TspSZ: An Efficient Parallel Error-Bounded Lossy Compressor for Topological Skeleton Preservation

Mingze Xia
University of Kentucky
Lexington, KY, USA
mingze.xia@uky.edu

Bei Wang
University of Utah
Salt Lake City, UT, USA
beiwang@sci.utah.edu

Yuxiao Li
The Ohio State University
Columbus, OH, USA
li.14025@osu.edu

Pu Jiao
University of Kentucky
Lexington, KY, USA
pujiao@uky.edu

Xin Liang*
University of Kentucky
Lexington, KY, USA
xliang@uky.edu

Hanqi Guo
The Ohio State University
Columbus, OH, USA
guo.2154@osu.edu

***Abstract*—Data compression is a powerful solution for addressing big data challenges in database and data management. In scientific data compression for vector fields, preserving topological information is essential for accurate analysis and visualization. The topological skeleton, a fundamental component of vector field topology, consists of critical points and their connectivity (i.e., separatrices). While previous work has focused on preserving critical points in error-controlled lossy compression, little attention has been given to preserving separatrices, which are equally important. In this work, we introduce TspSZ, an efficient error-bounded lossy compression framework designed to preserve both critical points and separatrices. Our key contributions are threefold. First, we propose TspSZ, a topological-skeleton-preserving lossy compression framework that integrates two algorithms, enabling existing critical-point-preserving compressors to also retain separatrices, significantly enhancing their topology preservation capabilities. Second, we optimize TspSZ for efficiency through tailored improvements and parallelization. Specifically, we introduce a new error control mechanism to achieve high compression ratios and implement a shared-memory parallelization strategy to boost compression throughput. Third, we evaluate TspSZ against state-of-the-art lossy and lossless compressors using four real-world scientific datasets. Experimental results show that TspSZ achieves compression ratios of up to $7.7\times$ while effectively preserving the topological skeleton, ensuring efficient storage and transmission of scientific data without compromising topological integrity.**

***Index Terms*—Scientific data management, high-performance computing, lossy compression, vector field topology**

I. Introduction

With the recent advancement in high-performance computing, scientific applications are generating extremely large volumes of data that require remote access and long-term storage. This poses grand challenges because of the limited data transfer bandwidth and/or retrieval rate from the storage systems, which significantly hinders the speed of scientific discoveries in a wide range of applications.

Compression is regarded as one of the most promising ways to address such big data problems and has been extensively used in the field of database and data management, including the design of databases [1]–[5], acceleration of queries [6], [7] and speedup for analytics [8], [9]. However, generic lossless compression techniques [10]–[12] only have modest compression ratios on scientific data, necessitating the need for effective lossy compression. Meanwhile, traditional lossy compression techniques for image compression [13], [14] cannot provide a quantifiable error bound and thus fail to preserve the integrity of scientific data. As such, error-controlled lossy compressors [15]–[22] have been proposed to produce decent compression ratios while enforcing user-specified constraints.

Most existing error-controlled lossy compressors, including SZ [17], ZFP [19], and MGARD [20], enable error control in terms of point-wise error bounds or mean-squared errors. While providing error control is important, it may still alter critical features essential for scientific discoveries. For example, previous studies [23] have shown that directly applying off-the-shelf error-controlled compressors can distort the detection of blobs—key structures used to analyze the separatrix of high-energy particles in fusion energy science.

Topological data analysis offers powerful tools for extracting topological features from scientific data across various domains, including turbulent combustion [24], cosmology [25], climatology [26], and computational physics [27]. In vector field data, the topological skeleton is a key topological feature that provides a concise representation of flow behavior. It consists of critical points and separatrices, which connect specific pairs of critical points. In fluid dynamics, the topological skeleton is utilized to examine vortex structure and boundary layer behavior [28], [29]. In medical imaging, it assists in analyzing blood and airflow patterns [30]–[32]. And in climate studies, it helps characterize the internal structure of atmospheric rivers [33].

Many of these scientific applications produce vast amounts of data, straining storage and transmission systems. For instance, Phase 6 of the Coupled Model Intercomparison Project (CMIP) in climate simulation generates 28 PB of data [34], posing significant challenges for archiving and data transfer.

*Corresponding author: Xin Liang, Department of Computer Science, University of Kentucky, Lexington, KY 40506.

While several error-controlled lossy compressors have been proposed to reduce data size while preserving critical points (e.g. [35], [36]), none have focused on preserving the topological skeleton during compression.

Preserving the entire topological skeleton of the target vector field during lossy compression is nontrivial. First, preserving critical points is the first step in preserving the topology, which is a challenging problem by itself. Second, separatrices are constructed from numerical integration, where minor errors could accumulate to make a huge difference. For the same reason, it is hard to derive a theoretical bound to enable preservation. Third, the extraction of separatrices is computationally expensive and thus requires careful parallelization. These gaps motivate us to develop a high-performance compression framework that preserves the topological skeleton while efficiently reducing the data size.

In this work, we propose Topological-Skeleton-Preserving-SZ (TspSZ), a novel error-controlled lossy compression framework that encapsulates two methods to preserve the topological skeleton for scientific data. Building upon the recently developed critical-point-preserving lossy compressor cpSZ [35], TspSZ features the preservation of the entire topological skeleton, which comprises both critical points and the separatrices, during lossy compression. This is fundamentally different from cpSZ, which only preserves critical points and overlooks the preservation of separatrices. We further optimize the TspSZ to achieve high compression ratios with decent compression performance, and evaluate it on four real-world scientific datasets. In summary, our contributions are as follows.

- We design and develop TspSZ, an error-controlled lossy compression framework capable of preserving the full vector field topology with a two-phase workflow. By enhancing cpSZ with a lossless encoding of cells with present critical points in the first phase, TspSZ preserves all critical points with exact positions and eigenvectors; it then encapsulates two methods in the second phase for the reservation of separatrices: an intuitive method with bounded execution time and an iterative method that trades off performance for high compression ratios.
- We analyze the error accumulation in the computation of separatrices and identify the potential low-data-quality problem in TspSZ due to the use of point-wise relative error bound in cpSZ. We then replace it with point-wise absolute error bound through rigorous derivation to significantly improve the quality of the reconstructed data and thus reduce the number of wrong separatrices in the second phase of TspSZ.
- We optimize TspSZ through careful implementation along with tailored parallelization in a shared-memory environment. In particular, we parallelize both the critical-point-preserving compression in cpSZ and the newly proposed methods for preserving separatrice to ensure the entire workflow is highly scalable.
- We evaluate TspSZ and compare it with cpSZ, a derived variant of cpSZ [36], and lossless compressors using four real-world datasets. Experimental results demonstrate that TspSZ faithfully preserves the topology of vector fields while delivering up to $7.7\times$ compression ratios, which is significantly better than existing lossless compressors. In addition, TspSZ demonstrates high scalability with hundreds of threads in a shared-memory environment, leading to very high performance for decompression with acceptable performance for compression.

The remaining sections of the paper are organized as follows. Section II discusses related works. Section III reviews the background on vector fields and topological skeleton. Section IV formulates the research problem and provides an overview. In Section V, we introduce the methods for preserving separatrices in TspSZ. In Section VI, we present the analysis of error accumulation in the computation of separatrices, followed by our optimization strategy to replace point-wise relative error control with absolute error control in cpSZ. In Section VII, we demonstrate our parallelization strategies to achieve high efficiency and scalability. In Section VIII, we present and analyze the experimental results. In Section IX, we conclude our work with a vision for future work.

II. Related Works

In this section, we review the literature on scientific data compression and topology-preserving compression.

A. Error-controlled lossy compression for scientific data

As scientific computing scales in size and complexity, the volume of generated data increases correspondingly, necessitating the need for efficient data compression methods to support data storage and transmission. Lossless compression techniques, such as GZIP [10], ZSTD [11], and BLOSC [12], suffer from limited compression ratios (often less than two according to existing studies [37], [38]). As such, lossy compression methods capable of providing much higher compression ratios are increasingly considered a viable alternative to address the growing data challenge.

Nevertheless, traditional lossy compression methods, such as JPEG [13] and JPEG2000 [14] from the image processing domain, generally fail to provide a quantifiable bound on the reconstruction error. This failure severely limits their use in scientific applications that require error guarantees to ensure the accuracy of downstream data analytics. As a result, researchers have been focusing on developing error-controlled lossy compression techniques to ensure that compressed data remains analytically valid for scientific purposes.

Error-controlled lossy compression methods are broadly categorized as prediction-based [15]–[18] and transform-based approaches [19]–[21]. SZ [15] is a typical prediction-based compressor comprising three major components in the compression pipeline: prediction, quantization, and lossless encoding. The input data is first decorrelated by various prediction algorithms and then quantized to integers with guaranteed error control and reduced entropy. These integers are then fed into lossless encoders such as Huffman [39] and ZSTD [11] for actual size reduction. ZFP [19] is a typical transform-based compressor that compresses data in a block-wise fashion.

The input data is divided into individual blocks and operated independently. Inside each block, the data is aligned to the same exponent and converted to fix-point representation, and then a near-orthogonal transform is performed to decorrelate the data. To this end, embedded encoding is applied to the transformed data to reduce the size with error control.

B. Topology-preserving lossy compression

Given the critical role of topological features in scientific data, certain research efforts concentrate on how to effectively preserve these features within lossy compression frameworks. Topological structures often represent essential data characteristics, such as flow structures or segmented regions, that are crucial for scientific interpretation and analysis. Several approaches have been proposed to address this need in scalar fields. For example, MSz [40] is a specialized compression algorithm that preserves piecewise linear Morse-Smale segmentations, making it suitable for applications requiring the retention of gradient and segmentation information within scalar fields. Soler et al. [41] developed a topology-controlled compression method that preserves the persistence diagram by adaptively quantizing data based on a specified persistence simplification threshold. Yan et al. [42] proposed TopoSZ, which enhanced the SZ 1.4 compression algorithm by incorporating topological constraints derived from segmentations guided by contour trees. Despite the usefulness of these methods in preserving scalar field topology, they cannot be directly generalized to vector fields.

Vector field compression has also been studied for a long time. Lodha et al. [43] presented a method to compress 2D vector fields via iterative clustering, but the underlying algorithm is difficult to generalize to 3D cases. Dey et al. [44] applied a Delaunay simplification to vector fields, but it did not provide explicit preservation of the topological features. Theisel et al. [45] also leveraged edge collapse algorithms for guaranteed topology preservation, but this method is very time-consuming and enforces no point-wise error control. Recently, variations of error-controlled lossy compressors [35], [36] have been proposed to preserve critical points in vector fields while enabling point-wise error control. This includes guaranteed preservation of critical points extracted from both numerical methods [35] and sign-of-determinant predicates [36]. Nonetheless, both these methods overlook the preservation of separatrices, which is at least of equal importance to critical points in vector field topology.

