DOI: 10.1111/cgf.14772
EUROGRAPHICS 2023 / K. Myszkowski and M. Nießner
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 2

Interactive design of 2D car profiles with aerodynamic feedback

Nicolas Rosset Guillaume Cordonnier Regis Duvigneau Adrien Bousseau

Inria, Université Côte d'Azur

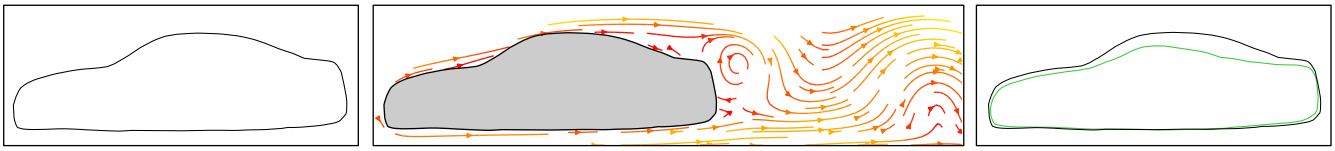

Figure 1: *Our system takes as input the profile of a car (a) and predicts the flow field around the car (b). We perform shape optimization in a latent space of cars to suggest how to improve the aerodynamic properties of the profile (c, here by reducing drag by 11%). Both the fluid flow visualization and the shape optimization are computed within milliseconds, enabling an interactive workflow where designers can iterate between sketching a car profile and evaluating its performance.*

Abstract

The design of car shapes requires a delicate balance between aesthetic and performance. While fluid simulation provides the means to evaluate the aerodynamic performance of a given shape, its computational cost hinders its usage during the early explorative phases of design, when aesthetic is decided upon. We present an interactive system to assist designers in creating aerodynamic car profiles. Our system relies on a neural surrogate model *to predict fluid flow around car shapes, providing fluid visualization and shape optimization feedback to designers as soon as they sketch a car profile. Compared to prior work that focused on time-averaged fluid flows, we describe how to train our model on instantaneous, synchronized observations extracted from multiple pre-computed simulations, such that we can visualize and optimize for dynamic flow features, such as vortices. Furthermore, we architectured our model to support gradient-based shape optimization within a learned latent space of car profiles. In addition to regularizing the optimization process, this latent space and an associated encoder-decoder allows us to input and output car profiles in a bitmap form, without any explicit parameterization of the car boundary. Finally, we designed our model to support pointwise queries of fluid properties around car shapes, allowing us to adapt computational cost to application needs. As an illustration, we only query our model along streamlines for flow visualization, we query it in the vicinity of the car for drag optimization, and we query it behind the car for vortex attenuation.*

Keywords: Interactive design, fluid simulation, surrogate model, shape optimization, neural network, implicit representation

1. Introduction

The design of everyday objects often requires balancing conflicting objectives between aesthetic and functionality. For instance, the profile of a new car should not only look more appealing than previous cars, it should also offer efficient aerodynamic properties. But aerodynamics is hard to guess even for expert designers, while aesthetic is difficult to encode as a mathematical objective for automatic shape optimization [Oth14].

In this paper, we describe an interactive car design system that allows users to sketch the 2D profile of a car and obtain immediate feedback on its aerodynamic properties, including suggestions for improvements (Fig. 1). Users of our system can alternate between creative sketching and aerodynamic evaluation to quickly converge towards a novel, efficient design. While designing 2D profiles is a simplified version of real-world 3D car design, it allows us to study key challenges that would also occur in a more realistic setting (see Sec. 7 for additional discussion on the extension to 3D).

Existing tools to evaluate and optimize car aerodynamics typically rely on expensive simulation over a fine volumetric mesh of the car and its surrounding [Ope07]. While necessary for downstream engineering, such accurate fluid flow simulations are too costly for rapid exploration of early design alternatives. This computational bottleneck has motivated the development of so-called *surrogate models*, which are machine learning models that ap-

proximate costly simulators by being trained on large datasets of simulations pre-computed over a family of shapes of interest [RNA22, LDM22]. We follow this methodology and present a neural-based surrogate model tailored to sketch-based interactive design of car profiles. Specifically, we develop our model to support designers in multiple tasks, from streamline visualization to shape optimization under various criteria.

Achieving our goal raises several unique challenges. First, in contrast to prior work that predicts fluid quantities averaged over time [UB18, CCHT21], we are interested in predicting the instantaneous pressure and velocity fields of vortical flows to display and optimize for dynamic physical criteria. But training a machine learning model to predict flow fields at any time step would require enormous amounts of simulation data. We alleviate this challenge by extracting a *representative frame* for each simulation in our dataset. Observing that 2D flow fields are often periodic, we synchronize the simulations such that their representative frames correspond to the same instant within a phase of the periodic flow field. The resulting instantaneous fields thus react continuously to changes of car shapes, and as such are easier to regress by a neural network.

Our second challenge is to parameterize the input car profile to be fed to our surrogate model. Prior work addresses this challenge by representing the input with parametric curves and surfaces [UB18, BRFF18], such that all shapes share the same number of control points, and that these control points correspond to consistent parts across all shapes. But extracting consistent parametric shapes from arbitrary car profiles sketched by users would require error-prone vectorization or template-matching. Instead, we use a convolutional auto-encoder to learn a latent descriptor of each profile in our dataset. We then train our surrogate model to take as input this latent descriptor and to predict fluid properties of the corresponding profile. We also train our model to predict an implicit surface of the profile from its latent descriptor, and we describe how to compute some aerodynamic criterias from this implicit representation to perform shape optimization in latent space. This approach benefits from the low-dimensional structure of the latent space to ease the optimization task and to prevent it from producing shapes that differ too much from the training data.

Finally, a third challenge we address is the computational efficiency of the prediction over potentially large domains. Several prior methods predict flow fields around a shape using convolutional neural networks (CNNs), such that fluid properties are predicted at each pixel of a finite grid [GLI16, TWPH20, CCHT21]. Yet, we observe that not all parts of the domain are relevant for the applications we target. For instance, for streamline visualization, the velocity along a few particle trajectories suffices; for shape optimization based on the drag, only the pressure field along the car silhouette matters; and for other optimization criteria, such as vortex attenuation, the velocity field is only needed in specific regions of the domains. Inspired by recent work on implicit shape representations [PFS*19, SMB*20], we adapt to these diverse application scenarios by implementing our surrogate model as a multi-layer-perceptron (MLP) that predicts the fluid pressure and velocity for a given car profile at a given spatial position. We then query this network multiple times to get values where needed.

In summary, we make the following contributions:

- A surrogate model based on a MLP to predict fluid flow properties of a given shape, at a given spatial position, along with an optimization that leverages this model to improve aerodynamic properties expressed over an implicit representation of the shape.
- A simple algorithm to synchronize multiple fluid simulations, such that frames extracted from these simulations correspond to similar instants of the periodic flow. These frames form a coherent dataset for training our model to predict instantaneous flow fields from car profiles.
- Based on the above ingredients, an end-to-end interactive pipeline that takes as input the profile of a car sketched by a user, visualizes the flow field around the car and suggests how to modify the profile to improve its aerodynamics.

