TopoSZ: Preserving Topology in Error-Bounded Lossy Compression

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

Abstract—Existing error-bounded lossy compression techniques control the pointwise error during compression to guarantee the integrity of the decompressed data. However, they typically do not explicitly preserve the topological features in data. When performing post hoc analysis with decompressed data using topological methods, preserving topology in the compression process to obtain topologically consistent and correct scientific insights is desirable. In this paper, we introduce TopoSZ, an error-bounded lossy compression method that preserves the topological features in 2D and 3D scalar fields. Specifically, we aim to preserve the types and locations of local extrema as well as the level set relations among critical points captured by contour trees in the decompressed data. The main idea is to derive topological constraints from contour-tree-induced segmentation from the data domain, and incorporate such constraints with a customized error-controlled quantization strategy from the SZ compressor (version 1.4). Our method allows users to control the pointwise error and the loss of topological features during the compression process with a global error bound and a persistence threshold.

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

1 Introduction

Advances in high-performance computing allow researchers to generate extremely large volumes of scientific data. Such data pose significant challenges for memory resources and transmission bandwidths in scientific analysis and visualization. *Lossy compression*, a class of data compression techniques that involves some loss of information via inexact approximations and partial data discarding, has the potential to address these challenges as it enables a significant reduction in data size. Lossy compression has been widely used in computer graphics (e.g., [50]), scientific visualization (e.g., [19, 53]), storage systems, databases, and software applications; see [15, 36] for surveys.

To preserve data integrity, many scientific applications require *error-bounded* lossy compression to control the extent to which data are altered during compression. For instance, compressors such as SZ [39, 58], ZFP [42], and FPZIP [43] guarantee the pointwise error between the original and the decompressed data. However, a constraint on pointwise error alone may alter the features of interest from the original data. Inconsistencies between features extracted from the original and the decompressed data may lead to inconsistent or incorrect scientific insights during the post hoc analysis of the decompressed data.

Topological descriptors for scalar field data, such as merge trees [10], contour trees [16], Reeb graphs [49], and Morse and Morse-Smale complexes [25, 26, 28], have been widely used in scientific data analysis and visualization. Topological methods based on these descriptors have found applications in science and engineering, such as symmetry detection in materials science [51, 52, 56, 59, 60] and cloud tracking [23, 27, 45] in climate science; see [65] for a survey. Preserving topological descriptors is important if data compression is part of the analysis or storage pipeline within these applications. For example, Yan et al. [64] utilized merge trees to summarize the distribution of ocean eddies. Missed or added topological features during data compression may lead to misleading observations of the climate system.

In this paper, we introduce TopoSZ, an error-bounded lossy compression technique that preserves topological features in 2D and 3D scalar field data. We use the *contour tree*, a topological descriptor that records the connectivity among level sets of a scalar field, to aid our compression process and evaluate its effectiveness. When performing post hoc analysis with decompressed data using topological methods, TopoSZ guarantees topological consistencies between the original and the decompressed data. Specifically, we aim to preserve the types and locations of local extrema as well as the level set relations among critical points captured by contour trees in the decompressed data. The main idea is to derive topological constraints from contour-tree-induced segmentation from the data domain, and incorporate such constraints with a customized error-controlled quantization strategy from the SZ compressor (version 1.4 [58], referred to as SZ-1.4 in the rest of the paper).

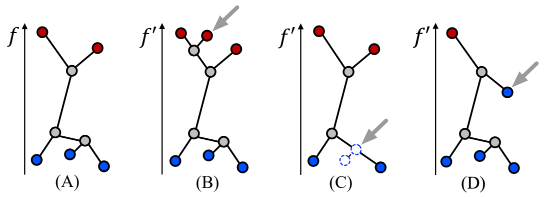


Fig. 1: Examples of false cases (pointed by arrows) in preserving the contour trees during compression. (A) The contour tree that arises from the original scalar field f. (B-D) Contour trees that arise from the decompressed scalar field f': (B) false positive, (C) false negative, and (D) false type.

To quantify the level of topology preservation, we introduce three types of false cases in preserving the topology captured by contour trees, which are inspired by the work of Liang et al. [40]: false positives (FPs), false negatives (FNs), and false types (FTs). A false positive occurs when a new branch appears in the contour tree of the decompressed data, which does not exist in the same position of the contour tree from the original data; c.f. Fig. 1(A) and (B). A false negative occurs when a branch of the contour tree from the original data is missing from the contour tree of the decompressed data; c.f. Fig. 1(A) and (C). A false type occurs when the corresponding critical points do not match in type between the contour trees of the original and the decompressed data; c.f. Fig. 1(A) and (D). Given a persistent simplification threshold ε that controls the importance of features to be preserved, our method preserves a simplified contour tree from the original data without any FPs, FNs, and FTs. In this paper, we aim to preserve topological features and retain pointwise error control in the lossy compression of 2D and 3D scalar fields. Our contributions include:

- A contour-tree-based iterative strategy to derive topological constraints as lower and upper bounds for lossy compression;

- *Lin Yan is with Argonne National Laboratory. E-mail: lyan@anl.gov*
- *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.*
- *Bei Wang is with the University of Utah. E-mail: beiwang@sci.utah.edu*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

- A customized error-controlled quantization strategy to incorporate topological constraints into the data compression process;
- Comparison with the state-of-the-art error-bounded lossy compressors in terms of compression ratio, PSNR (peak signal-to-noise ratio), and topological metrics, using a number of scientific datasets.

2 Related Work

Error-bounded lossy compression. Data compression techniques can be classified based on information loss during data reduction, namely, lossless and lossy compression. Lossless compressors, such as [12, 13, 34, 62], ensure that the decompressed (reconstructed) data inherit all properties of the original data; however, these techniques often suffer from insufficient compression rates in practice. Lossy compressors usually achieve a higher data reduction ratio than lossless compressors with an acceptable amount of information loss. For example, lossless compression ratios range from about 1.5 to 3 for image compression, whereas lossy compression techniques offer compression ratios in excess of 20 with virtually no loss in visual fidelity. Lossy compressors can be further categorized into non-error-bounded and error-bounded ones. This paper focuses on error-bounded lossy compressors since they usually deliver higher compression ratios than the lossless ones and are more precise than the non-error-bounded lossy compressors.

Error-bounded lossy compression may be classified as truncation-based, prediction-based, and transformation-based methods. Truncation-based error-bounded lossy compressors include the work of Gong et al. [30]. The authors designed MLOC for large scientific analysis by truncating double-precision floating-point numbers into multiple precision levels, which are further used to support data access with certain resolution requirements. An example of a prediction-based method is ISABELA [37], which transfers data points to smooth curves using B-splines for prediction tasks. Another prediction-based method is FPZIP [43], which uses the Lorenzo predictor [35] to estimate the value on an unknown vertex based on known cube vertices. Both the predicted and the actual values are mapped to an integer representation to avoid underflow, together with arithmetic coding for residuals. SZ [39, 41, 58, 67, 68] is a family of prediction-based compressors featuring on an adaptive selection of the best-fit prediction methods (e.g., Lorenzo predictor, regression-based predictors, and interpolation-based predictors). Instead of using arithmetic coding for residuals as FPZIP does, SZ performs a linear-scaling quantization [58] to convert the difference between the predicted value and the original value for each datapoint to an integer. These quantization integers are encoded by customized Huffman coding and can be further compressed by a lossless compressor such as ZSTD [6] and GZIP [3]. ZFP [42] is a transformation-based error-bounded compressor. It uses a custom orthogonal block transform to de-correlate the values of blocks and encodes the transform coefficients for compression.

In this paper, we customize the SZ-1.4 compressor to preserve topology, since several prior studies [22, 38, 68] have verified that SZ yields the best compression quality under the same error bound among all the prediction-based compressors. However, in general, any prediction-based compressor can be customized for topology-preserving compression using our framework.

Topology-preserving compression. To the best of our knowledge, Soler et al. [55] presented the first and only compression technique for scalar field data with some topological guarantees. Their technique, referred to as TopoQZ in this paper, is developed based on a topologically adaptive quantization of the data range. In particular, TopoQZ supports a controlled loss of topological features. That is, given a persistent threshold ε that quantifies the target feature size, TopoQZ "preserves the critical point pairs with persistence greater than ε and destroys all pairs with a smaller persistence." [55]

TopoSZ inherits the compression pipeline in SZ (SZ-1.4 [58]) and modifies the quantization process (see Sec. 4.2 for details). Thus, our framework differs significantly from TopoQZ [55]. First, TopoQZ does not enforce pointwise error control. TopoSZ controls pointwise error explicitly, thus ensuring the pointwise compression quality besides topology preservation. Second, TopoQZ uses a quantization of the scalar field, whereas TopoSZ uses a customized error-controlled quantization encoder to reduce data size significantly during the compression process. We give a detailed comparison between TopoSZ and TopoQZ in terms of compression qualities and compression ratios in Sec. 5.

Contour trees. We use contour trees, which capture the connectivity among level sets of scalar fields, to design and evaluate our topology-preserving compression technique. Contour trees have been used to detect symmetry in bimolecular structures [59], identify and track the oxy-deoxy process in hemoglobin dynamics [54], and design shape matching algorithms for electrostatic potentials [66]. Contour trees are also used in studying scalar field ensembles, such as their distributions [66], summarization [44], and uncertainty visualization [63].

Mesh simplification is another effective way to reduce data for analysis and storage. Chiang and Lu [18] introduced a mesh simplification technique that preserves isosurfaces with geometric error control. However, we consider mesh simplification and compression to be fundamentally different approaches in data reduction. Therefore, we do not compare against mesh simplification in this paper.

3 Technical Background

Our approach has four technical gradients: SZ-1.4, contour tree, persistence simplification, and contour-tree-induced segmentation.

3.1 SZ-1.4 Compressor

We review the SZ-1.4 compressor and customize it in Sec. 4.2 to build TopoSZ. A newer version of the SZ compressor (referred to as SZ3 [41, 67]) uses an interpolation algorithm to improve the compression ratios when the error bound is relatively high, and switches to the Lorenzo predictor when the error bound becomes low. Since TopoSZ usually targets the high-precision (low-error-bound) cases, we directly use the Lorenzo predictor in our implementation, which is part of SZ-1.4.

