A General Framework for Augmenting Lossy Compressors with Topological Guarantees

Nathan Gorski, Xin Liang, Hanqi Guo, Lin Yan, and Bei Wang

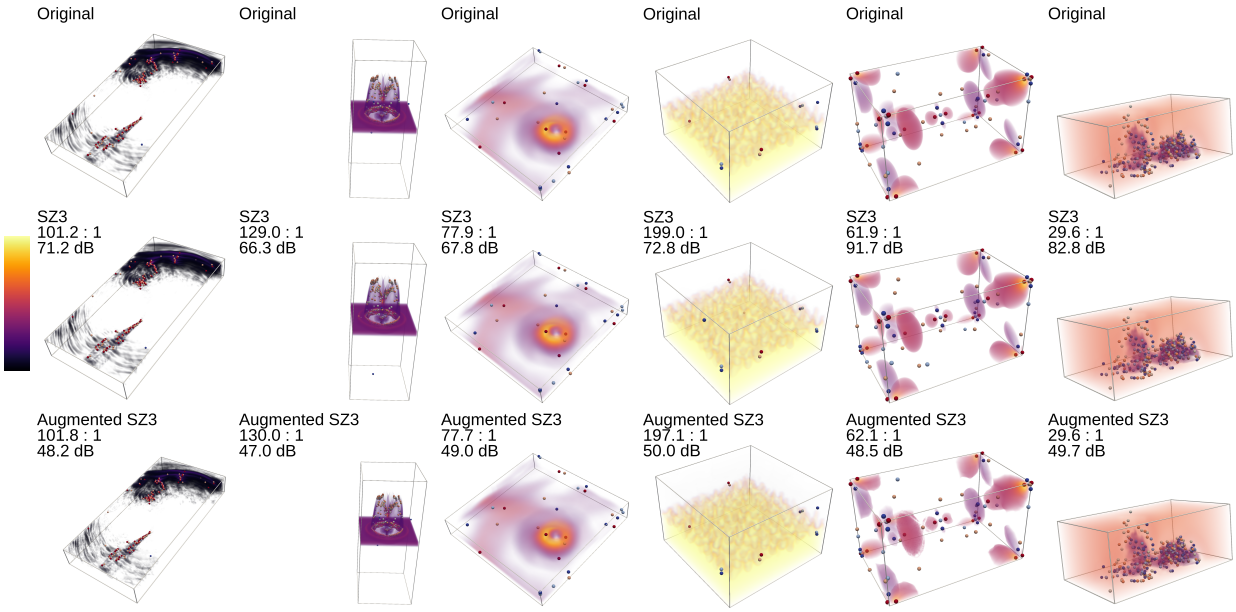


Fig. 1: Visualizing scientific datasets compressed with classic SZ3 and Augmented SZ3 compressors with topological controls. Each dataset is visualized. Top: input data visualized with critical points of the contour tree, maximum are in red, minimum are in blue, 1-saddles are in orange, 2-saddles are in white. Middle: reconstructed data using SZ3. Bottom: reconstructed data using Augmented SZ3. From left to right: Earthquake, Ionization, Isabel, Miranda, QMCPack, and Tangaroa datasets.

Abstract— Topological descriptors such as contour trees and Morse–Smale complexes are widely utilized in scientific data analysis and visualization, with applications from materials science to climate simulations. It is thus desirable to preserve topological descriptors when data compression is part of the scientific workflow for these applications. However, classic error-bounded lossy compressors for volumetric data do not guarantee the preservation of topological descriptors, despite imposing strict pointwise error bounds. In this work, we introduce a general framework for augmenting *any* lossy compressor to preserve the topology of the data during compression. Specifically, our framework quantifies the adjustments (to the decomposed data) needed to preserve the contour tree and then employs a custom variable-precision encoding scheme to store these adjustments. We demonstrate the utility of our framework in augmenting classic compressors (such as SZ3, TTHRESH, and ZFP) and deep learning-based compressors (such as Neurcomp) with topological guarantees.

Index Terms—Lossy compression, contour tree, topology preservation, topological data analysis, topology in visualization

1 Introduction

Modern scientific simulations generate enormous amounts of data, sometimes on the order of terabytes per minute [6]. Data compression helps reduce the size of data for storage and transmission in scientific data management systems. There are two types of compression techniques: lossy and lossless. Lossy compression techniques allow for some distortion of the data to achieve smaller compressed file sizes and have wide applicability in the compression of images [15], audio [17], and scientific data [6]. Among them, error-bounded lossy compressors, such as SZ [25], ZFP [26], and TTHRESH [4], play a crucial role in reducing the storage demand of large-scale scientific data. Not only can such compressors significantly reduce the data volume, but also they can control the data distortion and guarantee the validity of the reconstructed data for post hoc analysis, based on user-defined error bounds [28]. For instance, error-bounded lossy compressors have been shown to improve the I/O performance dramatically without significant degradation of the visual quality on the reconstructed (decompressed) data (e.g., [4, 24, 53]).

In the analysis of scientific data, topological data analysis (TDA) employs topological descriptors, such as contour trees [5] and Morse-Smale Complexes [11], to describe, summarize, and draw conclusions about scientific data (e.g., [2, 7, 39]); see [52] for a survey. While error-bounded lossy compressors typically allow the user to impose pointwise error bounds that are maintained during compression, such compressors seldom make guarantees about how the compression will affect the geometry and topology of the reconstructed data.

For example, Lu et al. [29] examined the impact of lossy compression on data fidelity and complex scientific data analytics. They studied

- *Nathan Gorski is with the University of Utah. E-mails: gorski@sci.utah.edu*
- *Xin Liang is with the University of Kentucky. E-mail: xliang@uky.edu.*
- *Hanqi Guo is with the Ohio State University. E-mail: guo.2154@osu.edu.*
- *Lin Yan is with the Iowa State University. E-mail: linyan@iastate.edu.*
- *Bei Wang is with the University of Utah. E-mail: beiwang@sci.utah.edu.*

the detection of *blobs*, features used by fusion scientists to study the trajectory of high-energy particles. Blobs are defined by areas with high electric potentials, i.e., areas enclosed by contours above certain thresholds. Their experiments demonstrated that as the error bound increases, the blobs will change both in number and position. They concluded that "lossy compression may seriously distort data, thus having a disastrous impact on data analytics," and therefore "determining a proper error bound is key to performing meaningful lossy compression in science production." [29] We argue that determining a proper error bound is only part of the story. It is also important to develop error-bounded lossy compressors that explicitly preserve features of interest to domain scientists—such as topological features—during compression.

In this paper, we introduce a general framework that augments *any* lossy compressor for volumetric scalar fields in order to impose topological control, while maintaining a user-specified pointwise error bound. Specifically, our framework augments any lossy compressor in order to preserve the *contour tree*, in terms of its critical points and the connectivity between those critical points. Preserving the contour tree of the reconstructed data is crucial to support a variety of *post hoc* scientific visualization tasks, as the contour tree has been used for feature extraction, tracking, comparison, and interactive contour exploration (e.g., [19, 55]). Previous work by Yan et al. [51] modified the classic SZ compressor with a customized error-controlled quantization strategy to preserve the contour tree. Instead of modifying individual compressors [51], our general framework effectively leverages the capabilities of a wide variety of data compressors. Additionally, as data compressors continue to improve, our framework can be used to augment increasingly effective compressors and thereby achieve better results. During augmentation, our framework first computes specific upper and lower bounds for each data point which, if maintained, will guarantee that the contour tree is preserved and that the pointwise error bound is maintained. It then uses a novel variable-precision encoding scheme to store any adjustments that must be made to the output of an augmented compressor, in order to ensure that these upper and lower bounds are maintained. Our contributions include:

- A custom framework that calculates and stores any adjustments that must be made to the output of an augmented compressor in order to preserve the contour tree.
- A custom variable-precision encoding scheme to efficiently store these adjustments.
- A systemic comparative study that evaluates five lossy compressors (ZFP, SZ3, TTHRESH, Neurcomp, Cubic Spline) augmented with topological control, and two state-of-the-art topology-preserving compressors, across a number of scientific datasets.

2 Related Work

We give a brief review of data compression for volumetric data. We then discuss the use of contour trees in topological data analysis, followed by related work for topology-preserving compression techniques.

Lossy compression. Lossless compression techniques allow the original data to be perfectly reconstructed, but they usually suffer from limited compression ratios (less than $2\times$ according to [41]) in scientific data and thus are not practical. As such, lossy compression is regarded as an alternative way to reduce the unprecedented data size of scientific data. Traditional lossy techniques such as JPEG/JPEG2000 leverage wavelet theories and bitplane encoding to compress image data, but they are not adept at dealing with multidimensional scientific data in floating-point format. Recently, there has been an increasing trend to leverage deep learning techniques, such as the autoencoder [21] and Implicit Neural Representation (INR) [30], for data compression. An autoencoder is a neural network composed of two components: an encoder and a decoder. The encoder is trained to produce low-dimensional representations of the input data, whereas the decoder is trained to reconstruct the original input data from the output of the encoder. An INR model trains a small neural network that can be used to recreate the ground truth. The neural network itself is shipped as a compressed file, and to decompress it, one must simply evaluate the network on an appropriate input. However, these general lossy techniques lack precise error control on the data, which limits their use on scientific data.

Error-controlled lossy compressors [4,20,26,53] have been proposed and leveraged by the scientific computing community to reduce the data size while controlling the distortion in the decompressed data. In general, such compressors can be categorized into transform-based compressors and prediction-based compressors. Transformation-based lossy compressors rely on domain transforms for data decorrelation. For instance, ZFP [26] divides data into small blocks and then compresses each block independently. The compression procedure inside each block includes exponent alignment for fixed point conversion, a near-orthogonal domain transform, and embedded encoding. TTHRESH [4] is another transform-based compression that leverages singular-value decomposition to improve the decorrelation efficiency for high dimensional data.

Prediction-based compressors employ prediction methods such as interpolation to approximate the ground truth. The differences between original and predicted data are quantized and then encoded using entropy encoding and lossless techniques. ISABELA [20], as one of the pioneering error-controlled prediction-based compressors, uses B-splines to predict data. SZ3 [23, 25, 53], the most recent general release in the SZ compressor family, uses a combination of a Lorenzo predictor [16], cubic spline interpolation, and linear interpolation. In addition, AE-SZ [27] is proposed as a variation of SZ that incorporates autoencoders in the prediction pipeline. Recently, spatial super-resolution (SSR) models employ neural networks to accurately upscale low-resolution representation of data as a form of interpolation. Several volumetric scalar field compressors incorporate SSR models, such as SSR-TVD [14] and the deep hierarchical model [50].

Contour trees. Our augmented compressors aim to preserve the contour tree of an input scalar field. Contour trees capture the relationships among contours of scalar fields. They have been used to support data analysis and visualization tasks across diverse disciplines, such as astronomy [39], fluid dynamics [2], and medicine [3, 42, 47]. They have also been incorporated into algorithms in computer vision [32] and visualization [19, 55] for interactive exploration of contours.

Topology-preserving compression. To the best of our knowledge, only two compressors have been developed for data compression with topological guarantees. The first compressor was developed by Soler et al. [40], which we shall refer to as TopoQZ. TopoQZ allows the user to specify a single parameter ϵ. It preserves all critical point pairs with finite persistence greater than ϵ and eliminates all critical points with persistence less than ϵ. TopoQZ is not designed to perfectly preserve the contour tree. Therefore, the locations of preserved critical points may shift slightly during compression, and the connectivity of the critical points in the contour tree may be altered. TopoQZ also allows the user to guarantee that the reconstructed values differ from the ground truth at most by a user-specified error bound ξ. In the original implementation, it was required that $\xi > \epsilon$, but a more recent implementation allows for $\xi \leq \epsilon$. This recent implementation has been incorporated into the Topology Toolkit [31, 45].

