Efficient Lossy Compression of Ultrasound Data

Petr Kleparnik*, Pavel Zemcik*, and Jiri Jaros†
*Department of Computer Graphics and Multimedia, †Department of Computer Systems
Faculty of Information Technology, Brno University of Technology
Brno, Czech Republic
ikleparnik@fit.vutbr.cz, zemcik@fit.vutbr.cz, jarosjir@fit.vutbr.cz

***Abstract*—Large-scale numerical simulations of high-intensity focused ultrasound (HIFU), important for model-based treatment planning, generate large amounts of data. Typically, it is necessary to save hundreds of gigabytes during simulation. We propose a novel algorithm for time varying simulation data compression specialized for HIFU. Our approach is especially focused on the on-the-fly parallel data compression during simulations. Now, we are able to compress 3D pressure time series of linear and nonlinear simulations with very acceptable compression ratios and errors (over 80 % of the space can be saved with an acceptable error). The proposed compression will be helpful for significant reduction of resources, such as storage space, network bandwidth, CPU time, etc., enabling better treatment planning using fast volume data visualizations. The paper describes the proposed method, its experimental evaluation, and comparison to the state of the art.**

***Index Terms*—compression, ultrasound simulation, high intensity focused ultrasound, k-Wave toolbox**

I. Introduction

High-intensity focused ultrasound (HIFU) is an emerging non-invasive therapy. The focused beam of ultrasound, typically made using a big transducer, is sent into the tissue. The cells of tissue in a localized region are rapidly heated which causes the tissue death while the surrounding tissue is left unharmed. Large-scale numerical HIFU simulations are important for precise model-based treatment planning (precise placement of the focus and dosage assessment [1]).

However, two key challenges must be addressed. First, it is necessary to create an acoustic and thermal model that is physically complex due to heterogeneous medium and nonlinear wave propagation. Second, the simulations are computationally very intensive and expensive as they must be executed on large domains with billions of gridpoints [2]. Although the computational requirements have been alleviated by a distributed multi-GPU implementation [1], [2], the amount of generated data quickly becomes unmanageable; moreover, the I/O operation becomes a dominating factor of the simulation run time [1], [3].

Due to large distances traveled by the ultrasound waves relative to the wavelength of the highest harmonic frequency and in order to set precise results and applications in medical treatments, very large simulations have to be performed. At the time being, the resolutions of the ultrasound simulation grid come up to $4096 \times 2048 \times 2048$ in 3D space. Usually, a few time steps in a small area encompassing the focus (sensor mask) or a discontinuous field of point sensors have to be sampled and saved for further processing, e.g. thermal modelling [1], [3]. A typical size of sampled data reaches 0.5TB.

In this paper, we present a novel compression method for time-varying HIFU simulation data (acoustic pressure and velocity). Our approach is focused on the on-the-fly parallel data compression during distributed simulation in which, every grid point of interest in 3D space is processed separately.

At this point, we have ready an experimental implementation of the compression method for the offline datasets processing. We have made first experiments on datasets representing clinical situations with heterogeneous tissue. The results show very promising compression ratios and errors.

II. Related work

Generally, specialised sophisticated compression methods for various application data exists but no any compression method/tool designed especially for HIFU simulation data exists. The most relevant works are focused on the compression of signals gained by ultrasonic imaging.

3D Ultrasound Computed Tomography (USCT) data can be reduced with a compression method based on discrete wavelet transform [4]. Many techniques have been applied to process 1D ultrasonic signals (e.g. Matching Pursuit, 1-D discrete cosine transform (DCT), Walsh-Hadamard transform (WHT)) [5], [6]. Radio frequency (RF) signals after analog to digital conversion before beamforming are compressed with techniques such as peak gates (PG), trace compression (TC), largest variation (LAVA), linear predictive coding (LPC), or also with well-known ZIP, JPEG or MPEG [7], [8]. Some of the mentioned techniques should be applied to the HIFU simulation data; however, they are not focused on the on-the-fly parallel data compression during the large-scale simulations.

