# AeroAgent: A Vision–Physics–Decision Framework for Aerodynamic Vehicle Design

Ye Liu[1,2], Shouyi Liu[1,2], Huiyu Yang[3], Jianghang Gu[4], Wenhao Fan[1,2], Zhongxin Yang[4], Ding Wang[1,2], Simeng Chen[3], Zirun Jiang[3], Yuanwei Bin[1,5], Shiyi Chen[1], Yuntian Chen[1*]
[1]Eastern Institute of Technology, Ningbo [2]Shanghai Jiao Tong University
[3]Southern University of Science and Technology [4]Peking University
[5]Shenzhen TenFong Technology Co., Ltd. *Corresponding author
liuye66a@gmail.com; shouyiliu@sjtu.edu.cn; ychen@eitech.edu.cn

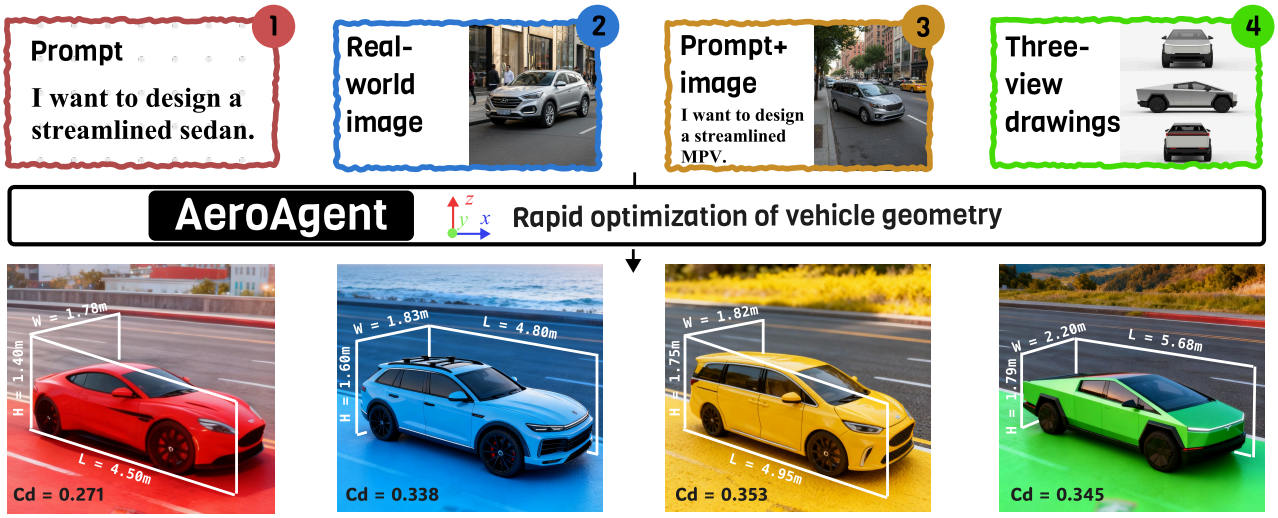



Figure 1. **AeroAgent** takes heterogeneous design intent as input: (1) a pure text prompt, (2) a real-world vehicle image, (3) a text-conditioned image, or (4) three-view drawings. The bottom row shows representative optimized designs with inferred overall dimensions, predicted drag coefficients $C_d$, illustrating rapid aerodynamic refinement of vehicle shape directly from high-level user intent.

## Abstract

*Modern generative models can propose striking 3D vehicle shapes from text and images, but turning these sketches into aerodynamically efficient, regulation-compliant designs still requires weeks of high-fidelity computational fluid dynamics (CFD) and manual iteration. As a result, fast 3D generation without trustworthy physics in the loop does little to reduce end-to-end design time. We study how an AI agent can close this loop under a strict CFD budget. We introduce **AeroAgent**, a vision–physics–decision framework built around a single 3D, editable surface representation for vehicle shapes. A vision module turns text and 2D references into diverse, standardized 3D candidates and supports image-level edits. A physics module, **AeroFormer**, is a geometry-guided Transformer surrogate trained on a large-scale vehicle aerodynamics dataset of roughly 50k CFD simulations; three task-specific heads predict drag ($C_d$), surface pressure, and velocity fields on shared 3D grids. A decision module encodes regulatory size limits and aesthetic constraints as feasibility tests, uses prototype priors and surrogate sensitivities to guide free-form deformation edits, and runs a budget-aware propose–evaluate–refine loop in which only the final top-K shapes are confirmed by high-fidelity CFD. In extensive experiments across five common vehicle classes, running only five propose–evaluate–refine iterations per vehicle reduces drag by an average of 2–12% and cuts high-fidelity CFD calls by 50–80% compared to baseline workflows, while preserving or improving styling quality.*

## 1. Introduction

Early-stage vehicle styling must balance aesthetics, low drag ($C_d$), and regulatory size constraints in a high-dimensional shape space [8, 24, 53]. In practice, design-

ers iterate between sketches, 3D models, and simulation engineers, with high-fidelity computational fluid dynamics (CFD) and manual edits consuming most of the calendar time. Recent generative models can produce striking 3D vehicles from text or images [41], but turning these sketches into aerodynamically efficient, regulation-compliant designs still typically requires weeks of CFD runs and human iteration [7, 9, 48, 53]. Fast 3D generation without trustworthy physics in the loop does little to reduce end-to-end design time. Several research threads are relevant but address only parts of this loop. Text-to-3D [35, 36] and CAD-focused generative models explore rich shape spaces and enable interactive styling, but they treat downstream physics as an afterthought [32, 63]. Flow surrogates and neural operators for CFD approximate pressure and velocity fields orders of magnitude faster than high-fidelity solvers [38, 42, 61], but they are usually trained in isolation and not integrated with editing or decision-making. Classical shape optimization and aero tuning, including adjoint methods, directly minimize drag but require a CFD solve at each iteration and are often tied to specific parameterizations and boundary conditions. In other words, generation-only, surrogate-only, and CFD-only pipelines each solve a single leg of the problem; none provides a coherent, budget-aware design loop that a styling team can run end-to-end.

We study how an AI agent can close this loop under a CFD budget. As shown in Figure 1, we introduce **AeroAgent**, a vision–physics–decision framework built around a single 3D vehicle model. A vision module turns text and 2D references into diverse, standardized 3D candidates and supports lightweight image-level and mesh-level edits. A physics module, **AeroFormer**, is a geometry-guided Transformer trained on a large-scale vehicle aerodynamics dataset; three task-specific surrogates predict drag, surface pressure, and velocity fields on shared 3D grids. A decision module encodes regulatory size limits and aesthetic constraints as feasibility tests, uses prototype priors and surrogate sensitivities to guide deformation edits, and runs a budget-aware propose–evaluate–refine loop in which only the final top-$K$ shapes are confirmed by high-fidelity CFD.

In extensive experiments across five common vehicle classes, AeroAgent achieves an average of 2–12% lower drag with 50–80% fewer CFD runs than baseline workflows, while preserving or improving perceived styling quality. Compared to strong 3D aerodynamics flow surrogates, AeroFormer attains lower relative $L_2$ error and higher $R^2$ on pressure and velocity fields across five common passenger-vehicle classes. Case studies further show that the planner can progressively refine a given vehicle shape toward lower drag in small, interpretable steps. Our contributions can be summarized as follows:

- We propose AeroAgent, an integrated vision, physics and decision framework for early-stage aerodynamic vehicle design under strict CFD budgets, built on a single standardized, editable 3D representation.
- We introduce AeroFormer, a mesh-free, geometry-guided Transformer surrogate trained on a large and high-fidelity vehicle aerodynamics dataset, and show that it outperforms prior 3D flow surrogates on both scalar and field-level metrics.
- We demonstrate that AeroAgent generalizes across five common vehicle classes, reliably reduces drag on production-style models with far fewer CFD simulations, and provides field-level rationales that guide editable design updates.

## 2. Related work

**Generative 3D.** Recent years have seen rapid progress in 3D generative models that synthesize shapes from images, text, or sparse partial observations, including point-cloud GANs [37] and diffusion models [59], neural implicit representations [46], and mesh/CAD generative models [18, 31, 39, 56, 57, 62]. CAD-oriented LLM frameworks such as CAD-Llama [32] and related programmatic shape generators model parametric curves and surfaces for text- or sketch-driven CAD authoring. While they greatly expand the stylistic design space, physical performance is typically evaluated only ex post and aerodynamic feedback is not integrated into the generation loop [60]. In contrast, our vision module is explicitly closed-loop: it outputs standardized, CFD-ready meshes and lightweight edit handles that can be repeatedly evaluated and refined by a downstream physics-informed planner.

**Surrogates for fluid dynamics.** Neural surrogates for PDEs and CFD span convolutional, graph-based, and operator-learning approaches, including neural operators for continuous fields [12, 33, 34, 42, 61], mesh-based architectures for fluid simulation [47, 54], and domain-specific surrogates for RANS and LES in aerodynamics [9, 13, 14, 45]. These methods dramatically accelerate inference over high-fidelity solvers but are often trained on narrow flow families or require solver-specific mesh inputs. AeroFormer builds on this line of work while targeting external vehicle aerodynamics under realistic regulatory constraints: it employs a mesh-free, geometry-conditioned Transformer with coupled volumetric and surface branches, trained on a large corpus of Lattice–Boltzmann simulations [22] for five common body classes to jointly predict drag, surface pressure, and velocity fields.

**Aerodynamic design and interactive agents.** Classical aerodynamic shape optimization relies on adjoint methods, response surfaces, and free-form deformation (FFD) to reduce drag under constraints [55]. More recent pipelines couple surrogates with low-dimensional design loops and selective CFD queries [15, 51], but remain largely offline and require manual parameterization. In parallel, AI

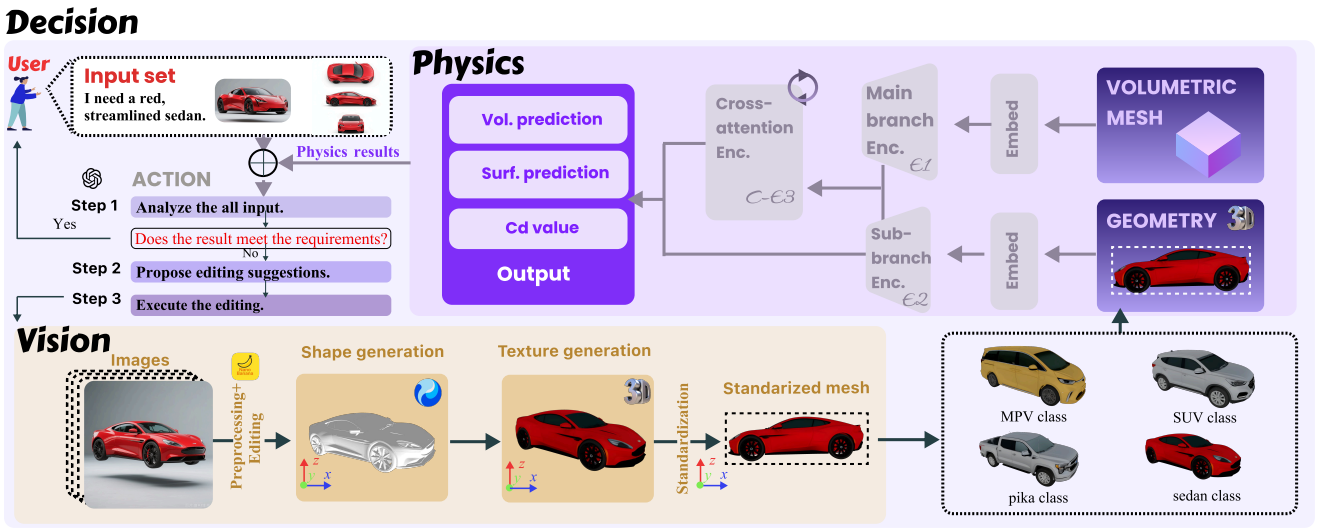



Figure 2. **Overall architecture of AeroAgent.** The Vision module maps user inputs (text or images) to standardized 3D car meshes via perceptual pre-processing, shape and texture generation, and class-specific normalization. The Physics module consumes these meshes and, through a cross-attentional encoder with main and auxiliary branches, predicts volumetric and surface flow fields together with the drag coefficient $C_d$. The Decision module closes the loop by combining user intent and physics predictions to test requirement satisfaction, triggering mesh or image edits and iterative refinements until a satisfactory design is reached.

agents and vision-based assistants have been explored for interactive editing [58], yet rarely operate on full 3D flow fields. AeroAgent unifies these directions by combining a text/image-to-3D front end, a CFD-trained surrogate, and an integrated planner that enforces regulatory size and aesthetic feasibility, forming a budget-aware loop that explores diverse vehicle shapes toward lower drag in a few iterations.

## 3. Method

In this section, we first construct a large-scale vehicle simulation dataset to train an AI surrogate that predicts drag, pressure, and velocity. We then present the AeroAgent framework, which unifies vision, physics, and decision into a propose–evaluate–refine loop: the surrogate provides fast aerodynamic feedback, the planner enforces regulatory size feasibility and user intent, and only a small number of high-fidelity computational fluid dynamics (CFD) simulations are used to confirm the final designs. The overall AeroAgent framework structure is shown in Figure 2.

### 3.1. Problem Setup

**Problem (feasibility form).** Given structured user intent $U$, an initial pool $C_0$ of standardized STL candidates in the shape space $\mathcal{X}$, a feasible set $\mathcal{F} \subset \mathcal{X}$ encoding regulatory size limits, a surrogate $S$ that provides aerodynamic feedback (e.g., $\hat{C}_d$, surface pressure), and a budget $B_{\text{hf}}$ of high-fidelity CFD confirmations for final designs, find one or more $x^\star \in \mathcal{X}$ such that

$$\begin{cases} x^\star \in \mathcal{F} & \text{(regulatory size feasibility)} \\ \overline{C_d}(x^\star) \le \tau_d & \text{(low drag, if requested)} \\ \mathrm{E}(x^\star; U) = 1 & \text{(aesthetics, if requested)} \end{cases} \tag{1}$$

using a closed-loop propose, evaluate and refine procedure with $S$. Only the converged top-$K$ candidates, with $K \le B_{\text{hf}}$, are confirmed by high-fidelity CFD simulations, and all inner-loop iterations rely solely on the surrogate. Here $\overline{C_d}$ denotes drag, $\mathrm{E}(\cdot; U) \in \{0, 1\}$ is a binary predicate that indicates whether a design satisfies the user's aesthetic intent, $C_0 \subset \mathcal{X}$ is the initial candidate pool, and $\tau_d$ is the drag threshold. If the user does not specify drag or aesthetic requirements, we still predict and log $\overline{C_d}$ and E but do not enforce the corresponding constraints.

### 3.2. Vision: Candidate Generation and Editing

This module converts user intent into high-throughput standardized STL candidates, supports both image-level and mesh-level local edits, and enforces geometric normalization and mesh health checks so that downstream physics and decision modules see consistent inputs.

**Intent parsing and prompt normalization.** Natural language intent is first parsed into a structured visual specification (viewpoint, background/lighting, body type, era/style anchors, and negative constraints) that is shared by both text-to-image and image-to-3D. We fix a front three-quarter, neutral setup, restrict body type to {sedan, SUV, MPV, pickup, sport car}, and inject modern style anchors and negative phrases (e.g., no extreme track/camber, no truncated crops). For text-only inputs, we synthesize a 2D conditioning image under this specification with the nano banana model [17, 19], covering both shape and texture generation, whereas for user images we apply 2D standardization—matting to remove background clutter, mild exposure/white-balance correction, and local retouch on salient parts such as headlamps, grille, and roofline.

**Image-to-3D and mesh post-processing.** A unified

image-to-3D model, instantiated with Hunyuan3D 3.0 [30], then reconstructs meshes from the standardized images and exports STL. We apply a fixed post-processing routine: PCA-based axis alignment with nose-direction disambiguation, up-direction and ground-plane fitting using wheel cues, recentering and unit normalization (m), and mesh-health checks for watertightness and normal consistency. Before entering the design loop, each candidate's global dimensions $(L, W, H)$ are projected into the regulatory/manufacturing interval of its body class using minimum-change anisotropic scaling with priority $L \rightarrow H \rightarrow W$, and the scale factors are logged in the metadata. For localized fine edits we provide small-step free-form deformation (FFD): typical actions include adjustments of windscreen angle, roof arc, boat-tail taper, diffuser angle, and wheel-arch shaping. All edits are parameterized, step-limited, and fully logged so that centimetre-scale mesh-level micro-edits complement coarse image-level appearance changes.

**Outputs and interface.** Each candidate is finally packaged as a standardized STL plus a compact `meta.json` containing bounding box and wheelbase, the coordinate transform (X-forward, Y-width, Z-up; units m), a mesh-health summary, the body-class tag, and, when used, the FFD lattice/masks and edit history. This interface supplies consistent geometry and traceable edit information to the physics and decision stages without exposing implementation-specific details.

### 3.3. Physics: AI Surrogate for Aerodynamic

This module provides fast and scalable aerodynamic evaluation: given a standardized vehicle geometry, it produces pressure and velocity fields or drag estimates quickly enough to drive the propose–evaluate–refine loop. To achieve this, we introduce **AeroFormer**, a geometry-guided Transformer surrogate designed for industrial-scale inference. The overall AeroFormer framework structure is shown in Figure 3.

**General input encoding.** AeroFormer adopts a unified point-based representation for both the volumetric flow domain and the vehicle surface. Let

$$X = \{\mathbf{x}_i \in \mathbb{R}^3 \mid i = 1, \dots, N\} \tag{2}$$

denote the set of 3D query positions, where $N$ is the number of sampled points in the domain or on the surface. Each point is first mapped to a latent token by a coordinate MLP,

$$\mathbf{z}_0 = \phi(W_0 X + \mathbf{b}_0), \mathbf{z}_\ell = \phi(W_\ell \mathbf{z}_{\ell-1} + \mathbf{b}_\ell), \ell = 1, \dots L, \tag{3}$$

where $\mathbf{z}_\ell \in \mathbb{R}^{N \times d}$ is the hidden embedding at layer $\ell$, $d$ is the embedding dimension, $W_\ell$ and $\mathbf{b}_\ell$ are learnable weights and biases, and $\phi$ is a nonlinearity. We distinguish two input types: volumetric samples on a regular grid and surface samples on the STL. Each type uses its own MLP encoder (non-shared weights), allowing specialization to volumetric and geometric cues. Optional priors such as region flags or surface normals can be concatenated to the coordinates before the MLP.

**Backbone.** AeroFormer decouples the surrounding volumetric domain from the surface geometry and fuses them through cross attention so that volumetric features are conditioned on the current shape. Let $\mathbf{z}^{\mathrm{vol}}$ and $\mathbf{z}^{\mathrm{surf}}$ denote volumetric and surface token sets. Queries are derived from volumetric tokens and keys/values from surface tokens:

$$Q = \mathbf{z}^{\mathrm{vol}} W_Q, \quad K = \mathbf{z}^{\mathrm{surf}} W_K, \quad V = \mathbf{z}^{\mathrm{surf}} W_V, \tag{4}$$

with $W_Q, W_K, W_V \in \mathbb{R}^{d \times C}$. Standard softmax cross-attention scales quadratically with token count and is prohibitive for million-point domains. We therefore adopt a linear-attention formulation: a kernel map $\Psi(\cdot)$ is applied row-wise,

$$\tilde{q}_i = \Psi(q_i), \qquad \tilde{k}_i = \Psi(k_i), \tag{5}$$

and the attention update is computed as

$$\mathbf{z}_s^{\mathrm{vol}} = \frac{\tilde{q}_s^\top \left( \sum_i \tilde{k}_i \otimes v_i \right)}{\sum_j \tilde{q}_s^\top \tilde{k}_j}, \tag{6}$$

where $\sum_i \tilde{k}_i \otimes v_i$ is computed once, reducing the overall complexity to near $O((N_{\mathrm{vol}} + N_{\mathrm{surf}})C^2)$. Residual connections and normalization are applied around each attention block to stabilize training. Stacking such blocks yields geometry-aware volumetric representations that remain scalable without per-shape CFD meshing.

**Volumetric branch encoders.** The volumetric branch samples points on a Cartesian grid and treats them as tokens. After coordinate MLP embeddings, tokens enter cross-attention blocks where queries come from the volumetric tokens and keys/values come from the surface branch. Linear attention [49] is implemented by aggregating $\sum_i k_i \otimes v_i$ and then multiplying by $q$, yielding near-linear compute/memory in sequence length. Residual connections stabilize training and preserve identity mappings, allowing updates of volumetric representations conditioned on geometry.

**Surface/geometry branch encoders.** The surface branch performs area-weighted sampling on the STL to obtain surface points and normals as geometry tokens. Two encoder options are supported: (i) a linear self-attention Transformer as a lightweight, robust baseline; and (ii) a Transolver-style slicing/aggregation encoder that maps points to a small set of learnable physical states, performs attention in state space, and deslices back to points. We default to the slicing encoder and keep the self-attention baseline for general inputs. Either encoder can model surface fields directly or

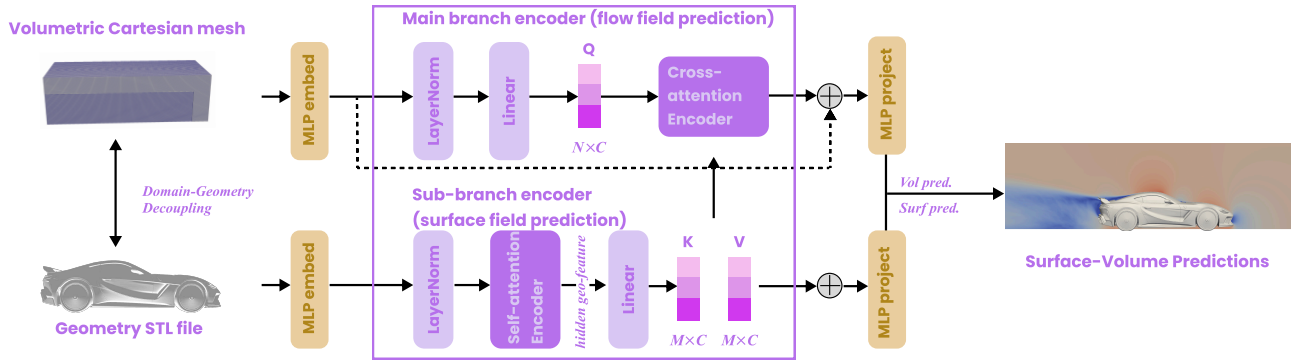



Figure 3. **Architecture of AeroFormer.** A volumetric Cartesian mesh and the corresponding surface geometry (STL) are first mapped into a shared latent space via MLP-based embeddings, achieving domain–geometry decoupling. The main branch encoder operates on volumetric tokens to predict the 3D flow field, while the sub-branch encoder processes surface tokens to predict wall fields; cross-attention from the surface branch provides key features to the volumetric branch. Separate MLP projection heads then produce coupled volumetric and surface predictions.

provide conditional geometry features to modulate the volumetric branch via cross attention.

**Outputs and interface.** Let $t \in \{\mathbf{p}, \mathbf{u}, C_d\}$ denote pressure, velocity, and drag. We expose a single interface

$$S_t(x;\, \mathcal{Q}_t) \;=\; \{\hat{y}_t(\mathbf{q})\}_{\mathbf{q}\in\mathcal{Q}_t}, \tag{7}$$

where $x$ is a standardized STL and $\mathcal{Q}_t$ is the task-specific query set ($\mathcal{Q}_p \in \{\Omega, S\}$ for volumetric or surface pressure, $\mathcal{Q}_u = \Omega$ for velocity, $\mathcal{Q}_d = \{\varnothing\}$ for the scalar drag). Accordingly, $\hat{y}_p = \hat{\mathbf{p}}(\cdot)$, $\hat{y}_u = \hat{\mathbf{u}}(\cdot)$, and $\hat{y}_d = \hat{C}_d$.

**Training objective.** Each task is trained independently. For field outputs ($t \in \{\mathbf{p}, \mathbf{u}\}$) we use a unified squared-error loss

$$\mathcal{L}_t \;=\; \frac{1}{|\mathcal{Q}_t|} \sum_{\mathbf{q}\in\mathcal{Q}_t} \big\|\hat{y}_t(\mathbf{q}) - y_t(\mathbf{q})\big\|_2^2, \tag{8}$$

and for drag we use a scalar regression loss

$$\mathcal{L}_d \;=\; \big(\hat{C}_d - C_d\big)^2. \tag{9}$$

### 3.4. Decision: Integrated Planner.

The decision module is the orchestration center and single source of truth: it parses user intent, enforces regulatory size feasibility, turns multiple requirements into executable edits , evaluations and coordinates the closed loop. During inner iterations it relies only on the surrogate $S$ for fast evaluation; high-fidelity computational fluid dynamics (CFD) is invoked only to confirm the final top-$K$ designs.

