TFZ: Topology-Preserving Compression of 2D Symmetric and Asymmetric Second-Order Tensor Fields

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

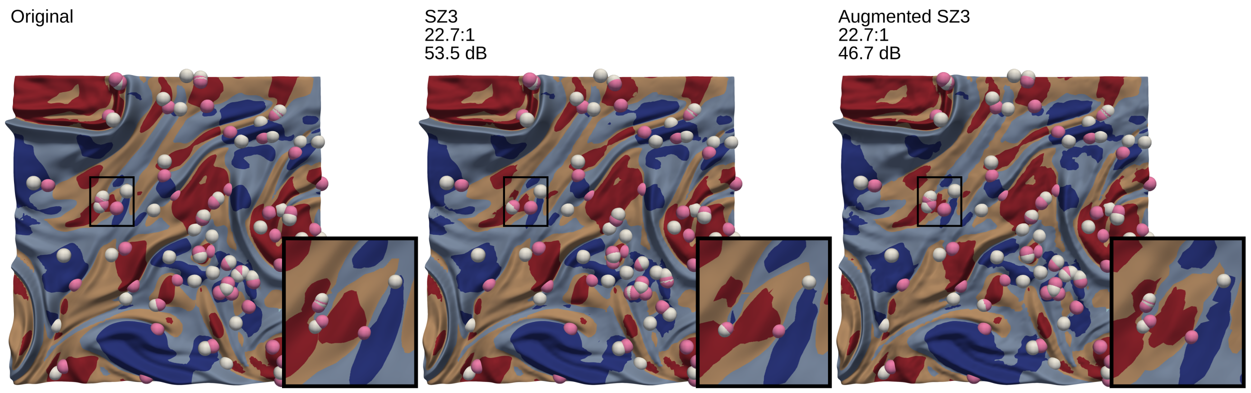

Fig. 1: Visualizing the eigenvector partition of a 2D asymmetric second-order tensor field compressed with classic SZ3 and Augmented SZ3 compressors with topological controls. Left: input data visualized with degenerate points of the dual-eigenvector field. Trisectors are in white, wedges are in pink. Middle: reconstructed data using SZ3, labeled with compression ratio and Peak Signal-To-Noise Ratio (PSNR). Right: reconstructed data using Augmented SZ3 along with compression ratio and PSNR. We provide zoomed-in views that highlight the differences between the classic and Augmented SZ3 results. The colormap is based on a partition of eigenvector manifold. The position of each point on the Z axis corresponds to the Frobenius norm with smoothing applied.

Abstract— In this paper, we present a novel compression framework, TFZ, that preserves the topology of 2D symmetric and asymmetric second-order tensor fields defined on flat triangular meshes. A tensor field assigns a tensor—a multi-dimensional array of numbers—to each point in space. Tensor fields, such as the stress and strain tensors, and the Riemann curvature tensor, are essential to both science and engineering. The topology of tensor fields captures the core structure of data, and is useful in various disciplines, such as graphics (for manipulating shapes and textures) and neuroscience (for analyzing brain structures from diffusion MRI). Lossy data compression may distort the topology of tensor fields, thus hindering downstream analysis and visualization tasks. TFZ ensures that certain topological features are preserved during lossy compression. Specifically, TFZ preserves degenerate points essential to the topology of symmetric tensor fields and retains eigenvector and eigenvalue graphs that represent the topology of asymmetric tensor fields. TFZ scans through each cell, preserving the local topology of each cell, and thereby ensuring certain global topological guarantees. We showcase the effectiveness of our framework in enhancing the lossy scientific data compressors SZ3 and SPERR.

Index Terms—Lossy compression, tensor fields, topology preservation, topological data analysis, topology in visualization

1 Introduction

The concept of a tensor is intricate and has evolved across various disciplines from differential geometry to machine learning [13]. Intuitively, a *tensor* is a mathematical entity that generalizes scalars, vectors, and matrices to higher dimensions while adhering to specific transformation rules. It can be represented as a multidimensional array of numbers. A *tensor field* is a function that assigns a tensor to each point in a space. Tensor fields naturally arise across a variety of scientific domains. In fluid dynamics, the velocity gradient tensor field describes how the fluid's velocity changes from one point to another within the flow, providing information about the deformation and rotation of fluid elements [14, 36]. In general relativity, the energy-momentum tensor field describes the matter and energy content of the universe [5], whereas the Riemann curvature tensor field describes the geometry of spacetime itself [1]. In materials science, the stress and strain tensor fields are fundamental concepts that respectively describe the internal forces within a material and how the material deforms under external loads [41, 42]. In neuroscience, diffusion tensor fields (from diffusion tensor imaging or DTI) captures the diffusion of water molecules in the white matter for the study of neural pathways [27].

The topology of tensor fields reflects the core structure of data and proves valuable across various fields, including graphics [2, 6, 48] (for manipulating shapes and textures) and neuroscience [19] (for analyzing brain structures through DTI). The topology of a tensor field depends on the dimension of the domain, the size of the tensor, and whether or not the tensors are symmetric. For instance, the stress tensor is symmetric, while the velocity gradient tensor is asymmetric. In this work, we consider 2D symmetric and asymmetric second-order tensor fields, where each field maps points in $\mathbb{R}^2$ to 2×2 matrices.

The topology of a 2D symmetric second-order tensor field is defined by its corresponding eigenvector fields [19], which are obtained by calculating the eigenvectors and eigenvalues of each tensor. The topology includes *degenerate points* (where the eigenvalues are equal) and their connections through *tensorlines* (lines aligned with the eigenvector fields). The topology determines high-level patterns in the tensorlines, which are one of the main tools used to analyze symmetric tensor fields. It is also often used to derive various types of visualizations [19, 46]. Further, in certain scenarios, the degenerate points of tensor fields

- *Nathaniel Gorski is with the University of Utah. E-mail: gorski@sci.utah.edu.*
- *Xin Liang is with the University of Kentucky. E-mail: xliang@uky.edu.*
- *Hanqi Guo is with the Ohio State University. E-mail: guo.2154@osu.edu.*
- *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

have real physical interpretations [11, 25, 51]. On the other hand, the topology of a 2D asymmetric second-order tensor field is derived from certain partitions of the domain according to decompositions of asymmetric tensors. Asymmetric tensor fields are very difficult to analyze, and their topology is one of the only tools available for analysis and visualization [3, 15, 20, 33, 38, 50].

Despite their utility, working with tensor fields is often constrained by significant storage requirements. For instance, storing a tensor field composed of 2×2 matrices necessitates 256 bits of storage per data point (assuming double precision). Consequently, the storage and transmission of tensor field data can present substantial challenges within scientific workflows. One possible solution to address this issue is to utilize lossy compressors developed specifically for scientific data, which typically ensure strict pointwise error bounds. However, applying these compressors to tensor fields often distorts high-level geometric or topological patterns in the data, even with minimal error bounds, thereby hindering downstream analysis and visualization. This is especially problematic if topological analysis is prioritized.

In this paper, we introduce TFZ, a novel framework that augments any error-bounded lossy compressor to preserve the topology of 2D symmetric and asymmetric second-order tensor fields defined on flat triangular meshes. It iterates through the cells in the mesh, enforcing specific local properties that, when preserved across all cells, ensure that global topological properties are preserved. In summary:

- TFZ could be used to enhance any error-bounded lossy compressor to provide topological guarantees. For symmetric tensor fields, it preserves the degenerate points in the eigenvector fields. For asymmetric tensor fields, it maintains the integrity of eigenvector and eigenvalue graphs. Additionally, TFZ guarantees a strict pointwise error bound for each entry of every tensor.
- As a secondary contribution, we present a new topological invariant that characterizes the topology of a mesh cell in a 2D asymmetric tensor field.
- We demonstrate the effectiveness of our framework through a comprehensive evaluation, enhancing two lossy scientific data compressors—SZ3 and SPERR—while ensuring topological guarantees.

2 Related Work

We review error-bounded lossy compression for scientific data, the topology and visualization of tensor fields, and topology-preserving compression.

Lossy data compression. Lossless compression ensures perfect data recovery but achieves limited compression ratios for scientific data, typically less than $2\times$ according to [44]. Given the massive size of scientific datasets, lossy compression is generally preferred, and error-bounded lossy compression has been developed for applications that require guarantees on data distortion. See [9] for a survey.

There are two main types of error-bounded lossy compressors: prediction-based and transformation-based. Prediction-based methods use interpolation strategies to generate an initial guess for the data. They also compute and encode any corrections that must be made to the initial guess to ensure that a strict error bound is maintained. ISABELA [24], one of the first error-controlled prediction-based compressors, uses B-splines to predict data. SZ3 [31, 32, 52], the most recent general release in the SZ compressor family, uses a Lorenzo predictor [18], cubic spline interpolation, and linear interpolation.

Recently, deep learning models have been employed as predictors for data compression, such as the autoencoder [26] and implicit neural representation (INR) [35]. An autoencoder is a neural network with two components: an encoder and a decoder. The encoder produces low-dimensional representations of the input data, while the decoder reconstructs the original input data from the output of the encoder. An INR trains a small neural network that can be used to approximate the ground truth. The neural network itself is shipped as a compressed file, and to decompress it, one must simply evaluate the network on an appropriate input. One notable INR model for volumetric scalar fields is Neurcomp [35]. However, neural-based compression is computationally expensive and suffers from limited performance.

Transform-based lossy compressors rely on domain transformations for data decorrelation. For instance, ZFP [34] divides data into small blocks. Each block is separately compressed using exponent alignment for fixed point conversion, a near-orthogonal domain transform, and embedded encoding. TTHRESH [4] treats the entire dataset as one tensor, and uses singular value decomposition (SVD) to improve the decorrelation efficiency for high-dimensional data. SPERR [28] uses wavelet transforms and SPECK coding [39] to compress data.

Tensor compression. Data are typically stored in multidimensional arrays across a variety of domains. As such, tensor compression strategies have been developed to compress multidimensional arrays. However, these strategies are not designed for compressing tensor fields specifically. Rather, they compress large individual multidimensional arrays. Some strategies use tensor decompositions such as SVD, canonical polyadic decomposition or Tucker decomposition. One example is TTHRESH [4], which uses SVD. Recently, deep-learning based approaches have been proposed for tensor compression, such as NeuKron [22], TensorCodec [23], and ELiCiT [21]. While tensor compressors are not fundamentally different from scientific compressors, scientific compressors are not typically labeled as tensor compressors due to differences in terminology across domains.