Methods designed for on-the-fly data compression are audio codecs. We tried some best-known state-of-the-art codecs, such as MP3, MPEG-4 ACC, OGG, FLAC, OPUS and ADPCM, but most of these methods had worse compression ratio under comparable peak signal-to-noise ratio (with the settings of the best possible quality), than the proposed method for HIFU data. The main reason for this is that some of these methods are based on discarding irrelevancies and redundancies in signal by applying the psychoacoustic features of human perception that tolerates loss of some signal features. However, this is inapplicable on HIFU ultrasound signals.

III. Proposed method

A typical HIFU simulation output contains two basic types of datasets - 3D spatial data, e.g. maximum pressure at defined grid points, and 4D spatial-temporal datasets with time-varying pressure series at defined grid points. In 3D datasets, its aggregated values are stored (min, max, rms) or its final values across the whole simulation domain. In case of time-varying 4D datasets, acoustic pressure, acoustic velocity and intensity can be stored across the defined set of locations (sensor mask).

Our compression method is focused on time-varying signals and designed to 1D. Since the sensor mask can be an arbitrary and sparse set of locations, we are not dealing with spatial coherence. Moreover, by treating every spatial gridpoint independently, there is no need for additional communication on distributed clusters. The goals of the compression method are low memory complexity and high computation speed. Small errors are not so significant. These are the main features that differ from other state-of-the-art compression approaches.

The input signal is emitted from a transducer into the modeled tissue where it interacts with the tissues. Multiple intersecting beams of ultrasound are concentrated on a target. Some phenomena such as attenuation, time delay, scattering or non-linear distortion may occur during the propagation. HIFU simulations inputs are always defined by a known harmonic function of a given frequency, usually a pressure sinusoid of 1-2 MHz.

We can assume that the time-varying quantities such as pressure and velocity at every grid point have a harmonic character as well with only small amplitude and phase changes. They are usually amplitude modulated and they can be composed of the basic and several harmonic frequencies. The number of harmonics depends on simulation grid size and the coefficient of nonlinearity. At this time, we have up to 5-6 harmonics for real simulations which corresponds to HIFU in thermal mode. However, for histotripsy, as many as 50 harmonics may be needed.

A typical sampled pressure signal recorded at one grid point is shown in Figure 1. An output signal, containing, eg. time-variable acoustic pressure, can be divided on three stages according to the amplitude magnitude and its changes. In the first stage, the signal has a character of noise with the amplitude close to zero. The length of this stage depends on the distance of the sampled grid point from the transducer. The second stage has a big amplitude rise (an incoming edge). This transient stage is usually very short, approx 4 % of the total simulation length. The third stage has a relatively steady amplitude. This part of signal is the most important part to calculate heat deposition. The steady stage usually starts at the moment when all waves emitted from the transducer have arrived at the focus point.

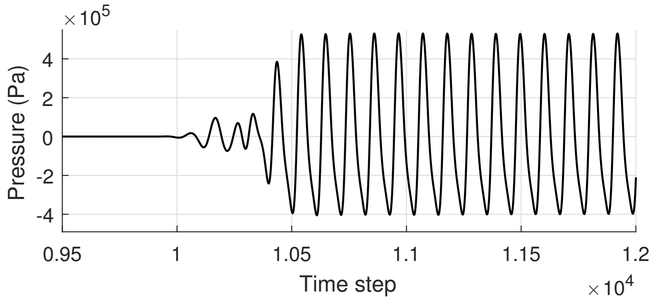

Fig. 1. The illustration of a HIFU simulation signal at one point in 3D space.