**Inputs/Outputs.** Inputs: (i) user intent package $U$, (ii) initial standardized STL pool $C_0$, (iii) feasible set $\mathcal{F}$, (iv) surrogate $S$, and (v) a high-fidelity budget $B_{\mathrm{hf}}$. Outputs: (i) executable edit, (ii) improved candidate pool $C$, and (iii) top - $K$ set for high-fidelity CFD confirmation.

**Feasibility-based acceptance.** We treats regulatory size, drag, and aesthetics as concurrent hard constraints. A candidate is accepted as feasible if and only if it satisfies

$$\Pi(x;U) \;=\; \mathbf{1}\big[x \in \mathcal{F}\big]\, \mathbf{1}\big[\overline{C_d}(x) \le \tau_d\big]\, \mathbf{1}\big[\mathrm{E}(x;U) = 1\big], \tag{10}$$

The feasible set is simply

$$\mathcal{A} \;=\; \{\, x \in C \mid \Pi(x;U) = 1 \,\}, \tag{11}$$

and any $x \in \mathcal{A}$ is considered acceptable for downstream use. Subsequent choices within $\mathcal{A}$ (e.g., which designs to send to high-fidelity CFD) may use additional preferences such as lower drag, but these do not change the feasibility definition in (10).

**From fuzzy intent to executable edits.** To generate concrete edits from fuzzy intent, the planner combines prototype priors and physics feedback. A curated bank of low-drag and high-aesthetic exemplars provides shape priors such as tail taper, windscreen angle, and roof arc. Given a current candidate and its surrogate evaluation (drag and field visualizations), the planner selects a small set actions from this bank that are consistent with the priors and likely to improve feasibility (e.g., lowering drag while preserving style). These actions are sent to the vision module for mesh-level updates.

**Closed loop and stopping.** The planner alternates propose–evaluate–refine with $S$ until one of the following holds: (i) $|\Delta\overline{C_d}| < \epsilon$ for $m$ consecutive rounds (drag converged within tolerance); (ii) $\mathrm{E}(x;U) = 1$ and no further feasible edit improves acceptance; (iii) no candidate in $C$ admits a feasible improvement; or (iv) the time or high-fidelity budget limit is reached. Only the final top-$K$ designs are sent to high-fidelity CFD for confirmation ($K \le B_{\mathrm{hf}}$), and the inner loop never calls high-fidelity solvers. All projections and edits are logged, including size scale factors, surrogate and ranker snapshots, random seeds. This log supports exact replay and cross-team auditability.

### 3.5. Large-scale Vehicle Aerodynamics Dataset

We curate a large-scale vehicle aerodynamics dataset for surrogate training, whose geometries match production vehicle models. ∼50k standardized STL vehicle shapes spanning five body classes, each paired with labels of volumetric

pressure, volumetric velocity, and scalar drag $C_d$. Geometries are obtained from licensed public models and compliant generative pipelines, then normalized as in the vision module (coordinate frame, units, watertightness, mesh health) and projected to class-specific size ranges. Labels are generated under a standard CFD setup; we use a GPU-accelerated lattice-Boltzmann solver to produce fields and $C_d$, with unified boundary conditions, residual-based convergence, and short time-averaging for stability. Full simulation protocol, sampling densities, and quality control criteria are detailed in the supplementary materials. Our mesh-free surrogate design and data organization follow the geometry field decoupling principles shown effective for industrial-scale field prediction in prior work. The dataset was simulated in parallel on 160 NVIDIA RTX 4090 GPUs over roughly two months. The average wall-clock per simulated case is about 4 GPU-hours, for an aggregate of approximately $2 \times 10^5$ GPU-hours to produce the training labels.

## 4. Experiments

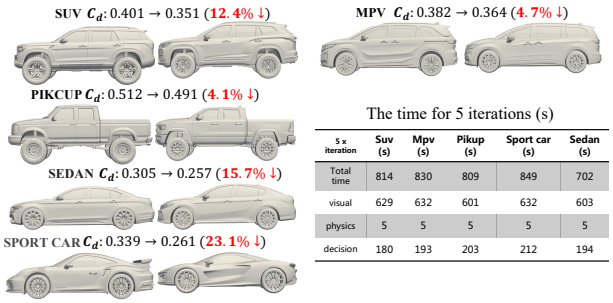
In this section, we provide a detailed analysis of the experimental setup and present the results to assess the effectiveness of our proposed method.

### 4.1. Implementation Details

We train all surrogates on the large-scale vehicle aerodynamics dataset in Sec. 3.5, using approximately 50k simulated samples across five body classes, with geometries and labels following the same standardization and CFD protocol. All surrogates share the AeroFormer backbone with separate heads for pressure, velocity, and drag, trained independently with AdamW and a cosine learning-rate schedule in mixed precision on NVIDIA A100 GPUs. For the vision stack, we employ a commercial diffusion-based text-to-image model (Nano Banana) and Tencent Hunyuan-3D for image-to-3D reconstruction, both run with fixed configurations chosen on a small validation subset. In the decision loop, the default high-fidelity budget is $K = 1$, i.e., only the top-ranked candidate per vehicle is confirmed by CFD; a large multimodal language model (e.g., GPT-5) acts as an automatic aesthetic assessor, providing the binary predicate $\mathrm{E}(x; U)$ for user intent satisfaction. We evaluate drag and field prediction on the held-out test set using relative $L_2$ error and $R^2$, and report aesthetic satisfaction as the fraction of designs with $\mathrm{E}(x; U) = 1$. Additional details on data splits, architectures, CFD settings, and hyperparameters are provided in the supplementary material.

### 4.2. Five-step Styling and Drag Refinement

To illustrate how AeroAgent jointly explores styling and reduces drag in a small fixed number of iterations, we run five propose–evaluate–refine steps on one representative vehicle from each body class (SUV, MPV, pickup, sports car, sedan). Starting from a baseline, the planner applies image-level and mesh-level edits driven by surrogate feedback, and the final design is re-evaluated with high-fidelity CFD.




The time for 5 iterations (s)

| 5 x iteration | Suv (s) | Mpv (s) | Pikup (s) | Sport car (s) | Sedan (s) |
|---|---|---|---|---|---|
| Total time | 814 | 830 | 809 | 849 | 702 |
| visual | 629 | 632 | 601 | 632 | 603 |
| physics | 5 | 5 | 5 | 5 | 5 |
| decision | 180 | 193 | 203 | 212 | 194 |

Figure 4. Five-step refinement across five body classes. For each class (SUV, MPV, pickup, sedan, sports car), we show the baseline shape and the result after five propose–evaluate–refine iterations, together with the CFD drag change (e.g., $C_d$ : $0.401 \rightarrow 0.351$, $12.4\%$ decrease for the SUV). The table on the right reports wall-clock time for five iterations per vehicle (total, vision, physics, decision), with runs completing in 700–850 s and surrogate inference accounting for only about 5 s in total.

As shown in Figure 4, all five vehicles exhibit clear geometric refinement while remaining plausible, and their CFD-verified drag drops consistently: for example, $C_d$ decreases by $12.4\%$ for the SUV, $4.7\%$ for the MPV, $5.1\%$ for the pickup, $15.7\%$ for the sedan, and $23.1\%$ for the sports car. A full five-step loop per vehicle finishes in 700–850 seconds on a single RTX 4090, of which roughly 600–630 s are spent in vision (T2I/I2D and mesh post-processing), about 180–210 s in planning, and only $\sim$5 s in surrogate inference.

Beyond these illustrative cases, we apply the same five-step loop to a larger set of production-style vehicles across the five body classes. Averaged over this set, running only five propose–evaluate–refine iterations per vehicle reduces drag by about 2–12% while cutting high-fidelity CFD calls by 50–80% compared to our baseline workflow, and the GPT-5-based aesthetic predicate $\mathrm{E}(x; U)$ remains unchanged or slightly improved. This confirms that AeroAgent can perform fast visual exploration and reliably drive diverse body styles toward lower drag within a handful of iterations under a strict CFD budget.

### 4.3. Progressive Drag Reduction on a Single Vehicle

To assess whether AeroAgent can reliably steer a design toward lower drag using only surrogate feedback, we conduct a progressive editing experiment on a single coupe (Fig. 5). Starting from a high-drag baseline, the planner applies four rounds of localized edits near the wheels, rear end, and front end; for each step we show the edited 2D image, the standardized 3D model, and a longitudinal pressure slice predicted by AeroFormer and by high-fidelity CFD. The rightmost plot in Fig. 5 reports the drag coefficient $C_d$ for all five

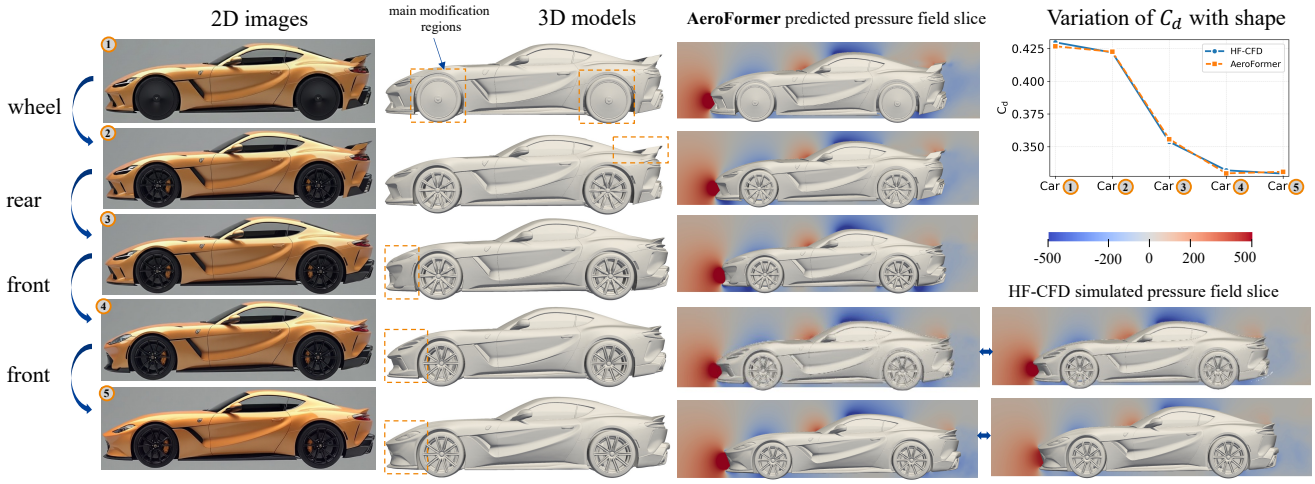



Figure 5. Progressive drag reduction on a single coupe. Car 1–5 show a sequence of planner-generated edits, first around the wheels, then the rear, then the front (orange boxes highlight modified regions). For each step we visualize the 2D rendering, the standardized 3D mesh, and a longitudinal pressure slice from AeroFormer and high-fidelity CFD. The plot on the right shows $C_d$ versus shape index; the AeroFormer and CFD curves nearly coincide and decrease monotonically, indicating that the surrogate faithfully tracks drag changes and that AeroAgent's edits robustly reduce drag.

Table 1. AeroFormer experimental results on pressure field, velocity field and $C_d$. Lower relative $L_2$ and $L_1$ are better for fields, higher $R^2$ is better for $C_d$. The best result is in **bold**, second-best is underlined. "Promotion" refers to our method's relative error reduction or gain w.r.t. the second-best model.

| Model | Pressure field | | Velocity field | | $C_d$ |
|---|---|---|---|---|---|
| | Rel $L_2$ ↓ | Rel $L_1$ ↓ | Rel $L_2$ ↓ | Rel $L_1$ ↓ | $R^2$ ↑ |
| 3D-GeoCA [12] | 0.1853 | 0.0661 | 0.0521 | 0.0274 | 0.9310 |
| Transolver [61] | 0.1797 | 0.1090 | 0.0681 | 0.0377 | 0.9134 |
| Transolver++ [42] | 0.2084 | 0.1259 | 0.0775 | 0.0448 | 0.9012 |
| AB-UPT [5] | 0.2524 | 0.1396 | 0.1021 | 0.0720 | 0.7821 |
| TripNet [11] | 0.1921 | 0.1377 | 0.1007 | 0.0686 | 0.9045 |
| **AeroFormer (Ours)** | **0.1072** | **0.0566** | **0.0279** | **0.0152** | **0.9484** |
| Relative Promotion | 40.3% | 14.3% | 46.4% | 44.5% | 1.86% |

shapes from both AeroFormer and CFD: the curves almost overlap and decrease monotonically, indicating that the surrogate captures both the sign and magnitude of drag changes under small shape modifications, and that the planner consistently drives the design toward lower drag while maintaining a plausible exterior. This supports using the surrogate exclusively in the inner loop and reserving high-fidelity CFD for final confirmation.

### 4.4. Comparison with Aerodynamics Surrogates

To assess AeroFormer as an AI surrogate for vehicle aerodynamics, we compare it against strong baselines on predicting pressure and velocity fields in the cuboid domain surrounding the vehicle. We benchmark Transolver, 3D-GeoCA, and TripNet, all using the same standardized STL geometry, domain, and data splits, with an identical subset of 1,000 training and 100 test cases sampled from Sec. 3.5.

We report volumetric $L_2$ error and $R^2$ on held-out test volumes for five passenger-vehicle classes (sedan, SUV, MPV, pickup, sports car). Across all classes and for both fields, AeroFormer consistently achieves the lowest $L_2$ and highest $R^2$, indicating superior preservation of global flow structure and near-wall detail; Tab. 1 summarizes these gains. Fig. 6 visualizes representative cases. Notably, 3D-GeoCA and TripNet are restricted to volumetric outputs and cannot produce surface fields. Relative to all baselines, AeroFormer better resolves high-gradient regions around the A-pillar, roof separation, and rear wake, with error maps

### 4.5. Performance and Scalability

We next compare the computational efficiency and scalability of AeroFormer with 3D-GeoCA and Transolver on the same CFD task. Table 2 reports GPU memory, wall-clock time, and throughput for training and inference as the number of points in the cuboid domain increases from $10^5$ to $5 \times 10^6$.

On the training side, AeroFormer uses substantially less memory than 3D-GeoCA and slightly less than Transolver, with comparable or lower wall-clock time; its throughput scales nearly linearly with point count and remains highest at $5 \times 10^6$ points. At inference, AeroFormer again offers the best trade-off: it matches or improves upon the baselines in memory, is consistently faster, and achieves the highest throughput across all resolutions. These results indicate that the mesh-free, linear-attention design makes AeroFormer more scalable in both training and deployment, enabling practical use on large 3D domains without prohibitive resource cost.

### 4.6. Generalization Across Vehicle Classes

While the previous comparison used a fixed 1,000/100 subset for all methods, we next examine AeroFormer trained on the full large-scale dataset. We first assess drag prediction across the five body classes (sedan, SUV, MPV, pickup,

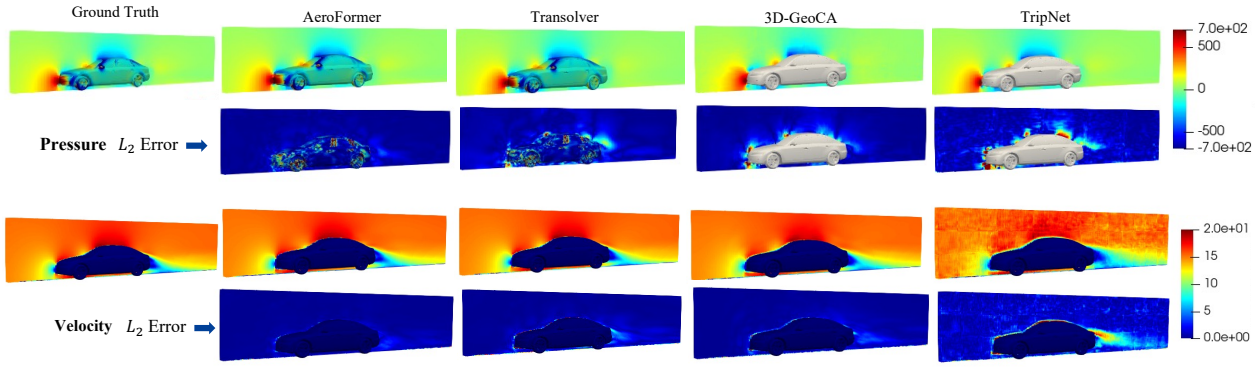



Figure 6. Comparison of aerodynamic surrogates on a sedan. Top rows: pressure slices and $L_2$ error maps. Bottom rows: velocity slices and error maps for CFD, AeroFormer and other methods. AeroFormer best matches CFD, especially in high-gradient regions.

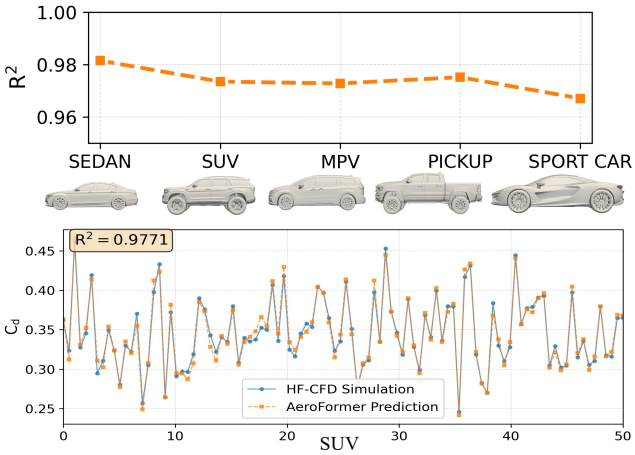



Figure 7. Generalization of AeroFormer across five common passenger-vehicle classes. **Top:** $R^2$ of $C_d$ prediction for sedans, SUVs, MPVs, pickups, and sport cars; all values exceed 0.96, showing consistent accuracy across classes. **Bottom:** drag coefficients for 50 SUVs from the test set, comparing high-fidelity CFD (blue) and AeroFormer predictions (orange). The two curves nearly overlap ($R^2 \approx 0.98$) with no obvious bias, indicating that AeroFormer can be used as a reliable fast surrogate for per-shape drag assessment and early-stage design screening.

Table 2. Training and inference performance on different point counts. Lower memory and time, higher throughput are better.

| Model | Memory (GB) ↓ | | | Time (s) ↓ | | | Throughput (pts/s) ↑ | | |
|---|---|---|---|---|---|---|---|---|---|
| | $10^5$ | $10^6$ | $5 \times 10^6$ | $10^5$ | $10^6$ | $5 \times 10^6$ | $10^5$ | $10^6$ | $5 \times 10^6$ |
| **Training** | | | | | | | | | |
| 3D-GeoCA | 2.1 | 22.1 | – | 0.15 | 0.30 | – | $0.8 \times 10^6$ | $2.1 \times 10^6$ | – |
| AeroFormer | 1.2 | 10.5 | 54.3 | 0.08 | 0.20 | 1.00 | $1.1 \times 10^6$ | $4.2 \times 10^6$ | $4.1 \times 10^6$ |
| Transolver | 1.7 | 13.4 | 70.2 | 0.09 | 0.22 | 1.10 | $1.3 \times 10^6$ | $3.6 \times 10^6$ | $3.4 \times 10^6$ |
| **Inference** | | | | | | | | | |
| 3D-GeoCA | 0.6 | 4.1 | – | 0.1 | 0.13 | – | $1.1 \times 10^6$ | $8.2 \times 10^6$ | – |
| AeroFormer | 0.5 | 2.8 | 7.5 | 0.03 | 0.1 | 0.2 | $4.0 \times 10^6$ | $2.5 \times 10^7$ | $3.3 \times 10^7$ |
| Transolver | 0.5 | 3.1 | 9.3 | 0.04 | 0.1 | 0.3 | $3.5 \times 10^6$ | $1.8 \times 10^7$ | $2.6 \times 10^7$ |

sports car) by computing $R^2$ between surrogate-predicted and high-fidelity CFD $C_d$ on held-out samples. As shown in Fig. 7, $R^2$ remains above 0.96 for all categories, indicating good generalization to diverse passenger-vehicle shapes rather than overfitting to a single class.

To inspect per-shape behavior, we plot the drag coefficients of 50 randomly selected SUVs, comparing AeroFormer and CFD one by one (bottom of Fig. 7). The two curves almost overlap ($R^2 = 0.98$) with no systematic bias across the $C_d$ range, suggesting that AeroFormer is reliable at the individual-vehicle level. In practice, this agreement is sufficient to use the surrogate as a fast screening and ranking tool in early-stage styling, reserving high-fidelity CFD for only a small set of final candidates.

## 5. Conclusion and Limitations

We presented AeroAgent, a unified vision–physics–decision framework for early-stage aerodynamic design under limited CFD budgets. With standardized geometry and a large CFD dataset, AeroFormer provides accurate drag and flow surrogates, while a budget-aware planner enforces size and intent constraints and reserves high-fidelity CFD for a few final candidates. Across five vehicle classes, AeroAgent supports reliable and efficient screening and refinement of candidate shapes. Our framework is trained on five common passenger-vehicle classes and may not generalize to very different geometries (e.g., heavy trucks, buses, or extreme aero concepts) without additional data. Aesthetics and brand intent are estimated by an LLM-based image scorer without user-study validation, so they should be treated as a proxy rather than a substitute for professional design judgment.

## Acknowledgments

We sincerely thank Professor Jianchun Wang from Southern University of Science and Technology for his valuable guidance and support. We also thank Shouyi Liu again for his important contributions to this work. This work is financially supported by the National Natural Science Foundation of China (No. 12572266), National Key Research and Development Program (2024YFF1500600), Yongjiang Talent Program of Ningbo (No. 2022A-242-G), as well as by the High-Performance Computing Centers at Eastern Institute of

Technology, Ningbo, and Ningbo Institute of Digital Twin.

## References


[1] The size and dimensions of the 2023 chevy silverado 1500. https://www.stokestrainor.com/blog/dimensions-2023-silverado-1500, 2023. 8

[2] What are the average dimensions of a car? https://www.nimblefins.co.uk/cheap-car-insurance/average-car-dimensions, 2024.

