

FeatureZ: A General Framework for Feature-Preserving Compression via Pointwise Bounds and Star Classification

Nathaniel Gorski , Xin Liang , Hanqi Guo , and Bei Wang 

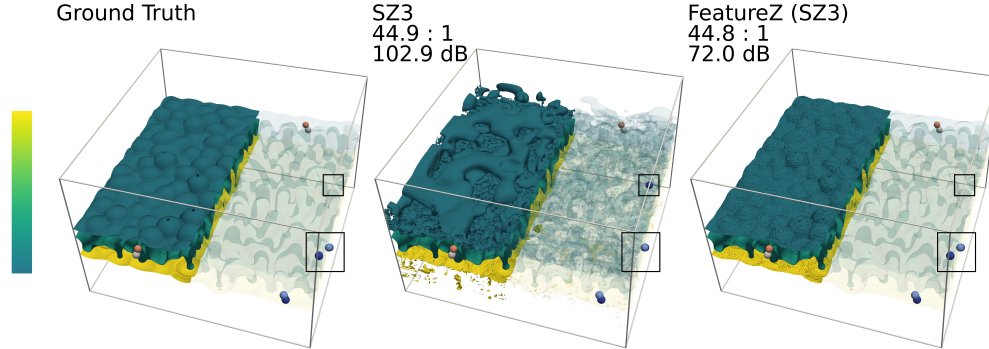


Figure 1: The Miranda dataset visualized with three isocontours and the vertices of the simplified contour tree with a persistence threshold of 4% of the range. Left: ground truth. Center: reconstruction using SZ3. Right: reconstruction using *FeatureZ*, denoted as *FeatureZ*(SZ3), which augments SZ3 to preserve isocontours and the simplified contour tree simultaneously. Both SZ3 and *FeatureZ* operate at comparable compression ratios. SZ3 introduces substantial errors in both the isocontours and the contour tree, whereas *FeatureZ* preserves these structures. This example highlights *FeatureZ*’s ability to simultaneously preserve multiple features.

Abstract—Geometric and topological features, such as isosurfaces, quantiles, merge trees, and Morse–Smale complexes, are central to the analysis and visualization of scientific data across diverse domains, including medical imaging, climate science, materials science, and astronomy. However, most lossy compressors for scientific data provide only pointwise error guarantees and do not preserve derived features. Existing feature-preserving compressors are often difficult to develop and typically tailored to a single feature. In this paper, we introduce *FeatureZ*, a lossy compression framework for structured volumetric scalar fields that can preserve a wide class of geometric and topological features. In particular, *FeatureZ* frames feature-preserving compression as preserving pointwise upper and lower bounds together with a consistent classification of each point’s star (i.e., its incident cells) in the underlying structured mesh. Although not all notions of feature preservation fit this framework, many features and topological descriptors targeted by existing compressors do. *FeatureZ* provides an efficient implementation of feature-preserving compression under this formulation. In particular, *FeatureZ* operates as an augmentation layer that refines the output of an existing lossy compressor. It first applies quantization to enforce pointwise bounds, and then employs an iterative procedure to ensure consistent star classification. We demonstrate that *FeatureZ* preserves diverse features during compression with minimal overhead, achieving compression ratios comparable to or better than methods specialized for individual features.

Index Terms—Lossy compression, error-bounded compression, generalized feature preservation, topological and geometric features, topological data analysis, scientific visualization, extensible frameworks

1 INTRODUCTION

Geometric and topological features are derived quantities that describe the shape, structure, and connectivity of scientific data, and they play a central role in its analysis and visualization across diverse domains. For example, in medical imaging, isocontours are widely used to reconstruct anatomical structures such as soft tissue and bone surfaces [35]. In astronomy, critical points of the matter density field and their connectivity define the skeleton of the cosmic web [3], while contour trees have been employed to denoise ALMA data cubes [39]. In quantum chemistry, Morse–Smale complexes enable detailed analysis of atomic basins as well as the extraction of molecular graphs [8]. In climate science, critical points identify cyclone centers [49], whereas isocontours and quantiles support the analysis of temperature, pressure, and humidity fields, facilitating the study of extreme events such as atmospheric

rivers [22] and atmospheric blocking [24].

Despite their importance, most lossy compressors for scientific data are designed to control pointwise reconstruction error and do not explicitly account for these derived quantities. Consequently, even when global error bounds are small, compression can introduce significant distortions in geometric and topological structures [18], compromising the fidelity of downstream analysis and visualization. To address this limitation, recent work has proposed feature-preserving compression techniques that explicitly target specific structures, such as contour trees [18, 50] and Morse–Smale segmentations [27, 28].

However, existing feature-preserving compressors require significant time to develop and are typically specialized to a single feature, with designs that do not generalize across different types of structures. For example, a previous compressor [18] (referred to as TopoA in this paper) computes the contour tree of the decompressed data and applies corrections when deviations from the expected topology are detected. While effective, this approach is intrinsically tied to contour trees and does not readily extend to other feature classes, reflecting a broader limitation of current methods. As a result, feature-preserving compression remains inflexible and fragmented. There is no systematic pathway for scientists to preserve new, application-specific features without designing bespoke solutions from scratch. Furthermore, each feature typically requires a dedicated compressor (e.g., TopoA [18] for contour trees or pMSz [28] for Morse–Smale segmentation), forcing users to

- Nathaniel Gorski is with the University of Utah. E-mail: gorski@sci.utah.edu.
- Xin Liang is with Oregon State University. E-mail: lianxin@oregonstate.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. 202x; accepted xx xxx. 202x. Date of Publication xx xxx. 202x; date of current version xx xxx. 202x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxx/TVCG.202x.xxxxxx

identify, install, and manage an expanding collection of specialized tools. This fragmented ecosystem underscores the need for a flexible, unified framework that can support user-defined feature preservation within a single, generalizable compression paradigm.

In this work, we introduce *FeatureZ*, a general and extensible compression framework designed to preserve a broad class of features while satisfying user-specified error bounds. *FeatureZ* is *general* in that it provides a novel formulation of feature preservation that unifies a number of geometric and topological features within a single framework. It is also *extensible*, allowing user-defined features to be incorporated by adding a few new functions without redesigning the compression pipeline.

While *FeatureZ* does not preserve arbitrary derived features, it supports a broad and practically relevant class. Specifically, it supports any feature whose preservation can be expressed through pointwise upper and lower bounds, together with a consistent classification of each point’s star (i.e., the collection of incident cells) in the underlying structured mesh. We show that many commonly used features, including isocontours, quantiles, critical points, contour trees [10], merge trees [37], and Morse–Smale segmentations [38], fit naturally within this formulation and can be preserved by *FeatureZ*. Notably, all features targeted by existing topology-preserving compressors for scalar-field data can be expressed within this framework.

We further develop an efficient algorithm that preserves features under this novel formulation. Similar to prior topology-preserving approaches, *FeatureZ* operates as an augmentation layer on top of an existing base compressor such as ZFP [31] or SZ3 [30]. It first applies the base compressor and analyzes the decompressed data, then enforces feature preservation through quantization-based bound correction and an iterative scan procedure that ensures consistent star classification. Because this process relies only on local information—the function values at a point and within its star—*FeatureZ* avoids repeated global feature computation, enabling efficient and scalable compression. Our empirical results demonstrate that *FeatureZ* achieves compression ratios and runtimes comparable to specialized methods. We introduce *FeatureZ*, a general framework for feature-preserving compression based on star classification:

- We introduce a unified formulation of feature-preserving compression based on pointwise upper and lower bounds together with local star consistency. We show that this formulation captures many geometric and topological features targeted by existing feature-preserving compressors, as well as several previously unsupported feature classes.
- We develop an efficient framework that augments existing lossy compressors with feature-preservation guarantees. We prove that the resulting compressed data preserves the specified features under the proposed formulation.
- We conduct a comprehensive evaluation demonstrating that our method preserves diverse combinations of features across multiple base compressors while achieving compression ratios and runtimes that are competitive with, and often superior to, existing feature-preserving approaches.

2 RELATED WORK

Lossy compression for scientific data. Modern scientific instruments and simulations generate massive data volumes, often on the order of terabytes [9], posing significant challenges for storage and transmission. To address these issues, scientists rely on data compression, which can be broadly categorized as *lossless* or *lossy*. Lossless compression guarantees exact reconstruction of the original data but typically achieves limited reduction (often under $2\times$ [41]). In contrast, lossy compression permits controlled distortion in exchange for substantially higher compression ratios and is therefore preferred in practice.

Scientific workflows, however, demand careful control and quantification of compression-induced errors. As a result, modern lossy compressors for scientific data provide strict guarantees on error bounds [25, 30, 31]. These compressors generally fall into two categories: *prediction-based* and *transformation-based*. Prediction-based methods estimate data values via interpolation and encode only the residuals required to satisfy prescribed error bounds. Early work such

as ISABELA [21] employs B-spline interpolation, while more recent approaches like SZ3 [30] combine multiple predictors, including Lorenzo [19], cubic splines, and linear interpolation.

Recently, deep learning techniques have been increasingly incorporated into prediction-based compression strategies. Implicit Neural Representations (INRs), such as [13, 36], train compact neural networks that take spatial coordinates as input and approximate the underlying data, with the learned network parameters serving as the compressed representation. Although INRs are computationally expensive to train and evaluate, they have gained popularity due to their ability to provide a continuous representation of the data. In contrast, autoencoder-based approaches [32, 34, 46] learn paired *encoder* and *decoder* networks, where the encoder maps data to a low-dimensional latent space and the decoder reconstructs the data from this representation.

Complementary to prediction-based methods, transformation-based lossy compressors achieve compression by decorrelating data through mathematical transforms. For instance, ZFP [31] partitions data into small blocks and compresses each block independently using exponent alignment, a near-orthogonal transform, and embedded encoding. Other methods exploit tensor decomposition techniques: TTHRESH [7] treats the dataset as a tensor and applies singular value decomposition (SVD) for decorrelation, while SPERR [26] combines wavelet transforms with SPECK coding.

Derived features in scientific visualization. Derived geometric and topological features play a central role in scientific data analysis and visualization. Among these, isocontours are fundamental primitives that provide intuitive representations of level sets of scalar fields. They are widely used in applications ranging from elevation mapping in cartography to exploring slices of deep neural network loss landscapes [11]. Global statistical summaries also play an important role; for instance, quantiles are commonly used for uncertainty visualization in direct volume rendering [4] and in medical image analysis [51].