Although built upon cpSZ [35], the proposed Tsp-SZ is significantly different from cpSZ in three aspects. First, Tsp-SZ enhances cpSZ by providing two algorithms to preserve separatrices during lossy compression, which enables the preservation of the topological skeleton, which is essential for scientific applications. Second, Tsp-SZ replaces the point-wise relative error control in cpSZ with absolute error control to improve the quality of decompressed data, which in turn reduces the number of wrong separatrices after enabling critical point preservation. Third, Tsp-SZ features a careful parallelization on shared-memory systems, delivering decent compression ratios with high performance and scalability. Note that the algorithms in Tsp-SZ can also be applied to cpSZ-sos [36] to preserve the topological skeleton computed from critical points extracted by the simulation of simplicity [46]. However, this may create discrepancies in the visualization because these critical points may not align with the numerically extracted ones and the numerically integrated separatrices. As such, we implement Tsp-SZ upon cpSZ instead of cpSZ-sos.

TABLE I
NOTATIONS

Symbol	Meaning
n_v	Number of vertices/data points.
n_c	Number of cells.
$\mathbf{u}, \mathbf{v}$	Components of the vector fields.
$\mathbf{u}', \mathbf{v}'$	Decompressed forms of $\mathbf{u}, \mathbf{v}$.
u_i, v_i	One element in the corresponding component.
μ_k	Barycentric coordinates of critical points.
h	Step size used in RK4.
$\mathbf{s}$	Position of a critical point.

III. BACKGROUND

In this section, we review the background on vector field topology and critical-point-preserving lossy compression. The commonly used notations in the paper are described in Table I.

A. Vector field and streamlines

Vector fields are functions of a space whose value at each point is a vector quantity. They are usually used to represent velocity-related variables in scientific applications, such as wind speed in climatology, current speed in oceanology, and magnetic field directions in physics. A 2D vector field is usually defined as $\mathbf{V} = (\mathbf{u}, \mathbf{v})$, where (u_i, v_i) represents the vector quantity at the i-th vertex.

One normal operation in a vector field is streamline tracing, which computes the separatrix of a particle when it is placed in the flow using numerical integration. The most widely used method to perform such integration is the Runge–Kutta method (RK4) [47], [48], which computes the discrete points along the separatrix using the formula below:

$$\begin{aligned} &k_1 = f(t_n, y_n), \quad k_2 = f\left(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_1\right), \\ &k_3 = f\left(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_2\right), \quad k_4 = f(t_n + h, y_n + hk_3), \\ &y_{n+1} = y_n + \tfrac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{1}$$

where k_i represents the intermediate slopes calculated at different points in the interval $[t_n, t_{n+1}]$. Specifically, k_1 is the slope at the beginning of the interval; k_2 is the slope at the midpoint, estimated using k_1; k_3 is another midpoint slope, this time using k_2; and k_4 is the slope at the end of the interval, calculated using k_3.

The RK4 method has a local truncation error of order $O(h^5)$ at each integration step, where h is the step size. After t steps, the global truncation error for a fixed time interval $[0, T]$ becomes $\sim O(h^4)$.

B. Topological skeleton

The topological skeleton gives a compact description of the global flow behavior in a vector field by separating the flow into areas of different behaviors. It mainly comprises critical points and their connections, a.k.a separatrices. We introduce

these concepts using 2D data as an example, and they can be directly generalized to 3D cases. In this work, we focus on piece-wise linear cells, which is one of the most widely adopted manifold maps in literature.

Critical points: A critical point is defined as the location where the vector field vanishes, i.e., $\mathbf{V}(\mathbf{x}) = \mathbf{0}$. In a 2D piecewise linear vector field, this problem can be formulated as solving a linear equation with barycentric coordinates:

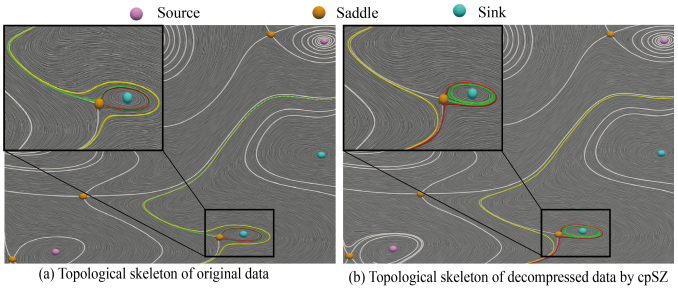
$$\begin{bmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \end{bmatrix} = \mathbf{0} \text{ and } \mu_0 + \mu_1 + \mu_2 = 1, \quad (2)$$

where (μ_0, μ_1, μ_2) is the barycentric coordinates, and (u_i, u_j) are the vector quantities in the vertices that constitute the cell. A critical point exists in the cell if and only if $0 \leq \mu_k \leq 1$ holds for any $k \in \{0, 1, 2\}$.

Critical points can be categorized into sinks, sources, and saddles, in general, depending on the eigenvalues of the Jacobian matrix [29], [49], which characterizes flow behaviors in the local region. Two positive eigenvalues indicate a source with repelling behaviors, while two negative eigenvalues imply a sink with attracting behaviors, and the presence of imaginary parts in these two cases results in the circulation behavior of the source/sink. One positive eigenvalue and one negative eigenvalue lead to a saddle, which may have attracting and repelling behaviors in different local regions.

Separatrices: Separatrices are a special set of streamlines that connect critical points. They essentially act as boundaries that separate different types of motion within a flow field.

Separatrices can be constructed by tracing streamlines around saddles. For any saddle point $\mathbf{s}$ with the Jacobian matrix $\mathbf{J}(\mathbf{s})$, four separatrices can be spawned for both eigenvectors of $\mathbf{J}(\mathbf{s})$ in the forward and backward direction. In other words, separatrices are streamlines traced at $(\mathbf{s}+\epsilon_p * \mathbf{j_1})$, $(\mathbf{s}-\epsilon_p * \mathbf{j_1})$, $(\mathbf{s}+\epsilon_p * \mathbf{j_2})$, and $(\mathbf{s}-\epsilon_p * \mathbf{j_2})$ in both forward and backward directions, where ϵ_p is sufficiently small perturbation and $\mathbf{j_1}$ and $\mathbf{j_2}$ are the eigenvectors of $\mathbf{J}(\mathbf{s})$. Fig. 1(a) illustrates an example of the topological skeleton in the ocean dataset, which characterizes the flow behaviors into separate regions.



(a) Topological skeleton of original data (b) Topological skeleton of decompressed data by cpSZ

Fig. 1. Topological skeletons of (a) the original data and (b) the decompressed data of cpSZ [35], with the same color encoding separatrices traced from the same location. Although cpSZ retains critical points in the data, it distorts separatrices, resulting in misinterpretation of local flow behavior.

C. cpSZ – error-controlled lossy compression with critical point preservation

We review the coupled compression scheme proposed in cpSZ, which serves as a foundation and baseline of the proposed work. As outlined in Algorithm 1, cpSZ visits each data point (also called a vertex) in a predefined order, and performs prediction and quantization using the traditional prediction-based compression pipeline in SZ. The key innovation in cpSZ is to derive a sufficient error bound for each vertex, which ensures that the critical points in all the adjacent cells can be preserved (lines 3-7). This error bound is further quantized to reduce the storage overhead (line 8), and the quantized value is then used as an error bound to compress the current vertex using an existing algorithm (line 9). Note that the original value needs to be overwritten by the decompressed value after processing, to ensure the proper derivation of error bounds for later points. This process allows for the correct execution during decompression when the original data is unavailable.

Algorithm 1 Coupled Feature-Preserving Lossy Compression for 2D data with cpSZ

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

1: buffer=$\{\emptyset\}$ ▷ integer buffer for compression
2: **for** $i \leftarrow 0$ to $n_v - 1$ **do** ▷ iterate vertices
3: **for** $j \in$ vertex_cells(i) **do** ▷ iterate cells connected to vertex i
4: $\{i_0, i_1, i_2\} \leftarrow$ cell_vertices(j) ▷ vertices of cell j
5: $\xi_i^{(j)} \leftarrow$ `eb_coupled`$((\mathbf{v}_{i_0}, \mathbf{v}_{i_1}, \mathbf{v}_{i_2}), (\mathbf{x}_{i_0}, \mathbf{x}_{i_1}, \mathbf{x}_{i_2}))$ ▷ derive error bound that ensure critical point preservation
6: **end for**
7: $\xi_i \leftarrow \min_j \xi_i^{(j)}$ ▷ aggregate error bound for vertex i
8: $\widehat{\xi_i} \leftarrow$ `quant`(ξ_i) ▷ quantize error bound of vertex i
9: bytes$\leftarrow$ `lossy_compress`$(\mathbf{v_i}, \widehat{\xi_i})$ ▷ quantize vector data with ξ_i
10: $\mathbf{v}_i' \leftarrow$ `decode`(bytes, $\widehat{\xi_i}$) ▷ calculate decompressed value $\mathbf{v}_i'$
11: $\mathbf{v}_i \leftarrow \mathbf{v}_i'$ ▷ replace input value with the decompressed value
12: buffer.append($\mathbf{d}$)
13: **end for**
14: **return** compress_losslessly(buffer, $\{\widehat{\xi_i}\}$)

Rigorous theories have been proposed in cpSZ to derive the sufficient error bound to avoid false negative (FN, a critical point that exists in the original data but is absent in the decompressed data), false positive (FP, a critical point that is absent in the original data but presents in the decompressed data), and false type (FN, a critical point that presents in both original and decompressed data with different types) cases. We introduce the key theorem to avoid FP in cpSZ as it will be optimized in our work, and we refer the readers to [35] for a full treatment. Note that cpSZ leverages a point-wise relative error bound ξ to perform the derivation, which ensures $|\frac{x_i - x_i'}{x_i}| < \epsilon_r$ for any data x_i and its decompressed value x_i'.

Theorem 1: Given vertices $\mathbf{V_0} = (u_0, v_0)$, $\mathbf{V_1} = (u_1, v_1)$, and $\mathbf{V_2} = (u_2, v_2)$ of a piece-linear cell and let $m_0 = |\begin{smallmatrix} u_1 & u_2 \\ v_1 & v_2 \end{smallmatrix}|$, $m_1 = |\begin{smallmatrix} u_0 & u_2 \\ v_0 & v_2 \end{smallmatrix}|$, $m_2 = |\begin{smallmatrix} u_0 & u_1 \\ v_0 & v_1 \end{smallmatrix}|$, and $M = \sum_k m_k \neq 0$. A sufficient error bound for the current vertex (u_2, v_2) to avoid FP in this cell is:

$$\epsilon_r = \max_{k \in \{k | \mu_k \notin [0,1]\}} \min(\psi(m_k; \mathbf{v_2}), \psi(M - m_k; \mathbf{v_2}), \quad (3)$$

where $\psi(f; \mathbf{V})$ is the function that returns the maximal error bound allowed on $\mathbf{V}$ to keep the sign of f.

