CAD-NATIVE TRANSFORMER OPERATORS FOR AI-AIDED ENGINEERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern engineering systems, from automobiles to aircraft, are designed by using precise, continuous parametric computer-aided design (CAD) models. Evaluating design changes through numerical simulation requires meshing the continuous geometry, a computationally expensive and often brittle process that can require manual intervention and replaces the continuous representation with a discrete approximation. Most neural surrogates accelerate the simulation, but inherit this representation gap by relying on meshes, point clouds, voxels, or other sampled approximations of geometry. We introduce CANTO, a transformer neural operator that maps directly from continuous CAD geometry to physical fields, without meshing the input geometry. We develop a theoretical framework for learning operators from geometric manifolds to function spaces of physical fields, representing geometry through sequences of parametric patches. CANTO instantiates this framework by directly tokenizing non-uniform rational B-spline (NURBS) patches from their control points, knot vectors, and weights, and predicts continuous surface and volume fields at arbitrary query locations. We evaluate CANTO on four automotive and aircraft aerodynamics industry benchmarks: AhmedML, WindsorML, DrivAerML, and HiLiftAeroML. CANTO achieves state-of-the-art accuracy on most evaluated surface and volume prediction tasks, including a 19.8% reduction in surface-pressure relative L_2 error compared with AB-UPT on HiLiftAeroML. Differentiability with respect to CAD parameters further enables gradient-based inverse design of designs. On AhmedML, CANTO identifies designs with 4.4 to 20.4% lower drag than the best dataset designs satisfying the same volume and lift constraints, with the improvements verified using the same CFD setup used to generate the original dataset.

1 INTRODUCTION

Modern engineering design is an iterative loop: designers modify parametric computer-aided design (CAD) models, and numerical simulations evaluate their performance. In vehicle and aircraft aerodynamics, this simulation is computational fluid dynamics (CFD), which solves the governing Partial Differential Equations (PDEs) of fluid flow to predict velocity, pressure, and wall-shear stress and combines them into integrated quantities such as drag and lift. The geometry used by designers is exact and continuous: CAD models represent surfaces parametrically, often using Non-Uniform Rational B-Splines (NURBS). A NURBS surface is defined by a compact set of control points, rational weights, and knot vectors, which together specify a smooth map from a low-dimensional parameter domain to physical space.

The representation gap. CFD solvers do not typically operate directly on parametric CAD models. Instead, the geometry is first discretized into a tessellated surface mesh and a volumetric mesh, often comprising 10^7 to 10^8 cells, on which the solver then solves the governing PDEs using finite-volume, finite-element, or related numerical schemes. For industrial geometries, mesh generation is often the most brittle and labor-intensive stage of the analysis pipeline, requiring hours of manual cleaning, repair, and refinement. The mesh, not the CAD model, then becomes the input representation for simulation and for most learned surrogates. We refer to the gap between the exact CAD design and its sampled, mesh-based proxy as the *representation gap* between design and analysis. This gap keeps design and analysis separated and bottlenecks the iterative engineering cycle.

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.

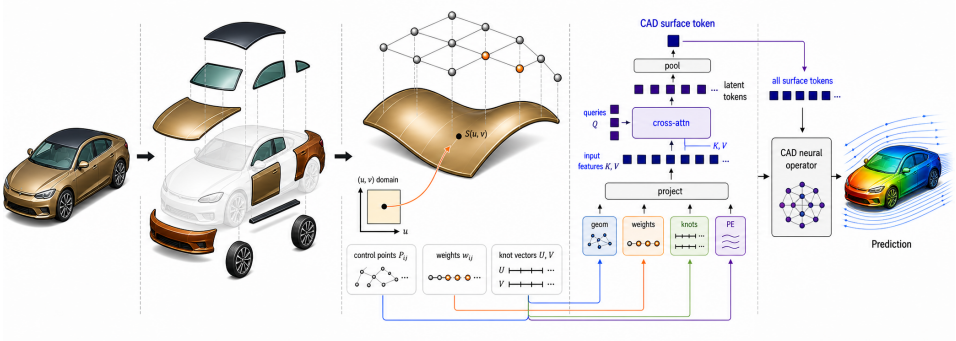


Figure 1: **Overview of our approach.** CANTO obviates the need for meshing step used by traditional AI surrogate pipelines by instead directly tokenizing continuous CAD NURBS geometries. First, the original CAD geometry is expressed as a sequence of NURBS surfaces, before being encoded by their control points, knot vectors, and weights. These parametrized NURBS surfaces are then tokenized with a Perceiver-based encoder and used to condition our neural operator to predict the target physical fields.

AI surrogates inherit the gap. Recent neural surrogates can accelerate CFD evaluation from days to milliseconds, but they largely use the same kind of sampled geometric discretizations as classical solvers. Graph neural operators and radius-graph models (Li et al., 2020; 2023c), point-cloud encoders (Alkin et al., 2025; Wu et al., 2024; Hao et al., 2023), and voxel-based networks all replace exact NURBS surfaces with a discretized mesh or point cloud before inference. As a result, they accelerate analysis without closing the representation gap: the model never sees the design representation used by the engineer, and its predictions remain conditioned on a discretized approximation of the geometry. We discuss these methods and their resolution behavior in Section 2.

Our approach. We propose the CAD-Native Transformer Operator (CANTO), a transformer neural operator that operates directly on the native parametric NURBS representation of CAD geometry and predicts physical fields as continuous functions queryable at arbitrary spatial locations. Instead of meshing, voxelizing, or spatially sampling the input geometry, CANTO encodes each untrimmed NURBS surface into a single latent token computed from its spline definition.

A transformer then combines these patch tokens into a global geometry context, which conditions a transformer backbone on query points coordinate for surface and volume field prediction, as well as engineering quantities. This geometry encoding is independent of query resolution: it depends only on the compact CAD description, instead of a discretized mesh approximation with millions of cells or points. In summary, we make the following contributions:

- **Framework.** We formalize CAD-to-physics surrogate modeling as a map from a class of compact embedded C^1-manifolds, represented through sequences of oriented boundary patch parametrizations, to a Banach function space of physical fields, providing a principled foundation for tokenizing parametric geometries.
- **Architecture.** We introduce a NURBS-native tokenization that constructs geometry and surface tokens directly from control points, weights, and knot vectors. Integrated into the transformer backbone of CANTO, these tokens enable continuous field prediction without mesh, voxel, or point-cloud-based discretization of the input geometry.
- **Empirical results.** On the AhmedML, WindsorML automotive aerodynamics benchmarks, as well as the HiLiftAeroML aircraft aerodynamics benchmark, CANTO achieves state-of-the-art surface-field accuracy and is uniquely independent to the input and output discretization, maintaining its accuracy as the spatial-context budget is reduced by orders of magnitude where point-cloud baselines collapse. On DrivaerML, CANTO achieves very competitive results on surface variables, and state of the art accuracy for volume fields Moreover, we demonstrate that CANTO enables inverse design, identifying configurations that achieve lower drag than any design in the original AhmedML dataset under the constraints considered.

The remainder of this paper is organized as follows: we review related works in 2 and introduce relevant background material on NURBS in Section 3. We then introduce our proposed approach in Section 4 and validate it empirically in Section 5.

2 RELATED WORKS

Neural operators. Neural operators (Kovachki et al., 2023), such as the Fourier Neural Operator (Li et al., 2021) and DeepONet (Lu et al., 2021), learn maps between function spaces and have been applied broadly to systems governed by partial differential equations (PDEs) (Azizzadenesheli et al., 2024; Kovachki et al., 2024). For PDEs on complex geometries, however, the input domain is typically represented by a finite geometric proxy, such as a mesh, point cloud, voxel grid, signed-distance field, or graph. FNO and many of its variants operate on regular grids. Graph Neural Operators (Li et al., 2020) extend neural operators to irregular domains by aggregating information over neighborhoods defined by an ϵ-radius graph. GINO (Li et al., 2023c) uses GNO layers to map between point-cloud or signed-distance representations and a regular latent grid, where an FNO can be applied. These approaches improve geometric flexibility, but the geometry is still supplied through a sampled approximation rather than the native CAD representation.

Geometry-aware PDE and CFD surrogates. Recent work has extended neural operators and transformer-based surrogates to irregular domains and large-scale engineering flows. GNOT (Hao et al., 2023), OFormer (Li et al., 2023a), Transolver (Wu et al., 2024), GAOT (Wen et al., 2025), and GINOT (Liu et al., 2025) use graph, point-based, query-based, or attention mechanisms to model fields on complex domains. DoMINO (Ranade et al., 2025), AB-UPT (Alkin et al., 2025), FigConvNet (Choy et al., 2025), MNO (Wang et al., 2025), and Fusion-DeepONet (Peyvan et al., 2025) further demonstrate strong performance on engineering and CFD benchmarks, including automotive aerodynamics and high-speed flows.

These methods improve scalability and geometric flexibility, and some can predict fields at arbitrary query locations. However, their geometry encoders still operate on discretized geometric approximations: STL tessellations, mesh nodes, surface or volume point clouds, signed-distance samples, implicit grids, supernodes, or restricted parametric shape descriptions. Thus, even when the output field is queried continuously, the input geometry has already been discretized before reaching the model. In contrast, CANTO tokenizes the native CAD spline structure, replacing mesh- or point-based geometry encoders with NURBS patch tokens derived from its parametric representation.

CAD representations for geometry learning. A growing body of work tries to use CAD models for geometric tasks. UV-Net (Jayaraman et al., 2021) encodes boundary representations, DeepCAD (Wu et al., 2021) generates parametric CAD models, NURBS-Diff (Prasad et al., 2022) provides differentiable NURBS layers for surface fitting, NeuroNURBS (Fan et al., 2024) encodes NURBS surfaces directly from their parameters, and NURBSGen (Usama et al., 2026) fine-tunes Qwen3-4B (Yang et al., 2025) using LoRA (Hu et al., 2021) to map text descriptions to JSON sequences encoding NURBS surface parameters, which are subsequently converted into BRep geometry. These methods preserve CAD structure, but primarily target geometric representation learning, generation, or reconstruction, rather than physical-field prediction.

Our approach is also related in spirit to isogeometric analysis (IGA) (Hughes et al., 2005), which uses spline representations for both geometry and numerical analysis. However, IGA remains a numerical PDE discretization method: like traditional finite-element or finite-volume approaches, it solves the governing equations for each new configuration, using spline basis functions, typically NURBS, to represent the geometry and approximate the solution. In contrast, CANTO uses a parametric NURBS representation of the CAD geometry as the input space of a learned neural operator. Rather than performing a new numerical solve for each geometry, CANTO learns the solution operator from high-fidelity simulation data and directly predicts continuous physical fields and quantities of interest for unseen configurations at inference time.