Topological features further provide powerful tools for understanding complex data. Critical points—locations where the gradient of a function vanishes—have been used to track clouds [23] and tropical cyclones [49]. Contour trees support the analysis of scalar field topology and have been applied to transfer function design [52] as well as to domains such as fluid dynamics [5] and medicine [6, 42, 45]. Similarly, Morse–Smale complexes have been used to study molecular structures [8] and to derive meaningful visualizations from histopathology images [2].

Feature-preserving lossy compression. A variety of feature-preserving compressors have been developed for scalar field data, each targeting different structural properties. Building on topological abstractions, Soler et al. [40] preserved the 0-dimensional persistence diagram. TopoSZ [50] preserves the contour tree up to a user-specified persistence threshold ε and was subsequently extended by TopoA [18]. MSz [27] preserves the piecewise-linear Morse–Smale segmentation of a scalar field, with further improvements introduced in pMSz [28]. Similarly, TopoSZp [1] preserves extrema (minima and maxima) while retaining a large subset of saddle points. Beyond strictly topological features, Jiao et al. [20] proposed a method that preserves quantities of interest (QoIs)—numerical functions evaluated pointwise—as well as contours. These strategies were improved upon by QPET [33]. Finally, LOPC [15] adopts a local perspective by preserving the *local order* of a scalar field, i.e., maintaining “less-than” relationships between neighboring points. Related feature-preserving approaches have also been developed for vector fields [29, 47, 48] and tensor fields [17].

3 TECHNICAL BACKGROUND

3.1 Piecewise-Linear Scalar Fields

A *piecewise-linear* (PL) scalar field on a simplicial complex $\mathbb{X}$, $f : \mathbb{X} \rightarrow \mathbb{R}$, is defined by assigning scalar values to the vertices of $\mathbb{X}$ and extending f over each simplex via linear interpolation. It is often convenient to assume that f is injective on the vertices of $\mathbb{X}$. To enforce injectivity, ties between vertices $x, y \in \mathbb{X}$ with identical values are resolved using *simulation of simplicity* (SOS) [14]. Several SOS schemes exist; we adopt the implementation of simplicial complexes and SOS provided by the Topology Toolkit [44]. Specifically, if the

structured grid coordinates of x precede those of y in column-major (Fortran) order, we define $f(x) < f(y)$; otherwise, $f(x) > f(y)$.

For a vertex $x \in \mathbb{X}$, the *star* $\text{St}(x)$ is the set of simplices containing x , along with their faces. The *link* $\text{Lk}(x)$ consists of simplices in $\text{St}(x)$ that do not contain x . Given a PL scalar field f , the *upper link* $\text{Lk}^+(x)$ is the subset of simplices $\sigma \in \text{Lk}(x)$ such that $f(y) > f(x)$ for all vertices y of σ . The *lower link* $\text{Lk}^-(x)$ is defined analogously.

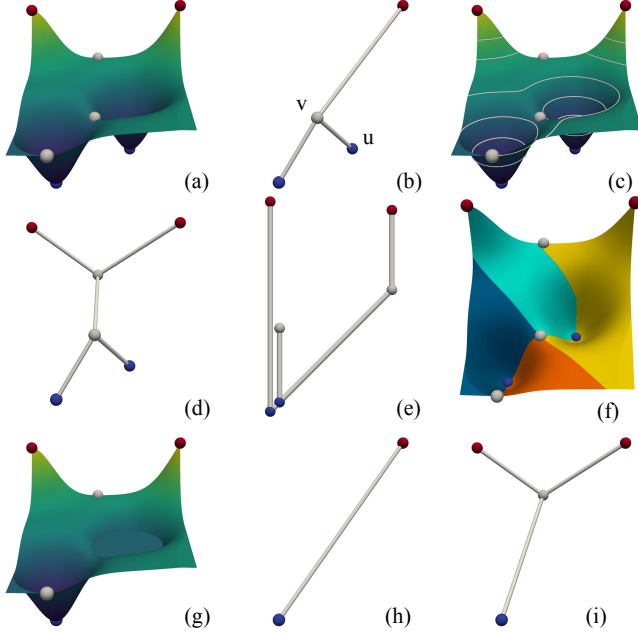


Figure 2: Various topological descriptors for PL scalar fields. (a) Scalar field with critical points (minima blue, saddles white, maxima red). (b) Merge tree with feature uv . (c) Scalar field with selected contours. (d) Contour tree. (e) 0-dimensional persistence diagram: segments denote features; endpoints are colored by criticality types; $y = x$ is the diagonal. (f) Morse–Smale segmentation. (g) Persistence simplification removing uv . (h) Simplified merge tree. (i) Simplified contour tree.

3.2 Topology of Piecewise-Linear Scalar Fields

We use *FeatureZ* to preserve several topological features; accordingly, we review common topological descriptors for PL scalar fields. Most of these descriptors originate from Morse theory, where one considers smooth functions. By a slight abuse of notation, we write $f : \mathbb{M} \rightarrow \mathbb{R}$ for a smooth function on a manifold and $f : \mathbb{X} \rightarrow \mathbb{R}$ for its PL counterpart, where $\mathbb{X}$ is a triangulation of $\mathbb{M}$.

Critical points. For a smooth function $f : \mathbb{M} \rightarrow \mathbb{R}$, critical points satisfy $\nabla f(x) = 0$. An analogous notion exists in the PL setting. Let $f : \mathbb{X} \rightarrow \mathbb{R}$ be a PL scalar field. A vertex $x \in \mathbb{X}$ is *ordinary* if both $\text{Lk}^+(x)$ and $\text{Lk}^-(x)$ are nonempty and simply connected; otherwise, it is *critical*. In 3D, a critical point is *degenerate* if $\text{Lk}^+(x)$ and $\text{Lk}^-(x)$ have more than three total connected components.

Nondegenerate critical points are classified by their *index*, defined as the number of independent directions in $\text{Lk}^-(x)$. In 3D, minima have index 0, maxima index 3, and saddles index 1 or 2. We visualize a scalar field with its critical points in Fig. 2(a).

Merge trees. The *sublevel set* at value t is $f^{-1}(-\infty, t] = \{x \in \mathbb{X} : f(x) \leq t\}$. As t increases, connected components of sublevel sets are created at minima and merge at saddles, eventually forming a single component at the global maximum. The *merge tree* encodes this evolution, with vertices at critical points and edges tracking the merge of components; see Fig. 2(b).

Persistence diagram. The evolution of sublevel sets induces a pairing between minima (where components are created) and saddles (where they merge), yielding the 0-dimensional persistence diagram. A component that appears at time a and merges with another component at time b gives rise to a point (a, b) in the diagram with persistence

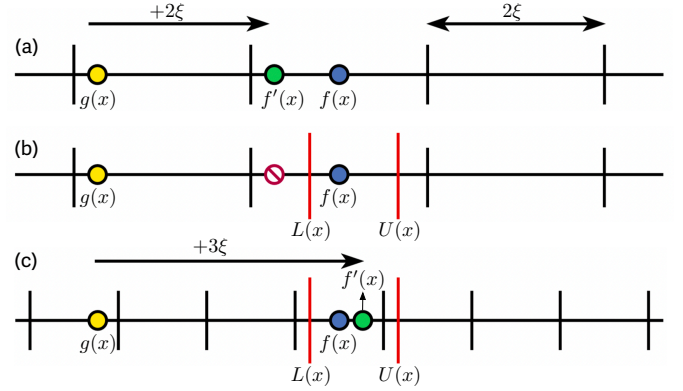


Figure 3: (a) Illustration of standard linear-scaling quantization. Each quantization interval has width 2ξ . The value $f(x)$ denotes the ground truth, and $g(x)$ is an initial approximation. By shifting $g(x)$ to the right by one quantization level ($n_x \leftarrow +1$), we obtain a final value $f'(x)$ that lies within a distance ξ of $f(x)$. (b) Example where prescribed upper and lower bounds, $U(x)$ and $L(x)$, cannot be satisfied using standard linear-scaling quantization. (c) Using logarithmic-scaling quantization, the interval widths are effectively halved. This allows $g(x)$ to be shifted upward by 3ξ , yielding a final value $f'(x)$ that satisfies the bounds.

$b - a$, measuring its significance. The diagram also contains analogous components for the evolution of super level sets $f^{-1}[t, \infty)$. A 0-dimensional persistence diagram is visualized in Fig. 2(e).

Contours and contour trees. A *contour* (level set) of $f : \mathbb{X} \rightarrow \mathbb{R}$ at value $t \in \mathbb{R}$ is a connected component of $f^{-1}(t)$. Examples are shown in Fig. 2(c). As t increases, contours may appear, merge, split, or disappear. The *contour tree* records these events (see Fig. 2(d)).

Components appear and merge at minima and saddles, as captured by the merge tree of f , and may split or disappear at saddles and maxima, as captured by the merge tree of $-f$. Accordingly, the contour tree is obtained by combining the merge trees of f and $-f$ [10].

Persistence simplification removes features whose persistence is below a threshold ε . This corresponds to eliminating branches in the merge trees of f and $-f$ associated with features below the threshold. A simplified contour tree is then obtained by combining the simplified merge trees of f and $-f$. An example of a simplified scalar field is shown in Fig. 2(g), with the corresponding simplified merge and contour trees in Fig. 2(h) and (i).

PL Morse–Smale segmentation. The *integral lines* of a PL function $f : \mathbb{X} \rightarrow \mathbb{R}$ are discrete monotone paths that follow the local gradient through the vertices. For a vertex $x \in \mathbb{X}$ that is not a maximum, the *forward integral line* from x proceeds to its highest neighbor and ultimately terminates at a maximum. Backward integral lines are defined analogously and terminate at minima.

The *PL Morse–Smale segmentation* partitions the domain such that two points lie in the same region if their forward and backward integral lines terminate at the same (minimum, maximum) pair. Each region therefore consists of points whose flow originates at a common minimum and terminates at a common maximum; see Fig. 2(f).

3.3 Quantization in Lossy Compression

A quantization strategy maps continuous values to a discrete set of representatives to enable compression. Suppose that for some $x \in \mathbb{X}$, a predictive model produces an estimate $g(x)$ of the ground truth value $f(x)$ that may not satisfy a global error bound ξ . Quantization-based strategies can then be used to adjust $g(x)$ to a value $f'(x)$ such that $|f'(x) - f(x)| < \xi$.

