Time-varying Vector Field Compression with Preserved Critical Point Trajectories

Mingze Xia
Oregon State University
Corvallis, OR, USA
xiami@oregonstate.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

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

Xin Liang*
Oregon State University
Corvallis, OR, USA
lianxin@oregonstate.edu

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

***Abstract*—Scientific simulations and observations are producing vast amounts of time-varying vector field data, making it hard to store them for archival purposes and transmit them for analysis. Lossy compression is considered a promising approach to reducing these data because lossless compression yields low compression ratios that barely mitigate the problem. However, directly applying existing lossy compression methods to time-varying vector fields may introduce undesired distortions in critical-point trajectories, a crucial feature that encodes key properties of the vector field. In this work, we propose an efficient lossy compression framework that exactly preserves all critical-point trajectories in time-varying vector fields. Our contributions are threefold. First, we extend the theory for preserving critical points in space to preserving critical-point trajectories in space-time, and develop a compression framework to realize the functionality. Second, we propose a semi-Lagrange predictor to exploit the spatiotemporal correlations in advection-dominated regions, and combine it with the traditional Lorenzo predictor for improved compression efficiency. Third, we evaluate our method against state-of-the-art lossy and lossless compressors using four real-world scientific datasets. Experimental results demonstrate that the proposed method delivers up to 124.48× compression ratios while effectively preserving all critical-point trajectories. This compression ratio is up to 56.07× higher than that of the best lossless compressors, and none of the existing lossy compressors can preserve all critical-point trajectories at similar compression ratios.**

***Index Terms*—Scientific Data, Spatiotemporal Data Compression, Feature Preservation, Critical Point Trajectories**

I. Introduction

Large-scale scientific simulations and observations produce massive amounts of data that overwhelm data storage and transmission systems. For instance, direct numerical simulation (DNS) of forced isotropic turbulence on the Frontier supercomputer [1] generates 0.5 PB of data in a single run [2], and the ERA5 Atmospheric Reanalysis, which records climate data from 1979 to the present, already accumulates to 5 PB [3]. Such large data sizes not only make it difficult for producers to store the data but also hinder researchers and scientists from accessing it from remote sites.

Time-varying vector fields, such as flow velocity in DNS and atmospheric wind speed in ERA5, are a significant component of scientific data. These data describe how vectors change over both space and time, and are widely used to compute derived quantities and extract features of interest [4]–[7]. As such, enabling timely access to time-varying vector fields is a pressing need across many scientific and engineering disciplines. This often requires efficient data compression due to the unprecedented sizes of data.

While lossless compression is the most general way to reduce data size, it is inefficient for floating-point scientific data because of the random mantissas. As demonstrated in prior work [8], [9], generic lossless compressors, such as GZIP [10] and ZSTD [11], achieve modest compression ratios of less than 2× on most scientific datasets, which is insufficient to address the scientific data challenge. Alternatively, error-controlled lossy compression [12]–[18] is regarded as a promising approach for reducing scientific data, as it achieves much higher compression ratios while enforcing user-specified error constraints.

Despite the success of error-controlled lossy compression in several scientific domains [19]–[21], directly applying it to time-varying vector fields can result in undesired distortions of essential features, with critical-point trajectories being a typical example. A critical-point trajectory is a time-parametrized curve such that every point on it has the value $\mathbf{0}$, and it is broadly used across multiple scientific disciplines to represent key features [22], [23]. In climatology, critical points denote the centers of hurricanes, so their trajectories reveal how hurricanes traverse. In fluid dynamics, critical points are stagnation points, and their trajectories are crucial for understanding complex fluid flow patterns. Given the importance of critical-point trajectories, they should not be altered in time-varying vector field compression, because distorted trajectories would lead to misinterpretations of the underlying dynamic structures and hinder accurate prediction of system evolution. While

*Corresponding author: Xin Liang, School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331.

prior works [24]–[26] enable the preservation of critical points during error-controlled lossy compression of single-snapshot data, they may still distort critical-point trajectories because the connections of critical points in space-time are overlooked.

In this work, we propose the first highly effective lossy compression framework that exactly preserves critical-point trajectories for time-varying vector fields. Specifically, we extend the theory in [26] for preserving critical points in space to that of preserving critical-point trajectories in space-time, and then develop a lossy compression framework with exact critical-point trajectory preservation. To further improve compression efficiency, we propose a semi-Lagrange predictor that better exploits spatiotemporal correlation in advection-dominated regions of time-varying vector fields, and combines it with the traditional Lorenzo predictor [27] for enhanced data decorrelation. To this end, we evaluate our framework on four real-world scientific datasets with six baselines. In summary, our contributions are as follows.

- We design and develop the first lossy compression framework that preserves critical-point trajectories in time-varying vector fields, by extending the theory in [26] to account for critical-point connections in space-time.
- We propose a semi-Lagrange predictor which leverages particle advection theory to exploit spatiotemporal correlation in time-varying vector fields. We further leverage a mixture of predictors that combines the semi-Lagrange predictor and the traditional Lorenzo predictor for improved compression efficiency.
- We evaluate the proposed method and compare it with three widely used lossless compressors and three leading lossy compressors using four scientific datasets. Experiments demonstrate that our method faithfully preserves all critical-point trajectories while delivering up to 56$\times$ higher compression ratios than lossless compressors. Meanwhile, existing lossy compressors produce noticeable distortions in critical-point trajectories at similar or lower compression ratios.

The remaining sections of the paper are organized as follows. Section II discusses related works. Section III reviews the background on time-varying vector fields, critical-point trajectories, and the existing critical-point preserving compression algorithm. Section IV formulates the research problem. In Section V, we introduce the design of our compression framework with the preservation of critical-point trajectories. In Section VI, we present the semi-Lagrange predictor and how we combine it with the Lorenzo predictor. In Section VII, we present and analyze the evaluation results. Finally, we conclude with a vision for future work in Section VIII.

II. Related Works

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

A. Lossy Compression for Scientific Data

Traditional lossless compression, for example, LZ-family compressors such as gzip [10], Zstd [11], and LZ4 [28], as well as floating-point oriented tools like FPZIP [29], typically achieve a limited compression ratio on scientific data [9]. This limitation has motivated the community to adopt error-bounded lossy compression to achieve a higher compression ratio while maintaining user-specified error metrics. Error-bounded lossy compressors usually follow the steps: decorrelation, quantization, and encoding. Depending on the decorrelation method, we can categorize these compressors into prediction-based compressors and transform-based compressors. The SZ compressor family [12], [30]–[32] is a set of representative prediction-based compressors. They use predictors such as Lorenzo and regression for decorrelation and the decorrelated data is further quantized and encoded. Transform-based compressors leverage specific transforms for data decorrelation, and then perform quantization and encoding in the transformed domains. ZFP [33] is a typical transform-based compressor that divides data into non-overlapped data blocks for independent compression. Other notable transformed-based compressors include TTHRESH [34], SPERR [35], and FAZ [36]. MGARD [15], [37], [38] is another lossy compressor that lies in the middle of prediction-based and transform-based compressors. It relies on multilinear interpolation with L^2 projection for data decorrelation, and applies level-wise quantization.

Beyond engineering practice, the community has also studied error propagation and numerical stability, including error analyses, statistical modeling of compression error distribution [9]. Together, these results support replacing lossless compression with controlled-lossy approaches when application-level quantities of interest(QoIs) [39], [40] remain within tolerance. In short, given the limited ratios of lossless methods on floating-point data, error-bounded lossy compression has become the mainstream solution for scientific data reduction and lays the groundwork for higher-level, structure-aware guarantees. This motivates topology-aware compressors that explicitly preserve structural invariants.

B. Topology-Aware Compression

Although error-bounded lossy compressors can enforce constraints such as L^∞ or L^2 bounds, many scientific analyses care primarily about the topological structure of scalar or vector fields. For example, the number and locations of critical points, the connectivity of level sets like contours and isocontours, and the stability of contour/merge trees and Morse-Smale complexes. Purely amplitude-based error bounds can create or annihilate critical points or alter connectivity, undermining feature tracking and subsequent analysis.

For the scalar field data, compressor building on persistent homology has been proposed [41], this approach guarantee that high-persistence features are preserved exactly under a user-specified persistence threshold, while low-persistence noise is simplified and encoded. Methods such TopoSZ [42] incorporate topological invariants directly into the prediction-quantization pipeline of SZ [43], preserving the contour tree of scalar fields while meeting pointwise error bounds. MSz and its derived variants [44], [45] are capable of preserv-

ing the Morse-Smale complex structure while simultaneously leveraging parallelization to significantly reduce compression time. A complementary line of work [46] performs local or global topological simplification with L^∞ control. Although this method is not compression algorithms per se, it can be integrated with existence compressor framework to preserving the topological structure of the data.

The need for topology-aware constraints naturally extends from scalar fields to vector and tensor fields. For example, the cpSZ framework [25] is capable of preserving critical points during lossy compression, while cpSZ-SoS [26] further extends this framework into a parallelized compressor suitable for large-scale datasets. At the same time, it replaces the original numerical scheme with the Simulation of Simplicity (SoS) method for critical-point computation, thereby avoiding numerical instabilities that can occur in extreme cases. Building upon these foundations, subsequent work [47] further extends such preservation to separatrix structures. More recently, the TFZ framework [48] has introduced a topology-aware lossy compression strategy for 2D second-order tensor fields, illustrating that topology-aware compression is evolving from scalr fields toward higher-dimensional and more complex data types.

III. Background and Preliminaries

In this section, we introduce the necessary background and preliminaries related to our work, including the notations, fundamental concepts, and techniques that form the basis of our proposed method.

A. Notation and Assumptions

Let $D \subset \mathbb{R}^2$ be a polygonal domain endowed with a time-invariant triangular mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, where $\mathcal{V}$,$\mathcal{E}$, and $\mathcal{F}$ denote the sets of vertices, edges and trianular faces, respectively. The vertex positions are fixed in space across discrete times $\mathcal{T} = \{t_0, \ldots, t_T\}$. A time-varying 2D vector field is given by

$$\mathbf{V}(x, y, t) = (u(x, y, t),\, v(x, y, t)) \in \mathbb{R}^2.$$

We assume piecewise-linear interpolation in both space and time, which is a common setting in literature [24], [49], [50]. We then use $\mathbf{v}_i^k = (u_i^k, v_i^k)$ to denote the vector value at vertex $i \in \mathcal{V}$ and time t_k.