An overview of SZ-1.4. SZ-1.4 is a prediction-based lossy compressor designed for scientific data that strictly controls the pointwise compression error. That is, a global error bound ξ guarantees $|f(x) - f'(x)| < \xi$, where $f(x)$ and $f'(x)$ are the original and the decompressed values of any datapoint x in the domain. The implementation of SZ-1.4 involves four steps: data prediction, linear-scaling quantization, customized variable-length encoding, and lossless compression.

Step 1: data prediction. SZ-1.4 predicts the value of each datapoint with a *Lorenzo predictor* [35] based on the decompressed values of its neighbors.

Step 2: linear-scaling quantization. SZ uses linear quantization on the difference between the original and the predicted values for each datapoint. In SZ-1.4 [58], the difference is converted to an integer based on the global error bound ξ.

Step 3: customized variable-length coding. SZ-1.4 encodes the quantization integers from the previous step by a customized Huffman encoding [7] to reduce the data size for storage.

Step 4: lossless compression. In the last step, SZ-1.4 further compresses the unpredictable data from step 2 and the encoded quantization from step 3 by a lossless compressor ZSTD [6].

Linear-Scaling quantization. The key differences between TopoSZ and SZ-1.4 are that TopoSZ derives the lower/upper bounds for each datapoint, and it revises the quantization method of SZ-1.4 (referred to as step 2) to guide the quantization process for topology preservation. Therefore, we review an SZ-1.4 implemented by Tao et al. that introduces the notion of an error-controlled *quantization encoder*; see [58] for details. Let ξ denote the global error bound. The design of the SZ-1.4 quantization encoder is shown in Fig. 2. Let m denote the number of bits needed to encode $2^m - 1$ number of quantization codes. We set $m = 16$.

For a datapoint $x \in \mathbb{X}$, we first compute its *1st-phase predicted value* by using a prediction model. For example, Tao et al. used a multilayer prediction model (e.g., a Lorenzo predictor is a special case of the multilayer predictor). Such a value is presented by a red point in Fig. 2. Then, we generate a number of *2nd-phase predicted values*

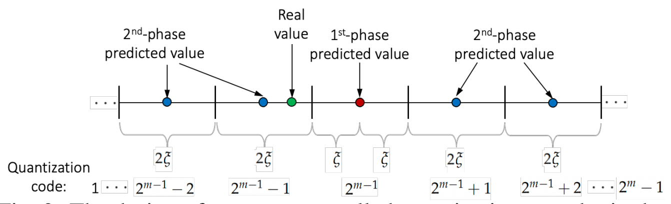


Fig. 2: The design of an error-controlled quantization encoder in the SZ-1.4 based on a linear scaling of the error bounds. Image reproduced from [58, Fig. 2].

by scaling the error bound linearly. These values are represented by the blue points in Fig. 2. The distance between any two adjacent 2nd-phase predicted values equals 2ξ. By construction, each (1st-phase or 2nd-phase) predicted value lies in the middle of an interval with the length of 2ξ, and these intervals do not overlap. In total, there are $2^m - 1$ intervals indexed by $\{1, \dots, 2^m - 1\}$. The 1st-phase predicted value (red point) falls in the interval indexed by 2^{m-1}.

In SZ-1.4, when the real value of the datapoint $f(x)$ (green point) falls into a certain interval, we mark it as being *predictable* and use its corresponding predicted value from the same interval to represent its value $f'(x)$ in the compression. By construction, $|f(x) - f'(x)| \leq \xi$. However, if $f(x)$ does not fall into any interval, we mark the datapoint as being *unpredictable*.

Since there are $2^m - 1$ intervals, each predicted value can be encoded by the index of its corresponding interval. Then "we use the codes of $1, \dots, 2^m - 1$ to encode predictable data and use the code of 0 to encode unpredictable data" [58]. This process is referred to as the quantization encoding.

3.2 Contour Tree and Persistence Simplification

Given a scalar field $f : \mathbb{X} \to \mathbb{R}$ defined on a simply connected domain $\mathbb{X}$, a *contour* is a connected component of the *level set* $\mathbb{X}_t = f^{-1}(t)$ of f, for some $t \in \mathbb{R}$. A contour tree captures the connectivity among the contours of f. Two points $x, y \in \mathbb{X}$ are *equivalent*, denoted as $x \sim y$, if $f(x) = f(y) = t$, and x and y belong to the same contour of f. The *contour tree*, $T(\mathbb{X}, f) = \mathbb{X}/\sim$, is the quotient space obtained by identifying equivalent points. For a point $x \in \mathbb{X}$, all points $y \in \mathbb{X}$ that are equivalent to x forms its *equivalence class*.

As t increases in the range of f, a contour tree captures the evolution of the contours of f; see [24, Sect. VI.4] for a formal treatment. Intuitively, the contour tree of f is the graph obtained by a continuous contraction of each contour of f to a single point. Nodes in the contour tree have a one-to-one correspondence with the *critical points* of f, at which the topology of the contours changes. Assume f is a Morse function, and critical points are locations where the derivative of f is zero. For a 2D scalar field, critical points are local maxima, local minima, and saddles. Fig. 3 gives an example. Blue nodes represent the creation of a component at a local minimum, gray nodes (saddles) represent the merging or splitting of components, and red nodes represent the disappearance of a component at a local maximum; see a straight-line drawing in Fig. 3(C). We display a scalar field with its corresponding contour tree embedded in the scalar field and the graph of the scalar field; see Fig. 3(A) and (B), respectively. For a 3D scalar field, critical points are local maxima, local minima, and saddles of order 1 and 2. Local maxima and local minima are referred to as local extrema. An *edge* in a contour tree is a line joining a pair of critical points.

A contour tree of real-world data typically contains a number of tiny branches due to the noise in the data. A persistence-based [9, 17, 31] simplification strategy may be used to simplify a contour tree, which also leads to the simplification of the scalar field [17, 46, 57]. Fig. 3 (E-F) illustrates the persistence simplification of a contour tree.

3.3 Contour Tree Induced Segmentation

By construction, a contour of a scalar field $f : \mathbb{X} \to \mathbb{R}$ maps to a point in the contour tree $T := T(\mathbb{X}, f)$. f can be decomposed into $f = \psi \circ \phi$, where $\phi : \mathbb{X} \to T$ maps each point in $\mathbb{X}$ to its equivalence class in T and $\psi : T \to \mathbb{R}$ maps each point in T to its f value (see Fig. 4).

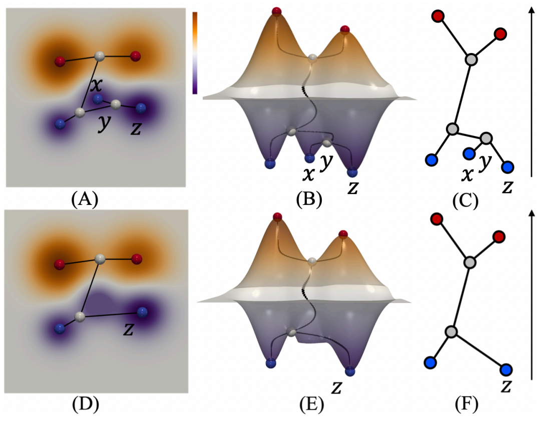
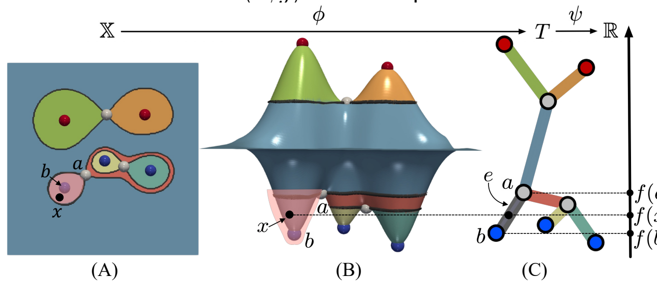


Fig. 3: A contour tree of a 2D scalar field f generated by a mixture of Gaussian functions. (A) The scalar field visualized with an embedded contour tree T. (B) The graph of f—the set of all ordered pairs $(x, f(x))$—is visualized together with the corresponding contour tree of f. (C) Abstract (straight-line) visualization of the contour tree T where nodes are equipped with scalar field values. (D)-(F) The minimum-saddle branch (x, y) of T is simplified.



$\mathbb{X} \xrightarrow{\phi} T \xrightarrow{\psi} \mathbb{R}$
a
b
x
e
f(a)
f(x)
f(b)
(A)
(B)
(C)

Fig. 4: A segmentation induced by the contour tree in Fig. 3. (A) f, (B) the graph of f, and (C) the contour tree T. Edges of the contour tree in (C) and their pre-images (i.e., the topological regions) in (A) and (B) are shown with the same color.

The pre-image of ϕ at critical points induces a partition of $\mathbb{X}$. By construction, the pre-image $\phi^{-1}(e)$ of an edge $e \in T$ is connected and its function values vary monotonically. The pre-images of all edges in the contour tree thus give rise to a partition of $\mathbb{X}$, referred to as the *contour tree induced segmentation*, and each region in the partition is referred to as a *topological region* (see Fig. 4A). A topological region is monotonic and contains only regular points in its interior. Furthermore, we can apply persistence simplification of the contour tree T using branch decomposition, which is equivalent to iteratively removing its shortest (by height) leaf-connecting edges.

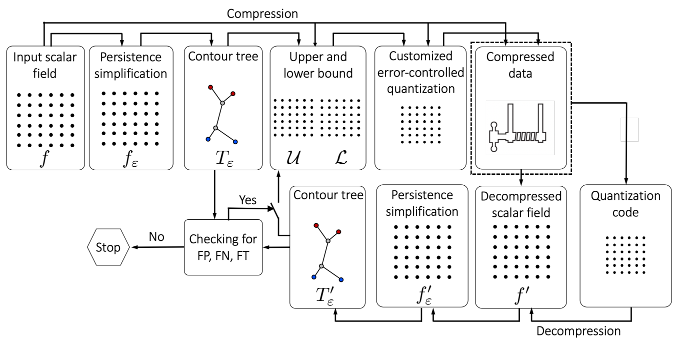


Fig. 5: TopoSZ pipeline for topology-preserving compression.