Topology and visualization of 2D tensor fields. 2D tensor fields are inherently difficult to visualize. Visualization typically involves glyphs [10], color plots, [33], or streamlines [53]; see [15] for a survey. The topology of 2D symmetric tensor fields was first applied to visualization by Delmarcelle [7]. Since then, the topology of 2D symmetric tensor fields has been applied to numerous problems in computer graphics such as remeshing [2], street layout generation [6] and scene reconstruction [48]. It has also been used in the visualization of tensor fields [19, 46] and rotation fields [37]. Outside of visualization, the degenerate points have been shown to have meaningful physical interpretations in certain domains [11, 25, 51]. The topology of asymmetric tensor fields has been utilized to support their visualization [20, 33, 38].

Topology-preserving compression. Recently, a number of topology-preserving data compression techniques have been developed. Most of the work thus far has pertained to scalar fields; to our knowledge, no topology-preserving compression technique has been developed for tensor field data. Soler et al. [43] developed a compressor that preserves the persistence diagram of a scalar field, up to a user-specified persistence threshold. Yan et al. [49] developed TopoSZ, which preserves the contour tree of a scalar field, up to a user-specified persistence threshold. Gorski et al. [12] developed a contour tree preserving compression strategy with improved performance compared to TopoSZ. Li et al. developed mSZ [29], which preserves the Morse–Smale segmentation of a scalar field. For vector fields, Liang et al. [30] developed cpSZ, which preserves the critical points of a vector field. Later, Xia et al. [47] developed TspSZ, which preserves the entire topological skeleton.

3 Technical Background

3.1 Tensors and Tensor Fields

A tensor may be represented as a multidimensional array. The rank of a tensor indicates the number of indices required to specify its components: a scalar has rank 0, a vector has rank 1, and a matrix has rank 2. Tensors play a significant role in science and engineering, such as the Riemann curvature tensor from general relativity, the stress and strain tensors in mechanics, and diffusion tensor imaging in medicine. We consider 2D symmetric and asymmetric second-order tensor fields, which we refer to simply as tensor fields when the context is clear. In this paper, a *tensor* T is a linear operator that maps any vector v to another vector $u = Tv$, where both v and u belong to the vector space $\mathbb{R}^2$. With a chosen basis of $\mathbb{R}^2$, T can be represented by a 2×2 matrix

$$T = \begin{pmatrix} T_{11} & T_{12} \\ T_{12} & T_{22} \end{pmatrix}.$$

T is *symmetric* when $T_{ij} = T_{ji}$ and *asymmetric* otherwise. Let $\mathbb{T}$ denote the space of second-order tensors. A *tensor field* $f : \mathbb{X} \to \mathbb{T}$ assigns each point x from a domain $\mathbb{X}$ to a tensor $f(x) \in \mathbb{T}$.

Consistent with previous work [19, 33], we work with piecewise-linear (PL) tensor fields. For TFZ, $\mathbb{X}$ is a flat 2D triangular mesh (i.e. all vertices have zero Gaussian curvature). f is stored at vertices of the mesh and PL interpolated. Specifically, let σ be a triangular cell with vertices v_1, v_2, and v_3. Any point $x \in \sigma$ can be written uniquely based on barycentric coordinates, that is, $x = a_1v_1 + a_2v_2 + a_3v_3$ where $a_i \geq 0$ and $\sum_i a_i = 1$. We set $f(x) = a_1f(v_1) + a_2f(v_2) + a_3f(v_3)$. When studying the topology of tensor fields, Khan et al. claimed that this interpolation scheme can lead to discontinuities when vertices have nonzero Gaussian curvature [20]. Thus, we restrict $\mathbb{X}$ to be flat.

3.2 Topology of Tensor Fields

The topology of a tensor field varies significantly depending on the order, dimension, and symmetry of the tensor field (e.g., [16, 17, 19, 40, 50]). We focus on the topology of 2D second-order tensor fields.

3.2.1 Topology of Second-Order Symmetric Tensor Fields

The topology of a second-order symmetric tensor field f consists of *degenerate points* (i.e., points with equal eigenvalues) and their connections via *tensorlines* (i.e., lines that follow the direction of the eigenvector fields) [8]. At a fixed location x, let $f(x) = T$. A symmetric tensor T has two linearly independent real eigenvectors v_1 and v_2 that correspond to real eigenvalues λ_1 and λ_2 respectively. If we order the eigenvalues such that $\lambda_1 \geq \lambda_2$, then v_1 are v_2 are the major and minor eigenvectors, respectively. The major (resp., minor) *eigenvector field* of f, denoted as e_1 (resp., e_2), maps each x to the major (resp., minor) eigenvector of $f(x)$. Integrating the eigenvector fields yields two families of continuous curves, referred to as the major and minor *tensorlines*.

A *degenerate point* of f is a point where the eigenvalues are identical and the eigenvectors are no longer defined uniquely, that is, $\lambda_1 = \lambda_2$. Degenerate points behave like the zeros of a vector field. The topology of f is defined in terms of its degenerate points, and the tensorlines that connect them. In the PL setting, there are two types of degenerate points that can occur within a cell: a trisector and a wedge, as shown in Fig. 2(A) and (B) respectively. They can be classified and detected according to the behavior of the eigenvector fields around them.

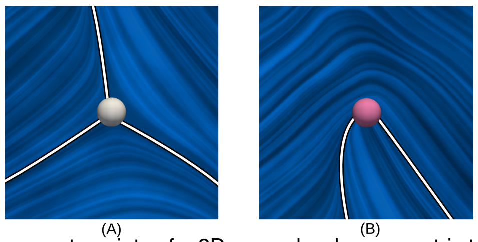

Fig. 2: Degenerate points of a 2D second-order symmetric tensor field. The major eigenvector field is visualized using the Line Integral Convolution (LIC) together with major tensorlines passing through the degenerate points; trisectors (A) are in white, wedges (B) are in pink.

We detect a degenerate point within a cell σ using the concept of a deviator following Jankowai et al. [19]. Given a cell σ, let T_1, T_2, and T_3 be tensors (represented as matrices) associated with its vertices in a clockwise order. Define the deviator of a tensor T by $D(T) := T - \frac{1}{2}\text{tr}(T)$, where $\text{tr}(T)$ is the trace. It has been shown that $D(T)$ has the same eigenvectors as T, and $\text{tr}(D(T)) = 0$ by construction [19]. For each T_i, denote the entries of its deviator by

$$D(T_i) := \begin{pmatrix} \Delta_i & F_i \\ F_i & -\Delta_i \end{pmatrix}. \quad (1)$$

For a pair T_i and T_j, let

$$l_{i,j} := \text{sign}(F_j\Delta_i - F_i\Delta_j). \quad (2)$$

Then σ contains a wedge if $l_{1,2} = l_{2,3} = l_{3,1} = 1$, and a trisector if $l_{1,2} = l_{2,3} = l_{3,1} = -1$; otherwise, if $l_{i,j} \neq 0$ for all pairs, σ contains no degenerate points. Jankowai et al. [19] did not consider the scenario where $l_{i,j} = 0$ and instead applied small perturbations to prevent it from occurring. We address such a scenario explicitly in Sec. 4.2.

3.2.2 Topology of Second-Order Asymmetric Tensor Fields

Zhang et al. [50] introduced the tensor decomposition of any second-order tensor T,

$$T = \gamma_d \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \gamma_r \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} + \gamma_s \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{pmatrix}, \quad (3)$$

where $\gamma_d = \frac{T_{11}+T_{22}}{2}$, $\gamma_r = \frac{T_{21}-T_{12}}{2}$, and

$$\gamma_s = \frac{\sqrt{(T_{11}-T_{22})^2 + (T_{12}+T_{21})^2}}{2}.$$

γ_d, γ_r, and γ_s reflect the strengths of isotropic scaling, rotation, and anisotropic stretching, respectively [50]; where $\gamma_s \geq 0$ and $\gamma_d, \gamma_r \in \mathbb{R}$. $\theta \in [0, 2\pi)$ is the angular component of the vector

$$\begin{pmatrix} T_{11} - T_{22} \\ T_{12} + T_{21} \end{pmatrix}.$$

In a PL setting, γ_d and γ_r interpolate linearly, whereas the interpolation of γ_s is more involved; see [50] for details.

Dual-eigenvector field. Although the eigenvectors of an asymmetric tensor can be complex, Zheng and Pang [54] introduced the major and minor *dual-eigenvector fields*, which are real-valued eigenvector fields derived from an asymmetric tensor field f. For an asymmetric tensor T, the major and minor *dual-eigenvectors* of T are the major and minor eigenvectors of a symmetric tensor [50, Theorem 4.2]:

$$P_T = \frac{\gamma_r}{|\gamma_r|}\gamma_s \begin{pmatrix} \cos\left(\theta + \frac{\pi}{2}\right) & \sin\left(\theta + \frac{\pi}{2}\right) \\ \sin\left(\theta + \frac{\pi}{2}\right) & -\cos\left(\theta + \frac{\pi}{2}\right) \end{pmatrix}. \quad (4)$$

Thus the dual-eigenvector fields are the major and minor eigenvector fields of the symmetric tensor field $x \mapsto P_{f(x)}$.

Recall that any square matrix T can be written as $T = T_S + T_A$, where $T_S = \frac{1}{2}(T + T^\top)$ is the symmetric part of T and T_A is the antisymmetric part. The symmetric part of a tensor T is defined as the average of the tensor with its transpose. Following Eq. (3),

$$T_S = \gamma_d \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \gamma_s \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) \end{pmatrix}. \quad (5)$$

Accordingly, let f_S be the symmetric part of f. A point is an isolated degenerate point (and not part of a larger degenerate curve or region) in the dual-eigenvector field of f if and only if it is an isolated degenerate point in the eigenvector field of f_S. Further, the type of that degenerate point (trisector or wedge) remain the same [50, Theorem 4.4].