The proposed approach is to model an output signal such as a decomposition of a 1D signal (one point in 3D space) to sum of overlapped exponential basis multiplied by a window function. Each base is defined by its complex coefficients (amplitude and phase respectively). We decided to use complex exponential basis because such basis can well represent base harmonic function including phase changes as well as its higher harmonics, and window functions whose sum is constant if they are half-overlapped, because of on-the-fly processing by parts of signal.

It is important that the input frequency emitted by the transducer is known and can be used by the compression algorithm. A complex exponential function with the angular frequency ω can be defined for one time step n as

$$f(n) = e^{-jh\omega n} \tag{1}$$

where h is number of harmonic frequency (wave number), that can be generated by nonlinear wave propagation. By multiplying by a window function w (e.g. Triangular window or Hann window) defined as

$$w(n,a) \begin{cases} 0 & ad > n > ad + 2d \\ w_0(n - ad) & otherwise \end{cases} \tag{2}$$

where d is half-width of the window, we obtain a linearly independent sliding window basis of width $2d$

$$b(n,a) = f(n) w_a(n,a) \tag{3}$$

where a is the basis (frame) index. The half-width d should be an integer multiply of the input period $(1/f)$ because typically leads to more precise results. The whole reconstructed signal can be expressed as

$$s(n) = \sum_{i=0}^{N/d} b(n,i)\widehat{k}(i) \tag{4}$$

where N is the length of the signal, I is the number of complex frames $(I = floor(N/d))$, and $\widehat{k}(i)$ are desired complex coefficients.

An example of three half-overlapped windows and the real part sum of window functions with $2d = 4/f$ is shown in Figure 2.

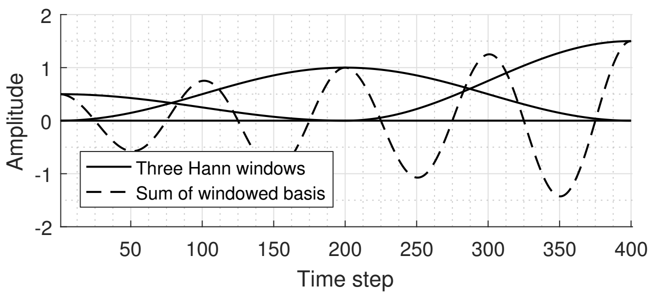

Fig. 2. The illustration of three Hann window bases.

The complex coefficients are computed approximately by correlation (dot product) of the original signal x and the windowed exponential basis for a selected frame i as

$$\widehat{k}(i) = \sum_{i=0}^{N/d} \overline{b(n,i)}x(i)) \tag{5}$$

The coefficients for other harmonic frequencies can be computed independently (as they are linearly independent) and are summed in reconstruction phase.

Every point in 3D space can be processed separately and in parallel within both encoding and decoding phase.

Within the coding phase, two dot products are gradually computed for one time step (one signal sample), because of two overlapped window basis components. The amount of memory c (the number of single precision floating numbers), necessary for computing interim results in one step depends on number of harmonics H and it can be evaluated as

$$c = 4H \tag{6}$$

because we need two complex numbers for every harmonic.

Within the decoding phase, the amount of memory is the same as in the encoding phase We need two complex coefficients for every harmonic frequency, but the decoded samples are computed independently, which can be useful e.q. for fast data visualization.

The compression ratio of the presented method is computed as

$$CR = length(x)/(length(\widehat{k})2H) \tag{7}$$

and it is linearly dependent on width $2d$ of the overlapping basis.

This method of signal modeling yields in a very small error for the steady part of signal where the error depends only on the number of harmonic frequencies considered. In the transient part of the signal the error is bigger.

IV. Implementation

We currently implementation in Matlab intended for experiments with 1D signals only, and in C++ version for processing large 4D datasets. For now, both versions processes the simulation data sequentially and offline by reading HDF5 datasets. Parallel on-the-fly implementation is planned as a future step.

The Matlab version was developed especially for processing 1D data (time series at a single grid point). It mediates simple ways for the compression method debugging, comparison with other coding methods (e.g. audio codecs) and the visualizations of processing steps and results.