4 Method

In this section, we propose a novel lossy compressor—TopoSZ—that preserves the topology of data. TopoSZ modifies the SZ compressor (version 1.4) by explicitly imposing pointwise upper and lower error bounds that encode topological constraints derived from contour trees.

Notations. Let ξ and ε be the user-defined global error bound and the persistence threshold, respectively. Let $f : \mathbb{X} \to \mathbb{R}$ denote the original scalar field and f_ε its ε-simplification. Similarly, let f' denote the decompressed scalar field and f'_ε its ε-simplification. Let T_ε and T'_ε be the contour trees of f_ε and f'_ε, respectively. In practice, we assume f is defined on $\mathbb{X} := \{x_1, \dots, x_N\}$, a finite set of sampled datapoints. The input to TopoSZ includes f, ξ, and ε.

Overview. TopoSZ compresses the original scalar field f such that the decompressed scalar field f' contains the same set of local extrema and gives rise to the same contour tree up to persistent simplification. In other words, TopoSZ ensures that f_ε and f'_ε contain the same set of local extrema, and $T_\varepsilon = T'_\varepsilon$, i.e., T'_ε is free from false cases.

An overview of our pipeline is shown in Fig. 5. First, we impose topological constraints by computing the upper and lower error bounds $\mathcal{U} := (u_1, ..., u_N)$ and $\mathcal{L} := (l_1, ..., l_N)$ for all datapoints, using the topological regions defined by the contour-tree-induced segmentation (Sec. 3.3). These upper and lower bounds quantify the range of functional perturbations that do not change the level set relations and thus maintain topology during compression.

During the compression process, TopoSZ takes $\mathcal{U}$, $\mathcal{L}$ and ξ as input. We compare the contour tree T'_ε after decompression with the original contour tree T_ε. If no false cases occur, TopoSZ stops the compression process. Otherwise, f'_ε and its contour tree T'_ε are used as input for another round of compression with updated upper and lower bounds for a subset of datapoints until all false cases are eliminated from the decompressed data; see Sec. 4.1 for details.

Second, we propose in Sec. 4.2 a novel error-controlled quantization encoder for TopoSZ by modifying the one from the SZ-1.4. Using such a customization, TopoSZ takes as input pointwise upper and lower error bounds $\mathcal{U}$ and $\mathcal{L}$ together with a global error bound ξ, and controls the compression error for every datapoint.

4.1 Computing Contour-Tree-Based Error Bounds

To preserve the contour tree during compression, we need to preserve the locations and types of local extrema by eliminating false positives (FPs), false negatives (FNs), and false types (FTs). The key idea is to use contour-tree-induced segmentation (Sec. 3.3) to provide fine-grained controls of error bounds in topological regions. In particular, we preserve the critical points and the monotonicity of the scalar field in topological regions up to persistence simplification. Computing the upper and lower bounds for TopoSZ involves two key steps: initialization and iteration. Iteration may not be needed if the initialization step eliminate all false cases.

4.1.1 Initialization

As illustrated in Fig. 4, given an edge e of the contour tree T, let $a, b \in \mathbb{X}$ denote the critical points of f in $\mathbb{X}$ that correspond to the endpoints of e. Without loss of generality, we suppose $f(a) > f(b)$. For a point $x \in \mathbb{X}$ in the pre-image $\phi^{-1}(e)$, its lower bound l and upper bound u are defined as

$$\begin{cases} l = f(b), u = f(a), & \forall x \in \phi^{-1}(e), x \neq a, x \neq b \\ l = f(x), u = f(x), & x = a \text{ or } b \end{cases} \tag{1}$$

By definition, regular points in $\phi^{-1}(e)$ share the same lower and upper bounds, whereas critical points retain their original values. See Fig. 4(B) for an example, where regular points in the pink region use $f(b)$ and $f(a)$ as their initial lower and upper bounds. For all datapoints $\{x_1, \dots, x_N\}$ in the domain, we define their pointwise upper and lower bounds as high-dimensional vectors, i.e., $\mathcal{U} = (u_1, ..., u_N)$ and $\mathcal{L} = (l_1, ..., l_N)$.

A toy example. We use f from Fig. 3 as an example with a persistence threshold $\varepsilon = 0.1$. The initial upper and lower bounds $\mathcal{U}$ and $\mathcal{L}$ are computed via Eqn. (1) based on the simplified contour tree T_ε. The boundaries of $\mathcal{U}$ and $\mathcal{L}$ follow the boundaries of the topological regions induced by T_ε (c.f. Fig. 4A, assuming T to be T_ε in Fig. 4C). Using $\mathcal{U}$ and $\mathcal{L}$ as pointwise error bounds for TopoSZ, we study the decompressed scalar field f'_ε. We observe two FPs in the contour tree T'_ε of f'_ε. Indeed, using the customized SZ compressor with initial pointwise lower and upper bounds does not necessarily eliminate all false cases. Our next step is to iteratively apply the customized SZ by updating the upper and lower bounds until no false cases are detected.

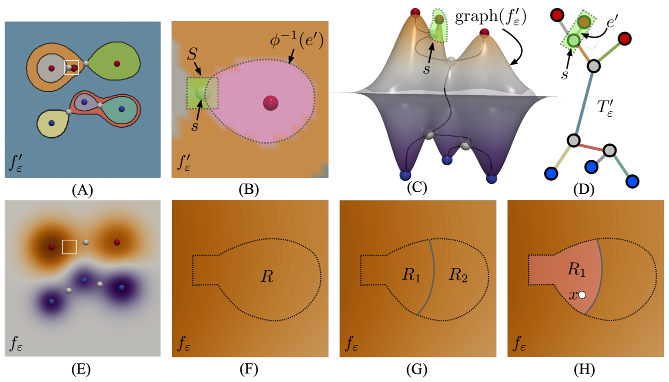


Fig. 6: Updating the lower and upper bounds when an FP occurs. (A) A decompressed scalar field f'_ε colored by topological regions induced by T'_ε. (B) The zoomed-in view of the white square from (A), where an FP occurs. (C) The graph of f'_ε. (D) The contour tree T'_ε. (E) The original simplified scalar field f_ε colored by the f_ε values. (F) The zoomed-in view of the white square from (E) that contains the region S where the upper and lower bounds need to be updated. (G) A refinement of S into two monotonic regions according to P2. (H) The lower and upper bounds for x are decided by the partition x belongs to.

4.1.2 Eliminating False Cases With Iterations

Before introducing our iterative strategy, we first define the multilayer neighborhoods around a given point $x \in \mathbb{X}$, inspired by the multilayer prediction model in [58].

Multilayer neighborhoods. Consider a 2D scalar field defined on a uniform grid. The multilayer neighborhoods of a datapoint x is defined such that the 0-layer neighborhood of x is the datapoint itself, the 1-layer neighborhood includes its 8 neighbors, and so on. A datapoint could grow its k-layer neighborhood (for any positive integer k) until it reaches the domain boundary.

1st iteration: dealing with false positives. During the 1st iteration, if an FP is detected from the decompressed data f'_ε, we find its corresponding edge in T'_ε that is not in T_ε. As shown in Fig. 6(D), an FP occurs at $e' \in T'_\varepsilon$ (c.f., Fig. 3C). We then update the upper and lower bounds for a subset of datapoints in $\mathbb{X}$ for the 1st iteration as follows:

<u>P1.</u> Let s be a point in $\mathbb{X}$ that maps to the saddle point of e'. We first combine s with its 1st-layer neighborhood to form a region S; see the green square of Fig. 6(B). We then combine S with the pre-image $\phi^{-1}(e')$ (pink region in Fig. 6B) to form a region R; see Fig. 6(F). For all datapoints in R, we need to update their upper and lower bounds using their values in the original simplified scalar field f_ε to eliminate the FP involving s. Notice that s is a saddle point in f'_ε but a regular point in f_ε.

<u>P2.</u> Since this is the 1st iteration, we refine the region R by evenly dividing the datapoints in R into two subregions R_1 and R_2 based on the original scalar field f_ε. Let $\min(R)$ and $\max(R)$ denote the minimum and maximum f_ε values among datapoints in R. The refinement of R satisfies two conditions: (1) R_1 and R_2 are monotonic in f; and (2) $\max(R_1) \leq \min(R_2)$; see Fig. 6(G).

<u>P3.</u> For any regular point x in the refined region R_1, its new upper bound and lower bound are updated to be $u := \max(R_1)$ and $l := \min(R_1) = f(s)$; see Fig. 6(H). The upper and lower bounds for datapoints in R_2 are updated similarly.

1st iteration: dealing with false negatives. During the 1st iteration, if an FN is detected from f'_ε, we find the corresponding edge $e \in T_\varepsilon$ that is missing in T'_ε. As shown in Fig. 7(A), an FN occurs w.r.t. Fig. 7(C). We then update the upper and lower bounds for a subset of datapoints

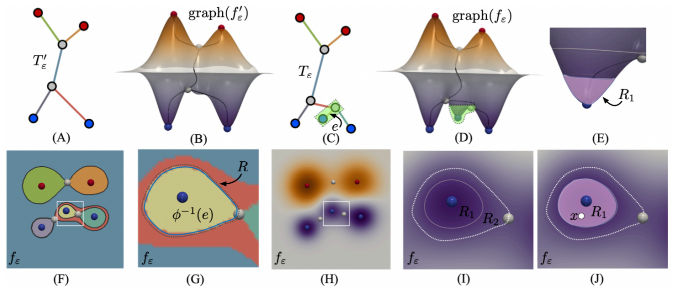


Fig. 7: Updating the lower and upper bounds when an FN occurs. The contour tree T'_ε (A) and the graph (B) of a 2D decompressed scalar field f'_ε. The contour tree T_ε (C) and the graph of a 2D scalar field f_ε (D), together with the T_ε induced segmentation (F). (G) The zoomed-in view of (F), where an FN occurs. (I) A refinement of the region R according to P2. (E), (I) and (J) are the zoomed-in views of the white box in (H). The lower and upper bounds for x are decided by the partition x belongs to.

in $\mathbb{X}$ as follows:

N1. Let R represents the set of datapoints that combines the pre-image $\phi^{-1}(e)$ (yellow region) and their 1-layer neighborhood; see the region enclosed by a dotted line in Fig. 7(G).

