information

MDPI

Article

Efficient Low-Resource Compression of HIFU Data

Petr Kleparnik *, David Barina, Pavel Zemcik and Jiri Jaros

Centre of Excellence IT4Innovations, Faculty of Information Technology, Brno University of Technology, Bozetechova 1/2, 612 66 Brno, Czech Republic; ibarina@fit.vutbr.cz (D.B.); zemcik@fit.vutbr.cz (P.Z.); jarosjir@fit.vutbr.cz (J.J.)

* Correspondence: ikleparnik@fit.vutbr.cz

Received: 10 May 2018; Accepted: 24 June 2018; Published: 26 June 2018

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 specialised for HIFU. Our approach is particularly focused on on-the-fly parallel data compression during simulations. The algorithm is able to compress 3D pressure time series of linear and non-linear simulations with very acceptable compression ratios and errors (over 80% of the space can be saved with an acceptable error). The proposed compression enables significant reduction of resources, such as storage space, network bandwidth, CPU time, and so forth, enabling better treatment planning using fast volume data visualisations. The paper describes the proposed method, its experimental evaluation, and comparisons to the state of the arts.

Keywords: compression; ultrasound simulation; high-intensity focused ultrasound; k-Wave toolbox

1. Introduction

The application of high-intensity focused ultrasound (HIFU) in human medicine is an emerging non-invasive perspective therapy. The focused beam of ultrasound, typically generated using a large transducer, is sent into the tissue. The cells of tissue in a localised region are rapidly heated, which causes tissue death while the surrounding tissue is left unharmed. In recent years, many HIFU clinical trials for the treatment of tumours in the prostate, kidney, liver, breast, or brain have been performed. The most important issue is the precise placement of the ultrasound focus and dosage assessment [1]. There are some tissue properties that can significantly distort ultrasound distribution, for example, in the skull. This is the main reason that large-scale numerical HIFU simulations are needed.

However, two key challenges must be addressed. First, it is necessary to create an acoustic and thermal model that is physically complex due to a heterogeneous medium and non-linear wave propagation. Second, the simulations are computationally very intensive and expensive as they must be executed on large domains with billions of grid points [2]. Although the computational requirements were alleviated by a distributed multi-graphics processing unit (GPU) implementation [1,2], the amount of generated data quickly becomes unmanageable; moreover, the input/output (I/O) operations become the dominating factor for the simulation run time [1,3].

Because of large distances travelled by the ultrasound waves relative to the wavelength of the highest harmonic frequency, and in order to obtain 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 reach up to $4096 \times 2048 \times 2048$ samples in 3D space. Usually, the limited number of 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, for example, in thermal modelling [1,3]. The typical size of sampled data for a clinically applicable simulation reaches 0.5 TiB.

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

We completed an experimental implementation of the compression method for offline dataset processing. We performed several experiments on datasets representing clinical situations with heterogeneous tissue. The results showed very promising compression ratios with low data distortion.

The rest of this paper is organised as follows. Section 2 briefly presents the existing compression methods. Section 3 presents the proposed method. The subsequent Sections 4 and 5 discuss our current implementation and the experiments performed. Finally, Section 6 concludes the paper.

2. Related Work

In general, sophisticated compression methods for various application data types exist. However, no method has been tailored for HIFU simulation data. Most of the related papers focus on the compression of data obtained by general ultrasonic imaging.

Three-dimensional ultrasound computed tomography (USCT) data can be reduced with a compression method based on the discrete wavelet transform [4]. Many techniques have been applied to process one-dimensional (1D) ultrasonic signals, for example, matching pursuit, the 1D discrete cosine transform (DCT), and the 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, trace compression, largest variation (LAVA), and linear predictive coding (LPC), or also with the well-known ZIP, JPEG, or MPEG methods [7,8]. Some of the above-mentioned techniques can be applied to HIFU simulation data; however, they are not focused on on-the-fly parallel data compression during large-scale simulations. A relevant overview and comparison of the compression standards for ultrasound imaging can be found in [9]. Unfortunately, all these standards process the data as 2D slices, leading to unsuitability for parallel on-the-fly processing. Plenty of coding methods have been developed to deal with the compression of audio and speech signals. Lossy compression methods provide higher compression ratios at the cost of fidelity. These methods rely on human psychoacoustic models to reduce the fidelity of less audible sounds. In contrast to this, lossless methods produce a representation of the input signal that decompresses to an exact duplicate of the original signal. However, the compression ratios are around 50–75% of the original size. For the purpose of our comparison, we selected several common audio coding methods, implemented in the FFmpeg framework. These are the Advanced Audio Coding (AAC), AC-3, Opus, adaptive differential pulse-code modulation (ADPCM), Free Lossless Audio Codec (FLAC), Apple Lossless Audio Codec (ALAC), and uncompressed pulse-code modulation (PCM). The individual formats are briefly discussed below in this section. The inquiring reader is referred to [10] for further details.

As mentioned earlier, the typical size of the ultrasound data to be compressed reaches about 0.5 TiB. Because the existing audio coding methods were not designed for use in extremely memory-limited environments, they consume the signal samples in frames typically containing thousands of samples. In conjunction with the previous point, such large frame sizes make the common audio codecs unusable for compression of the ultrasound data. Another issue to deal with is the sampling rate. Because the human hearing range is commonly given as 20 to 20,000 Hz, the common audio codecs were designed to process signals of comparable rates (e.g., 44.1, 48, 96, or 192 kHz). However, ultrasound signals are acquired (simulated) at a much higher sampling rate. For instance, the rate of our testing signals was 20 MHz. We were therefore forced to slow down the ultrasound data to normally audible frequencies. We chose the sampling rate of 48 kHz, as this was the greatest common denominator of all the examined audio formats. This step did not involve any subsampling signal. It was merely a matter of instructing codecs. Another technical issue was the appropriate audio bit depth selection. We used 16 bits per sample (bps), as this was the common bit depth supported by all the examined codecs.