Although cpSZ can successfully preserve all the critical points, it fails to provide any guarantees on the separatrices. Fig. 1(b) depicts the topological skeleton of cpSZ on the same region as Fig. 1(a). According to the figure, cpSZ clearly

distorts several separatrices (highlighted by colors), leading to a wrong interpretation of flow behaviors in this local region.

IV. Overview

In this section, we formulate our research target, followed by an overview of the proposed framework.

A. Problem formulation

Our goal is to compress a 2D/3D vector field $\mathbf{V}$ as much as possible while preserving the topology. Let f_c and f_d denote the compression and decompression functions, respectively. Further denote $\texttt{size}(\cdot)$ as the operator to obtain the size of the data, $\texttt{cp}(\cdot)$ as the operator to extract critical points, and $\texttt{sep}(\cdot,\cdot)$ as the operator to compute separatrices. Given the acceptable tolerances τ_p for the distance between critical points and τ_t for separatrices, the generic problem can be mathematically formulated as follows:

$$\begin{aligned} &\max \frac{\texttt{size}(\mathbf{V})}{\texttt{size}(f_c(\mathbf{V}))} \\ \text{subject to:}\quad &\|\mathbf{V}-\mathbf{V}'\|_\infty \le \epsilon \\ &\|\mathcal{M}_p(\texttt{cp}(\mathbf{V}),\texttt{cp}(\mathbf{V}'))\|_\infty \le \tau_p \\ &\|\mathcal{M}_t(\texttt{sep}(\texttt{cp}(\mathbf{V}),\mathbf{V}),\texttt{sep}(\texttt{cp}(\mathbf{V}'),\mathbf{V}'))\|_\infty \le \tau_t \end{aligned}$$

where $\mathbf{V}' = f_d(f_c(\mathbf{V}))$ represent the decompressed data, and $\mathcal{M}_s$ and $\mathcal{M}_p$ are the respective metrics to measure the differences between critical points and separatrices.

Different metrics can be used to target different levels of topology preservation, and we use the following setting to preserve the original topology as much as possible with a reasonable computational cost. Specifically, we require exact matches of critical points in terms of types and positions, or equivalently $\texttt{cp}(\mathbf{V}) = \texttt{cp}(\mathbf{V}')$. This requirement ensures the same number of separatrices with the same starting points. For separatrices, we define the following metrics based on user inputs. First, we define a distant threshold ϵ_p to check if the streamline vanishes. If the distance between the current position and a sink/source is less than ϵ_p, we say the streamline vanishes at this critical point. Second, we define the maximal number of steps t to trace the streamline. This setting is necessary because closed streamlines [50] and orbits [51] may never reach a destination. We also define the step size h in the RK4 method as a constant. Third, we define a tolerance threshold τ_t to indicate the maximal allowable tolerance between the separatrices in the original data and the corresponding ones in the decompressed data. We use Fréchet distance [52], a widely used metric to assess the similarity of two trajectories, to measure this distance. These input parameters and their default values are summarized in Table II.

B. System design

We present the design of Tsp-SZ in Figure 2, which is built upon the critical-point-preserving lossy compressor cpSZ [35]. We propose to base our work on cpSZ for topological skeleton preservation because it (1) preserves all the critical points and lays a solid foundation; and (2) provides point-wise error control that is crucial for scientific data. To ensure the exact preservation of critical point positions, we revise cpSZ to apply lossless compression on cells with critical points in 2D cases (cpSZ automatically falls back to this strategy in 3D cases). This process automatically eliminates all FN and FT cases. We further propose using absolute error control in the error bound solver to eliminate FP cases, which leads to significantly better quality of the decompressed data compared with the point-wise relative error control used in cpSZ. This optimization ensures the exact correspondence between critical points in original and decompressed data with improved quality.

TABLE II
Input parameter and default values for topology preservation

Parameter	Default	Meaning
ϵ	-	Maximal allowable point-wise error on raw data.
ϵ_p	1E-3	Threshold for being absorbed by a sink/source.
t	1000	Maximal number of step in RK4.
h	0.05	Step size used in RK4.
τ_t	$\sqrt{2}$	Maximal allowable error in Fréchet distance.

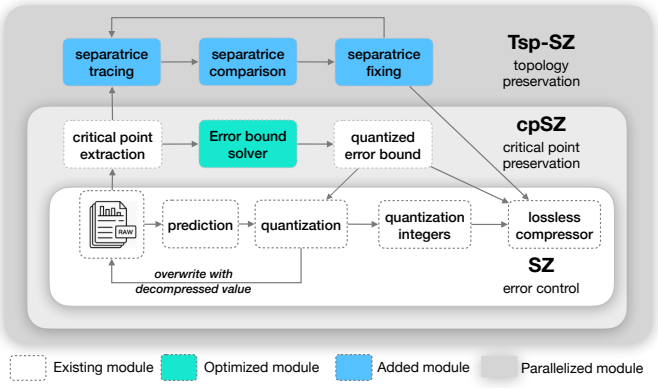


Fig. 2. Overview of the proposed framework.

We then enhance the framework for preserving separatrices using the newly added modules, which enables the preservation of the entire topological skeleton. In particular, we explore two methods for the preservation of separatrices. On the one hand, we propose an intuitive approach that losslessly encodes the cells that are passed by any separatrices. This approach generally leads to a high percentage of losslessly encoded data (and thus a relatively low compression ratio) but provides guaranteed topology preservation with fixed running time. On the other hand, we explore an iterative approach that gradually fine-tunes the decompressed data until the separatrices are fully preserved, reducing the amount of data requiring lossless compression at the cost of additional computational overhead. To this end, we carefully optimize the proposed approaches and parallelize them in shared-memory environments to achieve high throughput for topology-preserving lossy compression.

V. Preservation of Topological Skeleton

In this section, we introduce our methods to ensure the consistency of the topological skeleton between the original and decompressed data in vector fields when users specify their integration methods and parameters. As illustrated in Fig. 2,

we rely on the revised cpSZ for critical point preservation while using the proposed modules to enable the preservation of the topological skeleton. In particular, we propose two methods to fix the incorrect separatrices, including a single-pass compression method that produces exact separatrices with guaranteed runtime (TspSZ-l) and an iterative refining procedure that improves the compression ratios at the cost of compression throughput (TspSZ-i).

A. TspSZ-l: Full topology preservation with selective lossless encoding and single-pass compression

We introduce TspSZ-l for the exact preservation of topological skeleton in Algorithm 2. The key idea is to identify the cells that have been involved in the computation of separatrices and compress them losslessly. As shown in the algorithm, we first initialize a bitmap to store the vertices that require lossless encoding (line 1) and another bitmap to indicate the presence of critical points in a cell (line 2), and then extract all the critical points in the original data and mark the vertices with at least one critical point in their adjacent cells (lines 3-11). After that, we trace the separatrices using the original data, and mark all vertices that are involved in this computation as lossless (lines 12-18). To this end, we use the slightly revised cpSZ algorithm to perform error-controlled lossy compression with critical point preservation while encoding the specific set of vertices in a lossless fashion (line 19).

Algorithm 2 TspSZ-l for 3D data

Input: Vector field dataset $\mathbf{d} = \{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$, user-specified error bound ϵ, RK4 parameter set $\theta = \{\epsilon_p, t, h\}$
Output: Compressed bytes.
1: $M \leftarrow \texttt{BitMap}(n_v, 0)$ ▷ initialize bitmap for lossless vertices
2: $C \leftarrow \emptyset$ ▷ initialize a vector to store critical points
3: **for** $i \leftarrow 0$ to $n_c - 1$ **do** ▷ compute critical points in each cell
4: **if** $\texttt{check\_cp\_existence}(\mathbf{d}, i)$ **then** ▷ critical points exist in cell
5: $C.append(\texttt{extract\_cp}(\mathbf{d}, i))$ ▷ insert critical point
6: $i_0, i_1, i_2, i_3 \leftarrow \texttt{cell\_vertices}(i)$ ▷ get vertex indices in cell
7: **for** $j \leftarrow 0$ to 3 **do**
8: $M[i_j] \leftarrow 1$ ▷ mark vertices as lossless
9: **end for**
10: **end if**
11: **end for**
12: **for** $c \in C$ **do** ▷ compute and compare separatrices
13: **if** c is a saddle **then**
14: **for** $s \in S_c$ **do** ▷ iterate all starting location
15: **for** $j \in \{-1, 1\}$ **do** ▷ iterate two directions
16: **for** v_i involved in $\texttt{RK4}(\mathbf{d}, s, \theta, j)$ **do** ▷ identify the vertices that are involved in computing separatrices
17: $M[i] \leftarrow 1$ ▷ mark vertices as lossless
18: **end for**
19: **end for**
20: **end for**
21: **end if**
22: **end for**
23: $compressed \leftarrow \texttt{cpSZ\_with\_lossless\_vertices}(\mathbf{d}, \epsilon, M)$
24: **return** $compressed$

Complexity and quality analysis: Because extracting critical points (lines 3-11), computing separatrices (lines 12-22), and performing the revised cpSZ (line 23) take $\mathcal{O}(n_c)$, $\mathcal{O}(n_s t)$ (where n_s indicates the number of saddles in the original data), and $\mathcal{O}(n_c + n_v)$ time, respectively, this algorithm has a deterministic runtime of $\mathcal{O}(n_c + n_v + n_s t)$. It also produces exact separatrices when compared with those from the original data because of the lossless encoding on all related vertices. However, it may suffer from relatively low compression ratios when the separatrices span the entire domain, which leads to a large percentage of losslessly encoded vertices.