N2. We refine R into two regions R_1 and R_2 following P2; see Fig. 7 (I) for an example. We then update the upper and lower bounds for datapoints in R following P3. For example, any regular point x in the refined region R_1 has updated upper and lower bounds as $u := \max(R_1)$ and $l := \min(R_1)$; see Fig. 7(E) and (J).

1st iteration: dealing with false types. During the 1st iteration, if an FT is detected from f'_ε, we find the corresponding edge $e \in T_\varepsilon$ whose end point (a local extremum) has a wrong type. An FT is dealt with similarly to the scenario involving a FN and is therefore omitted from the discussion.

k-th iteration: dealing with false cases. We have described the region refinement process for the 1st iteration. During the k-th iteration (for $k \geq 1$), to eliminate false cases, we will consider the k-layer neighborhood of s in P1 as well as the k-layer neighborhood of $\phi^{-1}(e)$ in N1. We then refine the region R in P2 by dividing it into $k+1$ subregions $R_1, \ldots, R_{k+1}$ such that (1) each R_i is monotonic in f, for $1 \leq i \leq k+1$; and (2) $\max(R_i) \leq \min(R_{i+1})$, for $1 \leq i \leq k$. We update the upper and lower bounds in P3 according to these $k+1$ subregions. By design, the refinement process gets aggressive over time as k increases to eliminate false cases. We provide a discussion on the worst-case performance of iterations in Appendix E.

4.2 A Customized Error-Controlled Quantization

To incorporate topological constraints—defined by the pointwise upper bound $\mathcal{U} := [u_1, \ldots, u_N]$ and the lower bound $\mathcal{L} := [l_1, \ldots, l_N]$—into the data compression process, we propose a novel error-controlled quantization encoder for TopoSZ by modifying the one from the SZ-1.4 in Sec. 3.1 (see a detailed description in [58]).

For TopoSZ, assume we are given the global error bound ξ and a persistence threshold ε. For a datapoint $x \in \mathbb{X}$ with an original value $f(x)$, we first calculate its lower error bound l and upper error bound u using the approach described in Sec. 4.1.

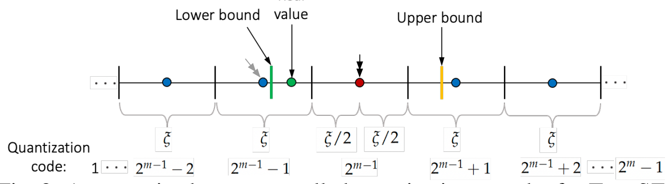


Fig. 8: A customized error-controlled quantization encoder for TopoSZ based on the upper and lower bounds.

In the original work of Tao et al. [58], the distance between any two adjacent 2nd-phase predicted values equals 2ξ; however, in our setting, this distance is set to be ξ. In TopoSZ, we need to consider the location of l (green bar) and u (yellow bar) w.r.t. to the $2^m - 1$ intervals that contain 1st- and 2nd-phase predicted values. Any predicted value smaller than l or greater than u is not considered. As illustrated in Fig. 8, when the real value of the datapoint $f(x)$ (green point) falls into a certain interval, the predicted values that are close to the real value in both directions are considered (gray and black double arrows). In the illustrated example, both predicted values in the $(2^{m-1}-1)$-th and 2^{m-1}-th intervals serve as candidates, and both are less than ξ away from the real value. However, only the predicted value in the (2^{m-1})-th interval falls inside $[l, u]$ (black double arrow) and is selected. Therefore, x will have 2^{m-1} as its quantification code under TopoSZ. An unpredictable datapoint and its quantification code are sent to a lossless compressor for further compression.

5 Experimental Results

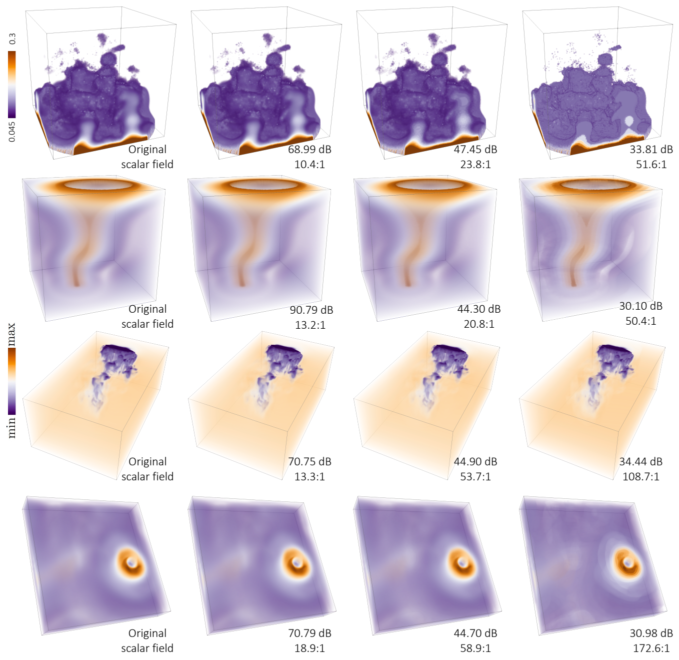


Fig. 9: 3D visualization for the Viscous Fingers (1st row), Tornado (2nd row), Tangaroa (3rd row), and Isabel (4th row) datasets. Columns 1-4: the original data, the decompressed data with high, medium, and low quality, respectively. We also report the PSNR and the compression ratio for each decompressed dataset.

We present experimental results using two 2D and four 3D scientific datasets described in Table 2. The details of these datasets are described in Appendix A.

Preprocessing. For each dataset, the original scalar field is normalized to $[0, 1]$ and saved using 32-bit floating points. Such a normalization helps us specify the persistence simplification level and pointwise error bound. For example, $\varepsilon = 0.1$ means a 10% persistence simplification of the range of the scalar field, and $\xi = 0.01$ means a 1% maximum pointwise error bound. For computing the contour tree and applying persistence simplification, we use algorithms by Tierny and Pascucci [62] and Gueunet et al. [32], with implementations available in the Topology ToolKit (TTK) [61]. All results are obtained on a desktop computer with an Intel Core i7 CPU (2.8 GHz, 4 cores) and 16 GB RAM.

Evaluation metrics. We explore a number of metrics, including the number of false cases (in the decompressed data), data compression ratio, PSNR, bottleneck [20] and Wasserstein distances [24, page 183] between f and f'; see the supplementary material for a review.

Overview of results. We first give a snapshot of compression capabilities of TopoSZ in Sec. 5.1, in comparison with other error-bounded com-

Table 1: Comparing run time between TopoSZ and other lossy-compressors, including SZ3, ZFP, FPZIP, TTHRESH, and TopoQZ. Compression of TopoSZ contains two parts: initialization and iteration. CT, UBLB, and CSZ-1.4 represent the run time for computing the simplified contour tree with its induced segmentation, updating the lower and upper bound, and applying a customized SZ-1.4. # means the number of iterations. CM and DC stand for compression and decompression, respectively. All experiments use a global error bound $\xi = 0.06$. For TopoSZ and TopoQZ, the persistence threshold $\varepsilon = 0.06$. All times are in seconds.

Dataset	**Initialization**			**Iteration**				**DC**	**SZ3**		**ZFP**		**FPZIP**		**TTHRESH**		**TopoQZ**	
	CT	**UBLB**	**CSZ-1.4**	**CT**	**UBLB**	**CSZ-1.4**	**#**		**CM**	**DC**	**CM**	**DC**	**CM**	**DC**	**CM**	**DC**	**CM**	**DC**
Heated Flow	3.59	0.06	0.01	-	-	-	0	0.003	0.003	7.9e-4	0.001	9.1e-4	0.002	2.09e-3	0.20	0.005	3.12	1.68
E3SM Wind	12.18	0.93	0.04	7.09	0.50	0.04	4	0.02	0.03	0.01	0.02	0.02	0.04	0.04	2.73	0.36	5.11	3.93
Tornado	51.13	1.78	0.07	-	-	-	0	0.03	0.05	0.02	0.03	0.02	0.05	0.06	0.65	0.47	7.13	4.57
Viscous Fingers	21.68	1.83	0.04	11.76	0.95	0.07	7	0.04	0.06	0.02	0.03	0.02	0.06	0.07	0.67	0.51	6.70	4.29
Tangaroa	83.79	5.62	0.20	49.53	2.61	0.20	3	0.1	0.19	0.08	0.11	0.06	0.13	0.14	1.72	0.96	27.36	12.22
Isabel	286.8	19.31	0.74	142.7	8.78	0.72	1	0.31	0.51	0.18	0.32	0.14	0.54	0.62	9.08	4.80	114.95	44.39

Table 2: Datasets used in our experiments.

Dataset	**Dimensions**	**Size**
Heated Flow	150×450	270 KB
E3SM Wind	1440×720	4.1 MB
Viscous Fingers	$128 \times 128 \times 128$	8.4 MB
Tornado	$128 \times 128 \times 128$	8.4 MB
Tangaroa	$300 \times 180 \times 120$	25.9 MB
Isabel	$500 \times 500 \times 90$	90 MB

pressors such as SZ-1.4, SZ3, ZFP, FPZIP, and TTHRESH; since these compressors are topology-agnostic, they typically lead to worse topology preservation by nature. Fig. 10 shows that TopoSZ consistently preserves topology of the input scalar field without introducing false cases. We then dive deeper into the algorithmic process of TopoSZ by visualizing intermediate decompressed data during iterations (Sec. 5.2), and study how its performance is affected by the parameter setting (Sec. 5.3). We then compare TopoSZ against TopoQZ, the compression framework most relevant to ours that also offers some topological guarantees (see Sec. 5.4). We conclude with a detailed run time analysis of TopoSZ in Sec. 5.5. See Appendix D for an analysis of the compression quality w.r.t. the size of data using larger datasets in comparison with those described in Table 2.