B. Space–Time Extrusion and Tetrahedralization

Existing approaches [24], [26] cannot reliably maintain the correctness of critical point tracking path, as they lack a unified mechanism that couples spatial and temporal domains. Since critical points evolve over space and time through event such as birth, death, split, and merge [50], any lossy compression performed without proper space-time alignment may produce artificial critical points within time slabs. This results in erroneous event identification and, ultimately, incorrect topological transitions. Figure. 1 depicts the evolution of critical points under various topological events and highlights how potential spurious critical points may distort the underlying topological structure, resulting in incorrect event identification and topology alteration.

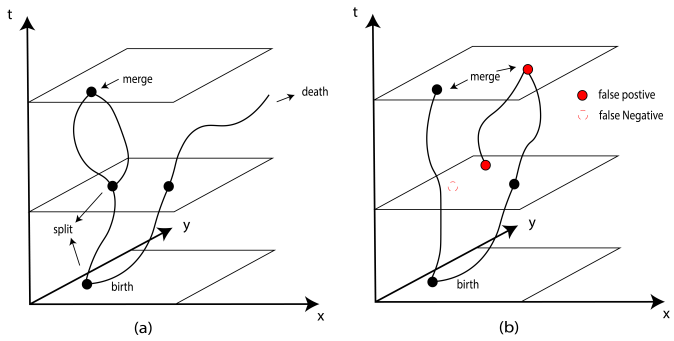


Fig. 1. (a) presents an example of the temporal evolution of the critical point and the corresponding events; (b) illustrates how lossy compression alters the critical point, leading to a change in its trajectory.

To address this issue, we adopt the method proposed by De Loera et al. [51], which provides a unified representation of space and time by extruding the spatial domain along the temporal dimension, followed by a consistent space–time decomposition. At each time step t_k, we have a consistent triangular mesh $\Delta \in \mathcal{F}$. We denote by $\mathcal{V}_t^a$, $\mathcal{V}_t^b$, and $\mathcal{V}_t^c$ the three vertices of a triangle in Δ at time t. Then we extrude Δ over the temporal interval $[t_k, t_{k+1}]$ to form a triangular prism. This prism is then subdivided using a fixed 3-tetrahedron split:

$$\begin{aligned}
\tau_1 &= (\mathcal{V}_t^a,\, \mathcal{V}_t^b,\, \mathcal{V}_t^c,\, \mathcal{V}_{t+1}^c),\\
\tau_2 &= (\mathcal{M}_t^a,\, \mathcal{M}_t^b,\, \mathcal{V}_{t+1}^b,\, \mathcal{V}_{t+1}^c),\\
\tau_3 &= (\mathcal{V}_t^a,\, \mathcal{V}_{t+1}^a,\, \mathcal{V}_{t+1}^b,\, \mathcal{V}_{t+1}^c).
\end{aligned}$$

Within any tetrahedron $\tau \subset D \times [t_k, t_{k+1}]$, the components (u, v) are affine in (x, y, t); hence the joint zero-set $\{\mathbf{V} = 0\}$ is either empty or a straight line segment in τ. As illustrated in Figure 2, this subdivision yields a refined and temporally coherent discretization of the data, ensuring a uniform topological structure across consecutive time steps. This consistency is crucial for the accurate detection and tracking of critical points in time-dependent vector fields.

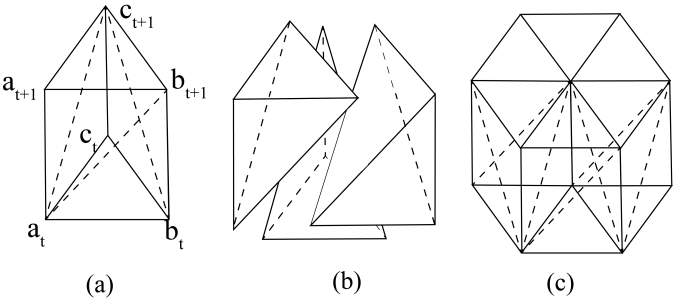


Fig. 2. Space-Time extrusion of simplicial mesh. (a) A 2D triangle is extruded along the temporal dimension to form a prism. (b) A triangular prism is structurally subdivided into three tetrahedra. (c) A 2D simplicial mesh is extruded into a 3D prismatic mesh and subsequently subdivided.

C. Critical Point and Trajectory

A *critical point* is defined as the location where the vector field vanishes. Therefore, a critical point at time t_k is $\mathbf{p} \in D$

with $\mathbf{V}(\mathbf{p}, t_k) = \mathbf{0}$. A *Critical point trajectory* over $[t_k, t_{k'}]$ is a polyline obtained by concatenating the line segments $\{\mathbf{V} = 0\}$ across adjacent tetrahedra in $D \times [t_k, t_{k'}]$.

Under piecewise-linear assumption, the intersection $\{\mathbf{V} = 0\} \cap (D \times \{t_k\})$ is governed by a *face-level test*: for a space–time triangular face $f = \{i, j, k\}$, if the origin lies in $\mathrm{conv}\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k\} \subset \mathbb{R}^2$, then the zero-set crosses f and yields a slice critical point.

D. Face-Level test

Consider a triangular face f of the space–time tetrahedral mesh. Let the three incident vertices carry vector values $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\} \subset \mathbb{R}^2$. Because $\mathbf{V}$ is piecewise-linear, the restriction of $\mathbf{V}$ to f is the convex hull of $\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ in $\mathbb{R}^2$:

$$\{\mathbf{V}|_f\} = \Big\{\alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \;\Big|\; \alpha, \beta, \gamma \geq 0,\ \alpha + \beta + \gamma = 1\Big\}.$$

Define the 2D scalar cross product $\det(\mathbf{x}, \mathbf{y}) = x_x y_y - x_y y_x$. Let

$$D_f = \det(\mathbf{a}, \mathbf{b}) + \det(\mathbf{b}, \mathbf{c}) + \det(\mathbf{c}, \mathbf{a}).$$

$$\mathrm{sign}\{x, y, z\} = \begin{cases} 1 & \text{if } x > 0,\ y > 0,\ z > 0, \\ -1 & \text{if } x < 0,\ y < 0,\ z < 0, \\ 0 & \text{otherwise.} \end{cases}$$

In general position, the origin $\mathbf{0}$ lies in $\mathrm{conv}\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ iff

$$\mathrm{sign}\big\{\det(\mathbf{a}, \mathbf{b}),\ \det(\mathbf{b}, \mathbf{c}),\ \det(\mathbf{c}, \mathbf{a})\big\} \neq 0, \tag{1}$$

equivalently, the barycentric coefficients of the origin are

$$(\alpha^\star, \beta^\star, \gamma^\star) = \frac{1}{D_f}\big(\det(\mathbf{b}, \mathbf{c}),\ \det(\mathbf{c}, \mathbf{a}),\ \det(\mathbf{a}, \mathbf{b})\big), \tag{2}$$

which satisfy $\alpha^\star, \beta^\star, \gamma^\star > 0$ exactly when (1) holds. If (1) is true, we say *face f is crossed by a critical point*. The point of crossing in the geometric triangular face (in x–y–t) is the same barycentric combination of the three face vertices.

E. Simulation of Simplicity (SoS)

We adopt the *Simulation of Simplicity (SoS)* [52] paradigm to avoid ambiguous degeneracies in critical point detection. Typical degeneracies include: (i) $\det(\cdot, \cdot) = 0$ in Eq. (1); (ii) zero-line intersections exactly passing through a vertex or an edge; and (iii) multiple faces (three or more) being simultaneously intersected within one tetrahedron. SoS introduces consistent symbolic perturbations (e.g., lexicographic tie-breaking by vertex or time indices) that deterministically select a side, thereby ensuring that Lemma 1 and Theorems 1–2 hold under all configurations. Algorithm 1 outlines the robust inside-test under SoS for a 2D simplex.

F. Derivation of error bounds for critical point preservation

Previous work [47] derived a sufficient but not necessary condition on the absolute error bound to avoid false positive case(spurious critical points) in critical point preservation. The detail algorithm is shown in Alg.2. We will utilize this derivation in Section V to ensure that the proposed framework can preserve critical-point trajectories. For the complete proof, please refer to the referenced paper.

Algorithm 1 Robust Critical Point Detection in a 2D Simplex

Input: $V = \{(v_0, id_0), (v_1, id_1), (v_2, id_2)\}$, where $v_i \in \mathbb{R}^2$ are vertex coordinates and $id_i \in \mathbb{Z}$ are their global indices
Output: $b \in \{\text{true}, \text{false}\}$

$x \leftarrow (0, 0)$
Extract coordinate matrix $P \leftarrow [v_0, v_1, v_2]^\top$ and index array $idx \leftarrow [id_0, id_1, id_2]$
$s \leftarrow \textsc{Orient2\_SoS}(P, idx)$ ▷ Compute orientation sign (± 1) via SoS: evaluate the 3×3 homogeneous determinant; flip sign if permutation parity is odd
for $i \leftarrow 0$ to 2 **do**
    $Y \leftarrow P$; $Y[i] \leftarrow x$
    $id' \leftarrow idx$; $id'[i] \leftarrow -1$ ▷ -1 serves as the virtual index of x for SoS tie-breaking
    $s_i \leftarrow \textsc{Orient2\_SoS}(Y, id')$ ▷ Consistency test after replacing the i-th vertex by x
    if $s_i \neq s$ **then**
        return false
return true

Algorithm 2 Derivation of eb for critical point preservation

Input: $u_0, u_1, u_2, v_0, v_1, v_2$ ▷ triangle vertices' vector components
Output: eb ▷ absolute error bound to preserve critical point

$M_0 \leftarrow u_2 v_0 - u_0 v_2$, $M_1 \leftarrow u_1 v_2 - u_2 v_1$
$M_2 \leftarrow u_0 v_1 - u_1 v_0$, $M \leftarrow M_0 + M_1 + M_2$
if $M = 0$ **then**
    return 0
$eb \leftarrow \dfrac{|M|}{|u_1 - u_0| + |v_0 - v_1|}$
if $|u_1| + |v_1| \neq 0$ **then**
    $eb \leftarrow \mathrm{MINF}\left(eb,\ \dfrac{|u_1 v_2 - u_2 v_1|}{|u_1| + |v_1|}\right)$
else
    return 0
if $|u_0| + |v_0| \neq 0$ **then**
    $eb \leftarrow \mathrm{MINF}\left(eb,\ \dfrac{|u_0 v_2 - u_2 v_0|}{|u_0| + |v_0|}\right)$
else
    return 0

if $\mathrm{same\_sign}(u_0, u_1, u_2)$ **then**
    $eb \leftarrow \mathrm{MAX}(eb,\ |u_2|)$
if $\mathrm{same\_sign}(v_0, v_1, v_2)$ **then**
    $eb \leftarrow \mathrm{MAX}(eb,\ |v_2|)$
return eb

IV. Problem Formulation and Guarantees

In this section, we formally define the problem, specify the constraints that guide our compression design, and establish the theoretical guarantees for preserving critical-point trajectories under lossy compression.

