Flow-Corrected Shape Optimization: Taming Manifold Drift in High-Dimensional 3D Models

Emilien Seiler
CVLab, EPFL
emilien.seiler@epfl.ch

Nicolas Talabot
CVLab, EPFL
nicolas.talabot@epfl.ch

Yingxuan You
CVLab, EPFL
yingxuan.you@epfl.ch

Federico Stella
CVLab, EPFL
federico.stella@epfl.ch

Pascal Fua
CVLab, EPFL
pascal.fua@epfl.ch

Abstract

Optimizing 3D shapes within the latent spaces of deep generative models is fundamental to computer assisted engineering, yet remains prone to a critical failure mode we term *manifold drift*: the tendency of gradient-based optimization to move latent vectors away from the manifold of valid shapes. This problem is exacerbated in state-of-the-art 3D shape generative models that operate in increasingly high-dimensional latent spaces where valid shapes occupy a vanishingly small fraction of the full space. Existing mitigation strategies, including latent regularization and flow-matching approaches, either sacrifice expressiveness, demand a difficult trade-off between objective guidance and generative fidelity that remains prone to manifold drift, or are computationally infeasible to scale to modern, large-capacity 3D shape models. We introduce a novel optimizer-corrector framework that alternates between gradient steps for objective minimization and guided flow matching to drive the latent state back to the valid shape manifold. By decoupling objective minimization from flow-based correction, optimizing freely and correcting strictly, this alternating design avoids inherent trade-offs, preserving geometric validity without sacrificing expressiveness while remaining computationally feasible on modern 3D shape models. We demonstrate its effectiveness across generative priors of varying complexity, from simple vector latent spaces to large-scale architectures across a variety of downstream optimization tasks, including aerodynamic drag reduction and object compliance optimization.

1 Introduction

Refining 3D shapes to maximize a measure of their performance is central to many engineering disciplines, from Computer Assisted Design to Computer Vision and Graphics, with applications in aeronautics, healthcare, and manufacturing [42, 80, 31, 61]. With the advent of generative models, the approaches have shifted from traditional mesh-based refinement to continuous optimization within learned latent spaces [66, 10, 9]. To capture finer geometric details, state-of-the-art shape models [30, 76, 70] have grown large and now operate in very high-dimensional latent spaces.

While this increase in complexity has boosted both descriptive power and accuracy for shape *reconstruction* and *generation*, it also introduces new challenges for shape *optimization*. In practice, one often starts with a shape that is within a training distribution. This distribution often lives on a manifold occupying only a small part of the complete latent space. The initial shape is then refined to minimize an objective function, with no guarantee that it will stay close to that manifold. Thus, as the optimization progresses, the shape may end up far from it, making it potentially nonsensical

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

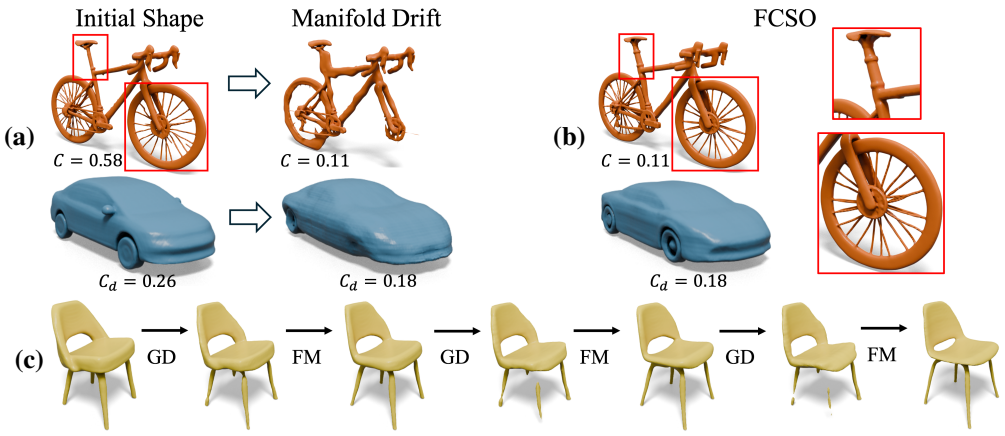

Figure 1: Visualizing (a) manifold drift in gradient descent optimization for bike stiffness and car drag vs. (b) our results on the same tasks. (c) Our proposed framework, which alternates between Gradient Descent (GD) and Flow Matching (FM) correction, applied here to volume reduction.

as shown in Fig. 1(a), a phenomenon we will refer to as *manifold drift*. This problem was already observed when using early neural representations [53] but is more acute for new, larger ones due to the *Curse of Dimensionality*. In high-dimensional spaces, the manifold of valid shapes occupies an ever smaller fraction of the complete space, making manifold drift even more of an issue.

As this is a critical vulnerability of generative methods, mitigation strategies have been proposed. Latent regularization techniques typically trade off the expressiveness of the search space for geometric validity [39, 55]. More recent ones rely on flow matching [38] and diffusion models [58]. Some approaches optimize directly with respect to the starting point of the flow matching process, which is effective for small models but quickly becomes computationally intractable when using modern 3D architectures [5, 62, 10]. Other approaches use the optimization objective to provide guidance at inference time [24, 78]. However, because these methods attempt to perform both objective optimization and geometric correction in a single pass, they force a complex trade-off. Too much guidance breaks the generative prior, causing manifold drift and degenerate shapes, while too little guidance results in failure to meaningfully optimize the objective.

In this paper, we address these issues by introducing Flow-Corrected Shape Optimization (FCSO), a novel framework that explicitly separates the gradient descent from the correction steps to ensure robust convergence while remaining computationally tractable, effectively preventing manifold drift, as shown in Fig. 1(b). Conceptually, our algorithm operates as an optimizer-corrector mechanism. The gradient descent steps act as the optimizer that moves the latent vector strictly in the direction of steepest descent to minimize the objective. The flow matching acts as the corrector, driving the updated state back to the valid shape manifold. As illustrated in Fig. 1(c), by alternating these steps, we make progress on the optimization landscape without accumulating geometric errors over successive iterations. We can leverage pre-trained flow models when provided, without requiring any retraining.

We demonstrate the effectiveness and scalability of FCSO with generative shape priors, from simple vector latent spaces [50] to newer much more complex ones [30]. Thus, our contribution is twofold:

- We show that the increased expressivity of modern 3D models fundamentally degrades the reliability of gradient-based latent optimization and that current diffusion and flow-based models are insufficient to mitigate this.
- We introduce a robust shape optimization algorithm that alternates between gradient-based objective minimization and flow-matching corrections. This approach effectively addresses the manifold drift problem while remaining computationally tractable, even when scaling to highly complex architectures.

Our experimental results show that we outperform state-of-the-art baselines on several representative shape optimization tasks; the code and benchmark will be released at `https://github.com/eseiler18/fcso`.

2 Related work

2.1 Shape modeling

Recent advances in 3D generative models leverage expressive latent spaces to map compact latent vectors into complex 3D geometries [15, 74, 21]. These vectors can be decoded into various representations, such as voxel grids [69], point clouds [18], or implicit neural fields [50, 46]. The latter continuously describe surfaces that can subsequently be extracted as explicit meshes [43, 57, 59]. To deliver higher-fidelity and more precise 3D representation, the complexity of these latent spaces has significantly increased. Early approaches, such as DeepSDF [50], relied on a single global latent vector to represent an entire shape and, while effective for simple topologies, this makes it difficult to capture fine details. More recent models, such as 3DShape2VecSet [76], Dora [11] or Hunyuan3D [30], replace the single vectors by sets of vectors and incorporate attention layers into the decoders to increase expressivity. On the other hand, local latent representations in the form of sparse featurized voxels allow each vector to locally represent a specific region of the shape [70, 27]. Driven by scaling laws, these recent architectures require a high-dimensional parameter space and massive training datasets. For example, Hunyuan3D [30] represents a single shape using 4,096 vectors of size 64, offering state-of-the-art geometric fidelity. While complex shape priors are essential for high-quality representations, they exacerbate manifold drift in optimization as the set of latents that decode into meaningful shapes becomes increasingly sparse. We propose a solution that mitigates this for arbitrary shape priors of any complexity as long as a differentiable mapping exists between the latent representation and the optimization objective.

2.2 Gradient-based optimization

Because the sophisticated models discussed above have so many parameters, black-box optimization, including Bayesian optimization [56] and evolutionary strategies [26], are not always practical and gradient-based methods are better suited. Thus, the gradients have to be computed and back-propagated to the latent parameters [75, 53]. While some objectives can be computed directly from the shape or the neural field [66, 75], others depend on complex physical quantities estimated using numerical simulations that are both computationally expensive and non-differentiable [48]. A common workaround is then to train surrogate models based on Gaussian Processes or Neural Networks [4, 23, 68] to predict these physical quantities. The former are well established in many engineering disciplines under the name of *Kriging* [33] but are less than ideal to deal with large numbers of parameters. The latter can handle high-dimensional latent vectors, enabling fast gradient evaluation via back-propagation through the surrogate network. Another approach is to integrate physical surrogates directly as auxiliary decoders operating on the latent space, coupling shape generation and physical prediction [72]. However, as observed in recent studies [23, 10], optimizing latent vectors inherently pushes them away from the manifold of valid shapes, yielding shapes that are invalid even if they minimize the objective. This is compounded by the use of surrogate objective functions, which become unreliable out-of-distribution and can erroneously assign high scores to these degenerate shapes. This challenge echoes the out-of-domain artifacts identified in 2D GAN inversion [79, 64], those early 2D works used adversarial regularizers to keep latent codes valid, they inherit standard GAN instabilities.

2.3 Flow matching for optimization

Diffusion [58] and Flow Matching [38] have emerged as promising ways to provide the necessary control by bringing back the drifting data to the training distribution. Related work in image restoration often uses diffusion or flow models for inverse problems [5, 12, 35, 78, 51]. Like us, they minimize an objective while maintaining meaningful outputs. However, these methods typically rely on a data-driven reconstruction term to anchor the generation process [34], mitigating manifold drift, however, no such reference exists in our setting. One approach differentiates directly through the flow process by back-propagating the objective from the final output to the initial noise vector [5, 10, 62]. While mathematically elegant, back-propagating through an entire ODE solver incurs massive overheads, scaling poorly to high-dimensional latent spaces. Moreover, our experiments show that even these intensive approaches fail to prevent manifold drift. To avoid the heavy back-propagation cost, one can instead rely on gradient guidance within the generation process. Recent work has explored optimization via guided diffusion [24] extended to flow matching models by incorporating a

trajectory regularization guided by the score function and the trace of the velocity field's Jacobian [78]. However, guidance methods require sensitive hyper-parameter tuning to balance a fundamental trade-off: while strong guidance causes manifold drift by breaking the generative prior, weak guidance fails to meaningfully minimize the objective. Yet another approach is to introduce a stochastic score approximation regularization during the optimization [9] but relying on soft regularization often fails to strictly prevent manifold drift in highly complex spaces. In contrast, alternating between gradient and flow matching steps, as we propose, effectively handles manifold drift without any of the above drawbacks.

3 Background: flow matching

We briefly review the standard Flow Matching (FM) [38, 1, 41] approach.

Following the convention, Gaussian noise is sampled at time $t = 0$ and de-noised. The clean data is obtained at time $t = 1$. To this end, FM constructs a probability density path $p_t(\mathbf{x})$ that transforms a simple prior p_0 ($\mathcal{N}(\mathbf{0}, \mathbf{I})$ in the case of Gaussian noise) into a complex data distribution p_1 over time $t \in [0, 1]$. This transformation is governed by an Ordinary Differential Equation (ODE)

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t), \quad \text{with} \quad \mathbf{x}_0 \sim p_0, \tag{1}$$

where $\mathbf{v}$ is the vector field. A neural network $\mathbf{v}_\theta$, typically a U-Net [54] or a Transformer [17, 65], is trained to estimate $\mathbf{v}$ from $\mathbf{x}_t$ and t. As doing so for all possible values of $\mathbf{x}$ would be intractable, we use the Rectified Flow formulation [41], which relies on the straight-line interpolation between sampled noise $\mathbf{x}_0$ and data point $\mathbf{x}_1$

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1 \ . \tag{2}$$

The target conditional vector field reduces to the constant velocity $\mathbf{v}(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) = \mathbf{x}_1 - \mathbf{x}_0$ and the network is trained by minimizing the loss

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1}\left[\left\|\mathbf{v}_\theta(\mathbf{x}_t, t) - (\mathbf{x}_1 - \mathbf{x}_0)\right\|^2\right]. \tag{3}$$

4 Method

Our goal is to refine a 3D shape S to satisfy a specific numerical objective, while leveraging known geometric priors learned by a pre-trained generative model. We assume that we are given a pre-trained decoder $\mathcal{D} : \mathcal{Z} \to \mathcal{S}$ that maps a latent $\mathbf{z}$ to its explicit 3D spatial representation $S = \mathcal{D}(\mathbf{z})$. Given a task-specific objective function $\mathcal{O} : \mathcal{S} \to \mathbb{R}$ and an initial latent vector $\mathbf{z}_s$, our task is to minimize $\mathcal{O}$ with respect to $\mathbf{z}$ to find

$$\mathbf{z}_s^* = \arg\min_{\mathbf{z}} \mathcal{O}\left(\mathcal{D}(\mathbf{z})\right), \quad \text{initialized at } \mathbf{z}^{(0)} = \mathbf{z}_s \ . \tag{4}$$

In practice, $\mathbf{z}$ can be either a single global vector as in DeepSDF [50], or a set of latent vectors as in 3DShape2VecSet [76] or Hunyuan3D [30].

4.1 The manifold drift challenge

While seemingly straightforward in theory, the optimization of Eq. 4 is fraught with problems in practice: As discussed in Section 2.2, when the latent space $\mathcal{Z}$ is high-dimensional, the physically plausible and valid shapes usually live on a much-lower dimensional manifold $\mathcal{M} \subset \mathcal{Z}$ [20, 19]. When using a gradient-based method and taking optimization steps of the form

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} - \eta \nabla_{\mathbf{z}} \mathcal{O}\big(\mathcal{D}(\mathbf{z}^{(k)})\big) \ , \tag{5}$$

with η a step size, the task gradient $\nabla_{\mathbf{z}}\mathcal{O}$ is computed solely to minimize the objective and $\mathbf{z}^{(k)}$ may drift further away from $\mathcal{M}$ with each new iteration. The pre-trained decoder $\mathcal{D}$ being trained on elements of $\mathcal{M}$, the predictions risk gradually becoming less and less meaningful, devolving into adversarial or unrealistic shapes that numerically minimize the objective $\mathcal{O}$ without representing a valid 3D structure.