3 PRELIMINARIES

NURBS parametrizations. The NURBS parameterization of curves and surfaces is based on B-spline basis functions constructed via the Cox–de Boor recursion De Boor (1977). They are defined via a *knot vector*, which is a non-decreasing sequence of real numbers

$$U = (u_0, u_1, \ldots, u_{m+p+1}), \qquad u_k \leq u_{k+1},$$

where p denotes the polynomial degree and $(m+1)$ the number of basis functions. Given a knot vector U, the B-spline basis functions $N_{k,p}^U$ are defined recursively, as described in Appendix D.1. Given $n+1$ control points $P_i \in \mathbb{R}^3$, for each $0 \le i \le n$, and associated weights $w_i \ge 0$, a ***NURBS curve*** of degree p is defined as the mapping $C : [0,1] \to \mathbb{R}^3$,

$$C(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)\, w_i\, P_i}{\sum_{i=0}^{n} N_{i,p}(u)\, w_i}.$$

To define surfaces, we introduce an additional knot vector $V = (v_0, v_1, \ldots, v_{s+q+1})$ where $v_l \le v_{l+1}$ and its corresponding basis $N_{l,q}^V$. A surface can then be defined via the tensor product $N_{k,p}^U \otimes N_{l,q}^V$. In particular, given a tensor-product grid of control points $P \in \mathbb{R}^{m\times s\times 3}$ and non-negative weights $w \in \mathbb{R}^{m\times s}$, a ***NURBS surface*** of bi-degree (p,q) is given by the mapping $S : [0,1]^2 \to \mathbb{R}^3$,

$$S(u,v) = \frac{\sum_{k=0}^{m}\sum_{l=0}^{s} N_{k,p}^U(u)\, N_{l,q}^V(v)\, w_{kl}\, P_{kl}}{\sum_{k=0}^{m}\sum_{l=0}^{s} N_{k,p}^U(u)\, N_{l,q}^V(v)\, w_{kl}}. \tag{1}$$

We will only consider NURBS surfaces that are non-intersecting, making S injective. The unit surface normal is

$$\nu(u,v) = \frac{\partial_u S(u,v) \times \partial_v S(u,v)}{\|\partial_u S(u,v) \times \partial_v S(u,v)\|}, \tag{2}$$

with orientation induced by the surface parametrization, and hence by the ordering of the control points P_{kl} along the u- and v-directions. Reversing either parametric direction reverses ν.

Trimmed NURBS surfaces. Trimming enables complex surface boundaries and holes to be represented while leaving the underlying NURBS parameterization unchanged. A ***trimmed NURBS surface*** restricts the parametric domain of a NURBS surface using a set of trimming curves $\mathcal{T} = \{C_1, \ldots, C_N\}$. Specifically, while the underlying surface $S(u,v)$ is defined over a $[0,1]^2$, the trimming curves in $\mathcal{T}$, defined in the (u,v)-parameter space, delimit a restricted parameter domain $\mathcal{D} \subseteq [0,1]^2$. The image of $\mathcal{D}$ under S defines the corresponding *patch* $\Sigma := S(\mathcal{D}) \subset \mathbb{R}^3$, with restricted parameterization $\widetilde{S} := S|_{\mathcal{D}} : \mathcal{D} \to \Sigma$. We assume that $S_{\mathcal{D}}$ is a diffeomorphism onto its image, so that each point on the patch Σ is uniquely associated with a point in the trimmed parameter domain $\mathcal{D}$ (which is the case for regular, non-self intersecting surfaces). In this work, we *untrim* these surfaces by approximately decomposing them into collections of tensor-product NURBS patches, as detailed in Appendix D.2. Specifically, each surface Σ is decomposed as $\Sigma = \bigcup_{j=1}^{n_\Sigma} \Sigma_j$, where each patch Σ_j is diffeomorphic to $[0,1]^2$.

4 Methodology: CAD-Native Transformer Operators

Herein, we introduce our framework for tokenizing manifolds and apply it to parametrize CAD geometries and condition a neural transformer operator resulting in our proposed method, CANTO.

4.1 A framework for manifold operator learning

We present a general framework for the approximation of maps from a class of manifolds to a space of functions. To that end, let $\Omega \subseteq \mathbb{R}^d$ be an open set and denote by $\mathcal{M}$ the set of finite-dimensional topological manifolds. Let $\mathcal{U}(\Omega;\mathbb{R}^k)$ be a Banach function space of $\mathbb{R}^k$-valued functions with domain Ω. We consider the approximation of maps

$$F : \mathcal{M}' \subseteq \mathcal{M} \to \mathcal{U}(\Omega;\mathbb{R}^k),$$

where $\mathcal{M}'$ is a subclass of manifolds relevant to the specific problem. Such maps arise naturally in computational science whenever the solution of a physical system depends on its geometry, with examples including fluid dynamics, elasticity, and plasticity Li et al. (2020; 2023b;c). While our framework is general and can be applied to a variety of problems, we focus on a prototypical case arising in computational fluid dynamics, where the map F represents the domain-to-solution map of a PDE and the manifold represents a solid obstacle within the computational domain.

To make this precise, suppose that $\Omega \subset \mathbb{R}^d$ is a bounded, Lipschitz domain. We will consider the set of pairs (M, φ) where M is a compact d-dimensional C^1-manifold with boundary and $\varphi : M \to \Omega$ is a C^1-embedding such that $\varphi(M) \subset \Omega$ is a compact d-dimensional embedded submanifold with boundary and $\Omega \setminus \varphi(M)$ is a bounded, Lipschitz domain. We then define $\mathcal{M}'$ to be the set of equivalence classes of such pairs, where two pairs are identified if their embeddings have the same image. In particular, the two manifold, embedding pairs (M, φ) and (N, ψ) are equivalent if there exists a homeomorphism $h : M \to N$ such that $\varphi = \psi \circ h$. We can therefore identify a point $(M, \varphi) \in \mathcal{M}'$ with the bounded Lipschitz domain $\Omega_{M,\varphi} := \Omega \setminus \varphi(M)$. Given a PDE, we can then define F as the domain-to-solution map extended to Ω.

To illustrate this, we consider the Poisson equation, in particular,

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega_{M,\varphi}, \\ u &= 0 \qquad \text{in } \partial\Omega_{M,\varphi}, \end{aligned} \tag{3}$$

for some fixed $f \in H^{-1}(\Omega)$. Standard elliptic theory implies that (3) has a unique weak solution $u \in H_0^1(\Omega_{M,\varphi})$ Gilbarg & Trudinger (2001). Hence we can associate each point in $\mathcal{M}'$ with the PDE solution $u \in H_0^1(\Omega_{M,\varphi})$. We can then identify each solution with its zero extension $\tilde{u}$ to all of Ω so that $\tilde{u}|_{\Omega_{M,\varphi}} = u$ and $\tilde{u}|_{\Omega \setminus \Omega_{M,\varphi}} = 0$. Clearly $\tilde{u} \in H_0^1(\Omega)$, so we define $F : \mathcal{M}' \to H_0^1(\Omega)$ by $(M, \varphi) \mapsto \tilde{u}$. PDEs with more complex boundary conditions can be treated with Sobolev extension operators Stein (1970). In practice, we are usually only interested in u on $\Omega_{M,\varphi}$ and do not directly approximate the extension.

Numerically, it is usually much more efficient to work with the boundary of the solid rather than the solid volume. To do this, we further restrict $\mathcal{M}'$ to manifolds with boundary for which the oriented boundary determines a unique embedded solid up to the above equivalence relation. For each representative (M, φ), we associate the boundary parametrization $\varphi|_{\partial M} : \partial M \to \Omega$ together with the outward unit normal field $\nu : \partial M \to \mathbb{S}^{d-1}$ of the embedded solid $\varphi(M)$. Thus, on this restricted class, the equivalence class of (M, φ) may be represented by the boundary data $(\partial M, \varphi|_{\partial M}, \nu)$. The domain-to-solution map is unchanged, because this oriented boundary data determines the same embedded obstacle $\varphi(M)$, and hence the same PDE domain $\Omega \setminus \varphi(M)$.

4.2 Tokenized manifolds by patch embeddings

To construct an approximation architecture, we represent the embedded boundary by a finite collection of local parametrizations. Rather than requiring a single global parametrization of the considered geometry, we assume that, for each admissible embedded solid $\varphi(M)$, its boundary $\Gamma = \partial\varphi(M)$ admits a finite patch representation

$$\Gamma = \bigcup_{j=1}^{n} \Gamma_j,$$

where each patch Γ_j is parametrized by a C^1 map $\gamma_j : [0,1]^{d-1} \to \Gamma_j \subset \mathbb{R}^d$. Each γ_j is assumed to be a diffeomorphism from $(0,1)^{d-1}$ onto the interior of Γ_j, with suitable compatibility conditions on overlapping patch boundaries. For each patch, we denote by $\nu_j : [0,1]^{d-1} \to \mathbb{S}^{d-1}$ the associated outward normal field and consider the geometric input as a finite sequence of functions $\big((\gamma_1, \nu_1), \ldots, (\gamma_n, \nu_n)\big)$ where $\gamma_j \in C^1([0,1]^{d-1}; \mathbb{R}^d)$ and $\nu_j \in C([0,1]^{d-1}; \mathbb{S}^{d-1})$. The domain-to-solution map can then be regarded as a map from a sequence space of boundary parametrizations to the function space of solutions,

$$F : \mathcal{L} \subset \bigcup_{n \geq 1} \big(C^1([0,1]^{d-1}; \mathbb{R}^d) \times C([0,1]^{d-1}; \mathbb{S}^{d-1})\big)^n \to \mathcal{U}(\Omega; \mathbb{R}^k),$$

where $\mathcal{L}$ denotes the set of admissible sequences of boundary parametrizations.

The domain-to-solution map is well-defined on $\mathcal{M}'$ and passing to the patch parametrizations induces a pullback of this map to the sequence space $\mathcal{L}$ which we continue to denote by F. This representation is not injective in that different patch decompositions, orderings, or parametrizations may describe the same embedded solid and must therefore be assigned the same output. This non-uniqueness is a general feature of parametrically representing geometric data. We address it by (i) representing each γ_j using coefficients in a prescribed basis, as described in Section 4.3, and (ii) training with multiple patch decompositions of the same surface.

In the next subsection, we demonstrate a particular parameterization of each γ_j from which each surface normal may be computed directly. In this way, our choice in defining γ_j carries with it the orientation of the manifold. Let $\tilde{\gamma}_j$ denote these parameterizations. We then define our approximation architecture as a transformer $F_\theta : \mathcal{L}' \times \Omega \to \mathbb{R}^k$ trained to approximately satisfy

$$F_\theta\big((\tilde{\gamma}_1, \ldots, \tilde{\gamma}_n), x\big) = F\big((\gamma_1, \nu_1), \ldots, (\gamma_n, \nu_n)\big)(x) \qquad \forall\, x \in \Omega, \tag{4}$$

where $\mathcal{L}'$ denotes the set of such parameterizations. The main mechanism in our architecture can be viewed as a generalization of the functional patched attention formulated in Calvello et al. (2025). In the CAD setting considered in the next subsection, these representations are given directly by the NURBS descriptions of the surface patches.

