Toward Feature-Preserving Vector Field Compression

Xin Liang, *Member, IEEE,* Sheng Di, *Senior Member, IEEE,* Franck Cappello, *Fellow, IEEE,* Mukund Raj, *Member, IEEE,* Chunhui Liu, Kenji Ono, Zizhong Chen, *Senior Member, IEEE,* Tom Peterka, *Member, IEEE,* and Hanqi Guo, *Member, IEEE*

Abstract—The objective of this work is to develop error-bounded lossy compression methods to preserve topological features in 2D and 3D vector fields. Specifically, we explore the preservation of critical points in piecewise linear and bilinear vector fields. We define the preservation of critical points as, without any false positive, false negative, or false type in the decompressed data, (1) keeping each critical point in its original cell and (2) retaining the type of each critical point (e.g., saddle and attracting node). The key to our method is to adapt a vertex-wise error bound for each grid point and to compress input data together with the error bound field using a modified lossy compressor. Our compression algorithm can be also embarrassingly parallelized for large data handling and in situ processing. We benchmark our method by comparing it with existing lossy compressors in terms of false positive/negative/type rates, compression ratio, and various vector field visualizations with several scientific applications.

Index Terms—lossy compression, critical points, vector field visualization.

1 Introduction

Lossy compression of floating-point data has become a promising technique for data reduction, as the disparity between data generation rate and available I/O bandwidth continues to grow in today's and future supercomputers. Data generated from large-scale ocean, atmosphere, and fluid dynamics simulations can be compressed in situ, and then the decompressed data can be used both in situ and post hoc for data analysis and visualization. In order to preserve insights in the compressed data, error-bounded lossy compressors such as SZ [1], ZFP [2], TTHRESH [3], and FPZIP [4], as opposed to traditional JPEG [5] image compressors, are used to strictly guarantee the desired accuracy while acceptable compression ratio is achieved.

The motivation of this study is to preserve the accuracy of features extracted in error-bounded lossy compressed data. We examine critical points in 2D and 3D vector fields as an example of a feature. Critical points—locations where the vector field vanishes—are important because they are the key constituents of vector field topology and thus essentially determine the characteristics of flow visualizations based on geometry [6], texture [7], and topology [8], [9]. The extraction of critical points leads to both locations and types (e.g., sources, sinks, and saddles), and both properties must be preserved in order to deliver authentic insights into the decompressed data.

With today's lossy compressors, failure to preserve critical points can result in false positives (FPs), false negatives (FNs), and false types (FTs). A false positive happens if a critical point is localized in the decompressed data but such a point does not exist in the same vicinity of the original data. A false negative means that a critical point is missed in the decompressed data. A false type occurs when the critical point type does not match in the original and decompressed data. For example, an attracting critical point may turn into a repelling one during compression/decompression.

In this work, we aim at improving error-bounded lossy compressors to preserve critical points in piecewise linear 2D/3D and bilinear vector fields. We define the preservation of critical points as, without any false positive, false negative, or type change in the decompressed data, (1) keeping each critical point in its original cell and (2) retaining the type of each critical point. The key to our method is to adapt a vertex-wise error bound for each grid point and to compress input data together with the vertex-wise error bounds using SZ [1], which is a prediction-based lossy compressor. We develop both decoupled and coupled compression pipelines. The decoupled approach estimates vertex-wise error bounds for all vertices and then compresses the vector field based on the error bounds. The coupled approach estimates the error bound and compresses the data on the fly during the lossy compression. The coupled method delivers higher compression ratios but is more computationally expensive than the decoupled method. We demonstrate that our methods outperform existing lossy compressors in terms of FP, FN, and FT rates and compression ratio with several scientific applications.

Existing efforts to compress 2D vector fields based on topology [10], [11] (1) are nontrivial to generalize to 3D, (2) do not guarantee local error bounds and may lead to large distortions, (3)

- *X. Liang is with the Department of Computer Science, University of Kentucky, Lexington, KY 40506. E-mail: xliang@uky.edu*
- *S. Di, F. Cappello, and T. Peterka are with the Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, 60439. E-mail: sdi1@anl.gov, {cappello, tpeterka}@mcs.anl.gov.*
- *M. Raj is with the Stanley Center for Psychiatric Research, Broad Institute of MIT and Harvard, Cambridge, MA 02142, USA. E-mail: mraj@broadinstitute.org*
- *C. Liu is with the Department of Mathematics, Kyoto University, Kyoto, Japan. E-mail: chunhui.liu@math.kyoto-u.ac.jp*
- *K. Ono is with the Department of Informatics, Kyushu University, Fukuoka, Japan. E-mail: keno@cc.kyushu-u.ac.jp*
- *Z. Chen is with the Department of Computer Science and Engineering, University of California, Riverside, Riverside, CA 92521. E-mail: chen@cs.ucr.edu*
- *H. Guo is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, 43210. E-mail: guo.2154@osu.edu*

require undetermined iterations to converge, or (4) are difficult to parallelize. More discussion on the differences between our method and previous efforts is in the following section. The contributions of this paper can be summarized as follows:

- Theoretical framework to preserve critical points in 2D/3D piecewise linear and 2D bilinear vector fields based on the vertex-wise error bound derivation. Note that our method does not generalize to 3D trilinear vector fields due to the unavailability of closed form solutions, as will be discussed in Section 7.4.
- Two feature-preserving compressor designs that enforce vertex-wise error bounds: a decoupled method optimized for speed and a coupled method optimized for storage.

2 Background

We review the related work on error-bounded lossy compression and vector field compression and then formalize the critical point extraction problem. Notations used in this paper are in Table 1.

2.1 Error-Bounded Lossy Compression

Data compression can be either lossless or lossy, and lossy compression can be further categorized into *non-error-bounded lossy* and *error-bounded lossy* methods. This study focuses on error-bounded lossy compressors, which guarantee local error within designated error bounds. Error-bounded lossy compressors usually deliver higher compression ratios than the lossless compressors such as FPC [12], while are more precise than non-error-bounded lossy compressors such as JPEG [5]. We refer to the literature [13], [14] for comprehensive reviews of scientific data compression; hence we focus mainly on error-bounded lossy compressors.

Error-bounded lossy compression can be prediction- or transformation-based. Prediction-based error-bounded compressors include FPZIP [4] and SZ [1]. FPZIP uses a Lorenzo predictor [15] with integer mapping on both the predicted and actual data for avoiding underflow, followed by an arithmetic encoding on the prediction residuals. SZ is a multialgorithm compressor with blockwise selection on the best-fit predictor, including both Lorenzo and regression-based predictors. Different from the arithmetic encoding used in FPZIP, SZ performs linear quantization on the prediction residuals and encodes the quantization integers by customized Huffman encoding and lossless compressors such as GZIP [16] and ZSTD [17]. An example of transformation-based error-bounded compressors is ZFP [2]. ZFP first performs the exponent alignment and fixed-precision points conversion and applies a fine-tuned orthogonal/inorthogonal transformation for each block and then encodes the coefficients for compression.

While a user-given single error bound is mandatory for all existing error-bounded lossy compressors, the key difference of our study is that we derive vertex-wise error bounds. Specifically, we construct vertex-wise error bounds that adapt the numerical tolerance in order to preserve features in the decompressed data. Formally, the relative error between the exact value $d \in \mathbb{R}_{\neq 0}$ and its approximation $d' \in \mathbb{R}$ is defined as

$$\delta(d,d') \stackrel{\text{def}}{=} |d-d'|/|d|. \tag{1}$$

The relative error bound $\Delta(\cdot)$ is an arbitrary value such that no relative error exceeds the error bound.[1] We generalize the notation of δ to represent the *maximal relative error* between the input data $\mathbf{d}$ and its approximation $\mathbf{d}'$ as $\delta(\mathbf{d},\mathbf{d}') \stackrel{\text{def}}{=} \max_i \delta(d_i,d_i')$, where $\mathbf{d}$ and $\mathbf{d}'$ are vectors of arbitrary dimensions in $\mathbb{R}$ and i is the linear index of each element in $\mathbf{d}$ and $\mathbf{d}'$. We also use the notation $\Delta(\mathbf{d})$ to represent the relative error bound of each element in $\mathbf{d}$, and we use $||\Delta(\mathbf{d})||$ as the maximal element of $\Delta(\mathbf{d})$.

Although most error-bounded compressors provide guranteed point-wise error control on the raw data, little has been done to generalize such error control to specific user requirements. MGARD [18], [19], [20] offers error control on the outcome of bounded-linear analysis, but the error analysis relied heavily on the linear assumption thus cannot be generalized easily. Underwood et al. [21] proposed an iterative approach to meet certain requirements (e.g., fixed ratios) from users, but it suffered from high computational overhead due to the iterative process and low efficiency because of the unified error bound. To the best of our knowledge, no existing error-bounded lossy compressors can preserve topological information such as the critical points in piecewise linear and bilinear vector fields.

2.2 Vector Field Compression

Vector field compression has been studied to preserve 2D topological features. Lodha et al. [11] used an iterative clustering method [22] to simplify and compress 2D vector fields. The iteration stops when all critical points remain identical to the original topology and the designated local error bound is met. Compared with that method, our method is noniterative, has fixed time complexity, and works for both 2D and 3D vector fields in regular and unstructured meshes. Theisel et al. [10] iteratively collapsed edges in the 2D mesh in order to guarantee topology preservation, but without local error control. Dey et al. [23] proposed a Delaunay simplification for the vector field based on error-bounded edge collapsing, but the simplification did not explicitly preserves topological features. Koch et al. [24] presented a segmentation-based compression approach based on region-wise linear approximation; a simplified mesh grid can be generated by iteratively adding new segmentations and testing topological equivalence.

To the best of our knowledge, this study is the first attempt to tailor error-bounded lossy compressors to preserve features in 2D and 3D vector field data. The following are the key differences between our method and the state of the art. First, existing error-bounded lossy compressors are not aware of important vector field features, whereas our method preserves critical points. By compressing individual vector components, existing compressors produce FP, FN, and FT critical points in the decompressed data. Second, existing topology-based vector field compression does not bound local error, whereas our method enforces local error bounds. For example, the iterative edge collapsing approach [10] does not control local error, which may lead to large distortions in the decompressed data. Third, existing vector field compression algorithms are challenging to generalize to 3D, whereas our technique applies to both 2D and 3D vector fields. Because 3D vector field topology is much more complicated than 2D, the generalization of clustering- [11], mesh simplification- [10], [23], and segmentation- [24] based algorithms can be convoluted and computationally expensive. Fourth, most existing vector field compressors [10], [11], [23], [24] are iterative, whereas our method compresses a vector field in a single pass with fixed-time complexity. In addition, our method can be easily parallelized

1. In this work, we limit relative error bounds to $[0,1]$, and in this case, d' and d always have the same sign.

TABLE 1
Nomenclature.

Symbol	Domain	Meaning
n_d	2 or 3	Dimensionality of vector field
m,n,i,j,k	$\mathbb{N}$	Integer numbers and indices
$\mathbf{v}(\cdot)$, $\mathbf{v}'(\cdot)$	$\mathbb{R}^{n_d} \to \mathbb{R}^{n_d}$	Input and decompressed vector fields
$u(\cdot)$, $v(\cdot)$, $[w(\cdot)]$	$\mathbb{R}^{n_d} \to \mathbb{R}$	Components of $\mathbf{v}(\cdot)$
$\mathbf{J}(\cdot)$	$\mathbb{R}^{n_d} \to \mathbb{R}^{n_d \times n_d}$	Jacobian of $\mathbf{v}(\cdot)$
$\mathbf{v}_i = (u_i, v_i, [w_i])^\intercal$	$\mathbb{N} \to \mathbb{R}^{n_d}$	Input vector values at ith vertex
$\mathbf{v}'_i = (u'_i, v'_i, [w'_i])^\intercal$	$\mathbb{N} \to \mathbb{R}^{n_d}$	Decompressed vector values at ith vertex
$\mathbf{V}_j = (\mathbf{v}_{(j,0)}, \mathbf{v}_{(j,1)}, \mathbf{v}_{(j,2)}, [\mathbf{v}_{(j,3)}])$	$\mathbb{R}^{n_d \times (n_d+1)}$	All vector values of jth piecewise linear cell
$\mathbf{J}_j$	$\mathbb{N} \to \mathbb{R}^{n_d \times n_d}$	Jacobian of $\mathbf{v}$ in jth cell
$\mathbf{x}_i = (x_i, y_i, [z_i])^\intercal$	$\mathbb{R}^{n_d}$	Coordinates of ith vertex
$\boldsymbol{\mu} = (\mu_0, \mu_1, \mu_2, [\mu_3])^\intercal$	$\mathbb{R}^{n_d+1}$	Barycentric coordinates of critical points
$(m_0, m_1, m_2, [m_3])^\intercal$	$\mathbb{R}^{n_d+1}$	Auxiliary barycentric coordinates; Equations (10) and (11)
$\delta(\cdot,\cdot')$	$\mathbb{R}^{n \times n} \to [0,1)$	Relative error; Equation (1)
$\Delta(\cdot)$	$\mathbb{R}^n \to [0,1)^n$	Relative error bounds of each element
$\psi(f(\cdot);\cdot)$ or $\psi(f(\cdot))$	$\mathbb{R}^n \to [0,1)$	Sign-preserving error function (SPEF) of scalar function f; Equation (5)
η_j	$[0,1)$	Relative error bound for all vertices associated with jth cell
ξ_i, $\widehat{\xi}_i$	$[0,1)$	Relative error bound and quantized relative error bound for ith vertex; $\widehat{\xi}_i \le \xi_i$
$(\lambda_0, \lambda_1, [\lambda_2])^\intercal$	$\mathbb{C}^{n_d}$	Eigenvalues of Jacobian matrices
$\mathrm{Re}(\cdot), \mathrm{Im}(\cdot)$	$\mathbb{C} \to \mathbb{R}$	Real and imaginary part denotation
$\mathrm{sgn}(\cdot)$	$\mathbb{R} \to \{-1, 0, 1\}$	Sign function
$R^+(\cdot), R^-(\cdot)$	$\mathbb{R} \to \mathbb{R}$	Positive and negative ramp functions; $R^+(\cdot) = \max(0,\cdot)$ and $R^-(\cdot) = \min(0,\cdot)$

and thus can (de)compress data for high-performance data storage, analysis, and visualization.

