Fast simulation of wind-obstacle interactions. Applications to desertscape modeling and car design

Nicolas Rosset

Simulation rapide d'interactions vent-obstacle Application à la modélisation de paysages désertiques et à la conception de voiture

Nicolas ROSSET

Centre Inria d'Université Côte d'Azur - Équipe Graphdeco

Présentée en vue de l'obtention du grade de docteur en Informatique d'Université Côte d'Azur

Dirigée par : Adrien Bousseau

Co-encadrée par : Guillaume Cordonnier et Régis Duvigneau

Soutenue le : 10 Décembre 2024

Devant le jury composé de :

Antoine Rousseau, Directeur de Recherche,
Antenne Inria de l'Université de Montpellier

Eric Paquette, Professeur,
École de Technologie Supérieure de Montréal

Julie Digne, Directrice de Recherche,
Université Claude Bernard - Lyon 1

Chris Wojtan, Professeur,
Institute of Science and Technology Austria (ISTA)

Kiwon Um, Maitre de Conférences,
Telecom Paris

Adrien Bousseau, Directeur de Recherche,
Centre Inria d'Université Côte d'Azur

Guillaume Cordonnier, Chargé de Recherche,
Centre Inria d'Université Côte d'Azur

Régis Duvigneau, Directeur de Recherche,
Centre Inria d'Université Côte d'Azur

Inria

Simulation rapide d'interactions vent-obstacle Application à la modélisation de paysages désertiques et à la conception de voiture

---

Fast simulation of wind-obstacle interactions Applications to desertscape modeling and car design

Jury:

Président du jury / President of the jury
Antoine Rousseau, Directeur de Recherche, Antenne Inria de l'Université de Montpellier

Rapporteurs / Reviewers
Éric Paquette, Professeur, École de Technologie Supérieure de Montréal
Julie Digne, Directrice de Recherche, Laboratoire d'Informatique en Image et Systèmes d'information (LIRIS), Université Claude Bernard - Lyon 1

Examinateurs / Examiners
Kiwon Um, Maître de Conférences, Telecom Paris
Chris Wojtan, Professeur, Institute of Science and Technology Austria (ISTA)

Co-encadrants / Cosupervisors
Guillaume Cordonnier, Chargé de Recherche, Centre Inria d'Université Côte d'Azur
Régis Duvigneau, Directeur de Recherche, Centre Inria d'Université Côte d'Azur

Directeur de thèse / Thesis supervisor
Adrien Bousseau, Directeur de Recherche, Centre Inria d'Université Côte d'Azur

Acknowledgements

Dijo que su libro se llamaba el Libro de Arena, porque ni el libro ni la arena tienen ni principio ni fin.

-Jorge Luis Borges, El libro de arena

Mener à bien une thèse nécessite d'être bien entouré. Pour cette raison, je voudrais remercier les personnes qui ont rendu cela possible.

Tout d'abord, je voudrais remercier mes rapporteurs Julie Digne et Éric Paquette pour leur lecture attentive de mon manuscrit et les riches retours qui m'ont permis de profiter de leur large expérience de recherche. Ensuite, mes examinateurs Chris Wojtan et Kiwon Um, ainsi que le président de mon jury Antoine Rousseau pour l'ensemble de leurs questions qui m'ont permis de mettre mon travail en perspective avec leurs connaissances et de partager l'expérience que j'ai acquise durant cette thèse.

J'ai eu la chance d'être encadré par trois superviseurs brillants : Adrien Bousseau, Guillaume Cordonnier et Régis Duvigneau qui m'ont formé en m'introduisant à l'Informatique Graphique, la simulation de Fluides et à la recherche en général. J'ai particulièrement apprécié l'interface qui m'a été offerte avec les autres chercheurs de mon domaine en France et dans le monde. En particulier, je remercie l'équipe de Elmar à Delft et Pierre à Montréal de m'avoir accueilli et d'avoir éveillé ma curiosité scientifique. Je suis aussi reconnaissant d'avoir pu bénéficier de cette formation dans un cadre qui m'a permis de développer ma vie tout en poursuivant aussi les choses importantes pour mon équilibre. Je me sens grandi de mon expérience à leurs côtés. Je voudrais par la même occasion remercier l'ensemble des professeurs que j'ai eus au cours de ma scolarité et qui m'ont transmis une curiosité et une joie d'apprendre qui m'est chère. Cette formation vient aussi compléter la leur, dont j'ai eu la chance de profiter gratuitement, pourvu que cela reste le cas pour les générations à venir.

Je voudrais ensuite remercier l'équipe GraphDeco qui a été un environnement agréable et enrichissant pour étudier. Merci à tous les membres passés et présents : Yorgos, Siddhant, Stavros, Felix, David, Léo, Yannis, Clément, Nazar, Anran, Bernhard, Thomas, Melike,

Ishaan, Henro, Andreas, Alexandre, Aryamaan, Petros, Gilda, Riccardo, Jiayi. Chacun de vous a contribué à cette bonne ambiance qui prospère dans l'équipe en développant la précieuse entraide qui la caractérise. En particulier, je voudrais remercier George et Sophie pour le constant effort que vous mettez à créer les conditions de notre réussite. Je voudrais remercier Nicolás, Giuliana, Juan Sebastian et Marzia pour m'aider à pratiquer l'espagnol et l'italien quand l'omniprésence de l'anglais et l'avalanche de mauvaises nouvelles qui l'accompagne pouvaient devenir oppressantes : 10 minutes avec vous et mon cœur latin était prêt à repartir. Remercions Panagiotis pour la vie qu'il apporte au groupe par sa perpétuelle curiosité. J'ai pu compter sur le soutien de Yohan, Alban, Berend et Capucine qui m'a été précieux tout au long de ma thèse ; j'ai par là trouvé des amis. Je voudrais finalement remercier Jasmine et Mubasharah pour les moments passés ensemble et pour m'avoir fait regretter de quitter Nice.

Je voudrais finalement remercier Armand pour m'accompagner dans mes choix et pour découvrir le monde, Arthur G. et H. pour m'épauler depuis des années, Emilie pour m'aider à avancer et Gauthier pour attiser ma curiosité culturelle. Merci à ma petite sœur pour être une complice depuis toutes ces années, à mes parents pour me soutenir inconditionnellement dans mes choix, et à ma famille pour leur présence et leur soutien constant.

Résumé

L'omniprésence de l'air autour de nous en fait un élément indispensable à prendre en compte pour simuler des phénomènes naturels et concevoir des objets immergés dans ce fluide. En fonction de leur taille et de leur nature, les objets sont des obstacles qui peuvent être transportés, déformés ou ralentis au contact de l'air. Ainsi, le vent érode et modèle les paysages naturels, et les véhicules sont conçus pour lui offrir une moindre résistance.

Étudier ces phénomènes implique de comprendre et de modéliser les interactions vent-obstacle. Cette modélisation représente un défi de par la nature non linéaire des équations qui gouvernent l'écoulement des fluides. Représenter précisément le comportement d'un fluide requiert souvent de faire appel à des solveurs très couteux en temps, ce qui limite grandement leur utilisation dans certains contextes.

Nous explorons dans cette thèse des méthodes permettant de décrire efficacement ces influences vent-obstacle afin de les simuler et d'anticiper leurs impacts dans deux cas d'usage où le besoin de résultats rapides est crucial : Dans un premier temps, nous nous intéressons à la conception de voiture en proposant un outil fournissant des retours aérodynamiques aux designers automobiles sur les formes qu'ils proposent. Afin de permettre de rapides itérations sur le design, ces retours sur le comportement de l'écoulement autour des formes proposées doivent être interactifs. Dans un second temps, nous étudions différentes approches permettant de modéliser des paysages désertiques, permettant à la fois de simuler des dunes et de décrire les motifs créés par érosion/déposition du sable autour de bâtiments. Ici, l'obstacle, à savoir le terrain, évolue constamment à mesure que le sable est érodé puis déposé par le vent. Le vent doit être à son tour mis à jour lors de chacune de ces étapes. Ces itérations nécessitent une méthode appropriée afin de ne pas produire des temps de calcul trop importants.

Nous surmontons ces problèmes en proposant des méthodes cherchant les meilleurs compromis temps de calcul/précision des phénomènes à l'œuvre dans chacun des cas. Nous identifions pour cela les caractéristiques nécessaires de l'écoulement afin de limiter la complexité de nos algorithmes et présentons notamment des méthodes à base d'apprentissage qui nous permettent d'accélérer nos algorithmes.

Dans le cas du design de voiture, nous montrons comment entraîner notre système d'aide sur des observations instantanées et synchronisées, plus riches en informations que des données moyennées. Le modèle neuronal que nous obtenons, associé à une paramétrisation apprise des formes, nous permet en outre d'inverser la formulation du problème et de proposer au designer des formes optimisées. Nous assemblons ces outils et démontrons leur efficacité dans le cas de profils 2D.
Dans le cas de la modélisation de paysages désertiques, nous notons que la saltation est le mode de transport de sable prédominant, ce qui nous permet de simplifier notre algorithme. Couplé à une simulation rapide de vent, nous obtenons une méthode efficace inspirée des sciences naturelles et de l'informatique graphique. Nous validons nos approximations en comparant nos résultats à des mesures effectuées dans le monde réel.
Enfin, dans la perspective d'inverser l'algorithme de dépôt de sable pour le design inverse d'infrastructure, nous décrivons des résultats préliminaires d'accélération de la simulation de l'air en développant une formulation auto-apprenante prédisant un vent moyenné au-dessus d'un terrain. Cette méthode étant basée sur des réseaux de neurones, elle est prometteuse pour le design inverse.

Mots-clés: Simulation de fluide, Modélisation de paysage, Conception de formes aérodynamiques, Problème inverse

---

Abstract

The air being ubiquitous around us, it is a vital element to take into account to simulate natural phenomena, and design object immersed in this fluid. Depending on their size and nature, objects can be transported, deformed or slowed down by contact with air. In this way, wind erodes and shapes natural landscapes, and vehicles are designed to offer them less resistance.

Studying these phenomena involves understanding and modeling wind-obstacle interactions. This is a challenging task, given the non-linear nature of the equations governing fluid flow. Accurately representing fluid behavior often requires the use of time-consuming solvers, which severely limits their use in certain contexts.

In this thesis, we explore methods for efficiently describing these wind-obstacle influences in order to simulate them and anticipate their impacts in two use cases where the need for rapid results is crucial: Firstly, we focus on car design, proposing a tool to provide aerodynamic feedback to car designers on the shapes they propose. To enable rapid iterations on the design, this feedback on the flow behavior around the proposed shapes must be interactive. In a second step, we study different approaches to modeling desert landscapes, both simulating dunes and describing the patterns created by sand erosion/deposition around buildings. Here, the obstacle - the terrain - is constantly changing as sand is eroded and deposited by the wind. The wind must in turn be updated at each of these stages. These iterations require an appropriate method to avoid excessive computation times.

We overcome these problems by proposing methods that seek the best compromise between computation time and accuracy of the phenomena at work in each case. We identify the necessary flow characteristics to limit the complexity of our algorithms, and present learning-based methods to speed up our algorithms.

In the case of car design, we show how to train our aid system on instantaneous, synchronized observations, which are richer in information than averaged data. The neural model we obtain, combined with a learned parameterization of shapes, enables us to invert the problem formulation and propose optimized shapes to the designer. We assemble these

tools and demonstrate their effectiveness in the case of 2D profiles.
In the case of modeling desert landscapes, we note that saltation is the predominant mode of sand transport, which enables us to simplify our algorithm. Coupled with a fast wind simulation, we obtain an efficient method inspired by both natural sciences and computer graphics. We validate our approximations by comparing our results with real-world measurements.
Finally, with a view to inverting the sand deposition algorithm for reverse infrastructure design, we describe preliminary results for accelerating air simulation by developing a self-learning formulation predicting averaged wind over a terrain. As this method is based on neural networks, it shows promise for inverse design.

Keywords: Fluid simulation, Landscape modeling, Aerodynamic shape design, Inverse design

Contents

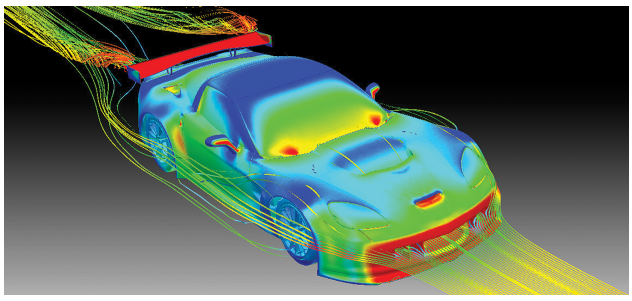
Chapter 1

Introduction

1.1 Context

Fluids significantly shape matter that surrounds us in various aspects. For example, landscapes are naturally sculpted by fluids in canyons, valleys, snowscapes. Also, a considerable number of items of our everyday life were designed to account for the influence of fluids, be it to guide these fluids with canals, dykes, or to convert their energy by building dams and wind turbines. At the heart of these processes lies the incompressible nature of fluids: Objects in contact with a fluid can be seen as *obstacles* preventing that fluid from occupying all the space available. This results in forces applied on these obstacles. In return, obstacles impact the trajectory of the fluid flow. Typically, obstacles like vehicles deflect the air and create vortices in the wake of the flow, as illustrated in Fig. 1.1b.

(a) A car drawn by a designer. Sketches visually represent shapes without precisely defining surfaces.



(b) Streamlines around a car. While the air comes with a uniform speed, it is deflected by the obstacles, thus creating vortices.

Figure 1.1: In the automotive industry, designers often use sketches to convey their ideas (a). Once drawn, these sketches are then modeled in 3D, and physical simulations are run to compute and visualize aerodynamic properties (b). These back-and-forth iterations between designers and engineers who use different tools can hinder the exploratory phase of car design.

Figure 1.2: A star dune in Namibia. Star dunes are intriguing landmarks of desertscapes. They are giant, pyramid-shaped dunes composed of interlaced arms with sharp ridges. They can be from tens to hundreds of meters high, and shape under very specific condition where predominant winds come from several direction, defining the directions of the arms.

This interaction can manifest through various events, depending on the characteristics of the fluid and obstacle in question. In particular, the air is ubiquitous in our environment. Consequently, many objects are influenced and shaped by the air, under various conditions. Studying the wind-obstacle interaction is therefore essential.

At short time scales, engineers can be interested in shaping objects to control or mitigate the impact of the air. To do so, they *design* objects by anticipating the fact that their objects will be immersed in fluids. For example bicycle helmets, airfoils or cars should offer a low resistance to the air as they move into it.

At longer time scales, when other phenomena prevails, the wind can *transport* tiny particles above terrains. When accumulated over long periods, these movements can create specific patterns in snowscapes or desertscapes, as illustrated in Fig. 1.2. Under extreme conditions, these depositions can also damage infrastructure by accumulating on them. We can see several such examples in Fig. 1.3.

In this thesis, we propose to investigate two problems involving obstacles interacting with the air: The design of an aerodynamic vehicles and the modeling of a natural environment shaped by the wind.

1.2 The influence of the wind on obstacles

Several phenomena can arise from the interaction between the wind and obstacles. We focus on two such phenomena: vehicles being slowed down by the air as they move into it, and small particles being transported by the wind blowing above a terrain.

When a vehicle moves on a road, air flows around it. This flow has a non-uniform pressure. Depending on these varying pressures, the air applies efforts on the surface of this obstacle. Once summed, these efforts result in a force, which horizontal component, the *drag*, is opposed to the movement because the pressure is higher in front of the car than in the wake of the flow. However, the magnitude of this force depends on the shape of the obstacle. The system we propose assists designers by providing feedback on the aerodynamic properties of the object they draw. Car sketches and air flow visualization example are illustrated in Fig. 1.1. We focus on 2D profiles and propose shape modifications to ease the designing process.

When snow flakes fall, rocks disintegrate and dust gathers, tiny particles accumulate on terrains. The wind blowing above these terrain can *transport* these particle and, under certain conditions, considerably shape landscapes. On the one hand it creates iconic landmarks of sceneries like the sand dune we can see in Fig. 1.2 or snow cornices. On the other hand, these accumulations can damage human structures. For example, snow can accumulate on roofs and make them collapse, dust can deposit on solar panels and prevent them from receiving sun energy and sand can be blown on railways and block trains in remote areas with intervention is needed or on other structures. We can see various examples of sand damaging infrastructures in Fig. 1.3. By focusing on desertscapes, we propose a fast and reliable methods to model sand transport and produce realistic deposition patterns. We further describe a neural-based approach to accelerate air flow computation over evolving terrains.

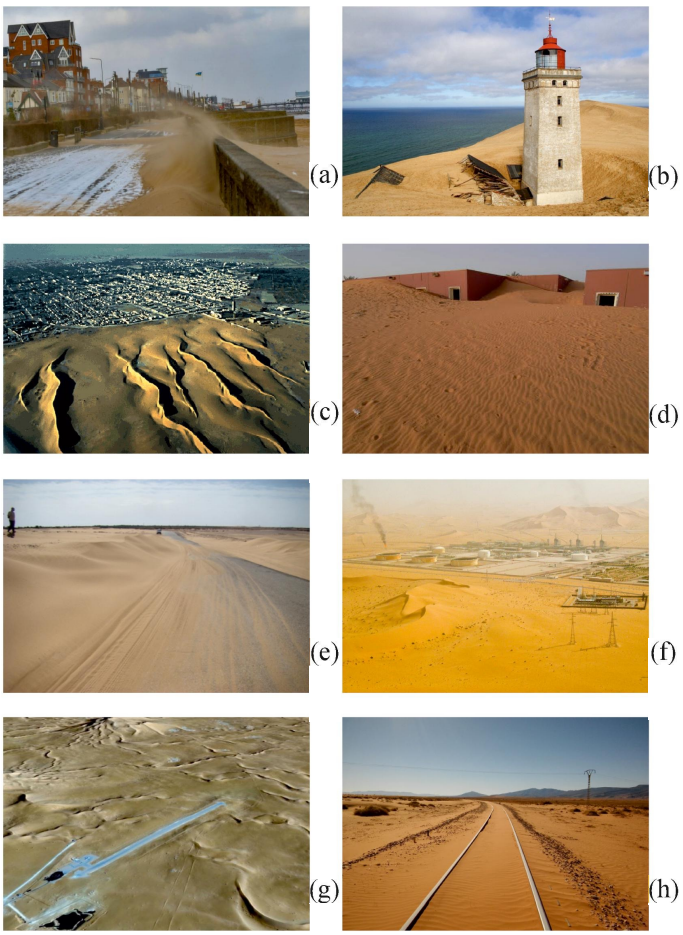


Figure 1.3: Figure from [L. Raffaele, 2019] illustrating various human structures damaged by sand with a) huge piles of sand on a road in Cleethorpes, UK, necessitating important clean-up work, b) a lighthouse damaged by sand in Denmark, c) enormous dunes in an urban environment in Nouakchott, Mauritania, d) houses covered by sand in In-Salah, Algeria. And civil infrastructure affected by windblown sand: e) a road, f) an oil refinery, g) an airport runway h) railways.

1.3 Challenges

Modeling the interactions between obstacles and the air requires solving several challenging tasks.

Air flow modeling: We first need to model the wind behaviour around obstacles. To account for the effects of the air on a surface, be it an object or terrain, one needs to solve for the Navier-Stokes equation that governs the motion of air. Its non linearity makes it notably hard to solve and precise solutions cannot be easily computed as they require using time consuming solvers. Moreover, the chaotic/dynamic nature of fluids makes it challenging to study.

Existing tools to evaluate and optimize obstacles subject to air flow typically rely on expensive simulation over a fine volumetric mesh of the obstacle and its surroundings [OpenCFD, 2007]. While necessary for downstream engineering, such accurate fluid flow

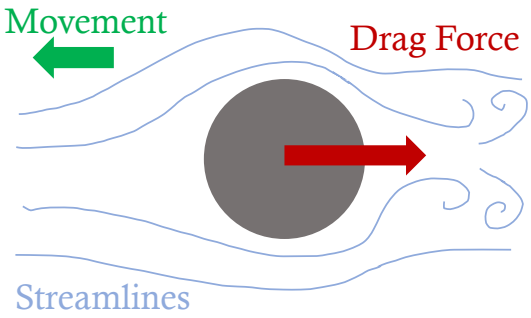


Figure 1.4: The effect of the air on an object: streamlines help visualizing the fluid flow and drag quantifies the force in the opposite direction of the movement that slows down the object.

simulations are too costly for rapid exploration of early design alternatives and rapid landscape modeling. Indeed, our design system is meant for designers to quickly test ideas and find balance between aesthetic and functionality. Thus, it has to be *interactive*. And our sand transport method is meant for creating large scene in a reasonable time. Moreover, we are interested in proposing shape modifications, and therefore *inverting* the fluid flow. Considering the frames of every time step of a simulation makes this task very challenging. While considering time averaged values can be sufficient for sand transport, more instantaneous effects, like the accumulation of vortices in the rear flow behind a car are rich visual feedbacks and necessitate considering snapshots of the fluid flow. Extracting these require a fine description of the fluid flow. Finding these trade-offs between precise but costly computation and requirements for time efficiency is particularly challenging.

Car description: In the case of car design, we need to be able to take as input a profile drawn by the designer, and to obtain a parametrization of it on which we can perform shape optimization. Prior work addresses this challenge by representing the input with parametric curves and surfaces [Umetani and Bickel, 2018; Baque et al., 2018], 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.

Sand transport: Modeling large landscapes of several tens or hundreds of meters by describing the movement of particle of sub-millimeter size is a challenging task. On

the one hand, methods from computer graphics offer high speed and user control, but make simplifying assumptions that are seldom validated against real-world experiments [Taylor and Keyser, 2023]. On the other hand, methods from geo-sciences achieve high accuracy by accounting for multiple physical effects, but these effects are costly to simulate [Lo Giudice and Preziosi, 2020]. Furthermore, while natural sand beds are usually gentle due to the sand repose angle ($\approx 30°$), we want our method to be able to also describe sand erosion and deposition around buildings and human facilities.

1.4 Strategies

We propose to overcome these difficulties by proposing the following methodologies:

Air flow modeling; To design methods that are invertible, we rely on *neural surrogate models*, which are machine learning models that approximate costly simulators [Regenwetter et al., 2022; Li et al., 2022].

Air flow around cars: We propose a model to support designers in multiple tasks, from streamline visualization to shape optimization under various criteria. To do so, we propose a neural-based method that predicts, from a profile drawn by the designer, a velocity field to visualize the flow field and optimize the shape. We alleviate the challenge of outputing an instantaneous velocity field 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. To achieve interactivity, while 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 [Guo et al., 2016; Thuerey et al., 2020; Chen et al., 2021], 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 [Park et al., 2019; Sitzmann et al., 2020], 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.

Wind above terrains: While we first rely on Stable Fluids [Stam, 1999b] to solve for the wind above our terrains, we then draw inspiration from the characteristic *physics-informed losses* of physics informed neural networks (PINNs) [Raissi et al., 2019; Chu et al., 2022] to design a neural-based surrogate to perform this task. The invertible method we obtain is self-supervised to avoid depending on pre-defined terrains.

Car description: We want to allow designers to freely draw profiles. To avoid relying on parametric curves fitting on these profiles, 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 criteria 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.

Sand transport: We draw inspiration from both geo-science and Computer Graphics to propose a simulator of sand erosion and deposition that focuses on *saltation* – the phenomenon of sand particles hoping over the terrain under the action of the wind, which dominates sand deposition patterns [Kok et al., 2012]. We derive our sand transport model from geo-science laws that only act at the surface of the terrain to reduce the computational times. While simpler than advanced simulators from the engineering literature, we qualitatively validate our approach against real-world measurements.

1.5 Contributions

- First, we propose an interactive system for 2D car design than enable the user to get physical feedback on the vehicle he/she is designing based on surrogate models. The algorithm proposes visualizations to better grasp the fluid impact on the car as well as shape optimization proposals.

- Then, we develop a simulator for sand erosion and deposition that can model dunes as well as erosion patterns around obstacles. We couple a fast fluid simulation with

a sand transport algorithm and qualitatively validate our resulting scenes against real-world experiments.

- To further speed-up the sand simulation, we finally present a surrogate model to quickly infer an averaged velocity field above a terrain. Interested in inverting the global simulation pipeline, we rely on a self-supervised neural network.

1.6 Publications

Chapters 3 and 4 are based on the two following peer-reviewed publications:

- *Nicolas Rosset, Guillaume Cordonnier, Régis Duvigneau, Adrien Bousseau, 2023. Interactive design of 2D car profiles with aerodynamic feedback. In* Computer Graphics Forum 42 (2), *pages 427-437*
- *Nicolas Rosset, Régis Duvigneau, Adrien Bousseau, Guillaume Cordonnier ,2024. Wind-blown sand around obstacles-simulation and validation of deposition patterns. In* Proceedings of the ACM on Computer Graphics and Interactive Techniques 1 (7) *pages 1-13*

Chapter 5 presents preliminary results of an ongoing work that has not been published yet.

Chapter 2

Context and Related work

2.1 Physical phenomena induced by wind-obstacle interaction

The modeling of the interactions between fluids and objects is a broad-reaching and interdisciplinary subject, spanning from urban planning [Mittal et al., 2018], to aerodynamic optimization [Martins, 2022], fluid animation in movies and video games [Tan and Yang, 2009] and many more, such that we cannot exhaustively discuss them here. Therefore, we first aim to provide an overview of how two communities have approached the modeling of fluid-obstacle interactions.

On the one hand, scientists in Computational Fluid Dynamics (CFD) mostly work on finding concrete real-life applications and engineering testing. For example, they have been looking into the specific reactions of water with underwater vehicles in order to optimize their shapes [Gao et al., 2016], but also the interactions between the air and various objects like wind turbines to produce energy [Xudong et al., 2009], a plane's airfoils to maximize its performance [Buckley et al., 2010], or the optimal ways to deviate wind to prevent dust from accumulating on solar panels [Raillani et al., 2022]. While often very precise, these methods are quite time consuming, which hinders their usage in many use cases, some of which being the development of industrial designs, movies, video games or the design of vehicles.