[3] Car dimensions of all makes with size comparison tools. https://www.automobiledimension.com/, 2025. 8

[4] David E. Aljure, Joan Calafell, Aleix Báez Vidal, and Asensi Oliva. Flow over a realistic car model: Wall modeled large eddy simulations assessment and unsteady effects. *Journal of Wind Engineering and Industrial Aerodynamics*, 147:225–240, 2018. 2

[5] 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. 7, 20, 21

[6] Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024. 24

[7] CJ Baker and NJ Brockie. Wind tunnel tests to obtain train aerodynamic drag coefficients: Reynolds number and ground simulation effects. *Journal of Wind Engineering and Industrial Aerodynamics*, 38(1):23–28, 1991. 2

[8] Vivek D Bhise. *Ergonomics in the Automotive Design Process: Advanced Topics, Measurements, Modeling and Research*. CRC press, 2024. 1

[9] Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022. 2

[10] CarSized. Compare car design and dimensions in a virtual showroom. https://www.carsized.com/, 2025. Interactive tool to compare car dimensions (length, width, height) side-by-side for thousands of models. Accessed 17 Nov 2025. 8

[11] Qian Chen, Mohamed Elrefaie, Angela Dai, and Faez Ahmed. Tripnet: Learning large-scale high-fidelity 3d car aerodynamics with triplane networks. *arXiv preprint arXiv:2503.17400*, 2025. 7, 19, 20

[12] Jingyang Deng, Xingjian Li, Haoyi Xiong, Xiaoguang Hu, and Jinwen Ma. Geometry-guided conditional adaptation for surrogate models of large-scale 3d pdes on arbitrary geometries. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 5790–5798. International Joint Conferences on Artificial Intelligence Organization, 2024. Main Track. 2, 7, 17, 18

[13] Mohamed Elrefaie, Faez Ahmed, and Angela Dai. Drivaernet: A parametric car dataset for data-driven aerodynamic design and graph-based drag prediction. page V03AT03A019, 2024. 2

[14] Mohamed Elrefaie, Florin Morar, Angela Dai, and Faez Ahmed. Drivaernet++: A large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. *NeurIPS*, 2024. 2, 7

[15] Mohamed Elrefaie, Janet Qian, Raina Wu, Qian Chen, Angela Dai, and Faez Ahmed. Ai agents in engineering design: a multi-agent framework for aesthetic and aerodynamic car design. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, page V03BT03A048. American Society of Mechanical Engineers, 2025. 2

[16] Qihang Fan, Huaibo Huang, and Ran He. Breaking the low-rank dilemma of linear attention. In *CVPR*, 2025. 24

[17] Alisa Fortin, Guillaume Vernade, Kat Kampf, and Ammaar Reshi. Introducing gemini 2.5 flash image, our state-of-the-art image model. *Google Developers Blog*, 2025. Accessed 2025-11-XX. 3

[18] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances in neural information processing systems*, 35:31841–31854, 2022. 2

[19] Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. 3

[20] Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention. *Advances in Neural Information Processing Systems*, 37:79221–79245, 2024. 24

[21] HITEM3D. Hitem3d ai 3d generator. https://www.hitem3d.ai/create, 2024. Accessed: 2025-11-20. 22

[22] Seyed Ali Hosseini, Pierre Boivin, Dominique Thévenin, and Ilya Karlin. Lattice boltzmann methods for combustion applications. *Progress in Energy and Combustion Science*, 102: 101140, 2024. 2

[23] Tencent Hunyuan. Hunyuan 3d. https://3d.hunyuan.tencent.com/, 2024. Accessed: 2025-11-20. 22

[24] Iqbal Husain. *Electric and hybrid vehicles: design fundamentals*. CRC press, 2021. 1

[25] Hyper3D. Hyper3d ai mesh generator. https://hyper3d.ai/, 2024. Accessed: 2025-11-20. 22

[26] D. Ibiknle. Average car sizes: Length, width, and height. https://www.neighbor.com/storage-blog/average-car-sizes-dimensions/, 2024. 8

[27] David Ibiknle. Average car sizes: Length, width, and height. https://www.neighbor.com/storage-blog/average-car-sizes-dimensions/, 2024. Blog article reporting average car length, width and height for different size classes, widely cited as a practical reference. Accessed 17 Nov 2025. 8

[28] ANSYS Inc. Driving innovation in automotive aerodynamics with high-fidelity cfd. https://www.ansys.com/blog/accelerating-automotive-cfd-with-next-generation-technologies, 2025. Accessed: 2025-11-21. 1

[29] Suad Jakirlić, Laura Kutej, Daniel Hanssmann, Branislav Basara, and Cameron Tropea. Eddy-resolving simulations of the notchback 'drivaer' model: Influence of underbody geometry and wheels rotation on aerodynamic behaviour. In *SAE 2016 World Congress & Exhibition*, number 2016-01-1602. SAE International, 2016. 2

[30] Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, et al. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. *arXiv preprint arXiv:2506.16504*, 2025. 4

[31] Chenghao Li, Chaoning Zhang, Joseph Cho, Atish Waghwase, Lik-Hang Lee, Francois Rameau, Yang Yang, Sung-Ho Bae, and Choong Seon Hong. Generative ai meets 3d: A survey on text-to-3d in aigc era. *arXiv preprint arXiv:2305.06131*, 2023. 2

[32] Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. Cad-llama: leveraging large language models for computer-aided design parametric 3d model generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18563–18573, 2025. 2

[33] 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. 2

[34] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *arXiv preprint arXiv:2309.00583*, 2023. 2

[35] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 300–309, 2023. 2

[36] Daizong Liu, Yang Liu, Wencan Huang, and Wei Hu. A survey on text-guided 3-d visual grounding: Elements, recent advances, and future directions. *IEEE Transactions on Neural Networks and Learning Systems*, 2025. 2

[37] Hao Liu, Hui Yuan, Junhui Hou, Raouf Hamzaoui, and Wei Gao. Pufa-gan: A frequency-aware generative adversarial network for 3d point cloud upsampling. *IEEE Transactions on Image Processing*, 31:7389–7402, 2022. 2

[38] Ye Liu and Yuntian Chen. Dragsolver: A multi-scale transformer for real-world automotive drag coefficient estimation. In *Forty-second International Conference on Machine Learning*, pages 38102–38118, 2025. 2

[39] Zhen Liu, Yao Feng, Michael J Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. *arXiv preprint arXiv:2303.08133*, 2023. 2

[40] Rainald Löhner, Carsten Othmer, Markus Mrosek, Alejandro Figueroa, and Atis Degro. Overnight industrial LES for external aerodynamics. *Computers & Fluids*, 214:104771, 2021. 2

[41] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9970–9980, 2024. 2

[42] 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. *arXiv preprint arXiv:2502.02414*, 2025. 2, 7, 18, 19

[43] Liam McManus. More with LES on GPUs – 3 high-fidelity CFD simulations that now run while you sleep. https://blogs.sw.siemens.com/simcenter/les-on-gpus/, 2022. Simcenter STAR-CCM+ blog, accessed 2025-03-XX. 2

[44] Mercedes-Benz. Concept iaa: Aerodynamics world champion with cd = 0.19. https://media.mbusa.com/releases/release-94c791070b0a436c9f47e3903ac1741f-mercedes-benz-concept-iaa-intelligent-aerodynamic-automobile, 2015. Accessed: 2025-11-21. 1

[45] Chair of Aerodynamics and Technical University of Munich Fluid Mechanics. Drivaer model geometry. https://www.epc.ed.tum.de/en/aer/research-groups/automotive/drivaer/geometry/, 2024. Accessed: 2024-05-21. 2

[46] Yunlong Ran, Jing Zeng, Shibo He, Jiming Chen, Lincheng Li, Yingfeng Chen, Gimhee Lee, and Qi Ye. Neurar: Neural uncertainty for autonomous 3d reconstruction with implicit neural representations. *IEEE Robotics and Automation Letters*, 8(2):1125–1132, 2023. 2

[47] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Meshsdf: Differentiable iso-surface extraction. *Advances in Neural Information Processing Systems*, 33:22468–22478, 2020. 2

[48] W Rodi. Comparison of les and rans calculations of the flow around bluff bodies. *Journal of wind engineering and industrial aerodynamics*, 69:55–75, 1997. 2

[49] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. *CoRR*, abs/1812.01243, 2018. 4, 23, 24

[50] Siemens Digital Industries Software. Siemens simcenter powers aerodynamic optimization of concept iaa. https://blogs.sw.siemens.com/simcenter/the-most-aerodynamic-car-designed-with-siemens-software-webinar/, 2018. Accessed: 2025-11-21. 1

[51] Binyang Song, Chenyang Yuan, Frank Permenter, Nikos Arechiga, and Faez Ahmed. Surrogate modeling of car drag

coefficient with depth and normal renderings. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, page V03AT03A029. American Society of Mechanical Engineers, 2023. 2

[52] Tripo Studio. Tripo3d ai workspace. https://studio.tripo3d.ai/workspace/generate, 2024. Accessed: 2025-11-20. 22

[53] Mohd Nizam Sudin, Mohd Azman Abdullah, Shamsul Anuar Shamsuddin, Faiz Redza Ramli, and Musthafah Mohd Tahir. Review of research on vehicles aerodynamic drag reduction methods. *International Journal of Mechanical and Mechatronics Engineering*, 14(02):37–47, 2014. 1, 2

[54] Guocheng Tao, Chengwei Fan, Wen Wang, Wenjun Guo, and Jiahuan Cui. Multi-fidelity deep learning for aerodynamic shape optimization using convolutional neural network. *Physics of Fluids*, 36(5), 2024. 2

[55] Zhi Tao, Weiqi Li, Zhendong Guo, Yun Chen, Liming Song, and Jun Li. Aerothermal optimization of a turbine rotor tip configuration based on free-form deformation approach. *International Journal of Heat and Fluid Flow*, 110:109644, 2024. 2

[56] Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025. 2

[57] Tencent Hunyuan3D Team. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details, 2025. 2

[58] Parsa Vatani, Mohamed Elrefaie, Farhad Nazarpour, and Faez Ahmed. Tripoptimizer: Generative 3d shape optimization and drag prediction using triplane vae networks. *arXiv preprint arXiv:2509.12224*, 2025. 3

[59] Chen Wang, Hao-Yang Peng, Ying-Tian Liu, Jiatao Gu, and Shi-Min Hu. Diffusion models for 3d generation: A survey. *Computational Visual Media*, 11(1):1–28, 2025. 2

[60] Ethan Weber, Aleksander Holynski, Varun Jampani, Saurabh Saxena, Noah Snavely, Abhishek Kar, and Angjoo Kanazawa. Nerfiller: Completing scenes via generative 3d inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20731–20741, 2024. 2

[61] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *ICML*, 2024. 2, 7, 18, 19

[62] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 2

[63] Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. *arXiv preprint arXiv:2411.04954*, 2024. 2

[64] Le Xue, Ning Yu, Shu Zhang, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. *arXiv preprint arXiv:2305.08275*, 2023. 18

[65] Raman Yazdani. Steady and unsteady numerical analysis of the drivaer model. 2015. 12, 13

# AeroAgent: A Vision–Physics–Decision Framework for Aerodynamic Vehicle Design

Supplementary Material

## Appendix Contents



## 6. Details of Problem Formulation

Here, we first formalize the design problem that AeroAgent is meant to address. We ground our formulation in real automotive workflows by (i) reviewing the industrial context of early-stage exterior styling and its reliance on high-fidelity CFD and wind tunnel evaluations, (ii) casting this process as a budgeted feasibility problem over a standardized shape space with explicit regulatory and aesthetic constraints, and (iii) mapping this abstraction onto the concrete vision–physics–decision architecture of AeroAgent.

### 6.1. Industrial background.

Early-stage exterior styling for passenger vehicles is inherently multi-objective. Design studios must balance visual appearance, aerodynamic drag, and regulatory size and packaging constraints in a high-dimensional shape space [8, 24, 53]. In current workflows at automotive manufacturers, this process is not a single global optimization run, but an iterative loop between design studios, computer-aided engineering teams, and package engineers. A typical vehicle program explores tens to hundreds of exterior variants across several styling "themes". Each theme undergoes multiple rounds of surface refinement and aerodynamic tuning before the design is frozen, with successive loops modifying the front area, roofline, rear end, underbody and local details such as mirrors or spoilers.

For each round, engineering teams evaluate a subset of candidates using high-fidelity computational fluid dynamics or wind-tunnel tests under standardized conditions. Public case studies and vendor reports indicate that a single production vehicle can easily accumulate tens to low hundreds of geometry variants that are evaluated with high-fidelity simulations over its development cycle. For example, reports on the Mercedes-Benz Concept IAA, a showcase sedan with a drag coefficient of $C_d = 0.19$, describe more than 300 simulation-driven design iterations and on the order of $10^5$–$10^6$ core-hours of high-performance computing before the first physical prototype was built [44, 50]. Industrial simulation platforms likewise describe vehicle programs with "many hundreds" of high-fidelity runs over multiple years, spanning exterior aerodynamics, aeroacoustics, cooling and thermal management [28].

Despite advances in hardware and numerical methods, high-fidelity external aerodynamics simulations remain expensive. For steady Reynolds-averaged Navier–Stokes simulations on full vehicles with tens of millions to hundreds of millions of control volumes, published studies report solver wall-clock times on the order of several hours to a full day per design point on multi-node high-performance computing clusters, not counting geometry clean-up and mesh generation [28]. In practice, many teams still follow an informal "overnight" rule: engineers submit one or a batch of simulation jobs at the end of the day and in-

spect the results the following morning, which effectively yields roughly one simulation-supported design iteration per calendar day [40, 43]. Including computer-aided design adjustments, feature clean-up, mesh generation and post-processing, the end-to-end turnaround for a single high-fidelity design evaluation typically falls in the range of one to several days. At higher fidelities, such as detached-eddy simulations or large-eddy simulations, a single vehicle case can easily require thousands to tens of thousands of core-hours of computation [4, 29], which further restricts how many shapes can be explored within a fixed program timeline.

Under these constraints, styling teams rarely have the computational budget to exhaustively search shape space or to run full high-fidelity simulations for every intermediate edit. Instead, they rely on coarse screening with low-fidelity tools and historical prototypes, as well as experience-driven manual changes, while reserving the most accurate simulations for a comparatively small set of shortlisted designs near the end of the process. This motivates our formulation of early-stage aerodynamic styling as a budgeted feasibility problem with an explicit upper bound on the number of high-fidelity simulation confirmations.

### 6.2. Budgeted feasibility on a shape space.

We formalize early-stage aerodynamic styling as a feasibility problem on a standardized shape space equipped with a simulation budget. Let $\mathcal{X}$ denote the space of watertight, aligned exterior meshes represented as surface files. An initial candidate pool $C_0 \subset \mathcal{X}$ is obtained by combining text- and image-driven three-dimensional generation with existing computer-aided design templates, followed by the normalization pipeline in the main paper, which fixes global scale, pose and wheel placement.

Regulatory and packaging constraints define a feasible subset $\mathcal{F} \subset \mathcal{X}$. In this work, $\mathcal{F}$ encodes hard bounds on overall length, width and height, as well as simple constraints on wheelbase and overhangs where applicable. Formally, we write

$$\mathcal{F} = \{x \in \mathcal{X} : g_j(x) \le 0,\ j = 1, \dots, J\}, \qquad (12)$$

where each function $g_j$ is a geometric constraint that can be evaluated directly from the mesh.

Structured user intent $U$ specifies optional targets on aerodynamics and styling. We model the aerodynamic requirement as a threshold $\tau_d$ on the drag coefficient, and the styling requirement as a binary predicate $\mathrm{E}(\cdot; U) \in \{0, 1\}$ that indicates whether a design is consistent with the user's aesthetic intent (for example, a "sleek fastback sport utility vehicle" or a "boxy multi-purpose vehicle"). In practice, E is implemented via a vision–language model and a small set of prototype designs.

A learned surrogate model $S$ provides fast aerodynamic feedback. Given a shape $x \in \mathcal{X}$, the surrogate returns a scalar drag prediction $\hat{C}_d(x)$ together with field-level quantities such as surface pressure $\hat{\mathbf{p}}(\cdot \mid x)$ and velocity $\hat{\mathbf{u}}(\cdot \mid x)$ on a fixed three-dimensional grid. These predictions are used both for feasibility checks and for gradient-free edit proposals in the inner loop. Finally, the design loop is constrained by a budget $B_{\mathrm{hf}} \in \mathbb{N}$ on the number of high-fidelity simulations that can be run for a given project. All inner-loop iterations must therefore rely solely on the surrogate model $S$, and only the final top-$K$ candidates, with $K \le B_{\mathrm{hf}}$, are evaluated with high-fidelity simulations. Under these definitions, the feasibility problem in the main paper (Problem 1) can be written as: find one or more $x^\star \in \mathcal{X}$ such that

$$\begin{cases} x^\star \in \mathcal{F}, & \\ \overline{C_d}(x^\star) \le \tau_d & \text{(if specified in } U\text{)}, \\ \mathrm{E}(x^\star; U) = 1 & \text{(if specified in } U\text{)}, \end{cases} \qquad (13)$$

subject to the constraint that at most $B_{\mathrm{hf}}$ designs are confirmed by high-fidelity simulations. Here $\overline{C_d}(x)$ denotes the (possibly) time-averaged drag coefficient for shape $x$, and $C_0 \subset \mathcal{X}$ is the initial candidate pool produced by the vision module.

### 6.3. Connection to the AeroAgent architecture.

Within this formulation, the three modules of AeroAgent play distinct roles. The vision module maps user intent $U$ into an initial pool $C_0$ and supports local edits $x \mapsto x'$ within $\mathcal{X}$ via text and image-driven styling operations. The physics module implements the surrogate model $S$, returning drag predictions and field-level quantities that both score candidates and provide flow-based rationales (for example, over-pressure regions or separated wakes). The decision module enforces feasibility $x \in \mathcal{F}$, encodes the simulation budget $B_{\mathrm{hf}}$, and runs the propose–evaluate–refine loop that selects the final top-$K$ candidates for high-fidelity confirmation. In this way, AeroAgent instantiates the abstract budgeted feasibility problem with a concrete vision–physics–decision agent that can operate end-to-end on standardized three-dimensional vehicle representations.

## 7. Details of AeroAgent Architecture

### 7.1. Vision module: detailed pipeline

Figure 8 illustrates the vision module for three typical input modes: (i) pure text, (ii) text plus a two-dimensional image, and (iii) text plus a three-dimensional mesh. In all cases, the module produces standardized, CFD-ready triangulated surface meshes (STL files) together with an operation log in JSON format that records all pre-processing and edits. This section provides additional details beyond the main paper description in Section 3.2.

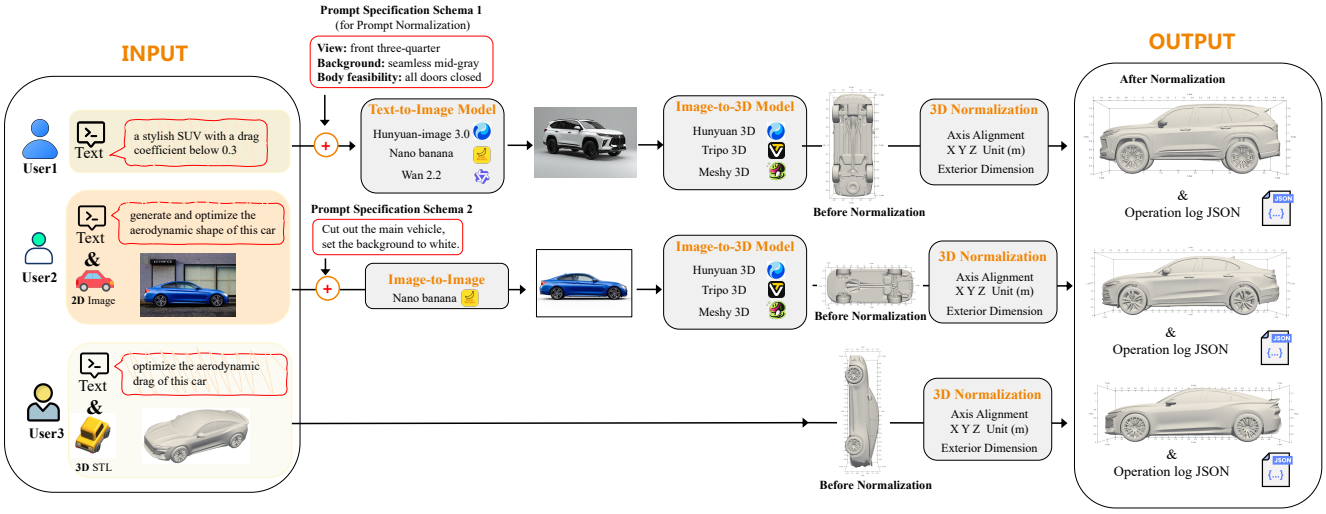



Figure 8. **Vision module for three input modes.** User 1 provides only text; the system generates a standardized image via prompt specification schema 1 and a text-to-image model, then reconstructs a three-dimensional mesh and normalizes it. User 2 provides text and a photograph; the system standardizes the image using prompt specification schema 2 and an image-to-image model before three-dimensional reconstruction. User 3 provides text and a three-dimensional mesh; the system skips image generation and only performs three-dimensional normalization and logging. In all cases the output is a normalized STL mesh and an operation-log JSON file.

### 7.1.1. Design goals and overview

The vision module has three design goals.

1. **Heterogeneous intent support.** It should accept heterogeneous user intent (text, images, sketches, or meshes) and convert all of them into a common standardized three-dimensional representation.
2. **CFD-ready outputs.** The output must be suitable for downstream aerodynamics simulation and surrogates: watertight geometry, consistent coordinate frame and units, and realistic exterior dimensions.
3. **Traceability.** Every operation (generation, cropping, normalization and free-form deformation) is fully logged so that designs can be replayed and audited.

To achieve this, the module is organised in three stages: (1) Intent parsing and prompt specification, which converts natural language into structured visual specifications shared by all back-end generators. (2) Image generation or standardization, which produces a clean, front three-quarter conditioning image under controlled imaging conditions. (3) Image-to-three-dimensional reconstruction and normalization, which reconstructs an STL mesh, aligns it to a canonical coordinate frame, projects its size into class-specific ranges, and writes a paired metadata file. The remainder of this section details how these stages are instantiated for three input modes.

### 7.1.2. Text-only input (User 1)

For a text-only request such as "*a stylish sport utility vehicle with a drag coefficient below 0.3*", we apply the following pipeline.

**Prompt specification schema 1 (text to image).** Natural-language intent is first parsed into a structured visual specification, which we call prompt specification schema 1. It contains:

- **Viewpoint:** fixed front three-quarter view.
- **Background:** seamless mid-gray studio background to avoid clutter and shadows.
- **Body feasibility:** all doors closed, wheels aligned straight, no exaggerated ride height or extreme camber.
- **Body class:** one of the five supported classes (sedan, sport utility vehicle, multi-purpose vehicle, pickup, sports car), inferred from the text or specified explicitly.
- **Negative constraints:** phrases such as "no truncated crops" or "no extreme race-car features", which help avoid unrealistic geometries.

Schema 1 is appended to the user text to form a structured prompt that is passed to a diffusion-based text-to-image model (for example Hunyuan-image 3.0 or Nano Banana). We generate a small batch of images at a fixed resolution.

**Image-to-three-dimensional reconstruction.** The selected standardized image is then passed to a unified image-to-three-dimensional model instantiated with Hunyuan3D 3.0 or similar commercial systems. The model reconstructs a textured three-dimensional asset and exports a triangulated surface mesh (STL). No user-specific camera parameters are required because the viewpoint is fixed by schema 1.

**Three-dimensional normalization and logging.** The raw mesh is normalized as described in the main paper. We perform principal-component-analysis-based axis alignment with nose-direction disambiguation, up-direction and ground-plane fitting using wheel cues, translation to place the origin on the ground beneath the vehicle centre, and uniform unit normalization to metres. Mesh-health checks ensure watertightness and normal consistency; meshes that fail basic checks are discarded. The overall length, width and height $(L, W, H)$ are then projected into the class-specific interval for the predicted body class using minimum-change anisotropic scaling with priority $L \rightarrow H \rightarrow W$. The final output is the normalized STL and a `meta.json` file containing dimensions, transform, body class and a complete operation log describing all upstream steps (prompt, generator version, random seed, and any rejection decisions).

#### 7.1.3. Text plus two-dimensional image (User 2)

For a user who provides a text description together with a photograph (for example "*generate and optimize the aerodynamic shape of this car*" plus a real-world side-view image), the goal is to respect the input image while standardizing imaging conditions for reconstruction. We therefore insert an image-to-image standardization stage.

**Prompt specification schema 2 (image standardization).** We construct a second schema that describes how to clean up the input image before reconstruction. In our implementation schema 2 specifies:

- **Foreground extraction:** cut out the main vehicle instance, using saliency and object segmentation.
- **Background canonicalization:** replace the background with a uniform white or mid-gray colour to remove clutter.
- **Mild retouching:** apply small exposure and white-balance corrections and fill in motion blur if needed.

The original image and schema 2 are fed to an image-to-image diffusion model (Nano Banana) that outputs a cleaned, standardized rendering with the same viewpoint but studio-like lighting and background. This step makes the downstream reconstruction more robust and reduces the variance between user photos.

**Downstream reconstruction and normalization.** The standardized image is then passed to the same image-to-three-dimensional pipeline as in the text-only case, producing a raw STL mesh. The three-dimensional normalization, dimension projection, and metadata logging are identical to those described above, ensuring that generated and user-provided images converge to the same geometric interface.

#### 7.1.4. Text plus three-dimensional mesh (User 3)

For a user who already provides a three-dimensional car mesh (for example a manufacturer CAD export or an STL from another tool), the vision module skips image generation entirely.

**Direct three-dimensional normalization.** The provided mesh is first converted to our internal STL format if needed, then passed directly through the normalization and health-check pipeline: axis alignment, ground-plane fitting, re-centering, unit normalization, mesh integrity checks, and projection of $(L, W, H)$ into the class-specific size ranges. Any free-form deformation edits requested by the user (for example "make the roof slightly lower" or "extend the rear overhang") are applied via the mesh-level edit tools and recorded in the operation log. The resulting STL and `meta.json` file are then forwarded to the physics and decision modules.

#### 7.1.5. Outputs and interface for downstream modules

Regardless of the input mode, the vision module guarantees that the physics and decision modules see a uniform interface:

- a standardized, watertight STL mesh in a canonical coordinate frame (X-forward, Y-left, Z-up; units metres),
- an accompanying `meta.json` file containing global dimensions, wheelbase, body class, normalization transform, mesh-health summary, and a detailed operation log including all image-generation and mesh-edit steps.

Because all geometry passes through the same normalization and logging pipeline, AeroAgent can treat text-only sketches, real-world photographs and existing meshes in a unified manner. This allows the physics module to operate on consistent CFD-ready inputs, and enables the decision module to reason about edits and constraints without relying on input-specific heuristics.

### 7.2. Decision module: agent architecture

The decision module in AeroAgent is implemented as a lightweight agent driven by a large language model (LLM). Its job is to orchestrate the vision and physics modules under a strict budget of high-fidelity computational fluid dynamics (CFD) evaluations, while keeping all design decisions traceable and replayable.

Figure 9 gives an overview of this agent architecture. User intent and context enter on the left, are translated into plans by the planning component, and are executed through a set of domain-specific tools (vision tools, physics tools and bookkeeping tools) that interact with the external environment (for example Paraview, TenFong and file systems). The LLM sits in the centre and produces tool calls based on three sources of information: (i) the current plan, (ii)

short-term and long-term memory, and (iii) prior knowledge about automotive aesthetics and aerodynamics. Tool outputs are fed back into the LLM as observations, closing the loop between perception, memory, planning and action.