Linear-scaling quantization was introduced in SZ3 1.4 [43]. It can be viewed as partitioning the real line into intervals of width 2ξ centered at $f(x)$. By construction, any value within the same interval as $f(x)$ deviates from it by at most ξ . Linear-scaling quantization determines an integer *quantization index* n_x indicating how many such intervals $g(x)$ must be shifted to fall into the interval containing $f(x)$, thereby satisfying the error bound. This process is illustrated in Sec. 3.3(a). The reconstructed value is then given by $f'(x) \leftarrow g(x) + n_x(2\xi)$.

This procedure produces a set of integers n_x for all $x \in \mathbb{X}$. These indices are subsequently compressed using an entropy coding scheme (e.g., Huffman coding) and stored alongside the compressed data. When the predictor $g(x)$ is accurate, most n_x values are zero, yielding high compressibility and low storage overhead.

Logarithmic-scaling quantization. Consider the additional constraint that each reconstructed value must satisfy pointwise bounds $L(x) < f'(x) < U(x)$ for given functions $L(x), U(x) \in \mathbb{R}$. This setting arises, for example, in topology-preserving compression [18, 50].

To address this constraint efficiently, Gorski et al. [18] introduced *logarithmic-scaling quantization*, a multiscale extension of linear-scaling quantization. For each $x \in \mathbb{X}$, the method assigns two integers: a *precision* p_x and a *quantization index* n_x . The procedure begins with $p_x \leftarrow 0$ and applies linear-scaling quantization using intervals of width $2\xi/2^{p_x}$ to obtain n_x and a reconstructed value $f'(x)$.

If $L(x) < f'(x) < U(x)$, the process terminates; otherwise, the quantization precision is increased and linear-scaling quantization is repeated until the constraint is satisfied. If no admissible reconstruction is found with p_x at or below a maximum precision P , the value $f(x)$ is stored losslessly. For a fixed p_x , multiple n_x may yield admissible reconstructions; in this case, the value with minimal magnitude is selected to reduce the total entropy of the set of $\{n_x\}$.

For encoding, a single integer is formed as $a_x \leftarrow n_x \cdot 2^{P-p_x}$, and the sequence $\{a_x\}$ is compressed using an entropy coder. During decompression, the reconstruction is computed as

$$f'(x) \leftarrow g(x) + \frac{a_x(2\xi)}{2^P}.$$

This construction provably reconstructs the correct value of $f'(x)$. Consequently, only a_x needs to be stored as the quantization index.

4 PRESERVATION OF FEATURES

FeatureZ enforces a global error bound ξ while simultaneously preserving user-defined features whose invariance is characterized by pointwise upper and lower bounds together with a consistent star classification. We formalize the class of features supported by *FeatureZ* in Sec. 4.1, and subsequently detail how specific feature types can be preserved in Sec. 4.2.

4.1 Overview of Feature Preservation

FeatureZ is designed to preserve a broad class of geometric and topological features that share common sufficient conditions for preservation.

Let $f : \mathbb{X} \rightarrow \mathbb{R}$ denote the original scalar field and f' the reconstructed field obtained after compression and decompression with *FeatureZ*. We model a derived feature as the application of a transformation Φ to the data, i.e., $\Phi(f)$. For example, if Φ denotes contour tree extraction, then $\Phi(f)$ is the contour tree of f . Under this formulation, our objective is to ensure feature preservation, i.e., $\Phi(f) = \Phi(f')$.

To preserve the feature $\Phi(f)$, we define a star classification function $C_{\Phi,f} : \mathbb{X} \rightarrow \mathbb{Z}$ that assigns to each point in the domain a categorical label according to the role of its star in the construction of $\Phi(f)$. We also prescribe pointwise upper and lower bound functions $U_{\Phi,f}, L_{\Phi,f} : \mathbb{X} \rightarrow \mathbb{R}$.

During compression, for each $x \in \mathbb{X}$, *FeatureZ* enforces the following conditions:

- (a) $L_{\Phi,f}(x) < f'(x) < U_{\Phi,f}(x)$ or $f'(x) = f(x)$
- (b) $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$.

Condition (a) constrains each reconstructed value to lie within its prescribed bounds (or is stored losslessly if, e.g., $L_{\Phi,f}(x) = U_{\Phi,f}(x)$), while condition (b) ensures consistency of the star classification between the original and reconstructed fields.

The class of features that *FeatureZ* can preserve consists precisely of those whose invariance can be guaranteed under the above conditions. In practice, this class is broad: in particular, all features targeted by existing topology-preserving compressors can be handled within this framework (see Sec. 4.2). Moreover, multiple features can be preserved simultaneously by ensuring that these conditions are met for each feature.

4.2 Preservation of Diverse Features

We now describe how different features Φ can be preserved using *FeatureZ*. To this end, we construct the bound functions $U_{\Phi,f}$ and $L_{\Phi,f}$, together with the star classification function $C_{\Phi,f}$, such that the guarantees enforced by *FeatureZ* imply that $\Phi(f)$ is preserved. In some cases, preservation can be achieved using only the bound functions or the star classification alone; not all features require both conditions. Correctness proofs are provided in the supplement.

Critical points. In Morse theory, each point in the domain is classified as either regular or critical (i.e., a minimum, maximum, or saddle), referred to as its *criticality type*. For vertices in a structured mesh, the criticality type depends only on the star, $\text{St}(x)$. Accordingly, we define $C_{\Phi,f}(x)$ to be the criticality type of x . It follows directly that if $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$ for all $x \in \mathbb{X}$, then the critical points of f coincide with those of f' . For efficiency, we compute the criticality type of each point using the lookup table strategy from [37], except that we use additional tables for the boundary. We provide details in the supplement.

Simplified merge trees and contour trees. Denote the contour tree of f by $\mathcal{T}(f)$ and its ε -simplified version by $\mathcal{T}_\varepsilon(f)$. Denote the merge tree of f by $\mathcal{M}(f)$ and its simplified merge tree by $\mathcal{M}_\varepsilon(f)$. Let $\eta \leq \varepsilon$ be a persistence threshold. We construct pointwise upper and lower bounds for each $x \in \mathbb{X}$ using a strategy inspired by TopoA with persistence parameter η . Specifically, if x is a vertex of $\mathcal{T}_\eta(f)$, we set $L_{\Phi,f}(x) = U_{\Phi,f}(x) = f(x)$, thereby forcing $f'(x) = f(x)$. Otherwise, x lies on an edge (a, b) of $\mathcal{M}_\eta(f)$ with endpoints a and b . Similarly, x lies on an edge (c, d) of $\mathcal{M}_\eta(-f)$. We define $U_{\Phi,f}(x) = \max(f(b), \min(f(d), f(x)))$ and $L_{\Phi,f}(x) = \min(f(c), \max(f(a), f(x)))$.

These bounds guarantee that every branch of $\mathcal{T}_\eta(f)$ is preserved in $\mathcal{T}_\eta(f')$, although $\mathcal{T}_\eta(f')$ may contain additional branches. TopoA uses a similar (but not identical) strategy with $\eta = \varepsilon$ and removes additional branches through a progressive bound-tightening procedure. In contrast, we show that every additional branch has persistence at most $\eta + 2\xi$, where ξ is the pointwise error bound (a formal proof is provided in the supplement). Thus, setting $\eta = \varepsilon - 2\xi$ ensures that any such additional branches have persistence below ε . Consequently, simplifying both $\mathcal{T}_\eta(f)$ and $\mathcal{T}_\eta(f')$ with threshold ε yields $\mathcal{T}_\varepsilon(f) = \mathcal{T}_\varepsilon(f')$, and no progressive correction is required. This strategy requires $\varepsilon \geq 2\xi$, a condition commonly satisfied in practice. A similar construction can be used to preserve simplified merge trees.

Unsimpified merge trees and contour trees. To preserve merge and contour trees without persistence simplification, we use the same upper and lower bound construction as in the simplified case, setting $\eta = 0$ so that $\mathcal{T}_\eta(f) = \mathcal{T}(f)$. These bounds ensure that every branch of $\mathcal{T}(f)$ is preserved in $\mathcal{T}(f')$, although $\mathcal{T}(f')$ may contain additional branches with persistence at most 2ξ .

The number of branches in $\mathcal{T}(f')$ equals the number of extrema of f' minus one. Accordingly, we define $C_{\Phi,f}(x)$ to classify each point as either an extremum or a non-extremum. If $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$ for all $x \in \mathbb{X}$, then the number of extrema—and hence the number of branches—coincides between f and f' . It follows that $\mathcal{T}(f') = \mathcal{T}(f)$.

Merge trees without simplification can be preserved using an analogous construction.

0-dimensional persistence diagram. For simply connected domains, preservation of the 0-dimensional persistence diagram—whether simplified or unsimplified—follows directly from preservation of the corresponding contour tree.

Piecewise-linear Morse–Smale segmentation. The Morse–Smale segmentation is defined by integral lines connecting minima to maxima. At each $x \in \mathbb{X}$, the construction of integral lines is determined by the highest vertex in $\text{Lk}^+(x)$ and the lowest vertex in $\text{Lk}^-(x)$ (assuming they are not empty). We first preserve whether or not each $x \in \mathbb{X}$ is an extremum (i.e., a minimum or maximum). We then define $C_{\Phi,f}(x)$ to encode the identities of the highest and lowest vertices in each of $\text{Lk}^+(x)$ and $\text{Lk}^-(x)$. If x is a maximum, we only encode the smallest neighbor of x , and proceed analogously if x is a minimum.

If $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$ for all $x \in \mathbb{X}$, then our construction guarantees that each integral line will travel to the correct neighbors of each

$x \in \mathbb{X}$, thus preserving the Morse–Smale segmentation. This strategy is analogous to the C-loops used in pMSz [28].

Local order. For each $x \in \mathbb{X}$, the local order of $\text{St}(x)$ is determined by the partition of $\text{Lk}(x)$ into $\text{Lk}^+(x)$ and $\text{Lk}^-(x)$. We define $C_{\Phi,f}$ accordingly so as to encode this partition, ensuring that the local order is preserved.

Isocontours, sublevel and superlevel sets. We demonstrate the preservation of isocontours (level sets) for 3D scalar fields; the 2D case follows analogously. In 3D, for a given isovalue a , the intersection of an isocontour with a cell σ can be detected using the marching cubes algorithm [35], which determines whether each vertex has a function value above or below a .