Computer Graphics, on the other hand, focuses on the visual synthesis of these interactions. For example, researchers have developed tools to model phenomena like the shaping of landscapes by different types of erosion [Hartley et al., 2024], including fluvial erosion [Galin et al., 2019], [Cordonnier et al., 2016] and the movement of objects on water [Huang et al., 2021], but also to simulate the behavior of smoke around buildings [Yoshida and Nishita, 2000]. Drastically improving over the past decades, research in this field has been able to reach a state where scenes containing fluids and obstacles can be modeled quickly and are visually close to indistinguishable from reality. When compared

side-to-side with actual real-world situations however, these models still struggle to accurately represent the complexity of all influences that can be involved in the shaping and behavior of fluids and objects, and are thus unable to provide representations that are entirely true to reality.
We review the literature to describe the phenomena induced by the wind-obstacle interaction, and study under which form these phenomena can be visible for humans. This will help us create a car design assistant that produces rich aerodynamics feedback, and a sand transport algorithm that can represent the variety of dunes and deposition patterns present in nature.

A crucial point of interest for modeling wind-obstacle interaction is the study of the pressure forces applied by the fluids on objects. Due to the ubiquitous nature of air, its study makes it paramount for modeling natural phenomena induced by wind, and in the practical design of many human constructions and productions. In this thesis, we will especially focus on this specific fluid and will be interested in wind-obstacle interactions. In fact, as illustrated in Fig. 2.1, the varying air pressure around an object necessarily applies forces on it directed by the normal of the contact surface, whether in static situations or for moving objects.
Depending on the size and weight of the object immersed in the air and on the duration of that immersion in it, these forces will result in different phenomena. Larger objects will interact with the air, reshaping the fluid movements and possibly impacting the objects themselves as well: vehicles for example are immediately slowed down by surrounding air as soon as they start moving within it. Tiny, light particles like sand or dust, when blown by the wind during long time spans can erect astounding desertscapes. These two opposed scales of object study imply modeling several parts of the fluid-object interaction with fundamentally different requirements and concerns. In this thesis, we focus on these two modes of interaction and study how researchers tackled these challenges from both points of view : vehicles' movement in air, and particles effected by the wind.

2.1.1 At the vehicle scale

The aforementioned forces, when applied to a moving object immersed in the air, result in a force with a vertical component, the lift, and a horizontal component, the drag. While lift is optimized to make real planes [Liu, 2021] or paperplanes [Umetani et al., 2014] fly, drag is used to model the slowing down of objects. The air flow behind the obstacles is

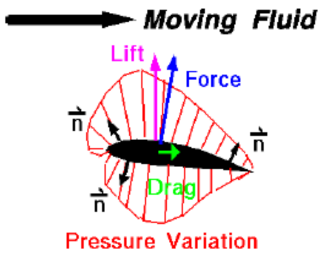


Figure 2.1: Conceptual representation of the pressure force of the air on an airfoil from grc.nasa.gov. The pressure variations apply varying forces on the surface of the airfoil. Once integrated, these forces result in a global horizontal component: the drag, and a vertical component: the lift.

also altered. As we can see in Fig. 1.1b, from a uniform air flow coming with parallel streamlines, vortices are created in the wake of the flow. The wake contains valuable information about the overall flow [Bearman, 1984]. Modeling and representing the air flow around obstacles is therefore important in a design context. Particular characteristics, like vortices close to the rear glass of a vehicle can be an important feature to consider in order to prevent dust from being projected on this glass and visibility alteration [Kim and Han, 2011]. For this reason, we want our system to consider snapshots of the velocities that, contrary to averaged values, contain these vortices.

The air patterns created by an obstacle deviating the wind flow, which we can see in Fig. 2.2 can cause a particular particle deposition distribution around these obstacles. We will detail this process in Sec. 2.1.2.
Since the object shape impacts these forces, it can also be optimized to allow it to move faster and consume less energy. In some cases, researchers seek inspiration from nature to propose such designs [Chen et al., 2018]. We review these approaches in Sec. 2.3.

To better understand these phenomena and test their modeling, scientists have first relied on measurements and have developed real-world experimental set-ups. A common experiment to carry out to study these interactions is *wind tunnel*. When optimizing a car shape for example, air is projected in front of the car to mimic the vehicle movement while driving, and the car resistance is measured [Ansari and Mourya, 2014] to assess how it will be slowed down by the wind. With this set-up, methods for visualizing vortices and studying the fluid behavior in the wake of the flow have been developed, like projecting smoke the visualize vortex creation and obstacle breaking the air [Oda and Hoshino, 1974]. Similarly, to better understand how sand sceneries with infrastructure should look

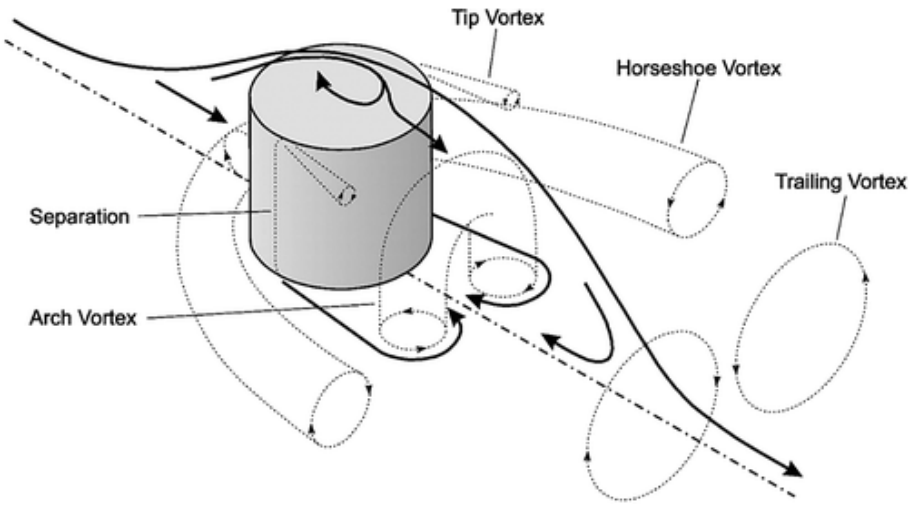


Figure 2.2: Conceptualization of coherent flow structures surrounding a wall-mounted cylinder from Tominaga et al. [2018], redrawn from [Pattenden et al., 2005]. The air flow was divided in different areas, some containing vortices in the wake of the flow. These different areas will produce different erosion/deposition behaviors.

like and how sand deposits around obstacles like human facilities, engineers relied on wind tunnel experiments to closely study the wind flow around obstacles, as we can see in Fig. 2.2, and measure erosion and deposition around these obstacles [Tominaga et al., 2018]. We will make use of these measurements, and qualitatively compare our results to the deposition patterns measurements by [Poppema et al., 2022] around cuboïd facilities on a beach in the Netherlands. However, to better understand and model how sand is blown by the wind, we will first have to look a different scale.

2.1.2 At the particle scale

Looking at these forces from a slightly different perspective, we notice that they can also provoke movement to small and light objects: *particles*. In natural landscapes, complex patterns like snowdrifts, sand dunes and dust piles shape through complex interactions between the particle bed and the air flowing above it, referred to as *aeolian processes* [Nichols, 2009]. While understanding the exact physics behind the driving of particles from a particle bed is still an active area of research [Zhao et al., 2024], scientists have, during the years, come up with models to explain how wind blows particles:
Intuitively, wind erodes the particle bed where its intensity is high, and it deposits the eroded particles where its intensity is low. Several modes of transportation of the latter

can be considered [Bagnold, 1941], including *reptation* and *creep erosion* where heavy particles hope or roll on the ground over a few centimeters, *suspension* where light particles are carried by the wind over kilometers, and *saltation* where medium particles hope on the ground over distances of a few meters. Note however that there is no strict separation between these modes, and that some studies adopt a definition of saltation that includes reptation and creeping [Kok et al., 2012]. Once deposited, particles are also subject to *avalanching*, a process that typically triggers when the overall surface reaches a critical angle.

In our research, we focus on *sand transport* and will model desertscapes. However, the behavior of particles in sandbeds and snowbeds are very similar, with one major difference that snow can change state, and solidify or liquefy. Thus, we also take inspiration from the snow transport literature [G. E. Liston, 1998; Vionnet et al., 2014; Schneiderbauer et al., 2008; Schneiderbauer and Prokop, 2011]. Our method focuses on saltation since it is responsible for the majority of the sand mass transported by wind over short distances [Valance et al., 2015].

This particular wind-obstacle phenomenon, can cause the shaping of snow and sand landscapes, but can also cause hazards when snow, sand or dust are transported under severe weather conditions and interact with human facilities [Middleton et al., 2019; Middleton, 2017; Rooney Jr, 2014].

Given that sand is widespread around the world, as we can see in Fig. 2.3, storytellers will inevitably want to narrate fictions in sandy environments. The movie industry, as well as video games producers therefore needs to be able to models virtual sand scenes [Davies et al., 2024]. To do so, one has to account for the visual richness of sand dunes. As we can see in Fig. 2.4, depending on the sand supply and the wind orientation in an area, different sand dunes will emerge. For example, barchan dunes should emerge when there is few particle supply and a unidirectional wind, and star dunes when the wind comes from several directions. More precisely, researchers have tried to understand the precise principles that make dunes emerge by coupling the representation of the wind with the movement of sand beds [Zhang et al., 2012], [Narteau et al., 2009], [Hugenholtz and Barchyn, 2012].

Sand is also a key indicator or environmental changes, and the drifting sand can provoke hazards, for example around deserts by damaging crops [Wang et al., 2023] or around coasts by eroding them [Gracia et al., 2018] [Prasad and Kumar, 2014] [Williams et al.,

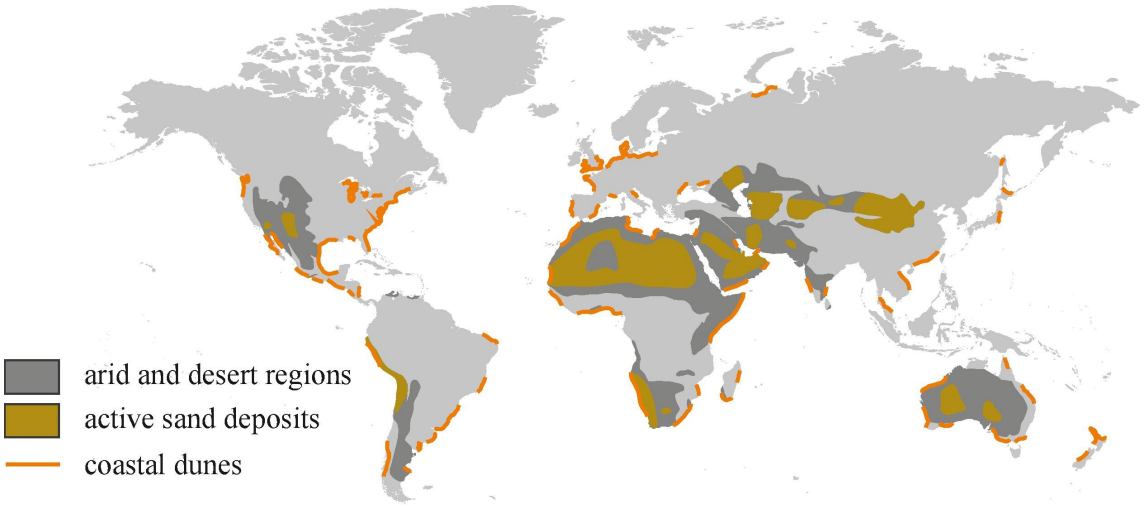


Figure 2.3: Geographical location of the regions potentially covered by sand, divided between coastal dunes, desertic regions and area with active sand deposition, from [Raffaele and Bruno, 2019].

2018]. Moreover, windblown sand can damage civil structures by accumulating on them, like railways [Zhang et al., 2010], houses [L. Raffaele, 2019], and even pyramids [Fahmy et al., 2023]. To anticipate these hazards, scientists wanting for example to forecast meteorological events like avalanches produce models that use real-world measurements through sensors to produce wind fields above the terrain they study [Liston and Sturm, 1998; Bellaire et al., 2017; Luijting et al., 2018; Quéno et al., 2023].
Providing efficient simulation of sand transport is therefore important both to create realistic desertscapes for virtual entertainment and can here help communicate environmental issues in which tiny particles are involved.

The real-world experiments we described are very informative, but are costly and long to set-up. To overcome this problem, we look for *numerical methods* in Sec. 2.2.1 for modeling the fluid flow. Still, we will use several of these measurements to validate our methods.

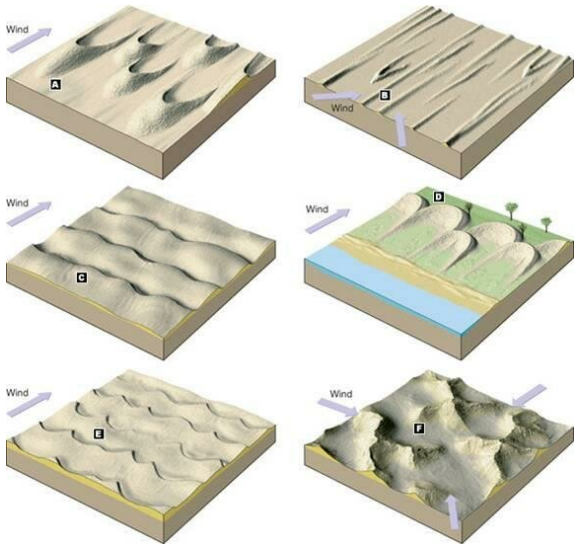


Figure 2.4: Categorization of dunes by [McKee, 1979], redrawn by [Al-Zubaydi, 2017]. They distinguish a) Barchan dunes; b) Longitudinal dunes, c) Transverse dunes, d) Parabolic dunes, e) Reversing dunes, f) Star dunes.

2.2 Simulating fluids and their interactions with obstacles

With the increasing computational power of computers and the various fields of application of fluid dynamics modeling we mentioned in Sec. 2.1, scientists have rapidly developed a number of numerical tools to simulate fluid-obstacles phenomena. We will first review methods to solve for the dynamics of fluids. Then, we will see how these tools can be used to simulate the wind-obstacle interactions we described in Sec. 2.1.

2.2.1 Simulating fluids dynamics

Several possible paths have been proposed and explored over the years to improve the simulation of fluids. Scientists have developed numerical tools to solve for the Navier-Stokes equation [Navier, 1822; Stokes, 1845], governing the motion of fluids.
In engineering contexts, Computational Fluid Dynamics (CFD) methods have been proposed to produce highly accurate results for fluid problems [Harlow et al., 1965; OpenCFD, 2007]. To produce such results, a common way for modeling the fluid is to use an averaged version of the Navier-Stokes equations called Reynolds-Averaged-Navier-Stokes (RANS). We will not detail these methods but rather refer to the following studies to describe how CFD methods were used to simulate wind-obstacle phenomena in urban planning [Tominaga and Stathopoulos, 2013], aerospace vehicle design [Rumsey and Ying, 2002],

or wind turbine design [Baungaard et al., 2022]. Nevertheless, we will use the extensive studies carried out in this field, such as the lid-driven cavity [Kuhlmann and Romanò, 2019], to validate our approximations.
While simulating the air with high precision is necessary for numerous industrial applications, other contexts like movie making, video game developing and artists creation require methods that are fast enough to be used in a realistic business timeline. Since many tools used in engineering do not satisfy this requirement, Computer Graphics researchers have developed methods to model, visualize and animate fluid flow around obstacles. Generally, fluid simulation methods can be divided into three main categories.

The *Eulerian* viewpoint considers a specific, fixed location in space, on which the physical quantities are stored. A common strategy for this is to use a staggered grid, which makes the computation of the incompressibility constraint more accurate. Methods following this viewpoint study the fluid passing through these locations. This approach was used by Stam to develop Stable Fluid Stam [1999b]. This grid-based method is widely used by artists and can produce plausible results, like for smoke simulations [Mullen et al., 2009]. Eulerian methods are usually ill-suited for simulation requiring a non-uniform computation in space, but can produce plausible 3D velocity fields quickly. In particular, numerous researchers relied on this method to model wind-induced natural phenomena. For example, [Hädrich et al., 2021] and [Kokosza et al., 2024] used a Eulerian fluid solver for modeling wild fire propagation, [Overby et al., 2002] for modeling clouds above an urban scene, and [Dobashi et al., 2006] for simulating clouds at the planet scale. Furthermore, the grid structure of Eulerian methods makes them amenable for machine learning acceleration [Tompson et al., 2017].

The *Lagrangian* viewpoint on the other hand, studies the fluid flow by following moving air particles as the fluid flow evolves over time. These quantities are advected with the flow field and tracked by storing their position. [Müller et al., 2003] used this approach to bring Smooth Particles Hydrodynamics (SPH) to fluid simulation. When combined with a GPU implementation [Yan et al., 2009], such methods can reach a high level of realism quickly. We refer to [Koschier et al., 2022] for a more complete description of the recent enhancements and challenges of this approach.
However, Lagrangian methods usually struggle producing divergence-free velocity fields, even if works show promising result in solving this limitation [Bender and Koschier, 2016].

Finally, *hybrid* methods were also proposed, taking advantage of both the Eulerian and the Lagrangian viewpoints. Physical quantities have to be transported between grid-based and air particle-based representations. For example, the fluid implicit-particle method (FLIP) was used to simulate water splashes, and flow around obstacles like cylinders [Ferstl et al., 2016].

Because these simulations do not allow to develop interactive tools, researchers have developed wind models that are even more approximated by computing *procedural winds*. By producing wind fields very quickly, scientists have been able to generate movement in a scene. This approach has been used, for example, to model wind-induced movements on grass in prairies [Perbet and Cani, 2001], but also skyscapes [Vimont et al., 2020], cloudscapes [Webanck et al., 2018], gases in the air [Ebert, 1995], and for hurricanes and tornadoes [Amador Herrera et al., 2024].

We now review how these methods were used in the literature to simulate wind-obstacle interaction in our two use cases. As we will see, while studying the aerodynamic property of a car can be done simply by describing a static object, like for wind tunnel experiments, the simulation of lighter particles like sand, requires *coupling* the air simulation to a moving sand volume. For this reason, the following section contains a more detailed review of the literature regarding sand simulation specifically.

2.2.2 Simulating wind-obstacle phenomena

Since the air flow around an obstacle is responsible for the *aerodynamic properties* of this obstacle, this flow contains crucial information about these properties. Therefore, researchers use simulation tools to study shapes in aeronautics [Olivier et al., 2003] or to design boats [He et al., 2019]. In particular, [Umetani and Bickel, 2018] proposed a method to visualize flow fields around 3D cars and optimize these cars. To communicate the aerodynamics of the shape to the user, they use streamlines and pressure on the surface, as we can see in Fig. 2.5. These visualizations rely on an underlying *velocity field*, a field of 3D vectors describing the direction of the flow. Methods to obtain such a field within a domain were described in Sec. 2.2.1.

In the automotive industry, when an advanced design of a car is proposed, modeling it in 3D and running a costly finite-element method simulation can help refining the shape and imply modifications. However, the aesthetic aspect of a car also has to be taken

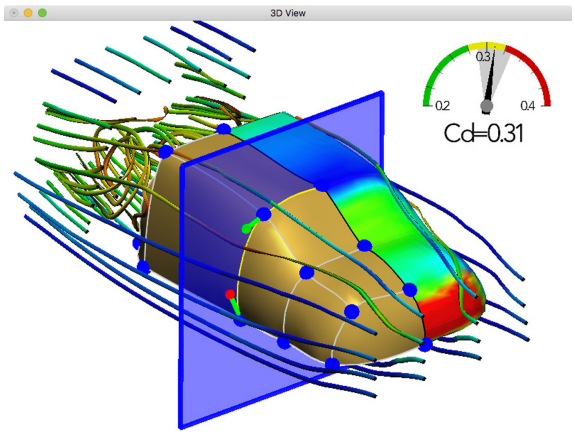


Figure 2.5: Visualization of the air flow around a car in [Umetani and Bickel, 2018]. The visualization of the fluid flow is done using streamline, parallel to the air velocity field, and pressures on the surface. Together, they give rich information on the behavior of the obstacle when moving in the air.

into account in the design process and is difficult to encode as a mathematical objective [Othmer, 2014]. Therefore, *iterations* have to be made between designers and engineers. Using time consuming methods for giving these feedbacks hinders the early explorative phases of design, when aesthetic is decided upon.

To overcome this problem, methods rely on a coarsely simulated velocity field that they upsample. Indeed, the necessity to visualize more precise details led researchers to develop super-resolution techniques which, from coarse input velocity field, produce a higher resolution output. Works in this direction have been carried out for liquids [Roy et al., 2021], smoke [Bai et al., 2020], [Xie et al., 2018] and fluids in general [Bai et al., 2021b].

Because it is able to quickly produce dynamic feature like vortices in a reasonable amount of time, we rely on an implementation of Stable Fluids Stam [1999b] to visualize the fluid flow around vehicles.

Simulating sand

The movement of sand particle can produce richly diverse patterns, as we can see in Fig. 2.4. The enormous amount of particles involved in this process has led researchers to simulate this wind-particle interaction with different viewpoints.

Simulating sand particles: Researchers carried out studies [Finn et al., 2016] to precisely

quantify the movement of particle induced by the wind. They took into account particle characteristics like their non spherical shape [Dun et al., 2018], [Wang et al., 2014] and tried, through real world experiments, describing the exact movement of particles [Zou et al., 2007]. These measurements helped describe average behaviors that first enabled to describe sand bed surface, with their irregularities [Zampiron et al., 2024], and typical ripples [Huo et al., 2021] [Yu and Baranoski, 2024]. These precise measurements also helped developing empirical laws for saltation [Strypsteen et al., 2021] enabling us to consider broader scene.

Simulating sand as a fluid: In Computer graphics, researchers noticed that when granular materials like sand are moving, like when we pour them with our hands or when a sand castle falls down, sand particles have similar characteristics as ordinary Newtonian fluids, but in contrast, granular material dissipates energy quickly. This approximation is particularly valid for wet sand. Researchers have leveraged this fluid-like behavior to animate fluid using the Material Point Method (MPM) hybrid method to simulate sand [Tampubolon et al., 2017] and snow [Stomakhin et al., 2013]. SPH was also used by [Zhu and Bridson, 2005] and [Bender et al., 2012]. Hybrid methods are indeed well-suited for this problem as describing a sand volume with a grid can lead to major approximation. The sand particle are here abstracted and sand is seen as a continuum, and frictional constraints are added to model particle cohesion and thus differentiate sand from other liquids, as we can see in Fig. 2.6. While producing astonishing visual details for sand animation, these method are challenging to scale to landscape modeling and are not necessarily validated against real-world measurements. We therefore look for larger scale methods.

Coupling wind simulation with sand bed evolution: Researchers have used averaged rules to model saltation [Strypsteen et al., 2021] and were able to reach large-scale sand simulation. CFD tool have been used in this sense to model the wind and study precisely particle erosion and deposition. Simulating the wind above terrain is indeed important for forecasting purposes [G. E. Liston, 1998] [Gisnås, 2020], [Joubert et al., 2012]. Closer to our problem, [Lo Giudice and Preziosi, 2020] proposed a pipeline to couple a precise 3D CFD simulation of the wind with an erosion/deposition algorithm to model sand scene. They obtain precise result but at the cost of long computation, up to several days. Interestingly, studies have been conducted to simulate precise air flow around obstacles to reproduce snow deposition patterns [Zhou and Zhang, 2023] as well as sand deposition

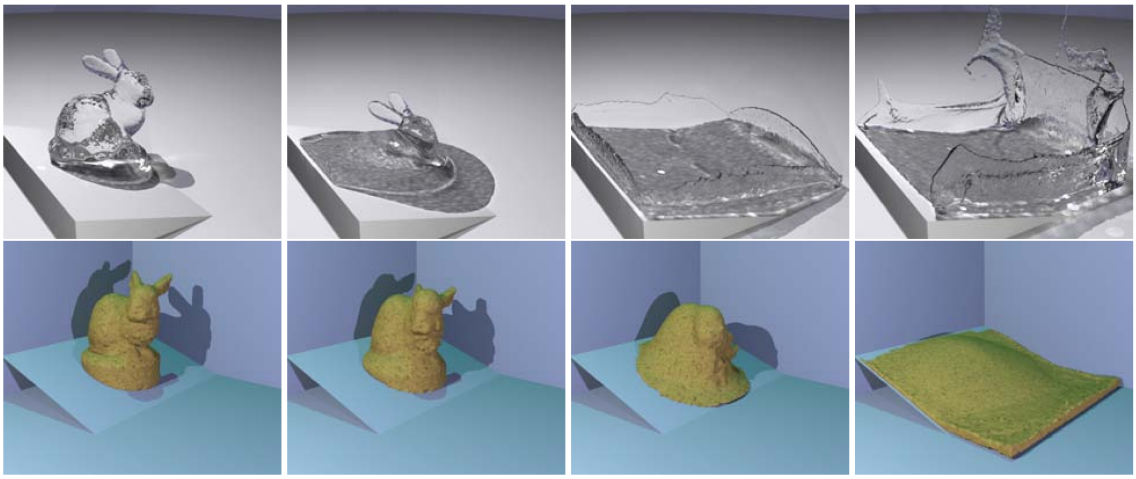


Figure 2.6: Figure from [Zhu and Bridson, 2005]. Sand is here simulated like a fluid with additional friction forces. These friction forces prevents the simulated sand (bottom) to behave like a fluid (top) by splashing on a surface. This is due to an increased dissipation on energy for sand compared to water.

patterns:[Tominaga et al., 2018]. However, these methods, while crucial for real-world forecast and decision-making applications, take up to days to simulate. They do not match our requirements as we are interested in generating sand landscapes quickly for them to be usable in the video game or movie industries. Therefore, we look for faster though realistic methods.
Using a procedural wind, [Paris et al., 2019] produce very convincing result and let varied sand dunes emerge. They achieve this by combining this wind to stochastic sand transportation rules. Recently, [Taylor and Keyser, 2023] further improved their method to produce a real-time algorithm to model desertscapes. Surprisingly, with a very coarse wind description, they obtained an interactive tool able to model large scenes. However, as we will describe in Sec. 4, the deposition patterns they obtained around steep obstacles like buildings are still not faithful to reality. We take inspiration from this work by considering that a coarse description of the wind is enough. Indeed, due to the long time-spans of sand deposition, typical patterns are more caused by averaged behavior of the wind than by specific small-scale wind patterns.