2.3 Critical Point Extraction

A critical point is defined as the location where the vector field vanishes. Formally, let n_d be the dimensionality of the data and the vector field be $\mathbf{v} : \mathbb{R}^{n_d} \to \mathbb{R}^{n_d}$; the vector value $\mathbf{v}$ at a critical point $\mathbf{x}_c \in \mathbb{R}^{n_d}$ must be $\mathbf{0}$. In this study, we focus on non-degenerate (or first-order) critical points, where the determinant of the vector gradient tensor (Jacobian) of the vector field $|\mathbf{J}_\mathbf{v}(\mathbf{x}_c)| \neq 0$. The vector field $\mathbf{v}$ is defined on a simplicial (triangular or tetrahedral) mesh; the interpolation scheme for each cell is piecewise linear or bilinear. Without loss of generality, we use 2D cases for illustration and derivation; the same techniques apply to 3D unless otherwise noted. Figure 1 depicts an example of a critical point in 2D simplicial and square cells.

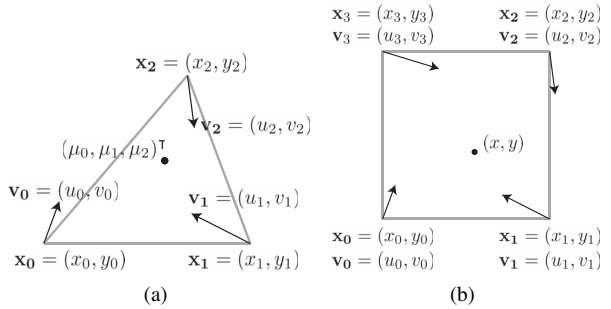


Fig. 1. Critical point in (a) a linearly-interpolated triangular cell and (b) a bilinearly interpolated rectangular cell.

Critical points in piecewise linear vector fields. The extraction of critical points involves finding zero points in each linearly interpolated cell:

$$\mathbf{V} \cdot \boldsymbol{\mu} = \begin{bmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \mathbf{0} \text{ and } \mu_0 + \mu_1 + \mu_2 = 1, \quad (2)$$

where $\boldsymbol{\mu}$ is defined as the (normalized) barycentric coordinates of the critical point and $\mathbf{V}$ is a 2×3 matrix consisting of all vector values for the vertices. If the condition $0 \le \mu_k \le 1$ holds for all $k \in \{0,1,2\}$, the critical point resides inside of the cell; otherwise the cell contains no critical point.

Critical points in piecewise bilinear vector fields may be extracted in an exact analytical manner. Bilinear interpolation gives a smooth representation of vector fields within the cells; each cell contains four vector values, as illustrated in Figure 1(b). The bilinear interpolation of the vector components u and v is $u(x,y) = (1-x)(1-y)u_0 + x(1-y)u_1 + xyu_2 + (1-x)yu_3$ and $v(x,y) = (1-x)(1-y)v_0 + x(1-y)v_1 + xyv_2 + (1-x)yv_3$, which then boils down into the following form

$$\begin{aligned} u(x,y) &= A_u xy + B_u x + C_u y + D_u, \\ v(x,y) &= A_v xy + B_v x + C_v y + D_v, \end{aligned} \quad (3)$$

where x and y are 2D coordinates; coefficients $A_u, B_u, \cdots, D_v$, which are constants for each cell, are functions of vector values on the corners ($\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$, and $\mathbf{v}_3$). With each cell, a critical point (x,y) inside the quadrilateral exist if $u(x,y) = v(x,y) = 0$. The equation $u(x,y) = v(x,y) = 0$ can be reformulated as

$$\begin{bmatrix} B_u & D_u \\ B_v & D_v \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix} = -y \begin{bmatrix} A_u & C_u \\ A_v & C_v \end{bmatrix} \begin{bmatrix} x \\ 1 \end{bmatrix}, \quad (4)$$

which is a *generalized eigenvalue problem*[2]. The equation has up to two roots that can be derived in closed form[3]; after that, a point-in-quadrilateral test further confirms if the critical point is inside the cell.

Type of critical points. Critical points can be categorized into various types based on the signs of eigenvalues of the Jacobian $\mathbf{J}_\mathbf{v}(\mathbf{x}_c)$ [25], [26]. In general, negative and positive eigenvalues

2. In general, generalized eigenvalue problems are defined in the form of $\mathbf{Ax} = \lambda\mathbf{Bx}$, where $\mathbf{A}$ and $\mathbf{B}$ are given $n \times n$ matrices; λ and $\mathbf{x}$, respectively, represent the eigenvalue and eigenvector. Assume that $\mathbf{B}$ is nonsingular, the problem may be converted into a classical eigenvalue problem $\mathbf{B}^{-1}\mathbf{Ax} = \lambda\mathbf{x}$.

3. Note that 2×2 eigenvalue problems have closed-form solutions. Specific to Eq.(4), for simplicity, we use $\mathbf{Q}$ to denote the product $-\begin{bmatrix} A_u & C_u \\ A_v & C_v \end{bmatrix}^{-1} \begin{bmatrix} B_u & D_u \\ B_v & D_v \end{bmatrix}$ in this paper, assuming the left matrix is nonsingular. In case the left matrix is singular, we use lossless compression in the corresponding cell, as further discussed in the rest of this paper.

indicate attracting and repelling, respectively; eigenvalues with imaginary parts imply circulation behavior. In 2D piecewise linear fields, the critical point can be determined analytically based on $\mathbf{J}$, because the *characteristic polynomial* $|\lambda\mathbf{I}-\mathbf{J}| = \lambda^2 - \mathrm{tr}(\mathbf{J})\lambda + |\mathbf{J}|$ is quadratic and thus has a closed-form solution, where $\mathbf{I}$ is the identity matrix and $\mathrm{tr}(\cdot)$ is the trace of a matrix. In 3D piecewise linear and bilinear fields, the characteristic polynomial can be determined in closed form as well.

3 Overview

In this work, we propose a novel approach to compress data while preserving critical points in piecewise linear and bilinear vector fields. Our key idea is to adapt a vertex-wise error bound for each grid point based on how critical points are extracted, and to compress data with the adapted error bounds using a modified error-bounded lossy compressor. This enables feature preservation with local error control, thus is fundamentally different from existing error-bounded lossy compressors which are feature-agnostic and provide only a uniform global error bound over all points.

We structure the rest of the paper as follows to present the proposed approach. In Section 4, we first formulate the research problem, where we define the requirements for feature preservation in piecewise linear and bilinear vector fields and rigorously derive the generic formulas of sufficient error bounds to meet such requirements. These mathematical formulations lay a solid theorectical foundation for the proposed methods. We then present the decoupled and coupled feature-perserving compression schemes in Section 5, along with how the actual error bounds are computed in these schemes for different vector fields based on the derived formulas. As a comparison, the decoupled scheme features lower computational cost and lighter dependancy across vertices compared with the coupled scheme, while the coupled scheme delivers higher compression ratios under the same requirements with higher overhead. After that, we demonstrate how one existing lossy compressor is customized to support our schemes with optimizations toward efficient storage of the derived error bounds in Section 6. At last, we present the evaluation results in Section 7 and conclude with a vision for future work in Section 8. Details about the supported combinations of schemes and dimensions in this paper are listed in Table 2.

TABLE 2
Combinations of Schemes and Dimensions for Different Vector Fields

	Schemes		Dimensions	
	Decoupled	Coupled	2D	3D
Piece-wise linear	✓	✓	✓	✓
Piece-wise bilinear		✓	✓	

4 Mathematical Formulation

This section introduces the mathematical formulation before outlining the distinct approaches to achieve feature-preserving compression. Specifically, we first formulate the requirements for feature-preserving compression, followed by the sufficient conditions on vertex-wise error bounds to meet the requirements.

We formulate the feature-preserving compression problem for the non-degenerative critical point extraction in piecewise linear vector fields and bilinear vector fields as follows. In general, the original and decompressed 2D/3D vector fields $\mathbf{v}$ and $\mathbf{v}'$ respectively are defined on the same simplicial (triangular or tetrahedral) or Cartesian mesh; the vector values on the ith vertex are $\mathbf{v}_i$ and $\mathbf{v}'_i$, respectively; and the relative error bound of $\mathbf{v}_i$ is denoted as ξ_i. For simplicity, we introduce our problem using piecewise linear vector fields as an example; our objectives/definitions can be generalized to bilinear vector fields in a similar way.

For each cell in the mesh, assuming the barycentric coordinates of the critical point in the original and decompressed data are $(\mu_0,\mu_1,\mu_2,[\mu_3])^\intercal$ and $(\mu'_0,\mu'_1,\mu'_2,[\mu'_3])^\intercal$, respectively, and the Jacobian eigenvalues are $(\lambda_0,\lambda_1,[\lambda_2])^\intercal$ and $(\lambda'_0,\lambda'_1,[\lambda'_2])^\intercal$, respectively, the objective of this study is to guarantee the following three conditions by finding proper relative error bounds ξ_i:

- **Non-FN**: If $\mu_k \in [0,1]$ holds for all k, $\mu'_k \in [0,1]$ holds for all k as well;
- **Non-FP**: If there exists k such that $\mu_k \notin [0,1]$, there exists k' such that $\mu_{k'} \notin [0,1]$;
- **Non-FT**: The "non-FN" condition is met, and there exists a one-to-one mapping between l and l' such that $\mathrm{sgn}(\mathrm{Re}(\lambda_l)) = \mathrm{sgn}(\mathrm{Re}(\lambda_{l'}))$ and $\mathrm{sgn}(\mathrm{Im}(\lambda_l)) = \mathrm{sgn}(\mathrm{Im}(\lambda'_{l'}))$.

Here, $\mathrm{sgn}(\cdot)$ is the sign function; $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ are the real and imaginary part operators, respectively; and $k,k' \in \{0,1,\ldots,n_d\}$, $l,l' \in \{0,1,\ldots,n_d-1\}$. For bilinear case, similar conditions need to hold for x, y, and λ_1,λ_2.

To derive sufficient error bounds for these conditions, we introduce the *sign-preserving error function* (SPEF) of any given scalar function. Formally, denoting the ball $B(\mathbf{d},\gamma) = \{\mathbf{d}' \mid \delta(\mathbf{d},\mathbf{d}') \le \gamma\}, \gamma \in [0,1)$, we define the SPEF of any given scalar function $f:\mathbb{R}^{m\times n} \to \mathbb{R}$ as

$$\psi(f;\mathbf{d}) \stackrel{\text{def}}{=} \sup\{\gamma \mid f(\mathbf{d})f(\mathbf{d}') \ge 0, \forall \mathbf{d}' \in B(\mathbf{d},\gamma)\}. \tag{5}$$

As illustrated in Figure 2, $\psi(f;\mathbf{d}) \in [0,1]$ is defined as the supremum of $\delta(\mathbf{d},\mathbf{d}')$ such that $f(\mathbf{d})$ and $f(\mathbf{d}')$ keep the same sign. In general, finding the closed form of ψ functions is challenging. Thus we instead attempt to find a relaxed error bound $\Delta(\mathbf{d})$ that is less than or equal to $\psi(f;\mathbf{d})$, in order to preserve the sign of $f(\mathbf{d}')$. In the following, we present the possible relaxed error bounds for preserving critical points in piecewise linear and bilinear vector fields. Before diving into the details, we first show the following properties of SPEFs which will be repeatedly used in our derivation:

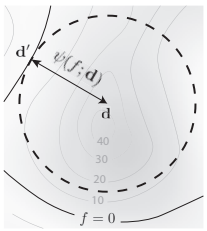


Fig. 2. Illustration of SPEF function ψ.

$$\psi(fg;\mathbf{d}) \ge \min(\psi(f;\mathbf{d}),\psi(g;\mathbf{d})), \tag{6}$$

$$\psi(f+g;\mathbf{d}) \ge \min(\psi(f;\mathbf{d}),\psi(g;\mathbf{d})), \text{if } \mathrm{sgn}(f(\mathbf{d})) = \mathrm{sgn}(g(\mathbf{d})), \tag{7}$$

where f and g are two given scalar function $\mathbb{R}^{m\times n} \to \mathbb{R}$ that take the same arguments. We further generalize the notation of ψ for multiple functions $f_0,f_1,\ldots,f_{n-1}$ to represent the maximal error bound to keep the sign of each function, and we have

$$\psi(f_0,f_1,\ldots,f_{n-1};\mathbf{d}) \stackrel{\text{def}}{=} \min_i \psi(f_i;\mathbf{d}). \tag{8}$$

4.1 Critical Point Preservation in Piecewise Linear Vector Fields

We derive the sufficient error bounds for 2D/3D piecewise linear vector fields in this subsection. The conclusions are summarized

in Equation (10), (11), and (12) with the proofs presented in Proclaims 1, 2, and 3.