A. Research Problem and Goal

Given the original vector field dataset $\mathbf{D} = \{\mathbf{u}, \mathbf{v}\}$ and a user-specified error bound ε,our goal is to find a decompressed dataset $\widehat{\mathbf{D}}$ that maximizes the compression ratio while satisfying:

a) Constraint on point-wise error.:

$$\|\hat{\mathbf{D}} - \mathbf{D}\|_\infty \leq \varepsilon,$$

b) Constraint on critical-point trajectories.: For every space–time triangular face $f = \{i, j, k\}$ (including both time-slice faces and cross-time slab faces), define the face-level critical point predicate

$$P_f(\mathbf{d}_i, \mathbf{d}_j, \mathbf{d}_k) := \mathbb{I}\big(\ \mathbf{0} \in \mathrm{conv}\{\mathbf{d}_i, \mathbf{d}_j, \mathbf{d}_k\}\ \big),$$

We require

$$\forall f: \quad P_f(\widehat{\mathbf{d}}_i, \widehat{\mathbf{d}}_j, \widehat{\mathbf{d}}_k) \ = \ P_f(\mathbf{d}_i, \mathbf{d}_j, \mathbf{d}_k). \tag{3}$$

Under piecewise-linear interpolation and general position (or SoS), the face-level invariance (3) is *equivalent* to preserving the set of critical-point trajectories:

$$\mathcal{T}(\widehat{\mathbf{D}}) \ = \ \mathcal{T}(\mathbf{D}),$$

where $\mathcal{T}(\cdot)$ denotes the track set obtained by linking face crossings across the space–time mesh.

B. Guarantees

We now provide theoretical guarantees showing that the proposed constraints are sufficient to ensure the preservation of per-time critical points and their temporal trajectories.

Lemma 1 (Zero-set in a tetrahedron)**.** *Under piecewise-linear interpolation in x–y–t, the set $\{\mathbf{V} = 0\}$ restricted to any tetrahedron τ is either empty or a straight line segment whose endpoints lie on two distinct faces of τ. In general position, intersections with the boundary are either 0 or 2 points.*

Theorem 1 (Per-time critical point preservation)**.** *If the critical point predicate on every triangular face of the space–time mesh is invariant under compression, then for every time slice t_k, the critical point set on $D \times \{t_k\}$ is unchanged (same cardinality and locations via piecewise-linear barycentric mapping).*

Sketch. A critical point at time t_k is precisely an intersection of the zero-set with a time-slice face lying in $D \times \{t_k\}$; by Lemma 1, such intersections are exactly the face crossings decided by the predicate in Sec. III-D. If the predicate outcome on all time-slice faces is preserved, the set of intersection points on each slice is identical; their in-face barycentric coordinates (hence geometric positions) are preserved by construction. □

Theorem 2 (Track equivalence)**.** *Assume general position so that in every tetrahedron the zero-set is either empty or a single segment with endpoints on two faces. If face-level critical point decisions are preserved on all faces, then the adjacency of segments across shared faces is identical before and after compression, hence the reconstructed track graph (number of tracks, their connectivity, and combinatorial structure) is unchanged.*

Sketch. Within a tetrahedron, the two boundary intersections determine a unique segment; preserving which faces are crossed preserves which segment exists. Across tets, two segments meet if and only if they share the same crossing point on a shared face. Because the face predicate and the barycentric crossing on that face are preserved, the segment gluing is unchanged, yielding an isomorphic track graph. □

According to Theorem 1 and Theorem 2, we can conclude that preserving all per-time critical points as well as all cross-time-slab critical points constitutes a sufficient condition for ensuring the preservation of critical point trajectories.

V. Design Overview

This section mainly illustrates how our method preserves the critical point tracking paths. Figure 3 presents an overview of our framework.

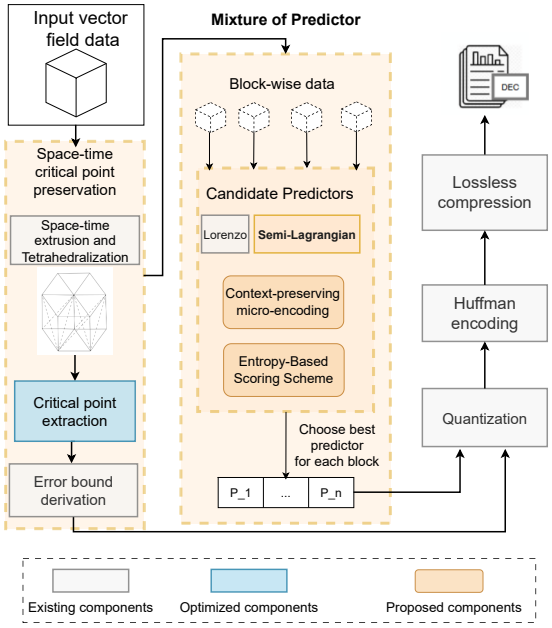


Fig. 3. Overview of the proposed framework.

A. Compression Pipeline

Our compression pipeline is built on the prior work cpSZ-sos [26], with several key enhancements and modifications highlighted in colors. Specifically, we introduce a space-time discretization module as we discussed in Section III-B, which enables our algorithm to handle time-dependent 2D data by constructing a consistent spatio-temporal mesh representation. We further modify the existing critical point extraction module so that it can accurately identify critical points within each individual time slab. Finally, we optimize the prediction module through a Mixture of Predictors (MoP) scheme, which adaptively selects the most suitable predictor for each data block. This design not only enhances the efficiency of Huffman coding by producing smaller residuals and thus a more compact quantization distribution, but also introduces stronger structural patterns into the quantized data, thereby achieving a significantly higher compression ratio in the lossless compression stage. Algorithm 3 presents the baseline version of our compression pipeline without any optimizations. Our algorithm first converts the input data into

Algorithm 3 Critical-Point-Tracking–Preserving Lossy Compression for 2D Temporal Vector Fields

Input: 2D Temporal Vector Fields $\{\mathbf{d}\} = \{\mathbf{U}, \mathbf{V}\}$ of size $N = H \times W \times T$; error bound ϵ.
Output: Compressed bytes.

Allocate $U_{\text{fp}}, V_{\text{fp}} \in \mathbb{Z}^N$ (int64)
`scale`, $U_{\text{fp}}, V_{\text{fp}} \leftarrow$ `Convert_FixedPoint`$(\mathbf{U}, \mathbf{V})$ // convert to fixed point
$\{\boldsymbol{\tau}\} \leftarrow$ `Tetrahedralization`$(\mathbf{U}, \mathbf{V})$
Initialize $\mathcal{C}$ // empty unordered set of FaceKey
for $tet \in \boldsymbol{\tau}$ **do** // iterate each tetrahedron
  for $f \in tet$ **do** // iterate each triangular face
    $\mathcal{C}$.insert(`SoS_robust_test`(f)) // perform critical point test for face with global index, see Alg.1
$\tau' = \tau *$ `scale`
for $t = 0, \ldots, T-1$ **do**
  for $i = 0, \ldots, H-1$ **do**
    for $j = 0, \ldots, W-1$ **do**
      $v \leftarrow VID(t, i, j)$
      $\xi^v \leftarrow \tau'$
      $\xi^v, \mathcal{L}_d \leftarrow$ `derive_min_eb`$(v, U_{\text{fp}}, V_{\text{fp}}, \mathcal{C}, \tau')$ // Alg. 4
      $Q_\xi^v \leftarrow$ `quantization`(ξ^v)
      $Q_d^v, \mathcal{L}_d \leftarrow$ `Prediction_and_quant`$(\mathbf{d}^v, \hat{\mathbf{d}}^v)$
`output_bytes` $\leftarrow$ `lossless_comp`$(\{Q_\xi\}, \{Q_d\}, \mathcal{L}_d)$
return `output_bytes`

Algorithm 4 Derive Minimum Error Bound for Each Vertex v

Input: Vertex $v = \text{VID}(t, i, j)$; fixed-point fields $U_{\text{fp}}, V_{\text{fp}}$; precomputed face set $\mathcal{C}$; initial bound τ'
Output: Updated per-vertex bound $\xi^{(v)}$; updated lossless-vertex set $\mathcal{L}_d$

procedure UPDATEBYFACE($v, (a, b, c), U_{\text{fp}}, V_{\text{fp}}, \mathcal{C}, \xi$)
  if $v \notin \{a, b, c\}$ **then**
    return ξ
  if $\triangle(a, b, c)) \in \mathcal{C}$ **then**
    return 0
  $e \leftarrow$ `Derive_eb`$\big((U_{\text{fp}}[a], U_{\text{fp}}[b], U_{\text{fp}}[c]),\ (V_{\text{fp}}[a], V_{\text{fp}}[b], V_{\text{fp}}[c])\big)$ // see Alg. 2
  return $\min(\xi, e)$
$\xi^{(v)} \leftarrow \tau'$
must_lossless $\leftarrow$ **false**
`// Case 1: compute 6 surrounding` $\triangle$ `of points`(i, j) `at` t
for $c_i \in \{i-1, i\}$ **with** $0 \le c_i$ **and** $c_i + 1 < H$ **do**
  for $c_j \in \{j-1, j\}$ **with** $0 \le c_j$ **and** $c_j + 1 < W$ **do**
    $v_{00} \leftarrow \text{VID}\,(t, c_i, c_j)$; $v_{10} \leftarrow \text{VID}\,(t, c_i, c_j + 1)$
    $v_{01} \leftarrow \text{VID}\,(t, c_i + 1, c_j)$; $v_{11} \leftarrow \text{VID}\,(t, c_i + 1, c_j + 1)$
    $\xi^{(v)} \leftarrow$ UPDATEBYFACE $\big(v, (v_{00}, v_{01}, v_{11}), U_{\text{fp}}, V_{\text{fp}}, \mathcal{C}, \xi^{(v)}\big)$
    $\xi^{(v)} \leftarrow$ UPDATEBYFACE $\big(v, (v_{00}, v_{10}, v_{11}), U_{\text{fp}}, V_{\text{fp}}, \mathcal{C}, \xi^{(v)}\big)$
    if $\xi^{(v)} = 0$ **then**
      must_lossless $\leftarrow$ **true**; **break**
  if *must_lossless* **thenbreak**
`// Case 2:` $\triangle \perp t$ `on [`t`,` $t+1$`]`
`// Case 3: complementary` $\triangle \perp t$ `on [`$t-1$`,` t`]`
`// Case 4: internal split` $\triangle$ `on [`t`,` $t+1$`]`
`// Case 5: complementary internal split` $\triangle$ `on [`$t-1$`,` t`]` // Codes for the above four cases are omitted for brevity.
if *must_lossless* **then**
  $\mathcal{L}_d$.insert (v) // mark vertex for lossless storage
  return 0, $\mathcal{L}_d$
else
  return $\xi^{(v)}$, $\mathcal{L}_d$

a fixed-point representation to enhance the robustness of the critical point test. Next, we tetrahedralize the dataset according to the convention described previously. We then iterate over all tetrahedra and their triangular faces, performing a robust critical point test(SoS) on each face and recording the results. In the next stage, we traverse all vertices and derive their respective error bounds(4). Once all vertex-wise error bounds are obtained, we quantize them, then we perform prediction and quantization for each vertex's data values. Finally, we package$\{Q_\xi\}, \{Q_d\}, \mathcal{L}_d$, and other metadata, then pass them to Zstd for further lossless compression.