Eigenvector manifold and eigenvalue manifold. To illustrate the structures in asymmetric tensor fields, Zhang et al. [50] introduced the notions of eigenvalue manifold (a hemisphere) and eigenvector manifold (a sphere), that encode the relative strengths of three components in Eq. (3) affecting the eigenvalues and eigenvectors in the tensor. For this purpose, we assume that $\gamma_d^2 + \gamma_r^2 + \gamma_s^2 = 1$. Following [50], the *eigenvector manifold* is defined as a sphere

$$\{(\gamma_r, \gamma_s, \theta) \mid \gamma_r^2 + \gamma_s^2 = 1 \text{ and } \gamma_s \geq 0 \text{ and } 0 \leq \theta < 2\pi\}, \quad (6)$$

whereas the *eigenvalue manifold* is defined as a hemisphere

$$\{(\gamma_d, \gamma_r, \gamma_s) \mid \gamma_d^2 + \gamma_r^2 + \gamma_s^2 = 1 \text{ and } \gamma_s \geq 0\}. \quad (7)$$

Eigenvector partition. We visualize the eigenvector manifold of Eq. (6) in Fig. 3(A). To construct the manifold, we first define a semicircle $\gamma_r^2 + \gamma_s^2 = 1$ (where $\gamma_r \geq 0$). Then, we rotate such a semicircle around a vertical axis to obtain a sphere. On this sphere, each longitude line that runs from the North Pole to the South Pole corresponds to a value of θ, and each latitude line corresponds to a value of γ_r.

Next, we partition the eigenvector manifold based on (a) whether $|\gamma_r|$ or γ_s is greater and (b) the sign of γ_r. Fig. 3(A) illustrates four regions from such a partition using a categorical colormap, from the top to bottom: the first region near the North Pole corresponds to the

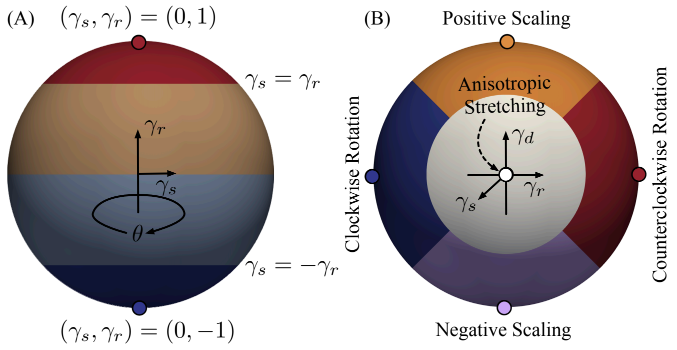

Fig. 3: (A) The eigenvector manifold colored according to its partition based on γ_r and γ_s. (B) The eigenvalue manifold colored according to its partition based on γ_d, γ_r, and γ_s.

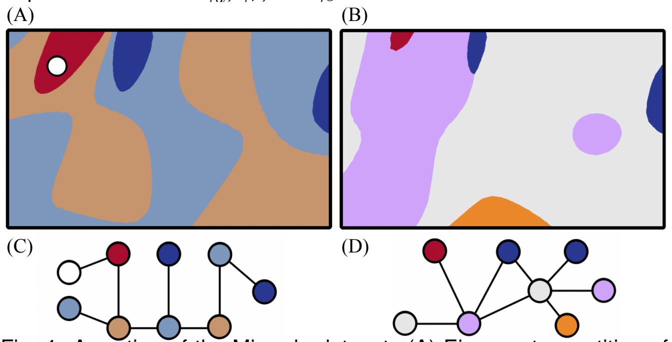

Fig. 4: A portion of the Miranda dataset. (A) Eigenvector partition of the domain following the colormap of Fig. 3(A); degenerate points of the dual-eigenvector field are shown in white (trisectors) or pink (wedges). (B) Eigenvalue partition of the domain following the colormap of Fig. 3(B). (C) Eigenvector graph corresponding to the eigenvector partition in (A). (D) Eigenvalue graph corresponding to the eigenvalue partition in (B).

parameter setting where $\gamma_r > 0$ and $|\gamma_r| > \gamma_s$; the second region corresponds to $\gamma_r > 0$ and $|\gamma_r| < \gamma_s$; the third region corresponds to $\gamma_r < 0$ and $|\gamma_r| < \gamma_s$; and the fourth region near the South Pole corresponds to $\gamma_r < 0$ and $|\gamma_r| > \gamma_s$. The two regions near the equator correspond to real eigenvalues, and the regions near the Poles correspond to complex eigenvalues.

For a tensor field f, each tensor $f(x)$ at a location $x \in \mathbb{X}$ maps to a point within a partition of the eigenvector manifold. Therefore, we can visualize the domain of f based on the partition of its corresponding eigenvector manifold; this is referred to as the *eigenvector partition*, as illustrated in Fig. 4(A). The north and south poles correspond to isolated degenerate points of the dual eigenvector field, and such points are typically part of the partition visualization.

Eigenvalue partition. The eigenvalue manifold of Eq. (7) is designed to highlight the dominant operation between isotropic scaling (γ_d), rotation (γ_r), and anisotropic stretching (γ_s). We visualize its corresponding hemisphere in Fig. 3(B), where the point $(\gamma_r, \gamma_d, \gamma_s) = (0, 0, 1)$ corresponds to the point labeled "anisotropic stretching."

We then partition the eigenvalue manifold according to (a) which coefficient has the largest magnitude and (b) the sign of that coefficient. Each region in the partition corresponds to a different type of linear transformation. In Fig. 3(B): the top orange region (Positive Scaling) corresponds to when γ_d has the largest magnitude and $\gamma_d > 0$; the red region (Counterclockwise Rotation) corresponds to when γ_r has the largest magnitude and $\gamma_r > 0$; the purple region (Negative Scaling) corresponds to when γ_d has the largest magnitude and $\gamma_d < 0$; the blue region (Clockwise Rotation) corresponds to when γ_r has the largest magnitude and $\gamma_r < 0$; and finally, the white region in the middle corresponds to when γ_s has the largest magnitude.

Similarly, a partition of the eigenvalue manifold could be used to visualize the domain of a tensor field, referred to as the *eigenvalue partition*, as illustrated in Fig. 4(B).

Eigenvector graph and eigenvalue graph. To describe the topology of an asymmetric tensor field, Lin et al. [33] introduced the notion of an *eigenvector graph*, that is, the dual graph of the eigenvector partition of the domain. Each node corresponds to a region of the partition or one of the poles of the eigenvector manifold, and is colored by the region type (or the degenerate point type). Each edge encodes an adjacency relation among a pair of regions (or a region and a degenerate point). We visualize a tensor field based on its eigenvector partition in Fig. 4(A), and its eigenvector graph in (C).

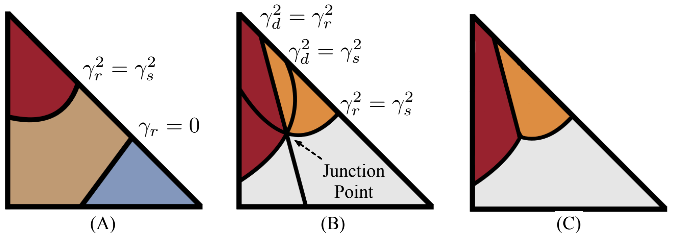
$\gamma_r^2 = \gamma_s^2$
$\gamma_r = 0$
(A)
$\gamma_d^2 = \gamma_r^2$
$\gamma_d^2 = \gamma_s^2$
$\gamma_r^2 = \gamma_s^2$
Junction Point
(B)
(C)

Fig. 5: (A) A cell is partitioned into three regions according to the eigenvector manifold by tracing boundary curves $\gamma_r^2 = \gamma_s^2$ and $\gamma_r = 0$. (B) A cell is partitioned into six regions by tracing the boundary curves of $\gamma_d^2 = \gamma_r^2$, $\gamma_d^2 = \gamma_s^2$ and $\gamma_r^2 = \gamma_s^2$ that intersect at the junction point. The regions are classified according to the eigenvalue partition. (C) Adjacent regions of the same type based on eigenvalue partition in (B) are merged inside a cell.

As illustrated in Fig. 5(A), to compute the eigenvector graph, we first trace the boundary curves defined by $\gamma_r^2 = \gamma_s^2$ (a conic section) and $\gamma_r = 0$ (a line) within each cell. These curves do not intersect with each other and simply divide a cell according to the eigenvector partition. Next, we iteratively merge adjacent regions of the same type in neighboring cells. Finally, we detect each connected component of the partition, from which we compute the eigenvector graph.

Analogously, Lin et al. [33] also introduced the *eigenvalue graph* as the dual of the eigenvalue partition of the domain. We visualize a tensor field based on its eigenvalue partition in Fig. 4(B), and its corresponding eigenvalue graph in Fig. 4(D). Following [33], as illustrated in Fig. 5(B), to compute the eigenvalue graph, we trace the boundary curves within each cell: $\gamma_d^2 = \gamma_r^2$ (two intersecting lines), $\gamma_d^2 = \gamma_s^2$ (a conic section), and $\gamma_r^s = \gamma_s^2$ (another conic section). We then merge adjacent regions of the same type in the same cell (see Fig. 5(C)) and among neighboring cells to construct the eigenvalue graph.

For compression purposes, we focus on preserving the topology inside each cell, that is, the cell-wise eigenvector (or eigenvalue) partition, without explicitly constructing the eigenvector (or eigenvalue) graph.

3.3 Quantization in Lossy Compression

We review two quantization techniques, linear-scaling quantization and logarithmic-scaling quantization, used by TFZ.

Linear-scaling quantization. A popular lossy data compressor, SZ1.4 [45], introduces *linear-scaling quantization* that ensures a pointwise absolute error bound ξ is maintained. Suppose that $f : \mathbb{X} \to \mathbb{R}$ (defined on a finite domain $\mathbb{X}$) is a scalar field to be compressed. Suppose that $g : \mathbb{X} \to \mathbb{R}$ is an initial guess for f (e.g., from a regression predictor). Let $f' : \mathbb{X} \to \mathbb{R}$ be the final decompressed data. Then for each $x \in \mathbb{X}$, we can shift $g(x)$ by an integer multiple of 2ξ to obtain a final value of $f'(x)$ such that $|f'(x) - f(x)| \le \xi$. To conceptualize linear-scaling quantization for a point x in Fig. 6, we divide the real line into intervals of width 2ξ, one of which is centered on $g(x)$. Then we compute how many intervals to shift $g(x)$ to obtain a value for $f'(x)$ that lies in the same interval as $f(x)$. By construction, if $f(x)$ lies in an interval of width 2ξ centered on $f'(x)$, it must hold that $|f(x) - f'(x)| \le \xi$.

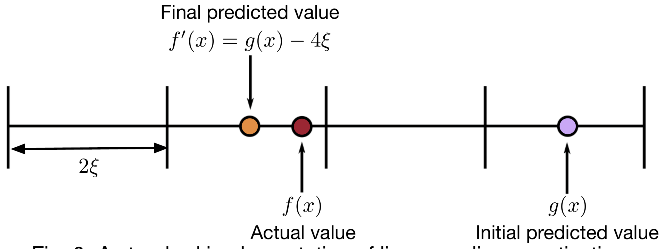

Fig. 6: A standard implementation of linear-scaling quantization.

