Spatiotemporal Wavelet Compression for Visualization of Scientific Simulation Data

Shaomeng Li*†, Sudhanshu Sane*, Leigh Orf‡, Pablo Mininni§, John Clyne†, and Hank Childs*
*Department of Computer and Information Science, University of Oregon, Eugene, OR
†Computational Information Systems Lab, National Center for Atmospheric Research, Boulder, CO
‡ Space Science and Engineering Center, University of Wisconsin, Madison, WI
§Department of Physics, University of Buenos Aires, Buenos Aires, Argentina

***Abstract*—Data reduction through compression is emerging as a promising approach to ease I/O costs for simulation codes on supercomputers. Typically, this compression is achieved by techniques that operate on individual time slices. However, as simulation codes advance in time, outputting multiple time slices as they go, the opportunity for compression incorporating the time dimension has not been extensively explored. Moreover, recent supercomputers are increasingly equipped with deeper memory hierarchies, including solid state drives and burst buffers, which creates the opportunity to temporarily store multiple time slices and then apply compression to them all at once, i.e., spatiotemporal compression. This paper explores the benefits of incorporating the time dimension into existing wavelet compression, including studying its key parameters and demonstrating its benefits in three axes: storage, accuracy, and temporal resolution. Our results demonstrate that temporal compression can improve each of these axes, and that the impact on performance for real systems, including tradeoffs in memory usage and execution time, is acceptable. We also demonstrate the benefits of spatiotemporal wavelet compression with real-world visualization use cases and tailored evaluation metrics.**

I. Introduction

Post hoc processing is the dominant paradigm for visualizing data from scientific simulations. In this paradigm, a simulation code periodically outputs data to disk as it advances, and a visualization program loads this data from disk, applies algorithms, and renders the results. Each individual output to disk by a simulation code typically represents the state of the simulation at a moment in time, often referred to as a "time slice." The policy for when to store a time slice to disk varies by simulation code, with examples such as "every 100^{th} cycle," "every five seconds of simulation time," and "every one hour of computation time." Despite different policies, the result is a temporal sampling of the simulation's state, most typically saved at a frequency coarser than the model's internal time increment. The output frequency is often a compromise between providing sufficient temporal coherence in the model outputs for subsequent analysis, and what is possible within the limitations of the compute system's I/O performance and/or storage capacity.

Recent trends, especially on high-performance computers, show that computational power is increasing quickly, whereas disk speeds are increasing relatively slowly. As a result, a simulation's ability to generate data is increasing faster than its ability to store it. There are multiple strategies for addressing this trend, including in situ processing, decreasing temporal frequency, and reducing the size of the data to store. This latter approach can be done via subsetting (only storing pieces of the data, for example certain regions of the mesh or certain variables but not others) or via compression (both lossy and lossless).

With this research, we consider lossy compression. To date, lossy compression has been frequently investigated as a potential means to combat the I/O problem. In a typical case, a compression operator is inserted as the simulation saves its state, such that the operator is applied to single time slices one by one. This compression can be quite effective, since neighboring data points in a mesh are often coherent (very smooth) and many compression operators perform best on coherent data. Simulation data is often temporally coherent as well, and thus our research looks at compression across time as well as space. This alternate approach, referred to as "spatiotemporal compression" in this paper, enables compression operators to exploit temporal coherency while continuing to take advantage of spatial coherency.

Spatiotemporal approaches were not feasible previously on supercomputers, since simulation codes have traditionally been memory-constrained and thus only had room to operate on a single time slice. But emerging changes in architecture now make it reasonable to consider multi-time slice compression, specifically architectures with deeper memory hierarchies including solid state drives and burst buffers. These additional memory hierarchies often have much more storage than primary memory and much faster I/O than the parallel filesystem, creating opportunities to temporarily store multiple time slices as a "window" and apply spatiotemporal compression on this window.

This research explores the benefit of including the time dimension for wavelet-based compression. While other techniques besides wavelets could be considered, we find wavelets to be an excellent operator for our evaluation, since the reference point of spatial-only compression is so well studied. Moreover, the feasibility of wavelet-based spatiotemporal compression was previously impractical due to the relatively wide temporal "window size" required. Our results show that spatiotemporal (4D) wavelet compression is superior to spatial (3D) wavelet compression for each of

the following propositions:

- **P1:** Improve accuracy, while maintaining temporal resolution and storage costs.
- **P2:** Reduce storage costs, while maintaining temporal resolution and accuracy.
- **P3:** Increase temporal resolution, while maintaining storage costs and accuracy.

Our study consists of two phases. First, we evaluate the efficacy of the approach with respect to our three propositions, as well as the performance impacts of operating on multiple time slices jointly. This phase also includes the study of multiple available parameters with particular relevance to wavelet transforms in the time domain, and helps inform their best combinations in practice. Second, we consider real-world visualization use cases which demonstrate how spatiotemporal compression improves analyses by providing more information per byte.

II. Basics of Wavelet Compression

This section provides an abbreviated description of the wavelet transform from the standpoint of compression applications. Excellent and more detailed introductory descriptions are available in [1], [2].

A. One-dimensional Wavelet Transform

It is often advantageous to express a function or signal $x(t)$ as a linear expansion about a set of basis functions. In the case of wavelet bases, such a decomposition is given by:

$$x(t) = \sum_{k\in\mathbb{Z}} \sum_{j\in\mathbb{Z}} a_{j,k}\psi_{j,k}(t), \tag{1}$$

where $a_{j,k}$ are real-valued coefficients, and $\psi_{j,k}(t)$ are wavelet functions typically forming an orthonormal basis. Conceptually, the coefficients, $a_{j,k}$, measure the similarity between the input signal and the basis functions. For finite, discrete $x[n]$ under a "non-expansive" wavelet transform, the number of coefficients $a_{j,k}$ has the same size as $x[n]$. If $x[n]$ exhibits sufficient coherence, and $\psi_{j,k}(t)$ is suitably chosen, then many of the coefficients $a_{j,k}$ will tend toward zero with only a fraction of the remaining non-zero coefficients containing most of the energy or information content of the signal.

B. Wavelet Transforms for Compression

Compression may be achieved by simply discarding and treating a subset of the coefficients as zero in Equation 1. If the discarded coefficients are already zero, then the compression is lossless. However, non-zero coefficients may be discarded as well, resulting in lossy compression, which is the focus of our study.

Certain wavelet kernels (orthogonal and a few biorthogonal) concentrate information into a relatively small number of coefficients, where the information content of each coefficient is proportional to its magnitude. Thus, a sensible approach to compression using wavelet transforms is to retain only coefficients with the largest magnitudes. Efficient coding and storage of these high-information coefficients is a sizable area of research that we do not address here. Interested readers can consult other sources for SPECK [3], SPIHT [4], and EBCOT [5], for example, and their corresponding high-dimensional derivatives.

Most useful wavelet decompositions exhibit "multiresolution"; coefficients parameterized by different values (or levels) of j in Equation 1 capture features at different scales. As a result, more available levels may lead to further information compaction. However, the length of the input data array imposes a limit on the reasonable numbers of levels. We discuss our handling of this limit in Section IV-B.

C. Multi-dimensional Wavelet Compression

The one-dimensional wavelet transform can be extended to multiple dimensions by successively applying a one-dimensional transform along each axis, i.e., output coefficients of one transform become the input of the next transform along a different axis. This practice takes advantage of data coherence along all dimensions, including the temporal dimension. The ordering of axes to apply transforms on may differ though, as discussed in more details in [1], [2]. Nevertheless, after applying multi-dimensional transforms, compression is still achieved by discarding coefficients with the smallest magnitudes.

