uPDEt: Efficient Transformer for Unstructured Physics Simulations

Anonymous authors
Paper under double-blind review

Abstract

Deep learning surrogate models have emerged as powerful tools for accelerating Partial Differential Equation (PDE) simulations on complex, unstructured domains and arbitrary geometries. However, existing state-of-the-art architectures often exhibit excessive memory consumption, steep computational overhead, and limited capacity to perform autoregressive dynamics. In this work, we introduce *uPDEt*, a highly efficient and scalable Transformer-based neural operator designed for unstructured physical systems. uPDEt yields highly competitive accuracy while, at the same time, significantly reducing memory footprint and runtime. This results in substantially reduced wall-clock training and inference times compared to top-performing models. With this pairing of high accuracy and high efficiency, uPDEt outperforms existing unstructured neural operators in terms of Pareto optimality across a range of complex 3D benchmarks. Furthermore, while many unstructured models are primarily restricted to steady-state predictions, uPDEt extends to autoregressive rollouts for transient dynamics. Our results highlight uPDEt as a versatile, high-performance foundation for large-scale physical surrogate modeling.

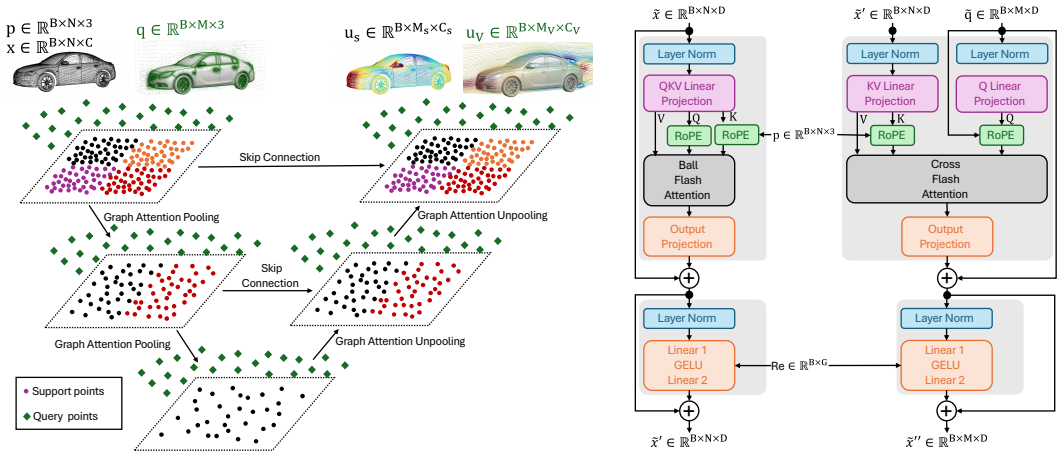
1 Introduction

Traditional numerical simulations remain the gold standard for solving complex scientific and engineering problems in fields such as aerospace design (Evans et al., 2012; Tong, 2023; Anderson, 1995; Ferziger et al., 2020). While these solvers offer remarkable accuracy, their high computational cost remains a major bottleneck, often hindering their use in real-time or large-scale applications. Neural surrogates have emerged as a significantly faster alternative (Lino et al., 2023; Carleo et al., 2019; Vinuesa & Brunton, 2022), with data-driven models evolving rapidly over the last decade, from early convolutional architectures to modern graph-based and Transformer-based neural operators.

Despite these advancements, modeling complex physics on unstructured geometries and 3D domains continues to pose severe computational challenges. Contemporary state-of-the-art unstructured models, such as AB-UPT (Alkin et al., 2025) and SMART (Hagnberger & Niepert, 2026), frequently incur prohibitive memory footprints and steep compute costs when scaled to large parameter capacities or dense spatial meshes. Furthermore, many existing models are restricted to steady-state predictions, lacking the capability to perform autoregressive rollouts for transient systems.

To address these limitations, we present *uPDEt* (pronounced as ”up-det”), a highly efficient Transformer-based neural operator. Its core components, a novel fusion of multiscale local window attention from PDE-Transformer (Holzschuh et al., 2025) with point clouds organized as ball trees (Zhdanov et al., 2025), significantly reduce memory overhead while maintaining solution fidelity. On the one hand, multiscale hierarchies are essential for simulating complex fluid flows characterized by broad spatial frequency spectra, strong non-linearities, and cascading energy transfers. This principle directly mirrors multigrid solvers, where distinct resolution levels target specific frequency bands of the error spectrum, and was adapted to neural operators by Holzschuh et al. (2025). However, PDE-Transformer is restricted to regular grids, whereas industrial applications necessitate unstructured geometries. On the other hand, the Erwin architecture (Zhdanov et al., 2025) has explored multiscale concepts with efficient throughput but lacks in terms of predictive accuracy compared to recent baselines. This motivates the main research question of this work: *can we identify and improve the key components of these previous approaches to deliver a highly accurate model while maintaining efficiency and scalability?* In particular, uPDEt inherits from PDE-Transformer

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) uPDEt multiscale structure

(b) uPDEt attention blocks

Figure 1: Main structure of uPDEt. (a) shows its multiscale structure similar to U-Nets. (b) shows the two attention blocks for geometry and for queries. In our experiments, surface predictions are performed through the geometry blocks, and volume predictions through the query blocks. In this sketches, x represent the point features, p the input coordinates, q the output coordinates, u_S the surface fields, and u_V the volume fields.

and draws inspiration from Erwin's ball trees, introducing four critical architectural advancements. uPDEt leverages a Transformer based hierarchy and works as a continuous neural operator, enabling a coupled prediction of surface and volumetric fields. Rather than relying on message passing, we additionally propose an attention-based graph pooling and unpooling mechanism. Finally, we integrate a 3D extension of RoPE (Su et al., 2024) for precise spatial encoding. Together, these enhancements preserve high throughput while delivering substantial gains in predictive accuracy both in steady-state and autoregressive inference tasks.

Our main contributions are summarized as follows:

- **Memory-Efficient & Fast Operator:** We propose a novel neural operator for unstructured geometries that achieves substantially lower memory usage and faster training/inference throughput compared to current leading baselines.
- **High Accuracy Across Domains:** We demonstrate that uPDEt delivers competitive accuracy across challenging 3D benchmark simulations.
- **Unified Steady & Autoregressive Dynamics:** uPDEt seamlessly generalizes to temporal rollouts, enabling robust autoregressive predictions for transient physical dynamics.

2 Related Work

Grid-Based Neural Operators and Regular Discretizations Early approaches to neural physics surrogates primarily relied on regression models using CNNs on uniform Cartesian grids (Thuerey et al., 2020). To learn mappings between function spaces independently of grid resolution, universal operators such as DeepONet (Lu et al., 2021) and Fourier Neural Operators (FNO) (Li et al., 2021) were introduced. Following the success of Transformers in Natural Language Processing (Vaswani et al., 2017) and Computer Vision (Dosovitskiy et al., 2021; Peebles & Xie, 2023; Liu et al., 2021), recent architectures adapt attention mechanisms to physical domains on regular grids. FactFormer (Li et al., 2023b) factorizes self-attention along spatial dimensions to achieve linear complexity, while PDE-Transformer leverages multi-scale attention backbones for multi-PDE pre-training. To address full 3D spatiotemporal problems efficiently, P3D (Holzschuh et al., 2026) introduces space-time factored operator architectures that lower the memory burden of volumetric convolutions and attention. Besides, frameworks like ReViT (Wei et al., 2026) incorporate explicit rotation equivariance into vision transformer primitives.

Table 1: Feature comparison of uPDEt against state-of-the-art neural PDE solvers, geometric operators, and point-cloud transformers.

	GNN	PTV3	Erwin	Transolver++	GeoTransolver	AB-UPT	SMART	uPDEt (ours)
Versatility (Steady / Autoreg.)	✓	✓	✓	✓	✓	✗	✗	✓
Efficient (Low Memory)	✗✗	✓	✓	✓	✗	✗	✗	✓
Speed	✗✗	✓	✓	✓	✗	✗	✗	✓
Accuracy	✗	✗	✗	✓	✓	✓	✓	✓
Scalable (Param. Scaling)	✗	✓	✓	✓	✓	✓	✓	✓
Decoupled Enc/Dec	✗	✗	✗	✗	✓	✓	✓	✓
Global Params (e.g., Re, α)	✓	✗	✗	✓	✓	✗	✓	✓
Mesh-Free	✗	✓	✓	✓	✓	✓	✓	✓
Mesh Info Supported	✓	✓	✓	✓	✓	✗	✗	✓

Unstructured Domain and Graph Neural Operators As the demand for simulating real-world engineering geometries grew, Graph Neural Networks (GNNs) emerged as the primary framework for unstructured meshes (Pfaff et al., 2021; Battaglia et al., 2016; Lino et al., 2022; Hernández et al., 2024). Early GNN physics solvers relied on iterative spatial message passing over node neighborhoods, but they inherently struggled to propagate long-range physical interactions across large 3D meshes with millions of degrees of freedom. To overcome message-passing bottlenecks, modern approaches blend graph representations with global attention mechanisms (Veličković et al., 2018). Hybrid models include OFormer (Li et al., 2023a), OctFormer (Wang, 2023), which uses octree representations for 3D sparse spatial attention, GNOT (Hao et al., 2023), RIGNO (Mousavi et al., 2025), UPT (Alkin et al., 2024), and GIST (Rigotti et al., 2026; Thumiger et al., 2026).

Because standard attention exhibits prohibitive $\mathcal{O}(N^2)$ computational complexity, recent models employ specialized reduction strategies. Localized windowing, such as Erwin (Zhdanov et al., 2025), HAMLET (Bryutkin et al., 2024), GINO (Li et al., 2023c), GAOT (Wen et al., 2026), and PTV3 (Wu et al., 2024b), restricts attention to dynamic local point neighborhoods or k-NN graphs. Alternatively, anchor-based latent mappings, exemplified by AB-UPT and SMART, compress fine mesh nodes into a smaller set of latent query anchors before performing global attention. Transolver (Wu et al., 2024a) and its extensions, such as GeoTransolver (Adams et al., 2025), introduce sliced physics attention to dynamically aggregate features across learnable physical slices.

Recent frameworks also explore domain-specific operators (Curvo et al., 2026; Zhong et al., 2026; Zhang et al., 2026), notably CarCrashNet (Elrefaie et al., 2026) for non-linear structural crash dynamics and AeroJEPA (Giral et al., 2026) for joint-embedding aerodynamic flow prediction. While these recent frameworks push the frontier of specific downstream domains, uPDEt focuses on providing a general, memory-efficient, and highly scalable core operator capable of unified steady-state and autoregressive rollout learning across generic unstructured geometries. As summarized in Table 1, uPDEt achieves great accuracy with top efficiency and versatility.

3 Methodology

Problem formulation Autoregressive surrogate models for PDEs can be formulated as geometry-conditioned dynamical systems. Let $\Omega(\mathcal{S}) \subset \mathbb{R}^n$ denote a spatial domain defined by a geometry $\mathcal{S}$, e.g., via boundary conditions on $\partial\Omega$, and let $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^m$ be the physical state. We consider PDEs of the form $\frac{\partial \mathbf{u}}{\partial t} + \mathcal{N}(\mathbf{u}, \mathbf{x}; \mathcal{S}, p, t) = 0, \quad \mathbf{x} \in \Omega(\mathcal{S}), \quad t > 0$, where $\mathcal{N}$ encodes differential operators subject to a geometry $\mathcal{S}$ and global parameters p. Discretizing $\Omega(\mathcal{S})$ by a point cloud $x = \{\mathbf{x}_i\}_{i=1}^N$, the state is represented as $\mathbf{u}_t \in \mathbb{R}^{N \times m}$. Then, the evolution operator is conditioned on both the current state and the geometry, and we seek a surrogate $f_\theta : (\mathbf{u}_t, \mathbf{x}, \mathcal{S}, p) \mapsto \mathbf{u}_{t+1}$. Training proceeds via minimization over trajectories, $\min_\theta \sum_{j,t} \left| f_\theta(\mathbf{u}_t^{(j)}, \mathbf{x}_t^{(j)}, \mathcal{S}^{(j)}, p^{(j)}) - \mathbf{u}_{t+1}^{(j)} \right|^2$. In particular, steady-state solutions correspond to fixed points of the geometry-conditioned and appropriately averaged dynamics, i.e., $\bar{\mathbf{u}}(\mathcal{S}, p)$ s.t. $\mathcal{N}(\bar{\mathbf{u}}, \mathbf{x}; \mathcal{S}, p) = \mathbf{0}$, with the mapping $f_\theta : (\mathbf{0}, \mathbf{x}, \mathcal{S}, p) \mapsto \bar{\mathbf{u}}$. Due to this independence from previous states, training for steady-state scenarios is formulated as $\min_\theta \sum_j \left| f_\theta(\mathbf{0}, \mathbf{x}^{(j)}, \mathcal{S}^{(j)}, p^{(j)}) - \bar{\mathbf{u}}^{(j)} \right|^2$.

3.1 MULTISCALE HIERARCHY AND SPATIAL PARTITIONING

We propose **uPDEt** (**u**nstructured **PDE**-**T**ransformer), a multiscale hierarchical Transformer for PDEs on unstructured domains and complex geometries. Figure 1 and Algorithm 1 summarize the overall multiscale forward pass. Inspired by the hierarchical processing of U-Nets, multigrid solvers, and recent spatial transformers (Lino et al., 2022; Holzschuh et al., 2025; Zhdanov et al., 2025), uPDEt was designed to process physical phenomena across a broad spectrum of spatial frequencies. Given a physical support/geometry with N nodes on a D-dimensional space $p \in \mathbb{R}^{B\times N\times D}$ and associated node features $x \in \mathbb{R}^{B\times N\times C_{\text{in}}}$, the architecture processes spatial features recursively over H hierarchical levels. For steady-state problems, x concatenates node coordinates with geometric attributes (e.g., surface normals, cell volumes, or boundary indicators); for transient simulations, x contains previous physical field states. Prior to entering the multiscale blocks, features are projected into continuous latent representations: $\tilde{x} = \text{MLP}_{\text{in}}(x), \quad \tilde{q} = \text{MLP}_{\text{in}}(q)$ where $q \in \mathbb{R}^{B\times M\times D}$ denotes arbitrary spatial query coordinates for the output if available. At each level $h \in \{1, \dots, H\}$, a ball tree partitioning algorithm organizes the spatial domain into $M_h = N_h/b_h$ non-overlapping spatial balls $\mathcal{B}_h = \{\mathcal{B}_h^{(i)}\}_{i=1}^{M_h}$, each containing b_h localized support nodes. This spatial partitioning restricts local token interactions to localized neighborhood clusters, maintaining linear computational complexity $\mathcal{O}(N)$ with respect to the total node count. Within each spatial ball, tokens update their representations via localized self-attention. Relative position information is seamlessly injected into the attention mechanism using a multidimensional extension of Rotary Position Embeddings (RoPE) (Su et al., 2024), enabling translation-invariant and orientation-aware geometric interactions across unstructured meshes. Following self-attention, the latent state of each token is processed by an MLP modulated by global system parameters $\xi \in \mathbb{R}^{B\times G}$ (e.g., Reynolds number, Mach number or angle of attack). After propagating features through all hierarchical levels, the refined tokens are independently projected to the physical target variables via a shared output MLP, $y = \text{MLP}_{\text{out}}(\tilde{x}') \in \mathbb{R}^{B\times M\times C_{\text{out}}}$.