To summarize, precise particle size simulation helped modeling sand transport rules that is, in these cases, modeled as a *horizontal mass flux*, depending on the sand supply and the wind intensity and direction. As an exception, [Lo Giudice and Preziosi, 2020] considers

a fully coupled 3D transport of both the wind and the sand. While we take inspiration from this coupled model by simulating 3D fluid dynamics, we note that saltation and avalanches occur at the surface of the terrain, making these phenomena amenable to a 2.5D representation that tracks the evolution of the sand thickness h over the terrain, and transport sand height on a height map by computing horizontal mass fluxes.

2.3 Shape optimization

2.3.1 General formulation

In the process of designing an object, we saw that numerical simulations can help produce informative quantitative and visual feedback for the designer. Shape modifications can then be made based on these feedbacks. To automate this procedure, researchers have developed shape optimization methods that are used for various objects such as arch dams [Wassermann, 1983], seats [Brienza et al., 1996], furniture [Ma et al., 2021].

Formally, shape optimization problems can be written as follow:

$$\begin{cases} S^\star = \underset{S}{\operatorname{argmin}}\, \alpha(S) \\ \alpha_{pen,i}(S) \geq c_i, i = 1, ..., m \\ \alpha_{pen,j}(S) = d_j, j = 1, ..., n \end{cases} \tag{2.1}$$

where $\alpha(S)$ is an *objective function* to optimize, and $\alpha_{pen,i}, \alpha_{pen,j}$ are constraint function to satisfy. For example, a concrete case would be that a seat should fit in a given environment, while being able to support a given weight and being able to stay static.
We refer the reader to [Haslinger and Mäkinen, 2003] for a global description of shape optimization formalism and approaches, and will focus on aerodynamic shape optimization. Formally, this means that we will be interested in optimizing quantities $\alpha(S)$ that are linked with the forces that the air applies on shapes.

2.3.2 Aerodynamic specification

In the context aerodynamic shape optimization, as we described in Sec. 2.1, quantities of interest are typically linked with the lift or drag forces, that are the vertical and horizontal component or the forces of the air on the object of interest.
For example, engineers are interested in diminishing the fuel consumption and running

costs of airplanes by optimizing their airfoils [Jameson, 2003], [Reuther et al., 1996], or reducing the noise made by these airfoils [Marsden et al., 2001]. [Jang et al., 2016] optimize the shape of radar to avoid for this radar to be detected. Closer to our work, researchers proposed methods for drag reduction and therefore fuel consumption reduction of trucks and cars [Katz, 2006]. Shape optimization of racing cars [Granados-Ortiz et al., 2023] is an interesting case because one has to optimize for a minimal drag, while keeping the lift low so that the car stays in contact with the road. Additionally, this optimization has to follow the International Federation constraints on the size and weight of the car.
We refer to [Martins, 2022] for a description of the challenges of aerodynamic shape optimization and to [Skinner and Zare-Behtash, 2018] for state of the art review in this field.

In our case, we successively focus on drag and vortex attenuation behind the car for the objective quantities $\alpha(S)$, and a volume preserving penalty $\alpha_{pen}(S)$ to enforce the optimized shape to have a similar volume than the one drawn by the designer. We describe in detail the computation of these quantities in Sec. 3. We now review the optimization methods we can draw inspiration from to solve for this shape optimization task.

2.3.3 Optimization methods

In the recent development of optimization techniques, two main categories can be distinguished: gradient-based and gradient-free strategies.

Gradient-based strategies rely on the computation of the gradient of the objective function with respect to the shape $\nabla_S \alpha$. Using this method therefore requires being able to compute this quantity. When this is the case, gradient-based strategies like Adjoint Method [Pironneau, 1974] have proved to be very efficient. Moreover, such methods can handle large dimension parametrizations, but can be trapped in local minima.

Gradient-free methods, in contrast, enable finding minima in cases where the derivation is impossible. For example, genetic algorithms [Mirjalili and Mirjalili, 2019] use a bio-inspired selection of candidate parameters and are better at finding global minima. Some works [Castellani and Franceschini, 2004] have used this method in the context of car design. Main drawbacks of such strategies are that high dimension parameterization hinders their usage, and that they are usually slower that gradient-based methods.

Driven by the requirement of proposing interactive shape modification for our car design

assisting tool, we chose to rely on a fast gradient-based strategy. However, to be able to compute $\nabla_S \alpha$, we need to describe how to parametrize the shape.

2.3.4 Shape parametrization

Parametrizing shapes is a challenging task because shapes might have different resolutions or structures. In 3D, prior work used parametrization like B-splines [Martin et al., 2014], Computer-Aided-Design (CAD) [Fudge et al., 2005], deformed polycubes [Umetani and Bickel, 2018] to solve for aerodynamic shape optimization tasks. We focus on 2D shape parametrization as we want to propose a car design assistant in 2D. The challenges are similar, and researchers typically use parametric curves like Bézier curves [Fakhari and Mrad, 2024] 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 [Pu et al., 2016] to learn a latent descriptor of each profile in our dataset. This provides low dimensional latent codes that we can use to perform optimization onto and get enhanced shapes.

We now review the literature to find relevant ways of computing, from this shape descriptor, the aerodynamics coefficients.

2.3.5 Neural networks for fluid simulation

Computing the objective functions requires simulating fluid problems. As described in Sec. 2.2.1, simulating fluids dynamics is very costly. Indeed, typical Computational Fluid Dynamic methods [OpenCFD, 2007], that rely on meshes to describe the shapes, can take up to days to run. However, the recent rise of learning-based approaches and the need for invertible pipelines has paved the way to other methods.
Researchers have recently developed neural-based methods to accelerate fluid solvers Kim et al. [2019]; Wiewel et al. [2019]; Tompson et al. [2017]; Bai et al. [2021a]. Notably, a body of work adapted neural architectures to meshes that is a typical structure for shapes [Pfaff et al., 2020; Janny et al., 2023; Fortunato et al., 2022]. Recent works also described how to train neural networks with so-called *physics-informed losses* Raissi et al. [2019]; Chu et al. [2022], with the potential of replacing traditional solvers and alleviating the need for large simulation datasets Wandel et al. [2021]. Since sand transport is a

very slow process, we leverage physics-informed losses to propose a mapping between a height map that describes a sand bed, and the averaged wind field flowing above it. While very approximate, this enables us to get a wind field very quickly, which paves the way for developing a fast and invertible sand deposition pipeline.

Other methods used supervised machine learning to produce fast and invertible models to map a shape to fluid-related quantities. For example, to predict pressure over the surface of an object, [Durasov et al., 2021] employed graph neural networks, and [Baque et al., 2018] introduced a geodesic convolutional neural network. They do this by predicting averaged values of a simulation they computed beforehand to obtain training data. 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. However, they are part of the *surrogate models*. These were developed to propose fast and invertible solutions, though approximate.

Many such approaches rely on convolutional neural networks (CNNs) to predict fluid quantities over the entire spatial domain surrounding the object of interest [Guo et al., 2016; Thuerey et al., 2020; Chen et al., 2021]. They usually focus on 2D cases and describe their shape as either bitmap images or distance field images. Other methods leverage the efficiency of point-cloud approaches to predict fluid states [Kashefi et al., 2021]. However, these methods usually rely on averaged values for the velocity field they predict. In order to create databases between input shapes and output fields, they rely on pre-computed simulations. We take inspiration from these methods and create our own database of synchronized fluid flow snapshots to train our neural surrogate model to map cars to instantaneous snapshots. This finally enables us to get a fast invertible pipeline to suggest shape modification to the designer's drawing.

Chapter 3

Interactive design of 2D car profiles with aerodynamic feedback

As discussed in the introduction, the interaction between an object and the fluid it is immersed into can lead to specific behaviors. Designing objects requires taking these behaviors into account to propose relevant shapes to take advantage of some behaviors, and reduce undesired ones. For example, an object moving in the air is slowed by a force called drag. Producing such an object often requires minimizing this phenomenon.

While designing shapes like airfoils or wind turbines can be characterized in a strictly mathematical framework, designing cars requires a delicate balance between aesthetic and performance. To carry out this task, in the automotive industry, designers first convey their proposals for new shapes as sketches. These proposals are then refined after iterating with engineers: the latter mesh the shape and run CFD simulations on them. After deriving aerodynamic properties from these simulations, they provide feedback to the designers. These iterations, because they are long, costly and call upon different people with different expertises, can hinder the early explorative phases of design, when aesthetic is decided upon.

To tackle these challenges, we present in this chapter a system to assist designers in

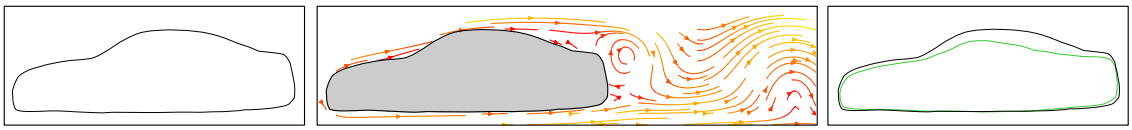


Figure 3.1: Our system takes as input the profile of a car (a) and predicts the flow field around the car (b). We perform shape optimization in a latent space of cars to suggest how to improve the aerodynamic properties of the profile (c, here by reducing drag by 11%). Both the fluid flow visualization 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.

creating aerodynamic car profiles. We investigate how to give rich feedbacks to designers while satisfying the key constraint of doing so in an interactive way to ease the artistic workflow. We showcase the ability of our system by taking into consideration 2D car profiles.

To meet the crucial interactivity constraint, we rely on a neural surrogate model to predict fluid flow around car shapes. While prior work focuses 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, like vortices. Furthermore, we architecture our model to support gradient-based shape optimization within a learned latent space of car profiles and are able to provide new shape proposals to the designer. 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. We can see in Fig. 4 an input sketch produced by the designer, the visualization of the air flow around it and the proposed shape correction proposed by our system.

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 a shape optimization framework 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.

This chapter is mainly based on the following publication:

Nicolas Rosset, Guillaume Cordonnier, Régis Duvigneau, Adrien Bousseau, 2023. Interactive design of 2D car profiles with aerodynamic feedback. In Computer Graphics Forum 42 (2), *pages 427-437*

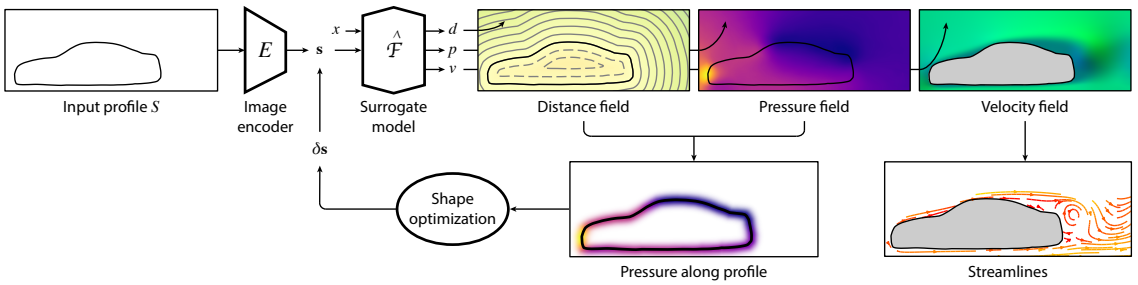


Figure 3.2: **Overview.** We take as input the profile of a car represented as a bitmap S. We train an encoder E to compute a low-dimensional latent descriptor of this profile $\mathbf{s}$, which we feed to our surrogate model along with the spatial coordinates x of the point at which we want to predict 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.

3.1 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 [Zhu et al., 2011], 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:

$$\alpha_{\text{drag}}(S) = \frac{1}{T}\int_0^T \oint_\Gamma -p(S,x,t)n_1(S,x)\,\mathrm{d}\Gamma\,\mathrm{d}t, \tag{3.1}$$

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 [Himeno and Fujitani, 1993], 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. 3.1 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:

$$\alpha_{\text{vortex}}(S) = \frac{1}{T}\int_0^T \int_{\bar{\Omega}} \|\nabla \times \vec{v}(S,x,t)\|\,\mathrm{d}\bar{\Omega}\,\mathrm{d}t, \tag{3.2}$$

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 [Umetani and Bickel, 2018; Chen et al., 2021]. We overcome this limitation by considering instantaneous synchronized observation times, which allows us to predict dynamical phenomena like vortices without significant increase in complexity. We detail in Sec. 3.3.2 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 $\alpha(S)$:

$$S^\star = \operatorname*{argmin}_S \alpha(S). \tag{3.3}$$

Minimizing Eq. 3.3 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.

Fig. 3.2 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 (Sec. 3.2). We then explain how we model and train the surrogate $\hat{\mathcal{F}}$ to approximate flow fields around car profiles (Sec. 3.3).

3.2 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

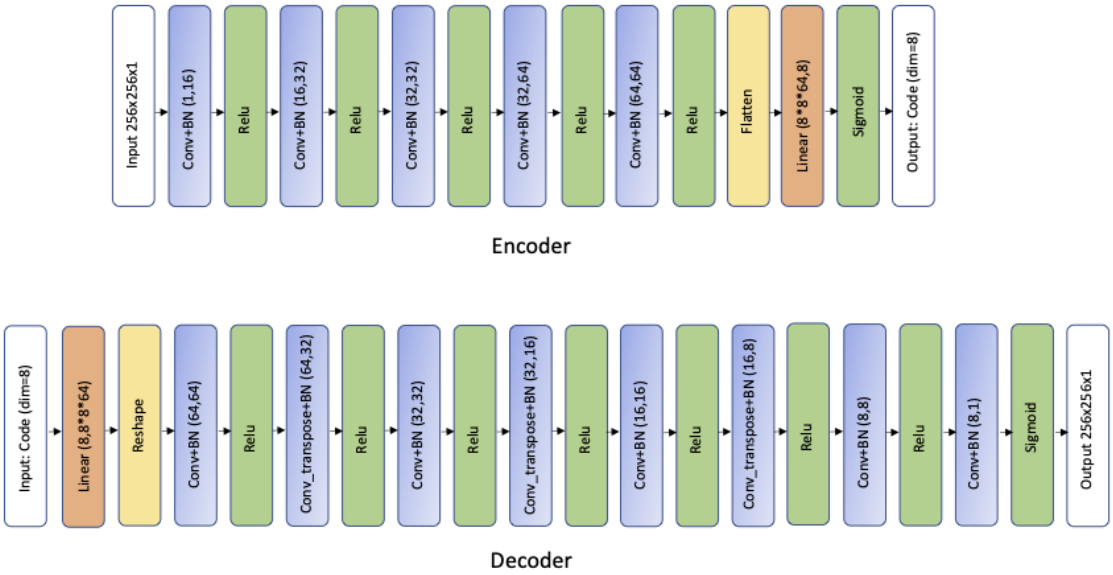


Figure 3.3: Architecture of the autoencoder of car profiles.

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. The exact architecture of the auto-encoder we use is shown if Fig. 3.3.

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

$$\mathbf{s}^\star = \underset{\mathbf{s}}{\operatorname{argmin}}\, \alpha(\mathbf{s}). \tag{3.4}$$

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 (Fig. 3.4).

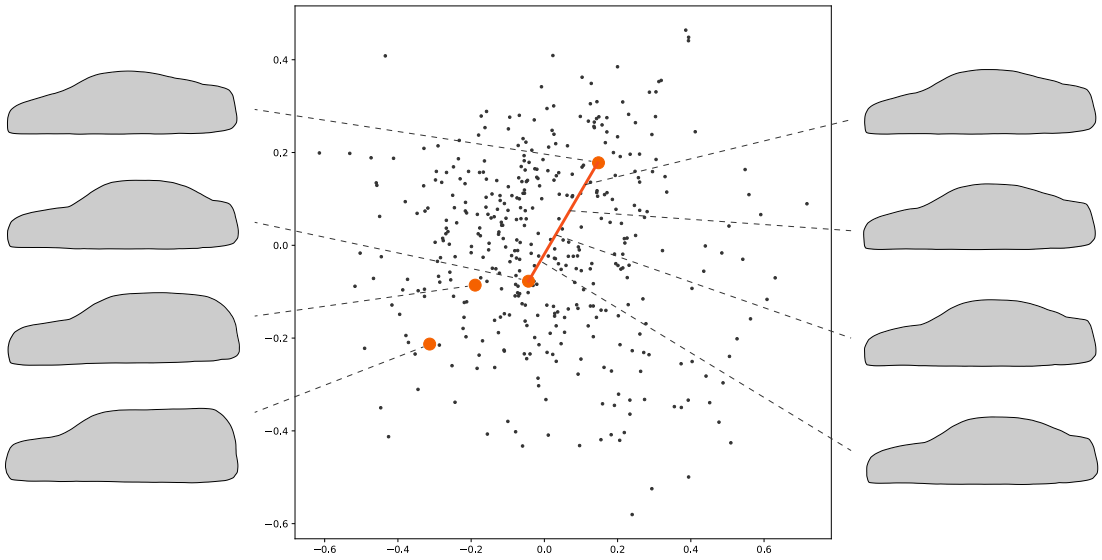


Figure 3.4: **Visualization of our learned latent space.** This latent space captures the distribution of car profiles in our training set (left). Walking along this space produces a smooth interpolation between car shapes (right).

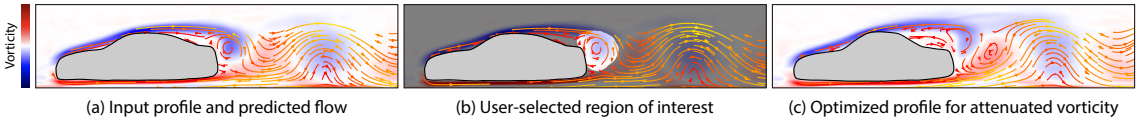


Figure 3.5: **Interactive workflow.** Our system predicts the presence of vortices right behind this input car profile (a). Users can indicate a region where vortices should be attenuated (b). After shape optimization, vorticity is reduced by 30% in the region of interest (c).

Drag computation over an implicit shape representation. Solving Eq. 3.4 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. 3.1 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 Sec. 3.3). Thanks to this implicit representation, we can follow [Chen et al.,

2021] and rewrite Eq. 3.1 as

$$\alpha_{\text{drag}}(S) = \oint_{\Gamma} -p(S,x)n_1(S,x)\,\mathrm{d}\Gamma \approx \int_{\Omega} -p(S,x)n_1(S,x)\delta(d(S,x))\,\mathrm{d}\Omega \tag{3.5}$$

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 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. 3.5 at any point of the domain. We provide in Fig. 3.11 an evaluation of this formulation, which achieves an error of 1.7% on average compared to the exact linear integral computed with Eq. 3.1.

Volume conserving optimization As illustrated in Fig. 3.17, a limitation of optimizing only one target value is that the optimized latent code can end up far from the ones in the dataset, and thus correspond to a shape which does not describe a vehicle in our case. For this reason, we propose a regularisation on the volume of profiles, both to reinforce the optimization process by making it more robust to the input shape and to allow the system to propose more relevant shapes. This feature will facilitate the use of our toolbox for any scientist desiring to build its own surrogate model with *its own dataset* matching its problem.

While one solution could be to the restrict the optimization by adding an additional loss which penalizes the code from travelling far away from the original one, such as KNN loss, we propose a more promising one: A KNN loss would penalize every distant latent code the same way, while a *loss on volume* would consider more distant shapes as well and would therefore be more flexible as it would broaden the explorations, while keeping it close to the dataset.

Formally, we add the weighted term as follow:

$$\alpha_{\text{total}}(S) = \alpha_{\text{drag}}(S) + \beta * \alpha_{\text{volume}}(S) \tag{3.6}$$

Since our surrogate model outputs an estimate of the SDF of the shape, we can subsequently compute an estimate of the volume of any shape S. It is given by the Cumulative Distribution Formula, still with $\sigma = 1.2dx$:

$$\mathcal{A}(S) = \oint_{\Omega} (1 - \frac{1}{2}(1 + erf(\frac{d(S,x)}{\sigma\sqrt{2}}))) \, \mathrm{d}\Omega \tag{3.7}$$

We ultimately obtain our additional loss term:

$$\alpha_{\text{volume}}(S) = \|\mathcal{A}(S_{init}) - \mathcal{A}(S)\| \tag{3.8}$$

Interactive optimization. Equipped with the formulation of α_{drag} in Eq. 3.5, we can solve Eqs. 3.4 and 3.8 using gradient descent with line-search to obtain progressive updates of the shape in latent space, denoted as δs. 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. Fig. 3.5 illustrates shapes designed and optimized within our interactive system.

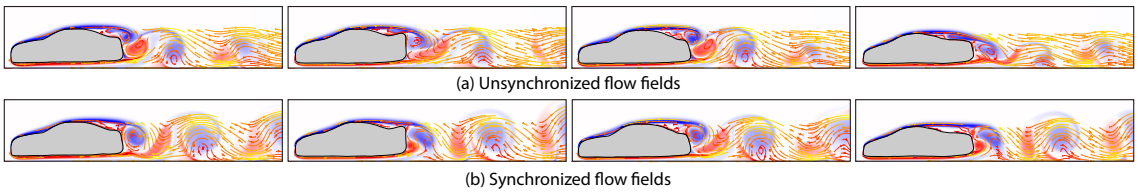


Figure 3.6: For a given time frame, vortices appear at different locations behind the car for simulations with different car shapes (a). By analysing the periodic behavior of the flow field, we synchronize the simulations to extract a representative frame where vortices appear in similar locations (b). These synchronized flow fields are easier to regress by a neural network.

3.3 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 3.6). 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[Muyl et al., 2004]. 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.

3.3.1 Car profiles extraction and encoding

We created our dataset by extracting the central vertical cross-section of 3D cars from a subset of ShapeNet [Chang et al., 2015]. 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. 3.7). 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.

3.3.2 Fluid flow simulation and synchronization

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

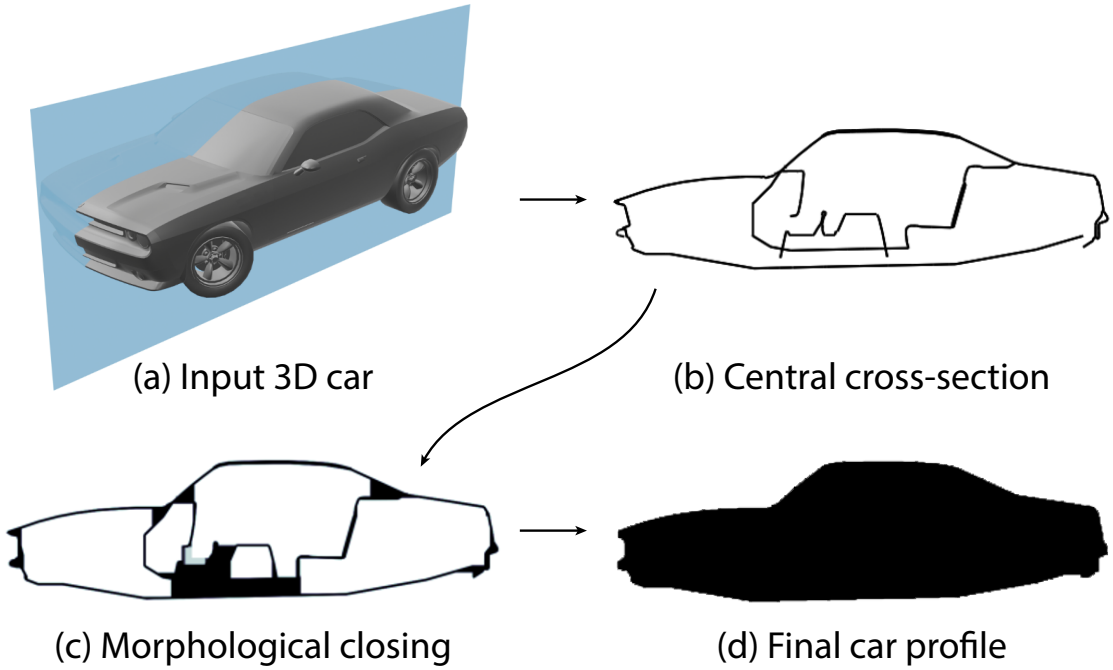


Figure 3.7: We extract car profiles from the central vertical section of 3D models from ShapeNet (a,b). We apply morphological closing (c) and region filling (d) to obtain a binary image of the car outer boundary.