We note, however, that the amount of coherence is likely to differ along spatial and temporal axes of a scientific data set. In the case of numerical simulations, the spatial grid resolution is typically fixed by the properties of governing numerical equations. Thus the nature of the problem to be solved determines how strongly correlated samples are along a given spatial axis. The simulation scientist, however, may often easily control the output sampling rate — typically much coarser than the internal model time stepping — making the degree of temporal data coherence a parameter that can be readily adjusted.

III. Related Work

A. Spatiotemporal Wavelet Compression

The benefits of wavelet-based, spatiotemporal compression are not well explored for scientific data, especially compared to studies considering only the spatial domain. Some of the earliest work on spatiotemporal wavelets was performed by Villasenor et al. [6] and Trott et al. [7], who both applied a one-dimensional wavelet filter bank over all four dimensions on seismic reflection and fluid dynamics data, respectively. Zeng et al.[8], [9] established the feasibility of spatiotemporal wavelet compression of time-varying echocardiography images. In their earlier work [8], the authors pointed out that the degree of coherence present may differ between dimensions, and thus warranted different handling. Lalgudi et al. [10], [11] evaluated 4D

spatiotemporal compression on functional MRI (fMRI) data obtained as a time series of 3D images of the brain. Wang et at. [12] employed multi-resolution representations from 4D wavelet transforms in their visualization framework for time-varying data.

In all of the preceding work the authors found significant, albeit varying, benefit to 4D spatiotemporal compression over 3D spatial compression. Our work differs in the following ways: 1) our application domain is floating point data arising from numerical simulation; 2) we evaluate information loss with respect to key visualization algorithms, including algorithms that are sensitive to cumulative errors over time; 3) we evaluate the impact of various parameters of wavelet transforms in the context of temporal compression; and 4) we provide an evaluation of how 4D compression works with Solid State Drives (SSDs) now found on HPC systems. All these differences target simulation data on HPC systems, which is a less studied space.

B. Other Temporal Compression Techniques

Motion compensated prediction (MCP) is a family of techniques stemming from video compression, such as the MPEG standard [13]. Researchers have explored the use of MCP on time-varying scientific data, for example, in [14], [15], [16]. In theory, MCP could be useful for Eulerian type flow computations, since the fields move through the grid rather than the grid following the flow. However, it is not well understood how MCP's premise that pixels are moving in groups affect its application on scientific data.

Recent visualization research has also looked at spatiotemporal compression techniques specifically designed for scientific data. Ibarria et al. [15] proposed a "Lorenzo predictor" that operates on arbitrary dimensions, and a compression scheme based on it. Lakshminarasimhan et al. [17] introduced ISABELA as a lossy spatiotemporal compression technique. ISABELA sorts data sequences in a window before performing B-Spline fitting to reduce fitting errors. Lehmann et al. [18] extended ISABELA by using a snapping mechanism to further improve the compression rate in certain cases. Finally, Agranovsky et al. [19] employed a Lagrangian flow-based approach for vector field data that results in reduced file sizes and increased accuracy over spatiotemporal intervals.

C. Burst Buffers

The idea of burst buffers has been proposed to cope with the exploding data pressure from scientific applications. Scientific applications typically have well defined execution phases. For example, they alternate between computation and I/O phases, which results in bursty and non-overlapping I/O. This is one of the fundamental motivations for designing burst buffers. The majority of research on burst buffer usage is recent. Liu et al. [20] designed a simulator of the burst buffer for the IBM Blue Gene/P architecture. Bing et al. [21] characterized output burst absorption on Jaguar and furthered quantitative models of storage system performance behaviors. BurstMem is a prototype burst buffer system developed by Teng Wang et al. [22]. It is built on top of Memcached which is an open source, distributed caching system. BurstMem enhances Memcached by using a log-structured data organization with AVL indexing for fast I/O absorption and low-latency, semantic-rich data retrieval, coordinated data shuffling for efficient data flushing, and CCI-based communication for high-speed data transfer. BurstMem is able to speed up I/O performance of scientific applications by up to $8.5X$ on leading supercomputers. In total, these works demonstrate that the burst buffer is a viable element for spatiotemporal compression, especially since they are being incorporated into many newer supercomputers.

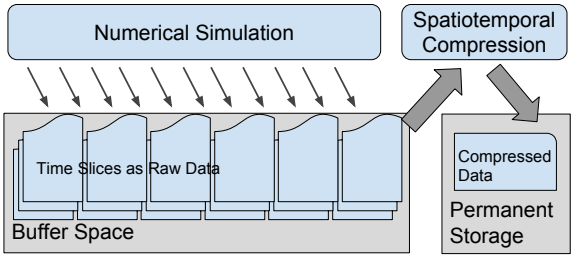


Figure 1: Our spatiotemporal compression work flow. For a window size of T, a simulation code writes T raw time slices to a buffer space. Then, for each variable, a compressor reads in the variable from the T time slices and applies spatiotemporal compression. The resulting compressed data is written to permanent storage. The process then continues for the next temporal window.

IV. Method

A. Processing in Windows

Incorporating spatiotemporal compression into a simulation's output process requires careful consideration of memory. The length of the time dimension can vary widely from tens to thousands depending on the application. For simulations that may already be memory constrained, even with the availability of aforementioned, emerging deep memory hierarchies, retaining large numbers of time steps in memory may be challenging-to-impossible, thus limiting possibilities for temporal wavelet transform.

To address this issue in our implementation, we partition all time slices into "windows" and apply spatiotemporal compression on each window independently. Figure 1 illustrates our work flow.

At its peak, a buffer space will contain $T \times S \times N$ bytes, given a window size T, a single variable of S bytes, and N variables computed by the numerical simulation. We note that S reflects the number of grid points in this equation. Since each of the N variables can be compressed one at

a time, the required available system memory is smaller, specifically $T \times S$. We believe many simulations can spare this much memory provided T is small, since simulations often allocate buffers for temporary usage, and these buffers can be used for our compression.

Wavelet transforms within each window involve two steps: first spatial and second temporal. The forward wavelet transforms, which we have not yet discussed, and the inverse wavelet transform (Equation 1) are implemented as recursively applied filter banks [2]. Each pass of the filter bank corresponds to the multiresolution level parameter j. The coefficients resulting from a single pass of the filter bank are evenly divided into so-called "approximation" and "detail" coefficients. Subsequent filter passes are applied only to the "approximation" coefficients.

In the spatial step we perform what is referred to in wavelet literature as the "non-standard decomposition" [1], [2] to extend the one-dimensional transform to three dimensions. A single pass of the one-dimensional forward wavelet transform is applied first along the X axis, then Y, finally Z. The process then repeats filtering only the approximation coefficients from the previous pass. The filter bank is iterated up to J times, where allowable values of J are determined by properties of the filter bank and signal length described later.

In the second step, coefficients resulting from the first step are first partitioned temporally into chunks of fixed size that we term the *window size*. We then apply a one-dimensional wavelet transform in time at each grid point location for each window. For a N^3 grid, N^3 temporal wavelet transforms are applied per window for example. We note that the relatively smaller window size compared to the lengths of spatial domains limits the levels of wavelet transform that can be applied.