Space partitioning We employ a bounding volume hierarchy Pharr et al. (2016) with spheres, i.e. *balls*, to hierarchically partition the physical domain into local spatial neighborhoods. Compared to other partitioning schemes such as k-d trees or octrees, these trees eliminate directional grid bias and offer tighter spatial bounds on irregular 3D manifolds. Although spherical bounding volumes are inherently isotropic, they adapt effectively to severe geometric scale disparities, such as the ultra-thin, high-aspect-ratio boundary layers common in fluid mechanics. This adaptability arises because boundaries typically have a high density of mesh nodes rather than fixed Euclidean radii: regions with dense discretization, such as refined wall boundaries, naturally cause the bounding balls to contract into tightly focused micro-clusters. Combined with our multiscale architecture, this partitioning allows the network to automatically adapt to the refinement of high-quality simulation meshes. For window-based attention it is additionaly crucial to ensure communication across windows to prevent boundary artifacts. This is addressed with uPDEt's multiscale structure, where higher level communicate across the boundaries of lower levels. This is augmented with rotating the tree structure in consecutive layers (Zhdanov et al., 2025), similar to the shifted windows in SWIN (Liu et al., 2021) architectures.

Queryable decoder Up to now, the description of uPDEt has considered identical input and output discretizations, typical in autoregressive rollouts. However, for discretization-invariant operators, such as mapping geometry to steady-state physical fields, our architecture accommodates arbitrary query locations similar to AB-UPT and SMART. Crucially, query points remain mutually independent throughout the entire network, without inter-query communication, preserving the continuous evaluation property of a neural operator. To yield fine-grained predictions at target coordinates, continuous query tokens $\tilde{q}$ cross-attend to localized geometric representations $\tilde{x}$ within each of the corresponding hierarchical levels. Spatial RoPE is likewise applied across query–support coordinate pairs, enabling queries to selectively aggregate multiscale features from proximal support nodes.

3D Spatial Rotary Position Embedding (RoPE) Unlike standard Transformers relying on absolute Fourier position encodings, uPDEt incorporates relative geometry directly into attention layers via a D-dimensional extension of RoPE (Su et al., 2024). Per-head channels d_h are evenly partitioned across spatial axes ($d_{\text{axis}} = d_h/D$). For each coordinate axis $c \in \{1, \dots, D\}$ and channel pair k, projection vectors are modulated by continuous frequencies $\omega_k = 10000^{-2k/d_{\text{axis}}}$ scaled by coordinate

Algorithm 1: uPDEt Multiscale Forward Pass

Input: Geometry positions $p \in \mathbb{R}^{B\times N\times D}$, query positions $q \in \mathbb{R}^{B\times M\times D}$, input values $x \in \mathbb{R}^{B\times N\times C_{\text{in}}}$, parameters $\boldsymbol{\xi} \in \mathbb{R}^{B\times G}$

Output: Predictions $y \in \mathbb{R}^{B\times M\times C_{\text{out}}}$

$\tilde{x} \leftarrow \text{MLP}_{\text{in}}(x) \in \mathbb{R}^{B\times N\times C_{\text{emb}}}$;

$\tilde{q} \leftarrow \text{MLP}_{\text{in}}(q) \in \mathbb{R}^{B\times M\times C_{\text{emb}}}$;

Construct local spatial balls $\mathcal{B} = \{\mathcal{B}_h\}_{h=1}^{2H-1}$ where $\mathcal{B}_h = \{B_h^{(i)}\}_{i=1}^{M_h}$, with $M_h = N_h/b_h$ balls of capacity b_h at level h;

for level $h = 1$ **to** $2H-1$ **do**

  $\tilde{x} \leftarrow \text{Graph Pooling/Unpooling}(\mathcal{B}_h, \mathcal{B}_{h-1}, \tilde{x}) \in \mathbb{R}^{B\cdot M_h\times b_h\times C_h}$;

  for layer $l = 1$ **to** L **do**

    `// Self-Attention over support geometry`

    $\tilde{x} \leftarrow \text{LayerNorm}(\tilde{x})$;

    $Q, K, V \leftarrow \text{Projection}(\tilde{x}) \in \mathbb{R}^{B\cdot M_h\times H_h\times b_h\times (C_h/H_h)}$;

    $Q, K \leftarrow \text{ApplyRoPE}(Q, K)$;

    $\tilde{x} \leftarrow \tilde{x} + \text{Projection}\left(\text{softmax}\left(\frac{\text{Re}(QK^\top)}{\sqrt{d_h}}\right)V\right)$;

    $\tilde{x} \leftarrow \text{LayerNorm}(\tilde{x})$;

    $\tilde{x} \leftarrow \tilde{x} + \text{PM-MLP}(\tilde{x}, \boldsymbol{\xi})$;

    `// Cross-Attention decoding into continuous query points`

    $\tilde{x} \leftarrow \text{LayerNorm}(\tilde{x}), \quad \tilde{q} \leftarrow \text{LayerNorm}(\tilde{q})$;

    $K, V \leftarrow \text{Projection}(\tilde{x}), \quad Q \leftarrow \text{Projection}(\tilde{q})$;

    $K \leftarrow \text{ApplyRoPE}(K), \quad Q \leftarrow \text{ApplyRoPE}(Q)$;

    $\tilde{q} \leftarrow \tilde{q} + \text{Projection}\left(\text{softmax}\left(\frac{\text{Re}(QK^\top)}{\sqrt{d_h}}\right)V\right)$;

    $\tilde{q} \leftarrow \text{LayerNorm}(\tilde{q})$;

    $\tilde{q} \leftarrow \tilde{q} + \text{PM-MLP}(\tilde{q}, \boldsymbol{\xi})$;

$y \leftarrow \text{MLP}_{\text{out}}(\tilde{q})$;

p_c. The inner product between transformed queries and keys thereby evaluates to:

$$\langle \mathcal{R}(Q(p_i)), \mathcal{R}(K(p_j))\rangle = \sum_{c=1}^{D}\sum_{k} \text{Re}\left(Q_{c,k}(p_i)K_{c,k}(p_j)^*\, e^{\text{i}s\omega_k(p_{i,c}-p_{j,c})}\right), \tag{1}$$

where s is a spatial scale factor. This formulation guarantees that attention weights depend strictly on relative spatial displacements $(p_i - p_j)$, ensuring translation invariance and directional awareness across unstructured geometries without requiring explicit mesh connectivity.

Parameter-Modulated MLP (PM-MLP) Global scalar parameters $\xi \in \mathbb{R}^{B\times G}$ fundamentally dictate the operational regime of physical systems. To condition intermediate latent features on these global parameters, we employ a Parameter-Modulated MLP (PM-MLP) (Perez et al., 2018) following each self- and cross-attention block:

$$\text{PM-MLP}(\tilde{x}, \xi) = W_2\left(\sigma(W_1\tilde{x} + b_1) \odot \gamma(\xi) + \beta(\xi)\right) + b_2 \tag{2}$$

where $\gamma(\xi), \beta(\xi) \in \mathbb{R}^{C_h}$ are scale and shift modulation vectors generated by auxiliary hypernetworks, $\sigma(\cdot)$ is an activation function (e.g., GELU), and $\odot$ represents elementwise multiplication.

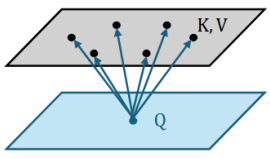


Figure 2: Graph pooling.

Attention Graph Pooling Transferring data between levels of a hierarchy with unstructured data requires an efficient graph coarsening and refinement paradigm. We employ cross-attention for this step, while previous work has, e.g., relied on message passing to aggregate information of child nodes (Zhdanov et al., 2025). For pooling, the parent nodes of one level are the center of mass of the balls in the previous level. Each parent node becomes the query (Q) and cross-attend to its corresponding child nodes (K, V). For unpooling/refinement we repeat the same operation inverting the roles of parent and child nodes.

Table 2: Overview of the datasets used for steady and transient evaluations.

Dataset	Time	Dim.	Type	# Samples	# Points (S+V)	# Fields
DrivAerML (Ashton et al., 2025)	Steady	3D	S+V	500	$\sim$ 9M + 160M	4+4
SHIFT-Wing (Luminary Cloud, 2025)	Steady	3D	S+V	6000	$\sim$ 3M + 6M	4+4
KS (ours)	Transient	2D	V	630×30	4096	1
Ellipse (Valencia et al., 2025)	Transient	2D	V	1200×100	1024	3
SHIFT-Crash (Luminary Cloud, 2026)	Both	3D	S	5128×12	$\sim$ 400K	4

4 RESULTS

Steady-state datasets A variety of datasets for steady-state simulations have recently become available for aerodynamics. To evaluate these problems, we utilize: DrivAerML (Ashton et al., 2025), a large-scale open-source dataset focused on automotive external aerodynamics with 500 variants of the DrivAer notchback, and SHIFT-Wing (Luminary Cloud, 2025), targeting transonic aerospace applications and based on thousands of parametric variations of the NASA Common Research Model (CRM). In both cases, we predict the surface pressure p_s and wall shear stresses $\boldsymbol{\tau}$, alongside volume pressure p_v and velocities $\mathbf{u}$. Secondly, as an additional test case with completely different physics, we use SHIFT-Crash (Luminary Cloud, 2026), an automotive crash simulation dataset.

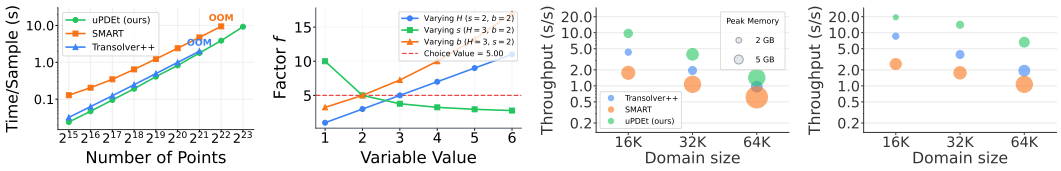
Transient datasets In unstructured settings, transient datasets are less common; consequently, we employ a combination of synthetic and established datasets. First, we generate synthetic data by interpolating regular 2D Kuramoto-Sivashinsky (KS) simulations onto a Gaussian-deformed grid. The second transient dataset is sourced from DGN (Valencia et al., 2025) and consists of 2D flows around an ellipse (predicting pressure and velocity). Finally, we include SHIFT-Crash again. This dataset provides the crash dynamics for 12 time steps. For autoregressive experiments, we use the entire trajectory; for the steady-state case, we simply predict the final stage of the crash based on the initial geometry and global parameters. Further details regarding these datasets are provided in Appendix B and summarized in Table 2.

Baselines We evaluate uPDEt against several established baselines in the field: SMART (Hagnberger & Niepert, 2026), AB-UPT (Alkin et al., 2025), Transolver++ (Luo et al., 2025), GeoTransolver (Adams et al., 2025), Erwin (Zhdanov et al., 2025), and PTV3 (Wu et al., 2024b). In our first comparison, all models are budgeted to a comparable parameter regime of approximately 10–15M parameters (further details are provided in Appendix D). Regarding input representations, AB-UPT and SMART strictly operate on spatial coordinates. However, for architectures that support multi-channel inputs, including uPDEt, we provide all available auxiliary geometric attributes from the dataset, such as face normals and cell areas. Finally, we assess uPDEt on both steady-state problems and autoregressive transient rollouts. SMART and AB-UPT are excluded from the transient rollout benchmarks since they cannot be used autoregressively, although they can use time conditioning for transient problems.

4.1 EFFICIENCY COMPARISON

Beyond its predictive accuracy, uPDEt offers substantial computational efficiency. While matching state-of-the-art accuracy across benchmarks, our architecture achieves reduced memory footprints and lower latency compared to the best recent baselines. For the attention-based processing of the support points, consider a hierarchical encoder-decoder structure with H levels, where level H forms the latent bottleneck. Let L denote the number of attention layers per level, s the coarsening ratio during spatial pooling/unpooling between levels, and b the scaling factor for ball capacities across consecutive levels. The local attention cost at level h scales with the number of balls multiplied by the quadratic intra-ball attention complexity: $\frac{N_h}{B_h} \cdot B_h^2 = N_h B_h$. Summing over the encoder, decoder, and bottleneck stages yields:

$$
\begin{aligned}
C &= \left(2 \sum_{h=1}^{H-1} \frac{N_h}{B_h} B_h^2 + \frac{N_H}{B_H} B_H^2 \right) L = \left(2 \sum_{h=1}^{H-1} \left(\frac{N}{s^{h-1}} \right) \left(b^{h-1} B_1 \right) + \left(\frac{N}{s^{H-1}} \right) \left(b^{H-1} B_1 \right) \right) L \\
&= N B_1 L \left[\left(\frac{b}{s} \right)^{H-1} + 2 \sum_{h=1}^{H-1} \left(\frac{b}{s} \right)^{h-1} \right] = N B_1 L \left[\left(\frac{b}{s} \right)^{H-1} + 2 \frac{1 - (b/s)^{H-1}}{1 - b/s} \right] \qquad (3)
\end{aligned}
$$



(a) Resolution scaling (b) Extra compute term (c) DrivAerML efficiency (d) SHIFT-Crash efficiency

Figure 3: Discussion of uPDEt's efficiency (GPU memory 96GB). (a) shows how the throughput scales with the resolution of the point clouds (more details in Appendix F), (b) plots the architectural factor $f(H, s, b)$ of the theoretical compute, (c) and (d) compares memory footprint and speed of different models under two different applications with varying number of points.

where the final equality applies the finite geometric series identity $\sum_{i=0}^{n-1} r^i = \frac{1-r^n}{1-r}$ with common ratio $r = b/s$ and $n = H - 1$. Consequently, the total computational complexity scales strictly linearly with respect to the input resolution N, modulated by the initial ball capacity B_1, layer depth L, and an architectural factor $f(H, s, b)$. A similar equation applies for the queries processing blocks.