For simplicity, we first introduce *auxiliary barycentric coordinates* $(m_0, m_1, m_2)^\intercal$ to represent the 2D critical point solution in Equation (2), such that in non-degeneracy cases, we have

$$\mu_k = m_k/(m_0+m_1+m_2) = m_k/M, k \in \{0,1,2\}, \tag{9}$$

where $M = \sum_k m_k \neq 0$, $m_0 = \left|\begin{smallmatrix} u_1 & u_2 \\ v_1 & v_2 \end{smallmatrix}\right|$, $m_1 = \left|\begin{smallmatrix} u_0 & u_2 \\ v_0 & v_2 \end{smallmatrix}\right|$, and $m_2 = \left|\begin{smallmatrix} u_0 & u_1 \\ v_0 & v_1 \end{smallmatrix}\right|$; We use similar notations to represent 3D critical point solutions with auxiliary barycentric coordinates. Based on the definitions of the auxilary barycentrice coordinates and SPEFs, a sufficient condition for non-FN, non-FP, and non-FT in 2D piecewise linear vector fields is

$$||\Delta(\mathbf{V})|| \leq \psi(m_0, m_1, m_2; \mathbf{V}), \tag{10}$$

$$||\Delta(\mathbf{V})|| \leq \max_{k \in \{k | \mu_k \notin [0,1]\}} \min\left(\psi(m_k), \psi(\sum_{k' \neq k} m_{k'}; \mathbf{V})\right), \text{ and} \tag{11}$$

$$||\Delta(\mathbf{V})|| \leq \begin{cases} \psi(|\mathbf{J}|; \mathbf{V}) & \text{if} \quad |\mathbf{J}| \leq 0 \\ \psi(\text{tr}^2(\mathbf{J}) - 4|\mathbf{J}|, \text{tr}(\mathbf{J}); \mathbf{V}) & \text{if} \quad |\mathbf{J}| > 0 \end{cases}, \tag{12}$$

respectively, where $\mathbf{V}$ is the 2x3 matrix defined in Equation (2); m_0, m_1, and m_2 are *auxilary barycentrice coordinates*; and $k \in \{0,1,2\}$. Equations (10) and (11) can be directly generalized to 3D cases, and we use $||\Delta(\mathbf{V})|| = 0$ to guarantee the 3D non-FT condition. In what follows, we prove Equations (10), (11), and (12) by proving the three proclaims below, respectively.

Proclaim 1. *A sufficient condition to avoid FN in a triangular cell is* $||\Delta(\mathbf{V})||_{\max} \leq \psi(m_0, m_1, m_2; \mathbf{V})$.

Proof. The "non-FN" condition is equivalent to finding a proper maximal error bound $||\Delta(\mathbf{V})||_{\max}$ such that

$$||\Delta(\mathbf{V})||_{\max} \leq \psi(\mu_0, \mu_1, \mu_2, 1-\mu_0, 1-\mu_1, 1-\mu_2; \mathbf{V}) \stackrel{\text{def}}{=} \phi_{\text{FN}}(\mathbf{V}). \tag{13}$$

Based on Equations (9) and (6), we have

$$\begin{aligned} \phi_{\text{FN}}(\mathbf{V}) &\stackrel{(9)}{=} \psi\left(\frac{m_0}{M}, \frac{m_1}{M}, \frac{m_2}{M}, \frac{m_1+m_2}{M}, \frac{m_0+m_2}{M}, \frac{m_0+m_1}{M}; \mathbf{V}\right) \\ &\stackrel{(6)}{=} \psi(m_0, m_1, m_2, m_1+m_2, m_0+m_2, m_0+m_1, M; \mathbf{V}) \\ &\stackrel{(7)}{\geq} \psi(m_0, m_1, m_2; \mathbf{V}). \end{aligned} \tag{14}$$

The last inequality holds because $\text{sgn}(m_0) = \text{sgn}(m_1) = \text{sgn}(m_2)$, which is deduced from $\mu_k = m_k/M \geq 0$ for all k. □

Proclaim 2. *A sufficient condition to avoid FP in a triangular cell is* $||\Delta(\mathbf{V})||_{\max} \leq \max_{k \in \{k | \mu_k \notin [0,1]\}} \min\big(\psi(m_k), \psi(\sum_{k' \neq k} m_{k'}; \mathbf{V})\big)$.

Proof. In the "non-FP" condition, there exists at least one k such that $\mu_k \notin [0,1]$. Thus, a sufficient condition to avoid FP is to adapt $||\Delta(\mathbf{V})||_{\max}$ satisfying $\mu_k' \notin [0,1]$ in the decompressed data for any $k \in \{k \mid \mu_k \notin [0,1]\}$ such that

$$||\Delta(\mathbf{V})||_{\max} \leq \max_{k \in \{k | \mu_k \notin [0,1]\}} \psi(\mu_k(1-\mu_k); \mathbf{V}). \tag{15}$$

For $\psi(\mu_k(1-\mu_k); \mathbf{V})$, we have

$$\begin{aligned} \psi(\mu_k(1-\mu_k); \mathbf{V}) &= \psi\left(\frac{m_k}{M} \cdot \frac{M-m_k}{M}; \mathbf{V}\right) = \psi\left(\frac{m_k(\sum_{k' \neq k} m_{k'})}{M^2}; \mathbf{V}\right) \\ &\stackrel{(6)}{\geq} \min\big(\psi(m_k), \psi(\textstyle\sum_{k' \neq k} m_{k'}); \mathbf{V}\big), \end{aligned} \tag{16}$$

and thus the proclaim is proved. □

Proclaim 3. *A sufficient condition to avoid FT of noncenter critical points in a triangular cell is*[4]

$$||\Delta(\mathbf{V})||_{\max} \leq \begin{cases} \psi(|\mathbf{J}|; \mathbf{V}) & \textit{if} \quad |\mathbf{J}| \leq 0 \\ \psi(\text{tr}^2(\mathbf{J}) - 4|\mathbf{J}|, \text{tr}(\mathbf{J}); \mathbf{V}) & \textit{if} \quad |\mathbf{J}| > 0 \end{cases}. \tag{17}$$

Proof. Let $B = -\text{tr}(\mathbf{J})$ and $C = |\mathbf{J}|$ for simplicity. The root of $\lambda^2 + B\lambda + C$ are $\lambda_0, \lambda_1 = (-B \pm \sqrt{B^2 - 4C})/2$. Consider the case of $C < 0$. The discriminant will be larger than 0; thus the roots are both real. Furthermore, the roots have to be one positive and one negative for $|B| < \sqrt{B^2 - 4C}$. Therefore, we need only to preserve the negativity of $\lambda_0\lambda_1$:

$$\psi(\lambda_0\lambda_1; \mathbf{V}) = \psi\big(B^2 - (B^2 - 4C); \mathbf{V}\big) = \psi(C; \mathbf{V}). \tag{18}$$

However, when $C > 0$, the sign of discriminant $B^2 - 4C$ has to be preserved because it determines the number of real roots. When the discriminant is greater than 0, we also need to preserve B only because $\text{sgn}(-B + \sqrt{B^2 - 4C}) = \text{sgn}(-B - \sqrt{B^2 - 4C}) = \text{sgn}(-B)$. The proclaim is thus proved. □

4.2 Critical Point Preservation in Bilinear Vector Fields

We derive the sufficient error bounds for 2D bilinear vector fields in this subsection. The conclusions are summarized in Equation (19) and (20) with the proofs presented in Proclaims 4 and 5.

In bilinear vector fields, the 2D coordinates (x, y) of the critical points are derived one after another based on the generalized eigen solving problem. Thus, it is natural to preserve the coordinates one by one. Specifically, we derive sufficient conditions for y instead of (x, y); nevertheless, the same approach can be applied to x to prevent FN and FP of x. Note that we avoid corner cases by setting $||\Delta(\mathbf{V})|| = 0$ when the determinant or trace of the matrix in the generalized eigen problem equals 0. Denoting $p(\lambda) = \lambda^2 - \text{tr}(\mathbf{Q})\lambda + |\mathbf{Q}|$ and $q = \text{tr}(\mathbf{Q})/2$, a sufficient condition for non-FN and non-FP in terms of y is:

$$||\Delta(\mathbf{V})|| \leq \begin{cases} 0 & \text{if} \quad p(q) = 0, q \in [0,1) \\ \psi(p(0), p(1); \mathbf{V}) & \text{if} \quad p(0)p(1) < 0 \\ \psi(p(q), p(0), p(1), q, q-1; \mathbf{V}) & \\ \qquad \text{if} \quad p(q) < 0, p(0) > 0, p(1) > 0, q \in [0,1) & \end{cases} \tag{19}$$

$$||\Delta(\mathbf{V})|| \leq \begin{cases} \psi(p(q); \mathbf{V}) & \text{if} \quad p(q) > 0 \\ \min(\psi(p(0), p(1)), \max(\psi(q), \psi(q-1)); \mathbf{V}) & \\ \qquad \text{if} \quad p(q) \leq 0, p(0) > 0, p(1) > 0, q \notin [0,1) & \\ \psi(p(0), p(1); \mathbf{V}) \quad \text{if} \quad p(q) \leq 0, p(0) < 0, p(1) < 0 & \end{cases} \tag{20}$$

Note that $\mathbf{V}$ is the 2x4 matrix representing points in a square cell. Also note that we only preserve the FN and FP of coordinate y in the critical points. Unlike the positions, types of critical points in bilinear vector fields are hard to preserve using SPEFs, because Jacobian in this case involve functions of x and y which have unknown perturbations on their own values. Nevertheless, we notice that preserving FN and FP of y automatically preserve x and the type in most cases. Thus, we perform compression of current data point using the error bound that preserves the positions of y and verify whether the decompressed data affect the positions of

4. The preservation of a center ($\text{Re}(\lambda_0) = \text{Re}(\lambda_1) = 0$ and $\text{Im}(\lambda_0), \text{Im}(\lambda_1) \neq 0$) requires $\text{tr}(\mathbf{J}) = 0$, which leads to the error bound of 0. However, centers are seldom found because the floating-point representation of $\text{tr}(\mathbf{J})$ is normally nonzero.

x and the types of critical points. In particular, the decompressed value will be recovered and used to check whether it incurs FN/FP on x or changes the type in all of its adjacent cells. If both x and types can be preserved, we will move to the next data point; otherwise the current data point will be re-compressed with error bound 0. As such, this approach is only applicable to the coupled compression framework, which will be introduced in details in the next section.

We then present the proofs of Equations (19) and (20) with the two proclaims below.

Proclaim 4. *A sufficient condition to avoid FN of coordinate y in a critical point (x,y) for a bilinear square cell is in Equation* (19).

Proof. A valid y needs to fall in one of the three cases below: 1) there is one double root in [0, 1); 2) there is one and only one root in [0, 1); 3) there are two roots in [0, 1). Case 1) happens when $p(q) = 0$ so an error bound 0 is required. Case 2) only requires $p(0)p(1) < 0$ so preserving signs of $p(0)$ and $p(1)$ will satisfy. Case 3) is the most complex, which requires four conditions to be met at the same time. As $q \in [0,1)$ is equavalent to the preservation of signs in both q and $q-1$, five signs need to be preserved in this case. □

Proclaim 5. *A sufficient condition to avoid FP of coordinate y in a critical point (x,y) for a bilinear square cell is in Equation* (20).

Proof. According to Equation (4), whether y is a valid solution depends on the number of roots in the equation $p(\lambda) = 0$ and their locations. Thus, one of the following conditions needs to be met when y is not valid: the equation $p(\lambda) = 0$ has 1) no real root; or 2) more than one real roots but no real roots in [0, 1). Falling into case 1) is equivalent to $p(q) > 0$, thus avoiding FP turns out to preserve the sign of $p(q)$, which yields the first condition. Two subcases exist for case 2) when $p(q) \leq 0$, depending on the sign of $p(0)$. If $p(0) > 0$, y is not valid if and only if $p(1) > 0$ and $q \notin [0,1)$. Avoiding FP reduces to preserve signs of $p(0)$, $p(1)$, and q or $q-1$. If $p(0) < 0$, having no roots in [0, 1) only requires $p(1) < 0$, so we only needs to preserve the sign of $p(0)$ and $p(1)$ to keep y invalid. Note that the sign of $p(q)$ may not need to be preserved for both the two subcases because there will not be roots if the sign of $p(q)$ changes. □

Although the above proclaims only preserve FN and FP for y, they can be applied to preserve x by rewritting the generalized eigen problem in Equation (4) using x as the eigen value. Setting $||\Delta(\mathbf{V})||$ to the minimum of two derived error bounds in terms of x and y will yield non-FN and non-FP for both x and y. However, as we have to perform a verification to check the preservation of types, we only preserve FN and FP for either x or y to save the computational cost.

5 Decoupled and Coupled Feature-Preserving Compression Schemes

This section presents both decoupled and coupled approaches to compress the vector field while preserving critical points. The decoupled method compresses data in a pipelined manner (Figure 3 and Section 5.1), while the coupled method compresses data in a single pass (Figure 4 and Section 5.2). Comparison between the two methods is in Section 5.3. We use n_v and n_c to represent the number of vertices and cells, respectively, in the descriptions. We only develop the coupled approach for 2D bilinear vector fields because of the complexity of error derivation in the decoupled scheme and trilinear fields. Without loss of generality, we present all the compression algorithms using piecewise linear vector fields as an example. The extension to bilinear vector fields is straightforward thus omitted. Nevertheless, the error bound derivation for bilinear vector fields is different and will be detailed in Section 5.2.

Algorithm 1 Error bound computation for critical point preservation for a cell in either decoupled or coupled compression

Input: values $\mathbf{V} = \begin{pmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{pmatrix}$ and coordinates $\mathbf{X} = \begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \end{pmatrix}$ of vertices
Output: maximal error bound $e_r \in [0,1)$ for $\mathbf{V}$

```
function EB_{DECOUPLED|COUPLED}(V, X)
    A ← [u0 u1 u2; v0 v1 v2; 1 1 1], b ← (0, 0, 1)^T
    if |A| = 0 then                                  ▷ check if the system is singular
        return 0;                                    ▷ use lossless compression for this cell
    else
        if μ = A^{-1}b ∈ [0,1]^3 then                ▷ check if critical point exists
            return min(eb_FN_{decoupled|coupled}(V),
                       eb_FT_{decoupled|coupled}(V, X))   ▷ Sections 5.1.2, 5.2.2
        else
            return eb_FP_{decoupled|coupled}(V)      ▷ Sections 5.1.2, 5.2.2
        end if
    end if
end function
```

5.1 Decoupled Compression

We first present the overview of the compression scheme, followed by the derivation of error bounds in this scheme.