AAC is an international standard for lossy audio compression. It was designed to be the successor of the ubiquitous MP3 format. Because AAC generally achieves better sound quality than MP3 at

the same bit rate, we omitted the MP3 format in our comparison. The compression process can be described as follows. First, a modified discrete cosine transform (MDCT) based filter bank is used to decompose the input signal into multiple resolutions. Afterwards, a psychoacoustic model is used in the quantisation stage in order to minimise the audible distortion. The time-domain prediction can be employed in order to take advantage of correlations between multiple resolutions. The format supports arbitrary bit rates.

Dolby AC-3 (also Dolby Digital, ATSC A/52) is a lossy audio compression standard developed by Dolby Laboratories. The decoder has the ability to reproduce various channel configurations from 1 to 5.1 from the common bitstream. During the compression, the input signal is grouped into blocks of 512 samples. However, depending on the nature of the signal, an appropriate length of the subsequent MDCT-based filter bank is selected. The format supports selected bit rates ranging between 32 and 640 kbit/s.

Opus is an open lossy audio compression standard developed by the Xiph.Org Foundation. It is the successor of the older Vorbis and Speex methods. The codec is composed of a layer based on the linear prediction and a layer based on the MDCT, but both layers can be used at the same time (hybrid mode). Opus supports all bit rates from 6 to 510 kbit/s.

ADPCM is a lossy compression format with a single fixed compression ratio of about 4:1. ADPCM compression works by separating the input signal into blocks and predicting its samples on the basis of the previous sample. The predicted value is then adaptively quantised and encoded to nibbles, giving rise to the 4:1 compression ratio (192 kbit/s for 48 kHz sampling rate). A psychoacoustic model is not used at all.

FLAC is an open lossless audio format now covered by the Xiph.org Foundation. FLAC uses linear prediction followed by coding of the residual (Golomb-Rice coding). For music, it was reported from various benchmarks that FLAC typically reduces the input signal size to 50–75% of the original size.

ALAC is the open lossless audio format by Apple. Similarly to FLAC, ALAC uses linear prediction with Golomb-Rice coding of the residual. Additionally, the achievable compression performance is similar to that of FLAC.

As a reference format, an uncompressed signal was used. Technically, this format is referred to as linear PCM. Because we used 16 bps and a 48 kHz sampling rate, the uncompressed signal resulted in a bit rate of 768 kbit/s.

3. Proposed Method

A typical output of the HIFU simulation contains two basic types of data—3D spatial data, for example, maximum pressure at defined grid points, and 4D spatial–temporal data with time-varying pressure series at defined grid points. Considering the 3D datasets, final simulation values across the whole simulation domain, or at least aggregated values (such as the minimum, maximum, or root mean square), have to be stored in permanent storage. In the case of time-varying 4D datasets, an acoustic pressure, acoustic velocity, and intensity have to be stored across the defined set of locations (sensor mask) [11,12].

Our compression method is focused on time-varying signals and is designed for 1D data. Because the shape of the sensor mask can be an arbitrary and sparse set of locations [12], we decided not to deal with spatial coherence. Moreover, by treating every spatial grid point independently, there was no need for additional communication on the distributed clusters. The goals of the compression method are low memory complexity and high processing speed; negligible data distortion is acceptable. These intentions differentiate our method from other state-of-the-art compression techniques.

During HIFU simulation, the source ultrasound signal is emitted from a transducer into the tissue with which it interacts. Multiple intersecting beams of ultrasound are concentrated on the target (focus point). Some phenomena, such as attenuation, time delay, scattering, or non-linear distortion, may occur during the ultrasound propagation [13–15]. The source ultrasound signals are always defined by

a known harmonic function of a given frequency, usually a pressure sinusoid of frequency ranging from 0.5 to 1.5 MHz [15].

We can assume that the time-varying quantities, such as pressure and velocity at every grid point, also have a harmonic character, with only a small amplitude and phase deviations. These quantities are usually amplitude-modulated and can be composed of the fundamental and several harmonic frequencies. The number of harmonics depends on the size of the simulation grid and a factor of non-linearity. Currently, we use up to five or six harmonics for real simulations, which corresponds to the HIFU in thermal mode. However, for histotripsy, as many as 50 harmonics may be needed.

A typical pressure signal recorded at a single grid point is shown in Figure 1. An output signal (containing, e.g., time-variable acoustic pressure) can be subdivided into three stages according to the amplitude magnitude and its changes. During the first stage, the signal resembles noise with an amplitude close to zero. The length (in samples) of this stage depends on the distance of the grid point from the transducer. The second stage can be characterised by a large increase in amplitude (a leading edge). This transient stage is usually very short, for example, around 5% of the total simulation length. The third stage carries a relatively stable amplitude. This part of the signal is the most important part for the calculating of a heat deposition [15]. The stable stage usually starts at the moment when all waves emitted from the transducer arrive at the focus point.

