner products, where the corresponding vectors are of length n_{s_1}. Furthermore, the IR $\mathbf{h}$ can be reshaped into a matrix $\mathbf{H} = \begin{bmatrix} \mathbf{h}^{(1)} & \ldots & \mathbf{h}^{(n_{s_2})} \end{bmatrix} \in \mathbb{R}^{n_{s_1} \times n_{s_2}}$. Assuming that this matrix is rank-1, i.e., it can be written as the outer product $\mathbf{H} = \mathbf{s}_1 \circ \mathbf{s}_2$, for two vectors $\mathbf{s}_1 \in \mathbb{R}^{n_{s_1}}$ and $\mathbf{s}_2 \in \mathbb{R}^{n_{s_2}}$, we have that

$$\mathbf{H} = \begin{bmatrix} \mathbf{s}_1\mathbf{s}_2[1] & \mathbf{s}_1\mathbf{s}_2[2] & \ldots & \mathbf{s}_1\mathbf{s}_2[n_{s_2}] \end{bmatrix}, \tag{3}$$

i.e., the jth column of $\mathbf{H}$, corresponding to $\mathbf{h}^{(j)}$, is the vector $\mathbf{s}_1$ scaled by $\mathbf{s}_2[j]$, $j = 1, 2, \ldots, n_{s_2}$. The following property is readily verified,

$$\mathbf{x}_k^{(j)} = \mathbf{x}_{k+an_{s_1}}^{(j+a)}, a \in \mathbb{Z}. \tag{4}$$

Because of (3) and (4), only the first inner product of the sum in (2) has to be computed per output sample k, as

$$y(k) = \left(\mathbf{x}_k^{(1)^T}\mathbf{s}_1\right)\mathbf{s}_2[1] + \sum_{j=2}^{n_{s_2}} \left(\mathbf{x}_k^{(j)^T}\mathbf{s}_1\right)\mathbf{s}_2[j]. \tag{5}$$

The other inner products of the sum, i.e., $\mathbf{x}_k^{(j)^T}\mathbf{s}_1$, for $j = 2, \ldots, n_{s_2}$, have already been computed for a previous time sample, and can therefore be fetched from memory and multiplied with the appropriate entry from $\mathbf{s}_2$. This reduces the number of multiplications per sample to be carried out, from $n_h = n_{s_1}n_{s_2}$ to $n_{s_1} + n_{s_2}$. These ideas can be extended to a matrix $\mathbf{H}$ of arbitrary rank R. Instead of $\mathbf{H}$ being just the outer product of two vectors, it is now a sum of R outer products,

$$\mathbf{H} = \mathbf{S}_1\mathbf{S}_2^T = \sum_{r=1}^{R} \mathbf{S}_1[:,r] \circ \mathbf{S}_2[:,r] = \sum_{r=1}^{R} \mathbf{S}_1[:,r]\mathbf{S}_2[:,r]^T, \tag{6}$$

for $\mathbf{S}_1 \in \mathbb{R}^{n_{s_1} \times R}$, and $\mathbf{S}_2 \in \mathbb{R}^{n_{s_2} \times R}$. We have a similar pattern as in (3) in that

$$\mathbf{h}^{(j)} = \mathbf{H}[:,j] = \sum_{r=1}^{R} \mathbf{S}_1[:,r]\mathbf{S}_2[j,r], \tag{7}$$

which enables us to extend (5) to

$$y(k) = \sum_{r=1}^{R} \left(\mathbf{x}_k^{(1)^T}\mathbf{S}_1[:,r]\mathbf{S}_2[1,r] + \sum_{j=2}^{n_{s_2}} \mathbf{x}_k^{(j)^T}\mathbf{S}_1[:,r]\mathbf{S}_2[j,r]\right), \tag{8}$$

where only R inner products have to be computed for each time sample. Similar to (5), this reduces the number of multiplications to $R(n_{s_1} + n_{s_2})$.