5.1.1 Decoupled Feature-preserving Compression Scheme

Figure 3 illustrates the pipeline of our decoupled compression scheme, which consists of cell-wise error bound computation, vertex-wise error bound aggregation, and vertex-wise compression.

Cell-wise error bound computation The pseudocode for cell-wise error bound computation is in Algorithm 1. The algorithm takes in the values and coordinates of the vertices in the given cell and returns a sufficient error bound to keep the critical point information in the cell. If the underlying linear system is deficient, we use zero as the error bound for the cell, and the vector values will be compressed losslessly in the compression stage. Otherwise, we check whether the critical point exists in the cell. If the critical point exists, we return a sufficient error bound to avoid FN and FT; otherwise, we return a sufficient error bound to avoid FP. Notice that the procedure for error bound computation is similar in the coupled compression method, and the mathematical derivation of sufficient error bounds is detailed in Section 5.1.2.

Vertex-wise error bound computation We calculate vertex-wise error bounds based on cell-wise error bounds that are computed in the previous step. As shown in Algorithm 2, we iterate over each cell and compute the cell-wise error bound η with Algorithm 1. For each vertex of the cell, we assign the minimum of the current error bound and η as the updated error bound for the vertex.

Vertex-wise compression We use the vertex-wise error bound to guide the error-bounded lossy compression; the pseudocode is in Algorithm 3. The `quant()` function in the pseudocode takes the derived error bound ξ_i as input and returns the quantized value $\widehat{\xi}_i$, in order to reduce the storage of vertex-wise error bounds in the compression. The details on the quantization are discussed in Section 6.2. In the algorithm, we first initialize a byte buffer to stage quantized values of the vector field data. We then iterate each vertex by quantizing the error bound and compressing the data

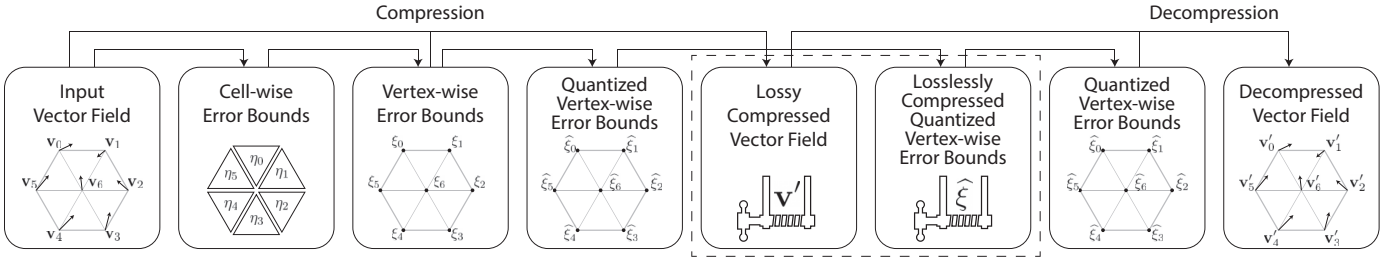


Fig. 3. Decoupled compression pipeline for critical-point-preserving compression.

Algorithm 2 Vertex-wise error bound computation based on cell-wise error bounds in decoupled compression.

Input: values $\{\mathbf{v}_i\}$ and coordinates $\{\mathbf{x}_i\}$ of all vertices
Output: vertex-wise error bounds $\{\xi_i\}$

```
function DECOUPLED_ERROR_BOUND_MAP_DERIVATION({v_i}, {x_i})
    {ξ_i} ← {1}                                  ▷ initialize each error bounds with 1
    for j ← 0 to n_c − 1 do                      ▷ iterate cells
        {i_0, i_1, i_2} ← cell_vertices(j)
        η ← eb_decoupled((v_{i_0}, v_{i_1}, v_{i_2}), (x_{i_0}, x_{i_1}, x_{i_2}))   ▷ see Alg. 1
        for i ∈ {i_0, i_1, i_2} do
            ξ_i ← min(ξ_i, η)                    ▷ vertex-wise error based on cell-wise error
        end for
    end for
    return {ξ_i}
end function
```

Algorithm 3 Decoupled feature-preserving compression

Input: values $\{\mathbf{v}_i\}$, coordinates $\{\mathbf{x}_i\}$, and error bounds $\{\xi_i\}$ for all vertices
Output: compressed byte stream

```
buffer={∅}                                       ▷ buffer for vector field compression
for i ← 0 to n_v − 1 do
    ξ̂_i ← quant(ξ_i)                             ▷ quantize error bound for vertex i
    bytes ← lossy_compress(v_i, ξ̂_i)             ▷ compress v_i lossily while
guarantee the error bound ξ̂_i
    buffer.append(bytes)
end for
return compress_losslessly(buffer, {ξ̂_i})
```

values. The `lossy_compress()` function lossily compresses the vector data by guaranteeing the given error bound and returns the compressed data in bytes. In our implementation, we use the prediction and quantization scheme in SZ for compression; more details are in Section 6. In the last step, the byte buffer and the quantized vertex-wise error bounds are losslessly encoded and compressed.

5.1.2 Error Bound Derivation in the Decoupled Scheme

This section derives the error bounds used in decoupled compression scheme, namely the functions `eb_FN_decoupled`, `eb_FP_decoupled`, and `eb_FT_decoupled` used in Algorithm 1. We use the following lemma for the error bound derivation in decoupled compression. The *positive ramp* $R^+(\cdot)$ and *negative ramp* $R^-(\cdot)$ used in the lemmas are defined as

$$\left(R^+(\mathbf{d})\right)_i = \max\{0, d_i\} \text{ and } \left(R^-(\mathbf{d})\right)_i = \min\{0, d_i\}, \tag{21}$$

respectively, for vector $\mathbf{d}$ of arbitrary dimensions, where i is the linear index of the elements. Note that Equation (22) is the key formula that is used to perform the derivation using Equations (10), (11), and (12) in this scheme.

Lemma 1. *For* $\mathbf{d} = \{d_{ij}\} \in \mathbb{R}^{m\times n}$, *if* $\sum_i \prod_j d_{ij} \neq 0$, *we have*

$$\psi\left(\textstyle\sum_i \prod_j d_{ij}; \mathbf{d}\right) \geq \left| \frac{\sqrt[n]{\sum_i R^+\left(\prod_j d_{ij}\right)} - \sqrt[n]{-\sum_i R^-\left(\prod_j d_{ij}\right)}}{\sqrt[n]{\sum_i R^+\left(\prod_j d_{ij}\right)} + \sqrt[n]{-\sum_i R^-\left(\prod_j d_{ij}\right)}} \right|. \tag{22}$$

Proof. Consider the case of $\sum_i \prod_j d_{ij} \geq 0$. Assuming the adopted cell-wise error bound is e_r, we have

$$\begin{aligned} &\textstyle\sum_i \prod_j d'_{ij} = \sum_i R^+\left(\prod_j d'_{ij}\right) + \sum_i R^-\left(\prod_j d'_{ij}\right) \\ \geq &(1-e_r)^n \textstyle\sum_i R^+\left(\prod_j d_{ij}\right) + (1+e_r)^n \sum_i R^-\left(\prod_j d_{ij}\right). \end{aligned} \tag{23}$$

Let the second line of Equation (23) be greater than or equal to 0. The problem turns out to be solving

$$(1-e_r)^n \textstyle\sum_i R^+\left(\prod_j d_{ij}\right) + (1+e_r)^n \sum_i R^-\left(\prod_j d_{ij}\right) \geq 0. \tag{24}$$

By elementary calculation, we obtain e_r in the form of Equation (22) in this case. The case of $\sum_i \prod_j d_{ij} \leq 0$ can be proved similarly. □

We then can derive the sufficient error bounds for all the SPEFs in Proclaims 1, 2, and 3, because all the functions in the SPEFs can be written in the form of $\mathbf{d} = \{d_{ij}\} \in \mathbb{R}^{m\times n}$. Notice that Proclaim 3 cannot be generalized to 3D because of the prohibitive complexity and limited benefit; we use $||\Delta(\mathbf{V})||_{\max} = 0$ as the sufficient condition to guarantee "non-FT". The reasons are detailed as follows.

The problem of preserving the type of critical points in 3D piecewise linear vector fields boils down to solving a 6 degree multivariate polynomial under certain circumstances. Taking the case when the characteristic polynomial has one real root and two complex roots for example. For simplicity, we let $B = -\text{tr}(\mathbf{J})$, $C = \frac{1}{2}(\text{tr}^2(\mathbf{J}) - \text{tr}(\mathbf{J}^2))$, and $D = \det(\mathbf{J})$ and reduce the polynomial to the decompressed form $\lambda^3 - p\lambda = q$, where $p = \frac{3C-B^2}{3}$, $q = \frac{2B^3-9BC+27D}{27}$. The real root of the polynomial is $\lambda_0 = \sqrt[3]{-\frac{q}{2} + \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}}$. Theoretically speaking, the maximal/minimal of λ'_0 with respect to the error bound can be estimated because it is a multi-variate polynomial function with each variable defined in $[-e_r, e_r]$ where e_r is the error bound. However, finding the maximal/minimal of $\frac{q^2}{4} + \frac{p^3}{27}$ requires computing the maximal/minimal of a 12-variable polynomial function up to 6 degrees, which is hard to solve. The problem is mitigated in the coupled approach, but we still need to solve a 3-variable polynomial function up to 6 degrees. Therefore, we directly use 0 as a sufficient error bound (i.e., lossless compression) for cells with critical points in 3D data.

5. Approximated upperbound of compression ratio for 3D piecewise linear vector fields by loosing the requirements to preserve FP and FN only.

We further study the impact of using derived error bound and lossless compression for cells with critical points for both 2D and 3D piecewise linear vector fields. The impact of deriving a bound for type preservation is shown in Table 3, where we estimated the compression ratios of the derivation-based 3D type preservation by loosening the error bound of cells with critical points to preserve FP/FN only. We find that critical points in the 3D dataset are more sparse than those in the 2D dataset, so lossless compression on those cells do not make a big difference to the overall compression ratio. From this table, we can see that the increase of compression ratio is almost 10% for the 2D type preservation while 0.2% for the 3D one. As such, we carefully derive the error bound preservation for 2D datasets and use an error bound of 0 for 3D or higher dimensional cases.

TABLE 3
Benefit of non-FT Derivation for Piecewise Linear Fields

Dataset	Method	CR
Ocean	lossless	10.93
	derivation	11.58
Nek5000	lossless	7.48
	derivation	7.49[5]

5.2 Coupled Compression

The coupled compression scheme couples error bound estimation and compression on the fly in the iteration of each vertex. The key to the coupled compression is to incorporate decompressed values, which are available during the process, in order to obtain more relaxed error bounds than that of decoupled compression. Similar to the previous section, we first present the compression scheme and then the derivations of error bounds under this scheme.

5.2.1 Coupled Feature-preserving Compression Scheme

As detailed in Algorithm 4, for each vertex i, we first compute the vertex-wise error bound $\xi_i^{(j)}$ based on each of its adjacent cells j. We then aggregate the vertex-wise error bound to ξ_i, quantize ξ_i to $\widehat{\xi}_i$, and perform lossy compression on vector field data $\mathbf{v}_i$. The output bytes are appended to a preallocated buffer for further compression. The `quant()` and `lossy_compress()` functions are the same as those in the decoupled compression. Note that the coupled compression needs to use the `decode()` function to calculate the decompressed data $\mathbf{v}'_i$ on the fly and to overwrite the original $\mathbf{v}_i$.

Algorithm 4 Coupled feature-preserving lossy compression

Input: values $\{\mathbf{v_i}\}$ and coordinates $\{\mathbf{x_i}\}$ of all vertices
Output: compressed byte stream

```
buffer={∅}                                   ▷ integer buffer for compression
for i ← 0 to n_v − 1 do                      ▷ iterate vertices
    for j ∈ vertex_cells(i) do               ▷ iterate cells connected to vertex i
        {i_0, i_1, i_2} ← cell_vertices(j)   ▷ vertices of cell j
        ξ_i^(j) ← eb_coupled((v_i0, v_i1, v_i2), (x_i0, x_i1, x_i2))   ▷ see Alg. 1
    end for
    ξ_i ← min_j ξ_i^(j)                      ▷ aggregate error bound for vertex i
    ξ̂_i ← quant(ξ_i)                         ▷ quantize error bound of vertex i
    bytes ← lossy_compress(v_i, ξ̂_i)         ▷ quantize vector values with SZ
    v'_i ← decode(bytes, ξ̂_i)                ▷ calculate decompressed value v'_i on-the-fly
    v_i ← v'_i                               ▷ replace the input value with the decompressed value
    buffer.append(d)
end for
return compress_losslessly(buffer, {ξ̂_i})
```

Algorithm 4 can be proved by mathematical induction. We would like to show that in the ith iteration, if the dataset $\{\mathbf{v}'_0,\ldots,\mathbf{v}'_{i-1},\mathbf{v}_i,\ldots,\mathbf{v}_{n_v-1}\}$ preserves all critical points, $\{\mathbf{v}'_0,\ldots,\mathbf{v}'_{i-1},\mathbf{v}'_i,\ldots,\mathbf{v}_{n_v-1}\}$ preserves all critical points as well. Actually, we can obtain an error bound such that all adjacent cells of i preserve critical points; critical points in nonadjacent cells remain unchanged. The base condition ($i=0$) holds as well because no decompressed data are available. The correctness of the algorithm is thus proved.