Algorithm 3 TspSZ-i for 3D data

Input: Vector field dataset $\mathbf{d} = \{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$, user-specified error bound ϵ, integration method parameter set $\theta = \{\epsilon_p, t, h\}$, and tolerance τ
Output: Compressed bytes.
1: $M \leftarrow \texttt{BitMap}(n_v, 0)$ ▷ initialize bitmap for lossless vertices
2: $C \leftarrow \emptyset$ ▷ initialize a vector to store critical points
3: **for** $i \leftarrow 0$ to $n_c - 1$ **do** ▷ compute critical points in each cell
4: **if** $\texttt{check\_cp\_existence}(\mathbf{d}, i)$ **then** ▷ critical points exist in cell
5: $C.append(\texttt{extract\_cp}(\mathbf{d}, i))$ ▷ insert critical point
6: $i_0, i_1, i_2, i_3 \leftarrow \texttt{cell\_vertices}(i)$ ▷ get vertex indices in cell
7: **for** $j \leftarrow 0$ to 3 **do**
8: $M[i_j] \leftarrow 1$ ▷ mark vertices as lossless
9: **end for**
10: **end if**
11: **end for**
12: $compressed_1, \mathbf{d}' \leftarrow \texttt{cpSZ\_with\_lossless\_vertices}(\mathbf{d}, \epsilon, M)$ ▷ compress data using cpSZ and obtain decompressed data $\mathbf{d}'$
13: $T_{\mathbf{d}}, T_{\mathbf{d}'} \leftarrow \emptyset$ ▷ record separatrices in original and decompressed data
14: $V \leftarrow \emptyset$ ▷ initialize vertices require lossless encoding
15: $Q \leftarrow \emptyset$ ▷ initialize queue for trajectories that need to fix
16: $i \leftarrow 0$ ▷ initialize index for separatrices
17: **for** $c \in C$ **do** ▷ compute and compare separatrices in **d** and **d'**
18: **if** c is a saddle **then**
19: **for** $s \in S_c$ **do** ▷ iterate all starting location
20: **for** $j \in \{-1, 1\}$ **do** ▷ iterate two directions
21: $t_d \leftarrow \texttt{RK4}(\mathbf{d}, s, \theta, j)$ ▷ compute separatrices in $\mathbf{d}$
22: $t_{d'} \leftarrow \texttt{RK4}(\mathbf{d}', s, \theta, j)$ ▷ compute separatrices in $\mathbf{d}'$
23: $T_{\mathbf{d}}.append(t_d)$, $T_{\mathbf{d}'}.append(t_{d'})$
24: **if** $\texttt{check\_traj}(t_d, t_{d'}, \tau)$ is false **then**
25: $Q.push(i)$ ▷ record incorrect separatrices
26: **end if**
27: $i \leftarrow i + 1$ ▷ increment i
28: **end for**
29: **end for**
30: **end if**
31: **end for**
32: **while** $Q \neq \emptyset$ **do** ▷ iteratively fix separatrices
33: **for** $i \in Q$ **do** ▷ fix wrong separatrices in current decompressed data
34: $\texttt{fix\_traj}(T_{\mathbf{d}}[i], T_{\mathbf{d}'}[i], \tau, \theta, \mathbf{d}', V)$ ▷ see Algorithm 4
35: **end for**
36: $Q \leftarrow \emptyset, i \leftarrow 0$ ▷ re-initialize Q and i
37: **for** $c \in C$ **do** ▷ compute and verify separatrices in **d'**
38: **if** c is a saddle **then**
39: **for** $s \in S_c$ **do** ▷ iterate all starting location
40: **for** $j \in \{-1, 1\}$ **do** ▷ iterate two directions
41: $T_{\mathbf{d}'}[i] \leftarrow \texttt{RK4}(\mathbf{d}', s, \theta, j)$ ▷ update separatrices in $\mathbf{d}'$
42: **if** $\texttt{check\_traj}(T_{\mathbf{d}}[i], T_{\mathbf{d}'}[i], \tau)$ is false **then**
43: $Q.push(i)$ ▷ record incorrect separatrices
44: **end if**
45: $i \leftarrow i + 1$ ▷ increment i
46: **end for**
47: **end if**
48: **end for**
49: **end if**
50: **end for**
51: **end while**
52: $compressed_2 \leftarrow \texttt{lossless}(V)$ ▷ compress value and index of V
53: **return** $\{compressed_1, compressed_2\}$

B. TspSZ-i: Error-controlled topology preservation with iterative correction

As data size is becoming the dominant factor for scientific data management tasks such as data transfer and I/O, high compression ratios are usually preferred, if not required.

To address the limited compression ratios of TspSZ-l, we propose TspSZ-i, an algorithm that iteratively refines the decompressed data to achieve topology preservation. Instead of using a generic compression-verification paradigm in existing works [53], we propose to operate on the decompressed data of the revised cpSZ, and record additional information that is required to preserve separatrices. This approach eliminates the repeated executions of the compression procedure in cpSZ, which turns out to dominate the runtime in 3D cases. We further optimize TspSZ-i by identifying bifurcation points to reduce the amount of data requiring lossless compression.

We present the detailed algorithm in Algorithm 3. The first few steps (lines 1-12) are very similar to TspSZ-l, where we extract the critical points and use the revised cpSZ to compress the data. After that, we obtain the decompressed data $\mathbf{d}'$ and compute separatrices in both $\mathbf{d}'$ and original data $\mathbf{d}$ (lines 13-31). During the computation, we also compare the two sets of separatrices and record the wrong ones that require further processing (lines 24-26).

If the revised cpSZ fails to preserve the topological skeleton, we will iteratively correct these separatrices until the process is complete (lines 32-50). This process involves the procedure for correcting a single separatrix (lines 33-35), which is detailed in Algorithm 4. In this algorithm, we first identify the positions in the original data where the separatrix diverges based on Euclidean distance (lines 2-4), and then iteratively correct the separatrix (lines 9-23). Specifically, we replace all the data that affect the divergence of the separatrix with the original values (lines 11-16) to ensure that both trajectories are identical before the divergence point. Next, we recompute the separatrix on the decompressed data and compare it with the separatrix from the original data (lines 17-18). If the result does not meet the required criteria, we increment the index by 1 and repeat the process. In the worst case, the entire separatrix will be losslessly compressed. This method enables the preservation of topological features by losslessly compressing parts of the data. The compression ratio is significantly improved since only portions of the separatrix are losslessly compressed.

After fixing the incorrect separatrices in Algorithm 3 (lines 33-35), we compute the new separatrices based on the updated data and compare them with those from the original data (lines 36-49). This procedure continues until no wrong separatrices are found in the current decompressed data. This process guarantees the preservation of vector field topology under user-specified requirements. To this end, we will losslessly compress the data and indices of V, and concatenate the results to the compressed data produced by cpSZ (line 12) to generate the final compressed data. In practice, we merge the positions of indices to the bitmap M (line 1) for optimized storage cost.

Complexity and quality analysis: The first two parts of the algorithm (lines 1-31) yield almost the same runtime as TspSZ-l, namely $\mathcal{O}(n_c+n_v+n_st)$. The iterative correction of separatrices may take $\mathcal{O}(n_in_st)$ time, where n_i is the number of iterations. This algorithm is guaranteed to converge because, in the worst case, all the data can be losslessly compressed in a fixed number of iterations. In practice, we observe the number of iterations is usually less than 5 for smooth data and less than 10 for turbulent data. Compared with TspSZ-l, it has a longer running time because of the iterative procedure, but it usually delivers better compression ratios due to a smaller percentage of losslessly compressed vertices.

Algorithm 4 Separatrix Correction

Input: i^{th} separatrix for $\mathbf{d}$, i^{th} separatrix for $\mathbf{d}'$, integration parameter $\theta = \{\epsilon_p, t, h\}$, tolerance τ, decompressed data $\mathbf{d}'$, and lossless vertex set V

1: index $\leftarrow 0$
2: flag $\leftarrow false$
3: **for** $i \leftarrow 0$ to len($T_{\mathbf{d}}[i]$) **do** ▷ find the diverging vertex
4: **if** distance(p_1, p_2) $\geq \tau$ **then** ▷ check vertices in the two separatrices
5: index $\leftarrow i$
6: break
7: **end if**
8: **end for**
9: $\theta' = \{\epsilon_p, t, h\}$
10: **while** flag is false **do**
11: $P \leftarrow T_{\mathbf{d}}[i]$
12: $Q \leftarrow \texttt{RK4}(\mathbf{d}', c, \theta')$
13: **for** v_i involved in Q **do**
14: $\mathbf{d}'[v] \leftarrow \mathbf{d}[v]$ ▷ replace original data for all vertices related to the separatrix
15: $V.append(v_i)$ ▷ record lossless vertices
16: **end for**
17: $Q' \leftarrow \texttt{RK4}(\mathbf{d}', c, \theta)$
18: **if** $\texttt{check\_traj}(P, Q', \tau)$ is true **then**
19: **return**
20: **else**
21: $\theta' = \{\epsilon_p, \text{index} + 1, h\}$ ▷ update parameter to proceed
22: **end if**
23: **end while**

VI. Optimizing critical point preservation with absolute error control

As mentioned in Section II, cpSZ uses point-wise relative error bound for error control (i.e., ensuring $|\frac{x_i - x_i'}{x_i}| < \epsilon_r$ for a point-wise relative error bound ϵ_r and any data point x_i) in the compression procedure. Although approach this leads to relatively straightforward error bound derivation, it produces lower-quality decompressed data in terms of mean squared errors (MSE) when compared with absolute error control (i.e., ensuring $|x_i - x_i'| < \epsilon_a$ for an absolute error bound ϵ_a and any data point x_i) according to existing studies [54]. In this section, we carefully analyze the impact of using point-wise relative error control for topology-preserving data compression, and propose to optimize the efficiency of cpSZ by utilizing absolute error control for critical point preservation. This significantly reduces the MSE of the decompressed data under the same compression ratios and thus benefits topology preservation due to a reduced number of wrong separatrices and reduced errors in the numerical integration.

A. Analysis of error accumulation in computing separatrices

When the vector field is compressed in a lossy fashion, errors will be introduced in each component. We analyze how such errors could accumulate and impact the results for separatrices using 2D cases as an example. We perform the analysis using absolute error bounds on each data point, followed by a discussion of the situations when point-wise relative error

bounds are used. For any data point inside a linear cell, its vector components u' and v' in the decompressed data are interpolated using the formula below:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u_0 + \xi_{u0} & u_1 + \xi_{u1} & u_2 + \xi_{u2} \\ v_0 + \xi_{v0} & v_1 + \xi_{v1} & v_2 + \xi_{v2} \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \end{bmatrix}, \quad (4)$$

where $\{\xi_{ui}\}$ and $\{\xi_{vi}\}$ are the absolute errors introduced in each node, which are subject to $|\xi_{ui}| \leq \epsilon_a$ and $|\xi_{vi}| \leq \epsilon_a$ under the given absolute error bound ϵ_a. This implies that the resulting errors in the two components are $\sum_i \xi_{ui}\mu_i$ and $\sum_i \xi_{vi}\mu_i$, respectively. Since the intermediate terms (i.e., k_1, k_2, k_3, k_4) in RK4 are all interpolated in this way, the final error in a single step of RK4 is the weighted average of the errors in the four terms according to Eq. (1). As separatrices tend to spread in the entire data domain, lower average errors in the raw data usually indicate lower errors in the separatrices.