Another topology-preserving compressor is TopoSZ [51]. TopoSZ modifies the classic SZ pipeline to perfectly preserve the contour tree of the ground truth data up to the persistence threshold of ϵ. That is, the contour tree of the output of TopoSZ will be equal to that of the ground truth after both datasets have been topologically simplified with persistence ϵ. TopoSZ also allows the user to impose a strict error-bound ξ, and when compared with TopoQZ, yields generally higher compression ratios and reconstruction quality; although the algorithm takes longer to execute. Our general framework borrows some elements from the TopoSZ pipeline.

While it does not preserve any common topological descriptor, cpSZ [22]—a variation of SZ—preserves the critical points of a vector field, which is related to the contour tree. cpSZ also introduces a log-scale quantization technique to store different error bounds for individual points.

3 Technical Background

3.1 Contour Tree and Persistence Simplification

Let $f : \mathbb{X} \to \mathbb{R}$ be a continuous scalar field defined on a simply connected domain $\mathbb{X}$. The level set of f at a threshold $t \in \mathbb{R}$ is defined to be $f^{-1}(t)$. The connected components of $f^{-1}(t)$ are called *contours* of f. Intuitively, the contour tree $T(\mathbb{X}, f)$ is constructed by contracting each contour of f into a single point. Mathematically, define an equivalence relation between a pair of points $x, y \in \mathbb{X}$ such that $x \sim y$ iff they belong to the same contour. The contour tree is the quotient space of X under the equivalence relation, $T = T(\mathbb{X}, f) := X/\sim$.

As t increases, the places where contours of $f^{-1}(t)$ appear or disappear correspond to the local extrema of f and the leaves of the contour tree. The places where contours merge or split correspond to saddles of f and branching nodes of T. Thus, there is a one-to-one correspondence between the leaves of T and the local extrema of f. However, the only saddle points that appear are those where a merge or split occurs in $f^{-1}(t)$ as t increases and other saddle points may not be vertices of the contour tree [38]. Fig. 2 (A) and (D) visualize a scalar field and its corresponding contour tree.

The construction of a contour tree also yields a segmentation of the data domain $\mathbb{X}$. Let $\phi : \mathbb{X} \to T$ be the canonical map that maps each datapoint $x \in \mathbb{X}$ to its equivalence class in T under $\sim$. If e is an edge in T, then $\phi^{-1}(e)$ is a connected region in $\mathbb{X}$. Given a set of edges in a contour tree T (see Fig. 2 E), segmenting $\mathbb{X}$ by the pre-image of each edge of T under ϕ is called the *contour-tree-induced segmentation*, visualized in Fig. 2 (B).

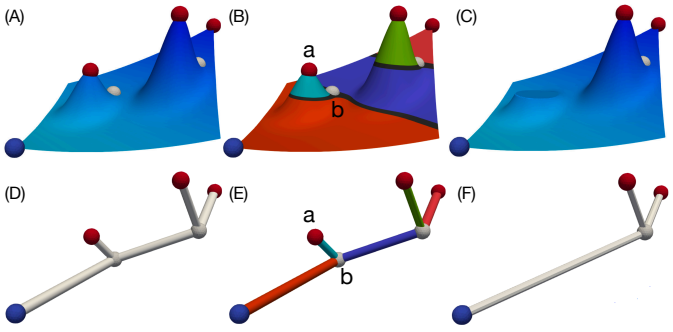


Fig. 2: (A) A 2D scalar field defined on a square domain, where scalar field value corresponds to height. Local minima are in blue, saddles are in white, and local maxima are in red. (B) A contour-tree-induced segmentation of the domain. (C) The 2D scalar field after a persistence simplification that removes a pair of a local maximum and a saddle. (D) The contour tree of the scalar field in (A). (E) Contour tree edges are colored to indicate the regions in the domain segmentation from (C). (F) The contour tree of the scalar field after persistence simplification.

Persistent homology uses tools from algebraic topology to separate topological features from noise in data. In the context of contour trees, ordinary persistent homology pairs a local extremum (a peak or a valley) with a nearby saddle and assigns the pair a value of *persistence*, which describes the scale at which the pair disappears via a perturbation to the function. A contour tree can be simplified by canceling pairs of critical points below a certain persistence threshold ε. In the example shown in Fig. 2 (B) and (E), a local extremum a is paired with a saddle b with a persistence $|f(a) - f(b)|$. Assuming a persistence threshold of $\varepsilon = |f(a) - f(b)|$, the pair (a, b) will be canceled based on persistence simplification. That is, the smaller peak a is flattened out to the height of saddle b. This corresponds to removing a branch (a, b) from the contour tree. Such a simplification is depicted in Fig. 2 (C) and (F).

In practice, real-world data typically contain noise that creates many small branches in the contour tree, where persistence simplification can be used to eliminate these small branches and thereby separate topological features from noise.

3.2 A Review on TopoSZ

Our framework builds upon a few key ingredients from TopoSZ [51]. TopoSZ, in turn, modifies the pipeline from the error-bounded lossy compressor SZ version 1.4 [43].

Let f represent the input scalar field, and f' be the reconstructed scalar field (after compression and decompression). Let T be the contour tree of f and T_ε the persistence simplified contour tree at a threshold of ε. Let T' and T'_ε be defined analogously for f'.

SZ1.4 allows the user to specify ξ, a pointwise error bound during compression. In turn, there are two user-defined parameters in TopoSZ: a persistence threshold ε, and a pointwise error bound ξ. TopoSZ guarantees the preservation of the persistence simplified contour tree during compression while maintaining the pointwise error bound. That is, it guarantees that $T_\varepsilon = T'_\varepsilon$, and $|f(x) - f'(x)| \leq \xi$ for each $x \in \mathbb{X}$.

False Cases. Yan et al. [51] introduced three types of false cases to quantify the level of contour tree preservation: false positives, false negatives, and false types, which are illustrated in Fig. 3. A false positive occurs when a new edge appears in the contour tree of the reconstructed data that does not exist in the same position of the contour tree of the original data. A false negative occurs when an edge of the contour tree from the original data is missing from the contour tree of the reconstructed data. A false type occurs when the critical type (maximum, minimum, saddle) of one or both endpoints of an edge of the contour tree do not match between the original and reconstructed data. TopoSZ focused on false cases involving extremum-saddle pairs. The algorithm terminates when there are no such false cases.

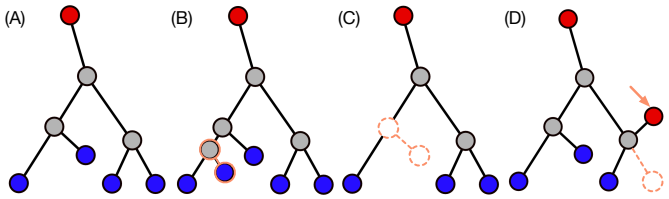


Fig. 3: Three types of false cases. (A) The original contour tree. (B) A false positive: an extra edge is added. (C) A false negative: an edge is missing. (D) A false type: an edge contains a critical point as its endpoint that changes its type.

Pipeline. The TopoSZ pipeline is as follows:

Step 1: Upper and lower bound calculation. TopoSZ first calculates $T_\varepsilon(\mathbb{X}, f)$. For each point $x \in \mathbb{X}$, a lower bound $L(x)$ and an upper bound $U(x)$ are assigned to x according to the contour-tree-induced segmentation and ξ. If x belongs to the segmented region corresponding to an edge $(a, b) \in T_\varepsilon(\mathbb{X}, f)$, then $L(x) = \min(f(a), f(b))$ and $U(x) = \max(f(a), f(b))$. The vertices of the contour tree are stored losslessly.

Step 2: Prediction. TopoSZ uses a Lorenzo predictor [16] to predict the values of each datapoint based on the decompressed values of its neighbors.

Step 3: Linear-scaling quantization. TopoSZ uses a modified linear-scaling quantization technique to ensure that the pointwise upper and lower bounds, as well as the global error bound ξ, are maintained for each $x \in \mathbb{X}$.

Step 4: Iterative upper and lower bound tightening. If the results from Step 3 do not perfectly preserve the contour tree, that is, if there are false cases presented in the reconstructed data, then the upper and lower bounds are tightened around points in the segmented regions containing the false cases, and then Step 3 is repeated. This continues until there are no false cases.

Step 5: Lossless compression. The numbers from linear-scaling quantization are encoded using Huffman Coding. The other relevant information is stored in a binary file. All of the data is then compressed using ZSTD [9].

We provide additional details on Step 3 and Step 4 below.

Linear-scaling quantization. SZ ensures that a strict absolute error bound ξ is maintained using a linear-scaling quantization. In standard linear-scaling quantization, for each point $x \in \mathbb{X}$ with a ground truth value $f(x)$, an initial guess for its value $g(x)$ (e.g., from a Lorenzo predictor) is shifted by an integer multiple of 2ξ to obtain a new value $f'(x)$ such that $|f'(x) - f(x)| \leq \xi$.

This process can be conceptualized as follows. Divide the real line into intervals of length 2ξ, where one interval is centered on $g(x)$. The compressor then calculates how many intervals to shift $g(x)$, so that it can assign a value to $f'(x)$ that is a distance less than ξ from $f(x)$. By construction, if $f(x)$ lies in an interval of length 2ξ centered on $f'(x)$, it must hold that $|f(x) - f'(x)| \leq \xi$. This process is illustrated in Fig. 4.

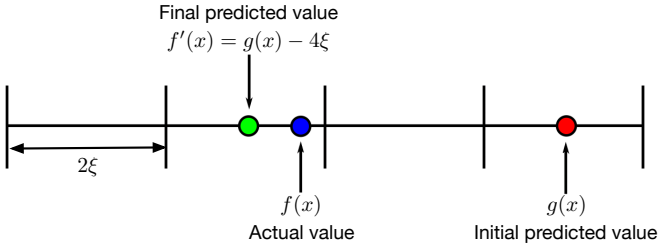


Fig. 4: A standard implementation of a linear-scaling quantization.

Following this construction, each $x \in \mathbb{X}$ is assigned an integer n_x such that $f'(x) = g(x) + 2\xi n_x$. In standard linear-scaling quantization, these quantization numbers $\{n_x\}$ are encoded and stored in the compressed file. TopoSZ specifically stores the quantization interval for each x as a 16 bit positive integer equal to $n_x + 2^{15} - 1$. If n_x is such that $n_x + 2^{15} - 1$ is not a 16 bit positive integer, $f(x)$ is stored losslessly.

If the distribution of $\{n_x\}$ has low entropy (e.g., if $\{n_x\}$ are mostly zeros), then they can be compressed to a very small size using an entropy-based compressor, such as Huffman coding. Having more accurate predictions $g(x)$ generally leads to $\{n_x\}$ with lower entropy.

TopoSZ modifies the standard implementation of linear-scaling quantization to respect the pointwise upper and lower bounds. First, it lowers the length of each interval to ξ, down from 2ξ. Due to this modification, there are always two (instead of one) possible choices for $f'(x)$ that are at a distance ξ from $f(x)$. During quantization, if either of these two choices for $f'(x)$ is between the upper and lower bounds imposed on x, then TopoSZ will select the valid option that is closest to $f(x)$. Otherwise, $f(x)$ will be stored losslessly. Lowering the interval length to ξ improves the chances that there will be a valid choice for $f'(x)$ between $L(x)$ and $U(x)$.