This theoretical derivation is corroborated empirically in Figures 3a and 3b. In comparative benchmarks (Figures 3c and 3d), uPDEt systematically outperforms other baselines in throughput and memory footprint. Most notably, under a matched $\sim$10M parameter regime, uPDEt reduces peak memory consumption by over $2\times$ and attains an order-of-magnitude speedup relative to SMART. Crucially, these primary baseline comparisons do not rely on activation checkpointing; enabling this feature further compresses uPDEt's training memory footprint by roughly $3\times$ at the expense of a modest $\sim 20\%$ time overhead in the backward pass. More details are presented in Appendix F.

4.2 Steady Experiments

Baseline Comparison We train all models with sample resolutions of 32,768 surface points and 32,768 volume points, and perform inference on the entire samples with chunks of the same size. Table 3 summarizes the steady-state performance across all three datasets within the 10–15M parameter regime. Overall, uPDEt achieves highly competitive accuracy across all benchmarks, consistently outperforming established baselines such as AB-UPT and Transolver++, while trailing only slightly behind SMART.

Iso-Resource Comparison with SMART While uPDEt performs below SMART in raw error metrics under parameter-matched settings, it requires an order of magnitude fewer FLOPs and a substantially smaller memory footprint. Parameter matching fails to reflect real-world deployment, where models are fundamentally constrained by GPU memory limits and fixed training compute budgets. To ensure a fair practical evaluation, Table 4 compares, under matched hardware requirements, the previous uPDEt configuration with a smaller version of SMART. To do this, we first find a smaller configuration of SMART with similar memory to our base uPDEt, and then, comparing the final throughput of this smaller version, we adjust the compute budget or total training time (details in Appendices D and F). Under these resource-constrained conditions, uPDEt achieves superior accuracy. We restrict this compute-matched analysis to SMART, since it is the only baseline that consistently outperforms uPDEt in the fixed-parameter setting; for other relevant baselines, such as AB-UPT or Transolver++, uPDEt already achieves superior or comparable accuracy while consuming fewer resources, so an iso-resource comparison would only widen this gap. These results demonstrate that our hierarchical local attention mechanism scales far more favorably, delivering a superior trade-off between predictive accuracy and computational cost.

Pareto Frontier Analysis Jointly evaluating predictive accuracy (Table 3) and computational cost (Figure 3) reveals a clear accuracy-efficiency trade-off across existing architectures. Visualizing accuracy, peak memory consumption, and runtime throughput in Figure 4a highlights two distinct operational regimes: high-accuracy baselines (SMART, AB-UPT and GeoTransolver) suffer from high latency and high memory footprints, limiting their scalability to dense discretizations. Conversely, lightweight models (Erwin and PTV3) achieve high throughput but exhibit insufficient predictive fidelity for practical engineering workflows. uPDEt effectively bridges this gap, defining a dominant Pareto frontier by almost matching state-of-the-art accuracy with great memory usage and throughput.

Table 3: Test relative L_2 error ($\times 10^{-2}$) across DrivAerML, SHIFT-Wing, and SHIFT-Crash benchmarks. Vector quantities ($\boldsymbol{\tau}$, $\mathbf{u}$, $\mathbf{d}$) represent the average error over their 3 spatial components. Test performed with the entire samples by using chunks of 32,768 points. All models are configured with ∼10–15M parameters. **Best** and **second-to-best** results are highlighted.

Model	DrivAerML				SHIFT-Wing				SHIFT-Crash	
	p_s	$\boldsymbol{\tau}_s$	p_v	$\mathbf{u}_v$	p_s	$\boldsymbol{\tau}_s$	p_v	$\mathbf{u}_v$	$\mathbf{d}_s$	τ_s
PTV3	20.38	34.98	29.90	49.11	2.77	26.90	4.75	60.77	26.49	61.53
Erwin	18.66	31.80	16.50	30.05	1.48	14.87	3.28	50.46	34.64	64.69
Transolver++	6.40	12.96	7.63	14.84	0.66	10.00	0.90	31.75	18.26	41.07
GeoTransolver	5.76	11.85	6.28	12.43	**0.64**	**9.78**	**0.42**	33.05	**17.69**	**40.47**
AB-UPT	6.93	12.04	5.70	9.87	0.89	13.12	1.04	33.14	34.26	55.81
SMART	**4.41**	**9.11**	**3.96**	**8.02**	0.72	10.36	0.96	**12.15**	**16.94**	**31.44**
uPDEt	**5.72**	**11.65**	**4.32**	**8.39**	**0.62**	**9.95**	**0.83**	**27.92**	17.90	40.69

Table 4: Test relative L_2 error ($\times 10^{-2}$) across DrivAerML, SHIFT-Wing, and SHIFT-Crash benchmarks. Comparison under the similar memory footprint and compute budget.

Model	DrivAerML				SHIFT-Wing				SHIFT-Crash	
	p_s	$\boldsymbol{\tau}_s$	p_v	$\mathbf{u}_v$	p_s	$\boldsymbol{\tau}_s$	p_v	$\mathbf{u}_v$	$\mathbf{d}_s$	τ_s
SMART-S (same conditions)	6.69	13.15	6.01	12.08	1.01	14.50	1.15	**17.05**	20.72	48.97
uPDEt	**5.72**	**11.65**	**4.32**	**8.39**	**0.62**	**9.95**	**0.83**	27.92	**17.90**	**40.69**

Ablation of key components A comparison with PDE-Transformer and Erwin is particularly instructive for evaluating our individual architectural enhancements. As shown in Table 5, each design choice substantially improves the accuracy. Specifically, attention pooling provides expressive feature transitions across multiscale levels; 3D RoPE refines the Swin-like positional encoding used in both baselines; and the queryable decoder enables a more precise joint prediction of surface and volumetric fields.

Table 5: Ablation study on key uPDEt components evaluated on DrivAerML (rel. L_2 error, $\times 10^{-2}$).

Model	**Attn**	**RoPE**	**Query**	**Error**
uPDET (Erwin/PDE-T)	✗	✗	✗	26.12
+ Attn. pooling	✓	✗	✗	24.29
+ RoPE	✓	✓	✗	9.56
+ Query Dec. (final)	✓	✓	✓	8.81

Point Cloud Resolution Scaling Leveraging uPDEt's linear complexity and lightweight memory footprint, our model can scale to substantially denser point cloud resolutions during training, as shown in Figure 3a and Table 31 (Appendix). A fundamental question is whether increasing spatial resolution during training unlocks performance gains that cannot be realized on coarse point clouds. Figures 4b and 4c show the relative L_2 error normalized with the first 16,384 points chunk case. uPDEt benefits from increased training resolution, yielding sharper accuracy improvements and steeper scaling curves compared to baselines that exhibit smaller improvements.

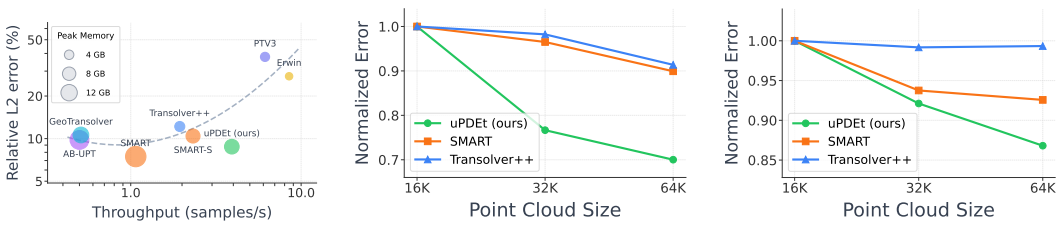
Table 6: Resolution generalization (DrivAerML rel. L_2 error, $\times 10^{-2}$).

Train. res.	**Inf. res.**	**Error**
32K	32K	8.81
32K	64K	8.41
32K	128K	8.32

Finally, we demonstrate that uPDEt achieves cross-resolution generalization simply by scaling the ball sizes. Although trained on point clouds of 32,768 points, the model performs inference at substantially higher resolutions while maintaining predictive accuracy, as shown in Table 6.

4.3 Transient Experiments

Baseline comparison We evaluate the temporal modeling capabilities of uPDEt and the baseline architectures across three transient benchmarks: 2D KS equation, 2D Ellipse, and the highly nonlinear 3D crash simulation. Table 7 reports the normalized root-mean-square error (NRMSE) across autoregressive rollout horizons of 1, 5, and 10 timesteps. These results confirm that our hierarchical local attention mechanism not only excels in static multi-field prediction but also effectively preserves physical consistency across multi-step autoregressive trajectories.



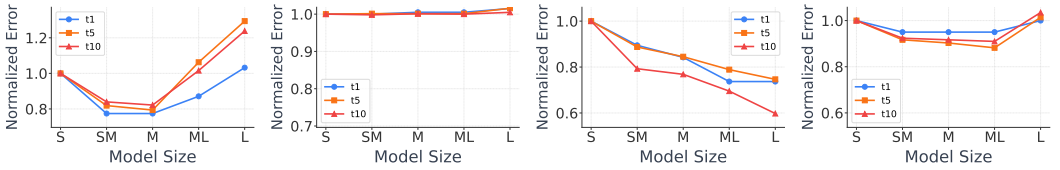
(a) Pareto front in DrivAerML (b) DrivAerML resolution scaling (c) SHIFT-Crash resolution scaling

Figure 4: Scaling analysis. (a) shows the trade-off between accuracy and efficiency in DrivAerML. (b) and (c) show the final accuracy changes with the training resolution and chunk size during inference.

Table 7: NRMSE comparison for transient problems across various architectures and rollout steps.

	2D KS ↓			2D Ellipse ↓			3D Crash ↓		
Timesteps	1	5	10	1	5	10	1	5	10
PTV3	0.200	0.764	**1.026**	0.060	0.264	0.517	0.081	0.261	0.490
Erwin	**0.124**	**0.634**	1.147	0.011	0.046	0.080	0.073	0.275	0.299
Transolver++	0.199	0.761	1.034	**0.004**	**0.012**	**0.019**	**0.040**	**0.144**	**0.154**
uPDEt	**0.031**	**0.160**	**0.449**	**0.004**	0.013	0.020	**0.019**	**0.071**	**0.082**

Model Parameter Scaling uPDEt also exhibits favorable scaling dynamics with respect to parameter capacity. Figure 5 illustrates the performance progression across model configurations spanning from ∼10M (Small) up to ∼200M (Large) parameters. While the performance curve of Transolver++ remains largely flat, uPDEt achieves error reductions with increased capacity. This scaling behavior is particularly prominent when we are not limited by the amount of data, e.g. on rich, large-scale benchmarks such as SHIFT-Crash. In contrast, for smaller-scale systems like the KS equation, the performance plateau is governed by dataset capacity rather than architectural limits, similar to scaling laws in LLMs. When provided with sufficient data, uPDEt exhibits strong neural scaling properties.



(a) uPDEt, KS (b) Transolver++, KS (c) uPDEt, SHIFT-Crash (d) Transolver++, Crash

Figure 5: Model size scaling in transient experiments.

5 Conclusions, Limitations, and Future Work

We introduced *uPDEt*, a novel neural operator for efficient high-fidelity surrogate modeling on arbitrary geometries. With multiscale spatial interactions, uPDEt addresses the computational and memory footprints of prevailing baselines. Through extensive empirical evaluations under parameter-matched experiments, uPDEt demonstrates competitive accuracy while substantially reducing latency and memory utilization. Beyond that, we show that the efficiency/accuracy scaling is substantially better than existing unstructured operators. Moreover, the inherent flexibility allows the model to generalize to dynamic, transient phenomena, achieving superior autoregressive rollout accuracy.

Despite these encouraging results, several challenges outline fruitful avenues for future work. Resolving severe geometric scale disparities, such as thin boundary layers and high-aspect-ratio elements, remains non-trivial. Additionally, while uPDEt currently operates in a purely data-driven regime, integrating physics-informed inductive biases (e.g., hard conservation constraints or soft PDE residual regularization) could further improve physical consistency. Finally, uncertainty estimation remains as an open problem. Overall, the efficiency and flexibility of uPDEt provide a strong foundation for real-time digital twins and large-scale scientific surrogate modeling.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, literature search and code development. We have not used generative AI tools for generating core research ideas, formulating hypotheses or data collection/generation tasks. Additionally, we used generative AI tools for manuscript proofreading and refining LaTeX code structure. We have reviewed all AI-assisted work. Specifically, LLM-generated code was thoroughly verified, profiled, and unit-tested for correctness by the authors, and all AI-refined text was manually checked to ensure accuracy, proper citation of existing literature, and original technical phrasing. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops deep learning surrogates and neural operators for solving partial differential equations (PDEs) in computational physics and engineering settings. All experiments rely on synthetic simulation data; the study does not involve human subjects, personal identifiers, or the collection of sensitive information of any kind.

Two ethical considerations are worth noting given the nature of this research. First, training large neural operators on physics simulations at scale demands substantial computational resources, and the associated energy use carries a non-trivial environmental cost that should be weighed against the long-term efficiency gains such surrogates offer over repeated classical solves. Second, because the proposed methods are designed as general-purpose physical surrogates, the underlying techniques could in principle be repurposed for aerodynamic or structural analysis in dual-use or military-related settings. We note, however, that all models in this paper are trained and evaluated exclusively on publicly available academic benchmarks, and this research is intended solely to advance peaceful scientific understanding and civilian engineering practice.

REPRODUCIBILITY STATEMENT

To ensure the full reproducibility of our results, we have taken several key measures. While high-level architecture designs and core experimental setups are outlined in the main text, full technical specifications—including complete hyperparameter configurations, network layer details, training schedules, and data pre-processing steps for all PDE benchmarks—are thoroughly detailed in the Appendix. Furthermore, the complete source code, training and evaluation scripts, along with open model weights, will be made publicly available in an open-source repository upon publication.

REFERENCES

Corey Adams, Rishikesh Ranade, Ram Cherukuri, and Sanjay Choudhry. GeoTransolver: Learning physics on irregular domains using multi-scale geometry aware physics attention transformer, 2025. URL https://arxiv.org/abs/2512.20399.

Benedikt Alkin, Andreas Fürst, Simon Lucas Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal Physics Transformers: A framework for efficiently scaling neural operators. In The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=oUXiNX5KRm.

Benedikt Alkin, Maurits Bleeker, Richard Kurle, Tobias Kronlachner, Reinhard Sonnleitner, Matthias Dorfer, and Johannes Brandstetter. AB-UPT: Scaling neural CFD surrogates for high- fidelity automotive aerodynamics simulations via anchored- branched universal physics transformers. Transactions on Machine Learning Research, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=nwQ8nitlTZ.

J.D. Anderson. Computational Fluid Dynamics: The Basics with Applications. McGraw-Hill International Editions: Mechanical Engineering. McGraw-Hill, 1995. ISBN 9780071132107.

Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, Burkhard Hupertz, and Danielle Maddix. DrivAerML: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics, 2025. URL https://arxiv.org/abs/2408.11969.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray kavukcuoglu. Interaction networks for learning about objects, relations and physics. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 4509–4517. Curran Associates Inc., 2016. ISBN 9781510838819.

Andrey Bryutkin, Jiahao Huang, Zhongying Deng, Guang Yang, Carola-Bibiane Schönlieb, and Angelica Aviles-Rivero. HAMLET: graph transformer neural operator for partial differential equations. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org, 2024.

Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. Rev. Mod. Phys., 91:045002, Dec 2019. doi: 10.1103/RevModPhys.91.045002.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. Symbolic discovery of optimization algorithms. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=ne6zeqLFCZ.

Pedro M. P. Curvo, Jan-Willem van de Meent, and Maksim Zhdanov. MSPT: Efficient large-scale physical modeling via parallelized multi-scale attention, 2026. URL https://arxiv.org/abs/2512.01738.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Mohamed Elrefaie, Dule Shu, Matthew Klenk, and Faez Ahmed. CarCrashNet: A large-scale dataset and hierarchical neural solver for data-driven structural crash simulation, 2026. URL https://arxiv.org/abs/2605.07098.

G. Evans, J. Blackledge, and P. Yardley. Numerical Methods for Partial Differential Equations. Springer Undergraduate Mathematics Series. Springer London, 2012. ISBN 9781447103776. doi: 10.1007/978-1-4471-0377-6.

Joel H. Ferziger, Milovan Perić, and Robert L. Street. Computational Methods for Fluid Dynamics. Springer, 2020. ISBN 978-3-319-99691-2. doi: 10.1007/978-3-319-99693-6.

Francisco Giral, Abhijeet Vishwasrao, Andrea Arroyo Ramo, Mahmoud Golestanian, Federica Tonti, Adrian Lozano-Duran, Steven L. Brunton, Sergio Hoyas, Hector Gomez, Soledad Le Clainche, and Ricardo Vinuesa. AeroJEPA: Learning semantic latent representations for scalable 3d aerodynamic field modeling, 2026. URL https://arxiv.org/abs/2605.05586.

Jan Hagnberger and Mathias Niepert. SMART: Scalable mesh-free aerodynamic simulations from raw geometries using a transformer-based surrogate model. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=2dzU8sTOO3.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. GNOT: A General Neural Operator Transformer for operator learning. In Proceedings of the 40th International Conference on Machine Learning, ICML'23. JMLR.org, 2023.

Quercus Hernández, Alberto Badías, Francisco Chinesta, and Elías Cueto. Thermodynamics-Informed Graph Neural Networks. IEEE Transactions on Artificial Intelligence, 5(3):967–976, 2024. doi: 10.1109/TAI.2022.3179681.

Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-transformer: Efficient and versatile transformers for physics simulations. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=3BaJMRaPSx.

Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3D: Scalable Neural Surrogates for High-Resolution 3D Physics Simulations with Global Context. In The Fourteenth International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=8UdCE5nhFl.

Felix Koehler, Simon Niedermayr, rüdiger westermann, and Nils Thuerey. APEBench: A benchmark for autoregressive neural emulators of PDEs. In The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2024. URL https://openreview.net/forum?id=iWc0qE116u.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. Transactions on Machine Learning Research, 2023a. ISSN 2835-8856. URL https://openreview.net/forum?id=EPPqt3uERT.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for PDE surrogate modeling. In Thirty-seventh Conference on Neural Information Processing Systems, 2023b. URL https://openreview.net/forum?id=djyn8Q0anK.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for parametric partial differential equations. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-Informed Neural Operator for large-scale 3d PDEs. In Thirty-seventh Conference on Neural Information Processing Systems, 2023c. URL https://openreview.net/forum?id=86dXbqT5Ua.

Mario Lino, Stathi Fotiadis, Anil A. Bharath, and Chris D. Cantwell. Multi-scale rotation-equivariant graph neural networks for unsteady Eulerian fluid dynamics. Physics of Fluids, 34(8):087110, 08 2022. ISSN 1070-6631. doi: 10.1063/5.0097679.

Mario Lino, Stathi Fotiadis, Anil A. Bharath, and Chris D. Cantwell. Current and emerging deep-learning methods for the simulation of fluid dynamics. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 479(2275):20230058, 2023. doi: 10.1098/rspa.2023.0058.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 10012–10022, October 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence, 3(3):218–229, March 2021. ISSN 2522-5839. doi: 10.1038/s42256-021-00302-5.

Luminary Cloud. SHIFT-Wing: High-fidelity computational fluid dynamics dataset for transonic aerospace external aerodynamics, 2025. URL https://huggingface.co/datasets/luminary-shift/Wing.

Luminary Cloud. SHIFT-Crash: High-fidelity crash simulation dataset for automotive structural mechanics, 2026. URL https://huggingface.co/datasets/luminary-shift/SHIFT-Crash.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=AM7iAh0krx.

Sepehr Mousavi, Shizheng Wen, Levi Lingsch, Maximilian Herde, Bogdan Raonic, and Siddhartha Mishra. RIGNO: A graph-based framework for robust and accurate operator learning for PDEs on arbitrary domains. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025. URL https://openreview.net/forum?id=ahJfROJOYt.

NVIDIA Corporation. PhysicsNeMo-CFD: DrivAerML benchmarking workflow. https://github.com/NVIDIA/physicsnemo-cfd/tree/main/workflows/bench_example/drivaer_ml_files, 2024. Accessed: 2026-04-29.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4172–4182, 2023. doi: 10.1109/ICCV51070.2023.00387.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: visual reasoning with a general conditioning layer. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Matt Pharr, Greg Humphreys, Saty Raghavachary, and Pat Hanrahan. Physically based rendering. Elsevier, 2016.

Mattia Rigotti, Nicholas Thumiger, and Thomas Frick. GIST: Gauge-Invariant Spectral Transformers for scalable graph neural operators, 2026. URL https://arxiv.org/abs/2603.16849.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. Neurocomput., 568(C), February 2024. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for Reynolds-Averaged Navier–Stokes simulations of airfoil flows. AIAA Journal, 58(1):25–36, 2020. doi: 10.2514/1.J058291.

Nicholas Thumiger, Andrea Bartezzaghi, Mattia Rigotti, Cezary Skura, Thomas Frick, Elisa Serioli, Fabrizio Arbucci, and A. Cristiano I. Malossi. Faster by design: Interactive aerodynamics via neural surrogates trained on expert-validated CFD, 2026. URL https://arxiv.org/abs/2604.18491.

Mingming Tong. Review of particle-based computational methods and their application in the computational modelling of welding, casting and additive manufacturing. Metals, 13(8), 2023. ISSN 2075-4701. doi: 10.3390/met13081392.

Mario Lino Valencia, Tobias Pfaff, and Nils Thuerey. Learning distributions of complex fluid simulations with diffusion graph networks. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=uKZdlihDDn.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Ricardo Vinuesa and Steven L. Brunton. Emerging trends in machine learning for computational fluid dynamics. Computing in Science and Engg., 24(5):33–41, September 2022. ISSN 1521-9615. doi: 10.1109/MCSE.2023.3264340.

Peng-Shuai Wang. Octformer: Octree-based transformers for 3d point clouds. ACM Trans. Graph., 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592131.

Hao Wei, Bjoern List, and Nils Thuerey. ReViT: Rotational-equivariant vision transformers for neural PDE solvers. In Forty-Third International Conference on Machine Learning, 2026.

Shizheng Wen, Arsh Kumbhat, Levi Lingsch, Sepehr Mousavi, Yizhou Zhao, Praveen Chandrashekar, and Siddhartha Mishra. Geometry aware operator transformer as an efficient and accurate neural surrogate for PDEs on arbitrary domains. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2026. URL https://openreview.net/forum?id=HXFvNkNt0n.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org, 2024a.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point Transformer V3: Simpler, Faster, Stronger. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4840–4851, 2024b. doi: 10.1109/CVPR52733.2024.00463.

Zhuo Zhang, Xi Yang, Ying Miao, Xiaobin Hu, Yifu Gao, Yong Yang, Canqun Yang, and Boocheong Khoo. PGOT: A physics-geometry operator transformer for complex PDEs, 2026. URL https://arxiv.org/abs/2512.23192.

Maksim Zhdanov, Max Welling, and Jan-Willem van de Meent. Erwin: A tree-based hierarchical transformer for large-scale physical systems. In Forty-second International Conference on Machine Learning, 2025. URL https://openreview.net/forum?id=MrphqqwnKv.

Weiheng Zhong, Jing Bi, Victor Oancea, and Hadi Meidani. PGD-NO: A neural operator with precomputed geometry decomposition for 3d million-scale physics simulations. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=h5vFCxA37Y.

Hang Zhou, Haixu Wu, Haonan Shangguan, Yuezhou Ma, Huikun Weng, Jianmin Wang, and Mingsheng Long. Transolver-3: Scaling up transformer solvers to industrial-scale geometries. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=DScXVaBXFy.

TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

These following appendices provide additional details on the following topics:

- A: Problem formulation.
- B: Datasets and training setup.
- C: Hardware requirements.
- D: Model and training hyperparameters.
- E: uPDEt ablation studies.
- F: Efficiency comparisons.
- G: Aerodynamic coefficients.
- H: Model interpretability.
- I: Numerical results with standard deviations.
- J: Additional visual results.

A DETAILS ON PROBLEM DEFINITION

Fundamental physical laws in engineering are primarily formulated as Partial Differential Equations (PDEs), governing phenomena such as fluid dynamics (Navier–Stokes) and structural mechanics (e.g., non-linear crash analysis). While classical numerical solvers provide high accuracy, their computational cost renders them prohibitively expensive for real-time applications and iterative engineering design. Consequently, developing deep surrogate models and neural operators that achieve comparable fidelity while dramatically reducing computational overhead remains a central objective.

As detailed in Section 3, our framework addresses both transient and steady-state regimes. Although both can be expressed within a unified operator learning formulation, their underlying learning dynamics are fundamentally distinct. In autoregressive transient modeling, consecutive temporal states exhibit high mutual correlation, effectively reducing the task to learning local residual updates on top of the current state. In contrast, steady-state problems, which traditional solvers solve via iterative pseudo-time stepping until convergence, require neural surrogates to map the input geometry and global operating conditions directly to the converged equilibrium field. Predicting this non-local mapping is particularly demanding because input geometries and output physical fields occupy distinct feature domains; however, direct steady-state inference inherently circumvents the severe distribution shift and compounding error accumulation typical of long-horizon autoregressive rollouts.

Furthermore, while structured grids allow architectures to focus purely on learning physical dynamics, real-world engineering workflows rely on arbitrary geometries and unstructured discretizations. In this regime, models must capture both intricate spatial geometry and the tight coupling between boundary morphology and physical fields. Industrial CAD geometries are typically discretized as surface meshes or volumetric finite elements. Although explicit topological connectivity, face normals, and cell volumes offer informative priors, many recent approaches favor unstructured point clouds to preserve discretization flexibility. Regardless of the underlying representation, computational complexity for both classical solvers and neural surrogates scales steeply with spatial resolution, highlighting the need for efficient, geometry-aware architectures.

B DETAILS ON DATASETS AND TRAINING SETUP

Scientific and engineering benchmark datasets for physics surrogate modeling typically comprise collections of high-fidelity simulations synthetically generated via classical numerical solvers. Within these datasets, each sample corresponds to a distinct boundary geometry or morphology paired with varying global operating conditions and physical parameters. In this work, we evaluate our approach on rigorous benchmarks spanning both computational fluid dynamics (external aerodynamics) and transient non-linear structural mechanics. A few sample examples are shown in Figure 6.

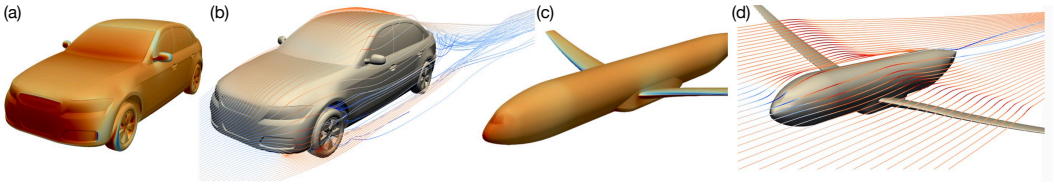


Figure 6: Samples showing surface (pressure in a,c) and volume (velocity streamlines in b,d) fields: DrivAerML (car, left) and SHIFT-Wing (aircraft, right).

DrivAerML comprises high-fidelity automotive external aerodynamics simulations based on the open-source DrivAer model. The underlying flows are computed via scale-resolving hybrid RANS–LES formulations, specifically Delayed Detached Eddy Simulations (DDES) in OpenFOAM, to accurately capture complex, time-averaged turbulent boundary layer separation and unsteady wake structures across both surface and volumetric domains. With each of the 500 samples discretized by over 100 million nodes and requiring approximately $60{,}000$ CPU hours of compute to generate, DrivAerML presents a realistic, industrial-scale testbed characterized by massive spatial resolution and extreme geometric fidelity.

SHIFT-Wing evaluates steady-state transonic aerospace aerodynamics across thousands of compressible Reynolds-Averaged Navier–Stokes (RANS) simulations. The dataset systematically explores parametrically varied wing–body configurations derived from the NASA Common Research Model (CRM). Crucially, the simulations employ adaptive mesh refinement to resolve steep shock–boundary layer interactions, localized pressure gradients, and transonic wave drag phenomena across diverse flight regimes and angles of attack.

SHIFT-Crash extends our evaluation from computational fluid dynamics to transient, highly non-linear solid mechanics. It consists of explicit dynamic finite element simulations executed in OpenRadioss, modeling full-vehicle frontal New Car Assessment Program (NCAP) crash impacts based on a detailed 2010 Toyota Yaris model. This benchmark tracks transient structural responses across successive time steps.

2D KS equation The Kuramoto–Sivashinsky (KS) equation is a canonical fourth-order non-linear PDE widely adopted to benchmark dynamical systems and neural surrogates under chaotic regimes. Governed by

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \alpha \Delta \mathbf{u} + \beta \Delta^2 \mathbf{u} = \mathbf{0}, \tag{4}$$

its dynamics are driven by a continuous interplay between energy production, convective transport, and dissipation. Due to its extreme sensitivity to initial conditions and rich spatio-temporal chaos, the KS equation serves as a rigorous testbed for assessing error accumulation and long-term stability in autoregressive rollouts. This equation is typically solved on regular grids. Specifically, we obtain the base data from APEBench (Koehler et al., 2024) with $\alpha = \beta = 1$, periodic boundary conditions, domain length uniformly sampled from $[10, 130]$, three distinct initial conditions, and a numerical resolution of 256×256. To convert this structured data into unstructured point clouds, we first downsample the grid to 64×64 and apply a Gaussian coordinate perturbation. This deformation is constructed such that the resulting mesh remains intersection-free. We then sample the original KS solution at these new node coordinates via bilinear interpolation. Although synthetic, since the KS dynamics are not directly integrated on unstructured meshes, this setup provides a practical, controlled testbed to assess the autoregressive rollout stability of point-cloud models on chaotic physical systems.