2. Related work

Our work enables an interactive workflow where designers iterate between creative shape exploration (via a simple sketch-based interface) and simulation-based shape improvement (via streamline visualization and shape optimization). Similar workflows that combine modeling and simulation have been proposed for other application domains, such as furniture design [UIM12], garment design [UKIG11, BSK*16], paper airplane design [UKSI14]. Closest to our work is the system by Umetani and Bickel [UB18], who train a surrogate model based on Gaussian Processes to predict 3D fluid flow around car shapes. Our approach differs on several key aspects. First, they rely on a custom polycube template of the car and its ambient space to obtain a consistent parameterization across their dataset of car simulations. In contrast, we train an encoder to automatically project each car profile into a low-dimensional latent space, which makes our approach applicable to input shapes and representations for which an explicit parameterization is difficult to obtain. Second, they train their surrogate model to predict pressure and velocity averaged over time. In contrast, we describe how to extract synchronized instantaneous observations of the fluid flow across a simulation dataset, allowing us to capture dynamic flow patterns such as vortices. Finally, the system by Umetani and Bickel only provides visualizations of drag, pressure and velocity, letting users explore the shape space by trial-and-error. In contrast, our neural surrogate model allows to perform gradient-based shape optimization and to suggest shape improvements to users.

Approximating costly simulations with machine learning models is common practice in shape optimization [RNA22, LDM22]. In contrast to approaches that predict a single property to be optimized, such as drag or lift [ZSM18], we designed our surrogate model to predict elementary physical quantities, from which we derive optimization objectives for various applications. Many existing approaches rely on convolutional neural networks (CNNs) to predict fluid quantities over the entire spatial domain surrounding the object of interest [GLI16, TWPH20, CCHT21]. Alternatively, Baque et al. [BRFF18] and Durasov et al. [DLDF21] employ graph neural networks to predict pressure over the surface of an object. While graph neural networks are well suited to optimize on-surface quantities like drag, they cannot be used to visualize fluid flow away from the surface or to optimize for flow features like vortices. Our originality is to rely on a multi-layer perceptron (MLP) to provide

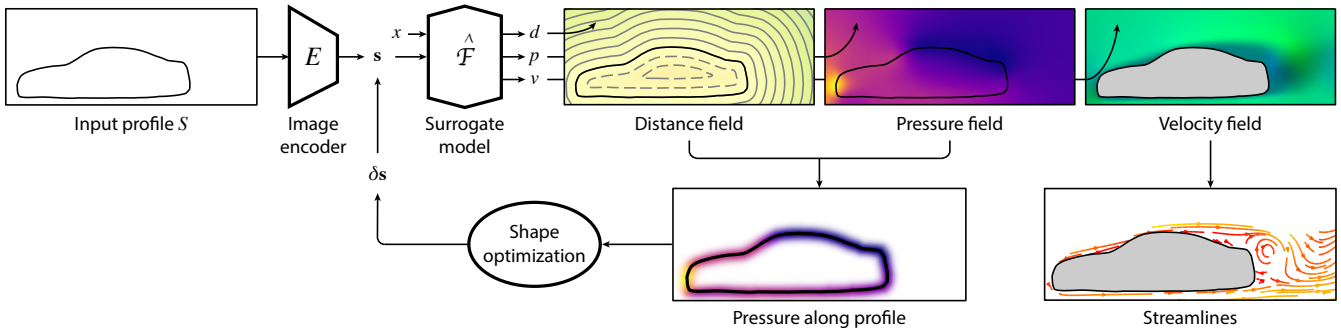

Figure 2: ***Overview.*** *We take as input the profile of a car represented as a bitmap S. We train an encoder E to compute a low-dimensional latent descriptor of this profile* $\mathbf{s}$*, which we feed to our surrogate model along with the spatial coordinates x of the point at which we want to predict fluid flow properties. Our surrogate model $\hat{\mathcal{F}}(\mathbf{s},x)$ predicts the pressure p and velocity $\vec{v}$ at the queried point. We also train our model to predict the distance d of that point to the profile of the shape. We use these quantities for various applications. For visualization, we query the surrogate model along trajectories of particles advected along the velocity field to display streamlines. For shape optimization, we query the surrogate model to predict the pressure and distance field around the car, from which we deduce the drag coefficient. We can also query the surrogate model behind the car to detect and attenuate vortices in the velocity field. Since the entire computation is differentiable, we can use gradient descent to walk in the latent space by small steps* $\delta\mathbf{s}$ *that improve the aerodynamic properties of the profile.*

pointwise predictions, such that we only predict fluid quantities where necessary for the application at hand. While we focus on 2D car profiles, our pointwise approach has a greater potential of scaling to 3D than the regular grids of convolutional networks [GLI16].

We also designed our model to take as input a compact latent descriptor of the shape, allowing optimization in a latent space representative of car shapes. Such latent-space optimization is not possible with methods based on the UNet convolutional neural network [TWPH20], as the skip connections transmit shape information while bypassing the encoder-decoder bottleneck. Latent-space optimization has proven to be a very effective regularization strategy for other ill-posed inverse problems, such as material recovery from few photographs [GLD*19, GSH*20] and shape completion from partial point clouds [PFS*19].

While we rely on deep learning to obtain a fast predictor of fluid flows from pre-computed simulations, other work seeks to leverage deep learning to accelerate fluid solvers [KCAT*19, WBT19, TSSP17, BWDL21]. Recent work also describes how to train neural networks with so-called *physics-informed losses* [RPK19, CLZ*22], with the potential of replacing traditional solvers and alleviating the need for large simulation datasets [WWK21].

3. Problem statement

Aiming to support the design of aerodynamic objects, we propose a sketch-based system that provides fluid simulation feedback to designers at interactive rate. We demonstrate this system on the task of 2D car design, and we illustrate its capabilities by providing visual feedback in the form of streamlines, and optimization feedback in the form of suggestions of shape improvements. Our interactive 2D tool could also serve in an educational context [ZIH*11], for instance to illustrate aerodynamic concepts such as drag and vorticity, and to show their relation to car shapes.

In a domain $\Omega \subset \mathbb{R}^2$, we represent a shape $S \subset \Omega$ as a closed region that defines the inner boundary conditions of a boundary-value problem, which solution is a time-dependant flow field $\mathcal{F}(S,t,x)$. In this work, we focus on the velocity $\vec{v}$ and pressure p of the flow field, in the context of unsteady incompressible viscous flows.

The interaction between the shape and the flow is responsible for several key aerodynamic properties. Some of these properties are defined on the surface of the shape, such as *pressure drag*, which measures the resistance against the motion of the object due to air pressure:

$$\alpha_{\text{drag}}(S) = \frac{1}{T}\int_0^T \oint_\Gamma -p(S,x,t)n_1(S,x)\,\mathrm{d}\Gamma\,\mathrm{d}t, \qquad (1)$$

where $p(S,x,t)$ is the pressure of the flow field at point x and time t, and $n_1(S,x)$ is the horizontal component of the normal on the surface Γ of the shape at x. In practice, pressure drag accounts for around 80% of the total drag of passenger cars [HF93], which is why we ignore other sources of drag such as friction. Aerodynamic studies often use the *drag coefficient*, which differs form the drag force in Eq. 1 by a factor that depends on the projected frontal area of the car (in our 2D case, that would be the height of the vehicle). In our setup, optimizing the drag force or the drag coefficient is similar because we target small changes of the original design, and therefore the size of the car remains mostly constant. Other properties of interest are defined over specific regions of the domain, such as *vorticity*, which measures the degree of rotation of the fluid around a given point, and that designers may seek to reduce behind the car, e.g. to prevent back-projection of rain over the rear glass:

$$\alpha_{\text{vortex}}(S) = \frac{1}{T}\int_0^T \int_{\bar{\Omega}} \|\nabla \times \vec{v}(S,x,t)\|\,\mathrm{d}\bar{\Omega}\,\mathrm{d}t, \qquad (2)$$

where $\bar{\Omega}$ denotes the region of interest.