Iterative upper and lower bound tightening. For each false case that is detected in Step 4, TopoSZ calculates a region R of $\mathbb{X}$. Then, it tightens the upper and lower bounds $U(x)$ and $L(x)$ of points in $x \in R$. The region R and how aggressively $L(x)$ and $U(x)$ are tightened depends on how many times this step has been repeated. Let n be the number of times that Step 4 has occurred, with $n = 1$ initially.

First, a region R is calculated. Define an m-layer neighborhood of a point x as the set of all points y such $||x - y||_\infty \leq m$. In the event of a false positive edge e that is in T'_ε but absent in T_ε, R is initially set equal to the region corresponding to e in the segmentation induced by T'_ε. R is then expanded by adding an n-layer neighborhood of R to R.

In the event of a false negative edge e' that is present in T_ε but not T'_ε, R is initially set equal to the region corresponding to e' in the segmentation induced by T_ε. Let s be the saddle point which is a vertex of e'. An n-layer neighborhood surrounding s is added to R. TopoSZ handles false types in the same way that it handles false negatives.

Then, TopoSZ tightens the $L(x)$ and $U(x)$ bounds around points $x \in R$. Let $k_0 = \min\{f(x) : x \in \mathbb{R}\}$ and $k_{n+1} = \max\{f(x) : x \in \mathbb{R}\}$. TopoSZ calculates $n + 1$ intervals $[k_0, k_1], [k_1, k_2], \ldots, [k_n, k_{n+1}] \subset \mathbb{R}$ such that, for each interval I, approximately $\frac{1}{n+1}$ of points $x \in R$ satisfy $f(x) \in I$. Then, for each point $x \in R$, if $f(x) \in [k_i, k_{i+1}]$, then $L(x)$ and $U(x)$ are adjusted according to $L(x) \leftarrow \max(L(x), k_i)$ and $U(x) \leftarrow \min(U(x), k_{i+1})$.

4 METHOD

We give an overview of our framework in Sec. 4.1, followed by technical details on the novel ingredients, including the logarithmic-scaling quantization (Sec. 4.2), the upper and lower bound tightening (Sec. 4.3), and the lossless compression (Sec. 4.4).

4.1 Overview

We now describe our framework for augmenting any lossy compressor (called a *base compressor*) to preserve contour trees and maintain strict error bounds. Our framework requires two user-specified parameters, a persistence threshold ε and a pointwise absolute error bound ξ. It also requires user-specified parameters associated with the specific base compressor being augmented.

We denote the original input data as f, and the reconstructed (decompressed) data as f'. Let T be the contour tree of f, and T_ϵ be T subjected to persistent simplification with threshold ε. Let T' and T'_ϵ be defined analogously for f'. Our framework guarantees that, for any augmented compressor, $T_\varepsilon = T'_\varepsilon$ and $|f(x) - f'(x)| \leq \xi$ for every $x \in \mathbb{X}$. Starting with a standard compressor as the base compressor, we start with a step-by-step overview of our framework.

Step 1: Upper and bound calculation. We calculate the initial pointwise upper and lower bounds using a technique similar to the one used by TopoSZ. If a point x lies in the segmentation region corresponding to an edge (a, b), we let $L(x) = \min(f(a), f(b)) + \zeta$ and $U(x) = \max(f(a), f(b)) - \zeta$, where ζ is 0.1% of the range. We found that adjusting by ζ leads to fewer false cases in data with very high precision. We also adjust $L(x)$ and $U(x)$ as needed to ensure that if $L(x) \leq f'(x) \leq U(x)$ then $|f(x) - f'(x)| \leq \xi$. Critical points are stored losslessly.

Step 2: Base compressor. We apply the base compressor to the input data f. We compress then decompress the data to assess changes that need to be made during decompression. We refer to the compressed-then-decompressed data as the *intermediate data*.

Step 3: Logarithmic-scaling quantization. We introduce a novel quantization technique that respects the pointwise upper and lower bounds imposed in Step 1. If possible, the entropy of the quantization numbers $\{n_x\}$ will be identical to that of standard linear-scaling quantization with an interval length 2ξ. However, when necessary, individual points are quantized with more precision to respect pointwise upper and lower bounds without the point needing to be stored losslessly.

Step 4: Iterative upper and lower bound tightening. We iteratively tighten the upper and lower bounds of certain data points until the contour tree is perfectly preserved. Different from TopoSZ, our implementation alters the fashion in which upper and lower bounds are tightened in order to speed up convergence and decrease the number of points that must be stored losslessly.

Step 5: Lossless compression. We process the quantization numbers using a novel technique that leverages spatial correlation to reduce their entropy. We then encode them using Huffman coding. If the ground truth data uses a 64 bit encoding, all points that would be stored losslessly are stored with 32 bits if this still preserves the contour tree. The output of the base compressor, the encoded quantization numbers, and any losslessy stored values are written to a binary file which is further losslessly compressed using xz, a general-purpose data compression tool available via XZ Utils [10].

4.2 Logarithmic-Scaling Quantization

We now describe our variable precision quantization technique that preserves tight pointwise upper and lower bounds without significantly compromising the entropy of the overall distribution of quantization numbers. For each $x \in \mathbb{X}$, the intermediate data contains an estimated value $g(x)$ for the ground truth value $f(x)$. Let $L(x)$ and $U(x)$ denote the lower and upper bounds assigned to x. To ensure that $L(x) \leq f'(x) \leq U(x)$, we assign to each $x \in \mathbb{X}$ a numerator $a_x \in \mathbb{R}$ and a precision $p_x \in \mathbb{N}$. Our reconstructed value is

$$f'(x) = g(x) + \frac{2\xi \cdot a_x}{2^{p_x}}. \tag{1}$$

To calculate each a_x and p_x, we first set $p_x = 0$. We then look for the value of a_x satisfying

$$L(x) \leq g(x) + \frac{2\xi \cdot a_x}{2^{p_x}} \leq U(x)$$

such that $|a_x|$ is minimized. If there is no valid value of a_x, we increase p_x by 1 and search again. This process is repeated until a valid a_x is found. If p_x reaches an arbitrary threshold, we stop searching and instead store $f(x)$ losslessly. We set this threshold equal to 50.

When $p_x = 0$, the above process is the same as the standard linear-scaling quantization, except that we also seek to maintain the upper and lower bounds. Each time a linear-scaling quantization fails to identify a valid choice for a_x that yields a value of $f'(x)$ within the upper and lower bounds for x, we cut the interval lengths in half by increasing p_x by 1 and continue searching. When the interval lengths are smaller, it is more likely that a valid choice of a_x exists. It is also possible that during an iteration, multiple valid choices of a_x exist, so we choose the the option with the smallest absolute value to minimize the entropy of $\{a_x\}$.

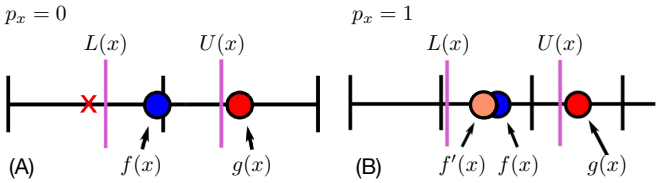


Fig. 5: (A) If $p_x = 0$, there are no valid quantization intervals. (B) Increasing p_x to 1 allows for a valid quantization interval.

This process is illustrated in Fig. 5. (A) contains an example where there are no valid quantization intervals in which we can place $f'(x)$ to respect the upper and lower bounds. In (B), by raising the precision p_x by 1, the quantization intervals are halved, giving a valid choice for $f'(x)$.

When encoding the data, we store a single quantization number n_x for each $x \in \mathbb{X}$. To calculate each n_x, we first calculate a precision cutoff k_p and a cardinality cutoff k_c. Any point $x \in X$ is stored losslessly if $p_x > k_p$, or if the total number of points with the same values of n_x and p_x is less than k_c. The remaining points are assigned the single quantization number $n_x = a_x \cdot 2^{k_p - p_x}$ and the precision cutoff k_p is stored in the compressed output. During decompression, the point x is assigned the value

$$f'(x) = g(x) + \frac{2\xi \cdot n_x}{2^{k_p}}. \tag{2}$$

Setting $n_x = a_x \cdot 2^{k_p - p_x}$ in Eq. (2) means that,

$$g(x) + \frac{2\xi \cdot n_x}{2^{k_p}} = g(x) + \frac{2\xi \cdot a_x \cdot 2^{k_p - p_x}}{2^{k_p}} = g(x) + \frac{2\xi \cdot a_x}{2^{p_x}} \tag{3}$$

Therefore, the formulation in Eq. (2) is equivalent to the original formulation of f' in Eq. (1).

In comparison with TopoSZ, the above variable precision technique allows us to store fewer points losslessly overall. It also allows us to have a quantization interval of length 2ξ, leading to a distribution of quantization numbers with lower entropy. Our strategy employs precision and cardinality cutoffs k_p and k_c as it is possible that infrequent occurrences of large quantization numbers can take more bits to encode using our strategy than if their corresponding datapoints were stored losslessly. This is because of the overhead involved in the Huffman coding. Our current implementation uses a greedy algorithm to calculate locally optimal values of k_p and k_c, respectively.

4.3 Upper and Lower Bound Tightening

Like TopoSZ, our framework seeks to eliminate false cases involving extremum-saddle pairs by tightening the upper and lower bounds in the regions corresponding to these false cases in the contour-tree-induced segmentation. However, our upper and lower bound tightening strategy differs from that of TopoSZ in several ways. First, during the nth iteration, we calculate 2^{n+1} rather than $n+1$ intervals for tightening the upper and lower bounds, which leads to faster convergence. We also grow our regions more slowly during the first few iterations. Empirically, we have found that this leads to fewer points with very tight upper and lower bounds.

In the case of false positives, during each step, TopoSZ only grows the region where the upper and lower bound is tightened by adding points adjacent to the saddle point of the false-positive edge, rather than adding points adjacent to the entire region. Our algorithm does this initially, but after several iterations, it switches to growing the region by adding points that border the entire region. We have found that this speeds up convergence when the topological regions corresponding to false cases are very thin.

4.4 Lossless Compression

Given a list of quantization numbers, $[n_1, n_2, \ldots, n_N]$ (for N points), we aim to store it losslessly. We first preprocess the list to reduce its entropy in a way that takes advantage of spatial correlation. To that end, we subtract from each quantization number the previous one. That is, the list to be encoded is $[n_1, n_2 - n_1, n_3 - n_2, \ldots, n_N - n_{N-1}]$.

For example, if our list of quantization numbers is $[1, 1, 1, 2, 2, 2]$, the quantization numbers that would get stored would be $[1, 0, 0, 1, 0, 0]$. Because quantization numbers tend to have similar values to their neighbors, this transformation yields a list of numbers with lower absolute values, and thus lower entropy. This allows the numbers to be stored more efficiently using an entropy-based compressor. While this technique is very simple, to our knowledge, our framework is the first compression algorithm to implement it.

After this, the quantization numbers are encoded using a Huffman coding. They are then stored along with the compressed output of the base compressor and the losslessly stored points, and the entire file is further compressed using the xz compressor [10]. We have experimented with several combinations of entropy encodings and final lossless compressors, and have found that a combination of Huffman coding with the xz compressor leads to the best compression ratios.