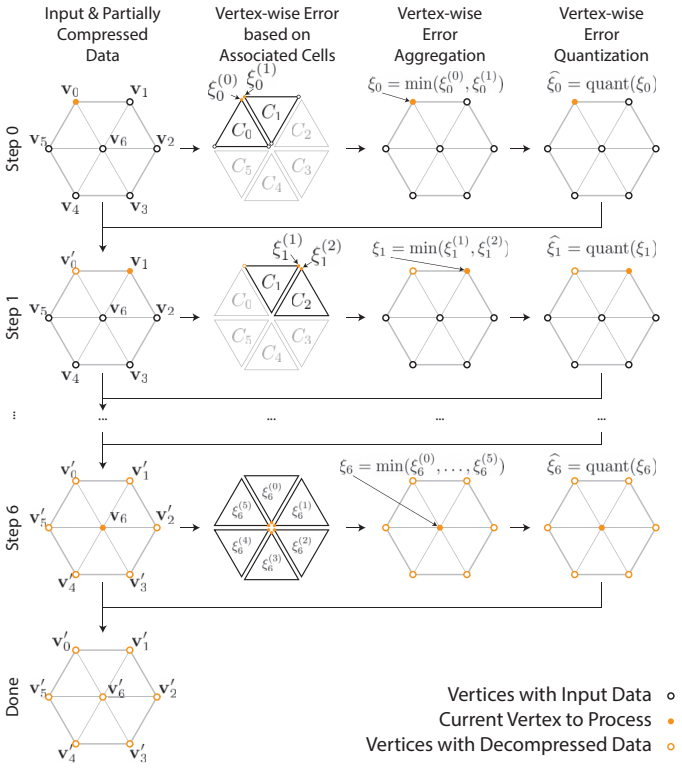


Fig. 4. Illustration of coupled compression algorithm.

Figure 4 illustrates the coupled compression on a tiny example ($n_v=7$ and $n_c=6$). At step 0, the error bound of vertex 0 is obtained per adjacent cell (C_0 and C_1). Then the error bound ξ_0 can be calculated by aggregation. The decompressed vector value is then calculated on the fly as $\mathbf{v}'_0$, which will be used in the error bound computation of $\mathbf{v}_1$, $\mathbf{v}_5$, and $\mathbf{v}_6$ in the next few iterations. Notice that we must use $\mathbf{v}'_0$ instead of $\mathbf{v}_0$ for the error bound derivation; otherwise, the use of the original value $\mathbf{v}_0$ violates the proof above.

5.2.2 Error Bound Derivation in the Coupled Scheme

We present the derivation in details for `eb_FN_coupled`, `eb_FP_coupled`, and `eb_FT_coupled` used in Algorithm 1 for coupled compression in this section. We first introduce three lemmas that are widely used in the derivation, followed by how they are applied to derive the sufficient error bounds. Note that Equation (25) is the key formula that is used to perform the derivation using Equations (10), (11), (12), (19) and (20) in this scheme.

Lemma 2. *Let* $\mathbf{a}\in\mathbb{R}^n$, $\mathbf{b}\in\mathbb{R}^n$, *and* $c\in\mathbb{R}$, *if* $R^+(\mathbf{a})^\intercal\mathbf{b}-R^-(\mathbf{a})^\intercal\mathbf{b}\neq 0$. *Then we have*

$$\psi(\mathbf{a}^\intercal\mathbf{b}+c;\mathbf{b}) \geq \frac{|\mathbf{a}^\intercal\mathbf{b}+c|}{R^+(\mathbf{a})^\intercal\mathbf{b}-R^-(\mathbf{a})^\intercal\mathbf{b}}. \tag{25}$$

Proof. First, we consider the case of $\mathbf{a}^\intercal\mathbf{b}+c>0$. The decompressed data $\mathbf{a}^\intercal\mathbf{b}'+c$ can be scaled as follows by assuming the

adopted cell-wise error bound is e_r. Then we have

$$\begin{aligned}\mathbf{a}^\intercal\mathbf{b}' + c &= (R^+(\mathbf{a}) + R^-(\mathbf{a}))^\intercal\mathbf{b}' + c \\ &\geq R^+(\mathbf{a})^\intercal\mathbf{b}(1-e_r) + R^-(\mathbf{a})^\intercal\mathbf{b}(1+e_r) + c. \end{aligned} \tag{26}$$

Let the second line of Equation (26) be larger than or equal to 0. Then we have

$$e_r \leq \frac{R^+(\mathbf{a})^\intercal\mathbf{b} + R^-(\mathbf{a})^\intercal\mathbf{b} + c}{R^+(\mathbf{a})^\intercal\mathbf{b} - R^-(\mathbf{a})^\intercal\mathbf{b}} = \frac{\mathbf{a}^\intercal\mathbf{b} + c}{R^+(\mathbf{a})^\intercal\mathbf{b} - R^-(\mathbf{a})^\intercal\mathbf{b}}. \tag{27}$$

Second, the case for $\mathbf{a}^\intercal\mathbf{b} + c < 0$ can be proved similarly. This completes the proof. □

Lemma 3. *Let $f(\varepsilon) = w_0\varepsilon^2 + w_1\varepsilon + w_2$ where $w_0, w_1, w_2 \in \mathbb{R}$. The maximal e_r ($e_r \in [0,1]$) that keeps the sign of $f(\varepsilon)$ in $[0, e_r]$ can be solved in closed form.*

Proof. We only show the solutions when $w_0 > 0$ and $w_2 > 0$; the solutions for other cases can be derived in a similar way but omitted here for simplicity. Denote $\Delta_f = w_1^2 - 4w_0w_2$, $x_1 = \frac{-w_1 - \sqrt{\Delta_f}}{2w_0}$, and $x_2 = \frac{-w_1 + \sqrt{\Delta_f}}{2w_0}$. In this case, if $\Delta_f < 0$, $f(\varepsilon) > 0$ will always hold so the maximal e_r will be 1. If $\Delta_f \geq 0$, $f(\varepsilon) = 0$ will have either negative roots or positive roots. If the roots are negative, $f(\varepsilon) > 0$ will always hold and again it is safe to have $e_r = 1$. However, if the roots are positive, $f(\varepsilon)$ will be less than 0 when $x_1 < \varepsilon < x_2$. Thus, we have

$$e_r = \begin{cases} 1 & \text{if} \quad \Delta_f < 0 \\ 1 & \text{if} \quad \Delta_f \geq 0, x_1 < 0 \\ x_1 & \text{if} \quad \Delta_f \geq 0, x_1 > 0 \end{cases}. \tag{28}$$

□

Lemma 4. *Let $u \in \mathbb{R}$, $v \in \mathbb{R}$, and $f(u,v) = (u,v,1)\mathbf{W}(u,v,1)^\intercal$ where $W \in \mathbb{R}^{3\times3}$ is an upper triangular coefficient matrix. Then $\psi(f(u,v);u,v)$ can be solved in closed form.*

Proof. Let $\varepsilon_u, \varepsilon_v \in [-e_r, e_r]$ indicate the introduced errors for u, v and $f'(\varepsilon_u, \varepsilon_v; e_r)$ be the transformed form of $f(u,v)$ by letting $u = u(1+\varepsilon_u)$ and $v = v(1+\varepsilon_v)$. Because $f(u,v)$ is a quadratic function of u and v, $f'(\varepsilon_u, \varepsilon_v; e_r)$ will be a quadratic function of ε_u and ε_v with $f'(0,0;e_r) = f(u,v)$. Then we have

$$\psi(f(u,v);u,v) \leq \begin{cases} 0 & \text{if } f'(0,0;e_r) = 0 \\ \max_{e_r}(\min f'(\varepsilon_u, \varepsilon_v; e_r) > 0) & \text{if } f'(0,0;e_r) > 0 \\ \max_{e_r}(\max f'(\varepsilon_u, \varepsilon_v; e_r) < 0) & \text{if } f'(0,0;e_r) < 0 \end{cases}. \tag{29}$$

As $f'(\varepsilon_u, \varepsilon_v; e_r)$ is defined on a bounded interval $[-e_r, e_r] \times [-e_r, e_r]$, its extremum is either on the only critical point or the boundaries of the interval. For example, if the extremum is located on one of the corners, we need to find the maximal e_r such that $f'(-e_r, -e_r; e_r)$, $f'(e_r, -e_r; e_r)$, $f'(-e_r, e_r; e_r)$, and $f'(e_r, e_r; e_r)$ have the same sign as $f'(0,0;e_r)$ at the same time. Note that $f'(-e_r, -e_r; e_r)$, $f'(e_r, -e_r; e_r)$, $f'(-e_r, e_r; e_r)$, and $f'(e_r, e_r; e_r)$ all reduce to some univariate quadratic functions in the form of $w_0e_r^2 + w_1e_r + w_2$, the closed form for e_r can be solved by Lemma 3. Similar derivations apply when the extremum falls onto the edges, where an additional relaxation is used to reduce the target function to a univariate form. When the target extremum is located in the critical point, it will be safe to have $e_r = 1$ if the extremum has the same sign as $f'(0,0;e_r)$; otherwise we should decrease the maximal bound of e_r to ensure the critical point is excluded from the interval so the extremum will be located in the boundaries again. As $\psi(f(u,v);u,v)$ have closed form in all the cases, this completes the proof.

□

With the above lemmas, we derive the error bounds for piecewise linear vector fields and bilinear vector fields as follows.

Derivation for Piecewise Linear Vector Fields: According to the sufficient condition to avoid FNs and FPs in coupled compresion (i.e., `eb_FN_coupled` and `eb_FP_coupled`) for piecewise linear vector fields in Proclaims 1 and 2, the actual error bounds can be calculated using Lemma 2.

Next, we explain how to derive the sufficient error bounds to avoid FTs for 2D piecewise linear vector field. The 2D piecewise constant Jacobian can be formulated as

$$\mathbf{J} = \widehat{\mathbf{X}}^{-1}\widehat{\mathbf{V}} = \begin{bmatrix} x_0 - x_2 & x_1 - x_2 \\ y_0 - y_2 & y_1 - y_2 \end{bmatrix}^{-1} \begin{bmatrix} u_0 - u_2 & u_1 - u_2 \\ v_0 - v_2 & v_1 - v_2 \end{bmatrix}. \tag{30}$$

Without loss of generality, we assume $\mathbf{v}_0 = (u_0, v_0)$ is the "current" value that is being processed in Algorithm 4 and u_1, v_1, u_2, v_2 are constants. By elementary calculation, $|\mathbf{J}|$ can be written as an affine function of $\mathbf{v}_0 = (u_0, v_0)$:

$$|\mathbf{J}|(u_0, v_0) = \alpha_0u_0 + \alpha_1v_0 + \alpha_2, \tag{31}$$

where α_0, α_1, and α_2 are constants that can be derived from u_1, v_1, u_2, v_2, and $\widehat{\mathbf{X}}$. Likewise, the trace of $\mathbf{J}$ can be written as another affine function of $\mathbf{v}_0 = (u_0, v_0)$ with derived constants β_0, β_1, and β_2:

$$\text{tr}(\mathbf{J})(u_0, v_0) = \beta_0u_0 + \beta_1v_0 + \beta_2. \tag{32}$$

Therefore, the sufficient error bounds for $\psi(\text{tr}(\mathbf{J}); \mathbf{v}_0)$ and $\psi(|\mathbf{J}|; \mathbf{v}_0)$ can be directly derived by using Lemma 2.

As $\text{tr}(\mathbf{J})$ is a linear function and $|\mathbf{J}|$ is a quadratic function of $\mathbf{v}_0$, the discriminant $\text{tr}^2(\mathbf{J}) - 4|\mathbf{J}|$ can be written as a quadratic function of $\mathbf{v}_0$. Thus the sufficient error bound $\psi(\text{tr}^2(\mathbf{J}) - 4|\mathbf{J}|; \mathbf{v}_0)$ can be solved using Lemma 4.

Given the sufficient error bound for $\psi(\text{tr}(\mathbf{J}); \mathbf{v}_0)$, $\psi(|\mathbf{J}|; \mathbf{v}_0)$, and $\psi(\text{tr}^2(\mathbf{J}) - 4|\mathbf{J}|; \mathbf{v}_0)$, we obtain the `eb_FT_coupled` value required in Algorithm 4 based on Proclaim 3.

Derivation for 2D Bilinear Vector Fields: Without loss of generality, we assume $\mathbf{v}_2 = (u_2, v_2)$ which locates in the right upper corner of the square in Figure 1(b) is the "current" point that is being processed and $u_0, v_0, u_1, v_1, u_3, v_3$ are constants; otherwise we can always perform a rotation to move the current point to the right upper corner, which does not affect the presence/absence of critical points. In this case, the bilinear representation $u(x,y) = A_uxy + B_ux + C_uy + D_u$ and $v(x,y) = A_vxy + B_vx + C_vy + D_v$ can be solved explicitly: A_u and A_v will be a linear function of $\mathbf{v}_2$ while $B_u, B_v, C_u, C_v, D_u, D_v$ are all constants. Thus, each entry in the matrix $\mathbf{Q}$ will be a linear fractional transformation of $\mathbf{v}_2$ with the same denominator. Accordingly, $\text{tr}(\mathbf{Q})$ will be a linear fractional transformation and $|\mathbf{Q}|$ will be a quadratic transformation of $\mathbf{v}_2$. Let $g_i(\mathbf{v}_2)$ be linear functions and $h_i(\mathbf{v}_2)$ be quadratic functions of $\mathbf{v}_2$, the expressions for $p(0)$, $p(1)$, $p(q)$, q, and $q-1$ can be represented as follows:

$$p(0) = |\mathbf{Q}| = \frac{h_0(\mathbf{v}_2)}{g_0(\mathbf{v}_2)^2}, p(1) = 1 - \text{tr}(\mathbf{Q}) + |\mathbf{Q}| = \frac{h_1(\mathbf{v}_2)}{g_0(\mathbf{v}_2)^2}$$
$$p(q) = -\frac{\text{tr}^2(\mathbf{Q})}{4} + |Q| = \frac{h_2(\mathbf{v}_2)}{g_0(\mathbf{v}_2)^2}, q = \frac{g_1(\mathbf{v}_2)}{g_0(\mathbf{v}_2)}, q-1 = \frac{g_2(\mathbf{v}_2)}{g_0(\mathbf{v}_2)}$$

Then, the SPEF for these variables will be $\psi(p(0)) = \psi(h_0(\mathbf{v}_2))$, $\psi(p(1)) = \psi(h_1(\mathbf{v}_2))$, $\psi(p(q)) = \psi(h_2(\mathbf{v}_2))$, $\psi(q) = \psi(g_0(\mathbf{v}_2), g_1(\mathbf{v}_2))$, and $\psi(q-1) = \psi(g_0(\mathbf{v}_2), g_2(\mathbf{v}_2))$. As

$\psi(g_i(\mathbf{v}_2))$ can be solved using Lemma 2 and $\psi(h_i(\mathbf{v}_2))$ can be solved using Lemma 4, the sufficient error bounds for non-FP and non-FN in terms of the y coordinates of the critical points can be solved in closed form.