During construction, we assign to each $x \in \mathbb{X}$ some integer n_x such that $f'(x) = g(x) + n_x(2\xi)$, e.g., $n_x = -2$ in Fig. 6. If the entropy of the $\{n_x\}$ is low (e.g. the $\{n_x\}$ are mostly zero), then the $\{n_x\}$ can be efficiently compressed using an entropy-based compressor, such as the Huffman coding. More accurate predictions for $g(x)$ generally lead to distributions of $\{n_x\}$ with lower entropy.

Logarithmic-scaling quantization. In topology-preserving data compression, it can be advantageous for certain data points to have predicted values that are nearly identical to the ground truth. Gorski et al. [12] introduced logarithmic-scaling quantization, enabling a compressor to apply linear-scaling quantization with varying error bounds for different points without needing to store additional information about the specific error bound used. Each $x \in \mathbb{X}$ is assigned a precision p_x, so that its error bound is $2\xi/(2^{p_x})$. The value of p_x is set independently for each point. If the compressor determines that, for a point x, $f'(x)$ must be very close to $f(x)$, then it can set p_x to be high.

To encode a point using logarithmic-scaling quantization, we perform standard linear-scaling quantization on each $x \in \mathbb{X}$ using an error bound of $2\xi/(2^{p_x})$ to obtain a quantization number n_x, such that x will have the decompressed value $f'(x) = g(x) + 2\xi n_x/(2^{p_x})$. Let P be the maximum value for p_x. When encoding the compressed data, we store the integer $a_x \leftarrow n_x \times 2^{P-p_x}$. Then, when decompressing the data, the decompressed value for x is $f'(x) = g(x) + 2\xi a_x/(2^P)$. Notice that x will have its correct decompressed value because

$$f'(x) = g(x) + \frac{2\xi a_x}{2^P} = g(x) + \frac{2\xi n_x 2^{P-p_x}}{2^P} = g(x) + \frac{2\xi n_x}{2^{p_x}}. \quad (8)$$

4 Method

Although the topology and topological guarantees differ for symmetric and asymmetric tensor fields, our overall compression pipeline remains largely the same in both cases. We describe our general pipeline in Sec. 4.1. We describe the specifics for symmetric tensor fields in Sec. 4.2 and asymmetric tensor fields in Sec. 4.3.

4.1 An Overview of Compression Pipeline

We store an asymmetric tensor field as four scalar fields that correspond to the four entries of a 2×2 matrix. For a symmetric tensor field, we only store three scalar fields due to symmetry (i.e., for any symmetric tensor T, $T_{1,2} = T_{2,1}$). In addition to providing topological guarantees, our compression pipeline ensures that individual entries of a tensor do not vary by more than a user-specified error bound ξ.

Following the previous work by Gorski et al. [12], we design an *augmentation layer* that runs on top of an existing compressor (referred to as a *base compressor*) and corrects the topology of the decompressed output. TFZ can augment any error-bounded lossy compressor to provide topological guarantees. Our pipeline is as follows:

Step 1. Base compressor. We compress each scalar field using the base compressor with an error bound ξ. This operation ensures that the error of individual entries of each tensor is at most ξ. We then decompress the data to analyze any changes that must be made to preserve topology. We refer to the compressed-then-decompressed output of the base compressor as the *intermediate data*.

Step 2. Cell correction. TFZ ensures that the topology of each individual cell is maintained. We scan across each cell, making corrections to the tensors at the vertices of each cell as needed. We describe this step in more detail in Sec. 4.2 and Sec. 4.3.

If, when processing a cell σ, we alter the tensor $f(x)$ at some vertex x, other cells that share x as a vertex may be altered topologically. Thus, we re-process every previously processed cell σ' that shares x as a vertex, and track which cells must be re-processed using a stack. This strategy can create cycles between adjacent cells. To prevent infinite loops, we store a tensor losslessly if it is modified 20 times.

Step 3. Lossless compression. Our cell compression step outputs integer values for each cell according to quantization strategies detailed in Sec. 3.3. We also generate some integers that serve as flags. We encode these integers using Huffman coding. We combine the output of Huffman coding, any information stored losslessly, and the four compressed fields from the base compressors into one tar archive, and then compress the archive using ZSTD (a lossless compressor).

4.2 Compression of Symmetric Tensor Fields

Topological guarantees. TFZ ensures that each cell retains its classification (i.e., degeneracy type): whether it contains no degenerate points, a single degenerate point, a degenerate line (i.e., a line of degenerate points), or is entirely degenerate (i.e., a cell of degenerate points). For cells with a single degenerate point in their interior, we further distinguish between trisectors and wedges. Additionally, for cells containing a degenerate line or a degenerate vertex, TFZ preserves the precise location of the degenerate feature.

Theoretical basis. We provide the theoretical basis for the cell preservation step, where the lemmas are proven in the supplement. In our first lemma, we apply the decomposition in Eq. (3) to symmetric tensors. In the symmetric case, we must have $\gamma_r = 0$. We then obtain the decomposition of Eq. (5), where the γ_s term corresponds to the deviator.

Lemma 1. *Let T_1 and T_2 be two 2×2 symmetric tensors with nonzero deviators. Let $l_{1,2}$ be defined following Eq. (2). Then $l_{1,2}$ depends only on θ_1 and θ_2, where each θ_i is taken from the decomposition in Eq. (5).*

Lemma 2. *Let x_1, x_2, and x_3 be the vertices of a cell.*

(a) Suppose that exactly one vertex has a tensor T with a deviator $D(T)$ equal to zero. (w.l.o.g., suppose $D(f(x_1)) = 0$).

(i) If $l_{2,3} \neq 0$, then x_1 is the only degenerate point in the cell.

(ii) If $l_{2,3} = 0$, then there exists some k such that $D(f(x_2)) = kD(f(x_3))$. If $k > 0$ then x_1 is the only degenerate point in the cell. If $k < 0$ then the cell contains a degenerate line.

(b) If exactly two vertices have a tensor with a deviator equal to zero, then the cell contains a degenerate line connecting them.

Cell correction. We scan through each cell σ of the mesh. Let T_1, T_2, and T_3 be the tensors at the vertices of σ for the ground truth data, and let T'_1, T'_2, and T'_3 be the corresponding tensors in the intermediate data. Following the tensor decomposition of Eq. (5), we obtain coefficients $\gamma_{d,1}$ for T_1, $\gamma'_{d,1}$ for T'_1, etc. We compute $l_{1,2}$, $l_{2,3}$, and $l_{3,1}$ for the ground truth data at σ and evaluate the following two cases.

Case 1. All $l_{i,j} \neq 0$. We first verify whether the degeneracy type of σ (i.e., trisector, wedge, or non-degenerate) in the intermediate data matches the ground truth. If it does, we proceed to the next cell. Otherwise, we examine θ'_1, θ'_2, and θ'_3 and identify the θ'_i value that deviates the most from the ground truth. This value is then corrected to be θ''_i using linear-scaling quantization with an error bound of $2\pi/(2^6 - 1)$, ensuring that the quantization number n_x can be stored using six bits. Once θ''_i is computed, we use it along with the coefficients $\gamma'_{d,i}$ and $\gamma'_{s,i}$ to reconstruct a new tensor T''_i based on the decomposition.

According to Lemma 1, adjusting the values of θ'_i is sufficient to ensure that the $l_{i,j}$ match between the ground truth and reconstructed data. Likewise, Jankowai et al. [19] showed that the $l_{i,j}$ determine the degeneracy type of a cell σ. If adjusting one of the θ'_i does not correct the topology of the cell, we adjust the other two values as necessary.

In rare cases, due to limited precision, adjusting all three of the θ'_i using linear-scaling quantization does not preserve the topology. It is also possible that the error bound may not be maintained. In such cases, we store the deviator losslessly, and if necessary the entire tensor. We provide specifics in the supplement.

Case 2. One of the $l_{i,j} = 0$. In this case, if none of the T_i are equal to zero, we store the vertices of σ losslessly. Otherwise, we store the T_i that equal zero losslessly, and compute the topology of σ following Lemma 2 for both the ground truth and reconstructed data. If the topology between the ground truth and the reconstructed data does not match, we store the vertices of σ losslessly. If any of the T_i satisfy $D(T_i) = 0$ but $T_i \neq 0$, then we also store the vertices of σ losslessly.

4.3 Compression of Asymmetric Tensor Fields

Topological guarantees. For each cell σ, TFZ preserves the topological structure of both the eigenvalue and eigenvector partitions. Consequently, the eigenvector and eigenvalue graphs are maintained across the entire domain. Since the isolated degenerate points of the dual-eigenvector fields are part of the eigenvector graph, we also preserve the topology of the dual-eigenvector fields using the same approach as for symmetric tensor fields, operating on the symmetric component of the tensor field. However, when the deviator would be stored losslessly, we instead store θ losslessly. TFZ allows users to choose whether to preserve topology based on the eigenvector manifold, eigenvalue manifold, or both. While we describe the process for preserving both, it can be easily adapted to maintain only one.

For any cell σ, preserving its topology involves two main steps. First, we ensure that each vertex of σ is correctly classified within the appropriate type of region in both the eigenvalue and eigenvector partitions. Next, we preserve the internal topology of σ by maintaining the locations and connectivity of the different partition regions. These processes are detailed in Sec. 4.3.1 and Sec. 4.3.2, respectively. Additionally, our method accounts for various special cases that require careful handling, which we discuss in the supplement.

4.3.1 Asymmetric Cell Correction: Vertex Correction

In this portion of the algorithm, we ensure that if x is a vertex of a cell σ, and then the classifications of $f(x)$ and $f'(x)$ match according to the eigenvalue and eigenvector partitions.

Theoretical basis. We provide the theoretic basis below, see the supplement for proofs. We define a function sign : $\mathbb{R} \to \mathbb{R}$ that satisfies $\text{sign}(x) = 1$ if $x \geq 0$ and $\text{sign}(x) = -1$ otherwise.

