A Topology-Preserving Coreset for Kernel Regression in Scientific Visualization

Weiran Lyu, Nathaniel Gorski, Jeff M. Phillips, and Bei Wang

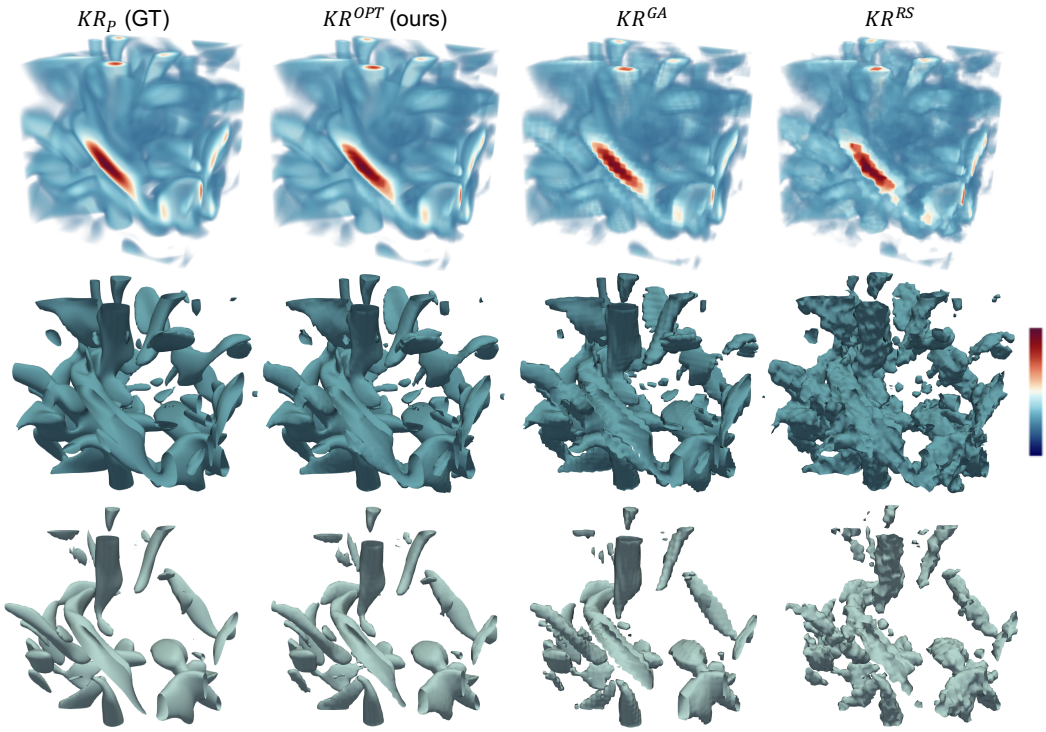

Fig. 1: Visualizations of a 3D Vortex dataset modeled by kernel regressions with a kernel bandwidth of 1. From left to right: kernel regression models based on the original data (KR_P) considered as the ground truth (GT), our optimized coreset ($\mathsf{KR}^{\mathsf{OPT}}$), grid-aggregated coreset ($\mathsf{KR}^{\mathsf{GA}}$), and randomly sampled coreset ($\mathsf{KR}^{\mathsf{RS}}$). From top to bottom: volume rendering of each kernel regression model (top row), isosurface renderings at isovalue 4 (middle row) and 5 (bottom row). With approximately 1.5% of the original data size, our optimized coreset construction achieves high-quality visualization while preserving vortex structures more effectively than state-of-the-art coreset methods based on grid aggregation and random sampling.

Abstract— Modern simulations and observations generate vast amounts of data, fueling a growing interest in replacing discrete data with continuous surrogate models, such as functional models and implicit neural networks, to enhance data storage, transfer, analysis, and visualization. To that end, kernel regression is a well-known class of non-parametric techniques useful for surrogate modeling. In this paper, we propose a new framework to use coresets for kernel regression—a small dataset that is used as a proxy for the original data—in scientific visualization. Using kernel regression as a surrogate for scientific data, we construct an optimized coreset that is both compact and highly accurate, reducing errors from randomly-sampled and grid-based coresets by orders of magnitude. We evaluate our framework on large spatial datasets and demonstrate that it incurs negligible error while preserving the underlying topological features.

Index Terms—Kernel regression, coreset, comparative measures, topological data analysis, scientific visualization, merge trees

1 Introduction

Modern simulations and observations generate vast amounts of data, presenting exciting opportunities and challenges for analysis and visualization. There has been growing interest in replacing discrete data with continuous surrogates to improve data storage, transfer, and analysis. These surrogates—such as splines [28, 33, 34], reduced-order models [37], and implicit neural networks [29, 30, 48, 49]—aim to represent complex systems' behavior through simplified functions, enabling the analysis and visualization of complex data that would otherwise be too large to handle.

On the other hand, topology plays a key role in the identification, tracking, and visualization of features in scientific data, including burning cells in turbulent combustion [5], vortices in fluid dynamics simulation [53], and pores in a porous material [19]. To support topological data analysis and visualization, surrogates could be designed to preserve important topological features. To that end, we propose kernel regression as a reliable, adaptable, and topology-preserving surrogate for visualizing scientific data.

• *Weiran Lyu, Nathaniel Gorski, Jeff M. Phillips, and Bei Wang are with the University of Utah. E-mails: wlyu@sci.utah.edu, gorski@sci.utah.edu, jeffp@cs.utah.edu, beiwang@sci.utah.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Kernel regression is a widely-used procedure to convert discrete datasets into continuous models, dating back to the 1960s [31, 55]. It provides valuable properties in terms of stability, robustness, and data smoothing. However, while kernel regression is widely used in statistics [41, 47] and machine learning [18, 40], its application in scientific data visualization is largely underexplored. One specific property that remains underutilized is the use of coresets [35] for well-structured functional approximations such as kernel regression [60]. Coresets are highly-compact subset representations that ensure the preservation of essential data properties.

Contributions. In this paper, we advocate for kernel regression as a smoothing surrogate model for scientific visualization. This offers several notable advantages in modeling, by enabling enormous data reduction through coresets while provably preserving key topological features. Our contributions are as follows:

- We propose a novel optimized coreset method for kernel regression using gradient descent that is easy to implement and achieves error reduction by several orders of magnitude, compared to state-of-the-art coreset methods based on grid aggregation and random sampling.
- Using coresets for kernel regression, we provide a strong theoretical guarantee regarding the stability of topological descriptors such as persistence diagrams and merge trees.
- Through experiments, we demonstrate that our optimized coresets achieve exceptionally high accuracy at approximately 0.25% of the original data size for 2D and 1.5% for 3D datasets. Moreover, they are highly effective for topology-based scientific visualization, preserving topological descriptors such as isosurfaces, merge trees, and persistence diagrams. The code is available at https://github.com/tdavislab/coreset-kr/.

Overall, our framework is the first of its kind to utilize coresets for kernel regression in scientific visualization. Our optimized coreset method achieves negligible error, making it particularly effective as a surrogate for large-scale time-varying datasets and ensembles.

2 Related Work

In this section, we give a brief review of surrogate models for scientific visualization and kernel regression.

Surrogate modeling for scientific visualization. Surrogate models have become powerful tools in scientific visualization, enabling tasks such as data reduction, parameter space exploration, and ensemble simulation [23, 54].

To reduce data storage, surrogate models have been used to compress scalar fields via learned representations. Lu et al. [26] proposed a neural implicit model that approximates scalar fields by mapping spatial coordinates to scalar values. Gobbetti et al. [13] introduced COVRA, which builds an octree of cubical bricks from a volumetric dataset and learns a sparse linear combination using K-SVD on a coreset. Liu et al. [25] developed an in-situ compression method using Generative Adversarial Networks (GANs) for computational fluid dynamics data. These approaches treat compression as function approximation, enabling efficient rendering and storage. However, they typically require trading off precision or reducing data resolution (e.g., decreasing the number of input data points [17]).

From the reconstruction perspective, surrogate models have been used in the simulation process. Shi et al. [46] introduced a Graph Neural Network (GNN) based surrogate that models unstructured mesh ocean simulations for adaptive parameter space sampling. He et al. [16] developed InSituNet, using a GAN to generate ensembles of volumes based on learning conditioned on parameter spaces. Shi et al. [45] also explored a view-dependent latent-space surrogate model designed to preserve visual coherence in the rendered output.

In summary, existing surrogate modelings in scientific visualization are dominated by deep learning models—particularly implicit neural networks, convolutional neural networks, and graph-based architectures—designed to approximate scalar fields or full simulations. However, there are few prior studies exploring the use of non-parametric surrogates such as kernel regression, or explicitly consider topology-preservation as a design objective. Recent work by Ma et al. [27, 28] explored the extraction of topology from splines models known as Multivariate Functional Approximation (MFA) models.

Kernel regression. Kernel regression is widely-used for prediction, modeling, and visualization across diverse fields. It has found applications in economics [4, 57], image processing [51], weather forecasting [42], and spatial data analytics [59]. In this work, we focus specifically on the Nadaraya–Watson (NW) formulation of kernel regression. Alternative kernel-based methods exist. For example, kernel ridge regression, which forms the foundation of Gaussian processes [38], offers a probabilistic interpretation and enables uncertainty quantification.

Despite the widespread use of kernel regression in data analysis, its application to scientific visualization remains underexplored. Our research establishes the first connection between kernel regression, coreset, and scientific visualization, offering a novel perspective on surrogate modeling in this field. Our optimized coreset framework for kernel regression not only enables efficient data approximation but also provides theoretical guarantees on preserving topological features for visualization.

3 Background

In this section, we first review technical background on kernel regression and coresets, followed by a discussion on the stability of topological descriptors.

3.1 Coresets for Kernel Regression

Kernel regression is a non-parametric regression technique that (with an appropriate kernel choice) provides a smooth and robust estimate for a function observed over a discrete domain. It takes as input a dataset $P = (X, y) = \{p_i\} = \{(x_i, y_i)\}$ (where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$) and provides a continuous function $\text{KR}_P : \mathbb{R}^d \to \mathbb{R}$. It also requires a choice of a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$, i.e., a bi-variate similarity function defined on $\mathbb{R}^d$. In this work, we adopt the (most commonly used) Gaussian kernel

$$K(x, x') = \exp(-\|x - x'\|^2/2\sigma^2).$$

Here, σ is the bandwidth parameter that defines the approximate distance within which items are considered similar. While a Gaussian distribution may include a normalization coefficient in front of the $\exp(\cdot)$ term, it is not necessary in this case, as it will cancel out. As a result, we can restrict the range of K to $[0, 1]$.

A kernel regression function $f_P(\cdot)$ is generally of the form

$$f_P(x) = \sum_{p_i \in P} \alpha_i K(x_i, x).$$

There are two main forms of kernel regression. In this paper, we focus on *Nadaraya-Watson (NW) kernel regression* [31,55], where each $\alpha_i = y_i / \left(\sum_{p_i \in P} K(x_i, x)\right)$ is a function of the query point x, and the NW kernel regression function is defined as

$$\begin{aligned}\text{KR}_P(x) &= \sum_{p_i \in P} \frac{y_i}{\sum_{p_i \in P} K(x_i, x)} K(x_i, x) \\ &= \frac{\frac{1}{|P|}\sum_{p_i \in P} y_i K(x_i, x)}{\frac{1}{|P|}\sum_{p_i \in P} K(x_i, x)}.\end{aligned}$$

The denominator represents the kernel density estimate (KDE), i.e., $\text{KDE}_X(x) = \frac{1}{|P|}\sum_{x_i \in X} K(x_i, x)$, while the numerator is a KDE weighted by the response values y_i. When a smooth, continuous kernel is used (such as the Gaussian), it ensures that KR_P smoothly and robustly averages the y_i values. When the gradient of the kernel K is controlled (e.g., it is at most $1/\sigma$ for the Gaussian), the contribution of a nearby point x_i to the weighted average changes gradually as x approaches x_i. Consequently, the predicted response also changes smoothly. For exponentially decaying kernels (e.g., the Gaussian with a squared-exponential decay), when all domain points x_i are far from the query point x, the model predicts approximately the value y_i of the closest domain point x_i.

Another common kernel regression variant is *kernel ridge regression*, which provides a fixed choice of the α_i values, independent of the query point x. These α_i values are trained using ridge regression to minimize the L_2 error to the response value y_i. The ridge term has a hyperparameter that induces a bias in the model which has three effects: it favors smaller α_i which makes for less variable models; it keeps the function f_P from interpolating exactly the recorded response values; and it can in some cases prevent overfitting. Because this method requires an extra training step, an extra ridge parameter, and the α_i are less interpretable, we favor the Nadaraya-Watson variant in this paper.

Coresets. Given an input dataset P, a coreset S is a carefully constructed smaller set that approximates P. The coreset S serves as a proxy for P. For example, S may be a subset of P; however, each $x \in S$ may also differ from the original points in P. A key property of a coreset S is that it provides strong approximation guarantees for the original dataset P with respect to a given measure, for example, performance in kernel regression. The study of coresets typically focuses on how these approximation guarantees scale with the coreset size $|S|$, which is often independent of the size of the original dataset $|P|$. Coreset methods are particularly useful when the original dataset P is very large, as a coreset S can be substantially smaller than P while still maintaining strong approximation guarantees. For a more detailed introduction to other types of coresets, see the survey by Phillips [35].

Coresets for NW kernel regression. We review several existing coreset constructions to compress spatial data for NW kernel regression.

Random sampling. A basic coreset construction is to draw a uniform random sample from P. The resulting simple coreset S is a subset of P where it preserves the sparsity of P and has interpretability. **Random sampling (RS)** is commonly used for data aggregation and to approximate KDE [12, 14, 20, 36]. Zheng and Phillips [60] showed that this technique provides a coreset for NW kernel regression.

Grid-based methods. Given $P \subset \mathbb{R}^d$, define a grid G_γ that is composed of cubical cells of size γ^d. For each cell g, define the points from P that are in g as $P_g := P \bigcap g$. A basic grid-based method, **Grid-Random (GR)**, randomly selects one point from each cell, $p = (x, y) \in P_g$, to place into S, assigning weight as $|P_g|$.

Zheng and Phillips [60] further extended the basic grid method to **Grid-Aggregate (GA)**. Instead of randomly selecting one point from each cell, GA takes the average point. For instance, in 2D, the average point per cell is defined as $s_g = (s_x, s_y)$ with $s_x = \frac{1}{|P_g|}\sum_{p\in P_g} x$ and $s_y = \frac{1}{|P_g|}\sum_{p\in P_g} y$. Each s_g is then placed into S.

Zheng and Phillips [60] also discussed sorting-based methods, which can be extended to higher dimensions using space-filling curves. However, since this paper focuses on scientific visualization, where the input data resides on a dense regular grid, these methods essentially become grid-based approaches, albeit with additional complexity and overhead. Therefore, we do not include comparisons with these methods.

In practice, the grid-based methods consistently outperform the random sampling and other coreset methods discussed above in 1D and 2D datasets [60]. This suggests that aggregating based on the locations and function values of the points is more effective. Grid compression can be applied when the data is modeled as a random sample. However, in higher dimensions, random sampling has better theoretical bounds than grid-based methods [60]. In this paper, we perform for the first time 3D exploration of grid-based methods.

Error bounds of coresets. Zheng and Phillips [60] considered coresets for NW kernel regression for a strong (worst-case) L_∞ error guarantee. That is, for a point set P and a coreset S, KR_S consistently approximates KR_P, ensuring that there are no spurious regression values.

Formally, this is expressed with a slight restriction by considering only a domain $\mathbb{U}_\rho = \{x \in \mathbb{R}^d \mid \text{KDE}_X(x) \geq \rho\}$. If $\text{KDE}_X(x) < \rho$ (for some small ρ), then $\text{KR}_P(x)$ can become highly unstable with respect to noise in x, since the weights $K(x, x')$ change exponentially with $\|x - x'\|^2$. As a result, $\text{KR}_P(x)$ can be written as a weighted average of these response values y, with $\text{KDE}_X(x)$ in the denominator.