However, accounting for the time-dependency is prohibitive in practice, both in terms of data size and of difficulty of learning. Therefore, most of the works in the literature resort to time-averaged properties, yielding reduced and smoothed data [UB18, CCHT21]. We overcome this limitation by considering instanta-

neous synchronized observation times, which allows us to predict dynamical phenomena like vortices without significant increase in complexity. We detail in Section 5.2 how we synchronize simulations performed over multiple shapes of a dataset and how we extract a representative observation of each simulation. In what follows, we drop the time dependency and only compute performance measures $\alpha(S)$ over instantaneous observations.

Shape optimization consists in searching for the shape $S^\star$ that offers the best aerodynamic properties, *i.e.*, that minimizes a given performance measure $\alpha(S)$:

$$S^\star = \underset{S}{\operatorname{argmin}}\, \alpha(S). \tag{3}$$

Minimizing Equation 3 raises multiple challenges:

1. Evaluating the fluid flow $\mathcal{F}(S,x)$ around a given shape involves a costly, potentially non-differentiable simulation, which is especially problematic for iterative shape optimization algorithms that typically need to perform multiple evaluations to reach a minimum. We tackle this challenge by replacing the simulator by a fast *surrogate model* $\hat{\mathcal{F}}$, which we implement as a neural network trained to predict fluid flow for a class of car shapes.
2. Minimizing the aerodynamic property using gradient-based optimization algorithms requires the computation of the gradient of $\alpha(S)$ with respect to the shape S. However, the definition of $\alpha_{\text{drag}}(S)$ involves an integral over the surface of the shape itself, for which we do not have an explicit parameterization. We tackle this challenge by expressing the shape as the zero level-set of an implicit function d defined over the entire spatial domain Ω, such that we can rewrite the integral over that domain to drop the dependency on the shape surface.
3. Both our neural-based surrogate and our gradient-based optimization algorithm require that all possible shapes share a common, continuous parameterization. Defining this parameterization is especially challenging for complex objects like cars that exhibit strong variations. We tackle this challenge by encoding the profile of the car in a learned, low-dimensional latent space.

Figure 2 illustrates how these different ingredients interact in our system. We first describe how we perform shape optimization within the latent space of an implicit shape representation (Section 4). We then explain how we model and train the surrogate $\hat{\mathcal{F}}$ to approximate flow fields around car profiles (Section 5).

4. Shape Optimization

Our approach relies on three different representations of the shape of interest, each serving a different purpose. Users provide their design intent in the form of a binary bitmap S. We encode this bitmap into a low-dimensional latent descriptor $\mathbf{s}$, which serves both to condition the surrogate model on this specific shape, and to perform shape optimization by navigating in the latent space of car profiles. Finally, our surrogate model decodes this latent representation into a signed distance field d, which allows us to express drag as an integral over the entire spatial domain Ω, simplifying gradient-based optimization of the shape via automatic differentiation.

Shape optimization in the latent space of car profiles. The input to our system is the profile of a car drawn by the user as a binary mask. We circumvent the difficulty of vectorizing this user input by relying on a *learned* parameterization, in the form of an image encoder E that maps the input profile S into a fixed-size latent descriptor $\mathbf{s}$ (8 dimensions in our experiments). We implement the encoder as a convolutional neural network, which we train jointly with a symmetric decoder D on a standard image reconstruction task according to the binary cross-entropy loss (see details about the network architecture in supplemental materials).

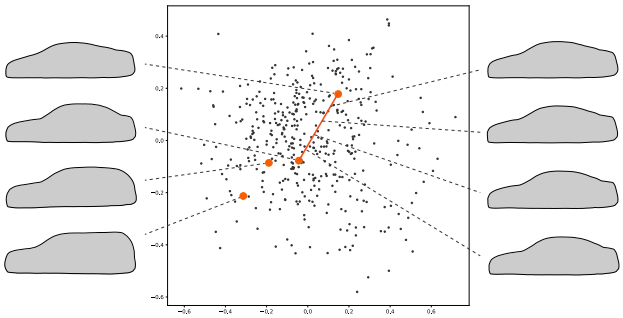

Figure 3: ***Visualization of our learned latent space.*** *This latent space captures the distribution of car profiles in our training set (left). Walking along this space produces a smooth interpolation between car shapes (right).*

Given this parameterization, we now express the optimization problem over the space of latent descriptors:

$$\mathbf{s}^\star = \underset{\mathbf{s}}{\operatorname{argmin}}\, \alpha(\mathbf{s}). \tag{4}$$

In addition to its continuous, low-dimensional structure, this learned latent space behaves like a smooth interpolant between the car profiles in our training dataset, preventing the optimization to produce shapes that do not resemble cars (Figure 3).

Drag computation over an implicit shape representation. Solving Eq. 4 requires converting the latent code $\mathbf{s}$ back to a geometric representation S, and computing interactions $\alpha(S)$ between this geometry and the fluid flow. But evaluating $\alpha_{\text{drag}}(S)$ with Eq. 1 involves computing an integral over the surface of this geometry, for which we lack a consistent parameterization. Our solution to this challenge is to express the shape implicitly as the zero level-set of a signed distance function, $S = \{x | d(S,x) = 0\}$. For a given latent descriptor, we predict this signed distance function over Ω with a neural network $\hat{d}(\mathbf{s},x)$ similar to the one we use to model the flow field $\hat{\mathcal{F}}(\mathbf{s},x)$ around the shape (see Section 5). Thanks to this implicit representation, we can follow Chen et al. [CCHT21] and rewrite Eq. 1 as

$$\alpha_{\text{drag}}(S) = \oint_\Gamma -p(S,x)n_1(S,x)\,\mathrm{d}\Gamma \approx \int_\Omega -p(S,x)n_1(S,x)\delta(d(S,x))\,\mathrm{d}\Omega \tag{5}$$

where $\delta(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{\frac{-x^2}{2\sigma^2}}$ is a smoothed Dirac function modeled as a Gaussian (we fixed $\sigma = 1.2\,\mathrm{d}\Omega$ in all of our experiments). Since $\delta(x)$ quickly vanishes away from the surface of the shape, it effectively restricts the integral to only measure pressure in the vicinity of the shape. Note that expressing S via a signed distance function

(a) Input profile and predicted flow (b) User-selected region of interest (c) Optimized profile for attenuated vorticity

Figure 4: ***Interactive workflow.*** *Our system predicts the presence of vortices right behind this input car profile (a). Users can indicate a region where vortices should be attenuated (b). After shape optimization, vorticity is reduced by 30% in the region of interest (c).*

also equips us with an estimate of the surface normal on and around the surface as $\vec{n}(S,x) \approx \nabla d(S,x)$, which is necessary to compute Eq. 5 at any point of the domain. We provide as supplemental materials an evaluation of this formulation, which achieves an error of 1.7% on average compared to the exact linear integral computed with Eq. 1.