Lemma 3. *Suppose T and T' are 2×2 tensors, where each entry of T' differs from the corresponding entry of T by at most ξ, i.e., $|T_{1,1} - T'_{1,1}| \leq \xi$, and so on. Denote their coefficients from the decomposition as γ_d, γ_r, γ_s, γ'_d, γ'_r, and γ'_s, respectively. Then $|\gamma_d - \gamma'_d| \leq \xi$, $|\gamma_r - \gamma'_r| \leq \xi$, and $|\gamma_s - \gamma'_s| \leq \sqrt{2}\xi$.*

Lemma 4. *Suppose that $x \in \mathbb{R}$ and x' is a guess for x within an error bound ξ, i.e., $|x - x'| \leq \xi$.*

- *If $x > 0$ but $x' < 0$. Then $x' + \xi > 0$ and $|x - (x' + \xi)| \leq \xi$. That is, $x' + \xi$ is a valid guess for x that is within an error bound ξ and has the same sign as x.*
- *If $x < 0$ but $x' > 0$, then $x' - \xi$ is analogously valid.*

By applying Lemma 4, we can generate a guess x' for x that shares the same sign as x. In this context, if x' and x already have the same sign, no further adjustment is needed.

Lemma 5. *Suppose that $x, y \in \mathbb{R}$ and x' and y' are guesses for x and y, respectively, within an error bound ξ, such that $|x - x'| \leq \xi$ and $|y - y'| \leq \xi$. Suppose that $|x| > |y|$ but $|x'| < |y'|$.*

Let $x'' = \text{sign}(x')|y'|$, $y'' = \text{sign}(y')|x'|$. Then $|x - x''| \leq \xi$ and $|y - y''| \leq \xi$, and $|x''| > |y''|$. That is, by swapping the magnitudes of x' and y', we can obtain two valid guesses x'' and y'' with $|x''| > |y''|$.

Vertex correction algorithm. Let T be a tensor from the ground truth and T' be the corresponding tensor in the intermediate data. Denote their coefficients from the decomposition as γ_d, γ_r, γ_s, γ'_d, γ'_r, and γ'_s respectively. To ensure that the classifications of T' matches T, we must preserve four properties. For the eigenvector manifold, we must preserve: (1) whether $|\gamma_r| > \gamma_s$, and (2) the sign of γ_r. For the eigenvalue manifold, we must preserve: (3) which of $|\gamma_r|$, $|\gamma_d|$, or γ_s is the largest, and (4) the sign of that coefficient. We also ensure that each of γ'_d, γ'_r and γ'_s differs from its ground truth value by at most ξ.

For each T represented as a matrix, we check whether these four properties have been preserved in T'. If not, we correct T'. To do so, we introduce five variables, S, R, D, RS, and $\mathsf{D_m}$. Each variable serves as a flag indicating whether an adjustment is needed for γ'_d, γ'_r, or γ'_s to ensure the preservation of the four properties. In particular,

- S is enabled when γ'_s needs adjustment to satisfy $|\gamma_s - \gamma'_s| \leq \xi$. According to Lemma 3, this can be achieved by adding or subtracting $(\sqrt{2} - 1)\xi$ from γ'_s.
- R is enabled when γ'_r requires a sign change. The sign is adjusted using the approach outlined in Lemma 4.
- D is enabled when γ'_d requires a sign change, which is performed using the strategy described in Lemma 4.
- RS is enabled if the ordering of $|\gamma'_r|$ and γ'_s is incorrect (i.e., one is larger than the other when it should be smaller). Their magnitudes are swapped using the approach outlined in Lemma 5.
- $\mathsf{D_m}$ is enabled if $|\gamma'_d|$ should have a larger magnitude than $|\gamma'_r|$ and $|\gamma'_s|$, but does not. We also set $\mathsf{D_m}$ if $|\gamma'_d|$ has the largest magnitude, but it should be smaller than $|\gamma'_r|$ or $|\gamma'_s|$. In such cases, we swap the magnitudes using the strategy in Lemma 5.

Once we have made appropriate adjustments to γ'_d, γ'_r and γ'_s, we reassemble T' from the coefficients according to Eq. (3). Although unlikely, it is possible that after doing so, T' may not adhere to the element-wise error bound ξ. In such a case, we quantize γ'_d, γ'_r and γ'_s using logarithmic-scaling quantization with increased precision and repeat; we report specifics on these adjustments in the supplement.

4.3.2 Asymmetric Cell Correction: Cell Topology Correction

Preserving the classifications of the vertices is often (but not always) sufficient to preserve the cell topology. In this step, we ensure the topology of each cell σ is preserved.

First, we preserve the degeneracies of the symmetric component of the tensor field using the same strategy applied to symmetric tensor fields. However, when the procedure would allow the deviator to be stored losslessly, we instead store the entire matrix losslessly.

Second, we preserve the topology of each cell σ to ensure that the global eigenvector and eigenvalue graphs are maintained. These graphs are computed by determining the topology of each cell individually and then merging regions of the same type in adjacent cells. We preserve both the eigenvector and eigenvalue graphs for σ, as well as the edges and vertices of σ that border each region of the corresponding partitions. This approach guarantees that the merging step in the global eigenvalue and eigenvector graph computation will produce consistent results.

Theoretical basis. The following lemmas are relevant to computing the cell topology of σ, whose proofs are in the supplement.

Lemma 6. *Except in special cases, the topology of the eigenvector partition of σ is determined by the following conditions:*

(a) Whether each of the curves $\gamma_r = \gamma_s$ and $-\gamma_r = \gamma_s$ intersects the interior of σ.
(b) How many times the curves $\gamma_r = \gamma_s$ and $-\gamma_r = \gamma_s$ intersect each edge of σ.
(c) The classification of each vertex of σ according to the eigenvector partition.

Lemma 7. *Except in special cases, the topology of the eigenvalue partition of σ is determined by the following conditions:*

(a) The classification of each vertex in the eigenvalue partition.
(b) For each of the curves $\gamma_d = \gamma_s$, $-\gamma_d = \gamma_s$, $\gamma_r = \gamma_s$, and $-\gamma_r = \gamma_s$, determine the order in which the following points are encountered when traveling counterclockwise around the curve.
 (i) Each boundary point (i.e, a point on the boundary between two regions of the eigenvalue partition) where the curve enters or leaves σ, and the orientation of the curve at that point.
 (ii) Each junction point, along with the orientations of the curves that intersect at that point.
(c) For each edge e of σ, for each point identified in (b.i) that lies on e, determine which point is closest to each vertex of e.
(d) For each region type in the eigenvalue partition, except where $\gamma_s > |\gamma_d|$ and $\gamma_s > |\gamma_r|$, if its boundary curve does not intersect any edges or other curves, check whether that region is present in the eigenvalue partition of σ.

Both Lemma 6 and Lemma 7 admit special cases proved and discussed in the supplement.

Cell topology preservation algorithm. To accomplish our goal, we create a topological invariant that allows us to verify whether the topology of σ in the intermediate data aligns with the ground truth. This invariant is represented as a data structure containing information about the cell. As part of the invariant, we include the classification of each vertex based on the eigenvector and eigenvalue manifolds. Additionally, we incorporate other relevant information, which we will describe next.

Eigenvector manifold invariant. By Lemma 6, the topology of the eigenvector partition of σ is determined by the boundaries of the regions where $\gamma_r > \gamma_s$ and $-\gamma_r > \gamma_s$. We divide the curve $\gamma_r^2 = \gamma_s^2$ into two connected components: $\gamma_r = \gamma_s$ and $-\gamma_r = \gamma_s$. Next, in our invariant, we track how many times each curve intersects each edge. If the curves do not intersect any edges but intersect the interior of σ, we also record this case in the invariant. In Fig. 7(A), we illustrate one possible example. In this example, we would record that the curve

$\gamma_r = \gamma_s$ intersects the hypotenuse once and the left edge once, while the curve $-\gamma_r = \gamma_s$ does not intersect the cell.

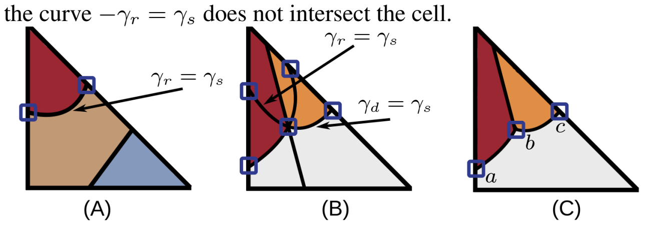

Fig. 7: (A) We find the edge intersections for the curve $\gamma_r = \gamma_s$. (B) We compute the edge intersections and junction points for the curves $\gamma_d = \gamma_s$ and $\gamma_r = \gamma_s$. (C) We only retain cell intersections that are topologically significant.

Eigenvalue manifold invariant. Following Lemma 7, to describe the cell topology based on the eigenvalue manifold, we need to track where each of $\gamma_d^2 = \gamma_s^2$ and $\gamma_r^2 = \gamma_s^2$ intersects each edge of the cell σ, as well as where they intersect each other. To accomplish this, we divide each curve into two components, resulting in four total curves: $\gamma_d = \gamma_s$, $-\gamma_d = \gamma_s$, $\gamma_r = \gamma_s$, and $-\gamma_r = \gamma_s$. We then trace each curve clockwise, recording the order in which it intersects an edge or another curve, and noting the specific edge or curve it intersects in each case, and the orientation. If an intersection does not affect the topology, we omit it from our tracking. This process produces four lists of intersections, which we incorporate into our invariant.

In Fig. 7(B), we provide an example showing the intersection points we compute, which are four edge intersections and one junction point. However, two of these intersections do not affect the topology, so we exclude them from our tracking. In Fig. 7(C), we show the three intersections that we do track. Ultimately, our invariant records that the curve $\gamma_d = \gamma_s$ starts by intersecting the hypotenuse at c and ends at a junction point at b. The curve $\gamma_r = \gamma_s$ starts at a junction point at b and ends by intersecting the left edge at a. The curves $-\gamma_d = \gamma_s$ and $-\gamma_r = \gamma_s$ do not contribute to the topology. In our invariant, we do not track the specific locations a, b, and c; instead, we track whether the intersection is a junction point, and if not, which edge it lies on.

Topology correction. We compute our invariant for σ for both the ground truth and reconstructed data. If there is a mismatch, then we procedurally quantize θ using linear-scaling quantization and lower the error bounds of γ_d, γ_r, and γ_s using logarithmic-scaling quantization. We provide details in the supplement. After the degenerate points are preserved, it is possible that one will lie in the wrong region type in the eigenvector partition. In such a case, we use the same adjustment process used to correct degenerate point errors.

Encoding of adjustments. Finally, once the correction for each cell σ is complete, we generate several lists of integers. Since many of our variables require fewer than 8 bits, we combine multiple variables into a single byte. For each matrix T, we combine S with the quantization number for θ to create an 8-bit integer. We also group R, D, RS, and $\mathrm{D_m}$ into another 8-bit integer. These integers, along with the quantization numbers for γ_d, γ_r, and γ_s, are encoded using Huffman coding and included in the final compressed file.