The same analysis applies to the case of point-wise relative error bounds, where the absolute error bound in each data point turns out to be the multiplication of the error bound and the data values. Meanwhile, existing studies [54] have demonstrated that point-wise relative error control yields much higher mean-squared errors than absolute error control under the same compression ratio, which also indicates much higher average errors in usual cases. This inspires us to seek absolute error control for full topology preservation with better efficiency.

One can also estimate a pessimistic upper bound for the separatrices when a global absolute error bound ϵ_a is used. In particular, the upper bounds of the errors in u' and v' are $|\sum_i \xi_{ui}\mu_i| \leq |\sum_i \epsilon_a\mu_i| = \epsilon_a$ and $|\sum_i \xi_{vi}\mu_i| \leq |\sum_i \epsilon_a\mu_i| = \epsilon_a$. This implies a maximal error of ϵ_a for k_1, k_2, k_3, k_4, which leads to an upper bound of $\frac{h}{6}(\epsilon_a + 2\epsilon_a + 2\epsilon_a + \epsilon_a) = h\epsilon_a$ for the final error in a single step of RK4 according to Eq. (1). Thus, the upper bound of the integration error can be expressed as $th\epsilon_a$, where t is the number of steps taken in the integration. Although these bounds could be used as an alternative way to determine an absolute error bound for topology-preserving compression, it usually leads to over-preservation in certain regions due to the pessimistic estimation.

B. Enabling absolute error control with cpSZ

To address the limitations mentioned above, we propose to revise the error bound solver in cpSZ to enable absolute error control. This optimization is done by adjusting Eq. (3) in Theorem 1 to solve an absolute error bound instead of the current point-wise relative error bound. In the following text, we first prove a lemma that derives a sufficient absolute error bound to keep the sign of a specific function and then apply it in the context of avoiding FP cases in cpSZ. We use 2D vector fields as an example for demonstration purposes, and this easily generalizes to 3D cases.

Lemma 1: If $|A| + |B| \neq 0$, the maximal absolute error bound ϵ to keep the sign of $A * \xi_1 + B * \xi_2 + C$ for any $\xi_1, \xi_2 \in [-\epsilon, \epsilon]$ is $\frac{|C|}{|A|+|B|}$.

Proof: Assuming $C \geq 0$, we have $A*\xi_1 + B*\xi_2 + C \geq 0$. This requires $C \geq -(A*\xi_1 + B*\xi_2)$ for any $\xi_1, \xi_2 \in [-\epsilon, \epsilon]$. As $-(A*\xi_1 + B*\xi_2) \leq |-(A*\xi_1 + B*\xi_2)| \leq |A*\xi_1| + |B*\xi_2| \leq |A|\epsilon + |B|\epsilon$, we have $C \geq |A|\epsilon + |B|\epsilon$ and thus $\epsilon \leq \frac{|C|}{|A|+|B|}$. The proof holds when $C < 0$, which completes the proof. ■

Then, we use this lemma to solve the absolute error bound that eliminates FP in the decompressed data. Since this derivation is only needed when no critical point is present in a cell, there exists at least one k such that $\mu_k \notin [0, 1]$. Without loss of generality, we assume $\mu_0 \notin [0, 1]$. According to Theorem 1, it requires that m_0 and $M - m_0 = m_1 + m_2$ have the same sign in the original and decompressed data. Assuming that an absolute error bound ϵ is used to compress the current vertices (both u_2 and v_2), this assumption will introduce the errors of $\xi_u, \xi_v \in [-\epsilon, \epsilon]$. For the former, we know that $m_0 = \left|\begin{smallmatrix} u_1 & u_2 \\ v_1 & v_2 \end{smallmatrix}\right| = u_1v_2 - u_2v_1$; and this problem reduces to preserve the sign of $u_1(v_2 + \xi_v) - (u_2 + \xi_u)v_1$, which can be directly solved using Lemma 1. Similarly, the latter reduces to the sign preservation of $u_0(v_2+\xi_v) - (u_2+\xi_u)v_0 + u_0v_1 - u_1v_0$, where the error bound can be solved in the same way. Then the final error bound for u_2 and v_2 that ensures no FP in this cell is $\min(\frac{|m_0|}{|u_1|+|v_1|}, \frac{|m_1+m_2|}{|u_0|+|v_0|})$. This approach also applies when $k = 1$ and $k = 2$, which can be solved using Lemma 1 with a slightly different analytical form.

We evaluate the effectiveness of the absolute error control in terms of data quality using the 2D ocean data and present the comparison with cpSZ under similar compression ratios in Fig. 3. In the figure, one can clearly see the large error magnitudes caused by the point-wise relative error control in cpSZ. In contrast, the proposed revision with absolute error control demonstrates small average errors both quantitatively and qualitatively. This optimization significantly reduces the number of incorrect separatrices after cpSZ, as will be detailed in Section VIII.D.

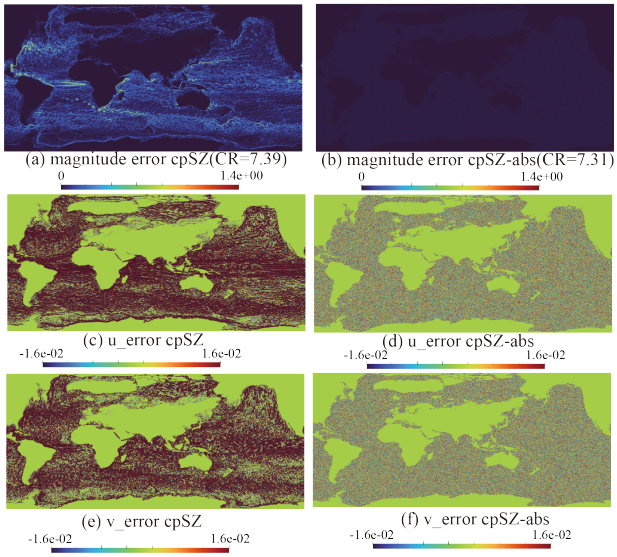


Fig. 3. Visualization of errors in Ocean data after applying cpSZ with point-wise relative and absolute error bounds under similar compression ratios (CR).

VII. Parallelization

Due to the computationally intensive nature of the algorithms, we parallelize the proposed framework on shared-memory environments with openMP [55]. Note that we parallelize both cpSZ and the added modules in TspSZ to ensure high end-to-end efficiency.

We first present our parallelization strategy for cpSZ. The key idea is to serialize the dependency in the predication stage. Specifically, cpSZ has strong dependencies in neighboring vertices, which limits the use of traditional embarrassing parallelization strategies. To address this issue, we adopt a multi-stage algorithm inspired by the parallelization strategy in [36]. Specifically, we first partition the data into independent blocks, with each thread processing only the internal data of a block (i.e., skipping the data that form surfaces and edges with other blocks, as these data points are interdependent during prediction). After processing the internal data of the blocks, we degrade the 3D Lorenzo predictor into 2D Lorenzo for handling surfaces (similarly, for edges, we reduce it to 1D Lorenzo). We first process all planes perpendicular to the x-axis in parallel, followed by those perpendicular to the y-axis, and finally those perpendicular to the z-axis. After processing the planes, we adopt a similar strategy to handle the edges. Finally, we perform lossless processing on the data points located at the intersections of the three planes.

For the numerical integration steps, we adopt an embarrassingly parallel approach for high efficiency. Since there are no dependencies between threads, the parallel efficiency is close to 100%, according to our experiments. We further parallelized the module responsible for correcting the trajectories (Algorithm 4) via speculated execution. In particular, we allow the threads to read decompressed data $\mathbf{d}'$ from shared memory and compute the separatrices in parallel. Although this approach involves the use of stale data and may lead to discrepancies in the intermediate topological skeleton, it will not impact the final result because we always verify the final result after the correction (lines 36-49 in Algorithm 4).

VIII. Evaluation

We evaluate our methods using four real-world datasets from real-world applications and compare them with state-of-the-art lossy and lossless compressors, including cpSZ [35], ZSTD [11], and GZIP [10]. We present quantitative evaluation in terms of Peak Signal-to-Noise Ratio (PSNR), the number of incorrect separatrices, and Fréchet distance, as well as qualitative evaluation using visualization. Throughout this section, we use CR to present the compression ratio and T_c and T_d to represent compression and decompression time, respectively.

A. Experimental setup

We conduct our experiments using four scientific datasets listed in Table III, where n_d represents the number of dimensions, n_v stands for the numbers of vertices, n_{cp}, n_s and n_{sep} denote the number of critical points, saddles, and separatrices, respectively. All datasets are stored as single-precision floating points with the following detailed information.

- **Ocean**: A simulated dataset representing ocean currents.
- **CBA:** Simulation of a 2D flow generated by a heated cylinder, using Boussinesq approximation [56], [57].
- **Hurricane:** A simulation of Hurricane-ISABEL from the National Center for Atmospheric Research [58].
- **Nek5000:** A fluid simulation generated by Nek5000 [59].

Topological skeletons are used in these datasets to extract and characterize features for insight generation. In the Ocean dataset, they identify major flow structures, track eddies, and examine water mass transport. In computational fluid dynamics datasets like CBA and Nek5000, they partition complex flow regions, detect transient phenomena such as vortex shedding, and enhance flow control strategies. For the Hurricane dataset, they characterize key storm structures (e.g., the eye, eyewall, and rainbands), analyze hurricane evolution, and assess the impact of large-scale environmental flow patterns.

TABLE III
Benchmark datasets

Dataset	n_d	n_v	n_{cp}	n_s	n_{sep}	Size
CBA	2	450×150	78	38	152	0.51MB
Ocean	2	3600×2400	20938	10376	41504	65.92 MB
Hurricane	3	$100 \times 500 \times 500$	1026	833	4998	286.10MB
Nek5000	3	$512 \times 512 \times 512$	10587	9145	54870	1.50GB

All our evaluations are performed on a high-performance cluster [60], where each compute node contains 2 AMD EPYC ROME 7702P processors with 128 cores and 512 GB memory in total. T_c and T_d are reported with 128 threads unless specifically noted.