The C++ version differs from the Matlab in processing of large amount of 4D data. Individual HDF5 datasets are loaded by 3D blocks depending on the main memory size. Output coefficients are stored into new datasets for every harmonic frequency.

For the moment, both implementations are tested on a desktop computer with 24 GB DDR4 RAM, Intel Core i5-6500 CPU and ASUS Z170 PRO GAMING motherboard. For experiments with extensive HDF5 files, the Salomon cluster with 128 GB DDR4 RAM, 2 $\times$ Intel Xeon E5-2680v3 processors and Lustre shared storage space with maximal theoretical throughput 6 GB/s for one computing node is used. Our source codes are multiplatform. We will provide the link to the source codes on request by email.

We are planning to deploy the compression algorithm in parallel environment during a HIFU simulation. The main advantage will be the possibility to avoid storing the whole output simulation data, that will save much storage space of a lot of time required IO operations.

V. Experiments and results

Two sets of experiments were performed. The first set was focused on testing the compression method on 1D signals while the the second type of experiments were made with large 4D datasets. A Triangular window was used as a window function for all the experiments. The main purposes of the experiments were to get the peak signal-to-noise ratio (PSNR) with respect to the compression ratio (CR), and to compare the proposed method with other compression methods, especially with 1D audio codecs.

For 1D signals Matlab compression tests, 6 different signals were selected from 5 different HDF5 files. We always chose a 1D signal with the maximal absolute amplitude within the whole dataset. The schematic overview of test signals is shown in Table I.

The signals differed mainly in the simulation size, sensor mask size, number of sampled simulation steps, input period and type of the simulation. The sensor mask was always defined within the highly focused HIFU area where the highest amplitude and the most harmonics are observed and was not sparse in these examples. The signal named S15_SC1_ux contains particle velocity for x-axis, other signals contain

TABLE I
THE TYPES OF MATLAB TESTING SIGNALS

Name	Description	Simulation size	Sensor mask size	Time steps	Period
Linear	Linear (1 harmonic), acoustic pressure	256×256×350	55×55×82	3301	15
S15_SC1	Nonlinear (max cca 2 harmonics), acoustic pressure	512×384×384	101×101×101	10105	35
S15_SC1_ux	Nonlinear (max cca 2 harmonics), particle velocity	512×384×384	101×101×101	10105	35
S15_SC5	Nonlinear (max cca 6 harmonics), acoustic pressure	1536×1152×1152	101×101×101	30604	106
Non-realistic	Nonlinear (max cca 6 harmonics), acoustic pressure	192×192×128	172×172×108	1945	65

TABLE II
TESTED CODECS AND THEIR SETTINGS

Codec	Settings
ADPCM	Input: 16-bit WAV, ffmpeg adpcm_ms
FLAC	Bits per sample: 24, lossless
MP3	Bit rate: 320
MP4	Bit rate: 192
OGG	Quality: 100
Opus	Input: 16-bit WAV
HCM	Multiple of overlap size: 1
HCM16bit	Multiple of overlap size: 1, compression output converted to 16-bit values

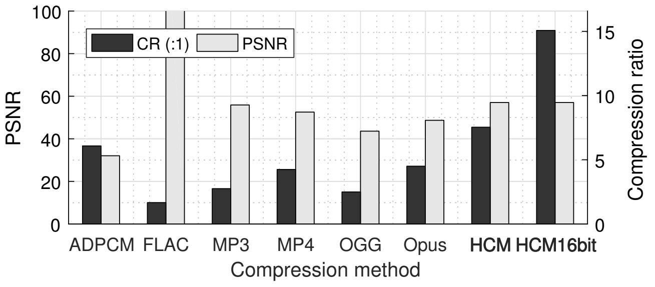

Fig. 3. PSNR and CR for different methods with Linear 1D signal.