Below we describe the four components highlighted in Figure 9 in more detail: profile, memory, planning and action.

#### 7.2.1. Profile: role, tools and constraints

The profile specifies the persistent configuration of the agent: its role, domain knowledge, available tools, and global constraints.

**Role and system prompt.** At the language-model level we use a domain-specific system prompt that casts the agent as an "aerodynamic vehicle design assistant under a limited high-fidelity CFD budget". The prompt (omitted for brevity) defines: (i) the high-level goal (find designs that have low drag, satisfy regulatory size limits and remain visually consistent with the intent), (ii) the available tools (for example "generate two-dimensional images", "select the best reference image", "generate a three-dimensional mesh", "evaluate the surrogate", "apply free-form deformation edits", "check regulatory feasibility", "schedule high-fidelity CFD"), and (iii) safety and logging requirements (never overwrite a mesh without saving a new version, never exceed the CFD budget, always record edit history and surrogate outputs). The profile also fixes project-level hyperparameters such as the high-fidelity budget $B_{\mathrm{hf}}$, the drag threshold $\tau_d$, the list of allowed body classes, and the size ranges that define the feasible set $\mathcal{F}$.

**Tool bindings.** At the code level, tools are registered with a `ToolExecutor` object that stores a mapping from tool names to Python functions and their natural-language descriptions. In the current implementation the following tools are exposed to the agent:

- `Generate2DImages`: call the Tencent Hunyuan image service to generate a batch of standardized front three-quarter vehicle renderings for a requested body type (for example sport utility vehicle, sedan, multi-purpose vehicle, pickup, sports car).
- `Generate3DModel`: call the Tencent Hunyuan image-to-three-dimensional service and post-processing pipeline to obtain a normalized STL mesh from a two-dimensional image.
- `EvaluateSurrogate`: evaluate AeroFormer on a standardized STL and return predicted drag, surface pressure and velocity fields.
- `ApplyFFD`: apply a parameterized free-form deformation edit (for example change roof arc, windscreen angle or diffuser angle) to the current mesh and write out a new STL.
- `CheckFeasibility`: compute global dimensions and check whether the current mesh lies in the feasible set $\mathcal{F}$.
- `ScheduleCFD`: submit a small set of shortlisted candidates to a high-fidelity CFD solver and update the remaining budget $B_{\mathrm{hf}}$.

The natural-language descriptions of these tools are included verbatim in the agent prompt so that the language model can reason about when and how to call them. Adding a new tool (for example a different three-dimensional generator or a post-processing script) only requires registering it with the executor; the planning logic remains unchanged.

#### 7.2.2. Memory: design-session state and logs

The memory component maintains what the agent knows about the current design session and how previous actions have changed the state. It has three layers that correspond to the left side of Figure 9: short-term memory, long-term memory and prior knowledge.

**Short-term memory inside the language model.** Within a single reasoning episode we keep a concise textual history of the agent's actions and observations, which is fed back into the prompt at every reasoning step. For the reasoning-and-acting style controller, this history consists of lines of the form "`Action: Generate2DImages[..., ...]`" and "`Observation: (tool output)`", stored in a list that is re-serialized into the prompt at each step. This short-term memory allows the agent to avoid repeating failed actions, to refine queries (for example adjusting prompts when previous images were rejected), and to synthesize a final answer that refers back to earlier tool calls.

**Structured design-session memory.** Outside the language model, we maintain structured per-design state on disk. Every candidate generated by the agent is written as a standardized STL file together with a `meta.json` metadata record that stores: bounding box and wheelbase, coordinate transform, body-class label, size scaling factors, free-form deformation parameters, surrogate outputs (drag and field-level summaries), and, when used, the identifier of the high-fidelity CFD run that confirmed it. This metadata acts as a persistent memory of the design session and enables exact replay of the propose–evaluate–refine loop for a given candidate.

**Long-term priors and knowledge.** In addition to session-specific memory, the agent has access to static prior knowledge, represented in Figure 9 as Automotive aesthetics knowledge and Aerodynamic priors knowledge. These are implemented as curated prototype libraries rather than as trainable parameters. For example, for aesthetic priors we store templates describing design patterns such as "long

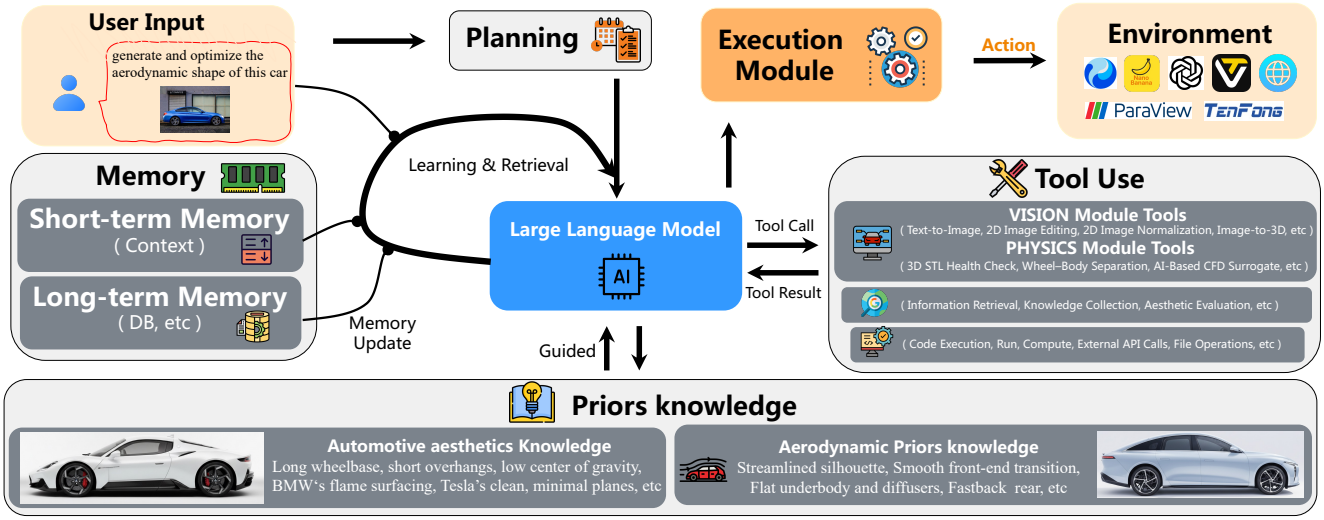



Figure 9. **Decision module and agent architecture.** The user provides a design request (for example "generate and optimize the aerodynamic shape of this car"). The planning component decomposes this request into sub-tasks and queries a large language model (LLM). Guided by prior knowledge (bottom) and memory (left), the LLM issues tool calls to vision and physics tools (right), such as text-to-image generation, two-dimensional image editing, three-dimensional reconstruction, mesh health checks and surrogate evaluation. An execution module carries out these tool calls in the external environment (for example Paraview for visualization, a GPU LBM solver), and returns the tool results to the LLM. Short-term memory stores the current context, long-term memory stores persistent logs and prototypes, and both are updated during the session. In this way the decision module realises a closed loop of planning–tool use–feedback under CFD budget.

wheelbase and short overhangs", "low centre of gravity", "clean, minimal surfacing", and similar cues inspired by typical production vehicles. For aerodynamic priors we store patterns such as "streamlined silhouette", "smooth front-end transition", "flat underbody with diffuser" and "fastback rear". The planner can refer to these priors when deciding which edits are likely to improve drag while remaining stylistically consistent.

### 7.2.3. Planning: from user intent to design edits

The planning component maps user intent and the current design state into a sequence of tool calls and edit decisions. We distinguish between a high-level planner that decomposes the task and a low-level controller that interacts with tools.

**High-level planning.** Given a user request such as "*generate and optimize the aerodynamic shape of this car*" (Figure 9, top left), the planner first asks the LLM to decompose the task into an ordered list of natural-language sub-steps, for example: `["generate SUV reference images", "select best image", "generate 3D mesh", "evaluate drag", "apply low-risk FFD edits", ...]`. This list is represented as a Python list of strings. The execution module then iterates over this list and, for each step, invokes the LLM with the original question, the full plan and the history so far, asking it to focus only on the current step and to propose a concrete action or answer. This hierarchical "plan-and-solve" mode is mainly used for orchestration tasks and for connecting the different modules; it does not need fine-grained access to the surrogate.

**Tool-centric planning with feedback.** For geometry editing and candidate selection inside the design loop we use a tool-centric controller that alternates between "thinking" and "acting". At each step the agent receives: (i) the current design-state summary (including drag estimates and feasibility status), (ii) the user intent package $U$, and (iii) the short-term history of actions and observations. The prompt instructs the LLM to produce:

- a Thought, which is a free-form reasoning trace that analyses the current state and decides what to do next; and then
- an Action, which must either be a tool invocation (for example `ApplyFFD[roof_arc_minus_2deg]`) or a terminal `Finish[final answer]`.

The tool executor runs the requested tool and returns its output as an Observation, which is appended to the history and becomes available to the next step. This loop continues until the model emits a `Finish` action or a predefined maximum number of steps is reached. In this way the agent can iteratively try edits, inspect surrogate feedback (for example which regions of the pressure field indicate separation),

and adjust its plan without human intervention.

**Budget awareness and feasibility checks.** At the planning level, the agent never calls high-fidelity CFD directly. Instead, the controller maintains counters for how many candidates have already been simulated at high fidelity and compares them with the budget $B_{\mathrm{hf}}$. Only when the inner-loop optimization has converged (for example the drag improvement falls below a threshold for several rounds, or the agent has produced a ranked top-$K$ list) does the planner select a small set of candidates and call the `ScheduleCFD` tool. Every time a new candidate is produced, the planner also calls `CheckFeasibility` to ensure that global dimensions remain inside the feasible region $\mathcal{F}$ and discards edits that would violate size limits.

#### 7.2.4. Action: tool-based interaction

The action component specifies how the agent can change the environment or obtain new information. In AeroAgent, all actions are grounded as tool calls, each of which runs deterministic Python code and interacts with external services or local solvers.

**Action targets and production.** The main action targets are: (i) generating new candidates, (ii) editing existing meshes, (iii) evaluating candidates, and (iv) selecting final designs under a budget. Tool calls are produced either by the high-level plan-and-solve executor (for coarse-grained steps) or by the tool-centric controller (for fine-grained edits). In both cases, the LLM emits a textual command of the form `ToolName[tool_input]`, which is parsed by the controller and mapped to a concrete Python function and argument list.

**Action space and impact.** The current action space includes:

- Vision tools for generating two-dimensional images, selecting the best image for three-dimensional reconstruction and calling the three-dimensional generator.
- Geometry tools for applying free-form deformation edits, snapping meshes back to the standardized coordinate system and updating metadata.
- Physics tools for evaluating the surrogate on a candidate, computing scalar and field-level metrics and scheduling high-fidelity CFD runs for final candidates.
- Bookkeeping tools for loading and saving metadata, updating drag rankings and checking feasibility conditions.

All tools are side-effectful at the level of the file system (for example they create images, meshes and metadata) but side-effect-free from the perspective of the LLM: the only information it sees is the textual observation returned by each tool. Through these actions the agent changes the candidate pool, updates the surrogate's evaluation set and consumes high-fidelity simulation budget. By design, the agent cannot directly modify the surrogate or the high-fidelity solver; it can only propose geometries and choose which of them to evaluate. This keeps the overall system robust and makes it straightforward to swap in different surrogates or solvers without changing the decision module.

Overall, the decision module implements a thin, tool-based agent layer on top of AeroAgent's vision and physics components. The agent adds high-level reasoning, budget awareness and interaction with the human designer, while delegating all heavy numerical work to dedicated geometry and simulation tools.

## 8. Dataset

### 8.1. Generating a Wide Range of Car Designs

The goal of our dataset is to provide a large collection of 3D vehicle shapes that (i) cover the main passenger-vehicle segments used in practice, (ii) are geometrically clean and dimensionally realistic, and (iii) expose part-level structure for physics-aware modeling. Unlike purely parametric datasets such as DrivAerNet++ that start from a few CAD templates and morph them in a controlled way (e.g., [14]), our shapes come from a mixture of real-world designs and AI-generated concepts and therefore require a dedicated filtering and standardization pipeline.

**Body classes and design space.** We focus on five common passenger-vehicle body classes that cover a wide range of exterior proportions and flow topologies:

- **Sedan** – three-box configuration with a distinct trunk and relatively low roofline;
- **SUV** – taller, two-box silhouette with a relatively upright rear end;
- **MPV** – high roof and long greenhouse, optimized for interior volume;
- **Pickup** – cab-plus-bed layout with a separated cargo box;
- **Sports car** – low, streamlined profile with an aggressive rear taper.

For each class we collect between $\mathcal{O}(10^3)$ and $\mathcal{O}(10^4)$ shapes, resulting in approximately $5\times10^4$ standardized STL meshes in total. The per-class sample counts are visualized in Fig. 11, which shows that all five classes are well represented and that no single class dominates the dataset.

To give an impression of the variability within each class, Fig. 10 shows a gallery of randomly selected shapes: each row corresponds to one body class and each column shows a different sample after standardization. Even within the same class we observe large variation in roof arc, rear overhang, hood length, wheelbase, and cabin proportion, which is crucial for training surrogates that generalize beyond a single reference model. Fig. 10 provides a qualitative overview of our dataset. Each row shows multi-

| Body class | Length [m] | Width [m] | Height [m] |
|---|---|---|---|
| Sedan | 3.2–5.3 | 1.5–1.9 | 1.5–1.7 |
| SUV | 4.3–5.2 | 1.8–2.0 | 1.6–1.9 |
| MPV | 4.6–5.9 | 1.8–2.1 | 1.6–2.5 |
| Pickup | 5.2–5.9 | 1.8–2.1 | 1.8–2.0 |
| Sports Car | 4.0–5.0 | 1.6–2.0 | 1.1–1.4 |

Table 3. **Indicative exterior dimension ranges for common passenger-vehicle body classes.** Ranges are aggregated from international car-dimension surveys and manufacturer data for currently sold models (see, e.g., [3, 10, 26, 27]). We use slightly widened intervals as soft priors when normalizing the generated shapes, ensuring that sedans remain relatively low and compact, SUVs and MPVs are taller and bulkier, pickups have longer overall length due to the cargo bed, and sports coupés maintain a low, wide stance.

ple standardized meshes from one of the five body classes (sedan, pickup, MPV/minivan, sports car, SUV), illustrating the diversity of exterior proportions and local styling details within and across classes. Fig. 11 reports the exact sample counts for each body class. In total we obtain 53,456 standardized vehicle meshes, comprising 10,904 sedans, 11,325 SUVs, 9,553 MPVs, 11,080 pickups, and 10,594 sports cars. The distribution is deliberately kept nearly uniform — the difference between the smallest and largest class is less than 13% — so that our surrogate does not overfit to any single body type and can learn class-agnostic aerodynamic trends across the full passenger-vehicle design space.

## 8.2. Dimension priors for different body classes

Publicly available vehicle–dimension surveys and manufacturer specification sheets report fairly consistent exterior sizes for mainstream passenger vehicles across body classes [1–3, 26]. We use these statistics to define dimension priors for each body class in our dataset. These priors are not hard regulations, but empirically observed ranges within which the majority of contemporary production vehicles fall. During shape normalization (Sec. 3.2, main paper), we project each generated mesh into the corresponding range for its body class with a minimum-change anisotropic scaling, and discard extreme outliers that would fall far outside any realistic production vehicle.

Table 3 summarizes the indicative ranges we adopt. The numbers are derived by slightly widening the min–max intervals reported in dimension surveys (typically compiled from dozens of current production models) to leave room for design variation while remaining physically plausible. All dimensions refer to overall length, width (excluding mirrors), and height in meters.

### 8.2.1. Sources of 3D vehicle meshes.

We construct the dataset from two complementary sources:

1. **Licensed production-style CAD / mesh models.** We purchase or obtain under license several thousand high-quality meshes from online repositories and design partners. These models closely match real production vehicles in terms of proportions, wheelbase, and underbody layout, and serve as "anchors" for realistic geometry.
2. **AI-generated 3D assets from text- and image-conditioned pipelines.** To increase stylistic diversity, we run a text- and image-to-3D pipeline based on commercial diffusion models and Hunyuan3D, using prompts that target the five body classes while varying style, brand cues, and minor accessories. Generated meshes are subjected to the same standardization and quality checks as real CAD models, and only geometries that pass all filters are retained.

Across both sources we keep only samples for which we can reliably detect four wheels and a plausible ground plane, and for which the bounding-box dimensions fall into conservative class-specific ranges (e.g., sedans between $4.2$–$4.9$ m in length, SUVs between $4.4$–$5.1$ m, etc.). This eliminates caricatured or toy-scale shapes while still allowing for creative variation.

### 8.2.2. Standardizing size and orientation.

Raw meshes obtained from online repositories or generative models exhibit large variation in global pose and scale: the up direction is ambiguous, the wheels may not touch a consistent ground plane, and the overall dimensions can be unrealistically small or large. To make shapes comparable and physically plausible, every asset is passed through a unified normalization pipeline.

First, we recover a canonical coordinate system by performing PCA on the outer body shell and using wheel locations to disambiguate the nose and up directions. We fix the axes so that $x$ points forward, $y$ points to the left, and $z$ points up, and we translate each mesh such that the origin lies on the ground beneath the geometric center of the vehicle.

Second, we enforce realistic exterior dimensions by projecting the raw bounding-box sizes $(L, W, H)$ onto the class-specific ranges summarized in Tab. 3. We apply a minimum-change anisotropic scaling (priority $L \rightarrow H \rightarrow W$) that keeps aspect ratios as close as possible to the original design while ensuring that the normalized vehicle lies within the length/width/height interval of its body class. Meshes that would require extreme scaling factors to enter these ranges are discarded as outliers.

Finally, the rigid transform and per-axis scale factors used during normalization are stored in the metadata for each sample, so that original dimensions can be recovered if needed and downstream surrogates can operate on a consistent, physically meaningful representation of all vehicles.

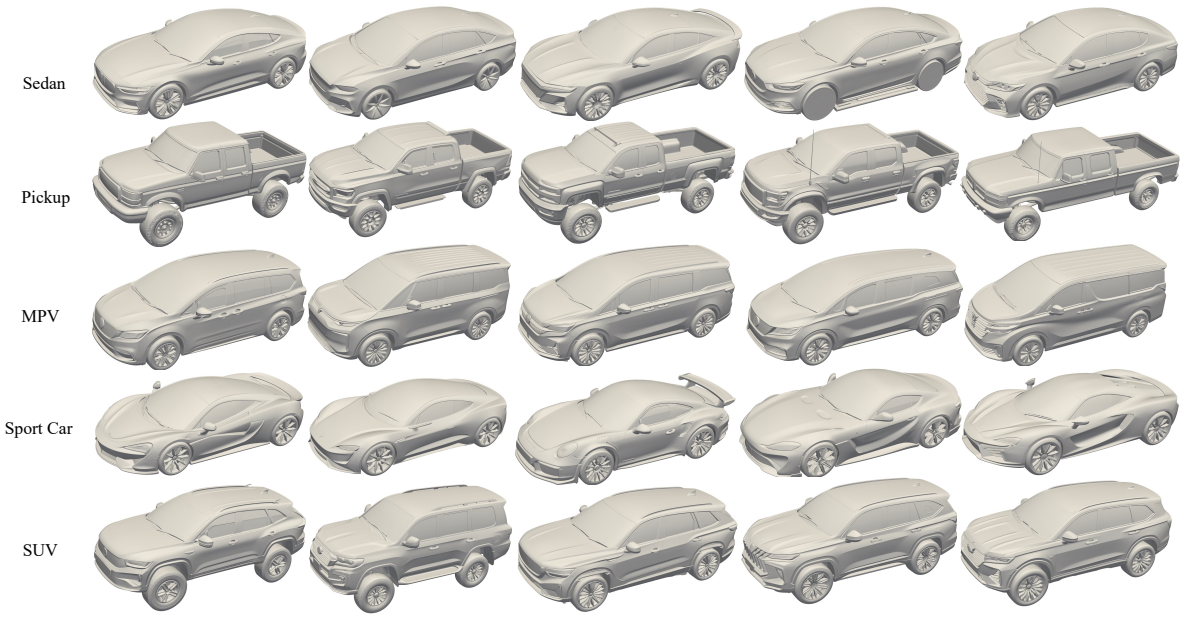



Figure 10. **Qualitative overview of vehicle shapes in our dataset.** We visualize representative standardized meshes from five common passenger-vehicle body classes: sedan, pickup, MPV, sports car, and SUV. Each row corresponds to one body class, and each column shows a different mesh after our alignment, scale normalization, and class-specific size projection. The gallery highlights the large intra-class variation in rooflines, wheelbase, overhangs, and greenhouse proportions, demonstrating that our dataset spans a wide and realistic design space across these body classes.

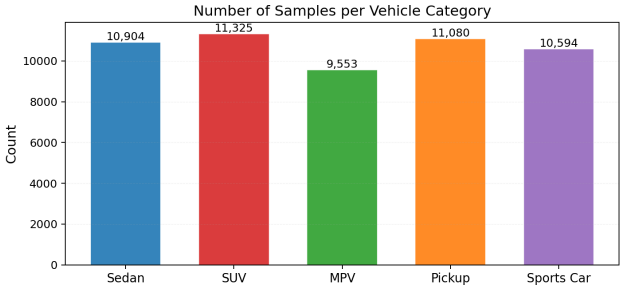



Figure 11. **Number of samples per vehicle body class.** Our dataset contains a total of 53,456 standardized meshes, distributed across five common passenger-vehicle categories: 10,904 sedans, 11,325 SUVs, 9,553 MPVs, 11,080 pickups, and 10,594 sports cars. The counts are intentionally kept within a narrow range to avoid biasing the surrogate toward any single body class.

### 8.2.3. Geometry cleaning and standardization.

Raw meshes vary widely in scale, coordinate frame, and mesh quality. Before any LBM meshing or surrogate training, each candidate passes through a unified standardization pipeline (summarized here for completeness; see Sec. 3.2 of the main paper for details):

1. **Axis alignment and up-direction recovery.** We perform PCA on the outer body shell to obtain the principal axes and use wheel locations to disambiguate nose direction and up direction. The final coordinate system is fixed as $x$-forward, $y$-left, $z$-up.
2. **Ground plane and scale normalization.** We fit a ground plane through the lowest points of the wheels, recenter the vehicle so that the origin lies on the ground under the center of the car, and normalize all meshes to metric units (meters).
3. **Class-specific size projection.** The raw global dimensions $(L, W, H)$ are projected into a regulatory / manufacturing interval for the corresponding body class by a minimum-change anisotropic scaling (priority $L \rightarrow H \rightarrow W$). The applied scale factors are stored in metadata so that physical dimensions can be recovered exactly.
4. **Mesh-health checks.** We run automatic checks for watertightness, flipped normals, self-intersections, isolated components, and extremely thin features that would be under-resolved at our minimum grid size. Candidates that fail any check are either repaired (simple normal flips, small holes) or discarded (severe self-intersections).
5. **Part-level region tagging.** For all standardized meshes we keep separate triangle groups for the upper body shell, underbody / chassis, and each wheel assembly. These region tags are used later when assigning boundary conditions (e.g., rotating-wall vs. stationary wall) in the LBM setup and also allow us to visualize and analyze

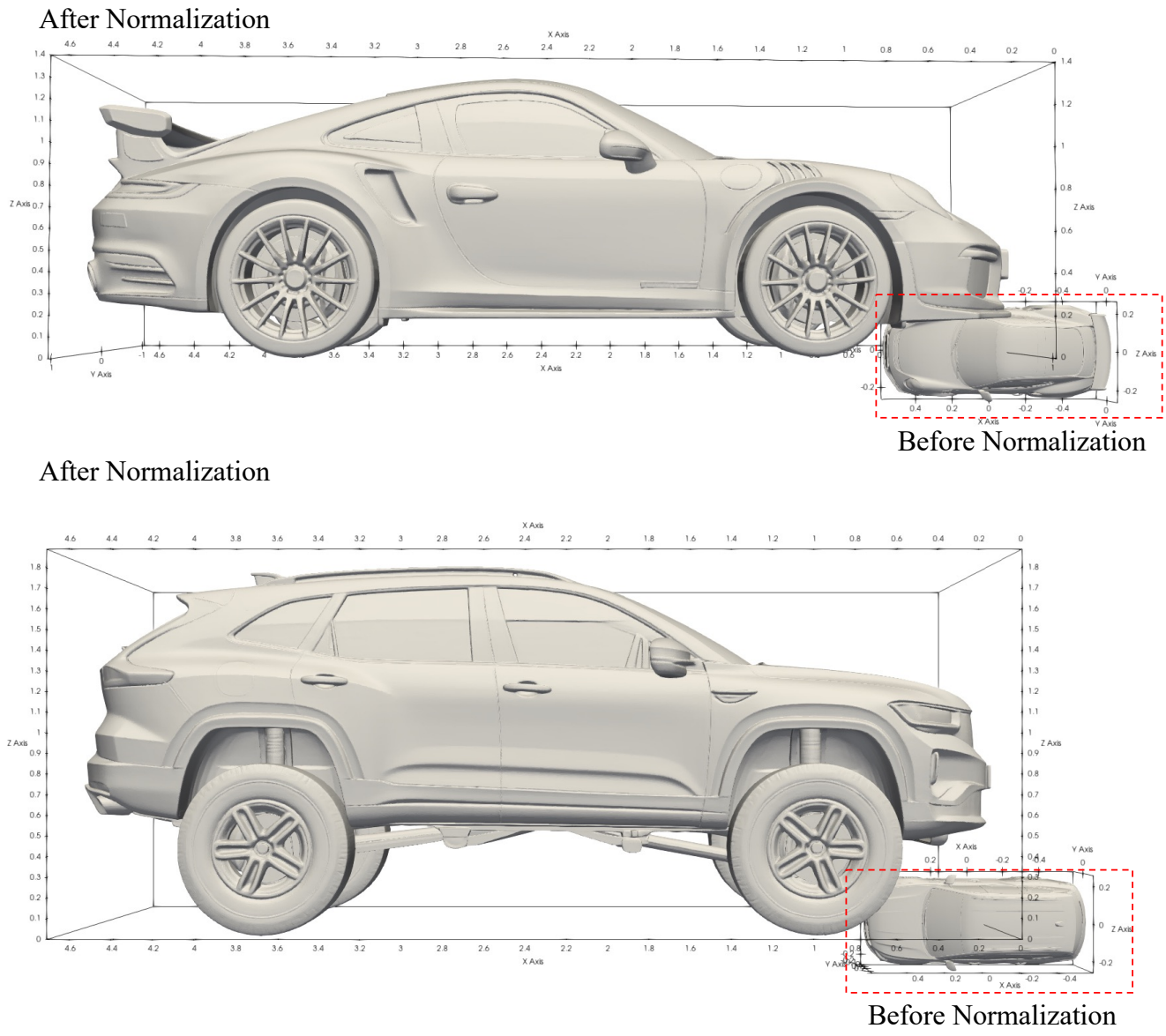



Figure 12. **Effect of our geometric normalization pipeline.** For two representative shapes (a sports car on top and an SUV on the bottom), we compare the raw meshes before normalization (small inset, red dashed box) and the corresponding standardized meshes after normalization (main view). Before normalization, the assets come with arbitrary global scale, orientation, and ground reference, which prevents meaningful comparison across body classes. After applying our alignment and scaling procedure, both vehicles are expressed in a common coordinate frame (forward $x$, left $y$, up $z$), sit on a consistent ground plane, and fall into the class-specific length/width/height ranges defined in Tab. 3.

part-wise diversity.