B. Evaluation Metrics

Our objective is to preserve all critical points under lossy compression while ensuring that separatrices are retained to varying degrees. Therefore, the following metrics are particularly important:

1) **Compression Ratio (CR):** The Compression Ratio is defined as $\text{CR} = \text{Size}_{\text{original}}/\text{Size}_{\text{compressed}}$. Alternatively, we use bitrate to represent CR in rate-distortion graphs, which is computed as 32 (number of bits to represent one single-precision number) divided by the compression ratio in this section.
2) **Peak Signal-to-Noise Ratio (PSNR):** PSNR measures the quality of a compressed or reconstructed signal compared to the original, and it is calculated based on the Mean Squared Error (MSE) between the original and decompressed data. Specifically, it is defined as $\text{PSNR} = 20 \cdot \log_{10}(\text{data range}) - 10 \cdot \log_{10}(\text{MSE})$.
3) **Number of Incorrect Separatrices (IS):** We consider a separatrix to be incorrect if the separatrix from the original data and that from the decompressed data result in different topological structures, including cases where they end at different critical points or when the Fréchet distance between the two trajectories exceeds the predefined threshold τ_t.
4) **Fréchet Distance:** The Fréchet Distance helps evaluate how well the trajectories are preserved after compression, offering insights into the topological distortions introduced. Given two discrete curves $P = (p_1, p_2, \ldots, p_n)$ and $Q = (q_1, q_2, \ldots, q_m)$, composed of the point sets P and Q, the discrete Fréchet distance is defined as $d_F(P, Q) = \min_{\sigma,\tau} \max_k \|p_{\sigma(k)} - q_{\tau(k)}\|$, where σ and τ are the valid reparameterization that mapping points along curve P and Q, respectively. We use the minimum,

maximum, median, and standard deviation of Fréchet Distance across all the trajectories to measure how close the original trajectories are to decompressed trajectories.

C. Rate-distortion

We first assess the quality of our decompressed data using the widely used rate-distortion curves in Fig. 4. The x-axis in this curve represents the bit-rate, and the y-axis represents PSNR. According to the figure, it is observed that the proposed absolute error control yields much higher PSNR when compared to the point-wise relative error control at the same compression ratios for both cpSZ and TspSZ. Compared with TspSZ-l, TspSZ-i provides better compression ratios due to the reduced number of losslessly encoded vertices, and this difference seems bigger when the data is harder to compress (Ocean and Nek5000). While cpSZ generally provides the best compression ratios, it cannot preserve separatrices, as will be detailed below.

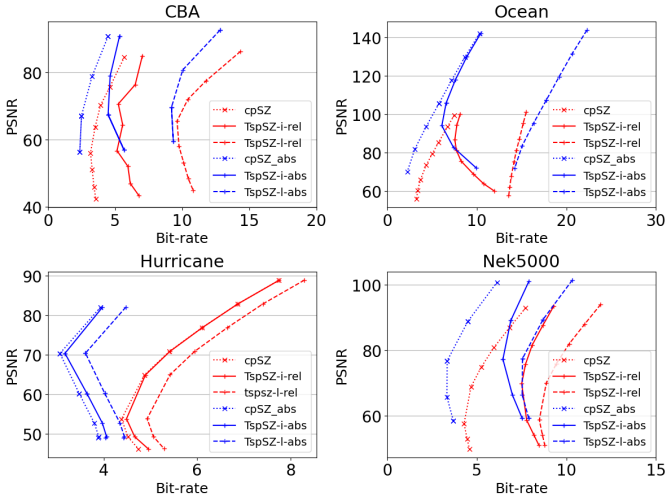


Fig. 4. Rate-distortion under different datasets

D. Topology preservation

We present the quantitative results for all four datasets in Table IV - VII. Since no existing lossy compressors are capable of preserving separatrices with error control, we compare our methods with lossless compressors in terms of benefits in improving compression ratios. We also present the results for cpSZ, TspSZ-l, and TspSZ-i with both point-wise relative error control and absolute error control (-abs). We further include a comparison with cpSZ-sos [36], a variant of cpSZ that preserves critical points extracted by the Simulation of Simplicity (SoS) [46] method. Note that such differences may lead to discrepancies in the number of detected critical points when compared to numerical methods. Since cpSZ-sos does not have a parallel implementation in a shared memory environment, we report the compression and decompression times in a serial execution setting. We use the most relaxed setting "ST4" in cpSZ-sos, as it leads to the best compression ratio in general. Note that we adjust the error bounds for the two error control modes and cpSZ-sos to achieve similar compression ratios for a fair comparison.

According to the tables, both cpSZ and cpSZ-sos fail to preserve the separatrices, as evidenced by their large maximal Fréchet distance. Meanwhile, TspSZ methods preserve the topological skeleton much better, and they yield much higher compression ratios than lossless compressors because they compress the data in a lossy fashion. In particular, TspSZ-l methods preserve the exact topology and yield $3 \sim 6\times$ compression ratios on the CBA and Hurricane datasets because they are relatively smooth. This gain reduces to $1.2 \sim 2.7\times$ in more turbulent datasets (Ocean and Nek5000). TspSZ-i methods produce consistently better compression ratios than TspSZ-l at the cost of slower compression speed and slightly altered separatrices. It leads to up to 50% improvement in the compression ratios, especially in the turbulent datasets. Meanwhile, absolute error control exhibits higher compression ratios than point-wise relative error control in most cases, demonstrating the benefits of improved data quality.

TABLE IV
QUANTITATIVE RESULTS ON 2D CBA DATA

Compressor	Setting	CR	PSNR	#IS	Fréchet Distance			T_c	T_d
					Max	Mean	Std		
ZSTD	/	1.09	/	0	0	0	0	0.01	0.01
GZIP		1.11	/	0	0	0	0	0.03	0.01
cpSZ-sos	ϵ=5E-6,h=1,τ=$\frac{1}{2}$, t=3E3	4.15	111.03	4	1.63	0.17	0.26	0.16	0.01
cpSZ	ϵ=5E-2,h=1,τ=$\frac{1}{2}$, t=3E3	4.99	56.20	35	77.93	25.09	55.37	0.14	0.04
TspSZ-l		3.28	57.96	0	0	0	0	0.11	0.02
TspSZ-i		3.61	57.08	0	0.41	0.02	0.08	3.17	0.02
cpSZ-abs	ϵ=5E-4,h=1,τ=$\frac{1}{2}$, t=3E3	4.80	79.01	17	116.70	4.34	17.00	0.15	0.04
TspSZ-l-abs		3.19	80.77	0	0	0	0	0.11	0.02
TspSZ-i-abs		4.06	79.18	0	0.433	0.06	0.10	2.18	0.02

TABLE V
QUANTITATIVE RESULTS ON 2D OCEAN DATA

Compressor	Setting	CR	PSNR	#IS	Fréchet Distance			T_c	T_d
					Max	Mean	Std		
ZSTD	/	1.60	/	0	0	0	0	0.2	0.1
GZIP		1.59	/	0	0	0	0	2.054	0.45
cpSZ-sos	ϵ=1E-5,h=2.5E-2	5.12	110.59	1565	280.57	0.39	5.33	21.55	1.55
cpSZ	ϵ=2E-2,h=2.5E-2	6.64	73.36	912	525.37	1.54	11.54	0.70	0.11
TspSZ-l		1.91	75.12	0	0	0	0	8.17	0.12
TspSZ-i		3.64	75.58	0	1.41	0.16	0.23	324.44	0.17
cpSZ-abs	ϵ=5E-2,h=2.5E-2	7.00	93.60	337	287.40	0.62	5.53	0.67	0.11
TspSZ-l-abs		1.93	95.36	0	0	0	0	8.09	0.13
TspSZ-i-abs		5.03	94.00	0	1.41	0.16	0.22	260.57	0.17

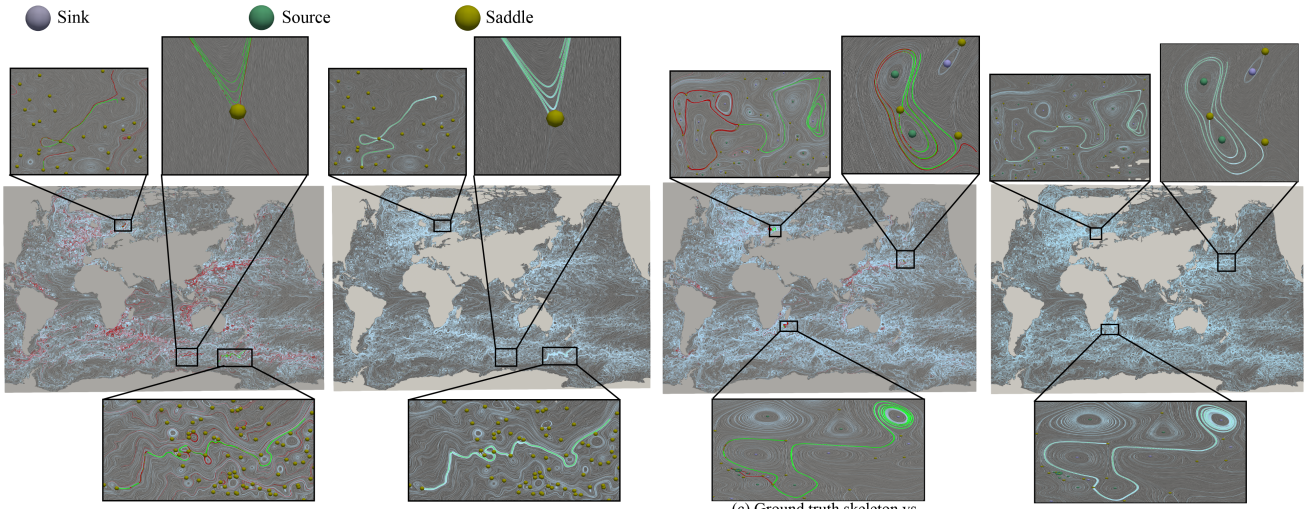
TABLE VI
QUANTITATIVE RESULTS ON 3D HURRICANE DATA

Compressor	Setting	CR	PSNR	#IS	Fréchet Distance			T_c	T_d
					Max	Mean	Std		
ZSTD	/	1.10	/	0	0	0	0	1.00	0.01
GZIP		1.11	/	0	0	0	0	13.85	2.53
cpSZ-sos	ϵ= 3E-5	6.93	87.84	329	444.90	5.72	32.01	348.89	5.44
cpSZ	ϵ=5E-2	7.13	53.81	34	233.37	0.55	4.55	2.43	0.36
TspSZ-l		6.35	53.88	0	0	0	0	12.43	0.45
TspSZ-i		6.97	53.82	0	1.38	0.15	0.23	72.36	0.42
cpSZ-abs	ϵ= 1E-2	7.82	81.91	19	135.44	0.34	3.14	1.88	0.17
TspSZ-l-abs		6.90	81.99	0	0	0	0	12.24	0.26
TspSZ-i-abs		7.74	81.92	0	1.41	0.10	0.18	45.89	0.34