4.3 Parameterizing CAD Geometries

We focus on $d = 3$ and consider CAD boundaries represented by trimmed NURBS surfaces. Each trimmed surface is decomposed into a finite collection of untrimmed NURBS patches. We may therefore write the boundary of the embedded solid as

$$\Gamma = \bigcup_{j=1}^{n} \Sigma_j,$$

where each patch Σ_j is parameterized by a NURBS map $S_j : [0,1]^2 \to \Sigma_j$. This is the patch representation introduced in Section 4.2, with the abstract patch maps γ_j instantiated by $\gamma_j = S_j$.

Furthermore, the outward normal field associated with each patch is determined by its NURBS parameterization (2), with the orientation chosen consistently with the outward orientation of Γ. Thus, both γ_j and ν_j are determined by the NURBS description of S_j.

In particular, the finite-dimensional patch representation is given in the CAD setting by

$$\widetilde{\gamma}_j = \left(U^{(j)}, V^{(j)}, P^{(j)}, w^{(j)}\right),$$

where $U^{(j)}$, $V^{(j)}$, $P^{(j)}$ and $w^{(j)}$ denote the knot vectors, control points and weights defining S_j. Consequently, the sequence $(\widetilde{\gamma}_1, \ldots, \widetilde{\gamma}_n)$ provides the finite-dimensional representation of the CAD boundary supplied to the approximation architecture.

4.4 NURBS tokenization and operator architecture

CANTO realizes the operator in Eq. (4) by directly tokenizing the parametric NURBS representation of CAD geometry. The resulting geometry context conditions a transformer backbone that predicts surface and volume fields at arbitrary query locations, without meshing the input geometry.

Tokenizing NURBS surfaces. A key challenge in tokenizing CAD geometries is that each NURBS surface is parameterized by a control net and knot vectors whose sizes and configurations vary across surfaces. Directly treating all NURBS parameters as geometry tokens would result in a variable and potentially large number of tokens per surface. We therefore propose an attention-based geometry encoder which, for each surface, takes the set of parameters in $\tilde{\gamma}_j$ and maps them to a fixed d-dimensional representation.

Applying this to all surfaces gives us a $\mathbb{R}^{n \times d}$ representation of the geometry, noting that n can vary across geometries. This representation is the tokenized version of our input manifold as defined via the NURBS parameterization.

For the j-th surface, we first concatenate the control-point coordinates $\mathbf{P}^{(j)} \in \mathbb{R}^{m_j \times s_j \times 3}$ with the corresponding NURBS weights $\mathbf{w}^{(j)} \in \mathbb{R}^{m_j \times s_j}$ to form the weighted control-point tensor $\mathbf{P}_w^{(j)} \in \mathbb{R}^{m_j \times s_j \times 4}$. Each control point is additionally augmented with its two-dimensional control-net indices (i, j), providing its position within the tensor-product control grid.

For the two knot vectors in the U- and V-directions, we augment each knot value with its position in the corresponding knot sequence,

$$U^{(j)} \mapsto \left\{\left(U_k^{(j)}, 1/k\right)\right\}_{k=1}^{m_j+p_j+1}, \qquad V^{(j)} \mapsto \left\{\left(V_\ell^{(j)}, 1/\ell\right)\right\}_{\ell=1}^{s_j+q_j+1},$$

using the inverse of the index to ensure the positional encoding is bounded.

The resulting sequences are processed by three individual Perceiver Pooling modules, which independently aggregate information from the weighted control points and the two knot vectors. The modules share the same general architecture but use component-specific hyperparameters, as detailed in Appendix A.

Each module first projects the control-point (resp. knot-vector) inputs to $d_{cp}^{\mathcal{P}}$-dimensional (resp. $d_{kv}^{\mathcal{P}}$-dimensional) embeddings, which are combined with a sinusoidal positional encoding. A Perceiver-style pooling block (Jaegle et al., 2021) then uses a fixed set of $q_{cp}^{\mathcal{P}}$ (resp. $q_{kv}^{\mathcal{P}}$) learned latent queries to cross-attend to the input sequence, followed by self-attention among the latent tokens. This cross-attention/self-attention pooling block is repeated for $K_{cp}^{\mathcal{P}}$ (resp. $K_{kv}^{\mathcal{P}}$) layers. The three pooled representations of the weighted control net and the two knot vectors are then concatenated, and the resulting feature vector is projected through an MLP to yield a fixed-dimensional patch token.

Global geometry context. The n patch tokens are jointly processed by K_g DiT-style self-attention blocks (Peebles & Xie, 2023), allowing each token to incorporate global context from the complete CAD geometry. The resulting geometry tokens form an $\mathbb{R}^{n\times d}$ representation, where n can vary across geometries.

Transformer backbone. For the solution branch of CANTO, we follow the Anchored-Branched Universal Physics Transformer (AB-UPT) Alkin et al. (2025), a Transformer-based multi-branch neural operator with separate branches for geometry, surface, and volume data. The NURBS-derived geometry tokens condition the surface and volume streams through cross-attention. We replace the spatially sampled surface anchors with surface tokens obtained directly from the NURBS representation (Appendix A.3) and encoded using the above Perceiver Embedding procedure. Volume anchors remain randomly sampled from the volume coordinate set, as in the standard AB-UPT. We use the same anchored neural field decoder, which predicts fields at arbitrary query locations without quadratic attention over query points. Further architectural details are provided in Appendix A.

5 Experiments

We evaluate CANTO against state-of-the-art methods on three industry standard benchmark datasets for automotive computational fluid dynamics: AhmedML Ashton et al. (2024b), WindsorML Ashton et al. (2024a) and DrivaerML Ashton et al. (2024c). We further validate the performance of CANTO on a recent, non-automotive dataset, HiLiftAeroML Ashton et al. (2026). It contains $1,800$ high-lift aircraft geometries and large-scale CFD simulations with approximately 200 million volume cells and 30 million surface cells per sample. We expand on each dataset in the Appendix C. Details regarding model architectures and training are provided in Section A.

5.1 Performance

Table 1 compares CANTO with state-of-the-art methods across four benchmark datasets. In particular, we trained AB-UPT on WindsorML and HiLiftAeroML, while the remaining baseline results are taken from Alkin et al. (2025). On AhmedML dataset, CANTO achieves the lowest error on three of four fields, and , especially, reduces the surface pressure error $\boldsymbol{p}_s$ of the state-of-the-art method by 11.6%. On DrivAerML, CANTO is on par with AB-UPT Alkin et al. (2025); it achieves the lowest volume velocity error $\boldsymbol{u}$ while AB-UPT marginally remains better for the surface pressure. Finally, on the HiLiftAeroML dataset, CANTO achieves the lowest error on all considered fields: it reduces the relative L_2 error by 19.8% for surface pressure, 2.3% for wall-shear stress and 16.9% for velocity. CANTO obtains nearly perfect accuracy in drag coefficients, with an R^2 of 0.99, improving over AB-UPT (0.96). On WindsorML dataset, CANTO attains the lowest error on every field, lowering the surface pressure error of AB-UPT by 5.5%.

The inference latency, peak memory requirements, and an end-to-end runtime comparison of our method and the traditional CFD framework used to generate these datasets are presented in Appendices B.2– B.4.

5.2 Ablation Study

Table 1: **Comparison of CANTO with prior methods.** We report the relative L2 errors (%) for the surface pressure p_s, volume velocity u, volume vorticity ω, and wall shear stress τ fields on the AhmedML, DrivAerML, WindsorML, and HiLiftAeroML datasets.

	AhmedML				DrivAerML			WindsorML			HiLiftAeroML		
Method	p_s	u	ω	τ	p_s	u	τ	p_s	u	τ	p_s	u	τ
PointNet	8.02	5.44	66.04	10.09	23.63	28.13	41.85	–	–	–	–	–	–
Graph U-Net	6.46	4.15	53.66	7.29	16.13	17.98	27.84	–	–	–	–	–	–
GINO	7.90	6.23	71.81	8.18	13.03	40.58	21.71	–	–	–	–	–	–
LNO	12.95	7.59	72.49	11.50	20.51	23.27	36.44	–	–	–	–	–	–
UPT	4.25	2.73	15.03	5.80	7.44	8.74	12.93	–	–	–	–	–	–
OFormer	4.12	3.63	15.06	4.60	4.85	6.64	8.92	–	–	–	–	–	–
Transolver	3.45	2.05	8.22	4.00	4.81	6.78	8.95	–	–	–	–	–	–
Transformer	3.41	2.09	6.76	4.03	4.35	6.21	8.26	–	–	–	–	–	–
AB-UPT	3.01	**1.90**	6.52	3.88	**3.82**	5.93	7.29	7.21	7.04	5.65	5.65	7.06	8.72
CANTO	**2.66**	2.04	**4.83**	**3.53**	3.97	**5.14**	**7.15**	**6.81**	**6.69**	**5.46**	**4.53**	**5.87**	**8.52**

Fixed-size vs. Variable-size NURBS features. First, we ablate the geometry encoder of CANTO with fixed-size and variable-size NURBS features. In the fixed-size variant, the encoder is a simple MLP, since the network sees the same shape of inputs. In the variable-size variant, a perceiver-based encoder is employed to handle the varying sizes of control points and knot vectors. Table 2 compares the error of these variants. They achieve comparable error across all four fields; the fixed-size variant performs slightly better on the surface pressure (p_s , while the variable-size variant performs better on the wall-shear stress τ) and on the volume fields (u and ω). While the fixed-size variant is simpler, the variable-size variant is practically preferred since real-world CAD surfaces are highly heterogeneous.

Table 2: **Evaluation of CANTO with fixed-size and variable-size NURBS features** on the AhmedML dataset.

Method	p_s	u	ω	τ
Fixed-size	**2.50**	2.27	5.80	3.54
Variable-size	2.66	**2.04**	**4.83**	**3.53**

Robustness to sampling. Figure 2 highlights a key advantage of CANTO: its geometry tokens are derived directly from the NURBS representation and are therefore independent of mesh resolution, anchor count, and sampling strategy. In contrast, AB-UPT exhibits strong sensitivity to its surface-anchor distribution, with different sampling strategies converging to distinct error levels even at large anchor budgets. CANTO avoids this sampling dependency entirely and achieves lower errors across all tested anchor budgets and quantities.