TABLE III
RESULTS FOR LINEAR 1D SIGNAL.

	ADPCM	FLAC	MP3	MP4	OGG	Opus	HCM	HCM16bit
CR (:1)	6.1	1.7	2.8	4.2	2.5	4.5	7.5	15.1
Mean error (%)	1.7	0.0	0.0	0.1	0.5	0.2	0.0	0.0
Max error (%)	7.6	0.0	5.8	2.3	1.7	6.5	1.9	1.9
MSE	8.2e+09	2.2e-02	3.4e+07	7.3e+07	5.8e+08	1.8e+08	2.6e+07	2.6e+07
PSNR (dB)	32.0	147.7	55.9	52.6	43.6	48.7	57.0	57.0

acoustic pressure. The signal named Linear is the output of a linear simulation, thus there are no harmonics. The other signals are the outputs from nonlinear simulations and the number of harmonics depends, among other things, on the simulation resolution. Thus, e.g. a realistic simulation that is a representative of the clinical situation with heterogeneous tissue with simulation grid size 1536×1152×1152 contains about 6 significantly strong harmonics (named S15_SC5). The same simulation (named S15_SC1), but with a smaller simulations grid size (512×384×384), has only 2 harmonics. We also generated a smaller simulation non-linear data with higher number of harmonics (named Non-realistic) for testing purposes.

The Matlab compression experiments compare the results of the proposed compression method (HCM) with several audio codecs, specifically with ADPCM (ffmpeg adpcm_ms), FLAC (lossless), MP3, MP4, OGG, Opus and WAV. All of the codecs were set to the best compression quality. The settings of the codecs including the proposed method is shown in Table II. The mentioned coding methods were selected for the comparison only. None of them have the appropriate computational properties for implementation in parallel distributed environment, possibly apart from ADPCM.

Since several audio codecs require the signal to be normalised on $\langle 0, 1\rangle$ interval, we used the maximal absolute signal value for the normalization. In some cases, the input signal had to be first saved as a 16-bit WAV file before the codec could be applied (ADPCM, Opus). This conversion was performed with the use of maximum 16-bit integer value and 16-bit integer cropping.

The Measured PSNR with gained CR for signal named Linear is shown in Figure 3 and complete measured values are in Table III. The peak signal-to-noise ratio reaches comparable values for all methods except ADPCM, which is significantly worse. Our proposed (HCM) method has the best compression ratio.

We can get better compression ratio with the proposed method by setting the multiple of overlap size (MOS) to higher values. PSNR and CR with MOS 1, 2, 3, and 4 are shown in Figure 4. PSNR decreases almost with the third power.

The results for nonlinear simulation signals are slightly different from the Linear signal. In Figure 5, PSNR and CR for S15_SC1 signal is shown. Complete measured values are in Table IV. We measured a little bit worse PSNR for the proposed method and better CR for OGG. It is probably because of more harmonics.

We have achieved similar results in case of S15_SC1_ux signal. This signal contains particle velocity values and has a very small range of values (ca. from -0.4 to 0.4).

In the case of the S15_SC5 signal (bigger simulation

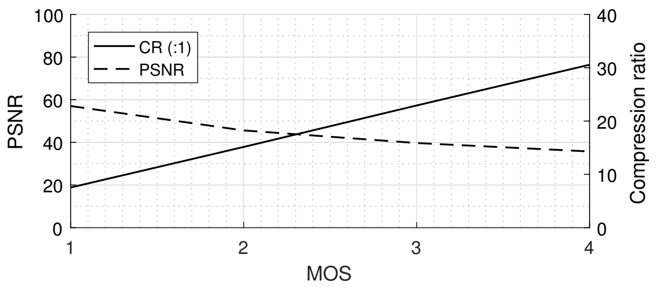

Fig. 4. Different MOS with Linear 1D signal.