5 Experimental Results

We provide an overview of our experiments in Sec. 5.1. We evaluate the performance of TFZ with a comparison between augmented and base compressors in Sec. 5.2. We include an analysis of run time in Sec. 5.3. We aim to preserve both eigenvalue and eigenvector graphs; preserving only one of them is included in the supplement, where we also provide error maps, as well as statistics on the number of times that cells are visited and on the topological errors made by the base compressors.

Highlighted results. We highlight our experimental results below.

- Applying SZ3 or SPERR to any of our eight datasets results in numerous topological errors, whereas augmenting them with TFZ eliminates topological errors in every case.
- Augmented SPERR produces a better tradeoff between bit-rate and Peak Signal-To-Noise Ratio (PSNR) compared to augmented SZ3.
- TFZ takes $O(nk)$ time, where n is the number of cells and k is the maximum number of times a cell is processed before being stored losslessly. In practice, most of the run time is spent on cell correction.

5.1 Overview of Experiments

We present an experimental study of our framework, TFZ, by augmenting two state-of-the-art error-bounded lossy compressors for 2D scientific data: SZ3 [32] and SPERR [28]. We also experimented with ZFP [34], TTHRESH [4], and Neurcomp [35]. However, augmented ZFP [34] yielded poor compression ratios, while TTHRESH and Neurcomp did not natively support 2D data; therefore these compressors were excluded from our analysis.

We tested TFZ on four symmetric and four asymmetric tensor fields (Tab. 1). Each contains a number of 2D slices, and we compress each slice individually and report evaluation metrics that are aggregated across all slices. The Stress datasets are stress tensor fields. The Brain datasets are from brain MRI scans. The Ocean data is from an ocean flow simulation, the Miranda data is from a turbulence simulation, and the Vortex Street and the Heated Cylinder data come from fluid dynamics simulations. See the supplement for details on the datasets.

When running a base compressor (SZ3 or SPERR) on each slice, we compress each of the four fields separately, similar to TFZ. We then combine the four compressed outputs into one tar archive, and compress it losslessly with ZSTD. For the Brain datasets, empty regions outside the brain are set to zero. In those regions, small perturbations introduced by lossy compressors may lead to many topological errors. When testing the lossy compressors on the brain datasets, we store all zero tensors losslessly, introducing minimal overhead to the compressed file while eliminating the topological errors outside of the brain. We denote the choice of a dataset, base compressor, and error bound as a trial. We run each trial on a personal computer with an Intel Core i9-12900HK processor and 32GB of RAM. Our implementation is written in Julia version 1.10.2, and does not include any parallelization.

Table 1: Scientific datasets used for compression analysis. "Sym." means symmetric and "Asym." means asymmetric dataset respectively. "Slice Dim." means the dimension of each 2D slice.

Dataset	Type	Slice Dim.	#Slices	Size (MB)
Stress A	Sym.	65×65	25	3.4
Stress B	Sym.	65×65	25	3.4
Brain A	Sym.	66×108	76	17.3
Brain B	Sym.	148×190	157	141.3
Ocean	Asym.	101×101	27	8.8
Miranda	Asym.	384×384	256	1208.0
Vortex Street	Asym.	640×80	1501	2459.2
Heated Cylinder	Asym.	150×450	2000	4320.0

Evaluation metrics. For each symmetric tensor field, we evaluate how many cells have their degeneracy type (trisector, wedge, non-degenerate etc.) preserved compared to the ground truth. For each asymmetric tensor field, we evaluate how many cells have their eigenvector and eigenvalue partitions preserved. We check the topology using the invariant described in Sec. 4.3.2. For each trial, we also study the tradeoff between compressed size and reconstruction quality, by measuring the standard compression metrics such as compression ratio, bit-rate and PSNR. Recall the compression ratio is the original file size divided by the compressed file size. Bit-rate is the average number of bits used to encode each tensor. PSNR measures the reconstruction quality (higher values are better). Finally, we quantify total compression and augmentation time, and decompression time.

Parameter configurations. We define the range of a tensor field as the largest entry in any tensor minus the smallest entry in any tensor. We vary the pointwise error bound ξ, which represents the error bound as a percentage of the range. For example, $\xi = 0.01$ means that the global error bound is 1% of the range. We recompute ξ for each slice. We set the pointwise error bound for SZ3 and SPERR to ξ.

5.2 Performance Evaluation

Topological guarantees. We empirically verify that TFZ perfectly preserves the topology of each cell in each trial. By contrast, when we run the base compressors SZ3 and SPERR with very low error bounds, such that the resulting data has a comparable compressed size to TFZ, we still notice a large number of topological errors.

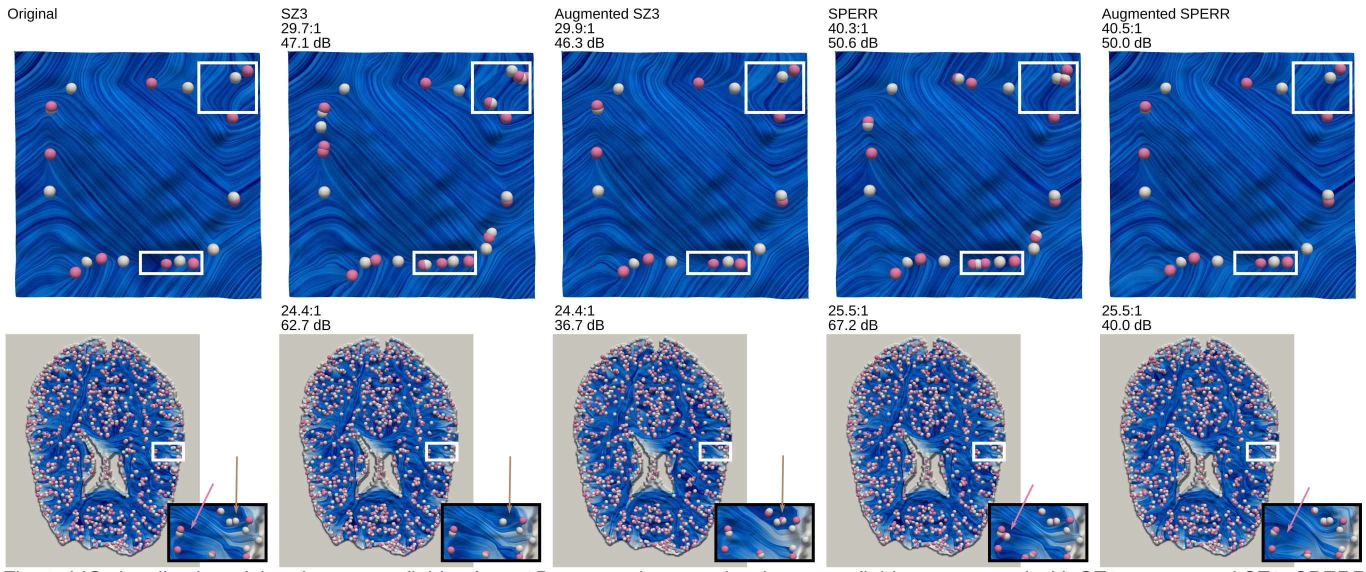

Fig. 8: LIC visualization of the eigenvector fields of two 2D symmetric second-order tensor fields compressed with SZ3, augmented SZ3, SPERR, and augmented SPERR, along with the ground truth. Trisectors are in white, wedges are in pink. Top: Stress B data slice 10. Bottom: Brain B data slice 50. In the top row, we highlight a region of interest. In the bottom row, we highlight and provide a zoomed-in view of a region of interest. We also provide an orange arrow highlighting a discrepancy from SZ3, and a pink arrow highlighting a discrepancy of SPERR. The Z position of each point corresponds to the Frobenius norm with smoothing applied.

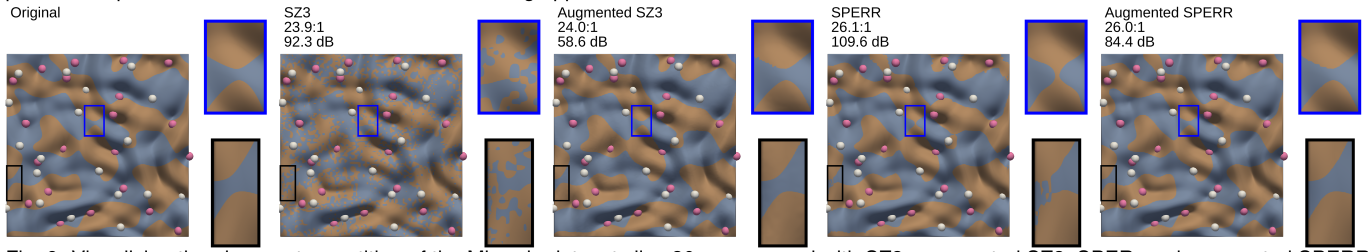

Fig. 9: Visualizing the eigenvector partition of the Miranda dataset slice 30 compressed with SZ3, augmented SZ3, SPER, and augmented SPERR, along with the ground truth. We provide zoomed-in views that highlight the differences between the compressors and the ground truth. We also label compression ratio and PSNR. We use the same colormap as Fig. 3. Z positions correspond to the Frobenius norm with smoothing applied.

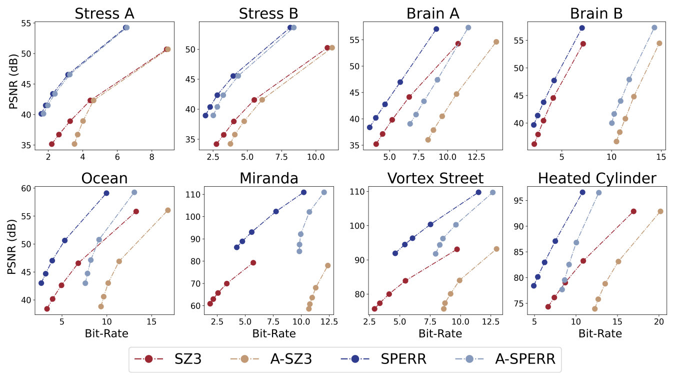

Fig. 10: Curves showing the tradeoff between bit-rate and PSNR for SZ3 and SPERR on each dataset, as well as our augmented compressors given as A-SZ3 and A-SPERR.