After a time window is spatially and temporally transformed a third step then takes place to compress the coefficients. Given a target compression ratio $n:1, (n>1)$, we find the coefficient with the $(num\_of\_coefficients)/n$ largest magnitude, and discard coefficients with magnitudes smaller than the threshold (i.e., they are treated as zeros in our study) to achieve a $n:1$ compression ratio.

B. Temporal Domain Considerations

We consider two spatiotemporal compression parameters that require extra care: the aforementioned window size and wavelet kernel. The choice of window size is connected to the wavelet kernel choice; they need to be reconsidered under the premise that the window size is limited by the number of time slices that can fit in computer memory.

Larger window sizes allow more levels of wavelet transform to be performed before boundary conditions dominate the calculation. In turn, more levels of wavelet transform are favorable because they can exploit coherence at multiple scales. In our implementation, for a given window size, we set the number of levels J of wavelet transform to perform following the equation:

$$J = \lfloor \log_2 \frac{window\_size}{filter\_size} \rfloor + 1, \quad (2)$$

where filter size is determined by the wavelet kernel. The two parameters, window size and wavelet kernel, interact in such a way that a smaller window size limits the available levels of wavelet transform to perform, while a wavelet kernel with a smaller filter size can potentially increase the level of wavelet transforms possible.

In terms of wavelet kernel choices, the Cohen-Daubechies-Feauveau (CDF) wavelet kernel [23] has proven to be a suitable choice for scientific data compression as well as virtually all compression applications involving aperiodic data such as image compression [24], [5], [25], [26]. The CDF family of *biorthogonal* wavelets are the only compactly supported wavelets besides the Haar that preserve symmetry across scales, and thus allow for a non-expansive transform of finite signals when the signal boundaries are similarly symmetrically extended [27]. A potential difficulty with the CDF 9/7, however, is that its relatively wide filter size (nine) may limit the levels of transforms that are practical with small window sizes. For example, with a window size of ten, CDF 9/7 is only able to do one level of transform. Another kernel from the same wavelet family and having satisfactory compression performance, the CDF 5/3 kernel, has a filter size five, and thus permits two levels of transform. The best practice is thus not trivial. Section V-B1 presents evaluation results on different combinations of wavelet kernels and windows sizes.

V. Our Study

This section explores spatiotemporal wavelet compression, aiming to identify best practices regarding various parameters, as well as to quantify benefits gained from exploiting temporal coherency. The baseline results we use for comparisons are from 3D wavelet compression with CDF 9/7 wavelet kernels, which was found to be a top choice for compression in an earlier study [25]. CDF 9/7 is also what we used for the spatial step of our spatiotemporal wavelet transform (see Section IV-A). Section V-A describes the experiments, Section V-B describes the results, and Section V-C describes performance impacts. Section V-D relates these results back to our three propositions for domain scientists. Finally, Section V-E discusses the limitations we observed regarding spatiotemporal compression.

A. Overview of Experiment Parameters

We varied five parameters to study:

- Wavelet kernel: "CDF 9/7" and "CDF 5/3";
- Window size in the time domain: 10, 20, and 40;
- Data set and variable: 7 variables from 3 simulations;
- Temporal resolution: 3 or 4 options for each variable;
- Compression ratio: 4 or 5 steps from 8:1 to 128:1.

1) Wavelet Kernel and Window Size: Wavelet kernel and window size together determine how many levels of the wavelet transform can be performed in the time dimension, which has a direct impact on the compression result (see Section IV-B). We consider two wavelet kernel candidates on the temporal domain: CDF 9/7 and CDF 5/3 with filter sizes nine and five, respectively. Though CDF 9/7 yields better compression results in most settings, the narrower width of CDF 5/3 is also compelling. We also consider three window sizes: 10, 20, and 40. With these window sizes, CDF 9/7 is able to perform 1, 2, and 3 levels of wavelet transform, respectively, and CDF 5/3 is able to perform one more level at each window size (i.e., 2, 3, and 4 levels) due to a shorter filter size.

2) Temporal Resolution: Compression in the time domain relies on the data coherency between available time slices, which is a direct result of available temporal resolution. A higher temporal resolution provides more data coherency, and thus is more suitable for temporal compression. This experiment quantifies how temporal resolution affects spatiotemporal compression accuracy.

We focus on results from a forced incompressible hydrodynamic turbulent flow simulation from the Ghost simulation code to study temporal resolution in this subsection. Details of the physics of a similar Ghost simulation are given by Mininni et al. [28]. We ran the simulation and saved time slices at a base temporal resolution: every 100^{th} simulation cycle. We denote this base resolution as "1." When experimenting with various temporal resolutions, we reduce this base resolution by using every 200^{th} cycle and using every 400^{th} cycle. We denote the lowered temporal resolution as "1/2" and "1/4," respectively.

These chosen temporal resolutions are required for certain analyses on the finer structures in a turbulent flow. For example, the sampling frequency needs to be high enough that a complete rotation of an eddy is captured by approximately 4 samples, otherwise the eddy may deform substantially. Smaller scale eddies take less time to make a complete rotation, so they require the simulation code to save time slices more frequently to enable a meaningful study.

3) Data Sets and Variables: In addition to the Ghost [29] data set, which simulates a homogeneous turbulent flow, we also used two other simulation outputs in our evaluation: CloverLeaf3D and Tornado. Tornado [30], [31] simulates the dynamics of an F5 tornado, and CloverLeaf3D [32] solves the compressible Euler equations in hydrodynamics settings. We focused our study on temporal regions of interest for the simulations being carried out. In the case of Ghost and Tornado, we used only the later portion of the simulation when interesting phenomena occur for both, which actually imposes greater challenge for compression. For CloverLeaf3D, we used the entire life span of this simulation.

We used two or three different data fields from each of these simulations. For the CloverLeaf3D data sets, we used energy and the X-component of velocity, since these are important, dynamic variables with distinct physical characteristics. For the Ghost and Tornado data sets, we used enstrophy and again the X-component of velocity, for the same reasons. We also used the cloud ratio scalar for the Tornado data set, since this variable determines what the clouds look like to human eyes. In terms of grid size, the two fields from Ghost are on 512^3 grids; the three fields from Tornado are on $490^2 \times 280$ grids with 280 in the Z direction; and CloverLeaf3D X-velocity has a size of 97^3 while CloverLeaf3D energy has a size of 96^3, since the latter is a cell-centered field. We note that the tornado domain analyzed in this paper is significantly smaller than the full model domain, and yet is also typical of what tornado researchers would use. This is because for studies of tornado morphology, features dozens of kilometers away from the tornado are not of primary interest.

We ran our experiments on these variables using multiple temporal resolutions with the same notations described in Section V-A2. The base resolution (res=1) differs from simulation to simulation though: every simulation cycle in CloverLeaf3D; every one simulation second in Tornado; and every 100^{th} simulation cycle in Ghost. We note that the twice coarser resolution (res=1/2) for Tornado data is essentially the common practice of our collaborating scientist, who normally uses every two simulation seconds in his research.