In this work, we consider X as being a dense set of gridded points over a rectangular domain, and only seek to evaluate KR_P over this domain, which will be a subset of $\mathbb{U}_\rho$; so controlling errors outside of $\mathbb{U}_\rho$ is not relevant.

Now, for any query $q \in \mathbb{U}_\rho$ and parameters $\rho, \varepsilon \in (0, 1)$, we define an L_∞ norm within the region $\mathbb{U}_\rho$,

$$\|\text{KR}_P(q) - \text{KR}_S(q)\|_{\mathbb{U}_\rho,\infty} := \max_{q\in\mathbb{U}_\rho} |\text{KR}_P(q) - \text{KR}_S(q)| \leq \varepsilon M,$$

where $M = \max_{p_i,p_{i'}\in P} |y_i - y_{i'}|$ is the maximum difference between reported response values of P. We call a set S which satisfies this error bound is an (ρ, ε)-*coreset of* P.

Zheng and Phillips [60] showed that if one samples $s = O(\frac{1}{\varepsilon^2\rho^2}(d\log(1/\rho) + \log(2/\delta))$ points into S from P, then S is a (ρ, ε)-coreset of P with probability at least $1 - \delta$. They also showed that for any of the grid-based methods, if the length of a cell is set to $\gamma = \frac{\varepsilon\sigma\rho}{8\sqrt{d}}$, then with S the representative from each cell is a (ρ, ε)-coreset for P.

3.2 Topological Descriptors and Their Stability

We now review topological descriptors and their stability, in particular, persistence diagrams and merge trees; see [58] for a review.

Persistence diagrams and stability. A *persistence diagram* visualizes the lifespan of topological features (such as connected components, loops, or voids) in data analyzed at different scales. It is a popular tool in topological data analysis and visualization that captures the shape of data. As illustrated in Fig. 2, given a scalar field $f : \mathbb{X} \to \mathbb{R}$, we study the evolution of sublevel sets of f, denoted as $f_t := f^{-1}(-\infty, t]$, as t increases. The 0-dimensional persistence diagram in Fig. 2(d) shows the appearances and disappearances of connected components in f_t as t increases. Each point in the persistence diagram represents a topological feature (e.g., a connected component): its x- and y-coordinates encode when the feature appears (birth) and disappears (death) respectively. The (vertical) distance from the diagonal—referred to as *persistence*—is the significance of the feature. *Persistence simplification* removes noise from the data that has persistence less than a prescribed threshold [11]. See [10] for a technical introduction to persistence.

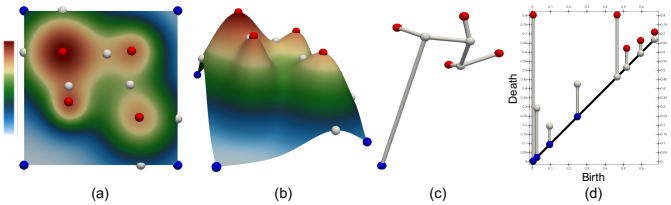

Fig. 2: Given a scalar field f in (a) whose graph is visualized in (b), we compute the merge tree of $-f$ (c) and 0-dimensional persistence diagram (d). Local maxima are in red, saddles are in white, and local minima are in blue.

We now review the bottleneck and Wasserstein distances between persistence diagrams. Given two persistence diagrams $\mathcal{D}_1$, $\mathcal{D}_2$ and a bijection $\eta : \mathcal{D}_1 \to \mathcal{D}_2$ (i.e., allowing points from one diagram to match those in another; points can also be matched to the diagonal), the *bottleneck distance* [6] between $\mathcal{D}_1$ and $\mathcal{D}_2$ is defined as

$$d_\infty(\mathcal{D}_1, \mathcal{D}_2) = \inf_{\eta:\mathcal{D}_1\to\mathcal{D}_2} \sup_{x\in\mathcal{D}_1} ||x - \eta(x)||_\infty. \tag{1}$$

The *q-Wasserstein distance* [7] is defined as

$$d_W^q(\mathcal{D}_1, \mathcal{D}_2) = \left[\inf_{\eta:\mathcal{D}_1\to\mathcal{D}_2} \sum_{x\in\mathcal{D}_1} ||x - \eta(x)||_\infty^q\right]^{\frac{1}{q}}. \tag{2}$$

In the experiments we set $q = 2$ and work with 2-Wasserstein distance d_W^2 (referred to as Wasserstein distance when clear from context).

d_∞ is stable [6] w.r.t. small perturbations of the input function.

Theorem 1 (Stability of Bottleneck Distance [6])**.** *Let $\mathbb{X}$ be a triangulable space with continuous tame functions, $f_1, f_2 : \mathbb{X} \to \mathbb{R}$. Then their persistence diagrams $\mathcal{D}_1, \mathcal{D}_2$ satisfy*

$$d_\infty(\mathcal{D}_1, \mathcal{D}_2) \leq ||f_1 - f_2||_\infty. \tag{3}$$

A topological space is called *triangulable* if there exists a (finite) simplicial complex whose underlying space is homeomorphic to it [6]. Additionally, *tameness* is a technical condition that guarantees that f_1 and f_2 behave well.

d_W^q is also proven to be stable for a reasonably large class of functions. In our setting, we employ a particular version of the Wasserstein Stability Theorem from [7].

Theorem 2 (Stability of Wasserstein Distance [7])**.** *Let $\mathbb{X}$ be a triangulable, compact metric space with bounded degree-1 total persistence (i.e. sum of persistence of all points in $\mathcal{D}_1$ and $\mathcal{D}_2$). Let $f_1, f_2 : \mathbb{X} \to \mathbb{R}$ be tame Lipschitz functions with Lipschitz constant $\mathrm{Lip}(f_1)$ and $\mathrm{Lip}(f_2)$ respectively. Then for every $q > 1$,*

$$d_W^q(\mathcal{D}_1, \mathcal{D}_2) \leq C^{\frac{1}{q}} \cdot ||f_1 - f_2||_\infty^{1-\frac{1}{q}}, \tag{4}$$

where C is a constant that depends on $\mathbb{X}$, $\mathrm{Lip}(f_1)$ and $\mathrm{Lip}(f_2)$.

When $q = 2$, Theorem 2 implies that

$$d_W^2(\mathcal{D}_1, \mathcal{D}_2) \leq C^{\frac{1}{2}} \cdot ||f_1 - f_2||_\infty^{\frac{1}{2}}. \tag{5}$$

Theorem 2 also implies for $q = \infty$,

$$d_W^\infty(\mathcal{D}_1, \mathcal{D}_2) \leq ||f_1 - f_2||_\infty. \tag{6}$$

Merge trees and stability. Given $f : \mathbb{X} \to \mathbb{R}$, a merge tree records the connectivity of its sublevel sets $f^{-1}(\infty, t]$ as t increases. Two points $x, x' \in \mathbb{X}$ are *equivalent*, $x \sim x'$, if $f(x) = f(x')$, and if they belong to the same connected component of the sublevel set, for some $t \in \mathbb{R}$. A *merge tree* is the quotient space $\mathbb{X}/\sim$ obtained by gluing together points in $\mathbb{X}$ that are equivalent under the relation $\sim$; see Fig. 2(c) for an example. Here, we compute the merge tree of $-f$ which highlights the connectivity among four local maxima and saddles.

For stability, we work with the *q-Gromov-Wasserstein (GW) distance* between merge trees [8, 22]. A merge tree $\mathcal{T}$ is modeled as a measure network represented by a triple (V, μ, W), where V is the set of n nodes, μ is a probability measure on V (typically uniform), and W is a pairwise relation matrix between nodes (i.e., a weighted adjacency matrix). Then the q-GW distance (for $q \geq 1$) between a pair of merge trees $\mathcal{T}_1 = (V_1, \mu_1, W_1)$, $\mathcal{T}_2 = (V_2, \mu_2, W_2)$ is defined as

$$d_{GW}^q(\mathcal{T}_1, \mathcal{T}_2) = \frac{1}{2} \min_{J \in \mathcal{J}} \left(\sum_{i,j,k,l} |W_1(i,k) - W_2(j,l)|^q J_{i,j} J_{k,l} \right)^{\frac{1}{q}}, \tag{7}$$

where $J \in \mathcal{J}$ is a non-negative matrix that encodes the joint probability on $V_1 \times V_2$ whose marginals agree with μ_1 and μ_2. The stability result surrounding the GW distance requires mild assumptions on $f : \mathbb{X} \to \mathbb{R}$ and μ. Specifically, we assume f is obtained from a function $f : V \to \mathbb{R}$ defined on the vertex set of the mesh and extending linearly over higher-dimensional mesh elements. Recall μ is a probability distribution over V. We assume that μ is *balanced*, in the sense that for any $u, v, w \in V$, we have $\mu(u) \cdot \mu(v) \leq \mu(w)$; this property holds for the uniform distribution. We define a family of weighted norms on the space of functions $f : V \to \mathbb{R}$ by

$$\|f\|_{L^q(\mu)} := \left(\sum_{v \in V} |f(v)|^q \mu(v) \right)^{\frac{1}{q}}. \tag{8}$$

Theorem 3 (Stability of Gromov-Wasserstein Distance [22])**.** *Assume $f_1, f_2 : \mathbb{X} \to \mathbb{R}$ satisfies the mild condition above and μ is a balanced probability distribution. Then*

$$d_{GW}^q(\mathcal{T}_1, \mathcal{T}_2) \leq \frac{1}{2} |V|^{\frac{2}{q}} \|f_1 - f_2\|_{L^q(\mu)}. \tag{9}$$

When μ is uniform, the constant improves to be $\frac{1}{2}|V|^{\frac{1}{q}}$.

Following Eq. (8), we see that $\|f_1 - f_2\|_{L^q(\mu)} = (\mathsf{E}_{x \sim \mu} |f_1(x) - f_2(x)|^q)^{1/q}$, where E stands for expectation. For any $q \geq 1$, we have $\|f_1 - f_2\|_{L^q(\mu)} \leq \|f_1 - f_2\|_\infty$. As a result, we can conclude that

$$d_{GW}^q(\mathcal{T}_1, \mathcal{T}_2) \leq \frac{1}{2} |V|^{\frac{2}{q}} \|f_1 - f_2\|_\infty. \tag{10}$$

Moreover, since as we send $q \to \infty$, then the $\frac{2}{q}$ term goes to 0, it implies

$$d_{GW}^\infty(\mathcal{T}_1, \mathcal{T}_2) \leq \frac{1}{2} \|f_1 - f_2\|_\infty. \tag{11}$$

4 Kernel Regression for Scientific Visualization

In this section, we present our main contribution by demonstrating how kernel regression serves as a natural and valuable tool for scientific visualization—one that we believe remains underutilized. As a highlight of its properties, we present topological stability results.

4.1 Kernel Regression Properties

We describe the properties of Nadaraya-Watson (NW) kernel regression KR_P in contrast to analyzing the raw dataset $P = (X, y)$, or its linear interpolation. In particular, KR_P induces several advantageous properties where it enforces smoothness, reduces high-frequency noise, and retains large but shallow features.

We consider some domain $\mathbb{U}$ where X is dense and uniform in $\mathbb{U}$ with respect to the kernel K. Specifically, we consider $\mathbb{U} \subset \mathbb{U}_\rho$ (as defined in Sec. 3.1) for a constant $\rho|P| > 0$ (e.g., $\rho|P| = 1$); this ensures that each $x \in \mathbb{U}$ is within $O(\sigma)$ of some $x_i \in X$ (likely several); where σ is the bandwidth of the kernel K. To ensure that P is uniform, we define $\mathbb{U}_{\bar{\rho}} = \{x \in \mathbb{R}^d \mid \mathrm{KDE}_X(x) \leq 2\rho\}$, which upper bounds the density within $\mathbb{U}_{\bar{\rho}}$. We say P is *dense and uniform* in $\mathbb{U}$ if there exists some ρ where $\mathbb{U} \subset \mathbb{U}_\rho$ (see Sec. 3.1) and $\mathbb{U} \subset \mathbb{U}_{\bar{\rho}}$. For instance, this is true when X is a uniform grid over $\mathbb{U}$ and $\sigma > d$.

Now within $\mathbb{U}$, KR_P is a weighted average of the y_i values, where the weights are smoothly varying over the nearby x_i. Because P is dense and uniform over $\mathbb{U}$ then KDE_X is always in $[\rho, 2\rho]$, so the denominator of KR_P stays within a factor 2. Then since the effect of a point $p_i = (x_i, y_i)$ is $\frac{1}{|P|} y_i K(x_i, x)$ and $K(x_i, x)$ is $O(1/\sigma)$-Lipschitz with respect to x, then overall $\mathrm{KR}_P(x)$ changes smoothly with x in $\mathbb{U}$. As a result, if P contains high-frequency noise (where its reported response values fluctuate rapidly within a small region of the domain), KR_P will dampen or eliminate this noise. Observe that these features may exhibit high persistence, meaning persistence-preserving simplification will not eliminate them, as illustrated in Fig. 3.

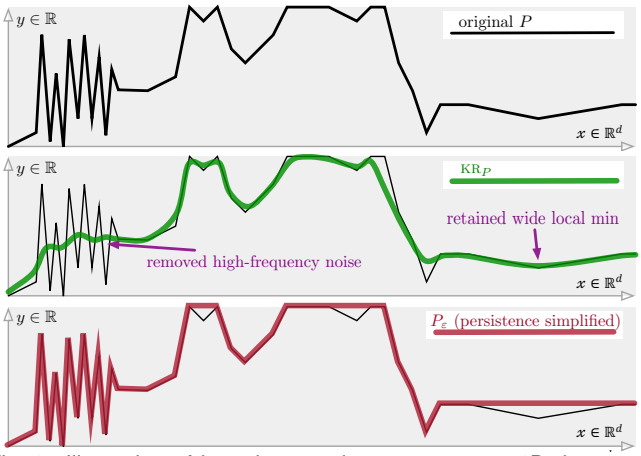

Fig. 3: Illustration of kernel regression KR_P on a toy 1D dataset P, compared to persistence simplification P_ε. Note that KR_P removes high-frequency noise while preserving wide, but low persistence, features.

At the same time, small-valued features that are spread out will persist. If a large region contains a well-defined but shallow local minimum—where the response values $y_i \in P$ are relatively similar—then

KR_P will retain such features. However, because the y_i values are close to each other, these features may have low persistence, making them removable through persistence-preserving simplification. In contrast, features that are large both in area and response value will be preserved by both persistence simplification and kernel regression; see Fig. 3.

The choice between applying kernel regression to P, persistence simplification of P, or both should be made on a case-by-case basis, depending on the specific domain and data. However, we believe that in many scenarios, kernel regression smoothing is not only reasonable but also highly desirable.

4.2 Stability Results for Coresets of NW Kernel Regression

Another benefit of modeling data P as KR_P is that we can apply coreset constructions to get S that can significantly reduce the size so $|S| \ll |P|$. Recall that we can create S as an (ε, ρ)-coreset of P using grid methods with $\gamma = \frac{\varepsilon\sigma\rho}{8\sqrt{d}}$ or as a sample of size $O\left(\frac{1}{\varepsilon^2\rho^2}(d\log(\frac{1}{\rho}))\right)$. Not only is the function preserved in a strong L_∞ sense, but we observe that these are compatible with key topological stability results. As a result, invoking stability theorems from Sec. 3.2 implies that much of the topological structure of KR_P—in particular, persistence diagram and merge tree—is preserved in KR_S. We summarize in our main theoretical contribution below.