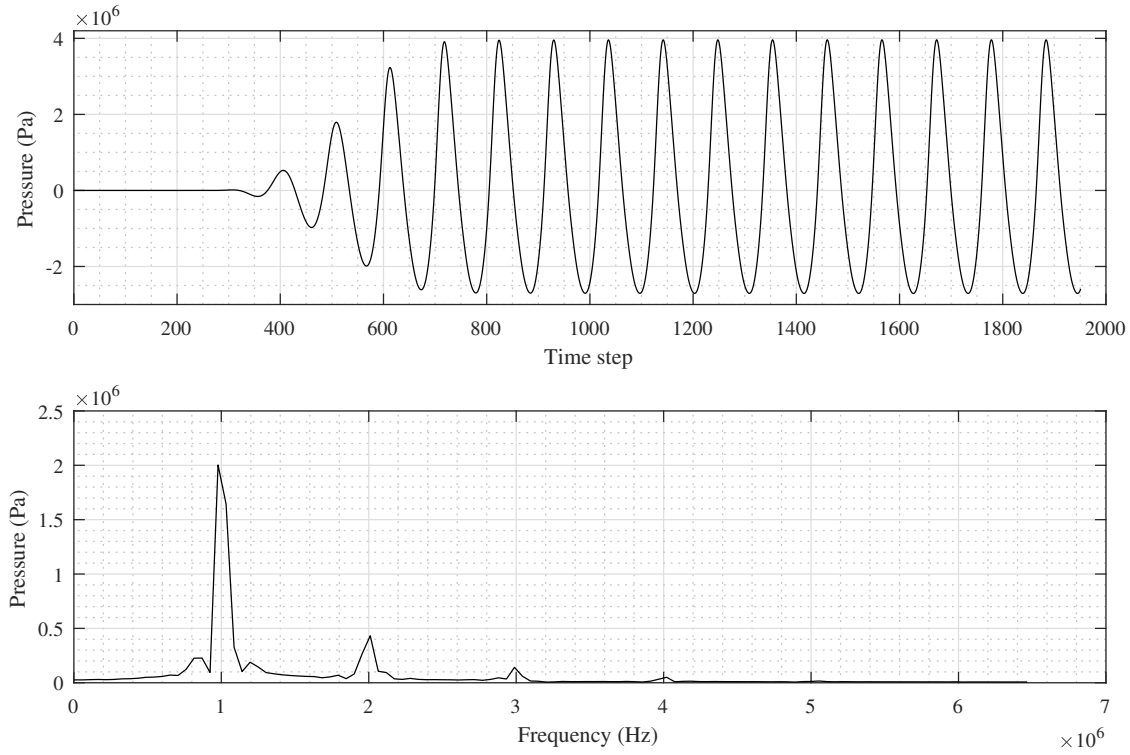


Figure 1. The illustration of a high-intensity focused ultrasound (HIFU) simulation signal at one point in 3D space.

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

It is important that the input frequency emitted by the transducer is known and that it 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

$$f(n) = e^{-\mathrm{i}h\omega n}, \tag{1}$$

where h is the number of the harmonic frequency (wave number), which can be generated by non-linear wave propagation. By multiplying by a shifted window function w defined as

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

where w_0 is a window function (e.g., triangular or Hann window) and d is the half-width of the window, we obtain linearly independent sliding-window basis vectors

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

with $2d$ width, where a is the basis (frame) index. The half-width d should be an integer multiple of the input period $(1/f)$, as such a width typically leads to more precise results. The whole reconstructed signal can then 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 $I = \lfloor N/d \rfloor$, and $\widehat{k}(i)$ are resulting complex coefficients.

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

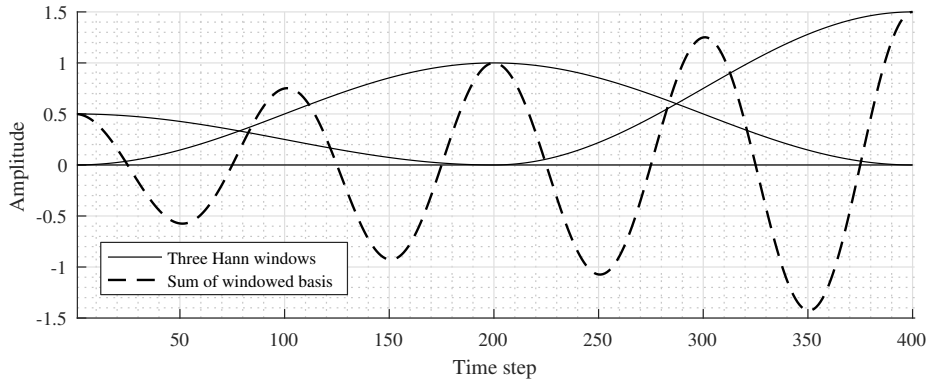


Figure 2. The illustration of three Hann window bases.

The complex coefficients are computed approximately by correlation (dot product) of the original signal x and the windowed exponential basis vector 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 the reconstruction phase. Every point in 3D space can be processed separately and in parallel within both the encoding and decoding phases.

Within the coding phase, two dot products are gradually computed for a single time step (one signal sample) as a result of two overlapped window basis vectors. The number of memory cells c

(single-precision floating-point numbers) required for computing intermediate results in one time step depends on the number of harmonics H, and it can be evaluated as

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

as two complex numbers are needed per every harmonic frequency.

Within the decoding phase, the memory requirements remain the same as in the encoding phase. Two complex coefficients are needed for every harmonic frequency; however, the decoded samples are computed independently, which can be useful, for example, for fast data visualisation.

The compression ratio of the method can be expressed as

$$\text{length}(x)/(\text{length}(\widehat{k})2H). \tag{7}$$

The ratio is proportional to the width $2d$ of the overlapping window bases. This signal approximation method yields a very small distortion for the stable parts of the signal where the distortion depends only on the number of considered harmonic frequencies. The distortion is higher in the transient parts of the signal.

4. Implementation

Currently, two baseline implementations of the proposed methods are available—an implementation in Matlab and FFmpeg for processing 1D signals and a second implementation in C++ for processing large 4D datasets. Both of these versions process the simulation data sequentially and offline by reading datasets in HDF5 format. Another parallel on-the-fly implementation with CUDA and with the OpenMP version is also available.

The Matlab and FFmpeg implementations were developed specifically for processing 1D data (time series at a single grid point). The FFmpeg implementation mediates simple ways for the compression method's debugging, comparison with other coding methods (e.g., audio codecs), and the visualisations of processing steps and results. The C++ implementation differs from the Matlab implementation in the ability to process a large amount of 4D data. Individual HDF5 datasets are loaded by 3D blocks depending on the main memory size. Both implementations were tested on a desktop computer with 24 GiB memory and Intel Core i5-6500 processor. For experiments with extensive HDF5 files, the Salomon cluster with 128 GiB memory, two Intel Xeon E5-2680v3 processors, and Lustre shared storage space with a maximal theoretical throughput of 6 GiB/s for one computing node was used. We will provide the link to cross-platform source code on request by e-mail.

We plan to deploy the compression algorithm in the parallel environment during a HIFU simulation. The main advantage is the possibility to avoid storage of the whole output simulation data. This would save considerable storage space and time as a result of omitted I/O operations. Experiments with the evaluation of computational resources will follow in future work.

5. Experiments and Results

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

For 1D signal compression tests, three different types of signals were selected. We always chose 50 random 1D signals (points within the sensor mask) from the given datasets. The schematic overview of the test signals is shown in Table 1.