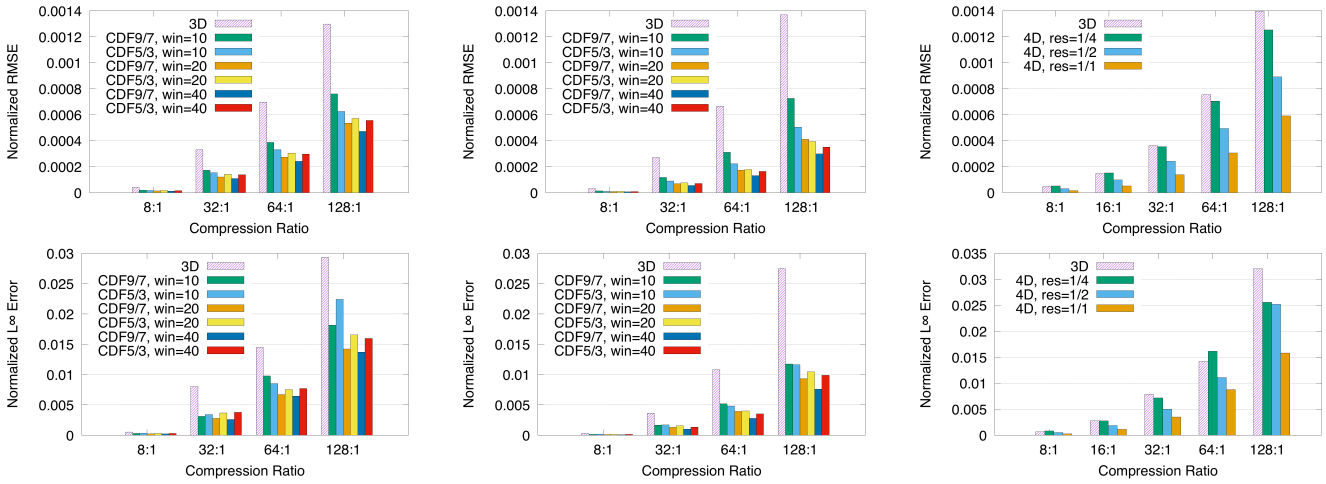
4) Compression Ratios: Our implementation of wavelet compression works by retaining a portion of the wavelet coefficients and discarding the rest. We refer to the parameter that determines the size of the retained portion as compression ratio. With our notation, 8:1 means one eighth of the total coefficients are retained, and so on.

When compressing multiple time slices, the process of discarding coefficients happens on each time slice individually with spatial compression, and on the entire group of time slices with spatiotemporal compression. Given a certain compression ratio, the total number of retained coefficients stays the same no matter spatial or spatiotemporal compression. We use compression ratios 8:1, 16:1, 32:1, 64:1, and 128:1 in most of the following tests.

B. Results

The results considered in this section are error measurements from comparing a wavelet-compressed data set (either 3D or 4D) with its original version in a point-wise fashion. We use two statistical metrics: normalized root mean square error (NRMSE) and normalized L^∞ norm. NRMSE provides an average error across all vertices in the volume a user could expect, while the normalized L^∞ norm captures the largest deviation introduced to a single data point.

To understand the effects of our study parameters, we ran experiments in three phases, varying some factors and



(a) Evaluation on the enstrophy field of the Ghost simulation.

(b) Evaluation on the X-component of velocity of the Ghost simulation.

(c) Evaluation on the enstrophy field of the Ghost simulation.

Figure 2: Normalized Root Mean Square Error (NRMSE, top row) and normalized L^∞-norm (bottom row) evaluation on data from the Ghost simulation. Error values are normalized by the range of the data. Purple bars with line patterns represent spatial-only compression (3D), and bars with solid colors represent spatiotemporal compression (4D) with different parameters combinations for temporal compression. Subfigure (a) and (b) examines two wavelet kernels (CDF9/7, CDF5/3) and three window sizes (win=10, 20, 40). Subfigure (c) examines three temporal resolutions (res=1/4, 1/2, 1/1).

holding the rest constant for each phase. The study results are reported in the following three subsections, and will be revisited as we consider the three propositions for domain scientists in Section V-D.

1) Wavelet Kernel and Window Size: This phase looks at the relationship between wavelet kernels and window sizes, with a goal of finding favorable combinations of the two. The experiments run were on data from the Ghost simulation, with the base temporal resolution.

Figure 2a and 2b plot the evaluation results. Compression ratios are grouped together; spatial-only compression (3D) is leftmost within a group, and to its right are spatiotemporal compression with different parameters. All evaluations clearly show a decrease in error when comparing spatiotemporal to spatial-only compression.

The CDF 9/7 and CDF 5/3 wavelet kernels perform differently with different window sizes. CDF 9/7 yields lower errors than CDF 5/3 with window sizes 20 and 40, but CDF 5/3 is superior with window size 10. This is because, with a window size of 10, the CDF 9/7 kernel only permits one level of wavelet transform, while the CDF 5/3 allows for two levels.

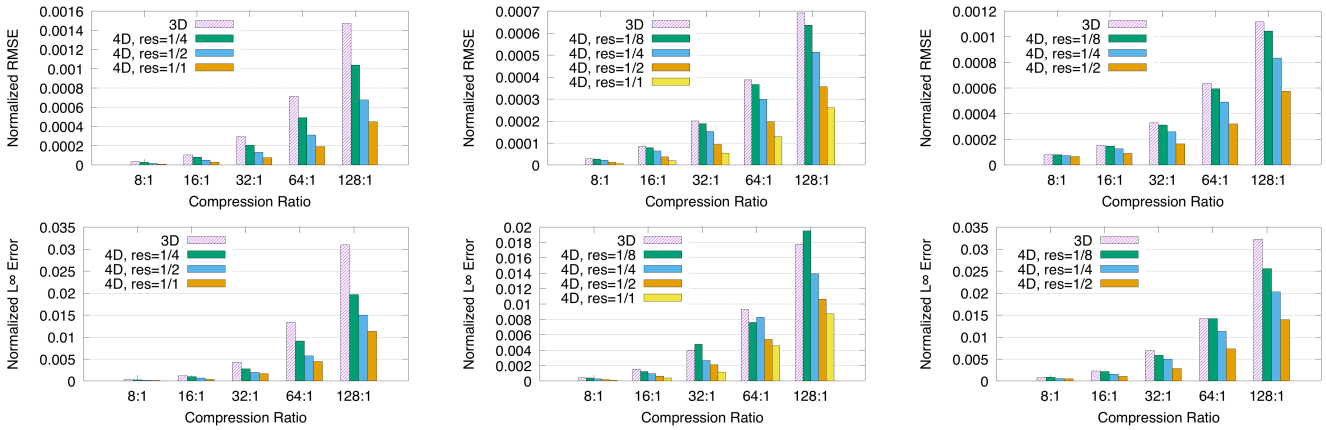
In terms of the window size, a larger window increases accuracy in almost every test. That said, the expected improvement from a larger window size varies. Errors drop more significantly when moving from window size 10 to 20 than moving from 20 to 40. Given the potential limitation from available memory, we consider the combination of windows size 20 and the CDF 9/7 wavelet kernel to be a "sweet-spot." A choice of CDF 5/3 and a window size of 10 for memory sparse situations would also make sense.

2) Temporal Resolution: This phase looked at the effects of temporal resolution. The experiments were run with the "sweet-spot" combination of wavelet kernel and window size from Section V-B1, again on the Ghost simulation.

Figure 2c plots the results from this phase's experiments using NRMSE and normalized L^∞-norm metrics. Its 3D results are from all time slices at the base temporal resolution from the entire studied period of simulation. They serve as a baseline result to compare with. The 4D results are from multiple temporal resolutions (res=1, 1/2, 1/4). The 4D and 3D results cover the same period of simulation in each test.