3.2. Fast Convolution by Tensor Approximation

Next, we show how the ideas from [4] can extended, and propose a detailed algorithm for the three-dimensional case.. Let the IR $\mathbf{h}$ be reshaped into a tensor $\mathcal{H} \in \mathbb{R}^{n_{s_1} \times n_{s_2} \times \cdots \times n_{s_D}}$, and assume that $\mathcal{H}$ is of rank R. Then, analogously to (6),

$$\mathcal{H} = \sum_{r=1}^{R} \mathbf{S}_1[:,r] \circ \mathbf{S}_2[:,r] \circ \cdots \circ \mathbf{S}_D[:,r], \tag{9}$$

where $\mathbf{S}_d \in \mathbb{R}^{n_{s_d} \times R}$, $d = 1, 2, \ldots, D$, and in analog to (7) we have that

$$\mathcal{H}[:, j_2, j_3, \ldots, j_D] = \sum_{r=1}^{R} \mathbf{S}_1[:,r] \mathbf{S}_2[j_2, r] \ldots \mathbf{S}_D[j_D, r]. \tag{10}$$

The equality of (4) can be generalized according to

$$\mathbf{x}_k^{(j_2, j_3, \ldots, j_D)} = \mathbf{x}_{k+\sum_{m=2}^{D} a_m \prod_{d=1}^{m-1} n_{s_d}}^{(j_2+a_2, j_3+a_3, \ldots, j_D+a_D)}, \tag{11}$$

where $\mathbf{x}_k^{(j_2, j_3, \ldots, j_D)} \in \mathbb{R}^{n_{s_1}}$ is a vector containing the n_{s_1} latest samples of $\mathbf{x}$, in reversed order, starting at $x(k - \sum_{m=2}^{D}(j_m - 1)\prod_{d=1}^{m-1} n_{s_d}))$, and $a_2, a_3, \ldots, a_D \in \mathbb{Z}$. The pattern from (2) extends to

$$y(k) = \sum_{j_2=1}^{n_{s_2}} \cdots \sum_{j_D=1}^{n_{s_D}} \mathbf{x}_k^{(j_2, j_3, \ldots, j_D)^T} \mathbf{h}^{(j_2, j_3, \ldots, j_D)}, \tag{12}$$

where $\mathbf{h}^{(j_2, j_3, \ldots, j_D)} = \mathcal{H}[:, j_2, j_3, \ldots, j_D]$ is a vector containing n_{s_1} consecutive elements of $\mathbf{h}$, starting at $h(\sum_{m=2}^{D}(j_m - 1)\prod_{d=1}^{m-1} n_{s_d})$. Subsequently, the property of (8) is generalized to

$$y(k) = \sum_{r=1}^{R} \sum_{j_2=1}^{n_{s_2}} \ldots \sum_{j_D=1}^{n_{s_D}} \mathbf{x}_k^{(j_2, \ldots, j_D)^T} \mathbf{S}_1[:,r] \mathbf{S}_2[j_2, r] \ldots \mathbf{S}_D[j_D, r]. \tag{13}$$

Here, it may be noted that only R inner products of length n_{s_1} have to be computed for each k, reducing the number of multiplications to $R\sum_{d=1}^{D} n_{s_d}$. However, if naively implemented, the sum in (13) will yield many superfluous operations, computing inner products where one of the vectors contains only zeroes. In order to fully exploit the IR structure and to maximize efficiency, it is therefore important to keep track of which operations actually need to be carried out. We here propose an explicit algorithm for the case $D = 3$.

Let $\mathcal{H} = \sum_{r=1}^{R} \mathbf{S}_1[:,r] \circ \mathbf{S}_2[:,r] \circ \mathbf{S}_3[:,r]$, where $\mathcal{H} \in \mathbb{R}^{n_{s_1} \times n_{s_2} \times n_{s_3}}$ and $\mathbf{S}_d \in \mathbb{R}^{n_{s_d} \times R}$, for $d = 1, 2, 3$. The operator $\mathcal{I} : \mathbb{R}^n \to \mathbb{R}^n$ denotes the reversion of the order of the elements in a vector, i.e., $\mathcal{I}(\mathbf{x}) = \begin{bmatrix} x(n_x) & x(n_x - 1) & \ldots & x(1) \end{bmatrix}^T$, and $\mathbf{0}_n \in \mathbb{R}^n$ is a vector of zeros. We introduce the matrix $\mathbf{C} \in \mathbb{R}^{n_h \times R}$ to store the already computed inner products. The rationale of the algorithm is to, for each k, first compute the R necessary inner products, with appropriate zero-padding, store to memory, and add to $y(k)$ with appropriate scaling with corresponding element from $\mathcal{H}$. Next, the remaining non-zero inner products of the sum of (13) are fetched from memory, scaled with the corresponding entry of $\mathcal{H}$ and added to $y(k)$. The proposed algorithm is summarized in Algorithm 1.