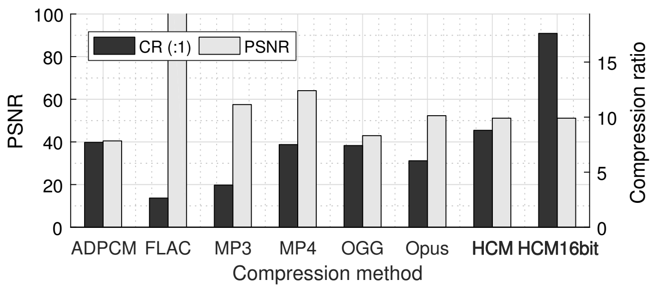

Fig. 5. PSNR and CR for different methods with S15_SC1 1D signal.

grid size and about 6 harmonics), the proposed method gives similar results to the previous cases. ADPCM has comparable PSNR and is slightly worse in CR than the proposed method. OGG has considerably better CR. Very good CR and PSNR gives the MP4 method. These results are shown in Figure 6 and Table V.

A short part of 1D S15_SC5 original signal, signal reconstructed by HCM and their difference with MOS 1 is shown in Figure 7. We can see the maximal error in a part with the very non-steady signal values.

TABLE IV
RESULTS FOR S15_SC1 1D SIGNAL.

	ADPCM	FLAC	MP3	MP4	OGG	Opus	HCM	HCM16bit
CR (:1)	7.7	2.6	3.8	7.5	7.4	6.0	8.8	17.6
Mean error (%)	0.6	0.0	0.0	0.0	0.5	0.1	0.0	0.0
Max error (%)	3.1	0.0	7.7	0.2	1.7	8.4	4.5	4.5
MSE	8.1e+08	1.7e-02	1.6e+07	3.5e+06	4.6e+08	5.3e+07	6.9e+07	6.9e+07
PSNR (dB)	40.5	147.2	57.5	64.0	43.0	52.3	51.1	51.1

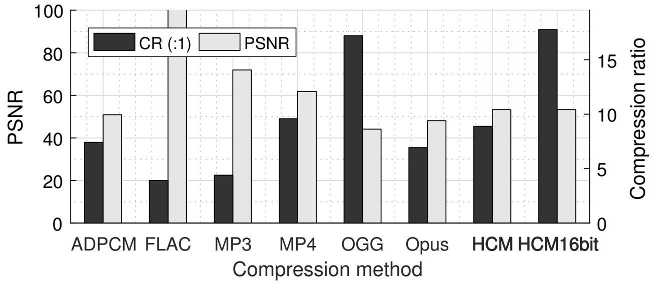

Fig. 6. PSNR and CR for different methods with S15_SC5 1D signal.

TABLE V
RESULTS FOR S15_SC5 1D SIGNAL.

	ADPCM	FLAC	MP3	MP4	OGG	Opus	HCM	HCM16bit
CR (:1)	7.4	3.9	4.4	9.6	17.2	6.9	8.9	17.8
Mean error (%)	0.2	0.0	0.0	0.1	0.4	0.3	0.0	0.0
Max error (%)	1.3	0.0	2.4	0.5	1.6	3.1	3.9	3.9
MSE	1.3e+08	2.2e-02	1.0e+06	1.0e+07	6.1e+08	2.4e+08	7.4e+07	7.4e+07
PSNR (dB)	50.9	148.6	71.9	61.9	44.1	48.1	53.3	53.3