Given a set of isovalues $\{a_i\}$ with $a_1 < a_2 < \dots < a_n$, we preserve the set of cells intersected by each isocontour using only pointwise upper and lower bounds. For each $x \in \mathbb{X}$ that does not lie on an isocontour, we identify i such that $a_i < f(x) < a_{i+1}$ and set $L_{\Phi,f}(x) \leftarrow a_i$ and $U_{\Phi,f}(x) \leftarrow a_{i+1}$. If x lies on an isocontour, we set $L_{\Phi,f}(x) = f(x) = U_{\Phi,f}(x)$. This construction also preserves the sublevel and superlevel sets for each isovalue.

Quantiles. Preserving quantiles (e.g., quartiles or percentiles) amounts to preserving the *quantile membership* of each point, i.e., the quantile to which each point belongs relative to the rest of the data. We first compute function value thresholds $\{a_i\}$ that partition the data into quantiles. We then preserve the corresponding sublevel and superlevel sets induced by $\{a_i\}$, ensuring that quantile membership is maintained.

Combinations of features. Any combination of the above features can be preserved simultaneously by enforcing all corresponding upper and lower bounds and maintaining all associated star classifications.

Existing feature-preserving capabilities. Existing compressors target a limited set of features. Local order is preserved by LOPC [15]. Critical points are partially preserved by TopoSZp [1] and fully preserved by LOPC (as a consequence of local order preservation), although *FeatureZ* preserves the critical points without stronger guarantees on local ordering. Simplified contour trees (up to persistence ε) are preserved by TopoSZ [50] and TopoA [18], but no method guarantees preservation of the full (unsimplified) contour tree. The piecewise-linear Morse–Smale segmentation is preserved by MSz [27] and pMSz [28]. Isocontours are preserved by QPET [33] and the compressor of Jiao et al. [20]. To our knowledge, no existing compressor explicitly targets the preservation of quantiles, nor supports the simultaneous preservation of multiple feature types among features listed above.

5 CORE COMPRESSION ALGORITHM

FeatureZ is designed as an augmentation layer that operates on top of an existing lossy compressor, referred to as the *base compressor*, similar in spirit to topology-preserving compressors such as TopoA [18]. *FeatureZ* applies corrections to the output of the base compressor to enforce preservation of a user-defined feature Φ . These corrections are stored in a compact representation and included alongside the base compressor output in the final compressed file.

We provide an overview of the pipeline in Sec. 5.0.1. We give details of the most intricate component, iterative star correction, in Sec. 5.0.2, followed by how it is specifically applied for individual features in Sec. 5.0.3. We describe the process of extending *FeatureZ* to preserve new features in the supplement. As before, let f denote the original data and f' the reconstructed data. *FeatureZ* preserves the feature Φ while enforcing a pointwise error bound ξ , i.e., it guarantees that $\Phi(f) = \Phi(f')$ and, for all $x \in \mathbb{X}$, $|f(x) - f'(x)| \leq \xi$.

5.0.1 Pipeline

We give an overview of each step of *FeatureZ*.

Step 1: Base compressor. The data is compressed and decompressed using the base compressor. The decompressed output is called the *intermediate data*.

Step 2: Bound computation. The pointwise upper and lower bounds $U_{\Phi,f}$ and $L_{\Phi,f}$ are computed.

Step 3: Initial quantization. Logarithmic scaling quantization is used to ensure that the upper and lower bounds, as well as the pointwise error bound ξ , are maintained.

Step 4: Iterative star correction. We iterate over each vertex $x \in \mathbb{X}$ and inspect $\text{St}(x)$. We check whether $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$. If the classifications do not match, some vertices will be labeled as *bad*, meaning that they have contributed to the misclassification. If no vertices are bad, this indicates that $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$. The error bounds of a subset of the bad vertices are tightened and quantization is reapplied (see Sec. 5.0.2 for details). This process is repeated until there are no bad vertices, meaning that $\text{St}(x)$ is correctly classified.

Since modifying a vertex value may affect the classification of neighboring stars that have already been processed, multiple passes are required to ensure that all stars are correctly classified.

Step 5: Lossless compression. The quantization numbers from logarithmic-scaling quantization are compressed using Huffman coding. The quantization numbers, lossless values, and output of the base compressor are combined into a tar and compressed using XZ [12].

5.0.2 Iterative Star Correction

During star correction, we check whether $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$ for each $x \in \mathbb{X}$. Rather than directly computing the values of $C_{\Phi,f}(x)$ and $C_{\Phi,f'}(x)$, each point is directly labeled as good or bad. If all points are good, this indicates that $C_{\Phi,f}(x) = C_{\Phi,f'}(x)$. The exact strategy for labeling points as good versus bad varies by feature (see Sec. 5.0.3). If any points are bad, then we update values in $\text{St}(x)$ as follows.

Initially, each vertex $x \in \mathbb{X}$ is assigned a *precision* $p_x \leftarrow 0$. If points $y \in \text{St}(x)$ are marked as bad, those with the smallest precision p_y are selected and their precision is incremented by one. These vertices are then re-quantized using logarithmic-scaling quantization with upper bound $\min(U_{\Phi,f}(x), f(x) + \xi \cdot 2^{-p_x})$ and lower bound $\max(L_{\Phi,f}(x), f(x) - \xi \cdot 2^{-p_x})$. If there are multiple valid choices of quantization intervals, the quantization interval that yields the most accurate reconstructed value is selected. If p_x surpasses the maximum allowable precision, $f(x)$ is stored losslessly. After the values of the bad points are updated, each point in the star is re-classified as either good or bad. This process is repeated until there are no bad points. To ensure termination, we ensure that if any points $\{y_i\}$ are marked as bad, at least one y_i satisfies $f(y_i) \neq f'(y_i)$.

Updating a vertex's value may affect the classification of neighboring stars. Therefore, the star correction step is performed iteratively, and the entire dataset is scanned multiple times. To improve efficiency, we maintain a set of *dirty* vertices x for which $C_{\Phi,f}(x) \neq C_{\Phi,f'}(x)$ may hold. In each iteration, only dirty vertices are processed. Initially, all vertices are marked as dirty. Whenever a vertex x is updated, both x and its neighbors are marked as dirty for the next iteration. The dirty set is first represented as a bit array; once the number of dirty vertices falls below 20% of the domain, we switch to a sparse representation storing only the indices of dirty vertices. Switching to a sparse representation significantly speeds up star correction when the number of dirty points is small.

5.0.3 Identification of Bad Points

Let $x \in \mathbb{X}$. For each feature, we classify points in $\text{St}(x)$ as good or bad. We describe this process for all features whose preservation involves star correction.

Critical points. We ensure that the criticality type of x agrees for f and f' . If the types coincide, then all points in $\text{St}(x)$ are labeled good. Otherwise, x is labeled bad. In this case, for each $y \in \text{St}(x)$, if $y \in \text{Lk}^+(x)$ but should lie in $\text{Lk}^-(x)$ (or vice versa), then y is also labeled bad.

Unsimplified merge and contour trees. We preserve whether x is a minimum or maximum using a similar strategy to preserving critical points.

Local order. We preserve membership in $\text{Lk}^+(x)$ and $\text{Lk}^-(x)$. For each $y \in \text{Lk}(x)$, if $y \in \text{Lk}^+(x)$ but it should lie in $\text{Lk}^-(x)$ (or vice versa), then y is labeled bad. If any such y is bad, then x is also labeled bad.

Morse–Smale segmentation. We preserve: (a) whether x is a minimum or maximum, and (b) the largest vertex in $\text{Lk}^+(x)$ (if any) and the smallest vertex in $\text{Lk}^-(x)$ (if any). Condition (a) is handled similarly to the critical point case. For condition (b), if x is not a maximum

Table 1: Datasets along with their dimensions and file sizes.

Dataset	Dimensions	File Size (MB)
QMCPACK	$69 \times 69 \times 115$	4.4
Viscous Fingering	$128 \times 128 \times 128$	16.8
Tangaroa	$300 \times 180 \times 120$	25.9
Earthquake	$375 \times 188 \times 50$	28.2
Ionization	$310 \times 128 \times 128$	40.6
Isabel	$500 \times 500 \times 90$	90.0
Miranda	$384 \times 384 \times 256$	302.0
S3D	$500 \times 500 \times 500$	1000.0
SCALE-LETKF	$1200 \times 1200 \times 98$	1129.0

in f , let y and y' be the largest elements of $Lk^+(x)$ under f and f' , respectively. If $y \neq y'$, then x , y , and y' are labeled bad. An analogous rule is applied to $Lk^-(x)$.

6 EXPERIMENTAL RESULTS

We begin with an overview of our experimental setup in Sec. 6.1. We then evaluate reconstruction quality, including comparisons with both classical and topology-preserving compressors, in Sec. 6.2. Finally, we analyze runtime performance in Sec. 6.3.

6.1 Overview of Experiments

Compression tasks. We evaluate the capabilities of *FeatureZ* to preserve seven distinct features. We consider four features for which dedicated feature-preserving compressors already exist: the simplified contour tree, Morse–Smale segmentation, local order, and isocontours.

We also examine two additional features for which no dedicated feature-preserving compressor has been explicitly developed, namely critical points and percentiles. Finally, we assess the ability of *FeatureZ* to preserve multiple features simultaneously by jointly preserving the simplified contour tree and Morse–Smale segmentation. We refer to this setting as the preservation of the *combined feature*.

Comparison with existing compressors. We evaluate *FeatureZ* in conjunction with four base compressors: ZFP, SZ3, SPERR, and HPEZ, denoted as *FeatureZ* (ZFP), *FeatureZ* (SZ3), etc. We also include results for each base compressor without augmentation by *FeatureZ*. We evaluate HPEZ with maximum optimizations enabled. We also evaluate the neural compressor Neurcomp [36]; its poor performance is reported in the supplement.

Whenever possible, we compare against state-of-the-art compressors designed to preserve specific features, namely TopoA for the simplified contour tree, pMSz for Morse–Smale segmentation, QPET for isocontours, and LOPC for local order. We also compare against TopoSZp; since it supports only 2D data, the corresponding results are provided in the supplement. Overall, *FeatureZ* achieves better reconstruction quality and feature preservation, while TopoSZp is significantly faster.

Like *FeatureZ*, both TopoA, pMSz, and QPET augment a base compressor. For TopoA, we evaluate the same base compressors (ZFP, SZ3, SPERR, and HPEZ), denoted as TopoA (ZFP) etc. pMSz only supports ZFP and SZ3, while QPET only supports SPERR and HPEZ. We evaluate pMSz and QPET on the base compressors that they support.