Ellipse The ELLIPSE and ELLIPSEFLOW dataset from Valencia et al. (2025) benchmarks two-dimensional laminar, unsteady incompressible flow past an elliptical cylinder exhibiting periodic vortex shedding. Trajectories were generated using the spectral/hp element framework Nektar++ across Reynolds numbers $Re \in [500, 1000]$ and relative thicknesses $b \in [0.5, 0.8]$, with additional test splits evaluating out-of-distribution variations in viscosity, geometry, and angle of attack. The unstructured discretizations contain approximately 71 nodes for the immersed boundary wall and 2,340 nodes for the surrounding field.

Problem Formulation and Training Objective Given a training dataset $\mathcal{D}_{\text{train}} = \{(G_i, \xi_i, u_i)\}_{i=1}^{N}$, where $G_i \subset \mathbb{R}^{M_i \times 3}$ denotes the spatial coordinates of the geometric domain, $\xi_i \in \mathbb{R}^{d_\xi}$ represents global operating parameters, and $u_i = u(G_i, \xi_i)$ is the target physical field, we train a neural surrogate f_θ by solving

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(G,\xi,u)\sim\mathcal{D}_{\text{train}}} \left[\mathcal{L}\big(f_\theta(G, \xi), u\big) \right], \tag{5}$$

where $\mathcal{L}$ is a prescribed loss function.

Subsampling and Patch Inference Modern engineering datasets routinely contain millions of nodes per simulation, which exceeds GPU memory capacity under full-domain graph or attention operations. To mitigate out-of-memory (OOM) bottlenecks during training, we randomly subsample each geometry to $N_{\text{sub}} = 32{,}768$ points per forward pass, following established practices in recent literature such as SMART (Hagnberger & Niepert, 2026). In Figure 4, we evaluate surrogate performance across different subsampling resolutions. During full-scale evaluation, the complete mesh is partitioned into random spatial chunks of $32{,}768$ points and processed independently in batches before reassembly.

Loss Function and Evaluation Metric To prevent dominant physical fields from biasing optimization, we employ the channel-averaged relative L_2 error as both our primary training loss and test metric. For a ground truth field $y \in \mathbb{R}^{M \times C}$ and model prediction $\hat{y} \in \mathbb{R}^{M \times C}$, the relative L_2 error is defined as

$$\mathcal{L}_{\text{rel}}(y, \hat{y}) = \frac{1}{C} \sum_{c=1}^{C} \frac{\|y_{:,c} - \hat{y}_{:,c}\|_2}{\|y_{:,c}\|_2 + \epsilon}, \tag{6}$$

where $y_{:,c}$ denotes the spatial profile of the c-th physical channel, $\|\cdot\|_2$ is the Euclidean norm over spatial nodes, and ϵ ensures numerical stability. We report the mean and standard deviation across test instances. The predicted channels across each benchmark comprise:

- **DrivAerML** ($C = 8$): Surface pressure (1), wall shear stress vector (3), volumetric pressure (1), and volumetric velocity vector (3).
- **SHIFT-Wing** ($C = 8$): Surface pressure (1), wall shear stress vector (3), volumetric pressure (1), and volumetric velocity vector (3).
- **SHIFT-Crash** ($C = 4$): Displacement vector (3) and equivalent scalar von Mises stress (1).

Data Normalization Because multiscale simulations encompass heterogeneous physical quantities spanning orders of magnitude (e.g., displacements of $\sim 10^{-2}$ m versus acoustic/aerodynamic pressures of $\sim 10^5$ Pa), unnormalized gradients would destabilize training. We normalize each channel independently using standard z-score normalization:

$$\tilde{y}_{:,c} = \frac{y_{:,c} - \mu_c}{\sigma_c}, \tag{7}$$

where μ_c and σ_c denote the empirical mean and standard deviation computed across all points and samples in $\mathcal{D}_{\text{train}}$.

Dataset Partitioning While sample counts are summarized in Table 2, the partitioning strategies differ across benchmarks. For SHIFT-Wing and SHIFT-Crash, we employ uniform random splits. For DrivAerML, we adhere to the benchmark protocol established in NVIDIA PhysicsNeMo (NVIDIA Corporation, 2024): samples are sorted by total aerodynamic drag force, and the test split is selected from the extreme lower and upper tails of the distribution. This setup evaluates out-of-distribution (OOD) extrapolation under severe aerodynamic loading shifts rather than standard in-distribution interpolation.

C HARDAWARE REQUIREMENTS

The experiments were conducted in a combination of NVIDIA GPUs, consisting of L40S (mostly for transient and DrivAerML experiments), A100 (for SHIFT-Wing), and H100 (for SHIFT-Crash). This is clearly indicated in Appendix D. Training times are maximum a couple of days for the slowest model.

D MODEL AND TRAINING HYPERPARAMETERS

Unlike standard paradigms in computer vision and natural language processing, which predominantly rely on Adam-based optimizers, large batch sizes, and mean squared error (MSE) objectives, this conventional training recipe often yields suboptimal convergence and generalization in physical surrogate modeling. In line with recent observations in architectures such as SMART, predictive accuracy improves substantially when replacing MSE with a relative L_2 formulation, adopting smaller batch sizes, and employing the Lion optimizer (Chen et al., 2023). Following these empirical findings, we train our models using Lion paired with a compact batch size across all benchmark tasks.

Regarding model hyperparameters and parameter counts, we first compared fix architectures of 10-15M parameters. While uPDEt, SMART, and AB-UPT jointly predict surface and volumetric fields within a unified architecture, the remaining baselines require separate networks for each domain. To maintain a fair total parameter count, we halve the capacity of each individual network for the latter group.

The rest of this appendix comprises tables with exact hyperparameter values.

Table 8: Hyperparameters for autoregressive training.

Category	Hyperparameter	Value / Description
Data	Training/Test split	KS: 580/50 trajectories Ellipse: 1000/200 trajectories Crash: 5096/32 trajectories
Optimization	Optimizer	Lion
	Max learning Rate	1e-4
	Lr. schedule	Cosine & Warmup
	Batch Size	4
Training	Epochs	50
	Precision	Single (FP32)
	Hardware	1 x NVIDIA L40S

Table 9: Hyperparameters for steady DrivAerML.

Category	Hyperparameter	Value / Description
Data	Training/Test split	436/64 samples
	# points	32,768 (S) + 32,768 (V)
Optimization	Optimizer	Lion
	Learning Rate	1e-4
	LR schedule	Cosine & Warmup
	Batch Size	1
Training	Epochs	200
	Training time	13 hours (uPDEt)
	Precision	Single (FP32)
	Hardware	1 x NVIDIA L40S

Table 10: Hyperparameters for steady SHIFT-Wing.

Category	Hyperparameter	Value / Description
Data	Training/Test split	5000/1000 samples
	# points	32,768 (S) + 32,768 (V)
Optimization	Optimizer	Lion
	Learning Rate	1e-4
	LR schedule	Cosine & Warmup
	Batch Size	1
Training	Epochs	50
	Training time	40 hours (uPDEt)
	Precision	Single (FP32)
	Hardware	1 x NVIDIA A100

Table 11: Hyperparameters for steady SHIFT-Crash.

Category	Hyperparameter	Value / Description
Data	Training/Test split	5096/32 samples
	# points	32,768
Optimization	Optimizer	Lion
	Learning Rate	1e-4
	LR schedule	Cosine & Warmup
	Batch Size	1
Training	Epochs	20
	Training time	2 hours (uPDEt)
	Precision	Single (FP32)
	Hardware	1 x NVIDIA H100

Table 12: SMART (base) model configuration (10-15M parameters).

Hyperparameter	Value / Description
spatial dim	3
Surface channels	4
Volume channels	4
Parameter channels	16 (DrivAerML) 1 (SHIFT-Wing) 6 (SHIFT-Crash)
Latent dim	256
Latent geometry points	4096
Subsampled geometry points	16,384
Num encoder/decoder blocks	8
Num heads	8
pos scale factor	1000
dropout	0.0

Table 13: SMART (small) model configuration (same efficiency as uPDEt).

Hyperparameter	Value / Description
spatial dim	3
Surface channels	4
Volume channels	4
Parameter channels	16 (DrivAerML) 1 (SHIFT-Wing) 6 (SHIFT-Crash)
Latent dim	128
Latent geometry points	4096
Subsampled geometry points	16,384
Num encoder/decoder blocks	8
Num heads	4
pos scale factor	1000
dropout	0.0

Table 14: AB-UPT model configuration.

Hyperparameter	Value / Description
ndim	3
input dim	3
output dim surface	4
output dim volume	4
dim	192
geometry depth	3
num heads	8
blocks	pscscs
num volume blocks	8
num surface blocks	8
radius	0.25
supernode fraction	0.125
num anchors	8192

Table 15: Erwin transformer model configuration.

Hyperparameter	Value / Description
dimensionality	3
c in	3 + 16 (DrivAerML)
c hidden	[4, 64, 128, 128, 256]
ball sizes	[256, 256, 256, 256, 128]
encoder num heads	[1, 4, 8, 16, 16]
encoder depths	[1, 2, 2, 2, 4]
decoder num heads	[1, 4, 8, 16]
decoder depths	[2, 2, 2, 2]
mp steps	0
rotate	True
decode	True
mlp ratio	4

Table 16: GeoTransolver model configuration.

Hyperparameter	Value / Description
space dim	3
n layers	6
n hidden	256
n heads	8
dropout	0.0
activation	gelu
mlp ratio	1.0
fun dim	7
out dim	4
slice num	64
Parameter channels	16 (DrivAerML) 1 (SHIFT-Wing) 6 (SHIFT-Crash)

Table 17: Transolver++ model configuration.

Hyperparameter	**Value / Description**
space dim	3
n layers	6
n hidden	256
n heads	8
dropout	0.0
activation	gelu
mlp ratio	1.0
fun dim	4
out dim	4
slice num	64
Parameter channels	16 (DrivAerML) 1 (SHIFT-Wing) 6 (SHIFT-Crash)

Table 18: PTV3 model configuration.

Hyperparameter	Value / Description
in channels	19 (SHIFT-Wing)
order	z, z-trans, hilbert, hilbert-trans
stride	[2, 2, 2]
enc depths	[2, 2, 2, 2]
enc channels	[32, 64, 128, 256]
enc num heads	[2, 4, 8, 16]
enc path size	[1024, 1024, 1024, 1024]
dec depths	[2, 2, 2]
dec channels	[4 (SHIFT-Wing), 64, 128]
dec num heads	[1, 4, 8]
dec path size	[1024, 1024, 1024]
mlp ratio	2.0

Table 19: Model scaling configurations and hyperparameter specifications across model sizes.

Model	Hyperparameter	**S**	**SM**	**M**	**ML**	**L**
uPDEt	Parameters (M)	15	28	55	102	209
	dim	256	256	512	512	1024
	num heads	4	4	8	8	16
	num layers	2	4	2	4	2
Transolver++	Parameters (M)	14	26	54	106	218
	n hidden	512	512	1024	1024	2048
	n heads	16	16	32	32	64
	n layers	6	12	6	12	6

E Model Ablation Studies

To rigorously evaluate the architectural design choices underpinning uPDEt, we conduct systematic ablation experiments. Unless stated otherwise, all ablations are benchmarked under a standardized training setting on the steady external aerodynamics DrivAerML suite.

E.1 Spatial Partitioning and Window Types

Local attention mechanisms depend critically on the choice of geometric partitioning scheme. In physical systems, continuum dynamics naturally exhibit isotropic or spherical influence radii. However, alternative tree structures and spatial partitionings introduce distinct trade-offs:

- **Octrees and Quadtrees:** While octrees are computationally efficient to construct (27 ms in Table 20), axis-aligned splitting introduces artificial directional anisotropy. Furthermore, on non-uniform boundary layer meshes, octrees frequently result in highly unbalanced tree depths and empty leaf cells.
- **Binary Space Partitioning (BSP):** Although arbitrary hyperplanes accommodate irregular domain contours, BSP trees do not enforce bounded, compact spatial support regions and suffer from substantially higher construction overhead (175 ms).
- **Space-Filling Curves (Z-order / Morton, Hilbert):** Space-filling curves enable fast sequential indexing (14 ms), yet mapping multidimensional manifolds to a 1D sequence inevitably introduces non-local discontinuities across adjacent index boundaries.
- **Uniform Grids and Spatial Hashing:** Regular voxel grids require minimal overhead (13 ms) but cannot adapt to multi-scale resolution gradients where local vertex density varies by orders of magnitude.

In contrast, **Ball Trees** enforce strict, isotropic metric bounds with balanced cluster sizes across adaptive multi-scale levels. Although constructing ball trees incurs a moderate preprocessing overhead (83 ms), it ensures optimal spatial locality for local attention kernels.

Table 20: Tree construction time comparison (2D domain, $N = 4096$ points, window size $W = 64$).

Partitioning Method	Construction Time (ms) ↓
Spatial Grids	13
Z-order (Morton)	14
Octree / Quadtree	27
Ball tree	**83**
BSP tree	175

E.2 Implementation of Ball Tree Construction

Because ball tree construction precedes attention computation, algorithmic efficiency at high resolutions is essential for training throughput. We evaluate two implementations: a standard baseline implementation in native PyTorch and the optimized tree builder introduced by Erwin (Zhdanov et al., 2025). As reported in Table 21, while native PyTorch operations remain fast on small point counts ($N \leq 2^{15}$), the Erwin implementation exhibits superior asymptotic scaling. At large resolutions ($N = 2^{23} \approx 8.4 \times 10^6$ points), Erwin reduces tree construction latency from $9.602\,\text{s}$ to $5.357\,\text{s}$, nearly doubling tree construction throughput.

E.3 FlashAttention Kernel Integration

We investigate the impact of exact attention kernel backends on overall compute throughput. While PyTorch provides automated kernel dispatching (via `scaled_dot_product_attention`), its heuristic selection yields suboptimal tile schedules on irregular sequence allocations. Integrating custom FlashAttention kernels directly into our local ball-attention blocks produces mathematically identical numerical outputs while reducing attention runtime by 67% and trimming peak GPU

Table 21: Ball tree construction time (s) across spatial resolutions.