Algorithm 1: Fast Low-latency Convolution by Low-rank 3-D Tensor Approximation

```
Input: $\mathcal{H} = \sum_{r=1}^{R} \mathbf{S}_1[:,r] \circ \mathbf{S}_2[:,r] \circ \mathbf{S}_3[:,r]$, $\mathbf{x}$
Output: $\mathbf{y}$
for $k = 1, 2, \ldots, n_y$ do
    for $r = 1, 2, \ldots, R$ do
        if $k \leq n_{s_1} + n_x - 1$ then
            $z_{\text{b}} = \max(k - n_x, 0)$;
            $z_{\text{a}} = \max(n_{s_1} - k, 0)$;
            $x_{\text{b}} = \max(k - n_{s_1} + 1, 1)$;
            $x_{\text{e}} = \min(k, n_x)$;
            $\mathbf{x}_k = \begin{bmatrix} \mathbf{0}_{z_{\text{b}}}^T & \mathcal{I}(\mathbf{x}[x_{\text{b}} : x_{\text{e}}])^T & \mathbf{0}_{z_{\text{a}}}^T \end{bmatrix}^T$;
            $\mathbf{C}[\text{mod}(k-1, n_h) + 1, r] = \mathbf{x}_k^T \mathbf{S}_1[:,r]$;
            $y(k) = \mathbf{S}_2[1,r]\mathbf{S}_3[1,r]\mathbf{C}[\text{mod}(k-1,n_h)+1,r]$;
        $l = \max\left(\lfloor (k - n_x)/n_{s_1} \rfloor + 1, 2\right)$;
        $u = \min\left(\lfloor (k - 1)/n_{s_1} \rfloor + 1, n_{s_2} n_{s_3}\right)$;
        for $c = l : 1 : u$ do
            $j_2 = \text{mod}(c - 1, n_{s_2}) + 1$;
            $j_3 = \lfloor (c-1)/n_{s_2} \rfloor + 1$;
            $\tilde{c} = k - ((j_2 - 1)n_{s_1} + (j_3 - 1)n_{s_1} n_{s_2}) - 1$;
            $y(k) = y(k) +$
              $\mathbf{S}_2[j_2, r]\mathbf{S}_3[j_3, r]\mathbf{C}[\text{mod}(\tilde{c}, n_h) + 1, r]$;
```

3.3. Complexity

As noted by the authors of [4], an output sample $y(k)$ requires $R(n_{s_1} + n_{s_2})$ multiply-add instructions in the two-dimensional case, whereas a conventional time-domain FIR filter requires n_h operations, where we remind the reader that $n_h = \prod_{d=1}^{D} n_{s_d}$. The computational complexity for a general, D-dimensional tensorization, is $R\sum_{d=1}^{D} n_{s_d}$, meaning that the computational complexity of the proposed algorithm generalizes the one of [4]. Furthermore, the contribution to the end result from the different entries in the sums of (13) are independent from each other. It is therefore possible to parallelize the computations.

The two-dimensional algorithm from [4] requires a memory of size $R(n_{s_1} + n_{s_2} + n_h) + n_{s_1}$ variables, compared to $2n_h$ for a conventional FIR filter. For the proposed method, a memory of size $R(\sum_{d=1}^{D} n_{s_d} + n_h) + n_{s_1}$ is required, i.e., the memory complexity of the proposed algorithm directly generalizes the one of [4].