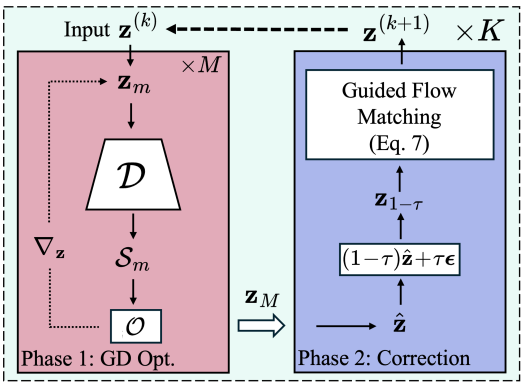

Figure 2: Method Diagram Overview. We alternate between gradient-based optimization (Phase 1) and correction via guided Flow Matching (Phase 2) to safely navigate the shape manifold and optimize the objective $\mathcal{O}$.

Algorithm 1 Flow-Corrected Shape Optimization

Require: Initial $\mathbf{z}^{(0)}$, Objective $\mathcal{O}$
Require: Flow Model $\mathbf{v}_\theta$, Decoder $\mathcal{D}$
Require: Optimization iterations K, gradient steps M, step size η, noise ratio τ, flow steps N

for $k = 0, \dots, K-1$ **do**
  Phase 1: Gradient-Based Minimization
  $\mathbf{z}_0 \leftarrow \mathbf{z}^{(k)}$
  for $m = 0, \dots, M-1$ **do**
    $\mathbf{z}_{m+1} \leftarrow \mathbf{z}_m - \eta \nabla_{\mathbf{z}} \mathcal{O}\big(\mathcal{D}(\mathbf{z}_m)\big)$ (Eq. 5)
  end for
  $\hat{\mathbf{z}} \leftarrow \mathbf{z}_M$, $c \leftarrow \mathcal{O}\big(\mathcal{D}(\hat{\mathbf{z}})\big)$
  Phase 2: Flow-Based Correction
  $\mathbf{x} \leftarrow (1-\tau)\hat{\mathbf{z}} + \tau\boldsymbol{\epsilon}$ (Eq. 6)
  for $j = 0, \dots, N-1$ **do** (Solve Eq. 9)
    $t_j \leftarrow (1-\tau) + j\frac{\tau}{N}$
    $\mathbf{x} \leftarrow \mathbf{x} + \Delta t \cdot \tilde{\mathbf{v}}_\theta(\mathbf{x}, t_j, c)$ (Eq. 7 or Eq. 8)
  end for
  $\mathbf{z}^{(k+1)} \leftarrow \mathbf{x}$
end for
return $\mathbf{z}^{(K)}$

4.2 Flow-corrected shape optimization

To prevent this, we propose using a pre-trained Flow Matching model to keep the optimization trajectory close to the manifold. In particular, we argue that alternating between direct gradient-descent optimization and flow-based correction, as outlined in Fig. 2 and Algorithm 1, allows a more robust optimization, which we demonstrate in our experiment.

Phase 1: Gradient-Based Minimization. Given the current latent state $\mathbf{z}^{(k)}$ at optimization cycle k, we first take M gradient steps (Eq. 5) to minimize $\mathcal{O}$ with respect to $\mathbf{z}$. After M steps, this yields a potentially off-manifold latent vector $\hat{\mathbf{z}}^{(k)}$. This phase is the primary driver of the optimization to minimize the target objective.

Phase 2: Flow-Based Correction. To pull $\hat{\mathbf{z}}^{(k)}$ back onto the valid manifold $\mathcal{M}$, we use a Flow Matching model $\mathbf{v}_\theta$ pre-trained on this manifold of the shape prior's latent space.

For this, we first perform a partial noising step and then exploit the straight-line probability path of the Flow Matching framework. For a chosen noise injection ratio $\tau \in (0, 1)$, we directly interpolate between the latent code $\hat{\mathbf{z}}^{(k)}$ and sampled noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This yields

$$\mathbf{z}_{1-\tau}^{(k)} = (1-\tau)\hat{\mathbf{z}}^{(k)} + \tau\boldsymbol{\epsilon}\,, \tag{6}$$

which pulls $\hat{\mathbf{z}}^{(k)}$ towards the Gaussian noise, thereby attenuating the high-frequency adversarial artifacts that may be introduced by direct gradient descent.

Directly using the FM network $\mathbf{v}_\theta$ of Sec. 3 for the correction towards the data manifold tends to undo the improvement brought about by gradient descent. Instead, we guide the flow field with the objective value attained at the end of the gradient phase, that is explicitly $c = \mathcal{O}\big(\mathcal{D}(\hat{\mathbf{z}}^{(k)})\big)$. See Appendix E for an ablation study showing that guidance helps achieve faster convergence. This yields a guided vector field $\tilde{\mathbf{v}}_\theta$ that takes as input the current noised latent $\mathbf{z}_t$, the time t, and this condition c. It is computed using one of the two following ways. For notational simplicity, we omit the optimization cycle index (k).

Option A: Classifier-Free Guidance (CFG) [29].

The flow model $\mathbf{v}_\theta$ accepts an optional condition c as additional input. During training, c is set as the objective value of the current target shape $\mathcal{O}\big(\mathcal{D}(\mathbf{z}_1)\big)$. We use either the condition c defined above or no condition denoted by $\emptyset$ to compute

$$\tilde{\mathbf{v}}_\theta(\mathbf{z}_t, t, c) = \mathbf{v}_\theta(\mathbf{z}_t, t, \emptyset) + \omega\Big(\mathbf{v}_\theta(\mathbf{z}_t, t, c) - \mathbf{v}_\theta(\mathbf{z}_t, t, \emptyset)\Big)\,, \tag{7}$$

where $\omega > 1$ dictates the guidance strength.

Option B: Gradient-based Guidance. The guided vector field is dynamically adjusted at each step to minimize the discrepancy between the current and target objective scores. We write

$$\tilde{\mathbf{v}}_\theta(\mathbf{z}_t, t, c) = \mathbf{v}_\theta(\mathbf{z}_t, t) - \lambda \nabla_{\mathbf{z}_t} \left(\mathcal{O}\big(\mathcal{D}(\mathbf{z}_t)\big) - c \right)^2 , \tag{8}$$

where $\lambda > 0$ is the guidance scale.

Option A is the preferred approach, as it leverages classifier-free guidance for faster execution. However, it requires the flow model to be conditioned on the optimization objective, which is not always possible. In contrast, option B leverages pre-trained models and is designed for large-scale approaches where retraining is impractical.

We integrate the ODE forward from $t = 1 - \tau$ to $t = 1$, as in standard FM we write:

$$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)}_{1-\tau} + \int_{1-\tau}^{1} \tilde{\mathbf{v}}_\theta(\mathbf{z}^{(k)}_t, t, c)\, dt\,. \tag{9}$$

In practice, this continuous integral is approximated using a standard numerical ODE solver. In Appendix D, we leverage established flow theory to show why Phase 2 corrects manifold drift while maintaining optimization progress.

As discussed above and shown in the experiments, pure guided flow matching is not enough to effectively drive the optimization. Therefore, we propose this alternating strategy with gradient-descent. This process is repeated for K cycles, allowing the shape to progressively optimize the objective $\mathcal{O}$ while safely traveling along the valid shape manifold. To prevent accumulated momentum from interfering with the updated latent trajectory, the optimizer is completely reset after every flow-matching correction step.

5 Results

To demonstrate the efficacy and scalability of our proposed approach to complex models, we test it on several representative tasks and against state-of-the-art baselines. We discuss our experimental setup below and then present our comparative results.

5.1 Experimental setup

Currently, no established benchmarks exist to evaluate 3D shape optimization within latent spaces. We designed a comprehensive set of tasks, models, and metrics aimed at a fair and extensive comparison. This benchmark will be released at `https://github.com/eseiler18/fcso`.

Tasks. We design three tasks of increasing complexity to evaluate our method. The first involves reducing the volume of a given chair, which can be seen as a simple emulation of trying to make an object lighter. This is a simple objective that can be evaluated directly from the shape itself, yet can easily cause manifold drift by rewarding the partial disappearance of the object. Therefore, it serves as a simple yet useful testbed to demonstrate manifold drift. The second, both more realistic and more complex, consists in refining the shape of a car to minimize its aerodynamic drag as evaluated by a surrogate model. This is a meaningful but challenging engineering task because surrogate models are highly non-linear and often yield nonsensical results for shapes that are too far from the training examples, magnifying the manifold drift that we aim to tame. The third experiment demonstrates scalability to the most recent and complex state-of-the-art models. We work in the latent space of Hunyuan3D [30] and minimize the displacements under load by minimizing the *compliance*, as defined in [75], which can be understood as maximizing the stiffness of the objects. Finally, to show that our method can extend beyond physics-based objectives, we evaluate an additional 2D silhouette-matching objective via differentiable rendering; its results are provided in Appendix I.

Baselines. We compare our approach against several methods: standard Gradient Descent (GD) [23, 75] and Flow Matching Guidance (FMG), either gradient-based or classifier-free [29]. We also compare to D-Flow [5], which optimizes the starting point by differentiating through the entire flow trajectory; to ICTM [78], which enhances guidance by incorporating the score function and the trace of the velocity field's Jacobian; and to SGO [9], a score-based regularization approach during gradient descent optimization. Since our optimization starts from an initial shape rather than pure noise, we rely on an SDEdit-style [45] process for FMG and ICTM to enable editing rather than pure generation. Further details are provided in Appendix B.

Table 1: Chair volume optimization metrics at 70% of volume reduction. Metrics are scaled for readability: KD_{incep} ($\times 10^3$), FD_{point} ($\times 10^1$), and KD_{point} ($\times 10^4$).

Method	DeepSDF						3DShape2VecSet					
	FD_{incep}	KD_{incep}	FD_{dino}	KD_{dino}	FD_{point}	KD_{point}	FD_{incep}	KD_{incep}	FD_{dino}	KD_{dino}	FD_{point}	KD_{point}
GD	47.37	13.00	368.42	19.51	2.79	3.11	88.12	55.63	659.34	32.43	3.94	6.44
FMG	43.67	11.47	351.58	19.66	2.66	3.82	43.14	14.25	333.72	15.18	2.72	5.02
D-Flow	**36.29**	10.28	**334.13**	21.29	**2.36**	**2.88**	56.92	23.45	426.55	19.01	3.24	4.81
ICTM	43.65	10.06	351.45	**16.78**	2.62	3.19	91.71	12.90	576.96	15.22	3.89	3.25
SGO	42.08	10.27	339.18	19.43	2.80	3.97	32.73	5.68	240.94	11.27	2.20	1.79
Ours	41.12	**9.72**	335.42	19.66	2.49	3.00	**31.76**	**4.83**	**234.61**	**10.66**	**1.95**	**1.42**



Figure 3: Qualitative comparison of chair optimization using DeepSDF and 3DShape2VecSet with our method and baselines. V denotes the volume. More examples are provided in Appendix F.2

Metrics. Beyond minimizing the objective, optimized shapes must remain meaningful. To quantify this, we employ metrics commonly used to assess generative capabilities [70, 76], specifically computing the Fréchet Inception Distance [28] and Kernel Inception Distance [6] between optimized shapes and a reference distribution of test shapes. Distances are computed using 3D latent features from PointBert [40, 73] and 2D features from 10 rendered views extracted via Inception [60] and DINOv2 [49]. For a fair comparison, we evaluate all methods at matched optimization progress: each shape is assessed once its objective has been reduced by a fixed relative amount from its initial value (e.g., 50% of initial volume). This ensures that differences in shape quality reflect the optimizer's behavior rather than how far it has driven the objective. Additionally, to quantify how much the shape identity is preserved during optimization, we measure the perceptual similarity between the initial and optimized 3D shapes using the LPIPS metric [77] applied to 2D renderings. Details are provided in Appendix A.

5.2 Chair volume reduction

Given a pre-trained decoder that generates chairs, our goal is to minimize the shape's volume while keeping them realistic, as shown in Fig. 3. To demonstrate how all methods behave when the shape model increases in complexity, we use two: the older and simpler DeepSDF [50] that decodes a single 256-D latent vector with an MLP and the more recent and complex 3DShape2VecSet [76] that decodes a set of 256 vectors of size 8 with a Transformer. We train them and volume-conditioned Flow Matching models on 2,419 watertight ShapeNet chairs [8]. We then use an SDF-based approximation to compute and minimize the volume of 100 test chairs, as described in Appendix F.1. Given this conditioning, we use Classifier Free Guidance (Eq. 7) for both our method and FMG. Additional implementation details are given in Appendix F. In Tab. 1, we report our quantitative results at 70% volume reduction for all baselines. We provide more results in Appendix F.2.

As evidenced by Tab. 1, when using the smaller DeepSDF model, our approach operates on par with the baselines and brings no particular benefit. This was to be expected: As DeepSDF operates in a small, compact latent space, it does not suffer from severe manifold drift. In contrast, the much more expressive 3DShape2VecSet operates on a much sparser higher-dimensional manifold where

Table 2: Car drag optimization (3DShape2VecSet) at 10% and 30% drag reduction. Metrics are scaled for readability: KD_{incep} ($\times 10^3$), FD_{dino} ($\times 10^{-1}$), FD_{point} ($\times 10^2$), and KD_{point} ($\times 10^5$).