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 $\rho = 1\,\mathrm{kg/m}^3$. 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 $12\mathrm{m} \times 5.12\mathrm{m}$ 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

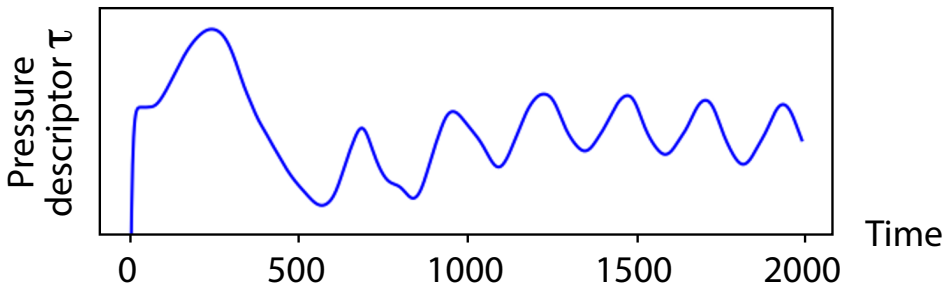


Figure 3.8: After an initial transitory period, the average pressure behind the car exhibits periodic oscillations. We detect the period of this signal to synchronize simulations across car profiles.

speed of the car, $v(0) = 54\mathrm{km/h}$. 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 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 $2\mathrm{cm}$ and the time step $1/750\mathrm{s}$. We use a solver based on the Stable-Fluids Helmholtz's decomposition method [Stam, 1999a], with bilinear interpolation and 3$^{\mathrm{rd}}$-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. Fig. 3.8 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.* [Vlachos Michail, 2005] 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:

$$ACF_k(T) = \frac{1}{N} \sum_{i \in [k-N,k]} \tau_i \tau_{i-T}. \tag{3.9}$$

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.

3.3.3 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* [Park et al., 2019], 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 coordinate dimensions using positional encoding [Mildenhall et al., 2020; Tancik et al., 2020] 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* [Elfwing et al., 2017] 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 [Liu et al., 2022] 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.

Fig. 3.9 details the architecture of the MLP we use to implement our method, and Fig. 3.10 the architecture of the CNN we used for comparison.

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. 3.5) requires values farther from the surface, both inside and outside the shape. Yet, these values evaluated at a small distance from the surface should

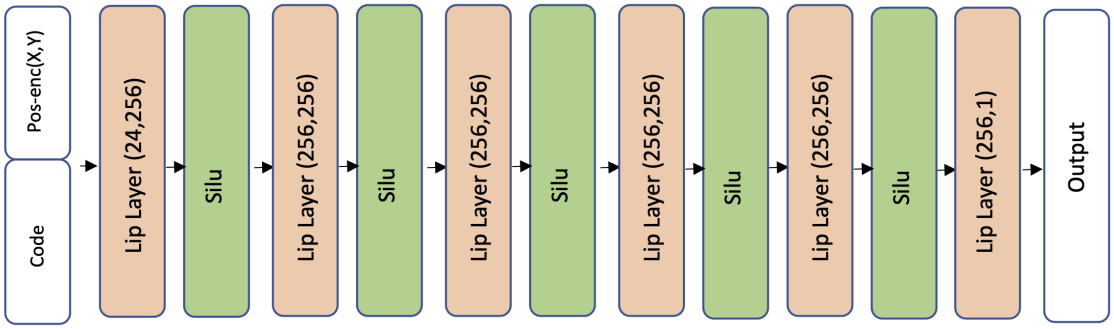


Figure 3.9: Architecture of the multi-layer perceptron used to implement our surrogate model, with *SiLU* [Elfwing et al., 2017] activation functions and Lipschitz regularization [Liu et al., 2022] layers. Some variations appear between the different flavors of our networks: the MLP that predict the SDF has one layer less, and the one that predicts the pressure uses *ReLU* activation functions instead.

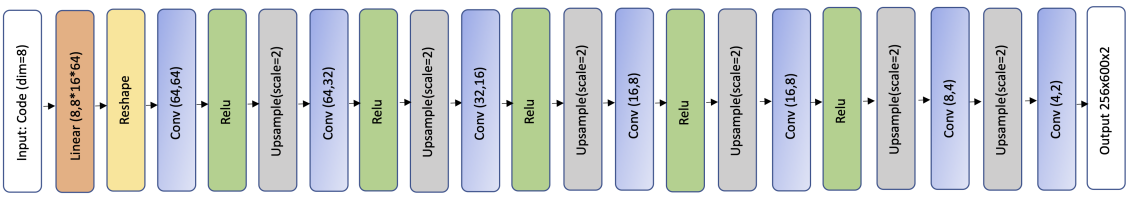


Figure 3.10: Architecture of the CNN we used as a baseline for comparison, which we trained to predict pressure and signed distance fields necessary for drag computation.

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. We validate these intuitions carrying out the following experiment:

We considered the two strategies mentioned above to create a pressure field where points in the vicinity of the profile have similar pressure values as their closest point along the profile. The *projection* strategy consists in replacing the pressure anywhere in the domain by the pressure of the closest point on the shape. In contrast, the *diffusion* strategy produces a smoother field by solving a Laplace equation with pressure values along the profile set as Dirichlet boundary conditions.

We evaluate these two strategies by using the two resulting pressure fields to compute

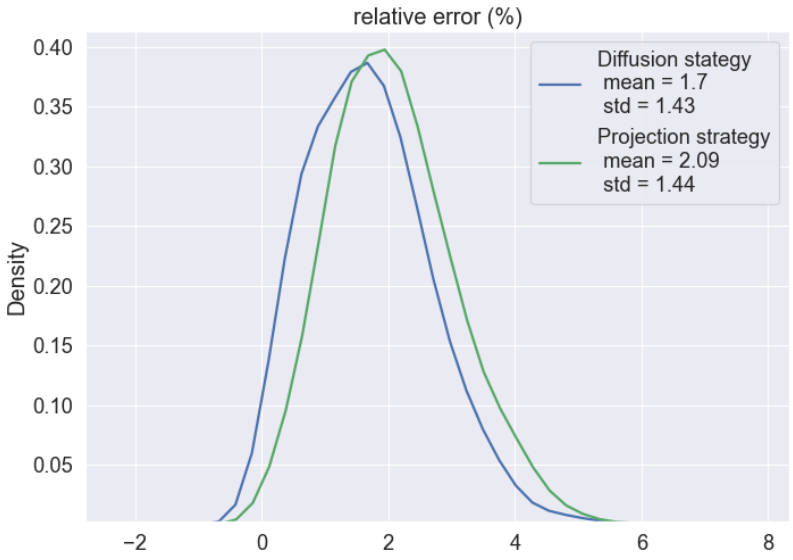


Figure 3.11: Distributions of errors for the two propagation strategies we considered to compute the drag coefficient using our smoothed dirac formulation. Both strategies produce an accurate estimation of the drag compared to the exact integration along the car profile (1.7% of error for diffusion, 2% for projection, on average).

the drag coefficient for each profile in our dataset. We compared these approximate measures obtained with a smoothed dirac to the exact measure computed by a linear integral along the profile. Fig. 3.11 plots the histograms of the relative error achieved by each strategy. This evaluation reveals that the diffusion strategy gives slightly more accurate evaluations of the drag.

Moreover, the diffusion strategy produces a smooth field that is easier to regress than the discontinuous field produced by the projection strategy. We validate this intuition by a second experiment, where we trained two versions of our surrogate model, one for each strategy of propagation. For each strategy, we then compared the drag computed from the predicted field with the drag computed from the corresponding ground truth field, on our test dataset. Fig. 3.12 plots the histograms of the relative prediction error achieved by each strategy. The diffusion strategy again outperforms the projection strategy.

.

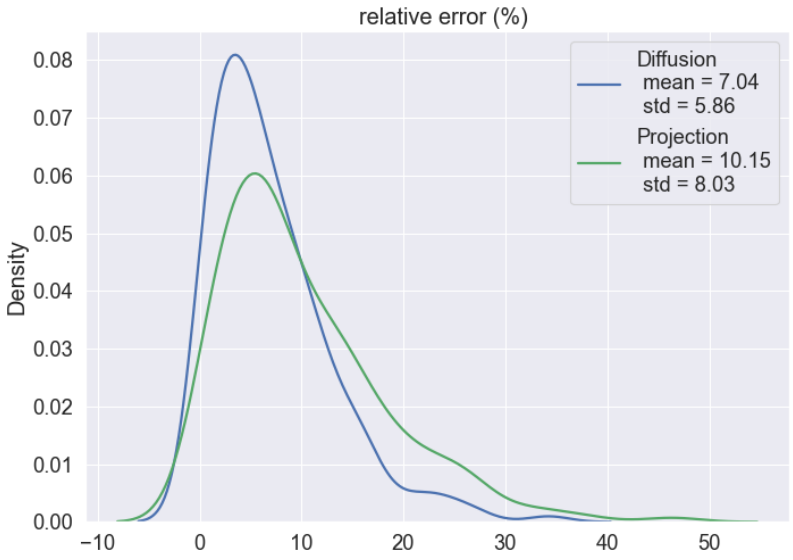


Figure 3.12: Distributions of errors between drag computed on ground truth propagated pressure fields and on predicted propagated pressure fields. The pressure field propagated with the diffusion strategy is easier to regress, yielding a lower relative error between prediction and ground truth.

3.4 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 [Guo et al., 2016; Thuerey et al., 2020; Chen et al., 2021], 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. 3.5 (the architectures of these two networks are detailed in Fig. 3.9 and Fig. 3.10). We configured this CNN to have roughly the same number of parameters as our MLP (200k vs. 150k parameters, respectively).

Fig. 3.13 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 3.1 quantifies the accuracy of our model and of the CNN baseline in terms of Mean

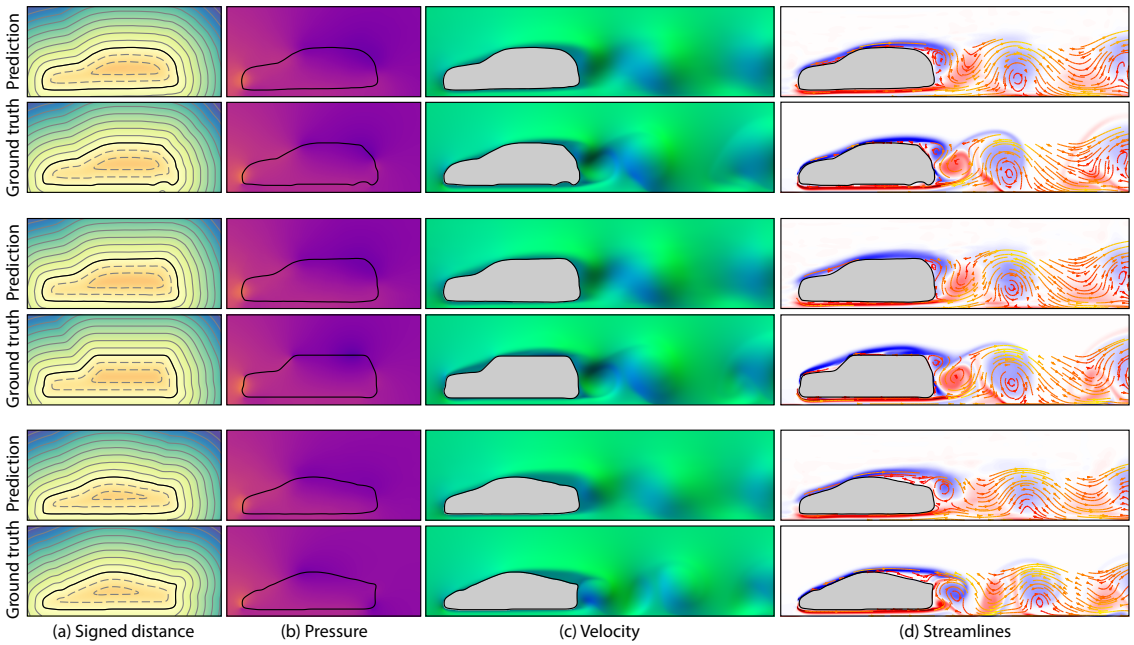


Figure 3.13: **Qualitative evaluation of fluid flow prediction.** Comparison between the distance, pressure and velocity fields predicted by our surrogate model, and the corresponding ground truth fields. We also include a streamline visualization to ease visual comparison of the velocity. Our model reproduces the overall behavior of the fluid flow, albeit smoothing out fine details.

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.
As we can see in the histogram of error for drag in Fig. 3.12, it exhibits a Gaussian-like shape, which is why we summarize it with mean and variance in Table 3.1.

Shape optimization. Fig. 3.16 illustrates results of our shape optimization for drag and vorticity attenuation without the volume preserving loss. For each result, we used our user interface to perform a few gradient descent steps such that the profile improves yet stays close to the input. We showcase our interface in `https://www.youtube.com/`

Table 3.1: **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	0.0189	0.0454	7.15%
	0.0013	0.0107	0.0153	5.74%
CNN	0.0037	0.0251		15.2%
	0.0016	0.0148		10.1%

watch?v=qseZFqFDLcM. In this video, we can see 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 now evaluate separately the different terms of our loss in the optimization process.

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.

We then evaluated the **improvement in vorticity**. For this quantity, the user has to select a region (mask) in the domain in which he wants the vorticity to decrease. It means that for a given shape, an infinite number of regions could be selected. This makes the evaluation a bit more difficult. While we have first been thinking about generating random masks to perform the optimization upon, we realized that this would lead, in many cases, to considering areas where there is almost no vorticity. After picking random shapes from the test set, we thus chose to manually select masks in which the predicted flow actually exhibits some vorticity. As we did for the evaluation of drag optimization, we then compared the predicted improvement to the one computed with the re-simulated ground truth. For an average gain of 20.53%, our prediction differs by 11.02%. Overall, only 13/25 optimizations led to actually optimized shapes when we re-simulate them. We propose some tracks to try explain the phenomena causing these errors and better understand them.

- The vorticity values are computed based on the velocity field. Since the networks tends to smooth the values, the estimation of the vorticiy field can vary a lot from its real value. One way to help overcoming this would be to directly predict the vorticity field.

- Proposing a shape that is not actually optimized can happen when, in the area described by the selected mask, the ground truth does not exhibits the same order of magnitude of vorticity as predicted by the network in the interface. This high range of values for the vorticity (way higher than for the drag) makes this optimization task very hard and can lead to exploding errors.

We eventually test our **loss term on volume**: We tried different values of β on different shapes from the test set. In Fig. 3.14, we can see that a compromise has been found with $\beta = 10^3$, for which the drag was more optimized than with $\beta = 10^5$, while keeping a similar volume, unlike with $\beta = 0$ where the shape is shrinked.
Testing with a different test car, we can see in Fig. 3.15 that the intermediate optimized shape, with $\beta = 10^3$, is in between the others, but still seems to optimize in priority the most important parts of the shape first.

We conjecture that the quality of the optimization process depends on the neighborhood of the test car in the dataset. In any case, the regularizing role played by the additional loss term is beneficial in the design process since it gives more guarantee for the optimization pipeline to produce valid optimized shapes.

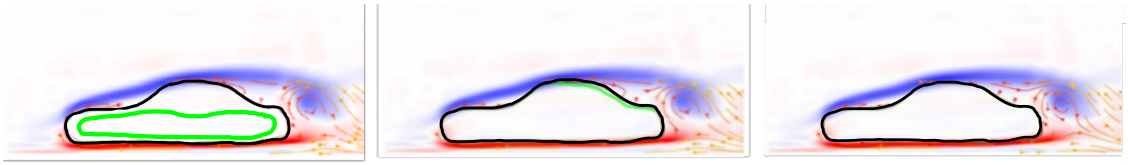


Figure 3.14: Other example of shape optimization in the interface with different coefficients. The original shape is in black and the optimized one in light green, with from left to right: $\beta = 0, \beta = 10^3, \beta = 10^5$.

Timings. Providing interactive feedback requires an efficient surrogate model. We now evaluate the inference time of our neural networks for different target problems, and compare them to the CNN baseline. All timings were measured on a computer equipped

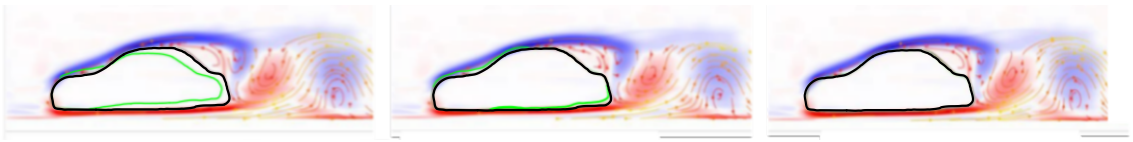


Figure 3.15: Shape optimization in the interface with different coefficients. The original shape is in black and the optimized one in light green, with from left to right: $\beta = 0, \beta = 10^3, \beta = 10^5$.

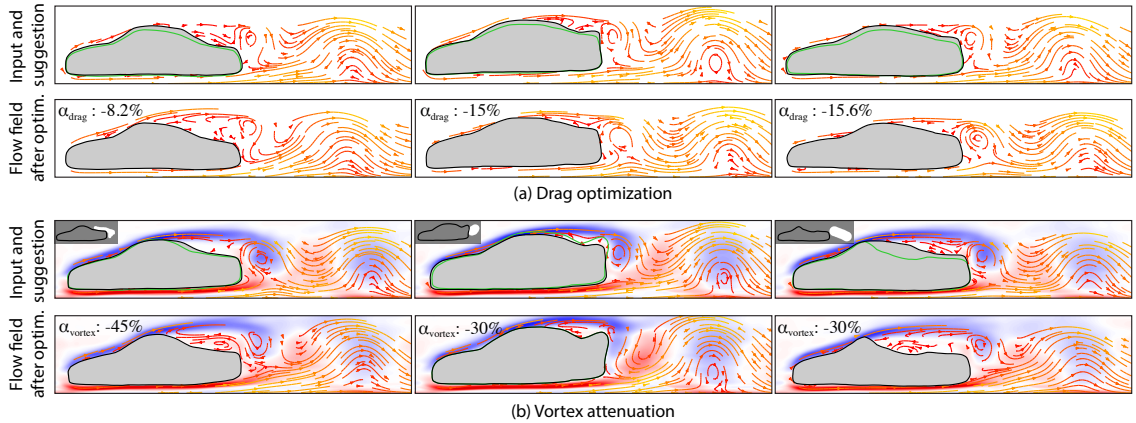


Figure 3.16: **Application to shape optimization.** Our method suggests effective shape modifications to reduce drag (a) or to attenuate vorticity (b). For each profile, we display the suggested improvement with a green outline (top) and then visualize the impact of the shape optimization on the flow field (bottom).

Table 3.2: **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			

with an Intel Xeon E5-2650 CPU (64GB RAM), and a single Nvidia RTX A5000 with 24GB of dedicated memory.

Table 4.1 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.

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. 3.5, which is negligible at distances greater than $5\,\sigma$ 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 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 $20,000$ samples 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. While we provide tools to circumvent the problem of latent codes exiting the densely sampled region illustrated in Fig. 3.17, these still need to be refined by the user if he wants to use its own dataset. Moreover, the smoothing effect of the neural surrogate models prevents our pipeline to consider sharp edges in the shape like rear spoilers. 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 for the drag, and regularly fails at precisely optimizing the vortex accumulation in an area, 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 [Raissi et al., 2019], or by treating the predicted fluid flow as an initialization for a precise, differentiable solver [Holl et al., 2020].

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.

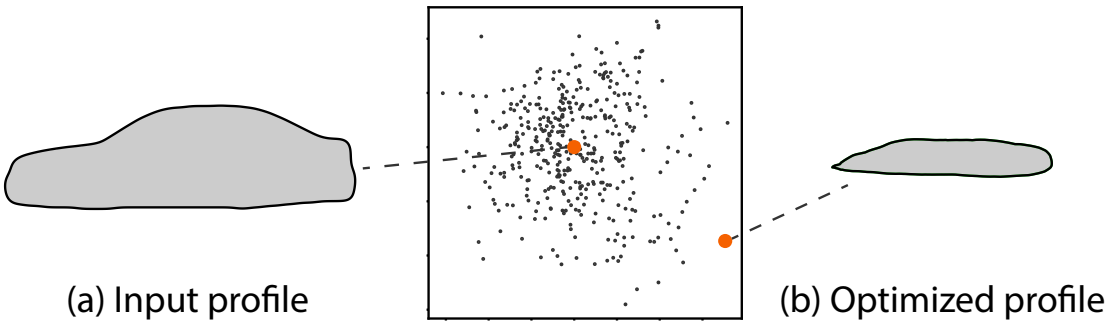


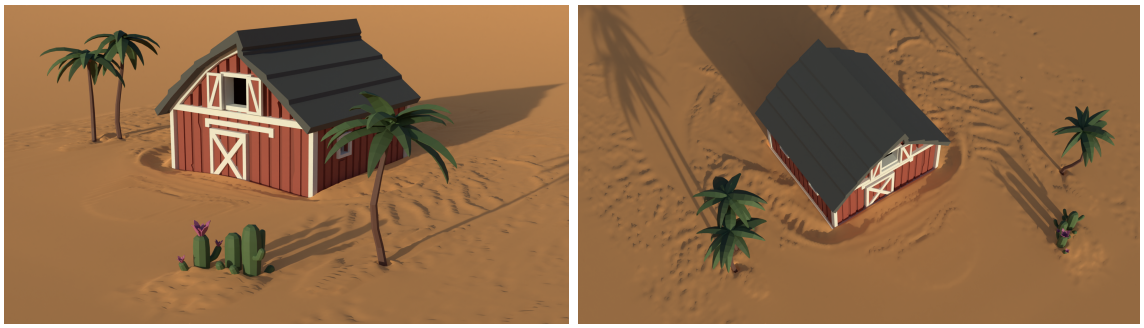
Figure 3.17: **Limitation.** Despite our volume conserving feature, shape optimization might travel outside of the relevant region of the latent space if coefficients are not chosen relevantly, producing profiles that do not resemble cars anymore (b). We alleviate this issue by letting users iterate the optimization until they find a proper trade-off between aerodynamic improvement and shape preservation.

3.5 Conclusion

We presented a tool to assist car designers in their sketching phase by producing aerodynamic feedbacks directly onto these sketches, and by proposing shape modifications to enhance the drag coefficient of the vehicle, or attenuate the vortex accumulation behind this vehicle. We further introduced an additional penalty term in the loss to enforce volume conservation in the shape optimization procedure.
We built our own dataset and focused on 2D cases to showcase our pipeline. By doing so, we proved the relevance of using neural networks for solving such tasks.
After considering large obstacles like vehicles, we will now have a look at how wind can transport tiny particles to shape natural landscapes.

Chapter 4

Windblown sand around obstacles – simulation and validation of deposition patterns



After studying a phenomenon arising from the interaction between the wind and obstacles, we take a look at these interaction from a different point of view. As described in Sec. 2.1, objects with different characteristics might occasion different phenomena when interacting with the air. While big objects like cars are slowed down by this fluid, others, like particles which smaller and lighter, can be *blown* by the wind. This is the case for particles, and in particular sand particle which size span from $60\mu m$ to some mm and usually weigh under $1\mu g$. The time span of the phenomenon is also different: cars are directly slowed down by the wind while it can take days or months for sand particles to accumulate and create sand dunes.

The sand dunes created by particles are iconic landmarks of deserts. Being able to correctly model their formation is of particular importance since around $\frac{1}{15}^{th}$ of the Earth's lands is covered by sand, mainly in the regions highlighted in Fig. 2.3. These areas feature very diverse wind conditions and terrain arrangements, including urban ones. These varying conditions under which sand is blown by the wind give rise to fascinating and varied landscapes, but also to unintended deposition of sand on infrastructures such as

buildings, roads, railways.

In this context, we present a simulator of sand erosion and deposition to predict how dunes form around and behind obstacles under the effect of wind. Inspired by both computer graphics and geo-sciences, our algorithm couples a fast wind flow simulation with physical laws of sand saltation and avalanching, which suffices to reproduce characteristic patterns of sand deposition. In particular, we validate our approach via a qualitative comparison of the erosion and deposition patterns produced by our simulator against real-world patterns measured by prior work under controlled conditions.

This chapter is mainly based on the following publication:

Nicolas Rosset, Régis Duvigneau, Adrien Bousseau, Guillaume Cordonnier, 2024. Windblown sand around obstacles-simulation and validation of deposition patterns. In Proceedings of the ACM on Computer Graphics and Interactive Techniques 1 (7) *pages 1-13*

4.1 Simulating windblown sand

We now describe our model to simulate sand transport and deposition under the effect of wind. While we seek for a simple model, we build it on laws derived from geo-science experiments.

4.1.1 Sand transport

As mentioned in Sec. 2.1, sand transport under wind is dominated by saltation and avalanches [Kok et al., 2012]. Most previous works in geo-sciences [O. Rozier, 2013] and computer graphics [Paris et al., 2019; Taylor and Keyser, 2023] model sand as a height field and rely on proxies to simplify the wind dynamics at the sand surface. As an exception, Lo Giudice and Preziosi [2020] considers a fully coupled 3D transport of both the wind and the sand. While we take inspiration from this coupled model by simulating 3D fluid dynamics, we note that saltation and avalanches occur at the surface of the terrain, making these phenomena amenable to a 2.5D representation that tracks the evolution of the sand thickness h over the terrain. In what follows, we allow for a non-flat ground of altitude g, and define the surface altitude as $f = h + g$.

In computer graphics, transport of sand [Paris et al., 2019] or snow [Cordonnier et al., 2018b] is simplified as directly linked to the wind velocity: sand is eroded by strong wind, and is deposited when the wind weakens or is shadowed by obstacles. While this approach is easily controllable and fits well with procedural wind fields, it fails to capture the complex erosion and deposition patterns induced by physically-based 3D wind flows. Instead, we propose a formulation based on a sand transport rate $\mathbf{Q}$ (unit: kg/m/s), which readily expresses the evolution of the sand thickness through volume conservation:

$$\frac{\partial h}{\partial t} + \frac{1}{\rho}\nabla \cdot \mathbf{Q} = 0, \tag{4.1}$$

where ρ is the sand density. Intuitively, if we consider a small volume of sand, the sand transport rate $\mathbf{Q}$ expresses the mass of sand that flows through the bound of the volume, and its divergence gives the net balance in sand mass per time unit.