Each standardized sample is finally packaged as an STL plus a compact metadata JSON record containing the bounding box, wheelbase, coordinate transform, body-class label, region masks, and any subsequent free-form deformation (FFD) edits applied during the design loop.

### 8.3. LBM-based aerodynamic simulations

**Computational budget for dataset generation.** Constructing this dataset required a substantial amount of high-fidelity CFD computation. For each of the 53,456 standardized vehicle meshes we run an LBM simulation of the digital wind-tunnel setup until statistical steady state is reached and sufficient time averaging is performed.

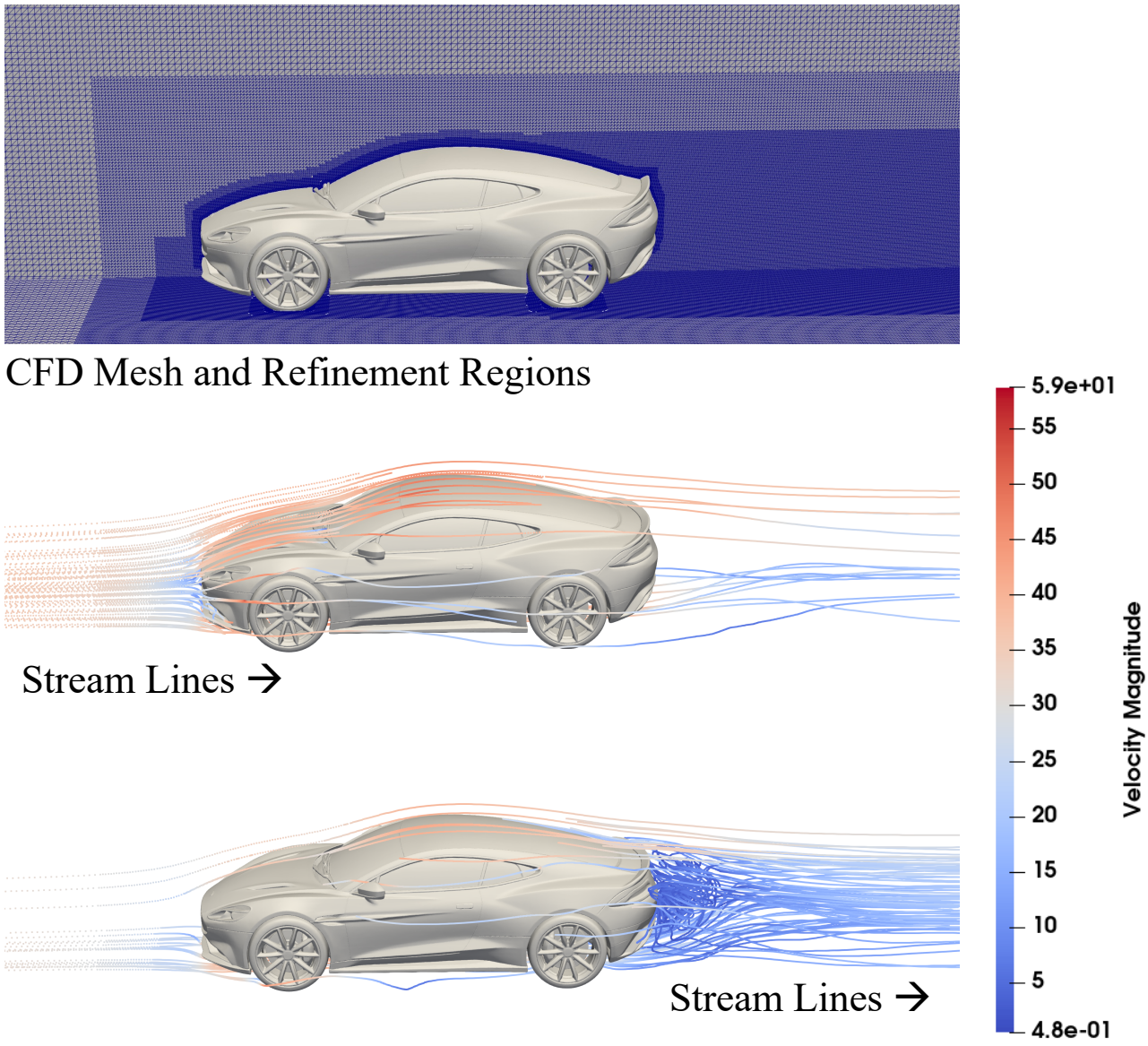



Figure 13. **Digital wind tunnel and LBM mesh refinement.** Top: Cartesian CFD mesh with nested refinement regions used for the lattice–Boltzmann simulations. Finer cells are placed around the vehicle, the underbody, and in the near wake, while coarser cells are used in the far field. Middle and bottom: instantaneous streamlines colored by velocity magnitude for a uniform inflow of $U_\infty = 33\,\mathrm{m\,s^{-1}}$ from front to rear. The figure illustrates the oncoming flow direction, the development of the boundary layer and separation over the roof and rear, and the resolved wake structures behind the vehicle.

On our hardware, a single simulation of one vehicle—including transient spin-up and the $T_{\mathrm{sim}} = 3.0\,\mathrm{s}$ averaging window—takes on average about $4$ GPU-hours on an NVIDIA RTX 4090.

All simulations are executed in parallel on an in-house cluster equipped with up to 160 NVIDIA RTX 4090 GPUs. In total, generating the dataset amounts to more than $5 \times 10^4$ samples $\times\, 4$ GPU-hours per sample, i.e. approximately $2.1 \times 10^5$ GPU-hours of LBM computation. In practice this corresponds to roughly two months of continuous utilisation of 160 GPUs. This computational investment allows us to build a large-scale, high-quality aerodynamic dataset whose flow fields and drag coefficients are anchored to experimental DrivAer data (Sec. 8.4), and to train a single surrogate that generalises reliably across all five common passenger-vehicle body classes (sedan, SUV, MPV, pickup, sports car).

**Digital wind tunnel configuration.** To generate training data for the surrogate, we run high-fidelity lattice–Boltzmann simulations in a virtual wind-tunnel domain that surrounds each standardized vehicle. The global domain is a rectangular box aligned with the canonical vehicle coordinate frame, extending from $x_{\min} = -25\,\mathrm{m}$ upstream to $x_{\max} = 50\,\mathrm{m}$ downstream of the vehicle center, and from $y_{\min} = -25\,\mathrm{m}$ to $y_{\max} = 25\,\mathrm{m}$ laterally. The vertical extent ranges from the moving ground plane at $z_{\min} = 0\,\mathrm{m}$ to $z_{\max} = 30\,\mathrm{m}$. At the inlet we prescribe a uniform free-stream velocity $U_\infty = 33\,\mathrm{m\,s^{-1}}$ aligned with the $+x$ direction, while the outlet uses a pressure-based outflow condition. The top and lateral boundaries are treated as slip walls, and the bottom boundary is modeled as a moving wall with velocity $U_\infty$ (moving ground) to mimic a rolling-road wind tunnel. All vehicle surfaces are no-slip walls. Front and rear wheels are modeled as rotating walls with angular speed chosen to match the prescribed inflow velocity at the contact patch, and their geometry is tagged separately in the mesh.

**Mesh hierarchy and refinement regions.** We employ a block-structured Cartesian mesh with a minimum cell size of $\Delta x_{\min} = 4\,\mathrm{mm}$ around the vehicle. Starting from a coarse background grid that covers the entire domain, we add up to eight nested refinement levels. Refinement boxes are centered on the vehicle, with progressively finer levels covering (i) the entire car plus a margin in all directions, (ii) the underbody and ground region beneath the vehicle, and (iii) the near wake up to approximately one vehicle length downstream. The finest body-fitted level follows the vehicle surface with a fixed number of cells across the near-wall region, and the underbody and wake refinement boxes are chosen such that they extend beyond the projected outline of the car in the horizontal plane. Figure 13 (top) shows a representative cross-section of the resulting mesh and refinement regions.

**Simulation control and configuration file.** All simulation settings are collected in a JSON configuration file (`car_regions.json`), which is shared across shapes and only parameterized by the standardized vehicle STL. This file encodes the domain extents, fluid properties, lattice resolution, simulation duration and averaging window, and the mapping from STL regions (`body`, `wheel_front`, `wheel_rear`) to boundary-condition types (stationary wall, rotating wheel) and local refinement levels. During dataset generation we keep these settings fixed, so that variations in the resulting flow fields and drag primarily reflect changes in vehicle geometry rather than numerical parameters.

Table 4 summarizes the key physical and numerical parameters used in our lattice–Boltzmann simulations.

| Quantity / setting | Value |
|---|---|
| Free-stream velocity $U_\infty$ | $33\,\mathrm{m\,s^{-1}}$ |
| Air density $\rho$ | $1.184\,\mathrm{kg\,m^{-3}}$ |
| Kinematic viscosity $\nu$ | $1.51 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$ |
| Reference pressure $p_\infty$ | 101,325 Pa |
| Domain extents $(x_{\min}, x_{\max})$ | $(-25,\ 50)\,\mathrm{m}$ |
| Domain extents $(y_{\min}, y_{\max})$ | $(-25,\ 25)\,\mathrm{m}$ |
| Domain extents $(z_{\min}, z_{\max})$ | $(0,\ 30)\,\mathrm{m}$ |
| Minimum cell size $\Delta x_{\min}$ | 4 mm |
| Number of grid levels | 9 refinement levels |
| Moving ground | enabled (wall speed $U_\infty$) |
| Wheel rotation | enabled |
| Total simulation time $T_{\mathrm{sim}}$ | 3.0 s |
| Time-averaging window | $t \in [2.0,\ 3.0]\,\mathrm{s}$ |

Table 4. **Key parameters of the LBM simulations used for dataset generation.** All vehicles are simulated in the same digital wind-tunnel setup so that differences in the resulting drag coefficient and flow fields can be attributed to geometry rather than numerical settings.

## 8.4. Anchoring the LBM simulations to DrivAer experiments

To ensure that our lattice–Boltzmann (LBM) simulations provide physically reliable aerodynamic data, we calibrate and validate the digital wind-tunnel setup against an internationally recognised benchmark: the DrivAer generic car model with full-scale wind-tunnel measurements from the Volvo Cars PVT facility (denoted EXP PVT in this work) [65].

The DrivAer model has become a de-facto standard for validating vehicle-aerodynamics methods, because it combines realistic exterior geometry with a fully specified experimental protocol, including moving ground, rotating wheels and detailed underbody options. The PVT data set consists of full-scale tests in a closed-circuit automotive wind tunnel with ground-effect simulation and blockage correction, and is widely regarded as one of the most accurate drag data sets available for DrivAer configurations [65]. We use these measurements as the primary experimental reference for anchoring our high-fidelity CFD (HF–CFD) solver.

**Matching the DrivAer configurations.** For validation we reproduce a set of canonical DrivAer configurations that span different rear-end geometries and underbody treatments. Each case corresponds to a particular combination of fastback vs. notchback rear end, detailed vs. smooth underbody, and closed vs. open wheel houses, following the configurations used in the PVT measurements. In all cases we adopt the full-scale DrivAer geometry, use the same moving-ground and rotating-wheel setup as in our production simulations, and match the Reynolds number of the ex-

periments. The digital wind tunnel and LBM parameters are identical to those used for dataset generation, so that no case-specific tuning is introduced for the validation.

**Comparison with EXP PVT.** Figure 14 compares the drag coefficients obtained from our HF–CFD simulations with the corresponding EXP PVT measurements. Each marker pair represents one DrivAer configuration; black triangles denote the experimental drag coefficients, and red squares denote the time-averaged values from our LBM solver. Across all tested configurations, the curves are nearly indistinguishable: the maximum deviation is below $0.01$ in $C_d$ (i.e. less than $\mathcal{O}(10)$ drag counts), and most cases differ by only a few counts. This level of agreement is comparable to the spread between different RANS and DDES solvers reported in previous DrivAer correlation studies (e.g. [65]), and confirms that our digital wind tunnel reproduces both the absolute drag level and the relative trends across configurations.

Because the same LBM setup is then used unchanged for all vehicle shapes in our dataset, this anchoring step implies that systematic biases in $C_d$ are dominated by the experimental uncertainty of the PVT data rather than by numerical artefacts of our solver. In other words, once anchored to DrivAer, the HF–CFD simulations can be treated as a consistent surrogate for full-scale wind-tunnel testing when generating training labels for AeroFormer.

**Qualitative assessment of LBM solutions.** Figure 15 provides a qualitative overview of the flow fields produced by our LBM solver for several representative body classes. Across all shapes, the simulations reproduce the expected high–pressure stagnation zones at the front bumper, suction peaks over the windshield and roof, and low–pressure regions in the near wake. The velocity contours and streamlines further reveal class–dependent wake topologies (e.g., large recirculation behind MPV and SUV, strong shear layers over the pickup bed, and relatively thin wakes behind the more streamlined coupé and sports car), which is critical for learning geometry–aware surrogates for drag, pressure and wall shear stress.

**Drag statistics over the generated dataset.** Figure 16 summarises the distribution of drag coefficients $C_d$ produced by our LBM solver for all 53,456 vehicles in the dataset, grouped by body class. The observed ranges are consistent with typical values reported for real-world passenger vehicles—streamlined sedans and sports coupés cluster around $C_d \approx 0.25$–$0.32$, taller SUVs and MPVs occupy $C_d \approx 0.32$–$0.40$, and pickups exhibit substantially higher drag with $C_d \approx 0.45$–$0.55$. These statistics confirm that the combination of our geometric priors (Tab. 3) and the anchored LBM simulations (Sec. 8.4) yields a diverse yet realistic distribution of aerodynamic performance across the five body classes, without introducing unphysical low- or high-drag outliers.

## 8.5. Rationale for a canonical operating condition

All simulations in our dataset are performed under a single, canonical operating condition: a uniform, head–on free–stream of $U_\infty = 33\,\mathrm{m\,s^{-1}}$ (approximately $120\,\mathrm{km\,h^{-1}}$) aligned with the vehicle's longitudinal axis, with moving ground, rotating wheels and fixed ambient air properties (Tab. 4). Only the exterior vehicle geometry is varied. This design choice follows standard practice in automotive aerodynamics: when comparing the drag of different vehicle designs, both wind–tunnel tests and CFD campaigns are typically conducted at one or a few reference operating points (highway–representative speeds around $120$–$140\,\mathrm{km\,h^{-1}}$ with moving ground and rotating wheels), so that differences in the measured drag reflect the shape rather than changes in test conditions.

From a fluid–mechanics perspective, the drag coefficient is defined by $F_d = \frac{1}{2}\rho U_\infty^2 C_d A_{\mathrm{ref}}$, and for road vehicles at typical highway speeds the corresponding Reynolds numbers are $\mathcal{O}(10^7)$. In this regime the flow is fully turbulent and the drag coefficient depends only weakly on Reynolds number and Mach number for a given shape and inflow direction. Fixing $U_\infty$ and the inflow direction therefore provides a clean benchmark in which variations in $C_d$ and in the surrounding flow field can be attributed almost entirely to geometric changes.

For learning–based surrogates this has two important advantages. First, it turns the problem into a well–posed supervised mapping from geometry to aerodynamic response under a clearly defined, industry–relevant operating point. The model does not have to disentangle shape effects from variations in freestream speed, density or yaw angle, and can devote its capacity to resolving subtle geometric influences on drag, pressure and wall shear stress across the five body classes. Second, using a single canonical condition dramatically improves data efficiency: instead of sparsely sampling a high–dimensional parameter space of speeds, directions and ambient conditions, we invest our full computational budget into densely covering the shape space with more than $5 \times 10^4$ high–fidelity simulations. As a result, the trained surrogate acts as a reliable "virtual wind tunnel" for comparative design studies at a standard highway condition, which is exactly how drag coefficients are used in industrial vehicle development. Extensions to multiple operating points (different speeds, yaw angles or ride heights) are straightforward in principle and constitute promising future work, but are beyond the scope of this paper.

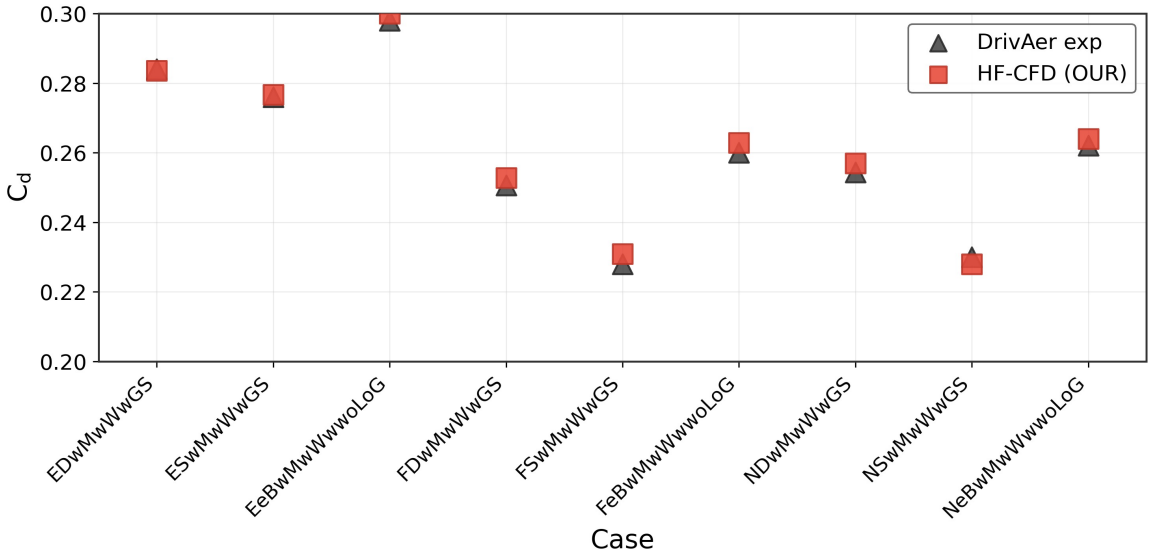



Figure 14. **Anchoring our HF–CFD solver to DrivAer PVT experiments.** Comparison of drag coefficients for several canonical DrivAer configurations. Each case corresponds to a specific combination of rear-end type and underbody/wheel-house configuration. Black triangles show the full-scale experimental drag coefficients measured in the PVT automotive wind tunnel (EXP PVT); red squares show the corresponding time-averaged values from our lattice–Boltzmann simulations. The excellent agreement (differences of at most a few drag counts across all cases) demonstrates that our HF–CFD solver is quantitatively consistent with internationally recognised DrivAer experiments, and justifies using it as a trusted label generator for the large-scale AeroAgent dataset.

### 8.6. Evaluation metrics

Let $\{y_i\}_{i=1}^N$ denote the ground–truth quantities (drag coefficients or field values) on the test set and $\{\hat{y}_i\}_{i=1}^N$ the corresponding predictions of a surrogate model. For field–level quantities (surface pressure, wall shear stress, volumetric velocity), $y_i$ and $\hat{y}_i$ are flattened into vectors over all degrees of freedom of sample $i$. We report four standard error norms and the coefficient of determination.

**Absolute and relative $L_1$ error.** The per-sample $L_1$ error is defined as

$$L_1^{(i)} = \frac{1}{n_i}\sum_{j=1}^{n_i}|y_{ij}-\hat{y}_{ij}|, \tag{14}$$

where $n_i$ is the number of degrees of freedom of sample $i$. To account for the different magnitudes of the targets, we also report a relative $L_1$ error,

$$\text{rel-}L_1^{(i)} = \frac{\sum_{j=1}^{n_i}|y_{ij}-\hat{y}_{ij}|}{\sum_{j=1}^{n_i}|y_{ij}|+\varepsilon}, \tag{15}$$

with a small $\varepsilon$ to avoid division by zero. Unless stated otherwise, the values reported in the tables are averages of $L_1^{(i)}$ or rel-$L_1^{(i)}$ over all test samples.

**Absolute and relative Ł$_2$ error.** Analogously, we define the per-sample $L_2$ error as the root-mean-square (RMS) of the prediction error,

$$L_2^{(i)} = \left(\frac{1}{n_i}\sum_{j=1}^{n_i}\left(y_{ij}-\hat{y}_{ij}\right)^2\right)^{1/2}, \tag{16}$$

and the relative $L_2$ error as

$$\text{rel-}L_2^{(i)} = \frac{\left(\sum_{j=1}^{n_i}\left(y_{ij}-\hat{y}_{ij}\right)^2\right)^{1/2}}{\left(\sum_{j=1}^{n_i}y_{ij}^2\right)^{1/2}+\varepsilon}. \tag{17}$$

The $L_1$ metrics emphasise average deviations, whereas the $L_2$ metrics are more sensitive to larger local errors.