In terms of results, we see that the benefit of spatiotemporal compression improves as temporal resolution increases. At the finest resolution tested (res=1), spatiotemporal compression leads to substantial decrease in error compared to spatial compression. In most cases, both NRMSE and normalized L^∞-norm are cut by half when incorporating temporal compression. However, at the coarsest resolution tested (res=1/4), temporal compression brings modest benefit. In fact, at the compression ratios of 8:1 and 64:1, normalized L^∞-norm is even higher than the 3D baseline results. We believe this is partially due to a lack of temporal coherence, and also partially due to the nature of L^∞-norm being more sensitive to single extreme values.

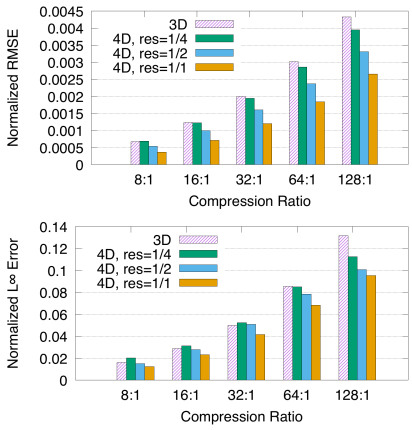
3) Results on Multiple Data Sets: This final phase of the initial study looked at the effectiveness of spatiotemporal compression on multiple data sets. The experiments were



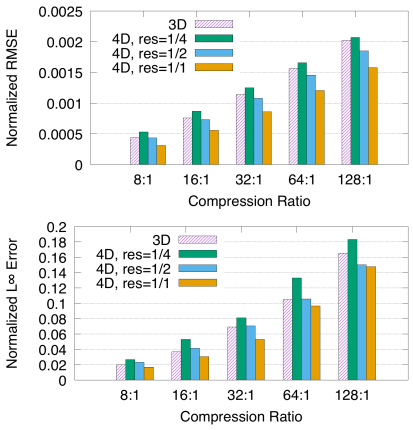
(a) Evaluation on the X-component of velocity from the Ghost simulation.

(b) Evaluation on the X-component of velocity from the CloverLeaf3D simulation.

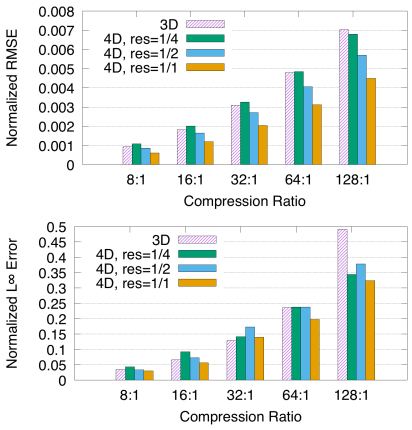
(c) Evaluation on the energy field from the CloverLeaf3D simulation.



(d) Evaluation on the X-component of velocity from the Tornado simulation.



(e) Evaluation on the enstrophy from the Tornado simulation.



(f) Evaluation on the cloud ratio field from the Tornado simulation.

Figure 3: Normalized Root Mean Square Error (NRMSE, top of each subfigure) and normalized L^∞-norm (bottom of each subfigure) evaluation on data sets and variables described in Section V-B3. Error values are normalized by the range of the data. The purple bar with line patterns represents only spatial compression (3D), and the bars filled with solid colors represent spatiotemporal compression (4D) with different temporal resolutions.

again run with the "sweet-spot" combination of wavelet kernel and window size from Section V-B1. Varying in this phase were both data set (see Section V-A3) and temporal resolution (see Section V-A2).

Figure 3 plots the results from this phase's experiments using NRMSE and normalized L^∞-norm metrics. These results confirm the effectiveness of spatiotemporal compression in most test cases, but also show that the amount of benefit relies on the temporal frequency of the data.

C. Performance Impacts

Table I reports on performance impacts for spatiotemporal compression, as well as measurements for spatial-only and no compression. The column "Buffer W+R" indicates writing and reading time spent on the buffer space, "Perm. Write" indicates writing time spent on the permanent storage, and "Total I/O" indicates the sum of the buffer and permanent I/O costs. "Comp. Time" provides the computational cost. Our test system was a compute node with 2 Xeon CPUs at 3.2GHz (16 cores in total), 256GB main memory, and a 2TB SSD serving as a buffer space. The test data is 20 time slices of the enstrophy field from the Ghost simulation at the

Table I: Performance impacts of spatiotemporal compression (4D) and spatial-only compression (3D) compared to no compression (Raw). Error values are indicated by NRMSE that is reported in Figure 2c.

Tech.	Buffer W+R	Perm. Write	Total I/O	File Size	Comp. Time	Error
4D	6.78+6.5s	1.20s	14.48s	625MB	57.49s	5.18e-5
3D	0	1.20s	1.20s	625MB	55.34s	1.47e-4
Raw	0	18.90s	18.90s	10GB	0	0

base temporal resolution (res=1/1). With each time slice at a 512^3 resolution, the total data size is approximately 10GB in raw format, and 625MB after a 16:1 compression (see the "File Size" column). Spatiotemporal compression here used the "sweet-spot" settings. Finally, note that for a specific grid resolution and number of time slices, the buffer I/O and computation cost numbers are independent of the data set and compression ratio — meaning that the results are applicable to data sets aside from Ghost and compression ratios aside from 16:1.

Compared to spatial-only compression, spatiotemporal compression introduces extra I/O time for buffer operations, and a modest amount of extra computation. In return, it encodes more information per byte. Compared to no compression, spatiotemporal compression introduces reconstruction errors and additional computational burden. In return, it significantly reduces the size of data to save on permanent storage, and also saves on total I/O time (14.48s, compared to 18.90s).

D. Examining the Three Propositions

Section 1 introduced three propositions. These propositions are related, in that spatiotemporal compression provides more information per byte. That said, they attract domain scientists for different reasons. We examine each of these propositions separately here.

P1: *improve accuracy, while maintaining temporal resolution and storage costs.* **P1** is supported by Figure 3. For example, take the grouping for 128:1 compression in Subfigure 3a. In this grouping, the NRMSE for 3D compression is 1.47e-3, while 4D compression with sparse temporal sampling has NRMSE of 1.04e-3, and dense temporal sampling has NRMSE of 4.50e-4. In this case, then, 4D compression is anywhere from 40% more accurate to 200% more accurate for the same storage cost. The rest of the figure provides similar results, except for the Tornado data set, which shows that the most sparse temporal sampling sometimes leads to slightly worse results (there is more discussion on the effect of temporal coherence in Section V-E).

P2: *reduce storage costs, while maintaining temporal resolution and accuracy.* **P2** is also supported by Figure 3. Again consider an example from Subfigure 3a. The NRMSE with 64:1 compression with 3D wavelets is comparable to the error with 128:1 compression with 4D wavelets with "1/2" temporal resolution. In this case, a domain scientist could maintain accuracy and temporal resolution, but use half the storage. Similar examples are visible throughout the table. However, the table is oriented around powers-of-two compression ratios, and **P2** does not always hold for $2X$ reductions in storage. Sticking with the example from Subfigure 3a, the NRMSE with 64:1 compression with 3D wavelets is more accurate than the 128:1 compression with "1/4" temporal resolution, which is coarser. In this case where time slices are sampled less frequently, a 2X reduction in storage was not obtained, but a smaller reduction (such as $1.5X$) likely would be possible.

P3: *increase temporal resolution, while maintaining storage costs and accuracy.* Proposition **P3** follows directly from **P2**: if it is possible to reduce storage costs and maintain accuracy, then it would also be possible to use the difference in storage to increase the temporal resolution. Revisiting the comparison between 3D+64:1 and 4D+128:1, a domain scientist could, instead of halving storage costs, opt to keep storage costs constant and double temporal resolution.