We split the sand transport rate into two terms: $\mathbf{Q} = \mathbf{Q}_s + \mathbf{Q}_a$, where $\mathbf{Q}_s$ accounts for the sand moved by saltation, while $\mathbf{Q}_a$ models the sand moved by avalanching. G. Strypsteen [2021] compares several formulations of saltation rate $\mathbf{Q}_s$ and suggests that the following

law by Zingg [1952] is a good fit to measured data:

$$\mathbf{Q}_s = \begin{cases} k_s \|\mathbf{u}_*\|^2 \mathbf{u}_*, & \text{if } h > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{4.2}$$

where k_s depends on the grain size; we use $k_s = 0.13\, kg.s^2 m^{-4}$ which corresponds to a median grain size of $1.3mm$, and $\mathbf{u}_*$ is the shear velocity of the wind at the sand surface. The shear velocity is used in fluid mechanics to characterize the flow velocity profile at the wall boundary, and therefore represents here the capacity of the flow to entrain sediments.

We note that the wind velocity is null at the ground altitude, and therefore, linearizing the velocity profile at the vicinity of the ground gives the following relationship:

$$\mathbf{u}_* = \frac{1}{50\epsilon} \mathbf{u_t}(f + \epsilon), \tag{4.3}$$

where $\epsilon = 10^{-1} m$ is the distance to the flow surface, and we obtained the coefficient $1/50$ by comparing our velocities with the velocities and shear velocities reported by G. Strypsteen [2021] from real measurements.

Sand is a granular material and as such flows down when the surface slope exceeds a critical *Coulomb angle* θ_{cr}. We express this *avalanching* process in the avalanche rate $\mathbf{Q}_a$. We set $\mathbf{Q}_a = 0$ if $h = 0$, otherwise we follow a rotation-invariant expression from geo-sciences [Lo Giudice et al., 2019]:

$$\mathbf{Q}_a = k_a \nabla f \frac{\max(\|\nabla f\| - \tan\theta_{cr}, 0)}{\|\nabla f\| \sqrt{1 + \|\nabla f\|^2}}. \tag{4.4}$$

The constant k_a depends on the adhesive properties of the sand but is poorly constrained [Lo Giudice et al., 2019; Lo Giudice and Preziosi, 2020]. Compared to the time scale of saltation, avalanches are much faster and can be considered at equilibrium. Therefore, we assign a specific timestep to this phenomenon and set $k_a\, dt_a = 0.24 \rho dx^2$ – a compromise between numerical stability and fast convergence to equilibrium.

Discretization. We use a 2D staggered grid embedded in the heightmap (Fig. 4.1). We store the elevations at the centers of the cells (*e.g.*, $h(i, j)$ for a cell centered at position i, j), while we store the transport rates and shear velocity at the interfaces

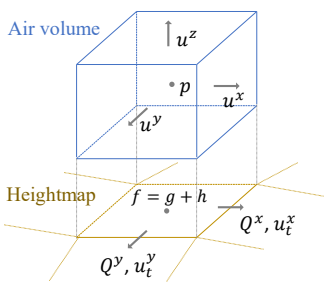


Figure 4.1: Notations.

between cells (*e.g.*, $Q^x(i+1/2, j)$ or $Q^y(i, j+1/2)$). The surface gradient ∇f arising in the avalanche term naturally complies with this discretization, but the norms $\|\mathbf{u}_*\|$ and $\|\nabla f\|$ require interpolating the vector components to obtain their values at the flux locations. Note that when the surface is not flat, the tangential velocity (and hence the shear velocity) has a vertical component. This component is discarded in the definition of the transport rate but is still required for the computation of the norm of the shear velocity and therefore stored in the center of the cells.

The cases $\mathbf{Q}_s = 0$ and $\mathbf{Q}_a = 0$ when $h = 0$ require checking the presence of sand at the cells interface, while h is stored at the center. Furthermore, a largely divergent transport rate where h is small can lead to a negative sand thickness. To solve both issues, we consider independently each component of the transport rate, stored at a cell interface, *e.g.*, $Q^x(i+1/2, j)$. We want to limit the amount of material transported by the amount of material available in the incoming cell, called the *upwind* cell. We determine the cell *upwind* to an interface from the sign of the transport rate component stored on that interface (*e.g.*, (i, j) if $Q^x(i+1/2, j) > 0$), and denote by the subscript $*_{up}$ a quantity stored in the upwind cell. Finally, we adjust the transport rate to the maximum amount that can be transported from the cell *upwind* during a time dt: $Q_{max} = h_{up} \frac{\rho\, dx}{dt\, N_{up}}$ where $N \in \{1, \ldots, 4\}$ is, for each cell, the number of interfaces where the sign of the transport rates indicates that the sand leaves the cell. The final transport rate is then given component-wise ($i \in \{x, y\}$) by:

$$Q^i = \mathrm{clamp}(Q^i_s + Q^i_a, -Q^i_{max}, Q^i_{max}) \tag{4.5}$$

4.1.2 Wind flow

We assume that the wind is a Newtonian fluid and follows the incompressible Navier-Stokes Equation:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \nu \Delta \mathbf{u}, \tag{4.6}$$

where $\rho = 1.293 kg\, m^{-3}$ is the air density and $\nu = 1.5\, 10^{-5} m^2\, s^{-1}$ is the air viscosity. We add the incompressibility condition

$$\nabla \cdot \mathbf{u} = 0. \tag{4.7}$$

We do not account for turbulence phenomena, which could be present in real life experiments, due to their complexity of implementation. Nevertheless, turbulence mainly acts as additional diffusion effects, which are inherently included in numerical diffusion.

We set the boundary conditions $\mathbf{u} = 0$ at the sand-air interface and we force a constant air inflow $\mathbf{u} = u_{\text{inflow}}$ on the front, left, right and top sides of the domain. We leave an open boundary condition at the back side, with Neumann boundary conditions for the velocity and Dirichlet conditions ($p = 0$) for the pressure.

We use Stable Fluid's [Stam, 1999b] Helmholtz decomposition: we first advect the velocity with an RK3 semi-lagrangian scheme, then we compute a pressure field that projects back the residual velocities into the divergence-free space. The Poisson problem arising from the pressure projections requires additional Neumann boundary conditions at the interface with the sand and the inflow domain bounds: $\mathbf{n} \cdot \nabla p = 0$, where $\mathbf{n}$ is the normal to the obstacle or to the fluid domain. We use a sparse preconditioned conjugate gradient method for the pressure projection. Specifically, we run it with a threshold of $t = 10^{-6}$ and a maximum number of 6000 iterations as our convergence parameters.

Sand-wind interactions. We store the wind velocity and pressure in a 3D staggered grid (Fig. 4.1), with each three components of the velocity stored on the faces of the cell and pressure at the center of the cells. We align the 2D staggered grid of transport rates with the 3D staggered grid of wind velocities. Indeed, the transport rate for saltation depends on the shear velocity, itself depending on the wind velocity tangent to the sand surface. To compute the tangent velocity, we use a vertical linear interpolation of the wind velocity to fetch it close to the sand surface ($\mathbf{u}(f + \epsilon)$), and we deduce the tangential velocity with $\mathbf{u}_t = \mathbf{u} - \mathbf{n}\,(\mathbf{u} \cdot \mathbf{n})$. Note that non-flat sand surfaces lead to a vertical tangential velocity component, which is stored at the center of the 2D staggered grid. It might be confusing to handle this vertical component because the saltation transport rate $\mathbf{Q}_s$ is inherently 2D and only embeds the two horizontal components, but the computation of $\mathbf{Q}_s$ in Eq. 4.2 includes the norm of the 3D shear velocity which does include a vertical component.

The sand surface itself acts as a boundary condition for the wind. Encoded in a heightmap, the altitude values freely vary vertically and do not conform to a specific cell or intercell altitude. More generally, handling non-regular domains is a common problem in computer graphics, and we follow [Ng et al., 2009] who estimate the proportion α of the cell's face

that intersects with the free air volume. Following a finite-volume approach, they scale the pressure gradients and the velocity by these proportions, effectively changing the equation for pressure projection to:

$$\nabla \cdot \alpha \nabla p = -\frac{\rho}{dt} \nabla \cdot \alpha \bar{\mathbf{u}}, \tag{4.8}$$

where α is the free fluid proportion and $\bar{\mathbf{u}}$ is the intermediate velocity after advection. We also set the velocity to $\mathbf{u} = 0$ when the associated face is entirely below the surface ($\alpha = 0$).

4.1.3 Summary of our algorithm

In summary, our algorithm starts with an initial heightmap with ground altitude g and sand thickness h, and velocities initialized at $\mathbf{u} = 0$ except for the prescribed inflow values. Then, we iterate over the following steps:

- Compute the proportions of the cell interface α intersecting with the fluid domain above the sand surface $f = h + g$.
- Project $\bar{\mathbf{u}}$ in the divergence-free space by solving Eq. 4.8. This results in the wind velocity $\mathbf{u}$.
- Fetch $\mathbf{u}$ close to the sand surface f (altitude $f + \epsilon$), deduce the tangential velocity $\mathbf{u}_t$ and then the shear velocity $\mathbf{u}_*$ from Eq. 4.3.
- Compute the saltation transport rate ($\mathbf{Q}_s$, Eq. 4.2) and the avalanche transport rate ($\mathbf{Q}_a$, Eq. 4.4), compute the component-wise maximum flux Q_{max} and deduce the final transport rate $\mathbf{Q}$ from Eq. 4.5.
- Evaluate erosion/deposition from mass conservation (Eq. 4.1) and update the sand thickness h.

In practice, we let the fluid flow simulation run for 100 iterations before starting the erosion and deposition processes.

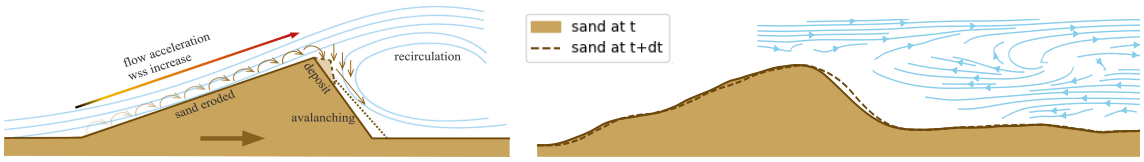
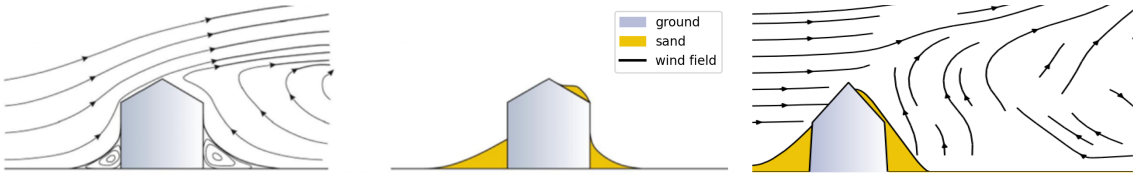


Figure 4.2: Evolution of a transverse dune, schematically (a, from Lo Giudice and Preziosi [2020]) and simulated by our method (b). Note that we produced this figure by running a 3D simulation and extracting a 2D slice of the sand bed and wind field, yielding more complex vortex patterns than the 2D idealized case. Our method (b) aligns well with the idealized scenario (a) by eroding sand on the left side of the dune, which faces the wind, and by depositing sand on the right side of the dune, which faces the recirculation area where the wind forms a vortex.



(a) Schematic windfield and deposition (b) Our simulation

Figure 4.3: Windfield and sand deposition over a gable roof, schematically (a, from L. Raffaele [2019]) and simulated by our method (b). Note that we produced this figure by running a 3D simulation and extracting a 2D slice of the sand bed and wind field.

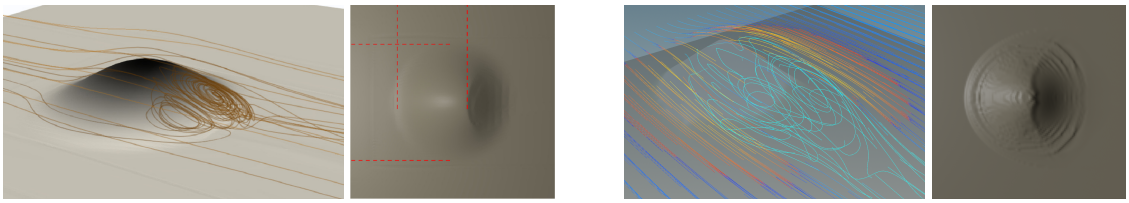
4.2 Results

We validate our algorithm by reproducing sand deposition patterns documented in prior work. We start with idealized patterns for which we have reference diagrams or simulations, followed by real-world patterns for which we have reference measurements.

4.2.1 Idealized deposition patterns

Transverse dune. Prior work describes how transverse dunes exhibit a characteristic triangle profile that moves in the direction of the wind as sand gets eroded on the side facing the wind, and gets deposited on the side facing away from the wind, where wind recirculation occurs [Bruno and Fransos, 2015; Lo Giudice and Preziosi, 2020]. Fig. 4.2a illustrates this configuration schematically. Fig. 4.2b shows that running our simulator on a triangle-shaped dune for a single time step produces a similar behavior.

Gable roof. Predicting sand deposition over human-made structures is of critical importance to preserve such structures. [L. Raffaele, 2019] report typical sand deposition



(a) Reference windfield and resulting barchan dune [Lo Giudice and Preziosi 2020]

(b) Our windfield and simulated dune

Figure 4.4: Formation of a barchan dune by blowing wind on a Gaussian-shaped pile of sand. Reference simulation by Lo Giudice and Preziosi [2020](a) and our simulation (b). Our method produces similar results yet employs a faster fluid solver and only considers saltation and avalanching.

(a) Our simulation of a barchan dune

(b) Real barchan dune

Figure 4.5: Visual comparison between our simulation (a) and the picture of a real barchan dune (b, Mesquite Sand Dunes in the Death Valley by Daniel Mayer, Wikimedia Commons).

(a) Star dunes simulated by Zhang et al. [2012]

(b) Our simulations

Figure 4.6: Formation of star dunes by blowing wind on a truncated cone of sand, alternating over three (left) and five (right) wind directions. We refer to Fig. 2a and Fig. 2c from Zhang et al. [2012] for a comparison.

patterns around common structures, including gable roofs (Fig. 4.3(a)). Our method produces similar patterns on a simple house, with sand deposition against the front and back wall, as well as over the back side of the roof (Fig. 4.3(b)).

Barchan dune. Barchans are crescent-shaped dunes that commonly appear in sand deserts all over the world [Courrech du Pont, 2015]. Lo Giudice and Preziosi [2020] reproduced such dunes by initializing their high-end simulator with a Gaussian-shaped pile of sand under a strong uni-directional wind. After a few time steps, the crescent shape emerges. Fig. 4.4 reports their visualizations, along with the results of our simulator. While approximate, our fast method reproduces well the characteristic shape of the barchan. Fig. 4.5 illustrates the result of our simulation when we let it run longer, along with the picture of a real barchan dune for visual comparison.

Star dune. [Zhang et al., 2012] used cellular automata to simulate the formation of star dunes. In their experiments, star dunes emerge from a truncated conical sand pile when wind alternates over several directions. They observed that star arms form in the presence of an odd number of wind directions, the number of arms being equal to the number of directions. We reproduced their experiment for wind alternating over three and five directions, and obtained similar star-shaped patterns, as shown in Fig. 4.6. However, because this experiment amounts to simulate wind flow over multiple, alternating expositions, it requires more iterations and computation time than our other experiments (Table 4.1).

4.2.2 Real-world patterns

We now compare our results to sand deposition patterns measured in real-world experiments. In particular, we build on the extensive series of experiments conducted by Poppema et al. [2022], who measured sand deposition patterns around cuboid shapes placed under diverse configurations (spacing, orientation with respect to the wind) on a windy sand beach in the Netherlands.

We reproduced these configurations with our simulator and provide a visual comparison to their measurements in Fig. 4.7. We adjusted the wind speed and viscosity of our simulator to best match their measures. While qualitative, this evaluation shows that our simulator reproduces well the erosion and deposition pattern around obstacles, as a function of the spacing between the obstacles and the inclination of the wind.

We also use measurements from Poppema et al. [2022] as a reference to compare our method to the real-time simulator of Taylor and Keyser [2023], which extends the work of Paris et al. [2019] to handle obstacles (see Fig. 4.8). This simulator is based on stochastic

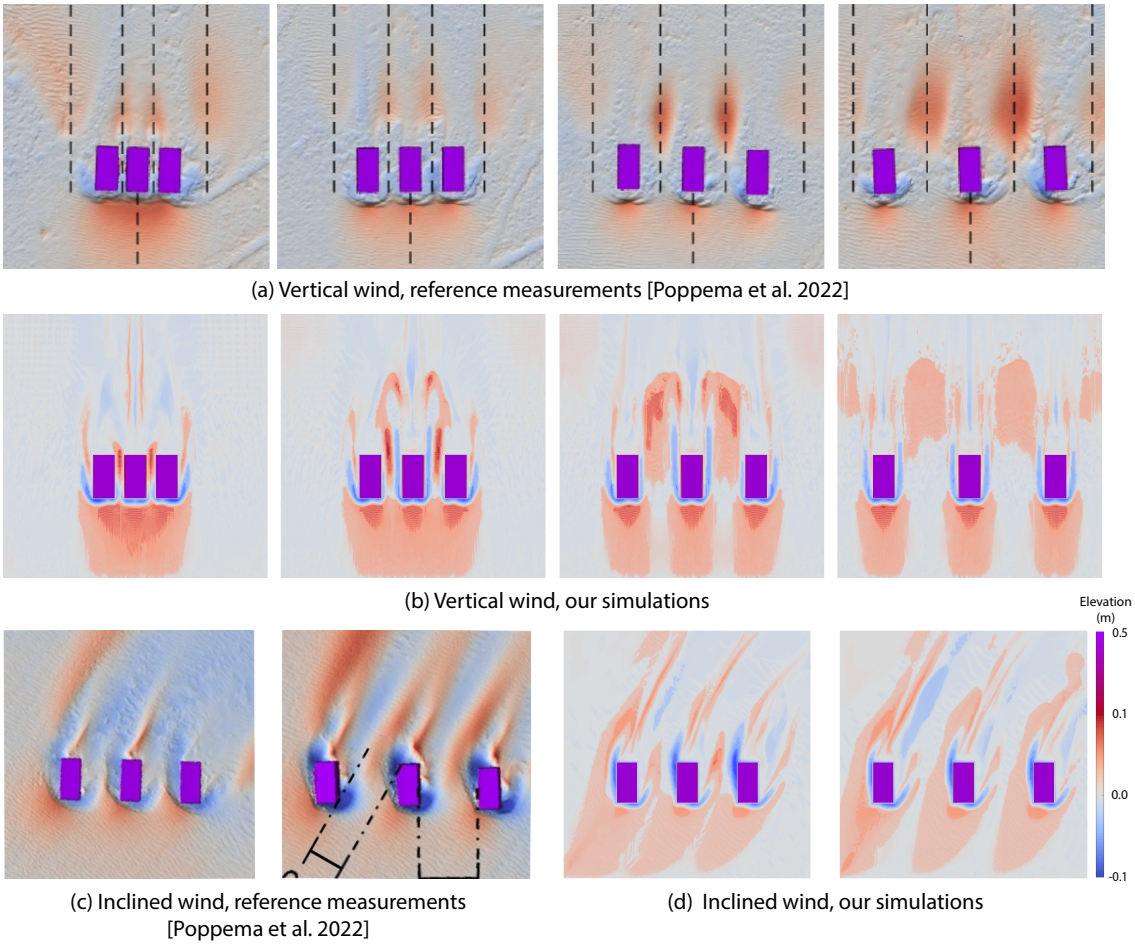
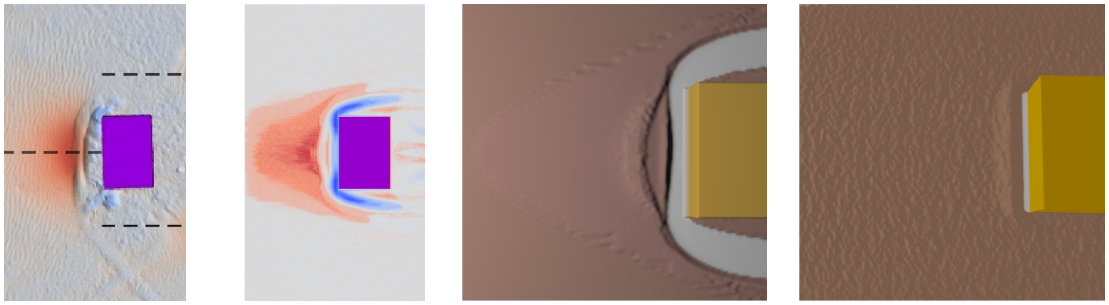


Figure 4.7: Comparison to real-world measurements by Poppema et al. [2022] for different configurations of cuboids. Blue denotes erosion and red denotes deposition. Our method faithfully reproduces sand erosion and deposition patterns for cuboids with varying spacing under a vertical wind (a,b) and inclined wind (c,d). Note in particular how sand is eroded in front and around the cuboids, and deposited behind and between the cuboids as they are more spaced.

rules that erode sand in front of the obstacle and deposit sand behind the obstacle. As such, this simulator cannot reproduce the complex erosion and deposition patterns observed in the real-world experiment, where the wind flow wraps around the obstacle, creating arc-shaped sand ripples in front and on the side of the cuboid. On the contrary, our method is able to reproduce such patterns.



(a) [Poppema et al. 2022] (b) Our simulation (c) Our simulation, shaded (d) [Taylor and Keyser 2023]

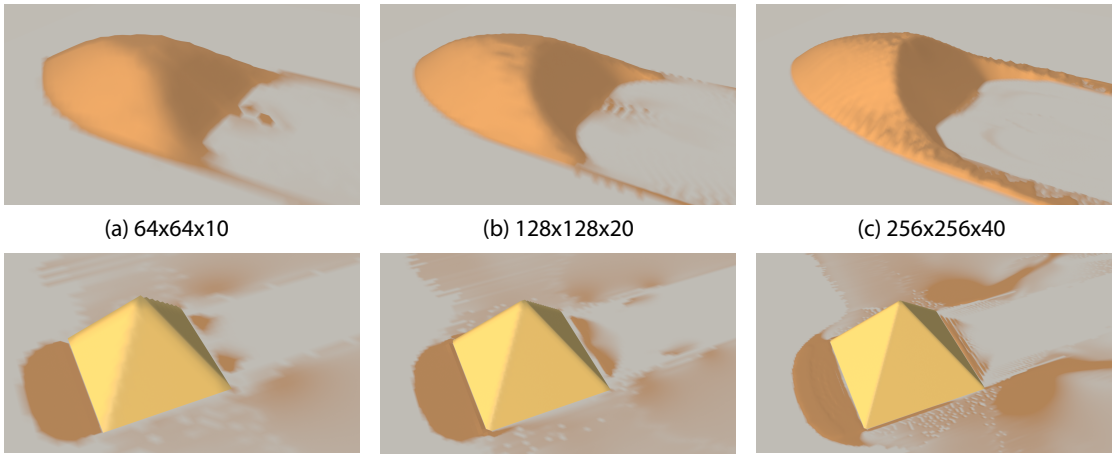
Figure 4.8: Comparison to the real-time method by Taylor and Keyser [2023]. The measurements by Poppema et al. [2022] provide a reference of how sand erodes and deposits around a cuboid (a). Our simulator reproduces well this pattern (b,c), where sand level increases progressively in front of the object, then drops quickly and rises again to form a thin secondary dune, before dropping again to form a crescent-shaped cavity that wraps around the cube. In contrast, the procedural rules of Taylor and Keyser [2023] only produce a straight dune and cavity parallel to the front face of the cube (d, reproduced from Taylor and Keyser [2023]). Note that we used a lower initial sand thickness in (c) to better match the conditions in (d).

4.2.3 Impact of resolution

The spatial accuracy of our simulator depends on the resolution of the underlying heightmap and volumetric grid. Fig. 4.9 (top) visualizes barchan dunes obtained by simulating unidirectional wind over a Gaussian pile of sand, on spatial domains of increasing resolution. The characteristic crescent shape of the dune emerges in all cases, but the ridge of the dune becomes sharper at higher resolution. Fig. 4.9 (bottom) shows the result of simulating a thin bed of sand under unidirectional wind around a pyramid, at different resolutions. At higher resolution, the sand deposition in front of the obstacle and in the wake of the flow is more detailed. But high resolution comes at an increased computational cost, as reported in Table 4.1 and discussed next.

4.2.4 Time performance

We implemented our algorithm in Python, using PyTorch for easy GPU integration. We provide our source code at `https://gitlab.inria.fr/nrosset/windblown_sand`. All timings are reported from a desktop computer equipped with a single NVidia RTX 3070 GPU with 8 GB dedicated memory. We ran experiments on domains with varying discretizations – more complex scenes requiring higher resolutions (Table 4.1). All simu-



(a) 64x64x10 (b) 128x128x20 (c) 256x256x40

(d) 64x64x10 (e) 128x128x20 (f) 256x256x40

Figure 4.9: Rendering of the simulations at different resolutions of a barchan dune emerging from a gaussian stack of sand (a, b, c), and of a thin layer of sand blown of a pyramid (d, e, f). While the different resolutions yield a similar pattern of depositions overall, higher resolution allows to capture sharper details.

lations shown in this Chapter took between a few seconds to a few minutes, depending on the size of the grid and the number of iterations, which is orders of magnitude faster than the complex simulators used in engineering [Lo Giudice and Preziosi, 2020]. Since all the erosion/deposition updates are made on a 2D grid, the bottleneck for our time performances is the 3D fluid simulation, and in particular, the pressure projection that requires the inversion of a matrix. The number of iterations here specified is the number of times we performed this projection during a given simulation.