Method	**Drag 10%**						**Drag 30%**					
	FD_{incep}	**KD_{incep}**	**FD_{dino}**	**KD_{dino}**	**FD_{point}**	**KD_{point}**	**FD_{incep}**	**KD_{incep}**	**FD_{dino}**	**KD_{dino}**	**FD_{point}**	**KD_{point}**
GD	84.88	56.77	65.33	49.60	16.89	32.40	132.82	106.26	97.97	75.63	37.50	123.20
FMG	83.43	42.62	62.54	43.62	17.62	20.40	100.20	67.02	77.05	56.52	25.00	73.00
D-Flow	72.33	42.74	61.76	45.86	15.38	25.20	108.79	76.44	82.85	62.68	29.95	76.70
ICTM	74.87	47.25	60.58	46.23	15.71	30.20	101.73	75.52	75.71	58.31	24.97	77.60
SGO	78.26	49.37	63.53	47.49	16.58	27.80	107.72	76.89	81.54	61.25	30.75	83.70
Ours	**62.42**	**34.81**	**52.79**	**40.68**	**11.35**	**12.80**	**89.08**	**55.91**	**69.20**	**52.72**	**20.96**	**51.40**

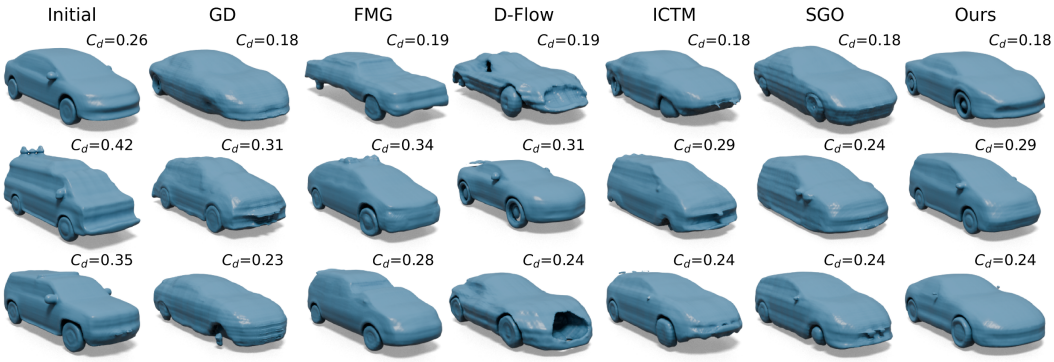

Figure 4: Qualitative comparison of drag optimization. C_d is the drag predicted by the surrogate. See Appendix G.3 for more examples.

manifold drift has much more severe consequences and our approach comes into its own. Here, we clearly outperform the other methods, yielding much more realistic chairs as shown in Fig. 3. An exception is SGO [9] that is a close second. However, that does not generalize to more complex tasks, as shown in the following experiments. This highlights our method's ability to effectively leverage the expressivity of sophisticated models without suffering from manifold drift, whereas baselines struggle with the intricacies of navigating a more complex latent space. This trend is confirmed in the following experiments with more complex cases.

5.3 Car drag optimization

Moving to a true engineering task, we now seek to minimize the aerodynamic drag (C_d) of a car. We focus this study on the 3DShape2VecSet [76] prior, for its higher capacity, and compare at two different levels of optimization progress. We train this shape prior and a drag-conditioned Flow Matching model on 836 watertight ShapeNet cars [8] with corresponding Computational Fluid Dynamics (CFD) simulation using OpenFOAM [48]. In theory, Adjoint Differentiation [3, 22] can be used to compute C_d in a differentiable manner. However, this is extremely expensive because it requires a new simulation to be run each time [2] and can be complex to deploy. Thus, following standard engineering practice [32, 36, 44, 63, 71], we use instead a surrogate model, specifically the GraphSAGE GCNN [25, 7]. Afterwards, we optimize 100 test cars to minimize their drag. We report quantitative results in Tab. 2 and qualitative ones in Fig. 4. As above, we use CFG (Eq. 7) for ours and for FMG. See Appendix G.1 for implementation details and Appendix G.3 for additional results.

Our approach outperforms the baselines on all metrics, with the gap increasing the further the optimization progress (10% vs. 30% drag reduction targets). It can be seen in Fig. 4 that the baselines exhibit manifold drift with over-smoothing, surface artifacts, and disappearing wheels. The SGO results were relatively close to ours in the simpler volume reduction task of the previous section but the difference is much starker here. This is presumably due to the surrogate being highly non-linear with complex gradients and SGO's regularization, based on a stochastic score approximation, fails to maintain geometric validity and results in over-smoothing. We also confirm that these optimizations effectively minimize true physical drag by running simulations [48], see Appendix G.2.

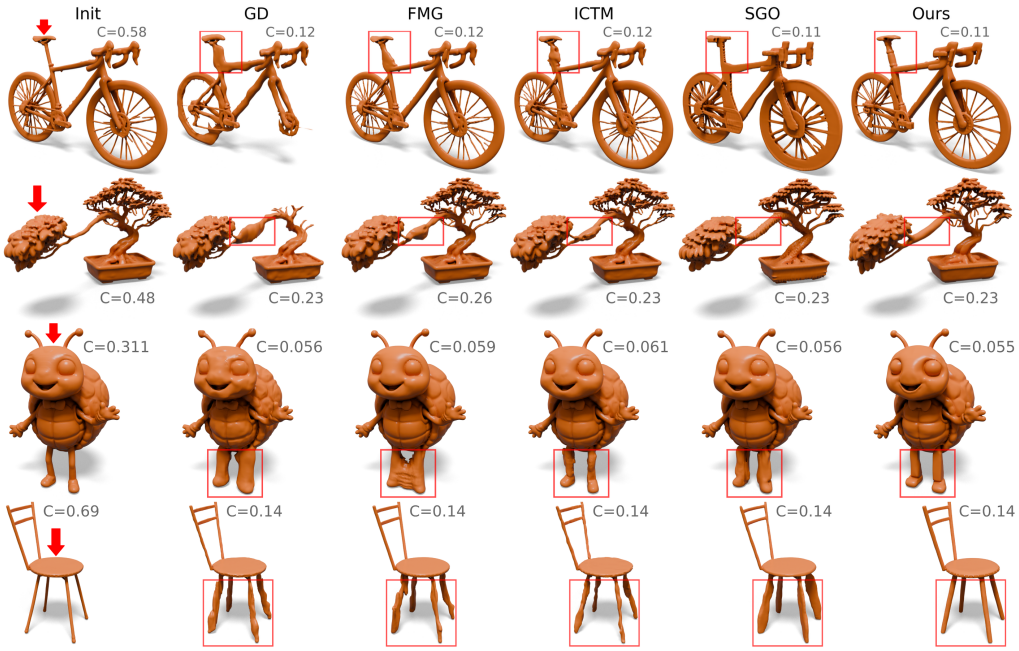

Figure 5: We show the initial shapes with the applied force (red arrow) and optimized shapes across methods. The compliance (C) is reported, where lower values indicate better structural stability.

5.4 Minimizing displacements under load in Hunyuan3D

To show that our approach scales to truly complex state-of-the-art shape models, we test it on Hunyuan3D [30]. It is a foundation model for high-fidelity 3D generation. It encodes shapes in a set of vectors of size 4096×64, which enables the capture of intricate geometric details. Unlike the category-specific priors used in our previous experiments, Hunyuan3D acts as a large-scale general prior trained on a vast range of objects. While such a high-dimensional space is normally reserved for generation, we show that our method makes it possible to effectively exploit it for optimization.

Table 3: Compliance (C) optimization on BIKED bicycles and ABO chairs at 85% and 65% reduction, respectively. Scaling: $\text{KD}_{\text{incep}}(\times 10^3)$, $\text{FD}_{\text{point}}(\times 10^2)$, $\text{KD}_{\text{point}}(\times 10^5)$. Examples in Appendix H.2.

Method	**Bicycles**						**Chairs**					
	FD_{incep}	**KD_{incep}**	**FD_{dino}**	**KD_{dino}**	**FD_{point}**	**KD_{point}**	**FD_{incep}**	**KD_{incep}**	**FD_{dino}**	**KD_{dino}**	**FD_{point}**	**KD_{point}**
GD	43.09	19.39	267.63	19.71	20.84	28.82	68.18	29.69	575.84	39.67	32.71	27.80
FMG	47.34	15.17	263.37	12.59	22.21	21.51	55.74	19.45	431.28	26.57	27.21	16.00
D-Flow	*OOM*						*OOM*					
ICTM	47.41	14.87	278.98	12.61	23.03	25.63	55.92	19.37	432.10	26.80	28.05	18.33
SGO	21.13	1.54	84.32	1.08	12.97	4.48	55.08	16.20	426.41	22.49	29.31	12.40
Ours	**20.65**	**1.12**	**82.37**	**0.81**	**12.41**	**0.38**	**41.27**	**9.22**	**282.41**	**15.89**	**23.07**	**5.20**

We minimize the physics-driven compliance C introduced in PhysiOpt [75], which is equivalent to maximizing structural stiffness. In practice, this ensures the shape deforms as little as possible under loads such as the weight applied to a chair's seat or a bicycle's saddle. See details in Appendix H.1. While foundation models like Hunyuan3D are trained on highly diverse datasets (e.g., Objaverse [14]), automatically assigning physically meaningful boundary conditions to arbitrary complex geometries remains a challenge. We therefore select the chairs from the ABO dataset [13] (see Appendix H.2 for Tables) and Bicycles from the BIKED dataset [52] to evaluate our method on complex topologies where physical constraints can be reliably automatically defined. We report our quantitative results in Tab. 3. Fig. 5 features qualitative results on a diverse set of shapes. Implementation details are given in Appendix H.1 and more results in Appendix H.2.

Table 4: Shape identity preservation measured by LPIPS (mean $\pm$ std). Lower values indicate better preservation of the original shape's visual identity.

Method	**Chair Volume (Tab. 1)**		**Car Drag (Tab. 2)**		**Hunyuan3D Compliance (Tab. 3)**	
	DeepSDF	3DShape2VecSet	10%	30%	Bicycles	Chairs
GD	0.26 ± 0.05	0.29 ± 0.07	0.07 ± 0.02	0.14 ± 0.02	0.07 ± 0.05	0.08 ± 0.06
FMG	0.28 ± 0.06	0.27 ± 0.05	0.06 ± 0.03	0.10 ± 0.04	$\mathbf{0.04 \pm 0.03}$	0.07 ± 0.04
D-Flow	0.33 ± 0.08	0.25 ± 0.06	0.07 ± 0.04	0.11 ± 0.03	*OOM*	
ICTM	0.27 ± 0.05	0.24 ± 0.05	0.06 ± 0.02	0.08 ± 0.04	0.05 ± 0.03	0.07 ± 0.05
SGO	$\mathbf{0.25 \pm 0.05}$	$\mathbf{0.22 \pm 0.04}$	0.04 ± 0.02	0.08 ± 0.03	0.05 ± 0.03	$\mathbf{0.05 \pm 0.07}$
Ours	$\mathbf{0.25 \pm 0.05}$	$\mathbf{0.22 \pm 0.03}$	$\mathbf{0.03 \pm 0.02}$	$\mathbf{0.06 \pm 0.03}$	$\mathbf{0.04 \pm 0.02}$	$\mathbf{0.05 \pm 0.02}$

Our approach again outperforms all baselines across all metrics, demonstrating superior preservation of the original data distribution, and leads to cleaner shape edits compared to the artifacts produced by other methods. Notably, D-Flow proved computationally infeasible causing Out-Of-Memory (OOM) errors (NVIDIA A100 GPU with 40GB VRAM) because Hunyuan3D's architecture is too large, hence the missing numbers in the table. Standard GD causes widespread geometric distortion, while FMG and ICTM correctly localize modifications to stressed regions but generate unnatural, bumpy surfaces. SGO performs better but still yields blocky and bumpy deformations, such as the bicycle frame and bonsai branch. In contrast, our method naturally thickens the targeted regions while maintaining smooth, high-fidelity geometry.

5.5 Shape identity preservation

As shown in Tab. 4, FCSO matches or achieves the lowest divergence from the original shape across all tasks. This confirms that our method effectively preserves visual identity, rather than simply exploiting the optimization landscape to minimize the objective. Nevertheless, aggressive targets can still alter the style of the shape: in Fig. 4 (third row), the SUV adopts a more sedan-like profile. While our method preserves identity better than the baselines, there remains room for improvement.

6 Limitations

While more efficient than back-propagation through the entire flow matching process, our approach is more computationally demanding than standard gradient descent because each optimization cycle requires N forward evaluations of the vector field during the correction phase (see Appendix C). Additionally, under high optimization targets, our method can occasionally introduce minor geometric artifacts or alter the style of the shape (Sec. 5.5). However, it consistently preserves the overall shape semantics (see Appendix J). We leave sparse voxel latents [70] to future work, as adding material there requires an inpainting step [75] that dominates the geometric change. Furthermore, the hyperparameters governing the process, specifically the number of gradient steps M and the noise injection ratio τ, are currently fixed. Future work can focus on developing adaptive mechanisms, potentially by leveraging surrogate model uncertainty or real-time shape quality metrics to dynamically balance objective minimization and shape quality preservation.

7 Conclusion

We have proposed a mechanism that tames the manifold drift problem when optimizing 3D shapes parameterized by high-dimensional latent vectors for performance. This means that we can now handle complex composite objects that require such models. Our next step will be to also enforce consistency constraints between the different parts of such objects to guarantee that, when one part is deformed, the others are deformed in a way that maintains the coherence of the whole object. In other words, we will move from the unconstrained optimization described in this paper to optimization under consistency constraints.

Acknowledgments and Disclosure of Funding

This work was funded in part by the Swiss National Science Foundation.

References

[1] Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations*, 2023.

[2] J. Alexandersen, O. Sigmund, and N. Aage. Large Scale Three-Dimensional Topology Optimization of Heat Sinks Cooled by Natural Convection. *International Journal of Heat and Mass Transfer*, 100(Supplement C):876–891, 2016.

[3] G. Allaire. A Review of Adjoint Methods for Sensitivity Analysis, Uncertainty Quantification and Optimization in Numerical Codes. *Ingénieurs de l'Automobile*, 836:33–36, July 2015.

[4] P. Baqué, E. Remelli, F. Fleuret, and P. Fua. Geodesic Convolutional Shape Optimization. In *International Conference on Machine Learning*, 2018.

[5] Heli Ben-Hamu, Omri Puny, Itai Gat, Brian Karrer, Uriel Singer, and Yaron Lipman. D-Flow: Differentiating through Flows for Controlled Generation. In *International Conference on Machine Learning*, 2024.

[6] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. In *International Conference on Learning Representations*, 2018.

[7] F. Bonnet, J. A. Mazari, P. Cinnella, and P. Gallinari. AirfRANS: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. In *Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[8] A. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An Information-Rich 3D Model Repository. In *arXiv Preprint*, 2015.

[9] Huaguan Chen, Ning Lin, Luxi Chen, Rui Zhang, Wenbing Huang, Chongxuan Li, and Hao Sun. Optimization and generation in aerodynamics inverse design. *arXiv Preprint*, 2026.

[10] Long Chen, Emre Oezkaya, Jan Rottmayer, Nicolas R. Gauger, Zebang Shen, and Yinyu Ye. Adjoint-based aerodynamic shape optimization with a manifold constraint learned by diffusion models. *arXiv Preprint*, 2025.

[11] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *Conference on Computer Vision and Pattern Recognition*, 2025.