TABLE VII
QUANTITATIVE RESULTS ON 3D NEK5000 DATA

Compressor	Setting	CR	PSNR	#IS	Fréchet Distance			T_c	T_d
					Max	Mean	Std		
ZSTD	/	1.09	/	0	0	0	0	5.7	3.43
GZIP		1.10	/	0	0	0	0	72.04	12.18
cpSZ-sos	ϵ=1E-5, h=0.025	4.69	106.32	8074	621.65	10.59	39.21	1911.32	30.77
cpSZ	ϵ=5E-2, h=0.025	7.33	57.62	14576	634.61	19.67	51.64	13.46	1.92
TspSZ-l		3.74	58.77	0	0	0	0	122.84	3.12
TspSZ-i		4.08	58.69	0	1.41	0.13	0.26	1051.57	3.07
cpSZ-abs	ϵ=5E-2, h=0.025	6.92	88.73	6843	531.25	8.33	31.58	10.41	0.87
TspSZ-l-abs		3.04	101.40	0	0	0	0	120.07	1.80
TspSZ-i-abs		4.56	89.08	0	1.41	0.14	0.24	979.31	1.59

We also present qualitative visualization results for the two turbulent datasets, namely Ocean in Fig. 5 and Nek5000 in Fig.7. The light blue lines in (a) and (c) represent the ground truth topological skeleton, and those in (b) and (d) stand for the topological skeletons computed from the decompressed data in TspSZ-i and TspSZ-i-abs, respectively. According to their comparison, our TspSZ methods faithfully preserve the topological skeleton under relatively high compression ratios, demonstrating its benefits over lossless methods. We



(a) Ground truth skeleton vs. incorrect separatrices under cpSZ (#IS = 1137, CR = 7.39)

(b) TspSZ-i skeleton vs. ground truth separatrices (CR = 3.67)

(c) Ground truth skeleton vs. incorrect separatrices under cpSZ-abs (#IS = 309, CR = 7.31)

(d) TspSZ-i-abs skeleton vs. ground truth separatrices (CR = 5.23)

Fig. 5. The topological skeleton in Ocean data with surface line integral convolution (LIC) visualized as context. For (a) and (c), the light blue trajectories represent the ground truth topological skeleton under the two error control modes, respectively; for (b) and (d), the light blue trajectories represent the topological skeleton under TspSZ-i and TspSZ-i-abs, respectively. In (a) and (c), the red paths indicate the incorrect separatrices in cpSZ/cpSZ-abs, and the green paths highlight their corresponding ground truth. In (b) and (d), the highlighted green paths overlap with the topological skeleton, indicating the successful corrections of the incorrect separatrices.

also depict the incorrect separatrices in (a) and (c) using red lines, and highlight their corresponding ground truth in green. It is observed that cpSZ/cpSZ-abs leads to numerous incorrect separatrices and, thus, wrong vector field topology, and those discrepancies can be corrected by the proposed TspSZ framework.

(a) Data points preserved under rel-eb mode (b) Data points preserved under abs-eb mode

Fig. 6. Data points preserved losslessly

We further visualize the lossless vertices in cpSZ and TspSZ in Fig.6. In particular, the lossless vertices required by cpSZ/cpSZ-abs are shown in yellow, and those produced by TspSZ-i/TspSZ-i-abs are depicted in green; the other nodes, which are colored in pink, are lossily compressed. This figure indicates that TspSZ-i methods only lossless encode a small portion of data, and the proposed absolute error control yields a further smaller percentage when compared with the original point-wise relative error control in cpSZ. This is the key reason for the relatively high compression ratios in TspSZ-i-abs.

E. Scalability

We then present the scalability of the compression and decompression procedure in TspSZ with the increasing number of threads in Fig. 8. Since 2D data compression and decompression are extremely fast, we report results only for 3D datasets. In particular, we compare the five methods in this evaluation: cpSZ and TspSZ with both point-wise relative error control and the proposed absolute error control, as well as the vanilla SZ3 as baseline. It is observed that TspSZ achieves much higher parallel efficiency than that of cpSZ and SZ3 for compression, because it eliminates several serial dependencies for high scalability. The decompression scalability is not high, though, which is mainly caused by the extremely fast decompression time: under such circumstances, the overhead for enabling multithreads (e.g., context switch) becomes one performance bottleneck and limits the scalability.

F. Impact of varying configurations

As discussed in Section VI, the computation of separatrices could be affected by multiple variables. In this section, we examined the impact of various parameters to investigate these effects. The results on varying values of t, h and τ are presented in Table VIII.

According to the table, as the maximal RK4 step t increases, the compression ratio decreases while the compression time significantly increases due to continued integration, which causes some incomplete trajectories to eventually reach non-saddle points or orbits, leading to error accumulation and an increased number of trajectories requiring correction.

The trend for changing the integration step size h is similar to that of n. As the integration step size increases, the separatrix length grows, meaning it passes through more cells, which in turn leads to a greater number of cells being affected. As for the maximal allowable error tolerance τ in the Fréchet distance of the separatrices, the compression ratio of TspSZ increases with the threshold, as smaller thresholds impose stricter topological preservation requirements. Similarly, the compression time will also increase with the thread because of the increasing number of separatrices to be corrected. Note that the decompression time remains almost the same for all the parameters, as it is only slightly impacted by the decoding time that is relevant to the compression ratios.

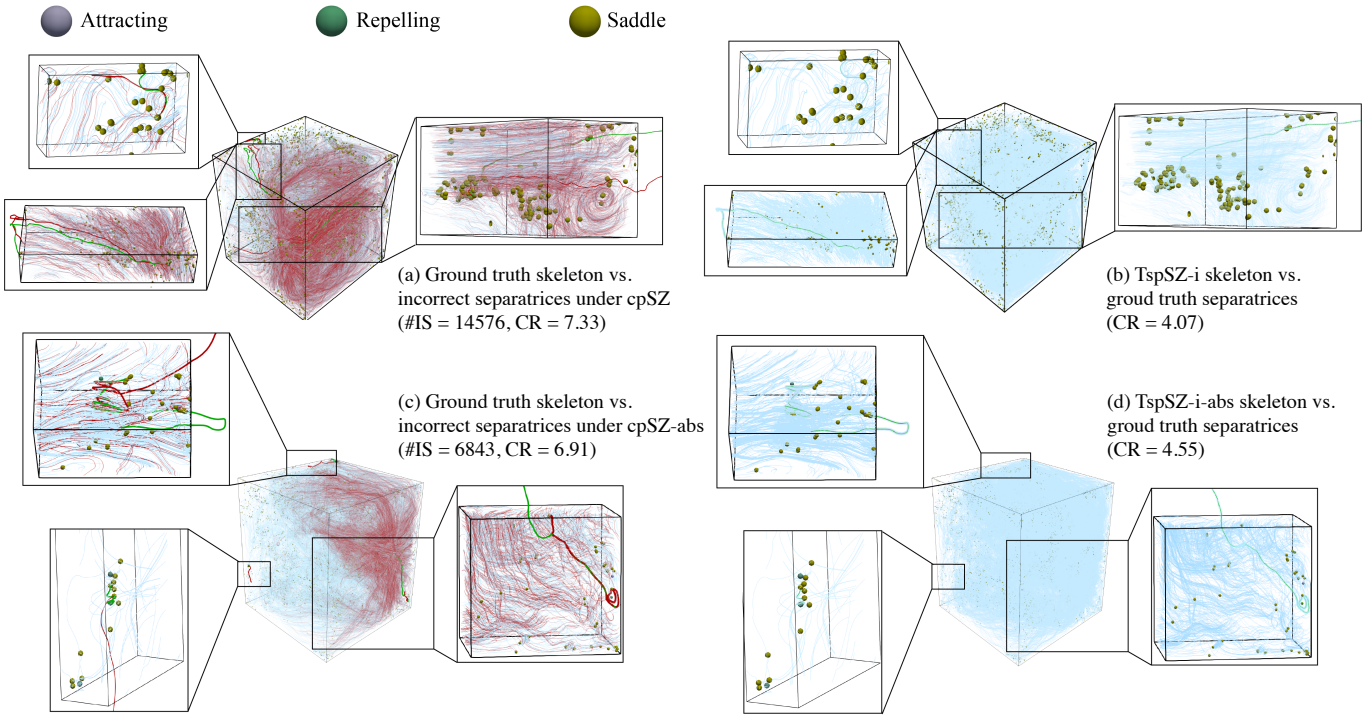


Fig. 7. The topological skeleton in Nek5000 data. The notations and styles are the same as those in Fig. 5.

TABLE VIII
IMPACT OF RK-4 STEP t, STEP SIZE h, AND FRÉCHET DISTANCE THRESHOLD τ

Metric	Maximal RK-4 Step t				RK-4 Step Size h				Fréchet Distance Threshold τ			
	$t = 500$	$t = 1000$	$t = 1500$	$t = 2000$	$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.01$	$\tau = 5$	$\tau = 3$	$\tau = \sqrt{2}$	$\tau = 1$
CR	6.41	5.03	4.15	3.65	2.35	3.18	5.03	6.78	6.10	5.78	5.03	4.59
T_c	71.57	260.57	569.58	1196.07	293.18	318.11	260.57	197.80	197.23	197.54	260.57	325.40
T_d	0.14	0.15	0.16	0.17	0.19	0.17	0.15	0.13	0.14	0.14	0.15	0.15

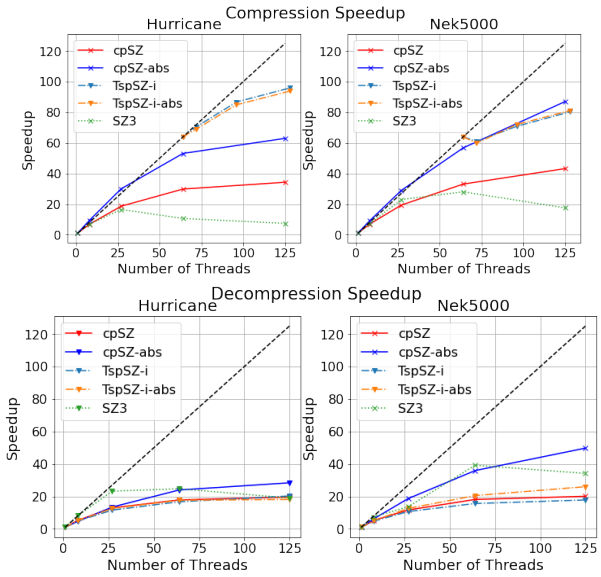


Fig. 8. Speedup in compression and decompression (black dashed line indicates ideal linear speedup).

IX. CONCLUSION

In this paper, we propose TspSZ, which significantly extends the cpSZ framework for vector field topology preservation. We propose two algorithms to enable the preservation of separatrices, and optimize the entire framework to achieve high compression ratios and throughput through careful derivation of absolute error control and tailored parallelization. Experimental results on four real-world scientific datasets demonstrate that TspSZ yields up to 7.7× with preserved topological skeletons, which is 7× higher than existing lossless compressors. Despite its relatively low compression throughput, TspSZ features high decompression throughput to facilitate its use in scientific applications because scientific data is usually compressed once and decompressed multiple times for diverse use cases. In the future, we will investigate how to further improve the efficiency of TspSZ in terms of compression ratios and compression throughput.