Theorem 4. *Consider a dataset $P = (X, y)$ with response normalized so that $(\max_i y_i) - (\min_i y_i) = 1$, and $X \in \mathbb{U}_\rho$. Let S be an (ε, ρ)-coreset of P. Then we have the following topological stability results when comparing KR_P to KR_S through their respective persistence diagrams $\mathcal{D}(\text{KR}_\text{P})$ and $\mathcal{D}(\text{KR}_\text{S})$, and merge trees $\mathcal{T}(\text{KR}_\text{P})$ and $\mathcal{T}(\text{KR}_\text{S})$:*

Persistence diagram bottleneck stability:

$$d_\infty(\mathcal{D}(\text{KR}_P), \mathcal{D}(\text{KR}_S)) \leq \varepsilon. \tag{12}$$

Persistence diagram q-Wasserstein stability:

$$d_W^q(\mathcal{D}(\text{KR}_P), \mathcal{D}(\text{KR}_S)) \leq C^{\frac{2}{q}}\varepsilon^{1-\frac{1}{q}} \tag{13}$$

for $q \geq 1$, and

$$d_W^\infty(\mathcal{D}(\text{KR}_P), \mathcal{D}(\text{KR}_S)) \leq \varepsilon. \tag{14}$$

Merge tree q-Gromov-Wasserstein stability: for $q = 1, 2$,

$$d_{GW}^q(\mathcal{T}(\text{KR}_P), \mathcal{T}(\text{KR}_S)) \leq \frac{1}{2}|V|^{\frac{2}{q}}\varepsilon. \tag{15}$$

And

$$d_{GW}^\infty(\mathcal{T}(\text{KR}_P), \mathcal{T}(\text{KR}_S)) \leq \frac{\varepsilon}{2}. \tag{16}$$

Proof. A (ρ, ε)-coreset S guarantees that $\|\text{KR}_\text{P} - \text{KR}_\text{S}\|_\infty \leq \varepsilon$ (with constant probability for the random sample) on the relevant domain $\mathbb{U}_\rho$. Then the results follow directly from the stability bounds for those topological characteristics based on the L_∞ distance between functions; that is $\|\text{KR}_\text{P} - \text{KR}_\text{S}\|_\infty \leq \varepsilon$. The individual bounds follow from Eq. (3), Eq. (4), Eq. (6), Eq. (10), and Eq. (11). The constant C is an absolute constant that depends on technical properties of $\mathbb{U}_\rho$ and Lipschitz factors which are bounded by $1/\sigma$ when P is dense and uniform as with a dense grid input. □

5 Optimized Coresets for NW Kernel Regression

We propose a novel, optimized coreset construction that utilizes gradient descent to refine the coreset points. We start with an overview of our method, then provide a detailed explanation of the key components in the initial coreset construction and optimization process. Lastly, we outline parameter configurations and evaluation.

Overview. An overview of our framework is shown in Fig. 4. Given a scalar field as input, we aim to construct a coreset for kernel regression that acts as a proxy for the original data. We first construct the initial coreset from the scalar field. We then optimize the initial coreset and compute the kernel regression of our optimized coreset over a finely covered number of evaluation points.

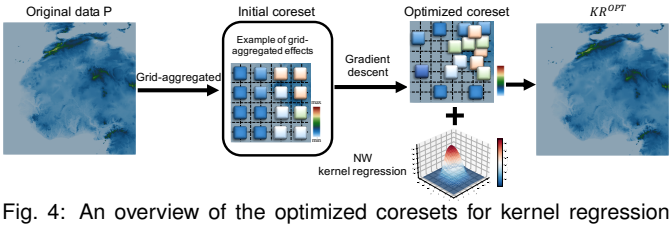

Fig. 4: An overview of the optimized coresets for kernel regression framework, illustrating the grid-aggregated effects and optimized coreset. Each color box represents a coreset point generated from each grid cell.

5.1 Optimized Coreset Construction

Initial coreset construction. We construct the initial coreset using state-of-the-art grid-aggregated (GA) method, which generally outperforms other strategies and aligns naturally with the structured nature of scientific datasets. An illustrative example of coreset construction using GA is presented in Fig. 4. Each coreset point is generated by applying GA to the corresponding grid cell. The grid structure is indicated by black lines, while colors of boxes represent different function values for coreset points. Given the original data $P = (X, y)$, where $X \subset \mathbb{R}^d$ is described by a uniform rectangular grid on $\mathbb{U}$. Let $\boldsymbol{m} = (m_1, m_2, \ldots, m_d)$, where m_j encodes the number of grid points along the jth dimension. We then define a sparser grid G_γ with cubical cells of size γ^d. For each cell $g \in G_\gamma$, we create a coreset point by averaging the point locations and function values to place into the initial coreset S. Next, we obtain an initial coreset with its size depending only on $\gamma > 0$, where $|S| = \left\lfloor \frac{\Pi_{j=1}^d m_j}{\gamma^d} \right\rfloor$.

However, the grid-aggregated coreset is not adaptive—they treat all regions uniformly, regardless of the complexity or flatness of the underlying scalar field. Due to its aggregating nature, GA often struggles to precisely capture critical regions with steep gradients or densely concentrated function values, resulting in under-representation. Conversely, it assigns the same number of coreset points to flat regions—such as ocean areas with all-zero function values—as to regions with significant variation, leading to over-representation. To address these limitations, we further refine the initial coreset points to better capture high-gradient regions while reducing point density in flat areas, enhancing representation efficiency and accuracy.

Gradient descent for coreset optimization. We use gradient descent [21] to iteratively optimize our coreset points by moving in the direction of steepest descent (i.e., the negative gradient of the function). An illustrative example of the optimized coreset is shown in Fig. 4. Given an initial coreset point $s_i = ((x_{i,1}, x_{i,2}, \ldots x_{i,d}), y_i)$, where $x_{i,j} = \frac{1}{|P_g|}\sum_{p_{i'} \in P_g} x_{i',j}$ and $y_i = \frac{1}{|P_g|}\sum_{p_{i'} \in P_g} y_{i'}$, we optimize both the spatial locations and function values of each coreset point, by updating:

$$s_i \leftarrow s_i - \eta \cdot \nabla\mathcal{L}$$

where η is the learning rate and $\nabla\mathcal{L}$ represents the gradient of the loss function. We define our loss function as the L_2-norm between KR_S and KR_P,

$$\mathcal{L} = \|\text{KR}_P(Q) - \text{KR}_S(Q)\|_2^2,$$

where $Q = \{q_k\}_{k=1}^N \subset \mathbb{X}$ is a set of evaluation points generated from the domain $\mathbb{U}$.

Gradient descent enables fine-tuning of both the spatial locations and function values of coreset points, allowing them to better capture the structure of the underlying scalar field. Instead of fixed positions (e.g., grid aggregation), this approach moves points to locations that minimize the overall kernel regression error, aligning them with important features such as peaks, ridges, and sharp transitions within the kernel smoothing. With a fixed coreset size, this optimization process enables adaptive reallocation of representation thus placing more emphasis on complex regions and less on flat or redundant areas. As the optimization iteratively adjusts coreset points to minimize error, the coreset gradually converges to an optimized configuration.

5.2 Parameter Configurations and Evaluation

There are a few parameters native to the optimization process, some of which are modeling choices (e.g., the bandwidth σ, the coreset size $|S|$) with inherent trade-offs. Others are associated with the optimization process (e.g., learning rate η and the number of iterations). We include default parameter settings here and different parameter choices on learning rate and the number of iterations are discussed in the supplement. We set $\eta = 1$ for 2D and $\eta = 0.1$ for 3D datasets, and use 30 for the number of iterations; however the optimization works well for a variety of choices. We use the Adam optimizer [2] for gradient descent.

To determine the coreset size, we target a data reduction to 0.25% in 2D by setting $\gamma = 20$. For more complex 3D datasets, we aim for a reduction to approximately 1.5% using $\gamma = 4$. These default values achieve substantial size reduction while introducing negligible error. In our experimental results, we explicitly analyze the trade-off between size reduction and accuracy; see Sec. 6.1.

Finally, evaluating our optimized coreset by measuring functional changes at every $x \in X$ would be computationally expensive and largely unnecessary. Our results show that in 2D, using only $1/64$ of the original data yields an evaluation set Q with sufficient precision. In 3D, where the data exhibits greater variability, we use $1/8$ of the points for Q; further details are provided in the supplement.

6 Experimental Results

We qualitatively and quantitatively evaluate our optimized coreset for kernel regression in scientific visualization:

- We show that our optimized coreset method effectively preserves topological descriptors—persistence diagrams and merge trees—of kernel regression on the original dataset while also reconstructing isosurfaces in time-varying datasets.
- We demonstrate that our optimized coreset method significantly outperforms existing coreset construction techniques in accuracy. On average, it reduces error by one to two orders of magnitude across all datasets, while maintaining coreset sizes of just 0.25% of the original data in 2D and 1.56% in 3D.
- We compare our optimized coreset for kernel regression against other data reduction techniques. In contrast to other methods, it results in a continuous output, and we quantify this property by observing an order of magnitude improvement on error for the function gradient.

Datasets. We conduct experiments on three 2D datasets and three 3D datasets defined on a structured mesh, as presented in Tab. 1: CESM [32] and two regions from GTOPO30 [9] in 2D; Viscous Fingers [1], Vortex [43, 44], and Vortex Street [50] (available at [56]) in 3D. For some but not all datasets (detailed below), we preprocess with persistence simplification [11] to separate features from noise.

Dataset	Grid Size	Coreset Size	Coreset %
CESM	1800×3600	$16,200$	
GTOPO-NA	4800×4800	$57,600$	0.25%
GTOPO-ME	2000×3000	$15,000$	
Viscous Fingers	$101 \times 101 \times 101$	$15,625$	
Vortex Street	$192 \times 64 \times 48$	$9,216$	1.56%
Vortex	$128 \times 128 \times 128$	$32,768$	

Table 1: Scientific datasets and coresets used in our experiments. GTOPO-NA stands for the North Africa region and GTOPO-ME stands for the Middle East region of the GTOPO30 dataset.

CESM represents the Community Earth System Model that provides comprehensive global climate data. We take the FLDSC variable, which represents the clear-sky downwelling long-wave flux on the surface [32]. **GTOPO30** is an open-source 2D global elevation dataset with 30 arc-second horizontal grid spacing, resulting in a dimension of 21601×43201. To better investigate the topology preservation locally, we extract two regions from the original GTOPO30 dataset: one from North Africa and another from the Middle East, specifically centered around the Gulf of Suez. We use a persistence simplification threshold of 0.03. **Viscous Fingers** dataset is an ensemble composed of 3D transient fluid flow obtained by a simulation with stochastic effects, formulating a special behavior named as viscous fingers. We select a salt resolution level 0.44 and a persistence simplification threshold of 0.01 for preprocessing. **Vortex** dataset is a simulation of vortex structures with 30 time steps. We consider the magnitude of vorticity as the scalar field. The Bénard von Kárman **Vortex Street** dataset captures the high vorticity regions sampled on a regular grid across 102 time steps. We consider the magnitude of velocity as the scalar field.

An overview of experiments. We demonstrate that our optimized coreset effectively approximates kernel regression of the original dataset. We experiment with kernel bandwidth $\sigma \in \{5, 10, 15, 20, 25, 30\}$ for 2D and $\sigma \in \{1, 2, 3, 4, 5, 6\}$ for 3D datasets. For a fixed σ, we perform multiple optimization iterations until the loss converges. We visualize each dataset using a small σ, ensuring effective smoothing while retaining the essential structures of the original dataset. For coreset construction methods that involve randomness, we apply each method five times and report the average error.

We use ParaView and the Topology ToolKit (TTK) [3, 15, 52] to visualize scalar fields, generate critical points, persistence diagrams, and merge trees. Our optimized coreset for kernel regression framework is implemented using PyTorch and performed on a single NVIDIA RTX 5090 GPU. See the supplement for detailed runtime analysis.

To quantitatively evaluate the accuracy of our coreset construction, we measure the L_∞ error between KR_P and each coreset variant denoted as KR_S (KR^{OPT}, KR^{GA}, KR^{GR}, or KR^{RS}). We uniformly sample a set Q of evaluation points over the original data domain to ensure coverage. Both KR_P and KR_S are evaluated on the same set of evaluation points. See the supplement for a comprehensive visual comparisons across σ.

Comparison with coreset baselines. We compare the kernel regression model KR^{OPT} with three coreset construction baselines: the random sample-based KR^{RS}, and two grid-based methods, KR^{GA} and KR^{GR}, using KR_P as the ground truth. The random sample method (RS) is the most commonly used approach for aggregating data and can be easily extended to other computational models. Additionally, we compare kernel regressions based on two grid-based methods [60]: grid-random (GR) and grid-aggregate (GA).

Comparison with data compressors. Constructing KR^{OPT} is a form of data reduction for KR_P. We compare KR^{OPT} to other data reduction methods, in particular, lossy data compressors such as SZ3 [24] and ZFP [**?**]. SZ3 and ZFP allow the user to bound the L_∞ error by a user-specified parameter ξ, thereby achieving similar topological guarantees to KR^{OPT}. We also compare against a topology-preserving data compressor [**?**] referred to as TopoQZ in this paper. TopoQZ augments ZFP by preserving topological features whose persistence exceeds a user-specified threshold ε while discarding all others.

We losslessly compress KR^{OPT} for each dataset. To evaluate each compressor, we resample KR_P for each dataset to the original grid, then compress it. We aim to choose parameter settings for each compressor that produce compressed file sizes similar to the compressed KR^{OPT}; see the supplement for details on parameter settings and file sizes.

For each reduction/compression method, we measure the L_∞ error of a reconstructed dataset and the average L_2 error of the gradient relative to the analytic gradient of KR_P. For KR^{OPT}, we use the analytic gradient, while for the grid-based compressors we use the discrete gradient computed with NumPy. All metrics are evaluated at designated evaluation points, with each dataset normalized so that the function values at these points lie within $[0, 1]$.

6.1 Parameter Analysis

We evaluate the effectiveness of coreset methods by varying the evaluation set Q from $3,000$ to $130,000$, using the CESM dataset with a fixed kernel bandwidth $\sigma = 10$ and a coreset size $16,200$.

We measure the L_∞ error between KR_P and each KR_S variant. As shown in Fig. 5(left), increasing the number of evaluation points initially leads to higher error, but the error stabilizes beyond a certain threshold. This trend supports our earlier observation that evaluating on a subset of the original data domain achieves accuracy similar to that of using the full dataset. In Fig. 5(right), we observe that KR^{OPT} consistently achieves the lowest error—an order of magnitude smaller than the other methods—while KR^{RS} performs the worst, with the grid-based methods falling in between.

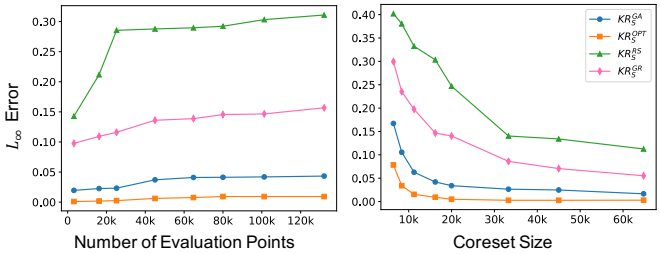

Fig. 5: Parameter analysis on the number of evaluation points and the coreset size for the CESM dataset. Left: L_∞ error vs. number of evaluation points with different coreset methods. Right: L_∞ error vs. coreset size with different coreset methods.

Fig. 5(right) also shows how coreset error decreases with increasing coreset size, using a fixed $\sigma = 10$ and $101,250$ evaluation points. All methods exhibit a similar decrease in L_∞ error as coreset size increases. KR^{OPT} achieves nearly an order of magnitude lower error and converges faster, attaining small error even with a small coreset size of $10{,}000$. To balance data reduction and accuracy, we identify a 'sweet spot' region where the coreset size is minimized without introducing errors exceeding an order of magnitude. For instance, when the coreset size is between $15{,}000$ and $20{,}000$, the errors become negligible as the size increases further, indicating convergence. Based on this observation, we employ a coreset size of $16{,}200$ (only 0.25% of the original data size) for the CESM dataset in practice.