[12] Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.

[13] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F. Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Conference on Computer Vision and Pattern Recognition*, 2022.

[14] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Conference on Computer Vision and Pattern Recognition*, 2023.

[15] Yuan Dong, Qi Zuo, Xiaodong Gu, Weihao Yuan, Zhengyi Zhao, Zilong Dong, Liefeng Bo, and Qi-Xing Huang. GPLD3D: Latent Diffusion of 3D Shape Generative Models by Enforcing Geometric and Physical Priors. In *Conference on Computer Vision and Pattern Recognition*, 2024.

[16] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 2011.

[17] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.

[18] H. Fan, H. Su, and L. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *Conference on Computer Vision and Pattern Recognition*, 2017.

[19] Tyler Farghly, Peter Potaptchik, Samuel Howard, George Deligiannidis, and Jakiw Pidstrigach. Diffusion models and the manifold hypothesis: Log-domain smoothing is geometry adaptive. *arXiv Preprint*, 2025.

[20] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *arXiv Preprint*, 2013.

[21] 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. In *Advances in Neural Information Processing Systems*, 2022.

[22] Y. Gao, Y. Wu, and J. Xia. Automatic Differentiation Based Discrete Adjoint Method for Aerodynamic Design Optimization on Unstructured Meshes. *Chinese Journal of Aeronautics*, 30(2):611–627, 2017.

[23] B. Guillard, E. Remelli, A. Lukoianov, S. Richter, T. Bagautdinov, P. Baque, and P. Fua. DeepMesh: Differentiable Iso-Surface Extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(11):7072–7087, 2024.

[24] Yingqing Guo, Hui Yuan, Yukang Yang, Minshuo Chen, and Mengdi Wang. Gradient Guidance for Diffusion Models: An Optimization Perspective. In *Advances in Neural Information Processing Systems*, 2024.

[25] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017.

[26] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv Preprint*, 2016.

[27] Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. Sparseflex: High-resolution and arbitrary-topology 3d shape modeling. In *International Conference on Computer Vision*, 2025.

[28] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017.

[29] J. Ho and T. Salimans. Classifier-Free Diffusion Guidance. In *arXiv Preprint*, 2022.

[30] Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, Qingxiang Lin, Zeqiang Lai, Xianghui Yang, Huiwen Shi, Zibo Zhao, Bowen Zhang, Hongyu Yan, Lifu Wang, Sicong Liu, Jihong Zhang, Meng Chen, Liang Dong, Yiwen Jia, Yulin Cai, Jiaao Yu, Yixuan Tang, Dongyuan Guo, Junlin Yu, Hao Zhang, Zheng Ye, Peng He, Runzhou Wu, Shida Wei, Chao Zhang, Yonghao Tan, Yifu Sun, Lin Niu, Shirui Huang, Bojian Zheng, Shu Liu, Shilin Chen, Xiang Yuan, Xiaofeng Yang, Kai Liu, Jianchen Zhu, Peng Chen, Tian Liu, Di Wang, Yuhong Liu, Linus, Jie Jiang, Jingwei Huang, and Chunchao Guo. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv Preprint*, 2025.

[31] Taimoor Iqbal, Ling Wang, Dichen Li, Enchun Dong, Hongbin Fan, Jun Fu, and Cai Hu. A general multi-objective topology optimization methodology developed for customized design of pelvic prostheses. *Medical Engineering and Physics*, 2019.

[32] S. Jeong, M. Murayama, and K. Yamamoto. Efficient Optimization Design Method Using Kriging Model. *Journal of Aircraft*, 42(2):413–420, 2005.

[33] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 1998.

[34] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022.

[35] Jeongsol Kim, Bryan Sangwoo Kim, and Jong Chul Ye. Flowdps: Flow-driven posterior sampling for inverse problems. In *International Conference on Computer Vision*, 2025.

[36] J. Laurenceau, M. Meaux, M. Montagnac, and P. Sagaut. Comparison of Gradient-Based and Gradient-Enhanced Response-Surface-Based Optimizers. *American Institute of Aeronautics and Astronautics Journal*, 48(5):981–994, 2010.

[37] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient Implementation of Marching Cubes' Cases with Topological Guarantees. In *Journal of Graphics Tools*, 2003.

[38] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023.

[39] Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization. In *ACM SIGGRAPH*, 2022.

[40] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. In *Advances in Neural Information Processing Systems*, 2023.

[41] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

[42] A. Lopez, G. Magri, U. Cella, G. Urso, F. Della Barba, and M. E. Biancolini. Efficient shape optimization in aeronautics: Integrating parametric cad and mesh morphing for enhanced aerodynamic performance. In *International Council of the Aeronautical Sciences*, 2024.

[43] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *ACM SIGGRAPH*, pages 163–169, 1987.

[44] Andrew March, Karen Willcox, and Qiqi Wang. Gradient-Based Multifidelity Optimisation for Aircraft Design using Bayesian Model Calibration. *The Aeronautical Journal*, 115(1174):729–738, 2011.

[45] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

[46] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[47] B.R. Munson, T.H. Okiishi, W.W. Huebsch, and A.P. Rothmayer. *Fluid Mechanics*. Wiley Singapore, 2013.

[48] Ltd OpenCFD. OpenFoam. https://www.openfoam.com/.

[49] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P. Huang, H. Xu, V. Sharma, S. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning Robust Visual Features Without Supervision. *Transactions on Machine Learning Research*, 2024.

[50] J. J. Park, P. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[51] Maitreya Patel, Song Wen, Dimitris N. Metaxas, and Yezhou Yang. Steering rectified flow models in the vector field for controlled image generation. In *International Conference on Computer Vision*, 2025.

[52] Lyle Regenwetter, Brent Curry, and Faez Ahmed. Biked: A dataset for computational bicycle design with machine learning benchmarks. *Journal of Mechanical Design*, 2021.

[53] E. Remelli, A. Lukoianov, S. Richter, B. Guillard, T. Bagautdinov, P. Baque, and P. Fua. MeshSDF: Differentiable Iso-Surface Extraction. In *Advances in Neural Information Processing Systems*, 2020.

[54] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 234–241, 2015.

[55] Dvir Samuel, Rami Ben-Ari, Nir Darshan, Haggai Maron, and Gal Chechik. Norm-guided latent space exploration for text-to-image generation. In *Advances in Neural Information Processing Systems*, 2023.

[56] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 2016.

[57] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler. Deep Marching Tetrahedra: A Hybrid Representation for High-Resolution 3D Shape Synthesis. In *Advances in Neural Information Processing Systems*, 2021.

[58] J. Song, C. Meng, and S. Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2021.

[59] F. Stella, N. Talabot, H. Le, and P. Fua. High Resolution UDF Meshing via Iterative Networks. In *Advances in Neural Information Processing Systems*, 2025.

[60] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *Conference on Computer Vision and Pattern Recognition*, pages 1–9, June 2015.

[61] N. Tan and R. J. van Arkel. Topology optimisation for compliant hip implant design and reduced strain shielding. *Materials*, 2021.

[62] Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. In *International Conference on Machine Learning*, 2024.

[63] D. Toal and A.J. Keane. Efficient Multipoint Aerodynamic Design Optimization via Cokriging. *Journal of Aircraft*, 48(5):1685–1695, 2011.

[64] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM Transactions on Graphics*, 2021.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

[66] Zhen Wei, Aobo Yang, Jichao Li, Michaël Bauerheim, Rhea Liem, and Pascal Fua. DeepGeo: Deep Geometric Mapping for Automated and Effective Parameterization in Aerodynamic Shape Optimization. In *AIAA AVIATION FORUM AND ASCEND*, 2024.

[67] D. Wieser, H.-J. Schmidt, S. Müller, C. Strangfeld, C. Nayeri, and C. Paschereit. Experimental Comparison of the Aerodynamic Behavior of Fastback and Notchback DrivAer Models. *SAE Int. J. Passeng. Cars - Mech. Syst.*, 7(2):682–691, November 2014.

[68] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. In *International Conference on Machine Learning*, 2024.

[69] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D Shapenets: A Deep Representation for Volumetric Shapes. In *Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[70] 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 *Conference on Computer Vision and Pattern Recognition*, 2025.

[71] G. Xu, X. Liang, S. Yao, D. Chen, and Z. Li. Multi-Objective Aerodynamic Optimization of the Streamlined Shape of High-Speed Trains Based on the Kriging Model. *PloS one*, 12(1):1–14, 01 2017.

[72] Yingxuan You, Chen Zhao, Hantao Zhang, Ming Xu, and Pascal Fua. PhysGen: Physically Grounded 3D Shape Generation for Industrial Design. In *Conference on Computer Vision and Pattern Recognition*, 2026.

[73] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Conference on Computer Vision and Pattern Recognition*, 2022.

[74] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis. LION: Latent Point Diffusion Models for 3D Shape Generation. In *Advances in Neural Information Processing Systems*, 2022.

[75] Xiao Zhan, Clément Jambon, Evan Thompson, Kenney Ng, and Mina Konaković Luković. Physiopt: Physics-driven shape optimization for 3d generative models. In *ACM SIGGRAPH Asia*, 2025.

[76] B. Zhang, J. Tang, M. Nießner, and P. Wonka. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. In *ACM SIGGRAPH*, 2023.

[77] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[78] Yasi Zhang, Peiyu Yu, Yaxuan Zhu, Yingshan Chang, Feng Gao, Ying Nian Wu, and Oscar Leong. Flow priors for linear inverse problems via iterative corrupted trajectory matching. In *Advances in Neural Information Processing Systems*, 2024.

[79] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *European Conference on Computer Vision*, 2020.

[80] Jihong Zhu, Han Zhou, Chuang Wang, Lu Zhou, Shangqin Yuan, and Weihong Zhang. A review of topology optimization for additive manufacturing: Status and challenges. *Chinese Journal of Aeronautics*, 2021.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our contributions are clearly stated in the abstract and listed at the end of the introduction (Section 1). These claims are fully validated by the experimental results presented in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, the limitations of our work are explicitly discussed in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

4. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: We don't claim new theoretical result

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

5. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, all necessary implementation details and experimental setups required to reproduce our work are provided in Appendices F.1, G.1, and H.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

6. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code and benchmark will be released at `https://github.com/eseiler18/fcso`. In the meantime, comprehensive details, including pseudo-code 1 and hyperparameters in Tables 7, 8, 10, are provided to allow independent reproduction.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

7. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, all necessary implementation details and experimental setups required to reproduce our work are provided in Appendices F.1, G.1, and H.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

8. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our primary metrics are FID and KID. As these are population-level metrics, computing error bars requires rerunning the entire optimization process for the full dataset multiple times. Given the high computational cost of 3D shape optimization, this was not performed. For the per-shape LPIPS metric, we report the mean and standard deviation across all optimized shapes (Tab. 4).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

9. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, full details regarding the hardware, memory configurations are provided in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

10. **Code of ethics**

    Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

    Answer: [Yes]

    Justification: We have reviewed the NeurIPS Code of Ethics and confirm that this research strictly adheres to it in every respect.

    Guidelines:

    - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
    - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
    - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

11. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This paper presents foundational algorithmic research on 3D latent space optimization. As a generic mathematical framework, it is not tied to specific deployments and does not have a direct path to foreseeable negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

12. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our paper introduces an optimization algorithm for 3D shapes rather than releasing a new high-risk foundational model or a scraped dataset. We rely on existing, standard public datasets (ShapeNet, ABO, BIKED) and pre-trained priors, which do not require additional safeguards for release.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

13. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Yes, all creators of the datasets (ShapeNet [8], ABO [13], BIKED [52]) and models (DeepSDF [50], 3DShape2VecSet [76], Hunyuan3D [30]) used in our study are properly cite.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

14. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We are not releasing the source code, new datasets, or new models at this time. Therefore, no new assets are introduced in this submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

15. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

16. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

17. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs,

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

Appendix

Contents of the Appendix

A Metrics 24
B Baselines 24
C Computational resources and runtimes 25
D Theoretical insights: preventing manifold drift 26
E Ablation study: guidance during flow matching 26
F Chair volume optimization 27
F.1 Implementation details 27
F.2 Additional results 29
G Car drag optimization 32
G.1 Implementation details 32
G.2 Validation of true aerodynamic performance 33
G.3 Additional results 34
H Compliance optimization in Hunyuan3D 35
H.1 Implementation details 35
H.2 Additional results 36
I Additional objective: 2D silhouette matching 40
J Limitations: minor artifact in high optimization targets 41

A Metrics

To quantify the quality of the optimized shapes, we measure the distributional discrepancy between set of optimized shapes and a set of reference shapes. For the chair (5.2) and car (5.3) experiments, the reference set is constructed using ground-truth shapes from ShapeNet. For the PhysiOpt optimization on the ABO and BIKED datasets (5.4), the reference set comprises shapes extracted via Hunyuan3D using the dataset's images. We enforce strict separation: the reference set is computed entirely independently of the shapes that we optimize. Furthermore, to guarantee a fair comparison across methods, we compute the set of optimized shapes at a fixed objective reduction threshold. This allows us to quantify shape quality once all methods have reached an equivalent optimization state.

The metrics below are computed on feature embeddings extracted from 3D geometries using pre-trained PointBert [40, 73], as well as from 10 multi-view 2D renders processed through pre-trained Inception [60] and DINOv2 [49] networks.

Fréchet Distance [28] This metric is most widely recognized by its seminal formulation, the Fréchet Inception Distance (FID). Although we extract features using multiple backbones (PointBert, DINOv2) in addition to Inception, we retain the widely adopted FID nomenclature and mathematical formulation. FID approximates the feature distributions as multivariate Gaussians and computes the Wasserstein-2 distance between them. Let (μ_r, Σ_r) and (μ_o, Σ_o) denote the mean and covariance of the features from the reference set and the optimized set, respectively. The FID is defined as:

$$FID = \|\mu_r - \mu_o\|^2 + \mathrm{Tr}\big(\Sigma_r + \Sigma_o - 2(\Sigma_r\Sigma_o)^{1/2}\big) \tag{10}$$

where Tr denotes the trace of a matrix. A lower FID indicates that the optimized shapes closely match the structural fidelity of the reference dataset.