Table 1. The types of testing signals. All signals represent an acoustic pressure.

Name	Description	Simulation Size	Sensor Mask Size	Samples	Period
Linear	One harmonic	$256 \times 256 \times 350$	$55 \times 55 \times 82$	3301	15
Non-linear 2	Max. two harmonics	$512 \times 384 \times 384$	$101 \times 101 \times 101$	10,105	35
Non-linear 6	Max. six harmonics	$1536 \times 1152 \times 1152$	$101 \times 101 \times 101$	30,604	106

The signals differ mainly in the simulation size, sensor mask size, number of sampled simulation steps, input period, and type of the simulation. The dense sensor mask was always defined within the highly focused HIFU area in which the highest amplitude and most of the harmonics are observed. All signals contain acoustic pressure stored in 32-bit floating-point format. The dataset referred to as Linear is the output of a linear simulation; thus, there are no harmonics. The other datasets comprise the outputs from non-linear simulations, and the number of harmonics depended, among other factors, on the simulation resolution. For a better connection with reality, a realistic simulation that was representative of the clinical situation with heterogeneous tissue used a simulation grid size of $1536 \times 1152 \times 1152$ and contained about six significantly strong harmonics (referred to as Non-linear 6). The same simulation (Non-linear 2) but with a smaller simulation grid size ($512 \times 384 \times 384$) contained only two harmonics.

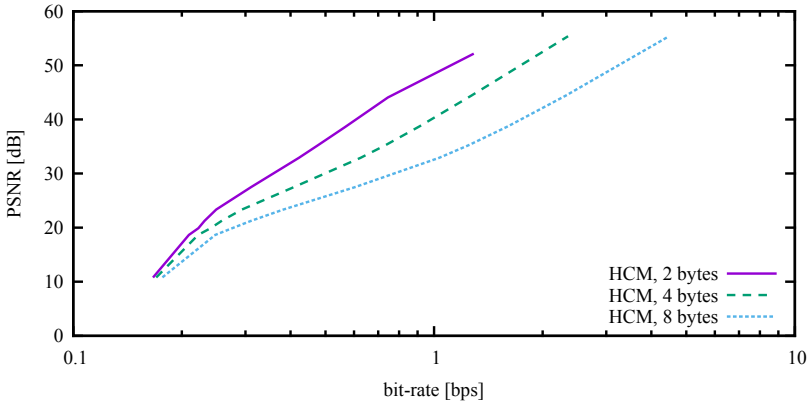
The compression experiments compared the results of the proposed HIFU compression method (HCM) with several audio codecs implemented in the FFmpeg framework (either natively or through an external library), specifically with ADPCM, FLAC, ALAC, AAC, AC-3, Opus, and PCM. The main properties and settings of these codecs, including the proposed method, are shown in Table 2. The above-listed methods were selected for comparison purposes only. Except for PCM, none of them meet the memory requirements needed for implementation in a parallel distributed environment. Because we target a memory-limited environment, we set the frame size to the minimum size supported by the specific sampling rate, sample format, and codec. We note that as a result of the frame-based system, a delay is introduced after the encoding and decoding processes. This delay is shown in the last column of the referenced table.

Table 2. Overview of tested formats and codec settings.

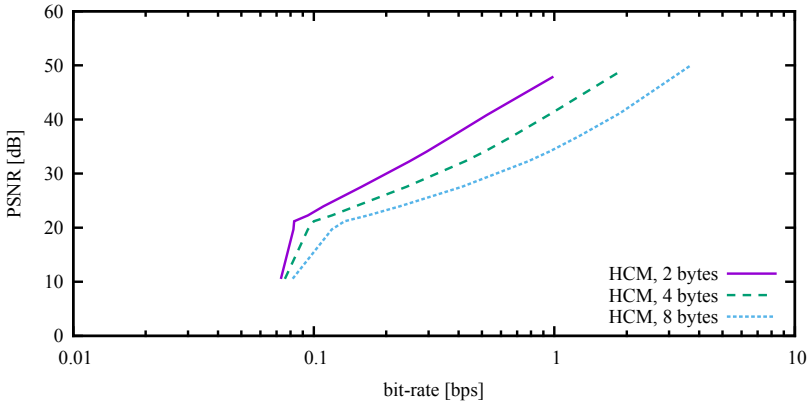
Codec	Long Name	Class	Frame Size	Delay
PCM	Pulse-code modulation	Lossless	1	0
FLAC	Free Lossless Audio Codec	Lossless	16	0
ALAC	Apple Lossless Audio Codec	Lossless	4096	0
ADPCM	Microsoft ADPCM	Lossy	2036	0
Opus	Opus	Lossy	120	120
AAC	Advanced Audio Coding	Lossy	1024	1024
AC-3	Dolby Digital (ATSC A/52)	Lossy	1536	256
HCM	HIFU compression method	Lossy	1	0

Because several audio codecs require the signal to be normalised on the $\langle 0, 1 \rangle$ interval, we used the maximal absolute signal value for the normalisation. In some cases, the input signal had to be stored in 16-bit PCM format before the codec could be applied (ADPCM and Opus). This conversion was performed with the use of the maximum 16-bit integer value.

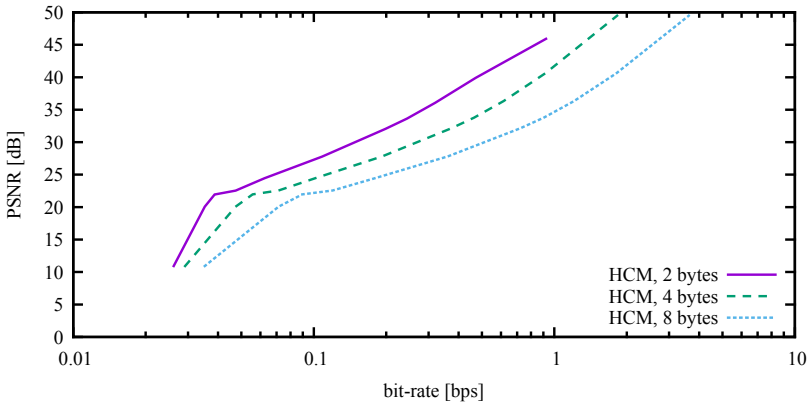
By default, the HCM expects an unknown maximum absolute signal value and therefore uses 32-bit floating-point numbers to store compression coefficients. If the dynamic range of values is known in advance, the method can convert 32-bit floating-point numbers into 16-bit or even 8-bit integers with negligible loss of information (4 or 2 bytes for one complex number). Thus, for normalised signals, another two modifications of the HCM were implemented. As shown in all figures and tables except Figures 3, 7 and 8, the HCM used 8-bit integers for coefficients and thus 2 bytes for one complex number. For a better understanding, a comparison of three types of HCM coding is shown in Figure 3.