4.2.5 Limitations

In addition to missing long-range sand transport modes (such as suspension in higher altitudes), the main limitations of our simulator stem from our representation of the ground as a heightfield defined over a grid. This representation is better suited to represent axis-aligned shapes, such as the cuboid in Fig. 4.8, than curved shapes that suffer from aliasing artifacts. Furthermore, heightfields cannot represent obstacles with overhangs, like arches and fences, while such obstacles can form effective shielding structures [Bruno et al., 2018].

experiment	resolution	# of iterations	time
Fig. 4.2	128*128*30	150	0min14s
Fig. 4.3	128*128*30	3500	5min32s
Fig. 4.4	128*128*30	150	0min14s
Fig. 4.5	128*128*30	600	1min14s
Fig. 4.6	256*256*30	72000	55min30s
Fig. 4.7	256*256*30	1300	9min10s
Fig. 4.9 (a)	64*64*10	200	0min18s
Fig. 4.9 (b)	128*128*20	400	0min50s
Fig. 4.9 (c)	256*256*40	800	5min58s
Fig. 4.9 (d)	64*64*10	120	0min11s
Fig. 4.9 (e)	128*128*20	240	0min28s
Fig. 4.9 (f)	256*256*40	440	3min02s

Table 4.1: Timings for the main results in this Chapter.

4.3 Conclusion

We introduced a simulator that combines the physical phenomena responsible for wind-blown sand deposition patterns. Specifically, a fluid flow simulator captures the wind dynamics over a heightfield, and drives the transportation of sand through a process called saltation. We derive the erosion and deposition of sand from the balance of saltation and of another transport mechanism, avalanching, that dominates when the slope of the sand surface exceeds a critical angle. Our experiments demonstrate that our method can reproduce reference patterns from alternative methods and field studies, and capture both the transient emergence of dunes and the static buildup of typical sand patterns around obstacles. While the simplicity of our simulator makes it faster than the ones used in engineering, it does not yet reach the speed necessary for the interactive design of obstacles for sand mitigation measures [Raffaele et al., 2022]. In this context, similarly to Chapter 3, a learning pipeline could rely on our simulator to train fast data-driven surrogate models suitable for inverse design applications.

Chapter 5

Neural surrogate model for wind prediction above a terrain

We showed in the previous Chapter that an approximate 3D fluid solver could be queried in conjunction with a sand transport algorithm to perform sand erosion/deposition in a scene and obtain fast and realistic results. Although very efficient, this formalism showed limitations.

First, the fluid solver we used outputs velocity fields that evolve over time. However, the sand bed evolves slowly over the iterations and the influence of the successive wind frames smooths out. This suggests that averaged values of the wind field could be sufficient for sand transport. Second, this fluid solver computes the velocity fields on a regular grid, which introduces *staircase artifacts* on the resulting sand dunes.

Driven by these motivations, and the promising results of *surrogate models* and *physics-informed losses* that we described in Sec. 2.3.5, we propose a self-supervised neural network that maps a height map to the average velocity values of the wind field layered above it. Adopting a time-averaged formulation of the wind alleviates the need to model its temporal fluctuations. Embedding the wind field into a deformed grid allows to better align boundary conditions to the sand bed and avoid artifacts.

We present in this Chapter preliminary work and results about developing such a surrogate model. In concrete terms:

- We investigate the use of a mapping between the terrain heights and a single wind field, corresponding to the wind velocities averaged over time. The values of this wind field are stored in a deformed grid to form layers aligned with the terrain.
- We develop a pipeline for encoding this mapping with a convolutional neural network, and show the benefits of such an architecture in a limited use case mimicking an iteration of sand transport

5.1 Overview

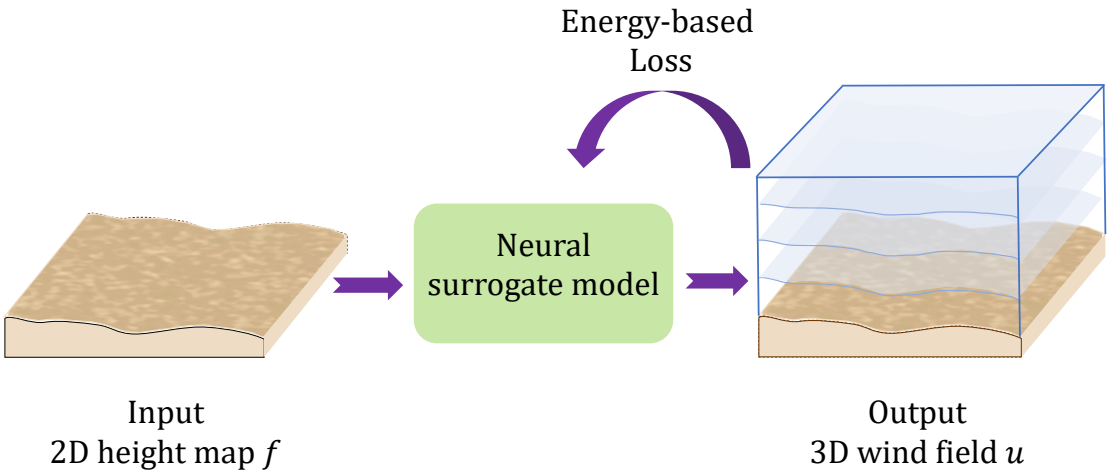


Figure 5.1: Our system takes as input a 2D height map f and predicts the 3D time-averaged velocity field u of the wind flowing above it. We use a neural surrogate model to map the terrain to the wind field, and propose a physics-based energy formulation for our loss, that allows the optimization of the neural network parameters in an unsupervised way. This framework enables on the fly refinement of our surrogate model when the terrain evolves.

Driven by the long term goal of getting a fast and invertible sand simulation pipeline, we investigate neural-based solutions for predicting the wind above a terrain. Given the long periods upon which the sand is blown by the wind and the small movements resulting, we conjecture that sand transport can be approximated by considering an average wind. Getting this averaged wind is complicated because it requires running simulations and averaging the velocities computed. Instead, we propose to use a neural surrogate model that takes as input the 2D terrain height map and directly outputs the 3D averaged velocity field of the wind flowing above the terrain. However, we want to avoid relying on a data-based learning approach for optimizing our neural network, which involves the creation of a costly dataset and prevents generalization outside the training data. To circumvent this challenge, we introduce a training loss based on an energy formulation of the underlying physics.
Practically, we store the terrain on a regular grid of elevations (height map) and the 3

components of the wind vector field on a staggered grid. We align this grid vertically with the terrain and adapt the elevation of the horizontal wind layers to accommodate for the terrain variations.
Our formulation permits on the fly refinement that makes our solution robust to changes in the input topography.

5.2 Method

5.2.1 Neural-enhanced optimization

We focus in this Chapter on the wind field prediction above a terrain. For this, we propose a neural surrogate model that maps a terrain to the air flow above it. We propose to use a convolutional neural network as our surrogate model. As illustrated if Fig. 5.2, we present a pipeline that takes as input a height map, and outputs the average wind field above it. Before inputing the height map to the convolutional neural network, we pre-process it by using stacked gaussian pyramids, as used in [Lefebvre and Hoppe, 2005], to capture the terrain geometry at multiple scales, allowing the neural network to account for information over long distances.

We expect our method to be particularly well suited for simulations of sand transport, and ultimately to inverse problems related to the positioning of sand dunes. Indeed, during the simulation of erosion and deposition, the terrain does not change significantly between subsequent keyframes. If the network is trained to output a velocity at time t, we expect the cost to finetune it to a slightly changed terrain at frame $t + dt$ to be low. Furthermore, neural networks are particularly suitable for inversion as they are a combination of simple operations, and that leveraging auto-differentiation makes this operation direct. We denote as n_w the function associated with this neural network, w its weight parameters, and $\mathbf{u}$ the velocity field this network predicts:

$$\mathbf{u} = n_w(f) \tag{5.1}$$

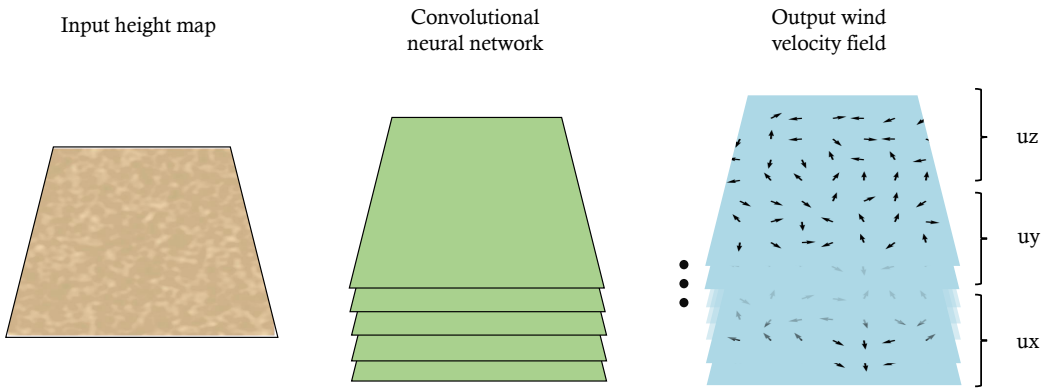


Figure 5.2: Neural network mapping the height map described as 2D grid to the 3D wind field, with its 3 components stacked. Practically, we transform the input height map by pyramid gaussians to get a 6 channels input. We feed this to a standard convolutional neural network with 5 layers with respectively 32,64,128,256 and 180 channels. From a height field of size 256×256, we thus output a velocity field of size $256 \times 256 \times 60\times$ with 3 components.

5.2.2 Energy-based fluid solver

While we could consider building a dataset by running many air simulations with the solver in Chap. 4 above various terrains to train our model, we want our output fields to have a stable precision no matter the input height map without depending on the similarity with a training dataset. We therefore rely on a self-supervised method.
Similarly to Chap. 4, we assume that the wind is a Newtonian fluid and follows the incompressible Navier-Stokes Equation:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \mu \Delta \mathbf{u}, \tag{5.2}$$

where $\rho = 1.293 kg\, m^{-3}$ is the air density and $\mu = 1.5\, 10^{-5} m^2\, s^{-1}$ is the air viscosity. We add the incompressibility condition:

$$\nabla \cdot \mathbf{u} = 0. \tag{5.3}$$

We believe that predicting the averaged velocities of the wind is enough. By decomposing the velocity as a sum of its averaged value and fluctuating value $\mathbf{u} = \overline{\mathbf{u}} + \mathbf{u}'$, inserting it in Eq. 5.2, and averaging the equation, we obtain the Reynolds-Averaged-Navier-Stokes equation. This equation introduces an additional Reynolds stress term. Fortunately, this equation was well studied in the CFD community and we can model this additional term as a viscosity term. We thus obtain the following equation:

$$\begin{cases} \underbrace{\overline{\mathbf{u}} \cdot \nabla \overline{\mathbf{u}}}_{\text{advection}} + \underbrace{-\frac{1}{\rho} \nabla \overline{\mathbf{p}}}_{\text{pressure}} - \underbrace{(\mu + \mu_{turb}) \Delta \overline{\mathbf{u}}}_{\text{viscosity}} = 0 \\ \nabla \cdot \overline{\mathbf{u}} = 0 \end{cases} \tag{5.4}$$

In practice, we set $\mu_{turb} = 0.5$. A promising way to solve for these equations is using the physics-informed losses we described in Sec. 2.3.5. However, instead of directly using residuals for the partial differential equations, [Li et al., 2021] and [Cordonnier et al., 2023] showed that variational counterparts have better convergence behavior. Indeed, compared to optimizing directly the strong form of Eq. 5.2, the variational form does not require a hyper-parameter search, and has a good convexity. Inspired by these works, we propose to use an energy term for the loss of our neural network. This energy takes as input the height map and outputs the velocities of the wind field.

We define the energy term as the sum of the three components of Eq. 5.4.

$$E = E_{viscosity} + E_{pressure} + E_{advection} \tag{5.5}$$

And we use this energy as a loss to be minimized.

$$\min_{w} E(n_w(f)) \tag{5.6}$$

where w are the weights of the neural network, and f the input height field above which we predict the wind.

We now detail how we compute the three energy terms to solve our optimization problem. We consider a 3D domain $\mathcal{D}$.

Viscosity energy: First, following [Takahashi and Batty, 2020], we use the variational form of the viscosity term, and obtain the expression:

$$E_{viscosity} = \frac{\mu}{2} \int_{\mathcal{D}} \|\nabla \mathbf{u}\|^2 \, d\tau \tag{5.7}$$

Pressure energy: We then use a penalty approach for the pressure [Xiao-liang and Shaikh, 2006] to push the optimization procedure to produce a divergence-free field. Formally, we have:

$$E_{pressure} = \alpha_p \int_{\mathcal{D}} (\nabla . \mathbf{u})^2 \, d\tau \tag{5.8}$$

where α_p is a penalty coefficient that we empirically set as $\alpha_p = 10$.

Advection energy: There is no closed-form expression for an energy term obtained as a variational form of the advection term of Eq. 5.4. However, by construction we know the gradient of this term with respect to the velocity:

$$\frac{\partial E_{advection}}{\partial u} = \mathbf{u} \cdot \nabla \mathbf{u} \tag{5.9}$$

Because we will optimize our neural network by using a first-order gradient descent optimization to minimize E, we only need the gradients of $E_{advection}$.

Finally, for our wind problem, we set Dirichlet boundary conditions on the top and inflow sides of our domain $\mathcal{D}$ to impose the input velocity. We force a null velocity at the interface with the terrain geometry, and free boundary conditions for the other lateral bounds. Equipped with this differentiable energy formulation, we now use it to solve for Eq. 5.6. We describe how to do it in the following sections.

5.2.3 Operating on a deformed grid

Solving for Eq. 5.6 requires minimizing the aforementioned loss on a domain. To better account for the boundary conditions of the terrain heights, and solve the artifact problems

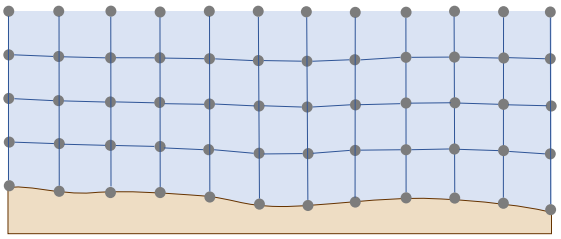


Figure 5.3: Deformed grid to store wind above terrain heights. The grid is morphed between the height at the bottom and a horizontal maximum height on top to produce more accurate velocity fields at the bottom boundary of the domain.

mentioned in Sec. 4.3, we propose to use a deformed grid to discretize the integrals. We show a schematic example of such a grid in Fig. 5.3.

To do so, we generalize our operators to perform in a non-regular grid. Indeed, the z coordinate does not evolve uniformly anymore but depends on x and y. To compute the spatial derivatives of the flow field, we introduce the notion of total derivative which represents the derivative in the direction of the mesh. For the velocity $u(x, y, z)$, the total derivative writes:

$$\frac{d\mathbf{u}}{dx} = \frac{d\mathbf{u}(x, y, z(x, y))}{dx} = \frac{\partial \mathbf{u}}{\partial x} + \frac{\partial \mathbf{u}}{\partial z}\frac{\partial z}{\partial x} \tag{5.10}$$

$$\frac{d\mathbf{u}}{dy} = \frac{d\mathbf{u}(x, y, z(x, y))}{dy} = \frac{\partial \mathbf{u}}{\partial y} + \frac{\partial \mathbf{u}}{\partial z}\frac{\partial z}{\partial y} \tag{5.11}$$

$$\frac{d\mathbf{u}}{dz} = \frac{d\mathbf{u}(x, y, z(x, y))}{dz} = \frac{\partial \mathbf{u}}{\partial z} \tag{5.12}$$

In the discretization of the energy, we need to estimate the partial derivatives. Therefore, in the previous equation, we approximate the total derivatives by finite differences:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial x} = \dfrac{\Delta \mathbf{u}}{\Delta x} - \dfrac{\partial \mathbf{u}}{\partial z}\dfrac{\partial z}{\partial x} \\ \dfrac{\partial \mathbf{u}}{\partial y} = \dfrac{\Delta \mathbf{u}}{\Delta y} - \dfrac{\partial \mathbf{u}}{\partial z}\dfrac{\partial z}{\partial y} \\ \dfrac{\partial \mathbf{u}}{\partial z} = \dfrac{\Delta \mathbf{u}}{\Delta z} \end{cases} \tag{5.13}$$

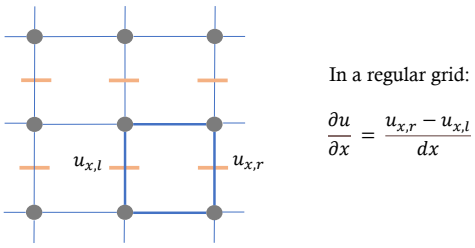


Figure 5.4: Difference computation in a 2D regular grid. The example here shown regards the tight blue difference on the bottom right.

where Δ represents the variations in the direction of the grid.

Finally, we compute derivatives as explained in Fig. 5.4 in a regular grid, and as shown in Fig. 5.5 in our deformed grid. We store z_i heights at the corners of the grid. These correspond to the heights of the corner points on top of the height map and are used to compute the $\Delta \mathbf{z}$ quantities.

Now that we described the pipeline for optimizing the velocity field, we can test its accuracy.

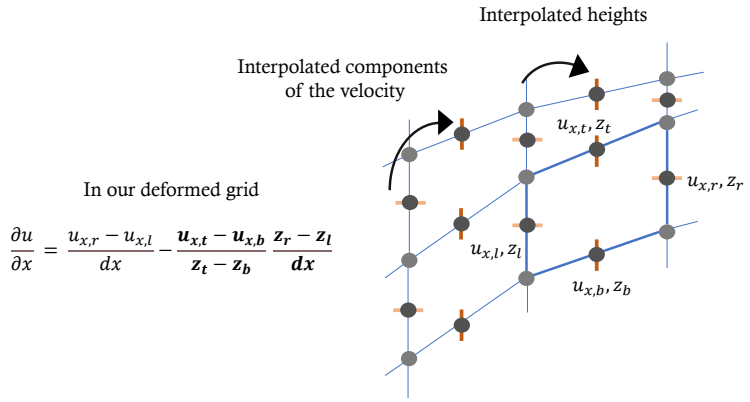


Figure 5.5: Difference computation in a 2D grid with vertical deformation. Here, we first need to interpolate the values of the x component stored on the vertical edges onto the horizontal edges, as well as the height values stored at the corners. Only then we are able to perform the differential operation. The example here shown regards the tight blue difference on the bottom right.

5.3 Results and discussion

5.3.1 Solver validation

We first validate our energy-based loss. For this, we remove the network to ensure that the loss we propose in Sec. 5.2.2 can be used to produce valid velocity fields. We compare against the well-studied problem of the lid-driven cavity to assess the quality of our formalism [Kuhlmann and Romanò, 2019].

We first consider a regular grid, with $dx = 1m$, and $dt = 1s$ within a cubic domain $\mathcal{D}$ of size $100m$ as shown in Fig. 5.6. We constrain the velocity at the top boundary to be equal to $\mathbf{u} = 1\frac{dx}{dt}$, and the velocity to be null on the other bounds of the domain $\mathbf{u} = 0$. We then sample regularly $100 \times 100 \times 100$ points and run our optimizer upon this domain. The velocities are stored in a staggered grid and the heights used to compute the $\Delta\mathbf{u}$ quantities in the corners of the grid. Starting from a null field, each value is adapted following a standard gradient descent scheme with the Adam optimizer.

We consider the cross section of our domain, as shown in Fig. 5.6, and visualize it by interpolating the values of the staggered grid in the centers of the cells, and obtain the field depicted in Fig. 5.7.

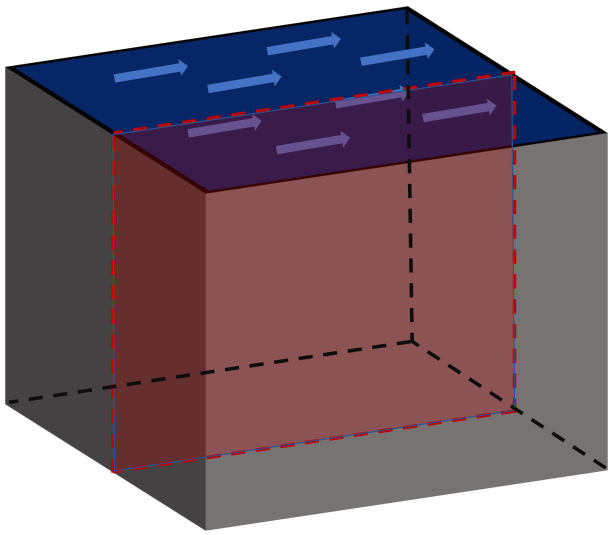


Figure 5.6: Visualization of the cavity, driven by a layer of fixed imposed velocities on top. The experiment is similar to filling a rectangular dish with water and applying a constant movement on one edge of it. We visualize in the next figures the red cross section in the middle, parallel to the imposed movement.

Deformed operators Then, we validate our differential operators on a deformed grid: Instead of solving the lid-driven cavity on a regular grid, we deform the original grid in the middle with a sinusoïd multiplied by a factor $\delta_{amplitude}$, and morph the in-between layers to obtain a new deformed grid as shown in Fig. 5.8.

We compare different deformations with the result obtained with the regular grid by plotting a green cross where the vortex is formed in this case. The field obtained with $\delta_{amplitude} = 0, 10, 20$ resemble the one computed one a regular grid. Only with the high distortion of $\delta_{amplitude} = 30$ the field starts diverging. This is due to cells being too vertical, in this case our approximations does not hold and we would need a more accurate meshing of the domain.

Equipped with this operational energy-based surrogate model, we now test the optimization process piped with a neural network mapping the height map directly to the velocity field.

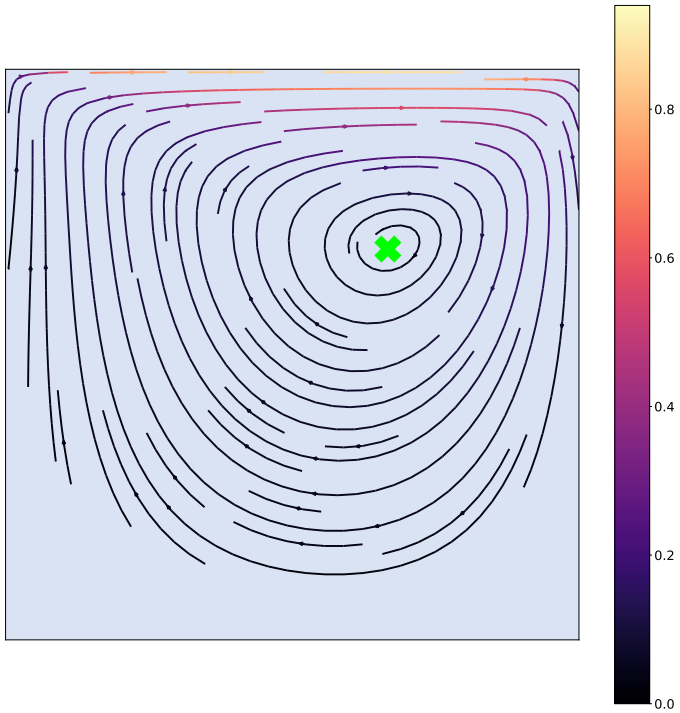


Figure 5.7: Air velocities produced by our solver within a lid-driven cavity on a regular grid. Note that we only draw the streamlines where the magnitude of the velocity is above $\epsilon = 10^{-2}$ to avoid unsignificant lines.

5.3.2 Learning-based optimization

We then test our fluid solver enhanced with the neural network, described in Sec. 5.2.1 and show its advantage compared to the standard optimization.

We consider a deformed grid, similar to Fig. 5.3, with $dx = 1m$, and $dt = 1s$, within a domain of size $256m \times 256m \times 60m$ and input a velocity of $\mathbf{u} = 1\frac{dx}{dt}$ as described in Sec. 5.2.2.

To assess for the advantage of our method, we consider a height map as input in the optimization framework. We again analyze the field by visualizing a cross section parallel to the input wind. We first optimize for the wind *without* neural network to get a reference field for comparison. For this, we let the solver optimize for 4000 iterations, and obtain the field in Fig. 5.10.

We then run two optimization procedures, with and without the neural network to compare their convergence performances. We assess their precision by computing the

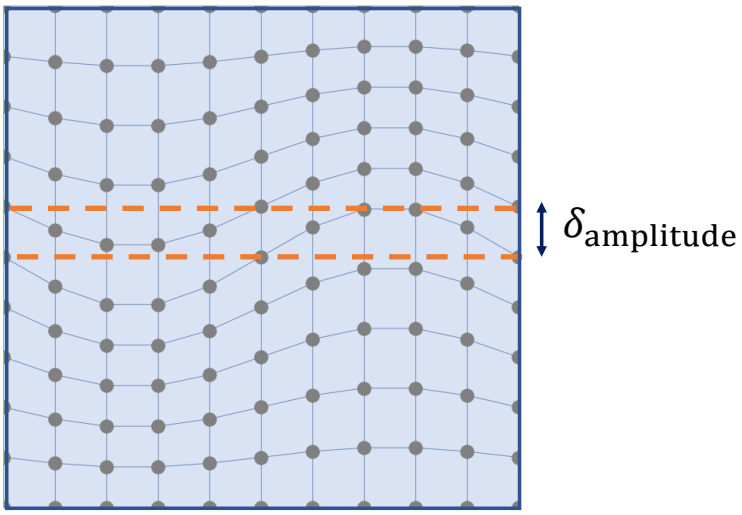


Figure 5.8: Cross section of the deformed grid for the experiment. Only $\frac{1}{10}$ cells are shown for clarity, the deformation factor is here $\delta_{amplitude} = 10$, corresponding to a grid being distorted with sinusoïd of amplitude $\delta_{amplitude} = 10$.

error to the reference, by computing the L_1 norm of the difference between the optimizing fields and the reference field. We use the Adam optimizer and empirically chose the learning rate for which the convergence to an error is below 1% is the fastest. We obtain the behavior shown in Fig. 5.12 for the two strategies. In this particular case, the CNN takes advantage of the self-similarity of the height map regions (almost all of them are flat), and achieves better convergence performance.