The Algorithm 4 illustrates how each vertex is processed and its constraints are derived to preserve critical points in 2D time-dependent data. Since each tetrahedron is formed by connecting data between time steps t and $t+1$, each vertex is associated with up to 36 triangular faces within its surrounding $3 \times 3 \times 3$ neighborhood. Therefore, we compute the error bound for every related triangle and take the minimum as the final bound for that vertex. We divided those 36 triangular faces into 6 cases, case 1 illustrates the six triangles lying in the same spatial plane, while cases 2-5 correspond to different types of triangles formed within the slabs$[t-1, t]$ and $[t, t+1]$.

To avoid false negative case(missing critical points), if any triangle involving a given vertex has previously been marked as containing a critical point in $\mathcal{C}$, we set the error bound for that vertex and all its related vertices as 0 i.e., losslessly sotre. Previous work [26] derived a sufficient but not necessary condition on the error bound to avoid false positive case(spurious critical points) in critical point preservation. By integrating this method, our approach ensures that all triangular faces within each time slice t and across temporal slabs preserve critical points correctly before and after compression, thereby guaranteeing accurate tracking of critical points over time.

VI. BLOCK-WISE ADAPTIVE MIXTURE OF PREDICTORS

Prediction plays a crucial role in improving compression efficiency, as accurate prediction effectively reduces the magnitude of residuals to be encoded, leading to a more compact entropy distribution and thus higher compression ratios. In particular, prior research [32], [53] has shown that block-based predictive coding effectively adapts to local data heterogeneity, which is crucial for scientific datasets exhibiting complex spatial and temporal patterns.

We focus on compressing 2D temporal vector field data $\mathbf{v}(x, y, t) = (u, v)$. Numerical simulation data often exhibit strong local physical heterogeneity. In different subdomains, the dominant physical mechanisms such as advection, convection or nonlinear transport, and diffusion can vary significantly. Most state-of-the-art scientific lossy compressors [30]–[32], [54] adopt the first-order Lorenzo predictor [27] as their prediction model. This model performs well in locally smooth or diffusion-dominated regions due to its trilinear reproduction property, but it produces large residuals in regions with significant translational or transport structures (for example, vortex-core migration or stripe advection), which reduces compression efficiency.

To address this limitation, we propose a **block-wise adaptive Mixture of Predictors** with two candidates: **First order 3D-Lorenzo(3DL)** and a **Semi-Lagrangian (SL)** predictor tailored for advection-dominated regions. We partition each

frame at time t into non-overlapping blocks B of size $B_x \times B_y$. For each spatial block we evaluate both predictors with a light-weight, context-preserving look-ahead, then select the mode that minimizes the estimated rate; this reduces residual entropy and improves overall compression ratio while strictly preserving the user-prescribed error bound.

A. Semi-Lagrangian Predictor

Let spatial steps be $\Delta x, \Delta y$, time step Δt(these parameters are included in the data's metadata or descriptive information), and denote the fixed-point scaling by S (the data are stored as S-scaled integers). Define per-axis **Courant–Friedrichs–Lewy(CFL)**-like factors

$$\mathrm{CFL}_x = \frac{\Delta t}{\Delta x} \cdot \frac{1}{S} \qquad \mathrm{CFL}_y = \frac{\Delta t}{\Delta y} \cdot \frac{1}{S}$$

At data (i, j) we form a backward displacement using the previous velocity field (u_{t-1}, v_{t-1}) and sample the previous frame at the departure point with bilinear interpolation:

$$(i^\star, j^\star) = \Big(i - v_{t-1}(i,j)\,\mathrm{CFL}_y,\;\; j - u_{t-1}(i,j)\,\mathrm{CFL}_x\Big), \tag{4}$$

$$\widehat{f}^{\mathrm{SL}}_{i,j,t} = \mathcal{I}_{\mathrm{bil}}[f_{t-1}](i^\star, j^\star), \tag{5}$$

where $\mathcal{I}_{\mathrm{bil}}$ is standard bilinear interpolation at (i_f, j_f) with clamping at domain boundaries, define as following:

$$\begin{aligned}\mathcal{I}_{\mathrm{bil}}[f](i_f, j_f) = (1-\alpha)(1-\beta) f_{i_0,j_0} + \alpha(1-\beta) f_{i_0,j_1} \\ + (1-\alpha)\beta f_{i_1,j_0} + \alpha\beta f_{i_1,j_1},\end{aligned} \tag{6}$$

where $i_0 = \lfloor i_f \rfloor$, $j_0 = \lfloor j_f \rfloor$, $i_1 = \min(i_0+1,\, H-1)$, $j_1 = \min(j_0+1,\, W-1)$, and $\alpha = i_f - i_0$, $\beta = j_f - j_0$.

Since the local flow velocities can vary significantly across different blocks, when the velocity in the current block is large, the displacement obtained by this method may cross multiple cells or even go out of bounds, leading to unstable time integration or large numerical errors. Therefore we design two strategies to handle this situation. We first estimate a local displacement magnitude d_∞ using the previous timestamp value.

$$d_\infty = \max\Big(\big|u_{t-1}(i,j)\big|\,\mathrm{CFL}_x,\;\; \big|v_{t-1}(i,j)\big|\,\mathrm{CFL}_y\Big).$$

RK2(mid-point). If displacement d_∞ is smaller than the threshold d_{max}(we set as 2 pixels), we apply the second-order Runge-Kutta method instead of Euler's method to improve the accuracy.

$$(i_{\frac12}, j_{\frac12}) = \Big(i - \tfrac12 v_{t-1}(i,j)\,\mathrm{CFL}_y,\; j - \tfrac12 u_{t-1}(i,j)\,\mathrm{CFL}_x\Big), \tag{7}$$

$$\begin{aligned}u_{t-1}(i_{\frac12}, j_{\frac12}) &= \mathcal{I}_{\mathrm{bil}}[u_{t-1}](i_{\frac12}, j_{\frac12}),\\ v_{t-1}(i_{\frac12}, j_{\frac12}) &= \mathcal{I}_{\mathrm{bil}}[v_{t-1}](i_{\frac12}, j_{\frac12})\\ (i^\star, j^\star) &= \Big(i - v_{t-1}(i_{\frac12}, j_{\frac12})\,\mathrm{CFL}_y,\; j - u_{t-1}(i_{\frac12}, j_{\frac12})\,\mathrm{CFL}_x\Big)\end{aligned} \tag{8}$$

After we obtain $(i^\star, j^\star)$, use equation.(5) to predict the u or v component. This yields a global trajectory error $\mathcal{O}(\Delta t^2)$ plus interpolation error $\mathcal{O}(h^2)$ where $h = \max\{\Delta x, \Delta y\}$.

Adaptive substepping. If $d_\infty > d_{\max}$, we split the backtracking into

$$N = \min\Big(\Big\lceil \tfrac{d_\infty}{d_{\max}} \Big\rceil,\; N_{\max}\Big) \quad \text{(we use } N_{\max} = 32 \text{ as default)},$$

and perform N explicit substeps (each of size at most $d_{\max}$ pixels) using locally sampled velocities:

$$\begin{aligned}(i_{s+1}, j_{s+1}) = (i_s, j_s) - \Big(v_{t-1}(i_s, j_s)\,\tfrac{\mathrm{CFL}_y}{N},\; u_{t-1}(i_s, j_s)\,\tfrac{\mathrm{CFL}_x}{N}\Big)\\ \text{where } s = 0, \ldots, N-1.\end{aligned} \tag{9}$$

with clamping to the data domain at each substep. and finally we have

$$(i^\star, j^\star) = (i_N, j_N)$$

then we use equation.(5) to obtain predicted value.

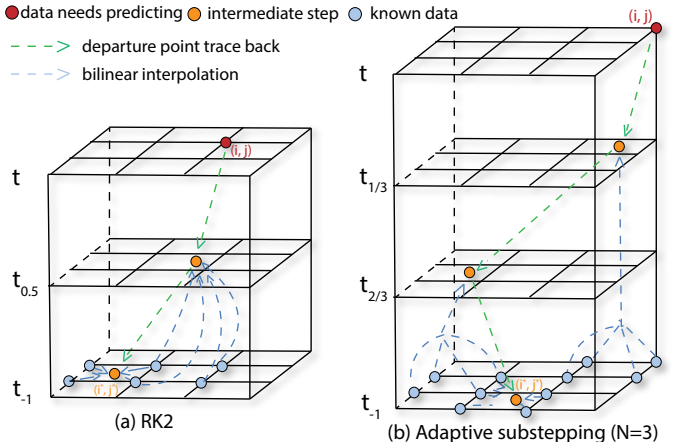


Fig. 4. RK2 and adaptive substepping method in semi-Lagrangian predictor. The green dashed arrows indicate the departure point traceback (Eq. 4), while the light blue dashed arrows represent the bilinear interpolation (Eq. 5).

Figure. 4 provides a schematic illustration of two prediction methods we proposed. The read point indicates the target location to be predicted. We first compute the coordinates of all intermediate steps(yellow points) using (7) or (9). Then, for each intermediate step, we perform bilinear interpolation of the components u and v at time $t-1$ from its four surrounding points to obtain the local velocity(u, v) at current intermediate position. This process is repeated iteratively until the final position $(i^\star, i^\star)$ is reached, where we again perform bilinear interpolation of the four neighboring points at time $t-1$ to obtain the predicted value.