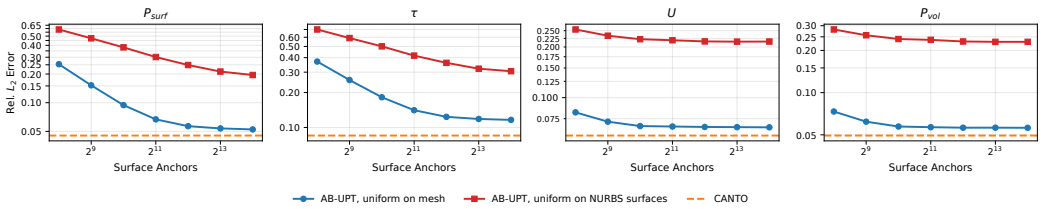


Figure 2: **Robustness to sampling procedure on the HiLiftAeroML dataset.** Here, *uniform-on-mesh* denotes anchors sampled uniformly from the surface mesh points, as in AB-UPT, while *uniform-on-NURBS surfaces* samples anchors uniformly in (u, v) parameter space on each NURBS surface, with the number of anchors allocated to each surface proportional to its area. .

5.3 CANTO-BASED INVERSE DESIGN

By operating directly on the parametric CAD representation, CANTO is differentiable with respect to the underlying geometry parameters, enabling gradient-based optimization for engineering design objectives. In automotive aerodynamics, for instance, reducing drag while controlling lift or down-

force is a central design objective, typically subject to geometric and other engineering constraints. In what follows, we demonstrate how CANTO can be used to formulate and efficiently solve such a constrained shape-optimization problem directly in the CAD parameter space.

Problem statement. We minimize aerodynamic drag over the six-parameter AhmedML body family, namely, *body length*, *body height*, *body width*, *front arc diameter*, *slant angle length* and *slant angle height*, subject to a lower bound on enclosed volume and an upper bound on lift,

$$\min_{x \in \mathcal{X}} D(x) \quad \text{s.t.} \quad V(x) \geq V_0, \qquad L(x) \leq \varepsilon, \tag{5}$$

where x collects the design parameters, $\mathcal{X}$ is the axis-aligned box spanned by the dataset described in Table 14, and $\varepsilon = 0.05$.

CANTO-based optimization. D and L are predicted by CANTO, trained exclusively on the AhmedML training split, with the remaining validation and test cases held out from training. Because both the mapping from the design variables $\mathbf{z}$ to the NURBS geometry and the CANTO surrogate are differentiable, the gradients $\nabla_{\mathbf{z}} D$, $\nabla_{\mathbf{z}} V$, and $\nabla_{\mathbf{z}} L$ can be obtained directly through automatic differentiation. We leverage these gradients to solve (5) using SLSQP (Kraft, 1994), supplying them explicitly to the optimizer. Starting from the lowest-drag feasible design in the entire dataset, the optimization then explores the continuous design space using the surrogate-derived gradient information.

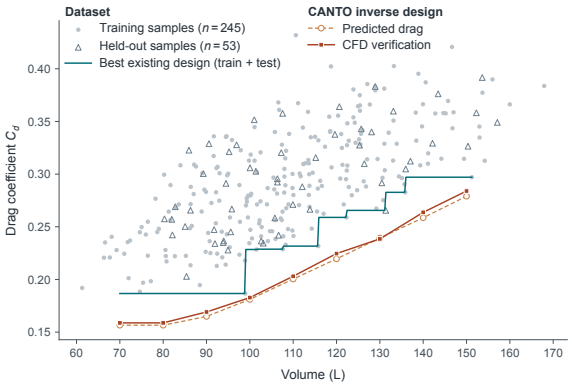


Figure 3: **Optimized Pareto front via inverse design.** Grey and blue markers show the ground-truth drag of training and held-out designs satisfying $C_l \leq 0.05$, while the blue staircase denotes the best dataset design satisfying each volume floor $V \geq V_0$. The dashed and solid red curves show the CANTO-predicted optima and their CFD validations, respectively. At every volume floor, CANTO identifies valid designs beyond the dataset benchmark front, with predictions closely matching CFD.

Results. The optimized designs achieve improvements across the entire front, with gains ranging from 4.4% to 20.4% relative to the best dataset designs at the corresponding volume thresholds, as shown in Figure 3. We further validate the optimized candidates using a traditional CFD solver (see Appendix E), which confirms the predicted performance gains. Across these designs, CANTO's predictions remain within 2.2% of the corresponding CFD values, demonstrating the surrogate's accuracy even for geometries discovered through optimization and not present in the original dataset. The CFD simulations also confirm that the lift constraint is satisfied at every volume threshold. Together, these results demonstrate that CANTO can successfully leverage its differentiable CAD representation to perform constrained design optimization and identify improved designs beyond those available in the original dataset.

6 Conclusion

Our proposed CAD-Native Transformer Operator closes the representation gap between CAD and learned physics. It takes parametric CAD directly as input to a neural transformer operator, without first reducing it to a mesh, point cloud, or implicit field. On AhmedML, WindsorML, DrivaerML and HiLiftAeroML, it reaches state-of-the-art accuracy on most evaluated surface-field and volume-field prediction tasks, with a 19.8% relative reduction in surface-pressure error on HiLiftAeroML. By learning directly from NURBS representations, CANTO eliminates the need for meshing at inference and remains robust to changes in discretization and sampling. CANTO is differentiable with respect to its NURBS inputs, enabling gradient-based shape optimization and inverse design directly in the CAD parameter space.

AI USE STATEMENT

In this work, we used generative AI tools for proofreading, language editing. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks or formulating mathematical claims. The authors reviewed all AI-assisted revisions and take full responsibility for the final content.

ETHICS STATEMENT

This work uses CAD geometries and computational fluid dynamics data and involves no human participants or personal data. We have identified no specific ethical concerns associated with this study.

REPRODUCIBILITY STATEMENT

The methodology is described in details in our manuscript. Specifically we introduce the details of the method in Section 4, with architecture and training details in Appendix A, dataset descriptions in Appendix C, CAD preprocessing in Appendix D.2, and inverse-design validation in Appendix E. We plan to publicly release the code upon acceptance.

REFERENCES

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. *arXiv preprint arXiv:2502.09692*, 2025.

Neil Ashton, Jordan B Angel, Aditya S Ghate, Gaetan K Kenway, Man L Wong, Cetin Kiris, Astrid Walle, Danielle C Maddix, and Gary Page. Windsorml: High-fidelity computational fluid dynamics dataset for automotive aerodynamics. *Advances in Neural Information Processing Systems*, 37:37823–37835, 2024a.

Neil Ashton, Danielle C Maddix, Samuel Gundry, and Parisa M Shabestari. Ahmedml: High-fidelity computational fluid dynamics dataset for incompressible, low-speed bluff body aerodynamics. *arXiv preprint arXiv:2407.20801*, 2024b.

Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, et al. Drivaerml: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics. *arXiv preprint arXiv:2408.11969*, 2024c.

Neil Ashton, Adam Clark, Liam Heidt, Christopher Ivey, Sanjeeb Bose, Rahul Agrawal, Konrad Goc, Rishi Ranade, Corey Adams, Peter Sharpe, Sheel Nidhan, Semit Akkurt, Daniel Leibovici, and Jean Kossaifi. Hiliftaeroml: High-fidelity computational fluid dynamics dataset for high-lift aircraft aerodynamics, 2026. URL https://arxiv.org/abs/2605.19565.

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024. doi: 10.1038/s42254-024-00712-5. URL https://doi.org/10.1038/s42254-024-00712-5.

Edoardo Calvello, Nikola B Kovachki, Matthew E Levine, and Andrew M Stuart. Continuum attention for neural operators. *Journal of Machine Learning Research*, 26(300):1–52, 2025.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

Chris Choy, Alexey Kamenev, Jean Kossaifi, Max Rietmann, Jan Kautz, and Kamyar Azizzadenesheli. Factorized implicit global convolution for automotive computational fluid dynamics prediction, 2025. URL https://arxiv.org/abs/2502.04317.

Carl De Boor. Package for calculating with b-splines. *SIAM Journal on Numerical Analysis*, 14(3): 441–472, 1977.

Jiajie Fan, Babak Gholami, Thomas Bäck, and Hao Wang. Neuronurbs: Learning efficient surface representations for 3d solids. *arXiv preprint arXiv:2411.10848*, 2024.

D. Gilbarg and N.S. Trudinger. *Elliptic Partial Differential Equations of Second Order*. Classics in Mathematics. Springer Berlin Heidelberg, 2001.

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 *International conference on machine learning*, pp. 12556–12569. PMLR, 2023.

Angelina I Heft, Thomas Indinger, and Nikolaus A Adams. Experimental and numerical investigation of the drivaer model. In *Fluids engineering division summer meeting*, volume 44755, pp. 41–51. American Society of Mechanical Engineers, 2012.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Thomas JR Hughes, John A Cottrell, and Yuri Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39-41):4135–4195, 2005.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11703–11712, 2021.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *JMLR*, 24(1), 2023.

Nikola B Kovachki, Samuel Lanthaler, and Andrew M Stuart. Operator learning: Algorithms and analysis. *Handbook of Numerical Analysis*, 25:419–467, 2024.

Dieter Kraft. Algorithm 733: Tomp–fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):262–281, 1994.

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.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

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, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023b.

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 *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NeurIPS '23, Red Hook, NY, USA, 2023c. Curran Associates Inc.

Qibang Liu, Weiheng Zhong, Hadi Meidani, Diab Abueidda, Seid Koric, and Philippe Geubelle. Geometry-informed neural operator transformer. *arXiv preprint arXiv:2504.19452*, 2025.

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, 2021. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5.

Fady Massarwi, Boris van Sosin, and Gershon Elber. Untrimming: Precise conversion of trimmed-surfaces to tensor-product surfaces. *Computers & Graphics*, 70:80–91, 2018.

Gary H Meisters. Polygons have ears. *The American Mathematical Monthly*, 82(6):648–651, 1975.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Ahmad Peyvan, Varun Kumar, and George Em Karniadakis. Fusion-deeponet: A data-efficient neural operator for geometry-dependent hypersonic and supersonic flows. *Journal of Computational Physics*, pp. 114432, 2025.

Anjana Deva Prasad, Aditya Balu, Harshil Shah, Soumik Sarkar, Chinmay Hegde, and Adarsh Krishnamurthy. Nurbs-diff: A differentiable programming module for nurbs. *Computer-Aided Design*, 146:103199, 2022.

Rishikesh Ranade, Mohammad Amin Nabian, Kaustubh Tangsali, Alexey Kamenev, Oliver Hennigh, Ram Cherukuri, and Sanjay Choudhry. Domino: A decomposable multi-scale iterative neural operator for modeling large scale engineering simulations, 2025. URL https://arxiv.org/abs/2501.13350.

E.M. Stein. *Singular Integrals and Differentiability Properties of Functions*. Princeton University Press, 1970.

Muhammad Usama, Mohammad Sadil Khan, Didier Stricker, and Muhammad Zeshan Afzal. Nurbgen: High-fidelity text-to-cad generation through llm-driven nurbs modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 9603–9611, 2026.

Qinxuan Wang, Chuang Wang, Mingyu Zhang, Jingwei Sun, Peipei Yang, Shuo Tang, and Shiming Xiang. Mno: Multiscale neural operator for computational fluid dynamics with 3d point cloud data. *arXiv preprint arXiv:2510.16071*, 2025.