Kernel Distance [6] Following a similar convention, while this metric computes the kernel Maximum Mean Discrepancy (MMD), it is commonly referred to in the generative literature as the Kernel Inception Distance (KID). We maintain this standard terminology across all our feature extractors. Unlike FID, KID does not assume a Gaussian distribution. It computes the squared MMD between feature representations using a kernel function. We also compute this metrics using multiple backbones (PointBert, DINOv2) in addition to Inception. Given the feature distribution of the reference set P_r, the distribution of the optimized set P_o, and a kernel function $k(x, y)$, the KID is formulated as:

$$KID = \mathbb{E}_{x,x'\sim P_r}[k(x, x')] + \mathbb{E}_{y,y'\sim P_o}[k(y, y')] - 2\mathbb{E}_{x\sim P_r, y\sim P_o}[k(x, y)] \tag{11}$$

We utilize a polynomial kernel $k(x, y) = (\frac{1}{d}x^T y + 1)^3$, where d is the feature dimension. Lower KID scores confirm that the optimized geometries remain valid and structurally aligned with the reference shapes.

LPIPS [77] To quantify the preservation of shape identity during optimization, we employ the Learned Perceptual Image Patch Similarity (LPIPS) metric. Unlike Fréchet or Kernel distances which measure the distance between population distributions, LPIPS computes the perceptual similarity between pairs of individual shapes (specifically, between the initial shape and its optimized counterpart). We extract multi-view 2D renderings of both the initial and optimized 3D shapes and compute the distance between their feature maps extracted from a pre-trained network (AlexNet). By operating in feature space, LPIPS effectively captures whether the original visual identity and structure are maintained, while naturally accommodating the valid, physics-driven deformations required by the optimization. A lower LPIPS score indicates that the optimized shape closely preserves the visual identity of the initial shape.

B Baselines

In this section, we detail the baselines used for comparison and explain how we adapt them, when necessary, to our specific task of shape optimization.

Gradient Descent (GD) GD serves as our most straightforward baseline, representing a standard approach widely employed in various shape optimization problems [23, 75]. Beyond acting as a baseline, it also functions as an ablation study: omitting Phase 2 of our proposed method effectively reduces the entire pipeline to standard gradient descent. To encourage the optimization to stay close

to the manifold of valid shapes, we augment the objective with an L_2 regularization term anchored to the initial latent code, adding $\lambda\|z - z_{init}\|_2^2$ to the loss, where z is the current latent vector, z_{init} is the initial starting point, and λ is the regularization weight.

Flow Matching Guidance (FMG) FMG serves as an additional baseline that also functions as an ablation study, as it isolates Phase 2 of our proposed pipeline by omitting the gradient descent step. Adopting an SDEdit-style approach [45], we partially noise the latent representation and subsequently perform guided flow matching to optimize the shape, utilizing the guidance formulations defined in either Eq. 7 or Eq. 8. Equation 7 was used for the chair volume (5.2) and car drag (5.3) experiments, as the underlying flow matching model could be conditioned on the optimization objective during training. For the Hunyuan3D experiment (5.4), where the model was unconditioned with respect to the optimization objective, we resorted to the gradient guidance formulation in Eq. 8.

D-Flow Originally designed to solve image inverse problems, D-Flow [5] directly optimizes the initial noise vector that serves as the input to the flow matching process. However, this approach is computationally expensive, as it requires back-propagating gradients through the entire ODE trajectory, from the generated data back to the initial noise distribution. To adapt D-Flow to our shape optimization task, we initialize the optimization process using the specific noise vector that generates our starting shape. This initial noise vector is pre-computed via an optimization-based latent inversion process prior to the shape optimization.

Iterative Corrupted Trajectory Matching (ICTM) Developed for linear inverse problems in image restoration, ICTM [78] enhances guided flow matching through a local MAP objective that combines a score-based prior with a regularization term derived from the velocity field's Jacobian trace. To adapt this framework to our shape optimization task, we employ an SDEdit-style approach [45] by initializing the generative trajectory from a partially noised latent vector rather than pure noise, thereby maintaining structural proximity to the initial shape.

Score Guided Optimization (SGO) Score Guided Optimization (SGO) [9] leverages a regularization term based on a stochastic score approximation, $-\nabla_{\mathbf{x}} \log p(\mathbf{x})$, which is controlled by a hyperparameter λ. This λ parameter is empirically tuned through trial and error to find the optimal balance between objective minimization and shape validity for each experiment.

C Computational resources and runtimes

All experiments were conducted on a GPU cluster. Specifically, the volume reduction experiments (Section 5.2) and the aerodynamic drag optimization experiments (Section 5.3) were executed using NVIDIA V100 GPUs (32GB VRAM). Due to the higher dimensionality and larger memory footprint of the Hunyuan3D model, the compliance minimization experiments (Section 5.4) required NVIDIA A100 GPUs (40GB VRAM).

Table 5: Average computational cost per shape optimization run across different experimental setups and optimization methods. The objectives are a 70% volume reduction for Chair (DeepSDF/VecSet), a 30% drag reduction for Car (VecSet), a 65% compliance reduction for Hunyuan Chair, and a 50% compliance reduction for Hunyuan Table.

Method	Chair (DeepSDF)	Chair (VecSet)	Car (VecSet)	Hunyuan Chair	Hunyuan Table
GD	38s	1m 02s	53s	3m 18s	2m 34s
FMG	10s	14s	15s	55s	59s
D-Flow	3m 51s	14m 56s	17m 10s	OOM	OOM
ICTM	30s	54s	55s	2m16s	2m11s
SGO	2m 10s	3m 20s	3m 55s	6m 09s	6m 58s
Ours	2m 15s	3m 25s	3m 12s	6m 21s	6m 21s

Table 5 provides a detailed overview of the average computational cost across the different experimental setups. FMG is as the most computationally efficient method, particularly for the first three experimental setups, as it only requires a single pass through the Flow Matching model (utilizing CFG). ICTM requires slightly more time with the additional term apply during guidance. Standard

gradient descent (GD) takes comparatively longer. Furthermore, SGO and our proposed method exhibit higher runtimes than pure GD due to the additional overhead of evaluating the Flow Matching model. Finally, D-Flow incurs the highest computational cost because it requires backpropagating through the entire ODE trajectory at each step. This intensive memory requirement makes D-Flow completely infeasible for larger architectures like Hunyuan3D, resulting in Out-Of-Memory (OOM) errors (NVIDIA A100 GPU with 40GB VRAM).

D Theoretical insights: preventing manifold drift

Recall from Section 4 that our alternating optimization scheme aims to refine a latent vector $\mathbf{z}$ to minimize a task-specific objective $\mathcal{O}(\mathcal{D}(\mathbf{z}))$. Let $\hat{\mathbf{z}}^{(k)}$ denote the intermediate latent vector obtained after the gradient-based minimization (Phase 1) at cycle k, achieving an objective value $c = \mathcal{O}\big(\mathcal{D}(\hat{\mathbf{z}}^{(k)})\big)$. Furthermore, in our Flow Matching formulation, time $t \in [0, 1]$ is defined such that $t = 0$ corresponds to the standard Gaussian prior and $t = 1$ corresponds to the clean data distribution.

Let $p_{\text{data}}(\mathbf{z})$ denote the probability density function of the shape prior's training data, which is also the distribution our Flow Matching model is trained to approximate. Meaningful 3D shapes correspond to a high-probability region, such that $p_{\text{data}}(\mathbf{z}) \approx 1$. Conversely, adversarial or meaningless shapes reside in off-manifold regions where $p_{\text{data}}(\mathbf{z}) \approx 0$. During Phase 1, the gradient descent purely minimizes the objective $\mathcal{O}$ without accounting for this underlying data distribution. Consequently, the intermediate latent vector can drift into low-probability regions, meaning $p_{\text{data}}(\hat{\mathbf{z}}^{(k)}) \to 0$.

To formalize how Phase 2 corrects this drift and pulls the latent vector back to regions where $p_{\text{data}} \approx 1$, we analyze the continuous-time dynamics of our flow-based correction. Under the Optimal Transport probability path, the noisy latent state follows $\mathbf{z}_t|\mathbf{z}_1 \sim \mathcal{N}(t\mathbf{z}_1, (1-t)^2\mathbf{I})$. The exact marginal vector field driving this generative flow is given by:

$$\mathbf{v}(\mathbf{z}_t, t) = \frac{\mathbb{E}[\mathbf{z}_1|\mathbf{z}_t] - \mathbf{z}_t}{1-t} \tag{12}$$

Leveraging Tweedie's formula [16], we can express the posterior expectation of the uncorrupted data $\mathbf{z}_1$ precisely through the score function of the smoothed marginal density $p_t(\mathbf{z}_t)$:

$$\mathbb{E}[t\mathbf{z}_1|\mathbf{z}_t] = \mathbf{z}_t + (1-t)^2 \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t) \tag{13}$$

Substituting this expectation into the marginal vector field yields the score-based representation of the flow:

$$\mathbf{v}(\mathbf{z}_t, t) = \frac{\mathbf{z}_t}{t} + \frac{1-t}{t} \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t) \tag{14}$$

The term $\nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)$ ensures the trajectory is continuously pulled toward high-probability regions, allowing $p_{\text{data}}(\mathbf{z}^{(k+1)})$ to approach 1 as $t \to 1$.

However, applying the unguided field $\mathbf{v}$ could discard the optimization progress of Phase 1. By substituting the score-based field into our guided formulation (e.g., using Option B from Eq. 8), the effective integration dynamic becomes:

$$\tilde{\mathbf{v}}_\theta(\mathbf{z}_t, t, c) \approx \underbrace{\frac{\mathbf{z}_t}{t} + \frac{1-t}{t} \nabla_{\mathbf{z}_t} \log p_t(\mathbf{z}_t)}_{\text{Manifold Correction}} - \underbrace{\lambda \nabla_{\mathbf{z}_t} \left(\mathcal{O}\big(\mathcal{D}(\mathbf{z}_t)\big) - c\right)^2}_{\text{Optimization Preservation}} \tag{15}$$

The score-based term acts as a correcting force that maximizes p_{data} to correct the manifold drift, while the guidance term acts as a soft constraint, attracting this correction to the objective c attained in Phase 1.

E Ablation study: guidance during flow matching

We evaluate the necessity of the guidance during flow-matching on the chair volume minimization task (3DShape2VecSet) targeting a 70% reduction. We compare our guided vector field $\tilde{\mathbf{v}}_\theta$ against an unguided baseline $\mathbf{v}_\theta$ during the Phase 2 correction (Equation 6).

As shown in Tab. 6, the unguided field $\mathbf{v}_\theta$ blindly pulls the latent code toward the unconditional data manifold, washing out the progress achieved during the gradient descent phase. Consequently, it requires 737.57 epochs on average and fails to reach the target for 30% of shapes within 1000 epochs. By contrast, our guided flow $\tilde{\mathbf{v}}_\theta$ prevents this wash-out, converging over three times faster (335.18 epochs) with a 100% success rate.

Table 6: Ablation study on the effect of guidance in the flow-based correction (Phase 2). Evaluated on the chair volume optimization task (3DShape2VecSet) targeting a 70% volume reduction.

Method	Epochs to reach 70% Vol. Red.	Success Rate (after 1000 epochs)
Ours (without guidance)	737.57	70%
Ours (with guidance)	**335.18**	**100%**

F Chair volume optimization

In this section, we present a more detailed explanation of our experimental setup. Furthermore, we provide extended results to further demonstrate the scalability and effectiveness of our method.

F.1 Implementation details

In this section, we provide comprehensive details regarding the dataset preprocessing, model architectures, and the exact optimization setup used for the chair volume minimization experiment described in the main text.

Dataset and Preprocessing. We utilize the chair category from the ShapeNet dataset [8]. All meshes are preprocessed to be strictly watertight, canonically aligned, and normalized to fit within a bounding box of $[-1, 1]^3$. We use a split of 2,419 chair shapes for training our shape priors and Flow Matching models.

To evaluate our optimization framework across different levels of latent complexity, we utilize two distinct generative priors and train a specific Flow Matching model for each.

DeepSDF Architecture Details. For a simple, continuous latent representation, we utilize the DeepSDF architecture [50]. Chair shapes are encoded into a single latent vector of dimension $D = 256$. The decoder is a multi-layer perceptron consisting of 8 layers with a hidden dimension of 512. This prior is trained for 2000 epochs using a batch size of 16 and the Adam optimizer, following a multi-step learning rate schedule. The corresponding Flow Matching model operating on this vector space is parameterized by a Residual MLP. Crucially, this model is conditioned on the target shape volume using Classifier-Free Guidance (CFG) [29]. It is trained for 5000 epochs with a batch size of 256 and the Adam optimizer with a constant learning rate of 1e-4.

3DShape2VecSet Architecture Details. For a more complex, structured representation, we follow the 3DShape2VecSet architecture [76]. The generative prior is configured with 12 layers, a hidden dimension of 512, and 8 attention heads of dimension 64. Here, chair shapes are encoded into a latent space represented by a vector set of $N = 256$ vectors, each with a dimensionality of $D = 8$. While the base architecture originally utilizes 512 vectors, we downscaled this set; since our shape prior encodes only a single shape category. The prior is trained for 2000 epochs using a batch size of 32, the AdamW optimizer with a learning rate of 5e-5, and a Warmup Cosine schedule. Its corresponding Flow Matching model operates directly on the latent set and is parameterized by a Transformer-based architecture [76]. Similar to the DeepSDF baseline, it is conditioned on the target volume via CFG. This model is trained for 2000 epochs with a batch size of 32 and the AdamW optimizer with a learning rate of 1e-4, also utilizing a Warmup Cosine schedule.

Differentiable Volume Estimation. The volume of a shape can be directly approximated using its Signed Distance Function (SDF). To optimize the shape's volume while allowing gradients to flow back to the latent representation, we formulate a differentiable volume approximation. During the optimization loop, we query the generative prior to predict the SDF values on a uniform 3D grid $\mathcal{G}$ of resolution 64^3 spanning the canonical bounding box $[-1, 1]^3$. To approximate the binary inside/outside occupancy in a smooth, differentiable manner, we apply a sharp sigmoid function to the negative SDF values. The estimated volume V given a latent code $\mathbf{z}$ is computed as the mean occupancy across all grid points:

$$V(\mathbf{z}) = \frac{1}{|\mathcal{G}|} \sum_{\mathbf{x} \in \mathcal{G}} \sigma\big(-\alpha \cdot f_\theta(\mathbf{x}, \mathbf{z})\big) \tag{16}$$

where $f_\theta(\mathbf{x}, \mathbf{z})$ is the predicted SDF value at point $\mathbf{x}$, σ is the standard sigmoid function, and $\alpha = 100$ is a sharpness factor that scales the SDF to approximate a step function at the surface boundary.