(**a**) Linear.



(**b**) Non-linear 2.



(**c**) Non-linear 6.

Figure 3. Different variations of the high-intensity focused ultrasound (HIFU) compression methods (HCMs). The average bit rate and the corresponding peak signal-to-noise ratio (PSNR) for testing datasets: (**a**) Linear, (**b**) Non-linear 2, and (**c**) Non-linear 6.

A dependency of the distortion on the bit rate is shown in Figure 4 for the datasets Linear, Non-linear 2, and Non-linear 6. The lossless methods are represented by a single point, as they do not have, by their nature, the ability to choose any custom bit rate. Except for the ADPCM, the lossy methods are represented as a curve for which the independent variable is the custom bit rate. Although the ADPCM is a lossy method, it has no ability to choose a bit rate. Thus, in this case, the method is also represented by a single point. Looking more closely at the referenced figure, we can see that the

HCM exhibited at least comparable compression performance. More precisely, the PSNR for the Linear case, as shown in Figure 4a, reached comparable values for all lossy methods except the ADPCM and Opus, which had significantly worse results. The proposed HCM reached the lowest bit rate. At its highest bit rate, the HCM also exhibited the best PSNR, which was about 52 dB. It is possible to obtain better bit rates with the proposed method by setting the multiple of overlap size (MOS) to higher values.

The results for the non-linear simulation signals were slightly different from those for the Linear dataset. In Figure 4b, the PSNR and bit rate for the Non-linear 2 dataset are shown. We obtained a slightly worse PSNR, approximately 48 dB, for the proposed HCM, and better values for the other methods. We believe this was due to the presence of more harmonics.

In the case of the Non-linear 6 dataset (larger simulation grid size and about six harmonics), the proposed method gave similar results as for the previous cases. ADPCM exhibited comparable PSNR, and it was only slightly worse in terms of bit rate than the proposed HCM. Moreover, the AAC codec achieved a slightly better bit rate, followed by the competitive AC-3. Further details can be seen in Figure 4c.

Different variations of the HCMs used for the Linear, Non-linear 2, and Non-linear 6 compression datasets can be observed in Figure 3. Importantly, the PSNR reached comparable-quality values for all cases. It can be noticed that the bit rates converged to a single point in the plot—this was caused by the use of a WAVE file header for storing the compressed data. The WAVE format was used only for testing purposes with the FFmpeg tool.

The particular results, including compression ratios, with the PSNRs set as close as possible to 50 dB are listed in Tables 3–5. The values correspond to Figure 5. We note the comparable bit rates of the HCM, AC-3, and AAC.

A short segment of a signal in the 1D Non-linear 6 dataset is shown in Figure 6. The individual sub-figures illustrate the original, HCM-reconstructed, and corresponding difference signals, respectively. All of these correspond to a MOS value set to 1. We note the maximal error in a part with very transient signal values.

Table 3. Results for Linear dataset. Bit rates taken as close as possible to 50 dB.

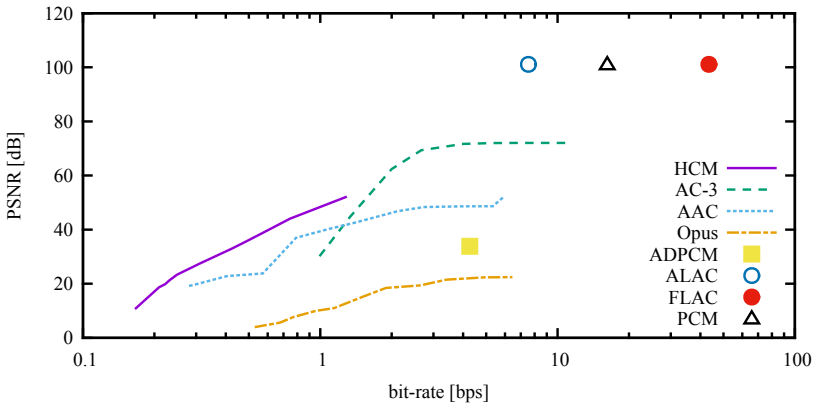
	HCM	**AC-3**	**AAC**	**Opus**	**ADPCM**	**ALAC**	**FLAC**	**PCM**
Bit rate (bps)	1.28	1.33	5.33	6.38	4.27	7.54	43.42	16.19
Compression ratio	25:1	24:1	6:1	5:1	7.5:1	4.2:1	0.7:1	2:1
PSNR (dB)	52.05	44.66	48.67	22.43	33.83	101.11	101.11	101.11

Table 4. Results for Non-linear 2 dataset. Bit rates taken as close as possible to 50 dB.

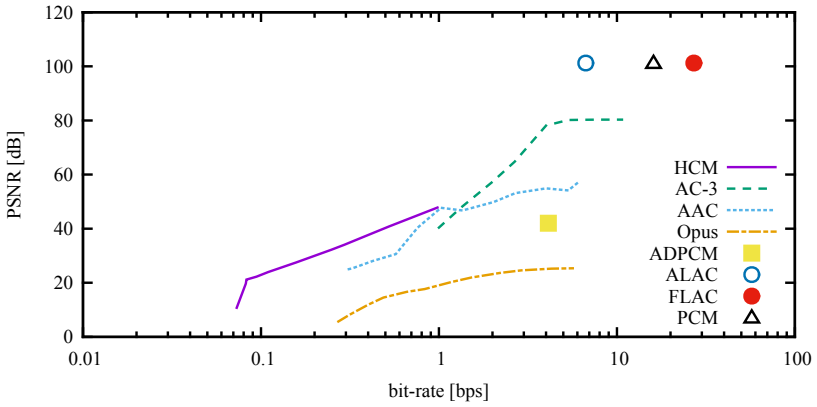
	HCM	**AC-3**	**AAC**	**Opus**	**ADPCM**	**ALAC**	**FLAC**	**PCM**
Bit rate (bps)	0.98	1.33	2.03	5.65	4.12	6.69	27.04	16.06
Compression ratio	32.6:1	24:1	15.8:1	5.7:1	7.8:1	4.8:1	1.2:1	2:1
PSNR (dB)	47.81	47.87	49.85	25.35	42.03	101.23	101.23	101.23

Table 5. Results for Non-linear 6 dataset. Bit rates taken as close as possible to 50 dB.