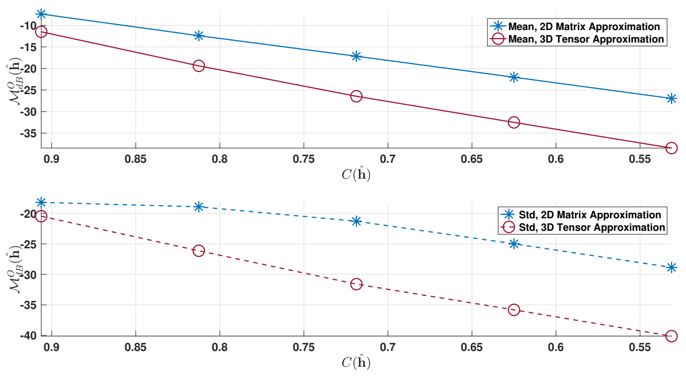

Fig. 1. Normalized output mean-squared error, as a function of compression rate, for speech signals and short RIRs: mean (top) and standard deviation (bottom).

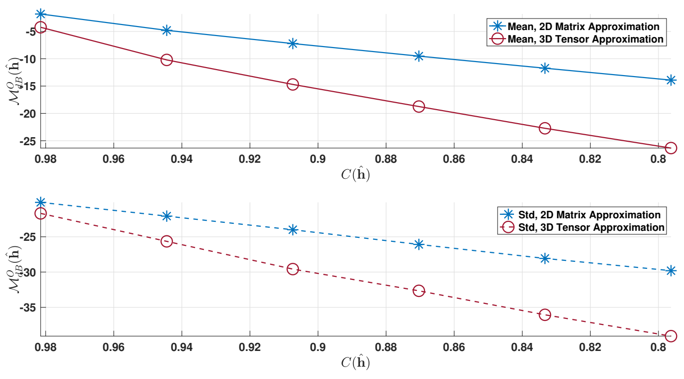

Fig. 2. Normalized output mean-squared error, as a function of compression rate, for music signals and long RIRs: mean (top) and standard deviation (bottom).

4. RESULTS

We denote by $\Upsilon(\mathcal{H}) = R\sum_{d=1}^{D} n_{s_d}$, with $D = 2, 3$, for the two algorithms compared here ($D = 2$ for [4], and $D = 3$ for Algorithm 1), the number of multiply-add instructions needed for a low-rank convolution of rank R. Furhter, by

$$C(\mathcal{H}) = 1 - \frac{\Upsilon(\mathcal{H})}{n_h}, \tag{14}$$

where $C(\mathcal{H}) \in [0, 1)$, we denote the *complexity reduction*, relative to time-domain convolution. For $C(\mathcal{H}) = 0$ there is no complexity reduction, whereas for $C(\mathcal{H})$ closer to 1, the degree of complexity reduction is larger. The accuracy is measured by the *normalized output mean-squared error*

$$\mathcal{M}^O_{\text{dB}}(\mathcal{H}) = 20\log_{10}\left(\mathbb{E}\left[\frac{\|\mathcal{H} * \mathbf{x} - \mathbf{h} * \mathbf{x}\|_2}{\|\mathbf{h} * \mathbf{x}\|_2}\right]\right), \tag{15}$$

where $\mathbf{h}$ denotes the original IR, and $\mathcal{H} * \mathbf{x}$ is the low-rank convolution, as defined in (13). The low-rank approximations of 2D-matrices are performed using a truncated SVD. For the numerical computation of the tensor decompositions we use the high-level function *cpd* of the Matlab toolbox Tensorlab [13]. To simplify the exposition, we will only consider the square case, i.e., $n_{s_1} = n_{s_2} = n_{s_3}$.

To demonstrate the performance of the proposed algorithm, we first apply it to RIRs from the *single- and multichannel audio recordings database* (SMARD) [14], and speech signals from the TSP speech database [15]. The RIRs from SMARD are from a regular office-sized room, with a reverberation time [1] of 0.15 seconds. The speech signals from TSP are simple utterances by both male and female speakers, in an anechoic environment. Both SMARD and TSP are sampled at 48 kHz. From the databases we randomly choose 100 RIRs and 100 speech signals. The RIRs are set to start right before the arrival of the direct component and set to be of length $n_h = 4096$ samples. The speech signals are truncated at one second, i.e., $n_x = 48000$. The averaged results are shown in Fig. 1. There it can be seen that the proposed algorithm outperforms the algorithm from [4] for all considered values of the compression rate. Further it can be noted that the standard deviation for the proposed method is lower throughout.

[1]The time required for the sound level to drop 60 dB after switching off a stationary source.