E. Discussions and Limitations

While we saw "factor of two" benefit in many cases (compared to spatial compression alone), other cases were below this amount. We believe limitations stem from an inadequate amount of temporal coherence between time slices to achieve a good compression. On the one hand, the physical model and simulation implementation dictate the amount of coherence in the spatial domains. This means errors are lower in some applications, but higher in others. On the other hand, the output data frequency also plays an important role in temporal coherence. This means for the same application, spatiotemporal compression is more beneficial if time slices are sampled more frequently. Our tests across multiple data sets exhibit these limitations.

Among three tested simulations, we notice that Ghost and CloverLeaf3D have significantly less errors with the same spatial compression settings. For example, looking at the X-velocity fields from three simulations, (Subfigure 3a, 3b, and 3d), both Ghost and CloverLeaf3D have NRMSE less than 5e-5 at 8:1 ratio with spatial compression, but Tornado has NRMSE greater than 5e-4. That is one order of magnitude difference. Tornado has significantly larger normalized L^∞ norm values at all spatial compression cases as well. Since there is no temporal compression involved, we observe that it is the characteristic of the Tornado simulation that less coherence is present. In this case, wavelet-based compression techniques perform more poorly.

The accuracy increase from spatial to spatiotemporal compression also differs in the three simulations. Ghost and CloverLeaf3D see more than $2X$ accuracy increase (less than half of the error) at each compression level with the base temporal resolution (res=1/1). However, none of the three fields from Tornado see this amount of accuracy increase. We believe that this difference in accuracy gain is due to the difference of available temporal coherence: Ghost and CloverLeaf3D had enough temporal coherence to demonstrate a $2X$ accuracy gain, while Tornado was too sparse to do so. In fact, Subfigure 3e and 3f reveals that 4D compression even increases NRMSE errors at the coarsest resolution level (res=1/4). We suspect this is because the benefit from spatiotemporal compression is so modest in this instance that the floating point arithmetic errors from the additional calculations begin to dominate. More discussions

on this topic could be found in [33] and [34]. That said, spatiotemporal still shows benefit over spatial compression; the extra accuracy from spatiotemporal compression can still make a significant difference in real-world analyses, as we demonstrate in Section VI-A.

Lastly, random access to individual time slices is easily lost during spatiotemporal compression. Specifically, during reconstruction of data from the compressed form, in the first step where inverse wavelet transforms are performed along the temporal domain, it needs to read in coefficients of other time slices that also belong to the same window. One can regain the ability of random access by employing smart coders on the resulting coefficients, for example, the one reported in [35].

VI. Applications to Real-World Analyses

In this section, we compare spatiotemporal (4D) wavelet compression with spatial-only (3D) wavelet compression on two real-world analyses. Both are representatives of regularly performed analyses by domain scientists, and they both operate on the Tornado simulation data set previously described in Section V-A3 The first analysis studied compression effects on pathlines across multiple time slices, while the second studied the effects on isosurfaces of a single time slice.

The mesh for the data set was a $490 \times 490 \times 280$ rectilinear grid, covering a geographic space of $14,670 \times 14,670 \times 8,370$ meters. The pathline analysis used data from 220 time slices, each stored as its own file. These 220 time slices were from the latter stages of the simulation when the tornado was most interesting to study. Each time slice advanced two seconds of simulation time past the previous one, with the first time slice being 8502 seconds into the simulation — i.e., $t_0 = 8502s$, $t_1 = 8504s$, ..., $t_{219} = 8940s$. Note this temporal resolution corresponds to "res=1/2" from Section V-A3), which is not the finest available. We used this resolution since it is what our domain scientist collaborator uses in his own research.

A. Pathline Analysis

Visualization: To better understand the tornado dynamics, we placed particles at the base of the tornado so that their trajectories could be observed. These particles are typically placed in a "rake" setting, i.e., densely seeding along a line segment. In this example, three rakes with 48 particles each were seeded, for a total of a 144 seed locations. The particles were advected using Runge-Kutta 4, with a step size of 0.01s. Velocity values between time slices were calculated using linear interpolation.

Compression: We worked with a total of nine versions of the data: the original version and eight wavelet-compressed versions. The wavelet-compressed versions included both spatiotemporal (4D) and spatial-only (3D) wavelet transforms over four compression levels (8:1, 32:1, 64:1, and 128:1). Both transforms used the CDF 9/7 wavelet kernel, and the spatiotemporal versions used a window size of 18. The three components of velocity were individually compressed for each wavelet version. We then generated a pathline for every combination of the 144 seed points and nine data sets, or $1,296$ pathlines overall. We set the baseline for each seed point as its pathline from the original version of the data set.

Table II: Our error metric for each wavelet-compressed data set, averaged over all 144 seed particles.

Data Set	D=10	D=50	D=150	D=300	D=500
8:1, 3D	8.5%	2.3%	1.3%	1.1%	1.0%
8:1, 4D	3.4%	1.3%	1.1%	0.8%	0.6%
32:1, 3D	35.9%	10.3%	4.5%	3.1%	2.4%
32:1, 4D	24.4%	6.4%	3.2%	2.1%	1.6%
64:1, 3D	48.4%	17.3%	**7.5%**	5.3%	3.9%
64:1, 4D	35.7%	9.8%	5.1%	3.3%	2.7%
128:1, 3D	60.7%	27.8%	10.8%	6.7%	5.2%
128:1, 4D	45.8%	16.3%	**7.5%**	5.1%	3.9%

Evaluation: We defined our evaluation metric incorporating observations from visually comparing baseline pathlines with their versions from wavelet-compressed data sets. Figure 4 visualizes example pathlines. Each image shows three pathlines generated by the advection of the same particle using the original and 3D or 4D compressed data at 128:1. Some particle trajectories would deviate midway through their courses, but then ultimately end up close to the correct positions, as the left two subfigures illustrate. So we designed an error metric that would value the case where a pathline stays close to its baseline throughout its entire trajectory, over one that deviates early but later returns. Specifically, let D be distance, let T be the total time of advection for a particle, and let T_0 be the first time that the pathline deviates distance D away from its baseline. Then we defined error as a percentage: (1.0 - T_0/T) $\times 100$. Taking an example: if a particle first deviates distance D from its baseline after six seconds and travels for a total of ten seconds, then we would score its error as 40%. We asked our domain scientist collaborator to select a good value for D and he picked a distance of 150 meters. We ran evaluations with that threshold, as well as bigger and smaller thresholds for comparative purposes. Five values of D were tested in our evaluation: D=10, 50, 150, 300, and 500.

Results: Table II contains the results from our evaluation. Each evaluation percentage is averaged from all 144 seed particles. This table shows a clear advantage of spatiotemporal compression, and supports both proposition **P1** and **P2**.

With respect to **P1** (improving accuracy while maintaining temporal resolution and storage costs), every combination of compression ratio and distance threshold shows the spatiotemporal compressed data to have superior accuracy (i.e., less error).