Optimization Hyperparameters The complete set of hyperparameters used in Algorithm 1 for this volume minimization experiment, comparing the DeepSDF and 3DShape2VecSet latent representations, is summarized in Tab. 7.

Table 7: Optimization hyperparameters for chair volume minimization across different latent representations.

Phase	**Parameter**	**DeepSDF**	**3DShape2VecSet**
Gradient Descent	Number of steps (M)	30	15
	Learning rate (η)	0.001	0.005
Flow Correction	Noise level (τ)	0.1	0.2
	Flow Matching start time (t_{start})	0.9	0.8
	Total ODE schedule steps	100	100
	Actual ODE steps (N)	10	10
	CFG scale (s)	1.0	1.0

F.2 Additional results

Additional Qualitative Results. In Figure 6, we show more visual examples for the chair volume optimization. As discussed in the main text, these examples show how baseline methods often break the chair's structure to reach the volume target (for example, by creating floating parts or removing legs), especially when using the more complex 3DShape2VecSet model. For the exact same optimization target, our method successfully preserves the shapes without breaking them. Furthermore, our approach produces higher quality results on 3DShape2VecSet because our framework manages to use the model's full expressivity without suffering from severe manifold drift. This proves that complex models are very powerful, as long as the manifold drift problem is controlled.



Figure 6: Additional qualitative results for chair volume optimization. We compare the original shape (Initial) with different baseline methods and our approach, using both DeepSDF and 3DShape2VecSet. In the figure, V indicates the measured volume of the shape.

Generative Quality vs. Optimization Targets. Figure 7 and 8 show evolution of metrics (FD and KD) for the chair optimization respectively with DeepSDF and 3DShape2VecSet across different volume reduction targets.

On Figure 7, we can see that when using a simpler model like DeepSDF, all methods exhibit similar behavior. In this simpler latent space, manifold drift is not a significant issue, and the metrics remain relatively stable across different volume reduction targets.

Conversely, transitioning to a more complex generative model such as 3DShape2VecSet reveals a stark difference, as shown in Figure 8. As the volume reduction objective becomes more aggressive, the metrics for most baseline methods increase sharply, highlighting the increased difficulty of optimization and a severe manifold drift in highly expressive models. As discussed in the experiments section, SGO also proves to be quite robust in this specific setting, maintaining relatively low metric values. However, this robustness does not generalize to our other experiments involving more complex optimization objectives. Ultimately, our method consistently exhibits the most controlled increase overall, demonstrating its robust effectiveness in mitigating manifold drift and preserving shape quality across different tasks and within highly complex latent spaces.

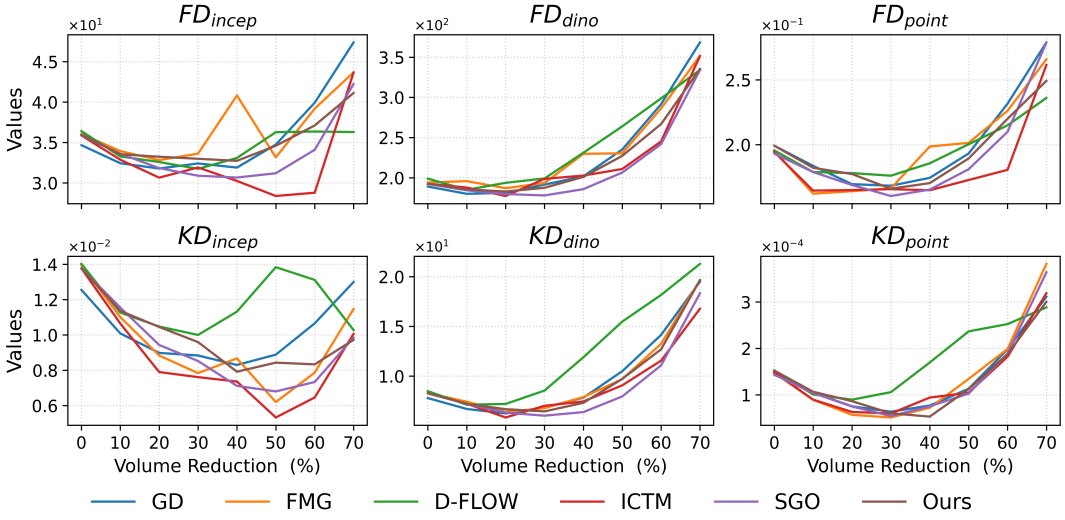

Figure 7: Evolution of generative metrics (FD and KD) for the **chair** optimization with **DeepSDF** across different volume reduction targets.

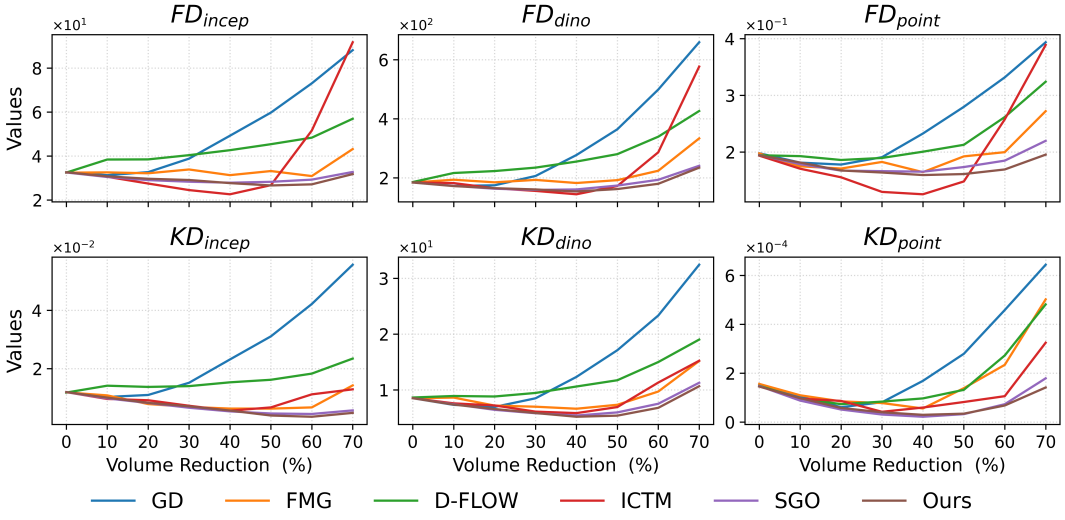

Figure 8: Evolution of metrics (FD and KD) for the **chair** optimization with **Vecset** across different volume reduction targets.

G Car drag optimization

In this section, we present a more detailed explanation of our experimental setup. Furthermore, we provide extended results to further demonstrate the scalability and effectiveness of our method.

G.1 Implementation details

In this section, we provide comprehensive details regarding the dataset preprocessing, model architectures, and the surrogate model used for the aerodynamic drag reduction experiment described in the main text (Section 5.3).

Dataset and Preprocessing. We utilize the car category from the ShapeNet dataset [8]. We preprocess the meshes to be strictly watertight and simulate the resulting dataset of 1036 cars using OpenFOAM to compute the drag. To ensure consistent aerodynamic evaluation, all meshes are aligned and normalized into a bounding box of $[-1, 1]^3$. Finally, we randomly split this dataset into 836 shapes for training the shape prior and 200 shapes for the test set.

Architecture Details. For our generative prior, we follow the 3DShape2VecSet architecture [76], though we adapt its dimensions to better suit our single-category training. The model is configured with 12 layers, a hidden dimension of 512, and 8 attention heads of dimension 64. Car shapes are encoded into a latent space represented by a vector set of $N = 256$ vectors, each with a dimensionality of $D = 8$. While the base architecture originally utilizes 512 vectors, we downscaled this set; since our shape prior encodes only a single shape category. The prior is trained for 2000 epochs using a batch size of 16, the AdamW optimizer with a learning rate of 5e-5, and a Warmup Cosine schedule.

Our Flow Matching model operates directly on this latent set and is parametrized by a Transformer-based architecture following the 3DShape2VecSet Denoising Network architecture [76]. The Flow Matching model is trained with a drag coefficient (C_d) conditioning using Classifier-Free Guidance (CFG) [29]. The model is trained for 2000 epochs with a batch size of 32 and the AdamW optimizer with a learning rate of 1e-4, also following a Warmup Cosine schedule.

Surrogate Model Computing the actual aerodynamic drag coefficient (C_d) during the optimization loop is computationally intractable and non-differentiable via standard CFD solvers. To guide our shape optimization, we train a surrogate model to provide a differentiable estimate of the surface pressure $\hat{p}(\mathbf{x})$ and the resulting drag.

The drag force is computed as the surface integral of the air pressure:

$$\text{drag}_p(\mathcal{S}) = \iint_{\mathcal{S}} -n_x(\mathbf{x}) \cdot p(\mathbf{x})\, d\mathcal{S}(\mathbf{x}) \tag{17}$$

where $p(\mathbf{x})$ is the pressure and $\mathbf{n}(\mathbf{x})$ is the surface normal at point $\mathbf{x}$, with n_x being its component along the x-axis (directed from the front to the back of the car). This value is then normalized to obtain the drag coefficient C_d [47]. Following standard practice for cars, we ignore friction drag as it remains negligible compared to pressure drag.

Our surrogate model is a Graph Neural Network [4] utilizing GraphSage convolution layers [25, 7]. It is trained to predict $\hat{p}(\mathbf{x})$ directly on the mesh vertices. To generate the ground-truth training labels, we simulate the airflow around the dataset of cars using the OpenFOAM (OpenCFD) solver [48], following the well-established DrivAer model setup [67].

The optimization objective is then defined by minimizing the drag coefficient predicted through the surrogate model:

$$\mathcal{L}_{\text{drag}} = C_d(\text{MC}(f_\theta, \mathbf{z})) \tag{18}$$

where MC denotes the Marching Cubes algorithm [37], f_θ is the decoder, and $\mathbf{z}$ represents the latent representation. This setup allows gradients to be backpropagated from the drag loss through the meshing step [53] and the generative prior to update the latent representation.

Optimization Hyperparameters Tab. 8 details the complete set of hyperparameters used in Algorithm 1. These parameters govern the alternating scheme between gradient-based drag minimization and generative flow correction.

Table 8: Optimization hyperparameters for aerodynamic drag reduction with our framework.

Phase	Parameter	Value
Gradient Descent	Number of steps (M)	20
	Learning rate (η)	0.01
Flow Correction	Noise level (τ)	0.2
	Flow Matching start time (t_{start})	0.8
	Total ODE schedule steps	100
	Actual ODE steps (N)	20
	CFG scale (s)	1.0

G.2 Validation of true aerodynamic performance

To validate the physical realism of the car drag reduction experiment, we conducted Computational Fluid Dynamics (CFD) simulations using OpenFOAM [48]. For this evaluation, we simulated 20 car shapes per method, with each shape optimized to target a 30% drag reduction according to the surrogate model.

Table 9: OpenFOAM simulation results at 30% drag reduction (20 cars simulate per method).

Metric	GD	FMG	D-Flow	ICTM	SGO	Ours
True Reduction (%)	19.38	20.63	14.47	20.98	19.27	17.35
Surrogate Error (MAE) $\times 10^2$	4.8	5.2	6.1	4.9	4.5	4.1

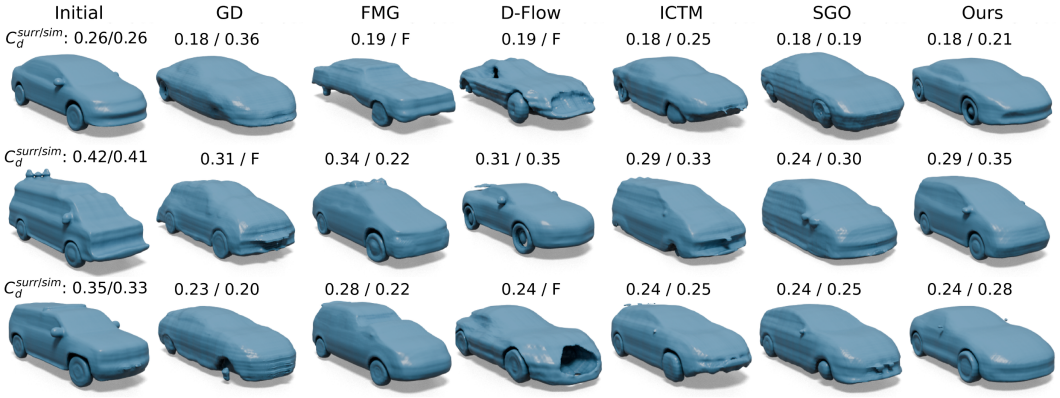

Figure 9: Comparison of optimized car shapes. For each shape, we provide the drag coefficient (C_d) evaluated by the surrogate model and the actual OpenFOAM simulation (surr / sim). 'F' indicates that the OpenFOAM simulation failed, typically due to severe, non-physical geometric distortions preventing proper meshing.

The quantitative results are summarized in Tab. 9. Our method achieves a true physical drag reduction of 17.35%. This gap between the targeted and simulated reduction is expected for two main reasons. First, surrogate models are inherently imperfect approximations, which is standard in this domain. Second, our surrogate model is trained to predict only the pressure component of the drag, whereas the OpenFOAM simulation computes the total drag, which encompasses both pressure and skin friction components.

Despite this expected systematic shift, we observe that certain baselines, such as FMG (20.63%) or SGO (19.27%), exhibit a slightly higher true drag reduction (roughly 3% to 4% better than ours). However, this numerical advantage comes at the complete cost of shape validity. As illustrated in Fig. 9, these baselines achieve a lower physical drag by fundamentally destroying the vehicle's geometry rather than optimizing a realistic car. They devolve into over-smoothed blobs, absorb wheels into the chassis, or introduce severe topological artifacts. While such degenerate, featureless shapes naturally offer less aerodynamic resistance in physics, they no longer represent valid or

functional vehicles. In contrast, our method preserves high-quality, realistic car structures while still achieving substantial aerodynamic improvements.

Furthermore, Tab. 9 indicates that our approach maintains the lowest surrogate prediction error (MAE of 4.1×10^{-2}). We hypothesize that this better alignment between the surrogate predictions and the actual CFD simulations stems from our flow-guided approach. Navigating the optimization closer to the valid data manifold likely keeps the generated shapes in regions where the surrogate remains accurate, whereas manifold drift in baseline methods can lead to out-of-distribution shapes where the surrogate's reliability decreases.