**Coefficient of determination.** For scalar quantities such as the drag coefficient $C_d$ we additionally report the coefficient of determination ($R^2$ score), which measures how much of the variance of the ground–truth values is explained by the model:

$$R^2 = 1-\frac{\sum_{i=1}^{N}\left(C_{d,i}-\hat{C}_{d,i}\right)^2}{\sum_{i=1}^{N}\left(C_{d,i}-\bar{C}_d\right)^2}, \qquad \bar{C}_d = \frac{1}{N}\sum_{i=1}^{N}C_{d,i}. \tag{18}$$

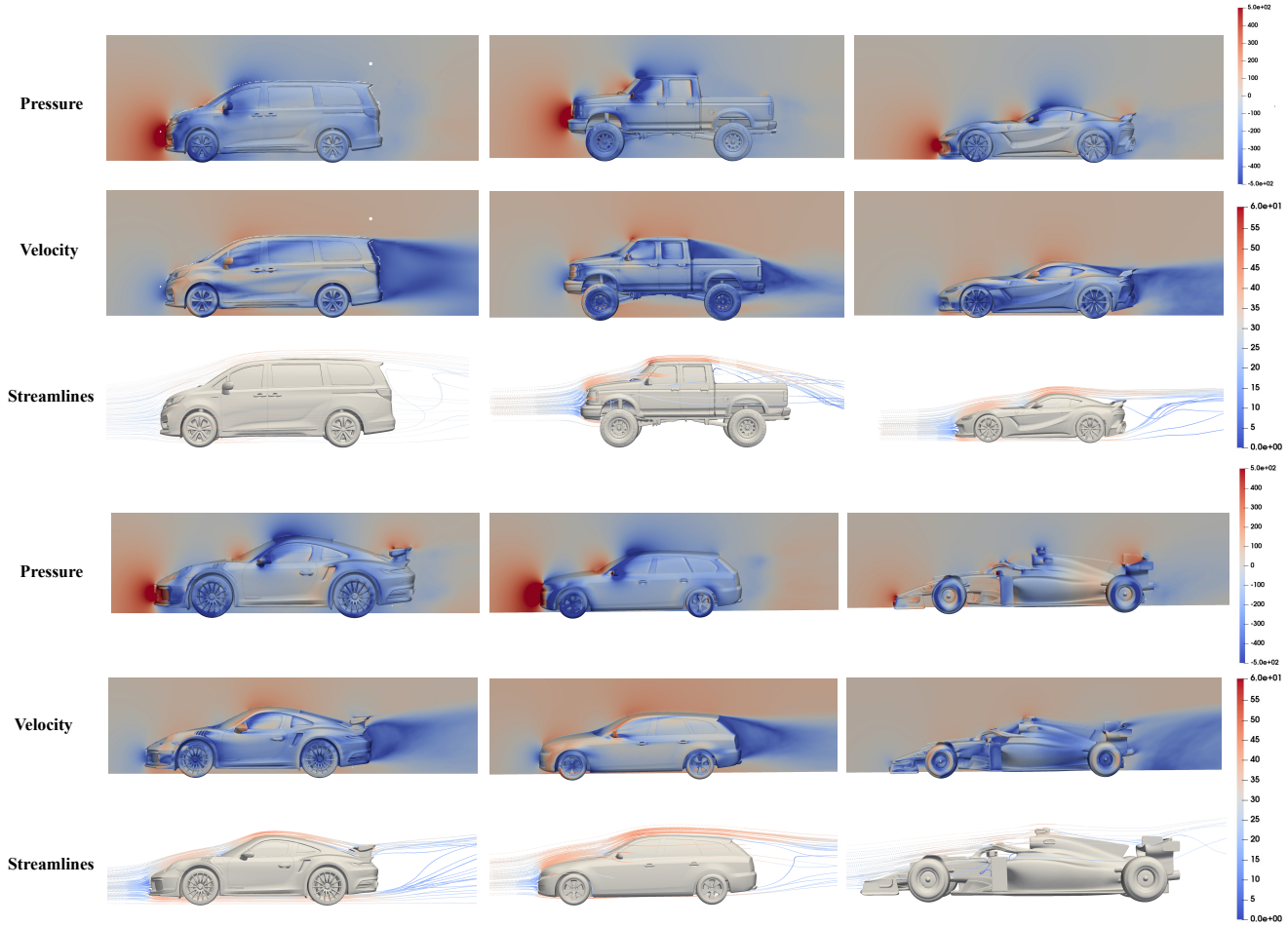



Figure 15. **Representative LBM flow fields for several vehicle body classes.** For each vehicle we show a longitudinal cut through the lattice–Boltzmann solution at a free–stream speed of $U_\infty = 33\,\mathrm{m\,s^{-1}}$: the top block corresponds to an MPV, a pickup, and a low sports car, while the bottom block shows a fastback coupé, an SUV, and an open–wheel race-car style shape. For every configuration, rows from top to bottom display (i) surface pressure and near–field pressure contours, (ii) velocity–magnitude contours, and (iii) streamlines seeded around the vehicle. The stagnation region at the front, accelerated flow over the bonnet and roof, and the separated wake behind the rear end are clearly visible and differ systematically between body styles: bluff MPV and SUV shapes produce large recirculation bubbles, the pickup exhibits a strong separated flow in the cargo–bed region, whereas the fastback and low sports coupé generate more attached flow and a thinner wake. These qualitative features are consistent with well–known trends in experimental vehicle aerodynamics and confirm that our LBM setup resolves the key pressure and velocity structures relevant for drag prediction across a wide range of geometries.

An $R^2$ score of 1 corresponds to a perfect fit, whereas $R^2 = 0$ indicates that the model performs no better than predicting the mean $\bar{C}_d$ of the test set. In all experiments we report both absolute and relative error norms together with $R^2$ to jointly capture average accuracy, magnitude-normalised error and explained variance of the surrogate predictions.

## 9. More Implementation Details

### 9.1. Hardware and training setup.

All surrogates and baselines are trained on a single multi-GPU machine equipped with 8×NVIDIA A100 GPUs (80 GB memory each), using mixed-precision training and data-parallel training across all devices. Unless otherwise stated, we use the same hardware configuration for all models so that training cost is comparable across surrogates and baselines. A typical training run of our main AeroFormer surrogate on the full 50k-sample dataset (Sec. 3.5) completes within a single multi-GPU job, while per-vehicle optimisation in the agent loop (Fig. 5, Fig. 19) runs on a single A100 and takes on the order of tens of minutes (about 20 minutes for a 10-step optimisation), after which only the final 1–2 candidates are evaluated with high-fidelity CFD.

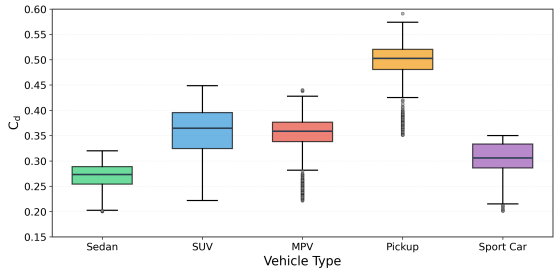



Figure 16. **Distribution of drag coefficients across body classes.** Box plots of the drag coefficient $C_d$ for all vehicles in our dataset, grouped by body class (sedan, SUV, MPV, pickup, sports car). Each box spans the interquartile range (IQR) of $C_d$ within a class, the horizontal line marks the median, whiskers indicate the 5th–95th percentiles, and isolated markers denote remaining outliers. As expected, streamlined sedans exhibit the lowest drag, with medians around $C_d \approx 0.27$ and most samples lying between 0.25 and 0.30. SUVs and MPVs show systematically higher drag (typical medians $C_d \approx 0.35$–$0.38$) due to their taller, bluff rear geometries. Pickups have the highest drag in our dataset, with a median of $C_d \approx 0.50$ and a long upper tail associated with strongly separated flows over the cargo bed. Sports cars fall in between sedans and SUVs: despite their low rooflines, additional aerodynamic devices and downforce-oriented shapes lead to medians around $C_d \approx 0.30$. Overall, the ordering and numerical ranges closely follow values reported for contemporary production vehicles, indicating that our LBM simulations produce physically plausible drag levels across all five body classes.

## 9.2. Network architectures

**Geometry–field decoupling.** For the experiments in this supplement we instantiate a compact two-branch Transformer tailored to our LBM dataset. Each training sample consists of a set of boundary points on the car surface and a set of volumetric query points in the flow domain, both represented only by their Cartesian coordinates $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$. The network predicts pressure at all boundary and volumetric points and can be seen as learning a mapping

$$(\mathcal{X}_{\text{surf}}, \mathcal{X}_{\text{vol}}) \longmapsto (p_{\text{surf}}, p_{\text{vol}})$$

under the fixed operating condition of our digital wind tunnel. To explicitly decouple the vehicle surface from the surrounding volume, we encode these two point sets through separate MLP branches and recombine them only via cross attention.

**Surface and volume embedding MLPs.** Let $X \in \mathbb{R}^{B\times T_{\text{surf}}\times 3}$ and $Y \in \mathbb{R}^{B\times T_{\text{vol}}\times 3}$ denote the batches of boundary and volumetric point coordinates. We process them with two lightweight coordinate MLPs

$$Z_{\text{surf}} = \phi_{\text{surf}}(X) \in \mathbb{R}^{B\times T_{\text{surf}}\times d}, \tag{19}$$

$$Z_{\text{vol}} = \phi_{\text{vol}}(Y) \in \mathbb{R}^{B\times T_{\text{vol}}\times d}, \tag{20}$$

where $\phi_{\text{surf}}$ and $\phi_{\text{vol}}$ are independent $3 \to 64$ MLPs with two residual hidden layers and GELU activation, and $d = 64$ is the embedding width. No hand-crafted geometric features (normals, curvature, etc.) are used; all geometric information is inferred from the raw coordinates, which keeps the interface to the CFD solver simple.

**Cross-attention backbone.** The embedded tokens are then passed through a stack of $L = 3$ cross-attention blocks. Each block operates on the pair $(Z_{\text{surf}}, Z_{\text{vol}})$ and applies (i) self attention on the surface tokens, and (ii) cross attention on the volumetric tokens, followed by residual connections and layer normalisation:

$$Z'_{\text{surf}} = Z_{\text{surf}} + \text{Drop}\big(\text{SelfAttn}(\text{LN}(Z_{\text{surf}}))\big), \tag{21}$$

$$Z'_{\text{vol}} = Z_{\text{vol}} + \text{Drop}\big(\text{CrossAttn}(\text{LN}(Z_{\text{vol}}), \text{LN}(Z_{\text{surf}}))\big). \tag{22}$$

Both attention modules use a kernelised linear attention variant implemented as

$$\text{Attn}(Q, K, V) = \tilde{Q}\,(\tilde{K}^\top V)/\big(\tilde{Q}\,(\tilde{K}^\top \mathbf{1})\big),$$

where $\tilde{Q}, \tilde{K}$ are feature-wise softmax transforms of the queries and keys and the normalisation term is computed once per block. We use a single head ($H = 1$) with head dimension $d_h = 64$ and zero dropout in both the attention and residual paths. This linear-attention design keeps the complexity essentially linear in the total number of points, which is crucial for training on tens of thousands of high-resolution LBM samples.

**Task-specific heads.** After $L$ blocks we obtain updated tokens $Z^{(L)}_{\text{surf}}$ and $Z^{(L)}_{\text{vol}}$. Pressure predictions are produced by two small MLP heads

$$\hat{p}_{\text{surf}} = h_{\text{surf}}\big(Z^{(L)}_{\text{surf}}\big), \qquad \hat{p}_{\text{vol}} = h_{\text{vol}}\big(Z^{(L)}_{\text{vol}}\big), \tag{23}$$

where each head is a $64 \to 64 \to 1$ MLP with two residual hidden layers and GELU activations. The network is therefore symmetric between the surface and volume: all shared computation happens in the cross-attention backbone, while only the final MLP heads are task-specific.

**Model size and design philosophy.** The complete configuration used in our experiments is summarised in Tab. 5. This is by design: our goal is building a geometry-aware surrogate that (i) cleanly separates the car surface from the

| Hyperparameter | Value |
|---|---|
| Input features per point | $(x, y, z) \in \mathbb{R}^3$ |
| Embedding width $d$ | 64 |
| Surface embedding MLP | $3 \to 64$, 2 hidden layers (64), GELU |
| Volume embedding MLP | $3 \to 64$, 2 hidden layers (64), GELU |
| # cross-attention blocks $L$ | 3 |
| Attention type | kernelised linear self/cross attention |
| # attention heads $H$ | 1 |
| Output head (surface pressure) | $64 \to 64 \to 1$, 2 hidden layers, GELU |
| Output head (volume pressure) | $64 \to 64 \to 1$, 2 hidden layers, GELU |

Table 5. **Architecture hyperparameters for the pressure-only AeroFormer variant used in this supplement.** The model is intentionally compact yet expressive enough to capture the coupling between car-surface geometry and volumetric flow fields on our large-scale LBM dataset.

surrounding flow, (ii) scales to millions of points per sample and $> 5 \times 10^4$ samples without excessive memory cost, and (iii) achieves strong accuracy with a simple and reproducible design. As shown in the main paper, this compact model already outperforms more elaborate baselines on both drag and field-level metrics while being easy to train and deploy as a "virtual wind tunnel" in AeroAgent.

**Model scale variants.** To adapt the surrogate capacity to datasets of different sizes and to enable fair comparisons with existing methods, we instantiate four model scales that share the same architecture but differ in width and depth: *Small* (S), *Medium* (M), *Large* (L), and *Extra-Large* (XL). All scales use exactly the same implementation and architectural building blocks; we only vary the embedding width, the number of cross-attention blocks, the number of attention heads, and the MLP depth. This keeps the inductive bias (geometry–field decoupling via cross-attention) fixed while exposing a single degree of freedom to trade accuracy against computational cost. The Medium model is our default configuration, the Small model is used when aligning parameter counts with competing surrogates, and the Large and Extra-Large models are employed in large-scale experiments on the full dataset with more than $5 \times 10^4$ samples, where the additional capacity yields modest but consistent accuracy gains.

Table 6 lists the corresponding hyperparameters. Here $d$ denotes the embedding width (`n_hidden` in the code), $L$ the number of cross-attention blocks (`n_layers`), $H$ the number of attention heads, and $L_{\mathrm{MLP}}$ the number of hidden layers inside each MLP (`mlp_layers`).

### 9.3. GeoCA3D baseline implementation

We use 3D-GeoCA (GeoCA3D) as a strong geometry-aware baseline, following the official implementation of Deng et al. [12] and adapting it to our LBM dataset. GeoCA3D has three components: (i) a pre-trained point-cloud geometry encoder, (ii) an MLP-based CFD surrogate,

| Variant | $d$ | $L$ | $H$ | $L_{\mathrm{MLP}}$ |
|---|---|---|---|---|
| S (Small) | 64 | 3 | 1 | 2 |
| M (Medium) | 128 | 6 | 2 | 4 |
| L (Large) | 256 | 12 | 4 | 6 |
| XL (Extra-Large) | 512 | 16 | 8 | 8 |

Table 6. **Model scale variants of the Transformer surrogate.** All variants share the same two-branch architecture (surface and volume MLPs, $L$ cross-attention blocks, and two task-specific output MLPs), and differ only in embedding width $d$, depth $L$, number of heads $H$, and MLP depth $L_{\mathrm{MLP}}$.

and (iii) geometry-conditioned adaptor blocks that fuse the two.

**Geometry encoder.** As in [12], we adopt the ULIP-PointBERT encoder pre-trained on large-scale 3D object datasets. Given a boundary point cloud $\mathcal{X}_{\mathrm{surf}}$, the encoder produces a 768-dimensional feature vector. We apply a trainable linear projection $W_g \in \mathbb{R}^{768\times512}$ and $L_2$ normalisation to obtain a 512-dimensional geometry code

$$z_g \;=\; \frac{W_g^\top f_{\mathrm{ULIP}}(\mathcal{X}_{\mathrm{surf}})}{\left\|W_g^\top f_{\mathrm{ULIP}}(\mathcal{X}_{\mathrm{surf}})\right\|_2} \in \mathbb{R}^{512}, \tag{24}$$

which is broadcast to all CFD nodes and used as conditioning in the adaptor blocks. The ULIP-PointBERT weights are kept frozen and only $W_g$ is updated.

**CFD surrogate.** The CFD backbone is a lightweight graph MLP closely following the public GeoCA3D configuration. Each CFD node is represented by its coordinates $(x, y, z) \in \mathbb{R}^3$. An encoder MLP maps $\mathbb{R}^3 \to \mathbb{R}^{64}$, followed by four hidden layers of width 64 and a decoder MLP $\mathbb{R}^{64} \to \mathbb{R}$ that predicts scalar pressure at each node. Neighbourhood information is provided by a radius graph; when edge attributes are available we reuse the edge-feature construction from the official code, otherwise message passing operates on node features only.

**Geometry-conditioned adaptor blocks.** To inject global geometry information, GeoCA3D inserts adaptor blocks before the encoder, between each pair of hidden layers, and before the decoder. For node features $x \in \mathbb{R}^{N\times64}$ and broadcast geometry context $z \in \mathbb{R}^{N\times512}$, a GeoCA3D block updates $x$ as

$$x \leftarrow x + \mathrm{FC}_1(\mathrm{LN}(x)) + \mathrm{FC}_2(\mathrm{LN}(x), z) + \mathrm{FFN}(\mathrm{LN}(x)), \tag{25}$$

where $\mathrm{FC}_1$ is a self-conditioned linear layer, $\mathrm{FC}_2$ is a geometry-conditioned linear layer, and FFN is a two-layer feed-forward network with GEGLU activation. All submodules are fully connected layers with dropout and layer

| Component | Hyperparameters |
|---|---|
| Geometry encoder | ULIP-PointBERT (frozen), output dim 768 |
| Geometry projection | Linear $768 \to 512$, $L_2$ normalisation |
| CFD encoder MLP | $3 \to 32 \to 64$, GELU |
| Hidden layers | 4 layers, width 64, GELU |
| CFD decoder MLP | $64 \to 32 \to 1$ |
| GeoCA3D adaptors | 6 blocks along the CFD backbone |
| Optimiser | AdamW, lr $10^{-3}$, weight decay $10^{-6}$ |
| LR schedule | Cosine annealing, 400 epochs |
| Loss | Averaged $L^2$ on node pressures |
| Trainable parts | CFD MLP, adaptors, geometry projection $W_g$ |

Table 7. **GeoCA3D baseline used in our experiments.** We follow the official implementation [12] and adapt it to our LBM dataset, keeping the pre-trained geometry encoder frozen while training the CFD backbone and geometry adaptors.

normalisation, acting as a lightweight alternative to multi-head cross-attention. With four hidden layers we obtain six adaptor insertion points in total.

**Training configuration.** GeoCA3D is trained under the same optimisation settings as our AeroFormer surrogate. We use AdamW with learning rate $10^{-3}$, weight decay $10^{-6}$ and a cosine-annealing schedule over 400 epochs. The loss is the averaged $L^2$ error between predicted and ground-truth node pressures. During training we update the CFD backbone, adaptor blocks and geometry projection $W_g$, while keeping the ULIP-PointBERT encoder fixed, in line with the ULIP philosophy of using pre-trained 3D understanding as a frozen geometric prior [64].

### 9.4. Transolver baseline implementation

We use Transolver [61] as a second strong baseline. Transolver is a Transformer-style surrogate for irregular CFD meshes built around a physics-informed attention mechanism. Instead of applying full self-attention to all $N$ mesh nodes with cost $\mathcal{O}(N^2)$, Transolver introduces a slice–deslice scheme that aggregates nodes into a small set of latent slice tokens, applies attention in this reduced space, and then maps the result back to the original nodes. In our setting, each sample consists of volumetric points $\mathbf{x} \in \mathbb{R}^{N\times 3}$ from the LBM grid, and the model predicts pressure at all $N$ points.

**Input preprocessing.** Each node is represented by its coordinates $(x, y, z)$ only. A small preprocessing MLP maps the raw inputs to a hidden feature of dimension $C$:

$$f_x = \phi_{\text{pre}}(\mathbf{x}) \in \mathbb{R}^{N\times C}, \tag{26}$$

where $\phi_{\text{pre}}$ is a two-layer MLP $3 \to 2C \to C$ with GELU activation. Following the original implementation, we also add a learned placeholder vector $p \in \mathbb{R}^C$ to all nodes, $f_x \leftarrow f_x + p$, which increases expressivity without changing the input dimensionality.

**Physics-informed attention on irregular meshes.** The core module, denoted PhysicsAttention, operates in three stages: slice, attention and deslice. Given $f_x \in \mathbb{R}^{B\times N\times C}$, we first project node features to per-head representations $f_x^{\text{mid}}, x^{\text{mid}} \in \mathbb{R}^{B\times H\times N\times D}$ with $H$ heads and head dimension $D = C/H$. In the slice stage, nodes are softly assigned to $G$ latent slices using a learned projection and a temperature parameter, producing slice weights $w \in \mathbb{R}^{B\times H\times N\times G}$ and slice tokens $s \in \mathbb{R}^{B\times H\times G\times D}$. Standard multi-head self-attention is then applied to the slice tokens, with complexity $\mathcal{O}(G^2)$ per head. Finally, in the deslice stage the updated slice tokens are projected back to node space using the same weights $w$, yielding $f'_x \in \mathbb{R}^{B\times N\times C}$. Overall, the attention cost scales as $\mathcal{O}(NGC + G^2C)$ instead of $\mathcal{O}(N^2C)$, which is crucial for our large, irregular LBM meshes. In all experiments we fix $G = 32$, which is orders of magnitude smaller than $N$.

**Transolver blocks and network depth.** A Transolver block wraps the physics-informed attention in a Transformer-style residual block:

$$f_x \leftarrow f_x + \text{Attn}(\text{LN}_1(f_x)), \tag{27}$$
$$f_x \leftarrow f_x + \text{MLP}(\text{LN}_2(f_x)), \tag{28}$$

where the block MLP is a two-layer GELU MLP $C \to rC \to C$ with expansion ratio $r$. In the final block, an additional layer normalisation and linear head produce the scalar pressure prediction per node. We stack $L = 3$ such blocks with hidden size $C = 64$, which makes Transolver comparable in capacity to our AeroFormer-Medium surrogate.

**Training configuration.** We train Transolver under the same optimisation settings as our own surrogate to enable a fair comparison. We use AdamW with learning rate $10^{-3}$, weight decay $10^{-6}$ and a cosine-annealing schedule. The loss is the averaged $L^2$ error between predicted and ground-truth node pressures over the LBM grid. The main hyperparameters are summarised in Table 8.

### 9.5. Transolver++ baseline implementation

We further include Transolver++ [42] as a strong Transformer baseline. Transolver++ builds on Transolver [61] by introducing an eidetic physics-attention mechanism with Gumbel-softmax slicing and adaptive temperatures, together with an implementation optimised for large-scale parallelism. We use the official code and adapt it to our car-aerodynamics setting, using only point coordinates as inputs and predicting pressure at all volumetric nodes of the LBM mesh.

| Hyperparameter | Value |
|---|---|
| Input features per node | $(x, y, z) \in \mathbb{R}^3$ |
| Hidden dimension $C$ | 64 |
| Number of blocks $L$ | 3 |
| Number of heads $H$ | 1 |
| Number of slices $G$ | 32 |
| MLP expansion ratio $r$ | 2 |
| Preprocessing MLP | $3 \to 128 \to 64$, GELU |
| Block MLP | $64 \to 128 \to 64$, GELU |
| Temperature initialisation | 0.5 (learnable) |
| Optimiser | AdamW, lr $10^{-3}$, wd $10^{-6}$ |
| LR schedule | Cosine annealing |
| Loss | Averaged $L^2$ on node pressures |

Table 8. **Implementation details of the Transolver baseline used in our experiments.** We follow the official implementation [61], using physics-informed slice–deslice attention with $G = 32$ slices and hidden size $C = 64$, which makes Transolver comparable in capacity to our AeroFormer surrogate while retaining its efficiency on large irregular LBM meshes.

**Input preprocessing.** Each volumetric node is represented by its Cartesian coordinates $(x, y, z)$. As in Transolver, Transolver++ first maps coordinates to a hidden feature space via a small two-layer MLP $3 \to 2C \to C$ with GELU activation:

$$f_x = \phi_{\text{pre}}(\mathbf{x}) \in \mathbb{R}^{N \times C}, \tag{29}$$

where $C$ is the hidden dimension. A learned placeholder vector $p \in \mathbb{R}^C$ is added to all nodes, $f_x \leftarrow f_x + p$, acting as a global bias token shared across the mesh. We disable all optional conditioning inputs and unified grid features so that the baseline uses exactly the same inputs as our own surrogate.

**Eidetic physics-attention with adaptive slicing.** The main architectural difference to Transolver is the Physics-Attention-Eidetic module, which replaces the original physics-attention with a more selective and adaptive slice mechanism. Given node features $f_x \in \mathbb{R}^{B \times N \times C}$, the module (1) projects them to per-head representations, (2) predicts a local adaptive temperature for each node and head via a small MLP, (3) computes slice assignments with Gumbel-softmax over $G$ latent slices, and (4) aggregates node features into slice tokens using these assignments. Multi-head self-attention is then applied among the $G$ slice tokens using the optimised scaled dot-product attention operator, and the updated slice tokens are finally mapped back to node space with the same slice weights. This eidetic slice–attention–deslice pipeline yields an overall complexity of $\mathcal{O}(NGC + G^2C)$ with $G \ll N$, while encouraging more distinct physical states than the original soft slicing. In all experiments we set the number of slices to $G = 32$.