5.1 A Snapshot of Compression Capabilities

Fig. 10 gives a snapshot of the compression results on the Viscous Fingers dataset, comparing TopoSZ against other state-of-the-art lossy compressors that do not focus on topology preservation (i.e., SZ-1.4, SZ3, ZFP, FPZIP, and TTHRESH). We set a persistence threshold $\varepsilon = 0.12$. For experimental purposes, we can use any persistence threshold. However, we recommend setting $\xi > \varepsilon$, since a smaller global error bound leads to smaller topological regions that need fine-grained control, and, therefore, less iterations and a higher compression ratio. We choose the current threshold that preserves a sufficient number of critical points while eliminating a certain amount of noise in the data, and is slightly larger than the largest global error bound tested in this experiment; see Appendix C for a detail discussion on the persistence threshold and compression capabilities of TopoSZ. As shown in Fig. 10 (A), TopoSZ generates a decompressed scalar field that preserves the simplified contour tree from the original scalar field without any false cases. In contrast, there are false cases within the decompressed data using other compressors, unless we specify a very small global error bound (e.g., 10^{-5} for FPZIP). These small error bounds indicate lossless compression for these off-the-shelf compressors.

We also present the rate-distortion plots of all the evaluated compressors in Fig. 10 (B) and (C), using PSNR and the number of false cases as the distortion metrics, respectively. Rate-distortion is a widely used assessment metric in the compression community, where rate represents the average number of bits in the compressed data (e.g., it can be computed by 32 over the compression ratio for single-precision floating point data). According to these two figures, TopoSZ has a slightly worse rate distortion in terms of PSNR compared to SZ3 and ZFP, but it outperforms all the error-bounded compressors on preserving topology.

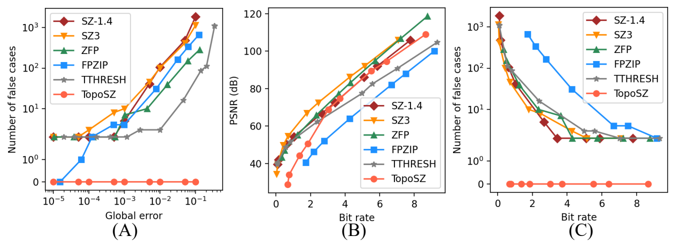


Fig. 10: Test lossy compressors with the Viscous Fingers dataset: (A) the number of false cases w.r.t. global error bound; (B) and (C): the PSNR and the number of false cases w.r.t. bit rate (i.e., average bits per compressed data sample). Lossy compressors include TopoSZ, SZ-1.4, SZ3, ZFP, FPZIP, and TTHRESH.

5.2 Improved Compression Quality with Iterations

TopoSZ couples a pointwise error-controlled compressor with topological constraints. It repeatedly applies a customized SZ (see Sec. 4.2) with updated lower and upper error bounds until no false cases are detected. We demonstrate that TopoSZ improves the compression quality during iterations using the Heated Flow, E3SM Wind, and Viscous Fingers datasets.

First, TopoSZ eliminates false cases during iterations. As shown in Fig. 12, given a persistence threshold $\varepsilon = 0.02$ and global error bound $\xi = 0.05$, the initialization gives rise to a number of false cases with the initial upper and lower bounds (F); some of these false cases are enclosed in the yellow boxes in (B). With updated upper and lower bounds (see Sec. 4.1.2), the 1st iteration eliminates most false cases; c.f. (B) and (C). However, a few false positives remain in the yellow box of (C) after the 1st iteration. Therefore, TopoSZ goes through two more rounds of iterations with updated error bounds. For the Heated Flow dataset, TopoSZ eliminates all false cases after iteration 3, thus preserving the types and positions of local extrema from the original data (E).

Second, TopoSZ provides a finer grained control of pointwise errors between the original and the decompressed data during each iteration. We apply TopoSZ to the E3SM Wind dataset with $\varepsilon = 0.06$ and $\xi = 0.02$. As shown in Fig. 11, we visualize the decompressed data and its absolute pointwise differences w.r.t. the original data for selected iterations. We observe that, with an increasing number of iterations, certain regions in the domain obtain lower pointwise errors compared with the previous iterations, for instance, the ones enclosed by the yellow boxes. TopoSZ updates the lower and upper bounds wherever false cases occur during iterations, and these updated bounds help decrease the errors within these regions in the decompressed data. Consequently, TopoSZ improves the compression quality in terms of pointwise error during iterations.

Third, we provide a detailed analysis of compression quality during iterations with the Viscous Fingers dataset ($\xi = 0.006$ and $\varepsilon = 0.06$) in Table 3. The number of false cases generally decreases with iterations, whereas PSNR remains unchanged. This is because we only

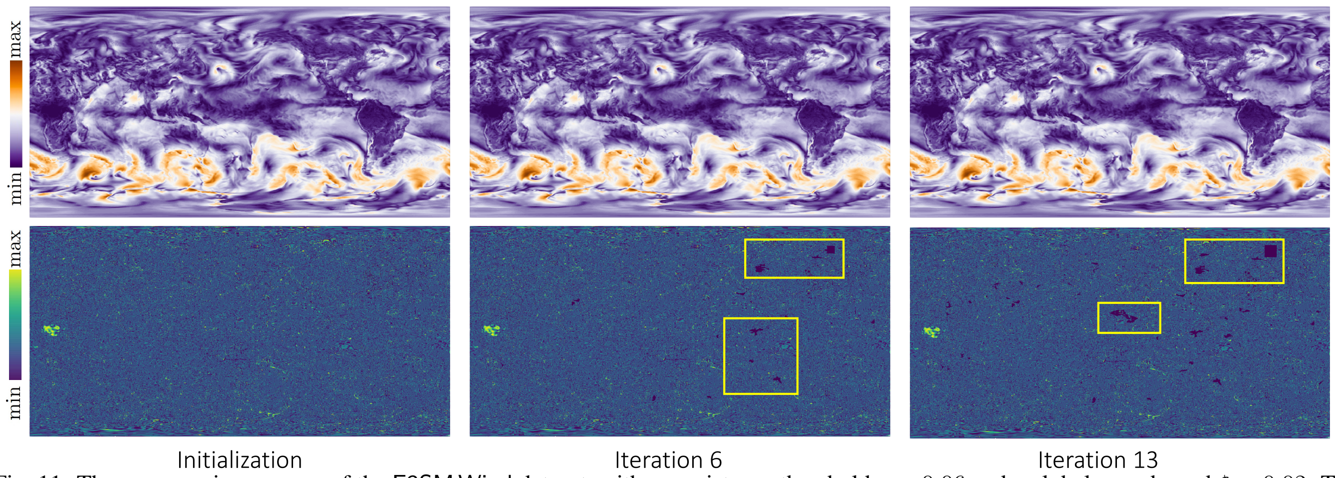


Fig. 11: The compression process of the E3SM Wind dataset, with a persistence threshold $\varepsilon = 0.06$ and a global error bound $\xi = 0.02$. Top: decompressed scalar fields during the initialization, the 6th iteration, and the 13th iteration, respectively. Bottom: pointwise error during the corresponding iterations.

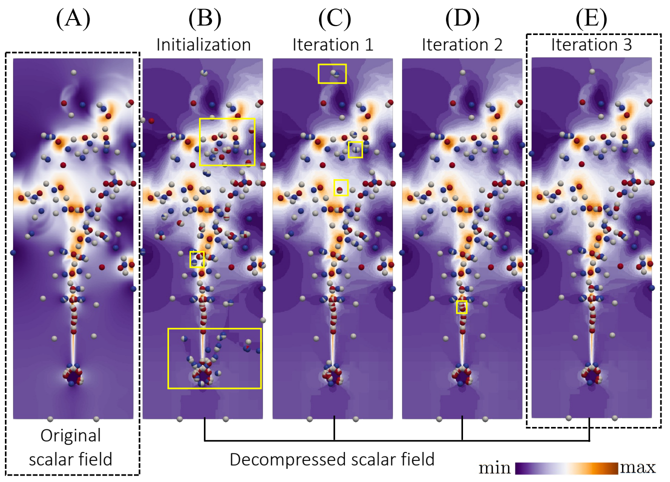


Fig. 12: The compression process of the Heated Flow dataset, with a persistence threshold $\varepsilon = 0.02$ and a global error bound $\xi = 0.05$. (A)-(E): the original scalar field, the decompressed scalar fields in the initialization step, 1st iteration, 2nd iteration, and 3rd iteration, respectively. Local maxima are in red, local minima are in blue, and saddles are in white. Yellow boxes contain false cases.

need to update the lower and upper bounds within a small region during iterations ($< 0.1\%$). These iterations effectively eliminate the detected false cases, but they cannot improve PSNR since the affected areas (by refinement) are too small. The compression ratio fluctuates with iterations, but it generally decreases. We notice a large decrease at the 8th iteration using the 8-layer neighborhood. Generally, a larger affected region leads to a larger decrease in the compression ratio and the number of false cases; see the 2nd and 8th iterations in Table 3.

5.3 Compression Performance

TopoSZ utilizes two parameters—a persistence threshold ε and a global error bound ξ—to control the compression performance. We visualize four 3D datasets before and after compression with three levels of compression quality in Fig. 9, that is, at high, medium, and low compression quality. We also display the PSNR and compression ratio next to each decompressed scalar field. As shown Fig. 9, we obtain lower compression rates when preserving finer scale topological structures, whereas we obtain higher compression rates with low PSNR measurements. We further discuss how ε and ξ affect the performance of TopoSZ.

Table 3: The analysis of compression quality during iterations for the Viscous Fingers dataset. Init. and Iter. represent the initialization and iteration steps. #FC, EB (%), and RO represent the number of false cases after each step, updated upper and lower bound w.r.t. the data domain (%), and the compression ratio, respectively.