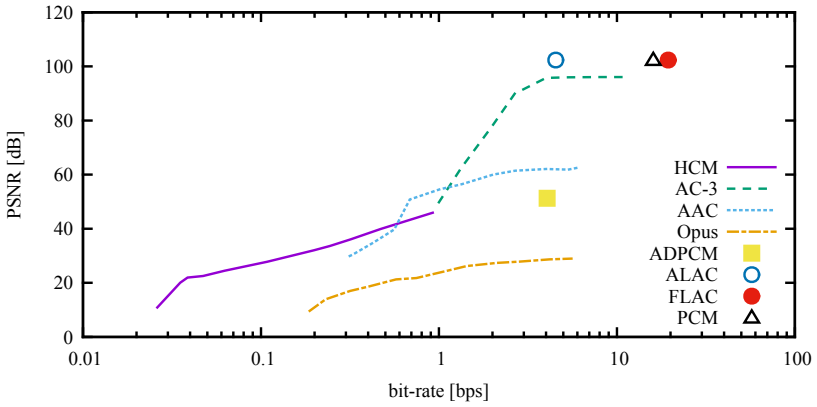
	HCM	**AC-3**	**AAC**	**Opus**	**ADPCM**	**ALAC**	**FLAC**	**PCM**
Bit rate (bps)	0.92	1.00	0.69	5.55	4.05	4.53	19.41	16.02
Compression ratio	34.6:1	32:1	46.7:1	5.8:1	7.9:1	7.1:1	1.6:1	2:1
PSNR (dB)	45.92	49.80	50.82	28.94	51.28	102.34	102.34	102.34



(**a**) Linear.



(**b**) Non-linear 2.



(**c**) Non-linear 6.

Figure 4. A relationship between the average bit rate and the corresponding peak signal-to-noise ratio (PSNR) for testing datasets: (**a**) Linear with one harmonic, (**b**) Non-linear 2 with two harmonics, and (**c**) Non-linear 6 with six harmonics.

We also conducted compression experiments with two 4D datasets. The tested signals were chosen from Non-linear 2 and Non-linear 6, and all the points stored in the sensor mask were compressed ($101 \times 101 \times 101 \times 10{,}105$ points for Non-linear 2 4D signal and $101 \times 101 \times 101 \times 30{,}604$ points for Non-linear 6 4D signal). The proposed compression method was tested with the setting of the MOS to values of 1, 2, 3, and 4.

We obtained interesting results for the Non-linear 2 case (Figure 7). The PSNR values for the 4D dataset were approximately 20 dB better than for the 1D signals.

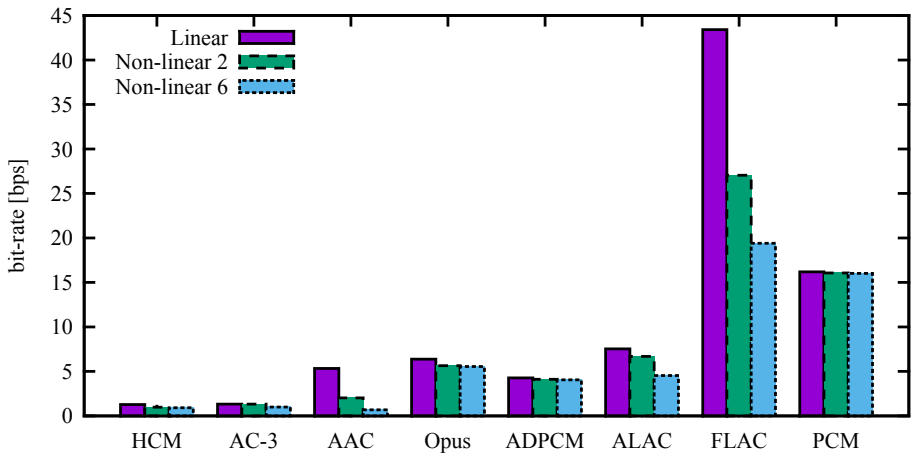


Figure 5. Average bit rates for the peak signal-to-noise ratio (PSNR) as close as possible to 50 dB for three testing datasets.