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. *arXiv preprint arXiv:2505.18781*, 2025.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6772–6782, 2021.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

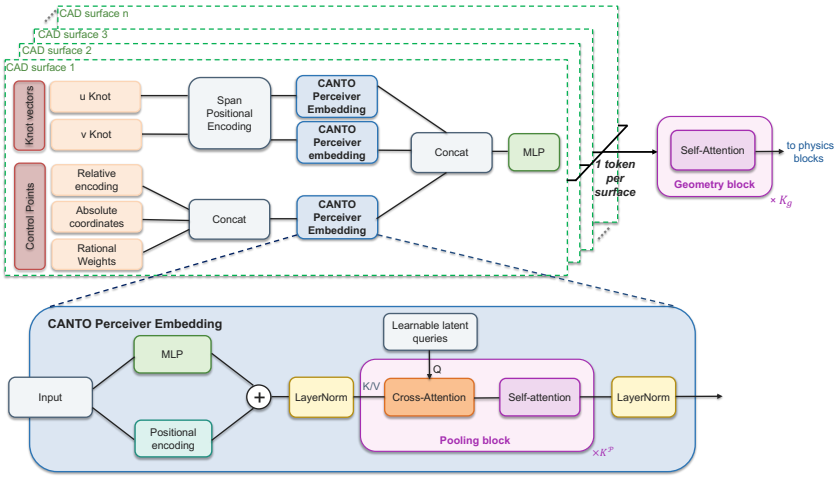


Figure 4: **CANTO geometry encoder architecture.** NURBS surfaces are represented by their control points, weights, and knot vectors. Three Perceiver Pooling modules independently encode the weighted control points and the two knot vectors. Their outputs are concatenated and projected to produce one fixed-dimensional token per patch. Self-attention across patch tokens incorporates global context from the complete CAD geometry.

A Model Designs

Here, we detail the transformer backbone and training configuration used by CANTO. Figure 4 illustrates the NURBS geometry encoder described in Section 4.4.

A.1 Backbone Architecture

For the solution branch of CANTO, we follow the Anchored-Branched Universal Physics Transformer (AB-UPT) Alkin et al. (2025), a Transformer-based multi-branch neural operator featuring dedicated streams for geometry, surface, and volume data. While AB-UPT relies on a point-cloud geometry branch, CANTO replaces this with the CAD geometry encoder described in Section 4.4 and illustrated in Figure 4. The resulting geometry tokens act as the geometry key-value cache for the cross-attention blocks, conditioning the surface and volume streams.

In AB-UPT, surface anchors are sampled from a point cloud. In CANTO, these are replaced by surface tokens obtained from a refined NURBS decomposition, as described in Appendix A.3. Each refined patch is encoded using the Perceiver Embedding procedure described in Section 4.4. The surface tokens provide a geometry-derived context set for the surface stream. During training and inference, arbitrary surface locations are supplied as query tokens decoded against this context. Volume anchors remain sampled from the volume coordinate set, as in the standard AB-UPT.

The key property we inherit is the anchored neural field decoder, which allows the model to predict fields on dense domains without applying quadratic attention to all query points. Let $X^s = \{\mathbf{x}_i^s\}_{i=1}^{N_s}$ and $X^v = \{\mathbf{x}_i^v\}_{i=1}^{N_v}$ denote the surface and volume coordinates. These coordinates specify the query locations at which surface and volume fields are predicted.

The first stage injects the NURBS-based latent geometry tokens $\mathbf{G}^{(L_{\text{geo}})}$ into the surface and volume streams via cross-attention (CA):

$$H_s^{(0)} = \text{CA}\left(E_s(X^s), \mathbf{G}^{(L_{\text{geo}})}, \mathbf{G}^{(L_{\text{geo}})}\right), \quad H_v^{(0)} = \text{CA}\left(E_v(X^v), \mathbf{G}^{(L_{\text{geo}})}, \mathbf{G}^{(L_{\text{geo}})}\right), \tag{6}$$

where E_s and E_v encode the coordinates. This conditions the predicted physical fields at every prediction location on the exact global shape.

Table 3: Geometry encoder hyper-parameters.

Parameter	AhmedML	WindsorML	DrivaerML	HiLiftAeroML
$q_{cp}^{\mathcal{P}}$	8	16	4	16
$d_{cp}^{\mathcal{P}}$	128	192	128	192
$K_{cp}^{\mathcal{P}}$	4	2	4	2
$q_{kv}^{\mathcal{P}}$	4	4	4	4
$d_{kv}^{\mathcal{P}}$	64	64	64	64
$K_{kv}^{\mathcal{P}}$	1	2	1	2
Perceiver block attention heads	4	2	4	2
d	192	192	192	192
K_g	6	6	18	6
Geom. block attention heads	3	3	3	3

Tokens then pass through a stack of shared physics blocks. Each block applies intra-branch self-attention (SA) followed by inter-branch cross-attention:

$$\tilde{H}_{s,v}^{(\ell)} = \mathrm{SA}_\ell\left(H_{s,v}^{(\ell-1)}\right), \tag{7}$$
$$H_s^{(\ell)} = \mathrm{CA}_\ell\left(\tilde{H}_s^{(\ell)}, \tilde{H}_v^{(\ell)}, \tilde{H}_v^{(\ell)}\right), \quad H_v^{(\ell)} = \mathrm{CA}_\ell\left(\tilde{H}_v^{(\ell)}, \tilde{H}_s^{(\ell)}, \tilde{H}_s^{(\ell)}\right).$$

Sharing weights enforces a consistent physical update rule across both streams. These layers use standard pre-normalized Transformer blocks with MLP sublayers and Rotary Position Embeddings (RoPE). Finally, independent decoders and linear heads predict fields like pressure, velocity, and wall shear stress.

To avoid $O(N^2)$ complexity for large N, AB-UPT uses a small subset of $M \ll N$ anchor tokens H_A. The anchors attend to one another and serve as a context set for decoding query tokens. This limits the quadratic bottleneck to $O(M^2)$, while decoding N queries scales linearly, enabling practical million-scale CFD predictions. In CANTO, NURBS-derived surface tokens provide the context for surface prediction, while volume anchors remain sampled from the volume coordinate set.

A.2 Hyper-parameters and Training

Training objective Following AB-UPT, we train the complete model end to end using

$$\mathcal{L} = \mathrm{MSE}(\mathbf{y}_s, \hat{\mathbf{y}}_s) + \lambda\,\mathrm{MSE}(\mathbf{y}_v, \hat{\mathbf{y}}_v),$$

where

$$\hat{\mathbf{y}}_s = D^s(F^s\left(X^s, E^g(X^g), Z^v\right)), \qquad \hat{\mathbf{y}}_v = D^v(F^v\left(X^v, E^g(X^g), Z^s\right)).$$

Here, X^s, X^v, and X^g denote the surface queries, volume queries, and parameterized NURBS geometry, respectively. The normalized targets are $\mathbf{y}_s = (p_s, \tau_x, \tau_y, \tau_z)$ and $\mathbf{y}_v = (p_v, u_x, u_y, u_z)$. The operators E, F, and D denote the geometry encoder, physics blocks, and field decoders, while Z^s and Z^v are the corresponding surface and volume latent representations. The coefficient λ balances the surface and volume objectives.

Hyper-parameter validation For our AB-UPT backbone, we follow the same parameters and settings as the authors Alkin et al. (2025), which are enumerated in Table 4. Note that in each case we optimize the hyper-parameters (e.g., learning rate) over a small validation set. We train our model with Lion optimizer Chen et al. (2023) with a cosine learning rate schedule with a linear warmup. The exact configuration and hyper-paramterers used for each dataset are provided in Tables 3 and 5.

Parameter counts Table 6 reports the number of trainable parameters of CANTO for each dataset, separating those associated with the geometry encoder from those of the remaining model components. The total parameter count ranges from 12.61M to 17.17M across datasets, with the variation stemming almost entirely from the geometry encoder (see Section 4.4), whose size depends on the dataset-specific geometry configuration. In contrast, the remainder of the model architecture is shared across datasets and maintains an approximately constant parameter count of 8M parameters.

Table 4: **Physics and decoder blocks hyper-parameters.**

Parameter	
Hidden dim.	192
Attention heads	3
Surface decoder depth	6
Volume decoder depth	6

Table 5: **Training hyper-parameters.**

Parameter	AhmedML	DrivAerML	WindsorML	HiLiftAeroML
Learning rate	5×10^{-5}	1×10^{-4}	3×10^{-4}	2×10^{-4}
Weight decay	0.05	0.05	0.05	0.05
Training steps	200k	200k	105k	630k
Warmup steps	1,000	2,500	675	31,500
Volume loss weight	0.2	0.2	0.2	0.2

A.3 NURBS Anchor Split Resolution.

Unlike point-cloud methods, CANTO generates surface anchors by parametrically splitting parent NURBS patches. Figure 5 shows that increasing the number of anchor splits from 1 to 4 sharply reduces errors across all fields by providing a finer geometric context. Crucially, while CANTO was trained exclusively at 4 splits, it shows stable performance also at 5 splits. This suggests that the model learns a robust representation of the continuous geometry, remaining invariant to the specific parametric decomposition used during training.

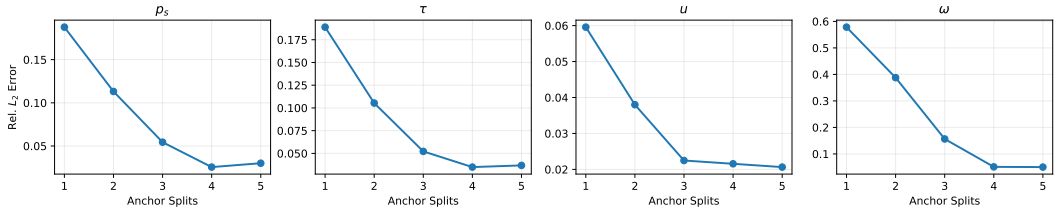


Figure 5: **Effect of the NURBS anchor split resolution on predictive accuracy.** Error drops sharply as the exact geometric context set is parametrically refined, plateauing at 4 splits, the level used for training.

A.4 Point vs. Geometry Anchors.

We consider another configuration of CANTO, closer to the AB-UPT backbone: **CANTO-PA** ("Point Anchors") employs the NURBS-based geometry encoder described in Section 4.4 and samples surface anchors directly from the training point cloud.

To isolate the benefit of the CAD encoder from the geometry anchors, Table 7 compares CANTO-PA (which encodes the NURBS geometry but retains spatially sampled point anchors for the surface) against our primary CANTO model on WindsorML. Furthermore, Figure 6 illustrates that while the CAD encoder alone grants CANTO-PA remarkable robustness to volume sparsity (b), the model still degrades as the surface anchor count drops (a). This confirms that extracting surface tokens directly from exact parametric patches is essential to completely bypass spatial sampling limits.