G.3 Additional results

Additional Qualitative Results. We provide additional qualitative comparisons of optimized car shapes in Fig. 10, where each method was tasked with achieving a targeted 30% drag reduction according to the surrogate model. As observed, while all methods successfully reduce the surrogate drag coefficient (C_d) to reach this target margin, the numerical performance of the baselines often comes at the severe cost of geometric integrity. Baseline methods frequently suffer from manifold drift, producing degenerate shapes such as over-smoothed bodies, absorbed wheels, or non-physical topological artifacts (e.g., see GD and D-Flow). In contrast, our approach consistently maintains the structural validity and visual realism of the vehicles while achieving the targeted aerodynamic efficiency.

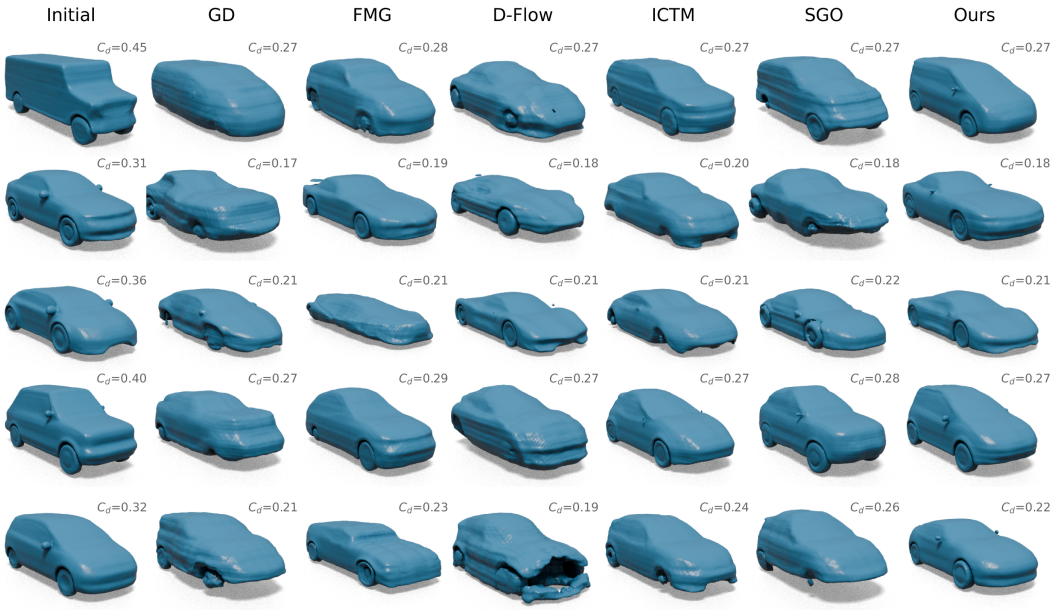

Figure 10: Additional qualitative comparisons of car shapes optimized for a targeted 30% drag reduction. We report the surrogate drag coefficient (C_d) for the initial shapes and the outputs of each optimization method.

Generative Quality vs. Optimization Targets. To further analyze the trade-off between aerodynamic drag reduction and shape quality, we evaluate the evolution of Fréchet Distance (FD) and Kernel Inception Distance (KD) across increasingly strict drag reduction targets. As illustrated in Fig. 11, pushing the optimization to extreme high reductions naturally degrades the generative metrics for all methods, reflecting a deviation from the reference shape distribution. However, our method consistently exhibits a significantly slower degradation, maintaining much lower FD and KD scores compared to the baselines (GD, FMG, D-Flow, ICTM, and SGO) across all evaluated feature extractors (Inception, DINOv2, and PointBert). This quantitative analysis confirms that our framework effectively attracts the optimization trajectory to the manifold of plausible car shapes.

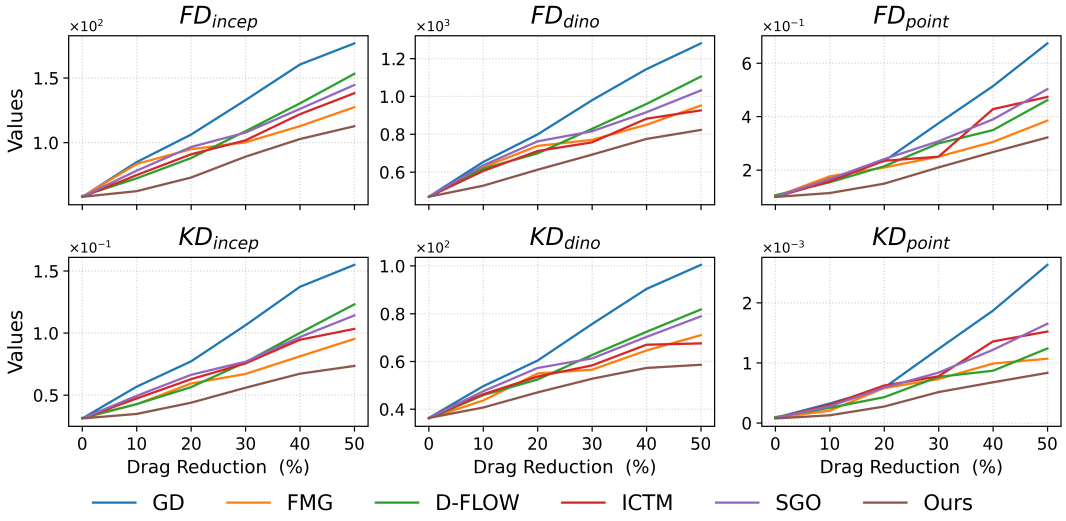

Figure 11: Evolution of metrics (FD and KD) across different drag reduction targets.

H Compliance optimization in Hunyuan3D

In this section, we present a more detailed explanation of our experimental setup. Furthermore, we provide extended results to further show the effectiveness of our method.

H.1 Implementation details

Foundation Model Prior: Hunyuan3D. To demonstrate the scalability of our approach, we use Hunyuan3D, a state-of-the-art generative foundation model capable of synthesizing high-fidelity 3D assets. In our pipeline, the initial 3D shapes are instantiated using the model's image-to-3D capabilities, generating complex geometries conditioned on 2D input images. Unlike category-specific models, Hunyuan3D captures immense structural diversity by representing shapes as continuous 4096×64 latent codes. While this vast, high-dimensional latent space is natively designed for feed-forward generation, our method successfully repurposes it as a highly expressive optimization domain.

Compliance Formulation To reinforce the generated objects against external forces, we adopt the differentiable physics optimization framework introduced by PhysiOpt [75]. Assuming static equilibrium, the object's displacement u is governed by the linear system $K(\pi)u(\pi) = f$, where f represents the external nodal forces and $K(\pi)$ is the stiffness matrix parameterized by the latent variables π. The compliance C is calculated as the work done by the external forces, defined as $C = f^T u$. To differentiably compute the compliance without explicit mesh extraction, PhysiOpt [75] maps the shape's implicit field onto a sparse voxel grid. Each finite element's continuous density ρ_e is derived from the implicit field at its nodes and scales its local stiffness matrix via $K_e = \rho_e K_e^{solid}$. Assembling these yields the global stiffness matrix K, enabling the finite element method (FEM) to compute displacements u from the static equilibrium $Ku = f$ and directly evaluate the shape's compliance. This voxel-based formulation enables seamless, fast gradient-based updates throughout the FEM solve, allowing us to backpropagate the physical loss directly into Hunyuan3D's high-dimensional latent space.

Simulation Setup. We use a finite element voxel resolution of $r = 64$ to accurately resolve the intricate structural details synthesized by Hunyuan3D. For the mechanical simulation, we assume a homogeneous isotropic material typical of rigid plastics or wood, setting the Young's modulus to $E = 0.5$ GPa and the Poisson's ratio to $\nu = 0.3$. Finally, to replicate a natural resting state, the boundary conditions are defined by completely anchoring the bottom 5% of the object's vertical bounding box (Y-axis) to simulate contact with the ground, while external loads are applied to category-specific functional regions.

Quantitative Evaluation on the ABO and BIKED Datasets. To move beyond isolated qualitative examples and rigorously evaluate the generative quality of our optimized shapes, we established a quantitative benchmark using the ABO dataset [13] and the BIKED dataset [52]. We specifically extracted real-world images of chairs and tables from ABO, and bicycles from BIKED, filtering the data to retain only valid and high-quality images. These images were then used to condition Hunyuan3D to generate the initial 3D shapes. To automate the physical optimization across this diverse set of objects, we defined generic forces, systematically targeting the functional surfaces of the objects (i.e., the seat for chairs, the top surface for tables, and the saddle for bicycles). In total, we optimized 100 objects for chairs and tables, and bicycles, across different baseline methods, evaluating them at fixed compliance reduction thresholds. To compute our generative metrics (FID and KID), we compared these optimized outputs against reference sets of unoptimized generated shapes for each category (including 162 chairs, 159 tables, 183 bicycles).

Optimization Hyperparameters Tab. 10 details the complete set of hyperparameters used in Algorithm 1. These parameters govern the alternating scheme between gradient-based compliance minimization and generative flow correction.

Table 10: Optimization hyperparameters for PhysiOpt optimization on Hunyuan3D with our framework.

Phase	**Parameter**	**Value**
Gradient Descent	Number of steps (M)	5
	Learning rate (η)	0.1
Flow Correction	Noise level (τ)	0.4
	Flow Matching start time (t_{start})	0.6
	Total ODE schedule steps	100
	Actual ODE steps (N)	40
	Gradient Guidance Scale (λ)	0.001

H.2 Additional results

ABO Dataset. To provide deeper insights into our quantitative evaluation, we present visual examples of the objects optimized during the experiments. Fig. 12 and Fig. 13 illustrate the structural modifications introduced by our optimization method across chairs and tables, respectively. Our method achieves comparable compliance reduction to baselines while better preserving the original geometry. In contrast, baseline approaches often introduce unnatural artifacts or bumpy surface distortions to minimize compliance.

In addition to the chairs reported in Tab. 3, we report in Tab. 11 the quantitative results for the ABO tables at 50% compliance reduction. Consistent with the other categories, our method outperforms all baselines across all metrics, while D-Flow runs out of memory. It also best preserves the identity of the initial shapes, with an LPIPS of 0.04 ± 0.03, compared to 0.05 ± 0.06 for SGO, 0.08 ± 0.08 for ICTM, 0.12 ± 0.13 for GD, and 0.15 ± 0.14 for FMG.

Table 11: Compliance (C) optimization on ABO tables at 50% reduction. Scaling: $\text{KD}_{\text{incep}}(\times 10^3)$, $\text{FD}_{\text{point}}(\times 10^2)$, $\text{KD}_{\text{point}}(\times 10^5)$.

Method	**FD**$_{\text{incep}}$	**KD**$_{\text{incep}}$	**FD**$_{\text{dino}}$	**KD**$_{\text{dino}}$	**FD**$_{\text{point}}$	**KD**$_{\text{point}}$
GD	81.52	19.31	684.09	32.19	51.98	31.80
FMG	69.18	13.71	530.29	18.93	47.96	17.40
D-Flow			*OOM*			
ICTM	67.11	9.43	506.05	15.56	45.87	9.50
SGO	55.56	4.83	402.84	12.43	37.36	1.80
Ours	**49.97**	**3.59**	**361.04**	**9.06**	**35.07**	**0.50**



Figure 12: Qualitative result on the ABO dataset with chairs. We show the initial shapes (Init) with the applied force (red arrow) and the optimized shapes produced by different methods (GD, FMG, ICTM, SGO, and Ours). The compliance (C) is reported for each shape, where lower values indicate better structural stability.

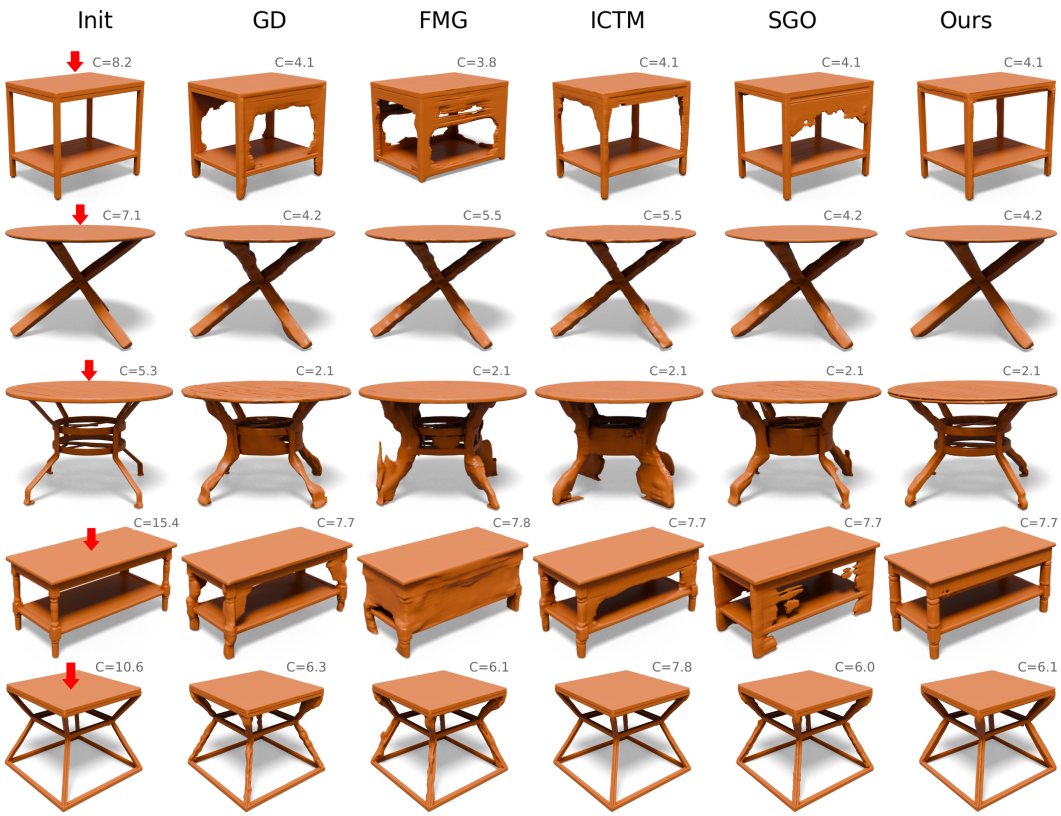

Figure 13: Qualitative result on the ABO dataset with tables. We show the initial shapes (Init) with the applied force (red arrow) and the optimized shapes produced by different methods (GD, FMG, ICTM, SGO, and Ours). The compliance (C) is reported for each shape, where lower values indicate better structural stability.