5 Experimental Results

We provide an overview in Sec. 5.1, describing the compressors and datasets used in our experiments, highlighting the main takeaways, and introducing the evaluation metrics. We include compressor configurations and implementation details in Sec. 5.2. We then describe in Sec. 5.3 the main utilities of our augmented compressors in preserving contour trees in the reconstructed data. We evaluate a number of augmented compressors qualitatively and quantitatively, followed by comparison against the state-of-the-art topology-preserving compressors in Sec. 5.4. We end this section with a runtime analysis in Sec. 5.5.

5.1 An Overview of Experiments

We present a comparative analysis that evaluates five error-bounded lossy compressors augmented using our general framework, including the classic compressors SZ3 [25], TTHRESH [4], and ZFP [26], a deep learning-based compressor Neurcomp [30], and a custom-built cubic spline interpolation (CSI) model. We test these augmented compressors—denoted as Augmented SZ3, Augmented TTHRESH, and so on—against two state-of-the-art topology-preserving compressors, TopoSZ [40] and TopoQZ [51].

We test the five augmented compressors and two topology-preserving compressors on six volumetric datasets from scientific simulations. See Tab. 1 and Appendix A for details on these datasets.

Dataset	Dimension	Size (MB)
Earthquake	$175 \times 188 \times 50$	28.2
Ionization	$310 \times 128 \times 128$	40.6
Isabel	$500 \times 500 \times 90$	105.0
Miranda	$384 \times 384 \times 256$	302.0
QMCPack	$69 \times 69 \times 115$	4.4
Tangaroa	$300 \times 180 \times 200$	27.0

Table 1: Scientific datasets used for compression analysis.

Highlighted results. We highlight our experimental results below.

- Applying any of the five original compressors to any of the six datasets produces a large number of topological false cases in the

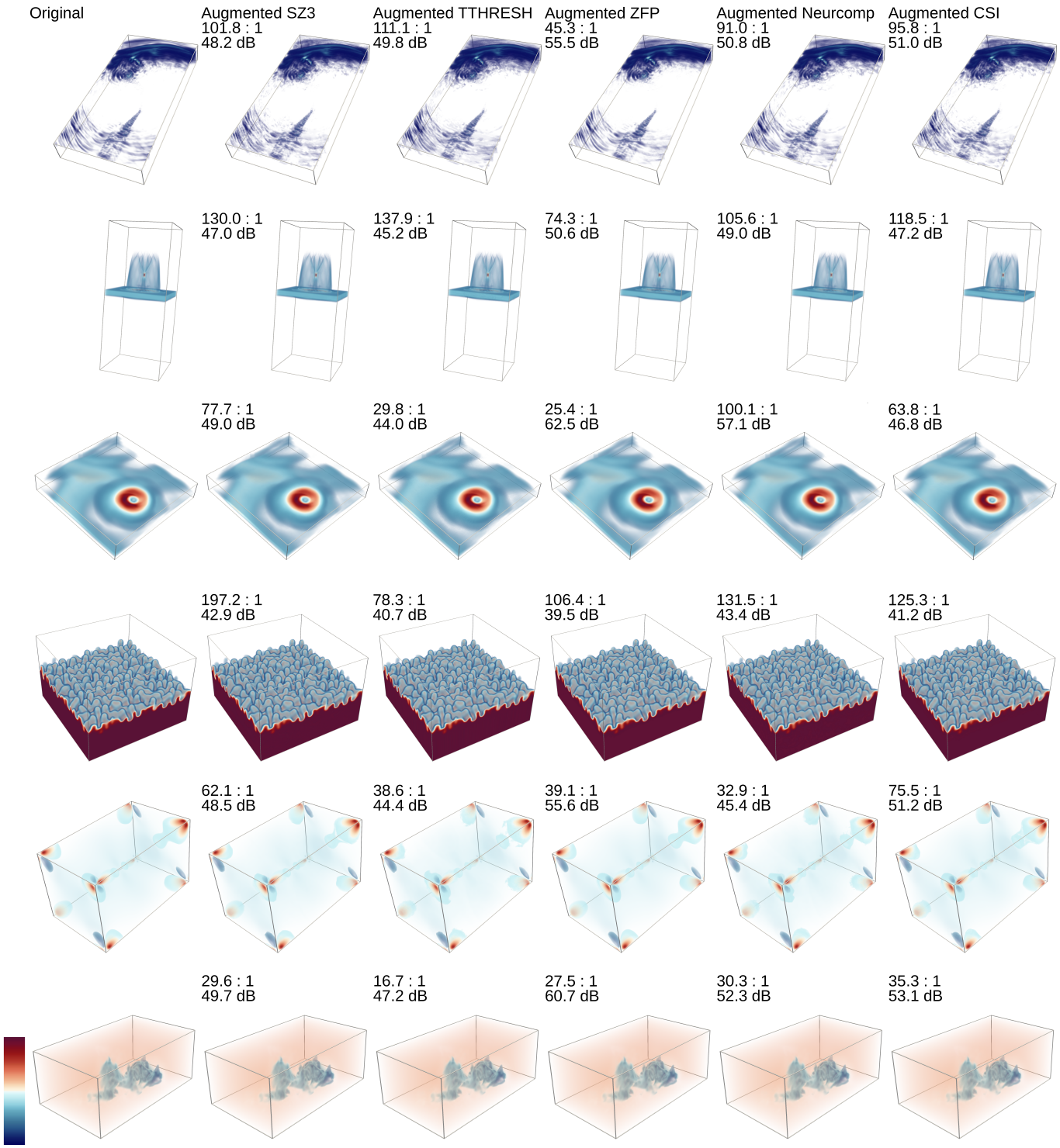


Fig. 6: Scientific datasets compressed using different augmented compressors with topological controls. From left to right: the original input dataset, the reconstructed datasets from Augmented SZ3, Augmented TTHRESH, Augmented ZFP, Augmented Neurcomp, and Augmented CSI respectively, that preserve the contour trees up to a persistence threshold $\varepsilon = 0.04$. From top to bottom: Earthquake, Ionization, Isabel, QMCPack, Tangaroa datasets, respectively. We also display the PSNR and compression ratio next to each decompressed dataset.

reconstruction, even with a small pointwise error bound. On the other hand, augmenting any compressor with our general framework completely eliminates these false cases while maintaining a user-specified error bound (Sec. 5.3).

- Augmented TTHRESH and Augmented SZ3 yield the best compression ratios and the fastest runtime among all the augmented compressors at the expense of some reconstruction quality (Sec. 5.3).
- Our augmented compressors generally produce higher compression ratios than TopoQZ and TopoSZ with comparable reconstruction quality and slower compression runtime (Sec. 5.4).
- The majority of the compression time is spent on computing the contour trees. We argue that in certain use cases prioritizing storage requirements, it is preferable to have a higher compression ratio over a faster runtime (Sec. 5.5).

Evaluation metrics. We evaluate whether the contour tree has been perfectly preserved in the reconstructed (decompressed) data. We also evaluate the standard compression metrics of compression ratio and

peak signal-to-noise ratio (PSNR). We further employ topology-based metrics of the bottleneck distances d_B [8] and the Wasserstein distances d_W [12, page 183] to quantify the topological similarity between the original data and the reconstructed data. The evaluation metrics are described in detail in Appendix B.

In general, higher values of PSNR indicate better reconstruction quality and lower values of d_B and d_W indicate higher topological similarity. We also measure the total compression time for each compressor, which includes (a) the total time to run the base compressor, and (b) the time to augment the output of the base compressor.

5.2 Compressor Configurations and Implementation

In addition to augmenting the out-of-box base compressors SZ3, TTHRESH, ZFP, and Neurcomp, we implement and augment our own super-resolution compressor, a simple custom-built cubic spline interpolation (CSI) model. It compresses a dataset by downsampling the data by a user-defined ratio in each direction (called a target scale factor) and uses a cubic spline interpolation technique for reconstruction that is similar to the one implemented in SZ3. We also considered the Sliced Wasserstein Autoencoder [18] used in the AE-SZ compressor [27]. However, this model is excluded from our experiments as it performed significantly worse than the other compressors during initial tests.

We compare our augmented compressors to TopoSZ and TopoQZ. We use the TopoQZ implementation in TTK [45]. The TTK implementation of TopoQZ incorporates ZFP [26] to improve the overall reconstruction quality. We chose to minimize the contribution from ZFP to maximize compression ratios at the cost of reconstruction quality.

Our general framework requires two user-defined parameters, a persistence threshold ε and a global absolute pointwise error bound ξ. ε represents, as a percentage of the range, the level of persistent simplification. For example, $\varepsilon = 0.01$ corresponds to a persistent simplification by 1% of the range of the scalar function. Similarly, ξ represents the percentage of the range that will be used as an absolute error bound.

Each base compressor takes a number of intrinsic parameters in order to run. Both SZ3 and ZFP require an absolute error bound, denoted as η and δ, respectively. TTHRESH takes in a target RMSE of τ. Neurcomp requires a target compression ratio c. CSI requires a target scale factor s. For our experiments, we set $\eta = \xi$, $\delta = \xi$ and $\tau = \xi$. We also set $c = 100$ and $s = 7$. We chose these configurations because they empirically led to the highest compression ratios. It is important to note that changing the intrinsic parameters of a base compressor will cause it to generate different intermediate data which will be augmented differently. As a result, while our augmented compressor guarantees topology preservation and maintains the user-defined global error bound, the compression results may vary.

We apply each compressor on each dataset for 11 parameter combinations of ε and ξ. For seven trials, we fix $\xi = 0.012$ and vary $\varepsilon \in \{0.01, 0.25, 0.04, 0.08, 0.12, 0.16, 0.2\}$. For five trials, we fix $\varepsilon = 0.04$ and vary $\xi \in \{0.004, 0.008, 0.012, 0.016, 0.02\}$. These cases overlap on $\varepsilon = 0.04$ and $\xi = 0.012$. The combination of a chosen compressor, a fixed dataset, a value of ε and ξ, is a trial. We perform each trial on a single cluster node running an Intel Xeon Sandy Bridge-E processor with 16 cores and 64GB of RAM. For Neurcomp, we perform the training on an RTX 2080ti GPU with 32GB of RAM.

5.3 Comparative Analysis of Augmented Compressors

In this section, we perform a comparative analysis of five augmented compressors, qualitatively and quantitatively. We visualize six scientific datasets before and after compression with our augmented compressors in Fig. 6. We also display the PSNR and compression ratio next to each decompressed dataset. Evaluation metrics are reported in Tab. 3 and compression and decompression times are reported in Tab. 4.

5.3.1 Topological Guarantees

When compressing a dataset with any base compressor, the contour tree of the data is often significantly distorted with a large number of false cases, whereas it is always perfectly preserved using our augmented compressor.

We demonstrate the behaviors of base compressors and augmented compressors in Fig. 1, where we visualize each dataset with the vertices of its contour tree (which are also the critical points of the underlying scalar fields), using a persistence threshold of $\varepsilon = 0.04$. On the top row, we visualize the original data. Below that, we visualize the reconstructed data after it has been compressed with SZ3 (second row) and augmented SZ3 (third row). For the base compressor SZ3, we chose η to produce compression ratios comparable to those obtained by Augmented SZ3, with $\eta = 0.012$, and $\xi = 0.012$ (the configuration shown on the third row). Comparing the middle row against the top row, the mismatches between the vertices indicate that SZ3 does not perfectly preserve the contour tree.