In Sec. 5.2, we list the number and percentage of cells topologically misclassified by SZ3 and SPERR. We chose error bounds for SZ3 and SPERR so that they achieve compression ratios similar to those from TFZ with $\xi = 0.01$ for symmetric data and $\xi = 0.001$ for asymmetric data. We provide these bounds in the supplement to ensure reproducibility. We observe that the base compressors exhibit significant topological errors when applied to asymmetric data. While the percentage (error rate) is relatively low for a symmetric dataset, the percentage of cells that contain a degenerate point is typically very small. Thus, having a relatively small percentage of incorrectly predicted cells can produce a significant effect on the topology of the decompressed data.

In Fig. 8, we visualize the Stress B and Brain B datasets before and after compression with SZ3, augmented SZ3, SPERR, and augmented SPERR. We chose parameter configurations such that the compression ratio achieved by both compressors is the same (see the supplement for details). In each visualization, we highlight a region of interest, and provide a zoomed-in view when needed. We display the Miranda dataset in the same fashion in Fig. 9, and provide zoomed-in views of the eigenvector and eigenvalue partitions. We also visualize the Ocean dataset before and after compression with SZ3 and augmented SZ3 in Fig. 1. In the figures, we can see that SZ3 and SPERR make noticeable topological errors, while TFZ perfectly preserves the topology.

Compressed file size. In Fig. 10, we visualize the tradeoff between bit-rate and PSNR for both the augmented and base compressors. Consistent with previous works on topology-preserving compression [12, 29], we observe that when ξ is high, increasing ξ can lead to lower compression ratios. When evaluating the tradeoff between bit-rate and PSNR, we use low values of ξ such that increasing ξ increases compression ratios. The values for ξ are different for each dataset and base compressor; we report them in the supplement for reproducibility.

In Fig. 10, we can see that the tradeoff between bit-rate and PSNR for the augmented compressors mirrors that of the base compressors with some amount of overhead. Overall, augmented SPERR seems to achieve better reconstruction quality than augmented SZ3.

5.3 Run Time Analysis

Compression and decompression times. TFZ processes each cell independently and limits the number of processing iterations before storing it losslessly. Let n be the number of cells and k the number of iterations, TFZ runs in linear time $O(nk)$. Recall $k = 20$.

Empirically, we measure the compression time for each dataset using both base and augmented compressors; see the results in Tab. 3. For asymmetric data, we report the compression time while preserving both eigenvector and eigenvalue partitions, while cases where only one is preserved are provided in the supplement.

Table 2: Number of cells whose topology is misclassified by SZ3 and SPERR when compressed to the same ratio as augmented SZ3 and augmented SPERR, respectively. For symmetric data, we set the error bound of TFZ to be $\xi = 0.01$ and for asymmetric data we set $\xi = 0.001$. #Cells: number of cells in the mesh. #SZ3: number of cells misclassified by SZ3. %SZ3: percent of cells misclassified by SZ3. #SPERR: number of cells misclassified by SPERR. %SPERR: percent of cells misclassified by SPERR.

Dataset	#Cells	#SZ3	%SZ3	#A-SZ3	#SPERR	%SPERR	#A-SPERR
Stress A	204,800	85	0.04%	0	47	0.02%	0
Stress B	204,800	582	0.28%	0	381	0.19%	0
Brain A	1,057,160	2,144	0.20%	0	1,660	0.16%	0
Brain B	8,723,862	16,831	0.19%	0	6,609	0.08%	0
Ocean	540,000	17,185	3.2%	0	13,536	2.5%	0
Miranda	75,104,768	16,359,113	21.8%	0	2,464,647	3.3%	0
Vortex Street	151,543,962	32,160,009	21.2%	0	21,881,208	14.4%	0
Heated Cylinder	267,604,000	50,817,728	19.0%	0	32,859,532	12.3%	0

Table 3: Compression and decompression times for each dataset using base and augmented compressors. A-SZ3 and A-SPERR denote augmented SZ3 and augmented SPERR respectively.

Dataset	SZ3	A-SZ3	SPERR	A-SPERR
Total Compression Time (s)				
Stress A	0.22	0.56	0.22	0.67
Stress B	0.22	0.57	0.22	0.66
Brain A	0.76	1.92	0.77	2.38
Brain B	2.11	9.17	2.22	10.93
Ocean	0.35	1.90	0.39	2.28
Miranda	7.00	274.14	9.53	287.16
Vortex Street	24.34	784.85	28.86	835.99
Heated Cylinder	37.98	762.21	48.23	713.11
Decompression Time (s)				
Stress A	0.28	0.25	0.28	0.25
Stress B	0.28	0.25	0.28	0.26
Brain A	0.82	0.86	0.81	0.81
Brain B	2.15	2.31	2.08	2.47
Ocean	0.34	0.38	0.36	0.42
Miranda	4.73	11.44	6.34	12.60
Vortex Street	19.78	33.76	22.81	35.13
Heated Cylinder	29.01	50.78	36.68	53.44

Table 4: Breakdown of compression times for each dataset when compressed with augmented SZ3 and augmented SPERR. Times are in seconds. All trials use an error bound of $\xi = 0.001$. BC: Base compressor. Cells: Cell correction step. LL: Lossless storage. Save: Save compressed file. Clean: Remove intermediate files.

Dataset	Setup	BC	Cells	LL	Save	Clean	Total
Augmented SZ3							
Stress A	0.01	0.20	0.13	0.03	0.06	0.13	0.56
Stress B	0.02	0.20	0.14	0.03	0.06	0.12	0.57
Brain A	0.05	0.48	0.69	0.08	0.22	0.39	1.92
Brain B	0.34	1.34	5.42	0.51	0.71	0.85	9.17
Ocean	0.13	0.32	1.13	0.06	0.09	0.18	1.90
Miranda	10.64	7.02	245.17	6.90	1.75	1.94	274.14
Vortex Street	21.05	19.86	708.43	15.73	7.99	10.39	784.85
Heated Cylinder	31.67	33.11	644.57	24.78	11.28	14.32	762.21
Augmented SPERR							
Stress A	0.02	0.30	0.13	0.03	0.06	0.13	0.67
Stress B	0.02	0.30	0.13	0.03	0.06	0.13	0.66
Brain A	0.05	0.82	0.80	0.08	0.22	0.41	2.38
Brain B	0.34	2.45	6.09	0.49	0.71	0.86	10.93
Ocean	0.08	0.57	1.27	0.07	0.09	0.18	2.28
Miranda	11.61	13.82	250.47	6.86	1.73	1.95	287.16
Vortex Street	21.27	37.22	744.16	13.64	7.78	10.54	835.99
Heated Cylinder	31.96	63.80	568.57	21.67	10.19	14.41	713.11

We find that augmented SPERR generally achieves slightly worse compression times than augmented SZ3. Varying the error bound ξ does not have a significant effect on the run times; see the supplement.

Analysis of compression times. In Tab. 4, we provide a more detailed breakdown of the run times reported in Tab. 3 for augmented SZ3 and augmented SPERR. As expected, the cell correction step (labeled as "Cells") accounts for the largest portion of the run time, typically dominating the overall processing time. For symmetric data, this step is dedicated entirely to preserving degenerate points. In contrast, for asymmetric data, it involves correcting vertices, degenerate points, and cell topology. Among these, cell topology correction is the most time-consuming, though all three contribute significantly to the total run time. We provide more detail in the supplement.

6 Limitations and Discussion

Our TFZ framework has its limitations. By enhancing a base compressor and generating additional output files, the augmented compressor typically achieves lower compression ratios and requires more time compared to the base compressor. We highlight the key takeaways here and report detailed analysis in the supplement.

Storage overhead. In Fig. 10, we can see that TFZ imposes up to $1\times$ storage overhead for symmetric data and $3\times$ for asymmetric data (at a fixed PSNR). These values are similar to other topology-preserving compressors at similar compression ratios [29, 49]. For asymmetric data, preserving only one of the eigenvector or eigenvalue partitions will lower the overhead. For a fixed compression ratio, we argue improved topological correctness can be more important than increased PSNR, depending on the application. We believe that there is room to improve the compression ratios by developing targeted strategies to preserve the internal cell topology of asymmetric tensor fields, as well as more efficient multiscale quantization schemes.

Time overhead. In Tab. 3, we observe that TFZ introduces a significant compression time overhead compared to the base compressors. TFZ typically achieves a throughput of 3–6 MB/second. We also timed TFZ compressing datasets with larger 2D slices and found similar throughputs. The results are in the supplement. Our throughputs are comparable to other topology-preserving methods [12, 29], but a direct comparison is challenging due to variations in processor speed. Also, scientific datasets are often compressed once, and the compressed file is distributed many times, so this overhead may represent a one-time cost. We report performance on a single CPU. TFZ could potentially benefit from parallel or GPU implementations. Since TFZ processes each cell independently, parallelization would be relatively straightforward.

Visual artifacts. Our strategy can produce visual artifacts, particularly in the asymmetric case. Such artifacts are most noticeable in areas where the magnitude of tensors is low compared to the absolute error bound. We discuss the causes of such artifacts in detail in the supplement. Managing these artifacts is a possible area of future work.

Other limitations. For symmetric tensor fields, TFZ makes no guarantees about the tensorlines that connect the degenerate points. The tensorlines are part of the definition of the topology of a symmetric tensor field, and one logical extension of our work is to preserve the tensorlines. However, our strategy still benefits applications that use the entire topology by preserving the degenerate points. Further, there are use cases that do not require tensorlines [11, 25, 46, 51].

7 Conclusion

We introduce a novel framework, TFZ, designed to augment any error-bounded lossy compressor in order to preserve the topology of 2D symmetric and asymmetric tensor fields. In both cases, we scan through each cell and correct the topology one cell at a time. Our experiments show that TFZ preserves the degenerate points of symmetric data and the eigenvalue and eigenvector partitions of asymmetric data, while introducing a reasonable overhead. Looking ahead, a common asymmetric tensor field is the gradient of a vector field, and the decomposition in Eq. (3) is used to visualize vector fields [3, 50]. TFZ could be applied to vector fields to preserve the topology of the gradient.

Acknowledgments

This work was supported in part by grants from National Science Foundation OAC-2313122, OAC-2313123, and OAC-2313124.