Interactive optimization. Equipped with this reformulation of α_{drag}, we can solve Eq. 4 using gradient descent with line-search to obtain progressive updates of the shape in latent space, denoted as $\delta\mathbf{s}$. We display these updates to users by extracting the zero level-set of the corresponding signed distance field. Users can then decide to accept these modifications, to modify them by re-sketching the profile, or to optimize the profile further by executing a few more gradient descent steps. The same workflow applies to α_{vortex}, for which we offer a simple interface to let users indicate the region in which they want to attenuate vortices. Figure 1 and 4 illustrate shapes designed and optimized within our interactive system.

5. Surrogate Fluid Flow Model

We use neural networks to learn the shape signed distance function $d(S,x)$, as well as the flow field $\mathcal{F}(S,x)$ obtained from a set of pre-computed simulations. While the flow field is defined in the whole domain, we are mostly interested in values within small regions, such as pressure along the surface of the car. This need for localized predictions motivated us to design an architecture that operates on spatial coordinates rather than on complete images, so that both training and inference can be adjusted to specific regions of interest.

The main challenge in training this surrogate model resides in generating a dataset of simulations that exhibit fine details of dynamic vortical flows, while ensuring that these details vary smoothly across simulations of similar shapes so they can be learned by a neural network. But vortices appear at chaotic locations in space and time, yielding very different flow patterns at a given frame of each simulation (Figure 5). Our key observation is that, after an initial transitory period, the fluid flow usually becomes *periodic*. This behavior is classical for such 2D simulations, in particular when using fluid models that damp turbulent high-frequency fluctuations. Such models (e.g. Reynolds-averaged Navier-Stokes) are commonly used in aerodynamic design to achieve reasonable accuracy at a computational time suitable for design iterations [MDH04]. By analyzing this periodicity, we extract a representative frame of the flow field where vortices appear at similar locations across different simulations.

We next describe how we created our dataset of car profiles and the corresponding simulations, and how we detect the periodic regime of each simulation to extract a representative frame that exhibits consistent flow patterns across simulations. We end with a detailed description of the architecture and training of our surrogate model given this data.

5.1. Car profiles extraction and encoding

We created our dataset by extracting the central vertical cross-section of 3D cars from a subset of ShapeNet [CFG*15]. Since the cross-sections might contain holes as well as interior parts, we apply morphological closing and region filling to obtain the outer boundary of the cars (Fig. 6). We adjusted the size of the morphological filter manually for each car, and we rejected any profile that would exhibit defects, such as large missing parts. This process yielded a total of around 2500 profiles, which we used to train our auto-encoder to form pairs $(S,\mathbf{s})$ of profiles and latent codes.

5.2. Fluid flow simulation and synchronization

We next associate each car profile with a representative frame of its flow field.

Fluid simulation. We run a flow simulation to compute the evolution of the velocity and pressure fields around each car profile. We model the fluid by the incompressible Navier-Stokes equations $\frac{\partial \vec{v}}{\partial t} + \vec{v}\cdot\nabla\vec{v} = -\frac{1}{\rho}\nabla p + \nu\Delta\vec{v}$, $\nabla\cdot\vec{v} = 0$. We fix the density of the air to $\rho = 1\,\text{kg/m}^3$. Our low-resolution domain yields additional diffusion effects in the flow simulation, which can be considered as a turbulence model that damps high-frequency fluctuations. This effect is far larger than the physical diffusion, which is why we fix the viscosity of the air ν to 0. While our flow model lacks high-frequency details, we stress that it captures unsteady, dynamic phenomena that are characteristic of flows around vehicles and that should be accounted for during design, such as vortex shedding.

We impose a $12\text{m} \times 5.12\text{m}$ domain, with open right and top boundaries. We scaled the cars to have equal width and we positioned them at a fixed distance from the bottom and left side of this domain. To guide users in drawing cars within that region, we initialize the interface with an average car that users edit by sketching (see accompanying video). We attach our observation frame relative to the car that moves toward the left. In this frame, the car is static and the surrounding air has an initial velocity opposed to the speed of the car, $v(0) = 54\text{km/h}$. The same applies to the boundary conditions for the velocity on the left and bottom sides of the domain, which correspond to the air inlet in front of the car, and to the road moving relatively to the car, respectively. The car itself is implemented as a zero-velocity Dirichlet boundary condition. We

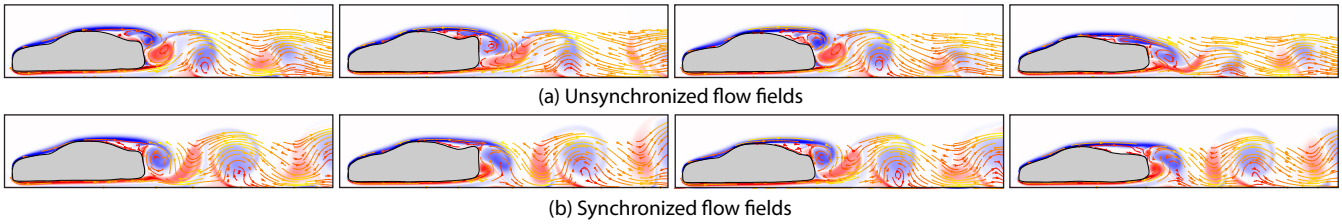

Figure 5: *For a given time frame, vortices appear at different locations behind the car for different simulations (a). By analysing the periodic behavior of the flow field, we synchronize the simulations to extract a representative frame where vortices appear in similar locations (b). These synchronized flow fields are easier to regress by a neural network.*

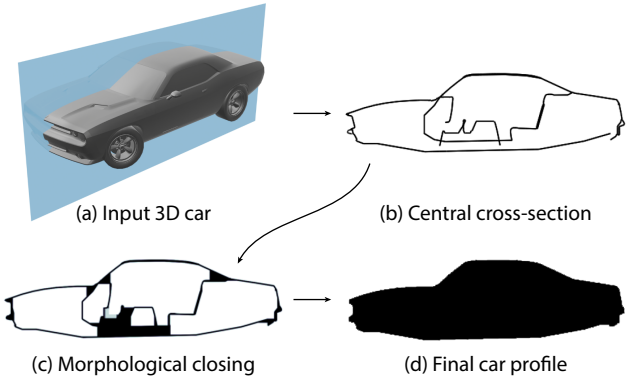

Figure 6: *We extract car profiles from the central vertical section of 3D models from ShapeNet (a,b). We apply morphological closing (c) and region filling (d) to obtain a binary image of the car outer boundary.*

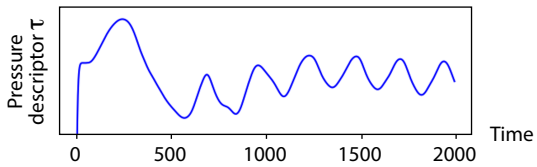

Figure 7: *After an initial transitory period, the average pressure behind the car exhibits periodic oscillations. We detect the period of this signal to synchronize simulations across car profiles.*

set pressure boundaries as Dirichlet for the right and top (open) bounds, and as Neumann for the road (bottom), the left bound, and the car.