Zoomed-in views of these images for the earthquake datasets are shown in Fig. 7. In those zoomed-in views, we can see that, for the earthquake dataset, the base compressor SZ3 fails to predict two critical points while altering the locations of the other critical points. We can also observe many false cases for the ionization dataset. In comparison, Augmented SZ3 successfully preserves all critical points of the contour trees, as well as their underlying connectivities that are not shown.

In general, the base compressors do not preserve the contour tree of the reconstructed data. By contrast, by comparing the top row against the bottom row in Fig. 1, our framework is shown to perfectly preserve the vertices of the contour tree. A figure analogous to Fig. 1 for TTHRESH is included in the supplementary material Appendix C.

As shown in Fig. 1, whereas SZ3 (and other base compressors such as TTHRESH) may preserve the locations of many critical points, the connectivity among these critical points is oftentimes distorted. In Tab. 2, we report the number of false cases, including both extremum-saddle and saddle-saddle connections in the contour tree reconstructed with SZ3 and TTHRESH. We again use parameter configurations that yield the same compression ratios as their augmented versions with $\eta = 0.012$, $\tau = 0.012$ and $\xi = 0.012$. Table 2 shows that these base compressors typically produce many false cases and therefore do not preserve the contour tree. Notably, for the Isabel dataset, SZ3 does poorly and predicts no edges correctly in the contour tree. On the other hand, we found no false cases of any kind across all trials when augmenting compressors with our framework. The parameter configurations used for SZ3 and TTHRESH are reported in Appendix D.

Dataset	SZ3	TTHRESH	Total #edges
Earthquake	(24, 24, 0)	(67, 67, 0)	169
Ionization	(82, 86, 0)	(18, 18, 0)	568
Isabel	(31, 29, 0)	(25, 25, 0)	29
Miranda	(5, 5, 0)	(0, 0, 0)	11
QMCPack	(42,42,0)	(34,34,0)	69
Tangaroa	(49, 51, 0)	(18, 18, 0)	418

Table 2: Reporting the number of false cases (false positives, false negatives, false types) produced by base compressors SZ3 and TTHRESH, respectively, together with the total number of edges of the input (ground truth) contour tree. Contour trees are simplified with $\varepsilon = 0.04$.

5.3.2 Evaluation Metrics

The evaluation metrics are reported in Tab. 3 for a fixed parameter configuration of $\varepsilon = 0.04$ and $\xi = 0.012$. We chose this parameter configuration because a small amount of persistence simplification preserves a large number of topological features in the input data, generating complex test cases for topology-preserving compression. For completeness, the results across different trials and parameter configurations are included in Appendix E. In this section, we compare the different augmented compressors. We leave the comparison with TopoQZ and TopoSZ to Sec. 5.4.

Compression ratios. As shown in Tab. 3, no single augmented compressor gives the best compression ratios across all trials. While Augmented TTHRESH and Augmented CSI produce the best compression ratios in some trials, Augmented SZ3 consistently produces competitive compression ratios. Augmented ZFP produces noticeably worse compression ratios than the other compressors.

Dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
	Compression Ratio							PSNR						
Earthquake	101.8	**111.1**	45.3	90.9	95.8	50.1	70.7	48.2	49.8	**55.5**	50.7	51	42.6	45.7
Ionization	130.1	**137.9**	74.4	105.9	118.5	25.1	76	47	45.2	**50.6**	48.9	47.2	48.1	43.2
Isabel	77.7	29.8	25.4	**100.1**	63.8	37.6	30.3	49	44	**62.5**	57.1	46.8	49	13.4
Miranda	**197.1**	78.3	106.4	126.6	125.3	95.9	72.3	42.9	40.7	39.5	42.1	41.2	**49.5**	43.6
QMCPack	62.1	38.6	39.1	32.9	**75.4**	27.8	34.2	48.5	44.4	**55.6**	45.4	51.2	46.6	42.8
Tangaroa	29.6	16.7	27.5	30.3	**35.3**	24.3	27.3	49.7	47.2	**60.7**	52.3	53.1	48.8	4.2
	Wasserstein Distance d_W							Bottleneck Distance d_B						
Earthquake	0.29	0.18	0.4	0.22	0.26	0.66	**0.13**	0.0099	0.0117	0.0063	0.0093	0.0082	**0.0060**	0.0183
Ionization	0.32	0.51	0.33	**0.32**	0.54	1.07	0.49	0.0109	0.0118	**0.0079**	0.0114	0.0120	0.0105	0.0527
Isabel	0.73	1.33	0.66	0.81	0.71	**0.22**	0.64	0.0104	0.0120	**0.0064**	0.0086	0.0115	0.0077	0.3184
Miranda	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QMCPack	0.18	**0.11**	0.24	0.27	0.23	0.12	0.14	0.0117	0.0104	**0.0083**	0.0110	0.0115	0.0089	0.0504
Tangaroa	0.45	0.77	**0.44**	0.47	0.55	1.05	3.38	0.0101	0.0119	**0.0075**	0.0089	0.0099	0.0083	0.4219

Table 3: Evaluation metrics for five augmented compressors, together with TopoSZ and TopoQZ. All trials are conducted with $\varepsilon = 0.04$ and $\xi = 0.012$. A-SZ3 is a shorthand for Augmented SZ3, and so on. The Miranda dataset yielded very large persistence diagrams for which d_W and d_B cannot be computed in a reasonable time. In order to compute d_W and d_B for TopoSZ in a reasonable time, for all datasets except for QMCPack, the d_W and d_B values associated with TopoSZ were calculated after persistent simplification with a threshold of $1.5e-6$.

Dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
	Total Compression and Augmentation Time					Compression Time		Decompression Time						
Earthquake	557.1	630.8	797.9	1682.8	934.9	47	**7.5**	24.9	24.7	25.6	37.7	123.9	**0.06**	2.6
Ionization	117.2	127.8	85.2	1361.4	261.6	428.2	**9.4**	36.2	36.4	36.1	52.3	170.7	**0.09**	3.9
Isabel	1114.7	1042.2	503.9	8193.5	1757.8	367.1	**75.4**	169.9	167.3	163.1	257.4	776.6	**0.41**	26.3
Miranda	613.8	649.3	575.7	9713.9	1576.8	438.3	**157.8**	274.4	294.4	272	476.9	1295.7	**0.68**	51.4
QMCPack	9.6	14.5	9.3	181.9	23.4	11.9	**0.9**	3.8	4.1	3.7	9.1	18.6	**0.01**	0.4
Tangaroa	190.7	203.6	92.6	1606.9	357.2	314.6	**13.7**	44.6	45.5	44	68.1	224.9	**0.12**	4.4

Table 4: Compression time (including augmentation time) and decompression time for different augmented compressors, topoQZ and TopoSZ, along with compression and decompression time, in seconds. All trials are conducted with $\varepsilon = 0.04$ and $\xi = 0.012$.

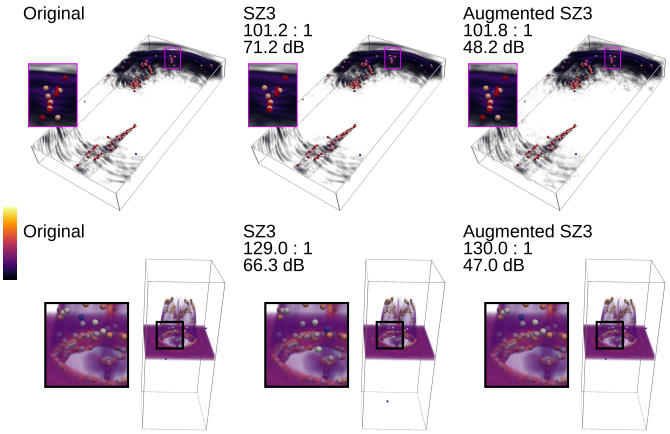


Fig. 7: Zoomed-in views of the critical points of the contour tree of the earthquake dataset with persistent simplification $\varepsilon = 0.04$. Local maxima are in red, local minima are in blue, 1-saddles are in orange, 2-saddles are in white. Top row: earthquake. Bottom row: ionization

Reconstruction quality. In every trial, our framework successfully maintains a pointwise error bound ξ. Augmented ZFP gives rise to the best reconstruction quality evaluated using PSNR, d_W, and d_B. However, this is not surprising, as Augmented ZFP gives the lowest compression ratios. Among the remaining compressors, not a single compressor clearly outperforms the others in terms of PSNR, although Augmented Neurcomp and Augmented CSI appear to have a slight upper hand. Aside from Augmented ZFP, no augmented compressor seems to have a clear upper hand for d_W or d_B. SZ3, the top-performing compressor in terms of compression ratio, is still competitive for these metrics.

In practice, we find that each round of upper and lower bound tightening does not affect the PSNR very much, with later iterations having minimal effect on the overall PSNR. Likewise, the points that require significant precision to maintain their upper and lower bounds are a small percentage of the overall volume. This is demonstrated in Fig. 8 with a slice of the Ionization dataset. It shows a map of the error over the course of the tightening process, as well as the final precisions used to store each point.

Runtime analysis. There are significant differences in running time among the augmented compressors. These times are affected by a variety of factors and the compressors with the best total time (compression plus augmentation) are not necessarily the ones with the fastest base compression time. This is discussed further in Sec. 5.5. However, the two slowest compressors, Neurcomp and CSI also happen to have the longest total compression times. Of the remaining three compressors, there is again not a clear best-augmented compressor, although Augmented ZFP appears to be the fastest, while Augmented SZ3 is slightly faster than Augmented TTHRESH. In terms of decompression time, Augmented CSI and Augmented Neurcomp are still the two slowest compressors. The remaining compressors give similar times.

Highlighted results. In general, among all augmented compressors, Augmented SZ3 appears to be the overall best base compressor for use with our general framework. Augmented SZ3 consistently produces high compression ratios, and has a competitive runtime. The second best-augmented compressor appears to be Augmented TTHRESH, as it sometimes produces higher compression ratios than Augmented SZ3 and has a comparable runtime. For the remainder of our analysis, we will primarily focus on Augmented SZ3 and Augmented TTHRESH.

5.4 Comparison with TopoQZ and TopoSZ

Topological guarantees. Our framework can augment any lossy compressor to preserve the contour tree during compression. This is the same topological guarantee made by TopoSZ. However, this is different from the topological control associated with TopoQZ. TopoQZ ensures that all pairs of critical points from the contour tree are preserved above a certain persistence threshold ε. However, because it is not intended to preserve the contour tree, the locations and connectivity of these critical points may be distorted after compression.

Compression ratio. In terms of compression ratio, our framework outperforms TopoSZ and TopoQZ in almost every trial, including those not shown in Tab. 3. While the optimal augmented compressors vary among trials, Augmented SZ3 outperforms both TopoQZ and TopoSZ in every trial except for a few with relatively high error bounds. Our augmented compressors outperform TopoSZ and TopoQZ more as ε increases, although a topology-preserving compressor is considered to be more useful when ε is small and the number of topological features is large.