Implementation	Number of Points (N)								
	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}
PyTorch	**0.028**	0.062	0.091	0.181	0.360	0.947	2.131	4.779	9.602
Erwin (Zhdanov et al., 2025)	0.032	**0.044**	**0.073**	**0.079**	**0.204**	**0.478**	**0.973**	**2.333**	**5.357**

memory usage by $\sim 10\%$. Consequently, we adopt dedicated FlashAttention kernels across all benchmarked uPDEt configurations.

E.4 Query Coordinate Embedding

For queryable volumetric evaluations, the continuous spatial coordinates must be mapped effectively into the latent feature space. We compare a standard Multi-Layer Perceptron (MLP), a frequency-modulated projection using sinusoidal Fourier features, and a hybrid formulation combining both. Table 22 shows that Fourier modulation achieves the lowest relative L_2 error (8.81%), outperforming both the MLP baseline (8.90%) and the combined variant (8.91%) by avoiding redundant over-parameterization.

Table 22: Ablation of query coordinate input projection formulations.

Projection Method	Relative L_2 Error (%) $\downarrow$
MLP + Modulated	8.91
MLP	8.90
Modulated	**8.81**

E.5 Coarsening Stride and Ball Capacity Hierarchy

The multiscale hierarchy in uPDEt is parameterized by the coarsening stride s between levels and the local ball capacity B_h. Table 23 ablates multiple hierarchical schedules under a constant total token capacity. Setting a stride of $s = 2$ with ball capacities scaling as $512 \to 2048 \to 8192$ yields optimal convergence (8.81% error). Restricting ball capacities to smaller sizes ($256 \to 512 \to 1024$) degrades performance by nearly 9% relative (9.60%), indicating that sufficiently wide local receptive fields are critical for capturing long-range convective interactions before coarsening.

Table 23: Ablation on coarsening stride s, ball capacities B_h, and per-level partition hierarchy.

Stride (s)	Ball Sizes (B_h)	Partition Hierarchy (Balls $\times$ Size)	Rel. L_2 (%) $\downarrow$
2	256 / 512 / 1024	$128 \times 256 \ \to \ 32 \times 512 \ \to \ 8 \times 1024$	9.60
8	128 / 256 / 512	$256 \times 128 \ \to \ 16 \times 256 \ \to \ 1 \times 512$	9.52
4	512 / 1024 / 2048	$64 \times 512 \ \to \ 8 \times 1024 \ \to \ 1 \times 2048$	8.94
2	**512 / 2048 / 8192**	$\mathbf{64 \times 512 \ \to \ 8 \times 2048 \ \to \ 1 \times 8192}$	**8.81**

E.6 Positional Encoding Formulation

Because local self-attention is permutation-invariant, positional encodings are required to preserve geometric spatial organization within ball clusters. We compare classical absolute Fourier embeddings, window-relative coordinate encodings (as employed in Swin Transformer, PDE-Transformer, and Erwin), and Rotary Position Embeddings (RoPE) generalized to continuous 3D coordinates. As summarized in Table 24, relative encodings struggle to maintain global coherence (22.55% error), whereas 3D RoPE achieves the strongest geometric inductive bias, reaching an 8.81% relative error.

Table 24: Ablation of positional encoding strategies.

Positional Encoding	Relative Error (%) ↓
Relative (Erwin / PDE-T)	22.55
Classical Fourier	12.33
Rotary (RoPE)	**8.81**

E.7 Graph Pooling and Unpooling Mechanisms

Information transfer across consecutive resolution levels can be mediated through message-passing graph convolutions or cross-attention. As shown in Table 25, replacing heuristic message-passing operations (used in Erwin, 9.20%) with parameterized cross-attention mechanisms reduces the error to 8.81%, as attention dynamic re-weighting better accommodates non-uniform spatial contractions during coarsening and refinement.

Table 25: Ablation of graph pooling and unpooling mechanisms.

Mechanism	Relative Error (%) ↓
Message Passing (Erwin)	9.20
Cross-Attention (uPDEt)	**8.81**

E.8 Surface Field Processing: Support Points vs. Decoupled Queries

In multi-field physical simulations, boundary surface quantities can either be integrated directly into the structural support hierarchy (encoder-decoder geometry path) or decoded separately via query cross-attention alongside the volumetric domain. Table 26 confirms that treating surface nodes as active support points within the hierarchical ball tree yields superior accuracy (8.81% vs. 9.56%), as the structural boundary layer geometry directly informs intermediate latent representations.

Table 26: Ablation on surface field processing representations.

Formulation	Relative Error (%) ↓
As Query Points (Decoupled)	9.56
As Support Points (Geometry)	**8.81**

E.9 Auxiliary Geometric Attributes

Finally, we assess the benefit of enriching input representations beyond raw Cartesian coordinates. Table 27 demonstrates that relying solely on spatial coordinates leads to substantial error (13.47%). Injecting available auxiliary mesh properties, specifically surface normal vectors and local cell areas, lowers the relative error to 8.81%, demonstrating that explicit differential geometry descriptors provide crucial physical priors. However, some models like SMART or AB-UPT were designed for only-coordinates inputs.

E.10 Architectural Hyperparameter Sizing

The internal hidden dimension D and the number of transformer blocks per hierarchy stage L govern total network capacity. To align with our standard 10–15M parameter evaluation regime across baselines, we fix the latent feature dimension to $D = 256$, utilize $L = 2$ attention blocks per hierarchical level, and set the multi-head attention width to 4 attention heads. This parameter configuration provides an optimal balance between expressive capacity and computational throughput. The final hyperparameters for our base uPDEt configuration are listed in Table 28 for steady scenarios and Table 29 for autoregressive experiments.

Table 27: Ablation on geometric input features and auxiliary mesh attributes.

Input Features	Relative Error (%) ↓
Spatial Coordinates Only	13.47
Coordinates + Normals & Cell Areas	**8.81**

Table 28: uPDEt base configuration for steady scenarios.

Hyperparameter	Value / Description
in channels	7 (DrivAerML)
out channels	4 (DrivAerML)
surface channels	0
volume channels	4 (DrivAerML)
dim	$[256, 256, 256]$
cond dim	16 (DrivAerML)
num layers	$[2, 2, 2]$
num heads	$[4, 4, 4]$
spatial dim	3
dropout	0.0
mlp ratio	2.0
ball size	$[512, 2048, 8192]$
stride	$[2, 2]$

Table 29: uPDEt base configuration for transient scenarios.

Hyperparameter	Value / Description
in channels	7 (SHIFT-Crash)
out channels	4 (SHIFT-Crash)
surface channels	0
volume channels	0
dim	$[256, 256, 256]$
cond dim	6 (SHIFT-Crash)
num layers	$[2, 2, 2]$
num heads	$[4, 4, 4]$
spatial dim	3
dropout	0.0
mlp ratio	2.0
ball size	$[512, 2048, 8192]$
stride	$[2, 2]$

F Efficiency comparisons

To quantify computational overhead under realistic deployment constraints, we benchmark uPDEt against competing baselines across inference throughput, peak GPU memory allocation, floating-point operations (FLOPs), and asymptotic scaling with respect to point resolution. For models requiring decoupled networks for disjoint surface and volumetric predictions (such as Transolver++, Erwin, and PTV3), reported compute metrics account for both sub-networks combined to ensure a strictly fair iso-capacity comparison against unified multi-field architectures (uPDEt, SMART, and AB-UPT).

Inference and Memory Profiling Table 30 summarizes performance benchmarks evaluated on a single NVIDIA L40S GPU across the steady DrivAerML (32K surface + 32K volume points) and SHIFT-Crash (32K points) domains. Key observations include:

- Throughput and Compute Complexity: High-capacity baselines like AB-UPT and SMART incur steep computational overheads (24.6 TFLOPs and 13.3 TFLOPs on DrivAerML),

translating to slow inference rates (0.50 and 1.07 samples/s). In contrast, uPDEt executes at 3.92 samples/s with only 1.7 TFLOPs—delivering a $\sim 3.7\times$ speedup and nearly an order-of-magnitude reduction in FLOPs over SMART.

- Memory Footprint: Standard uPDEt consumes 7.69 GB on DrivAerML, requiring roughly half the memory of SMART (14.97 GB) and AB-UPT (12.70 GB).
- Activation Checkpointing: Enabling activation checkpointing further compresses uPDEt's memory footprint by over $3\times$ (down to 2.49 GB on DrivAerML and 0.82 GB on Crash) with a minor throughput penalty ($\sim 20\%$), enabling execution on consumer-grade hardware.

Table 30: Performance comparison on DrivAerML and SHIFT-Crash benchmarks using an NVIDIA L40S GPU.

	DrivAerML			SHIFT-Crash		
Model	Speed (samples/s)	Memory (GB)	GFLOPs	Speed (samples/s)	Memory (GB)	GFLOPs
PTV3	6.18	3.01	802	12.90	3.98	401
Erwin	8.60	1.96	320	25.61	1.96	160
Transolver++	1.94	3.60	3512	3.89	3.60	1755
GeoTransolver	0.51	8.86	4998	0.92	14.67	2497
AB-UPT	0.50	12.70	24622	1.13	6.73	9599
SMART	1.07	14.97	13265	1.76	9.65	8242
SMART (small)	2.32	7.04	5837	4.29	4.78	3364
uPDEt	3.92	7.69	1698	14.11	2.83	643
uPDEt (checkpointing)	3.15	2.49	2325	11.38	0.82	901

High-Resolution Point Cloud Scaling In Table 31, we evaluate the scaling behavior of smaller variants of the models (uPDEt, SMART, and Transolver++) on a NVIDIA H100 GPU across point resolutions ranging from $N = 2^{15}$ (32K) up to $N = 2^{23}$ (16.8M) points. Transolver++ exhausts GPU memory (OOM) at $N = 2^{22}$, exhibiting steep growth in memory allocation. While SMART maintains linear memory scaling, its per-step runtime is $3\times$ to $5\times$ higher than uPDEt across all evaluated point counts, ultimately exceeding available memory at 2^{23} points. uPDEt exhibits linear time and memory complexity, scaling effortlessly to 16.8M points without out-of-memory errors.

	uPDEt (XS)		SMART (XS)		Transolver++ (XS)	
# Points	Time	Memory	Time	Memory	Time	Memory
2^{15} (32K)	0.024	0.45	0.128	0.55	0.032	0.87
2^{16}	0.047	0.84	0.205	0.99	0.063	1.69
2^{17}	0.095	1.61	0.346	1.86	0.124	3.32
2^{18}	0.192	3.16	0.642	3.62	0.247	6.59
2^{19}	0.409	6.24	1.234	7.14	0.494	13.12
2^{20} (2M)	0.823	12.42	2.406	14.17	0.994	26.19
2^{21}	1.791	24.77	4.743	28.23	2.008	52.32
2^{22}	3.872	48.47	9.433	56.36	–	–
2^{23} (16M)	9.238	95.47	–	–	–	–

Table 31: Scaling comparison of Time and Memory across different point counts for uPDEt, SMART, and Transolver++ on H100.

G Aerodynamic Force and Coefficient Integration

In industrial engineering workflows, accurately estimating aggregated aerodynamic loads acting on a vehicle or lifting surface is often the primary objective. The total net aerodynamic force vector $\mathbf{F}$ is governed by the joint action of normal pressure forces and tangential wall shear stresses acting across the boundary manifold S. We compute the net force via surface integration, discretized using a midpoint quadrature rule over the triangulated boundary mesh:

$$\mathbf{F} = \int_S (-p\mathbf{n} + \boldsymbol{\tau})\, \mathrm{d}S \approx \sum_{i=1}^{M} (-p_i \mathbf{n}_i + \boldsymbol{\tau}_i)\, S_i, \tag{8}$$

where M denotes the total number of surface boundary cells, $p_i \in \mathbb{R}$ is the predicted static pressure, $\boldsymbol{\tau}_i \in \mathbb{R}^3$ represents the wall shear stress vector at the centroid of cell i, $\mathbf{n}_i$ is the outward unit normal, and S_i is the corresponding face area. The scalar drag force F_D and lift force F_L are subsequently obtained by projecting $\mathbf{F}$ along the free-stream velocity vector and its orthogonal cross-flow direction, respectively, accounting for the angle of attack α in aerodynamic coordinate transformations.

Figures 7 and 8 evaluate the correlation between ground-truth and uPDEt-predicted drag (C_D) and lift (C_L) coefficients on the DrivAerML and SHIFT-Wing test suites. Our model achieves high predictive fidelity, with determination coefficients (R^2) matching or exceeding established benchmarks in the literature (e.g., on DrivAerML, AB-UPT reports R^2 values of 0.975 for C_D and 0.963 for C_L, while Transolver3 (Zhou et al., 2026) reports 0.985 and 0.972).

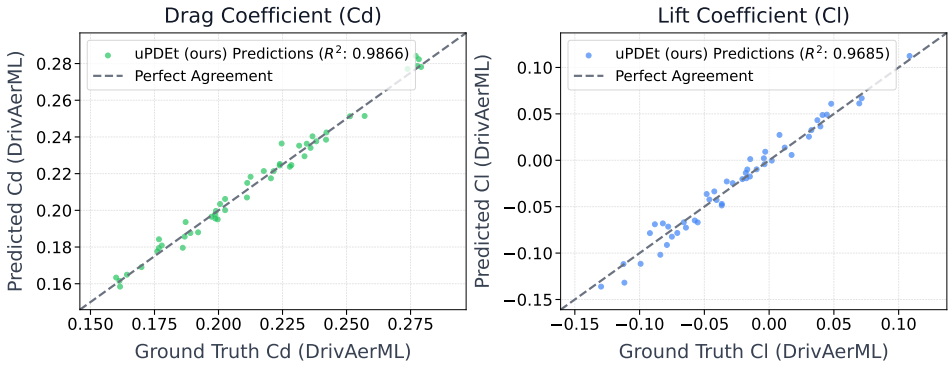


Figure 7: Parity plots and correlation metrics (R^2) for integrated drag (C_D) and lift (C_L) coefficients predicted by uPDEt on the DrivAerML test set.