We use a 600×256 2D Eulerian staggered grid that stores the pressure at the center of the cells and the horizontal and vertical components of the velocity in the vertical and horizontal edges, respectively. The cell size is 2cm and the time step $1/750$s. We use a solver based on the Stable-Fluids Helmholtz's decomposition method [Sta99], with bilinear interpolation and 3^{rd}-order Runge-Kutta Semi-Lagrangian advection, and a sparse preconditioned conjugate gradient method for the pressure projection.

Synchronization. For each profile, we run a simulation that produces many – possibly thousands of – frames, and we analyse the history of each simulation on-the-fly to determine when to stop it. To do so, we define a descriptor of the pressure of the flow field at a given frame k as $\tau_k = \sum_{k \in \mathcal{S}} p(x_k)$, where $\mathcal{S}$ is a set of 2200 points of coordinates x_k, uniformly distributed in the wake area behind the car. Figure 7 visualizes the evolution of this descriptor along a representative simulation, revealing that it adopts a periodic regime after around 1000 frames in this example.

We follow a method by Vlachos *et al.* [VM05] to detect the periodicity of this signal. At every 50 simulation steps, we extract the N previous values for τ. We choose $N = 600$ as the size of this sliding window because it significantly exceeds the periods we detected on our simulations (around 150 frames in average, up to 400 at most). Let us denote by $\mathcal{T}_k = \{\tau_{k-N}, .., \tau_k\}$ the time sequence of the signal extracted by this procedure at frame k. The method by Vlachos *et al.* first detects candidate periods by running a fast Fourier transform on the signal, and then selects the period that yields the highest auto-correlation value as measured by the Auto-Correlation Function (ACF). This function expresses the self-similarity of the signal, shifted by all possible periods T:

$$ACF_k(T) = \frac{1}{N} \sum_{i \in [k-N,k]} \tau_i \tau_{i-T}. \quad (6)$$

If a period exists, it forms a local maximum of the Auto-Correlation Function. Starting with candidate frequencies that have a high amplitude in the Fourier decomposition of $\mathcal{T}$, the algorithm refines each candidate by performing a hill climbing to the closest local maximum of the ACF. Given the predicted period T_k, we consider that the simulation has reached its periodic regime if the length of the period and its auto-correlation value $ACF_k(T_k)$ did not change significantly over the past 5 estimations, *i.e.*, $T_k \approx T_{k-50} \approx ... \approx T_{k-200}$ and $ACF_k(T_k) \approx ACF_{k-50}(T_{k-50}) \approx ... \approx ACF_{k-200}(T_{k-200})$.

If a period is found, we extract the representative frame as the frame for which the pressure descriptor τ reaches its maximum over the period. We then test if this maximum is close to the one measured over the previous period. If it is not, we perform additional simulation steps until we reach a regime where the maximum value of τ is stable from one period to the next. If no stable period is found after 6000 frames, we discard the profile from the dataset. This algorithm allowed us to extract periodic flow fields for 81% of

the profiles, yielding a dataset of 2013 pairs of profiles and simulations that we split in 1812 pairs for training our surrogate model, and 201 pairs for testing.

Note that this procedure could be easily extended to extract several representative frames per period to offer a more global representation of the flow dynamics.

5.3. Learning the physical values

Our goal is now to train neural networks to reconstruct the car profile and the corresponding flow field from a given latent code.

Network architecture. We use a different neural network per output (distance field d, velocity $\vec{v}$, pressure p), because it allows more generality in the applications and ease the fine-tuning of the network hyper-parameters for each task. Furthermore, these networks might be evaluated at different locations, for instance to predict pressure along the surface of the car, and velocity behind the car. Note that although we could extract the distance field from the input profile, this information is no longer available during shape optimization, for which we have only the latent code. This is why we need to predict the distance field along with the other physical quantities.

Building on the recent developments of *implicit shape representations* [PFS*19], each network takes as input a shape latent code $\mathbf{s}$, along with the coordinate x of the point of interest. We performed a couple of adjustments to the original DeepSDF MLP architecture. First, we expand the input coordinates dimensions using positional encoding [MST*20, TSM*20] to capture higher frequencies in the learned field. We then concatenate the encoded coordinates with the shape latent code to be processed by the MLP. Second, to better preserve the spatial gradients of the fields, we include an optional loss that minimizes the error between the spatial gradient of the predicted field and the spatial gradient of the ground truth field. Since our prediction is performed by a coordinate-based MLP, we compute its gradient via automatic differentiation. In contrast, since the ground truth field is stored as a bitmap in our dataset, we compute its gradient via finite differences. We activate this loss for the network in charge of predicting velocity, whose gradients serve in the computation of vorticity. We also use *SiLU* [EUD17] activation functions instead of *ReLUs* for the velocity network to obtain smoother signals for its gradients. Third, because our dataset is relatively small, we add Lipschitz regularization [LWJ*22] to prevent overfitting and to favor smooth reactions to small latent code perturbations. Finally, we also found that the use of small batches (15k samples) of input coordinates and codes randomized per epoch improves generalization compared to batches of coordinates sharing the same latent code.

Propagating pressure away from the profile. In theory, the computation of drag involves the pressure only at the surface of the car. In practice, our formulation with a smoothed Dirac (Eq. 5) requires values farther from the surface, both inside and outside the shape. Yet, these values evaluated at a small distance from the surface should represent the pressure at the location of the surface. While we could obtain these values by projecting any point of the domain to its closest point along the profile, the resulting pressure field exhibits discontinuities that are difficult to learn. Instead, we achieve a smooth propagation by solving a Laplace equation with pressure values along the profile set as Dirichlet boundary conditions (see supplemental materials for a numerical comparison between these two propagation strategies).

Table 1: ***Quantitative evaluation of fluid flow prediction.*** *Prediction error of our surrogate model measured over our test set. We provide the mean and standard deviation of MSE for the distance, pressure and velocity fields, and the relative MSE of drag expressed as a percentage of its ground truth value. We compare to a CNN baseline for the prediction of distance and pressure, and for the resulting drag.*

	d (avg/std)	p (avg/std)	$\vec{v}$ (avg/std)	α_{drag} (avg/std)
Ours	0.0023 0.0013	0.0189 0.0107	0.0454 0.0153	7.15% 5.74%
CNN	0.0037 0.0016	0.0251 0.0148		15.2% 10.1%

6. Results and discussion

Fluid flow prediction. We first evaluate the ability of our surrogate model to predict fluid flow quantities suitable for shape optimization. Since many prior methods rely on a CNN to predict similar quantities [GLI16, TWPH20, CCHT21], we compare our approach to a CNN baseline that takes as input a latent descriptor $\mathbf{s}$ and decodes it as 256×600 distance and pressure fields, from which we evaluate drag using Eq. 5 (see supplemental materials for architectural details). We configured this CNN to have roughly the same number of parameters as our MLP (200k vs. 150k parameters, respectively).