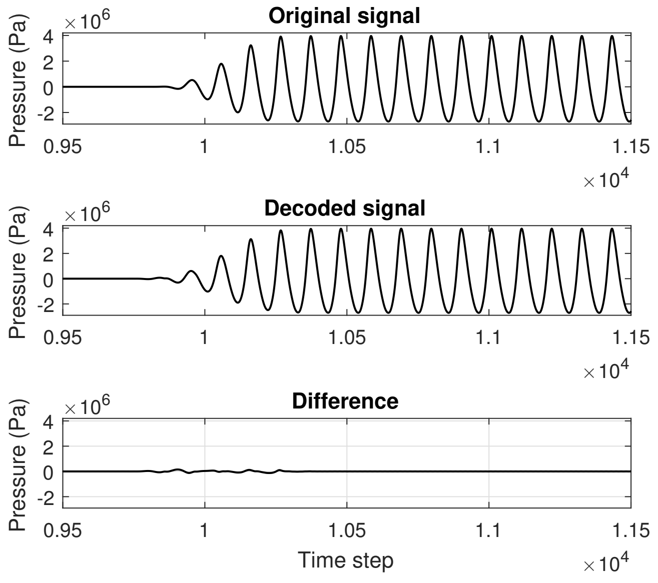

Fig. 7. Illustration of the S15_SC5 1D original, reconstructed and difference signals with MOS equals 1.

The Non-realistic signal testing results are shown in Figure 8. Here, we have got very poor PSNR results compared to other methods. The main reason is input simulation parameters that are not typical of HIFU.

Experiments with higher MOS values with the mentioned non-linear signals S15_SC1, S15_SC1_ux, and S15_SC5 showed similar results as in Linear signal case.

We also made compression experiments with two 4D datasets. The tested signals were S15_SC1 and S15_SC5 and all the points stored in the sensor mask were compressed ($101\times101\times101\times10105$ points for S15_SC1 4D signal and

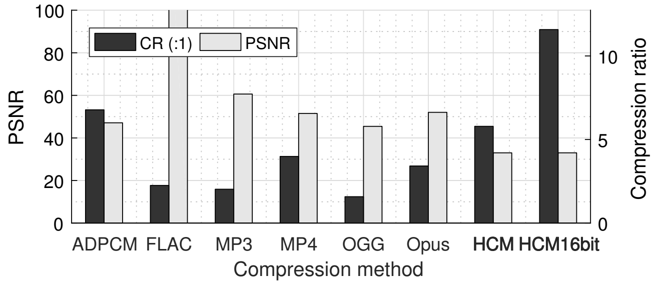

Fig. 8. PSNR and CR for different methods with Non-realistic 1D signal.

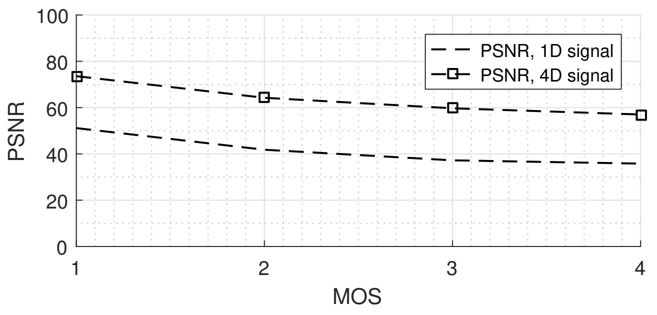

Fig. 9. PSNR for different MOS with S15_SC1 1D and 4D signal.

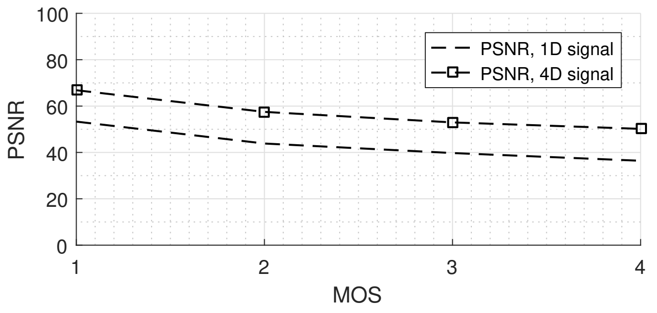

Fig. 10. PSNR for different MOS with S15_SC5 1D and 4D signal.

101×101×101×30604 points for S15_SC5 4D signal). The proposed compression was tested with setting of MOS to values 1, 2, 3, and 4.