Reconstruction quality. While our framework produces higher compression ratios than TopoSZ and TopoQZ, Augmented TTHRESH and Augmented SZ3 both produce outputs with PSNR values that are

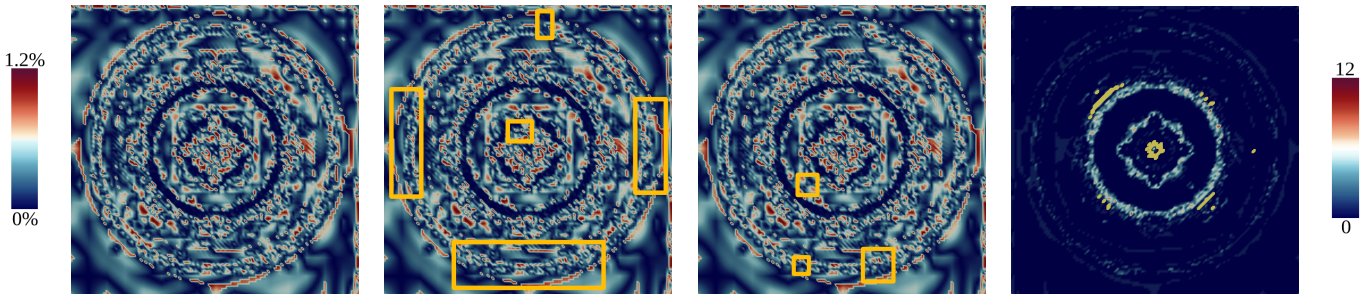


Fig. 8: (A-C) Error map of a slice of the Ionization dataset during first (A), second (B), and fifth/final (C) iterations of upper and lower bound tightening. (D) Map of the precision p_x used to encode each data point where datapoints requiring lossless compression are colored in yellow. During this trial, $\varepsilon = 0.01$ and $\xi = 0.012$.

comparable with TopoSZ and higher than TopoQZ. Additionally, Augmented SZ3 and Augmented TTHRESH produce values of d_W and d_B that are comparable to TopoSZ and TopoQZ.

Runtime analysis. In terms of compression time, Augmented SZ3 and Augmented TTHRESH have slower yet comparable compression times compared to TopoSZ, and significantly slower times compared to TopoQZ. It makes sense that our framework yields running times that are comparable to TopoSZ since both frameworks have identical topology-preserving objectives and share some key components in their pipelines. In order to preserve the contour tree of the data, our framework and TopoSZ both iteratively compute the contour tree of the reconstructed data. According to Yan et al. [51], this took the majority of the compression time for TopoSZ. As shown in Sec. 5.5, this is also the case for our framework. Because TopoQZ does not target the preservation of contour trees, it does not need to perform this time-consuming task.

While our compression times are slightly slower than TopoSZ, we argue that in many use cases, this is not a problem. The primary purpose of compressing data is to save storage space, for which our framework outperforms TopoSZ. Our framework often produces compressed file sizes that are less than half the size of those produced by TopoSZ. It should further be noted that our implementation is a prototype written in Python, while TopoSZ was partially implemented in C++, so it does not make sense to directly compare our runtimes. Optimizing the implementation of our framework is a possible area of future research.

Dataset	BC	CT	ULB	Fille	#I	Valid	LSQ	AEB	Total
Augmented SZ3									
Earthquake	9.4	245.3	7.2	5.8	2	139.6	3.8	2.6	557.1
Ionization	14.9	10.5	10.8	8.5	3	19.2	3	3	117.2
Isabel	68.8	231.3	45.2	37.1	7	87.5	10.6	7.6	1114.6
Miranda	107.7	119.4	75.4	58.6	1	211.9	40.6	0	613.7
QMCPack	1.6	2.2	1.1	1.2	1	1.4	0.3	0	7.9
Tangaroa	19.3	10.6	14	11.6	5	20.7	4.7	3.4	196.2
Augmented TTHRESH									
Earthquake	10.4	245.1	6.9	6	2	177.7	2.3	2.6	630.8
Ionization	15.8	11.5	10.7	9.1	4	14.7	3.3	2.9	127.7
Isabel	76.1	234.2	45.1	40.9	7	64.9	20.7	7.7	1042
Miranda	119.3	114.1	74.7	66.3	1	225.9	48.9	0	649.1
QMCPack	1.7	2.2	1.1	1.4	2	1.5	0.6	1.7	12.3
Tangaroa	19.7	10.8	13.1	13.2	5	19.6	6.4	3.5	200.7

Table 5: Runtime analysis for each component of the augmented framework involving Augmented SZ3 and Augmented TTHRESH with $\varepsilon = 0.04$ and $\xi = 0.012$. All times are in seconds. BC: running the base compressor. CT: computing the contour tree of the input data. ULB: calculating the initial upper and lower bounds. File: outputting the binary file. #I: the number of iterations. Valid: Average time to check that there are no false cases. LSQ: Average time to perform logarithm-scaling quantization. AEB: Average time to adjust the upper and lower bounds (does not occur on final iteration).

5.5 Runtime Analysis

Asymptotic analysis. In our general framework, the majority of the compression time is used on computing the contour trees, which is part of the iterative error bound tightening step. During each iteration of this, we compute the contour tree using the algorithm of Gueunet et al. [13] and apply persistence simplification using the algorithm in [46]. For a rectilinear mesh with n vertices, both have an average runtime of $O(n \log(n))$. The persistence algorithm has a worst-case complexity of $O(n^3)$ [33], which rarely occurs in practice. Assuming that our tightening step takes a small, constant number of iterations, then the total runtime of our algorithm is $O(n \log(n))$. In the worst case, our algorithm would take $O(n)$ iterations before all points are stored losslessly, giving our algorithm a worse-case runtime of $O(n^2 \log(n))$. In practice, this is also very rare.

Empirical analysis. To analyze the runtime empirically, we calculate the amount of time for each portion of our algorithm with Augmented SZ3 and Augmented TTHRESH, with $\varepsilon = 0.04$ and $\xi = 0.012$. These runtimes are shown in Tab. 5.

In Tab. 5, the most time-consuming task is the computation of contour trees. We compute a contour tree for the input data at the beginning of the algorithm and for the decompressed data during each iteration of error-bound tightening. These two runtimes are shown in Tab. 5 under the 'CT' and 'Valid' columns. The time spent on these two computations accounts for $42 - 95\%$ of the total runtime for each trial in Tab. 5. Similar to our analysis, Yan et al. noticed that computing the contour tree took the largest percentage of compression time for TopoSZ.

For most of the trials, the time to run the base compressor, shown in the 'BC' column, is a relatively small percentage of the overall compression time. In general, the time to run one iteration of error-bound tightening exceeds the time to run the base compressor. On the other hand, if a base compressor produces results that nearly preserve the contour tree, fewer iterations are needed for the error-bound tightening step. This suggests that the accuracy of the base compressor may have more effect on the total time than the speed of the base compressor.

6 Conclusion and Discussion

We introduce a novel framework for augmenting *any* lossy compressor to preserve the contour tree of a volumetric dataset while maintaining a user-specified global error bound. To do this, our framework first imposes topology-informed error bounds on each data point. It then iteratively tighten those error bounds until the contour tree is preserved. We also introduce a novel encoding scheme that efficiently stores individual points with variable precision and maintains these error bounds. When our framework is used to augment state-of-the-art lossy compressors, it is shown to preserve the contour trees of various scientific datasets. Our augmented compressors also achieve higher compression ratios than those obtained by existing topology-preserving compressors. Our framework will benefit from any advancement with lossy compression, as it can be used to augment increasingly effective lossy compressors to achieve higher-quality topology-preserving compression.

Our framework is not without limitations, as it produces relatively high compression times, the majority of which is spent computing the contour tree. However, if we prioritize storage requirements, it is preferable to have a higher compression ratio instead of a faster runtime. Regardless, our framework would benefit from more efficient implementations, both for the contour tree computation and the encoding

scheme. We would also like to apply our framework to a larger number of base compressors, in order to provide a user guide on selecting the best compressor and parameter combinations for topology-preserving compression.

REFERENCES

[1] Scientific Data Reduction Benchmarks. https://sdrbench.github.io/. 12

[2] D. B. Aydogan and J. Hyttinen. Characterization of microstructures using contour tree connectivity for fluid flow analysis. *Journal of The Royal Society Interface*, 11(95):20131042, 2014. 1, 2

[3] D. B. Aydogan, N. Moritz, H. T. Aro, and J. Hyttinen. Analysis of trabecular bone microstructure using contour tree connectivity. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013: 16th International Conference, Nagoya, Japan, September 22-26, 2013, Proceedings, Part II 16*, pp. 428–435. Springer, 2013. 2

[4] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. Tthresh: Tensor compression for multidimensional visual data. *IEEE transactions on visualization and computer graphics*, 26(9):2891–2903, 2019. 1, 2, 5

[5] R. L. Boyell and H. Ruston. Hybrid techniques for real-time radar simulation. In *Proceedings of the 1963 Fall Joint Computer Conference*, pp. 445–458. IEEE, 1963. 1

[6] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao, C. H. Yoon, X.-C. Wu, Y. Alexeev, and F. T. Chong. Use cases of lossy compression for floating-point data in scientific data sets. *The International Journal of High Performance Computing Applications*, 33(6):1201–1220, 2019. 1

[7] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the morse-smale complex and the connolly function. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pp. 351–360, 2003. 1

[8] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pp. 263–271, 2005. 7, 12

[9] Y. Collet and M. Kucherawy. Zstandard compression and the application/zstd media type. Technical report, Internet Engineering Task Force (IETF), 2018. doi: 10.17487/RFC8478 3

[10] L. Collin and J. Tan. XZ Utils. https://xz.tukaani.org/xz-utils/. 4, 5

[11] H. Edelsbrunner, J. Harer, and A. Zomorodian. Hierarchical morse complexes for piecewise linear 2-manifolds. In *Proceedings of the 17th Annual Symposium on Computational Geometry*, pp. 70–79. Medford, MA, USA, 2001. 1

[12] H. Edelsbrunner and J. L. Harer. *Computational topology: an introduction*. American Mathematical Society, 2022. 7, 12

[13] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based augmented merge trees with fibonacci heaps. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 6–15. IEEE, 2017. 9

[14] J. Han and C. Wang. Ssr-tvd: Spatial super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2445–2456, 2020. 2

[15] A. J. Hussain, A. Al-Fayadh, and N. Radi. Image compression techniques: A survey in lossless and lossy algorithms. *Neurocomputing*, 300:44–69, 2018. 1

[16] 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, pp. 343–348. Wiley Online Library, 2003. 2, 3

[17] P. Kavitha. A survey on lossless and lossy data compression methods. *International Journal of Computer Science & Engineering Technology*, 7(03):110–114, 2016. 1

[18] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde. Sliced-wasserstein autoencoder: An embarrassingly simple generative model. *arXiv preprint arXiv:1804.01947*, 2018. 7

[19] W. Köpp and T. Weinkauf. Temporal merge tree maps: A topology-based static visualization for temporal scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1157–1167, 2022. 2

[20] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Isabela for effective in situ compression of scientific data. *Concurrency and Computation: Practice and Experience*, 25(4):524–540, 2013. 2

[21] H. Le, H. Santos, and J. Tao. Hierarchical autoencoder-based lossy compression for large-scale high-resolution scientific data. *arXiv preprint arXiv:2307.04216*, 2023. 2

[22] X. Liang, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, T. Peterka, and H. Guo. Toward feature-preserving vector field compression. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 2

[23] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pp. 438–447. IEEE, 2018. 2

[24] X. Liang, S. Di, D. Tao, S. Li, B. Nicolae, Z. Chen, and F. Cappello. Improving performance of data dumping with lossy compression for scientific simulation. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1–11, 2019. doi: 10.1109/CLUSTER.2019.8891037 1