A.5 Model robustness for varying number of anchors

We compare the anchor efficiency of CANTO against AB-UPT by varying the number of surface and volume anchors (Figure 7). For point-cloud baselines like AB-UPT, reducing the number of surface anchors from 2^{14} to 2^{8} causes a catastrophic collapse in performance for surface fields (the error more than doubles for the surface pressure field). In contrast, because CANTO derives its

Table 6: **Model parameter counts across datasets.** Parameter counts are separated into geometry-related and non-geometry parameters.

Dataset	All	Geometry	Non-geometry
AhmedML	12.61M	4.82M	7.79M
WindsorML	13.47M	5.46M	8.01M
DrivAerML	17.17M	9.24M	7.93M
HiLiftAeroML	13.47M	5.46M	8.01M

Table 7: **Ablation of CANTO variants on the WindsorML dataset.**

Method	p_s	u	τ
CANTO-PA	8.04	7.30	7.55
CANTO	**6.81**	**6.69**	**5.46**

surface tokens directly from the analytical CAD geometry, it completely bypasses spatial surface sampling. Its performance is therefore fully decoupled from the mesh resolution (represented by the flat dashed line) and outperforms the baseline even when AB-UPT uses the maximum 2^{14} surface anchors.

CANTO's superior robustness also extends to the volume domain. As the number of volume anchors is reduced to a sparsity of 2^8 (256 anchors), AB-UPT's error increases sharply. However, CANTO remains remarkably robust. Because the geometry-derived context set captures exact boundary conditions and continuous structural information, the model's physical predictions on the surface (p_s, τ) rely far less on dense volume grids. This geometric foundation grants extreme sample efficiency, allowing CANTO to drastically reduce the attention context set (M) for substantial computational savings without sacrificing physical accuracy.

The previous section demonstrated the strong robustness of CANTO for predicting surface fields. We also evaluate the effect of anchor sparsity on the volumetric fields, u and ω (Figure 8). Similar to the surface fields, CANTO completely bypasses the need for spatial surface anchors, maintaining a flat error curve for both volume velocity and vorticity as the number of surface anchors is reduced. Furthermore, when reducing the number of volume anchors, CANTO still exhibits superior robustness compared to AB-UPT, delaying the performance degradation to much lower anchor counts.

B Model performance

B.1 Computational resources

Table 8 reports the hardware, numerical precision, number of training epochs, and wall-clock training time used for each dataset. All experiments were run on a single node.

Table 8: Computational resources and training time per dataset. All runs use a single node; per-step throughput is dominated by attention over the volume-anchor set.

Dataset	Hardware	Precision	Epochs	Training time
AhmedML	8× H100	FP32	1140	17 h
WindsorML	8× H100	FP32	1326	13 h
DrivAerML	4× GB200	FP16 (mixed)	500	11 h
HiLiftAeroML	4× GB200	FP16 (mixed)	500	69 h

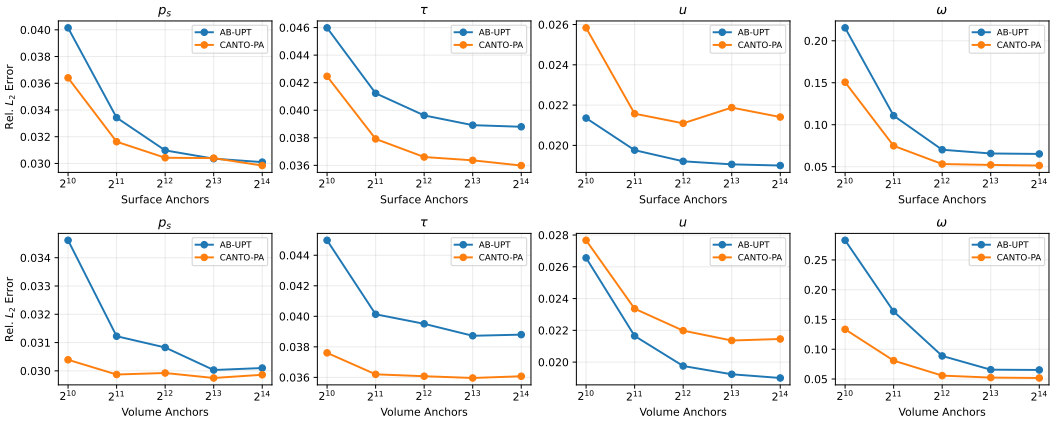


Figure 6: **Robustness to anchor sparsity for CANTO-PA.** Retaining point-based surface anchors causes performance to degrade when the number of surface anchors is reduced (a), even though the global CAD encoding provides strong robustness against volume sparsity (b).

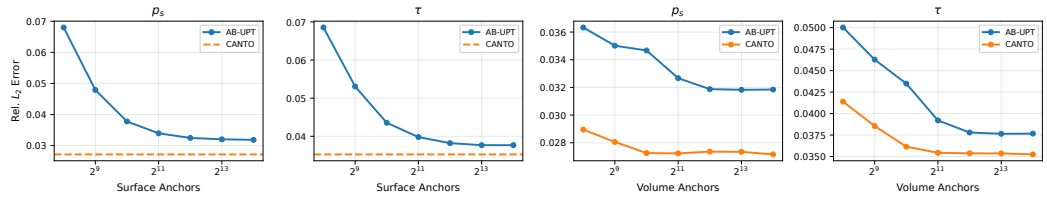


Figure 7: **Robustness to anchor sparsity for surface fields.** Relative L2 error is reported for pressure (p_s) and wall shear stress (τ) against varying the number of surface anchors (left two panels) and volume anchors (right two panels). CANTO completely bypasses the need for spatial surface anchors (shown as a flat dashed line). Furthermore, it maintains high accuracy on these surface fields even at lower volume anchor counts where the AB-UPT baseline degrades sharply.

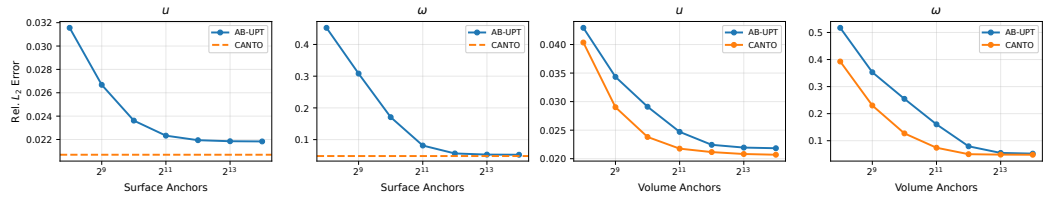


Figure 8: **Robustness to anchor sparsity for volume fields.** Relative L2 error is reported for volume velocity (u) and volume vorticity (ω) against varying surface anchor counts (left two panels) and volume anchor counts (right two panels). CANTO's performance on volumetric fields is completely decoupled from the surface anchor sampling resolution (represented by the flat dashed line), and exhibits superior robustness to volume sparsity compared to AB-UPT.

B.2 Inference latency and peak memory requirement

As shown in Table 9, our method scales to very large prediction problems, performing inference on up to 30 million surface points and 200 million volume points while maintaining a memory footprint below 55 GB. Notably, the geometry encoder (see Figure 4) uses the FlashAttention kernel, allowing us to deal with potentially a high number of token surfaces. Despite the substantial variation in CAD and mesh complexity across the datasets, inference remains practical, requiring approximately 75–160 seconds for the reported cases. The inference time represents the average inference time per element over the whole test set over 5 runs. The throughput is given as the number of query points inferred per second as indicated in Section B.3. Peak memory usage and inference time are directly linked through the chunk size of the query points processed during inference.

Table 9: Inference latency and peak memory.

Dataset	Hardware	Peak mem	Surf. pts	Vol. pts	Inference time	Throughput
AhmedML	H100	13 GB	∼1M	∼22M	∼75 s	∼0.3M queries/s
WindsorML	H100	55 GB	∼5M	∼290M	∼330 s	∼0.9M queries/s
DrivAerML	GB200	40 GB	∼9M	∼150M	∼160 s	∼1.0M queries/s
HiLiftAeroML	GB200	42 GB	∼30M	∼200M	∼155 s	∼1.5M queries/s

Table 10: Effect of query chunk size on inference runtime and peak memory.

Chunk size	10k	50k	100k	250k	500k	1M
Eval. runtime	543 s	209 s	169 s	146 s	138 s	131 s
Peak memory	3.6 GB	3.6 GB	4.5 GB	7.6 GB	12.6 GB	22.7 GB

B.3 Inference runtime-memory trade-off

Since CANTO predicts physical fields at arbitrary query locations, the query points are processed in chunks whose size can be adjusted according to the available GPU memory. Larger chunk sizes reduce the overall evaluation time at the cost of higher peak memory usage. The measured trade-off for CANTO on a full AhmedML evaluation on a single H100 GPU, for 22 million volume query points as well as 1 million surface query points, is summarized Table 10. These results demonstrate that inference can be adapted to the available GPU memory with only a moderate impact on runtime. For example, reducing the peak memory requirement from 22.7 GB to 3.6 GB increases the evaluation time from 131 s to 209 s using a chunk size of 50,000, while still remaining orders of magnitude faster than the underlying CFD simulation, as detailed in Section B.4.

B.4 End-to-end performance comparison with traditional CFD

Table 11 compares the end-to-end runtime of our approach against the corresponding CFD workflows. AhmedML and DrivAerML were generated using OpenFOAM, WindsorML using Volcano ScaLES, and HiLiftAeroML using Fidelity CHARLES. For AhmedML and DrivAerML, where both CANTO and CFD timings are available, our model achieves orders-of-magnitude faster inference than the underlying CFD simulations: approximately 2,300× for AhmedML and 900× for DrivAerML compared with the CFD solve alone.

When the one-time CAD untrimming step is included in the complete raw-CAD-to-field workflow rather than treated as offline preprocessing, the wall-clock ratio remains approximately 2,300× for AhmedML, which is already untrimmed, and greater than 48× for DrivAerML, even without including the unavailable meshing time. Untrimming is intended as an offline preprocessing step and was designed for high dataset throughput rather than low per-case latency. The current implementation processes each geometry serially while processing multiple geometries concurrently across a 144-core CPU node. The reported ratios include the current untrimming implementation; further engineering optimization may reduce this preprocessing cost. Because CANTO and the CFD solvers run on different hardware, these values are wall-clock ratios rather than hardware-normalized speedups.

C Datasets

AhmedML. The AhmedML dataset Ashton et al. (2024b) is an open-access dataset containing high-resolution computational fluid dynamics (CFD) simulations for 500 distinct geometric variants of the Ahmed body, a widely used benchmark in automotive aerodynamics. The simulations are performed in OpenFOAM using a hybrid RANS–LES turbulence modeling approach, capturing important flow phenomena such as pressure-induced separation and complex three-dimensional vortex structures. Each case is discretized on meshes comprising approximately 1 million surface cells and 20 million volume cells. The dataset was split into 400 training, 50 validation, and 50 test samples, following Alkin et al. (2025). The exact split is publicly available at `https://github.com/Emmi-AI/noether/blob/main/src/`

Table 11: **Inference and preprocessing cost compared with the CFD pipeline.** Inference speedup compares model inference with the CFD solve, while pipeline speedup compares untrimming and inference with CFD meshing and solve. "–" indicates that the corresponding information was not reported in the reference paper.