**Transolver++ blocks and overall network.** A Transolver++ block wraps the eidetic physics-attention in a standard Transformer encoder block with pre-normalisation:

| Hyperparameter | Value |
|---|---|
| Input features per node | $(x, y, z) \in \mathbb{R}^3$ |
| Hidden dimension $C$ | 64 |
| Number of blocks $L$ | 3 |
| Number of heads $H$ | 1 |
| Number of slices $G$ | 32 |
| MLP expansion ratio $r$ | 1 |
| Preprocessing MLP | $3 \to 128 \to 64$, GELU |
| Block MLP | $64 \to 64 \to 64$, GELU |
| Temperature network | small MLP, learnable bias (init. 0.5) |
| Attention kernel | scaled dot-product attention |
| Optimiser | AdamW, lr $10^{-3}$, wd $10^{-6}$ |
| LR schedule | Cosine annealing |
| Loss | Averaged $L^2$ on node pressures |

Table 9. **Implementation details of the Transolver++ baseline used in our experiments.** We follow the official implementation of Luo et al. [42], enabling eidetic physics-attention with Gumbel-softmax slicing and adaptive temperatures while keeping the hidden dimension moderate ($C = 64$) so that the model has similar capacity to our AeroFormer-Medium and Transolver baselines.

$$f_x \leftarrow f_x + \text{Attn}(\text{LN}_1(f_x)), \tag{30}$$
$$f_x \leftarrow f_x + \text{MLP}(\text{LN}_2(f_x)), \tag{31}$$

where the block MLP is a two-layer GELU MLP $C \to rC \to C$ with expansion ratio $r$. The final block applies an extra layer normalisation and a linear head $C \to 1$ to produce scalar pressure at each node. We use $L = 3$ such blocks and a hidden size of $C = 64$, which makes Transolver++ comparable in capacity to our AeroFormer-Medium surrogate and to the Transolver baseline.

**Training configuration.** Transolver++ is trained under the same optimisation setup as Transolver and our own surrogate: AdamW with learning rate $10^{-3}$, weight decay $10^{-6}$, a cosine-annealing learning-rate schedule, and an averaged $L^2$ loss on node-level pressures. This ensures a fair comparison across all surrogates. The main hyperparameters are summarised in Table 9.

## 9.6. TripNet baseline implementation

We adopt TripNet [11] as a triplane-based baseline. TripNet represents the 3D flow field using three orthogonal feature planes and decodes them with a compact 3D U-Net. This makes the model independent of the original mesh connectivity and allows querying the CFD solution at arbitrary locations. In our setting, TripNet takes a normalised car surface point cloud as input and predicts the pressure field on a fixed Cartesian grid covering the digital wind tunnel.

**Triplane representation.** TripNet maintains three learnable feature planes corresponding to the $xy$-, $yz$- and $xz$-planes,

$$\{\mathcal{E}_{xy}, \mathcal{E}_{yz}, \mathcal{E}_{xz}\}, \qquad \mathcal{E}_{\cdot} \in \mathbb{R}^{1 \times C_{\text{plane}} \times H_{\text{plane}} \times W_{\text{plane}}}, \tag{32}$$

with $H_{\text{plane}} = W_{\text{plane}} = 128$ and $C_{\text{plane}} = 32$. Each plane has its own 2D coordinate MLP that maps projected coordinates to 32-dimensional features; all three use the same architecture (two linear layers with hidden width 64 and ReLU). Given a batch of surface vertices normalised to $[0, 1]^3$, we randomly subsample 16,384 points per sample and project them onto each plane to obtain three dense $128 \times 128$ feature maps. This stage converts the irregular surface point cloud into regular 2D triplane features.

**3D query grid and feature fusion.** TripNet predicts the flow on a regular $H \times W \times D$ grid with resolution $200 \times 40 \times 100$, corresponding to the streamwise, vertical and spanwise directions. We construct a uniform grid $\mathcal{G}$ in $[0, 1]^3$ and affinely map the physical LBM domain to this normalised coordinate system. For each grid point, we compute its projections onto the three planes and sample the corresponding feature maps via 2D bilinear sampling. The three embeddings are then fused by summation, yielding a dense volumetric feature field $\mathcal{F} \in \mathbb{R}^{B \times 32 \times H \times W \times D}$ that serves as input to the decoder.

**3D U-Net decoder.** The fused triplane features are decoded with a lightweight 3D U-Net to produce the pressure field. The U-Net takes 32 input channels and outputs 1 channel, and has two downsampling and two upsampling stages with skip connections. The encoder gradually increases the number of channels from 32 to 256, while the decoder upsamples back to 64 channels before a final $1 \times 1 \times 1$ convolution produces the pressure field on the $200 \times 40 \times 100$ grid.

**Regularisation and training configuration.** Following the official implementation, we add two regularisers on the triplane feature maps: a total-variation term promoting spatial smoothness and an $L^2$ term penalising large feature magnitudes. The total objective is a weighted sum of the data loss and these two regularisers,

$$\mathcal{L}_{\text{total}} = \lambda_{\text{data}} \mathcal{L}_{\text{data}} + \lambda_{\text{TV}} \mathcal{L}_{\text{TV}} + \lambda_{\text{L2}} \mathcal{L}_{\text{L2}}, \qquad (33)$$

with $\lambda_{\text{data}} = 3 \times 10^{-1}$, $\lambda_{\text{TV}} = 10^{-2}$ and $\lambda_{\text{L2}} = 10^{-3}$. The data term is an averaged $L^2$ loss between predicted and ground-truth pressures on the 3D grid.

We train TripNet with the same optimiser and schedule as the other baselines: AdamW with learning rate $10^{-3}$, weight decay $10^{-6}$, a cosine-annealing learning-rate schedule, and minibatches of normalised surface point clouds. The main hyperparameters are summarised in Table 10.

| Hyperparameter | Value |
|---|---|
| Input geometry | surface point cloud in $[0, 1]^3$ |
| Surface samples per car | $128 \times 128 = 16{,}384$ |
| Triplane resolution | $128 \times 128$, 3 planes (xy, yz, xz) |
| Channels per plane | 32 |
| Coordinate MLP per plane | $2 \to 64 \to 32$, ReLU |
| 3D grid resolution | $200 \times 40 \times 100$ |
| Fused feature channels | 32 |
| 3D U-Net encoder | $32 \to 64 \to 128 \to 256$ |
| 3D U-Net decoder | skip connections, final $64 \to 1$ |
| Data loss weight | $3.0 \times 10^{-1}$ |
| TV regularisation weight | $1.0 \times 10^{-2}$ |
| L2 regularisation weight | $1.0 \times 10^{-3}$ |
| Optimiser | AdamW, lr $10^{-3}$, wd $10^{-6}$ |
| LR schedule | Cosine annealing |

Table 10. **Implementation details of the TripNet baseline used in our experiments.** We follow the official triplane CFD network [11], using three $128 \times 128$ planes with 32 channels, a $200 \times 40 \times 100$ query grid, and a compact 3D U-Net decoder.

## 9.7. AB-UPT baseline implementation

We adopt AB-UPT (Anchored-Branched Universal Physics Transformers) as our fifth baseline, following Alkin et al. [5] and the official implementation.[1] AB-UPT is a multi-branch Transformer neural operator with three branches: a geometry branch that encodes the input shape via supernode pooling, and surface and volume branches that exchange information through shared-weight physics blocks and are decoded with anchor attention. This design enables predictions at arbitrary query points while keeping attention cost tractable.

**Geometry encoding via supernode pooling.** The geometry branch takes as input a point cloud representing the vehicle geometry (independent of the CFD mesh) and compresses it into a small set of supernodes. We follow the official supernode pooling scheme: a radius graph with radius $r = 0.25$ in normalised coordinates and maximum degree 32 is built to connect each supernode to its geometric neighbours. Messages from neighbouring points are constructed from relative position features $(\Delta x, \Delta y, \Delta z, \|\Delta\|_2)$, mapped to a $d$-dimensional embedding by a continuous sinusoidal encoder, averaged, and concatenated with an absolute position embedding of the supernode. A linear projection fuses these signals into geometry tokens, which are refined by a single Transformer block with multi-head self-attention and a GELU MLP (expansion factor 4).

**Positional encoding.** All branches use continuous sinusoidal embeddings for input positions. For a 3D coordinate $p = (x, y, z)$, the positional encoder computes a fixed-frequency sincos embedding that splits the total dimension across $x$, $y$ and $z$; if the requested embedding dimension is not divisible by three, the remaining components are zero-padded. These continuous embeddings are shared between geometry, surface and volume branches. On top of this,

[1] Code: https://github.com/Emmi-AI/AB-UPT.

all attention layers use Rotary Positional Embeddings on queries and keys, and surface/volume tokens are further equipped with modality-specific position MLPs $\phi_{\text{surf}}$ and $\phi_{\text{vol}}$ (two-layer GELU MLPs that map $\mathbb{R}^d \to \mathbb{R}^d$), whose outputs are added to the corresponding token embeddings.

**Shared-weight physics branch.** To couple surface and volume tokens while keeping the parameter count moderate, AB-UPT uses a sequence of shared-weight Transformer blocks. The macro layout is defined by a short block string, for example "pscscs": perceiver blocks (p) apply cross-attention from surface and volume tokens to geometry tokens; shared self-attention blocks (s) perform independent self-attention within surface and volume tokens using shared weights; and shared cross-attention blocks (c) couple surface and volume tokens with bidirectional cross-attention, again sharing the attention weights. All physics blocks follow a standard pre-norm Transformer design with LayerNorm, multi-head attention, a GELU MLP (expansion factor 4) and residual connections.

**Modality-specific branches and anchor attention.** After the shared-weight physics blocks, surface and volume tokens are separated and processed in two modality-specific branches that use anchor attention. In each branch, the first $N_{\text{anchor}}$ tokens are treated as anchors and attend to each other with full self-attention, while the remaining tokens are optional query tokens that only attend to anchors. This reduces the complexity from quadratic in the number of tokens to

$$\mathcal{O}\big(N_{\text{anchor}}^2 + N_{\text{anchor}}(N_{\text{surf}} + N_{\text{vol}})\big),$$

while enabling dense querying at arbitrary positions during inference. Each modality-specific block consists of anchor attention followed by a GELU MLP and residual connections (again in pre-norm form). A final linear projection maps surface and volume tokens to scalar pressure values at surface anchors and volume query points. In our experiments we predict pressure only, to match the pressure-only setting used for the other baselines.

**Training setup and hyperparameters.** We follow the AB-UPT setup for automotive CFD datasets [5]. Unless stated otherwise, we use 16k surface and 16k volume anchor tokens per sample (obtained by random subsampling of the CFD mesh), a hidden dimension of $d = 192$, and a total of $L = 12$ Transformer blocks distributed across geometry, shared-weight physics, and modality-specific branches, with three attention heads and an MLP expansion factor of 4. AB-UPT is trained with AdamW, learning rate $10^{-3}$, weight decay $10^{-6}$, a cosine-annealing learning-rate schedule, and mixed-precision training. The loss is the relative $L_2$ error on surface and volume pressures, matching our evaluation metrics (Sec. 8.6); we report relative $L_1$, relative $L_2$ and $R^2$ for a fair comparison with the other baselines.

| Hyperparameter | Value |
|---|---|
| Hidden dimension $d$ | 192 |
| Number of attention heads | 3 |
| Geometry depth | 1 Transformer block |
| Shared physics blocks | 6 (layout "pscscs") |
| Surface branch blocks | 3 AnchorAttention blocks |
| Volume branch blocks | 3 AnchorAttention blocks |
| Geometry pooling radius | $r = 0.25$ (normalised space) |
| Geometry pooling max degree | 32 |
| Surface anchor tokens | 16,384 per sample |
| Volume anchor tokens | 16,384 per sample |
| Positional encoding | continuous sincos + RoPE |
| Modality-specific position MLP | two-layer GELU MLP, width $d$ |
| MLP expansion in all blocks | $4d$, GELU |
| Optimiser | AdamW, lr $10^{-3}$, wd $10^{-6}$ |
| LR schedule | cosine annealing |
| Loss | relative $L_2$ on $p_{\text{surf}}, p_{\text{vol}}$ |
| Precision | mixed FP16/FP32 |

Table 11. **Implementation details of the AB-UPT baseline used in our experiments.** We follow the base configuration of Alkin et al. [5], using a hidden dimension of 192, three-headed attention, and a total of 12 Transformer blocks distributed across geometry, shared physics and modality-specific branches, with 16k surface and 16k volume anchor tokens per sample.

The main hyperparameters used in our experiments are summarised in Table 11.

## 10. Ablation studies

### 10.1. Effect of prompt normalization for 2D images

Before three-dimensional reconstruction, our vision module applies a prompt-normalization stage to two-dimensional conditioning images (Section 3.2, main paper). This ablation isolates the effect of this stage: we keep the image-to-three-dimensional model fixed (Hunyuan3D 3.0), change only whether prompt normalization is applied to the two-dimensional input, and compare the resulting meshes from the perspective of computational-fluid-dynamics (CFD) suitability.

**Setup.** For text-only inputs we generate a batch of front three-quarter vehicle renderings with Hunyuan image. In the no-normalization variant, we pass a direct concatenation of the user text and a minimal body-type description to the text-to-image model. This allows the generator to freely produce "dynamic" automotive images with arbitrary wheel orientations, camera angles and foreground objects. In the with-normalization variant, we insert the prompt specification schema described in Section 3.2 of main paper: we fix a front three-quarter view, enforce a neutral studio background, require all doors to be closed, and add negative phrases that explicitly discourage turned wheels, camera crops and non-car objects. The same image-to-three-

dimensional model (Hunyuan3D 3.0) and the same post-processing pipeline are then applied in both cases to reconstruct STL meshes.

**Qualitative effects.** Figure 17 shows a representative example. Without prompt normalization (top row), the generated image and the resulting three-dimensional mesh exhibit two failure modes that are particularly harmful for CFD:

1. **Wheel orientations unsuitable for CFD.** The generator frequently produces "action-shot" renderings with front wheels turned relative to the body, different steer angles on left and right wheels, or wheels that are not tangent to a consistent ground plane. When reconstructed in three dimensions, these wheels intersect the body or the ground and break the symmetry that is assumed in our digital wind tunnel, leading to spurious underbody cavities and contact artefacts.
2. **Unrealistic non-body structures.** The model often hallucinates additional geometry such as display stands, floor plates, scanning artefacts around the rocker panels, or disconnected pieces under the body. These non-body structures are highlighted in the right-hand mesh in Figure 17 (top) and would be interpreted by the CFD mesher as solid obstacles, severely distorting the flow.

After applying prompt normalization (bottom row), the same text prompt leads to a clean studio rendering with straight wheels, a flat ground plane and no spurious foreground objects. The reconstructed mesh has wheels aligned with the vehicle longitudinal axis, a consistent contact patch with the ground and a clean underbody without hanging or floating parts. In practice we observe that prompt normalization substantially reduces the number of generated meshes that must be discarded or manually repaired before meshing.

**Discussion.** This ablation highlights that, even when the underlying image-to-three-dimensional model is fixed, the quality and robustness of the resulting geometry depend strongly on how two-dimensional prompts are specified. Prompt normalization acts as a lightweight, model-agnostic filter: it does not make the generator more powerful, but it steers it away from photographically appealing yet physically implausible scenes and towards CFD-compatible assets. By reducing wheel-orientation artefacts and eliminating non-body structures, it yields a more homogeneous, simulation-ready dataset for training AeroFormer and for running automated design loops with AeroAgent.

## 10.2. Choice of image-to-three-dimensional model

Our vision module relies on an image-to-three-dimensional model to reconstruct meshes from standardized two-dimensional conditioning images (Section 3.2 of the main paper). Before fixing a default choice for large-scale data generation, we compare several publicly available systems under identical conditions and select the one that yields the most reliable, simulation-ready geometry.

**Setup.** We fix a single standardized input image of a production-style sport utility vehicle in front three-quarter view (Figure 18, left). The image is produced by the same prompt-normalization pipeline used in our dataset generation (neutral studio background, straight wheels, no foreground clutter). We then reconstruct a three-dimensional mesh using four different image-to-three-dimensional engines: **Hunyuan 3D 3.0** [23], **Rodin 3D Gen-2 V 1.8** [25], **Tripo 3D** [52], **Hitem 3D** [21]. For all systems we disable texture rendering and export only the triangulated surface. Each output mesh is then passed through the same normalization and mesh-health pipeline described in Appendix 7.1, so that differences reflect the quality of the reconstruction rather than downstream processing.

**Qualitative comparison.** Figure 18 shows the resulting meshes rendered from the same viewpoint. We focus on geometry details that are critical for subsequent aerodynamic simulation: front fascia shape, wheel-house and tyre geometry, continuity of the underbody and the absence of spurious structures.

- **Hunyuan 3D 3.0** [23]. The reconstructed mesh (top right) reproduces the overall proportions and the local front-end geometry with high fidelity. The grille, bumper and headlamp volumes are well separated and aligned with the body panels. Wheel arches and tyres have realistic curvature, and no stray geometry appears under the body. The mesh passes our watertightness and normal-consistency checks without manual repair.
- **Rodin 3D Gen-2 V 1.8** [25]. The global shape is acceptable, but local details around the grille and bumper are softened and slightly collapsed. The front fascia appears "flattened"; the grille opening and emblem are merged into a single bulged patch, and sharp feature lines are washed out. This loss of detail is highlighted by the dashed region in Figure 18.
- **Tripo 3D** [52]. The Tripo 3D mesh has a reasonable silhouette but exhibits overly smoothed surfaces and thickened features. Bumper and lower intake regions are rounded, the headlamp volumes are barely distinguishable, and the front wheels blend into the body. In several test images we also observe small gaps and self-intersections near the underbody, which require additional cleaning before computational fluid dynamics.
- **Hitem 3D** [21]. The reconstruction from Hitem 3D introduces the largest geometric artefacts. The grille and licence-plate area are over-extruded, producing an unnaturally protruding block at the front end. Panel seams are

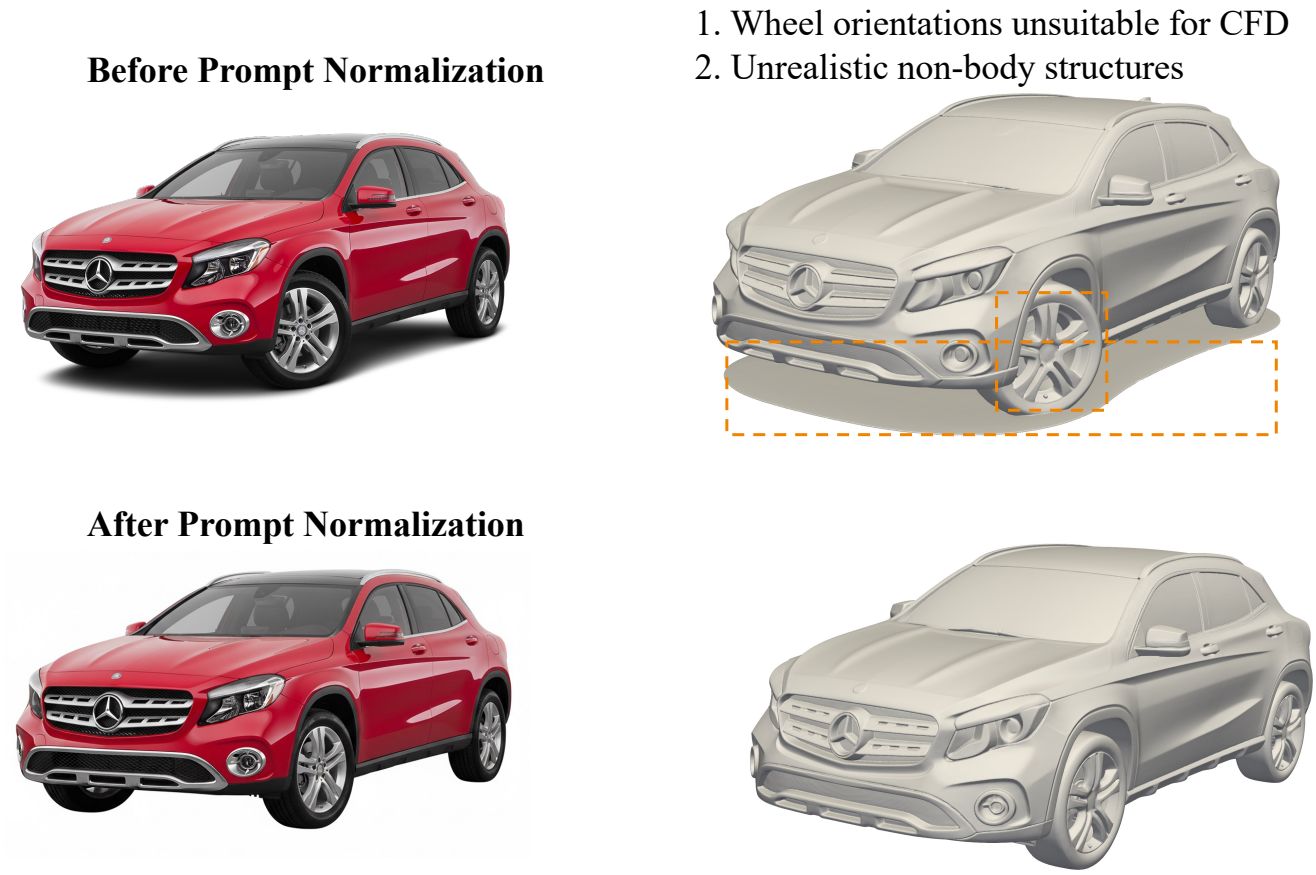



Figure 17. **Effect of prompt normalization on 2D conditioning images.** Top: without prompt normalization, the same text prompt often produces images with turned wheels and additional non-body structures (for example floor plates and artefacts under the sills), which lead to three-dimensional meshes that are unsuitable for CFD. Bottom: with prompt normalization, the generator produces a neutral studio view with straight wheels and no extra structures, resulting in a clean, CFD-ready mesh.

misplaced, some tyre volumes intersect the ground unevenly, and we occasionally observe detached or floating pieces near the bumper. These artefacts are particularly problematic for meshing and would significantly distort the simulated flow.

Overall, Hunyuan 3D 3.0 consistently produces the most faithful reconstructions across our test set: it preserves stylistic cues such as brand-specific front-end design, generates clean wheel and underbody geometry, and passes mesh-health checks with little or no manual intervention. The other engines either oversmooth important features or introduce non-physical geometry around the front fascia and underbody.

**Choice of default engine.** Based on this ablation we adopt Hunyuan 3D 3.0 as the default image-to-three-dimensional engine for all subsequent dataset generation and experiments in AeroAgent. Using a single, high-quality reconstruction model simplifies the normalization pipeline, reduces the number of meshes that fail mesh-health checks, and yields a more homogeneous, CFD-ready dataset for training AeroFormer and for running automated design loops.

### 10.3. Choice of linear attention in AeroFormer

AeroFormer relies on a linear-attention variant to scale cross-attention to millions of volumetric and surface tokens. In the main experiments we adopt the "Efficient Attention" formulation of Shen et al. [49], instantiated as the `attn_type = linear` kernel in our implementation. To verify that this choice is appropriate for our setting, we replace the attention kernel with three alternative linear-attention mechanisms proposed in recent work and compare their impact on pressure-field prediction accuracy.

**Setup.** We use AeroFormer with the "Medium" configuration (hidden width $d = 64$, three cross-attention blocks, one head). We train four models that differ only in the atten-

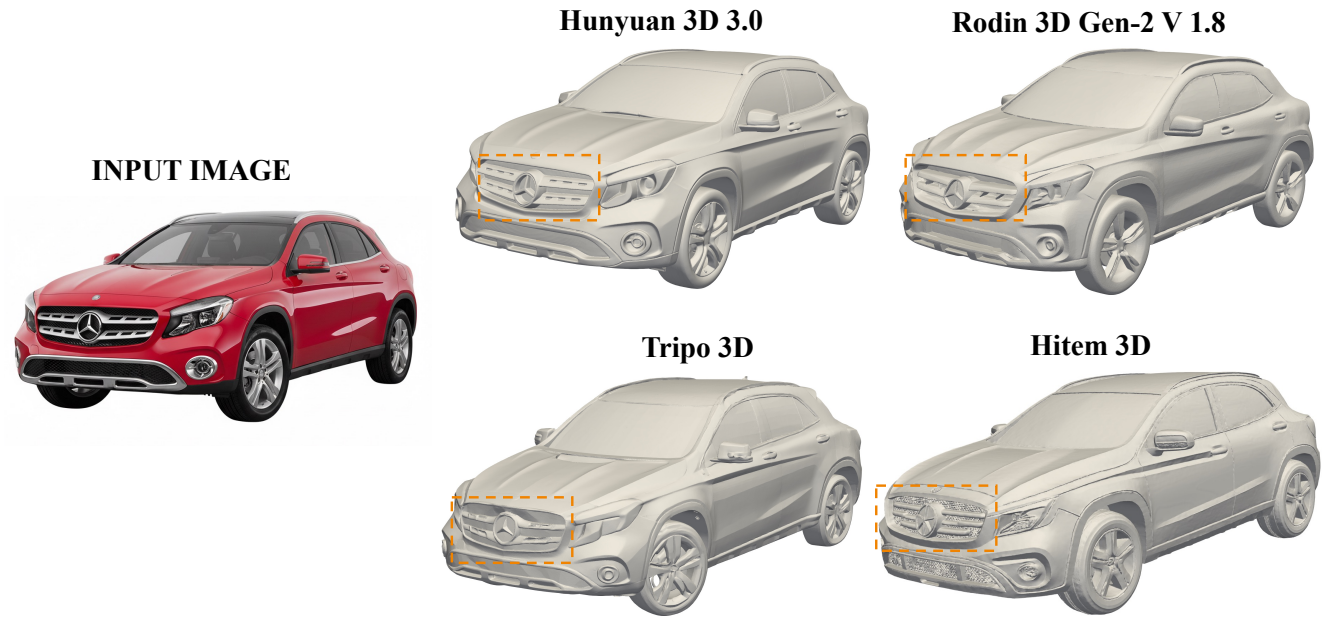



Figure 18. **Comparison of four image-to-three-dimensional models on the same standardized input image.** From left to right: input image, reconstruction by Hunyuan 3D 3.0, Rodin 3D Gen-2 V 1.8, Tripo 3D and Hitem 3D. The dashed boxes highlight the front fascia region, where accurate geometry is critical for aerodynamic simulation. Hunyuan 3D 3.0 best preserves grille, bumper and headlamp volumes and produces clean wheel and underbody geometry, whereas the other models either oversmooth features or introduce undesirable artefacts such as collapsed grilles and protruding blocks.

tion kernel used in the volumetric and surface branches; all other architecture and training hyperparameters (optimizer, learning-rate schedule, batch size and number of epochs) are kept identical. Evaluation is carried out on the same test split as in the main paper, using the relative $L_2$ error on the three-dimensional pressure field.