B. Data Sampling and Estimation

Context-preserving micro-encoding. For each block B and each mode p, we traverse *every* data in B, compute prediction $\hat{f}^{(p)}$, form residual $r = x - \hat{f}^{(p)}$, and execute quantize and dequantize process identical to the compressor. This procedure require an error bound ϵ' that preserves its critical point. In practice, ϵ' is typically tighter than the user-specified tolerance ϵ. For optimistic estimation, we use ϵ in our calculations. After that, we write back the reconstructed values into a candidate buffer so that subsequent data see the same neighborhood as in true encoding. This guarantees the scoring respects the casual context of the actual compressor. To reduce scoring overhead,

we collect statistics only on a subsampled grid with stride n along both direction.

Score Computation. Let $h_p[k]$ be the histogram over all subsampled and valid quantized residuals from both components in B under mode p, and let $n_p = \sum_k h_p[k]$ be the valid-sample count. Let $\mathcal{L}_p$ be the number of overflow/failure events among the subsampled sites (these data needs to be compressed losslessly). We compute the zero-order entropy (bits per sample) as

$$H_0(h_p; n_p) \;=\; -\sum_k \frac{h_p[k]}{n_p}\,\log_2\Big(\frac{h_p[k]}{n_p}\Big), \qquad n_p > 0 \quad (10)$$

The estimated bits per sample per component for mode p combines entropy, an overflow penalty, and the amortized block-mode metadata:

$$\widehat{R}_p(B) \;=\; H_0(h_p; n_p) \;+\; \lambda\,\widehat{\xi}_p \;+\; R_{\mathrm{meta}},\;\; \widehat{\xi}_p \;=\; \frac{\mathcal{L}_p}{n_p + \mathcal{L}_p}$$

Here λ is a fixed penalty in bits(We set it to 16 because the data are of float type, and we assume that subsequent lossless compression can achieve a $2\times$ compression ratio [8], [9]). Since we have only two modes, the metadata cost is one bit per block, i.e.,

$$R_{\mathrm{meta}} \;=\; \frac{1}{B_x B_y \times 2} \quad \text{bits/sample/component.}$$

We choose the mode with the smaller estimated rate, subject to a relative-improvement gate, if the improvement of $\widehat{R}_p^{SL}(B)$ is greater than the threshold(we set as 0.03%), then we use SL as predictor for that block, otherwise it will rollback to 3DL to prevents mode thrashing on noisy estimates.

C. Data dependencies and Streaming I/O

For the vector field data, $d \in \{u, v\}$ at (i, j, t), the two predictors have distinct dependency sets. **3DL** depends on current-frame causal neighbors and previous-frame samples:

$$\mathcal{D}_{\mathrm{3DL}}(i,j,t) = \{(i-1,j,t),(i,j-1,t),(i-1,j-1,t),$$
$$(i,j,t-1),(i-1,j,t-1),(i,j-1,t-1),(i-1,j-1,t-1)\}.$$

Thus within a raster scan, each data requires already processed left/top neighbors at time t and a 2×2 neighbors from time $t-1$. **SL.** According to Eqs. (4)–(5) (and (7)–(8) or (9)), it depends on the previous frame in two ways: (i) the velocity field (u_{t-1}, v_{t-1}) bilinear interpolated along the backward trajectory, and (ii) the target component f_{t-1} at the departure point $(i^\star, j^\star)$. Formally,

$$\mathcal{D}_{\mathrm{SL}}(i,j,t) \;= \{(i',j',t-1) : (i',j') \in \text{backtracked path}\}$$
$$\cup\, \{(i^\star, j^\star, t-1)\}.$$

Which makes no dependency on current-frame neighbors at time t.

Since MoP adaptively selects between the 3DL and SL predictors, its dependency is limited to the previous frame and the three neighboring points of the current data point. Compared with prediction-based compressors that require a larger spatiotemporal context such as multi-level interpolation/regression schemes or block-based approaches that operate on large spatial tiles, our framework maintains a constant working set through a two-slice sliding window, without constructing large intermediate layers or block buffers. Consequently, in memory-constrained edge devices, node-level low-memory environments, or ultra–high-resolution and long temporal scenarios, our method can be implemented to operate in a fully streaming manner with linear I/O and constant peak memory, while preserving consistent critical point trajectories.

VII. EVALUATION

In this section, we present a comprehensive evaluation of our proposed compression framework and compare it against state-of-the-art compressors.

A. Experimental Setup and Datasets

We conduct our exoeriments using four scientfic datasets listed in Table I, where $\#CP_t$ and $\#CP_{slab}$ denote the total number of critical points located at time t, and within a time slab, respectively. All these four datasets [55]–[58] are stored as single-precision floating points with the following detail information.

- **Synthetic Cylinder Flow(SCF)**: synthetic vector field models the generation of a simple von Krman vortex street. It was constructed by Jung, Tel, and Ziemniak.
- **Cylinder Flow with von Kármán Vortex (CFVKV)**: simulation of a viscous 2D flow around a cylinder.
- **Heated Cylinder with Boussinesq Approximation(HCBA):** Simulation of a 2D flow generated by a heated cylinder, using Boussinesq approximation.
- **Fluid Simulation Ensemble(FS):**An ensemble simulation of 8000 unsteady two-dimensional flows, with turbulence becoming more prominent in the later datasets. Without loss of generality, we randomly pick the 6666^{th} flow dataset for evaluation.

TABLE I
BENCHMARK DATASETS

Dataset	Slice Dim	Time Dim	Range	$\#\mathbf{CP_t}$	$\#\mathbf{CP_{slab}}$	Size(MB)
SCF	450×200	500	[-65.21, 54.11]	111576	237193	343.3MB
CFVKV	640×80	1501	[-0.93, 1.83]	6752	16141	586.3MB
HCBA	150×450	2001	[-1.25, 1.02]	44113	104388	1030.5 MB
FS	512×512	1001	[-0.90, 0.83]	5821	21040	2002MB

We conduct the experiments on a medium-sized HPC cluster [59]. Each node is equipped with AMD EPYC 7702 CPUs and 512 GB of DDR4 memory, running CentOS 8.4 and GCC 9.3.

B. Baselines

To the best of our knowledge, apart from lossless compression, no existing lossy compressor is capable of preserving the critical point trajectories in temporal 2D vector field data. Since lossless compressors naturally guarantee full topological fidelity, we use them as upper-bound baselines for comparison. In addition, we evaluate three representative lossy compressors: ZFP, SZ3, and cpSZ-SoS to investigate how lossy techniques perform in terms of critical point trajectories preservation.

C. Metrics

Our goal is to ensure that the critical points remain consistent before and after compression and to preserve the temporal coherence of their trajectories. We employ the following metrics for quantitative evaluation:

- **Compression Ratio (CR):** Measures the data reduction capability, defined as $\text{CR} = \frac{S_{\text{orig}}}{S_{\text{comp}}}$, where S_{orig} and S_{comp} denote the sizes of the original and compressed data, respectively.
- **Peak Signal-to-Noise Ratio (PSNR):** Evaluates overall field fidelity in the value domain, defined as $\text{PSNR} = 20 \cdot \log_{10}(\text{data range}) - 10 \cdot \log_{10}(\text{MSE})$.
- **False Cases of Critical Points:** This metric quantifies the incorrect detection or correspondence of critical points, including both false positives (spurious critical points) and false negatives (missing critical points). It is measured at two temporal scales: (a) Spatial level at time t (FC_t). (b) Space–time level at time slab$[t_k, t_{k+1}]$ (FC_s).
- **Number of Trajectories:** The number of critical point trajectories formed by connecting critical points detected at each time step t through their correspondences across consecutive time slabs.

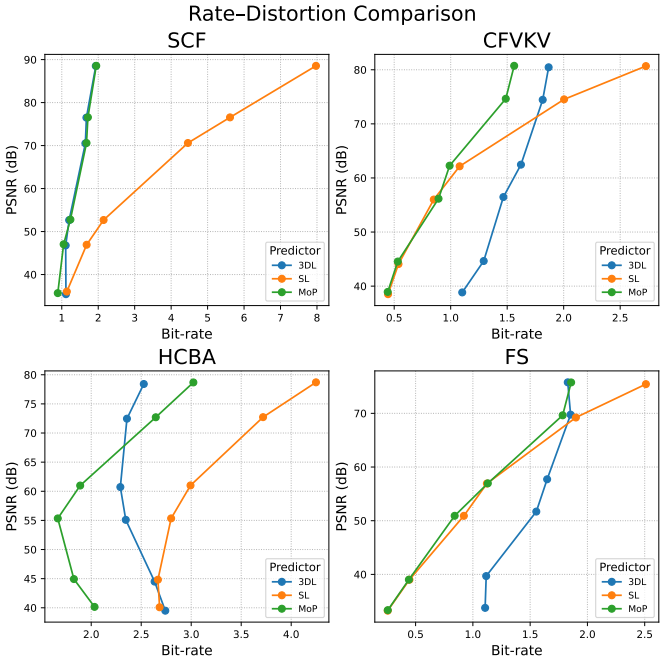


Fig. 5. Rate-distortion under different datasets.