With respect to **P2** (reducing storage costs while maintain-

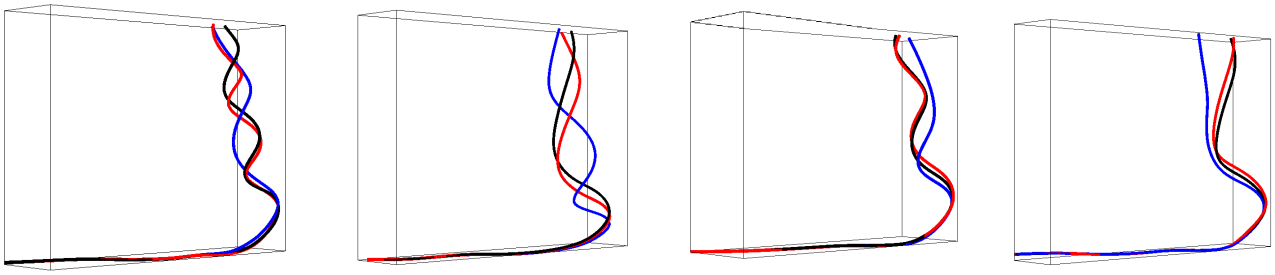


Figure 4: Each subfigure visualizes the pathlines for a single seed particle being advected using the original version of the data, as well as 3D and 4D compressed versions at 128:1 ratio. **Black** pathlines are from the original data set; **red** ones are from the 128:1+4D compression; and **blue** ones are from the 128:1+3D compression. The left two instances show that 4D (red) and 3D (blue) pathlines have similar ending positions, but the 4D ones closely resemble the baseline (black) for longer durations. The right two instances show a clear disadvantage of 3D pathlines in terms of both early deviation and far apart final positions.

ing temporal resolution and accuracy), comparisons between different compression ratios support this proposition. For example, for a distance threshold of 150, the 128:1+4D compression has the same error as the 64:1+3D compression (both are 7.5%, shown in bold font), meaning that the storage cost could be cut half. In fact, regardless of the distance threshold D, the error from 128:1+4D is always comparable to that from 64:1+3D, and the error from 64:1+4D is always comparable to that from 32:1+3D, supporting **P2** in a roughly $2X$ factor.

B. Isosurface Analysis

Visualization: Our domain scientist regularly studies isosurfaces in the Tornado data set. He provided us with three scalar variables that he often studies (pressure perturbation, cloud mixing ratio, and Z-component of velocity) as well as key isovalues appropriate for each of those variables.

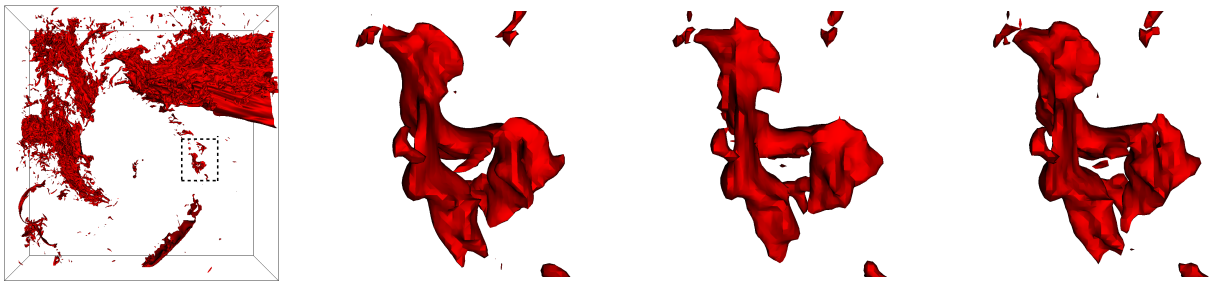
Compression: We worked with a total of 33 versions of the data: the original version of three scalar fields as well as ten wavelet-compressed versions of each. The wavelet-compressed versions included both spatiotemporal (4D) and spatial-only (3D) wavelet transformations and five compression levels (8:1, 16:1, 32:1, 64:1, and 128:1). We again used the CDF 9/7 kernel, and again the spatiotemporal versions used a window size of 18.

Evaluation: For each variable, we set the baseline isosurface to be the one from the original data set. Inspecting the isosurfaces visually, we found that the gross features from the baseline were well preserved within the wavelet-compressed versions, as Figure 5 shows. So our task shifted to capturing difference in fine details, and we opted to use the total surface area of the isosurfaces as our accuracy metric. That is, let SA_B be the surface area for the baseline isosurface and SA be the surface area for an isosurface from a wavelet-compressed data set. Then we defined our error metric again as a percentage: (1.0 - SA/SA_B) $\times$ 100. With this metric, 0% represents a perfect fit, with worse and worse fits moving away from 0 (in either the positive or negative directions). This metric creates the possibility for offsetting errors — compression may remove some features, but introduce others — but we did not observe this phenomenon in practice. Further, although we could have considered additional accuracy metrics (i.e., topological measures), we found that this simple metric corroborated our findings based on visual inspection, namely that 4D compression captures more surface details.

Table III: Our error metric for isosurfaces of each wavelet-compressed data set, comparing their surface area to the baseline surface area.

Variable	Compression	3D Error	4D Error
Cloud Mixing Ratio	8:1	-0.93%	0.35%
	16:1	-2.46%	0.59%
	32:1	-4.72%	0.47%
	64:1	-7.62%	-0.16%
	128:1	-11.24%	-1.34%
Z-Velocity	8:1	-0.85%	0.31%
	16:1	-2.23%	0.65%
	32:1	-4.69%	0.75%
	64:1	-8.85%	0.13%
	128:1	-15.320%	-1.66%
Pressure Perturbation	8:1	-2.4e-4%	-6.3e-3%
	16:1	-2.2e-2%	-1.8e-2%
	32:1	-4.9e-2%	-3.9e-2%
	64:1	-0.25%	-8.1e-2%
	128:1	-0.38%	6.7e-2%

Results: Table III contains the results from our evaluation. Like the analysis from Section VI-A, this table supports both propositions **P1** and **P2**. With respect to **P1** (improving accuracy while maintaining temporal resolution and storage costs), the 4D compressed data had less error for every variable and compression level, except for the 8:1 compressed version of pressure perturbation (which had very small total error). As an example, for 32:1 compression on cloud mixing ratio, the isosurface with 3D compressed data was 4.72% too small (in terms of total surface area), but with 4D



(a) Original data (b) Original data (c) 128:1+3D compression (d) 128:1+4D compression

Figure 5: Renderings of isosurfaces of the z-component of velocity from a Tornado data set. Subfigure (a) is the entire data set, while (b), (c), and (d) are the same zoomed-in region. Dashed lines in (a) indicate this region.

compressed data was only 0.47% too big. Similar disparities held with other combinations of cloud mixing ratio and z-velocity. However, for pressure perturbation, both techniques seem to capture the isosurface, even at high compression ratios, and so neither seems superior to the other.

With respect to **P2** (reducing storage cost while maintaining accuracy and temporal resolution), this analysis and error metric strongly favor 4D wavelet compression over 3D wavelet compression. For example, our error metric finds that 128:1 4D compressed data is more accurate than 16:1 3D compressed data for Z-velocity.

VII. Conclusions

Our study has considered the benefits, costs, and best practices for spatiotemporal wavelet compression compared to the traditional spatial-only approach. This direction is enabled by deeper memory hierarchies now increasingly available on leading-edge supercomputers. We studied the benefits of spatiotemporal compression by looking at both differences in reconstructed fields over a variety of settings and several real-world analyses in consultation with domain scientists. In nearly all cases, we found that incorporating the time dimension led to more "information per byte," realized with a variety of error metrics: NRMSE, normalized L^∞, and custom metrics for real-world applications. This property in turn enabled three propositions that can benefit domain scientists with their visualization needs — improving on accuracy, reducing storage, and increasing temporal frequency. While the magnitude of the benefit varies by use case, many of the results demonstrated factor-of-two improvements for their respective metrics.