ACKNOWLEDGMENT

This work was partially supported by grants from NSF OAC-2330367, OAC-2311756, OAC-2313122, OAC-2313123, and OAC-2313124. We would like to thank the University of Kentucky Center for Computational Sciences and Information Technology Services Research Computing for its support and use of the Lipscomb Compute Cluster, Morgan Compute Cluster, and associated research computing resources.

REFERENCES

[1] W. P. Cockshott, D. McGregor, and J. Wilson, "High-performance operations using a compressed database architecture," *The Computer Journal*, vol. 41, no. 5, pp. 283–296, 1998.

[2] C.-C. Chang, J.-C. Chuang, and Y.-S. Hu, "Retrieving digital images from a jpeg compressed image database," *Image and Vision Computing*, vol. 22, no. 6, pp. 471–484, 2004.

[3] L. Deri, S. Mainardi, and F. Fusco, "tsdb: A compressed database for time series," in *International Workshop on Traffic Monitoring and Analysis*. Springer, 2012, pp. 143–156.

[4] C. Taskiran, J.-Y. Chen, A. Albiol, L. Torres, C. A. Bouman, and E. J. Delp, "Vibe: A compressed video database structured for active browsing and search," *IEEE Transactions on Multimedia*, vol. 6, no. 1, pp. 103–118, 2004.

[5] T. Westmann, D. Kossmann, S. Helmer, and G. Moerkotte, "The implementation and performance of compressed databases," *ACM Sigmod Record*, vol. 29, no. 3, pp. 55–67, 2000.

[6] Z. Chen, J. Gehrke, and F. Korn, "Query optimization in compressed database systems," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 271–282.

[7] A. Arion, A. Bonifati, I. Manolescu, and A. Pugliese, "Xquec: A query-conscious compressed xml database," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 2, pp. 10–es, 2007.

[8] F. Zhang, Z. Pan, Y. Zhou, J. Zhai, X. Shen, O. Mutlu, and X. Du, "G-tadoc: Enabling efficient gpu-based text analytics without decompression," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1679–1690.

[9] F. Zhang, J. Zhai, X. Shen, O. Mutlu, and W. Chen, "Efficient document analytics on compressed data: Method, challenges, algorithms, insights," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1522–1535, 2018.

[10] J. loup Gailly and M. Adler, *gzip (GNU zip)*, Free Software Foundation, 2023, accessed: January 24, 2025. [Online]. Available: https://www.gnu.org/software/gzip/

[11] Y. Collet, "Zstandard - real-time data compression algorithm," http://facebook.github.io/zstd/, online.

[12] Blosc Development Team. (2009-2023) A fast, compressed and persistent data store library. Https://blosc.org.

[13] G. K. Wallace, "The jpeg still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1992.

[14] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.

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

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

[17] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao *et al.*, "Sz3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Transactions on Big Data*, 2022.

[18] P. Lindstrom and M. Isenburg, "Fast and efficient compression of floating-point data," *IEEE transactions on visualization and computer graphics*, vol. 12, no. 5, pp. 1245–1250, 2006.

[19] P. Lindstrom, "Fixed-rate compressed floating-point arrays," *IEEE transactions on visualization and computer graphics*, vol. 20, no. 12, pp. 2674–2683, 2014.

[20] M. Ainsworth, O. Tugluk, B. Whitney, and S. Klasky, "Multilevel techniques for compression and reduction of scientific data—the multivariate case," *SIAM Journal on Scientific Computing*, vol. 41, no. 2, pp. A1278–A1303, 2019.

[21] S. Li, P. Lindstrom, and J. Clyne, "Lossy scientific data compression with sperr," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2023, pp. 1007–1017.

[22] J. Liu, S. Di, K. Zhao, X. Liang, S. Jin, Z. Jian, J. Huang, S. Wu, Z. Chen, and F. Cappello, "High-performance effective scientific error-bounded lossy compression with auto-tuned multi-component interpolation," *Proc. ACM Manag. Data*, vol. 2, no. 1, Mar. 2024. [Online]. Available: https://doi.org/10.1145/3639259

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

[24] P.-T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, and J. Bell, "Interactive exploration and analysis of large-scale simulations using topology-based data segmentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 9, pp. 1307–1324, 2011.

[25] T. Sousbie, C. Pichon, and H. Kawahara, "The persistent cosmic web and its filamentary structure — ii. illustrations," *Monthly Notices of the Royal Astronomical Society*, vol. 414, no. 1, pp. 384–403, 2011. [Online]. Available: https://doi.org/10.1111/j.1365-2966.2011.18395.x

[26] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah, "An exploration framework to identify and track movement of cloud systems," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2896–2905, 2013.

[27] T. Agarwal, A. Chattopadhyay, and V. Natarajan, "Topological feature search in time-varying multifield data," in *Topological Methods in Visualization: Theory, Software and Applications*. Springer-Verlag, 2021.

[28] J. L. Helman and L. Hesselink, "Visualizing vector field topology in fluid flows," *IEEE Computer Graphics and Applications*, vol. 11, no. 3, pp. 36–46, 1991.

[29] H. Theisel, C. Rössl, and T. Weinkauf, "Topological representations of vector fields," in *Shape Analysis and Structuring*. Springer, 2008, pp. 215–240.

[30] V. Mazzi, U. Morbiducci, K. Calò, G. De Nisco, M. Lodi Rizzini, E. Torta, G. C. A. Caridi, C. Chiastra, and D. Gallo, "Wall shear stress topological skeleton analysis in cardiovascular flows: Methods and applications," *Mathematics*, vol. 9, no. 7, p. 720, 2021.

[31] T. Wischgoll, "Computing center-lines: an application of vector field topology," in *Topology-Based Methods in Visualization II*. Springer, 2009, pp. 177–190.

[32] S. Li, R. Pan, A. Gupta, S. Xu, Y. Fang, and H. Huang, "Predicting the risk of rupture for vertebral aneurysm based on geometric features of blood vessels," *Royal Society Open Science*, vol. 8, no. 8, p. 210392, 2021.

[33] F. Lan, B. Gamelin, L. Yan, J. Wang, B. Wang, and H. Guo, "Topological characterization and uncertainty visualization of atmospheric rivers," in *Computer Graphics Forum*. Wiley Online Library, 2024, p. e15084.

[34] S. Abdulah, A. H. Baker, G. Bosilca, Q. Cao, S. Castruccio, M. G. Genton, D. E. Keyes, Z. Khalid, H. Ltaief, Y. Song *et al.*, "Boosting earth system model outputs and saving petabytes in their storage using exascale climate emulators," in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 1–12.

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

[36] M. Xia, S. Di, F. Cappello, P. Jiao, K. Zhao, J. Liu, X. Wu, X. Liang, and H. Guo, "Preserving topological feature with sign-of-determinant predicates in lossy compression: A case study of vector field critical points," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 4979–4992.

[37] P. Lindstrom, "Error distributions of lossy floating-point compressors," Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2017.

[38] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary, "Data compression for the exascale computing era-survey," *Supercomputing frontiers and innovations*, vol. 1, no. 2, pp. 76–88, 2014.

[39] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[40] Y. Li, X. Liang, B. Wang, Q. Yongfeng, L. Yan, and G. Hanqi, "Msz: An efficient parallel algorithm for correcting morse-smale segmentations in error-bounded lossy compressors," in *2024 IEEE Visualization and Visual Analytics (VIS)*, 2024(In Press).

[41] M. Soler, M. Plainchault, B. Conche, and J. Tierny, "Topologically controlled lossy compression," in *Proceedings of 2018 IEEE Pacific Visualization Symposium*, 2018, pp. 46–55. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/PacificVis.2018.00015

[42] L. Yan, X. Liang, H. Guo, and B. Wang, "Toposz: Preserving topology in error-bounded lossy compression," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, p. 1302–1312, Nov. 2023. [Online]. Available: https://doi.org/10.1109/TVCG.2023.3326920

[43] S. K. Lodha, J. C. Renteria, and K. M. Roskin, "Topology preserving compression of 2d vector fields," in *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*. IEEE, 2000, pp. 343–350.

[44] T. K. Dey, J. A. Levine, and R. Wenger, "A delaunay simplification algorithm for vector fields," in *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. IEEE, 2007, pp. 281–290.
[45] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," *arXiv preprint arXiv:1703.00395*, 2017.
[46] H. Edelsbrunner and E. P. Mücke, "Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms," *ACM Trans. Graph.*, vol. 9, no. 1, p. 66–104, Jan. 1990. [Online]. Available: https://doi.org/10.1145/77635.77639
[47] M. W. Kutta, "Beitrag zur näherungsweisen Integration totaler Differentialgleichungen," *Zeitschrift für Mathematik und Physik*, vol. 46, pp. 435–453, 1901.
[48] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed. Springer, 1987.
[49] J. Helman and L. Hesselink, " Representation and Display of Vector Field Topology in Fluid Flow Data Sets ," *Computer*, vol. 22, no. 08, pp. 27–36, Aug. 1989. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/2.35197
[50] T. Wischgoll and G. Scheuermann, "Detection and visualization of closed streamlines in planar flows," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 165–172, 2001.
[51] D. Ebert, P. Brunet, and I. Navazo, "Locating closed streamlines in 3d vector fields," *methods*, vol. 16, p. 19, 2002.
[52] H. Alt and M. Godau, "Computing the fréchet distance between two polygonal curves," *International Journal of Computational Geometry & Applications*, vol. 5, no. 01n02, pp. 75–91, 1995.
[53] 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.
[54] 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 *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 179–189.
[55] *OpenMP Application Programming Interface*, OpenMP Architecture Review Board, 2023, available at https://www.openmp.org/specifications/.
[56] S. Popinet, "Free computational fluid dynamics," *ClusterWorld*, vol. 2, no. 6, 2004. [Online]. Available: http://gfs.sf.net/
[57] T. Günther, M. Gross, and H. Theisel, "Generic objective vortices for flow visualization," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 36, no. 4, pp. 141:1–141:11, 2017.
[58] H. I. dataset, http://sciviscontest-staging.ieeevis.org/2004/data.html, online.
[59] P. Fischer, J. Lottes, and H. Tufo, "Nek5000," Argonne National Lab.(ANL), Argonne, IL (United States), Tech. Rep., 2007.
[60] "Morgan Compute Cluster," https://docs.ccs.uky.edu, 2023, online.