6.2 L_∞ Error Evaluation

We present the L_∞ error evaluation for all datasets. We normalize y-values of all datasets to $[0, 1]$ for fair comparisons. Tab. 2 shows the L_∞ error between KR_P and KR_S for different coreset methods across multiple bandwidth σ. OPT outperforms the baselines across all σ values for all datasets, as highlighted in bold. Even with the smallest σ, OPT achieves a $2\times$ - $4\times$ error reduction compared to GR and RS, while maintaining lower error than GA. As σ increases, the performance gap gets larger: starting from $\sigma = 15$ for 2D and $\sigma = 3$ for 3D, OPT achieves at least an order of magnitude lower error than GR, RS, and GA on most datasets, with the exception of Vortex and Vortex Street. In general, coreset performance is slightly lower on 3D datasets due to their increased structural complexity. For the last three σ values tested, OPT outperforms GA by at least an order of magnitude.

Among the three baselines, grid-based methods generally outperform random sampling, with GA yielding better results than GR, suggesting that grid-based aggregation is more effective than random selection.

In general, the error decreases as we use a larger kernel bandwidth σ. σ controls the degree of smoothing: larger values yield smoother results but tend to blur the boundaries of salient features, whereas smaller values preserve finer details closer to the original data. This parameter choice is inherently a user-driven modeling choice. Although larger σ values can allow for smaller coresets, they risk over-smoothing regions, thereby oversimplifying topological features. In practice, small σ values are able to effectively suppress high-frequency noise while preserving broad local minima.

For the CESM dataset, we determine the evaluation points as $1/64$ of the original data, corresponding to $1/8$ sampling in each dimension, uniformly drawn from the original domain, resulting in an evaluation grid of size 225×450; see Tab. 1. Fig. 6 shows the scalar field visualization of different coreset methods for kernel regression and absolute pointwise differences w.r.t. KR_P.

We observe that KR^{OPT} significantly reduces pointwise errors on the evaluated grids compared to KR^{RS}, KR^{GR}, and KR^{GA}, as highlighted in orange blocks in Fig. 6. While KR^{GA} produces a visually similar result, it still exhibits block-wise artifacts due to grid aggregation, which become more apparent in the zoomed-in views. KR^{GR} and KR^{RS} shows more pronounced artifacts and blurred structures, particularly in regions with high function value changes. In contrast, KR^{OPT} effectively reduces the majority of errors in high-gradient regions. With a small σ, our optimized coreset provides a highly precise approximation while preserving essential data structures.

Dataset	Method	L_∞ Error					
bandwidth		5	10	15	20	25	30
CESM	RS	0.41688	0.30329	0.15394	0.16246	0.12543	0.07765
	GR	0.25217	0.14662	0.08334	0.06413	0.04781	0.02765
	GA	0.16194	0.04196	0.01470	0.00907	0.00593	0.00398
	OPT	**0.10645**	**0.00928**	**0.00659**	**0.00282**	**0.00112**	**0.00072**
GTOPO -NA	RS	0.49061	0.46461	0.22601	0.26169	0.17664	0.13146
	GR	0.48915	0.33288	0.14642	0.11575	0.12147	0.07018
	GA	0.29955	0.16792	0.11828	0.12100	0.11628	0.10912
	OPT	**0.21934**	**0.06524**	**0.00454**	**0.00398**	**0.00261**	**0.00187**
GTOPO -ME	RS	0.46226	0.33960	0.21871	0.16493	0.13234	0.14428
	GR	0.37220	0.24231	0.13922	0.11654	0.08555	0.08791
	GA	0.19111	0.07368	0.02863	0.01699	0.01157	0.00815
	OPT	**0.11751**	**0.01715**	**0.00450**	**0.00266**	**0.00224**	**0.00218**
bandwidth		1	2	3	4	5	6
Viscous Finger	RS	0.97275	0.61292	0.43585	0.31564	0.23863	0.16911
	GR	0.85620	0.34062	0.29723	0.19732	0.13921	0.12895
	GA	0.42686	0.15202	0.11376	0.10470	0.09518	0.09003
	OPT	**0.16203**	**0.00484**	**0.00445**	**0.00391**	**0.00350**	**0.00288**
Vortex Street	RS	0.81510	0.66091	0.42552	0.30870	0.21153	0.19753
	GR	0.80403	0.57642	0.40813	0.27581	0.21376	0.14560
	GA	0.54611	0.18838	0.06779	0.03919	0.02765	0.02086
	OPT	**0.46930**	**0.12380**	**0.01251**	**0.00764**	**0.00482**	**0.00303**
Vortex	RS	0.50764	0.34072	0.24223	0.18643	0.15651	0.13298
	GR	0.41637	0.25198	0.14546	0.10657	0.08447	0.06073
	GA	0.24685	0.07226	0.03199	0.02309	0.01777	0.01389
	OPT	**0.13835**	**0.01320**	**0.00373**	**0.00279**	**0.00277**	**0.00267**

Table 2: L_∞ error evaluation between KR_P and KR_S for different coreset methods for various bandwidths. OPT coreset method outperforms the baselines across all bandwidths for all datasets, as highlighted in bold.

6.3 Topology Preservation and Stability

In this section, we demonstrate the effectiveness of our optimized coreset in preserving topological descriptors.

6.3.1 Preserving Critical Points

We use the GTOPO30 dataset to demonstrate critical point preservation. Fig. 7 presents the kernel regression visualization for North Africa, with critical points extracted from the scalar field overlaid on each result.

In the zoomed-in view of Fig. 7 middle row, we observe that KR^{OPT} closely aligns with KR_P, accurately preserving important (high persistence) critical points, including local maxima, local minima, and saddles, demonstrating KR^{OPT} effectively captures essential topological features of the original dataset. In contrast, KR^{GA} and KR^{RS} introduce numerous spurious (false-positive) critical points, particularly in regions with high gradients; see orange blocks.

The bottom row of Fig. 7 shows the kernel regression for the Middle East region, where KR^{GA} and KR^{RS} fail to preserve many critical point pairs, particularly along the coastline. In contrast, KR^{OPT} preserves the majority of critical points, with only minor positional shifts near the domain boundaries. Overall, OPT method proves highly effective in maintaining critical points, especially in high-gradient regions.

6.3.2 Preserving Merge Trees

We use the Viscous Fingers dataset to demonstrate the effectiveness of our optimized coreset for kernel regression in preserving the merge tree. Fig. 8 shows the visualization of critical points embedded in scalar fields and the merge tree comparisons of corresponding kernel regression models. The merge tree of KR^{OPT} closely aligns with that of KR_P, preserving the overall structure. In contrast, the merge tree of KR^{GA} exhibits a significant shift in the root node and spurious maximum-saddle pairs. KR^{RS} exhibits even more noticeable misalignment in critical point pairs within the merge trees, particularly a significant positional shift of the root node and mismatch of several maximum–saddle pairs.

6.3.3 Stability Results

We provide empirical results for evaluating the topological stability described in Sec. 4.2. As shown in Fig. 10, we compute the bottleneck distance and Wasserstein distance between persistence diagrams of KR_P

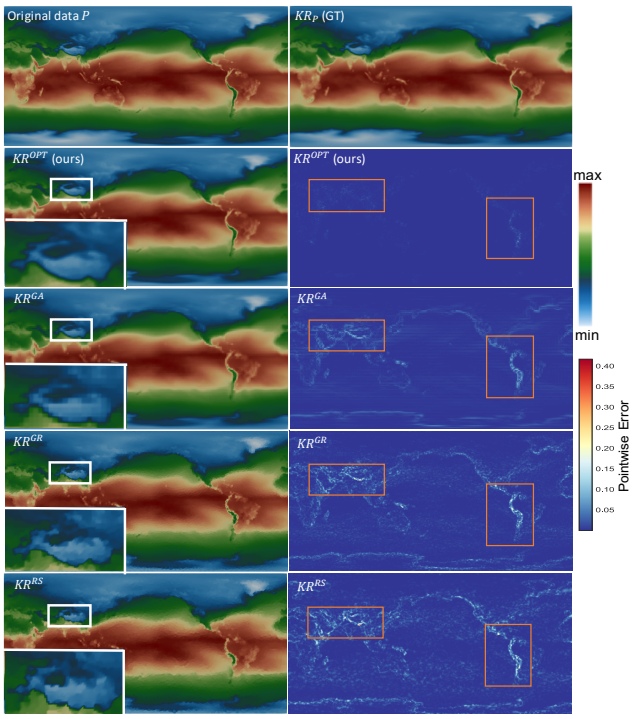

Fig. 6: CESM dataset: qualitative evaluation of kernel regression models with different coreset methods. Top row: original data and kernel regression model based on the original data (KR_P) considered as the ground truth (GT). Second row to last row: kernel regression models based on our optimized coreset (KR^{OPT}), grid-aggregated coreset (KR^{GA}), grid-random coreset (KR^{GR}), and randomly sampled coreset KR^{RS} on the left; pointwise error of kernel regression models against the ground truth on the right. The kernel bandwidth is 5.

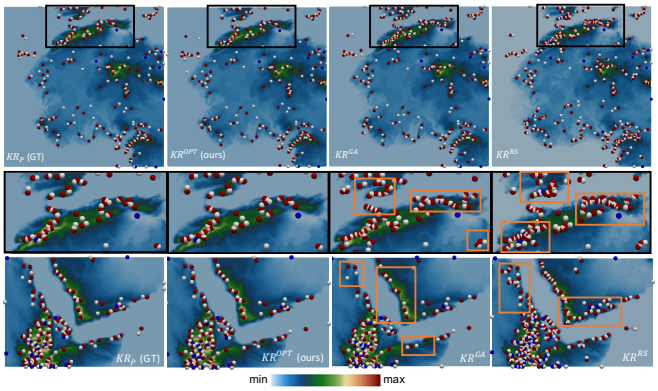

Fig. 7: GTOPO30 dataset. Top row: visualization of the critical points overlaid on the North Africa region. Middle row: zoomed-in views of the critical points in the black boxes. Bottom row: visualization of the critical points overlaid on the Middle East region. From left to right: kernel regression models based on the original data (KR_P), our optimized coreset (KR^{OPT}), grid-aggregated coreset (KR^{GA}), and randomly sampled coreset KR^{RS}. Orange blocks highlight examples of critical point shifts. Local maxima are in red, local minima are in blue, and saddles are in white. The kernel bandwidth is 10.

and each KR_S variant. We use GW distance to quantify the difference between their corresponding merge trees.

We observe that KR^{OPT} consistently yields smaller distances, whereas KR^{RS} and KR^{GR} generally exhibit larger distances, regardless of the choice of distance measure. Additionally, as the kernel bandwidth increases, the distances between topological descriptors tend to decrease. The stability results are consistent with our earlier observations, demonstrating that our optimized coreset method consistently outperforms baseline approaches, and providing experimental validation of the theoretical guarantees offered by our framework.

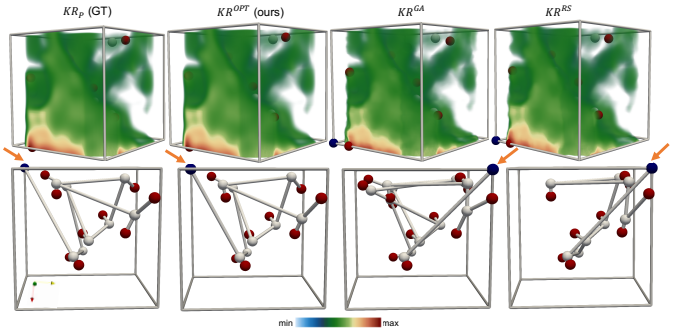

Fig. 8: Viscous Fingers dataset at time step 120. Top row: kernel regression models on the original data (KR_P), our optimized coreset (KR^{OPT}), grid-aggregated coreset (KR^{GA}), and randomly sampled coreset KR^{RS}. Bottom row: merge trees of the corresponding kernel regression models. Local maxima are red, local minima are blue, and saddles are white. Orange arrows indicate the structural shift of minima in merge trees. The kernel bandwidth is 2.

6.4 Isosurface Preservation

We validate the effectiveness of our optimized coreset for kernel regression in maintaining high-quality volume and isosurface renderings, revealing key features (such as vortices) for time-varying datasets.

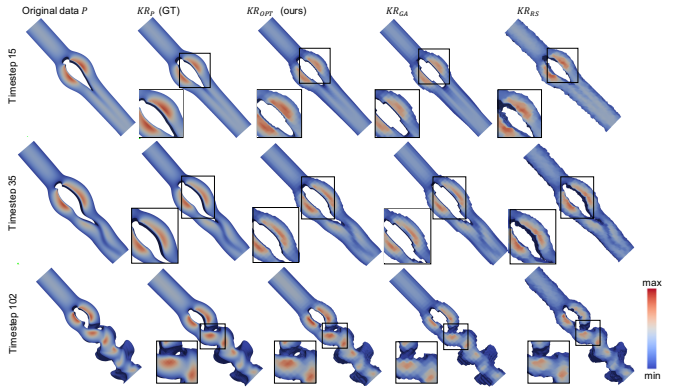

Fig. 9: Vortex street dataset: a comparison of KR_P and each KR_S variant on isosurface renderings at isovalue 0.8 extracted from timesteps 15, 35, and 102, respectively. The kernel bandwidth is 1.

We demonstrate the effectiveness of our framework using the time-varying Vortex dataset first. We determine the evaluation points as $1/8$ of the original data, resulting in an evaluation grid of size $64 \times 64 \times 64$. Fig. 1 shows the volume and isosurface visualization at timestep 2 with $\sigma = 1$. We present two isosurface renderings to highlight differences in isosurface fidelity across different coreset methods.

Both the volume and isosurface renderings produced by KR^{RS} show significant blurring and noise artifacts, which obscure crucial structural details. While KR^{GA} improves over random sampling by recovering a clearer vortex structure, it suffers from prominent zigzag artifacts in high-gradient regions and noticeable seam-line discontinuities due to the underlying grid partitioning. In contrast, KR^{OPT} preserves high-quality vortex structures in both volume and isosurface renderings. Using a small kernel bandwidth of $\sigma = 1$, the reconstruction achieves a balance between smoothness and feature complexity. Importantly, KR^{OPT} retains detailed isosurface geometry with minimal distortion, despite using only 1.56% of the original data size. This experiment highlights the utility of OPT method in visualization tasks such as feature preservation and detection, even under aggressive data reduction.