Figure 8 provides a visual comparison of our predictions of signed distance d, pressure p and velocity $\vec{v}$ against the respective ground truth fields, for representative profiles of our test set. Our surrogate model captures the overall flow field patterns, even though it lacks fine details.

Table 1 quantifies the accuracy of our model and of the CNN baseline in terms of Mean Squared Error (MSE), showing that our model is twice more accurate than the CNN on the end drag prediction. The poor performance of the CNN is likely due to the loss of spatial information within the encoder bottleneck. While prior work alleviated this issue by using a UNet architecture with skip connections to transmit high-frequency spatial information, this solution prevents using the encoder latent code as a descriptor of the shape for latent-space optimization. In contrast, by complementing the latent code with the coordinates of the point of interest, our MLP-based approach is better equipped to learn spatially-varying information. We provide as supplemental materials the histogram of error for drag, showing that it exhibits a Gaussian-like shape, which is why we summarize it with mean and variance in Table 1.

Shape optimization. Figure 9 illustrates results of our shape optimization for drag and vorticity attenuation. For each result, we used our user interface to perform a few gradient descent steps such that

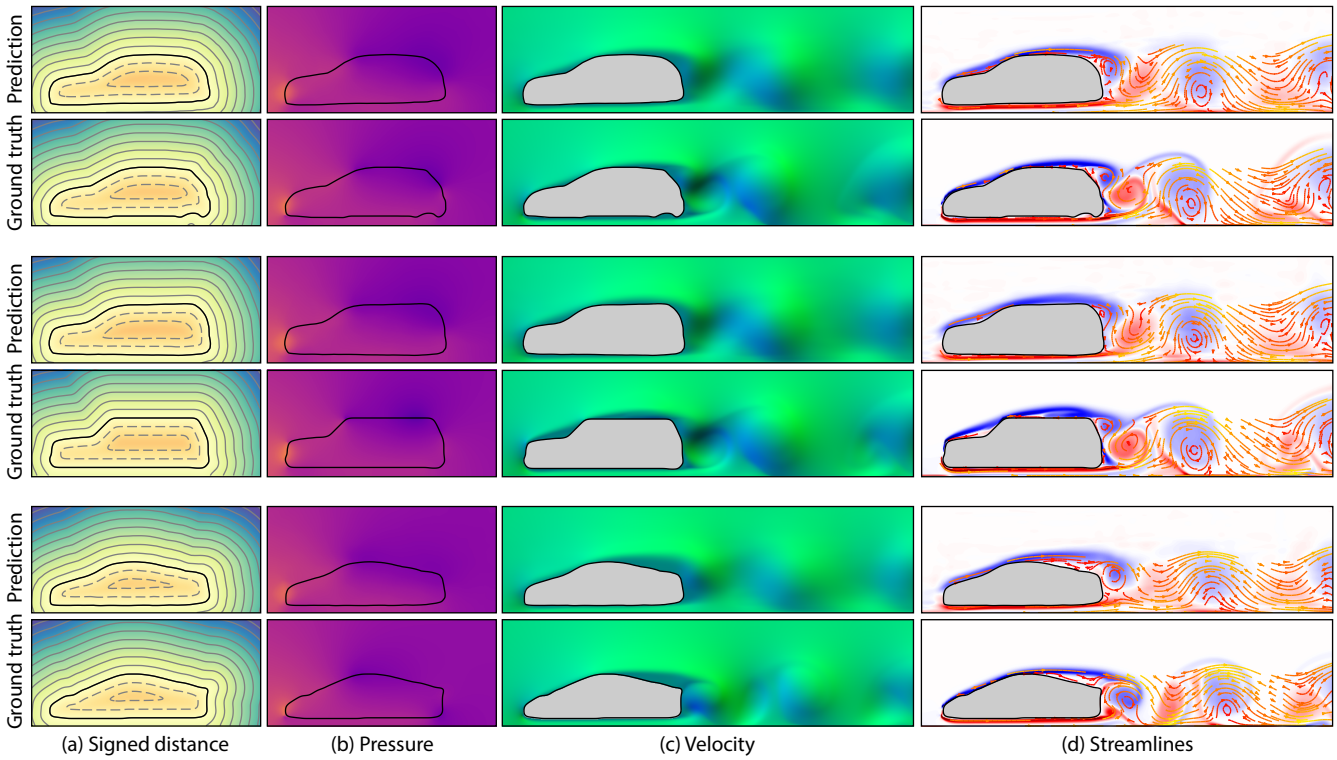

Figure 8: ***Qualitative evaluation of fluid flow prediction.*** *Comparison between the distance, pressure and velocity fields predicted by our surrogate model, and the corresponding ground truth fields. We also include a streamline visualization to ease visual comparison of the velocity. Our model reproduces the overall behavior of the fluid flow, albeit smoothing out fine details.*

the profile improves yet stays close to the input. The accompanying video showcases a few design sessions recorded with this interface, where users sketch car profiles, interactively evaluate the resulting fluid flow, and improve their designs with shape optimization.

We evaluated the improvement in drag achieved by our approach quantitatively by running 20 gradient descent steps on a random set of 25 test profiles, and by comparing the gain predicted by our method to the effective gain measured by running a precise fluid simulation on the output profile. For an average gain of 20.53%, our prediction differs by 11.02% from the effective gain, demonstrating the ability of our method to suggest effective shape improvements.

Timings. Providing interactive feedback requires an efficient surrogate model. We now evaluate the inference time of our neural networks for different target problems, and compares them to the CNN baseline. All timings were measured on a computer equipped with an Intel Xeon E5-2650 CPU (64GB RAM), and a single Nvidia RTX A5000 with 24GB of dedicated memory.

Table 2 summarizes the timings of our different problems, averaged over $1,000$ inferences. For each case, we leverage the implicit nature of our networks to evaluate them over a small set of spatial coordinates. In practice, we first define a large set of regularly-spaced samples that cover the whole domain, with a resolution equivalent to the output of the fluid simulation (600x256), and we select a subset of these points based on problem-specific criteria.

Table 2: ***Time efficiency of our networks.*** *Time required to solve each problem: drag evaluation, drag optimization, vorticity evaluation within a masked region, vorticity optimization, and computation of streamlines. We also provide timings of the CNN baseline for drag computation and optimization.*

	Drag	Drag opt.	Vort.	Vort. opt.	Str.lines
Ours	8.5 ms	26 ms	11 ± 2 ms	21 ± 7 ms	124 ms
CNN	39 ms	98 ms			

The estimation of drag requires the evaluation of pressure near the profile, as well as the evaluation of the signed distance function and of its horizontal normal computed analytically with automatic differentiation. The vicinity to the profile is localized by the smoothed Dirac function introduced in Eq. 5, which is negligible at distances greater than 5σ from the shape. This criterion allows us to select $6,000$ points on average, from which the total computation for the drag estimation requires 8.5 ms. In comparison, computing the same quantity over the entire pressure and distance fields predicted by the CNN – using finite differences for the evaluation of the normals – takes 39 ms.