Finally, unlike spatial coherence, temporal coherence is a function of how frequently the data is output. Often output frequency is constrained by storage space, and if too coarse, benefits from temporal compression may be small. That said, there are many analyses where higher output cadence is required, and spatiotemporal compression shows great promise when faced with constraints of limited I/O bandwidth and storage capacity.

References

[1] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann, 1996.

[2] C. S. Burrus, R. A. Gopinath, and H. Guo, "Introduction to wavelets and wavelet transforms: a primer," 1997.

[3] A. Islam and W. A. Pearlman, "Embedded and efficient low-complexity hierarchical image coder," in *Electronic Imaging'99*. International Society for Optics and Photonics, 1998, pp. 294–305.

[4] B.-J. Kim and W. A. Pearlman, "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)," in *Proceedings of the Data Compression Conference, DCC'97.* IEEE, 1997, pp. 251–260.

[5] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, 2000.

[6] J. D. Villasenor, R. Ergas, and P. Donoho, "Seismic data compression using high-dimensional wavelet transforms," in *Proceedings of the Data Compression Conference, DCC'96.* IEEE, 1996, pp. 396–405.

[7] A. Trott, R. Moorhead, and J. McGinley, "Wavelets applied to lossless compression and progressive transmission of floating point data in 3-D curvilinear grids," in *Proceedings of Visualization'96.* IEEE, 1996, pp. 385–388.

[8] L. Zeng, C. Jansen, M. A. Unser, and P. Hunziker, "Extension of wavelet compression algorithms to 3D and 4D image data: Exploitation of data coherence in higher dimensions allows very high compression ratios," in *International Symposium on Optical Science and Technology*. International Society for Optics and Photonics, 2001, pp. 427–433.

[9] L. Zeng, C. P. Jansen, S. Marsch, M. Unser, and P. R. Hunziker, "Four-dimensional wavelet compression of arbitrarily sized echocardiographic data," *IEEE Transactions on Medical Imaging*, vol. 21, no. 9, pp. 1179–1187, 2002.

[10] H. G. Lalgudi, A. Bilgin, M. W. Marcellin, and M. S. Nadar, "Compression of fMRI and ultrasound images using 4D SPIHT," in *IEEE International Conference on Image Processing*, vol. 2. IEEE, 2005, pp. II–746.

[11] H. G. Lalgudi, A. Bilgin, M. W. Marcellin, A. Tabesh, M. S. Nadar, and T. P. Trouard, "Four-dimensional compression of fMRI using JPEG2000," in *Proceedings of SPIE*, vol. 5747, 2005, p. 1029.

[12] C. Wang, J. Gao, L. Li, and H.-W. Shen, "A multiresolution volume rendering framework for large-scale time-varying data visualization," in *Fourth International Workshop on Volume Graphics*. IEEE, 2005, pp. 11–223.

[13] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, and M. Dietz, "ISO/IEC MPEG-2 advanced audio coding," *Journal of the Audio engineering society*, vol. 45, no. 10, pp. 789–814, 1997.

[14] S. Guthe and W. Straßer, "Real-time decompression and visualization of animated volume data," in *Proceedings of Visualization, VIS'01*. IEEE, 2001, pp. 349–572.

[15] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 343–348.

[16] V. Sanchez, P. Nasiopoulos, and R. Abugharbieh, "Efficient 4D motion compensated lossless compression of dynamic volumetric medical image data," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008*. IEEE, 2008, pp. 549–552.

[17] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova, "Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data," in *Euro-Par 2011 Parallel Processing*. Springer, 2011, pp. 366–379.

[18] H. Lehmann and B.-I. Jung, "In-situ multi-resolution and temporal data compression for visual exploration of large-scale scientific simulations," in *IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 2014, pp. 51–58.

[19] A. Agranovsky, D. Camp, C. Garth, E. W. Bethel, K. I. Joy, and H. Childs, "Improved Post Hoc Flow Analysis Via Lagrangian Representations," in *Proceedings of the IEEE Symposium on Large Data Visualization and Analysis (LDAV)*, Paris, France, Nov. 2014, pp. 67–75.

[20] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, 2012, pp. 1–11.

[21] B. Xie, J. Chase, D. Dillow, O. Drokin, S. Klasky, S. Oral, and N. Podhorszki, "Characterizing output bottlenecks in a supercomputer," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 8.

[22] T. Wang, S. Oral, Y. Wang, B. Settlemyer, S. Atchley, and W. Yu, "Burstmem: A high-performance burst buffer system for scientific applications," in *IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 71–79.

[23] A. Cohen, I. Daubechies, and J.-C. Feauveau, "Biorthogonal bases of compactly supported wavelets," *Communications on pure and applied mathematics*, vol. 45, no. 5, pp. 485–560, 1992.

[24] M. Unser and T. Blu, "Mathematical properties of the JPEG2000 wavelet filters," *IEEE Transactions on Image Processing*, vol. 12, no. 9, pp. 1080–1090, 2003.

[25] S. Li, K. Gruchalla, K. Potter, J. Clyne, and H. Childs, "Evaluating the Efficacy of Wavelet Configurations on Turbulent-Flow Data," in *Proceedings of IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, Chicago, IL, Oct. 2015, pp. 81–89.

[26] J. Woodring, S. Mniszewski, C. Brislawn, D. DeMarle, and J. Ahrens, "Revisiting wavelet compression for large-scale climate data using JPEG 2000 and ensuring data precision," in *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, Oct 2011, pp. 31–38.

[27] B. E. Usevitch, "A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000," *Signal Processing Magazine*, vol. 18, no. 5, pp. 22–35, 2001.

[28] P. Mininni, A. Alexakis, and A. Pouquet, "Nonlocal interactions in hydrodynamic turbulence at high reynolds numbers: The slow emergence of scaling laws," *Physical review E*, vol. 77, no. 3, p. 036306, 2008.

[29] ——, "Large-scale flow effects, energy transfer, and self-similarity on turbulence," *Physical Review E*, vol. 74, no. 1, p. 016303, 2006.

[30] L. Orf, R. Wilhelmson, and L. Wicker, "Visualization of a simulated long-track EF5 tornado embedded within a supercell thunderstorm," *Parallel Computing*, vol. 55, pp. 28–34, 2016.

[31] L. Orf, R. Wilhelmson, B. Lee, C. Finley, and A. Houston, "Evolution of a long-track violent tornado within a simulated supercell," *Bulletin of the American Meteorological Society*, vol. 98, no. 1, pp. 45–68, 2017.

[32] A. Mallinson, D. A. Beckingsale, W. Gaudin, J. Herdman, J. Levesque, and S. A. Jarvis, "Cloverleaf: Preparing hydrodynamics codes for exascale," *The Cray User Group*, pp. 6–9, 2013.

[33] F. Keinert, "Numerical stability of biorthogonal wavelet transforms," *Advances in Computational Mathematics*, vol. 4, no. 1, pp. 1–26, 1995.

[34] G. Plonka, H. Schumacher, and M. Tasche, "Numerical stability of biorthogonal wavelet transforms," *Advances in Computational Mathematics*, vol. 29, no. 1, pp. 1–25, 2008.

[35] F. F. Rodler, "Wavelet based 3d compression with fast random access for very large volume data," in *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications*. IEEE, 1999, pp. 108–117.