To better assess for the ability of the neural network to be used in this process, we shift the current height map by 10 cells in the direction of the wind to mimic the effect of erosion/deposition, and input this shifted height map to our network. We see in Fig. 5.11 that the neural network directly takes advantage of similarity of the two terrains to output a realistic wind field, while the previous wind field is more off.

This better guess for the initialization of the wind field above this new terrain is crucial, and leads to better performance when we re-optimize for the wind above the new height map with the two methods. We can see in Fig. 5.13 that, starting from a lower error, the network converges faster under the threshold error of the new terrain. Even if the error of the optimization without neural network seems quite low, it visually results in high

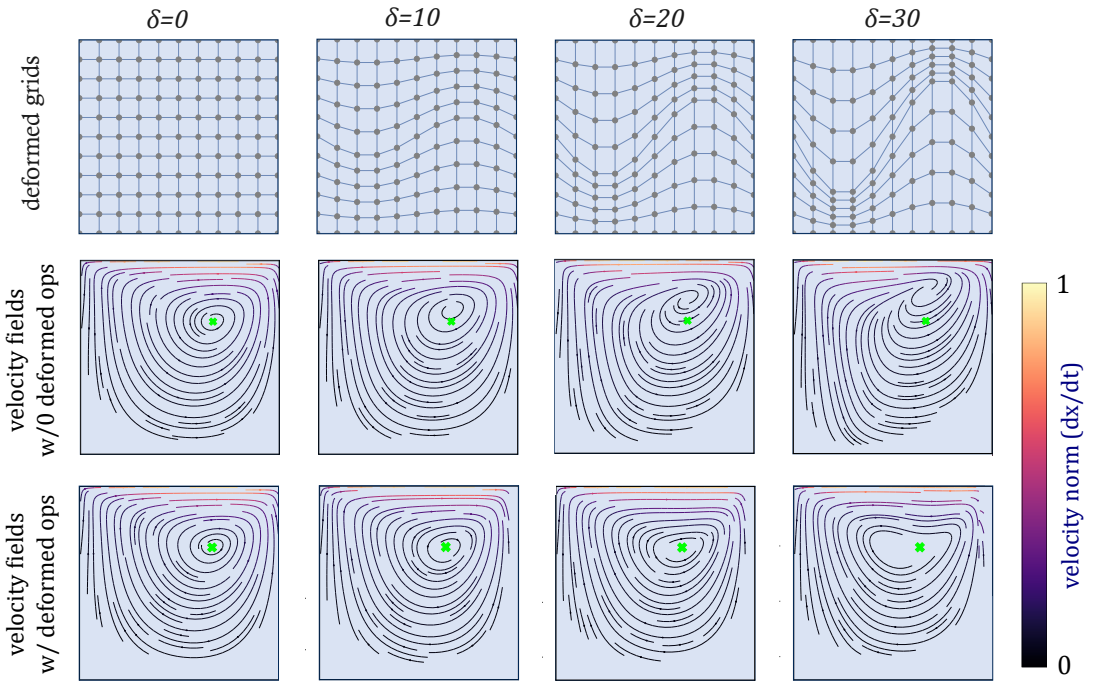


Figure 5.9: Velocity fields in the lid-driven cavity with different values of $\delta_{amplitude}$, and the corresponding deformed grid (top). We show the velocity field obtained without adapted deformed operators in the middle line, and with the adapted deformed operators in the bottom line. Here again, only $\frac{1}{10}$ cells are shown for clarity. The green cross pinpoints where the vortex forms for $\delta_{amplitude} = 0$, i.e. for a regular grid. Here again, we only draw the streamlines when the magnitude of the velocity is above $\epsilon = 10^{-2}$ to avoid insignificant lines.

difference, as we can see in Fig. 5.11.

Note here that in reality, from one frame to the other, the terrain will suffer fewer change, more around 1 or 2 cells difference, the re-optimization will therefore be quicker.

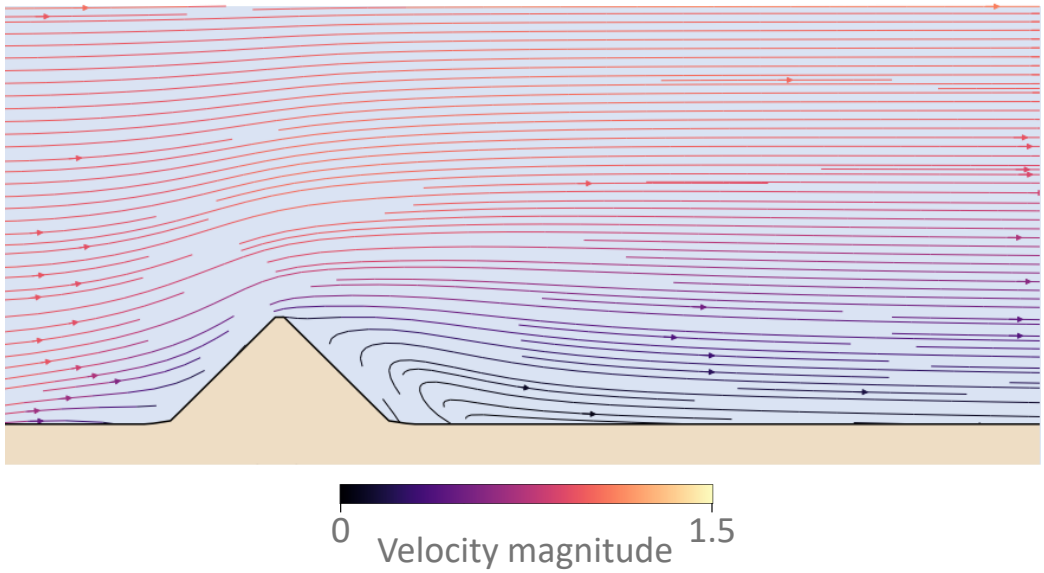


Figure 5.10: Cross section of the input terrain and the wind field after 4000 iterations of our solver, with $lr = 10^{-4}$. We use this field as reference for comparing our two optimization strategies.

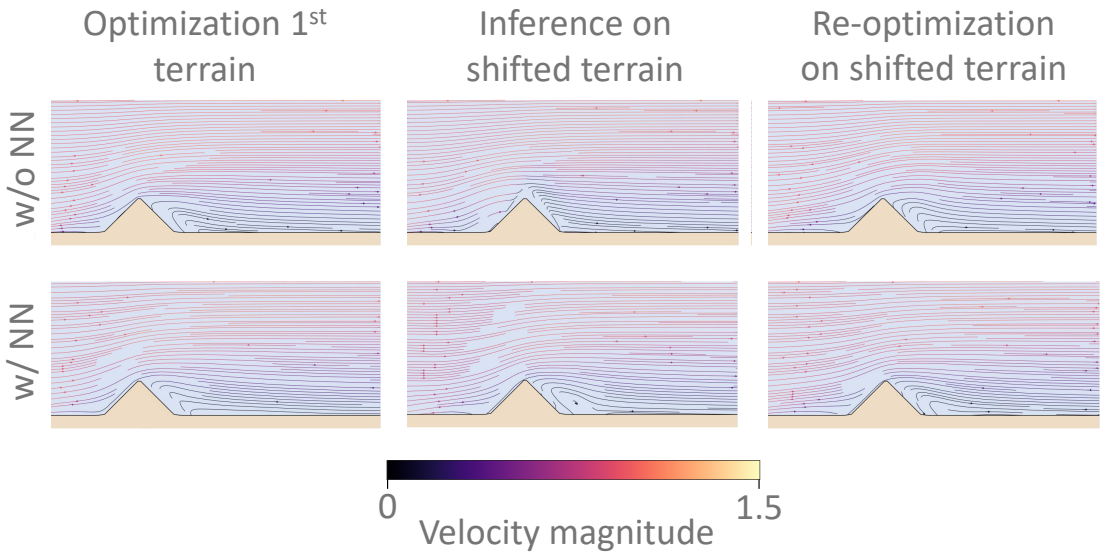


Figure 5.11: Comparison of the two strategies for re-optimizing the wind velocities. In both cases, with (bottom) and without (top) neural network, we optimize the velocities above a first terrain (left). We then shift this terrain (middle) and visualize the velocity field computed for the previous terrain (top) and the field produced by the inference of the neural surrogate model for the shifted terrain (bottom). We see that the neural network smoothly updates the wind field while the wind field produced by the first strategy is not accurate at the vicinity of the dune. This leads to a re-optimized wind field (right) that was quicker to compute with the neural network framework, as we show in Fig. 5.13. We visualize these wind fields with streamlines on a cross sections parallel to the input wind, and consider only the first half of the field for better visualization.

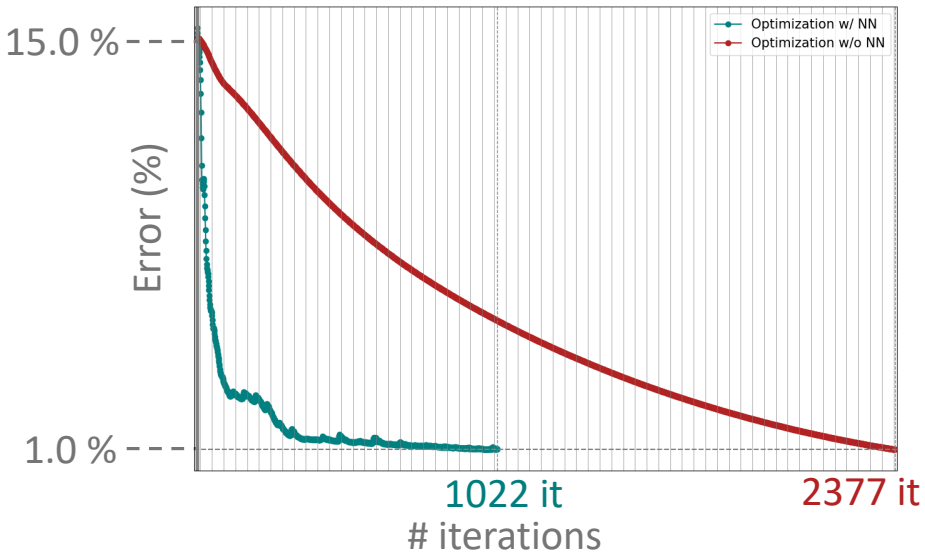


Figure 5.12: Optimization with both strategies on a first terrain.

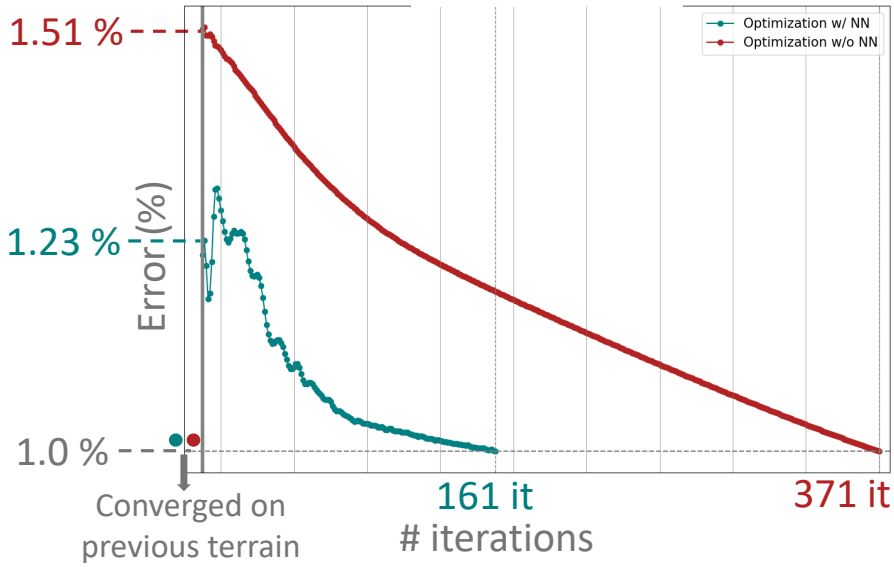


Figure 5.13: Optimization with both strategies on the shifted terrain. We first optimized both of them on the original terrain as described in Fig. 5.12, and re-started the optimization on the shifted terrain

5.4 Conclusion

We presented a self-supervised neural surrogate model that predicts, given a height map, the averaged wind flowing above it. It shows promising results in a reduced test case of an evolving terrain. A direct future work is integrating this optimization pipeline in a coupled simulation between wind prediction and sand transport, and confirm the gain in performance we measured in Sec. 5.3.2.

Moreover, so far, our pipeline does not enable for height maps with sharp discontinuities like buildings. However, they are of major importance for design applications such as Sand Mitigation Measures, where our differentiable neural surrogate could be used to optimize building placements.

Chapter 6

Conclusion and future work

In this thesis, we explored fast algorithms both to simulate interactions between fluid flows and obstacles and to optimize obstacles' shapes based on their impact on these fluid flows.
First, we presented a neural-based model to assist car designers in their exploratory phase. The system we propose gives aerodynamic feedback as soon as sketches are drawn, and additionally suggests shape modifications to optimize the vehicle movement in the air. Then, we developed an algorithm inspired by both Computer Graphics and geo-science to quickly model desertscapes, including scenes containing steep obstacles like human infrastrucures. We qualitatively validated our algorithm against real-world measurements. Finally, driven by the will of optimizing terrains to protect human facilities against sand deposition, we explored ways to invert this algorithm. To do so, we implemented and tested a self-supervised neural network for predicting the wind above a terrain, thus getting a primordial block for inverting the whole pipeline.

Bridging the gap between Design and Engineering

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 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.

We envision several future directions to further bridge the gap between design and engineering. One promising way for this is *dataset enrichment*. In the same vein than the idea proposed by [Durasov et al., 2021], a smart tool for assisting car designers could, in addition to fluid visualization and shape optimization, be refined by re-training the

model with optimized shapes *and* shapes proposed by designers. The predictions would therefore be more precise and could benefit from creative ideas from designers. The system we developed is a promising tool to explore this idea since it allows to directly take as input the designers' drawing. Data enrichment for car design could also be done by leveraging the recent rise of generative modeling. Following this idea, researchers developed methods based on generative artificial intelligence, trained on web-scale dataset. These methods can indeed generate models if they are physically-guided during generation [Arechiga et al., 2023] [Yuan et al., 2024].

However, bridging this gap between design and engineering will also need lifting car design to 3D. Indeed, 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 [GravitySketch, 2017], meshes, or point clouds. To be able to use our pipeline in 3D, one will also need to surface such 3D sketches to run simulations. Promising work have been carried out in this direction [Yu et al., 2022].

- 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. 4.2). We conjecture that adaptive sampling will yield even better performances in 3D where CNNs grow with cubic complexity. Works like [Kashefi et al., 2021] show promising results in this sense.

- 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.

Sand Mitigation Measures

We developed in Chapter 4 a fast and accurate simulation tool to model sand landscapes. It can be used in early stages of urban planning projects that involve sandy environments. Indeed, optimizing the wind flow in urban scenes to avoid high wind area or heat pockets is a on-going field of research [Liu et al., 2023]. Similarly, our method could be used to re-arrange buildings within small neighborhoods and give quick feedback on the areas most subjected to high sand deposition. One could therefore use it to protect structures., and avoid sand-related problems on infratructures [L. Raffaele, 2019].

Driven by the idea of automatically proposing engineering solutions to mitigate the impact of sand deposition on human infrastructures, we then developed a tool enabling the inversion of the whole sand transport pipeline. Integrating this tool in a complete sand transport simulation would help urban planners by automatically suggesting building re-arrangements. While we rely on a self-supervised method in Chapter 5 to invert wind prediction, future work could rely on our simulator to train fast data-driven surrogate models suitable for inverse design applications , like the one we present in Chapter 3.

However, to be fully integrated in an urban planning workflow, the simulation tool might have to be further validated. The tool we developed combines geo-science models and fast Computer Graphics wind approximations. Our experiments demonstrate that our method can reproduce reference patterns from alternative methods and field studies, and capture both the transient emergence of dunes and the static buildup of typical sand patterns around obstacles. However, getting more measurements is challenging due to the varying wind conditions and the difficulty to precisely measure them. Additionally, sand transport is a *dynamical process* and isolating wind conditions and height field measurement at a precise time is almost impossible. But carrying more precise and quantitative validation from real-world experiments could highlight flaws in our method and help us improve these.

Towards general wind-shaped landscape generation

Exhaustively and precisely evaluating our simulations against real-world experiments will probably lead to three main axes of enhancement for the method we proposed:

First, as described in Sec. 4.1, our method focuses on saltation only. However, sand can exhibit pattern because of particle being blown for long distances. Describing these patterns would require modeling *suspension* as well. However, this might come at a price in terms of performance, as, contrary to saltation and avalanche, suspension requires a 3D description to track the transport of particles. This could also enable us to model other types of particles more precisely: For example, dust, after being suspended in the air, can deposit on solar panels and impact energy production [Wang et al., 2017], and snow can cause disastrous damage in an urban environment [DeGaetano and Wilks, 1999].

Second, while also important in urban planning, snow particles also play an active role in shaping landscapes. The modeling of snow sceneries shares various common points with desertscape modeling [Goswami, 2024; Cordonnier et al., 2018a], as snow flakes can also be blown by the wind. Additional challenges for modelling snowscapes involve describing state changes based on a temperatures shifts: ice, one solidified, can no longer be transported like snow flakes. This specificity of snow leads to unique patterns like cornices, of which we can see an example in Fig. 6.1.

Third, describing patterns like cornices and overhangs in general also requires a more advanced description of terrain heights. One could for example rely on works that have been developed to model caves or arches, that can also have overhangs [Becher et al., 2017], [Paris et al., 2021], [Nilles et al., 2024]. Such methods could further help sand simulation as well, especially in the case of sand traps [Bruno et al., 2018], that our strictly 2D description of height fields prevents us from describing.

Multiscale Sand Simulation

Finally, as described in Sec. 2.3.4, sand patterns can be very varied. The formation of these different patterns usually depends on the scale considered. While star dunes can be hundreds of meters high and be shaped during hundreds of years, sand ripples are usually some centimeters high 6.2. Moreover, the grains' characteristics, the different wind conditions imply different predominant behaviors and modes of transport. While we demonstrated that our method can let emerge realistic sand dunes and deposition patterns for scene for some meters to tens of meters, we relied on an unique average behavior for sand grains. Investigating tools to allow for a multi-scale simulation of sand transport would to allow to automatically generate more complete desert sceneries. One could for example rely on a combination between our method and a simple mapping

Figure 6.1: A snow cornice. Such shapes appear when snow solidifies and the wind erodes the structure from below, thus producing overhangs.

between coarse wind field and the small-scale ripples. Motivated by a similar problem, [Pretorius et al., 2024] have developed a tool to generate smoke details within the air inside a volcano sky. They do this by procedural upsampling the smoke description given coarse representations of this smoke and the wind field. Drawing inspiration from this approach, one could produce ripple details on top of a sand bed given a coarse description of this sand and the wind field above it. This could be done using procedural or learning methods. This would finally close the gap between procedural generation of sand sceneries and simulation of sand deposition.

Figure 6.2: Sand ripples are typical features of sand bed that are usually some centimeters high. Their regularity and small size compared the sand dunes we model might motivated a specific treatment.

Bibliography

Jaffar HA Al-Zubaydi. Study of the engineering properties of dunes field at al-najaf governorate-middle of iraq. *Journal of Babylon University. Journal of Applied and Pure Sciences*, 25(3):1158–1172, 2017. 15

Jorge Alejandro Amador Herrera, Jonathan Klein, Daoming Liu, Wojtek Pałubicki, Sören Pirk, and Dominik L Michels. Cyclogenesis: Simulating hurricanes and tornadoes. *ACM Transactions on Graphics (TOG)*, 43(4):1–16, 2024. 17

Ashfaque Ansari and Rana Manoj Mourya. Drag force analysis of car by using low speed wind tunnel. *International Journal of Engineering Research and Reviews ISSN*, pages 144–149, 2014. 11

Nikos Arechiga, Frank Permenter, Binyang Song, and Chenyang Yuan. Drag-guided diffusion models for vehicle image generation. *arXiv preprint arXiv:2306.09935*, 2023. 84

Ralph Alger Bagnold. *The physics of blown sand and desert dunes*. Methuen & Co, London, 1941. 13

Kai Bai, Wei Li, Mathieu Desbrun, and Xiaopei Liu. Dynamic upsampling of smoke through dictionary-based learning. *ACM Transactions on Graphics (TOG)*, 40(1):1–19, 2020. 18

Kai Bai, Chunhao Wang, Mathieu Desbrun, and Xiaopei Liu. Predicting high-resolution turbulence details in space and time. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 40(6):1–16, 2021a. 23

Kai Bai, Chunhao Wang, Mathieu Desbrun, and Xiaopei Liu. Predicting high-resolution turbulence details in space and time. *ACM Transactions on Graphics (TOG)*, 40(6):1–16, 2021b. 18

Pierre Baque, Edoardo Remelli, Francois Fleuret, and Pascal Fua. Geodesic convolutional shape optimization. *International Conference on Machine Learning*, 2018. 5, 24

Mads Baungaard, Maarten Paul Van Der Laan, and Mark Kelly. Rans modeling of a single wind turbine wake in the unstable surface layer. *Wind Energy Science*, 7(2):783–800, 2022. 16

PW Bearman. Some observations on road vehicle wakes. *SAE Transactions*, pages 504–515, 1984. 11

Michael Becher, Michael Krone, Guido Reina, and Thomas Ertl. Feature-based volumetric terrain generation and decoration. *IEEE Transactions on Visualization and Computer Graphics*, 25(2):1283–1296, 2017. 86

Sascha Bellaire, Alec van Herwijnen, Christoph Mitterer, and Jürg Schweizer. On forecasting wet-snow avalanche activity using simulated snow cover data. *Cold Regions Science and Technology*, 144:28–38, 2017. 14

J Bender, A Kuijper, DW Fellner, and É Guérin. High-resolution simulation of granular material with sph. 2012. 19

Jan Bender and Dan Koschier. Divergence-free sph for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1193–1206, 2016. 16

DM Brienza, KC Chung, CE Brubaker, J Wang, TE Kang, and CT Lin. A system for the analysis of seat support surfaces using surface shape control and simultaneous measurement of applied pressures. *IEEE Transactions on Rehabilitation Engineering*, 4 (2):103–113, 1996. 21

Luca Bruno and Davide Fransos. Sand transverse dune aerodynamics: 3d coherent flow structures from a computational study. *Journal of Wind Engineering and Industrial Aerodynamics*, 147:291–301, 2015. 58

Luca Bruno, Nicolas Coste, Davide Fransos, Andrea Lo Giudice, Luigi Preziosi, and Lorenzo Raffaele. Shield for sand: an innovative barrier for windblown sand mitigation. *Recent Patents on Engineering*, 12(3):237–246, 2018. 63, 86

Howard P Buckley, Beckett Y Zhou, and David W Zingg. Airfoil optimization using practical aerodynamic design requirements. *Journal of Aircraft*, 47(5):1707–1719, 2010. 9

Francesco Castellani and Giordano Franceschini. Use of genetic algorithms as an innovative tool for race car design. *Racing Chassis and Suspension Design*, page 313, 2004. 22

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 35

Dengke Chen, Yang Liu, Huawei Chen, and Deyuan Zhang. Bio-inspired drag reduction surface from sharkskin. *Biosurface and Biotribology*, 4(2):39–45, 2018. 11

Li-Wei Chen, Berkay A. Cakal, Xiangyu Hu, and Nils Thuerey. Numerical investigation of minimum drag profiles in laminar flow using deep learning surrogates. *Journal of Fluid Mechanics*, 919:A34, 2021. doi: 10.1017/jfm.2021.398. 6, 24, 29, 32, 42

Mengyu Chu, Lingjie Liu, Quan Zheng, Erik Franz, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 41(4):119:1–119:14, aug 2022. 7, 23

Guillaume Cordonnier, Jean Braun, Marie-Paule Cani, Bedrich Benes, Eric Galin, Adrien Peytavie, and Éric Guérin. Large scale terrain generation from tectonic uplift and fluvial erosion. In *Computer Graphics Forum*, volume 35, pages 165–175. Wiley Online Library, 2016. 9

Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and M-P Cani. Interactive generation of time-evolving, snow-covered landscapes with avalanches. In *Computer Graphics Forum*, volume 37, pages 497–509. Wiley Online Library, 2018a. 86

Guillaume Cordonnier, Pierre Ecormier, Eric Galin, James Gain, Bedrich Benes, and Marie-Paule Cani. Interactive Generation of Time-evolving, Snow-Covered Landscapes with Avalanches. *Computer Graphics Forum*, 37(2):497–509, May 2018b. 53

Guillaume Cordonnier, Guillaume Jouvet, Adrien Peytavie, Jean Braun, Marie-Paule Cani, Bedrich Benes, Eric Galin, Eric Guérin, and James Gain. Forming terrains by glacial erosion. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023. 70

Sylvain Courrech du Pont. Dune morphodynamics. *Comptes Rendus Physique*, 16(1): 118–138, 2015. ISSN 1631-0705. 60

Jonathan Davies, Cheten Sharma, and Nicholas New. Large scale sand simulations on under the boardwalk. In *ACM SIGGRAPH 2024 Talks*, pages 1–2. 2024. 13

Arthur T DeGaetano and Daniel S Wilks. Mitigating snow-induced roof collapses using climate data and weather forecasts. *Meteorological Applications*, 6(4):301–312, 1999. 86

Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. A controllable method for animation of earth-scale clouds. *Proc. of CASA*, pages 43–52, 2006. 16

Hongchao Dun, Ning Huang, Jie Zhang, and Wei He. Effects of shape and rotation of sand particles in saltation. *Journal of Geophysical Research: Atmospheres*, 123(23):13–462, 2018. 19

Nikita Durasov, Artem Lukoyanov, Jonathan Donier, and Pascal Fua. Debosh: Deep bayesian shape optimization, 2021. 24, 83

David S Ebert. Procedural modeling, animation, and rendering of gases, fluids, and textures. *Course in Siggraph'95*, 1995. 17

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, 2017. 39, 40

Abdelrhman Fahmy, Salvador Domínguez-Bella, Javier Martínez-López, and Eduardo Molina-Piernas. Sand dune movement and flooding risk analysis for the pyramids of meroe, al bagrawiya archaeological site, sudan. *Heritage Science*, 11(1):136, 2023. 14

Seyyed Mojtaba Fakhari and Hatem Mrad. Aerodynamic shape optimization of naca airfoils based on a novel unconstrained conjugate gradient algorithm. *Journal of Engineering Research*, 2024. 23

Florian Ferstl, Ryoichi Ando, Chris Wojtan, Rüdiger Westermann, and Nils Thuerey. Narrow band flip for liquid simulations. In *Computer Graphics Forum*, volume 35, pages 225–232. Wiley Online Library, 2016. 17

Justin R Finn, Ming Li, and Sourabh V Apte. Particle based modelling and simulation of natural sand dynamics in the wave bottom boundary layer. *Journal of Fluid Mechanics*, 796:340–385, 2016. 18

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. *arXiv preprint arXiv:2210.00612*, 2022. 23

Daniel Fudge, David Zingg, and Robert Haimes. A cad-free and a cad-based geometry control system for aerodynamic shape optimization. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, page 451, 2005. 23

M. Sturm G. E. Liston. A snow-transport model for complex terrain. *Journal of Glaciology*, 44(148), 1998. 13, 19

P. Rauwoens G. Strypsteen, L.C. Van Rijn. Comparison of equilibrium sand transport rate model predictions with an extended dataset of field experiments at dry beaches with long fetch distance. *Aeolian Research*, 52, July 2021. 53, 54

Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. A review of digital terrain modeling. In *Computer Graphics Forum*, volume 38, pages 553–577. Wiley Online Library, 2019. 9

Ting Gao, Yaxing Wang, Yongjie Pang, and Jian Cao. Hull shape optimization for autonomous underwater vehicles using cfd. *Engineering applications of computational fluid mechanics*, 10(1):599–607, 2016. 9

Kjersti Gisnås. Wind simulations for use in local avalanche forecasting. 2020. 19

Prashant Goswami. Snow and ice animation methods in computer graphics. In *Computer Graphics Forum*, page e15059. Wiley Online Library, 2024. 86

A de Gracia, Nelson Rangel-Buitrago, Judith A Oakley, and AT Williams. Use of ecosystems in coastal erosion management. *Ocean & coastal management*, 156:277–289, 2018. 13

Francisco-Javier Granados-Ortiz, Pablo Morales-Higueras, Joaquín Ortega-Casanova, and Alejandro López-Martínez. Two-dimensional-based hybrid shape optimisation of a 5-element formula 1 race car front wing under fia regulations. *Machines*, 11(2):231, 2023. 22

GravitySketch. Gravity sketch. https://www.gravitysketch.com/, 2017. 84

Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2016. 6, 24, 42

Torsten Hädrich, Daniel T Banuti, Wojtek Pałubicki, Sören Pirk, and Dominik L Michels. Fire in paradise: Mesoscale simulation of wildfires. *ACM Transactions on Graphics (TOG)*, 40(4):1–15, 2021. 16

Francis H Harlow, J Eddie Welch, et al. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, 8(12):2182, 1965. 15

Marc Hartley, Nicolas Mellado, Christophe Fiorio, and Noura Faraj. Flexible terrain erosion. *The Visual Computer*, pages 1–15, 2024. 9

Jaroslav Haslinger and Raino AE Mäkinen. *Introduction to shape optimization: theory, approximation, and computation*. SIAM, 2003. 21

Ping He, Grzegorz Filip, Joaquim RRA Martins, and Kevin J Maki. Design optimization for self-propulsion of a bulk carrier hull using a discrete adjoint method. *Computers & Fluids*, 192:104259, 2019. 17

R. Himeno and K. Fujitani. Numerical analysis and visualization of flow in automobile aerodynamics development. In S. Murakami, editor, *Computational Wind Engineering 1*, pages 785–790. Elsevier, Oxford, 1993. 29

Philipp Holl, Nils Thuerey, and Vladlen Koltun. Learning to control pdes with differentiable physics. In *International Conference on Learning Representations*, 2020. 48

Libo Huang, Ziyin Qu, Xun Tan, Xinxin Zhang, Dominik L Michels, and Chenfanfu Jiang. Ships, splashes, and waves on a vast ocean. *ACM Transactions on Graphics (TOG)*, 40 (6):1–15, 2021. 9

Chris H Hugenholtz and Thomas E Barchyn. Real barchan dune collisions and ejections. *Geophysical Research Letters*, 39(2), 2012. 13

Xinghui Huo, Hongchao Dun, Ning Huang, and Jie Zhang. 3d direct numerical simulation on the emergence and development of aeolian sand ripples. *Frontiers in Physics*, 9: 662389, 2021. 19

Antony Jameson. Aerodynamic shape optimization using the adjoint method. *Lectures at the Von Karman Institute, Brussels*, 2003. 22

Byungwook Jang, Myungjun Kim, Jungsun Park, and Sooyong Lee. Design optimization of composite radar absorbing structures to improve stealth performance. *International Journal of Aeronautical and Space Sciences*, 17(1):20–28, 2016. 22

Steeven Janny, Aurélien Beneteau, Madiha Nadri, Julie Digne, Nicolas Thome, and Christian Wolf. Eagle: Large-scale learning of turbulent fluid dynamics with mesh transformers. *arXiv preprint arXiv:2302.10803*, 2023. 23

Eugéne C Joubert, Thomas M Harms, Annethea Muller, Martin Hipondoka, and Joh R Henschel. A cfd study of wind patterns over a desert dune and the effect on seed dispersion. *Environmental fluid mechanics*, 12:23–44, 2012. 19

Ali Kashefi, Davis Rempe, and Leonidas J Guibas. A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. *Physics of Fluids*, 33(2), 2021. 24, 84

Joseph Katz. Aerodynamics of race cars. *Annu. Rev. Fluid Mech.*, 38(1):27–63, 2006. 22

Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. Deep Fluids: A Generative Network for Parameterized Fluid Simulations. *Computer Graphics Forum (Proc. Eurographics)*, 38(2), 2019. 23

Jeong-Hyun Kim and Yong Oun Han. Experimental investigation of wake structure around an external rear view mirror of a passenger car. *Journal of Wind Engineering and Industrial Aerodynamics*, 99(12):1197–1206, 2011. 11

Jasper F Kok, Eric JR Parteli, Timothy I Michaels, and Diana Bou Karam. The physics of wind-blown sand and dust. *Reports on progress in Physics*, 75(10):106901, 2012. 7, 13, 53

Andrzej Kokosza, Helge Wrede, Daniel Gonzalez Esparza, Milosz Makowski, Daoming Liu, Dominik L Michels, Soren Pirk, and Wojtek Palubicki. Scintilla: Simulating combustible vegetation for wildfires. *ACM Transactions on Graphics (TOG)*, 43(4):1–21, 2024. 16

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. A survey on sph methods in computer graphics. In *Computer graphics forum*, volume 41, pages 737–760. Wiley Online Library, 2022. 16

Hendrik C Kuhlmann and Francesco Romanò. The lid-driven cavity. *Computational Modelling of Bifurcations and Instabilities in Fluid Dynamics*, pages 233–309, 2019. 16, 74

L. Bruno L. Raffaele. Windblown sand action on civil structures: Definition and probabilistic modelling. *Engineering Structures*, (148):88–101, 2019. 4, 14, 58, 85

Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. In *ACM SIGGRAPH 2005 Papers*, pages 777–786. 2005. 68

Jichao Li, Xiaosong Du, and Joaquim R.R.A. Martins. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*, 134, 2022. 6

Wei Li, Martin Z Bazant, and Juner Zhu. A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches. *Computer Methods in Applied Mechanics and Engineering*, 383:113933, 2021. 70

Glen E Liston and Matthew Sturm. A snow-transport model for complex terrain. *Journal of Glaciology*, 44(148):498–516, 1998. 14

Daoming Liu, Florian Rist, and Dominik L Michels. Urbanflow: Designing comfortable outdoor areas. In *2023 Annual Modeling and Simulation Conference (ANNSIM)*, pages 628–644. IEEE, 2023. 85

Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. Learning smooth neural functions via lipschitz regularization. In *ACM SIGGRAPH Conference Proceedings*, 2022. 39, 40

Tianshu Liu. Evolutionary understanding of airfoil lift. *Advances in Aerodynamics*, 3(1): 37, 2021. 10

A Lo Giudice, G Giammanco, D Fransos, and L Preziosi. Modeling sand slides by a mechanics-based degenerate parabolic equation. *Mathematics and Mechanics of Solids*, 24(8):2558–2575, 2019. 54

Andrea Lo Giudice and Luigi Preziosi. A fully eulerian multiphase model of windblown sand coupled with morphodynamic evolution: Erosion, transport, deposition, and avalanching. *Applied Mathematical Modelling*, 79:68–84, 2020. 6, 19, 20, 53, 54, 58, 59, 60, 63

Hanneke Luijting, Dagrun Vikhamar-Schuler, Trygve Aspelien, Åsmund Bakketun, and Mariken Homleid. Forcing the surfex/crocus snow model with combined hourly meteorological forecasts and gridded observations in southern norway. *The Cryosphere*, 12(6):2123–2145, 2018. 14

Jiaming Ma, Zhi Li, Zi-Long Zhao, and Yi Min Xie. Creating novel furniture through topology optimization and advanced manufacturing. *Rapid Prototyping Journal*, 27(9): 1749–1758, 2021. 21

Alison L Marsden, Meng Wang, Bijan Mohammadi, and P Moin. Shape optimization for aerodynamic noise control. *Center for Turbulence Research Annual Brief*, pages 241–47, 2001. 22

Mario J Martin, Esther Andres, Carlos Lozano, and Eusebio Valero. Volumetric b-splines shape parametrization for aerodynamic shape design. *Aerospace Science and technology*, 37:26–36, 2014. 23

Joaquim RRA Martins. Aerodynamic design optimization: Challenges and perspectives. *Computers & Fluids*, 239:105391, 2022. 9, 22

Edwin Dinwiddie McKee. *A study of global sand seas*, volume 1052. US Government Printing Office, 1979. 15

Nick Middleton, Peter Tozer, and Brenton Tozer. Sand and dust storms: underrated natural hazards. *Disasters*, 43(2):390–409, 2019. 13

Nick J Middleton. Desert dust hazards: A global review. *Aeolian research*, 24:53–63, 2017. 13

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 39

Seyedali Mirjalili and Seyedali Mirjalili. Genetic algorithm. *Evolutionary algorithms and neural networks: theory and applications*, pages 43–55, 2019. 22

Hemant Mittal, Ashutosh Sharma, and Ajay Gairola. A review on the study of urban wind at the pedestrian level around buildings. *Journal of Building Engineering*, 18:154–163, 2018. 9

Patrick Mullen, Keenan Crane, Dmitry Pavlov, Yiying Tong, and Mathieu Desbrun. Energy-preserving integrators for fluid animation. *ACM Transactions on Graphics (TOG)*, 28(3): 1–8, 2009. 16

Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159. Citeseer, 2003. 16

Frédérique Muyl, Laurent Dumas, and Vincent Herbert. Hybrid method for aerodynamic shape optimization in automotive industry. *Computers & Fluids*, 33(5):849–858, 2004. 35

Clément Narteau, Deguo Zhang, Olivier Rozier, and Philippe Claudin. Setting the length and time scales of a cellular automaton dune model from the analysis of superimposed bed forms. *Journal of Geophysical Research: Earth Surface*, 114(F3), 2009. 13

Claude Louis Marie Henri Navier. Mémoire sur les lois du mouvement des fluides. 1822. 15

Yen Ting Ng, Chohong Min, and Frédéric Gibou. An efficient fluid–solid coupling algorithm for single-phase flows. *Journal of Computational Physics*, 228(23):8807–8829, 2009. ISSN 0021-9991. 56

G. Nichols. *Sedimentology and Stratigraphy*. A John Wiley & Sons, Ltd., 2009. 12

Alexander Maximilian Nilles, Lars Günther, Tobias Wagner, and Stefan Müller. 3d real-time hydraulic erosion simulation using multi-layered heightmaps. 2024. 86

C. Narteau O. Rozier. A real-space cellular automaton laboratory. *Earth Surface Processes And Landforms*, 2013. 53

Norihiko Oda and Teruo Hoshino. Three-dimensional airflow visualization by smoke tunnel. *SAE Transactions*, pages 3187–3201, 1974. 11

Herbert Olivier, Thomas Reichel, and Mitja Zechner. Airfoil flow visualization and pressure measurements in high-reynolds-number transonic flow. *AIAA journal*, 41(8): 1405–1412, 2003. 17

OpenCFD. OpenFOAM - The Open Source CFD Toolbox. `http://www.openfoam.com`, 2007. 4, 15, 23

Carsten Othmer. Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, 4:1–23, 2014. 18

Derek Overby, Zeki Melek, and John Keyser. Interactive physically-based cloud simulation. In *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*, pages 469–470. IEEE, 2002. 16

Axel Paris, Adrien Peytavie, Eric Guérin, Oscar Argudo, and Eric Galin. Desertscape simulation. In *Computer Graphics Forum*, volume 38, pages 47–55. Wiley Online Library, 2019. 20, 53, 60

Axel Paris, Eric Guérin, Adrien Peytavie, Pauline Collon, and Eric Galin. Synthesizing geologically coherent cave networks. In *Computer Graphics Forum*, volume 40, pages 277–287. Wiley Online Library, 2021. 86

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6, 39

RJ Pattenden, SR Turnock, and Xin Zhang. Measurements of the flow over a low-aspect-ratio cylinder mounted on a ground plane. *Experiments in Fluids*, 39:10–21, 2005. 12

Frank Perbet and Maric-Paule Cani. Animating prairies in real-time. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 103–110, 2001. 17

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020. 23

Olivier Pironneau. On optimum design in fluid mechanics. *Journal of fluid mechanics*, 64 (1):97–110, 1974. 22

Daan W Poppema, Kathelijne M Wijnberg, Jan PM Mulder, and Suzanne JMH Hulscher. Deposition patterns around buildings at the beach: Effects of building spacing and orientation. *Geomorphology*, 401:108114, 2022. 12, 60, 61, 62

Durusoju Hari Prasad and Nandyala Darga Kumar. Coastal erosion studies—a review. *International Journal of Geosciences*, 2014, 2014. 13

Pieter C Pretorius, James Gain, Maud Lastic, Guillaume Cordonnier, Jiong Chen, Damien Rohmer, and M-P Cani. Volcanic skies: coupling explosive eruptions with atmospheric simulation to create consistent skyscapes. In *Computer Graphics Forum*, volume 43, page e15034. Wiley Online Library, 2024. 87

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in neural information processing systems*, 29, 2016. 23

Louis Quéno, Rebecca Mott, Paul Morin, Bertrand Cluzet, Giulia Mazzotti, and Tobias Jonas. Snow redistribution in an intermediate-complexity snow hydrology modelling framework. *EGUsphere*, 2023:1–32, 2023. 14

Lorenzo Raffaele and Luca Bruno. Windblown sand action on civil structures: Definition and probabilistic modelling. *Engineering Structures*, 178:88–101, 2019. 14

Lorenzo Raffaele, Nicolas Coste, and Gertjan Glabeke. Life-cycle performance and cost analysis of sand mitigation measures: Toward a hybrid experimental-computational approach. *Journal of Structural Engineering*, 148(7):04022082, 2022. 64

Benyounes Raillani, Mourad Salhi, Dounia Chaatouf, Samir Amraqui, and Ahmed Mezrhab. Optimization of a porous wind barrier to reduce soiling and avoid shading losses of photovoltaic panels. *Renewable Energy*, 189:510–523, 2022. 9

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019. 7, 23, 48

Lyle Regenwetter, Amin Heyrani Nobari, and Faez Ahmed. Deep Generative Models in Engineering Design: A Review. *Journal of Mechanical Design*, 144(7), 03 2022. ISSN 1050-0472. doi: 10.1115/1.4053859. 6

James Reuther, Antony Jameson, James Farmer, Luigi Martinelli, and David Saunders. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. In *34th aerospace sciences meeting and exhibit*, page 94, 1996. 22

John F Rooney Jr. The urban snow hazard in the united states. In *A Geography of Urban Places*, pages 439–457. Routledge, 2014. 13

Bruno Roy, Pierre Poulin, and Eric Paquette. Neural upflow: A scene flow learning approach to increase the apparent resolution of particle-based liquids. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–26, 2021. 18

Christopher L Rumsey and Susan X Ying. Prediction of high lift: review of present cfd capability. *Progress in Aerospace Sciences*, 38(2):145–180, 2002. 15

Simon Schneiderbauer and Alexander Prokop. The atmospheric snow-transport model: Snowdrift3d. *Journal of Glaciology*, 57(203):526–542, 2011. 13

Simon Schneiderbauer, Thomas Tschachler, Johann Fischbacher, Walter Hinterberger, and Peter Fischer. Computational fluid dynamic (cfd) simulation of snowdrift in alpine environments, including a local weather model, for operational avalanche warning. *Annals of glaciology*, 48:150–158, 2008. 13

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020. 6

Shaun N Skinner and Hossein Zare-Behtash. State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing*, 62:933–962, 2018. 22

Jos Stam. Stable fluids. In *Proc. Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, 1999a. 37

Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, page 121–128, USA, 1999b. ACM Press/Addison-Wesley Publishing Co. ISBN 0201485605. 7, 16, 18, 56

George Gabriel Stokes. On the theories of the internal friction of fluids in motion. 1845. 15

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32 (4):1–10, 2013. 19

G Strypsteen, LC Van Rijn, and P Rauwoens. Comparison of equilibrium sand transport rate model predictions with an extended dataset of field experiments at dry beaches with long fetch distance. *Aeolian Research*, 52:100725, 2021. 19

Tetsuya Takahashi and Christopher Batty. Monolith: a monolithic pressure-viscosity-contact solver for strong two-way rigid-rigid rigid-fluid coupling. 2020. 71

Andre Pradhana Tampubolon, Theodore Gast, Gergely Klár, Chuyuan Fu, Joseph Teran, Chenfanfu Jiang, and Ken Museth. Multi-species simulation of porous sand and water mixtures. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017. 19

Jie Tan and XuBo Yang. Physically-based fluid animation: A survey. *Science in China Series F: Information Sciences*, 52(5):723–740, 2009. 9

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 39

Brennen Taylor and John Keyser. Real-time sand dune simulation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 6, 20, 53, 60, 62

Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020. doi: 10.2514/1.J058291. 6, 24, 42

Yoshihide Tominaga and Ted Stathopoulos. Cfd simulation of near-field pollutant dispersion in the urban environment: A review of current modeling techniques. *Atmospheric environment*, 79:716–730, 2013. 15

Yoshihide Tominaga, Tsubasa Okaze, and Akashi Mochida. Wind tunnel experiment and cfd analysis of sand erosion/deposition due to wind around an obstacle. *Journal of Wind Engineering and Industrial Aerodynamics*, 182:262–271, 2018. 12, 20

Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning*, pages 3424–3433. PMLR, 2017. 16, 23

Nobuyuki Umetani and Bernd Bickel. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4), 2018. 5, 17, 18, 23, 29

Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt, and Takeo Igarashi. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 33(4), 2014. 10

Alexandre Valance, Keld Rømer Rasmussen, Ahmed Ould El Moctar, and Pascal Dupont. The physics of aeolian sand transport. *Comptes Rendus. Physique*, 16(1):105–117, 2015. 13

Ulysse Vimont, James Gain, Maud Lastic, Guillaume Cordonnier, Babatunde Abiodun, and Marie-Paule Cani. Interactive meso-scale simulation of skyscapes. In *Computer Graphics Forum*, volume 39, pages 585–596. Wiley Online Library, 2020. 17

V Vionnet, E Martin, V Masson, G Guyomarc'h, F Naaim-Bouvet, A Prokop, Y Durand, and C Lac. Simulation of wind-induced snow transport and sublimation in alpine terrain using a fully coupled snowpack/atmosphere model. *The Cryosphere*, 8(2):395–415, 2014. 13

Castelli Vittorio Vlachos Michail, Yu Philip. 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. 37

Nils Wandel, Michael Weinmann, and Reinhard Klein. Learning incompressible fluid dynamics from scratch - towards fast, differentiable fluid models that generalize. International Conference on Learning Representations, 2021. 23

Jingshu Wang, Hengxiang Gong, and Zheng Zou. Modeling of dust deposition affecting transmittance of pv modules. *J. Clean Energy Technol*, 5(3):217–221, 2017. 86

Yong Wang, Jie Zhang, Hongchao Dun, and Ning Huang. Numerical investigation on impact erosion of aeolian sand saltation in gobi. *Atmosphere*, 14(2):349, 2023. 13

Zhengshi Wang, Shan Ren, and Ning Huang. Saltation of non-spherical sand particles. *Plos one*, 9(8):e105208, 2014. 19

Klaus Wassermann. Three-dimensional shape optimization of arch dams with prescribed shape functions. *Journal of Structural mechanics*, 11(4):465–489, 1983. 21

Antoine Webanck, Yann Cortial, Eric Guérin, and Eric Galin. Procedural cloudscapes. In *Computer Graphics Forum*, volume 37, pages 431–442. Wiley Online Library, 2018. 17

Steffen Wiewel, Moritz Becher, and Nils Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pages 71–82. Wiley Online Library, 2019. 23

Allan T Williams, Nelson Rangel-Buitrago, Enzo Pranzini, and Giorgio Anfuso. The management of coastal erosion. *Ocean & coastal management*, 156:4–20, 2018. 13

Cheng Xiao-liang and Abdul Wasim Shaikh. Analysis of the iterative penalty method for the stokes equations. *Applied mathematics letters*, 19(10):1024–1028, 2006. 71

You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018. 18

Wang Xudong, Wen Zhong Shen, Wei Jun Zhu, Jens Nørkær Sørensen, and Chen Jin. Shape optimization of wind turbine blades. *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, 12(8):781–803, 2009. 9

He Yan, Zhangye Wang, Jian He, Xi Chen, Changbo Wang, and Qunsheng Peng. Real-time fluid simulation with adaptive sph. *Computer Animation and Virtual Worlds*, 20(2-3): 417–426, 2009. 16

Satoru Yoshida and Tomoyuki Nishita. Modelling of smoke flow taking obstacles into account. In *Proceedings the Eighth Pacific Conference on Computer Graphics and Applications*, pages 135–443. IEEE, 2000. 9

Andy Yu and Gladimir VG Baranoski. A study on sand ripple visualization. 2024. 19

Emilie Yu, Rahul Arora, J Andreas Baerentzen, Karan Singh, and Adrien Bousseau. Piecewise-smooth surface fitting onto unstructured 3d sketches. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022. 84

Chenyang Yuan, Frank Permenter, Nikos Arechiga, and Faez Ahmed. Data-driven car drag prediction with depth and normal renderings. *Journal of Mechanical Design*, 146: 051714–1, 2024. 84

Andrea Zampiron, Pablo Ouro, Stuart M Cameron, Thorsten Stoesser, and Vladimir Nikora. Conservation equations for open-channel flow: effects of bed roughness and secondary currents. *Environmental Fluid Mechanics*, pages 1–29, 2024. 19

Deguo Zhang, Clément Narteau, Olivier Rozier, and Sylvain Courrech du Pont. Morphology and dynamics of star dunes from numerical modelling. *Nature Geoscience*, 5(7): 463–467, 2012. 13, 59, 60

Ke-cun Zhang, Jian-jun Qu, Kong-tai Liao, Qing-he Niu, and Qing-jie Han. Damage by wind-blown sand and its control along qinghai-tibet railway in china. *Aeolian Research*, 1(3-4):143–146, 2010. 14

Xiao-Hu Zhao, Manousos Valyrakis, Thomas Pähtz, and Zhen-Shan Li. The role of coherent airflow structures on the incipient aeolian entrainment of coarse particles. *Journal of Geophysical Research: Earth Surface*, 129(5):e2023JF007420, 2024. 12

Xuanyi Zhou and Tiange Zhang. A review of computational fluid dynamics simulations of wind-induced snow drifting around obstacles. *Journal of Wind Engineering and Industrial Aerodynamics*, 234:105350, 2023. 19

Bo Zhu, Michiaki Iwata, Ryo Haraguchi, Takashi Ashihara, Nobuyuki Umetani, Takeo Igarashi, and Kazuo Nakazawa. Sketch-based dynamic illustration of fluid systems. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 30(6), 2011. 28

Yongning Zhu and Robert Bridson. Animating sand as a fluid. *ACM Transactions on Graphics (TOG)*, 24(3):965–972, 2005. 19, 20

AW Zingg. Wind tunnel studies of the movement of sedimentary material. In *Proc. 5th Hydraulics Conf., IAHR*, pages 111–135, 1952. 54

Xue-Yong Zou, Hong Cheng, Chun-Lai Zhang, and Yan-Zhi Zhao. Effects of the magnus and saffman forces on the saltation trajectories of sand grain. *Geomorphology*, 90(1):11–22, 2007. ISSN 0169-555X. doi: https://doi.org/10.1016/j.geomorph.2007.01.006. URL https://www.sciencedirect.com/science/article/pii/S0169555X07000244. 19