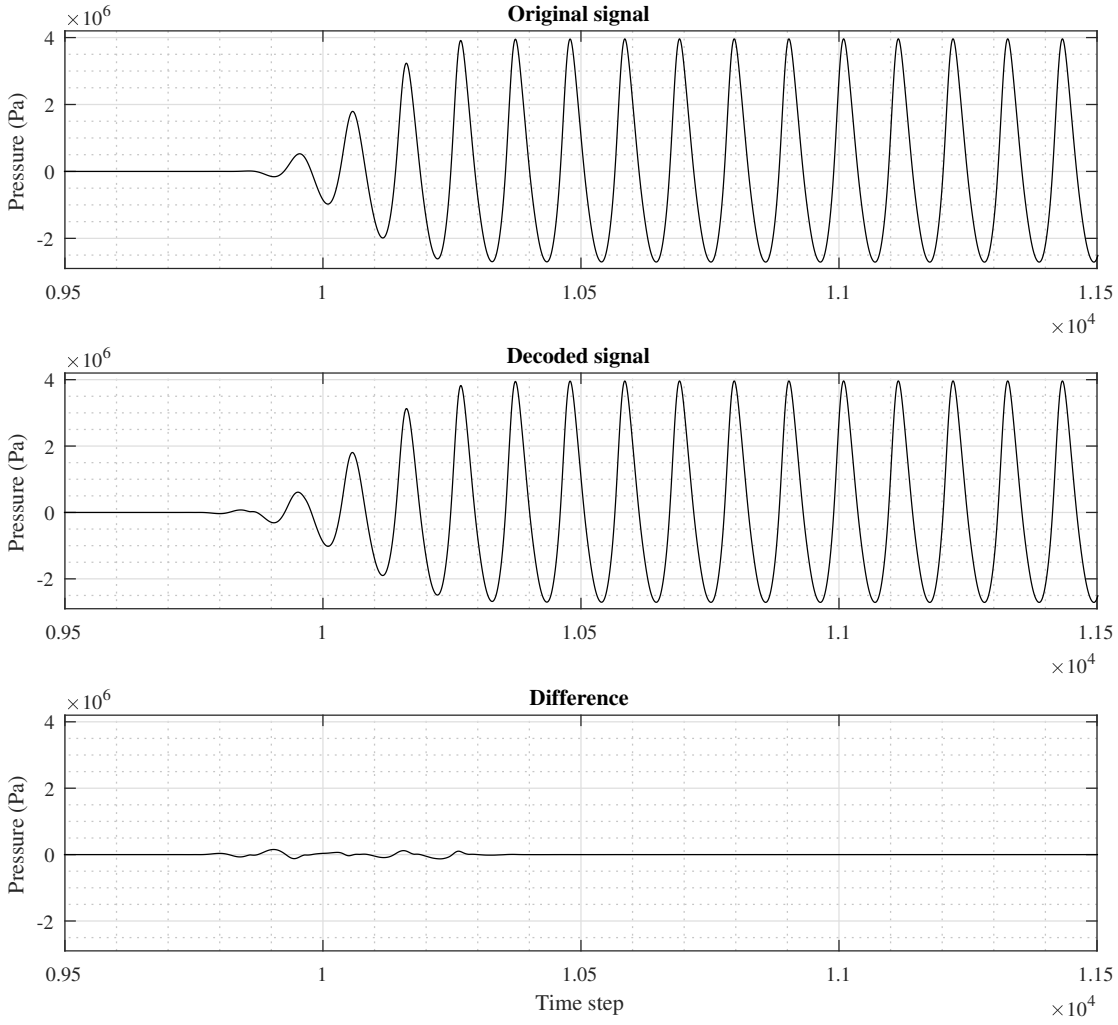


Figure 6. Illustration of the Non-linear 6 original, reconstructed, and difference signals with multiple of overlap size (MOS) equal to 1.

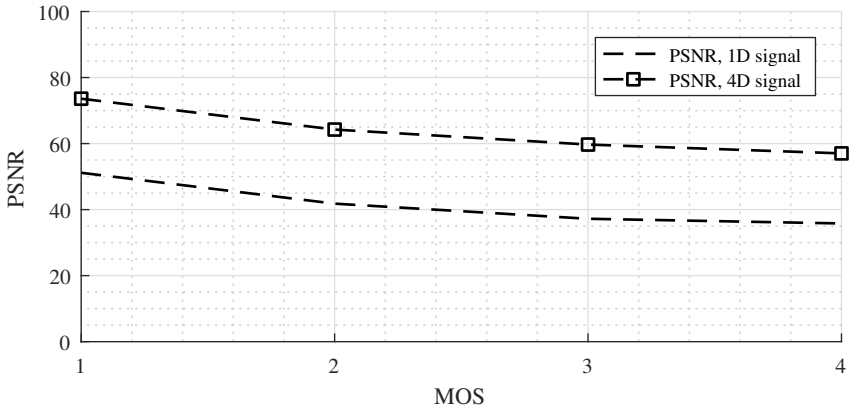


Figure 7. Peak signal-to-noise ratios (PSNRs) for different multiples of overlap size (MOSs) with Non-linear 2 1D and 4D signals.

In the case of the Non-linear 6 signal, the results were slightly different (Figure 8). The PSNR values for the 4D signal were about 13 dB better than for the 1D signal. This phenomenon was likely caused by a 3-fold larger simulation grid size in the Non-linear 6 case but the same size of the sensor mask. We could conduct experiments with a corresponding higher sensor mask size (e.g., $301 \times 301 \times 301$) and expect results similar to those of the Non-linear 2 case, but more than 3 TB of data would be need for the testing dataset, and the time for offline compression would be enormous.

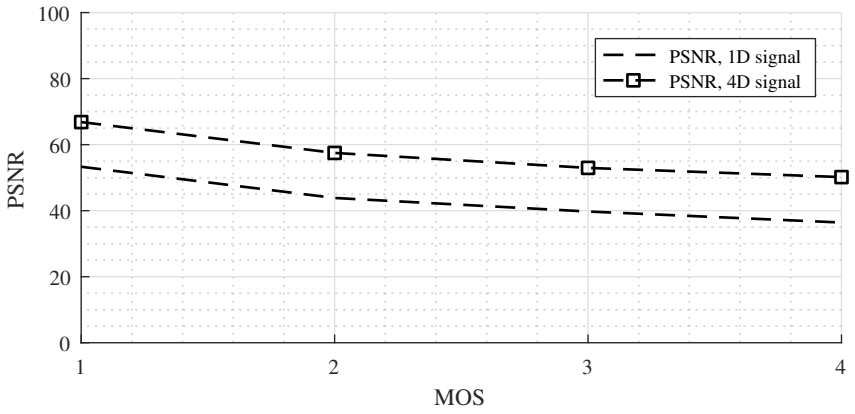


Figure 8. Peak signal-to-noise ratio (PSNR) for different multiples of overlap size (MOSs) with Non-linear 6 1D and 4D signals.

The overall results indicate useful properties of the proposed method. The PSNRs of all the signals had values comparable with those of the state-of-the-art audio codecs. The maximum errors occurred only in short transient parts of the signals. The errors in the stable segments were negligible. We expect better PSNRs with larger sensor masks and smaller MOS values.