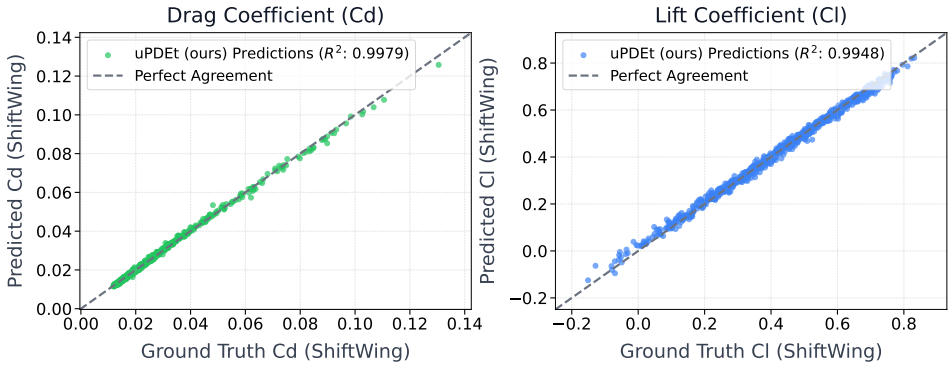


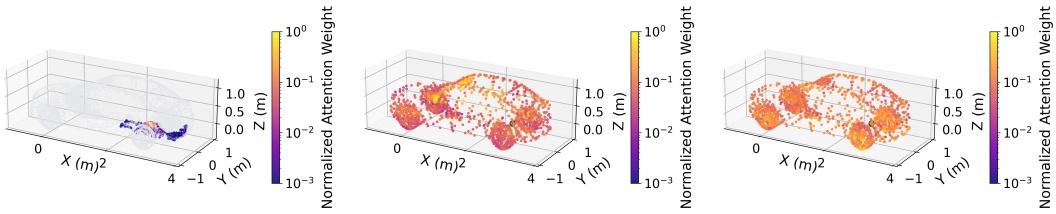
Figure 8: Parity plots and correlation metrics (R^2) for integrated aerodynamic coefficients (C_D, C_L) predicted by uPDEt across the SHIFT-Wing test set.

H Model Interpretability and Attention Analysis

To understand how uPDEt captures multiscale fluid dynamics, we inspect the learned self-attention maps across different hierarchical stages of the network. Figure 9 visualizes the spatial attention distribution conditioned on a query point located next to one of the rear wheels. The learned attention mechanisms reveal a clear multiscale decomposition corresponding to physical flow regimes:

- **Fine-Scale Local Attention (Level 0):** In the initial hierarchy level (Figure 9a), attention weights are strictly localized to the immediate geometric neighborhood. The query token focuses predominantly on adjoining boundary surface elements, effectively capturing high-gradient boundary layer shear and local curvature variations without being contaminated by distant flow fields.
- **Coarse-Scale Bottleneck Attention (Level 2):** At the deepest latent resolution, the receptive field expands across the entire vehicle body to process global aerodynamic interactions. Although the attention weights are distributed broadly, we observe that the first block of this level shows higher attention to the mirrors (Figure 9b) and the second to the rear wheels (Figure 9c).

This progression demonstrates that uPDEt inherently decouples micro-scale boundary-layer physics in its outer branches from macro-scale interactions within its multiscale latent bottleneck.



(a) Level 0, Block 0 (Local) (b) Level 2, Block 0 (Global) (c) Level 2, Block 1 (Global)

Figure 9: Visualizing learned spatial attention maps conditioned on a query point on the rear chassis (green dot). (a) Initial fine-scale level exhibits strict local boundary layer support. (b, c) Coarse bottleneck levels attend globally across the body, selectively weighting critical upstream geometry (e.g., side mirrors) and downstream separation zones (e.g., rear wheel wells).

I Results with Standard Deviations

To provide a better assessment of model accuracy beyond aggregated summary metrics, Tables 32, 33, and 34 report the per-channel relative L_2 error along with sample-wise standard deviations across the entire hold-out test sets.

Table 32: DrivAerML test relative L_2 error (%) with standard deviation across test samples.

Model	Surface				Volume			
	p	τ_x	τ_y	τ_z	p	u_x	u_y	u_z
PTV3	20.38 ± 1.59	26.86 ± 1.42	38.01 ± 2.00	40.06 ± 3.86	29.90 ± 4.44	32.39 ± 3.35	59.01 ± 6.59	55.93 ± 6.61
Erwin	18.66 ± 1.19	23.92 ± 1.09	33.47 ± 1.34	38.01 ± 3.12	16.50 ± 1.09	18.48 ± 0.79	33.99 ± 1.26	37.69 ± 3.72
Transolver++	6.40 ± 0.70	9.70 ± 0.89	13.77 ± 1.00	15.42 ± 2.00	7.63 ± 0.63	9.15 ± 0.60	16.43 ± 1.05	18.94 ± 2.01
GeoTransolver	5.76 ± 0.63	8.86 ± 0.84	12.64 ± 0.92	14.05 ± 1.72	6.28 ± 0.61	7.74 ± 0.58	13.95 ± 0.99	15.61 ± 1.89
AB-UPT	6.93 ± 0.65	9.22 ± 0.98	11.97 ± 1.07	14.93 ± 1.83	5.70 ± 0.66	6.28 ± 0.61	10.33 ± 1.06	12.99 ± 2.02
SMART	4.41 ± 0.69	7.28 ± 1.22	9.06 ± 1.19	10.98 ± 2.02	3.96 ± 0.67	4.88 ± 0.79	8.77 ± 1.35	10.41 ± 2.09
uPDEt	5.72 ± 0.71	8.85 ± 0.92	12.05 ± 1.09	14.06 ± 2.05	4.32 ± 0.64	5.07 ± 0.73	9.06 ± 1.34	11.05 ± 2.11

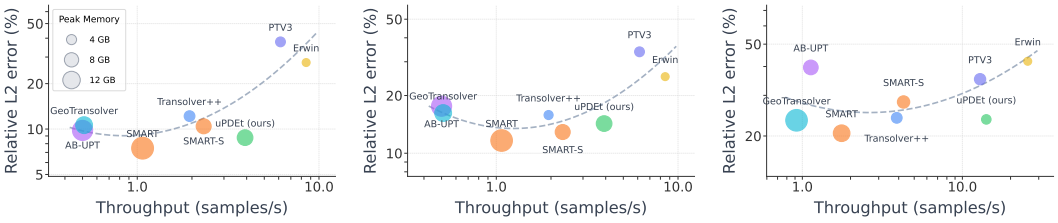
Table 33: SHIFT-Wing test relative L_2 error (%) with standard deviation across test samples.

Model	Surface				Volume			
	p	τ_x	τ_y	τ_z	p	u_x	u_y	u_z
PTV3	2.77 ± 1.13	20.35 ± 8.61	32.35 ± 9.15	27.99 ± 12.19	4.75 ± 2.65	48.08 ± 1.96	67.85 ± 5.17	66.37 ± 5.13
Erwin	1.48 ± 0.66	10.04 ± 1.83	18.49 ± 3.02	16.07 ± 3.25	3.28 ± 1.88	42.18 ± 1.52	56.17 ± 2.26	53.04 ± 1.83
Transolver++	0.66 ± 0.31	8.08 ± 0.75	11.70 ± 1.55	10.22 ± 2.07	0.90 ± 0.58	28.24 ± 1.27	33.29 ± 1.60	33.73 ± 1.78
GeoTransolver	0.64 ± 0.30	8.01 ± 0.76	11.18 ± 1.58	10.15 ± 2.01	0.42 ± 0.41	28.55 ± 1.53	34.63 ± 2.34	35.98 ± 2.32
AB-UPT	0.89 ± 1.03	11.98 ± 9.56	14.67 ± 9.17	12.72 ± 8.32	1.04 ± 0.79	28.90 ± 3.70	34.91 ± 4.16	35.60 ± 4.65
SMART	0.72 ± 0.65	8.18 ± 2.03	11.84 ± 4.26	11.06 ± 4.97	0.96 ± 0.77	8.54 ± 1.01	14.23 ± 4.32	13.69 ± 3.38
uPDEt	0.62 ± 0.33	8.10 ± 0.91	11.57 ± 1.91	10.18 ± 2.12	0.83 ± 0.59	23.83 ± 1.02	28.72 ± 1.61	31.22 ± 1.64

Table 34: SHIFT-Crash test relative L_2 error (%) with standard deviation across test samples.

Model	Surface			
	d_x	d_y	d_z	τ
PTV3	4.77 ± 0.93	50.73 ± 26.76	23.97 ± 6.22	61.53 ± 1.11
Erwin	9.95 ± 3.68	58.87 ± 26.24	35.09 ± 13.80	64.69 ± 1.17
Transolver++	2.52 ± 0.87	38.32 ± 20.35	13.95 ± 4.87	41.07 ± 0.88
GeoTransolver	2.45 ± 0.88	37.10 ± 20.01	13.53 ± 5.14	40.47 ± 0.86
AB-UPT	9.04 ± 2.57	55.28 ± 27.77	38.45 ± 9.02	55.81 ± 3.02
SMART	2.35 ± 1.06	35.39 ± 21.40	13.07 ± 5.28	31.44 ± 1.33
uPDEt	2.42 ± 0.88	37.46 ± 19.92	13.83 ± 5.19	40.69 ± 0.70

J VISUAL RESULTS



(a) DrivAerML (b) SHIFT-Wing (c) SHIFT-Crash

Figure 10: Trade-off between accuracy and efficiency for steady experiments.

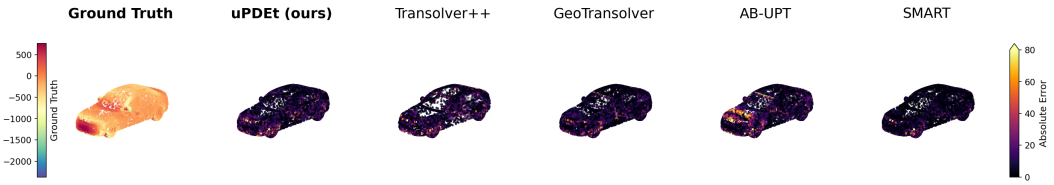


Figure 11: DrivAerML surface pressure on test sample 1.

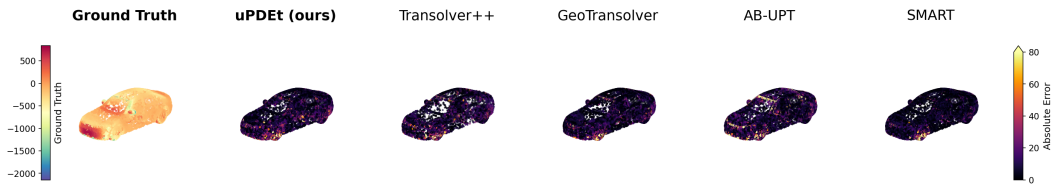


Figure 12: DrivAerML surface pressure on test sample 2.

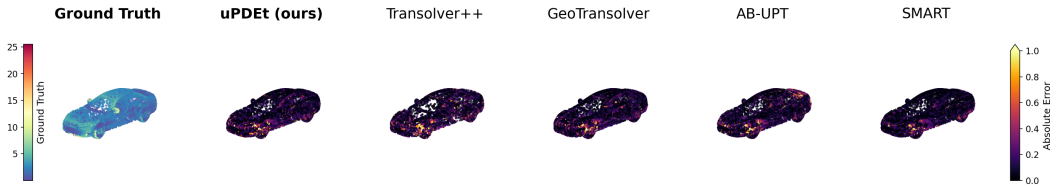


Figure 13: DrivAerML surface wss on test sample 1.

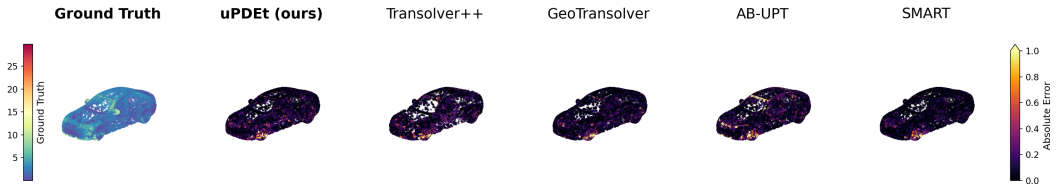


Figure 14: DrivAerML surface wss on test sample 2.

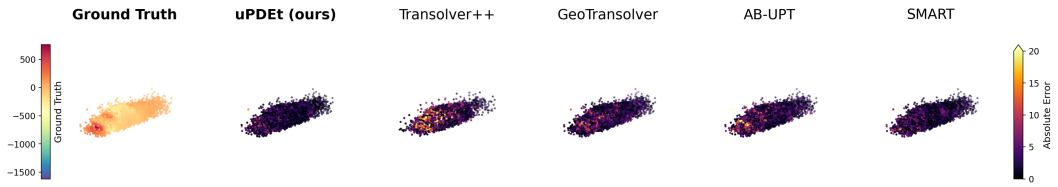


Figure 15: DrivAerML volume pressure on test sample 1.

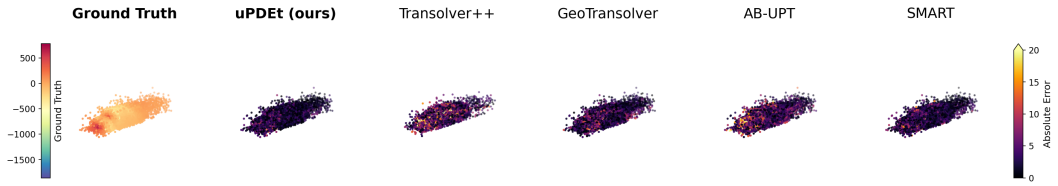


Figure 16: DrivAerML volume pressure on test sample 2.

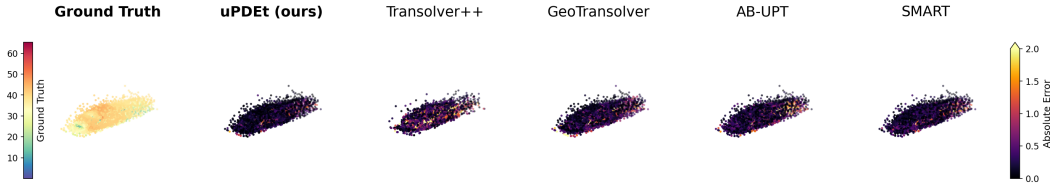


Figure 17: DrivAerML volume velocity on test sample 1.

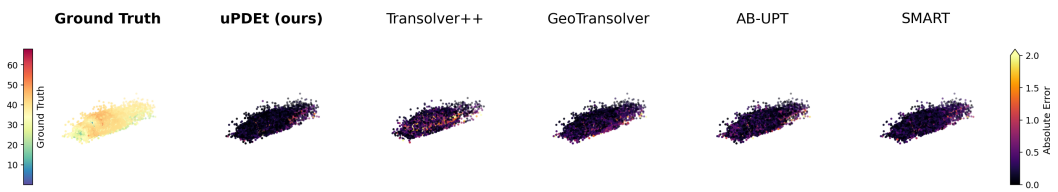


Figure 18: DrivAerML volume velocity on test sample 2.

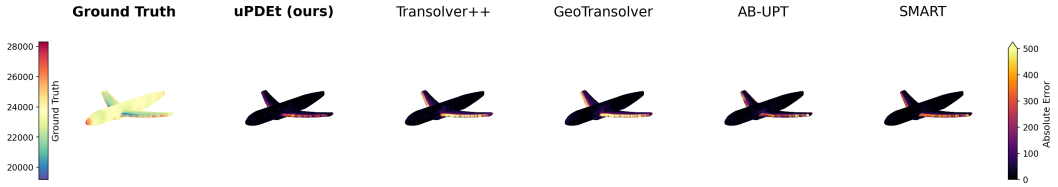


Figure 19: SHIFT-Wing surface pressure on test sample 1.