[25] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao, et al. Sz3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data*, 9(2):485–498, 2022. 1, 2, 5

[26] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE transactions on visualization and computer graphics*, 20(12):2674–2683, 2014. 1, 2, 5, 7

[27] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang, Z. Chen, and F. Cappello. Exploring autoencoder-based error-bounded compression for scientific data. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 294–306. IEEE, 2021. 2, 7

[28] Y. Liu, S. Di, K. Zhao, S. Jin, C. Wang, K. Chard, D. Tao, I. Foster, and F. Cappello. Optimizing error-bounded lossy compression for scientific data with diverse constraints. *IEEE Transactions on Parallel and Distributed Systems*, 33(12):4440–4457, 2022. doi: 10.1109/TPDS.2022.3194695 1

[29] T. Lu, Q. Liu, X. He, H. Luo, E. Suchyta, J. Choi, N. Podhorszki, S. Klasky, M. Wolf, T. Liu, et al. Understanding and modeling lossy compression schemes on hpc scientific data. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 348–357, 2018. 1, 2

[30] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. In *Computer Graphics Forum*, vol. 40, pp. 135–146. Wiley Online Library, 2021. 2, 5

[31] T. B. Masood, J. Budin, M. Falk, G. Favelier, C. Garth, C. Gueunet, P. Guillou, L. Hofmann, P. Hristov, A. Kamakshidasan, C. Kappe, P. Klacansky, P. Laurin, J. A. Levine, J. Lukasczyk, D. Sakurai, M. Soler, P. Steneteg, J. Tierny, W. Usher, J. Vidal, and M. Wozniak. An overview of the topology toolkit. In I. Hotz, T. Bin Masood, F. Sadlo, and J. Tierny, eds., *Topological Methods in Data Analysis and Visualization VI*, pp. 327–342. Springer International Publishing, Cham, 2021. 2

[32] S. Mizuta and T. Matsuda. Description of digital images by region-based contour trees. In *International Conference Image Analysis and Recognition*, pp. 549–558. Springer, 2005. 2

[33] D. Morozov. Persistence algorithm takes cubic time in the worst case. BioGeometry News, Department of Computer Science, Duke University, Durham, NC, 2005. 9

[34] K. Olsen, S. Day, B. Minster, R. Moore, Y. Cui, A. Chourasia, M. Thiebaux, H. Francoeur, P. Maechling, S. Cutchin, and K. Nunes. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2006/, 2006. 12

[35] K. Olsen, S. Day, J. Minster, Y. Cui, A. Chourasia, D. Okaya, P. Maechling, and T. Jordan. Terashake2: Spontaneous rupture simulations of m w 7.7 earthquakes on the southern san andreas fault. *Bulletin of the Seismological Society of America*, 98(3):1162–1185, 2008. 12

[36] M. Pont, J. Vidal, J. Delon, and J. Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):291–301, 2021. 12

[37] S. Popinet, M. Smith, and C. Stevens. Experimental and numerical study of the turbulence characteristics of airflow around a research vessel. *Journal of Atmospheric and Oceanic Technology*, 21(10):1575–1589, 2004. 12

[38] G. Reeb. Sur les points singuliers d'une forme de pfaff completement integrable ou d'une fonction numerique [on the singular points of a completely integrable pfaff form or of a numerical function]. *Comptes Rendus Acad. Sciences Paris*, 222:847–849, 1946. 3

[39] P. Rosen, A. Seth, E. Mills, A. Ginsburg, J. Kamenetzky, J. Kern, C. R. Johnson, and B. Wang. Using contour trees in the analysis and visualization of radio astronomy data cubes. In *Topological Methods in Data Analysis and Visualization VI: Theory, Applications, and Software*, pp. 87–108. Springer, 2021. 1, 2

[40] M. Soler, M. Plainchault, B. Conche, and J. Tierny. Topologically con-

trolled lossy compression. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 46–55, 2018. 2, 5

[41] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary. Data compression for the exascale computing era-survey. *Supercomputing frontiers and innovations*, 1(2):76–88, 2014. 2

[42] A. Szymczak. A categorical approach to contour, split and join trees with application to airway segmentation. In *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, pp. 205–216. Springer, 2010. 2

[43] D. Tao, S. Di, Z. Chen, and F. Cappello. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1129–1139. IEEE, 2017. 3

[44] R. Taylor, A. Chourasia, D. Whalen, and M. L. Norman. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2008/, 2008. 12

[45] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 2017. https://topology-tool-kit.github.io/. 2, 7

[46] J. Tierny and V. Pascucci. Generalized topological simplification of scalar fields on surfaces. *IEEE transactions on visualization and computer graphics*, 18(12):2005–2013, 2012. 9

[47] L. Wang, Q. Guo, J. Zhao, S. Zhang, and L. Yang. The fast contour tree-based medical volume rendering method. *Journal of Medical Imaging and Health Informatics*, 8(7):1451–1455, 2018. 2

[48] W. Wang, C. Bruyere, B. Kuo, and T. Scheitlin. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2004/, 2004. 12

[49] D. Whalen and M. L. Norman. Ionization front instabilities in primordial h ii regions. *The Astrophysical Journal*, 673(2):664, 2008. 12

[50] S. W. Wurster, H. Guo, H.-W. Shen, T. Peterka, and J. Xu. Deep hierarchical super resolution for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 2

[51] L. Yan, X. Liang, H. Guo, and B. Wang. Toposz: Preserving topology in error-bounded lossy compression. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 2, 3, 5, 9

[52] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum (CGF)*, 40(3):599–633, 2021. 1

[53] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello. Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation. In *Proceedings of IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 1643–1654, 2021. 1, 2

[54] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello. Sdrbench: Scientific data reduction benchmark for lossy compressors. In *2020 IEEE international conference on big data (Big Data)*, pp. 2716–2724, 2020. 12

[55] J. Zhou and M. Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Transactions on Visualization and Computer graphics*, 15(6):1481–1488, 2009. 2

A Details on the Datasets

We include details on each dataset used in our experiments. All data was normalized to a range of $[0, 1]$ before compression.

The **Earthquake** dataset originates from a TeraShake 2 earthquake simulation [35] and has been part of the 2006 IEEE Visualization Design Contest [34]. The dataset used in this paper was obtained from the public data repository of Pont et al. used for their publication [36]. The data was preprocessed by Pont et al. and came with a single field. It represents one time step of a simulation of an earthquake at the San Andreas fault. Specifically, we used time step 011700. The details of the preprocessing can be found on the repository.

The **Ionization** dataset originates from an ionization front simulation by Whalen and Norman [49] and has been featured in the 2008 IEEE Visualization Design Contest [44]. The simulation is done with 3D radiation hydrodynamical calculations of ionization front instabilities in which multifrequency radiative transfer is coupled to the primordial chemistry of eight species [49]. The single time step used in this paper comes from cluster 2, time step 0125 and was obtained from the same repository as the Earthquake dataset. It also was preprocessed and came with a single field. The details of the preprocessing can be found on the repository.

The **Isabel** dataset originates from a hurricane simulation from the National Center for Atmospheric Research, and has been included in the 2004 IEEE Visualization Design Contest [48]. While the original dataset has a size of $500 \times 500 \times 100$, we truncate the dataset to $500 \times 500 \times 90$ in order to avoid land regions that contain no data values. We used the wind speed field.

The **Miranda** dataset comes from the hydrodynamics code for large turbulence simulations conducted by Lawrence Livermore National Laboratory. We used the density field.

The **QMCPack** dataset comes from the QMCPACK performance test conducted by Argonne National Laboratory. Only the 145th orbital out of 288 was used for testing, for which only a single field is provided.

Both the Miranda and QMCPack datasets come from the SDR Bench [54] and have been accessed from the SDR Bench website [1]. These datasets involve contributions from the DOE NNSA ECP project and the ECP CODAR project.

The **Tangaroa** dataset comes from a single time frame simulating the wind flow around a 3D model of the Research Vessel Tangaroa [37]. The magnitude of the wind velocity is used as the scalar field of interest.

B Evaluation Metrics

We evaluated our tests on several metrics. We measured compression ratio, compression time, and peak signal-to-noise ratio (PSNR), which are common metrics for evaluating compressors.

The compression ratio is the size of the uncompressed file divided by the size of the compressed file, and higher compression ratios indicate smaller compressed file sizes.

PSNR is a number that measures reconstruction quality. If R is the range of the data and M is the mean squared error of our reconstruction, the PSNR is defined as:

$$\text{PSNR} = 10 \log_{10} \left(\frac{R^2}{M} \right) \tag{4}$$

In general, higher PSNR values indicate higher Reconstruction quality. We also measure the bottleneck distance [8] and Wasserstein distance [12, page 183].

To define the Wasserstein and Bottleneck distances, suppose that f and f' are two scalar fields that yield persistence diagrams D and D' which include their diagonal. If $\varphi : D \to D'$ represents any bijection, and $q \geq 1$, we define the q-Wasserstein distance W_q as:

$$W_q(D, D') = \left[\inf_{\varphi : D \to D'} \sum_{p \in D} ||p - \varphi(p)||_\infty^q \right]^{\frac{1}{q}} \tag{5}$$

For our evaluation, we set $d_W = W_2$ and $d_B = W_\infty$. In particular:

$$W_\infty(D, D') = \inf_{\varphi : D \to W'} \sup_{p \in D} ||p - \varphi(p)||_\infty \tag{6}$$

In general, lower values of W_q for any $q \geq 1$ indicate that the persistence diagrams D and D' are more similar. This in-turn means that the datasets that produced D and D' are more topologically similar.

C Preservation of Vertices of Contour Tree

Volume renderings of data compressed with TTHRESH and Augmented TTHRESH which are analogous to those in Fig. 1 are shown in Fig. 9.

D Parameter Configurations used for Figures and Tables

In order to generate Fig. 1, Fig. 9, and Tab. 2, we needed to identify parameter settings for the absolute error bound parameter η of SZ3 and the RMSE target parameter τ for TTHRESH which produced compression ratios comparable to Augmented SZ3 with $\eta = 0.012$ and TTHRESH with $\tau = 0.012$ after augmenting with $\xi = 0.012$ and $\epsilon = 0.04$. The values for η and τ which produce these comparable compression ratios are reported below.

Dataset	SZ3 η	TTHRESH τ
Earthquake	9.14e-4	3.84e-4
Ionization	3.75e-4	9.89e-6
Isabel	2.58e-3	1.46e-4
Miranda	6.45e-5	4.58e-8
QMCPack	1.76e-4	9.52e-6
Tangaroa	1.88e-4	2.49e-5

Table 6: Parameter configurations used to obtain Fig. 1, Fig. 9, and Tab. 2.

E Distributions of Experimental Results

In our experiments, we measured each evaluation metric on each dataset augmented with each compressor for 11 different evaluation metrics. Here we include boxplots for each combination of a compressor and dataset indicating the distribution of each evaluation metric across all of the parameter configurations. The distributions for various evaluation metrics are shown in the tables below as follows: compression ratio: Tab. 7; PSNR: Tab. 8; Wasserstein distance: Tab. 9; bottleneck distance: Tab. 10; compression time: Tab. 11; decompression time: Tab. 12.

The Miranda dataset produced persistence diagrams that were very large for which d_B and d_W cannot be computed in reasonable time, so we did not calculate d_B and d_W for any trials involving Miranda. In order to compute d_W and d_B for TopoSZ in reasonable time, for all datasets except QMCPack, the d_W and d_B values associated with TopoSZ were calculated after persistent simplification with a threshold of $1.5e - 6$.