	# FC	PSNR	EB (%)	RO		# FC	PSNR	EB (%)	RO
Init.	15	53.9	100	18.0	**5th**	3	53.9	0.021	17.9
1st	10	53.9	0.032	18.1	**6th**	1	53.9	0.052	17.8
2nd	10	53.9	0.039	18.2	**7th**	2	53.9	0.035	17.9
3nd	5	53.9	0.068	17.9	**8th**	0	53.9	0.070	17.5
4th	4	53.9	0.005	18.1					

5.3.1 Performance Analysis w.r.t. Global Error Bound

With a fixed persistence threshold $\varepsilon = 0.12$, we apply TopoSZ to all datasets in Table 2 with a global error bound $\xi \in \{0.004, 0.008, 0.012, 0.016, 0.02\}$. The 1st row of Fig. 13 shows the evolution of compression ratios, PSNRs, bottleneck distances, and Wasserstein distances as ξ increases. All datasets exhibit similar trends in the compression rates, PSNRs, and bottleneck distances as ξ increases. These trends imply that a higher ξ value poses a more relaxed constraint and leads to a higher compression rate and lower compression quality. The Wasserstein distances also exhibit an increasing trend across all datasets; in particular, for the E3SM Wind, Viscous Fingers and Tangaroa datasets. Comparing across datasets: the Heated Flow dataset contains only a few topological features due to its small size and coarse resolution; the Tornado and Isabel datasets contain one main topological feature; the E3SM Wind, Viscous Fingers and Tangaroa datasets contain many more features that are evaluated in the Wasserstein metric.

5.3.2 Performance Analysis w.r.t. Persistence Threshold

With a fixed global error bound $\xi = 0.012$, we apply TopoSZ to all datasets with a persistence threshold $\varepsilon \in [0.02, 0.04, \ldots, 0.2]$ (with a step size of 0.02). The evolution of the compression performance for an increasing ε is shown in Fig. 13 (the 2nd row). We observe that, regardless of the ε value, the compression performance has only a few changes for the Heated Flow, Tornado, and E3SM Wind datasets. However, Viscous Fingers and Tangaroa datasets contain a large number of critical points and are more difficult to handle with an error-controlled quantization (see Sec. 4.2). Therefore, a relaxed topological constraint (indicated by a higher ε) discards fine-error controls on regions where critical points have low persistence, and achieves a higher compression ratio. The Isabel dataset contains a main feature (describing the eye and its surroundings) with high persistence and numerous features with low persistence. Therefore, the compression ratio increases with ε varying

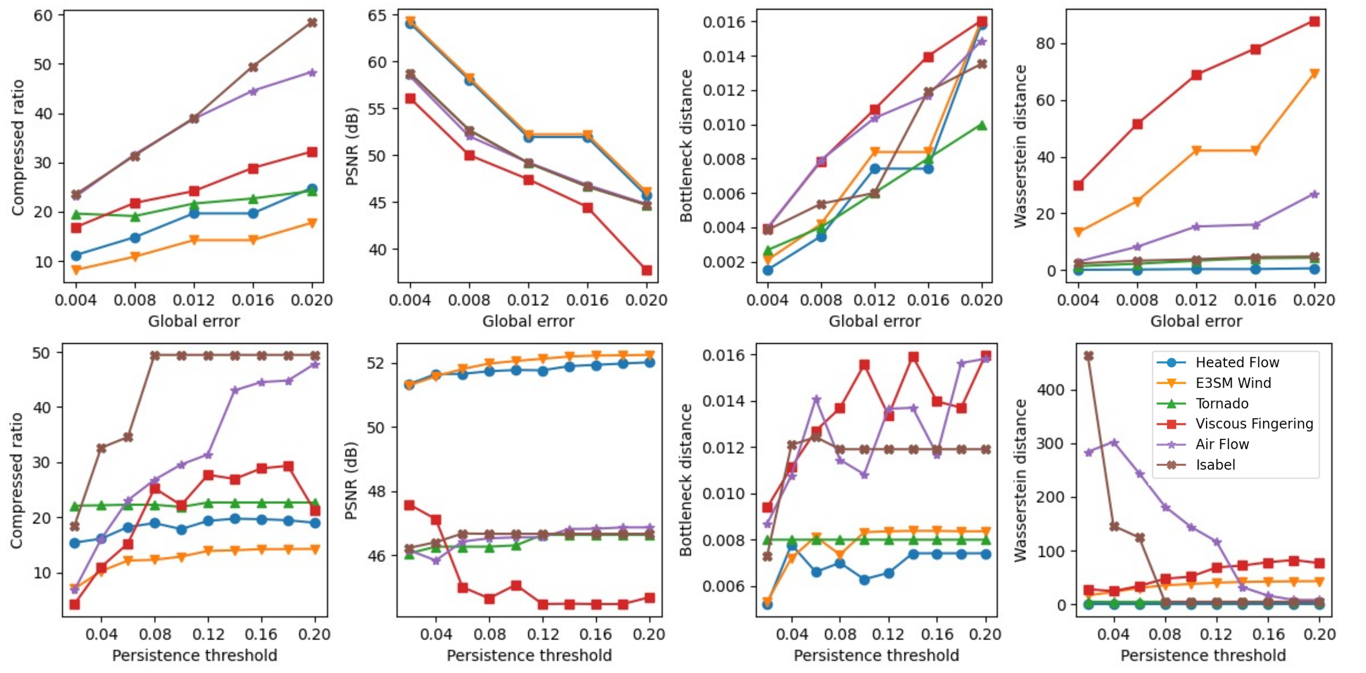


Fig. 13: Performance analysis of TopoSZ w.r.t. a global error bound ξ and a persistence threshold ε. Top row: compression performance with an increasing ξ; $\varepsilon = 0.12$. Bottom row: compression performance with an increasing ε; $\xi = 0.012$. Columns 1-4: the compression ratio, PSNR, bottleneck distance, and Wasserstein distance, respectively.

from 0.02 to 0.08 and remains the same after $\varepsilon = 0.08$. In general, the choice of ε value has less impact on the compression performance than the choice of ξ.

The two series of experiments in Fig. 13 suggest that we can use a global error bound ξ to control the compression quality, and the persistence threshold ε to add additional topological constraints to the decompressed data.

5.4 Comparison with TopoQZ

We now compare TopoSZ with TopoQZ [55]. TopoQZ is most relevant to our approach as it is designed to explicitly enforce a topological control. Its implementation is available via TTK [61].

We rerun the two series of experiments from Fig. 13 with TopoQZ. We use the same parameter setting as TopoSZ, that is, using a global error bound $\xi \in [0.004, 0.02]$ (with a step size of 0.004) and a persistence threshold $\varepsilon \in [0.02, 0.2]$ (with a step size of 0.02). Using TopoQZ, we also obtain decompressed data without any false cases. We compare the performance of TopoSZ with TopoQZ in Fig. 14.

5.4.1 Comparing Pointwise Error Control

Soler et al. claimed that TopoQZ can be extended to enforce pointwise error control [55]. However, their maximum pointwise error between the original and decompressed data is dependent on the persistence threshold. Appendix F demonstrates the evolution of the pointwise error control between TopoQZ and TopoSZ. It shows that, to the best of our knowledge, TopoSZ is the first lossy compressor that combines pointwise error control and topological guarantee during compression.

5.4.2 Comparing Compression Performance

We provide the evolution of the compression ratios, PSNRs, bottleneck distances, and Wasserstein distances, again averaged over all datasets, for a varying ξ or ε using TopoSZ (blue curves) and TopoQZ (orange curves) in Fig. 14. These average curves are meaningful since all our datasets are normalized within $[0, 1]$ during preprocessing.

The 1st row of Fig. 14 shows the compression performance with an increasing ξ and a fixed ε. The row indicates that TopoSZ has a better compression performance than TopoQZ in this experiment in terms of the compression ratio, PSNR, and topological preservation. Taking a closer look at PSNR, TopoQZ has a lower rate than TopoSZ, and the bottleneck and Wasserstein distances do not change with a varying ξ using TopoQZ. Since TopoQZ does not have a strict control on the pointwise error, an increasing ξ has less impact on the compression performance of TopoQZ.

On the other hand, as shown in the 2nd row of Fig. 14, with a fixed ξ, the compression performance of TopoQZ is highly dependent on the ε values. However, TopoSZ still exhibits better compression performance than TopoQZ under this setting for two reasons. First, data predictions from the SZ lead to the decompressed values being closer to the original values. Such a strategy also enforces a strict limitation on the decompressed data in the amount of deviation during compression. TopoQZ conducts a linear interpolation of the quantized simplified function, with a worse pointwise error when the dataset is nonlinear or irregular. Second, TopoSZ yields a higher compression capability; see Fig. 10.

However, TopoSZ appears to have a higher Wasserstein distance when ε is small (Fig. 14 bottom right) because TopoQZ simply removes all contour tree branches associated with the original scalar field when their persistences are smaller than ε. Therefore, the Wasserstein distance for TopoQZ increases when ε increases. TopoSZ, on the other hand, predicts values based on their neighborhoods. With a small ε, the predictor produces numerous contour tree branches with small persistence. Although they do not affect the PSNR or the bottleneck distance, they do affect the Wasserstein distance cumulatively.

In conclusion, TopoSZ, which is derived from the SZ-1.4, achieves a higher compression ratio than TopoQZ in these experiments.