D. Rate-distortion

Figure. 5 presents the rate distortion curves of our three methods across four different datasets, along with the proportion of blocks in which the MoP framework selects SL as the predictor. The bottom x-axis denotes the bit rate, computed as the data type size in bit divided by the compression ratio, while the left y-axis repesents PSNR. The top x-axis and right y-axis further indicate, for each error bound, the fraction of data block in the MoP that adopt SL as the selected predictor. We observe that under the same PSNR, the SL

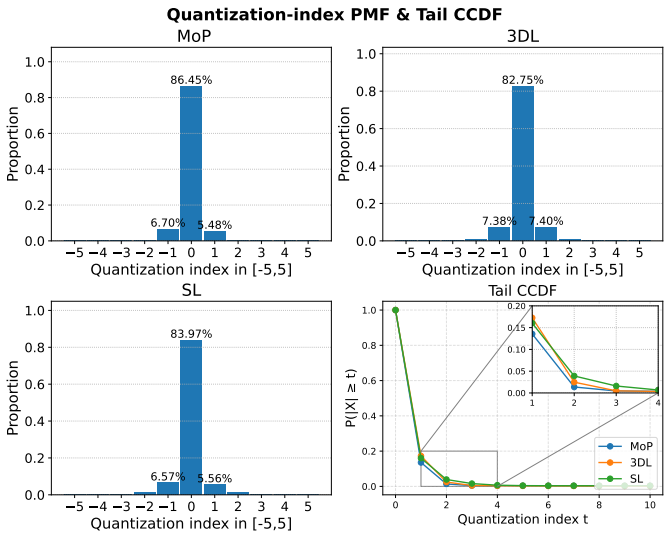
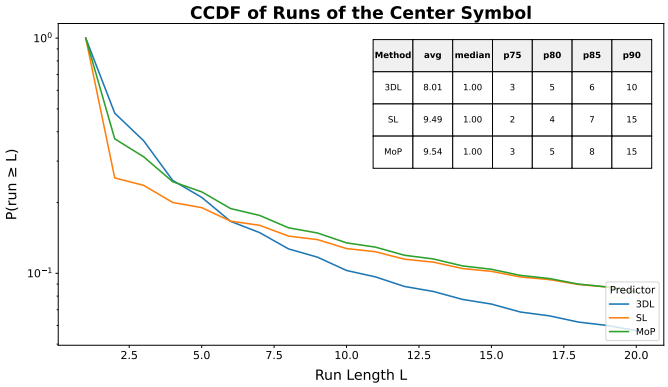


Fig. 6. Probability Mass Function(PMF) and complementary cumulative distribution function(CCDF) of three on the SCF dataset at eb=5.



Method	avg	median	p75	p80	p85	p90
3DL	8.01	1.00	3	5	6	10
SL	9.49	1.00	2	4	7	15
MoP	9.54	1.00	3	5	8	15

Fig. 7. CCDF of run length between 0 and 20 on SCF dataset at eb=5.

method perform exceptionally well in compression ratio on the SCF and HCBA dataset, while the 3DL method achieves better compression ratio on the SCF and FS dataset. In contrast, the MoP method effectively combines the strengths of both approaches, achieving a compression ratio that either surpasses or closely matches the best of the other two methods.

E. Encoding Efficiency

In our pipeline, two factors dominate the final bitrate: the fixed error bound after quantization $\{Q_\xi\}$that guarantees critical-point tracking after quantization, and the distribution of the quantization indices $\{Q_d\}$. The error bound is analytically derived from the requirement and is therefore not a tunable lever for further compression. Consequently, our effort focuses on shaping the index distribution via better prediction so that the subsequent entropy and lossless coders operate more efficiently. The rate gains of MoP come from two layers: (1) **Zero-order statistics.** Let $p(k)$ denote the PMF of the folded quantization index (central bins correspond to $|q| \leq 1$ under our toward-zero mapping). The ideal per-sample cost of Huffman coding is $H_0(p) = -\sum_k p(k) \log_2 p(k)$. By

concentrating probability mass near the center, MoP reduces $H_0(p)$ compared with 3DL and the SL predictor. Figure.6 plots the PMFs of the three methods on the SCF dataset at $\texttt{eb} = 5$; MoP exhibits a markedly sharper central peak (i.e., more indices in the interval $[-1, 1]$). Meanwhile, the tail CCDF illustrates the relative proportions of data in both tails. It can be seen that MoP exhibits a smaller proportion beyond the range $[-3, 3]$, suggesting a higher concentration of data around the center which directly translates into shorter Huffman codewords and a lower first-order rate. (2) **Higher-order structure.** After entropy coding, the bytestream is fed to a backend lossless compressor (zstd), which exploits repetition and context (LZ77-style matching) beyond zero-order statistics. Figure 7 shows the complementart cumulative distribution function(CCDF) of the run lengths for the centrer symbol obtained from three different predictors: 3DL, SL, and MoP. The x-axis represents the run length L, while the y-axis denotes the probabilty $P(run \geq L)$, i.e., the proportion of continuous run whose length is greater than or equal to L. This figure provides a statistical view of how long the correctly predicted regions persist continuously across space and time. In other words, a fatter tail corresponds to more regular and persistent higher-order structures. According to the figure, we can observe that both SL and MoP are able to capture more longer run-lengths compared with 3DL. In addition, the upper-right panel of the figure shows the corresponding means, and the 75th, 80th, 85th, and 90th percentiles for the three methods, which further confirm this observation. Because zstd leverages repeated substrings, a longer zero-run yields longer matches and thus additional savings on top of Huffman, further improving the overall compression ratio.

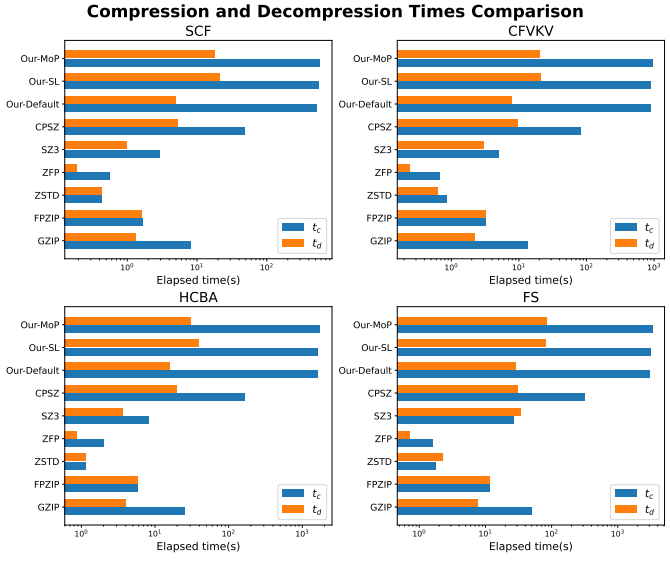


Fig. 8. Compression and decompression time(t_c and t_d) comparison under four datasets.

F. Comparison with State of the Arts

We present the quantitative results for all four datasets in Table.II–V. All reported experimental runtimes correspond to serial execution without any parallelization. To ensure a fair comparison, we adjusted the parameters of SZ3 and cpSZ so that their compression ratios are as close as possible to that of our method in the default (3DL) configuration. According to the results, since SZ3 does not provide any guarantee of preserving critical points, it consequently fails to ensure the correctness of critical point trajectories. Although cpSZ can preserve the critical points at each indivual time step, it cannot guarantee the consistency of critical points within a time slab, which lead to inconsistent trajectory counts and the emergence of false cases within the slab. Among all the lossless compression frameworks, FPZIP achieves the best performance, with a maximum compression ratio of 3.31. The results of GZIP and ZSTD are comparable, with compression ratios ranging between 1.1 and 1.3.

On the other hand, all three of our proposed methods effectively preserve critical point trajectories. Among them, MoP consistently outperforms 3DL across all four datasets, with particularly notable improvements on the CFVKV and FS datasets, achieving $72.29\times$ and $124.48\times$ compression ratios, respectively. Furthermore, when using SL alone as the predictor, it surpasses 3DL on three of the datasets with a maximum improvement of 32%, and performs comparably to 3DL on the SCF dataset. Overall, among all compressor capable of preserving critical point trajectories, our method achieve up to $56.07\times$ in compression ratio compared to the state-of-the-art

TABLE II
QUANTITATIVE RESULTS ON SCF DATA

Compressors	*Type*	*Settings*	CR_{all}	*PSNR*	$\#FC_t$	$\#FC_s$	$\#Traj$
GZIP	Lossless	/	1.10	/	0	0	999
ZSTD			1.10		0	0	999
FPZIP			3.31		0	0	999
ZFP	Lossy	$\epsilon = 0.5$	34.74	82.63	128640	347383	4467
SZ3		$\epsilon =$ 2E-4	34.18	122.71	34166	89657	4399
cpSZ(SoS)		$\epsilon = 5$	30.90	35.51	0	23486	2690
Our method		Default $\epsilon = 5$	28.71	35.43	0	0	999
		SL $\epsilon = 5$	28.10	36.04	0	0	999
		MoP$\epsilon = 5$	35.93	35.68	0	0	999

TABLE III
QUANTITATIVE RESULTS ON CFVKV DATA

Compressors	*Type*	*Settings*	CR_{all}	*PSNR*	$\#FC_t$	$\#FC_s$	$\#Traj$
GZIP	Lossless	/	1.32	/	0	0	37
ZSTD			1.32		0	0	37
FPZIP			2.59		0	0	37
ZFP	Lossy	$\epsilon = 0.4$	64.91	56.94	30274	75668	815
SZ3		$\epsilon =$ 6E-4	69.41	81.35	42116	133681	3825
cpSZ(SoS)		$\epsilon = 0.1$	30.71	39.31	0	414	39
Our method		Default $\epsilon = 0.1$	29.04	38.84	0	0	37
		SL $\epsilon = 0.1$	71.82	38.51	0	0	37
		MoP $\epsilon = 0.1$	72.29	38.93	0	0	37

TABLE IV
QUANTITATIVE RESULTS ON HCBA DATA

Compressors	*Type*	*Settings*	CR_{all}	*PSNR*	$\#FC_t$	$\#FC_s$	$\#Traj$
GZIP	Lossless	/	1.13	/	0	0	167
ZSTD			1.10		0	0	167
FPZIP			2.57		0	0	167
ZFP	Lossy	$\epsilon =$ 1E-3	14.31	96.57	76681	284941	8140
SZ3		$\epsilon =$ 4E-5	15.79	101.56	134907	446320	12110
cpSZ(SoS)		$\epsilon =$ 5E-3	15.14	61.08	0	9604	314
Our method		Default $\epsilon = 0.1$	11.68	39.50	0	0	167
		SL $\epsilon = 0.1$	11.92	40.09	0	0	167
		MoP $\epsilon = 0.1$	15.74	40.15	0	0	167

TABLE V
QUANTITATIVE RESULTS ON FS DATA

Compressors	*Type*	*Settings*	CR_{all}	*PSNR*	$\#FC_t$	$\#FC_s$	$\#Traj$
GZIP	Lossless	/	1.12	/	0	0	40
ZSTD			1.11		0	0	40
FPZIP			2.22		0	0	40
ZFP	Lossy	$\epsilon = 4$	121.72	35.80	12362	51535	107
SZ3		$\epsilon =$ 8E-4	125.59	75.51	1847	8022	44
cpSZ(SoS)		$\epsilon = 0.1$	81.50	34.06	0	4415	121
Our method		Default $\epsilon = 0.1$	28.91	33.77	0	0	40
		SL $\epsilon = 0.1$	124.71	33.24	0	0	40
		MoP $\epsilon = 0.1$	124.48	33.34	0	0	40