Next we showcase that the proposed algorithm works very well also for longer RIRs, which we take from [16], a binaural RIR database recorded at RWTH Aachen University. We use 24 different RIRs, recorded at 48 kHz in a lecture room, with a reverberation time of 0.78 seconds, and we let $n_h = 46656$. As input signals we use music from the EBU-SQAM database [17], which contains snippets of reverberant music from various genres, sampled at 44.1 kHz. We use 30 seconds of pieces by ABBA, Verdi, and Händel respectively. To have a sampling frequency matching that of the RIRs, we upsample the music to 48 kHz using Matlab's *resample*, yielding $n_x = 1.44 \cdot 10^6$. The averaged results are shown in Fig. 2. The proposed algorithm is superior to the algorithm from [4] also under these circumstances. The standard deviation is lower for the proposed algorithm in this case as well.

5. CONCLUSIONS

In this paper we have shown that the ideas of low-rank convolution, previously presented for two dimensions, are extendable to higher dimensions, to obtain fast low-latency convolution algorithms. We have outlined the ideas and provided an algorithm for a three-dimensional tensorization of the IR, and shown that this outperforms the previously presented method we extend upon, in terms of signal quality degradation, without increasing the computational complexity, or the memory usage. This is shown with simulations using real-life RIRs of various lengths, convolved with both music and speech signals. Future work will focus on the development of an algorithm for tensors of arbitrary dimensions, and how that algorithms performs with respect to perceptual measures.

6. REFERENCES

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[2] T. G. Stockham, "High-speed convolution and correlation," in *Proceedings of the April 26-28, 1966, Spring Joint Computer Conference*, New York, NY, USA, 1966, AFIPS '66 (Spring), pp. 229–233, Association for Computing Machinery.

[3] F. Wefers, *Partitioned Convolution Algorithms for Real-Time Auralization*, Logos Verlag, DEU, 2015.

[4] J. Atkins, A. Strauss, and C. Zhang, "Approximate convolution using partitioned truncated singular value decomposition filtering," in *Proc. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 176–180.

[5] W.-C. Lee, C.-M. Liu, C.-H. Yang, and J.-I. Guo, "Fast perceptual convolution for room reverberation," in *6th International Conference on Digital Audio Effects (DAFx-03)*, London, United Kingdom, 2003.

[6] N. Jillings, J. D. Reiss, and R. Stables, "Zero-delay large signal convolution using multiple processor architectures," in *Proc. 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 339–343.

[7] X. Zhou, C. Yang, H. Zhao, and W. Yu, "Low-rank modeling and its applications in image analysis," *ACM Comput. Surv.*, vol. 47, no. 2, dec 2014.

[8] J. Cai, E. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[9] G. Huang, J. Benesty, J. Chen, C. Paleologu, S. Ciochină, W. Kellermann, and I. Cohen, "Acoustic system identification with partially time-varying models based on tensor decompositions," in *Proc. International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2022, pp. 1–5.

[10] M. Jälmby, F. Elvander, and T. van Waterschoot, "Low-rank tensor modeling of room impulse responses," in *Proc. 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 111–115.

[11] M. Jälmby, F. Elvander, and T. van Waterschoot, "Low-rank room impulse response estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 957–969, 2023.

[12] M. Boussé, O. Debals, and L. De Lathauwer, "Tensor-based large-scale blind system identification using segmentation," *IEEE Trans. Signal Process.*, vol. 65, no. 21, pp. 5770–5784, 2017.

[13] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab 3.0," Mar. 2016, Available online, https://www.tensorlab.net.

[14] J. K. Nielsen, J. R. Jensen, S. H. Jensen, and M. G. Christensen, "The single- and multichannel audio recordings database (SMARD)," in *Proc. 2014 Int. Workshop Acoustic Signal Enhancement (IWAENC '14)*, Antibes, France, Sept. 2014.

[15] P. Kabal, "TSP speech database," Tech. Rep., McGill University, 2002.

[16] M. Jeub, M. Schafer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Proc. 16th International Conference on Digital Signal Processing*, 2009, pp. 1–5.

[17] G. Waters, "Sound quality assessment material—recordings for subjective tests: User's handbook for the EBU–SQAM compact disk," *European Broadcasting Union (EBU), Tech. Rep*, pp. 1–13, 1988.