5.5 Run Time Analysis

Table 4 provides a detailed run time analysis for TopoSZ, which contains two key components, initialization (see Sec. 4.1.1) and iteration (see Sec. 4.1.2). Average run time across iterations is reported for the iteration component. During initialization, we need to compute a simplified contour tree T_ε with its induced segmentation, update the pointwise upper and lower bounds $\mathcal{U}$ and $\mathcal{L}$, and apply a customized SZ. We also need to check for false cases within the simplified contour tree T'_ε from the decompressed data. If there are false cases, we proceed with iterations, where we update the error bounds for a small subset of points. We need to check for false cases in T'_ε at the end of each iteration. As shown in Table 4, the average run time for each iteration

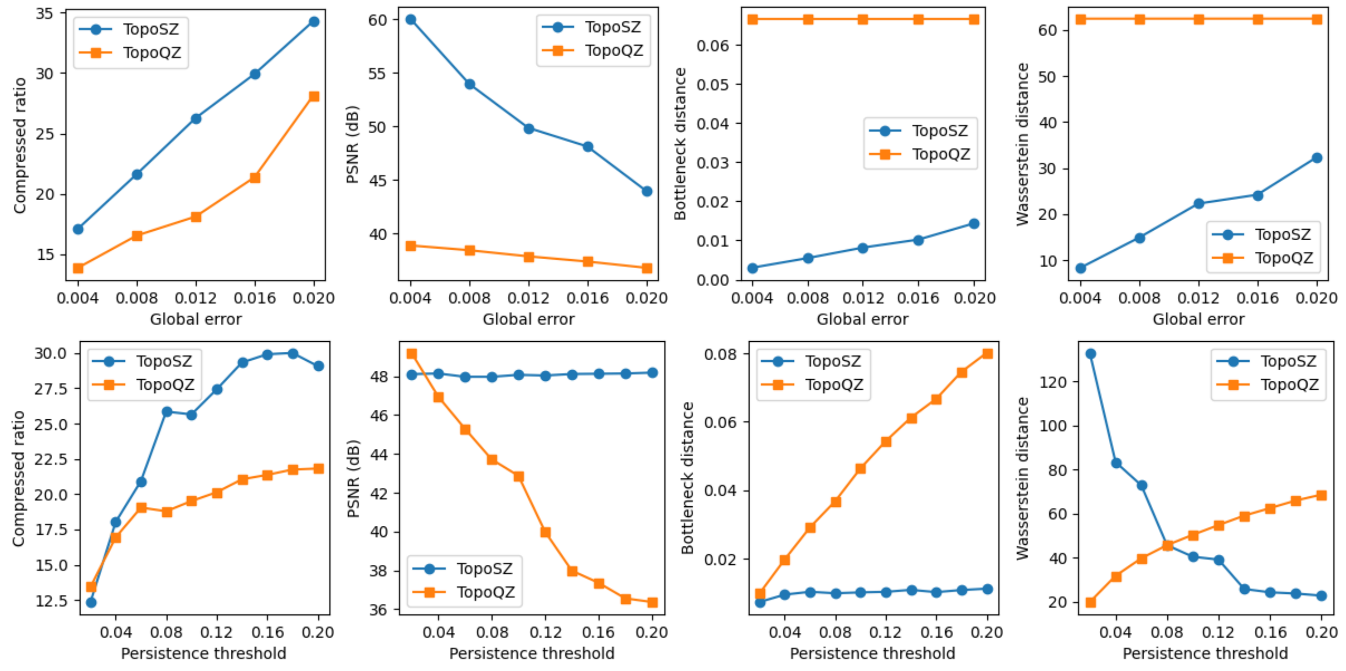


Fig. 14: Comparing TopoSZ (blue) with TopoQZ (orange). Top row: compression performance with an increasing ξ; $\varepsilon = 0.12$. Bottom row: compression performance with an increasing ε; $\xi = 0.016$. Columns 1-4: the average compression ratio, PSNR, bottleneck distance, and Wasserstein distance, respectively.

Table 4: A detailed run time analysis of TopoSZ with a varying persistence threshold ε. Notations in this table follow those of Table 1. All times are in seconds.

Dataset	ε	Initialization			Iteration			
		CT	UBLB	CSZ-1.4	CT	UBLB	CSZ-1.4	# Iterations
Heated Flow	0.01	3.85	0.07	0.03	2.03	0.03	0.03	1
	0.06	3.95	0.06	0.03	-	-	-	0
	0.12	3.92	0.06	0.03	-	-	-	0
E3SM Wind	0.01	13.60	1.07	0.16	7.59	1.83	0.14	5
	0.06	12.86	0.96	0.15	7.06	0.64	0.14	4
	0.12	11.99	0.94	0.16	6.25	0.46	0.14	1
Tornado	0.01	49.49	2.00	0.31	14.20	2.16	0.22	4
	0.06	51.13	1.78	0.26	-	-	-	0
	0.12	24.21	1.85	0.26	-	-	-	0
Viscous Fingers	0.01	20.99	1.93	0.20	11.72	6.56	0.20	15
	0.06	21.01	1.85	0.24	11.67	1.50	0.21	8
	0.12	21.43	1.84	0.23	11.45	0.87	0.20	5
Tangaroa	0.01	90.50	5.91	0.82	55.34	3.12	0.72	9
	0.06	83.79	5.62	0.72	49.53	4.11	0.62	3
	0.12	79.09	5.55	0.71	45.00	6.37	0.64	2
Isabel	0.01	472.10	19.66	2.67	228.33	117.82	2.24	16
	0.06	281.26	18.92	2.71	140.15	8.78	2.72	1
	0.12	263.29	19.09	2.59	-	-	-	0

is much less than the initialization since only a small set of points is affected to eliminate false cases. The run time bottleneck is computing the contour trees.

In addition, we investigate the run time of TopoSZ with a varying persistence threshold ε. The global error bound ξ has an impact on the iteration time, but it does not affect the run time for computing simplified contour trees. Let $\xi = 0.06$ for all experiments in Table 4. We vary the persistence threshold $\varepsilon = 0.01, 0.06$, and 0.12 for each dataset. Table 4 shows that the number of iteration decreases as we increase the persistence threshold. TTK also needs less time to compute the sampled contour trees with a large ε.

For comparison, we also provide run time analysis of SZ3, ZFP, FPZIP, TTHRESH, and TopoQZ, as shown in Table 1. All experiments are run with a global error bound $\xi = 0.06$. For TopoSZ and TopoQZ, we use a persistence threshold $\varepsilon = 0.06$. Notice that for these two topology preserving compressors, the majority (at least 90%) of the compression time is spent on computing the contour trees. We argue that in certain use cases (e.g., mitigating the requirement on storage capacity), achieving a higher compression ratio is more important than compression speed. Also, scientific data are usually compressed once when they are generated and written to the storage systems, but may be decompressed multiple times when they are retrieved for various post hoc data analytics. TopoSZ delivers decent compression ratios and fast decompression speed while preserving important topological information, which helps mitigate the extreme-scale data challenges in many uses cases. Finally, our contour tree computation uses the implementation from TTK and has not been optimized for performance, which is left for future work.

6 Conclusion

We introduce TopoSZ, a novel lossy compression framework that preserves topological features while enforces pointwise error control. In particular, TopoSZ imposes upper and lower error bounds during compression using topological information from contour trees. It achieves a pointwise error control based on a customized quantization derived from the SZ compressor (version 1.4). Our experimental results demonstrate the strengths of TopoSZ in preserving the topological features while achieving good compression quality, in comparison to state-of-the-art lossy compressors with comparable compression rates. In general, any prediction-based error-bounded compressor may be customized using a similar strategy for topology-preserving compression, and it is feasible to extend TopoSZ using a multilayer predictor [58].

For future work, we would like to further improve TopoSZ for in situ deployments and large-scale time-varying data. We also want to mitigate the effects of sequential topological simplifications. It may be possible to improve by reducing the number of iterations to eliminate false cases using machine learning techniques. We are also interested in preserving topological features using gradient-based topological descriptors, such as Morse and Morse–Smale complexes.

Acknowledgments

Argonne National Laboratory is a US Department of Energy laboratory managed by UChicago Argonne, LLC, under contract DE-AC02-06CH11357. This research was supported by DOE DE-SC0023157, DOE DE-SC0022753, DOE DE-SC0021015, NSF IIS 1910733, NSF IIS 2145499, NSF OAC-2311878, and NSF OAC-2330367.

References

[1] Computer Graphics Laboratory. https://cgl.ethz.ch/research/visualization/data.php. 12

[2] Energy Exascale Earth System Model. https://esgf-node.llnl.gov/projects/e3sm/. 12

[3] GZIP. https://www.gzip.org. 2

[4] IEEE Scientific Visualization Contest 2004. http://vis.computer.org/vis2004contest/index.html. 12

[5] IEEE Scientific Visualization Contest 2016. https://www.uni-kl.de/sciviscontest/. 12

[6] ZSTD. http://www.zstd.net. 2

[7] Y. Abu-Mostafa and R. J. McEliece. Maximal codeword lengths in Huffman codes. *Computers & Mathematics with Applications*, 39(11):129–134, 2000. doi: 10.1016/S0898-1221(00)00119-X 2

[8] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel. Nyx: A massively parallel amr code for computational cosmology. *The Astrophysical Journal*, 765(1):39, 2013. doi: 10.1088/0004-637X/765/1/39 12

[9] T. Athawale, D. Maljovec, L. Yan, C. Johnson, V. Pascucci, and B. Wang. Uncertainty visualization of 2D Morse complex ensembles using statistical summary maps. *IEEE Transactions on Visualization and Computer Graphics*, 28(4):1955–1966, 2020. doi: 10.1109/TVCG.2020.3022359 3

[10] K. Beketayev, D. Yeliussizov, D. Morozov, G. Weber, and B. Hamann. Measuring the distance between merge trees. *Topological Methods in Data Analysis and Visualization III: Theory, Algorithms, and Applications, Mathematics and Visualization*, pp. 151–166, 2014. doi: 10.1007/978-3-319-04099-8_10 1

[11] C. Biwer. Nyx cosmological simulation dataset, 2019. doi: 10.21227/zh6w-kt72 12

[12] M. Burtscher and P. Ratanaworabhan. High throughput compression of double-precision floating-point data. In *Data Compression Conference*, pp. 293–302, 2007. doi: 10.1109/DCC.2007.44 2

[13] M. Burtscher and P. Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Transactions on Computers*, 58(1):18–31, 2008. doi: 10.1109/TC.2008.131 2

[14] P. M. Caldwell, A. Mametjanov, Q. Tang, L. P. Van Roekel, J.-C. Golaz, W. Lin, D. C. Bader, N. D. Keen, Y. Feng, R. Jacob, et al. The DOE E3SM coupled model version 1: Description and results at high resolution. *Journal of Advances in Modeling Earth Systems*, 11(12):4095–4146, 2019. doi: 10.1029/2019MS001870 12