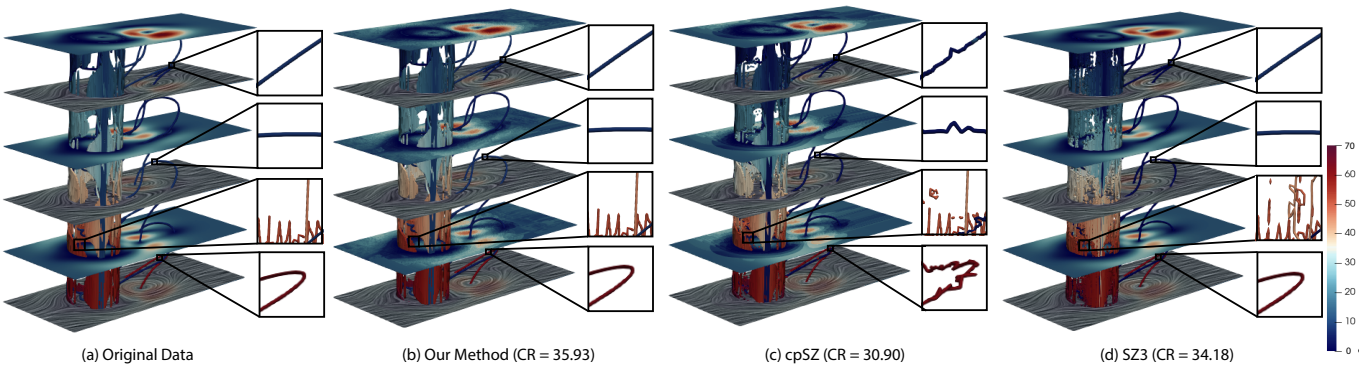


Fig. 9. Visualization of critical point trajectories in the SCF dataset.

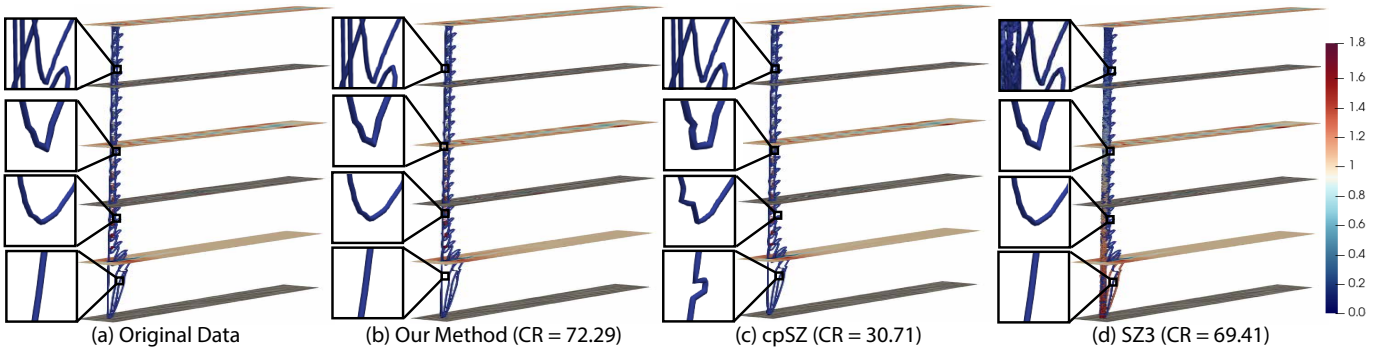


Fig. 10. Visualization of critical point trajectories in the CFVKV dataset.

compressors across all datasets. For compression time, ZSTD is the fastest among all lossless compressors, while SZ3 is the fastest among lossy compressors.

Figure. 8 shows the compression and decompression time across all the compressors. ZSTD is the fastest among all lossless compressors, while SZ3 achieves the highest speed among lossy compressors. Compared with cpSZ, our methods are more time-consuming because they aim to preserve trajectories, which requires examining up to 36 surrounding triangles for each point, including pre-computation of critical points and derivation of error bounds. Among our three proposed methods, 3DL and SL exhibit similar compression times, whereas MoP requires more time due to the additional block partitioning, sampling, and micro-encoding processes used for score calculation. For decompression time, 3DL and SL perform almost identically, while MoP incurs a slightly higher cost because it must switch predictors across different data blocks.

Although our algorithm is slower than some existing lossy compressors, it uniquely preserves the critical point trajectories throughout the compression–decompression process, ensuring topological consistency in the reconstructed vector fields. Meanwhile, the framework is highly amenable to parallelization, which can substantially improve its performance. One effective strategy is inter-block parallelization [47], where multiple data blocks are processed concurrently with a one-layer halo to remove inter-block dependencies. Within each block, all critical points are precomputed in parallel, followed by error-bound derivation and prediction. Moreover, our SL predictor achieves an even higher degree of parallelism than the traditional Lorenzo predictor, since it is embarrassingly parallel across all data within frame t without depending on any current-frame neighbors. Consequently, the MoP framework can be fully parallelized on CPUs, and existing studies [60]–[63] have demonstrated that GPU-based prediction compressors can further accelerate throughput significantly. In addition, the decompression process in our framework is relatively fast, and since data are typically compressed once but decompressed multiple times, the additional compression overhead has little impact on practical usability.

G. Qualitative Visualizations

We visualize the trajectories of critical points in the original data and the decompressed data produced by our method (MoP), cpSZ, and SZ3 across four datasets in Figures 9–12. The error bound settings are kept consistent with those in Tables II–V.

Critical point trajectories are extracted using the tracking algorithm provided in FTK [49]. Specifically, for each 2-simplex, we regard all other 2-simplices that share the same 3-simplex as its neighbors. These adjacency relations are then grouped into connected components using a union–find structure. Nodes with degrees greater than two are identified as branching points, while the remaining nodes (with degrees $\geq$2) are traversed sequentially along adjacency links to form ordered vertex chains. If the first and last vertices are adjacent, the chain is marked as a closed loop.

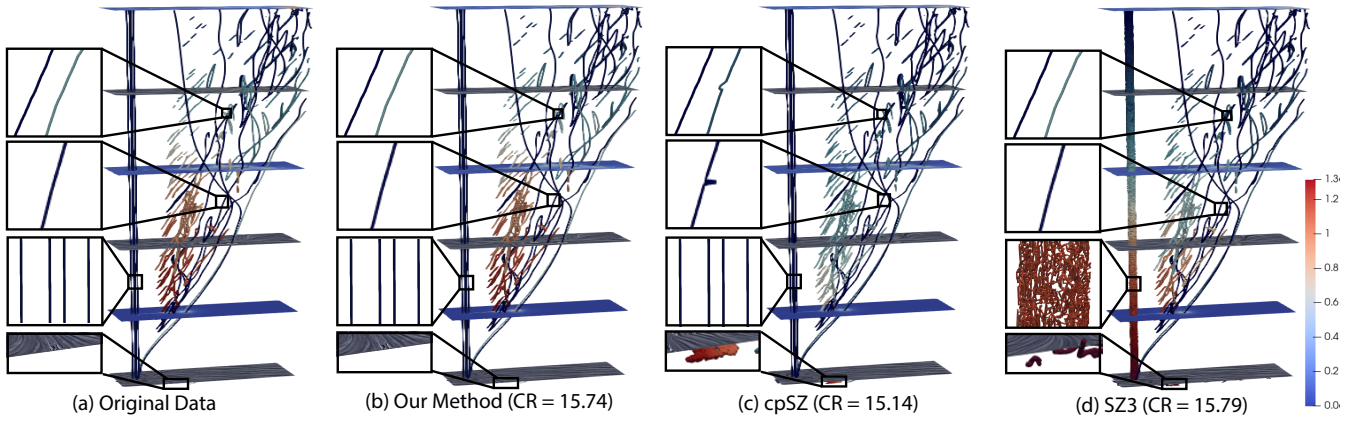


Fig. 11. Visualization of critical point trajectories in the HCBA dataset.

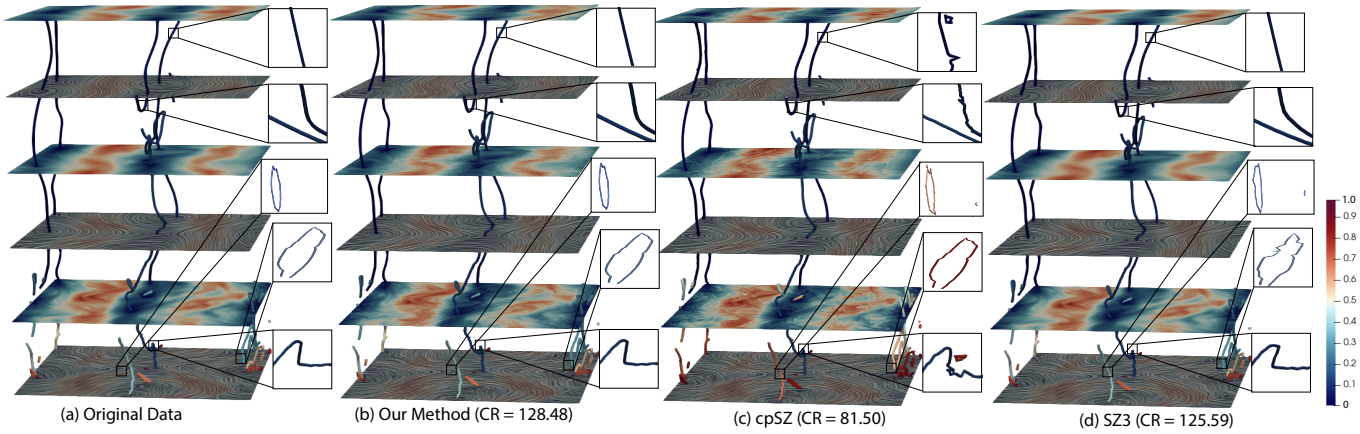


Fig. 12. Visualization of critical point trajectories in the FS dataset.

We overlay the line integral convolution (LIC) texture and the corresponding velocity field across time to display the critical point trajectories. As shown in the visualizations, our method completely preserves the original critical point trajectories. In contrast, SZ3 produces a large number of spurious trajectories in three of the four datasets except for FS, mostly located in regions where both u and v components are near zero. Moreover, both cpSZ and SZ3 distort certain trajectories in various ways, including but not limited to: (1) discontinuous trajectories accompanied by small oscillations, exhibiting zigzag patterns in the trajectories, which indicate slight positional deviations of critical points in the decompressed data; and (2) short-lived artifacts, manifested as "small bubbles" that suddenly appear, split and merge within a short time slab, where these critical points do not exist in the original data.