5.3 Comparison between Decoupled and Coupled Compression

We compare our decoupled and coupled compression with regard to both complexity and compression ratio.

Space and time complexities The space complexities of both methods are identical ($O(n_v)$); the time complexity of the decoupled and coupled algorithm is $O(n_c)$ and $O(\sum_i \text{card}(\text{adj\_cells}(i)))$, respectively, where $\text{card}(\cdot)$ is the number of elements in a set. The $O(n_c)$ complexity of the decoupled compression algorithm is based on the cell-wise iteration in Algorithm 2. The complexity of the coupled compression algorithm is $O(n_v \cdot (\sum_i \text{card}(\text{adj\_cells}(i))/n_v))$, where the fraction is the average number of adjacent cells for each vertex. This expression can be reduced to $O(\sum_i \text{card}(\text{adj\_cells}(i)))$.

Error bounds and compression ratio Decoupled compression has a lower compression ratio than coupled compression does, because the coupled method delivers more relaxed error bounds than the decoupled method does. The reason is that the decoupled method attempts to control errors on three vertices simultaneously, while the coupled method achieves the same goal by controlling the error on one single vertex.

6 Lossy Compressor Customization

We tailor the SZ lossy compressor to guarantee vertex-wise error bounds, and we optimize the storage of the vertex-wise bounds to achieve high compression ratios.

6.1 Baseline Compressor Selection

In general, any prediction-based compressor such as FPZIP and SZ can be customized for feature-preserving compression; we use SZ as an example in this study. Transform-based compressors may be used if the error can be bounded for individual vertices.

We review two important SZ features that are used in this study. First, SZ strictly guarantees the error bound. SZ uses linear-scale quantization [27] on the difference between the original data and predicted value from a Lorenzo predictor for a strict absolute error bound guarantee. Second, SZ uses a logarithmic transformation in its recent design [28] to transform a relative error compression problem in the original domain to an absolute error compression problem in the logarithmic domain. Specifically, SZ records the signs of the original data and transforms the original data to the logarithm of their absolute value. The transformed data there are compressed with the absolute error bound $\log(1+e_r)$ where e_r is the relative error bound. During the decompression, the data are transformed back to the original data by the exponential function and the corresponding signs.

6.2 Efficient Vertex-wise Error Bounds Storage

We incorporate the following three optimizations to SZ and use these optimized functions in Algorithms 3 and 4.

Logarithm-based error bound transform We extend the logarithmic transformation for relative error bound compression in SZ to the vector field compression, in order to store one transformed error bound instead of n_d absolute error bounds for different components. Note that we have to store the original data when the error bound is 0 or the logarithmic data need to be losslessly recorded, because the round-off error in the logarithmic and exponential function may lead to unexpected perturbations in the decompressed data.

Exponential-scale error bound quantization We use quantization to further compress the transformed relative vertex-wise error bounds. Instead of adopting the default linear-scale quantization in SZ, we use an exponential-scale quantization for more aggressive size reduction. Specifically, we quantize the each error bound e_r larger than ε_0 (e_r less than ε_0 is quantized to 0) to an integer $q = \lfloor \log_b \frac{e_r}{\varepsilon_0} \rfloor$, where b and ε_0 are two tuning parameters, and we apply Huffman encoding to the quantized integers. Unlike the linear-scale quantization, which quantizes e_r to $\lfloor \frac{e_r}{\varepsilon_0} \rfloor$, exponential-scale quantization has fewer values for the quantized integers, leading to higher compression ratios on the vertex-wise error bounds. In our experiments, we set ε_0 to machine precision and b to 2, which leads to satisfactory performance.

Global error bound restriction We also use a global error bound as a strict restriction, such that any derived error bound greater than the threshold will be set to the threshold. The reason is that the precision of the Lorenzo predictor relies heavily on the precision of decompressed data, especially when the error bound of the current data is small compared with its neighbors. Limiting the error bound mitigates such problems and also decreases the range of the quantization index, reducing the size of the vertex-wise error bounds. However, this could reduce compression ratio, and the threshold needs to be tuned to achieve the best trade-off. In our implementation, we empirically set the global error bound to 0.1 for 1D and 2D data and to 0.05 for 3D data.

7 Evaluation

In this section, we evaluate our work using the three scientific datasets listed in Table 4, and we compare the results with three state-of-the-art error-bounded lossy compressors—FPZIP [4], SZ [1], and ZFP [2]—using different error bound configurations. We show both quantitative results, involving the exact number of FPs, FNs, and FTs reported by the critical point detection algorithm, and qualitative results, displaying the consequent visual difference in the local area of the changed critical points.

We use three datasets from ocean simulation, Nek5000 fluid simulation, and large eddy simulation (LES). Ocean and Nek5000 data are available in 2D and 3D regular grids, respectively, and we tessellate each 2D/3D cube into two triangles or six tetrahedra to construct piecewise linear vector fields. The LES dataset includes 71M tetrahedral cells and 110M wedges with 67.8M nodes. We consider only the tetrahedral cells for our test. All the experiments are conducted on an Intel Broadwell node with two Intel Xeon E5-2695 v4 processors and 128 GB of memory.

TABLE 4
Datasets for benchmarking

Dataset	Size	n_d	n_v	n_c
Ocean	65.92 MB	2	3600×2400	$3599 \times 2399 \times 2$
Nek5000	1.536 GB	3	512^3	$511^3 \times 6$
LES	145.8 MB	3	12.74 M	71.19 M

7.1 Results with 2D Ocean Data

We first compare the number of FPs, FNs, and FTs by tuning all the lossy compressors to a similar compression ratio ($\sim 11.5\times$ with piecewise linear cells and $\sim 4.7\times$ with bilinear cells), as shown in Table 5. From this table, we can see that our approaches

TABLE 5
Benchmark of (lossy) compressors on 2D ocean data: e_a, e_r are the global absolute and relative error bound used by compressors, respectively; CR_u, CR_v, and CR_{all} are the compression ratio (input size over output size) of $\mathbf{u}$, $\mathbf{v}$, and all components, respectively; S_c and S_d are the speed for compression and decompression, respectively; #TP is the number of true-positive (preserved) critical points; #FP, #FN, and #FT are the number of false critical points.

Piecewise Linear Cells												
Compressor	Setting	e_a	e_r	CR_u	CR_v	CR_{all}	S_c (MB/s)	S_d (MB/s)	#TP	#FP	#FN	#FT
Our method	decoupled	-	-	-	-	7.54×	32.28	77.15	20,929	0	0	0
Our method	coupled	-	-	-	-	11.73×	27.43	60.81	20,929	0	0	0
FPZIP	-P 13	-	0.0625	11.48×	11.00×	11.23×	122.23	102.86	20,416	310	244	269
SZ	-A 0.05	0.05	-	11.19×	11.50×	11.35×	131.07	214.97	18,350	43,880	1,913	666
SZ	-P 0.07	-	0.07	11.34×	11.06×	11.20×	90.53	149.33	19,680	630	601	648
ZFP	-A 0.5	0.5	-	10.06×	10.73×	10.39×	223.51	366.85	17,816	46,364	2,455	658
ZFP	-P 10	-	0.125	11.11×	11.18×	11.14×	228.04	359.04	18,685	49,207	1,593	651
Bilinear Cells												
Compressor	Setting	e_a	e_r	CR_u	CR_v	CR_{all}	S_c (MB/s)	S_d (MB/s)	#TP	#FP	#FN	#FT
Our method	coupled	-	-	-	-	4.69×	13.51	47.83	20,345	0	0	0
FPZIP	-P 19	-	2^{-10}	4.53×	4.43×	4.48×	85.94	85.27	20,343	3	1	1
SZ	-A 1E-3	0.001	-	4.63×	4.61×	4.62×	110.41	121.91	20,298	35	32	15
SZ	-P 7E-4	-	0.0007	4.73×	4.50×	4.61×	78.00	131.04	20,336	1	1	8
ZFP	-A 1E-2	0.01	-	4.48×	4.62×	4.55×	194.76	208.565	20,302	34	31	12
ZFP	-P 16	-	2^{-9}	4.73×	4.74×	4.74×	205.81	218.16	20,319	10	16	10

can be free of FPs, FNs, and FTs at relatively high compression ratio, successfully preserving all the features, whereas existing general-purpose error-bounded lossy compressors have more or less altered critical points in their decompressed data. Because preserving critical points in bilinear vector fields has more complex constraints, our method on bilinear vector fields yields lower compression ratios compared with that on linear vector fields. We also study the performance of both compression and decompression for all the compressors. Our methods are slower than existing compressors in terms of compression performance because of the higher time complexity—$n_c \approx 2n_v$ for the decoupled approach and $\sum_i \text{card}(\text{adj\_cells}(i)) \approx 6n_v$ for the coupled approach.

We also compare the compression ratios of different compressors by tuning compressors to be free of FPs, FNs, and FTs. To do so, we manually tune the error bound settings and detect critical points until no FPs, FNs, and FTs are present. The results are displayed in Table 6. Under such circumstances, existing compressors have to perform near lossless compression with compression ratios less than 3 on the piecewise linear fields. In contrast, our decoupled and coupled approaches can lead to compression ratios of 7.54× and 11.73×, respectively, while automatically preserving critical points without manual intervention. Similarly, the compression ratio of our coupled approach on the bilinear fields is 1.25× that of the best existing approaches, when all the critical points need to be preserved.

TABLE 6
Compression ratio for lossy compressors to avoid FP/FN/FT in 2D ocean data.

Piecewise Linear Cells						
Compressor	Settings	e_a	e_r	CR_u	CR_v	CR_{all}
GZIP	-1	0	0	1.58×	1.58×	1.58×
FPZIP	-P 25	-	2^{-17}	2.86×	2.82×	2.84×
SZ	-A 1E-10	1E-10	-	1.60×	1.59×	1.59×
SZ	-P 1E-5	-	1E-5	2.73×	2.63×	2.68×
ZFP	lossless	0	0	1.90×	1.88×	1.89×
Our method	decoupled	-	-	-	-	7.54×
Our method	coupled	-	-	-	-	11.73×
Bilinear Cells						
Compressor	Settings	e_a	e_r	CR_u	CR_v	CR_{all}
GZIP	-1	0	0	1.58×	1.58×	1.58×
FPZIP	-P 21	-	2^{-13}	3.79×	3.72×	3.75×
SZ	-A 2E-5	2E-5	-	2.76×	2.78×	2.77×
SZ	-P 9E-5	-	9E-5	3.57×	3.44×	3.51×
ZFP	-A 1E-4	1E-4	-	2.68×	2.73×	2.70×
Our method	coupled	-	-	-	-	4.67×

We then compare the global compression fidelity of SZ, FPZIP, and our method with the rate-distortion plot in Figure 5. Only result on the piecewise linear field is presented for demonstration purposes. The plot is generated by first compressing data with different global error bounds, and then computing and plotting the peak signal-to-noise ratio (PSNR) and bit rate (average bits per compressed data sample). We can see that our method has comparable rate-distortion trends to those of SZ and FPZIP when the bit rate is high, because our derived error bounds are usually smaller than the global error bound. However, the bit rate of our method stops decreasing after it reaches 2.7, because the compression ratio is dominated by our derived error bounds instead of the global one.

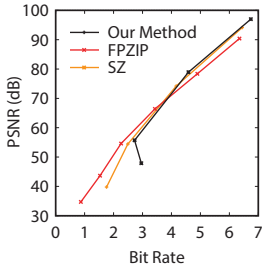


Fig. 5. Rate-distortion of different lossy compressors on the Ocean data.

We present the qualitative results by visualizing both the global view of critical point distribution and local topology. The global critical point distribution (including preserved, FPs, FNs, and FTs) of the different compressors is illustrated in Figure 6. From this figure, we can see that all of the existing compressors have FPs, FNs, and FTs across the global region, whereas our methods preserve all the critical points. We also show the derived error bound and the final relative error of the two approaches in Figs. 6(c), (d), (g), and (h), with the range of $[0, 0.1]$. These figures indicate that the coupled approach indeed allows for higher error bound, leading to higher higher compression ratio than the decoupled approach can. A detailed visualization of the derived error bounds on a zoomed region is displayed in Fig. 7. We can observe that data points that are closed to non-saddle critical points usually require a strict error bound in both schemes. Compared to the decoupled scheme, the coupled scheme allows for higher errors in regions without critical points. This is because we use a pessimistic estimation (Proclaim 2) when deriving the error bounds for avoiding false positive critical points. This estimation leads to strict error bounds when errors of multiple veritices are considered simultaneously in the decoupled scheme, and such impact is mitigated when the error of only one vertex is considered in the coupled scheme.