Suggesting a change to the profile that will improve drag also requires computing the analytical gradients of the drag with respect to the input latent code. Because the profile typically drifts away from

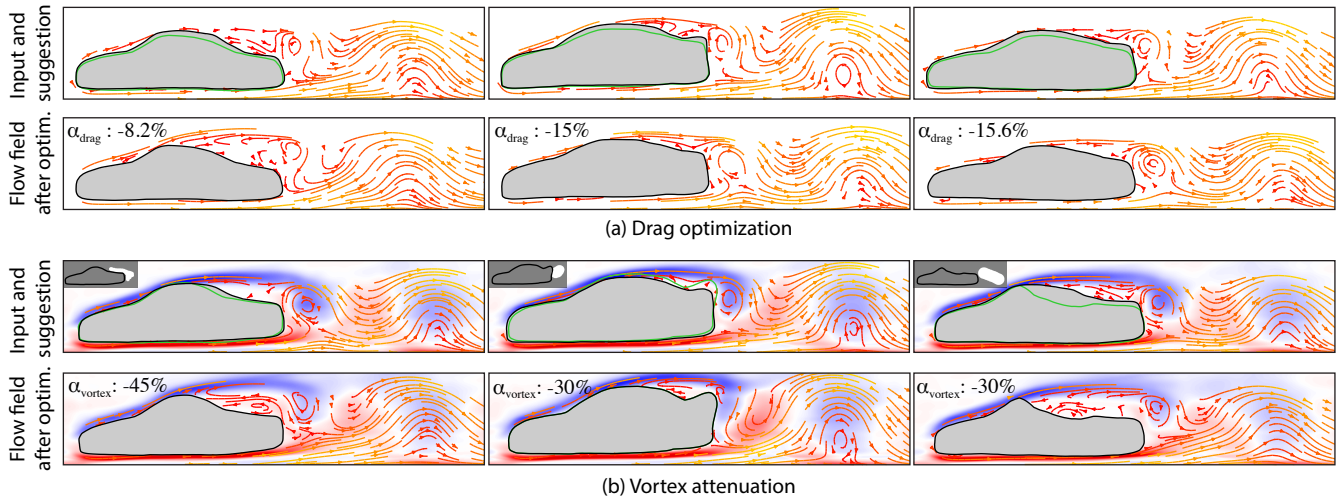

Figure 9: ***Application to shape optimization.*** *Our method suggests effective shape modifications to reduce drag (a) or to attenuate vorticity (b). For each profile, we display the suggested improvement with a green outline (top) and then visualize the impact of the shape optimization on the flow field (bottom).*

its initial state during successive steps of gradient descent, we potentially need to perform this computation for points that are further away from the input profile. Assuming a maximum displacement of 10% of the horizontal extent of the domain – which correspond to the average height of the cars in our dataset – we select 37,000 samples around the input profile on average. These points allow computing the suggestion in around 26 ms. The same task, for the CNN, would take 98 ms.

The vortex attenuation is prescribed by the user in a small region behind the car, and requires automatic differentiation on the horizontal and vertical components of the velocity to compute the orthogonal spatial derivatives. The selected region usually covers between 1,000 samples for small regions to 20,000 points for regions close to the size of the car itself. For these, the vorticity estimation takes from 9.0 to 13.4 ms. The corresponding suggestion for shape improvement takes from 15.0 to 28.6 ms.

One possibility to generate the streamlines is to sample around 200 seed points along and behind the car profile, and then iteratively displace the points in the direction of the evaluated velocity. In practice, this procedure requires around 50 successive evaluations of the network, for a total time of 124 ms. Note that we did not include in these timings the post-processing step that cuts overlapping streamlines. Since the GPU is scarcely used during this sequential process, the timing would not increase significantly for more seed points.

Finally, our networks are relatively fast to train: 16h for the pressure (learning rate 1×10^{-3}, 100 epochs), 15h for the signed distance function that has one layer less (learning rate 7.5×10^{-4}, 100 epochs). The velocity network is slower to train (8d, 3h) because of the additional gradient loss, and because we used a smaller learning rate with more epochs (learning rate 1×10^{-5}, 265 epochs). While we measured these timings by training the networks with points sampled uniformly in the simulation domain, training could be accelerated by adaptively sampling the input coordinates around and behind the car – where accurate prediction matters most for our applications.

Limitations. Our prototype interface demonstrates the potential of our implicit surrogate model to provide various forms of feedback to designers, from streamline visualization all the way to suggestions of shape improvements for various aerodynamic properties. However, our current implementation is limited by the relatively small dataset we used for training our model. In particular, shape optimization might exit the densely sampled regions of our latent space and produce implausible profiles, as illustrated in Fig. 10. Fortunately, our interactive workflow allows users to follow the optimization trajectory step-by-step and stop it before the profile degrades. Extending our dataset should offer designers greater flexibility by navigating in a more diverse space of car shapes.

Our quantitative evaluation also revealed that our predictions differ from ground truth by up to 7% on average, preventing our system to make relevant suggestions for subtle shape changes. We hope that our approach will benefit from recent progress on neural-based fluid flow prediction, for instance by training the surrogate model with physics-aware losses [RPK19], or by treating the predicted fluid flow as an initialization for a precise, differentiable solver [HTK20].

Finally, since our strategy to synchronize the simulations in our dataset relies on the periodicity of 2D fluid flows, it would not apply to more complex, chaotic flows.

7. Conclusion

Design and engineering have long been considered as sequential activities, where the role of the engineer was to *rationalize* the creative propositions of designers to make them physically efficient. We proposed a system that tightly integrates physical simulation

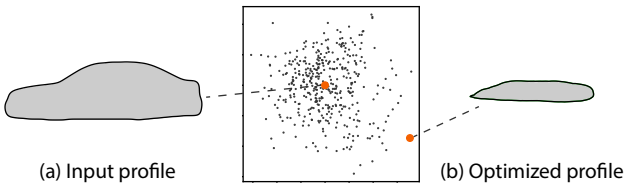

Figure 10: ***Limitation.*** *Shape optimization might travel outside of the relevant region of the latent space, producing profiles that do not resemble cars anymore (b). We alleviate this issue by letting users iterate the optimization until they find a proper trade-off between aerodynamic improvement and shape preservation.*

within the workflow of car profile design, such that designers can benefit from immediate feedback on the aerodynamic performance of the profiles they sketch. We achieved this goal by leveraging the ability of neural networks to encode complex visual signals – such as user-sketched car profiles, as well as to generate complex spatially-varying signals – such as flow fields.

Focusing on 2D car profiles allowed us to avoid the cost of generating a large training set of 3D simulations, and yet helped us identify key ingredients to offer real-time feedback and suggestions for aerodynamic design tasks. We hope that these ingredients will inspire research towards a 3D design tool. Specifically:

- We believe that processing the input with a learned shape encoder is a promising approach to accommodate shape representations that are difficult to parametrize consistently, such as 3D sketches [Gra17], meshes, or point clouds.
- We showed that a coordinate-based MLP is more efficient than a CNN because it can be adaptively sampled in the areas of interest, while CNNs are executed on the entire simulation domain (Sec. 6). We conjecture that adaptive sampling will yield even better performances in 3D where CNNs grow with cubic complexity.
- We showed that dynamic flow features like vortices can be learned if the surrogate model is trained with frames that are coherent across simulations. We leveraged the periodic behavior of 2D flow fields to identify these frames. Depending on the turbulence model used, our strategy might not be as effective in 3D where flows are more chaotic. Identifying similar frames, or features, across chaotic 3D simulations is a challenging direction for future research to go beyond learning time-averaged flow fields.