In further work, we will attempt to further optimise the basis and window functions so that the errors in transient signal segments will possibly be reduced. We also plan to apply follow-up compression algorithms to further compress the resulting coefficients so that higher compression ratios are achieved.

6. Conclusions

An efficient compression algorithm for HIFU simulation data is proposed, and offline experiments to evaluate it were performed. We have shown that our method produces very useful results. The important stable parts of the simulation signals are compressed with very small distortion (0.1%) at

compression ratios over 80%. The very short transient parts of signals are compressed with acceptable errors, and possible improvements are planned for the future.

In future work, we plan to implement the proposed compression algorithm into the existing implementation of the k-Wave simulation toolbox [11], particularly for the possibility of saving storage space as well as the time necessary for storing large HDF5 files during simulations. Fast visualisations of the simulation data should be the next possible utilisation of the compressed data. We also wish to use our approach to allow efficient computations of heat propagation from pressure values without the time-consuming computation of intensity. Important future steps also lie in an exhaustive evaluation of applications from the perspective of end users and utilisation of the computer resources.

Author Contributions: Conceptualisation: P.K. and P.Z.; data curation: J.J.; investigation: P.K. and D.B.; methodology: P.K. and P.Z.; resources: J.J.; software: P.K. and D.B.; supervision: P.Z. and J.J.; visualisation: P.K. and D.B.; writing—original draft: P.K. and D.B.

Funding: 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); the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); Project IT4Innovations Excellence in Science (LQ1602); and the European Union's Horizon 2020 Research and Innovation Programme H2020 ICT 2016-2017 under Grant Agreement No. 732411. It is an initiative of the Photonics Public Private Partnership.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

1D, 2D, 3D, 4D	One, two, three, or four-dimensional
bps	Bits per sample
CUDA	Compute Unified Device Architecture, a parallel-computing platform
GiB	Gibibyte (2^{30} bytes)
GPU	Graphics processing unit
HCM	HIFU compression method (proposed in this paper)
HDF5	Data model, library, and file format for storing and managing data
HIFU	High-intensity focused ultrasound
I/O	Input/output
MDCT	Modified discrete cosine transform
MOS	Multiple of overlap size
PSNR	Peak signal-to-noise ratio
WAVE	Waveform Audio File Format

References

1. Jaros, J.; Rendell, A.P.; Treeby, B.E. Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound. *Int. J. High Perform. Comput. Appl.* **2015**, *30*, 137–155. [CrossRef]
2. Jaros, J.; Vaverka, F.; Treeby, B.E. Spectral Domain Decomposition Using Local Fourier Basis: Application to Ultrasound Simulation on a Cluster of GPUs. *Int. J. Supercomput. Front. Innov.* **2016**, *3*, 39–54.
3. Suomi, V.; Jaros, J.; Treeby, B.E.; Cleveland, R. Nonlinear 3-D simulation of high-intensity focused ultrasound therapy in the Kidney. In Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Orlando, FL, USA, 16–20 August 2016; pp. 5648–5651.
4. Liu, R. Data Compression in Ultrasound Computed Tomography. Ph.D. Thesis, Karlsruhe Institute of Technology, Karlsruhe, Germany, 2011.
5. Freitas, M.D.A.; Jimenez, M.R.; Benincaza, H.; von der Weid, J.P. A new lossy compression algorithm for ultrasound signals. In Proceedings of the IEEE Ultrasonics Symposium, Beijing, China, 2–5 November 2008; pp. 1885–1888.
6. Blelloch, G.E. *Introduction to Data Compression*; Computer Science Department, Carnegie Mellon University: Pittsburgh, PA, USA, 2013.

7. Cheng, P.W.; Shen, C.C.; Li, P.C. Ultrasound RF channel data compression for implementation of a software-based array imaging system. In Proceedings of the IEEE International Ultrasonics Symposium, Orlando, FL, USA, 18–21 October 2011; pp. 1423–1426.
8. Di, S.; Cappello, F. Fast Error-Bounded Lossy HPC Data Compression with SZ. In Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS), Chicago, IL, USA, 23–27 May 2016; pp. 730–739.
9. Liu, F.; Hernandez-Cabronero, M.; Sanchez, V.; Marcellin, M.W.; Bilgin, A. The Current Role of Image Compression Standards in Medical Imaging. *Information* **2017**, *8*, 131, doi:10.3390/info8040131. [CrossRef]
10. Bosi, M.; Goldberg, R.E. *Introduction to Digital Audio Coding and Standards*; Springer Science+Business Media, Kluwer Academic Publishers: Norwell, MA, USA, 2002.
11. Treeby, B.E.; Cox, B.T. k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. *J. Biomed. Opt.* **2010**, *15*, 021314, doi:10.1117/1.3360308. [CrossRef] [PubMed]
12. Robertson, J.L.B.; Cox, B.T.; Jaros, J.; Treeby, B.E. Accurate simulation of transcranial ultrasound propagation for ultrasonic neuromodulation and stimulation. *J. Acoust. Soc. Am.* **2017**, *141*, 1726–1738, doi:10.1121/1.4976339. [CrossRef] [PubMed]
13. Georgiou, P.; Jaros, J.; Payne, H.; Allen, C.; Gibson, E.; Barratt, D.; Treeby, E.B. Beam Distortion Due to Gold Fiducial Markers During Salvage High-Intensity Focused Ultrasound in the Prostate. *J. Med. Phys.* **2017**, *44*, 679–693. [CrossRef] [PubMed]
14. Bakaric, M.; Martin, E.; Georgiou, P.S.; Cox, B.T.; Payne, H.; Treeby, B.E. Experimental study of beam distortion due to fiducial markers during salvage HIFU in the prostate. *J. Ther. Ultrasound* **2018**, *6*, 1. doi:10.1186/s40349-018-0109-3. [CrossRef] [PubMed]
15. Suomi, V.; Jaros, J.; Treeby, B.; Cleveland, R.O. Full Modeling of High-Intensity Focused Ultrasound and Thermal Heating in the Kidney Using Realistic Patient Models. *IEEE Trans. Biomed. Eng.* **2018**, *65*, 969–979, doi:10.1109/TBME.2017.2732684. [CrossRef] [PubMed]