Dataset	GPU	Untrim.	Inference	Meshing	CFD	Inf. speedup	Pipeline speedup
AhmedML	H100	0 s	∼75 s	30 min	48 h	∼2300×	∼2300×
WindsorML	H100	∼5 s	330 s	2 min	28 h	∼300×	∼300×
DrivAerML	GB200	∼2800 s	∼160 s	–	40 h	∼900×	∼50×
HiLiftAeroML	GB200	∼1000 s	∼155 s	–	–	–	–

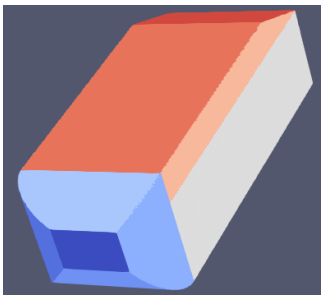


Figure 9: Surface decomposition of an AhmedML geometry into 13 untrimmed NURBS patches. The decomposition consists of nine planar surfaces and four ellipsoidal surfaces near the front, with each patch shown in a different color..

`noether/data/datasets/cfd/caeml/ahmedml/split.py`. Notably, the AhmedML dataset does not distribute CAD files for each configuration. However, each AhmedML geometry is defined by explicit geometric design parameters. We use these parameters to reconstruct the corresponding analytic CAD geometry exactly as 13 untrimmed NURBS surfaces per case, as illustrated in Fig. 9. This reconstruction is not obtained by fitting to a mesh or point cloud; it is generated directly from the parametric definition of the AhmedML design family. The resulting representation consists of nine planar surfaces and four curved surfaces near the front of the body. Two versions of the NURBS surfaces were utilized in this paper. A first *Variable-size* version parametrizes surfaces in the way a CAD file would, that is

- Planar surfaces are parametrized by NURBS surfaces of bi-degree $(1, 1)$, with four control points corresponding to the vertices of these surfaces and associated uniformly unit weights, and knot vectors $U = V = (0, 0, 1, 1)$
- Curved surfaces parametrized by NURBS surfaces of bi-degree $(2, 2)$, with nine control points corresponding to the four vertices of these surfaces, the middle of the straight edges, three vertices at the intersection of the tangents in the previously constructed points. The associated weight matrix is given by

$$W = \begin{bmatrix} 1 & 1 & 1 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

  and knot vectors $U = V = (0, 0, 0, 1, 1, 1)$

A second *Fixed-size* version uniformizes all surfaces, including the planar ones, to bi-quadratic surfaces with nine control points and knot vectors $U = V = (0, 0, 0, 1, 1, 1)$ (this is simply done by inserting, for the planar surfaces, control points at the middle of each edge and at the center of the surface).

WindsorML. WindsorML Ashton et al. (2024a) is a large-scale, high-resolution dataset comprising 350 geometric configurations of the Windsor body. Each configuration is simulated using wall-modeled large-eddy simulation (WMLES) on computational grids exceeding 280 million cells, with a Cartesian immersed-boundary approach to ensure accurate aerodynamic predictions. The dataset provides parameterized CAD models in `.stp` format, containing 42 NURBS surfaces each, defined

by seven geometric design variables relevant to automotive applications, along with aerodynamic force and moment coefficients. All cases are evaluated under identical flow conditions. As no explicit data split is provided in the original paper, we follow the proposed 60/20/20 partition using an ordered split without shuffling: the first 60% of samples are used for training, the next 20% for validation, and the final 20% for testing.

DrivaerML. The DrivaerML dataset Ashton et al. (2024c) comprises 500 parametrically modified variants of DrivAer vehicle geometries Heft et al. (2012), and was created to address the limited availability of open-access, large-scale computational fluid dynamics (CFD) data for automotive aerodynamics. The CFD simulations were performed on volumetric meshes containing approximately 140 million cells using a hybrid RANS–LES methodology Reynolds-Averaged Navier–Stokes–Large Eddy Simulation, which represents one of the highest-fidelity approaches currently employed in the automotive industry. Each sample includes surface meshes with roughly 8.8 million points, providing surface pressure and wall shear stress data, as well as volumetric fields for velocity, pressure, and vorticity. The available data, which includes the 500 variants minus 16 unavailable geometries, are divided into 400 training samples, 34 validation samples, and 50 test samples, following the split used in Alkin et al. (2025), and provided at `https://github.com/Emmi-AI/noether/blob/main/src/noether/data/datasets/cfd/caeml/drivaerml/split.py`. CAD geometries under .stp format were provided to us, each containing roughly 8.5k NURBS surfaces.

HiLiftAeroML. HiLiftAeroML Ashton et al. (2026) is a large-scale, high-fidelity CFD dataset for high-lift aircraft aerodynamics, comprising 1,800 simulations generated from 180 parametric variants of the NASA Common Research Model in high-lift configuration (CRM-HL), each evaluated at 10 angles of attack ranging from 4° to 22° in 2° increments. The geometries span an eight-dimensional design space controlling the inboard and outboard slat and flap deflection angles and gap multipliers, covering a broad range of high-lift configurations. Simulations were performed using the GPU-accelerated Fidelity Charles solver with explicit wall-modeled large-eddy simulation (WMLES) on solution-adapted Voronoi grids containing approximately 300–500 million control volumes, enabling the resolution of complex separated flows and both pre- and post-stall regimes. Each sample provides the aircraft geometry, time-averaged surface and volumetric flow fields, as well as integrated aerodynamic forces and moments. We use the full in-distribution split proposed by the authors, consisting of 1,260 training, 180 validation, and 360 test cases, corresponding to a 70/10/20 partition of the complete dataset. The angle of attack is incorporated by rotating the input CAD geometry and corresponding mesh before passing them to CANTO.

D Supplemental NURBS background and untrimming

D.1 B-spline basis functions

Given a knot vector U, the degree-p B-spline basis functions $N^U_{k,p}$ ($k \geq 0$) are defined recursively. For $p = 0$,

$$N^U_{k,0}(u) = \begin{cases} 1, & \text{if } u_k \leq u < u_{k+1}, \\ 0, & \text{otherwise,} \end{cases}$$

and for $p \geq 1$,

$$N^U_{k,p}(u) = \alpha_{k,p}(u)\, N^U_{k,p-1}(u) + \beta_{k,p}(u)\, N^U_{k+1,p-1}(u),$$

where

$$\alpha_{k,p}(u) = \begin{cases} \frac{u-u_k}{u_{k+p}-u_k}, & u_{k+p} \neq u_k, \\ 0, & \text{otherwise} \end{cases}, \quad \beta_{k,p}(u) = \begin{cases} \frac{u_{k+p+1}-u}{u_{k+p+1}-u_{k+1}}, & u_{k+p+1} \neq u_{k+1}, \\ 0, & \text{otherwise.} \end{cases}$$

D.2 Untrimming

Working with the trimming curves directly can be a practical challenge due to the complex nature of the way they specify the trimmed parameter domain $\mathcal{D}$. The work of Massarwi et al. (2018) proposes algorithms to untrim a NURBS surface by decomposing it into a collection of tensor-product NURBS patches. For a given patch Σ, this yields a decomposition $\Sigma = \bigcup_{j=1}^{n_\Sigma} \Sigma_j$, where each Σ_j is diffeomorphic to $[0,1]^2$ and represented by a NURBS parameterization S_j given by (1).

We denote by n_Σ the number of untrimmed surfaces and S_j their respective maps for $k = 1, \ldots, n_\Sigma$. These maps form the geometric inputs to our model (Section 4.2).

While exact, the untrimming procedure of Massarwi et al. (2018) significantly increases the number of patches, the bi-degree of the NURBS surfaces, and the number of control points. We instead use a boundary-first construction (Appendix D.2) that exactly preserves watertightness, reproduces the input surface up to a small geometric residual (Table 12), and keeps the number of patches, the bi-degree, and the number of control points comparable to those of the input.

Construction Given a trimmed patch Σ, let $S : [0,1]^2 \to \mathbb{R}^3$ be its underlying NURBS mapping of bi-degree (p, q). The trimmed parameter domain $\mathcal{D} \subseteq [0,1]^2$ is bounded by closed loops of NURBS edge curves $\mathcal{T} = \mathbf{c}^\ell$.

We decompose Σ into a sequence of untrimmed rational tensor-product NURBS patches Σ_k, each diffeomorphic to $[0,1]^2$ and parameterized by

$$S_k : [0,1]^2 \to \mathbb{R}^3.$$

Each S_k is defined by knot vectors $(U^{(k)}, V^{(k)})$ and a weighted control net $(P_{ij}^{(k)}, w_{ij}^{(k)})$.

As detailed below, the four boundary isocurves of each patch, obtained by fixing one parametric coordinate to 0 or 1, are exact pieces of the input trim. Watertightness therefore holds by construction: adjacent patches use a common lift of their shared edge and share the corresponding row of control points.

The trimmed parameter domain $\mathcal{D}$ of patch Σ is decomposed via the ear-clipping algorithm (Meisters, 1975) into four-edge subdomains $\mathcal{D}_m$ bounded by canonical edge curves $(\mathbf{c}_m^{u_0}, \mathbf{c}_m^{u_1}, \mathbf{c}_m^{v_0}, \mathbf{c}_m^{v_1})$. The four curves are lifted to their least common refinement by knot insertion and degree elevation, then assigned as the four boundary isocurves of Σ. The interior control points are determined by collocating the patch at Greville abscissae to match the parent surface evaluated through a transfinite Coons blend. Writing $\boldsymbol{\varphi}_m : [0,1]^2 \to \Sigma_m$ for the Coons blend that interpolates the four edge curves in the parent's UV and (ξ_i, η_j) for the Greville abscissae of $(\bar{U}_m, \bar{V}_m)$, the interior weighted control points solve

$$S_m(\xi_i, \eta_j) \;=\; S\big(\boldsymbol{\varphi}_m(\xi_i, \eta_j)\big)$$

at every interior node with boundary control points held fixed at their lifted values. The system is solved by Tikhonov-regularised least squares in the rank-deficient case. When $\boldsymbol{\varphi}_m$ is affine, $\mathbf{S}_k \circ \boldsymbol{\varphi}_m$ is itself a tensor-product NURBS and the construction is exact.

The control points of all emitted patches form a single homogeneous pool $\mathbf{P}_h \in \mathbb{R}^{N\times 4}$ whose rows store $(w_{ij}\mathbf{P}_{ij}, w_{ij})$. Patch Σ_m is specified by its knot vectors $(U^{(m)}, V^{(m)})$. Two patches sharing a canonical trim edge have identical pool indices on the corresponding boundary row, so watertightness is encoded in the index structure alone and holds across heterogeneous parent surfaces.