For the Vortex Street dataset, Fig. 9 presents isosurface renderings at isovalue 0.8 across kernel regression models, with zoomed-in views for each coreset method. We have evaluation points as $1/8$ of the original data, resulting in an evaluation grid of size $96 \times 32 \times 24$. While all KR_S variants capture the overall vortex street structure, KR^{OPT} stands out by producing smoother results with higher visual fidelity. In contrast, KR^{GA} displays blurring and noise artifacts, especially near the boundaries, where KR^{RS} exhibits even more pronounced artifacts. Although our optimized coreset is about 1.5% of the original dataset, KR^{OPT} preserves

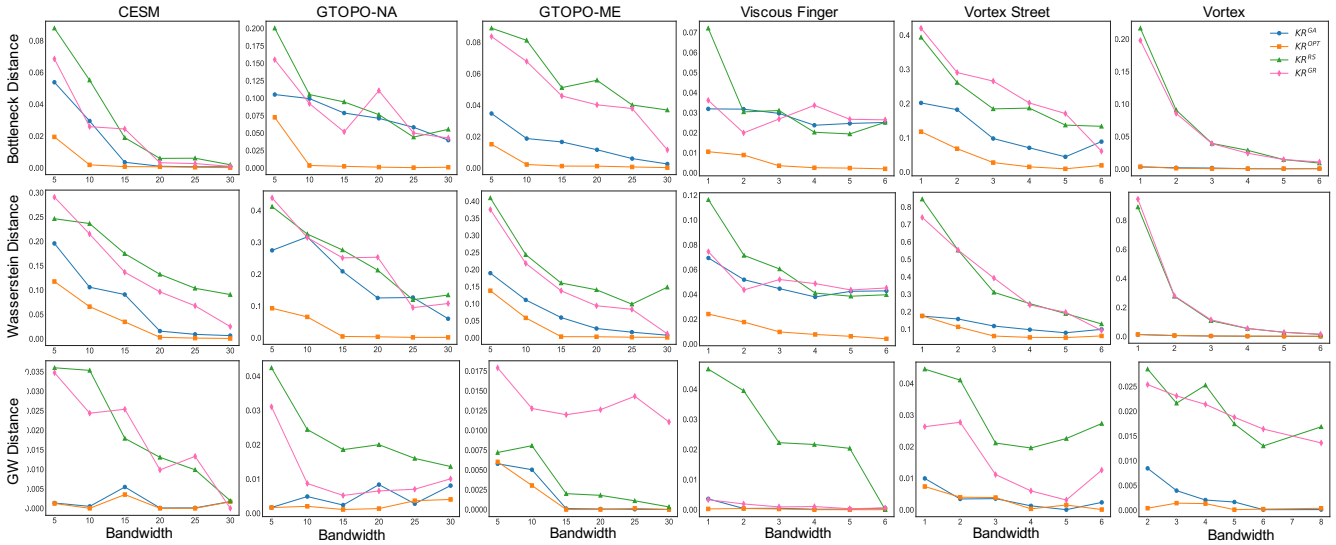

Fig. 10: Topological stability results showing distance measures between topological descriptors of $\mathrm{KR_P}$ and each $\mathrm{KR_S}$ variant: bottleneck and Wasserstein distances for persistence diagrams, and Gromov-Wasserstein (GW) distance for merge trees. Lower values are better.

both vortex features and symmetry, demonstrating the effectiveness of our framework in capturing key structure.

To further quantitatively evaluate the effectiveness of our optimized coreset in preserving key features, we measure the isosurface similarity by evaluating the Dice similarity coefficient (a.k.a., Dice score) between $\mathrm{KR_P}$ and each $\mathrm{KR_S}$ variant. Dice score is a widely used metric for quantifying the similarity between two sets. As shown in Fig. 11, we evaluate on four isovalues $\{0.2, 0.4, 0.6, 0.8\}$ as our kernel regression results are y-normalized to $[0, 1]$. $\mathrm{KR^{OPT}}$ consistently outperforms the other coreset methods across nearly all σ. For 2D datasets, its Dice score reaches above 0.9, with similarly high performance on 3D datasets.

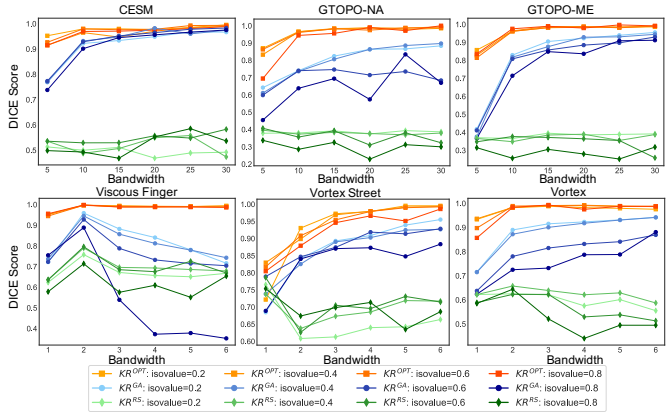

Fig. 11: Isosurface evaluation: Dice scores of different isosurfaces between $\mathrm{KR_P}$ and each $\mathrm{KR_S}$ (higher values are better). We use four isovalues: $\{0.2, 0.4, 0.6, 0.8\}$. The Dice scores coming from the same coreset method are grouped into similar colors.

6.5 Comparison with Lossy Compressors

Tab. 3 reports results from comparisons with existing lossy data compressors. For each dataset and reduction/compression method, we present the L_∞ error at a file size comparable to that of $\mathrm{KR^{OPT}}$, along with the average L_2 gradient error.

As shown, SZ3 and ZFP frequently achieve smaller L_∞ errors than $\mathrm{KR^{OPT}}$, which is expected since these compressors explicitly optimize for L_∞ error without additional guarantees. Nonetheless, $\mathrm{KR^{OPT}}$ outperforms ZFP on the 2D datasets and generally attains lower L_∞ error than the topology-preserving compressor TopoQZ.

Moreover, $\mathrm{KR^{OPT}}$ exhibits superior preservation of gradients compared to SZ3, ZFP, and TopoQZ. By retaining a smooth structural form $\mathrm{KR^{OPT}}$ can use an analytic gradient, whereas the grid-based lossy compressors default to discrete approximations, which inevitably introduce error even under very tight compression error bounds.

Dataset	SZ3	ZFP	TopoQZ	$\mathrm{KR^{OPT}}$
Overall L_∞ error				
CESM	$\mathbf{3.7 \times 10^{-5}}$	1.3×10^{0}	1.5×10^{-1}	6.6×10^{-3}
GTOPO-NA	$\mathbf{1.1 \times 10^{-5}}$	2.7×10^{-1}	1.7×10^{-1}	4.5×10^{-3}
GTOPO-ME	$\mathbf{1.4 \times 10^{-5}}$	3.8×10^{-1}	1.5×10^{-1}	4.5×10^{-3}
Viscous Finger	$\mathbf{2.3 \times 10^{-5}}$	1.4×10^{-4}	7.3×10^{-3}	4.5×10^{-3}
Vortex Street	$\mathbf{4.9 \times 10^{-5}}$	2.7×10^{-4}	2.7×10^{-3}	3.7×10^{-3}
Vortex	$\mathbf{5.3 \times 10^{-5}}$	5.5×10^{-4}	8.9×10^{-2}	1.3×10^{-2}
Average L_2 error of the gradient				
CESM	1.7×10^{-3}	1.2×10^{-3}	1.9×10^{-3}	$\mathbf{2.4 \times 10^{-4}}$
GTOPO-NA	7.4×10^{-4}	1.4×10^{-3}	8.7×10^{-4}	$\mathbf{7.3 \times 10^{-5}}$
GTOPO-ME	1.6×10^{-3}	2.5×10^{-3}	1.9×10^{-3}	$\mathbf{1.3 \times 10^{-4}}$
Viscous Finger	1.7×10^{-2}	1.7×10^{-2}	1.7×10^{-2}	$\mathbf{2.7 \times 10^{-4}}$
Vortex Street	1.9×10^{-2}	1.9×10^{-2}	1.9×10^{-2}	$\mathbf{4.5 \times 10^{-3}}$
Vortex	1.9×10^{-2}	1.9×10^{-2}	1.9×10^{-2}	$\mathbf{3.0 \times 10^{-4}}$

Table 3: Top: L_∞ error of $\mathrm{KR^{OPT}}$ and lossy compressors when reducing $\mathrm{KR_P}$ to a file size comparable to that of $\mathrm{KR^{OPT}}$. Bottom: average L_2 gradient error relative to the analytic gradient of $\mathrm{KR_P}$. All datasets are normalized so that function values at evaluation points lie within $[0, 1]$. Bandwidths are set to $\sigma = 15$ for 2D datasets and $\sigma = 3$ for 3D datasets.

7 Discussion and Limitations

We introduce kernel regression as a surrogate modeling framework that utilizes coresets for scientific visualization. In particular, we advocate for controlled kernel smoothing via Nadaraya-Watson (NW) kernel regression. It is a formal method for removing or reducing high-frequency noise while preserving wide local extrema, and we believe it should be an essential tool in any visualization task where noise is a concern. Importantly, it is amenable to very small coreset construction: datasets can be reduced by orders of magnitude with minimal loss of accuracy. In particular, we prove that our coresets preserve the stability of topological descriptors such as persistence diagrams and merge trees. We further introduce a new, optimized coreset construction that significantly improves upon prior state-of-the-art methods using a simple gradient descent algorithm, achieving error reductions of up to two orders of magnitude.

A limitation of this framework is its sensitivity to parameter choices, particularly the kernel bandwidth σ and coreset size. An overly large σ oversmooths the scalar field and removes salient features, whereas smaller bandwidths empirically strike a better balance by preserving topological structure while suppressing high-frequency noise. In comparison with SZ3 and ZFP, the optimized coreset achieves consistently lower average L_2 gradient error, albeit at the cost of higher L_∞ error at comparable file sizes. By contrast, when evaluated against a compressor designed for topology-preservation, TopoQZ, it demonstrates superior performance in both L_∞ error and average L_2 gradient error. Runtime optimization, specifically the trade-off among loss size, iteration count, and error, remains unaddressed and is left for future work.

Acknowledgments

WL, NG, and BW were partially supported by U.S. Department of Energy (DOE) grants DE-SC0021015 and DE-SC0023157, and by the U.S. National Science Foundation (NSF) under grant OAC-2313124. JP was supported by NSF grants CCF-2115677, III-2311954, and AST-2421782, as well as by Simons Foundation award MPS-AI-00010515.

References

[1] The IEEE SciVis Contest. https://cloud.sdsc.edu/v1/AUTH_sciviscontest/2016/README.html, 2016. 6

[2] J. Ba and D. P. Kingma. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 6

[3] T. Bin Masood, J. Budin, M. Falk, G. Favelier, C. Garth, C. Gueunet, P. Guillou, L. Hofmann, P. Hristov, A. Kamakshidasan, C. Kappe, P. Klacansky, P. Laurin, J. A. Levine, J. Lukasczyk, D. Sakurai, M. Soler, P. Steneteg, J. Tierny, W. Usher, J. Vidal, and M. Wozniak. An overview of the topology toolkit. *Topological Methods in Data Analysis and Visualization VI*, pp. 327–342, 2021. doi: 10.1007/978-3-030-83500-2_16 6

[4] R. Blundell and A. Duncan. Kernel regression in empirical microeconomics. *Journal of Human Resources*, 33(1):62–87, 1998. doi: 10.2307/146315 2

[5] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell. A topological framework for the interactive exploration of large scale turbulent combustion. In *2009 Fifth IEEE International Conference on e-Science*, pp. 247–254, 2009. doi: 10.1109/e-Science.2009.42 1

[6] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37:103–120, 2007. doi: 10.1007/s00454-006-1276-5 3, 4

[7] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, and Y. Mileyko. Lipschitz functions have l_p-stable persistence. *Foundations of Computational Mathematics*, 10:127–139, 2010. doi: 10.1007/s10208-010-9060-6 3, 4

[8] J. Curry, H. Hang, W. Mio, T. Needham, and O. B. Okutan. Decorated merge trees for persistent topology. *Journal of Applied and Computational Topology*, 6:371–428, 2022. doi: 10.1007/s41468-022-00089-3 4

[9] Earth Resources Observation and Science Center, U.S. Geological Survey, U.S. Department of the Interior. USGS 30 Arc-Second Global Elevation Data, GTOPO30, 1997. doi: 10.5065/A1Z4-EE71 6

[10] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Amercian Mathematical Society, 2010. doi: 10.1090/mbk/069 3

[11] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002. doi: 10.1007/s00454-002-2885-2 3, 6

[12] B. T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, S. Balakrishnan, and A. Singh. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42(6):2301–2339, 2014. doi: 10.1214/14-AOS1252 3

[13] E. Gobbetti, J. A. I. Guitián, and F. Marton. COVRA: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. *Computer Graphics Forum*, 31(3pt4):1315 – 1324, 2012. doi: 10.1111/j.1467-8659.2012.03124.x 2

[14] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13:723–773, 2012. 3

[15] E. L. Guillou, M. Will, P. Guillou, J. Lukasczyk, P. Fortin, C. Garth, and J. Tierny. TTK is getting MPI-ready. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):5875–5892, 2024. doi: 10.1109/TVCG.2024.3390219 6

[16] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. G. Nashed, and T. Peterka. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):23–33, 2020. doi: 10.1109/TVCG.2019.2934312 2

[17] D. Hoang, P. Klacansky, H. Bhatia, P.-T. Bremer, P. Lindstrom, and V. Pascucci. A study of the trade-off between reducing precision and reducing resolution for data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1193–1203, 2019. doi: 10.1109/TVCG.2018.2864853 2

[18] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008. doi: 10.1214/009053607000000677 2

[19] U. Homberg, D. Baum, A. Wiebel, S. Prohaska, and H.-C. Hege. Definition, extraction, and validation of pore structures in porous materials. In *Topological Methods in Data Analysis and Visualization III*, pp. 235–248, 2014. doi: 10.1007/978-3-319-04099-8_15 1

[20] S. Joshi, R. V. Kommaraji, J. M. Phillips, and S. Venkatasubramanian. Comparing distributions and shapes using the kernel distance. In *Proceedings of the 27th Annual Symposium on Computational Geometry*, pp. 47–56, 2011. doi: 10.1145/1998196.1998204 3

[21] C. Lemaréchal. Cauchy and the gradient method. In M. Grötschel, ed., *Optimization Stories*, Documenta Mathematica Extra Volume, pp. 251–254. European Mathematical Society, Berlin, 2012. Proceedings of the 21st International Symposium on Mathematical Programming (ISMP 2012), Berlin, Germany, August 19–24, 2012. doi: 10.4171/DMS/6/27 5

[22] M. Li, X. Yan, L. Yan, T. Needham, and B. Wang. Flexible and probabilistic topology tracking with partial optimal transport. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):7951–7969, 2025. doi: 10.1109/TVCG.2025.3561300 4

[23] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs. Data reduction techniques for simulation, visualization and data analysis. *Computer Graphics Forum*, 37(6):422–447, 2018. doi: 10.1111/cgf.13336 2

[24] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, and A. M. Gok. SZ3: A modular framework for composing prediction-based error-bounded lossy compressors. *IEEE Transactions on Big Data*, 9(2):485–498, 2023. doi: 10.1109/TBDATA.2022.3201176 6

[25] Y. Liu, Y. Wang, L. Deng, F. Wang, F. Liu, Y. Lu, and S. Li. A novel in situ compression method for CFD data based on generative adversarial network. *Journal of Visualization*, 22:95–108, 2019. doi: 10.1007/s12650-018-0519-x 2

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

[27] G. Ma, D. Lenz, H. Guo, T. Peterka, and B. Wang. Extracting complex topology from multivariate functional approximation: Contours, Jacobi sets, and ridge-valley graphs. *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, 2025. 2

[28] G. Ma, D. Lenz, T. Peterka, H. Guo, and B. Wang. Critical point extraction from multivariate functional approximation. In *IEEE Workshop on Topological Data Analysis and Visualization (TopoInVis)*, pp. 12–22, 2024. doi: 10.1109/TopoInVis64104.2024.00006 1, 2

[29] J. N. P. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein. ACORN: Adaptive coordinate networks for neural scene representation. *ACM Transactions on Graphics (TOG)*, 40(4):1–13, 2021. doi: 10.1145/3450626.3459785 1

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. doi: 10.1145/3503250 1

[31] E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9(1):141–142, 1964. doi: 10.1137/1109020 2

[32] R. B. Neale, A. Gettelman, S. Park, C.-C. Chen, P. H. Lauritzen, D. L. Williamson, A. J. Conley, D. Kinnison, D. Marsh, A. K. Smith, F. Vitt, R. Garcia, J.-F. Lamarque, M. Mills, S. Tilmes, H. Morrison, P. Cameron-Smith, W. D. Collins, M. J. Iacono, R. C. Easter, X. Liu, S. J. Ghan, P. J. Rasch, and M. A. Taylor. Description of the NCAR Community Atmosphere Model (CAM 5.0). Technical report, National Center for Atmospheric Research (NCAR), 2012. doi: 10.5065/wgtk-4g06 6