Evaluation metrics. We measure bit-rate (lower is better) and peak signal-to-noise ratio PSNR (higher is better). Bit-rate measures the average number of bits per data point in the compressed representation. It is computed by dividing the number of bits in the compressed file by the number of vertices in $\mathbb{X}$. PSNR quantifies reconstruction quality, where higher values correspond to lower error. If I is the range of f , and MSE is the mean squared error between f and f' , then PSNR is defined as: $PSNR = 10 \log_{10} (I^2 / MSE)$. We also evaluate compression and decompression throughput. They are computed by dividing the ground truth file size by the compression time and decompression time, respectively. For *FeatureZ*, reported times exclude Python startup overhead for the dispatcher and begin when the compressor is invoked.

Datasets. We evaluate *FeatureZ* on nine datasets spanning a range of scientific domains. These datasets are listed in Tab. 1 and described in detail in the supplement.

Parameter configurations. The global error bound ξ is specified as a fraction of the function range; for example, $\xi = 0.1$ corresponds to 10% of the range. For each feature, we evaluate five values of ξ .

In topology-preserving compression, it has been observed that, for larger error bounds, decreasing the global error bound can, counterintuitively, lead to higher compression ratios [17, 18, 27, 48]. A possible explanation is that larger error bounds introduce greater distortion, necessitating more corrections to restore topological structure. Such error bounds are suboptimal, since a smaller error bound generally yields both a smaller compressed file size and higher reconstruction quality, whereas the error parameter is intended to control the trade-off between these quantities. To ensure a fair comparison, we evaluate all compressors using only parameter settings that are optimal for each method. To do so, we choose ξ sufficiently small for each feature so that all compressors exhibit increasing compression ratios as ξ increases. This choice of bounds is consistent with prior work [17, 18]. To maintain this monotonic trend, we use dataset-specific error bounds in some cases and omit outlier configurations that violate this behavior. The values of ξ and the rationale for their selection are provided in the supplement.

For isocontour preservation, we select three isovalues per dataset that partition the volume into approximately four equal regions. For the Miranda dataset, these isovalues produced poor visualizations due to their placement in a noisy, low-gradient region. Similarly, the isocontours of the Viscous Fingering dataset obstructed much of the view. We therefore adjust the isovalues for these datasets to yield more meaningful isocontours. The exact values are given in the supplement.

For simplified contour tree preservation, we use a persistence threshold of $\varepsilon = 0.04$ (4% of the function range), consistent with the TopoA experiments of Gorski et al. [18]. The supplement includes a sensitivity analysis of the evaluation metrics with respect to the persistence threshold ε (for simplified contour tree preservation) and the number of isovalues (for isosurface preservation).

Correctness. We have empirically verified that each feature is preserved in every trial. A formal proof of correctness is provided in the supplement.

System. All experiments are conducted on a laptop equipped with an Intel Core i9-12900HK CPU, an NVIDIA GeForce RTX 3080 Ti laptop GPU, and 32 GB of RAM.

Additional experiments. The supplement includes an ablation study comparing linear- and logarithmic-scaling quantization and evaluating the contribution of each pipeline component to feature preservation and compression overhead.

6.2 Reconstruction Quality

In Fig. 4, we present the trade-off between bit-rate and PSNR for three representative datasets across seven features (see the supplement for the complete results). We consider 17 compressor configurations, including (i) general-purpose, unaugmented base compressors such as ZFP, SZ3, SPERR, and HPEZ, (ii) topology-preserving compressors such as TopoA, pMSz, QPET, and LOPC, and (iii) our framework *FeatureZ* augmenting the base compressors. We include curves for the unaugmented base compressors under the same error bounds, although these methods do not explicitly preserve any features.

Across the three base compressor choices, a consistent trend emerges for *FeatureZ*: the SPERR-based variant generally achieves the best performance, while the ZFP-based variant performs the worst.

Comparison with feature-preserving compressors. When preserving identical features, *FeatureZ* consistently achieves reconstruction quality that is competitive with, and often superior to, existing feature-preserving compressors (Fig. 4). Observations are summarized below:

- **Overall performance.** *FeatureZ* attains reconstruction quality comparable to TopoA and QPET and generally outperforms both pMSz and LOPC across features.
- **Simplified contour tree.** *FeatureZ* (SZ3), *FeatureZ* (SPERR), and *FeatureZ* (HPEZ) generally achieve reconstruction quality on par with or better than TopoA (SZ3) TopoA (SPERR), and TopoA (HPEZ), respectively. *FeatureZ* (ZFP) performs slightly worse than TopoA (ZFP).
- **Morse–Smale segmentation.** Compared to pMSz, both *FeatureZ* (SZ3) and *FeatureZ* (ZFP) consistently achieve comparable or better reconstruction quality than pMSz (SZ3) and pMSz (ZFP). *FeatureZ* (SPERR) consistently outperforms both pMSz variants.

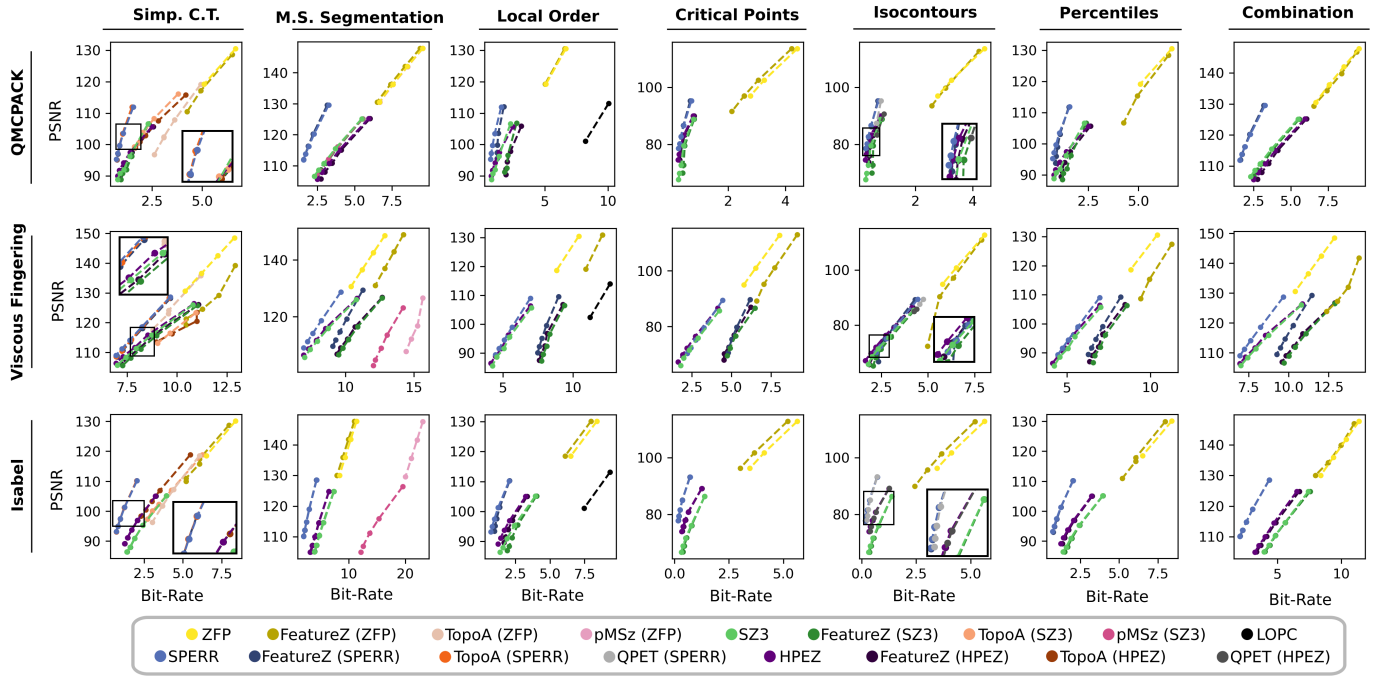


Figure 4: Rate-distortion curves with selected zoomed-in views illustrating the trade-off between bit-rate and PSNR for features preserved by *FeatureZ*. Results are shown for three datasets—QMCPACK, Viscous Fingering, Isabel—across seven features; the full set of plots is provided in the supplement. For reference, we include the base compressors ZFP, SZ3, TTHRESH, and HPEZ although these do not guarantee feature preservation. Where applicable, we also compare against state-of-the-art topology-preserving compressors, including TopoA, pMSz, QPET, and LOPC.

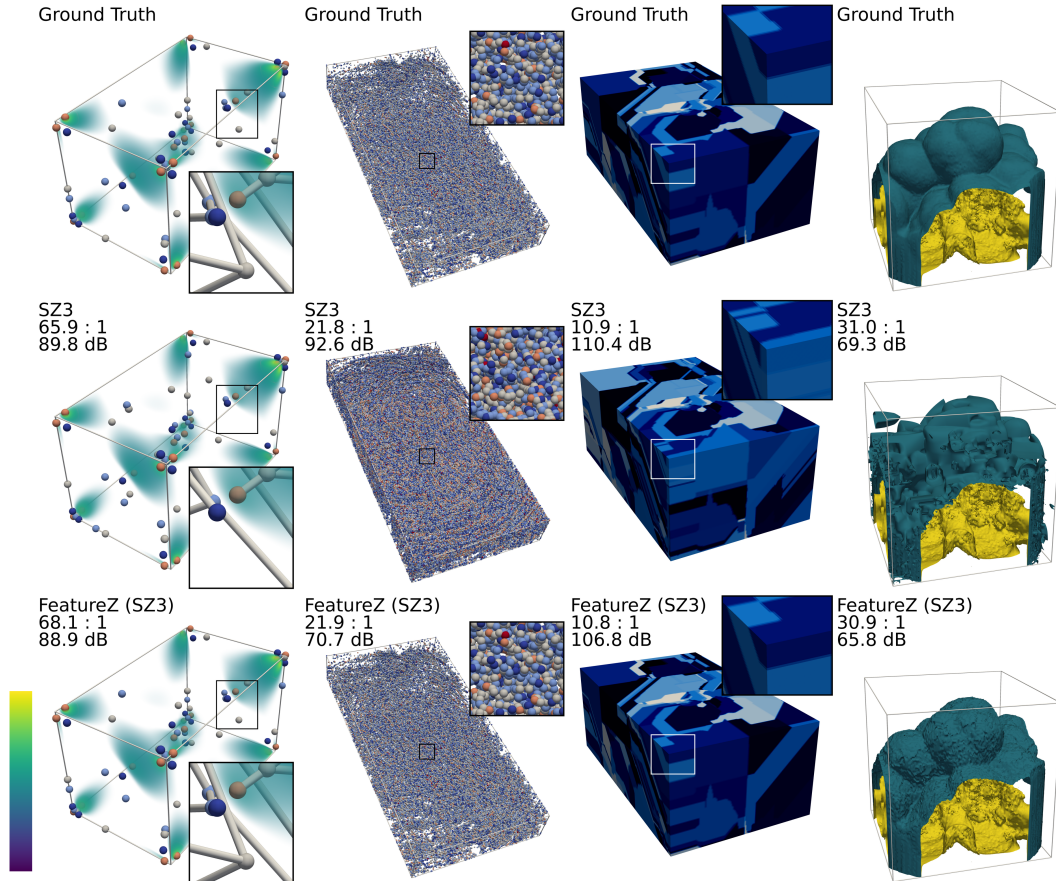


Figure 5: Volumetric datasets visualized with geometric or topological features: SZ3 compared with *FeatureZ* augmenting SZ3 at similar compression ratios. Zoomed-in views are provided to show extra detail. Top: ground truth. Center: datasets compressed with SZ3. Bottom: datasets compressed with *FeatureZ* (SZ3). From left to right, the datasets are: QMCPACK, visualized with its simplified contour tree (up to persistence $\varepsilon = 0.04$); Earthquake, with critical points colored by criticality type; Tangaroa, with Morse–Smale segmentation; and Viscous Fingering, with isocontours. In each case, SZ3 introduces errors in the extracted features, whereas *FeatureZ* (SZ3) preserves them without errors.