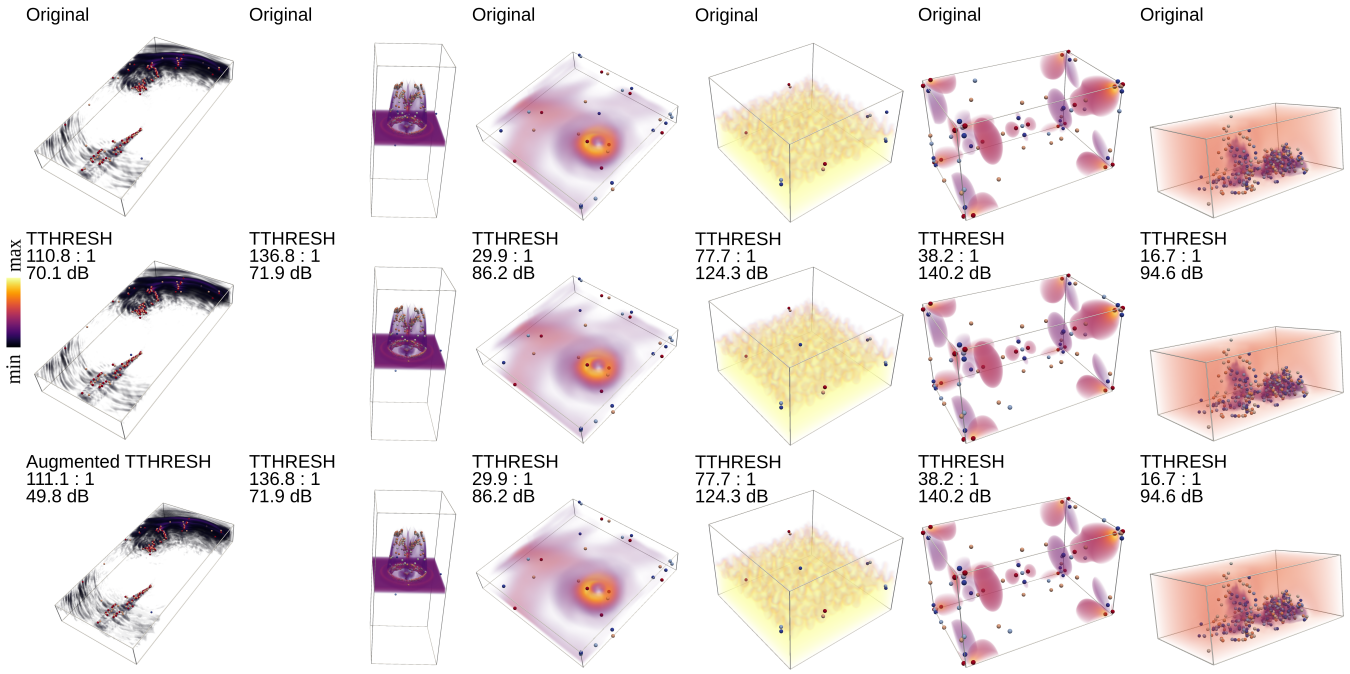


Fig. 9: Scientific datasets with vertices of contour trees when compressed with TTHRESH versus Augmented TTHRESH. (Top row) Vertices of the contour tree from the input data. (Middle row) Vertices of the contour tree from the reconstructed data using TTHRESH. (Bottom row) Vertices of the contour tree from the reconstructed data using augmented TTHRESH. Critical points are colored according to critical type. Red: maximum. Blue: minimum. White: saddle.

Table 7: Distributions of Compression Ratio across all trials.

dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
Earthquake	33.3 101.2 74.5 128.5 95.3 127.3	34.3 111.5 55.8 124.7 96.2 124.0	32.1 45.6 60.6 31.7 43.2 46.2 58.9 44.2 46.1	31.0 90.8 62.4 106.7 79.7 105.3	33.1 95.8 66.3 111.0 84.1 110.2	49.1 104.2 20.0 69.3 35.4 50.1	27.0 70.7 57.2 91.6 69.8 83.6
Ionization	139.3 68.6 318.1 125.5 288.9	147.2 85.0 248.1 116.5 233.5	81.2 61.3 89.6 73.7 89.1	107.1 69.3 248.1 97.8 231.9	121.4 80.2 232.2 109.8 215.1	24.9 13.6 49.0 22.3 35.4	80.5 146.5 58.2 101.7 71.7 101.5
Isabel	81.4 25.6 780.9 76.4 780.5	33.5 15.8 50.0 26.5	21.8 26.3 32.0 20.0 25.1 27.3 31.9 25.3	101.0 39.0 284.4 99.6 284.0	62.8 26.3 195.1 59.1 195.0	18.6 24.1 37.6 30.7 38.7 33.7	21.8 32.1 47.2 22.6 36.6 29.3 34.3
Miranda	166.2 197.0 187.9 215.6 194.7 203.7	18.0 79.5 77.3 81.0 77.8	87.1 106.4 134.0 104.5 106.6 133.9 105.6 106.5	132.5 361.8 126.6 176.6 131.6 160.2	89.1 112.2 127.0 144.3 122.6 134.5 123.8 130.7	95.9 80.6 98.2 90.5 97.9	58.4 72.7 81.4 104.7 72.2 74.9 72.3
QMCPack	46.2 61.5 46.1 64.5 54.1 57.3 62.5	38.6 54.1 29.1 44.0 32.1 39.3	30.7 38.9 30.6 32.0 43.9 36.4 39.5	32.9 24.3 34.3 29.6 33.9	75.4 51.1 79.7 65.9 76.7	26.1 8.6 34.2 11.8 27.9	36.5 45.2 32.4 38.3 34.2 37.4
Tangaroa	31.2 123.2 17.7 68.2 103.6 26.9 55.3	18.1 11.3 36.7 16.7 26.7	28.6 23.1 30.8 27.0 30.3	30.2 91.4 17.9 84.3 29.4 52.9	35.2 23.6 69.0 34.5 50.5	7.5 24.3 40.2 18.7 26.1 37.6 21.5 25.4	18.5 2.1 27.8 2.2 27.3

Table 8: Distributions of PSNR across all trials.

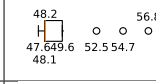
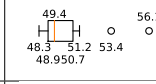
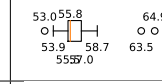
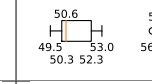
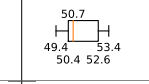
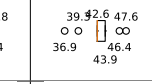
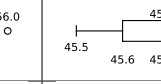
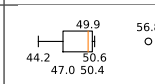
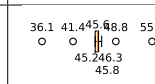
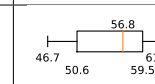
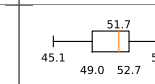
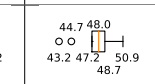
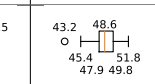
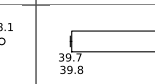
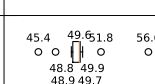
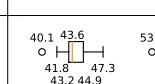
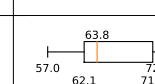
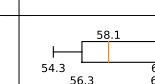
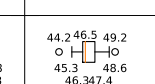
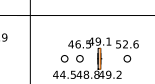
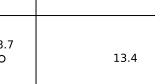
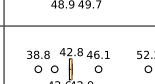
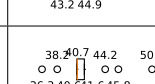
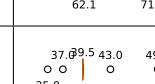
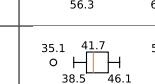
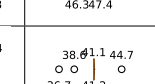
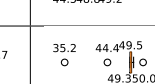
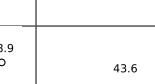
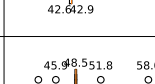
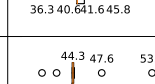
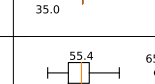
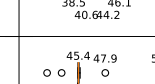
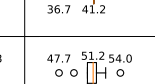
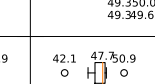
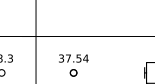
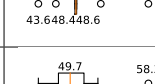
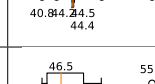
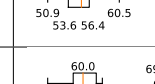
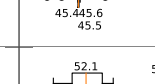
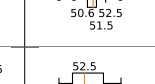
dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
Earthquake							
Ionization							
Isabel							
Miranda							
QMCPack							
Tangaroa							

Table 9: Distributions of Wasserstein Distance across all trials.

dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
Earthquake							
Ionization							
Isabel							
QMCPack							
Tangaroa							

Table 10: Distributions of bottleneck distance across all trials.

dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
Earthquake							
Ionization							
Isabel							
QMCPack							
Tangaroa							

Table 11: Distributions of compression time across all trials. Times measured in seconds.

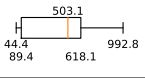
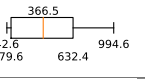
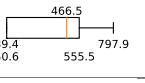
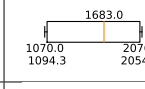
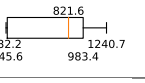
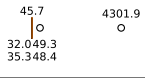
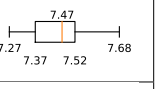
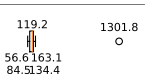
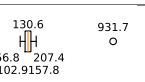
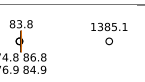
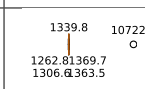
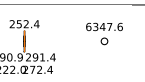
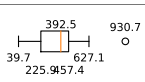
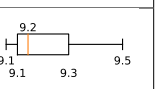
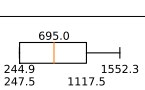
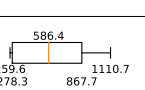
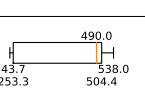
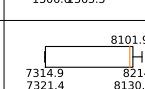
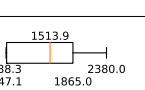
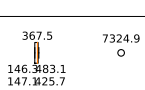
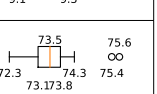
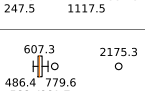
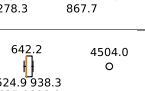
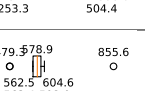
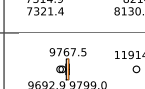
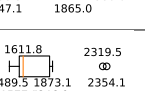
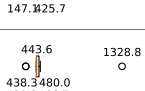
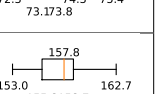
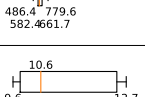
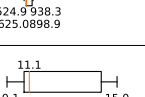
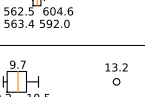
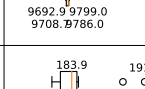
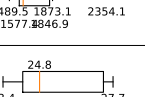
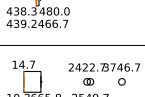
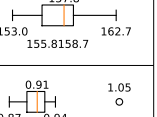
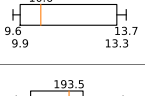
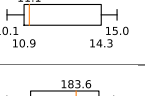
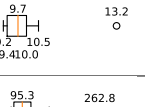
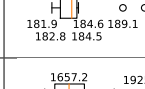
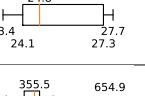
dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
Earthquake							
Ionization							
Isabel							
Miranda							
QMCPack							
Tangaroa							

Table 12: Distributions of decompression time across all trials. Times measured in seconds.

dataset	A-SZ3	A-TTHRESH	A-ZFP	A-Neurcomp	A-CSI	TopoSZ	TopoQZ
Earthquake							
Ionization							
Isabel							
Miranda							
QMCPack							
Tangaroa							