[15] 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. doi: 10.1177/1094342019853336 1

[16] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry: Theory and Applications*, 24(2):75–94, 2003. doi: 10.1016/S0925-7721(02)00093-7 1

[17] H. Carr, J. Snoeyink, and M. Van De Panne. Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree. *Computational Geometry: Theory and Applications*, 43(1):42–58, 2010. doi: 10.1016/j.comgeo.2006.05.009 3

[18] Y.-J. Chiang and X. Lu. Progressive simplification of tetrahedral meshes preserving all isosurface topologies. In *Computer Graphics Forum*, vol. 22, pp. 493–504. Wiley Online Library, 2003. doi: 10.1111/1467-8659.00697 2

[19] M. M. Chow. Optimized geometry compression for real-time rendering. *Proceedings of Visualization (Cat. No. 97CB36155)*, pp. 347–354, 1997. doi: 10.1109/VISUAL.1997.663902 1

[20] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37:103–120, 2007. doi: 10.1007/s00454-006-1276-5 5, 12

[21] R. Crawfis. Tornado data set generator. http://web.cse.ohio-state.edu/~crawfis.3/Data/Tornado/, 2003. 12

[22] S. Di, D. Tao, X. Liang, and F. Cappello. Efficient lossy compression for scientific data based on pointwise relative error bound. *IEEE Transactions on Parallel and Distributed Systems*, 30(2):331–345, 2018. doi: 10.1109/TPDS.2018.2859932 2

[23] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah. An exploration framework to identify and track movement of cloud systems. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2896–2905, 2013. doi: 10.1109/TVCG.2013.131 1

[24] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010. 3, 5, 12

[25] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. Morse-Smale complexes for piecewise linear 3-manifolds. *Proceedings of the 19th Annual Symposium on Computational Geometry*, pp. 361–370, 2003. doi: 10.1145/777792.777846 1

[26] H. Edelsbrunner, J. Harer, and A. J. Zomorodian. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete & Computational Geometry*, 30(87-107), 2003. doi: 10.1007/s00454-003-2926-5 1

[27] W. Engelke, T. B. Masood, J. Beran, R. Caballero, and I. Hotz. Topology-based feature design and tracking for multi-center cyclones. In *Topological Methods in Data Analysis and Visualization VI. Mathematics and Visualization*. Springer, Cham, 2021. doi: 10.1109/TVCG.2014.2346434 1

[28] S. Gerber and K. Potter. Data analysis with the Morse-Smale complex: The msr package for R. *Journal of Statistical Software*, 50(2), 2012. doi: 10.18637/jss.v050.i02 1

[29] J.-C. Golaz, P. M. Caldwell, L. P. Van Roekel, M. R. Petersen, Q. Tang, J. D. Wolfe, G. Abeshu, V. Anantharaj, X. S. Asay-Davis, D. C. Bader, et al. The DOE E3SM coupled model version 1: Overview and evaluation at standard resolution. *Journal of Advances in Modeling Earth Systems*, 11(7):2089–2129, 2019. doi: 10.1029/2019MS001870 12

[30] Z. Gong, T. Rogers, J. Jenkins, H. Kolla, S. Ethier, J. Chen, R. Ross, S. Klasky, and N. F. Samatova. MLOC: Multi-level layout optimization framework for compressed scientific data exploration with heterogeneous access patterns. In *Proceedings of the 41st International Conference on Parallel Processing*, pp. 239–248, 2012. doi: 10.1109/ICPP.2012.39 2

[31] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Contour forests: Fast multi-threaded augmented contour trees. In *IEEE 6th Symposium on Large Data Analysis and Visualization*, pp. 85–92, 2016. doi: 10.1109/LDAV.2016.7874333 3

[32] C. Gueunet, P. Fortin, J. Jomier, and J. Tierny. Task-based augmented contour trees with fibonacci heaps. *IEEE Transactions on Parallel and Distributed Systems*, 30(8):1889–1905, 2019. doi: 10.1109/TPDS.2019.2898436 5

[33] T. Günther, M. Gross, and H. Theisel. Generic objective vortices for flow visualization. *ACM Transactions on Graphics*, 36(4):141:1–141:11, 2017. doi: 10.1145/3072959.3073684 12

[34] P. G. Howard and J. S. Vitter. Analysis of arithmetic coding for data compression. *Information processing & management*, 28(6):749–763, 1992. doi: 10.1016/0306-4573(92)90066-9 2

[35] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. *Computer Graphics Forum*, 22(3):343–348, 2003. doi: 10.1111/1467-8659.00681 2

[36] U. Jayasankar, V. Thirumal, and D. Ponnurangam. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University - Computer and Information Sciences*, 33(2):119–140, 2021. doi: 10.1016/j.jksuci.2018.05.006 1

[37] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In *European Conference on Parallel Processing*, pp. 366–379, 2011. doi: 10.1145/3307681.3326608 2

[38] X. Liang, S. Di, S. Li, D. Tao, B. Nicolae, Z. Chen, and F. Cappello. Significantly improving lossy compression quality based on an optimized hybrid prediction model. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–26, 2019. doi: 10.1145/3295500.3356193 2

[39] 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 *IEEE International Conference on Big Data*, pp. 438–447, 2018. doi: 10.1109/BigData.2018.8622520 1, 2

[40] X. Liang, H. Guo, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, and T. Peterka. Toward feature-preserving 2D and 3D vector field compression. In *IEEE Pacific Visualization Symposium*, pp. 81–90, 2020. doi: 10.1109/PacificVis48177.2020.6431 1

[41] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao, Z. Chen, and F. Cappello. SZ3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data*, 2022. doi: 10.1109/TBDATA.2022.3201176 2

[42] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014. doi: 10.1109/TVCG.2014.2346458 1, 2

[43] P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006. doi: 10.1109/TVCG.2006.143 1, 2

[44] A.-P. Lohfink, F. Wetzels, J. Lukasczyk, G. H. Weber, and C. Garth. Fuzzy contour trees: Alignment and joint layout of multiple contour trees. *Computer Graphics Forum*, 39(3):343–355, 2020. doi: 10.1111/cgf.13985 2

[45] E. Nilsson, W. Engelke, A. Friederici, and I. Hotz. Tracking and visualizing multi-center cyclones. In *Leipzig Symposium on Visualization in Applications*, 2020. doi: 10.31219/osf.io/jqtua 1

[46] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *Proceedings of the IASTED Conference on Visualization, Imaging, and Image Processing*, pp. 452–290, 2004. 3

[47] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004. 12

[48] 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. doi: 10.1175/1520-0426(2004)021<1575:EANSOT>2.0.CO;2 12

[49] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique (on the singular points of a complete integral pfaff form or of a numerical function). *Comptes Rendus Acad. Science Paris*, 222:847–849, 1946. 1

[50] A. Said and W. A. Pearlman. An image multiresolution representation for lossless and lossy compression. *IEEE Transactions on image processing*, 5(9):1303–1310, 1996. doi: 10.1109/83.535842 1

[51] H. Saikia, H. P. Seidel, and T. Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. *Computer Graphics Forum*, 33(3):41–50, 2014. doi: 10.1111/cgf.12360 1

[52] H. Saikia, H.-P. Seidel, and T. Weinkauf. Fast similarity search in scalar fields using merging histograms. *Topological Methods in Data Analysis and Visualization IV (Proceedings of Topology-Based Methods in Visualization)*, pp. 121–134, 2015. doi: 10.1007/978-3-319-44684-4_7 1

[53] J. Schneider and R. Westermann. Compression domain volume rendering. In *IEEE Visualization*, pp. 293–300, 2003. doi: 10.1109/VISUAL.2003.1250385 1

[54] B.-S. Sohn and C. Bajaj. Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):14–25, 2006. doi: 10.1109/TVCG.2006.16 2

[55] M. Soler, M. Plainchault, B. Conche, and J. Tierny. Topologically controlled lossy compression. In *IEEE Pacific Visualization Symposium*, pp. 46–55, 2018. doi: 10.1109/PacificVis.2018.00015 2, 8

[56] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 26(3):1518–1531, 2020. doi: 10.1109/TVCG.2018.2873612 1

[57] S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49, 2004. doi: 10.1016/j.gmod.2003.08.002 3

[58] 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 *IEEE International Parallel and Distributed Processing Symposium*, pp. 1129–1139, 2017. doi: 10.1109/IPDPS.2017.115 1, 2, 3, 4, 5, 9

[59] D. M. Thomas and V. Natarajan. Symmetry in scalar field topology. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2035–2044, 2011. doi: 10.1109/TVCG.2011.236 1, 2

[60] D. M. Thomas and V. Natarajan. Multiscale symmetry detection in scalar fields by clustering contours. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2427–2436, 2014. doi: 10.1109/TVCG.2014.2346332 1

[61] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology Toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2017. doi: 10.1109/TVCG.2017.2743938 5, 8

[62] 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. doi: 0.1109/TVCG.2012.228 2, 5

[63] K. Wu and S. Zhang. A contour tree based visualization for exploring data with uncertainty. *International Journal for Uncertainty Quantification*, 3(3), 2013. doi: 10.1615/INT.J.UNCERTAINTYQUANTIFICATION.2012003956 2

[64] L. Yan, T. B. Masood, F. Rasheed, I. Hotz, and B. Wang. Geometry aware merge tree comparisons for time-varying data with interleaving distances. *IEEE Transactions on Visualization and Computer Graphics*, 2022. doi: 10.1109/TVCG.2022.3163349 1

[65] 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. doi: 10.1111/cgf.14331 1

[66] X. Zhang, C. L. Bajaj, and N. Baker. Fast matching of volumetric functions using multi-resolution dual contour trees. Technical report, Texas Institute for Computational and Applied Mathematics, Austin, Texas, 2004. 2

[67] 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 *Proceeding of the 37th IEEE International Conference on Data Engineering (ICDE)*, 2021. doi: 10.1109/ICDE51399.2021.00145 2

[68] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, Z. Chen, and F. Cappello. Significantly improving lossy compression for HPC datasets with second-order prediction and parameter optimization. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 89–100, 2020. doi: 10.1145/3369583.3392688 2