- **Local order.** All variants of *FeatureZ* consistently outperform LOPC, showing strong compression in local order preservation.
- **Isocontours.** *FeatureZ* (HPEZ) and *FeatureZ* (SPERR) perform similarly to QPET (HPEZ) and QPET (SPERR).

We note that pMSz and LOPC natively support different simplicial complex structures for 3D grids than *FeatureZ*, which may contribute to differences in compression overhead.

In summary, *FeatureZ* achieves reconstruction quality that is highly competitive with existing feature-preserving compression methods. Moreover, while existing approaches typically target a single feature, *FeatureZ* supports a broader range of features while delivering comparable or improved reconstruction quality.

Table 2: Error percentages for SZ3 on different datasets at compression ratios comparable to *FeatureZ* while preserving a given feature. C.T.: number of vertices that are false positives, false negatives, or misclassified in the reconstructed contour tree divided by number of vertices in the ground truth. M.S.: fraction of points incorrectly classified in the Morse–Smale segmentation. L.O.: fraction of points for which the local order among neighbors is violated. C.P.: number of critical points whose type (ordinary, minimum, saddle, maximum, degenerate) is misclassified divided by number of critical points in the ground truth. Perc.: fraction of points assigned to incorrect percentiles.

dataset	C.T.	M.S.	L.O.	C.P.	Perc.
QMCPACK	125.7%	20.4%	5.3%	108.7%	2.3%
Viscous Fingering	1.6%	51.5%	31.0%	30.5%	26.8%
Tangaroa	11.0%	34.1%	2.7%	141.9%	3.1%
Earthquake	7.1%	96.3%	21.2%	198.5%	12.0%
Ionization	12.3%	87.4%	48.0%	280.5%	38.6%
Isabel	153.3%	17.8%	5.7%	222.6%	1.0%
Miranda	83.3%	99.5%	69.2%	100.6%	61.9%
S3D	34.7%	9.8%	1.9%	83.2%	0.9%
SCALE-LETKF	47.3%	59.6%	17.4%	103.0%	20.9%

Comparison with base compressors. In Fig. 4, we observe that *FeatureZ* typically introduces only modest overhead relative to its unaugmented base compressors. In some cases, the overhead is negligible (e.g., isocontours for Viscous Fingering), while in others it can reach up to approximately $2\times$. Overall, this overhead is comparable to, or lower than, that of other topology-preserving methods [17]. Notably, *FeatureZ* (ZFP) occasionally outperforms ZFP alone, as observed for the Isabel dataset when preserving critical points. We attribute this to the combination of *FeatureZ*’s low overhead and the additional post hoc lossless compression (XZ) applied to ZFP outputs.

Although *FeatureZ* introduces additional overhead relative to base compressors, reducing the error bound of a base compressor to match the compressed file size achieved by *FeatureZ* leads to substantial feature distortion. In Tab. 2, we report feature errors for SZ3 under these conditions, which are often significant. For example, on the Earthquake dataset, even when SZ3 achieves a compression ratio comparable to *FeatureZ*, more than 96% of points are assigned to incorrect Morse–Smale segmentation regions after decompression. Notably, cases with large SZ3 errors often coincide with scenarios in which *FeatureZ* incurs higher overhead, reflecting the additional corrections required to preserve features under greater distortion. Corresponding results for ZFP and SPERR, with associated error bounds, are in the supplement, along with a detailed comparison of the trade-off between bit-rate and feature error. Visual comparisons in Fig. 1 and Fig. 5 further demonstrate that *FeatureZ* preserves a wide range of features, whereas SZ3 introduces substantial distortions even at comparable compression ratios.

6.3 Analysis of Compression Time

Runtime analysis for combined features. In Tab. 3, we report a breakdown of compression times for *FeatureZ* (SZ3) across several datasets while preserving the combined feature. For each dataset, we use the largest error bound for this setting. Results for other base compressors are similar and provided in the supplement. We focus on the combined feature because it exercises all stages of the *FeatureZ* pipeline, including both pointwise quantization and iterative star correction.

Overall, the majority of runtime is spent on bound computation, iterative star correction, and lossless compression. Notably, execution

Table 3: Runtimes for *FeatureZ* simultaneously preserving the simplified contour tree ($\epsilon = 0.04$) and Morse–Smale segmentation across datasets. Times are in seconds. Bounds: upper/lower bound computation; B.C.C.: base compressor (compression); B.C.D.: base compressor (decompression); Quant.: pointwise quantization; Star: iterative star correction; L.L.: lossless compression.

dataset	Bounds	B.C.C.	B.C.D.	Quant.	Star	L.L.	total
QMCPACK	0.46	0.03	0.01	0.00	0.06	0.06	0.61
Viscous Fingering	1.64	0.06	0.07	0.02	0.55	1.66	4.00
Tangaroa	1.86	0.12	0.09	0.01	1.02	0.67	3.77
Earthquake	1.82	0.10	0.09	0.01	1.69	1.00	4.72
Ionization	1.52	0.12	0.08	0.02	3.39	1.02	6.15
Isabel	6.16	0.46	0.36	0.19	2.21	4.48	13.87
Miranda	56.16	0.78	0.67	0.13	37.77	73.22	168.73
S3D	36.18	2.49	1.61	0.52	13.07	13.54	67.44
SCALE-LETKF	57.49	3.16	3.62	0.59	49.19	55.68	169.76

Table 4: Compression throughput (MB/s) for *FeatureZ* (SZ3) across datasets and tasks, along with TopoA (SZ3), pMSz (SZ3), and LOPC. C.T.: contour tree; M.S.: Morse–Smale segmentation; L.O.: local order; C.P.: critical points; I.S.: isocontours; Per.: percentiles; Com.: combined feature (simplified contour tree and Morse–Smale segmentation); T.A.: TopoA (SZ3). TopoA failed on SCALE-LETKF for unknown reasons.

dataset	C.T.	M.S.	L.O.	C.P.	I.S.	Per.	Com.	T.A.	pMSz	LOPC
QMCPACK	7.8	28.9	25.5	40.9	64.3	49.4	6.0	2.6	165.2	168.0
Viscous Fingering	8.1	6.6	6.4	5.7	62.1	12.3	3.9	2.0	157.6	5.6
Tangaroa	10.6	12.8	11.9	19.3	46.5	29.0	6.9	2.3	56.4	102.3
Earthquake	12.9	9.0	11.6	10.1	84.5	41.0	6.0	2.6	207.4	150.8
Ionization	21.7	5.9	7.5	7.0	53.7	43.3	6.6	2.7	119.2	57.0
Isabel	11.4	12.3	14.4	26.9	61.8	25.3	4.7	2.0	126.9	104.7
Miranda	5.1	1.9	2.8	2.6	91.7	4.0	1.8	0.9	63.3	65.1
S3D	19.4	14.7	26.9	63.0	125.7	54.4	14.8	4.3	36.3	139.0
SCALE-LETKF	14.4	6.4	11.4	12.1	117.1	33.0	6.7	–	158.2	206.5

Table 5: Decompression throughput (MB/s) for *FeatureZ* (SZ3) across datasets and tasks, along with TopoA (SZ3), and LOPC. Labels are consistent with Tab. 4.

dataset	C.T.	M.S.	L.O.	C.P.	I.S.	Per.	Com.	T.A.	LOPC
QMCPACK	147.6	143.3	141.2	174.5	168.2	157.8	131.4	84.0	470.9
Viscous Fingering	135.8	98.0	96.5	109.4	171.2	110.4	96.5	121.4	442.8
Tangaroa	108.5	79.3	76.1	126.5	133.0	100.2	81.9	108.5	353.3
Earthquake	190.4	93.9	111.3	129.3	199.2	126.3	97.5	195.1	492.8
Ionization	192.7	110.9	100.8	158.9	189.7	151.3	118.7	206.0	472.0
Isabel	117.2	74.5	102.9	141.7	144.7	113.4	79.1	115.3	367.9
Miranda	226.0	68.9	65.3	70.4	225.4	74.3	69.5	268.9	506.0
S3D	246.3	207.8	219.5	257.0	266.0	254.3	206.3	332.1	525.9
SCALE-LETKF	174.2	85.8	110.9	145.3	248.2	139.7	105.1	–	542.1

of the base compressor accounts for only a small fraction ($< 5\%$) of the total runtime, consistent other feature-preserving compressors [18]. In most cases, the dominant cost arises from computing upper and lower bounds, largely due to the expense of constructing the simplified contour tree. We consider this overhead justified, as computing a feature is a natural prerequisite for preserving it during compression.

The star correction step also contributes noticeable overhead. This is partly due to its iterative nature, although in practice the first iteration dominates runtime, with subsequent iterations affecting only a small subset of points. Additionally, per-point operations are relatively expensive: *FeatureZ* must access values from 3D neighborhoods, while the data is stored in a linear layout, leading to poor cache locality. Furthermore, star classification itself is computationally intensive.

Finally, lossless compression incurs nontrivial overhead. We use XZ due to its typically high compression ratios compared to alternatives such as gzip or bzip2 [16]. However, this choice comes with increased runtime; faster lossless compressors could reduce this overhead at the expense of lower compression efficiency.