In Figure 8, we further zoom into local regions to visualize the impact of critical point changes on local topology. We compare

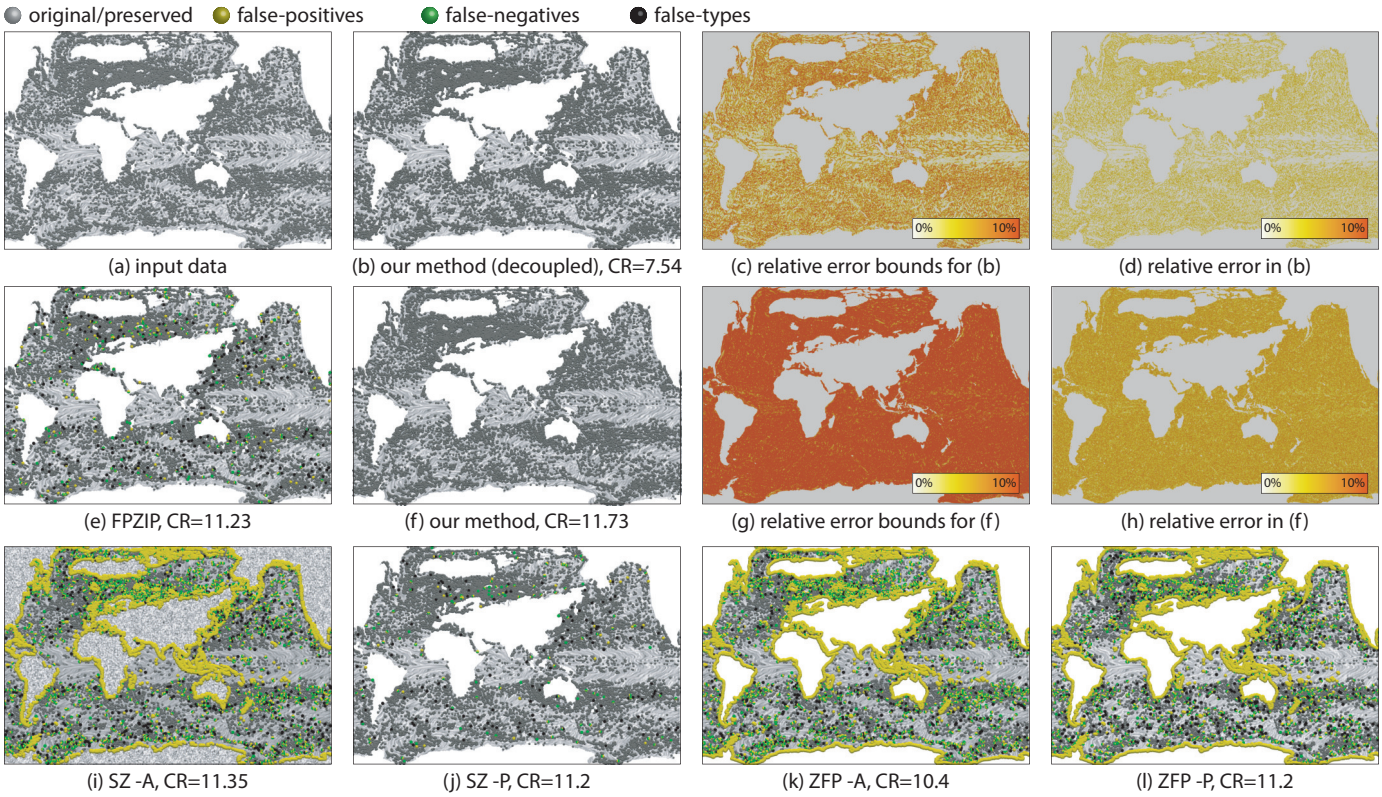


Fig. 6. Visualizations of 2D ocean benchmark with piecewise linear cells. More details are in Table 5.

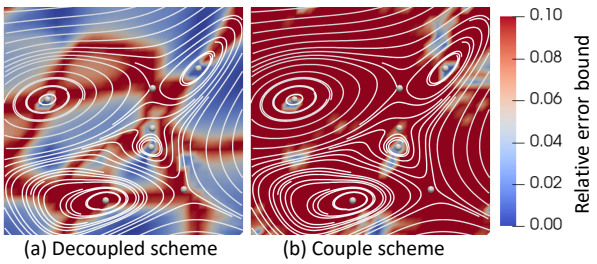


Fig. 7. Visualization of derived error bounds (encoded by color) from the two compression schemes, respectively. Streamlines and critical points are overlayed on top of error bound images for reference.

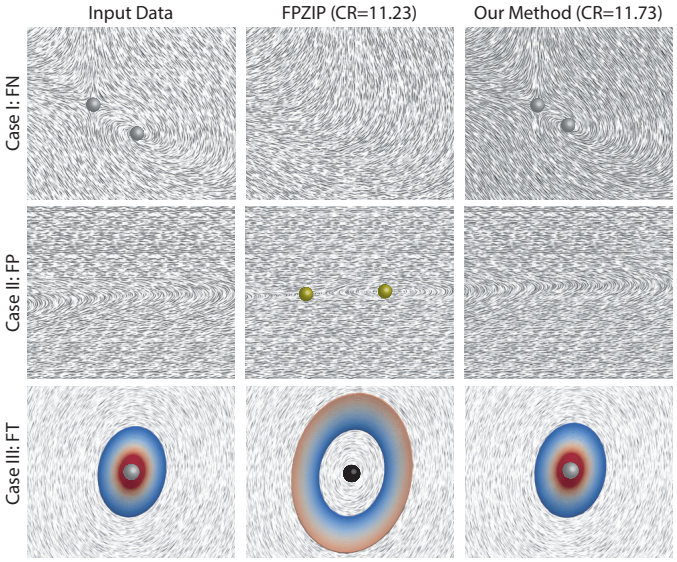


Fig. 8. Visualizations of FN (case I), FP (case II), and FT (case III) in FPZIP decompressed data, compared with the original data and decompressed data of our method. In case III, the critical point type changed from attracting focus to repelling focus with FPZIP; each plot has one single streamline seeded from a fixed location near the critical point, and color encodes the integration time of streamlines.

only with FPZIP for demonstration purpose, because it has the smallest number of FPs, FNs, and FTs. Specifically, we show the difference in the line integral convolution (LIC) of the local region near the critical points for FPs and FNs in case I and case II, where the gray spheres indicate the original/preserved critical points and the yellow spheres represent FPs. For example, the LIC patterns of the saddle and the focus in both the original data and decompressed data using our coupled approach can be observed in case I, while the patterns of the decompressed data of FPZIP shows no critical point. We also trace a streamline to show the impact of type change in case III, where an attracting focus in the original data is turned into a repelling focus in the decompressed data of FPZIP.

7.2 Results with Nek5000 Data

The quantitative results of the different lossy compressors are displayed in Table 7. We skip the absolute error bound mode because of its inefficiency in preserving critical point in previous experiments. Again, the other lossy compressors have critical point changes to achieve a similar compression ratio ($\sim 7.5\times$) to our coupled approach, which preserves all the critical points. Although GZIP can also preserve all the critical points, the compression ratio is only around $1.1\times$. Similarly, the compression

performance of our methods is hindered mainly by the $n_c \approx 6n_v$ and $\sum_i \text{card}(\text{adj\_cells}(i)) \approx 24n_v$ complexity in the 3D regular grid.

We present the qualitative results by visualizing the critical point distribution and local topology in the Nek5000 data. The global views are displayed in Figure 9, with traced streamlines from the same source. We also show the derived error bound and the resulting real errors in the coupled approach. We see in the figure that the streamlines generated from original data and decompressed data of different lossy compressors are almost the same. However, the changed critcal points in the decompressed data of FPZIP result in streamline changes in local regions, as shown in Figure 10. For example, the changed critical point type in case III leads to an attracting effect instead of the repelling effect in the original data.

7.3 Results with LES Data

The unstructured LES data yield similar results, where our coupled approach preserves all the critical points while the other compressors introduce FPs, FNs, and FTs at the same compression ratio, as summarized in Table 8. In this case, the compression performance of our method is affected by both the 22.4× complexity ($\sum_i \text{card}(\text{adj\_cells}(i)) \approx 22.4n_v$) and the construction of unstructured grid.

Figure 11 visualizes the global critical point distribution with traced streamlines as context. Again, we see little change in all the global streamlines, but the other lossy compressors lead to FPs, FNs, and FTs in different locations.

7.4 Limitations

Separatrix preservation Our method does not theoretically guarantee the preservation of separatrices, but empirical studies show that our method outperforms general-purpose lossy compressors in preserving separatrices. As shown in Figure 12(b), separatrices are changed in the FPZIP results because a saddle-source pair is missing. In Figure 12(c), the separatrices are preserved in our decompressed data. We discussed the limitation with ocean climate researchers, and the preservation of critical point locations and types are important because they may imply features such as eddies. The preservation of separatrices may be achieved by iteratively reducing the global error bound until all separatrices are kept; we leave the separatrix preservation for future work.

Eigenvector direction preservation Our method introduces distortion to the eigenvalues and eigenvectors of critical points, which may in turn change the topology of separatrices. However, experiments show that the impact to eigenvector directions is usually minimal, as demonstrated in Figure 12. Eigenvector directions can be strictly preserved by applying zero error bounds for corresponding cells, which may affect the compression ratio.

Generalization to trilinear vector fields Our compressor currently does not consider trilinear vector fields. The error bound to avoid FN, FP, and FT may be derived with the technique presented in this paper but with much higher degrees of polynomials involved. Thus, closed-form error bounds may not be available, and numerical approximations are necessary. We will study feature preserving compression in trilinear and higher-order interpolated vector fields in future work.

Stability of critical points Our compression scheme preserves every critical point in the input vector field, which may not be necessary for practice because not all critical points are equally important per recent studies [29]. One possible improvement is to allow thresholding of critical points with high stability measures instead of preserving every single one. However, difficulties exist in incorporating stability metrics and determining proper thresholds, which we leave for future work.

8 Conclusions and Future Work

This paper introduces a theoretical framework to strictly preserve critical points by adapting vertex-wise error bounds in lossy compression. We also present two approaches to achieve feature-preserving compression: decoupled and coupled methods; the decoupled method is optimized for performance, while the coupled method has higher compression ratio. Experiments demonstrate that the proposed methods can preserve the critical points and local topologies are preserved in the visualization results.

We would like to further investigate preserving more topological features—topological skeletons, vortex core lines, and boundary surfaces—with error-bounded lossy compression. We would also like to generalize our method to trilinear, higher-order finite elements, and spectrum mesh elements in addition to simplicial cells. In addition to Lorenzo predictors, we will also investigate regression- and statistics-based predictors to further improve the compression quality.

Acknowledgments

We thank Dr. Jeffery Larson, Dr. Todd Munson, and Dr. Chongke Bi for useful discussions. Work by Chunhui Liu was supported by JSPS KAKENHI Grant Number JP17F17730 and JSPS grant (S) 16H06335. This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357. This work is supported by the U.S. Department of Energy, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program and the Exascale Computing Project (ECP, Project Number: 17-SC-20-SC), a collaborative effort of two DOE organizations – the Office of Science and the National Nuclear Security Administration. This work is also supported by the National Science Foundation under grant OAC-2153451, OAC-2003709, and OAC-2104023. We acknowledge the computing resources provided on Bebop, which is operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

References

[1] 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 *Proc. IEEE International Conference on Big Data*. IEEE, 2018, pp. 438–447.

[2] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 2674–2683, 2014.

[3] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "TTHRESH: Tensor compression for multidimensional visual data," *IEEE Trans. Vis. Comput. Graph.*, 2019.

[4] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1245–1250, 2006.

[5] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[6] T. McLoughlin, R. S. Laramee, R. Peikert, F. Post, and M. Chen, "Over two decades of integration-based, geometric flow visualization," *Comput. Graph. Forum*, vol. 29, no. 6, pp. 1807–1829, 2010.

TABLE 7
Benchmark of (lossy) compressors on Nek5000 data.

Compressor	Setting	e_r	CR_u	CR_v	CR_w	CR_{all}	S_c (MB/s)	S_d (MB/s)	#TP	#FP	#FN	#FT
GZIP	-1	0	1.09×	1.09×	1.09×	1.09×	23.32	119.71	10,587	0	0	0
Our method	decoupled	-	-	-	-	3.27×	8.97	41.46	10,587	0	0	0
Our method	coupled	-	-	-	-	7.48×	6.38	62.14	10,587	0	0	0
FPZIP	-P 16	2^{-7}	6.87×	6.73×	7.59×	7.04×	94.90	82.76	10,499	98	72	16
SZ	-P 0.015	0.015	7.14×	6.74×	7.75×	7.19×	97.27	148.06	10,199	358	326	62
ZFP	-P 13	0.0625	6.59×	6.47×	6.95×	6.66×	129.29	306.60	9,927	695	566	94

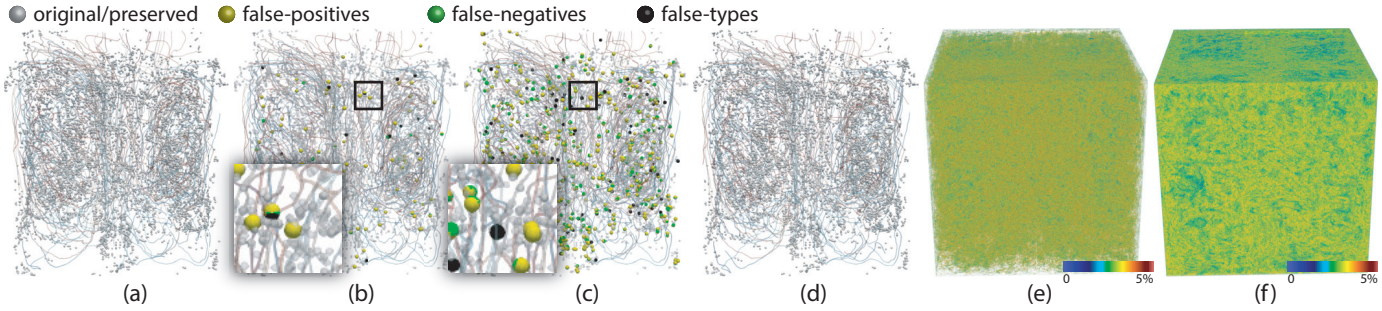


Fig. 9. Visualization of Nek5000 simulation data: (a) original data with all critical points, (b) FPZIP decompressed data with false critical points, (c) SZ decompressed data with false critical points, (d) decompressed data from our method with all critical points, and (e) vertex-wise error bound derived and (f) vertex-wise error by our method. Streamlines are visualized as context.