References

[1] Z. Ahsan. On the riemann curvature tensor in general relativity. *Progress of Theoretical Physics Supplement*, 172:224–227, 2008. doi: 10.1143/PTPS.172.224 1

[2] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic polygonal remeshing. *ACM Transactions on Graphics (TOG)*, 22(3):485–493, 2003. doi: 10.1145/882262.882296 1, 2

[3] C. Auer, J. Kasten, A. Kratz, E. Zhang, and I. Hotz. Automatic, tensor-guided illustrative vector field visualization. In *IEEE Pacific Visualization Symposium (PacificVis)*, pp. 265–272. IEEE, 2013. doi: 10.1109/PacificVis.2013.6596154 2, 9

[4] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. TTHRESH: Tensor compression for multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 26(9):2891–2903, 2020. doi: 10.1109/TVCG.2019.2904063 2, 7

[5] C. G. Callan Jr., S. Coleman, and R. Jackiw. A new improved energy-momentum tensor. *Annals of Physics*, 59(1):42–73, 1970. doi: 10.1016/0003-4916(70)90394-5 1

[6] G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang. Interactive procedural street modeling. *ACM Transactions on Graphics (TOG)*, 27(3):1–10, 2008. doi: 10.1145/1360612.1360702 1, 2

[7] T. Delmarcelle. *The visualization of second-order tensor fields*. PhD thesis, Stanford University, 1995. 2

[8] T. Delmarcelle and L. Hesselink. The topology of symmetric, second-order tensor fields. In *IEEE Visualization Conference (VIS)*, pp. 140–147. IEEE, 1994. doi: 10.1109/VISUAL.1994.346326 3

[9] S. Di, J. Liu, K. Zhao, X. Liang, R. Underwood, Z. Zhang, M. Shah, Y. Huang, J. Huang, X. Yu, et al. A survey on error-bounded lossy compression for scientific datasets. *ACM Computing Surveys*, 57(11):1–38, 2024. doi: 10.1145/3733104 2

[10] D. B. Ennis, G. Kindlman, I. Rodriguez, P. A. Helm, and E. R. McVeigh. Visualization of tensor fields using superquadric glyphs. *Magnetic Resonance in Medicine*, 53(1):169–176, 2005. doi: doi.org/10.1002/mrm.20318 2

[11] X. Gao. Applying 2D tensor field topology to symmetric stress tensors. Master's thesis, Oregon State University, 2018. 2, 9

[12] 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 (TVCG)*, 31(6):3693–3705, 2025. doi: 10.1109/TVCG.2025.3567054 2, 5, 8, 9

[13] H. Guo. *What Are Tensors Exactly?* World Scientific, 2021. doi: 10.1142/12388 1

[14] R. Haimes and D. Kenwright. On the velocity gradient tensor and fluid feature extraction. In *Computational Fluid Dynamics Conference*, pp. 1–10, 1999. doi: 10.2514/6.1999-3288 1

[15] C. Hergl, C. Blecha, V. Kretzschmar, F. Raith, F. Günther, M. Stommel, J. Jankowai, I. Hotz, T. Nagel, and G. Scheuermann. Visualization of tensor fields in mechanics. *Computer Graphics Forum (CGF)*, 40(6):135–161, 2021. 2

[16] L. Hesselink, Y. Levy, and Y. Lavin. The topology of symmetric, second-order 3D tensor fields. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 3(1):1–11, 1997. doi: 10.1109/2945.582332 3

[17] S.-H. Hung, Y. Zhang, H. Yeh, and E. Zhang. Feature curves and surfaces of 3D asymmetric tensor fields. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 28(1):33–42, 2022. doi: 10.1109/TVCG.2021.3114808 3

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

[19] J. Jankowai, B. Wang, and I. Hotz. Robust extraction and simplification of 2D symmetric tensor field topology. *Computer Graphics Forum (CGF)*, 38(3):337–349, 2019. doi: 10.1111/cgf.13693 1, 2, 3, 5

[20] F. Khan, L. Roy, E. Zhang, B. Qu, S.-H. Hung, H. Yeh, R. S. Laramee, and Y. Zhang. Multi-scale topological analysis of asymmetric tensor fields on surfaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 26(1):270–279, 2020. doi: 10.1109/TVCG.2019.2934314 2, 3

[21] J. Ko, T. Kwon, J. Jung, and K. Shin. ELiCiT: effective and lightweight lossy compression of tensors. In *IEEE International Conference on Data Mining (ICDM)*, pp. 171–180, 2024. doi: 10.1109/ICDM59182.2024.00024 2

[22] T. Kwon, J. Ko, J. Jung, and K. Shin. NeuKron: Constant-size lossy compression of sparse reorderable matrices and tensors. In *ACM Web Conference 2023 (WWW)*, pp. 71–81, 2023. doi: 10.1145/3543507.3583226 2

[23] T. Kwon, J. Ko, J. Jung, and K. Shin. TensorCodec: Compact lossy compression of tensors without strong data assumptions. In *IEEE International Conference on Data Mining (ICDM)*, pp. 229–238. IEEE, 2023. doi: 10.1109/ICDM58522.2023.00032 2

[24] 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: 10.1007/978-3-642-23400-2_34 2

[25] Y. Lavin, Y. Levy, and L. Hesselink. Singularities in nonuniform tensor fields. In *IEEE Visualization Conference (VIS)*, pp. 59–66. IEEE, 1997. 2, 9

[26] H. Le and J. Tao. Hierarchical autoencoder-based lossy compression for large-scale high-resolution scientific data. *Computing&AI Connect (CAIC)*, 1, 2024. doi: 10.69709/CAIC.2024.193132 2

[27] D. Le Bihan, J.-F. Mangin, C. Poupon, C. A. Clark, S. Pappata, N. Molko, and H. Chabriat. Diffusion tensor imaging: concepts and applications. *Journal of Magnetic Resonance Imaging (JMRI)*, 13(4):534–546, 2001. doi: 10.1002/jmri.1076 1

[28] S. Li, P. Lindstrom, and J. Clyne. Lossy scientific data compression with SPERR. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1007–1017. IEEE, 2023. doi: 10.1109/IPDPS54959.2023.00104 2, 7

[29] 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 (TVCG)*, 31, 2025. doi: 10.1109/TVCG.2024.3456337 2, 8, 9

[30] X. Liang, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, T. Peterka, and H. Guo. Toward feature-preserving vector field compression. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 29(12):5434–5450, 2023. doi: 10.1109/TVCG.2022.3214821 2

[31] 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 (Big Data)*, pp. 438–447. IEEE, 2018. doi: 10.1109/BigData.2018.8622520 2

[32] 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 (TBD)*, 9(2):485–498, 2023. doi: 10.1109/TBDATA.2022.3201176 2, 7

[33] Z. Lin, H. Yeh, R. S. Laramee, and E. Zhang. 2D asymmetric tensor field topology. In *Topological Methods in Data Analysis and Visualization II: Theory, Algorithms, and Applications*, pp. 191–204. Springer, 2011. doi: 10.1007/978-3-642-23175-9_13 2, 3, 4

[34] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 20(12):2674–2683, 2014. doi: 10.1109/TVCG.2014.2346458 2, 7

[35] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. *Computer Graphics Forum (CGF)*, 40(3):135–146, 2021. doi: 10.1111/cgf.14295 2, 7

[36] C. Meneveau. Lagrangian dynamics and models of the velocity gradient tensor in turbulent flows. *Annual Review of Fluid Mechanics*, 43:219–245, 2011. doi: 10.1146/annurev-fluid-122109-160708 1

[37] J. Palacios and E. Zhang. Interactive visualization of rotational symmetry fields on surfaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(7):947–955, 2011. doi: 10.1109/TVCG.2010.121 2

[38] D. Palke, Z. Lin, G. Chen, H. Yeh, P. Vincent, R. Laramee, and E. Zhang. Asymmetric tensor field visualization for surfaces. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 17(12):1979–1988, 2011. doi: 10.1109/TVCG.2011.170 2

[39] W. Pearlman, A. Islam, N. Nagaraj, and A. Said. Efficient, low-complexity image coding with a set-partitioning embedded block coder. *IEEE Transactions on Circuits and Systems for Video Technology (TVSCT)*, 14(11):1219–1235, 2004. doi: 10.1109/TCSVT.2004.835150 2

[40] L. Roy, P. Kumar, Y. Zhang, and E. Zhang. Robust and fast extraction of

3D symmetric tensor field topology. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 25(1):1102–1111, 2019. doi: 10.1109/TVCG.2018.2864768 3

[41] J. L. Schlenker, G. V. Gibbs, and M. B. Boisen Jr. Strain-tensor components expressed in terms of lattice parameters. *Acta Crystallographica Section A: Foundations and Advances*, 34(1):52–54, 1978. doi: 10.1107/S0567739478000108 1

[42] K. Shi, E. R. Smith, E. E. Santiso, and K. E. Gubbins. A perspective on the microscopic pressure (stress) tensor: History, current understanding, and future challenges. *Journal of Chemical Physics (JCP)*, 158:040901:1–040901:32, 2023. doi: 10.1063/5.0132487 1

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

[44] 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 2

[45] 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 (IPDPS)*, pp. 1129–1139. IEEE, 2017. doi: 10.1109/IPDPS.2017.115 4

[46] X. Tricoche, G. Scheuermann, and H. Hagen. Tensor topology tracking: A visualization method for time-dependent 2D symmetric tensor fields. *Computer Graphics Forum (CGF)*, 20(3):461–470, 2001. 1, 2, 9

[47] 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 *IEEE Conference on Data Engineering (ICDE)*, pp. 3682–3695. IEEE, May 2025. doi: 10.1109/ICDE65448.2025.00275 2

[48] K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-Or, and H. Huang. Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields. *ACM Transactions on Graphics (TOG)*, 36(6):1–15, 2017. doi: 10.1145/3130800.3130812 1, 2

[49] L. Yan, X. Liang, H. Guo, and B. Wang. TopoSZ: Preserving topology in error-bounded lossy compression. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 30(1):1302–1312, 2024. doi: 10.1109/TVCG.2023.3326920 2, 9

[50] E. Zhang, H. Yeh, Z. Lin, and R. S. Laramee. Asymmetric tensor analysis for flow visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 15(1):106–122, 2009. doi: 10.1109/TVCG.2008.68 2, 3, 9

[51] Y. Zhang, X. Gao, and E. Zhang. Applying 2D tensor field topology to solid mechanics simulations. In *Modeling, Analysis, and Visualization of Anisotropy*, pp. 29–41. Springer, 2017. 2, 9

[52] 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 *IEEE International Conference on Data Engineering (ICDE)*, pp. 1643–1654, 2021. doi: 10.1109/ICDE51399.2021.00145 2

[53] X. Zheng and A. Pang. HyperLIC. In *IEEE Visualization Conference (VIS)*, pp. 249–256. IEEE, 2003. 2

[54] X. Zheng and A. Pang. 2D asymmetric tensor analysis. In *IEEE Visualization Conference (VIS)*, pp. 3–10. IEEE, 2005. doi: 10.1109/VISUAL.2005.1532770 3