Throughput analysis. We report compression throughput for *FeatureZ* (SZ3) in Tab. 4. For each (dataset, feature) pair, we report the runtime corresponding to the largest tested error bound. A detailed analysis of the trade-off between error bound and runtime is provided in the supplement. Throughput results for *FeatureZ* with other base compressors are similar; no single base compressor consistently yields the fastest performance. Corresponding tables for other base compressors are included in the supplement. We also report throughput for TopoA (SZ3), pMSz (SZ3), and LOPC. We do not report QPET throughput in the main paper because it does not support SZ3 as a base compressor. Corresponding results for SPERR and HPEZ are provided in the supplement. Overall, QPET achieves throughput comparable to *FeatureZ*.

Throughput varies significantly across features, reflecting differences in preservation strategies. For example, percentile and isocontour

preservation achieve higher throughput, as they do not require star correction (see Tab. 3) and have inexpensive bound computations. In contrast, simplified contour tree preservation avoids star correction but requires computing the contour tree, which is costly. Critical point and Morse–Smale segmentation preservation incur lower throughput due to the expense of iterative star correction.

Across base compressors (ZFP, SZ3, SPERR, and HPEZ), no configuration consistently minimizes runtime for *FeatureZ*. This is because the base compressor contributes only a small fraction of the total runtime, limiting the impact of its intrinsic speed. Instead, runtime is largely influenced by the distortions introduced by the base compressor, which determine the extent of corrective processing. Since these distortions affect geometric and topological structures in complex ways, their impact on runtime is difficult to predict.

Compared to other topology-preserving compressors, *FeatureZ* is roughly $5\times$ faster than TopoA but slower than pMSz and LOPC when preserving the same features. pMSz is GPU-based, and thus expected to outperform *FeatureZ*. *FeatureZ* achieves throughput comparable to other feature-preserving methods not evaluated here [17, 50].

Decompression throughput for *FeatureZ* SZ3 are reported in Tab. 5. As expected, decompression is significantly faster than compression. There is no clear correlation between compression and decompression throughput, suggesting a complex relationship between feature-preserving edits and reconstruction cost. Compared to existing methods, *FeatureZ* achieves decompression performance comparable to TopoA and slower than LOPC. Decompression times for pMSz are not available, as its experimental implementation does not report them.

Asymptotic analysis. Because *FeatureZ* supports multiple feature-preservation strategies, a single universal asymptotic bound is not meaningful. Instead, we provide a general characterization. Let n denote the number of points in the domain $\mathbb{X}$. Let the cost of bound computation over the entire domain be $\mathcal{O}(T_b(n))$, and the cost of star correction per point be $\mathcal{O}(T_s(n))$. Let k denote the maximum precision for logarithmic-scaling quantization (we use $k = 8$), so that each point may be updated at most $k + 1$ times. Let S denote the largest possible number of points in a star. Let M denote the maximum number of first- and second-order neighbors of any $x \in \mathbb{X}$.

We now analyze the major pipeline stages. Initial quantization visits each point once and may update it up to $k + 1$ times, yielding $\mathcal{O}(nk)$. During star correction, each point in the star may have values in its star updated up to k times, and its star is re-evaluated whenever a neighboring star (within first- or second-order adjacency) is modified. Consequently, a star may be updated up to $\mathcal{O}(MSk)$ times, giving a total cost of $\mathcal{O}(nSMkT_s(n))$. In practice, both the base compressor and XZ operate in $\mathcal{O}(n)$ time. Combining these components, the overall runtime of *FeatureZ* is $\mathcal{O}(T_b(n) + nSMkT_s(n))$.

We now instantiate this bound for specific features. For contour tree preservation, bound computation is dominated by contour tree construction, which is $\mathcal{O}(n \log n)$. For preserving c isocontours, bound computation requires sorting isovalues and locating each point within the corresponding interval, yielding $T_b(n) = \mathcal{O}(n \log c + c \log c)$; in practice, since $c \ll n$, this reduces to $\mathcal{O}(n)$. Percentile preservation similarly yields $T_b(n) = \mathcal{O}(n)$ in practice, while other features have $T_b(n) = \mathcal{O}(1)$. For all implemented features, $T_s(n) \in \mathcal{O}(1)$, and k , S , and M are small constants. Therefore, the overall runtime is $\mathcal{O}(n \log n)$ for contour tree preservation and $\mathcal{O}(n)$ for all other features.

7 LIMITATIONS AND DISCUSSIONS

FeatureZ has several limitations, summarized below.

Storage overhead. *FeatureZ* introduces additional storage overhead (up to $\sim 2\times$), comparable to other topology-preserving methods. For a fixed compression ratio, preserving topological correctness may outweigh marginal PSNR gains.

Runtime overhead. *FeatureZ* incurs substantial runtime overhead compared to the underlying base compressors. While its throughput is comparable to some feature-preserving methods, it remains slower overall and is outperformed by certain specialized (e.g. GPU-based) approaches. We believe there is significant potential for optimization via parallel or GPU implementations. In particular, the star correction

step is largely pointwise and could be parallelized with minimal synchronization. Bound computation depends on the user-defined feature: simplified contour tree construction is largely parallelizable [37], while percentile and isocontour computations are highly parallel.

Visual artifacts. We observe noticeable artifacts in isocontour preservation. Since points on an isocontour may shift by up to the length of a grid cell, these artifacts diminish at higher resolutions, which are typical in practice.

Design choices. *FeatureZ* relies on a number of heuristics that promote computational efficiency but may not always produce optimal augmentation configurations. In particular, feature preservation is formulated entirely through local guarantees—namely, pointwise upper and lower bounds and consistent star classifications—allowing features to be preserved using only neighborhood-level operations and avoiding expensive global geometric or topological computations. This locality, however, comes at the cost of flexibility. Global optimization strategies may identify alternative augmentations that achieve lower storage overheads while preserving the same features. Furthermore, some applications may benefit from allowing small geometric perturbations in preserved features, such as slight shifts in critical point locations, in exchange for improved compression ratios. Such trade-offs are difficult to express within our current framework. Likewise, our star-correction algorithm employs a greedy strategy that corrects vertices one at a time. While computationally efficient, this approach may introduce more corrections than necessary. Alternative strategies that jointly optimize corrections across multiple vertices could potentially achieve the same preservation guarantees with lower storage overhead.

Feature specification. Defining new features in *FeatureZ* can be non-trivial, particularly for novice users, due to the underlying definitions of star classification. We describe the technical details in the supplement. Preserving a feature requires specifying sufficient conditions within the framework supported by *FeatureZ*; this process can be unintuitive and does not encompass all features of interest.

Parameter selection. Like other feature-preserving compressors, some error settings for *FeatureZ* may be suboptimal, and decreasing the error bound can sometimes produce both smaller compressed files and higher reconstruction quality. Since there is no systematic strategy for selecting optimal parameter values, initial parameter choices may yield suboptimal performance.

Degenerate cases. In certain adversarial cases, *FeatureZ* may require near-lossless storage to preserve the requested features, for example when preserving the unsimplified contour tree of an exceptionally topologically complex dataset.

Preserving arbitrary features. In preliminary experiments, we have explored a variant of *FeatureZ* that preserves arbitrary features that can be defined simply as categorical maps $\mathbb{X} \rightarrow \mathbb{Z}$ without star classification; for instance, assigning each point a Morse–Smale region ID. However, without explicit modeling of pointwise dependencies (e.g., via star classification), this variant requires repeated re-computation of the feature, leading to substantial overhead. This underscores the importance of a pipeline that enables efficient, pointwise verification. Developing more intuitive mechanisms for feature specification remains an important direction for future work.

8 CONCLUSION

We introduce *FeatureZ*, a compression framework that preserves a broad class of features in lossy compression while maintaining a user-specified error bound ξ . For a given feature Φ , *FeatureZ* first computes pointwise upper and lower bounds enforced via quantization, and then applies an iterative procedure to ensure correct star classification with respect to Φ . Our experiments demonstrate that *FeatureZ* preserves diverse features of interest while incurring moderate overhead in file size and runtime, comparable to existing feature-preserving compressors. Looking ahead, the approach may extend to other data types, such as vector and tensor fields, and could benefit substantially from parallel and GPU implementations.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation (NSF) under Grants OAC-2313122, OAC-2313123, OAC-2313124, and OAC-2628473. We used ChatGPT 5.2 for grammar correction and language polishing, and ChatGPT 5.2 Codex for limited coding assistance, including integrating base compressors and reformatting arguments for certain internal APIs.