We have got interesting results for the S15_SC1 case (Figure 9). The PSNR values for the 4D signal were about 20 dB better than for the 1D signal.

In case of S15_SC5 signal, the results were little bit different (Figure 10). The PSNR values for the 4D signal were about 13 dB better than for the 1D signal. This phenomenon is probably caused by a three times larger simulation grid size in S15_SC5 case, but the same size of sensor mask. We could have made the experiments with a corresponding higher sensor mask size (e.g. 301×301×301) and expect results similar to S15_SC1 case, but we would have needed more than 3 terabytes of data for the testing dataset and a time for offline compression would be enormous.

The overall results indicate useful properties of the proposed method. The maximal errors are occurred in short non-steady parts of signals and PSNR of the whole signals has comparable values with the state-of-the-art audio codecs. The errors in the steady parts are negligible. We can expect better PSNR with higher sensor mask sizes and MOS.

It is important to say that we haven't had any optimization of the basis and window functions yet, that could reduce the errors in non-steady signal parts. We also have not applied any follow-up compression algorithms to computed coefficients, thus higher compression ratios are real.

Conclusions

An efficient compression algorithm for HIFU simulation data has been proposed and the first offline experiments have been performed. We have showed that our method produced very useful results. The important steady parts of the simulation signals are compressed with very small errors (0.1 %) with compression ratios over 80 %. The very short transient parts of signals are compressed with acceptable errors and possible improvements are planned for the future.

The main future work is to apply our compression algorithm to the existing implementation of the k-Wave simulation toolbox, especially to provide the possibility to save storage space and also the time necessary for writing large HDF5 files during simulations. The fast visualizations of the simulation data should be the next possible utilization of compressed data. We also want to use our approach to efficient computations of heat propagation from pressure values without a time-consuming computation of intensity. Important future steps will be the evaluation of applications compressed and decompressed simulations data from the point of view of users (scientists or doctors) and utilization of the computer resources.

Acknowledgment

This work has been supported by the Technology Agency of the Czech Republic (TA CR) Competence Centres project V3C Visual Computing Competence Center (no. TE01020415) and the European Union's Horizon 2020 research and innovation programme H2020 ICT 2016-2017 under grant agreement No 732411 and is an initiative of the Photonics Public Private Partnership.

References

[1] J. Jaros, P. A. Rendell, and E. B. Treeby, "Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound," *The International Journal of High Performance Computing Applications*, vol. 2015, no. 2, pp. 1–19, 2015.

[2] J. Jaros, F. Vaverka, and E. B. Treeby, "Spectral domain decomposition using local fourier basis: Application to ultrasound simulation on a cluster of GPUs," *International Journal of Supercomputing Frontiers and Innovations*, vol. 3, no. 3, pp. 39–54, 2016.

[3] V. Suomi, J. Jaros, B. Treeby, and R. Cleveland, "Nonlinear 3-D simulation of high-intensity focused ultrasound therapy in the kidney," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Aug 2016, pp. 5648–5651.

[4] R. Liu, "Data compression in ultrasound computed tomography," Ph.D. dissertation, 2011.

[5] M. d. A. Freitas, M. R. Jimenez, H. Benincaza, and J. P. von der Weid, "A new lossy compression algorithm for ultrasound signals," in *2008 IEEE Ultrasonics Symposium*, Nov 2008, pp. 1885–1888.

[6] G. E. Blelloch, "Introduction to data compression," *Computer Science Department, Carnegie Mellon University*, 2013.

[7] P.-W. Cheng, C.-C. Shen, and P. C. Li, "Ultrasound RF channel data compression for implementation of a software-based array imaging system," in *2011 IEEE International Ultrasonics Symposium*, Oct 2011, pp. 1423–1426.

[8] S. Di and F. Cappello, "Fast error-bounded lossy HPC data compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 730–739.