TABLE 8
Benchmark of (lossy) compressors on unstructured LES Data.

Compressor	Setting	e_r	CR_u	CR_v	CR_w	CR_{all}	S_c (MB/s)	S_d (MB/s)	#TP	#FP	#FN	#FT
GZIP	-1	0	1.16×	1.07×	1.07×	1.10×	24.8	108.7	1,024	0	0	0
Our method	coupled	-	-	-	-	5.06×	1.82	60.55	1,024	0	0	0
FPZIP	-P 13	0.0625	6.78×	4.27×	4.27×	4.87×	81.95	73.88	992	31	25	7
SZ	-P 0.02	0.02	6.44×	4.39×	4.38×	4.91×	121.06	238.82	984	13	29	11
ZFP	-P 6	2	5.13×	4.75×	4.77×	4.88×	136.32	193.6	241	1870	708	75

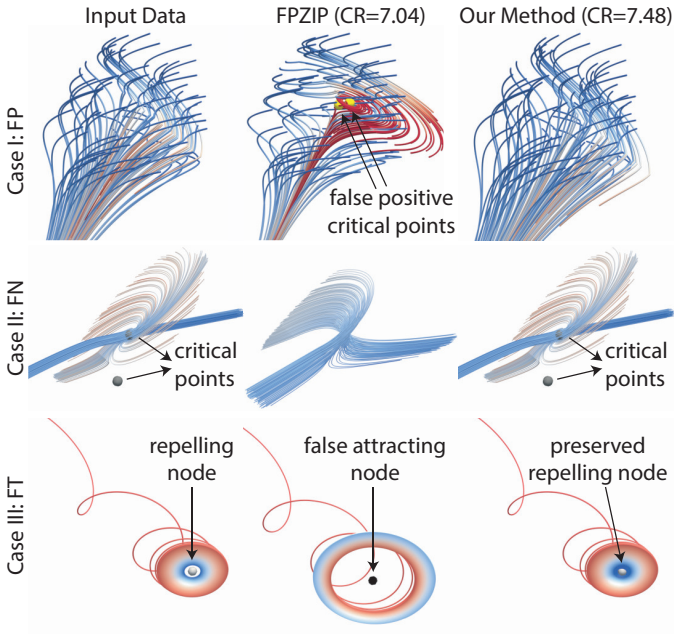


Fig. 10. Visualizations of local streamline changes in FPZIP decompressed data, compared with the original data and decompressed data of our method. In case III, the streamline in each figure is seeded closely to the critical point in the original data, and colors encode the integration time of streamlines.

[7] R. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. Post, and D. Weiskopf, "The state of the art in flow visualization: Dense and texture-based techniques," *Comput. Graph. Forum*, vol. 23, no. 2, pp. 203–222, 2004.
[8] R. S. Laramee, H. Hauser, L. Zhao, and F. H. Post, "Topology-based flow visualization, the state of the art," in *Proc. Topology-Based Methods in Visualization*, 2007, pp. 1–19.
[9] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth, "A survey of topology-based methods in visualization," *Comput. Graph. Forum*, vol. 35, no. 3, pp. 643–667, 2016.
[10] H. Theisel, C. Rössl, and H.-P. Seidel, "Compression of 2D vector fields under guaranteed topology preservation," *Comput. Graph. Forum*, vol. 22, no. 3, pp. 333–342, 2003.
[11] S. K. Lodha, J. C. Renteria, and K. M. Roskin, "Topology preserving compression of 2D vector fields," in *Proc. IEEE Visualization 2000*, 2000, pp. 343–350.
[12] M. Burtscher and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 18–31, 2008.
[13] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. P. Clyne, and H. Childs, "Data reduction techniques for simulation, visualization and data analysis," *Comput. Graph. Forum*, vol. 37, no. 6, pp. 422–447, 2018.
[14] M. Rodríguez, E. Gobbetti, J. Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. Suter, "State-of-the-art in compressed GPU-based direct volume rendering," *Comput. Graph. Forum*, vol. 33, no. 6, pp. 77–100, 2014.
[15] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," *Computer Graphics Forum*, vol. 22, no. 3, pp. 343–348, 2003.
[16] "GZIP," https://www.gzip.org, 2022.
[17] "ZSTD," http://www.zstd.net, 2022.
[18] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the univariate case," *Computing and Visualization in Science*, vol. 19, no. 5-6, pp. 65–76, 2018.
[19] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate

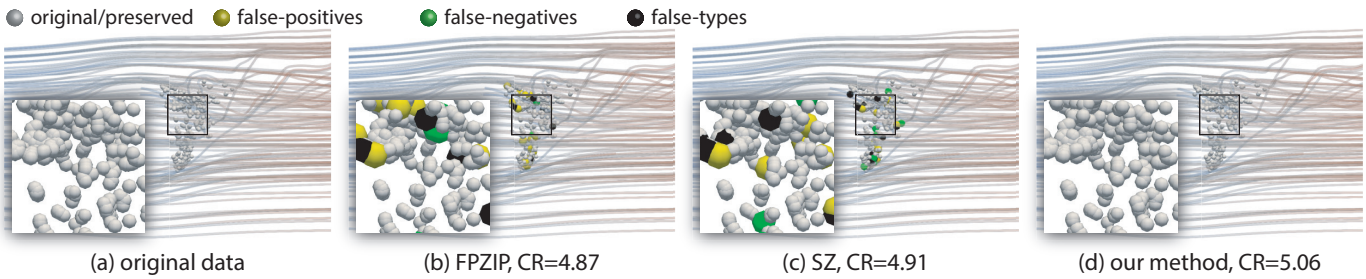


Fig. 11. Visualization of LES unstructured data: (a) original data with all critical points, (b) FPZIP decompressed data with false critical points, (c) SZ decompressed data with false critical points, and (d) decompressed data from our method with all critical points. Streamlines are visualized as context.

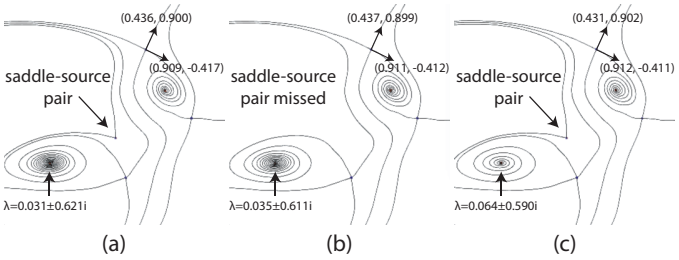


Fig. 12. Zoomed visualization of separatrices in (a) original ocean data, (b) decompressed data with FPZIP (CR=7.04), and (c) decompressed data with our method (CR=7.48).

case," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A1278–A1303, 2019.

[20] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data-quantitative control of accuracy in derived quantities," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2146–A2171, 2019.

[21] R. Underwood, S. Di, J. C. Calhoun, and F. Cappello, "Fraz: a generic high-fidelity fixed-ratio lossy compression framework for scientific floating-point data," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2020, pp. 567–577.

[22] A. Telea and J. J. van Wijk, "Simplified representation of vector fields," in *Proc. IEEE Visualization 1999*, 1999, pp. 35–42.

[23] T. K. Dey, J. A. Levine, and R. Wenger, "A Delaunay simplification algorithm for vector fields," in *Proc. Pacific Conference on Computer Graphics and Applications*, 2007, pp. 281–290.

[24] S. Koch, J. Kasten, A. Wiebel, G. Scheuermann, and M. Hlawitschka, "2D vector field approximation using linear neighborhoods," *The Visual Computer*, vol. 32, no. 12, pp. 1563–1578, 2016.

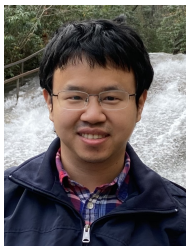
[25] J. Helman and L. Hesselink, "Representation and display of vector field topology in fluid flow data sets," *IEEE Computer*, vol. 22, no. 8, pp. 27–36, 1989.

[26] H. Theisel, C. Rössl, and T. Weinkauf, "Topological representations of vector fields," in *Shape Analysis and Structuring*, L. D. Floriani and M. Spagnuolo, Eds. Springer, 2008, pp. 215–240.

[27] 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 *Proc. IEEE International Parallel and Distributed Processing Symposium*, 2017, pp. 1129–1139.

[28] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello, "An efficient transformation scheme for lossy data compression with point-wise relative error bound," in *CLUSTER'18: Proc. IEEE International Conference on Cluster Computing*. IEEE, 2018, pp. 179–189.

[29] P. Skraba, P. Rosen, B. Wang, G. Chen, H. Bhatia, and V. Pascucci, "Critical point cancellation in 3d vector fields: Robustness and discussion," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 6, pp. 1683–1693, 2016.



Xin Liang is an assistant professor with the Department of Computer Science at University of Kentucky. Prior to that, he worked as an assistant professor at Missouri University of Science and Technology and a Computer/Data Scientist at Oak Ridge National Laboratory. He received his Ph.D. degree from University of California, Riverside in 2019 and his bachelor's degree from Peking University in 2014. His research interests include high-performance computing, parallel and distributed systems, scientific data management and reduction, big data analytic, and scientific visualization. He is a member of the IEEE. Email: xliang@uky.edu.

Sheng Di (Senior Member, IEEE) received his master's degree from Huazhong University of Science and Technology in 2007 and Ph.D. degree from the University of Hong Kong in 2011. He is currently a computer scientist at Argonne National Laboratory. Dr. Di's research interest involves resilience on high-performance computing (such as silent data corruption, optimization checkpoint model, and in-situ data compression) and broad research topics on cloud computing (including optimization of resource allocation, cloud network topology, and prediction of cloud workload/hostload). He is working on multiple HPC projects, such as detection of silent data corruption, characterization of failures and faults for HPC systems, and optimization of multilevel checkpoint models. He is the recipient of DOE 2021 Early Career Research Program Award. Email: sdi1@anl.gov.

Franck Cappello (Fellow, IEEE) is the director of the Joint-Laboratory on Extreme Scale Computing gathering six of the leading high-performance computing institutions in the world: Argonne National Laboratory, National Center for Scientific Applications, Inria, Barcelona Supercomputing Center, Julich Supercomputing Center, and Riken AICS. He is a senior computer scientist at Argonne National Laboratory and an adjunct associate professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He is an expert in resilience and fault tolerance for scientific computing and data analytics. Recently he started investigating lossy compression for scientific data sets to respond to the pressing needs of scientist performing large-scale simulations and experiments. His contribution to this domain is one of the best lossy compressors for scientific data set respecting user-set error bounds. He is a member of the editorial board of the *IEEE Transactions on Parallel and Distributed Computing* and of the *ACM HPDC* and *IEEE CCGRID* steering committees. He is a fellow of the IEEE. Email: cappello@mcs.anl.gov.

Mukund Raj is a senior software engineer at the Broad Institute of MIT and Harvard. He received his Ph.D. in computing from the University of Utah in 2018. From 2018 to 2021, he was a postdoctoral appointee at Argonne National Laboratory. His interests are primarily in data visualization, particularly in developing visualizations for ensembles, genomics data, and large-scale data. E-mail: mraj@broadinstitute.org.

Chunhui Liu is a JSPS international research fellow working in Department of Mathematics, Faculty of Science, Kyoto University. He gets his Ph. D. degree in mathematics from Université Paris Diderot - Paris 7, France.

Kenji Ono is currently a director of Research Institute for Information Technology in Kyushu University, and he holds an appointment at the University of Tokyo, after working on RIKEN Advanced Institute for Computational Science and Nissan Motor company. He received his degrees of Dr. Eng. in mechanical engineering from Kumamoto University in 2000. His research fields are computational fluid dynamics, parallel computation, visualization and equation discovery from large-scale dataset. Email: keno@cc.kyushu-u.ac.jp

Zizhong Chen (Senior Member, IEEE) received a bachelor's degree in mathematics from Beijing Normal University, a master's degree degree in economics from the Renmin University of China, and a Ph.D. degree in computer science from the University of Tennessee, Knoxville. He is a professor of computer science at the University of California, Riverside. His research interests include high-performance computing, parallel and distributed systems, big data analytics, cluster and cloud computing, algorithm-based fault tolerance, power and energy efficient computing, numerical algorithms and software, and large-scale computer simulations. He received a CAREER Award from the US National Science Foundation and a Best Paper Award from the International Supercomputing Conference. Email: chen@cs.ucr.edu.

Tom Peterka (Member, IEEE) received the PhD in computer science from the University of Illinois at Chicago in 2007. He is currently a computer scientist with Argonne National Laboratory, a scientist with the University of Chicago Consortium for Advanced Science and Engineering, an adjunct assistant professor with the University of Illinois at Chicago, and a fellow with the Northwestern Argonne Institute for Science and Engineering. He currently leads several DOE- and NSF-funded projects. He has authored or coauthored more than 100 peer-reviewed papers in conferences and journals that include the ACM/IEEE SC, IEEE IPDPS, IEEE VIS, IEEE TVCG, and ACM SIGGRAPH. His research focuses on large-scale parallel in situ analysis of scientific data. He was the recipient of the 2017 DOE Early Career Award and four best paper awards.

Hanqi Guo (Member, IEEE) received the BS degree in mathematics and applied mathematics from the Beijing University of Posts and Telecommunications in 2009 and the PhD degree in computer science from Peking University in 2014. He is an Associate Professor at the Department of Computer Science and Engineering in the Ohio State University. His research interests include data analysis, visualization, and machine learning for scientific data. He is an awardee of the DOE Early Career Research Program (ECRP) in 2022 and received multiple best paper awards in premiere visualization conferences. Email: guo.2154@osu.edu.