REFERENCES

- [1] T. Agarwal, S. Di, X. Liang, Z. Su, Y. Li, G. Gopalakrishnan et al. TopoSZp: Lightweight topology-aware error-controlled compression for scientific data. arXiv preprint arXiv:2602.17552, 2026. doi: 10.48550/arXiv.2602.17552 2, 5
- [2] Z. Ahmad, K. Al-Thelaya, M. Alzubaidi, F. Joad, N. U. Gilal, W. Mifsud et al. HistoMSC: Density and topology analysis for AI-based visual annotation of histopathology whole slide images. *Computers in Biology and Medicine*, 190:109991:1–109991:20, 2025. doi: 10.1016/j.combiomed.2025.109991 2
- [3] M. A. Aragón-Calvo, E. Platen, R. van de Weygaert, and A. S. Szalay. The spine of the cosmic web. *The Astrophysical Journal*, 723(1):364–382, 2010. doi: 10.1088/0004-637X/723/1/364 1
- [4] T. M. Athawale, B. Ma, E. Sakhaee, C. R. Johnson, and A. Entezari. Direct volume rendering with nonparametric models of uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1797–1807, 2021. doi: 10.1109/TVCG.2020.3030394 2
- [5] 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):20131042:1–20131042:10, 2014. doi: 10.1098/rsif.2013.1042 2
- [6] 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*, pp. 428–435. Springer, Berlin, 2013. doi: 10.1007/978-3-642-40763-5_53 2
- [7] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. THRESH: Tensor compression for multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics*, 26(9):2891–2903, 2020. doi: 10.1109/TVCG.2019.2904063 2
- [8] H. Bhatia, A. G. Gyulassy, V. Lordi, J. E. Pask, V. Pascucci, and P.-T. Bremer. TopoMS: Comprehensive topological exploration for molecular and condensed-matter systems. *Journal of Computational Chemistry*, 39(16):936–952, 2018. doi: 10.1002/jcc.25181 1, 2
- [9] F. Cappello, S. Di, S. Li, X. Liang, A. M. Gok, D. Tao et al. 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 2
- [10] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003. doi: 10.1016/S0925-7721(02)00093-7 2, 3
- [11] A. Chatzimichailidis, J. Keuper, F.-J. Pfrendt, and N. R. Gauger. GradVis: Visualization and second order analysis of optimization surfaces during the training of deep neural networks. In *2019 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, pp. 66–74. IEEE, New York, 2019. doi: 10.1109/MLHPC49564.2019.00012 2
- [12] L. Collin and J. Tan. XZ Utils. <https://www.tukaani.org/xz/>, 2025. 5
- [13] E. Dupont, H. Loya, M. Alizadeh, A. Golinski, Y. W. Teh, and A. Doucet. COIN++: Neural compression across modalities. *Transactions on Machine Learning Research*, 2022. 2
- [14] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990. doi: 10.1145/77635.77639 2
- [15] A. Fallin, N. Gorski, T. Agarwal, B. Wang, G. Gopalakrishnan, and M. Burtcher. Fast topology-aware lossy data compression with full preservation of critical points and local order. arXiv preprint arXiv:2603.26968, 2026. doi: 10.48550/arXiv.2603.26968 2, 5
- [16] J. Farncomb. Gzip vs Bzip2 vs XZ performance comparison. rootusers.com/gzip-vs-bzip2-vs-xz-performance-comparison/, 2015. 8
- [17] N. Gorski, X. Liang, H. Guo, and B. Wang. TFZ: Topology-preserving compression of 2D symmetric and asymmetric second-order tensor fields. *IEEE Transactions on Visualization and Computer Graphics*, 32(1):527–537, 2026. doi: 10.1109/TVCG.2025.3634844 2, 6, 8, 9
- [18] N. Gorski, X. Liang, H. Guo, L. Yan, and B. Wang. A general framework for augmenting lossy compressors with topological guarantees. *IEEE Transactions on Visualization and Computer Graphics*, 31(6):3693–3705, 2025. doi: 10.1109/TVCG.2025.3567054 1, 2, 4, 5, 6, 8
- [19] 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
- [20] P. Jiao, S. Di, H. Guo, K. Zhao, J. Tian, D. Tao et al. Toward quantity-of-interest preserving lossy compression for scientific data. *Proceedings of the VLDB Endowment*, 16(4):697–710, 2022. doi: 10.14778/3574245.3574255 2, 5
- [21] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross et al. Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In *Euro-Par 2011 Parallel Processing*, pp. 366–379. Springer, Berlin, 2011. doi: 10.1007/978-3-642-23400-2_34 2
- [22] F. Lan, B. Gamelin, L. Yan, J. Wang, B. Wang, and H. Guo. Topological characterization and uncertainty visualization of atmospheric rivers. *Computer Graphics Forum*, 43(3):e15084:1–e15084:12, 2024. doi: 10.1111/cgf.15084 1
- [23] M. Li, D. Chatterjee, F. Glassmeier, F. Senf, and B. Wang. Tracking low-level cloud systems with topology. In *2025 IEEE Workshop on Topological Data Analysis and Visualization (TopoInVis)*, pp. 89–99. IEEE, New York, 2025. doi: 10.1109/TopoInVis68599.2025.00013 2
- [24] M. Li, P. Nowack, and B. Wang. Spatiotemporal detection and uncertainty visualization of atmospheric blocking events. *IEEE Transactions on Visualization and Computer Graphics*, 32(6):4742–4752, 2026. doi: 10.1109/TVCG.2026.3694459 1
- [25] S. Li, P. Lindstrom, and J. Clyne. Lossy scientific data compression with SPERR. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1007–1017. IEEE, New York, 2023. doi: 10.1109/IPDPS54959.2023.00104 2
- [26] S. Li, P. Lindstrom, and J. Clyne. Lossy scientific data compression with SPERR. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1007–1017. IEEE, New York, 2023. doi: 10.1109/IPDPS54959.2023.00104 2
- [27] Y. Li, X. Liang, B. Wang, Y. Qiu, L. Yan, and H. Guo. MSz: An efficient parallel algorithm for correcting Morse–Smale segmentations in error-bounded lossy compressors. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):130–140, 2025. doi: 10.1109/TVCG.2024.3456337 1, 2, 5, 6
- [28] Y. Li, M. Xia, X. Liang, B. Wang, R. Underwood, S. Di et al. pMSz: A distributed parallel algorithm for correcting extrema and morse smale segmentations in lossy compression. arXiv preprint arXiv:2601.01787, 2026. doi: 10.48550/arXiv.2601.01787 1, 2, 5
- [29] X. Liang, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono et al. Toward feature-preserving vector field compression. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):5434–5450, 2023. doi: 10.1109/TVCG.2022.3214821 2
- [30] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok et al. SZ3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data*, 9(2):485–498, 2023. doi: 10.1109/TBDA.2022.3201176 2
- [31] 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 2
- [32] J. Liu, S. Di, K. Zhao, S. Jin, D. Tao, X. Liang et al. Exploring autoencoder-based error-bounded compression for scientific data. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 294–306. IEEE, New York, 2021. doi: 10.1109/Cluster48925.2021.00034 2
- [33] J. Liu, P. Jiao, K. Zhao, X. Liang, S. Di, and F. Cappello. QPET: A versatile and portable quantity-of-interest-preservation framework for error-bounded lossy compression. *Proceedings of the VLDB Endowment*, 18(8):2440–2453, 2025. doi: 10.14778/3742728.3742739 2, 5
- [34] T. Liu, J. Wang, Q. Liu, S. Alibhai, T. Lu, and X. He. High-ratio lossy compression: Exploring the autoencoder to compress scientific data. *IEEE Transactions on Big Data*, 9(1):22–36, 2023. doi: 10.1109/TBDA.2021.3066151 2
- [35] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pp. 163–169. ACM, New York, 1987. doi: 10.1145/37401.37422 1, 5
- [36] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. *Computer Graphics Forum*, 40(3):135–

- 146, 2021. doi: [10.1111/cgf.14295](https://doi.org/10.1111/cgf.14295) 2, 6
- [37] J. Lukasczyk, M. Will, F. Wetzels, G. H. Weber, and C. Garth. ExTreeM: Scalable augmented merge tree computation via extremum graphs. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1085–1094, 2024. doi: [10.1109/TVCG.2023.3326526](https://doi.org/10.1109/TVCG.2023.3326526) 2, 4, 9
- [38] R. G. C. Maack, J. Lukasczyk, J. Tierny, H. Hagen, R. Maciejewski, and C. Garth. Parallel computation of piecewise linear morse–smale segmentations. *IEEE Transactions on Visualization and Computer Graphics*, 30(4):1942–1955, 2024. doi: [10.1109/TVCG.2023.3261981](https://doi.org/10.1109/TVCG.2023.3261981) 2
- [39] P. Rosen, A. Seth, E. Mills, A. Ginsburg, J. Kamenetzky, J. Kern et al. Using contour trees in the analysis and visualization of radio astronomy data cubes. In I. Hotz, T. Bin Masood, F. Sadlo, and J. Tierny, eds., *Topological Methods in Data Analysis and Visualization VI*, pp. 87–108. Springer International Publishing, Cham, 2021. doi: [10.1007/978-3-030-83500-2_6](https://doi.org/10.1007/978-3-030-83500-2_6) 1
- [40] M. Soler, M. Plainchault, B. Conche, and J. Tierny. Topologically controlled lossy compression. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 46–55. IEEE, New York, 2018. doi: [10.1109/PacificVis.2018.00015](https://doi.org/10.1109/PacificVis.2018.00015) 2
- [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. doi: [10.14529/jsfi140205](https://doi.org/10.14529/jsfi140205) 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*, pp. 205–216. Springer, Berlin, 2010. doi: [10.1007/978-3-642-15014-2_17](https://doi.org/10.1007/978-3-642-15014-2_17) 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, New York, 2017. doi: [10.1109/IPDPS.2017.115](https://doi.org/10.1109/IPDPS.2017.115) 3
- [44] 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, 2018. doi: [10.1109/TVCG.2017.2743938](https://doi.org/10.1109/TVCG.2017.2743938) 2
- [45] 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. doi: [10.1166/jmihi.2018.2454](https://doi.org/10.1166/jmihi.2018.2454) 2
- [46] N. Wang, T. Liu, J. Wang, Q. Liu, S. Alibhai, and X. He. Locality-based transfer learning on compression autoencoder for efficient scientific data lossy compression. *Journal of Network and Computer Applications*, 205:103452:1–103452:12, 2022. doi: [10.1016/j.jnca.2022.103452](https://doi.org/10.1016/j.jnca.2022.103452) 2
- [47] M. Xia, S. Di, F. Cappello, P. Jiao, K. Zhao, J. Liu et al. Preserving topological feature with sign-of-determinant predicates in lossy compression: A case study of vector field critical points. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 4979–4992. IEEE, New York, 2024. doi: [10.1109/ICDE60146.2024.00378](https://doi.org/10.1109/ICDE60146.2024.00378) 2
- [48] M. Xia, B. Wang, Y. Li, P. Jiao, X. Liang, and H. Guo. TspSZ: An efficient parallel error-bounded lossy compressor for topological skeleton preservation. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pp. 3682–3695. IEEE, New York, 2025. doi: [10.1109/ICDE65448.2025.00275](https://doi.org/10.1109/ICDE65448.2025.00275) 2, 6
- [49] L. Yan, H. Guo, T. Peterka, B. Wang, and J. Wang. TROPHY: A topologically robust physics-informed tracking framework for tropical cyclones. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1249–1259, 2024. doi: [10.1109/TVCG.2023.3326905](https://doi.org/10.1109/TVCG.2023.3326905) 1, 2
- [50] L. Yan, X. Liang, H. Guo, and B. Wang. TopoSZ: Preserving topology in error-bounded lossy compression. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1302–1312, 2024. doi: [10.1109/TVCG.2023.3326920](https://doi.org/10.1109/TVCG.2023.3326920) 1, 2, 4, 5, 9
- [51] H. Yang, V. Baladandayuthapani, A. U. Rao, and J. S. Morris. Quantile function on scalar regression analysis for distributional data. *Journal of the American Statistical Association*, 115(529):90–106, 2020. doi: [10.1080/01621459.2019.1609969](https://doi.org/10.1080/01621459.2019.1609969) 2
- [52] 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. doi: [10.1109/TVCG.2009.120](https://doi.org/10.1109/TVCG.2009.120) 2