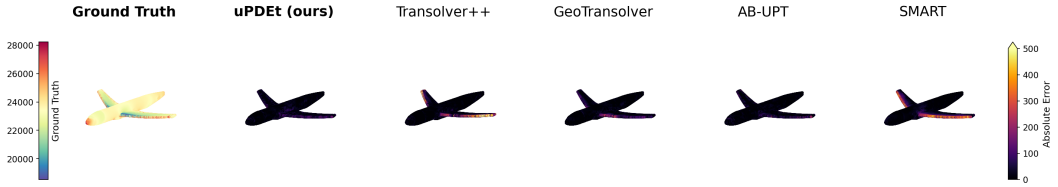


Figure 20: SHIFT-Wing surface pressure on test sample 2.

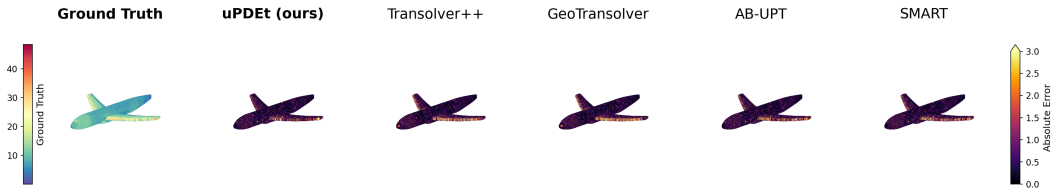


Figure 21: SHIFT-Wing surface wss on test sample 1.

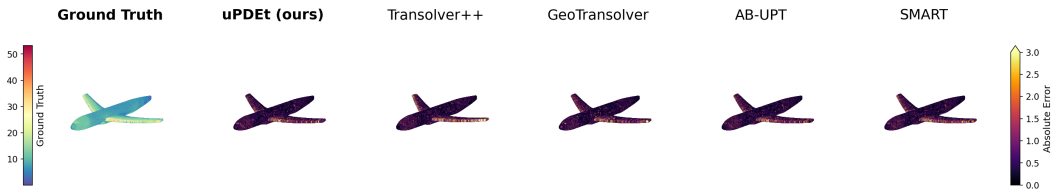


Figure 22: SHIFT-Wing surface wss on test sample 2.

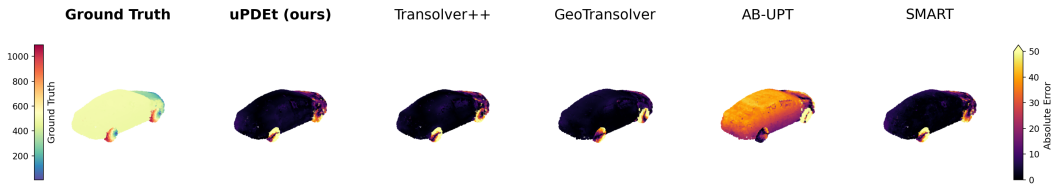


Figure 23: SHIFT-Crash deformation on test sample 1.

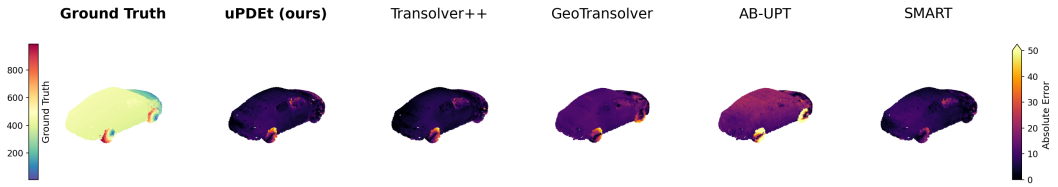


Figure 24: SHIFT-Crash deformation on test sample 2.

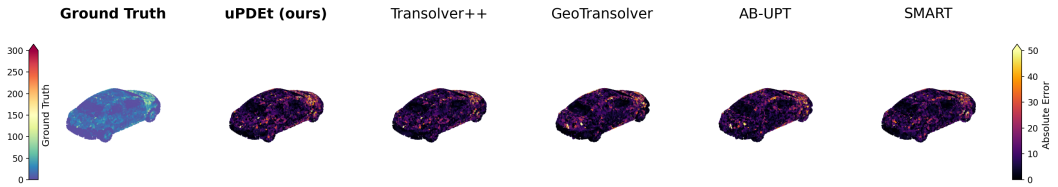


Figure 25: SHIFT-Crash stress on test sample 1.

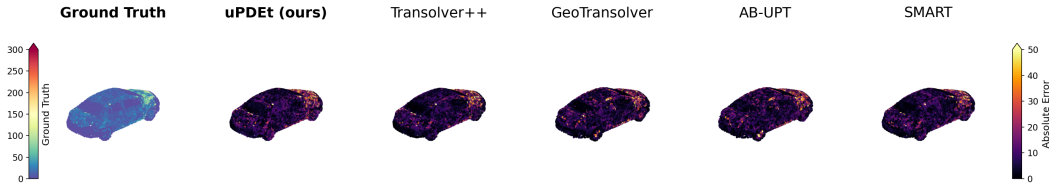


Figure 26: SHIFT-Crash stress on test sample 2.

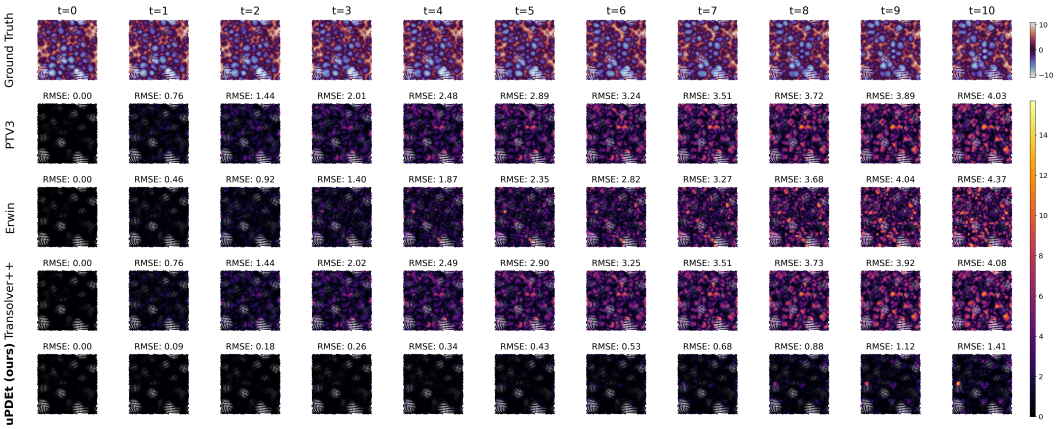


Figure 27: KS rollout plot on test sample 1.

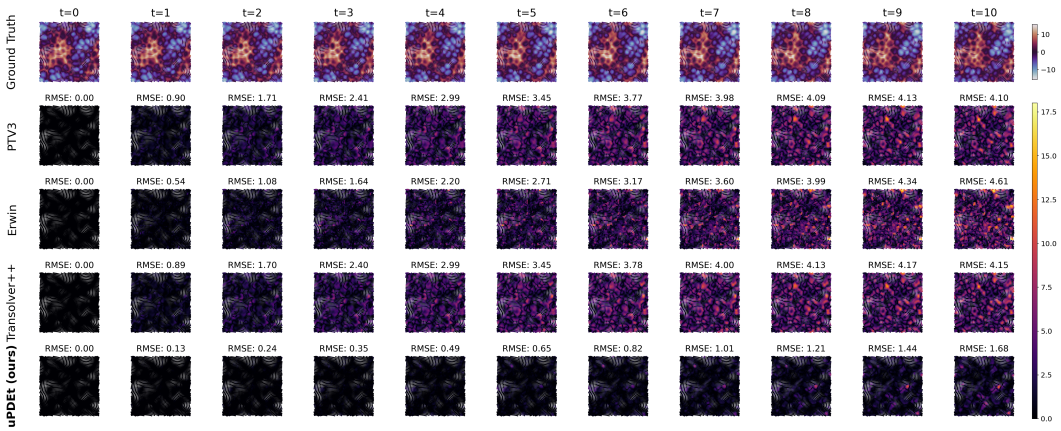


Figure 28: KS rollout plot on test sample 2.

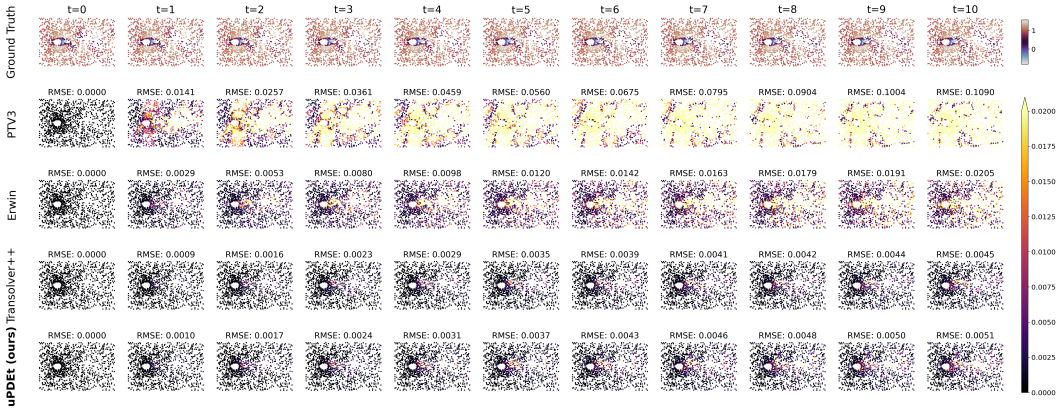


Figure 29: Ellipse rollout plot on test sample 1.

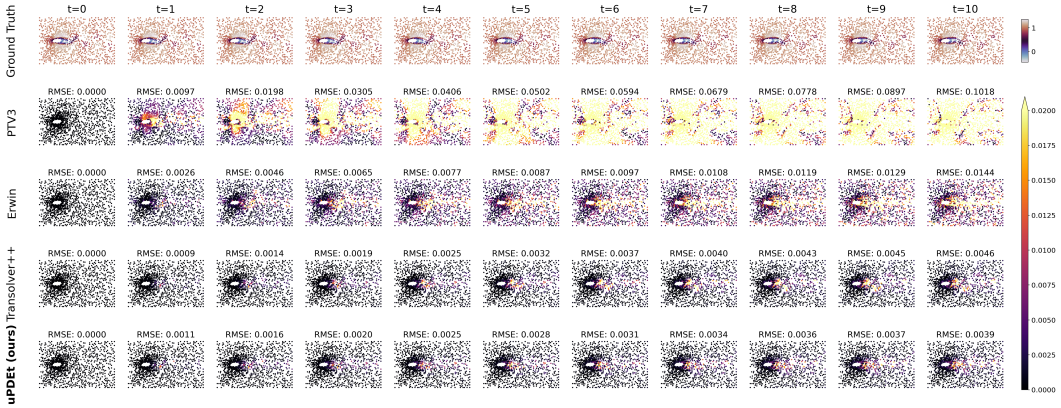


Figure 30: Ellipse rollout plot on test sample 2.

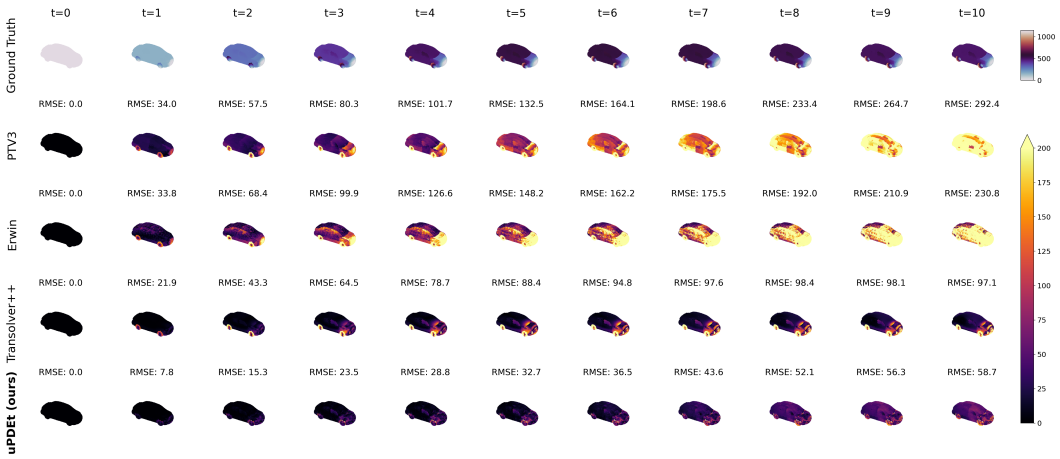


Figure 31: SHIFT-Crash rollout plot on test sample 1.

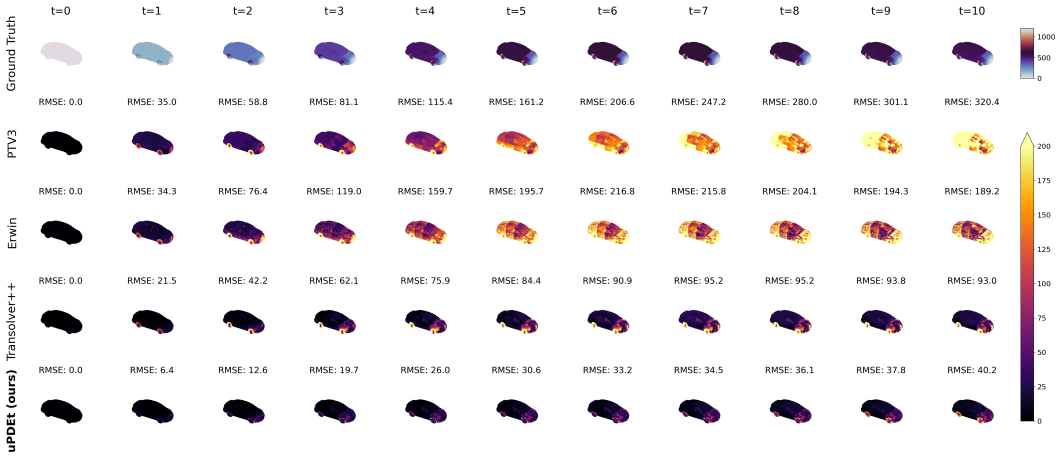


Figure 32: SHIFT-Crash rollout plot on test sample 2.