[33] T. Peterka, Y. Nashed, I. Grindeanu, V. Mahadevan, R. Yeh, and D. Lenz. Multivariate functional approximation of scientific data. In *In Situ Visualization for Computational Science*, pp. 375–397, 2022. doi: 10.1007/978-3-030-81627-8_17 1

[34] T. Peterka, Y. S. G. Nashed, I. Grindeanu, V. S. Mahadevan, R. Yeh, and X. Tricoche. Foundations of multivariate functional approximation for scientific data. In *IEEE 8th symposium on large data analysis and visualization (LDAV)*, pp. 61–71, 2018. doi: 10.1109/LDAV.2018.8739195 1

[35] J. M. Phillips. Coresets and sketches. In *Handbook on Discrete and Computational Geometry*. Chapman and Hall/CRC, 2017. doi: 10.1201/9781315119601 2, 3

[36] J. M. Phillips, B. Wang, and Y. Zheng. Geometric inference on kernel density estimates. In *31st International Symposium on Computational Geometry (SoCG 2015)*, vol. 34, pp. 857–871, 2015. doi: 10.4230/LIPIcs.SOCG.2015.857 3

[37] A. Quarteroni and G. Rozza. *Reduced Order Methods for Modeling and Computational Reduction*. Springer, Cham, 2014. doi: 10.1007/978-3-319

-02090-7 1

[38] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005. doi: 10.7551/mitpress/3206.001.0001 2

[39] P. Rosen, A. Seth, E. Mills, A. Ginsburg, J. Kamenetzky, J. Kern, C. R. Johnson, and B. Wang. Using contour trees in the analysis and visualization of radio astronomy data cubes. In *Topological Methods in Data Analysis and Visualization VI*, pp. 87–108, 2021. doi: 10.1007/978-3-030-83500-2_6 14

[40] B. Schölkopf, C. J. Burges, and A. J. Smola. *Advances in kernel methods: support vector learning*. MIT press, 1998. doi: 10.7551/mitpress/1130.001.0001 2

[41] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. Wiley, 1992. doi: 10.1002/9780470316849 2

[42] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy. Predicting solar generation from weather forecasts using machine learning. In *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 528–533. IEEE, 2011. doi: 10.1109/SmartGridComm.2011.6102379 2

[43] J. Shen, H. Li, J. Xu, A. Biswas, and H.-W. Shen. IDLat: An importance-driven latent generation method for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):679–689, 2023. doi: 10.1109/TVCG.2022.3209419 6

[44] J. Shen and H.-W. Shen. PSRFlow: Probabilistic super resolution with flow-based models for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):986–996, 2024. doi: 10.1109/TVCG.2023.3327171 6

[45] N. Shi, J. Xu, H. Li, H. Guo, J. Woodring, and H.-W. Shen. VDL-Surrogate: A view-dependent latent-based model for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):820–830, 2023. doi: 10.1109/TVCG.2022.3209413 2

[46] N. Shi, J. Xu, S. W. Wurster, H. Guo, J. Woodring, L. P. V. Roekel, and H.-W. Shen. GNN-Surrogate: A hierarchical and adaptive graph neural network for parameter space exploration of unstructured-mesh ocean simulations. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2301–2313, 2022. doi: 10.1109/TVCG.2022.3165345 2

[47] B. W. Silverman. *Density estimation for statistics and data analysis*. Taylor and Francis, New York, 1998. doi: 10.1201/9781315140919 2

[48] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 7462 – 7473, 2020. 1

[49] V. Sitzmann, M. Zollhoefer, and G. Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds., *Advances in Neural Information Processing Systems*, vol. 32, pp. 1121 – 1132. Curran Associates, Inc., 2019. 1

[50] A. Sohankar, C. Norberg, and L. Davidson. Simulation of three-dimensional flow around a square cylinder at moderate reynolds numbers. *Physics of Fluids*, 11:288–306, 1999. doi: 10.1063/1.869879 6

[51] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on Image Processing*, 16(2):349–366, 2007. doi: 10.1109/TIP.2006.888330 2

[52] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The Topology ToolKit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2018. doi: 10.1109/TVCG.2017.2743938 6, 13

[53] X. Tricoche and C. Garth. Topological methods for visualizing vortical flows. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, pp. 89–107. Springer, Berlin, Heidelberg, 2009. doi: 10.1007/b106657_5 1

[54] C. Wang and J. Han. DL4SciVis: A state-of-the-art survey on deep learning for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(8):3714–3733, 2023. doi: 10.1109/TVCG.2022.3167896 2

[55] G. S. Watson. Smooth regression analysis. *Indian Journal of Statistics, Series A*, 26(4):359–372, 1964. 2

[56] T. Weinkauf. 3D flow around a confined square cylinder. `http://tinoweinkauf.net/notes/squarecylinder.html`, 2024. 6

[57] J. R. Wolberg. *Expert Trading Systems: Modeling Financial Markets with Kernel Regression*. Wiley, 2000. 2

[58] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum*, 40(3):599–633, 2021. doi: 10.1111/cgf.14331 3

[59] Y. Zheng, Y. Ou, A. Lex, and J. M. Phillips. Visualization of big spatial data using coresets for kernel density estimates. *IEEE Transactions on Big Data*, 7(3):524–534, 2021. doi: 10.1109/TBDATA.2019.2913655 2

[60] Y. Zheng and J. M. Phillips. Coresets for kernel regression. In *KDD '17: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 645–654, 2017. doi: 10.1145/3097983.3098000 2, 3, 6

SUPPLEMENTAL MATERIALS

In this supplement, we provide details on L_∞ error evaluation in Appendix A, followed by model parameter analysis in Appendix B, runtime analysis for our optimized coreset OPT in Appendix C, parameters used for compressor experiments in Appendix D, as well as additional visual comparisons in Appendix E.

A L_∞ ERROR EVALUATION WITH EVALUATION POINTS

For each dataset, we assess the effectiveness of our optimized coreset by examining the change of function values at a set of evaluation points. Our experimental results demonstrate this: in 2D, using just $1/64$ of the original data P, we obtain an evaluation set Q with sufficient precision for evaluation; in 3D, where the data exhibits greater variability, we use $1/8$ of the points for the evaluation set Q.

We present the L_∞ error results for the 2D GTOPO30 Middle East dataset in Tab. 4 and the 3D Vortex Street dataset in Tab. 5. Specifically, we report the L_∞ distance between KR_P and each KR_S variant, evaluated using both the original data points and the smaller evaluation set. Using the smaller evaluation set, OPT consistently achieves better performance than all baseline coresets, while maintaining the similar L_∞ error on the original data.

Method	$\lvert Q\rvert = \lvert P\rvert = 6,000,000$	$\lvert Q\rvert = \lvert P\rvert/64 = 93,750$
RS	0.3267	0.3286
GR	0.2385	0.2271
GA	0.0769	0.0737
OPT	**0.0178**	**0.0172**

Table 4: GTOPO30 Middle East dataset of size 2000×3000. We compare the L_∞ error between KR_P and each KR_S variant, evaluated on both the full set of data points and on a $1/64$ subsample (the evaluation set). For randomized methods, results are averaged over five independent runs.

Method	$\lvert Q\rvert = \lvert P\rvert = 589,824$	$\lvert Q\rvert = \lvert P\rvert/8 = 73,728$
RS	0.8318	0.8150
GR	0.8303	0.7948
GA	0.4679	0.4654
OPT	**0.3443**	**0.3233**

Table 5: Vortex Street dataset of size $192 \times 64 \times 48$. We compare the L_∞ error between KR_P and each KR_S variant, evaluated on both the full set of data points and on a $1/8$ subsample (the evaluation set). For randomized methods, results are averaged over five independent runs.

B MODEL PARAMETER ANALYSIS

To justify our parameter choices, we analyze the impact of different learning rates η and the number of optimization iterations on the quality of the optimized coresets measured by L_2 error. We use the 2D GTOPO30 Middle East dataset and the 3D Vortex dataset as examples.

Fig. 12 presents the L_2 error over optimization iterations for the Vortex dataset, evaluated with four different learning rates $\eta \in \{0.001, 0.01, 0.1, 1.0\}$. Since the L_2 error with the highest learning rate ($\eta = 1.0$) shows instability in the early stages, we include an additional plot excluding this value to better illustrate the convergence behavior at smaller learning rates. Based on these observations, we select $\eta = 0.1$ with 30 iterations for the Vortex dataset, as it achieves smooth and stable convergence with a small number of iterations.

A similar trend is observed in Fig. 13 for the GTOPO30 Middle East dataset. We experiment with a number of learning rates $\eta \in \{0.01, 0.1, 1.0, 2.0\}$ on the left and provide the L_2 error convergence for smaller learning rates on the right. We select $\eta = 1.0$ with 30 iterations to achieve smooth and stable convergence.

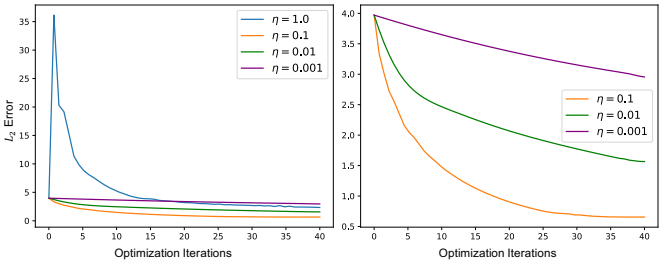

Fig. 12: Vortex dataset. L_2 error vs. the number of iterations across multiple learning rates ($\eta \in \{0.001, 0.01, 0.1, 1.0\}$) on the left, with a plot on the right excluding a high learning rate (of $\eta = 1.0$). The kernel bandwidth is 5.

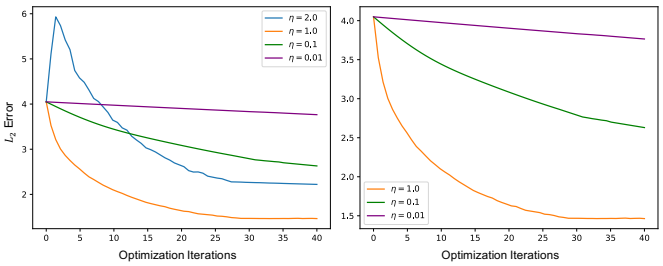

Fig. 13: GTOPO30 Middle East dataset. L_2 error vs. the number of iterations across multiple learning rates ($\eta \in \{0.01, 0.1, 1.0, 2.0\}$) on the left, with a plot on the right excluding a high learning rate (of $\eta = 2.0$). The kernel bandwidth is 5.

C RUNTIME ANALYSIS

We report runtime across datasets under the standard settings listed in Table 1, evaluated over 30 iterations with $\sigma = 5$ for 2D datasets and $\sigma = 1$ for 3D datasets. As shown in Tab. 6, smaller datasets such as CESM, GTOPO-ME, and Vortex Street complete all 30 iterations and reach an optimized configuration in about one minute. In contrast, larger datasets such as GTOPO-NA and Vortex require longer runtimes due to the greater number of query points and the larger coreset sizes involved in each iteration.

Dataset	CESM	NA	ME	Viscous Finger	Vortex Street	Vortex
Runtime	73.5	723.6	68.1	111.9	71.1	534.3

Table 6: Coreset OPT construction time across datasets. All times are reported in seconds. "NA" and "ME" denote GTOPO-NA and GTOPO-ME, respectively.

We analyze the runtime of our optimized coreset OPT for a single optimization iteration under varying parameter choices, including kernel bandwidth, evaluation set size Q, and coreset size, using the CESM dataset. The total runtime to construct the final coreset scales linearly with the number of iterations. For each experiment, we adopt the standard CESM setting with kernel bandwidth $\sigma = 5$, evaluation set size $Q = 1800 \times 3600$, and coreset size of 0.25%. We vary one parameter at a time while keeping the other two fixed, as shown in Fig. 14.

As the kernel bandwidth increases, the runtime per iteration grows approximately quadratically. In contrast, varying the evaluation set size Q exhibits an approximately linear trend, since a larger number of query points must be evaluated in each iteration while the bandwidth and coreset size remain fixed. Similarly, increasing the coreset size leads to longer runtimes per iteration, as more coreset points are optimized during gradient descent and more neighbors fall within the same bandwidth during evaluation.

We further analyze the trade-off between runtime and accuracy by varying the number of optimization iterations and measuring both the L_2 and L_∞ errors (see Fig. 15). Although our optimization procedure explicitly minimizes the L_2 error, the L_∞ error likewise shows a consistent decreasing trend and converges as the number of iterations increases.

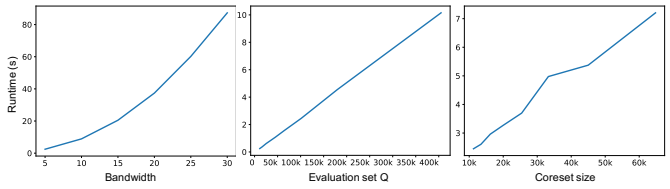

Fig. 14: Runtime analysis of a single optimization iteration under varying kernel bandwidths, evaluation sets Q, and coreset sizes for the CESM dataset. All times are reported in seconds.

Overall, the construction time of the coreset OPT is governed by three key parameters: kernel bandwidth, evaluation set size Q, and coreset size. Based on the coreset size analysis presented in the main paper and the L_∞ error evaluation with respect to Q in Appendix A, we observe that comparable accuracy can be achieved with a relatively small coreset, yielding substantial runtime savings compared to evaluations over the full domain. To balance computational efficiency and approximation quality, we adopt 30 optimization iterations as our standard configuration.

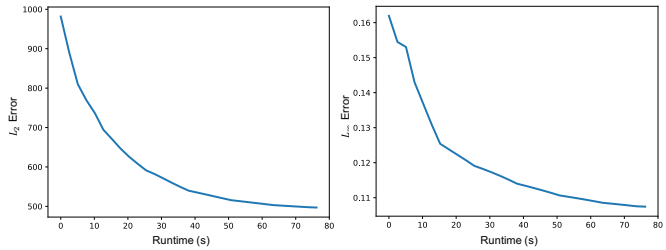

Fig. 15: Runtime–accuracy trade-off as the number of optimization iterations increases (up to 30) for the CESM dataset. All times are reported in seconds, and L_∞ errors are normalized to $[0, 1]$.

D Parameters Used for Compressor Experiments

We report the parameters and file sizes used in comparisons with the lossy compressors SZ3 and ZFP, as well as the topology-preserving compressor TopoQZ.

SZ3 and ZFP each take a parameter ξ, which specifies an upper bound on the L_∞ error of the decompressed file. TopoQZ, implemented in the Topology ToolKit [52], takes both an L_∞ parameter ξ and a persistence parameter ε. In our experiments, we set $\varepsilon = \xi$.

In Tab. 7, we report the values of ξ for SZ3 and ZFP, and ε for TopoQZ. These parameters are selected so that the resulting compressed file sizes are comparable to those of KR^{OPT}. The values of ξ are expressed as fractions of the function range (e.g., $\xi = 1.0$ corresponds to 100% of the range). Notably, ZFP is highly conservative, and the actual L_∞ errors it produces are often substantially smaller than the specified ξ.

In Tab. 8, we report the actual file sizes produced by each data reduction/compression method (in bytes). For each method, parameter values are determined using a binary search strategy to yield file sizes within 1% of that of KR^{OPT}. In some cases, however, it is not possible for the baseline compressor to achieve a file size within this tolerance. We observe that whenever the discrepancy exceeds 2%, the compressor producing the smaller file size also achieves a lower L_∞ error, thereby making it unambiguous which method performs best with respect to L_∞ error.

With regard to the average L_2 error of the gradient, we contend that the distortion introduced by compression itself is less consequential than the error incurred by computing a discrete gradient in place of an analytic one. This is evidenced by SZ3, which often attains very low L_∞ error yet exhibits comparatively high average L_2 gradient error. For this reason, we regard minor discrepancies in file size as inconsequential in the context of gradient error comparisons.