VIII. Conclusion

In this paper, we address the problem of preserving critical point trajectories during the lossy compression of 2D vector field data. We propose a semi-Lagrangian–based predictor, which achieves up to 4.31× higher compression ratios than the traditional Lorenzo predictor in advection-dominated datasets. Furthermore, we design a Mixture-of-Predictors module that adaptively selects the most suitable predictor for each local data block based on its spatial characteristics. This adaptive mechanism effectively reduces residual entropy and maintains a high compression ratio, achieving up to 56.07× over state-of-the-art lossless compressors in our experiments. In future work, we will further enhance the efficiency of our framework in terms of both compression ratio and throughput, and develop a GPU-accelerated implementation to achieve higher performance.

AI-Generated Content Acknowledgement

Portions of the text in this manuscript were refined using language-assistance tools such as Grammarly and ChatGPT(GPT-5). These tools were employed solely to improve grammar, clarity, and readability. All conceptualization, data analysis, and scientific conclusions are entirely the work of the authors.

References

[1] "Frontier exscale supercomputer," https://www.olcf.ornl.gov/frontier.
[2] P. Yeung, K. Ravikumar, R. Uma-Vaideswaran, D. L. Dotson, K. R. Sreenivasan, S. B. Pope, C. Meneveau, and S. Nichols, "Small-scale properties from exascale computations of turbulence on a periodic cube," *Journal of Fluid Mechanics*, vol. 1019, p. R2, 2025.

[3] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers *et al.*, "The era5 global reanalysis," *Quarterly journal of the royal meteorological society*, vol. 146, no. 730, pp. 1999–2049, 2020.

[4] T. Günther, M. Gross, and H. Theisel, "Generic objective vortices for flow visualization," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[5] A. Friederici, C. Rössl, and H. Theisel, "Finite time steady 2d vector field topology," in *Topological Methods in Data Analysis and Visualization*. Springer, 2015, pp. 253–266.

[6] L. Fritschi, I. B. Rojo, and T. Günther, "Visualizing the temporal evolution of the universe from cosmology simulations," in *2019 IEEE Scientific Visualization Conference (SciVis)*. IEEE, 2019, pp. 1–11.

[7] N. Rimensberger, M. Gross, and T. Günther, "Visualization of clouds and atmospheric air flows," *IEEE Computer Graphics and Applications*, vol. 39, no. 1, pp. 12–25, 2019.

[8] 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.

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

[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] K. Zhao, S. Di, M. Dmitriev, T.-L. D. Tonellot, Z. Chen, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data by dynamic spline interpolation," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1643–1654.

[13] 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.

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

[15] 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.

[16] X. Liang, B. Whitney, J. Chen, L. Wan, Q. Liu, D. Tao, J. Kress, D. R. Pugmire, M. Wolf, N. Podhorszki, and S. Klasky, "Mgard+: Optimizing multilevel methods for error-bounded scientific data reduction," *IEEE Transactions on Computers*, 2021.

[17] 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.

[18] 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," *Proceedings of the ACM on Management of Data*, vol. 2, no. 1, pp. 1–27, 2024.

[19] Z. Jian, S. Di, J. Liu, K. Zhao, X. Liang, H. Xu, R. Underwood, S. Wu, J. Huang, Z. Chen, and F. Cappello, "Cliz: Optimizing lossy compression for climate datasets with adaptive fine-tuned data prediction," in *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2024, pp. 417–429.

[20] D. Wang, J. Pulido, P. Grosset, S. Jin, J. Tian, J. Ahrens, and D. Tao, "Optimizing error-bounded lossy compression for three-dimensional adaptive mesh refinement simulations," *arXiv preprint arXiv:2204.00711*, 2022.

[21] A. M. Gok, S. Di, Y. Alexeev, D. Tao, V. Mironov, X. Liang, and F. Cappello, "Pastri: Error-bounded lossy compression for two-electron integrals in quantum chemistry," in *2018 IEEE International Conference on Cluster Computing*. IEEE, 2018, pp. 1–11.

[22] M. Scheffer, J. Bascompte, W. A. Brock, V. Brovkin, S. R. Carpenter, V. Dakos, H. Held, E. H. Van Nes, M. Rietkerk, and G. Sugihara, "Early-warning signals for critical transitions," *Nature*, vol. 461, no. 7260, pp. 53–59, 2009.

[23] X. Yang, Z.-H. Wang, and C. Wang, "Critical transitions in the hydrological system: early-warning signals and network analysis," *Hydrology and Earth System Sciences*, vol. 26, no. 7, pp. 1845–1856, 2022.

[24] X. Liang, H. Guo, S. Di, F. Cappello, M. Raj, C. Liu, K. Ono, Z. Chen, and T. Peterka, "Toward feature-preserving 2d and 3d vector field compression." in *PacificVis*, 2020, pp. 81–90.

[25] 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.

[26] 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.

[27] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields," in *Computer Graphics Forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 343–348.

[28] Y. Collet, "LZ4," https://github.com/lz4/lz4, online.

[29] 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.

[30] S. Di and F. Cappello, "Fast error-bounded lossy hpc data compression with SZ," in *2016 IEEE International Parallel and Distributed Processing Symposium*. Chicago, IL, USA: IEEE, 2016, pp. 730–739.

[31] 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.

[32] 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.

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

[34] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola, "Tthresh: Tensor compression for multidimensional visual data," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 9, pp. 2891–2903, 2019.

[35] 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.

[36] J. Liu, S. Di, K. Zhao, X. Liang, Z. Chen, and F. Cappello, "Faz: A flexible auto-tuned modular error-bounded compression framework for scientific data," in *Proceedings of the 37th ACM International Conference on Supercomputing*, ser. ICS '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1–13. [Online]. Available: https://doi.org/10.1145/3577193.3593721

[37] 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.

[38] ——, "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.

[39] P. Jiao, S. Di, H. Guo, K. Zhao, J. Tian, D. Tao, X. Liang, and F. Cappello, "Toward quantity-of-interest preserving lossy compression for scientific data," *Proc. VLDB Endow.*, vol. 16, no. 4, p. 697–710, Dec. 2022. [Online]. Available: https://doi.org/10.14778/3574245.3574255

[40] J. Liu, P. Jiao, K. Zhao, X. Liang, S. Di, and F. Cappello, "Qpet: A versatile and portable quantity-of-interest-preservation framework for error-bounded lossy compression," *Proc. VLDB Endow.*, vol. 18, no. 8, p. 2440–2453, Sep. 2025. [Online]. Available: https://doi.org/10.14778/3742728.3742739

[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] U. A. LLC and WSU, "SZ Lossy Compressor: An open and transparent lossy compression framework." https://szcompressor.org/, online.
[44] Y. Li, X. Liang, B. Wang, Y. Qiu, L. Yan, and H. Guo, "Msz: An efficient parallel algorithm for correcting morse-smale segmentations in error-bounded lossy compressors," *IEEE Transactions on Visualization and Computer Graphics*, 2024.
[45] Y. Li, M. Xia, X. Liang, B. Wang, and H. Guo, "Multi-tier preservation of discrete morse smale complexes in error-bounded lossy compression," 2025. [Online]. Available: https://arxiv.org/abs/2409.17346
[46] J. Tierny and V. Pascucci, "Generalized topological simplification of scalar fields on surfaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2005–2013, 2012.
[47] M. Xia, B. Wang, Y. Li, P. Jiao, X. Liang, and H. Guo, "Tspsz: An efficient parallel error-bounded lossy compressor for topological skeleton preservation," in *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 2025, pp. 3682–3695.
[48] N. Gorski, H. Guo, X. Liang, and B. Wang, " TFZ: Topology-Preserving Compression of 2D Symmetric and Asymmetric Second-Order Tensor Fields ," *IEEE Transactions on Visualization & Computer Graphics*, 2026, to appear.
[49] H. Guo, D. Lenz, J. Xu, X. Liang, W. He, I. R. Grindeanu, H.-W. Shen, T. Peterka, T. Munson, and I. Foster, "Ftk: A simplicial spacetime meshing framework for robust and scalable feature tracking," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 8, pp. 3463–3480, 2021.
[50] H. Theisel and H.-P. Seidel, "Feature flow fields," in *Proceedings of the Symposium on Data Visualisation 2003*, ser. VISSYM '03. Goslar, DEU: Eurographics Association, 2003, p. 141–148.
[51] J. De Loera, J. Rambau, and F. Santos, *Triangulations: structures for algorithms and applications*. Springer Science & Business Media, 2010, vol. 25.
[52] 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
[53] Z. Liu, S. Tan, A. Warmuth, F. Schuller, Y. Hong, W. Huang, Y. Gu, B. Zhu, G. Tan, and D. Tao, "Solarzip: An efficient and adaptive compression framework for solar euv imaging data-application to solar orbiter/eui data," *Astronomy & Astrophysics*, vol. 702, p. A160, 2025.
[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] C. Jung, T. Tél, and E. Ziemniak, "Application of scattering chaos to particle transport in a hydrodynamical flow," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 3, no. 4, pp. 555–568, 1993. [Online]. Available: https://doi.org/10.1063/1.165960
[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] J. Jakob, M. Gross, and T. Günther, "A fluid flow data set for machine learning and its application to neural flow map interpolation," *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization)*, 2020.
[59] "Morgan Compute Cluster," https://docs.ccs.uky.edu, 2023, online.
[60] J. Tian, C. Rivera, and D. Tao, "cuSZ: CUDA-Based Error-Bounded Lossy Compressor for Scientific Data," https://github.com/szcompressor/cuSZ, online.
[61] Y. Huang, S. Di, X. Yu, G. Li, and F. Cappello, "cuszp: An ultra-fast gpu error-bounded lossy compression framework with optimized end-to-end performance," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3581784.3607048
[62] J. Tian, S. Di, X. Yu, C. Rivera, K. Zhao, S. Jin, Y. Feng, X. Liang, D. Tao, and F. Cappello, "Optimizing error-bounded lossy compression for scientific data on gpus," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, ser. CLUSTER '21, Portland (virtual event), OR, USA, 09 2021.
[63] J. Liu, J. Tian, S. Wu, S. Di, B. Zhang, R. Underwood, Y. Huang, J. Huang, K. Zhao, G. Li, D. Tao, Z. Chen, and F. Cappello, "CUSZ-*i*: High-Ratio scientific lossy compression on GPUs with optimized multi-level interpolation," in *SC '24: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '24, Atlanta, GA, USA, 11 2024, co-first authors: Jinyang Liu, Jiannan Tian, and Shixun Wu. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SC41406.2024.00019