Acknowledgements

We are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support. We also thank Jingwei Tang for implementing the fluid solver. This work was supported by the European Research Council (ERC) starting grant D3 (ERC-2016-STG 714221) and research and software donations from Adobe Inc.

References

[BRFF18] BAQUE P., REMELLI E., FLEURET F., FUA P.: Geodesic convolutional shape optimization. *International Conference on Machine Learning* (2018). 2

[BSK*16] BARTLE A., SHEFFER A., KIM V. G., KAUFMAN D. M., VINING N., BERTHOUZOZ F.: Physics-driven pattern adjustment for direct 3d garment editing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 35*, 4 (jul 2016). 2

[BWDL21] BAI K., WANG C., DESBRUN M., LIU X.: Predicting high-resolution turbulence details in space and time. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 40*, 6 (2021), 1–16. 3

[CCHT21] CHEN L.-W., CAKAL B. A., HU X., THUEREY N.: Numerical investigation of minimum drag profiles in laminar flow using deep learning surrogates. *Journal of Fluid Mechanics 919* (2021), A34. doi:10.1017/jfm.2021.398. 2, 3, 4, 7

[CFG*15] CHANG A. X., FUNKHOUSER T., GUIBAS L., HANRAHAN P., HUANG Q., LI Z., SAVARESE S., SAVVA M., SONG S., SU H., XIAO J., YI L., YU F.: *ShapeNet: An Information-Rich 3D Model Repository*. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 5

[CLZ*22] CHU M., LIU L., ZHENG Q., FRANZ E., SEIDEL H.-P., THEOBALT C., ZAYER R.: Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics (Proc. SIGGRAPH) 41*, 4 (aug 2022), 119:1–119:14. 3

[DLDF21] DURASOV N., LUKOYANOV A., DONIER J., FUA P.: Debosh: Deep bayesian shape optimization, 2021. 2

[EUD17] ELFWING S., UCHIBE E., DOYA K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017. 7

[GLD*19] GAO D., LI X., DONG Y., PEERS P., XU K., TONG X.: Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Transactions on Graphics (Proc. SIGGRAPH) 38*, 4 (2019). 3

[GLI16] GUO X., LI W., IORIO F.: Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), Association for Computing Machinery. 2, 3, 7

[Gra17] GRAVITYSKETCH: Gravity sketch. https://www.gravitysketch.com/, 2017. 10

[GSH*20] GUO Y., SMITH C., HAŠAN M., SUNKAVALLI K., ZHAO S.: Materialgan: Reflectance capture using a generative svbrdf model. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 39*, 6 (2020). 3

[HF93] HIMENO R., FUJITANI K.: Numerical analysis and visualization of flow in automobile aerodynamics development. In *Computational Wind Engineering 1*, Murakami S., (Ed.). Elsevier, Oxford, 1993, pp. 785–790. 3

[HTK20] HOLL P., THUEREY N., KOLTUN V.: Learning to control pdes with differentiable physics. In *International Conference on Learning Representations* (2020). 9

[KCAT*19] KIM B., C. AZEVEDO V., THUEREY N., KIM T., GROSS M., SOLENTHALER B.: Deep Fluids: A Generative Network for Parameterized Fluid Simulations. *Computer Graphics Forum (Proc. Eurographics) 38*, 2 (2019). 3

[LDM22] LI J., DU X., MARTINS J. R.: Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences 134* (2022). 2

[LWJ*22] LIU H.-T. D., WILLIAMS F., JACOBSON A., FIDLER S., LITANY O.: Learning smooth neural functions via lipschitz regularization. In *ACM SIGGRAPH Conference Proceedings* (2022). 7

[MDH04] MUYL F., DUMAS L., HERBERT V.: Hybrid method for aerodynamic shape optimization in automotive industry. *Computers & Fluids 33*, 5 (2004), 849–858. 5

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 7

[Ope07] OpenCFD: OpenFOAM - The Open Source CFD Toolbox. http://www.openfoam.com, 2007. 1

[Oth14] Othmer C.: Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry 4* (2014), 1–23. 1

[PFS*19] Park J. J., Florence P., Straub J., Newcombe R., Lovegrove S.: Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019). 2, 3, 7

[RNA22] Regenwetter L., Nobari A. H., Ahmed F.: Deep Generative Models in Engineering Design: A Review. *Journal of Mechanical Design 144*, 7 (03 2022). doi:10.1115/1.4053859. 2

[RPK19] Raissi M., Perdikaris P., Karniadakis G. E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics 378* (2019), 686–707. 3, 9

[SMB*20] Sitzmann V., Martel J. N., Bergman A. W., Lindell D. B., Wetzstein G.: Implicit neural representations with periodic activation functions. In *Proc. NeurIPS* (2020). 2

[Sta99] Stam J.: Stable fluids. In *Proc. Annual Conference on Computer Graphics and Interactive Techniques* (1999), SIGGRAPH '99. 6

[TSM*20] Tancik M., Srinivasan P. P., Mildenhall B., Fridovich-Keil S., Raghavan N., Singhal U., Ramamoorthi R., Barron J. T., Ng R.: Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS* (2020). 7

[TSSP17] Tompson J., Schlachter K., Sprechmann P., Perlin K.: Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning* (2017), PMLR, pp. 3424–3433. 3

[TWPH20] Thuerey N., Weissenow K., Prantl L., Hu X.: Deep learning methods for reynolds-averaged navierâĂŞstokes simulations of airfoil flows. *AIAA Journal 58*, 1 (2020), 25–36. doi:10.2514/1.J058291. 2, 3, 7

[UB18] Umetani N., Bickel B.: Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (Proc. SIGGRAPH) 37*, 4 (2018). 2, 3

[UIM12] Umetani N., Igarashi T., Mitra N. J.: Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31*, 4 (2012). 2

[UKIG11] Umetani N., Kaufman D. M., Igarashi T., Grinspun E.: Sensitive couture for interactive garment modeling and editing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 30*, 4 (2011). 2

[UKSI14] Umetani N., Koyama Y., Schmidt R., Igarashi T.: Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 33*, 4 (2014). 2

[VM05] Vlachos Michail Yu Philip C. V.: On Periodicity Detection and Structural Periodic Similarity. *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM)* (2005). doi:10.1137/1.9781611972757.40. 6

[WBT19] Wiewel S., Becher M., Thuerey N.: Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum* (2019), vol. 38, Wiley Online Library, pp. 71–82. 3

[WWK21] Wandel N., Weinmann M., Klein R.: Learning incompressible fluid dynamics from scratch - towards fast, differentiable fluid models that generalize. International Conference on Learning Representations. 3

[ZIH*11] Zhu B., Iwata M., Haraguchi R., Ashihara T., Umetani N., Igarashi T., Nakazawa K.: Sketch-based dynamic illustration of fluid systems. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 30*, 6 (2011). 3

[ZSM18] Zhang Y., Sung W. J., Mavris D. N.: *Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient*. 2018. 2