Dataset	SZ3	ZFP	TopoQZ	KR^{OPT}
Overall L_∞ error				
CESM	$3.7 \cdot 10^{-5}$	$1.3 \cdot 10^{0}$	$1.5 \cdot 10^{-1}$	$6.6 \cdot 10^{-3}$
GTOPO-NA	$1.1 \cdot 10^{-5}$	$2.7 \cdot 10^{-1}$	$1.7 \cdot 10^{-1}$	$4.5 \cdot 10^{-3}$
GTOPO-ME	$1.4 \cdot 10^{-5}$	$3.8 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$	$4.5 \cdot 10^{-3}$
Viscous Finger	$2.3 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	$7.3 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$
Vortex Street	$4.9 \cdot 10^{-5}$	$2.7 \cdot 10^{-4}$	$2.7 \cdot 10^{-3}$	$3.7 \cdot 10^{-3}$
Vortex	$5.3 \cdot 10^{-5}$	$5.5 \cdot 10^{-4}$	$8.9 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$
Average L_2 error of the gradient				
CESM	$1.7 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$2.4 \cdot 10^{-4}$
GTOPO-NA	$7.4 \cdot 10^{-4}$	$1.4 \cdot 10^{-3}$	$8.7 \cdot 10^{-4}$	$7.3 \cdot 10^{-5}$
GTOPO-ME	$1.6 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$1.3 \cdot 10^{-4}$
Viscous Finger	$1.7 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$2.7 \cdot 10^{-4}$
Vortex Street	$1.9 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$	$4.5 \cdot 10^{-3}$
Vortex	$1.9 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$	$1.9 \cdot 10^{-2}$	$3.0 \cdot 10^{-4}$

Table 7: L_∞ parameter ξ used on each lossy compressor to produce a file size similar to KR^{OPT}. For TopoQZ, we set $\varepsilon = \xi$.

Dataset	SZ3	ZFP	TopoQZ	KR^{OPT}
CESM	124565	347792	175673	136168
GTOPO-NA	772902	1412704	909482	776365
GTOPO-ME	269155	311232	264226	268348
Viscous Finger	465716	422832	459987	464164
Vortex Street	211713	208928	206495	210300
Vortex	804302	682312	796523	803008

Table 8: Compressed file sizes of each dataset with each compression/reduction method (in bytes).

E Additional Experimental Results

Viscous Finger dataset. First, we give an example demonstrating the preservation of persistence diagrams using our optimized coreset OPT. For preprocessing, we apply a persistence simplification threshold of 0.01 for the Viscous Finger dataset. As illustrated in Fig. 16, while KR^{GA} maintains the general structure of the persistence diagram, it shows minor misalignment—especially within low-persistence (critical point) pairs—as indicated by the orange blocks. KR^{RS} exhibits more pronounced mismatches, including inconsistencies among high-persistence pairs. In contrast, KR^{OPT} retains a more complete set of persistence pairs, including those with low persistence.

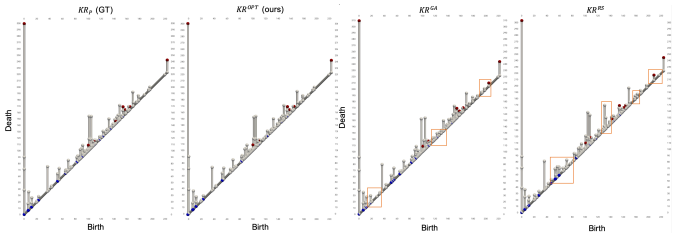

Fig. 16: Viscous Finger dataset. Comparing kernel regression models based on the original data (KR_P), our optimized coreset (KR^{OPT}), grid-aggregated coreset (KR^{GA}), and randomly sampled coreset (KR^{RS}) using persistence diagrams. Orange blocks indicate misalignment of persistence pairs for KR^{GA} and KR^{RS} in comparison with KR_P. Each persistence pair is formed by a pair of critical points: local maxima are in red, local minima are in blue, and saddles are in white. The kernel bandwidth is 2.

ALMA data cube. We further demonstrate the usefulness of kernel regression for smoothing high-frequency noise through a real-world example based on data from the Atacama Large Millimeter Array (ALMA). ALMA observes the sky at millimeter wavelengths, producing three-dimensional data cubes with two spatial dimensions and one spectral (frequency) dimension. These data cubes present unique challenges for scientific visualization: they often exhibit high spectral but

comparatively low spatial resolution, and their complex spatial–spectral structures require careful analysis. Moreover, ALMA data typically have a low signal-to-noise ratio, making it difficult to distinguish true astrophysical signals from background noise. In such cases, kernel regression offers an effective means to suppress high-frequency noise while preserving physically meaningful structures.

We use a data cube originally acquired to study the Ghost of Mirach [39], a galaxy located approximately 10 million light-years from Earth. We extract a 350×350 2D slice along the frequency axis centered on the main signal region. As shown in Fig. 17, applying kernel regression with bandwidth $\sigma = 3$ effectively smooths noise while retaining key signal structures. We further evaluate the kernel regression at a higher resolution by upscaling each spatial dimension by a factor of 2, resulting in an output four times larger in total area. The upscaled result preserves and enhances the structural details of the main signals, demonstrating the method's ability to maintain coherent features at higher resolutions compared to the original slice.

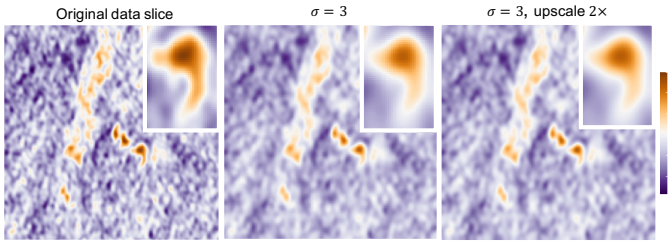

Fig. 17: A 2D slice of the ALMA data cube and corresponding kernel regression results with bandwidth $\sigma = 3$, including a higher-resolution evaluation with an upscale factor of $2\times$.

Additional visual comparisons. Finally, we demonstrate the effectiveness of our framework through additional visual comparisons. For each dataset, we compare KR_{P} and $\mathrm{KR}^{\mathrm{OPT}}$ across multiple kernel bandwidths σ; see Fig. 18, Fig. 19, Fig. 20, Fig. 21, Fig. 22, and Fig. 23. The visualizations show that as σ increases, the data become progressively smoother, with salient boundaries increasingly blurred, particularly in 3D datasets. For example, vortex and turbulent flow structures lose sharpness as their boundaries diffuse into broader regions. For completeness, we also include additional visualizations comparing against $\mathrm{KR}^{\mathrm{GR}}$; see Fig. 24, Fig. 25, and Fig. 26.

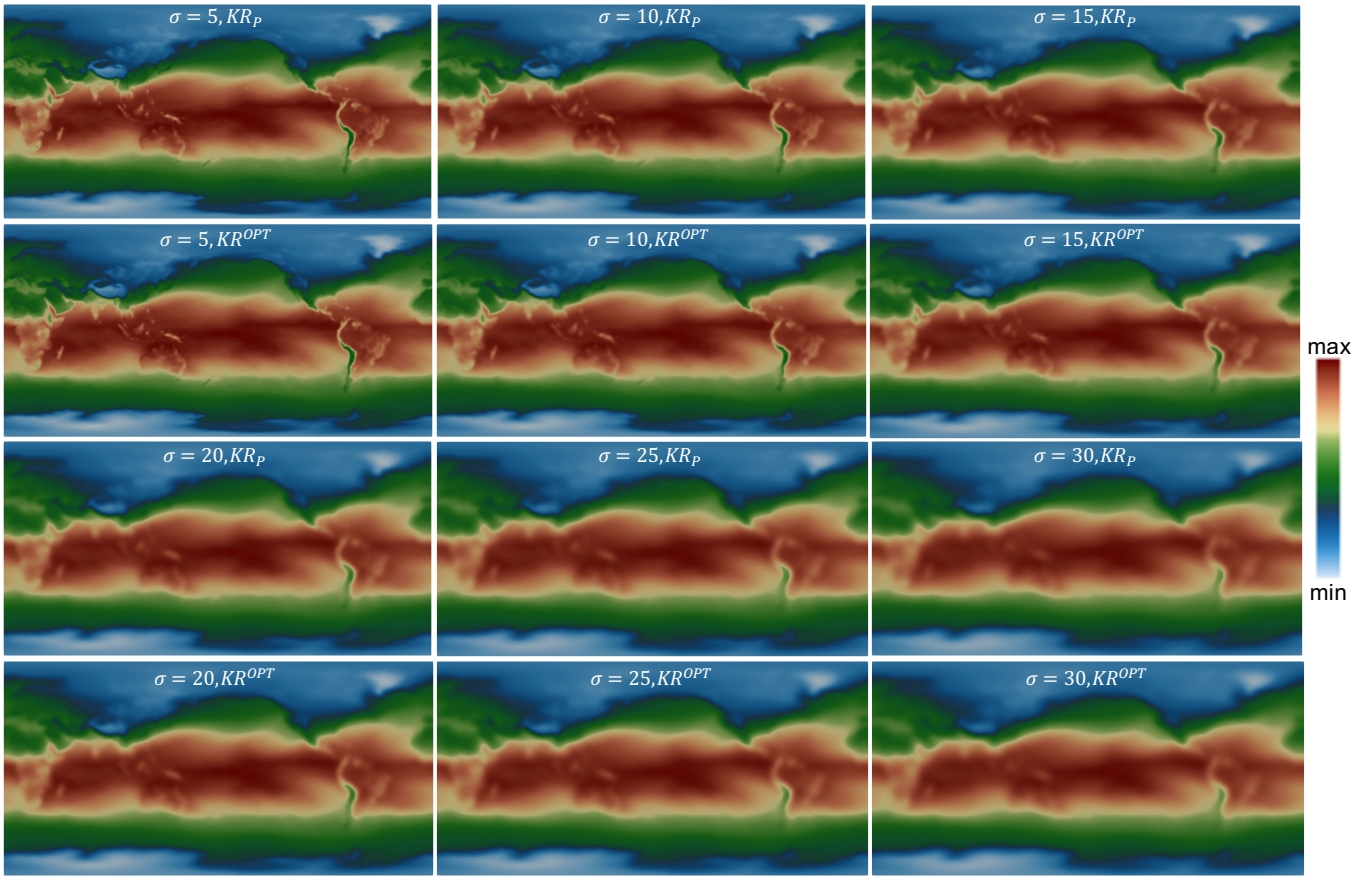

Fig. 18: CESM dataset. A visual comparison of KR_P and KR^{OPT} across a number of bandwidth σ.

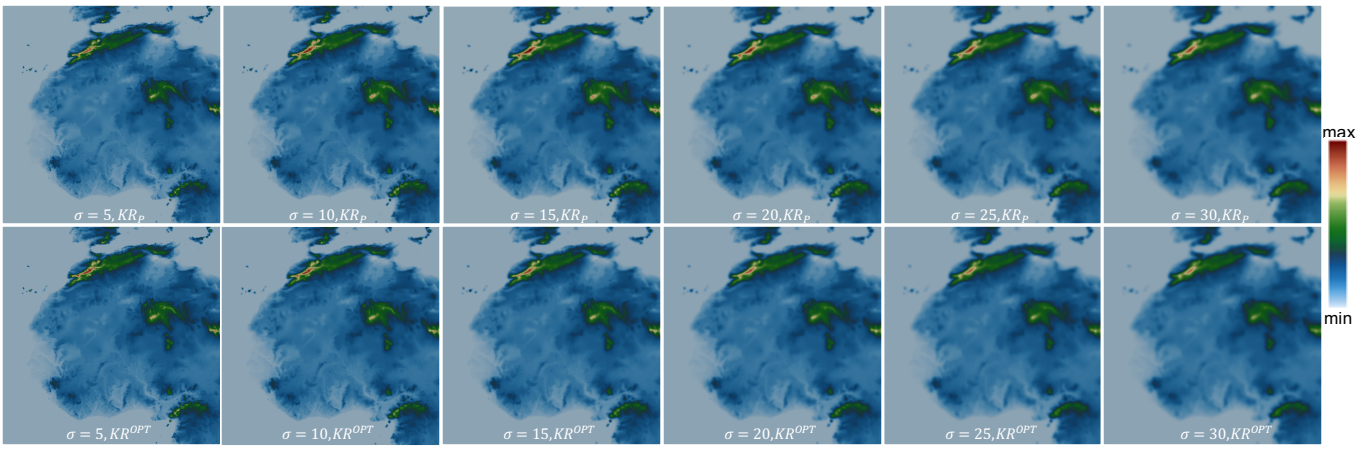

Fig. 19: GTOPO30 North Africa. A visual comparison of KR_P and KR^{OPT} across a number of bandwidth σ.

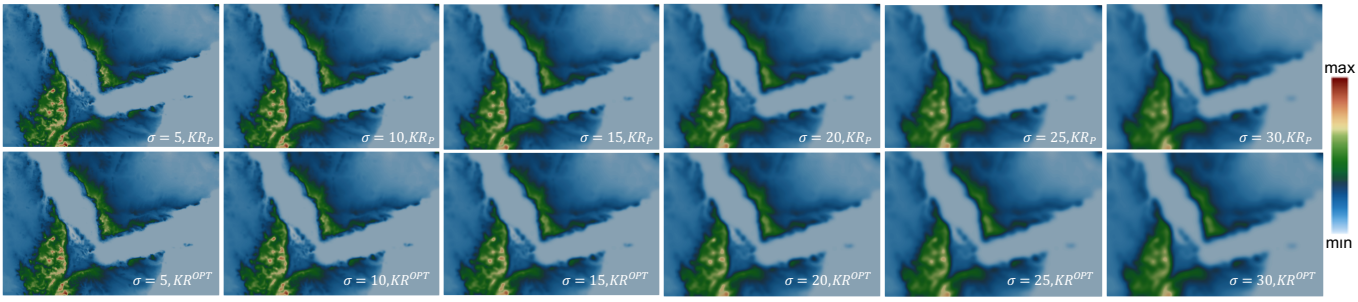

Fig. 20: GTOPO30 Middle East. A visual comparison of KR_P and KR^{OPT} across a number of bandwidth σ.

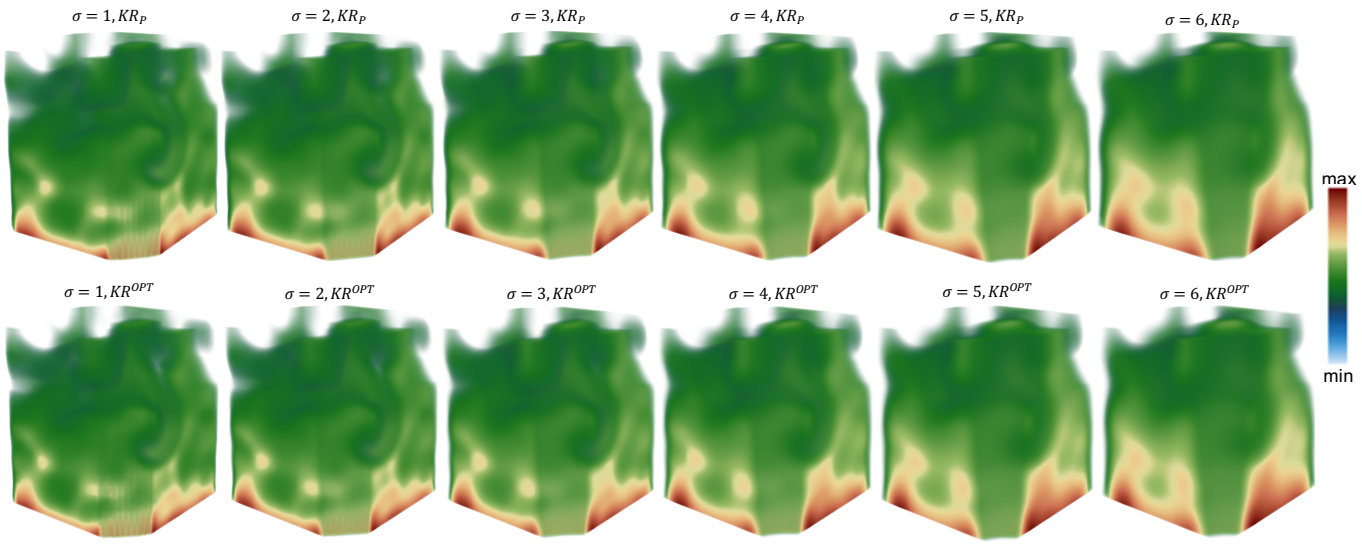

Fig. 21: Viscous Finger dataset. A visual comparison of KR_P and KR^{OPT} across a number of bandwidth σ.