**Compared linear-attention variants.** We consider the following kernels:

- **Efficient Attention** (ours). The baseline uses Efficient Attention [49], where queries and keys are passed through a positive kernel map (implemented as ELU + 1) and the attention update is computed as $q(k^\top v)$ with a single global normalisation term based on $\sum k$.
- **InLineAttention** [20]. The InLineAttention kernel from the InLine project (LeapLab) performs linear attention in a compressed subspace with input-dependent projections.[2]
- **GateLinearAttentionNoSilu** [16]. The gated linear attention from RALA ("GateLinearAttentionNoSilu") augments the linear kernel with learned multiplicative gates on queries and keys.[3]

[2]Implementation: https://github.com/LeapLabTHU/InLine.

[3]Implementation: https://github.com/qhfan/RALA.

| Attention variant | Pressure rel. $L_2$ [%] |
|---|---|
| Efficient Attention ( [49]) | **10.76** |
| InLineAttention ( [20]) | 11.59 |
| GateLinearAttentionNoSilu ( [16]) | 11.32 |
| Sliding-window Linear Attention ( [6]) | 11.02 |

Table 12. **Ablation on linear-attention mechanisms in AeroFormer.** We keep the AeroFormer architecture and all training settings fixed and replace only the attention kernel. The Efficient Attention formulation of Shen et al. [49] achieves the lowest relative $L_2$ error on the pressure field (10.76%), while alternative linear-attention mechanisms yield consistently worse accuracy on our large-scale vehicle aerodynamics dataset.

- **Sliding-window Linear Attention** [6]. The sliding-window variant from the BASED project restricts attention to a local window in the token sequence while using a linear kernel within each window.[4]

**Results.** Table 12 reports the relative $L_2$ error on the pressure field for the four variants. The Efficient Attention kernel achieves the best accuracy with a relative $L_2$ error of 10.76%. All three alternative linear-attention mechanisms lead to noticeably higher errors in our setting.

[4]Implementation: https://github.com/HazyResearch/based.

**Discussion.** This ablation indicates that, in our high-Reynolds-number external-flow regime, a simple Efficient Attention kernel strikes a better balance between stability and expressiveness than more involved linear-attention variants. InLineAttention and GateLinearAttention introduce additional gating and projection structure that may be beneficial in language modelling, but appear to over-regularize the flow surrogate and reduce its ability to capture long-range couplings between front-end geometry and wake structures. The sliding-window variant improves locality but restricts the effective receptive field, which is undesirable for vehicle aerodynamics where global interactions across the entire body are important. For this reason we adopt Efficient Attention as the default linear-attention mechanism in all AeroFormer experiments in the main paper.

### 10.4. Effect of cross attention

A central design choice in AeroFormer is to explicitly couple the vehicle surface and the volumetric flow domain through token-wise cross attention. To verify that this mechanism is necessary, we compare the full model against an ablated variant in which cross attention is removed and replaced by a simple multi-layer perceptron (MLP) fusion.

**Setup.** We use AeroFormer with hidden width $d = 64$, three blocks, one head, Efficient Attention kernel, and train two models:

- **AeroFormer (with cross attention).** The standard dual-branch architecture in which the surface branch attends to itself and the volumetric branch attends to the surface via cross attention in every block.
- **Ablation: MLP fusion without cross attention.** We keep the same surface and volumetric encoders but remove all cross-attention modules. Instead, we compute a single global surface descriptor by average pooling over surface tokens and concatenate this vector to every volumetric token. The fused feature is passed through a two-layer MLP $d \to 2d \to d$ with GELU activation. This provides a global conditioning signal from the surface to the volume but does not allow token-wise interactions. The overall parameter count is matched to the cross-attention variant within $5\%$.

Both models are trained with identical optimizer, learning-rate schedule, batch size and number of epochs, and evaluated on the same test split using the relative $L_2$ error on the three-dimensional pressure field.

**Results.** Table 13 reports the relative $L_2$ error for the two variants. The full AeroFormer with cross attention achieves a relative $L_2$ error of $10.75\%$ on the pressure field. Replacing cross attention with global MLP fusion degrades performance to $15.47\%$, an absolute increase of $4.7$ percentage points and a relative increase of approximately $44\%$ in error.

| Model variant | Pressure rel. $L_2$ [%] |
|---|---|
| AeroFormer (with cross attention) | **10.75** |
| Ablation: MLP fusion, no cross attention | 15.47 |

Table 13. **Ablation on cross attention between surface and volume in AeroFormer.** Both models share the same encoders and training setup; the only difference is how information flows from the surface branch to the volumetric branch. Token-wise cross attention yields substantially lower relative $L_2$ error on the pressure field than a global MLP fusion baseline, indicating that explicit local interactions between surface geometry and volumetric tokens are crucial for accurately capturing the flow.

**Discussion.** This ablation shows that merely providing the volumetric branch with a global summary of the surface is not sufficient. Vehicle aerodynamics is dominated by local interactions between geometric features (front bumper, windscreen, roof, C-pillar, diffuser) and the surrounding flow, and these interactions vary across the body. Cross attention allows each volumetric token to attend to the most relevant surface tokens in a data-driven way, effectively learning a spatially varying transfer of information from geometry to flow. In contrast, the global MLP fusion baseline can only apply the same conditioning everywhere and fails to model long-range but spatially structured effects, leading to significantly worse pressure predictions. For this reason we retain cross attention as a core component of AeroFormer in all main experiments.

## 11. Additional Results

### 11.1. Token usage per design iteration

Because AeroAgent relies on a large language model with tools, each design iteration consumes a non-negligible number of tokens. To understand this overhead, we approximate the token usage of a typical 5-step ReAct-style interaction between the planner and the language model.[5]

**Assumptions.** We consider a text-only session in which the user issues a request such as "generate and optimize the aerodynamic shape of this car", and the agent performs up to five reasoning-and-acting steps. At each step, the model sees:

- a fixed system prompt and tool descriptions (about $1{,}200$ tokens in our implementation),
- the original user request (about $150$ tokens),

[5]The numbers in Table 14 are based on counting tokens in our actual system prompt, tool descriptions and representative ReAct traces. They should be read as realistic budgets rather than exact measurements for every possible query.

| Step | Input tokens | Output tokens | Total per step |
|---|---|---|---|
| 1 | $\approx 1{,}500$ | $\approx 350$ | $\approx 1{,}850$ |
| 2 | $\approx 2{,}100$ | $\approx 320$ | $\approx 2{,}420$ |
| 3 | $\approx 2{,}600$ | $\approx 280$ | $\approx 2{,}880$ |
| 4 | $\approx 3{,}000$ | $\approx 240$ | $\approx 3{,}240$ |
| 5 | $\approx 3{,}400$ | $\approx 220$ | $\approx 3{,}620$ |
| Mean per step | $\approx 2{,}520$ | $\approx 282$ | $\approx 2{,}802$ |
| Total over 5 steps | $\approx 12{,}600$ | $\approx 1{,}410$ | $\approx 14{,}010$ |

Table 14. **Representative token usage for a 5-step design session (text-only).** Each row reports the approximate number of input and output tokens processed by the language model at a given reasoning step, including the system prompt, tool descriptions, user intent, accumulated ReAct history and the model's own current reasoning trace. As the interaction progresses, the growing history increases the number of input tokens per call. For a typical 5-step interaction, the total token budget is on the order of $1.4 \times 10^4$ tokens.

- the accumulated history of previous Thought, Action and Observation lines, and
- it must generate a new Thought + Action (typically a few hundred tokens of reasoning plus a tool call).

Tool outputs (for example lists of file paths, numerical summaries and short text snippets) are appended to the history and increase the context length for subsequent steps. As a result, both the input and the output token counts vary from step to step and generally grow over the course of a session.

**Representative token budget.** Table 14 reports a representative budget for a 5-step interaction, averaged over several design sessions. The first step sees only the system prompt, tools and user request, and therefore has the smallest input context. Later steps re-include all previous thoughts, actions and tool observations, so their input grows steadily. The model's own outputs are also longer in early steps (more exploratory reasoning) and shorter near the end (more focused, mostly tool calls).

**Multi-modal sessions.** In multi-modal sessions that include one to three images, the vision encoder contributes an additional few hundred to a few thousand "image tokens" per session, depending on image resolution and model architecture. In practice we observe that this increases the total budget for a 5-step session to roughly $1.5\times10^4$–$1.8\times10^4$ tokens. Three-dimensional inputs (for example STL files) add negligible overhead, since only file paths and compact geometric summaries are exposed to the language model; the bulk of the three-dimensional processing is handled by dedicated geometry and simulation tools outside the language model.

## 11.2. End-to-end runtime: AeroAgent vs. traditional CFD loop

Beyond token usage, we also measure how much wall-clock time AeroAgent consumes compared with a traditional simulation-driven workflow. Here we focus on a single vehicle and a fixed number of design iterations, and compare: (1) a traditional loop, where every iteration is driven by a full high-fidelity computational fluid dynamics (CFD) run, and (2) our AeroAgent loop, where most iterations use the surrogate only and high-fidelity CFD is invoked once at the end to confirm the final design.

**Experimental setup.** We start from the same standardized baseline car and perform five propose–evaluate–refine iterations. In the traditional loop, each iteration triggers one steady external aerodynamics CFD run on our reference setup (multi-GPU solver, turbulent steady flow, full vehicle). The measured average wall-clock time of a single run on this setup is approximately $4$ hours; five iterations therefore require about $20$ hours of CFD solver time alone, not counting design hand-offs between departments. In the AeroAgent loop, the planner performs four inner iterations using the surrogate only and then sends one final design to the high-fidelity CFD solver. Over multiple runs, we measure that the combined time for the agent, vision pipeline and surrogate evaluations over four inner iterations is approximately $800$ seconds. The final CFD confirmation still takes $4$ hours. Thus, the total wall-clock time for a five-iteration design session with AeroAgent is roughly $4$ hours $+\,800$ seconds $\approx 4.2$ hours. On the same test cases, five surrogate-guided iterations with a single CFD confirmation reduce the drag coefficient by around $8\%$ on average relative to the initial shape, which is comparable to or better than what can be achieved with manual CFD-only loops but at substantially reduced computational cost.

**Results.** Table 15 compares the average runtime of the two loops for five design iterations. We report both the total wall-clock time and the time spent specifically in high-fidelity CFD. This experiment shows that, when counting only solver time, replacing five high-fidelity CFD runs with one CFD confirmation plus four surrogate evaluations yields a fivefold reduction in CFD compute time. Because the surrogate and agent overhead (about $800$ seconds) is small compared to a single CFD run, the end-to-end wall-clock time for a five-iteration design session is reduced from approximately $20$ hours to about $4.2$ hours, that is, by a factor of roughly $4.7$, cuts high-fidelity CFD calls by $80\%$ compared to traditional workflows. In practice, traditional workflows involve additional waiting time for cross-department communication and manual setup, so the effective human-perceived speedup of AeroAgent is likely even

| Method | Surrogate evals (per 5 iters) | CFD runs (per 5 iters) | Total time [hours] | CFD time [hours] |
|---|---|---|---|---|
| Traditional CFD-only | 0 | 5 | $\approx 20.0$ | $\approx 20.0$ |
| AeroAgent (ours) | 4 | 1 | $\approx 4.2$ | $\approx 4.0$ |
| Speedup (total) | | | $\approx 4.7\times$ | – |
| Speedup (CFD only) | | | – | $5.0\times$ |

Table 15. **Average wall-clock time for five design iterations: traditional CFD loop vs. AeroAgent.** In the traditional loop, each iteration is driven by a full high-fidelity CFD run, so five iterations require approximately $5 \times 4$ hours $= 20$ hours of solver time. In our loop, four inner iterations use the AeroFormer surrogate and the last iteration is confirmed by a single CFD run. The surrogate and agent take about 800 seconds (about 13 minutes) in total, and the final CFD run takes about 4 hours, for a total of roughly 4.2 hours per five-iteration session. This corresponds to a $\sim 4.7\times$ reduction in end-to-end wall-clock time and a $5\times$ reduction in high-fidelity CFD time, while achieving an average drag reduction of about $8\%$ over the starting geometry.

larger than these purely computational numbers suggest.

### 11.3. Extended progressive drag-reduction results

To further evaluate the reliability of AeroAgent in long-horizon editing, we extend the single-coupe experiment of Fig. 5 to a sequence of thirteen shapes produced by ten additional planner iterations (Fig. 19).In each step, the agent proposes a localised edit guided only by AeroFormer feedback, focusing first on the wheel arches, then the rear deck and diffuser, and finally the front fascia and hood.For every intermediate shape we visualise the edited 2D reference, the standardised 3D mesh, and a longitudinal pressure slice from AeroFormer and high-fidelity CFD, together with the corresponding drag coefficient $C_d$.

Across the full sequence, AeroAgent achieves a total drag reduction of $30.79\%$ on this coupe, from an initial $C_d \approx 0.43$ to $C_d \approx 0.30$, while all intermediate designs remain visually close to the original sports-car styling. The HF-CFD and AeroFormer curves for $C_d$ almost overlap over all 13 shapes and exhibit the same monotone decreasing trend, with no oscillations or regressions as the planner continues to refine the body. This indicates that (i) the surrogate not only predicts absolute drag accurately but also preserves the ordering and relative increments induced by small, local shape edits, and (ii) the agent can exploit this consistency to perform a stable, multi-step optimisation rather than relying on one-shot proposals.

We also conduct a second case study on a production-style SUV (Fig. 20) to assess whether the same progressive editing strategy transfers across vehicle classes. Starting from an initial design with $C_d \approx 0.34$, the planner runs eight surrogate-only iterations and arrives at a final shape with $C_d \approx 0.27$, i.e., a $20.53\%$ reduction in drag, while preserving the characteristic SUV stance and proportions. The AeroFormer and HF-CFD curves for $C_d$ again almost overlap and show a consistent downward trend, confirming that the surrogate provides reliable local guidance not only for sporty coupes but also for taller, boxier geometries. Together with the coupe experiment, this suggests that AeroAgent can generalise its drag-reduction behaviour across distinct vehicle segments while maintaining plausible exterior styling.

In terms of practical efficiency, this kind of progressive drag-reduction loop would be prohibitively expensive with a CFD-only workflow: evaluating 10–15 variants for a single vehicle typically entails tens of hours of wall-clock time on our HF-CFD setup, once mesh generation, solver runtime and queueing delays are accounted for. In contrast, AeroAgent runs all inner-loop edits purely with AeroFormer; on a single GPU, a 10-step optimisation like the one in Fig. 19 completes in about 30 minutes, after which only the final 1–2 shortlisted designs are confirmed by high-fidelity CFD.This corresponds to roughly an order-of-magnitude reduction in high-fidelity compute per vehicle, while still delivering large drag improvements and preserving a high level of perceived styling quality, supporting our design choice of using the surrogate exclusively inside the agent loop and reserving HF-CFD evaluations for final validation.

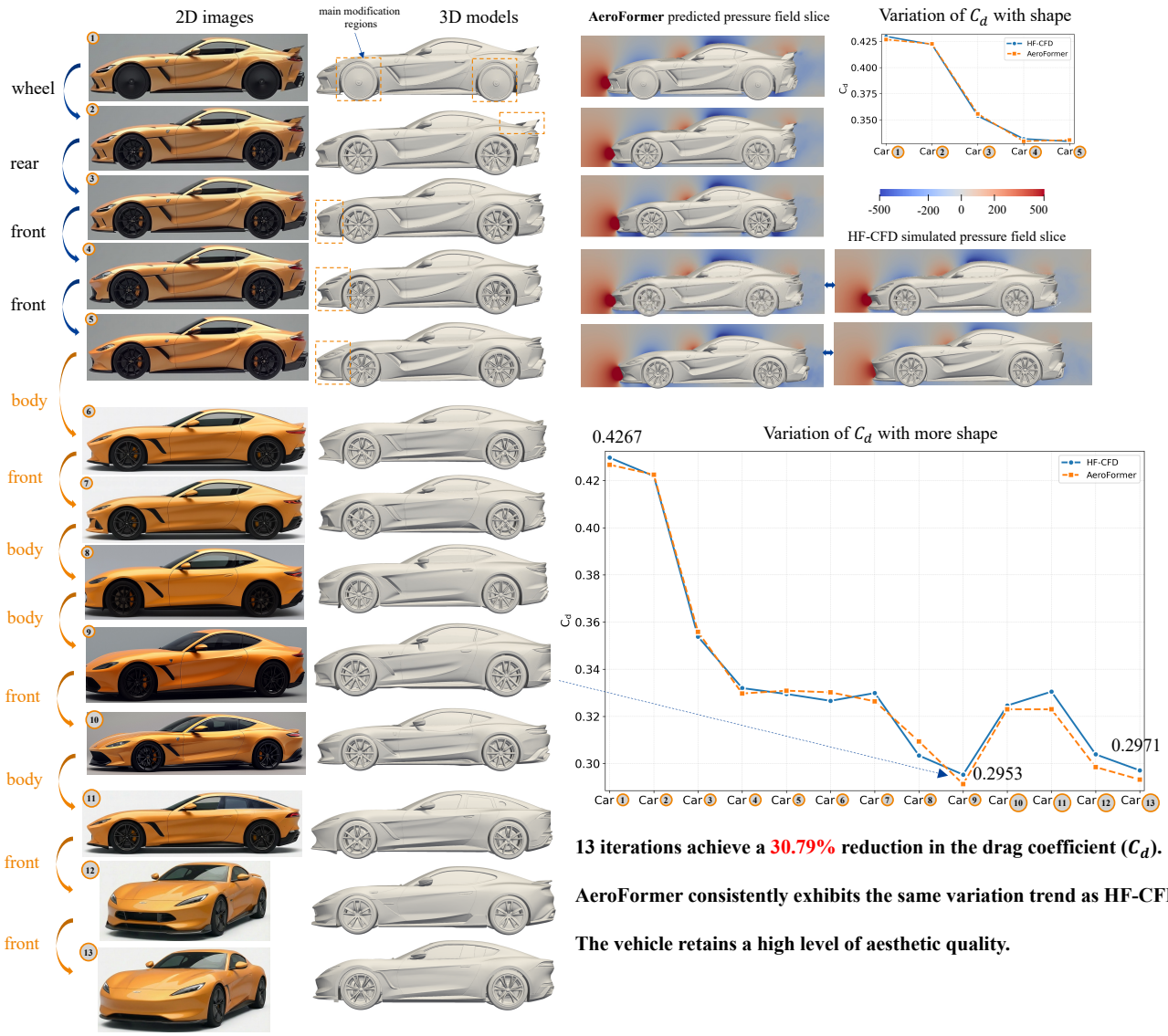



Figure 19. **Extended progressive drag reduction on a single coupe.** We extend Fig. 5 to a sequence of thirteen planner-generated edits on the same vehicle. Shapes 1–13 show how AeroAgent gradually refines the wheel arches, rear deck and diffuser, and front fascia using only AeroFormer feedback. For each step we visualise the edited 2D reference, the standardised 3D mesh, and a longitudinal pressure slice from AeroFormer and high-fidelity CFD, together with the corresponding drag coefficient $C_d$. Across the full sequence, $C_d$ decreases monotonically and the AeroFormer and CFD curves almost coincide, yielding a total drag reduction of 30.79% while preserving the coupe's overall styling. This extended example further demonstrates that AeroAgent can perform stable long-horizon optimisation using only surrogate feedback in the inner loop.

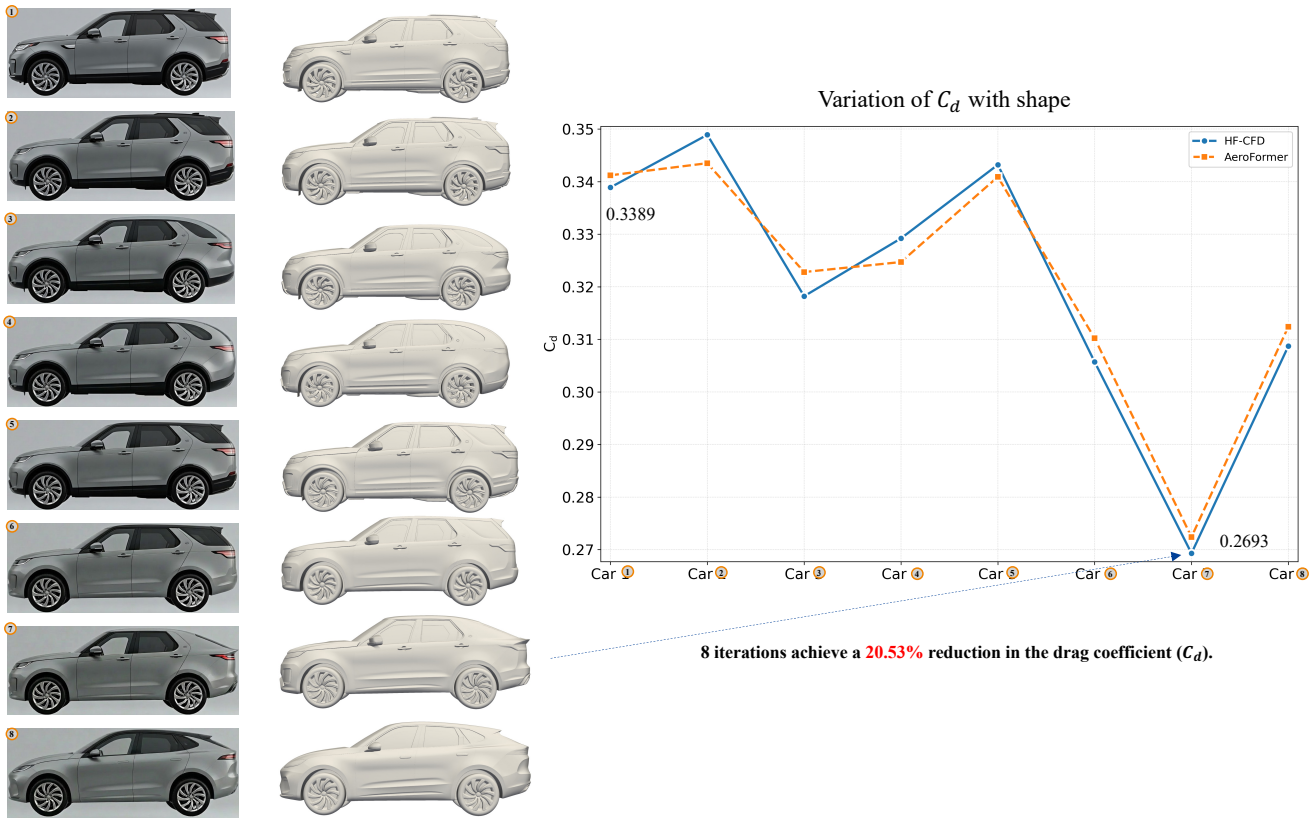



Figure 20. **Progressive drag reduction on a production-style SUV.** Starting from an initial SUV design, AeroAgent performs eight rounds of planner-guided edits using only AeroFormer feedback. The left panel shows, for each iteration, the edited 2D reference and the standardised 3D mesh; the right panel plots the drag coefficient $C_d$ of all eight shapes as predicted by AeroFormer and measured by high-fidelity CFD. The two curves closely match and decrease from $C_d \approx 0.34$ to $C_d \approx 0.27$, corresponding to a 20.53% reduction in drag, while the SUV retains its overall proportions and styling cues. This case study demonstrates that AeroAgent generalises beyond the coupe in Fig. 19 and can reliably reduce drag on a different vehicle class.