Generative Quality vs. Optimization Targets. To further analyze the trade-off between physical reinforcement and shape quality, we evaluate the evolution of Fréchet Distance (FD) and Kernel Inception Distance (KD) across increasingly strict optimization thresholds. We perform this analysis for both object categories, as illustrated in Fig. 14 for chairs and Fig. 15 for tables. As expected, pushing the optimization to extreme targets naturally degrades the generative metrics for all methods, reflecting a deviation from the reference shape distribution. However, our method consistently exhibits a significantly slower degradation, maintaining much lower FD and KD scores compared to the baselines (GD, FMG, ICTM, and SGO) across all evaluated feature extractors (Inception, DINOv2, and PointBert). This quantitative analysis confirms that our framework effectively attracts the optimization trajectory to a manifold of plausible shapes for both chairs and tables.

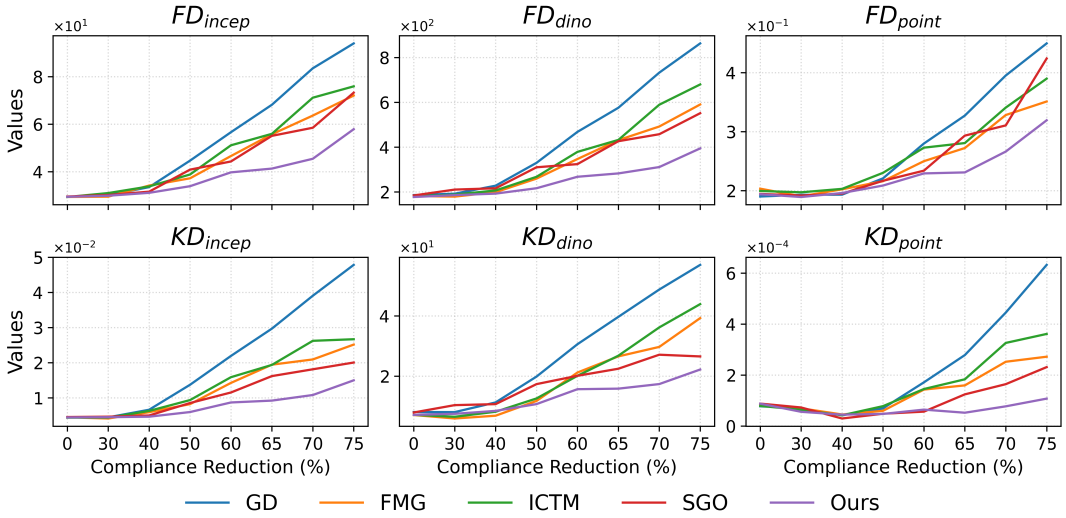

Figure 14: Evolution of generative metrics (FD and KD) for the **chair** category across different optimization thresholds.

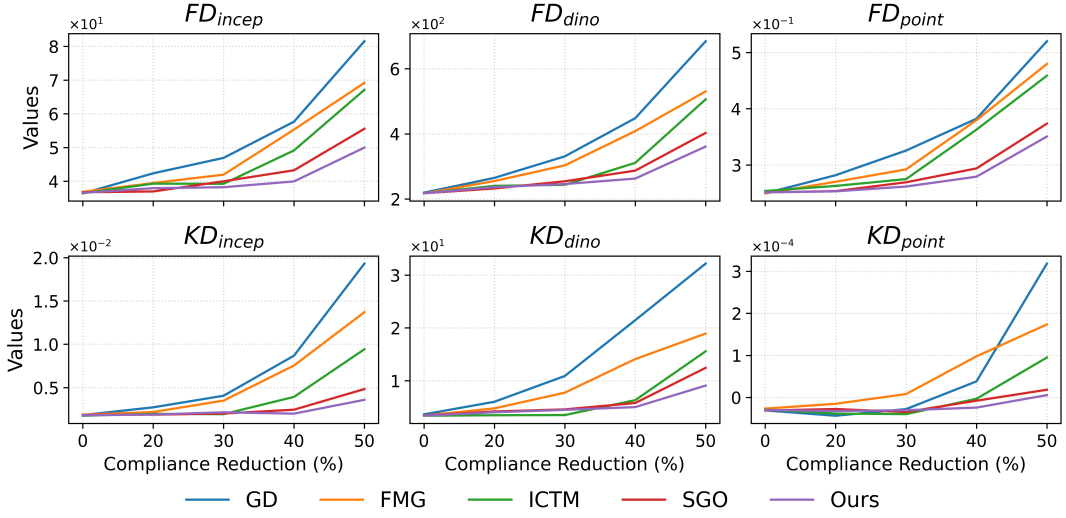

Figure 15: Evolution of generative metrics (FD and KD) for the **table** category across different compliance reduction targets.

BIKED Dataset. In Fig. 16, we present visual examples of bicycles optimized on the BIKED dataset. Similar to the results on the ABO dataset, our method demonstrates superior preservation of the original geometry, adjusting the bicycle frame naturally where needed, while baseline approaches produce blocky, bumpy deformations or missing parts.

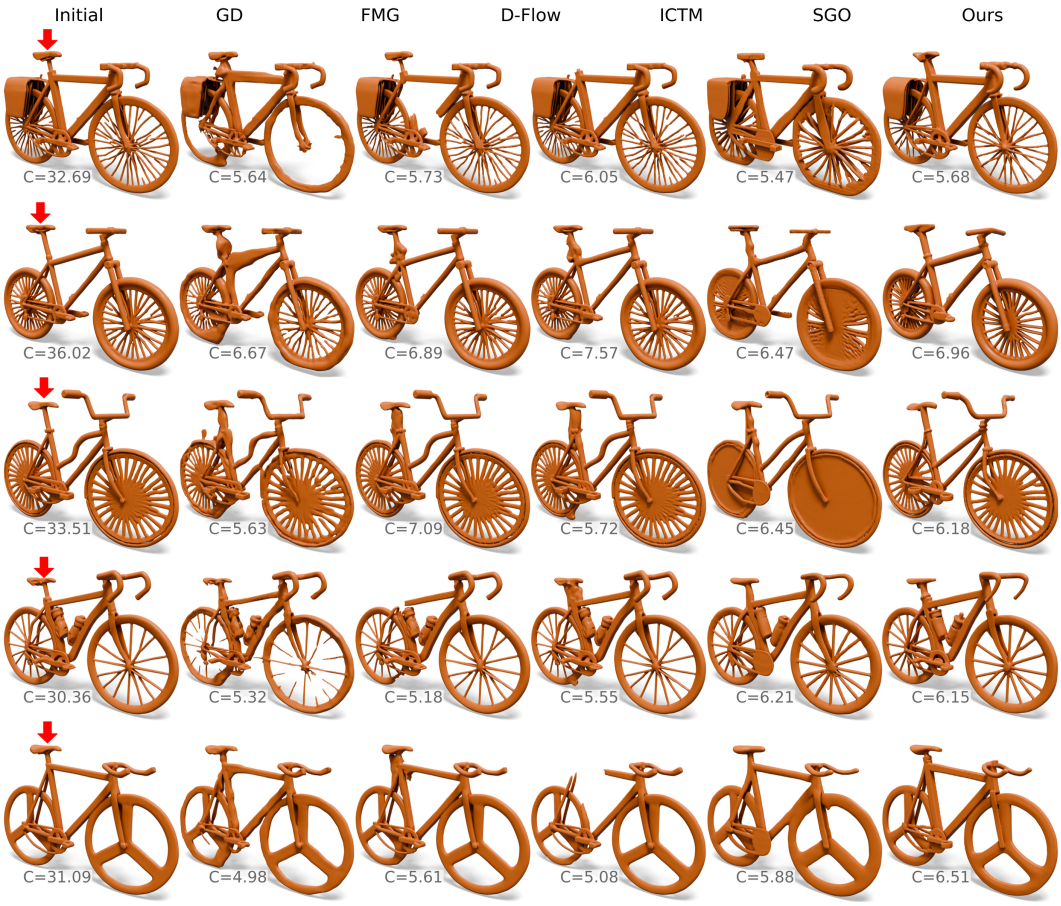

Figure 16: Qualitative result on the BIKED dataset with bicycles. We show the initial shapes (Init) with the applied force (red arrow) and the optimized shapes produced by different methods (GD, FMG, ICTM, SGO, and Ours). The compliance (C) is reported for each shape, where lower values indicate better structural stability.

I Additional objective: 2D silhouette matching

The primary goal of our paper is to introduce a robust framework specifically designed for physical 3D shape optimization. However, to demonstrate that our core idea transfers to other types of objectives that also suffer from manifold drift, we tackle a 2D silhouette matching task on chairs, using a differentiable renderer in a framework conceptually similar to MeshSDF [53].

Methodology. Starting from an initial latent z_0, we decode it into a 3D shape x and extract its binary 2D silhouette $S(x)$ via a differentiable renderer. We then optimize the latent vector to minimize the L_2 distance against a target silhouette S_{target}, derived from a different shape:

$$\mathcal{L} = \|S(x) - S_{\text{target}}\|_2^2 \tag{19}$$

We conducted this experiment on the Chair category, using 3DShape2VecSet as the underlying generative model. Because the objective explicitly matches a geometric target, it acts somewhat like a reconstruction term that structurally grounds the optimization. In contrast, open-ended objectives, like drag reduction, lack this direct geometric anchor. To ensure a fair comparison across all methods, we performed this silhouette matching on 100 source-target pairs and computed the Fréchet and Kernel

metrics on the resulting 3D shapes, after they reached an average per-pixel error of 0.2%, a threshold corresponding to convergence where the source and target silhouettes are visually indistinguishable.

Results. As shown in Tab. 12, our method consistently outperforms the baselines on the chair dataset. This category proves highly susceptible to manifold drift, leading to a significant performance gap between FCSO and standard Gradient Descent (GD). As illustrated in Fig. 17, because the objective only supervises a 2D projection, baselines tend to converge toward adversarial 3D configurations. While these geometries match the target silhouette from the rendering viewpoint, they fail to form valid 3D shapes. For instance, the optimizer frequently detaches components, such as armrests, from the main body to force alignment with the target render. In contrast, our method effectively regularizes the optimization, preserving the structural integrity of the generated shapes.

Table 12: Frechet and Kernel metrics for the 2D silhouette matching task on the Chair category (3DShape2VecSet)

	PointBert		**Inception**		**DINOv2**	
Method	FD ↓ ($\times 10^3$)	KD ↓ ($\times 10^1$)	FD ↓ ($\times 10^4$)	KD ↓	FD ↓	KD ↓
GD	78.33	48.66	403.10	21.39	3.19	5.91
FMG	80.44	55.52	450.29	28.32	3.26	6.29
D-Flow	102.38	63.22	466.12	30.23	3.56	6.98
ICTM	42.99	21.81	211.84	9.72	2.32	1.72
SGO	51.13	25.43	255.21	12.93	2.41	2.19
Ours	**39.45**	**16.51**	**199.06**	**8.99**	**2.00**	**1.55**

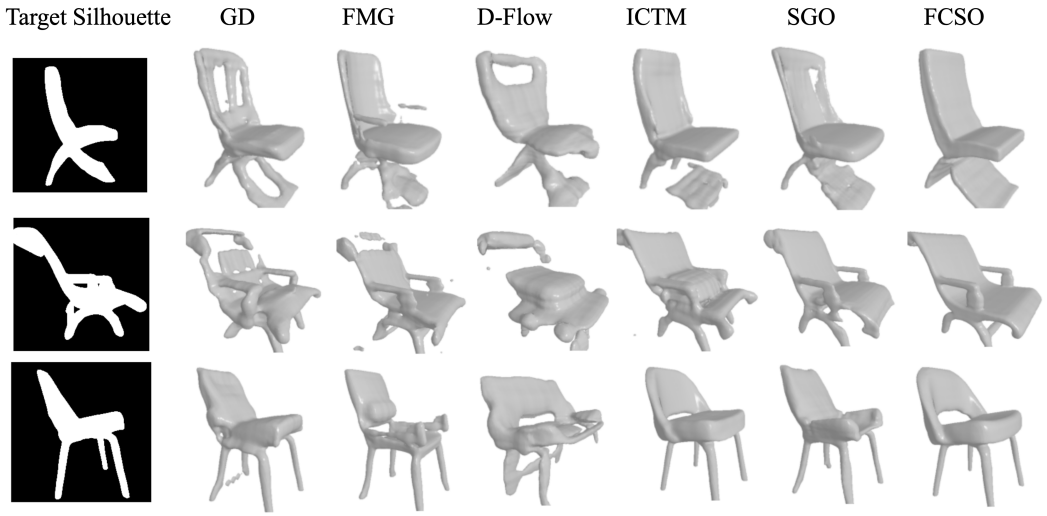

Figure 17: Qualitative results for the 2D silhouette matching experiment. While standard gradient descent often produces adversarial 3D geometries (e.g., floating parts on chairs) to perfectly match the 2D silhouette, our method correctly updates the full 3D structure and stays on the valid shape manifold.

J Limitations: minor artifact in high optimization targets

Under high optimization targets, our method can occasionally introduce minor geometric artifacts, as highlighted by the red bounding boxes in Figure 18. However, even in these rare cases, our approach consistently preserves the structure of the shape. Unlike baselines, a chair remains recognizably a chair, and a car retains its coherent vehicle profile.

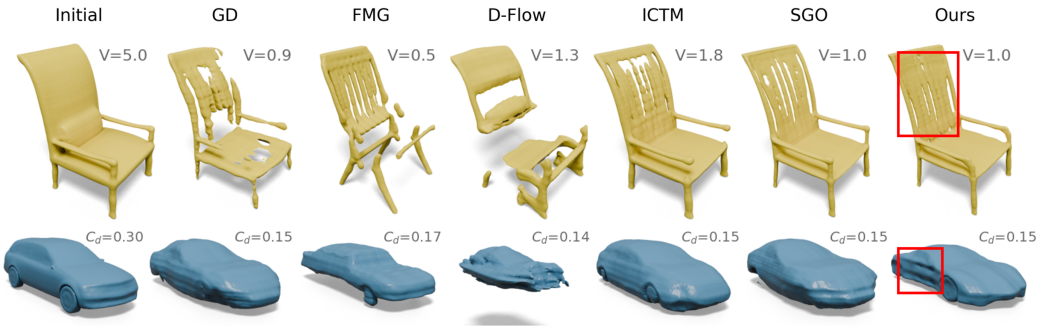

Figure 18: Minor artifacts under high optimization targets across different baselines and our method.