Empirical evaluation We report per-patch surface preservation error in Table 12: the worst case is sub-millimetre and the median is several orders of magnitude smaller.

Table 12: Per-patch surface preservation error of the untrimming pipeline on the Windsor dataset. Per-patch error is the maximum, over an 8×8 uniform grid in $[0,1]^2$, of the Newton-projected perpendicular distance from the output sample to the original CAD parent surface; we report the maximum and median across patches, normalised by the model's 3-D bounding-box diagonal.

Dataset	Bbox diag. (m)	Max rel. error	Median rel. error
Windsor	1.19	6.40×10^{-4}	1.73×10^{-14}

Untrimming memory requirements and runtime The current untrimming implementation processes each geometry serially on the CPU. At the dataset level, multiple geometries are processed concurrently on a 144-core CPU node with 2 TB of RAM. The implementation was designed for high dataset-level throughput rather than minimum latency for an individual geometry. Table 13

reports both the per-geometry measurements and the time required to preprocess each complete dataset.

Untrimming time varies with CAD complexity, including the number of trimmed surfaces, spline degrees, control points, trimming curves, and holes. This preprocessing is performed once per geometry, and the resulting parametric NURBS representation can be reused for subsequent model evaluations. It is separate from CANTO's model-inference time.

Table 13: Untrimming computational cost.

Dataset	**Runtime**	**Memory**	**Full preprocessing**	**Initial surf.**	**Final surf.**
WindsorML	∼5 s	0.5 GB	∼15 s	42	71
DrivAerML	∼2800 s	3.9 GB	∼8 h	∼8.5k	∼45k
HiLiftAeroML	∼1000 s	2.3 GB	∼10 h	∼350	∼4.5k

E Inverse design and Optimization

Problem statement. We minimize aerodynamic drag over the six-parameter AhmedML body family subject to a lower bound on enclosed volume and an upper bound on lift, as expressed in the problem (5). D and L are the drag and lift coefficients evaluated at the *fixed* dataset reference area $A_{\mathrm{ref}} = 0.112032\,\mathrm{m}^2$ (the nominal Ahmed cross-section $0.389 \times 0.288\,\mathrm{m}$), so they are proportional to drag and lift *force* rather than being normalized per geometry. The volume V, in turn, is the enclosed volume of the differentiable geometry map, obtained in closed form from the NURBS control net. The value 0.05 for the lift constraint defines a non-degenerate feasible set, comprising 298 of the 500 designs in the dataset.

Design space. The six parameters and their dataset bounds $[x_i^{\mathrm{lo}}, x_i^{\mathrm{hi}}]$ are given in Table 14, and define the feasible box $\mathcal{X} = \{x \in \mathbb{R}^6 : x^{\mathrm{lo}} \le x \le x^{\mathrm{hi}}\}$. The optimization is carried out in normalized coordinates obtained by the affine map

$$z_i = \frac{x_i - x_i^{\mathrm{lo}}}{s_i}, \qquad x_i = x_i^{\mathrm{lo}} + s_i z_i, \qquad s_i = x_i^{\mathrm{hi}} - x_i^{\mathrm{lo}}, \qquad i = 1, \dots, 6, \tag{8}$$

which carries $\mathcal{X}$ onto the unit cube $[0,1]^6$, so the bound constraints become $0 \le z \le 1$ and every objective and constraint gradient transforms by the diagonal scaling.

Table 14: Design parameters and bounds (mm).

Parameter	Lower	Upper
body length	800.0	1200.0
body height	250.0	315.0
body width	300.0	500.0
front arc diameter	160.0	240.0
slant angle length	75.0	246.2
slant angle height	26.0	216.5

Drag and Lift integration Drag and lift are computed by integrating the surrogate-predicted surface coefficients directly over the CAD geometry, without introducing a CFD mesh into the optimization loop. Each of the 13 NURBS patches of the Ahmed body is evaluated on a 32×32 midpoint grid in parameter space, giving 13,312 quadrature points in total. At each point, the exact NURBS derivatives provide the oriented normalized surface normal ν (see (2)) and the surface Jacobian $\left|\frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v}\right|$. Thus, for any surface quantity f, we approximate its integral over a patch Σ as

$$\int_\Sigma f\, dA \;\approx\; \sum_{a=1}^{32}\sum_{b=1}^{32} f(u_a, v_b) \left\| \frac{\partial S}{\partial u}(u_a, v_b) \times \frac{\partial S}{\partial v}(u_a, v_b) \right\| \Delta u\, \Delta v,$$

where (u_a, v_b) are the midpoint quadrature points.

At the same quadrature points, CANTO predicts the pressure and wall-shear stress (p_s, τ), which are integrated to obtain the drag D and lift L. We use the dataset's fixed reference area $A_{\text{ref}} = 0.112, \text{m}^2$ for all designs.

Surrogate-based optimization. As indicated above, D and L are provided by a neural surrogate trained on the AhmedML dataset, which comprises 500 CFD simulations: 400 are used for training, while the remaining 100 are held out for validation and testing. Since both the mapping from the design variables $\mathbf{z}$ to the NURBS geometry and the neural surrogate are differentiable, the gradients $\nabla_{\mathbf{z}} D$, $\nabla_{\mathbf{z}} V$, and $\nabla_{\mathbf{z}} L$ can be computed directly through automatic differentiation. We therefore solve (5) using SLSQP (Kraft, 1994), providing these exact gradients to the optimizer. The optimization is initialized from the lowest-drag feasible design in the dataset and subsequently explores the continuous design space using the surrogate-derived gradients.

CFD validation. To validate the optimized designs, each geometry is tessellated at the same mesh density used in the original dataset and re-simulated following the full AhmedML CFD protocol described in (Ashton et al., 2024b). Specifically, the simulations are performed using OpenFOAM v2306, with `snappyHexMesh` used to generate meshes containing approximately 22 million cells. The flow is then simulated using `pimpleFoam` for 133,333 time steps, corresponding to a final simulation time of $t = 80$,s. Consistent with the dataset protocol, the aerodynamic coefficients are time-averaged over the interval $t \in [20, 80]$,s.

Results. Figure 3 describes the validated front. Every optimized design improves on the best dataset design at its volume, by 4.4% to 20.5%, and the surrogate's predictions are accurate to within 2.2% throughout. The lift constraint is satisfied in CFD at every threshold.

The optimized designs achieve gains across the entire front, with improvements ranging from 4.4% to 20.4% relative to the best dataset designs at the corresponding volume thresholds. At $V_0 = 70$ and 80 L, the volume constraint is inactive, and both thresholds converge to the same design with a volume of 84.3 L, corresponding to the unconstrained optimum within the considered design family. Table 15 reports the results of these optimization and validation procedures in detail.

Table 15: **CFD-validated optimized front**. D_{cfd} is the protocol-averaged coefficient; "dataset" is the lowest-drag dataset design satisfying the same constraints.

V_0 [L]	L_{cfd}	D_{surr}	D_{cfd}	error	dataset	gain
70	0.044	0.1566	0.1586	$+1.3\%$	0.1867	$+15.1\%$
80	0.044	0.1566	0.1586	$+1.3\%$	0.1867	$+15.1\%$
90	0.020	0.1649	0.1685	$+2.2\%$	0.1867	$+9.7\%$
100	0.027	0.1811	0.1817	$+0.4\%$	0.2285	$+20.4\%$
110	0.013	0.2004	0.2026	$+1.1\%$	0.2317	$+12.5\%$
120	0.020	0.2196	0.2241	$+2.1\%$	0.2589	$+13.4\%$
130	-0.045	0.2393	0.2371	-1.0%	0.2656	$+10.7\%$
140	0.013	0.2586	0.2637	$+2.0\%$	0.2971	$+11.2\%$
150	0.028	0.2790	0.2840	$+1.8\%$	0.2971	$+4.4\%$

F Qualitative Surface-Pressure Results

Figures 10 and 11 provide representative qualitative visualizations of surface-pressure predictions on AhmedML and WindsorML. In both cases, the predicted fields capture the large-scale pressure distribution and the dominant high-gradient structures, with the largest residuals concentrated near geometric transition regions.

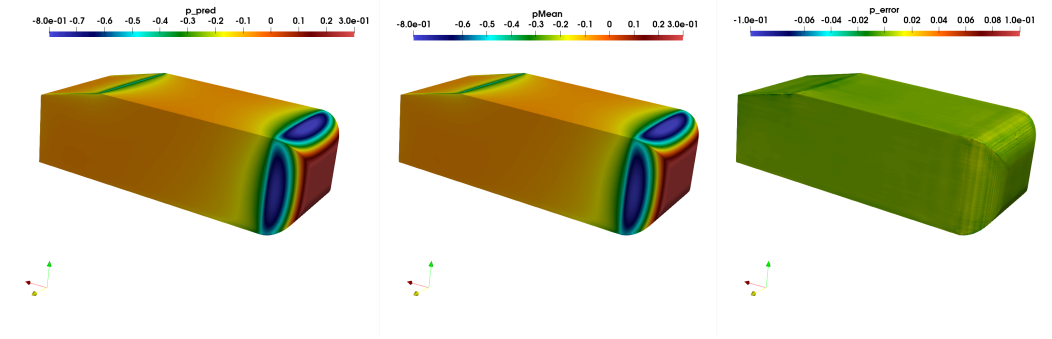


Figure 10: **Qualitative AhmedML surface-pressure prediction.** Left: CANTO prediction. Center: CFD reference field. Right: pointwise prediction error on the vehicle surface.

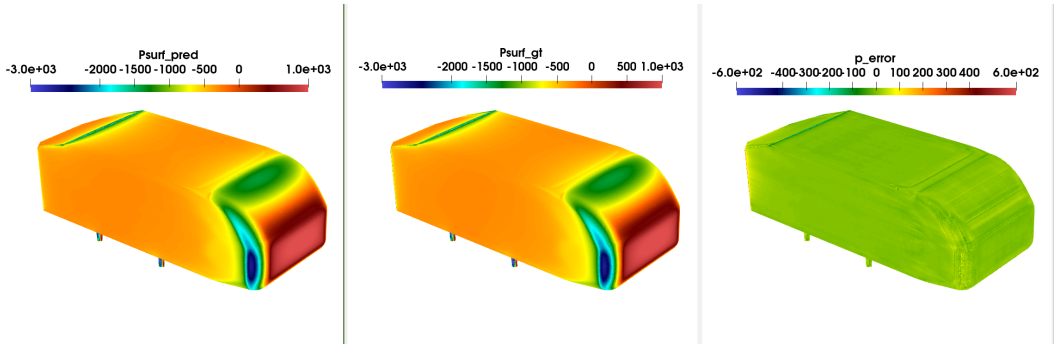


Figure 11: **Qualitative WindsorML surface-pressure prediction.** Left: CANTO prediction. Center: CFD reference field. Right: pointwise prediction error on the vehicle surface.