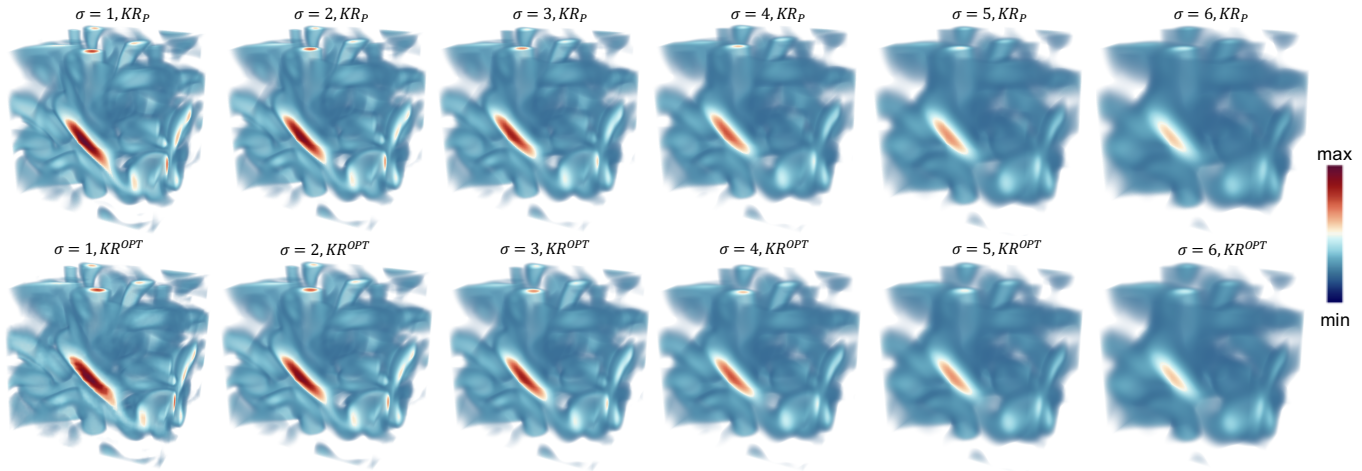

Fig. 22: Vortex dataset. A visual comparison of KR_P and KR^{OPT} across a number of bandwidth σ.

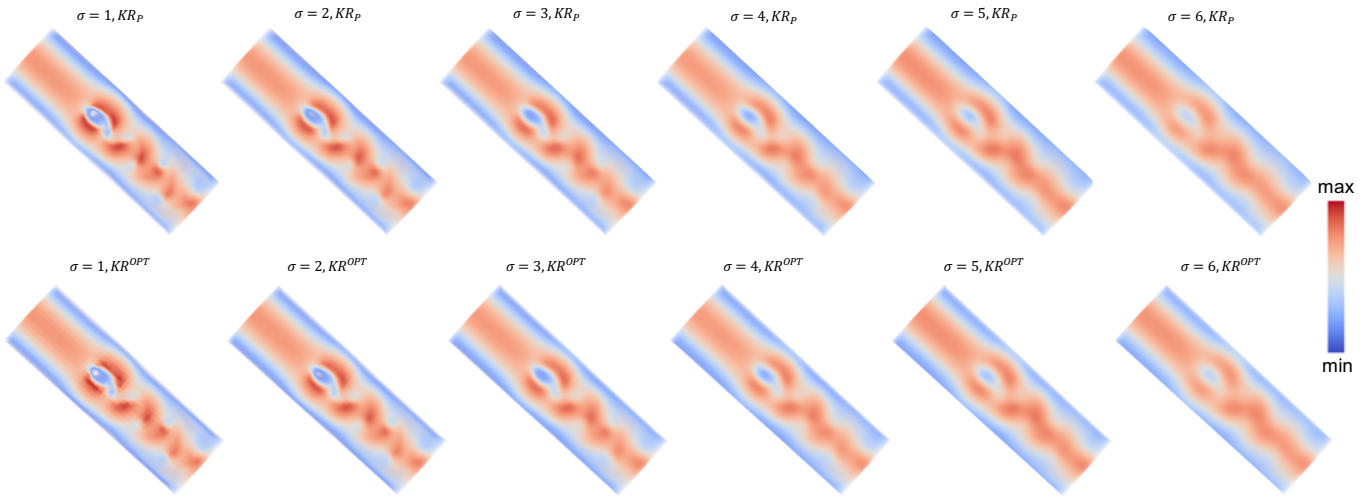

Fig. 23: Vortex Street dataset. A visual comparison of KR_P and KR^{OPT} across a number of bandwidth σ.

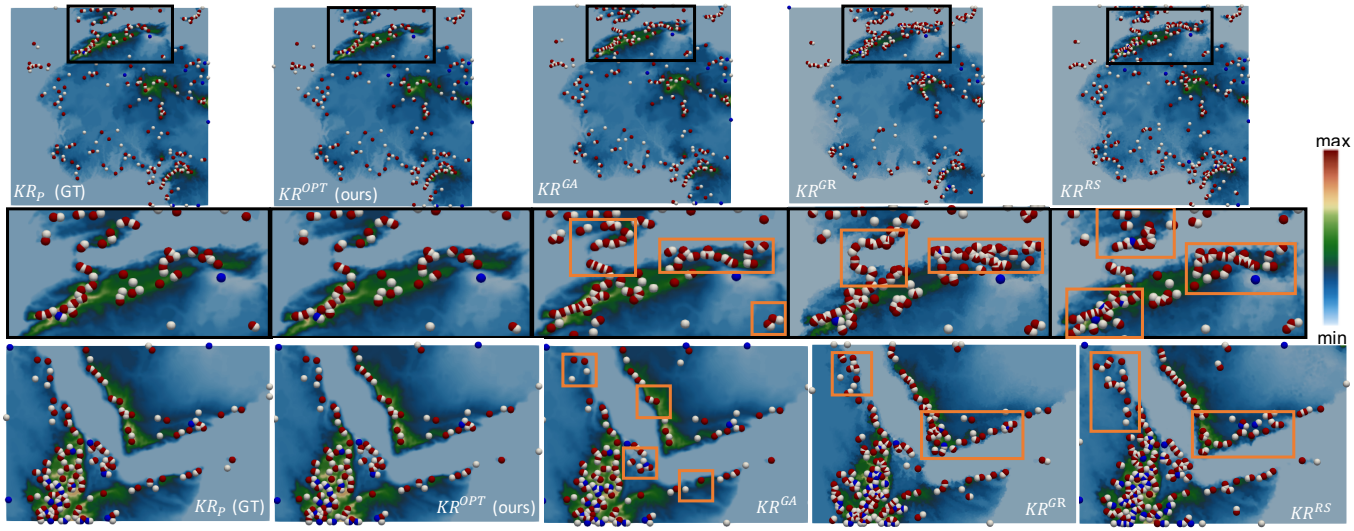

Fig. 24: GTOPO30 dataset. Additional visual comparison including $\mathrm{KR}^{\mathrm{GR}}$ for completeness. Top row: visualization of the critical points overlaid on the North Africa region. Middle row: zoomed-in views of the critical points in the black boxes. Bottom row: visualization of the critical points overlaid on the Middle East region. From left to right: kernel regression models based on the original data (KR_{P}), our optimized coreset ($\mathrm{KR}^{\mathrm{OPT}}$), grid-aggregated coreset ($\mathrm{KR}^{\mathrm{GA}}$), grid-random coreset ($\mathrm{KR}^{\mathrm{GR}}$), and randomly sampled coreset ($\mathrm{KR}^{\mathrm{RS}}$). Orange blocks highlight examples of critical point shifts. Local maxima are in red, local minima are in blue, and saddles are in white. The kernel bandwidth is 10.

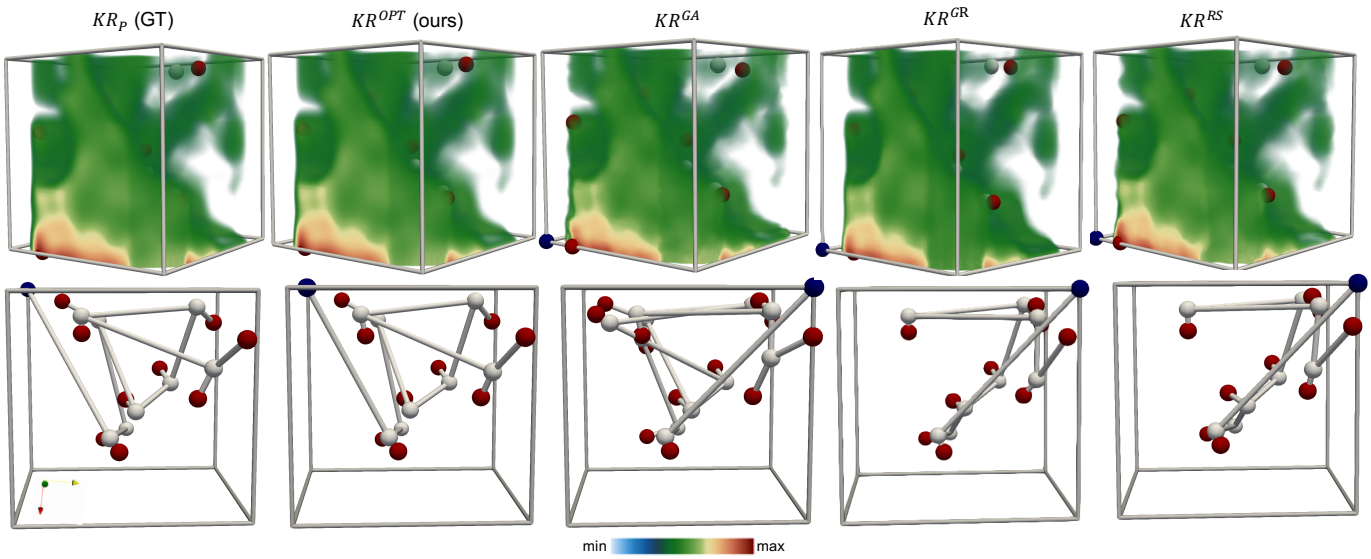

Fig. 25: Viscous Finger dataset. Additional visual comparison including $\mathrm{KR}^{\mathrm{GR}}$ for completeness. Top row: kernel regression models on the original data (KR_{P}), our optimized coreset ($\mathrm{KR}^{\mathrm{OPT}}$), grid-aggregated coreset ($\mathrm{KR}^{\mathrm{GA}}$), grid-random coreset ($\mathrm{KR}^{\mathrm{GR}}$), and randomly sampled coreset ($\mathrm{KR}^{\mathrm{RS}}$). Bottom row: merge trees of the corresponding kernel regression models. Local maxima are red, local minima are blue, and saddles are white. The kernel bandwidth is 2.

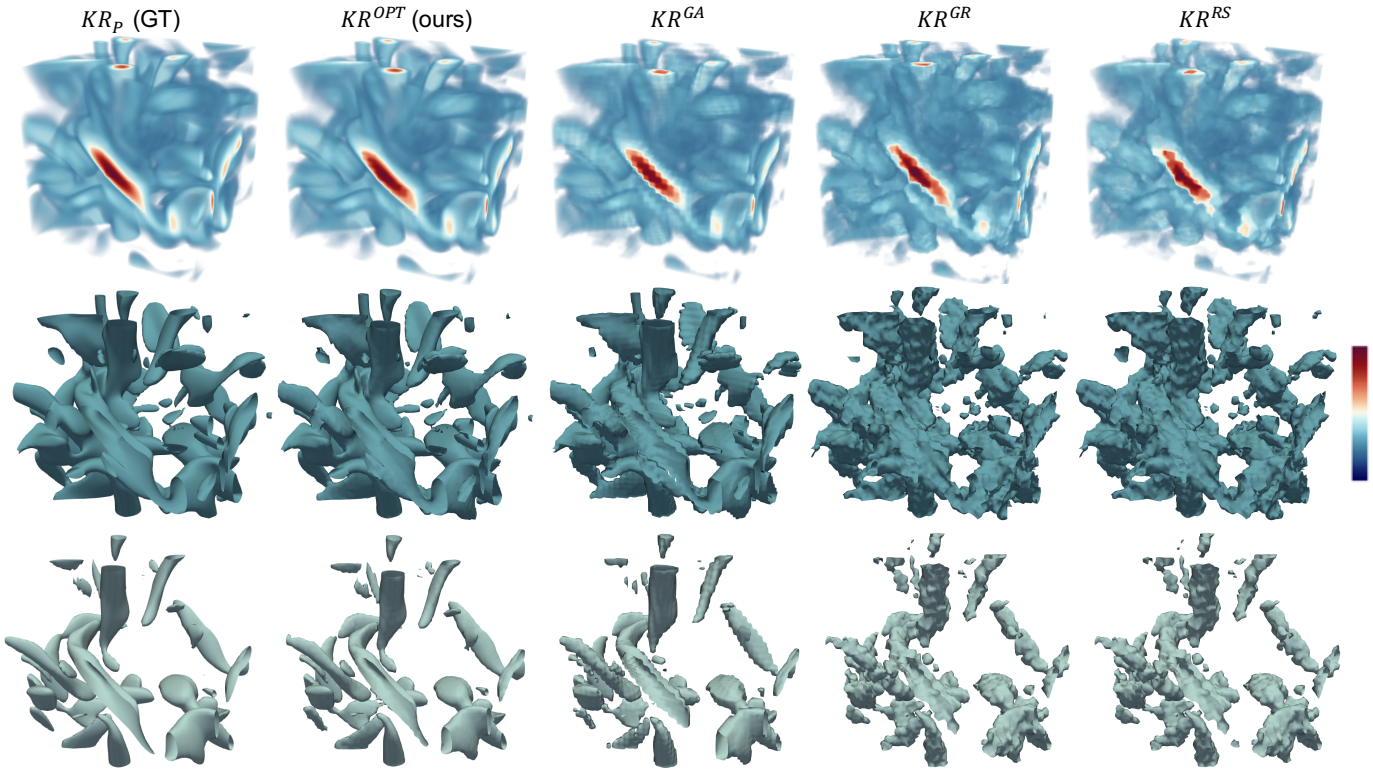

Fig. 26: Vortex dataset. Additional visual comparison including $\mathrm{KR}^{\mathrm{GR}}$ for completeness. From left to right: kernel regression models based on the original data (KR_{P}) considered as the ground truth (GT), our optimized coreset ($\mathrm{KR}^{\mathrm{OPT}}$), grid-aggregated coreset ($\mathrm{KR}^{\mathrm{GA}}$), grid-random coreset ($\mathrm{KR}^{\mathrm{GR}}$), and randomly sampled coreset ($\mathrm{KR}^{\mathrm{RS}}$). From top to bottom: volume rendering of each kernel regression model (top row), isosurface renderings at isovalue 4 (middle row) and 5 (bottom row). The kernel bandwidth is 1.